دانشگاه سمنان
Semnan University
پردیس فرزانگان

بسمه تعالی

فصل ششم داده کاوی

شبکه عصبی
Artificial Neural Networks

مدرس

فاطمه دارائی

f_daraei@semnan.ac.ir

https://fdaraei.profile.semnan.ac.ir

دانشگاه سمنان
Semnan University
پردیس فرزانگان

Artificial Neural Networks

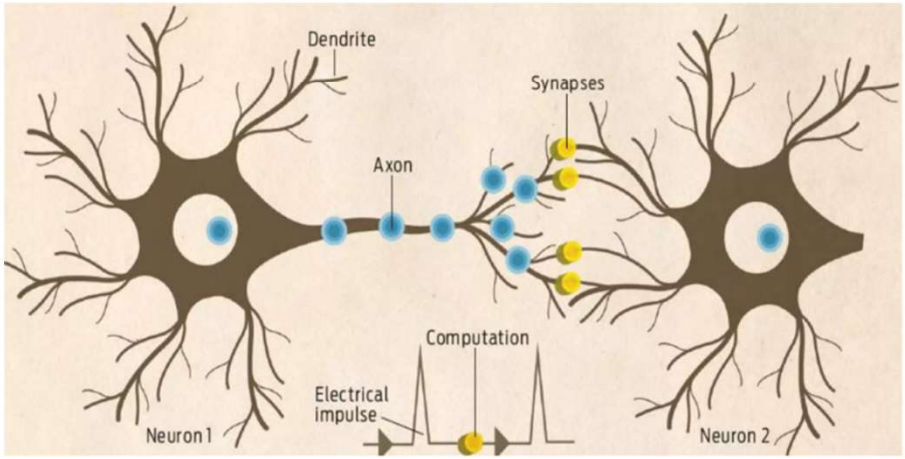
شبکه‌های عصبی مصنوعی یا به اختصار **ANN** الگوریتم‌های محاسباتی الهام‌گرفته از ساختار و عملکرد مغز انسان هستند که برای حل مسائل پیچیده در حوزه‌های مختلف مانند یادگیری ماشین، هوش مصنوعی و علم داده به کار می‌روند.

این شبکه‌ها با الهام از نورون‌های زیستی، سعی در تقلید روش پردازش اطلاعات در مغز دارند.

دانشگاه سمنان
Semnan University
پردیس فرزانگان

ساختار شبکه عصبی مغز انسان

سلولهای مغز انسان دارای ساختار متفاوتی از سایر سلولهای بدن انسان می‌باشند به این سلولهای مغزی نرون گفته می‌شود. هر نرون یک بدنه، یک آکسون و چندین دندریت داشته و واسط بین آکسون یک نرون و دندریتهای نرونهای دیگر سیناپس نام دارد. همچنین هر نرون بر اساس یک آستانه تحریک در یکی از دو وضعیت تحریک شده و ساکن قرار می‌گیرند.



ساختار سلول عصبی

نورون‌ها Neurons

نورون‌ها واحدهای اصلی پردازش اطلاعات در مغز هستند.

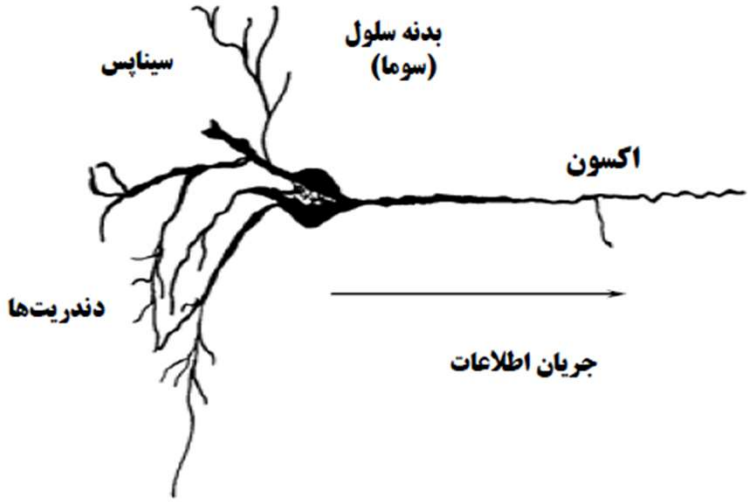
مغز انسان حدود **۸۶ میلیارد نورون** دارد که به یکدیگر متصل هستند.

هر نورون از سه بخش اصلی تشکیل شده است:

دندریت‌ها Dendrites شاخه‌هایی که اطلاعات را از نورون‌های دیگر دریافت می‌کنند.

محور عصبی Axon ساختاری بلند که اطلاعات را به نورون‌های دیگر انتقال می‌دهد.

سیناپس‌ها Synapses نقاط اتصال بین نورون‌ها که انتقال سیگنال‌ها را امکان‌پذیر می‌سازند.



این ساختار نرون در مغز انسان به تعداد 10^{11} تکرار می‌شود و از آنجا که هر نرون حداقل به ۱۰۰۰۰ نرون دیگر متصل می‌باشد، در مغز انسان 10^{15} اتصال سیناپسی وجود دارد که تمامی فعالیتهای ذهنی را به انجام می‌رسانند.

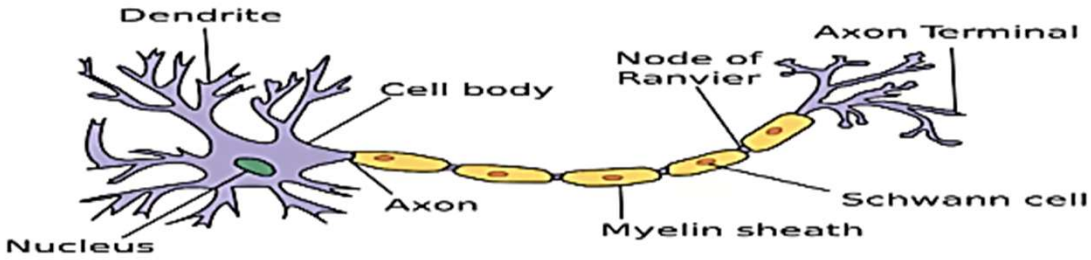
دانشگاه سمنان
Semnan University
پردیس فرزانگان

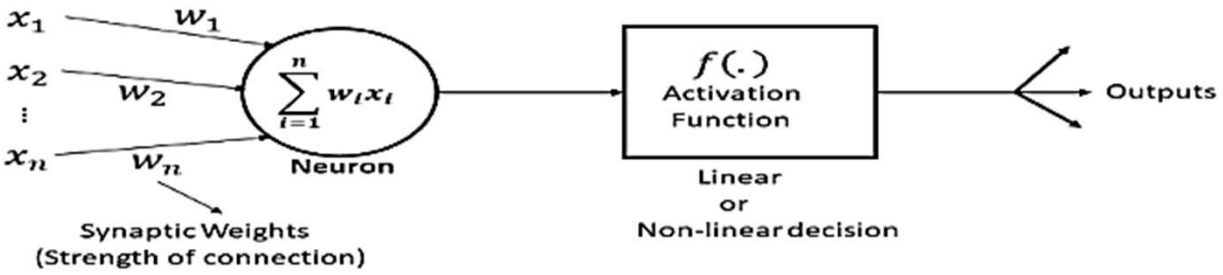
مدل ریاضی شبکه عصبی

1.وزن‌دهی و جمع خطی: هر ورودی با یک وزن **Weight** ضرب می‌شود و سپس نتایج با هم جمع می‌شوند.

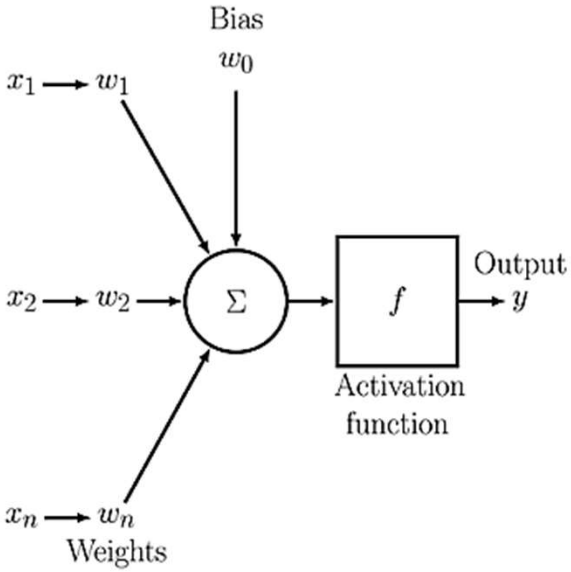
2.فعال‌سازی: جمع خطی حاصل، از یک تابع فعال‌سازی **Activation Function** عبور می‌کند تا خروجی نورون تعیین شود. این تابع می‌تواند غیرخطی باشد و به مدل اجازه دهد روابط پیچیده‌تری را یاد بگیرد.

3.انتقال داده: خروجی هر نورون به نورون‌های لایه بعد منتقل می‌شود

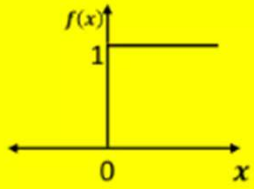
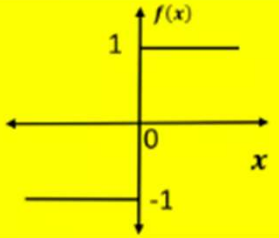
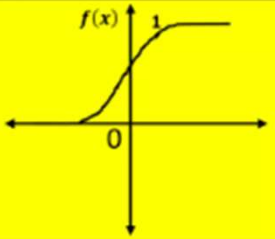




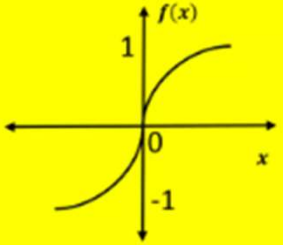
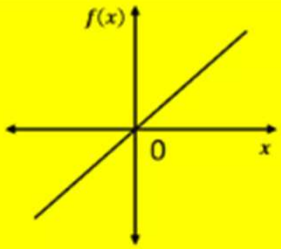
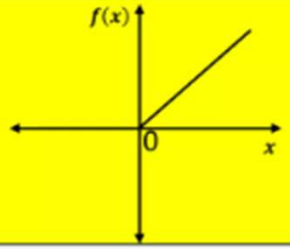
توابع فعال ساز



$$y = f(\sum_{i=0}^{n} w_i x_i)$$

Name	Geometrical Shape	Mathematical Expression
Hard Limit		$f(x) = \begin{cases} 1 & if\ x \geq 0 \\ 0 & otherwise \end{cases}$
Bipolar Hard Limit Signum Function		$f(x) = \begin{cases} 1 & if\ x > 0 \\ 0 & if\ x = 0 \\ -1 & if\ x < 0 \end{cases}$
Sigmoid Function		$f(x) = \frac{1}{1+e^{-x}}$

توابع فعال ساز

Name	Geometrical Shape	Mathematical Expression
Hyperbolic Tangent or Bipolar sigmoid		$f(x) = tanhx = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Bipolar Hard Limit Signum Function		$f(x) = x$
Rectified Linear Unit		$f(x) = max(0, x)$

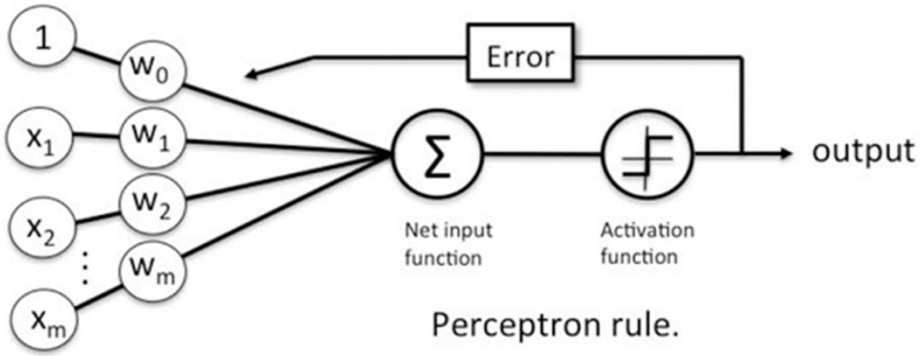
دانشگاه سمنان
Semnan University
پردیس فرزانگان

الگوریتم پرسپترون

- پرسپترون در دسته الگوریتم های یادگیری با نظارت قرار می گیرد. الگوریتم پرسپترون یک الگوریتم خطی است. شبکه عصبی پرسپترون از جمله ساده ترین معماری های شبکه عصبی مصنوعی است.

$$W_{new} = W_{old} + \eta \times (y - \hat{y}) \times X$$

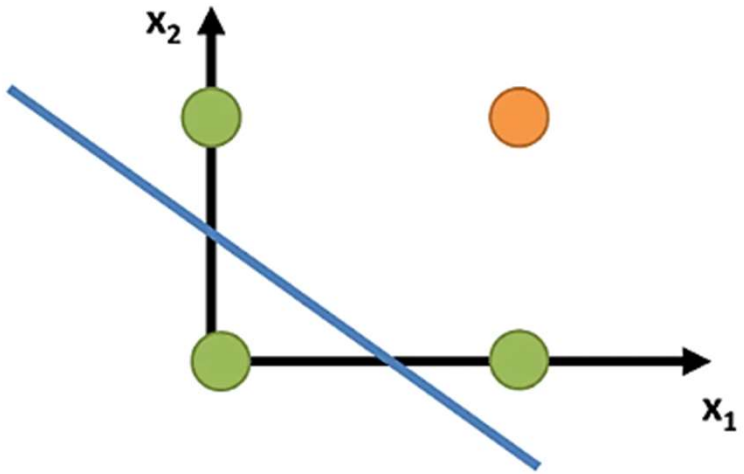
- W = weights
- η = learning rate
- y = actual output
- $\hat{y}$ = predicted output
- X = input feature vector



Perceptron rule.

مثال یادگیری پرسپترون: گیت AND

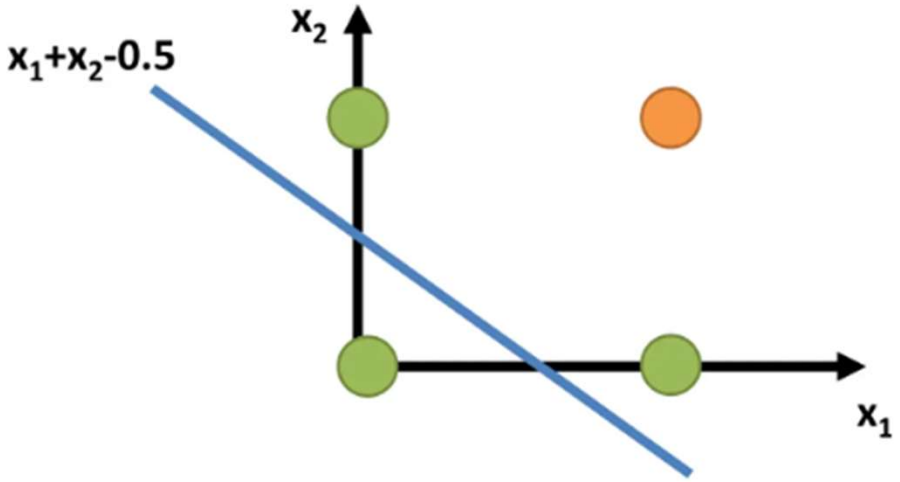
x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1



- $\mu=0.3$
- $w_0=-.5, w_1=1, w_2=1$
- $\theta = 0$ ⟶ $\theta(x)>0 = 1$ otherwise 0

ادامه مثال

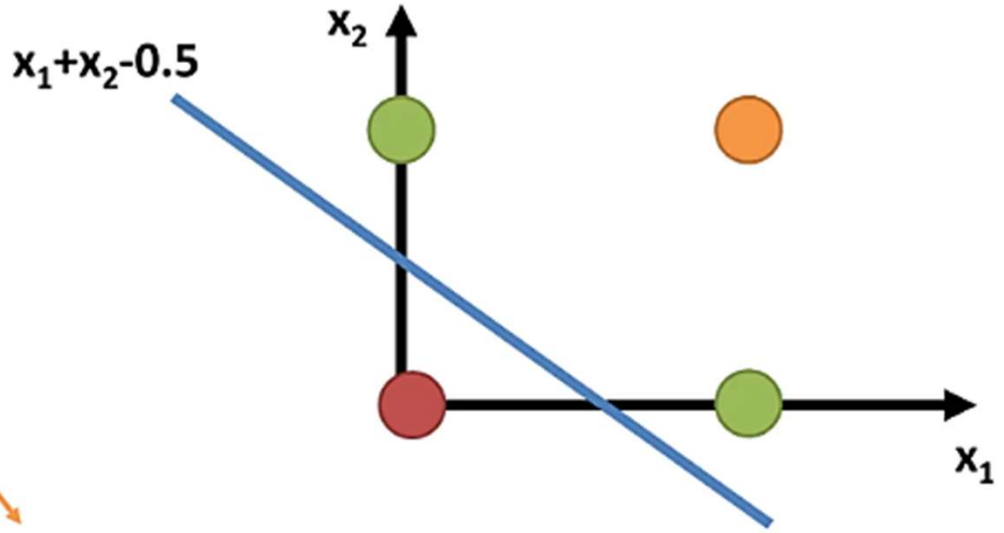
x_1	x_2	y_a
0	0	0
0	1	0
1	0	0
1	1	1



- w_0=-.5, w_1=1, w_2=1
- net= $w_1x_1+w_2x_2+w_0x_0 = x_1+x_2-0.5$

ادامه مثال

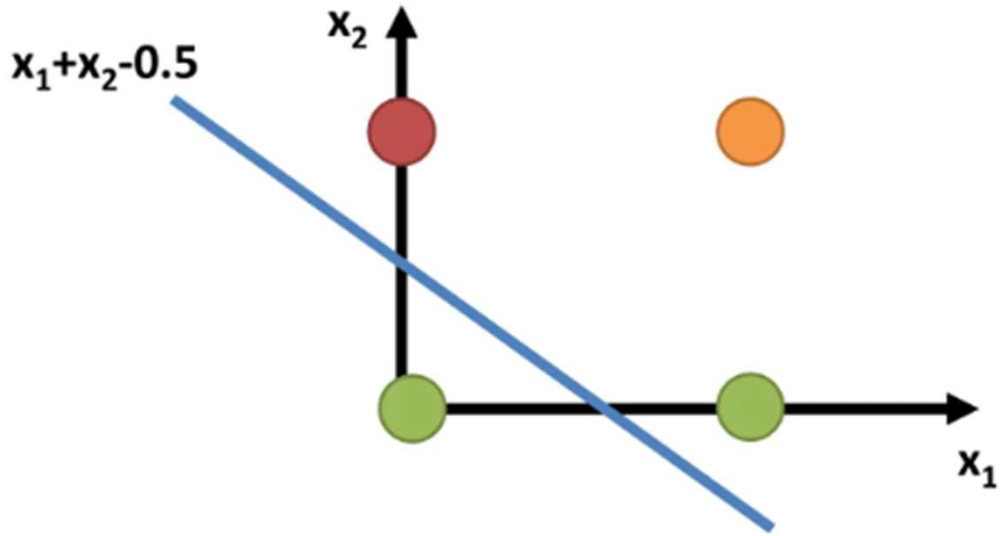
x_1	x_2	y_a
0	0	0
0	1	0
1	0	0
1	1	1



- $net = w_1x_1+w_2x_2+w_0x_0 = x_1+x_2-0.5 = 0+0-0.5=-0.5$
- $y_d = \theta(net) = \theta(-0.5) = 0$
- $y_d = y_a$ correct

ادامه مثال

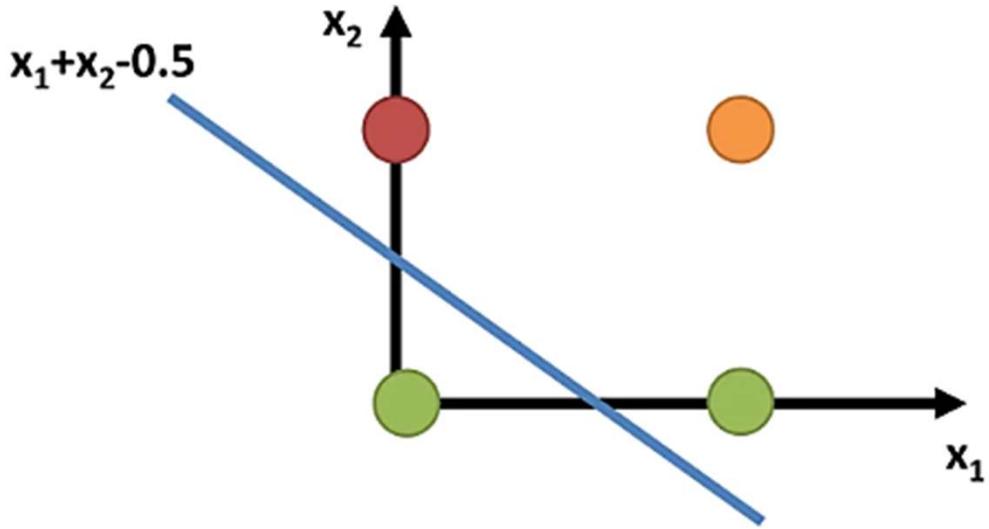
x_1	x_2	y_a
0	0	0
0	1	0
1	0	0
1	1	1



- $net = w_1x_1+w_2x_2+w_0x_0 = x_1+x_2-0.5 = 0+1-0.5 = +0.5$
- $y_d = \theta(net) = \theta(+0.5) = +1$
- $y_d \mathrel{!}= y_a$ incorrect

ادامه حل مثال

x_1	x_2	y_a
0	0	0
0	1	0
1	0	0
1	1	1

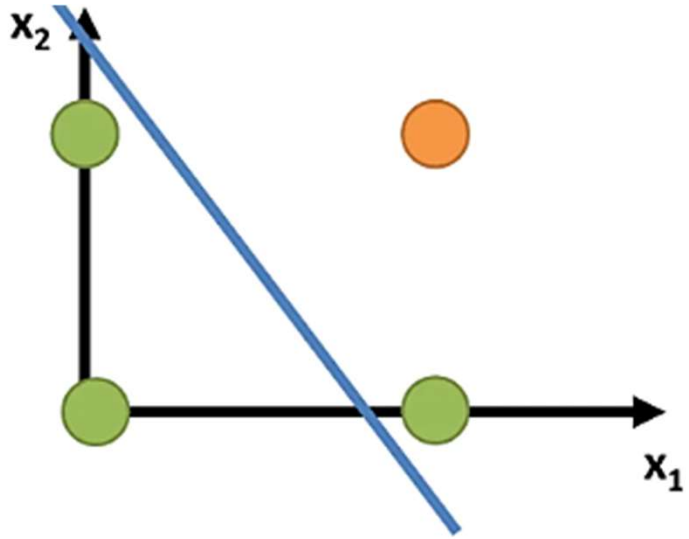


- Update weight $w_i=w_i - \mu * x_i$
- $w_0=-0.5-0.3*1=-0.8$ & $w_1= 1-0.3*0=1$ & $w_2=1-0.3*1=0.7$

ادامه حل مثال

x_1	x_2	y_a
0	0	0
0	1	0
1	0	0
1	1	1

$x_1+0.7x_2-0.8$

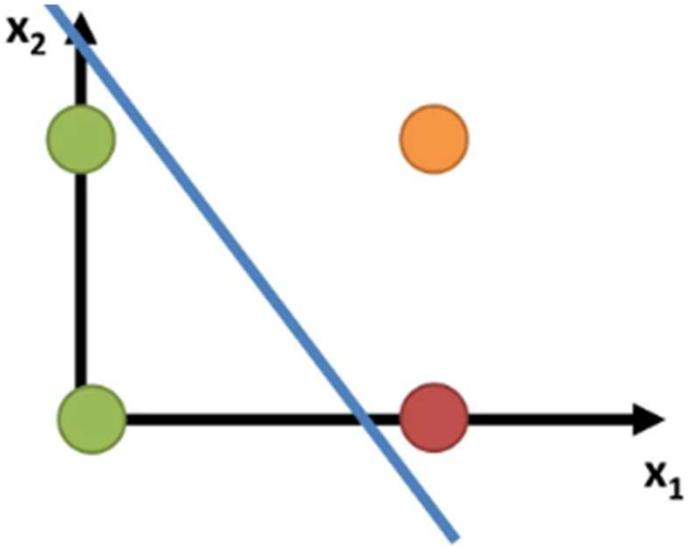


- $w_0=-0.8$ & $w_1=1$ & $w_2=0.7$

ادامه مثال

x_1	x_2	y_a
0	0	0
0	1	0
1	0	0
1	1	1

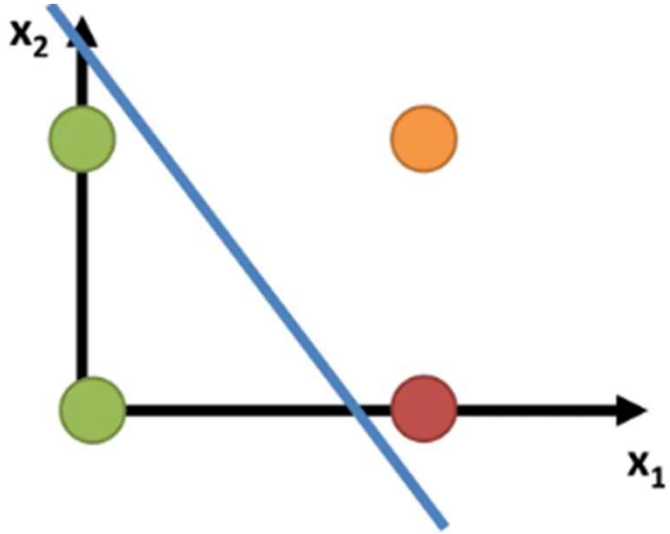
$x_1+0.7x_2-0.8$



- $net = w_1x_1+w_2x_2+w_0x_0 = x_1+0.7x_2-0.8 = 1+0-0.8 = +0.2$
- $y_d = \theta(net) = \theta(+0.2) = +1$
- $y_d \mathrel{!}= y_a$ incorrect

x_1	x_2	y_a
0	0	0
0	1	0
1	0	0
1	1	1

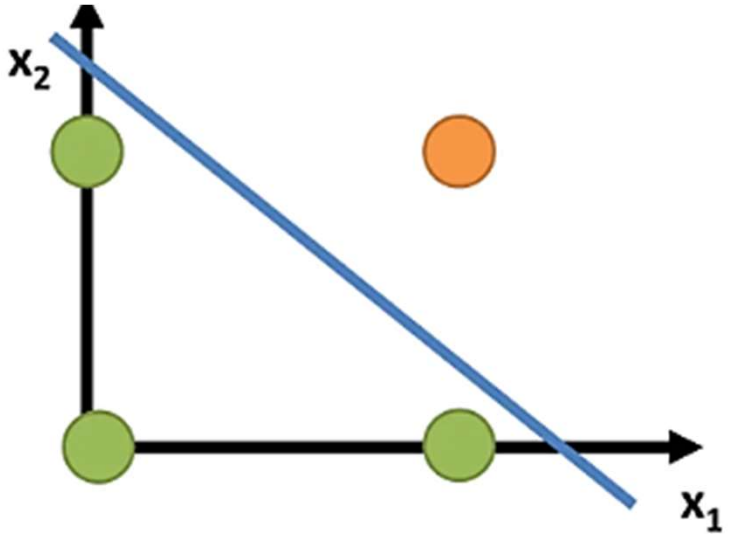
$x_1+0.7x_2-0.8$



- Update weight $w_i = w_i - \mu * x_i$
- w_0=-0.8-0.3*1=-1.1 & w_1= 1-0.3*1=0.7 & w_2=0.7-0.3*0=0.7

x_1	x_2	y_a
0	0	0
0	1	0
1	0	0
1	1	1

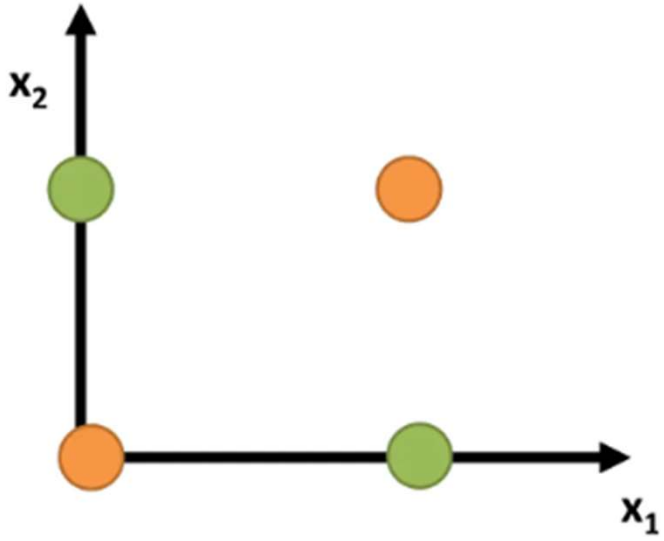
$0.7x_1+0.7x_2-1.1$



- $w_0= -1.1$ & $w_1= 0.7$ & $w_2= 0.7$

محدودیت پرسپترون در مسائل غیرخطی

پژوهشگران دریافتند که **پرسپترون** نمی‌تواند بسیاری از **توابع تصمیم‌گیری غیرخطی** را تقریب بزند. یکی از شناخته‌شده‌ترین مثال‌ها در این زمینه، **مسئله XOR** است.



پرسپترون فقط می‌تواند داده‌هایی را طبقه‌بندی کند که خطی جداپذیر باشند.

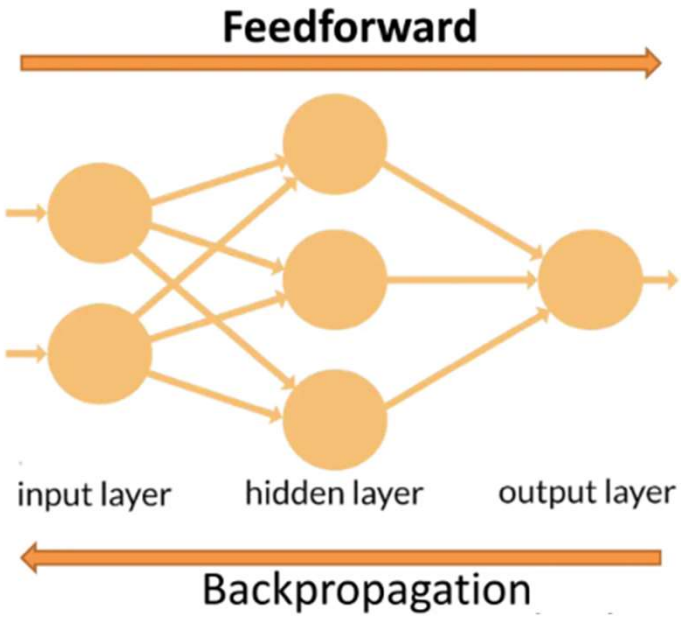
در مسئله XOR، هیچ خط مستقیمی وجود ندارد که بتواند نقاط را به‌درستی جدا کند.

راه حل

برای حل چنین مسائل غیرخطی، مدل‌های پیچیده‌تر توسعه داده شدند:

شبکه‌های چندلایه Multi-Layer Perceptron یا MLP

با اضافه کردن لایه‌های پنهان Hidden Layers و استفاده از توابع فعال‌سازی غیرخطی، می‌توان مسائل غیرخطی را حل کرد.



الگوریتم‌های یادگیری پیشرفته‌تر:

روش‌هایی مانند **پس‌انتشار خطا Backpropagation** برای آموزش شبکه‌های چندلایه معرفی شدند.

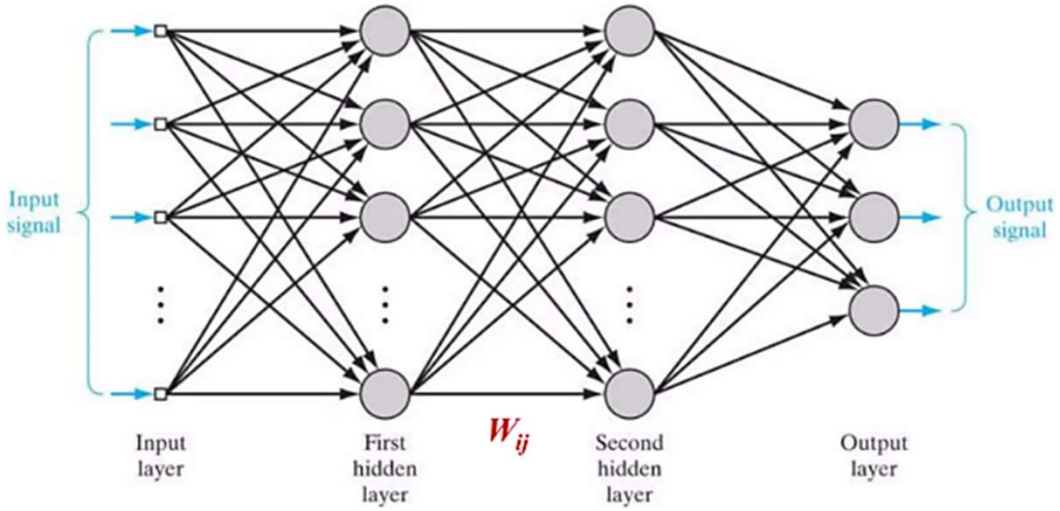
Multi-Layer Perceptron

آموزش در یک **پرسپترون چندلایه** MLP یا Multi-Layer Perceptron به فرآیندی گفته می‌شود که در آن شبکه یاد می‌گیرد وزن‌ها و بایاس‌های خود را تنظیم کند تا خطای بین خروجی پیش‌بینی‌شده و خروجی واقعی به حداقل برسد.

این فرآیند از دو مرحله اصلی تشکیل شده است:

پراکندگی رو به جلو Forward Propagation

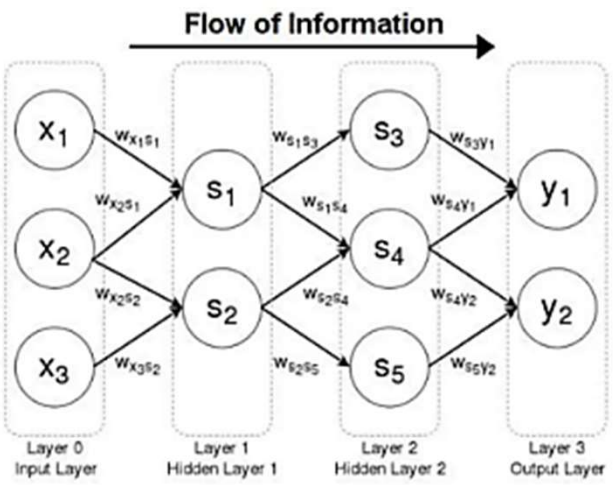
پراکندگی بازگشتی Backpropagation



دانشگاه سمنان
Semnan University
پردیس فرزانگان

Forward Propagation

این مرحله شامل محاسبات لازم برای پیش‌بینی خروجی توسط شبکه می‌باشد.



۱- داده ورودی

مقادیر ورودی (X1,X2,...,Xn) به نورون‌های لایه ورودی داده می‌شود.

۲- محاسبه مجموع وزن‌دار

برای هر نورون در لایه پنهان، مقادیر ورودی با وزن‌هایشان ضرب و سپس بایاس اضافه می‌شود.

۳- اعمال تابع فعال‌سازی Activation Function

خروجی نورون توسط یک تابع فعال‌سازی (مانند ReLU، سیگموید، یا تانژانت هایپربولیک) پردازش می‌شود تا غیرخطی بودن شبکه فراهم شود.

۴- محاسبه در لایه‌های بعدی

خروجی نورون‌های لایه پنهان به عنوان ورودی به نورون‌های لایه بعدی داده می‌شود و این فرآیند تکرار می‌شود تا به لایه خروجی برسیم.

۵- پیش‌بینی خروجی Output Prediction

در نهایت، نورون‌های لایه خروجی مقدار پیش‌بینی‌شده را تولید می‌کنند.

محاسبه خطا

پس از انجام پراکندگی رو به جلو، مقدار پیش‌بینی‌شده با مقدار واقعی مقایسه می‌شود. خطا با استفاده از یک تابع هزینه Loss Function محاسبه می‌شود. یکی از رایج‌ترین توابع هزینه **میانگین مربعات خطا MSE** است.

$$MSE(p,y) = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2$$

Binary Cross-Entropy Loss

$$BCE(p,y) = -[ylog(p) + (1-y)\log(1-p)]$$

Categorical Cross-Entropy Loss

$$CCE(p,y) = \sum_i y_i \log(p_i)$$

Backpropagation

هدف این مرحله، کاهش خطا با به‌روزرسانی وزن‌ها و بایاس‌ها است. این فرآیند با استفاده از الگوریتم گرادیان نزولی Gradient Descent انجام می‌شود.

۱- محاسبه گرادیان‌ها:

گرادیان‌ها نشان‌دهنده نرخ تغییر خطا نسبت به وزن‌ها و بایاس‌ها هستند. این مقدار با استفاده از مشتق تابع هزینه و قوانین زنجیره‌ای Chain Rule محاسبه می‌شود.

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial a_j} \cdot \frac{\partial a_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ij}}$$

۲- انتشار خطا به عقب:

خطای لایه خروجی به لایه‌های قبلی منتقل می‌شود. این خطا با وزن‌های نورون‌ها ضرب شده و توزیع می‌گردد.

۳- به‌روزرسانی وزن‌ها و بایاس‌ها:

وزن‌ها و بایاس‌ها با استفاده از گرادیان‌های محاسبه‌شده به‌روزرسانی می‌شوند.

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial L}{\partial w_{ij}}$$

مثال

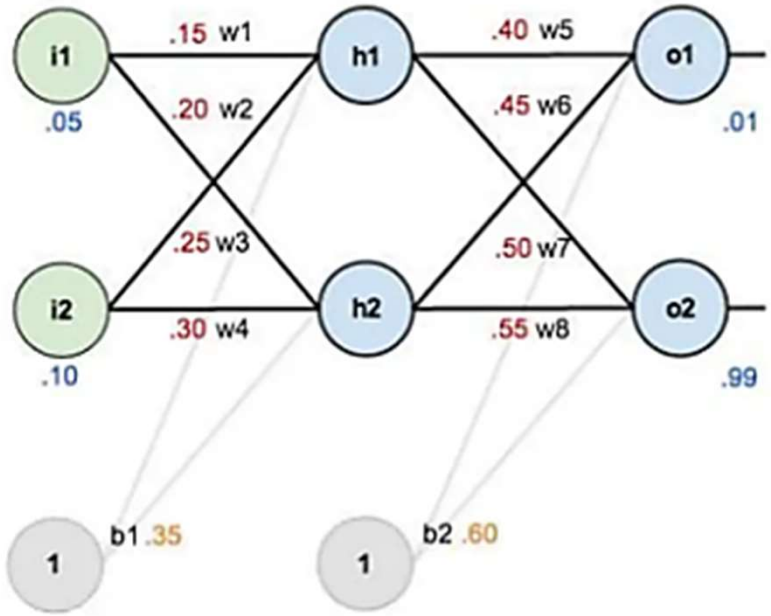
Outputs of the Hidden Layer neurons:

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1$$
$$= 0.15 * 0.05 + 0.2 * 0.1 + 0.35$$
$$= 0.3775$$

$$out_{h1} = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-0.3775}} = 0.593$$

$$net_{h2} = w_3 * i_1 + w_4 * i_2 + b_1$$
$$= 0.25 * 0.05 + 0.3 * 0.1 + 0.35$$
$$= 0.3925$$

$$out_{h2} = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-0.3925}} = 0.597$$



ادامه مثال

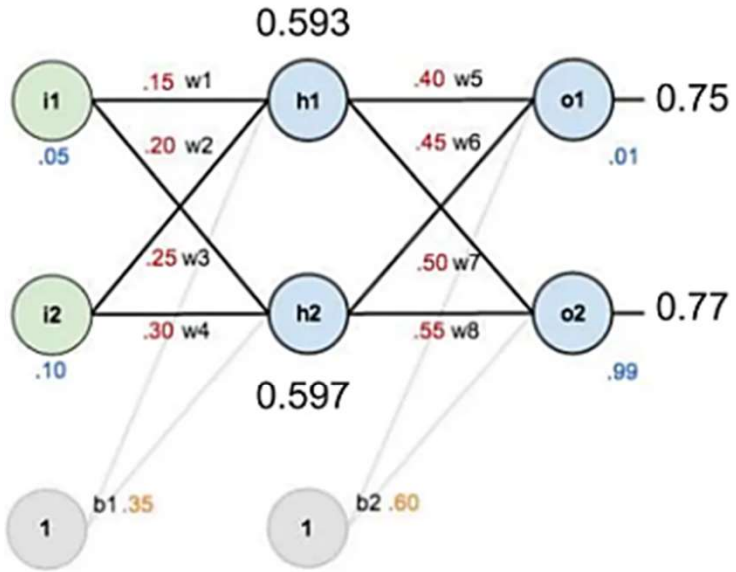
Outputs of the Hidden Layer neurons:

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2$$
$$= 0.4 * 0.593 + 0.45 * 0.597 + 0.60$$
$$= 1.1$$

$$out_{o1} = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-1.1}} = 0.75$$

$$net_{o2} = w_7 * out_{h1} + w_8 * out_{h2} + b_2$$
$$= 0.5 * 0.593 + 0.55 * 0.597 + 0.60$$
$$= 1.22$$

$$out_{o2} = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-1.22}} = 0.77$$



The neuron with the highest value determines the output.

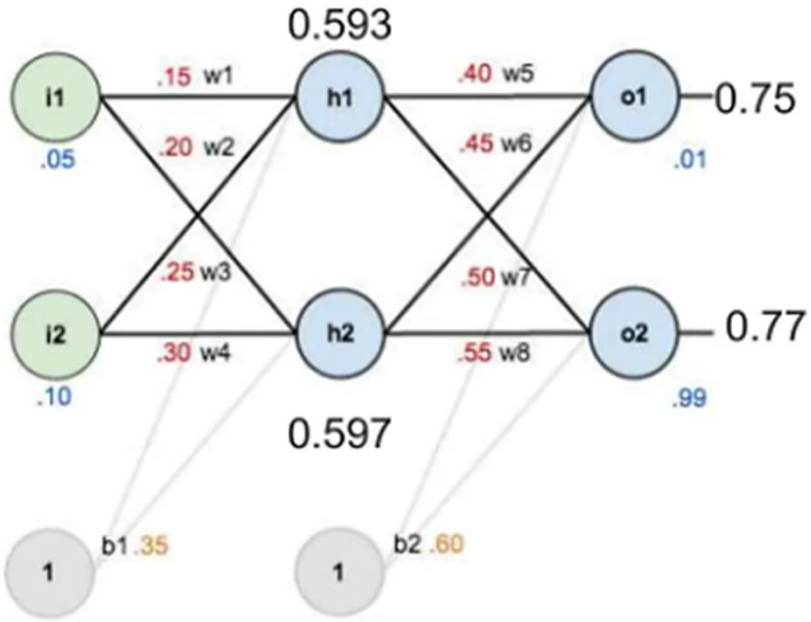
دانشگاه سمنان
Semnan University
پردیس فرزانگان

ادامه مثال: محاسبه خطا

$$E_{total} = MSE(p, y) = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2$$

$$E_{total} = \frac{1}{2}[(target_{o1} - out_{o1})^2 + (target_{o2} - out_{o2})^2]$$

$$E_{total} = \frac{1}{2}[(0.01 - 0.75)^2 + (0.99 - 0.77)^2] = 0.298$$



- Change in error with respect to the output:

$$E_{total} = \frac{1}{2}[(target_{o1} - out_{o1})^2 - (target_{o2} - out_{o2})^2]$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1}) * -1 + 0$$

$$= (0.01 - 0.75) * -1 = 0.74$$

- Change in output with respect net input:

$$out_{o1} = \frac{1}{1 + e^{-x}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) \quad \text{.....[23]}$$

$$= 0.75(1 - 0.75) = 0.186$$

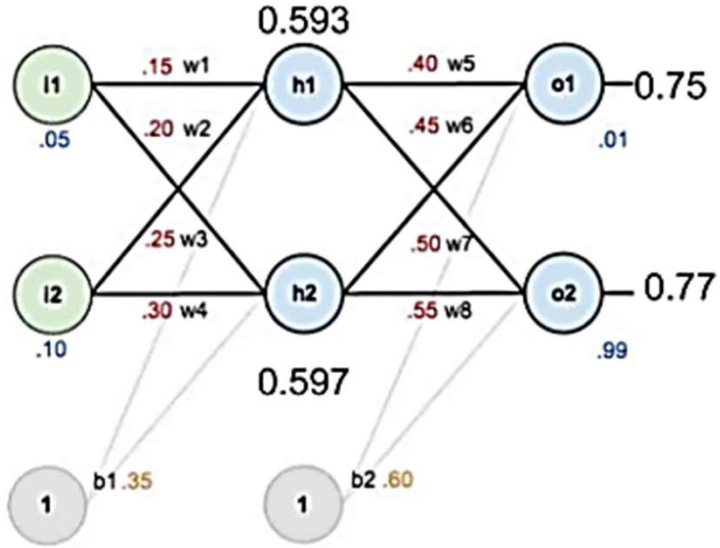
- Change in net input with respect to weight:

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} + 0 + 0 = 0.593$$

- Change in error with respect to the weight:

$$\frac{\partial E_{total}}{\partial w_5} = 0.74 * 0.186 * 0.59 = 0.082$$



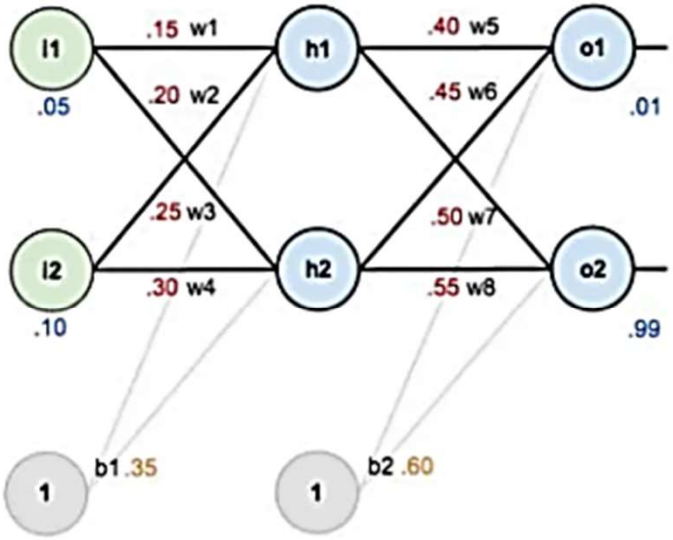
ادامه مثال: به روزرسانی وزن ها

- To decrease the error, we then update the weight
  - Assume the learning, $\mu = 0.5$

$$w_5^+ = w^5 - \mu * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082 = 0.358$$

Similarly, we get

$$w_6^+ = 0.408$$
$$w_7^+ = 0.511$$
$$w_8^+ = 0.561$$
$$w_1^+ = 0.149$$
$$w_2^+ = 0.199$$
$$w_3^+ = 0.249$$
$$w_4^+ = 0.299$$



- Repeat this process until the desired accuracy is reached.

نرخ یادگیری

نرخ یادگیری مناسب کلید آموزش موفق مدل است و تاثیر مستقیمی بر دقت و کارایی آن دارد.

نرخ یادگیری مشخص می‌کند که وزن‌ها و بایاس‌های مدل در پاسخ به خطای تخمین زده‌شده، در هر به‌روزرسانی چه مقدار تغییر می‌کنند.

نرخ یادگیری بسیار بزرگ:

باعث می‌شود مدل از مقدار بهینه Optimal Value عبور کند و همگرا نشود.

مدل به طور مکرر نوسان می‌کند و قادر به یافتن پاسخ بهینه نخواهد بود.

نرخ یادگیری بسیار کوچک:

باعث می‌شود فرآیند بهینه‌سازی بسیار کند شود.

مدل برای رسیدن به حداقل مقدار تابع هزینه به تعداد زیادی تکرار نیاز دارد