Perceptron algorithm (with Python)

Tutorial 2 Yang

- The perceptron algorithm is an example of a linear discriminant model(two-class model)

How to implement the Perceptron algorithm with Python?

Tutorial 2

Through this tutorial, you will know:

- How to load training and testing data from files
- How to import the packages
- How to train the model by the training data
- How to make predictions with the testing data
- How to plot the figures illustrated the algorithm
- How to tune the parameters in the models

Library

Homegrown libraries and third-party application:

- For scientific computing: >>> import somelibrary

- Numpy: provide high-performance vector, matrix and higher-dimensional data structures for Python
- SciPy: based on the low-level Numpy framework and provides a large number of higher-level scientific algorithms
- matplotlib: an excellent 2D and 3D graphics library for generating scientific figures
- Pandas: a python package providing fast, flexible and expressive data structures for easy and intuitive data analysis and data manipulation
- **scikit-learn**: a open-source machine learning library, simple and efficient tools for data mining and data analysis

Perceptron Algorithm

Algorithm PerceptronTrain(linearly separable set R)

1. $\boldsymbol{w} \leftarrow \boldsymbol{w}^{(0)}; b \leftarrow b^{(0)}; MaxIter = 100$ #Initialize weight, bias and iteration number
2. $for\ t\ in\ range\ of\ 0\ to\ MaxIter\ do$
3. $\quad choose\ each\ (x, y) \in R$
4. $\quad a \leftarrow \boldsymbol{w}^{\boldsymbol{T}} * \boldsymbol{x} + b$ #compute activation function
5. $\quad if\ y! = \boldsymbol{sign}(a)\ then$
6. $\quad\quad \boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} + \eta * y * \boldsymbol{x}^{\boldsymbol{t}}$ #update the weights
7. $\quad\quad b^{(t+1)} \leftarrow b^{(t)} + \eta * y$ #update bias
8. $\quad else$
9. $\quad\quad \boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} \quad b^{(t+1)} \leftarrow b^{(t)}$
10. $\quad end\ if$
11. $end\ for$
12. $return\ \boldsymbol{w}, b$

Algorithm PerceptronPredict($\boldsymbol{w}, b, \hat{\boldsymbol{x}}$)

1. $a \leftarrow \boldsymbol{w}^{T} * \hat{\boldsymbol{x}} + b$ #compute activation for testing data
2. $return\ sign(a)$

Example

- Assessing credit card application

Age	23 years old
Annual salary	NTD 1,000,000
Year in job	0.5 year
Current debt	200,000

Result:
Approved, 1
Rejected, -1

Abstract the feature vector

Training data:

x_1	x_2	y
0.8	2.6	1
1.3	2	-1
2.2	1.4	-1
2.1	2.8	1
⋮	⋮	⋮

Testing data:

x_1	x_2	y
0.8	2.6	?
1.3	2	?
2.2	1.4	?
2.1	2.8	?
⋮	⋮	⋮

Example

As $w_0 = b,\ x_0 = 1$, initialization W=[1,1], b=1, and eta=0.1

$W' = [b, 1,1], \boldsymbol{x} = [1, x_1, x_2],$

1. $a = W^T * \boldsymbol{x} = 1 + 0.8 + 2.6 = 4.4 > 0,$
$f(a) = 1 \quad is\ the\ same\ with\ y = 1,$
return W and b

2. $a = W^T * \boldsymbol{x} = 1 + 2.2 + 1.4 = 4.6 > 0,$
$f(a) = 1 \quad is\ different\ with\ y = -1,$

update W and b:
$W = [1,1] + \text{eta} * y\ * [2.2,1.4] = [0.78,0.86]$
$b = b + \text{eta} * \text{y} = 1 + 0.1 * (-1) = 0.9$
……
Repeat until MaxIter times

Training data:

x_1	x_2	y
0.8	2.6	1
1.3	2	-1
2.2	1.4	-1
2.1	2.8	1
⋮	⋮	⋮

If eta=1, W=[1.2,0.4], b=0

Import the packages

```
#Import modules and packages
import os

import numpy as np

import pandas as pd

from sklearn.linear_model import Perceptron

import matplotlib.pyplot as plt
```

Load the data into pandas

```
#get the path of current directory
path=os.getcwd()
# load data
traindata=pd.read_csv(path+'¥¥traindata.csv') # Loading data from CSV into pandas
train_x=traindata.iloc[:,:-1]
train_y=traindata.iloc[:,-1]
testdata=pd.read_csv(path+r'¥testdata.csv')
test_x=testdata.iloc[:,:-1]        #Position based selection: - except last column
                                                              - Select only the last column
test_y=testdata.iloc[:,-1]
```

The perceptron function

```
# introduce the perceptron
MaxIter=20
per=Perceptron(max_iter=MaxIter, eta0=0.1,shuffle=True)
per.fit(train_x, train_y)
Test_y=pd.Series(per.predict(test_x), name='y')
testdata=test_x.join(Test_y, how='outer')

#write the predict results to file
testdata.to_csv(path+r'¥test.csv', index=False)
```

Parameters of Perceptron

class sklearn.linear_model. Perceptron (*penalty=None, alpha=0.0001, fit_intercept=True, max_iter=None, tol=None, shuffle=True, verbose=0, eta0=1.0, n_jobs=1, random_state=0, class_weight=None, warm_start=False, n_iter=None*)

[source]

Parameters:

Penalty: The penalty (aka regularization term) to be used. Defaults='None'

shuffle: Whether or not the training data should be shuffled after each epoch, default= 'True'

eta0: Constant by which the updates are multiplied, default=1

max_iter: The maximum number of passes over the training data. It only impacts the behavior in the fit method, and not the partial_fit. Default=5, or 1000(from v0.21)

n_iter: The number of passes over the training data. Default=None. Deprecated from v0.19 will be removed in v0.21)

Attributes:

coef_: array, shape = [1, n_features] if n_classes == 2 else [n_classes, n_features]; Weights assigned to the features.

intercept_ : array, shape = [1] if n_classes == 2 else [n_classes];Constants in decision function.

n_iter_ : int; The actual number of iterations to reach the stopping criterion. For multiclass fits, it is the maximum over every binary fit.

http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html

Plot the training and test data

```
#plot the train data set
label=train_y.copy()
label[label<0]=0                #set the label to (0,1)
label=label.astype(int)
label=label.values
colormap=np.array(['r','b'])
plt.scatter(train_x.iloc[:,0], train_x.iloc[:,1], marker='o', c=colormap[label])

#plot the test data set
labelt=Test_y.copy()
labelt[labelt<0]=0
labelt=labelt.astype(int)
labelt=labelt.values
plt.scatter(test_x.iloc[:,0], test_x.iloc[:,1], marker='+', c=colormap[labelt])
```

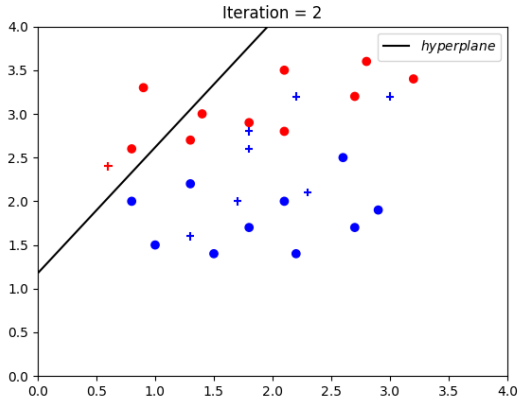
Plot the hyperplane

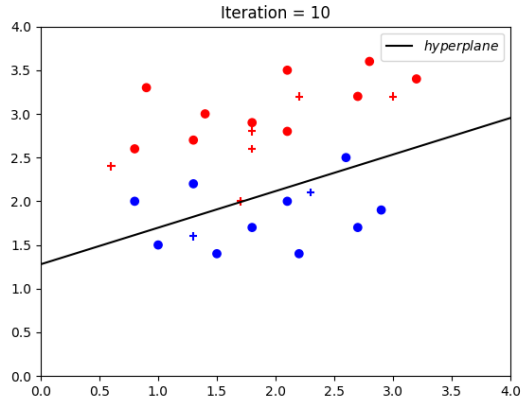
```
#calculate the hyperplane
w=per.coef_[0]
xx=np.linspace(0, 4)
yy=-(w[0]*xx+per.intercept_[0])/w[1]

#plot the line
plt.plot(xx, yy, 'k-', label='$hyperplane$')
plt.title(u'Iteration = %d' % MaxIter)
plt.legend()

plt.savefig(path+’¥¥perceptron.png’)
plt.show()
```

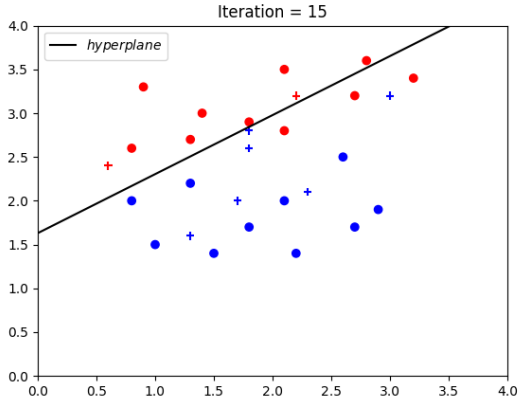
Accuracy rate

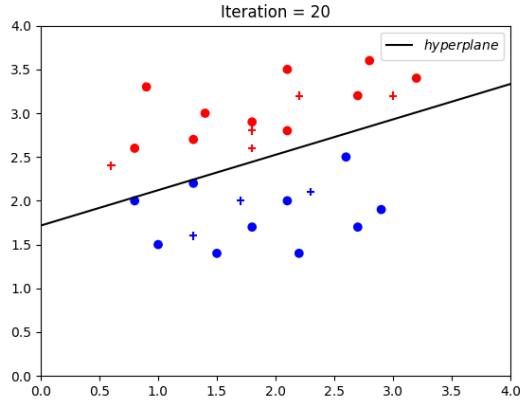
```
#calculate the accuracy rate for inseparable data sets
count=0
for i in range(len(Test_y)):
    if test_y.iloc[i]==Test_y.iloc[i]:
        count+=1.0

accuracy=count/float(len(Test_y))*100
print ('Accuracy rate: %.2f%%' % accuracy)
```

Example result









x_1	x_2	y
1.8	2.6	-1
0.6	2.4	-1
1.3	1.6	1
1.7	2.0	1
2.3	2.1	1
2.2	3.2	-1
1.8	2.8	-1
3.0	3.2	-1

Exercise 1:Simple Perceptron classifier and plot the results

- Copy the files of training data and testing data and Store in specified folder in your laptop
- Open a CMD window, change the directory path to the one stored the files -‘cd directory path’
- Run the jupyter notebook -’jupyter notebook’
- Copy the codes and paste in the jupyter file
- Plot the training data and testing data
- Plot the hyperplane

Exercise 2: Observe the behaviours of Perceptron for shuffle

- Create 8 subplots (2*4)
  - max iteration is set from 6 to 20 every 2 steps
  - Plot the training data
  - Plot the hyperplane
- Create 8 subplots (2*4)
  - shuffle is set to False
  - max iteration is set from 6 to 20 every 2 steps
  - Plot the training data
  - Plot the hyperplane

Exercise 3: Comparing the behaviours for eta0

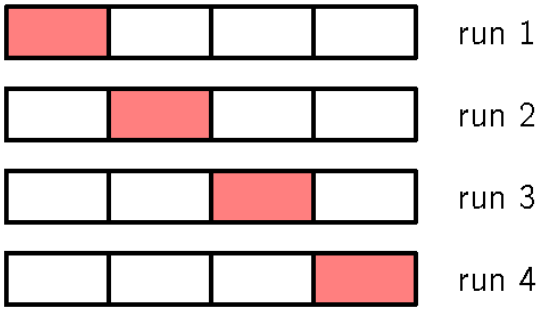
- Create 8 subplots (2*4)
  - eta0 is set to different value
  - max iteration is set from 6 to 20 every 2 steps
  - Plot the training data
  - Plot the hyperplane
  - print(w)

Exercise 4: Train the data by SGDClassifier

- Create 8 subplots (2*4)
  - Use the SGDClassifier function for classification
  - eta0 is set to 1
  - max iteration is set from 6 to 20 every 2 steps
  - Plot the training data
  - Plot the hyperplane
  - Comparing the behaviours for shuffle and eta0 to perceptron function

Exercise 5: How to select model by Accuracy rate

- Load the data from datafile.csv
- Partitioning it into S parts: $S-1$ is training data, remaining for testing
- Calculate the accuracy rate and repeat for all S possible choices
- Change the function from perceptron to SGDClassifier
- Tune the parameter for the model and observe



$S = 4$,
Repeat for 4 runs