CS89/189: Deep Learning Generalization and Robustness

Winter 2024
Homework#1

January 10, 2024

Due: January 24, 2024, 11:59 pm ET

This problem set requires you to train neural networks and compute empirical measures for generalization. Your trained neural networks would enable computing training and validation performances as well as margins and different measures along with generalization bounds leveraging the measures. Section 9 lists this assignment's deliverables, including code and data. The code and data files must be submitted electronically via Canvas as a single zipped directory. The directory must not contain any sub-directories. The directory's name should be in the format 'First Middle Last HW1', where 'First,' 'Middle' (if it exists), and 'Last' match your student name on Canvas. Your zipped directory should therefore have the format 'First Middle Last HW1.zip'. You will receive points for correct implementations. There are no "improvement" points for this assignment. Note that you have until **11:59 pm ET on January 24, 2024** to turn in your submission. As you may remember, you are given a total of 4 free late days to be used for homework assignments. The late days that you use will be deducted from your total of 4 days. Once these 4 days are used up, Homework scores will be divided by 2 for each additional late day. Make sure to upload all of your codes and the base models to Canvas as per instructions. We cannot accept forgotten code or data files past the submission deadline without penalizing them for late days.

1 Overview

Deep learning models are becoming increasingly popular due to their ability to learn complex relationships between inputs and outputs. However, the generalizability and robustness of these models are an ongoing challenge that needs to be addressed.

Generalizability refers to the ability of a deep learning model to perform well on new, unseen data that was not used during model training. If the model is not able to generalize well, it is likely to overfit the training data and perform poorly on new data. To improve generalizability, techniques such as regularization, data augmentation, and early stopping can be used. Robustness, on the other hand, refers to the ability of a deep learning model to perform well in the presence of variations or perturbations in the input data. These variations could be due to, e.g., changes in lighting conditions, occlusions, or noise in the data. Deep learning models that are less robust are more prone to making errors when presented with such variations in the input data. Techniques such as adversarial training, data augmentation with perturbations, and model ensembling can be used to improve the robustness of deep-learning models.

Overall, improving the generalizability and robustness of deep learning models is crucial for their widespread adoption in real-world applications. Researchers and practitioners in the field often focus on developing new techniques to improve these aspects of deep learning models. In light of this, one measure commonly used to evaluate deep learning models is the norm of the network weights. Different types of norms, such as L1-norm and L2-norm, have been used in neural network training.

Research has shown that while L2-norm regularization is commonly used in neural networks, other norms can also achieve similarly good results. For example, it has been shown that L1-norm regularization can significantly reduce the number of non-zero parameters in a network without sacrificing performance. Another measure that has been used to evaluate the generalizability of neural networks is the margin of the network. The margin measures the distance between the decision boundary of the network and the closest data point. A larger margin indicates that the network is more robust to small changes in the input data. Research has shown that maximizing the margin can improve the generalization performance of a network.

Overall, there are a variety of measures and norms that can be used to evaluate the generalizability of neural networks, and different measures may be appropriate for different types of problems and datasets.

In this homework, you will construct and train a neural network with provided parameters. In the next step, you will compute the margin of the trained network. Finally, you will compute some of the measures/norms as well as generalization bounds known in the literature.

2 Data Pre-processing and Training Model

In the first phase, your task would be to load the CIFAR-10 dataset and normalize each channel with mean and standard deviation. The mean and std values of the CIFAR-10 dataset are provided in the *main.py* stub code file. You are also supposed to perform data augmentation. For data augmentation, please use random cropping and horizontal flipping.

Your next task is to define a 2-layer neural network and train the model. Your network would be a fully connected neural net with a single hidden layer and ReLU activation. You will use the following parameters to train your network:

- nunits = 1024 (hidden units)
- lr = 0.001 (learning rate)

- mt = 0.9 (momentum)
- batchsize = 64
- epochs = 25
- stopcond = 0.01
- nchannels, nclasses = 3, 10 (channels and class number for CIFAR-10)

You should define the loss function (criterion) and the optimizer (Stochastic gradient descent). Please report your Training loss, Training accuracy, and Validation accuracy. Also, save the model checkpoint as `model1.pt` and submit it.

3 Computing Margin

The notion of margin indicates the minimum distance to a decision boundary. Consider a c-class classification task where the label with the maximum output score will be selected as the prediction. The margin operator can be defined as below: $\mu : \mathbf{R}^c \times [c] \to \mathbf{R}$ as a function that gives the scores $f(x) \in \mathbf{R}^c$ for each label and the correct label $y \in [c]$, it returns the difference between the score of the correct label and the maximum score among other labels, i.e.

$$\mu(f(x), y) = f(x)[y] - \max_{i \neq y} f(x)[i].$$

Note that the margin is computed using the probability vector after softmax (i.e., output). Please return the 5th percentile of the empirical margin distribution computed on outputs from data points and report the returned margin of your trained model on the Training set and Validation set. Plot the charts of the validation loss and validation margin of each epoch.

We expect the network to provide a positive margin. However, is it always the case? Can you explain why sometimes the margin can be negative?

4 Computing Measures and Bounds

Deep neural networks are trained with specified parameters, and often these trained networks can be over-parameterized during the training process. Over-parameterization of neural networks impacts the generalizability of the network. That is, the size of the neural network or the number of learnable parameters can affect the difference between training performance and test-time performance. In this homework, we want to analyze the impact of over-parameterization on the generalizability property. Different existing research ensures the generalizability of neural networks with different measures. You will compute some of these measures on your trained network. Compute the following norms and bounds on your trained network.

- Compute the following measures and bounds on your trained network:
  - Frobenius norm $Fr_1 = \|\mathbf{W}_1\|_F$ of the weight matrix $\mathbf{W}_1$ in the first layer.
  - Frobenius distance measured as below:

    $$F_d = D_1 \cdot Fr_2,$$

    where F_d represents the Frobenius distance, $Fr_2 = \|\mathbf{W}_2\|_F$ is the Frobenius norm of the weight matrix in the second layer, and $D_1 = \|\mathbf{W}_1 - \mathbf{W}_{1,\text{init}}\|$ stands for the Euclidean distance between the final weights to the initial weights of the weight matrix in the first layer.

- Generalization bound (G) defined as follows:

$$G = \left(8\sqrt{C} \cdot Fr_1 \cdot Fr_2 \cdot \sqrt{data_{L2}}/V_m + 3\sqrt{\log(m/\delta)}\right)^2,$$

  where C is the number of classes (which is equal to the output dimension), $Fr_1 = \|\mathbf{W}_1\|_F$ and $Fr_2 = \|\mathbf{W}_2\|_F$ are Frobenius norms of the weight matrix in the first layer and second layer, respectively; $data_{L2}$ stands for the L2 norm squared of the data (computed in the provided stub code), V_m is the 5th percentile margin you computed in Section 3, m is the number of training samples, and δ is the probability of the bound not hold. Note that in our code, we set $\delta = 0.01$.

Report the Frobenius norm Fr_1, Frobenius distance, and the generalization bound G that you computed from your trained model.

5 Comparing Numerical Bounds

Now that you have trained your network with the parameters discussed in Section 2 and computed its generalization bound. In this step, you will train another 2-layer network on CIFAR-10 with the following parameters:

- nunits = 256 (hidden units)
- lr = 0.001 (learning rate)
- mt = 0.9 (momentum)
- batchsize = 64
- epochs = 25
- stopcond = 0.01
- nchannels, nclasses = 3, 10 (channels and class number for CIFAR-10)

Also, save the model checkpoint as `model2.pt`, and report Training loss, Training accuracy, Training margin, Validation accuracy, and plot the charts of each epoch's validation loss and validation margin. Now, compute the generalization bound and report the result.

Do you see any difference in the bounds you computed on your two NNs? If so, what do you think is the reason behind such a difference, and can you think of a scenario in which the difference can be minimized and why?

6 Size of the CNN Output Volume

Let us denote with:

- $N \times N \times V$ the size of the input volume;
- $n \times n \times V$ the size of the filters;
- Z the number of zeros padded at the top/bottom/left/right of the image;
- S the stride;
- V' the number of filters.

Prove the output volume will have size $M \times M \times V'$, where $M = \frac{N-n+2Z}{S} + 1$.

[Hint] Drawing the convolution operation with Z and S on it can help you get some intuition of proving the property.

7 Academic Integrity

This homework assignment must be done individually. Sharing code or model specifications is strictly prohibited. Homework discussions are allowed only on Piazza, according to the policy outlined on the course web page: https://canvas.dartmouth.edu/courses/63219. You are not allowed to search online for auxiliary software, reference models, architecture specifications, or additional data to solve the homework assignment. Your submission must be entirely your own work. That is, the code and the answers that you submit must be created, typed, and documented by you alone, based exclusively on the materials discussed in class, and released with the homework assignment. You can obviously consult the class slides posted in Canvas, your lecture notes, and the textbook. Important: the models you will submit for this homework assignment must be trained exclusively on the specified data provided with this assignment. You can, of course, play with other datasets in your spare time. These rules will be strictly enforced, and any violation will be treated seriously.

8 Report

Your report must contain the following contents, and you can add more as you wish.

1. Your name
2. The results you obtain on each model (i.e., loss, accuracy, margin, charts, measures, and bounds) and your understanding of the bounds of the two NNs you trained.
3. Explain why sometimes the margin can be negative.
4. Your proof of the CNN output volume formula.

9 Submission Instructions

Please compress '`First Middle Last HW1/`' into one zip file with the name '`First Middle Last HW1.zip`' before uploading it to Canvas. The directory contains your codes, two model checkpoints, and the report. As listed below:

1. `main.py`: your code;
2. `model1.pt`: your first model checkpoint;
3. `model2.pt`: your second model checkpoint;
4. `report.pdf`: your report.

10 Grading

This homework will be graded based on (1) program correctness (**60 points**) and (2) report (**40 points**). Please check these items before your submission. We will test your code and models to reproduce the results. Therefore, please make sure to implement your code in the specified area and follow the submission instructions. You can use `grading.py` to evaluate your code and program; the expected results can be found in the following screenshot.

```
(test) ➜  hw1 python grading.py --dataset ./datasets --nunits 256 --checkpoint model2.pt
Files already downloaded and verified
Files already downloaded and verified
================== Summary ==================
 Training loss: 1.959   Training margin -0.988    Training accuracy: 0.516   Validation accuracy: 0.482

Frobenius1:      10.1
Frobenius2:      4.35
Distance1:       4.07
Distance2:       2.89
Spectral1:       1.53
Spectral2:       1.83
Fro_Fro:         44.0
L1max_L1max:     7.2e+02
Spec_Dist:       14.0
Dist_Spec:       1.19e+02
Spec_Dist_sum:   1.33e+02
Spec_L1max:      13.8
L1max_Spec:      1.16e+02
Spec_L1max_sum:  1.3e+02
Dist_Fro:        17.7
#parameter:      7.89e+05
VC bound:        2.03e+09
L1max bound:     1.19e+10
Your bound:      4.85e+09
```

11 FAQs

- Please use the default values for random cropping (`RandomResizedCrop(size=32)`) and horizontal flipping (`RandomHorizontalFlip(p=0.5)`) while performing data augmentation.
- To calculate the size of the CNN output volume, you are required to derive the proof in general instead of using explicit architecture parameters.