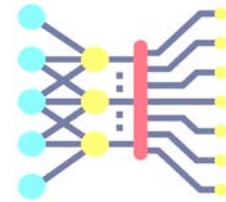


CS60010: Deep Learning

Spring 2021

Sudeshna Sarkar and Abir Das

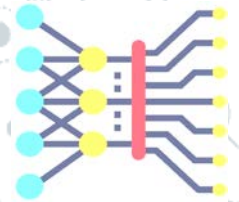
Module 2 Part 2

Linear Models for Classification

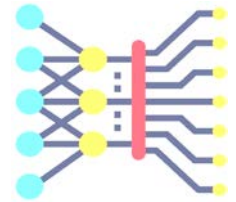
Sudeshna Sarkar

18 Jan 2021

Announcements



- Class Test 1 on 19th Jan 2021 (tomorrow)
- Assignment 1 due 22nd Jan 12 pm



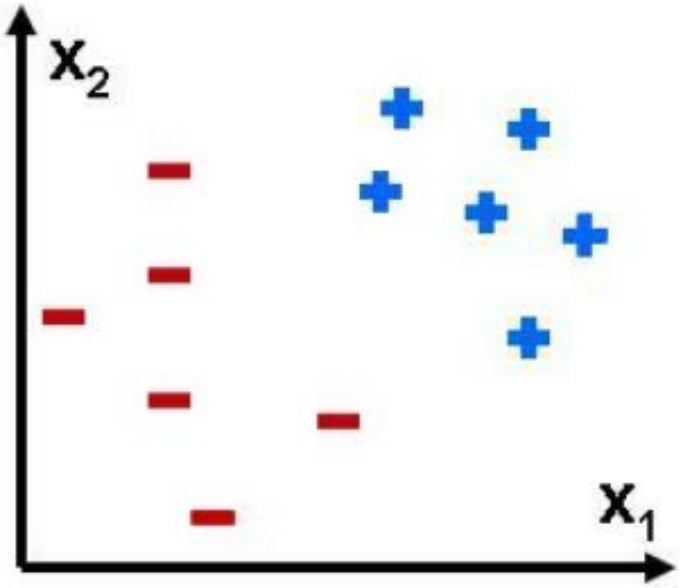
ML Background and Linear Models

Logistic Regression

Classification

A binary classifier is a mapping from $R^d \rightarrow \{-1, +1\}$

$$x \rightarrow \boxed{h} \rightarrow y$$



Training data set

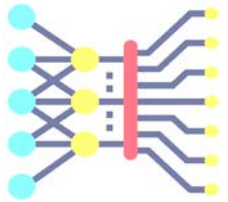
$$\mathcal{D}_m = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$$

- Assume that each $x^{(i)}$ is a $d \times 1$ column vector
- Given a training set $\mathcal{D}_m$ and a classifier h, we can define the training error of h to be

$$\varepsilon_m(h) = \frac{1}{m} \sum_{i=1}^{m} \begin{cases} 1 & h(x^{(i)}) \neq y^{(i)} \\ 0 & \text{otherwise} \end{cases}$$

- For now, we will try to find a classifier with small training error (and hope it generalizes well to new data, and has a small test error

Learning algorithm



- A hypothesis class $\mathcal{H}$ is a set (finite or infinite) of possible classifiers, each of which represents a mapping from $R^d \rightarrow \{-1, +1\}$
- A learning algorithm is a procedure that takes a data set $\mathcal{D}_n$ as input and returns an element $h \in \mathcal{H}$

$$x \rightarrow \boxed{\text{learning alg } (\mathcal{H})} \rightarrow y$$

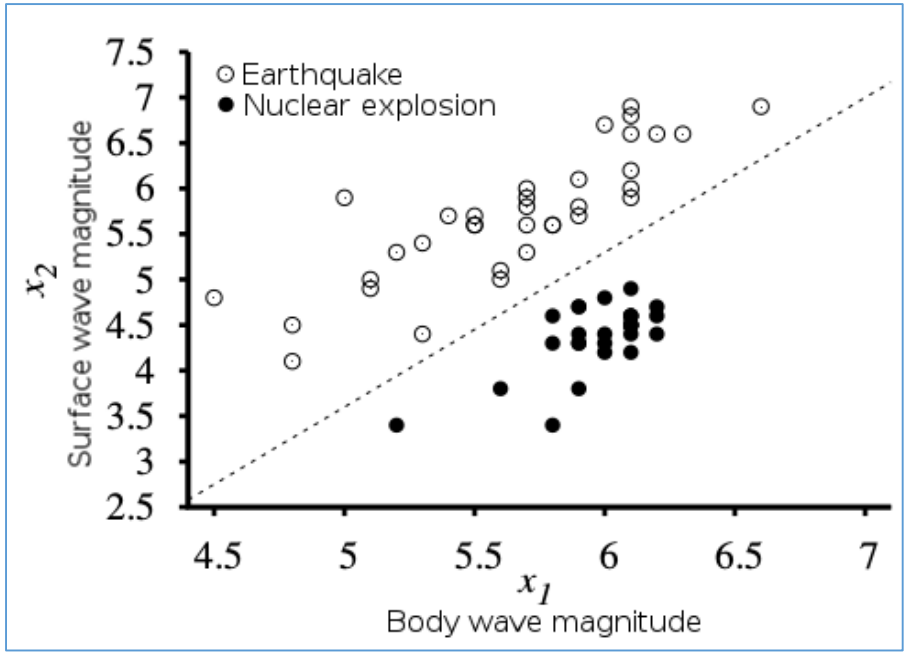
- Choice of $\mathcal{H}$ so as to get low test error

Hypothesis class :Linear classifiers

- A linear classifier in d dimensions is defined by
  - A vector of parameters $\theta \, \epsilon \, R^d$ and scalar $\theta_0 \epsilon \mathcal{R}$

    Assume a d × 1 column vector

$$h(x;\theta,\theta_0) = \text{sign}(\theta^T x + \theta_0) = \begin{cases} +1 & if\, \theta^T x + \theta_0 > 0 \\ -1 & \text{otherwise} \end{cases}$$



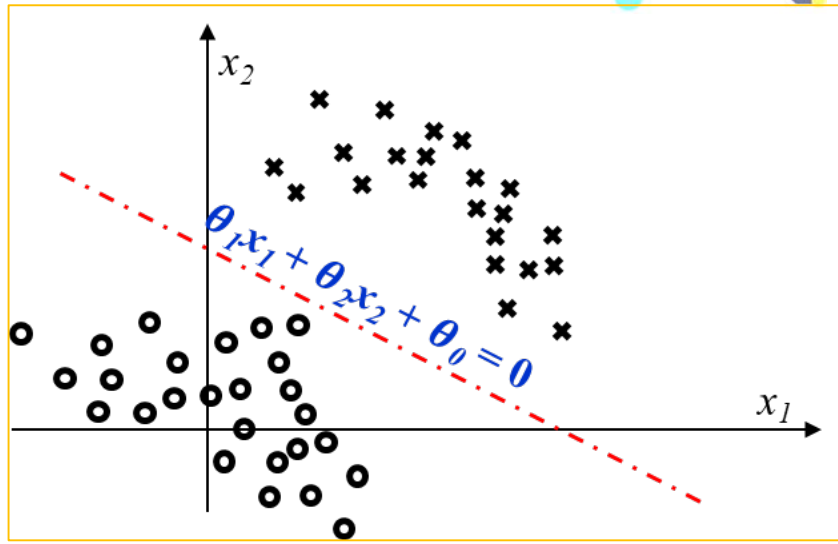
θ, θ_0 specifies a hyperplane (decision boundary) that divides the instance space into two half-spaces.

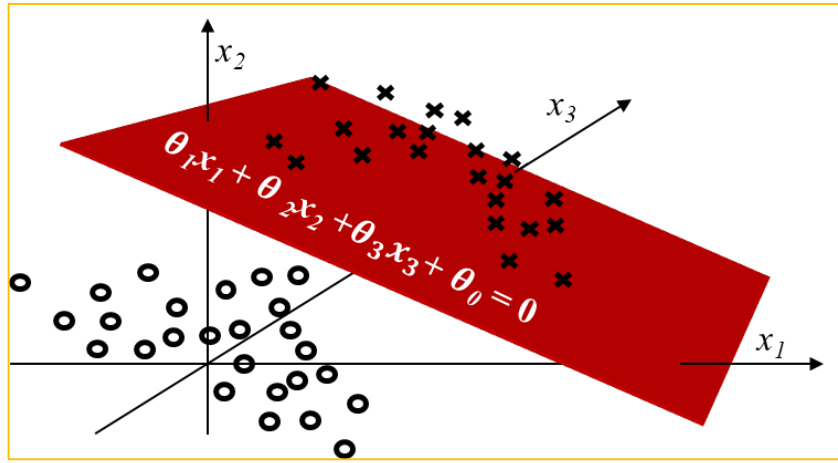
Linear classifiers

$$h(x;\theta,\theta_0) = sign(\theta^T x + \theta_0)$$

$$= \begin{cases} +1 & \text{if } \theta^T x + \theta_0 > 0 \\ -1 & \text{otherwise} \end{cases}$$

- θ, θ_0 specifies a hyperplane that divides the instance space into two half-spaces.
  - The one that is on the same side as the normal vector is the positive half-space, and we classify all points in that space as positive.
  - The half-space on the other side is negative and all points in it are classified as negative.





Linear Classifier with Hard Threshold

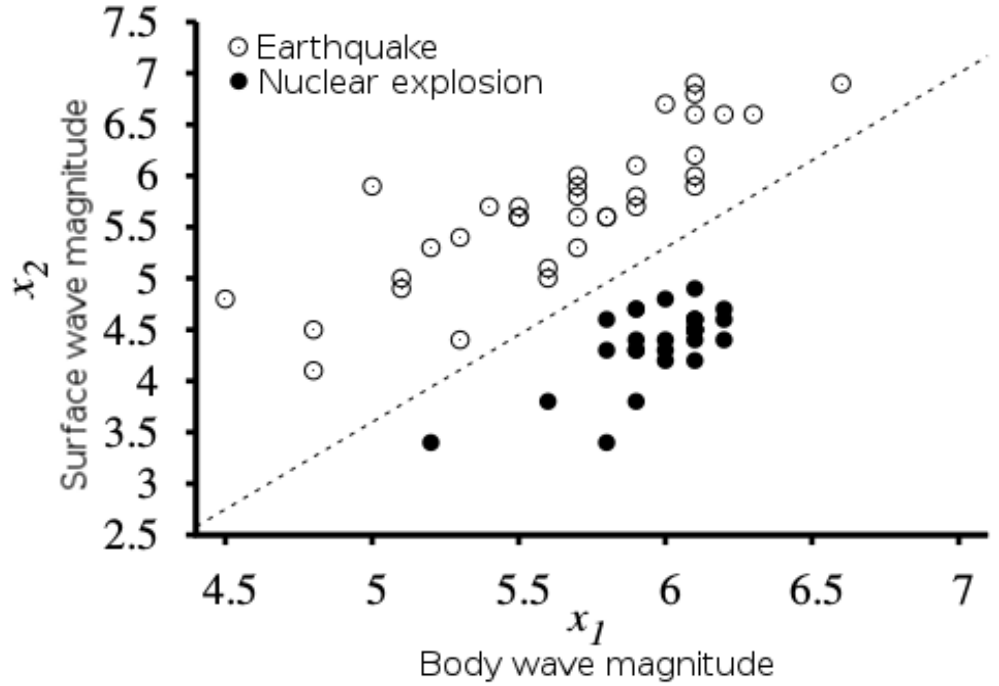
- The linear separator in the associated fig is given by

$$x_2 = 1.7x_1 - 4.9$$

$$\rightarrow -4.9 + 1.7x_1 - x_2 = 0$$

$$\rightarrow \begin{bmatrix} -4.9 & 1.7 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = 0$$

$$\boldsymbol{\theta}^T \mathbf{x} = 0$$



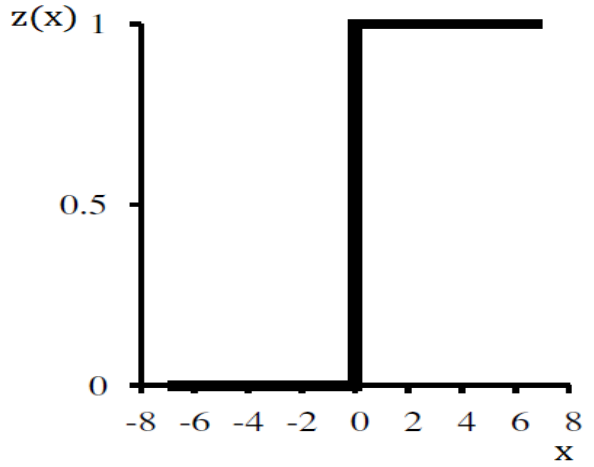
Classification Rule:

$$y(x) = \begin{cases} +1 & if\, \theta^T x > 0 \\ -1 & \text{otherwise} \end{cases}$$

Linear Classifier with Hard Threshold

Classification Rule:

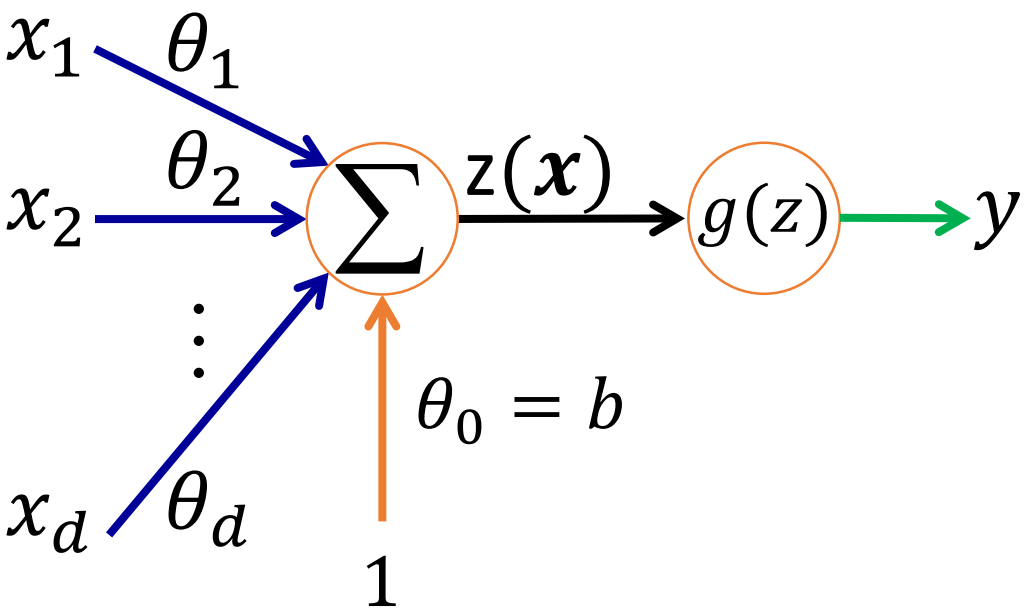
$$y(x) = \begin{cases} +1 & \text{if } \theta^T x > 0 \\ -1 & \text{otherwise} \end{cases}$$



We can think y as the result of passing the linear function $\theta^T x$ through a threshold function.

- Find the θ which minimizes classification error on the training set.
- We cannot use gradient descent at all points for the above threshold function

Perceptron



$$\boldsymbol{\theta} = [\theta_1\ \theta_2\ \dots\ \theta_d]^T \text{ and } \boldsymbol{x} = [x_1\ x_2\ \dots\ x_d]^T$$

$$z(\boldsymbol{x}) = \theta_0 + \sum_{i=1}^{d} \theta_i x_i = [\boldsymbol{\theta}^T\ b]\begin{bmatrix}\boldsymbol{x}\\ 1\end{bmatrix}$$

$$y = g(z(\boldsymbol{x}))$$

Terminologies

$\boldsymbol{x}$: input, $\boldsymbol{\theta}$: weights, $\boldsymbol{b}$: bias

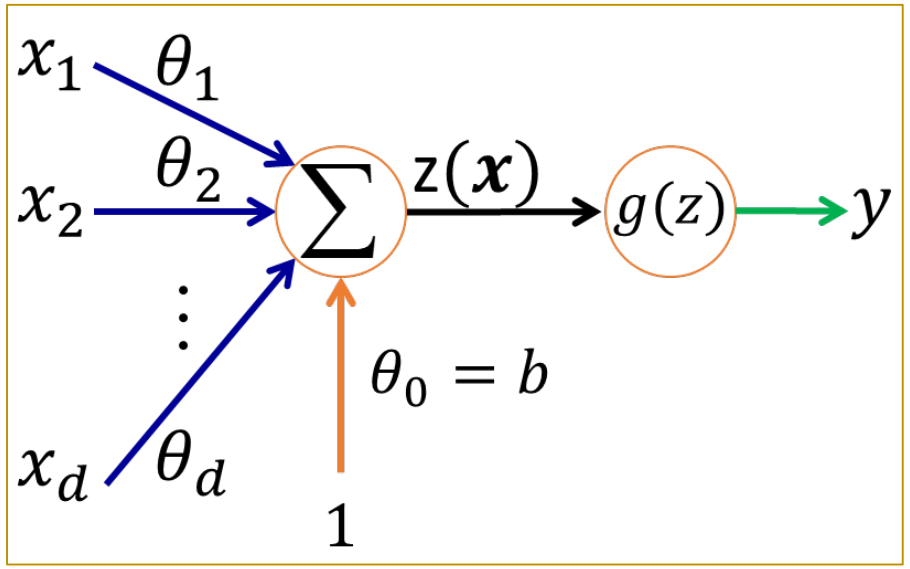
z: pre-activation (input activation)

g: activation function

y: activation (output activation)

Perceptron

$\boldsymbol{x} \in \mathcal{R}^d$ and $y \in \{0, 1\}$ for Binary Classification

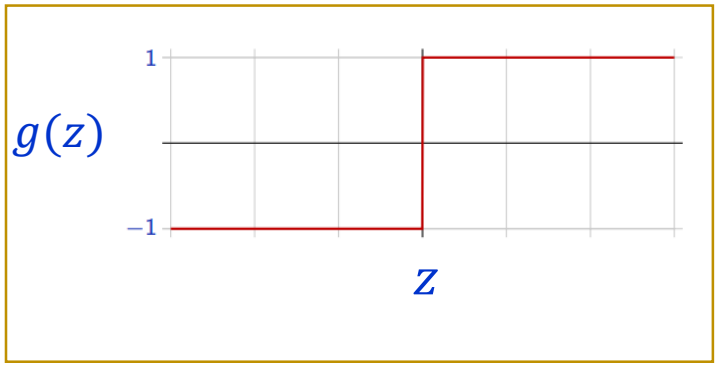


$$g(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

(Rosenblatt, 1957)

Or, the response may be taken as $y \in \{-1, 1\}$

$$g(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0 \end{cases}$$



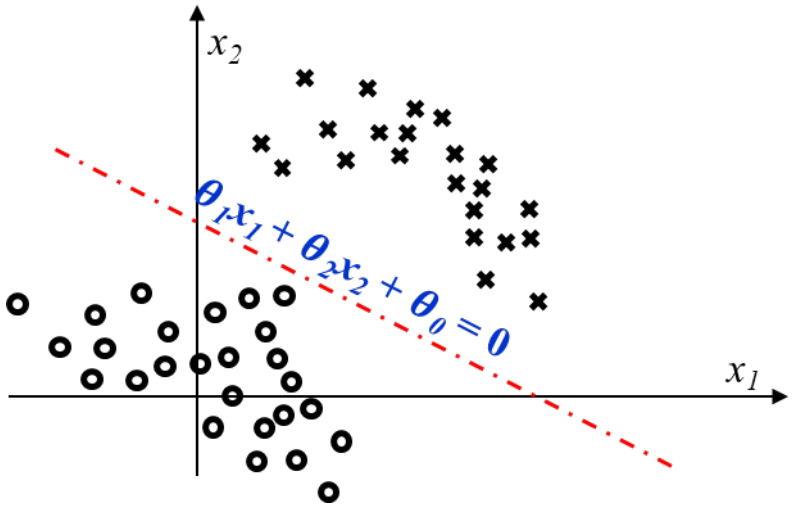
The perceptron classification rule, thus, translates to

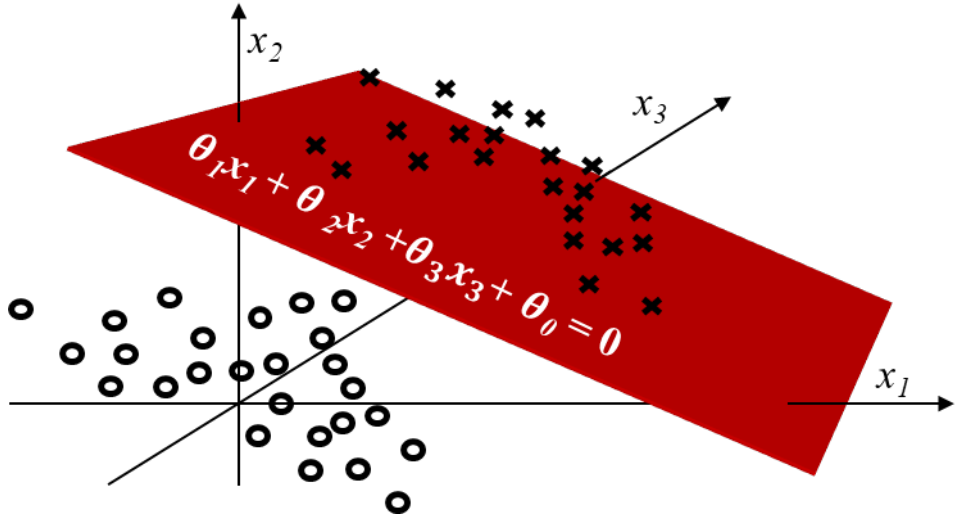
$$y = \begin{cases} 1, & \boldsymbol{\theta}^T \boldsymbol{x} + b \geq 0 \\ -1, & \boldsymbol{\theta}^T \boldsymbol{x} + b < 0 \end{cases}$$

$\boldsymbol{\theta}^T \boldsymbol{x} + b = 0$ represents a hyperplane.

Perceptron (Geometrically)

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$





- $\boldsymbol{\theta}^T \mathbf{x}$ is the (signed) distance of point **x** to hyperplane
  - Example: http://mathinsight.org/distance_point_plane

Perceptron Learning Algorithm

Training Set: $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

$x^{(i)} \in R^d;\ y^{(i)} \in \{-1, +1\}$

1. $t \leftarrow 1$; $\theta^{(t)} = \mathbf{0}$
2. // Loop until all examples are correctly classified

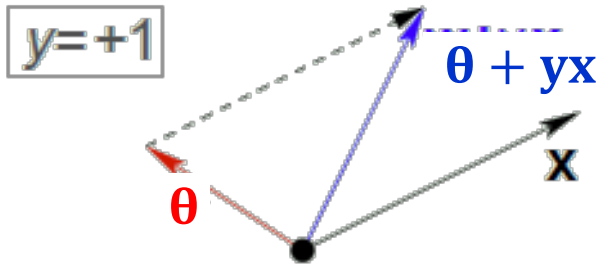
While exists i such that $x^{(m)}$ is not correctly classified

Pick a j s.t. $y^{(j)}.\langle \theta^{(t)}, x^{(j)} \rangle \leq 0$) then

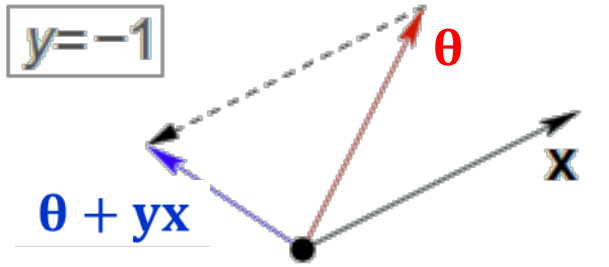
$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + y^{(j)} x^{(j)}$

$t \leftarrow t + 1$

3. Output $\boldsymbol{\theta}^{(t)}$



θ +x pushes vector θ towards x



θ −x pushes vector θ away from x

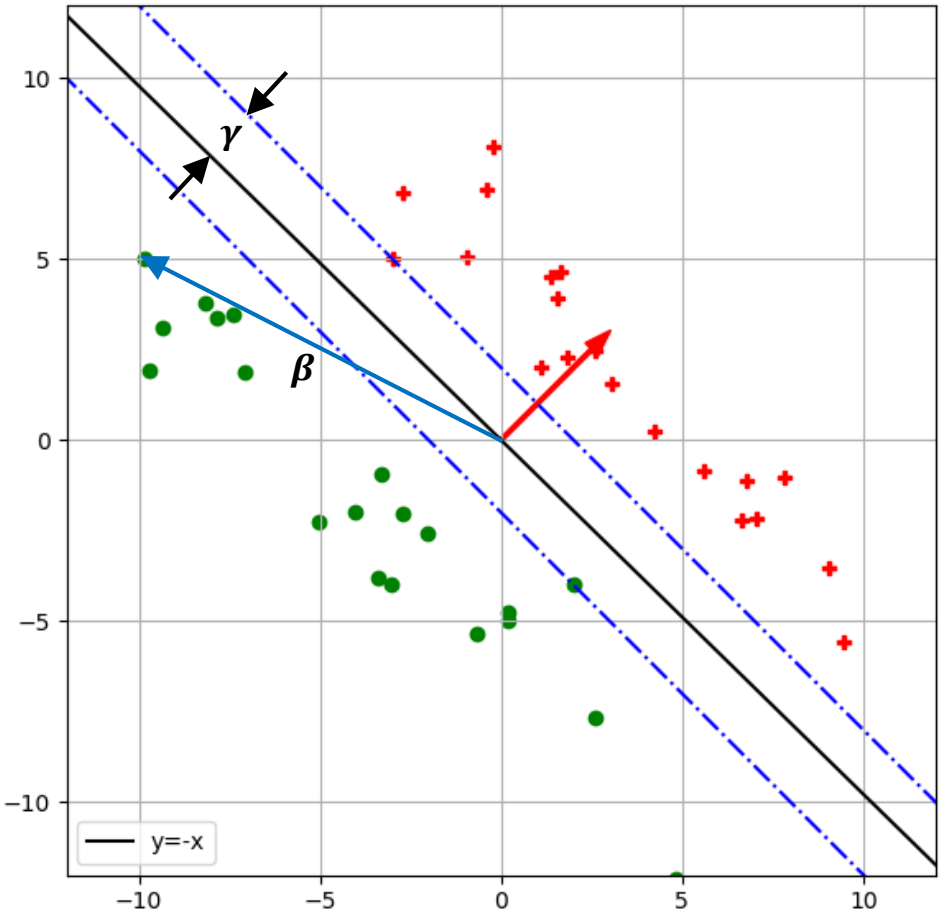
Perceptron Learning Algorithm

1. $t \leftarrow 1$; $\theta^{(t)} = \mathbf{0}$
2. While there exists i such that $x^{(m)}$ is not correctly classified

   Pick a j s.t. $y^{(j)}.\langle \theta^{(t)}, x^{(j)} \rangle \leq 0$)

   $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + y^{(j)}x^{(j)}$

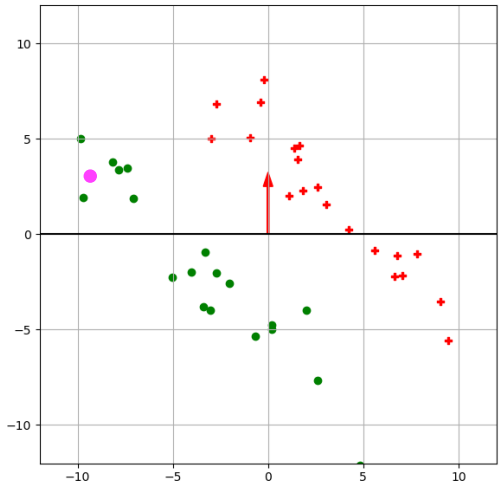
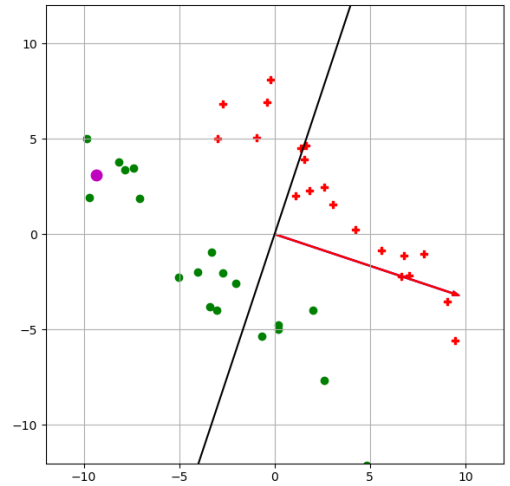
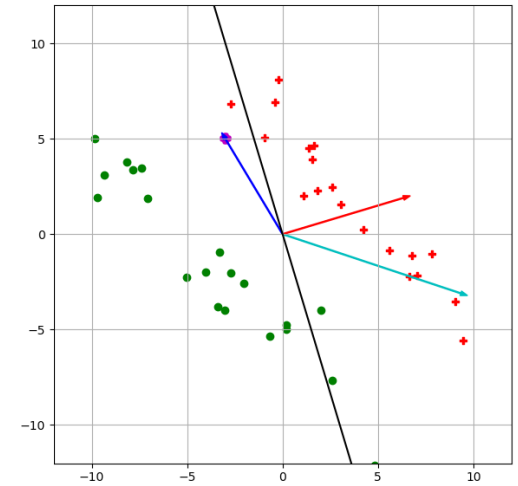
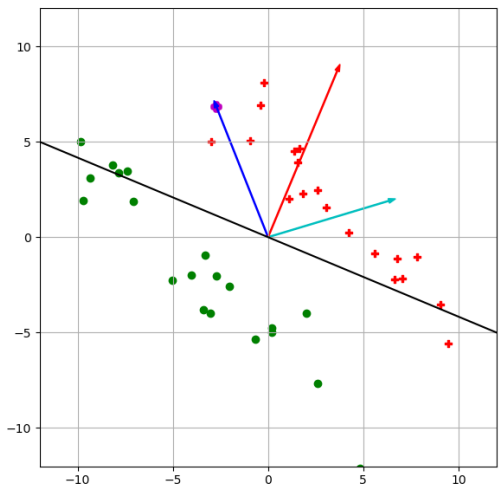
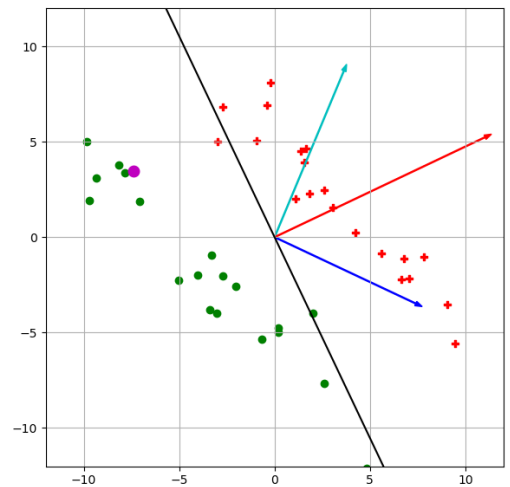
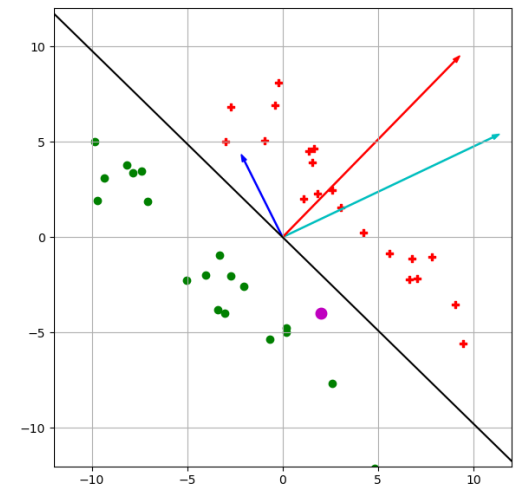
   $t \leftarrow t + 1$
3. Output $\boldsymbol{\theta}^{(t)}$



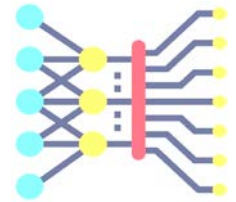
If such a separating hyperplane exists, then the data is known to be linearly separable.

<u>Convergence Theorem</u>

For a finite and linearly separable set of data, the perceptron learning algorithm will find a linear separator in at most $\frac{\beta^2}{\gamma^2}$ iterations where the maximum length of any data point is β and γ is the maximum margin of the linear separators.

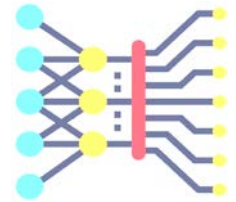







Perceptron Rule



- Perceptron Learning Rule can find a linear separator given the data is *linearly separable*.
- For data that are not linearly separable, the Perceptron algorithm fails.

Linear Classifiers by Gradient Descent



For a gradient based optimization approach, we need to approximate hard threshold function with something smooth.

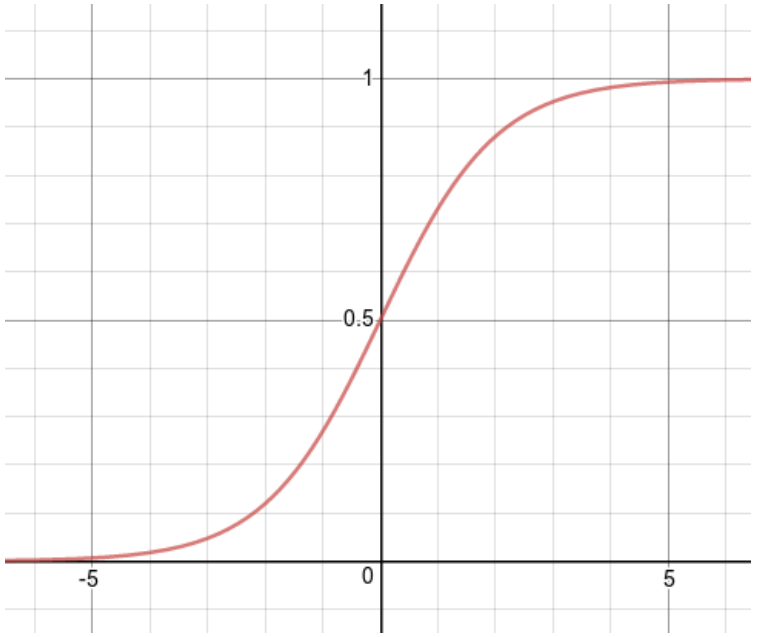
- Logistic regression classifier

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

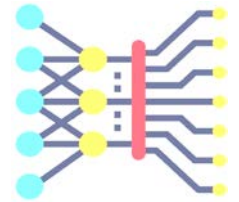
$$y = \sigma(h_\theta(\mathbf{x})) = \sigma(\boldsymbol{\theta}^T \mathbf{x})$$

$$z = \boldsymbol{\theta}^T \mathbf{x}$$

$\sigma(z)$



Likelihood Function for Logistic Regression



- The probability that an example belongs to class 1 is

$$P\left(y^{(i)} = 1 | x^{(i)}; \boldsymbol{\theta}\right) = \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)$$

Thus $$P\left(y^{(i)} = 0 | x^{(i)}; \boldsymbol{\theta}\right) = 1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)$$

Thus

$$P\left(y^{(i)} | x^{(i)}; \boldsymbol{\theta}\right)$$

$$= \left(\sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right)^{y^{(i)}} \left(1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right)^{1-y^{(i)}}$$

$$z^{(i)} = \theta^T x^{(i)}$$

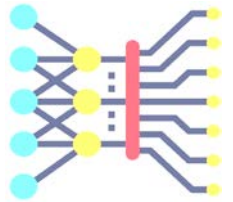
$$P\left(y^{(i)} = 1 | x^{(i)}\right) \sigma\left(z^{(i)}\right)$$

Thus $$P\left(y^{(i)} = 0 | x^{(i)}\right) = 1 - \sigma\left(z^{(i)}\right)$$

$$P\left(y^{(i)} | x^{(i)}\right)$$

$$= \left(\sigma\left(z^{(i)}\right)\right)^{y^{(i)}} \left(1 - \sigma\left(z^{(i)}\right)\right)^{1-y^{(i)}}$$

Maximum Likelihood Estimation of Logistic Regression



The probability that an example belongs to class 1 is $P\left(y^{(i)}=1|x^{(i)};\boldsymbol{\theta}\right)=\sigma\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)$

Thus $P\left(y^{(i)}=0|x^{(i)};\boldsymbol{\theta}\right)=1-\sigma\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)$

Thus $P\left(y^{(i)}|x^{(i)};\boldsymbol{\theta}\right)=\left(\sigma\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)\right)^{y^{(i)}}\left(1-\sigma\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)\right)^{1-y^{(i)}}$

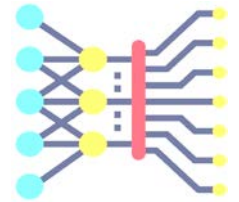
The joint probability of all the labels

$$\prod_{i=1}^{m}\left(\sigma\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)\right)^{y^{(i)}}\left(1-\sigma\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)\right)^{1-y^{(i)}}$$

So the log likelihood for logistic regression is given by

$$l(\theta)=\sum_{i=1}^{m}y^{(i)}log\left(\sigma\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)\right)+\left(1-y^{(i)}\right)log\left(1-\sigma\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)\right)$$

Maximum Likelihood Estimation of Logistic Regression



Derivative of log likelihood w.r.t. one component of $\boldsymbol{\theta}$

$$\frac{\partial l(\boldsymbol{\theta})}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \sum_{i=1}^{m} y^{(i)} \log \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right)$$

derivative of sum of terms

$$= \sum_{i-1}^{m} \left[\frac{y^{(i)}}{\sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)} - \frac{1 - y^{(i)}}{1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)}\right] \frac{\partial}{\partial \theta_j} \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)$$

derivative of log f (x)

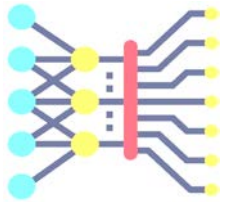
$$= \sum_{i=1}^{m} \left[\frac{y^{(i)}}{\sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)} - \frac{1 - y^{(i)}}{1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)}\right] \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\left(1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right) \mathbf{x}_j^{(i)}$$

chain rule + derivative of σ

$$= \sum_{i=1}^{m} \left[\frac{y^{(i)} - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)}{\sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\left(1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right)}\right] \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\left(1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right) \mathbf{x}_j^{(i)}$$

$$= \sum_{i=1}^{m} \left[y^{(i)} - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right] \mathbf{x}_j^{(i)} \qquad (12)$$

Calculating derivatives



- Since the likelihood function is a sum over all of the data, and in calculus the derivative of a sum is the sum of derivatives, we can focus on computing the derivative of one example. The gradient of theta is simply the sum of this term for each training data point.
- The derivative of gradient for one data point (x, y):
- $p = \sigma(\theta^T x)$
- $z = \theta^T x$

$$\frac{\partial l(\theta)}{\partial \theta_j} = \frac{\partial l(\theta)}{\partial p} \cdot \frac{\partial p}{\partial \theta_j}$$

$$= \frac{\partial l(\theta)}{\partial p} \cdot \frac{\partial p}{\partial z} \cdot \frac{\partial z}{\partial \theta_j}$$

Derivative of logistic function

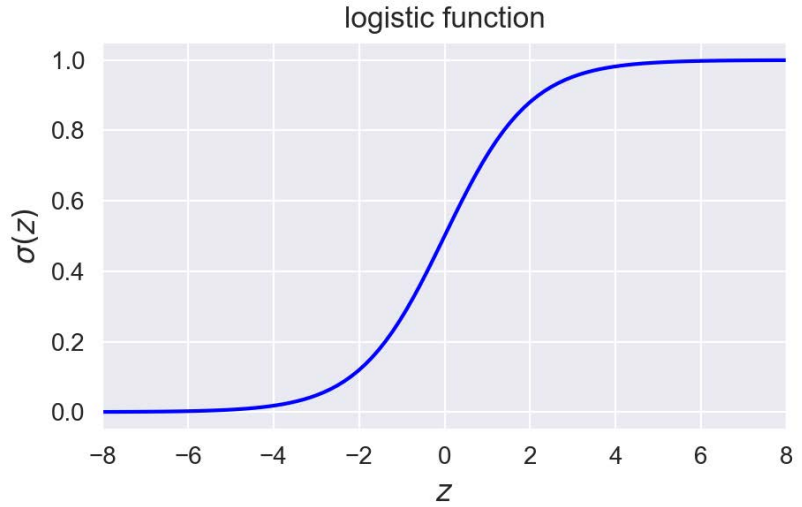
$$\frac{\partial}{\partial z}\sigma(z) = \frac{\partial \frac{1}{1+e^{-z}}}{\partial z}$$

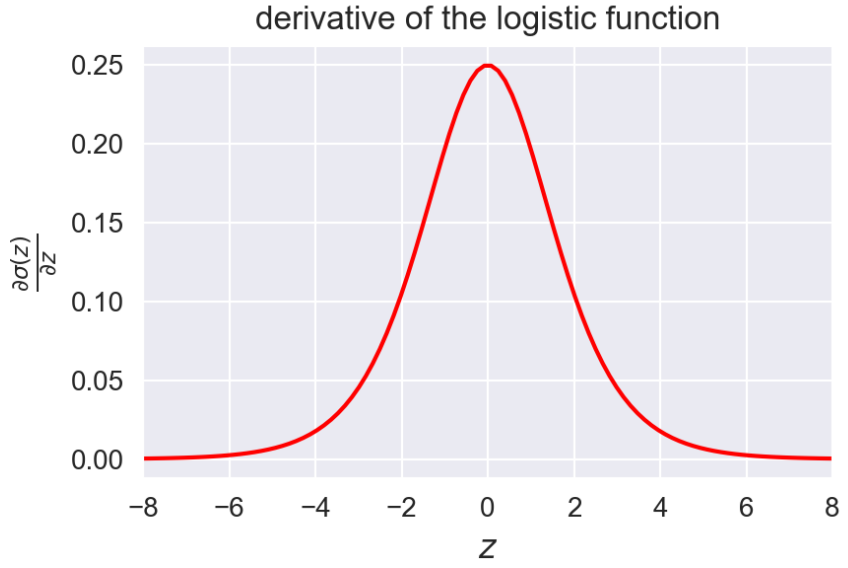
$$= \frac{-1}{(1+e^{-z})^2} \cdot e^{-z} \cdot -1$$

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

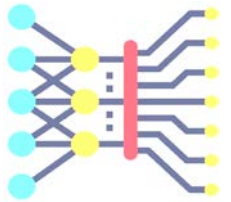
$$\sigma(z)\big(1-\sigma(z)\big)$$

$$= y(1-y)$$





Calculating derivatives



$$\frac{\partial l(\theta)}{\partial \theta_j} = \frac{\partial l(\theta)}{\partial p} \cdot \frac{\partial p}{\partial z} \cdot \frac{\partial z}{\partial \theta_j}$$

$$l(\theta) = y \log p + (1-y)\log(1-p)$$

$$\frac{\partial l(\theta)}{\partial p} = \frac{y}{p} - \frac{1-y}{1-p}$$

$$\frac{\partial p}{\partial z} = \sigma(z)\big(1-\sigma(z)\big) \qquad p = \sigma(z)$$

$$\frac{\partial z}{\partial \theta_j} = x_j \qquad z = \theta^T x$$

$$\frac{\partial l(\theta)}{\partial \theta_j} = \frac{\partial l(\theta)}{\partial p} \cdot \frac{\partial p}{\partial z} \cdot \frac{\partial z}{\partial \theta_j}$$

$$= \left[\frac{y}{p} - \frac{1-y}{1-p}\right] \cdot \sigma(z)\big(1-\sigma(z)\big) \cdot x_j$$

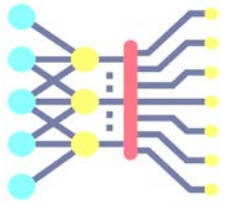
$$= \left[\frac{y}{p} - \frac{1-y}{1-p}\right] \cdot p(1-p) \cdot x_j$$

$$= [y(1-p) - p(1-y)] \cdot x_j$$

$$= [y-p] \cdot x_j$$

$$= [y - \sigma(\theta^T x)\,] \cdot x_j \qquad p = \sigma(\theta^T x)$$

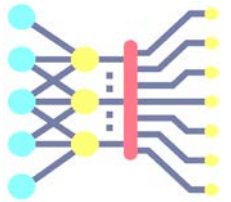
Gradient of Log Likelihood



$$\frac{\partial l(\theta)}{\partial \theta_j} = \sum_{i=1}^{m} \left[y^{(i)} - \sigma\left(\theta^T x^{(i)}\right) \right] x_j{}^{(i)}$$

- We need to chose the values of theta that maximize the log-likelihood.
- Unfortunately, if we try just setting the derivative equal to zero, there's no closed form for the maximum.
- However, we can find the best values of theta by using an optimization algorithm.

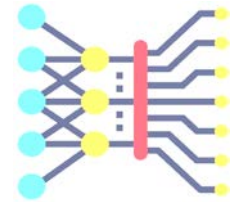
Gradient Ascent Optimization



$$\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \frac{\partial\, l\left(\theta^{\text{old}}\right)}{\partial \theta_j}$$

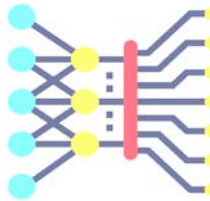
$$= \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{m} \left[y^{(i)} - \sigma\left(\theta^T x^{(i)}\right) \right] x_j{}^{(i)}$$

Cross-Entropy Loss Function



- We need a loss function $L(\hat{y}, y)$ that expresses, for an observation x, how close the classifier output $\hat{y}$ is to the correct output y (which is 0 or 1).
- A loss function that prefers the correct class labels of the training examples to be more likely. This is called conditional maximum likelihood estimation: we choose the parameters that maximize the log probability of the true y labels in the training data given the observations x.
- The resulting loss function is the negative log likelihood loss, generally called the cross-entropy loss
- Minimizing the negative of this function (minimizing the negative log likelihood) corresponds to maximizing the likelihood. This error function $L(\hat{y}, y)$ is typically known as the cross-entropy error function (also known as log-loss):

Cross entropy loss



$$p(y|x) = \hat{y}^{y}(1-\hat{y})^{1-y}$$

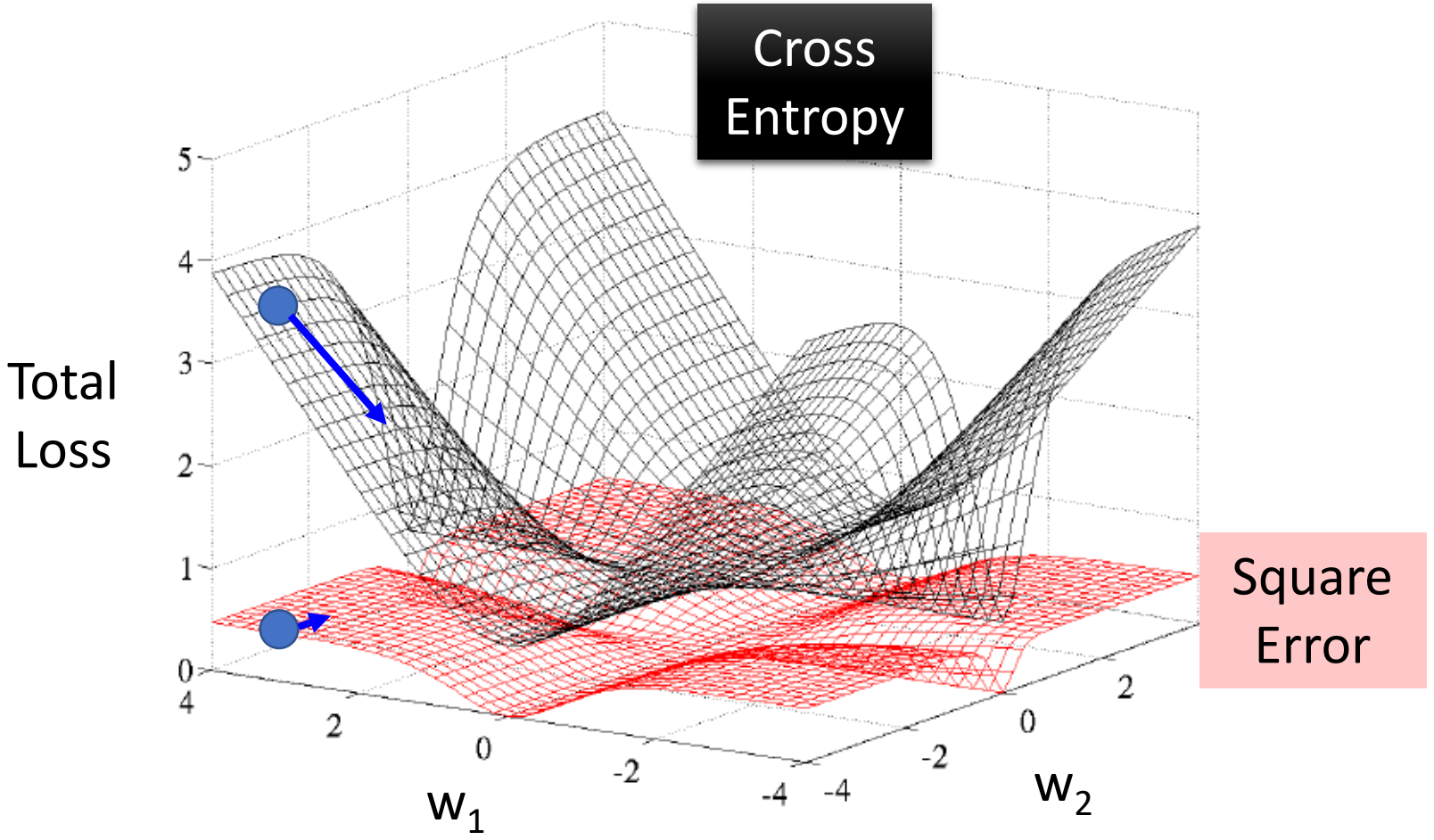
$$\log\big(p(y|x)\big) = y\log\hat{y} + (1-y)\log(1-\hat{y}) \quad \text{log likelihood}$$

cross-entropy loss:

$$L_{CE}(\hat{y}, y) = -\log\big(p(y|x)\big) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$$

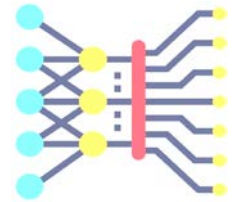
$$L_{CE}(\hat{y}, y) = -[y\log\sigma(\theta^{T}x) + (1-y)\log(1-\sigma(\theta^{T}x))]$$

Cross Entropy v.s. Square Error

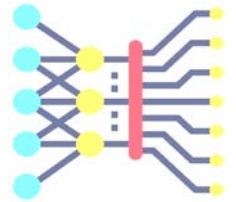


http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

Gradient Descent



Multiclass Classification



Multi-class Classification

C_1: θ_1, b_1 $\quad z_1 = \theta_1 \cdot x + b_1$

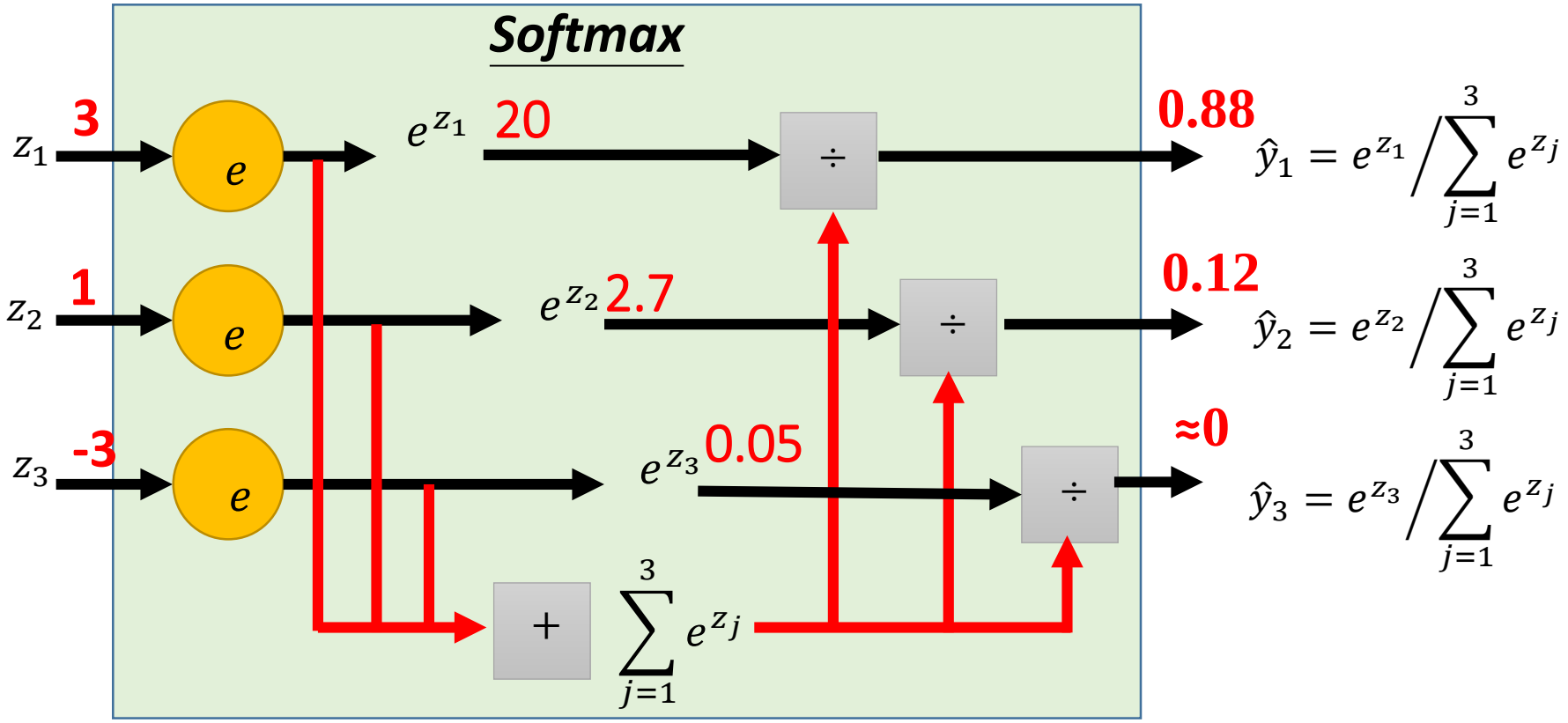
C_2: θ_2, b_2 $\quad z_2 = \theta_2 \cdot x + b_2$

C_3: θ_3, b_3 $\quad z_3 = \theta_3 \cdot x + b_3$

Probability:

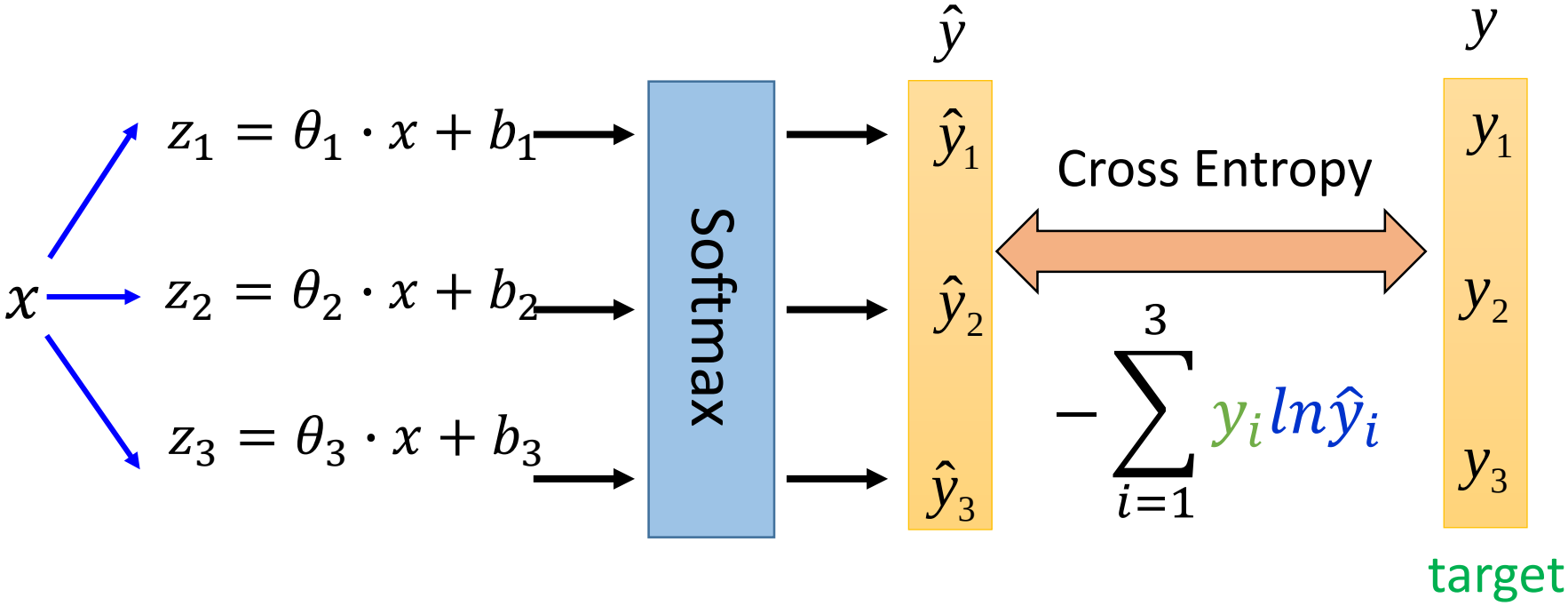
- $1 > \hat{y}_i > 0$
- $\sum_i \hat{y}_i = 1$

$y_i = P(C_i|x)$



[Bishop, P209-210]

Multi-class Classification



If x ∈ class 1

$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$-ln\hat{y}_1$

If x ∈ class 2

$$y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$-ln\hat{y}_2$

If x ∈ class 3

$$y = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$-ln\hat{y}_3$