Solution to homework 3: *k*-Nearest Neighbour and Naive Bayes

Dr. Juliane Klatt
juliane.klatt@bsse.ethz.ch

Giulia Muzio
giulia.muzio@bsse.ethz.ch

Dr. Bastian Rieck
bastian.rieck@bsse.ethz.ch

Prof. Dr. Karsten Borgwardt
karsten.borgwardt@bsse.ethz.ch

Homework Part 1: *k*-NN

Dataset

The original dataset was randomly partitioned into two groups: the `train` and `test` sets. The files for each group were saved in subdirectories named `train` and `test` respectively. The script used was `create_dataset.py` (available for download in the course website). The syntax to run the script from the command prompt is:

```
$ python create_dataset.py \
         --seed <number> \
         --datadir <path_to_data> \
         --header <yes/no> \
         --ptrain <percentage_as_decimal> \
         --ptest <percentage_as_decimal> \
         --outdir <path_to_output_dir_where_sets_will_saved>
```

where

`-seed` is the value to assign to the random seed to reproduce the results, i.e. to generate the same random partition of the data.

`-datadir` is the data directory. All files in this directory will be partitioned.

`-header` boolean flag that specifies if there is a header line in the file(s). Possible values are [yes/no].

`-ptrain` percentage of records used for training (e.g. 0.7).

`-ptest` percentage of records used for testing (e.g. 0.3). This is used in case `ptrain + ptest` < 1.0.

`-outdir` is the path to the output directory. Two subdirectories namely `train` and `test` will be created.

An important aspect of the data that was omitted in the text of the assignment is that the rows of the files `matrix_mirna_input.txt` and `phenotype.txt` must not necessarily be paired. This implies that we cannot assume the first row of `matrix_mirna_input.txt` corresponds to the same patient whose phenotype information is in the first row of `phenotype.txt`, and so forth. The column `patientId` in both files should be used to match the samples between the files.

Exercise 1.a Programming component.

The source files can be downloaded from the course website. The output file is shown in Figure 1.

Value of k	Accuracy	Precision	Recall
1	0.81	0.81	0.93
2	0.81	0.81	0.93
3	0.71	0.79	0.79
4	0.71	0.79	0.79
5	0.76	0.85	0.79
6	0.76	0.85	0.79
7	0.76	0.80	0.86
8	0.76	0.80	0.86
9	0.71	0.79	0.79
10	0.71	0.79	0.79

Figure 1: Contents of the output file `output_knn.txt` after executing the program.

Exercise 1.b Assume that your program is executed in a way that explores many different values of k. An expert will argue that by looking at the output file `output_knn.txt` we cannot determine what the best value of k is. How do we need to structure our data and what process do we need to follow in order to be able to determine the best k?

We need to implement an internal cross-validation on the training data in order to determine the best k without overfitting. The idea is to simply partition the data into `train` and `validation`. Subsequently, `train` is repartitioned as `train` and `test`. The `train + test` partitioning is done multiple times for different values of k. Then this value is applied to our `validation` set, which has not been used either to train or test the classifier. The partition of the data into `train` and `validation` must be done multiple times and the "internal" repartitioning as `train` and `test` is repeated as indicated before.

Another alternative to internal cross-validation is bootstrapping on the training data. Please refer to the lecture slides 83-85 and 94 of "Part 2: Classification Algorithms" for more details on internal cross-validation and bootstrapping.

Exercise 1.c Using big O notation and assuming all data have been preprocessed, what is the time complexity of the training step in k-NN? And the space complexity?

With the assumption that the data have been preprocessed, the training complexity of k-NN is non-existent because the algorithm does not spend any time in the training phase. The space complexity, on the other hand, is the amount of space in memory occupied by the training data points. For n data points, each of them of dimension d, the space complexity is $O(nd)$.

Exercise 1.d Now focusing on the prediction step, is its complexity different in a problem with c classes, where $c > 2$?

The complexity of k-NN is the same when the number of classes $c > 2$. As discussed in the lecture, a naive implementation of the k-NN algorithm has a complexity of $O(n + n \log n)$ (slide 90). For a given $c > 2$, finding which label is more frequent among the k closest neighbors does not increase the overall complexity of the algorithm.

Exercise 1.e In your implementation, you used the *Euclidean distance*, which is a metric. Does k-NN work with other metrics such as the Manhattan distance as well? Moreover, does it also work for semimetrics, i.e. functions that do not satisfy the triangle inequality, such as DTW? Briefly justify your answer.

k-NN does not explicitly require the use of a metric, and supports similarity measures as well, provided that one takes into account that one might have to switch the min in the distance calculations for a max.

Exercise 1.f So far, you have used k-NN for *classification*. Is it possible to use k-NN for *regression* as well? More precisely, suppose you have data points $\mathbf{X} := \{\mathbf{x}_1, \mathbf{x}_2, \dots\}$ with associated measurements $Y := \{y_1, y_2, \dots\}$, where each $y_i \in \mathbb{R}$. Given a new measurement $x' \notin \mathbf{X}$, can you predict its measurement y'?

It is possible to extend k-NN to a regression task by, for example, calculating the *mean* or *median* of neighbours.

Homework Part 2: Naive Bayes

Dataset

The original dataset was first modified to include missing values. Then it was randomly partitioned into the `train` and `test` sets as described in Part 1.

Exercise 2.a Programming component and calculations.

The source code can be downloaded from the course website. The script made use the `pandas` library in Python. The solution package includes a short tutorial on how to use `pandas` to summarize the data for this exercise. The file is named `using_pandas.html`.

The output files are shown in Figures 2 and 3.

Value	clump	uniformity	marginal	mitoses
1	0.315	0.836	0.821	0.970
2	0.103	0.081	0.080	0.019
3	0.209	0.053	0.067	0.005
4	0.145	0.019	0.011	0.000
5	0.182	0.000	0.007	0.002
6	0.034	0.005	0.009	0.000
7	0.002	0.002	0.000	0.002
8	0.009	0.002	0.000	0.002
9	0.000	0.002	0.002	0.000
10	0.000	0.000	0.002	0.000

Figure 2: Contents of the output file `output_summary_class_2.txt` after executing the program.

Value	clump	uniformity	marginal	mitoses
1	0.013	0.018	0.137	0.574
2	0.013	0.037	0.091	0.112
3	0.054	0.115	0.105	0.126
4	0.049	0.138	0.119	0.049
5	0.188	0.124	0.087	0.018
6	0.067	0.106	0.078	0.013
7	0.090	0.069	0.050	0.027
8	0.166	0.115	0.105	0.031
9	0.063	0.018	0.014	0.000
10	0.296	0.258	0.215	0.049

Figure 3: Contents of the output file `output_summary_class_4.txt` after executing the program.

Use the probabilities reported in `output_summary_class_<label>.txt` to predict the class label of the following data point:

```
[ clump = 6, uniformity = 2, marginal = 2, mitoses = 1 ]
```

Include details of the calculations and the probabilities in the same way as it was done during the lecture with slide 103 of "Part 2: Classification Algorithms".

Given $\mathbf{x} = [6, 2, 2, 1]$, our goal is to determine the maximum of $P(y = 2|X = \mathbf{x})$ and $P(y = 4|X = \mathbf{x})$. The label that attains the maximum probability is the one that will be predicted for $\mathbf{x}$. We know that (lecture slide 100)

$$\underset{y_i}{\text{argmax}} \; P(y = y_i|X = \mathbf{x}) \propto P(Y = y_i) \prod_{j=1}^{d} P(X_j = x_j|Y = y_i) \tag{1}$$

Translating equation 1 to our example, we will have for `class` = 2 (benign)

$$P(y = 2|X = [6, 2, 2, 1]) \propto P(Y = 2) \cdot P(\texttt{clump} = 6|Y = 2) \cdot P(\texttt{uniformity} = 2|Y = 2) \cdot P(\texttt{marginal} = 2|Y = 2) \cdot P(\texttt{mitoses} = 1|Y = 2)$$

We have to look up the probabilities in the output files shown in Figures 2-3 for each label $y_i \in \{2, 4\}$ and for each attribute $X_j \in \{\texttt{clump}, \texttt{uniformity}, \texttt{marginal}, \texttt{mitoses}\}$. The prior probabilities are computed based on the number of samples in each class. We then have $P(Y = 2) = \frac{440}{664} = 0.663$ and $P(Y = 4) = 1 - P(Y = 2) = 0.337$

Now we have all we need to predict the label of $\mathbf{x}$

$$P(y = 2|X = \mathbf{x}) \propto 0.663 \cdot 0.034 \cdot 0.081 \cdot 0.080 \cdot 0.970 = 0.000142$$
$$P(y = 4|X = \mathbf{x}) \propto 0.337 \cdot 0.067 \cdot 0.037 \cdot 0.091 \cdot 0.574 = 0.0000436$$

Therefore, the predicted class label for $\mathbf{x}$ is 2.

Exercise 2.b The data contain missing values. How do these affect the computation of the probabilities in Figures 2-3?

In Naive Bayes missing values for a given attribute can be ignored when computing the frequencies –here used as probabilities. In fact, the probabilities in Figures 2-3 were calculated by ignoring the missing values. For example, when `class` = 2 we have 137 samples of `clump` = 1 out of 435 non-missing values for that attribute (see second column in Figure 4). Hence, $P(\texttt{clump} = 1|Y = 2) = \frac{137}{435} = 0.315$. For the same class but when `uniformity` = 1 we have $P(\texttt{uniformity} = 1|Y = 2) = \frac{361}{432} = 0.836$

If the number of missing values in a specific attribute is very large, then its probabilities can potentially be noisier than those computed for attributes with no missing values.

Exercise 2.c What strategy can you suggest to overcome the problem known as "zero-frequency"? This occurs when you need to compute the probability of a feature/value and class but there are zero instances of it in the training data, i.e. $P(X_j = x_j|Y = y_i) = 0$ for a given i and j.

One approach consists in applying additive smoothing –also known as Laplace smoothing– to all the counts before computing the frequencies. In Naive Bayes this can be implemented by adding a *pseudocount*, for example, we can add 1 to all pairs of attribute and value.

Figure 4 shows an example of additive smoothing applied to the attribute `clump` when `class` = 2. The smoothing should be applied to all attributes and classes.

Homework Part 3: Bayes' Theorem

Exercise 3

Exercise 3.a Suppose you are invited to a Halloween party. There are two opaque bowls of candy, and you are being told that one contains 30 vanilla brownies and 10 chocolate brownies (bowl 1), while the other contains 20 vanilla brownies and 20 chocolate brownies (bowl 2). You pick one bowl at random and draw a vanilla brownie.

Value	clump (original)	clump (+1)	New probability
1	137	138	0.310
2	45	46	0.103
3	91	92	0.207
4	63	64	0.144
5	79	80	0.180
6	15	16	0.036
7	1	2	0.004
8	4	5	0.011
9	0	1	0.002
10	0	1	0.002
Sum	435	445	1.000

Figure 4: Counts of the number of samples for each value of the attribute `clump` when `class` = 2; the 2nd. column shows the original counts, which were used to compute the probabilities in Figure 2; the 3rd. column shows the smoothed counts with the addition of 1; the last column are the new smoothed probabilities.

Use Bayes' theorem to calculate the probability that the bowl you selected is bowl 1. Give your answer as precisely as possible and show how you arrived at it.

Let $B \in \{1, 2\}$ refer to the bowl number, and $T \in \{V, C\}$ refer to the type of brownie. Bayes' theorem states:

$$P(B = 1 \mid T = V) = \frac{P(T = V \mid B = 1)P(B = 1)}{P(T = V)} \tag{2}$$

We have $P(T = V) = P(B = 1, T = V) + P(B = 2, T = V) = \frac{1}{2}\frac{3}{4} + \frac{1}{2}\frac{1}{2} = \frac{5}{8}$. Moreover, we have $P(B = 1) = \frac{1}{2}$ and $P(T = V \mid B = 1) = \frac{3}{4}$. Plugging everything in, this yields a probability of $P(B = 1 \mid T = V) = \frac{3}{5}$.

Exercise 3.b Suppose you are taking an Uber ride home after the Halloween party. You notice that your driver's car has the number $D = 60$ stencilled onto it. Assuming that Uber numbers its cars sequentially and that there are no more than $N_{\text{max}} = 1000$ Uber cars in Basel, what would be a probabilistically-motivated "good guess" for the total number of Uber cars? ...

Task: Calculate the posterior probability $P(N \mid D)$ for all valid values of N (you can do this by writing a small Python script that implements the previous equations). For which N does the posterior distribution attain its maximum?

The script `naive_bayes_uber.py` has implemented this. The posterior is obtained as:

$$P(N|D) = \frac{P(D|N)P(N)}{P(D)} = \frac{1/N \cdot 1/N_{max}}{\sum_{N=D}^{N=N_{max}} 1/N \cdot 1/N_{max}}, \text{for } N \geq 60 \tag{3}$$

It turns out that the posterior maximum is at $D = 60$, which is not surprising. This means that, given only a single data point, said point maximizes the posterior probability. It is also possible to obtain the same result through analytical means.

Acknowledgements

This exercise sheet and its solution were created by Damian Roqueiro and Karsten Borgwardt and extended by Bastian Rieck.