Assignment 1 Report

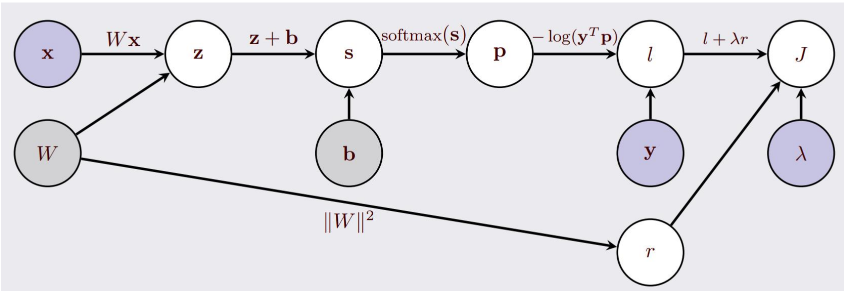
Name: Qiao Ren
Date: 2020 March
Course: DD2424 Deep Learning in Data Science

Part 1 Gradient Computing

Two ways can be used in computing gradient: numerically and analytically computing. The formula of them are written bellow.

1) Cost function:

    Cost Function
    = CrossEntropyLoss($P_{\text{predicted probability on the true class}}$, $Y_{\text{true class}}$) + Regularization
    = CrossEntropyLoss(Softmax(WX+b), $Y_{\text{true class}}$) + Regularization



2) numerically computed gradient:

$$\frac{\partial \cos t}{\partial W} = \frac{Cost(W+\Delta W) - Cost(W)}{\Delta W} \qquad \frac{\partial \cos t}{\partial b} = \frac{Cost(b+\Delta b) - Cost(b)}{\Delta b}$$

In which $\Delta W = 1\text{e-}6, \Delta b = 1\text{e-}6$

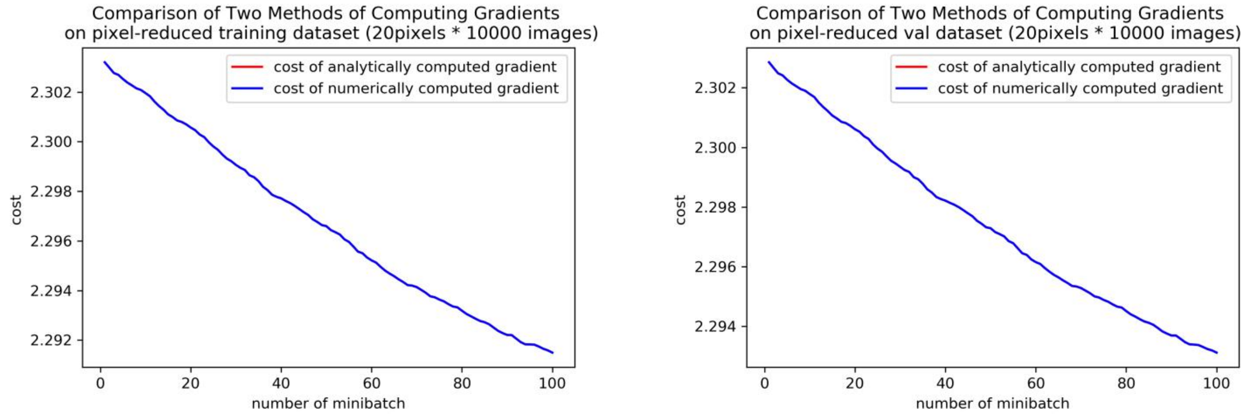
3) analytically computed gradient:

$$\begin{aligned} &\frac{\partial \cos t}{\partial W} \\ &= \frac{\partial \{loss + regularization\}}{\partial W} \\ &= \frac{\partial loss}{\partial W} + \frac{\partial regularization}{\partial W} \\ &= \frac{\partial loss}{\partial W} + 2\lambda W \\ &= \frac{1}{n_{BatchSize}} G_{batch} X_{batch}^T + 2\lambda W \end{aligned}$$

$$\begin{aligned} &\frac{\partial \cos t}{\partial b} \\ &= \frac{\partial \{loss + regularization\}}{\partial b} \\ &= \frac{\partial loss}{\partial b} + \frac{\partial regularization}{\partial b} \\ &= \frac{\partial loss}{\partial b} + 0 \\ &= \frac{1}{n_{BatchSize}} G_{batch} 1_{n_{BatchSize}} \end{aligned}$$

In which, G_{batch} = - (Y_{batch} - P_{batch}) ,
P_{batch} = SoftMax (WX_{batch} + b_{batch}),
$n_{BatchSize}$ = how many images in each mini batch.

In my experiment, the cost of analytically (g_a) and numerically (g_n) computed gradient are very close. It is shown in the graph bellow. The difference between g_a and g_n is e-12 on training data set. The difference is e-11 on validation data set. This means that my code of computing gradient analytically is correct.

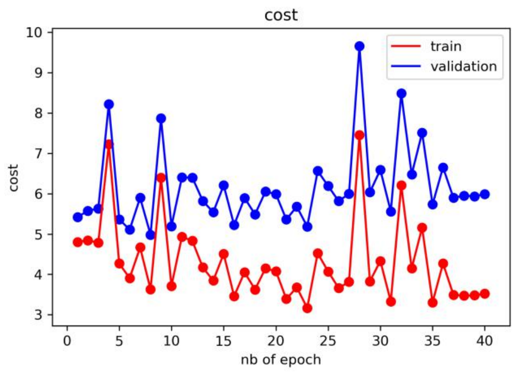


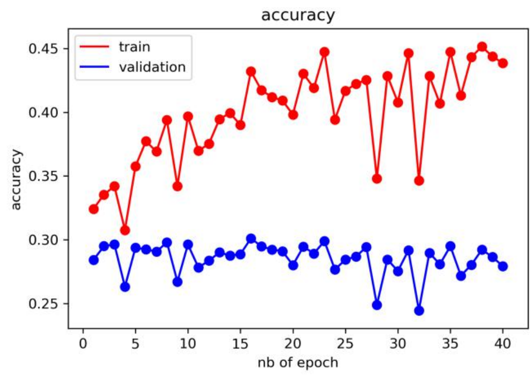
Comparison between analytically (g_a) and numerically (g_n) computed gradient,
on training data set (left) and validation data set (right)
(Learning rate =0.001, lambda = 1, number of images in each minibatch =100)
Here I reduced the original data set from 3072 pixels to 20 pixels, in order to speed up the gradient computing.

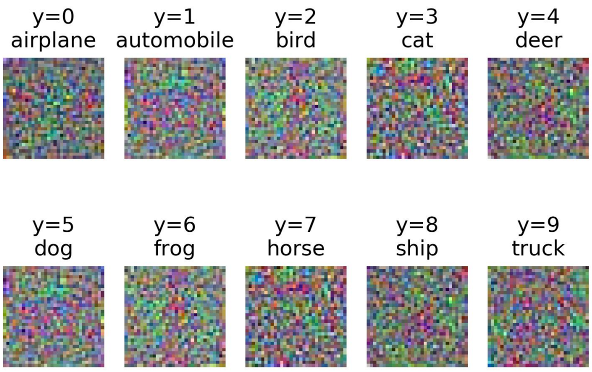
Part 2 Network Training

I trained the network with analytically computed gradient descent algorithm. Because it is faster than numerically computed gradient. The following table shows the cost graph, the accuracy graph and the weight visualization, when different parameters are given.

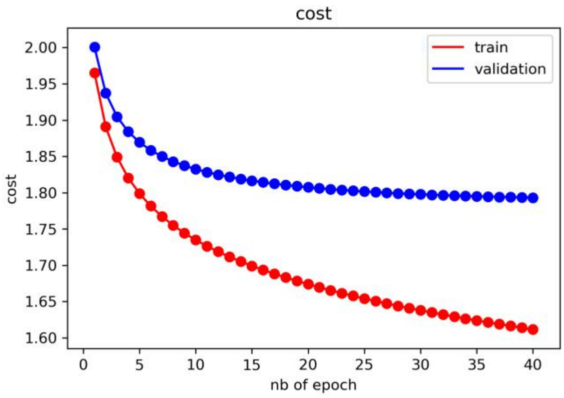
1) lambda=0, epochs=40, batch=100, eta=.1

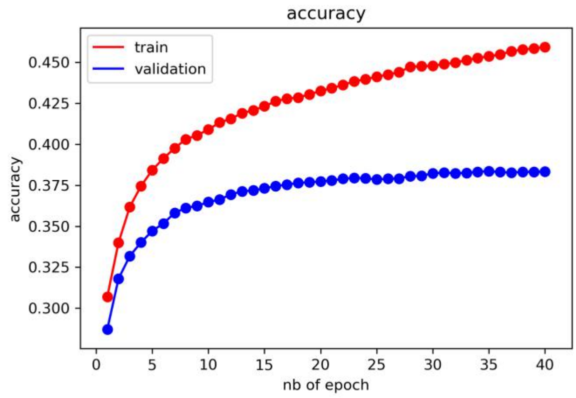


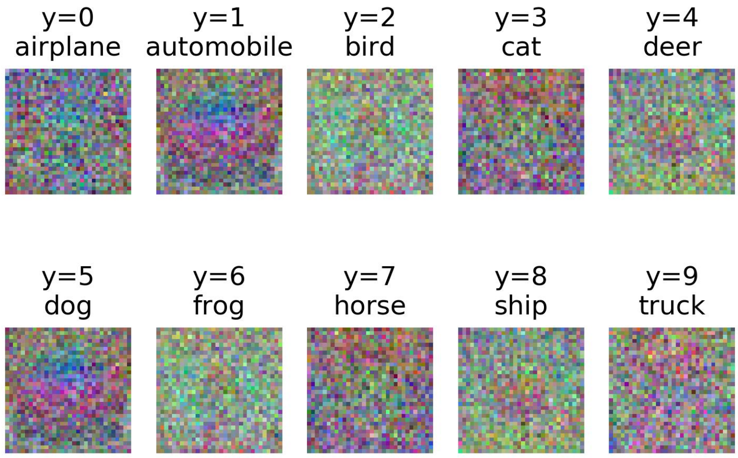




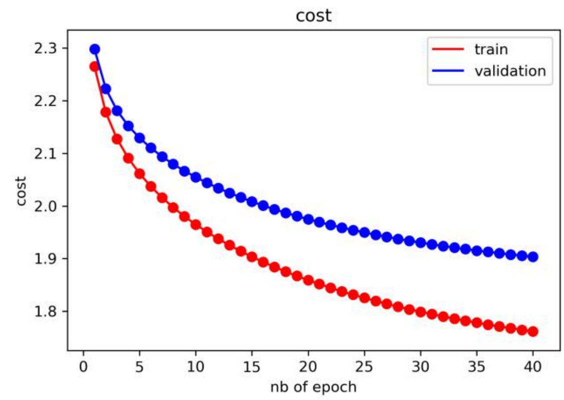
2) lambda=0, epochs=40, batch=100, eta=.001

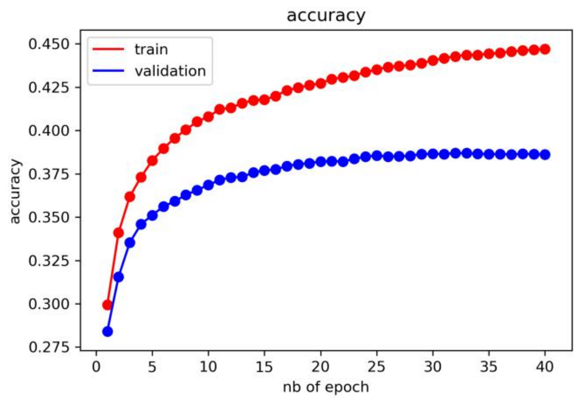


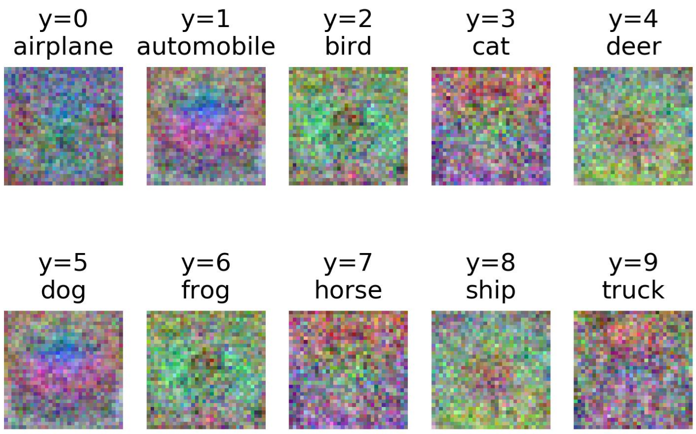




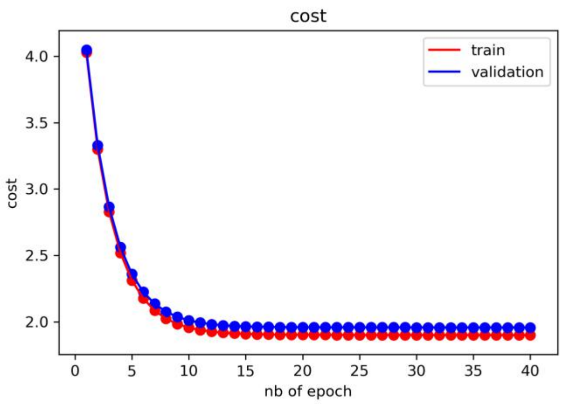
3) lambda=.1, epochs=40, batch=100, eta=.001

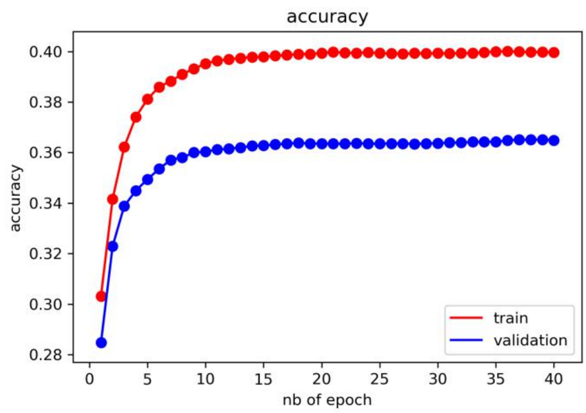


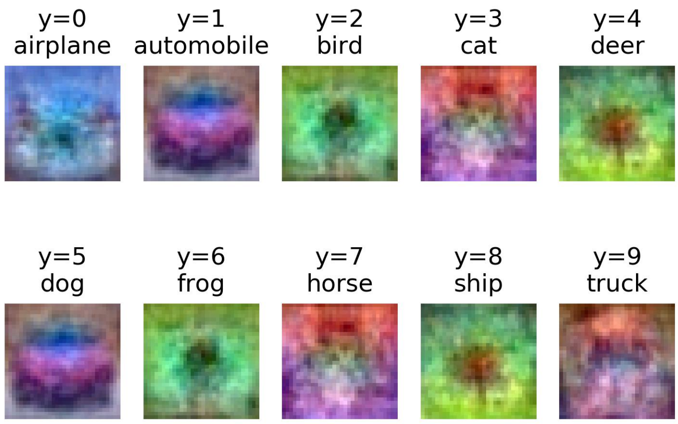




4) lambda=1, epochs=40, batch=100, eta=.001







The following table shows the cost and accuracy when different parameters are given. Eta is learning rate. Lambda is the parameter of regularization term.

Parameters (common parameters: epochs=40, batch size=100)	Cost on Training dataset	Cost on Validation dataset	Accuracy on training dataset	Accuracy on validation dataset	Accuracy on test dataset
lambda=0, eta=0.1	3.5215	5.9854	43.86%	27.93%	28.67 %
lambda=0, eta=0.001	1.6117	1.7932	45.92%	38.33%	39.11%
lambda=0.1, eta=0.001	1.7618	1.9036	44.70 %	38.62 %	39.17%
lambda=1, eta=0.001	1.8995	1.9570	39.96%	36.49 %	37.51%

Discussion:

The effect of Regularization: regularization penalizes the complexity of neural network. Large value of lambda generates big regularization, which is a big penalization. Large lambda makes the cost to be big. So, to respond to a large cost, the network will reduce the complexity of weight. the optimal lambda in the above 4 experiment is 0 or 0.1. When lambda is too big (for example =1), the accuracy drops.

The effect of Learning rate: the optimal learning rate in the above 4 experiment is 0.001. When learning rate is too big, the network would cross over the minimum of cost function. When learning rate is too small, the network have too take a long time to reach the minimum and it might stop at a local minimum. This explains why, when the learning rate =0.1, the accuracy is low.