通訊所 碩一 107064522 李曼鈴

Computer Assignment 1 : Adaboost algorithm in textbook

- 語言格式 : Python
- 程式流程
  1. 資料讀取
  2. Adaboost Algorithm (textbook version)
  3. Perceptron Learning
  4. Testing

- 程式說明與截圖輸出
  1. 首先取用第二次作業 knn 分類的 function。
  2. 讀取 training 及 testing data，將 Iris-setosa label 為'0'， Iris-versicolor label 為'1'。

```
## Data processing _ Training data
# Read Data
tr_data = pd.read_csv('training-data.txt', sep = ",", header = None)
columns = ["at1", "at2", "at3", "at4", "label"]
tr_data.columns = columns

# Label Iris-setosa as '0', Iris-versicolor as '1'
tr = tr_data.drop("label", axis = 1)
tr['label'] = 0
tr.loc[45:90, ['label']] = 1
tr_wol = tr.drop("label", axis = 1)

## Data processing _ Test data
# Read Data
te_data = pd.read_csv('testing-data.txt', sep = ",", header = None).iloc[0:10]
te_data.columns = columns

# Label Iris-setosa as '0', Iris-versicolor as '1'
te = te_data.drop("label", axis = 1)
te['label'] = 0
te.loc[5:10, ['label']] = 1
te_wol = te.drop("label", axis = 1)
```

  3. 接著對 training data 做 Adaboost
     (1) 首先設定一些初始值，一開始 90 組 example 被選取的機率分布 **p** 為 uniform；**T** 用來儲存九組 training subset。

```
## Adaboost
# Initialize examples distribution : Uniform
p = np.ones(len(tr_data))/len(tr_data)     # Initial distribution
T = list()     # 9 tr. subset with 10 examples/subset
```

     (2) 使用 **np.random.choice()** function 來選取 training subset

```
temp_idx = np.sort(np.random.choice(len(tr), 10, p = p))
```

(3) 將原先的 training set(90 組 examples)利用選出的 training subset(10 組 examples)做 1nn 分類，將正確分類數量的 **e** 記為 0；錯誤的則記為 1。

```
for i in range(len(tr)):
    output = knn(temp_T, 1, tr_wol.iloc[i])
    tr['output'][i] = output
    if (tr.iloc[i])['output'] == (tr.iloc[i])['label']:
        tr['e'][i] = 0  # True
    else:
        tr['e'][i] = 1  # False
```

(4) 透過 Adaboost 的規則去 update 機率分布 **p**，最後做 normalization 使 **p** 總和為 1(機率)。

```
epsilon = (tr['e']*p).sum()
beta = epsilon/(1-epsilon)

if beta != 0:
    for j in range(len(tr)):
        if tr['e'][j] == 0:
            p[j] = p[j]*beta    # Update examples distribution

    p = p/p.sum()   # Normalization
```

4. Perceptron

(1) 利用 perceptron learning 來 update 選出九組 classifier (training subset)的 weight (記為 **alpha**)，首先預設 weight 皆為 1。

(2) 將原先的 training set line-by-line 做 1nn，首先第一個 example 由九個 classifier 分類出九個預測結果，再用預設的 equally weighted 去加權投票出 master classifier 的最終分類結果，記做 **H**。

```
print("##### epoch = ", epoch, " #####")
for i in range(len(tr)):
    h = np.full((len(T)), -1)   # For storing output of subclassifiers
    H = -1  # Master classifier
    for j in range(len(T)):
        h[j] = knn(T[j], 1, tr_wol.iloc[i])
    # Weighted majority voting
    print(h)
    if (alpha[h == 1]).sum() > (alpha[h == 0]).sum():
        H = 1
    else:
        H = 0
```

(3) 接著比較 master classifier 分類結果 **H** 及實際的 label，若不相同則記錄 **err** 並且利用 perceptron learning rule 去更新每個 classifier 的 weight，將更新過後的 weight 拿到下個 example(回到步驟(2))去計算，iteration 計算完 90 組 examples 紀錄為 1 個 epoch。

```
    # Weight updating
    if H != tr["label"].iloc[i]:
        alpha = alpha + eta*(tr["label"].iloc[i] - h)
        err += 1
        print("Example ",i+1, ": miss")
    else:
        print("Example ",i+1, ": hit")
```

(4) 計算完一個 epoch 之後檢查是否達成 termination criteria。若 training data 中 90 組 examples 在 epoch 中皆正確分類則完成 perceptron，得到更新完成的 weight。

```
if err == 0:
    hold = 0
    print("##### DONE! #####\n")
    print("Final Weight: ", alpha)
else:
    epoch += 1
```

5. Testing

利用九組 classifier 配合 perceptron 得出的 weight 來分類 testing data，正確分類的記為 **True**，錯誤分類的記為 **False**，最後計算 accuracy。

```
## Testing
true = 0
false = 0

for i in range(len(te)):
    h = np.full((len(T)), -1)   # For storing output of subclassifiers
    H = -1  # Master classifier
    for j in range(len(T)):
        h[j] = knn(T[j], 1, te_wol.iloc[i])
    print(h)
    # Weight majority voting
    if (alpha[h == 1]).sum() > (alpha[h == 0]).sum():
        H = 1
    else:
        H = 0

    # Caculating accuracy
    if H == te["label"].iloc[i]:
        true += 1
        print("Example ",i+1, ": hit")
    else:
        print("Example ",i+1, ": miss")
        false += 1

accuracy = true/len(te)
print("Accuracy = ", accuracy)
```

6. 實驗結果

(1) Adaboost 跑九組 training subset，因此會更新 8 次 distribution p，以下為最後更新完的 **p**:

```
[4.33204004e-05 1.60167997e-04 1.60167997e-04 7.62139685e-03
 1.60167997e-04 4.33204004e-05 7.62139685e-03 1.60167997e-04
 1.72863022e-03 1.60167997e-04 1.60167997e-04 2.34564721e-04
 1.72863022e-03 1.72863022e-03 1.72863022e-03 1.72863022e-03
 2.34564721e-04 1.72863022e-03 1.26400620e-04 1.22920829e-02
 9.88987817e-02 8.42338796e-04 4.33204004e-05 2.34564721e-04
 4.33204004e-05 3.51567661e-02 8.42338796e-04 2.34564721e-04
 1.72863022e-03 1.60167997e-04 4.33204004e-05 2.90978164e-05
 1.60167997e-04 8.42338796e-04 4.33204004e-05 4.33204004e-05
 8.42338796e-04 8.42338796e-04 1.46212959e-01 3.10479099e-02
 1.72863022e-03 8.42338796e-04 1.72863022e-03 4.33204004e-05
 4.70794500e-02 4.07789385e-02 1.05398937e-03 1.97647654e-05
 1.97647654e-05 1.97647654e-05 2.76506258e-02 5.69531017e-05
 1.97647654e-05 5.77515312e-02 1.97647654e-05 9.65223965e-02
 1.05398937e-03 5.69531017e-05 5.69531017e-05 1.35340795e-04
 1.97647654e-05 1.05398937e-03 6.73719603e-03 5.69531017e-05
 1.05398937e-03 1.05398937e-03 5.69531017e-05 5.69531017e-05
 1.97647654e-05 1.97647654e-05 1.97647654e-05 1.05398937e-03
 6.73719603e-03 5.69531017e-05 6.09574732e-02 6.73719603e-03
 6.73719603e-03 1.05398937e-03 1.97647654e-05 5.69531017e-05
 6.73719603e-03 5.69531017e-05 1.05398937e-03 6.73719603e-03
 6.73719603e-03 6.73719603e-03 6.73719603e-03 1.05398937e-03
 2.29371492e-01 6.73719603e-03]
```

(2) 接著做 perceptron learning，分類正確為 hit；錯誤則為 miss 並更新 weight。若 epoch 中皆 hit 則結束 learning 並回傳 weight 值。

```
[1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1]
Example  80 : hit
[1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0]
Example  81 : hit
[1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1]
Example  82 : hit
[1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1 0]
Example  83 : hit
[1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0]
Example  84 : hit
[1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0]
Example  85 : hit
[1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0]
Example  86 : hit
[1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0]
Example  87 : hit
[1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1 0]
Example  88 : hit
[1 1 0 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1]
Example  89 : hit
[1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0]
Example  90 : hit
##### DONE! #####

Final Weight:  [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

(3) 最後預測 testing data 之 accuracy 為 0.9。

```
[0 0 0 0 0 0 0 0 0 0]
Example  1 : hit
[0 0 1 0 0 1 0 0 0]
Example  2 : hit
[0 0 0 0 0 0 0 0 0 0]
Example  3 : hit
[0 0 0 0 0 0 0 0 0 1]
Example  4 : hit
[0 0 1 1 0 1 0 0 1]
Example  5 : hit
[1 1 1 1 1 1 1 1 0]
Example  6 : hit
[1 0 0 0 0 1 1 1 0]
Example  7 : miss
[1 1 1 1 1 1 1 1 0]
Example  8 : hit
[1 1 1 1 1 1 1 1 0]
Example  9 : hit
[1 1 1 1 1 1 1 1 1]
Example  10 : hit
Accuracy =  0.9
```

Computer Assignment 2 : Original Adaboost algorithm

- 語言格式 : Python
- 程式流程
  1. 資料讀取
  2. Adaboost Algorithm (inventor version)
  3. Perceptron Learning
  4. Testing

- 程式說明與截圖輸出
  1. Assignment 2 與 Assignment 1 的差別在於機率分布 **p** 的更新方式，因此稍微修改 p 的 update criterion，最後一樣對 **p** 做 normalization 使機率總和為 1。

```
# Update the distribution
epsilon = (tr['e']*p).sum()
beta = epsilon/(1-epsilon)

if beta != 0:
    for j in range(len(tr)):
        if tr['e'][j] == 0: # True
            p[j] = p[j]*sqrt(beta)    # Update examples distribution
        else:
            p[j] = p[j]/sqrt(beta)    # Update examples distribution


    p = p/p.sum()   # Normalization
```

  2. 九個 classifier 的 weight 則用 epsilon 直接計算，而非透過 perceptron learning。

```
    # Voting weight
    if epsilon != 0:
        alpha[subset] = np.log((1-epsilon)/epsilon)/2
    else:
        alpha[subset] = alpha[subset-1]

print("##### DONE! #####\n")
print("Final Weight: ", alpha)
```

  3. 最後一樣計算 weight voting 套用在 testing data 的 accuracy，這部分與 Assignment 1 相同。

4. 實驗結果

(1) 利用 epsilon 直接計算之 weight:

```
Final Weight:  [1.68364791 0.86955787 0.64239499 1.19822418 0.78680786 1.22625665
 0.8710094  0.4170801  0.82874533]
```

(2) 最後預測 testing data 之 accuracy 為 0.9。

```
[0 0 0 0 0 1 0 0 0]
Example  1 : hit
[0 1 0 0 0 1 0 0 0]
Example  2 : hit
[0 0 0 0 0 1 0 0 0]
Example  3 : hit
[0 1 0 0 0 0 0 0 0]
Example  4 : hit
[0 1 0 1 0 1 1 1 0]
Example  5 : miss
[1 1 0 1 1 1 0 1 1]
Example  6 : hit
[1 0 0 1 0 1 0 1 1]
Example  7 : hit
[1 1 1 1 1 1 0 1 1]
Example  8 : hit
[1 1 0 1 1 1 0 1 1]
Example  9 : hit
[1 1 1 1 1 1 0 1 1]
Example  10 : hit
Accuracy =  0.9
```

(下一頁有 **Computer Assignment 3**)

Computer Assignment 3 : Perceptron Learning

- 語言格式 : Python
- 程式流程
  1. 資料讀取
  2. Perceptron Learning
  3. Testing

- 程式說明與截圖輸出
  1. Perceptron 部分
     (1) 以 while 迴圈執行 Perceptron Learning，當達成 termination criterion 時設定 hold=0 跳出迴圈。
     (2) 設定初始之 weight 及 learning rate 為 0.2。
     (3) 用 training data line-by-line 去 learn，將四個參數及 bias term (w_0)，加權相加之後若大於 0 則分類為 1，反之則分類為 0。
     (4) 計算 err 為真實 label 減去預測的 label，若兩者相同則 err 為 0。
     (5) 利用 perceptron learning 的 rule 去更新每個參數的 weight(下圖紅框處)。

```
err = np.full((len(tr)), 0)   # c(x) - h(x)
h = np.full((len(tr)), -1)   # Class return by classifier
err_temp = 0
print("##### epoch = ", epoch, " #####")
print(w)
for i in range(len(tr)):
    if (tr_wol.iloc[i]*w).sum() > 1e-10:
        h[i] = 1
    else:
        h[i] = 0
    err[i] = (tr.iloc[i])['label'] - h[i]
    w = w+eta*err[i]*tr_wol.iloc[i]
```

     (6) 分類完 90 個 example 之後即完成一個 epoch，計算 **err** 不為 0 的數量(表示有分類錯的 example)記為 **nerr**。

```
nerr = (err != 0).sum()
print("Number of errors: ", nerr)
```

     (7) 原先設定若 nerr 為零則表示 epoch 中 examples 皆分類正確，設定 hold=0 跳出迴圈，反之則繼續迴圈，但發現 perceptron 到最後一直有少數的 example 分類不正確而不易收斂。因此我更新 termination criteria 為「若連續十次分錯個數小於二」即社並 hold=0 停止 learning。(附圖見下頁)

```
if nerr != 0:
    epoch += 1
    if (nerr <= 2) & (abs(err_temp-nerr) <= 2):    # accuracy > 97%
        count += 1
    else:
        count = 0
    if count == 10:
        hold = 0
    err_temp = nerr
else:
    hold = 0
```

2. Testing

分類 testing data 的部分由前兩個小題稍做修改，原本以 knn()做分類，這邊以 perceptron learn 出來的 weight 來做分類的計算。

```
## Testing
true = 0
false = 0

for i in range(len(te)):
    if (te_wol.iloc[i]*w).sum() > 1e-10:
        h[i] = 1
    else:
        h[i] = 0
    err[i] = (tr.iloc[i])['label'] - h[i]

    # Caculating accuracy
    if h[i] == te["label"].iloc[i]:
        true += 1
        print("Example ",i+1, ": hit")
    else:
        print("Example ",i+1, ": miss")
        false += 1

accuracy = true/len(te)
print("\nAccuracy = ", accuracy)
```

5. 實驗結果

(1) 在做 perceptron learning 時一邊 print 出每次調整的 weight 以及發生 error 的個數，當我在上面步驟 1.(7)設定 training data 的 accuracy 至少要大於 97%時，大約會在 40 個 epoch 結束 learning(以下左圖顯示實際去跑的最後兩個 epoch，以及右圖跑完之後會顯示總 epoch 數還有最終 weight)。

```
##### epoch =  38  #####
at1   -0.04
at2    0.58
at3    1.68
at4    0.64
at0    0.40
dtype: float64
Number of errors:  2
##### epoch =  39  #####
at1   -0.10
at2    0.56
at3    1.78
at4    0.64
at0    0.40
dtype: float64
Number of errors:  2
```

```
##### DONE! #####

Number of epoch:  40
Final Weight:
 at1   -0.16
at2    0.54
at3    1.88
at4    0.64
at0    0.40
dtype: float64
```

(2) 套用在 testing data 時正確率只有 0.5，很明顯是將所有 example 直接判斷 label 為 1，但在 training data 上正確率有 97%的表現，我認為是由於在做 learning 時，classifier 為了去配合幾個比較 ambiguous 的 example，導致對 training data overfitting。

```
Example  1 : miss
Example  2 : miss
Example  3 : miss
Example  4 : miss
Example  5 : miss
Example  6 : hit
Example  7 : hit
Example  8 : hit
Example  9 : hit
Example  10 : hit

Accuracy =  0.5
```

更新

對程式稍做修改，將 while 迴圈的部分拿掉，讓 perceptron learning 只跑一個 epoch，結果顯示如下，accuracy 仍不理想，或許是因為一開始 weight 皆大於零，影響到加權之後恆正，判斷 label=1，而 training epoch 數又不夠導致這樣的結果。

```
##### epoch =  1  #####
[0.2 0.2 0.2 0.2 0.2]
Number of errors:  2
##### DONE! #####

Number of epoch:  1
Final Weight:
 at1    0.14
at2    0.18
at3    0.30
at4    0.20
at0    0.20
dtype: float64
Example  1 : miss
Example  2 : miss
Example  3 : miss
Example  4 : miss
Example  5 : miss
Example  6 : hit
Example  7 : hit
Example  8 : hit
Example  9 : hit
Example  10 : hit

Accuracy =  0.5
```

Comparison

將兩種 Adaboost 的方法跑了幾次之後，發現其對於 testing data 的 Accuracy 皆維持在 0.8 以上，而 perceptron 的 Accuracy 卻只有 0.5，可以發現像 Adaboost 的做法使用多個 classifier，classifier 雖然不是不出錯，但之間較能互相彌補，也能有較好的 performance。