Student Name: Leo-Adventure
Student ID: 3038745426

Deep Neural Networks(EECS182)
Homework4

---

1. Understanding Dropout(Coding Question)

   (a)

   $$w = (x^T x)x^T y$$

   And the result is

   $$[[1.08910891][0.10891089]]$$

   (b)

   The output is

   $$Weights : odict_values([tensor([[1.0784, 0.1078]]), tensor([0.1078])])$$

   The weight is slightly different from the closed-form least squares method, but we can assume that they are the same.

   (c)

   The output is

   $$\begin{aligned} x &= [[20, 2][20, 0][0, 2][0, 0]] \\ y &= [[11][11][11][11]] \\ w &= [[0.36666667][3.66666667]] \end{aligned}$$

   (d)

   The loss fluctuates during training, the result is

   $$tensor([[0.2027, 3.4874]]) \tag{1}$$

   which is not the same as the one calculated by the least-square formula.

   (e)

   The loss becomes smaller and smaller during training, and the process of converging is smoother than the previous one. They are different because the batch size is larger than the previous one, so the variability is reduced and the noise is reduced, which make a smoother curve.

   The weight result is

   $$tensor([[0.3667, 3.4892]])$$

   which is almost the same as the least-square formula.

(f)

(i)(ii)

Without dropping out, the w_1 is 10x larger than w_2, and will contribute 100x more than w_2. But with dropping out, the w_2 is 10x larger than w_1, and now they will contribute the same to the final result.

The reason is as follows, 1) when the w_1 is dropped out, then the w_2 will become 11; 2) when the w_2 is dropped out, then the w_1 will become 1.1; in that two cases, the w_2 is 10x larger than w_1; in the third case, when neither of w_1 or w_2 is dropped out, then if $w_1 = 1$ and $w_2 = 10$, then since the coefficient of w_1 is 10x larger than that of w_2, then the gradient descent will pulls w_1 down 10x more than w_2, which will result the w_1 will become 10 smaller than w_2.

(g)

The discontinuity appears because when the drop out rate is 0, then there will exist many loss-minimizing solutions.

(h)

When the drop out rate is 0.5, they are the same, since there are only a unique solution, but in other cases, since the Adam adaptive learning rate will make the w_1 and w_2 approximately equal while the SGD is the norm-minimizing solution.

(i)

The model with drop out will achieve a better accuracy on clean data.

2. Regularization and Dropout

(a)

Since $\|y - Xw\|_2^2 = \|y - pX\breve{w}\|_2^2$, we obtain that $w = p\breve{w}$.

Besides,

$$\begin{aligned} \|\Gamma w\|_2^2 &= p(1-p)\|\breve{\Gamma}\breve{w}\|_2^2 \\ &= \|\breve{\Gamma}\sqrt{p(1-p)}\breve{w}\|_2^2 \\ \Longrightarrow \Gamma &= \sqrt{\frac{1-p}{p}}\breve{\Gamma} \end{aligned}$$

(b)

$$\breve{w} = \frac{w}{p}$$

(c)

Let $\widetilde{w} = \Gamma w$, so that the left part of the $L(\widetilde{w})$ becomes $\|y - X\Gamma^{-1}\widetilde{w}\|$, so that

$$\widetilde{X} = X\Gamma^{-1}$$

(d)

From (c), we can obtain that

$$\widetilde{X} = X\Gamma^{-1} = X\sqrt{\frac{p}{1-p}}\breve{\Gamma}^{-1}$$

so that,

$$\widetilde{X}\breve{\Gamma} = X\sqrt{\frac{p}{1-p}}$$

which indicates that the dropout can can serve as batch-normalization.

3. Weights and Gradients in a CNN

4. Inductive Bias of CNNs(Coding Question)

   (a)

   The second layers has two obvious stripes, and compared to the untrained filters, the trained filters are dimmer and smoother.

   (b)

   (i)

   When the validation accuracy is lower than 90%, then the filter looks like random-generated, but when the validation accuracy is higher than 90%, then the filter looks like the edge detectors.

   (ii) Because of the bias-variance tradeoff, the CNN initialized by pretrained edge detector will introduce higher bias than that initialized with random weight. And the latter one will have a higher variance than the former one. Since the edge detection task is simple, when the number of images reaches 50, the more variance can bring advantages to the latter one, and since the former one has a higher bias, then its variance is less than the latter one, as a result, it will perform worse than the CNN initialized with random weights.

   (c)

   (i)

   All of the epoch is 500.

   (ii)

   The kernels look much like the edge detectors.

   (iii)

   As the model begin overfitting, it start being overconfident.

   (v)

   memorize

   (d)

   (i)

   Because the CNN will overfit to edge in the left half or upper half, so when the input image has a right half edge, the CNN will classify them as no edge. The MLP will overfit to edge 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.

   (ii)

   The CNN uses the regional kernel to learn the feature of each region, recognize their pattern instead of their location, but MLP will learn their location, when the location changes the MLP perform much worse the CNN.

   (e)

   (i)

The validation accuracy of CNN is always worse than MLP in each number of training images.

The reason is that the CNN will focus on a local region each time, and it is to utilize the local correlation to classify images. And the local correlation will destroy by the permutation matrix. But MLP is full-connected, and it will consider every pixel of the image as a whole, so it will not impact the performance of MLP, result in a better performance over CNN.

(ii)

It will increase, because it can take a larger region into consideration, the local corelation will increase in that case.

(iii)

it looks like a random-generated kernel. Since the local corelation is destroyed by the random permutation.

(f)

(i)

The performance of max pooling is better than average pooling in this task. The advantage of max pooling is that it is very easy to compute, and it retains the most significant features. The advantage of average pooling is that it leads to a smoother outputs and can preserve more information.

5. Memory considerations when using GPUs(Coding Question)

(a)

(i)

The ResNet-152 have 58164298 trainable parameters. The estimated size of the model is 232.657192 MB.

(ii)

I am using the GPU 0, the total memory is 15360MiB, which is 16057 MB.

(iii) After the model is loaded into memory, the memory overhead is 815 MiB, which is 848MB.

(b)

(i)

For SGD, it is 3193.0 MB, while for SGD with momentum, it is 2999MB. For Adam, it is 3359MB.

(ii)

The Adam optimizer uses the most memory, because it needs to store a moving average of the gradients and the squared gradients for each parameter, which requires more memory than SGD.

(c)

(i)

The memory utilization of 16 batch size is , I can train the 256 batch size.