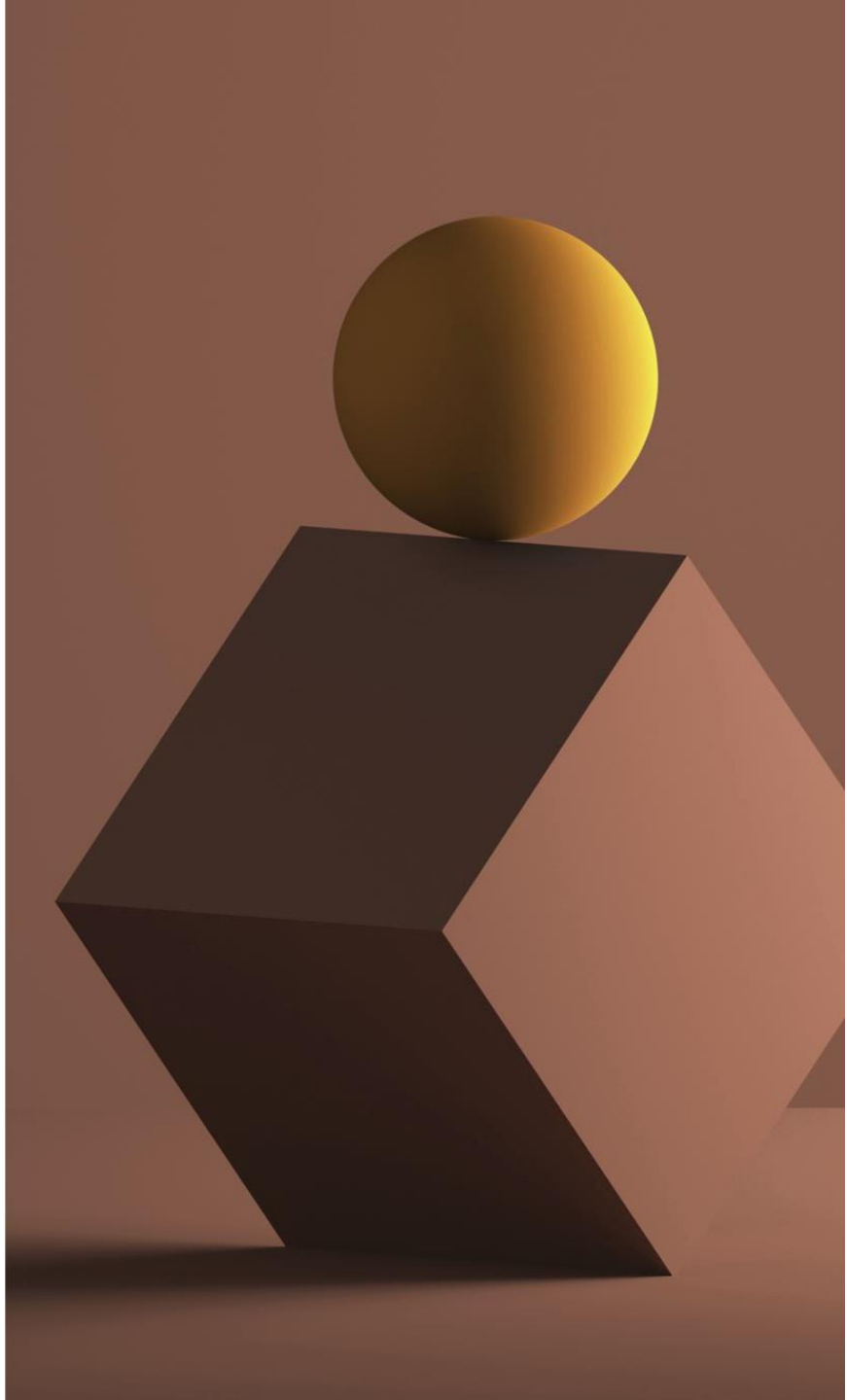
참고 슬라이드

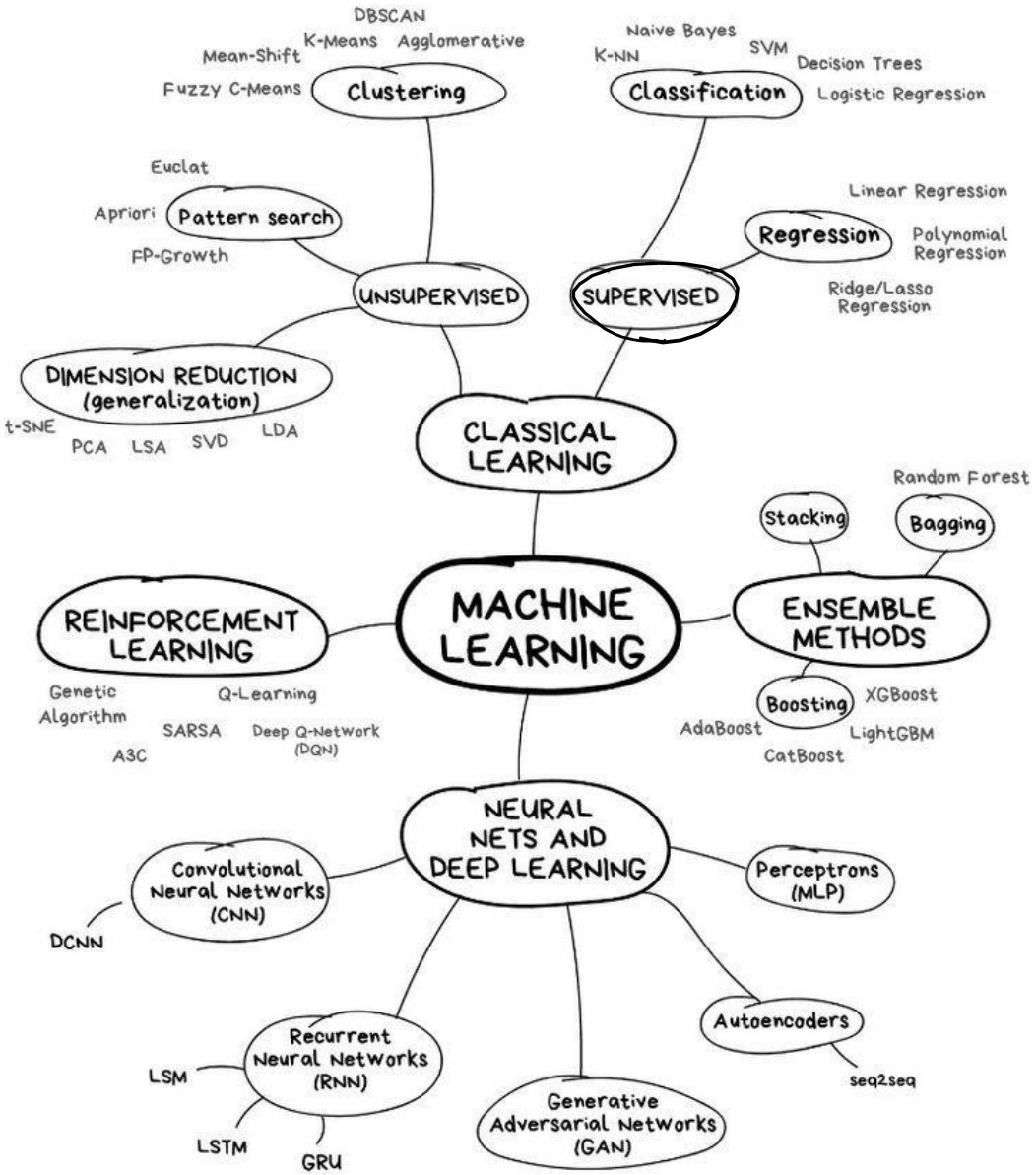
- Stanford CS230
  - https://cs230.stanford.edu/
  - Coursera: https://www.coursera.org/specializations/deep-learning
- 한양대 ERICA 로봇공학과 이영문 교수님 강의자료
- 서울대 전기정보공학부 심규석 교수님 강의자료

Basic models

Artificial Intelligence

Woohwan Jung




MACHINE LEARNING
CLASSICAL LEARNING
UNSUPERVISED
Clustering
DBSCAN
K-Means
Agglomerative
Mean-Shift
Fuzzy C-Means
Pattern search
Euclat
Apriori
FP-Growth
DIMENSION REDUCTION (generalization)
t-SNE
PCA
LSA
SVD
LDA
SUPERVISED
Classification
Naive Bayes
K-NN
SVM
Decision Trees
Logistic Regression
Regression
Linear Regression
Polynomial Regression
Ridge/Lasso Regression
REINFORCEMENT LEARNING
Genetic Algorithm
Q-Learning
SARSA
Deep Q-Network (DQN)
A3C
ENSEMBLE METHODS
Stacking
Bagging
Random Forest
Boosting
AdaBoost
XGBoost
LightGBM
CatBoost
NEURAL NETS AND DEEP LEARNING
Convolutional Neural Networks (CNN)
DCNN
Perceptrons (MLP)
Recurrent Neural Networks (RNN)
LSM
LSTM
GRU
Generative Adversarial Networks (GAN)
Autoencoders
seq2seq

Supervised Learning (input, output) — known

In addition to patterns (inputs) $(X_1, X_2, \cdots X_n)$

We also have access to the variables $(Y_1, Y_2, \cdots, Y_n)$

known input output is called Training data.

The goal is to generalize the input-output relationship.

→ facilitating the prediction of output associated with previously unseen inputs X → Testing data.

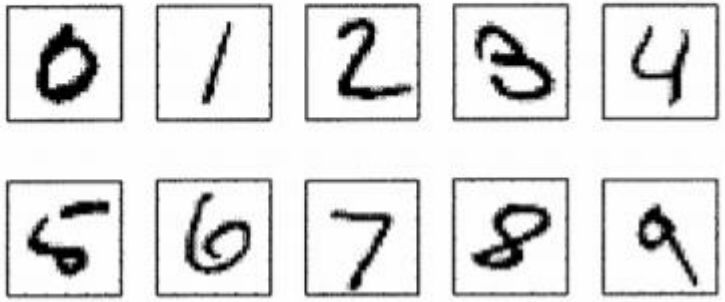
Two primary problems

① classification : $Y \in \{1, \cdots, n\}$ (defined # of labels)

② Regression : $Y \in \mathbb{R}$ (Real #)

Classification

Examples of patterns

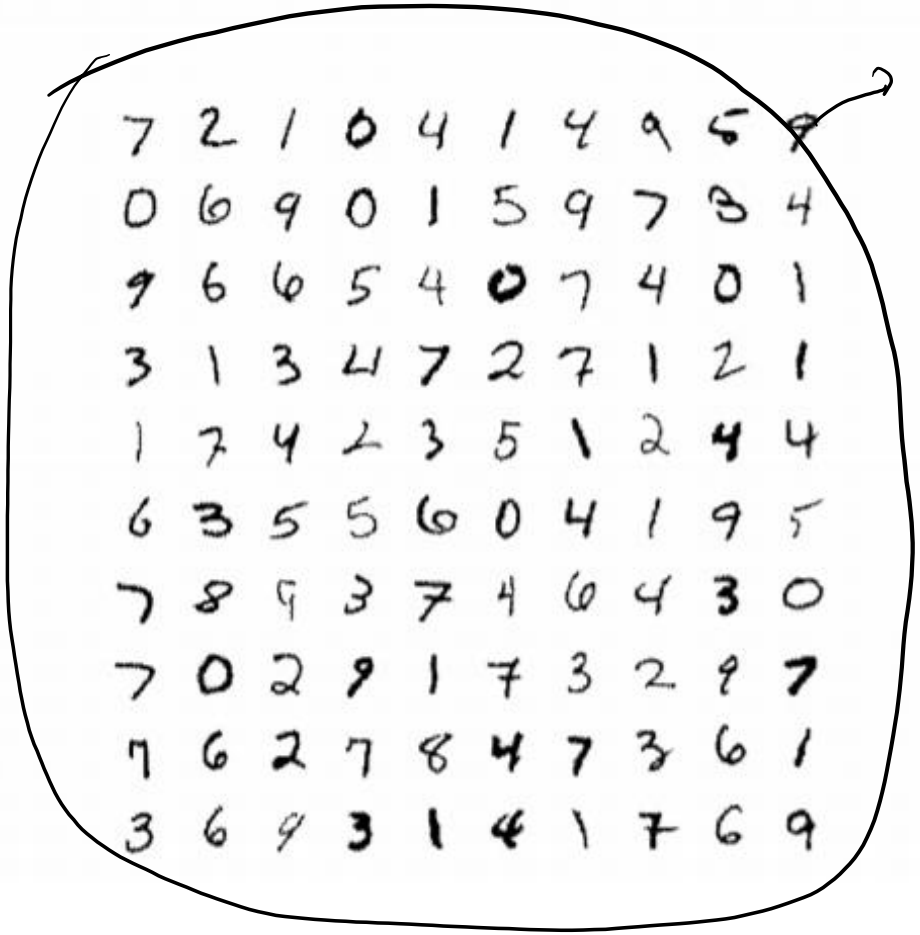


$Y \in \{0, 1, \cdots 9\}$

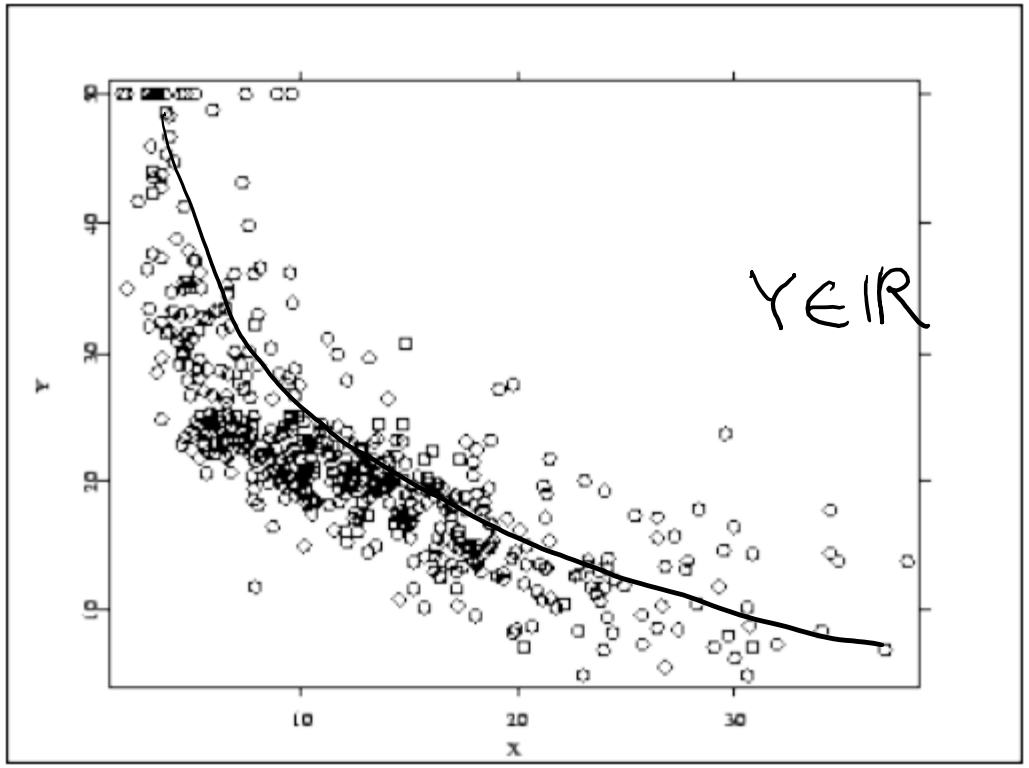
Goal: predict label of a future pattern

$6 \rightarrow$!!

Training data (suppose correct labels are provided)

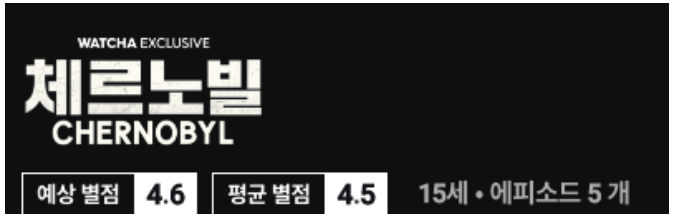


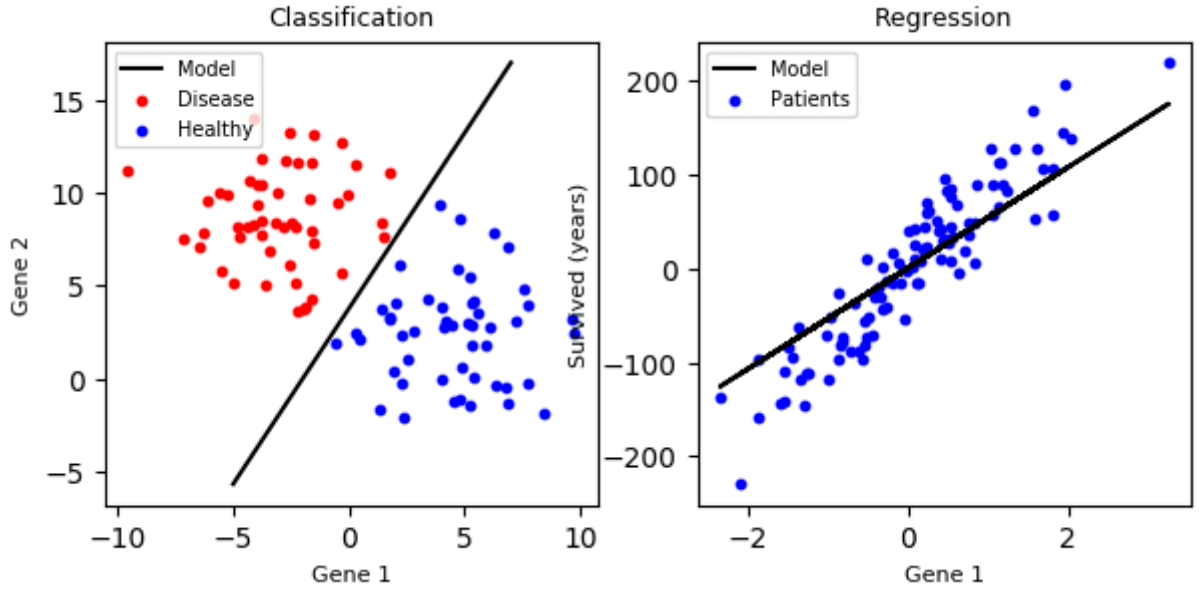
Regression

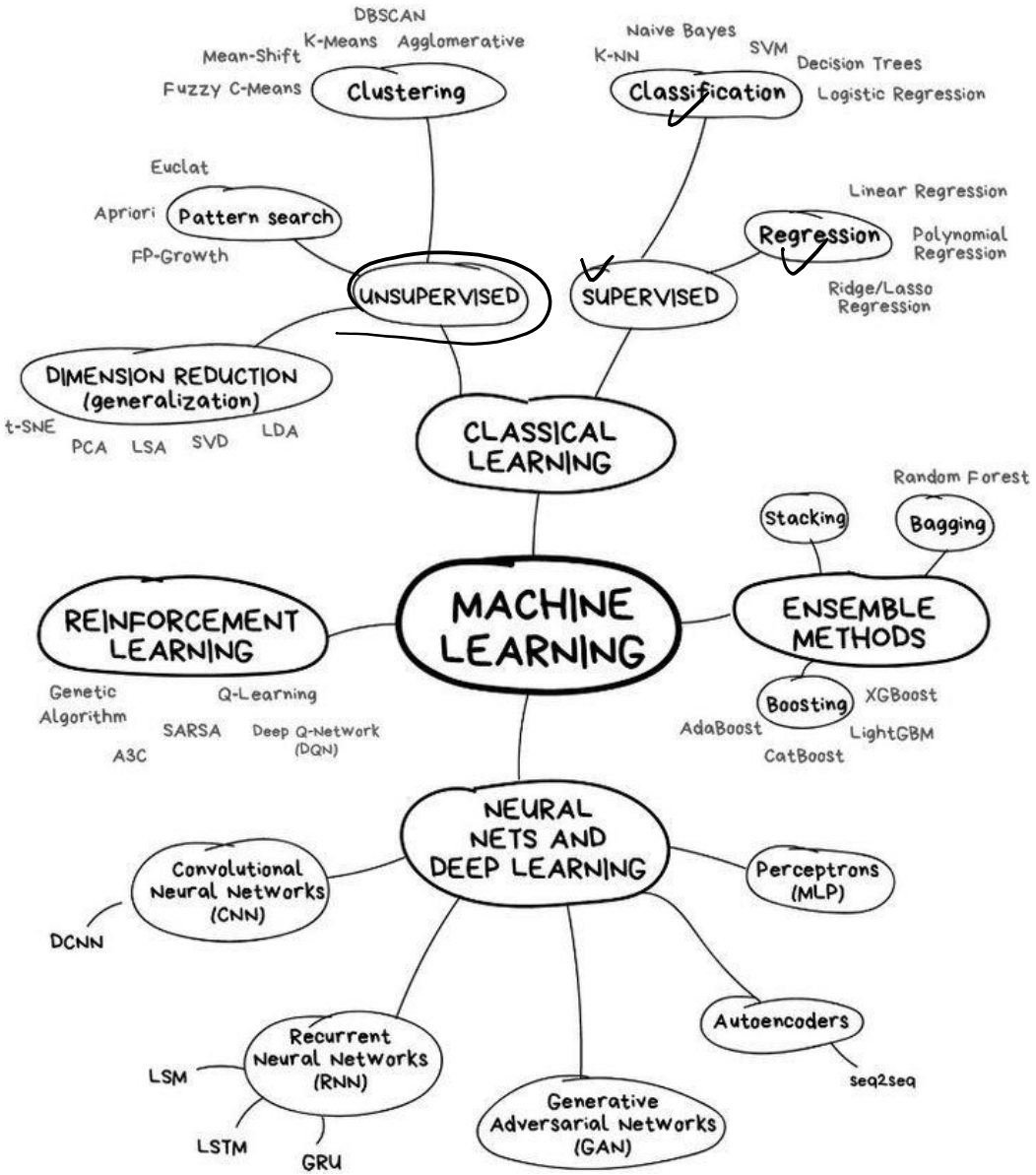


Classification vs Regression

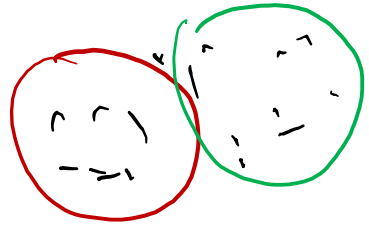
CAT






MACHINE LEARNING
CLASSICAL LEARNING
UNSUPERVISED
Clustering
DBSCAN
K-Means
Agglomerative
Mean-Shift
Fuzzy C-Means
Pattern search
Euclat
Apriori
FP-Growth
DIMENSION REDUCTION (generalization)
t-SNE
PCA
LSA
SVD
LDA
SUPERVISED
Classification
Naive Bayes
K-NN
SVM
Decision Trees
Logistic Regression
Regression
Linear Regression
Polynomial Regression
Ridge/Lasso Regression
REINFORCEMENT LEARNING
Genetic Algorithm
Q-Learning
SARSA
Deep Q-Network (DQN)
A3C
ENSEMBLE METHODS
Stacking
Bagging
Random Forest
Boosting
XGBoost
AdaBoost
LightGBM
CatBoost
NEURAL NETS AND DEEP LEARNING
Convolutional Neural Networks (CNN)
DCNN
Perceptrons (MLP)
Recurrent Neural Networks (RNN)
LSM
LSTM
GRU
Generative Adversarial Networks (GAN)
Autoencoders
seq2seq

Unsupervised Learning



The patterns $X_1, \cdots, X_n$

are NOT accompanied by output variables.

The goal of unsupervised learning is typically not related to future observations.

Instead, we seek to understand structure in the data sample itself,

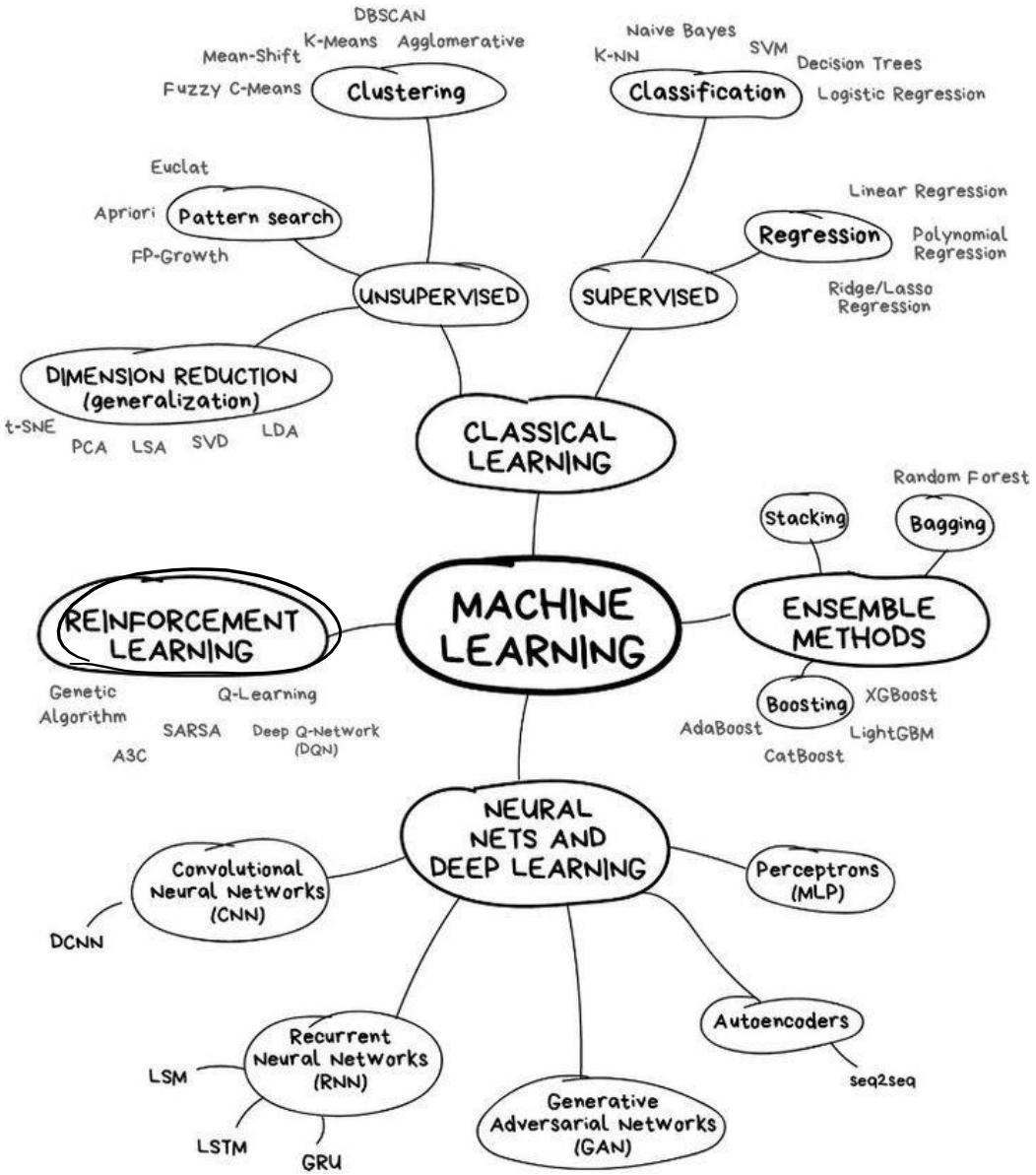
or to infer some characteristic of the underlying probability distribution

The primary problems: ① clustering (e.g. K-means)

② density estimation (Gaussian density)

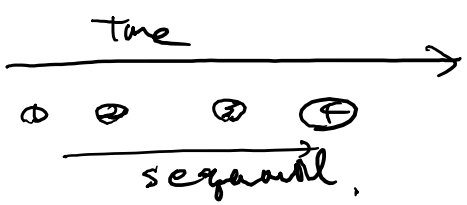
③ dimensionality reduction (Principal component Analysis) : PCA

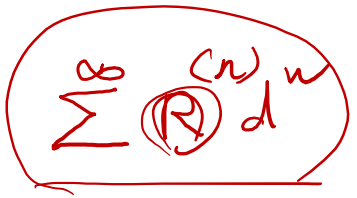
Supervised learning


MACHINE LEARNING
CLASSICAL LEARNING
UNSUPERVISED
Clustering
DBSCAN
K-Means
Agglomerative
Mean-Shift
Fuzzy C-Means
Pattern search
Euclat
Apriori
FP-Growth
DIMENSION REDUCTION (generalization)
t-SNE
PCA
LSA
SVD
LDA
SUPERVISED
Classification
Naive Bayes
K-NN
SVM
Decision Trees
Logistic Regression
Regression
Linear Regression
Polynomial Regression
Ridge/Lasso Regression
REINFORCEMENT LEARNING
Genetic Algorithm
A3C
SARSA
Q-Learning
Deep Q-Network (DQN)
ENSEMBLE METHODS
Stacking
Bagging
Random Forest
Boosting
AdaBoost
XGBoost
LightGBM
CatBoost
NEURAL NETS AND DEEP LEARNING
Convolutional Neural Networks (CNN)
DCNN
Perceptrons (MLP)
Recurrent Neural Networks (RNN)
LSM
LSTM
GRU
Generative Adversarial Networks (GAN)
Autoencoders
seq2seq

games, Robots

Reinforcement Learning (



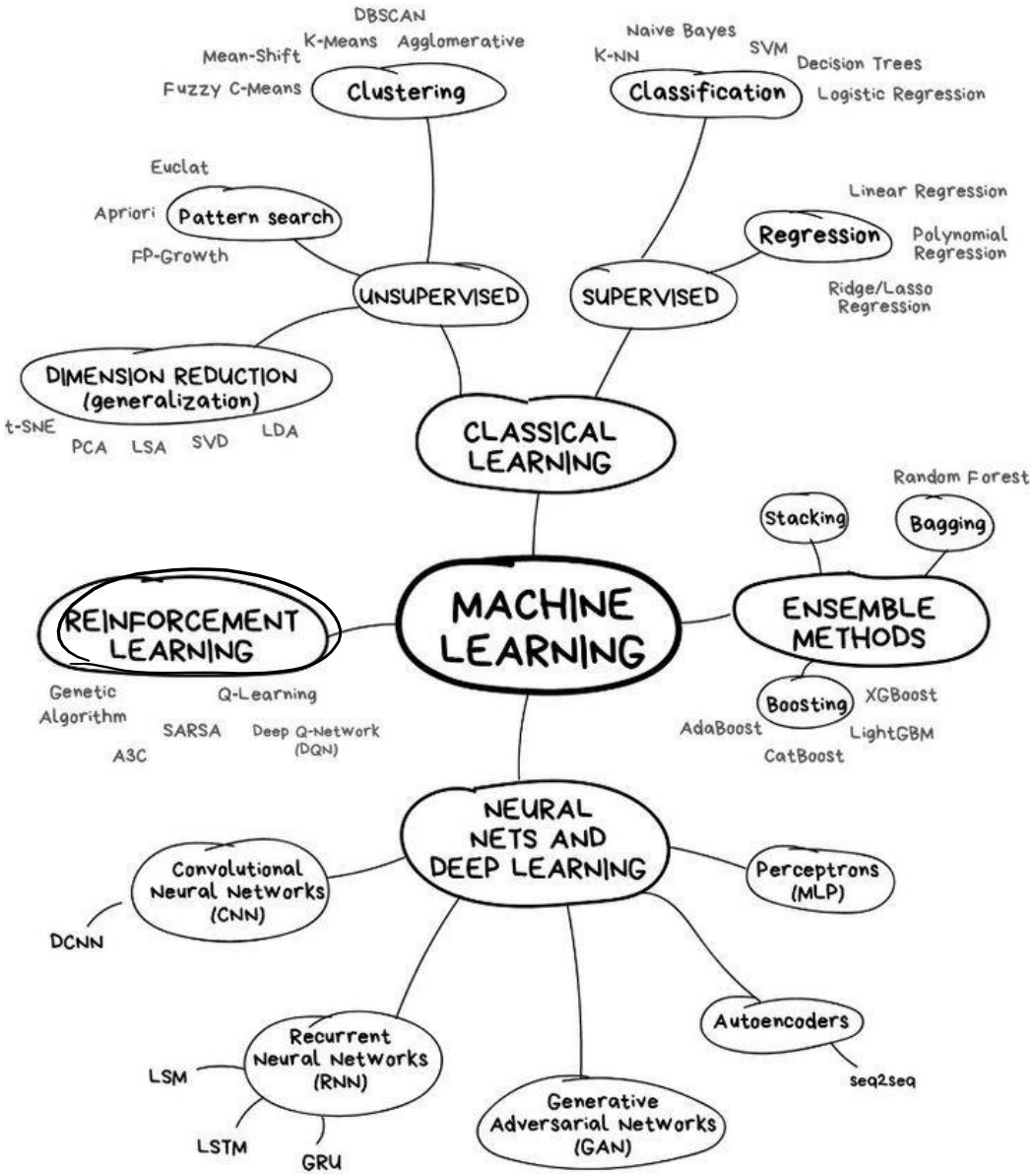


the pattern $X_1, X_2 \ldots$ are observed "sequentially".

After each X_i is observed, the learner receives a reward from the environment. The goal of the learner is to determine a "policy" (for selecting actions based on observations) to maximize "long-term" rewards.

↳ time notation

RL is important in Robotics (Navigatin, path planning) economics, and other areas.

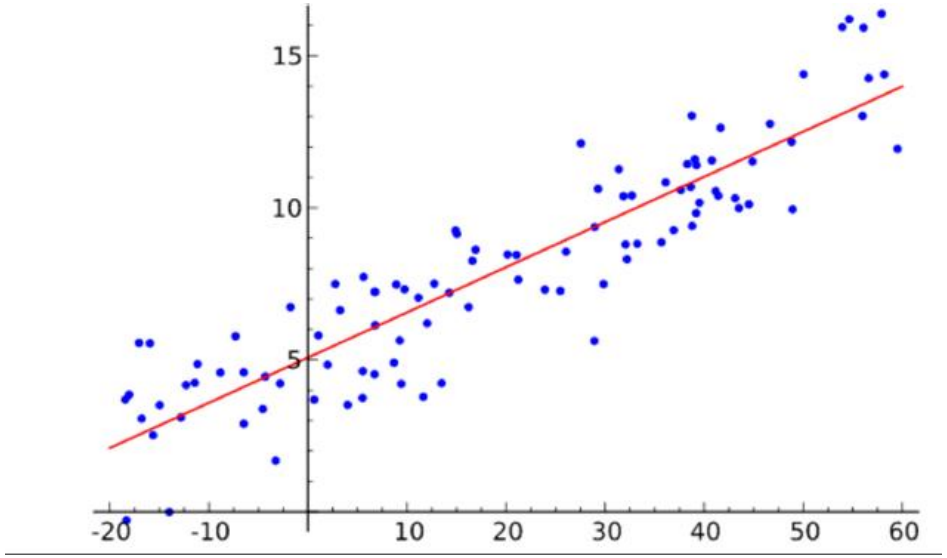

MACHINE LEARNING
CLASSICAL LEARNING
UNSUPERVISED
Clustering
DBSCAN
K-Means
Agglomerative
Mean-Shift
Fuzzy C-Means
Pattern search
Euclat
Apriori
FP-Growth
DIMENSION REDUCTION (generalization)
t-SNE
PCA
LSA
SVD
LDA
SUPERVISED
Classification
Naive Bayes
K-NN
SVM
Decision Trees
Logistic Regression
Regression
Linear Regression
Polynomial Regression
Ridge/Lasso Regression
REINFORCEMENT LEARNING
Genetic Algorithm
A3C
SARSA
Q-Learning
Deep Q-Network (DQN)
ENSEMBLE METHODS
Stacking
Bagging
Random Forest
Boosting
AdaBoost
XGBoost
LightGBM
CatBoost
NEURAL NETS AND DEEP LEARNING
Convolutional Neural Networks (CNN)
DCNN
Perceptrons (MLP)
Recurrent Neural Networks (RNN)
LSM
LSTM
GRU
Generative Adversarial Networks (GAN)
Autoencoders
seq2seq

Linear Regression

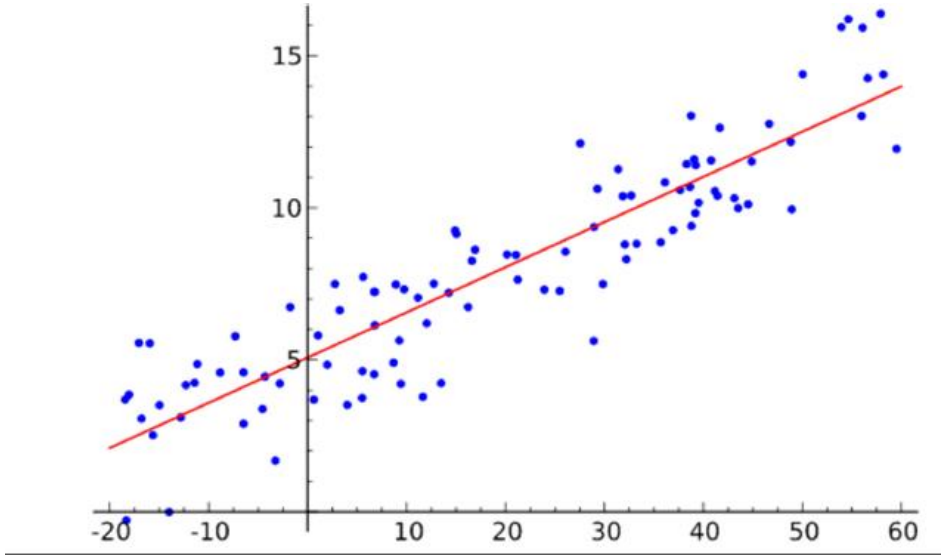
Linear Regression

- Numeric prediction (called "regression")
  - Classical statistical method (from 1805!)
  - Input: numerical data
  - Output: numerical data
  - Example) Sales ->Advertising

Year	Sales (Million Euro)	Advertising (Million Euro)
1	651	23
2	762	26
3	856	30
4	1,063	34
5	1,190	43
6	1,298	48
7	1,421	52
8	1,440	57
9	1,518	58



Linear Regression



- Given
  - Training data $D = \{(x_1, y_1), \dots (x_n, y_n)\}$
- Our goal
  - Find w, b that minimizes the mean squared error $E_{w,b}$
    - $E_{w,b} = \frac{1}{n}\sum$

$$minimize \ \frac{1}{n}$$

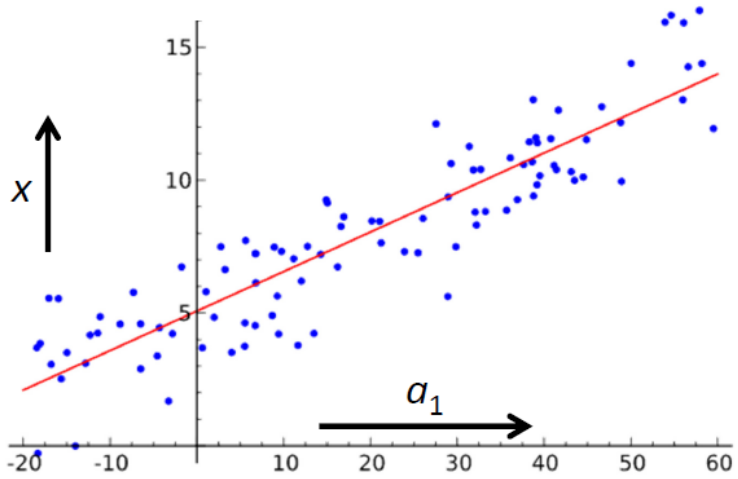
Linear Regression

$$x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$

- Calculate weights from training data

- Predicted value for first training instance $a^{(1)}$

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}$$



Linear Regression

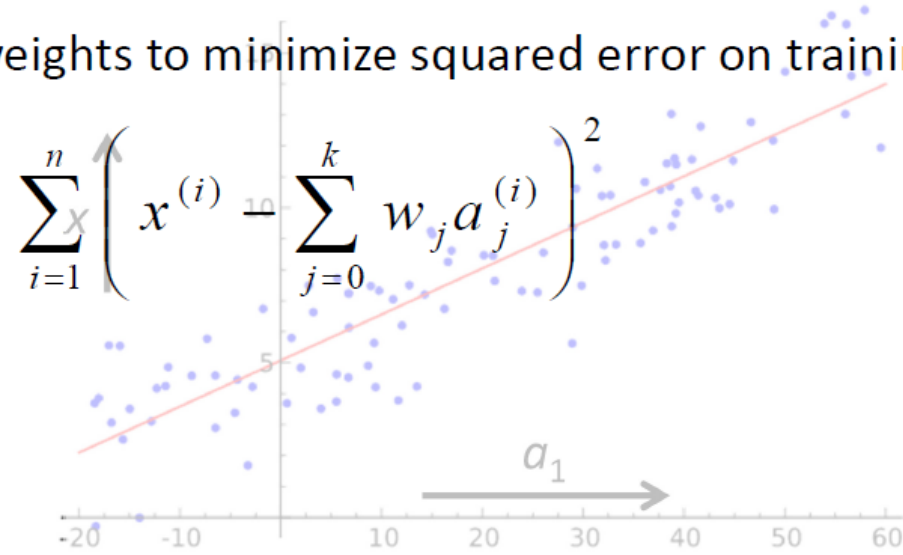
$$x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$

- Calculate weights from training data
- Predicted value for first training instance $a^{(1)}$

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1}$$

- Choose weights to minimize squared error on training data

$$\sum_{i=1}^{n} \left(x^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)} \right)^2$$



Multivariate Linear Regression

Gradient Descent: Scalar

- Algorithm to minimize a cost function $J(w)$
- η: Learning rate

Repeatedly update

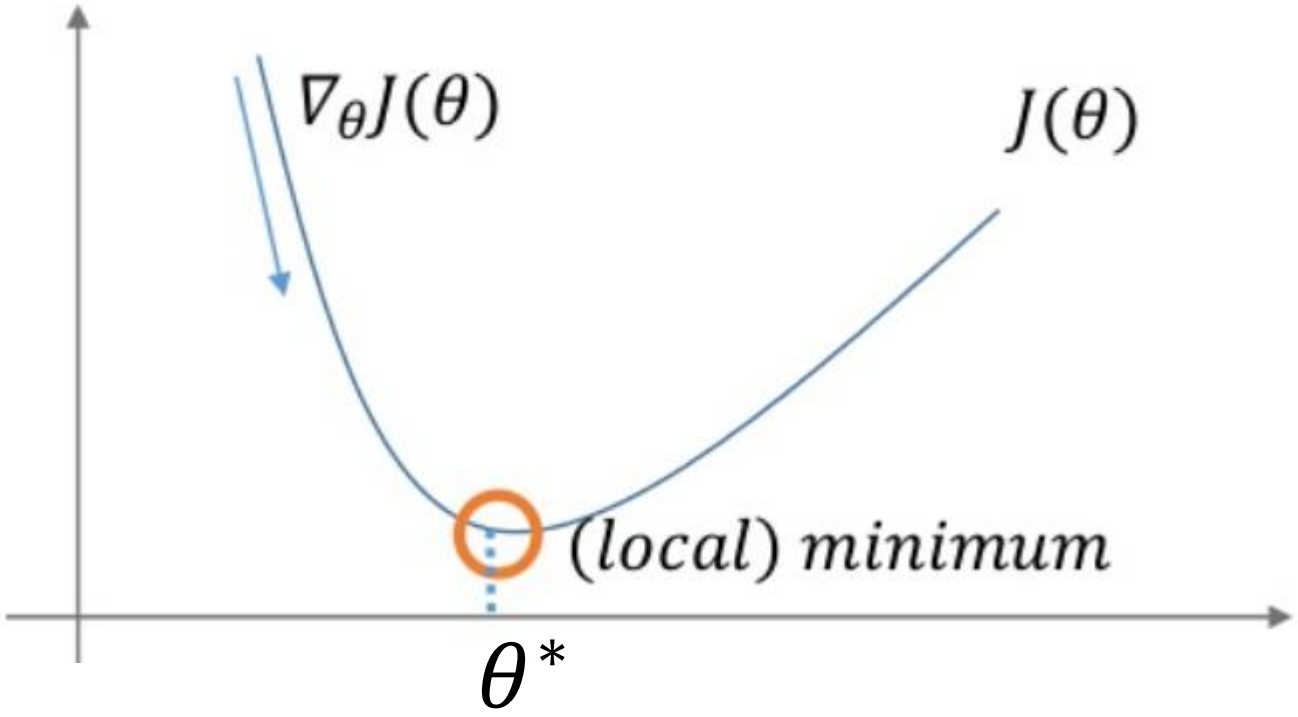
$$w = w - \eta \cdot \frac{dJ(w)}{dw}$$

Gradient Descent: Vector

- Algorithm to minimize a cost function $J(\theta)$
- $J(\theta)$: cost function
- θ: model parameters
- η: Learning rate

Repeatedly update

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$



By Hakky St

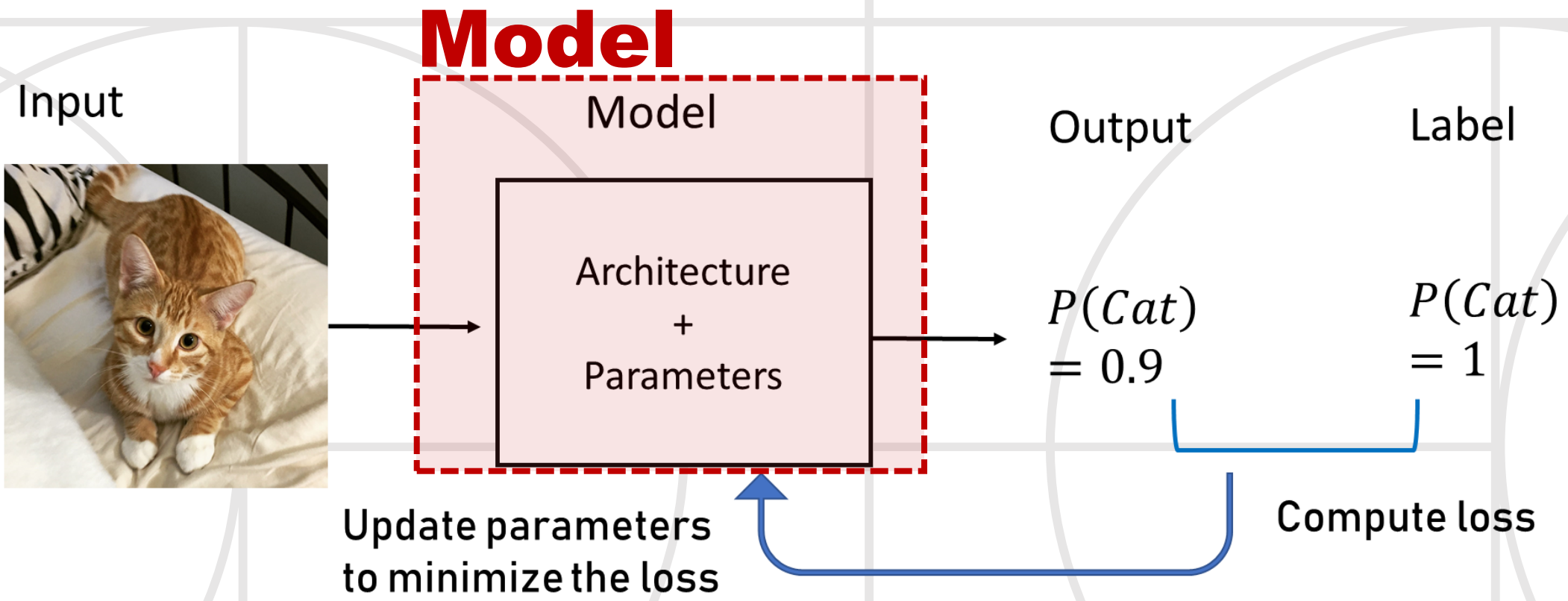
Logistic regression

Binary Classification

Model
=
Architecture
+
Parameters

1(cat) vs 0 (non cat)

$$x = \begin{bmatrix} 123 \\ 94 \\ \dots \\ 202 \\ 123 \\ 94 \\ \dots \\ 142 \end{bmatrix}$$



Logistic Regression

Logistic Regression

- A simple model for binary classification
- Maybe one of the simplest neural network
- A training example $(\boldsymbol{x}, y)$
  - Input: $\boldsymbol{x} \in \mathbb{R}^n$
  - Output: $y \in \{0,1\}$
- m training examples
  - $\{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots (\boldsymbol{x}^{(m)}, y^{(m)})\}$

Logistic Regression

Model = Architecture + Parameters

- Given $\boldsymbol{x} \in \mathbb{R}^n$
- Want $\hat{y} = P(y = 1|\boldsymbol{x})$

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

$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b$$

Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$

Logistic Regression

Model = Architecture + Parameters

- Given $\boldsymbol{x} \in \mathbb{R}^n$
- Want $\hat{y} = P(y = 1|\boldsymbol{x})$

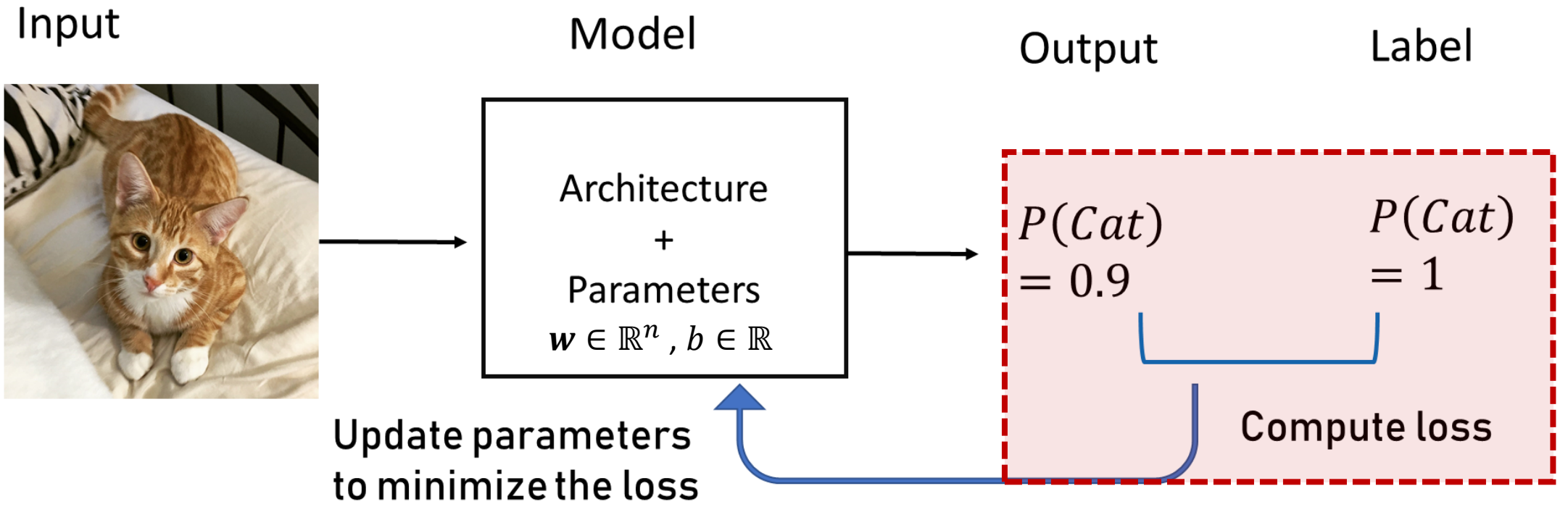
$$\hat{y} = \sigma\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right)$$

Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$

$$\Theta = \{\boldsymbol{w}, b\}$$

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

$$\sigma(-\infty) = 0$$
$$\sigma(+\infty) = 1$$



Logistic Regression

Cost function & Loss

Logistic Regression: Cost function

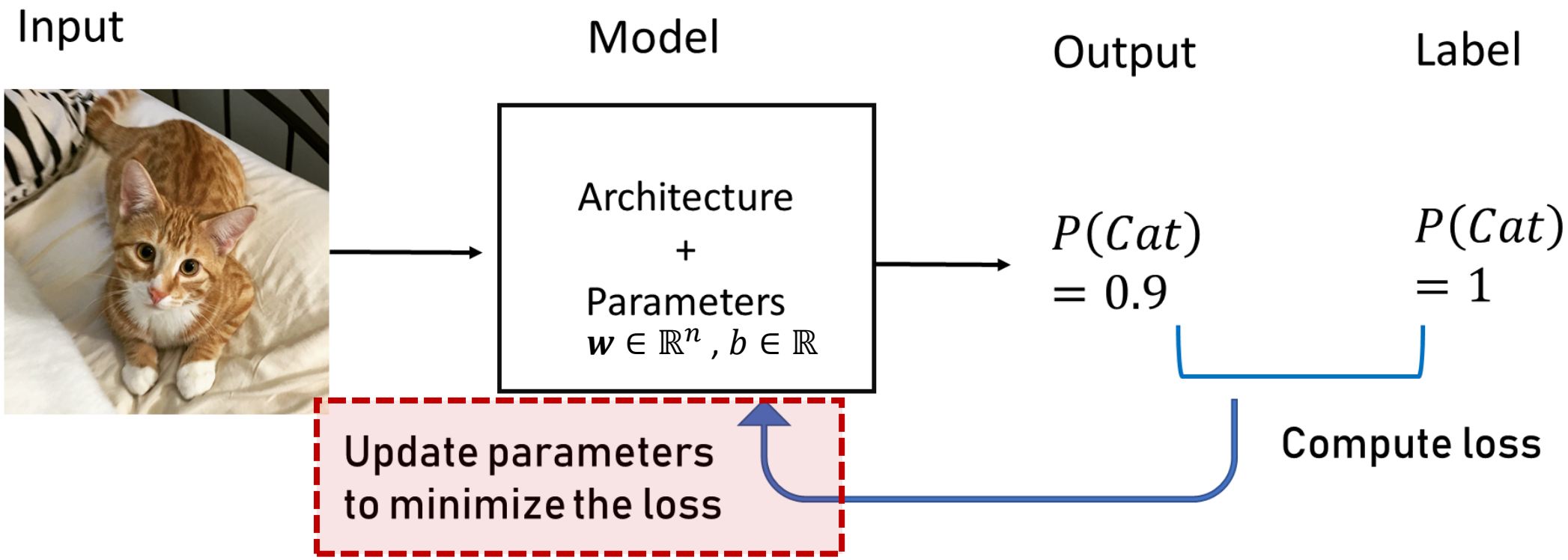
- $\hat{y} = \sigma\left(\boldsymbol{w}^{\top}\boldsymbol{x} + b\right)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$
- Given $\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \ldots \left(\boldsymbol{x}^{(m)}, y^{(m)}\right)\right\}$, want $\hat{y}^{(i)} \approx y^{(i)}$
- Loss function: Binary Cross Entropy

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

If $y = 1$: $L(\hat{y}, y) = -\log \hat{y}$
If $y = 0$: $L(\hat{y}, y) = -\log(1 - \hat{y})$

- Cost function:

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



Optimization

Logistic Regression

Optimization

- Logistic regression model
  - $\hat{y} = \sigma\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$
- Cost function

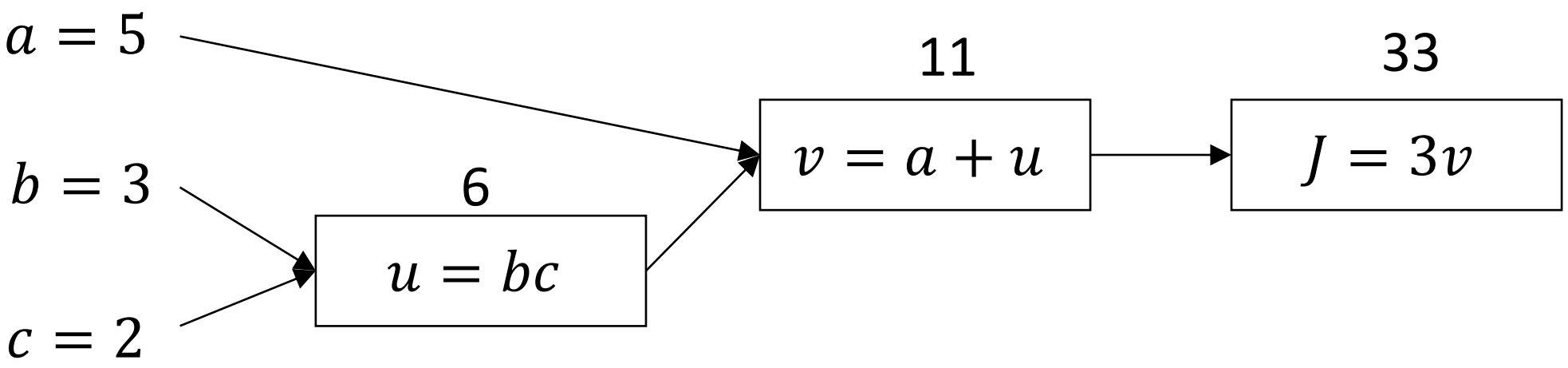
$$J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right]$$

- Our goal
  - Find parameters $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$ that minimize $J(\boldsymbol{w}, b)$
- Gradient Descent!

Derivative with a Computation Graph

Computation Graph

- $J(a, b, c) = 3(a + bc)$

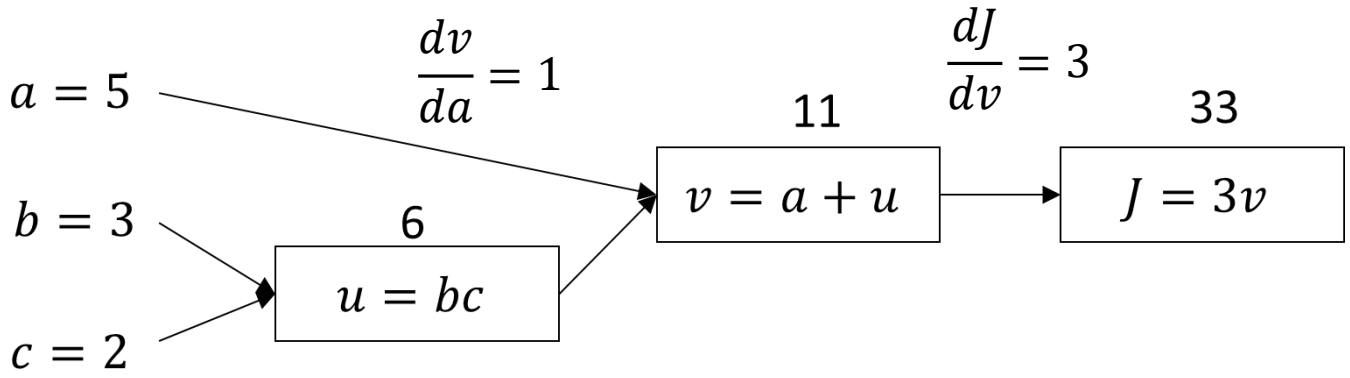


Chain Rule

$$(f \circ g)'(c) = f'(g(c)) \cdot g'(c)$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Derivatives with a Computation Graph

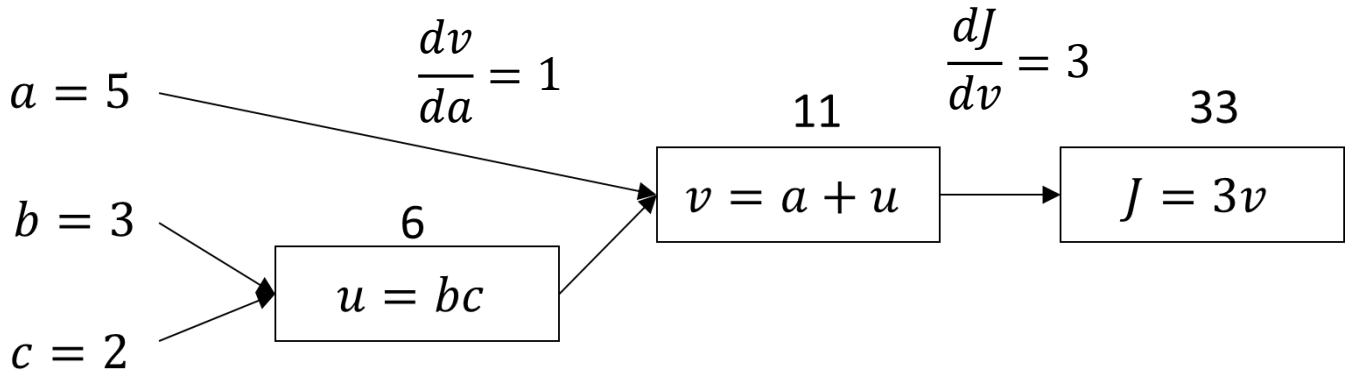


$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Chain rule

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

Derivatives with a Computation Graph



$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Chain rule

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

$$\frac{dJ}{db} = \frac{dJ}{du}\frac{du}{db} = 2\frac{dJ}{du} = 2\ \frac{dv}{du}\frac{dJ}{dv} = 6$$

$$\frac{dJ}{dc} = \frac{dJ}{du}\frac{du}{dc} = 3\frac{dJ}{du} = 3\ \frac{dv}{du}\frac{dJ}{dv} = 9$$

Gradient Descent :Logistic Regression

Background: Derivatives

1. $\frac{d}{dx}\log_{\mathrm{e}} x = \frac{1}{x}$

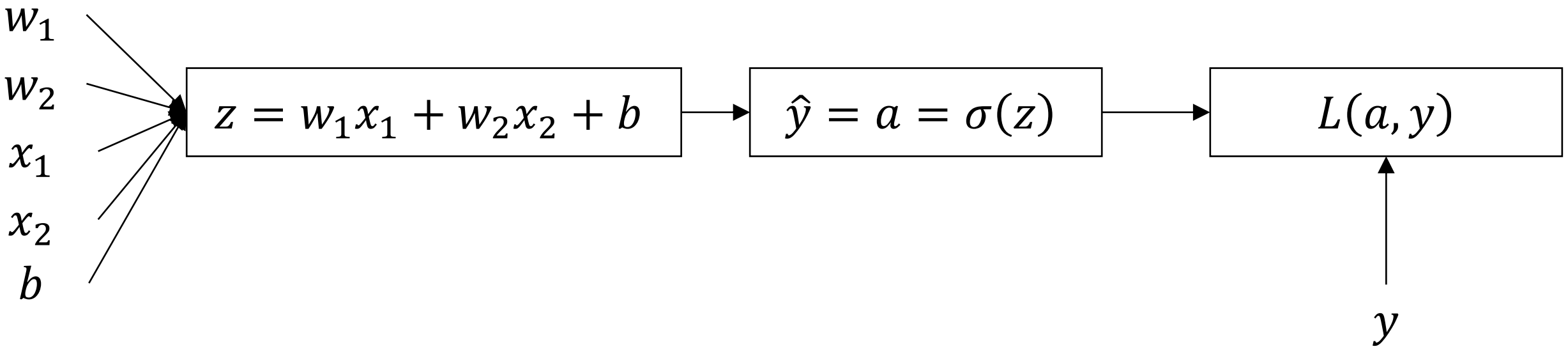
2. $\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$

Logistic Regression Recap

- $\hat{y} = \sigma\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$
- $L(\hat{y}, y) = -y \log \hat{y} - (1-y) \log(1-\hat{y})$

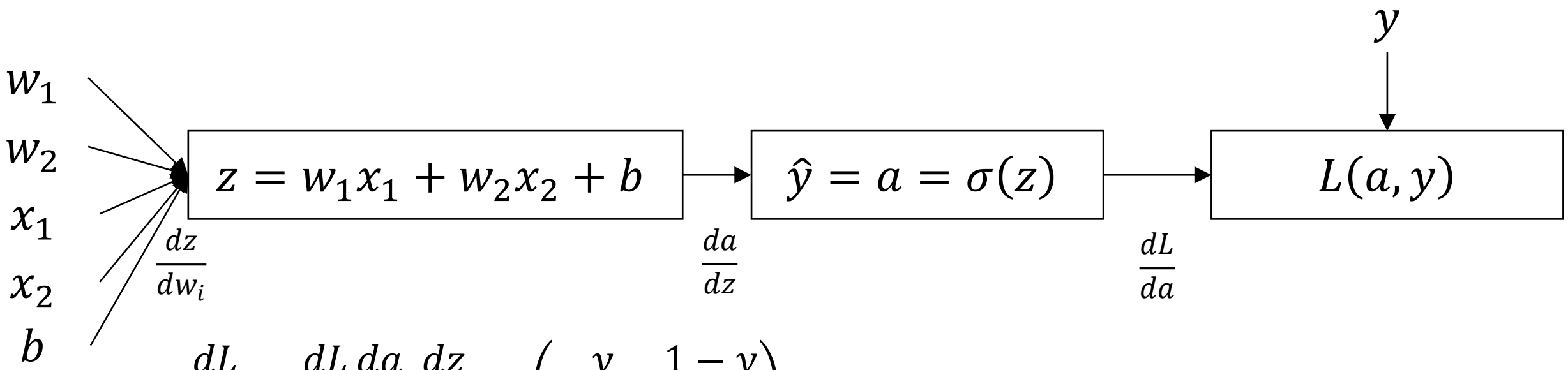
For the simplicity

$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$



$$L(a, y) = -y \log a - (1 - y) \log(1 - a)$$

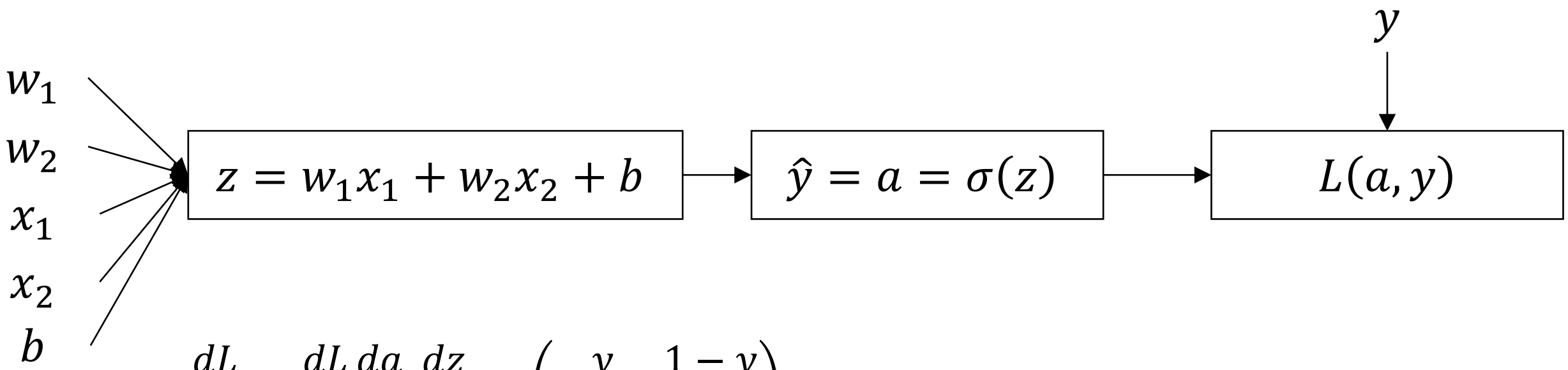
Logistic Regression Derivative



$$\frac{dL}{dw_i} = \frac{dL}{da}\frac{da}{dz}\frac{dz}{dw_i} = \left(-\frac{y}{a} + \frac{1 - y}{1 - a}\right)\sigma(z)(1 - \sigma(z))x_i$$

$$L(a, y) = -y \log a - (1 - y) \log(1 - a)$$

Logistic Regression Derivative



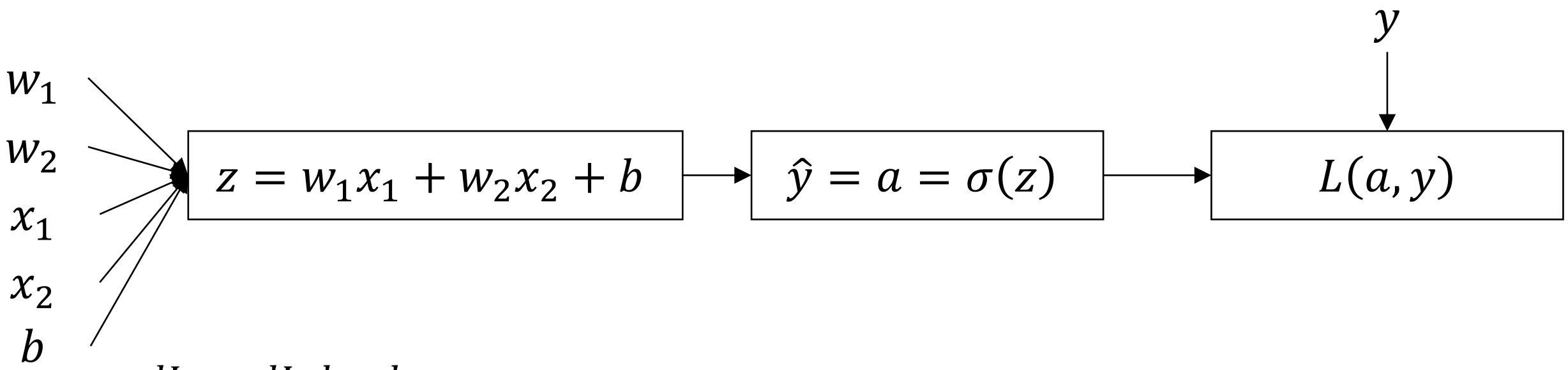
$$\frac{dL}{dw_i} = \frac{dL}{da}\frac{da}{dz}\frac{dz}{dw_i} = \left(-\frac{y}{a} + \frac{1-y}{1-a}\right)\sigma(z)(1 - \sigma(z))x_i$$

$$= \left(-\frac{y}{a} + \frac{1-y}{1-a}\right)a(1-a)x_i = -y(1-a)x_i + (1-y)ax_i$$

$$= (a - y)x_i$$

$$L(a, y) = -y \log a - (1 - y) \log(1 - a)$$

Logistic Regression Derivative



$$\frac{dL}{dw_1} = \frac{dL}{da}\frac{da}{dz}\frac{dz}{dw_1} = (a - y)x_1$$

$$\frac{dL}{dw_2} = \frac{dL}{da}\frac{da}{dz}\frac{dz}{dw_2} = (a - y)x_2$$

$$\frac{dL}{db} = \frac{dL}{da}\frac{da}{dz}\frac{dz}{db} = a - y$$

$$w_1 := w_1 - \eta \frac{dL}{dw_1}$$

$$w_2 := w_2 - \eta \frac{dL}{dw_2}$$

$$b := b - \eta \frac{dL}{db}$$

Gradient descent on m examples

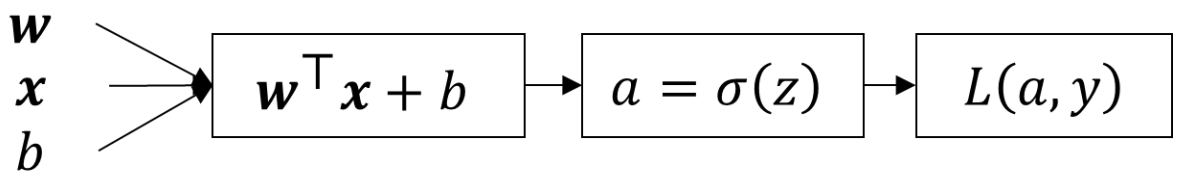
$$J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right)$$

$$\frac{d}{dw_i}J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \frac{d}{dw_k} L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m} \left(\hat{y}^{(i)} - y\right)x_k$$

$$\frac{d}{db}J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \frac{d}{db} L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m} \left(\hat{y}^{(i)} - y\right)$$

Logistic Regression on m example

- Randomly Initialize $\boldsymbol{w}, b$
- lr = 0.1
- For e = 1 to n_{epoch}:
  - J = 0; d_w1 = 0; d_w2 = 0; d_b=0
  - For i= 1 to m:
    - $z = w_1 x_1^{(i)} + w_2 x_2^{(i)} + b$
    - $a = \sigma(z)$
    - d_w1 += $(a - y)x_1^{(i)}$
    - d_w2 += $(a - y)x_2^{(i)}$
    - d_b += $a - y$
  - w_1 −= lr ∗ d_w1/m
  - w_2 −= lr ∗ d_w2/m
  - b −= lr ∗ d_b/m



$$\frac{dL}{dw_k} = (a - y)x_k$$

$$\frac{dL}{db} = a - y$$

Logistic Regression

Programming in Python

Logistic regression: logical AND

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

$$y = x_1 \; AND \; x_2$$

Data preparation

Logistic regression (AND)

In [ ]:
```
import random
from math import exp,log
```

Data prepration

In [12]:
```
X = [(0,0),(1,0),(0,1),(1,1)]
Y = [0,0,0,1]
```

Model

Model

In [14]:
```
class logistic_regression_model():
    def __init__(self):
        self.w = [random.random(), random.random()]
        self.b = random.random()

    def sigmoid(self,z):
        return 1/(1 + exp(-z))

    def predict(self,x):
        z = self.w[0] * x[0] + self.w[1] * x[1] + self.b
        a = self.sigmoid(z)
        return a
```

In [15]:
```
model = logistic_regression_model()
```

Training

Training

In [16]:
```
def train(X, Y, model, lr = 0.1):
    dw0 = 0.0
    dw1 = 0.0
    db = 0.0
    m = len(X)
    cost = 0.0
    for x,y in zip(X,Y):
        a = model.predict(x)
        if y == 1:
            cost -= log(a)
        else:
            cost -= log(1-a)

        dw0 += (a-y)*x[0]
        dw1 += (a-y)*x[1]
        db += (a-y)

    cost /= m
    model.w[0] -= lr * dw0/m
    model.w[1] -= lr * dw1/m
    model.b -= lr*db/m

    return cost
```

In [17]:
```
for epoch in range(10000):
    cost = train(X,Y, model, 0.1)
    if epoch %100==0:
        print(epoch, cost)
```

```
0 0.9799277803394626
100 0.4455359447221918
200 0.35278521282410236
300 0.29469845366603453
400 0.25432071172280113
500 0.22425431605184998
600 0.20079558352997323

...

9000 0.019352012397599427
9100 0.019139595049452222
9200 0.018931735521070026
9300 0.018728289777080104
9400 0.018529119755095403
9500 0.01833409306055272
9600 0.018143082680009734
9700 0.017955966711161366
9800 0.017772628111558907
9900 0.017592954455438015
```

Testing

Testing

```
In [22]: model.predict((0,0))
Out[22]: 1.2451625968657186e-05

In [23]: model.predict((0,1))
Out[23]: 0.020240526677753723

In [24]: model.predict((1,0))
Out[24]: 0.020240519389194 4

In [25]: model.predict((1,1))
Out[25]: 0.9716510306648906
```

Logistic Regression

Programming Assignment

Logistic regression: boolean operators

- Training logistic regression models for Boolean operators
- Requirements
  - AND, OR, XOR
    - You need to build a dataset for each operator
    - may not working for an operator
  - Use numpy arrays
    - Initialization with lists: x, y
    - Random initialization: w, b
  - Use numpy operator
    - Inner product
    - Addition

Logistic regression: boolean operators

- Due: 2021/9/21 PM 11:59
- Submission to HY-ON
- Submit following files
  - A short report (pdf or docx)
    - Source code for model & training
    - For each operator
      - loss plot
        - varying learning rate (at least 3 learning rate)
      - Predicted results
  - Source code

NumPy usages

```
In [26]: import numpy as np
```

Import numpy

```
In [31]: v1 = np.array([1.0,2.0])
         v2 = np.array([1.0,1.0])
```

Initialization with lists

```
In [32]: np.inner(v1,v2)
Out[32]: 3.0
```

Inner product

```
In [34]: v1+v2
Out[34]: array([2., 3.])
```

Addition

```
In [35]: 3*v1
Out[35]: array([3., 6.])
```

Scalar multiplication

```
In [33]: np.random.normal(size = 2)
Out[33]: array([ 0.25214782, -0.96463649])
```

Radom initialization with $N(0,1)$