Deep learning & applications

Practice#2

Tae Hyun Kim

Reference

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

Task1: 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

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.uniorm(-10, 10))
    x2_train.append(random.uniform(-10, 10))
    if x1_train[-1] + x2_train[-1] > 0:
        y_train.append(1)
    else:
        y_train.append(0)
x1_test=[], $x_2$_test=[], y_test=[] #generate 100 test samples!
```

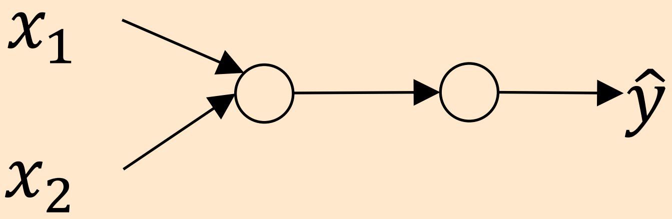
Step 2. Update W = $[w_1, w_2], b$ with 1000 samples for 2000 (=K) iterations: #K updates with the grad descent (Thr. = 0.5)

Step 2-1. print W, b every 10 iterations
Step 2-2. calculate the cost on the 'm' train samples!
Step 2-3. calculate the cost with the 'n' test samples!
Step 2-4. print accuracy for the 'm' train samples! (display the number of correctly predicted outputs/m*100)
Step 2-5. print accuracy with the 'n' test samples! (display the number of correctly predicted outputs/n*100)

Task2: 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

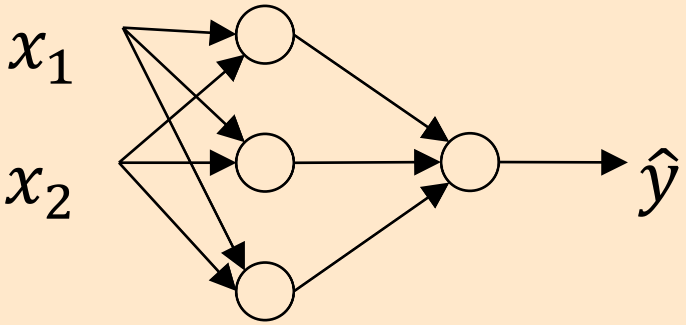
Step 2. Update $params$ with 'm' samples for 'K' iterations: #K grad updates!

Step 2-1. print W, b every 10 iterations
Step 2-2. calculate the cost on the 'm' train samples!
Step 2-3. calculate the cost with the 'n' test samples!
Step 2-4. print accuracy for the 'm' train samples! (display the number of correctly predicted outputs/m*100)
Step 2-5. print accuracy with the 'n' test samples! (display the number of correctly predicted outputs/n*100)

Task3: binary classification using wide 2-layered net (cross-entropy loss)

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

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 'K' iterations: #K grad updates!

Step 2-1. print W, b every 10 iterations
Step 2-2. calculate the cost on the 'm' train samples!
Step 2-3. calculate the cost with the 'n' test samples!
Step 2-4. print accuracy for the 'm' train samples! (display the number of correctly predicted outputs/m*100)
Step 2-5. print accuracy with the 'n' test samples! (display the number of correctly predicted outputs/n*100)

Report

- Submission due: (4/30, 1pm)
  - Late submission will not be counted

- Submissions: (through LMS system)
  - 3 source files: task1.py task2.py task3.py
  - Single page pdf: studentid_name.pdf
    - Should not be more than 3 pages
    - Should include
      - Accuracy (fill in the blanks in the tables below and add them to the report)
      - Discussion (what you learned in this experiment)

	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]			