Data Analysis and Model Classification
Guidesheet II: data exploration

Ruslan Aydarkhanov Bastien Orset Julien Rechenmann
Ricardo Chavarriaga José del R. Millán

Introduction

Any scientific study involves the collection of data. But in most cases, this proves to be time consuming and expensive, which leads to rather small datasets being collected. In order to draw valid conclusions from little data it has to be analyzed statistically. A good way to start the data exploration is to 'play around' with it, i.e. look at plots and statistical parameters to get a feeling for the data (more analytic methods exist but they are outside of the scope of this exercise). Then one can dissect the data a little more in depth with statistical tests and scores. This already gives an idea which properties of the whole dataset are specifically explanatory, and could therefore be used for modeling purposes.

Statistical significance

Often classes look different for a certain feature, but we do not know if this difference is just because we were lucky when measuring our samples. To objectively validate if there is a real difference or not, i.e. if the difference is **statistically significant**, we can use a statistical test.

Confidence interval The purpose of taking random samples from a population and computing a statistic, e.g. the mean from the data is to approximate the real mean of the population. How well the sample statistic estimates the underlying population value is always an issue. A confidence interval with confidence level of x% provides a range of values which is x%-likely to contain the population parameter of interest. Using the command `boxplot(X,'Notch','on')` will additionally show you the 95% confidence interval of the median.

t-test The two-sample t-test tests datasets from two classes A and B for the **null hypothesis** that the distributions from both classes have equal means (i.e. no difference between them). Here "two-sample" should be better named "two-class" to fit our nomenclature. The command `[h,p] = ttest2(x,y)` for x and y containing the samples of class A and B yields `h`, a binary decision about the null hypothesis (yes/no; default confidence level) as well as `p`, the p-value. The p-value is defined as the probability of obtaining a result equal to or "more extreme" than what was actually observed, given the null hypothesis is true. The lower the p-value, the more significant is the observed difference in the mean values.
Attention! The t-test is only valid if the samples come from normal distributions with equal covariances!

Hands on

- Use the histogram function to find examples of features with very similar and very different statistical distributions between classes (check after the 300th features).
- Create a boxplot of these features and compare it regarding the different classes.
- Repeat the boxplot you did before, but add the "Notch" parameter. How does this change your view?

- In which range are the p-values you get with the t-test for "good" and "bad" features (i.e. those that you identified before has having different and similar distributions per class, respectively)?
- Can you simply use the t-test for all of your features? If not, what exactly is the problem?
- If you apply your t-test to all of your features separately and you find one that is significant, does that mean that the distributions of the two classes are necessarily different?

Feature thresholding

Now that we are able to detect and visualize features with the highest discrimination between the two classes, we can already try a simple classification scheme. For one dimensional data a threshold can be defined where samples on one side of it (values below threshold) would be classified as class A, and samples on the other side (above threshold) as class B. The `line` command can be used to visualize the threshold in a 2D plot. For a feature `f` with the respective threshold `tf` you can use the comparative operators "<" and ">" to find the predicted classes of each sample. For a vector of samples `sf` (each of the elements is only the corresponding feature f) the command `(sf < tf)` gives back a vector which elements are one if the statement is correct (sample classified as class A) or zero if the statement is wrong (sample classified as class B). When we compare the labels from the classifier with the true labels that are given, then we can compute the **classification accuracy** and **classification error**:

$$\text{classification accuracy} = \frac{\text{number of correctly classified samples}}{\text{number of total samples}}$$
$$\text{classification error} = \frac{\text{number of misclassified samples}}{\text{number of total samples}} = 1 - \text{classification accuracy}$$

These numbers are good estimators of how well a classification algorithm is performing, but only if the classes are balanced, i.e. if both classes have approximately the same number of samples. For example, if 90% of the samples collected belong to class A, a classifier that always predicts class A will achieve more or less 90% classification accuracy. To balance the influence of errors in class A and class B, we define the **class error**:

$$\begin{aligned}\text{class error} =& \frac{1}{2}\frac{\text{number of misclassified samples of class A}}{\text{number of total samples of class A}} + \\ & \frac{1}{2}\frac{\text{number of misclassified samples of class B}}{\text{number of total samples of class B}}\end{aligned}$$

Hands on

- How can the class histograms for a feature help you to optimally place the threshold?
- Plot the datapoints of two features (2D) and plot the optimal threshold for feature one as a line. What are the particular shortcomings of feature thresholding?
- What is the classification error on your dataset? How different is it from the class error?
- Can you scale this method up to multiple dimensions (e.g. using two features at once)?
- Plot class error and classification error in function of threshold values. Describe what you see.

Generalization

Now that you have found a simple way of classifying your dataset, you should ask yourself how generalizable your classifier is, i.e. how well will your classifier perform on a new dataset.

Hands on

- Split your dataset into two sets `set1` and `set2` corresponding respectively to your training and testing set.
- Check the ratio of your classes on your training and testing and the original ratio.Is this information should be take into account in the split process?Why? Is this a problem in our case? If no, why?
- Find the optimal threshold on your training set and check the performance you obtain on the training and testing set with the classification error and the class error metrics.
- Comparing the results, explain which metrics you will use to assess the performance of your classifier?And on which set?
- Try different percentage for the split (10% to 90%). For each percentage estimate the error estimation and conclude about the generalization of your classifier.