Projects in machine learning and AI
HW 1

Xiao Jin

September 2021

1 Task 1

We have N training data samples $\mathbf{x}_n \in \mathbb{R}^m, n = 1, 2, \ldots, N$ and its corresponding labels $y_n \in \mathbb{R}, n = 1, 2, \ldots, N$. The dimension of data feature space is not very large and we need to do a classification task based on the training data and their corresponding labels. Comparing with using deep neural networks including CNN and RNN, logistic regression model is much simple. It's much easier to train the model and prevent over-fitting.

2 Task 2

The data we choose is MNIST[1] and I did normalization on the training data. We suppose $\boldsymbol{\mu}$ is the mean of training data $\{\mathbf{x}_n \in \mathbb{R}^m\}$ and $\boldsymbol{\sigma}$ is the standard deviation of the data. We did the data normalization with the following method.

$$\mathbf{x}_n = \mathbf{x}_n - \boldsymbol{\mu} \tag{1}$$

We didn't divide $\mathbf{x}_n$ by its standard deviation $\boldsymbol{\sigma}$ mainly because that some elements in $\boldsymbol{\sigma}$ is 0.

The reason we using data normalization is that we need to unify the distribution of each feature of the data and make it easier to optimize to the optimal solution.

3 Task 3

We implement the logistic regression by define the following loss function

$$\mathcal{L}(\mathbf{\Theta},\mathbf{x},\mathbf{y}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{10}\mathbf{y}[n][k]\log\frac{e^{(\mathbf{x}[n])^{\mathrm{T}}\mathbf{\Theta}_k+\Theta_{k,0}}}{\sum_{k=1}^{5} 10e^{(\mathbf{x}[n])^{\mathrm{T}}\mathbf{\Theta}_k+\Theta_{k,0}}} + \lambda\sum_{k=1}^{10}\mathbf{\Theta}_k^{\mathrm{T}}\mathbf{\Theta}_k + \Theta_{k,0}^2 \tag{2}$$

$$\mathbf{\Theta} = \begin{bmatrix}\mathbf{\Theta}_1 & \mathbf{\Theta}_2 & \mathbf{\Theta}_3 & \dots & \mathbf{\Theta}_{10}\\ \Theta_{1,0} & \Theta_{2,0} & \Theta_{3,0} & \dots & \Theta_{10,0}\end{bmatrix} \tag{3}$$

We compute the gradients of loss w.r.t model parameters $\mathbf{\Theta}$ by

$$\nabla_{\mathbf{\Theta}_k}\mathcal{L}(\mathbf{\Theta},\mathbf{x},\mathbf{y}) = \frac{\partial\mathcal{L}(\mathbf{\Theta},\mathbf{x},\mathbf{y})}{\partial\mathbf{\Theta}_k} = -\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}[n](\mathbf{y}[n][k] - \frac{e^{(\mathbf{x}[n])^{\mathrm{T}}\mathbf{\Theta}_k+\Theta_{k,0}}}{\sum_{k=1}^{10} e^{(\mathbf{x}[n])^{\mathrm{T}}\mathbf{\Theta}_k+\Theta_{k,0}}}) + 2\lambda\mathbf{\Theta}_k \tag{4}$$

$$\nabla_{\Theta_{k,0}}\mathcal{L}(\mathbf{\Theta},\mathbf{x},\mathbf{y}) = \frac{\partial\mathcal{L}(\mathbf{\Theta},\mathbf{X},\mathbf{y})}{\partial\Theta_{k,0}} = -\frac{1}{N}\sum_{n=1}^{N}(\mathbf{y}[n][k] - \frac{e^{(\mathbf{x}[n])^{\mathrm{T}}\mathbf{\Theta}_k+\Theta_{k,0}}}{\sum_{k=1}^{10} e^{(\mathbf{x}[n])^{\mathrm{T}}\mathbf{\Theta}_k+\Theta_{k,0}}}) + 2\lambda\Theta_{k,0} \tag{5}$$

We implement 2 variants of gradient descent: Gradient descent and gradient descent (mini batch).

4 Task 4

We choose Adam and Adagrad as the optimization method.

5 Results and Analysis

Algorithm	training loss	testing accuracy	# of Iteration
GD	0.727	88.48%	5000
GD with mini batch (500)	0.594	88.46%	27065

Table 1: Comparsion of 2 variants of gradient descent

Algorithm	training loss	testing accuracy	# of Iteration
GD with mini batch	0.594	88.46%	27065
Adam	0.607	87.44%	9654
Adagrad	0.594	88.46%	27065

Table 2: Comparsion of different optimization method, batch size = 500, lr = 10^{-2}

Generally, Adam performs best over all the algorithms. Although it takes less iterations for GD to reach the same testing accuracy as GD with mini batch (500). It requires more computation memories and time. As the result taking batch is a good method in model training. Furthermore, Adam performs better than GD or GD with mini batch (500) which indicates that Adam is a widely used optimization method in deep learning.

References

[1] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/