MACHINE LEARNING

Electrical Summer Workshops (ESW) 2022

Electrical Engineering Department - Sharif University of Technology

Instructors: *Alireza Gargoori Motlagh – Amir Mirrashid – Ali Nourian*

دانشگاه صنعتی شریف

دانشجویان دانشکده مهندسی برق

کانون علمی – فرهنگی رسانا

LOGISTIC REGRESSION

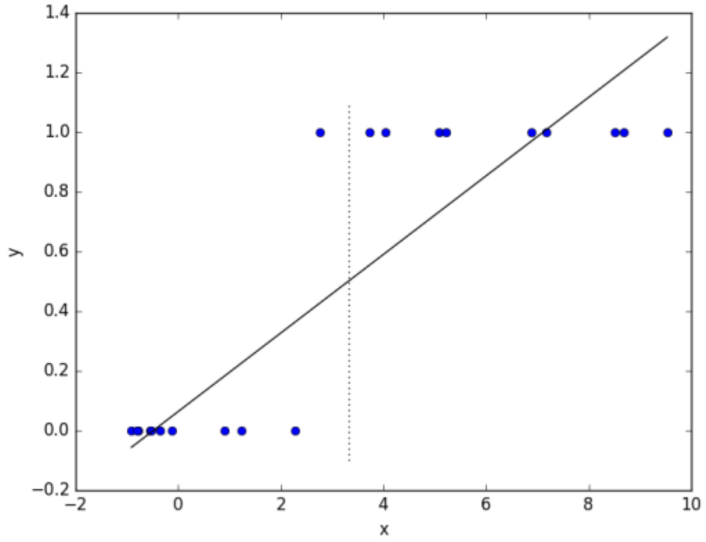
Linear Regression as Classifier

A basic approach to solve classification problems could use linear regression as follows: $\left(y^{(i)} \in \{0,1\}\right)$

$$RSS = \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)^2$$

$$\rightarrow y^{(i)} = \begin{cases} 1 & \boldsymbol{\beta}^T \boldsymbol{x}^{(i)} > 0.5 \\ 0 & otherwise \end{cases}$$

- Arbitrary threshold
- Optimal coefficients try to make data points hug the line as closely as possible, no matter the effect on classification performance



- Our loss function needs to be modified.

Logit Transform

Predicting probabilities : $\mathbb{R}^n \to (0,1)$

- Dot product to $\mathbb{R}_+^n$ through exponentiation
- Divide the result to $1 + itself$

$\to sigmoid$ function:

$$\sigma(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$$

where: $z = \boldsymbol{\beta}^T \boldsymbol{x}$ and $\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_p \end{bmatrix}$

$$\mathbb{P}(Y=1|\boldsymbol{X}) = \frac{1}{1+e^{-\boldsymbol{\beta}^T\boldsymbol{X}}}$$

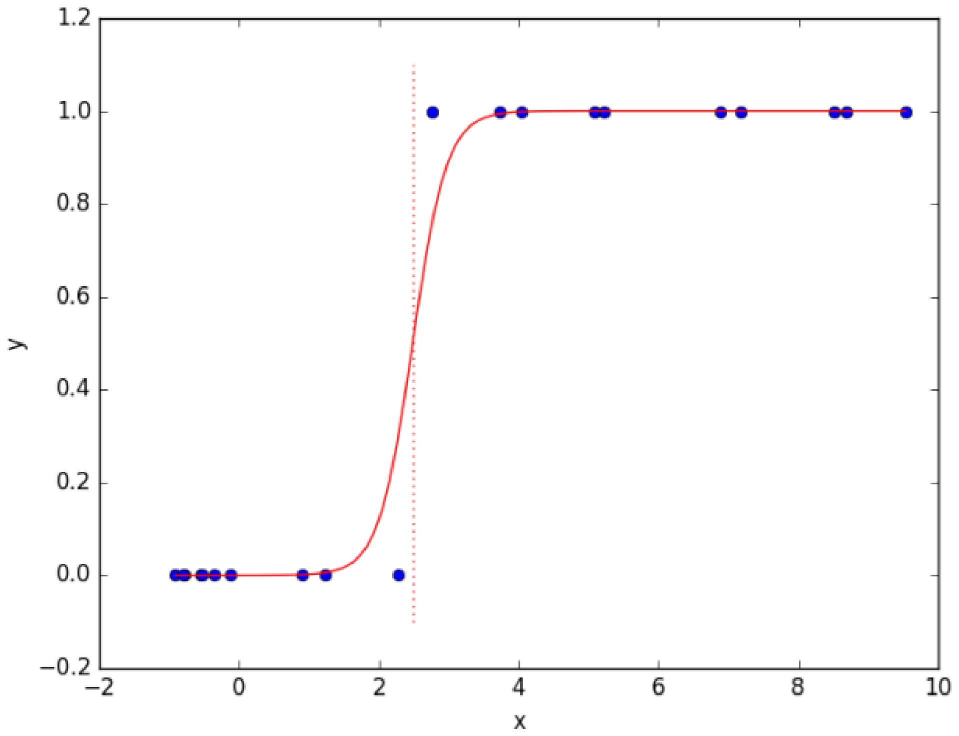
$$\mathbb{P}(Y=0|\boldsymbol{X}) = \frac{1}{1+e^{\boldsymbol{\beta}^T\boldsymbol{X}}}$$

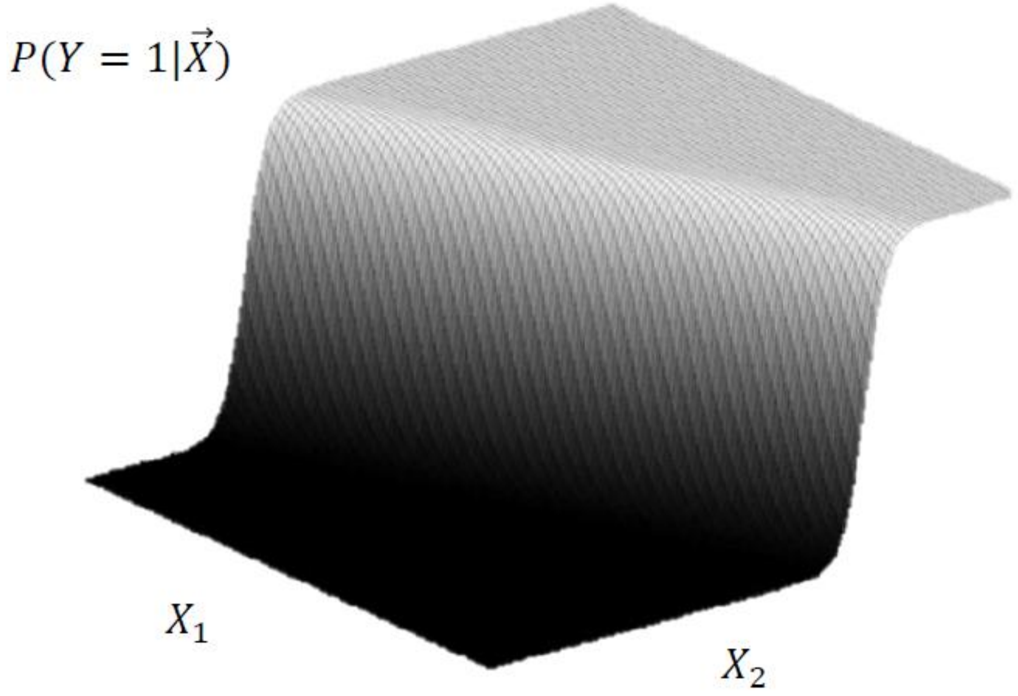
$$\xrightarrow{log-odds} \log\left(\frac{\mathbb{P}(Y=1|\boldsymbol{X})}{\mathbb{P}(Y=0|\boldsymbol{X})}\right) = \boldsymbol{\beta}^T\boldsymbol{X} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Logistic Regression Decision Boundary

$$y^{(i)} = \begin{cases} 1 & \sigma(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}) > 0.5 \\ 0 & otherwise \end{cases}$$

- Other thresholds might be needed in some cases!





Maximum Likelihood Estimation (MLE)

The problem of finding two probabilities reduces to estimation of regression parameters:

- *Maximum likelihood estimation (MLE)*: Finding the parameters that maximize the joint probability of occurring all observed samples

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \mathbb{P}\left(Y = y^{(i)} \middle| \boldsymbol{X} = \boldsymbol{x}^{(\boldsymbol{i})}\right) = \prod_{i=1}^{n} \mathbb{P}\left(Y = 1 \middle| \boldsymbol{X} = \boldsymbol{x}^{(\boldsymbol{i})}\right)^{y^{(i)}} \mathbb{P}\left(Y = 0 \middle| \boldsymbol{X} = \boldsymbol{x}^{(\boldsymbol{i})}\right)^{1-y^{(i)}}$$

$$\rightarrow L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(\boldsymbol{i})}\right)^{y^{(i)}} \left(1 - \sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(\boldsymbol{i})}\right)\right)^{1-y^{(i)}}$$

$$l(\boldsymbol{\beta}) = \log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} y^{(i)} \log\left(\sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(\boldsymbol{i})}\right)\right) + \left(1 - y^{(i)}\right)\left(\log\left(1 - \sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(\boldsymbol{i})}\right)\right)\right) =$$

$$= \sum_{i=1}^{n} \left(y^{(i)}\left(\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(\boldsymbol{i})}\right) - \log\left(1 + e^{\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(\boldsymbol{i})}}\right)\right)$$

Cost Function

Since sigmoid outputs probability, we use negative log likelihood to represent the error:

$$J(\boldsymbol{\beta}) = -\frac{1}{n} l(\boldsymbol{\beta}) = -\frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} \left(\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(\boldsymbol{i})} \right) - \log\left(1 + e^{\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(\boldsymbol{i})}} \right) \right)$$

We can update the parameters by minimizing the cost function (or equivalently, maximizing likelihood function)

Estimation of the Parameters

$$J(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}\right) - \log\left(1 + e^{\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}}\right)\right)$$

$$\frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)}\frac{e^{\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}}}{1 + e^{\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}}}\right) = -\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}^{(i)}\left(y^{(i)} - \frac{e^{\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}}}{1 + e^{\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}}}\right) =$$

$$\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}^{(i)}\left(\sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}\right) - y^{(i)}\right)$$

$$\rightarrow \frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}^{(i)}\left(\sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}\right) - y^{(i)}\right) = \frac{1}{n}\boldsymbol{X}^{T}\left(\boldsymbol{\sigma}(\boldsymbol{X\beta}) - \boldsymbol{y}\right)$$

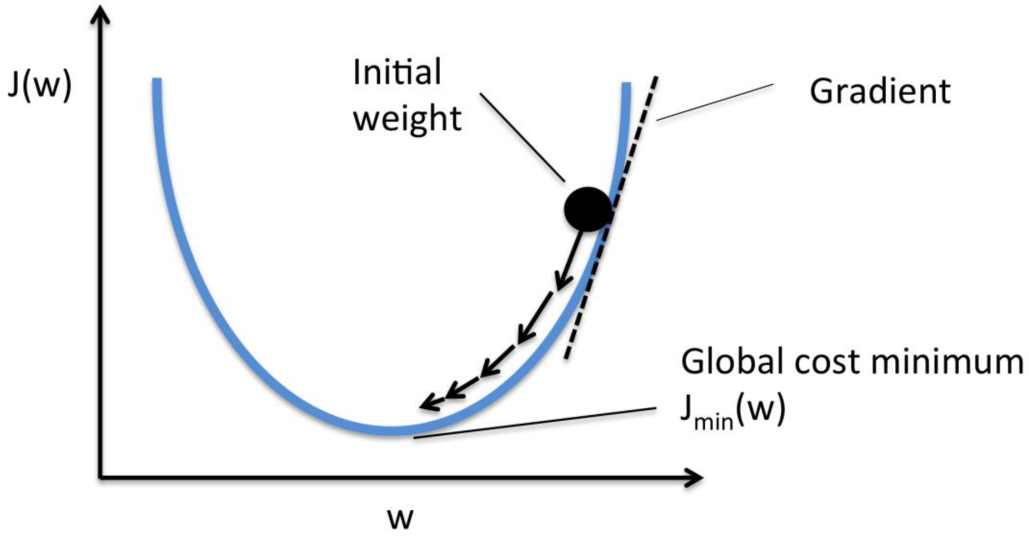
where $\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots\, x_p^{(1)} \\ \vdots & \vdots & \dots\quad \vdots \\ 1 & x_1^{(n)} & \dots\, x_p^{(n)} \end{bmatrix}$, $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$, $\boldsymbol{\sigma}(\boldsymbol{X\beta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{1})}\right) \\ \sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{2})}\right) \\ \vdots \\ \sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{n})}\right) \end{bmatrix}$

Updating the Parameters using Gradient Descent

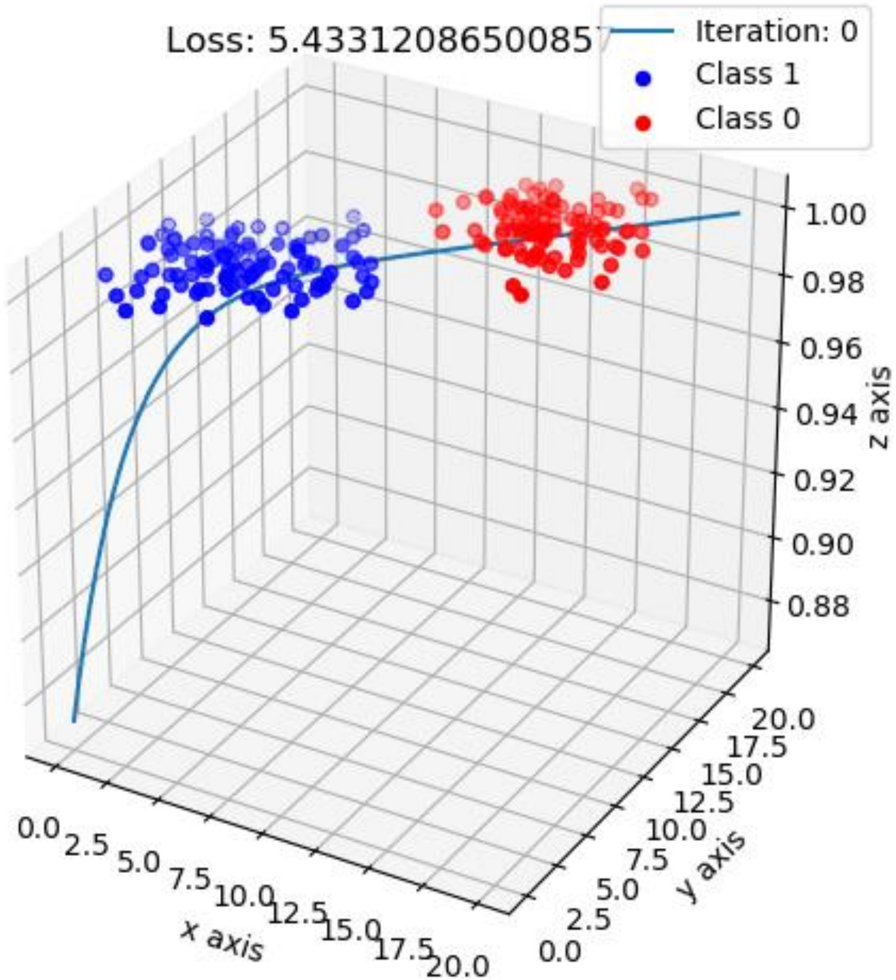
Now that we have the derivatives we can update the parameters using *Gradient Descent:*

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} - \alpha \frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta}^{old} - \alpha \frac{1}{n} \boldsymbol{X}^T (\boldsymbol{\sigma}(\boldsymbol{X\beta}) - \boldsymbol{y})$$

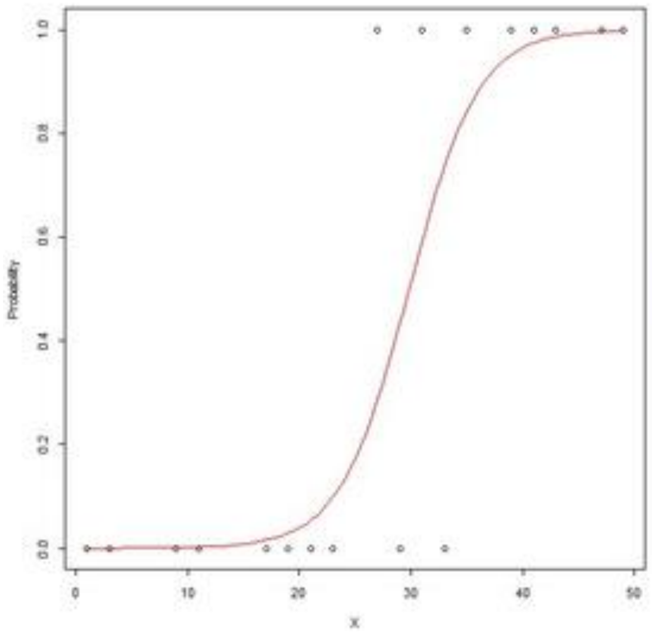
- $\alpha\ (learning\ rate)$: controls the rate of convergence (should be moderate).



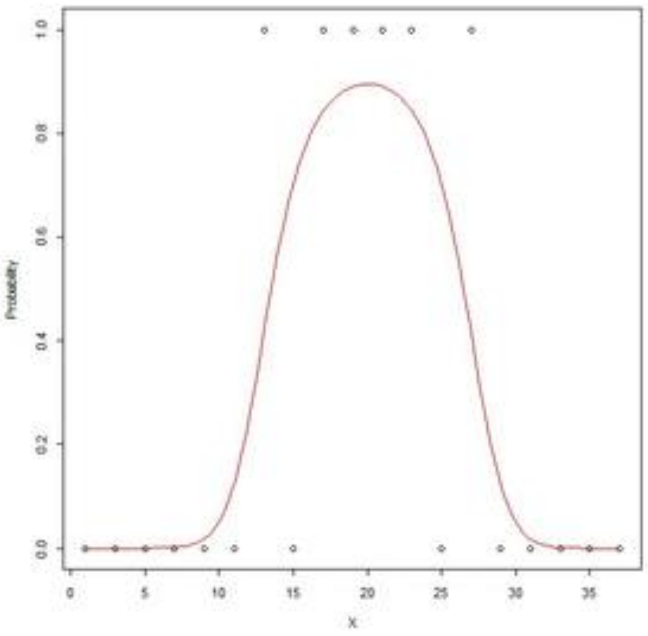
Example


Loss: 5.4331208650085
Iteration: 0
Class 1
Class 0
1.00
0.98
0.96
0.94
0.92
0.90
0.88
z axis
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
x axis
y axis

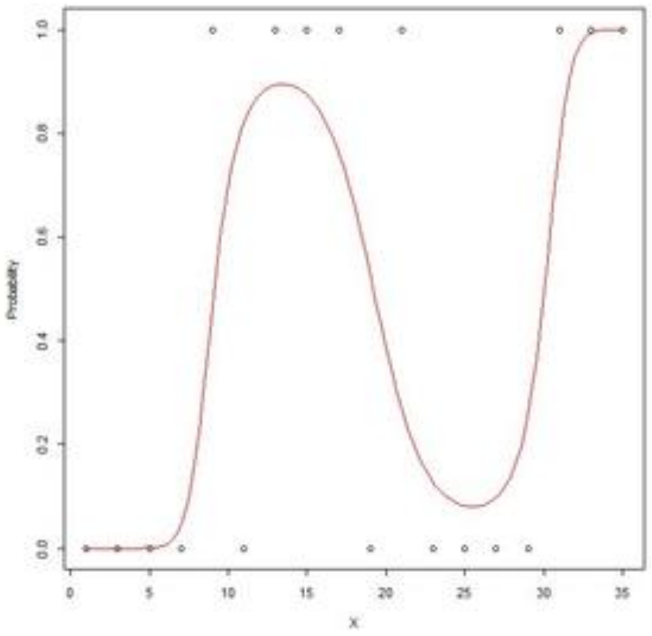
Non-Linear Logistic Regression



Linear



Quadratic



Cubic

*Multi-Class Logistic Regression

We can use logistic regression for multi-class classification as well. This is called *Multinomial Logistic Regression* or *Softmax Regression*.

- Each class has its own probability function with its own parameter vector, s.t. for any given x, they all sum to one.

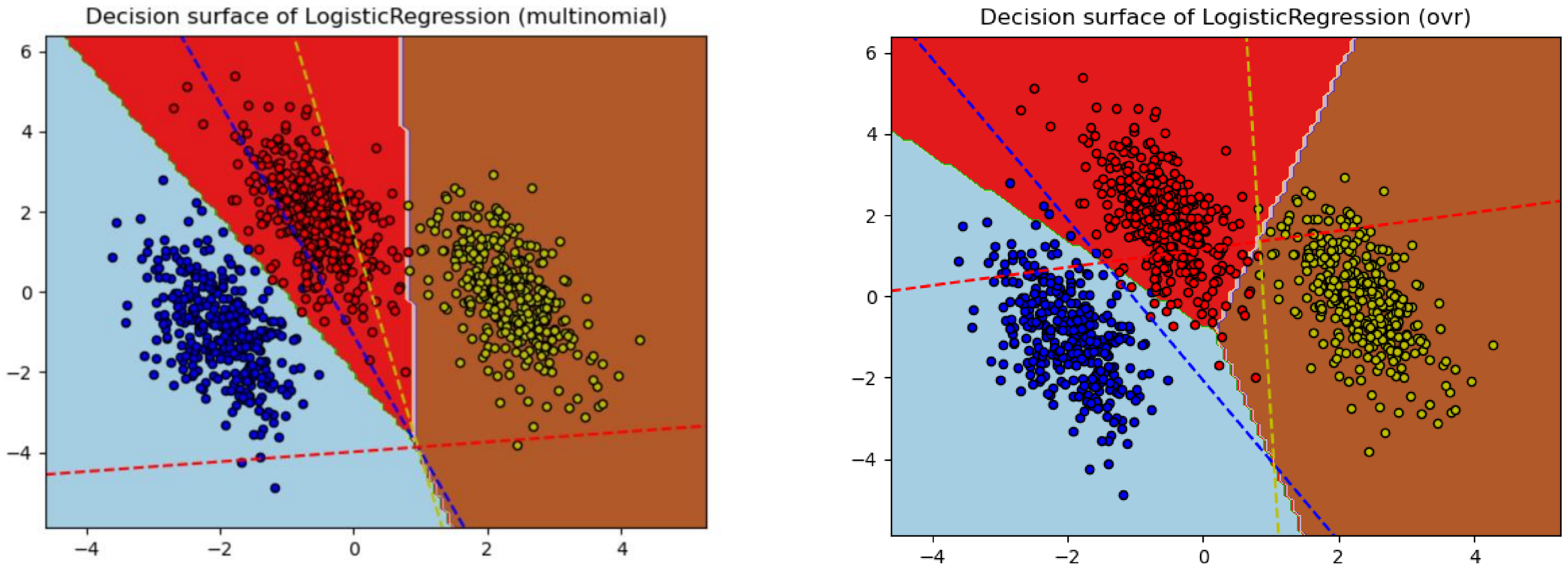
$$\mathbb{P}_j(\boldsymbol{x}, \boldsymbol{\beta}) = \mathbb{P}(Y = j | \boldsymbol{x}, \boldsymbol{\beta}) = \frac{e^{\boldsymbol{\beta}_j^T \boldsymbol{x}}}{\sum_{j=1}^{K} e^{\boldsymbol{\beta}_j^T \boldsymbol{x}}}, \qquad j = 1,2, \dots, K$$

Other approaches:

- One-vs-All
- One-vs-One

- In softmax model we should use penalty term to keep the parameters small. (L_2 or L_1 regularizers)

*Multi-Class Logistic Regression Decision Surface



*Regularized Logistic Regression

Like linear regression, in order to reduce the variance of the model, we can use regularization for the logistic regression as well:

- L_2 norm regularizer:

$$cost = -l(\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_{L2}^2 = -l(\boldsymbol{\beta}) + \lambda\sum_{j=1}^{p}\beta_j^2$$

- L_1 norm regularizer:

$$cost = -l(\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_{L1} = -l(\boldsymbol{\beta}) + \lambda\sum_{j=1}^{p}|\beta_j|$$

- Scikit-learn uses L_2 norm regularization by default.