Introduction to Computer Vision

Introduction to Deep Learning

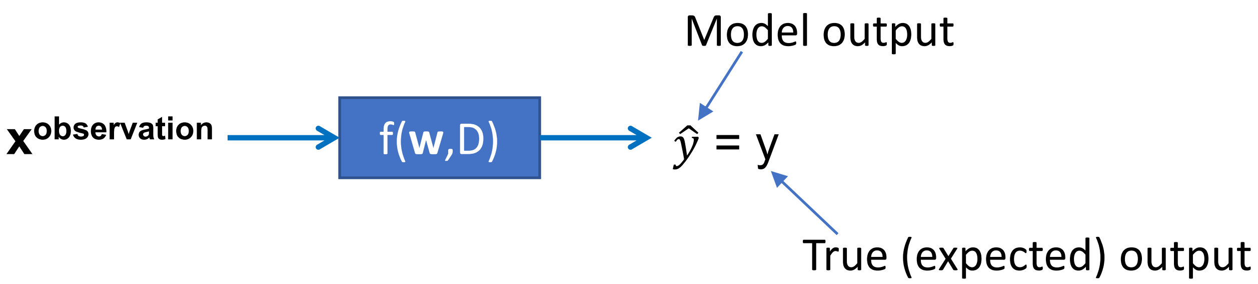
Dr. Sedat Ozer

Overview

- Linear Classifier,
- Logistic Regression,
- Loss Function for Logistic Regression
- Cost Function for Logistic Regression
- Gradient Descent Algorithm
- Computation Graph
- Derivatives for Logistic Regression
- Implementing Logistic Regression in Python
- Softmax Regression
- Neural Networks
- Fully Connected (FC) Neural Network

Introduction

Given observation (i.e., the data), derive a rule that can imitate the mechanism generating the observation: Model that mechanism as f() and then find (or fit) a function f() to mimic the system.



All the features from all the samples

Samples (observations)

$$\mathbf{X}_{nxm} = \begin{pmatrix} \mathbf{x}^1 \end{pmatrix} \begin{pmatrix} \cdots \\ \cdots \end{pmatrix} \begin{pmatrix} \mathbf{x}^m \end{pmatrix}$$

features

All the outputs

$$\mathbf{y}_{1xm} = \left(y^1, y^2, \ldots, y^m\right)$$

Each "column" in $\mathbf{X}_{nxm}$ represents a sample"

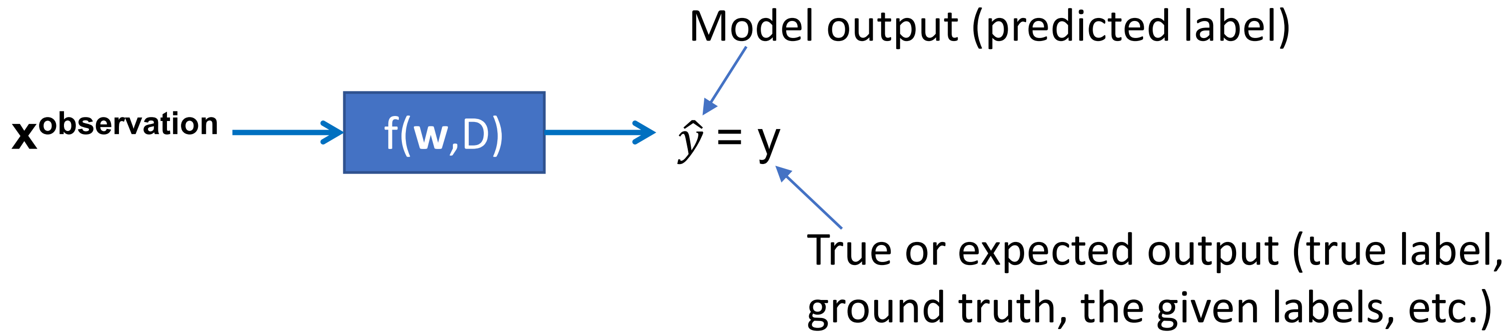
Note: Each sample can be written as a row vector as well.

n (or n_x): number of features

m: number of total observations (samples)

Introduction

Given observation (i.e., the data), derive a rule that can imitate the mechanism generating the observation: find the function f()



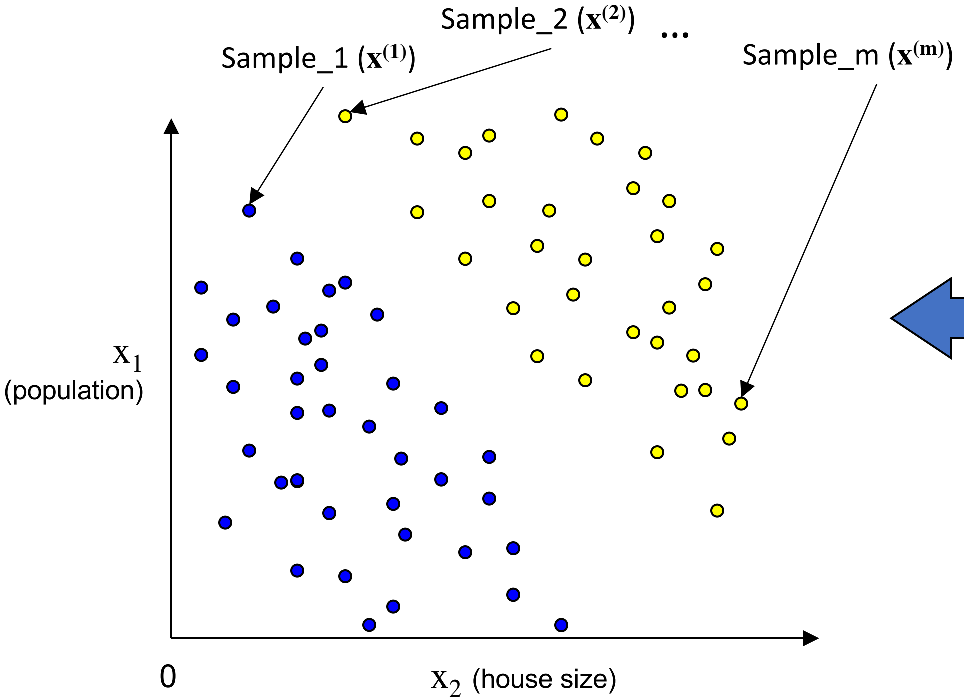
What is **supervised learning**?

$D=\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$ ==========> $w=g(\mathbf{X},\mathbf{y})$

Learn from the data with the labels.

Find the **model parameters** as a function of the data that fit the data the best.

Fundamentals: Classification



Yellow dot (∘): popular market
Blue dot (•): not popular market

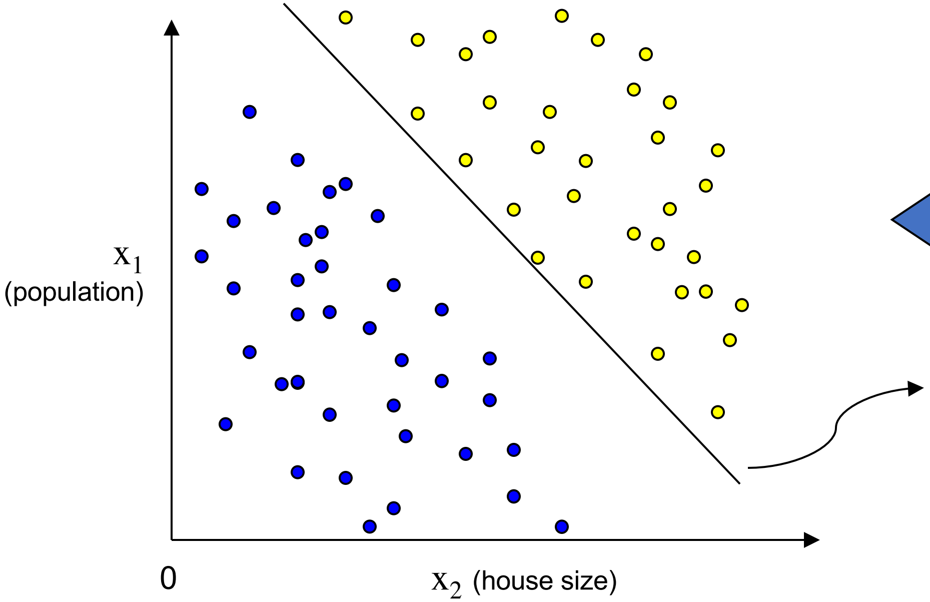
How to “separate” this data?

(how to categorize, classify this data being yellow or blue?)

Problem: Find an algorithm to classify the given data of a house coming from a popular market or from a non-popular market!

Fundamentals: Classification

Yellow dot (∘): popular market
Blue dot (•): not popular market



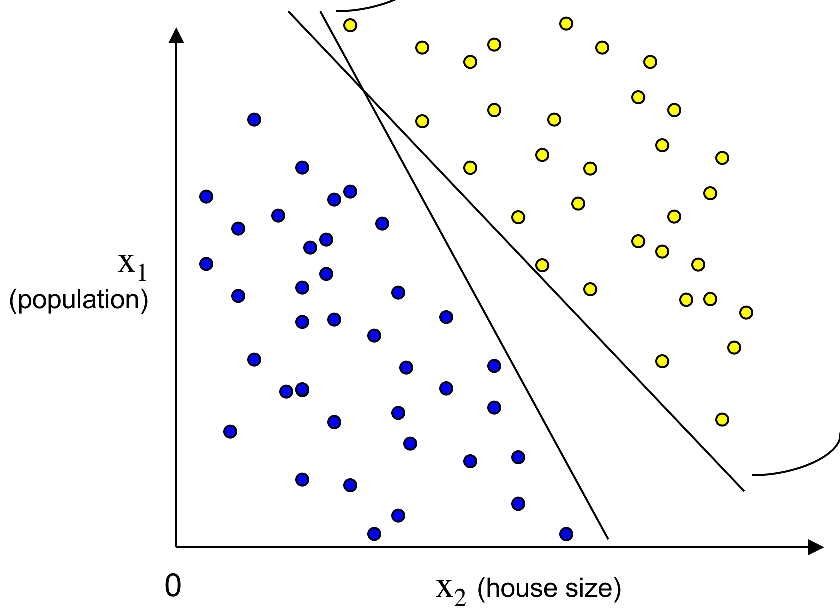
How to “separate” this data?

By drawing this line?

Fundamentals: Classification

Yellow dot (∘): popular market
Blue dot (•): not popular market

Or this one?



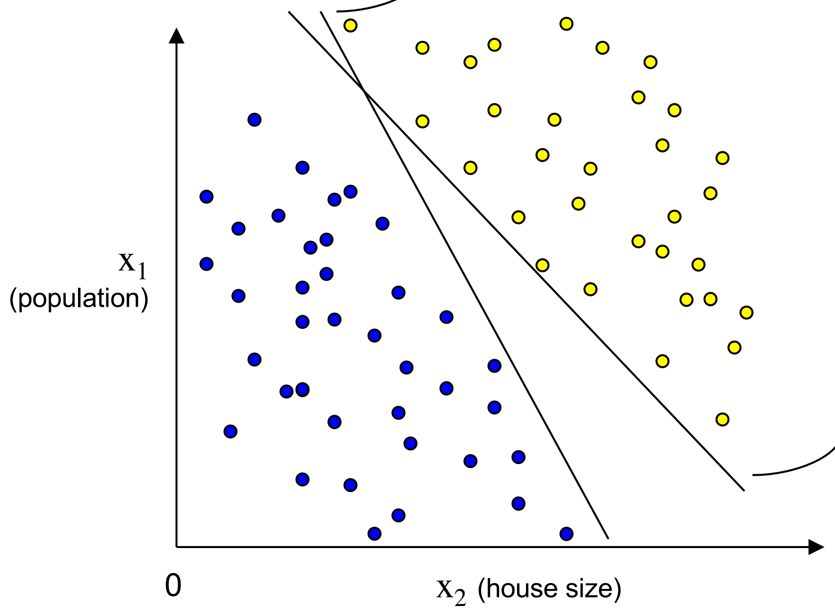
How to "separate" this data?

By drawing this line?

Fundamentals: Classification

Errmm, c'mon!
Does it really matter?

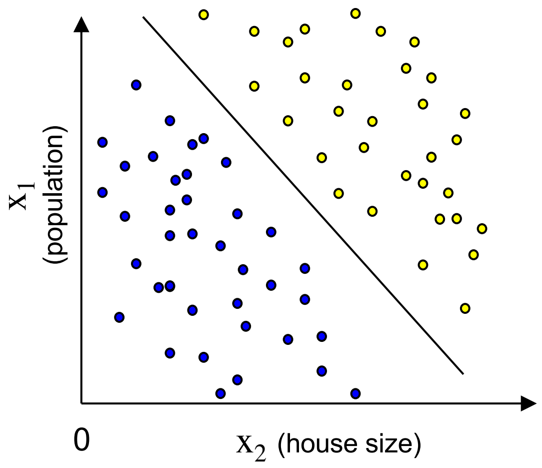
Or this one?



How to "separate" this data?

By drawing this line?

Linear Classifier



Yellow dot (∘): popular market = 1
Blue dot (•): not popular market = -1

Line equation for the above example is:

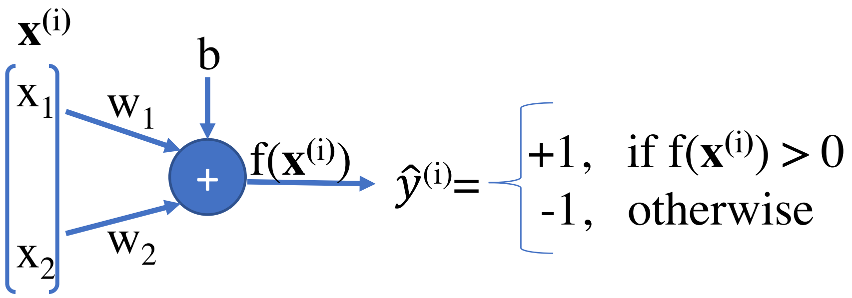
$$
\begin{aligned}
f(\mathbf{x}) &= w_1 x_1 + w_2 x_2 + b \\
&= \langle \mathbf{w}, \mathbf{x} \rangle + b \\
&= \mathbf{w}^T \mathbf{x} + b \\
&= \mathbf{w} \cdot \mathbf{x} + b
\end{aligned}
$$

} Alternative representations!

$f(\mathbf{x}) <$ threshold: (-1 class)

$f(\mathbf{x}) >$ threshold (+1 class)

$\mathbf{w}$: the weights vector - a (2x1) column vector,
$\mathbf{x}$: a training sample - a (2x1) column vector,
b: the bias value – a scalar (1x1) value.

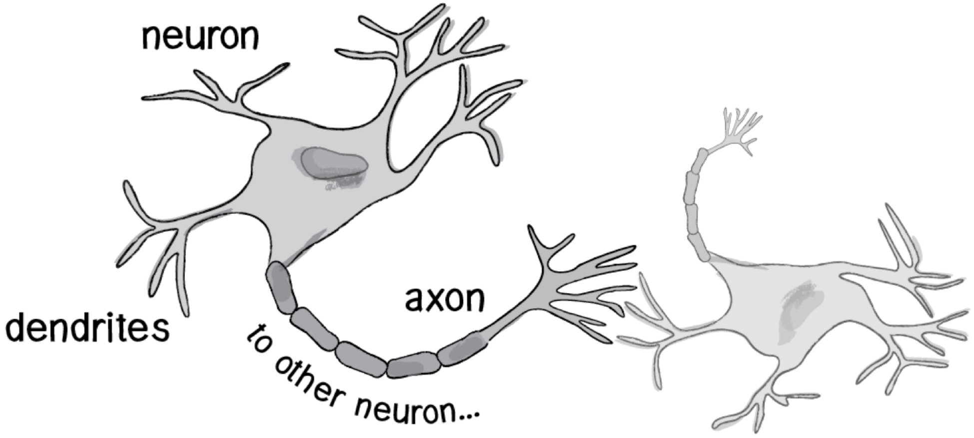


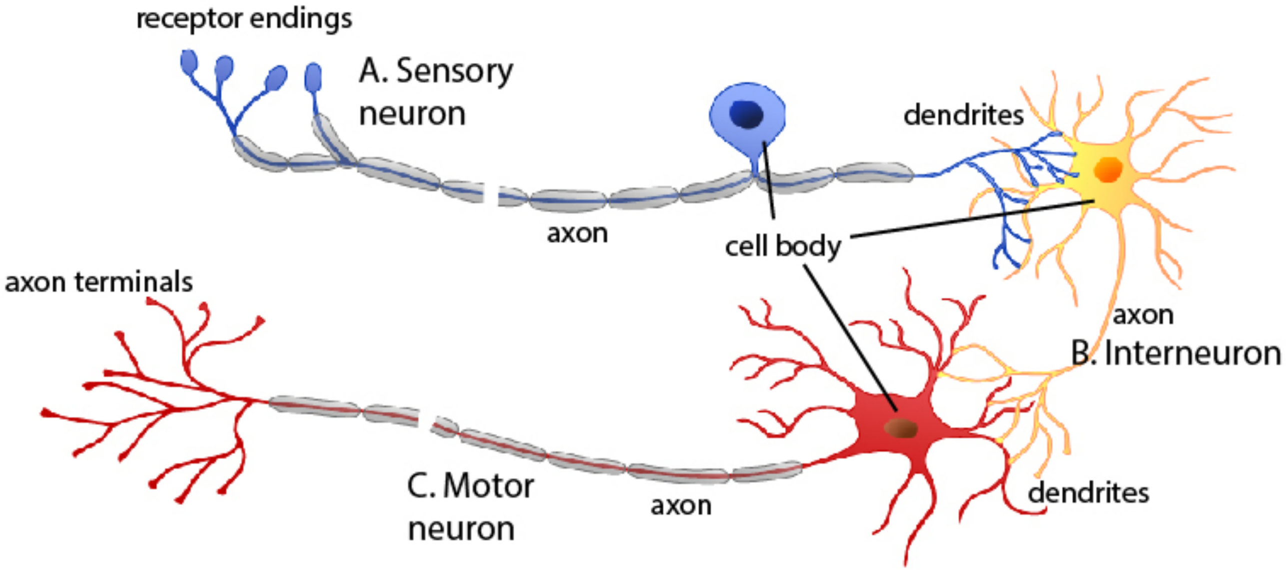
i=1,2,3,…,m. (sample number)

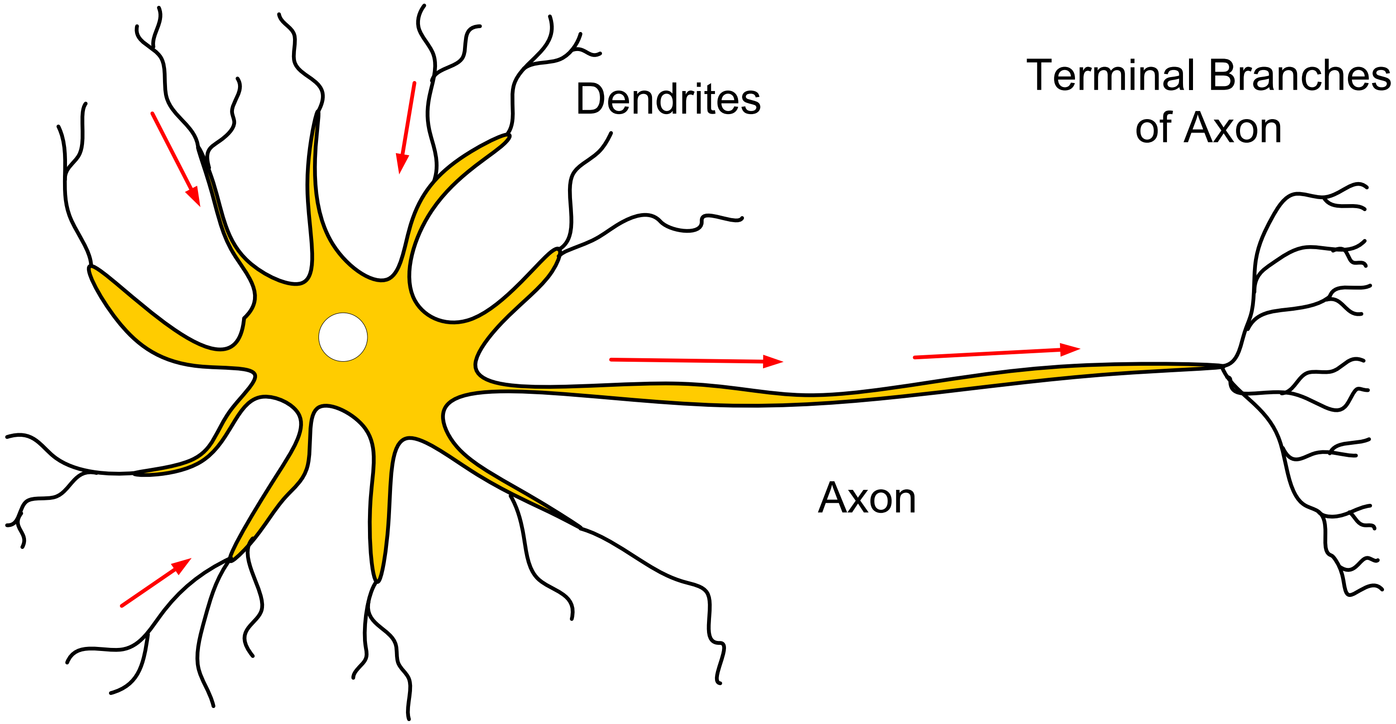
n=2 (number of features)

Training vs. testing – on the board.

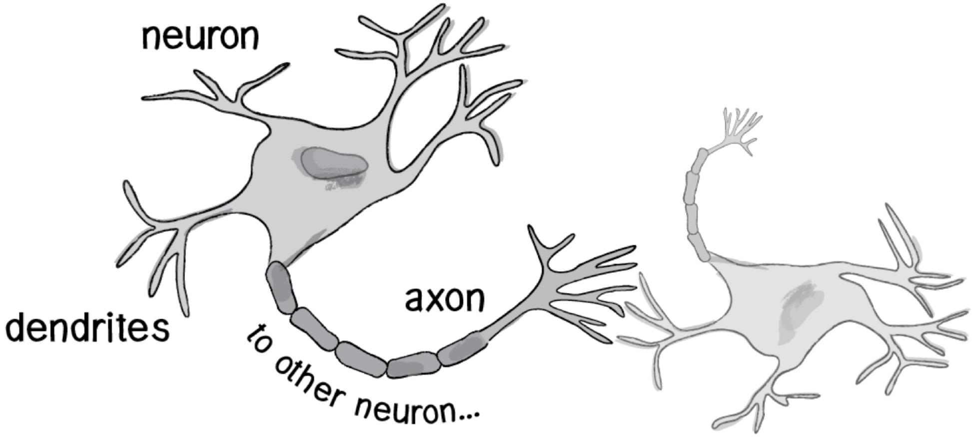
The (Biological) Analogy

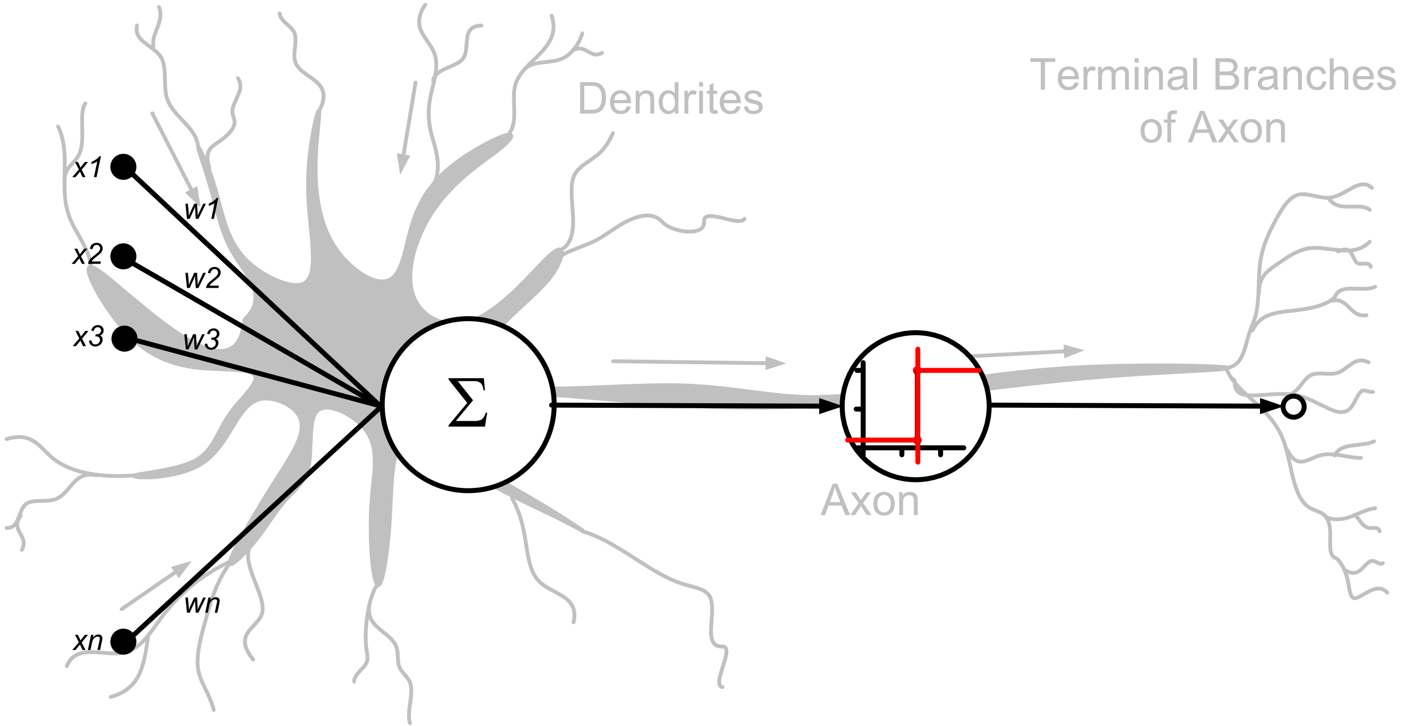






The (Biological) Analogy





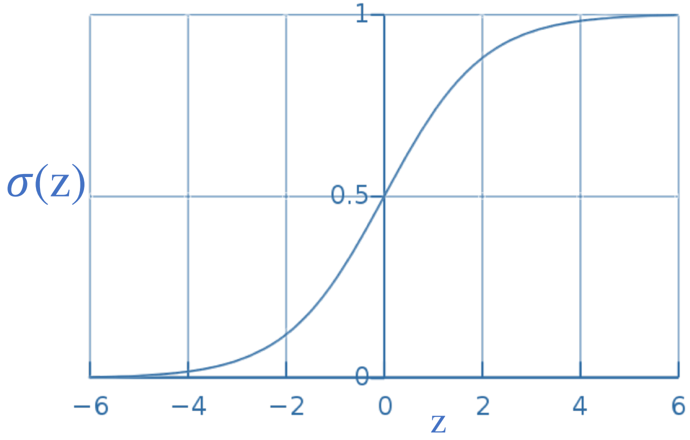
Logistic Regression

Given the input vector x, compute the output probability $\hat{y}$ such that:

- $\hat{y} = P(y=1 \mid x)$ ➡ This term can be read in many (similar) ways. One way to read: the probability of the output y being 1, while the the input data (or features) are given as x.

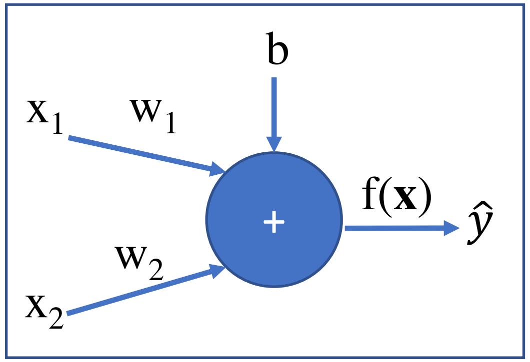
- Since the output prediction $\hat{y}$ is now a probabilistic term, its value has to be bounded between 0 and 1.
- **Logistic function**: $\sigma(z)$ does that job for us. (Also known as **Sigmoid function**)

Logistic function

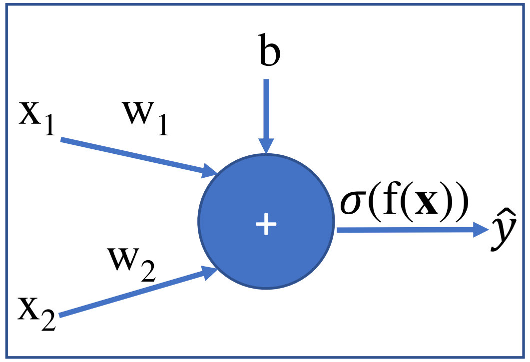


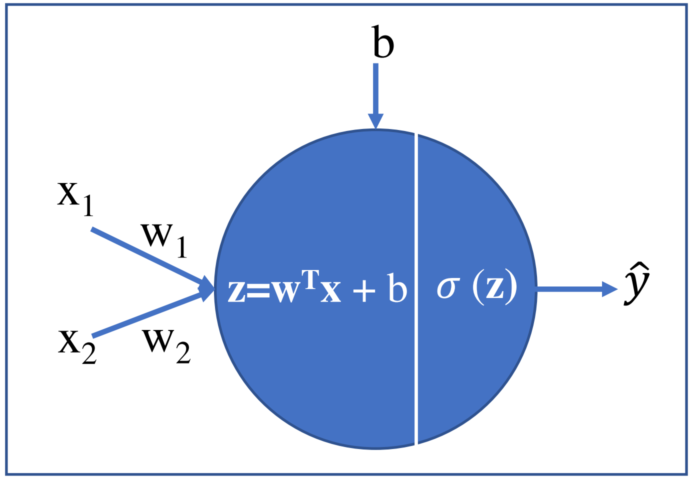
The output value of logistic function is always bounded between 0 and 1.

Logistic Regression



Becomes:





Two different illustrative representations of the "same model"

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{2x1} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_{2x1}$$

Remember the line equation:

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

$$f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + b$$

$$= \mathbf{w}^T \mathbf{x} + b$$

Now in logistic regression:

$$\hat{y} = \sigma(\mathbf{z})$$

$$\mathbf{z} = f(\mathbf{x})$$

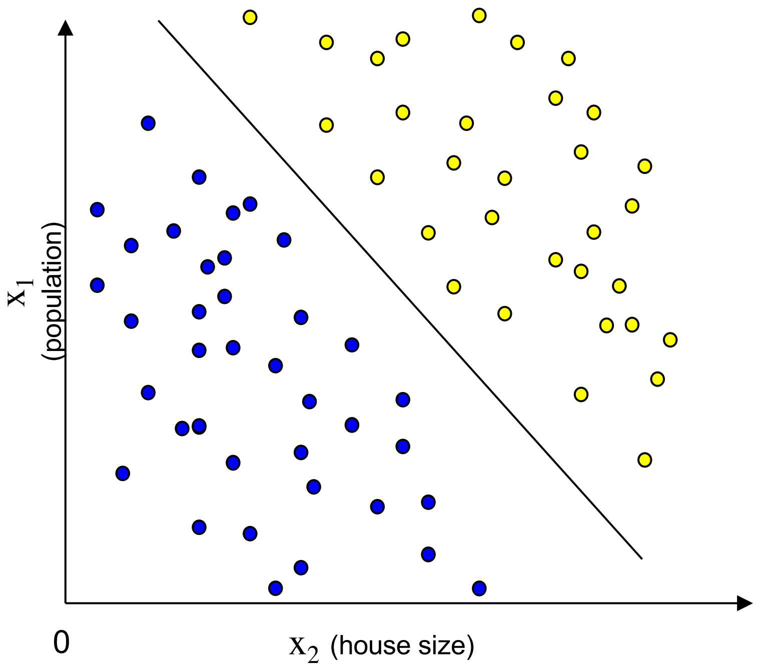
$$f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + b$$

$$= \mathbf{w}^T \mathbf{x} + b$$

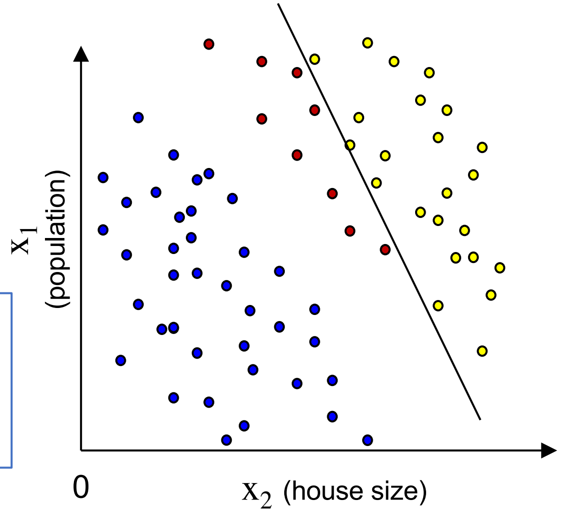
$$\hat{y} = \sigma(\mathbf{z}) = \sigma(f(\mathbf{x})) = \sigma(\mathbf{w}^T \mathbf{x} + b)$$

And…. $\sigma(\mathbf{z}) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(\mathbf{w}^T \mathbf{x} + b)}}$

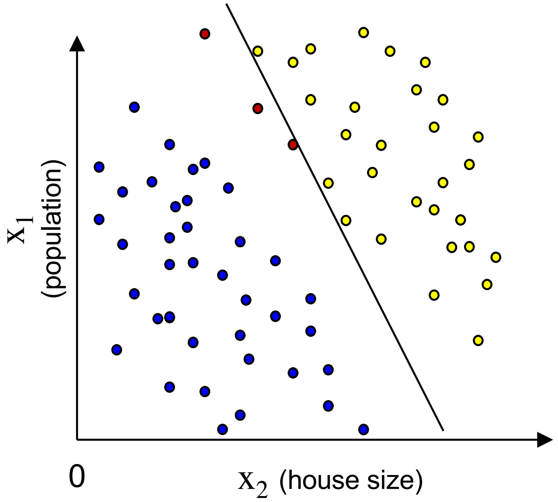
How to find the correct line?



Iteration 20
Training Error: 0 samples
(No error! Yay!)



Iteration 1
(Random initialization)
Training Error: 9 samples are misclassified



Iteration 10
Training Error: 3 samples are misclassified

Iterative computation of the parameters

- In the previous slide, we performed a form of optimization to find the better line (i.e., the better weights and the bias values) iteratively and intuitively.
  - Lets formalize that here next.
- We looked at a criteria to find better parameters (in the previous case, that was the total number of errors).
  - We need a criteria for an algorithm to figure out how well the algorithm is doing at that current iteration (at that moment):
  - Lets call that criteria a “**cost function**”!
    - Example: Total number of errors

Loss Function for Logistic Regression

- We need a way to compare the output of the algorithm's $\hat{y}$ to the expected (true) output value y. The error can be measured in various ways mathematically. Lets define that error measure as "loss function".

- Here is an example of a loss (error) function for any given data sample:

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = 0.5\big(\hat{y}^{(i)} - y^{(i)}\big)^2$$

- However this loss function does not work well for the main optimization algorithm that we will study next: gradient descent algorithm.

- For logistic regression algorithm, we will use the loss function below instead:

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -y^{(i)} \log \hat{y}^{(i)} - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

(for the meaning of this loss function: see the term: "**cross entropy**")

Cost Function for Logistic Regression

- Loss function $\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$ measures the error made for a single sample in the training data.

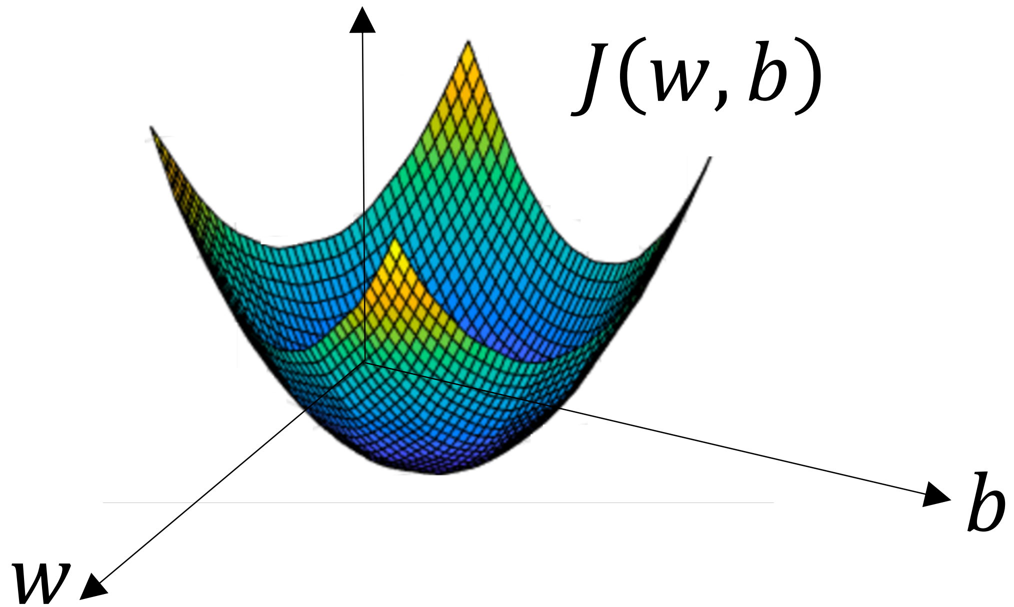
- Cost function $J(w, b)$ defines the global error over the entire dataset for the current parameters.

- Cost function for the logistic regression:

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right)$$

Gradient Descent Algorithm

The goal in optimization is finding the "optimal" model parameters: (w and b) that minimizes the given cost function.



Remember:

$$\hat{y} = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^{\mathrm{T}}\mathbf{x}+b)}}$$

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\hat{y}^{(i)} + (1 - y^{(i)})\log(1 - \hat{y}^{(i)})\right)$$

Gradient Descent Update Rule

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right)$$

Parameters that we learn!

Weight updating rule: $w := w - \alpha \frac{\partial J(w,b)}{\partial w}$

Bias updating rule: $b := b - \alpha \frac{\partial J(w,b)}{\partial b}$

Partial derivatives

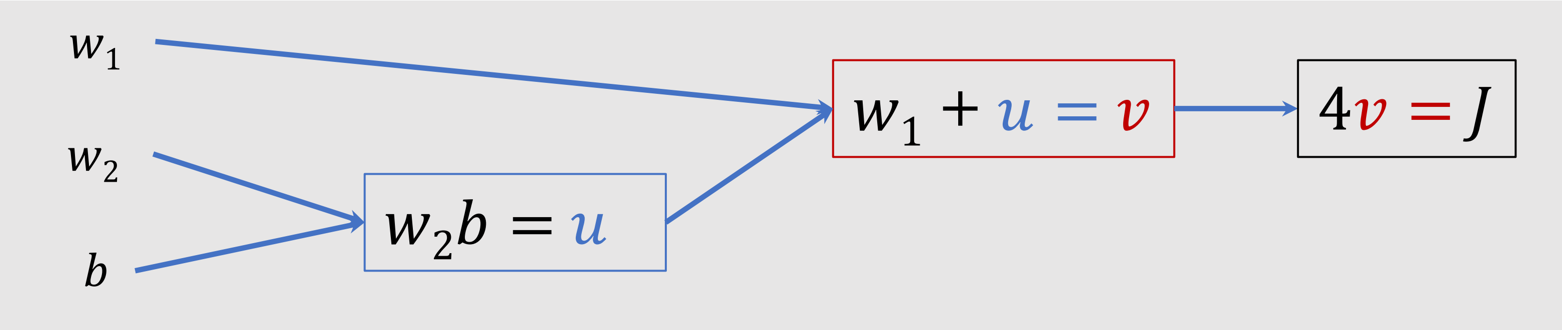
α : Learning rate

Computation Graph

- We can represent the cost function as a graph.
- Useful to understand the deep learning essentials (forward and backward computations).
- Example: Consider the following cost function and define new variables

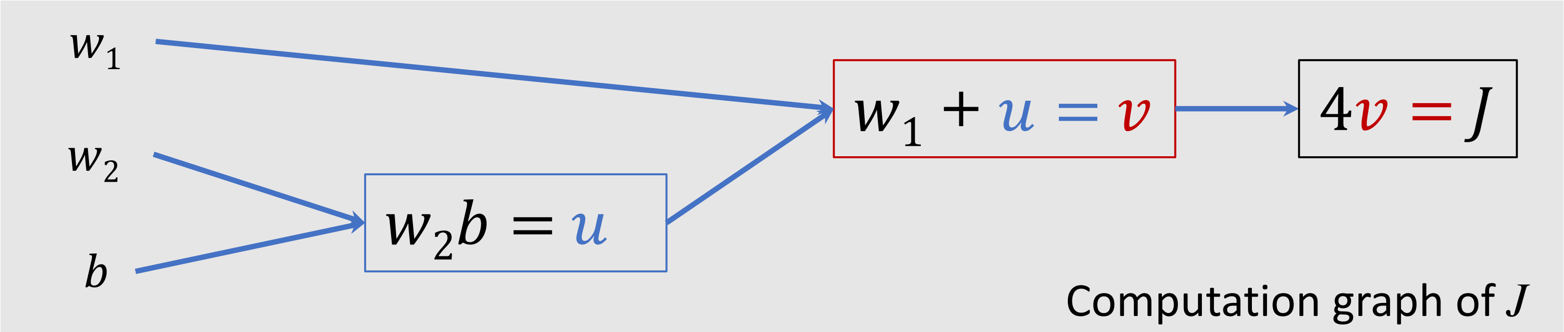
$J(w_1, w_2, b) = 4\,(w_1 + w_2 b)$

$u = w_2 b$ $v = w_1 + u$ $J = 4v$



Computation graph of J

Computation Graph for Chain Rule



Computation graph of J

Sample cost function:

$$J(w_1, w_2, b) = 4\,(w_1 + w_2 b)$$

$$\frac{\partial J}{\partial v} = 4$$

$$\frac{\partial J}{\partial u} = \frac{\partial J}{\partial v}\frac{\partial v}{\partial u} = 4 \times 1 = 4$$

$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial v}\frac{\partial v}{\partial w_1} = 4 \times 1 = 4$$

$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial u}\frac{\partial u}{\partial w_2} = 4 \times b = 4b$$

$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial u}\frac{\partial u}{\partial b} = 4 \times w_2 = 4w_2$$

if $w_1=3,\ w_2 = 2, b = 5$

$$\frac{\partial J}{\partial v} = 4$$

$$\frac{\partial J}{\partial u} = \frac{\partial J}{\partial v}\frac{\partial v}{\partial u} = 4 \times 1 = 4$$

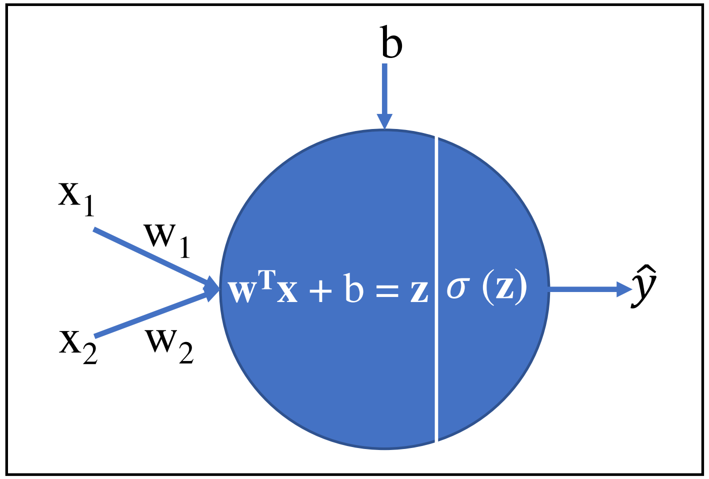
$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial v}\frac{\partial v}{\partial w_1} = 4 \times 1 = 4$$

$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial u}\frac{\partial u}{\partial w_2} = 4 \times b = 4x5=20$$

$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial u}\frac{\partial u}{\partial b} = 4 \times w_2 = 4x2 = 8$$

Logistic Regression Computation Graph

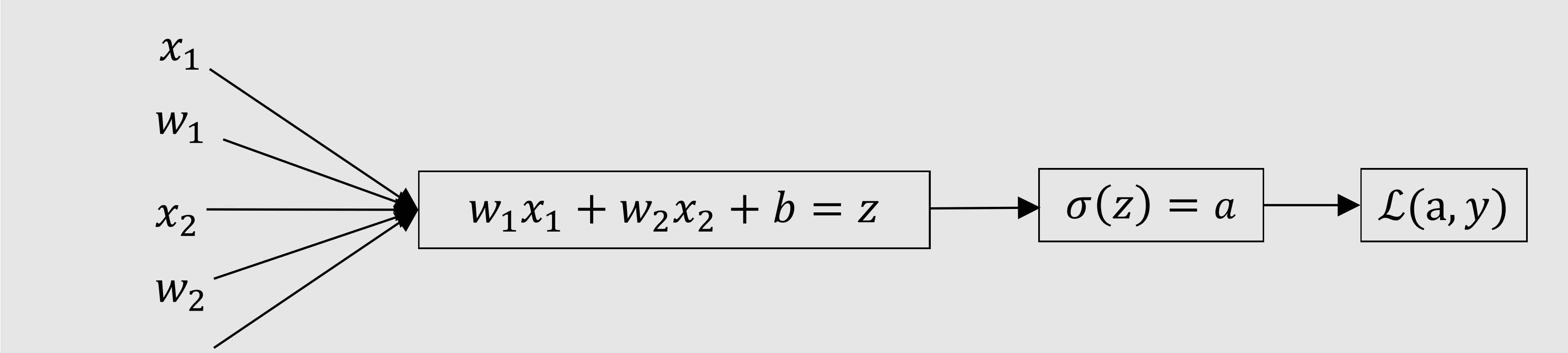
Remember:
$$\begin{cases} z = \mathbf{w}^T\mathbf{x} + b \\ \hat{y} = a = \sigma(z) \\ \mathcal{L}(a, y) = -(y\log(a) + (1-y)\log(1-a)) \end{cases}$$



For 1 (one) training example (say for i^th sample):

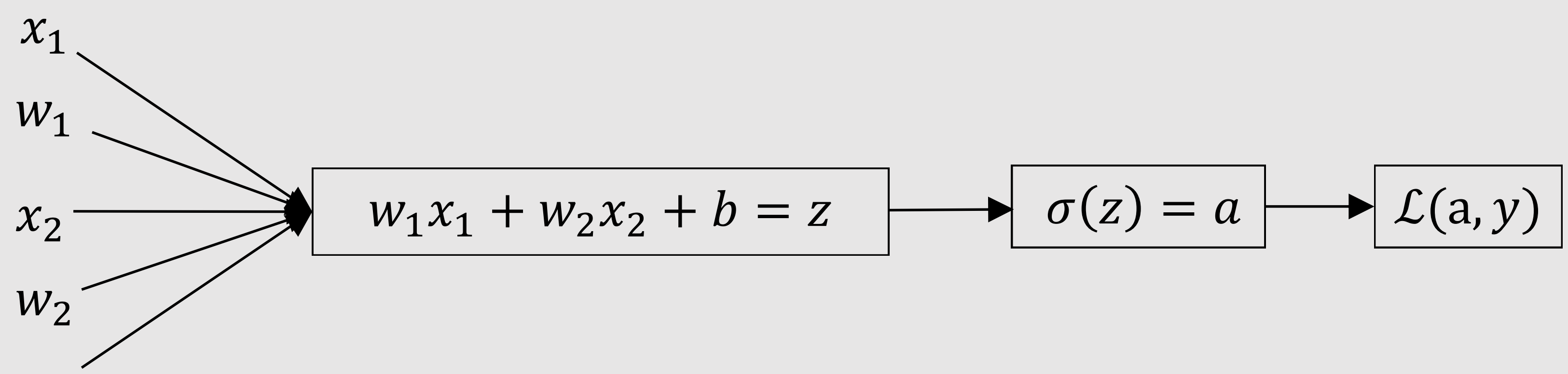
- First compute z
- Then compute $\sigma(z)$
- Then compute Loss: $\mathcal{L}(a, y)$

(later, we will also compute the cost for all the examples, however, here we consider only the one example case)



Computation graph of $\mathcal{L}$

Derivatives for Logistic Regression



Computation graph of $\mathcal{L}$

$$\frac{\partial \mathcal{L}(\mathrm{a},y)}{\partial a} = \frac{-y}{a} + \frac{1-y}{1-a}$$

$$\frac{\partial a}{\partial z} = a(1-a)$$

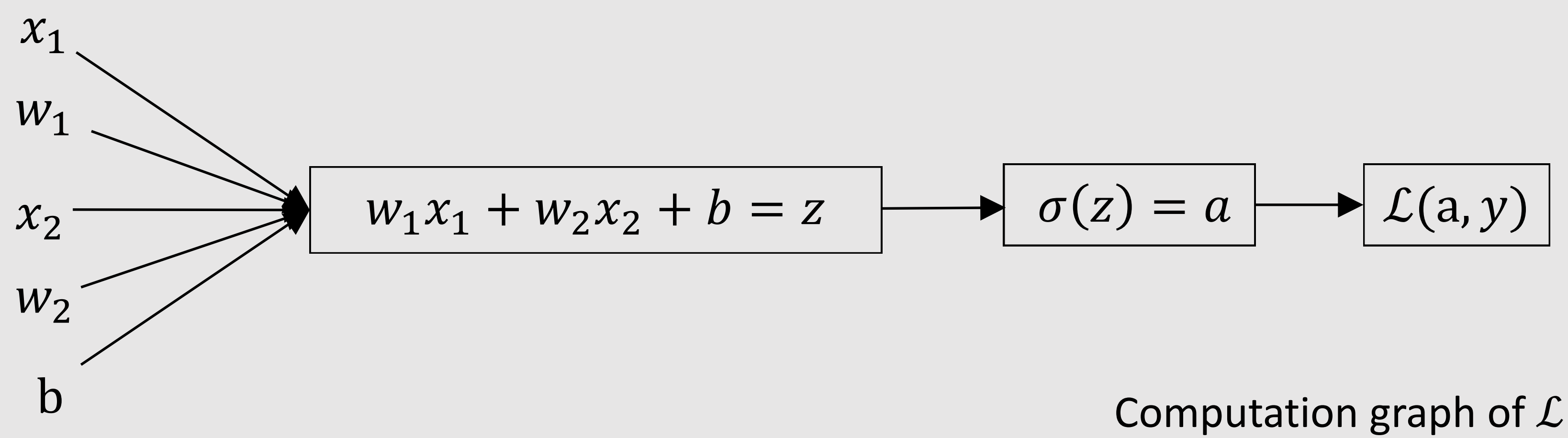
$$\frac{\partial \mathcal{L}(\mathrm{a},y)}{\partial z} = \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial z} = a - y$$

$$\frac{\partial \mathcal{L}(\mathrm{a},y)}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial z} x_1 = x_1(a-y)$$

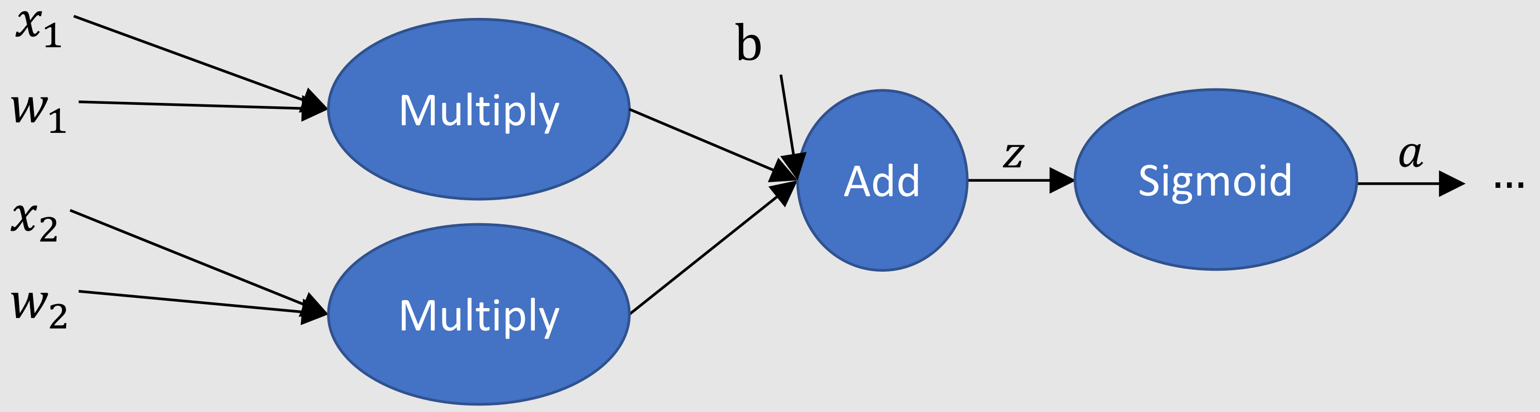
$$\frac{\partial \mathcal{L}(\mathrm{a},y)}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial z} x_2 = x_2(a-y)$$

$$\frac{\partial \mathcal{L}(\mathrm{a},y)}{\partial b} = \frac{\partial \mathcal{L}}{\partial z} = a - y$$

Derivatives for Logistic Regression

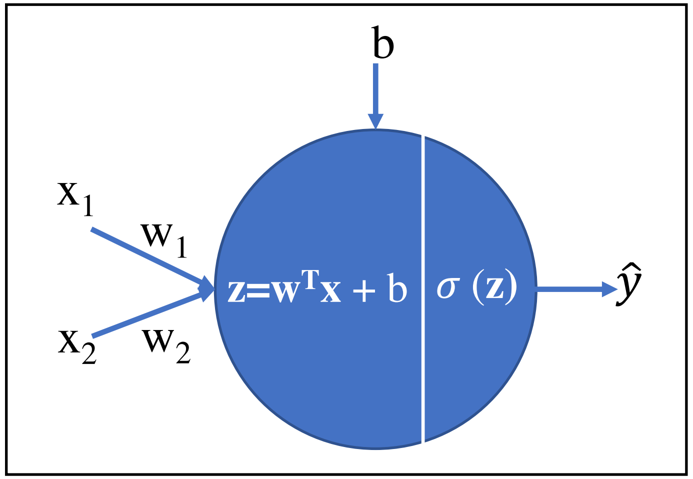


Computation graph of $\mathcal{L}$



An alternative representation of the computation graph of $\mathcal{L}$

Cost function and implementation



Remember the loss for single sample:

$$\frac{\partial \mathcal{L}(a,y)}{\partial w_1} = x_1(a-y)$$

$$\frac{\partial \mathcal{L}(a,y)}{\partial w_2} = x_2(a-y)$$

$$\frac{\partial \mathcal{L}(a,y)}{\partial b} = a-y$$

Cost Function:

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Final derivatives to be used:

$$\frac{\partial J(w,b)}{\partial w_1} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial \mathcal{L}(a^{(i)}, y^{(i)})}{\partial w_1}$$

$$\frac{\partial J(w,b)}{\partial w_2} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial \mathcal{L}(a^{(i)}, y^{(i)})}{\partial w_2}$$

$$\frac{\partial J(w,b)}{\partial b} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial \mathcal{L}(a^{(i)}, y^{(i)})}{\partial b}$$

Implement all that:

$J=0$; $dw_1=0$; $dw_2=0$; $db=0$; $\propto = 0.00001$

For i=1 to m

$z^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J \mathrel{+}= -[y^{(i)}\log a^{(i)} + (1-y^{(i)})\log(1-a^{(i)})]$

$dz^{(i)} = a^{(i)} - y^{(i)}$

$dw_1 \mathrel{+}= x_1^{(i)}\, dz^{(i)}$

$dw_2 \mathrel{+}= x_2^{(i)}\, dz^{(i)}$

$db \mathrel{+}= dz^{(i)}$

$J = J/m$; $dw_1 = dw_1/m$

$dw_2 = dw_2/m$; $db = db/m$

$w_1 := w_1 - \propto dw_1$

$w_2 := w_2 - \propto dw_2$

$b := b - \propto db$

Notes

This lecture has some content from Andrew Ng and Ulas Bagci.