EECS 182 Deep Neural Networks Fall 2022 Anant Sahai Homework 5

This homework is due on Saturday, October 15, 2022, at 10:59PM.

1. Understanding Dropout (Coding Question)

In this problem, you will fill out the dropout.ipynb notebook, which analyzes the effect of dropout in a simplified setting.

(A) Analytical gradient descent solution without dropout: **Fill out notebook section (A).**

(B) Empirical gradient descent solution without dropout. **Fill out notebook section (B).**

(C) Analytical gradient descent solution with dropout: **Fill out notebook section (C).**

(D) Empirical gradient descent solution without dropout. **Fill out notebook section (D)**

(E) Analyzing dropout's effect on the loss curve. **Run the cells in part (E), and explain the results. (Why is the loss curve so spiky, and why does the larger batch size help with matching the analytical solution?)** **Solution:** With batch-size 1, we alternate between really high and low loss for each data point depending on which elements are masked out. The worst loss is when both data points are dropped out, and this can't be reduced. The losses on the other data points are still nonzero since the network can't fit all equations simultaneously. Convergence is slow since each new datapoint tugs the weights in a different direction. With a larger batch size, we take the approximate average of the loss over the four datapoints, which means the gradient for each batch points in a similar direction, making it easier to converge (though loss at convergence is still nonzero).

(F) Refer back to the cells you ran in part (E). **Analyze how and why adding dropout changes the following: (a) How large were the final weights w_1 and w_2 compared to each other. (b) How large the contribution of each term (i.e. $10w_1$ + w_2) is to the final output. Why does this change occur?** (This does not need to be a formal math proof). **Solution:** Without dropout, w_1 is 10x larger than w_2 and contributes 100x more to the output. This occurs because the update step for w_1 is always 10x larger than the update step for w_2. With dropout, w_2 is approximately 10x larger than w_1 and the contributions to the final output are approximately the same. The reason this is happening is as follows. We have 3 different potential cases with dropout: one where w_1 is dropped, one where w_2 is dropped and one where neither are dropped. When w1 is dropped, gradient descent is pulling w_2 towards 11. When w_2 is dropped, gradient descent is pulling w_1 towards 1.1. However, these are in conflict with the case where nothing is dropped out, since if $w_1 = 1$ and $w_2 = 10$, our output is 20 and our mean squared error is huge (81). This third case pulls w_1 and w_2 down by gradient descent, and specifically pulls w_1 down much more than w_2 since small changes in w_1 get multiplied by a factor of 10 and result in a much larger decrease in our error. These three conflicting objectives settle in the middle of the analytic solution and this dropped out loss when w_1 being smaller and w_2 being larger than the no-dropout solution due to case 3 pulling w_1 down 10 times harder.

(G) (Optional) Sweeping over the dropout rate **Fill out notebook section (G).** You should see that as the dropout rate changes, w_1 and w_2 change smoothly, except for a discontinuity when dropout rates are 0. Explain this discontinuity. **Solution:** When dropout rates are positive, there is a unique loss-minimizing solution. Achieving it involves making w_2 larger than w_1. When dropout rates are zero, there are many loss-minimizing solutions, and the network chooses the norm-minimizing one, which makes w_1 larger than w_2.

(H) (Optional) Optimizing with Adam: Run the cells in part (H). **Does the solution change when you switch from SGD to Adam? Why or why not?** **Solution:** The solution with dropout = .5 doesn't change since there's a unique loss-minimizing solution. The solution without dropout changes since there are many loss-minimizing solutions, and Adam's adaptive learning rate makes it converge to the solution where w_1 and w_2 are approximately equal instead of SGD's norm-minimizing solution.

(I) Dropout on real data: **Run the notebook cells in part (G), and report on how they affect the final performance.** **Solution:** Dropout increases the test accuracy on clean data, and the network learns to rely a bit less on the cheating feature (though is still relies heavily on it)

2. Inductive Bias and Systematic Experimentation

In this problem, you will fill out the EdgeDetection.ipynb notebook. We will learn 1) inductive bias of CNN and 2) systematic experimentation

(A) Overfitting Models to Small Dataset: **Fill out notebook section (Q1).**

(a) Can you find any interesting patterns in the learned filters? **Solution:** Somewhat looks like edge detection filters. (Every reasonable answer is correct)

(B) Sweeping the Number of Training Images: **Fill out notebook section (Q2).**

(a) Compare the learned kernels, untrained kernels, and edge-detector kernels. What do you observe? **Solution:** When CNN's performance is low ($<90\%$), Kernels look almost random. Interestingly, with the dumb luck of initialization, there are some kernels that already look like edge detectors. When CNN's performance is high ($>90\%$), Kernels look like edge detectors. (Every reasonable answer is correct)

(b) We freeze the convolutional layer and train only final layer (classifier). In a high data regime, the performance of CNN initialized with edge detectors is worse than CNN initialized with random weights. Why do you think this happens? **Solution:** As mentioned earlier, inductive bias benefits both performance and training efficiency. But not always that's the case. The training result can be worse if we inject too much into the training process/model or incorrectly. We can understand it through the lens of bias-variance tradeoff. If we introduce inductive biases to the model or training process, it increases bias while decreasing variance. But note that a large number of data also reduce the variance. The edge detection problem is so simple that 50 images per class are large enough to decrease the decent amount of variance. But we inject too many inductive biases into the model: initializing with edge detection filters and freezing them. Therefore, the bias term dominates the generalization error in the high data regime. For more information, please refer to this paper Battaglia et al. (2018)

(C) Checking the Training Procedures: **Fill out notebook section (Q3).**

(a) List every epochs that you trained the model. Final accuracy of CNN should be at least 90% for 20 images per class. **Solution:** There is no definite answer in this question. If you achieve 90% for 20 images per class with CNN model, your answer is correct

(b) Check the learned kernels. What do you observe? **Solution:** Learned kernels look like edge detectors. (Every reasonable answer is correct)

(c) (optional) You might find that with the high number of epochs, validation loss of MLP is increasing whild validation accuracy increasing. How can we interpret this? **Solution:** As the model overfits to training set, the model becomes overconfident.

(d) (optional) Do hyperparameter tuning. And list the best hyperparameter setting that you found and report the final accuracy of CNN and MLP. **Solution:** There is no definite answer in this question. If you find any hyperparameter configuration that performs better than the given, your answer is correct.

(e) How much more data is needed for MLP to get a competitive performance with CNN? Does MLP really generalize or memorize? **Solution:** Any number you found is correct if the validation accuracy of MLP is similar to that of CNN. But you will find that MLP actually memorize the dataset. Considering the data generating process, we can find four significant variables for this dataset: 1) edge location, 2) edge width, 3) edge intensity and 4) background intensity. The first two variables mainly determine the patterns of images. As you can see in the data generation code, we randomly sample from [1, 2, 3, 4, 5] as the edge width and [1 $\sim$ 28] as the edge location for each edge. Hence there are 280 possible cases. CNN achieves almost 100% validation accuracy with about 50 training images, but MLP needs a much larger number of data points to achieve matched validation accuracy.

(D) Domain Shift between Training and Validation Set: **Fill out notebook section (Q4).**

(a) Why do you think the confusion matrix looks like this? Why CNN misclassifies the images with edge to the images with no edge? Why MLP misclassifies the images with vertical edge to the images with horizontal edge and vice versa? (Hint: Visualize some of the images in the training and validation set.) **Solution:** We can find that both models are overfitting to the training set. However, the patterns that they overfit are pretty different. CNN overfits to the edges that are located in the left half or upper half of the image. Therefore, it is more likely to classify validatation set with edges that are located in the right half or lower half of the image to the images with no edge. However, MLP overfits to the edges that are located in the fraction of edges that are located in the left upper part and right lower part of the image. Therefore, MLP classifies the images with vertical edge to the images with horizontal edge and vice versa.

(b) Why do you think MLP fails to learn the task while CNN can learn the task? (Hint: Think about the model architecture.) **Solution:** CNN is translational invariant. Therefore, CNN can learn the task even though the edges are located in the different half of the image. However, MLP is not translational invariant. Therefore, MLP fails to learn the task.

(E) When CNN is Worse than MLP: **Fill out notebook section (Q5).**

(a) What do you observe? What is the reason that CNN is worse than MLP? (Hint: Think about the model architecture.) **Solution:** CNN utilizes the local correlation of the image. However, the local correlation is destroyed by the random permutation. Therefore, CNN fails to learn the task. However, MLP is not affected by the random permutation because it is 'fully connected'. Therefore, MLP can learn the task.

(b) Assuming we are increasing kernel size of CNN. Does the validation accuracy increase or decrease? Why? **Solution:** The validation accuracy will increase. This is because as kernel gets larger, convolutional layer becomes more 'fully connected'. Therefore, CNN can learn the task even though the local correlation is destroyed by the random permutation.

(c) How do the learned kernels look like? Explain why. **Solution:** The learned kernels are not that different from the randomly initialized kernels. This is because CNN cannot capture the local correlation of the image. Recalling backpropagation of CNN, CNN kernels are optimized in the way that maximizes the correlation of grids of images. (Any reasonable answer is also correct)

(F) Increasing the Number of Classes: **Fill out notebook section (Q6).**

3. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

(c) **Roughly how many total hours did you work on this homework? Write it down here where you'll need to remember it for the self-grade form.**

References

Battaglia, P., Hamrick, J. B. C., Bapst, V., Sanchez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G. E., Vaswani, A., Allen, K., Nash, C., Langston, V. J., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018. URL `https://arxiv.org/pdf/1806.01261.pdf`.

Contributors:

- Olivia Watkins.
- Anant Sahai.
- Jake Austin.
- Suhong Moon.
- Kumar Krishna Agrawal.