Introduction to Artificial Intelligence (CS470): Assignment 1

Deadline:

Wednesday 29th March, 2023

Setup

Google Colaboratory

Please download a starter code containing Colab notebooks code address.

1. Unzip the starter code `.zip` file. You should see a `CS470_IAI_2023_Spring` folder. Create a folder in your personal Google Drive and upload `CS470_IAI_2023_Spring` 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.

1 Multi-Layer Perceptron (MLP) [100pts]

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 in your report.

1.1 A forward pass: compute a SoftMax loss [20pts]

In this part, you implement a fully-connected neural network with a ReLU activation function in the `forward_pass()` process. Let $\mathbf{x} \in \mathbb{R}^{n \times d}$ be the input matrix with n batch size and d input size. The network with l_1 hidden size (units) and l_2 output size (classes) layers outputs a matrix $\mathbf{y^{(2)}} \in \mathbb{R}^{n \times l_2}$, where the hidden layer uses weights $\mathbf{w}^{(1)}$ and biases $\mathbf{b}^{(1)}$. Likewise, for the output layer uses weights $\mathbf{w}^{(2)}$ and biases $\mathbf{b}^{(2)}$. In order to realize a complex approximator, we apply a nonlinear activation function, ReLU (Rectified Linear Unit): $\sigma(x) = \max(0, \mathbf{x})$. Note that the superscript denotes the index of the layer.

Then, you implement 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:

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

where n is the number of samples in input.

In your report,

- write down the equations for the forward pass from the input $\mathbf{x}$ to the output $\mathbf{y}^{(2)}$,
- write down the equations for the softmax loss,
- write down the dimension of variables used in the equations, and
- attach your solution code block to the report.

1.2 A backward pass: compute gradients [20pts]

You then 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 in the `grads` dictionary. For example, the gradient on `w1` should be stored in `grads['w1']`. For more information about calculating the derivatives, please refer here.

In your report,

- write down the equations for the backward pass from the loss $\mathcal{L}$ to the hidden layer weights $\mathbf{w}^{(1)}$ and biases $\mathbf{b}^{(1)}$,
- write down the dimension of variables used in the equations, and
- attach your solution code block to the report.

1.3 Training: Stochastic Gradient Descent (SGD) [20pts]

Now, train this neural network using an optimization method, stochastic gradient descent (SGD). (Note that SGD and mini-batch gradient descent are often used interchangeably.) Please fill out the `loss()` and `train()` functions. 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:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\nabla_{\boldsymbol{\theta}_t} J(\boldsymbol{\theta}_t; x_{(i)}, y_{(i)}) - \eta\lambda\boldsymbol{\theta}_t, \tag{2}$$

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, $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}; x_{(i)}; y_{(i)})$ is the gradient of the objective function regarding the parameters, and t represents the iteration index.

In your report,

- write down the equations for the softmax with loss in terms of the forward and backward passes,
- write down the equations for the regularization in terms of the forward and backward passes,
- print out training loss at every 100 iterations over 1000 iterations (no coding required),
- attach the loss plot,
- attach your solution code block to the report.

1.4 Prediction [10pts]

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,

- print out validation accuracy at every 100 iterations over 1000 iterations,
- attach training and validation accuracy plots,
- attach your solution code block to the report.

1.5 Visualization [10pts]

Here, visualize the weights learned in the network's first layer. In neural networks trained on visual data, the intermediate layer weights often learn to represent structures or patterns in a way that is visible. Please, fill out the `show_net_weights()` function.

In your report,

- attach the visualization result,
- analyze the result,
- attach your solution code block to the report.

1.6 Advanced - Activation functions [20pts]

Implement three other activation functions: Leaky ReLU, SWISH, and SELU,

- $\text{LeakyReLU}(\mathbf{x}) = \max(0.01\mathbf{x}, \mathbf{x})$,
- $\text{SWISH}(\mathbf{x}) = \mathbf{x} \cdot \mathit{sigmoid}(\mathbf{x})$ where $\mathit{sigmoid}(\mathbf{x}) = 1/(1 + \exp^{-\mathbf{x}})$,
- $\text{SELU}(\mathbf{x}) = \lambda \begin{cases} \alpha(e^x - 1) & \text{if } x < 0. \\ x & \text{otherwise.} \end{cases}$

In your report,

- describe the backward passes,
- print out training loss at every 100 iterations over 1000 iterations,
- attach the loss and accuracy plots,
- analyze/compare the classification performance (note that you can tune the parameters if you want),
- attach your solution code block to the report.

Submission Guide

1.7 Submission Requirement

You have to submit two types of materials: report and code.

- Report: Write a report answering all the questions in the assignments as a **PDF** file.
- Code: Change your code file name to `cs470_yourname_studentID_idx.ipynb` in Colab, where idx is the index of the problem. For example, after modifying the file `MLP_problem_1.ipynb`, save as `cs470_yourname_studentID_1.ipynb`, and for the `MLP_problem.ipynb`, save as `cs470_`

`yourname_studentID.ipynb`. Download and save on your machine. Please make sure that the submitted notebooks have been saved and the cell outputs are visible.

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.

1.8 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