Machine Learning
Assignment 1.1 - Classification and Concentration Bounds

Tobias Hallundbæk Petersen - xtv657

November 23, 2015

1 Classification

1.1 Nearest neighbor

I have used NumPy for this implementation as it greatly improves performance and generally makes your code neater and more readable. The following is my source code for my nearest neighbor classifier:

```
def knn(k, trainSetData, trainSetClass, target, distFun):
  dist = distFun(trainSetData, target)
  res = np.argsort(dist, axis=0)[0:k]
  counts = np.bincount(trainSetClass[res])
  return np.argmax(counts)
```

The implementation is very general and allows any given distance metric to be used, this function given to compute the distance, should be able to return a list of distances given the training set data and a target. For the next line, we find the indices of the k shortest distances, the next line counts the different classifications and then returns the maximum of these. If the program is run with the parameter `1.1` we get the following output:

```
Error of the 1-nn classifier: 0.184210526316
Error of the 3-nn classifier: 0.184210526316
Error of the 5-nn classifier: 0.315789473684
```

We can see both the 1-nn and 3-nn perform equally well, while the 5-nn has a quite much larger error, due to the small size of our training set this makes sense.

1.2 Hyperparameter selection using cross-validation

For the cross validation I have partitioned the training set into 5 sets, and run k-nn for k from $\{1, 3, 5, ..., 25\}$ for the five tests, these tests and their division can be seen in Figure 1.

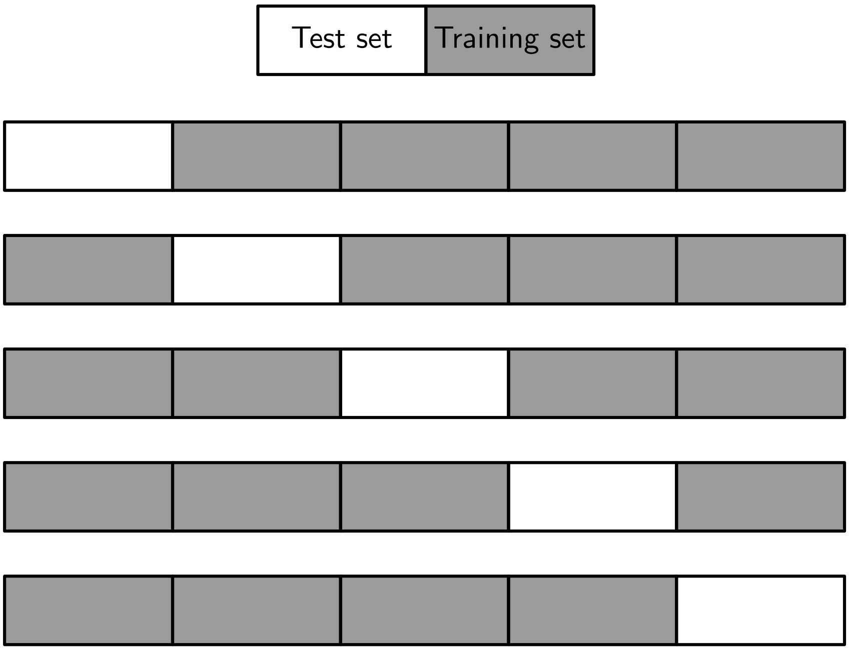

Figure 1: Division of the training set for use in cross-validation

For this I implemented the more general n-fold cross-validation. This was implemented in the following way.

```python
def nFoldCrossValidation(n, k, trainSetData, trainSetClass):
  dataSplit = np.split(trainSetData,n)
  classSplit = np.split(trainSetClass,n)
  errTotal = 0.0
  for i in range(0,n):
    trData = np.concatenate(dataSplit[:i] + dataSplit[(i + 1):])
    trClass = np.concatenate(classSplit[:i] + classSplit[(i + 1):])
    teData = dataSplit[i]
    teClass = classSplit[i]
    err = 0.0
    for j in range(0,len(teClass)):
      if knn(k, trData, trClass, teData[j], euclideanNoSqrt) != teClass[j]:
        err += 1.0
    errTotal += err / len(teClass)
  return errTotal/n
```

The data is split and then the cross validation is performed, it returns the average error over the n different runs.
To get the results for k from $\{1, 3, 5, ..., 25\}$ the program can be run with the `1.2` parameter, which yields the following response:

```
Crossvalidation error for the k-nn classifier for k=[1,3 ... 25]
1-nn: 0.22
3-nn: 0.21
5-nn: 0.25
7-nn: 0.23
9-nn: 0.22
11-nn: 0.22
13-nn: 0.22
15-nn: 0.21
17-nn: 0.21
19-nn: 0.22
21-nn: 0.21
23-nn: 0.22
25-nn: 0.21

Best k found was 3 with an error of 0.21
```

And we see that the program suggests setting $k = 3$, ties are in my program won by the first to get the best error.

1.3 Data Normalization

We want to normalize our training data and get a function f_{norm} that when applied on the training data will conform the training data to have a 0 mean, 1 variance features. This function will then be used

on the test set as well, this will not ensure that the test set will conform to the same restrictions as our training set but it should, given a large enough sample be close.
To normalize the data, the mean of the data is subtracted from the data, and the data is afterwards divided by its standard deviation, this will ensure that we get 0 mean, 1 variance features.
To get the results required in the assignment, the program can be run the `1.3` parameter, giving the following response:

```
Original training data:
First feature, mean: 5.756, variance: 0.688864
Second feature, mean: 0.3017, variance: 0.00174211

Normalized test data:
First feature, mean: 0.208375768039, variance: 1.07339795338
Second feature, mean: 0.432138257729, variance: 1.25222270424

Crossvalidation error for the k-nn classifier for k=[1,3 ... 25] on normalized test data
1-nn: 0.14
3-nn: 0.18
5-nn: 0.18
7-nn: 0.19
9-nn: 0.16
11-nn: 0.14
13-nn: 0.15
15-nn: 0.2
17-nn: 0.19
19-nn: 0.16
21-nn: 0.16
23-nn: 0.16
25-nn: 0.16

Best k found was 1 with an error of 0.14
Error of the 1-nn classifier on normalized data: 0.214736842105
```

We can see that the normalization of the test data is not strictly 0 mean, 1 variance, but its a good way there. It is a bit disappointing that the error after all this normalization is worse than the one before normalization, but I would suppose that given a larger sample the normalization could improve upon the k-nn algorithm.

2 Probability theory refreshment

We have an urn that contains five red, three orange, and one blue ball. We now select two balls at random

1. **What is the sample space of this experiment**
   $\{\{R,R\},\{R,O\},\{R,B\},\{O,O\},\{O,R\},\{O,B\},\{B,R\},\{B,O\}\}$

2. **What is the probability of each point in the sample space**
   If we use the same ordering as above, $\{5/18, 5/24, 5/72, 1/12, 5/24, 1/24, 5/72, 1/24\}$, the sum of these probabilities sum up to one, meaning that we have a complete sample space.

3. **Let X represent the number of orange balls selected. What are the possible values of X?**
   The possible values of X are $\{0, 1, 2\}$.

4. **Calculate $\mathbb{P}\{X = 0\}$**
   $5/18 + 5/72 + 5/72 = 5/12$

5. **Calculate $\mathbb{E}[X]$**
   $5/24 + 1/12 \cdot 2 + 5/24 + 1/24 + 1/24 = 2/3$

3 Probability theory refreshment

From probability theory we have the following definitions and properties:

$$(a)\ p_X(x) = \sum_{y \in \mathcal{Y}} p_{XY}(x,y)$$

$$(b)\ \text{If } X \text{ and } Y \text{ are independent, then } P_{XY}(x,y) = p_X(x)p_Y(y)$$

$$(c)\ \mathbb{E}[X] = \sum_{x \in \mathcal{X}} x p_X(x)$$

X and Y are discrete random variables that take values from in $\mathcal{X}$ and $\mathcal{Y}$. p_X is the distribution of X, p_Y the distribution of Y and p_{XY} the distribution of X and Y.

1.

We prove the following identity:

$$\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$$

By (c), the expected value of X is given by:

$$\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x p_X(x)$$

Therefore the expected value of $X+Y$ would be

$$\begin{aligned}
\mathbb{E}[X+Y] &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} (x+y) p_{XY}(x,y) \\
&= \sum_{x \in \mathcal{X}} x \sum_{y \in \mathcal{Y}} p_{XY}(x,y) + \sum_{y \in \mathcal{Y}} y \sum_{\in \mathcal{X}} p_{XY}(x,y) \\
&= \sum_{x \in \mathcal{X}} x p(x) + \sum_{y \in \mathcal{Y}} y p(y) \\
&= \mathbb{E}[X] + \mathbb{E}[Y]
\end{aligned}$$

In the last step we use the definition for the expected value of a random variable. We have now shown that $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$.

2.

To prove the following identity, we use that the random variables X and Y are independent.

$$\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$$

We can write $\mathbb{E}[XY]$ as

$$\mathbb{E}[XY] = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} xy p_{XY}(x,y)$$

This is where we use that X and Y are independent - using property (b):

$$\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} xy p_X(x) p_Y(y)$$

This can be reduced to prove our identity

$$\begin{aligned}
\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} xy p_X(x) p_Y(y) &= \sum_{x \in \mathcal{X}} x p_X(x) \sum_{y \in \mathcal{Y}} y p_Y(y) \\
&= \mathbb{E}[X]\mathbb{E}[Y]
\end{aligned}$$

This proves the identity $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.

3.

A bag has 2 red apples and 2 green apples. There is taken 2 apples from the bag without putting them back into the bag. Let X be the first apple and let Y be the second apple. The joint distribution table of X and Y is seen below:

X / Y	Red	Green
Red	$\frac{1}{6}$	$\frac{2}{6}$
Green	$\frac{2}{6}$	$\frac{1}{6}$

The probability of apple X being red is:

$$\mathbb{E}[X = \text{Red}] = \frac{1}{2}$$

Which is the same probability for apple Y being red. We have that

$$\mathbb{E}[X = \text{Red} \wedge Y = \text{Red}] = \frac{1}{6}$$

Since $\frac{1}{2}\frac{1}{2} = \frac{1}{4} \neq \frac{1}{6}$ then

$$\mathbb{E}[XY] \neq \mathbb{E}[X]\mathbb{E}[Y]$$

in this example.

4.

The identity to be proved:

$$\mathbb{E}[\mathbb{E}[X]] = \mathbb{E}[X]$$

We know that $\mathbb{E}[X] = k$ and that $\mathbb{E}[k] = k$. That means taking the expected value of an expected value will just return the constant you already found. This can be done more than 2 times and it will always be the constant k that is your result.

5.

We want to show that $\mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$.

$$\begin{aligned}\mathbb{E}[(X - \mathbb{E}[X])^2] &= \mathbb{E}[X^2 - 2 \cdot X \cdot \mathbb{E}[X] + (\mathbb{E}[X])^2] \\ &= \mathbb{E}[X^2] - 2 \cdot \mathbb{E}[X] \cdot \mathbb{E}[X] + (\mathbb{E}[X])^2 \\ &= \mathbb{E}[X^2] - (\mathbb{E}[X])^2\end{aligned}$$

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

We let $X_1, ..., X_{10}$ be i.i.d. Bernolli random variables with bias 1/2.

1. We want to use Markov's inequality to bound the probability that $\sum_{i=1}^{10} \geq 9$. We here define a new random variable $S = \sum_{i=1}^{10} X_i$ and we have that Markov's inequality states that:

   $$\mathbb{P}(S \geq 9) \leq \frac{\mathbb{E}(S)}{9}.$$

   We now have to find the expected value of S which is $\mathbb{E}[S] = 5$ giving us a bound by Markov's inequality of $\mathbb{P}[\sum_{i=1}^{10} \geq 9] \leq 5/9$

2. We now want to use Hoeffding's inequality to bound the probability of that same event.

   $$\mathbb{P}[S - \mathbb{E}[S] \geq 3] \leq e^{-2 \cdot 3^2/10} = e^{-18/10} = 0.1653$$

3. We now want to calculate the exact probability of the event.

   $$\mathbb{P}[S \geq 9] = \frac{\binom{10}{9}}{2^{10}} + \frac{1}{2^{10}} = \frac{10}{2^{10}} + \frac{1}{2^{10}} = 0.01074$$

4. For comparison we can see that Hoeffding's inequality gets closer to the exact probability compared to Markov's inequality, but neither of them give a specially tight bound.

5 Hoeffding's inequality

1. If there is 5% probability that a person does not show up to their flight, on a 99 seat plane we want to bound the probability that it will be overbooked selling 100 tickets.
   Using Hoeffding's inequality we can bound this probability to:

$$\begin{aligned}
\mathbb{P}\{\sum_{i=1}^{n} X_i - 95 > 4\} &\leq e^{-2 \cdot 4^2 / \sum_{i=1}^{n} (1-0)^2} \\
&= e^{-2 \cdot 4^2 / 100} \\
&= e^{-8/25} \\
&\approx 0.7261
\end{aligned}$$