OASIS ML Group TRAINING 02

◆ Logistic Regression simple practice – Binary Classification

In this practice, you have to try to separate data to two varieties which are "Setosa" and "Versocolour". You are asked to use "**Iris Data Set**" as training datasets, which is also used frequently in Machine Learning issues. Please remember that your goal is try to generate a decision boundary to classify datasets. Follow hints below to finish it.

▫ Practice：

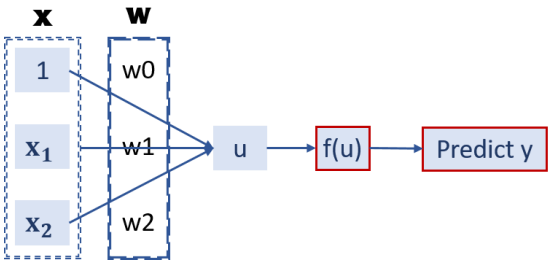
■ Analysis datasets：

Isis datasets contains three varieties of Iris：「Setosa」、「Versocolour」and 「Virginica」. Each variety has 50 data, which means there're total 150 data in the datasets. Each data contains 4 features：「Sepal length」、「Sepal width」、「Petal length」and「Petal width」.

■ $u = w_0 + w_1x_1 + w_2x_2$

■ Binary logistic regression prediction model：

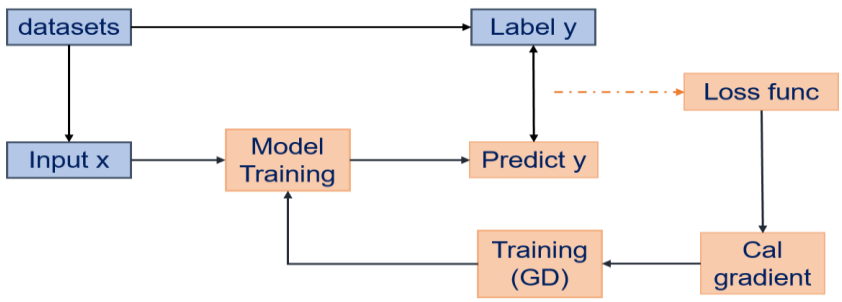
- Use Sigmoid function as f(u).
- Discuss why Sigmoid function?



■ Loss function：

- Using Cross entropy function as loss function.
- Discuss what is cross entropy function?
- $H = \frac{-1}{M}\sum_{m=0}^{M-1}(label\_y_m \ \log(pred\_y_m))$

■ Training flow structure：



■ Recommend initial parameter：

Epoch：1000,　learning rate：0.01

- **Hint 01**：Import library you needed, but do not call API.
- **Hint 02**：Get the datasets from sklearn.

```
from sklearn.datasets import load_iris
iris = load_iris()
```

- **Hint 03**：Cause you're asked to do binary classification and wanted to be easy to demonstrate in this practice, you **only need to choose 2 varieties and 2 features**.

(Don't be nervous, you will be asked to finish with whole datasets in next training practice)

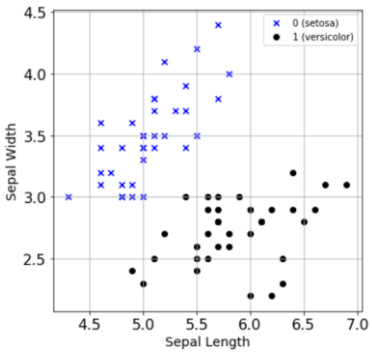
```
x_data, y_data = iris.data[:100, :2], iris.target[:100]
```

- **Hint 04**：Separate datasets into training dataset and testing datasets by using train_test_split from sklearn.

(Notice：You don't need to separate into "training", "testing" and "validation". Reason：It's a simple practice not a hige project, you only need to do testing after training to make sure your model is successful.)

```
from sklearn.model_selection import train_test_split
```

- **Hint 05**：Show the figure of datasets.



- **Hint 06**：Define function you need, for example：sigmoid, cross entropy……

```
def sigmoid(x):
    return ### ????? ###

def predict_(x, w):
    return ### ????? ###

def cross_entropy(yt, yp):
    ### Code here ###
    return ### ????? ###

def classify_result(y):
    return np.where(y<0.5, 0, 1)
```

- **Hint 07**：Calculating the accuracy score by using accuracy_score from sklearn.

```
from sklearn.metrics import accuracy_score
```

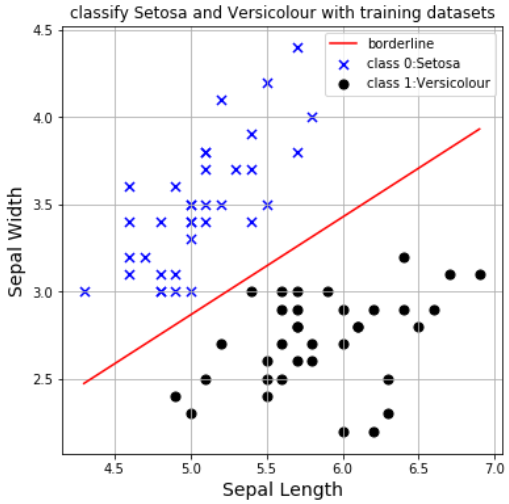
- **Hint 08**：After training, show the result.

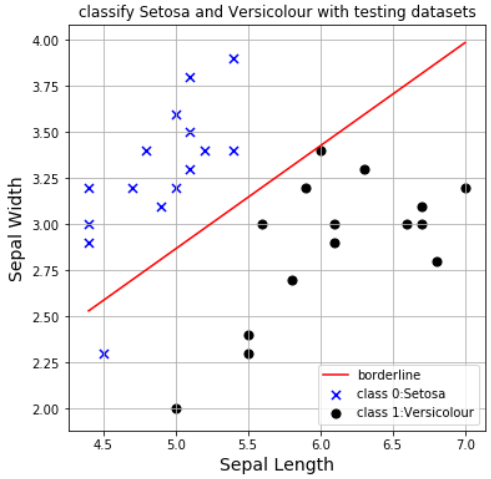
```
First epoch result: Loss:4.493842, Accur : 0.500000
Final epoch result: Loss:0.153947, Accur : 0.966667
```

- **Hint 09**：Plot the figure of training and testing datasets with border line.

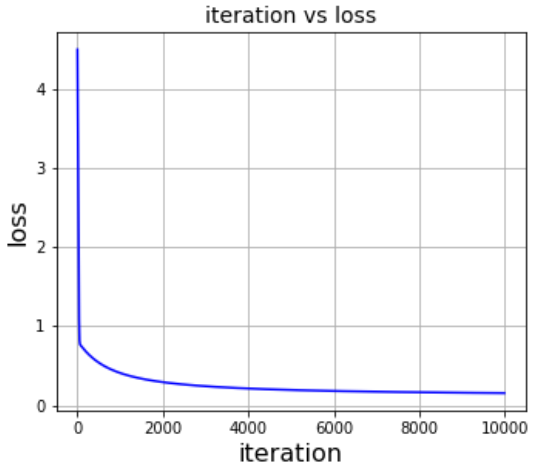
(Below are sample code and figure you should present)

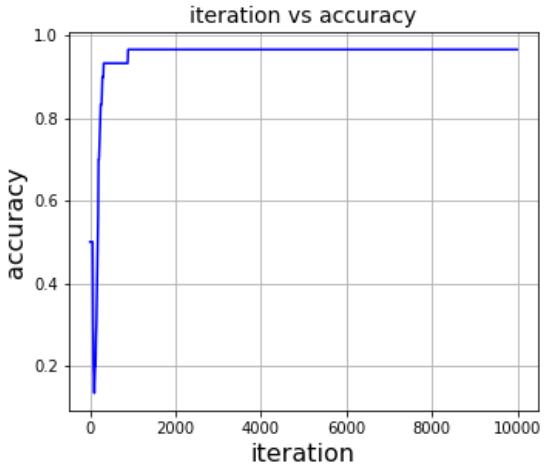
```
# plot the bord line
def cal_x2(x, w):
    return(-(w[0] + w[1] * x)/ w[2])

# for traininr figure
x_train0 = x_train[y_train==0]
x_train1 = x_train[y_train==1]

# find the max & min of x1
xl_train = np.asarray([x_train[:,1].min(), x_train[:,1].max()])
yl_train = cal_x2(xl_train, update_weight)

plt.figure(figsize=(6,6))
plt.title('classify Setosa and Versicolour with training datasets')
plt.scatter(x_train0[:,1], x_train0[:,2], marker='x', c='b', s=50, label='class 0:Setosa')
plt.scatter(x_train1[:,1], x_train1[:,2], marker='o', c='k', s=50, label='class 1:Versicolour')
plt.plot(xl_train, yl_train, c='r', label='borderline')
plt.xlabel('Sepal Length', fontsize=14)
plt.ylabel('Sepal Width', fontsize=14)
plt.xticks()
plt.yticks()
plt.legend()
plt.grid(True)
plt.show()
```





- **Hint 10**：Plot the figure of "**iteration vs. loss**" and "**iteration vs. accuracy**".

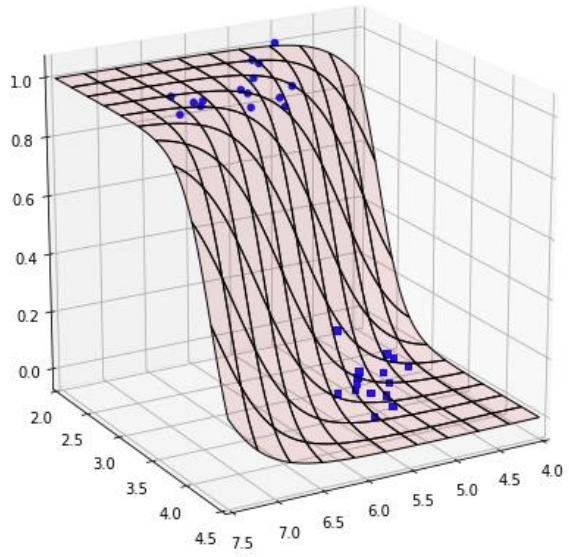




- **Hint 11**：Discuss what you find from figure in Hint 09 ?

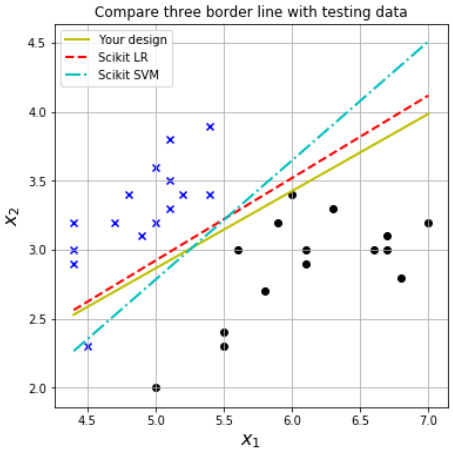
Bonus：Show in 3D plot.

```
from mpl_toolkits.mplot3d import Axes3D
x1 = np.linspace(4, 7.5, 100)
x2 = np.linspace(2, 4.5, 100)

xx1, xx2 = np.meshgrid(x1, x2) # Return coordinate matrices from coordinate vectors.
xxx = np.asarray([np.ones(xx1.ravel().shape), xx1.ravel(), xx2.ravel()]).T
c = predict_(xxx, update_weight).reshape(xx1.shape)
plt.figure(figsize=(8,8))
ax = plt.subplot(1, 1, 1, projection='3d')
ax.plot_surface(xx1, xx2, c, color='red', edgecolor='black', rstride=10, cstride=10, alpha=0.1)
ax.scatter(x_test1[:, 1], x_test1[:, 2], 1, s=20, alpha=0.9, marker='o', c='b')
ax.scatter(x_test0[:, 1], x_test0[:, 2], 0, s=20, alpha=0.9, marker='s', c='b')
ax.set_xlim(4, 7.5) # axis limit
ax.set_ylim(2, 4.5)
ax.view_init(elev=20, azim=60)
```



Bonus：Use Scikit-learn

➢ **Reference link：Scikit learn；SVM introduction**




```python
from sklearn.linear_model import LogisticRegression
from sklearn import svm

# Generate model：Logistic regression & SVM
model_lr  = LogisticRegression (solver='liblinear')
model_svm = svm.SVC (kernel='linear')
#calculate
model_lr.fit (x_train, y_train)
model_svm.fit(x_train, y_train)

# LR
lr_w0 = model_lr.intercept_[0] # 截距, 1x3 matrix
lr_w1 = model_lr.coef_[0, 1]   # x1(sepal_length)  # 斜率, 2x3 matrix
lr_w2 = model_lr.coef_[0, 2] # x2(sepal_width)
# SVM
svm_w0 = model_svm.intercept_[0]
svm_w1 = model_svm.coef_[0, 1] # x1(sepal_length)
svm_w2 = model_svm.coef_[0, 2] # x2(sepal_width)

#find x2
def rl(x):
    x2 = -(lr_w0 + lr_w1 * x)/lr_w2
    return(x2)
def svm(x):
    x2 = -(svm_w0 + svm_w1 * x)/svm_w2
    return(x2)

y_rl  = rl(xl_test)
y_svm = svm(xl_test)
print(xl_test, yl_test, y_rl, y_svm)

# plot scatter figure with border line
fig = plt.figure(figsize=(6,6))
ax = fig.add_subplot(1,1,1)
plt.title('Compare three border line with Testing data')
plt.scatter(x_test0[:,1], x_test0[:,2], marker='x', c='b')
plt.scatter(x_test1[:,1], x_test1[:,2], marker='o', c='k')

ax.plot(xl_test, yl_test, linewidth=2, c='y', label='Your design')
ax.plot(xl_test, y_rl,    linewidth=2, c='r', linestyle="--", label='Scikit LR')
ax.plot(xl_test, y_svm,   linewidth=2, c='c', linestyle="-.", label='Scikit SVM')

ax.legend()
ax.set_xlabel('$x_1$', fontsize=16)
ax.set_ylabel('$x_2$', fontsize=16)
plt.grid(True)
plt.show()
```