THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

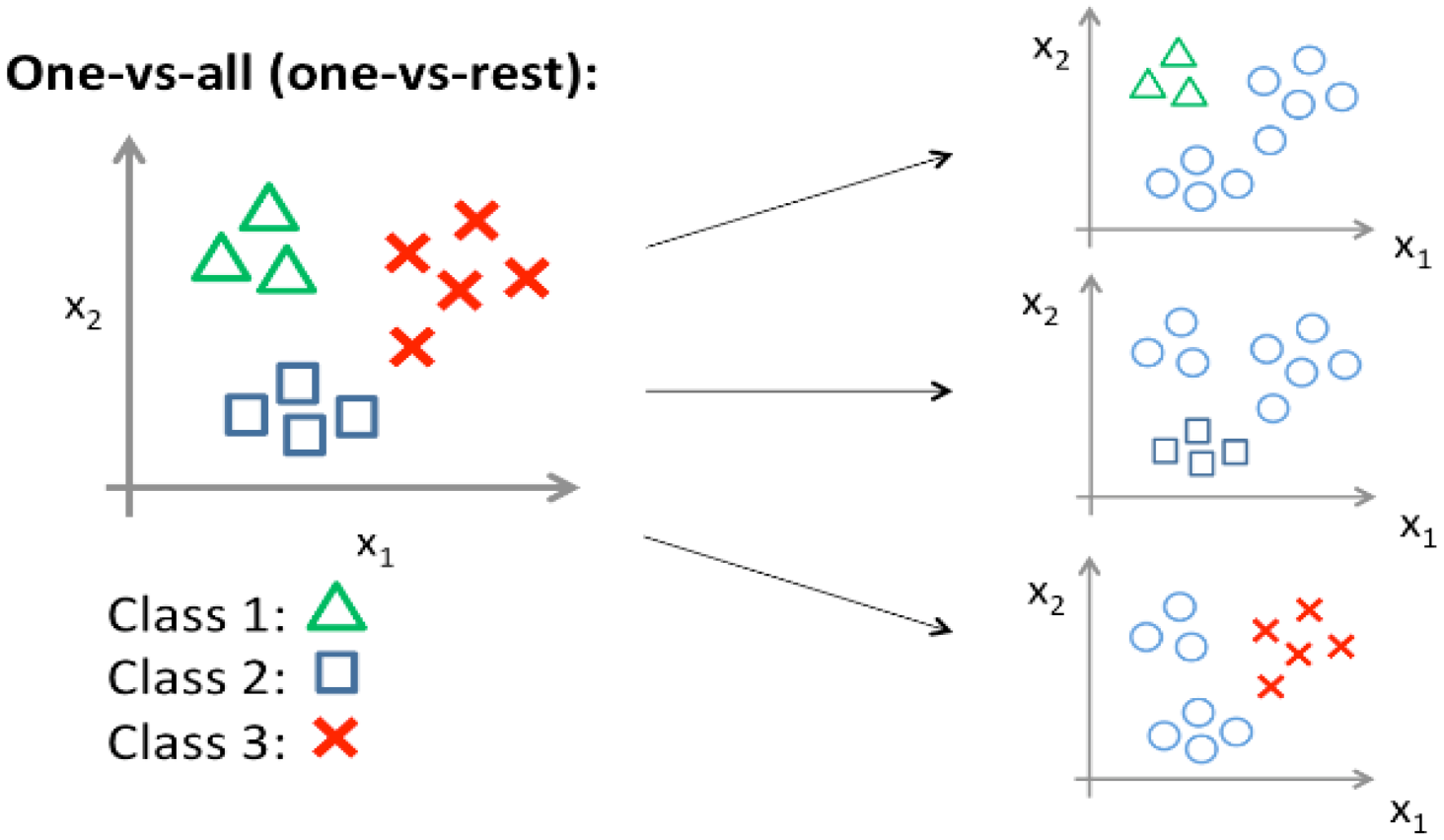
COMP4434 Big Data Analytics

Lab 5 One-vs-All Approach

HUANG Xiao

xiaohuang@comp.polyu.edu.hk

One-vs-All Approach

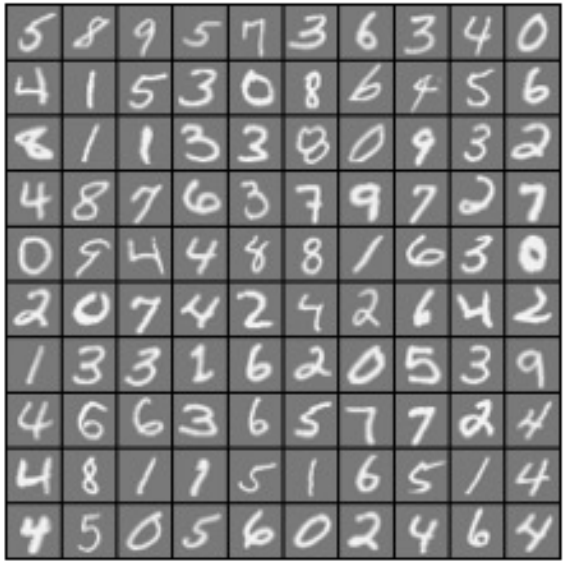


Load Dataset

```
def loadmat_data(fileName):
    return spio.loadmat(fileName)

def display_data(imgData):
    sum = 0
    pad = 1
    display_array = -np.ones((pad+10*(20+pad),pad+10*(20+pad)))
    for i in range(10):
        for j in range(10):
            display_array[pad+i*(20+pad):pad+i*(20+pad)+20,pad+j*(20+pad)
            sum += 1

    plt.imshow(display_array,cmap='gray')    #Display grayscale images
    plt.axis('off')
    plt.show()
```



Cost function and gradient decent

Gradient formulation won't change in One-vs-All

```
def costFunction(initial_theta,X,y,inital_lambda):
    m = len(y)
    J = 0

    h = sigmoid(np.dot(X,initial_theta))
    theta1 = initial_theta.copy()
    theta1[0] = 0

    temp = np.dot(np.transpose(theta1),theta1)
    J = (-np.dot(np.transpose(y),np.log(h))-np.dot(np.transpose(1-y),np.log(1-h))
    return J
```

```
def gradient(initial_theta,X,y,inital_lambda):
    m = len(y)
    grad = np.zeros((initial_theta.shape[0]))

    h = sigmoid(np.dot(X,initial_theta))
    theta1 = initial_theta.copy()
    theta1[0] = 0

    grad = np.dot(np.transpose(X),h-y)/m+inital_lambda/m*theta1
    return grad
```

← Normalization term

Predict Labels

```
def predict_oneVsAll(all_theta,X):
    m = X.shape[0]
    num_labels = all_theta.shape[0]
    p = np.zeros((m,1))
    X = np.hstack((np.ones((m,1)),X))
    h = sigmoid(np.dot(X,np.transpose(all_theta)))
    p = np.array(np.where(h[0,:] == np.max(h, axis=1)[0]))
    for i in np.arange(1, m):
        t = np.array(np.where(h[i,:] == np.max(h, axis=1)[i]))
        p = np.vstack((p,t))
    return p
```

The highest probability

Training using gradient decent

```
# Map y
for i in range(num_labels):
    class_y[:,i] = np.int32(y==i).reshape(1,-1)

#np.savetxt("class_y.csv", class_y[0:600,:], delimiter=',')

for i in range(num_labels):
    #optimize.fmin_cg
    result = optimize.fmin_bfgs(costFunction, initial_theta, fprime=gradient, args=(X,class_y[:,i],Lambda))
    all_theta[:,i] = result.reshape(1,-1)    # put into all_theta

all_theta = np.transpose(all_theta)
return all_theta
```

Training 10 logistic regression

Some Practice

Further tasks:

- Implement the fit function without 3rd party packages
- Print the model's Loss curve

Further readings:

- https://towardsdatascience.com/multiclass-classification-algorithm-from-scratch-with-a-project-in-python-step-by-step-guide-485a83c79992
- https://houxianxu.github.io/implementation/One-vs-All-LogisticRegression.html
- https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/
- https://github.com/lawlite19/MachineLearning_Python/tree/master/LogisticRegression