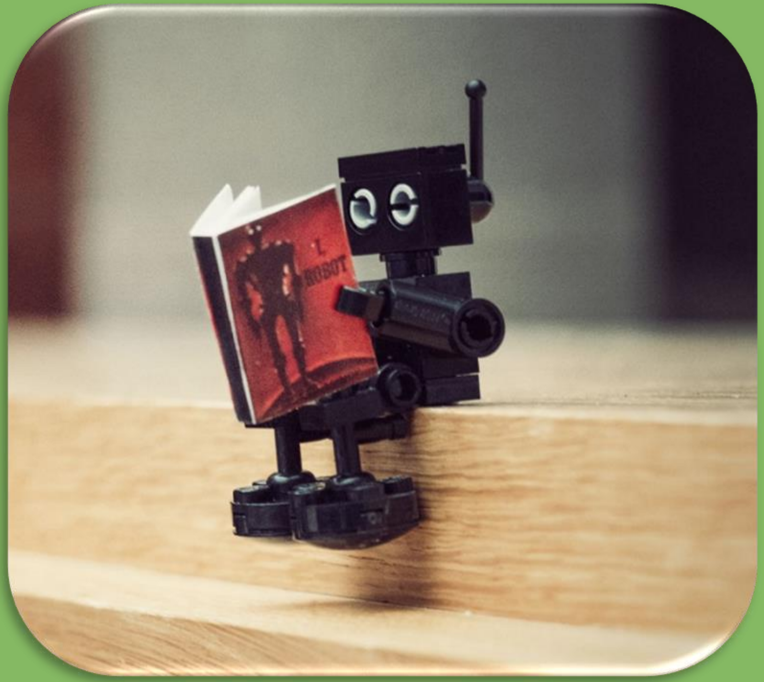
دوره‌ی آموزشی «علم داده»

Data Science Course

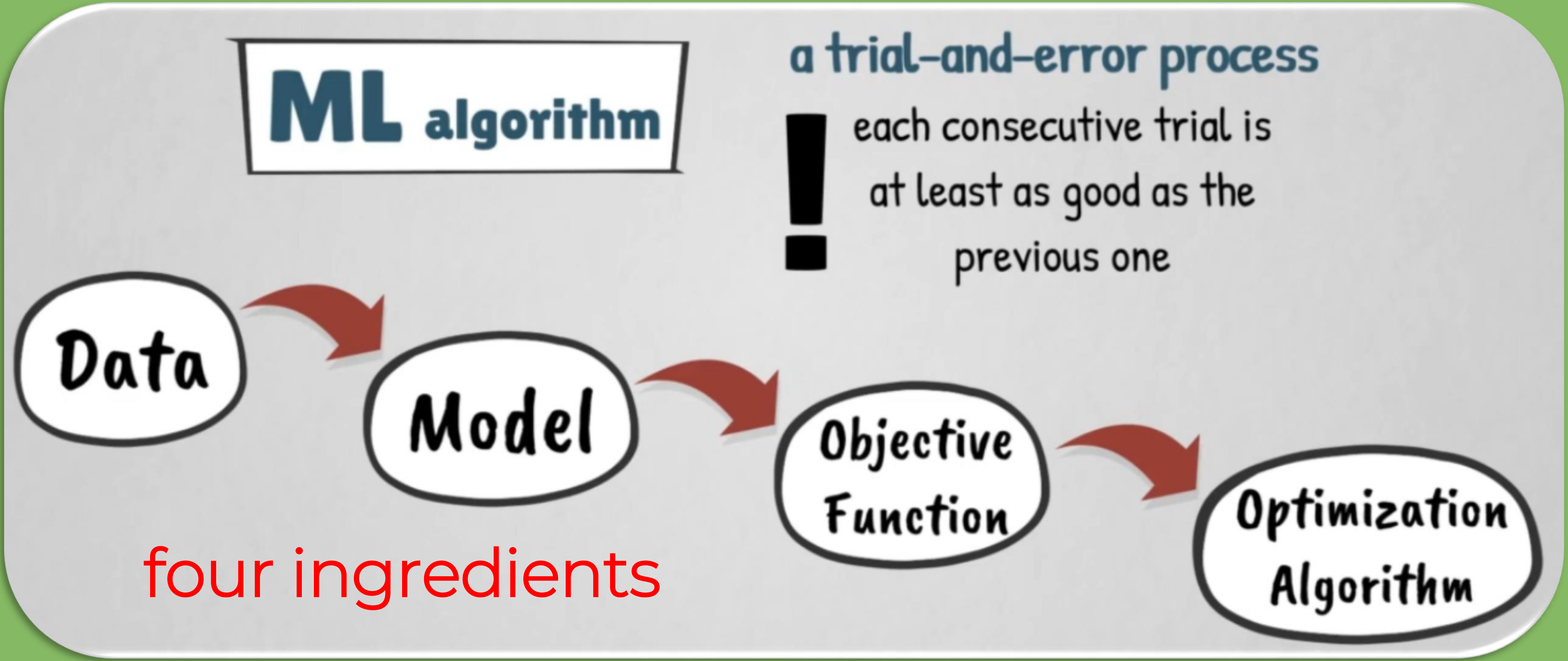
جلسه بیست و ششم (بخش اول)

اجزای یادگیری ماشین



مدرس: محمد فزونی

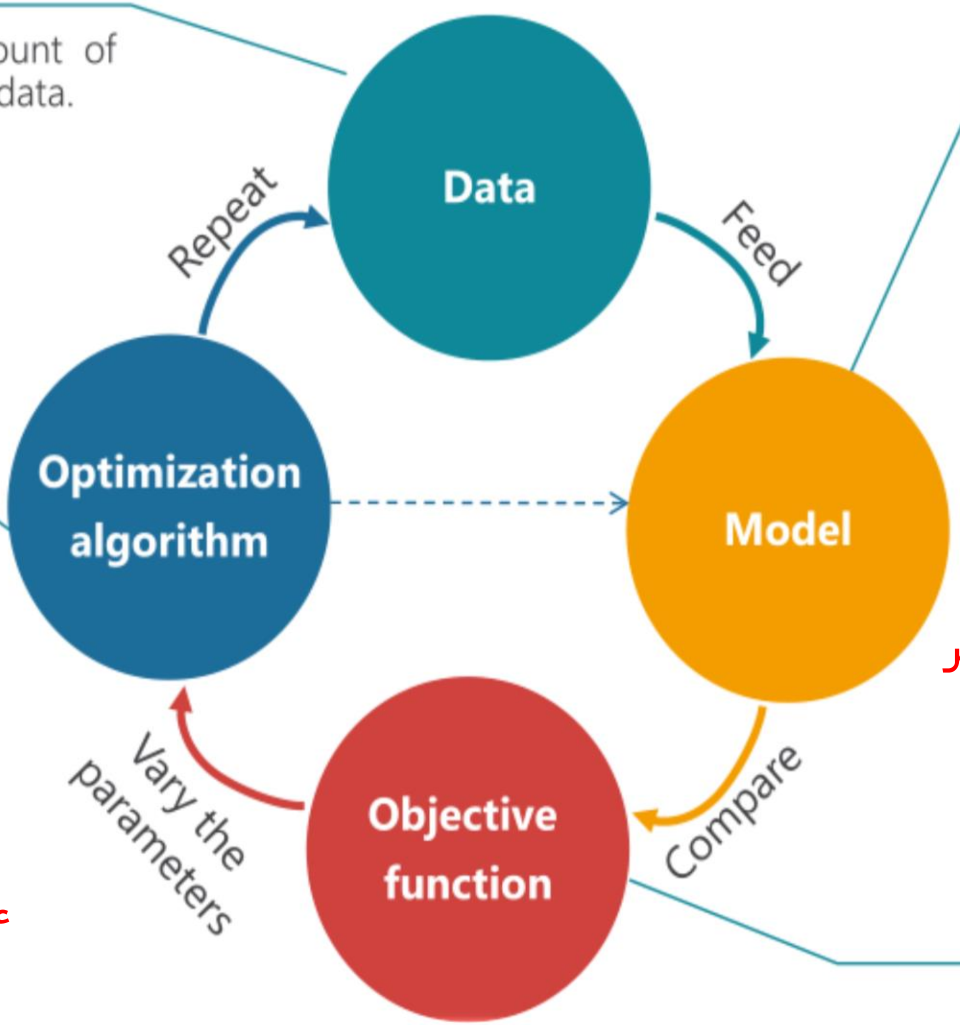
عضو هیئت علمی دانشگاه گنبدکاووس



four ingredients

First, we need to prepare a certain amount of **data** to train on. Usually, we take historical data.

۱ – اول کمی داده نیاز داریم



We choose the type of **model**. Roughly speaking, this is some function, which is defined by the *weights* and the *biases*. We feed the input data into the model. Essentially, the idea of the machine learning algorithm is to find the *parameters* for which the model has the highest predictive power.

۲– مدل رو انتخاب کن

۳– قدرت پیش‌بینی مدل رو اندازه بگیر

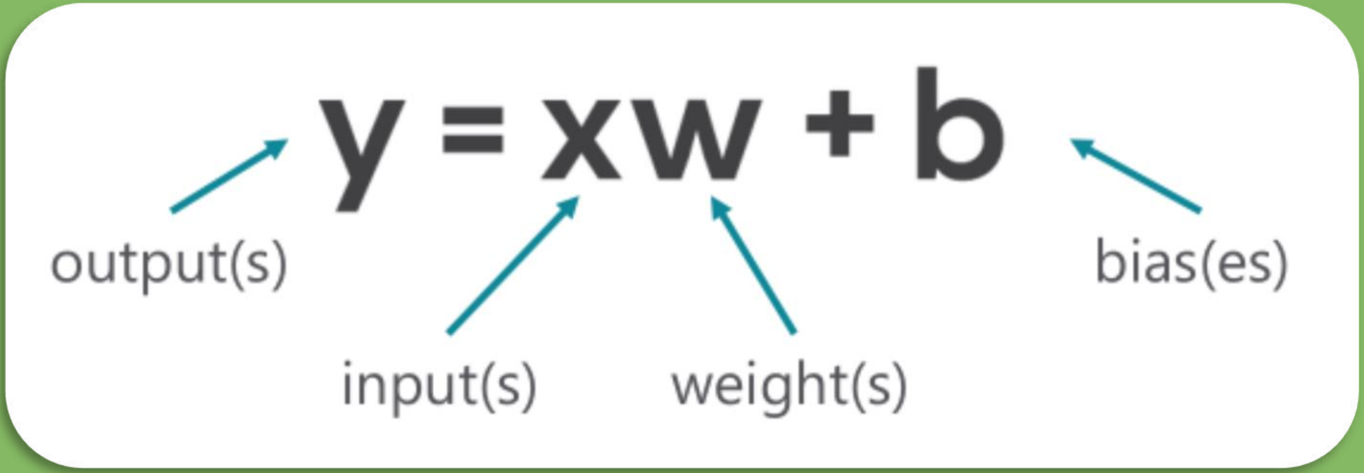
The **objective function** measures the predictive power of our model. Mathematically, the machine learning problem boils down to *optimizing* this function. For example, in the case of loss, we are trying to *minimize* it.

We achieve the optimization using an **optimization algorithm.** Using the value of the objective function, the optimization algorithm *varies the parameters* of the model. This operation is repeated until we find the values of the parameters, for which the objective function is optimal.

۴– پارامترها رو تغییر بده تا به بهترین تابع هدف برسیم

مدل (Model)
در یادگیری ماشین

سادهترین مدل ممکن، یک مدل خطی است.
برخلاف ظاهر بسیار ساده، در یادگیری عمیق، پایهی مدلهای بسیار پیچیده است

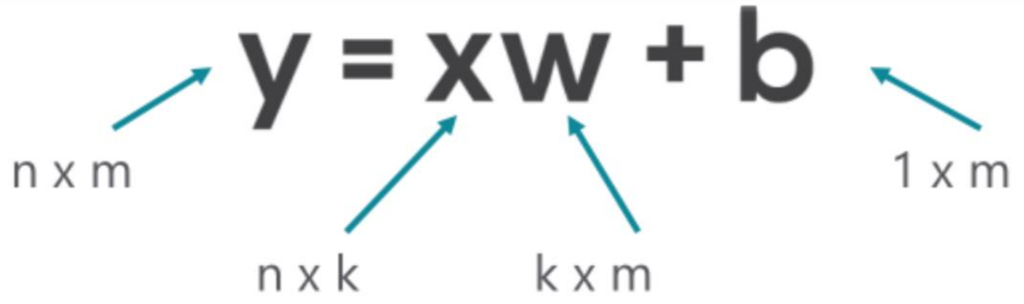


The weight(s) together with the bias(es) are called Parameters.
In statistics, w is called coefficient and b is called intercept.

هر مدلی توسط پارامترهایش مشخص میگردد. در نتیجه با یک فرایند آزمون و خطا به تغییر آنها میپردازیم تا به بهترین پارامترهای ممکن برسیم.

مدل ما می‌تونه چند بعدی و یا چندگانه باشه.

- **ورودی‌های چندگانه**
- **خروجی‌های چندگانه**



Where:

- n is the number of samples (observations)
- m is the number of output variables
- k is the number of input variables

The simplest linear model, where m = k = 1, n > 1

$$\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 w + b \\ x_2 w + b \\ \cdots \\ x_n w + b \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \begin{bmatrix} w \end{bmatrix} + \begin{bmatrix} b \end{bmatrix}$$

We can extend the model to multiple inputs where n, k > 1, m = 1.

$$\underset{n \times 1}{\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ \dots \\ y_n \end{bmatrix}} = \begin{bmatrix} x_{11}w_1 + x_{12}w_2 + \dots + x_{1k}w_k + b \\ x_{21}w_1 + x_{22}w_2 + \dots + x_{2k}w_k + b \\ \dots \\ \dots \\ x_{n1}w_1 + x_{n2}w_2 + \dots + x_{nk}w_k + b \end{bmatrix} = \underset{n \times k}{\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix}} \underset{k \times 1}{\begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_k \end{bmatrix}} + \underset{1 \times 1}{\begin{bmatrix} b \end{bmatrix}}$$

We can extend the model to multiple inputs where n, k, m > 1.

$$\underbrace{\begin{bmatrix} y_{11} & \cdots & y_{1m} \\ y_{21} & \cdots & y_{2m} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ y_{n1} & \cdots & y_{nm} \end{bmatrix}}_{n \times m} = \underbrace{\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}}_{n \times k} \underbrace{\begin{bmatrix} w_{11} & \cdots & w_{1m} \\ w_{21} & \cdots & w_{2m} \\ \cdots & \cdots & \cdots \\ w_{k1} & \cdots & w_{km} \end{bmatrix}}_{k \times m} + \underbrace{\begin{bmatrix} b_1 & \cdots & b_m \end{bmatrix}}_{1 \times m}$$

$$\underbrace{\begin{bmatrix} y_{11} & \cdots & y_{1m} \\ y_{21} & \cdots & y_{2m} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ y_{n1} & \cdots & y_{nm} \end{bmatrix}}_{n \times m} = \begin{bmatrix} x_{11}w_{11} + x_{12}w_{21} + \ldots + x_{1k}w_{k1} + b_1 & \cdots & x_{11}w_{1m} + x_{12}w_{2m} + \ldots + x_{1k}w_{km} + b_m \\ x_{21}w_{11} + x_{22}w_{21} + \ldots + x_{2k}w_{k1} + b_1 & \cdots & x_{21}w_{1m} + x_{22}w_{2m} + \ldots + x_{2k}w_{km} + b_m \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ x_{n1}w_{11} + x_{n2}w_{21} + \ldots + x_{nk}w_{k1} + b_1 & \cdots & x_{n1}w_{1m} + x_{n2}w_{2m} + \ldots + x_{nk}w_{km} + b_m \end{bmatrix} = \underbrace{\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}}_{n \times k} \underbrace{\begin{bmatrix} w_{11} & \cdots & w_{1m} \\ w_{21} & \cdots & w_{2m} \\ \cdots & \cdots & \cdots \\ w_{k1} & \cdots & w_{km} \end{bmatrix}}_{k \times m} + \underbrace{\begin{bmatrix} b_1 & \cdots & b_m \end{bmatrix}}_{1 \times m}$$

تابع هدف (Objective Function) در یادگیری ماشین

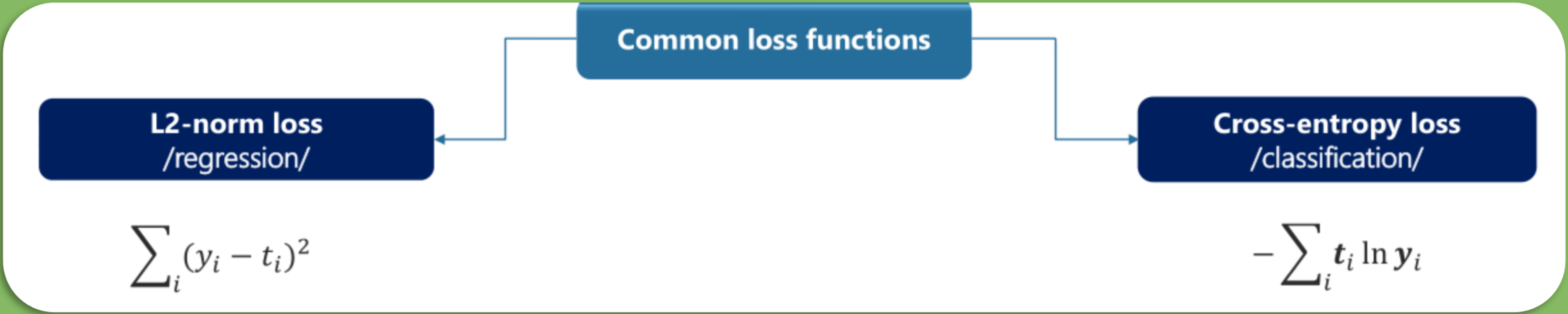
تابع هدف (Objective Function)، تابعیه که میزان خوب بودن مدل ما رو اندازه می‌گیره.

می‌تونیم تابع هدف رو به دو صورت زیر در نظر بگیریم:

- زیان (یادگیری نظارت شده – Supervised)
- سود (یادگیری تقویتی – Reinforcement)

در ادامه هدف ما، کار روی یادگیری نظارت شده هست.

دو تا تابع زیان مرسوم داریم به شکل زیر:



model: $\mathbf{x}_i w + b = y_i \longrightarrow t_i$

$L(y,t) \longrightarrow$ loss

$C(y,t) \longrightarrow$ cost

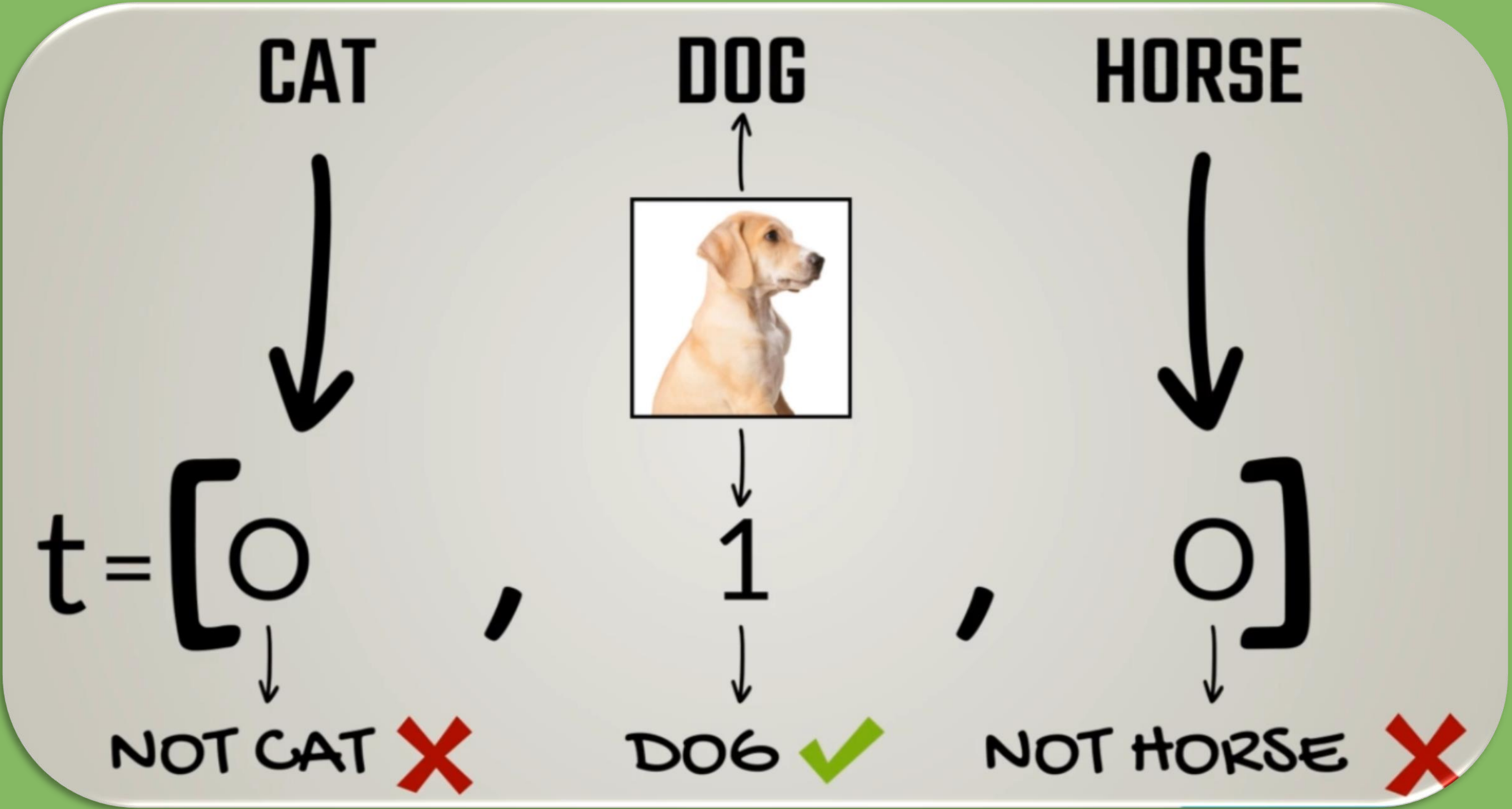
$E(y,t) \longrightarrow$ error

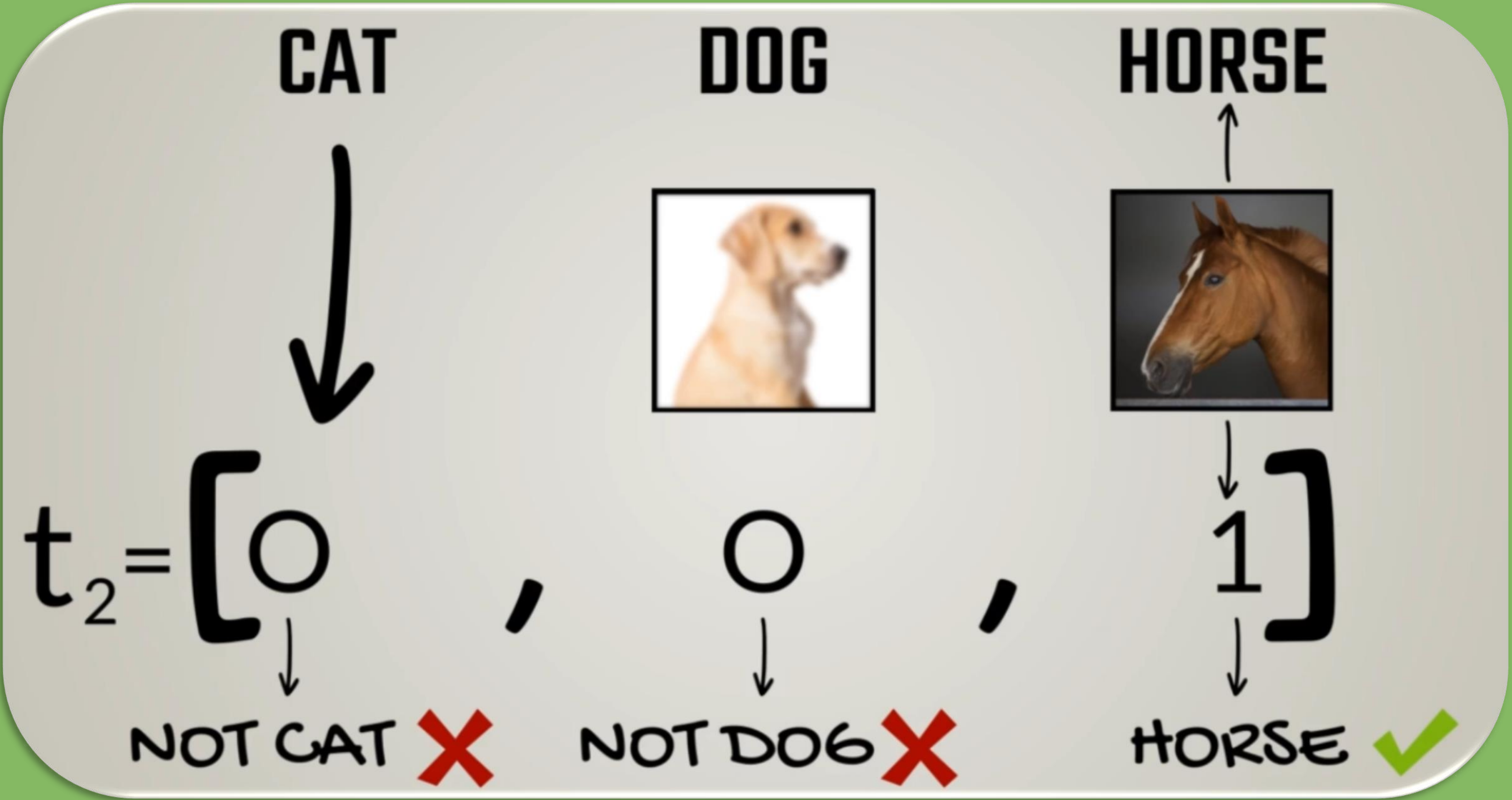
$$\text{L2-norm} = \sum_i (y_i - t_i)^2$$

THE LOWER THE ERROR

THE LOWER THE LOSS

*"norm" comes from the fact it is the vector norm, or Euclidean distance of the outputs and the targets


CAT
DOG
HORSE
t = [0 , 1 , 0]
NOT CAT
DOG
NOT HORSE


CAT
DOG
HORSE
$t_2=[0 , 0 , 1]$
NOT CAT
NOT DOG
HORSE

DOG

$y = [\,0.4, 0.4, 0.2\,]$

$t = [\,0, 1, 0\,]$

$$L(\mathbf{y}, \mathbf{t}) = -0 \times \ln 0.4 - 1 \times \ln 0.4 - 0 \times \ln 0.2$$
$$= 0.92$$

HORSE

$y = [\,0.1, 0.2, 0.7\,]$

$t = [\,0, 0, 1\,]$

$$L(\mathbf{y}, \mathbf{t}) = -0 \times \ln 0.1 - 0 \times \ln 0.2 - 1 \times \ln 0.7$$
$$= 0.36$$

The lower the loss, the more accurate the model

DOG

$y = [\,0.4, 0.4, 0.2\,]$

$t = [\,0, 1, 0\,]$

$L(\mathbf{y}, \mathbf{t}) = -0 \times \ln 0.4 \; -1 \times \ln 0.4 \; -0 \times \ln 0.2$

$= 0.92$

HORSE

$y = [\,0.1, 0.2, 0.7\,]$

$t = [\,0, 0, 1\,]$

$L(\mathbf{y}, \mathbf{t}) = -0 \times \ln 0.1 \; -0 \times \ln 0.2 \; -1 \times \ln 0.7$

$= 0.36$

The lower the loss, the more accurate the model

DOG

$y = [\,0.4, 0.4, 0.2\,]$

$t = [\,0, 1, 0\,]$

$$L(\mathbf{y}, \mathbf{t}) = -0 \times \ln 0.4 \; -1 \times \ln 0.4 \; -0 \times \ln 0.2 = 0.92$$

HORSE

$y = [\,0.1, 0.2, 0.7\,]$

$t = [\,0, 0, 1\,]$

$$L(\mathbf{y}, \mathbf{t}) = 0.36$$

better prediction

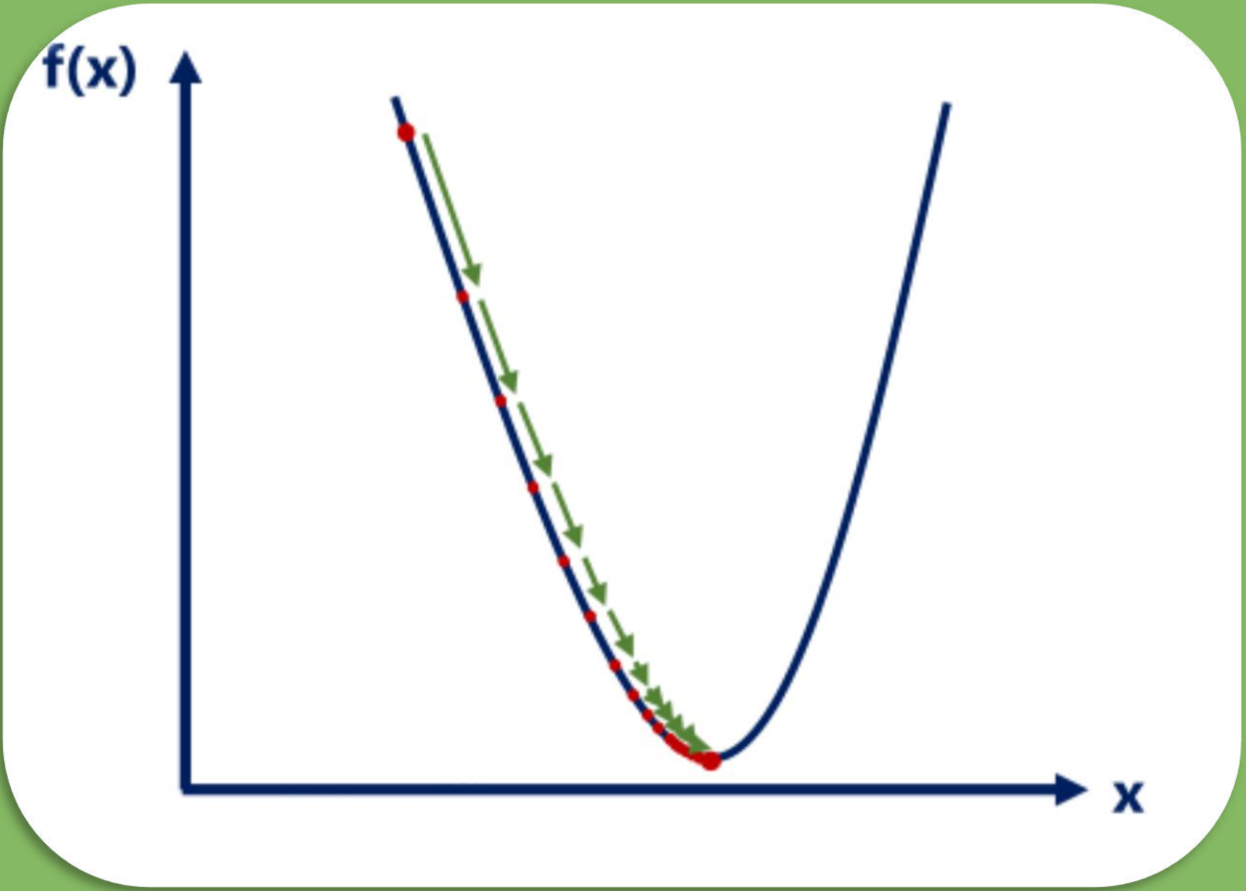
الگوریتم بهینه سازی
(Optimization algorithm)
در یادگیری ماشین

گرادیان کاهشی (Gradient descent) مرسوم‌ترین الگوریتمِ بهینه‌سازی است. نکته‌ی اصلی اینه که ما می‌تونیم مینیمم یک تابع رو باَ استفاده از رابطه‌ی زیر بدست بیاریم:

$$x_{i+1} = x_i - \eta f'(x_i)$$

عدد η یک مقدار مثبت و کوچیکه. در ادبیات یادگیری ماشین به این عدد نرخ یادگیری (Learning rate) می‌گن.

حالا چطوری فرمول بالا کمک میکنه تا مینیمم رو بدست بیاریم؟



اگر $f'(x_i) < 0$، پس ما در سمت چپ
سهمی هستیم. طبق فرمول

$$x_{i+1} = x_i - \eta f'(x_i)$$

نتیجه می‌گیریم که:

$$x_i < x_{i+1}$$

در نتیجه حرکت بسمت راست، یعنی
مینیمم هست.

اگر $f'(x_i) > 0$، پس ما در سمت راست
سهمی هستیم. در نتیجه طبق فرمول
داریم:

$$x_{i+1} < x_i$$

در نتیجه بازهم حرکت بسمت چپ، یعنی
بسمت مینیمم هست.

- نرخ یادگیری اگر خیلی بزرگ باشد، مشکل نوسان بین دو سمت (oscilation) پیش میاد و اگر خیلی کوچیک باشه، سرعت یادگیری خیلی میاد پائین. پس باید یه عدد منطقی و معقول رو انتخاب کنید. مثلاً از 0.1 شروع کنید و اگر خیلی بزرگ بود، 0.01 بگیرید.
- ضرایبی که باید تغییرشون بدیم تا در هر مرحله نتیجه‌مون بهتر بشه، وزن‌ها و اریبی‌ها هستند. پس در هر مرحله قاعده‌ی بروزرسانی اونها به شکل زیر هست:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \nabla_{\mathbf{w}} L(\mathbf{w}_i) = \mathbf{w}_i - \eta \sum_i \mathbf{x}_i \delta_i \quad \text{and} \quad b_{i+1} = b_i - \eta \nabla_b L(b_i) = b_i - \eta \textstyle\sum_i \delta_i$$

$\delta_i = y_i - t_i$ where y_i comes from our Model and t_i from our Target.

در حالتی که با چند متغیر سر و کار داریم، گرادیان کاهشی یه کوچولو تغییر شکل میده

$$x_{i+1} = x_i - \eta f'(x_i) \to w_{i+1} = w_i - \eta \nabla_W L(w_i)$$

که در رابطه‌ی فوق W یه ماتریسه که ما می‌خواهیم از تابع هزینه نسبت به اون گرادیان بگیریم.

بریم در اسلاید بعدی کمی محاسبات مشتق رو ببینیم

Model: $y = xw + b$

Loss: $L = \frac{1}{2}\sum_i (y_i - t_i)^2$

Update rule: $\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \eta \nabla_{\mathbf{w}} L(\boldsymbol{w}_i)$

$b_{i+1} = b_i - \eta \nabla_b L(b_i)$

ANY function that holds the basic property of being higher for worse results and lower for better results can be a loss function.

$$\nabla_{\mathbf{w}} L = \nabla_{\mathbf{w}} \frac{1}{2} \sum_i (y_i - t_i)^2 =$$

$$= \nabla_{\mathbf{w}} \frac{1}{2} \sum_i \left((\boldsymbol{x}_i \boldsymbol{w} + b) - t_i\right)^2 =$$

$$= \sum_i \nabla_{\mathbf{w}} \frac{1}{2} (\boldsymbol{x}_i \boldsymbol{w} + b - t_i)^2 =$$

$$= \sum_i \boldsymbol{x}_i (\boldsymbol{x}_i \boldsymbol{w} + b - t_i) =$$

$$= \sum_i \boldsymbol{x}_i (y_i - t_i) \equiv$$

$$\equiv \sum_i \boldsymbol{x}_i \delta_i$$

Coding or Excelling time!!!

Let’s go to the next video