Deep Learning in Data Science
DD2424

Solution to Assignment 2

Yiming Fan
yimingf@kth.se

1 Introduction

This assignment aims at training a double-layer network using *gradient descent* method. The dataset used in this assignment is CIFAR-10.

2 Methods & Mechanisms

The network consists of two sets of linear classifiers $\mathbf{W}$ together with bias vectors $\mathbf{b}$. The classifying functions are similar as in assignment 1 and the only addition is that we applied a activation function

$$\mathbf{h} = max(0, \mathbf{s}_1) \tag{1}$$

2.1 Gradient Calculation

The gradient is now calculated by

$$\begin{align}
\frac{\partial J}{\partial \mathbf{W}_2} &= \frac{1}{|\mathcal{D}|} \sum \mathbf{g}_2 \tag{2}\\
\frac{\partial J}{\partial \mathbf{b}_2} &= \frac{1}{|\mathcal{D}|} \sum \mathbf{g}_2^T \mathbf{x}^T + 2\lambda \mathbf{W}_2 \tag{3}\\
\frac{\partial J}{\partial \mathbf{W}_1} &= \frac{1}{|\mathcal{D}|} \sum \mathbf{g}_1 \tag{4}\\
\frac{\partial J}{\partial \mathbf{b}_1} &= \frac{1}{|\mathcal{D}|} \sum \mathbf{g}_1^T \mathbf{x}^T + 2\lambda \mathbf{W}_1 \tag{5}\\
& \tag{6}
\end{align}$$

where

$$\begin{align}
\mathbf{g}_2 &= -\frac{\mathbf{y}^T}{\mathbf{y}^T \mathbf{p}} \left(\text{diag}(\mathbf{P}) - \mathbf{P}\mathbf{P}^T\right) \tag{7}\\
\mathbf{g}_1 &= \mathbf{g}_2 \cdot \mathbf{W}_2 \cdot (\text{diag}(\text{Ind}(\mathbf{s}_1) > 0)) \tag{8}
\end{align}$$

the implementation of the *gradient descent* can be seen on the file `ComputeGradient.m`. We have not tried the file `ComputeGradientNum.m` for verification of correctness but are quite convinced that our implementation is correct, since by choosing some hyper-parameters the loss function converges to a satisfactory level.

2.2 The effect of momentum

We set #(number of epochs) = 10 and kept every hyper-parameter else unchanged but trained the network with/without using momentum. We set $\rho = 0.9$. After 10 epochs the loss function converges to 2.3011 without momentum while 1.9359 with momentum which is much smaller than the former. The loss function becomes 1.8403 when $\rho = 0.99$.

Conclusion: Adding momentum would increase the speed of convergence when the gradient direction is correct. But too high momentum may lead the gradient 'oscillate' across the minimum when the training process plateaus.

#	η	λ
1	0.0035	0.0129
2	0.0033	$3.2291 \cdot 10^{-5}$
3	0.0013	0.0079

Table 1: Coarse search.

#	η	λ
1	$5.9438 \cdot 10^{-4}$	$1.5142 \cdot 10^{-4}$
2	0.0028	0.0067
3	$1.0385 \cdot 10^{-4}$	$1.5504 \cdot 10^{-5}$

Table 2: Fine search.

2.3 Coarse search for `lambda` and `eta`

We set #(number of epochs) = 10 and kept every hyper-parameter else unchanged but trained the network with/without using momentum. We tried 100 sample where λ range from 10^{-1} to 10^{-6} and η from 10^{-1} to 10^{-4}. The top 3 combination of λ-ηs are shown below. We will do the fine search using η range from 0.001 to 0.003 and λ from $1 \cdot 10^{-5}$ to $1 \cdot 10^{-2}$.

2.4 Fine search for `lambda` and `eta`

We set #(number of epochs) = 10 and kept every hyper-parameter else unchanged but trained the network with/without using momentum. We tried 100 sample where η range from 0.001 to 0.003 and λ from $1 \cdot 10^{-5}$ to $1 \cdot 10^{-2}$. The top 3 combination of λ-ηs are shown below. Among the top 3 there are roughly two sets of magnitudes of η-λ: 10^{-4} and 10^{-3}. Basically those two parameters keep the same magnitude. Later on we will try those top 1 set of hyper-parameters.
Note!: In the *report for bonus* we used $\eta = 0.02$, $\lambda = 0.001$ and trained the network with more than 80% accuracy within 30 epochs. We guess the reason we ignored the seemingly 'better' combination is that the number of λ-η combinations we used in the coarse search. In the future we might try to simply increase the number of combinations in the coarse search. We will illustrate on the fine search in that report.

2.5 Training result using the best hyper-parameters

We tried $\eta = 0.0028$, $\lambda = 0.0067$ and $\eta = 5.9438 \cdot 10^{-4}$, $\lambda = 1.5142 \cdot 10^{-4}$ (the first 2 combinations) and trained the network by 30 epochs. Below are the training results. After 30 epochs they obtained 19.53% and 13.16% accuracy respectively.

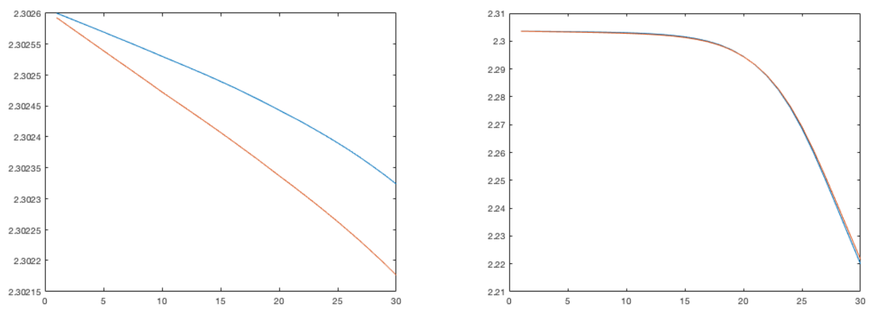

Figure 1: Training results using $\eta = 5.9438 \cdot 10^{-4}$, $\lambda = 1.5142 \cdot 10^{-4}$ (left) and $\eta = 0.0028$, $\lambda = 0.0067$ (right).