HW2 TA hours

TAs
ntu.mlta@gmail.com

Outline

1. Logistic Regresion with batch training
2. Generative model
3. Shell Script

Logistic Regression

If we let batch_size=25…

```
X.shape=(25, 106)
w.shape=(106,)
b.sahpe=(1,)
z.shape=(25,)
y.shape=(25,)
Y.shape=(25,)
```

Parallel運算
X = 25個input feature
y = 25 prob
Y = 25個label

Logistic Regression

Logistic Regression

Step 1: $f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$

```
z = np.dot(X, np.transpose(w)) + b
y = sigmoid(z)
```

Step 2: $\boxed{\hat{y}^n\text{: 1 for class 1, 0 for class 2}}$

$$L(f) = \sum_n C(f(x^n), \hat{y}^n)$$

```
cross_entropy = -(np.dot(Y, np.log(y)) + np.dot((1 - Y), np.log(1 - y)))
```

$$C(f(x^n), \hat{y}^n) = -\left[\hat{y}^n ln f(x^n) + (1 - \hat{y}^n) ln\left(1 - f(x^n)\right)\right]$$

Logistic Regression

Step 3: Find the best function

$$\frac{-\ln L(w,b)}{\partial w_i} = \sum_n -\left[\hat{y}^n \overset{\left(1-f_{w,b}(x^n)\right)x_i^n}{\boxed{\frac{\ln f_{w,b}(x^n)}{\partial w_i}}} + (1-\hat{y}^n)\overset{-f_{w,b}(x^n)x_i^n}{\boxed{\frac{\ln\left(1-f_{w,b}(x^n)\right)}{\partial w_i}}}\right]$$

$$= \sum_n -\left[\hat{y}^n \left(1-f_{w,b}(x^n)\right)x_i^n - (1-\hat{y}^n) f_{w,b}(x^n) x_i^n\right]$$

mean

```
w_grad = np.sum(-1 * X * (Y - y).reshape((batch_size,1)), axis=0)
```

$$= \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right)x_i^n$$

Larger difference, larger update

```
w = w - l_rate * w_grad
```

$$w_i \leftarrow w_i - \eta \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right)x_i^n$$

Logistic Regression

$$z = w \cdot x + b = \sum_i w_i x_i + b$$

Step 3: Find the best function

$$\left(1 - f_{w,b}(x^n)\right) x_i^n \qquad\qquad -f_{w,b}(x^n) x_i^n$$

$$\frac{-\ln L(w,b)}{\partial \cancel{w_i}\ bi} = \sum_n -\left[\hat{y}^n \boxed{\frac{\ln f_{w,b}(x^n)}{\partial \cancel{w_i}\ bi}} + (1-\hat{y}^n) \boxed{\frac{\ln\left(1 - f_{w,b}(x^n)\right)}{\partial \cancel{w_i}\ bi}}\right]$$

$$= \sum_n -\left[\hat{y}^n \left(1 - f_{w,b}(x^n)\right) \cancel{x_i^n} - (1-\hat{y}^n) f_{w,b}(x^n) \cancel{x_i^n}\right]$$

mean

```
b_grad = np.sum(-1 * (Y - y))
```

$$= \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right) \cancel{x_i^n}$$

Larger difference, larger update

```
b = b - l_rate * b_grad
```

Gaussian Distribution

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

$$= \frac{1}{1 + \frac{P(x|C_2)P(C_2)}{P(x|C_1)P(C_1)}} = \frac{1}{1 + exp(-z)} = \sigma(z)$$

Sigmoid function

Gaussian Distribution-Training(mean)

$$\mu^* = \underbrace{\frac{1}{79}\sum_{n=1}^{79} x^n}_{\text{average}}$$

#C1

```python
#calclulate mu1 and mu2
mu1 = np.zeros((dim,))
mu2 = np.zeros((dim,))
for i in range(train_data_size):
    if Y_train[i] == 1:
        mu1 += X_train[i]
        cnt1 += 1
    else:
        mu2 += X_train[i]
        cnt2 += 1
mu1 /= cnt1
mu2 /= cnt2
```

Gaussian Distribution-Training(sigma)

#C1

$$\Sigma^* = \frac{1}{79}\sum_{n=1}^{79}(x^n - \mu^*)(x^n - \mu^*)^T$$

```
#calculate sigma1 and sigma2
sigma1 = np.zeros((dim,dim))
sigma2 = np.zeros((dim,dim))
for i in range(train_data_size):
    if Y_train[i] == 1:
        sigma1 += np.dot(np.transpose([X_train[i] - mu1]), [(X_train[i] - mu1)])
    else:
        sigma2 += np.dot(np.transpose([X_train[i] - mu2]), [(X_train[i] - mu2)])
sigma1 /= cnt1
sigma2 /= cnt2
```

Gaussian Distribution-Training(sigma)

#C1 **#C2**

Shared sigma: $$\Sigma = \frac{79}{140}\Sigma^1 + \frac{61}{140}\Sigma^2$$

#TrainData

```
shared_sigma = (float(cnt1) / train_data_size) * sigma1 \
             + (float(cnt2) / train_data_size) * sigma2
```

Gaussian Distribution-predict

$$\Sigma_1 = \Sigma_2 = \Sigma$$

$$z = \underbrace{(\mu^1 - \mu^2)^T \Sigma^{-1}}_{\boldsymbol{w}^T} x \underbrace{- \frac{1}{2}(\mu^1)^T \Sigma^{-1} \mu^1 + \frac{1}{2}(\mu^2)^T \Sigma^{-1} \mu^2 + ln \frac{N_1}{N_2}}_{\mathrm{b}}$$

$$P(C_1|x) = \sigma(w \cdot x + b)$$

How about directly find ***w*** and b?

```
def predict(X_test, mu1, mu2, shared_sigma, N1, N2):
    sigma_inverse = np.linalg.inv(shared_sigma)
    w = np.dot( (mu1-mu2), sigma_inverse)
    x = X_test.T
    b = (-0.5) * np.dot(np.dot([mu1], sigma_inverse), mu1) \
        + (0.5) * np.dot(np.dot([mu2], sigma_inverse), mu2) + np.log(float(N1)/N2)
    a = np.dot(w, x) + b
    y = sigmoid(a)
    return y
```

Sigmoid

```python
def sigmoid(z):
    res = 1 / (1.0 + np.exp(-z))
    return np.clip(res, 0.00000000000001, 0.99999999999999)
```

使用np.clip() 避免數值太小或太大而overflow

Generative model -Naive Bayes Classifier

- 假設每個attribute都是獨立的，全部有T個attribute
- P(C1|X) = P(C1|X1) * P(C1|X2) *...* P(C1|XT)
- continuous attribute: 當成gaussian，一樣算mean跟var
- discrete: P(C1|X1) = N(C1, X1) / N(X1)
- 算出全部attribute的P(C1|X)，並相乘
- 比較P(C1|X)與P(C2|X)

Shell Scripts

What is Shell Script?

Shell script defined as:
"*Shell Script is* ***series of command*** *written* ***in plain text file***.

... 把很多指令寫成一個檔案，可以一次做很多事情。

甚至是做判斷式或是迴圈。

Script Tutorial and Example

Shell Script Tutorial: http://linux.vbird.org/linux_basic/0340bashshell-scripts.php

Example:

```
# feature extraction by yourself
python my_feature_extraction.py $1 $2
python hw2_logistic_train.py
python hw2_logistic_test.py $5 $6
```

Passing Arguments

How to pass arguments to your srctipts or .py files?

1. 在terminal或cmd中的輸入與script中變數對應關係：
/path/example.sh /path/to/data /path/to/output
$0 $1 $2

2. 在terminal或cmd中的輸入與.py中變數對應關係：
/path/example.py /path/to/data /path/to/output
sys.argv[0] sys.argv[1] sys.argv[2]

Passing Arguments

script:

```
echo 'script_arg1 is : '$0''
echo 'script_arg2 is : '$1''
echo 'script_arg3 is : '$2''
echo 'Start to run Python'
python test.py $1 $2
```

python:

```
import sys
a = sys.argv[0]
b = sys.argv[1]
c = sys.argv[2]
print 'python_arg1 is : %s' % a
print 'python_arg2 is : %s' % b
print 'python_arg3 is : %s' % c
```

Passing Arguments

Let’s run…

```
acetylSv:~ acetylSv$ ./test.sh path1 path2
script_arg1 is : ./test.sh
script_arg2 is : path1
script_arg3 is : path2
Start to run Python
python_arg1 is : test.py
python_arg2 is : path1
python_arg3 is : path2
```

PATH

- 絕對路徑 : 相對於**根目錄**的路徑。
  - Ex: /Users/MLTA/Desktop/hw2
- 相對路徑 : 相對於**目前資料夾**的路徑。
  - Ex: ./hw2/logistic.py、../hw1/linear_regression.py

★ 助教會在git clone整個 ML2017/hw2資料夾

★ 並在 ML2017/hw2資料夾執行程式(Ex : hw2_best.sh)

★ 若要 **讀/存model**請用**相對路徑**

★ Data的Path助教會用絕對路徑下在hw2_best.sh的argument裡。

Model相對路徑

- 現在路徑: ~/ML/HW2
  - model路徑 : ./logisticparameter.model
  - Data路徑:
    - 如果寫死路徑
    - 寫死助教的Data就傳不進去
    - 有可能error

Announcement

- hw2_best.sh
  - training可以用gpu
  - 請同學確認程式是可以在cpu模式下跑