ECE 449/ECE595 Machine Learning

Project 4: Fully Connected Neural Network

Two-Layer Neural Network

This project is to implement a fully connected 2-layer neural network for image classification. It lays a foundation for deep learning networks.

Data: CIFAR-10

CIFAR-10 consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

10 classes: airplane, automobile, bird, cate, deer, dog, frog, house, ship, truck.

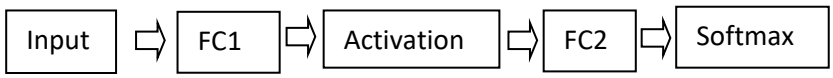
The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

You can select part of images for training and testing. The description of the dataset and the download link can be found at https://www.cs.toronto.edu/~kriz/cifar.html

Tasks

1. Read and display CIFAR-10 image files.
    a. Write utility functions to read CIFAR-10 image files. Reference the section of "Dataset layout" / "Python / Matlab versions" in CIFAR-10 link https://www.cs.toronto.edu/~kriz/cifar.html
    b. Read the dataset layout description. Organize the data in your own format if needed.
    c. Verify image and label reading
        - Randomly select 100 images and visualize them in 10x10 grids.
        - Display the class name or label on top of each image.
    d. Prepare images for training, validation and testing
    e. Prepare corresponding image labels (target classes) for loss calculation and performance evaluation.

2. Create 2-layer neural network structure:



- Input: images. Each image has the size of D = 32x32x3
- FC1 (Fully connect layer 1): The input of this layer is the image size D. The output size, H, is the number of neurons of the hidden layer. Shape of weights = D x H. (For example: H = 36 or 64, etc.)
- Activation: Nonlinear activation function. For example, ReLU or sigmoid.
- FC2 (Fully connected layer 2): Same as FC1. The output is 10 scores (the number of classes, C = 10). Shape of weights = H x C
- Scores: The output of FC2 is the score of each class
- Softmax: Probability of each class. The output of the network in inference, or the output of loss during training.

3. Define loss
    a. Option 1: Select an appropriate loss function implemented in the machine learning package you use. For example: Pytorch has implemented CrossEntropyLoss for classification by combining LogSoftmax and NLL(negative log-likelihood) loss functions.
    b. Option 2 (extra 20 points): Implement your own loss function. For example,

$$\text{Loss} = \text{Data Loss} + \text{L2 Regularization}$$

$$\text{Data Loss} = \text{Sum of (Target} - \text{Predicted)}^2$$

$$\text{L2 Regularization} = \text{Sum of (weights)}^2$$ (Note: biases are not included)

4. Choose optimizer: Use SGD implemented in pytorch, scikit learn, matlab or other platforms for automatic backpropagation and weight update.

5. Train the network
    a. Implement network training by combining the forward pass, loss calculation, backpropagation, and utility functions such as monitoring, visualization in one module
    b. Choose training parameters, such as the number of epochs, learning rate, etc.
    c. Monitor the loss
    d. Monitor the validation accuracy.

6. Evaluate the performance with your test dataset.

7. Submit your Python, Jupyter notebook or Matlab files to BrightSpace