Assignment 2

CMSC 678 — Introduction to Machine Learning

Due Friday March 15, 11:59 PM

Item	**Summary**
Assigned	Monday March 4
Due	Friday March 15
Topic	Loss Functions and Multiclass Classification (I)
Points	100

In this assignment you will become comfortable with the math of loss-based optimization; and implement, experiment with, and compare multiple types of multiclass classifiers.

You are to *complete* this assignment on your own: that is, the code and writeup you submit must be entirely your own. However, you may discuss the assignment at a high level with other students or on the discussion board. Note at the top of your assignment who you discussed this with or what resources you used (beyond course staff, any course materials, or public Discord discussions).

The following table gives the overall point breakdown for this assignment.

	Theory		App.: Classification		
Question	1	2	3	4	5
Points	20	25	5	35	15

What To Turn In Turn in a **PDF** writeup that answers the questions; turn in all requested code **and models** necessary to replicate your results. Turn in models by serializing the learned models and parameters; your programming language likely natively supports this (e.g., Python's `pickle`, or Pytorch's own format). Be sure to include specific instructions on how to build/compile (if necessary) and run your code. Answer the following questions in long-form. Provide any necessary analyses and discussion of your results.

How To Submit Submit the assignment on the submission site under "Assignment 2."

https://www.csee.umbc.edu/courses/graduate/678/spring24/submit.

Be sure to select "`Assignment 2`."

1. (**20 points**) For $1 \leq i \leq N$, let $x_i \in \mathbb{R}^2$ be a two-dimensional input, and $y_i \in \{-1, +1\}$ the binary label for x_i.

   (a) Describe, in both words and with a concrete example, when the perceptron algorithm would be guaranteed to converge (find an optimal solution to the training set). For the concrete example, give $N \geq 5$ data points $\{(x_i, y_i)\}$ for which the perceptron algorithm would converge.

   (b) Describe, in words and with a concrete example, when the perceptron algorithm would not be guaranteed to converge. For the concrete example, give $N \geq 5$ data points $\{(x_i, y_i)\}$ for which the perceptron algorithm would not converge.

   (c) Let the dataset $\mathcal{D}$ be

$i =$	x_i	y_i
1	(-1, 1)	1
2	(-1, -1)	-1
3	(0.5, 0.5)	1
4	(1, -1)	1
5	(0.5, -1)	-1

   Run, by hand, the perceptron algorithm on these five data points. Record your computations for each time step (data point examined, activation score for that data point and current weight vector, the predicted value, and the new weights) in the following table.

$t =$	Weights $w^{(t)}$	Data point $x_{(t\%N)+1}$	Activation score	Predicted value $\hat{y}$	New weights $w^{(t+1)}$
0	$(0, 0, \ldots, 0)$				
1					
2					
3					
4					
5					
6					

   For $t = 0$, please fix $w^{(t)}$ to have the correct dimensionality. The strange indexing $x_{(t\%N)+1}$ simply says to iterate through the data in the order provided above. Note the second column of each row should equal the last column of the previous row.

2. (**25 points**)

   (a) Discuss how the perceptron algorithm fits within the ERM framework. Be specific in saying how the two requirements of decision theory discussed in class are formulated, and be sure to discuss how an initial difficulty in working through the ERM framework is overcome.

   (b) Consider a C class classification problem, and let z be a C dimensional vector of scores that your model computes (one score per class). Consider the `nn.MultiMarginLoss` defined in Pytorch:

$$\ell(\text{scores } z, \text{target } y) = \frac{1}{C} \sum_{i \neq y} \max\left(0, 1 - z_y + z_i\right) \qquad \text{[Eq-1]}$$

   (Here, `margin`=1 and p=1.) First, describe in words what this is computing, Then, discuss how this generalizes the hinge loss that we covered in class.

   Hint: Check out the documentation for this function and try playing around with it! For example, for `z=torch.tensor([[0.1, 0.2, 0.3]])`, what values are computed if the class associated with the first coordinate (`y=0`) is the correct class? What are the values if instead the class associated with the second coordinate (`y=1`) is the correct class?

The next three questions lead you through building, experimenting with, reporting on the performance of, and analyzing a multiclass perceptron classifier. The **core deliverables** for these questions are:

1. any implementations, scripts, and serialized model files needed to compile and run your code; and
2. a written report discussing
   - your implementations, including any design decisions or difficulties you experienced [from questions 3-4];
   - evidence and analysis of internal development/experimentation on the perceptron model on the *development* set [from question 4]; and
   - results and analysis of a full comparison on the *test* set [from 5].

The data we'll be using is the MNIST digit dataset: in total, there are 70,000 "images" of handwritten digits (each between 0-9). The task is to predict the digit $y \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ from a 28x28 input grayscale image x. 60,000 of these are allocated for training and 10,000 are allocated for testing. Do **not** use the 10,000 testing portion until question 5.

Get the data from `https://www.csee.umbc.edu/courses/graduate/678/spring24/materials/mnist_rowmajor.jsonl.gz`. This is a gzipped file, where each line is a JSON object. Each line has three keys: "image" (a row-major representation of each image), "label" (an int, 0-9), and "split" (train or test). Each image x is represented as a 784 ($= 28 \times 28$) one-dimensional array (row-major refers to how each image x should be interpreted in memory).

First, get acquainted with the data: make sure that you can read it properly and you are sufficiently familiar with the storage format so you can easily use these files in later questions.[1] Second, you are to split the 60,000 training set into an internal training set (`int-train`) and an internal development set (`int-dev`). How you split (e.g., randomly, stratified sampling, taking the first M), and the resulting sizes of `int-train` and `int-dev` are up to you.

3. (**5 points**) Implement a "most frequent" baseline classifier. This baseline identifies the label that was most common in training (`int-train`) and always produces that label during evaluation (regardless of the input).
   (a) In your **report**, document the implementation (this should be fairly straight-forward).
   (b) In your **report**, report how well this baseline does on `int-dev`.
4. (**35 points**) Implement a multiclass perceptron. Train it on `int-train` and evaluate it on `int-dev`. For this question, you may use computation accelerators (e.g., blas, numpy, MATLAB) but you may not use any existing perceptron implementation. Further, you may implement either as part of an ERM

[1]Notice that each component of x is a float ($0 \le x_i \le 1$). If you want to visualize the images, you can apply the sign function (with $\tau = 0$) and print its output in a 28x28 grid

$$\text{sign}_\tau(x_i) = \begin{cases} 1 & x_i > \tau \\ 0 & x_i \le \tau. \end{cases} \qquad \text{[Eq-2]}$$

Note that [Eq-2] provides a **binary** representation of the input x.

You do *not* have to do the following but you may find it helps you get a handle on the data: visualize the representations (raw/original, binary, or with arbitrary τ) by creating a heatmap/tile plot. This way you can "look" at your data and verify that the labels line up to their respective images (and make sense!).

(optimization) framework using `nn.MultiMarginLoss` or as a generalization of the "looping" algorithm discussed in class.

(a) In your **report**, discuss your implementation and convergence criteria (this should also be fairly brief).

(b) In your **report**, discuss the concrete, quantifiable ways you validated your implementation was correct. One such way could be to run it on the toy data in Question 1, though there are other ways too.

(c) In your **report**, experiment with at least three different configurations, where you train each configuration on `int-train` and then evaluate on `int-dev`. Document this internal development progress through quantifiable and readable means, i.e., graphs and/or tables showing how different model configurations perform on `int-dev`. A configuration could include learning biases or not, the convergence criteria, or the input featurization, among others.

Implementation Hint if you decide to implement a generalization of the "looping" algorithm: Training a multiclass perceptron follows the training for a binary perceptron as discussed in class, with the following four changes: first, rather than there being a single weight vector $\mathbf{w}$, there is now a weight vector $\mathbf{w}_j$ per class j. Second, a single prediction is obtained from the maximum of a vector of predictions $\vec{y}$, using each of the class-specific weight vectors. That is, $\vec{y}[j] = \mathbf{w}_j^\intercal x$ and the predicted value $y = \arg\max_j \vec{y}[j]$. Third, if a prediction is incorrect, the correct weights $\mathbf{w}_{y^\star}$ are increased by x, and the mispredicted weights $\mathbf{w}_y$ are decreased by x. Fourth, a per-class bias replaces the single bias term in the binary perceptron algorithm.

5. (**15 points**) Using the best perceptron configuration found in the previous question, train new a perceptron model on the entire, original `train` set; also "re-train" a most-frequent classifier baseline. At the end of this training, you should have two models trained on the entire 60,000 `train`ing set. Evaluate these models on the original 10,000 image `test` set. Do not experiment with different configuration values here (that's what your `int-dev` set was for!). Include these evaluation results in your **report** and discuss the results: were they surprising? How similar were the test results to the best internal development results?