EECS 182 Deep Neural Networks
Spring 2023 Anant Sahai

Homework 4

This homework is due on Friday, February 24, 2022, at 10:59PM.

1. Understanding Dropout (Coding Question)

In this question, you will analyze the effect of dropout in a simplified setting. Please follow the instructions in the Jupyter notebook and answer the questions in your submission of the written assignment. The notebook does not need to be submitted.

(a) (No dropout, least-square) **The mathematical expression of the OLS solution, and the solution calculated in the code cell.**

(b) (No dropout, gradient descent) **The solution in the code cell. Are the weights obtained by training with gradient descent the same as those calculated using the closed-form least squares method**

(c) (Dropout, least-square) **The solution in the code cell.**

(d) (Dropout, gradient descent) **Describe the shape of the training curve. Are the weights obtained by training with gradient descent the same as those calculated using the closed-form least squares method?**

(e) (Dropout, gradient descent, large batch size) **Describe the loss curve and compare it with the loss curve in the last part. Why are they different? Also compare the trained weights with the one calculated by the least-square formula.**

(f) Refer back to the cells you ran in part (e). **Analyze how and why adding dropout changes the following: (i) How large were the final weights w_1 and w_2 compared to each other. (ii) 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).

(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.

(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?**

(i) Dropout on real data: **Run the notebook cells in part (I), and report on how they affect the final performance.**

2. Regularization and Dropout

You saw one perspective on the implicit regularization of dropout in HW, and here, you will see another one. Recall that linear regression optimizes the following learning objective:

$$\mathcal{L}(\mathbf{w}) = ||\mathbf{y} - X\mathbf{w}||_2^2 \tag{1}$$

One way of using *dropout* during SGD on the d-dimensional input features $\mathbf{x}_i$ involves *keeping* each feature at random $\sim_{i.i.d}$ $Bernoulli(p)$ (and zeroing it out if not kept) and then performing a traditional SGD step.

It turns out that such dropout makes our learning objective effectively become

$$\mathcal{L}(\check{\mathbf{w}}) = \mathbb{E}_{R\sim Bernoulli(p)}\left[||\mathbf{y} - (R \odot X)\check{\mathbf{w}}||_2^2\right] \tag{2}$$

where $\odot$ is the element-wise product and the random binary matrix $R \in \{0,1\}^{n\times d}$ is such that $R_{i,j} \sim_{i.i.d} Bernoulli(p)$. We use $\check{\mathbf{w}}$ to remind you that this is learned by dropout.

Recalling how Tikhonov-regularized (generalized ridge-regression) least-squares problems involve solving:

$$\mathcal{L}(\mathbf{w}) = ||\mathbf{y} - X\mathbf{w}||_2^2 + ||\Gamma\mathbf{w}||_2^2 \tag{3}$$

for some suitable matrix Γ.

(a) **Show that we can manipulate (2) to eliminate the expectations and get:**

$$\mathcal{L}(\check{\mathbf{w}}) = ||\mathbf{y} - pX\check{\mathbf{w}}||_2^2 + p(1-p)||\check{\Gamma}\check{\mathbf{w}}||_2^2 \tag{4}$$

with $\check{\Gamma}$ being a diagonal matrix whose j-th diagonal entry is the norm of the j-th column of the training matrix X.

(b) **How should we transform the $\check{\mathbf{w}}$ we learn using (4) (i.e. with dropout) to get something that looks a solution to the traditionally regularized problem (3)?**
(Hint: This is related to how we adjust weights learned using dropout training for using them at inference time. PyTorch by default does this adjustment during training itself, but here, we are doing dropout slightly differently with no adjustments during training.)

(c) With the understanding that the Γ in (3) is an invertible matrix, **change variables in (3) to make the problem look like classical ridge regression:**

$$\mathcal{L}(\widetilde{\mathbf{w}}) = ||\mathbf{y} - \widetilde{X}\widetilde{\mathbf{w}}||_2^2 + \lambda||\widetilde{\mathbf{w}}||_2^2 \tag{5}$$

Explicitly, what is the changed data matrix $\widetilde{X}$ in terms of the original data matrix X and Γ?

(d) Continuing the previous part, with the further understanding that Γ is a *diagonal* invertible matrix with the j-th diagonal entry proportional to the norm of the j-th column in X, **what can you say about the norms of the columns of the effective training matrix $\widetilde{X}$** and **speculate briefly on the relationship between dropout and batch-normalization.**

3. Weights and Gradients in a CNN

In this homework assignment, we aim to accomplish two objectives. Firstly, we seek to comprehend that the weights of a CNN are a weighted average of the images in the dataset. This understanding is crucial in answering a commonly asked question: does a CNN memorize images during the training process? Additionally, we will analyze the impact of spatial weight sharing in convolution layers. Secondly, we aim to gain an understanding of the behavior of max-pooling and avg-pooling in backpropagation. By accomplishing these objectives, we will enhance our knowledge of CNNs and their functioning.

Let's consider a convolution layer with input matrix $\mathbf{X} \in \mathbb{R}^{n\times n}$,

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,n} \end{bmatrix}, \tag{6}$$

weight matrix $\mathbf{w} \in \mathbb{R}^{k \times k}$,

$$\mathbf{w} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,k} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,1} & w_{k,2} & \cdots & w_{k,k} \end{bmatrix}, \tag{7}$$

and output matrix $\mathbf{Y} \in \mathbb{R}^{m \times m}$,

$$\mathbf{Y} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,m} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m,1} & y_{m,2} & \cdots & y_{m,m} \end{bmatrix}. \tag{8}$$

For simplicity, we assume the number of the input channel (of $\mathbf{X}$ is) and the number of the output channel (of output $\mathbf{Y}$) are both 1, and the convolutional layer has no padding and a stride of 1.

Then for all i, j,

$$y_{i,j} = \sum_{h=1}^{k} \sum_{l=1}^{k} x_{i+h-1,j+l-1} w_{h,l}, \tag{9}$$

or

$$\mathbf{Y} = \mathbf{X} * \mathbf{w}, \tag{10}$$

, where $*$ refers to the convolution operation. For simplicity, we omitted the bias term in this question.

Suppose the final loss is $\mathcal{L}$, and the upstream gradient is $d\mathbf{Y} \in \mathbb{R}^{m,m}$,

$$d\mathbf{Y} = \begin{bmatrix} dy_{1,1} & dy_{1,2} & \cdots & dy_{1,m} \\ dy_{2,1} & dy_{2,2} & \cdots & dy_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ dy_{m,1} & dy_{m,2} & \cdots & dy_{m,m} \end{bmatrix}, \tag{11}$$

where $dy_{i,j}$ denotes $\dfrac{\partial \mathcal{L}}{\partial y_{i,j}}$.

(a) **Derive the gradient to the weight matrix** $d\mathbf{w} \in \mathbb{R}^{k,k}$,

$$d\mathbf{w} = \begin{bmatrix} dw_{1,1} & dw_{1,2} & \cdots & dw_{1,k} \\ dw_{2,1} & dw_{2,2} & \cdots & dw_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ dw_{k,1} & dw_{k,2} & \cdots & dw_{k,k} \end{bmatrix}, \tag{12}$$

where $dw_{h,l}$ denotes $\dfrac{\partial \mathcal{L}}{\partial w_{h,l}}$. Also, **derive the weight after one SGD step with a batch of a single image**.

(b) The objective of this part is to investigate the effect of spatial weight sharing in convolution layers on the behavior of gradient norms with respect to changes in image size.
For simplicity of analysis, we assume $x_{i,j}, dy_{i,j}$ are independent random variables, where for all i, j:

$$\mathbb{E}[x_{i,j}] = 0, \tag{13}$$
$$\mathrm{Var}(x_{i,j}) = \sigma_x^2, \tag{14}$$
$$\mathbb{E}[dy_{i,j}] = 0, \tag{15}$$
$$\mathrm{Var}(dy_{i,j}) = \sigma_g^2. \tag{16}$$

Derive the mean and variance of $dw_{h,l} = \dfrac{\partial \mathcal{L}}{\partial W_{h,l}}$ **for each** i, j **a function of** n, k, σ_x, σ_g**. What is the asymptotic growth rate of the standard deviation of the gradient on** $dw_{h,l}$ **with respect to the length and width of the image** n**?**
Hint: there should be no m *in your solution because* m *can be derived from* n *and* k*.*
Hint: you cannot assume that $x_{i,j}$ *and* $dy_{i,j}$ *follow normal distributions in your derivation or proof.*

(c) **For a network with only 2x2 max-pooling layers (no convolution layers, no activations), what will be** $d\mathbf{X} = [dx_{i,j}] = [\dfrac{\partial \mathcal{L}}{\partial x_{i,j}}]$**? For a network with only 2x2 average-pooling layers (no convolution layers, no activations), what will be** $d\mathbf{X}$**?**
HINT: Start with the simplest case first, where $\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$*. Further assume that top left value is selected by the max operation. i.e.*

$$y_{1,1} = x_{1,1} = \max(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) \tag{17}$$

Then generalize to higher dimension and arbitrary max positions.

(d) Following the previous part, **discuss the advantages of max pooling and average pooling** in your own words.
Hint: you may find it helpful to finish the question "Inductive Bias of CNNs (Coding Question)" before working on this question

4. Inductive Bias of CNNs (Coding Question)

In this problem, you will follow the EdgeDetection.ipynb notebook to understand the inductive bias of CNNs.

(a) Overfitting Models to Small Dataset: **Fill out notebook section (Q1).**
  (i) Can you find any interesting patterns in the learned filters?

(b) Sweeping the Number of Training Images: **Fill out notebook section (Q2).**
  (i) Compare the learned kernels, untrained kernels, and edge-detector kernels. What do you observe?
  (ii) 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?

(c) Checking the Training Procedures: **Fill out notebook section (Q3).**
  (i) List every epochs that you trained the model. Final accuracy of CNN should be at least 90% for 20 images per class.
  (ii) Check the learned kernels. What do you observe?
  (iii) (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?
  (iv) (optional) Do hyperparameter tuning. And list the best hyperparameter setting that you found and report the final accuracy of CNN and MLP.
  (v) How much more data is needed for MLP to get a competitive performance with CNN? Does MLP really generalize or memorize?

(d) Domain Shift between Training and Validation Set: **Fill out notebook section (Q4).**
  (i) 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.)
  (ii) Why do you think MLP fails to learn the task while CNN can learn the task? (Hint: Think about the model architecture.)

(e) When CNN is Worse than MLP: **Fill out notebook section (Q5).**
  (i) What do you observe? What is the reason that CNN is worse than MLP? (Hint: Think about the model architecture.)
  (ii) Assuming we are increasing kernel size of CNN. Does the validation accuracy increase or decrease? Why?
  (iii) How do the learned kernels look like? Explain why.

(f) Increasing the Number of Classes: **Fill out notebook section (Q6).**
  (i) Compare the performance of CNN with max pooling and average pooling. What are the advantages of each pooling method?

5. Memory considerations when using GPUs (Coding Question)

In this homework, you will run GPUMemory.ipynb to train a ResNet model on CIFAR-10 using PyTorch and explore its implications on GPU memory.

We will explore various systems considerations, such as the effect of batch size on memory usage and how different optimizers (SGD, SGD with momentum, Adam) vary in their memory requirements.

It is strongly recommended that you start early on this question, since colab daily GPU limits may require you to complete this question over a few days with breaks in between.

(a) Managing GPU memory for training neural networks (Notebook Section 1).
   (i) **How many trainable parameters does ResNet-152 have**? What is the **estimated size of the model in MB**?
   (ii) Which GPU are you using? **How much total memory does it have**?
   (iii) After you load the model into memory, **what is the memory overhead (MB) of the CUDA context loaded with the model**?

(b) Optimizer memory usage (Notebook Section 2).
   (i) **What is the total memory utilization during training with SGD, SGD with momentum and Adam optimizers**? Report in MB individually for each optimizer.
   (ii) **Which optimizer consumes the most memory**? Why?

(c) Batch size, learning rates and memory utilization (Notebook Section 3)
   (i) **What is the memory utilization for different batch sizes (4, 16, 64, 256)? What is the largest batch size you were able to train?**
   (ii) **Which batch size gave you the highest accuracy at the end of 10 epochs?**
   (iii) **Which batch size completed 10 epochs the fastest (least wall clock time)? Why?**
   (iv) Attach your **training accuracy vs wall time plots** with your written submission.

6. 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.**

Contributors:

- Olivia Watkins.
- Anant Sahai.
- Jake Austin.
- Linyuan Gong.
- Saagar Sanghavi.
- Jerome Quenum.
- Peter Wang.
- Long He.
- Suhong Moon.
- Kumar Krishna Agrawal.
- Romil Bhardwaj.

