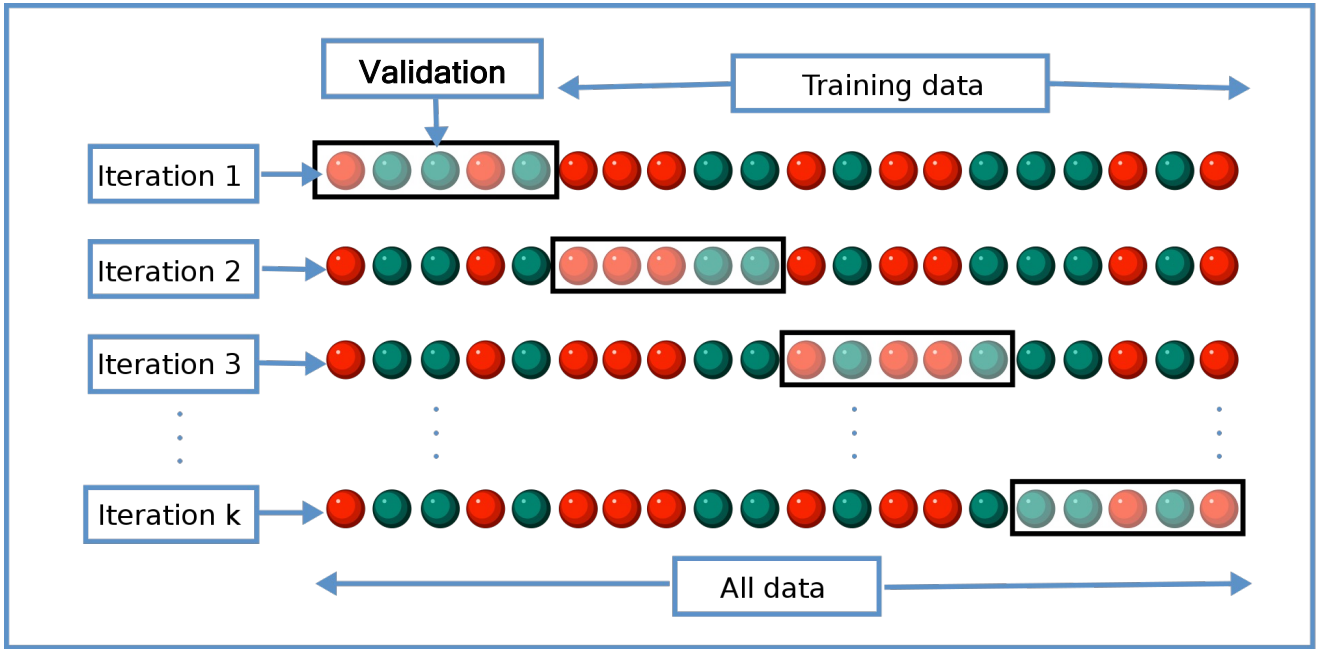
THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

COMP4434 Big Data Analytics

Lab 3 K-fold Cross-Validation & Logistic Regression Classifier

HUANG Xiao
xiaohuang@comp.polyu.edu.hk

k-fold Cross-validation



- The original sample is randomly partitioned into k equal sized subsamples
- Of the k subsamples, a single subsample is retained as the validation data for testing the model
- The remaining k − 1 subsamples are used as training data
- The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation data
- The k results can then be averaged to produce a single estimation

Implementation without using scikit-learn

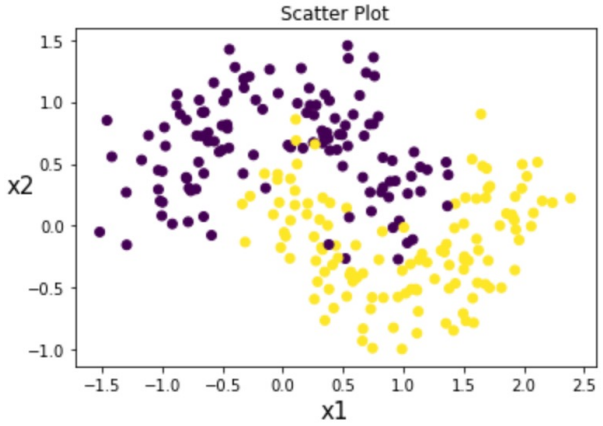
```
import numpy as np

def k_fold_cv(data, labels, k=5):
n = data.shape[0]
fold_size = n // k
indices = np.arange(n)
np.random.shuffle(indices)

for fold in range(k):
start = fold * fold_size
end = (fold + 1) * fold_size
test_indices = indices[start:end]
train_indices = np.concatenate([indices[:start], indices[end:]])
train_data, train_labels = data[train_indices], labels[train_indices]
test_data, test_labels = data[test_indices], labels[test_indices]

# train and evaluate model on train_data and train_labels
# test model on test_data and test_labels
```

Example when scikit-learn is used

- Load data and run models with cross-validation using sklearn.
- We have import LogisticRegression from sklearn.linear_model. We will learn it soon.
- Try different validation metrics by using
  from sklearn.metrics import
  We will learn mean_squared_error and accuracy_score soon.
- More functions and examples from sklearn:
  https://scikit-learn.org/stable/modules/cross_validation.html#computing-cross-validated-metrics

Logistic regression classifier

In [65]:
```
import numpy as np
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt

def Load_data():
    x, y = make_moons(250, noise=0.25)
    x_new = np.linspace(-1.5, 2.5)
    return x,y,x_new

x, y, x_new = Load_data()

col = ['r', 'b']
plt.figure()
plt.scatter(x[:,0],x[:,1],c=y)
plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Scatter Plot')
```



Initialize model parameters

- dim: the number of features
- w: weight
- b: bias
- eta: learning rate
- n_iterations: number of total iterations

In [29]:
```
class LogisticRegressionUsingGD:
    def __init__(self, eta, n_iterations):
        self.dim = 2
        self.w = np.array([1.0,1.0])
        self.b = 0
        self.eta = eta
        self.n_iterations = n_iterations
```

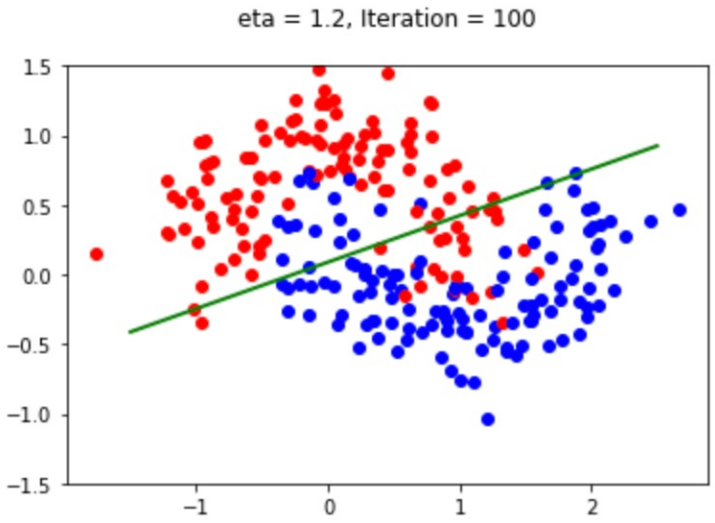
Define the Training function

Gradient formulation from lecture: $\frac{\partial J(\theta_0,\theta_1,\cdots)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$

In [31]:
```
def fit(self,x,y,eta):
    itr = 0
    self.eta = eta
    row, column = np.shape(x)
    print('number of samples:', row)
    while itr <= self.n_iterations:
        fx = np.dot(self.w, x.T) + self.b
        hx = self.sigmoid(fx)
        t = (hx-y)
        """
            Parameters
            ----------
            i[0]:t
            i[1][0]:x1
            i[1][1]:x2
            Returns
            -------
        """

        s = [[i[0]*i[1][0],i[0]*i[1][1]] for i in zip(t,x)]
        gradient_w = np.sum(s, 0)/row * self.eta
        gradient_b = np.sum(t, 0)/row * self.eta
        self.w -= gradient_w
        self.b -= gradient_b
        itr += 1
```

Train the model

```
if __name__ == '__main__':
    import matplotlib.pyplot as plt
    x, y = make_moons(250, noise=0.25)
    xpts = np.linspace(-1.5, 2.5)

    col = {0:'r',1:'b'}
    model = LogisticRegressionUsingGD(eta=1.2, n_iterations=100)
    model.fit(x,y)
    ypts= model.predict(xpts)

    plt.figure()
    for i in range(250):
        plt.plot(x[i, 0], x[i, 1], col[y[i]] + 'o')
    plt.ylim([-1.5,1.5])
    plt.plot(xpts,ypts, 'g-', lw = 2)
    plt.title('eta = %s, Iteration = %s\n' % (str(model.eta), str(model.n_iterations)))
    plt.show()
```



Some Practice

- Implement predict function
- Print the model's accuracy
- Using Logistic Regression to classify the IRIS dataset from sklearn

- Further readings:
  - https://realpython.com/logistic-regression-python/