

# Data Analysis and Model Classification

## Guidesheet I-4: PCA and Forward feature selection

Fumiaki Iwane      Ping-Keng Jao      Bastien Orset      Julien Rechenmann  
Ricardo Chavarriaga      José del R. Millán

In the previous guidesheet, you were requested to select among features based on a ranking. This week, we will expand the possible ways of information extraction by combining features either by changing feature selection method or by engineering new features.

### Principal Component Analysis

PCA is an unsupervised technique commonly used for dimensionality reduction. PCA allows to find, starting from a set of observations in a  $N$ -dimensional feature space, a new  $N$ -dimensional feature space having  $N$  orthonormal vectors (principal components or PCs), defined to maximize the variance of the original data.

By projecting the original data into the feature space defined by the PCs, one can generate uncorrelated features which are linear combinations of the original ones. A subset of these uncorrelated features are then usually used as input for further machine learning methods such as classification, regression, etc. In this exercise we will transform the original feature space by means of PCA and use the result of this transformation as input for classification.

**Hands on** Use the command `[coeff, score, variance] = pca(X)` in Matlab to apply PCA to your features. The function outputs the coefficients of the PCs, the data projected on principal components, and the variances of the principal components. BEWARE: by default, the function also centers the data before applying PCA. As a result, the output, `score`, is obtained through projecting `score = X_centered * coeff`.

- Compute the covariance matrix of the original data and the covariance matrix of the data projected on the PCs (use the Matlab function `cov`). You can visualize covariance matrix, for example, by using function `imshow`.
  - Compare the diagonal elements (variances) of the two covariance matrices. What does this information reveal about the variance of the original features and the ones transformed with PCA? What does this mean in terms of informative power of such features? How could this information be used to choose the most informative features?
  - Compare the off-diagonal elements (covariances) of the covariance matrices of the data in the two feature spaces. What does this information reveal about the correlation between the original features and the ones transformed with PCA? What does this mean in terms of correlation between the features? Which are the maximum covariances values for the original data and for the transformed data?
  - The number of PCs to be retained is a hyperparameter in your problem. One possible way of choosing this number is to look at the total information (variance) carried by such PCs and choose to keep  $\bar{M}$  PCs that represent  $m\%$  of the total variance of the data (the value of  $m$  is, obviously, dependent on the type of problem and the expected outcomes). Compute the cumulative variance (as function of every PC) w.r.t. to the total variance (use the command `cumsum(variance)/sum(variance)`) and graph this result. Which number of features would you select in order to represent  $\sim 90\%$  of the total variance of the data?

- Implement PCA within your cross-validation procedure without feature ranking, instead select first  $N$  PCs and treat  $N$  as a hyperparameter. You can fix the classifier type for now. Which subset(s) of data will you use to estimate PCs? How will you apply it on unseen data? What is the optimal  $N$ ? Is it the same as the number you got from thresholding 90% of cumulative variance?
- Normalize the data using `zscore`. Do you normalize before applying PCA or after? Does it change anything? Why? Which subset(s) do you use to estimate the normalization parameters?

## Forward feature selection

The downside of last week's *Fisher*-based algorithm is that it only looked at the individual discriminability of features. However, if two features are correlated (e.g. take the extreme case where the correlation is 1), then adding both of them in the classifier will not add any additional information compared to if we would take only one of them.

Forward selection starts by selecting the best performing (in terms of classification error) single feature. Then, all pairwise combinations of the remaining features with the already selected one are tested. The one with the best (joint) performance is selected, leaving us with two features. Then, we add a third feature that leads to the best joint performance with two features already selected, and so on.

MATLAB comes with this function already implemented: `sequentialfs()`. Please, read the help carefully before using it! `sequentialfs()` performs itself the *inner-loop* cross-validation to determine the optimal number of features; basically, the function uses a *criterion* to evaluate the performance of a feature combination, and stops adding features when there is no more improvements in the prediction criterion. In our case, we will continue define the criterion as follows:

```
fun = @(xT,yT,xt,yt) length(yt)*(your_error(yt,predict(fitcdiscr(xT,yT,'discrimtype',
classifiertype), xt)));
```

To start, set the `classifiertype` by `classifiertype = 'diaglinear'`; and the options by `opt = statset('Display','iter','MaxIter',100)`; The complete command for feature selection is:

```
[sel,hst] = sequentialfs(fun,features,labels,'cv',cp,'options',opt);
```

where `features` and `labels` are the training features and training labels, respectively, and `cp` is a cross-validation object created by `cvpartition()`. You will find the *optimal validation* error in `hst.Crit(end)`.

### General questions

- For both feature selection methods (*Fisher* score/forward feature selection), are they a *filter* or a *wrapper*? **Fisher is a filter, forward is a wrapper (features are evaluated by building a model)**
- How does the model selection differ in *Fisher* score and forward features selection?

### Hands on

- Apply `sequentialfs()` instead of the *rankfeat* feature selection. Do you want to use all the features or just an a priori pre-selected subset? **????????????? I would love to know it**
- How many features were selected? How does this compare with the *Fisher* score method?

**With the forward method a lower number of features is identified. This is due to the fact that, as with Fisher score the features selected can be correlated, a higher number of features will be needed compared to the Forward selection.**