16-720B Homework 4 Write-up

Gu, Qiao

November 13, 2019

Q1.1

The given translation is applied to all x_j for $j \in \mathbb{R}$. Then for each $x_i i$

$$softmax(x_i + c) = \frac{e^{x_i+c}}{\sum_j e^{x_j+c}} = \frac{e^{x_i}e^c}{\sum_j e^{x_j}e^c} = \frac{e^{x_i}e^c}{(\sum_j e^{x_j})e^c} = \frac{e^{x_i}}{\sum_j e^{x_j}} = softmax(x_i). \tag{1}$$

This shows that softmax is invariant to translation.

When $c = -\max x_i$, all $x_i + c < 0$ and $e^{x_i+c} \in (0, 1)$, and the difference between e^{x_I} is relatively small. And when $c = 0$, e^{x_i} can be exponentially large and may cause numerical instability.

Q1.2

- Each element of softmax $softmax(x_i)$ is in range $(0, 1)$, and the sum over all elements is $\sum_j softmax(x_j) = 1$.
- Probability.
- $s_i = e^{x_i}$ is to map each x_i to its probability weight. $S = \sum s_i$ is find the sum of the weights. $softmax(x_I) = \frac{1}{S} x_i$ is to normalize each weight by all weights to get probability.

Q1.3

Each layer of a neural network can be written mathmatically as $f_i(\mathbf{x}) = \mathbf{W}_i\mathbf{x}+\mathbf{b}$, and thus is we concatenate n layers together without non-linear layers, we get the output as

$$\begin{aligned}
\mathbf{y} &= f_n(f_{n-1}(\cdots f_1(\mathbf{x})\cdots)) && (2)\\
&= \mathbf{W}_n(\mathbf{W}_{n-1}(\cdots(\mathbf{W}_1\mathbf{x}+\mathbf{b}_1)\cdots)+\mathbf{b}_{n-1})+\mathbf{b}_n && (3)\\
&= \mathbf{W}_n\mathbf{W}_{n-1}\cdots\mathbf{W}_1\mathbf{x}+\mathbf{W}_n\mathbf{W}_{n-1}\cdots\mathbf{W}_2\mathbf{b}_1+\mathbf{W}_n\mathbf{W}_{n-1}\cdots\mathbf{W}_3\mathbf{b}_2+\cdots+\mathbf{W}_n\mathbf{b}_{n-1}+\mathbf{b}_n && (4)\\
&= \mathbf{W}\mathbf{x}+b, && (5)
\end{aligned}$$

where $\mathbf{W} = \mathbf{W}_n\mathbf{W}_{n-1}\cdots\mathbf{W}_1$ and $\mathbf{b} = \mathbf{W}_n\mathbf{W}_{n-1}\cdots\mathbf{W}_2\mathbf{b}_1+\mathbf{W}_n\mathbf{W}_{n-1}\cdots\mathbf{W}_3\mathbf{b}_2+\cdots+\mathbf{W}_n\mathbf{b}_{n-1}+\mathbf{b}_n$. This can be regarded as a single linear layer, and the whole network is equivalent to linear regression.

Q1.4

$$\frac{d\sigma(x)}{dx} = \frac{d}{dx}\frac{1}{1+e^{-x}} \tag{6}$$

$$= (-1)\frac{1}{(1+e^{-x})^2}\frac{d}{dx}(1+e^{-x}) \tag{7}$$

$$= (-1)\frac{1}{(1+e^{-x})^2}(-e^{-x}) \tag{8}$$

$$= \frac{1+e^{-x}-1}{(1+e^{-x})^2} \tag{9}$$

$$- \frac{1}{1+e^{-x}} - \frac{1}{(1+e^{-x})^2} \tag{10}$$

$$= \frac{1}{1+e^{-x}}(1 - \frac{1}{1+e^{-x}}) \tag{11}$$

$$= \sigma(x)(1-\sigma(x)) \tag{12}$$

Q1.5

The loss function is unknown and therefore we assume $\frac{\partial J}{\partial y_i} = \delta_i$. Therefore

$$\frac{\partial J}{\partial W_{ij}} = \frac{\partial J}{\partial y_i}\frac{\partial y_i}{\partial W_{ij}} = \delta_i x_j \tag{13}$$

$$\frac{\partial J}{\partial x_j} = \sum_{i=0}^{k}\frac{\partial J}{\partial y_i}\frac{\partial y_i}{\partial x_j} = \delta_i \sum_{i=0}^{k} W_{ij} \tag{14}$$

$$\frac{\partial J}{\partial b_i} = \frac{\partial J}{\partial y_i}\frac{\partial y_i}{\partial b_i} = \delta_i \tag{15}$$

And then we can further rewrite it to matrix form

$$\frac{\partial J}{\partial \mathbf{W}} = \delta \mathbf{x}^T \tag{16}$$

$$\frac{\partial J}{\partial \mathbf{x}} = \mathbf{W}^T \delta \tag{17}$$

$$\frac{\partial J}{\partial \mathbf{b}} = \delta \tag{18}$$

Q1.6

1. As shown in Figure. 1, when the input to the sigmoid x is far away from zero, the magnitude of the gradient becomes very close to zero and thus the gradient from higher layers are scaled by a very small number, even "vanishing". Therefore, when we update the network weights, the changes will be very small.

2. The output range of sigmoid function is $(0, 1)$ and the output range of $tanh(x)$ is $(-1, 1)$. If our input data are centered at 0, the output given by tanh are also centered at 0, which will make the input to different layers consistent.

3. From Figure. 1, we can see that $\tanh(x)$ has a stronger gradient (larger magnitude) than $\sigma(x)$ does.

4. $\tanh(x) = 2\sigma(2x) - 1$ as

$$\begin{aligned} \sigma(x) = \frac{1}{1+e^{-x}} \Rightarrow \sigma(2x) = \frac{1}{1+e^{-2x}} \Rightarrow 2\sigma(2x) = \frac{2}{1+e^{-2x}} \\ \Rightarrow 2\sigma(2x) - 1 = \frac{2-1-e^{-2x}}{1+e^{-2x}} = \frac{1-e^{-2x}}{1+e^{-2x}} = \tanh(x) \end{aligned} \tag{19}$$

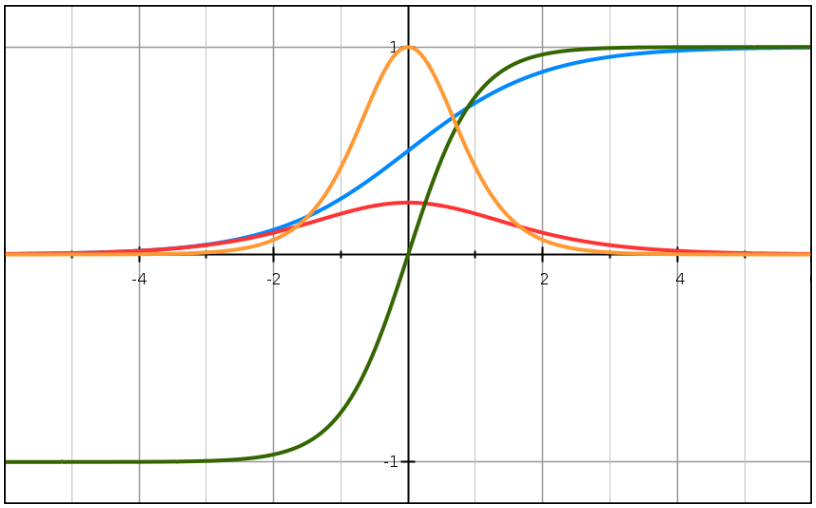


Figure 1: The plot of sigmoid function $\sigma(x)$ (in blue), its gradient $(1 - \sigma(x))\sigma(x)$ (in red), the $\tanh(x)$ function (in green) and its gradient $1 - \tanh^2(x)$ (in orange).

Q2.1.1

If the weights and biases of every layer are initialized with zeros, the outpuut layer pre-activation values will be zeros regardless the input data and the post-activation values will be a uniform probability distribution. In this case, there will still be gradients in the network due to the softmax loss, but they will be very small. Therefore the update of the network is very slow, the training process may terminate without convergence or the network may be stuck at local optima.

Q2.1.3

Because if we initialize all the weights with the same number, some of them will have the same effect on the output values, and the gradients on them will be the same during backpropagation. Therefore, they will move together and hard to converge during training.

Because in one layer of neural net $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$, the variance of the output $\mathbf{y}$ depends on both the variance of input $\mathbf{x}$ and variance of elements of the weight $\mathbf{W}$. This is also true during the backpropagation of gradients. Therefore we scale the weight variance down by the layer size, To make the output variance of each layer equal to the input variance.

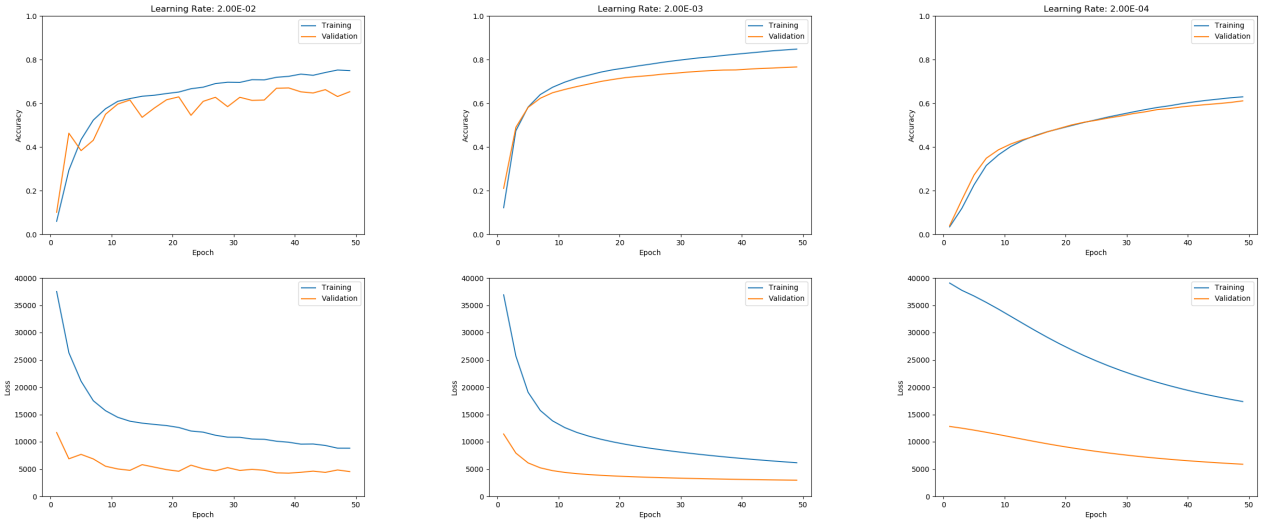
Q3.1.2

The progression of the loss and accuracy during the training process is shown in Figure. 2. Here the best learning rate I found is 2×10^{-3}.

Learning rates affect training process in the sense that the best learning rate gives the best convergence. Overly high or overly low learning rates will both slow down the convergence or make the training not able to converge to local optima.

Additionally, we can notice that large learning rates may cause the learning process to overshoot, which causes accuracy and loss to fluctuate during the training process.

The final test accuracy of the best network is 78%.



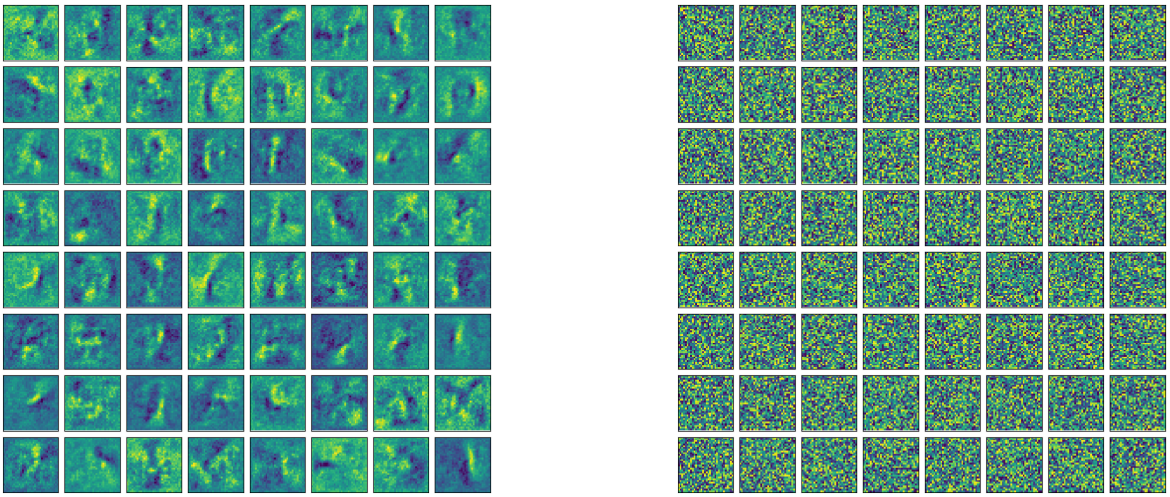
(a) Learning rate $= 2 \times 10^{-2}$ (b) Learning rate $= 2 \times 10^{-3}$ (c) Learning rate $= 2 \times 10^{-4}$

Figure 2: Training progress on loss and accuracy using different learning rates.

Q3.1.3

The weights of the first layer of the network is visualized in Figure. 3. As we can observe, the weights after training is more structured and some of them resemble stroke curves of different digits and characters, and thus when we apply these weights, the input image that have similar strokes will have higher activation.

And before training, the visualized weights are just random noise and no structures or patterns can be observed.



(a) After training

(b) Before training

Figure 3: Visualization of weights before and after training.

Q3.1.4

The confusion matrix of the best model on the validation dataset is shown in Figure. 4. As shown by some entries with high values in the matrix, the network is likely mix the digit "0" with the character ''O", the character "S" with the digit "5", and digit "1" with character "I". These results are quite reasonable as these pairs have very similar shape and even some human can get confused by them.

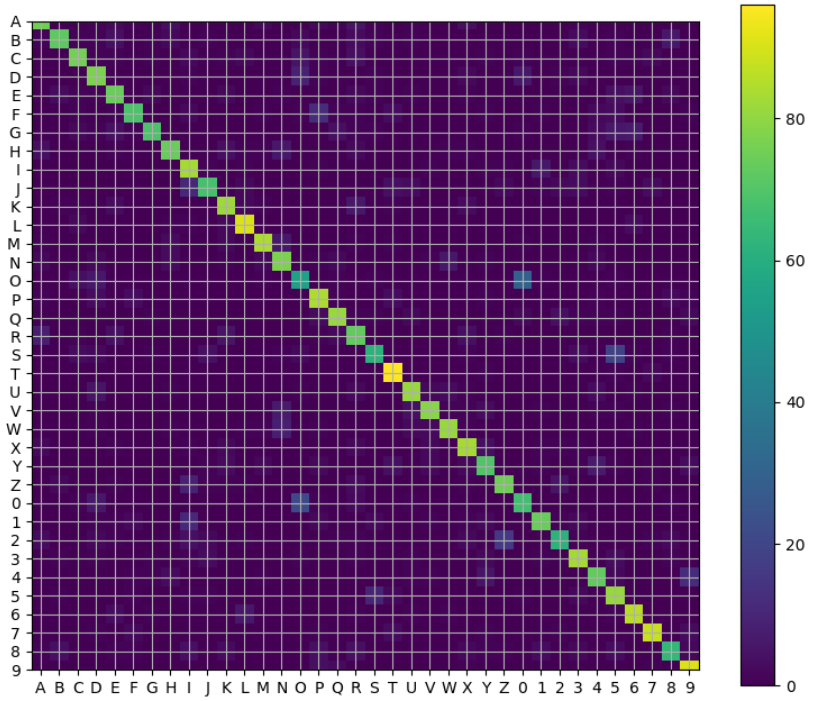


Figure 4: The confusion matrix of the best model on the validation dataset ($N = 3600$).