Classification of Samples with Gaussian Densities

```
x0data = np.transpose(np.array([[np.random.normal(-1.5, 1.0) for i in
xrange(5000)], [np.random.normal(-1.5, 1.0) for i in xrange(5000)]]))
x1data = np.transpose(np.array([[np.random.normal(1.5, 1.0) for i in
xrange(5000)], [np.random.normal(1.5, 1.0) for i in xrange(5000)]]))

t0vec = -np.ones(5000)
t1vec = np.ones(5000)
```

np.random.normal(*mu*, *sigma*) generates Gaussian data with mean *mu* and standard deviation *sigma*.

The above code therefore generates 5000 Class-0 samples with Gaussian distribution centered at (-1.5, -1.5) and 5000 Class-1 samples with Gaussian distribution centered at (1.5, 1.5)

Shuffle Training Data Set and Train a Perceptron

```
>>> xdata = np.concatenate((x1data, x0data), axis=0)
>>> tvec = np.concatenate((t1vec, t0vec))
>>> shuffle_index = np.random.permutation(10000)
>>> xdata, tvec = xdata[shuffle_index], tvec[shuffle_index]
>>>
>>> c_all = Perceptron(tol=1e-3, random_state=0)
>>> c_all.fit(xdata, tvec)
Perceptron(alpha=0.0001, class_weight=None, early_stopping=False,
eta0=1.0,
      fit_intercept=True, max_iter=None, n_iter=None,
n_iter_no_change=5,
      n_jobs=None, penalty=None, random_state=0, shuffle=True,
tol=0.001,
      validation_fraction=0.1, verbose=0, warm_start=False)
```

Solution

```
>>> c_all.coef_
array([[ 2.19840892,  2.05510185]])
>>> c_all.intercept_
array([-1.])
```

We expect the solution to be at 45 degrees pointing up, so the coefficients of the solution are correct.
The ideal solution goes through the origin so that the intercept is larger than expected.
Whether the intercept value is acceptable can be seen by a plot of the sample points. Are the two sample populations well separated from each other?

Cross-Validation

```
>>> from sklearn.model_selection import cross_val_score
>>> cross_val_score(c_all, xdata, tvec, cv=4, scoring="accuracy")
array([ 0.978 ,  0.982 ,  0.9784,  0.9844])
```

“cv=4” specifies 4-fold cross-validation

The returned values are the accuracies for each run

```
>>> np.average(cross_val_score(c_all, xdata, tvec, cv=10,
scoring="accuracy"))
0.97829999999999995
```

Average of accuracies found from 10-fold cross-validation

Get the Predicted Values Instead of Accuracy

```
>>> from sklearn.model_selection import cross_val_predict
>>>
>>> predict10 = cross_val_predict(c_all, xdata, tvec, cv=10)
>>> predict10.shape
(10000,)
```

Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> confusion_matrix(tvec, predict10)
array([[4883,  117],
       [ 100, 4900]])
```

Each row is a true class: Row 0 is Class 0 and Row 1 is Class 1
Each column is a predicted class: Column 0 is how many predicted in Class 0; Column 1 is how many samples are predicted as Class 1

Precision and Recall

```
>>> from sklearn.metrics import precision_score, 
recall_score
>>> precision_score(tvec, predict10)
0.97667929041259716
>>> recall_score(tvec, predict10)
0.97999999999999998
```