EECS 498/598 Deep Learning HW1

Fei Yi 53455316 feiyi@umich.edu

1. Logistic regression and beyond: Binary classification with a sigmoid output layer
   1. No hidden units:
      1. Performance: train acc: 0.964000; val acc: 0.952000
      2. Model: input\_dim=20, hidden\_dim=None, weight\_scale=1e-3, reg=0.2
      3. Solver: update rule: sgd\_momentum, learning\_rate=1, lr\_decay=0.95, num\_epochs=20, batch\_size = 50
   2. With hidden units:
      1. Performance: train acc: 0.962000; val\_acc: 0.952000
      2. Model: input\_dim=20, hidden\_dim=32, weight\_scale=1e-3, reg=0.3
      3. 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
   1. No hidden units:
      1. Performance: train acc: 0.964000; val\_acc: 0.944000
      2. Model: input\_dim=20, hidden\_dim=None, weight\_scale=1e-3, reg=0.1
      3. Solver: update rule: sgd\_momentum, learning\_rate=1, lr\_decay=0.95, num\_epochs=50, batch\_size = 50
   2. With hidden units:
      1. Performance: train acc: 0.968000; val\_acc: 0.928000
      2. Model: input\_dim=20, hidden\_dim=32, weight\_scale=1e-3, reg=0.3
      3. 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
   1. No hidden units:
      1. Performance: train acc: 0.953000; val\_acc: 0.935556
      2. Model: input\_dim=784, hidden\_dim=None, weight\_scale=1e-3, reg=0.01
      3. Solver: update rule: rmsprop, learning\_rate=5e-5, lr\_decay=0.9, num\_epochs=50, batch\_size = 100
   2. With hidden units:
      1. Performance: train acc: 0.993000; val\_acc: 0.977037
      2. Model: input\_dim=784, hidden\_dim=64, weight\_scale=1e-3, reg=0.01
      3. 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
   1. No batch normalization and drop out:
      1. Performance: train acc: 0.984000; val\_acc: 0.976200
      2. Model: input\_dim=(1,28,28), num\_filters=5, filter\_size =7, hidden\_dim=16, num\_classes=10, weight\_scale=1e-3, reg=0
      3. Solver: update rule: sgd\_momentum, learning\_rate=1e-3, lr\_decay=0.9, num\_epochs=50, batch\_size = 100
   2. With batch normalization and drop out:
      1. Performance: train acc: 0.914000; val\_acc: 0.901200
      2. Model: input\_dim=(1,28,28), num\_filters=8, filter\_size =7, hidden\_dim=16, num\_classes=10, weight\_scale=1e-3, reg=0
      3. 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.
   1. Performance: accuracy: 74%
   2. Architecture Improvement:
      1. Double the filter\_num of first two conv layers
      2. Add batch normalization and relu after each conv layers
      3. Add drop out after first fc layer
   3. Solver:
      1. Learning\_rate = 0.001
      2. Batch\_size = 100
      3. Criterion = crossentropy loss
      4. Optimizer = adam
6. Short answer questions
   1. 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.
   2. When perform W in testing, multiply x by p to account for missing activations in training.
   3. I will reduce learning rate.