Machine Learning - Assignment 2

Christoffer Thrysøe - dfv107

January 8, 2017

1. K-Nearest Neighbours

The function `main()`, found in the file `KNN.py` can be run to reproduce the results for this assignment.

Question: 1.1

The implemented K-Nearest Neighbour method is located in the file `KNN.py`. The method works by calculating the euclidean distance, for each input point, to the rest of the points in the data. These distances are then sorted in ascending order and the top k distances are retrieved. Each of these points have a target value, which counts as votes toward a prediction. The target value with the most votes is returned as the predictive class.
To measure the performance of the classifier, the zero-one loss function is used, resulting in 0 if the predicted value Y' is equal to the true target value Y and 1 otherwise:

$$\ell(Y', Y) = \mathbb{1}\{Y' = Y\} = \begin{cases} 0, & \textit{if } Y' = Y \\ 1, & \textit{otherwise} \end{cases}$$

The loss will be calculated in respect to the amount of prediction made N, that is the average loss. The empirical loss is defined as followed:

$$\hat{L}(h) = \frac{1}{N}\sum_{n=1}^{N} \ell(h(x_n), y_n)$$

The implemented method is applied to both the training data `IrisTrainML.dt` and the testing data: `IrisTestML.dt`, each with values of $k \in \{1, 3, 5\}$.
The below table shows the results of running the aforementioned experiments.

k	Test	Train
1	0.18	0.0
3	0.18	0.14
5	0.31	0.17

Table 1: Empirical loss when running the KNN classifier on the testing and training set for different values of k

Note from the above table, that the empirical loss of $k = 1$ on the training data, is always zero as the closest neighbour of a given point is always the point itself. The test data is treated as unknown values in the training set, therefore the empirical loss of $k = 1$ is not 0, as the entry doesn't exist in the data set.

Question 1.2

Cross validation is used for the K-Nearest Neighbour algorithm, in order to find the best suited value of k. The applied validation is a *5-fold cross validation*, which means that the dataset is split into 5 equal sized partitions, where for a fixed k, one of these partitions is used as a validation set. This is performed for each fold, thus 5 times. The average loss of these five folds is used to estimate the performance of the classifier. These folds are performed for $k \in \{1, 3, 5, ..., 25\}$ in order to find the value k, which results in the lowest average loss, this choosing of k will be denoted as k_{best}. The 5-fold validation is implemented in the function called `crossValidate` located in the file `KNN.py`. The method works by creating a random permutation of the input data, this permutation is then split into 5 equally sized partitions. The validation algorithm is then run for each value of k and each partition.

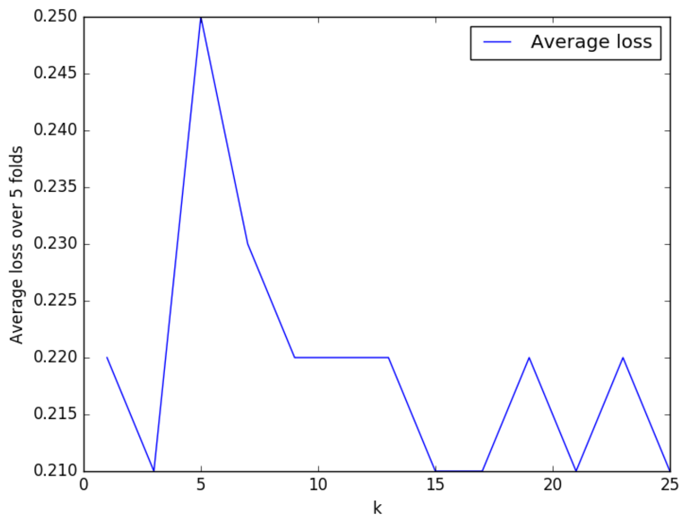

Figure 1: Average loss over 5 folds as a function of the values of k

The value of k with the lowest 0-1 error is $k = 3 = k_{best}$ The generalization performance of k_{best} is found evaluating the 3-KNN classifier on the test data. The general performance of 3-KNN is 0.18.
The result of the cross validation was somewhat surprising as i would have thought that the larger the k, the bigger the loss. However as evident in figure 1 there is no direct connection between the size of k and the loss. Also when running the cross validation multiple times, with different random partitions, the value of k_{best} would change. $k = 3$, has been chosen as k_{best} as it more often than not, provides the lowest cost, when performing cross validation. Table 2 shows the empirical loss from running the classifier on the training and testing data sets, with $k = k_{best} = 3$:

k	*Test*	*Train*
3	0.18	0.14

Table 2: Empirical loss when running the KNN classifier on the testing and training set with $k = 3$

Question 1.3

The objective of the normalization procedure is to perform a zero-mean, unit variance transformation on the input data. To do so, we apply a linear mapping from the input data to the normalized output data. Thus we wish to determine some t:

$$X_{norm} = t(X) \tag{1}$$

such that $\mu(X_{norm}) = 0$ and $Var(X_{norm}) = 1$
To achieve this, the mean of each input feature is found and subtracted. The mean is calculated by:

$$\mu(X) = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Next the variance is found of each feature. The variance is calculated by:

$$Var(X) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

The standard deviation σ is equal to the square root of the variance and is used to normalize the data. The normalization can then be expressed as the following:

$$X_{norm} = \frac{X - \mu(X)}{\sigma} \tag{2}$$

Thus defining the linear mapping from X to X_{norm}. Below is the mean and variance for the training data `IrisTrainML.dt` and the normalized test data `IrisTestML.dt`. The mean and variance of the training data is used to normalize the test data, using equation (2)

	Training	$Test_{norm}$
Mean	$(5.75, 0.3)$	$(0.20, 0.43)$
Variance	$(0.6888, 0.0017)$	$(1.07, 1.25)$

Table 3: Left: Mean and Variance calculated on the training data. Right: The Mean and Variance of the data which has been normalized using the values for mean and variance in the left column.

The variance and mean for the normalized test data is slightly larger than of the training data.
Running the cross validation algorithm on normalized data yields the following results:

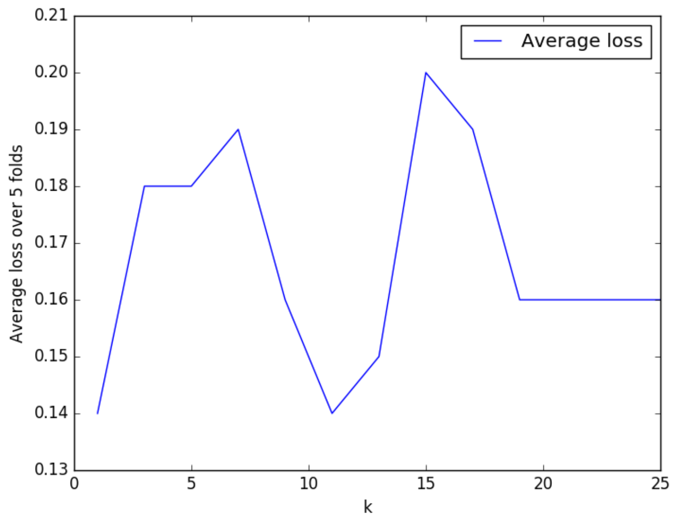

Figure 2: Average loss over 5 folds as a function of the values of k, found using the normalized data.

The above results suggests a $k_{best} = 1 \; or \; 11$, as in the cross validation for non-normalized data the the result is contra distinctive. A clear difference from the results of running the cross validation on non-normalized data and on normalized data is that the normalized data produce a much smaller loss.

k	*Test*	*Train*
1	0.18	0.0
11	0.42	0.23

Table 4: Empirical loss when running the KNN classifier on the testing and training set for different values of k

This feature normalization is performed to ensure that each feature counts equally. Features are measured on different scales, for example some features may range big values where others small. To ensure that features count equally they are scaled to lie in the same range. This is essential when dependent on distance metrics as non-normalized data can create a huge bias toward one feature. I believe the improved results of the cross validation, performed on normalized data, is due to the normalization of the data, and the removal of potential bias in the data.

2 Markov's inequality vs. Hoeffding's inequality vs. binomial bound

1

To bound the probability using markov's inequality, the expected value of S is determined:

$$\mathbb{E}[S] = \mathbb{E}\left[\sum_{i=1}^{10} X_i\right] = \sum_{i=1}^{10} \mathbb{E}[X_i]$$

The expected value of the Bernoulli random variable with bias $= \frac{1}{2}$ is defined as followed:

$$\mathbb{E}[X] = Pr[X=1] \times 1 + Pr[X=0] \times 0 = \frac{1}{2}$$

the expected value of S is:

$$\mathbb{E}[S] = \left[\sum_{i=1}^{10} X_i\right] = 10 \times \frac{1}{2} = 5$$

therefore the inequality can be written as followed:

$$Pr[S \geq 9] \leq \frac{\mathbb{E}[S]}{9} = \frac{5}{9} \approx 0.55$$

2

Given the independence of the variables and that $\mathbb{E}[X_i] = \mu$ Hoeffding's inequality can be defined as:

$$Pr\left[\frac{1}{n}\sum_{i=1}^{n} X_i - \mu \geq \epsilon\right] \leq e^{-2n\epsilon^2}$$

which can be rewritten as:

$$Pr\left[\sum_{i=1}^{n} X_i \geq n(\mu + \epsilon)\right] \leq e^{-2n\epsilon^2}$$

noting that, $n = 10$, $\mu = \dfrac{1}{2}$ solving for ϵ in

$$10\left(\epsilon + \frac{1}{2}\right) = 9, \textit{ yields: } \epsilon = \frac{2}{5}$$

the value of epsilon can now be input into the above inequality:

$$Pr\left[\sum_{i=1}^{n} X_i \geq 9\right] \leq e^{-2\times 10 \times \frac{2}{5}^2} = e^{-\frac{16}{5}} \approx 0.04$$

3

The probability of $S \geq 9$ is equal to the probability of $Pr[S = 9] + Pr[S = 10]$ as the events are independent:

$$Pr[S = 9] = \binom{10}{9} \times \frac{1}{2}^9 \times \left(1 - \frac{1}{2}\right)^1 = \frac{10}{1024}$$
$$Pr[S = 10] = \binom{10}{10} \times \frac{1}{2}^{10} \times \left(1 - \frac{1}{2}\right)^0 = \frac{1}{1024}$$

and therefore the probability is:

$$Pr[S \geq 9] = \frac{10}{1024} + \frac{1}{1024} = \frac{11}{1024} \approx 0.0107$$

4

The bounded probability using Markov's inequality is very loose compared to the bounded probability by Hoeffding's inequality. The bounded probability using Markov's inequality is about 50 times greater than the exact probability where the bounden probability by Hoeffding's inequality is about 13 times greater than the exact.

3. The effect of scale (range) and normalization of random variables in Hoeffding's inequality

Corollary 1.4 assumes $X_1, ..., X_n$ are independent, such that $Pr[X_i \in [0,1]] = 1$ and that $\mathbb{E}[X_i] = \mu$. Given the above, we can apply Theorem 1.2 , that is we can express the term as followed $\sum_{i=1}^n (b_i - a_i)^2 = \sum_{i=1}^n (1-0)^2 = \sum_{i=1}^n 1 = n$. We let $\epsilon' = n\epsilon$ in theorem 1.2 and make use of the above, therefore:

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

Because of independence we can define: $\mathbb{E}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \mathbb{E}[X_i] = \sum_{i=1}^n \mu = \mu n$ putting the expected value into the inequality:

$$Pr\left[\sum_{i=1}^n X_i - \mu n \geq \epsilon'\right] \leq e^{-2\epsilon'^2/n}$$

If we now replace ϵ' with $n\epsilon$, and multiply $\frac{1}{2}$ to get rid of n in the inner term:

$$Pr\left[\sum_{i=1}^n X_i - \mu n \geq n\epsilon'\right] = Pr\left[\frac{1}{n}\sum_{i=1}^n X_i - \mu \geq \epsilon\right] \leq e^{-2n\epsilon^2}$$

thus completing the proof that corollary 1.4 follows from theorem 1.2 for random variables in the $[0,1]$ interval.

4. Basic Statistics

1

To bound the probability, i will use Hoeffding's inequality. First we note that for each passenger there is a $\mu = 1 - 0.05$ change of him/her not showing up. We also note that $n(\mu + \epsilon) = 100$ and $n = 100$. As before we solve for ϵ:

$$100(\epsilon + \frac{95}{100}) = 100, \textit{ yields: } \epsilon = \frac{1}{20}$$

The value for ϵ can then be inserted into the inequality:

$$Pr\left[\sum_{i=1}^n X_i \geq 100\right] \leq e^{-2\times 100 \times \frac{1}{20}^2} = e^{-\frac{1}{2}} \approx 0.60$$

2

5. Linear Regression

The function `main()`, found in the file `linregs.py` can be run to produce the results for this assignment.

Question 5.1

1

The implemented linear regression algorithm can be found in the file `linregs.py`.
First the data is loaded from the file `DanWood.dt`, the data is then prepared, that is split into x and y values. The intercept is added for the $\mathbf{X}$, setting x_0 to 1. The weights w are then calculated as $w = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$. The predictions $\hat{y}$ are estimated as $\hat{y} = \mathbf{X}w$.

2

The linear model for the given data is as followed:

$$\hat{y} = 9.489 \times x - 10.426$$

thus the weights are: $w = [-10.426, 9.489]$
The mean squared error is defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

The RMSE of the linear model performed on the data is 0.0124

3

The below figure shows the data points and the linear model as a function of x.

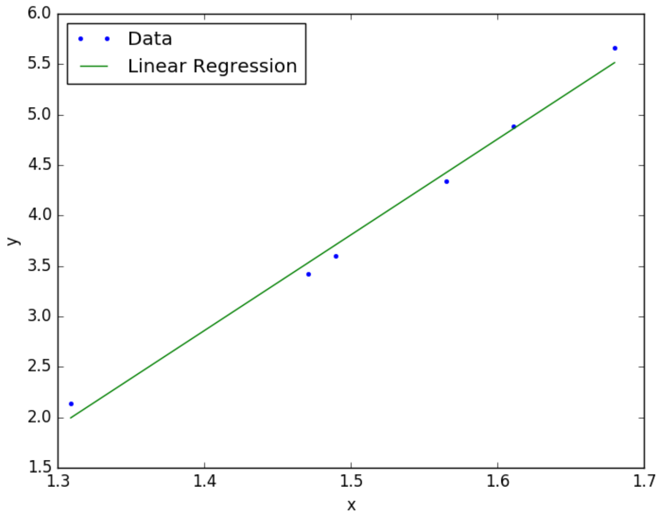

Figure 3: Data plotted as points, the linear regression model plotted as a line

Question 5.2

Proof that $\mathbf{H} = \mathbf{H}^T$, where $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$. A rule used throughout the proof regards the transpose of multiple values and it goes as followed: $(\mathbf{ABC})^T = \mathbf{C}^T\mathbf{B}^T\mathbf{A}^T$, i will refer to this rule as the *transpose rule*. Another rule, which i will refer to as the inverse rule goes as followed: $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$

$$\begin{aligned}
\mathbf{H} &= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \\
\mathbf{H}^T &= \left[\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\right]^T && \textit{Apply transpose rule to the three inner terms} \\
&= \mathbf{X}\left((\mathbf{X}^T\mathbf{X})^{-1}\right)^T\mathbf{X}^T && \textit{Apply the inverse rule} \\
&= \mathbf{X}\left((\mathbf{X}^T\mathbf{X})^T\right)^{-1}\mathbf{X}^T && \textit{Apply transpose rule to inner term} \\
&= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T \\
&= \mathbf{H}
\end{aligned}$$

Thus concluding the proof of $\mathbf{H} = \mathbf{H}^T$
To prove that $\mathbf{H}^k = \mathbf{H}$ i will prove that the matrix is idempotent meaning that $\mathbf{H} \times \mathbf{H} = \mathbf{H}$:

$$\begin{aligned}
\mathbf{HH} &= (\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T) \times (\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T) && (3)\\
&= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{X})(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T && (4)\\
&= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{I}\mathbf{X}^T && (5)\\
&= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T && (6)\\
&= \mathbf{H} && (7)
\end{aligned}$$

where step (4) of the proof uses the property that $(\mathbf{X}^T\mathbf{X})(\mathbf{X}^T\mathbf{X})^{-1} = \mathbf{I}$
Thus concluding the proof of $\mathbf{H} \times \mathbf{H} = \mathbf{H}$, note that this also imply that $\mathbf{H}^k = \mathbf{H}$.