CM146, Fall 2020
Problem Set 3: VC Dimension and Neural Networks
Due Nov. 30 at 11:59 pm

Submission Instructions

- Submit your solutions electronically on the course Gradescope site as PDF files.
- If you plan to typeset your solutions, please use the LaTeX solution template. If you must submit scanned handwritten solutions, please use a black pen on blank white paper and a high-quality scanner app.
- For questions involving math and derivation, please provide important intermediate steps and box the final answer clearly.
- You are required to submit the code only for Question 3 - Digit Recognizer on CCLE. For the sub-questions in Question 3 requiring you to complete a piece of code, you are expected to copy-paste your code as a part of the solution in the submission pdf to receive full credit.

Part of this assignment is adapted from the course materials by Hsuan-Tien Lin (National Taiwan University).

1 VC Dimension [16 pts]

For the following problems, we classify a point x to either a positive label $+1$ or a negative label -1 by a hypothesis set $\mathcal{H}$.

(a) *Positive ray classifier.* Consider $x \in \mathbb{R}$ and $\mathcal{H} = \{sign(x - b) \mid b \in \mathbb{R}\}$. That is, the label is $+1$ if x is greater than b otherwise -1. [8 pts]

i. For N points, prove that there are at most $N + 1$ label outcomes that can be generated by $\mathcal{H}$. For example, when we have 4 points, $x_1 \leq x_2 \leq x_3 \leq x_4$, there are at most 5 label outcomes can be generated by $\mathcal{H}$, as shown by the following.

x_1	x_2	x_3	x_4
$+1$	$+1$	$+1$	$+1$
-1	$+1$	$+1$	$+1$
-1	-1	$+1$	$+1$
-1	-1	-1	$+1$
-1	-1	-1	-1

ii. What is the VC dimension of $\mathcal{H}$?

(b) *Positive interval classifier.* Consider $x \in \mathbb{R}$ and $\mathcal{H} = \{sign(\mathbb{1}(x \in [a, b]) - 0.5) \mid a, b \in \mathbb{R}, a \leq b\}$, where $\mathbb{1}(.)$ is the indicator function. That is, the label is $+1$ if x in the interval $[a, b]$ otherwise -1. [8 pts]

i. For N points, prove that there are at most $(\frac{N^2+N}{2} + 1)$ label outcomes that can be generated by $\mathcal{H}$.

ii. What is the VC dimension of $\mathcal{H}$?

2 Bound of VC dimension [16 pts]

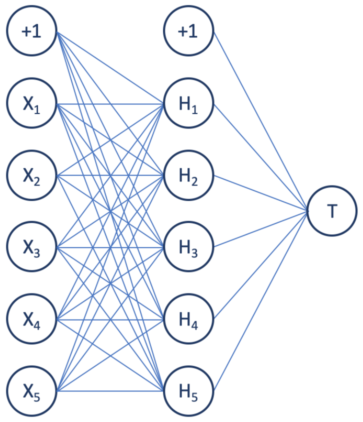
Assume the VC dimension of an empty set is zero. Now, we have two hypothesis sets $\mathcal{H}_1$ and $\mathcal{H}_2$.

(a) Let $\mathcal{H}_3 = \mathcal{H}_1 \cap \mathcal{H}_2$. Show that $VC(\mathcal{H}_1) \geq VC(\mathcal{H}_3)$. [6 pts]

(b) Let $\mathcal{H}_3 = \mathcal{H}_1 \cup \mathcal{H}_2$. Give an example to show that $VC(\mathcal{H}_1) + VC(\mathcal{H}_2) < VC(\mathcal{H}_3)$ is possible. [10 pts]

3 Neural Network [20 pts]

We design a neural network to implement the XOR operation of X_1, X_2, X_3, X_4, X_5. We use $+1$ to represent `true` and -1 to represent `false`. Consider the following neural network.



We use w_{ij} to represent the weight between X_i and H_j, and use w_{0j} to represent the weight between the first layer bias term $(+1)$ and H_j. We use v_i to represent the weight between H_i and T, and use v_0 to represent the weight between the second layer bias term $(+1)$ and T.

Now, let $X_i \in \{+1, -1\}$, $H_j = sign\left(\sum_{i=0}^{5} w_{ij} X_i\right)$, and $T = sign\left(\sum_{i=0}^{5} v_i H_i\right)$.

(a) Specify w_{ij} such that H_j is $+1$ if there are at least j positive values among X_1, X_2, X_3, X_4, X_5, otherwise -1. If there are multiple acceptable weights, you only need to write down one of them. [8 pts]

(b) Given w_{ij} and H_j defined as above, specify v_i such that the whole neural network behaves like the XOR operation of X_1, X_2, X_3, X_4, X_5. If there are multiple acceptable weights, you only need to write down one of them. [8 pts]

(c) Justify why the output of the neural network behaves like the XOR operation of X_1, X_2, X_3, X_4, X_5. [4 pts]

4 Implementation: Digit Recognizer [48 pts]

In this exercise, you will implement a digit recognizer in pytorch. Our data contains pairs of 28×28 images $\mathbf{x}_n$ and the corresponding digit labels $y_n \in \{0, 1, 2\}$. For simplicity, we view a 28×28 image $\mathbf{x}_n$ as a 784-dimensional vector by concatenating the row pixels. In other words, $\mathbf{x}_n \in \mathbb{R}^{784}$. Your goal is to implement two digit recognizers (`OneLayerNetwork` and `TwoLayerNetwork`) and compare their performances.

---

code and data

- code : `Fall2020-CS146-HW3.ipynb`
- data : `hw3_train.csv`, `hw3_valid.csv`, `hw3_test.csv`

Please use your *@g.ucla.edu* email id to access the code and data. Similar to *HW-1*, copy the colab notebook to your drive and make the changes. Mount the drive appropriately and copy the shared data folder to your drive to access via colab. For colab usage demo, check out the Discussion recordings for Week 2 in CCLE. The notebook has marked blocks where you need to code.

### ========= TODO : START ========= ###

### ========= TODO : END ========== ###

Note: For the questions requiring you to complete a piece of code, you are expected to copy-paste your code as a part of the solution in the submission pdf. Tip: If you are using LaTeX, check out the Minted package (example) for code highlighting.

Data Visualization and Preparation [10 pts]

(a) Randomly select three training examples with *different labels* and print out the images by using `plot_img` function. Include those images in your report. [2 pts]

(b) The loaded examples are numpy arrays. Convert the numpy arrays to tensors. [3 pts]

(c) Prepare `train_loader`, `valid_loader`, and `test_loader` by using `TensorDataset` and `DataLoader`. We expect to get a batch of pairs $(\mathbf{x}_n, y_n)$ from the dataloader. Please set the batch size to 10. [5 pts]

You can refer `https://pytorch.org/docs/stable/data.html` for more information about `TensorDataset` and `DataLoader`.

One-Layer Network [15 pts]

For one-layer network, we consider a *784–3* network. In other words, we learn a 784×3 weight matrix $\mathbf{W}$. Given a $\mathbf{x}_n$, we can compute the probability vector $\mathbf{p}_n = \sigma(\mathbf{W}^\top \mathbf{x}_n)$, where $\sigma(.)$ is the element-wise sigmoid function and $\mathbf{p}_{n,c}$ denotes the probability of class c. Then, we focus on the *cross entropy loss*

$$-\sum_{n=0}^{N}\sum_{c=0}^{C} \mathbb{1}(c = y_n)\log(\mathbf{p}_{n,c})$$

where N is the number of examples, C is the number of classes, and $\mathbb{1}$ is the indicator function.

(d) Implement the constructor of `OneLayerNetwork` with `torch.nn.Linear` and implement the `forward` function to compute the outputs of the single fully connected layer i.e. $\mathbf{W}^\top \mathbf{x}_n$. Notice that we do not compute the sigmoid function here since we will use `torch.nn.CrossEntropyLoss` later. [5 pts]

You can refer to `https://pytorch.org/docs/stable/generated/torch.nn.Linear.html` for more information about `torch.nn.Linear` and refer to `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html` for more information about using `torch.nn.CrossEntropyLoss`.

(e) Create an instance of `OneLayerNetwork`, set up a criterion with `torch.nn.CrossEntropyLoss`, and set up a SGD optimizer with learning rate 0.0005 by using `torch.optim.SGD` [2 pts]

You can refer to `https://pytorch.org/docs/stable/optim.html` for more information about `torch.optim.SGD`.

(f) Implement the training process. This includes forward pass, initializing gradients to zeros, computing loss, loss backward, and updating model parameters. If you implement everything correctly, after running the `train` function in main, you should get results similar to the following. [8 pts]

```
Start training OneLayerNetwork...
| epoch  1 | train loss 1.075387 | train acc 0.453333 | valid loss ...
| epoch  2 | train loss 1.021301 | train acc 0.563333 | valid loss ...
| epoch  3 | train loss 0.972599 | train acc 0.630000 | valid loss ...
| epoch  4 | train loss 0.928335 | train acc 0.710000 | valid loss ...
...
```

Two-Layer Network [7 pts]

For two-layer network, we consider a *784–400–3* network. In other words, the first layer will consist of a fully connected layer with 784×400 weight matrix $\mathbf{W}_1$ and a second layer consisting of 400×3 weight matrix $\mathbf{W}_2$. Given a $\mathbf{x}_n$, we can compute the probability vector $\mathbf{p}_n = \sigma(\mathbf{W}_2^\top \sigma(\mathbf{W}_1^\top \mathbf{x}_n))$, where $\sigma(.)$ is the element-wise sigmoid function. Again, we focus on the *cross entropy loss*, hence the network will impelement $\mathbf{W}_2^\top \sigma(\mathbf{W}_1^\top \mathbf{x}_n)$ (note the outer sigmoid will be taken care of implicitly in our loss).

(g) Implement the constructor of `TwoLayerNetwork` with `torch.nn.Linear` and implement the `forward` function to compute $\mathbf{W}_2^\top \sigma(\mathbf{W}_1^\top \mathbf{x}_n)$. [5 pts]

(h) Create an instance of `TwoLayerNetwork`, set up a criterion with `torch.nn.CrossEntropyLoss`, and set up a SGD optimizer with learning rate 0.0005 by using `torch.optim.SGD`. Then train `TwoLayerNetwork`. [2 pts]

Performance Comparison [16 pts]

(i) Generate a plot depicting how `one_train_loss`, `one_valid_loss`, `two_train_loss`, `two_valid_loss` varies with epochs. Include the plot in the report and describe your findings. [3 pts]

(j) Generate a plot depicting how `one_train_acc`, `one_valid_acc`, `two_train_acc`, `two_valid_acc` varies with epochs. Include the plot in the report and describe your findings. [3 pts]

(k) Calculate and report the test accuracy of both the one-layer network and the two-layer network. Explain why we get such results. [3 pts]

(l) Replace the SGD optimizer with the Adam optimizer and do the experiments again. Show the loss figure, the accuracy figure, and the test accuracy. Include the figures in the report and describe your findings. [7 pts]

You can refer to https://pytorch.org/docs/stable/optim.html for more information about `torch.optim.Adam`.

Submission instructions for programming problems

- Please export the notebook to a `.py` file by clicking the "File" → "Download.py" and upload to CCLE.

  Your code should be commented appropriately. The most important things:
  - Your name and the assignment number should be at the top of each file.
  - Each class and method should have an appropriate doctsring.
  - If anything is complicated, it should include some comments.

  There are many possible ways to approach the programming portion of this assignment, which makes code style and comments very important so that staff can understand what you did. For this reason, you will lose points for poorly commented or poorly organized code.

- Please submit all the plots and the rest of the solutions (other than codes) to Gradescope