CSC 580 Homework 4

Due: 12/05 (Mon) 3:00pm

Instructions:

- Submit your homework on time to gradescope. NO LATE DAYS, NO LATE SUBMISSIONS ACCEPTED.
- You must provide the derivation for obtaining the answer and full source code for whatever problem you use programming. Please include your source code in your gradescope submission whenever asked by the question; please also send a copy of your code to `csc580homeworks@gmail.com`.
- Tip: to make your code more efficient, use as many vectorized operations (instead of explicit loops) as possible.
- This is a self-paced homework, and the collaboration policy is different. You still need to do homework by yourself, but you are free to discuss, share parts of your code, or consult the solutions available online. However, for your learning outcome, please try to do it by yourself first, then consult with others. However, if your code is similar enough to other students' code or public codes (i.e., mindless copy-paste), you will still be subject to academic discipline.

Problem 1 (25pts).

Solve Q4 of Assignment 1 of Stanford CS231n: `https://cs231n.github.io/assignments2020/assignment1/#q4-two-layer-neural-network-25-points` Make sure that you are doing the 2020 version; answer the following questions.

Please follow the instructions in the link above to setup the environment, using Option A (Google Colab). You also need to learn how to save files correctly (described in the link above).

(a) For the block "Affine layer: forward", paste your implementation of function `affine_forward` here, as well as the result of the unit test.

(b) For the block "Affine layer: backward":

- Note that different from our lecture, the affine layer here includes a bias term b. In other words, using the notation in the class, at layer l, we have $a^{(l)} = W^{(l)} z^{(l-1)} + b^{(l)}$ instead of $a^{(l)} = W^{(l)} z^{(l-1)}$. With this change, what will need to be changed in the backpropagation algorithm in slide 22 of Lecture 13 "Neural Networks"?
- paste your implementation of function `affine_backward` here, as well as the result of the unit test.

(c) For the block "ReLU activation: forward", paste your implementation of function `relu_forward` here, as well as the result of the unit test.

(d) For the block "ReLU activation: backward", paste your implementation of function `relu_backward` here, as well as the result of the unit test.

(e) Answer Inline Question 1.

(f) For the block "Sandwich layers", paste the result of the unit test here.

(g) For the block "Loss layers: Softmax and SVM"; let C be the number of classes in a classification problem.

- The `svm_loss` here refers to following loss for multiclass classification: given a C-dimensional classification score $s = (s_1, \dots, s_C)$ and label $y \in \{1, \dots, C\}$, the SVM loss is defined as:

$$\ell_{\text{SVM}}(s, y) = \sum_{c \neq y} (1 - s_y + s_c)_+$$

  Explain why this is a generalization of the binary classification hinge loss. Paste your implementation of function `svm_loss` here, as well as the result of the unit test.

- The `softmax_loss` here refers to the composition of softmax and log loss for multiclass classification (also known as the multiclass logisitc loss): $s = (s_1, \dots, s_C)$ and label $y \in \{1, \dots, C\}$, the softmax loss is defined as:

$$\ell_{\text{Softmax}}(s, y) = \ln\left(1 + \sum_{c \neq y} e^{s_c - s_y}\right) = -\ln \sigma(s)_y,$$

  where $\sigma(s) = \frac{1}{\sum_{c=1}^{C} e^{s_c}} (e^{s_1}, \dots, e^{s_C})$ is the softmax operation. Explain why this is a generalization of the binary classification logistic loss. Paste your implementation of function `softmax_loss` here, as well as the result of the unit test.

(h) For the block "Two-layer network", paste your implementation of class `TwoLayerNet` here, as well as the result of the unit test. Here, note that the ℓ_2 regularization is only imposed on

`W1`, `W2`, not on `b1`, `b2`.

(i) For the block "Solver", paste your source code for model training that can achieve a 36% validation accuracy.

(j) For the block "Debug the training", paste the three figures here after running the code.

(k) For the block "Tune your hyperparameters" and "Test your model!", paste your code for hyperparameter tuning and report your best model's validation and test accuracy.

(l) Answer Inline Question 2.

Problem 2 (10pts, extra credit).

(This problem's score counts towards your final homework score; however your final homework score is still capped at 40%. So if you did well in your previous homeworks, solving this problem will have little effect on your final score.)

Based on the CIFAR-10 dataset obtained in Problem 1, train a convolutional neural network on it and report its training, validation, and test errors; paste your source code here. You can use any deep learning framework(e.g. PyTorch used in the assigned reading), any convolutional neural network architecture (e.g. AlexNet, VGG), and any hyperparameter settings you like. To get full credit, your validation accuracy should be at least 50%.