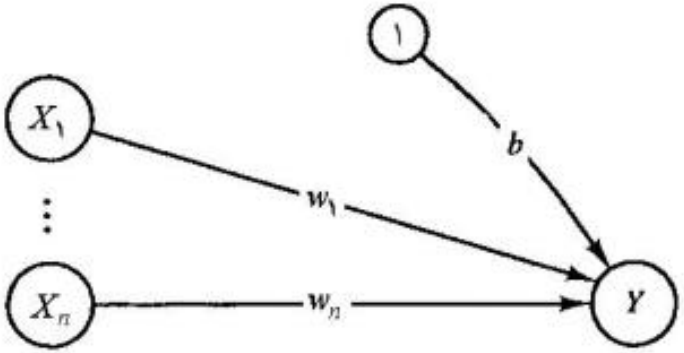
بسم الله الرحمن الرحيم

آدالاین = نرون خط وفقی(ADAptive LInear Neuron)

Widrow-Hoff Learning Rule

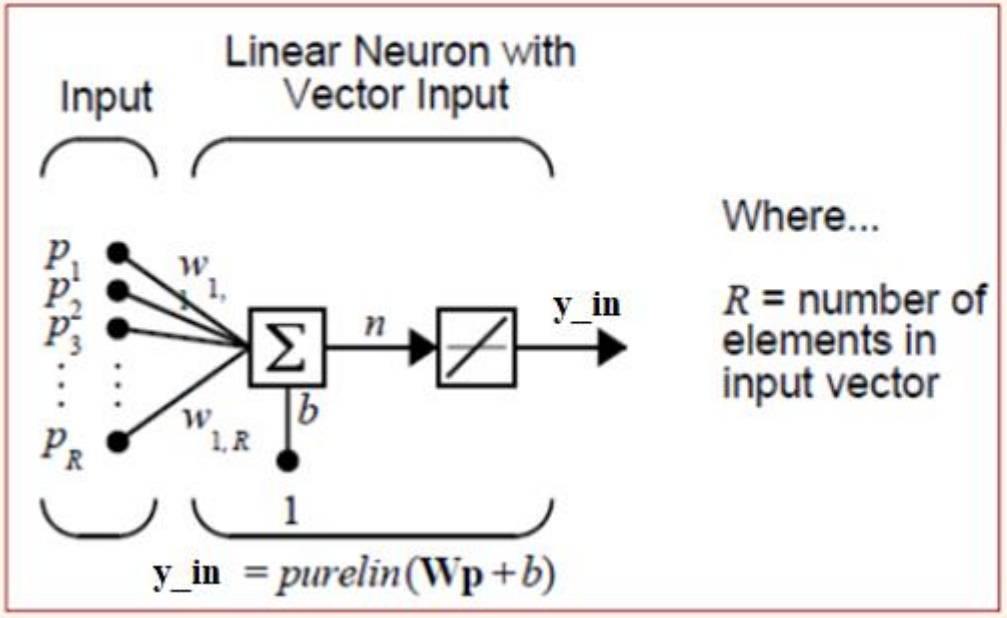
شبکه عصبی آدلاین

- توسط ویدور و هاف در سال ۱۹۶۰
- قانون یادگیری = قانون میانگین مربعات کمینه (LMS) = قانون ویدرو–هاف
  - میانگین مربعات خطای بین مقدار خروجی شبکه و مقدار هدف در هر مرحله از آموزش کاهش یابد
- تابع فعال‌سازی خروجی = تابع همانی
- ساختار مشابه با سایر شبکه‌های قبلی
  - چند ورودی
  - بایاس = ورودی برابر با ۱



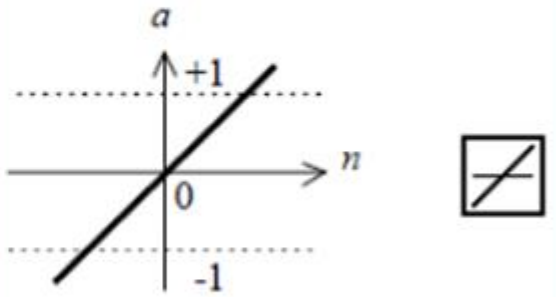
شبکه عصبی آدلاین

- ADALINE همانند پرسپترون می‌تواند مسائل جدایی‌پذیر خطی را حل کند.



Where...

R = number of elements in input vector



y_in = purelin(n)

Linear Transfer Function

محاسبه خطا در شبکه عصبی آدلاین

- با فرض این که واحد خروجی دارای تابع فعالیت (انتقال) خطی باشد.
- به ازای n ورودی مساله را بررسی می‌کنیم.

$X^1 \xrightarrow{W(1)} y_1$ e_1 will be generated

$X^2 \xrightarrow{W(1)} y_2$ e_2 "

$X^3 \xrightarrow{W(1)} y_3$ e_3 "

.....

$X^n \xrightarrow{W(1)} y_n$ e_n "

$$E = \sum_{i=1}^{N} [e_i]^2$$

SSE

$$E = \frac{\sum_{i=1}^{N} [e_i]^2}{N}$$

Mean SSE

آموزش شبکه عصبی آدلاین

- مرحلهٔ ۰- مقداردهی اولیه به وزن‌ها (مقادیر تصادفی کوچک)

  مقداردهی به نرخ یادگیری
- مرحلهٔ ۱- تا زمانی‌که شرایط توقف برقرار نیست، مراحل ۲ تا ۶ را انجام دهید.
- مرحلهٔ ۲- برای هر جفت آموزش دوقطبی $s:t$ مراحل ۳ تا ۵ را انجام دهید.
- مرحلهٔ ۳- فعال‌سازی‌های واحدهای ورودی را مشخص کنید: $x_i = s_i \qquad i = 1,\ldots,n$
- مرحلهٔ ۴- مقدار ورودی شبکه را به واحد خروجی محاسبه کنید: $y_in = b + \sum_i x_i w_i$
- مرحلهٔ ۵- مقادیر وزن‌ها و بایاس را به‌روز کنید:

$$\begin{cases} b(new) = b(old) + \alpha.(t - y_in) \\ w_i(new) = w_i(old) + \alpha.(t - y_in).x_i \end{cases}$$

- مرحلهٔ ۶- شرایط توقف را آزمایش کنید:

  اگر بزرگ‌ترین تغییر وزنی که در مرحلهٔ ۲ رخ داده است از یک مقدار کوچک کم‌تر باشد، الگوریتم را متوقف کنید، وگرنه ادامه دهید.

آموزش شبکه عصبی آدلاین

- نرخ یادگیری
  - تاثیر بر سرعت و روند همگرایی الگوریتم
  - روش: مقدار را کوچک فرض کرده (مثلاً 0.2)
  - اگر مقدار خیلی بزرگی باشد، فرآیند یادگیری همگرا نخواهد بود
  - اگر مقدار بسیار کوچکی باشد، یادگیری بسیار کند می‌شود

- الگوریتم LMS وزن‌ها و بایاس را به گونه‌ای تغییر می‌دهد که میانگین مربعات خطا (بین خروجی مطلوب و خروجی واقعی) سیستم را به حداقل برساند.

تابع AND در شبکه عصبی آدلاین

- تابع AND: ورودی‌های دودویی، هدف‌های دوقطبی

x_1	x_2	t
1	1	1
1	0	−1
0	1	−1
0	0	−1

- شبکه بعد از آموزش

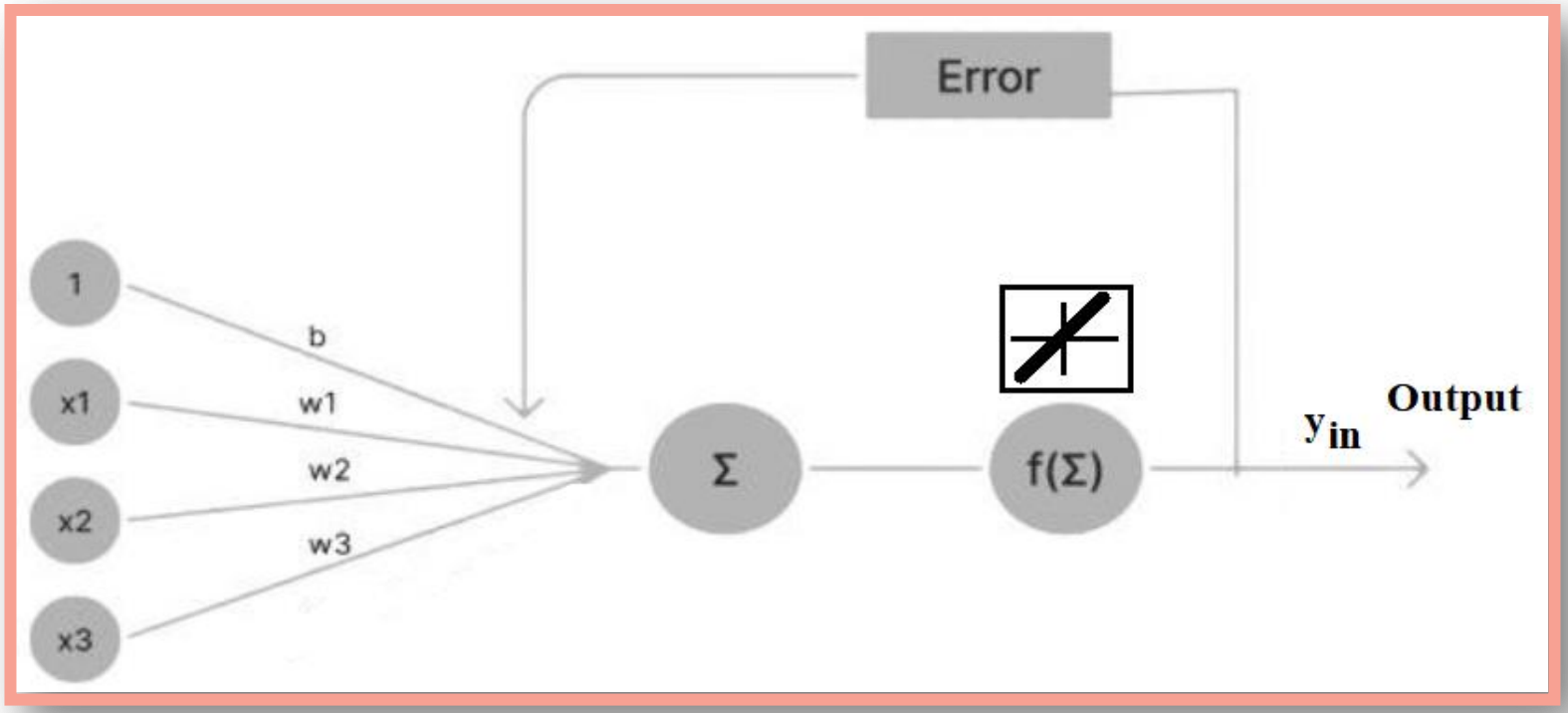
$$w_1 = 1 \quad w_2 = 1 \quad w_0 = -\frac{3}{2}$$

$$x_1 + x_2 - \frac{3}{2} = 0$$

- مربعات خطا برای چهار الگوی آموزش با این وزن‌ها = ۱

$$e = E\{(y-t)^2\} = \sum_{p=1}^{4} [\{x_1(p)w_1 + x_2(p)w_2 + w_0\} - t(p)]^2$$

ADALINE (Adaptive Linear Neural)



$$error : (t - y_{in})^2$$

Design OR gate using Adaline Network

Solution :

- Initially, all weights are assumed to be small random values, say 0.1, and set learning rule to 0.1.
- The weights will be updated until the total error is greater than the least squared error.

x_1	x_2	t
1	1	1
1	-1	1
-1	1	1
-1	-1	-1

Design OR gate using Adaline Network

- Calculate

$$y_{in} = \sum w_i x_i + b$$

$$y_{in} = 0.1 \times 1 + 0.1 \times 1 + 0.1 = 0.3$$ (when $x_1=x_2=1$)

- Now compute, $(t-y_{in})=(1-0.3)=0.7$
- Now, update the weights and bias

$$w_i(new) = w_i(old) + \alpha(t - y_{in})x_i$$
$$w_1(new) = 0.1 + 0.1(1 - 0.3)1 = 0.17$$
$$w_2(new) = 0.1 + 0.1(1 - 0.3)1 = 0.17$$

$$b(new) = b(old) + \alpha(t - y_{in})$$
$$b(new) = 0.1 + 0.1(1 - 0.3) = 0.17$$

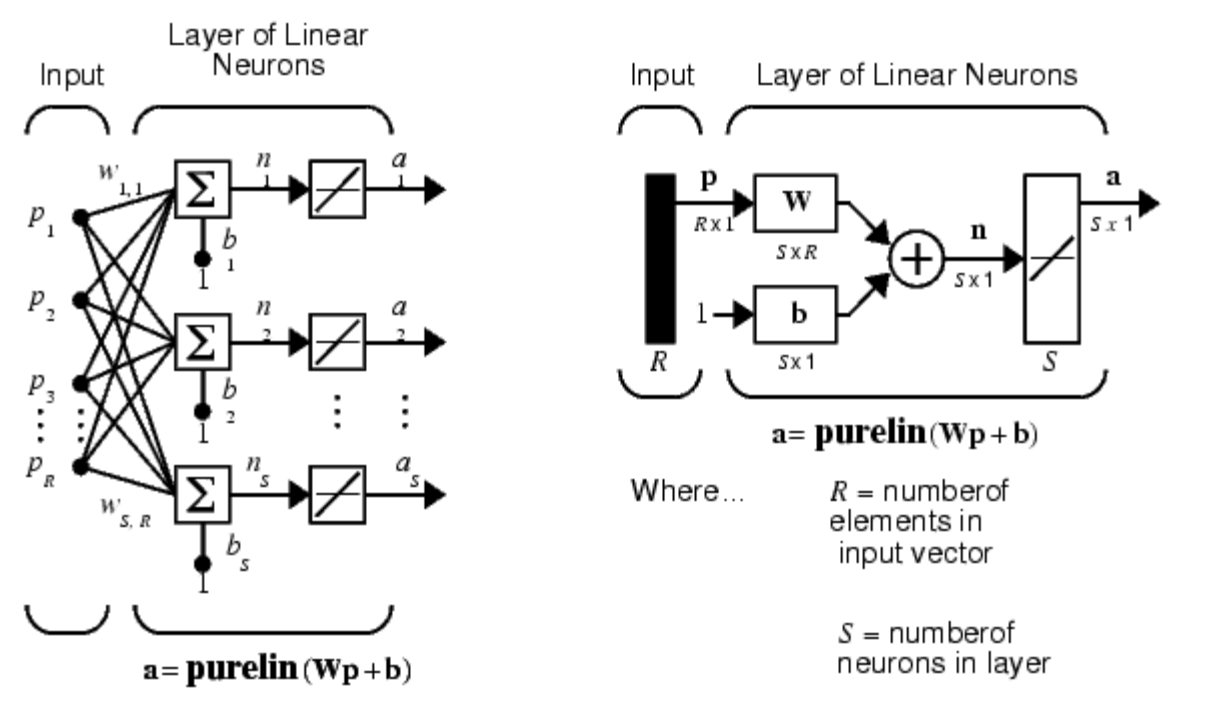
Design OR gate using Adaline Network

x_1	x_2	t	y_{in}	$(t-y_{in})$	Δw_1	Δw_2	Δb	w_1 (0.1)	w_2 (0.1)	b (0.1)	$error^2$
1	1	1	0.3	0.7	0.07	0.07	0.07	0.17	0.17	0.17	0.49
1	-1	1	0.17	0.83	0.083	-0.083	0.083	0.253	0.087	0.253	0.69
-1	1	1	0.087	0.913	-0.0913	0.0913	0.0913	0.1617	0.1783	0.3443	0.83
-1	-1	-1	0.0043	-1.0043	0.1004	0.1004	-0.1004	0.2621	0.2787	0.2439	1.01

$$e = E\{(y-t)^2\} = \sum_{p=1}^{4}[\{x_1(p)w_1 + x_2(p)w_2 + w_0\} - t(p)]^2$$

e=0.49+0.69+0.83+1.01=3.02

MADALINE: Multiple Adaptive Linear Neurons



MADALINE is a multilayer neural network that utilizes multiple ADALINE units to tackle intricate classification tasks with higher complexity.