P06.1 Sample Solution

1. Tracing the curse of dimensionality

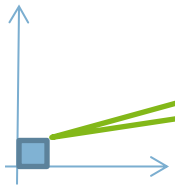
1.1 On average, what fraction of the available observations will be used to make the prediction ($p = 1$)?

➔ On average, 10%. For simplicity, cases where $x < 0.05$ and $x > 0.95$ are ignored. Reasoning: The range considered around a test instance spans 10% of the range of the data, and the data is evenly distributed.

Consider the "nearest neighbour interval" to have side length of 10% of the range of the data
➔ there are 10 such non-overlapping intervals in [0,1]

1.2 On average, what fraction of the available observations will be used to make the prediction ($p = 2$)?

➔ On average, $0.1^2 = 0.01 = 1\%$



Consider the square to have side length of 10% of the range of the data
➔ there are 100 such non-overlapping squares in the interval [0,1]x[0,1]

1.3 What fraction of the available observations will be used to make the prediction ($p = 100$)?

➔ On average, $0.1^{100} = 10^{-100} = 10^{-98}\%$.

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.

➔ As p increases linearly, training observations that are geometrically near decrease exponentially.

1.5 For p = 1, 2, and 100, what is the length of each side of the hypercube?

➔ $p = 1 \Rightarrow l = 0.1$
➔ $p = 2 \Rightarrow l = \sqrt{0.1} = 0.32$

➔ $p = 3 \Rightarrow l = 0.1^{\frac{1}{3}} = 0.46$

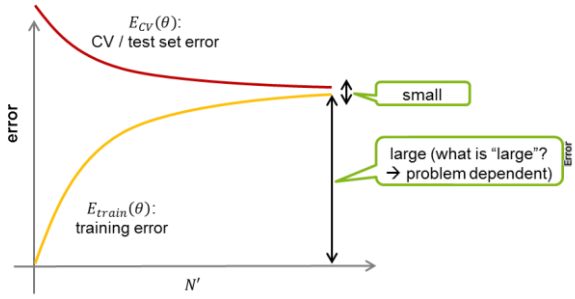
…

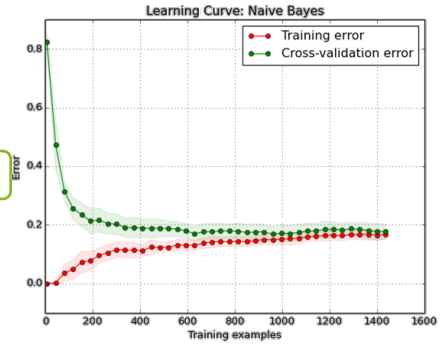
➔ $p = N \Rightarrow l = 0.1^{\frac{1}{N}}$

2. Applying learning curves

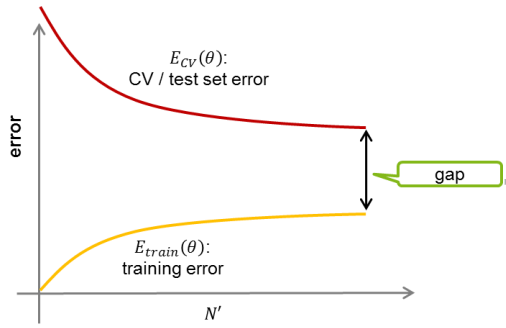
2.1 -

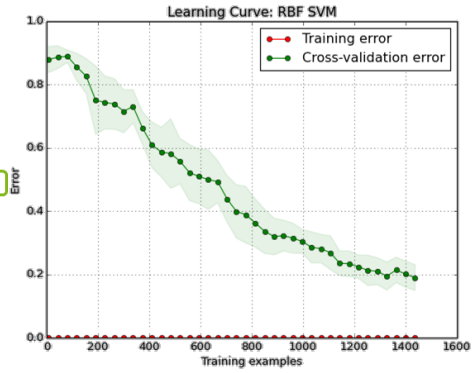
2.2 The digits data is a set of images of digits, so it is rather complex. When more analysis reveals what is possible on this dataset with more flexible learning algorithms, the ~17% error rate produced by Naïve Bayes are rather poor and are dominated by bias (compare the curve shape: both curves converge, but the error is rather high).



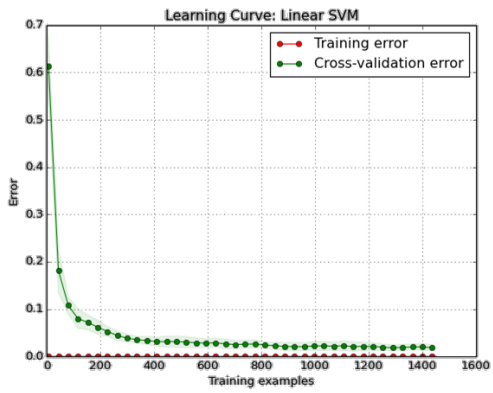


2.3 SVM with RBF kernel shows high variance symptoms and would maybe profit from more data:

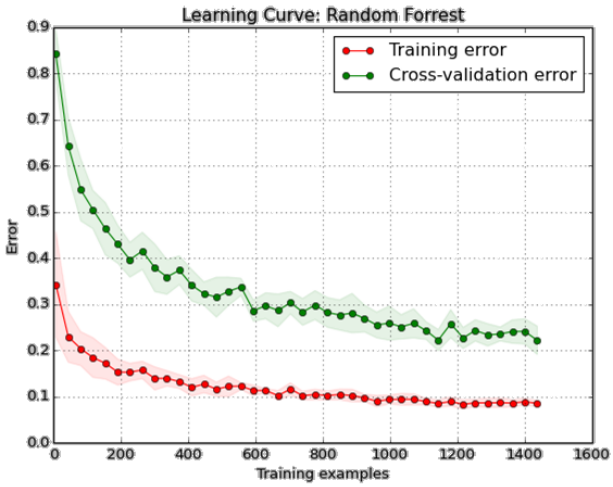
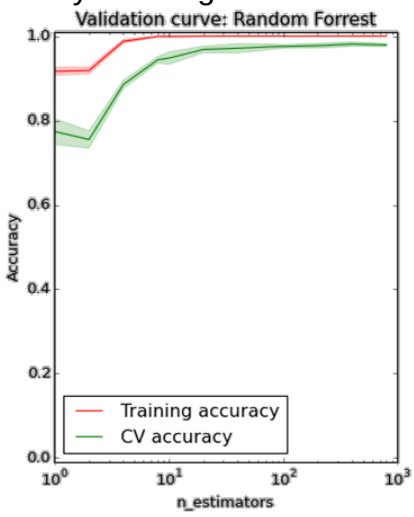
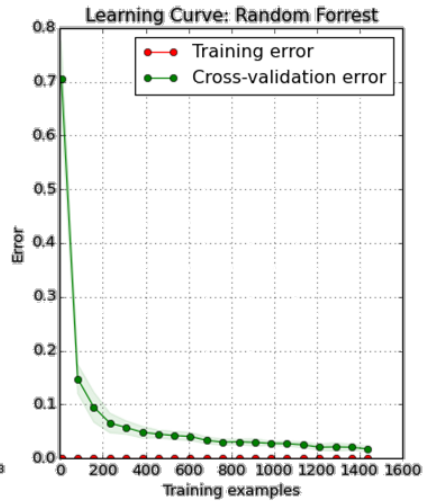




It seems easier to just try a less flexible model like SVM with linear kernel before collecting more data – this seems to work just fine (no distinguished bias or variance problem).



2.4 The parameter **`n_estimators`** determines the number of trees in the random forest. One tree shows great bias and variance…



… but with more than 10 trees in the forest the performance is very good and keeps slowly climbing: