# Solution for ML F16 assignment 2

Jonathan Hansen fdg890

December 5, 2016

## 1 K-Nearest Neighbors

I have implemented the $k$-nearest neighbor classifier in `Python3`. I have made substantial use of the libraries `Pandas`, `Numpy` and `MatPlotLib`. All the code is gathered in a single script called `irisknn.py`. Running it using "python3 irisknn.py" from the terminal produces the requested plots and reports required values.

### Question 1.1

With my implementation i get the results presented below. Notice that the training error for 1-NN is 0 as it should be. On the test data the classifier performs equally well/bad with $k$=1 and $k$=3, but the 5-NN performs substantially worse. This hints us that measuring size only is maybe not a sufficient method for classifying iris', if we want to get the error down below 0.18.

| Training and testresults of $k$-NN classifier for $k$ = 1,3,5 | | | |
|---|---|---|---|
| | $k = 1$ | $k = 3$ | $k = 5$ |
| Training error | 0.0 | 0.14 | 0.17 |
| Test error | 0.18 | 0.18 | 0.32 |

### Question 1.2

My cross-validation was done manually. I simply split the dataset into five partitions and hard coded the fold. Since the data does not precide in any particular order, this should not be a problem and it was the easiest way to keep track of the rows. Figure 1 shows the averaged cross-validation error as a function of the $k$ in the classifier. The classifier performs best with $k$=3 why the results are reported in the above table.

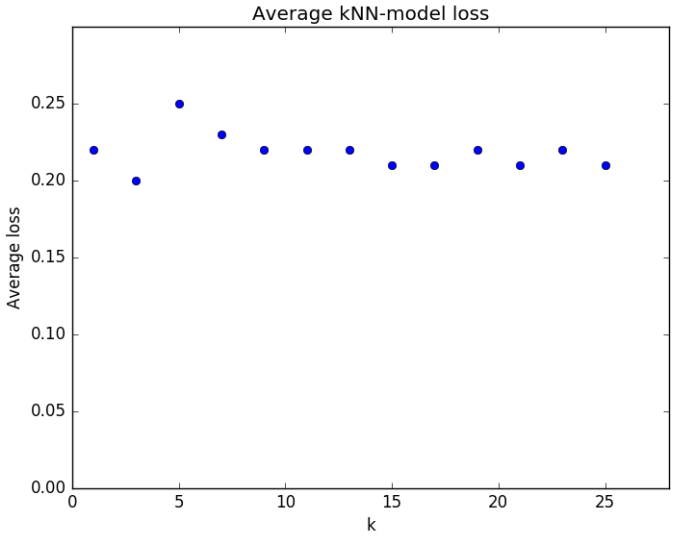


Figure 1: Error as a function of $k$ in k-NN classifier

**Question 1.3**

In the table below is the mean and variance of the input features and the mean and variance after data normalization. The normalized data is supposed to be zero-mean and it does come extremely close, but is not exactly zero. Variance is 1 as it should be. Using the affine linear mapping on the test data the sepal width is significantly further from being zero-mean and unit length.

| Mean and variance of data before and after transformation | | | | |
|---|---|---|---|---|
| | Length mean | Width mean | Legth var | Width var |
| Training data | 5.756 | 0.302 | 0.689 | 0.002 |
| Transformed training data | $\approx 0$ | $\approx 0$ | 1 | 1 |
| Transformed test data | 0.208 | 0.432 | 1.073 | 1.252 |

In Figure 2 is shown the averaged cross-validation error as a function of the $k$ in the classifier, this time run on the normalized date. Compared to the previous plot you can see, that errors on an average are lower, but also that they vary more. This time the classifier with $k$=11 performs better.

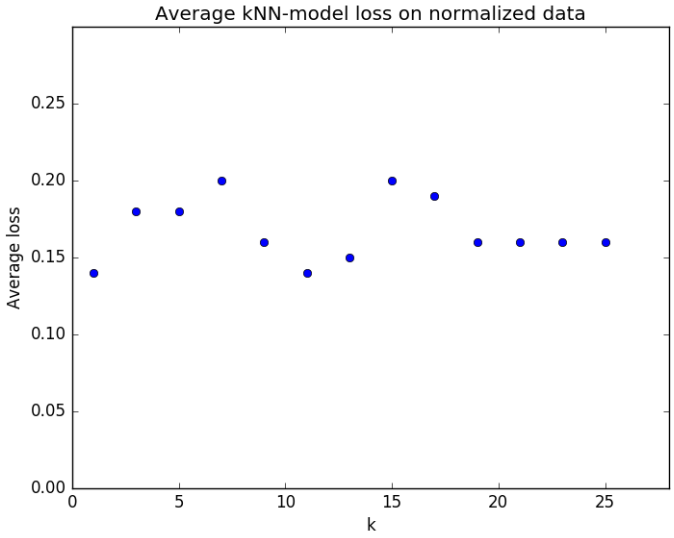


Figure 2: Error as a function of $k$ in k-NN classifier on normalized data

Finally the table below shows the training and test error of the 11NN classifier which the analysis pointed out as $k_{best}$. The table also shows the result of the initial 3NN classification of the raw data for comparison. It is interesting that 11NN classifier performs best, but only on training transformed training data. However we should remember, that the transformation of the test data was done with the affirm linear mapping that was found on the training data. This hints us that there might be a tendency in test data that is different than the training data. At the same time we have just two features to perform our analysis on and just 100 training data points.

| Error of initial 3NN and 11NN on normalized training and test data | |
|---|---|
| | Error |
| 3NN error, raw testdata | 0.18 |
| 11NN train error, transformed data | 0.14 |
| 11NN test error, transformed data | 0.26 |

## 2 Markovs inequality vs. Hoeffdings inequality vs. binomial bound

**Question 2.1**

To bound the probability that $\sum_{n=1}^{10} X_i$ we start by plugging the given values into Markovs inequality:

$$P\{S \geq 9\} \leq \frac{\mathbb{E}[S]}{9}$$

Using the random variable we are given and linearity of expectation we get:

$$P\{\sum_{n=1}^{10} X_i \geq 9\} \leq \sum_{n=1}^{10} \frac{\mathbb{E}[X_i]}{9}$$

Since we know that $X_1$,...,$X_{10}$ are i.i.d Bernoulli random variables with bias $\frac{1}{2}$ the expected value sums to $\frac{5}{9}$. Thus we get:

$$P\Big(\sum_{n=1}^{10} X_i \geq 9\Big) \leq \frac{5}{9}$$

**Question 2.2**

Focusing at first at the probability side of Hoeffding's inequeality we can use the information given, and the result of the sum of expected valus from above, to determine $\varepsilon$:

$$\mathbb{P}\left\{\sum_{n=1}^{10} X_i - \mathbb{E}\Big[\sum_{n=1}^{10} X_i\Big] \geq \varepsilon\right\} \Leftrightarrow \mathbb{P}\left\{\sum_{n=1}^{10} X_i \geq \varepsilon + 5\right\} \Rightarrow \varepsilon = 4$$

With this result we can find the bound we are asked for, since we know that $a_i = 0$ and $b_i = 1$:

$$P\Big(\sum_{n=1}^{10} X_i \geq 9\Big) \leq e^{-2(4)^2 / \sum_{n=1}^{10}(1-0)^2} = e^{-\frac{16}{5}}$$

**Question 2.3**

We can use the normal binomial PMF to calculate the exact probability of the event. For the sum of the 10 random variables to be equeal nine, we need success on 9 of them since they are Bernoulli random variables:

$$P\Big(\sum_{n=1}^{10} X_i = 9\Big) = \binom{10}{9} 0.5^9 (1 - 0.5)^{10-9} \approx 0.0098$$

**Question 2.4**

Comparing the three results the most interesting thing to note is that Hoeffdinger's inequality produces a much tighter bound than Markov's inequality. From Markov's you can get the (mis)understanding that $\sum_{n=1}^{10} X_i \geq 9$ is a pretty likely event, which is not the case. This inprecision could be much more dangorous in a sitaution where our understanding of the distribution or expectations is a lot less intuitive than it is in this case.

# 3 The effect of scale (range) and normalization of random variables in Hoeffdings inequality

We begin by substituting $\varepsilon$ with $n\varepsilon$ in `Theorem 1.2`:

$$\mathbb{P}\left\{\sum_{i=1}^{n} X_i - \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] \geq n\varepsilon\right\} \leq e^{-2(n\varepsilon)^2 / \sum_{i=1}^{n}(b_i - a_i)^2}$$

Looking first at the righthand side of the inequality, in the corollary that we have to prove $X_i$ can only assume values 0 and 1, why we know that:

$$\sum_{i=1}^{n}(b_i - a_i)^2 = n$$

And therefor the entire righthand side of the inequality can be reduced to:

$$e^{-2n\varepsilon^2}$$

Looking at the lefthand side we can dividide by $n$ on both sides of the inequality giving us:

$$\mathbb{P}\left\{\frac{1}{n}\sum_{i=1}^{n} X_i - \frac{1}{n}\mathbb{E}\left[\sum_{i=1}^{n} X_i\right] \geq \frac{1}{n}n\varepsilon\right\}$$

By linearity of expectation and the assumption of `Corollorary 1.4` we substitute $\mu$ instead of the expected value. Combining everything so far we get the final result:

$$\mathbb{P}\left\{\frac{1}{n}\sum_{i=1}^{n} X_i - \mu \geq \varepsilon\right\} \leq e^{-2n\varepsilon^2}$$

# 4 Basic Statistics

I did not have time to finish this task.

# 5 Linear regression

I have implemented the linear regression algorithm in `Python3`. I have made substantial use of the libraries `Pandas`, `Numpy` and `MatPlotLib`. All the code is gathered in a single script called `regression.py`. Running it using "python3 regression.py" from the terminal produces the requested plots and reports required values.

### Question 5.1.1

Look in the script to see how the regression algorithm has been implemented. I was a bit in doubt on whether to report $w$ as a row or coloumn vector. I have tried implementing the model so it works with data of all dimensions. Notice however that $N$ must be larger than the number of features for $X$.

### Question 5.1.2

Below is the parameters of the affine linear model build with the implemented linear regression algorithm. Notice that the functions used in this subsection are not designed to handle X-data of more than 1 dimension.

| Parameters of affine linear model and mean-squared-error | |
|---|---|
| Vector w | 9.48934569 |
| b | -10.42696146 |
| mean-squared-error | 0.01243422 |

### Question 5.1.3

Figure 3 shows a plot of the data and the regression line resulting from putting the data through the regression algorithm I have implemented.

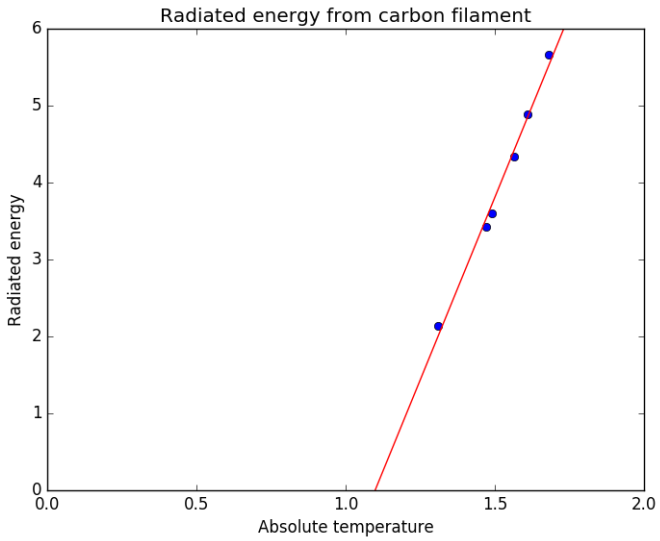


Figure 3: Plot showing the DanWood.dt data and the resulting regression line from the build model

### Question 5.2

I did not have time to finish this task.