Guide sheet summaries DAM

Contents

[Guide sheet 1 2](#_Toc528493858)

[Statistical significance 2](#_Toc528493859)

[Feature thresholding 2](#_Toc528493860)

[Guidesheet2 3](#_Toc528493861)

[Discriminant classifiers 3](#_Toc528493862)

[Important considerations regarding discriminant analysis 3](#_Toc528493863)

[Linear Discriminant Classifiers 4](#_Toc528493864)

[Quadratic Discriminant Classifiers 4](#_Toc528493865)

[Application 4](#_Toc528493866)

[Training and testing errors 4](#_Toc528493867)

[Cross-validation for performance estimation 4](#_Toc528493868)

[Guide sheet 3 5](#_Toc528493869)

[Cross validation for hyperparameter optimization 5](#_Toc528493870)

[Pseudocode 5](#_Toc528493871)

[Result 5](#_Toc528493872)

[Why test classification error estimates are optimistically biased 5](#_Toc528493873)

[Nested cross-validation for performance estimation 6](#_Toc528493874)

[Pseudocode 6](#_Toc528493875)

[Result 6](#_Toc528493876)

[Guide sheet 4 6](#_Toc528493877)

[Principle Component Analysis 6](#_Toc528493878)

# Guide sheet 1

## Statistical significance

We classify data by analysing samples at selected features. But how do we select these features? For a feature to be useful, we need class-respective feature data distributions to be significantly different. Naturally, since we need to examine class-respective distributions, we need to select features using training data (sample classification is given).

Approach to ensure statistical significance:

1. Histograms allow us to visually expect data distributions. Note that since data does not necessarily contain equal amounts of samples from each class, it is important to normalize histograms prior to comparing them.
2. Boxplots are another way of visually inspecting a data distribution. Boxplots information includes: median (different from mean-> middle value not average), 25&75 percentiles, outliers. Notched boxplots are especially interesting because a notch in the box provides the 95% confidence interval of the median.

When examining our data, we can compare class-respective notched boxplots at a given feature to determine whether their medians are significantly different (no overlap in confidence intervals).

1. Student’s T-test is useful because it allows us to compare the means of class-respective datasets at a given feature. Note that t-test operates under the assumption that the datasets we compare are normally distributed and of equal variance. This prevents us from using the T-test ubiquitously.

T-testing returns a p-value which is the probability of observing the given result under the assumption that dataset means are identical. Therefore, a very low p-value is indicative of a significant difference between datasets. Usually we refute the null hypothesis that dataset means are identical if p-value is below 0.05.

Using this approach, we were able to determine that class-respective data differed significantly (high-discriminability) at feature 712.

## Feature thresholding

At a feature displaying high-discriminability, we can define a threshold used to separate data from each class. A simple classification scheme can then be established: Given a sample, check whether its value at FOI (feature of interest) is greater or less than threshold.

Since this model is simplistic and probably not very accurate, we need to be able to evaluate it. We use the following accuracy quantifiers:

; ;

Note that class weights must sum to 1 and that they must reflect the composition of training dataset.

Class error is more advantageous than classification error when training is not balanced (more of one class than another). Say we have 99% class A and 1% class B and we systematically predict A, then model would have classification error of 1% despite it being very bad! For such a model class error would be 50%! However, if 50% was class A and 50% was class B then classification error and class error would be equal (50%).

The weights in class error formula are useful because we can attribute importance to different types of error. For example, we could choose to ignore a certain type of error or emphasize another. Why emphasize a particular error class? Once again to reflect dataset. If we are using a big weight on a class comprised of a few data points it doesn’t make sense. It would allow anecdotal data to override the general consensus.

In practice, we established a 1-D model based on thresholding at feature 712. Initially threshold was chosen visually using histograms. Then we plotted model errors as a function of this value and chose the value that minimized class error as the optimal threshold.

We decided that we could scale up the model but that we needed to choose additional features carefully to avoid adding noise. This means repeating the process that we went through to select 712.

# Guidesheet2

## Discriminant classifiers

We wish to sort our dataset comprised of 597 observations (each comprised of around 2’000 variables) into 2 groups. A discriminant function calculates a weighted average of values of independent variables for each observation. The weights are selected so that the resulting average separates the observations into groups. High values of the average come from one group, low values of the average come from another group. The problem reduces to one of finding the weights which, when applied to the data, best discriminate among groups according to some criterion.

How do we determine these weights? We start with sums of squares matrices: We determine two matrices Stotal; Sgrouped representing the cumulative distance of observations from overall mean and group means respectively. We then define a third matrice Sdiff = Stotal - Sgrouped . The weight coefficients we seek are contained in the eigenvectors of Sgrouped -1 Sdiff .

Other matrices of interest are: overall covariance matrix T, within-group covariance matrix W, between-group covariance matrix A. These are all simply calculated based on the S matrices previously mentioned.

### Important considerations regarding discriminant analysis

Data needs to be screened prior to discriminant analysis. Checklist:

1. We need more observations per group than independent variables. Sorting 597 observations into 2 groups, since error group has prior 30%, say we expect around 200 observations in error class. This means we need to reduce our 2’000 features (variables) down to sub-200. We reduce data set dimensionality by factor of 10!
2. If class frequency in training Data is not representative of prior probabilities, we need to explicitly specify them.
3. Check training data for outlier observations in each class. Try to eliminate them because they can could disrupt classifier building.
4. Singular covariance matrices disturb classifier building. When this arises, it means one predictor variable (feature) is linearly dependent on the others. Solution is to reduce dimensionality of predictor variables used (features).

### Linear Discriminant Classifiers

* Equal class covariance matrices (W). W can be either full or diagonal. Diagonal means that only variance of features is considered whereas full means that we account for covariance between features.

### Quadratic Discriminant Classifiers

* Different class covariance matrices (W). Same full/diagonal implications as for linear case.

### Application

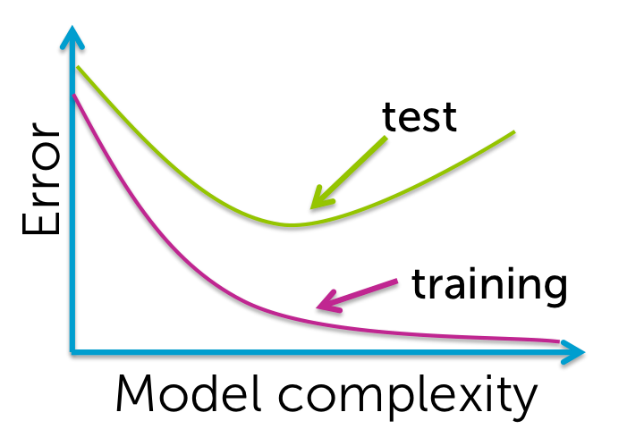
In practice, we see that best classifiers are those which account for feature covariance (full class covariance matrix). To use quadratic discriminant classifier with full class covariance matrices, we need to reduce feature dimensionality to avoid singular matrices appearing.

Specifying prior probabilities to uniform (equal) is a not a good idea. Class frequency in training Data reflects priors relatively well. Setting priors to uniform decreases classifier performance.

Class error is more useful than classification error to quantify classifier performance given that we do not know class frequencies in test Data.

## Training and testing errors

Still trying to determine which discriminant analysis classifier best suits our data, we decide to compute training and test errors using subgroups of the training dataset. Interestingly we see that models displaying reduced training error do not perform and better for test Data (test error). This is a sign of overfitting.



By default, without explicit specification, Matlab’s fitdiscr function uses all available features as parameters. Obviously, this yields a model that performs extremely well for training Data but extremely badly for real Data. It is not robust! We need to select parameter features more selectively.

When forming training subgroup of data, it is important to conserve class frequencies that reflect real prior probabilities. If this is not the case, then we must specify the priors explicitly.

## Cross-validation for performance estimation

Given a dataset, we want to maximize data used for training and minimize data used for testing while retaining an accurate performance assessment. To do this we use k-fold cross validation. This means we divide original data into k subsets of equal size. From there, we use k-1 of these sets to train a model and the remaining set to test it. We then cycle the training and test sets until every subset has been used for both training and testing.

Every cycle generates a prediction error. Values of interest we calculate are the mean and standard deviation across all cycles. The standard deviation gives us an indication of the stability of our model (stable is model error is consistent).

# Guide sheet 3

## Cross validation for hyperparameter optimization

The hyperparameter in question is the number of features to be used in a linear discriminant classifier (with a diagonal class covariance matrix). So, we aim to determine the optimal number of features through cross-validation.

### Pseudocode

For nFeatures=1 to maxNumberFeatures

For PartitionCycleIndex=1 to numberOfFolds

Compute training and testing markers

Use markers to establish testing and training subsets

Compute feature ranking based on testing data

Select the most powerful nFeatures from ranking

Establish linear classifier using selected features

Use classifier to compute predictions for training and testing data respectively

From these predictions compute training and test classification errors

Store these 2 errors

End

Compute average classification error and standard deviation displayed when using nFeatures

End

Plot number of features vs training; test classification error.

### Result

We can use plot to select number of features with minimum test classification error. We conclude that optimal number of feature to user for a linear discriminant classifier is 3.

### Why test classification error estimates are optimistically biased

Our approach worked great to find the hyperparameter (number of features) we sought out. It is still important to note that when our classifier with 3 features faces real data it will not perform with the same classification error. This is simply because we used the same data to both determine a model (hyper)parameter and evaluate performance. Therefore, we still need a different test set to get an unbiased evaluation our newly selected model. The way to overcome this problem is to have nested cross validations.

## Nested cross-validation for performance estimation

Take whole data set, establish subsets called outer folds. These folds are used for computation of test error that indicates selected model performance.

Each outer fold is subdivided into inner folds. These inner folds are used to compute validation error of a changing model with the goal of choosing a hyperparameter.

### Pseudocode

### Result

# Guide sheet 4

## Principle Component Analysis

A technique used for feature filtering (dimensionality reduction). Principal component analysis (PCA) is a statistical procedure that uses an [orthogonal transformation](https://en.wikipedia.org/wiki/Orthogonal_transformation) to convert a set of observations of possibly correlated features into a set of values of [linearly uncorrelated](https://en.wikipedia.org/wiki/Correlation_and_dependence) variables called principal components. A subset of PCs is then selected for input for classifiers.

To do so, PCA finds the best fitting line through the data that maximise the sum of square distances from the projected points to the origin. This given line is called *principal component*.

Terminology: the normalized vector along the principal component is called the *eigenvector* and the sum of square distance is referred as *eigenvalue* for the principal component.

By reducing dimensionality, PCA get rid of low variance dimensions and therefore maximise the variance of the original data. In terms of informative power, it means that the information content carried by the projected data is higher (*i.e.* lower entropy).

Each principal component cutting through the scatterplot represents a decrease in the system’s entropy, in its unpredictability.

*The first component of PCA, like the first if-then-else split in a properly formed decision tree, will be along the dimension that reduces unpredictability the most*.

## Forward feature selection

Up until now we used fisher scores to rank the discriminative power of features. The problem with this is that if features are correlated, we could be selecting redundant features. I.e. the number 2 on list is powerful but strongly correlated to feature number 1 and hence brings no new information (think entropy).

Note: in Matlab, when calling sequentialfs(), use ‘Prior’ uniform as input arguments (if not the number of features will not be greater than one).

### Procedure

1. Choose a Model
2. Use model limited to 1 feature. Determine which feature minimizes model classification error.
3. Use model limited to 2 features. Use a combination of feature selected in step 2 and any other feature. Determine which secondary feature (in conjunction with the primary one) minimizes classification error.
4. Use model limited to 3 features. Same thing as step 3. ETC
5. Continue until adding features no longer improves performance of the model.

### General questions

1. Compare fisher scoring to forward feature selection.

Fisher scoring is a filter; forward feature selection is a wrapper. Fisher scoring is great because we can apply it to all features in our data set (low computational cost) but it doesn’t account for correlation between features. Forward feature selection is also great and can be used in conjunction with fisher scoring. It allows to select features while taking into account how features interact together. Since it is a wrapper and costs a lot computationally, we would only use this method on a feature subset. Such a subset could be determined by selecting a group of high achieving features from fisher scoring.