Types of NNs

Wen Yu

https://www.ctrl.cinvestav.mx/~yuw/
yuw@ctrl.cinvestav.mx

Types of NNs

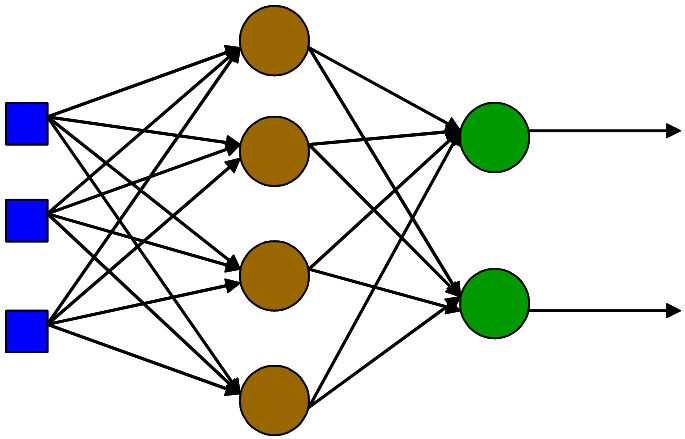
- Time domains
  1. Discrete-time
  2. Continuous-time
- Structure

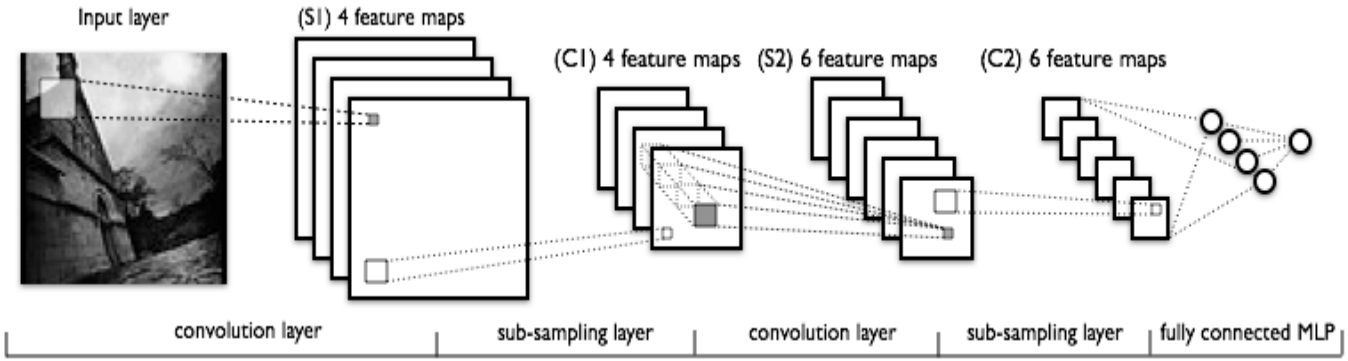
1. Feedforward
2. Recurrent

So

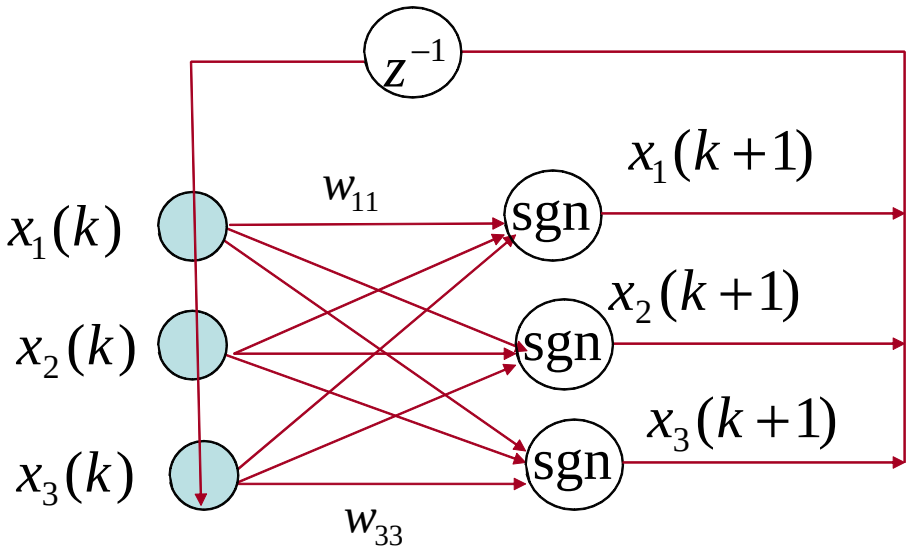
1. Discrete-time Feedforward: MLP
2. Discrete -time Recurrent: RNN
3. Continuous-time Feedforward: MLP
4. Continuous-time Recurrent: *Differential Neural Networks*

MLP




Input layer
(S1) 4 feature maps
(C1) 4 feature maps
(S2) 6 feature maps
(C2) 6 feature maps
convolution layer
sub-sampling layer
convolution layer
sub-sampling layer
fully connected MLP

Parallel form



Deep RNN

- Boltzmann machines
- Long-short term memory (LSTM)

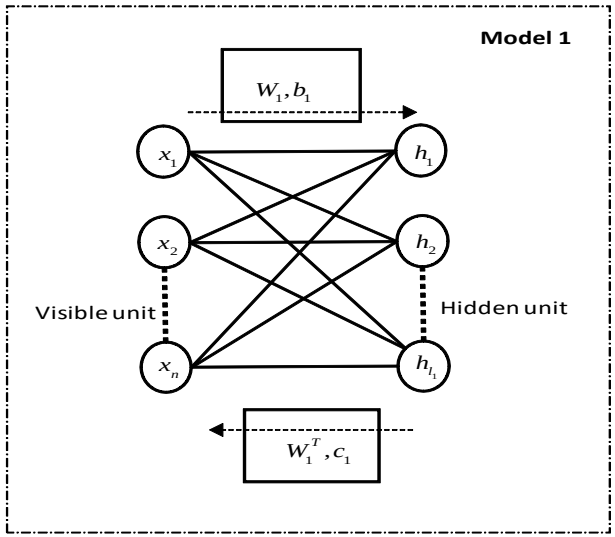
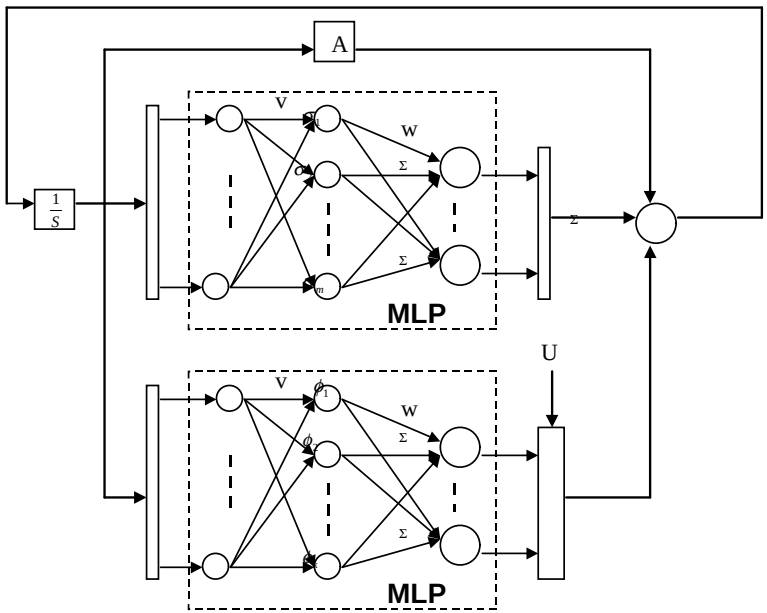


Figure: Markov sampling in a restricted Boltzmann machine

,

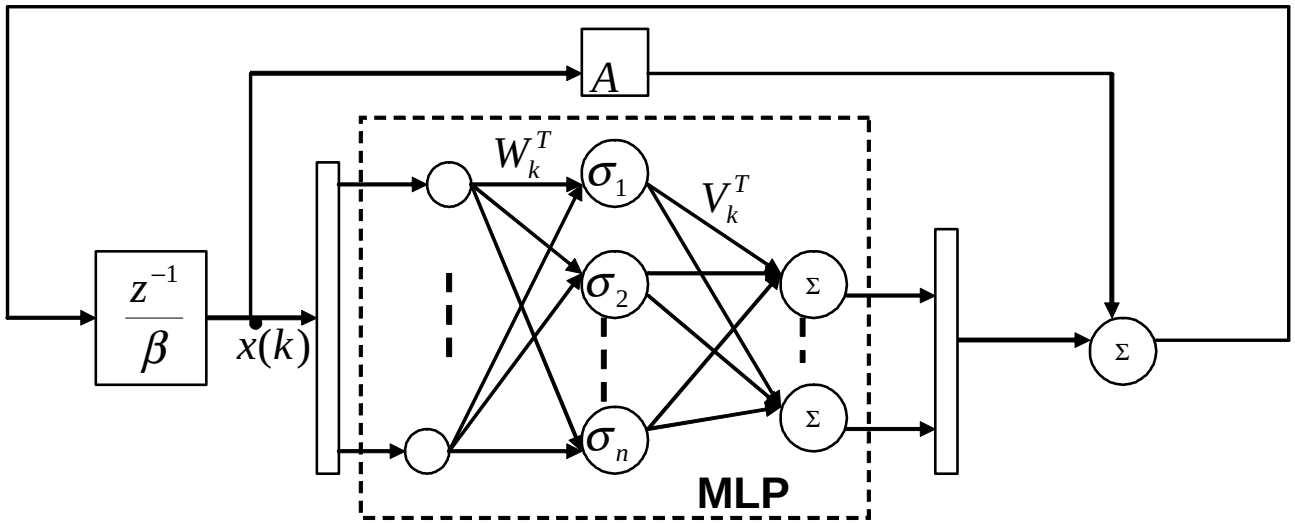
Alexander S Poznyak, Edgar N Sanchez & Wen Yu , *Differential Neural Networks for Robust Nonlinear Control*, World Scientific Publishing Co., 2001

Dynamic NNDifferential NN



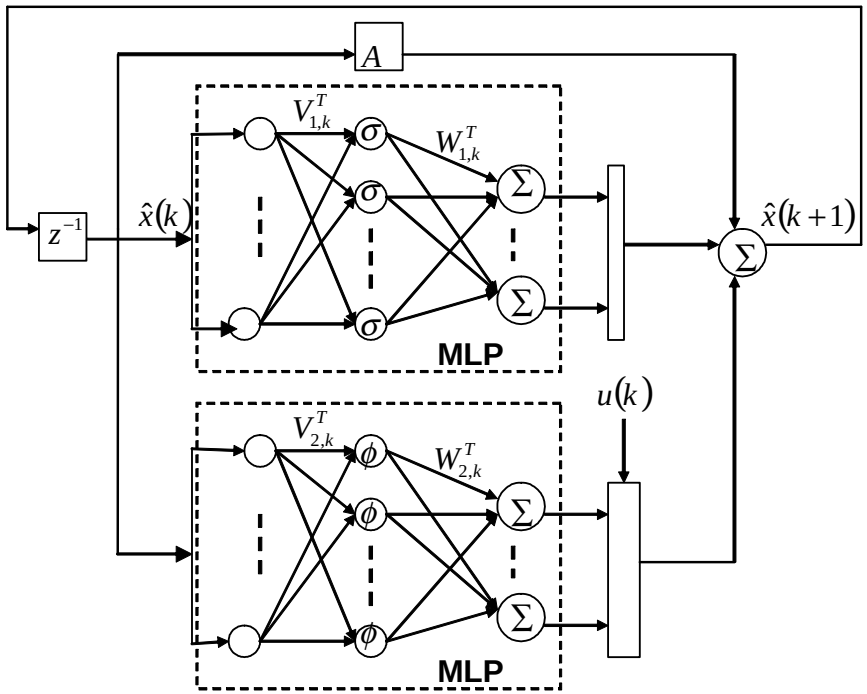
$$\dot{x}_t = Ax_t + W_{1,t}\sigma(V_{1,t}x_t) + W_{2,t}\phi(V_{2,t}x_t)u_t$$

Discrete-time recurrent neural networks



$$\beta\widehat{x}\left(k+1\right)=A\widehat{x}\left(k\right)+V_k\sigma\left[W_kx\left(k\right)\right]$$

Discrete-time recurrent neural networks



$$\hat{x}(k+1) = A\hat{x}(k) + V_{1,k}\sigma\left[W_{1,k}x(k)\right] + V_{2,k}\sigma\left[W_{2,k}x(k)\right]u_k$$

Univeral approximation of *Differential Neural Networks*

We use the Differential Neural Network

$$\frac{d}{dt}\widehat{x} = A\widehat{x} + W\Phi\left[V\widehat{x}\right]$$

to model a nonlinear system, $A \in R^{n\times n}$, $W \in R^{n\times m}$, $V \in R^{m\times n}$, $x \in R^{n}$

$$\dot{x} = f\left(x\right)$$

Theorem

Suppose the dynamic system and the differential Neural Network are initially at the same state $x_0 = \widehat{x}_0$. For any $\varepsilon > 0$ and any finite $T > 0$, there exists an integer m such that the state of differential Neural Network $\widehat{x}_t$ with the weight $\left[W^*, V^*\right]$ satisfies

$$\sup_{0\leq t\leq T} \left|x_t - \widehat{x}_t\right| < \varepsilon \tag{1}$$

Univeral approximation of *Differential Neural Networks*

The nonlinear system,can be expressed as

$$\dot{x} = Ax_t + f(x) - Ax_t = Ax_t + G(x)$$

where $G(x) = f(x) - Ax$
From Stone-Weierstrass theorem if the hidden layer V is large enough, the neural network with one hidden layer

$$\hat{y} = W\Phi[V\hat{x}]$$

can approximate any

$$y = G(x)$$

to any degree of accuracy,

$$\sup_x |G(x) - W\Phi[Vx]| \leq \delta \tag{2}$$

where δ is any small positive constant.

Proof

Wen Yu, Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms, Information Sciences, Vol.158, No.1, 131-147, 2004.

Let us define identification error as

$$\Delta_t = x_t - \widehat{x}_t \tag{3}$$

There exists an integer m and the weight $\left[W^*, V^*\right]$ such that

$$\dot{\widehat{x}}_t = A\widehat{x}_t + W^*\Phi\left[V^*\hat{x}\right]$$

because

$$\dot{x} = Ax_t + G\left(x\right)$$

so

$$\dot{\Delta}_t = A\Delta_t + G\left(x\right) - W^*\Phi\left[V^*\widehat{x}\right] \tag{4}$$

Proof

Since $x_0 = \widehat{x}_0$, $\Delta_0 = 0$. The solution of the differential equation

$$\dot{\Delta}_t = A\Delta_t + G_1$$

where $G_1 = G(x) - W^*\Phi\left[V^*\widehat{x}\right]$

$$\begin{aligned} \Delta_t &= \int_0^t e^{A(t-\tau)} G_1 \, d\tau \\ &= \int_0^t e^{A(t-\tau)} \left[W^*\Phi\left[V^*x\right] - W^*\Phi\left[V^*\hat{x}\right]\right] d\tau \\ &+ \int_0^t e^{A(t-\tau)} \left[G(x) - W^*\Phi\left[V^*x\right]\right] d\tau \end{aligned} \tag{5}$$

Proof

Since A is a stability matrix, there exists a positive constant α such that

$$\left\| e^{At} \right\| \leq e^{-\alpha t}, \quad \alpha > 0$$

So

$$\begin{aligned} |\Delta_t| \leq & \int_0^t \left\| e^{A(t-\tau)} \right\| \|W^*\| \left\{ \left| \Phi \left[V^* x \right] - \Phi \left[V^* \widehat{x} \right] \right| \right\} d\tau \\ & + \int_0^t \left\| e^{A(t-\tau)} \right\| \left\{ \left| G \left(x_t, u_t \right) - W^* \Phi \left[V^* x \right] \right| \right\} d\tau \end{aligned}$$

Using Lipschitz condition

$$\left| \Phi \left[V^* x \right] - \Phi \left[V^* \widehat{x} \right] \right| \leq l \left| x_\tau - \widehat{x}_\tau \right|$$

and Stone-Weierstrass theorem

$$\sup_{(\widehat{x}, u)} \left| G \left(x \right) - W^* \Phi \left[V^* x \right] \right| \leq \delta \tag{6}$$

Proof

So

$$\begin{aligned} |\Delta_t| &\leq \int_0^t e^{-\alpha(t-\tau)} L \left| x_\tau - \widehat{x}_\tau \right| d\tau + \int_0^t e^{-\alpha(t-\tau)} \delta d\tau \\ &= \int_0^t e^{-\alpha(t-\tau)} L \left| \Delta_t \right| d\tau + \frac{\delta}{\alpha} \left(1 - e^{-\alpha t}\right) \end{aligned}$$

where $L := I \left\| W^* \right\|$. Because $e^{-\alpha t} \leq 1$, for $\alpha t > 0$

$$|\Delta_t| \leq \frac{\delta}{\alpha} + \int_0^t e^{-\alpha(t-\tau)} L \left| \Delta_t \right| d\tau$$

Proof

For

$$|\Delta_t| \leq \frac{\delta}{\alpha} + \int_0^t L e^{-\alpha(t-\tau)} |\Delta_t| \, d\tau$$

We use Gronwall-Bellman inequality, if

$$u \leq u_0 + \int f(\tau) u(\tau) \, d\tau$$

then

$$u \leq u_0 e^{\int f(\tau) d\tau}$$

But

$$u_0 = L\frac{\delta}{\alpha} \int_0^t \left(1 - e^{-\alpha t}\right) e^{-\alpha(t-\tau)}$$

$$\begin{aligned} |\Delta_t| &\leq \tfrac{\delta}{\alpha} + L\tfrac{\delta}{\alpha} \int_0^t \left(1 - e^{-\alpha t}\right) e^{-\alpha(t-\tau)} e^{\int_\tau^t L e^{-\alpha(t-s)} ds} d\tau \\ &= \tfrac{\delta}{\alpha} + L\tfrac{\delta}{\alpha} \int_0^t \left(1 - e^{-\alpha t}\right) e^{-\alpha(t-\tau)} e^{\frac{L}{\alpha}\left(1 - e^{-\alpha(t-\tau)}\right)} d\tau \end{aligned}$$

Because $e^{-\alpha t} \leq 1$, for $\alpha t > 0$.

Proof

Using the inequality of

$$1 - e^{-x} \leq x, \quad x \geq 0$$

we have

$$\begin{aligned}
|\Delta_t| &\leq \frac{\delta}{\alpha} + L\frac{\delta}{\alpha}\int_0^t \left(1 - e^{-\alpha t}\right) e^{-\alpha(t-\tau)} e^{L(t-\tau)} d\tau \\
&= \frac{\delta}{\alpha} + L\frac{\delta}{\alpha}\int_0^t \left(1 - e^{-\alpha t}\right)\left(1 - e^{-\alpha \tau}\right) e^{(L-\alpha)(t-\tau)} d\tau \\
&= \frac{\delta}{\alpha} + \frac{(\alpha - L)e^{-\alpha t} - \alpha e^{-(\alpha-L)t} + L}{\alpha(\alpha - L)}\delta \\
&= \frac{\delta}{\alpha} + \left[\frac{1}{\alpha}e^{-\alpha t} - \frac{e^{-(\alpha-L)t}}{(\alpha - L)} + \frac{L}{\alpha(\alpha - L)}\right]\delta \\
&\leq \frac{\delta}{\alpha} + \left[\frac{1}{\alpha} - \frac{e^{(\alpha-L)t}}{(\alpha - L)} + \frac{L}{\alpha(\alpha - L)}\right]\delta \\
&= \frac{\delta}{\alpha} + \frac{\delta}{\alpha - L}\left[1 - e^{-(\alpha-L)t}\right] \\
&\leq \frac{\delta}{\alpha} + \frac{\delta}{\alpha - L} = \frac{(2\alpha - L)}{\alpha(\alpha - L)}\delta
\end{aligned}$$

Proof

If we define

$$\varepsilon := \frac{(2\alpha - L)}{\alpha\,(\alpha - L)}\delta$$

and select $\alpha \geq L$, so $\varepsilon > 0$, $|\Delta_t| = |x_t - \widehat{x}_t| \leq \varepsilon$. From function approximate theorem (2) we know δ can be any small. Because α and L are constants, so ε can also be any small.

Proof

If the initial condition of neural networks and dynamic system are different, $x_0 \neq \widehat{x}_0$, The solution of the differential equation (4) is

$$\Delta_t = \int_0^t e^{A(t-\tau)} \left[G\left(x_\tau, u_\tau\right) - W^* \Phi_{V^*}\left(\widehat{x}_\tau, u_\tau\right) \right] d\tau + \Delta_0 e^{At}$$

The additional term $\Delta_0 e^{At}$ will influence the finial result as

$$|\Delta_t| \leq \frac{(2\alpha - L)}{\alpha\left(\alpha - L\right)}\delta + \frac{1}{L} e^{-(\alpha - L)t} \left|\Delta_0\right|.$$

Because $\left(\alpha - L\right) t > 0$, I and Δ_0 are constants

$$\lim_{t \to \infty} \frac{1}{L} e^{-(\alpha - L)t} \left|\Delta_0\right| = 0$$

So the influence of initial condition of RMLP will decay exponentially to zero.

Learning in Neural Networks

There are three broad types of learning:

- Supervised Learning, learning with a teacher)
  - 1. Classification
    2. Control
    3. Function approximation
    4. Associative memory
- Unsupervised learning, learning without help)
  1. Clustering
- Reinforcement learning, learning with limited feedback

Learning in biological systems as optimization

- Learning = learning by adaptation
- The young animal learns that the green fruits are sour, while the yellowish/reddish ones are sweet. The learning happens by adapting the fruit picking behavior. The animal likes to eat many energy rich, juicy fruits that make its stomach full, and makes it feel happy.
- At the neural level the learning happens by changing of the synaptic strengths, eliminating some synapses, and building new ones

Learning principle for artificial neural networks

- Maintaining synaptic strength needs energy, it should be maintained at those places where it is needed, and it shouldn't be maintained at places where it's not needed
- ENERGY MINIMIZATION
- We need an appropriate definition of energy for artificial neural networks, and having that we can use mathematical optimization techniques to find how to change the weights of the synaptic connections between neurons.
- ENERGY = measure of task performance error

Unsupervised Learning

- ANN adapts weights to cluster input data
- Hebbian learning
  - Connection stimulus-response strengthened (Hebbian)
- Competitive learning algorithms
  - Kohonen & ART
  - Input weights adjusted to resemble stimulus

Shifting around Learning

- we can think of learning as the process of shifting around the optimal points until each training data is classified correctly.
- we need to formalize that process of "shifting around" into a systematic algorithm that can easily be implemented on a computer.

$$w(k+1) = w(k) + \Delta w(k)$$

The "shifting around" can conveniently be split up into a number of small steps.

Hebbian learning

Hebbian learning (Unsupervised learning)

$$w(k+1) = w(k) + \eta \hat{y}(k) u(k)$$

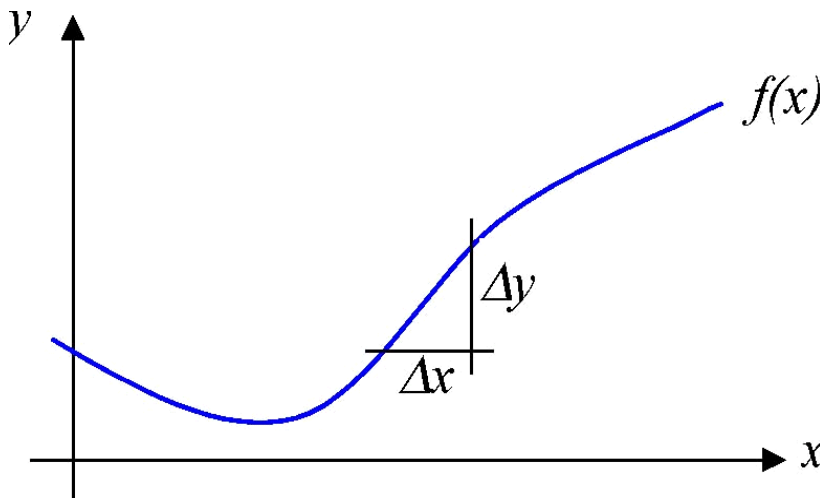
Perceptrons learning (supervised learning)

$$w(k+1) = w(k) + \eta [y(k) - \hat{y}(k)] u(k)$$

There is some similarity, but it is clear that Hebbian learning is not going to get our Perceptron to learn a set of training data. The are variations of Hebbian learning that do provide powerful learning techniques for biologically plausible networks, such as Contrastive Hebbian Learning, but we shall adopt another approach for formulating.

Delta Rule

We want to change the value of x to minimise $f(x)$



Delta Rule

The Delta rule is presented by Widrow and Hoff in 1960 as LMS (least mean square) learning procedure.
NN (Widrow's Adaline) is

$$\hat{y} = \sum_i w_i u_i + b$$

We eant to adjust the weights to minimize the difference between the network output and the desired output. The error function

$$E(w_i) = \frac{1}{2} \sum_{k=1}^{N} [y(k) - \hat{y}(k)]^2$$

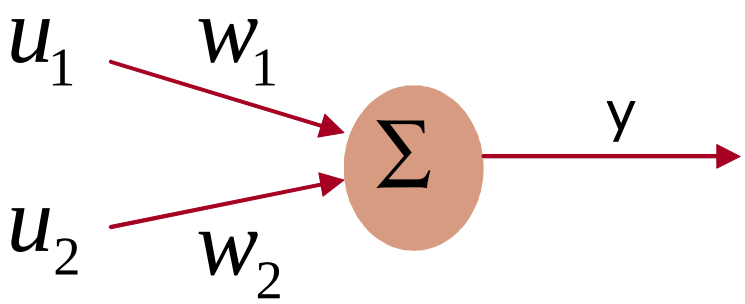
The idea is to make a change in the weight proportional to the negative of the derivative of the error as measured on the current data with respect to each weight: gradient descent.

Delta Rule

The weights are cahnged as

$$w_{new} = w_{old} - \eta \frac{\partial E}{\partial w}$$
$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_i} = -\left[y\left(k\right) - \hat{y}\left(k\right)\right] u_i$$
$$w\left(k+1\right) = w\left(k\right) + \eta e\left(k\right) u_i$$

where $e\left(k\right) = y\left(k\right) - \hat{y}\left(k\right)$



$$w_1\left(k+1\right) = w_1\left(k\right) + \eta\left[y\left(k\right) - \hat{y}\left(k\right)\right] u_1$$
$$w_2\left(k+1\right) = w_2\left(k\right) + \eta\left[y\left(k\right) - \hat{y}\left(k\right)\right] u_2$$