Report - 5.01.2003

By Arie Livshin

The following commentary addresses the classification program sources for musical instrument samples, which I was assigned to improve.

Bugs and Problems

1. Discriminant Analysis ("DA").
The software contains flags that specify whether to use DA in order to find optimal descriptors for classification, or to classify using all the descriptors.
It turns out that when the program is instructed to use DA, it does compute it, but later it ignores these results and does a classification using all the descriptors.
After I modified the program to use the DA results, the results of the classification were much worse than in the paper describing it - the classification into "musical instruments" produced only 66% success, compared to 82% success when the program used all the descriptors.
As the success percentages in the paper were quite similar in the cases where DA was used and when it was not, I suspected that the bug might have been in the program already during the writing of the article.
There may have been some modifications to the code before it was given to me and the bug might have been introduced at that stage (after the paper was written).

Ways to deal with the problem:
- There may exist a different version of the source.
- Trying a different module for DA calculations.

2. Mutual Information.
The paper compared descriptor selection using LDA, with a method using mutual information.
The mutual information module is missing. I understand it is gone as it did not produce optimal results and it is hard to locate all its sources.
BTW - Bertrand said that in his program mutual information works better than DA, so I think it could still be interesting to experiment with it.

Ways to deal with the problem:
- Trying a different module for Mutual Information.

3. Manual choice of descriptors.
The program has a flag which causes it not to use certain specific descriptors.
I was told that the reason for excluding these descriptors is that they're irrelevant for the classification, for example - the filename of the sample, the descriptors that perform calculations on energy and "f0"[1].

[1] I don't really understand why "f0" and descriptors that use the (normalized) energy are irrelevant. Regarding f0 - the clarinet, for example, has a different timbre at its lower and higher registers, so f0 could be relevant. Regarding energy - as the program misclassifies some obviously sustained signals as pizzicato, an "energy centroid" descriptor, for example, could help.

To remove these "irrelevant" descriptors, the program directly uses their indexes in the descriptor matrix, and has remarks ('%') that explain which descriptors are removed.
These indexes are wrong and do not point to the correct descriptors that appear in the remarks. Some of these indexes even point outside the descriptor matrix. Also, some of the descriptors in the remarks don't appear anywhere in the matrix (even in different indexes).

Ways to deal with the problem:
- To check exactly where descriptors specified by the remarks really appear in the descriptor matrix, and fix their indexes.
- I also think that it might be worth to reconsider whether all these descriptors are really worthless and need to be removed.

BTW:
When the program classifies using <u>all the descriptors</u> (without the above elimination), the success percentage is higher.
"Musical instruments": success rose from 82.27% to 89.94%.
"Instrument families": 89.77% -> 94.47%.
"Pizzicato" (worse): 99.47 -> 99.23%.
It is important to remember that some of the descriptors still might be "unfair" and should be eliminated.

4. The samples.
Some of the samples seem problematic[2].
- I wrote a program which classifies a sample as "pizzicato" if it has an energy peak at the beginning, followed by an almost constant slope down. My program had problems with the lowest samples of the "harp", which also made problems for the original classification program.
After listening to these samples and looking at their graphs, I think that they don't sound nor look like a pizzicato, and frankly doubt whether these are real harp sounds at all.
- There is a sample of an accordion which is 4 times longer than all the other samples in the set. This specific sample is also misclassified regularly (e.g. as a pizzicato) by the classification program.

Ways to deal with the problem:
To remove or modify the problematic samples and to check whether it improves the classification. It is also possible that when "bad" samples are learned, they actually hurt the classification of other "better" samples.

5. Success percentage.
The calculation of the success percentage was wrong:
```
a = diff([database_s.class] - found_class_v);
percentage=1 - (length(find(diff(a) > 0))/length(a))
```

[2] I believe it could be important which samples are learned. Randomly picking several samples from a list, performing DA by them, then learning them and classifying the other samples, might cause over-fitting or under-fitting if "bad" samples are chosen.

The diff command with a single parameter uses subtraction of subsequent cells in a vector. Using this calculation resulted, for example, in different success percentage according to the position of the misclassified samples in the sample list.

Dealing with the problem:
The calculation was changed to -

```
percentage=1-(length(find([database_s.class]-found_class_v)))/length(found_class_v);
```

It is interesting to note that now the classification into musical instruments reports higher success percentage, which rose from 82% to 85%.

6. Descriptors.
When I used Neural Networks for the classification, I received a warning that some of the inputs are constant, meaning that some descriptors produce the same values for all the samples. This raises the question whether all the descriptors are correctly calculated?

Novelties and additions:

1. I wrote a friendly GUI for the classification program, which:
- Allows specification of the desired classification type.
- Produces a list of the misclassified samples and how they were classified, along with a separate list of the classification of all the samples. It is easy (by pressing a push-button) to move between the same sample in both of the lists in order to compare differently classified samples from the same family.

There are push-buttons to copy these lists into the clipboard in order to paste them into other programs, and save them.
- The GUI shows the energy, fft and spectrogram graphs of the samples, with zooming options.
- The samples could be listened to.
- The GUI could automatically run a specified number of classification cycles, producing a list of all the misclassified samples and the average success percentage for all the classifications. This option is useful because the classification program randomly selects 2/3 of the samples for learning, and classifies the remaining 1/3.

2. I added a classification module which classifies using LVQ (Learning Vector Quantization) neural networks instead of Gaussian classification.
This experiment was not very successful:
In the "pizzicato/sustain" case the LVQ network gave 99.1% success percentage, a little lower than the Gaussian. The "musical instruments" classification produced only 60% success - much worse than the Gaussian 85%.
I believe that tweaking the parameters of the net as well as some extra tuning of the data will help to reach higher results.
It is also interesting to try other neural network types (e.g. Back-Propagation), as well as the K-Nearest Neighbors algorithm (which BTW worked for Bertrand better than the Gaussian classification).

3. All the components required by the classification program are now working under Microsoft-Windows (thanks to Axel!) - "additive", "sdif" and others.

General remark:
I still believe that hierarchical classification with specifically selected descriptors for each node in the classification tree (by using automatic techniques like DA, and also our knowledge of the exact role of each node) could help to focus the classification and improve the results.
BTW - Bertrand uses a similar technique in his program.

27/11/07 :
Slim Essid did exactly this and published it in a journal:
Essid, S., Richard, G., David, B. 2006: Musical instrument recognition by pairwise. classification strategies. IEEE Trans. on Speech and Audio Processing.

Perhaps a pity I didn’t persue this suggestion from 1.2003?

Automatic Classification of Musical Instrument Samples

By Arie Livshin, Xavier Rodet

(preliminary draft 0.1)

The work in this report is a continuation to the work of Peeters, Rodet, in "Automatically selecting signal descriptors for Sound Classification".

Abstract

This report describes a process of automatic classification of musical sounds and compares the results of three different classification algorithms:
Multi-dimensional Gauss classifier, K-nearest neighbors and Learning Vector Quantization neural networks.

Introduction

…
…

The Data

In this work, we used 1325 sound samples from Ircam's Studio On Line database. These monophonic samples are sampled in 44.1 KHz with 16 bit resolution.

The sound samples are categorized according to three different taxonomies:
Pizzicato / Sostenuto.
Musical instrument Families: Flute/Reeds, Brass, Bowed Strings and Plucked Strings.
Musical Instruments: Flute, Oboe, Clarinet (Bb), Bassoon (French), Alto Saxophone (Eb), French Horn (F), Trumpet, Trombone (Tenor/Bass), Tuba (Bass F), Guitar, Harp, Violin, Viola, Cello, Contrabass (5 str.) and Accordion.

For the task of automatic classification, 162 different sound descriptors were calculated for each sound sample. The descriptors are from the following types:
Temporal descriptors, Energy Descriptors, Spectral Descriptors, Harmonic descriptors and Perceptual descriptors [see "descriptors taxonomy" in Peeters et al] .

Data Preprocessing

In order to evaluate our classification process, we randomly select two thirds of the sound samples from each class of the desired taxonomy - these sounds will be called "The Learned Group". This group will be used to classify the remaining samples - "The Classified Group".
To get better separation between the classes, we apply Linear Discriminant Analysis to the descriptor matrix of the Learned Group, resulting in a new matrix which is a linear combination of the original descriptors. The same linear combination is also applied to the descriptor matrix of the classified group.

For convenience, I shall continue calling the space-transformed descriptors - "descriptors".

Learning Stage

Each classification algorithm will learn the Learned Group in order to classify the Classified Group.
Learning Vector Quantization - The neural net is trained on the descriptor vectors of the Learned Group until a satisfactory mean square error is reached.
Gaussian classifier - Each class is modeled by a multi-dimensional Gaussian function.
K-Nearest Neighbors - The best K is estimated by using the "Leave-one-out" method on the Learned Group.

Classification

After the learning process, the classification algorithms are ready for classification of the Classified Group.
Learning Vector Quantization **-** The neural network is activated on each descriptor vector from the Classified Group and produces its classification results.
Gaussian classifier **-** The probability of every descriptor vector from the Learned Group to belong to each classification class is estimated by a Bayesian function.
K-Nearest Neighbors **-** The Euclidian distance of each descriptor vector in the Classified Group from all the vectors in the Learned Group is computed, and the classes of the closest K "learned" vectors determine the classification.

Results

The following table shows the average success percentage over 20 classification experiments with each classification algorithm.

Algorithm ➔ ... ➔ Taxonomy	**Gauss**	**KNN** ✓	**LVQ**
Pizzicato/Sostenuto	99.58%	[*]99.97%	99.93%
Instrument Family	94.91%	95.24[†]%	95.22%
Instruments	94.12%	95.85%	88.57%

[*] Without LDA space-transformation we get 100% success in "Pizzicato/Sostenuto" classification using KNN! Never the less, in all other classifications space-transformation improves the results. This is why, for the sake of uniformity, I preferred that all classification algorithms will use the same data.
[†] When classifying musical samples from the S.O.L. database using KNN or Gauss, the results of classifying using the "musical instrument family" taxonomy are not better than classification into "musical instruments". Yet, when classifying in "harder" conditions, like using the S.O.L. database to classify another database (e.g. IOWA), or using less suitable algorithms like LVQ or Probabilistic Neural Networks, the difficulty level hierarchy of the classification taxonomies is much more evident - starting from Pizzicato/Sostenuto, then the Instrument Families and the hardest - Musical Instruments.

Conclusions

...
…

References

Peeters, G. and X. Rodet. (2002). "Automatic Classification of Musical Instrument Samples".
…
…

Report - 3.02.2003

By Arie Livshin

Automatic Descriptor Selection Using Discriminant Analysis

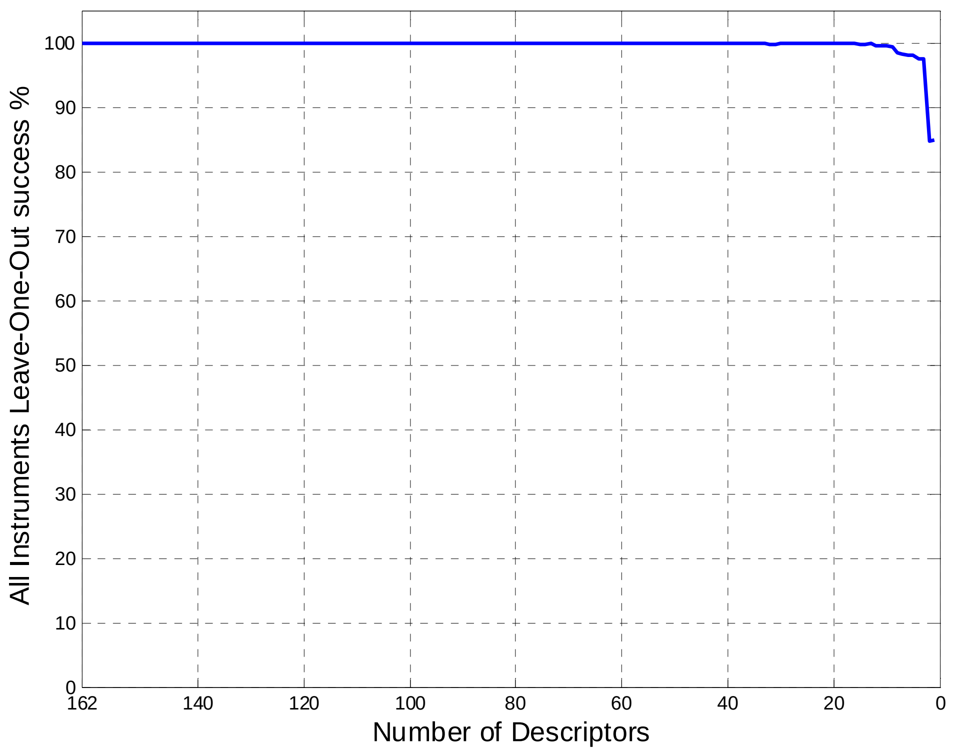
I have used DA (Discriminant Analysis) to calculate the linear combinations of descriptors that produce better class separation (and reduce the dimensionality of the descriptor matrix). By examining these linear combinations, we see which descriptors are multiplied by the biggest coefficients, which means these descriptors are the most "important" ones for the classification.
As DA produces a coefficient matrix with several dimensions, I have summed the coefficients of every descriptor over all the dimensions used (more details on the process later, in the "Differences between my program and the original one when performing descriptor selection with Discriminant Analysis" section).

At each round, after receiving this weight vector, I have disposed of the descriptor with the lowest weight - the less important one, and checked how this affects the classification. Each time I have saved the list of the remaining descriptors and the classification results. I keep files with all the appropriate descriptor lists and classification results starting from a classification with all the descriptors down to using a single descriptor.

The classification results were measured by using the leave-one-out method, checking the classification of every instrument in the descriptor matrix of all the samples and adding up the number of misclassified instruments.
In this way, the results do not depend on any random selection of a "learned" and a "classified" group or their sizes, and needs to be measured only once.
This process gives us the relation between the number of descriptors and the classification results. Naturally, the result of each "specific" classification (with specific "learned" and "classified" groups) will be different.

The "leave-one-out" classification used the KNN algorithm, for which an ideal K was calculated.

Pizzicato / Sustain Classification



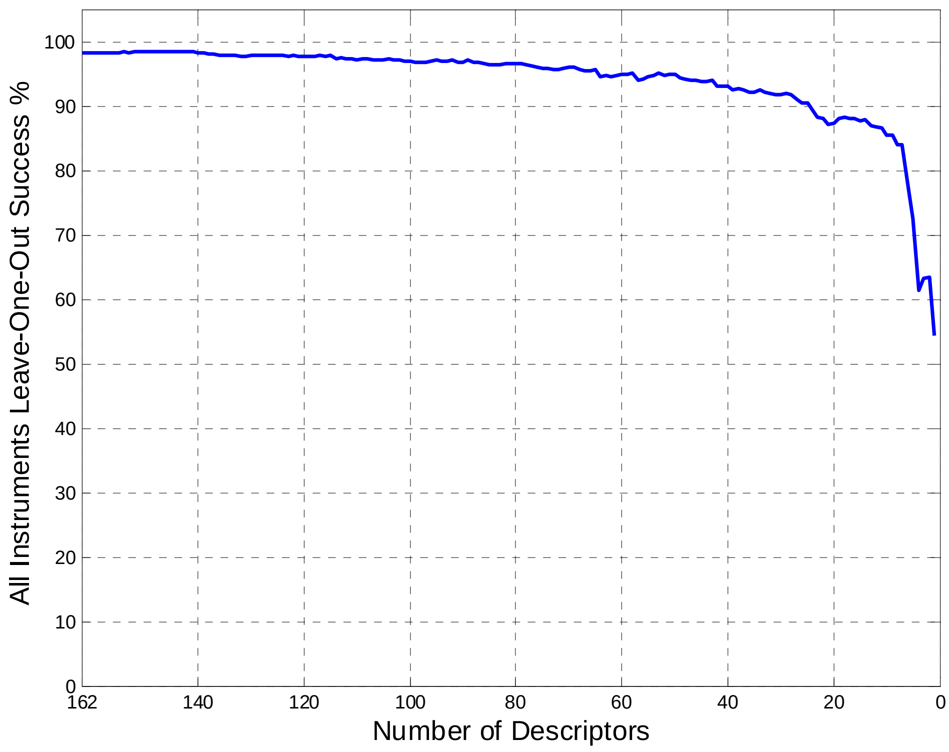
- The highest result in this classification[1] taxonomy (I shall mark the results "LOO" - "Leave-One-Out", to differentiate them from standard classification results), was 100% LOO.
This result is achieved by classifying using all the descriptors, but also in some classifications with less descriptors. The classification with the smallest number of descriptors producing 100% LOO, is using only 23 descriptors (out of 162).
To prove the applicability of these results, I have performed 20 "standard" classifications (with a "learned" group consisting of randomly selected 66% of the samples, used to classify the remaining 33%) in which only these 23 descriptors were used. The average success was 99.82%.
A reminder - the average result of a standard classification using all the descriptors (as written in the previous report), was 99.97%.

- If we allow a decrease of up to 10% compared to the maximum success results[2] (in case we need to classify in hard conditions and are willing to make that compromise) we need only 3 descriptors (97.43% LOO).
These descriptors are: '134-DS.i_sc_v', '138-DS.i_slope_v', '148-DT.g_ed'.
To show the authenticity of this result, I performed a 20-round standard classification using only these descriptors, which produced 97.27% success.

[1] Again, it is important to remember that the graph shows the successful results of classifying all the samples with "leave-one-out", and not the average success rate of a specific classification with randomly selected "trained" and "classified" groups.
[2] As already mentioned, I have retained files with all the classification results starting from all the descriptors and ending with a single one, so the 10% is only an example used for this report. I "know" which descriptors to remove for any other desired compromise percentage.

By the way, if we decide to keep only a single descriptor, it will be "DS.i_sc_v" - The spectral centroid, with 84.96% LOO.
As always I tested the applicability of this with 20 rounds - 84.57% success using only this descriptor.

Instrument Families Classification

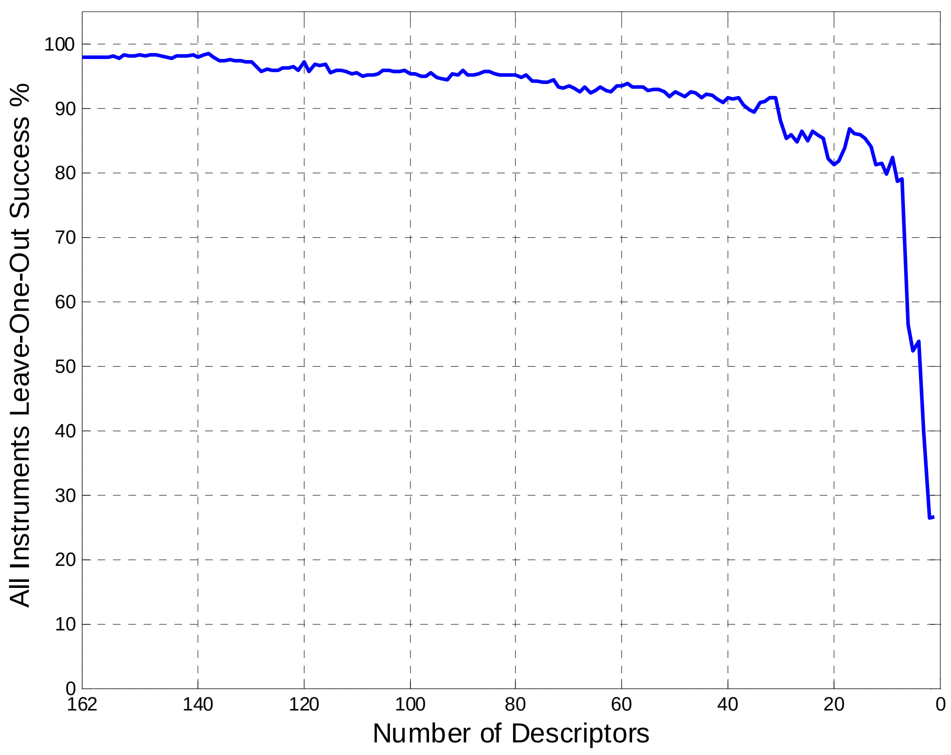


- The highest result which is 98.49% LOO, was received with 145 descriptors (out of 162). It is actually higher than the result of classifying using all the descriptors, which produces 98.26% LOO.

To show the applicability of this result, I perform 20 standard classification experiments with these 145 descriptors. The average result is 96.11% success.
The average result with all the descriptors (as mentioned in the previous report), was 95.24%.

- If we allow a decrease of up to 10% compared to the most successful classification, we could do with only 24 descriptors (89.27% LOO).
Performing 20 standard classifications with these 24 descriptors produces an average success of 88.79%.

Musical Instruments Classification



- Highest result is 98.34% LOO, and was attained by using 138 descriptors (out of 162). It is higher than the result of a classification with all the descriptors, which produced 97.96% LOO.

Performing 20 standard classifications with these 138 descriptors produced an average success of 95.92%. The result of classifying with all the descriptors was 95.85% success.

- By allowing a decrease of 10% compared to the highest results, we could get down to 31 descriptors (91.6% LOO).
Performing 20 standard classifications with these descriptors produced 93.87% success.

Differences between my program and the original one, when performing descriptor selection with Discriminant Analysis

First, here is the descriptor selection code copied from the original program:

```
liste_v = [];
for num_dim=1:param.gauss_nb_desc

  % === pour chaque dimension on ne garde que les plus importants
  tmp_v     = U_m(:, num_dim);
  max_value = max(tmp_v);
  pos_v     = find(tmp_v > param.desc_threshold*max_value);
  liste_v   = [liste_v, pos_v(:).'];

end, % === for num_dim

select_desc_v = unique(liste_v);
```

Besides the fact that my program continuously performs Discriminant Analysis, removes the "worst" descriptor, classifies and then records the results, there are other important changes:

- The Discriminant Analysis in the **original program** (both in the descriptor selection and in the space-transformation sections) has picked a constant ("gauss_nb_desc"=8) number of dimensions from the coefficient matrix without considering the specific classification that was used.

My program chooses the active dimensions, by picking the ones which are over a specific threshold, thus obtaining specific results for each taxonomy:
One dimension for Pizzicato/Sustain, 3 for Instrument Families and 15 for Musical Instruments.

- The **original program** after performing Discriminant Analysis on the descriptor matrix chooses the descriptors in this way:
From every dimension in the coefficient matrix, were chosen the descriptors with the coefficients that are higher than maximum the coefficient in that dimension ("max_value") multiplied by a threshold ("param.desc_threshold"=0.2).

My program <u>before</u> performing Discriminant Analysis:
1. First the program normalizes the descriptor vectors. It is important, because we shall try to understand which descriptors we should pick by looking at the "size" of their coefficients. Without normalization, some non-important descriptor might have, for example, very small values which will result in big coefficients - thus being mistakenly prioritized over more dominant descriptors.

My program <u>after</u> performing Discriminant Analysis:
1. First, the program converts the coefficient matrix to absolute values (the abs function). This is important, because there are descriptors which get negative coefficients with very high absolute values, but the original program wouldn't select these at all.

2. The program multiplies every dimension in the coefficient matrix by the corresponding Lambda value we got from the Discriminant Analysis, in order to give advantage to the "important" dimensions.

3. The coefficients of every descriptor (after the transformations written in the previous clauses), are summed over all the dimensions - thus descriptors which are "important" in more dimensions than others, gain an advantage.

4. Only then my program does the elimination of the less important descriptor and gathers the classification results.

The IOWA Samples

I have finished cutting the files I took from the IOWA sound database into single samples - I chose only the instruments that appear in Studio-Online (except Cello, for luck of space on the disk at the time - will be included in the future). I wrote two different programs that did the cutting - one of them used SilenceDetector. Both programs used the filenames of the samples (e.g. "AltoSax.NoVib.mf.Db3B3.wav") to calculate how many samples should result from the cutting and dynamically changing various thresholds to reach the right number of files. The last three files which the programs had trouble cutting, I did manually.

Total: 918 samples of 7 instruments - Bassoon, Clarinet, Flute, Tuba, French horn, Oboe and Trombone.

I have classified these samples using SOL samples as the "learned group", getting the results below. These results show the classification success of various groups of samples and the total success percentage.

<u>Abbreviations used:</u>
pp=pianissimo, mf=mezzo forte, ff=fortissimo - applicable for all instruments.
vib=vibrato, novib=no vibrato - applicable only for instruments that have vibrato samples in the database.
"good" means successfully classified.

RESULTS

Pizzicato / Sustain:
PP good : 265/289. This is 91.695502 Percent.
MF good : 315/315. This is 100.000000 Percent.
FF good : 324/324. This is 100.000000 Percent.

VIB good : 113/115. This is 98.260870 Percent.
NOVIB good : 105/112. This is 98.130841 Percent.

Total good : 904/928. This is 97.413793 Percent.

Instrument Families:
PP good : 249/289. This is 86.159170 Percent.
MF good : 283/315. This is 89.841270 Percent.
FF good : 219/324. This is 67.592593 Percent.

VIB good : 94/115. This is 81.739130 Percent.
NOVIB good : 102/112. This is 82.926829 Percent.

Total good : 751/928. This is 80.926724 Percent.

Musical Instruments:
PP good : 120/289. This is 41.522491 Percent.
MF good : 130/315. This is 41.269841 Percent.
FF good : 120/324. This is 37.037037 Percent.

VIB good : 103/115. This is 89.565217 Percent.
NOVIB good : 106/112. This is 89.830508 Percent.

Total good : 370/928. This is 39.870690 Percent.

I am just beginning to deal with the IOWA database, so these are only preliminary results.

Remarks about certain descriptors

- The following descriptors:

"3-DH.inharmo_v", "13-DH.i_tri1_v", "14-DH.i_tri2_v", "15-DH.i_tri3_v"

Are not calculated, and have a constant value of 0.
By the way, these are indeed the first 4 descriptors that are thrown away in the descriptor selection with Discriminant Analysis process.

- In the "Cuidado Report" it is written:
"Each descriptor has been designed in order to make the descriptor independent from the sound’s sampling rate and to the loudness of the sounds, so that the same sound recorded at two different levels and two different sampling rate has the same descriptors values."

I have taken a group of pianissimo samples of various instruments from the IOWA database, normalized them and calculated the descriptors for both groups. The results are different.

I have calculated the average of each descriptor in both groups, and looked at the differences among the groups.

Descriptors whose average differed by more than 10% are:

"17-DP.i_loud_v", "85-DP.flustr_v2", "86-DP.flustr_v3", "87-DP.flustr_v4", "93-DP.flustr_v10", "146-DT.g_am_am".

The biggest difference was 34.42%, encountered with "17-DP.i_loud_v".

Perhaps it is worth to normalize the samples before calculating the descriptors?, Or maybe there are descriptors which should not be used?, Maybe both.

Report - 18.02.2003

By Arie Livshin

Topics

- Classification of the IOWA sample database.
- Locating "bad" samples in a sample database.
- LOO ~ STND. Min-Max and Z-Score.

Terminology

Standard Classification - a classification method where a "learned group" of 66% of the samples is randomly selected from every classification group in the taxonomy, and is used to classify the rest of the samples in the database (called the "classified group").

Leave-One-Out - a method for evaluating the behavior of "standard classifications". In this method every single sample in the database is classified using all the other samples. The number of misclassified samples is summed. I call the percentage of misclassified samples, as well as the method itself, shortly - LOO.
The LOO in this report uses the KNN algorithm for classification, picking the best K out of a range of 1 to 20.
The LOO method has an advantage over the "standard classification" in the fact that it does not depend on any random selection of a "learned" group, and thus could be calculated only a single time without needing to average the results of many classification cases.

Classification of the IOWA sample database

In this section I shall present "standard classification" results of the IOWA database, averaged over 20 classifications. All the classifications were performed using the KNN algorithm, where the best K was picked out of the range of 1 - 20.

A reminder: in the previous report I presented classification results of classifying the IOWA samples using the SOL database as the "learned group". Here IOWA is used for creating both the "learned" and the "classified" groups.

IOWA database

Musical instruments:	97.68% success
Instrument families:	98.53% success
Pizzicato / Sustain:	N/A (all the samples in IOWA are sustained)

Although it was expected, it is still good to know that our classification method works well with different sample databases, and does not owe its success to some specific weirdness of SOL.

IOWA+SOL databases joined

Note: In all classifications where I use the IOWA+SOL joined database, I choose a "learned group" which contains 66% out of the samples in SOL and 66% out of the IOWA samples (contrary to picking 66% out of the merged database without distinction of the source databases).

Musical instruments:	94.45% success
Instrument families:	95.15% success
Pizzicato / Sustain:	99.81% success

We can see that by merging the two databases we got results which are worse than the results of every separated database - the databases somewhat "disturbed" the classifications of each other. After this said, it is still plain that the final results are not catastrophic.

Locating "bad" samples in a classification database

When a database is going to be used for classification, it is important to know whether it contains "problematic" samples - samples that will disturb the classification by being too different from other samples in the same classification category, and on the other hand resembling samples from other categories. Not only these samples will be misclassified themselves, but when used in the "learned group" they might actually "spoil" the classification of other samples.

I wrote a program which finds the "problematic" samples by first performing Space-Transformation of the descriptor matrix using discriminant analysis (in order to get dimension reduction and better class separation) and then doing a Leave-One-Out classification of the samples and reporting the misclassified ones.
Note: see next section for a "Monte-Carlo" demonstration which shows that LOO really correlates to the results of "standard" classifications.

The output of the program is the number of misclassifications for every sample, the most common classification mistake for that sample and whether this sample was also misclassified when using the best K for the current taxonomy.
I chose to present here only the output of the program regarding samples that were misclassified for every K - it is evident that such samples are too "far" from other samples in their classification group and resemble too much samples in other groups.

Various Results

Classification by Musical Instruments:

SOL database (1323 samples, 16 kinds of musical instruments)
The following 7 samples were misclassified for every K (1 - 20).
A total of 33 different samples were misclassified at least in one classification (with some K).

```
c:/Music_Research/samples/ADDSOL/13alto/alto_a_gref_mf_si5_12.wav
 Misclassified 20 times, instead of Viola - Violin

c:/Music_Research/samples/ADDSOL/12vln/vln_gref_e_mf_sib5_12.wav
 Misclassified 20 times, instead of Violin - Viola

c:/Music_Research/samples/ADDSOL/12vln/vln_gref_e_mf_fad5_12.wav
 Misclassified 20 times, instead of Violin - Viola

c:/Music_Research/samples/ADDSOL/11harp/harp_gref_mf_si2_12.wav
 Misclassified 20 times, instead of Harp - Guitar (Classic)

c:/Music_Research/samples/ADDSOL/11harp/harp_gref_mf_la3_12.wav
 Misclassified 20 times, instead of Harp - Guitar (Classic)

c:/Music_Research/samples/ADDSOL/11harp/harp_gref_mf_do5_12.wav
 Misclassified 20 times, instead of Harp - Guitar (Classic)

c:/Music_Research/samples/ADDSOL/09tubb/tubb_gref_mf_dod3_12.wav
 Misclassified 20 times, instead of Tuba Bass (F) - French Horn (F)
```

c:/Music_Research/samples/ADDSOL/05sax/sax_gref_pp_do4_12.wav
Misclassified 20 times, instead of Alto Saxophone (Eb) - Bassoon (French)

c:/Music_Research/samples/ADDSOL/05sax/sax_gref_pp_sib3_12.wav
Misclassified 20 times, instead of Alto Saxophone (Eb) - usually as Flute (18 times)

IOWA database (928 samples, 7 kinds of musical instruments)

The following 3 samples were misclassified for every K (1 - 20).
A total of 6 different samples were misclassified at least in one classification (with some K).

horn.pp.B1.wav
Misclassified 20 times, instead of French Horn (F) - Bassoon (French)

BbClar.pp.Db6.wav
Misclassified 20 times, instead of Clarinet (Bb) - Flute

AltoSax.Vib.pp.Db4.wav
Misclassified 20 times, instead of Alto Saxophone (Eb) - Clarinet (Bb)

IOWA+SOL databases (2251 samples, 16 kinds of musical instruments)

The following 24 samples were misclassified for every K (1 - 20).
A total of 129 different samples were misclassified at least in one classification (with some K).
We can see that when putting the databases together, they "confuse" each other a little - the results are worse than the total of their misclassified samples when classified separately.
This doesn't mean the results are "bad" - they are not. See previous section for actual classification results of IOWA+SOL.

oboe.pp.B4.wav
Misclassified 20 times, instead of Oboe - Flute

bassoon.ff.Bb1.wav
Misclassified 20 times, instead of Bassoon (French) - Trombone (Tenor/Bass)

bassoon.ff.B1.wav
Misclassified 20 times, instead of Bassoon (French) - French Horn (F)

BbClar.pp.D5.wav
Misclassified 20 times, instead of Clarinet (Bb) - Flute

BbClar.ff.F5.wav
Misclassified 20 times, instead of Clarinet (Bb) - Flute

AltoSax.Vib.pp.Db4.wav
Misclassified 20 times, instead of Alto Saxophone (Eb) - usually as Clarinet (Bb) (17 times)

AltoSax.Vib.ff.Eb5.wav
Misclassified 20 times, instead of Alto Saxophone (Eb) - usually as Clarinet (Bb) (18 times)

AltoSax.NoVib.mf.Ab5.wav
Misclassified 20 times, instead of Alto Saxophone (Eb) - Clarinet (Bb)

```
c:/Music_Research/samples/ADDSOL/16acco/acco_gref_mg_se_mf_la4_12.wav
 Misclassified 20 times, instead of Accordion - usually as Clarinet (Bb) (17 times)

c:/Music_Research/samples/ADDSOL/14vcl/vcl_gref_a_mf_sold4_12.wav
 Misclassified 20 times, instead of Cello - Viola

c:/Music_Research/samples/ADDSOL/13alto/alto_a_gref_mf_mi5_12.wav
 Misclassified 20 times, instead of Viola - Violin

c:/Music_Research/samples/ADDSOL/12vln/vln_gref_e_mf_sib5_12.wav
 Misclassified 20 times, instead of Violin - Viola

c:/Music_Research/samples/ADDSOL/11harp/harp_gref_mf_si2_12.wav
 Misclassified 20 times, instead of Harp - Guitar (Classic)

c:/Music_Research/samples/ADDSOL/11harp/harp_gref_mf_si1_12.wav
 Misclassified 20 times, instead of Harp - Guitar (Classic)

c:/Music_Research/samples/ADDSOL/11harp/harp_gref_mf_la3_12.wav
 Misclassified 20 times, instead of Harp - Guitar (Classic)

c:/Music_Research/samples/ADDSOL/10gui/gui_gref_b_mf_re3_12.wav
 Misclassified 20 times, instead of Guitar (Classic) - Harp

c:/Music_Research/samples/ADDSOL/05sax/sax_gref_pp_sol4_12.wav
 Misclassified 20 times, instead of Alto Saxophone (Eb) - Clarinet (Bb)

c:/Music_Research/samples/ADDSOL/05sax/sax_gref_pp_dod4_12.wav
 Misclassified 20 times, instead of Alto Saxophone (Eb) - Clarinet (Bb)

c:/Music_Research/samples/ADDSOL/05sax/sax_gref_pp_do4_12.wav
 Misclassified 20 times, instead of Alto Saxophone (Eb) - Bassoon (French)

c:/Music_Research/samples/ADDSOL/05sax/sax_gref_mf_la4_12.wav
 Misclassified 20 times, instead of Alto Saxophone (Eb) - Clarinet (Bb)

c:/Music_Research/samples/ADDSOL/03clsb/clsb_gref_pp_mi5_12.wav
 Misclassified 20 times, instead of Clarinet (Bb) - Flute

c:/Music_Research/samples/ADDSOL/03clsb/clsb_gref_pp_mi4_12.wav
 Misclassified 20 times, instead of Clarinet (Bb) - Flute

c:/Music_Research/samples/ADDSOL/02htb/htb_gref_pp_fa4_12.wav
 Misclassified 20 times, instead of Oboe - usually as Clarinet (Bb) (11 times)

c:/Music_Research/samples/ADDSOL/01fltu/fltu_gref_mf_do5_12.wav
 Misclassified 20 times, instead of Flute - Clarinet (Bb)
```

Classification by Instruments Families:

SOL database (1323 samples, 4 kinds of instrument families)
The following 16 samples were misclassified for every K (1 - 20).
A total of 56 different samples were misclassified at least in one classification (with some K).
We can see that although the "instrument groups" taxonomy contains only 4 classification groups compared to the 16 of "musical instruments", there are more misclassified SOL samples there than in the "musical instruments" taxonomy.

c:/Music_Research/samples/ADDSOL/16acco/acco_gref_mg_se_mf_sold0_12.wav
Misclassified 20 times, instead of Flute/Reeds - Bowed Strings

c:/Music_Research/samples/ADDSOL/16acco/acco_gref_mg_se_mf_sol2_12.wav
Misclassified 20 times, instead of Flute/Reeds - Bowed Strings

c:/Music_Research/samples/ADDSOL/16acco/acco_gref_mg_se_mf_sol0_12.wav
Misclassified 20 times, instead of Flute/Reeds - Bowed Strings

c:/Music_Research/samples/ADDSOL/16acco/acco_gref_mg_se_mf_mib2_12.wav
Misclassified 20 times, instead of Flute/Reeds - Bowed Strings

c:/Music_Research/samples/ADDSOL/16acco/acco_gref_mg_se_mf_dod2_12.wav
Misclassified 20 times, instead of Flute/Reeds - Bowed Strings

c:/Music_Research/samples/ADDSOL/08tbtb/tbtb_gref_pp_mi4_12.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_pp_re5_12.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_mf_re5_12.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_mf_dod5_12.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_mf_do5_12.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/06cor/cor_gref_pp_re4_12.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/06cor/cor_gref_pp_do4_12.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/04bsn/bsn_gref_pp_sib0_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

c:/Music_Research/samples/ADDSOL/04bsn/bsn_gref_pp_si0_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

c:/Music_Research/samples/ADDSOL/04bsn/bsn_gref_pp_fad1_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

c:/Music_Research/samples/ADDSOL/04bsn/bsn_gref_mf_sib0_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

SOL database (928 samples, 2 kinds of instrument families)
The following single sample was misclassified for every K (1 - 20).
A total of 3 different samples were misclassified at least in one classification (with some K).

bassoon.pp.Gb2.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

SOL database (2251 samples, 4 kinds of instrument families)
The following 47 samples were misclassified for every K (1 - 20).
A total of 161 different samples were misclassified at least in one classification (with some K).

horn.pp.F5.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

horn.ff.F5.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

horn.ff.E5.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

bassoon.pp.C2.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

bassoon.pp.Bb1.wav
Misclassified 20 times, instead of Flute/Reeds - Plucked Strings

bassoon.pp.B1.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

bassoon.ff.B1.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

BbClar.pp.G5.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

BbClar.mf.D3.wav
Misclassified 20 times, instead of Flute/Reeds - Bowed Strings

BbClar.mf.C6.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

c:/Music_Research/samples/ADDSOL/16acco/acco_gref_mg_se_mf_sol2_12.wav
Misclassified 20 times, instead of Flute/Reeds - Bowed Strings

c:/Music_Research/samples/ADDSOL/16acco/acco_gref_mg_se_mf_sol0_12.wav
Misclassified 20 times, instead of Flute/Reeds - Bowed Strings

c:/Music_Research/samples/ADDSOL/16acco/acco_gref_mg_se_mf_mib2_12.wav
Misclassified 20 times, instead of Flute/Reeds - Bowed Strings

c:/Music_Research/samples/ADDSOL/16acco/acco_gref_mg_se_mf_fa2_12.wav
Misclassified 20 times, instead of Flute/Reeds - Bowed Strings

c:/Music_Research/samples/ADDSOL/16acco/acco_gref_mg_se_mf_fa0_12.wav
Misclassified 20 times, instead of Flute/Reeds - Bowed Strings

```
c:/Music_Research/samples/ADDSOL/15cb/cb_g_gref_mf_mi3_12.wav
 Misclassified 20 times, instead of Bowed Strings - Flute/Reeds

c:/Music_Research/samples/ADDSOL/13alto/alto_c_gref_mf_fa2_12.wav
 Misclassified 20 times, instead of Bowed Strings - Flute/Reeds

c:/Music_Research/samples/ADDSOL/12vln/vln_gref_g_mf_do3_12.wav
 Misclassified 20 times, instead of Bowed Strings - Flute/Reeds

c:/Music_Research/samples/ADDSOL/12vln/vln_gref_a_mf_do4_12.wav
 Misclassified 20 times, instead of Bowed Strings - Flute/Reeds

c:/Music_Research/samples/ADDSOL/09tubb/tubb_gref_pp_sold1_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/09tubb/tubb_gref_pp_sol1_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/09tubb/tubb_gref_pp_fad1_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/09tubb/tubb_gref_pp_dod3_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/08tbtb/tbtb_gref_pp_si0_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/08tbtb/tbtb_gref_pp_mi4_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/08tbtb/tbtb_gref_pp_dod1_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/08tbtb/tbtb_gref_mf_do4_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_pp_sold4_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_pp_si4_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_pp_re5_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_pp_la4_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_pp_do5_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_mf_sold4_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_mf_si4_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_mf_re5_12.wav
 Misclassified 20 times, instead of Brass - Flute/Reeds
```

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_mf_dod5_12.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/07trpu/trpu_gref_mf_do5_12.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/06cor/cor_gref_pp_do4_12.wav
Misclassified 20 times, instead of Brass - Flute/Reeds

c:/Music_Research/samples/ADDSOL/06cor/cor_gref_mf_sol0_12.wav
Misclassified 20 times, instead of Brass - Plucked Strings

c:/Music_Research/samples/ADDSOL/04bsn/bsn_gref_pp_sib3_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

c:/Music_Research/samples/ADDSOL/04bsn/bsn_gref_pp_sib0_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

c:/Music_Research/samples/ADDSOL/04bsn/bsn_gref_pp_si0_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

c:/Music_Research/samples/ADDSOL/04bsn/bsn_gref_pp_re1_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

c:/Music_Research/samples/ADDSOL/04bsn/bsn_gref_pp_fa1_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

c:/Music_Research/samples/ADDSOL/04bsn/bsn_gref_mf_sib0_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

c:/Music_Research/samples/ADDSOL/03clsb/clsb_gref_mf_mib5_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

c:/Music_Research/samples/ADDSOL/02htb/htb_gref_mf_sold3_12.wav
Misclassified 20 times, instead of Flute/Reeds - Brass

Classification by Pizzicato / Sustain:

SOL database (1323 samples, 2 classification groups)
All samples classified correctly.

The IOWA database contains only Sustained samples.

SOL+IOWA databases (2251 samples, 2 classification groups)
There are no samples that were misclassified in every K (1 - 20).
A total of 4 different samples were misclassified at least in one classification (with some K).

Remark
I believe there are probably lots of techniques for locating bad samples in Statistics and related fields - Data-Mining, Learning Systems and others.

LOO ~ STND. Min-Max and Z-Score

Purpose

- I have used LOO to quantify the success of different processes (descriptor elimination, "bad" samples location, etc). In this section I will try to plainly show that there is indeed a strong correlation between the results of LOO and the behavior of the results of "standard" classifications.
For more explanation on LOO and its advantages see "Definitions" at the beginning of the report.

- Using the opportunity that we shall have so much "standard" classification results, we could see by directly comparing absolute classification results (LOO only shows us the behavior of the results graph), which normalization method actually produces the best classification success percentage.
We shall find which the best method is when:

- Using all the required descriptors to get the maximum success percentage.
- Using the minimum number of descriptors and still get "acceptable" results.

Content explanation:

This section will show several graphs. Each graph shows the results of a process where Discriminant-Analysis was performed on the descriptor matrix of the SOL database, and by using the resulting coefficient matrix a gradual elimination of the least important descriptors was performed with a "standard" classification of the remaining descriptor matrix in each step (for details on the process of descriptor elimination, see previous report).
The difference between this process and the one performed in the previous report, is that here we actually use "standard" classifications in every step, and not just produce LOO results.

Each graph is labeled by: X - Y.
X - Specifies the normalization method that was performed on the data before the Discriminant-Analysis was done.
Y - Which normalization was performed on the data before it was classified with the specified classification method. Although the descriptor hierarchy is calculated using normalized data, we might still choose to use a different normalization method when performing the actual classification.

The various normalization methods used are:

"**Min-max normalization**

It performs a linear transformation on the original data.

V' = (V - min) * (new_max - new_min) / (max - min) + new_min

The advantage of this method is that it preserves all relationships of the data values exactly. It does not introduce any potential bias into the data. The disadvantage is that it will encounter an "out of bounds" error if a future input case falls outside of the original data range.

Z-score normalization

The values are normalized based on the mean and standard deviation.

V' = (V - mean) / std

This method works well in cases you do not know the actual minimums and maximums of your input data or when you have outlier values that dominate a min-max normalization."

(Excerpts from http://home.olemiss.edu/~xzhou/progress.html, by Xiaodong Zhou).

None
Using the descriptor matrix without normalization.

Classification methods:
LOO - in these graphs the classification was done using the Leave-One-Out method.
Reminder - every sample was in its turn removed from the descriptor matrix, and classified using the rest of the samples. The number of misclassified samples was summed.
STND - in this graphs the classification was done using "standard classification".
Reminder - a random group consisting of 66% of the samples was chosen out of every classification group, and was used to classify the remaining samples. This process was repeated 20 times. The graph shows the average success percentage[1].

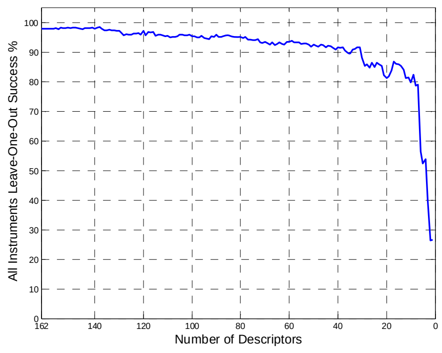
Remarks:
- Both the LOO and STND use the KNN algorithm, where the best K is selected out of the range of 1 - 20 by using LOO on the "learned group", and choosing the K which produces the least internal misclassifications.

- There is no use of calculating the Discriminant Analysis without performing some normalization on the data first. That is why there aren't such graphs.
The reason why I do show graphs that depict classifications on non-normalized data is that classifying without normalizing the descriptor matrix can often yield better success percentage when using a high number of descriptors. It is obvious that these graphs will resemble the LOO graphs less than graphs that show classification results of data that was normalized using the same method before the Discriminant-Analysis, as after.

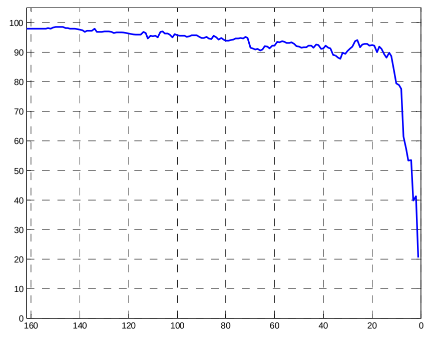
[1] This is a "Monte-Carlo showcase" which comes to show that LOO results really reflect STND results. This sort of "proofs" use the method of exemplifying a claim so many times, that it convinces that at least usually, the claim is true.

Musical Instruments

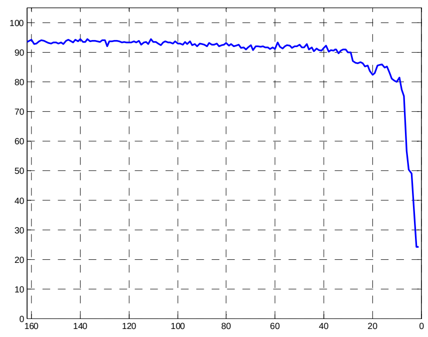
min-max - min-max LOO



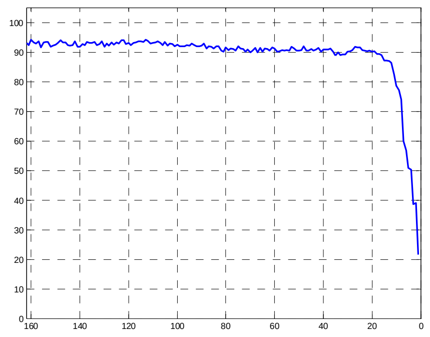
z-score - z-score LOO



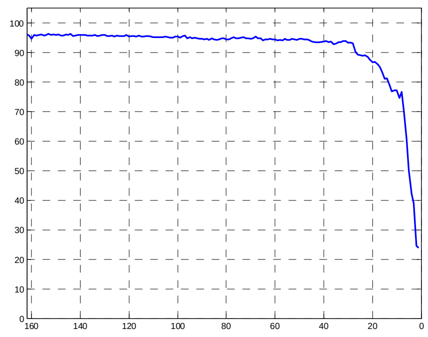
min-max - min-max STND



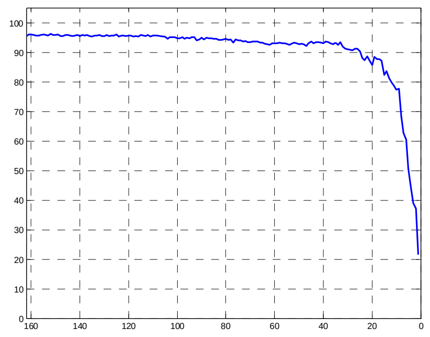
z-score - z-score STND



min-max - none STND



z-score - none STND



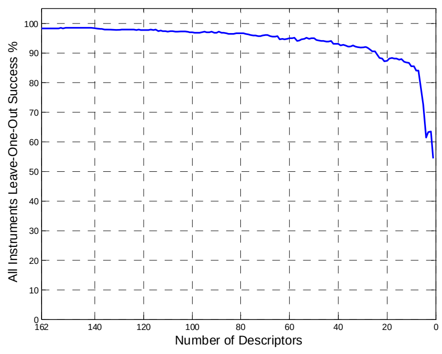
	M by none	**Z by none**	**M by M**	**Z by Z**
Max score	153, 96.14%	152, 96.14% ✓	111, 94.41%	160, 94.15%
90% min Desc.	27, 90.03%	25, 90.35%	31, 90.90%	19, 90.26% ✓

Remarks:
- Every cell of the table shows the number of descriptors and the average success percentage of the classification.
- The scores shown in the table are taken only out of the "standard" classification graphs. There is no use in directly comparing LOO and STND success percentages.
- "max-score" - this is the highest classification score, achieved with any number of descriptors.

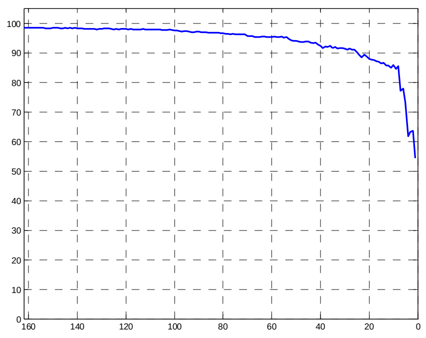
- "90% min Desc." - this means the lowest number of descriptors needed to achieve at least 90% success. The number 90% is just an arbitrary choice (let's imagine that a client is willing to compromise on classifications that produce a maximum error of 10%).

Instrument Families

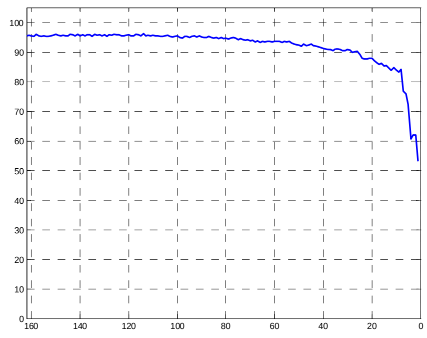
min-max - min-max LOO



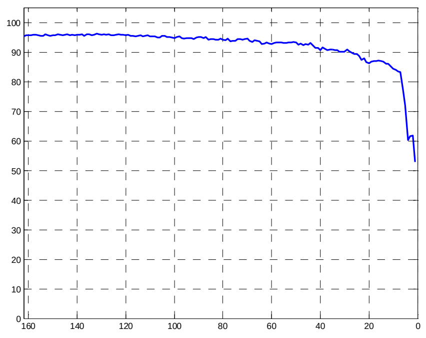
Z-score - z-score LOO



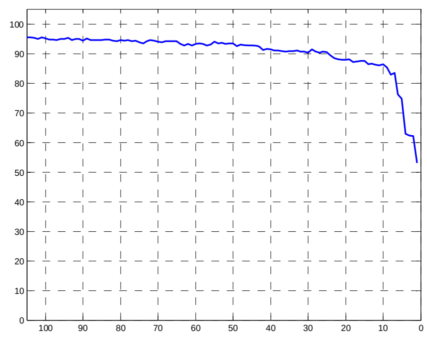
min-max - min-max STND



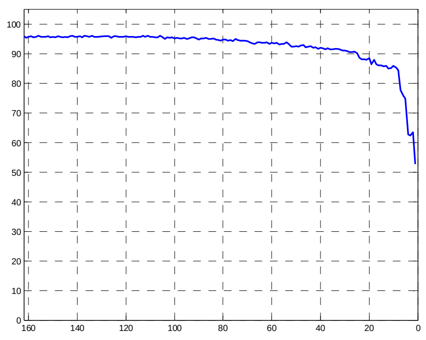
z-score - z-score STND



min-max - none STND



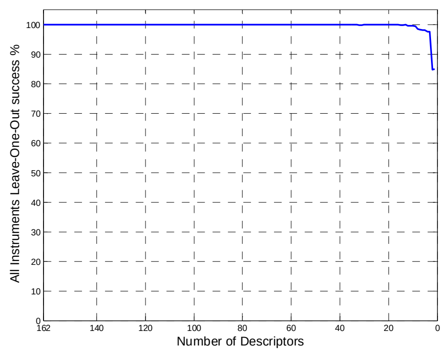
m-score - none STND



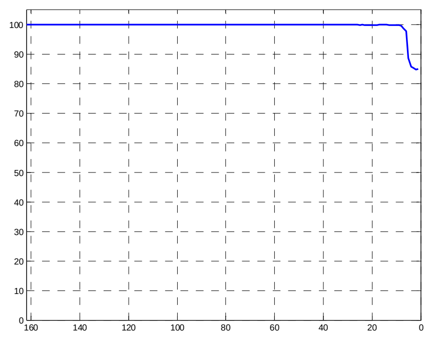
	M by none	Z by none	M by M	Z by Z
Max score	127, 96.47% ✓	134, 96.09%	132, 96.14%	114, 96.13%
90% min Desc.	25, 90.48% ✓	25, 90.11%	28, 90.32%	26, 90.24%

Pizzicato / Sustain

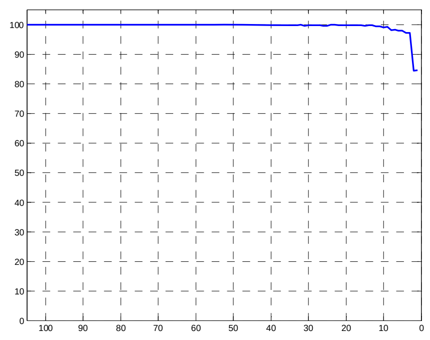
min-max - min-max LOO



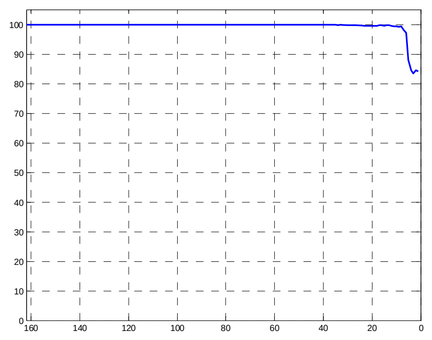
Z-score - z-score LOO



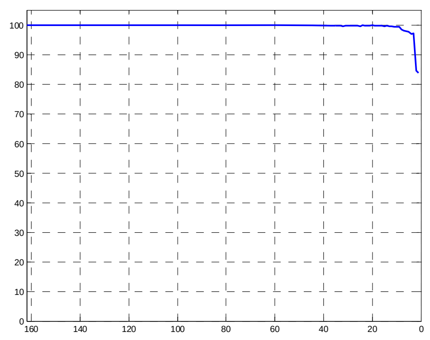
Min-max - min-max STND



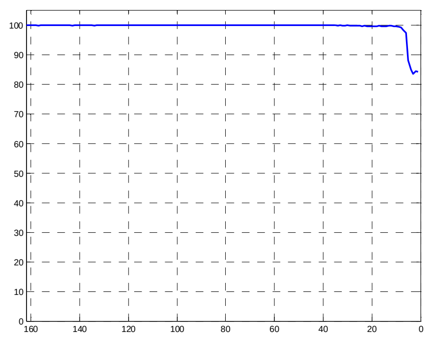
z-score - z-score STND



min-max - none STND



Z-score - none STND



	M by none	Z by none	M by M	Z by Z
Max score	101, 100%	98, 100%	68, 100% ✓	52, 99.99%
90% min Desc.	3, 97.09%	6, 97.35%	3, 97.13% ✓	6, 97.15%

Results:

- We see that the graphs of the LOO results and the ones with the results that use the same kinds of normalization are indeed similar, although the absolute values and the slopes could vary a bit.

- When looking at the maximum results, we can see that classifying non-normalized data using the descriptors chosen with the Min-Max classification, always yields the maximum success percentage (although sometimes using more descriptors than other methods).

- When looking for a minimal number of descriptors, it is harder to put the finger on the winning method.

- In the "musical instruments" taxonomy, Z-score has beaten the rest. It is evident from the graph that with a low number of descriptors this method still retains high success results until the "big fall".
- In the "instrument families" taxonomy, "M by none" has won, but we can see by examining the graphs that it was because the 90% was a lucky number for this classification method and that the two methods that use normalized data have actually kept high-results quite longer - with less descriptors, before the "big fall".
- In "pizzicato / sustain", Z-score has lost. It chose worse descriptors than Min-Max, and was left with 6 descriptors at a score above 90%. The graph "fell" quicker than the methods that used a descriptor hierarchy calculated with discriminant analysis that used the Min-Max normalization.

  By the way - the last descriptor that was left when using Z-score in "Pizzicato / Sustain" classification is the same one that Min-max has left (see previous report) - the Spectral Centroid.

Conclusions:

- At least for me, LOO has proven itself as a worthy behavior quantifier for the success percentage of "standard" classifications.

- In my work I am more interested in higher classification success percentage than in descriptor elimination, so it seems that using non-normalized data for classifications is best for my cause.

Report - 03.03.2003
By Arie Livshin

Regarding my paper:
When I wrote an abstract for the paper, I discovered two things that I miss:

1. I wanted to show in the article that our classification results could be improved by removing "bad" samples. I performed various tests (and read some articles) which convinced me that it is very hard to do that when we are dealing with a database which is almost without noise (outliers), like our SOL and IOWA.
What I did manage to do, was to improve the self consistency of the database by removing outliers using Leave-One-Out. More details on that in the following section.

Remark: I shall use the concepts "Noise", "Bad samples" and "outliers" interchangeably during this report.

2. I wanted to claim that by enriching the classifier sound database by new versatile sound samples, we get better and better results with classification of new sounds, thus with time, we get closer to actually defining the differences between the musical instruments rather than just learning to classify specific samples. Descriptor reduction on such a database will help us to find out which attributes really differentiate among the instrument classes.
A humble and intuitive way to show that this claim might be true, is to demonstrate that classifying sound database C by databases A+B leads to better results than by A or B alone.
Unfortunately, I couldn't get a 3'rd sound database yet. Romain Mules told me he has a sound database on the Intranet, but I need to get special passwords from Olivier Labat - I didn't manage to reach him yet.

Removal of bad samples

In short
The experiments could be divided into two groups:
1. The program first selects the "learned group" (66%) and removes the outliers out of it. Then this group is used to classify the rest of the samples.
In these experiments I always got classification results which were worse than when not removing outliers at all.
2. The program removes the outliers from all of the database and only after doing that it selects the "learned group" and makes the classifications. Using this technique, only my method of removing outliers with Leave-One-Out produced better results than without removing outliers, and still the difference was quite small - 96.84% success compared with 95.85% without removing outliers(in the "musical instruments" taxonomy).

Algorithms

- Removal of bad samples using Leave-One-Out (LOO):
Every sample in the Database is removed in its turn from the database and classified by all the rest. Samples which were misclassified are removed. The process reiterates until all samples are correctly classified.

- Interquantile Range - as used by Geoffroy Peeters in his code (IQR):
For every descriptor, let P1 be the value bigger than 99% of the values of this descriptor, and let P2 be the value that is bigger than 1% of the values.
Remove the values that are larger than P1+(P1-P2)*C and the ones smaller than P2-(P1-P2)*C. C is some scalar (e.g. 1).
The process is repeated 20 times over all the descriptors.

Remarks on methods from this family:
1. In this method the whole DB is searched and the outliers are removed before a "learned group" is selected. I believe that in such cases we cannot actually claim to achieve an "improvement in classification" as we are not really "allowed" to remove samples which are going to be in the "test group" (our mission is to classify all of it), unless we count the samples removed from the "test group" as misclassifications when computing the success percentage.

2. IQR as implemented here, does not use classification information but looks for outliers relative to all of the samples in the database. Because of that, it will not detect "Class noise" - samples which are quite normal by themselves but were classified to the wrong class. In this sense, I think that IQR might be more appropriate for removing freak samples ("Attribute noise") from databases that don't include class information than for our case.

- Modified IQR:
This is my variation of Interquantile Range.
Change 1: Perform IQR on each class separately and not on all the samples together.
Change 2: Do not immediately remove a sample containing an outlier in one of its descriptors, but rather count for every sample the number of descriptors which produce outliers. At the end of the process remove the samples which have the largest number of outliers.

- Matlab version:
In this outlier removal version I used an example from the Matlab help section: Calculate the mean and standard deviation (STD) of every descriptor, and then remove the samples which have values of that descriptor larger than C*STD, where C is some scalar.

Results:

In the following experiments the outliers were removed from the whole database before selecting the "learned group":

IQR:

(Using Geoffroy's code) - I performed tests with the original parameters, and also altered them in various ways, and the results I achieved (an average of 20 classification experiments) is:

Musical instruments:
95.58% success after removing the outliers (without removing outliers - 95.85%)

Instrument Families:
95.4% success (without IQR - 95.24%)

Pizzicato / Sustain:
99.88% success (without IQR - 99.97%).

Number of samples removed: 14.

All in all: the results are not as good as without outlier removal.

LOO:

The samples I removed were the ones that were incorrectly classified with LOO for every K=1..20 using the KNN algorithm.
As already mentioned, these results do not reflect actual classification improvements, because the "test group" shouldn't be touched. They do reflect the self consistency of the database - how well the samples are divided into classes within the database.
As this method of removing outliers takes into account the actual classification, the number of samples removed depends on the taxonomy.

Average over 20 classification experiments:

Musical instruments - 10 samples removed.
96.84% success (without removing outliers - 95.85%)

Instrument families - 21 samples removed.
96.85% success (without removing outliers - 95.24%)

Pizzicato / Sustain - none were removed.

All in all: The database consistency is improved.

Both the Matlab-version and the modified-IQR algorithms produced worse results after removing outliers than without removing them.

When the outliers were removed only from the "learned group" and then the rest of the samples were classified by it, the results were worse than without removing the outliers, <u>with all 4 algorithms</u>.
In each experiment I first selected the "learned" and the "test" groups, and then compared the results of classifying <u>the same</u> "test" group with the "learned group" with and without the outliers.

I don't bring here the results of these classifications, as I really did **A lot** of experimenting playing with the various constants and parameters, and evaluating usually less than 20 rounds - just to get convinced that a specific configuration doesn't work.

<u>Conclusions</u>

Our sound databases almost contain no noise - the samples are recorded well (not much "attribute noise"), and practically all are correctly classified (no "class noise").
All the methods for outlier removal are also removing a small number of "good" samples ([1], [2], [3]). When I try to remove outliers only from our random "learned group" of 66% of the samples, the good samples removed cause more damage than is gained by removal of the "bad" samples.
An easy way to show that an outlier removal algorithm really works well, is to deliberately introduce noise into the database, and then show how the classification results improve after removing this noise using our algorithms and what is the noise ratio of the removed samples (how many "bad" samples removed compared to how many "good" ones). If I was to write an article solely on outlier removal, I would definitely do that.
But… <u>should I bother with it in my current paper</u>, or just make do with "let's improve the self consistency of our database by removing outliers", then use LOO on the entire database and finally show that the classification improves as an indication that the database is more consistent?

More on the IOWA database

Enlarging the database

I have cut and added several new instruments to my IOWA collection:
Piano, Cello - bowed and pizzicato, Contrabass - bowed and pizzicato.
Now my IOWA collection has grown to 2440 samples.

I classified Piano as pizzicato. We could see that this is an appropriate classification, because when IOWA is classified by SOL using the Pizzicato/Sustain taxonomy (in this case piano does not classify itself - SOL has no piano), the piano is indeed classified as pizzicato.
In the "Instrument family" taxonomy, I classified it in a new group as "Hammered Strings".

Cello pizzicato and Bass pizzicato were classified as new instruments in the "musical instruments" taxonomy, and not as Cello and Bass. A good indication that this is the right way to classify, is that when we classify IOWA using SOL in the "musical instruments" taxonomy, these samples are always classified either as Harp or Guitar (SOL doesn't include pizzicato versions of the violin family), and never as Cello or Bass.

Classification results of "Big IOWA" (IOWA which includes the new samples)
2440 samples, results averaged over 20 experiments with a "learned group" of 66% of the samples:
Musical Instruments (12 categories): 98.08%
Instrument Families (5 categories): 98.86%
Pizzicato / Sustain : 99.78%

Classification of SOL+Big IOWA (3763 samples, 20 experiments)
Musical Instruments (19 categories): 94.74%
Instrument Families (5 categories): 95.15%
Pizzicato / Sustain: 99.52%

A point to discuss:

When classifying IOWA using SOL (and vice versa), the classification results are not good. The highest results I achieved up to date in the "musical instruments" taxonomy, are 51%.
On the other hand, if I do space transformation by calculating the coefficient matrix using Discriminant Analysis on SOL+IOWA together, and then take SOL multiplied by this matrix as the "learned group" and classify IOWA multiplied by this matrix, the results are very high.

Classifying Big-IOWA by SOL
Musical instruments: 94.39% (without piano, pizzicato Cello, and pizzicato bass - these instruments are not included in SOL)

Instrument Family: 97.43% (no piano - hammered strings)

Pizzicato / Sustain: 99.8% (all instruments are included - even piano and pizzicato strings were correctly classified)

Classifying SOL by Big-IOWA:
Musical instruments: 83.5% (removed 7 instruments not in Big-IOWA)
Instrument Families: 91.16% (all included)
Pizzicato / Sustain: 99.55%   (all included)

As noted in the results, when I classify one database using the other, I remove the instruments from the classified database that the classifier doesn't have.

Question:

Do these classification results have any meaning or use at all, or does the space transform just somehow puts all the samples of the same class together, and then it's not a problem anymore for SOL to classify IOWA without it being an indication for anything?

We could of course benefit from computing the coefficient matrix by discriminant analysis on the IOWA+SOL merged database, for the reason I mentioned before:
As the database becomes bigger and bigger, with more and more versatile and different samples of the same instrument, then when we do descriptor reduction using DA we are left with descriptors that really tell us what are the main differences between the sounds of the actual musical instruments rather than how to distinguish a specific type of samples SOL has from another one.
(And we have seen that IOWA+SOL classifies itself well, so we don't lose precision by merging these databases).

References:

1. "Robust decision trees: removing outliers from databases" - 1995.
   G. H. John. Stanford University.

2. "Informal identification of outliers in medical data" - 2000.
   J. Laurikalla, M. Juhola, E. Kentala. Tampere University, Helsinki University.

3. "Ensemble of classifiers for Noise detection in PoS tagged corpora" - 2000.
   H. Berthelsen, B. Megyesi. Stockholm University.

Mini-Report - 0503..2003

By Arie Livshin

Confidence Intervals

I recalculated the classification results of SOL with and without removing outliers using LOO and computed their confidence intervals with 95% confidence levels. The reason I had to repeat the experiments and couldn't use my old data is that in order to calculate the confidence intervals one needs the standard deviation, but I didn't use to keep the results of each separate experiment, only the average over 20 classifications.

This time, each experiment includes **50** "standard" classification rounds.
(The reason I use "standard" classification to test the results and not just perform LOO on the whole database and report the number of misclassifications, is because by definition LOO will find 0 misclassified samples after removing the samples which were misclassified by LOO).

"Musical instruments" taxonomy

Without removing outliers:
mean (over 50 experiments): 95.913832, standard deviation : 0.899466
Confidence interval of 95 percent: 95.664513 to 96.163152

Repeating the experiment (just to make sure the results are consistent):
mean : 95.773243, standard deviation : 0.963968
Confidence interval of 95 percent: 95.506044 to 96.040441

After removing outliers from the whole database: (10 samples removed)
mean : 96.315068, standard deviation : 0.876400
Confidence interval of 95 percent: 96.072143 to 96.557994

"Instrument families" taxonomy

Without removing outliers:
mean : 95.578231, standard deviation : 0.769654
Confidence interval of 95 percent: 95.364894 to 95.791568

After removing outliers from the whole database: (21 samples removed)
mean : 96.857143, standard deviation : 0.990515
Confidence interval of 95 percent: 96.582586 to 97.131700

Conclusions
The fact that by repeating a set of 50 classifications I still managed to get different confidence intervals (though not different by "a lot"), shows that even 50 experiments are "not so much", and that I overestimated the prediction accuracy of my usual 20 experiments.

We can see in the "Musical Instruments" taxonomy that the confidence intervals of the results when removing and not removing outliers are almost non-overlapping, meaning that the "true averages" probably are different from each other.

In the "Instrument families" taxonomy the difference between the means is bigger than in the "Musical instruments", and the confidence intervals do not overlap at all. This means we can predict that at least in 95% of the classification cases (over 50 experiments) the average success result with removing outliers will be higher than the result of a classification which uses all the samples.

Report - 10.03.2003
By Arie Livshin

In this report I will try to show that by enriching the classifier database with sound samples from other databases, we help it to generalize better the actual taxonomies it describes (i.e. define what a violin sound is "really" like, and not just what specific kinds of samples SOL has), thus making it better suited for classification of samples from new databases.

In this demonstration, I shall use several sound databases:

	# samples	# instruments	# families	Pizzicato/sustain
SOL	1323	16	4	Both
IOWA	2440	12	5	Both
McGill	85	7	3	Sustain
Prosonus	262	9	3	Sustain
Vitus	271	8	3	Sustain

Every sound database will be classified by every other database, using the Leave-One-Out method. Next it will be classified using all the other databases put together - I shall call this method "minus-1", as "Leave-One-Out" is already used in many other connotations.

Before presenting the results, I would like to mention several items:

It is important to note that before every classification I have performed Space-Transformation using Discriminant Analysis on the classifier database, and then multiplied the test database with the resulting matrix.
In previous reports, when IOWA was classified using SOL, doing this kind of Space-Transformation had worsened the classification results - the coefficient matrix was only adapted to the specifics of the classifier database.
In the tables below, we can see that when classifying by several databases together, the multiplication of the descriptors of the classified database by the coefficient matrix calculated using the classifier, actually improves the classification results in most cases - an indication that the descriptors which got high coefficients are nearer this time to the "actual instrument" descriptors - the ideal descriptors which really encompass the difference between the instruments, not just the difference among classes in a specific database.

We can also see that clearly in the following test:
I have merged all the databases together (a total of 4381 samples), and performed a gradual descriptor elimination using discriminant analysis with the Pizzicato / Sustain taxonomy. As reported previously, doing that with SOL has left us with the last descriptor - "spectrum centroid" - a weird choice.
This time, the last descriptor which survived (with a very high grade of 97% LOO) was "148 - DT.g_ed" - the "effective duration" !

From the Cuidado report:
"The effective duration is a measure of the time the signal is perceptually meaningful. It allows distinguishing percussive sounds from sustained sounds but depends on the recording length. It is approximated by the time the energy envelop if above a given threshold".

Exactly what was desired.

In the future, if I shall have a very big / diverse database, I will be happy to write an article about the "right" descriptors - the ones which really encompass the differences among instruments. At this stage, unfortunately, the highest results with the "Instruments" taxonomy were 69% - too soon for such a list.

<u>Remark:</u>
In the following classifications, the instruments of the classified database which were not present in the classifier database were removed. Because of this fact, we don't have the situation in which adding databases to the classifier could improve the results because it adds missing instruments, on the contrary - adding an instrument to the classifier just complicates the classification:
Either the classified database also has this instrument; then it will not be removed from it and we have to classify an extra class (which was removed during previous classifications), or the classified database doesn't have the new class; this means the new class will be present only in the classifier, and could only worsen the results by confusing the classification.

The actual results:

The first column in the following tables is the name of the classified database. The first row shows which database was used as the classifier.
"Minus-1": this means that the database was classified by the rest of the databases put together (a kind of Leave-One-Out).
"No S-T": No space transformation using discriminant analysis was performed before the classification.

Each classification had to be performed only once, as the whole classified database was classified by the whole classifier and no randomness was introduced.

Instruments classification	SOL	IOWA	McGill	Prosonus	Vitus	**Minus-1**	**Minus-1** No S-T
SOL by		42.44	20.14	37.17	54.86	64.63	61.72
IOWA by	46.84		35.22	28.45	57.71	47.03	52.58
McGill by	44.70	43.53		48.23	54.12	69.41	52.82
Prosonus by	26.33	46.18	26.58		51.40	51.91	48.47
Vitus by	48.34	42.07	30.12	47.97		69.00	68.63

Families classification	SOL	IOWA	McGill	Prosonus	Vitus	**Minus-1**	**Minus-1** No S-T
SOL by		44.60	56.42	61.67	56.50	62.13	64.02
IOWA by	75.03		74.13	66.06	70.52	86.14	78.89
McGill by	80.00	74.12		74.12	84.70	91.76	84.70
Prosonus by	77.86	58.40	74.04		55.34	84.35	77.48
Vitus by	67.90	59.78	71.95	69.74		79.70	85.98

Pizzicato classification	SOL	IOWA	**Minus-1**	**Minus-1** No S-T
SOL by		98.18	98.03	97.73
IOWA by	96.68		97.05	100
McGill by	100	98.82	98.82	100
Prosonus by	99.62	98.47	100	99.62
Vitus by	98.15	94.09	97.78	100

Questions about the article:

- If I write that SOL classifies IOWA badly and that "we need a lot of new DB's for a "true instrument classification", otherwise it is just a classification of specific samples in a specific database, having no relation to the real qualities of the musical instruments" - isn't that as if I say that all my nice classification results (e.g. 95%) are actually meaningless?
Wouldn't that present my article as worthless? - In all the articles I read about classification of musical instruments, they never mentioned classification of one database by another (although perhaps they should have!).

- About the same matter - should I mention that the last descriptor left in the BIG database after elimination with DA is now the "correct one", and that when doing that to SOL we were left with the Centroid?. What does that say about my gradual elimination process with SOL?

- And a similar question about elimination of "bad samples" - the fact that I almost don't find any of them in SOL, doesn't that mean that the whole section is actually redundant in the article? ("If you didn't find any, why are you telling us stories?")

I really need your advice on these matters, as I have no experience writing scientific articles.

Mini-Report – 11.03.2003

By Arie Livshin

Gain of Removing Outliers

I was hoping to show that when classifying one database by another, removing outliers ("bad samples") improves the classification results.

I have used the 5 databases in my possession, classifying every one of them by all the rest combined ("Minus-1"). Every such classification was first evaluated using all the samples in the Classifier database, then the Outliers were removed from the Classifier database using the LOO method and the classification was repeated.

A reminder from my report of 03.03.2003:
Removal of bad samples using Leave-One-Out (LOO):
Every sample in the Database is removed in its turn from the database and classified by all the rest. Samples which were misclassified are removed. The process reiterates until all samples are correctly classified.

Results:
The first column is the database which was classified.
"Minus-1" contains the classification results by using all the other databases together as the Classifier.
"Outliers Removed" contains the results of the classification after removing the outliers from the Classifier database. The number in parenthesis is the amount of samples removed.

Instruments classification	**Minus-1**	**Outliers Removed**
SOL by	64.63	65.74 (43)
IOWA by	47.03	49.29 (137)
McGill by	69.41	63.53 (105)
Prosonus by	51.91	51.91 (88)
Vitus by	69.00	67.9 (101)

Families classification	**Minus-1**	**Outliers Removed**
SOL by	62.13	62.21 (47)
IOWA by	86.14	87.47 (77)
McGill by	91.76	90.59 (149)
Prosonus by	84.35	80.53 (137)
Vitus by	79.70	82.66 (146)

We can see that removing the outliers from the classifier databases sometimes improved the classification results and sometimes made them worse. All in all, I believe that all these databases had "real" sound samples which were all classified into the correct groups, so removing the samples which were "far" from the others either impaired the classification results a little or improved them a little, but we did not really gain from the process.
I believe that outlier removal is only useful for noisy databases.

Remark: I did not evaluate the Pizzicato taxonomy as each evaluation takes a very long time to compute, and the results of the other two taxonomies were enough to convince me as to the outcome of the process.

Report - 31.03.2003

By Arie Livshin

In this report I shall compare two methods for classification of musical samples:
In the first method, the descriptors go through Discriminant Analysis and then classified by KNN, as it was done in most of the experiments in the paper submitted to ICMC.
In the second method, the descriptors go through Nonlinear Discriminant Analysis (NLDA) and classified using a neural network.
The idea to use a Neural Network for NLDA came to me from:
Matlab 6.5 help - Neural Network Toolbox - speed and memory comparison - backpropagation.
This help page compares different Neural Network training algorithms for the tasks of function approximation and classification.

Neural Network

The network I used is a Feed Forward Back Propagation neural network with 2 hidden layers ("BPNN2") - each with 100 neurons, 162 inputs (as the number of descriptors) and a number of outputs corresponding to the number of classes in the classification taxonomy. All the neurons use the "tansig" transformation function.
For example: when classifying SOL using the "Musical Instruments" taxonomy, the network has a 162-100-100-16 structure.
The network is trained up to 2000 rounds (epochs) until a Mean Square Error (MSE) of 0.005 is reached.
Before the data is fed into the net, it is normalized using Min-Max normalization. Several experiments showed me that Min-Max produces better results than Z-Score for the current task.

I have tried many algorithms for training the net (I specify them in the "remarks" section at the end of the report) and the one which produced best results in most of the cases (and also in somewhat similar cases in the Matlab Help page mentioned above) was Conjugate Gradient with Powell/Beale Restarts (CGB). Sometimes, in the "less complex" classifications of the Pizzicato and Instrument Groups taxonomies, CGB did not converge (the network couldn't learn the examples). In these cases I used the Resilient Backpropagation (RP) algorithm instead, which is an algorithm that converges quickly and relatively easily, but produces slightly worse results.

The Experiments

As we have seen in my ICMC article, the classification experiments where one database is classified by others are more interesting than when a test and a learning group are chosen out of the same database. If this database is "consistent" and the number of descriptors is large, a successful "self" classification does not necessarily mean that the classification algorithm can generalize and classify correctly new samples of the classified instruments. Therefore, in order to show that BPNN2 can generalize better than LDA, I shall use "Minus-1" classifications - each database will be classified by the other databases put together.
After presenting these results, mostly out of historical reasons, I shall also present the average results of 20 classification rounds using BPNN2, where 66% of the samples of SOL will be used to classify the other 33%, and compare the results to the other algorithms I used in the ICMC paper.

Results - Minus 1

The following tables have appeared in my paper for ICMC, except the column marked "BPNN2". The first five columns show the results of classifying every database by every other database. The "Minus-1" columns show the classification results of classifying a database by all the rest put together. In all the columns except the "BPNN2", the classification is done by first performing LDA on the data and then classifying it using the KNN algorithm. In the "BPNN2" column, the data was only normalized before fed to the net (naturally, no LDA was performed).

Instruments classification	SOL LDA ⇨ KNN	IOWA LDA ⇨ KNN	McGill LDA ⇨ KNN	Prosonus LDA ⇨ KNN	Vitus LDA ⇨ KNN	**Minus-1** LDA ⇨ KNN	**Minus-1** BPNN2 **(!)**
SOL by		42.44	20.14	37.17	54.86	64.63	**73.79**
IOWA by	46.84		35.22	28.45	57.71	47.03	**63.22**
McGill by	44.70	43.53		48.23	54.12	69.41	**72.94**
Prosonus by	26.33	46.18	26.58		51.40	51.91	**61.83**
Vitus by	48.34	42.07	30.12	47.97		69.00	**80.81**

Families classification	SOL LDA ⇨ KNN	IOWA LDA ⇨ KNN	McGill LDA ⇨ KNN	Prosonus LDA ⇨ KNN	Vitus LDA ⇨ KNN	**Minus-1** LDA ⇨ KNN	**Minus-1** BPNN2
SOL by		44.60	56.42	61.67	56.50	62.13	**61.75**
IOWA by	75.03		74.13	66.06	70.52	86.14	**88.89**
McGill by	80.00	74.12		74.12	84.70	91.76	**87.06**
Prosonus by	77.86	58.40	74.04		55.34	84.35	**88.17**
Vitus by	67.90	59.78	71.95	69.74		79.70	**91.14**

Pizzicato classification	SOL LDA ⇨ KNN	IOWA LDA ⇨ KNN	**Minus-1** LDA ⇨ KNN	**Minus-1** BPNN2
SOL by		98.18	98.03	**96.9**
IOWA by	96.68		97.05	**98.73**
McGill by	100	98.82	98.82	**100**
Prosonus by	99.62	98.47	100	**99.24**
Vitus by	98.15	94.09	97.78	**99.63**

Results - SOL

The following table, which also appeared in my paper to ICMC, except the "BPNN2" column, shows the results of 20 classification rounds. In each round a learning group of 66% of the samples was randomly selected out of SOL and used to classify the other 33%. Before every classification algorithm except BPNN2, the data was processed by LDA.

Algorithm ➢ **⮟ Taxonomy**	Gauss	K-NN	LVQ	**BPNN2**
Pizzicato/Sustain	99.58%	99.97%	99.93%	**99.91%**
Instrument Families	94.91%	95.24%	95.22%	**97.66%**
Instruments	94.12%	95.85%	88.57%	**95.34%**

Discussion of the results

We can see that there is real improvement when using NLDA instead of LDA in the Musical Instruments taxonomy in the Minus-1 classifications, where the classification task is the most complex.
In the simpler taxonomies (Pizzicato and Instrument Families), as well as in the "SOL-only" classifications, the difference between NLDA and LDA results is not so radical (although it is worth to compare BPNN2 to KNN in the "SOL-only" experiments, where BPNN2 has improved the results of the Instrument Families taxonomy classification and "moved it up" between the Pizzicato and Musical Instruments, where it was somehow expected to be).
It is also interesting to note that in quite a few cases BPNN2 produced worse results than LDA (which is supposed to be a private case of NLDA). This shows that our NLDA algorithm is not perfect or at least that some extra tuning is needed to adapt it to each classification type.

Conclusions

The first table has clearly shown us that using NLDA (BPNN2 in our case) improves significantly the results of "Minus-1" classifications in the Musical Instruments taxonomy. The Minus-1 experiments resemble "real-world" classification problems, where we need to classify new samples using the databases we already have.
This table has convinced me that LDA does not generalize "good enough" for classification of samples of musical instruments, and that NLDA is preferred.

Various Remarks

- In order to decide which algorithm works best for training the neural network, I tried many algorithms before choosing Conjugate Gradient with Powell/Beale Restarts and Resilient Backpropagation. Each of the following algorithms was tested through all the Minus-1 experiments performed in this report:
Levenberg-Marquardt, Quasi-Newton, Resilient Backpropagation, Scaled Conjugate Gradient, Conjugate Gradient with Powell/Beale Restarts, Fletcher-Powell Conjugate Gradient, Polak-Ribiére Conjugate Gradient, One-Step Secant, Variable Learning Rate Backpropagation.

- In contrast with Learning Vector Quantization (LVQ) neural networks that I used in one section of the article for ICMC, which classify each sample strictly into a single class, each output neuron of a Back Propagation (BP) neural network produces some value. This means that we get a sort of a priority list for the classification of the samples.
During this report I have used the output neuron with the maximum value to classify the sample, but it is also interesting to note the other output neurons - how much is the net "confident" about a classification? What is the second group in the priority list the net would classify a sample to? etc.

- In the "Minus-1" experiments, when KNN was used, each classification had to be performed only once as there was no randomization either in choosing the data or in the classification algorithm. On the other hand, using a neural network produces small variations in the results due to choosing random initial weights for the network each time before the training begins. Although it is possible to conduct a great number of experiments and produce confidence intervals for the results, but as each training of the net consumes time, I didn't bother with that. In have repeated several experiments and the results varied by about 0.5%.

- Regarding the removing of outliers, my initial guess is that the best algorithm in the case of BPNN2, out of the algorithms I used in the ICMC paper, would be the Modified Interquantile Range (MIQR) - a supervised version of IQR.
IQR is in disadvantage as it does not use the classification of the learning database, although it is known.
The problem with Leave-One-Out (LOO) is that if we use KNN and LDA (like in the paper) for finding and removing misclassified samples in the learning database, we obviously miss the point of NLDA. On the other hand, if we try to use BPNN2 in the LOO, we will have to train the net all over again for every sample in the database - a process which is very expensive.

- The following article also deals with LDA vs. NLDA - it compares the results of using PCA, ICA, LDA and NLDA for phone classification. Note that the NLDA algorithm is somewhat different than the one I use - the author uses a simpler neural network (MLP with one hidden layer), and the differences in results are much less dramatic than in our case.

Somervuo, Panu. 2003. "EXPERIMENTS WITH LINEAR AND NONLINEAR FEATURE TRANSFORMATIONS IN HMM BASED PHONE RECOGNITION". *icassp2003*.
URL: www.icsi.berkeley.edu/ftp/global/pub/ speech/papers/icassp03-somervuo.pdf

- Perhaps I could write for ISMIR 2003 or WASPAA 2003 (both have a deadline of 25.4) a short article (4 pages) in which I describe the classification process used in our ICMC article, move to show that classifying with several databases improves generalization and that picking a test group and a learning group out of the same database is not a good indication to the strength of a classification algorithm, and then show that LDA is not good enough for such classification and that NLDA is better.
The sections from the ICMC article which will be removed are the Gradual Descriptor Elimination (I did not work yet on doing it with NLDA, but from first looks it seems much less straightforward than with LDA), and the part about removing outliers.

Mini-Report - 17.04.2003

By Arie Livshin

In my previous report I have shown using Minus-1 experiments, that a back-propagation neural network, a net which can do nonlinear discriminant analysis, classifies musical instrument samples considerably better than LDA+KNN (or LDA+Gauss, or LDA+LVQ). There was a discussion about this, where it was suggested that such a big network (I used a network with 2 hidden layers each sporting 100 neurons) might just learn the classification of the learning group "by heart" and not generalize at all (total overfitting). The samples of the test group resemble so much the learned samples, that the net simply(?) performs a kind of KNN for each test sample and returns the class of the learned sample which most resembles it.

I shall try to deal to address this claim in 2 stages:
1. I have managed to decrease the network size from 2 hidden layers with 100 neurons each to a single hidden layer with 80 neurons without damaging the results much. In fact, most of the results even improved a bit.
The following tables show the classification results of each DB by every other one using LDA and KNN, of each DB by all the rest put together ("Minus-1") using LDA and KNN, Minus-1 using the 2-hidden layer net (BPNN2) and Minus-1 using the 1-hidden layer net (BPNN1).
Just for reference - The table showing the classification results of the "musical instruments" taxonomy also includes a column which shows the results of first applying LDA and then classifying using theBPNN2[1] network. It is obvious that the results of such classification would not be as good as classifying using only the Neural Network, yet we can see that the results are still considerably better than when using LDA and KNN. This technique might be useful (although I doubt it) to save storage space (e.g. 15 descriptors per sample instead of 162 for a 16 classes classification) and to save NN training time[2], compromising the results yet still getting better recognition rate than LDA+KNN (or Gauss, or LVQ).

Instruments classification	SOL LDA ⇨ KNN	IOWA LDA ⇨ KNN	McGill LDA ⇨ KNN	Prosonus LDA ⇨ KNN	Vitus LDA ⇨ KNN	**Minus-1** LDA ⇨ KNN	**Minus-1** BPNN2 (100)	**Minus-1** LDA ⇨ BPNN2 (100)	**Minus-1** **BPNN1 (80)**
SOL by		42.44	20.14	37.17	54.86	64.63	73.79	66.43	**75.59**
IOWA by	46.84		35.22	28.45	57.71	47.03	63.22	57.35	**65.22**
McGill by	44.70	43.53		48.23	54.12	69.41	72.94	74.12	**75.29**
Prosonus by	26.33	46.18	26.58		51.40	51.91	61.83	62.21	**58.01**
Vitus by	48.34	42.07	30.12	47.97		69.00	80.81	69.74	**74.9**

[1] - The results with BPNN1 should be similar. I made this experiment before I changed to BPNN1, and I didn't want to repeat with BPNN1 it as it consumes time and does not seem really important at the moment.

[2] Storage space depends on the application. If the net is to be trained only once, unlike KNN, it does not require the learning data any more and the data could be stored separately. Same is true about reducing the training time - it might be a single time affair.

Families classification	SOL LDA ⇨ KNN	IOWA LDA ⇨ KNN	McGill LDA ⇨ KNN	Prosonus LDA ⇨ KNN	Vitus LDA ⇨ KNN	**Minus-1** LDA ⇨ KNN	**Minus-1 BPNN2 (100)**	**Minus-1 BPNN1 (80)**
SOL by		44.60	56.42	61.67	56.50	62.13	61.75	**62.21**
IOWA by	75.03		74.13	66.06	70.52	86.14	88.89	**87.98**
McGill by	80.00	74.12		74.12	84.70	91.76	87.06	**95.29**
Prosonus by	77.86	58.40	74.04		55.34	84.35	88.17	**89.31**
Vitus by	67.90	59.78	71.95	69.74		79.70	91.14	**91.51**

Pizzicato classification	SOL LDA ⇨ KNN	IOWA LDA ⇨ KNN	**Minus-1** LDA ⇨ KNN	**Minus-1 BPNN2 (100)**	**Minus-1 BPNN1 (80)**
SOL by		98.18	98.03	96.9	**98.11**
IOWA by	96.68		97.05	98.73	**97.62**
McGill by	100	98.82	98.82	100	**100**
Prosonus by	99.62	98.47	100	99.24	**99.62**
Vitus by	98.15	94.09	97.78	99.63	**100**

The following table shows the "self classification" of SOL - the mean results of 20 experiments in which a random learning group of 66% of the samples was selected.

Algorithm ➢ **➢ Taxonomy**	Gauss	K-NN	LVQ	**BPNN2 (100)**	**BPNN1 (80)**
Pizzicato/Sustain	99.58%	99.97%	99.93%	99.91%	**99.88%**
Instrument Families	94.91%	95.24%	95.22%	97.66%	**97.77%**
Instruments	94.12%	95.85%	88.57%	95.34%	**95.6%**

We see that the results of BPNN1 in the Minus-1 classifications are usually slightly better than of BPNN2 and considerably better than LDA+KNN. In the case of SOL, both of the BPNN have made the classification results more "normal" - the instrument families taxonomy results have found their "right place" in between the Pizzicato and Instruments taxonomies (unlike KNN).
We can also see that both of the BPNN are not ideal - it (rarely) happens that one network or the other gets much better results (see Vitus in Instruments and McGill in the Families). We still don't get the maximum out of the data.

2. After diminishing the network:
In order to show that the net cannot learn "by heart" so many classifications I have trained it on normalized random data , the same size of the original learning database, and the network indeed couldn't converge to the desired Mean Square Error.
Axel has proposed a more strict experiment:
"How about this test. You take your descriptors as input, and assign a random class out of N target classes to each of them (N being the number of classes you have in your real application). You train with different randomized training classes. If the training error is in the same order than for the real classes the network can learn by heart because the result does not degrade when the relation between input descriptors and classes is destroied.

However, you will not be able to test the generalization performance because by construction you destroy all neighboring relations between the training examples.

If their is a difference in training error then this shows that the examples are at least not learned independently. In this case groups of input examples would be somehow combined which is already quite good."

I have trained the 80 neuron-one hidden layer network on the Minus-1 data of mcGill - 4296 samples, with the stopping conditions of 4000 epochs (training rounds) or a Mean Square Error (MSE) of 0.004, using the Conjugate Gradient with Powell/Beale Restarts (trainCGB) and Resilient Backpropagation (trainRP) learning algorithms.

<u>Using the original classes:</u>
TRAINCGB-srchcha, Epoch 600/4000, MSE 0.00399468/0.0005, Gradient 0.00677285/1e-006

TRAINRP, Epoch 1850/4000, MSE 0.00399676/0.0005, Gradient 0.00283508/1e-006

Both of the algorithms learned and converged (reached the desired MSE).

<u>Using random classification:</u>
TRAINCGB-srchcha, Epoch 50/4000, MSE 0.0497887/0.0005, Gradient 0.000879575/1e-006
TRAINCGB, Minimum step size reached, performance goal was not met.

TRAINRP, Epoch 4000/4000, MSE 0.0418077/0.0005, Gradient 0.00132736/1e-006

Both of the algorithms did not converge. TrainCGB just started learning and already the learning curve has totally straightened (it couldn't learn anymore).
Trainrp also learned a bit, and then the training curve almost straightened and remained that way until the maximum epoch was reached.
Both managed to reach only 10 times the desired MSE of 0.004.

This test shows that the classification makes a difference and that the net does not just learn the classes "by heart".

(apropos)
It is possible to repeat the experiment several times with different random classifications, and with the minus-1 learning databases resulting from omitting every database, not just McGill, but I hope the point that the net does generalize is already quite clear (otherwise, waiting 4000 epochs for every experiment takes time). Besides, I feel that repeating the test for every database is not really needed in the case of minus-1 classifications. If the net generalizes samples that resemble each other, then in other tests in which other databases are removed, a single database will be put out of the learning set and mcGill will be put in. As we already know from the classification results, McGill resembles samples which are in the learning set - so McGill could be partially generalized along with the samples which resemble it, and the other databases could be generalized, as they still resemble each other like in the mcGill case - unless a single database

resembles all the rest and the rest are independent (then one of the minus-1 experiments will fail), all the classification rounds would generalize in some way.

Report - 19.5.2003

By Arie Livshin

Algorithms for musical instrument recognition in solo recordings

Algorithm #1 - Reduction to the problem of classification of separate sounds

i. Find places in the solo piece where notes start.
The simplest case is to use silence detection. The obvious disadvantage is that we shall get only a small number of notes.
My experiments with Ircam's "silence detector" have shown that almost any solo recording requires its own thresholds for successful silence detection (such as silence velocity, minimum duration and modulation amplitude). It might be required to build a "smart" silence detection program which will analyze the sound file and attempt to compute the right thresholds itself.

ii. Search for the places where the selected notes end.
I believe a good way to do that could be by finding where f0 changes by more than a quarter of a tone. Adding to that, a strong outburst of energy could signify the attack of the next note (even if it is of the same pitch).
If (i) and (ii) produce only sounds which are too short ("too short" is yet to be approximated) the program could move to a different algorithm, like the algorithm in the next section ("Blind Classification").
Axel has proposed that instead of searching for the place where each note ends, we could take only the minimum length which is required for good classification. In order to approximate this length, we could repeat the tests I did with classification of separate sounds, but this time shorten each sample to a tested minimal length and see if we get good classification results.

iii. Compute the descriptors using the resulting samples.

iv. Use the same databases (SOL, IOWA, etc.) and the same classification techniques which were used for separate sound recognition to classify the new samples.

v. The solo pieces are classified as the most common classification of their samples (e.g. if 40% of the samples in a melody were classified as "Flute", 35% as "oboe" and 25% as "bassoon", the solo piece would be classified as "Flute").

Algorithm #2 - "Blind Classification"[1]

i. Solo pieces of different instruments are cut into chunks of equal length by using a sliding window.

ii. The descriptors are calculated using these chunks.
As these chunks don't have a well defined evolution (these are not separate notes), it is possible to get rid in advance of non-relevant descriptors (like "attack time"). Another option is to trust the classification process to disregard irrelevant descriptors automatically (like LDA or NN for example).

[1] By "blind" I mean that the samples are "blindly" cut out of the solo pieces, without regard to the starting/ending times of the notes. Long silences in the solo are better avoided, though.

iii. The classification algorithm learns the descriptors of the solo pieces in the learning group and classifies the ones in the test group.

iv. The solo pieces are classified as the most common classification of their samples.

<u>Algorithm #3 - Classification using the "evolution" of the sounds</u>
i. The evolution turn points of separate notes (either in sample files of discrete notes or in solo pieces) of every instrument we wish to learn are manually marked - the locations where the attack begins, the attack ends and the sustain part ends. These marked notes are learned by a separate Markov model for each instrument.

ii. A sliding window is moved across the solo piece. The HMM which resembles best the contents of each window is found.

iii. The solo is classified as the most common HMM for this piece.

This technique is described in:
Casey, M., "Generalized Sound Classification and Similarity in MPEG-7", Organized Sound, 6:2, Cambridge University Press, 2002.

Geoffroy suggested that perhaps it is possible to use neural nets which can study sequences, instead of HMMs. Such neural networks exist and are called "Recurrent networks" - for example the Elman and Hopfield networks.

<u>Remark</u> - it is possible to report the "certainty level" of the classification algorithm regarding a solo piece by using the neural network's output weights, or the similarity level to the various HMMs.

Moving to a new descriptor module

Geoffroy has given me the code for his improved module for the computation of descriptors. This module provides besides the mean value of every descriptor, also the standard deviation and the descriptor values in every time frame. The module includes several new descriptors.
I believe this module could improve my classification results as well as provide important information about the evolution of the sounds.

Initial Experiments

Disclaimer - The following experiments are only preliminary and superficial. They do not really reflect which classification method is preferred or the expected recognition rate. The number of samples is very small and the classification techniques are just caricatures of the full processes. All this said, I still think the results are quite interesting and that is why I bring them here.

In all the experiments I have used a back-propagation network with 80 hidden neurons in a single hidden layer ("BP80"). In every experiment the descriptor matrix was normalized and the neural net retrained.

<u>**Classification using Algorithm #1 - Reduction to the problem of classifying separate sounds**</u>

The "BP80" neural network was trained on the same sound database I have used for classification of separate sounds. This database contains samples of the 7 instruments common to the 5 databases I used (McGill, SOL, etc). Each sample is 2 seconds long. The sampled instruments are: Bassoon, Contrabass, Clarinet, French horn, Flute, Oboe and Cello.

Only 3 solo pieces were classified. The beginning of each note was found by the Silence Detector program, and the length of the note was adjusted manually. The instruments are Clarinet, Bassoon and Flute.

The descriptor matrix (162 parameters for each sound) was calculated using the old descriptor module, as I did not have all the sample files of the learning group at the time (they consume a LOT of disk space) - just the descriptor matrix.

<u>1'st experiment results (test group consisting of only Flute and Bassoon sounds):</u>

```
Total good : 7/11. This is 63.636364 Percent.
STATISTER_START
Class Bassoon (French): Correct: 20.000000 percent, common mistake (40.000000 percent):Flute
Class Flute: Correct: 100.000000 percent
STATISTER_END
```

<u>2'nd experiment results (test group consisting of Flute, Bassoon and Clarinet):</u>

```
Total good : 4/22. This is 18.181818 Percent.
STATISTER_START
Class Bassoon (French): Correct: 20.000000 percent, common mistake (80.000000 percent):Cello (Bowed)
Class Clarinet (Bb): Correct: 0.000000 percent, common mistake (81.818182 percent):Cello (Bowed)
Class Flute: Correct: 50.000000 percent, common mistake (50.000000 percent):Cello (Bowed)
STATISTER_END
```

These experiments produced very bad results. The net just classified the entire test group into the same class.

It is important to note that the net was trained on 2 second sounds while the classified samples were much shorter, starting from mere 260 msec.

I suspect the size of the samples might prove to be important. I think it is worthy to try to decide on a "standard" sample length (testing it as Axel suggested in the description of "algorithm #1") and to remove from the test group samples which are too short. If too few samples are left in the test group then a different classification algorithm will be used. The samples in the training group should be cut down to this same "standard" size.

Classification using Algorithm #2 - "Blind classification"

We shall start by using 8 solo pieces of different instruments, each cut into 60 windows of 1 second with a sliding distance of ½ second.
The instruments are: Accordion (polyphonic solo piece), Cembalo (polyphonic), Classical guitar (polyphonic), Church organ (polyphonic), Piano (polyphonic), Violin (somewhat polyphonic), Clarinet (monophonic) and Flute (monophonic).

The descriptors were computed using the new descriptor module. The mean and the standard deviation of every descriptor were used. This resulted in a total of 480 parameters per sample.

Self classification:
First, in order to see that there is enough information for class separation[2] I performed a 66%/33% classification of the samples of the 8 instruments (which are to be used later as the learning group). The result was 98.75% success.
As the cutting windows were half overlapping (window size 1 sec., sliding size ½ sec.) it meant that almost the entire sampled section appeared twice. I repeated the experiment, this time using only the odd numbered windows (30 samples of each instrument) without any overlapping.

Results:
```
Total good : 78/80. This is 97.500000 Percent.
STATISTER_START
Class Accordion: Correct: 100.000000 percent
Class Clarinet (Bb): Correct: 100.000000 percent
Class Flute: Correct: 90.000000 percent, common mistake (10.000000 percent):Accordion
Class Guitar: Correct: 100.000000 percent
Class Harpsichord: Correct: 100.000000 percent
Class Organ: Correct: 100.000000 percent
Class Piano: Correct: 100.000000 percent
Class Violin: Correct: 90.000000 percent, common mistake (10.000000 percent):Flute
STATISTER_END
```

We can see that self classification produced very good results, meaning that there is good class separation in this intended learning group.

Next, I have used this group to classify a different test group containing samples from solo pieces which were not learned.
This test group consists of 4 solo pieces which are quite different from the ones used in the leaning group:
Guitar - the learned piece is classical; the test piece is flamenco.
Piano - the learned piece sounds "clean"; the test piece has a lot of reverb and sounds like an electric piano.

[2] As the samples were "blindly cut" I wasn't sure that the classes were separated, although 480 parameters for each sample are quite a lot.

Clarinet - both pieces are by the same performer, but are taken from different CD's. The learned piece was recorded with an extremely low volume (that whole CD was very badly recorded and mixed). The test piece is fine.
Violin - old recordings (noisy) by different performers.

<u>Results:</u>

```
Total good : 76/114. This is 66.666667 Percent.
STATISTER_START
Class Clarinet (Bb): Correct: 0.000000 percent, common mistake (37.500000 percent):Violin
Class Guitar: Correct: 100.000000 percent
Class Piano: Correct: 66.666667 percent, common mistake (33.333333 percent):Guitar
Class Violin: Correct: 86.666667 percent, common mistake (10.000000 percent):Accordion
STATISTER_END
```

We see that 3 out of the 4 solo pieces were correctly classified.
I feel that these results are not bad. We need to remember that only a single example of every instrument was learned and that the learning group was quite different from the test group[3]. The learned solo of the clarinet for instance, which was the solo that was incorrectly classified, as already mentioned, was very badly recorded while the test solo was recorded fine.
On the other hand, we should note that most of the 8 instruments used were very different from each other. I did not try to distinguish a violin from a viola, for example.

To get a good feeling of the algorithm, I should perform many "Minus-1" experiments with lots of samples from each instrument as well as include instruments that resemble each other more.

I have not tested Algorithm #3 - "Sound evolution" yet.

<u>Remark regarding normalization</u>
Retraining of the neural network each time new test samples arrive consumes much time and should be avoided when possible.
Let's say that I am interested in a Min-Max normalization exactly between 0 and 1. What I do for now, is to keep along with the trained neural network the extreme values of the descriptors used for its training (before they were normalized of course), and if the descriptors of the test data lie between these values I use these extreme values to normalize the test data. In these cases there is no need for retraining of the network.
The problem is that many times at least one of the descriptors in the test data is out of the range of the "extreme" values in the learned group. In these cases I perform the normalization of all the data again and retrain the net.

Is there some mathematical way of getting over this time consuming problem?

Acknowledgement:
I wish to thank Geoffroy Peeters who discussed with me various ways for instrument recognition in solo pieces and directed me to Michael Casey's work.

[3] Even when we classified separate sounds, which is supposed to be a simpler task than solo pieces, classification by a single database produced quite bad results. Classifying SOL by IOWA for example, produced only 40% success.

Report - 5.6.2003

By Arie Livshin

In this report I addressed instrument classification of Solo music using what I call "Blind Classification".

"Blind Classification"[1] algorithm

i. Solo pieces of different instruments are cut into chunks of equal length by using a sliding window.
ii. The descriptors are calculated using these chunks.
iii. The classification algorithm learns the descriptors of the solo pieces in the learning group and classifies the ones in the test group.

Parameters

Every Solo piece was cut into windows of 1 second, overlapping by ½ second. The maximum number of windows taken from one piece is 60 (if the piece is 30.5 seconds or longer). Shorter pieces were cut into fewer windows.

Different goals

- "Window Recognition" - the solo piece is cut into partially overlapping windows of equal size. Our (ultimate) goal is to classify each one of these windows correctly. We do not assume that all of the windows are played by the same instrument, but we do assume that each window contains a sample of a single instrument.

- "Solo recognition" - we assume that the whole musical piece is played by the same instrument and want to find out by which. We don't care about the classification of partial pieces (e.g. windows).

The descriptors

I have used Geoffroy's new descriptor module for computing the descriptors for every window of a solo piece. The module calculates most of the descriptors for every 60ms frame of a sample - I have used the mean and standard deviation of these descriptors (like Geoffroy recommended).
The total number of parameters per window (sample) is: 513. The old module (which I used in my previous reports and papers) produced 162 parameters per sample.
In appendix A I shall compare the classification results of separate sounds using the old and the new descriptor modules.

The solo pieces

The solo pieces were collected from various sources. As they are very hard to find, I had to compromise on the quality - some of the solo pieces are very short (starting from 4 seconds) [2], low quality or compressed with a lossy compression scheme.
The low quality and short length of the Solos must have affected the results considerably.

---

[1] By "blind" I mean that the samples are "blindly" cut out of the solo pieces, without regard to the starting/ending times of the notes. Long silences in the solo are better avoided, though.

[2] In my Minus-1 DB classification this means that sometimes only 4 seconds of an instrument solo were learned, and then used to classify other solos - quite far from optimal learning.

The musical instruments are: Bassoon, Cello, Clarinet, Flute, Guitar, Piano and Violin. For a full description of the Solo pieces, see appendix B.

Solo processing - Solo pieces compressed or recorded in a low frequency were resampled and saved as non-compressed 44Khz WAV files. Only the right channel of Solos which were recorded in stereo was used.
Silences of more than 1 second were manually removed. I might do it automatically in the future.

Experiment #1: Minus-1 DB
I have divided the Solos into 5 groups. Every group has only one solo of each musical instrument. Unfortunately, I did not have enough solos to include all the 7 instruments in all the groups. Each of the groups was classified by the rest joined together.

I have compared the results of 3 classification algorithms:
LDA+KNN and 2 neural networks: BP80 (back propagation with 80 hidden neurons) that learns using the CGB method (Conjugate Gradient with Powell/Beale Restarts) and a BP80 which learns using the RP method (Resilient Backpropagation).

The results are present in the table below.
Every cell contains the results of classifying a group by the other groups put together.
The Window Classification results of every instrument in the group are shown, as well as the mean Window Classification results of the whole group of instruments and the number of solos correctly classified (a solo is classified using the most common classification of its windows).

For example - if we look at the first cell of LDA+KNN, we see that:
13.33% of the windows of the "Bassoon DAT" solo were classified as Bassoon. Looking at the bottom line in the cell, we see that the most common classification of 5 solos out of 7 in the group was actually the correct instrument. 64.58% of all the windows in the group were classified correctly.

The bottom row in the table shows the instrument names, each with its number of solos in parenthesis. The mean number of solos correctly classified for each instrument is given in percents as well as the mean percentage of windows correctly classified for each instrument ("window classification").
The total mean of all the groups together for each classification algorithm is shown.

For example - if we look at the last cell of LDA+KNN, we see that:
There were 3 Basson solos. 33% of them were correctly classified (1 solo), and 66.78% of the Bassoon windows were classified as Bassoon. The mean classification of solos using LDA+KNN gave 79.52% success, and the mean classification of windows gave 66.78%/

In the CGB80 column (which will turn out to be the best), if an instrument gets less than 50% in Window classification, I show in parenthesis the most common mistake.

Minus-1 DB	LDA+KNN	BP80 CGB	BP80 RP
Bassoon DAT	13.33	80	
Cello Emanuel	60	71.67	
Clarinet amazing	87.5	79.17	
Flute Bach	71.67	81.67	
Guitar baldi	100	100	
Piano moon1	33.3	45 (guitar)	
Violin kremer	100	100	
Mean	5/7 64.58	6/7 79.69	6/7 73.44
Bassoon berio	36.67	43.33 (piano)	
Cello italian	42.85	100	
Clarinet_hatikva	21.67	3.33 (flute)	
Flute italian	65.38	53.85	
Guitar kayath	86.67	76.67	
Piano thepiano	51.67	90	
Violin menuhin	88.33	90	
Mean	4/7 57.36	5/7 60.96	5/7 52.85
Cello berlin **BY**	56.76	2.7 (clarinet)	
Clarinet emanuel	86.67	30 (flute)	
Flute berio	21.67	63.33	
Guitar italian	72.73	54.54	
Piano emanuel	86.67	68.33	
Violin italian	100	78.95	
Mean	5/6 66.8	4/6 48.18	4/6 51.42
Bassoon_vitus	65	75	
Cello_academy**BY**	86.36	81.82	
Clarinet italian	73.68	36.84 (violin)	
Flute ircam	95	83.33	
Guitar berio	48.33 (26.67)	75	
Piano italian	0	0 (cello)	
Violin academy	92	100	
Mean	6/7 67.29	5/7 70.63	6/7 62.45
Clarinet Berio **BY**	70	85	
Piano academy	48.27 (27.59)	75.86	
Violin Berio	100	100	
Mean	3/3 77.85	3/3 89.26	2/3 71.14
Mean of all instruments	Solos Windows 79.52 66.78	Solos Windows 79.05 69.74	Solos Windows 75.24 62.26
Bassoon (3)	33 38.33	66 66.11	
Cello (4)	75 61.49	75 64.04	
Clarinet (5)	80 67.9	40 46.87	
Flute (4)	75 63.43	100 70.54	
Guitar (4)	100 76.93	100 76.55	
Piano (5)	60 43.98	60 55.84	
Violin (5)	100 96.07	100 93.79	

By examining the bottom row, we see that the best algorithm for Window Classification was BP80 CGB, with an average recognition rate of 69.74%.
LDA+KNN did slightly better with Solo Classification (by the way - KNN was about 20 times faster than the neural nets as it did not require learning. On the other hand, it consumes much more memory).
The RP network was the worst in all the group classifications, so I did not bother to write its results for every solo and every instrument.

<u>Experiment #2 - confidence level</u>
When a BP network classifies, each one of the output produces some value. This value is the probability (in the network's "opinion") that the classified data belongs to that class. In a previous report I have shown that when classifying separate notes, the different outputs of the net could help us predict whether the classification is correct (how much is the net "sure" of the classification) - usually, when the net had a mistake, the difference in the output values was not very big - there wasn't one output neuron with a value which was much higher than the rest.
I tried to use the different output values of the network for improving Solo Classification.

Algorithm:
1. When classifying a window with the network, keep along with the classification result also its "Confidence level" (explanation follows below).
2. Sort the results according to their confidence level.
3. Remove some of the results with the lowest confidence level.
4. Classify the Solo using the most common classification of the remaining windows.

In this experiment I used 2 different confidence levels:
"Max among outputs" - the highest probability the net produced for the window.
"Max confidence ratio" - the highest output divided by the second highest output.

I shall use BP80 CGB Minus-1 DB classifications.

The results are presented below.
As I did not want to guess which percentage of the windows with the lowest confidence level to throw out, I removed repeatedly the window with the lowest confidence level. If most of the remaining windows were classified correctly, I have displayed the 'Y' letter, otherwise '.'
For example, if we examine the 4'th row of Group 1 (see below) with the Confidence Level "Max among outputs", we can see that the Piano was not classified correctly when all the windows were used (the first symbol is '.'). After removing 13 of the windows with the lowest confidence level, it was classified correctly during another 7 removals of windows, then incorrectly for another 3, etc.

```
Group 1:
MAX AMONG OUTPUTS
Bassoon (French) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Clarinet (Bb) -> YYYYYYYYYYYYYYYYYYYYYYYY
Flute -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Guitar (Classic?) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Piano -> .............YYYYYYY...Y.Y.....YYYYYYYYYYYYYYYYYYYYYYYYY.YYY
Cello (Bowed) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Violin -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
```

```
MAX CONFIDENCE RATIO
Bassoon (French) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Clarinet (Bb) -> YYYYYYYYYYYYYYYYYYYYYYYY
Flute -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Guitar (Classic?) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Piano -> ...................Y.Y.................................Y.YYY
Cello (Bowed) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Violin -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
```

Group 2:

```
MAX AMONG OUTPUTS
Bassoon (French) -> ............................................................
Clarinet (Bb) -> ............................................................
Flute -> ...Y.YYYYYYYYYYYYYYY.Y.Y.Y
Guitar (Classic?) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Piano -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Cello (Bowed) -> YYYYYYY
Violin -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY

MAX CONFIDENCE RATIO
Bassoon (French) -> ............................................................
Clarinet (Bb) -> ............................................................
Flute -> .Y.Y...........YYYYYYYYYYY
Guitar (Classic?) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Piano -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Cello (Bowed) -> YYYYYYY
Violin -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
```

Group 3:

```
MAX AMONG OUTPUTS
Clarinet (Bb) -> ................................YYYYY.YYYYYYYYYYYYYYYYYYYY..
Flute -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY...YYY.Y
Guitar (Classic?) -> Y.Y.YYY....
Piano -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY.....................Y
Cello (Bowed) -> .....................................
Violin -> YYYYYYYYYYYYY......

MAX CONFIDENCE RATIO
Clarinet (Bb) -> ...................................................Y.YYY....
Flute -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Guitar (Classic?) -> YYYYYYYYYYY
Piano -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Cello (Bowed) -> .....................................
Violin -> YYYYYYYYYYYYYYY.Y..
```

Group 4:

```
MAX AMONG OUTPUTS
Bassoon (French) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Clarinet (Bb) -> ...................
Flute -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Guitar (Classic?) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY....
Piano -> .......................
Cello (Bowed) -> YYY.YYYY.YYYYYYYYYYYYY
Violin -> YYYYYYYYYYYYYYYYYYYYYYYYY

MAX CONFIDENCE RATIO
Bassoon (French) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Clarinet (Bb) -> ..................Y
Flute -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY.Y
Guitar (Classic?) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY..
Piano -> .......................
Cello (Bowed) -> YYYYYYYYYYYYYYYYYYYYYY
Violin -> YYYYYYYYYYYYYYYYYYYYYYYYY
```

Group 5:

```
MAX AMONG OUTPUTS
Clarinet (Bb) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Piano -> YYYYYYYYYYYYYYYYY............
Violin -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY

MAX CONFIDENCE RATIO
Clarinet (Bb) -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
Piano -> YYYYYYYYYYYYYYYYY.Y..........
Violin -> YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
```

If we examine the second half of the results of each Solo and compare it to the classification using all the windows (which is the first symbol in every row), we see that:
Max among outputs - improved 3 Solo classifications, ruined 4.
Max confidence ratio - improved 1, ruined 1.

Conclusion: we see that removing windows classified with low confidence level did not improve Solo classification results.

Remark #1: While the number of solos in my experiments became higher, this method worked better and better (when I started using it and had only very few solos, its results were quite chaotic). It is possible that when (and if) I will have much larger databases, this method will start actually improving Solo Classifications.

Remark #2: The results in this experiment should not be directly compared with the results in experiment #1. The reason is simple - this is not the same experiment. The initialization of the neural net uses random values, so the results are slightly different each time, although it is the same classification algorithm and the same data in both experiments.

<u>Experiment #3 - Mutual Classification</u>

In this experiment I shall classify each groups by every other one (separately). The results will be compared with Minus-1 DB classification of the groups.

Each row in the following table shows the classification of the same group by different ones (Window Classification). When most windows of a classified Solo do not classify correctly, the most common mistake is shown in the parenthesis.

	Bassoon DAT Cello emanuel Clarinet amazing Flute bach Guitar baldi Piano moon1 Violin kremer	Bassoon berio Cello italian Clarinet hatikva Flute italian Guitar kayath Piano_thepiano Violin menuhin	Cello berlin Clarinet emanuel Flute berio Guitar italian Piano emanuel Violin italian	Bassoon Vitus Cello_academy Clarinet italian Flute Ircam Guitar berio Piano italian Violin academy	Clarinet Berio Piano academy Violin Berio	Minus-1 CGB net
Bassoon dat **BY**		45 (Clarinet)		86.67		80
Cello emanuel		1.67 (Guitar)	68.33	48.33 (Guitar)		71.67
Clarinet amazing		4.17 (Violin)	20.83 (Cello)	12.5 (Cello)	91.67	79.17
Flute bach		40 (Violin)	20 (Clarinet)	96.67		81.67
Guitar baldi		100	45 (Cello)	100		100
Piano moon1		56.67	100	0 (Guitar)	13.33 (Clarinet)	45 (guitar)
Violin kremer		100	1.67 (Clarinet)	35	90	100
						6/7 79.69
Bassoon berio **BY**	3.33 (Piano)			98.33		43.33 (piano)
Cello italian	42.86 (Guitar)		57.14	28.57 (Guitar)		100
Clarinet hatikva	0 (Bassoon)		0 (Flute)	5 (Guitar)	95	3.33 (flute)
Flute italian	57.69		7.69 (Clarinet)	23.08 (23.1 bsn)		53.85
Guitar kayath	85		23.33 (Piano)	95		76.67
Piano thepiano	86.67		100	0 (Guitar)	15 (clarinet)	90
Violin menuhin	85		8.33 (Clarinet)	11.67 (flute)	73.33	90
						5/7 60.96
Cello berlin **BY**	13.51 (Clarinet)	24.32 (Violin)		32.43 (27 Flute)		2.7 (clarinet)
Clarinet emanuel	0 (Flute)	0 (Violin)		0 (Flute)	91.67	30 (flute)
Flute berio	26.67 (Violin)	25 (Violin)		80		63.33
Guitar italian	90.91	72.73		36.36 (Cello)		54.54
Piano emanuel	25 (Cello)	96.67		0 (Cello)	40 (Clarinet)	68.33
Violin italian	100	94.74		36.84 (Piano)	94.74	78.95
						4/6 48.18
Bassoon Vitus	33.33 (Flute)	73.33				75
Cello academy**BY**	90.9	0 (Guitar)	0 (Piano)			81.82
Clarinet italian	26.31 (Violin)	0 (Violin)	21.05 (Violin)		10.53 (Violin)	36.84 (violin)
Flute Ircam	88.33	28.33 (Violin)	58.33			83.33
Guitar berio	78.33	86.67	6.67 (Cello)			75
Piano italian	0 (Guitar)	0 (Cello)	0 (Violin)		13.04 (Violin)	0 (cello)
Violin academy	88	80	68		100	100
						5/7 70.63
Clarinet Berio **BY**	18.33 (Violin)	3.33 (Violin)	63.33	6.66 (Flute)		85
Piano academy	6.9 (Flute)	0 (Flute)	82.76	0 (Bassoon)		75.86
Violin Berio	100	100	73.33	51.67		100
						3/3 89.26

It is interesting to see that unlike classification of samples of separate notes, which made some sense even when they were wrong (for instance classifying a Clarinet as Flute), here it is very hard to find similarity among the correct and wrong instruments. It also happens sometimes that a solo covers a wide range of sounds, and "invades the realm" of another instrument - a learned solo of an instrument A which has more in common with test instrument B than the learned solo of instrument B.

We can see such an example in the second column. In all the classifications done by group #2, the Clarinet was classified as Violin. It seems that for some mystic reason, the violin of Yehudi Menuhin quite resembles the Clarinet.

If we compare the results of the mutual classifications with the minus-1 DB results (the last column), we see that the Minus-1 DB results usually are much better than the average classification results of each Solo, but not better than its maximum result. This shows that when we put together many Solos into the same learning Database, there is a lot of interference between them - much more than we had when we classified separate notes.

Early experiments with removing outliers did not improve the results. It seems that instruments really invade each other's domains (at least with the descriptors I use) and we are not dealing here with freak cases of "outlandish" windows that could be easily detected and removed.

Appendix A - comparing the old and the new descriptor modules

I have used Minus-1 DB classification of separate notes using the CGB BP80 network (exactly as I did in my ISMIR paper) to compare the results using Geoffroy's new and old descriptor modules. In the new module I used the mean+standard deviation statistics, as Geoffroy does.

Results:

Instruments classification	**New Module**	**Old Module**
SOL	81.06	87.78
IOWA	73.31	74.71
McGill	82.35	80
Pro	87.97	84.18
Vi	91.16	89.16

Minus-1 DB results using BP80

We see that the results are not very different. The largest difference is in the SOL database, which was classified better by the old module by 6.6%.

Although the difference is not big, I shall continue using the new module (which I used in this report). The reason is its versatility - besides providing the mean and the standard deviation for every descriptor, it also offers the descriptor values in every time frame of the signal (a frame is 60 ms at the moment), which is useful for silence detection, segmentation, etc.

Appendix B - Solo pieces used in this report

Following is the list of the solo pieces:

Cello Emanuel, Clarinet Emanuel, Piano Emanuel - taken from Emanuel who works on music segmentation.
Originally 22Khz. Longer than the 30.5 seconds required for my full 60 classification windows. Good quality.

Flute Italian, Guitar Italian, Clarinet Italian, Cello Italian, Violin Italian - samples taken from an Italian web-site.
Very short (4-12 seconds), noisy, 11KHZ, compressed with ADPCM.

Bassoon DAT, Flute Ircam. Copied from IRCAM DAT tapes. Quality OK. Over 30.5 secs. Weird and jumpy.

Bassoon Vitus. As bassoon solos were extremely hard to find, I glued together the bassoon files (a separate file for each note) from the Vitus database.

Bassoon Berio, Flute Berio, Guitar Berio, Clarinet Berio, Violin Berio - taken from the Sequenzas by Luciano Berio.
Weird and jumpy - lots of sound effects (done with the instrument). Longer than 30.5 seconds and good quality.

Guitar Baldi, Piano moon1, violin kremer, Guitar Kayath, Piano_thepiano, Flute Bach. Good recordings, longer than 30.5 secs. from various CD's.

Clarinet amazing, Cello Berlin - quality OK but short (<15 secs.)

clarinet hatikva, violin menuhin. Longer than 30.5 secs. but old and not recorded very well.

Cello Academy, Violin Academy, PianoAcademy - taken from the web site of some academy. Short (up to 13 secs.), bad quality.

Appendix C - Lossless compression[3]

Uncompressed sound files take a lot of storage space. The ones I use for my experiments, for example, consume 7 GB of my hard disk.
There are several programs for performing lossless compression of sound files. At the moment I am using the APE format:
http://www.monkeysaudio.com/

The advantages of APE are that there are versions for multiple platforms, the source code is available and there is a plug-in for the sound editor I use (Cool-Edit).
Various comparison charts show that the APE format gives the best lossless compression results (for sound files) and works faster than other protocols.
The common compression rate of APE is around 50%, but samples of "simple" sounds, without very complex spectra (like Solo pieces, or sample files with single notes) produce much better results. A good example is the Ali-Khan disk with Solo singing - the whole 64 minute disk was compressed into 77MB.

For me, the main advantage of using lossless compression is the comfort of storing more sample files together without losing information.

[3] Although this section seems more appropriate for [iii] than for the report, I still thought it's interesting to include it.

Report - 07.08.2003

By Arie Livshin

This report presents different attempts to improve the classification of solo pieces.

Contents

General remarks

Test Databases:
I have separated my solo pieces into 5 databases of solos, each containing at most a single solo of each instrument.
The instruments are: Piano, Guitar, Bassoon, Clarinet, Cello, Violin and Flute.
A 30 seconds piece or shorter (in solos which are shorter than 30 seconds) was taken out of each solo and cut using a sliding window. The descriptors were calculated on these cuts (mostly called "windows" throughout the report, except in cases where it might cause confusion).
I have used the "Minus-1 DB" method in most of this report - each solos database was classified using the rest put together.
The grading is done in 2 categories - the average classification success percentage of the cut windows ("Windows Grade") per database and the average "percentage" of solos ("Solos Grade") recognized correctly per database. In order to prevent confusion among the two grades, the Windows Grade is given in percents out of 100 (e.g. 78.52%) while the Solos Grade is given out of 1 (e.g. 0.88). Although this means that the Solos Grade has 2 digits less after the decimal point than the Windows Grade, this presents no practical drawbacks.

Abbreviations:
LDA+KNN - A classification algorithm, where the descriptor values first go through Linear Discriminant Analysis, then the best K for KNN is searched using Leave One Out on the learning group with K values ranging from 1 - 20, and then the test group is classified using KNN and the elected value for K.

BP80 - A back propagation neural network with one hidden layer of 80 neurons. For training, I have used the CGB algorithm (Conjugate gradient backpropagation with Powell-Beale restarts), which showed good results in the past, classifying separate notes.

MSE - Mean Square Error. When I write MSE in this report, I mean the stopping condition for the net's training - i.e. when the MSE of the net reaches a certain minimal value, the training stops. It is hard to find the optimal MSE, because when the value is too big, the net does not learn well enough the learning group, while when the value is too small the net might learn the learning group "by heart" and overfit (more details on MSE selection can be found in "Estimating the best MSE" in the "Regarding batch experiments" section).

Min-MAX - A normalization method where the data is normalized to the range 0 - 1. During the report all the data is normalized using min-max unless it is stated differently.

Z-Score - A normalization method where the data is normalized to have a mean value of 0 and a standard deviation of 1.

Intersecting the results of several algorithms

In Solos Classifications, i.e. deciding which instrument is playing a solo using the knowledge that there is only a single instrument in each solo, I classify the solo by the most popular classification of its windows.
For this purpose there is no need to classify all the windows of the solo, and it should be enough to select several windows out of the solo which are most likely to be classified correctly, and use them to determine the playing instrument.
I have used such an approach in the previous report, when I removed windows from the test group which the neural net "was not sure" about their classification, i.e. windows classified with low probabilities. This technique did not improve the results.

In this section I shall try a different approach for getting rid of problematic windows - intersecting several classification algorithms.
When we compare the Windows classification of the LDA+KNN algorithm and the classifications made by several BP80 nets, we can see that the specific classification of many windows change (each BP80 network classifies somewhat differently due to the random initial weights of the net).

In this section the solos classification will be done using only windows which were classified the same throughout several different algorithms and classifications.

Musical Instrument Taxonomy. 4 BP80 nets (MSE 0.007) and a single LDA+KNN (in Minus-1 DB classifications, LDA+KNN always produces the same results, as there are no random values).
The results show the number of correct solos classifications in each of the 5 databases, the average Solos Grade per classification experiment and the results of using just the common (intersected) classifications.

```
all_results_T1_A2.mat - KNN
7/7 5/7 5/6 6/7 3/3   : 0.880952
all_results_T1_A3_1.mat - BP80
7/7 5/7 4/6 5/7 3/3   : 0.819048
all_results_T1_A3_2.mat - BP80
6/7 5/7 4/6 5/7 3/3   : 0.790476
all_results_T1_A3_3.mat - BP80
6/7 5/7 5/6 5/7 3/3   : 0.823810
all_results_T1_A3_4.mat - BP80
6/7 5/7 4/6 5/7 3/3   : 0.790476
AVERAGE: 0.8209524

COMMON -
6/7 6/7 5/6 6/7 3/3   : 0.880952
```

Note 1: One should take care when merging results from several algorithms not to lose solos - too many classifications and there might be solos where no windows are left which are classified the same by all the algorithms. This is the reason why in this experiment only 4 BP80 nets were used while in the next one I used 5 nets - intersecting the 5'th BP80 caused a solo to disappear.
Note 2: we can already see that LDA+KNN classify solos in the Musical Instruments taxonomy better than BP80.

Pizzicato/Sustain Taxonomy. 5 BP80 nets (MSE 0.0006) and one LDA+KNN

```
all_results_T3_A2.mat - KNN
7/7 5/7 6/6 6/7 2/3   : 0.847619
all_results_T3_A3_1.mat - BP80
7/7 7/7 6/6 6/7 2/3   : 0.904762
all_results_T3_A3_2.mat - BP80
7/7 7/7 6/6 6/7 2/3   : 0.904762
all_results_T3_A3_3.mat - BP80
7/7 7/7 6/6 6/7 2/3   : 0.904762
all_results_T3_A3_4.mat - BP80
7/7 7/7 5/6 5/7 2/3   : 0.842857
all_results_T3_A3_5.mat - BP80
7/7 6/7 6/6 5/7 2/3   : 0.847619
AVERAGE: 0.875396833

COMMON -
7/7 7/7 6/6 5/7 2/3   : 0.876190
```

By examining the results of these experiments we can see that classifying using the common results of several classifications produces a result which is higher than the average classification result of these classifications, but not higher than the maximum grade.

I have performed many other "intersected classifications" using a single BP80 net and an LDA+KNN. Sometimes the result is higher than the results of both the algorithms and sometimes just around the average. It is uncommon that the intersected group yields to the lower result among the two classifications. It never produces results lower than the intersected algorithm with the lowest results.

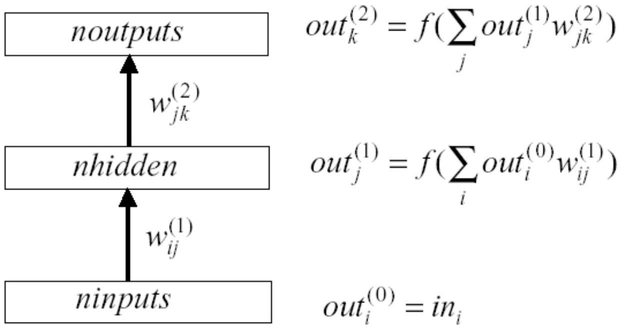
Conclusions
The method might have some use, taking into account that it usually produces results above the average classification result, but it is not "revolutionary". It seems that the main limitation of the classifications at this stage is the classified data and not the classification algorithms.

Normalization issues

Neural Nets and Normalization?

Following one of the previous reports, it was asked why the data should be normalized before being used for training a neural network.

This is a diagram of an MLP (multi layer perceptron) network:

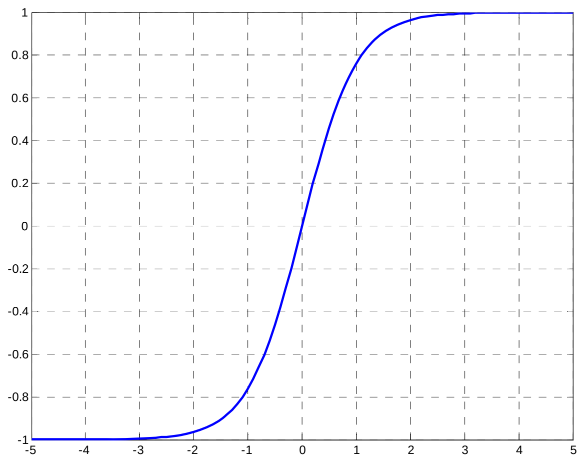


noutputs
$w_{jk}^{(2)}$
nhidden
$w_{ij}^{(1)}$
ninputs
$out_k^{(2)} = f(\sum_j out_j^{(1)} w_{jk}^{(2)})$
$out_j^{(1)} = f(\sum_i out_i^{(0)} w_{ij}^{(1)})$
$out_i^{(0)} = in_i$

The "BP80" network which I use throughout this report is an MLP network with 80 hidden units in one hidden layer, trained using the 'Conjugate gradient backpropagation with Powell-Beale restarts' algorithm.

The transfer function I use to compute $out^{(1)}, out^{(2)}$ is f = tansig (x).

Tansig(x):



If the input values for the net are very large, the net cannot find the difference between one result and another when it alters the weights, and thus cannot find the learning gradient (by which it knows how to learn), and the training process stops. That is why it is required to normalize the data and prevent it from falling in the far ends of the tansig function.
On the other hand, if the inputs are very small and fall around the very middle of the tansig function, tansig will function as an almost linear function and the net will have hard time learning non-linear data.

Zscore vs. Min-Max for training neural nets

Experiments using the Zscore normalization method (mean=0, std=1) compared with MinMax (0 - 1) show that my BP80 network classifies with about the same success percentage with both, but learns much faster with Zscore.

The following results show how many training cycles ("epochs") are required for the net to get down to a specific MSE in each solo database.

Musical Instruments taxonomy, number of epochs for MSE 0.004:

ZScore:	94, 103, 99, 96, 99
minMax:	231, 273, 308, 276, 236

Average Minmax/Zscore: 2.68

Pizzicato/Sustain taxonomy, number of epochs for MSE 0.001
(the Pizzicato taxonomy requires a lower MSE than the Instruments):

ZScore:	109, 86, 172, 101, 115
minMax:	299, 302, 351, 315, 399

Average Minmax/Zscore: 2.98

Conclusions
We see that the net trains much faster with data normalized with Zscore than with Minmax; the time requirements for future experiments with neural nets will be shortened.

"Gluing" separate notes together to create "solos"

One of the possible ways for me to deal with the shortage in solo pieces was to try to use "mock solo" files created by "gluing" together separate sounds out of the sample databases in my possession (e.g. SOL, IOWA, etc.)

Qualities of glued files:
- Contain all the notes in the range of the instrument, or at least enough notes to define the instrument sound quite well (the sample DB's are usually intended for Sampler Synthesizers and are supposed to encapsulate the pitch range of the instruments). On the other hand, the performance technique is very limited and uniform (at least in the samples I used; in SOL for example, there are also samples of various different (and weird) playing techniques for each instrument).
- The sounds are uniform, separate and complete - every glued sample contains a single, full note - this note has an attack at the beginning (not a legato), it is separate from the previous sound and monophonic (in the two senses that the note is played by itself - unlike the polyphonic piano and guitar solos I have used, and that there is no reverb from the previous sound).
- Glued files have no real transitions between the sounds.

Remark - Because glued files often had very long sounds as well as silence periods, I have cut out of the files parts where the energy was too low or the note was too long, thus creating unnatural sound clippings. This "problem" could be easily solved by removing the extra parts (silence, etc.) after cutting the solos with the sliding window instead of before.

In order to test these "glued" files, I have created a 6'th database of solos, called "database #6", which consists entirely of solos glued out of single note samples. Besides this database there are other 5 databases with "natural" solos.

Database #6 was glued out of:
Vitus: Cello, Flute, Clarinet, Bassoon, Violin.
Iowa: Piano.
SOL: Guitar.

The reason for most of the samples being taken out of Vitus, is that when classifying single sounds using "mutual classification" (classifying every sound database by every other one), the highest classifier grades were for Vitus. Regarding Piano and Guitar, they were taken out of the only databases I had which contained them.

In contrast with the real solos, where I took a 30 seconds clip out of each solo (or the whole solo in cases when the solos are shorter than 30 seconds) and cut it up using a sliding window of 1 seconds, moving at 0.25 second, into up to 120 windows, in database #6 I have used the full glued files and cut all of them with the sliding window, so the number of windows in database #6 is much higher than in the other 5 databases.

The following table shows the average results of 3 BP80 experiments and one LDA+KNN. In order to find the best MSE, I used gradually descending stopping conditions, classifying the data each time (details could be found in "Estimating the best MSE" in the "regarding batch experiments" section).
The first data column shows the highest grade for Solos Classification and the MSE which produced it. The second column shows the best Windows Grade and the corresponding MSE. The third shows the MSE which produced the highest average between the Windows and Solos Grades. The fourth column shows the Windows and Solos Grades of an LDA+KNN algorithm.

<u>Musical Instruments taxonomy, Minus-1 DB experiments, 6 solo databases:</u>

	<u>BP80</u> Max Solos (MSE)	<u>BP80</u> Max Windows (MSE)	<u>BP80</u> Average max (MSE) Solos, windows	LDA+KNN
6 databases 3 experiments 1 sec, 0.25 mov. max 120 windows (except database #6)	0.83 (0.03)	68.013 (0.001)	74.61 (0.0007) 0.81 , 68.011	Solos 0.78 Windows 66.89
Only the 5 real-solo databases	0.8 (0.03)	70.97 (0.005)	75 (0.0007) 0.79 , 70.63	Solos 0.81 Windows 72

The second row in the table shows only the results of classifying the 5 databases containing real solos; each is classified using the rest of the databases together including the glued database #6.
We can see from the second row that adding database #6 actually reduced the average Solos Grade for the 5 real-solo databases (its highest result was 0.88, with LDA+KNN) and remained the same in the Windows Classification (70.97).

Let us examine "Mutual Classification" using database #6 (i.e. classifying every other database using database #6):

	Database of glued samples
	Bassoon Vitus
	Cello Vitus
	Clarinet Vitus
	Flute Vitus
	Guitar Vincent3 (SOL?)
	Piano IOWA MF
	Violin Vitus LN
Bassoon dat **BY**	73.33
Cello emanuel	100
Clarinet amazing	2.13 (Flute)
Flute bach	4.17 (Cello)
Guitar baldi	14.17 (Cello)
Piano moon1	4.17 (Guitar)
Violin kremer	3.33 (Cello)
Bassoon berio **BY**	78.33
Cello italian	100
Clarinet hatikva	0 (Guitar)
Flute italian	15.69 (Violin)
Guitar kayath	45 (Cello)
Piano thepiano	70.83
Violin menuhin	59.17
Cello berlin **BY**	100
Clarinet emanuel	0 (Cello)
Flute berio	42.5
Guitar italian	0 (Cello)
Piano emanuel	35.83 (Guitar)
Violin italian	100
Cello academy**BY**	100
Clarinet italian	0 (Violin)
Flute Ircam	68.33
Guitar berio	77.5
Piano italian	0 (Cello)
Violin academy	100
Clarinet Berio **BY**	0 (Flute)
Piano academy	0 (Cello)
Violin Berio	100
AVERAGE:	
Bassoon (2)	75.83
Cello (4)	100
Clarinet (5)	0.43
Flute (4)	32.67
Guitar (4)	34.17
Piano (5)	22.17
Violin 5)	72.5

We could have expected the bad results in Piano and Guitar - the glued files are monophonic while the real solos are polyphonic.
Surprisingly, the worst result was for Clarinet (0.43%). This is not due to clipping, as the glued Clarinet file contains almost no clippings (not of long sounds nor of silence).

Remark: It is important to remember that this table shows only a simplified view of the situation, because it is not only important how much database #6 is a good classifier, but also how much it interferes with the other databases and their interaction (the ability of one database to classify another).

Conclusion
The benefit of using glued files is not obvious[1]. In any case, it does not seem to be a revolutionary method, as might have been hoped due to its encapsulation of the entire pitch range of instruments.

It is quite obvious that the results depend on the specific sample database being used. We have seen in classifications of separate sounds that there are databases better suited for "general classifications" than others. As already mentioned, this was the reason for using Vitus, which was comparatively successful at the time.
Something I did not try was to enrich the glued "solo" files with "special-technique" sounds out of SOL. The Bassoon samples in SOL, for example, include key-clicks, trills, blowing without the mouthpiece (which is really too much), etc. On the other hand, it is possible that these additions will just add to the confusion of the classifier.

I could have tried a totally different classification method. First Learning the evolution of the sounds of different instruments using Markov models (or recurrent networks), then going through the solos and finding the most fitting Markov model for each window. In such an evolutionary method, it might be possible that the separate note samples would be very useful for training the Markov models, as they contain complete sounds.
Also, I could have used transient detection and try to determine where notes start and end instead of using "blind" cutting. The separate note samples might have been very useful for learning, as examples for windows with perfectly detected transients.
I might try these methods in the future.

Self Classification (a reminder)

I shall perform again a "self classification" experiment with the solos, because the cutting and classification parameters were changed since my first report on the subject. Out of each solo a 30 second clip is taken (or less - as already been mentioned, some solos are shorter than 30 seconds), and cut it using a sliding window of 1 second moving at 0.25 second steps (up to 120 windows from each solo).
A learning group consisting of 66% of the windows is taken from each instrument class (regardless of the solo the windows belong to). This group is used to classify the rest of the windows - the test group.
This is a Windows Classification, as I do not try to classify whole solos but to classify all the windows cut out of them. I remind that the maximum Windows Grade achieved with the Minus-1 DB classification method was 71.38%.

Using a step size of 0.25 second with a 1 second window creates samples containing a lot of common signal (almost the entire signal is duplicated 4 times). For this reason I will present also the results of using every 4'th window ("no duplicity").

[1] I know they could serve some purpose though, because when I added only the glued Bassoon file it has considerably improved the Bassoon classifications.

The following results are the average of 3 experiments.

<u>"5 databases" (only real solos)</u>

Musical instruments taxonomy - all windows:	99.5%
Musical instruments taxonomy - no duplicity:	96.38%
Pizzicato/Sustain taxonomy - no duplicity:	99.27%

<u>"6 databases" (including the glued files of Database #6)</u>

Musical instruments taxonomy - all windows:	99.59%
Musical instruments taxonomy - no duplicity:	96.59%
Pizzicato/Sustain taxonomy - no duplicity:	97.74%

<u>Conclusion</u>

I am using many descriptors (513 values of 69 different features) and it was not very hard for the classification algorithms to find the common qualities for the instruments in the joined database.

As already seen in the past (and written for ISMIR), Self Classification results could be very misleading.

Comparing window sizes

In the following experiments I have compared the classification results of different window sizes. As in many other experiments, a 30 seconds clip (or less, in solos which were too short) was cut out of each solo. This clip was then cut using a sliding window of a specific size moving at 0.25 second steps. The resulting windows were classified using Minus-1 DB.

The following table shows classifications of the 5 "true" solo databases, 3 columns for BP80 and one for LDA+KNN. The first data column shows the maximum Solos Grade and the corresponding MSE (explanation could be found in the section "regarding batch experiments"), the second column shows the maximum Windows Grade and the corresponding MSE, and the third the MSE which produced the best average of the Windows and the Solos Grades.
The fourth column shows the Solos and Windows Grades of LDA+KNN classifications.

	Max Solos (MSE)	Max Windows (MSE)	Shared max (MSE) Solo,windows	LDA+KNN Solo,windows
0.25 sec window size, 0.25 mov 5 experiments MSE limit – 0.001	0.82 (0.007)	64.09 (0.005)	72.61 (0.007) 0.82 , 63.6	0.85, 64.42
0.5 sec window size, 0.25 mov 5 experiments, MSE limit – 0.001	0.81 (0.004)	66.74 (0.006)	74.09 (0.004) 0.81 , 66.66	0.85, 68.9
1 sec window size, 0.25 mov **10** experiments, MSE limit 0.001	0.84 (0.007)	70.28 (0.007)	77.18 (0.007) 0.84 , 70.28	0.88, 71.38
1.25 sec window size, 0.25 mov **5** experiments, MSE limit 0.001	0.81 (0.006)	70.64 (0.005)	75.61 (0.006) 0.81 , 70.26	0.82, 69.6

We see that the winner is LDA+KNN with 1 second windows. Almost all the window sizes produced better Solos Grades with LDA+KNN than with BP80.

It could be interesting to try clips of more than 30 seconds. It seems reasonable to believe that the more we learn, even out of the same solo piece (with the same instrument), the higher the grade will be. There might be a problem with solos which are already shorter than 30 seconds - these will have comparatively even less windows if the number of windows out of longer solos gets bigger.

In the previous report using BP80 with 60 windows of 1 second cut with a sliding window moving at 0.5 second, the grades were: 0.79 69.74. The results for BP80 now with 120 windows of 1 second cut with a sliding window moving at 0.25 second, are better: 0.84 70.28.
What has improved the results (the window size is the same)? Is it the higher number of windows (120 compared to 60) or the sliding window step (0.25 compared with 0.5)?
It is possible to try different step sizes. The problem with all these experiments intended for finding optimal parameters, is the long time they require for computing the descriptors (when recomputing is needed, e.g. when testing various window sizes), training and retraining BP80 nets several times, etc.

Removing outliers

The samples ("windows") I am working with have been blindly cut out of solo files, as opposed to "well behaved" samples taken out of professional sample databases. For this reason there might be various non-standard samples present in the solo databases, including silence (although I manually removed long silences), special playing techniques (trill, glissando, etc.) and samples with structural "problems" (having no attack, or containing several attacks due to fast playing), etc. It seems to make sense to "remove outliers", trying to get rid of those non-standard samples.

When doing Windows Classification it is possible to remove outliers only from the learning group. Note that samples containing special playing techniques in the learning set might actually help to recognize similar samples in the test group.
It is important when removing outliers to make sure not to remove entire solos which are very short - the entire collection of windows from a very short solo might be considered as outliers and removed.

In Solo classifications we can remove outliers also from the test group, as a solo is classified by the majority of its windows. From first looks it makes sense that removing samples from the test group which are likely to be misclassified should help.
On the other hand, when a solo is misclassified as instrument X, it means that the majority of its samples are misclassified as this instrument. Following from that is that removing outliers will more likely remove samples which are correctly classified than the ones classified as instrument X.

In this section I have used two methods for removing outliers: IQR (interquantile range) and MIQR (modified IQR).
A reminder: MIQR is a method in which I perform IQR to every class separately, and while IQR removes all the samples which contain at least one outlier descriptor, in MIQR I remove the samples which have the highest number of outlying descriptors.
An advantage of IQR is that it could be applied to the test group, as it does not require knowledge regarding the classification of the samples.

In the following tables I will present the results of several different outlier removing techniques. The samples were cut using a sliding window of 1 second moving in 0.25 second intervals.

The tables show the number of correctly classified solos in each database, the Windows Grade and the percentage of samples removed. At the bottom of each column there are the averages of the number of correct Solos Classifications, the percentage of the Windows which were correctly classified and the percentage of removed windows. X-Y in the column titles shows the parameters I used for the IQR. The classifications were performed using a BP80 network.

Note - the results of Windows Classification are present only for a "comparative look" - to see an indication of how many of the removed samples were originally misclassified. We should not compare the Windows Grade directly with the Windows Grade of classifications where there were no outliers removed, as outlier samples were eliminated from the test group.

<u>Removing outliers only from the test group:</u>

IQR 10-90 Solos, windows, (removed %)	IQR test 5-95 Solos, windows, (removed %)	IQR test 2-98 Solos, windows, (removed %)
6/7 76.37(35.98) 4/7 80.97(53.31) 5/6 80.9 (61.58) 5/7 56.1 (69.46) 3/3 81.77 (39.06)	6/7 76.84(16.7) 6/7 78.52(32.7) 4/6 62.83(31.1) 5/7 62.74(42.53) 3/3 79.53(14.48)	6/7 76.22 (4.04) 4/7 68.52(18.67) 4/6 62.03(18.09) 5/7 64.07(18.62) 3/3 82.5 (5.72)
0.79 75.22 (51.88)	0.82 72.09 (27.5)	0.76 70.67 (13.03)

<u>Removing outliers from both the learning and the test groups:</u>

1.0,0.25,120 IQR learning and test 2-98	1.0,0.25,120 MIQR learning, IQR test 2-98
6/7 81.96 (29.14 learn, 17.6 test) 4/7 71.66 (26.04, 25.6) 4/6 63.8 (24.24, 33.74) 4/7 62.16 (23.02, 37.99) 3/3 87.55 (27.43, 13.47)	7/7 83.75 (4.92,15.77) 4.5/7 57.21(4.82, 6.02) 3/6 53.76 (4.81, 10.77) 5/7 65.79 (4.77, 22.16) 3/3 85.38 (4.92, 12.46)
0.73 73.43 (25.97, 25,68)	0.77 69.18

We see that after removing the outliers the Solos Grades actually became worse (0.82 compared to 0.84 (in BP80) without removing outliers). On the other hand we see that the Windows Grade became better (73.43 compared to 70.28 without removing outliers).
These results show that we have removed more problematic samples than "good" ones. A possible explanation to how it is possible to improve Windows Classification while harming Solos Classification, is that the scatter of the removed outliers is non-uniform; many bad samples are removed from solos without altering their classification, while the good samples removed by mistake might come from fewer solos, harming their classification.

Other experiments I did with removing outliers using different parameters (including experiments with LDA+KNN) did not improve the Solos Grades enough - they remained lower than the grades of experiments where no outliers were removed.

Regarding batch experiments

"Failsafe" training

Sometimes it happens that due to bad initial weights (the initial Neural Network weights are chosen randomly at initialization), the neural net gets "stuck" in a local minimum and fails to converge to the desired MSE. When a batch of experiments is ran without human supervision, such local minima could waste a lot of computer time - the net learns in every training epoch just enough so the training algorithm will not stop automatically, but the training speed is actually too low for the net to ever reach the desired MSE.

To deal with this problem I train the net in two stages. First I limit the training to 50 epochs and if the MSE does not get below 0.1 (50 and 0.1 are parameters suited for my experiments), the net restarts the training using different initial weights. If it did reach below 0.1, I continue the training, this time until the net reaches the desired MSE and allowing a large number of epochs.

Simply telling the training algorithm to stop the training if the learning gradient is too small will not work, as the learning gradient decreases naturally during training and setting such a stopping condition for the gradient will stop the net from reaching the desired MSE.

If for some reason the net has passed the failsafe but stopped without reaching the desired MSE (virtually never happens), the experiment is automatically repeated.

Estimating the best MSE

A neural network is trained until it reaches a specified MSE (mean square error) or until it cannot learn anymore (learning gradient too small). It is hard to decide which MSE is optimal - if an MSE is too big, the net might not learn the examples well enough and will not be able to generalize well. An MSE which is too small will result in the net actually "learning by heart" the examples and overfitting. The optimal MSE depends on the specific data being learned and the initial weights of the net.

In order to get some insight regarding the MSE's I should use and because the training process takes a long time, I wrote a training routine that instead of specifying the desired final MSE to the training algorithm, it trains the net in stages using smaller and smaller values for MSE each time. When each intermediate MSE is reached the net is activated on the test group and the results are recorded. As the initial random net weights affect the results of the training, the decreasing-MSE process is repeated with several nets.

At the end, the routine reports the best MSE for Windows Classification, the best MSE for Solos Classification, and the MSE which produced the highest average between the Solos and Windows Grades.

This routine produced many of the MSE values I used in the BP80 experiments in this report.

Pausing for rest

During the last few weeks there were several VERY hot days (today for example, the temperature reached 38 degrees Celsius). A very uncomfortable side effect was that my laptop computer used to turn itself off spontaneously from time to time, without any warning or saving its memory (the official limit for its operating temperature is 35 degrees Celsius). It bothered me the most when I was leaving my computer to make

experiments during the night, and in the morning I would find out that it just switched itself off somewhere in the initial stages of the experiment.
One of the possible solutions[2] (besides buying an air conditioner), is to make the program pause from time to time (the "pause XX" command in Matlab), and let the CPU rest for several seconds and cool off.

A new simple Pizzicato/Sustain descriptor

In previous reports we have seen confusion matrices which showed musical instruments which apparently have no connection to each other (like Piano and Bassoon), being confused by the classification algorithms. This has raised the question whether the descriptor model I am using contains a descriptor which takes into account the slope of the signal. How could Piano, which has a "Pizzicato" slope, be confused with Bassoon, which should have a "Sustain" slope?

First I have used LDA to check whether there are such descriptors and whether the classifying algorithm uses them.

<u>In the Pizzicato/Sustain taxonomy</u>, the 10 descriptors with highest weights were:
(the LDA was computed on the Minus-1 DB of solo database #1 - I had no reason to believe it would be different for the other solo databases)

'DPi_slope_v6-mm' 'DTi_xcorr_m2-mm' 'DTi_xcorr_m3-mm' 'DPi_slope_v3-mm'
'DTi_xcorr_m4-mm' 'DSi_sc_v6-mm' 'DPi_sc_v6-mm' 'DSi_sc_v3-mm'
'DTi_xcorr_m5-mm' 'DPi_skew_v3-mm'

With the corresponding weights:
0.5667 0.4183 0.3557 0.2860 0.1990 0.1772 0.1700 0.1364 0.1299 0.1138

<u>In the Musical Instrument Taxonomy</u>:

'DPi_slope_v6-mm' 'DPi_sc_v6-mm' 'DPi_skew_v3-mm' 'DPi_slope_v3-mm' 'DPi_sc_v3-mm'
'DSi_sc_v6-mm' 'DSi_sc_v3-mm' 'DSi_skew_v6-mm' 'DPi_ss_v6-mm' 'DSi_slope_v6-mm'

With the corresponding weights:
2.3122 1.7762 1.6912 1.6231 1.5139 1.2991 1.0492 0.8165 0.6866 0.6665

We see that the most important descriptor in both the taxonomies is indeed one dealing with the slope of the signal (this descriptor represents the amount of decreasing of the spectral amplitude; it is computed by linear regression of the spectral amplitude in Mel bands (DP are supposed to be "perceptual" descriptors)).

[2] Besides explicitly trying to solve the problem of overheating, it is also important to save a lot of temporary results in order not to lose the data the computer did manage to compute before switching itself off. Otherwise, after each failure one needs to start the experiments batch from the beginning.

To address the question of the slope even more directly, I have built a very simple "pizzicato detector", which works in the following way:

- The solo is cut into non overlapping windows
- The maximum value in each window is found
- The number of windows with a bigger maximum value than the windows which follow them is divided by the number of windows with a smaller maximum value than the following ones. For example if 4 windows have the following maximum values - 0.2 0.5 0.3 0.1, then we compute: 2/1, as we have two windows with higher maximum values than the next ones, and one with a lower value than the next one (the first window).
- If the resulting number is bigger than a specific constant (1.155 in my experiments), then the solo is classified as Pizzicato (otherwise Sustain).

The intuition behind these calculations is that in Pizzicato there is a fading of the signal after the attack, while in the sustained instruments the signal's average after the attack remains more constant (regardless of vibrato).

In the following results the algorithm is applied to each of the 5 solo databases. The last number in each row is '1' if the algorithm decided that the solo is pizzicato, and 0 if sustain:

```
>> isPizzicatoupDownDir('C:\Music_Research\samples\cont\new-blindCut\groups\1')
bassoon-1-dat.wav : down/up 0.829787, 0
cello-1-emanuel.wav : down/up 1.004689, 0
clarinet-1-giora-amazing.wav : down/up 0.837333, 0
flute-1-bach.wav : down/up 0.961843, 0
guitar-1-baldi.wav : down/up 1.290374, 1
piano-1-moonlight-1.wav : down/up 1.156025, 1
violin-1-kremer.wav : down/up 1.067895, 0
>> isPizzicatoupDownDir('C:\Music_Research\samples\cont\new-blindCut\groups\2')
bassoon-2-berio.wav : down/up 0.992742, 0
cello-2-italian.wav : down/up 1.028037, 0
clarinet-2-giora-hatikva.wav : down/up 0.959111, 0
flute-2-italian.wav : down/up 1.075000, 0
guitar-2-kayath.wav : down/up 1.281159, 1
piano-2-thepiano.wav : down/up 1.213267, 1
violin-2-menuhin.wav : down/up 1.038057, 0
>> isPizzicatoupDownDir('C:\Music_Research\samples\cont\new-blindCut\groups\3')
Cello-3-berlin.wav : down/up 0.967985, 0
clarinet-3-emanuel.wav : down/up 1.224719, 1
flute-3-berio.wav : down/up 1.023163, 0
guitar-3-italian.wav : down/up 1.881356, 1
piano-3-emanuel.wav : down/up 1.192552, 1
violin-3-italian.wav : down/up 1.217391, 1
>> isPizzicatoupDownDir('C:\Music_Research\samples\cont\new-blindCut\groups\4')
bassoon_vitusMerged.wav : down/up 0.780026, 0
cello-4-academy.wav : down/up 0.981595, 0
clarinet-4-italian.wav : down/up 1.091241, 0
flute_ircam.wav : down/up 1.299788, 1
guitar-4-berio.wav : down/up 1.102772, 0
piano-4-italian.wav : down/up 1.361404, 1
violin-4-academy.wav : down/up 0.961538, 0
>> isPizzicatoupDownDir('C:\Music_Research\samples\cont\new-blindCut\groups\5')
clarinet-5-berio.wav : down/up 1.121980, 0
piano-5-academy.wav : down/up 1.348571, 1
violin-5-berio.wav : down/up 1.082852, 0
```

There were 4 misclassifications. Average Solos Grade per database: 0.88.
(Per-Instruments grade (explanation is in the section about grading methods): 26/30=0.867).

Not bad for such a primitive method, although this grade (0.88) is lower than the grade achieved by using all the descriptors and BP80 (0.93). It is interesting that this algorithm classified correctly all the 3 solos in database #5, while both the BP80 and LDA+KNN classified correctly only 2 solos from this database.

Regarding the misclassifications:
"violin-3-italian" misclassified as Pizzicato - it is a very short and very fast solo, which has an intentional decrease of playing 'volume'.
"flute_ircam.wav" misclassified as pizzicato - this solo clip has a large section played in staccato - after each staccato note we hear the reverb descending in volume. Another part in this solo clip has an intentional volume decrease.
"clarinet-3-emanuel.wav" misclassified as pizzicato - a very staccato' clip with a lot of reverb, so every sound is short and descends in volume due to the reverb.
"guitar-4-berio.wav" is misclassified as sustain - a very fast clip, too fast to hear degradation of volume, very "messy" and uses non standard playing techniques.

The fact that the "descriptors" algorithms still provide better results, shows us an example to the situation where a feature selection algorithm which uses a lot of collected features and tries to blindly find common qualities of the classes (and the best example for that are the "black boxes" of the neural networks) performs better than "direct" algorithms which try to address a problem specifically (and simplistically), while the problem itself is not well defined.

I have added a version of this descriptor to the descriptor matrix - the Down/Up value, without using the Pizzicato threshold constant (1.15). This descriptor is calculated separately for each window.

The following results show how the addition of this new descriptor has influenced the classification results. The values are the Solos and Windows Grades for each of the 5 solo databases, and the means.

<u>Musical Instruments taxonomy, LDA+KNN</u>
7/7 76.14, 5/7 64.01, 5/6 68.7, 6/7 65.73, 3/3 79.46
mean: 0.88, 70.81

The classification results without the new descriptor were 0.88, 71.38.
Adding the descriptor decreased the Windows Grade a little.

Pizzicato/Sustain taxonomy, LDA+KNN
7/7 99.22, 6/7 78.31, 6/6 93.09, 6/7 80.82, 2/3 80.47
mean: 0.88, 86.38

The results without the new descriptor were 0.85, 86.19. The addition of the new descriptor improved the Solos Grade and also the Windows Grade a little (reminder - it is enough to conduct LDA+KNN Minus-1 DB experiments just once, as the result does not change when a classification is repeated).

Musical Instruments taxonomy, BP80 (Zscore), 3 experiments, MSE 0.003:
6/7 77.18, 6/7 67.47, 4/6 58.54, 6/7 73.37, 3/3 84.85
mean: 0.85, 72.28
6/7 78.88, 5/7 61.29, 6/6 59.96, 5/7 69.65, 3/3 82.15
mean: 0.86, 70.39
6/7 77.84, 4/7 66.11, 6/6 62.4, 5/7 66.67, 3/3 80.81
mean: 0.83, 70.77

Average: 0.85, 71.15

The results without the new descriptor (average over 10 experiments) were 0.84, 70.28. The addition of the new descriptor improved the results a little.

Pizzicato/Sustain taxonomy, BP80 (Zscore), 3 experiments, MSE 0.001:
7/7 98.96, 7/7 90.96, 5/6 91.87, 5/7 80.82, 3/3 91.58
mean: 0.91, 90.84
7/7 98.3, 6/7 81.77, 6/6 93.9, 5/7 81.19, 3/3 92.25
mean: 0.91, 89.48
7/7 98.83, 6/7 84.04, 6/6 95.32, 6/7 81.56, 2/3 84.51
mean: 0.88, 88.85

average: 0.9, 89.72

The results without the new descriptor (average over 10 experiments) were 0.936, 90.2. The addition of the new descriptor has reduced the results, especially the solos.

We see that the addition of the new descriptor did not actually improve the results. The best grade for the Musical Instruments taxonomy remains of the LDA+KNN algorithm without the new descriptor (0.88, 71.38), while the best grade for Pizzicato/Sustain remains of BP80 without the new descriptor (0.936, 90.2).

We can check using LDA[3] how much this descriptor is important (at least when using KNN). The following results show us the relative importance of the new descriptor compared with the 514 descriptors which were used. The LDA is computed on the

[3] Naturally, these techniques could be used to evaluate every new descriptor added to the descriptor matrix, and are not limited to this one.

minus-1 DB of each database (which means that all the other databases are merged and LDA is computed on them), and the average. Although I could just merge all the databases and compute LDA, the LDA of the Minus-1 DB of the databases reflects exactly the actual classification experiments:

Musical Instruments taxonomy - database 1:456, 2:481, 3:448, 4:387, 5:489
Average placement: 452.2

Pizzicato/Sustain taxonomy - database 1:405, 2:475, 3:396, 4:405, 5:405
Average placement: 417.2

We see that the new descriptor is relatively non-important (has a very low placing) when compared to the rest of the descriptors and is hardly in use.

Conclusions:
Adding this new descriptor does not improve the results. Obviously, we could try to gain better results by adding other new descriptors (e.g. wavelet decomposition), although the current number of descriptors is already very large and covers many features.

Another interesting conclusion is that the feature selection algorithms I used are not perfect (LDA and BP networks). We have seen that adding a descriptor can actually hurt the classification results (at least a little, according to the experiments). If the descriptor selection process was perfect, it should not have happened.
This means that manual removal of non-relevant descriptors might actually improve the results (at least a little). Another way to deal with this problem is to look for alternative feature selection algorithms (e.g. NLDA with kernel functions, ReliefF or CFS).

Hierarchical Classification

Examining the confusion matrices from my previous reports reveals that unlike classifications of separate sounds, where the misclassifications were among intuitively similar instruments (e.g. Violin and Cello), we can see that when classifying solos it is hard to understand intuitively the misclassifications (e.g. Piano and Bassoon).
This hints that a hierarchical classification might be more beneficial for solo classifications than for classifying separate sounds. A successful 1'st level in the hierarchy could prevent a lot of the weird misclassifications occurring in classifications of solo pieces.

Using Pizzicato/Sustain as the 1'st level makes sense due to the apparent simplicity of the task of distinguishing Pizzicato from Sustained sounds as well as the fact that this classification classifies all sounds into just 2 classes.
In the 7 instruments I am using, the same solos which are pizzicato (Piano and Guitar) are also the exact same ones which are polyphonic - thus we have another major distinguishing attribute for the same taxonomy (although there is a possibility of the polyphony actually making the Pizzicato/Sustain discrimination harder, so it is not exactly 2 "orthogonal" distinguishing features).

The reason for all the Pizzicato/Sustain experiments in this report is to find out whether the Pizzicato/Sustain grade is high enough to be used in hierarchical classifications; our real goal remains the Musical Instruments taxonomy.

Pizzicato/Sustain taxonomy, average of 10 experiments using BP80:

	Max Solos (MSE)	Max Windows (MSE)	Shared max (MSE) Solo , windows
"Database of 5", 1 sec, 0.25 mov, **10** experiments, MSE limit 0.001	0.936(0.001)	90.2 (0.004)	91.87 (0.001) 0.936 , 90.1

Let's examine the Pizzicato/Sustain vs. Musical Instrument grades:

Solos Grade:	Pizzicato	BP80	0.94,	Pizzicato	LDA+KNN	0.85
	Instruments	BP80	0.84,	Instruments	LDA+KNN	0.88
Windows:	Pizzicato	BP80	90.2,	Pizzicato	LDA+KNN	86.19
	Instruments	BP80	70.28,	Instruments	LDA+KNN	70.97

We can see that in hierarchical classification, we need the levels beneath the 1'st to produce grades higher than the following ones, in order for the hierarchical classification to be better than a "flat" classification:

Solos:	0.94*X>0.88	=>	X>0.94
Windows:	0.9*X>0.71	=>	X>0.79

"LEVEL 2" -
The following classifications are in limited Musical Instruments taxonomies, trained and tested separately for the Pizzicato instruments (Piano and Guitar) and the Sustain instruments (Violin, Flute, Clarinet, Bassoon, Cello).

The Pizzicato Group (Guitar and Piano)

LDA+KNN, min-max normalization, (Solos and Windows Grades):
2/2 89.17, 2/2 70.42, 2/2 84.51, 1/2 46.39, 1/1 96.49
mean: 0.9, 77.4

BP80, MSE 0.0004, Zscore normalization, average over 3 experiments:
2/2 80.42, 2.2 91.25, 2/2 76.76, 1/2 69.88, 1/1 100
2/2 93.75, 2/2 95, 2/2 92.25, 1/2 66.26, 1/1 100
2/2 85, 2/2 95, 2/2 96.48, 1/2 62.05, 1/1 100

mean: 0.9, (average of 83.66, 89.45, 87.7) 86.94

The Sustain Group (Violin, Flute, Clarinet, Bassoon, Cello)

LDA+KNN, min-max normalization:
4/5 71.35, 4/5 66.27, 3/4 61.71, 5/5 83.83, 2/2 87.92
mean: 0.87, 74.22

BP80, MSE 0.0004, Zscore normalization, average over 3 experiments:
5/5 88.23, 4/5 63.21, 3/4 55.14, 4/5 80.32, 2/2 83.33
5/5 88.8, 4/5 70.52, 2/4 54, 4/5 88.59, 2/2 90.83
5/5 82.92, 4/5 64.86, 3/4 60.57, 4/5 80.59, 2/2 90.42

mean: 0.85, (74.05, 78.55, 75.87) 76.16

Average among the highest results of the Pizzicato and Sustain groups:
(the score of the Pizzicato and Sustain groups is multiplied by the number of instruments in the group)

Solos:	0.9*2, 0.87*5	average 0.88	< required 0.94.
Windows	86.94*2, 76.16*5	average 79.24	> required 79%

Conclusion
We see that hierarchical classification with the Pizzicato/Sustain taxonomy in the 1'st level and the limited Musical Instruments taxonomies in the 2'nd level will most probably not benefit Solo Classifications but might benefit a little to Windows Classifications.
All in all, the hierarchical approach does not seem to cause radical improvements.

Scoring methods - "per database" vs. "per instrument"

During this report, in the Minus-1 DB experiments, each solo database is classified in its turn by the other databases put together. The classification scores (Solos and Windows Grades) are computed for each database. The average scores among the databases are used for comparison among algorithms, improvement techniques, etc.
This method imitates "real-world" conditions, where the classification algorithm learns an existing learning database, then several new solos arrive (the new database) and should be classified. Although this method is intuitive, it is not optimally informative when we deal with databases that differ quantitatively from each other.

In most of my experiments, for example, I use 5 solo databases. These databases contain a different number of instruments and a different number of windows (samples) for each solo.

Number of solos -
database #1:7, database #2:7, database #3:6, database #4:7, database #5:3

We can see that when the percentage of correctly classified solos is calculated per database and then averaged over the databases, database #5 with its 3 solos influences the grade as much as database #1 with its 7.

Regarding Windows classification -
The shortest solo clip is the "cello-2-italian" in database #2, with just 13 windows (the solo length is 4 seconds). Most solos are represented by 120 windows (30 seconds). This means that in the average Windows Grade, computed as the average percentage of the windows correctly classified in each database (ignoring which solos they belong to), the Piano solo from database #2 (120 windows) has 9.23 times the weight of the Cello solo from the same database.

A more informative grade calculation is to calculate the grade per musical instrument instead of per database:

<u>Solos</u>
Compute the solos percentage classified correctly for every musical instrument out of the total number of solos of this instrument. Compute the average of these results, i.e. the average percentage of solos classified correctly per instrument.

<u>Windows</u>
Compute the percentage of Windows correctly classified in every solo, then the average of these percentages for every instrument. Compute the average of these averages, i.e. the average Windows percentage classified correctly per instrument.

Even these computations are not optimal under Minus-1 DB - it matters which solos are grouped together in each database.
For example, let us say that a database which contains a piano solo also contains a Guitar solo which for some reason interferes a lot with Piano classifications. When this database is classified by the rest put together (Minus-1 DB), its Piano solo classification will benefit from the fact that the disturbing Guitar solo from the same database is not being learned.
Instead of removing a whole database each time, it might be better to remove just a single solo ("Minus-1 Solo") in every classification. The main drawback of this approach is that it will virtually eliminate the possibility of using Neural Networks for the classifications, as training a neural network the same number of times as the number of solos we have, would simply consume too much time.

<u>All this does not mean that the results of my experiments in this report are meaningless</u>.

I have compared the "per-DB" grades of many experiments with their corresponding "per-instrument" grades. The differences I found reached up to 3% (meaningful but not critical) and were relatively consistent.

Following are several casual examples for differences in results between these two grading methods.
In the Per-DB grades, 'X/Y' is the number of solos classified correctly in the database. It is followed by the Windows Grade for this database.
In the Per-instrument grades, we see the windows classification of every solo in its corresponding database. Y or N indicates whether this solo was classified correctly (a solo being classified correctly does not mean the Windows Grade is above 50%, it only means that the correct instrument got the highest Windows Grade among the instruments).

<u>Musical Instrument taxonomy, "5 databases", LDA+KNN, min-max normalization</u>

<u>Per-DB grades:</u>
7/7 76.66, 5/7 63.7, 5/6 69.1, 6/7 65.92, 3/3 79.46
mean: 0.88 (solos), 70.97 (windows)

<u>Per-instrument grades:</u>

Bassoon:	34.17	Y	43.33	N			67.5	Y		
Clarinet:	93.62	Y	5	N	87.5	Y	73.68	Y	71.67	Y
Flute:	82.5	Y	78.43	Y	20.83	N	99.17	Y		
Guitar:	100	Y	92.5	Y	77.27	Y	33.33	Y		
Piano:	50.83	Y	81.67	Y	89.17	Y	19.56	N	52.63	Y
Cello:	85.83	Y	76.92	Y	72.6	Y	63.64	Y		
Violin:	100	Y	88.33	Y	89.19	Y	100	Y	100	Y

Averages -

Bassoon:	48.33	0.67
Clarinet:	66.29	0.8
Flute:	70.23	0.75
Guitar:	75.77	1
Piano:	58.77	0.8
Cello:	74.75	1
Violin:	95.5	1

Mean 69.95 (windows), 0.86 (solos)

Results:

Solos	0.88 Per-DB, 0.86 Per-Instrument.
Windows	70.97 Per-DB, 69.95 Per-Instrument

We see that the difference is quite small.

Another example.

<u>Pizzicato/Sustain taxonomy, "5 databases", BP80, Zscore normalization, MSE 0.0004</u>

<u>Per-DB grades:</u>
7/7 98.83, 6/7 82.98, 6/6 93.7, 6/7 82.87, 2/3 87.54
mean (DB): 0.88 (solos), 89.18 (windows)

<u>Per-instrument grades:</u>

Bassoon:	99.17	Y	45.83	N			100	Y		
Clarinet:	100	Y	76.67	Y	100	Y	100	Y	97.5	Y
Flute:	100	Y	68.63	Y	97.5	Y	100	Y		
Guitar:	100	Y	99.17	Y	63.64	Y	75.83	Y		
Piano:	100	Y	97.5	Y	83.33	Y	0	N	40.35	N
Cello:	93.33	Y	100	Y	100	Y	61.36	Y		
Violin:	100	Y	100	Y	100	Y	100	Y	100	Y

Averages -

Bassoon:	81.67	0.67
Clarinet:	94.83	1
Flute:	91.53	1
Guitar:	84.66	1
Piano:	64.24	0.6
Cello:	88.67	1
Violin:	100	1

Mean: 86.51 (windows), 0.89 (solos)

Results:

Solos	0.88 Per-DB, 0.89 Per-Instrument.
Windows	89.18 Per-DB, 86.51 Per-Instrument

The difference in Solos is small, in Windows it is 2.67%.

<u>Conclusion</u>
In my future experiments I shall examine results both Per-DB (to be compared with the past results) and Per-Instrument (more informative).

Some possibilities for improvements

Classification methods

- Instead of using constant size "blind" cuts, attempt to perform transient detection in the solo (with Axel's module for example), and try to cut it into separate notes.
- These cuts could be classified in the "regular way" (descriptors and classification algorithms) or by using evolutionary models of the sounds with HMM's or recurrent neural networks.

Algorithms

- Try more classification algorithms - binary SVM, RCE networks, simple-Bayes.
- The results of several of these algorithms could be intersected (like in the corresponding section in this report).
- Try new feature selection algorithms (NLDA with kernel functions, ReliefF, CFS).

Descriptors

- Get rid manually of non-relevant descriptors, as we have seen that the feature selection algorithms are not perfect.
- Add new descriptors (for example wavelet decomposition, or try to distinguish between monophony/polyphony).
- Mix per-window descriptors with descriptors calculated on the whole solo for Solo Classifications.

Databases

- Use more than 120 windows from each solo, i.e. take longer clips than 30 seconds out of each solo. The results with clips of various lengths could be compared.
- Add more solos.
- Use several solos out of the same disk - up till now I have avoided it, as I wanted solos from different instruments and different recording conditions.
- Add new instruments, like human voice, etc.
- Create and add more "glued" solos (like "database #6"), containing sounds from other databases besides the ones I used.

- Try different sliding window stepping (other than 0.25 second).
- Try more uniform classification databases - maybe get rid of very short solos (like the 4 second Italian Cello), or restrict the solos to more standard playing techniques (without the "modern" solos or Ircam experimental recordings). Could be nice to have the same number of instruments in all the databases, but it is hard to find solos, and a pity to throw away the ones I have.

Removing bad samples

- Maybe it could be possible to find specific characteristics of misclassified samples (e.g. maybe most transitions are misclassified?). Try to detect such samples and remove them.

  For example - find all the misclassified samples in the learning group using LOO and KNN and then train a neural network to identify them against the ones which were correctly classified. Use this net to eliminate samples from the test group which are likely to be misclassified, and then classify the rest in the regular way.

- There are other methods for removing outliers besides MIQR and IQR which I used in this report. For example the LOO+KNN method which I used in the ICMC paper.
- Get rid of the clippings in the "glued" solos ("database #6").

Preliminary results of instrument recognition in duets

Arie Livshin using f0 detection by Chunghsin Yeh

Piece	Lengths	Bassoon	Clarinet	Flute	Guitar	Piano	Cello	Violin	Total %	**Solo**
Castelnuovo:	48.53% / 68.64	21.86	0	1.16	0	18.89	0.75	0	V 95.54	V 100.0
Sonatina	94 , 71.56	48.9	0	4.3	0	45.7	1.1	0	V 94.6	
Stockhausen:	72.25% / 32.2	0	19.59	19.55	0	0	0	0	V 100	V 100.0
Tierkreis	74 , 79.72	0	47.3	52.7	0	0	0	0	V 100	
Scelsi:	57.16% / 59.5	0	18.23	26.94	0	0	0	6.85	V 86.83	V 100.0
Suite	102 , 91.61	0	30.4	63.7	0	0	0	5.9	V 94.1	
Carter:	66.53% / 67.07	0	38.32	33.83	0	0	1.41	7.44	V 89.07	V 97.5
Esprit rude	98 , 140.77	0	51	38.8	0	0	2	8.2	V 89.8	
Ravel:	99.14% / 63.13	0	4.4	2.9	1.36	0	41.18	42.77	V 90.65	V 97.5
Sonata	184 , 244.25	0	0.5	1.6	0.5	0	48.4	48.9	V 97.3	
Martinu:	71.72% / 61.27	0	1.13	3.31	0.68	0	34.69	22.18	V 91.74	V 97.3
Duo	162 , 147.38	0	0.6	3.1	0.6	0	64.8	30.9	V 95.7	
Pachelbel:	98.91% / 65.58	0	13.92	53.74	9.57	0	50.97	2.9	V 79.86	V 95.1
Canon in D	182 / 242.08	0	7.1	46.7	5.5	0	39.6	1.1	V 86.3	
Procaccini:	76.12% / 39.89	16.55	0	0	0	16.53	0.68	0	V 97.98	V 94.4
Trois pieces	82 / 81	41.5	0	0	0	57.3	1.2	0	V 98.8	
Bach:	96.44% / 50.93	0	18.09	35.33	4.13	0	34.74	7.21	V 70.42	V 90.4
Cantata BWV	154 / 161.99	0	16.2	37.7	1.9	0	37.7	6.5	V 75.4	
Sculptured:	74.55% / 58.46	0	6.17	15.55	3.13	0	33.11	2.63	V 80.31	V 88.9
Fulfillment	146 , 125.98	0	11	24.7	3.4	0	58.2	2.7	V 82.9	
Ohana:	75.62% / 62.31	0	16.05	47.12	0	0	0	2.18	V 72.1	V 86.8
Flute duo	102 , 167.07	0	6.9	91.2	0	0	0	2	V 91.2	
Bach:	97.35% / 73.65	0	0	0	5.38	0	67.71	35.60	V 95.31	~~V~~ 86.3
Cantata BWV	210 / 229.38	0	0	0	1.4	0	74.3	24.3	V 98.6	
Pachelbel:	98.66% / 64.26	0	11.75	6.53	22.63	0	47.21	38.87	V 67.79	~~V~~ 84.6
Canon in D	178 , 226.84	0	6.7	3.4	11.8	0	44.9	33.1	V 78	
Idrs:	52.39% / 173.9	43.88	11.29	82.54	0.77	22.74	1.77	0	V 77.56	~~V~~ 59.4
Aria	208 / 209.66	23.1	7.2	55.3	0.5	12.5	1.4	0	V 78.4	
Copland:	51.58% / 71.56	0.73	7.89	8.16	10.20	17.28	13.29	0	~~V~~ 44.22	~~V~~ 45.5
Sonata	124 , 100.95	1.6	10.5	10.5	13.7	37.1	26.6	0	~~V~~ 47.6	
Guiliani:	23.40% / 68.12	1.16	5.26	10.36	6.92	0	5.8	0	V 58.58	~~V~~ 40.5
Iglou	50 , 36.87	4	18	42	16	0	20	0	~~V~~ 58	

First column ('Piece') – partial name of piece
Second column ('Lengths') – 'X/Y' and under it 'Z , W'.

X – net percentage of musical piece covered by chunks (without overlap).
Y – length of musical piece in seconds.
Z – total number of chunks and antichunks (each chunk is filtered twice – 'chunk' and 'anti-chunk', in order to recognize both instruments).
W – total time of chunks, including overlapping and anti-chunks.

Columns 3-9 (instrument names) – A and under it B. for each instrument:

A – recognized net length in seconds of this instrument ('Net length') – without chunk overlap.
B – percentage of chunks recognized as this instrument.
The actual instruments playing in this piece are marked in black.

Column 10 ('Total') – C and under it D.

C – percentage of recognized net length of the correct instruments in the total recognized net length of all instruments.
D – percentage of chunks recognized as the correct instruments.
V – means that the two most common recognized instruments are the correct instruments.
~~V~~ – Only one of the two most common recognized instruments is a correct one.

Column 11 ('Solo') – the percentage of 1 second pieces of the music classified as one of the correct instruments by the solo-recognition program used in my ICMC/DAFX articles. V and ~~V~~ mean the same as above.

Some conclusions from the table:
- We see in the percentage parameter in the ‘Lengths’ column that the coverage by chunks and antichunks is almost always much better than the coverage in the solo classification. With the solo-classifier each 1 second piece was classified only as a single instrument, so excluding some parts where only a single instrument plays, the coverage by the classified 1-second pieces is actually only 50% of the musical piece, and thus to compare chunk coverage and solo-classified coverage, the percentage covered by the solo-classifier should be devided by 2.
- We see in the ‘total’ and ‘solo’ columns, that majority detection of the correct instruments is much better with the chunk-classifier than the solo-classifier. Only in one piece I there was an instrument which did not get majority. **Majority detection is very important** – after the two instruments are correctly recognized, the piece could be segmented again, and this time using only the two correct instruments in the classifier (the learning group). BTW – this program can be also used with solo performances, so we actually deal here with instrument segmentation of **up-to** two instruments playing concurrently, where the instruments themselves could be polyphonic (e.g. the piano-bassoon examples).
- We see in the ‘total’ column, that grading using the number of chunks usually produces a better result than using the net lengths of each instrument. This hints that long chunks are not necessarily classified better than short ones.

24/01/05

Source Reduction Issues and Experiments
Report by Arie Livshin

Reminder and terminology

Basically, my current filtering and instrument-recognition process for duets is:

0. Run chunghsin's program to get the estimated 2-polyphony streams of partials for the musical piece.
1. Perform an STFT of the piece using the same windowing parameters as Chunghsin.
2. Using the 2-polyphony partial streams, find out which notes are sustained for at least 0.5 a second. The STFT frame sequences of these notes are called **chunks**.
3. For each chunk:
    - 3.1 Go through its frames and in the spectrum domain of each frame:
        - 3.1.1 Take the peaks with a height above a constant threshold.
        - 3.1.2 If a peak does not contain partials of the sustained note, set the values of its bins to zero.
    - 3.2. Perform section 3.1 again, but this time instead of trying to keep the partials of the sustained note in each chunk, try to keep the partials of the opposite stream. A chunk filtered in this reverse way is called an **anti-chunk**.
    - 3.3. Add the 2 previous unfiltered frames to the chunk and anti-chunk.
    - 3.4 Use ISTFT to convert the chunk and anti-chunk back to wave.
    - 3.5 Classify both the chunk and the anti-chunk. The classifier was trained on the "real-time" descriptors of solo-pieces – each solo was "blindly" cut into 1-second pieces with 50% overlap. Each descriptor is calculated by using a 60ms sliding window with a 66% overlap and averaged over the piece. The classification algorithm first uses Linear Discriminant Analysis and then K-Nearest-Neighbors.
4. We now have a segmentation of the musical piece into parts played by different musical instruments.

Note 1: Not all the musical piece is segmented. As we demand a chunk length of 0.5 second, some parts of the piece which contain shorter notes in both of the polyphonic channels (or where there were local f0-estimation errors), are not classified at all. The percentage of the piece which was classified (correctly or otherwise), is referred to as **coverage** of the piece.
Note 2: The **grade** of the piece is calculated by taking the length of the chunks classified as the correct instruments and dividing it into the total length of all the instruments. Note that I use a non-overlapping length for each instrument – e.g. if four 0.5-second overlapping chunks (and/or anti-chunks) are all classified as violin but cover together only 0.75 second (because they partially overlap), in the grade calculation this counts as 0.75 second and not as 2 seconds.

Prolonging chunks beyond their constant note limits

A chunk consists of several consecutive frames with a common note (f0 quantized to note boundaries). I try to add more consecutive frames to the chunk, even although they do not have the same f0 as the chunk, if they contain a pitch which is close enough to the pitch of the chunk, while the other pitch detected in these frames is far. I assume that the closer pitch probably still belongs to the same instrument as the rest of the chunk, because it seldom happens in duets that a note stream of one instrument suddenly jumps and crosses over the note stream of the other instrument.
This chunk-prolongation could benefit us, by providing:
1. A better coverage of the piece. As we only use chunks beginnibg with a minimum constant length (e.g. 05 second), shorter chunks that are prolonged by the closely-pitched frames can become "legitimate"-length chunks.
2. Longer chunks are recognized better than very short ones (which is the reason for imposing a minimum-length). For example it was seen that when the minimum chunk length goes down from 0.5-second to 0.25, the total grade declines.

Many variations of this algorithm were tried. Experiments have shown that although the piece coverage has significantly increased, the overall grade has declined.
Some examples:
1. Algorithm ver. 1 - Prolong a chunk with notes from neighboring frames which differ from the chunk-pitch by less than the notes of the other stream. Stop prolonging the chunk when its length is longer than 1-second. The motivation for this length restriction is that I am not interested in classifying the whole piece as a single chunk, and the 1-second limit was chosen because the classifier was trained on 1-second solo pieces.
The Aria for Flute and Bassoon (quite a complex piece which I like to use in tests) with a minimal chunk length of 0.5-second was originally classified producing a a grade of 77.5%[1], with the Flute getting 50.6% and the Bassoon with 26.9%. The coverage was only 52.39%.
After using the chunk-prolongation process, the grade went down to 71.44%, with Flute 44.47% and Bassoon 26.97%. The coverage went up to 82.93%.

2. Ver. 2 – Prolong a chunk with neighboring frames only if the pitch distance of the frame from the chunk-pitch is up to 1 tone, while the note from the other stream is estimated to be distanced by more than 2 tones (much more moderate than ver. 1).
The same Aria for Flute and Bassoon: The grade went up to 77.53% (from 77.5), while the coverage went up only to 58.48% (from 52.39).

This might seem as a nice improvement, but in other examples usually the grade was slightly decreased with Ver 2, while the coverage was only moderately increased.

Examples:

[1] You might notice that sometimes throughout this report the same musical piece might get different grades without an apparent reason. This is due to different parameters used in the filtering program, which do not have much to do with the current experiment, e.g. which version of Chunghsin's program I used for f0 estimation at the time or whether I added extra frames to the chunks. This usually means that the grades should be only compared locally, within each report section.

Bach BWV on Violin & Cello – chunk prolongation resulted in grade decrease to 94.7% from 95%, and coverage increase to 98.28% from 97.35%.
Copland, Piano & Clarinet (my “worst graded” duet) – with chunk-prolongation - grade 39.87% (was 43.7% without), coverage 58.74% (was 51.58%).

Experiments with other parameters did not produce better results. We have seen that chunk-prolongation naturally does give a better coverage, but the grades are decreased.
Some reasons for this decrease could be:
1. The note streams might cross occasionally, producing chunks containing notes of both instruments, thus confusing the classifier. This rarely happens and does not seem to be the main reason for the decrease in recognition rate.
2. Perhaps chunks with multiple notes are recognized worse than single-note chunks. This will be checked by performing a focused comparison of the average grades of chunks (contain single-notes) vs. anti-chunks (which contain multiple notes). Past experiments did not show such tendency of the anti-chunks, and we should also remember that the classifier was trained on blindly cut solo pieces where multiple notes in the same 1-second piece occurred all the time.
(A note about the grades – we should remember that the <u>minus-1 solo</u> recognition grade was only 85% with the real-time descriptor-set, so demanding a higher grade when classifying duet pieces is not reasonable.)

What do we prefer – better coverage or better grades? Should non-covered pieces considered as incorrectly classified?

<u>Using a One Vs. All solo classifier</u>

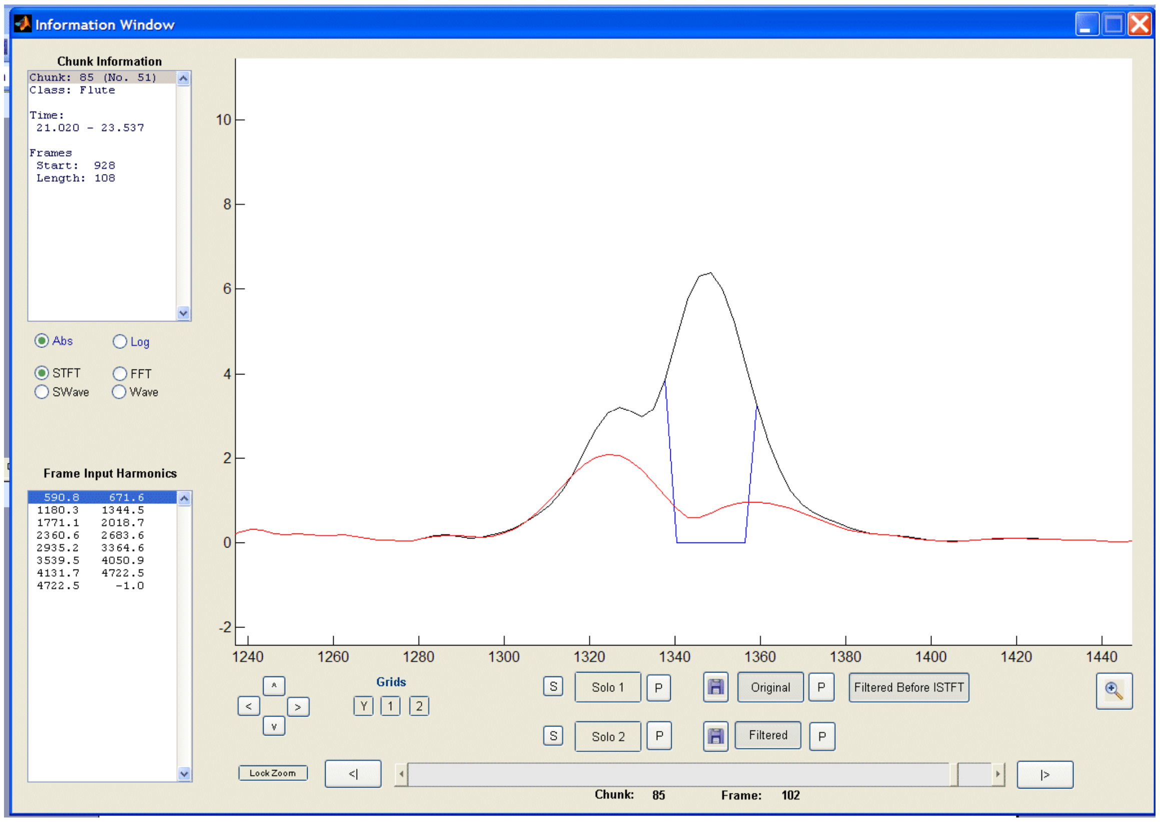
Currently I have a single solo classifier which was trained on all the instruments, thus performing a 7-way classification. Instead of using this, I tried using 7 classifiers, each trained on a single instrument against all the rest of them.
The motivation for this was that perhaps my classifier has weaknesses and gets confused by too much information with too many subtle differences, and that it will be easier for it to focus on a single instrument against all the rest.
In the ideal case only one of the 7 classifiers should recognize the 1-second musical piece as its distinguished instrunment (against all the other instruments) while the rest of the classifiers should not. What actually happens is that sometimes several classifiers might recognize the piece as their instrument, and sometimes none at all. In order to try to deal with that a confidence level was computed by dividing the number of knn neighbors of the selected classification, by the total number of knn neighbors (e.g. if 10 out of 15 neighbors say it is a flute, then it is classified as a flute with 2/3 confidence level). This does not always work, as it turns out that many classifications are very condifent and produce either a 1 or a 0 confidence level.
When several of the classifiers claim to recognize the piece as their instrument, the confidence level is used to decide. If all the classifiers reject a piece, the classifier with the lowest confidence level gets it (as it is the least sure that the piece does not belong to its instrument).

Performing a minus-1 solo evaluation with oneVsAll did not produce improved results (using LDA+KNN, “real-time features”, 1-second solo pieces with 50% overlap):
The grade went down from 85.24% with the regular allVsAll to 81.2% with oneVsAll.

I will try also OneVsOne, where a special classifier for each pair of instruments is trained. In SVM theory there are different net structures for all these cases (oneVsOne, oneVsAll and others), so it seems that there might be some advantages/disadvantages to the different types. We have also seen that the grade did change (although in a bad way), so this means that the different methods are not completely equivalent.

Non-perfect filtering

Let’s examine the following picture:



Chunk Information

```
Chunk: 85 (No. 51)
Class: Flute

Time:
 21.020 - 23.537

Frames
 Start:  928
 Length: 108
```

[✓ Abs] Log
[✓ STFT] FFT
SWave Wave

Frame Input Harmonics

```
 590.8     671.6
1180.3    1344.5
1771.1    2018.7
2360.6    2683.6
2935.2    3364.6
3539.5    4050.9
4131.7    4722.5
4722.5      -1.0
```

Chunk: 85 Frame: 102

The black line is the original signal. The blue one is my filtering intention – in this case of zeroing this peak. The red one is what wav actually produced by my zeroing – the red line was plotted after filtering, performing ISTFT and then analysing again with STFT.
It seems that the main reason for the difference between what I wanted to do and the ISTFT results, is the 75% overlap in the frames of the STFT. Although this peak might have been zeroed in this frame, but the following or the previous frames still have it, influencing this frame because of the overlap-addition of the frames. Sometimes, the opposite happens when a peak gets smaller because of filtering in neighboring frames.

Is there a way to prevent this? i.e. not having interference from neighboring frames, e.g. concatenating only the middles of the windowed frames after the STFT without their margins? Any other ways to prevent influence?

I have consulted Axel and he suggested to reduce this phenomenon by using only a 50% overlap between the frames. The problem with this is that Chunghsin does use a 75% overlap and it would be a pity to lose the middle-frame partial information.

<u>Peak selection algorithms</u>
Another thing Axel suggested is that instead of using a constant threshold for peak selection, to select peaks by finding the local maxima and minima of the spectrum and divide into peaks by taking a local maxima which is 1.5 times the height of the local minimas on its sides. This way we will have more peaks, the problem of "common" peaks will appear less and thus the difference in peak height with neighboring frames will be smaller (as a common peak will not be suddenly created because a "bad" partial moves into the vicinity of a "good" peak) and affect less the ISTFT.

I have tried using this peak-selection algorithm instead of my thresholded one and the results were very bad. Sounds of Cello and Violin were classified as Clarinet, the Flute as Bassoon, etc. This actually reminds very much what happened when we tried to resynthesize a violin or a Cello chunk using additive Analysis/Synthesis which resulted in a clarinet classification. Using Axel's peak selection method, most of the noise was filtered out. Also, double headed peaks were splitted. As we already seen in the past, my current descriptors are very influenced by the noise, and having a sound reduced to its harmonics produces a bad classification.

I have also tried a hybrid peak-selection approach – first to select the peaks using my threshold algorithm and then divide these into sub-peaks with Axel's method. This way the noise was retained, while double headed peaks were split and higher resolution was gained there.
This has resulted in similar grades to the my constant threshold method, although with a slightly reduced grade. For example Aria for Flute and Basson went down from 70.27% to 70.1%.
A possible explanation for this reduction, is that splitting of double-headed peaks is not necessarily a good thing, because sometimes, as in the case of the violin for example, peaks with double heads are natural and do not indicate actual polyphony.

<u>The noise problem</u>
Here I term low-volume non-harmonic spectral content as "noise", not meaning necessarily high-frequency sounds. As we saw in the experiments with additive analysis/resynthesis and Axel's peak selection algorithm, it seems that the noise is a very important ingredient for my classifier.
A problem with this is that if I only try to separate the sources using the estimated partial information I get from Chunghsin and I deal with instruments which are differentiated mostly by their noise, I have no way to fully filter one of them out, and only partial filtering produces many times unexpected results.

Possible solutions:
- I could discard the noise and use only partials and their peaks. This could be done by filtering the noise out of the learning and test databases by doing additive analysis/synthesis, by removing low-volume partials, or use descriptors which are immune to low-level noise. In either case it's a pity to lose all this information.
- Find some ways to measure the noise as well as filter specific characteristics of it out, and somehow relate this to the harmonic information we have. For example, first separate the noise from the harmonics and then compute separate discriptors for both, while finding some way to filter certain characteristics out of the noise.

How does the noise differ for different instruments? What orthogonal properties does it have?

<u>Dealing with peaks containing more than one partial ("common peaks")</u>

As explained in the "reminder" section, in the filtering process I keep peaks containing partials of the note that I want to retain and zero all the rest. The kept peaks might contain also partials of the "bad" note, the note playing in polyphony with the note I wish to keep, which I wish to remove.

I could try to reduce the height of these "common" peaks to a size which is more similar to their size in the preferred instrument, and thus to reduce the influences of the "bad" peaks on the classification.

Some experimental algorithms to deal with this:

Alg. 1. We check whether the frames neighboring to the one containing the common peak, have another peak in the same place which belongs only to the preferred instrument without the competing partial. If we find that, we compute the relation of its height to the height of the peaks in that frame containing its neighboring partials (these neighboring peaks should not be "common" peaks) and then reduce our common peak in relation to its neighboring partial peaks according to that relation.

Alg. 2. Check the peaks with the neighboring partials (in the same frame), and if they do not contain partials of the bad note (they're not "common peaks"), reduce the size of the common peak relatively to them, using interpolation, or even naively, an average.

Besides size reduction, we should also consider the shape of the peak – the partial from the "bad" note has changes it and just reducing the peak size still retains this deformity. One possibility is to create an artificial peak out of a sinusoid with the right height and window function, and then replace the common peak with the artificial one. Another similar option is to create an artificial peak of the partial we wish to remove, and then subtract this from the common peak – the advantage of this approach is that we do not remove extra information or noise, but in this case we need to find out what size the "bad" partial is supposed to be.

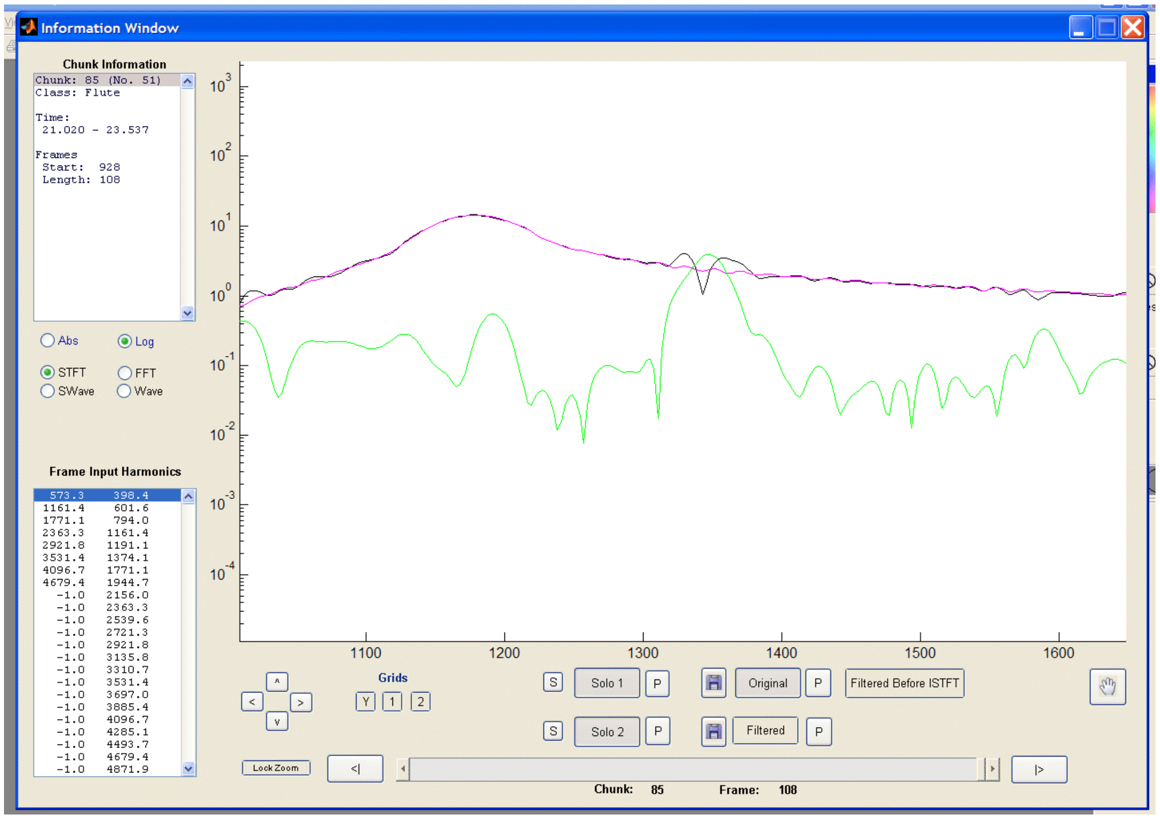
I have just began experimenting with dealing with "common" peaks, and the first experiment is very simple. If a peak is "common" and the peaks with its neighboring partials are not "common", change the height of this peak to the average height of its neighbors. Although it is obvious that the results of this would not be accurate, it is some place to start.

Concerning the form of the peak – there could be double-headed peaks where the height of the “bad” partial could be much higher than the “good” one which could retain a deformed peak even after the size reduction. Double-headed peaks occur due to my usage of a constant threshold and taking everything above as a single peak.
To eliminate many double-headed peaks I could use the “hybrid” peak selection algorithm – it will select both peak heads as separate peaks, and thus optimally, the height of the peak head with the “good” partial would be reduced to the average of the neighbors while the peak head with the bad partial should disappear.

Results – although this process resulted in many chunks and frames containing peaks that were reduced in size, the total grade did not change dramatically.
For example the Bach BWV with Violin and Cello, was reduced to 89.01% from 90.73% (using minimum chunk length of 0.25-second and with a 99.62% coverage). Both the Violin and Cello have produced a slightly lower grade. A similar change appeared in the grade of other duets.
The reasons for these grades could be different – on one hand the average of its neighbors is not a good measure for a peaks height, and I did not use any special shaping of the peak form. On the other hand, as the grade was not really affected very much after my different manipulations, it seems that anyway the relative sparsity of the common peaks in a frame makes their importance for the classification not very dramatic.

<u>A possible problem with polyphony peak analysis</u>
A good way to evaluate a common-peak treatment is by doing an artificial mix of two solos and then comparing the original peaks to the ones resulting from the filtering process.

The following picture shows an example view of a clip of a frame of an artificial piece, where two solos, magenta and green, were mixed together to produce the black one.



```
Chunk: 85 (No. 51)
Class: Flute

Time:
 21.020 - 23.537

Frames
 Start:  928
 Length: 108
```

Abs [✓ Log]

[✓ STFT] FFT SWave Wave

```
  573.3    398.4
 1161.4    601.6
 1771.1    794.0
 2363.3   1161.4
 2921.8   1191.1
 3531.4   1374.1
 4096.7   1771.1
 4679.4   1944.7
   -1.0   2156.0
   -1.0   2363.3
   -1.0   2539.6
   -1.0   2721.3
   -1.0   2921.8
   -1.0   3135.8
   -1.0   3310.7
   -1.0   3531.4
   -1.0   3697.0
   -1.0   3885.4
   -1.0   4096.7
   -1.0   4285.1
   -1.0   4493.7
   -1.0   4679.4
   -1.0   4871.9
```

We see that for some reason the joined peak (around 1350) does not look at all as an addition of the two solos, but rather as a subtraction, with extra peaks at its sides which appeared without an apparent reason.
It is almost impossible to tell what the goal is (without having the originals) – which peak to remove and which to keep? The real signal I was trying to achieve with my filtering, the green one, is none of the peaks in the resulting mix. This creates a sertious analysis problem with polyphonial music, unless the mixing program ("adobe audition") could be somehow blamed for this (shifting the time somehow or doing something complex to the signals). Maybe the phase is to blame?

<u>Size of learned solo-pieces</u>

I have used a classifier trained on 1-second solo pieces in order to classify variable-length filtered duet chunks of 0.5 second or more. The coverage of the musical piece naturally increases as the minimum size required of the chunks decreases. On the other hand, very small chunks are relatively hard to classify correctly and thus the total recognition rate ("non-overlapping") goes down as the number of very small chunks increases.
For example, The Aria for Flute and Bassoon, with chunks of 0.5-second or more, gives us a coverage of 51.23% with a grade of 74.17%, while with chunks of 0.25-second or more, the coverage rises dramatically to 92.41% with a reduced grade of 68.6%.
As the coverage is important as well as the grade and I wanted to reduce the minimal chunk size but still have a good grade, I thought of using a solo Database trained on smaller solo pieces than 1-second. I assumed that using a classifier trained on solo-pieces with a similar size to the classified chunks should result in better recognition, as the

learning set will be more similar to the test set. If this was proved to be the case, i.e. that for smaller chunks it is better to use a solo database trained on shorter solo-pieces, I could have several classifiers trained on different sized solo pieces, and classify each chunk with the classifier best adapted to its size.
I have created another solo database, this time with solo pieces of 0.5-second each, instead of 1-second, with 0.5 second hop (0% overlap), with the "real-time" descriptors. Although the minus-1 solo grade of this DB is only 82.39% compared to 85.24% of the 1-second DB, it could still be expected that it should classify 0.5-second chunks better than the 1-second DB, as they resemble it more.

It turns out that this is not the case. For example the Aria for Flute and Bassoon with a minimal chunk length of 0.25-second produced a grade of 67.4% with the 0.5-second solo DB, and 70.52% with the 1-second solo DB.

Let's examine the distribution of different chunk-sizes and their recognition score in Aria for Flute and Bassoon. The chunk length column is the size of the chunk in seconds, e.g. 0.2 means 0.1-0.2, 0.4 means 0.3 to 0.4 etc. The total #chunks column is the number of chunks of this size, correct1 is how many of them were correctly classified with the 1-second solo database, score1 is the relative score for them. correct 0.5 and score 0.5 is the same for the 0.5-second solo database.

chunk Length	total#chunks	correct 1	score 1	correct 0.5	score 0.5
0.1	0	0	NaN	0	NaN
0.2	0	0	NaN	0	NaN
0.3	488	341	0.69877	341	0.69877
0.4	228	171	0.75	167	0.73246
0.5	94	71	0.75532	67	0.71277
0.6	50	37	0.74	35	0.7
0.7	30	27	0.9	26	0.86667
0.8	20	14	0.7	12	0.6
0.9	12	9	0.75	10	0.83333
1	2	2	1	2	1
1.1	8	7	0.875	7	0.875
1.2	10	6	0.6	6	0.6
1.3	20	20	1	18	0.9
1.4	6	5	0.83333	5	0.83333
1.5	4	3	0.75	3	0.75
1.6	4	3	0.75	3	0.75
1.7	6	6	1	5	0.83333
1.8	6	5	0.83333	5	0.83333
1.9	0	0	NaN	0	NaN
2	2	1	0.5	1	0.5
2.1	0	0	NaN	0	NaN
2.2	2	2	1	2	1
2.3	0	0	NaN	0	NaN
2.4	0	0	NaN	0	NaN
2.5	0	0	NaN	0	NaN
2.6	0	0	NaN	0	NaN
2.7	0	0	NaN	0	NaN
2.8	4	2	0.5	1	0.25
2.9	0	0	NaN	0	NaN
3	0	0	NaN	0	NaN

We can see that the 1-second DB always wins (except in one fluke). It seems that there is no need to keep several different length solo DBs, as at least in this case the 1-second DB outperforms the 0.5-second regardless of chunk lengths.

One of the possible explanations to this somewhat peculiar phenomenon where 0.5-sec pieces are classified better by 1-sec pieces than by 0.5-sec pieces, could be because pieces of 1-second length produce more uniform descriptors due to the averaging, and perhaps lda+knn, due to its linear nature, has more problems of placing the different groups of 0.5-sec descriptors into the same categories than the 1-second pieces (and we did see also a decline in the minus-1 solo results when going down from 1-sec pieces to 0.5-sec). I will try using a neural-network instead of the lda+knn in order to check this possible linearity problem.

VNC

VNC is a multi-platform software which allows several people to remotely connect to a single computer, view its screen and control it with their mice and keyboard.
This proved to be a very effective tool for my telecommuting. I have contacted Axel using VNC and the phone, ran my classification program and showed him several graphs which exemplified problems with my filtering. He controlled my program while we both viewed the screen and gave me an advice on how to improve my filtering.

19/05/2005

Report by Arie Livshin

Choosing the Best Features for Instrument Recognition in Multi-Instrumental Music

Reminder and Terminology

Basically, my current filtering and instrument-recognition process for each duet (could also be adapted for general multi-instrumental music) is:

1. f0-estimation:
    Run Chunghsin's multiple-f0 estimation program and get 2 streams of partials.
2. STFT:
    Perform an STFT of the duet using the same window sizes used by the multiple-f0 estimation program.
3. Define chunks:
    Using the 2-polyphony partial streams, find out which notes are sustained for at least 0.5 a second. The STFT frame sequences of these notes are called **chunks**.
4. Filter and classify chunks:
    For each chunk:
    4.1 Go through its frames and in the spectral domain of each frame:
        4.1.1 Find the peaks with energy above a constant threshold.
        4.1.2 Set the bins of peaks that do not contain partials of the sustained note to zero.
    4.2. Perform section 4.1 again, but this time instead of keeping the partials of the sustained note in each chunk, keep the partials of the opposite stream. A chunk filtered in this reverse way is called an **anti-chunk**.
    4.3. Add the 2 previous unfiltered frames to the chunk and anti-chunk.
    4.4 Use ISTFT to convert the chunk and anti-chunk back to wave.
    4.5 Classify both the chunk and the anti-chunk. The classifier was trained on the "real-time" feature set computed on solo-pieces – each solo was "blindly" cut into 1-second pieces with 50% overlap. Each descriptor is calculated by using a 60ms sliding window with a 66% overlap and averaged over the piece. The classification algorithm first uses LDA (Linear Discriminant Analysis) and then KNN (K-Nearest-Neighbors).
5. Results:
    We have obtained a segmentation of the musical piece into parts played by different musical instruments.

Some notes:

The main reason for using solo-pieces as the learning group instead of filtered duet chunks is that solo-pieces contain only a single instrument while filtered chunks might not be filtered perfectly and may contain sounds from several instruments.

In cases where one of the instruments is ‘highly-polyphonic’, e.g. guitar or piano, I rely on an article by Egging and Brown[1] which shows that in accompanied solos usually the volume of the solo instrument is higher than of the accompaniment. This usually causes the f0-estimation program to recognize a note played by the solo instrument as one of the two notes it reports.

Throughout this report I will be referring to both chunks and anti-chunks simply as ‘chunks’ as the distinction makes no difference for the current topic.

Challenge

In standard classification procedures, the classifier is trained on a learning set which contains samples of the same ‘kind’ as the ones in the test set, i.e. the statistical conclusions which could be made by analyzing the learned information apply also for the classified information.
In our case this is not exactly so. In order to filter out one of the playing instruments in a chunk I use information on its f0 and partials reported to me by the f0-detection program, information which is not necessarily correct. Afterwards I use my filtering algorithm, which is not perfect, to remove one of the sounds. These imperfections prevents in many cases the filtered chunks from having completely ‘clean’, one-instrument notes as present in the learning set – the solos.
As mentioned in ‘reminder and terminology’, the classifier first reduces the dimensions of the test set and improves its class separability by multiplying it by an LDA matrix of descriptor-weights which was computed using the learning set. But does an LDA computed on solo-pieces really produce ideal weights for classification of filtered-chunks? If this was the case, we could have been able to add as many new descriptors as we want without performing any manual selection and just rely on the LDA to correctly weigh them and prevent ‘bad’ descriptors from influencing the classification.

The problem is that we do not know whether the solos are actually that similar to filtered-chunks. It is possible that some of the descriptors are good for solo classification but bad for filtered chunks, therefore getting high LDA weights and producing misleading results with the duets.

Mission

To choose a feature set which will produce the best results of classifying filtered chunks by solo-pieces.

Feature Sets

My **Full** feature set was written by Geoffroy Peeters[2] and consists of 62 **feature** vectors that produce a total of 513 different parameters (which I will refer to as ‘**descriptors**’). The **R-T** (‘real-time’) feature set is a subset of features from the Full set which can be computed very fast (about 12 times faster than the Full set) and which I originally

[1] J. Eggink and G. J. Brown, “Instrument recognition in accompanied sonatas and concertos,” To appear in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP’04),* 2004.
[2] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," 2003. URL: http://www.ircam.fr/anasyn/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf

selected in order to perform instrument recognition in solos in real-time. This set has produced a slightly lower grade for solos (85.24% 'minus-1 solo' grade) compared with the Full feature set (88.13%). I have been using the R-T set for most of my duet classification experiments due to its comparatively fast computation speed.

Algorithms

In order to choose the best features[3] I introduce two variations of a brutal force algorithm:

Forward Algorithm

```
1.   A={Full feature set}
2.   B=ø
3.   max_grade=0
4.   for every feature a ∈ A use the features in { a ∪ B } to classify the filtered
     chunks by the solo-pieces. Let a' be the feature which produced the highest grade
     among these classifications. This grade is max_grade'
5.   if (max_grade' > max_grade) then
             (max_grade = max_grade') , (A=A \ a') , (B=B ∪ a')
     otherwise go to 7
6.   if (A ≠ ø) go to 4
7.   output B – it contains the 'ideal' feature set
```

Backward Algorithm

```
1.   B={Full feature set}
2.   max_grade = grade of classifying the filtered chunks by the solo-pieces using the
     Full feature set
3.   for every feature b ∈ B use the features in { B \ b } to classify the filtered chunks
     by the solo-pieces. Let b' be the feature which removal produced the highest grade
     among these classifications. This grade is max_grade'
4.   if (max_grade' > max_grade) then
           (max_grade = max_grade') , (B=B \ b')
     otherwise go to 6
5.   if (B ≠ ø) go to 3
6.   output B – it contains the 'ideal' feature set
```

Grading

The grade I wished to maximize is "instrument grade", which is the average grade per musical instrument. The reason I did not just use "musical-piece grade", i.e. the average grade per duet, is to allow instruments play in different numbers of duets[4].

[3] Another possibility instead of choosing feature vectors, is to use a more detailed selection algorithm and choose the best descriptors, e.g. choosing the best spectral centroid type or the best MFCC coefficients, etc. Running such an algorithm requires a very long time (starting from 513 possibilities per round instead of only 62) while the theoretical advantages of such a specific selection are not obvious.

[4] Initially I did use an average grade per musical-piece instead of average grade per musical-instrument. The algorithm 'cheated' by choosing features which gave the highest grades to the most popular instruments (like cello) 'on the back' of more rare instruments (like guitar) which received much lower grades, thus considerably raising the average grade per musical-piece - up to 88.95%.

The grade of an instrument in a duet is calculated by dividing the total non-overlapping[5] length of the filtered chunks classified as this instrument by the total length of all the instruments in the duet.
To compute the average 'instrument grade' I first calculate the average grade for each instrument over the duets where it actually plays, and then compute the average of all these averages.

Example:
Let us suppose that we have a duet of clarinet and flute and that the classifier has reported that the clarinet has played for 10 seconds (non-overlapping), flute 5, piano 10 and violin 20 seconds (misclassifications). The grades will be:
Flute: 5/(10+5+10+20)=11%
Clarinet: 10/(10+5+10+20)=22%
The total grade of this duet: 11+22=33%

We can see that the maximal grade a duet can get is 100%, while on average the **maximum grade for an instrument is 50[6]**.

Classification of a chunk as either one of the instruments playing in the duet is currently considered as correct classification, e.g. in our example, I do not check whether a chunk classified as flute does not actually contain clarinet.

Databases
The duets include the following instruments: Bassoon (appears in 3 duets), flute (9), clarinet (5), violin (5), cello (8), piano (3), guitar (2) – a total of 18 duets. The reason I have used (at this stage) this specific set of duets is in order to be able to compare the results directly with many of my former results (including 2 articles) obtained from experimenting with this same set of duets. As we can see, the number of duets in this set is quite small and each instrument appears in a different number of duets. The implications of this will be discussed later in the report.
The learning set is much bigger and includes a total of 108 solos.

The multiple f0-detection program was run with two different analysis window sizes that Chunghsin recommended to try - 92ms and 184ms. I used two different window sizes in

[5] The multiple-f0 detection program attempts to find in each duet-frame two notes that play concurrently. This means that the classifier might classify every frame up to 4 times – two chunks resulting from the 2-polyphony notes that the f0-detection found and their corresponding anti-chunks. Several of these classifications usually yield the same class, in which cases the overall length of the non-overlapping frames of an instrument is used for the grade calculation. On the other hand, when overlapping chunks are classified as belonging to different instruments, each of these chunks is counted in the total length of each of these instruments.
[6] Although an instrument could certainly get in a duet a grade G which is higher than 50%, but then the grade of the second instrument will be equal or less than (100 – G) as the total grade of all the instruments in a duo is 100%. BTW – in one of the duets, the Ohana flute duo, the flute could theoretically get the whole 100% as this is a duet with two identical instruments – flute. I have not removed this duet for reasons of consistency with previous experiments. Fortunately, a single flute duet does not affect the average grade very much.

order to see which one is better and also to have 2 variations of every algorithm run, in order to check its stability.
It is important to keep in mind that I have changed the window sizes only in the f0-detection program and in the filtering process, but not in the feature computation routines. The feature computations are almost always performed using a sliding window of 60ms with a hop size of 20ms, and the final descriptors of the piece are the mean and standard deviation computed over these sliding windows.

Results

Table 1 presents the instrument-grades of different feature sets. The columns marked 'Fore' contain the grades obtained from using the feature sets selected by the Forward algorithm while the 'Back' columns show the results with the feature sets selected by the Backward algorithm. The 'ALL' columns show the results when the full feature set was used for the classification. The 'R-T' columns show the results when the Real-Time feature set was used.
The numbers – 92 and 184, appearing below the title of each column are the window size in milliseconds used in the f0-estimation program and in the filtering algorithm.

Instrument	**#**	**Fore 184**	**Back 184**	**R-T 184**	**ALL 184**		**Fore 92**	**Back 92**	**R-T 92**	**ALL 92**
Bassoon	3	38.92	44.97	38.28	45.31		37.76	50.59	45.71	48.96
Flute	9	46.52	46.49	44.64	44.40		45.47	49.29	47.06	46.81
Clarinet	5	35.24	36.42	34.24	36.46		31.41	41.28	35.25	39.65
violin	5	40.30	42.03	37.82	41.27		46.00	47.47	42.79	44.59
cello	8	50.60	47.91	48.32	46.06		36.15	37.95	40.89	37.68
Piano	3	43.90	42.99	40.46	40.72		43.31	33.53	33.29	29.63
Guitar (!)	2	50.68	36.97	29.34	31.90		58.87	38.44	24.45	31.72
Instr. average		**43.74**	**42.54**	**39.01**	**40.88**		**42.71**	**42.65**	**38.49**	**39.86**

Table 1. Average instrument-grades

When discussing the results I will usually compare columns with the same window size.
We see in Table 1 that the average grades from lowest to highest are obtained by using the following feature sets: R-T, ALL, 'Back' and 'Fore'. We see that as could be expected from the algorithm's simplicity, I did succeed in raising the average instrument grade. I wish to remind again that the maximum average grade for an instrument is 50%.

When examining the grades of each instrument we see that the results in the 'Back" columns, obtained by using the features selected by the Backward algorithms, seem as direct improvement over the 'ALL' columns – results obtained from using the Full feature set, almost with all the instruments (in Back 184 there are tiny decreases in the bassoon and clarinet but otherwise all the instruments have improved in both the Back 92 and 184).
Unlike the feature sets of the Backward algorithm, the results obtained from the Forward algorithm's feature sets in the 'Fore' columns, are more different than the results in the 'ALL' columns, with some of the instruments recognized considerably better and some worse, when all in all the average grade is the highest among all the feature sets.

The grades with the Real-Time set ('R-T' columns) are usually lower than the ones produced using the Full set ('ALL columns') although we can see that in some instruments there are improvements when using the Real-Time sets, for example the piano in 'R-T 92'. We have already seen in a previous paper where I classified solo-pieces by solo-pieces, that the Real-Time set, although being a subset of the Full feature set, has reduced the grades of all the instruments except the flute, which actually had a better grade. This improvement of the flute grade has shown that LDA, which I use for dimension reduction and class 'tightening' before performing the KNN classification, does not perform feature selection which is ideal for all the classes (at least not when KNN follows it).
In this report, where the learning set is solo-pieces but the test set is filtered chunks, sets that differ from each other more than in the paper where I used solo-pieces in both the learning and test sets, we can expect even more problems with the LDA feature weighting.

In order for my algorithms to be optimal, it is required that the grade could be improved monotonously by adding or removing features until the perfect feature set is reached. Unfortunately when we compare the columns of the 'Fore' and 'Back' algorithms we can see that this is not the case – the final grades that these algorithms produce are not identical. If it was possible to monotonically increase the grade until a perfect set was reached, the 'Fore' and 'Back' algorithms would have continued running until they both meet and stop at the feature set which produces the highest grade.
Already when I was classifying single tones and using 66% / 33% evaluation method (before discovering the dire need for minus-1 evaluation) and the learning and test sets were really similar, we could see that the resulting graphs of my GDE feature selection algorithm, which in each round was using LDA in order to get rid of the worst descriptor, were not monotonic and had small accents and descents. This has clearly shown that LDA is not perfect for feature selection. Again, if LDA was perfect, removing a descriptor should have never resulted in grade improvement.

Even if LDA was optimal, the lack of full compatibility among the learning and test sets in this report could have still caused problems for my algorithms[7]. When adding a feature which is actually good for distinguishing between instruments in the duets, the LDA which calculates the weight matrix using the solos, could choose such unfortunate weights that it will actually decrease the grade of the duet recognition instead of improving it.
An example of this phenomenon has occurred while the 'Forward 92' algorithm was run. Adding the 'DTi_zcr_v' feature in the second round has reduced the grade from 35.77 to 35.33 and thus the feature was rejected. On the last round of the algorithm, adding the same feature has surprisingly increased the grade from 42.43 to 42.71 and so it was added. This shows feature selection inconsistency – a feature should either always improve or worsen the results, but not both depending on the other features present.

I will return to these problems later.

[7] On the other hand if the learning and test sets were known to be completely compatible, there was no need for this report.

Duet	Fore 184	Back 184	R-T 184	ALL 184		Fore 92	Back 92	R-T 92	ALL 92
13_fagotizm_procaccini _bassoon_piano.cnd	**95.28 bs: 53.53 po: 41.75**	**100.00 bs: 50.78 po: 49.22**	**88.01 bs: 49.04 po: 38.97**	**100.00 bs: 58.83 po: 41.17**		**88.74 bs: 46.59 po: 42.15**	**98.60 bs: 66.57 po: 32.03**	**89.1 bs: 57.67 po: 31.43**	**89.86 bs: 63.09 po: 26.77**
6_fagotzim_castelnuovo _bassoon_piano.cnd	**98.50 bs: 41.12 po: 57.38**	**96.12 bs: 50.89 po: 45.23**	**92.81 bs: 41.75 po: 51.06**	**93.50 bs: 46.27 po: 47.23**		**93.60 bs: 32.54 po: 61.06**	**93.13 bs: 51.76 po: 41.37**	**92.66 bs: 54.97 po: 37.69**	**87.82 bs: 52.33 po: 35.50**
Aria_flute_bassoon _noSpaceEnd.cnd	**69.18 fl: 47.06 bs: 22.12**	**80.34 fl: 47.09 bs: 33.25**	**71.84 bs: 24.04 fl: 47.8**	**77.22 fl: 46.38 bs: 30.84**		**88.53 bs: 34.15 fl: 54.38**	**87.72 bs: 33.43 fl: 54.29**	**75 bs: 24.48 fl: 50.5**	**85.28 bs: 31.46 fl: 53.82**
Giuliani_Flute_ Guitar.cnd	**69.79 fl: 40.04 gu: 29.75**	**55.24 fl: 40.46 gu: 14.78**	**63.55 fl: 40.89 gu: 22.66**	**50.29 fl: 40.14 gu: 10.14**		**80.62 fl: 32.49 gu: 48.13**	**73.80 fl: 46.82 gu: 26.98**	**48.18 fl: 33.76 gu: 14.42**	**62.45 fl: 46.82 gu: 15.63**
bach_flute_cello.cnd	**75.10 fl: 37.84 vc: 37.27**	**76.45 fl: 44.09 vc: 32.36**	**74.36 fl: 37.26 vc: 37.1**	**73.31 fl: 40.94 vc: 32.37**		**73.46 fl: 47.00 vc: 26.45**	**77.18 fl: 51.85 vc: 25.33**	**74.44 fl: 38.82 vc: 35.61**	**71.93 fl: 45.48 vc: 26.45**
bach_violin_cello.cnd	**99.50 vl: 46.82 vc: 52.68**	**94.88 vl: 39.80 vc: 55.08**	**93.85 vl: 33.21 vc: 67.24**	**89.13 vl: 41.38 vc: 47.75**		**83.40 vl: 44.04 vc: 39.36**	**96.56 vl: 49.46 vc: 47.10**	**90.33 vl: 38.28 vc: 52.05**	**92.91 vl: 46.69 vc: 46.22**
bohuslav_violin_cello.cnd	**97.59 vl: 35.47 vc: 62.12**	**96.60 vl: 34.87 vc: 61.74**	**89.73 vl: 32.17 vc: 57.56**	**97.73 vl: 31.69 vc: 66.04**		**91.59 vc: 32.49 vl: 59.10**	**92.77 vc: 45.56 vl: 47.21**	**89.73 vl: 34.86 vc: 54.57**	**92.43 vc: 55.22 vl: 37.21**
carter_flute_clarinet.cnd	**90.37 fl: 42.57 cl: 47.80**	**88.24 fl: 43.77 cl: 44.47**	**85.87 fl: 44.03 cl: 41.84**	**87.86 fl: 41.01 cl: 46.85**		**93.65 fl: 48.07 cl: 45.58**	**86.97 fl: 46.30 cl: 40.68**	**88.92 fl: 46.14 cl: 42.78**	**85.27 fl: 43.47 cl: 41.80**
copland_piano_clarinet.cnd	**45.93 cl: 13.34 po: 32.58**	**47.91 cl: 13.38 po: 34.53**	**45.52 po: 31.34 cl: 14.18**	**49.46 cl: 15.68 po: 33.78**		**47.37 po: 26.73 cl: 20.65**	**57.66 po: 27.18 cl: 30.47**	**56.74 po: 30.76 cl: 25.98**	**56.49 po: 26.63 cl: 29.86**
feidman_clarinet_guitar.cnd	**92.84 cl: 21.23 gu: 71.61**	**86.51 cl: 27.35 gu: 59.16**	**69.17 cl: 33.14 gu: 36.03**	**81.92 cl: 28.27 gu: 53.65**		**84.82 cl: 15.22 gu: 69.61**	**82.30 cl: 32.39 gu: 49.90**	**70.53 cl: 36.04 gu: 34.49**	**81.05 cl: 33.24 gu: 47.82**
karlheinz_flute_clarinet.cnd	**96.41 fl: 51.57 cl: 44.84**	**100.00 fl: 42.96 cl: 57.04**	**100 fl: 54.22 cl: 45.78**	**100.00 fl: 47.97 cl: 52.03**		**100.00 fl: 52.81 cl: 47.19**	**100.00 fl: 40.79 cl: 59.21**	**97.75 fl: 53.05 cl: 44.7**	**97.33 fl: 45.75 cl: 51.59**
kirchner_violin_cello.cnd	**89.26 vl: 46.00 vc: 43.26**	**92.58 vl: 44.71 vc: 47.87**	**85.65 vl: 46.41 vc: 40.04**	**92.72 vl: 45.25 vc: 47.47**		**91.60 vl: 48.19 vc: 43.41**	**89.13 vl: 58.40 vc: 30.73**	**83.01 vl: 56.72 vc: 26.29**	**87.21 vl: 56.57 vc: 30.64**
ohana_flute_duo.cnd	**87.31 fl: 87.31**	**75.69 fl: 75.69**	**79.83 fl: 79.83**	**64.59 fl: 64.59**		**67.08 fl: 67.08**	**66.14 fl: 66.14**	**71.9 fl: 71.9**	**62.01 fl: 62.01**
pachelbell_flute_cello.cnd	**81.74 fl: 37.65 vc: 44.09**	**83.47 fl: 43.73 vc: 39.74**	**81.59 fl: 39.44 vc: 42.15**	**75.54 fl: 40.51 vc: 35.03**		**61.68 fl: 35.51 vc: 26.17**	**77.63 fl: 49.72 vc: 27.91**	**79.36 fl: 45.19 vc: 34.17**	**66.57 fl: 46.60 vc: 19.96**
pachelbell_violin_cello.cnd	**92.12 vl: 29.62 vc: 62.50**	**83.63 vl: 37.18 vc: 46.45**	**74.23 vl: 31.49 vc: 42.74**	**71.85 vl: 33.69 vc: 38.16**		**69.47 vl: 34.74 vc: 34.74**	**67.64 vl: 32.97 vc: 34.67**	**64.87 vl: 34.72 vc: 30.15**	**65.80 vl: 33.85 vc: 31.95**
ravel_violin_cello.cnd	**95.85 vl: 43.59 vc: 52.27**	**98.40 vl: 53.59 vc: 44.81**	**93.85 vl: 45.83 vc: 48.02**	**97.27 vl: 54.35 vc: 42.92**		**85.12 vl: 43.91 vc: 41.21**	**91.30 vl: 49.33 vc: 41.97**	**90.98 vl: 49.36 vc: 41.62**	**86.66 vl: 48.62 vc: 38.04**
scelsi_flute_clarinet.cnd	**84.49 fl: 35.49 cl: 49.00**	**81.72 fl: 41.85 cl: 39.87**	**80.8 fl: 44.55 cl: 36.25**	**82.70 fl: 43.24 cl: 39.47**		**76.75 fl: 48.35 cl: 28.40**	**91.19 fl: 47.56 cl: 43.63**	**83.71 fl: 56.98 cl: 26.73**	**87.31 fl: 45.56 cl: 41.75**
sculptured_cello_flute.cnd	**89.77 fl: 39.15 vc: 50.62**	**94.05 fl: 38.80 vc: 55.26**	**76.4 fl: 24.71 vc: 51.69**	**93.56 fl: 34.77 vc: 58.78**		**68.87 fl: 23.51 vc: 45.36**	**90.48 fl: 40.12 vc: 50.35**	**79.86 fl: 27.2 vc: 52.66**	**84.68 fl: 31.73 vc: 52.95**
AVERAGE	**86.17**	**85.10**	**80.39**	**82.15**		**80.35**	**84.46**	**79.28**	**80.17**

Table 2. Grades of individual duets

Selected Features

DEi_harmosig_v	**total harmonic energy of additive-resynthesized signal**	**DPi_scm_m**	**spectral crest**
DEi_noise_v	**total noise energy**	**DPi_sfm_m**	**spectral flatness**
DEi_noisesig_v	**total noise energy of additive-resynthesized signal**	**DPi_skew_v**	**perceptual spectral skewness**
DEi_tot_v	**total energy**	**DPi_slope_v**	**perceptual spectral slope**
DHg_mod_f	**f0 frequency modulation**	**DPi_specloud_m**	**relative specific loudness**
DHi_kurto_v	**harmonic spectral kurtosis**	**DSi_sc_v**	**spectral centroid**
DHi_oeratio_v	**odd to even harmonic ratio**	**DSi_ss_v**	**spectral spread**
DPi_DDmfcc_m	**delta delta MFCC**	**DSi_variation_v**	**spectral variation**
DPi_decs_v	**perceptual spectral decrease**	**DTg_ed**	**temporal effective duration**
DPi_loud_v	**loudness**	**DTg_tc**	**temporal centroid**
DPi_mfcc_m	**MFCC**	**DTi_xcorr_m**	**signal auto-correlation function**
DPi_oeratio_v	**odd to even percdeptual band ratio**	**DTi_zcr_v**	**signal zero-crossing rate**

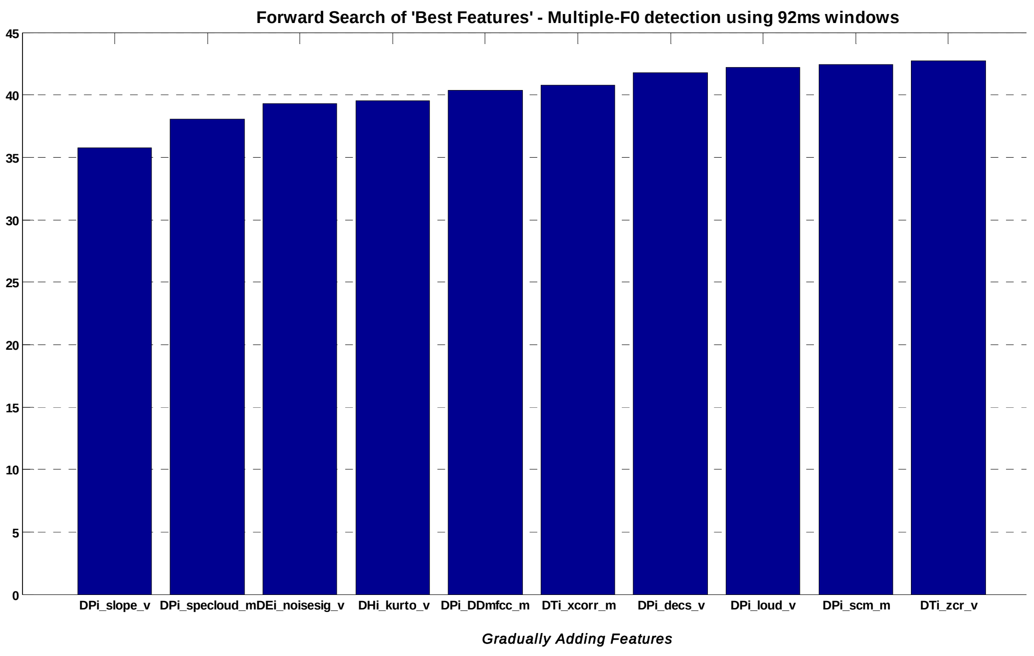
Table 3. Feature names mentioned in the report and their meanings

<u>Features selected by 'Forward 92' (total 10 features) and the associated grades</u>
Presented by the order they were added by the algorithm

DPi_slope_v	(35.77)	DPi_specloud_m	(38.09)	DEi_noisesig_v	(39.29)
DHi_kurto_v	(39.56)	DPi_DDmfcc_m	(40.38)	DTi_xcorr_m	(40.75)
DPi_decs_v	(41.77)	DPi_loud_v	(42.21)	DPi_scm_m	(42.43)
DTi_zcr_v	(42.71)				

The 'best K' selected from the range of 1..80, and used for KNN in the last classification (with all the above features present), was 32.

Figure 1



We see in Figure 1 that most of the grade – 35.77, was already achieved with the first feature (DPi_slope_v – perceptual spectral slope). Afterwards, there is a gradual increase of another 7 points as more features are added.

Features selected by 'Forward 184' (total 7 features) and the associated grades
Presented by the order they were added by the algorithm

DPi_slope_v	(35.81)	DPi_mfcc_m	(37.8)	DSi_ss_v	(39.63)
DSi_sc_v	(42.23)	DPi_DDmfcc_m	(43.03)	DHi_oeratio_v	(43.52)
DEi_noisesig_v	(43.74)				

The 'best K' selected from the range of 1..80, and used for KNN in the last classification (with all the above features present), was 73.

We can see that in 'Forward 184' as in 'Forward 92', 'DPi_slope_v' was selected as the first feature resulting in an average grade around 35% with a moderate increase of another 8 points when adding additional features. Only 7 features were selected in 'Forward 184' compared to 10 with 'Forward 92'.

We can see that although the grades are similar in Forward 92 and 184 the different window sizes used in the f0-detection program and the filtering process have resulted in different features being selected. This has affected the recognition results considerably as could be seen by comparing the 'Fore 92' and 'Fore 184' columns in Table 1 and Table 2. For example the cello at 'Fore 184' has received an average grade of 50.6 but only 36.15 with 'Fore 92'.
It is not easy to understand by looking at the tables and the features why certain window sizes used with Chunghsin's f0-detection program had specific influences on the recognition of certain instruments – it seems most likely that it was due to popular note lengths and perhaps also popular pitches of the sounds in the specific duets.

The following table portrays in more details the influence of every feature on the recognition of different instruments.

	DPi_ slope_v	DPi_ mfcc_m	DSi_ ss_v	DSi_ sc_v	DPi_ DDmfcc_m	DHi_ oeratio_v	DEi_ noisesig_v
Bassoon (3)	38.02	31.96	29.38	40.75	37.14	40.29	38.92
Clarinet (5)	21.78	32.37	32.15	31.44	31.87	32.52	35.24
Flute (9)	30.89	37.66	48.63	47.70	47.21	46.16	46.52
Guitar (2)	34.95	43.94	34.64	44.64	50.93	49.38	50.68
Piano (3)	46.67	47.00	48.33	44.07	45.40	45.47	43.90
Cello (8)	29.54	35.92	46.57	48.00	52.43	52.12	50.60
Violin (5)	48.84	35.76	37.72	39.00	36.25	38.68	40.30
Average	35.81	37.80	39.63	42.23	43.03	43.52	43.74

Table 4. Forward 184 instrument grades, while adding each feature (left to right)

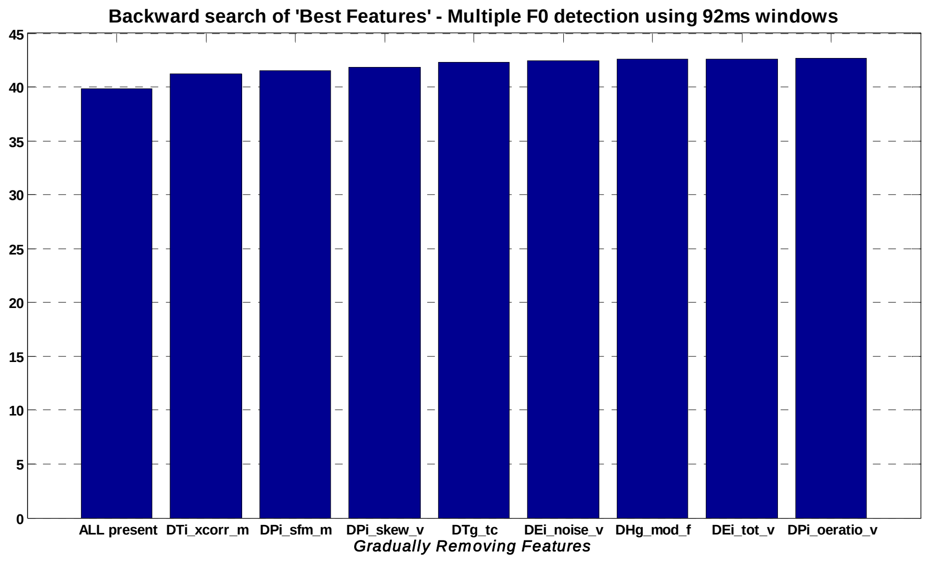
We can see again that adding features increases the grade of some instruments but lowers others while the average grade rises. Explaining why addition of a specific feature increases the grade of some instruments while decreasing others is out of the scope of this report. On the other hand, if I did want to check how descriptors influence the recognition of different instruments, I would do that by performing feature selection for one-vs.-one classifications using solo recordings, and checking which features are the most important for distinguishing between every pair of instruments.

Features removed from the Full set by ‘Backward 92’ (total 8 features) and the associated grades
Presented by the order they were removed

ALL features	(39.86)	DTi_xcorr_m	(41.2)	DPi_sfm_m	(41.51)
DPi_skew_v	(41.87)	DTg_tc	(42.3)	DEi_noise_v	(42.44)
DHg_mod_f	(42.56)	DEi_tot_v	(42.58)	DPi_oeratio_v	(42.65)

The first grade (‘ALL features) is the result of using all the features for classification, the second grade is the result of removing ‘DTi_xcorr_m’, etc.
The ‘best K’ selected from the range 1..80 and used for KNN in the last classification round, was 50.

Figure 2



If we examine the feature names carefully, we can see another proof that my algorithms are not optimal – the second feature **removed** in ‘Backward 92’ - ‘DT_xcorr_m’, is also the sixth feature **selected** in ‘Forward 92’ (see

Figure **1**). Like most of the other features, this feature does not have a very strong impact on the grade (adding it in ‘Forward 92’ has increased the average grade by 0.37 points while removing it in ‘Backward 92’ has increased the grade by 1.34 points) and because of the lack of perfect monotonicity using LDA, ‘DT_xcorr_m’ was selected by two algorithms which are supposed to oppose each other.

Features removed from the Full set by 'Backward 184' (total 6 features) and the associated grades
Presented by the order they were removed

ALL features	(40.87)	DTg_tc	(41.22)	DPi_skew_v	(41.65)
DTi_xcorr_m	(42.02)	DTg_ed	(42.23)	DSi_variation_v	(42.35)
DEi_harmosig_v	(42.54)				

The 'best K' selected from the range 1..80 and used for KNN in the last classification round, was 19.

Conclusions

Improvement in Grades
My goal was to find the subset of features that will allow me to best recognize instruments in filtered chunks while using a learning set of solo-pieces.
I indeed managed to raise the recognition rate using the proposed algorithms. The improvement in grade compared to using the Full feature set is less than 3 points on average (see Table 1) which is nice but not very dramatic. One of the main reasons for this moderate increase is the simple fact that the grade calculated when using all of the features (the Full set) was not very bad to begin with, which by the way shows that 'after all is said and done', there is quite a statistical compatibility between the learning set of solo-pieces and the test set of filtered duet-chunks, although not a perfect one.

Limited and non-homogenic test set
A major disadvantage of the experiments in this report is the small size of the test database and the fact that each instrument plays in a different number of duets. This set of duets most probably did not represent well the possible sounds that could be made by each instrument. For this reason, currently I doubt whether I should switch to using any one of the resulting subsets of features in future classifications in which there will be new multi-instrumental databases, with more instruments and musical pieces than the current duet set. On the other hand, when I will have these new and diverse databases, I could easily use the framework of the presented experiments in order to optimize the results and obtain small feature sets that will produce higher grades, not only for duets but also for more complicated multi-instrumental music. The resulting feature sets depend not only on the learning and test sets but also on the classification process itself – if I change the filtering algorithm or use another classification algorithm, a feature set computed before making such changes will produce less optimal results than before.

Speed Considerations
In order to save much time, in experiments which I will have to perform many times it seems worthwhile to continue working with the Real-Time set as its calculation is about 12 times faster that the Full set or any other sets in which some of the descriptors use the 'additiv' utility. A possible optimization could be instead of using 'additiv' in the feature

calculation routines, to use partials information supplied by Chunghsin's multiple-f0 detection program, a program I need to run anyway. A problem with removing 'additiv' is that there are many descriptors which use the synthesized and noise files 'additiv' creates. Having to create these files 'manually' will consume extra time.

Non-Monotonicity
Against my methods we could argue that in order for my algorithms to be optimal we need to assume that it is possible to improve the average grade monotonically by adding features (as in the 'Forward' algorithm) or subtracting them (in 'Backward') until an optimal feature set is reached. We have seen that this is not the case, as the sets obtained from the 'Forward' and 'Backward' algorithms are different. An easy but unpractical way to find the optimal set could be to check all the possible feature combinations in order to find the one which produces the best grade. Unfortunately, such a process would take too much time.

By the way, we need to remember that even if we could check all the possible descriptor sets then we would obtain a set which is *optimal for training on our solos and classifying our filtered chunks with LDA+KNN*. As LDA produces 'semi-optimal' weights for classifying the solo-pieces and 'knows nothing' about the filtered chunks of the duets, it is possible that the filtered chunks would have been classified better by the same solo-pieces if we used different weights for the descriptors than the ones LDA supplies.

Pre-Knowledge of the Test Set
Another claim against my methods could be that I am using knowledge of the test set which renders my algorithms 'illegal'. It is true that I search for an optimal feature set by choosing features which result in the best recognition of our known test set, but we do not have here a case of actual over-fitting or self-classification, as the classifier does not learn the test set at all (the filtered chunks) but rather a completely different learning set (the solo-pieces).
It could have been somewhat 'cleaner' to use 'minus-1' evaluation, i.e. choosing first the best feature set by using a subset of the multi-instrumental music and then classifying the rest of the musical pieces with this feature set and showing that the resulting grade is higher than when I have been using the Full feature set. Such a test repeated on a high number of subsets would indeed show both the generality of the selected features as well as keeping the selection algorithms 'cleaner' by not exposing the test sets to them.
It is currently hard to perform such experiments. On one hand such an algorithm would take too much time to run, while on the other hand the number of duets is much too small at this stage to perform experiments with duet subsets.

Multiple f0-detection and filtering window sizes
Regarding the window sizes used with Chunghsin's multiple-f0 detection program and my filtering algorithm – I cannot deduce globally which window size is better - 92ms or 184ms, the two sizes Chunghsin recommended. Although the 184ms does result in a slightly higher average grade, but overall the average grades for each instrument are very different when using the two window sizes (see Table 1) and therefore it is hard to detect an actual supremacy of a certain window size.

It might have been possible, after approximation of which instruments play in a musical-piece (e.g. by having obvious majority to certain classifications of the chunks) and estimating the common note lengths (by transient detection) to improve the instrument segmentation by repeating the f0-detection process with the best fitting window-sizes for certain instruments and certain note lengths. This seems quite complicated.
We could even try using a dynamic window size which changes in different parts of the musical piece depending on context, but this seems to me as something Chunghsin, who knows his algorithms well enough, may consider and in which I doubt whether I could give good advices to him.

<u>Using Filtered Chunks in the Learning Set</u>
If we want to use filtered chunks in the *learning set* (exclusively or in addition to the solo-pieces) we could either try to use a regular, single class per sample approach, or try to implement a multi-class per sample method.
In order to use a single-sample approach, we need first to listen to each filtered chunk in order to determine which instrument it contains (or use the musical score if we have it).
Such classification will post a problem, as many filtered chunks are not filtered perfectly and thus it is hard to classify them as a single instrument – especially piano and guitar chunks, for which the f0-estimation program does not report all the notes in the polyphony and where at least in half of the chunks the filtering algorithm tries to remove one of the piano (for example) notes thus leaving behind a chunk with multi-instrumental content.

A hybrid between single-class per sample and multi-class approaches is to add new 'combination' classes, where each such class symbolizes a chunk which contains two instruments (or more). The problem here is how to determine which chunks to classify as belonging to these combination classes. For example if we have a chunk for a flute-bassoon duet, where almost all of the flute sound has been filtered out and mostly only bassoon remains. Should this chunk be classified as bassoon or as flute+bassoon? How will the classifying algorithm (LDA in our case) deal with the fact that such a bassoon+flute chunk seems to have descriptor values which resemble too much the bassoon class, while other bassoon+flute chunks might actually resemble very much the flute class?

Although I do not know any existing **multi-valued classification** algorithms, it is possible to invent classification algorithms which will allow samples in the learning group to belong to more than one class while classifying the test group samples into single classes (which is what I need).
For example, I suggest a variation on KNN, where each sample in the learning group belongs to a vector of classes (e.g. a duet chunk could belong to two instruments classes while a solo-piece belongs only to one). In order to classify a sample, I take its K nearest neighbors and choose the most common class among them. Although I wouldn't be able to use standard LDA for dimension reduction because of its requirement for single class per sample, I could still use PCA or other non-supervised dimension reduction algorithms before performing my KNN.

It seems to me that classification algorithms which use class modeling, like neural networks or Gaussian classifiers are harder to adapt to this multi-class approach than KNN. For example if we try to give a neural network weighted classification for duo chunks (e.g. ‘0.5’ for flute and ‘0.5’ for bassoon) the network might have a problem of modeling both of these classes, as it is not clear why a sample which seems (by its descriptors) to belong to the bassoon class is suddenly classified as ‘0.5’ flute.

Next Stages of My Work

- I understand that there is a requirement to create a solo database for the Music-Discover project. I will create one with Slim Essid as soon as possible.

- I shall start working on instrument recognition in multi-instrumental music with 3 or 4 instruments playing concurrently. This will require among other things, the addition of more instruments, especially ones that are popular in chamber music, like Oboe and Contrabass. Highly polyphonic instruments like cembalo, piano and guitar, although treated up to now just as any other instrument, due to the extra f0-detection and filtering challenges they pose will require to be dealt with separately (the f0 detection does not work well with very polyphonic music and with closely harmonic sounds and so causes my filtering not to work well).

Convergence towards writing my PhD thesis

I am very interested in getting instructions from you on how I should proceed in order to converge towards writing my PhD thesis in order to finish my PhD in a reasonable time (about a year).

I have a few preliminary questions about the different thesis parts:

-- Regarding the introduction and background:
'Motivation for the subject of instrument recognition – I have already written about that in the DMDTS scholarship proposal and have also added newer technological motivations in the white paper I have submitted to the IEEE signal processing magazine, such as performing smart searches in MP3 personal players, etc.
History of the research – I shall update the history of musical instrument recognition I have written in the DMDTS proposal, which currently lacks research done in the last year and perhaps also add a few older articles which are missing.

Should I add more subjects to the introduction?

-- Results:
Could a reasonable recognition of instruments in chamber music (let’s say up to 4 instruments) as well as what I already did and published up to day be enough for the PhD?

-- Theory:
I am not completely sure what the thesis should include in the theoretical field. For example, could it suffice for the thesis if I show and explain why certain descriptors allow distinguishing between specific instruments, discuss the instruments' structure and the differences between them, and explain the filtering and classification techniques I have used to get my results?

Which other tasks must I perform in order to converge towards finishing of my PhD? What should be my next stages? Would it be possible for us to create a (more or less) concrete time table?

Harmonic Resynthesis vs. Source Reduction
Report of work in progress

My Ismir 2006 paper shown (among other things) that for separate tones, using solely the harmonic series is enough for achieving high instrument recognition rates. As Chunghsin's f0-estimation program estimates the harmonic serieses in polyphonic music, it is natural now to try performing instrument recognition in multi-instrumental music using solely the harmonic serieses.

My different approaches to instrument recognition in multi-instrumental music

<u>**"Source Reduction"**</u>

-- Use Chunghsin's multiple-f0 program to find the different notes in a musical piece
-- For each detected note over a minimal length:
--- Filter out everything except its partials
--- Classify this "cleansed" note using solos as the training set.

<u>**"Harmonic Resynthesis"**</u>

-- Use Chunghsin's multiple-f0 program to find the different notes in a musical piece
-- For each detected note over a minimal length:
--- resynthesize it using its estimated harmonic series
--- classify it using a learning set consisting from notes resynthesized from harmonic serieses of solos.

Theoretical Comparison

-- Both methods rely on a multiple-f0 detection program (currently Chunghsin's).
-- The Resynthesis method is simpler as it doesn't require filtering. It can be concluded from my Ismir2006 paper that if the multiple-f0 detection is good, the Resynthesis method should work well. The Reduction method requires <u>also</u> the filtering to be good.
-- The Reduction method has less strict demands from the f0-estimation program than the Resynthesis method; it requires less precision in the partial frequencies and doesn't require any partial amplitudes estimation:
The filtering in the Reduction method leaves the STFT peaks intact – it doesn't try to surgically remove each partial of an undesired note, but rather to remove whole peaks which do not contain partials of the "desired" note inside them, the note we wish to keep. This method, while rude, is also somewhat "safe", as it mostly leaves partials of the desired note untouched and only removes "undesired" information. This means that the Reduction method does not require the f0-estimation program to supply energy levels of the partials, which are very hard to estimate in cases of partial collisions, but only frequencies, while the Resynthesis method requires amplitudes of partials for its synthesis.
This also means that the f0-estimation program has some freedom of error regarding the partial frequencies, as long as they still "lend" inside the partial peaks.
In my ICMC04 paper, I have shown that if the training set is solos and the test set is duos, then 1-second pieces of the duos <u>even without any filtering</u> will still be mostly classified

as one of the participating instruments in the duo. This means for the Reduction method, that even if the filtering process didn't filter much due to proximity of "bad partials" to "good partials", so they reside in the same peak, one of the correct instruments is still likely to be identified (at least in duos).

Current "Harmonic Resynthesis" Results

I have used 5 instruments: Bassoon, Clarinet, Flute, Cello and Violin; The reason for this is that my Solo database includes 7 instruments, from which I removed the two polyphonic instruments – piano and guitar in order to allow simple limits on the concurrent number of notes for the f0-estimation program.

Solos (real performances)

The process:
-- Chunghsin's f0-estimation program was run on the solos in my solo database (without the polyphonic guitar and piano).
-- All notes in the estimation results which are at least 0.25-second long (to prevent accidental estimation errors) were found.
-- The notes are resynthesized using their estimated harmonic information.
-- The descriptors are computed on these notes.
-- The notes of the solos are classified using the "minus-1 solo" method, in which each solo is classified by the rest together.

"Minus-1 Solo" recognition results using resynthesized solo notes:

Instrument	Bassoon	Clarinet	Flute	Cello	Violin	**Average**
Average rate	89.83%	81.68%	86.81%	78.39%	75.22%	82.39%

These results are lower than the average rate for non-synthesized, original solo-pieces (average 87.42% for 5 instruments, 88.13 with 7 instruments).
It should be noted that the classified samples were selected very differently in the old and new experiments - while the previous solo recognition method sequentially cut 1-second pieces out of the solos, the current method looks for continuous notes (of at least 0.25-second) in the f0-estimation results, and then resynthesizes them. Due to the considerable number of errors in the f0-estimation process, in spite of its much lower sample-size limits, the resynthesized database consists of only 7204 samples, while the original solo database had 15367 samples (of these 5 instruments).
The results show that while the Resynthesis method is working for solos, the results are not "too good".
Perhaps adding more solos to the database can improve the results, of the Violin for example. This will be checked.

Duos (real performances)

Duo Name	Bassoon	Clarinet	Flute	Cello	Violin	Resynth	*Reduction
Aria flute bassoon	179	11	149	11	5	92.2% V	77.5% V
Bach Violin Cello	1			71	92	99.4% V	95.0% V
Bach Flute Cello	2	13	52	36	6	80.7% V	70.4% V
Carter Flute Clarinet	1	39	34	7	2	87.9% V	89.1% V
Karlheinz Flute Clarinet	1	19	30	1		96.0% V	100% V
Ohana Flute x 2		10	42	9	4	64.6% V	72.1% V
Pachelbell Flute Cello	1	8	33	40	1	87.9% V	79.9% V
Pachelbell Violin Cello	1			53	30	98.8% V	67.8% V
Ravel Violin Cello	1			53	22	98.7% V	90.7% V
Scelsi Flute Clarinet	5	33	75	18	3	80.6% V	86.8% V
Sculptured Flute Cello		26	31	74	2	78.9% V	80.3% V
Average						87.8% V	82.7% V

* As the Resynthesis and Reduction experiments were quite different, the Reduction results are brought for performance reference, not for complete and "fair" comparison of the methods.

Except the last column, the table shows Harmonic Resynthesis results of duos. Columns 2-6 show the number "classification points", i.e. the number of "notes", classified as each instrument. Column 7 shows the average "multi-class" recognition rate, which is the percentage of "notes" recognized as either one of the correct instruments. If both of the playing instruments got the highest marks, i.e. the instruments playing in the duo could be identified, then a 'V' mark is put next to the average score.
Column 8 shows results obtained by the Source Reduction method. The results are somewhat hard to compare, as they were computed on a different number of Notes (the Reduction method used a minimum of 0.5 second notes + the note anti-chunks, while the Resynthesis used a minimum of 0.25 second). Also, the Reduction results were computed using 7 instruments in the learning set instead of 5. Note that more instruments in the learning set do not necessarily mean lower results, as they could help reducing confusion between somewhat similar classes.

We can see from the table that the Resynthesis method seems not too bad, as it has outperformed on average the Reduction method by 5%. Both methods recognized the correct playing instruments in every piece.

Duos (artificial mixes of solos)

As it is rather hard to obtain real duo performances with our desired instruments, artificial mixes of 2 solos were made to represent different instrument combinations and the

coupling of solos to create each duo was done randomly. The same mixes were used both with the Resynthesis and Reduction methods.
Here the results of the Reduction method are even harder to compare to the Resynthesis method. In the Reduction method, f0-estimation results performed on the solos were used to select notes of at least 0.25-second length from the solo-mixes. This was done to reduce errors in classification originating in incorrect f0-estimation. In the Resynthesis method however, the note boundaries were taken from f0-estimation results ran on the mixed solos, which are much less precise.

Method	**min Len**		**Multi class**	**Solo-Instr. Identified**	**Mix-Instr. Identified**
***Reduction**	**Min 0.5**		**83.44%**	**75.00%**	**70.59%**
Resynthesis	**Min 0.25**		**81.36%**	**88.97%**	**73.53%**

Total number of mixes: 34. Number of instruments: 5 for both methods.

* - Again, the Reduction results are brought for reference only, not for complete and “fair” comparison.

The “Multi-class” grade shows how many of the Notes were classified as either one of the playing instruments. The “Solo-Instr. Identified” column shows how many of the correct instruments got a majority classification and thus could be recognized correctly as one of the participating instruments in the mix. The “Mix-Instr. Identified” column shows in how many pieces both playing instruments got a majority grade (same as the ‘V’ mark in the previous table).

We can see that again, the Resynthesis method seems to be working. The Solo-Instr grade is much higher in the Resynthesis method - there seems to be less competition between specific instrument pairs than in the Resynthesis method, meaning that when there’s a misclassification it spreads more randomly, producing higher majority grades for correct instruments.

Conclusions
It seems that at least for 2-instrument pieces, the Resynthesis method produces intelligible results.

Regarding Further Work
- In order to run the Resynthesis method on 3 to 5 voices, a more exact grading is needed, one which will tell whether a note is indeed classified as its playing instrument, and not just as one of the instruments which play in the piece. Otherwise, a “multi-class” grade with 3 to 5 participating instruments out of 5 possible instruments, will absurdly rise with the number of participating instruments instead of declining.

To do such exact evaluation, I could try either manually listening and marking each note (which is not always easy when listening to Clarinet/Flute duos for example), or search the notes located in the mix, inside f0-estimation data ran on the original solos in order to find which is the original instrument.

- I was thinking to include multi-instrumental Harmonic Resynthesis results as the final addition to my PhD and also extend my Ismir 2006 paper into a journal article by adding such multi-instrumental results (Ismir 2006 only has separate tones). Perhaps for that purpose, even the “real-performance” duo results are already sufficient.

- Initially, it seems that for the PhD Thesis it would be nice to have Resynthesis results of up to 5 voices, not just duos. However, as I already shown that the method works OK with duos (and very well with separate tones), it is obvious that for higher number of instruments playing concurrently, the results will depend further solely on the quality of the multiple-f0 estimation program, and have nothing to do anymore with my methods.

What do you think?

Arie

Harmonic Resynthesis vs. Source Reduction (part 2 of 2) – Solo Mixtures

My Ismir 2006 paper shown (among other things) that for separate tones, using solely the harmonic series is enough for achieving high instrument recognition rates. As Chunghsin's f0-estimation program estimates the harmonic serieses in polyphonic music, it is natural now to try performing instrument recognition in multi-instrumental music using solely the harmonic serieses.
In the previous report I have presented results for resynthesized solos and real duos. In this report I shall use the resynthesized solo database in order to classify mixes of solos from 2 to 5 concurrent voices.

Reminder - My different approaches to instrument recognition in multi-instrumental music

<u>"Source Reduction (SR)"</u>

-- Use Chunghsin's multiple-f0 program to find the different notes in a musical piece
-- For each note which was detected over a minimal number of continuous frames (vibrato/glissando are allowed):
--- Filter out everything except the STFT peaks which contain its partials
--- Classify this "cleansed" note using a solo database as the training set.

<u>"Harmonic Resynthesis (HR)"</u>

-- Use Chunghsin's multiple-f0 program to find the different notes in a musical piece
-- For each detected note (vibrato/glissando are allowed) over a minimal length:
--- Resynthesize it using its estimated harmonic series (frequencies and amplitudes)
--- classify it using a learning set consisting from notes resynthesized from harmonic serieses of solos.

I have used LDA+KNN for the classification.

See previous report for more details.

Solo Mixtures

It is very difficult to obtain recordings of real music along with its <u>precise</u> transcription; This is required in order to perform exact instrument recognition evaluation. On the other hand, music resynthesized from MIDI files, while producing exact performances of the symbolic notation, is somewhat too easy to perform instrument recognition on, as the sounds are very similar to each other, lacking articulations and synthetically clean. Therefore, evaluation with resynthesized MIDI files will not indicate well the ability of the recognition program to handle real music.

Artificially mixed real solo performances are a compromise; On the negative side, the solo sounds do not influence each other and do not create real, common room echo. Unfortunately also, musical composition rules could not be assumed to apply to the mixtures as the mixed solos have no relevancy to each other. On the other hand, all the instrument articulations are present and the files could be quite precisely labeled.

The instrument labeling was performed in the following way:
- Chunghsin's F0-detection program was ran on the original solos and on the solo mixtures.
- Notes detected in the mixtures which lasted for at least a specific length of time (e.g. 0.25 of a second or more), were searched in the f0-detection data of the solos mixed in that mixture, in the appropriate temporal locations.
- When a note was found to belong to a certain solo in the mix it was labeled by the same instrument as that solo.
- Notes found only in the mixture but not in the solos were discarded.

This labeling method seems to work rather well. I have listened to different notes taken from the solo-mixes and they were always labeled correctly by this process. The percentage of discarded, unfound notes, was quite small and will be indicated in the results-table below.

Independent Evaluation

Before classifying each solo-mixture, the solos which were mixed into it are removed from the learning database.

Grading

Several different grade types were calculated in order to show different strengths of the recognition approaches:

- Instrument Grade
-- Each note is checked whether it is classified correctly as its instrument label.
-- An average grade for each participating instrument is computed for each solo-mixture.
-- An average grade per instrument is reported over all the mixtures.

The Instrument Grade shows how well each instrument is classified, and confusion matrices could be computed. I have the confusion matrices of all the experiments in this report and could send them to you if you're interested.

- Pieces Grade
-- Each note is checked whether it is classified correctly as its instrument label.
-- The average recognition rate per note is computed for each mixture.
-- The average recognition rate per note is reported over all the mixtures.

The Pieces Grade differs from the Instrument Grade because the number of notes performed by various instruments in a mixture is completely different.

- "Either" Grade
This is a version of the Pieces Grade, where a note is considered to be classified correctly if it is classified as either one of the instruments mixed in the mixture.

The Either Grade attempts to produce a grading which diminishes the influence of recognition errors among participating instruments, such as produced by bad filtering or incorrect partial-amplitude estimation. When I listened to different resynthesized and filtered notes it indeed happen sometimes that I heard both instruments playing concurrently in a classified sample.

- "Vote" Grade

This is a complex grade which basic idea is (without going into the gory details):

-- Produce a sorted list of the instruments in a mixture by the number of samples each of the instruments "got" from the classification.
-- If correct instruments got the highest places in the list, each gets a point. If some participating instruments got however the same number of "votes" as some incorrect instruments, a point is divided among all of the instruments with the same number of "votes".

For example, if we had a clarinet and bassoon mixture and the number of samples different instruments received by the classification were:

Bassoon:	8 samples
Clarinet:	4
Flute:	4
Violin:	4
Cello:	0

Then bassoon which has the majority got 100% (a "point"), and "takes" one of the 2 places reserved for correct instruments (this is a duo, so 2 instruments are playing).
Clarinet however has 4 samples, like the incorrect flute and violin, and thus it gets "number of remaining places"/"number of instruments with same number of votes"=1/3=33.3%. Average grade is (100+33.3)/2=66.7%.

This grade shows how well we could recognize which instruments participate in the mixture, for example in order to know if my instrument lists are worth to be given to Chunghsin's program, so it could use their instrument models to refine the estimation.

- "Vote All" Grade

This complete simplification of the "Vote" Grade, simply gives 100% to mixes where all participating instruments got full majority and 0% to all the rest, i.e. it shows in how many mixes we guessed correctly all the participating instruments.

Results

Reading the results-table below:

- #1 column:

-- 2 – 5 - number of solos mixed together (“solo polyphony” or “number of voices”)
-- 0.25 or 0.5 - minimal length of each note classified, in seconds
-- H or S - Harmonic resynthesis or Source Reduction results

- #2 column:

-- Number - Average number of samples classified per mix
-- (Percentage) – the percentage of the samples in the mixtures, on average, which could not be detected in the solos and thus this samples were removed and not labeled. This percentage is shown only once in each cell, for the 0.25-second case only, as in 0.5-second classifications the samples which were shorter than 0.5-second were removed from the 0.25-second collection after it was already labeled.

- #3 column:

How many solo-mixtures were classified / how many different solos participated in these mixtures

- #4 column – Average Instrument Grade
- #5 column – Average Pieces Grade
- #6 column – Average Either Grade
- #7 column – Average Vote Grade
- #8 column – Average Vote All Grade

In columns 4 – 8 the numbers in the brackets indicate the K which was used for KNN in each test type. This K produced the highest results for this kind of test from a range of 1-80.
It is reasonable to use different K values depending on which type of grade we are most interested in. However, in case where we do want to use the same K value for all the different numbers of voices (2 - 5), the difference between using different K’s for each voice number or a constant one are small. For example for pieces of 0.25-second and up, using the HR method and a constant K=33, we get average instrument grades of: 65.94, 55.35, 51.88, 41.48 for 2 – 5 voices respectively, while with the best K’s we get 66.0, 56.6, 53.5, 42.6.

- The last double-row of the table shows average results for each technique over the different numbers of voices.

#v	#sam/rem	#mix	instr%	pieces%	either%	vote%	voteAll%

2
025HS
	29.6
	(9.1%)
		34/68	**66.0(44)**
			65.5(45)
				66.1(33)
				67.6(45)
					76.5(33)
					79.0(46)
						87.5(20)
						82.1(4)
							73.5(21)
							67.6(3)

0.5HS	16.1		**67.8(44)**
			66.8(39)
				66.2(44)
				68.7(46)
					76.5(75)
					80.3(46)
						84.1(44)
						88.2(3)
							64.7(21)
							76.5(3)

3
025HS
	37.7
	(7.6%)
		20/60	**56.6(23)**
			52.2(6)
				58.0(23)
				55.5(9)
					83.4(28)
					85.7(67)
						89.2(14)
						82.5(43)
							70.0(14)
							45.0(43)

0.5HS	**19.3**		**59.1(22)**
			55.2(6)
				59.0(23)
				54.4(6)
					84.1(21)
					87.0(46)
						86.0(20)
						86.4(2)
							50.0(1)
							55.0(2)

4
025HS
	41
	(5.4%)
		14/56	**53.5(17)**
			50.0(4)
				58.0(8)
				55.2(4)
					92.3(2)
					95.0(2)
						92.9(2)
						96.4(2)
							71.4(3)
							85.7(6)

0.5HS	20.1		**52.0(1)**
			53.2(4)
				59.2(8)
				57.3(22)
					90.8(9)
					94.0(2)
						94.0(9)
						95.8(2)
							78.6(9)
							78.6(2)

5
025HS
	41.1
	(2.1%)
		10/50	**42.6(43)**
			38.1(39)
				44.0(27)
				41.5(3)
					100.0(1)
					100.0(1)
						100.0(1)
						100.0(1)
							100.0(1)
							100.0(1)

0.5HS	17.1		**33.5(20)**
			42.3(57)
				35.4(2)
				41.0(8)
					91.1(9)
					99.4(1)
						91.2(2)
						98.3(1)
							60.0(2)
							90.0(1)

AVG 025HS	37.3		**54.7** 51.4	**56.5** 54.9	**88.0** 89.9	**92.4** 90.2	**78.7** 74.6
0.5HS	18.1		**53.1** 54.4	**54.9** 55.3	**85.6** 90.2	**88.8** 92.2	**63.3** 75.0

Results Table

Looking at the Average double-row (bottom one) we see that Source Reduction method produces better average grades using notes of at least 0.5-second long than when using a minimal length of 0.25, while Harmonic Resynthesis produces higher recognition rates using notes of 0.25-second and up. There is an average of 37.3 samples per piece when the minimal length is 0.25-seconds and only 18.1 (hardly half) when the minimal length of a note is raised to 0.5-second. This means that with Harmonic Resynthesis we get higher grades when using double recognition resolution.

Comparing the best average grades between Harmonic Resynthesis and Source Reduction (0.5-second minimum for SR and 0.25-second minimum for HR), we can see that the grades are very similar, with HR leading very slightly. Therefore, with the current configuration using HR is better because of the double recognition resolution which was already mentioned.

Conclusions

We have seen that both methods are relevant and produce much higher recognition rates than random instrument labeling.

In the general case, the preferred method depends on the precision qualities of the *f*0-estimation program. It was shown in our Ismir-2006 paper and the Solo-recognition grades for resynthesized solo notes that when using the HR method, the instrument recognition rate depends almost completely on a good estimation of the frequencies and amplitudes of the partials of the different notes. If the *f*0-estimation is very good, then the HR is a better method as it does not require filtering, is comparatively simple, straight-forward and can lead to very high recognition rates.

However, if the *f*0-estimation program is less precise, SR may perform better as it allows the frequency estimation to be less exact (as long as the estimated partials fall inside the correct spectral peaks) and partial amplitude information is not required at all.

Harmonic Resynthesis vs. Source Reduction (part 2 of 2) – Solo Mixtures

This is version 2 of report 18. I have added 5 new solos to the solo database and so the grades were changed, but only slightly.
Now the recognition rates of the resynthesized solos are:

Instrument	Bassoon	Clarinet	Flute	Cello	Violin	**Average**
Average rate	89.83%	81.68%	86.81%	78.34%	75.22%	82.39%

My Ismir 2006 paper shown (among other things) that for separate tones, using solely the harmonic series is enough for achieving high instrument recognition rates. As Chunghsin's f0-estimation program estimates the harmonic serieses in polyphonic music, it is natural now to try performing instrument recognition in multi-instrumental music using solely the harmonic serieses.
In the previous report (report 17) I have presented results for resynthesized solos and real duos. In this report I shall use the resynthesized solo database in order to classify mixes of solos from 2 to 5 concurrent voices.

Reminder - My different approaches to instrument recognition in multi-instrumental music

"Source Reduction (SR)"

-- Use Chunghsin's multiple-f0 program to find the different notes in a musical piece
-- For each note which was detected over a minimal number of continuous frames (vibrato/glissando are allowed):
--- Filter out everything except the STFT peaks which contain its partials
--- Classify this "cleansed" note using a solo database as the training set.

"Harmonic Resynthesis (HR)"

-- Use Chunghsin's multiple-f0 program to find the different notes in a musical piece
-- For each detected note (vibrato/glissando are allowed) over a minimal length:
--- Resynthesize[1] it using its estimated harmonic series (frequencies and amplitudes)
--- classify it using a learning set consisting from notes resynthesized from harmonic serieses of solos.

I have used LDA+KNN for the classification.

See previous report for more details.

[1] Resynthesis is not compulsory; theoretically, harmonic series could be used directly for descriptor computation. However, as the descriptor routines are written by 3'rd party and require a waveform as input, the notes were resynthesized "back" into waveforms from the harmonic serieses without loss of generality (WLOG).

Solo Mixtures

It is very difficult to obtain recordings of real music along with its precise transcription, which is required in order to perform exact instrument recognition evaluation. On the other hand, music resynthesized from MIDI files, while producing exact performances of the symbolic notation, is somewhat too easy to perform instrument recognition on, as the sounds are very similar to each other, lacking articulations and synthetically clean. Therefore, evaluation with resynthesized MIDI files will not indicate well the ability of the recognition program to handle real music.

Artificially mixed real solo performances are a compromise; on the negative side, the solo sounds do not influence each other and do not create real, common room echo. Unfortunately also, musical composition rules could not be assumed to apply to the mixtures as the mixed solos have no relevancy to each other. On the other hand, all the instrument articulations are present, the resulting music is polyphonic, and the files could be quite precisely labeled.

The instrument labeling was performed in the following way:

- Chunghsin's F0-detection program was ran on the original solos and on the solo mixtures.
- Notes detected in the solo mixtures, which lasted for at least a specific length of time (e.g. 0.25 of a second or more), were searched in the f0-detection data of the solos which were mixed in that mixture, in the appropriate temporal locations.
- When a note from the mix was found in a solo, it was labeled by the same instrument as that solo.
- Notes found only in the mixture but not in the solos were discarded.

This labeling method seems to work rather well. I have listened to different notes taken from the solo-mixes and they were always labeled correctly by this process. The percentage of discarded, unfound notes, was small and will be indicated in the results-table below.

Independent Evaluation

Before classifying each solo-mixture, the solos which were mixed into it are removed from the learning database (somewhat lowering the recognition rate).

Grading

Several different grade types were calculated in order to show different strengths of the recognition approaches:

- Instrument Grade
-- Each note is checked whether it is classified correctly as its instrument label.
-- An average grade for each participating instrument is computed for each solo-mixture.
-- An average grade per instrument is reported over all the mixtures.

The Instrument Grade shows how well each instrument is classified, and confusion matrices could be computed.

- Pieces Grade
-- Each note is checked whether it is classified correctly as its instrument label.
-- The average recognition rate per note is computed for each mixture.
-- The average recognition rate per note is reported over all the mixtures.

The Pieces Grade differs from the Instrument Grade because the number of notes performed by various instruments in a mixture is completely different.

- “Either” Grade
This is a version of the Pieces Grade, where a note is considered to be classified correctly if it is classified as either one of the instruments mixed in the mixture.

The Either Grade attempts to produce a grading which diminishes the influence of recognition errors among participating instruments, such as produced by bad filtering or incorrect partial-amplitude estimation. When I listened to different resynthesized and filtered notes it indeed happen sometimes that I heard both instruments playing concurrently in a classified sample.

- “Vote” Grade
This is a complex grade which basic idea is (without going into the gory details):

-- Produce a sorted list of the instruments in a mixture by the number of samples each of the instruments “got” from the classification.
-- If correct instruments got the highest places in the list, each gets a point. If some participating instruments got however the same number of “votes” as some incorrect instruments, a point is divided among all of the instruments with the same number of “votes”.

For example, if we had a clarinet and bassoon mixture and the number of samples different instruments received by the classification were:

Bassoon:	8 samples
Clarinet:	4
Flute:	4
Violin:	4
Cello:	0

Then bassoon which has the majority got 100% (a “point”), and “takes” one of the 2 places reserved for correct instruments (this is a duo, so 2 instruments are playing).
Clarinet however has 4 samples, like the incorrect flute and violin, and thus it gets “number of remaining places”/”number of instruments with same number of votes”=1/3=33.3%. Average grade is (100+33.3)/2=66.7%.

This grade shows how well we could recognize which instruments participate in the mixture, for example in order to know if my instrument lists are worth to be given to Chunghsin’s program, so it could use their instrument models to refine the estimation.

- "Vote All" Grade

This utter simplification of the "Vote" Grade, simply gives 100% to mixes where all participating instruments got full majority and 0% to all the rest, i.e. it shows in how many mixes we guessed correctly all the participating instruments.

Results

Reading the results-table below:

- #1 column:

-- 2 – 5 - number of solos mixed together ("solo polyphony" or "number of voices")
-- 0.25 or 0.5 - minimal length of each note classified, in seconds. Note that sometimes it happened that there were no notes of a certain instrument over 0.5 second long (due to *f0*-estimation errors) with the result that this instrument actually "disappeared". This is the reason why some of the 0.5-second grades seem bizarre. For example the HR grade of 5 voice mixes is only 60% instead of the normally expected 100%. Yet this is fair – if only notes of 4 instruments were left, classifying a sample as the 5'th instrument is plainly wrong.
-- HR or SR - Harmonic Resynthesis or Source Reduction results

- #2 column:

-- Number - Average number of samples classified per mix
-- (Percentage) – the percentage of the samples in the mixtures, on average, which could not be detected in the solos and thus this samples were removed and not labeled. This percentage is shown only once in each cell, for the 0.25-second case only, as in 0.5-second classifications the samples which were shorter than 0.5-second were removed from the 0.25-second collection after it was already labeled.

- #3 column:

How many solo-mixtures were classified / how many different solos participated in these mixtures

- #4 column – Average Instrument Grade
- #5 column – Average Pieces Grade
- #6 column – Average Either Grade
- #7 column – Average Vote Grade
- #8 column – Average Vote All Grade

In columns 4 – 8 the numbers in the brackets indicate the K which was used for KNN in each test type. This K produced the highest results for this kind of test from a range of 1-80.
It is reasonable to use different K values depending on which type of grade we are most interested in. However, in case where we do want to use the same K value for all the different numbers of voices (2 - 5), the difference in grade between using "best" K's for each voice number or just a constant K are small, around 2%.

- The last double-row of the table shows average results for each technique over the different numbers of voices.

#v	#sam/rem	#mix	instr%	pieces%	either%	vote%	voteAll

2
025SR
HR
	29.6
	(9.1%)
		34/68	66.4(15)
			66.3(26)
				68.0(15)
				66.3(30)
					79.5(74)
					77.4(12)
						82.8(6)
						84.6(6)
							67.6(6)
							67.6(6)

0.5SR
HR
	16.1		67.6(68)
			68.9(53)
				69.2(18)
				66.7(26)
					80.8(74)
					78.3(53)
						89.0(3)
						83.6(21)
							76.5(3)
							64.7(19)

3
025SR
HR
	37.7
	(7.6%)
		20/60	53.3(13)
			57.3(7)
				56.8(13)
				59.0(20)
					84.7(17)
					86.2(7)
						82.5(12)
						89.2(4)
							40.2(14)
							65.0(4)

0.5SR
HR
	19.3		54.6(3)
			60.5(11)
				54.7(2)
				60.2(11)
					86.0(17)
					85.4(43)
						84.3(6)
						86.2(34)
							50.0(6)
							60.0(3)

4
025SR
HR
	41
	(5.4%)
		14/56	49.6(9)
			56.9(44)
				55.4(9)
				58.9(4)
					94.8(70)
					92.8(2)
						98.2(4)
						91.7(6)
							92.9(4)
							64.3(3)

0.5SR
HR
	20.1		51.1(5)
			58.2(8)
				57.0(9)
				60.1(68)
					94.0(68)
					91.6(12)
						95.8(17)
						94.0(12)
							85.7(17)
							78.6(12)

5
025SR
HR
	41.1
	(2.1%)
		10/50	38.8(62)
			43.3(27)
				41.7(64)
				45.5(27)
					100.0(1)
					100.0(1)
						100.0(1)
						100.0(1)
							100.0(1)
							100.0(1)

0.5SR
HR
	17.1		43.0(62)
			40.1(24)
				41.9(62)
				38.0(24)
					99.4(1)
					86.3(1)
						98.3(1)
						85.8(5)
							90.0(1)
							60.0(4)

AVG 025SR HR	37.3		52.0 55.9	55.5 57.4	89.7 89.1	90.9 91.4	75.2 74.2
0.5SR HR	18.1		54.1 56.9	55.7 56.2	90.0 85.4	91.8 87.4	75.5 65.8

Results Table

Looking at the Average double-row (bottom one), we can see that HR produces slightly higher Instrument Grades and Pieces Grades than SR, while SR leads somewhat with the Either and Vote-All Grades.
Both methods produce better average grades using notes of at least 0.5-second long than with 0.25-second. There is an average of 37.3 samples per piece (2'nd column last row) when the minimal length is 0.25-seconds and only 18.1 (hardly half) when the minimal length of a note is raised to 0.5-second. This means that although the 0.5-second grade is slightly higher, we get much better recognition resolution with 0.25-second pieces and therefore it should be preferred.
The highest Vote-All grade for 2-voice mixes is only 76.5 (the Vote grade is 89%), which means that it is not really advisable to blindly rely on these classifiers (in their current states) in order to determine the participating instruments in a mix, e.g. for selecting instrument modules for an *f0*-estimation program.

Conclusions

We have seen that both methods, Harmonic Resynthesis and Source Reduction, are relevant and produce much higher recognition rates than random instrument labeling.
In a way, with the harmonic resynthesis method, the problem of multi-instrumental, polyphonic instrument recognition was reduced to correct multiple *f0-estimation*; this is because in HR, there is no fundamental difference between instrument recognition with resynthesized notes from solos and resynthesized notes from multi-instrumental music.

In the general case, the preferred recognition method depends on the precision qualities of the *f*0-estimation program. It was shown in our Ismir-2006 paper that when using the HR method, the instrument recognition rate depends almost completely on a good estimation of the frequencies and amplitudes of the partials of the different notes. If the *f*0-estimation is very good, then the HR is a better method as it does not require filtering, is comparatively simple, straight-forward and can lead to very high recognition rates.
However, if the *f*0-estimation program is less precise, SR may be preferred as it allows the frequency estimation to be less exact as long as the estimated partials fall inside the correct spectral peaks.

Improving the Consistency of a Sound Database

In instrument recognition, like other machine learning fields, the learning (classifying) database might contain samples which could disturb the classification process:

- **Attribute Noise**
  Badly sampled sounds or garbled data

- **Class Noise**
  Samples mislabeled as belonging to the wrong class

- **Distance Outliers**
  Samples correctly recorded and labeled but still mislead the classification process by differing too much from other samples in their class

The presence of such samples can lead to inflated error rates and substantial distortions of parameter and statistic estimates when using either parametric or nonparametric tests (Zimmerman 1998).

For a thorough historical summary on Outliers see (Jason and Overbay 2004).

In this report I shall compare 3 different algorithms for removing outliers.

Algorithms for Removing Outliers

Interquantile Range (IQR) (Draper 1999)

IQR is a commonly used outlier removal approach:

For every descriptor, let **P1** be some value bigger than X% of the values of this descriptor, and let **P2** be a value bigger than Y% of the values, X>Y. For example - X=99, Y=1.

Remove samples where the descriptor has values that are larger than

P1+(P1-P2)*C

or smaller than

P2-(P1-P2)*C

C is some scalar (e.g. 1).

Instead of using percentages, a common modification of IQR[1] is to calculate the mean and standard deviation (STD) of every descriptor, and then remove the samples where that descriptor has absolute (abs) values which are several times bigger than the STD.

[1] See for example the subsection “Removing Outliers” in section “Data Analysis” in the Matlab R2006b documentation.

Notice that IQR is not a supervised method, meaning that it does not utilize the classification information about the learning database and thus is appropriate for usage with non-labeled databases.

Modified IQR (MIQR)

This is a proposed supervised variant of Interquantile Range.

- Modification 1: Perform IQR on each class separately instead of all the database samples together.
- Modification 2: When a sample with an outlier descriptor is found, do not remove it automatically, but rather count for every sample the number of descriptors which produce outliers. At the end of the process remove the samples which have more outlier descriptors than a specified threshold.

Self-Classification Outlier removal (SCO)

This proposed outlier removal method is a kind of a "wrapper method" in the sense that it utilizes the specific classification algorithm itself for its purpose.
Like in Self-Classification evaluation (see section "Evaluation Techniques" in the thesis), the learning set consists of a certain percentage of the samples from each class (66% in this document) which are selected randomly, while the test set is made of the rest (reminder - the learning set is used to classify the test set). This process repeats an N number of times (50 times in this document). After each classification round the indices of the misclassified samples are recorded.
At the end of the process, samples which were misclassified more than a certain number of times are removed.
Note that this method differs significantly from my older LOO outlier removal method (Livshin, Peeters and Rodet 2003) which has classified each sample in the database using Leave-One-Out and removed the misclassified ones. SCO uses partial, randomly selected groups of samples (66%/34%) at each classification step, creating a kind of "bagging" effect and thus lowering the distortion in classifications caused by outliers in the learning group (François et al. 2003), which the older LOO method suffers from.

Contaminated Database

Our excerpt of the Studio OnLine database has 16 different instruments with a total of 1323 samples. As this database was professionally recorded it is very self consistent, as is evident from its average self-classification grade - 95.7% for 50 self-classification experiments of 66%/34% split (see section "Evaluation techniques" in the thesis). 162 Feature Descriptors were computed on each sample.
In order to compare the affectivity of the 3 different outlier removal methods, I have "contaminated" the SOL database with 4 kinds of outlying samples:

- **"Class Noise"**: The class labels of random 5% of the database samples was changed to a different, randomly selected, class.

- **“Random Samples”**: samples with descriptors selected randomly from the range of [0 256] for each descriptor were added to the database with random classes. The quantity of these samples is 5% of the original database size.
- **“Random Bound Samples”**: the MINimum and MAXimum of each descriptor in the original database were found. 5% of random samples were added to the database, each random descriptor in these samples is bound by its respective MIN and MAX.
- **“Random Samples Class Bound”**: 5% of random samples were added to each class, with descriptors bound by their respective MIN and MAX values in that class.

Experiment

Each of the outlier removing algorithms was performed on the contaminated database. As there is a tradeoff between the number of good and bad samples removed by the algorithms, each algorithm was evaluated twice; first allowing up to 1% of “good“ samples to be removed (Table 1) and second time with up to 10% good samples removed (Table 2).

Results

Reading Tables 1 and 2:
All the results are in percentages.
The columns

- “Clean“: this column shows the average self classification result of classifying only the good samples in the contaminated database (the contaminated database with all bad samples removed). Note that this “Clean” database is 5% smaller than the original SOL database because of the removal of the distorted Class Noise samples.
- “Contaminated”: the average result with the contaminated database.
- IQR, MIQR, SCO – the classification results of the contaminated database after it was purged with each of these algorithms.

The rows

- “Grade” – the average grade of 50 66%/34% self-classification rounds. Numbers in the brackets are the 95% confidence intervals.
- “Class noise”, “Random256, “Random Bound”, “Random Bound Class” – the percentage of each type of bad samples removed by the algorithm. For example, in Table 1, MIQR has removed 53% of the Class Noise.

- “Bad Removed” – the total percentage of bad samples removed.
- “Good Removed” - the total percentage of good samples removed.

	Clean	**Contaminated**	**IQR**	**MIQR**	**SCO**
Grade	92.7 **(92.0– 93.4)**	79 **(78.6 – 79.5)**	88.1 **(87.7 – 88.4)**	91.6 **(91.3 – 91.8)**	86.2 **(85.8 – 86.6)**
Class Noise	NA	0	0	53	51.5
Random256	NA	0	100	100	39
Random Bound	NA	0	81.8	100	43.9
Random Bound Class	NA	0	12.5	31.8	7.6
Bad Removed	NA	0	49.6	70.1	35.5
Good Removed	NA	0	0.9	0.9	0.95

Table 1. Outlier removal results with up to 1% of good samples removed

	Clean	**Contaminated**	**IQR**	**MIQR**	**SCO**
Grade	92.7 **(92.0– 93.4)**	79.2 **(78.6 – 79.5)**	89.8 **(89.5 – 90.2)**	92.3 **(91.5 – 93.1)**	96.8 **(96.4 – 97.1)**
Class Noise	NA	0	18.2	75.7	100
Random256	NA	0	100	100	100
Random Bound	NA	0	100	100	98.5
Random Bound Class	NA	0	50	86.4	51.6
Bad Removed	NA	0	67.2	90.4	87.8
Good Removed	NA	0	9.9	8.8	9.5

Table 2. Outlier removal results with up to 10% of good samples removed

Looking at the types of bad samples removed by each algorithm we can see:

IQR - As could be expected from its non-supervised nature, IQR has dealt badly with Class Noise, being unable to detect it. Random256 and Random Bound outliers were removed well as the probability of getting at least a single descriptor out of 162 with an "edge" value is high with these contamination types, and a single outlying descriptor is enough for IQR to remove a sample. Samples from the Random Bound Class are much more difficult for IQR to detect – many descriptors in many classes do not have edge values compared to the Min/Max values of these descriptors over the entire database. For example in class X, the maximum and minimum values of descriptor Y could be 10 and -10, while the maximum and minimum values of descriptor Y over the entire database are 50 and -50. And so, Random Bound Class samples from class X will never have an outlying descriptor Y. As IQR does not use class information, it cannot detect such descriptors even if they do have a "local" edge value in their class.

MIQR - We can see that the MIQR method has outperformed the other two, removing higher percentages of bad samples. As it uses class information, it did not have the disadvantages of IQR regarding Class Noise and Random Bound Class samples. Another reason for its higher "data-to-noise" ratios is that it did not remove every sample with a single outlying descriptor, but rather removed samples which had at least N outlying descriptors. Naturally, in systems where a single sensor may go wrong and produce sometimes random values, N could be simply set to 1.

SCO – As the SCO algorithm does not try guessing whether samples should be removed by examining their values, its behaviour is the same with all types of outliers as long as they are misclassified. However, as the Random Bound Class samples had the highest probability of being actually classified as their appointed class (while possibly having outlying values which could be detected by MIQR) SCO had the least success removing them. Random Bound Class samples were the toughest samples to handle for all the outlier removing algorithms.
In Table 2 we see that SCO has produced the purged database with the highest mean self-classification grade – 96.8%, which is even noticably higher than the grade the Non-Contaminated database produced – 92.7%. This high self-classification grade was achieved while removing only 87.8% of the contaminated samples for 9.5% of good samples (worse than MIRQ). This is actually not surprising – allowing the SCO algorithm to remove as much as 10% of the good samples, we let it tailor the remaining database for itself, SCO being a wrapper method. However, this does not mean that SCO outperformed the other algorithms in this case. Our goal was not to get the highest self-classification grade but rather to get rid of the most "bad" samples for a certain percentage of good samples removed. See the "Conclusions" for some more discussion on this topic.

Conclusions

For non-labeled data, out of our 3 algorithms, IQR is the "only way to go" as the other two algorithms require class information. For getting rid of contaminated data in labeled databases, MIQR seems to be the best. If highest classification results are required, specifically tailored wrapper-type methods may well be the answer, such as SCO.
Note that not all outliers should be always removed – there are many arguments about the desirability of the whole business of removing outliers, as diversity in a database is not necessarily bad and may actually model a special, interesting population rather than indicate sampling errors. The general rule is to "know your data" and being able to "intelligently guess" which percentage of erroneous samples could be expected thus providing the outlier removing algorithms with appropriate limiting parameters, such as the percentage of samples to remove, the number of descriptors which are likely to go wrong, or even tailor special outlier removing algorithms for specific data types such as the one in (Adam, Rivlin and Shimshoni 2001) for removing outliers from different views (graphical images) of the same scenery.

References

Adam, A., Rivlin, E., Shimshoni, I., 2001. “ROR: Rejection of Outliers by Rotations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol 23, No 1, January 2001.

Draper, N. 1999. "Statistics 201 lectures". *Online statistics lectures by the Statistics Department of the Wisconsin Madison University*. URL: www.stat.wisc.edu/~jyan/st201/pdf/dis10.pdf

François, J., Grandvalet, Y., Denoeux, T., Roger, J. M., 2003. “Resample and combine: an approach to improving uncertainty representation in evidential pattern classification,” *Information Fusion* 4(2): 75-85 (2003).

Livshin, A., Peeters, G., Rodet, X. 2003. "Studies and Improvements in Automatic Classification of Musical Sound Samples," In Proceedings of the International Computer Music Conference (ICMC’03).

Osborne, J. W., Overbay A., 2004. “The power of outliers (and why researchers should always check for them)”. *Practical Assessment, Research & Evaluation*, 9(6), 2004.

Zimmerman, D. W., 1998. “Invalidation of parametric and nonparametric statistical tests by concurrent violation of two assumptions”. *Journal of Experimental Education,* 67(1), 55-68.

Slight improvement of polyphonic multi-instrumental recognition rate using MIQR

In report #19 I have introduced the MIQR database purging method (see end of this document for reminder) which outperformed other two outlier-removal methods, SCO and IQR.
In report #18_v2 I presented results of instrument recognition performed on mixes of solos of 5 instruments, with 2 to 5 instruments playing concurrently.

In this report, after performing instrument recognition on the same solo mixes as in report 18_v2, I used MIQR in order to find outliers in the test group's <u>obtained</u> classifications and remove them.
This was performed with all solo-mixture experiments from report 18_v2 (2-5 polyphony, Harmonic Resynthesis and Source Reduction, 0.25 and 0.5 minimal note lengths) and has resulted in a rather consistent slight increase of around 1% in recognition rates while removing around 10% of the test samples.

<u>**Detailed Results:**</u>
Under each original grade in the following table (written in black), a grade written in red presents the results after applying MIQR. Below that grade, also in red, is the percent of test samples removed by MIQR. For example in column 4 of the first row, we can see that the average Source-Reduction recognition rate for solo mixes of 2 solos with a minimal note length of 0.25-second was originally 66.4% (with k=15 in KNN), but after using MIQR which removed 11.2% of the test samples, the grade changed to **68.3% (using k=26 for KNN)**.

<u>As the table is rather complex, I attach report 18_v2 to this email, which explains thoroughly the grade types and all the information in the table.</u>

#v	#sam/rem	#mix	instr%	pieces%	either%	vote%	voteAll

<u>**2**</u>
025SR
	29.6
	(9.1%)
		34/68	**66.4(15)**
			68.3(26)
			11.2%
				68.0(15)
				70.7(45)
				11.2%
					79.5(74)
					80.7(70)
					11.7%
						82.8(6)
						87.2(4)
						11.5%
							67.6(6)
							76.5(4)
							11.5%

HR			66.3(26)
			67.0(26)
			10.7%
				66.3(30)
				67.2(12)
				10.5%
					77.4(12)
					77.9(12)
					10.3%
						84.6(6)
						86.8(2)
						10.8%
							67.6(6)
							73.4(2)
							10.8%

0.5SR	**16.1**		**67.6(68)**
			68.9(70)
			12.8%
				69.2(18)
				71.3(46)
				12.3%
					80.8(74)
					81.2(70)
					12.8%
						89.0(3)
						86.8(5)
						12.3%
							76.5(3)
							73.5(5)
							12.3%

HR			68.9(53)
			70.2(61)
			14.3%
				66.7(26)
				68.1(61)
				14.0%
					78.3(53)
					78.7(69)
					14.3%
						83.6(21)
						82.3(43)
						14.1%
							64.7(19)
							64.7(2)
							14.1%

<u>**3**</u>
025SR
	37.7
	(7.6%)
		20/60	**53.3(13)**
			53.1(9)
			8.8%
				56.8(13)
				57.9(9)
				8.8%
					84.7(17)
					86.9(67)
					9.1%
						82.5(12)
						83.3(3)
						8.7%
							40.2(14)
							45.0(3)
							8.7%

HR			57.3(7)
			58.2(8)
				59.0(20)
				61.5(8)
					86.2(7)
					87.2(4)
						89.2(4)
						89.2(3)
							65.0(4)
							60.0(3)

	19.3		10.8%	10.2%	10.8%	11.0%	11.0%

0.5SR
HR
			54.6(3)
			55.8(6)
			9.5%
			60.5(11)
			62.7(7)
			12.6%
				54.7(2)
				56.6(6)
				10.9%
				60.2(11)
				62.6(11)
				12.6%
					86.0(17)
					88.1(46)
					9.5%
					85.4(43)
					84.4(5)
					12.3%
						84.3(6)
						89.2(2)
						10.0%
						86.2(34)
						85.6(25)
						12.1%
							50.0(6)
							60.0(2)
							10.0%
							60.0(3)
							60.0(4)
							13.4%

4
025SR
HR
	41
	(5.4%)
		14/56	49.6(9)
			50.1(4)
			9.6%
			56.9(44)
			57.7(44)
			10.2%
				55.4(9)
				55.8(17)
				9.6%
				58.9(4)
				59.9(44)
				10.4%
					94.8(70)
					95.6(4)
					8.9%
					92.8(2)
					93.6(12)
					10.4%
						98.2(4)
						97.3(2)
						9.1%
						91.7(6)
						92.0(2)
						10.0%
							92.9(4)
							85.7(2)
							9.1%
							64.3(3)
							64.3(1)
							11.0%

0.5SR
HR
	20.1		51.1(5)
			53.4(4)
			7.8%
			58.2(8)
			58.0(8)
			11.2%
				57.0(9)
				57.5(22)
				8.1%
				60.1(68)
				60.2(68)
				11.6%
					94.0(68)
					94.1(2)
					7.2%
					91.6(12)
					92.0(12)
					11.2%
						95.8(17)
						95.8(2)
						7.7%
						94.0(12)
						95.8(12)
						11.2%
							85.7(17)
							78.6(2)
							7.7%
							78.6(12)
							85.7(12)
							11.2%

5
025SR
HR
	41.1
	(2.1%)
		10/50	38.8(62)
			37.8(39)
			10.8%
			43.3(27)
			44.4(49)
			8.6%
				41.7(64)
				42.4(3)
				10.8%
				45.5(27)
				46.7(22)
				8.6%
					100.0(1)
					100.0(1)
					10.8%
					100.0(1)
					99.7(2)
					10.0%
						100.0(1)
						100.0(1)
						10.8%
						100.0(1)
						100.0(2)
						10.0%
							100.0(1)
							100.0(1)
							10.8%
							100.0(1)
							100.0(2)
							10.0%

0.5SR
HR
	17.1		43.0(62)
			42.3(57)
			7.6%
			40.1(24)
			43.6(5)
			12.2%
				41.9(62)
				42.3(57)
				7.6%
				38.0(24)
				39.7(14)
				12.2%
					99.4(1)
					98.7(1)
					8.0%
					86.3(1)
					84.6(2)
					13.4%
						98.3(1)
						98.3(4)
						8.0%
						85.8(5)
						83.7(2)
						11.2%
							90.0(1)
							90.0(4)
							8.0%
							60.0(4)
							40.0(2)
							11.9%

AVG 025SR HR	37.3		52.0 52.3 10.1% 55.9 56.8 10.1%	55.5 56.7 10.1% 57.4 58.8 9.9%	89.7 90.8 10.1% 89.1 89.6 10.4%	90.9 91.9 10.0% 91.4 92.0 10.4%	75.2 76.8 10.0% 74.2 74.4 10.7%
0.5SR HR	18.1		54.1 55.1 9.4% 56.9 58.6 12.6%	55.7 56.9 9.7% 56.2 57.6 12.6%	90.0 90.5 9.4% 85.4 85.4 12.8%	91.8 92.5 9.5% 87.4 86.8 12.1%	75.5 75.5 9.5% 65.8 62.6 12.6%

Results Table

We can see in the two last rows that for 0.25-second pieces all the average grades were improved when MIQR was applied. The average grades using 0.5-second pieces were all improved for the Source-Reduction method, while in Harmonic Resynthesis the "vote" and "vote-all" grades were

a little decreased. As our main interests are the Instrument and Pieces grades it seems that on the whole MIQR did more good than evil. While admittedly 1% is not much, it could still be claimed that this improvement is not a quirk of the classification algorithms but a “real” improvement, as this small increase was evident in almost all experiments.
It is a pity that the results did not rise higher, but it could have been expected. What MIQR basically does here, is take the test set and its obtained classification and check whether it is consistent with the learning group. But it should be, as the learning group was used for making these classifications! This process is somewhat similar to using two classification algorithms to classify a group of samples and then discard the samples which are classified differently by the algorithms.

<u>Reminder - MIQR:</u>

Interquantile Range (IQR)

IQR is a commonly used outlier removal approach:

For every descriptor, let **P1** be some value bigger than X% of the values of this descriptor, and let **P2** be a value bigger than Y% of the values, X>Y. For example - X=99, Y=1.

Remove samples where the descriptor has values that are larger than

P1+(P1-P2)*C

or smaller than

P2-(P1-P2)*C

C is some scalar (e.g. 1).

Instead of using percentages, a common modification of IQR is to calculate the mean and standard deviation (STD) of every descriptor, and then remove the samples where that descriptor has absolute (abs) values which are several times bigger than the STD.

Notice that IQR is not a supervised method, meaning that it does not utilize the classification information about the learning database and thus is appropriate for usage with non-labeled databases.

Modified IQR (MIQR)

This is a proposed supervised variant of Interquantile Range.

- Modification 1: Perform IQR on each class separately instead of all the database samples together.
- Modification 2: When a sample with an outlier descriptor is found, do not remove it automatically, but rather count for every sample the number of descriptors which produce outliers. At the end of the process remove the samples which have more outlier descriptors than a specified threshold.