CS 6350/DS 4350: HW4 Linear concepts/classes with perceptron algorithms

Samir Abdelrahman

Yujin Song, Shubham Sanjay Sawant

Fall 2025

Question 1 [50 pts]— Linear Separability & LTUs

Inputs and labels are in $\{-1, +1\}$.

Please write your answer inside the pdf, the name is: username_userID_answers.pdf

For additional details not covered in this file, please refer to the class slides:

Lecture7_linear_models.pdf and More_linear_Classifications_Concepts.pdf

Linear Threshold Unit (LTU). An LTU is a function

$$h(x) = \operatorname{sign}(w^\top x + b), \qquad \operatorname{sign}(z) = \begin{cases} +1, & z \geq 0 \\ -1, & z < 0 \end{cases}$$

with weights $w = (w_1, \ldots, w_d)$ and bias b (threshold $\theta = -b$). Setting $w_j = 0$ ignores feature x_j. Negation is written as $\neg x$ and equals $-x$ in $\{-1, +1\}$.

Task. For each target function f below, *write an explicit LTU* $h(x) = \operatorname{sign}(w^\top x + b)$ (i.e., provide concrete w and b) that computes f on every input, and include a short justification (truth table or brief argument). If a function is *not* representable by a single LTU, state "not linearly separable" and justify. *Tip:* choose thresholds to avoid ties ($w^\top x + b \neq 0$ for all inputs).

Q1.1 [10 pts]

- $f_1(x_1, x_2) = x_1 \wedge \neg x_2$ (i.e., $+1$ iff $x_1 = +1$ and $x_2 = -1$; else -1).

Q1.2 [10 pts]

- $f_2(x_1, x_2, x_3)$ is *odd parity* on $\{x_1, x_2, x_3\}$: $+1$ iff an odd number of $\{x_1, x_2, x_3\}$ are $+1$; otherwise -1. (Equivalently, $f_2(x) = x_1 x_2 x_3$.)

Q1.3 [10 pts]

- $f_3(x_1, x_2, x_3, x_4) = +1$ iff at least three of $\{x_1, x_2, x_3, x_4\}$ are $+1$; otherwise -1.

Q1.4 [20 pts]

- **General m-of-n.** Let the d input features be $x_1, \ldots, x_d$ and let $R \subseteq \{1, \ldots, d\}$ be the set of n relevant indices. Define

$$f_{m\text{-of-}n}(x) = \begin{cases} +1, & \text{if at least } m \text{ of } \{x_j : j \in R\} \text{ are } +1, \\ -1, & \text{otherwise.} \end{cases}$$

  Write a single LTU $h(x) = \operatorname{sign}(w^\top x + b)$ (specify all w_j and b) that realizes $f_{m\text{-of-}n}$ for the given m, n, R, and briefly explain why your choice avoids ties.

Question 2 [100 pts + 10 pts for bonus]— Coding of Perceptron Algorithm.

2.1 The Task and Data

For this homework, we will be using the **Banknote Authentication dataset** from the UCI Machine Learning repository[1]. This dataset is a binary classification problem, designed to determine if a given banknote is genuine or forged. The original labels in the dataset are 0 (genuine) and 1 (forged). For this assignment, you must map these labels to **1 (genuine)** and **-1 (forged)** to align with the Perceptron algorithm's formulation.
The dataset consists of four continuous features extracted from images using a Wavelet Transform Tool. Using this labeled data, our goal is to build a classifier that accurately identifies whether a given banknote is genuine or forged.
The data has been preprocessed and split into training, development, and test files: `train.csv`, `validation.csv`, and `test.csv`. For details about the data format, refer to the `README.md` file provided with the dataset.

2.2 Algorithms

You will implement several variants of the Perceptron algorithm. Note that each variant has different hyper-parameters, as described below. Use 5-fold cross-validation to identify the best hyper-parameters as you did in the previous homework. To help with this, we have split the training set into five parts: `fold1.csv` – `fold5.csv` in the folder `cv`.
For additional details not covered in this file, please refer to the class slide:
Lecture12–Perceptron_Continued.pdf

2.2.1 Simple Perceptron (25 points)

Implement the simple batch version of Perceptron as described in the class. Use a fixed learning rate $\eta \in \{1, 0.1, 0.01\}$. An update will be performed on an example $(\mathbf{x}, y)$ if $y(\mathbf{w}^\top \mathbf{x} + b) \leq 0$ as:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta y \mathbf{x}, \qquad b \leftarrow b + \eta y.$$

Hyper-parameter: Learning rate $\eta \in \{1, 0.1, 0.01\}$.

Two things bear additional explanation.

(a) First, note that in the formulation above, the bias term b is explicitly mentioned. This is because the features in the data do not include a bias feature. Of course, you could add a constant feature to each example and not have the explicit extra b during learning. (See the class lectures for more information.) However, here, we will see the version of Perceptron explicitly has the bias term.

(b) Second, in this specific case, if $\mathbf{w}$ and b are initialized with zero, then the fixed learning rate will have no effect. To see this, recall the Perceptron update from above. Now, if $\mathbf{w}$ and b are initialized with zeroes and a fixed learning rate η is used, then we can show that the final parameters will be equivalent to having a learning rate 1. The final weight vector and the

[1] https://archive.ics.uci.edu/dataset/267/banknote+authentication

bias term will be scaled by η compared to the unit learning rate case, which does not affect the sign of $\mathbf{w}^\top\mathbf{x} + b$. To avoid this, you should initialize all elements of the weight vector $\mathbf{w}$ and the bias b to a small random number between -0.01 and 0.01.

2.2.2 Decaying the learning rate (25 points)

Instead of fixing the learning rate, implement a version of the Perceptron algorithm whose learning rate decreases as

$$\eta_t = \frac{\eta_0}{1+t},$$

where η_0 is the starting learning rate, and t is the time step. Note that t should keep increasing across epochs. (That is, you should initialize t to 0 at the start and keep incrementing it after each epoch.)

Hyper-parameter: Initial learning rate $\eta_0 \in \{1, 0.1, 0.01\}$.

2.2.3 Margin Perceptron (25 points)

This variant of Perceptron will perform an update on an example $(\mathbf{x}, y)$ if

$$y(\mathbf{w}^\top\mathbf{x} + b) < \mu,$$

where μ is an additional positive hyper-parameter, specified by the user. Note that because μ is positive, this algorithm can update the weight vector even when the current weight vector does not make a mistake on the current example. You need to use the decreasing learning rate as before.

Hyper-parameters:

(a) Initial learning rate $\eta_0 \in \{1, 0.1, 0.01\}$

(b) Margin $\mu \in \{1, 0.5, 0.1, 0.01\}$

Note: When there is more than one hyper-parameter to cross-validate, you need to consider all combinations of the hyper-parameters. In this case, you will need to perform cross-validation for all pairs (η_0, μ) from the above sets.

2.2.4 Averaged Perceptron (25 points)

Implement the averaged version of the original Perceptron algorithm from the first question. Recall from class that the averaged variant of the Perceptron asks you to keep two weight vectors (and two bias terms). In addition to the original parameters $(\mathbf{w}, b)$, you will need to update the averaged weight vector $\mathbf{a}$ and the averaged bias b_a as:

$$\mathbf{a} \leftarrow \mathbf{a} + \mathbf{w}$$

$$b_a \leftarrow b_a + b$$

This update should happen *once for every example in every epoch, irrespective of whether the weights were updated or not for that example.* In the end, the learning algorithm should return the averaged weights and the averaged bias.

2.2.5 Aggressive Perceptron with Margin (Bonus 10 points)

This algorithm is an extension of the margin Perceptron and performs an aggressive update as follows:
If $y(\mathbf{w}^\top \mathbf{x} + b) \leq \mu$ then:

(a) Update $\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{old}} + \eta y \mathbf{x}$

(b) Update $b_{\text{new}} \leftarrow b_{\text{old}} + \eta y$

Unlike the standard Perceptron algorithm, here the learning rate η is given by

$$\eta = \frac{\mu - y(\mathbf{w}^\top \mathbf{x} + b)}{\mathbf{x}^\top \mathbf{x} + 1}.$$

As with the margin Perceptron, there is an additional positive parameter μ.

Hyper-parameters: $\mu \in \{1, 0.5, 0.1, 0.01\}$.

Explanation of the update. We call this the aggressive update because the learning rate is derived from the following optimization problem: When we see that $y(\mathbf{w}^\top \mathbf{x} + b) \leq \mu$, we try to find new values of $\mathbf{w}$ and b such that $y(\mathbf{w}^\top \mathbf{x} + b) = \mu$ using

$$\min_{\mathbf{w}_{\text{new}}, b_{\text{new}}} \quad \frac{1}{2}(\|\mathbf{w}_{\text{new}} - \mathbf{w}_{\text{old}}\|^2 + (b_{\text{new}} - b_{\text{old}})^2) \quad \text{such that} \quad y(\mathbf{w}_{\text{new}}^\top \mathbf{x} + b_{\text{new}}) = \mu.$$

That is, the goal is to find the smallest change in the weights so that the current example is on the right side of the margin. By substituting (a) and (b) from above into this optimization problem, we will get a single variable optimization problem whose solution gives us the η defined above. You can think of this algorithm as trying to tune the weight vector so that the current example is correctly classified right after the update.

2.3 Experiments

For all 5 settings above, you need to do the following things:

1. Run cross-validation for **ten epochs** for each hyper-parameter combination to get the best hyper-parameter setting. Note that for cases when you are exploring combinations of hyper-parameters (such as the margin Perceptron), you need to try out all combinations. Use the mean development accuracy across the 5 folds as the score for each hyper-parameter setting (do not take the best single fold).

2. Train the classifier for **20 epochs**. At the end of each training epoch, you should measure the accuracy of the classifier on the validation set (which is sometimes also called the development set). For the averaged Perceptron, use the average classifier to compute accuracy.

3. Use the classifier from the epoch where the development set accuracy is highest to evaluate on the test set.

2.4 What to report [25% of each variant points]

Your grade will be based on your code and your report that you upload in the Canvas. For each of the five variants, you need to report the following:

1. **Majority baseline:** Consider a classifier that always predicts the most frequent label. What is its accuracy on the test and development set?
2. For each variant, you need to report:
   (a) Briefly describe the design decisions that you have made in your implementation. (E.g, what programming language, how do you represent the vectors, etc.)
   (b) The best hyper-parameters
   (c) The cross-validation accuracy for the best hyper-parameter
   (d) The total number of updates the learning algorithm performs on the training set
   (e) Development set accuracy
   (f) Test set accuracy
   (g) You can plot a *learning curve* where the x-axis is the epoch id and the y-axis is the dev set accuracy using the classifier (or the averaged classifier, as appropriate) at the end of that epoch. Or You can choose to list them (the epoch id and the dev set accuracy). Note that you should have selected the number of epochs using the learning curve (but no more than 20 epochs).

2.5 Submission Instructions

If your code is totally wrong or there is no code in your file and/or you don't give us the results in username_userID_answers.pdf or you give us a partial of the username_userID_answers.zip, then you will get 0 in the related question.

You will submit two components on Canvas:

- **Report:** Write your report inside the pdf, the name is: username_userID_answers.pdf
- **Code:** Submit the code file inside the zip file we give you: username_userID_answers.zip