CS 412 Machine Problem 3

1 Question Answering

Q1. What is the difference between discriminative and generative models? (10 points)

Q2. What is the core assumption in Naïve Bayes? This is also the reason why it is called naïve. (10 points)

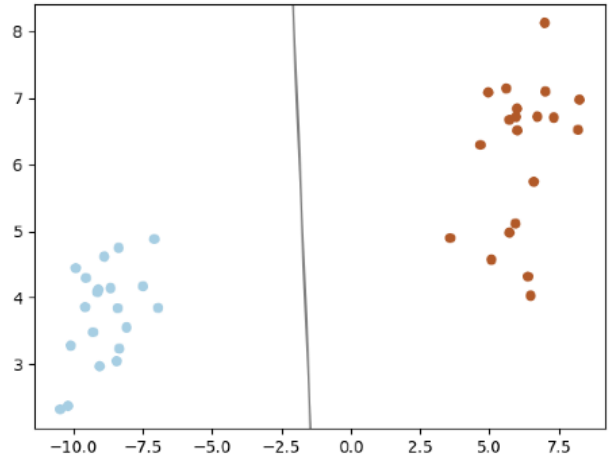
Q3. Given a dictionary of 4 letters (“a”, “b”, “c” and “d”), and a training dataset of 10 strings belonging to two classes (see the table below), please manually learn a Naïve Bayes classifier with the bag of words representations to classify 2 testing strings: “abbd” and “bbcc”. For simplicity, please treat the existence of a letter as a random variable. Your answer should include (1) the learned parameters, (2) the inference/prediction process, and (3) the classification results. Hint: to learn a Naïve Bayes via maximum likelihood estimation, what you need to do is counting. (10 points)

String	“abc”	“abbc”	“aacd”	“bcdd”	“abc”	“ccd”	“add”	“bdd”	“aac”	“ad”
Class	1	1	1	1	1	2	2	2	2	2

Q4. A learned multi-class logistic regression model for 2D input $[x_1, x_2]$ consists of 3 parameter vectors for 3 classes, respectively: [1, 1, 2], [1, -1, 1], and [-1, 1, -1]. The three elements of each parameter vector are weights corresponding to x_1, x_2 and the bias, respectively. Please use this model to classify the 5 testing samples below. Your answer should include (1) the inference/prediction process, and (2) the classification results. (10 points)

x_1	1	0	-10	100	-20
x_2	1	0	10	-50	35

Q5. Suppose a support vector machine (SVM) has learned a decision boundary as shown below. Please draw (1) marks on all support vectors, (2) margin boundaries, and (3) the margin (the distance from the decision boundary to the closest point). (10 points)



2 Programming

The goal of this programming assignment is to let you explore different classification models. It is mandatory to use Python 3. Unless otherwise stated, only the numpy package can be used to implement model learning and prediction, and cross-validation.

P1. Please follow the steps below to implement a K-nearest neighbors (KNN) classifier with K=3. (20 points)

1. Use the code snippet below to generate your training data belonging to two classes. Be careful on the indentation when copying the code.

```
def generate_random_points(size=10, low=0, high=1):
    data = (high - low) * np.random.random_sample((size, 2)) + low
    return data

N = 20 # number of samples in each class
X1 = generate_random_points(N, 0, 1)
y1 = np.ones(N)
X2 = generate_random_points(N, 1, 2)
y2 = np.zeros(N)
X = np.concatenate((X1, X2), axis=0)
y = np.concatenate((y1, y2), axis=0)
indices = np.arange(2*N)
np.random.shuffle(indices)
X = X[indices, :]
y = y[indices]
```

2. Implement the KNN (K=3) to predict the class of a testing sample [1.0, 1.0]. You can use the KNN (K=1) code in our code tutorial as a template (available in Blackboard) or build your own code from scratch. Print (1) the predicted class, and (2) the K nearest neighbors (each is a 2D vector) and their corresponding classes.
3. Use two different colors to respectively draw in a single figure (1) the training data and (2) the testing sample. Then, highlight in this figure the K nearest neighbors, for example, using a different shape or drawing circles around them.

P2. Please follow the steps below to find the optimal K in a KNN classifier on a linearly non-separable dataset. (15 points)

1. Generate a new training set by changing the lines that generate X1 and X2 with the two lines below.

```
X1 = generate_random_points(N, 0, 1.5)
X2 = generate_random_points(N, 0.5, 2)
```

2. Use 5-fold cross-validation to find the optimal K among {1, 3, 5, 7, 9}. For each value of K, print the classification accuracy obtained in each iteration of the cross-validation, and print the average classification accuracy on all validation folds. Print the optimal value of K, which achieves the highest average classification accuracy.

P3. Please follow the steps below to explore logistic regression and the support vector machine (SVM). (15 points)

1. Generate a linearly separable training set via the code snippet in P1. Also generate a testing set consisting of three samples: [0.5, 0.5], [1, 1] and [1.5, 1.5].
2. Train a logistic regression model on the training set and make predictions on the testing set. You can use the logistic regression code in our code tutorial (available in Blackboard) or use the implementation[1] from the scikit-learn[2] package. Print the predicted classes for the three testing samples.
3. Study the tutorial[3] on the SVM implementation in the scikit-learn package. Then apply it to train a SVM on the training set and make predictions on the testing set. You may use the same setting as in the tutorial (kernel='linear', C=1000). Print the predicted classes for the three testing samples. Be careful on the data type of ground truth labels.
4. Use three different colors to respectively draw in a single figure (1) the training data, (2) the decision boundary learned via logistic regression, and (3) the decision boundary learned via SVM.

4 Submission

Please follow the instructions below for submission.

- You need to upload two files to Blackboard: a PDF file and a .py file[4]. Do not compress them into a single ZIP file.
- The PDF file contains all your solutions to this homework. For Question Answering, you can either type answers or handwrite them and take a photo. For Programming, you need to print the results or draw figures, and take screenshots.
- The .py file contains all your code for the programming problems.

[1] This website includes the detailed usage and an example of logistic regression in scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[2] Installation of scikit-learn: https://scikit-learn.org/stable/install.html
[3] https://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html#sphx-glr-auto-examples-svm-plot-separating-hyperplane-py
[4] Using Jupyter Notebook and submitting a .ipynb file instead of a .py file are fine.