CSCI 5521: Introduction to Machine Learning (Fall 2021)[1]

Homework 4

Due date: Dec 15, 2021 11:59pm

1. (**30 points**) Suppose we use a linear SVM classifier for a binary classification problem with a set of data points shown in Figure 1, where the samples closest to the boundary are illustrated: samples with positive labels are (0, 1), (-2, -1), (1, 3), (-1, 1.5), (-1.5, 0.5), and samples with negative labels are (0.5, -0.5), (-0.5, -2), (1.5, -1), (3, 0.5), (-1, -2.5).

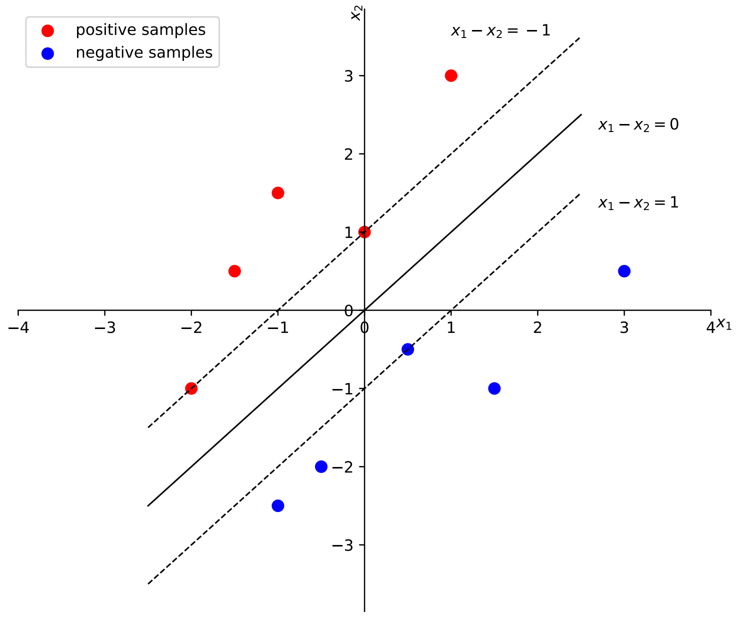

Figure 1: For a binary classification problem where positive samples are shown red and negative ones shown blue. The solid line is the boundary and dashed ones define the margins.

(a) List the support vectors.

(b) Pick three samples and calculate their distances to the hyperplane $x_1 - x_2 = 0$.

(c) If the sample (0.5, -0.5) is removed, will the decision boundary change? What if we remove the sample (-1, -2.5) instead?

[1]Instructor: Catherine Qi Zhao. TA: Shi Chen, Xianyu Chen, Helena Shield, Jinhui Yang, Yifeng Zhang. Email: csci5521.f2021@gmail.com

(d) If a new sample (0, 2) comes as a negative sample, will the decision boundary change? If so, what method will you use in this case?

(e) In the soft margin SVM method, C is a hyperparameter (see Eqs. 13.10 or 13.11 in Chapter 13.3 of the textbook). What would happen when you use a very large value of C? How about using a very small one?

(f) In real-world applications, how would you decide which SVM methods to use (hard margin vs. soft margin, linear vs. kernel)?

2. (**30 points**) Table 2 shows data collected on a person's decision to go for a run or not go for a run depending on the weather conditions that day. Answer the following questions:

Outlook	**Temperature**	**Humidity**	**Run?**
Rainy	Mild	High	No
Overcast	Cool	Normal	No
Rainy	Cool	High	No
Overcast	Mild	Normal	No
Rainy	Cool	Normal	No
Sunny	Hot	High	Yes
Sunny	Mild	Normal	Yes
Sunny	Cool	Normal	Yes
Sunny	Cool	Normal	Yes
Rainy	Hot	High	No
Sunny	Mild	Normal	Yes
Rainy	Mild	Normal	Yes
Sunny	Mild	Normal	Yes
Sunny	Hot	High	Yes
Sunny	Cool	Normal	Yes

(a) We wish to build a decision tree to help decide if the runner will go for run tomorrow. Draw the decision tree that fits this data and show how to calculate each node split using entropy as the impurity measure.
Note: If the entropy is the same for two or more features, you can select any of the features to split.

(b) Based on the decision tree, will the runner go for a run in the following cases?

- Tomorrow is **Overcast**, the temperature is **Cool**, and the Humidity level is **High**.
- Tomorrow is **Rainy**, the temperature is **Hot**, and the Humidity level is **Normal**.

3. (**40 points**) In this programming exercise you will implement a Decision Tree for optical-digit classification. You will train your Decision Tree using the `optdigits_train.txt` data, tune the minimum node entropy using `optdigits_valid.txt` data, and test the prediction performance using the `optdigits_test.txt` data. For each file, the first 64 columns correspond to the binary features for different samples while the last one stores the labels. The minimum node entropy is used as a threshold to stop splitting the tree, and set the node as a leaf.

   (a) Implement a Decision Tree with the minimum node entropy θ=0.01, 0.05, 0.1, 0.2, 0.4, 0.8, 1.0 and 2.0.
   Report the training and validation accuracy by different θ.
   What θ should you use? Report the accuracy on the test set using this θ.

   (b) What can you say about the model complexity of the Decision Tree, given the training and validation accuracy? Briefly explain.

   We have provided the skeleton code `MyDecisionTree.py` for implementing the algorithm. `MyDecisionTree.py` is written in a *scikit-learn* convention, where you have a *fit* function for model training and a *predict* function for generating predictions on given samples. To verify your implementation, call the main function `hw4.py`.

   More details can be found in comments of the source files.

Submission

- **Things to submit:**

  1. hw4_sol.pdf: a document containing all your answers for the written questions (including answers for problem 3).
  2. `MyDecisionTree.py`: a Python source file containing your implementation of the decision tree algorithm. Use the skeleton file `MyDecisionTree.py` found with the data on the class web site, and fill in the missing parts, including the *predict* function and auxiliary functions for the *fit* function (see comments for details about each function) . **The *fit* function should recursively construct the decision tree with the auxiliary function *generate_tree*.** The *predict* function should take features as inputs and return the predicted class labels.

- **Submit**: All material must be submitted electronically via Gradescope. **Note that There are two entries for the assignment, *i.e.,* Hw4-Written (for hw4_sol.pdf) and Hw4-Programming (for a zipped file containing the Python code), please submit your files accordingly.** We will grade the assignment with vanilla Python, and code submitted as iPython notebooks will not be graded.