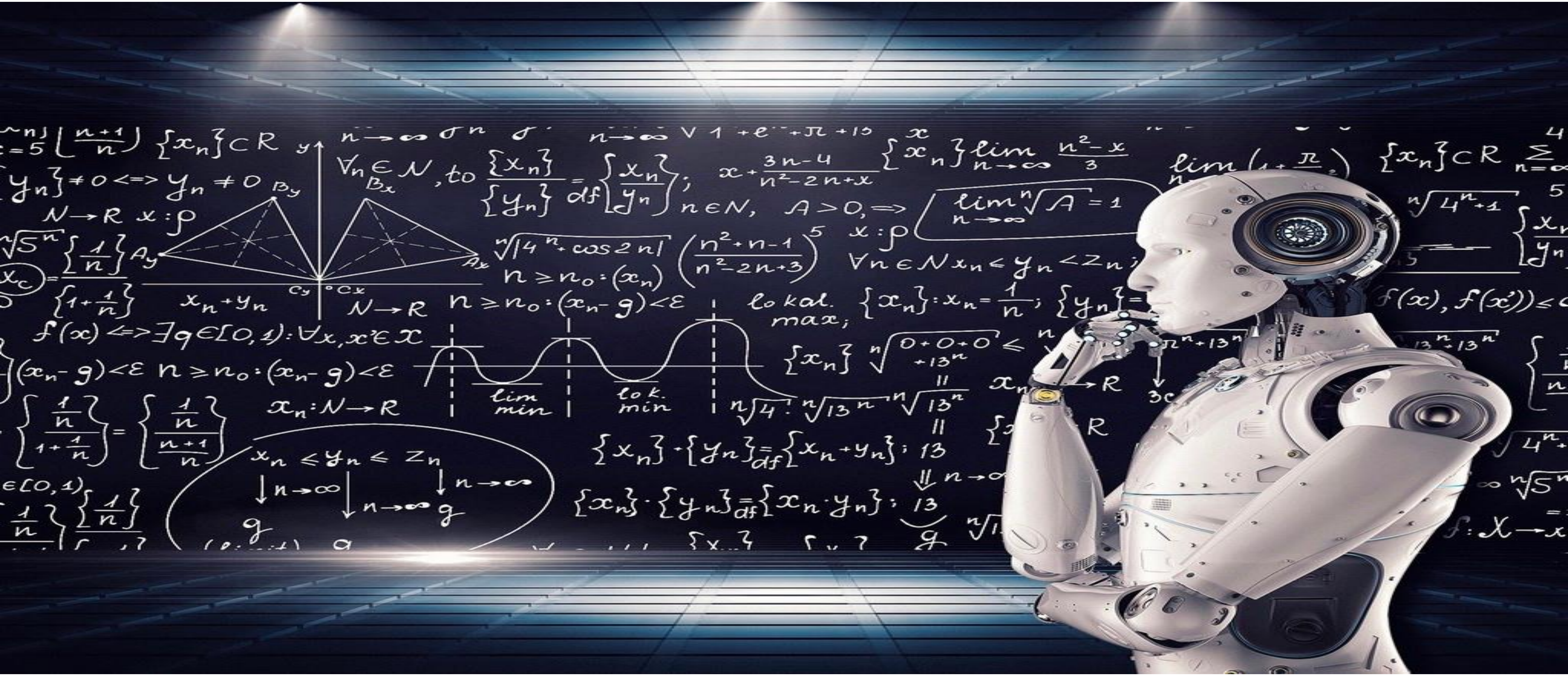
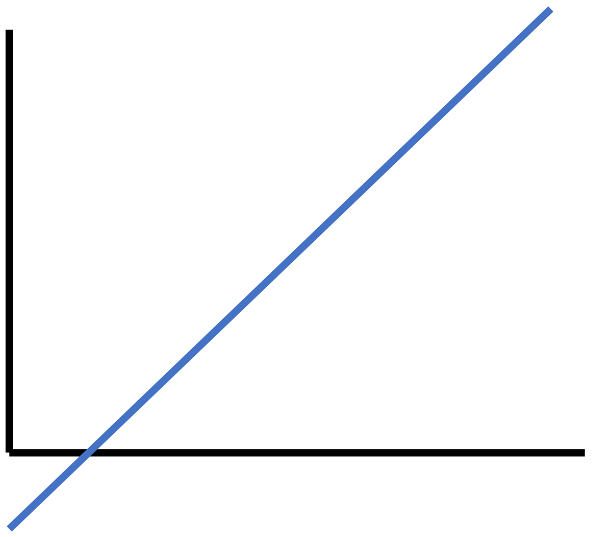
Perceptron



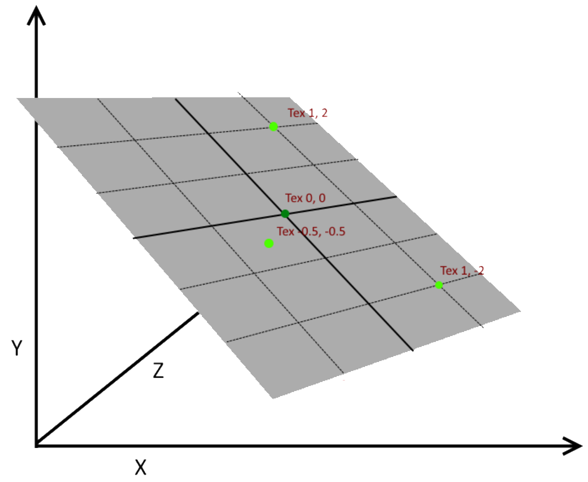
Some Algebra reminder - Hyperplane

- Hyperplane
  - A subspace whose dimension is one less than that of its ambient space.
  - 1D : A point

  - 2D : A line



3D : A plane



Some Algebra reminder

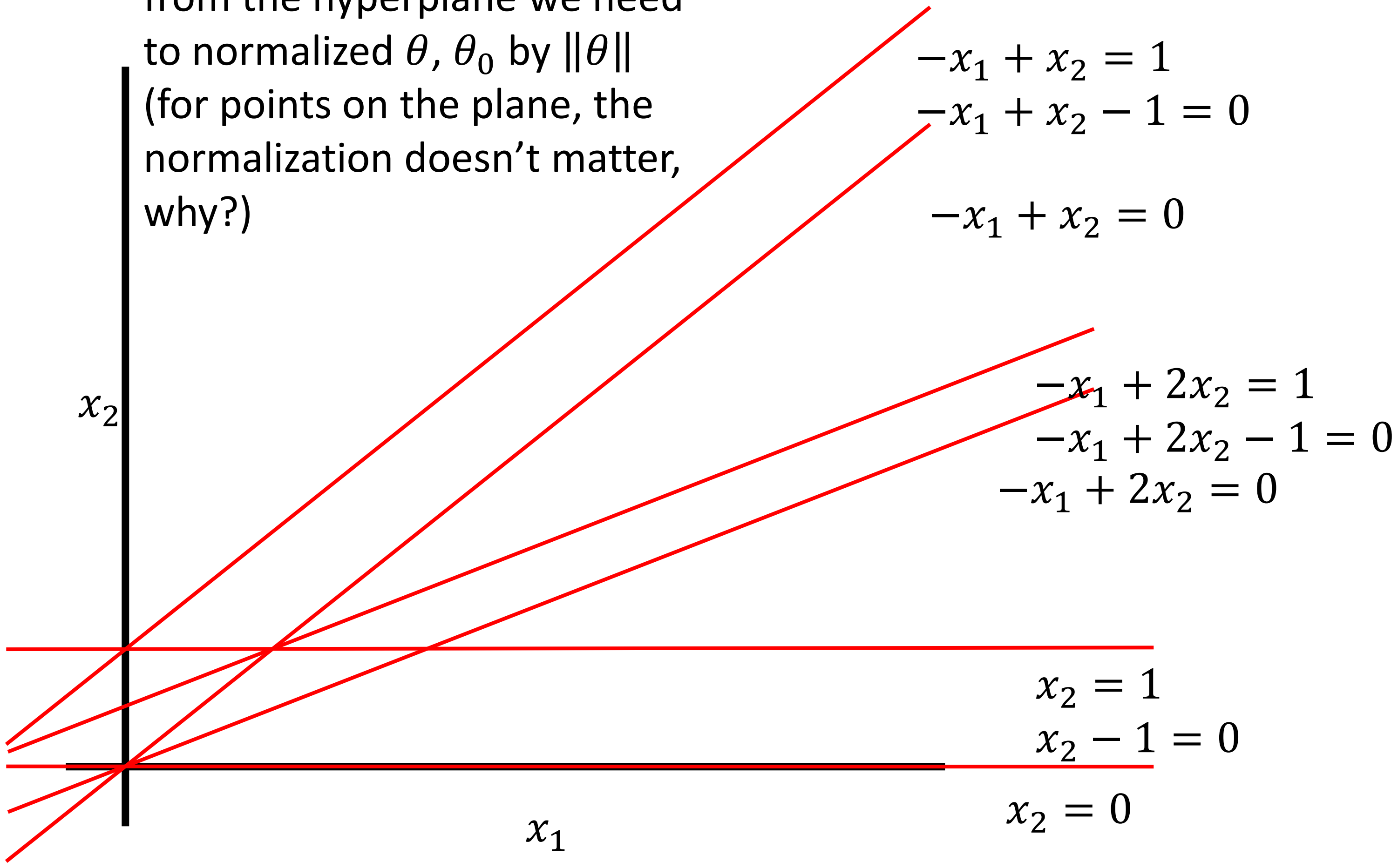
- How do we define a hyperplane in the space?
  - The space of the hyperplane is n-1 (if n is the space that we work on)
  - All the point on the hyperplane solve the equation

$$\theta_1 x_1 + \cdots + \theta_n x_n = b \quad (= \theta_0)$$

    - Where x are the point coordinates
  - The hyperplane separates the space into two half-spaces
    - All the point that the equation result > b
    - All the point that the equation result < b

Hyperplane - examples

In order to find the distance from the hyperplane we need to normalized θ, θ_0 by $||\theta||$ (for points on the plane, the normalization doesn't matter, why?)

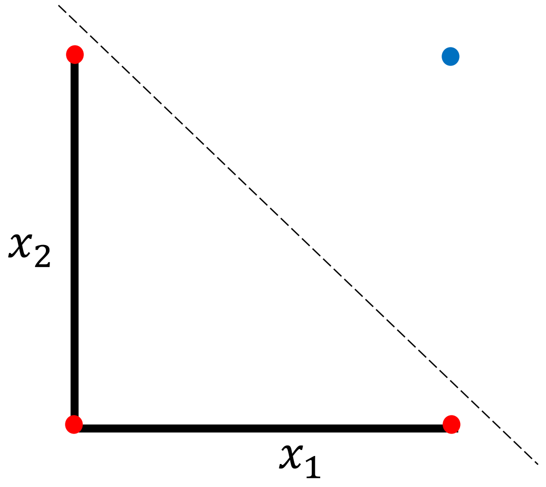


Linear separator

- We want to find linear separator:
  - All point above with result greater than 0, will be belong to the +1 class (or -1)
  - All point under with result lower than 0, will be belong to the -1 class (or +1)
- So, what do we need to find?
  - The hyperplane weights $\theta \in R^{n+1}$ (n hyperplane weights & the bias θ_0)
  - We will predict 1 if $\sum_{i=1}^{n} \theta_i x_i + \theta_0 > 0$ and -1 otherwise

Boolean functions – AND

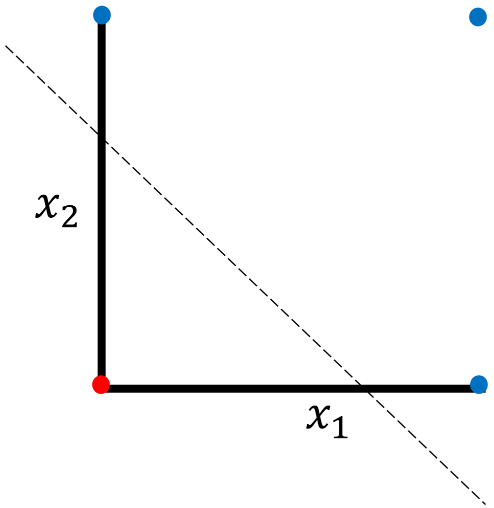
- X_1 AND X_2



- Solution?
- If $1 \times X_1 + 1 \times X_2 - 1.5 > 0$ predict 1
- Otherwise predict -1.
- i.e. $\theta_0 = -1.5, \theta_1 = 1, \theta_2 = 1$

OR

- X_1 OR X_2

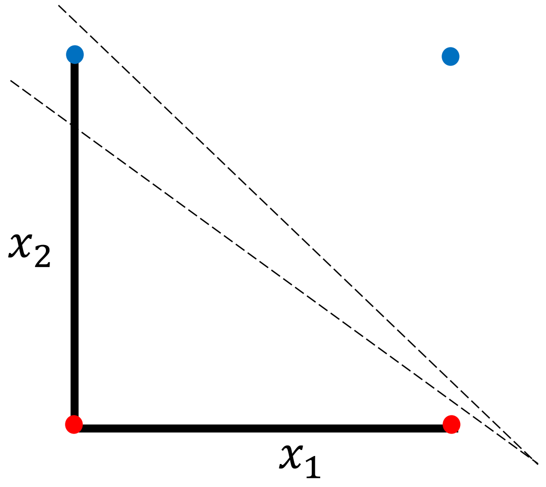


- Solution?
- $X_1 + X_2 - 0.5 > 0$ predict 1
- Otherwise -1

Perceptron

- After we know what we are looking for, linear separator, we need to know how to find it
- Simplest way
  - Start with random weights
  - In each step, improve if necessary – if error exist
  - Error = output - target

Perceptron



- The change in the weight will be

$$\Delta\theta_i = -\eta \sum_{d \in D} \left(o^{(d)} - t^{(d)}\right) x_i^{(d)}$$

for each dimension

Perceptron update rule

$$\Delta\theta_i = -\eta \sum_{d \in D} \left(o^{(d)} - t^{(d)}\right) x_i^{(d)}$$

- If $o^{(d)} - t^{(d)} = 0$, there is no error = no update

0	t	o-t	x_i	$\Delta\theta_i$	$x_i \cdot \theta_i$
-1	+1	<0	>0	>0	increased
-1	+1	<0	<0	<0	increased
+1	-1	>0	>0	<0	decreased
+1	-1	>0	<0	>0	decreased

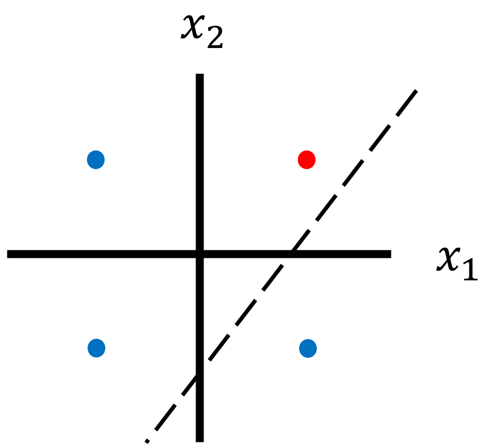
Perceptron Algorithm

- The algorithm:
  - Initialize weights to some small random number
  - Repeat until convergence (no error = no weight update):
    - For each $x^{(d)}$ in D compute: (* $x^{(d)} = \bar{x}_d$):
      - $o^{(d)} = sgn(\theta \cdot x^{(d)})$
    - For each θ_i do:
      - $\Delta\theta_i = -\eta \sum_{d \in D} \left(o^{(d)} - t^{(d)}\right) x_i^{(d)}$ for each i
      - Update $\theta_i = \theta_i + \Delta\theta_i$

Stochastic Perceptron

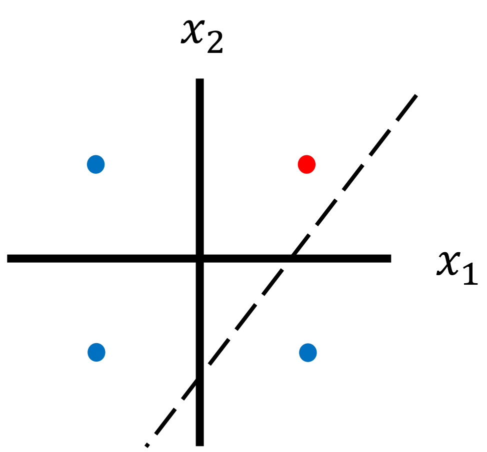
- The algorithm:
  - Set weights randomly
  - Repeat until convergence:
    - Choose d randomly (or in some order)
    - Calculate $o^{(d)} = sgn(\theta \cdot x^{(d)})$
    - Calculate $\Delta\theta_i = -\eta\left(o^{(d)} - t^{(d)}\right)x_i^{(d)}$ for each i
    - Then update $\theta_i = \theta_i + \Delta\theta_i$

Perceptron Example - Stochastic



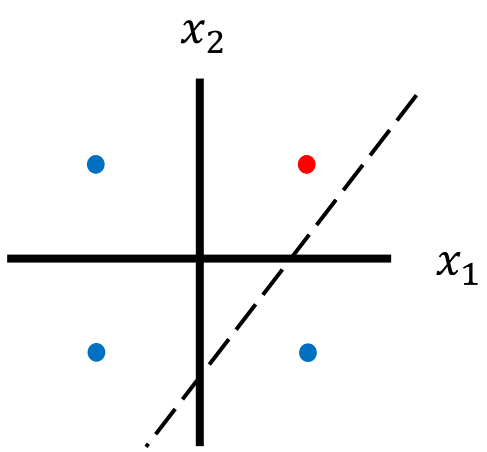
- Training data:
  - (-1,-1)->+1, (-1,+1)->+1, (+1,-1)->+1, (+1,+1)->-1
- Weight init:
  - θ_0 = 0.1 , θ_1 = -0.2, θ_2 = 0.15, η = 0.05

Perceptron Example - Stochastic



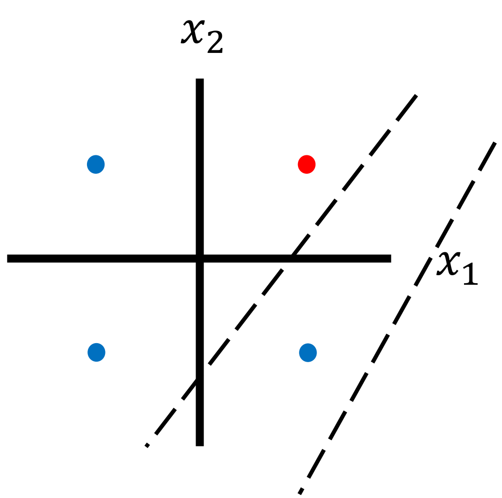
- Check (-1,-1)->+1
- $sgn(\theta \cdot x^{(d)}) = 0.1 - 0.2 * (-1) + 0.15 * (-1) = 0.15 > 0$
- o = +1
- Since t=o no update required

Perceptron Example - Stochastic



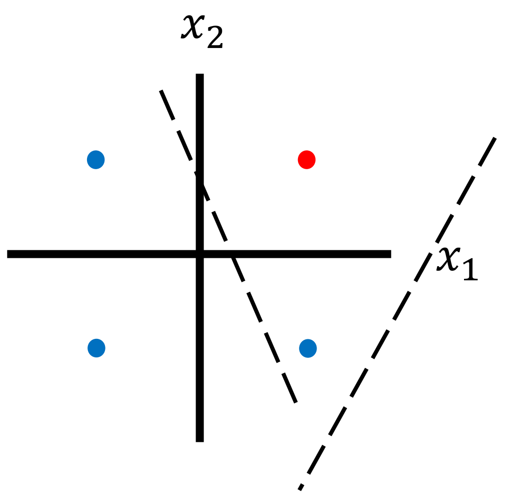
- Check (-1,+1)->+1
- $sgn(\theta \cdot x^{(d)}) = 0.1 - 0.2 * (-1) + 0.15 * (+1) = 0.45 > 0$
- o = +1
- Since t=o no update required

Perceptron Example - Stochastic



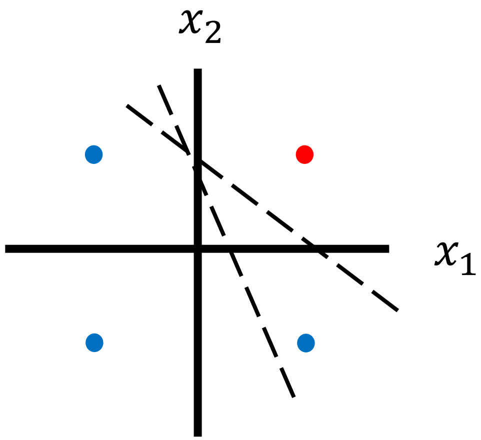
- Check (+1,-1)->+1
- $sgn(\theta \cdot x^{(d)}) = 0.1 - 0.2 * (+1) + 0.15 * (-1) = -0.25 < 0$
- o = -1
- Since t!=o update required:
  - $\theta_{0(new)} = 0.1 - 0.05 * (-1 - 1) * 1 = 0.2$
  - $\theta_{1(new)} = -0.2 - 0.05 * (-1 - 1) * 1 = -0.1$
  - $\theta_{2(new)} = 0.15 - 0.05 * (-1 - 1) * (-1) = 0.05$

Perceptron Example - Stochastic



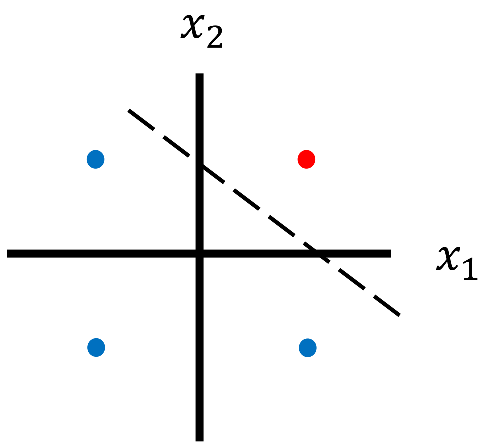
- Check (+1,+1)->-1
- $sgn(\theta \cdot x^{(d)}) = 0.2 - 0.1 * (+1) + 0.05 * (+1) = 0.15 > 0$
- o = +1
- Since t!=o update required:
  - $\theta_{0(new)} = 0.2 - 0.05 * (+1 - (-1)) * 1 = 0.1$
  - $\theta_{1(new)} = -0.1 - 0.05 * (+1 - (-1)) * 1 = -0.2$
  - $\theta_{2(new)} = 0.05 - 0.05 * \big(+1 - (-1)\big) * 1 = -0.05$

Perceptron Example - Stochastic



- Check (+1,-1)->+1
- $sgn(\theta \cdot x^{(d)}) = 0.1 - 0.2 * (+1) - 0.05 * (-1) = -0.05 < 0$
- o = -1
- Since t!=o update required:
  - $\theta_{0(new)} = 0.1 - 0.05 * (-1 - 1)) * 1 = 0.2$
  - $\theta_{1(new)} = -0.2 - 0.05 * (-1 - 1)) * 1 = -0.1$
  - $\theta_{2(new)} = -0.05 - 0.05 * (-1 - 1) * (-1) = -0.15$

Perceptron Example - Stochastic



- We got the linear separator:

$$\bar{\theta} \cdot \bar{x} = -0.1 * x_1 - 0.15 * x_2 + 0.2 = 0$$

Perceptron problem

- What is the problem in the perceptron algorithm?
- How can we solve it?

LMS – Least Mean Squares

- The algorithm:
  - Initialize weights to some small random number
  - Repeat until convergence (no error = no weight update):
    - For each $x^{(d)}$ in D compute: (* $x^{(d)} = \bar{x}_d$):
      - $o^{(d)} = (\theta \cdot x^{(d)})$
    - For each θ_i do:
      - $\Delta\theta_i = -\eta \sum_{d \in D} \left(o^{(d)} - t^{(d)}\right) x_i^{(d)}$
      - Update $\theta_i = \theta_i + \Delta\theta_i$

LMS – Least Mean Squares

- Our goal here is not to find linear hyperplane that separate the data, but to minimize the distance from the offset (either +1 or -1) of the hyperplane
- The linear hyperplane that we will use to predict new instance is the outcome of this minimization

LMS – Least Mean Squares

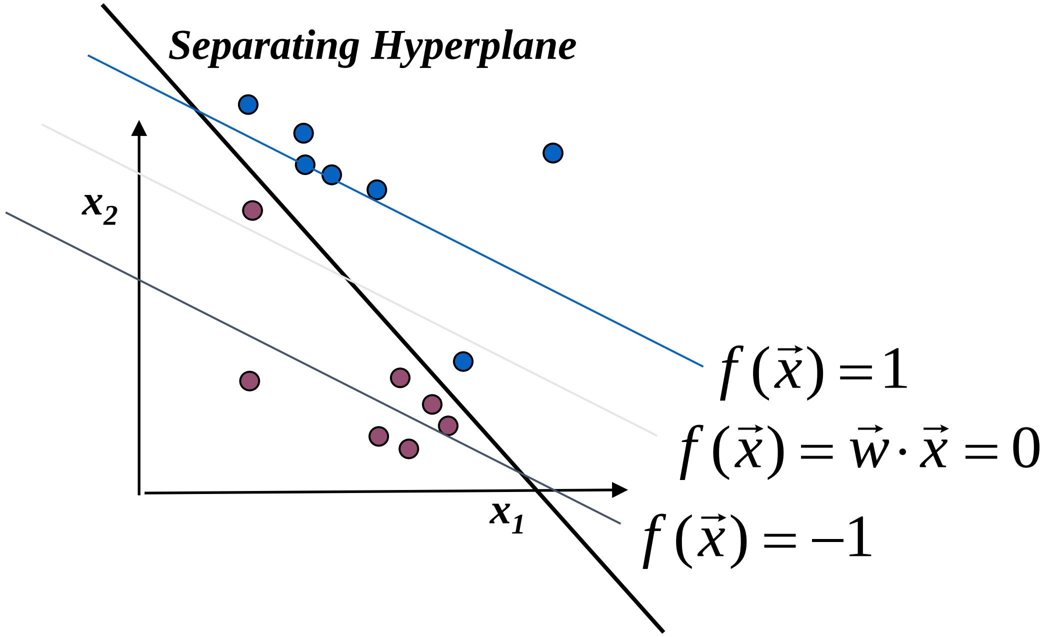
$$E[\vec{\theta}] = \frac{1}{2}\sum_{d\in D}\left(o^{(d)} - t^{(d)}\right)^2 = \frac{1}{2}\left[\sum_{d\in D^+}\left(o^{(d)} - 1\right)^2 + \sum_{d\in D^-}\left(o^{(d)} + 1\right)^2\right]$$

Minimize the distance between the positive instances and the +1 iso-line of the function

Minimize the distance between the negative instances and the -1 iso-line of the function

LMS – Least Mean Squares

$$E[\vec{\theta}] = \frac{1}{2}\sum_{d\in D}\left(o^{(d)} - t^{(d)}\right)^2 = \frac{1}{2}\left[\sum_{d\in D^+}\left(o^{(d)} - 1\right)^2 + \sum_{d\in D^-}\left(o^{(d)} + 1\right)^2\right]$$



LMS – Least Mean Squares

- What is the differences between Perceptron and LMS?
  - The target is numerical value (not class) – in this case, +1 or -1
  - The output calculation – the result is a number & not a class (the real value of $\theta \cdot x^{(d)}$ and not $\operatorname{sgn}(\theta \cdot x^{(d)})$)
  - The optimization function – in LMS minimum distance from +1 & -1 hyper planes. In perceptron zero classification error
  - LMS will converge, perceptron only if the data are linear separable (no error on training data)

Logistic Regression

- Linear Regression - Recap
- Predict a continuous value

x_1	y
	10000
	21011
	19213
	15213
	18212
	22001

- Hypothesis function :

$$h_\theta(x) = \theta^T x$$

- Cost function (MSE):

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

- Goal :

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

Linear Regression -> Classification

- What if instead of predicting a continues value we try to predict a class?

- Hypothesis function :

$$h_\theta(x) = \theta^T x$$

- Cost function (MSE):

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

- Goal :

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

x_1	y
	1
	1
	0
	1
	0
	1

Linear Regression -> Classification

- What if instead of predicting a continues value we try to predict a class?

- Hypothesis function :

$$h_\theta(x) = \theta^T x$$

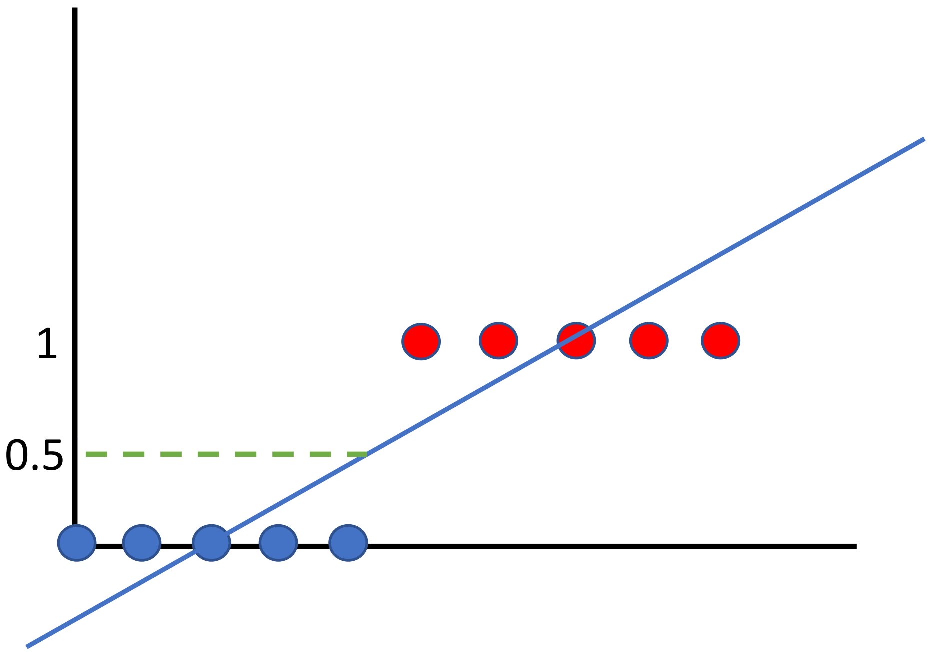
- Predict

$$if\ h_\theta\,(x) > 0.5\ then\ 1$$
$$else\ 0$$

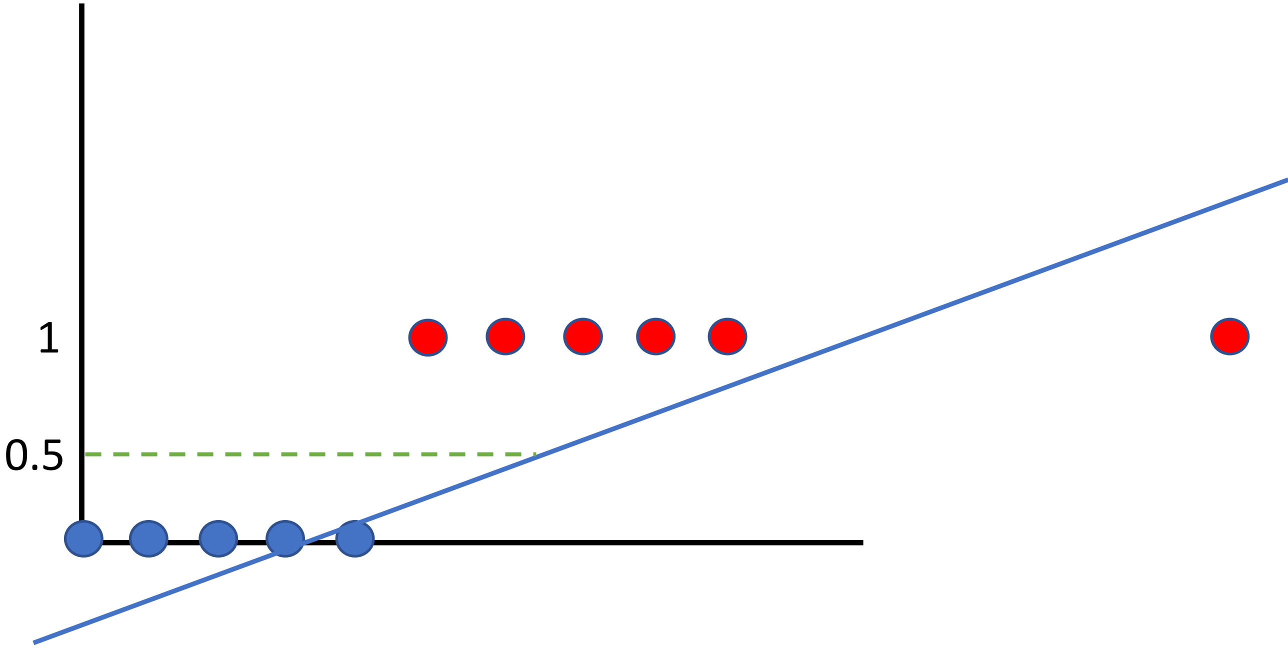
x_1	y
	1
	1
	0
	1
	0
	1

Usually a Bad Idea

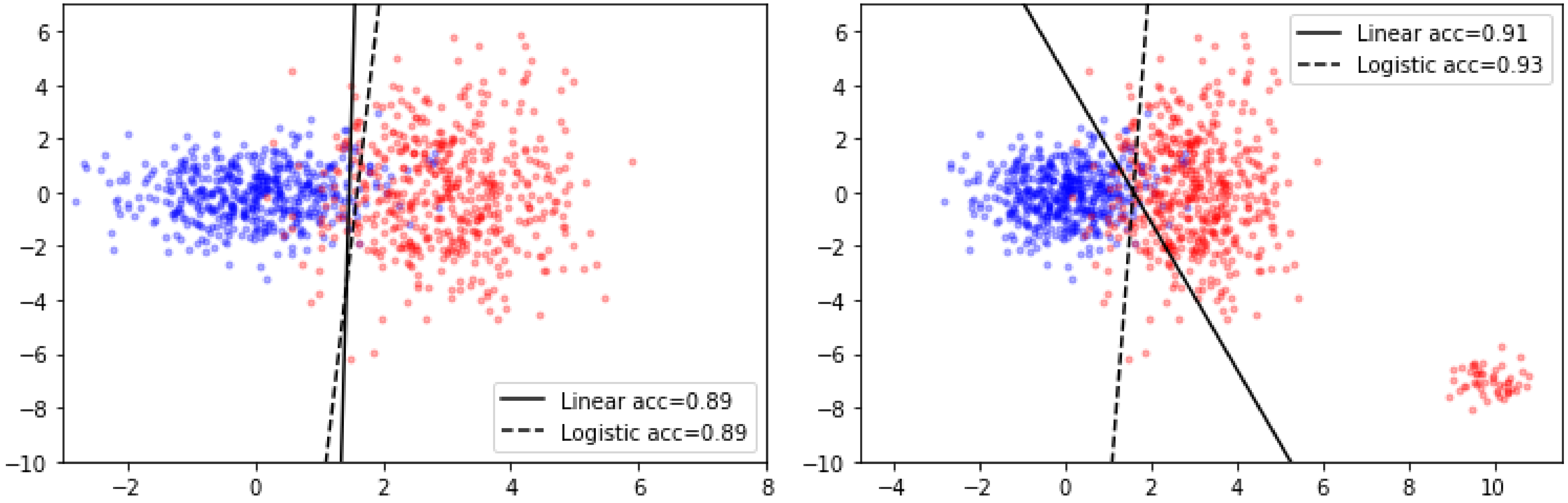
Works ok



Doesn't Work



Usually a Bad Idea



Logistic Regression

- We will try to predict the probability the instance belongs to class 1.

- Hypothesis function :

$$h_\theta(x) = P(1|x)$$

- How do we go from score to probability?

x_1	y
	1
	1
	0
	1
	0
	1

Score to Probability

- Sigmoid function for logistic regression:

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

Logistic Regression Function

- Sigmoid Function:

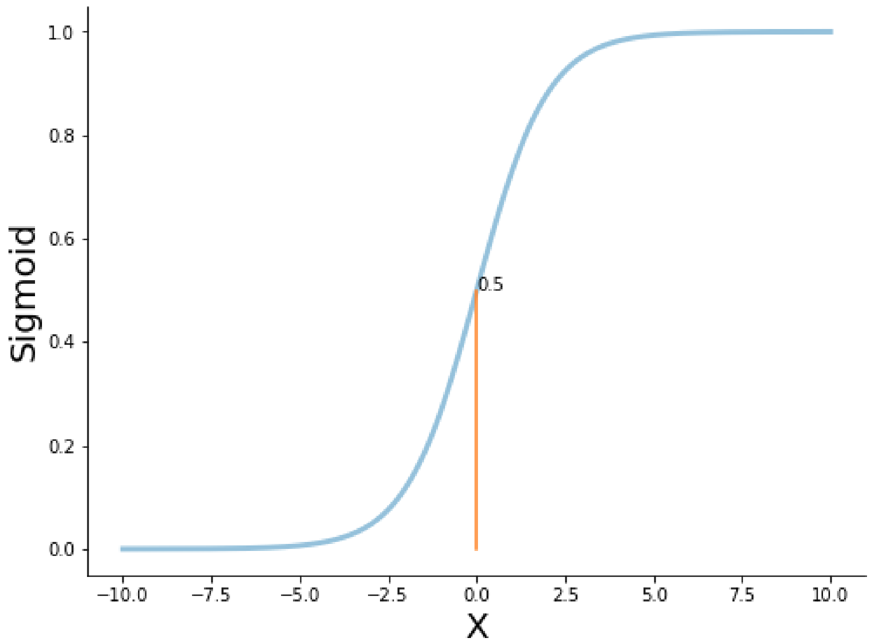
$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

Score	-∞	-2	0	+2	+∞
Sigmoid (Score)	$\frac{1}{1+e^{\infty}}$ $= 0$	$\frac{1}{1+e^{2}}$ $= 0.12$	$\frac{1}{1+e^{0}}$ $= 0.5$	$\frac{1}{1+e^{-2}}$ $= 0.88$	$\frac{1}{1+e^{-\infty}}$ $= 1$
	?	?	?	?	?

Logistic Regression Function

- Sigmoid Function:

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$



Logistic Regression Classification

- What if instead of predicting a continues value we try to predict a class?

- Hypothesis function :

$$h_\theta(x) = S(\theta^T x)$$

- Prediction :

$$if\ h_\theta(x) > 0.5\ then\ 1$$
$$else\ 0$$

x_1	y
	1
	1
	0
	1
	0
	1

Logistic Regression - Loss

- We have data and we defined the hypothesis function $h_\theta(x)$.
- We need to find a cost function so we can improve $h_\theta(x)$.
- We will use Maximum likelihood to find the appropriate cost function.

Logistic Regression

- $P(y|x,\theta) = \left(h_\theta(x)\right)^y \cdot \left(1 - h_\theta(x)\right)^{1-y}$

- Remember :

$$h_\theta(x) = \text{probability x belongs to class 1}$$

- And so we get :
  - $P(0|x,\theta) = 1 - h_\theta(x)$
  - $P(1|x,\theta) = h_\theta(x)$

Logistic Regression - ML

$$P(y|x,\theta) = \left(h_\theta(x)\right)^y \cdot \left(1 - h_\theta(x)\right)^{1-y}$$

Assuming independent instances

$$P(D|\theta) = \prod_{d=1}^{m} P\left(y^{(d)} \,\middle|\, x^{(d)}, \theta\right) =$$

$$\prod_{d=1}^{m} \left(h_\theta\left(x^{(d)}\right)\right)^{\boldsymbol{y}^{(d)}} \cdot \left(1 - h_\theta\left(x^{(d)}\right)\right)^{\mathbf{1}-\boldsymbol{y}^{(d)}}$$

Logistic Regression - ML

$$\underset{\theta}{argmax} \prod_{d=1}^{m} \left(h_\theta\left(x^{(d)}\right)\right)^{y^{(d)}} \cdot \left(1 - h_\theta\left(x^{(d)}\right)\right)^{1-y^{(d)}} =$$

$$\underset{\theta}{argmax} \ln\left(\prod_{d=1}^{m} \left(h_\theta\left(x^{(d)}\right)\right)^{y^{(d)}} \cdot \left(1 - h_\theta\left(x^{(d)}\right)\right)^{1-y^{(d)}}\right) =$$

$$\underset{\theta}{argmax} \sum_{d=1}^{m} \ln\left(\left(h_\theta\left(x^{(d)}\right)\right)^{y^{(d)}}\right) + \ln\left(\left(1 - h_\theta\left(x^{(d)}\right)\right)^{1-y^{(d)}}\right) =$$

$$\underset{\theta}{argmax} \sum_{d=1}^{m} y^{(d)} \cdot \ln\left(h_\theta\left(x^{(d)}\right)\right) + \left(1 - y^{(d)}\right) \cdot \ln\left(1 - h_\theta\left(x^{(d)}\right)\right) =$$

$$\underset{\theta}{argmin} \sum_{d=1}^{m} -y^{(d)} \cdot \ln\left(h_\theta\left(x^{(d)}\right)\right) - \left(1 - y^{(d)}\right) \cdot \ln\left(1 - h_\theta\left(x^{(d)}\right)\right)$$

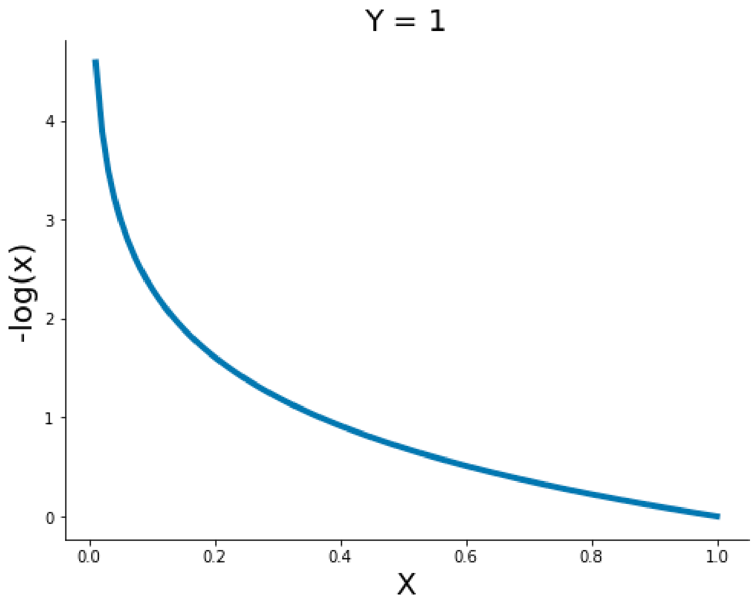
Logistic Regression Cost Function

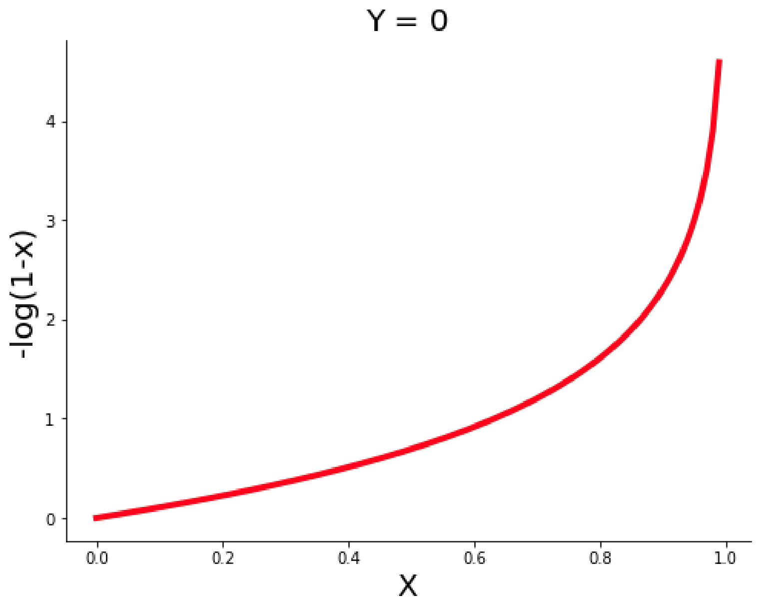
Binary Cross Entropy

$$\frac{1}{m}\sum_{d=1}^{m} -y^{(d)} \cdot \ln\left(h_\theta\left(x^{(d)}\right)\right) - \left(1 - y^{(d)}\right) \cdot \ln\left(1 - h_\theta\left(x^{(d)}\right)\right)$$

Cost Function Intuition

- $$Cost(x, \theta) = \begin{cases} -\log(h_\theta(x)) & y = 1 \\ -\log(1 - h_\theta(x)) & y = 0 \end{cases}$$





Learn Logistic Regression

$$cost(\vec{\theta}) = -\sum_{d=1}^{m} y^{(d)} \ln\left(S\left(\theta, \vec{x}^{(d)}\right)\right) + \left(1 - y^{(d)}\right) \ln\left(1 - S\left(\vec{\theta}, \vec{x}^{(d)}\right)\right)$$

$$\ln\left(S\left(\vec{\theta}, \vec{x}^{(d)}\right)\right) = \ln\left(\frac{1}{1 + e^{-\theta^T x^{(d)}}}\right) = -\ln\left(1 + e^{-\theta^T x^{(d)}}\right)$$

$$\ln\left(1 - S\left(\vec{\theta}, \vec{x}^{(d)}\right)\right) = \ln\left(1 - \frac{1}{1 + e^{-\theta^T x^{(d)}}}\right)$$

$$= \ln\left(\frac{1 + e^{-\theta^T x^{(d)}}}{1 + e^{-\theta^T x^{(d)}}} - \frac{1}{1 + e^{-\theta^T x^{(d)}}}\right) = \ln\left(\frac{e^{-\theta^T x^{(d)}}}{1 + e^{-\theta^T x^{(d)}}}\right)$$

$$= -\theta^T x^{(d)} - \ln\left(1 + e^{-\theta^T x^{(d)}}\right)$$

Learn Logistic Regression

$$cost(\vec{\theta}) = -\sum_{d=1}^{m} -y^{(d)} \ln\left(1 + e^{-\theta^T x^{(d)}}\right) + \left(1 - y^{(d)}\right)\left(-\theta^T x^{(d)} - \ln\left(1 + e^{-\theta^T x^{(d)}}\right)\right)$$

$$= -\sum_{d=1}^{m} y^{(d)}\theta^T x^{(d)} - \theta x^{(d)} - \ln\left(1 + e^{-\theta^T x^{(d)}}\right) =$$

$$-\sum_{d=1}^{m} y^{(d)}\theta^T x^{(d)} - \theta^T x^{(d)} - \ln\left(\frac{1 + e^{\theta^T x^{(d)}}}{e^{\theta^T x^{(d)}}}\right)$$

$$-\sum_{d=1}^{m} y^{(d)}\theta^T x^{(d)} - \ln\left(1 + e^{\theta^T x^{(d)}}\right)$$

Learn Logistic Regression

$$cost(\vec{\theta}) = -\sum_{d=1}^{m} y^{(d)}\theta^T x^{(d)} - \ln\left(1 + e^{\theta^T x^{(d)}}\right)$$

- The derivation for each instance will give:

$$\frac{\partial}{\partial\theta_i} cost(\vec{x},\vec{\theta}) = -\left(y - S(\vec{\theta},\vec{x})\right) x_i$$

  - i – is the i feature

- And for all m training data:

$$\frac{\partial}{\partial\theta_i} cost(\vec{\theta}) = \sum_{d=1}^{m} \left(S(\vec{\theta},\vec{x}^{(d)}) - y^{(d)}\right) x_i^{(d)}$$

- You can now use gradient descent for finding the best $\vec{\theta}$

Logistic Regression Summary

- Hypothesis function :

$$h_\theta(x) = S(\theta^T x)$$

- Cost function:

$$J(\theta) = \frac{1}{m}\sum_{d=1}^{m} -y^{(d)} \cdot \ln\left(h_\theta\left(x^{(d)}\right)\right) - \left(1 - y^{(d)}\right) \cdot \ln\left(1 - h_\theta\left(x^{(d)}\right)\right)$$

- Goal :

$$\underset{\theta}{min}\, J(\theta)$$

Multi-class classification

- How can we convert 2 classes linear separator to solve multi-class problem?
- We will create a predictor for each class (one vs. all)
- Predict i if $f_i(x) > f_j(x)\ \forall j \neq i$

Questions

?