CS324: Deep Learning

Assignment 1

From Jianguo Zhang

February 20, 2022

1 Part I: the perceptron (20 points)

In this first task you're asked to implement and test a simple artificial neuron: a perceptron (see **perceptron-slides.pdf**).

1.1 Task 1

Generate a dataset of points in $\mathbb{R}^2$. To do this, define two Gaussian distributions and sample 200 points from each. Your dataset should then contain a total of 400 points, 200 from each distribution. Keep 160 points per distribution as the training (320 in total), 40 for the test (80 in total).

1.2 Task 2

Implement the perceptron following the specs in **perceptron.py** and the pseudocode in **perceptronslides.pdf**.

1.3 Task 3

Train the perceptron on the training data (320 points) and test in on the remaining 80 test points. Compute the classification accuracy on the test set.

1.4 Task 4

Experiment with different sets of points (generated as described in Task 1). What happens during the training if the means of the two Gaussians are too close and/or if their variance is too high?

2 Part II: the mutli-layer perceptron (65 points)

In this second part of Assignment I you're asked to implement a multi-layer perceptron using numpy. Using scikit-learn and the `make_moons` method[1], create a dataset of $2,000$ two-dimensional points. Let S denote the dataset, i.e., the set of tuples $\{(x^{(0),s}, t^s)\}_{s=1}^{S}$, where $x^{(0),s}$ is the s-th element of the dataset and t^s is its label. Further let d_0 be the dimension of the input space and d_n the dimension of the output space. In this assignment we want the labels to be one-hot encoded[2]. The network you will build will have N layers (including the output layer). In particular, the structure will be as follows:

- Each layer $l = 1, \cdots, N$ first applies the affine mapping

$$\tilde{x}^{(l)} = W^{(l)} x^{(l-1)} + b^{(l)},$$

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html#sklearn.datasets.make_moons`

[2] Remember to transform the original dataset labels using one-hot encoding `https://en.wikipedia.org/wiki/One-hot`

where $W^{(l)} \in \mathbb{R}^{d_l \times d_{(l-1)}}$ is the matrix of the weight parameters and $b^{(l)} \in \mathbb{R}^{d_l}$ is the vector of biases. Given $\tilde{x}^{(l)}$, the activation of the l-th layer is computed using a ReLU unit

$$x^{(l)} = \max(0, \tilde{x}^{(l)}).$$

- The output layer (i.e., the N-th layer) first applies the affine mapping

  $$\tilde{x}^{(N)} = W^{(N)} x^{(N-1)} + b^{(N)},$$

  and then uses the softmax activation function (instead of the ReLU of the previous layers) to compute a valid probability mass function (pmf)

  $$x^{(N)} = \text{softmax}(\tilde{x}^{(N)}) = \frac{\exp(\tilde{x}^{(N)})}{\sum_{i=1}^{d_N} \exp(\tilde{x}^{(N)})_i}.$$

  Note that both max and exp are element-wise operations.

- Finally, compute the cross entropy loss L between the predicted and the actual label,

  $$L(x^{(N)}, t) = -\sum_i t_i \log x_i^{(N)}.$$

2.1 Task 1

Implement the MLP architecture by completing the files **mlp_numpy.py** and **modules.py**.

2.2 Task 2

Implement training and testing script in **train_mlp_numpy.py**. (Please keep 70% of the dataset for training and the remaining 30% for testing. Note that this is a random split of 70% and 30%)

2.3 Task 3

Using the default values of the parameters, report the results of your experiments using a jupyter notebook where you show the accuracy curves for both training and test data.

3 Part III: stochastic gradient descent (15 points)

In this third part of Assignment I, you will implement an alternative training method in **train_mlp_numpy.py** based on stochastic gradient descent.

3.1 Task 1

Modify the train method in **train_mlp_numpy.py** to accept a parameter that allows the user to specify if the training has to be performed using batch gradient descent (which you should have implemented in Part II) or stochastic gradient descent.

3.2 Task 2

Using the default values of the parameters, report the results of your experiments using a jupyter notebook where you show the accuracy curves for both training and test data.

4 Submission instructions

The submission will include:

- A written report describing what you did, the results and your analysis.
- Code for producing **all** results for all parts and tasks.
- Instructions on how to run the code.

Create a ZIP archive with the submission of Assignment 1 (all parts and tasks). Give the ZIP file the name **studentnumber_assignment1.zip**, where you insert your student number. Please submit the archive through the Blackboard.

Make sure all files needed to run your code are included or you may be given 0 points for it.

The deadline for assignment 1 (all parts and all tasks) is the 21st of March 2022 at 23:55 (Beijing Time).