MM 225 – AI and Data Science

Day 24: Supervised Learning : Logistic Regression

Instructors: Hina Gokhale, MP Gururajan, N. Vishwanathan

8 OCTOBER 2024

Outline

Data preparation for regression and logistic regression

Logistic Regression

Weights estimation using Gradient Descent

Data Preparation – EDA

1. Exploratory Data Analysis to get the “feel” for the data
2. Notice any outliers
    1. Decision to keep / replace / correct / remove
3. Missing Values
    1. Decision to replace or remove
4. Decision on the method of analysis – Supervised / unsupervised

Data Preparation - Scaling

Optimization methods are sensitive to scale

Rescaling of the data is necessary for reliable convergence

Rescaling of data to [0,1]

- let $x_j^{min} = \min\left(x_j^{(1)}, x_j^{(2)}, \dots, x_j^{(n)}\right); j = 1, 2, \dots, p$
- let $x_j^{max} = \max\left(x_j^{(1)}, x_j^{(2)}, \dots, x_j^{(n)}\right); j = 1, 2, \dots, p$
- $v_j^{(i)} = \frac{x_j^{(i)} - x_j^{min}}{x_j^{max} - x_j^{min}}$; $i = 1, 2, \dots, n$ *and* $j = 1, 2, \dots, p$
- Note that $0 \le v_j^{(i)} \le 1$

In case of regression $y^{(i)}, i = 1, 2, \dots, n$ also need to be rescaled similarly

Data Preparation – Scaling ….

Normalizing data

- rescaling the data such that its mean is 0 and variance is 1
- $\overline{x_j} = \sum_i x_j^{(i)}$ and $s_j = \sqrt{\frac{1}{n-1} \sum_i \left(x_j^{(i)} - \overline{x_j}\right)^2}$; $j = 1, 2, \dots, p$
- $v_j^{(i)} = \frac{x_j^{(i)} - \overline{x_j}}{s_j}$

In case of regression $y^{(i)}, i = 1, 2, \dots, n$ also need to be rescaled similarly

Data preparation – Training and Testing sets

Input features data = $\left(x_1, x_2, \dots x_p\right) \in \mathbb{R}^p \equiv \boldsymbol{x} \in \mathbb{R}^p$

Output data = $y \in \{0,1\}$

Total data set = $\left\{\left(\boldsymbol{x}^{(j)}, y^{(j)}\right): j = 1, 2, \dots, m\right\}$

Divide the data set in two parts randomly

- Training and Testing data
- General proportion is Training : Testing :: 70:30

Training data set = $\left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right): i = 1, 2, \dots, n\right\}$

Logistic Regression Model - Notation

Experiments are performed at various levels of input variable $(x_1, x_2, \dots x_p)$

Response y is binary:

- Success or failure
- Defective or Non defective
- …

Consider the scaled training set $\left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right): i = 1, 2, \dots, n\right\}$

the regression model :

$$\boldsymbol{y^{(i)} = w_0 + \sum_{j=1}^{p} w_j x_j^{(i)} + \epsilon_i}$$

For notational convenience we take $\boldsymbol{x}^{(i)} = (1, x_1^{(i)}, x_2^{(i)}, \dots, x_p^{(i)})$ **and w =**$(w_{\mathbf{0}}, w_1, \dots, w_p)$

Regression model can be written as dot product

$$\boldsymbol{y^{(i)} = w \cdot x^{(i)} + \epsilon_i}$$

Examples

Temp	Faulire
53	1
56	1
57	1
63	0
66	0
67	0
67	0
67	0
68	0
69	0
70	0
70	1
70	1
70	1
72	0
73	0
75	0
75	1
76	0
76	0
78	0
79	0
80	0
81	0

X_1	X_2	y
-0.869	0.389	0
-0.993	-0.611	0
-0.834	0.239	0
-0.136	0.632	1
0.404	0.311	1
-0.569	-0.247	0
-0.110	0.931	1
0.289	-0.533	1
0.320	0.665	1
0.559	-0.621	1
0.886	-0.777	0
0.289	-1.000	0
0.498	0.344	1
0.127	0.966	1
-0.728	0.331	0

Logistic Regression

Regression model :

$$y^{(i)} = \boldsymbol{w} \cdot \boldsymbol{x}^{(\boldsymbol{i})} + \epsilon_i$$

Note the mismatch:

LHS takes values either 0 or 1

RHS can take any value in $\mathbb{R}$

Way out is sigmoid transformation of $z_i = \boldsymbol{w} \cdot \boldsymbol{x}^{(\boldsymbol{i})}$

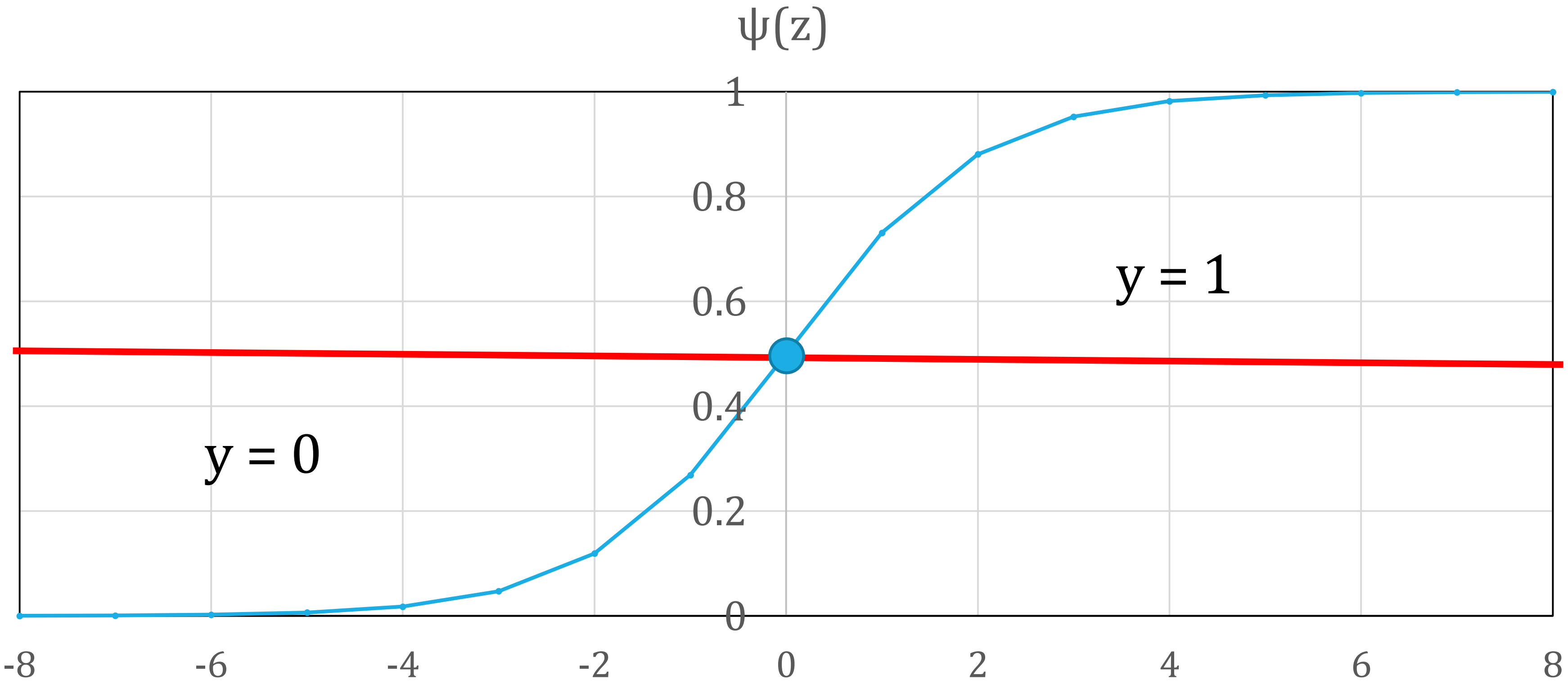
$$\psi(z_i) = \frac{1}{1+e^{-z_i}} = \frac{1}{1+\exp(-z_i)}$$

The sigmoid function & Threshold

if $\psi(z) >= 0.5$ then $y = 1$

if $\psi(z) < 0.5$ then $y = 0$

0.5 is called threshold



Weight estimation

$P(y^{(i)} = 1) = \psi(\boldsymbol{w} \cdot x^{(i)})$

$P(y^{(i)} = 0) = 1 - \psi(\boldsymbol{w} \cdot x^{(i)})$

Simplifying (recall Bernoulli trials!)

$$P(y^{(i)} | \boldsymbol{w}, x^{(i)}) = \left(\psi(\boldsymbol{w} \cdot x^{(i)})\right)^{y^{(i)}} \left(1 - \psi(\boldsymbol{w} \cdot x^{(i)})\right)^{1-y^{(i)}}$$

This is likelihood function of **w**,

$$L(\boldsymbol{w}) = \prod \left(\psi(\boldsymbol{w} \cdot x^{(i)})\right)^{y^{(i)}} \left(1 - \psi(\boldsymbol{w} \cdot x^{(i)})\right)^{1-y^{(i)}}$$

Taking natural log we get

$$\log(L(\boldsymbol{w})) = \sum y^{(i)} \log(\psi(\boldsymbol{w} \cdot x^{(i)})) + (1 - y^{(i)}) log(1 - \psi(\boldsymbol{w} \cdot x^{(i)}))$$

w can be estimated as maximum likelihood estimates by maximizing Log(L(**w**))

Weight Estimation

To estimate **w** it is convenient to minimize the – log(L(**w**))

$$= -\sum y^{(i)} \log\left(\psi\left(\boldsymbol{w} \cdot x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - \psi\left(\boldsymbol{w} \cdot x^{(i)}\right)\right)$$

This is called ***Cross Entropy Error Function $\mathcal{E}(w)$***

$$\mathcal{E}(\mathbf{w}) = -\sum y^{(i)} \log\left(\psi\left(\boldsymbol{w} \cdot x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - \psi\left(\boldsymbol{w} \cdot x^{(i)}\right)\right)$$

Weight Estimation

Want to minimize

$$\mathcal{E}(\mathbf{w}) = -\sum y^{(i)} \log\left(\psi\left(\boldsymbol{w} \cdot x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - \psi\left(\boldsymbol{w} \cdot x^{(i)}\right)\right)$$

This is not possible to minimise analytically

We need to use numerical methods

The method to be used is Gradient Descent

Cost Function / Loss Function

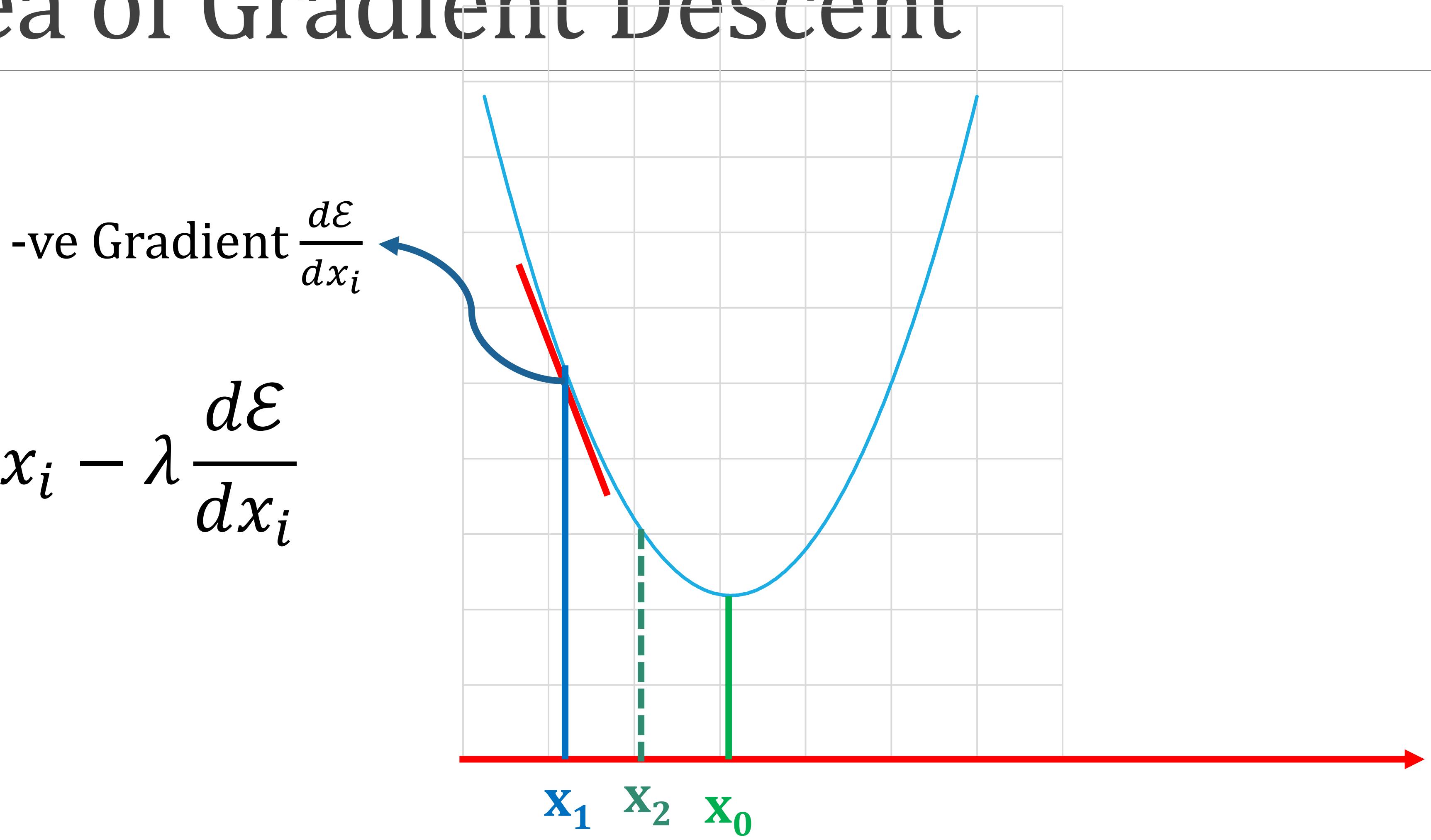
The functions

$$\sum_{i=1}^{n}\left[Y_i - \left(\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots + \hat{\beta}_k x_{ik}\right)\right]^2$$

$$-\sum y^{(i)} \log\left(\psi\left(\boldsymbol{w} \cdot x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - \psi\left(\boldsymbol{w} \cdot x^{(i)}\right)\right)$$

are in general called cost function or loss function

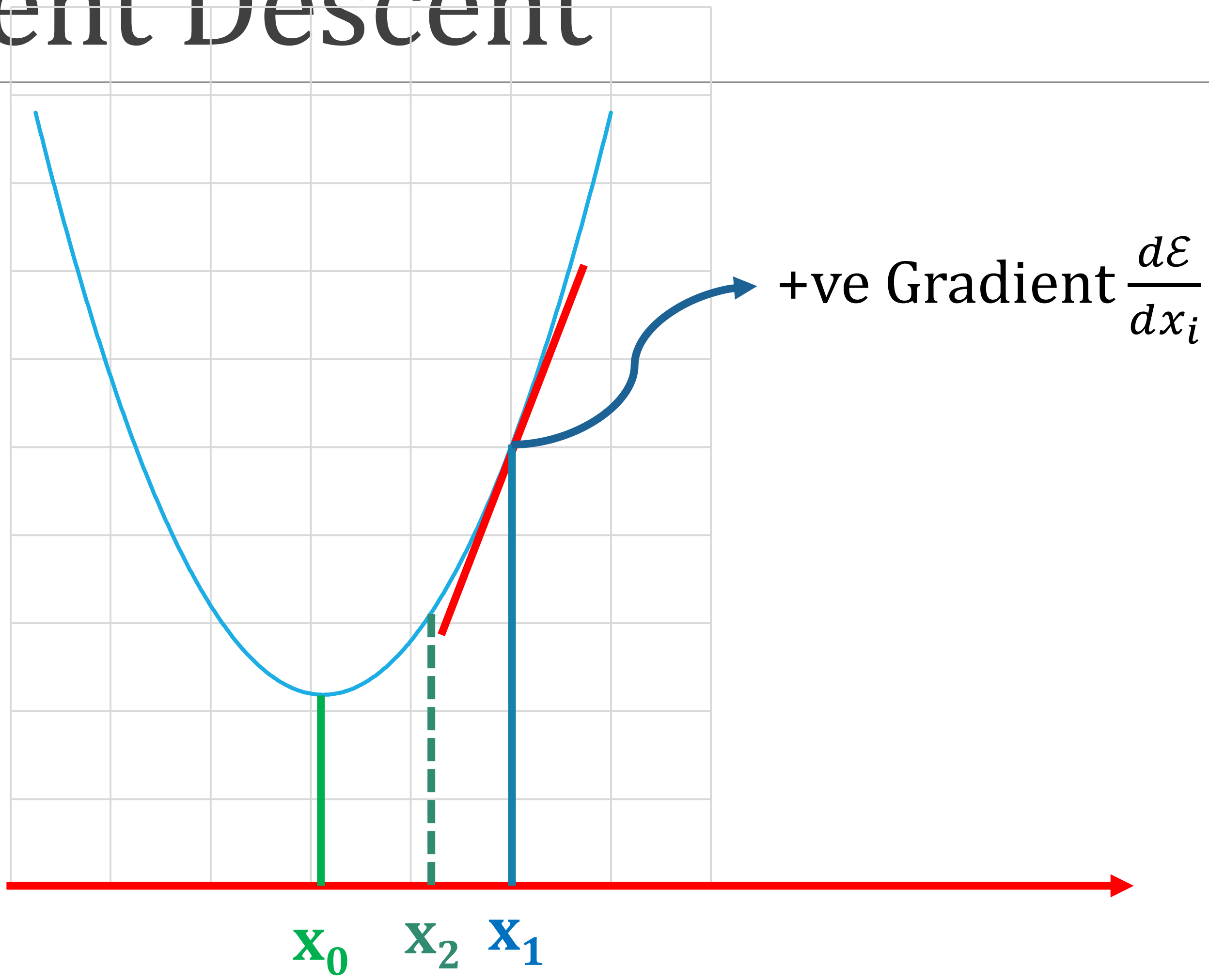
Idea of Gradient Descent



$$x_{i+1} \leftarrow x_i - \lambda \frac{d\mathcal{E}}{dx_i}$$

Idea of Gradient Descent

$$x_{i+1} \leftarrow x_i - \lambda \frac{d\mathcal{E}}{dx_i}$$



Gradient Descent Steps

Suppose we want to find $\min_{x} f(x)$

The steps are

1. Choose an initial value x_1
2. Find $\frac{df(x)}{dx_1}$
3. $x_2 = x_1 - \frac{df(x)}{dx_1} * \lambda$, λ is called learning rate
4. Find $\frac{df(x)}{dx_2}$
5. $x_3 = x_2 - \frac{df(x)}{dx_2} * \lambda$ …. and so on

Gradient Descent Steps

Suppose we want to find $\min_{x} f(x)$

The steps are

1. Choose an initial value x_1
2. Find $\frac{df(x)}{dx_1}$
3. $x_2 = x_1 - \frac{df(x)}{dx_1} * \lambda$, λ is called learning rate
4. Find $\frac{df(x)}{dx_2}$
5. $x_3 = x_2 - \frac{df(x)}{dx_2} * \lambda$ …. and so on

Gradient Descent Algorithm

1. Choose an initial value x_1
2. For a given x_k find x_{k+1} as $x_{k+1} = x_k - \frac{df(x)}{dx_k} * \lambda$

λ is called **learning rate** or a **step size**

- Depends on data
- Involves some trial and error
- Generally good idea to begin with default value (if given!)
- Experience counts

GDA Multidimensional Cost Function

Cost function: $J(\boldsymbol{w})$

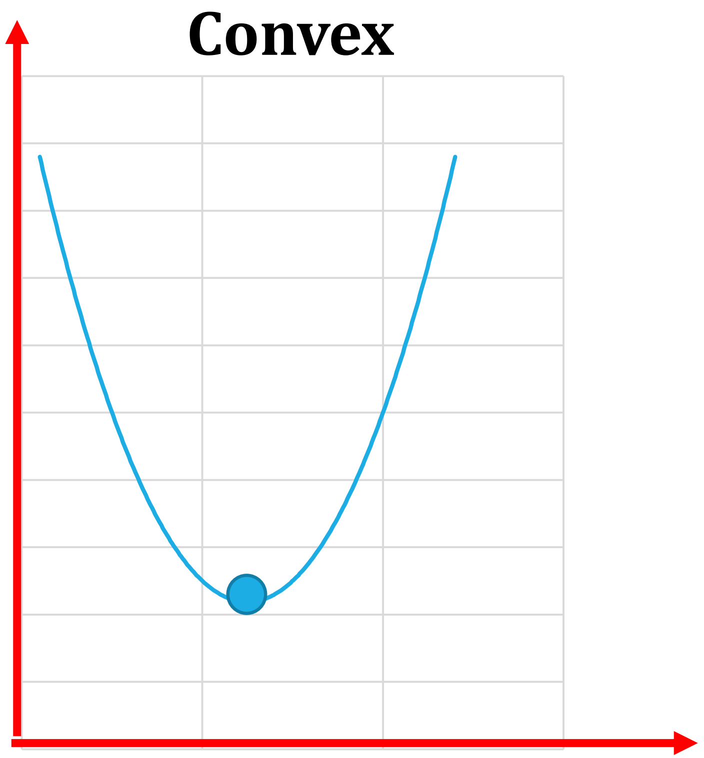
Initiate with $\boldsymbol{w}^1$

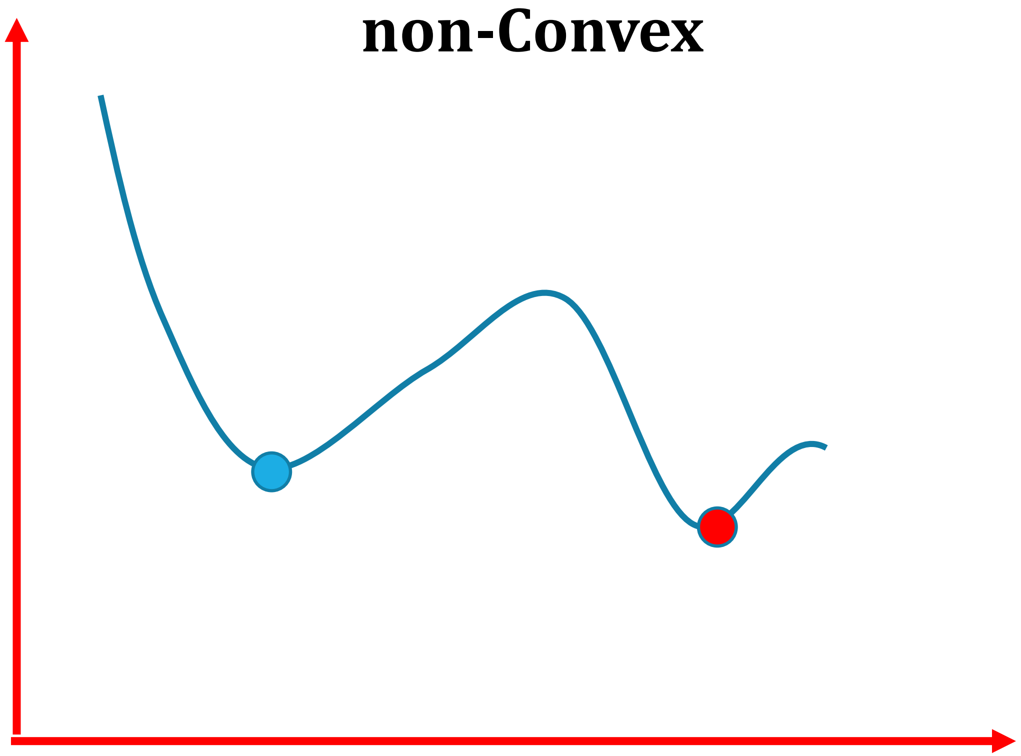
$\boldsymbol{w}^{\boldsymbol{k+1}} \leftarrow \boldsymbol{w}^{\boldsymbol{k}} - \lambda \frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} |_{\boldsymbol{w}=\boldsymbol{w}^{k}}$ for k =1, 2,…

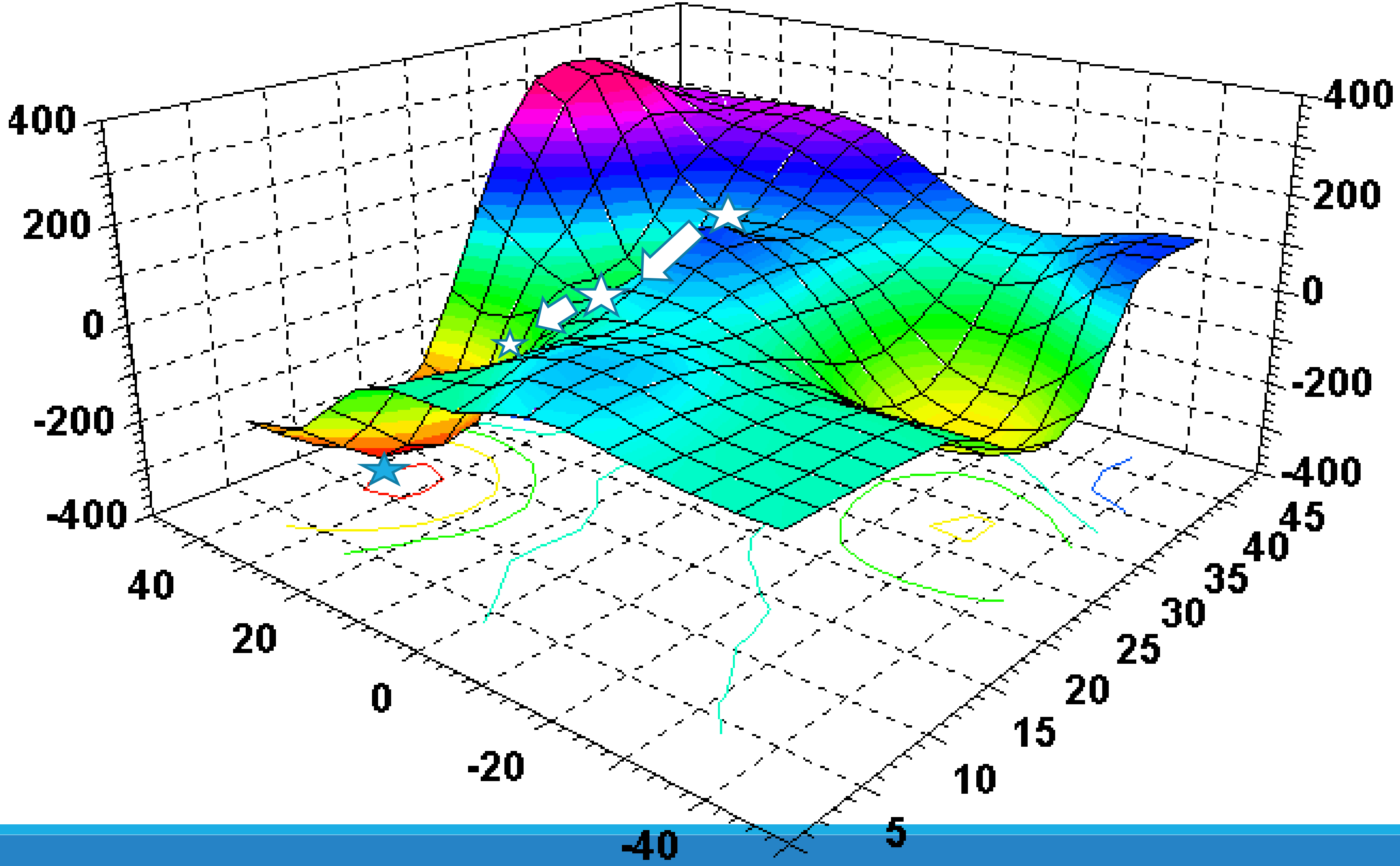
Continue till it converges or reaches maximum iterations

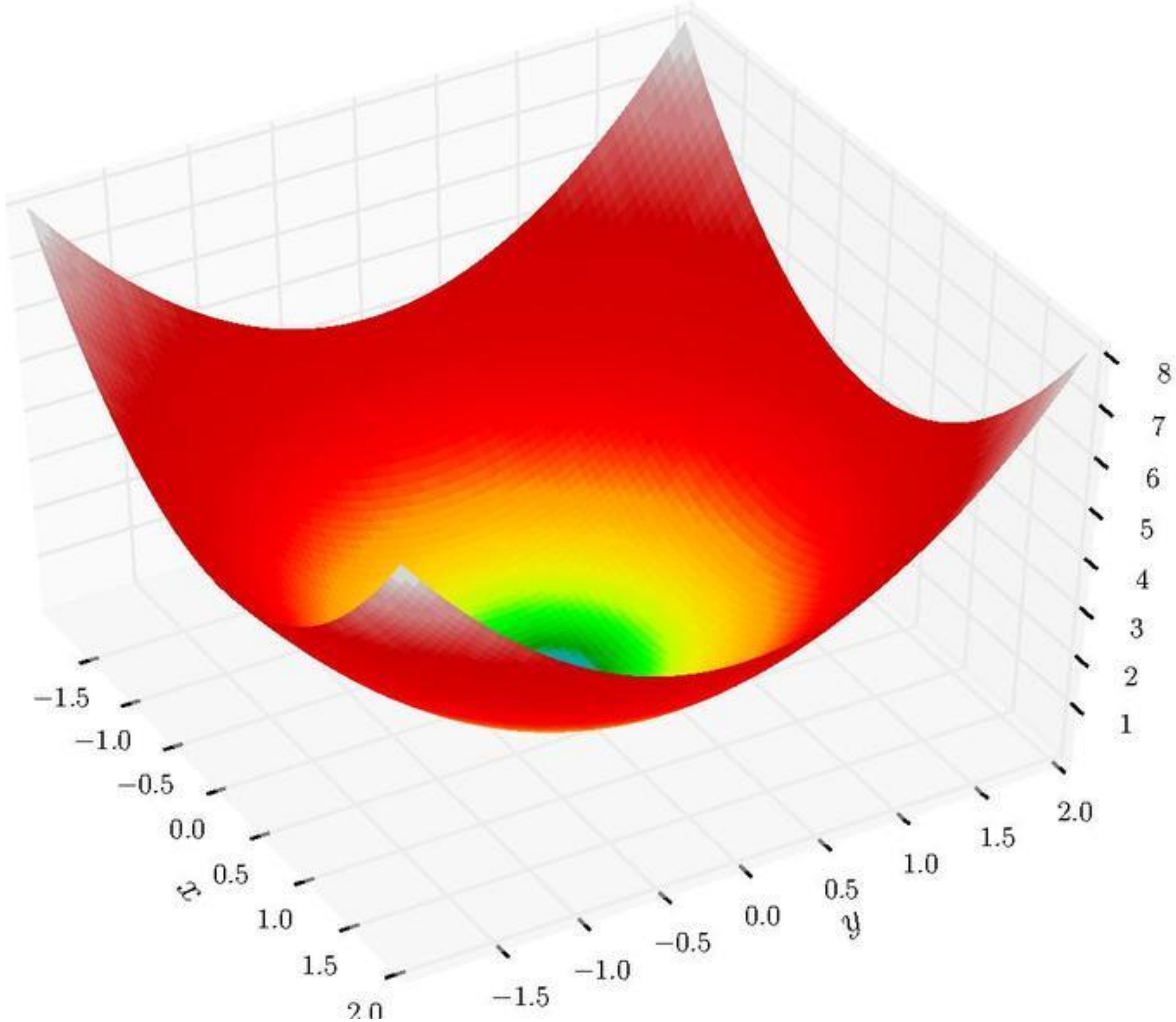
Does it work?

Gradient descent algorithm converges to global optimum when the function is convex






400
200
0
-200
-400
400
200
0
-200
-400
40
20
0
-20
-40
5
10
15
20
25
30
35
40
45



−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
y
1
2
3
4
5
6
7
8

Good news!

LSE function for multiple regression is a convex function:

$$\boldsymbol{SSE} = \sum \left(y^{(i)} - \boldsymbol{w} \cdot \boldsymbol{x}^{(\boldsymbol{i})}\right)^2$$

Cross entropy error function is also convex

$$\mathcal{E}(\mathbf{w}) = -\sum y^{(i)} \log\left(\psi\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(\boldsymbol{i})}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - \psi\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(\boldsymbol{i})}\right)\right)$$

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

Regression

$$\boldsymbol{SSE} = \sum \left(y^{(i)} - \boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)^2 = \left[\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{w} - 2\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}\right)^{\boldsymbol{T}}\boldsymbol{w}\right] + const$$

Initialize with $\boldsymbol{w}_1$ at random

set t = 1 and choose λ

continue until convergence

1. compute the gradient $\nabla \boldsymbol{SSE} = \boldsymbol{X}^{\boldsymbol{T}}(\boldsymbol{X}\boldsymbol{w}_{\boldsymbol{t}} - \boldsymbol{y})$
2. update $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \lambda \nabla \boldsymbol{SSE}$
3. $t \leftarrow t + 1$

Logistic Regression

$$\boldsymbol{\mathcal{E}}(\mathbf{w}) = -\sum y^{(i)} \log\left(\psi\left(\boldsymbol{w}\cdot\boldsymbol{x}^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - \psi\left(\boldsymbol{w}\cdot\boldsymbol{x}^{(\boldsymbol{i})}\right)\right)$$

$\psi(z) = \frac{1}{1+e^{-z}}$ then $\frac{d\psi(z)}{dz} = \psi(z)(1 - \psi(z))$

It can be shown that

$$\frac{\partial\boldsymbol{\mathcal{E}}(\mathbf{w})}{\partial\boldsymbol{w}} = \sum_i \left[\psi\left(\boldsymbol{w}\cdot\boldsymbol{x}^{(\boldsymbol{i})}\right) - y^{(i)}\right]\boldsymbol{x}^{(\boldsymbol{i})}$$

Algorithm for Logistic Regression

1. Choose λ
2. Initiate $\boldsymbol{w}_1$
3. Iterate as $\boldsymbol{w}_{\boldsymbol{t+1}} = \boldsymbol{w}_t - \lambda \sum_i \left[\psi\left(\boldsymbol{w}_t \cdot \boldsymbol{x}^{(\boldsymbol{i})}\right) - y^{(i)}\right]\boldsymbol{x}^{(\boldsymbol{i})}$
4. Iterate until convergence

Choosing λ in practice

Try out λ = 0.001, 0.01, 0.1 on a test data set. Choose the λ that gives stable and fast convergence.

To reach true minimum, reduce λ by factor of 10 as learning saturates.

Validation

Validation with test data

Model estimation using training data

Apply the same model to test data and see if the results are “same”!

What is meant by “same”?

Misclassification Quantification

Total data size = P + N		Predicted Condition	
		Positive(PP)	Negative (PN)
Actual Condition	Positive (P)	**True Positive (TP)**	**False Negative (FN)**
	Negative (N)	**False Positive (FP)**	**True Negative (TN)**

$$\text{TPR} = \text{True Positive Rate} = \frac{\mathbf{TP}}{\mathbf{TP+FN}}$$ = Sensitivity / Hit Rate / Recall

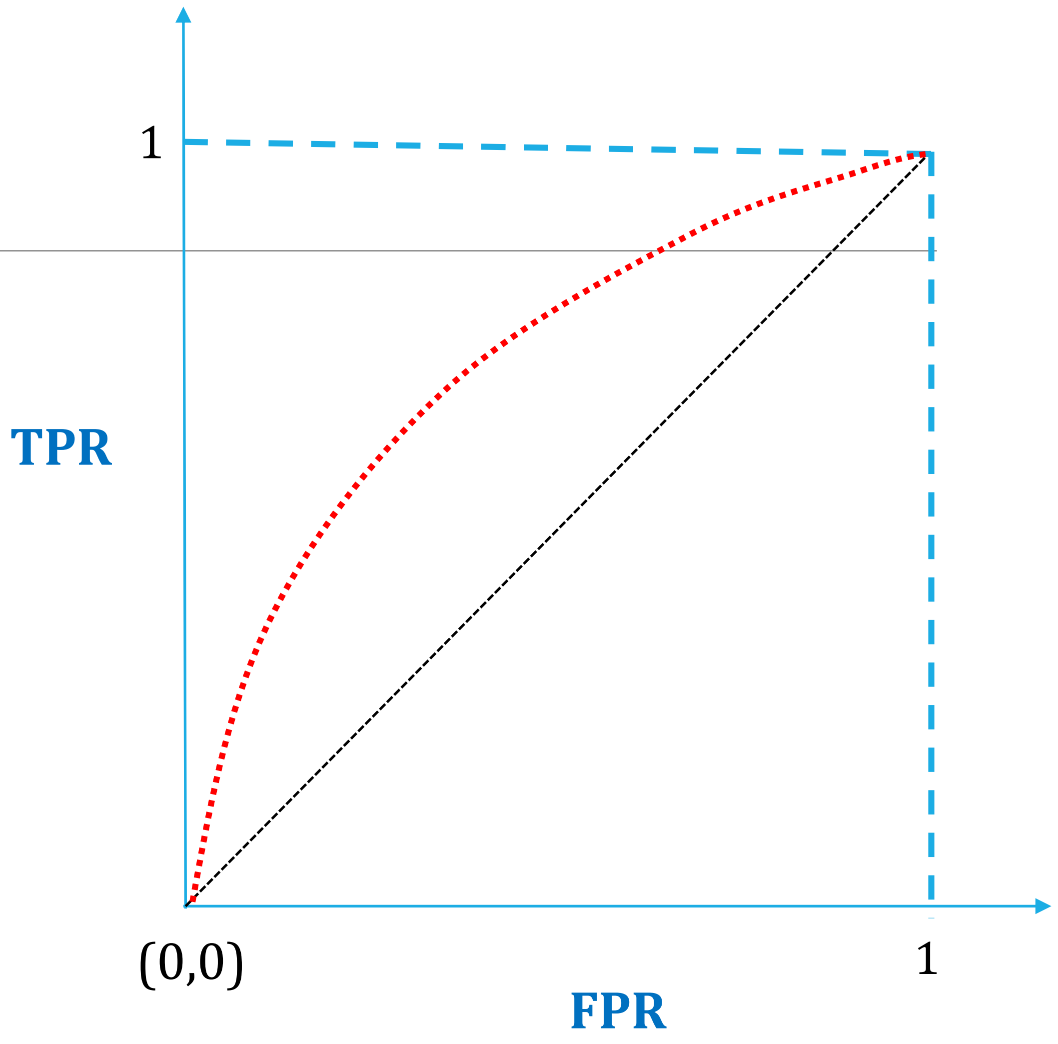
$$\text{FPR} = \text{False Positive Rate} = \frac{\mathbf{FP}}{\mathbf{FP+TN}}$$ = Probability of false alarm = 1- specificity

ROC Curve & AUC

ROC Curve = Receiving Operating Characteristic Curve

ROC is a plot of FPR vs. TPR calculated at different threshold values.

Area Under the ROC Curve is called AUC

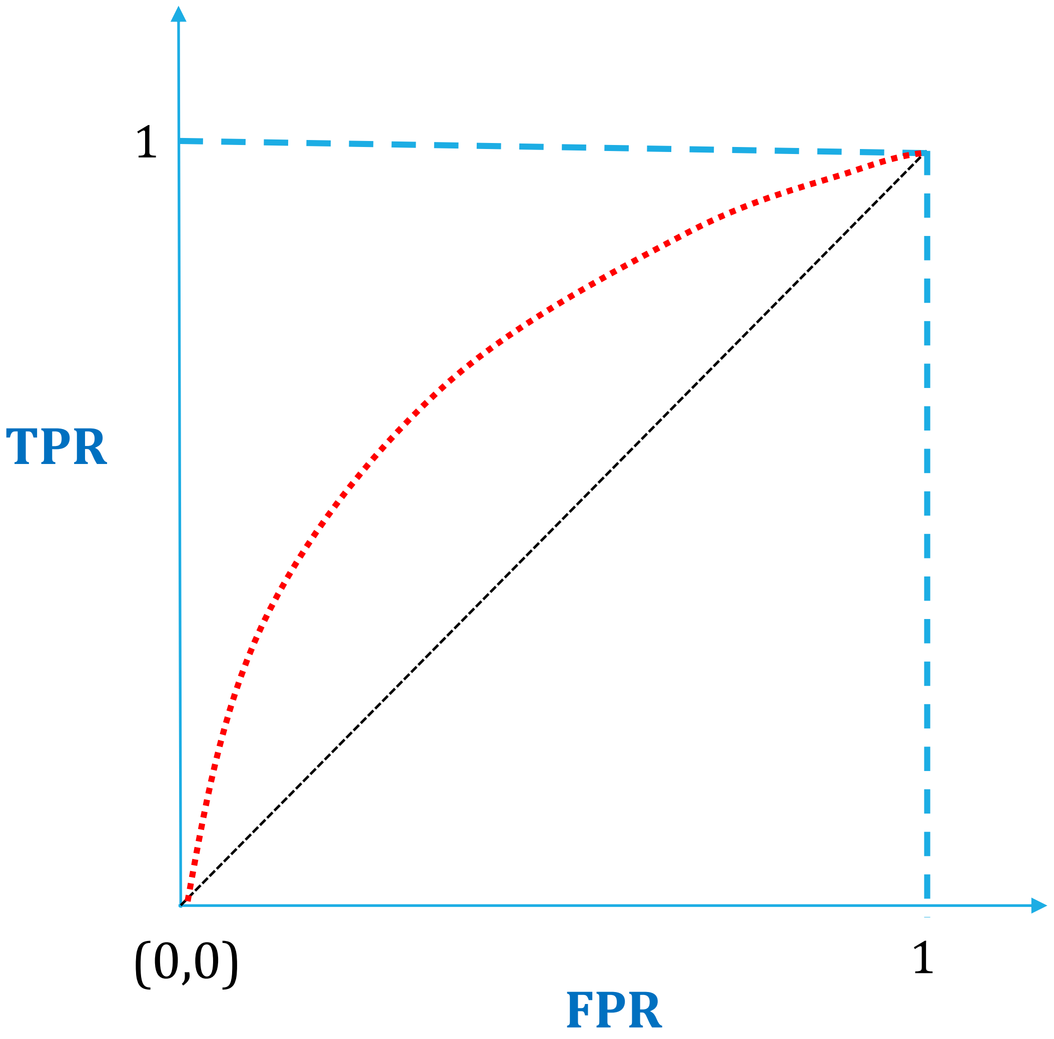


ROC Curve & AUC

ROC Curve = Receiving Operating Characteristic Curve

ROC is a plot of FPR vs. TPR calculated at different threshold values.

Area Under the ROC Curve is called AUC



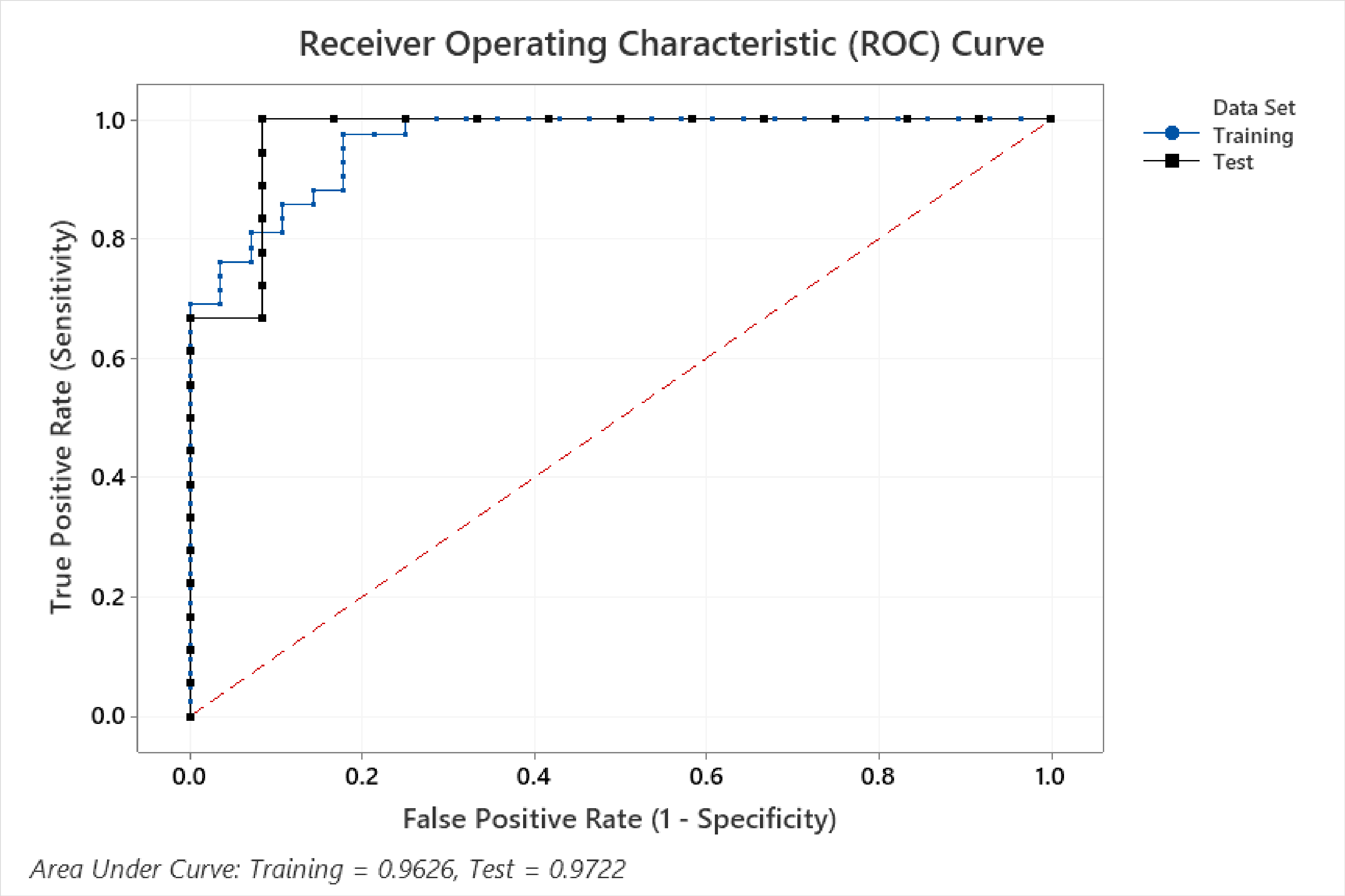
What is a good fit?

ROC is above the FPR = TPR line

AOC is more than 50%

ROC and AOC for Test data is close to that of Training data.

Large data set - Validation



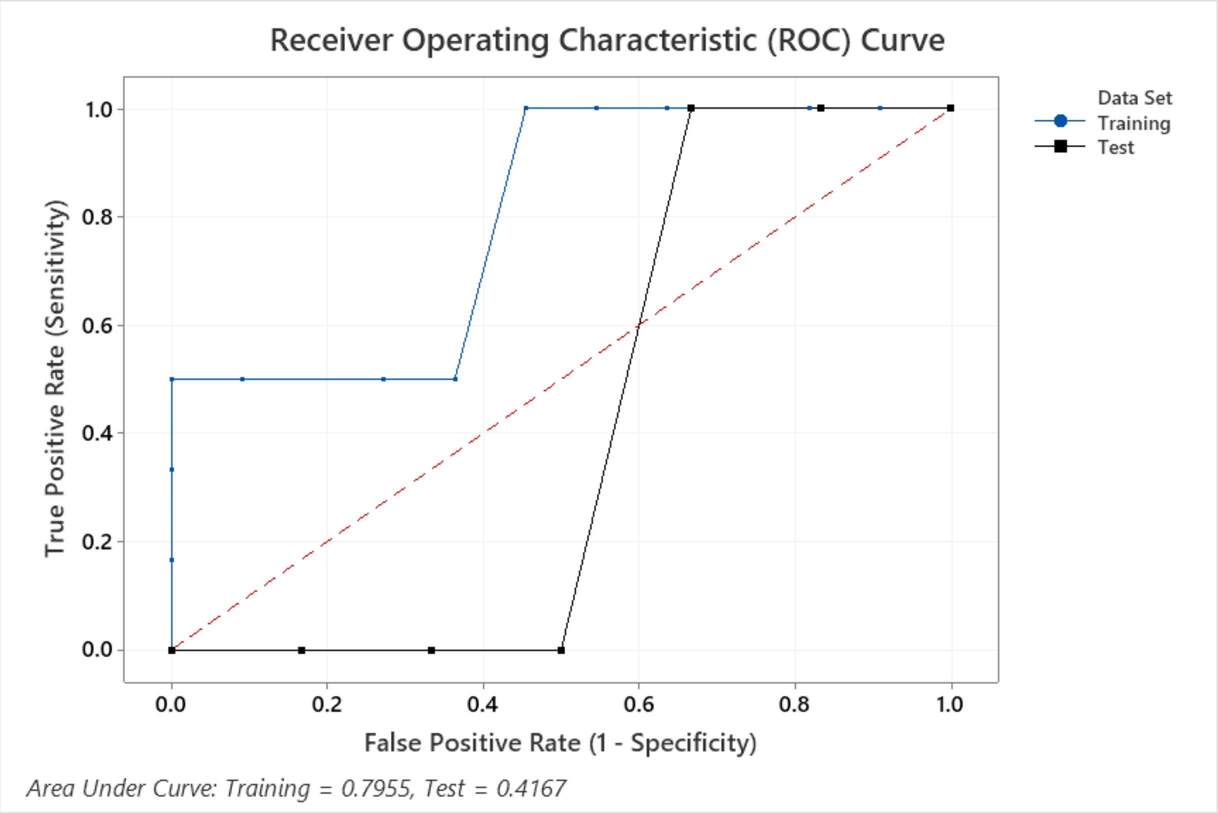
Validation

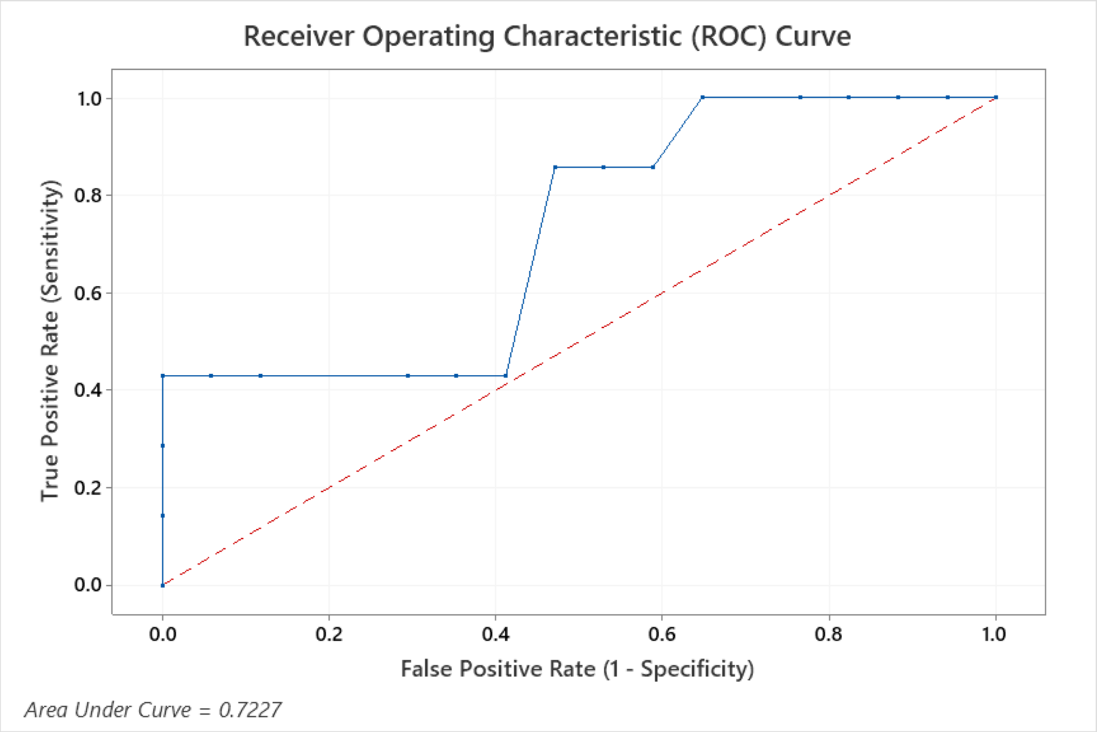
If ROC and AUC for Training Data and Test data are close then the model is validated.

If not valid:

- Too many features? Reduce the number of features.
- Correlated features? Keep only uncorrelated features.
- Training and test data sets chosen randomly? If not, correct it.
- Is data too small? Then avoid dividing the data. Keep only training data for estimation and make sure that the model fits well.

Small data – Caution!





Summary

Data preparation

Logistic Regression for classification

Method of gradient descent

- GD for multiple regression
- GD for logistic regression

Validation methods

Thank you…