Deep learning

Dr. Aissa Boulmerka
a.boulmerka@centre-univ-mila.dz

2023-2024

CHAPTER 1
LOGISTIC REGRESSION

Introduction to Deep Learning

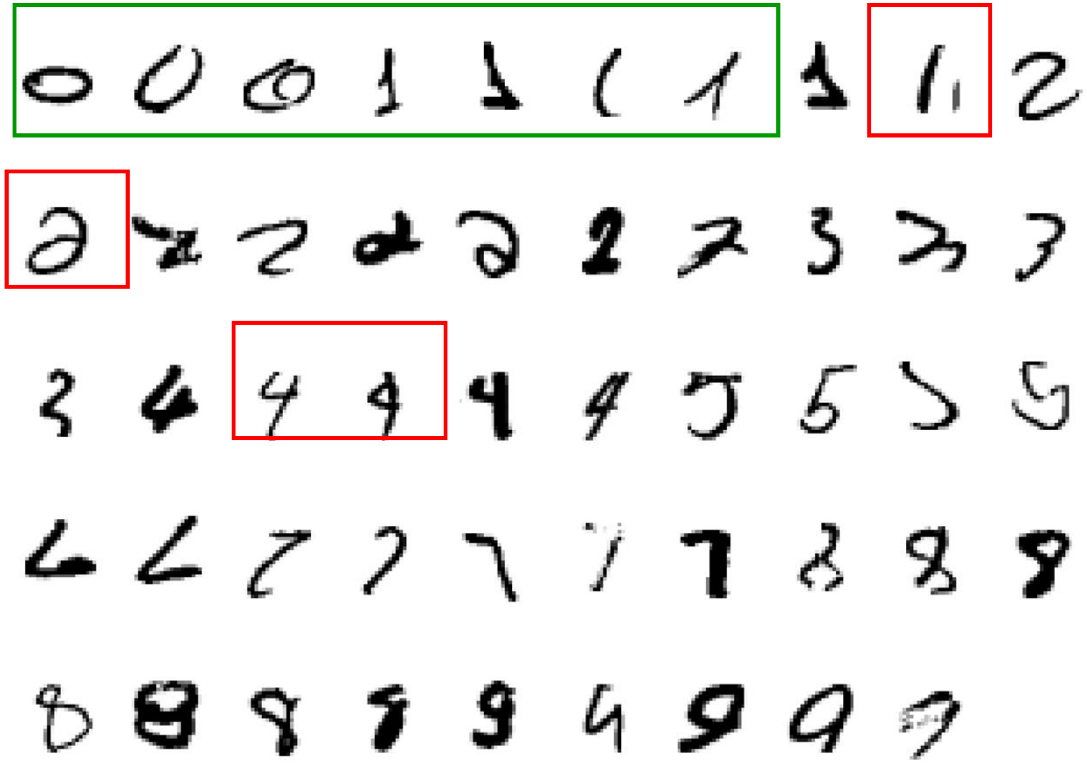
- Artificial Intelligence is the new Electricity.
- Electricity had once transformed countless industries: transportation, manufacturing, healthcare, communications, and more
- AI will now bring about an equally big transformation.

Andrew Ng

The Machine Learning Approach

- Instead of writing a program by hand for each specific task, we collect lots of examples that specify the correct output for a given input.
- A machine learning algorithm then takes these examples and produces a program that does the job.
  - The program produced by the learning algorithm may look very different from a typical hand-written program. It may contain millions of numbers.
  - If we do it right, the program works for new cases as well as the ones we trained it on.
  - If the data changes the program can change too by training on the new data.
- Massive amounts of computation are now cheaper than paying someone to write a task-specific program.

It is very hard to say what makes a 2



Some examples of tasks best solved by learning

- Recognizing patterns:
  - Objects in real scenes
  - Facial identities or facial expressions
  - Spoken words
- Recognizing anomalies:
  - Unusual sequences of credit card transactions
  - Unusual patterns of sensor readings in a nuclear power plant
- Prediction:
  - Future stock prices or currency exchange rates
  - Which movies will a person like?

Types of learning task

- Supervised learning
  - Learn to predict an output when given an input vector.
- Reinforcement learning
  - Learn to select an action to maximize payoff.
- Unsupervised learning
  - Discover a good internal representation of the input.

What you will learn in this course?

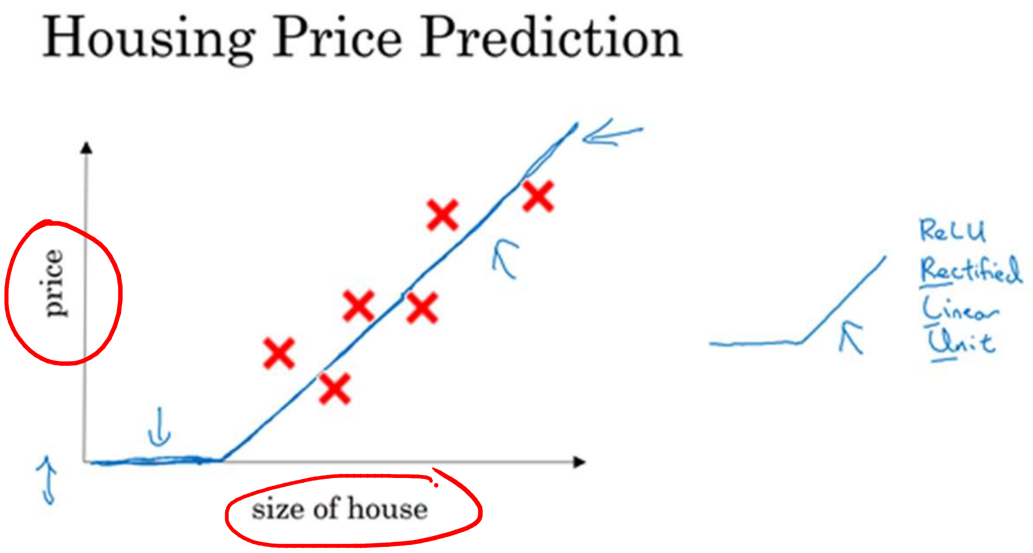
1. Neural Networks and Deep Learning
2. Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization
3. Structuring your Machine Learning project
4. Convolutional Neural Networks
5. Sequential Models such as Natural Language Processing and Time series.

What is neural network?

It is a powerful learning algorithm inspired by how the brain works.

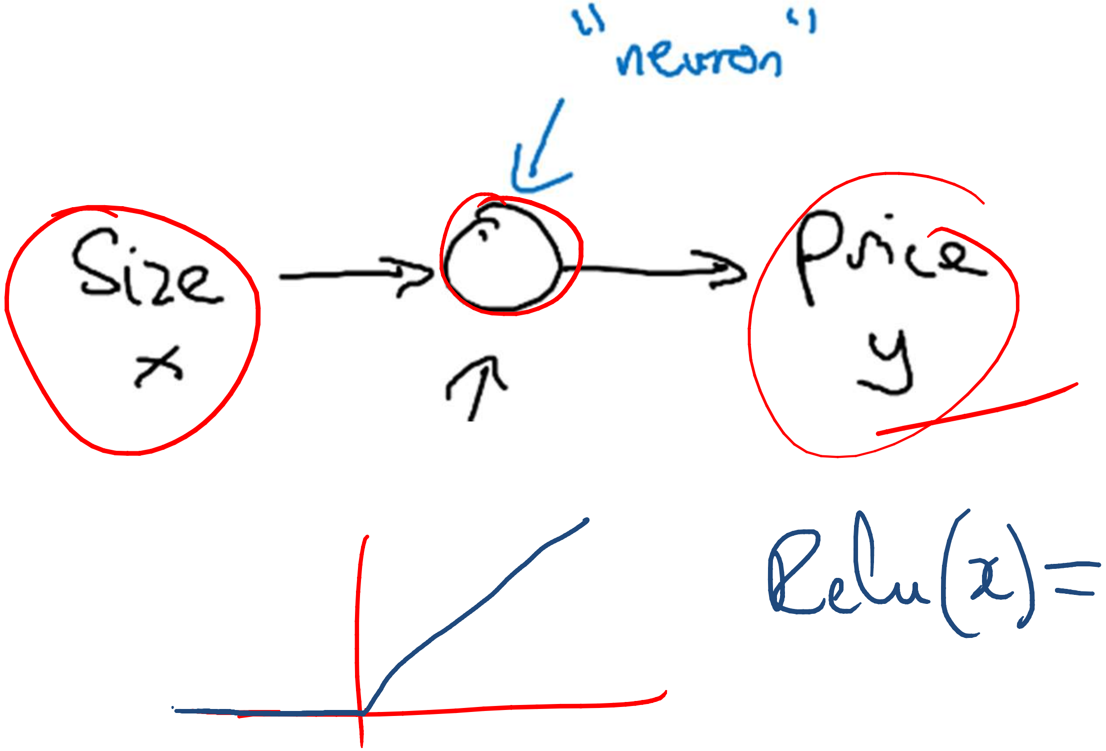
Example 1 – single neural network

- Given data about the size of houses on the real estate market and you want to fit a function that will predict their price. It is a linear regression problem because the price as a function of size is a continuous output.
- We know the prices can never be negative so we are creating a function called Rectified Linear Unit (ReLU) which starts at zero.



Example 1 – single neural network

- The input is the size of the house (x)
- The output is the price (y)
- The “neuron” implements the function ReLU (blue line)



$$\mathrm{Relu}(x) = \begin{cases} x & \text{si } x > 0 \\ 0 & \underline{\text{sinon}} \end{cases}$$

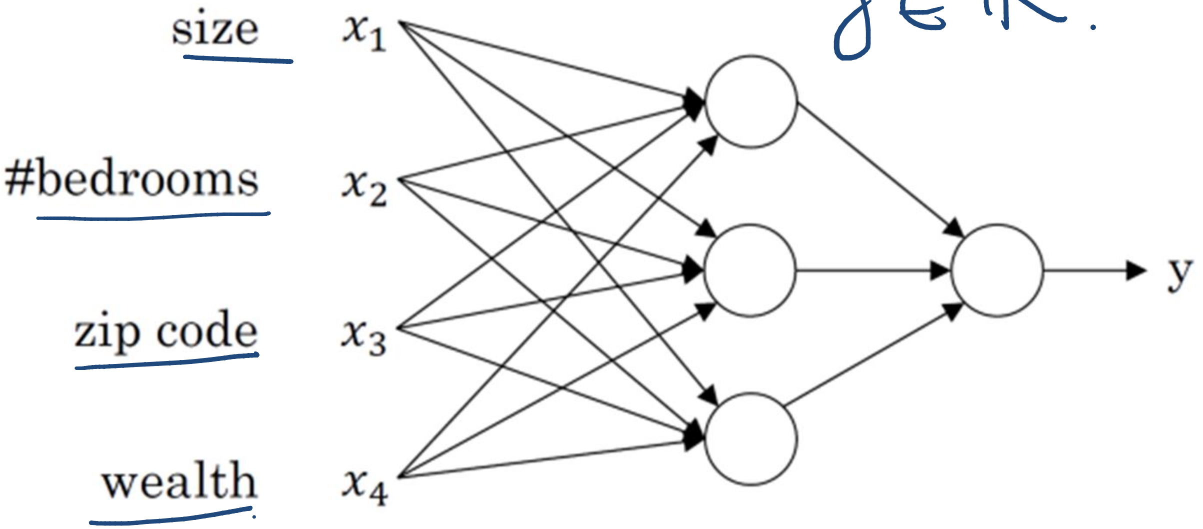
Example 2 – Multiple neural network

The price of a house can be affected by other features such as size, number of bedrooms, zip code and wealth. The role of the neural network is to predict the price and it will automatically generate the hidden units. We only need to give the inputs x and the output y.

Housing Price Prediction

$x = [x_1, x_2, \dots x_4]$

$y \in \mathbb{R}.$



Supervised learning for Neural Network

- In **supervised learning**, we are given a dataset and already know what our correct output should look like, having the idea that there is a relationship between the input and the output.
- Supervised learning problems are categorized into "**regression**" and "**classification**" problems.
- In a regression problem, we are trying to predict results within a **continuous output**, meaning that we are trying to map input variables to some continuous function.
- In a classification problem, we are instead trying to predict results in a **discrete output**. In other words, we are trying to map input variables into discrete categories.

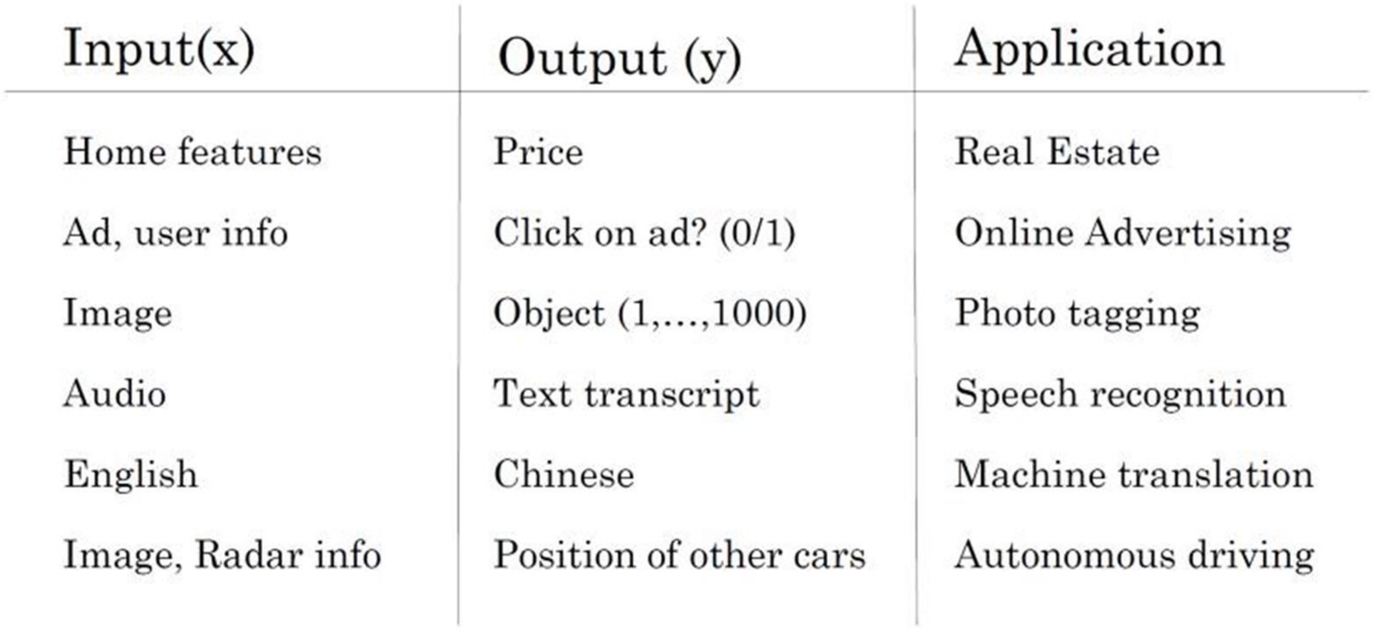
$\boxed{1} \rightarrow 1$

dataset : $\{(\underline{x}_i, y_i) : i \in 1 \ldots N\}$

$y \in \mathbb{R}$.

Examples of supervised learning

- Here are some examples of supervised learning



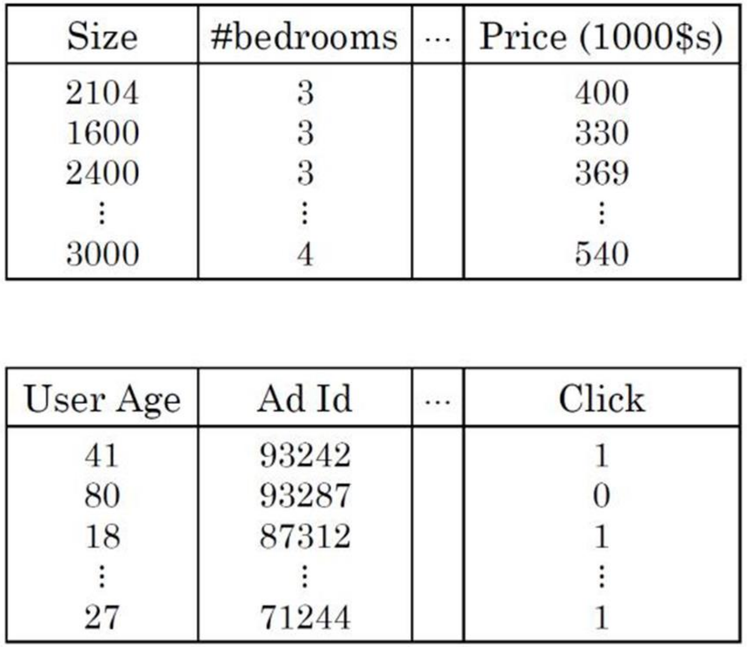
Input(x)	Output (y)	Application
Home features	Price	Real Estate
Ad, user info	Click on ad? (0/1)	Online Advertising
Image	Object (1,...,1000)	Photo tagging
Audio	Text transcript	Speech recognition
English	Chinese	Machine translation
Image, Radar info	Position of other cars	Autonomous driving

- There are different types of neural network, for example :
  - **Convolution Neural Network (CNN):** used often for image application
  - **Recurrent Neural Network (RNN):** used for one-dimensional sequence data such as translating English to Chinese or a temporal component such as text transcript.
  - **Hybrid neural network architecture :** for the autonomous driving.

Structured vs unstructured data

- **Structured data** refers to things that has a defined meaning such as price, age.
- **Unstructured data** refers to thing like pixel, raw audio, text.

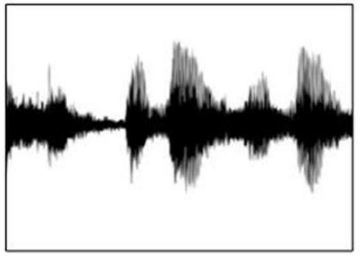
Structured Data



Size	#bedrooms	⋯	Price (1000$s)
2104	3		400
1600	3		330
2400	3		369
⋮	⋮		⋮
3000	4		540

User Age	Ad Id	⋯	Click
41	93242		1
80	93287		0
18	87312		1
⋮	⋮		⋮
27	71244		1

Unstructured Data



Audio

Image

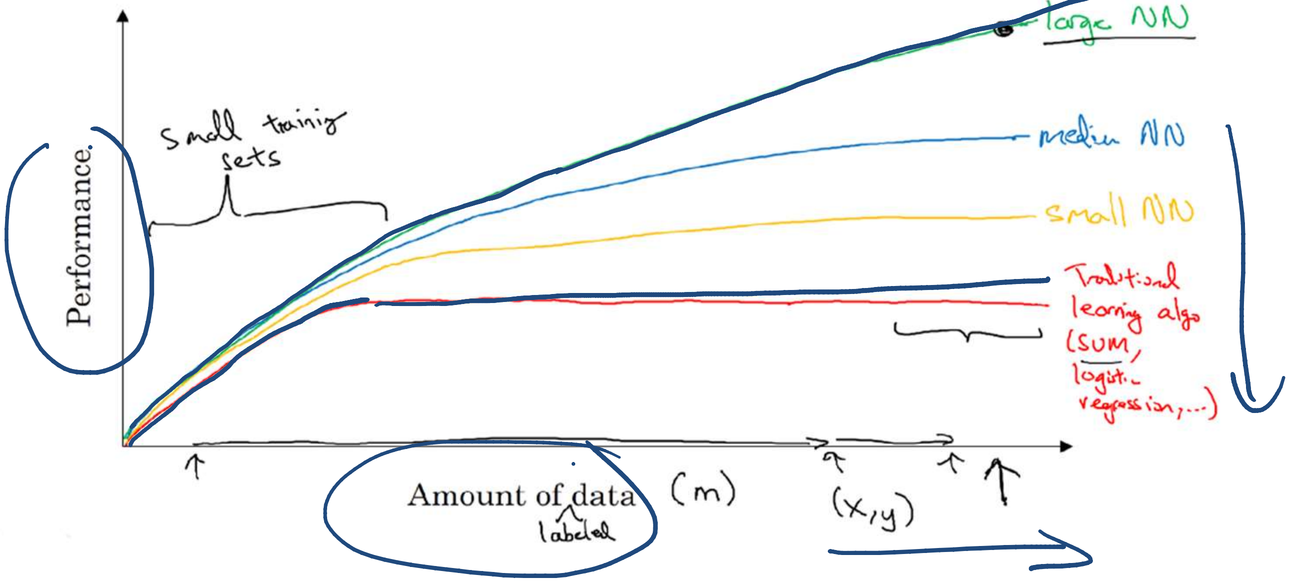
Four scores and seven years ago…

Text

Why is deep learning taking off?

- Deep learning is taking off due to 1 a large amount of data available through the digitization of the society, 2 faster computation and 3 innovation in the development of neural network algorithm.

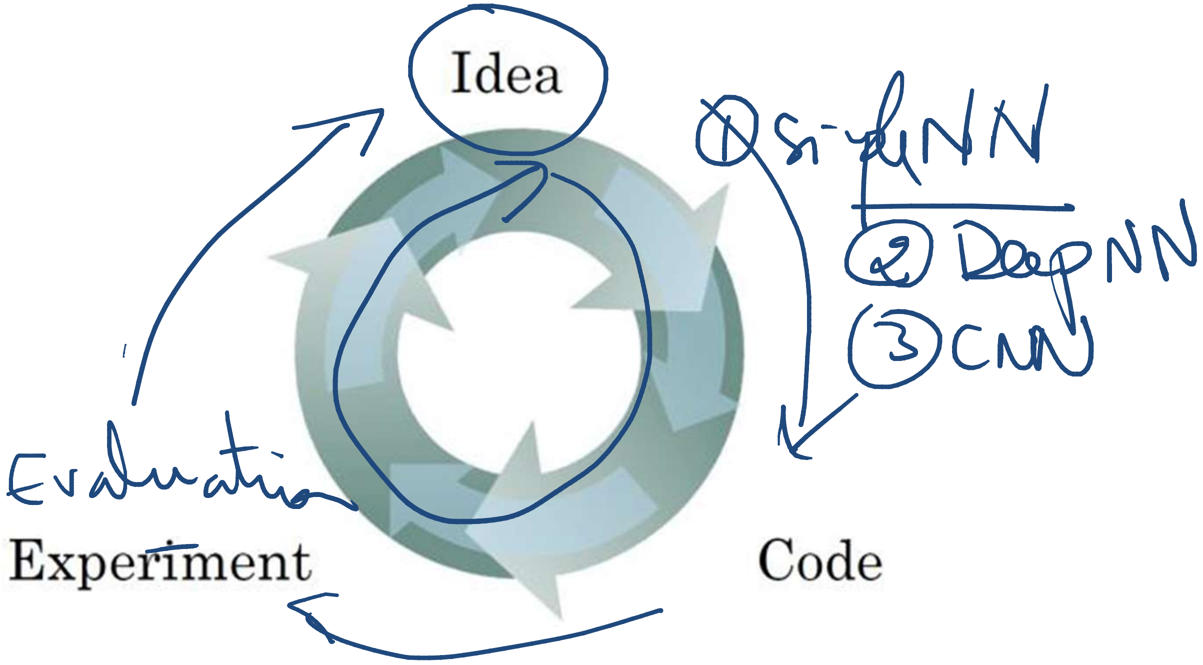
Scale drives deep learning progress



- Two things have to be considered to get to the high level of performance:
  1. Being able to train a big enough neural network
  2. Huge amount of labeled data

Training a neural network

- The process of training a neural network is iterative.



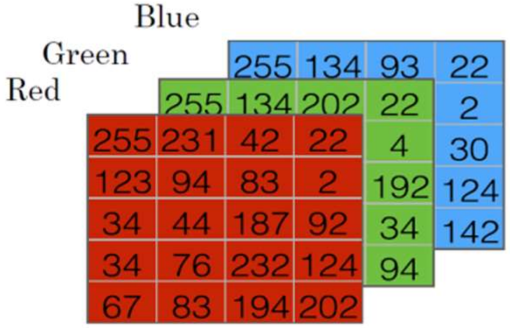
- It could take a good amount of time to train a neural network, which affects your productivity. Faster computation helps to iterate and improve new algorithm.

Binary Classification

- In a binary classification problem, the result is a discrete value output.
- For example
  - account hacked (1) or compromised (0)
  - a tumor malign (1) or benign (0)

Example: Cat vs Non-Cat

- The goal is to **train a classifier** that the input is an image represented by a feature vector, x, and predicts whether the corresponding label y is 1 or 0. In this case, whether this is a cat image (1) or a non-cat image (0).



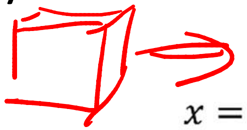
		255	134	93	22
	255	134	202	22	2
255	231	42	22	4	30
123	94	83	2	192	124
34	44	187	92	34	142
34	76	232	124	94	
67	83	194	202		

Binary Classification

- An **image** is store in the computer in three separate matrices corresponding to the Red, Green, and Blue color channels of the image.
- The three matrices have the same size as the image, for example, the resolution of the cat image is **64 pixels** x **64 pixels**, the three matrices (RGB) are **64 X 64** each.
- The value in a cell represents the pixel intensity which will be used to create a feature vector of n dimension.
- In pattern recognition and machine learning, a feature vector represents an object, in this case, a cat or no cat.

  - To create a feature vector, x, the pixel intensity values will be "unroll" or "reshape" for each color.
  - The dimension of the input feature vector x is:

$$\boldsymbol{n_x = 64 \times 64 \times 3 = 12\,288.}$$

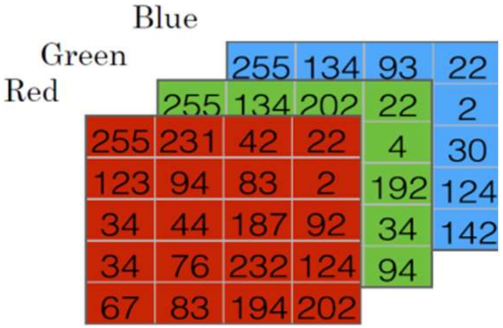


$$x = \begin{bmatrix}255\\231\\42\\\vdots\\255\\134\\202\\\vdots\\255\\134\\93\\\vdots\end{bmatrix}\begin{matrix}\left.\begin{matrix}\\ \\ \\ \\ \end{matrix}\right\}\text{Red}\\ \left.\begin{matrix}\\ \\ \\ \\ \end{matrix}\right\}\text{Green}\\ \left.\begin{matrix}\\ \\ \\ \\ \end{matrix}\right\}\text{Blue}\end{matrix}$$

Binary Classification

input

→ 1 (cat) vs 0 (non cat)



$$x = \begin{bmatrix} 255 \\ 231 \\ 42 \\ \vdots \\ 255 \\ 134 \\ 202 \\ \vdots \\ 255 \\ 134 \\ 93 \\ \vdots \end{bmatrix}$$

$$64 \times 64 \times 3 = 12288$$

$$n = n_x = 12288$$

$$x \rightarrow y$$

Notation

- (x, y) $\qquad x \in \mathbb{R}^{n_x}, y \in \{0,1\}$

- m training examples: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

- $m = \#$ train examples $\qquad m_{test} = \#$ test examples

D_{train} : m examples

D_{test} : m_{test} "

- $$X = \underbrace{\left[\begin{array}{cccc} | & | & \cdots & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & \cdots & | \end{array}\right]}_{m} \left.\vphantom{\begin{array}{c} | \\ x \\ | \end{array}}\right\} n_x \qquad X \in \mathbb{R}^{n_x \times m} \qquad X.shape = (n_x, m)$$

- $Y = [y^{(1)}, y^{(2)}, \cdots, y^{(m)}] \qquad Y \in \mathbb{R}^{1 \times m}$ $\qquad$ Y.$shape = (1, m)$

Logistic Regression

- Logistic regression is a learning algorithm used in a supervised learning problem when the output y are all either zero or one. The goal of logistic regression is to minimize the error between its predictions and training data.

Example: Cat vs No - cat

- Given an image represented by a feature vector x, the algorithm will evaluate the probability of a cat being in that image.

$$Given\ x, \hat{y} = P(y = 1|x), where\ 0 \leq \hat{y} \leq 1$$

The parameters used in Logistic regression are:

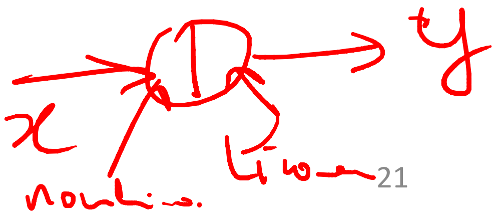
- The input features vector: $x \in \mathbb{R}^{n_x}$, where n_x is the number of features
- The training label: $y \in \{0,1\}$
- The weights: $w \in \mathbb{R}^{n_x}$, where n_x is the number of features
- The threshold: $b \in \mathbb{R}$
- The output: $\hat{y} = \sigma(w^T x + b)$

w^Tx+b

- Sigmoid function: $s = \sigma(w^T x + b) = \sigma(z) = \frac{1}{(1+ e^{-z})}$

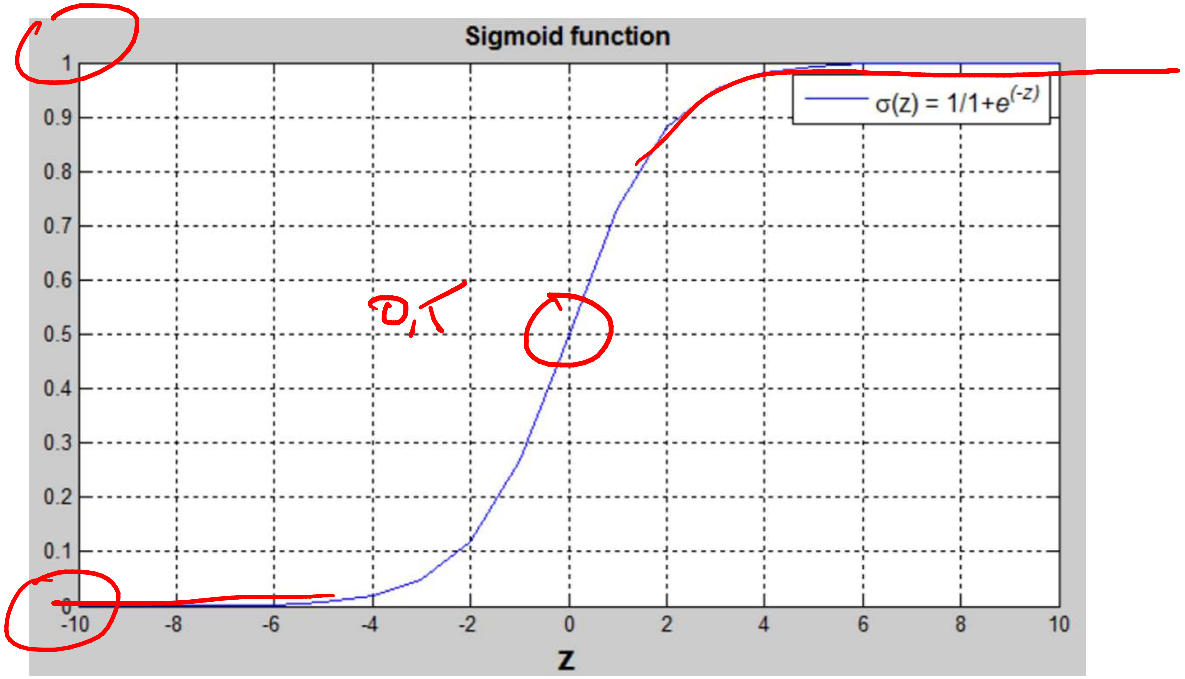
non-linear

linear



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

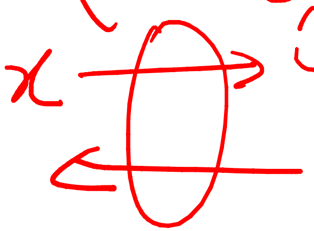
Logistic Regression



0

- $(w^T x + b)$ is a linear function $(ax + b)$, but since we are looking for a probability constraint between $[0,1]$, the sigmoid function is used. The function is bounded between $[0,1]$ as shown in the graph above.
- Some observations from the graph:
  - If z is a large positive number, then $\sigma(z) = 1$
  - If z is small or large negative number, then $\sigma(z) = 0$
  - If $z = 0$, then $\sigma(z) = 0.5$

Logistic Regression (x, y)



$\mathcal{L}(y, \hat{y})$

→ $x \in \mathbb{R}^{n_x}$

$Given\ x,$ $\hat{y} = P(y = 1|x)$

$x \in \mathbb{R}^{n_x}$ $0 \leq \hat{y} \leq 1$ ✓

Parameters: $w \in \mathbb{R}^{n_x}, b \in \mathbb{R}$ ✓

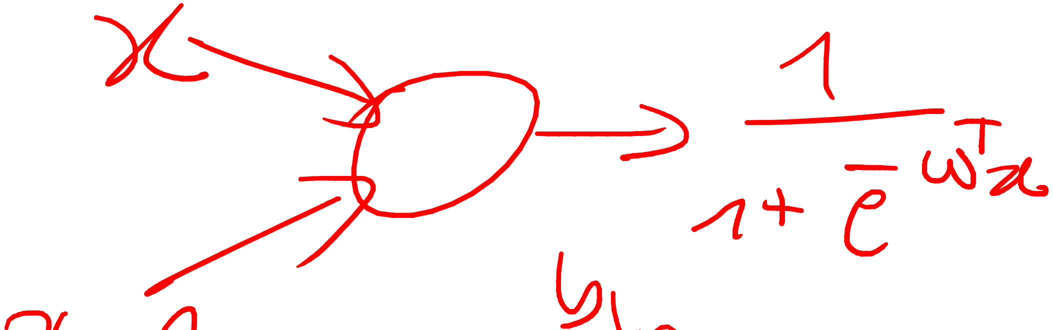
Output: $\hat{y} = \sigma(w^T x + b)$

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

$x_0 = 1,$ $x \in \mathbb{R}^{n_x + 1}$

$\hat{y} = \sigma(\theta^T x)$

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix}$$



$w = [\overset{b}{w_0}, w_1 \cdots w_{n_x}]$

Logistic regression cost function

- To train the parameters w and b, we need to define a cost function.

Recap:

- $\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where $\sigma(z^{(i)}) = \frac{1}{(1+ e^{-z^{(i)}})}$ $\quad z^{(i)} = w^T x^{(i)} + b$
- Gieven $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$, we want : $\hat{y}^{(i)} \approx y^{(i)}$

prediction $(\hat{y})$

Ground truth (y)

Logistic regression cost function

$\text{Dataset} = \{(x^{(i)}, y^{(i)})\ i=1..m\}$

Loss (error) function:

The loss function computes the error for a single training example.

- $\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2$ (mean squared error) RMSE
- $\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -(y^{(i)} log(\hat{y}^{(i)}) + (1 - y^{(i)}) log(1 - \hat{y}^{(i)}))$ (cross-entropy)
  - If $y^{(i)} = 1$: $\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -log(\hat{y}^{(i)})$ where $log(\hat{y}^{(i)})$ =0 and $\hat{y}^{(i)}$ should be close to 1
  - If $y^{(i)} = 0$: $\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -log(1 - \hat{y}^{(i)})$ where $log(1 - \hat{y}^{(i)})$=0 and $\hat{y}^{(i)}$ should be close to 0

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

$$E = \frac{1}{2}(\hat{y}-y)^2$$

$E = 0$ ✗

y	$\hat{y}$
0	0
1	1
0	1
1	0

$$-\left(y\log\hat{y} + (1-y)\log(1-\hat{y})\right)$$

$$-\left(0\,\cancel{\log}\,1 + \underset{1}{(1-0)}\log(1-1)\right)$$

$$E = -\log 0$$

$$E = +\infty$$

Logistic regression cost function

Cost function

- The cost function is the average of the loss function of the entire training set. The goal is to find the parameters w and b that minimize the overall cost function.

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

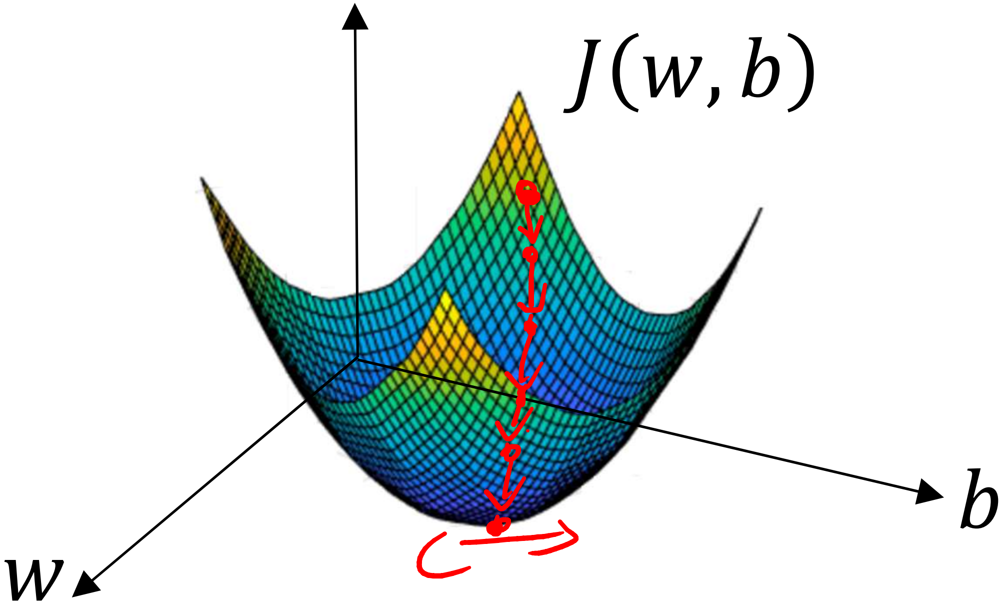
Gradient Descent

- Recap:

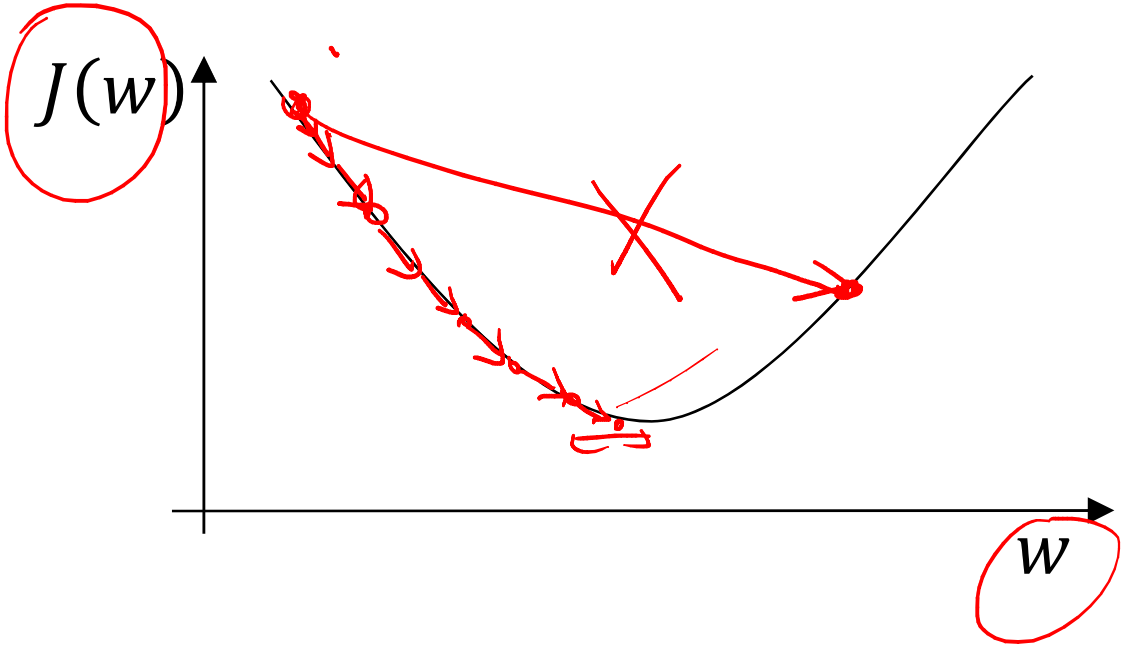
$$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b), \text{ where } \sigma(z^{(i)}) = \frac{1}{(1+ e^{-z^{(i)}})}$$

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

Want to find w, b that minimize $J(w,b)$

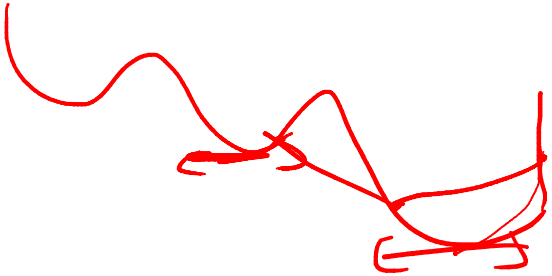


Gradient Descent



```
Repeat {
    $w \leftarrow w - \alpha \frac{dJ}{dw}$
}
```

α: Learning rate



$J(w,b)$: cost function

$$\frac{\partial J(w,b)}{\partial w} = \frac{dJ(w,b)}{dw} = dw$$

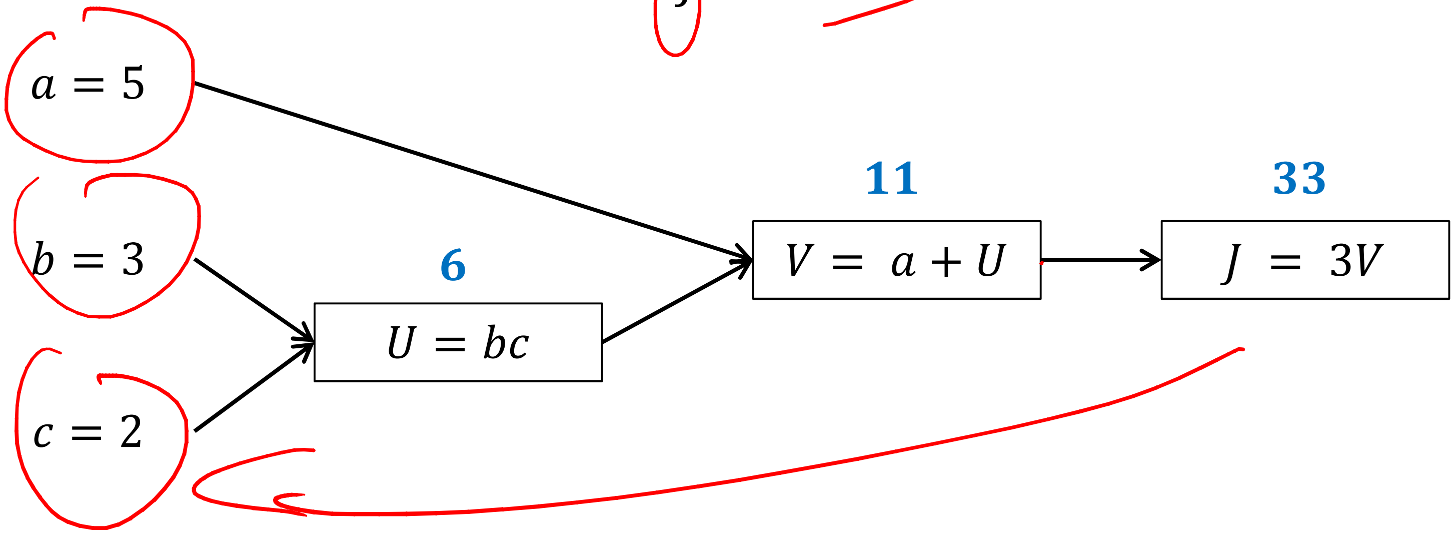
$$\frac{\partial J(w,b)}{\partial b} = \frac{dJ(w,b)}{db} = db$$

DAG

Computation Graph

$$J(a,b,c) = 3(a+bc) = 3(5 + 3*2) = 33$$

$$U = bc$$
$$V = a + U$$
$$J = 3V$$



$\frac{dJ}{dc} = \frac{dJ}{dv} \cdot \frac{dv}{du} \cdot \frac{du}{dc} = 9$

$\uparrow 3 \quad 1 \quad b=3$

Computation Graph

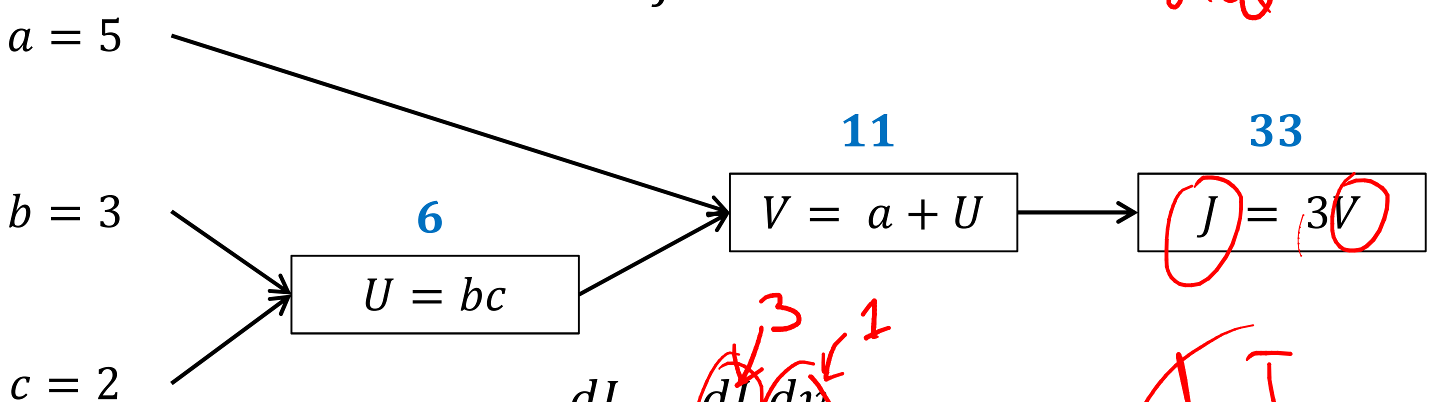
$J(a,b,c) = 3(\underline{a} + \underline{b}\underline{c}) = 3(5 + 3 * 2) = 33$

$\frac{dJ}{dc} = \frac{dJ}{dv} \frac{dv}{dc}$

3

$U = bc$
$V = a + U$
$J = 3V$

$\frac{dJ}{dv} = 3$



$$\frac{dJ}{du} = \overset{3}{\frac{dJ}{dv}}\overset{1}{\frac{dv}{du}} = 3$$

$$\frac{dJ}{db} = \frac{dJ}{du}\frac{du}{db} = 6$$

$$\frac{dJ}{dc} = \frac{dJ}{du}\frac{du}{dc} = 9$$

$\frac{dJ}{da} = \frac{dJ}{dv} \cdot \frac{dv}{da}$

$\uparrow 3 \quad \uparrow 1$

$= 3$

$\frac{dJ}{da}$

$\frac{dJ}{db}$

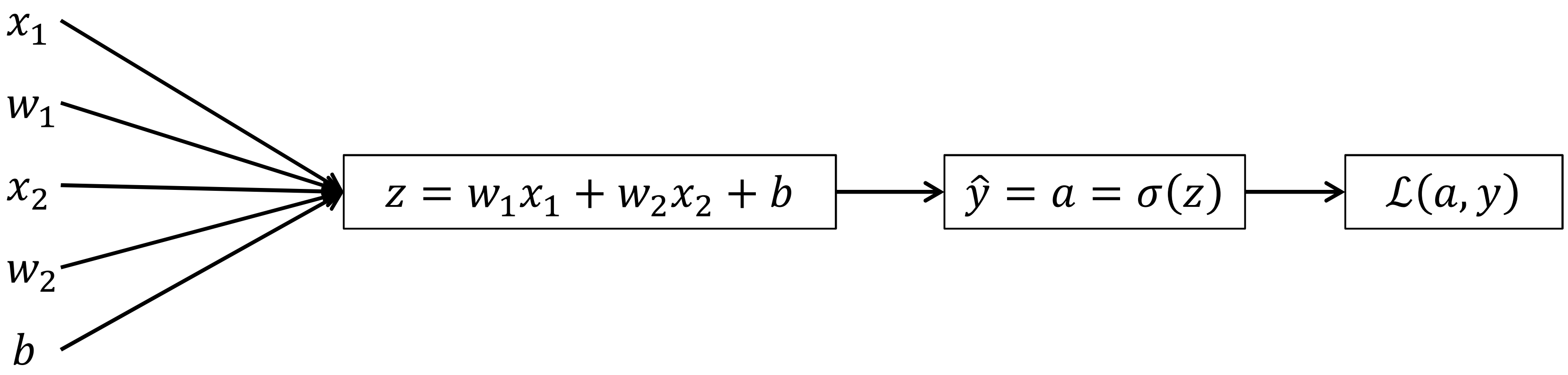
$\frac{dJ}{dc}$

Logistic Regression Gradient descent

$$z = w^T x + b,$$

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

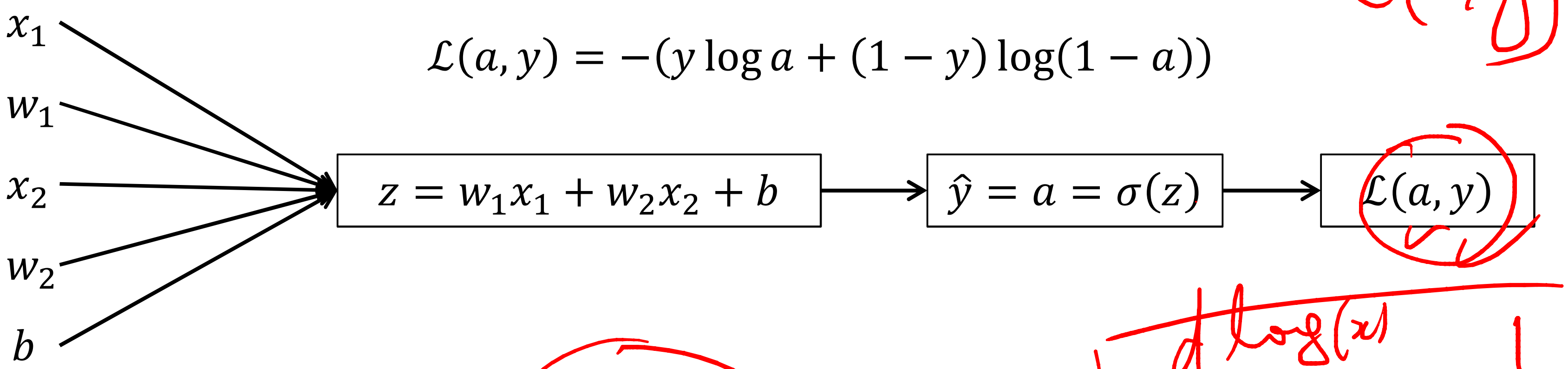
$$\mathcal{L}(a, y) = -(y \log a + (1 - y) \log(1 - a))$$



Logistic Regression Gradient descent

$\mathcal{L}(a, y)$

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



$$\boxed{\frac{d \log(x)}{dx} = \frac{1}{x}}$$

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

$$\frac{da}{dz} = a(1-a)$$

$$dz = \frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}}{da}\frac{da}{dz} = a - y$$

$$dw_1 = \frac{d\mathcal{L}}{dw_1} = x_1 dz$$

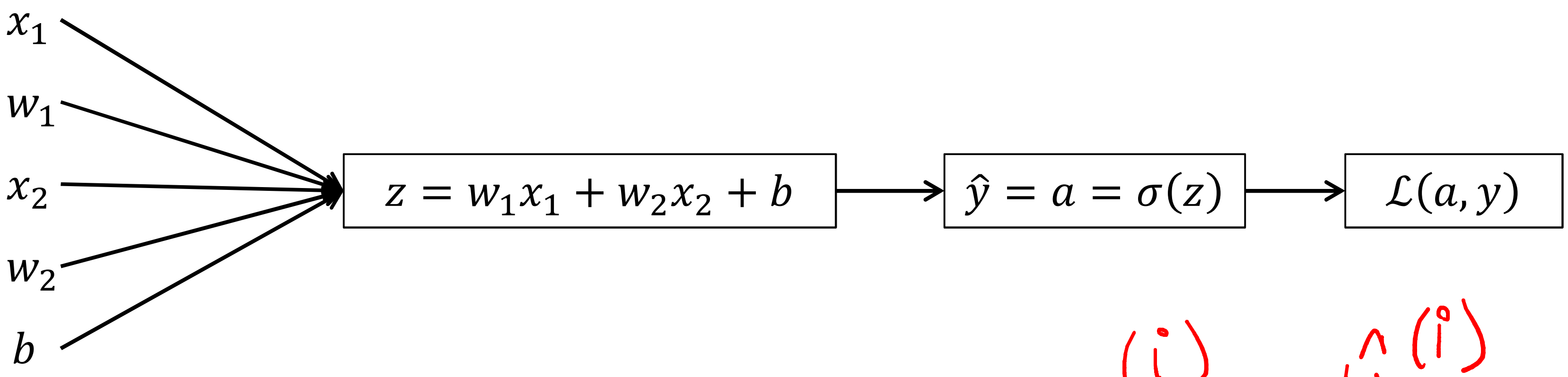
$$dw_2 = \frac{d\mathcal{L}}{dw_2} = x_2 dz$$

$$db = \frac{d\mathcal{L}}{db} = dz$$

$$\left\{\begin{aligned} w_1 &\leftarrow w_1 - \alpha dw_1 \\ w_2 &\leftarrow w_2 - \alpha dw_2 \\ b &\leftarrow b - \alpha db \end{aligned}\right.$$

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

Logistic regression on m examples



$a^{(i)} = \hat{y}^{(i)}$

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

$$a^{(i)} = \hat{y}^{(i)} = \sigma\left(z^{(i)}\right) = \sigma\left(w^T x^{(i)} + b\right)$$

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

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

Logistic regression on m examples (iterative)

```
$J = 0; dw_1 = 0; dw_2 = 0; db = 0;$
for i=1 to m:
    $z^{(i)} = w^T x^{(i)} + b;$
    $a^{(i)} = \sigma(z^{(i)});$
    $J \mathrel{+}= -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})];$
    $dz^{(i)} = a^{(i)} - y^{(i)};$
    $dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)};$
    $dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)};$
    $db \mathrel{+}= dz^{(i)};$
$J /= m; dw_1 /= m; dw_2 /= m; db /= m;$
$w_1 = w_1 - \alpha dw_1;$
$w_2 = w_2 - \alpha dw_2;$
$b = b - \alpha db;$
```

$J = J +$

Forward

$\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Backward

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

What is vectorization?

$$z = w^T x + b$$

$$w = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \quad w \in \mathbb{R}^{n_x} \qquad\qquad x = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \quad x \in \mathbb{R}^{n_x}$$

Iterative (non-vectorial)	Vectorial

`z=0;`
`for i in range(nx):`
`    z+=w[i]*x[i];`
`z+=b;`
	`z = np.dot(w,x) + b`

Neural network programming guideline

Whenever possible, avoid explicit for loops

Iterative (non-vectorial)	Vectorial

`U = Av`
$U_i = \sum_i \sum_j A_{ij} v_j$
`U = np.zeros((n,1))`
`for i=1...`
`    for j=1...`
`    u[i] += A[i][j]*v[j];`
	`u = np.dot(A,v)`

Vectors and matrix valued functions

Say you need to apply the exponential operation on every element of a matrix/vector.

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ v_n \end{bmatrix} \longrightarrow u = \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \\ \vdots \\ e^{v_n} \end{bmatrix}$$

Iterative (non-vectorial)	Vectorial

`u = np.zeros((n,1))`
`for i in range(n):`
`    u[i] = math.exp(v[i])`
	`import numpy as np`
	`u = np.exp(v)`
	
	`# other functions:`
	`np.log(v)`
	`np.abs(v)`
	`np.maximum(v)`

Logistic regression derivatives

Iterative (non-vectorial)	Vectorial
J = 0; dw1 = 0; dw2 = 0; db = 0; for i=1 to m: $z^{(i)} = w^T x^{(i)} + b$ $a^{(i)} = \sigma(z^{(i)})$ $J += -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$ $dz^{(i)} = a^{(i)} - y^{(i)}$ $\mathbf{dw1^{(i)} += x1^{(i)} dz^{(i)}}$ $\mathbf{dw2^{(i)} += x2^{(i)} dz^{(i)}}$ $db += dz^{(i)}$ $J/=m;$ $dw1/=m;$ $dw2/=m;$ $db/=m;$	J = 0; dw = np.zeros((n_x,1)); db = 0; for i=1 to m: $z^{(i)} = w^T x^{(i)} + b$ $a^{(i)} = \sigma(z^{(i)})$ $J += -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$ $dz^{(i)} = a^{(i)} - y^{(i)}$ $\mathbf{dw += x^{(i)} dz^{(i)}}$ $db += dz^{(i)}$ $J/=m;$ $dw/=m;$ $db/=m;$

Vectorizing Logistic Regression

$$\boxed{\begin{aligned} z^{(1)} &= w^T x^{(1)} + b, \\ a^{(1)} &= \sigma\left(z^{(1)}\right) \end{aligned}} \qquad \boxed{\begin{aligned} z^{(2)} &= w^T x^{(2)} + b, \\ a^{(2)} &= \sigma\left(z^{(2)}\right) \end{aligned}} \qquad \boxed{\begin{aligned} z^{(3)} &= w^T x^{(3)} + b, \\ a^{(3)} &= \sigma\left(z^{(3)}\right) \end{aligned}}$$

$$X = \begin{bmatrix} | & | & \cdots & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & \cdots & | \end{bmatrix}$$

$$Z = [\underbrace{z^{(1)} \quad z^{(2)} \quad \ldots \quad z^{(m)}}] = w^T X + [b\ b\ \cdots b] = \left[w^T x^{(1)} + b, w^T x^{(2)} + b, \cdots, w^T x^{(3)} + b\right]$$

➔ **`z = np.dot(w.T,x) + b`**

$$A = [\underbrace{a^{(1)} \quad a^{(2)} \quad \ldots \quad a^{(m)}}] = \sigma(Z)$$

➔ **`A = sigmoid(Z)`**

Implementing Logistic Regression

Iterative (non-vectorial)	Vectorial
for iter in rang(1000): J = 0; dw1 = 0; dw2 = 0; db = 0; for i=1 to m: $z^{(i)} = w^T x^{(i)} + b$ $a^{(i)} = \sigma(z^{(i)})$ $J += -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$ $dz^{(i)} = a^{(i)} - y^{(i)}$ $dw_1 += x_1^{(i)} dz^{(i)}$ $dw_2 += x_2^{(i)} dz^{(i)}$ $db += dz^{(i)}$ $J = J/m$, $dw_1 = dw_1/m$, $dw_2 = dw_2/m$ $db = db/m$	for iter in rang(1000): Z = np.dot(w.T,X)+b $A = \sigma(Z)$ $dZ = A - Y$ $dw = \frac{1}{m} X dZ^T$ $db = \frac{1}{m} \mathrm{np.sum}(dZ)$ w = w - αdw b = b - αdb

References

- Andrew Ng. Deep learning Specialization. Deeplearning.AI.
- Geoffrey Hinton. Neural Networks for Machine Learning.