Introduction to Artificial Intelligence (CS470): Assignment1

Deadline: September 18th, 2022

Setup

Option A: Google Colaboratory (Recommended)

Please download a starter code containing Colab notebooks code address.

1. Unzip the starter code `.zip` file. You should see a `cs470_assignment1` folder. Create a folder in your personal Google Drive and upload `cs470_assignment1` folder to the Drive folder.

2. Each Colab notebook (e.g., files ending in `.ipynb`) corresponds to an assignment problem. In Google Drive, double click on the notebook and select the option to open with Colab.

3. Once you have completed the assignment problem, you can save your edited files back to your Drive and move on to the next problem. Please ensure you are periodically saving your notebook File → Save so that you don't lose your progress.

Option B: Local Development

1. Please download a starter code containing Jupyter notebooks code address.

2. Activate your environment and run

```
pip install -r requirements.txt
```

3. Download the CIFAR-10 dataset. You can run the following from the `cs470\_assignment1` directory:

```
./get_datasets.sh
```

4. Please start the Jupyter server from the `cs470_assignment1` directory by executing Jupyter notebook in your terminal.

Problem 1: Multi-Layer Perceptron (MLP) [80pts]

In this problem, you will implement a fully-connected neural network on the CIFAR-10 dataset for an image classification task. The network is a two-layer network with an activation and softmax classifier:

$$Linear \rightarrow Activation \rightarrow Linear \rightarrow SoftMax.$$

You will also implement forward and backward passes with gradient computations. Finally, you will train and test the model by using a stochastic gradient descent (SGD) method. To do that, you have to fill your code in the blank section following the "`PLACE YOUR CODE HERE`" comments in the `MLP_assignment.ipynb` file. Note that you have to write down all the necessary equations and their derivation processes in your report.

a. A forward pass: compute a SoftMax loss [10pts]

In this part, you implement a fully-connected neural network with a ReLU activation function in the `forward_pass()` function. The mathematical notations for the first layer's linear transformation followed by a ReLU activation are as follows:

$$\mathbf{y} = \mathbf{w}\mathbf{x} + \mathbf{b} \tag{1}$$

$$\mathbf{h} = \max(0, \mathbf{y}), \tag{2}$$

where $\mathbf{x}$, $\mathbf{h}$, $\mathbf{w}$, and $\mathbf{b}$ are input tensor, output tensor, weight matrix, and bias matrix, respectively.

Then, you implement compute the forward pass to predict the output (i.e., class labels) and the loss by filling out the `softmax_loss()` function. You will compute each SoftMax loss L_i (details in the class note) materials and sum it for the total loss computation:

$$loss = \frac{1}{n} \sum_i L_i, \tag{3}$$

where n is the number of samples in input.

In your report, explain the forward process.

b. A backward pass: compute gradients [15pts]

In this part, you compute the backward pass by implementing the partial derivatives of the loss with respect to each parameter such as weights and biases. Your code must be in the `backward_pass()` function in which you can store the results to the `grads` dictionary. For example, the gradient on `w1` should be stored in `grads['w1']`. For more information about calculating the derivatives, please refer to here.

In your report, explain the backward process.

c. Training: Stochastic Gradient Descent (SGD) [15pts]

Now, train this neural network using an optimization method, stochastic gradient descent (SGD). Please fill out the `train()` function. To perform SGD via updating a parameter for each training example $x^{(i)}$ and label $y^{(i)}$. You should update network parameters according to the update rule from SGD with weight decay. You need to regularize the weights using regularization term $\frac{\lambda}{2}\theta^2$. You can use the following formula:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}; x^{(i)}, y^{(i)}) - \eta \lambda \boldsymbol{\theta}, \tag{4}$$

where $\boldsymbol{\theta}$ denotes the parameters of the model (e.g., a weight $\boldsymbol{w}$), η is the learning rate, λ is the regularization parameter, and $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}; x^{(i)}; y^{(i)})$ is the gradient of the objective function regarding the parameters.

In your report, 1) explain the training process, 2) print out training loss at every 100 epochs over 1000 iterations, 3) attach the loss plot, and 4) analyze your result.

d. Prediction [15pts]

Lastly, complete the prediction code in the `predict()` function so that you can compute the validation accuracy. To do that, the function has to select the best class label per input using the trained weights in `self.params`.

In your report, 1) explain the prediction process, 2) print out validation accuracy at every 100 epochs over 1000 iterations, 3) attach training and validation accuracy plots, and 4) analyze your result.

e. Visualization [10pts]

Here, visualize the weights learned in the network's first layer. In most neural networks trained on visual data, the first layer weights typically show

visible structure when visualized. Please fill out the `show_net_weights()` function.

In your report, 1) attach the visualization result and 2) analyze it.

f. Advanced - Activation functions [15pts]

Implement two other activation functions: Leaky ReLU and SWISH,

- $\text{LeakyReLU}(\mathbf{x}) = \max(0.01\mathbf{x}, \mathbf{x})$
- $\text{SWISH}(\mathbf{x}) = \mathbf{x} \cdot \textit{sigmoid}(\mathbf{x})$ where $\textit{sigmoid}(\mathbf{x}) = 1/(1 + \exp^{-\mathbf{x}})$.

In your report, 1) describe the backward passes, 2) print out training loss at every 100 epochs over 1000 iterations, 3) attach the loss and accuracy plots, and 4) analyze/compare the classification performance.

Problem 2: Hyperparameter Tuning [20pts]

You will tune the hyperparameters by developing an intuition for how they affect the final performance. By doing so, we want you to get a lot of practice. You must experiment with different values of the various hyperparameters, specifically learning rate, the dimension of hidden layer, and regularization strength (option). Note that your classification accuracy should be over 47% on validation data. In addition, the tuned model should give over 52% of classification accuracy.

In your report, 1) print out the results, 2) write down the best parameters and corresponding validation accuracy, and 3) analyzation with followings:

- plot of validation accuracy (e.g., x-axis is the number epochs, y-axis is the validation accuracy).
- plot of validation loss at each 100 epochs (e.g., 200, 300) computed over 1000 iterations.
- visualization of the weights that were learned in the first layer of the model.

Submission Guide

a. Submission Requirement

Change your file name to `cs470_yourname_studentID_idx.ipynb` in Colab. Download and save in your machine. Write a report explaining how

you implemented, comparing the models and discussing the test performance in PDF file. Generate a zip file of your code and report, then save your zip file as `cs470_yourname_studentID.zip`. Please submit the `.zip` file via KLMS.

Please make sure that the submitted notebooks have been saved and the cell outputs are visible.

b. Academic Integrity Policy

This is homework for each student to do individually. Discussions with other students are encouraged, but you should write your own code and answers. Collaboration on code development is prohibited. There will be given no points in the following cases:

- Plagiarism detection
- Peer cheating
- The incompleteness of the code
- The code does not work