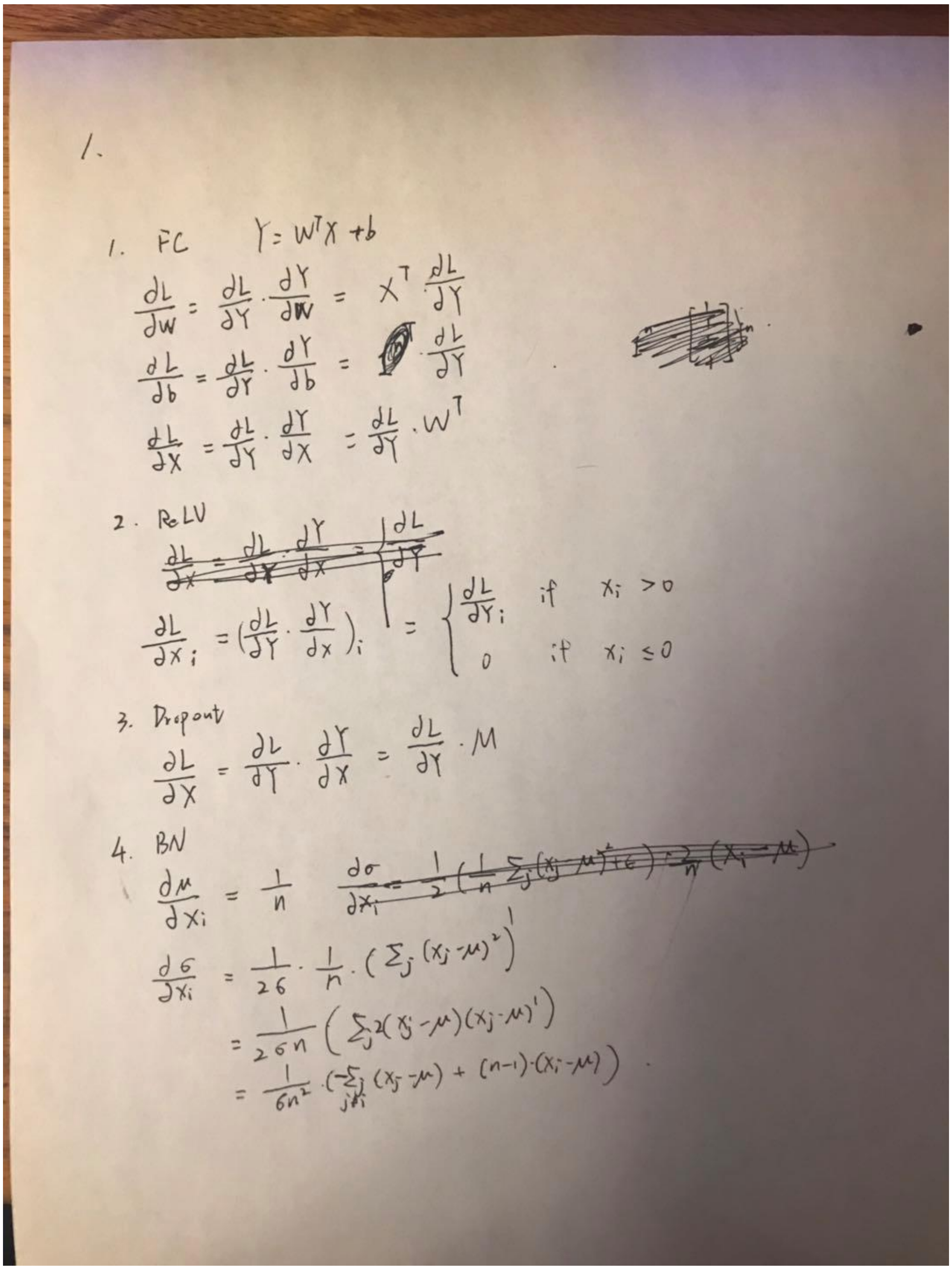

1.

1. FC $Y = W^T X + b$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Y} \cdot \frac{\partial Y}{\partial W} = X^T \frac{\partial L}{\partial Y}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial Y} \cdot \frac{\partial Y}{\partial b} = [illegible] \cdot \frac{\partial L}{\partial Y}$$

[illegible]

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} \cdot \frac{\partial Y}{\partial X} = \frac{\partial L}{\partial Y} \cdot W^T$$

2. ReLU

~~$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} \cdot \frac{\partial Y}{\partial X} = \frac{\partial L}{\partial Y}$~~

$$\frac{\partial L}{\partial x_i} = \left(\frac{\partial L}{\partial Y} \cdot \frac{\partial Y}{\partial x}\right)_i = \begin{cases} \frac{\partial L}{\partial Y_i} & \text{if } x_i > 0 \\ 0 & \text{if } x_i \le 0 \end{cases}$$

3. Dropout

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} \cdot \frac{\partial Y}{\partial X} = \frac{\partial L}{\partial Y} \cdot M$$

4. BN

$$\frac{\partial \mu}{\partial x_i} = \frac{1}{n}$$

~~$\frac{\partial \sigma}{\partial x_i} = \frac{1}{2}\left(\frac{1}{n}\sum_j (x_j - \mu)^2 + \epsilon\right) \cdot \frac{2}{n}(x_i - \mu)$~~

$$\frac{\partial \sigma}{\partial x_i} = \frac{1}{2\sigma} \cdot \frac{1}{n} \cdot \left(\sum_j (x_j - \mu)^2\right)'$$

$$= \frac{1}{2\sigma n}\left(\sum_j 2(x_j - \mu)(x_j - \mu)'\right)$$

$$= \frac{1}{\sigma n^2} \cdot \left(-\sum_{j \ne i} (x_j - \mu) + (n-1) \cdot (x_i - \mu)\right).$$

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial Y_i} \cdot \frac{\partial Y_i}{\partial x_i}$$

$$= \cancel{\frac{\partial L}{\partial Y}\left(\gamma - \frac{(x_i-\mu)\frac{d\sigma}{\partial x_i}}{\sigma^2}\right)}$$

$$= \frac{\partial L}{\partial Y}\gamma \cdot \frac{\frac{\partial(x_i-\mu)}{\partial x_i}\sigma - (x_i-\mu)\cdot\frac{d\sigma}{\partial x_i}}{\sigma^2}$$

$$= \frac{\partial L}{\partial Y}\cdot\gamma\frac{1}{\sigma^2}\left(\sigma\cdot\left(1-\frac{1}{n}\right) - (x_i-\mu)\cdot\frac{1}{\sigma n^2}\cdot\left(-\sum_{j \neq i}(x_j-\mu)+(n-1)(x_i-\mu)\right)\right)$$

5. Conv

$$H-H'+1+2(H'-1) = H+H'-1$$

$$W-W'+1+2(W'-1) = W+W'-1$$

$$Y_{n,f} \; [illegible] = \sum_c X_{n,c} *_{\text{valid}} \overline{W_{f,c}}$$

$$(Y_{n,f})_{ij} = \sum_{m',n'} X_{m',n'} \cdot \overline{W}_{i-m'+H',\, j-n'+W'}$$

$$\left(\frac{\partial L}{\partial X_{n,c}}\right)_{ij} = \left(\frac{\partial L}{\partial Y_{n,f}} \cdot \frac{\partial Y_{n,f}}{\partial X_{n,c}}\right)_{ij} = \sum_f \sum_{m',n'} W_{m',n'} \left(\frac{\partial L}{\partial Y_{n,f}}\right)_{i-m'+1,\, j-n'+1}$$

$$\frac{\partial L}{\partial X_{n,c}} = \sum_f W_{f,c} *_{\text{full}} \frac{\partial L}{\partial Y_{n,f}}$$

$$\left(\frac{\partial L}{\partial W_{f,c}}\right)_{ij} = \left(\frac{\partial L}{\partial Y_{n,f}} \cdot \frac{\partial Y_{n,f}}{\partial W_{fc}}\right)_{ij} = \sum_n \cdot \sum_{m',n'} X_{nc\; 1+m'-1,\, j+n'-1} \cdot \left(\frac{\partial L}{\partial Y_{n,f}}\right)_{m'n'}$$

$$\frac{\partial L}{\partial W_{f,c}} = \sum_n X_{n,c} *_{\text{filt}} \frac{\partial L}{\partial Y_{n,f}}$$

EECS 498/598 Deep Learning HW1

Fei Yi 53455316 feiyi@umich.edu

1. Logistic regression and beyond: Binary classification with a sigmoid output layer
    a. No hidden units:
        i. Performance: train acc: 0.964000; val acc: 0.952000
        ii. Model: input_dim=20, hidden_dim=None, weight_scale=1e-3, reg=0.2
        iii. Solver: update rule: sgd_momentum, learning_rate=1, lr_decay=0.95, num_epochs=20, batch_size = 50
    b. With hidden units:
        i. Performance: train acc: 0.962000; val_acc: 0.952000
        ii. Model: input_dim=20, hidden_dim=32, weight_scale=1e-3, reg=0.3
        iii. Solver: update rule: sgd_momentum, learning_rate=8e-2, lr_decay=0.9, num_epochs=20, batch_size = 50

2. SVM and beyond: binary classification with a hinge-loss output layer
    a. No hidden units:
        i. Performance: train acc: 0.964000; val_acc: 0.944000
        ii. Model: input_dim=20, hidden_dim=None, weight_scale=1e-3, reg=0.1
        iii. Solver: update rule: sgd_momentum, learning_rate=1, lr_decay=0.95, num_epochs=50, batch_size = 50
    b. With hidden units:
        i. Performance: train acc: 0.968000; val_acc: 0.928000
        ii. Model: input_dim=20, hidden_dim=32, weight_scale=1e-3, reg=0.3
        iii. Solver: update rule: sgd_momentum, learning_rate=1e-1, lr_decay=0.95, num_epochs=50, batch_size = 50

3. Softmax regression and beyond: multi-class classification with a softmax output layer
    a. No hidden units:
        i. Performance: train acc: 0.953000; val_acc: 0.935556
        ii. Model: input_dim=784, hidden_dim=None, weight_scale=1e-3,

reg=0.01

iii. Solver: update rule: rmsprop, learning_rate=5e-5, lr_decay=0.9, num_epochs=50, batch_size = 100

b. With hidden units:
   i. Performance: train acc: 0.993000; val_acc: 0.977037
   ii. Model: input_dim=784, hidden_dim=64, weight_scale=1e-3, reg=0.01
   iii. Solver: update rule: rmsprop, learning_rate=1e-4, lr_decay=0.9, num_epochs=50, batch_size = 100

4. Convolutional Neural Network for multi-class classification

a. No batch normalization and drop out:
   i. Performance: train acc: 0.984000; val_acc: 0.976200
   ii. Model: input_dim=(1,28,28), num_filters=5, filter_size =7, hidden_dim=16, num_classes=10, weight_scale=1e-3, reg=0
   iii. Solver: update rule: sgd_momentum, learning_rate=1e-3, lr_decay=0.9, num_epochs=50, batch_size = 100

b. With batch normalization and drop out:
   i. Performance: train acc: 0.914000; val_acc: 0.901200
   ii. Model: input_dim=(1,28,28), num_filters=8, filter_size =7, hidden_dim=16, num_classes=10, weight_scale=1e-3, reg=0
   iii. Solver: update rule: sgd_momentum, learning_rate=1e-2, lr_decay=0.9, num_epochs=50, batch_size = 100

5. Convolutional Neural Networks for CIFAR10 image classification.

a. Performance: accuracy: 74%

b. Architecture Improvement:
   i. Double the filter_num of first two conv layers
   ii. Add batch normalization and relu after each conv layers
   iii. Add drop out after first fc layer

c. Solver:
   i. Learning_rate = 0.001
   ii. Batch_size = 100
   iii. Criterion = crossentropy loss
   iv. Optimizer = adam

6. Short answer questions

a. Sigmoid projects large inputs to 0 or 1, causing its dsigmoid = sigmoid * (1 – sigmoid) to vanish. In DNN, the gradient vanishing is even more severe due to chain rule, resulting in difficulty in training.

b. When perform W in testing, multiply x by p to account for missing activations in training.
c. I will reduce learning rate.