Logistic Regression (Binary Classification)

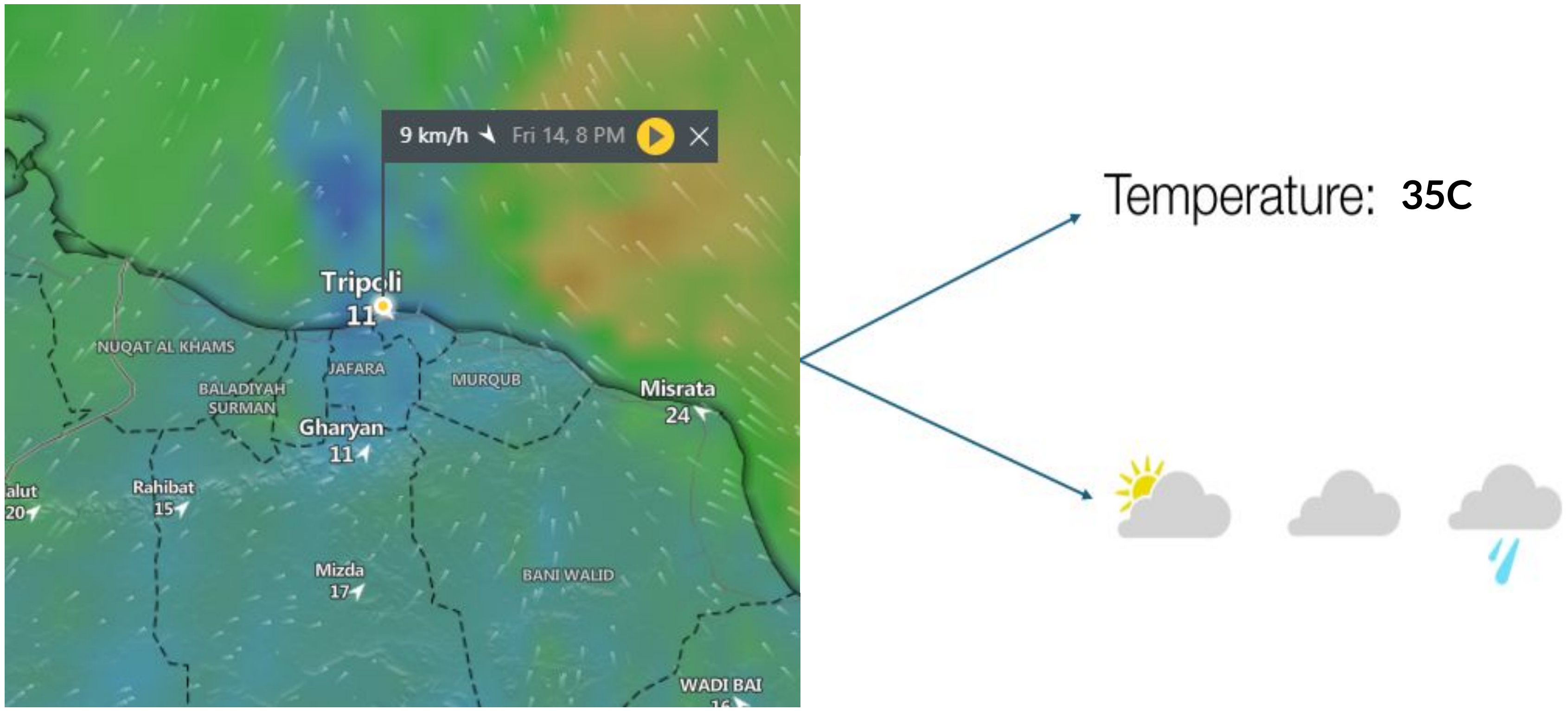
Ali Aburas, PhD

Slide credit:Andrew Ng

Outline

Regression vs Classification

- Regression: Given input x, predict <u>continuous</u> value $y \in \mathbb{R}$
- Classification: Given input x, predict <u>discrete</u> value $y \in \mathcal{Y}$



Classification

- The **classification** problem is just like the regression problem, except that the values *y* we now want to predict take on only a small number of discrete values.

Question	Answer "*y*"	
Is this email spam? Spam / Not spam	no	yes
Is the transaction fraudulent?	no	yes
Is the tumor malignant? Malignant/ Benign	no	yes

y can only be one of **two** values false

"**binary classification**"

0 1

useful for classification

"negative class" "positive class"

≠ "bad" ≠ "good"

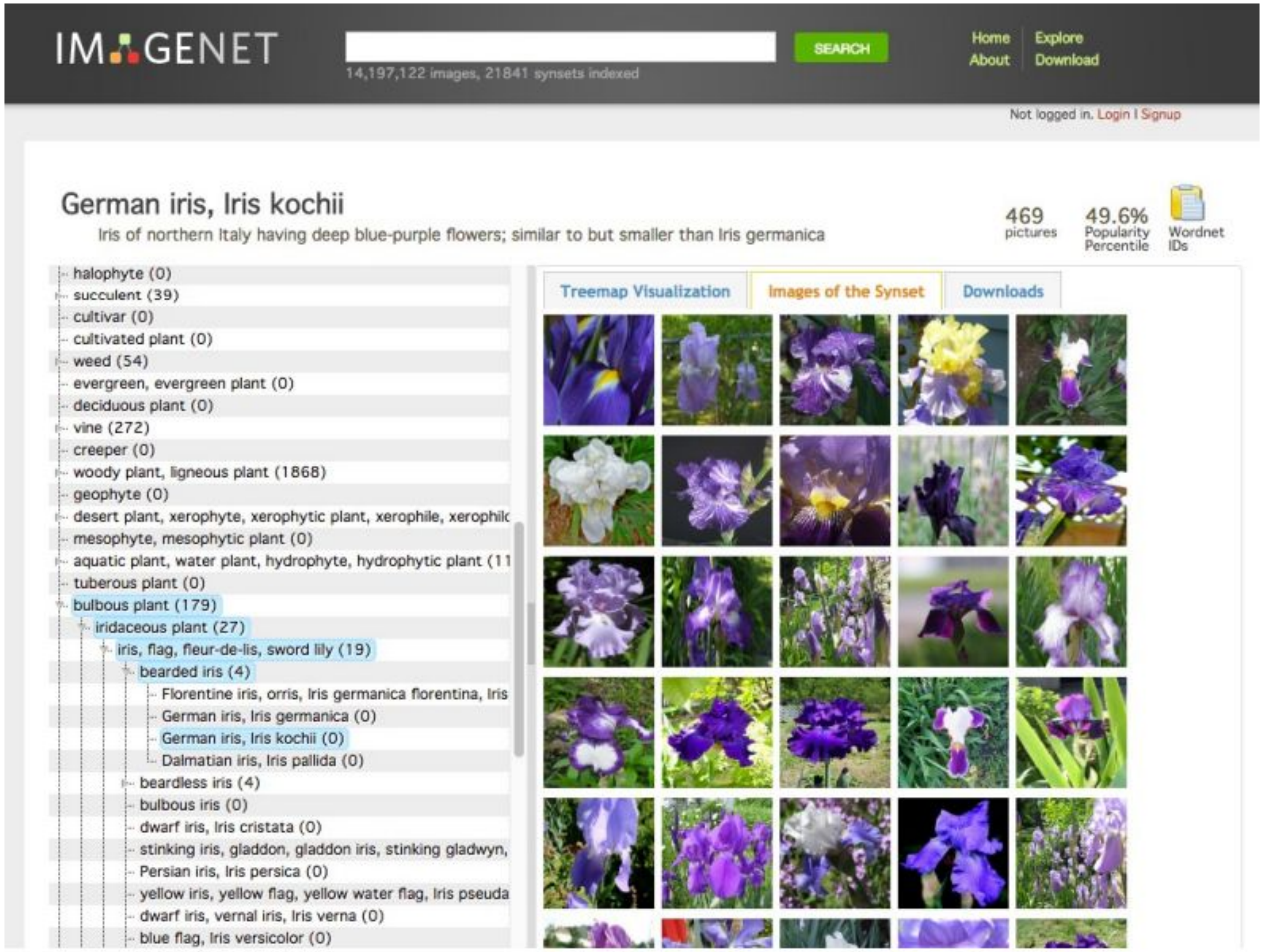
absence presence

$$y \in \{0, 1\}$$

0: "Negative Class" (e.g., benign tumor)

1: "Positive Class" (e.g., malignant tumor)

Example: Image Classification



14,197,122 images, 21841 synsets indexed

German iris, Iris kochii

Iris of northern Italy having deep blue-purple flowers; similar to but smaller than Iris germanica

469 pictures

49.6% Popularity Percentile

Wordnet IDs

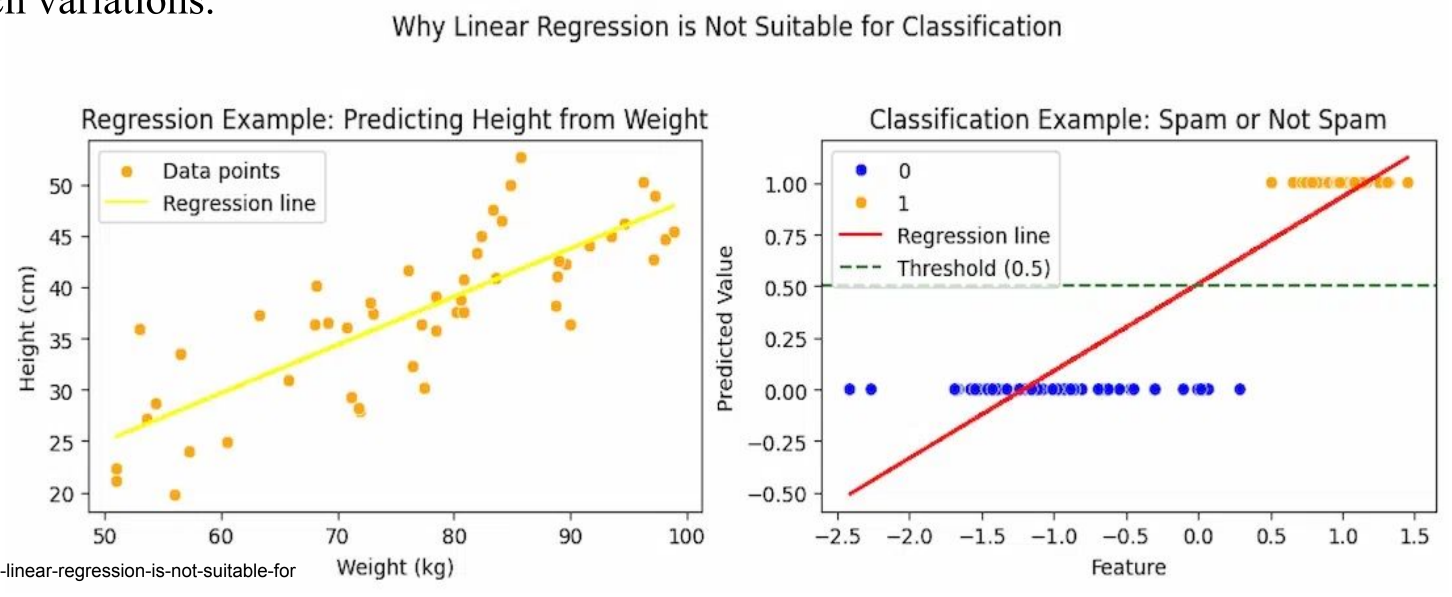
- halophyte (0)
- succulent (39)
- cultivar (0)
- cultivated plant (0)
- weed (54)
- evergreen, evergreen plant (0)
- deciduous plant (0)
- vine (272)
- creeper (0)
- woody plant, ligneous plant (1868)
- geophyte (0)
- desert plant, xerophyte, xerophytic plant, xerophile, xerophilc
- mesophyte, mesophytic plant (0)
- aquatic plant, water plant, hydrophyte, hydrophytic plant (11
- tuberous plant (0)
- bulbous plant (179)
  - iridaceous plant (27)
    - iris, flag, fleur-de-lis, sword lily (19)
      - bearded iris (4)
        - Florentine iris, orris, Iris germanica florentina, Iris
        - German iris, Iris germanica (0)
        - German iris, Iris kochii (0)
        - Dalmatian iris, Iris pallida (0)
      - beardless iris (4)
      - bulbous iris (0)
      - dwarf iris, Iris cristata (0)
      - stinking iris, gladdon, gladdon iris, stinking gladwyn,
      - Persian iris, Iris persica (0)
      - yellow iris, yellow flag, yellow water flag, Iris pseuda
      - dwarf iris, vernal iris, Iris verna (0)
      - blue flag, Iris versicolor (0)

https://image-net.org/index.php

Why Linear Regression is not suitable for classification?

There are three things that explain why Linear Regression is not suitable for classification.

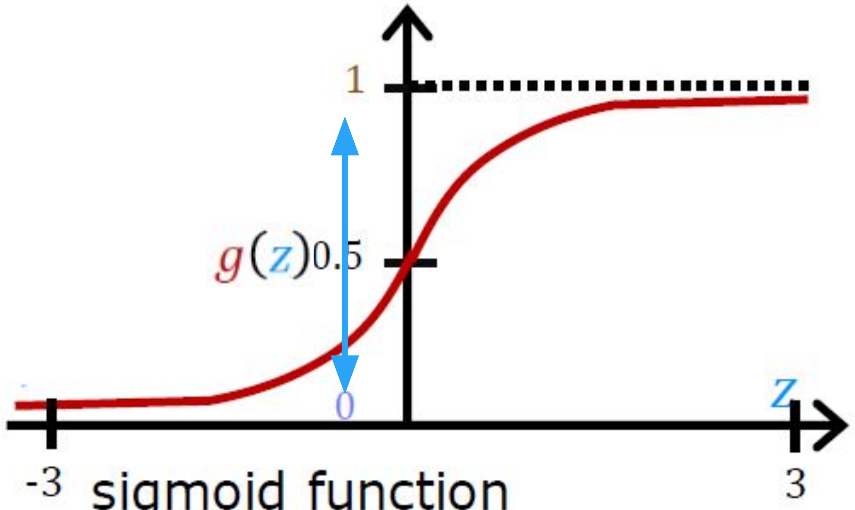
1. The first one is that Linear Regression deals with continuous values whereas classification problems mandate discrete values.
2. Linear regression assumes a linear relationship between the independent variables and the target variable. In classification, the decision boundaries that separate different classes are rarely linear.
3. Linear regression is sensitive to outliers, meaning that a single extreme data point can significantly affect the slope and intercept of the line. In classification, outliers are common and might belong to either class, so we need a model that's more robust to such variations.



https://www.geeksforgeeks.org/why-linear-regression-is-not-suitable-for-classification/

Logistic Regression

Want outputs between 0 and 1



sigmoid function

logistic function

$$\boxed{\begin{array}{l}\text{outputs between 0 and 1} \\ g(z) = \frac{1}{1+e^{-z}} \qquad 0 < g(z) < 1\end{array}}$$

$$\boxed{\begin{tikzcd} f_{\vec{w},b}(\vec{x}) \\ \boxed{\vec{w}\cdot\vec{x}+b} \arrow[d] \\ z \arrow[d] \\ \boxed{g(z) = \frac{1}{1+e^{-z}}} \end{tikzcd}}$$

$$f_{\vec{w},b}(\vec{x}) = g(\underbrace{\vec{w}\cdot\vec{x}+b}_{z}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$$

"logistic regression"

=2.7

$$\operatorname{argmax}_w \sum_{i=1}^{n} \log\left(\frac{1}{1+e^{-y_i w^\top x_i}}\right)$$

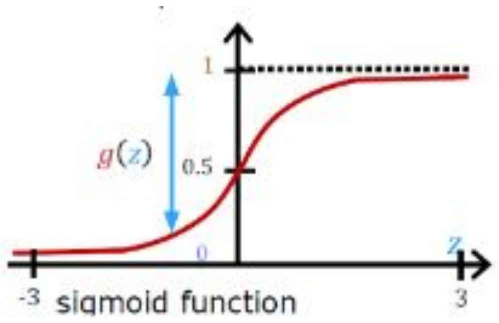
Logistic Regression (Example)

$Z=5X + 10$

$$\hat{Y} = \frac{1}{1+e^{-Z}}$$

X	**-9**	**-8**	**0**	**8**	**9**
Z	-35	-30	10	50	55
$\hat{Y}$	0	0	1	1	1

$\hat{Y}$ >= 0.5 threshold then =1 $\hat{Y}$ < 0.5 threshold then =0



Inference:

If Z value is a large positive number, $\hat{Y} = 1$

If Z value is a large negative number, $\hat{Y}=0$

Loss Function and Cost Function for Logistic Regression

Loss function measures how far an estimated value is from its true value.

Let us remember the loss function of Linear Regression; we used Mean Square Error as a loss function

$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \hat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

But in the case of **Logistic Regression**, we cannot use **Mean Squared Error** as a loss function because the hypothesis function of logistic is **non-linear**.

Another reason for not using Mean Squared Error in Logistic Regression is that our output lies between 0 - 1. In classification problems, the target value is either 1 or 0. The output of the Logistic Regression is a probability value between 0 to 1.

Loss Function for Logistic Regression

Binary Cross Entropy Loss Function (or) Log Loss:

$$L(y, \hat{y}) = \; -\left(y \log \hat{y} + (1-y) \log\left(1-\hat{y}\right)\right)$$

When y =1, $\implies L(1, \hat{y}) = \; -\left(1 \log \hat{y} \; + \; (1-1) \log\left(1-\hat{y}\right)\right) = -\log \hat{y}$

When y=0 $\implies L(0, \hat{y}) = \; -\left(0 \log \hat{y} \; + \; (1-0) \log\left(1-\hat{y}\right)\right) = -\log\left(1-\hat{y}\right)$

Loss Function for Logistic Regression

$$\hat{Y} = \frac{1}{1+e^{-Z}}$$

Binary Cross Entropy Loss Function (or) Log Loss:

$$L(y,\hat{y}) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$$

When y =1, $\implies L(1,\hat{y}) = -(1 \log \hat{y} + (1-1)\log(1-\hat{y})) = -\log \hat{y}$

We always want a smaller Loss Function value, hence, $\hat{Y}$ should be very largy, so that (- log $\hat{Y}$) will be a large negative number.

When y=0 $\implies L(0,\hat{y}) = -(0 \log \hat{y} + (1-0)\log(1-\hat{y})) = -\log(1-\hat{y})$

We always want a smaller Loss Function value, hence, $\hat{Y}$ should be very small, so that (- log (1- $\hat{Y}$)) will be a large negative number.

Example:

- - log(0.9) = - 0.045757 while - log(1-0.1) = -0.045757

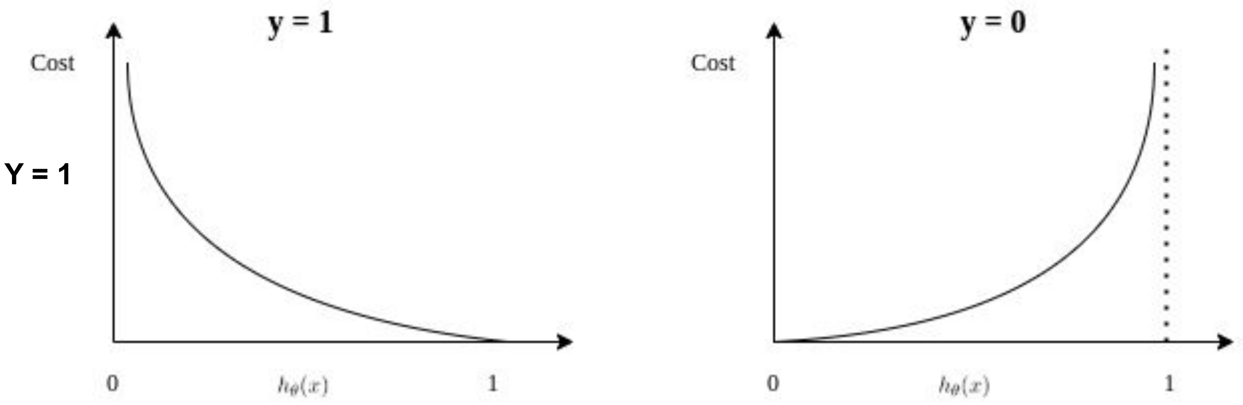
Loss Function for Logistic Regression

We can define the cost for two cases separately:

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

Which then results in:

Part 1 : When Y = 1



Because when the actual outcome $y = 1$, the cost is 0 for $h_\theta(x) = 1$ and takes the maximum value for $h_\theta(x) = 0$. Similarly, if $y = 0$, the cost is 0 for $h_\theta(x) = 0$.

Cost Function for Logistic Regression

- **Loss Function (L)** mainly applies for a single training set as compared to the
- **Cost Function (J)** which deals with a penalty for a number of training sets or the complete batch.

$$L(y, \hat{y}) = -(y \log \hat{y} + (1 - y) \log (1 - \hat{y}))$$

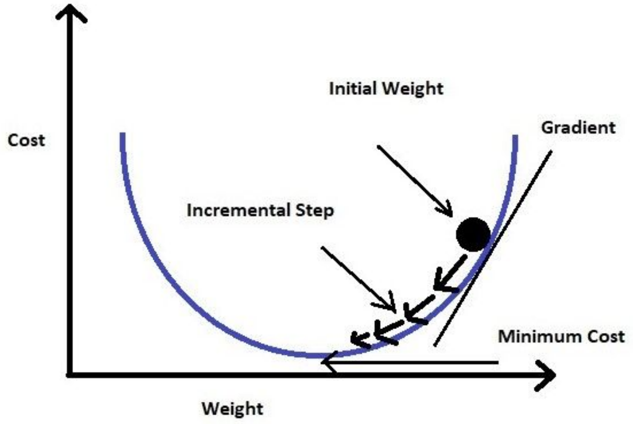
$$J(w, b) = -\frac{1}{m}\sum_{i=1}^{m}(L(y^{(i)}, \hat{y}^{(i)})) = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)}))$$

m denotes the number of data points in the training set.

Gradient Descent for Logistic Regression

Gradient Descent is an optimization algorithm used for minimizing the cost function in various machine learning algorithm. It is used for updating the parameters of the learning model.

$$w_2 = w_1 - L \cdot dw$$
$$b_2 = b_1 - L \cdot db$$



W ⟶ weight

b ⟶ bais

L ⟶ Learning Rate

dw ⟶ Partial Derivative of cost function with respect to w

db ⟶ Partial Derivative of cost function with respect to b

$$dw = \frac{1}{m} \cdot (\hat{y} - y) \cdot X$$

$$db = \frac{1}{m} \cdot (\hat{y} - y)$$

Gradient Descent for Logistic Regression

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

$$J(\vec{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right]$$

repeat { □

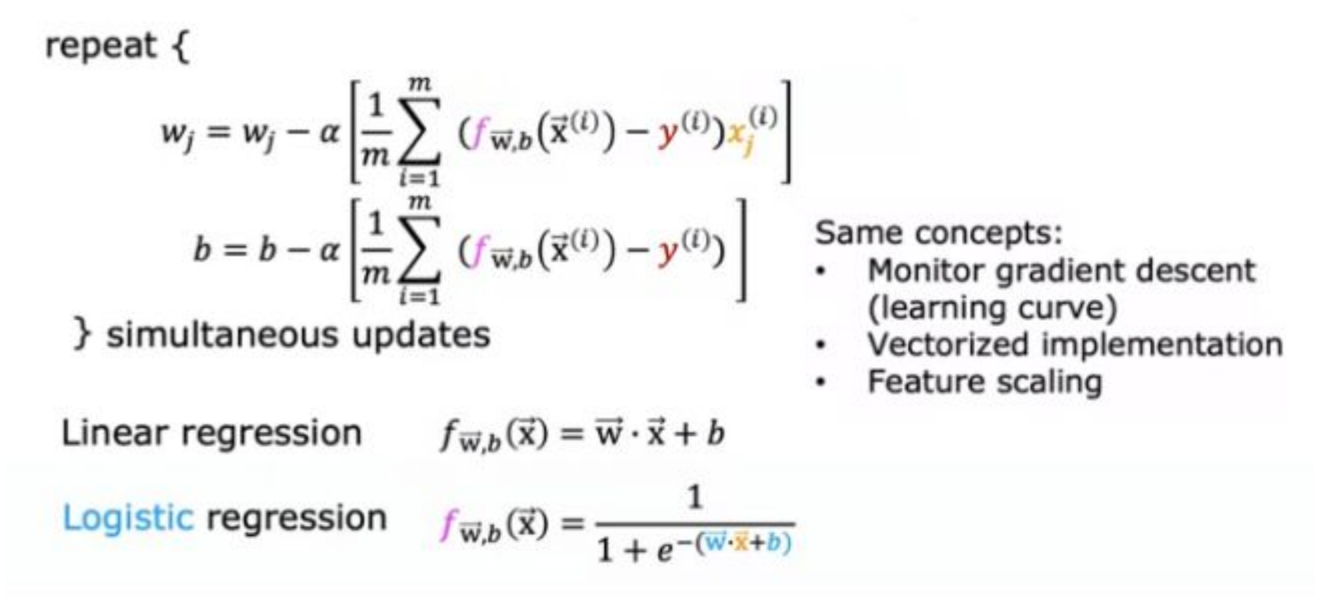
$$w_j = w_j - \alpha \overbrace{\frac{\partial}{\partial w_j}J(\vec{w}, b)} \qquad \frac{\partial}{\partial w_j}J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})\underbrace{x_j^{(i)}}$$

$$b = b - \alpha \underbrace{\frac{\partial}{\partial b}J(\vec{w}, b)} \qquad \frac{\partial}{\partial b}J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})\underbrace{\quad}$$

} simultaneous updates

Gradient Descent for Logistic Regression

- The formula for gradient descent for logistic regression is the same as that for linear regression



repeat {

$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_j^{(i)} \right]$$

$$b = b - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) \right]$$

} simultaneous updates

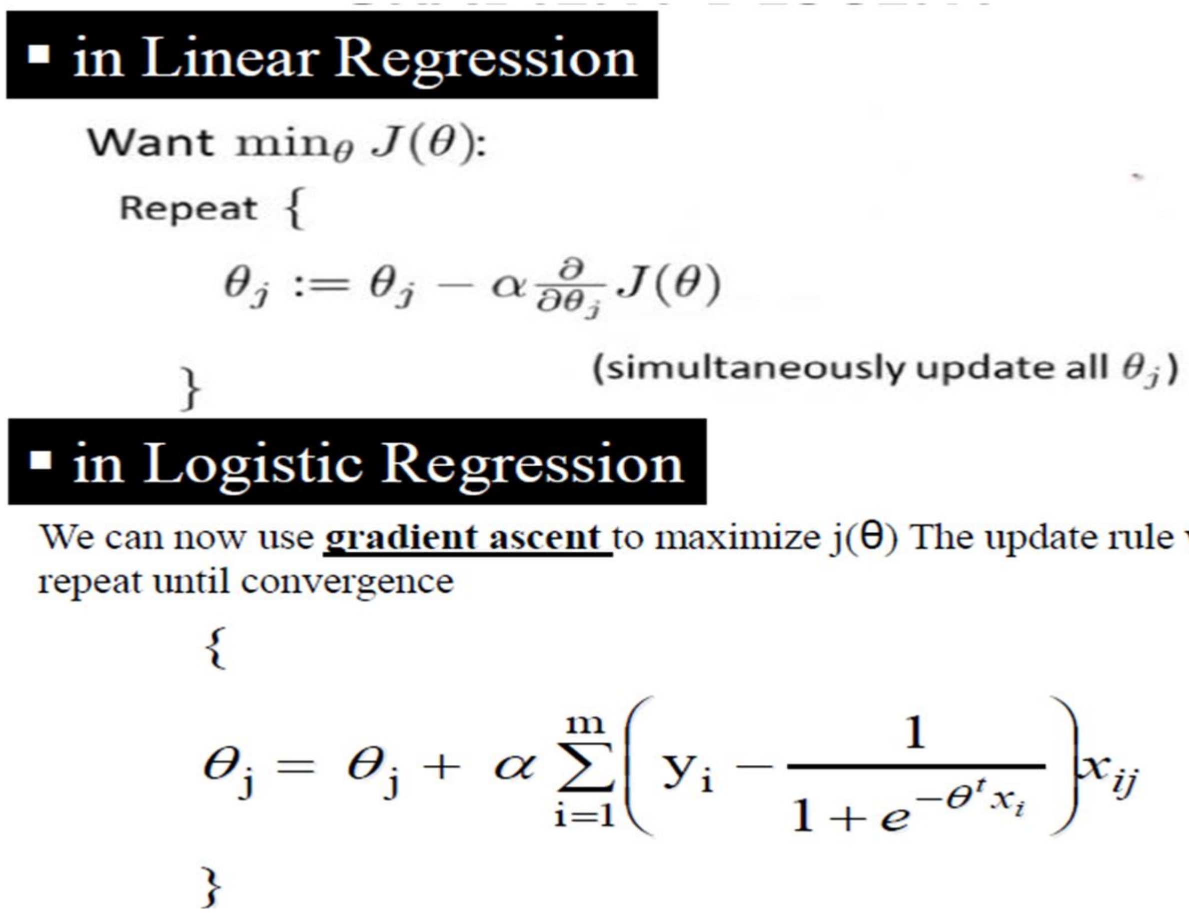
Same concepts:
- Monitor gradient descent (learning curve)
- Vectorized implementation
- Feature scaling

Linear regression $f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

Logistic regression $f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$

Gradient Descent for Logistic Regression

Gradient Descent is an optimization algorithm used for minimizing the cost function in various machine learning algorithm. It is used for updating the parameters of the learning model



- in Linear Regression

Want $\min_\theta J(\theta)$:

Repeat $\{$

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

$\}$ (simultaneously update all θ_j)

- in Logistic Regression

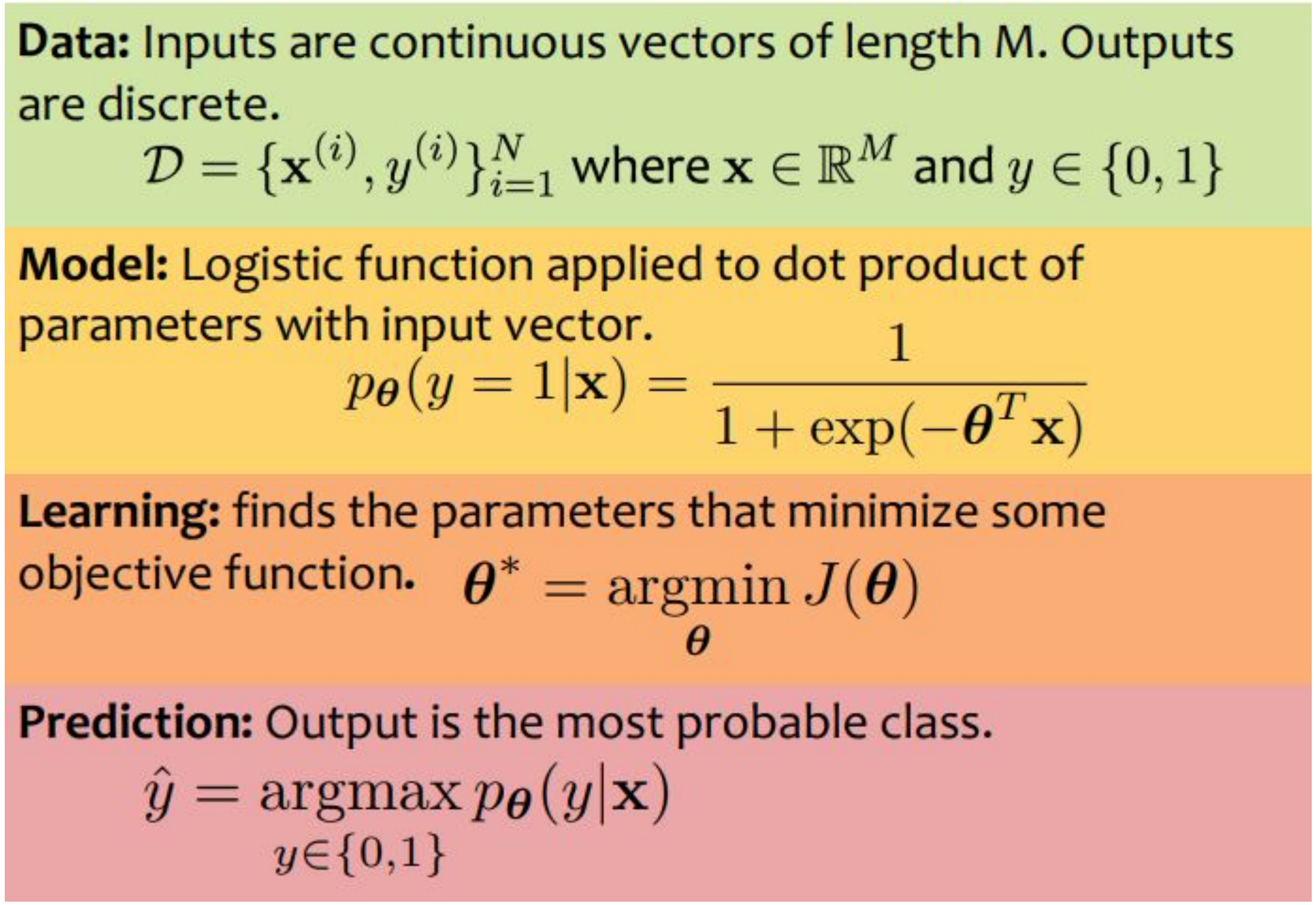
We can now use <u>**gradient ascent**</u> to maximize j(Θ) The update rule will be:
repeat until convergence

$\{$

$$\theta_j = \theta_j + \alpha \sum_{i=1}^{m} \left(y_i - \frac{1}{1+e^{-\theta^t x_i}} \right) x_{ij}$$

$\}$

Logistic Regression



Data: Inputs are continuous vectors of length M. Outputs are discrete.

$$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N} \text{ where } \mathbf{x} \in \mathbb{R}^M \text{ and } y \in \{0,1\}$$

Model: Logistic function applied to dot product of parameters with input vector.

$$p_{\boldsymbol{\theta}}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})}$$

Learning: finds the parameters that minimize some objective function. $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$

Prediction: Output is the most probable class.

$$\hat{y} = \underset{y \in \{0,1\}}{\operatorname{argmax}}\, p_{\boldsymbol{\theta}}(y|\mathbf{x})$$