Logistic Regression

- Overview
  - Aim to categorizing cases into one of the classes.
  - The dependent variable Y is also called label, and especially Y is binary case, Y is expressed as 0 or 1.
  - Logistic regression uses logistic function:

$$g(z) = \frac{1}{1 + \exp(-z)}$$

Logistic Regression

- Logistic regression model
  - p(y=1) is success probability

$$log\left(\frac{p(y=1)}{1-p(y=1)}\right) = \alpha + \beta \cdot x$$

  - β determines the rate of increase or decrease of the curve

한양대학교 HANYANG UNIVERSITY

Logistic Regression

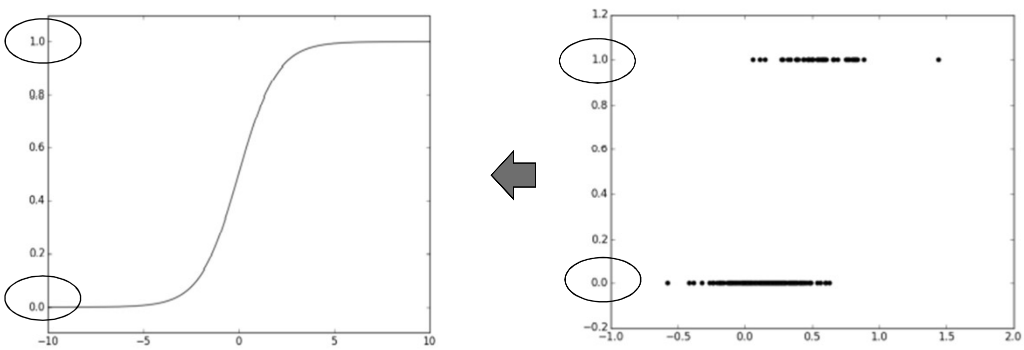
- Logistic function
  - The success probability is expressed as a logistic function

$$p(y=1) = \frac{1}{1+\exp(-\alpha - \beta \cdot x\ )}$$

  - For high dimensional vectors, the geometry is only determined by $\alpha + \beta \cdot x$!
  - Regression coefficients can be solved by numerical algorithms..!

Logistic Regression

- Logistic function



한양대학교 HANYANG UNIVERSITY

Logistic Regression

- Recall:
  - Benoulli's distribution

$$y_i = \begin{cases} 1, & p \\ 0, & 1-p \end{cases}$$

  - Likelihood function

$$L(\theta) = \prod_{i=1}^{n} p^{y_i}(1-p)^{1-y_i}$$

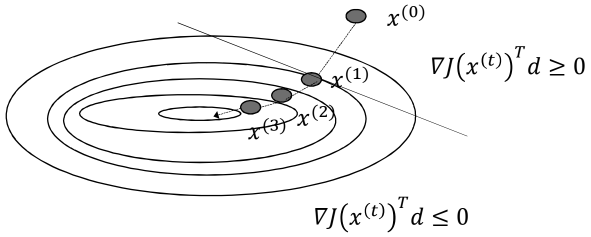
  - In general, y_i can be also converted to (1, -1) instead of (0, 1)

Logistic Regression

- Gradient descent algorithm

$$x^{(t+1)} = x^{(t)} - \gamma^{(t)} \cdot \nabla f\left(x^{(t)}\right)$$

  - $\gamma^{(t)}$ is step size for moving



한양대학교 HANYANG UNIVERSITY

Logistic Regression

- Gradient descent algorithm
  - $\nabla f\left(x^{(t)}\right)$ is perpendicular to the contour
  - descent direction (d) can be either positive or negative side.
  - Only those on the negative side reduce the cost!

Logistic Regression

- Gradient descent algorithm

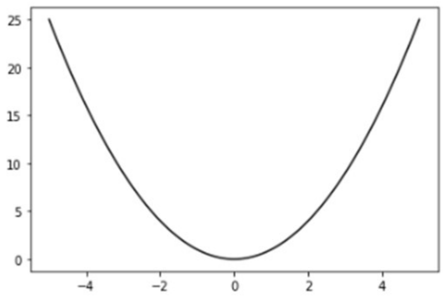
$$f\left(x^{(t)} + d\right) \approx f\left(x^{(t)}\right) + \nabla f\left(x^{(t)}\right)^T d + \frac{1}{2\gamma} \|d\|^2$$

  - To minimize the above approximated function, d needs to be - $\gamma \cdot \nabla f\left(x^{(t)}\right)$
  - when γ of the approximation is not too big, it converges.!

Logistic Regression

- Example
  - Find the minimum points using gradient descent algorithm

$$f(x) = x^2$$



한양대학교 HANYANG UNIVERSITY

Logistic Regression

- Example
  - First you have to define target and gradient functions.

```
In [17]: def target_func(x):
             return x*x

         def step(x, direction, step_size):
             return x + step_size * direction

         def gradient_func(x):
             return 2*x
```

한양대학교 HANYANG UNIVERSITY

Logistic Regression

- Example
  - Find the minimum points

```
def finding_min(target_func, gradient_func, theta_0, tolerance=0.0000001):
    learning_rates = [100, 10, 1, 0.1, 0.01, 0.001, 0.0001, 0.00001]

    theta = theta_0
    value = target_func(theta)


    while True:
        gradient = gradient_func(theta)
        n_thetas = [step(theta, gradient, -step_size) for step_size in learning_rates]

        n_theta = min(n_thetas, key = target_func)
        n_value = target_func(n_theta)

        print("theta and value :",(n_theta, n_value))

        if abs(value- n_value) < tolerance:
            return theta
        else :
            theta, value = n_theta, n_value
```


Logistic Regression

- Example
  - Conducting the code..

```
In [7]: x = np.linspace (-5, 5, 100)
        theta_0 = random.random()

        theta, value = finding_min(target_func, gradient_func, theta_0, tolerance=0.0000001)
        print("")
        print("theta and value are {} and {}".format(np.round(theta,6), np.round(value, 6)))
```

Logistic Regression

- Example

```
theta and value : (0.026377626764730416, 0.0006957791937394224)
theta and value : (0.021102101411784334, 0.0004452986839932304)
theta and value : (0.016881681129427468, 0.0002849911577556675)
theta and value : (0.013505344903541975, 0.0001823943409636271 8)
theta and value : (0.010804275922833579, 0.0001167323782167213 9)
theta and value : (0.008643420738266863, 7.470872205870169e-05)
theta and value : (0.006914736590613491, 4.781358211756908 4e-05)
theta and value : (0.005531789272490793, 3.060069255524421e-05)
theta and value : (0.004425431417992634, 1.9584443235356296e-05)
theta and value : (0.003540345134394107 3, 1.253404367062803e-05)
theta and value : (0.002832276107515285 6, 8.021787949201937e-06)
theta and value : (0.002265820886012228 4, 5.13394428748924e-06)
theta and value : (0.001812656708809782 7, 3.285724343993113 6e-06)
theta and value : (0.001450125367047826 2, 2.1028635801555926e-06)
theta and value : (0.001160100293638261, 1.3458326912995792e-06)
theta and value : (0.000928080234910608 7, 8.613329224317307e-07)
theta and value : (0.000742464187928487, 5.512530703563076e-07)
theta and value : (0.000593971350342789 6, 3.528019650280369e-07)
theta and value : (0.000475177080274231 7, 2.2579325761794362e-07)
theta and value : (0.0003801416642193853 5, 1.4450768487548391e-07)
```

```
theta and value are 0.000475 and 0.0
```

Logistic Regression

- Appling with a dataset

  - Train set — Fitting a model
  - Test set — Making prediction

Logistic Regression

train_test_split (data, test_size, shuffle, stratify)

-. *data: list, array or dataframe*
-. *test_size: percentage of the dataset to test split, between 0 and 1*
-. *shuffle = True or False: for True, shuffle before splitting*
-. *stratify : None is default. Stratified using class labels (y)*

Logistic Regression

LogisticRegression(tol, solver)

-. *tol: set the tolerance for stopping,1e-4 by default*
-. *solver: choose a solver such as ‘liblinear’ , ‘lbfgs’ and etc*

Logistic Regression

.fit(x,y) *: fit the model with x and y*

-. .intercept_: return the intercept of the model
-. .coef_: return the regression coefficients
-. .classes_ : return the class labels
-. .predict_proba(x) : the first column is the probability of the output being zero, and the second column is that of being one, p(x)
-. .predict (x) : return the predicted output values as a 1-d array
-. .score (x, y) : return the ratio of the correct prediction

Logistic Regression

- Example
  - Read the 'Lec10_logi.csv'.

```
In [8]: data1 = pd.read_csv("Lec10_logi.csv")
        data1.head()
```

Out [8]:

	Y	X1	X2	X3	X4	X5	X5.1
0	1	0.8	0.83	0.66	1.9	1.10	1.00
1	1	0.9	0.36	0.32	1.4	0.74	0.99
2	0	0.8	0.88	0.70	0.8	0.18	0.98
3	0	1.0	0.87	0.87	0.7	1.05	0.99
4	1	0.9	0.75	0.68	1.3	0.52	0.98

Class labels

X

Logistic Regression

- Example
  - Data split

```
In [10]: from sklearn.model_selection import train_test_split

         x_train, x_test,y_train, y_test = train_test_split(data1[['X1','X2','X3','X4','X5','X5.1']],
                                                            data1['Y'],
                                                            test_size=0.2,
                                                            shuffle=True)

In [14]: print("the shape of x_train is ", x_train.shape)
         print("the shape of x_test is ", x_test.shape)
         print("the shape of y_train is ", y_train.shape)
         print("the shape of y_test is ", y_test.shape)

         the shape of x_train is  (21, 6)
         the shape of x_test is  (6, 6)
         the shape of y_train is  (21,)
         the shape of y_test is  (6,)
```

한양대학교 HANYANG UNIVERSITY

Logistic Regression

- Example
  - Fitting the model

```
In [15]: from sklearn.linear_model import LogisticRegression

         model1 = LogisticRegression(tol=1e-06).fit(x_train, y_train)

In [18]: print("class labels : ", model1.classes_)
         print("regression parameters are ", model1.coef_)
         print("intercept is ", model1.intercept_)
         print("model score is ", model1.score(x_train, y_train))

         class labels :  [0 1]
         regression parameters are  [[ 0.37970473  0.07453902  0.26393052  1.06698789  0.56709841 -0.02842297]]
         intercept is  [-2.71462321]
         model score is  0.7619047619047619
```

한양대학교 HANYANG UNIVERSITY

Logistic Regression

- Example
  - Prediction with test dataset

```
In [21]: model1.predict_proba(x_test)

Out[21]: array([[0.5929804 , 0.4070196 ],
                [0.51677186, 0.48322814],
                [0.80189279, 0.19810721],
                [0.57321136, 0.42678864],
                [0.6185483 , 0.3814517 ],
                [0.75809132, 0.24190868]])

In [24]: y_pred = model1.predict(x_test)
         print("predicted labels are ", y_pred)

         predicted labels are  [0 0 0 0 0 0]
```


Logistic Regression

- Example
  - Evaluation : How well ?

```
In [26]: from sklearn import metrics

         conf_matrix = metrics.confusion_matrix(y_test, y_pred)
         print(conf_matrix)

         [[4 0]
          [2 0]]
```

Logistic Regression

- Understanding confusion matrix
  - Contingency table between the true labels and the predicted labels

	Predicted label =0	Predicted label =1
True label=0	True Negative (TN)	False Positive(FP)
True label=1	False Negative (FN)	True Positive (TP)

  - To summarize those values, we can make indices!!

한양대학교 HANYANG UNIVERSITY

Logistic Regression

- Understanding confusion matrix
  - Accuracy $\frac{TP+TN}{TP+TN+FP+FN}$
  - Recall $\frac{TP}{TP+FN}$
  - Precision $\frac{TP}{TP+FP}$

한양대학교 HANYANG UNIVERSITY

Logistic Regression

.classification_report(y_true, y_pred, label_name)
: return accuracy, recall, precision, f1-score

-. *y_true: true value of y labels*
-. *y_pred: predicted labels*
-. *label_name : set the label name in the table*

Logistic Regression

- Example
  - Evaluation : How well ?

```
In [27]: from sklearn.metrics import classification_report

         print(classification_report(y_test, y_pred, target_names=['class 0','class 1']))
```

```
              precision    recall  f1-score   support

     class 0       0.67      1.00      0.80         4
     class 1       0.00      0.00      0.00         2

    accuracy                           0.67         6
   macro avg       0.33      0.50      0.40         6
weighted avg       0.44      0.67      0.53         6
```