Machine Learning HW2
TA Hours

MLTAs
ntumlta2019@gmail.com

Announcement

- Strong baseline released !!
- Kaggle deadline postponed to 03/23/2019 11:59:59 (GMT+8)
- Github deadline postponed to 03/24/2019 23:59:59 (GMT+8)

Probabilistic generative model

Outline

- Load data & Normalization
- Posterior probability
- Predict

Load data & Normalization

```python
def read(self,name,path):
    with open(path,newline = '') as csvfile:
        rows = np.array(list(csv.reader(csvfile))[1:] ,dtype = float)
        if name == 'X_train':
            self.mean = np.mean(rows,axis = 0).reshape(1,-1)
            self.std = np.std(rows,axis = 0).reshape(1,-1)
            self.theta = np.ones((rows.shape[1] + 1,1),dtype = float)
            for i in range(rows.shape[0]):
                rows[i,:] = (rows[i,:] - self.mean) / self.std

        elif name == 'X_test':
            for i in range(rows.shape[0]):
                rows[i,:] = (rows[i,:] - self.mean) / self.std

        self.data[name] = rows
```

$$Z = \frac{X - \mu}{\sigma}$$

Posterior probability

If $P(C_1|x) > 0.5$ ➡ x belongs to class 1

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

$$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right\}$$

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

average

Posterior probability

$$\mu^* = \frac{1}{79}\sum_{n=1}^{79} x^n$$

average

```
class_0_id = []
class_1_id = []
for i in range(self.data['Y_train'].shape[0]):
    if self.data['Y_train'][i][0] == 0:
        class_0_id.append(i)
    else:
        class_1_id.append(i)

class_0 = self.data['X_train'][class_0_id]
class_1 = self.data['X_train'][class_1_id]
```

```
mean_0 = np.mean(class_0,axis = 0)
mean_1 = np.mean(class_1,axis = 0)
```

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

```
n = class_0.shape[1]
cov_0 = np.zeros((n,n))
cov_1 = np.zeros((n,n))

for i in range(class_0.shape[0]):
    cov_0 += np.dot(np.transpose([class_0[i] - mean_0]), [(class_0[i] - mean_0)]) / class_0.shape[0]

for i in range(class_1.shape[0]):
    cov_1 += np.dot(np.transpose([class_1[i] - mean_1]), [(class_1[i] - mean_1)]) / class_1.shape[0]

cov = (cov_0*class_0.shape[0] + cov_1*class_1.shape[0]) / (class_0.shape[0] + class_1.shape[0])
```

Posterior probability

$$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}$$

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

$$P(C_1|x) = \sigma(z)$$

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

```
self.w = np.transpose(((mean_0 - mean_1)).dot(inv(cov)) )
self.b =  (- 0.5)* (mean_0).dot(inv(cov)).dot(mean_0)\
    + 0.5 * (mean_1).dot(inv(cov)).dot(mean_1)\
    + np.log(float(class_0.shape[0]) / class_1.shape[0])
```

Predict

```python
def func(self,x):
    arr = np.empty([x.shape[0],1],dtype=float)
    for i in range(x.shape[0]):
        z = x[i,:].dot(self.w) + self.b
        z *= (-1)
        arr[i][0] = 1 / (1 + np.exp(z))
    return np.clip(arr, 1e-8, 1-(1e-8))
```

```python
def predict(self,x):
    ans = np.ones([x.shape[0],1],dtype=int)
    for i in range(x.shape[0]):
        if x[i] > 0.5:
            ans[i] = 0;
    return ans
```

If $P(C_1|x) > 0.5$ ➡ x belongs to class 1

$$P(C_1|x) = \sigma(z)$$

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

Logistic regression

Outline

- Load data
- Tips in ML training
- Logictic regression
- Gradient descent

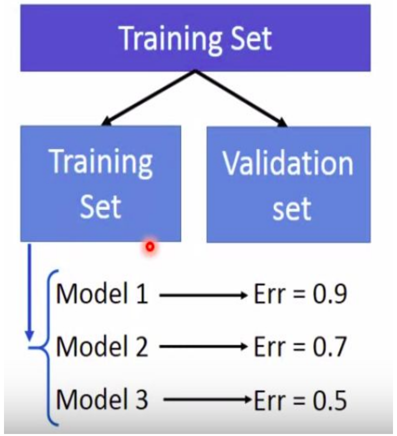
Load data

```
import sys
X_train_fpath = sys.argv[1]
Y_train_fpath = sys.argv[2]
X_test_fpath = sys.argv[3]
output_fpath = sys.argv[4]
```

```
In [3]: X_train = np.genfromtxt(X_train_fpath, delimiter=',', skip_header=1)
        Y_train = np.genfromtxt(Y_train_fpath, delimiter=',', skip_header=1)
```

Tips in ML training

- Validation set



- Normalization

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \qquad Z = \frac{X - \mu}{\sigma}$$

Tips in ML training

- One-hot encoding
- Discretization

Education
2
1
1
2
3

fnlwgt
226802
89814
336951
160323
103497

(0 = others, 1 = graduate school, 2 = university, 3 = high school)

others	graduate school	university	high school
0	0	1	0
0	1	0	0
0	1	0	0
0	0	1	0
0	0	0	1

0~100k	100k~200k	200k~300k	300k~400k
0	0	1	0
1	0	0	0
0	0	0	1
0	1	0	0
0	1	0	0

Logictic regression

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

Output: between 0 and 1

`_sigmoid` : to compute the sigmoid of the input. Use np.clip to avoid overflow. The smallest representable positive number is

```
>> np.finfo(np.float32).eps
>> 1.1920929e-07
```

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

Hence, we choose to clip at 1e-6 and 1-1e-6.

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

`get_prob` : given weight and bias, find out the model predict the probability to output 1

`infer` : if the probability > 0.5, then output 1, or else output 0.

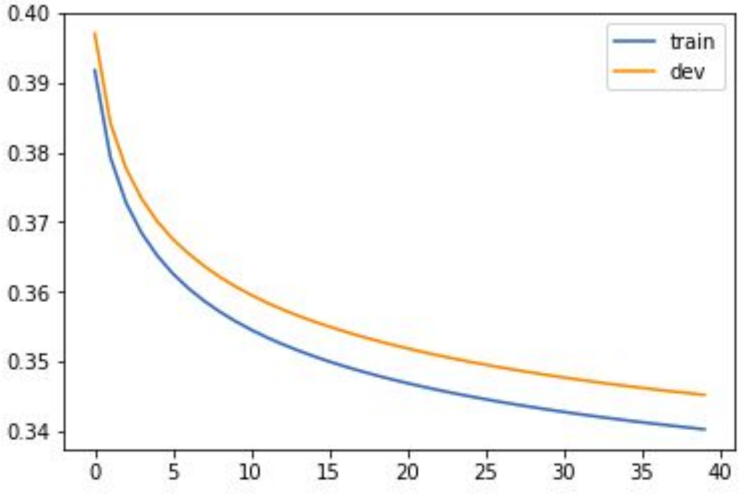
`_cross_entropy` : compute the cross-entropy between the model output and the true label.

`_compute_loss` : to compute the loss function $L(w)$ with input X, Y and w

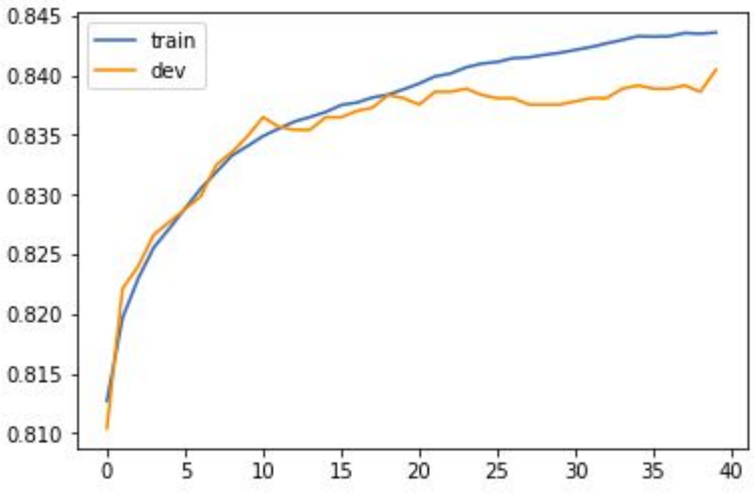
`_gradient` : With math derivation, the gradient of the cross entropy is $\sum_n -(\hat{y}^n - f_{w,b}(x^n))x_i^n$

Gradiend descent

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



Loss



Accuracy

Shell script example

report.pdf
hw2_best.sh
hw2_generative.py
hw2_generative.sh
hw2_logistic.py
hw2_logistic.sh

$1: raw data (train.csv) $2: test data (test.csv)

$3: provided train feature (X_train.csv) $4: provided train label (Y_train.csv)

$5: provided test feature (X_test.csv) $6: prediction.csv

```
x = np.genfromtxt(sys.argv[1], delimiter=',')
s = np.genfromtxt(sys.argv[2], delimiter=',')
s = np.genfromtxt(sys.argv[3], delimiter=',')
f = open(sys.argv[4],"w")
```

```
#!/bin/bash
python3 hw2_logistic.py $3 $4 $5 $6
```