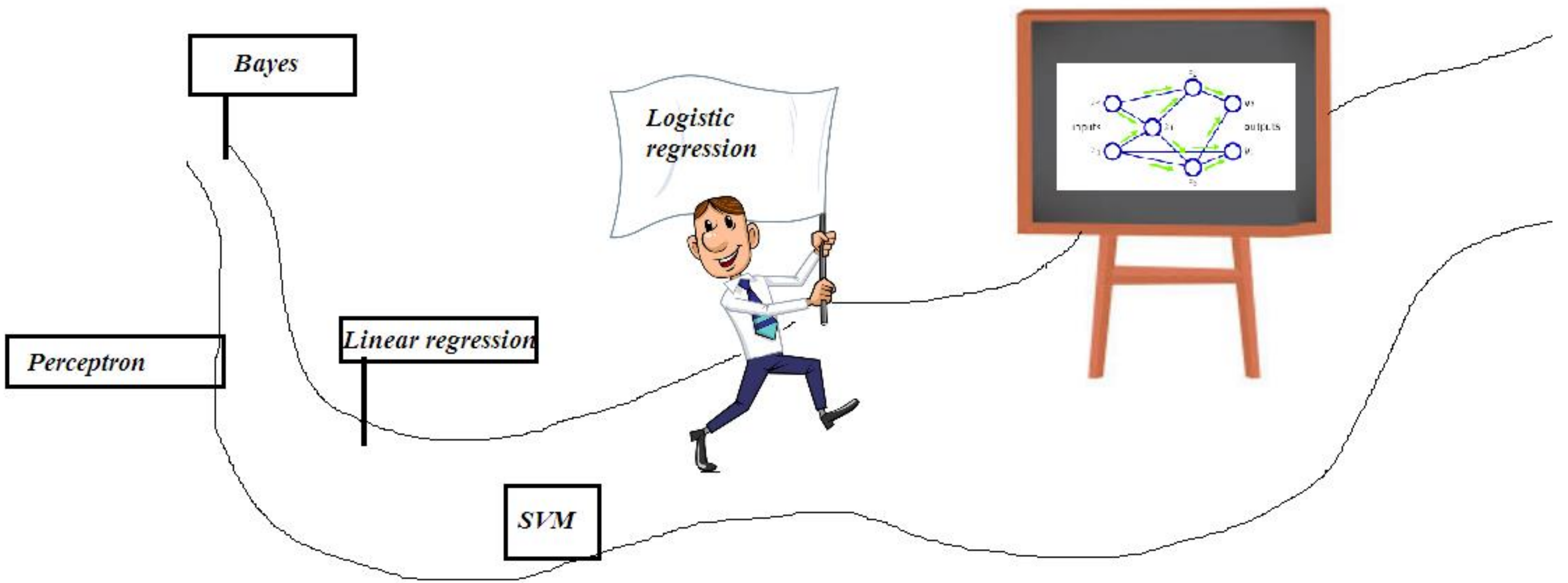
Logistic Regression

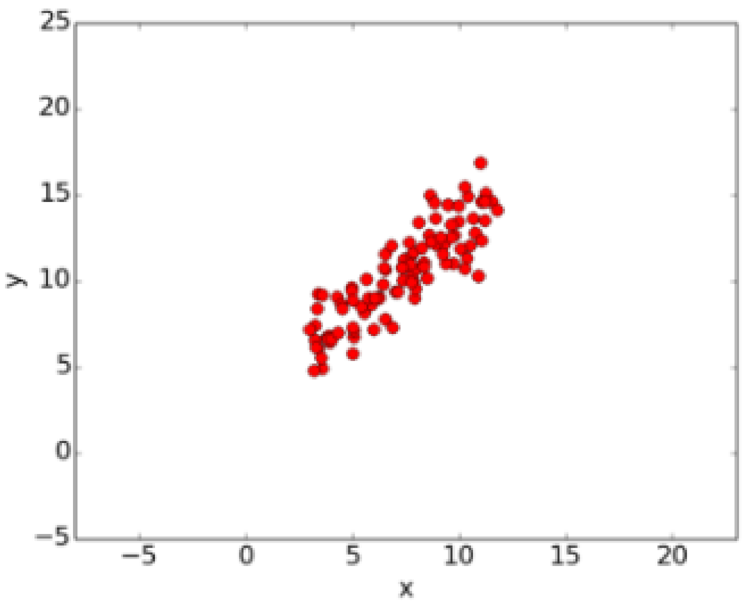
Sigmoid Activation in Perceptron


Bayes
Logistic regression
Perceptron
Linear regression
SVM

Overview

- Linear Regression – Review
- Linear Regression – as classifier – Problem with outliers
- Logistic Regression – Sigmoid
  - Classifier ?
  - Criterion used in logistic regression
    - Why don't we use sum of squared error ?
  - Solution

Linear regression



$$y = mx + b$$

We want to fit a straight line (in 2D case). The sample we are given with is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$.

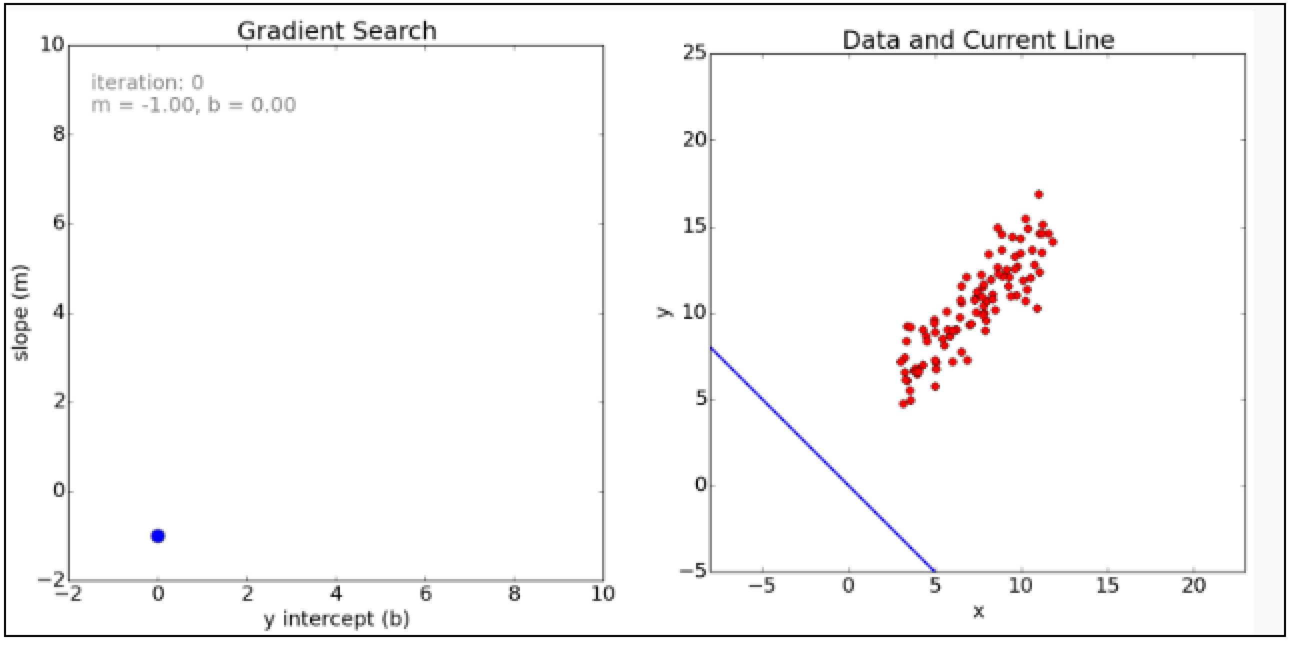
We want to find (m, b).

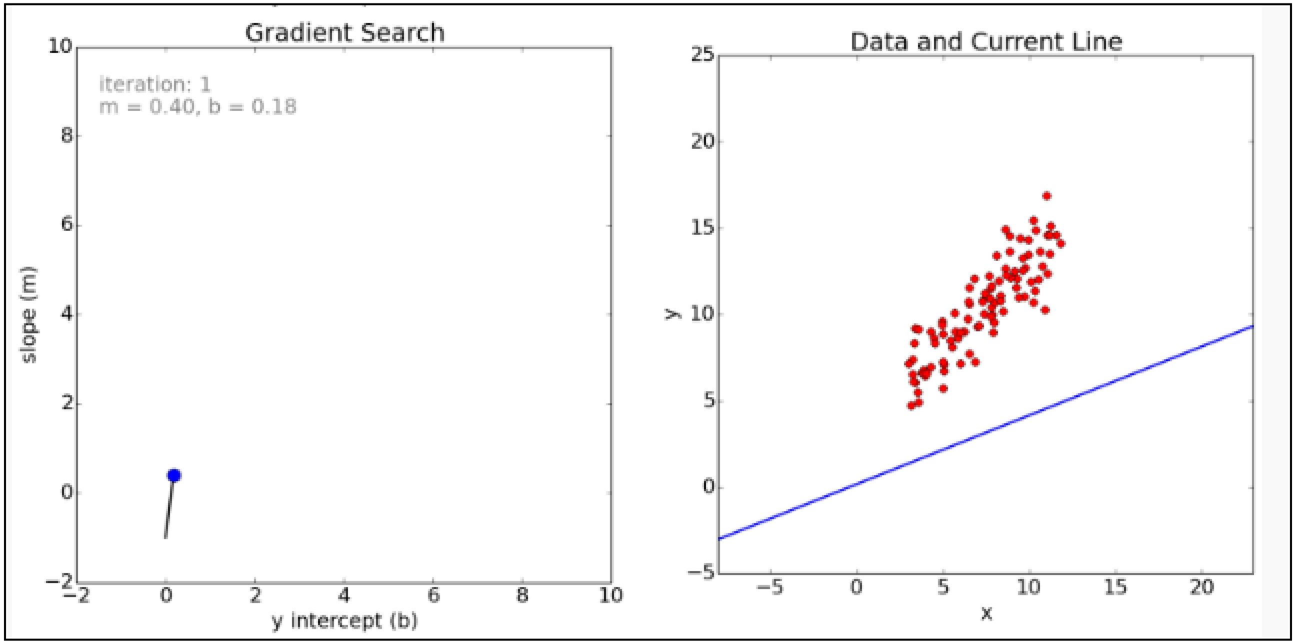
In the space (m, b) what is the function we are going to define? Minimum value of that function should be a solution for us.

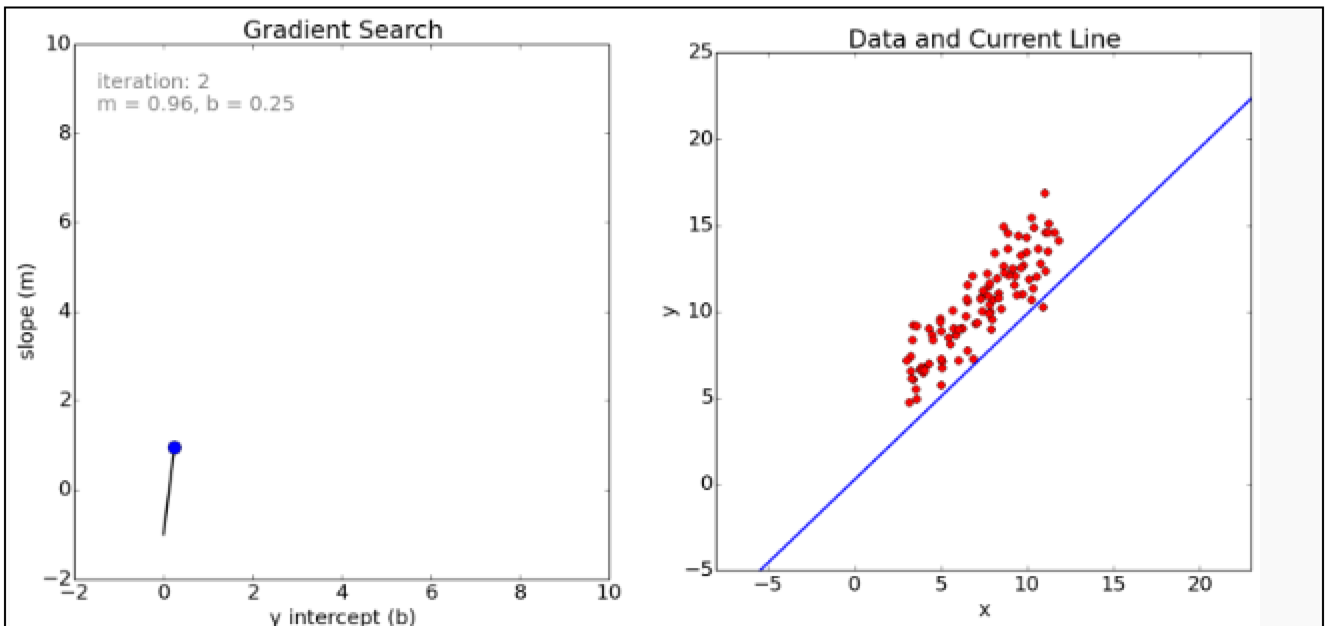
$$\text{Error}_{(m,b)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$

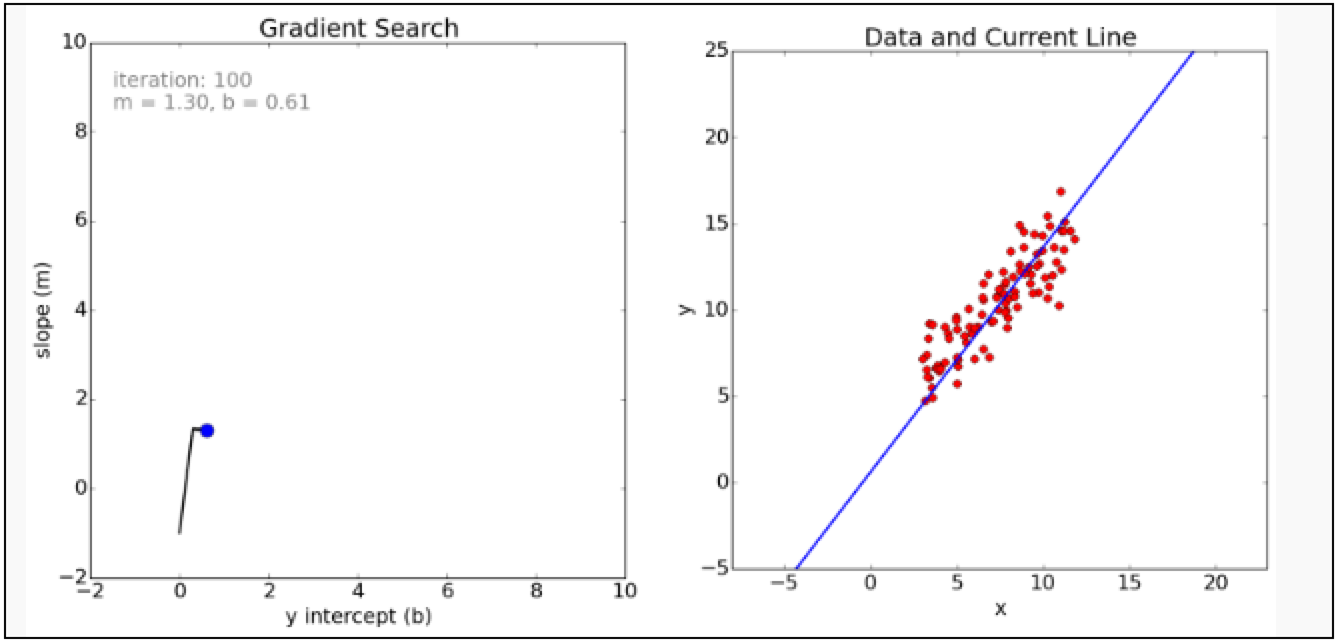
$$\frac{\partial}{\partial m} = \frac{2}{N}\sum_{i=1}^{N} -x_i(y_i - (mx_i + b))$$

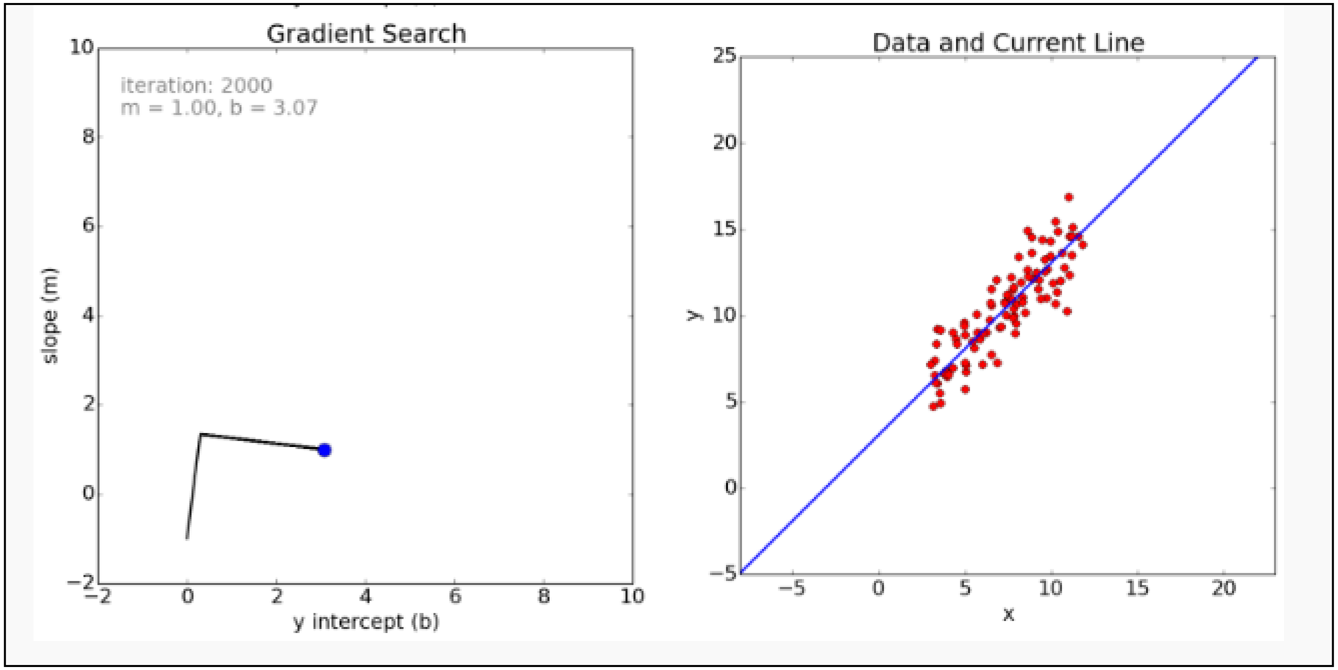
$$\frac{\partial}{\partial b} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (mx_i + b))$$




Gradient Search
iteration: 1
m = 0.40, b = 0.18
slope (m)
y intercept (b)
Data and Current Line
y
x


Gradient Search
iteration: 2
m = 0.96, b = 0.25
slope (m)
y intercept (b)
Data and Current Line
y
x


Gradient Search
iteration: 100
m = 1.30, b = 0.61
slope (m)
y intercept (b)
Data and Current Line
y
x

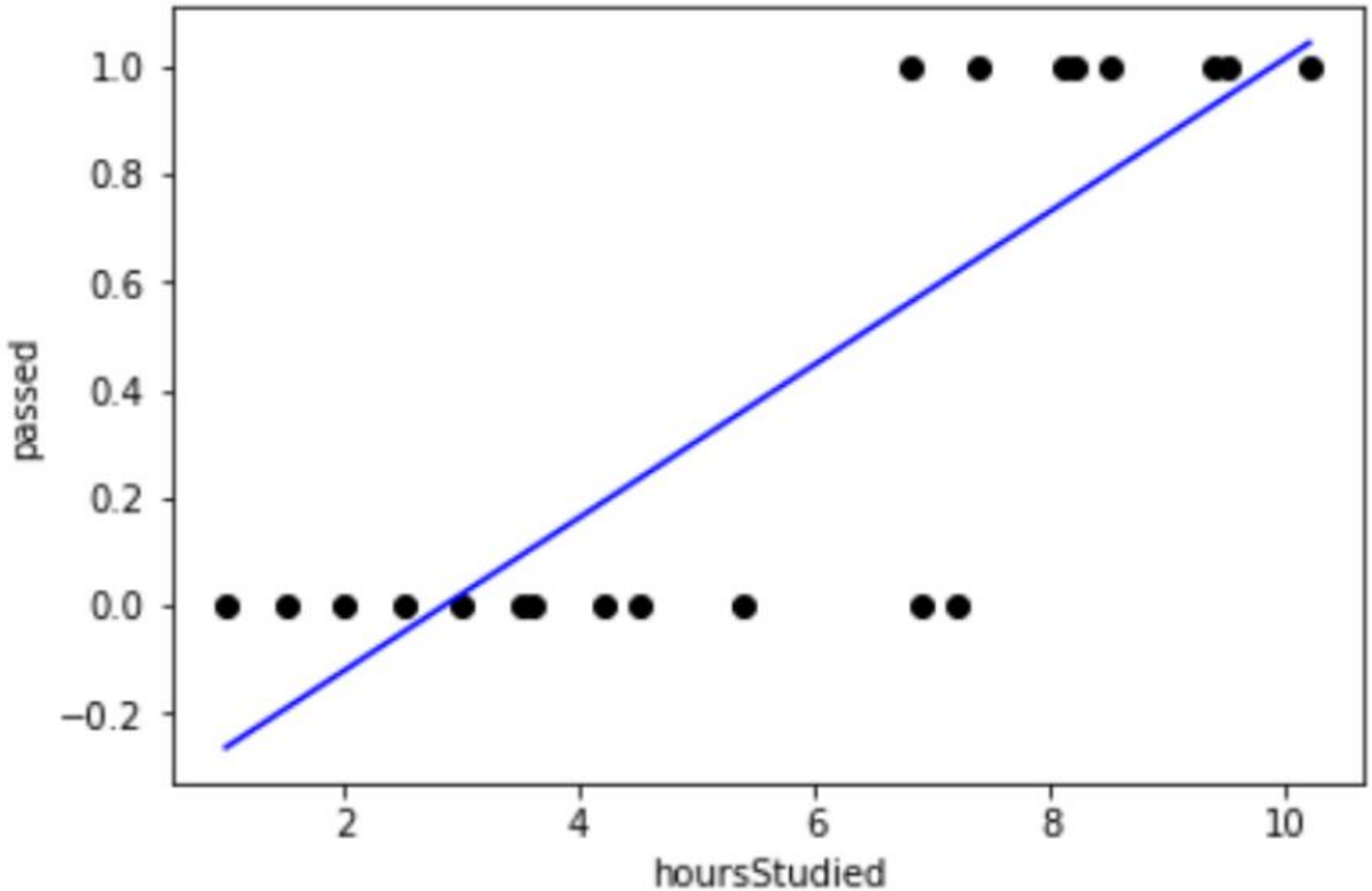

Gradient Search
iteration: 2000
m = 1.00, b = 3.07
slope (m)
y intercept (b)
Data and Current Line
y
x

Closed form solution

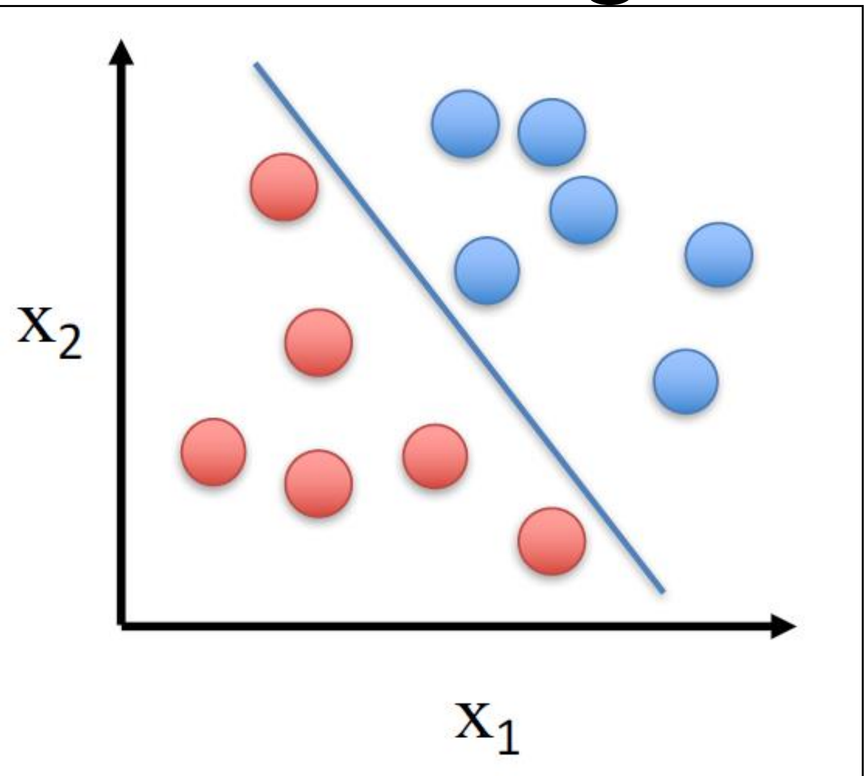
- Since the criterion that is minimized is quadratic, the linear regression problem must be having a closed form solution.
- This is nothing but applying the Newton's descent method.

Linear Regression for classification

Example with 1D data

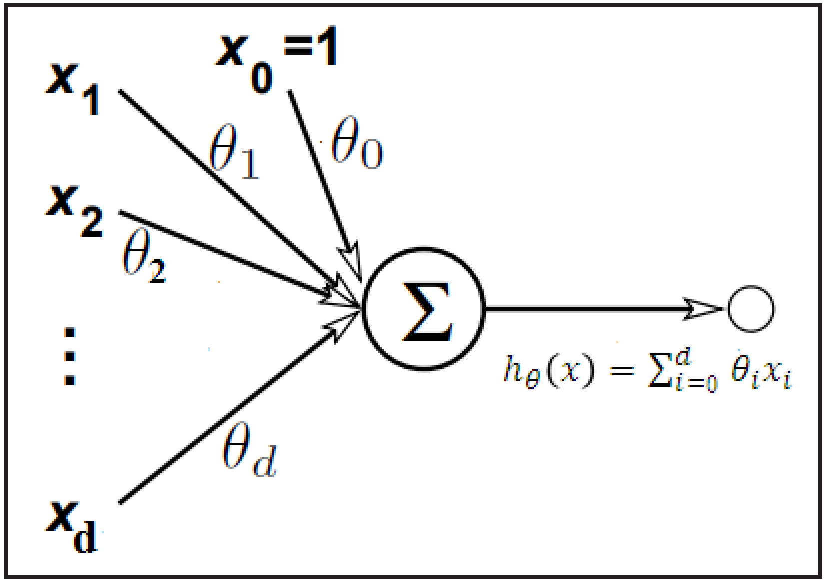


Linear Regression for classification



Example with 2D data

Direct attempt, in learning the linear discriminant



$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots \theta_d x_d$$

The parameter vector θ is learnt from the data such that the sum of squared error

$$E(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - h_\theta(x^{(i)})\right)^2$$

Where the training set

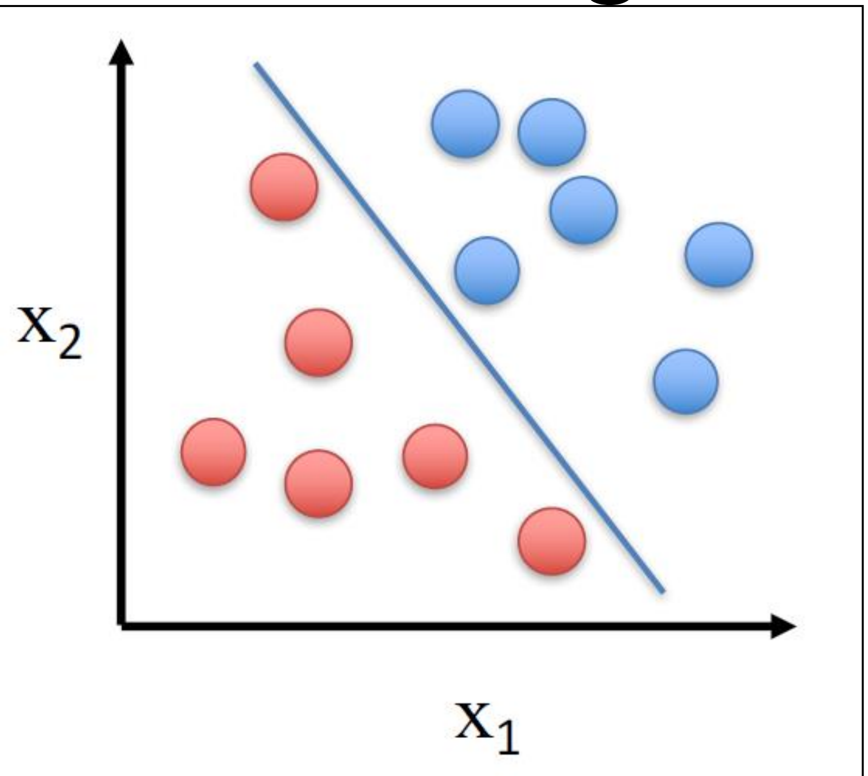
$$E(\theta)\ is\ convex(also\ quadratic)$$
$$\Rightarrow no\ local\ minima\ problem\ (closed\ form\ solution)$$

Training Procedure

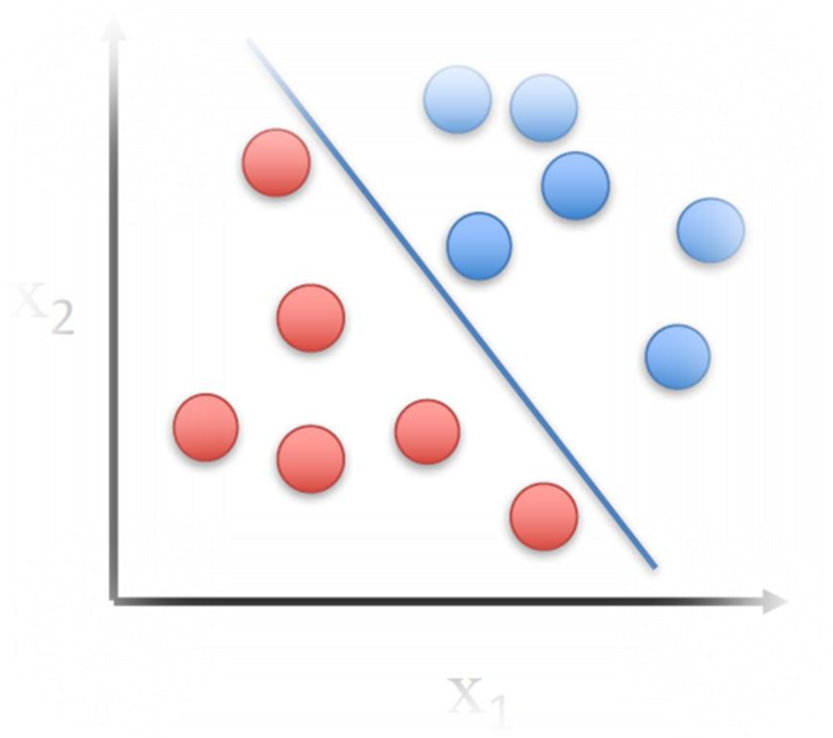
- $\nabla_\theta(E) = \sum_{i=1}^{n}\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)\left(-x^{(i)}\right)$
- *Batch Method:*
- $\theta_{new} = \theta + \eta \sum_{i=1}^{n}\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right) x^{(i)}$
- Single Sample (Stochastic update)
- $\theta_{new} = \theta + \eta\left(y^{(i)} - h_\theta(x^{(i)})\right)x^{(i)}$

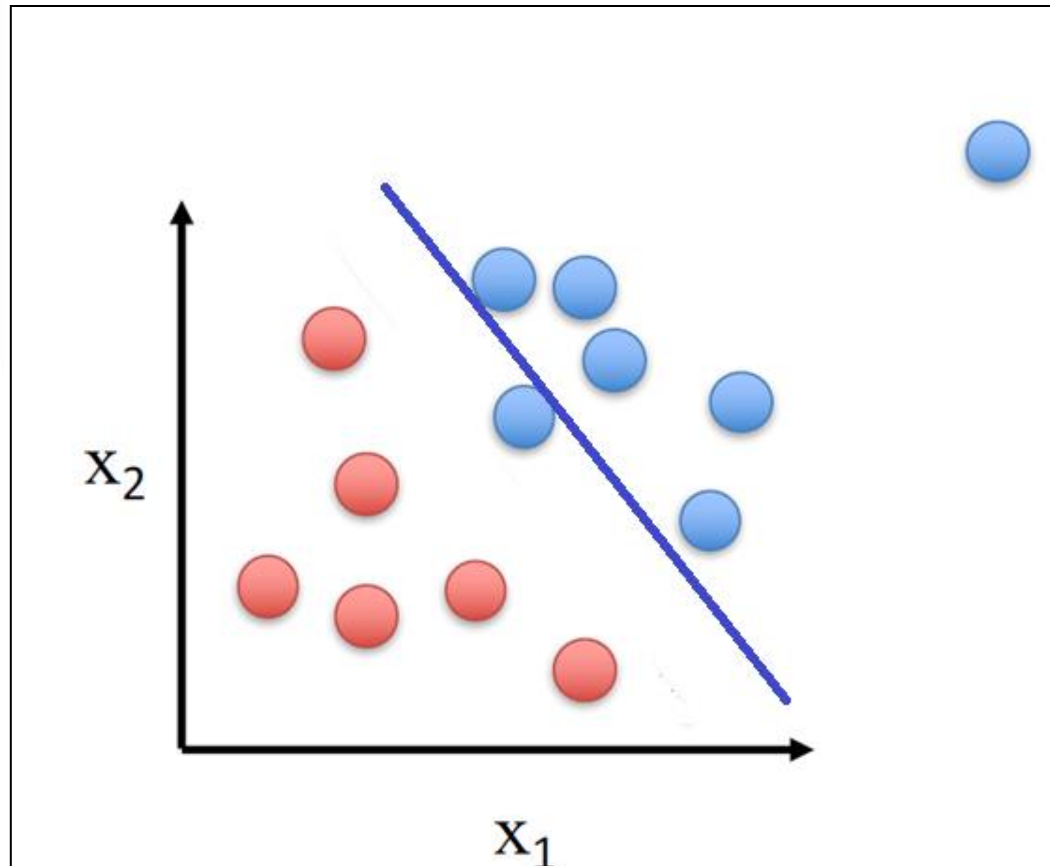
- $h_\theta\left(x^{(i)}\right) = \theta^T x^{(i)}$
- $E(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - h_\theta(x^{(i)})\right)^2$

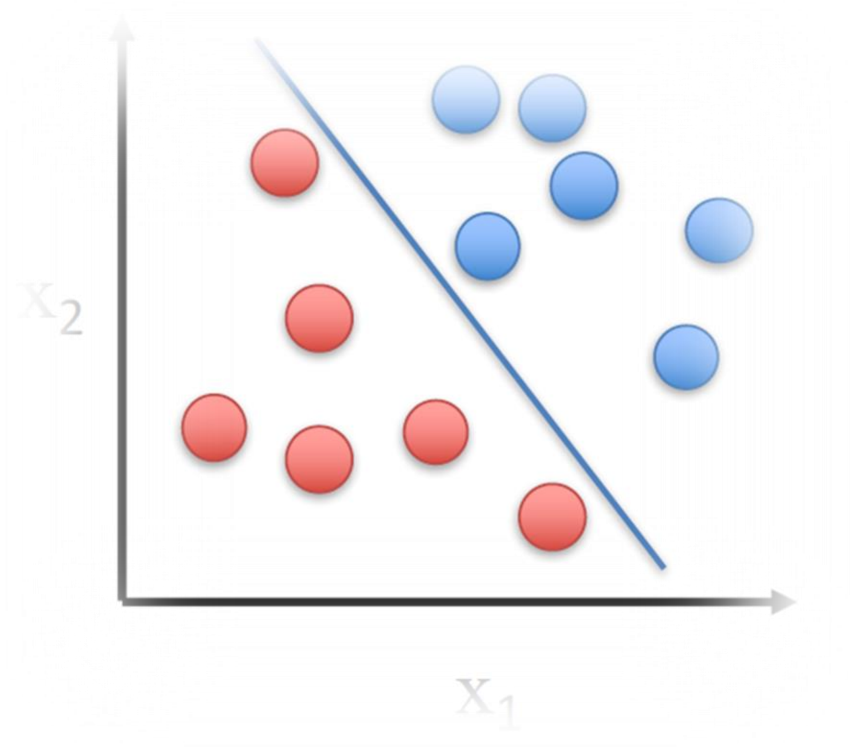
Linear Regression for classification



- But, outliers can be a big problem.

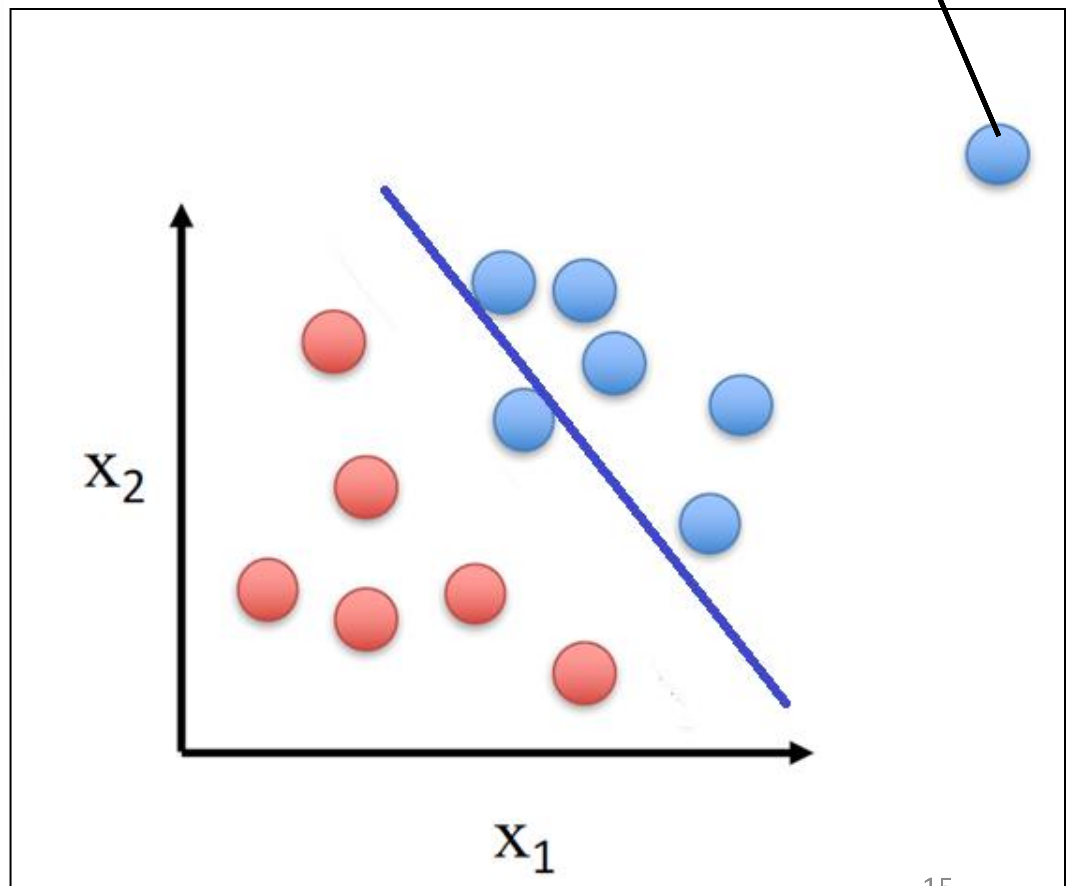

x_2
x_1

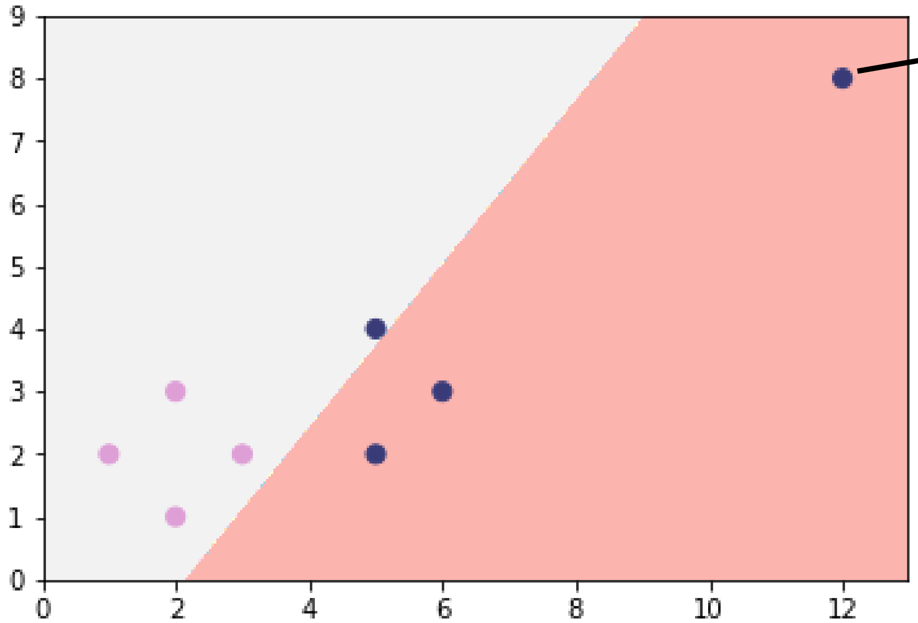

x_2
x_1



This is well within the blue class. This is a good training example.

Why this is causing us problem??





This is well within the blue class. This is a good training example.

Why this is causing us problem??

Note

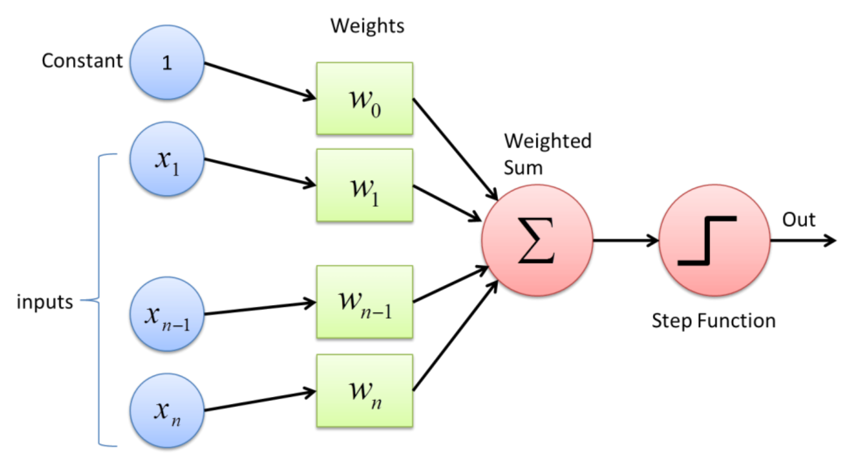
- Perceptron (Rosenblatt) does not have this problem (i.e., problem with seeming outliers).

Note

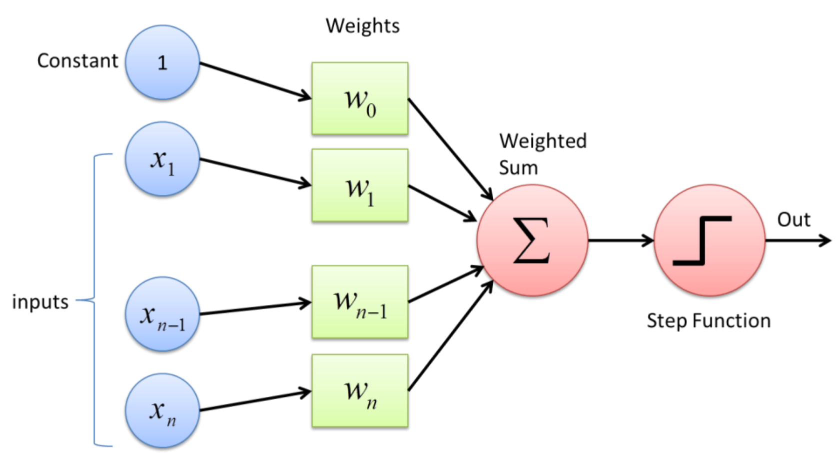
- Perceptron (Rosenblatt) does not have this problem (i.e., problem with seeming outliers).
- But works only for linearly separable data.

Note

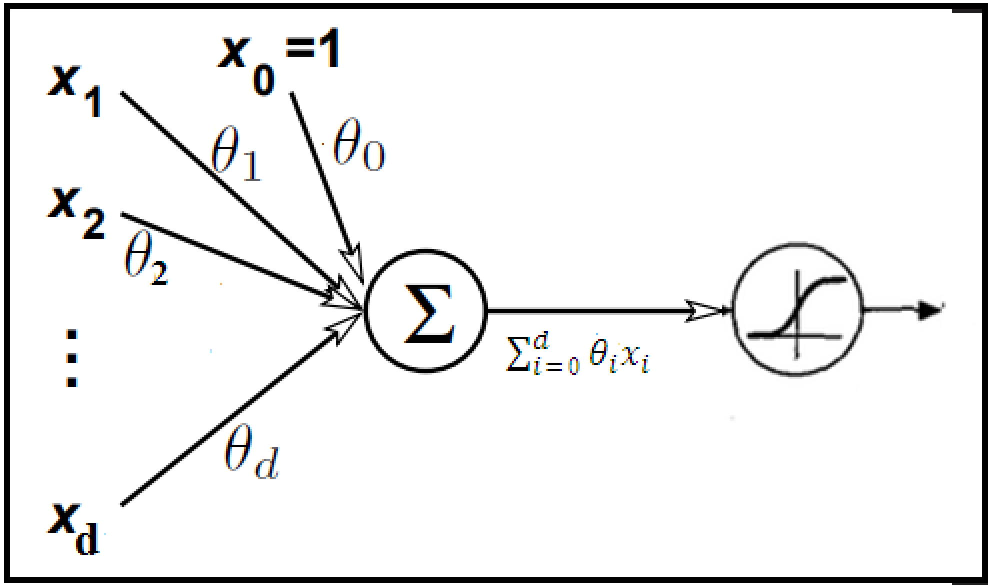
- Perceptron (Rosenblatt) does not have this problem (i.e., problem with seeming outliers).
- But works only for linearly separable data.
- Actually, it is



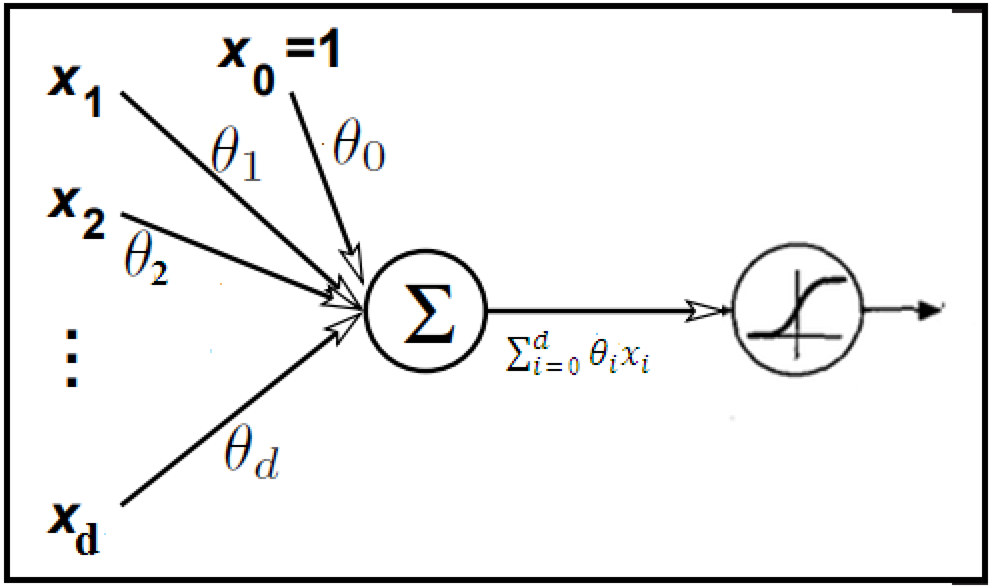
- Step (or Sign) function is not differentiable.
- So we can't employ gradient descent to get the minimum error (in this formulation) solution.



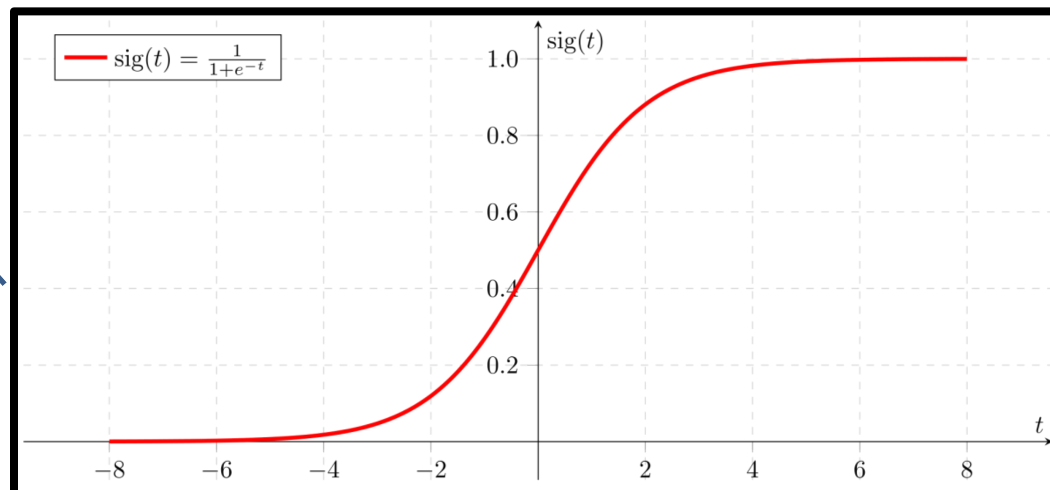
Logistic Regression



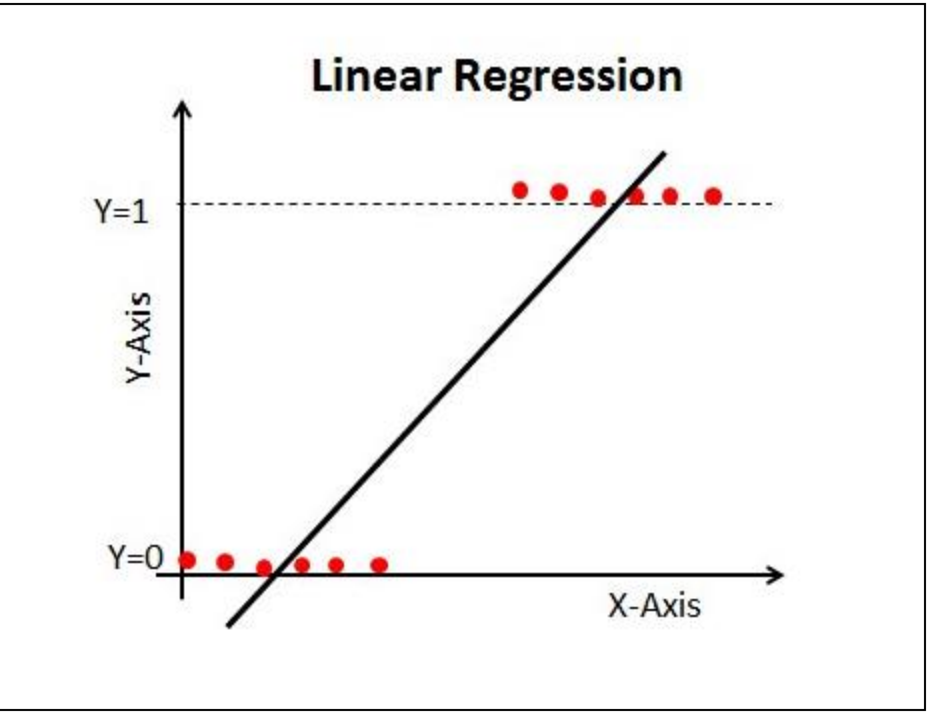
$x_0 = 1$
x_1
x_2
$\vdots$
x_d
θ_0
θ_1
θ_2
θ_d
Σ
$\sum_{i=0}^{d} \theta_i x_i$



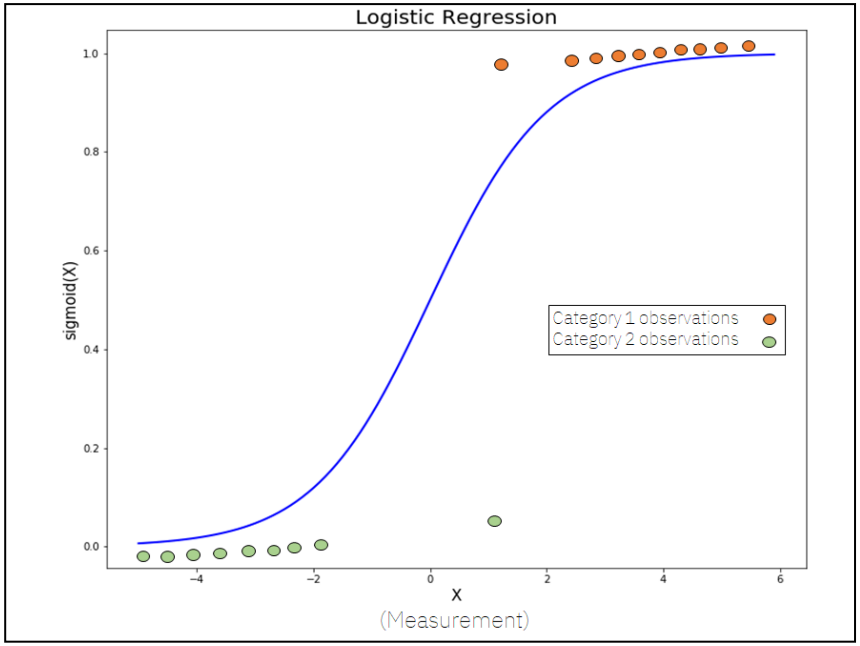
Sigmoid Activation Function



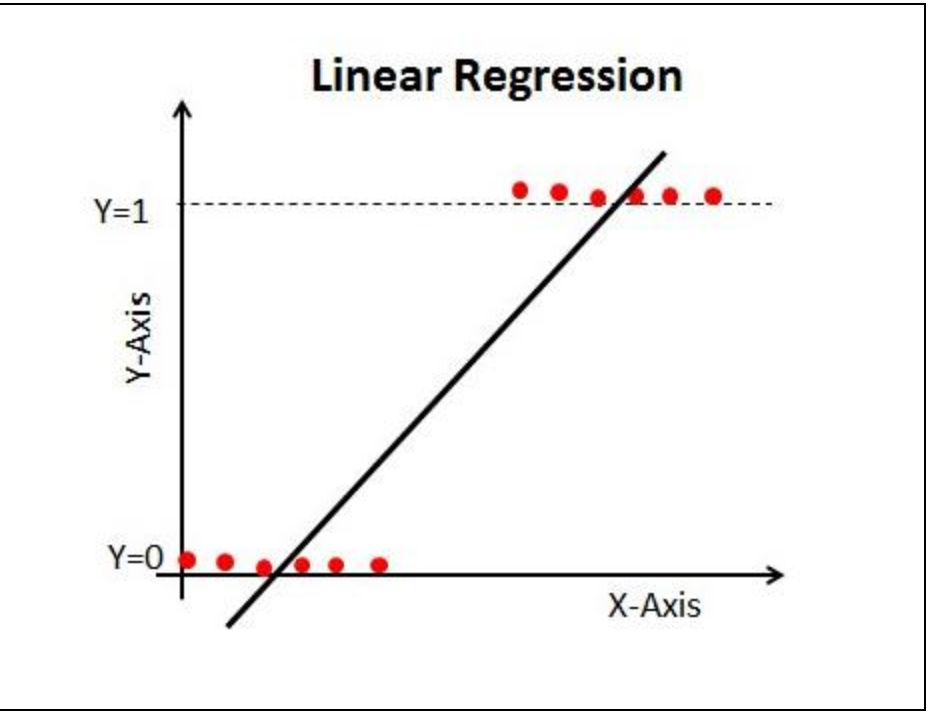
1D : what it does



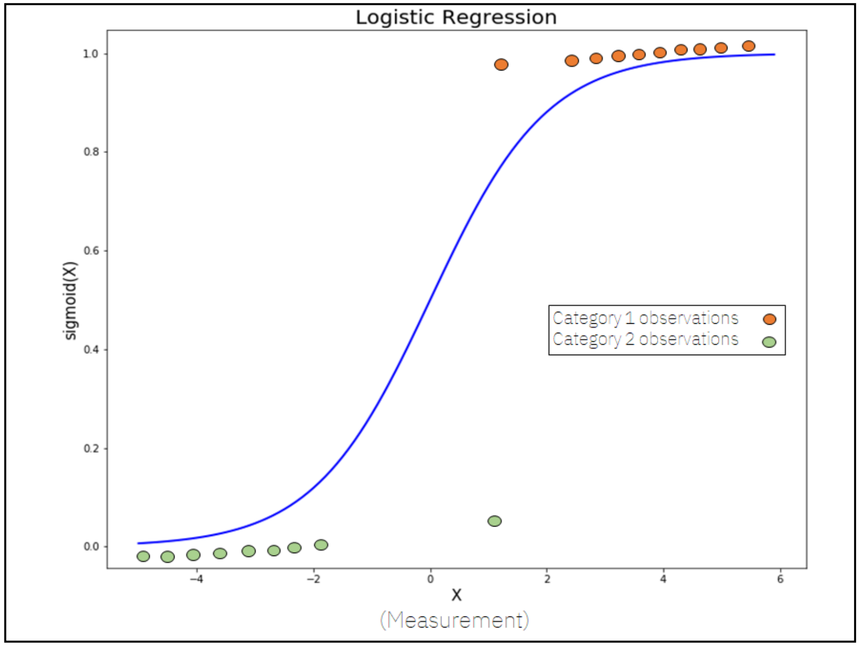
Logistic Regression



1D : what it does

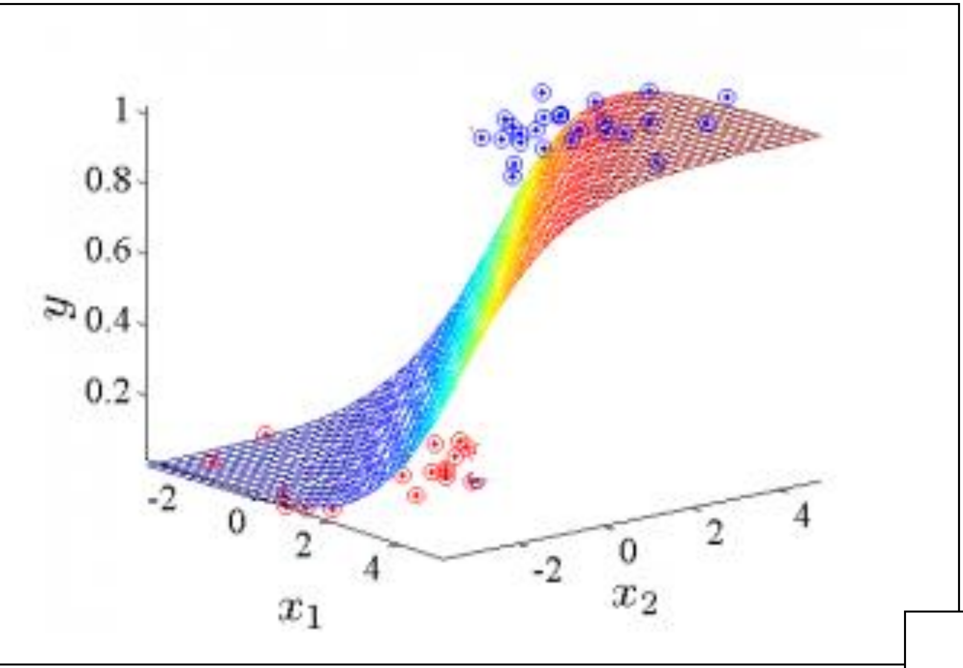


Logistic Regression

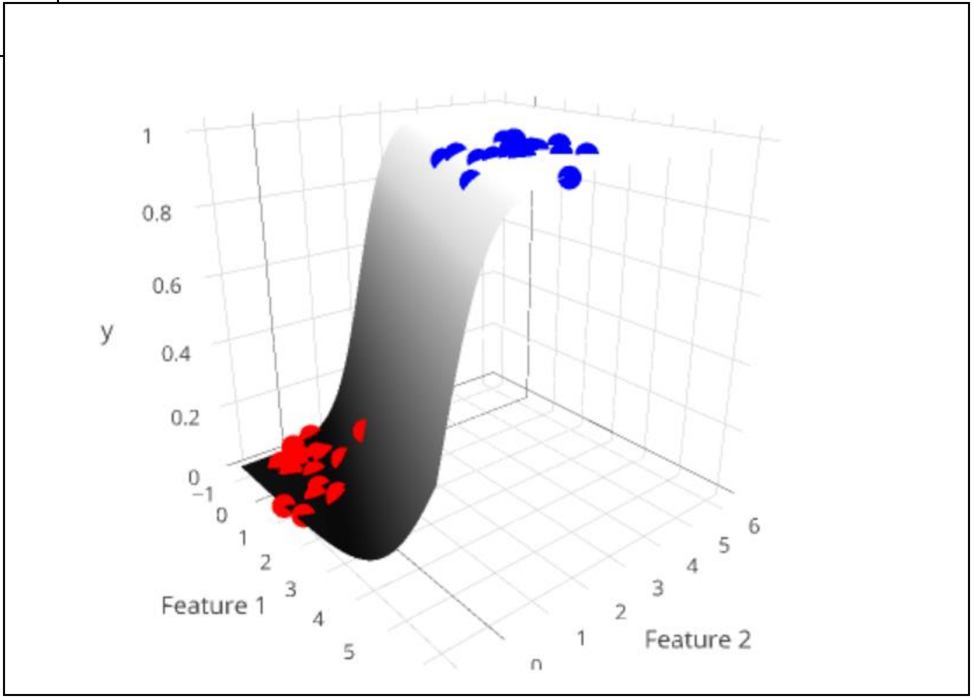


See, the seeming outliers are indeed very good fits.

2D data



1
0.8
0.6
0.4
0.2
y
-2
0
2
4
x_1
-2
0
2
4
x_2



1
0.8
0.6
0.4
0.2
0
y
−1
0
1
2
3
4
5
Feature 1
1
2
3
4
5
6
Feature 2

Notation used

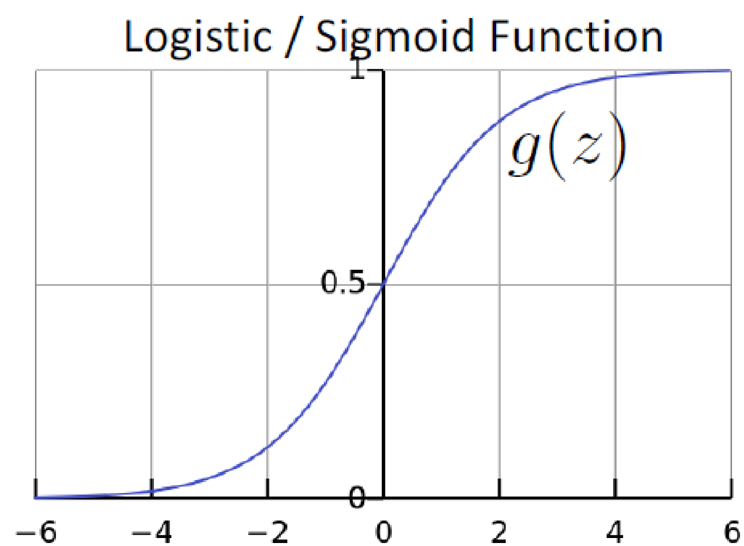
- $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$, $x = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$

- Logistic regression model:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^\intercal \boldsymbol{x}\right)$$

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

$$\boxed{h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{\theta}^\intercal \boldsymbol{x}}}}$$



Notation used

- Given $\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}$

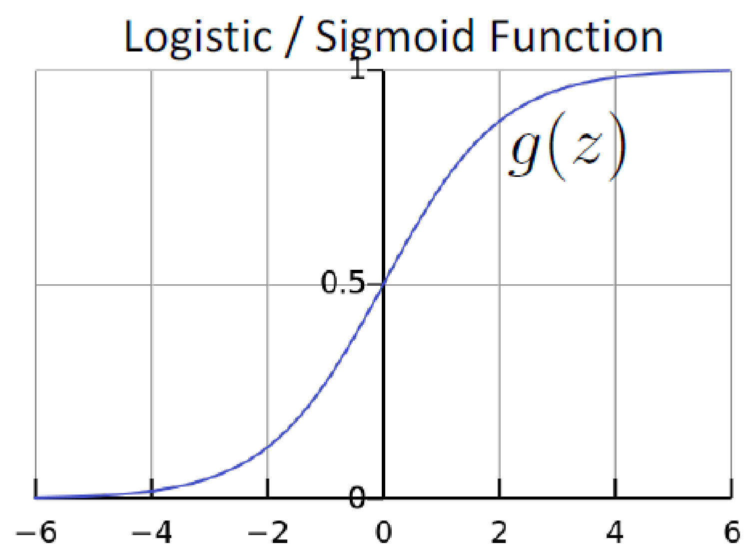
  where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d,\ y^{(i)} \in \{0, 1\}$

- Logistic regression model:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^\intercal \boldsymbol{x}\right)$$

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

$$\boxed{h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\intercal \boldsymbol{x}}}}$$



Differentiation of the sigmoid

$$
\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z)).
\end{aligned}
$$

Logistic Regression Objective Function

- Can’t just use squared loss as in linear regression:

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2$$

  – Using the logistic regression model

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\top \boldsymbol{x}}}$$

  results in a non-convex optimization

Classification Based on Probability

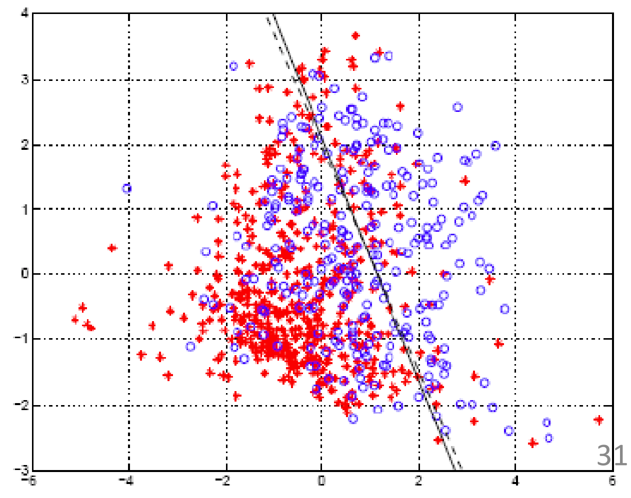
- Instead of just predicting the class, give the probability of the instance being that class
  - i.e., learn $p(y \mid x)$

- Comparison to perceptron:
  - Perceptron doesn't produce probability estimate

- Recall that:

$$0 \leq p(\text{event}) \leq 1$$

$$p(\text{event}) + p(\neg\text{event}) = 1$$



- Takes a probabilistic approach to learning discriminative functions (i.e., a classifier)
- $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ should give $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$
  - Want $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$

Can't just use linear regression with a threshold

Interpretation of Hypothesis Output

$h_{\boldsymbol{\theta}}(\boldsymbol{x})$ = estimated $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

Example: Cancer diagnosis from tumor size

$$\boldsymbol{x} = \left[\begin{array}{c} x_0 \\ x_1 \end{array} \right] = \left[\begin{array}{c} 1 \\ \text{tumorSize} \end{array} \right]$$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0.7$$

→ Tell patient that 70% chance of tumor being malignant

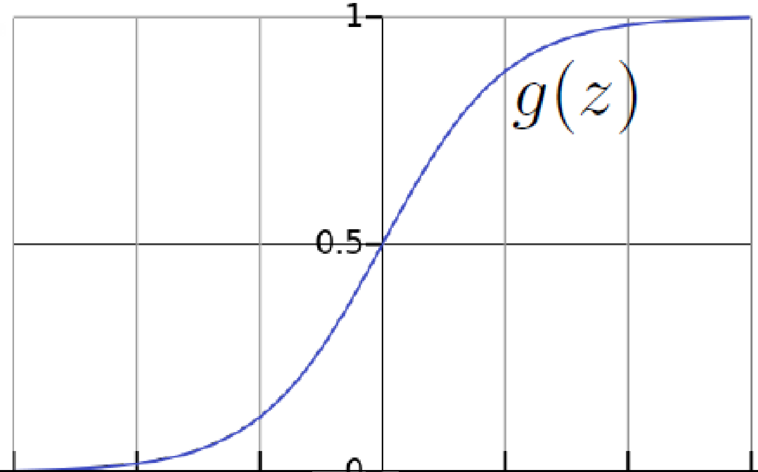
Note that: $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) + p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1$

Therefore, $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1 - p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

Logistic Regression

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

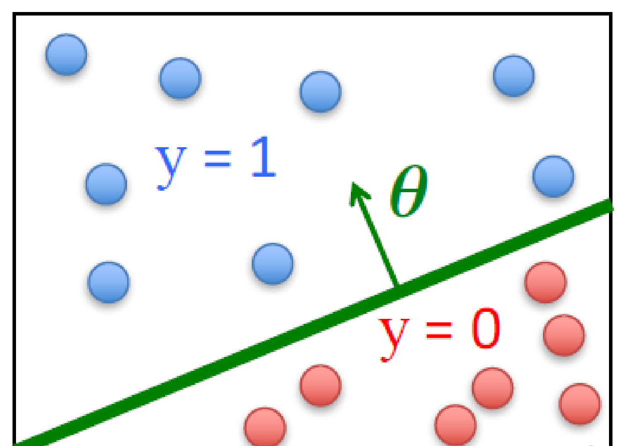
$$g(z) = \frac{1}{1 + e^{-z}}$$



$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large negative values for negative instances

$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large positive values for positive instances

- Assume a threshold and...
  - Predict y = 1 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$
  - Predict y = 0 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$

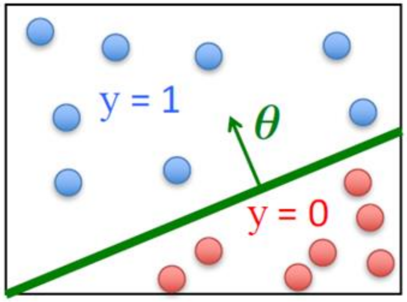


What we want …??

- We have

$$
\begin{aligned}
P(y=1 \mid x;\theta) &= h_\theta(x) \\
P(y=0 \mid x;\theta) &= 1-h_\theta(x)
\end{aligned}
$$

- Given



- Find θ so that the data agrees to the maximum extent.

- We can use maximum likelihood parameter estimation (MLE)

- Likelihood of data is given by: $l(\boldsymbol{\theta}) = \prod_{i=1}^{n} p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$

- So, looking for the $\boldsymbol{\theta}$ that maximizes the likelihood

$$\boldsymbol{\theta}_{\mathrm{MLE}} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

- Can take the log without changing the solution:

$$\begin{aligned}\boldsymbol{\theta}_{\mathrm{MLE}} &= \arg\max_{\boldsymbol{\theta}} \log \prod_{i=1}^{n} p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})\end{aligned}$$

$$
\begin{aligned}
P(y = 1 \mid x; \theta) &= h_\theta(x) \\
P(y = 0 \mid x; \theta) &= 1 - h_\theta(x)
\end{aligned}
$$

Note that this can be written more compactly as

$$
p(y \mid x; \theta) = (h_\theta(x))^y \left(1 - h_\theta(x)\right)^{1-y}
$$

Likelihood

$$
\begin{aligned}
L(\theta) &= \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta) \\
&= \prod_{i=1}^{n} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}}
\end{aligned}
$$

- Log-likelihood

$$
\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \sum_{i=1}^{n} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))
\end{aligned}
$$

Find θ that maximizes $l(\theta)$

- $l(\theta)$ is taken to be the criterion, but, note this has to be maximized.
- We can employ gradient ascent method.
- We can show that, the negative of the log-likelihood, i.e., $-l(\theta)$ which should be minimized is convex (hence no local minima problem)
- For proof of this refer the link http://mathgotchas.blogspot.com/2011/10/why-is-error-function-minimized-in.html

- $\nabla_\theta l(\theta) = \big(y - h_\theta(x)\big)\, x$
  - Note, here x is a vector.
  - This is for a single training example.

- This we obtained from,

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} \ell(\theta) &= \left(y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1 - g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\
&= \left(y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1 - g(\theta^T x)} \right) g(\theta^T x)(1 - g(\theta^T x)) \frac{\partial}{\partial \theta_j} \theta^T x \\
&= \left(y(1 - g(\theta^T x)) - (1-y) g(\theta^T x) \right) x_j \\
&= \left(y - h_\theta(x) \right) x_j
\end{aligned}
$$

Stochastic or single sample update rule

- $\theta_{new} \quad = \theta + \eta \nabla_\theta l(\theta)$
  $= \theta + \eta\big(y - h_\theta(x)\big)\, x$

Batch Method

$$\theta_{new} = \theta + \eta \textstyle\sum_{i=1}^{n} \Big(y^{(i)} - h_\theta\big(x^{(i)}\big)\Big)\, x^{(i)}$$

Stochastic or single sample update rule

- $$\theta_{new} = \theta + \eta \nabla_\theta l(\theta)$$
  $$= \theta + \eta\big(y - h_\theta(x)\big)\, x$$

Batch Method

$$\theta_{new} = \theta + \eta \sum_{i=1}^{n} \left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right) x^{(i)}$$

This looks IDENTICAL to linear regression!!!

- *However, the form of the model is very different:*

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}}}$$

- The End

- Supplementary Material

- $h_\theta(x) = \sum_{i=1}^{d} \theta_i x_i$