Homework 3: k-Nearest Neighbour and Naive Bayes

Theresa Wakonig
Data Mining I

November 9, 2020

1 Exercise 1

k-NN algorithm.

1.a

See *knn.py*, *knn_classifier.py* and *evaluation.py*.

Output with –maxk = 10:

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

1.b

The number of neighbours 'k' we check is a hyperparameter of the nearest neighbour classification problem. If we simply chose the 'best' k for the results of running the script with a large range of ks, the k we chose would be overfit. In fact it would only be suitable for the one specific dataset we work with and might perform poor on more general data. We therefore need to split our data into smaller subsets and optimize k in a cross-validation loop (slide 81) where we alternate between using our subsets of data as training and test sets. Moreover, one should pay attention that the class distribution in the data is not heavily skewed as this would also falsify the performance metrics.

1.c

As k-NN classification is a type of lazy learning, there is no training step where a model is trained. We directly perform the prediction step on the training data.
Both time and space complexity are therefore $O(1)$.

1.d

The prediction step comprises computing the distances to all n neighbours, sorting them and determining the most common class among the k nearest neighbours.
Complexity for distance computation and sorting: $O(n + nlog(n))$

More classes do not result in a change in complexity, as one still searches for the mode of the nearest k labels. Ties could occur more often, but as all distances are already computed and sorted this does not affect the complexity of the problem.

1.e

Yes it works with other metrics as well. I think one can use DTW but has to be aware of the fact that the 'nearest neighbour' might have a small DTW distance but might geometrically speaking not be very close. The DTW distance can also result in zero despite the fact that the two compared series may not be the same. When using the DTW distance in k-NN, the computation could better be described as 'k most similar series'.

1.f

Yes k-NN can also be used for regression. Instead of fitting a regression model one can look at the nearest neighbours from the dataset to predict the dependent variable y for a given regressor x and fit a regression line step by step.

2 Exercise 2

Naive Bayes algorithm.

2.a

See *nbayes_summarize_data.py*.

Output for class 2 (benign):

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

Output for class 4 (malignant):

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

predicted label of data point = 2 (benign)
See lines 116-134 in *nbayes_summarize_data.py* for details of the calculation (implementation of slide 100). Estimating the prior probability P(y) using the class frequencies of the training data and assuming that the two classes have the same probability, both yield the result that the given data point is most likely to be of class label 2.

2.b

For the computation of the likelihoods the missing values are ignored. Each entry in the tables depicted above is calculated by counting the number of occurrences of a specific value for a given feature and dividing it by the total number of measurements/values for that feature (relative frequency). Hence, for every missing value of a feature the total number of measurements decreases by one.

2.c

When the computed likelihood $P(x_j|y_i)$ yields zero it should be left out or dropped when computing the posterior probability. The zero term multiplied with all other likelihoods would just result in a posterior of zero, which is unwanted. Dropping the zero frequency is essentially the same as replacing it with a one. Multiplication by one does not alter the result in an unwanted way.

3 Exercise 3

Bayes' Theorem.

3.a

Shortcuts:

B1, bowl 1
B2, bowl 2
V, vanilla

Probabilities:

$P(B1) = P(B2) = 0.5$ (pick random bowl)
$P(V|B1) = \frac{30}{40} = 0.75$; $P(V|B2) = \frac{20}{40} = 0.5$

$$P(B1|V) = \frac{P(V|B1)P(B1)}{P(V)} = \frac{P(V|B1)P(B1)}{P(V|B1)P(B1)+P(V|B2)P(B2)} = \frac{0.75 \cdot 0.5}{0.75 \cdot 0.5 + 0.5 \cdot 0.5} = 0.6 = 60\%$$

The probability that the selected bowl is bowl 1 is 60%.

3.b

See *nbayes_uber.py*.

Output of my script:

```
Maximum posterior probability at N = 60 with a value P(N|D) = 0.005905417875729855 = 0.59%
```