INTRODUCTION TO NEURAL NETWORKS

Lecture 2. Linear Regression & Logistic Regression

Dr. Jingxin Liu

School of AI and Advanced Computing

Xi'an Jiaotong-Liverpool University
西交利物浦大学

25.02.2022

Table of Contents & Learning Goals

Table of Contents:

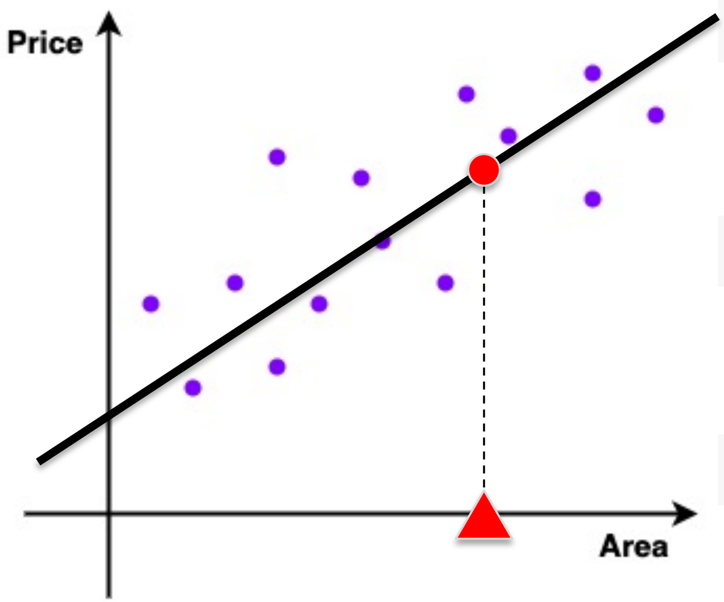
Learning Goals:

- Formulate a machine learning task mathematically
- Derive both the closed-form solution and the gradient descent updates for linear regression and logistic regression
- Learn some common terms in machine learning glossary

Linear Regression – Flat price prediction

Flat price prediction - regression problem

Living Area (m^2)	Price (*M Yuan*)
120	5.4
95	4.1
143	7.1
78	3.9
124	5.9
131	6.3
…	…



Linear Regression – Flat price prediction

Living Area (m^2)	Price (*M Yuan*)
120	5.4
95	4.1
143	7.1
78	3.9
124	5.9
131	6.3
…	…

Training Set

Input
Feature

Target Output
Target / Label / Groundtruth

Linear Regression – Notations

Living Area (m^2)	Price (*M Yuan*)
120	5.4
95	4.1
143	7.1
78	3.9
124	5.9
131	6.3
…	…

m

Notations:

m = number of training examples

x = input variables / feature

y = output variables / target variables

(x, y) one pair of training example

(x^i, y^i) *i* th training example

$X = \{x^1, x^2, \dots, x^i, \dots, x^m\}$ Feature set of the training set

$Y = \{y^1, y^2, \dots, y^i, \dots, y^m\}$ Target set of the training set

Linear Regression – Notations

Living Area (m^2)	# Bedroom	Price (*M Yuan*)
120	3	5.4
95	2	4.1
143	4	7.1
78	2	3.9
124	4	5.9
131	3	6.3
…		…

1st dim *2nd dim*

Notations(cont.):

x_1 = 1st dim of X / 1st feature

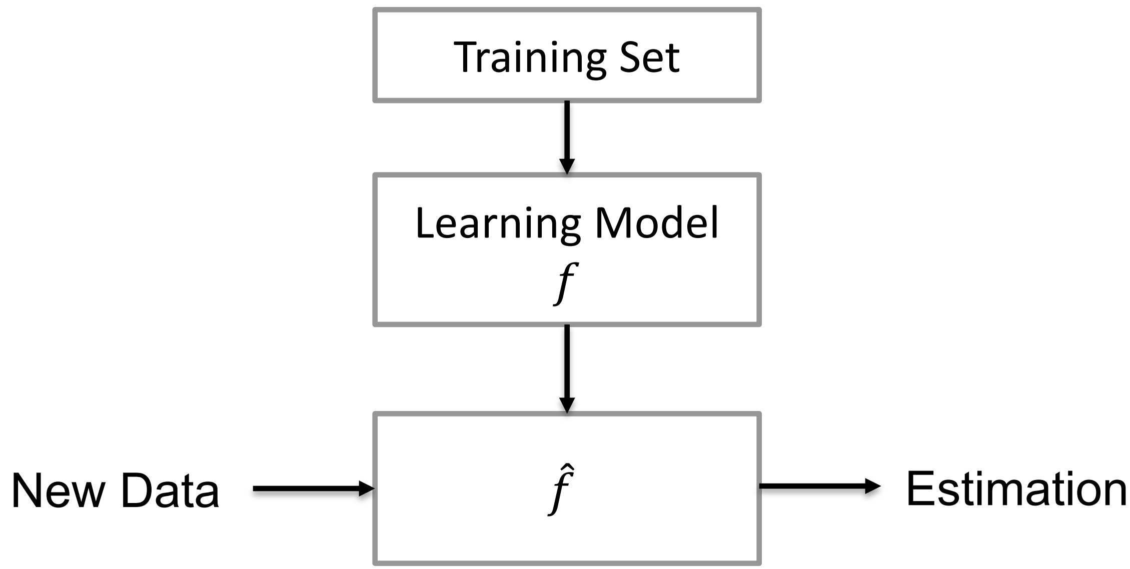
x_j = j th dim of X / j th feature

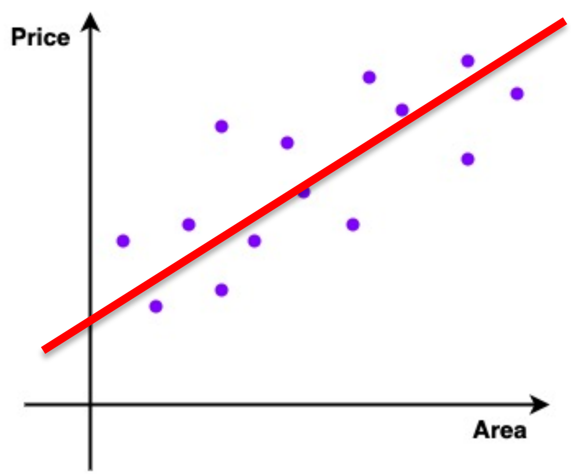
n = number of dimensions/features

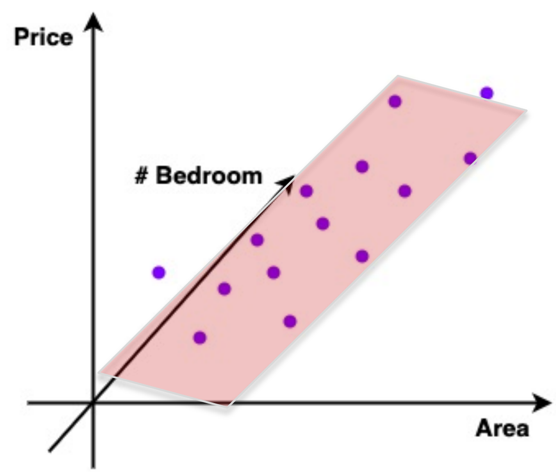
$$X = \begin{bmatrix} \boxed{x_1^1 & \cdots & x_n^1} \\ \vdots & \ddots & \vdots \\ x_1^m & \cdots & x_n^m \end{bmatrix}$$

1st example with n dim

Linear Regression – Learning

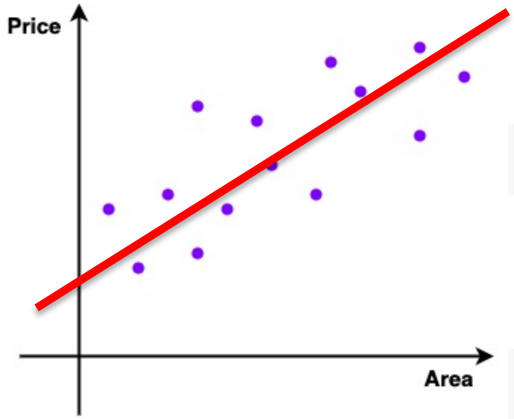






Linear Regression – Learning Algorithm

The linear mapping function / hypothesis / model / learning algorithm can be represented as



$$f(x) = \theta_0 + \theta_1 x_1$$

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \hat{y}$$

$$f(x) = f_\theta(x) = \sum_{j=0}^{n} \theta_j x_j = \hat{y} \qquad (x_0 = 1)$$

θ are parameters of learning algorithms

$\hat{y}$ is the predicted result of $f(x)$

The job of training is to use the training set to choose or learn appropriate parameters θ of learning algorithms.

Linear Regression – Loss Function

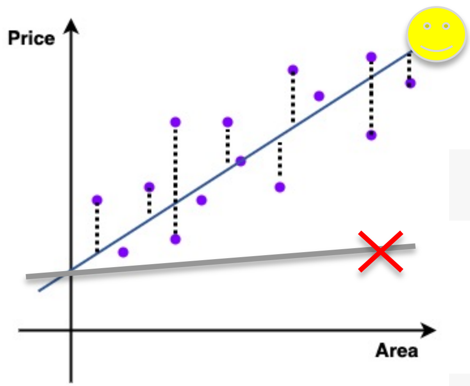
Some of these linear fits are better than others. In order to quantify how good the fit is, we define a **loss function.**

$$\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(f(x) - y)^2$$

Squared Error

Predict value

True value



The best model with respect to θ should have the minimum sum of $\mathcal{L}$ on the training set.

When we combine *our model* $f(x)$ and *loss function* $\mathcal{L}$, we get an **optimization problem**.

Linear Regression – Cost Function

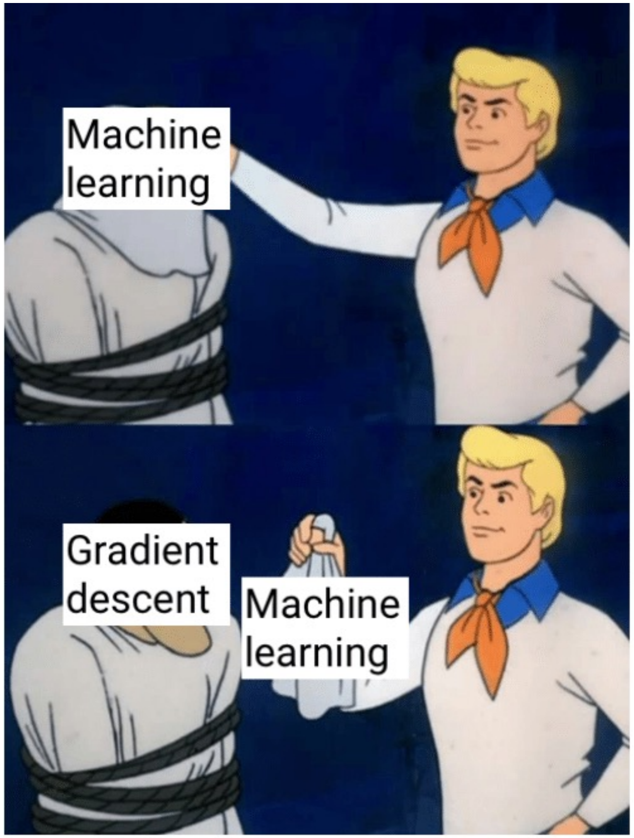
To solve the optimization problem, we try to minimize a **cost function** with respect to the model parameters θ .

For linear regression, the cost function is simply the loss, averaged over all the training examples (MSE, Mean Squared Error).

$$\mathcal{J}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(f\left(x^{i}\right) - y^{i})^{2}$$

Note the difference between the loss function and the cost function. The loss is a function of the predictions and targets, while the cost is a function of the model parameters.

Linear Regression – Gradient Descent



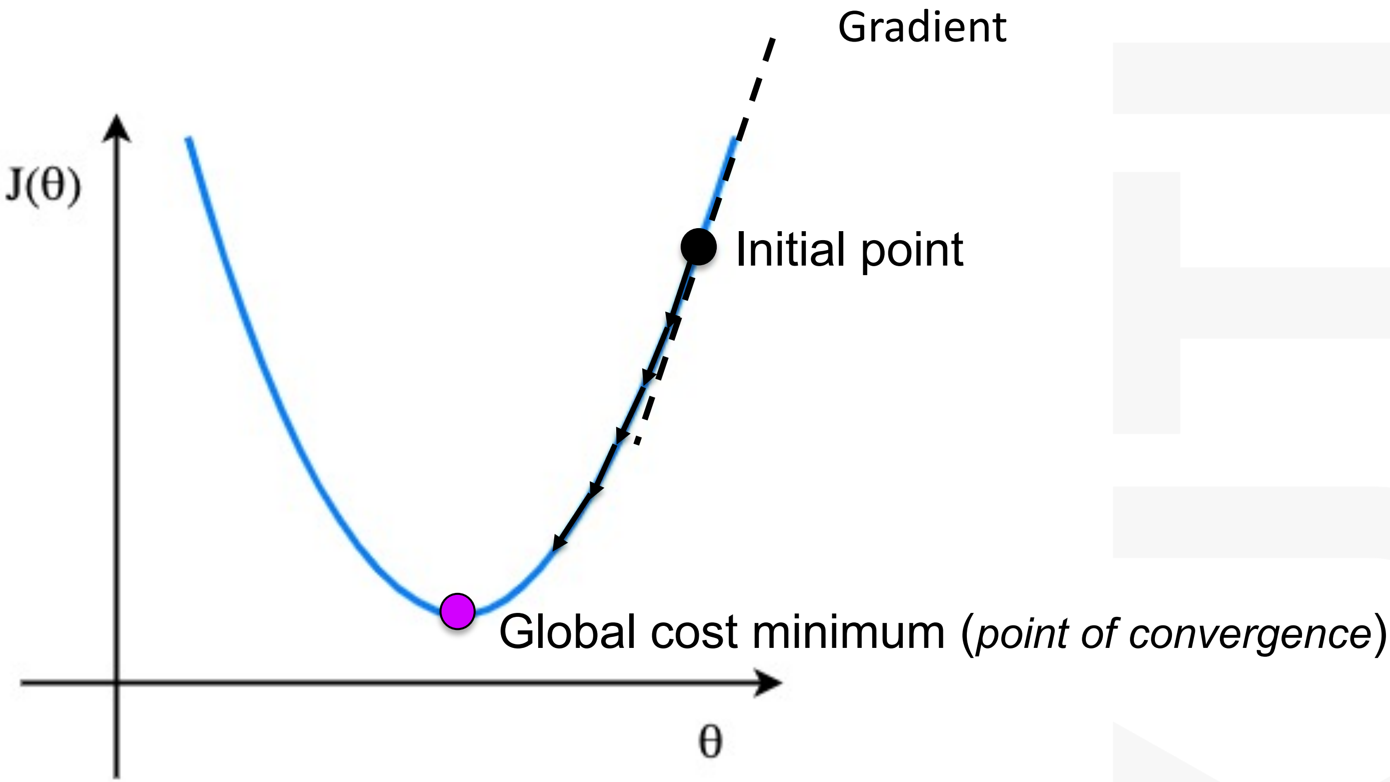
Machine learning behind the scenes

Source: https://me.me/i/machine-learning-gradient-descent-machine-learning-machine-learning-behind-the-ea8fe9fc64054eda89232d7ffc9ba60e

Linear Regression – Gradient Descent

$$\mathcal{J}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(f(x^i) - y^i)^2$$

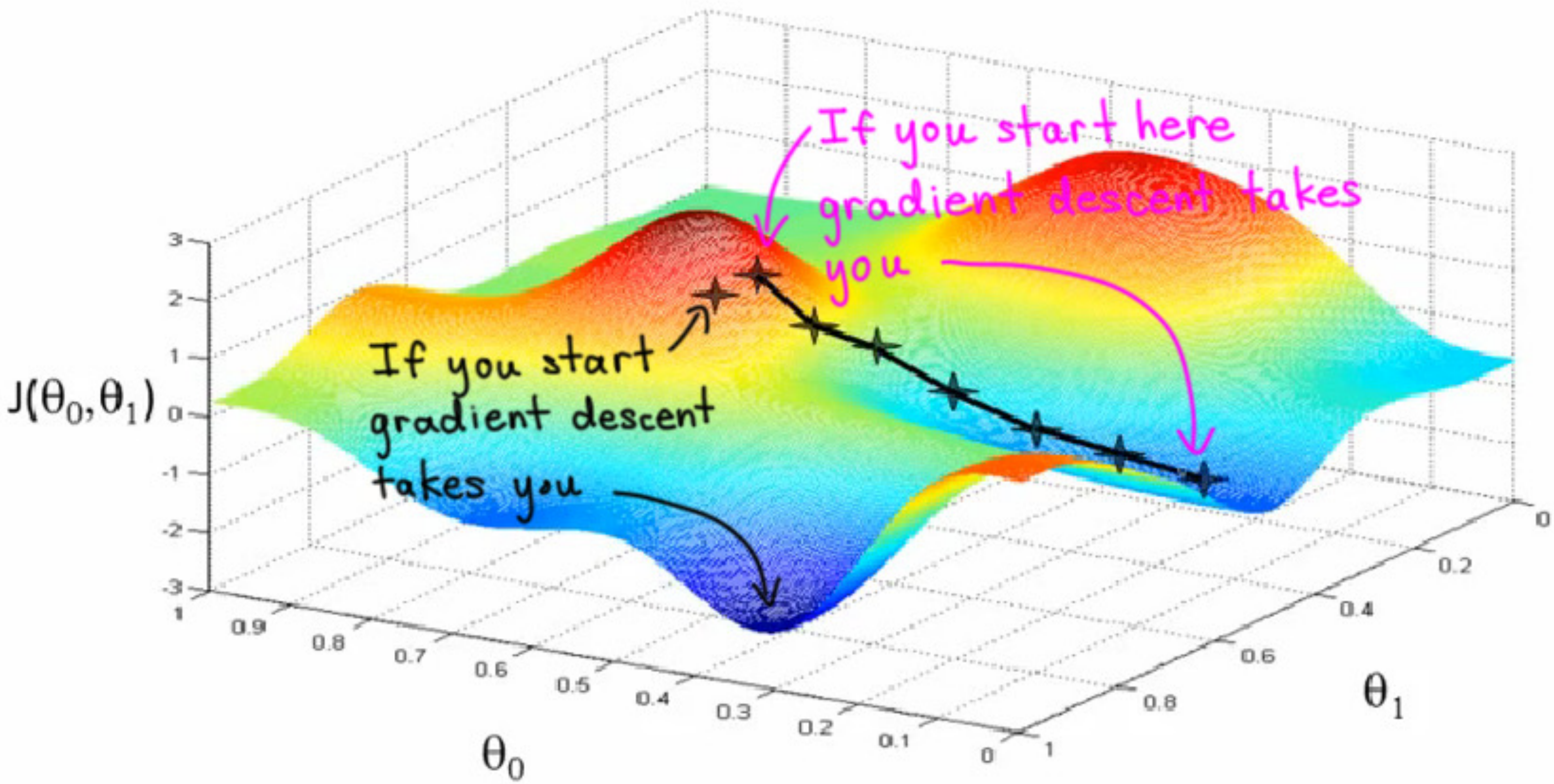
second-order equation



To get the gradient / slope, we take the derivative of cost function at θ.

Linear Regression – Gradient Descent

When a function is multivariate, we use partial derivatives to get the slope of a function at a given point.



Linear Regression – Gradient Descent

In order to do gradient descent, we require two data points:

a direction -> partial derivative
a learning rate -> α (alpha) *(set by yourself)*

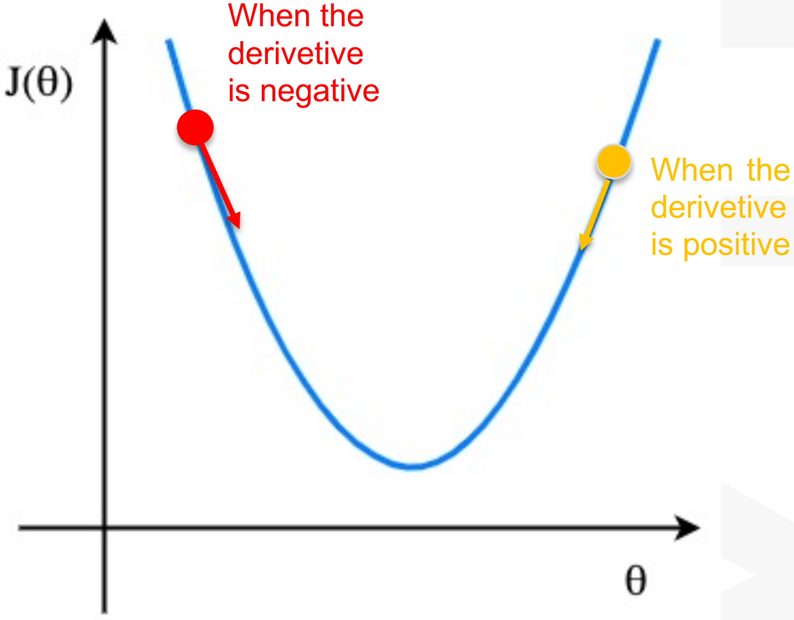
Mathematically the formula of gradient descent is :

Repeat until convergence {

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathcal{J}(\theta)$$

}

Learning rate



Linear Regression – Gradient Descent

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathcal{J}(\theta)$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathcal{J}(\theta) \longrightarrow \begin{cases} \theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} \mathcal{J}(\theta) \\ \theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} \mathcal{J}(\theta) \end{cases}$$

$$\frac{\partial}{\partial \theta_j} \mathcal{J}(\theta) = \frac{\partial}{\partial \theta_j} \left(\frac{1}{2m} \sum_{i=1}^{m} (f(x_i) - y_i)^2\right)$$

$$\frac{\partial}{\partial \theta_j} \mathcal{J}(\theta) = \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - y_i) \cdot \frac{\partial}{\partial \theta_j} (f(x_i) - y_i)$$

$$\frac{\partial}{\partial \theta_j} \mathcal{J}(\theta) = \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - y_i) \cdot \frac{\partial}{\partial \theta_j} [(\theta_0 x_0 + \cdots + \theta_j x_j + \cdots + \theta_n x_n) - y_i]$$

$$\frac{\partial}{\partial \theta_j} \mathcal{J}(\theta) = \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - y_i) \cdot x_i$$

Linear Regression – Gradient Descent

$$\frac{\partial}{\partial \theta_j} \mathcal{J}(\theta) = \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - y_i) \cdot x_i$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathcal{J}(\theta)$$

Therefore,

$$\theta_j := \theta_j - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - y_i) \cdot x_i$$

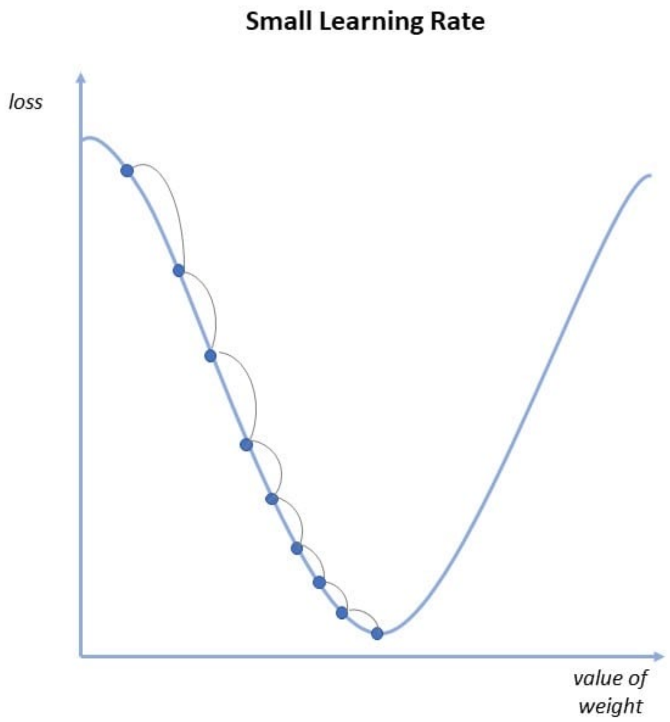
So consequently,

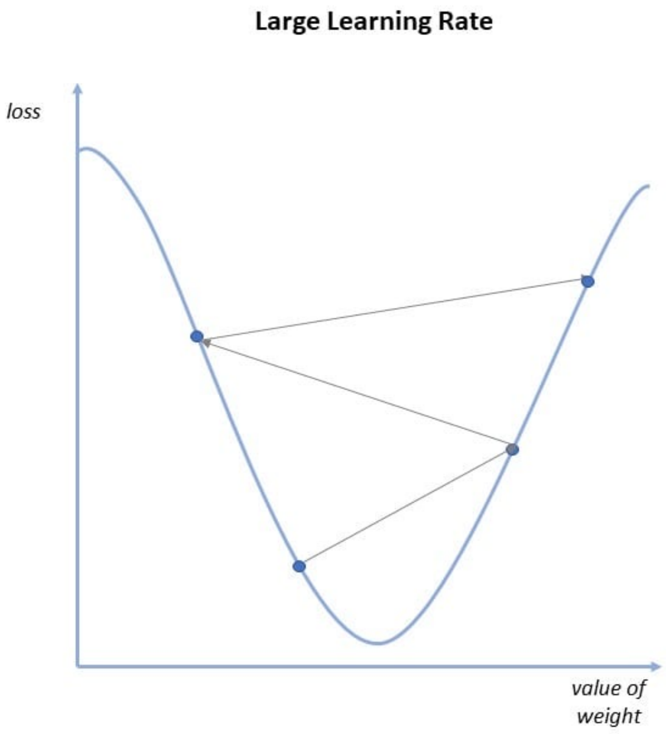
Repeat until convergence {

$$\theta_j := \theta_j - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - y_i) \cdot x_i$$

}

Linear Regression – Gradient Descent





Linear Regression – Batch GD VS SGD

Batch Gradient Descent:

Batch Gradient Descent involves calculations over the full training set at each step as a result of which it is very slow on very large training data.

Stochastic Gradient Descent (SGD):

SGD is stochastic in nature i.e it picks up a “random” instance of training data at each step and then computes the gradient making it much faster as there is much fewer data to manipulate at a single time, unlike Batch GD.

Logistic Regression – Notations

Living Area (m^2)	# Bedroom	Luxury
120	3	Yes
95	2	No
143	4	No
78	2	Yes
124	4	No
131	3	Yes
…		…

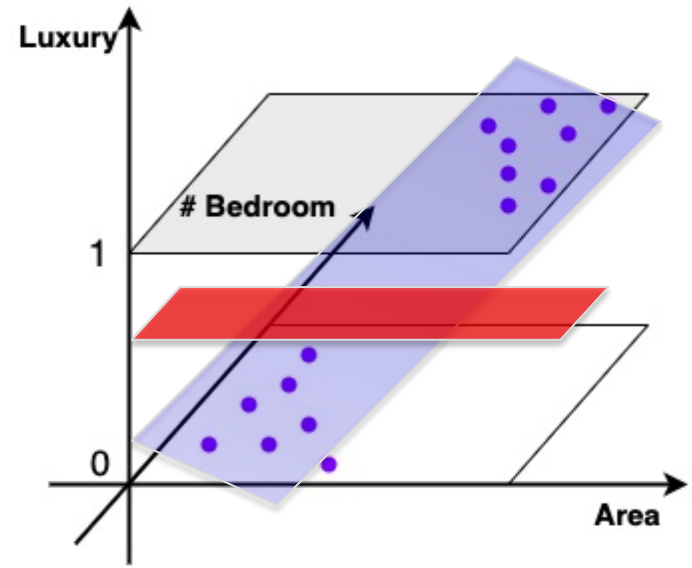
Notations:

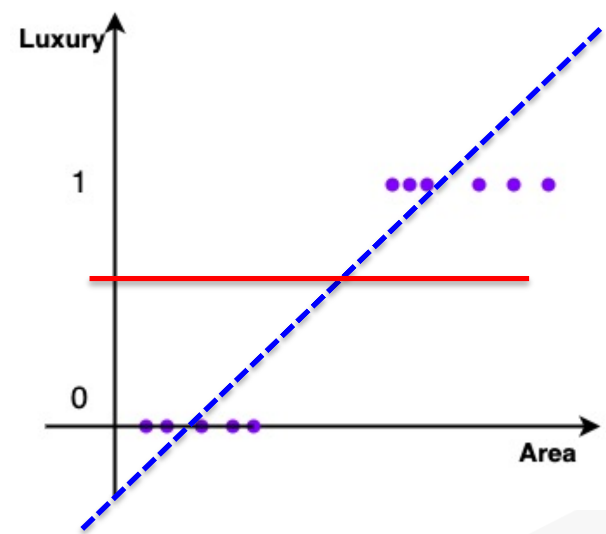
$$X = \begin{bmatrix} x_1^1 & \cdots & x_n^1 \\ \vdots & \ddots & \vdots \\ x_1^m & \cdots & x_n^m \end{bmatrix}$$

$$Y = \{0, 1\} \begin{cases} 0 : \text{'Negative Class'} \\ 1 : \text{'Positive Class'} \end{cases}$$

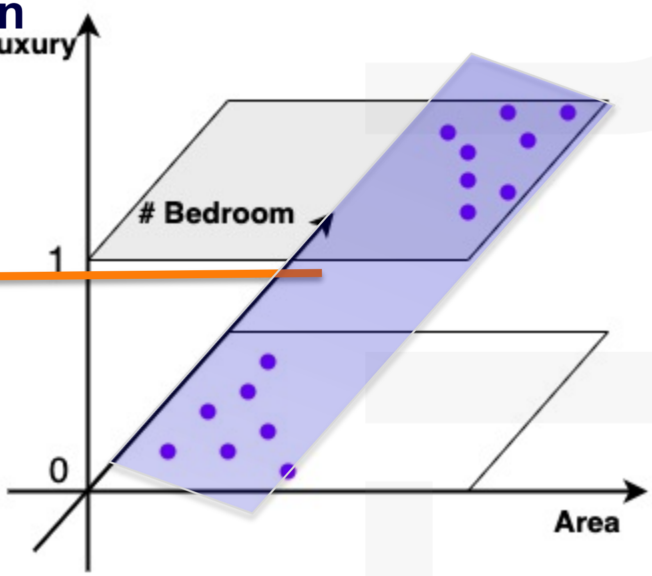
Logistic Regression

Living Area (m^2)	# Bedroom	Luxury
120	3	Yes
95	2	No
143	4	No
78	2	Yes
124	4	No
131	3	Yes
…		…





Logistic Regression – Model Description



For linear regression

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

The idea in logistic regression is to cast the problem in form of generalized linear regression model.

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$ unchanged

$\hat{y} \in (-\infty, +\infty)$ ---> $\hat{y} \in \{0, 1\}$

Predict the probability that $y = 1$ $\quad p \in [0,1]$ $\quad Threshold = 0.5$

$$y = \begin{cases} 0 & if\ p < 0.5 \\ 1 & if\ p \geq 0.5 \end{cases}$$

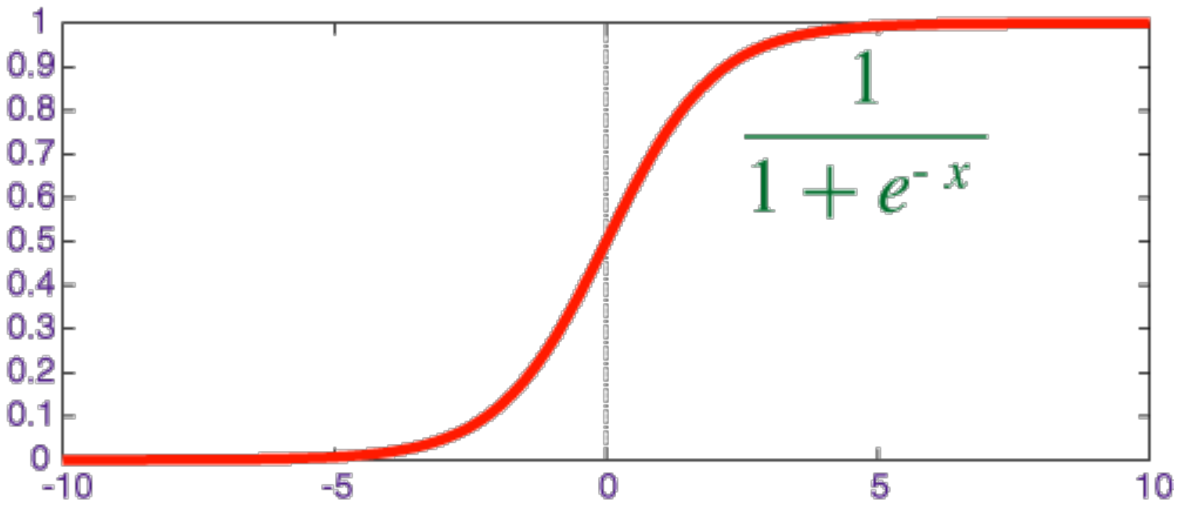
Logistic Regression – Model Description

So, instead of predict $\hat{y}$, we need to predict the probability p

But the predicted output may $< 0 \; or > 1$

Squeezes the output from $(-\infty, +\infty)$ to $[0,1]$.

Sigmoid (logistic) function

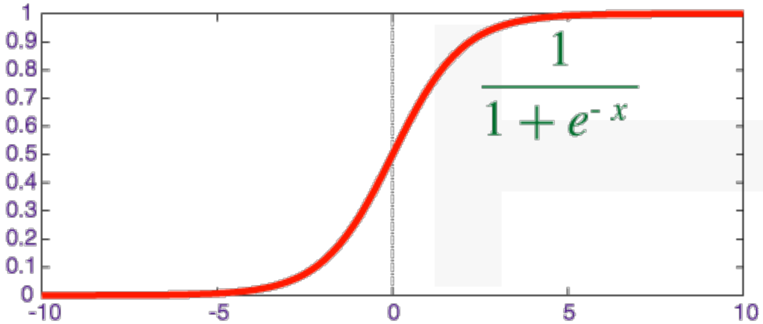


Logistic Regression – Model Description

$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ mapping function in linear regression

$h(x) = sigmoid(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n)$ mapping function in logistic regression

where $sigmoid(z) = \dfrac{1}{1 + e^{-z}}$



$h(x) = h_\theta(x) = \dfrac{1}{1 + e^{-\sum_0^n \theta_j x_j}}$ $\quad 0 \le h(x) \le 1$

$h(x)$ estimated probability that $y = 1$ on input x

Logistic Regression – Model Description

$h(x)$ estimated probability that $y = 1$ on input x

$h(x) = P(y = 1|x; \theta)$ Probability that $y = 1$, given x, parameterized by θ

$$P(y = 1|x; \theta) + P(y = 0|x; \theta) = 1$$

$$P(y = 0|x; \theta) = 1 - P(y = 1|x; \theta)$$

So consequently,

$h(x)$ is the probability '$y = 1$'
$1 - h(x)$ is the probability '$y = 0$'

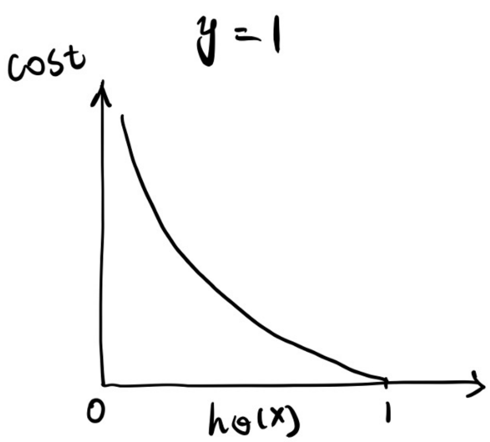
Logistic Regression – Model Description

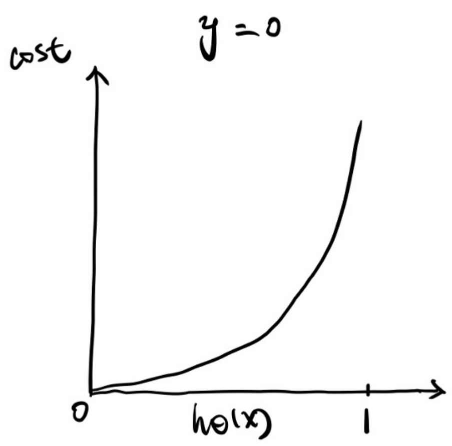
$\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(f(x) - y)^2$ Loss function for linear regression

We want to assign more punishment when predicting 1 while the actual is 0 and when predict 0 while the actual is 1. The loss function of logistic regression is doing this exactly which is called **Logistic Loss**.

$$\mathcal{L} = \begin{cases} -\log(h(x)) & if\ y = 1 \\ -\log(1 - h(x)) & if\ y = 0 \end{cases}$$

Loss function for logistic regression





Logistic Regression – Lost Function & Cost Function

$$\mathcal{L} = \begin{cases} -\log\big(h(x)\big) & if\ y = 1 \\ -\log\big(1 - h(x)\big) & if\ y = 0 \end{cases}$$

It can be written as one single formula which brings convenience for calculation:

$$\mathcal{L} = -y \cdot \log\big(h(x)\big) - (1 - y)\log(1 - h(x))$$

So the cost function of the model is the summation from all training data samples:

$$\mathcal{J}(\theta) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m}[y^i \cdot \log\left(h\left(x^i\right)\right) + \left(1 - y^i\right) \cdot log(1 - h(x^i))]$$

Logistic Regression – Batch GD for Logistic Regression

$$\min_{\theta} \mathcal{J}(\theta)$$

Again, we use gradient descent for optimization.

Repeat until convergence {

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathcal{J}(\theta)$$

}

Repeat until convergence {

$$\theta_j \coloneqq \theta_j - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i) \cdot x_i$$

}

Surprise!

Logistic Regression

$$z=\sum_{j=1}^{n}\theta_j x_i \rightarrow h(x)=\frac{1}{1+e^{-z}} \rightarrow \mathcal{L}=-y\cdot\log(h(x))-(1-y)\log(1-h(x)) \rightarrow J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\cdot\mathcal{L}$$

we need to find $\frac{\partial J(\theta)}{\partial\theta}$, Let $A=-y\cdot\log(h(x))$, $B=-(1-y)\cdot\log(1-h(x))$

To find $\frac{\partial A}{\partial\theta}$,

$$\frac{\partial A}{\partial\theta}=\frac{\partial A}{\partial h(x)}\cdot\frac{\partial h(x)}{\partial z}\cdot\frac{\partial z}{\partial\theta}$$

$$=-\frac{y}{h(x)}\cdot\frac{\partial h(x)}{\partial z}\cdot\frac{\partial z}{\partial\theta}$$

$$=-\frac{y}{h(x)}\cdot h(x)\cdot(1-h(x))\cdot x$$

$$=-yx(1-h(x))$$

To find $\frac{\partial B}{\partial\theta}$,

$$\frac{\partial B}{\partial\theta}=\frac{\partial B}{\partial h(x)}\cdot\frac{\partial h(x)}{\partial z}\cdot\frac{\partial z}{\partial\theta}$$

$$=\frac{1-y}{1-h(x)}\cdot\frac{\partial h(x)}{\partial z}\cdot\frac{\partial z}{\partial\theta}$$

$$=\frac{1-y}{1-h(x)}\cdot h(x)\cdot(1-h(x))\cdot x$$

$$=x\,h(x)\cdot(1-y)$$

$$\frac{\partial J(\theta)}{\partial\theta}=\frac{1}{m}\sum_{i=1}^{m}\left(\frac{\partial A}{\partial\theta}+\frac{\partial B}{\partial\theta}\right)$$

$$=\frac{1}{m}\sum_{i=1}^{m}\left[-y\cdot x\cdot(1-h(x))+x\cdot h(x)\cdot(1-y)\right]$$

$$=\frac{1}{m}\sum_{i=1}^{m}\left[-yx+\cancel{y\cdot x\cdot h(x)}+x\cdot h(x)-\cancel{x\cdot y\cdot h(x)}\right]$$

$$=\frac{1}{m}\sum_{i=1}^{m}\left[x\cdot(h(x)-y)\right]$$

Note:

$$\frac{d}{dx}\ln(x)=\frac{1}{x}$$

$$\frac{d}{dx}\log_b(x)=\frac{1}{\ln(b)\cdot x}$$

$$\frac{d}{dx}\mathrm{sig}(x)=\mathrm{sig}(x)\cdot(1-\mathrm{sig}(x))$$

sig(x): sigmoid function

Conclusion

Regression

Linear Regression

$$f(x) = f_\theta(x) = \sum_{j=0}^{n} \theta_j x_j = \hat{y}$$

$$\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(f(x) - y)^2$$

$$\mathcal{J}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(f\left(x^i\right) - y^i\right)^2$$

Classification

Logistic Regression

$$h(x) = h_\theta(x) = \frac{1}{1 + e^{-\sum_0^n \theta_j x_j}}$$

$$\mathcal{L} = -y \cdot \log\big(h(x)\big) - (1 - y)\log(1 - h(x))$$

$$\mathcal{J}(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \cdot \log\left(h\left(x^i\right)\right) + \left(1 - y^i\right) \cdot log(1 - h(x^i))\right]$$

Gradient Descent

Conclusion

Supervised Learning

⬇

Classification / Regression

⬇

Choose an algorithm

↓

$$f(x) = \hat{y}$$

Loss function

Cost function

Optimizer