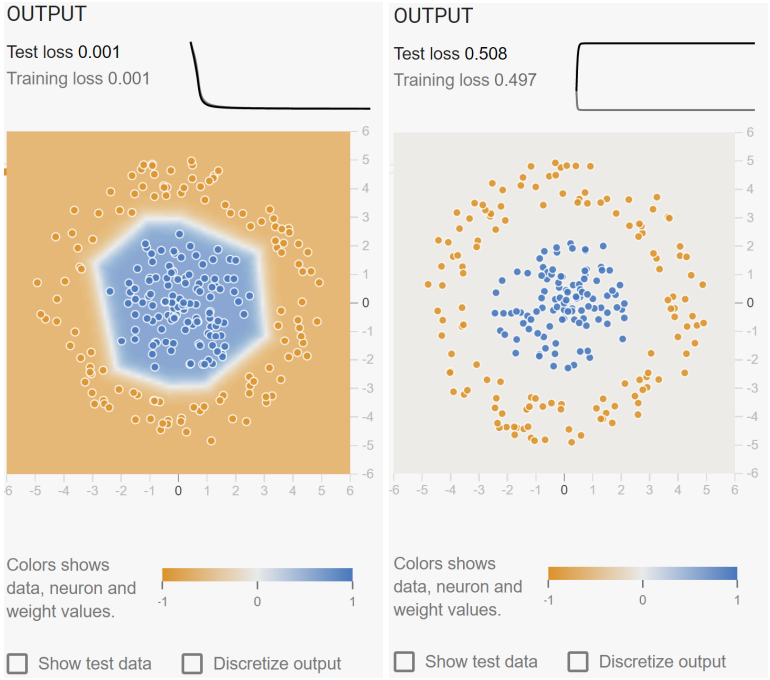
Machine Learning & Data Mining
Caltech CS/CNS/EE 155
Homework 4

Changhao Xu
UID: 2103530
February 4th, 2020

1 Decision Trees

Question A:



The 1st neural network has initial layer weights of random numbers, while the 2nd neural network has initial layer weights of 0. From back propagation, we know that

$$\frac{\partial L}{\partial \mathbf{W}^{(l)}} = \frac{\partial \mathbf{s}^{(l)}}{\partial \mathbf{W}^{(l)}} \times \frac{\partial \mathbf{x}^{(l)}}{\partial \mathbf{s}^{(l)}} \times \frac{\partial \mathbf{s}^{(l+1)}}{\partial \mathbf{x}^{(l)}} \times \frac{\partial \mathbf{x}^{(l+1)}}{\partial \mathbf{s}^{(l+1)}} \times \frac{\partial L}{\partial \mathbf{x}^{(l+1)}}$$

, where

$$\frac{\partial \mathbf{s}^{(l)}}{\partial \mathbf{W}^{(l)}} = \mathbf{x}^{(l-1)}, and \frac{\partial \mathbf{s}^{(l+1)}}{\partial \mathbf{x}^{(l)}} = \mathbf{W}^{(l+1)}$$

. At the last layer

$$\frac{\partial L}{\partial \mathbf{W}^{(l)}} = \frac{\partial \mathbf{s}^{(l)}}{\partial \mathbf{W}^{(l)}} \times \frac{\partial \mathbf{x}^{(l)}}{\partial \mathbf{s}^{(l)}} \times \frac{\partial L}{\partial \mathbf{x}^{(l)}}$$

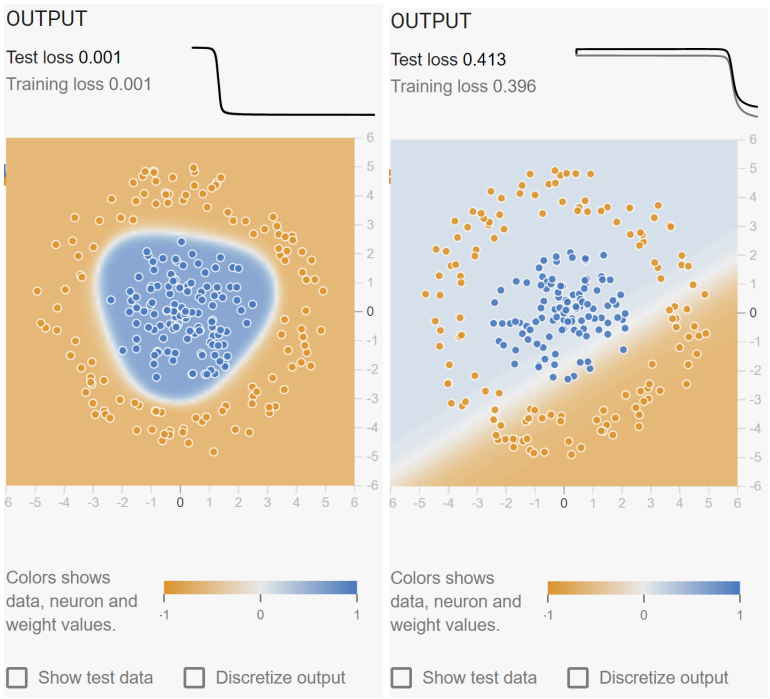
, where

$$\frac{\partial \mathbf{s}^{(l)}}{\partial \mathbf{W}^{(l)}} = \mathbf{x}^{(l-1)}$$

In addition, the activation function ReLU(x) = max(x,0) = 0 when x = 0. Therefore, when the layer weights are initialized with 0, weights can not be updated with gradient descent. As shown in the figures, the 1st neural network performs well, while the 2nd neural network have a high training and test loss.

Question B:

Changhao Xu
UID: 2103530



The 1st neural network has initial layer weights of random numbers, while the 2nd neural network has initial layer weights of 0. From back propagation, we know that

$$\frac{\partial L}{\partial \mathbf{W}^{(l)}} = \frac{\partial \mathbf{s}^{(l)}}{\partial \mathbf{W}^{(l)}} \times \frac{\partial \mathbf{x}^{(l)}}{\partial \mathbf{s}^{(l)}} \times \frac{\partial \mathbf{s}^{(l+1)}}{\partial \mathbf{x}^{(l)}} \times \frac{\partial \mathbf{x}^{(l+1)}}{\partial \mathbf{s}^{(l+1)}} \times \frac{\partial L}{\partial \mathbf{x}^{(l+1)}}$$

, where

$$\frac{\partial \mathbf{s}^{(l)}}{\partial \mathbf{W}^{(l)}} = \mathbf{x}^{(l-1)}, and \frac{\partial \mathbf{s}^{(l+1)}}{\partial \mathbf{x}^{(l)}} = \mathbf{W}^{(l+1)}$$

. What's different is that the activation function Sigmoid(x) = $\frac{exp(x)}{exp(x)+1} \neq 0$ when x = 0. Therefore, starting from the last layer, weights can still be updated since

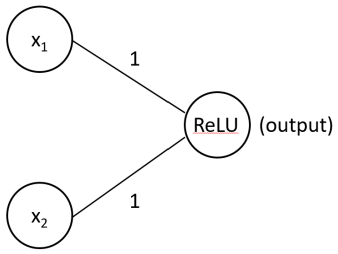
$$\frac{\partial L}{\partial \mathbf{W}^{(l)}} = \frac{\partial \mathbf{s}^{(l)}}{\partial \mathbf{W}^{(l)}} \times \frac{\partial \mathbf{x}^{(l)}}{\partial \mathbf{s}^{(l)}} \times \frac{\partial L}{\partial \mathbf{x}^{(l)}} \neq 0$$

.
Meanwhile, since $\frac{\partial L}{\partial \mathbf{W}^{(l)}}$ is proportional to $\prod \mathbf{W}^{(l)}$, when initial weights are 0, the gradient update is small, making learning speed slow compared with initial weights of random numbers.

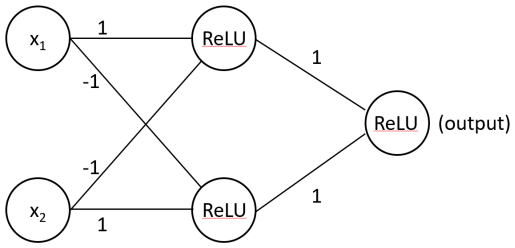
Question C: If we train neural network with ReLU activations using SGD and loop through all the negative examples only, ReLU(x) = max(x,0) = 0 when $x \leq 0$. Then the weights **W** will become 0. In this case, as shown above, weights can not be updated with gradient descent, even with positive examples later.

Question D:

Changhao Xu
UID: 2103530



Question E:



To implement an XOR, minimum 2 layers: an input layer + a hidden layer + output. This is because XOR is not linearly separable, so single layer is not possible.

2 Depth vsWidth on the MNIST Dataset

Question A: torch version: 1.2.0, torchvision version: 0.4.0

Question B: Image height = 28, Image width = 28, no. of channels = 1.
1st index: len = 60000, number of images in the training set
2nd index: len = 2, which represents image (Tensor) and corresponding digit (int)
3rd index: len = 1, which represents no. of channels
4th index: len = 28, image height
5th index: len = 28, image width
There are 60000 images in the training set, and 10000 images in the test set.

Question C: See Jupyter Notebook. Test accuracy = 0.9766.

Question D: See Jupyter Notebook. Test accuracy = 0.9810.

Question E: See Jupyter Notebook. Test accuracy = 0.9830.

Changhao Xu
UID: 2103530

---

3 Convolutional Neural Networks

Question A: Zero-padding scheme benefit: it preserves spatial size, so that the boarder of the image can also be counted. Drawback: We add zeros at the edges, and then the data is changed during convolution. Also we need to do more computations to perform convolutions.

Question B: number of parameters (weights) = $(5 \times 5 \times 3 + 1) \times 8 = 608$.

Question C: shape of the output tensor = $28 \times 28 \times 8$.

Question D: $\begin{bmatrix} 1 & 0.5 \\ 0.5 & 0.25 \end{bmatrix}, \begin{bmatrix} 0.5 & 1 \\ 0.25 & 0.5 \end{bmatrix}, \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \begin{bmatrix} 0.5 & 0.25 \\ 1 & 0.5 \end{bmatrix}$

Question E: $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

Question F: Pooling will remove the noise of missing pixels, and will smooth the image data taken at various angles/locations.

Question G: See Jupyter Notebook. As shown below, test accuracy = 0.9893.

```
Epoch 1/10:..........
       loss: 0.5945, acc: 0.8859, val loss: 0.0622, val acc: 0.9799
Epoch 2/10:..........
       loss: 0.0962, acc: 0.9719, val loss: 0.0487, val acc: 0.9849
Epoch 3/10:..........
       loss: 0.0805, acc: 0.9772, val loss: 0.0506, val acc: 0.9855
Epoch 4/10:..........
       loss: 0.0744, acc: 0.9796, val loss: 0.0463, val acc: 0.9853
Epoch 5/10:..........
       loss: 0.0694, acc: 0.9803, val loss: 0.0483, val acc: 0.9863
Epoch 6/10:..........
       loss: 0.0669, acc: 0.9810, val loss: 0.0610, val acc: 0.9833
Epoch 7/10:..........
       loss: 0.0663, acc: 0.9817, val loss: 0.0680, val acc: 0.9816
Epoch 8/10:..........
       loss: 0.0630, acc: 0.9831, val loss: 0.0515, val acc: 0.9861
Epoch 9/10:..........
       loss: 0.0649, acc: 0.9822, val loss: 0.0583, val acc: 0.9844
Epoch 10/10:..........
       loss: 0.0612, acc: 0.9837, val loss: 0.0391, val acc: 0.9893
```

Changhao Xu
UID: 2103530

Below is the test accuracy for the 10 dropout probabilities.

```
p = 0.0:..........
        loss: 0.4287, acc: 0.9413, val loss: 0.0652, val acc: 0.9805
p = 0.1111111111111111:..........
        loss: 0.4703, acc: 0.8885, val loss: 0.0521, val acc: 0.9848
p = 0.2222222222222222:..........
        loss: 0.4737, acc: 0.9250, val loss: 0.0784, val acc: 0.9763
p = 0.3333333333333333:..........
        loss: 0.4459, acc: 0.9008, val loss: 0.0772, val acc: 0.9792
p = 0.4444444444444444:..........
        loss: 0.6755, acc: 0.8090, val loss: 0.0953, val acc: 0.9724
p = 0.5555555555555556:..........
        loss: 0.6432, acc: 0.8130, val loss: 0.1249, val acc: 0.9631
p = 0.6666666666666666:..........
        loss: 0.5966, acc: 0.8671, val loss: 0.0955, val acc: 0.9703
p = 0.7777777777777777:..........
        loss: 0.7372, acc: 0.7992, val loss: 0.1612, val acc: 0.9547
p = 0.8888888888888888:..........
        loss: 1.3541, acc: 0.5495, val loss: 0.3817, val acc: 0.9119
p = 1.0:..........
        loss: 2.3027, acc: 0.1084, val loss: 146.6301, val acc: 0.1011
```

Most effective strategy for me is to choose proper regularization strength according to convolution layer complexity. Adding # of filters of convolution layer will accordingly add # of parameters dramatically, so regularization is very important. I used layerwise regularization of dropout and batch norm, and turns out batch norm should be put immediately after convolution Conv2d, and before activation function ReLU. Then I put MaxPool2d and Dropout for additional regularization. In my case, batch norm is the most effective as without it I could not get required test performance.
One problem of validating the hyperparameters here is that we choose the parameters depending on its test set performance. This is kind of data snooping as we are learning from the test set. We should instead use cross-validation in out training set and use testing set only when we have finalized the model.