CM146, Winter 2021
Problem Set 3: Deep learning, Learning theory, Kernels
Due February 26,2021, 11:59pm

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.

1 VC-Dimension [8 pts]

(a) Consider the space of instances X corresponding to all points in the x, y plane. What is the VC-dimension of the hypothesis space defined by H_c = circles in the x, y plane, with points inside the circle are classified as positive examples? Justify your answer (e.g. with one or more diagrams).

(b) This problem investigates a few properties of the VC dimension, mostly relating to how $VC(H)$ increases as the set H increases. For each part of this problem, you should state whether the given statement is true, and justify your answer with either a formal proof or a counter-example.

i. Let two hypothesis classes H_1 and H_2 satisfy $H_1 \subseteq H_2$. Prove or disprove: $VC(H_1) \leq VC(H_2)$.

ii. Let $H_1 = H_2 \cup H_3$. Prove or disprove: $VC(H_1) \leq VC(H_2) + VC(H_3)$.

2 Kernels [8 pts]

(a) For any two documents $\boldsymbol{x}$ and $\boldsymbol{z}$, define $k(\boldsymbol{x}, \boldsymbol{z})$ to equal the number of unique words that occur in both $\boldsymbol{x}$ and $\boldsymbol{z}$ (i.e., the size of the intersection of the sets of words in the two documents). Is this function a kernel? Give justification for your answer.

(b) One way to construct kernels is to build them from simpler ones. We have seen various "construction rules", including the following: Assuming $k_1(\boldsymbol{x}, \boldsymbol{z})$ and $k_2(\boldsymbol{x}, \boldsymbol{z})$ are kernels, then so are

- (scaling) $f(\boldsymbol{x})k_1(\boldsymbol{x}, \boldsymbol{z})f(\boldsymbol{z})$ for any function $f(\boldsymbol{x}) \in \mathbb{R}$

Parts of this assignment are adapted from course material by Tommi Jaakola (MIT), and Andrew Ng (Stanford), and Jenna Wiens (UMich).

- (sum) $k(\boldsymbol{x}, \boldsymbol{z}) = k_1(\boldsymbol{x}, \boldsymbol{z}) + k_2(\boldsymbol{x}, \boldsymbol{z})$
- (product) $k(\boldsymbol{x}, \boldsymbol{z}) = k_1(\boldsymbol{x}, \boldsymbol{z}) k_2(\boldsymbol{x}, \boldsymbol{z})$

Using the above rules and the fact that $k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x} \cdot \boldsymbol{z}$ is (clearly) a kernel, show that the following is also a kernel:

$$\left(1 + \left(\frac{\boldsymbol{x}}{||\boldsymbol{x}||}\right) \cdot \left(\frac{\boldsymbol{z}}{||\boldsymbol{z}||}\right)\right)^3$$

(c) Given vectors $\boldsymbol{x}$ and $\boldsymbol{z}$ in $\mathbb{R}^2$, define the kernel $k_\beta(\boldsymbol{x}, \boldsymbol{z}) = (1 + \beta \boldsymbol{x} \cdot \boldsymbol{z})^3$ for any value $\beta > 0$. Find the corresponding feature map $\phi_\beta(\cdot)$[1]. What are the similarities/differences from the kernel $k(\boldsymbol{x}, \boldsymbol{z}) = (1 + \boldsymbol{x} \cdot \boldsymbol{z})^3$, and what role does the parameter β play?

3 SVM [8 pts]

Suppose we are looking for a maximum-margin linear classifier *through the origin*, i.e. $b = 0$ (also hard margin, i.e., no slack variables). In other words, we minimize $\frac{1}{2}||\boldsymbol{\theta}||^2$ subject to $y_n \boldsymbol{\theta}^T \boldsymbol{x}_n \geq 1, n = 1, \ldots, N$.

(a) Given a single training vector $\boldsymbol{x} = (a, e)^T$ with label $y = -1$, what is the $\boldsymbol{\theta}^*$ that satisfies the above constrained minimization?

(b) Suppose we have two training examples, $\boldsymbol{x}_1 = (1, 1)^T$ and $\boldsymbol{x}_2 = (1, 0)^T$ with labels $y_1 = 1$ and $y_2 = -1$. What is $\boldsymbol{\theta}^*$ in this case, and what is the margin γ?

(c) Suppose we now allow the offset parameter b to be non-zero. How would the classifier and the margin change in the previous question? What are $(\boldsymbol{\theta}^*, b^*)$ and γ? Compare your solutions with and without offset.

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 : `CS146-Winter2021-PS3.ipynb`
- data : `ps3_train.csv`, `ps3_valid.csv`, `ps3_test.csv`

Please use your *@g.ucla.edu* email id to access the code and data. Similar to *PS1*, copy the colab notebook to your drive and make the changes. Mount the drive appropriately and copy the shared

[1]You may use any external program to expand the cubic.

data folder to your drive to access via colab. 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*

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

where N is the number of examples, C is the number of classes, and $\mathbf{1}$ 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. How can you improve the performance of the two-layer network ? [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`.