Machine Learning, 2021 Spring
Homework 3

Due on 23:59 APR 7, 2021

1 Training Logistic Regression Models via Scikitlearn

1.1 Dependencies:

This homework requires a **python** implementation. Followings are the packages this work may depend on:

- Scikitlearn: `https://scikit-learn.org/stable/install.html`,
- Numpy: `https://numpy.org/install/`,
- Matplotlib: `https://matplotlib.org/stable/users/installing.html`.

We strongly recommend that you use Anaconda to install them.

1.2 Your tasks:

We provide three data sets from Scikitlearn:

1. $\mathcal{D}_1$: an almost linearly separable data set,
2. $\mathcal{D}_2$: a "moon"-like data set,
3. $\mathcal{D}_3$: a "circle"-like data set.

Please train logistic regression classifiers on these three data sets. Note that you **do not** need to write a logistic regression model by yourself, just use the one provided in Scikitlearn. We left the requirement details for each data set in the following. For your convenience, we also provide a jupyter notebook for the code (named 'hw3.ipynb'). You can find the data sets generation part in file *data.py* as well.

Task I:

After executing the data generation code for Task I, you can get a scatter plotting of the data points as shown in 1.

Problem 1: It is your turn to use the logistic regression model (module *linear_model.LogisticRegression* in Scikitlearn) for training a linear classifier on $\mathcal{D}_1$. **You need to:** [3pts]

(i) provide the specific values of parameters you choose, e.g., *tol* (tolerance), *max_iter*;

(ii) report the score of your classifier (the mean accuracy on the given test data and labels);

(iii) plot your decision boundary (actually a line in this 2-dimension case).
(**Hint:** you can refer to the example code provided in "Task II" for the details of plotting.)

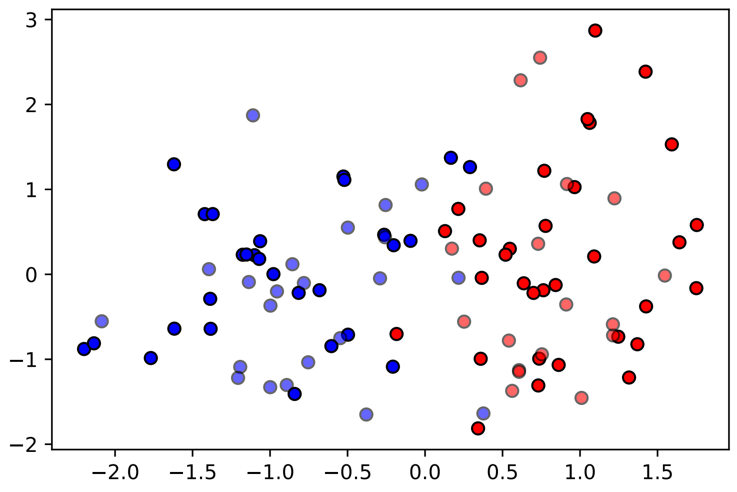

Figure 1: The linearly separable data set.

1.2.1 Task II:

After executing the data generation code for Task II, you can get the scatter plotting of the data points as shown in 2.

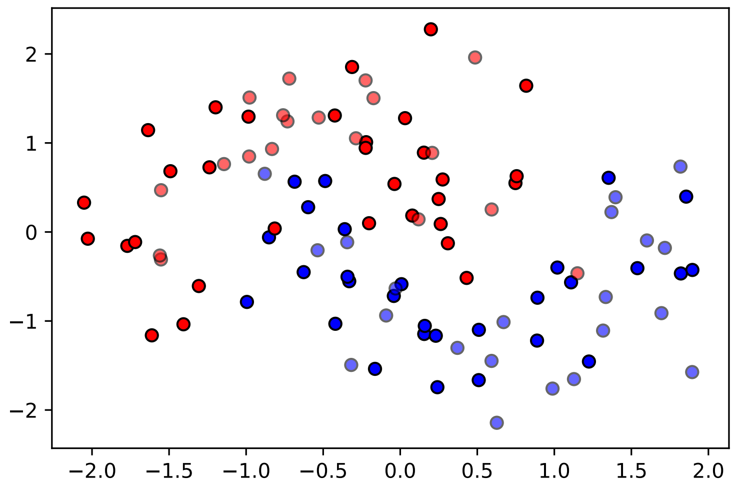

Figure 2: The “moon”-like data set.

Problem 2: It is your turn to use the logistic regression model (module *linear_model.LogisticRegression* in Scikitlearn) for training a linear classifier on $\mathcal{D}_2$. **You need to:** [3pts]

(i) provide the specific values of parameters you choose;

(ii) report the score of your classifier (the mean accuracy on the given test data and labels);

(iii) plot your decision boundary (actually a line in this 2-dimension case).

Let us think a little bit further of this task. While finishing the training of this task, you may not completely satisfy with the classifier you have got, since you can not fit a linearly non-separable data set well with a linear classifier. Unfortunately, the logistic regression model is essentially a linear model. However, we are not at a loss what to do.

Often it's useful to add complexity to the model by considering nonlinear features of the input data. A simple and common method to use is polynomial features, which can get features' high-order and interaction terms. For example, if we perform a polynomial transformation with degree$= 2$ on features (x_1, x_2), they will be transformed to $(1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$. Then, training a linear model on the transformed data set will lead to a polynomial model of the original data set. We will not go further for the theoretical analysis, instead providing the implementation and the associated results, which is called the kernel method.

```
'''Define polynomial regression, the value of degree can adjust
the characteristics of polynomial'''
poly_reg = PolynomialFeatures(degree=3)
# Feature processing (get the sample data corresponding to the
    corresponding feature of the polynomial)
X_poly = poly_reg.fit_transform(X)

# Training model
model = linear_model.LogisticRegression()
model.fit(X_poly, y)

'''The returned estimates for all classes are ordered by the
    label of classes.'''
z = model.predict_proba(poly_reg.fit_transform(np.c_[xx.ravel(),
    yy.ravel()]))[:, 1]
# '''Predict class labels for samples in X.'''
# z = model.predict(poly_reg.fit_transform(np.c_[xx.ravel(), yy.
    ravel()]))

z = z.reshape(xx.shape)
score = model.score(X_poly, y)

# Plot the contour of the classifier
plt.contourf(xx, yy, z, cmap=cm, alpha=.8)
# Plot the training points
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train,
cmap=cm_bright, edgecolors='k')
# Plot the testing points
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright,
edgecolors='k', alpha=0.6)

name = 'Logistic Regression'
plt.title(name)
plt.text(xx.max() - .3, yy.min() + .3, ('%.3f' % score),
size=15, horizontalalignment='right')
fig2p = plt.gcf()
plt.show()
```

Then we get the classifier shown in Figure 3. (We only focus on the classification performance and do not take the generalization of the model into account of this task.)

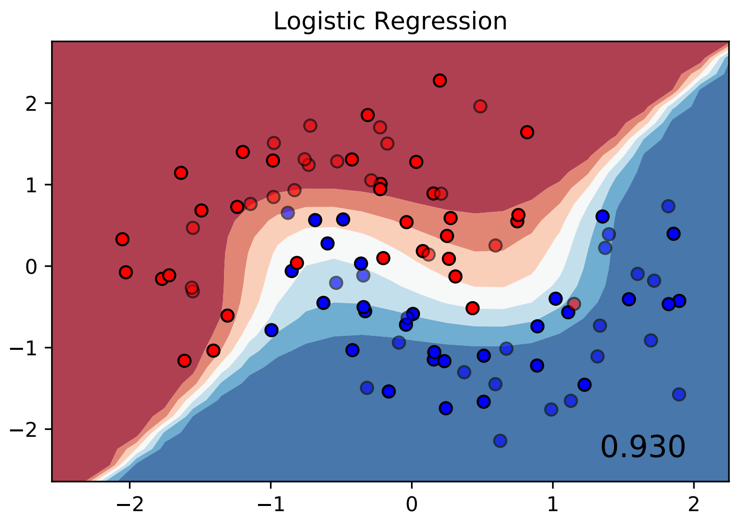

Figure 3: Polynomial transformed classifier.

1.2.2 Task III:

After executing the data generation code for Task III, you can get the scatter plotting of the data points as shown in 2.

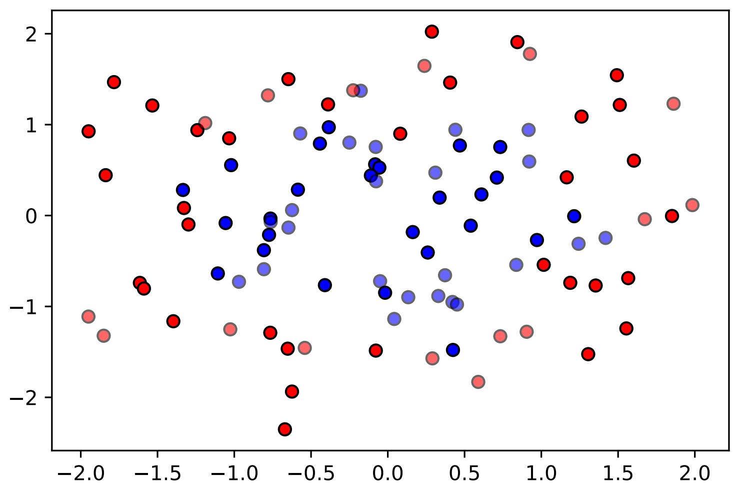

Figure 4: The "circle"-like data set.

Problem 3: It is your turn to use the logistic regression model (module *linear_model.LogisticRegression* in Scikitlearn) for training a linear classifier on $\mathcal{D}_3$. **You need to:** [2pts]

(i) provide the specific values of parameters you choose;

(ii) report the score of your classifier (the mean accuracy on the given test data and labels);

(iii) plot your decision boundary (actually a line in this 2-dimension case).

Extra task: perform a polynomial transformation on $\mathcal{D}_3$ and re-train a logistic regression classifier on the transformed data set. **You need to:** [2pts]

(i) provide the specific values of parameters you choose;

(ii) report the score of your classifier (the mean accuracy on the given test data and labels);

(iii) plot your decision boundary (actually a line in this 2-dimension case).