Homework 3

Lucas Cosier

Data Mining I

ETH zürich

November 15, 2022

Problem 1

a) **Solution.** Check the provided implementation.

b) **Solution.** One could try to use k-fold cross validation by splitting the training set into k distinct sets, where $k-1$ would be used for training and 1 for validation. Then, the best k (as in, nearest neighbour) value would be chosen by inspecting the performance of the model on the validation set.

c) **Solution.** Preliminaries;
Input: Labelled data sets for training and testing, K
$X_{trn} \in \mathbb{R}^{N \times d}$, $X_{tst} \in \mathbb{R}^{N_{tst} \times d}$
$Y_{trn} \in \mathbb{R}^{N}$, $Y_{tst} \in \mathbb{R}^{N_{tst}}$, $K \in \mathbb{N}$
The **Computational complexity** is $\mathcal{O}(1)$ since there is no training phase in kNN. The training phase just consists of storing the feature vectors and class labels of the training samples, so the **spatial complexity** is $\mathcal{O}(Nd)$, where d is the dimensionality and n the number of samples in the set.

d) **Solution.** For prediction, data movement can be broken down into three parts: **1)** First, row-wise accesses to data points from the test and train matrices (representing d-dimensional feature vectors) are performed for computing the l_2-norm between each observation in the test and training set. **2)** Then, once the Euclidean distances have been computed, they are sorted. **3)** Lastly, the indices of these sorted distances are copied into a matrix to form the KNN matrix ($N_{tst} \times N$).

Therefore, the **Computational complexity** is $\mathcal{O}(N_{tst} \cdot NlogN)$ because we need to sort N_{tst} entries. Since this term does not depend on k, it matters not how many neighbours we consider, since the sorting is *exhaustive*.

e) **Solution.** kNN can work with any distance metric as long as it is a metric. It is up to the practitioner to define the notion of similarity, or distance. One can even consider kernels with kNN.

Since semimetrics are metrics, then kNN would in theory work with something like DTW for time series classification, for example. One case I could think about is when one would want to measure the similarity between two time sequences, and DTW could be used to accommodate sequences of varying lengths to incorporate this measure of similarity.

f) **Solution.** kNN can be extended to regression by computing statistical quantities (like the mean) wrt. the nearest neighbours. For the point x' we can find its k-nearest neighbours x_k which have associated measurements y_k. We can take the mean of those measurements to predict y'. However, I doubt anyone would use this algorithm in practice.

y_i	$\mathbb{P}(Y = y_i)$	$\mathbb{P}(\text{clump} \mid y_i)$	$\mathbb{P}(\text{uniformity} \mid y_i)$	$\mathbb{P}(\text{marginal} \mid y_i)$	$\mathbb{P}(\text{mitoses} \mid y_i)$
2	0.663	0.182	0.081	0.067	0.97
4	0.337	0.188	0.037	0.105	0.574

Table 1: Reported probabilities for test point [clump = 5, uniformity = 2, marginal = 3, mitoses = 1]

Problem 2

a) **Solution.** The reported probabilities for test point $X = [5, 2, 3, 1]^T$ can be found in in Table 1. For prediction my implementation uses log probabilities. For X the predicted class label was 2 since $\log \mathbb{P}(Y = 2 \mid X) = -7.37$ while $\log \mathbb{P}(Y = 4 \mid X) = -8.86$.

b) Missing values are omitted in the calculation of the likelihood probabilities. The absence of these values makes some probabilities be 0. This in practice is partially addressed by imputing values in place of the missing entries.

c) The problem is called the zero-frequency problem, and is usual when you have products of probabilities, and some are 0. The fix is called additive smoothing, and applies small noise to frequencies which are 0. This is offered under the current implementation.

Problem 3

a) Encode bowl 1 and 2 as the categorical variable Y taking values in $\{1, 2\}$. Additionally, encode X to be the probability that the candy is vanilla or chocolate. For Bowl 1 drawing a vanilla type of candy amounts to $\mathbb{P}(Y = 1 \mid X = \text{"vanilla"}) = \frac{\mathbb{P}(X=\text{"vanilla"}|Y=1)\mathbb{P}(Y=1)}{\mathbb{P}(X=\text{"vanilla"})}$
We are already given that $\mathbb{P}(X = \text{"vanilla"} \mid Y = 1) = \frac{30}{40} = 3/4$. Additionally, we know $\mathbb{P}(Y = 1) = \frac{40}{80} = 1/2$ since there are 40 candies in bowl 1 and 80 in total. Since $\mathbb{X} = \text{"vanilla"} = \frac{50}{80} = 5/8$ 50 total vanilla candies out of 80, we end up with the probability of selecting bowl 1 given that we have drawn a vanilla candy of

$$\frac{\mathbb{P}(X = \text{"vanilla"} \mid Y = 1)\mathbb{P}(Y = 1)}{\mathbb{P}(X = \text{"vanilla"})} = \frac{3}{5}$$

b) (a) The maximum posterior is achieved at $N = 60$. Check the implementation provided in `bayes.py` for details.

(b) The expected value of the posterior rounded to the nearest integer is 333. Details found in `bayes.py`.