CSI 5138 Intro:DL/RL
Homework Exercise 2
Wei Li 0300113733 report date: 10/12/2019

Question 1

Question 1 requires to minimize the scalar function $\mathcal{L}(x)$ given an input x and a predefined computation graph to compute $\mathcal{L}(x)$. The computation graph can be drawn as below:

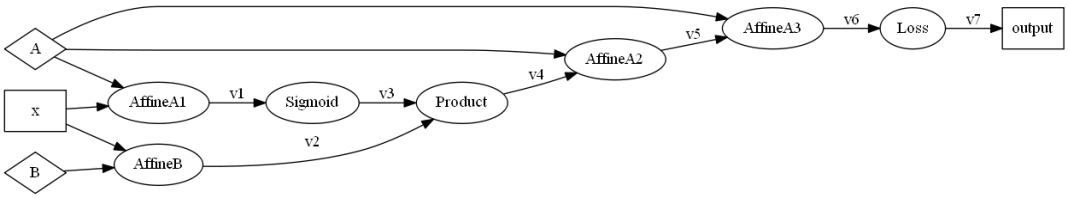


Figure 1: Forward Computation Graph

In the graph to compute $\mathcal{L}(x)$, there are two parameters, matrix A and matrix B, to be optimized over. The implement uses back-propagation algorithm to generate the gradient of $\mathcal{L}(x)$ given A, $\frac{\partial \mathcal{L}}{\partial A}$, and the gradient of $\mathcal{L}(x)$ given B, $\frac{\partial \mathcal{L}}{\partial B}$, and then use SGD to update A and B respectively. If the algorithm run correctly, A and B can reach to a optimal value that minimize $\mathcal{L}(x)$ with any input x.

The algorithm uses classes to represent computation vertices. In this case, there are 5 classes. These classes are:

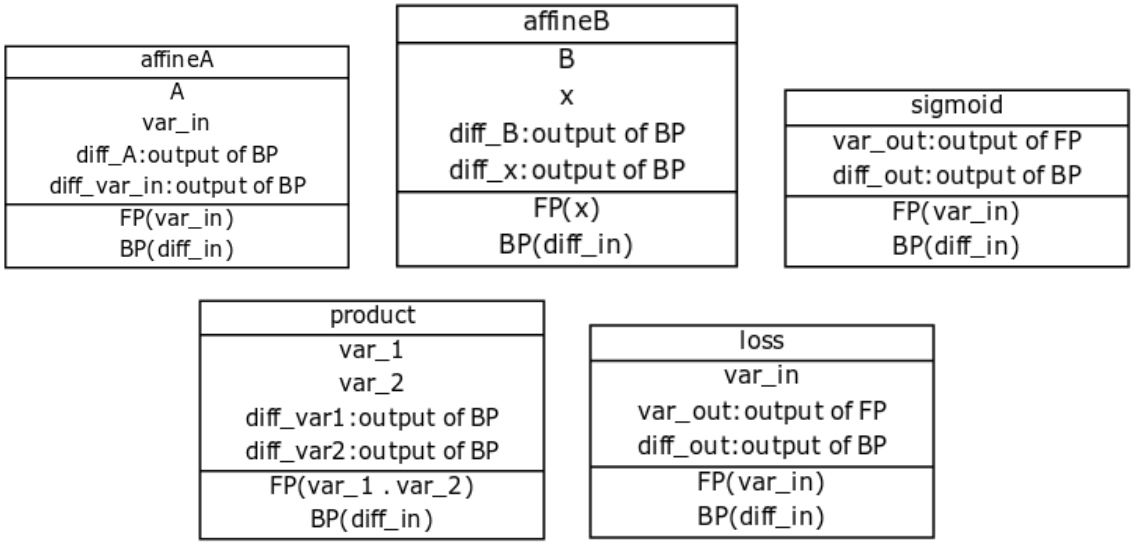


Figure 2: Classes in Algorithm

Each class has *FP* method and *BP* method to do forward propagation and backward propagation. Parameters and outcomes that are useful during the calculation are also stored inside each class.

The backward propagation computation graph is defined as below:

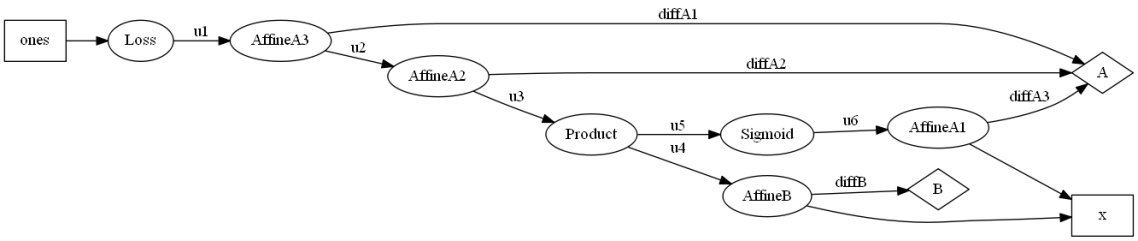


Figure 3: Backward Computation Graph

A network class is defined according to the backward computation graph. The parameters A and B is also stored in the network class. Each time gradients is calculated, network will call its *UpdatePara* method to update A and B.

Since it is only a simple optimization problem, we ran 15000 steps, and the learning rate was set to 0.01. The dimension of x, A and B was set to 50, i.e. K = 50. And the input size N was set to 100. The learning curve was drawn as below:

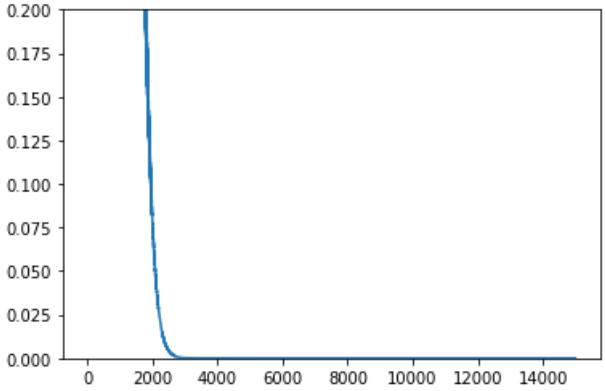


Figure 4: Learning Curve

And we got A as a zero matrix, while B had some random values. Since $\mathcal{L}(x)$ should be not smaller than 0, A can certainly reduce our target to its minimum.

Question 2

For hypothesis space $\mathcal{H}_1$ and $\mathcal{H}_1$:

$$\mathcal{H}_1 := \left\{\text{softmax}(Wx) : W \in \mathbb{R}^{K\times m}\right\} \tag{1}$$

$$\mathcal{H}_2 := \left\{\text{softmax}((A+B)Cx) : A \in \mathbb{R}^{K\times K}, B \in \mathbb{R}^{K\times K}, C \in \mathbb{R}^{K\times m}\right\} \tag{2}$$

Prove: $\mathcal{H}_1 = \mathcal{H}_1$

Proof:

- Proof $\mathcal{H}_1 = \mathcal{H}_1$ is equal to proof that W and $(A+B)C$ are in the same vector space.
- Since vector space $\mathbb{R}^{K\times K}$ is closed under addition, $\forall$ A and $B \in \mathbb{R}^{K\times K}$, $\exists U \in \mathbb{R}^{K\times K}$ such that $A + B = U$, so we can simply denote $A + B$ as U

- Since we know that:

  1. $\forall \xi, \eta \in \mathbb{R}^{K\times m}, U(\xi+\eta) = U(\xi) + U(\eta)$
  2. $\forall \xi \in \mathbb{R}^{K\times m}, a \in \mathbb{R}, U(a\xi) = aU\xi$

  We can see U as a linear transformation $\sigma(C)$ that map C from the vector space $\mathbb{R}^{K\times m}$ to another vector space which is defined in $\mathbb{R}$ and has a dimension of $K \times K$.

  The vector space is denoted as V. Since every matrix in V has a dimension of $K \times m$, it is clear that V is a subspace of $\mathbb{R}^{K\times m}$.

- **Theorem 1** *Two finite dimensional vector spaces are isomorphic if and only if they have the same dimension*

- $\mathbb{R}^{K\times m}$ and V have the same dimension, hence they are isomorphic, i.e. there exists one to one mappings that map from V to $\mathbb{R}^{K\times m}$ and vice versa.

- As a result, $\|V\|$ and $\|\mathrm{R}^{K\times m}\|$ are the same. As we know that V$\subseteq \mathbb{R}^{K\times m}$, $V = \mathbb{R}^{K\times m}$.

- So $\mathcal{H}_1 = \mathcal{H}_1$.

Question 3

Question 3 requires to build three classifiers for the MNIST dataset using three different models: soft-max regression, MLP and CNN, and to examine their behavior with and without dropout/ batch normalization.

The dataset was downloaded from THE MNIST DATABASE. Firstly, a histogram of the training set and test set was drawn. It seems like the data fits to a uniform distribution and no

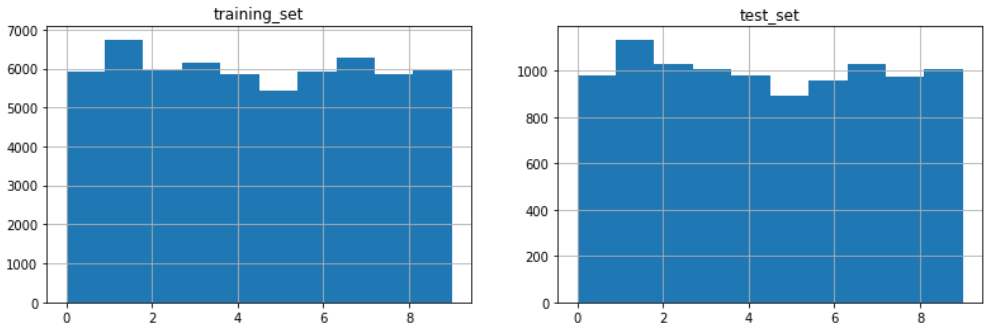


Figure 5: Distribution of Data

skewness is detected.

Since it is not desirable to encounter huge number during the training, the unpacked data was then normalized to a mean value of 0.1, and a standard deviation of 0.5.

The mini batch data was selected by pytorch's DataLoader, the batch size was 100 for the training set, and 1000 for the test set. Data was also shuffled before selected into a mini batch. Also, data was flatten before feeding into softmax classifier and MLP classifier.

In the first experiment, we compared classify accuracy of soft-max regression, MLP and CNN,

and their respective performance with and without dropout and batch normalization. In each model, the result came from 15 independent training. For all the training and testing in the first experiment, some training parameters was set to:

1. every 600 batches we call it 1 epoch, There are 20 epoches. In total, we train 12000 batches in each training.
2. the batch size of the training set is 100, while the batch size of the test set is 1000.
3. learning rate is set to 0.06
4. we used SGD with momentum, the momentum is set to 0.5
5. we used Cross Entropy loss

The structure of each model can be represent as:

Softmax	MLP	CNN
InputLayer	InputLayer	Conv: C=10, kernal=5
(BatchNorm)	FullConnect:50	(BatchNorm)
FullConnect:10	FullConnect:30	ReLU
(dropout,p=0.5)	ReLU	MaxPool:kernal=2
Softmax	(dropout, p=0.5)	Conv: C=20, kernal=5
	(BatchNorm)	(BatchNorm)
	FullConnect:10	ReLU
	Softmax	MaxPool:kernal=2
		(dropout,p=0.5)
		FullConnect:200
		(BatchNorm)
		ReLU
		FullConnect:10
		Softmax

Table 1: Models' Structure

Then we had the result:

#test	Softmax	Softmax+dropout	Softmax+BatchNorm
1	0.900	0.493	0.929
2	0.898	0.503	0.930
3	0.901	0.479	0.930
4	0.901	0.491	0.929
5	0.900	0.492	0.929
6	0.901	0.490	0.929
7	0.901	0.497	0.930
8	0.898	0.488	0.930
9	0.895	0.479	0.929
10	0.901	0.488	0.929
11	0.900	0.500	0.930

12	0.901	0.487	0.930
13	0.898	0.491	0.929
14	0.901	0.470	0.929
15	0.903	0.489	0.929
Avg	0.900	0.489	0.929

Table 2: Softmax accuracy

#test	MLP	MLP+dropout	MLP+BatchNorm
1	0.956	0.936	0.869
2	0.956	0.856	0.956
3	0.961	0.954	0.937
4	0.951	0.945	0.940
5	0.943	0.946	0.953
6	0.961	0.941	0.940
7	0.949	0.959	0.951
8	0.958	0.937	0.961
9	0.958	0.943	0.931
10	0.937	0.871	0.934
11	0.948	0.951	0.872
12	0.962	0.955	0.931
13	0.940	0.834	0.861
14	0.953	0.947	0.955
15	0.959	0.952	0.959
Avg	0.953	0.929	0.930

Table 3: MLP accuracy

#test	CNN	CNN+dropout	CNN+BatchNorm
1	0.992	0.967	0.992
2	0.991	0.980	0.993
3	0.990	0.982	0.992
4	0.990	0.976	0.993
5	0.989	0.977	0.993
6	0.990	0.982	0.992
7	0.991	0.896	0.992
8	0.992	0.977	0.993
9	0.990	0.981	0.991
10	0.990	0.980	0.991
11	0.992	0.976	0.992
12	0.991	0.980	0.993
13	0.990	0.978	0.992
14	0.992	0.979	0.992
15	0.991	0.977	0.992
Avg	0.991	0.972	0.992

Table 4: CNN accuracy

From the average value, it is concluded that: for one thing, MLP is generally better than softmax regression, while CNN is generally better than MLP. For another thing, after applying dropout, the classifiers' performance decreased, and the phenomenon is quite extreme in the case of softmax. Finally, using batch normalization can increase classifier's performance. All the result passed the D'Agostino and Pearson's Normal Distribution test, so paired t-test is done, the result is:

ttest	softmax-mlp	softmax-cnn	mlp-cnn
pvalue	1.08E-20	1.29E-43	3.76E-17
ttest	mlp-mlpDropout	mlp-mlpBN	
pvalue	0.029205025	0.017402723	
ttest	soft-softDropout	soft-softBN	
pvalue	1.33E-44	2.12E-30	
ttest	cnn-cnnDropout	cnn-cnnBN	
pvalue	0.002946129	5.46E-07	

Table 5: Paired t-test

t-test shows that all the differences are significant statistically.

A possible reason for dropout's underperformance is that: on one hand, MNIST dataset already has a data size that is big enough and clean enough, and we also have good sampling. These factors can pretty much prevent overfitting, which make dropout unuseful here. On the other hand, models in the experiment are small, shallow and without many parameters, a dropout layer with a possibility of 0.5 may greatly limit these models' capacity.

In the second experiment, MLP is explored further. Three different MLP structures were examined:

MLP1	MLP2	MLP3
InputLayer FullConnect:12288 Softmax	InputLayer FullConnect:2048 ReLU FullConnect:2048 ReLU FullConnect:2048 ReLU FullConnect:1024 ReLU FullConnect:1024 ReLU Softmax	the same as MLP2, with dropout layers in between. The dropout possibility is: 0.15

Table 6: MLP structures

All the models had the same number of parameters, the only difference is how they were arranged. Each model was trained 35 epochs, and their training losses and test losses was tracked. Parameters regarding training is:

1. every 600 batches we call it 1 epoch, There are 35 epoches. In total, we train 21000 batches in each training.
2. learning rate is set to 0.06
3. we use SGD with momentum, the momentum is set to 0.5
4. we use Cross Entropy loss

The resulting accuracy per test is given below:

#test	MLP1	MLP2	MLP3
1	0.970	0.962	0.957
2	0.969	0.954	0.949
3	0.970	0.952	0.964
4	0.970	0.927	0.924
5	0.970	0.955	0.944
6	0.969	0.964	0.961
7	0.970	0.963	0.943
8	0.971	0.950	0.945
9	0.969	0.958	0.917
10	0.968	0.949	0.959
11	0.971	0.938	0.951
12	0.969	0.943	0.954
13	0.969	0.960	0.946
14	0.971	0.959	0.939
15	0.969	0.959	0.964
Avg	0.970	0.953	0.948
StdD	0.000856505	0.009872782	0.013126892

Table 7: MLPs accuracy

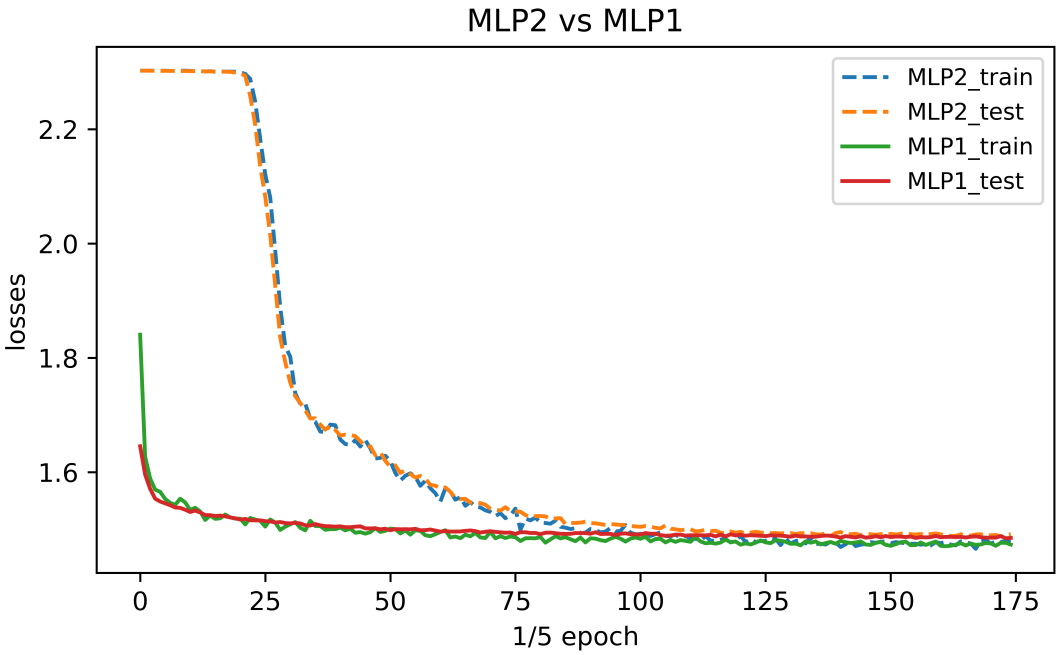
ttest	MLP1-MLP2	MLP1-MLP3	MLP2-MLP3
pvalue	7.86E-07	1.03E-06	0.253

Table 8: paired t-test

Surprisingly, MLP with only 1 layer got the best result out of the three. There is no significant difference between using dropout or not in the second structure.

Then the three models' learning curve was examined, the loss was calculated by averaging over 15 tests.

The first model, again, turns out to be the best one in terms of training speed. The second and the third structure are close, but the one with dropout(MLP3) was trained a little slower, and



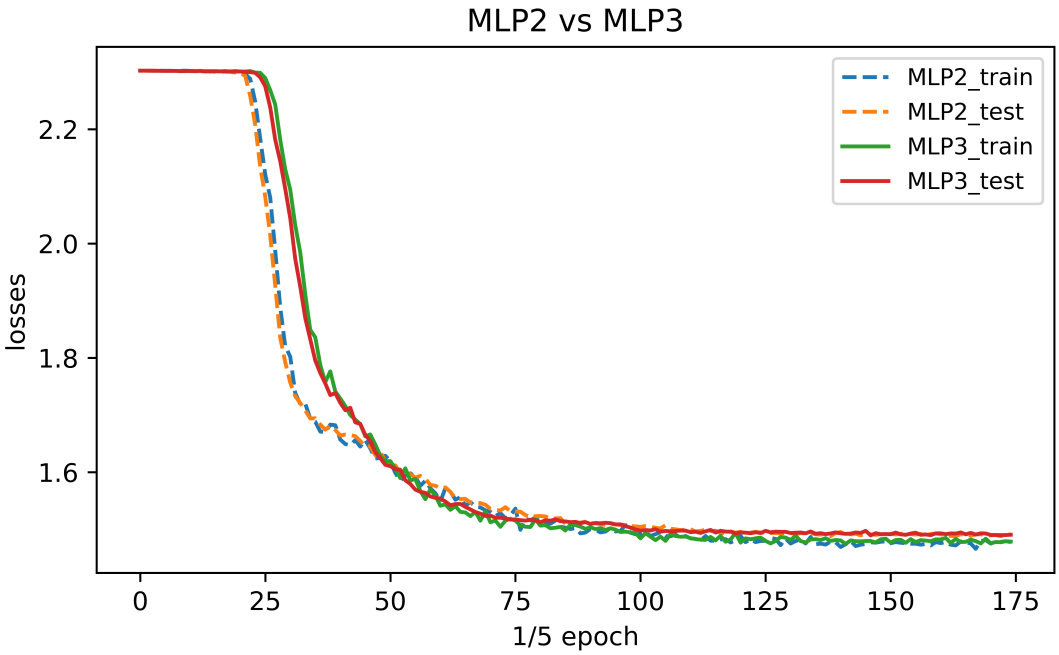


Figure 6: Loss Function Comparison

also converged at a higher loss comparing to MLP2.

Finally, we compared the euclidean distance of MLP2s' training and test error and euclidean distance of MLP3s' training and test error, i,e, we computed: $d_{euclidean}(MLP2\_train, MLP2\_test)$ and $d_{euclidean}(MLP3\_train, MLP3\_test)$), The result is $d(MLP2\_train, MLP2\_test) = 0.2121$ and $d(MLP2\_train, MLP2\_test) = 0.2015$, which means that MLP with dropout obtained a closer training losses curve and test losses curve, indicating that dropout suppressed overfitting to some extent in the experiment.