Report : Operational Feature Selection in Gaussian Mixture Models

Adrien Lagrange
Dynafor

June 1, 2016

1 Introduction

...

2 Non linear parsimonious features selection

In the remaining, the following notations are used. $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ is the training set where $\mathbf{x}_i \in \mathbb{R}^d$ is the vector of features of the i^{th} sample, $y_i = 1, ..., C$ the associated label, C the number of labels, n the number of samples and n_c the number of samples of class c.

2.1 Gaussian Mixture Models

The hypothesis of Gaussian mixture models (GMM) is that the distribution of a given sample is a mixture (convex combination) of several class conditioned distribution:

$$p(\mathbf{x}) = \sum_{c=1}^{C} \pi_c f_c(\mathbf{x}|\theta) \tag{1}$$

where π_c is the prior i.e. the proportion of class c. The Gaussian model assumes that each f_c is , conditionally to c, a Gaussian distribution of parameters $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ ($f_c(\mathbf{x}|\theta) = \mathcal{N}_c(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$) and so $f_c(\mathbf{x}|\theta)$ can be written

$$f_c(\mathbf{x}|\theta) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_c|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^t \boldsymbol{\Sigma}_c^{-1}(\mathbf{x} - \boldsymbol{\mu}_c)\right)$$

Such a model is used in the case of supervised learning and the class parameters $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ can be estimated using the training samples. In our work, we choose to compute them with the following unbiased empirical estimator

$$\hat{\pi}_c = \frac{n_c}{n} \tag{2}$$

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{n_c} \sum_{\{i|y_i=c\}} \mathbf{x}_i \tag{3}$$

$$\hat{\boldsymbol{\Sigma}}_c = \frac{1}{(n_c - 1)} \sum_{\{i|y_i=c\}} (\mathbf{x}_i - \boldsymbol{\mu}_c)(x_i - \boldsymbol{\mu}_c)^t \tag{4}$$

To determine the class of a samples using the Bayes' law, the maximum a posteriori (MAP) estimate is used and thus the decision rule can be written as

$$\begin{aligned}\mathbf{x} \text{ belongs to } c &\Leftrightarrow c = \arg\max_{c \in C} p(c|\mathbf{x}) \\ &\Leftrightarrow c = \arg\max_{c \in C} \frac{p(c)p(\mathbf{x}|c)}{p(x)} \\ &\Leftrightarrow c = \arg\max_{c \in C} p(c)p(\mathbf{x}|c)\end{aligned}$$

By taking the log, a simplified decision formula is obtained

$$\begin{aligned}Q_c(\mathbf{x}) &= 2\log\left(p(c)p(\mathbf{x}|c)\right) \\ &= 2\log\left(\pi_c \frac{1}{(2\pi)^{\frac{d}{2}}|\mathbf{\Sigma}_c|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^t \mathbf{\Sigma}_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c)\right)\right) \\ &= 2\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^t \mathbf{\Sigma}_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c)\right) + 2\log\left(\frac{1}{(2\pi)^{\frac{d}{2}}|\mathbf{\Sigma}_c|^{\frac{1}{2}}}\right) + 2\log(\pi_c) \\ &= -(\mathbf{x}-\boldsymbol{\mu}_c)^t \mathbf{\Sigma}_c^{-1}(\mathbf{x}-\boldsymbol{\mu}_c) - \log((2\pi)^d|\mathbf{\Sigma}_c|) + 2\log(\pi_c) \qquad (5)\end{aligned}$$

It is interesting to notice the covariance and its inverse are a key element of the decision function and that the estimation of these elements suffer from the curse of dimensionality. More precisely, the number of parameters to estimate (means, covariance matrices, proportions) increases quadratically relatively to the number of features and we need at least as many samples as parameters which can be an issue. For example, with hyperspectral data, we face high dimensional samples but very few samples are available.

A lack of samples induces ill-conditioned covariance matrices and so unstable inversion. There are two major solutions to this problem. The first option is to use a regularization method to stabilize the inversion of the covariance matrices. The second option is to use a features extraction method in order to reduce the dimensionality of the samples.

In our study based on GMM, a Ridge regularization method described in Section 2.2 is implemented but we mainly focus on a feature selection method named sequential forward features selection presented in Section 2.3 and an amelioration of this algorithm the sequential floating forward features selection introduced in Section 2.4.

2.2 Ridge Regularization

As introduced in the previous section, the aim of a regularization method is to stabilize the inversion of the covariance matrix in the context of high dimensionality which often imply a ill-conditioned matrix. In particular, the Ridge regularization proposes to solve the problem in adding a small value τ to each eigenvalues of the covariance matrix. In our case, we choose to add the same constant to each eigenvalues but it is not compulsory in a general case.

Thus, to perform the regularization, the covariance matrix is decomposed with a diagonalization algorithm for symmetric matrices s.t. $\mathbf{\Sigma} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^t$ where $\mathbf{Q}$ is the matrix of eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues.

When the decomposition is available, the regularization almost finish and the decision function is simply rewritten as follow

$$Q_c(\mathbf{x}) = -(\mathbf{x} - \boldsymbol{\mu}_c)^t \mathbf{Q}_c (\boldsymbol{\Lambda}_c + \mathbf{T})^{-1} \mathbf{Q}_c^t (\mathbf{x} - \boldsymbol{\mu}_c) - \log((2\pi)^d |\boldsymbol{\Sigma}_c|) + 2\log(\pi_c) \tag{6}$$

where $\mathbf{T}$ is a diagonal matrix with all element set to τ. It is to be noticed that $(\boldsymbol{\Lambda} + \mathbf{T})$ is also a diagonal matrix and so the inverse is easily obtained in inverting each element of the matrix.

Finally, the main step of the regularization is the decomposition of the covariance matrix. But we need also to take into account that the choice of τ is not automatic. In order to find the right τ, the user has to perform a gridsearch which is a major drawback of this method.

2.3 Sequential forward features selection

The Sequential Forward Selection (SFS) starts with an empty set of selected features. Then it tests at each step for all the remaining features the value of a given criterion function when the feature is added to the pool of selected features. The feature that maximizes the criterion function is definitively added to the pool of selected features and it moves to the next iteration. The choice of the criterion function is discussed afterward in Section 2.5. Depending of the stopping criterion chosen by user, the algorithm stops either when a given number of variables has been selected or either when the percentage of increase of the criterion function is too low.

To summarize, the features are selected one by one and the set of selected features are never put into questions. It results in a reasonable computational time but a suboptimal solution to the selection problem.

2.4 Sequential floating forward feature selection

The Sequential Floating Forward Selection (SFFS) is actually based on two algorithms: the SFS described above and the Sequential Backward Selection (SBS). The SBS is the backward equivalent of SFS. The difference is that it starts with every features in the pool of selected features and tries at each step to remove the less significant one in term of a given criterion function.

The SFFS works as the SFS but between each step of the SFS algorithm a backward selection is operated. At the end of the SBS step, the value of the criterion function is compared to the best value ever obtained with a set of features of the same size and if the new value is better the feature puts into question is effectively taken away and the next step is again a SBS but if the new value is not better the SBS step is forgotten and it moves to the next SFS step. The algorithm stops when a given number of features has been selected.

This SFFS algorithm eventually tests more solutions than the SFS algorithm. The results are expected to be better but the trade-off is an increased computational time which is dependent of the complexity of the dataset.

2.5 Criterion function

Both of the presented algorithms rely on a criterion function which can be chosen by the user. We decided to test the most frequent criterion functions. These functions can be divided in two groups. The first group is based on measures of good classification and the second on measure of similarity between distribution of probabilities.

2.5.1 Measures of good classification

All this measures of good classification are based on an error matrix M called confusion matrix which is defined so that M_{ij} is the number of samples of class i classified as class j. The confusion matrix allows the computation of several interesting values relatively to each class the number of True Positive (TP) corresponding to good prediction, True Negative (TN) corresponding to good classification of the other class, False Negative (FN) corresponding to the samples of the class labeled as an other class and the False Positive (FP) corresponding to the samples wrongly classified as part of of this class. Figure 1 illustrates the definition of this values.

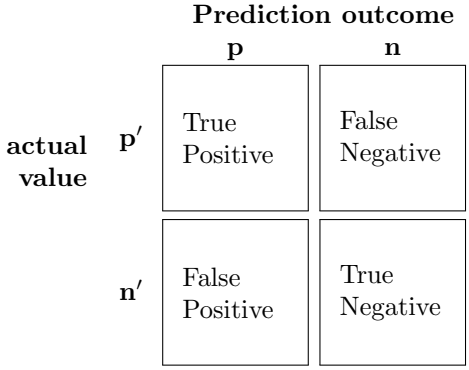

Figure 1: Confusion matrix

The overall accuracy is simply the rate of the number of samples with the correct predicted label over the number of samples. This metric is easy to interpret but is biased in the case of unbalanced classes.

The Cohen's kappa is a statistic which measure the probabilities of agreement between predictions and ground-truth. This metric tends to give an equal importance to each class but is hard to interpret.

The mean F1 score is the average of the F1 score for each class and the F1 score is the harmonic mean of the precision (number of True Positive over True Positive plus False Positive) and the recall (number of True Positive over True Positive plus False Negative).

$$\text{Overall Accuracy} = \frac{\text{TP}}{n} \tag{7}$$

$$\text{Kappa} = \frac{pa - pr}{1 - pr} \tag{8}$$

$$\text{F1 Mean} = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}} \tag{9}$$

where TP stands for True Positive, FN for False Negative, FP for False Positive, pa is the probability of agreement defined by $pa = OverallAccuracy$ and pr the probability of random agreement defined by $pr = \sum_{c=1}^{C} \frac{TP_c}{FP_c} \frac{TP_c}{FN_c}$.

In order to compute these metrics, the predicted labels are needed and so we need to compute the decision function (Equation 5). Moreover to estimate this measures of good classification, a cross-validation method is used which reduces both bias and variance of the estimation.

2.5.2 Measures of similarity between distributions

The Kullback–Leibler divergence is a measure of distance between two distributions. It actually measures the amount of information lost when the first distribution is approximated by the second one. The formal definition is

$$\text{Div}_{KL}(c_i, c_j) = \int_{\mathbf{x}} p(\mathbf{x}|c_i) \ln(\frac{p(\mathbf{x}|c_i)}{p(\mathbf{x}|c_j)}) d\mathbf{x} \tag{10}$$

And in the case of Gaussian model, it can be rewritten as follows

$$\text{Div}_{KL}(c_i, c_j) = \frac{1}{2}\Bigg(\text{Tr}(\boldsymbol{\Sigma}_{c_i}^{-1}\boldsymbol{\Sigma}_{c_j}) + (\boldsymbol{\mu}_{c_i} - \boldsymbol{\mu}_{c_i})^t \boldsymbol{\Sigma}_{c_i}^{-1}(\boldsymbol{\mu}_{c_i} - \boldsymbol{\mu}_{c_i}) - k + \log\left(\frac{|\boldsymbol{\Sigma}_{c_i}|}{|\boldsymbol{\Sigma}_{c_j}|}\right)\Bigg) \tag{11}$$

where Tr is the trace operator and k the dimension of the distribution.

It can be noticed that the KL divergence is not symmetric and so the symmetrized version is used to compute the criterion function. In the case of Gaussian model, the symmetrization induces the following simplification of the formula

$$\frac{1}{2}(\text{Div}_{KL}(c_i, c_j) + \text{Div}_{KL}(c_j, c_i)) = \frac{1}{4}\Bigg(\text{Tr}(\boldsymbol{\Sigma}_{c_i}^{-1}\boldsymbol{\Sigma}_{c_j} + \boldsymbol{\Sigma}_{c_j}^{-1}\boldsymbol{\Sigma}_{c_i}) + (\boldsymbol{\mu}_{c_i} - \boldsymbol{\mu}_{c_i})^t (\boldsymbol{\Sigma}_{c_i}^{-1} + \boldsymbol{\Sigma}_{c_j}^{-1})(\boldsymbol{\mu}_{c_i} - \boldsymbol{\mu}_{c_i}) \Bigg) \tag{12}$$

Moreover, the divergence is computed between two classes and to obtain a unique value the weighted mean of divergence measures is taken

$$\text{Div}_{KL} = \sum_{c_i=1}^{C} \sum_{c_j=c_i+1}^{C} \pi_{c_i}\pi_{c_j} \frac{1}{2}(\text{Div}_{KL}(c_i, c_j) + \text{Div}_{KL}(c_j, c_i)) \tag{13}$$

The Bhattacharyya distance is an other measure of similarity between two distributions. In fact, we do not discuss the efficiency of this measure in our work and simply used it to compute the next measure. The Bhattacharyya distance is computed as follows

$$\mathrm{B}_{ij} = -\ln\left(\int_{\mathbf{x}} \sqrt{p(\mathbf{x}|c_i)p(\mathbf{x}|c_j)}d\mathbf{x}\right) \tag{14}$$

And in the case of Gaussian model:

$$\mathrm{B}_{ij} = \frac{1}{8}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^t \left(\frac{\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j}{2}\right)^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) + \frac{1}{2}\ln\left(\frac{|\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j|}{\sqrt{|\boldsymbol{\Sigma}_i||\boldsymbol{\Sigma}_j|}}\right) \tag{15}$$

The Jeffries–Matusita distance is a measure based on the Bhattacharyya distance but transform in a way that the distance saturate if the separability between the two distribution increases. The JM distance is defined according to

$$\mathrm{JM}_{ij} = \sqrt{\int_{\mathbf{x}} \left[\sqrt{p(\mathbf{x}|c_i)} - \sqrt{p(\mathbf{x}|c_j)}\right]^2 d\mathbf{x}} \tag{16}$$

And the Jeffries–Matusita distance can be rewritten according to the Bhattacharyya distance

$$\mathrm{JM}_{ij} = \sqrt{2\{1 - \exp[-B_{ij}]\}} \tag{17}$$

As for the KL divergence, a weighted mean of the distance between two classes is computed to aggregate the measures in a single value.

According to [1], it is interesting to notice that the KL divergence increases quadratically with respect to the distance between the mean vectors of the class distributions whereas the measures of good classification we used asymptotically tends to one when distributions are perfectly separable. On the contrary, the JM distance tends to saturate as these measures of good classification.

3 Implementation

In term of computational efficiency, the main issue is to find an efficient way to compute the mean vector and the covariance matrix and its inverse and to do it only when it is absolutely necessary. More specifically, we develop an optimized way to derive the submodel for cross-validation from the full model and we minimize the number of costly operations during a step of the selection algorithm.

3.1 Update for cross validation

Based on [2], a method to accelerate the cross-validation process in the case of criterion functions based on correct classification measures was implemented. The idea is to estimate the GMM model with the whole training set and when the training set is split in several folds to use this model to derive the model corresponding to a subset of the original training set.

The following formulae can be obtained (details of calculation in Appendix A)

$$\boldsymbol{\mu}_c^{n_c-\nu_c} = \frac{n_c\boldsymbol{\mu}_c^{n_c} - \nu_c\boldsymbol{\mu}_c^{\nu_c}}{n_c-\nu_c} \tag{18}$$

$$\boldsymbol{\Sigma}_c^{n_c-\nu_c} = \frac{1}{n_c-\nu_c-1}((n_c-1)\boldsymbol{\Sigma}_c^{n_c} - (\nu_c-1)\boldsymbol{\Sigma}_c^{\nu_c} - \frac{n_c\nu_c}{(n_c-\nu_c)}(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})^t) \tag{19}$$

where ν_c is the number of samples of class c removed from the set.

Additionally, the update of the GMM model is the occasion to precompute the constants $2\log(\pi_c)$ which is a term of the decision function constant for a given training set.

3.2 Criterion function computation

As explained earlier, at each iteration the SFS and SFFS algorithms compute the value of a criterion function for every possible set of features composed of the selected ones augmented by one of the remaining features. One of the main achievements of this work is to reduce the computational time needed to compute the criterion function of the augmented sets.

We note $\boldsymbol{\Sigma}_{k-1}$ the covariance matrix of the $k-1^{th}$ iteration, i.e., the covariance of the selected features and $\boldsymbol{\Sigma}_k$ the covariance matrix of the k^{th} iteration, i.e., the covariance of the augmented set. Then, the terms $\boldsymbol{\Sigma}_k^{-1}$, $\mathbf{x}_k^t\boldsymbol{\Sigma}_k^{-1}\mathbf{x}_k$ and $|\boldsymbol{\Sigma}_k|$ can be expressed in function of terms of the $k-1^{th}$ iteration. These update rules are summed up hereafter and are available in [3] (chapter 9.2).

As $\boldsymbol{\Sigma}_k$ is a positive definite symmetric matrix, we can use the following notation

$$\boldsymbol{\Sigma}_k = \begin{bmatrix} \boldsymbol{\Sigma}_{k-1} & \mathbf{u} \\ \mathbf{u}^t & \sigma_{kk} \end{bmatrix}$$

where $\mathbf{u}$ is the k^{th} column of the matrix without the diagonal element i.e. $\mathbf{u}_i = \boldsymbol{\Sigma}_{i+1,k}$ with $i \in [1, k-1]$.

Using the formula of the inverse of a block matrix, the following formula expressing $\boldsymbol{\Sigma}_k^{-1}$ in function of $\boldsymbol{\Sigma}_{k-1}^{-1}$ is obtained

$$\boxed{\boldsymbol{\Sigma}_k^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}_{k-1}^{-1} + \frac{1}{\alpha}\boldsymbol{\Sigma}_{k-1}^{-1}\mathbf{u}\mathbf{u}^t\boldsymbol{\Sigma}_{k-1}^{-1} & -\frac{1}{\alpha}\boldsymbol{\Sigma}_{k-1}^{-1}\mathbf{u} \\ -\frac{1}{\alpha}\mathbf{u}^t\boldsymbol{\Sigma}_{k-1}^{-1} & \frac{1}{\alpha} \end{bmatrix}} \tag{20}$$

where $\alpha = \sigma_{kk} - \mathbf{u}^t\boldsymbol{\Sigma}_{k-1}^{-1}\mathbf{u}$.

In the case of a backward step in SFFS algorithm, we need to invert the formula in order to compute $\boldsymbol{\Sigma}_{k-1}^{-1}$ knowing $\boldsymbol{\Sigma}_k^{-1}$. Then, if we use the following notation for the previous matrix

$$\boldsymbol{\Sigma}_k^{-1} = \begin{bmatrix} \mathbf{A} & \mathbf{v} \\ \mathbf{v}^t & \frac{1}{\alpha} \end{bmatrix}$$

with $\mathbf{A} = \boldsymbol{\Sigma}_{k-1}^{-1} + \frac{1}{\alpha}\boldsymbol{\Sigma}_{k-1}^{-1}\mathbf{u}\mathbf{u}^t\boldsymbol{\Sigma}_{k-1}^{-1}$ and $\mathbf{v} = -\frac{1}{\alpha}\boldsymbol{\Sigma}_{k-1}^{-1}\mathbf{u}$, we can calculate

$$\begin{aligned} \mathbf{A} - \alpha\mathbf{v}\mathbf{v}^t &= \boldsymbol{\Sigma}_{k-1}^{-1} + \frac{1}{\alpha}\boldsymbol{\Sigma}_{k-1}^{-1}\mathbf{u}\mathbf{u}^t\boldsymbol{\Sigma}_{k-1}^{-1} - \alpha(-\frac{1}{\alpha}\boldsymbol{\Sigma}_{k-1}^{-1}\mathbf{u})(-\frac{1}{\alpha}\mathbf{u}^t\boldsymbol{\Sigma}_{k-1}^{-1})) \\ &= \boldsymbol{\Sigma}_{k-1}^{-1} \end{aligned}$$

$$\boxed{\boldsymbol{\Sigma}_{k-1}^{-1} = \mathbf{A} - \alpha \mathbf{v}\mathbf{v}^t} \tag{21}$$

A formula can also be deduced for the quadratical term of the decision function. If we note $\mathbf{x}_k^t = \left[\begin{array}{cc} \mathbf{x}_{k-1}^t & x_k \end{array} \right]$,

$$\begin{aligned}
\mathbf{x}_k^t \boldsymbol{\Sigma}_k^{-1} \mathbf{x}_k &= \left[\begin{array}{cc} \mathbf{x}_{k-1}^t & x_k \end{array} \right] \left[\begin{array}{cc} \mathbf{A} & \mathbf{v} \\ \mathbf{v}^t & \frac{1}{\alpha} \end{array} \right] \left[\begin{array}{c} \mathbf{v} \\ x_k \end{array} \right] \\
&= \left[\begin{array}{cc} \mathbf{x}_{k-1}^t \mathbf{A} + x_k \mathbf{v}^t & \mathbf{x}_{k-1}^t \mathbf{v} + \frac{x_k}{\alpha} \end{array} \right] \left[\begin{array}{c} \mathbf{v} \\ x_k \end{array} \right] \\
&= \mathbf{x}_{k-1}^t \mathbf{A} \mathbf{x}_{k-1} + x_k \mathbf{v}^t \mathbf{x}_{k-1} + \mathbf{x}_{k-1}^t \mathbf{v} x_k + \frac{x_k^2}{\alpha} \\
&= \mathbf{x}_{k-1}^t (\boldsymbol{\Sigma}_{k-1}^{-1} + \frac{1}{\alpha} \boldsymbol{\Sigma}_{k-1}^{-1} \mathbf{u}\mathbf{u}^t \boldsymbol{\Sigma}_{k-1}^{-1}) \mathbf{x}_{k-1} + 2 x_k \mathbf{v}^t \mathbf{x}_{k-1} + \frac{x_k^2}{\alpha} \\
&= \mathbf{x}_{k-1}^t (\boldsymbol{\Sigma}_{k-1}^{-1} + \alpha \mathbf{v}\mathbf{v}^t) \mathbf{x}_{k-1} + 2 x_k \mathbf{v}^t \mathbf{x}_{k-1} + \frac{x_k^2}{\alpha} \\
&= \mathbf{x}_{k-1}^t \boldsymbol{\Sigma}_{k-1}^{-1} \mathbf{x}_{k-1} + \alpha (\mathbf{x}_{k-1}^t \mathbf{v}\mathbf{v}^t \mathbf{x}_{k-1} + 2 \frac{x_k}{\alpha} \mathbf{v}^t \mathbf{x}_{k-1} + \frac{x_k^2}{\alpha^2}) \\
&= \mathbf{x}_{k-1}^t \boldsymbol{\Sigma}_{k-1}^{-1} \mathbf{x}_{k-1} + \alpha (\mathbf{x}_{k-1}^t \mathbf{v} + \frac{x_k}{\alpha})^2 \\
&= \mathbf{x}_{k-1}^t \boldsymbol{\Sigma}_{k-1}^{-1} \mathbf{x}_{k-1} + \alpha (\left[\begin{array}{cc} \mathbf{v}^t & \frac{1}{\alpha} \end{array} \right] \mathbf{x}_k)^2
\end{aligned}$$

$$\boxed{\mathbf{x}_k^t \boldsymbol{\Sigma}_k^{-1} \mathbf{x}_k = \mathbf{x}_{k-1}^t \boldsymbol{\Sigma}_{k-1}^{-1} \mathbf{x}_{k-1} + \alpha (\left[\begin{array}{cc} \mathbf{v}^t & \frac{1}{\alpha} \end{array} \right] \mathbf{x}_k)^2} \tag{22}$$

Finally, using the formula of the determinant of a block matrix and after taking the log, we obtain

$$\boxed{\log(|\boldsymbol{\Sigma}_k|) = \log(|\boldsymbol{\Sigma}_{k-1}|) + \log \alpha} \tag{23}$$

To give an example, if a measure of good classification is used, we need to compute the decision rule and with the update rules, it can be rewritten as follow

$$\begin{aligned}
Q(\mathbf{x}) = &- (\mathbf{x}_{k-1} - \boldsymbol{\mu}_{k-1})^t \boldsymbol{\Sigma}_{k-1}^{-1} (\mathbf{x}_{k-1} - \boldsymbol{\mu}_{k-1}) \\
&+ \alpha (\left[\begin{array}{cc} \mathbf{v}^t & \frac{1}{\alpha} \end{array} \right] (\mathbf{x}_k - \boldsymbol{\mu}_k))^2 \\
&- \log(|\boldsymbol{\Sigma}_{k-1}|) + \log \alpha + 2 \log(\pi)
\end{aligned}$$

It can be noticed that this new formula allow us to precompute the computationally heavy terms because they are the same for all possible augmented sets at a given iteration of the selection.The Figure 2 illustrates the optimization of the selection algorithm.

Nevertheless, during an iteration of the selection algorithm, it still needs to invert the covariance matrix of the previous iteration. In order to so, we choose to perform a decomposition in eigenvalues and eigenvectors using algorithm specific for symmetric matrices. In other words, we find $\boldsymbol{\Lambda}$ the diagonal matrix of eigenvalues and $\mathbf{Q}$ the matrix of eigenvectors s.t. $\boldsymbol{\Sigma} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^t$.

This decomposition has several advantages. First, it is very simple to compute the determinant once the eigenvalues are available because the determinant is equal to the product of these values. Secondly, it is possible to check

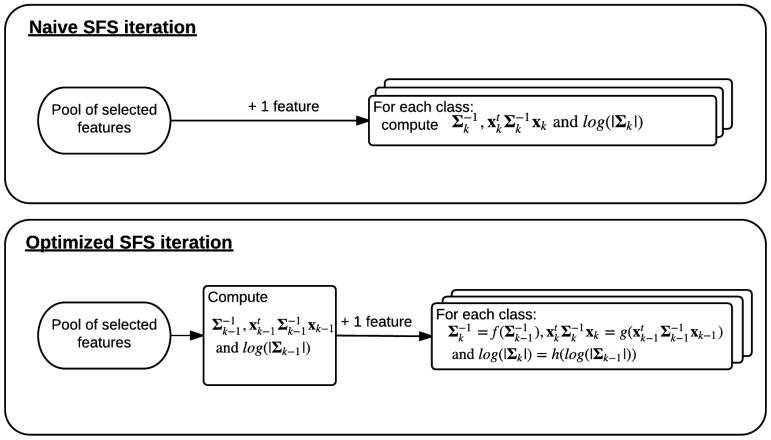

Figure 2: Iteration of sequential forward features selection.

the eigenvalues and so to assure a better computational stability. Due to the curse of dimensionality, the covariance matrix is sometimes ill-estimated and the results is that some eigenvalues are really small (below computational precision) and the decomposition helps us to check their actual values and if they are below EPS_FLT which the machine maximum precision for a float, we set these eigenvalues to EPS_FLT.

4 Test with real data

5 Implementation of the Orfeo Toolbox module

After the calibration of the algorithm with a Python implementation, we aim to realise a C++ implementation of this selection method. And, to make it easily available, we choose to develop a remote module for the Orfeo Toolbox (OTB) designed by the CNES. The current section describes all the choice made during the C++ implementation.

5.1 Integration in Orfeo Toolbox

The module developed is available on Github [1] and is actually a fork of the template for remote module furnished by OTB developers.

We choose to implement our own version of a GMM classifier in a class named *GMMMachineLearningModel* in order to assure better performance in term of computational time. The second step is to implement a subclass of the GMM classifier including the features selection method.

[1] `https://github.com/Laadr/otbExternalFastFeaturesSelection`

5.2 GMM classifier

The *GMMMachineLearningModel* class inherits from the OTB class *MachineLearningModel*. It implements the virtual method *Train()* and *Predict()* inherited from the super class and on additional method *Decomposition()* used to decompose a symmetric matrix in eigenvalues and eigenvectors.

In this class, all measurement vectors are of type *itk::VariableLengthVector* and all other vectors are of type *std::vector*. The various matrix are of type *itk::VariableSizeMatrix*.

The super class of GMMMachineLearningModel enables the management of a list of samples used for training. This class includes 6 members describing the model and computed in the *Train()* function : the number of class, the number of features, a vector containing the proportion of each class in the training set, a vector containing the number of samples in each class, a vector containing the mean vector of each class and finally a vector containing the covariance matrix of each class.

Moreover, the decision function used for prediction is rewritten in order to assure faster computation. We use the following equivalent of Equation 5

$$Qc(\mathbf{x}) = -||\mathbf{\Lambda}_c^{-\frac{1}{2}}\mathbf{Q}_c^t(\mathbf{x}-\boldsymbol{\mu}_c)||^2 - \log(|\mathbf{\Sigma}_c|) + 2\log(\pi_c) \tag{24}$$

where $\mathbf{\Lambda}_c$ and $\mathbf{Q}_c$ are obtained in decomposing $\mathbf{\Sigma}_c$ in eigenvalues and eigenvectors s.t. $\mathbf{\Sigma}_c = \mathbf{Q}_c\mathbf{\Lambda}_c\mathbf{Q}_c^t$. Thus, we can precompute several terms of the decision function in the *Train()* function. Eigenvalues, eigenvectors, the terms $\mathbf{\Lambda}_c^{-\frac{1}{2}}\mathbf{Q}_c^t$ and the terms $-\log(|\mathbf{\Sigma}_c|) + 2\log(\pi_c)$ are stored in vectors as class members.

The decompositions in eigenvalues and eigenvectors of the covariance matrices are performed with the *Decomposition()* function which is a wrapper to instance the class *itk::symmetricEigenAnalysis* and call its method *ComputeEigenValuesAndVectors* (this method uses netlib routines). Immediately after decomposition, eigenvalues are examined and all eigenvalues inferior to EPSILON_FLT (or EPSILON_DBL) are reset to this value. The idea is to limit computational error when parameters are ill-estimated due to an insufficient amount of training samples.

The Ridge regularization described in Section 2.2 is also implemented in this class. If the value of τ is set before training, the regularization is performed during the precomputation of the terms of the decision function and if τ is modified after training, the function *SetTau()* updates the precomputation in addition to changing the value of τ.

A Update for cross validation (calculation)

A.1 Mean update

$$\begin{aligned}
\boldsymbol{\mu}_c^{n_c} &= \frac{1}{n_c}\sum_{j=1}^{n_c}\mathbf{x}_j \\
&= \frac{1}{n_c}\sum_{j=1}^{n_c-\nu_c}\mathbf{x}_j + \frac{1}{n_c}\sum_{j=n_c-\nu_c+1}^{n_c}\mathbf{x}_j \\
&= \frac{n_c-\nu_c}{n_c}\boldsymbol{\mu}_c^{n_c-\nu_c} + \frac{\nu_c}{n_c}\boldsymbol{\mu}_c^{\nu_c} \\
&= \boldsymbol{\mu}_c^{n_c-\nu_c} + \frac{\nu_c}{n_c}(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c})
\end{aligned}$$

$$\boxed{\boldsymbol{\mu}_c^{n_c} = \boldsymbol{\mu}_c^{n_c-\nu_c} + \tfrac{\nu_c}{n_c}(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c})} \tag{25}$$

$$\boxed{\boldsymbol{\mu}_c^{n_c-\nu_c} = \tfrac{n_c\boldsymbol{\mu}_c^{n_c} - \nu_c\boldsymbol{\mu}_c^{\nu_c}}{n_c-\nu_c}} \tag{26}$$

A.2 Covariance matrix update

$$\begin{aligned}
\boldsymbol{\Sigma}_c^{n_c} &= \frac{1}{n_c-1}\sum_{j=1}^{n_c}(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c})(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c})^t \\
&= \frac{1}{n_c-1}\sum_{j=1}^{n_c}(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-\nu_c} - \frac{\nu_c}{n_c}(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c}))(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-\nu_c} - \frac{\nu_c}{n_c}(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c}))^t \\
&= \frac{1}{n_c-1}\sum_{j=1}^{n_c}(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-\nu_c})(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-\nu_c})^t \\
&\quad + \frac{\nu_c^2}{n_c^2}(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c})(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c})^t \\
&\quad - \frac{\nu_c}{n_c}(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-\nu_c})(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c})^t \\
&\quad - \frac{\nu_c}{n_c}(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c})(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-\nu_c})^t
\end{aligned}$$

- $(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c})(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c})^t = \frac{n_c^2}{(n_c-\nu_c)^2}(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c})^t$
- $\frac{1}{n_c}\sum_{j=1}^{n_c}(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-\nu_c})(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c})^t = \frac{n_c\nu_c}{(n_c-\nu_c)^2}(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c})^t$
- $\frac{1}{n_c}\sum_{j=1}^{n_c}(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c-\nu_c})(\mathbf{x}_j - \boldsymbol{\mu}_c^{n_c-\nu_c})^t = \frac{n_c\nu_c}{(n_c-\nu_c)^2}(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^{\nu_c} - \boldsymbol{\mu}_c^{n_c})^t$

$$
\begin{aligned}
\boldsymbol{\Sigma}_c^{n_c} &= \frac{1}{n_c-1}\sum_{j=1}^{n_c}(\mathbf{x}_j-\boldsymbol{\mu}_c^{n_c-\nu_c})(\mathbf{x}_j-\boldsymbol{\mu}_c^{n_c-\nu_c})^t - \frac{\nu_c^2}{(n_c-\nu_c)^2}(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})^t \\
&= \frac{1}{n_c-1}\sum_{j=1}^{n_c-\nu_c}(\mathbf{x}_j-\boldsymbol{\mu}_c^{n_c-\nu_c})(\mathbf{x}_j-\boldsymbol{\mu}_c^{n_c-\nu_c})^t + \frac{1}{n_c-1}\sum_{j=n_c-\nu_c+1}^{n_c}(\mathbf{x}_j-\boldsymbol{\mu}_c^{n_c-\nu_c})(\mathbf{x}_j-\boldsymbol{\mu}_c^{n_c-\nu_c})^t \\
&\quad - \frac{n_c\nu_c^2}{(n_c-1)(n_c-\nu_c)^2}(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})^t \\
&= \frac{n_c-\nu_c-1}{n_c-1}\boldsymbol{\Sigma}_c^{n_c-\nu_c} + \frac{1}{n_c-1}\sum_{j=n_c-\nu_c+1}^{n_c}(\mathbf{x}_j-\boldsymbol{\mu}_c^{n_c-\nu_c})(\mathbf{x}_j-\boldsymbol{\mu}_c^{n_c-\nu_c})^t \\
&\quad - \frac{n_c\nu_c^2}{(n_c-1)(n_c-\nu_c)^2}(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})^t
\end{aligned}
$$

$$
\begin{aligned}
\sum_{j=n_c-\nu_c+1}^{n_c}(\mathbf{x}_j-\boldsymbol{\mu}_c^{n_c-\nu_c})(\mathbf{x}_j-\boldsymbol{\mu}_c^{n_c-\nu_c})^t &= \sum_{j=n_c-\nu_c+1}^{n_c}(\mathbf{x}_j-\boldsymbol{\mu}_c^{\nu_c}+\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c-\nu_c})(\mathbf{x}_j-\boldsymbol{\mu}_c^{\nu_c}+\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c-\nu_c})^t \\
&= (\nu_c-1)\boldsymbol{\Sigma}_c^{\nu_c} + \nu_c(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c-\nu_c})(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c-\nu_c})^t \\
&\quad + \sum_{j=n_c-\nu_c+1}^{n_c}(\mathbf{x}_j-\boldsymbol{\mu}_c^{\nu_c})(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c-\nu_c})^t \\
&\quad + \sum_{j=n_c-\nu_c+1}^{n_c}(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c-\nu_c})(\mathbf{x}_j-\boldsymbol{\mu}_c^{\nu_c})^t \\
&= (\nu_c-1)\boldsymbol{\Sigma}_c^{\nu_c} + \frac{n_c^2\nu_c}{(n_c-\nu_c)^2}(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})^t
\end{aligned}
$$

$$
\boxed{\boldsymbol{\Sigma}_c^{n_c} = \tfrac{n_c-\nu_c-1}{n_c-1}\boldsymbol{\Sigma}_c^{n_c-\nu_c} + \tfrac{\nu_c-1}{n_c-1}\boldsymbol{\Sigma}_c^{\nu_c} + \tfrac{n_c\nu_c}{(n_c-1)(n_c-\nu_c)}(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})^t} \tag{27}
$$

$$
\boxed{\boldsymbol{\Sigma}_c^{n_c-\nu_c} = \tfrac{1}{n_c-\nu_c-1}((n_c-1)\boldsymbol{\Sigma}_c^{n_c} - (\nu_c-1)\boldsymbol{\Sigma}_c^{\nu_c} - \tfrac{n_c\nu_c}{(n_c-\nu_c)}(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})(\boldsymbol{\mu}_c^{\nu_c}-\boldsymbol{\mu}_c^{n_c})^t\ )} \tag{28}
$$

References

[1] Lorenzo Bruzzone and Claudio Persello. A novel approach to the selection of spatially invariant features for the classification of hyperspectral images with improved generalization capability. *Geoscience and Remote Sensing, IEEE Transactions on*, 47(9):3180–3191, 2009.

[2] Mathieu Fauvel, Clément Dechesne, Anthony Zullo, and Frederic Ferraty. Fast forward feature selection of hyperspectral images for classification with gaussian mixture models. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 8(6):2824–2831, 2015.

[3] Andrew R Webb. *Statistical pattern recognition*. John Wiley & Sons, 2003.