Lab Three

COMP 219 - Advanced Artificial Intelligence

Xiaowei Huang
Cameron Hargreaves

October 8th 2018

1 Reading

Begin by reading chapter three of Python Machine Learning until page 68 (p75 2nd edition) found in the learning resources section of the vital page for COMP219. Code for this book is available online via the vital page or the book website, however it is recommended that you do not copy and paste this, try to type each line and understand what it is you are doing as you go along adding informative comments

2 Implement the Perceptron Learning Algorithm and the Logistic Regression classifier using sklearn

1. Implement the perceptron class from the book in your preferred IDE and run the code up to page 55 (p58 2nd edition), the full code for this program can be found at the end of this document

2.1 Tasks

1. Modify the code so that instead of inputting the first and third colummns of the iris dataset (sepal length and petal length), we input the sepal width (first column) and petal length. How many epochs does this take before we no longer update the weights of our perceptron

2. Update the code to instead classify "Iris-versicolor" from "Iris-virginica" on sepal and petal length. Will our classifier ever be able to correctly classify all instances of versicolor in this dataset?

3. Add the data for the iris species virginica onto the scatter plot from page 29 and plot these values together. Which species are not linearly separable based on sepal and petal width?

3 Further Questions

1. Below is the logic table for an AND gate, using a learning rate, η of 0.5, an initial weight vector of $\mathbf{w}$ = [0, 0] and an initial threshold, ϕ of 1 what is the final weight vector after running through each value in the below table for three epochs

X_1	X_2	Y
0	0	-1
0	1	-1
1	0	-1
1	1	1

W_0	W_1	W_2
1	0	0

2. Read up to page 42 and implement the AdaLine perceptron that is described in this chapter

3. The AdaLine perceptron reuses a lot of the same code as the binary perceptron. Reimplement AdaLine by inheriting the Perceptron class only overriding the `fit()` and `activation()` functions

4 Code for sklearn perceptron

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from matplotlib.colors import ListedColormap

def plot_decision_regions(X, y, classifier,
                test_idx=None, resolution=0.02):

    # setup marker generator and color map

    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                        np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot all samples
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
            alpha=0.8, c=cmap(idx),
            marker=markers[idx], label=cl)

    # highlight test samples
    if test_idx:
        X_test, y_test = X[test_idx, :], y[test_idx]
        plt.scatter(X_test[:, 0], X_test[:, 1],
            c='', edgecolor='black', alpha=1.0,
            linewidth=1, marker='o',
            s=100, label='test set')

iris = datasets.load_iris()
X = iris.data[:, [2, 3]]
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(
                                X, y, test_size=0.3, random_state=0)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

ppn = Perceptron(n_iter=40, eta0=0.1, random_state=0)
ppn.fit(X_train_std, y_train)

y_pred = ppn.predict(X_test_std)

print('Misclassified samples: {0}'.format((y_test != y_pred).sum()))
print('Accuracy: {0:.2f}'.format(accuracy_score(y_test, y_pred)))

X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X=X_combined_std,
                      y=y_combined,
                      classifier=ppn,
                      test_idx=range(105,150))

plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()
```