Exercises

1. Suppose that the vocabulary is {cat, dog, car, bus, ran, fast, the, and, sat}. There are two document classes, namely *animal* and *vehicle*. The training dataset of a text classification model consists of only three sentences, which include “the dog ran fast”, “the cat sat” and “the car and the bus ran fast”. The first two sentences are labeled *animal* and the last sentence is labeled *vehicle*.

   (a) Draw count-based vector representations of all three sentences using bag-of-word features.

   (b) Manually cluster the training examples using 2-means.

   (c) Manually calculate the values of all parameters for a Naïve Bayes classifier, using them to predict the class label of each training example, and the class label of the unseen test example “the dog sat”.

   (d) Manually calculate the feature vectors for multi-class SVM for all the training example above, and the feature vector of the unseen example “the dog sat” if the output is *vehicle*.

2. Verify the 2-means and 3-means clustering results for the examples in Table 1 (b), discussed in the end of Section 3.1.1.

3. Feature vectors are arrays mathematically. On the other hand, they are highly sparse. Discuss the advantages of using hash table to store feature vectors. Compare the time complexity of calculating model scores of SVMs and perceptrons using array data structures and hash table data structures to implement feature vectors and parameter vectors.

4. **k-nearest-neighbor** (kNN) is a **non-parametric** text classifier, which uses no fixed set of model parameters, but takes *instance-based learning*. Given a set of inputs, it records the feature representation of each input, and their correctponding output labels. For testing, given an unseen input, kNN uses the k nearest neighbours of the input in the feature vector space to determine the output class label. In particular, Euclidean distance can be used to measure vector space distance, and simple voting of training data class labels can be used to determine the class label of the test input.

   (a) Compare kNN with Naïve Bayes for their training speed and testing speed theoretically.

   (b) Does kNN require linear separable training data?

   (c) Does the value of k affect the decision of kNN? Given examples to demonstrate your conclusion.

5. Prove that the distance between a vector $\vec{v}_0$ and a hyperplane $\vec{\omega}^T\vec{v}+b=0$ is

$$r = \frac{|\vec{\omega}^T\vec{v}_0 + b|}{||\vec{\omega}||}$$

(Hint: find a vector $\vec{v}_1$ on the hyperplane, such that $\vec{v}_1 - \vec{v}_0$ is perpendicular to the hyperplane. You have $\vec{\omega}^T \vec{v}_1 + b = 0$ ($\vec{v}_1$ on hyperplane) and $\vec{v}_1 - \vec{v}_0 = \alpha\vec{\omega}$ (perpendicular to hyperplane). Solve the equations for $\vec{v}_1$. The distance is then $|\vec{v}_1 - \vec{v}_0|$.)

6. Suppose that we have defined three feature templates for document classification, including c, wc, and $bi \cdot c$, where c represents a document class, w represents a vocabulary word and bi represents a bigram.

   (a) How large is the size of a feature vector for representing any labeled document?

   (b) The size of such a feature vector can be intolerably large due to the number of bigrams that theoretically exist, which is $|V|^2$, where $|V|$ is the vocabulary size. In practice, one can define elements in a feature vector using only feature instances that exist in a set of training data. As a result, OOV words will not exist in feature vectors. How can this method reduce the number of possible feature instances for the feature templates c, wc and $bi \cdot c$, respectively?

   (c) If feature vectors are defined using the method (b) above, what happens if a feature instance in an unseen test sample is not an element in the feature vector defined using the training data? For this test instance, will a perceptron model trained using the feature vector in (b) give a different classs label compared to one trained using the feature vector in (a)?

   (d) When training SVMs and perceptrons, we consider not only gold-standard training instances, but also *violated constraints*, namely incorrectly labeled training inputs that receive high model scores. These incorrectly labeled samples are referred to as *negative examples*, in contrast to the *positive examples* in the gold-standard training data. Intuitively, there can be feature instances from negative examples that do not exist in the feature vectors defined in (b). For example, the feature instance $\langle w =$"football", $c =$"food"$\rangle$ from a negative example is highly unlikely to exist in a set of gold-standard training data. We call such feature instances **negative features**. It has been shown empirically that using negative features to augment the feature vector defined in (b) can lead to better results by SVMs and perceptrons. Discuss why negative features can be useful.

   (e) Extract feature instances for the document "A cat sat on the mat." with the class label "hobby". (Note that tokenisation is a necessary pre-processing step).

   (f) Extract feature instances for the document "The cat sat on the mat." with the class label "sports". If the instance in (e) is a gold-standard example, then the instance here is a negative example. Extract feature instances for this sample. Which feature instances are likely negative features?

7. Recall the WSD task introduced in Chapter 1. Given a word (e.g. bank) and a context window in a sentence, which typically consists of k words to the left and k words to the right of the target word, the goal is to predict the sense of the target word in the sentence (e.g. financial bank).

   Given a training corpus $D = \{(x_i, y_i)\}|_{i=1}^{N}$, where $x_i = (w_i, c_i)$, with c_i denoting the context window of w_i, WSD can be modelled as a supervised classification task. It turns out that two classes of features are highly useful. One is bag-of-word features, with template $w \in c_i$, and the other is *collocational features*, with $2k$ different feature templates $w_j^c \in c_i, j \in [-k, -k+1, \ldots, -1, 1, 2, \ldots, k]$. Here j denotes the relative position of the context word w_j^c with respect to the target word. The feature template w_j^c can also be denoted as $w\textsc{Position}(w)$, which combines a word and its relative position index. For example, given a context window with $k = 3$ "went to a **bank** to withdraw some", the feature template w_{-3}^c is instantiated once with "went", and the feature template w_2^c is instantiated once with the word "withdraw".

   (a) If only bag-of-word features are used, derive a Naïve Bayes classifier for the WSD task.

   (b) The model above can be extended by integrating collocational features also, resulting in a "bag-of-features" Naïve Bayes model, where each feature instances is generated conditionally independently given a word sense. While more features can empirically improve the accuracies, do you find this model theoretically elegant? Why?

   (c) If both bag-of-word and collocational features are used for discriminative WSD, and the vocabulary size is $|V|$, how large is a feature vector for $|C|$ word senses? (Hint: there are $2k + 1$ combined feature templates in total)?

   (d) Further, if position-sensitive part-of-speech (POS) labels in the context window are also used as features, and there are in total $|L|$ different POS labels, how large is a feature vector?

8. Recall the multi-class SVM definition in Eq **??**, in which we have a bias term b, which can be regarded as a prior for the positive class. One alternative way to define Eq **??** is to have a bias b_c for each individual class c, resulting in $\hat{\vec{\omega}} =_{\vec{\omega}} \frac{1}{2}||\vec{\omega}||^2$,
   $s.t.foralli, x_i \in D\, \{ \quad \vec{\omega}^T(x_i, c_i) + b_{c_i} \geq 1$
   $forallc \neq c_i, \vec{\omega}^T(x_i, c) + b_c \leq -1$

   Now follow the same simplification process of Eq 3.7 and Eq 3.8, deriving a definition of multi-class SVM with multiple bias terms. Which features in the Naive Bayes model do these bias terms correspond to?

Features	d_1	d_2	d_3	d_4	d_1	d_2	d_3	d_4
$w_1 =$“a”	1	1	0	2	0.415	0.415	0	0.83
$w_2 =$“ah”	0	0	1	0	0	0	2.0	0
...								
$w_{1001} =$“book”	1	1	0	1	0.415	0.415	0	0.415
$w_{2017} =$“bought”	1	0	0	0	2.0	0	0	0
$w_{2100} =$“boy”	0	0	0	1	0	0	0	2.0
$w_{3400} =$“I”	0	0	1	1	0	0	1.0	1.0
$w_{4400} =$“is”	0	1	0	0	0	2.0	0	0
...								
$w_{5002} =$“know”	0	0	1	0	0	0	2.0	0
$w_{6013} =$“reading”	0	1	0	1	0	1.0	0	1.0
$w_{7034} =$“saw”	0	0	0	1	0	0	0	2.0
$w_{8400} =$“Tim”	1	1	1	0	0.415	0.415	0.415	0
...								
$w_{13200} =$“,”	0	0	1	0	0	0	2.0	0
$w_{13201} =$“.”	1	0	1	0	1.0	0	1.0	0
...								

(a) count-based vectors (b) TF-IDF vectors

Table 1: Vector representations of documents.