Homework Assignment 6

COGS 118A: Supervised Machine Learning Algorithms

Due: Dec 6th, 2020, 11:59pm (Pacific Time).

Instructions: Combine your answer to the questions below with your notebook to create a PDF file; submit your file via Gradescope.

1 (8 points) Conceptual Questions

1. Which one below best describes what support vectors are:

   A. The decision boundary.

   B. The positive and the negative planes.

   C. The training samples that determine the positive and the negative planes.

   D. The test samples that determine the positive and the negative planes.

2. When tuning the hyper-parameters of a model, we find the hyper-parameters that

   A. minimize error on the test set

   B. minimize error on the test set

   C. maximize the margin of the classifier

   D. minimize error on the validation set

3. Select **all** that apply. k-fold cross validation is

   A. Not necessary when you have huge amounts of labelled data

   B. A way to do model selection while minimizing over-fitting

C. The only way to do hyper-parameter tuning

D. Subject to the bias-variance trade-off

4. When you increase the k in cross-validation while keeping the dataset the same, you

   A. Get a decrease in both the generalization error and the validation error

   B. Get an increase both the generalization error and the validation error

   C. Get a decrease in the variability of the validation error across folds

   D. Get an increase in the variability of the validation error across folds

2 (10 points) Cross-Validation

Given a training dataset $S_{\text{training}} = \{(x_i, y_i)\}, i = 1, \dots, 6\}$ where $x_i \in \mathbb{R}$ is the feature scalar and $y_i \in \{-1, +1\}$ is the corresponding label. The data points in the dataset S_{training} are given below:

$$(x_1, y_1) = (2, -1), \quad (x_2, y_2) = (7, -1), \quad (x_3, y_3) = (4, +1),$$

$$(x_4, y_4) = (1, -1), \quad (x_5, y_5) = (3, +1), \quad (x_6, y_6) = (6, +1).$$

Suppose you are training a linear classifier $f(x; a, b) = \text{sign}(ax + b)$ with 2-fold cross-validation where $\text{sign}(z)$ is defined as:

$$\text{sign}(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0. \end{cases}$$

- You have split the dataset S_{training} into:

$$S_1 = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$$

$$S_2 = \{(x_4, y_4), (x_5, y_5), (x_6, y_6)\}$$

- After training the classifier $f(x; a, b)$ on S_1, you have obtained the parameters $a_1 = -1, b_1 = 5$ and then try to validate the classifier on S_2.
- After training the classifier $f(x; a, b)$ on S_2, you have obtained the parameters $a_2 = 2, b_2 = -3$ and then try to validate the classifier on S_1.

Please finish the tasks below:

1. Calculate the **average training error** in the 2-fold cross-validation.

   Note: The definition of average training error is the mean of the error of classifier $f(x; a_1, b_1)$ on S_1 and the error of classifier $f(x; a_2, b_2)$ on S_2.

2. Calculate the **average validation error** (i.e. the cross-validation error) in the 2-fold cross-validation.

3 (12 points) Shattering

In this problem, consider a classifier $f(\mathbf{x}; a, b) = \text{sign}(a\mathbf{x}^\top\mathbf{x} - b)$ where the feature vector $\mathbf{x} = [x_1, x_2]^\top \in \mathbb{R}^2$ and its prediction $f(\mathbf{x}; a, b) \in \{-1, +1\}$. Besides, $a, b \in \mathbb{R}$ are the model parameters and $\text{sign}(z)$ is defined as:

$$\text{sign}(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0. \end{cases}$$

The classifier $f(\mathbf{x}; a, b)$ performs a binary classification on an input feature vector $\mathbf{x}$ under model parameters $a, b \in \mathbb{R}$ that can be learned.

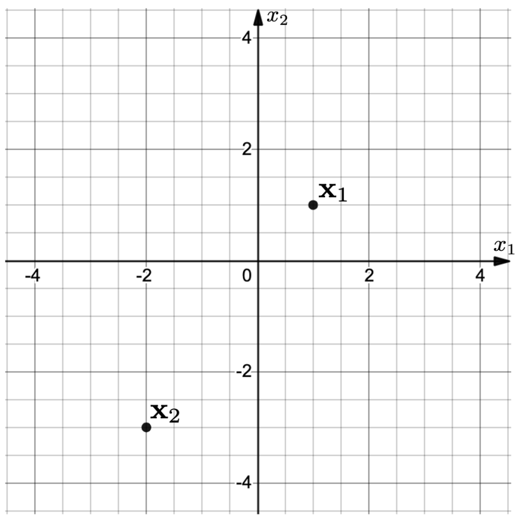
In this question, please attempt to show that if the classifier $f(\mathbf{x}; a, b)$ can shatter two points, $\mathbf{x}_1 = [1, 1]^\top$ and $\mathbf{x}_2 = [-2, -3]^\top$.

- If you think $f(\mathbf{x}; a, b)$ **can** shatter $\mathbf{x}_1$ and $\mathbf{x}_2$, you need to show that for each possible label configuration $(y_1, y_2) \in \{(+1, +1), (-1, -1), (+1, -1), (-1, +1)\}$ of $\mathbf{x}_1$ and $\mathbf{x}_2$, there exists a classifier $f(\mathbf{x}; a, b)$ that classifies the $\mathbf{x}_1$ and $\mathbf{x}_2$ correctly, and you should illustrate each classifier by the following steps:

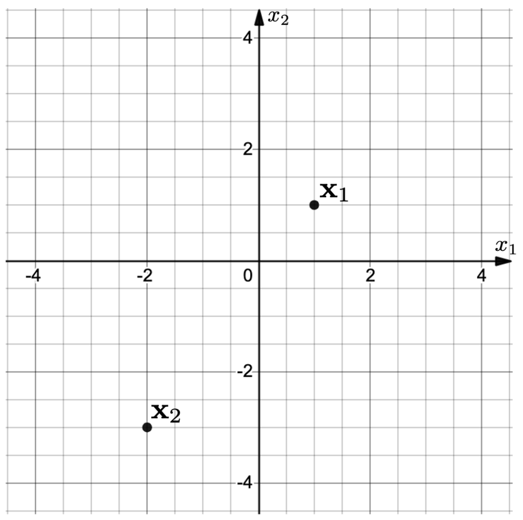
  - Draw the decision boundary of the classifier.
  - Shade the area where the classifier makes the positive prediction.

  You can use the figures of the coordinate system in the next page to show your drawings.

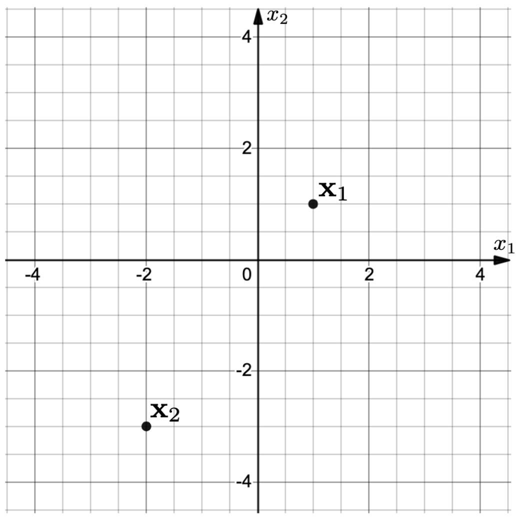
- If you think classifier $f(\mathbf{x}; a, b)$ **cannot** shatter $\mathbf{x}_1$ and $\mathbf{x}_2$, please explain the reason.



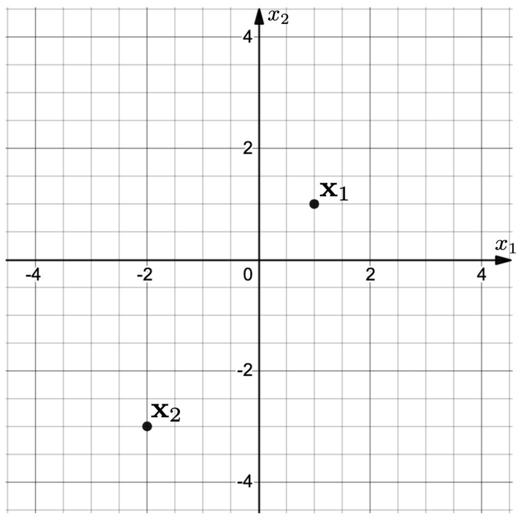
$(y_1, y_2) = (+1, +1)$



$(y_1, y_2) = (-1, -1)$



$(y_1, y_2) = (+1, -1)$



$(y_1, y_2) = (-1, +1)$

4 (10 points) SVM: Gradient

Given a training dataset $S_{\text{training}} = \{(\mathbf{x}_i, y_i)\}, i = 1, \ldots, n\}$, we wish to optimize the loss $\mathcal{L}(\mathbf{w}, b)$ of a linear SVM classifier:

$$\mathcal{L}(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||_2^2 + C \sum_{i=1}^{n} \big(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\big)_+ \tag{1}$$

where $(z)_+ = \max(0, z)$ is called the rectifier function and C is a scalar constant.

The optimal weight vector $\mathbf{w}^*$ and the bias b^* used to build the SVM classifier are defined as follows:

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w}, b} \mathcal{L}(\mathbf{w}, b) \tag{2}$$

In this problem, we attempt to obtain the optimal parameters $\mathbf{w}^*$ and b^* by using a standard gradient descent algorithm.

Hint: To derive the derivative of $\mathcal{L}(\mathbf{w}, b)$, please consider two cases:
(a) $1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 0$, (b) $1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) < 0$

1. Derive the derivative: $\dfrac{\partial \mathcal{L}(\mathbf{w}, b)}{\partial \mathbf{w}}$.

2. Derive the derivative: $\dfrac{\partial \mathcal{L}(\mathbf{w}, b)}{\partial b}$.

5 (10 points) SVM: Margin

As shawn in the Figure 3, we have the decision boundary (marked as a black line) defined as Eq. 3 given $\mathbf{w}, b$:

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{3}$$

In parallel to the decision boundary, we have the positive plane (marked as a red line) defined as Eq. 4 and the negative plane (marked as a blue line) defined as Eq. 5:

$$\mathbf{w}^T\mathbf{x} + b = +1 \tag{4}$$

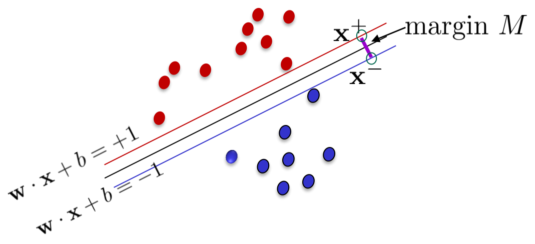
$$\mathbf{w}^T\mathbf{x} + b = -1 \tag{5}$$

We pick an arbitrary point $\mathbf{x}^-$ on the negative plane, and draw a purple line that passes $\mathbf{x}^-$ and is perpendicular to the negative plane. The intersection between this purple line and the positive plane can be denoted as $\mathbf{x}^+$. Thus, we have the following Eq. 6 that indicates the relation between $\mathbf{x}^+$ and $\mathbf{x}^-$:

$$\mathbf{x}^+ = \mathbf{x}^- + \lambda\mathbf{w} \tag{6}$$

where $\lambda \in \mathbb{R}$ is an undetermined scalar. The margin M is defined as the distance between the positive and the negative planes, which can be calculated from Eq. 7:

$$M = ||\mathbf{x}^+ - \mathbf{x}^-||_2 = \sqrt{< \lambda\mathbf{w}, \lambda\mathbf{w} >} \tag{7}$$



$\mathbf{x}^+$
margin M
$\mathbf{x}^-$
$\mathbf{w} \cdot \mathbf{x} + b = +1$
$\mathbf{w} \cdot \mathbf{x} + b = -1$

Figure 3: The decision boundary, the positive plane and the negative plane.

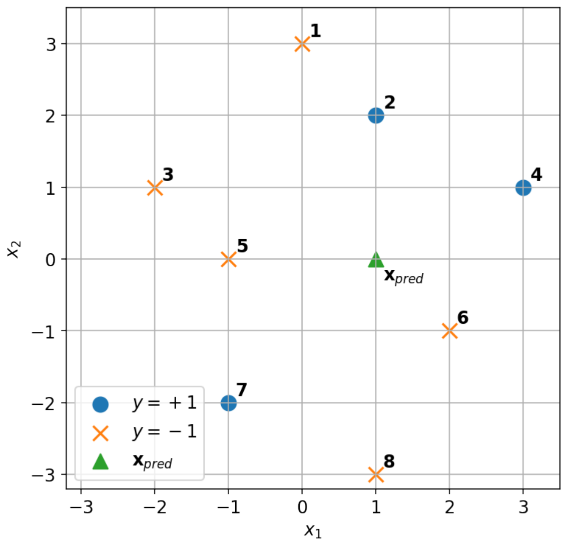
Please derive the following according to Eq. 7:

$$M = \frac{2}{\sqrt{< \mathbf{w}, \mathbf{w} >}}$$

Hint: You can firstly represent λ in the form of $\mathbf{w}$ by using Eq. 4, 5, 6.

6 (10 points) K Nearest Neighbors

Consider a training dataset $S_{\text{training}} = \{(\mathbf{x}_i, y_i), i = 1, 2, ..., 8\}$ where each data point $(\mathbf{x}, y)$ has a feature vector $\mathbf{x} = [x_1, x_2]^\top$ and the corresponding label $y \in \{-1, +1\}$. The points with the corresponding labels in the dataset are shown in the figure below.



In the figure above, the index i for each training example $\mathbf{x}_i$ is given in **bold** near the point. You are asked to predict the label of a point $\mathbf{x}_{\text{pred}} = [1, 0]^\top$ shown in the figure as a triangle ▲ using k nearest neighbors (k-NN) method under **Euclidean distance**.

1. List the indices of all the data points in S_{training} and their corresponding labels.

2. Determine the predicted label for $\mathbf{x}_{\text{pred}}$ using the k-NN with different k.

   (a) $k = 1$.

   (b) $k = 3$.

   (c) $k = 5$.

7 (20 points) Coding: K Nearest Neighbors

In this problem, you need to implement the k nearest neighbors (k-NN) algorithm and apply it to the binary classification. Here we use the modified Iris dataset $S = \{(\mathbf{x}_i, y_i)\}$ where each feature vector $\mathbf{x} \in \mathbb{R}^2$ and label $y \in \{-1, +1\}$. You are **not** allowed to use `sklearn.neighbors.KNeighborsClassifier()` in your code, but you can use it to validate your implementation.

- Load the modified Iris dataset. The dataset S is split to three subsets: The training set S_{training}, the validation set $S_{\text{validation}}$ and the test set S_{test}. In the code, we use `X_train`, `Y_train` for the feature vectors and labels of the training set respectively. Similar notations are also used for the validation and the test sets.

- Implement k-NN algorithm in 3 steps.

  1. For each feature vector $\mathbf{x}$ you are predicting a label, you need to calculate the distances between **this feature vector $\mathbf{x}$** and **all the feature vectors in the training set S_{training}**.
  2. Then sort all distances in ascending order and pick the labels for the k minimum distances.
  3. Count the number of negative labels $N_{y=-1}$, and the number of the positive labels $N_{y=+1}$ from k labels picked in step 2. Use the following decision rule to predict label $\hat{y}$ for each feature vector $\mathbf{x}$:

  $$\hat{y} = \begin{cases} +1, & N_{y=-1} < N_{y=+1}, \\ -1, & N_{y=-1} \geq N_{y=+1}. \end{cases}$$

  Here we assume **Euclidean distance** as the distance metric. For more details, please refer to the code and the corresponding part in the slides.

- Use the validation set to obtain optimal k^*. In k-NN, there is a hyper-parameter k which adjusts the number of nearest neighbors. You would need to perform a grid search on the following list of k:

  $$k \in \{1, 2, 3\}$$

  For each k, you need to form a k-NN classifier with the training set S_{training}. Then, use the classifier to make predictions on the validation set $S_{\text{validation}}$ and calculate the error $e_{\text{validation}}$. We aim to obtain the best hyper-parameter k^* corresponding to the **minimum validation error** $e^*_{\text{validation}}$ among all ks.

- Use the obtained classifier corresponding to the best hyper-parameter k^* to calculate the test error e_{test} on test set S_{test}.

Please download the notebook `knn.ipynb` from the course website and fill in the missing blanks. Follow the instructions in the skeleton code and report:

1. Your code.
2. Plot of validation set along with decision boundary (implicitly shown in the background) corresponding to each k.
3. Validation error corresponding to each k.
4. The best hyper-parameter k^* corresponding to minimum validation error $e^*_{\text{validation}}$.
5. Test error e_{test} corresponding to the best hyper-parameter k^*.

8 (20 points) Coding: Decision Tree

In this problem, you need to implement the decision tree algorithm and apply it to the binary classification. Here we use the Ionosphere dataset $S = \{(\mathbf{x}_i, y_i)\}$ where each feature vector $\mathbf{x} \in \mathbb{R}^{34}$ and label $y \in \{-1, +1\}$. You are allowed to use the functions from `scikit-learn` in this question.

- Load the Ionosphere dataset. The dataset S is split to two subsets: The training set S_{training} and the test set S_{test}. In the code, we use `X_train`, `Y_train` for the feature vectors and labels of the training set respectively. Similar notations are also used for the test set.
- Train the decision tree classifier with the **entropy criterion**. In the decision tree, there is a hyper-parameter D which controls the maximum depth. You would need to perform a grid search on the following list of D:

  $$D \in \{1, 2, 3, 4, 5\}$$

  For each D, you need to form a decision tree classifier with the training set S_{training}. Specifically, you need to conduct a **10-fold** cross-validation on S_{training} and calculate the cross-validation error $\bar{e}$ (i.e. average validation error over the splits in cross-validation). We aim to obtain the best hyper-parameter D^* corresponding to the **minimum cross-validation error** $\bar{e}^*$ among all Ds.
- Use the obtained classifier corresponding to the best hyper-parameter D^* to calculate the test error e_{test} on test set S_{test}.

Please download the notebook `decision-tree.ipynb` and the data file `ionosphere.npy` from the course website and fill in the missing blanks. Follow the instructions in the skeleton code and report:

1. Your code.
2. A heatmap of cross-validation errors corresponding to all Ds.
3. The best hyper-parameter D^* corresponding to the minimum cross-validation error $\bar{e}^*$.
4. Test error e_{test} corresponding to the best hyper-parameter D^*.