Feature Selection with the CLOP Package

Isabelle Guyon
ETH Zürich, Switzerland
isabelle@clopinet.com

Jiwen Li
University of Zürich, Switzerland
li@ifi.unizh.ch

Theodor Mader, Patrick A. Pletscher,
Georg Schneider, Markus Uhr
ETH Zürich, Switzerland
{*tmader,patrickp,scgeorg,uhrm*}*@student.ethz.ch*

March, 2006

Abstract

We used the datasets of the NIPS 2003 challenge on feature selection as part of the practical work of an undergraduate course on feature extraction. The students were provided with a toolkit implemented in Matlab. Part of the course requirements was that they should outperform given baseline methods. The results were beyond expectations: the student matched or exceeded the performance of the best challenge entries and achieved very effective feature selection with simple methods. We make available to the community the results of this experiment and the corresponding teaching material: `http://clopinet.com/isabelle/Projects/ETH/Feature_Selection_w_CLOP.html`.

1 Introduction

In the recent years, it has been recognized by the machine learning and neural network communities that competitions are key to stimulate research and bring improvement. Large conferences are now regularly organizing competitions (e.g. KDD, CAMDA, ICDAR, TREC, ICPR, CASP, and IJCNN). In 2003, we organized a competition on the theme of feature selection, the results of which were presented at a workshop on feature extraction [12]. The outcomes of that effort were compiled in a book including tutorial chapters and papers from the proceedings of that workshop [11]. The website of the challenge remains open for post-challenge submissions: `www.nipsfsc.ecs.soton.ac.uk`. Meanwhile, we have been organizing a second challenge on the theme of model selection: `www.modelselect.inf.ethz.ch`. As part of that effort, we have developed a Matlab toolkit based on the Spider package [19]. All this material constitute a great teaching resource that we have exploited in a course on feature extraction: `clopinet.com/isabelle/Projects/ETH`. We are reporting on our teaching experience with two intentions:

- encouraging other teachers to use challenge platforms in their curricula, and
- providing to graduate students simple competitive baseline methods to attack problems in machine learning.

The particular theme of the class is feature extraction, which we define as the combination of feature construction and feature selection. These past few years, feature extraction and space dimensionality reduction problems have drawn a lot of interest. More than a passing fancy, this trend in the research community is driven by applications: bioinformatics, chemistry (drug design, cheminformatics), text processing, pattern recognition, speech processing, and machine vision provide machine learning problems in very high dimensional spaces, but often with comparably few examples. A lot of attention was given in class to feature selection because many successful applications of machine learning have been built upon a large number of very low level features (e.g. the "bag-of-word" representation in text processing, gene expression coefficients is cancer diagnosis, and QSAR features in cheminformatics). Our teaching strategy is to make students gain hands on experience by working on large real world datasets (those of the NIPS 2003 challenge [8]), rather than providing them with toy problems.

We reviewed in class basic machine learning techniques (linear predictors, neural networks, kernel methods, and decision trees), all of which are included in the provided software package. The course devoted to feature construction included various transforms such as the Fourier transform, and convolutional methods. We also reviewed a number of preprocessing methods involving noise modeling. The students could experiment with such methods using the Gisette dataset, a handwriting recognition task based on the MNIST data [16]. The rest of the curriculum covered feature selection methods (filters, wrappers, and embedded methods), information theoretic and ensemble methods. The datasets were introduced progressively in the homework to illustrate algorithms learned in class. The class requisites included making one complete challenge submission with results on the five datasets of the challenge. The students had then to present their methods and results in a poster presentation. Another requisite was to make a slide presentation of one of the chapters of the book reporting results of the challenge.

In the remainder of the paper, we provide some details on the data and software used. We review and analyze the results obtained by the students and draw lessons of this experience to encourage others to use challenges as teaching resources.

2 Datasets and synopsis of the challenge

The NIPS 2003 challenge included five datasets (Table 1) from various application domains. All datasets are two-class classification problems. The data were split into three subsets: a training set, a validation set, and a test set. All three subsets were made available at the beginning of the challenge. The class labels for the validation set and the test set were withheld.

The identity of the datasets and of the features (some of which were random features artificially generated) were kept secret during the challenge but have been revealed since then. The datasets were chosen to span a variety of domains and difficulties (the input variables are continuous or binary, sparse or dense; one dataset has unbalanced classes.) One dataset (Madelon) was artificially constructed to illustrate a particular difficulty: selecting a feature set when no feature is informative by itself. We chose datasets that had sufficiently many examples to create a large enough test set to obtain statistically significant results [8]. To facilitate the assessment of feature selection methods, we introduced a number of artificial features called **probes** drawn at random from a distribution resembling that of the real features, but carrying no information about the class labels. A good feature selection algorithm should eliminate most of the probes. The details of data preparation can be found in a technical memorandum [8].

Table 1: **NIPS 2003 challenge datasets.** For each dataset we show the domain it was taken from, its type T (d=dense, s=sparse, or sb=sparse binary), the number of features **#F**, the percentage of probes **%P**, the number of examples in the training, validation, and test sets, and the fraction of examples in the positive class **%[+]**, and the ratio number of training examples to number of features **Tr/F**. All problems are two-class classification problems.

Dataset	**Domain**	**T**	**#F**	**%P**	**#Tr**	**#Va**	**#Te**	**%[+]**	**Tr/F**
Arcene	Mass Spectrometry	d	10^4	30	100	100	700	44	0.01
Dexter	Text classification	s	2.10^4	50	300	300	2000	50	0.015
Dorothea	Drug discovery	sb	10^5	50	800	350	800	10	0.008
Gisette	Digit recognition	d	5000	50	6000	1000	6500	50	1.2
Madelon	Artificial	d	500	96	2000	600	1800	50	4

The challenge participants could submit prediction results on the validation set and get their performance results and ranking on-line during a development period. The validation set labels were then revealed and the participants could make submissions of test set predictions, after having trained on both the training and the validation set. For details on the benchmark design, see [12]. For the class, the students had access to the training and validation set labels, but not the test labels. Thus far, the test set labels have not been released to the public and we intend to leave it this way to keep an on-going benchmark. The students made post-challenge submissions to the web site of the challenge (`www.nipsfsc.ecs.soton.ac.uk`) to obtain their performance on the test set.

The distributions of the original results of the challenge participants for the various datsets are represented in Figure 1.[1] The distribution is rather peaked for Gisette, indicating that this is perhaps the easiest task. Conversely, the results are very spread out for Dorothea, probably the hardest task of all. Madelon has a bimodal distribution, symptomatic of a particular difficulty, which was not overcome by all the participants. We will provide explanations in Section 6. These distributions guided us to set standards for the students accomplishments:

- We introduced in class the datasets in order of presumed complexity.
- We provided the students with baseline methods approximately in the tenth percentile of the best methods.
- We asked students to try to outperform the baseline method and gave them extra credit for outperforming the best challenge entry.

In the remainder of the paper, we give details on the software toolbox provided and the students results.

3 Learning object package

The machine learning package we used for the class called CLOP (Challenge Learning Object Package) is available from the website of the "performance prediction challenge" `http://clopinet.com/isabelle/Projects/modelselect/Clop.zip`. We have written an QuickStart guide[2] and FAQs are available.[3] We present

[1]We show the performance on the validation set because we have many more validation set entries than final test set entries. The results correlate well with those on the test set.

[2]http://clopinet.com/isabelle/Projects/modelselect/QuickStart.pdf

[3]`http://clopinet.com/isabelle/Projects/modelselect/MFAQ.html`

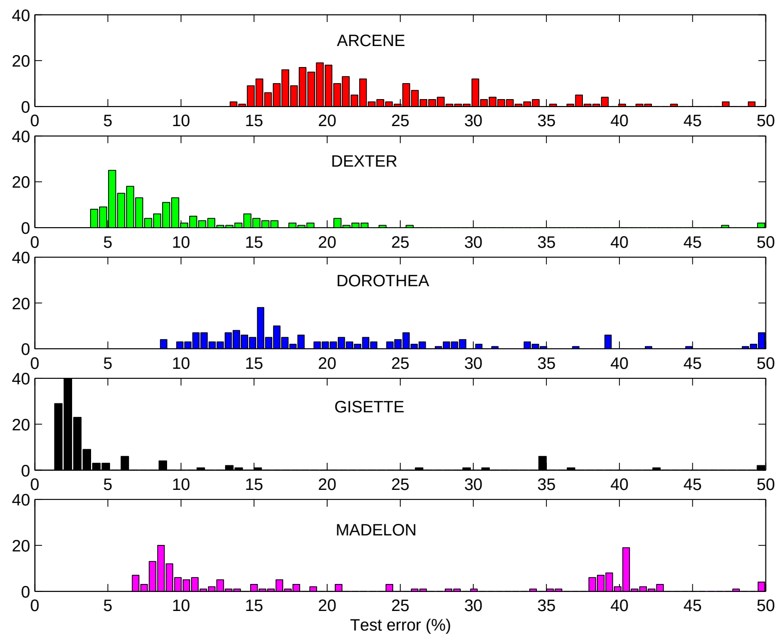

Figure 1: **Distribution of the challenge participant results.** We show histograms of the balanced error rate (BER) for the five tasks.

below only a high level overview of the package.

Data and algorithm objects

The methods were implemented using the interface of the Spider package developed at the Max Planck Institute for Biological Cybernetics [19]. The Spider package on top of which our package is built, uses Matlab (R) objects (The MathWorks, http://www.mathworks.com/). Two simple abstractions are used:

- **data**: Data objects include two members X and Y, X being the input matrix (patterns in lines and features in columns), Y being the target matrix (i.e. one column of +-1 for binary classification problems).
- **algorithms**: Algorithm objects representing learning machines (e.g. neural networks, kernel methods, decision trees) or preprocessors (for feature construction, data normalization or feature selection). They are constructed from a set of hyper-parameters and have at least two methods: train and test. The train method adjusts the parameters of the model. The test method processes data using a trained model.

For example, you can construct a data object `D`:

```
> D = data(X, Y);
```

The resulting object has 2 members: `D.X` and `D.Y`. Models are derived from the class algorithm. They are constructed using a set of hyperparameters provided as a cell array of strings, for instance:

```
> hyperparam = {'h1=val1', 'h2=val2'};
> model0 = algorithm(hyperparam);
```

In this way, hyperparameters can be provided in any order or omitted. Omitted hyperparameters take default values.

To find out about the default values and allowed hyperparameter range, one can use the "default" method:

```
> default(algorithm)
```

The constructed model `model0` can then be trained on data `Dtrain` and tested on data `Dtest`:

```
> [Dout, model1] = train(model0, Dtrain);
> Dout = test(model1, Dtest);
```

The trained model `model1` is an object identical to `model0`, except that its parameters (some data members) have been updated by training. Matlab uses the convention that the object of a method is passed as first argument as a means to identify which overloaded method to call. Hence, the "correct" `train` method for the class of `model0` will be called. Since Matlab passes all arguments by value, `model0` remains unchanged. By calling the trained and untrained model with the same name, the new model can overwrite the old one. Repeatedly calling the method "train" on the same model may have different effects depending on the model.

The output data object `Dout` stores the result of calling the test method on the input data in `Dout.X`. If the algorithm is a preprocessing, `Dout.X` will be the preprocessed data matrix. If the algorithm is a classifier, `Dout.X` will be a vector of discriminant values. The other member `Dout.Y` remains unchanged.

To save the model is very simple since Matlab objects know how to save themselves:

```
> save('filename', 'modelname');
```

This feature is very convenient to make results reproducible, particularly in the context of a challenge.

Compound models: chains and ensembles

The Spider (with some CLOP extensions) provides ways of building more complex "compound" models from the basic algorithms with two abstractions:

- **chain**: A chain is a learning object (having a train and test method) constructed from an array of learning objects. Each array member takes the output of the previous member and feeds its outputs to the next member.
- **ensemble**: An ensemble is a also learning object constructed from an array of learning objects. The trained learning machine performs a weighted sum of the predictions of the array members. The individual learning machines are all trained from the same input data. The voting weights are set to one by default. An interface is provided for user-defined methods of learning the voting weights.

Compound models behave like any other learning object: they can be trained and tested. In the following example, a chain object `cm` consists of a feature standardization for preprocessing followed by a neural network:

```
> cm=chain({standardize, neural});
```

While a chain is a "serial" structure of models, an ensemble is a "parallel" structure. The following command create an ensemble model `em`:

```
> em=ensemble({neural, kridge, naive});
```

To create more complex compound models, models of the same class with different hyperparameters or different models can be combined in this way; chains can be part of ensembles or ensembles can be part of chains.

4 CLOP objects provided for the class

It is easy to be overwhelmed when starting to use a machine learning package. CLOP is an extremely simplified package, limited to a few key methods, each having just a few hyper-parameter values with good default values. This makes it suitable for teaching a machine learning class. More advanced students can venture to use other methods provided in the Spider package, on top of which CLOP is built.

The CLOP modules correspond to methods having performed well in the feature selection challenge [12]. There are five classifiers:

- **kridge** Kernel ridge regression (our own implementation [9]).
- **naive** Naive Bayes classifier (our own implementation [10]).
- **neural** Neural network (using Netlab [1]).
- **rf** Random Forest (using the original implementation [3]).
- **svc** Support vector classifier (using LibSVM [4]).

We limited the number of hyperparameters of each method to avoid making the model selection problem too complex. For instance, the two kernel methods "kridge" and "svc" use a single kernel: $k(\mathbf{x}, \mathbf{x}') = (coef_0 + \mathbf{x} \cdot \mathbf{x}')^d exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2)$. In addition to the three hyperparameters $coef_0$, d, and γ, the kernel methods have a "shrinkage" parameter for regularization, corresponding to a small value added to the diagonal of the kernel matrix.

The CLOP modules also include five preprocessors:

- `standardize`: Standardization of the features (the columns of the data matrix are divided by their standard deviation; optionally, the mean is first subtracted if center=1.)
- `normalize`: Normalization of the lines of the data matrix (optionally the mean of the lines is subtracted first.)
- `shift_n_scale`: Performs $\mathbf{x} \leftarrow (\mathbf{x} - \mathbf{offset})/\mathbf{scale}$ globally on the data matrix. Optionally performs in addition $\log(1 + \mathbf{x})$.
- `pc_extract`: Extraction of features with principal component analysis.
- `subsample`: Takes a subsample of the training patterns. May be used to downsize the training set or exclude outliers.

We also provide five feature selection methods:

- **s2n**: Signal-to-noise ratio coefficient for feature ranking [6].
- **relief**: Relief ranking criterion [15].

- **gs**: Forward feature selection with Gram-Schmidt orthogonalization [17].
- **rffs**: Random Forest used as feature selection filter (see [11], Chapter 7).
- **svcrfe**: Recursive Feature Elimination filter using svc [13].

All the feature selection methods chosen are either individual feature ranking methods or nested subset methods (forward selection or backward elimination). Hence, they all provide a ranking of features as part of the trained "model". The number of features to be selected is a hyperparameter, which may be varied after training to avoid recomputing the ranking.

For the feature extraction class, we provided the students with other examples of learning objects that are not part of CLOP, but can be found in the homework instructions (`http://clopinet.com/isabelle/Projects/ETH/#homework1`). Some of them ended up being used in the best student entries:

- **convolve**: Data preprocessing for images, using two-dimensional convolution.
- **gauss_ker** and **exp_ker**: Two examples of convolutional kernels.
- **probe**: Object that can be wrapped around a feature selection filter to compute pvalues with the "probe" method (see [11], Chapter 2).
- **TP**: "True Positive" feature ranking filter, using as ranking index the fraction of feature values that are positive when the target value is positive. It is suitable for unbalanced data with few example of the positive class and a sparse binary data matrix (like the DOROTHEA dataset).
- **bias**: Post-processing adjusting the bias value to optimize the BER.

The Spider provides several objects for cross-validation and other model selection methods. A model selection object is derived from the class "algorithm" and possesses train and test methods. For instance, 10-fold cross-validation is performed as follows:

```
> cv_model=cv(my_model, {'folds=10'});
> cv_output = train(cv_model, Dtrain);
```

We created an archive containing the CLOP software, the datasets, and the baseline methods, provided to the students, see `http://clopinet.com/isabelle/Projects/ETH/Feature_Selection_w_CLOP.html`. Sample code is provided including examples of combinations of modules into chains and ensembles, and a script to load data, train a model, and save the model and the results into an archive ready to upload to the challenge web site (`http://www.nipsfsc.ecs.soton.ac.uk/submit/`).

5 Performance assessment and class requisites

Performance is assessed using several metrics:

- BER: The balanced error rate, that is the average of the error rate of the positive class and the error rate of the negative class. This metric was used because some datasets (particularly DOROTHEA) are unbalanced.
- AUC: Area under the ROC curve. The ROC curve is obtained by varying a threshold on the discriminant values (outputs) of the classifier. The curve represents the fraction of true positive as a function of the fraction of false negative. For classifiers with binary outputs, BER=1-AUC.
- Ffeat: Fraction of features selected.
- Fprobe: Fraction of probes found in the feature set selected.

As part of the class requirements, the student had to make one full submission to the challenge web site, including results on the training, validation, and test set, and the feature subset used. They were also required to submit their CLOP model. For each dataset, the students would each earn one point if they obtained a better BER than the baseline method, or a smaller feature set for the same BER. They would earn two points if they matched the BER of the best challenge entry (within the statistical error bar.) Additionally, they earned points for a poster presentation of their results.

To build the baseline methods, we were inspired by the methods of the top ranking challenge participants and experiments we conducted after the challenge (see [11], Chapter 9). The top ranking challenge entries all used a combination of:

- a feature selection filter, and
- a multivariate regularized classifier.

The filters chosen ranged from using the simple Pearson correlation coefficient to using elaborate methods such as Random Forests. The best classifiers were Bayesian neural networks, kernel methods (regularized least square or SVM), Random Forests and naïve Bayes.

We chose the simplest possible baseline methods, which attain performances approximately in the best tenth percentile of the challenge entries. We made sure that outperforming those baseline methods was easy. In Table 2, we provide the Matlab code of the baseline methods. In Table 3, we provide the results of both the baseline methods and the best challenge entries.[4]

6 Student work

During the curriculum, the datasets were introduced one at a time to illustrate topics addressed in class, based on progressive difficulty. We briefly describe the homework assignments associated with each dataset.

[4]Two separate rankings were done before and after the release of the validation set labels. The Best challenge performance are the best of all entries, with or without the knowledge of the validation set labels. The performances of the baseline method are obtained by training with both training and validation data, except for DOROTHEA, where it was found that including the validation set data as part of training degrades performance.

Table 2: **Baseline methods provided to the students.** For each dataset we provided a simple method attaining performances approximately in the best tenth percentile of challenge entries.

Dataset	Code of the baseline method
ARCENE	`my_svc=svc({'coef0=1','degree=3','gamma=0','shrinkage=0.1'});` `my_model=chain({standardize,s2n('f_max=1100'),normalize,my_svc})`
DEXTER	`my_svc=svc({'coef0=1','degree=1','gamma=0','shrinkage=0.5'});` `my_model=chain({s2n('f_max=300'),normalize,my_svc})`
DOROTHEA	`my_model=chain({TP('f_max=1000'),naive,bias})`
GISETTE	`my_svc=svc({'coef0=1','degree=3','gamma=0','shrinkage=1'});` `my_model=chain({normalize, s2n('f_max=1000'),my_svc})`
GISETTE (pixels)	`my_svc=svc({'coef0=1','degree=4','gamma=0','shrinkage=0.1'});` `my_model=chain({convolve(exp_ker({'dim1=9', 'dim2=9'})), ...` `normalize, my_svc})`
MADELON	`my_svc=svc({'coef0=1','degree=0','gamma=1','shrinkage=1'});` `my_model=chain({probe(relief,{'p_num=2000','pval_max=0'}), ...` `standardize,my_svc})`

Table 3: **Performances comparison.** The table shows the balanced error rate of the baseline method BER_0, the corresponding number of features n_0, the fraction of features used by the baseline method, the performance of the best challenge entry BER^* with its error bar δBER, and the best performance achieved by the students BER with their post-challenge entries. The students earned one point for $BER < BER_0$ or $\{BER = BER_0$ and $n < n_0\}$. They earned 2 points for $BER < BestBER + \delta BER$. We also show the training time of the best student models on a 1.5GHz Pentium.

Dataset	Baseline BER_0	Feat# n_0	% feat.	Challenge best $BER^* \pm \delta_{BER}$	Student BER	Training time (s)
ARCENE	14.70	1100	11	11.90 ± 1.20	10.48	3.8
DEXTER	5.00	300	1.5	3.30 ± 0.40	3.25	1.2
DOROTHEA	12.37	1000	1	8.54 ± 0.99	9.3	0.7
GISETTE	1.80	1000	20	1.26 ± 0.14	1.11	11.2
GISETTE (pixels)	1.06	784	NA	NA	0.78	127.9
MADELON	7.33	20	4	6.22 ± 0.57	6.22	7.8

GISETTE: **Feature construction.**

The students worked first on the GISETTE dataset for several weeks. GISETTE is a handwriting recognition task, providing easy-to-visualize patterns. This allows students to experiment with feature construction and preprocessing. In the first three homework assigments (`http://clopinet.com/isabelle/Projects/ETH/#homework1`), the students were asked to write their first "learning object", play with data visualization tools, and make their first entry on the challenge web site. We made several suggestions of simple-to-implement preprocessing objects, such as raising the input variables to a given power or binarizing the data. No knowledge of the identity of the features of the GISETTE data was provided at this stage. The next two homework (`http://clopinet.com/isabelle/Projects/ETH/#homework4` focused on trying feature construction objects. The identity of the features of the GISETTE data was revealed (for the challenge, the features included original pixels, products of pixels, and meaningless "probe" features). A program to visualize the digit images was provided as well as a number of examples of feature construction learning objects, implementing principal component analysis (PCA), k-means clustering, convolutional methods, and filter bank methods, including Fourier and Hadamard transforms. The most successful method turned out to be the simple "blurring" with a smoothing kernel (Figure 2). This result is consistent with experiments on digit recognition formerly reported [7].

Dexter: univariate filters.

We chose Dexter, the test classification dataset, for the next series of assignments (`http://clopinet.com/isabelle/Projects/ETH/#homework7`). For Dexter, univariate feature selection filters combined with a simple linear SVM classifier perform well. It is therefore a terrain of choice to learn about univariate filter methods and how to optimize the number of features with statistical tests or cross-validation. To that end, we taught in class a number of univariate statistical tests, including the Ttest and the Ftest. We implemented example learning objects, which rank the features according to a given test statistic and compute the pvalue and False Discovery Rate (FDR).[5] The pvalues of well know test statistics such as the Student T statistic[6] are either known analytically or tabulated. Since ranking indices are not limited to well studied test statistics, permutation tests or the "probe" method are used to empirically estimate pvalues and FDRs (see [11], Chapter 2). According to the probe method, random meaningless features bearing no predictive power are added to the original features and ranked with them. One way to generate such probes is to permute randomly columns of the original matrix. For a given value of the ranking index r_0 , the pvalue is estimated as the fraction of probes having ranking indices r larger than r_0. The FDR is estimated as the ratio of the pvalue and the fraction of real features with $r \geq r_0$. We provided the students with the "probe" learning object, an object wrapping around any feature ranking filter and calculating pvalues and FDRs using the "probe" method. The number of features can be determined by setting a threshold on the pvalue or the FDR. When using cross-validation, the student had also to devise strategies to optimize both the number of features and the hyper-parameters.

Madelon: Multivariate feature selection.

The Madelon dataset was artificially generated. It can be thought of as a mega-XOR problem. Clusters of data points were put at the vertices of a 5 dimensional hypercube and randomly labeled. In this way, no projection in 1, 2, 3, or 4 dimensions shows a reasonable class separation, making the problem intrinsically multivariate. This difficulty may explain the bimodal distribution of the BER of the challenge participants (Figure 1). Working on Madelon was an opportunity for students to experiment with multivariate filters such as Relief, or try wrappers or embedded methods with non-linear multivariate classifiers, such as SVMs (see `http://clopinet.com/isabelle/Projects/ETH/#homework10`).

[5]For a given value of a test statistic t_0 used for feature ranking (the larger, the more relevant), the pvalue or "false positive rate" is the fraction of irrelevant features whose test statistic t exceeds t_0. The False Discovery Rate (FDR) is the ratio of the number of irrelevant features with $t \geq t_0$ over the total number of features with $t \geq t_0$.

[6]The T statistic used for feature selection is the ratio of the difference of the means of the 2 classes to the class standard deviation.

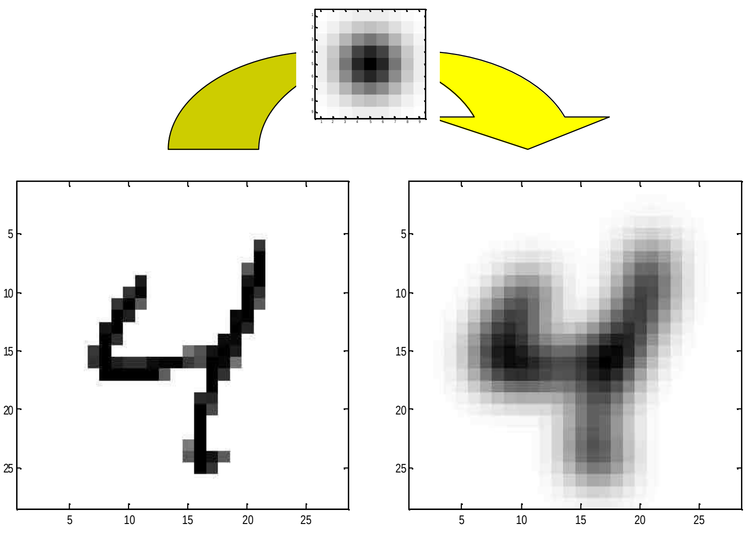

Figure 2: **Example of digit from the** GISETTE **dataset.** The original digit (left) is a 28x28 gray level pixel image. It was transformed to the right image by convolving with a smoothing kernel (top).

ARCENE: **Heterogeneous data.**

The ARCENE dataset was the object of study for the next two homework assignments (`http://clopinet.com/isabelle/Projects/ETH/#homework12`). It presents two difficulties: a small training set and heterogeneous data (coming from 3 sources). Therefore, it is a good dataset to further study multivariate feature selection algorithms, and experiment with mixtures of experts or ensemble methods.

DOROTHEA: **Unbalanced data.**

The DOROTHEA dataset was the object of the last homework assignments (`http://clopinet.com/isabelle/Projects/ETH/#homework14`). We think it is the hardest dataset, based on the huge spread in the distribution of results of the participants (Figure 1): it is easy to badly overfit on that problem because there are 100,000 features but only 800 training examples. It is also a very unbalanced dataset, with less than ten percent of the examples belonging to the positive class. The other datasets were either balanced or almost balanced. The students learned that SVMs can overfit and that Naïve Bayes is sometimes a safe choice. The students also learned to post-fit the bias to compensate for class imbalance.

The best student entries are shown in Table 4. The training set and validation set of the challenge were merged into a single training set. The hyper-parameters were adjusted by cross-validation (usually 5-fold cross-validation). The final model is trained on all the available labeled data (training plus validation set).[7] As indicated in Table 4, the training times are modest (of the order of 1-10 seconds per model, except for GISETTE when the pixel representation is used, because of the time consuming preprocessing). Most students matched the performances of the

[7]For DOROTHEA, where it was found that including the validation set data as part of training degrades performance, only training data were used.

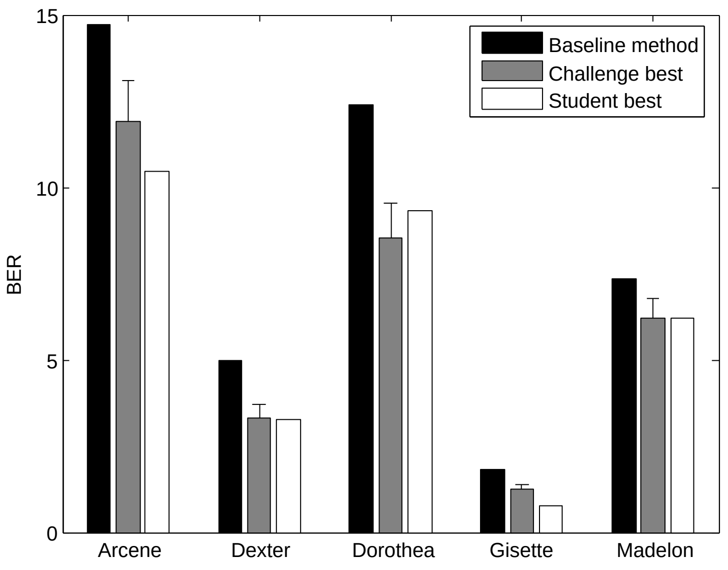

Figure 3: **Result comparison.** The student performances matched (within the statistical error bar) or exceeded the performances of the best challengers.

best challenge entries (within the statistical error bar) and therefore earned the maximum number of points (Table 3 and Figure 3). Some student entries even exceeded the performance of the best challenge entries. We think that the achievements of the students are remarkable. They exceeded our expectations. Of course, it can be argued that they had several advantages over the competitors, and in particular access to test results on-line. However, this does not diminish significantly their achievement since many other post-challenge entries were made since the end of the challenge and hardly any matched or outperformed the best results of the challengers.

It is noteworthy that, for their final entries, the student methods do not depart very much from the baseline method. However, this does not mean that they did not explore other methods. Indeed, at the beginning, they were rather daring and even implemented algorithms they had heard about in other classes. Unfortunately, given the time that they could devote to that class, it is not surprising that they could not easily outperform with new ideas the baseline methods, which are the result of learning from the designs of the best challenge entrants and extensive cross-validation experiments (see [11], Chapter 9). The grading system prompted them to get the best possible performance, but also rewarded them for their creativity by encouraging them to report negative results in their poster presentation.

The order of introduction of the datasets was based on their difficulty estimated by the spread of the BER distribution of the challenge entrants (Figure 1). This turned out to be a good choice. However, one drawback of using GISETTE for the first homework assignment is that it is a large dataset. Some students ran into computational problems or memory limitation problems on their laptops, which generated some frustration.

General observations can be made about the best models. All of them use simple classifiers linear in their parameters (naïve Bayes [10] and SVMs [2]). The naïve

Table 4: **Methods employed by the best student entries.**

Dataset	Code of the methods employed
ARCENE	`my_svc=svc({'coef0=2','degree=3','gamma=0','shrinkage=0.1'});` `my_model=chain({relief('f_max=1400'),normalize,my_svc})`
DEXTER	my_svc=svc({'coef0=1','degree=1','gamma=0','shrinkage=0.5'}); `my_model=chain({s2n('f_max=4500'),normalize,my_svc})`
DOROTHEA	`my_model=chain({TP(f_max=15000),normalize, ...` `relief(f_max=700'),naive,bias});`
GISETTE	`my_svc=svc({'coef0=1','degree=3','gamma=0','shrinkage=1'});` `my_model=chain({s2n('f_max=1000'), normalize, my_svc})`
GISETTE (pixels)	`my_svc=svc({'coef0=1','degree=4','gamma=0','shrinkage=0.1'});` `my_model=chain({convolve(gauss_ker({'dim1=5', 'dim2=5'})), ...` `normalize,my_svc})`
MADELON	`my_svc=svc({'coef0=1','degree=0','gamma=0.3','shrinkage=0.3'});` `my_model=chain({relief('f_max=20'),standardize,my_svc})`

Bayes and linear SVM are also linear in their input components: the decision function is of the form $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, where $\mathbf{x}$ is the feature vector to be classified, $\mathbf{w}$ is the weight vector, and b the bias. The non-linear SVM has a decision function of the form: $f(\mathbf{x}) = \sum_k \alpha_k k(\mathbf{x}_k, \mathbf{x}) + b$, where $k(.,.)$ is a similarity measure called "kernel" function and the α_k are weights. The sum runs over a subset of the training examples called "support vectors". The learning algorithms differ in the way the weights $\mathbf{w}$, α_k, and b are computed. Both the naïve Bayes and linear SVM are known to be robust against overfitting: they get good predictive performance (generalization) even when the dimensionality of input space (number of features) is large compared to the number of training examples. But, there are gradings in the robustness: naïve Bayes>linear SVM>non-linear SVM. Many theoretical arguments have been made to explain this, principally based on the computation of performance bounds [18]. There is an informal way of understanding this in terms of "effective dimensionality": The naïve Bayes classifier is a "univariate" linear classifier in the sense that the overall decision is made by a simple voting, using the individual predictive power of the feature as voting weight. Thus its effective dimensionality is very low. The linear SVM is a "multivariate" linear classifier, thus its effective dimensionality will be generally higher than that of the naïve Bayes. The non-linear SVM is formally equivalent to a linear SVM in a large dimensional feature space of ϕ vectors: $f(\mathbf{x}) = \mathbf{w} \cdot \phi + b$, where $\mathbf{w} = \sum_k \alpha_k \phi_k$ and $k(\mathbf{x}_k, \mathbf{x}) = \phi_k \cdot \phi$. Hence its effective dimensionality tends to be larger than that of the linear SVM.

If we use the ratio of number of training examples to number of features to assess the danger of overfitting (Table 1), we can see that MADELON and GISETTE are in the overfitting green zone (ratios larger than one), DOROTHEA is in the red zone (ratio much smaller than one), and DEXTER and ARCENE are in the middle range. Using this as a rule of thumb to choose the classifier, the non-linear SVM may be applied to GISETTE and MADELON without fear, the more conservative linear SVM to ARCENE and DEXTER, and the very conservative naïve Bayes to DOROTHEA. This rule-of-thumb predicts rather well the best student entries, with the exception of ARCENE: here, we would have expected the linear SVM to perform best, but the non-linear SVM has a slight performance advantage, perhaps due to the heterogeneous nature of the ARCENE data. This demonstrates the robustness of the non-linear SVM against overfitting, particularly its "soft-margin" version [5], which provides additional regularization, via the so-called "shrinkage" parameter in our implementation.

Another observation is that feature selection is performed with only three simple filters:

- `TP`: A filter useful for highly unbalanced classes like DOROTHEA to pre-select "true positive" features being at least sometimes "present" in representatives of the positive class [20].
- `s2n`: A simple univariate filter, which ranks features according to the "signal-to-noise" (s2n) ratio [6], that is, for a given variable/feature, the separation of the means of the two classes, relatively to the standard deviation.
- `relief`: A filter which ranks features according to their separating power "in the context of other features". The Relief algorithm [14] uses an approach based on the nearest-neighbor algorithm. For each example, the closest example of the same class (nearest hit) and the closest example of a different class (nearest miss) are selected. The score of the i^{th} variable/feature is computed as the average over all examples of the magnitude of the difference between the distance to the nearest hit and the distance to the nearest miss, in projection on the i^{th} variable.

The simplest question we could ask about feature selection is: "Can we get better performance by reducing the feature set?" The answer to this question is not easy to give because good classifiers, which are robust against overfitting, perform well without feature selection. In the analysis of the results of the challenge (see [11], Chapter 9), we could not answer this question satisfactorily because the winning entry used 100% of the features. In this study, our entries outperform or match the best challenge entries, within the statistical error bar, and they use less than 100% of the features. Hence, feature selection can indeed yield performance improvements. But we prefer rephrasing the question as follows: **"Can we effectively reduce the feature set while improving or not significantly degrading performance?"**. Three indicators are used to give a quantitative answer to this question:

- **The fraction of features used F_{feat}.**
- **The false discovery rate (FDR).** The FDR is the fraction of irrelevant features in the selected feature set. It is estimated by the ratio of the fraction of "probes" in the selected features and the fraction of probes in the original feature set: $FDR \simeq F_{probe}/F^*_{probe}$. A large FDR is indicative of Type I errors (false positive, i.e. irrelevant features wrongly selected).
- **The Z_{BER} statistic.** A statistically significant increase in balanced error rate BER, compared to BER^* obtained using all the features, is indicative of Type II errors (false negative, i.e. relevant features wrongly discarded). We define $Z_{BER} = (BER - BER^*)/\sigma_\Delta$, where σ_Δ is the standard deviation of $BER - BER^*$.[8] For small training sets, the benefit of reducing the dimensionality may outweigh the loss of information by discarding useful features, so the BER may actually be better than BER^*, making Z_{BER} an imperfect indicator of Type II errors.

[8] Paired test statistics would be more powerful, but since in this analysis the differences are quite significant, we adopt the simple Z statistic. σ_Δ is computed as $\sqrt{\sigma^{*2} + \sigma^2}$ where for n test examples, $\sigma^{*2} = BER^*(1 - BER^*)/n$, and $\sigma^2 = BER(1 - BER)/n$. For unbalanced classes, we replace the number of samples n by twice the number of samples of the smallest class.

Table 5: **Effectiveness of feature selection.** The table shows the number n of "balanced" samples, the BER of the full model, the BER of the model built with the reduced feature set, the standard deviation of the difference, the corresponding Z statistic, the fraction of features used, the fraction of probes of the full and the reduced model, and the false discovery rate $FDR \simeq F_{probe}/F^*_{probe}$.
For all datasets, the reduced model is the best student model and the full model is built with the same hyperparameters. for DEXTER, we show both the student model (s) and the baseline model (b).

Dataset	n	BER^*	BER	σ_Δ	$-Z_{BER}$	F_{feat}	F^*_{probe}	F_{probe}	FDR
ARCENE	700	0.1186	0.1048	0.0127	1.09	0.14	0.30	0.04	0.14
DEXTER (s)	2000	0.0410	0.0325	0.0049	1.75	0.23	0.50	0.55	1.1
DEXTER (b)	2000	0.0410	0.0395	0.0049	0.31	0.02	0.50	0.08	0.15
DOROTHEA	160	0.3094	0.0930	0.0372	5.82	0.01	0.50	0.03	0.05
GISETTE	6500	0.0180	0.0111	0.0020	3.49	0.20	0.50	0.00	0
MADELON	1800	0.4872	0.0622	0.0120	35.53	0.04	0.96	0.00	0

For simplicity, we use the 2σ rule: if $Z_{BER} > 2$, we consider BER to be significantly worse than BER^*. Conversely, $Z_{BER} < -2$ indicates that BER is significantly better than BER^*.[9] If we find a classifier for which $Z_{BER} < -2$ and for which F_{feat} is significantly lower than one, we will give a positive answer to the first question: Yes, we can get better performance by reducing the feature set. But, if the FDR remains large, the feature selection is ineffective, even though the performance has improved.

To evaluate the impact of selection, we reran the best models without feature selection, keeping the same hyper-parameters. We summarize the results in Table 5. For the three last datasets, feature selection is very effective: the feature selection yields significantly better results at the 2% risk level ($-Z_{BER} > 2$)[10] and the FDR is zero or near zero. Let us take a closer look at some of the results:

ARCENE:

For ARCENE, without performance degradation, the feature set can be reduced to 14% of its initial size, with a 14% false discovery rate. Therefore, feature selection is quite effective despite the fact that no performance improvement is gained.

DEXTER:

For DEXTER, the student entry selected many features, in an effort to match the performance of the best challenge entry. Consequently, the FDR suffers. But, the baseline method, with a performance only moderately worse, obtains a good FDR of 15%. Therefore, here again feature selection is quite effective if we are willing to sacrifice a little in performance.

DOROTHEA:

For DOROTHEA, compared to the full model, our reduced model has very significantly better performances. The fraction of features used is very small (1%), and so is the FDR (5%). The best student entry did not quite match the performance of the best challenge entry (the BayesNN-large of Radford Neal, BER=0.0854) using

[9] This corresponds for a one-tailed ztest to a risk of 2% of being wrong.
[10] At the 4% risk level, the 4 last ones are significant.

100% of the features. Yet, sacrificing a little in performance, very significant and effective feature set reduction is achieved.

GISETTE:

The best results on the GISETTE dataset were obtained with the knowledge of the identity of the features, using smoothed images (Figure 2. However, some student entries, which did not make use of this additional knowledge, also outperformed the best challenge entry. This is the result shown in Table 5. We see that the feature selection is very effective for GISETTE: Using only 20% of the features, a significant reduction in error rate is achieved, and an undetectable false discovery rate is attained.

MADELON:

For the MADELON dataset, the data being artificial, we know by construction that only 20 features are relevant (with some redundancy). Yet one entry in the challenge (the BayesNN-DFT-combo+v of Radford Neal and Jianguo Zhang) reaches the smallest test BER achieved so far (6.22%), using 100% of the features. Both the baseline method and the best student challenge entry select all 20 relevant features. Hence the performance improvement of the student entry compared to the baseline method is due to a better hyper-parameter choice. The FSPP2(unpublished) method of Shen Kaiquan achieves the same performance with only 12 features. This illustrates the effectiveness of Relief to select the relevant features, but its ineffectiveness to remove unnecessary feature redundancy.

7 Conclusions and future work

A challenge can be more than a one-time event. It can become an on-going life benchmark and a teaching tool. By leaving the website of the NIPS2003 feature selection challenge open for post-challenge submissions (`http://www.nipsfsc.ecs.soton.ac.uk/`), we have given to graduate students and researchers the opportunity to compare their algorithms to well established baseline results. Since the end of the challenge, the number of entrants has almost doubled. During the only down time of the site (when the building burnt!) numerous requests of re-activating it were received.

In this paper, we have demonstrated that even undergraduate students can get their hands dirty and "learn machine learning from examples", with a success that exceeded our expectations. All of them easily outperformed the baseline methods we provided them and most of them matched the performances of the best challengers (within the statistical error bar) or even exceeded them. We hope that this experience will be followed by similar other attempts. In the mean time, we make available all of our teaching material, data and code.

The results obtained mark also a victory of simple methods. All the models used to match or outperform the best challenge entries use a combination of simple normalization, feature selection filters (signal-to-noise ratio, Relief, or fraction of true positive), and a naïve Bayes or a support vector machine classifier. There was no need to use ensemble methods or transduction. However, univariate feature selection and linear classifiers did not always suffice.

With this study, we could reach more conclusive results regarding the effectiveness of feature selection than by analyzing the results of the challenge. The winning challenge entry used no feature selection, *i.e.* had significantly better results than other entries using feature selection. The student entries used a reduced feature set while matching or outperforming the performance of the best challenge entry. For three datasets the reduced model using a small fraction of the original feature set significantly outperformed the full model and the false discovery rate approached zero.

This paper by no means marks an end point to the problem of feature selection or even to solving the tasks of the NIPS2003 feature selection challenge. Our explorations indicate that there is still much room for improvement. In particular, since we have released the identity of the features, it is now possible to introduce domain knowledge in the feature construction process. To a limited extent we have seen that this strategy show promises on the GISETTE datasets: a simple smoothing of the pixel image allowed us to boost performances.

Acknowledgments

The experiments were carried out as part of a lecture on feature extraction taught at ETH Zürich. We are extremely grateful to Prof. Joachim Buhmann for his support and encouragements and to Peter Orbanz for help and advices. We are very thankful to the institutions that have contributed data: the National Cancer Institute (NCI), the Eastern Virginia Medical School (EVMS), the National Institute of Standards and Technology (NIST), DuPont Pharmaceuticals Research Laboratories, Reuters Ltd., and the Carnegie Group, Inc. We also thank the people who formatted the data and made them available: Thorsten Joachims, Yann Le Cun, and the KDD Cup 2001 organizers. The Challenge Learning Object Package (CLOP) is based on code to which many people have contributed: The co-developers and beta-testers of CLOP: Amir Reza Saffari Azar and Gideon Dror. The creators of the Spider: Jason Weston, Andre Elisseeff , Gikhan BakIr , Fabian Sinz. The developers of the packages attached: Chih-Chung Chang and Chih-JenLin Jun-Cheng (LIBSVM), Chen, Kuan-Jen Peng, Chih-Yuan Yan, Chih-Huai Cheng, and Rong-En Fan (LIBSVM Matlab interface), Junshui Ma and Yi Zhao (second LIBSVM Matlab interface), Leo Breiman and Adele Cutler (Random Forests), Ting Wang (RF Matlab interface), Ian Nabney and Christopher Bishop (NETLAB), Thorsten Joachims (SVMLight), Ronan Collobert (SVM Torch II), Jez Hill, Jan Eichhorn, Rodrigo Fernandez, Holger Froehlich, Gorden Jemwa, Kiyoung Yang, Chirag Patel, Sergio Rojas. The CLOP project was partially supported by the National Science Foundation under Grant N0. ECS-0424142. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Christopher M. Bishop. *Neural networks for pattern recognition.* Oxford University Press, Oxford, UK, 1996.

[2] Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, 1992. ACM.

[3] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. `http://citeseer.ist.psu.edu/breiman01random.html`.

[4] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.

[6] T. R. Golub et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[7] I. Guyon, V. Vapnik, B. Boser, L. Bottou, and S. A. Solla. Structural risk minimization for character recognition. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4. Proceedings of the 1991 Conference*, pages 471–479, San Mateo, CA, 1992. Morgan Kaufmann.

[8] Isabelle Guyon. Design of experiments of the NIPS 2003 variable selection benchmark. Technical report, 2003. `http://www.nipsfsc.ecs.soton.ac.uk/papers/Datasets.pdf`.

[9] Isabelle Guyon. Kernel ridge regression tutorial. Technical report, 2005, `http://clopinet.com/isabelle/Projects/ETH/KernelRidge.pdf`.

[10] Isabelle Guyon. Naïve bayes algorithm in CLOP. Technical report, 2005, `http://clopinet.com/isabelle/Projects/ETH/NaiveBayesAlgorithm.pdf`.

[11] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti Zadeh, Editors. *Feature Extraction, Foundations and Applications.* Studies in Fuzziness and Soft Computing. Physica-Verlag, Springer, `http://clopinet.com/isabelle/Projects/NIPS2003/call-for-papers.html`, 2006, in press. See also on-line supplementary material: `http://clopinet.com/isabelle/Projects/NIPS2003/analysis.html`.

[12] Isabelle Guyon, Steve R. Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the NIPS 2003 feature selection challenge. In *NIPS*, 2004. `http://books.nips.cc/papers/files/nips17/NIPS2004_0194.pdf`.

[13] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[14] K. Kira and L. Rendell. A practical approach to feature selection. In D. Sleeman and P. Edwards, editors, *International Conference on Machine Learning*, pages 368–377, Aberdeen, July 1992. Morgan Kaufmann.

[15] K. Kira and L. A. Rendell. A Practical Approach to Feature Selection. In *Proceedings of the 9th International Conference on Machine Learning, ICML'92*, pages 249–256, San Francisco, CA, 1992. Morgan Kauffman.

[16] Y. LeCun and C. Cortes. The MNIST database of handwritten digits. *`http://yann.lecun.com/exdb/mnist/`*.

[17] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3:1399–1414, 2003.

[18] V. Vapnik. *Statistical Learning Theory.* John Wiley &amp; Sons, N.Y., 1998.

[19] J. Weston, A. Elisseeff, G. BakIr, and F. Sinz. The Spider machine learning toolbox. *`http://www.kyb.tuebingen.mpg.de/bs/people/spider/`*, 2005.

[20] Jason Weston, Fernando Pérez-Cruz, Olivier Bousquet, Olivier Chapelle, André Elisseeff, and Bernhard Schölkopf. Feature selection and transduction for prediction of molecular bioactivity for drug design. *Bioinformatics*, 19(6):764–771, 2003.