Deep learning & applications

Practice#3

Tae Hyun Kim

Reference

- Python + Numpy tutorial
  - http://cs231n.github.io/python-numpy-tutorial

Task1: *Quick training for* binary classification using logistic regression (cross-entropy loss)

Input: 2-dim vector, $\boldsymbol{x} = \{x_1, x_2\}$
Output: label of the input, **y** $\in \{0,1\}$

Pseudo code #you can use numpy module!

Step 1. Generate 1000(=m) train samples, 100(=n) test samples:

```
x1_train=[], $x_2$_train=[], y_train=[]
for i in range(m):
    x1_train.append(random.randint(-2, 2))
    x2_train.append(random.randint(-2, 2))
    if x1_train[-1]*x1_train[-1] > x2_train[-1]:
        y_train.append(1)
    else:
        y_train.append(0)
x1_test=[], $x_2$_test=[], y_test=[] #generate ‘n’ test samples!
```

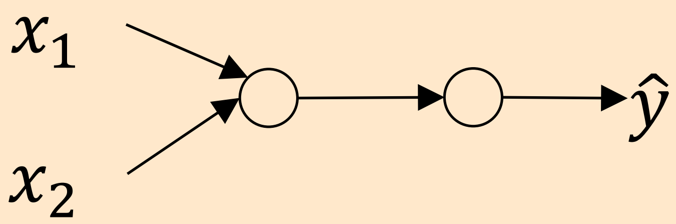
Step 2. Update $params$ with ‘m’ samples for (1000=K) iterations: #K grad updates!

Step 2-1. calculate the cost with m train samples!
Step 2-2. calculate the cost with n test samples!
Step 2-3. print accuracy with m train samples! (display the number of correctly predicted outputs/m*100)
Step 2-4. print accuracy with n test samples! (display the number of correctly predicted outputs/n*100)

Task2: *Quick training for* binary classification using 2-layered net (cross-entropy loss)

Input: 2-dim vector, $\boldsymbol{x} = \{x_1, x_2\}$
Output: label of the input, $\mathbf{y} \in \{0,1\}$

Pseudo code #you can use numpy module!



Step 1. Load generated ‘m’ train samples, ‘n’ test samples in task1

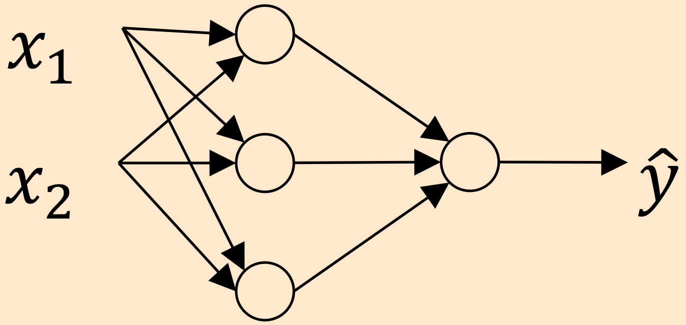
Step 2. Update $params$ with ‘m’ samples for (1000=K) iterations: #K grad updates!

Step 2-1. calculate the cost with m train samples!
Step 2-2. calculate the cost with n test samples!
Step 2-3. print accuracy with m train samples! (display the number of correctly predicted outputs/m*100)
Step 2-4. print accuracy with n test samples! (display the number of correctly predicted outputs/n*100)

Task3: *Quick training for* binary classification using 2-layered net (cross-entropy loss)

Input: 2-dim vector, $\boldsymbol{x} = \{x_1, x_2\}$
Output: label of the input, **y** $\in \{0,1\}$

Pseudo code #you can use numpy module!



Step 1. Load generated 'm' train samples, 'n' test samples in task1

Step 2. Update $params$ with 'm' samples for (1000=K) iterations: #K grad updates!

Step 2-1. calculate the cost with m train samples!
Step 2-2. calculate the cost with n test samples!
Step 2-3. print accuracy with m train samples! (display the number of correctly predicted outputs/m*100)
Step 2-4. print accuracy with n test samples! (display the number of correctly predicted outputs/n*100)

Report

- Submission due: (4/14, 3pm)
  - English only
  - Late submission will not be counted
- Submissions: (through blackboard system)
  - 3 source files: task1.py task2.py task3.py
  - Single page pdf: studentid_name.pdf
    - Include the table filled in

	Results in Task #1	Results in Task #2	Results in Task #3
Accuracy (with train set)			
Accuracy (with test set)			
Train time [sec]			
Inference (test) time [sec]			

- Explain what you learnt from this practice
  - Within 5 lines