HW 3 Perceptrons and SVMs

Kendra Chalkley

May 24, 2018

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

1. Perceptron

Below I've read in the data including a column of ones to augment the set and interface with the intercpet/threshold.

In [2]:

```
dataX= np.array([[1,1,1,1,1,1,1,1,1],
                 [-1,2,2,0,.5,3.5,3,5,5.5],
                 [-1,0,1,1,1.5,2.5,4,2,3]])
dataX=dataX.T

datay=np.array([1,1,1,1,1,-1,-1,-1,-1])
```

Perceptron algorithm is below:

In [18]:

```
def perceptron(X, y):
    n=X.shape[0]
    d=X.shape[1]
    classified=['F']*n

    i=0
    t=0
    iternum=16
    w=np.ones((2,d))
    weights=np.random.randn(d)
    w[0]=weights

    while (('F' in classified) & (t<iternum)):
        datum=X[i]
        yhat=np.dot(weights,datum)
        sign=y[i]*yhat

    ##Print Statements
#        print('t:',t)
#        print('i:',i)
#        print('datum:',datum)
#        print('weights:',weights)
#        print('class:',classified)
#        print('y:',y[i])
#        print('yhat:',yhat)
    ##Print Statements

        if sign >0:
            classified[i]='T'
        else:
            weights=weights-(datum*y[i])
#            print('weights after mod:',weights)
            classified[i]='F'
        i=(i+1)%n
        t+=1
    w[1]=weights

    return w
```

Running the algorithm

In [19]:

```
w=perceptron(dataX,datay)
```

Data manipulation:

In [23]:

```
dataX
pdata=pd.DataFrame({'X':dataX[:,1],
                   'Y':dataX[:,2],
                   'Class':datay})

class1=pdata.where(pdata['Class']==1)
class2=pdata.where(pdata['Class']==-1)
```

Plotting: class 1 in blue, class 2 in red, initial weights in purple, final weights in green.

In [24]:

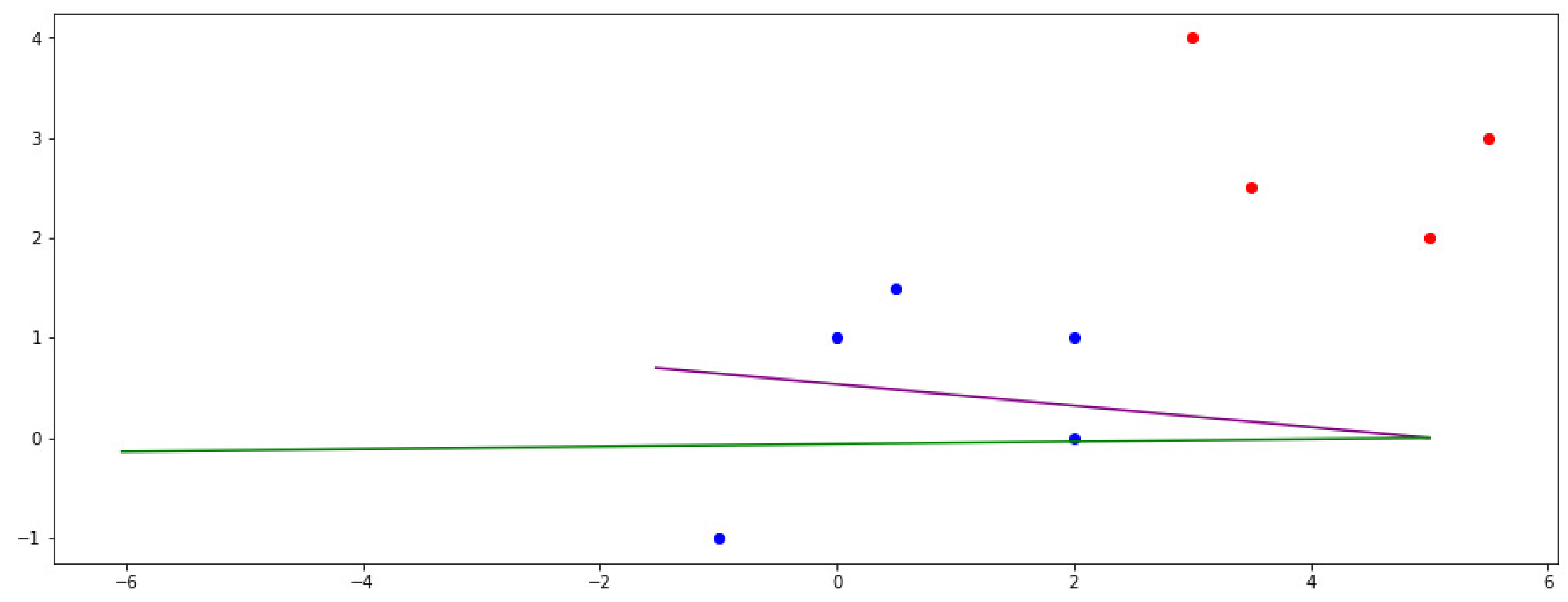
```
plt.figure(figsize=(16,6))
plt.plot(class1['X'],class1['Y'],'o', color='blue')
plt.plot(class2['X'],class2['Y'],'o', color='red')

weights=w[0]
plt.plot(
    [5, (-5*weights[1]/weights[2])],
    [0,(-weights[0]/weights[2])], color='purple'
)

weights=w[1]
plt.plot(
    [5, (-5*weights[1]/weights[2])],
    [0,(-weights[0]/weights[2])],color='green'
)
```

Out[24]:

```
[<matplotlib.lines.Line2D at 0x7f5a7c1740b8>]
```



Summary:

Initial weights:

In [25]:

```
w[0]
```

Out[25]:

```
array([-1.06033109,  0.46278717,  1.5149618 ])
```

Final weights:

In [26]:

```
w[1]
```

Out[26]:

```
array([ 2.93966891, 25.96278717, 21.5149618 ])
```

Plot and code above.

Questions: a) Is data linearly seperable? Yes. b) Does it work? Nope. Not entirely sure why not. But nope. c) It does not converge in <n iterations like it should, I put an arbitrary cap on iterations of 15 to prevent it from running forever.

2 SVM

Load training data:

In [7]:

```
pima_train=pd.read_csv('pima_train.txt',sep='  ', engine='python',header=None)

pima_train_x=pima_train.iloc[:,0:-1]
pima_train_y=pima_train.iloc[:,-1].replace(0,-1)
```

Load test data:

In [8]:

```
pima_test=pd.read_csv('pima_test.txt',sep='  ', engine='python',header=None)

pima_test_x=pima_test.iloc[:,0:-1]
pima_test_y=pima_test.iloc[:,-1].replace(0,-1)
```

Call solver to learn boundary:

In [9]:

```
from sklearn.svm import SVC

clf = SVC(C=.5,kernel='linear', max_iter=10000, random_state=3, tol=0.005)

clf.fit(pima_train_x, pima_train_y)
```

```
/home/kchalk/anaconda3/lib/python3.6/site-packages/sklearn/svm/base.py:218: ConvergenceWarning: Solver terminated early (
max_iter=10000).  Consider pre-processing your data with StandardScaler or MinMaxScaler.
  % self.max_iter, ConvergenceWarning)
```

Out[9]:

```
SVC(C=0.5, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=10000, probability=False, random_state=3, shrinking=True,
  tol=0.005, verbose=False)
```

Mean misclassification error for training set:

In [10]:

```
clf.score(pima_train_x, pima_train_y)
```

Out[10]:

```
0.686456400742115
```

Mean misclassification error for test set:

In [11]:

```
clf.score(pima_test_x, pima_test_y)
```

Out[11]:

```
0.6768558951965066
```

Shenanigans and confusion matrix:

In [12]:

```
combined=pd.DataFrame({
    'truth':pima_test_y,
    'predict':clf.predict(pima_test_x)
})

combined['n']=combined['truth']-combined['predict']

#combined=combined.replace(0,'y')
#combined=combined.replace(2,'n')
#combined=combined.replace(-2,'n')

combined.groupby(['truth','predict']).count()
```

Out[12]:

		n
truth	predict	
-1.0	-1.0	103
	1.0	58
1.0	-1.0	16
	1.0	52