Machine Learning and Deep Learning I Homework 3

Instructor: Joonseok Lee

Deadline: 2022/5/26 Thu, 23:59

- No unapproved extension of deadline is allowed. Late submission will result in 0 credit.
- Optimize your code as much as you can. We do not guarantee to run unreasonably inefficient codes for grading. Remember, vectorization is important for efficient computation!
- You will be given skeleton files for doing your assignment. Detailed instructions are given in the comments below each method to complete. Please read them carefully before jumping into implementation!
- Most of experiment/visualization sections are already given to you. Just plot the results, and use them to verify your implementation unless other instructions are provided.
- Each assignment is built and tested under Google Colaboratory. If you work on a local machine, you need to handle version issue on your own.
- Explicitly mention your collaborators or reference (*e.g.*, website) if any. If we detect a copied code without reference, it will be treated as a serious violation of student code of conduct.

1 k-Nearest Neighbor Classifier [20 pts]

Follow the instructions below to complete `hw3_knn.ipynb` provided on the ETL.

In this exercise, we are going to implement k-nearest neighbor classifier directly working with pixel values. For this assignment, do NOT use PyTorch, TensorFlow, or other neural network packages.

(You may suffer from the lack of memory space if you use the entire MNIST dataset for training and inference. To ease this problem, the provided code randomly selects 1000 training and 200 test examples by default. Please feel free to adjust these numbers depending on your computing environment.)

(a) Implement `compute_distance(self, X_test, dist_metric)`. [5 pts]
`compute_distance` should return the distance between each test example in `X_test` and the training data for `k` nearest neighbor classifier based on three metrics: Euclidean(L2) distance, dot-product similarity, and cosine similarity.

(b) Implement `predict_labels(self, X_test, dists, k)`. [5 pts]

(c) Experiment with multiple values of `k` and briefly report what you observe, including results based on different metrics. [10 pts]

2 Two-Layer Neural Network [30 (+10) pts]

Follow the instructions below to complete `hw3_2nn.ipynb` provided on the ETL.

In this question, we are going to implement a simple two-layer fully-connected neural network to classify the MNIST dataset. Specifically, the network we will build is given by

$$\hat{\mathbf{Y}} = \texttt{softmax}(\sigma(\mathbf{X} \cdot W_1 + b_1) \cdot W_2 + b_2),$$

where $\mathbf{X} \in \mathbb{R}^{N \times M}$ is the input matrix containing N images with M pixels, $\mathbf{Y} \in \mathbb{R}^{N \times C}$ is one-hot encoded ground truth with C classes.

For this assignment, do NOT use PyTorch, TensorFlow, or other neural network packages. This will be the only opportunity for you to learn what happens behind the scene of the forward pass and backpropagation.

(a) Implement `initialize_parameters(self, input_dim, num_hiddens, num_classes)`. [5 pts]

(b) Implement `forward(self, X)`. [10 pts]

(c) Implement `backward(self, X, Y, ff_dict)`. `ff_dict` is the output from the forward step. [10 pts]

(d) Set aside some training examples as validation set, and tune hyperparameters (*e.g.*, learning rate, hidden dimensions, number of epochs, batch size) to optimize the validation accuracy. Report the best combination of hyperparameters you found along with your final test accuracy. [5 pts]

(e) **(Bonus)** Change the activation function to ReLU, and repeat (d). Report the best combination of hyperparameters you found along with your final test accuracy. [10 pts]

3 Convolutional Neural Network [50 (+10) pts]

In this exercise, we are going to implement 2-D convolution function from scratch. We will then implement a customized PyTorch dataset to process and augment CIFAR-10 dataset. Armed with data augmentation, we are going to implement and train a ConvNet to classify CIFAR-10 images. You have to work on the skeleton code `hw3_cnn.py` provided on the ETL.

(a) Follow the instructions and complete the `Convolution(image, filter, stride=1, padding=0)` function. For this question, do NOT use PyTorch, TensorFlow, or other neural network packages. [20 pts]

(b) **(Bonus)** Complete your own data augmentation function `custom_augmentation(self, image)` without using PyTorch, TensorFlow, or other neural network packages. Based on the visualization results, briefly explain what you implemented, and justify the reason you think this is a good augmentation to solve image classification task. Be creative! [10 pts]

(c) Implement `transform(self, image)` method that converts the applies probabilistic data augmentation to the given images. [10 pts]

(d) Implement `collate_fn(self, data)` method. [10 pts]

(e) Define your own `ConvNet` model and perform architectural search. [10 pts]
You may try different combinations of training options (optimizers, hyperparameters) and model architectures. Upon successful training, the test accuracy should be above **70%**. You may use your own data augmentation function implemented in (b) as well.

What to Submit

Please upload a single zip file named with your student ID (e.g., `2021-20000.zip`) on eTL, containing

- Your complete `hw3_knn.ipynb`, `hw3_2nn.ipynb`, and `hw3_cnn.ipynb` files