

Applied Machine Learning

Assignment 2

Professor: Aude Billard
Assistant: Basilio Noris
Student Assistant: Nicolas Sommer
contacts:
aude.billard@epfl.ch
basilio.noris@epfl.ch
n.sommer@epfl.ch

Winter Semester 2011

1 Goals

Assignment 2 covers 4 machine learning techniques that perform *classification* and *regression* on 2-class and multi-class data. This assignment will run through 4 practical sessions dedicated respectively to:

- Binary Classification (LDA, SVM, Boosting)
- Multi-Class Classification (SVM, GMM)
- Regression (SVR, GMR)

This assignment will be graded through an oral presentation that will take place on December 16. This presentation counts for 25% of the total grade of the course. The practicals are conducted in teams of two or exceptionally three. Unless told otherwise, we assume that the work has been shared equally by the members of the team and hence both members will be given the same grade. More information on content of the general purpose of the assignment and on the way the work should be presented are given below. The concrete instruction to follow for each practical session are provided in Section 5 and below.

2 Practical Sessions

The practical sessions revolve around the evaluation of machine learning algorithms on real data. They will guide you through the process of classification and regression on datasets of different types, size and dimensionality. Your task will be to learn how the different parameters of each method affect the results. Most of your effort should be spent in understanding the results you are obtaining. You should ask yourself: "Is this what I was expecting?"; "is this surprising?"; can you relate the mathematical formulations of these methods to what you are seeing on the screen? Take the opportunity to discuss the effects and results you obtain within your team. In case of doubt, do not hesitate to ask the assistants for help during the hour dedicated for this (11-12am each Friday).

2.1 MLDemos

The practicals will use primarily the MLDemos software. The software (downloadable at <http://mldemos.epfl.ch>) provides a graphical interface for visualizing the data and algorithms you will use throughout this year. The image below shows an example of its interface.

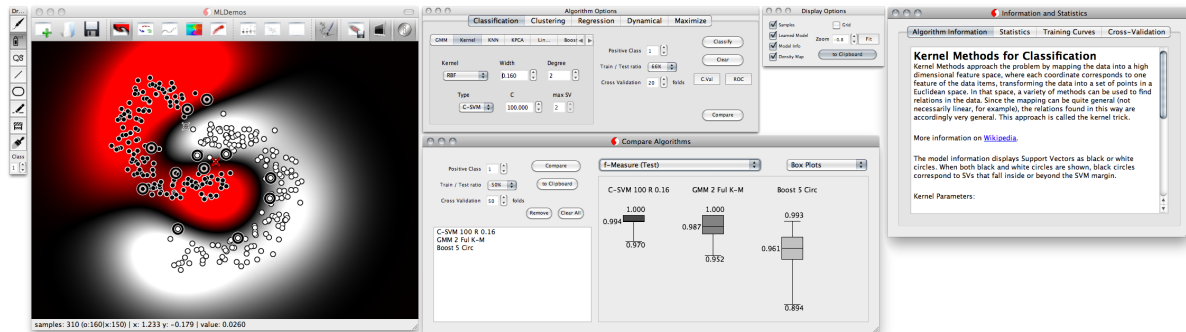


Figure 1: The MLDemos user interface, with its main canvas for drawing/visualizing data, parameters and options dialogs for changing and comparing different algorithms, and the information panel.

3 Datasets

In these practicals you will be working on the same data you used in the previous practical, namely:

- Face/Object images datasets from PCAFaces
- A Standard Benchmark datasets from the UCI Machine Learning Database¹

To ensure that each group produces original work, you should by preference use the Faces dataset or a dataset assigned by the class assistants.

4 Performance Measures

The performance measure for a binary classifier is composed of two main elements: a measure of True and False Positives. These two measures reflect respectively:

1. The samples from Class A that are correctly classified as samples from Class A
2. The samples from Class B that are incorrectly classified as samples from Class A.

The term *Positives* comes from the fact that usually a binary classifier is used to detect samples that belong to a given class (the positives) as compared to the samples that do not belong to this class (negatives).

Regardless of the nomenclature, a problem arises from having multiple values for performance estimation. This is even more important when we take into account the scope of usage of the classifier. If you are developing a vegetables-sorting machine and want to classify potentially infected vegetables, you might want to be sure to obtain 100% accuracy in terms of True Positives (you do not want to let any infected vegetable slip away unnoticed) even if this means

¹<http://archive.ics.uci.edu/ml/datasets.html>

sacrificing a certain amount of healthy vegetables (False Negatives). This is not necessarily true if you are trying to detect faces for the auto-focus mechanism of your camera. These differences are usually modelled through measures called Precision, Recall and the F-Measure.

4.1 Precision, Recall and the F-Measure

Two measures are often used to cope with the multiplicity of the classification performance results. These are:

$$recall = \frac{TP}{TP + FN}$$

Which tells us how many samples from Class 1 we can expect to correctly classify (True Positives) among all samples from Class 1 (True Positives and False Negatives). This however tells us nothing about the samples from Class 2, which is why we measure

$$precision = \frac{TP}{TP + FP}$$

Which tells us that, among a number of detections (True and False Positives), we have a certain amount of good detection (True Positives). We can then combine these two measures into a single value:

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

This is called a F-Measure² and is very often used as a good estimate of a classifier performance. In the practicals you will use the F-Measure to evaluate the accuracy of your classifiers.

4.2 Multi-Class classification performance

In multi-class problems it is more difficult to compute a measure such as the F-Measure. In these cases, the most common measure for performance is the number of misclassified samples (classification errors) over all the classes and all samples. This measure assigns the same importance to all classes, in contrast to the generalised form of the F-Measure seen above.

Data display

The software displays the response of the classifier at each position of the input space with a coloring scheme (See Figure 2). These colors correspond to

- **red**: the sample belongs to the positive class ($f(\mathbf{x}) > 0$)
- **white**: the sample belongs to the negative class ($f(\mathbf{x}) < 0$)
- **black**: the sample is close to the class boundary ($f(\mathbf{x}) \in [-1, 1]$)

Therefore, the darker the color the more uncertain the response, which is important to visualize how narrow (or broad) the boundary between the two classes is. Alternatively, the software displays the value of the classifier at the position indicated by the mouse cursor.

The Comparison dialog displays the results in terms of F-Measure, Precision, Recall or Classification Error, as well as their variance (see Figure 5)

²This form gives the same amount of importance to the misclassification of samples from Class 1 and Class 2. A generalised form of the F-Measure exists which takes into account the relative importance of the two classes. This is, however, beyond the scope of this practical.

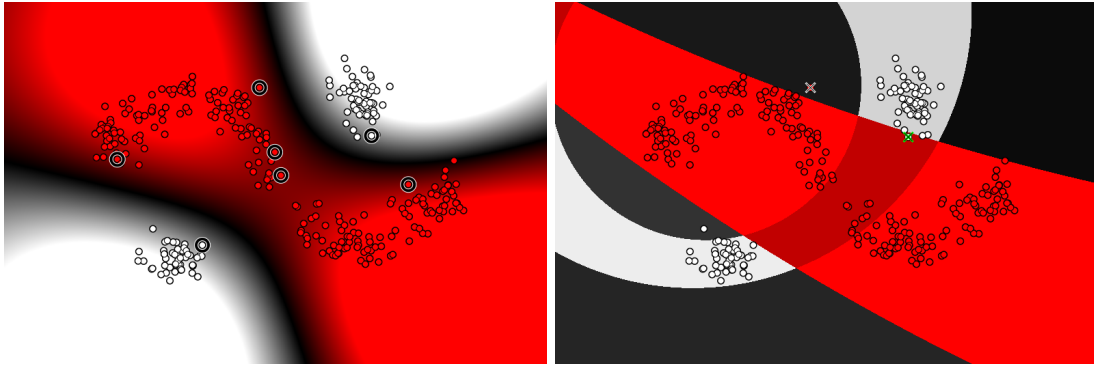


Figure 2: Example of the classification boundary: red pixels are classified as positive, white pixels as negative, and the amount of black displays the indecision. Left: classification using Support Vector Machines, Right: classification using Boosting of Random Circles.

Structure of the practicals

The following sections — Sections 6, 7 and 8 — describe the work to be conducted during the 4 weeks of practical from November 11 through December 2. Each section corresponds to about 3 hours of work (this includes the time needed to write the report). This is composed of one hour of contact with your assistants (between 11 and 12am) during which you can ask questions regarding the practical. It is then expected that you will work on your own for about two hours each week. We advise you to start writing your presentation as you gather data, i.e. jot down your thoughts as you progress in your analysis and save the plots you think are most relevant.

5 Part I: Binary Classification

5.1 Goals

In this first part of the practical, you will study two algorithms for binary classification. We use the term binary classification for two-class datasets. If the dataset contains several classes, binary classification is the task of recognising one single class against all other classes combined. The objective of the practical is to familiarise you with the classical methodology to measure performance in machine learning. Indeed, estimating the performance of a classifier is not easy, as the error measure is not a single value but is expressed in terms of true and false classifications. In particular, you should understand how cross-validation on your classification tasks is important to have an accurate estimation of the performance (see Section 1.3 of the Lecture Notes).

5.2 Linear Discriminant Analysis and Fisher-LDA

LDA and Fisher Linear Discriminant (Fisher-LDA) are very similar techniques for finding a direction along which separation of 2-class data is good. In contrast to PCA, which identified the direction of maximum variance of the data, irrespective of which class it belonged to, LDA explicitly looks for a projection that maximises the separation of two classes (see Figure 3).

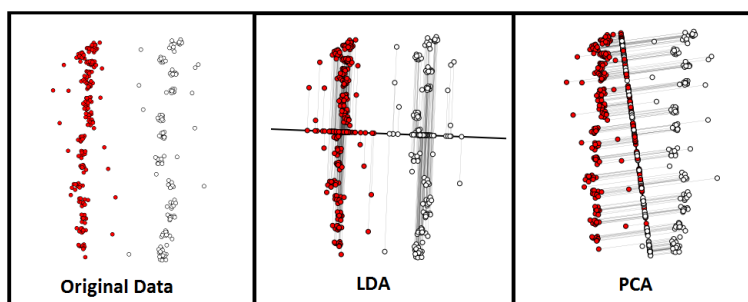


Figure 3: PCA and LDA projections. PCA is unable to provide a direction that facilitates the classification. In this case the information that distinguishes the two classes is not along the dimension of maximum spread. In contrast, the LDA projection is easily separable.

Standard LDA and Fisher-LDA differ in the way they model each class. Indeed LDA supposes the two classes to be of equal variance, while Fisher-LDA computes the variance separately for each class. You will find in Section 3.2 of the Lecture Notes the formal differences between these two methods. In practice, most implementations use Fisher-LDA by default. It can be informative to study their differences, although it can be difficult to find examples where it becomes apparent (see Figure 4). Currently implemented in MLDemos are the following flavours of LDA

- *LDA*: Standard LDA, takes into account the means and variance of the data (the same variance is assigned to each classes)
- *Fisher-LDA*: Fisher linear discriminant, takes into account the means and variance of the data, but computes two different variances for each class.

5.3 Support Vector Machines

The Support Vector Machine (SVM) is a supervised algorithm that can be used both for classification and (upon extension) for regression (so-called Support Vector Regression - SVR). The

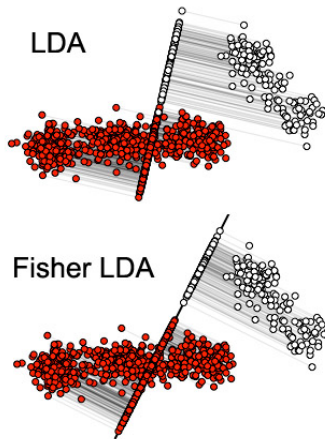


Figure 4: Differences between LDA and Fisher LDA. Since LDA does not make a difference in the covariance of the data, it is unable to cope with the unbalance in the data.

course material will enable you to understand more in depth the functioning of this algorithm. In this practical we will use C-SVM, which is the standard implementation of soft-margin SVM where a penalty (cost) is associated to classification errors during the training phase in which the algorithm selects its Support Vectors (SV). By adjusting the value of C it is possible to find a tradeoff between complexity of the decision function (i.e. how many SVs are selected) and the generalisation power of the algorithm (e.g. to avoid overfitting). The type of kernel used with SVM in MLDemos are:

$$\begin{array}{ll} \text{Polynomial} & k(x_i, x_j) = (x_i \cdot x_j)^d \\ \text{Radial Basis Function} & k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|) \end{array}$$

A Polynomial kernel of degree 1 ($d = 1$) corresponds to the linear kernel. The Radial Basis Function (RBF) kernel is usually called a Gaussian kernel. The γ parameter is sometimes referred to as the *kernel width*, as it determines how far a point can influence others. In the gaussian function γ is the inverse of the variance.

5.4 Boosting

Boosting is based on a simple concept: Instead of creating a single strong classifier right away, one first creates a large number of very simple classifiers (called weak learners) and then find a smart way of combining these together. The training process starts by generating a large number of weak learners (in MLDemos several thousands are generated) and identifying the one that provides the best classification, while giving equal importance to all samples. Once the first iteration is finished, the samples that the first learner is unable to classify are given more importance than the ones which are already properly classified, and a second learner is identified. Again, the samples which are incorrectly classified by both the first and second learner are given more importance, and the training continues. This effectively allows us to incrementally improve the classification by adding as many classifiers as we want.

MLDemos provides three types of weak learners:

- *Random Projections*: Project the samples along a random direction, and perform a naive bayes classification on this 1-Dimensional projection

- *Random Rectangles*: Select a random portion of space (a rectangle in 2D), if the sample is inside, the classifier returns 1
- *Random Circles*: Select a random point in space and compute the distance of the sample from this point, perform a naive bayes classification on this distance.

5.5 Performance Evaluation

Evaluating the performance of the classification is often done by playing with different amounts of training and testing samples. By changing this ratio, you will be giving more weight to the training of the classifier or to the test of its performance. By looking at the value of both testing and training F-Measures it is possible to identify problems with the data (e.g. very unbalanced data) or see when overfitting occurs.

Depending on the datasets you have evaluated, you might stumble upon some peculiar cases. For instance, you might obtain a 100% performance in training and 50% in testing; what could you infer from such a result? And from a 100% accuracy in both training and testing? These are just two examples of the type of question that you should consider when discussing the performances you are obtaining with respect to the training and testing samples amounts you have chosen.

5.5.1 Cross Validation

For the training/testing ratios you found in the previous section, run the classification several times, a different set will be generated each time. Evaluate in a quantitative way the performance. Is this performance consistent with what you found in the previous section? Evaluate the variance of your results. Is this affected by the choice of the training/testing ratio. You can do that using the COMPARE button (see Figure 5).

5.6 What to do

Study the differences between LDA, SVM and Boosting for the task of classifying the data in a binary way. For each algorithm, test the different parameters:

- *LDA*: test the 2 types of LDA projection
- *SVM*: test the 2 types of kernel, testing different values of C (the range 1-100 is a good starting point) and different kernel parameters (degree or kernel width, depending on the kernel type)
- *Boosting*: test the 3 types of weak learners; change the amount of weak learners (testing 1-20)
- *Train/Test ratio*: test the algorithms using different ratios of training/testing samples.

Run a cross-validation on these parameters to obtain an estimation of the variance of the results you obtain. Generate graphs and tables of the results you obtain (see an example in Figure 6).

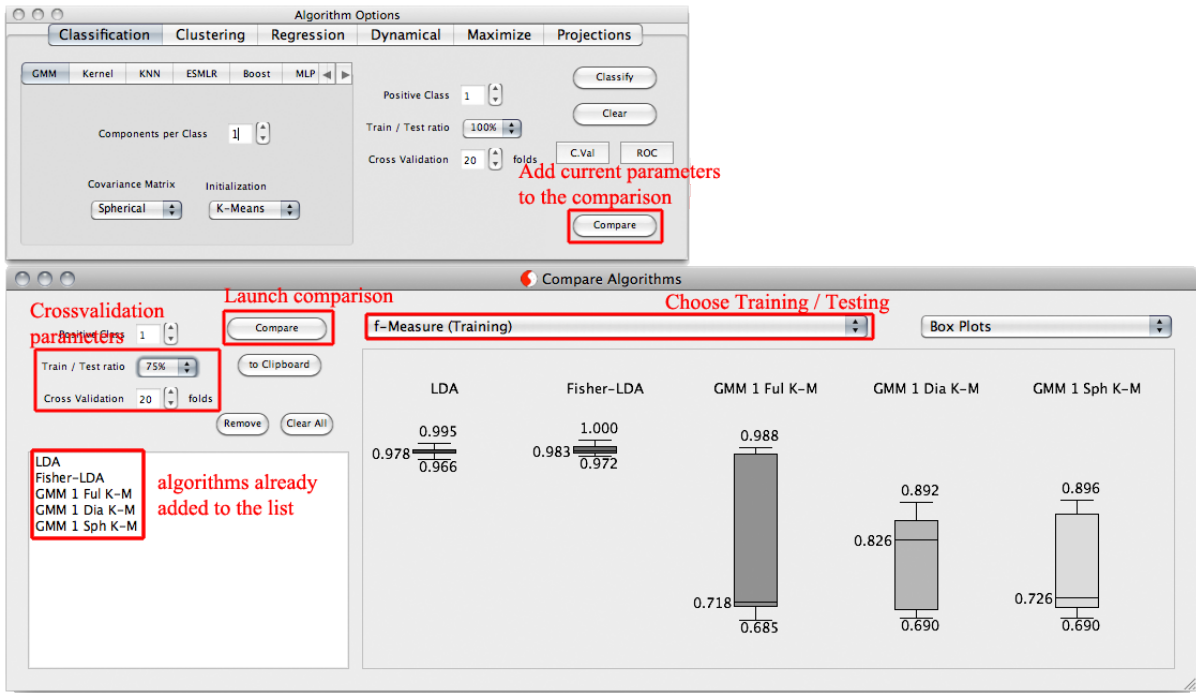


Figure 5: The Compare panel, to add an algorithm with its current parameters (e.g. covariance type, initialization, etc.) use the highlighted Compare button. Launching the compare process will test all algorithms and display the corresponding results.

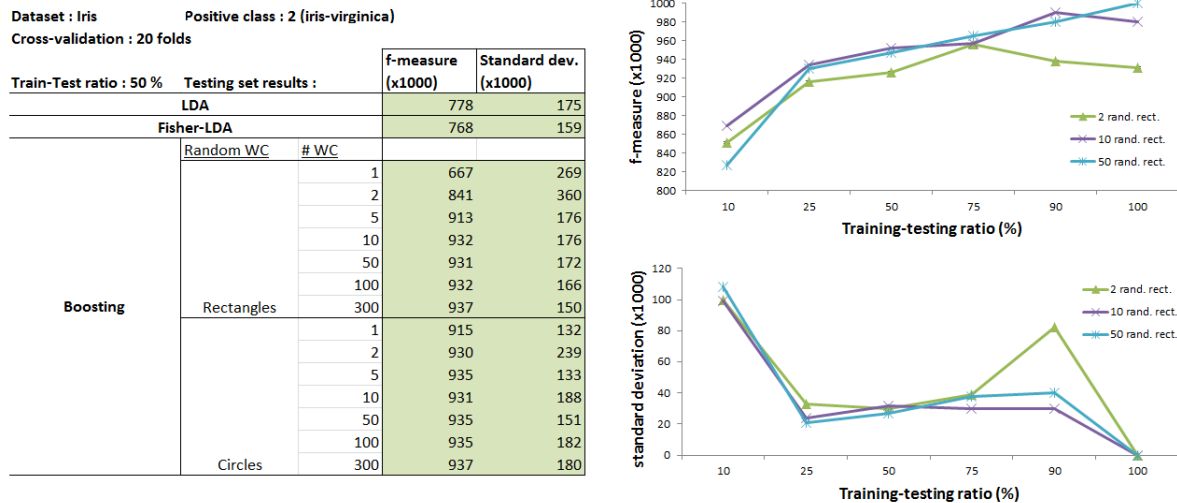


Figure 6: Classification performance (F-Measure) on the Iris dataset using LDA and Boosting. *LEFT* : table of the results using 50% of samples for training. *RIGHT* : graph of the variation of the F-Measure with different ratios of training/testing samples: notice how the variance of the results change with the amount of samples used for training.

6 Part II: Multi-Class Classification

6.1 Goals

The goal here is to study the general case of multi-class classification. Instead of a positive and negative class, we have now multiple class that we would like to identify concurrently. While

some algorithms allow to predict multiple classes, more often only a binary classifier is available. In this case a Winner Take All strategy is adopted:

1. Train a classifier to classify between pairs of classes (if c is the number of classes, there will be $\frac{c(c-1)}{2}$ classifiers)
2. Test a sample using each classifier individually (this will provide a *vote* to one class each time)
3. The class with the majority of votes wins
4. (*optional*) Compute the confidence of the classifier by looking at how distant the winning class is from the other classes

This strategy allows to use any binary classifier as a multi-class algorithm. However this increases the computation time quadratically with the number of classes.

6.2 Gaussian Mixture Models (GMM)

You have seen how GMM can be used to perform clustering. However, this algorithm can be used for classification as well. To do this, a separate GMM is trained for each class (in the case of clustering we took all samples from all classes together). Classification of a new data point is obtained by computing the difference between the response (likelihood) of each GMM. More formally, for a class model Θ_c , the classification response of a sample \mathbf{x} is given by

$$y = \underset{c}{\operatorname{argmax}} (p(\mathbf{x}|\Theta_c)) \quad (1)$$

In the case of GMM, the multi-class classification can be accomplished directly without learning multiple pairwise classifiers. Indeed, for each new class, we only need to learn one new GMM, thus growing linearly with the number of classes.

6.3 Support Vector Machines

Here we will use SVM with the Winner Take All approach to classify the data. Other approaches exist that can train a multi-class SVM in one single classifier, but they are outside the scope of this practical. Even in this form, however, SVM provides a powerful tool for multi-class classification.

6.4 What to do

Study the differences between GMM and SVM for the task of classifying the multi-class data. For each algorithm test the different parameters available:

- *GMM*: test the 3 types of covariance; change the amount of components per class (testing 1-10) and the initialization
- *SVM*: test the 2 types of SVM kernels; change the kernel degree or width (depending on the kernel); change the penalty factor C (testing 1-100)
- *Train/Test ratio*: test the algorithms using different ratios of training/testing samples.

NOTE: the number of components selected in GMM is provided for each class, if you have three class and two components, a total of 6 components will be drawn.

Run a cross-validation on these parameters to obtain an estimation of the variance of the results you obtain. Generate graphs and tables of the results you obtain (see an example in Table 1 and Figure 7).

<i>GMM (init: K-Means)</i>			<i>SVM</i>			
Cov. matrix	#of Comp	Error ($\pm std$)	Kernel	Degree	C	Error ($\pm std$)
Spherical	1	75 \pm 22	Poly.	1	1	13 \pm 54
	2	44 \pm 23		1	10	27 \pm 53
	5	31 \pm 30		1	100	13 \pm 67
	10	46 \pm 53		2	1	27 \pm 40
	20	321 \pm 218		2	10	27 \pm 40
Diagonal	1	44 \pm 22		2	100	27 \pm 53
	2	35 \pm 22		5	1	20 \pm 60
	5	54 \pm 55		5	10	27 \pm 40
	10	95 \pm 83		5	100	13 \pm 54
	20	376 \pm 181				
Full	1	25 \pm 18		γ	C	
	2	31 \pm 28		.01	1	240 \pm 507
	5	111 \pm 138		.01	10	173 \pm 427
	10	278 \pm 294		.01	100	240 \pm 440
	20	451 \pm 239		.1	1	7 \pm 246
			RBF	.1	10	0 \pm 160
				.1	100	20 \pm 193
				.5	1	27 \pm 26
				.5	10	13 \pm 67
				.5	100	0 \pm 67

Table 1: Classification performance (testing error) on the Iris dataset using GMM and SVM. The table displays the results using 50% of samples for training and 20 cross-validation folds.

7 Part III: Regression

7.1 Goals

In this part of the practical, you will study another application of some algorithms you have already seen: regression. While in a classification task the goal is to tell whether a sample belongs to a specific class, regression algorithms try to learn a (usually) continuous function. Once the algorithm has been trained, it is possible to estimate the value of the learned function for each sample in the input space.

7.2 Gaussian Mixture Regression

Gaussian Mixture Regression (GMR) uses a pre-existing GMM to learn the underlying function. During training, the output function is learned as one of the dimensions of the data. During testing, all other dimensions (the inputs) are used to estimate the most likely output. For this reason, GMR can compute not only the most likely output (the mean) but also the confidence associated to it (the likelihood). Figure 8 shows an example of this.

7.3 Support Vector Regression

In classification, SVM constructs the class boundary function using some training samples that are selected as Support Vectors. Support Vector Regression (SVR) differs only in the fact that instead of learning a boundary function, it learns the output function, by combining the Support Vectors together. To provide flexibility in the shape of the function, it introduces a parameter ϵ that determines the complexity of the function (see Figure 9).

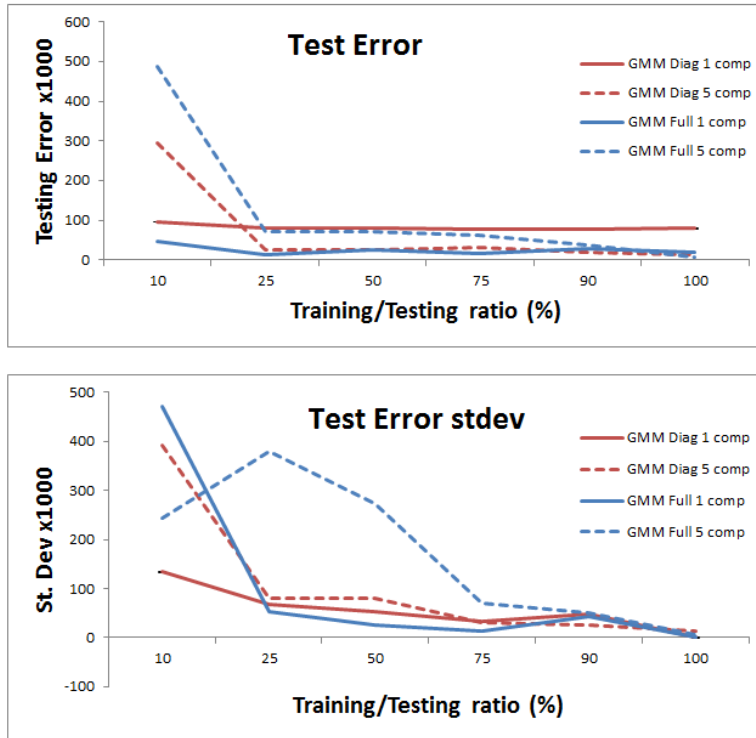


Figure 7: Graph of the variation of the testing error using different ratios of training/testing samples: notice how the variance of the results change with the amount of samples used for training.

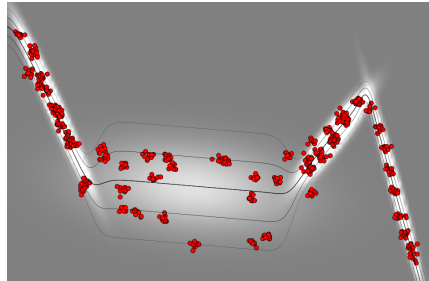


Figure 8: Regression using GMR. The grayscale background shows the likelihood in the different zones of the input/output domain. This allows to estimate how confident the output of the regression is.

7.4 What to do

For this task, you will use 2 dataset, one for a *qualitative* assessment of the algorithms, one for the *quantitative* evaluation of their performance. First, you will manually draw a 2D dataset to study *graphically* the effect of changing the parameters of the two algorithms (see Figures 8 and 9). Try to find a function that shows explicitly the advantages or disadvantages of each method. Use this 2D dataset to make a *qualitative* evaluation of the algorithms.

Second, you will be assigned a regression dataset on which you will perform the *quantitative* evaluation. The regression error is computed in terms of Mean Square Error (MSE) between the testing sample real value and its estimation. Moreover, the variance of the errors is provided.

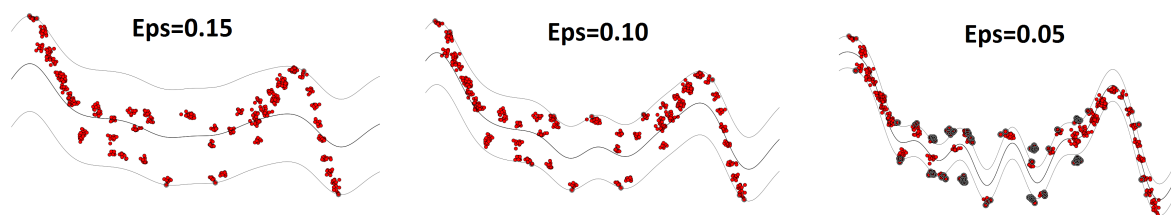


Figure 9: Regression using SVR. The image shows different values for the ϵ parameters, a *tube* is drawn around the mean corresponding to the value of ϵ . A large tube does not need to be very complex in shape to contain most samples, whereas a smaller one will result in a more complex shape.

This is done to get a feeling of how well the algorithm is modelling the underlying function. Indeed a method with a small error but a large variance may be less preferable than a method with a larger error but a very small variance: the second one is not good but at least it's reliable!

For both datasets, study the performance using different parameters:

- *GMR*: test the 3 types of covariance; change the amount of components per class (testing 1-10) and the initialization
- *SVR*: test the 3 types of SVM kernels; change the kernel degree or width (depending on the kernel); change the penalty factor C (testing 1-100); change the size of the ϵ -tube
- *Train/Test ratio*: test the algorithms using different ratios of training/testing samples.

Run cross-validation on these parameters to obtain an estimation of the variance of the results you obtain. Generate graphs and tables of the results you obtain (see examples in Figures 6 and 1).

8 Presentation

If you work in team of two, the presentation lasts 10 minutes. Each of the member of the team must speak for 5 minutes. If you are a team of three, the total duration of the presentation will be 15 minutes. A good rule of thumb is 1 slide per minute, so you should limit yourself to 10 slides. A possible structure for a presentation might be

- introduction to the problem (1 slide)
- datasets you are working with (e.g. source images + projections, peculiarities) (2 slides)
- methods compared, strengths and weaknesses (3 slides)
- qualitative + quantitative results (3 slides)
- conclusion and take-home-message (1 slide)

of course this is just a starting point and might not be optimal for your specific presentation. Present solely what you have done. Do not explain how the methods (e.g. SVM) works, but rather explain how you used it, which parameters you changed and what results you obtained. This is important as it will help you make a more general discussion of the methods you studied without focusing only on the practical aspects of what the software did.

Note A suggestion: rehearse your presentation! The golden rule is to go through it 4-5 times speaking aloud, and guess what, it actually works.

8.1 Format

As a general rule, the presentation should contain both a *qualitative* and a *quantitative* estimation of the performance of the system:

A qualitative evaluation should contain images (e.g. screenshots) which exemplify the concepts you want to explain (e.g. an image of a good projection and an image of a bad one). Make sure to plot only a subset of all the plots you may have visualized during the practical. Choose the ones that are most representative. Make sure that there is no redundancy in the information conveyed by the graphs and thus that each graph presents a different concept.

The quantitative evaluation should contain graphs and tables of the performance measures. If you include performance values, you should ALWAYS include mean and standard deviation of your data (preferably in the form of $mean \pm stdev$). If you cannot compute this (e.g. you only have one single performance value) you have done something wrong: rinse and repeat. Computing the standard deviation of your results also allows you to get an insight of how accurate the system is and to make reasonable comparisons (e.g. a method with an accuracy of $95\% \pm 10\%$ is not necessarily better than a method with an accuracy of $94\% \pm 1\%$).

The titles, axes and legends must be clearly explained in the graph or in the caption: unlike several reptiles and birds, human eyes do not have a zoom capability: be sure to write your axes, parameters, legends and tables in a properly sized font. The images below show an example of what you should and shouldn't do. A penalty will be given for each improperly defined graph!

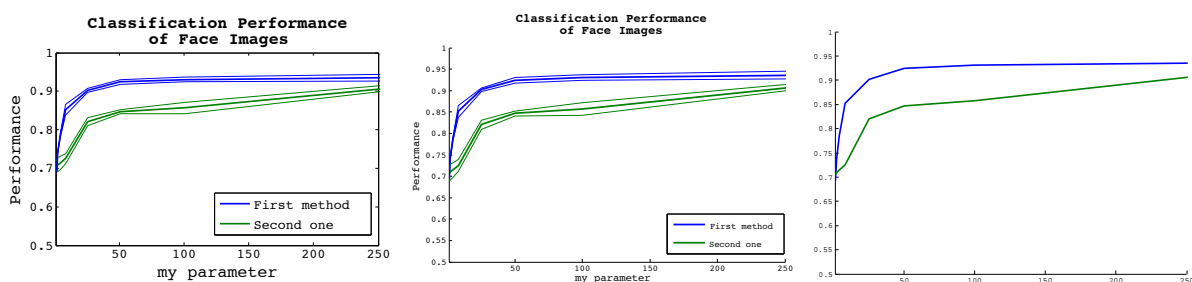


Figure 10: Examples of good, bad and worse ways of presenting your graphs. LEFT: the title and axes are readable, the legend is clear. CENTER: the title is there but is hardly readable, the axes and parameter names are too small to see. RIGHT: the axes are unreadable, we have no idea what the graph is representing, there is no legend, no standard deviation on the graphs themselves: graphs presented this way will be penalized.