P06 – ML Diagnostics

1. Tracing the curse of dimensionality[1]

When the number of features p is large, there tends to be a deterioration in the performance of kNN and other *local* approaches that perform prediction using only observations that are *near* the test observation for which a prediction must be made.

This phenomenon is known as the *curse of dimensionality*, and it corresponds to the fact that non-parametric approaches often perform poorly when p is large. You will now investigate this "curse".

1.1 Suppose that you have a set of observations with p = 1 features, $x_{i1}, i = 1..N$. Assume that the x_{i1}'s are uniformly (evenly) distributed on [0, 1]. Associated with each observation is a response value (label, y).

Suppose that one wishes to predict a test observation's response using the majority response from only those training observations that are within 10%[2] of the range of the x_{i1} closest to that test observation. For instance, in order to predict the response for a test observation with x_1= 0.6, one will use observations in the range [0.55, 0.65].

On average, what fraction of the available observations will be used to make the prediction?

1.2 Now suppose that you have a set of two-dimensional observations, each with measurements on p = 2 features, x_{i1}, x_{i2}. Assume that they are uniformly distributed on [0, 1] × [0, 1].

One wishes to predict a test observation's response using only observations that are within 10% of the range of x_{i1} and within 10% of the range of x_{i2} closest to that test observation. For instance, in order to predict the response for a test observation with x_1 = 0.6 and x_2= 0.35, one will use observations in the range [0.55, 0.65] for x_1 and in the range [0.3, 0.4] for x_2.

On average, what fraction of the available observations will be used to make the prediction?

[1] Based on exercise 4 of chapter 4.7 in [ISL, 2014].
[2] Meaning $\pm 5\%$, based on the range of the data → since the data range is [0,1], this is ± 0.05.

1.3 Now suppose that you have a set of observations on p = 100 features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. One wishes to predict a test observation's response using observations within the 10% of each feature's range that is closest to that test observation. What fraction of the available observations will be used to make the prediction?

1.4 Using your answers to parts 1-3, argue that a drawback of kNN when p is large is that there are very few training observations "near" any given test observation.

1.5 Now suppose that one wishes to make a prediction for a test observation by creating a p-dimensional hypercube centred on the test observation that contains, on average, 10% of the training observations. For p = 1, 2, and 100, what is the length of each side of the hypercube? Comment on your answer.

Hints:

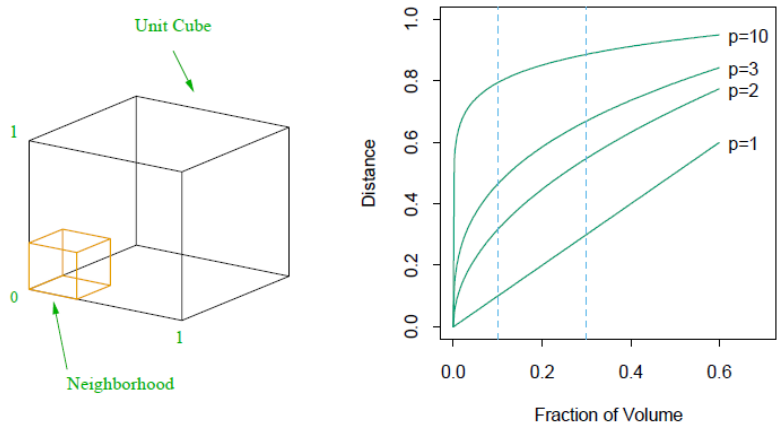


Figure 1: A small cube of side length s inside a larger unit cube (left). Right: Edge length of a cube needed to cover a given volume of the unit cube as a function of the number of dimensions. The vertical axis called "distance" plots the side length s.

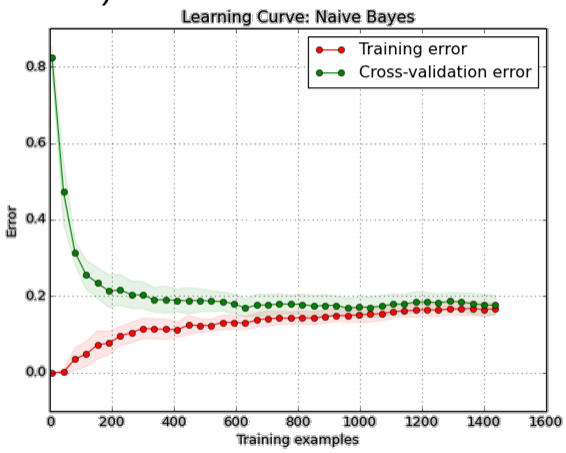
While still in $p < 3$ dimension, draw sketches to get intuition into the connections between the fraction and dimensionality like seen above (figure 2.6 from [ESL, 2009]). See section 2.5 in [ESL, 2009] for explanations that lead to a formulated solution.

2. Applying learning curves

In this exercise you will learn to apply learning curves to get insight into the learning process. Consider the Python script **`learning_curves.py`**. It makes heavy use of **`scikit-learn`** and is directly ready to run.

2.1 Make yourself familiar with the script:

- Its behaviour is controlled via the hyper parameters at the beginning of the code
- The following section defines a data structure **`estimators`** that contains some classifier objects as well as meta data for each (the names)
- The main function loads a dataset that ships with scikit-learn, then trains a classifier several times on different sizes of the dataset using a function from the **`learning_visualization.py`** library (just a wrapper around **`scikit-learn`**)



2.2 Execute the script:

- How well does naïve Bayes do on the **`digits`** data in your opinion?
- Read about the **`digits`** data (link is in the code) – does this explain the observed performance?
- Is there a bias or variance problem (or none)?

2.3 Experiment with different classifiers and diagnose if they also suffer from high bias or high variance. Interpret the results you get.

- SVM with RBF kernel
- SVM with linear kernel

2.4 Try Random Forest as a fourth algorithm and find out which influence the **`n_estimators`** parameter has on the result.

- What is the meaning of **`n_estimators`**? Read the **`scikit-learn`** documentation (online).
- Try several values. How does increasing this parameter affect bias & variance?
- Can you put the function **`plot_vc()`** from the **`learning_visualization.py`** library to use to help you get an overall picture of the effect of **`n_estimators`**?

2.5 [Optional] There are several other datasets included in **`scikit-learn`**, for example the famous **`iris`** data. Try your findings on some other dataset. Consult the **`scikit-learn`** documentation to find out more about the data and how to handle it. Additionally, try more algorithms and parameters to get familiar with their **`scikit-learn`** interface.