# k-Nearest Neighbour and Naive Bayes

Andreas Dahlberg

November 3, 2018

## Exercise 1

### Exercise 1b)

The $k$ that seems to be optimal might only be the best $k$ for this particular test dataset. It is not certain that this dataset generalizes to other data sets. In order to determine the best $k$ for a more "general" data set we need to do a cross-validation. This can be done by dividing all of our data into a set TRAIN and a set TEST. Furthermore, we divide the TRAIN set into $k$ smaller sets. For the $k$ smaller sets we train on $k-1$ sets and choose the best parameters by testing on the $k$:th set. This is done $k$ times and we let the optimal parameters be the average of the optimal parameters found in each step. The reason this procedure is better than just choosing the optimal parameters from one set is that it removes any biases we might have. When we have chosen the best parameters, we use the TEST set to see how well our algorithm performs with these parameters.

### Exercise 1c)

We store all our training data in a $n \times d$ matrix where $n$ is the size of the training set and $d$ is the length of the vectors. The space complexity is therefore $O(nd)$. Since there is no training (kNN is lazy algorithm, all the work happens once we classify a point) the time complexity of training is $O(1)$.

### Exercise 1d)

To predict the label of a test point we need to calculate the distances to all points in the training set. This takes $O(nd)$. Then we need to sort these $n$ points, which is $O(nlogn)$. Finally, among the $k$ nearest neighbours we should find the class that occurs most frequently. If we only have two classes then this can be done by simply looping over these elements and counting the number of occurencies of both classes and then selecting the one that occurs the most. This takes $O(k)$. If we have $c$ classes then this step can be done in $O(k+c)$ by looping over the k elements and storing the frequency of each class in a hashmap and then looping over the hashmap and taking the most common element. So with only two classes the time complexity is $O(nd + nlogn + k) = O(nd + nlogn)$ and with $c$ classes the time complexity is $O(nd + nlogn + k + c) = O(nd + nlogn)$. This means that the time complexity isn't affected by the number of classes!

### Exercise 1e)

Suppose $k = 1$, then we want to find the closest point. In order for the word "closest" to make sense our function $d$ should satisfy: $d(x, y) \geq 0$ and 0 if and only if $x = y$ and also $d(x, y) = d(y, x)$, i.e it should be a semi- metric. For such a function we can perform the kNN algorithm. In particular, it doesn't have to fulfill the triangle inequality. However, if $d$ satisfies the triangle inequality then we can use this to speed up the computations in the sense that we do not have to calculate the distance to all the points. Since the Manhattan distance is a semi-metric (and also a metric) then we can use this in the kNN algorithm.

### Exercise 1f)

When selecting the class that occurs most frequently among the k-nearest neighbours, there will probably almost always be a tie if there are uncountably many classes (since then two neighbours are seldom in the same class). A workaround to this problem is to simply take the average of the k-nearest neighbour and assign this class/label to the test point.

## Exercise 2

### Exercise 2a)

We will let $A$ denote the event that $clump = 5$, $B$ the event that $uniformity = 2$, $C$ the event that $marginal = 3$ and $D$ the event that $mitoses = 1$. Our goal is to calculate $P(benignant|A, B, C, D)$ and $P(malignant|A, B, C, D)$ and then assign the point to the class that has the highest probability. Using Bayes theorem we have

$$\begin{aligned} P(benignant|A, B, C, D) \sim & P(A|benignant)P(B|benignant)P(C|benignant)P(D|benignant)P(benignant) \\ & = 0.18 \cdot 0.08 \cdot 0.07 \cdot 0.97 \cdot 0.66 \approx 0.000645. \end{aligned}$$

Where the corresponding probabilities are obtained by looking at the output file. Note that we skip the denominator since it only changes the two probabilities with a constant factor. Similarly,

$$\begin{aligned} P(malignant|A, B, C, D) \sim & P(A|malignant)P(B|malignant)P(C|malignant)P(D|malignant)P(malignant) \\ & = 0.19 \cdot 0.04 \cdot 0.11 \cdot 0.57 \cdot 0.337 \approx 0.00016. \end{aligned}$$

Since the data point is more likely to be *benignant* than *malignant* we assign it to the class *benignant*.

### Exercise 2b)

In the code, what happens when a missing value occurs is that we simply ignore it and continue with the execution as normal. If we have a lot of data and only a few missing values then this doesn't affect the probabilities in the end that much. Of course, if we have some missing values then we get slightly less accurate results than if we had all the data. For instance, if we have 10 rows of data and 4 missing values in total, this could be interpreted as if we only have 9 rows of data where no row have any missing values.

### Exercise 2c)

Let's consider the case $P(clump = 3|benignant)$. To calculate this probability we should find all entries with $clump = 3$ in the data corresponding to *benignant* and then divide by the total number of *clump* entries in *benignant*. We let $clump(3, benignant)$ be the total number of entries that have $clump = 3$ in *benignant* and $clump(benignant)$ the total number of *clump* entries in *benignant*. We can first note that $\sum_{j=1}^{10} P(clump = j|benignant) = 1$ since *clump* can only take on integer values from 1 to 10. In order to solve the "zero-frequency-problem" we can simply define $P(clump = j|benignant)$ to be $(clump(j, benignant) + 1)/clump(benignant)$ instead of $clump(j, benignant)/clump(benignant)$. However, then we have that $\sum_{j=1}^{10} P(clump = j|benignant) = 1 + 10/clump(benignant)$ which is not 1. Instead, if we define $P(clump = j|benignant)$ to be $(clump(j, benignant) + 1)/(clump(benignant) + 10)$ then these probabilities are never 0 and they sum up to 1. This is known as Laplace smoothing.

# Exercise 3

## Exercise 3a)

Using Bayes formula and law of total probability we get:

$$P(bowl\ 1|vanilla) = \frac{P(vanilla|bowl\ 1)P(bowl\ 1)}{P(vanilla)} = \frac{P(vanilla|bowl\ 1)P(bowl\ 1)}{P(vanilla|bowl\ 1)P(bowl\ 1) + P(vanilla|bowl\ 2)P(bowl\ 2)}$$
$$= \frac{3/4 \cdot 1/2}{3/4 \cdot 1/2 + 1/2 \cdot 1/2} = 3/5.$$

## Exercise 3b)

We first note that for $n < 60$ we have $P(N = n|D = 60) = 0$. For $n \geq 60$ we have

$$P(N = n|D = 60) = \frac{P(D = 60|N = n)P(N = n)}{\sum_{k=60}^{1000} P(D = 60|N = k)P(N = k)}.$$

Using that $P(N = k) = \frac{1}{1000}$ for $1 \leq k \leq 1000$ and $P(D = 60|N = k) = \frac{1}{k}$ we get

$$P(N = n|D = 60) = \frac{1}{n \cdot \sum_{k=60}^{1000} \frac{1}{k}}. \tag{1}$$

This expression maximizes when $n$ minimizes. So a good guess would be that there are $n = 60$ Uber cars.

Next we want to calculate $E[N|D = 60]$. By using the expression obtained in (1) we get

$$E[N|D = 60] = \sum_{n=60}^{1000} n \cdot P(N = n|D = 60) = \sum_{n=60}^{1000} n \cdot \left( \frac{1}{n \cdot \sum_{k=60}^{1000} \frac{1}{k}} \right) = \sum_{n=60}^{1000} \left( \frac{1}{\sum_{k=60}^{1000} \frac{1}{k}} \right).$$

We notice that $\sum_{k=60}^{1000} \frac{1}{k}$ is approximately the area under the curve $y = 1/x$ for $x$ between 60 and 1000. Calculating this as a normal integral gives us the approximation

$$\sum_{k=60}^{1000} \frac{1}{k} \approx \ln \frac{1000}{60}.$$

Finally, we get that

$$E[N|D = 60] \approx \frac{941}{\ln \frac{1000}{60}} \approx 334.$$