Shanghai Jiao Tong University ECE4530J

Mini Project 3: Due 2022.7.27

Problem 1

Suppose that the response y is generated by

$$y = f(x) + \epsilon,$$

where ϵ is a zero-mean Gaussian noise with variance 1.

a) Suppose that $f(x) = x$. Randomly generate 10 x's and generate the corresponding y's; you need to generate two random numbers (i.e., x and ϵ for each of the 10 points). Fit the data with linear regression and plot the scatter points.

b) Suppose that $f(x) = x^2$. Randomly generate 10 (x, y) pairs. Fit the data with linear regression and plot the scatter points.

c) Suppose that $f(x) = 1/x$. Randomly generate 10 (x, y) pairs. Fit the data with linear regression and plot the scatter points.

Problem 2

Consider the data in the attachment "MP3P2.xlsx".

(a) Show that linear regression does not work for this classification problem.

(b) Use the logit function to train a linear classifier. Indicate the boundaries that you find. Report the misclassification rate.

Tools and tutorials for linear regression:

Python:

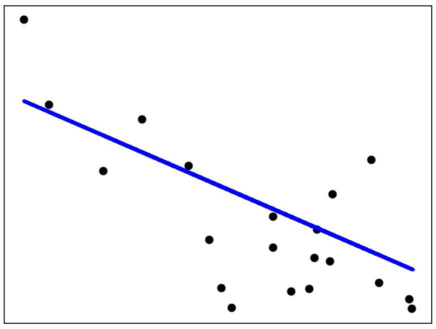
Scipy is a very powerful Python library to solve regressionproblem. Below is a vivid example about linear regression. For other details, please check the official guidance here:
sklearn.linear_model.LinearRegression — scikit-learn 1.1.1 documentation

Example:

The example below uses only the first feature of the diabetes dataset, in order to illustrate the data points within the two-dimensional plot. The straight line can be seen in the plot, showing how linear regression attempts to draw a straight line that will best minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation.

The coefficients, residual sum of squares and the coefficient of determination are also calculated.

```
Coefficients: [938.23786125]
Mean squared error: 2548.07
Coefficient of determination: 0.47
```



```
# Code source: Jaques Grobler
# License: BSD 3 clause

import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 2]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)

# The coefficients
print("Coefficients: \n", regr.coef_)
```

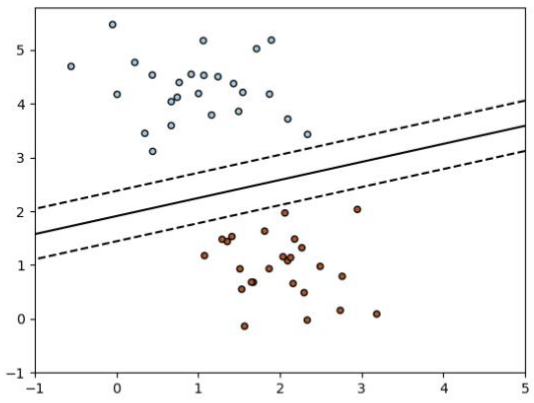
```
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test,
diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(diabetes_y_test,
diabetes_y_pred))

# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test, color="black")
plt.plot(diabetes_X_test, diabetes_y_pred, color="blue", linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```

Tools and tutorials for linear classification:

Python:

Below is a vivid example about linear classification. For other details, please check the official guidance here: sklearn.linear_model.SGDClassifier — scikit-learn 1.1.1 documentation

Example: SGD: Maximum margin separating hyperplane



```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.datasets import make_blobs

# we create 50 separable points
```

```
X, Y = make_blobs(n_samples=50, centers=2, random_state=0,
cluster_std=0.60)

# fit the model
clf = SGDClassifier(loss="hinge", alpha=0.01, max_iter=200)

clf.fit(X, Y)

# plot the line, the points, and the nearest vectors to the plane
xx = np.linspace(-1, 5, 10)
yy = np.linspace(-1, 5, 10)

X1, X2 = np.meshgrid(xx, yy)
Z = np.empty(X1.shape)
for (i, j), val in np.ndenumerate(X1):
    x1 = val
    x2 = X2[i, j]
    p = clf.decision_function([[x1, x2]])
    Z[i, j] = p[0]
levels = [-1.0, 0.0, 1.0]
linestyles = ["dashed", "solid", "dashed"]
colors = "k"
plt.contour(X1, X2, Z, levels, colors=colors, linestyles=linestyles)
plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired, edgecolor="black",
s=20)

plt.axis("tight")
plt.show()
```