Intelligent Control

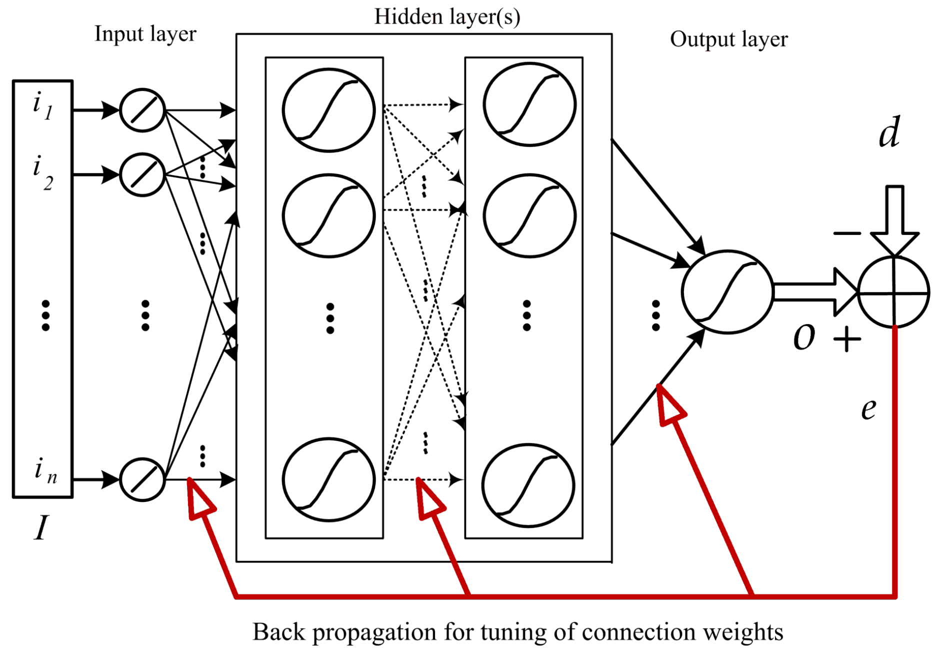
Multilayer Perceptron and Backpropagation Learning

Hassan Bevrani
Professor, University of Kurdistan

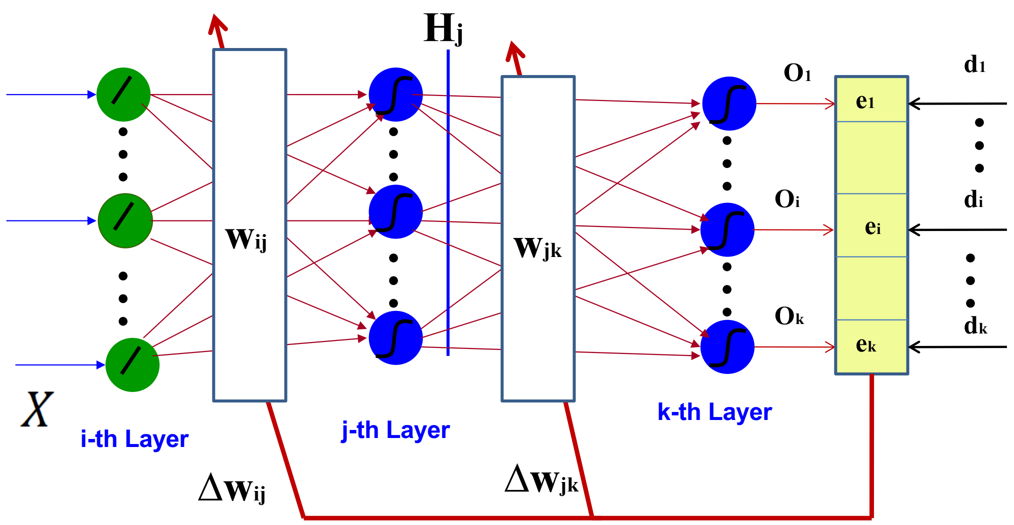
Fall 2023

Contents

MLP with BP learning



MLP with BP learning



BP Learning Mechanism

–Let the squared error is:

$$E = \frac{1}{2}(d_k - o_k)^2$$

–Modify weights between *j* and *k* layers by gradient descent approach

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}}$$

$$\Rightarrow \quad \frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial w_{jk}}$$

BP Learning Mechanism

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial o_k} = -(d_k - o_k)$$

$$\frac{\partial o_k}{\partial w_{jk}} = \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}} = f'(net_k) \frac{\partial net_k}{\partial w_{jk}}$$

$$net_k = w_{1k}H_1 + w_{2k}H_2 + \ldots + w_{jk}H_j \quad \Rightarrow \quad \frac{\partial net_k}{\partial w_{jk}} = H_j$$

$$\Rightarrow \quad \Delta w_{jk} = \eta(d_k - o_k).f'(net_k).H_j$$

$$\Delta w_{jk} = \eta.e_k.f'(net_k).H_j$$

BP Learning Mechanism

Assume $\delta_k = e_k f'(net_k)$

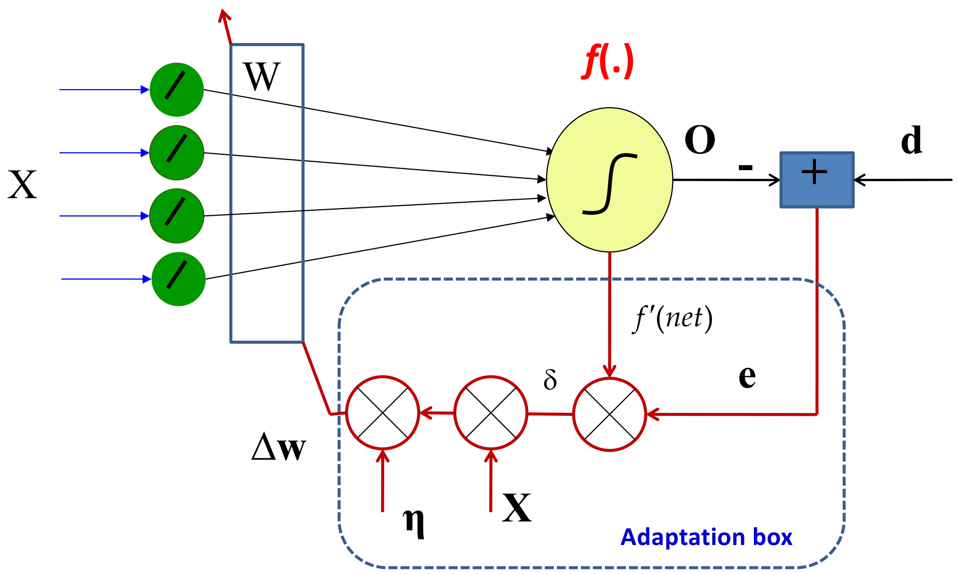
Therefore: $$\Delta w_{jk} = \eta \delta_k H_j$$

Class homework: **Show that**

$$f(net) = \frac{1}{1+e^{-\lambda . net}} \xrightarrow{\lambda = 1} f'(net_k) = o_k(1 - o_k)$$

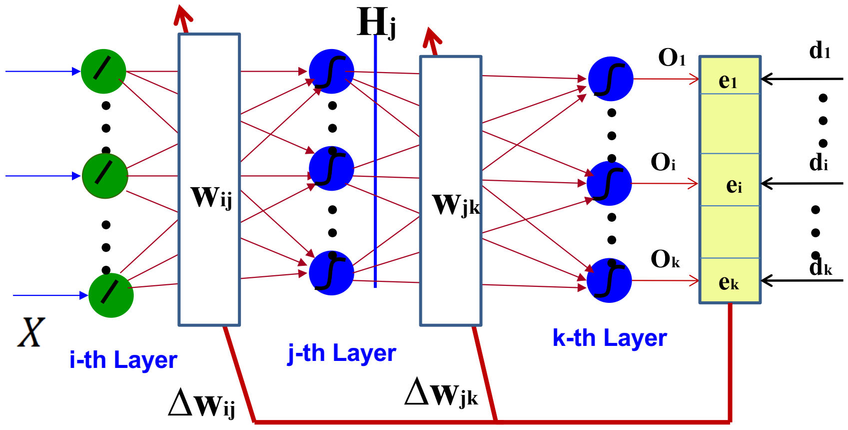
$$f(net) = \frac{1-e^{-\lambda . net}}{1+e^{-\lambda . net}} \xrightarrow{\lambda = 1} f'(net_k) = \frac{1}{2}(1 - o_k{}^2)$$

Example 1



BP Learning Mechanism

- Now, update weights between *i* and *j* layers by gradient descent approach $\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$



$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k}\frac{\partial net_k}{\partial H_j}\frac{\partial H_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}$$

BP Learning Mechanism

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k}\frac{\partial net_k}{\partial H_j}\frac{\partial H_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial o_k} = -(d_k - o_k) \qquad \frac{\partial o_k}{\partial net_k} = f'(net_k)$$

$$\frac{\partial net_k}{\partial H_j} = \frac{\partial (w_{k1}H_1 + w_{k2}H_2 + ... + w_{kj}H_j)}{\partial H_j} = w_{kj}$$

$$\frac{\partial H_j}{\partial net_j} = f'(net_j) = \begin{cases} H_j(1-H_j) \; ; \; f(.): USF \\ \frac{1}{2}(1-H_j^2) \; ; \; f(.): BSF \end{cases}$$

$$\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial (w_{1j}x_1 + w_{2j2}x_2 + ... + w_{ij}x_i + ... + w_{jj}x_j)}{\partial w_{ij}} = x_i$$

$$\Rightarrow \quad \Delta w_{ij} = -\eta[-(d_k - o_k)]f'(net_k)w_{kj}f'(net_j)x_i$$

BP Learning Mechanism

$$\Delta w_{ij} = -\eta[-(d_k - o_k)]f'(net_k)w_{kj}f'(net_j)x_i$$

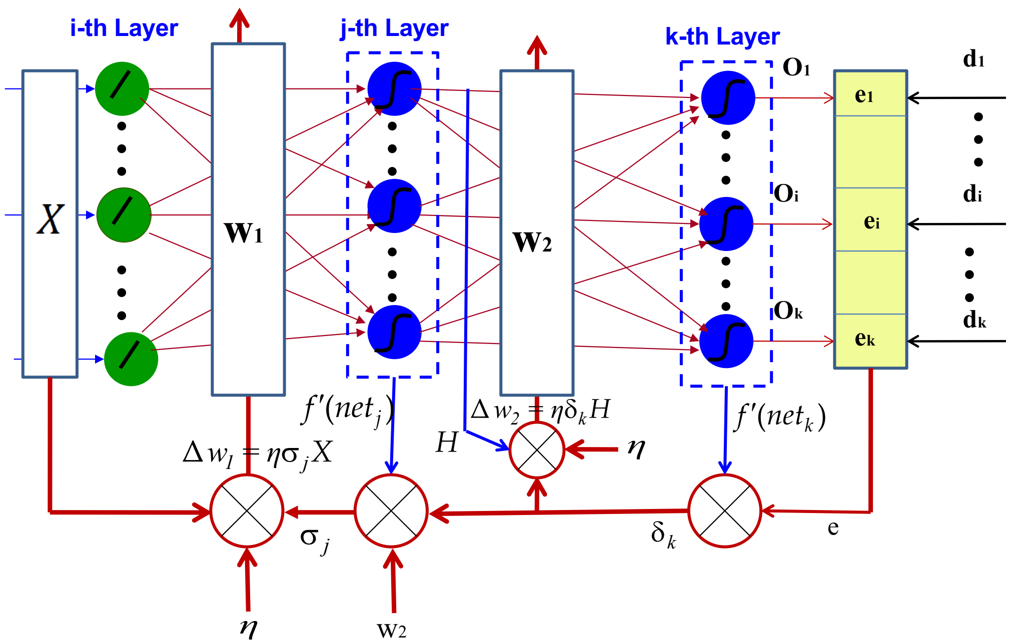
$$\delta_k = (d_k - o_k)]f'(net_k)$$

Assume $\sigma_j = \delta_k w_{kj} f'(net_j)$

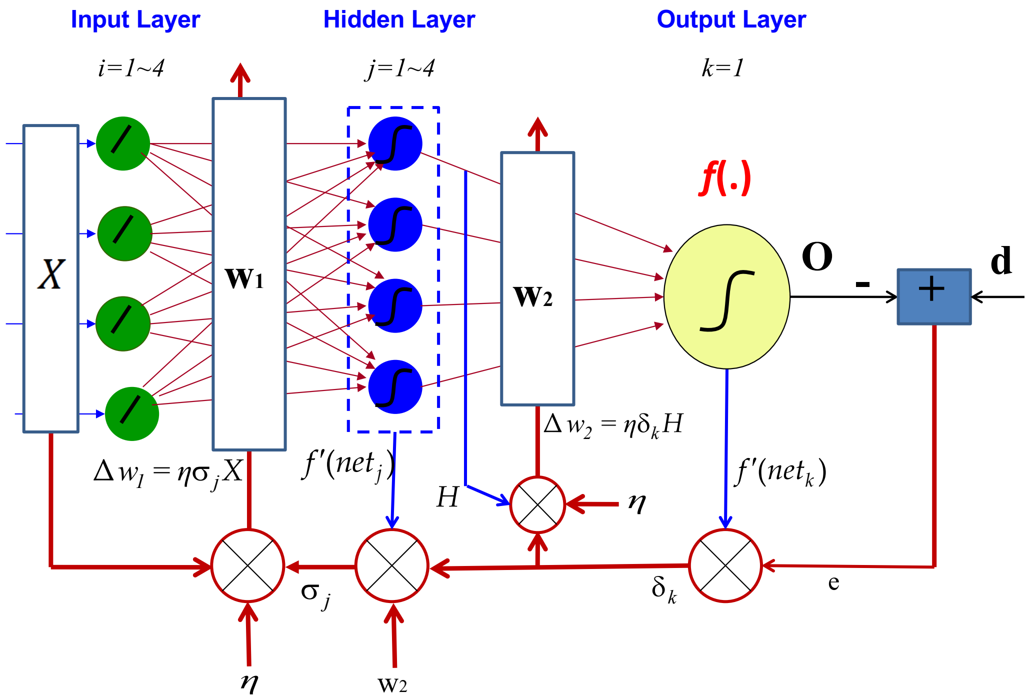
Therefore: $\Delta w_{ij} = \eta\sigma_j x_i$

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij} = w_{ij}(k) + \Delta w_{ij} + \eta\sigma_j x_i$$

BP learning



Example 2



Momentum Method

$$\Delta w_{jk} = \eta\delta_k H_j$$

$$w_{jk}(k+1) = w_{jk}(k) + \Delta w_{jk} = w_{jk}(k) + \eta\delta_k H_j$$

Choice of learning rate: $0<\eta<1$

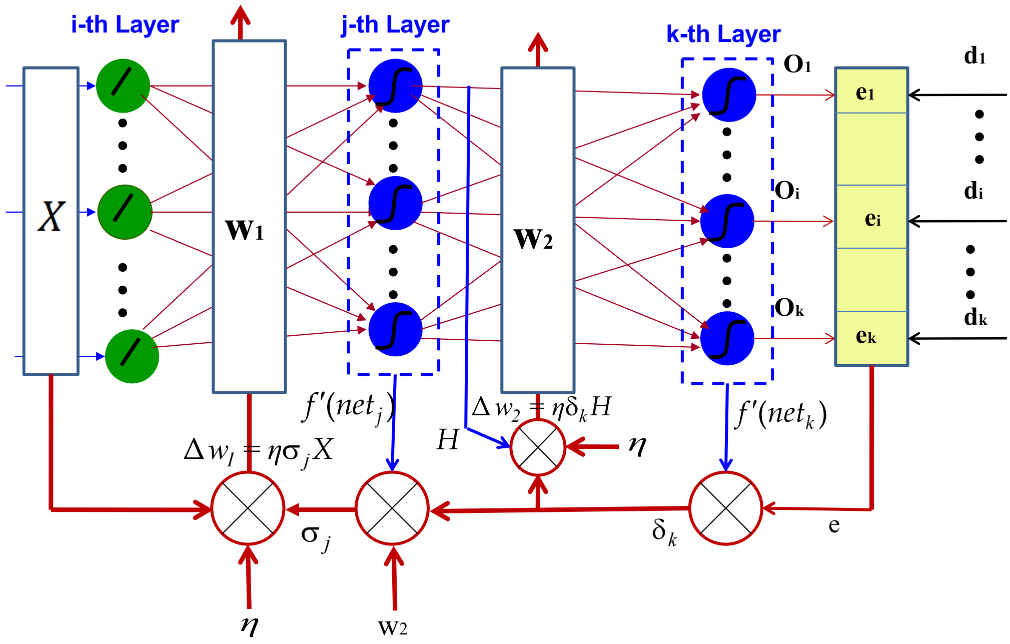
If $\eta \Rightarrow 1$:

1. Speed of learning increase
2. Degree of stability decrease

Solution: using Momentum term $0 \lesssim \alpha < 1$

$$\Delta w_{jk} = \eta\delta_k H_j + \alpha[w_{jk}(k+1) - w_{jk}(k)]$$

BP learning



BP training flowchart

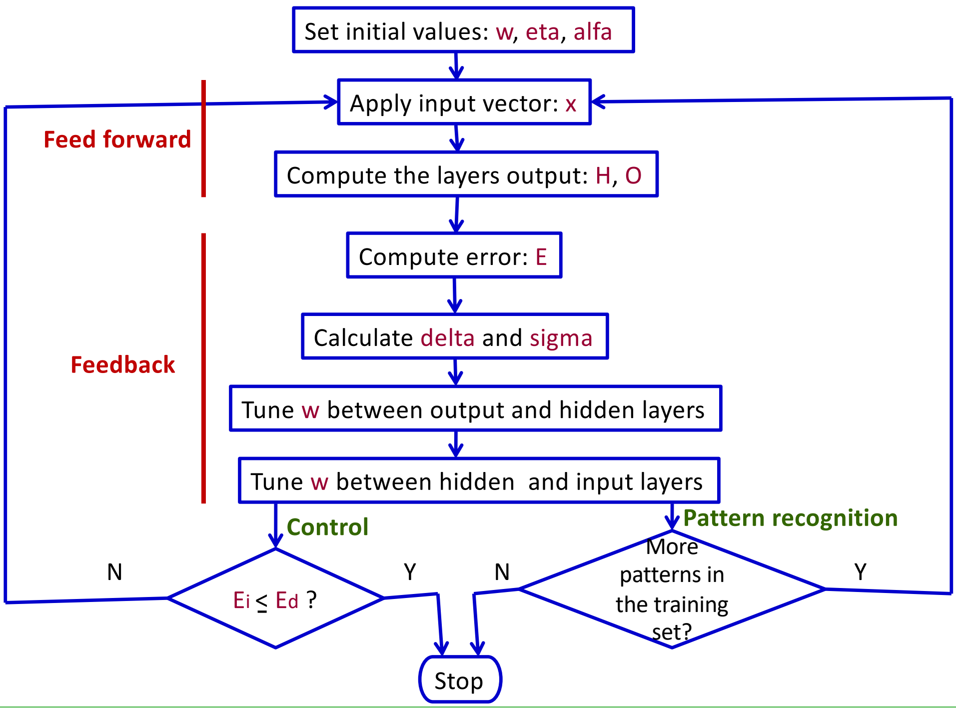
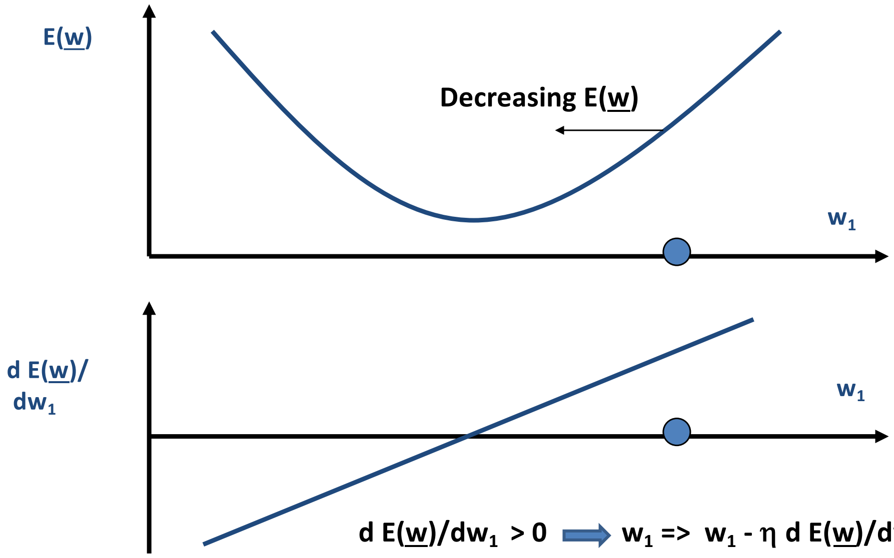


Illustration of Gradient Descent

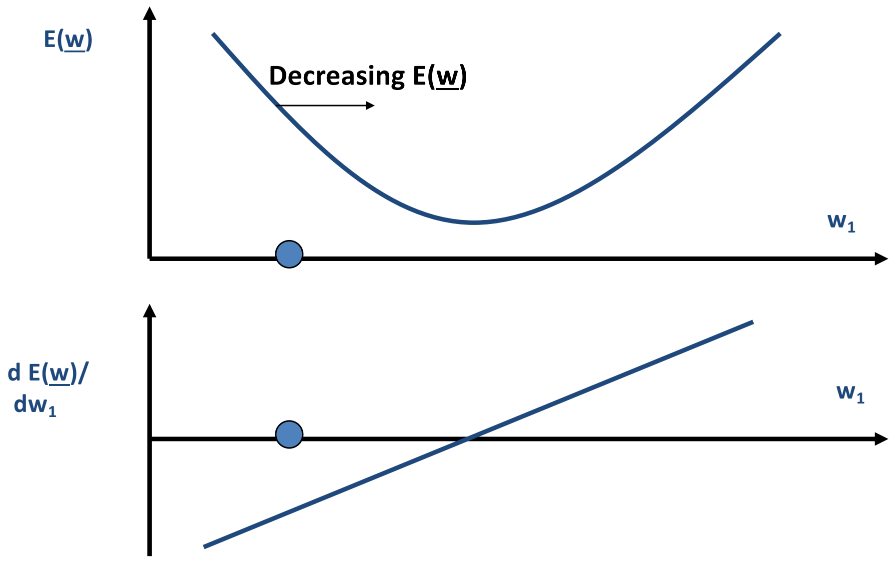
- Move in direction of negative derivative



$d\,E(\underline{w})/dw_1 > 0 \Longrightarrow w_1 \Rightarrow w_1 - \eta\, d\,E(\underline{w})/dw_1$

i.e., the rule decreases w_1

Illustration of Gradient Descent



$d\,E(\underline{w})/dw_1 < 0 \Longrightarrow w_1 \Leftarrow w_1 - \eta\, d\,E(\underline{w})/dw_1$

i.e., the rule increases w_1

Illustration of Gradient Descent

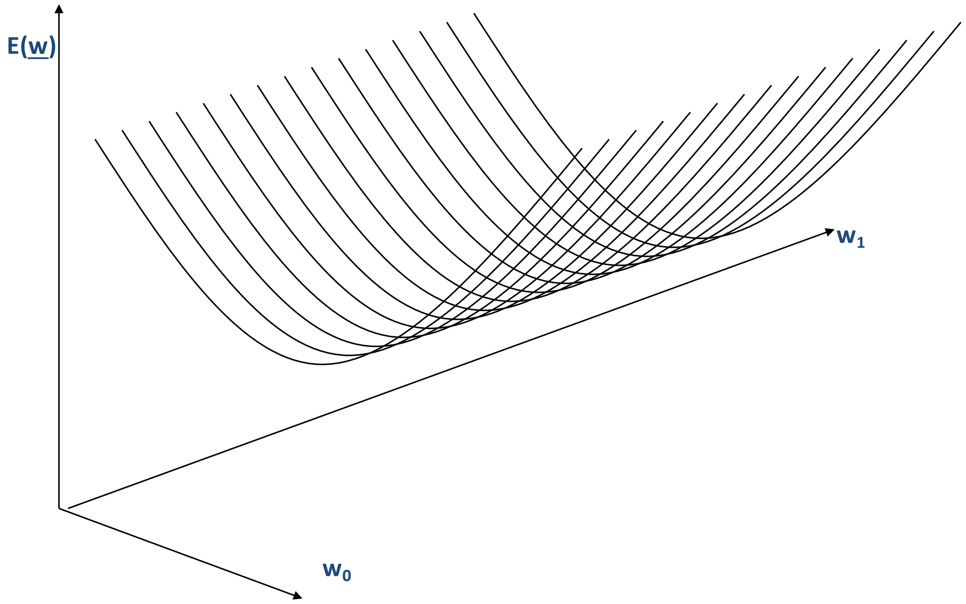


Illustration of Gradient Descent

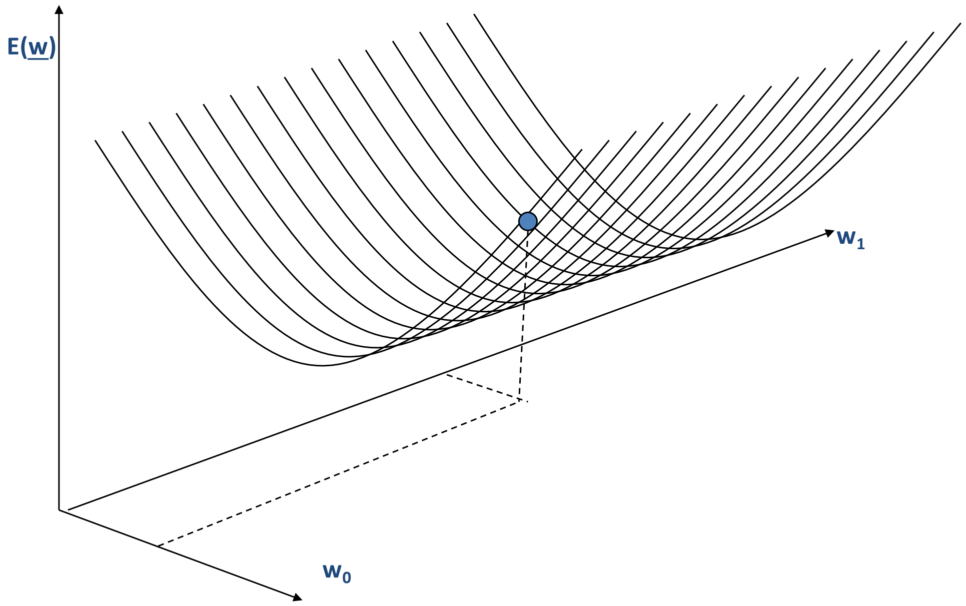


Illustration of Gradient Descent

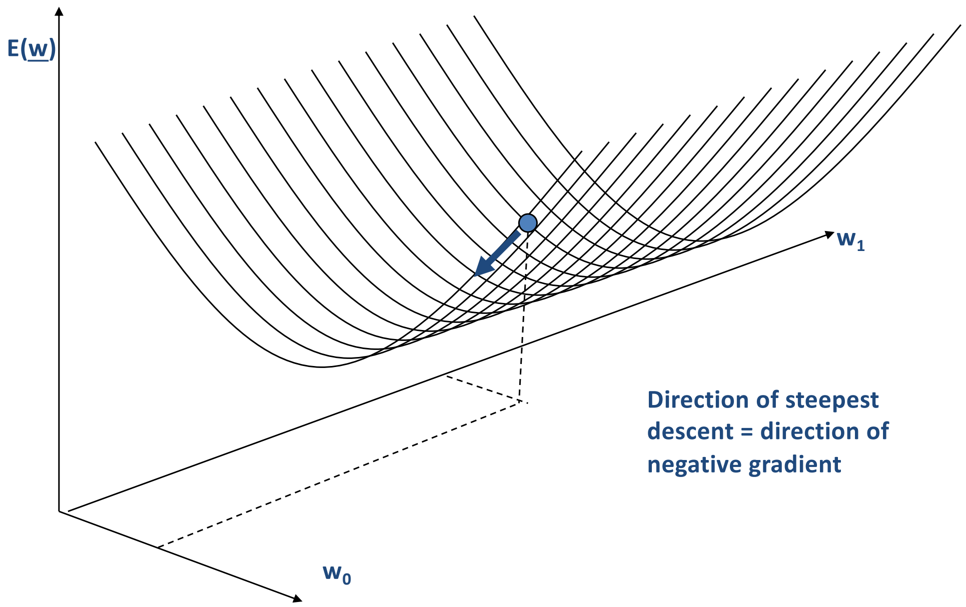
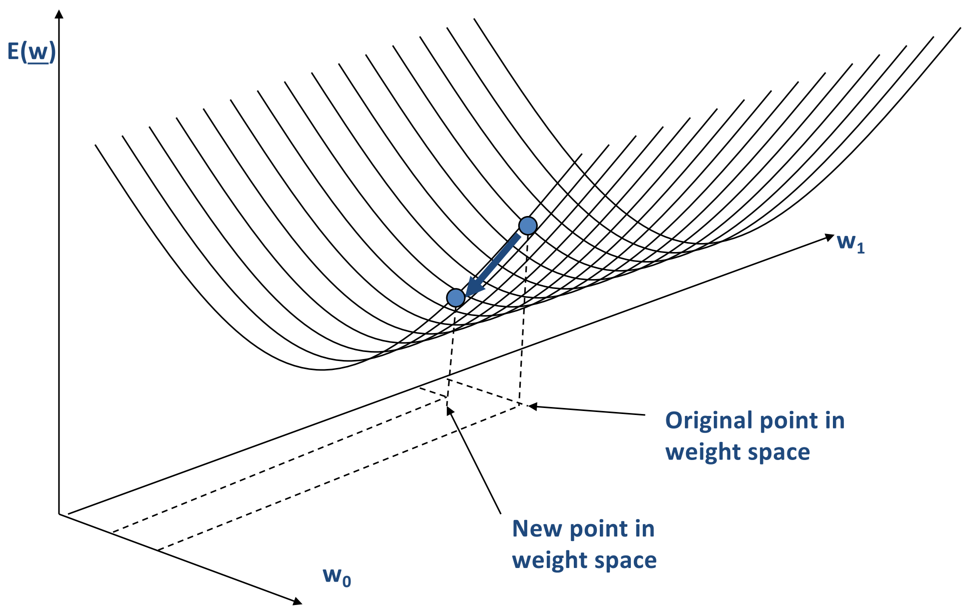
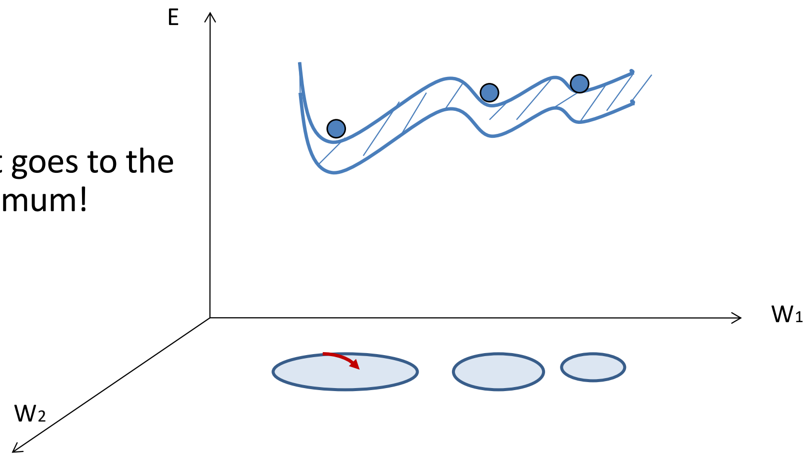


Illustration of Gradient Descent



Local minimum

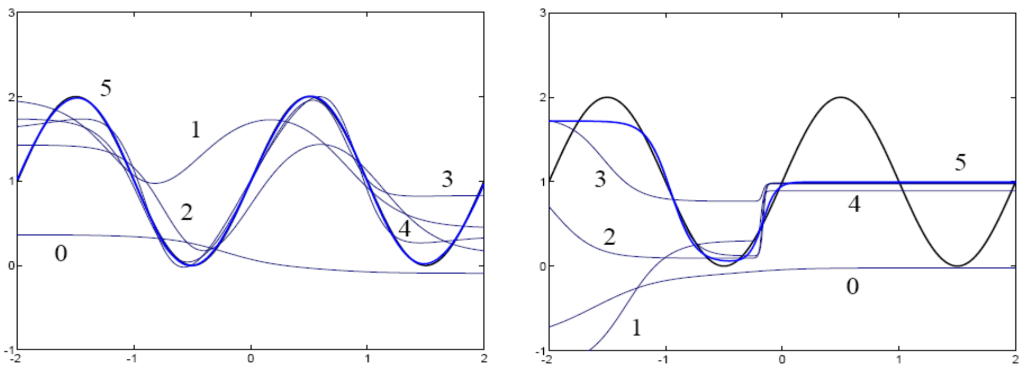
Gradient descent goes to the closest local minimum!



A Solution: random restarts from multiple places (initial weights) in weight space

Convergence to Local/Global minimums

$$g(p) = 1 + \sin(\pi p)$$

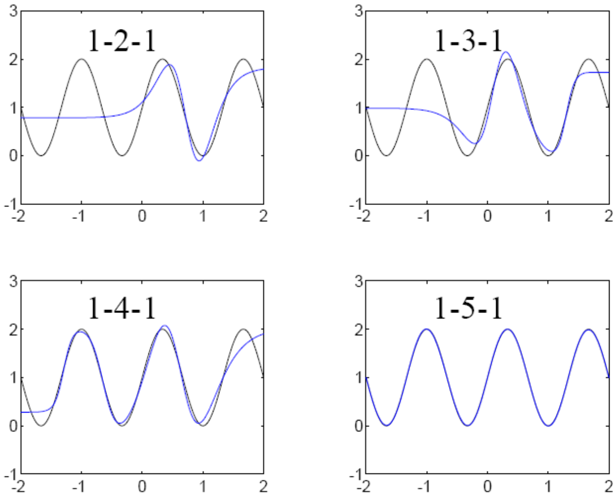


Global minimum **local minimum**

Impacts of ANN Architecture

$$g(p) = 1 + \sin\left(\frac{6\pi}{4}p\right)$$

An example for 3-layer ANN with different number of neurons in hidden layer



Heuristics to alleviate the problem of local minima

1. Add momentum

2. Use stochastic gradient descent rather than true gradient descent.

3. Train multiple nets with different initial weights using the same data (Proper selection of initial weights).

4. Fine tuning of learning rate parameters

Over training

- The major problem in training a NN is deciding **when to stop training**. Since the ability to generalize is fundamental for the networks which predict future, **overtraining** is a serious problem.

- Overtraining occurs when the system memorizes patterns and thus looses the ability to generalize (decreasing **generalization accuracy** over other unseen examples)

Techniques to overcome over training problem

- **Stopping criterion** (Termination condition): Until the error E falls below some predetermined threshold. This is a poor strategy

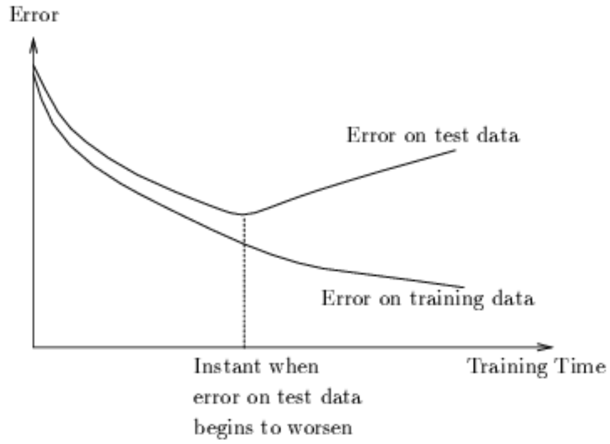
- **Weight decay** : Decrease each weight by some small factor during each iteration. The motivation for this approach is to keep weight values small.

- **Cross-validation**: a set of validation data in addition to the training data. The algorithm monitors the error for this validation data while using the training set to drive the gradient descent search.
  - **How many weight-tuning iterations should the algorithm perform?**
    It should use the number of iterations that produces the lowest error over the validation set.

Techniques to overcome over training problem

- **Generalization** is not guaranteed even if the error is reduced to 0.

  – Over-fitting/over-training problem: trained net fits the training samples perfectly (*E* reduced to 0) but it does not give accurate outputs for inputs not in the training set

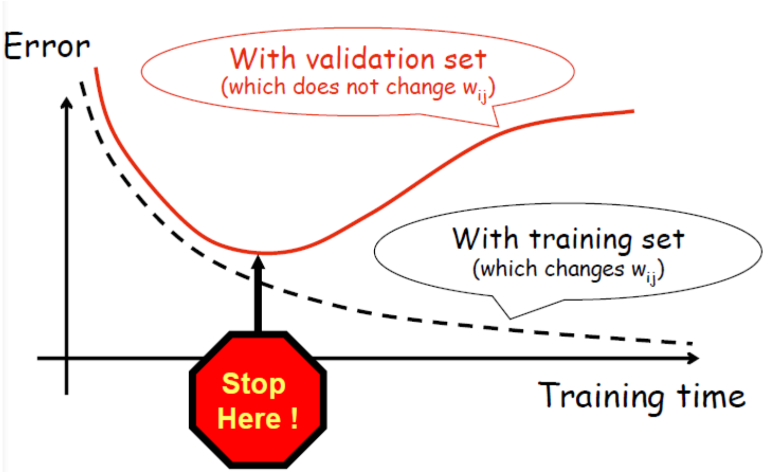


Cross-Validation

– leave some (~10%) samples as test data (not used for weight update)

– periodically check error on test data

– Learning stops when error on test data starts to increase



Strengths of BP Learning

- **Great representation power**

– Nonlinear function can be represented by a BP net

– Many such functions can be approximated by BP learning (gradient descent approach)

- **Wide applicability of BP learning**

– Only requires that a good set of training samples is available

– Does not require substantial prior knowledge or deep understanding of the domain itself

– Tolerates noise and missing data in training samples (graceful degrading)

Strengths of BP Learning

- **Easy to implement** the core of the learning algorithm

- **Good generalization power**

– Often produce accurate results for inputs outside the training set

Strengths of BP Learning

- **Learning often takes a long time to converge**

–Complex functions often need hundreds or thousands of epochs

- **The net is essentially a black box**

–It may provide a desired mapping between input and output vectors (***x, o***) but does not have the information of why a particular ***x*** is mapped to a particular ***o.***

Strengths of BP Learning

- **A way to assess the high quality of learning is unknown**

–There is no theoretically well-founded way to **assess the quality** of BP learning

○What is the confidence level one can have for a trained BP net, with the final E (which may or may not be close to zero)?

○What is the confidence level of ***o*** computed from input ***x*** using such net?

- **Problem with gradient descent approach**

–only guarantees to reduce the total error to a **local minimum**. (***E*** may not be reduced to zero)

–Cannot escape from the local minimum error state

Strengths of BP Learning

- **Sensitivity to initial conditions**
- **Instability if learning rate is too large**

Despite above disadvantages, it is popularly used in control community. There are numerous extensions to improve BP algorithm.

Practical Considerations

For a good BP, many parameters must be carefully selected to ensure a good performance.

- **Proper selection of Initial weights (and biases)**
  - Random, [-0.05, 0.05], [-0.1, 0.1], [-1, 1]
  - Normalize weights for hidden layer ($w^{(1,0)}$)
    - Random assign initial weights for all hidden nodes
    - For each hidden node j, normalize its weight by

$$w_{j,i}^{(1,0)} = \beta \cdot w_{j,i}^{(1,0)} / \left\| w_j^{(1,0)} \right\|_2 \quad \text{where } \beta = 0.7\sqrt[n]{m}$$

$$m = \#\text{ of hident nodes}, \; n = \#\text{ of input nodes}$$

    - Avoid bias in weight initialization

Practical Considerations

- **Training samples**:
  - Quality and quantity of training samples often determines the quality of learning results
  - Samples must collectively represent well the problem space
    - Random sampling
    - Proportional sampling (with prior knowledge of the problem space)
  - # of training patterns needed: There is no theoretically idea number.

- **Adding momentum term** (to speedup learning)
  - Avoid sudden change of directions of weight update (smoothing the learning process)

Practical Considerations

- **Number of hidden layers and hidden nodes per layer**
  - Theoretically, one hidden layer (possibly with many hidden nodes) is sufficient for any nonlinear functions
  - There is no theoretical results on minimum necessary # of hidden nodes
  - Practical rule :
    - n = # of input nodes; m = # of hidden nodes
    - For binary/bipolar data: m = 2n
    - For real data: m >> 2n
  - Multiple hidden layers with fewer nodes may be trained faster for similar quality in some applications

Practical Considerations

- **Proper tuning of learning rate η**

– Fixed rate much smaller than 1

– Start with large η, gradually decrease its value

– Start with a small η, steadily double it until MSE start to increase

– Find the maximum safe step size at each stage of learning (to avoid overshoot the minimum E when increasing η)

Dynamic NN: Recurrent NN

- **Feed forward networks:**
  - Information only flows one way
  - One input pattern produces one output
  - No sense of time (or memory of previous state)

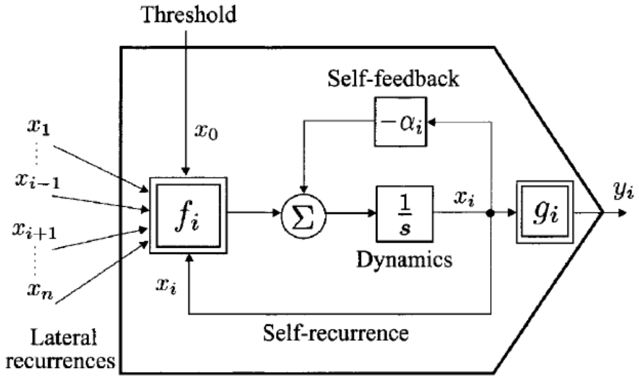
- **Recurrency**
  - Nodes connect back to other nodes or themselves
  - Information flow is multidirectional
  - Sense of time and memory of previous state(s)

Biological nervous systems show high levels of recurrency

Dynamic neural unit

The NN that have been discussed so far contain no time-delay elements or integrators. Such NN are called *non-dynamic* as they do not have any memory (Recurrent or Dynamic neural network (DNN): NN with memory)

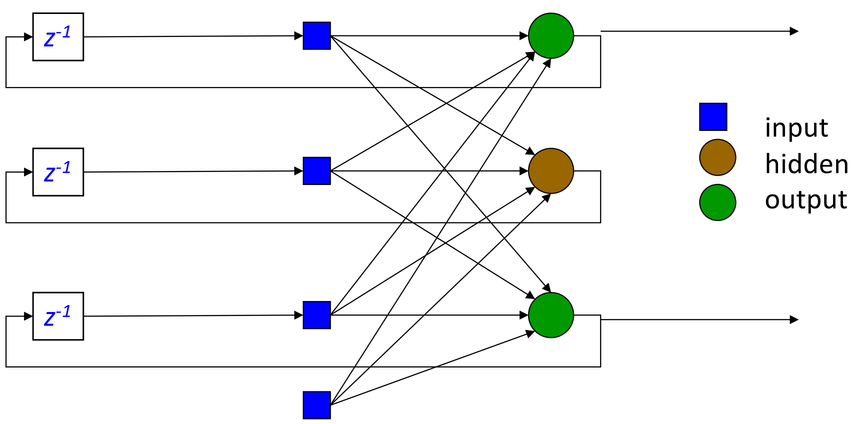
A model:



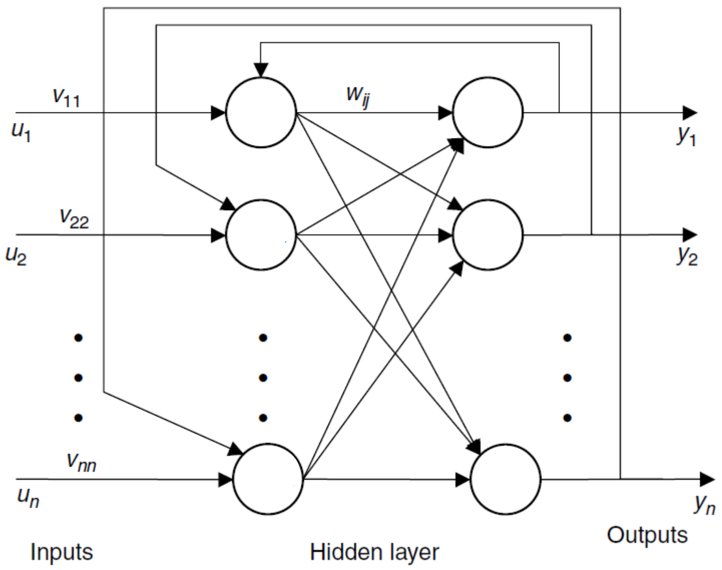
$$\frac{dx_i(t)}{dt} = -\alpha_i x_i(t) + f_i(\boldsymbol{w}_{ai}, \boldsymbol{x}_a)$$
$$y_i(t) = g_i(x_i(t))$$

Recurrent NN

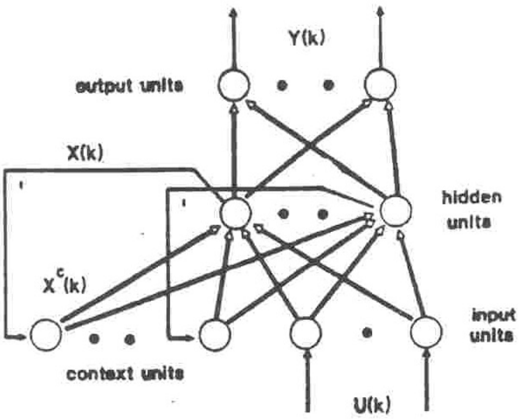
Recurrent Network with ***hidden neuron***: unit delay operator z^{-1} is used to model a dynamic system

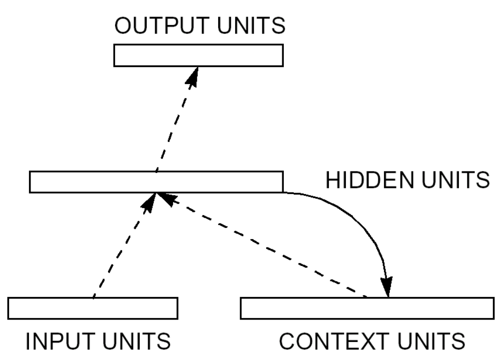


Hopfield ANN



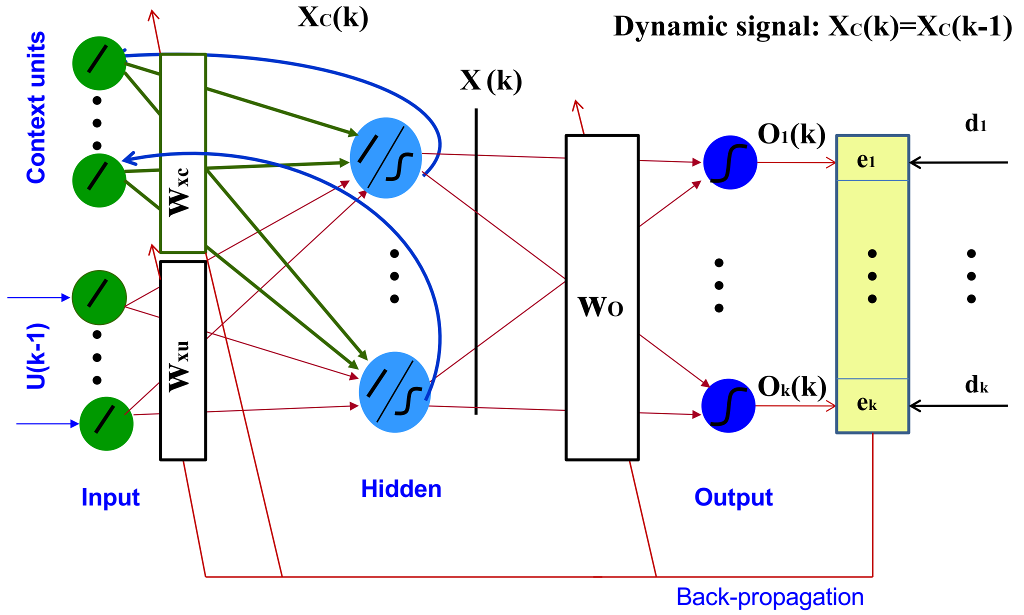
The Elman network



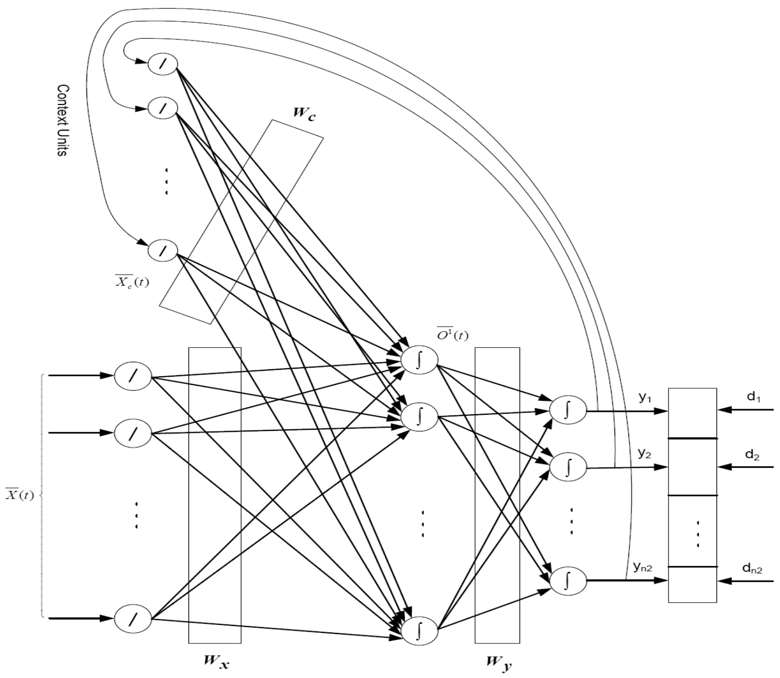


The Elman network

Partial recurrent network with dynamic BP



The Jordan network

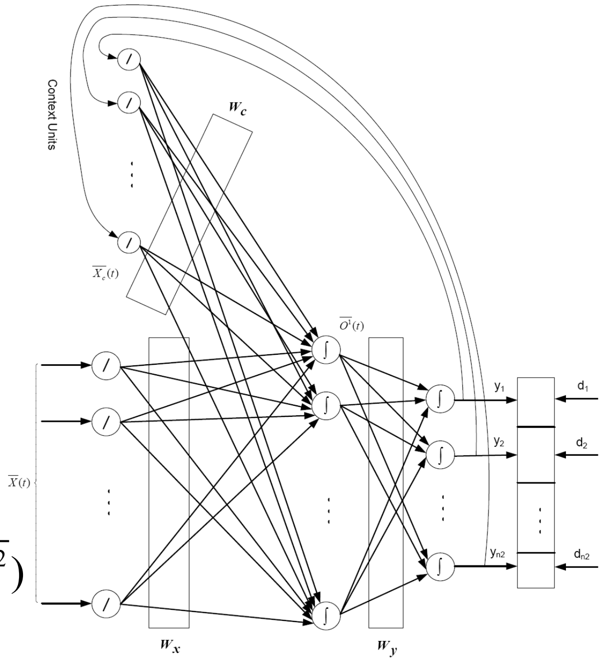


Jordan Learning

$$\begin{cases}\overline{net^1} = \boldsymbol{W_x}.\overline{X}(t) + \boldsymbol{W_c}.\overline{X_c}(t)\\ where:\\ \overline{X_c}(t) = \overline{O^2}(t-1) = \overline{y}(t-1)\end{cases}$$

$$\overline{F^1}(\overline{net^1}) = \overline{O^1}(\overline{net^1})$$

$$\begin{cases}\overline{net^2} = \boldsymbol{W_y}.\overline{O^1}(t)\\ \overline{F^2}(\overline{net^2}) = \overline{O^2}(\overline{net^2}) = \overline{y}(\overline{net^2})\end{cases}$$

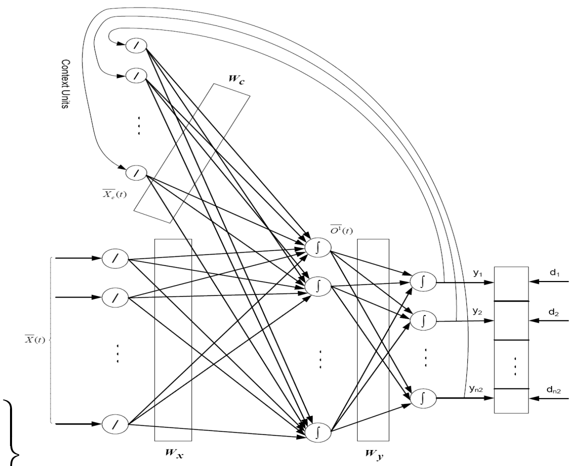


Jordan Learning

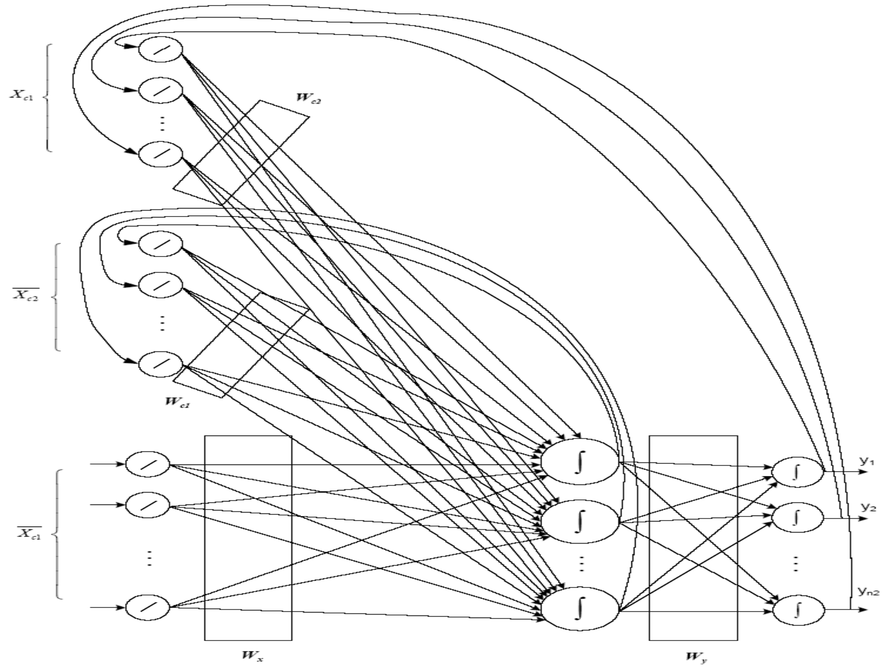
$$\begin{cases}\Delta\boldsymbol{W_x} = -\eta\dfrac{\partial E}{\partial \boldsymbol{W_x}}\\ \Delta\boldsymbol{W_y} = -\eta\dfrac{\partial E}{\partial \boldsymbol{W_y}}\\ \Delta\boldsymbol{W_c} = -\eta\dfrac{\partial E}{\partial \boldsymbol{W_c}} = -\eta.\dfrac{\partial E}{\partial \overline{O^2}}.\dfrac{\partial \overline{O^2}}{\partial \overline{net^2}}.\dfrac{\partial \overline{net^2}}{\partial \overline{O^1}}.\dfrac{\partial \overline{O^1}}{\partial \overline{net^1}}.\dfrac{\partial \overline{net^1}}{\partial \boldsymbol{W_c}}\end{cases}$$

$$= \eta.\underbrace{\overline{e}.\overline{F^{2\prime}}.\boldsymbol{W_y}.\overline{F^{1\prime}}}_{\overline{\delta^1}}.\left\{\overline{X_c}(t) + \boldsymbol{W_c}.\frac{\partial \overline{X_c}(t)}{\partial \boldsymbol{W_c}}\right\}$$

$$\Delta\boldsymbol{W} = \eta.\overline{\delta^1}.\left\{\overline{X_c}(t) + \boldsymbol{W_c}.\frac{\partial \overline{X_c}(t)}{\partial \boldsymbol{W_c}}\right\}$$



Elman and Jordan

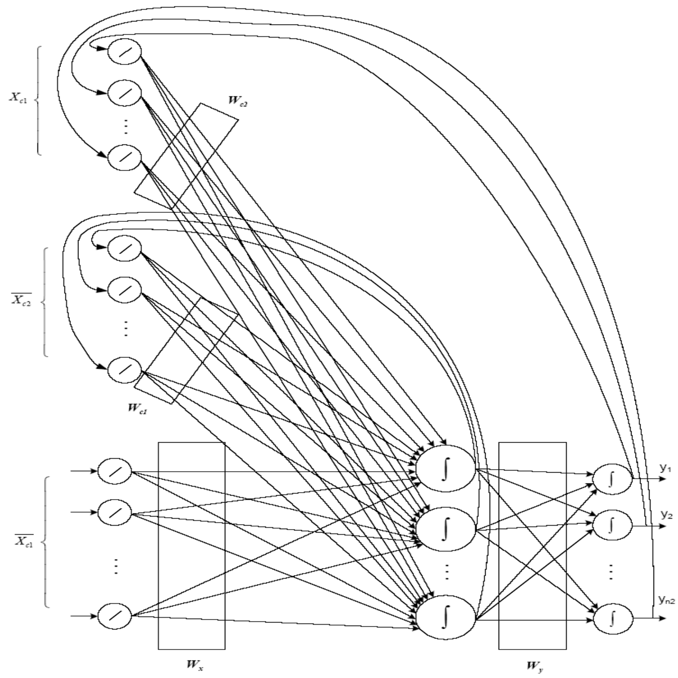


Elman and Jordan Learning

$$\begin{cases} \overline{net^1} = \boldsymbol{W_x}.\overline{X}(t) + \boldsymbol{W_{c1}}.\overline{X_{c1}}(t) + \boldsymbol{W_{c2}}.\overline{X_{c2}}(t) \\ where: \\ \overline{X_{c1}}(t) = \overline{O^1}(t-1) = \overline{y}(t-1) \\ \overline{X_{c2}}(t) = \overline{O^2}(t-1) = \overline{y}(t-1) \end{cases}$$

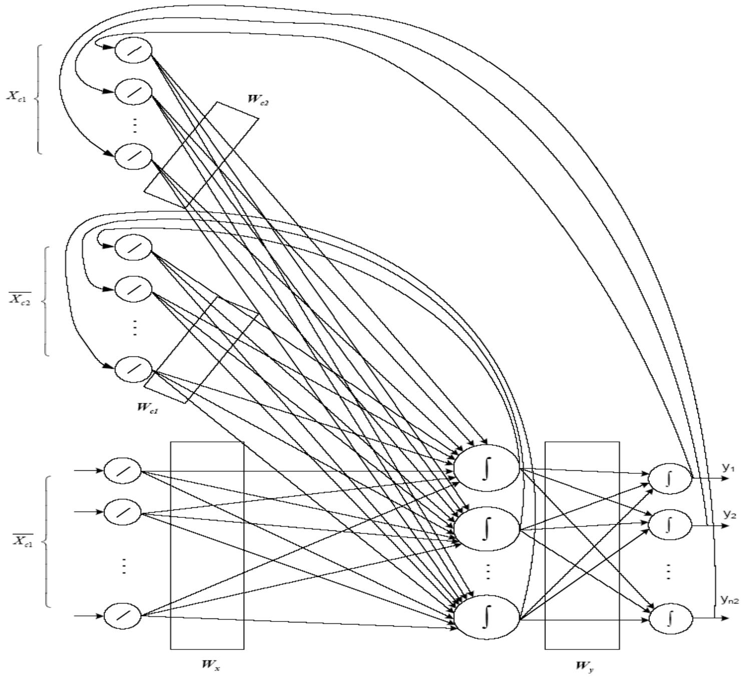
$$\overline{F^1}(\overline{net^1}) = \overline{O^1}(\overline{net^1})$$

$$\begin{cases} \overline{net^2} = \boldsymbol{W_y}.\overline{O^1}(t) \\ \overline{F^2}(\overline{net^2}) = \overline{O^2}(\overline{net^2}) = \overline{y}(\overline{net^2}) \end{cases}$$

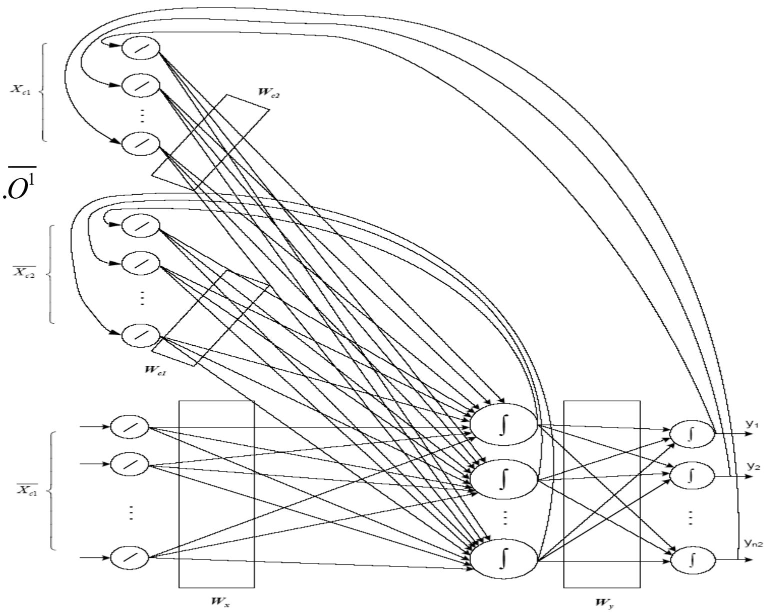


Elman and Jordan Learning

$$\begin{cases} \Delta \boldsymbol{W}_y = -\eta \dfrac{\partial E}{\partial \boldsymbol{W}_y} & (1) \\ \Delta \boldsymbol{W}_x = -\eta \dfrac{\partial E}{\partial \boldsymbol{W}_x} & (2) \\ \Delta \boldsymbol{W}_{c1} = -\eta \dfrac{\partial E}{\partial \boldsymbol{W}_{c1}} & (3) \\ \Delta \boldsymbol{W}_{c2} = -\eta \dfrac{\partial E}{\partial \boldsymbol{W}_{c2}} & (4) \end{cases}$$



Elman and Jordan Learning



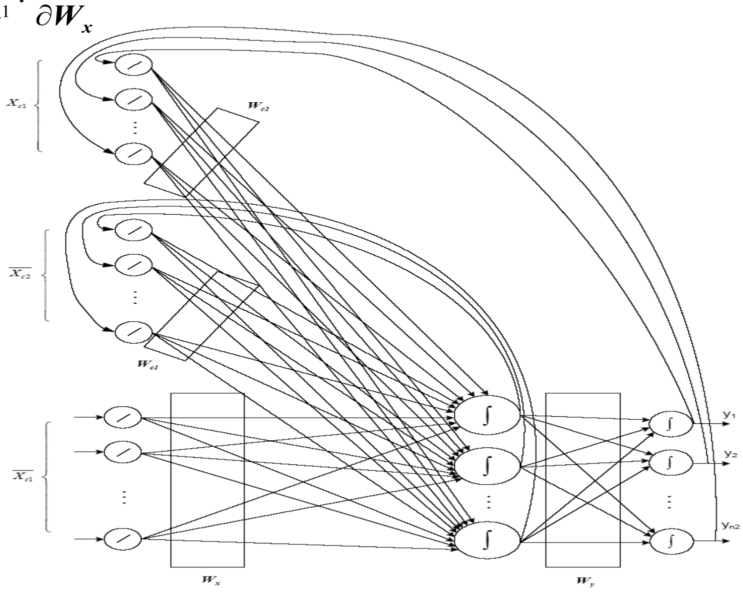
1) $$\Delta \boldsymbol{W}_y = -\eta \frac{\partial E}{\partial \boldsymbol{W}_y} = -\eta . \frac{\partial E}{\partial \overline{O^2}} . \frac{\partial \overline{O^2}}{\partial \overline{net^2}} . \frac{\partial \overline{net^2}}{\partial \boldsymbol{W}_y} = \eta . \underbrace{\bar{e} . \overline{F^{2\prime}}(.)}_{\delta^2} . \overline{O^1}$$

Elman and Jordan Learning

$$2)\quad \Delta \boldsymbol{W}_{\boldsymbol{x}} = -\eta \frac{\partial E}{\partial \boldsymbol{W}_{\boldsymbol{x}}} = -\eta.\frac{\partial E}{\partial \overline{O^2}}.\frac{\partial \overline{O^2}}{\partial \overline{net^2}}.\frac{\partial \overline{net^2}}{\partial \overline{O^1}}.\frac{\partial \overline{O^1}}{\partial \overline{net^1}}.\frac{\partial \overline{net^1}}{\partial \boldsymbol{W}_{\boldsymbol{x}}}$$

$$= \eta.\underbrace{\overline{e}.\overline{F^{2\prime}}.\boldsymbol{W}_{\boldsymbol{y}}.\overline{F^{1\prime}}}_{\overline{\delta^1}}.\overline{X}(t)$$

$$\Delta \boldsymbol{W}_{\boldsymbol{x}} = \eta.\overline{\delta^1}.\overline{X}(t)$$



Elman and Jordan Learning

$$3)\quad \Delta \boldsymbol{W}_{\boldsymbol{c1}} = -\eta \frac{\partial E}{\partial \boldsymbol{W}_{\boldsymbol{c1}}} = -\eta.\frac{\partial E}{\partial \overline{O^2}}.\frac{\partial \overline{O^2}}{\partial \overline{net^2}}.\frac{\partial \overline{net^2}}{\partial \overline{O^1}}.\frac{\partial \overline{O^1}}{\partial \overline{net^1}}.\frac{\partial \overline{net^1}}{\partial \boldsymbol{W}_{\boldsymbol{c1}}}$$

$$= \eta.\underbrace{\overline{e}.\overline{F^{2\prime}}(.).\boldsymbol{W}_{\boldsymbol{y}}.\overline{F^{1\prime}}(.)}_{\overline{\delta^1}}.\left\{\overline{X_{c1}}(t) + \boldsymbol{W}_{\boldsymbol{c1}}.\frac{\partial \overline{X_{c1}}(t)}{\partial \boldsymbol{W}_{\boldsymbol{c1}}}\right\}$$

$$\Delta \boldsymbol{W}_{\boldsymbol{c1}} = \eta.\overline{\delta^1}.\left\{\overline{X_{c1}}(t) + \boldsymbol{W}_{\boldsymbol{c1}}.\frac{\partial \overline{X_{c1}}(t)}{\partial \boldsymbol{W}_{\boldsymbol{c1}}}\right\}$$

$$= \eta.\overline{\delta^1}.\left\{\overline{O^1}(t-1) + \boldsymbol{W}_{\boldsymbol{c1}}.\frac{\partial \overline{O^1}(t-1)}{\partial \boldsymbol{W}_{\boldsymbol{c1}}}\right\}$$

Elman and Jordan Learning

$$4)\quad \Delta \boldsymbol{W}_{c2} = -\eta \frac{\partial E}{\partial \boldsymbol{W}_{c2}} = -\eta . \frac{\partial E}{\partial \overline{O^2}} . \frac{\partial \overline{O^2}}{\partial \overline{net^2}} . \frac{\partial \overline{net^2}}{\partial \overline{O^1}} . \frac{\partial \overline{O^1}}{\partial \overline{net^1}} . \frac{\partial \overline{net^1}}{\partial \boldsymbol{W}_{c2}}$$

$$= \eta . \underbrace{\bar{e} . \overline{F^{2\prime}}(.) . \boldsymbol{W}_y . \overline{F^{1\prime}}(.)}_{\overline{\delta^1}} . \left\{ \overline{X_{c2}}(t) + \boldsymbol{W}_{c2} . \frac{\partial \overline{X_{c2}}(t)}{\partial \boldsymbol{W}_{c2}} \right\}$$

$$\Delta \boldsymbol{W}_{c2} = \eta . \overline{\delta^1} . \left\{ \overline{X_{c2}}(t) + \boldsymbol{W}_{c2} . \frac{\partial \overline{X_{c2}}(t)}{\partial \boldsymbol{W}_{c2}} \right\}$$

$$= \eta . \overline{\delta^1} . \left\{ \overline{O^2}(t-1) + \boldsymbol{W}_{c2} . \frac{\partial \overline{O^2}(t-1)}{\partial \boldsymbol{W}_{c2}} \right\}$$

Thank you!