Program assignment 1: report

1. Theory and Steps

1.1 Stochasic gradient descent

The idea of SGD is divding the dataset into k mini-batches, randomly picking and training the data. Usually k-1 of them have the equal batch size, the rest one usually contain the rest of the data.

In my code, the batch size is 50 samples. The way I implement SGD is: at each iteration, I first shuffle the training data indices, and use this shuffled indices to obtain the data.

1.2 Hyper parameters

There are two hyperparameters:

1. Learning rate: 0.01. This is the length of step each update of parameters should take. I choose 0.01.
2. Lambda: 0.01. This is the weight for the regularizer, a large weight will hurt the training performance. I choose 0.01.
3. Batch_size: 50. Number of samples within a batch.

1.3 Forward

$W \in \mathbb{R}^{784\times 5}$, $w_0 \in \mathbb{R}^{5\times 1}$, $X \in \mathbb{R}^{784\times 1}$, $y \in \mathbb{R}^{5\times 1}$.

- **Regression**:

$$\hat{y}[m] = \frac{1}{M}\sum_{m=1}^{M} W^\top X[m] + w_0$$

- **Probability** $\log p(y|X)$:

$$\begin{aligned}\log p(y|X) =& \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\log p(y[m]=k|X[m]) \\ =& \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\mathbb{I}_{y[m]=k}(\hat{y}[m][k] - \log\sum_{k'=1}^{K}\exp(\hat{y}[m][k']))\end{aligned}$$

- **Loss function** (include regression and probability equation to help coding):

$$\begin{aligned}\mathcal{L}(D;W) &= -\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\log p(y[m]=k|x[m]) + \lambda||W||_2 \\ &= -\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\mathbb{I}_{y[m]=k}[\log\frac{\exp([W_k, w_{k,0}]^\top X[m]}{\sum_{k'=1}^{K}\exp([W_{k'}, w_{k',0}]^\top X[m]}] + \frac{\lambda}{2}\left((\sum_{k=1}^{K}W_k^2)^{1/2}\right)^2 \\ &= \quad -\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\mathbb{I}_{y[m]=k}\left[[W_k, w_{k,0}]^\top X[m] - \log\sum_{k'=1}^{K}\exp([W_{k'}, w_{k',0}]^\top X[m])\right] \\ &\qquad + \frac{\lambda}{2}\left((\sum_{k=1}^{K}W_k^2)^{1/2}\right)^2\end{aligned}$$

Note that in the real implementation, in order to be consistent to tensorflow code, I use $X \in \mathbb{R}^{\text{None}\times 784}$, I switched the dims of X. I also did this to y.

1.4 Backward

- **The gradient of** W_k:

$$\begin{aligned}\triangledown_{W_k}\mathcal{L}(D;W_k) =& \frac{1}{M}\sum_{m=1}^{M} -\mathbb{I}_{y[m]=k}x[m] + \frac{\exp([W_k, w_{k,0}]^\top X[m])}{\sum_{k'=1}^{K}\exp([W_{k'}, w_{k',0}]^\top X[m])}x[m] + \lambda W_k \\ =& \frac{1}{M}\sum_{m=1}^{M} -(\mathbb{I}_{y[m]=k} - \sigma_M[k])X[m] + \lambda W_k\end{aligned}$$

- **The gradient of** $w_{k,0}$:

$$\begin{aligned}\triangledown_{w_{k,0}}\mathcal{L}(D;w_{k,0}) =& \frac{1}{M}\sum_{m=1}^{M} -\mathbb{I}_{y[m]=k} + \frac{\exp([W_k, w_{k,0}]^\top X[m])}{\sum_{k'=1}^{K}\exp([W_{k'}, w_{k',0}]^\top X[m])} + \lambda w_{k,0} \\ =& \frac{1}{M}\sum_{m=1}^{M} -(\mathbb{I}_{y[m]=k} - \sigma_M[k]) + \lambda w_{k,0}\end{aligned}$$

- **The update of** W:

$$\begin{aligned}W^{t+1} =& \begin{bmatrix} W_1^t - \eta\triangledown_{W_1}\mathcal{L}(D;W_1) & \dots & W_5^t - \eta\triangledown_{W_5}\mathcal{L}(D;W_5) \\ w_{1,0}^t - \eta\triangledown_{w_{1,}}\mathcal{L}(D;w_{1,0}) & \dots & w_{5,0}^t - \eta\triangledown_{w_{5,0}}\mathcal{L}(D;w_{5,0}) \end{bmatrix} \\ =& W^t - \eta\begin{bmatrix} \triangledown_{W_{1:k}}\mathcal{L}(D;W_{1:5}) \\ \triangledown_{w_{1:k,0}}\mathcal{L}(D;w_{1:5,0}) \end{bmatrix}\end{aligned}$$

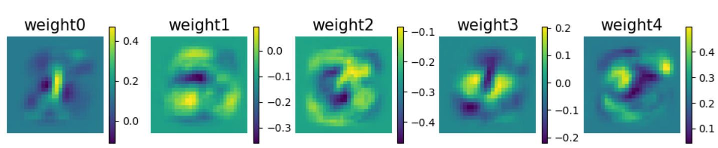
However, in my real implementation I update W and W_0 separately.

2. Experiments and results

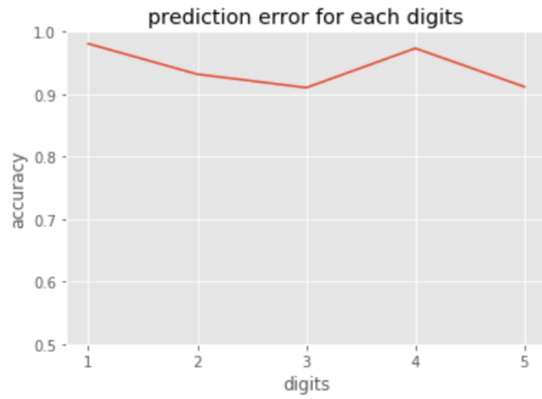
2.1 Experiments setting

I run 20 iterations (denoted as max_episodes) with SGD.

2.2 Weights



2.3 Classfication tables



2.4 Overall error train vs. test

What I obersve is the the test accuracy is always higher than that of the training test. This confused me a little bit, beuase what we want to train is on training dataset. The prediction accuracy is around 9.45, with parameters lr = 0.001, lam =0.001. A carefully tunning might return a better result.