COMP 540 Assignment #5

Yunda Jia yj32
Yu Wu yw92

1. Deep neural networks

a) Although shallow networks could perform as good as the deeper ones, the reason why deep networks outperform might be a shallow network could need more neurons and with the task complexity grows, shallow networks might needs more units and become extreme bigger than deep networks[1].

b) One problem we see in ReLU is the Dying ReLU problem where some ReLU Neurons essentially die for all inputs and remain inactive no matter what input is supplied, here no gradient flows and if large number of dead neurons are there in a Neural Network it's performance is affected, this can be corrected by making use of what is called Leaky ReLU where slope is changed left of x=0 in above figure and thus causing a leak and extending the range of ReLU.[2]

c) Because the purpose of the pooling layers is to achieve spatial invariance by reducing the resolution of the feature maps. Each pooled feature map corresponds to one feature map of the previous layer. [3]

d)

Architecture	Number of layers	Top-5 error rate	Salient Feaure
Alexnet	8	15.3%	Deeper
VGG-Net	19	7.3%	Fixed_sized kernals
GooleNet	22	6.67%	Efficient inception module; No FC layers
ResNet	152	3.6%	Shortcut connection

Reference:

[1]Ian Goodfellow DeepLearning
[2] https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e
[3] Scherer D., Müller A., Behnke S. (2010) Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In: Diamantaras K., Duch W., Iliadis L.S. (eds) Artificial Neural Networks – ICANN 2010. ICANN 2010. Lecture Notes in Computer Science, vol 6354. Springer, Berlin, Heidelberg

2. Decision trees, entropy

a) $\frac{\partial H}{\partial q} = -\log q + 1 + \log(1-q) - 1 = 0$

$$\log(1-q) = \log q$$

$$1-q = q$$

$$q = \frac{1}{2}$$

let $q = \frac{p}{p+n}$, when $p=n$, $\frac{p}{p+n} = \frac{1}{2}$

$\Rightarrow H(S) = -\frac{1}{2}\log\frac{1}{2} - (1-\frac{1}{2})\log\frac{1}{2} = 1$

b) ① Misclassification

A: $\frac{100+100}{400+400} = \frac{1}{4}$

Reduction: $\frac{1}{2} - (\frac{400}{800} \times \frac{1}{4} + \frac{400}{800}\,4) = \frac{1}{4}$

B: $\frac{200}{200+400} = \frac{1}{3}$, $\frac{100}{100+300} = \frac{1}{4}$, $\frac{0}{200} = 0$

Reduction: $\frac{1}{2} - (\frac{600}{800} \cdot \frac{1}{3} + \frac{200}{800} \times 0] = \frac{1}{4}$

② Entropy

A: $H(D_1) = -\frac{3}{4}\log\frac{3}{4} - (1-\frac{3}{4})\log(1-\frac{3}{4}) = 0.811$

$H(D_2) = 0.811$

entropy gain $= 1 - \frac{1}{2}\times 0.811 - \frac{1}{2}\times 0.811 = 0.189$

B: $H(D_1) = -\frac{2}{3}\log(\frac{2}{3}) - (1-\frac{2}{3})\log(1-\frac{2}{3}) = 0.918$

$H(D_2) = 0$

$1 - \frac{600}{800}\times 0.918 - 0 = 0.311$

$\Rightarrow$ B is better

③ Gini

A: $G(D_1) = 2 \times \frac{1}{4} \times \frac{3}{4} = \frac{3}{8}$

$G(D_2) = 2 \times \frac{3}{4} \times \frac{1}{4} = \frac{3}{8}$

gini index A : $\frac{1}{2} - \left(\frac{1}{2} \times \frac{3}{8} + \frac{1}{2} \times \frac{3}{8}\right) = \frac{1}{8}$

B: $G(D_1) = \frac{4}{9}$ $G(D_2) = 0$

gini index B : $\frac{1}{2} - \frac{1}{2} \times \frac{4}{9} = \frac{1}{6}$

$\Rightarrow$ Model B is better

c) Splitting features won't increase misclassification rate

since splitting a node does not lead to a information gain

e.g. Assume root node P with C children, each with p_i positive

q_i negative examples and $i = 1, \cdots, C$

$$E_p = \frac{\min(\sum p_i, \sum q_i)}{\sum_{i=1}^{c} (p_i + q_i)} \qquad E_c = \frac{\sum \min(p_i, q_i)}{\sum (p_i + q_i)}$$

$\because \min(p_i, q_i) \le p_i$, $\min(p_i, q_i) \le q_i$

$\Rightarrow m(\sum p_i, \sum q_i) \le \sum \min p_i$

$\Rightarrow \min(\sum p_i, \sum q_i) \le \sum \min(p_i, q_i)$

shows that misclassification error rate will not increase when splitting features.

4 Fully connected neural networks and convolutional neural networks (20 extra credit points) (150 points)

4.1 Fully connected feedforward neural networks: a modular approach

4.1.1 Affine layer: forward (5 points)

Testing affine_forward function:
difference: 9.769847728806635e-10

4.1.2 Affine layer: backward (5 points)

Testing affine_backward function:
dx error: 1.48702666670524e-10
dtheta error: 6.965290680293478e-10
dtheta0 error: 3.275578582297488e-12

4.1.3 ReLU layer: forward (2 points)

Testing relu_forward function:
difference: 4.999999798022158e-08

4.1.4 ReLU layer: backward (3 points)

Testing relu_backward function:
dx error: 3.2756115125730404e-12

Testing affine_relu_backward:
dx error: 5.684387515941462e-11
dtheta error: 1.680798196555248e-10
dtheta0 error: 3.275631801255082e-12

Testing svm_loss:
loss: 9.00039925749647
dx error: 1.4021566006651672e-09

Testing softmax_loss:
loss: 2.30262543728152
dx error: 7.270166644468682e-09

4.1.5 Two layer network (5 points)

Testing initialization ...
Testing test-time forward pass ...
Testing training loss (no regularization)
Running numeric gradient check with reg = 0.0
theta1 relative error: 1.52e-08
theta1_0 relative error: 6.55e-09
theta2 relative error: 3.48e-10
theta2_0 relative error: 4.33e-10
Running numeric gradient check with reg = 0.7
theta1 relative error: 8.18e-07
theta1_0 relative error: 1.09e-09
theta2 relative error: 2.85e-08
theta2_0 relative error: 9.09e-10

4.1.6 Overfitting a two layer network (5 points)

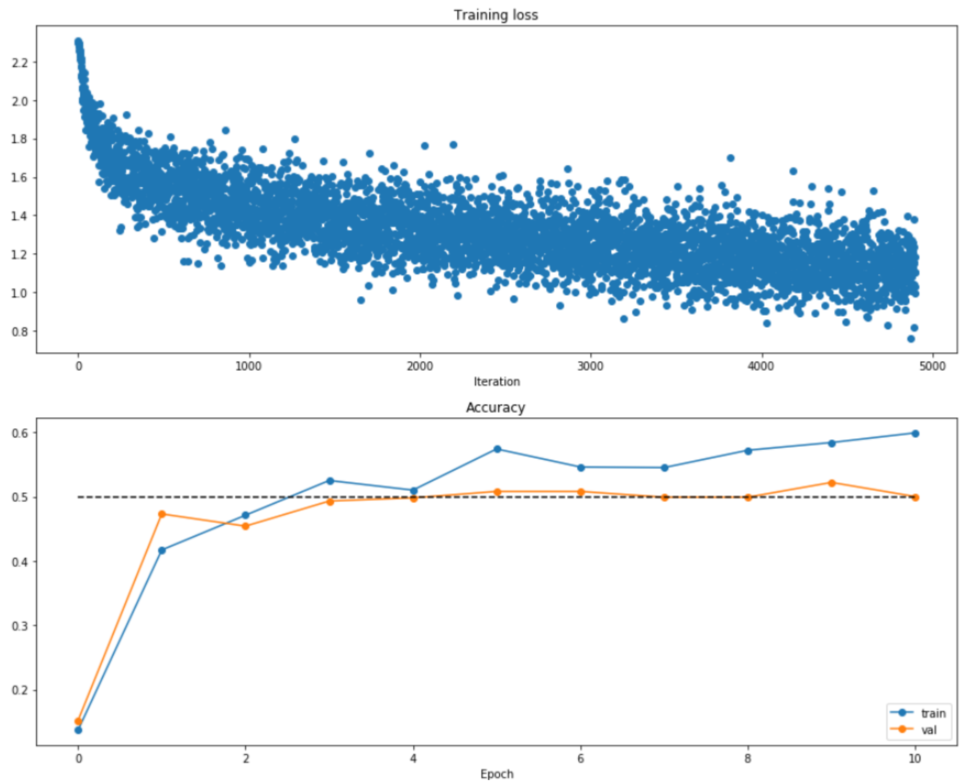

Figure 1: Loss and Accuracy

4.1.7 Multilayer network (10 points)

Running check with reg = 0
Initial loss: 2.300313187547858
theta1 relative error: 3.11e-07
theta1_0 relative error: 6.08e-07
theta2 relative error: 1.43e-06
theta2_0 relative error: 6.77e-08
theta3 relative error: 3.64e-07
theta3_0 relative error: 1.00e-10
Running check with reg = 3.14
Initial loss: 5.7580916578447905
theta1 relative error: 7.88e-08
theta1_0 relative error: 4.63e-08
theta2 relative error: 7.70e-08
theta2_0 relative error: 1.07e-08
theta3 relative error: 1.00e+00
theta3_0 relative error: 1.52e-10

4.1.8 Overfitting a three layer network (2 points)

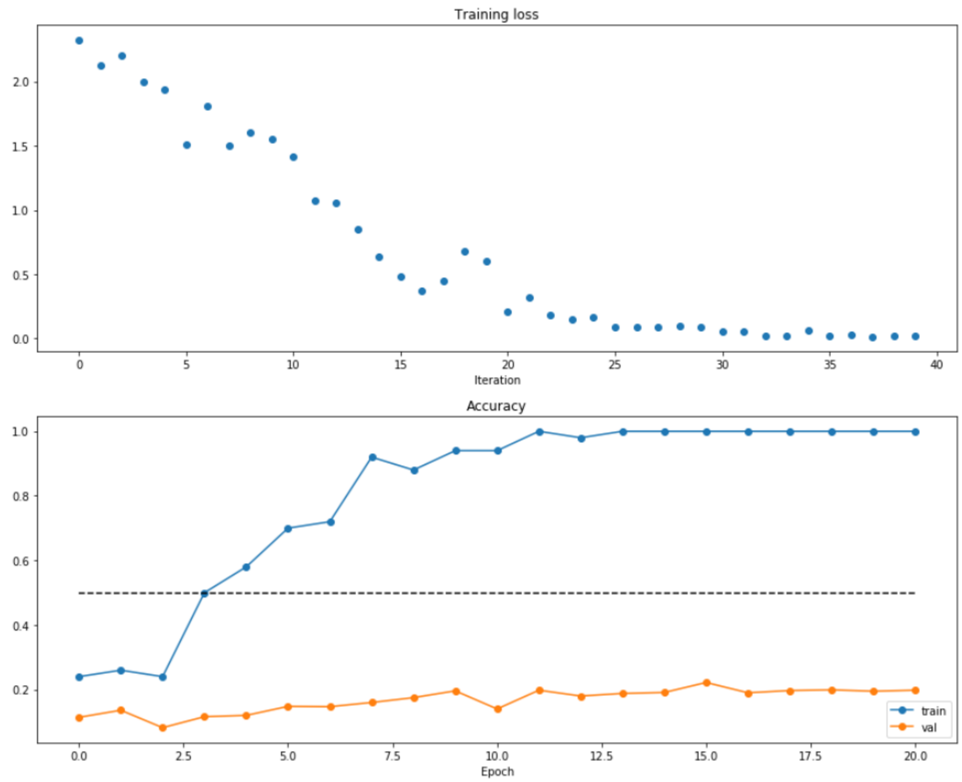

Figure 2: Loss and Accuracy

4.1.9 Overfitting a five layer network (3 points)

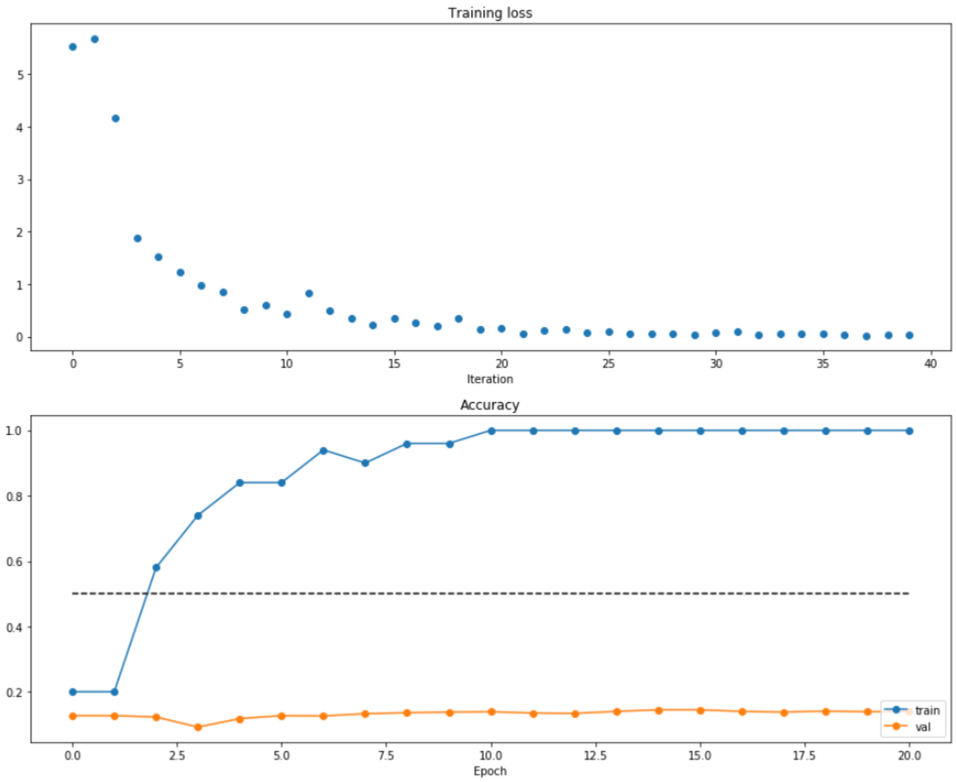

Figure 3: Loss and Accuracy

Answer:
Training five layer net is hard to improve validation accuracy because it is easier to overfit.

4.1.10 GD+Momentum (5 points)

next_theta error: 8.882347033505819e-09
velocity error: 4.269287743278663e-09

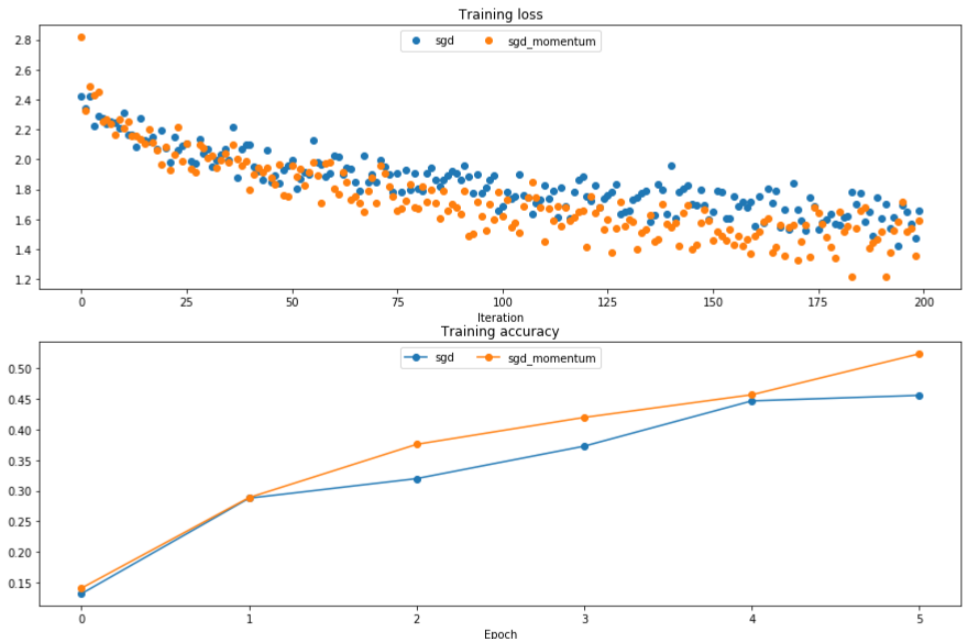

Figure 4: Loss and Accuracy

4.1.11 RMSProp (5 points)

next_theta error: 9.524687511038133e-08
cache error: 2.6477955807156126e-09

4.1.12 Adam (5 points)

next_theta error: 1.1395691798535431e-07
v error: 4.208314038113071e-09
m error: 4.214963193114416e-09

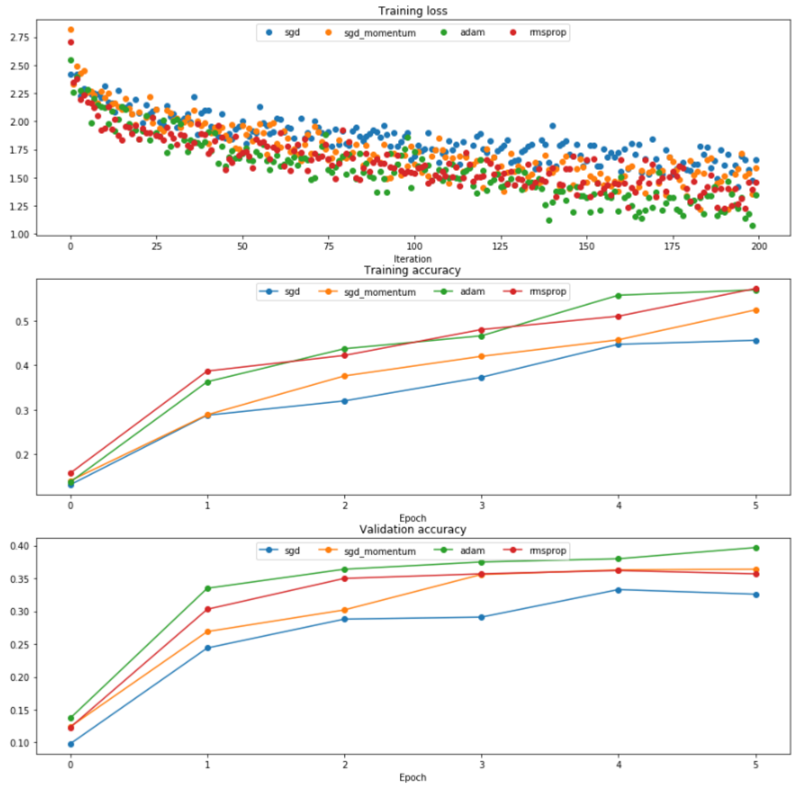

Figure 5: Loss and Accuracy

4.2 Dropout

4.2.1 Dropout forward pass (5 points)

Running tests with p = 0.3
Mean of input: 9.999944062082756
Mean of train-time output: 9.991211707362872
Mean of test-time output: 9.999944062082756
Fraction of train-time output set to zero: 0.3005
Fraction of test-time output set to zero: 0.0

Running tests with p = 0.6
Mean of input: 9.999944062082756
Mean of train-time output: 9.969972097801993
Mean of test-time output: 9.999944062082756
Fraction of train-time output set to zero: 0.601104
Fraction of test-time output set to zero: 0.0

Running tests with p = 0.75
Mean of input: 9.999944062082756
Mean of train-time output: 9.989474533174528
Mean of test-time output: 9.999944062082756
Fraction of train-time output set to zero: 0.750284
Fraction of test-time output set to zero: 0.0

4.2.2 Dropout backward pass (5 points)

dx relative error: 1.892906517448476e-11

4.2.3 Fully connected nets with dropout (5 points)

Running check with dropout = 0
Initial loss: 2.3051948273987857
theta1 relative error: 2.53e-07
theta1_0 relative error: 2.94e-06
theta2 relative error: 1.50e-05
theta2_0 relative error: 5.05e-08
theta3 relative error: 2.75e-07
theta3_0 relative error: 1.17e-10

Running check with dropout = 0.25
Initial loss: 2.310865407789547
theta1 relative error: 3.40e-07
theta1_0 relative error: 6.89e-08
theta2 relative error: 2.91e-07
theta2_0 relative error: 3.77e-09
theta3 relative error: 1.77e-07
theta3_0 relative error: 1.68e-10

Running check with dropout = 0.5
Initial loss: 2.302437587710995
theta1 relative error: 4.55e-08
theta1_0 relative error: 1.87e-08
theta2 relative error: 2.97e-08
theta2_0 relative error: 5.05e-09
theta3 relative error: 4.34e-07
theta3_0 relative error: 8.01e-11

4.2.4 Experimenting with fully connected nets with dropout (5 points)

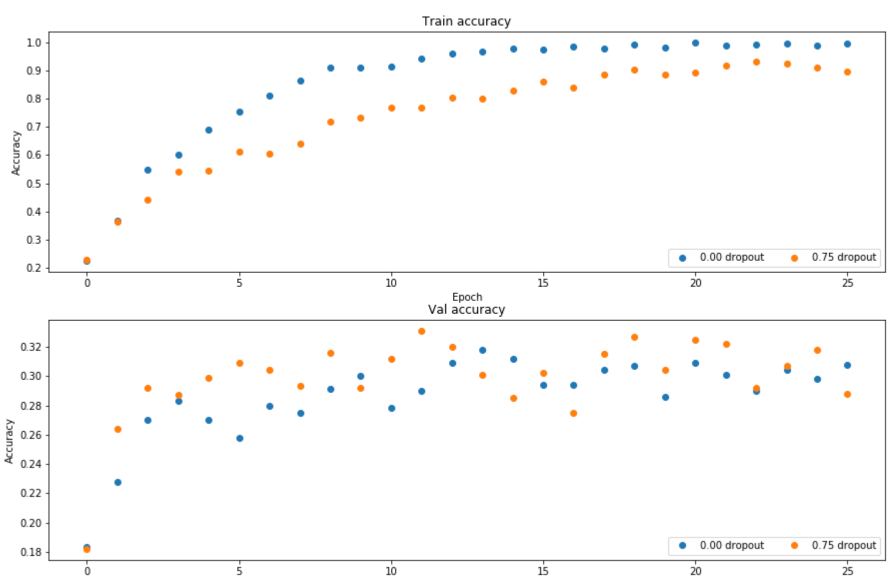

Figure 6: Loss and Accuracy

4.3 Training a fully connected network for the CIFAR-10 dataset with dropout (10 points)

4.4 Convolutional neural networks

4.4.1 Convolution: naive forward pass (10 points)

Testing conv_forward_naive
difference: 2.2121476417505994e-08

4.4.2 Convolution: naive backward pass (10 points)

Testing conv_backward_naive function
dx error: 5.605049280767101e-09
dtheta error: 8.981928317952683e-10
dtheta0 error: 1.3855783007788618e-11

4.4.3 Max pooling: naive forward pass (5 points)

Testing max_pool_forward_naive function:
difference: 4.1666665157267834e-08

4.4.4 Max pooling: naive backward pass (5 points)

Testing max_pool_backward_naive function:
dx error: 3.275632601200459e-12

4.4.5 Three layer convolutional neural network (10 points)

4.4.6 Train the CNN on the CIFAR-10 data (5 points)

4.4.7 Train the best model for CIFAR-10 data (20 points)

4.4.8 Train the best model for CIFAR-10 data using PyTorch (20 points)

See my implementation in PyTorch.ipynb, I got 77.80% accuracy on test set.