CISC452/CMPE452/COGS 400
Adaline

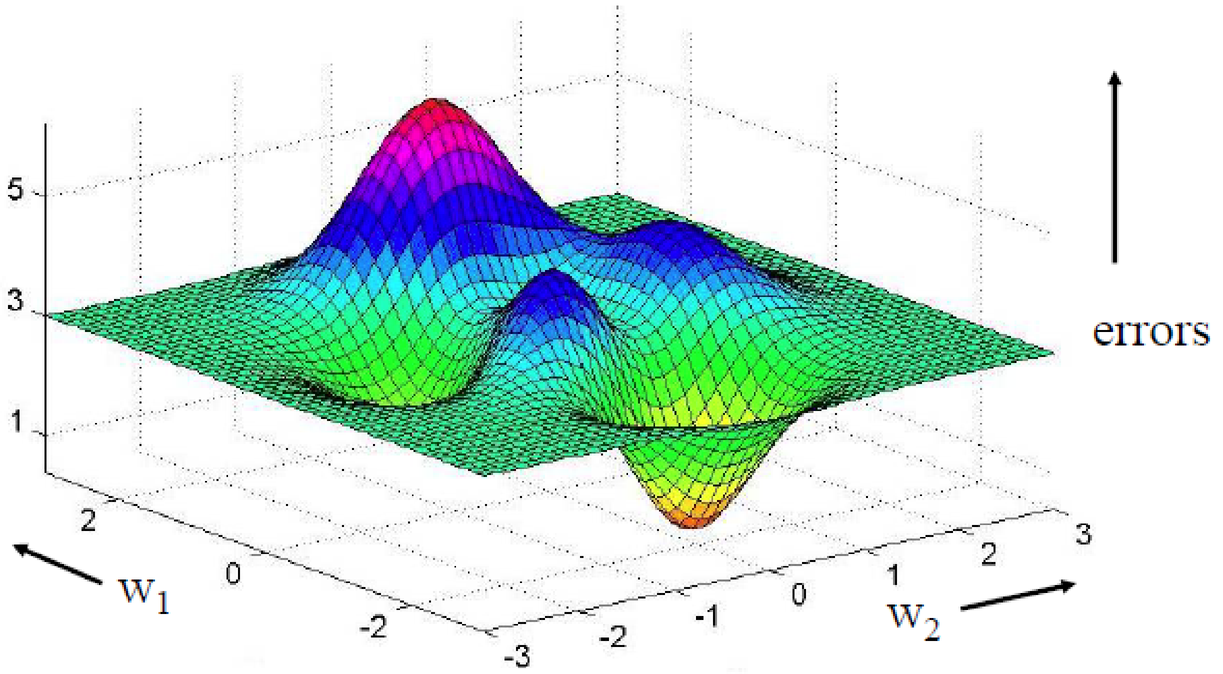
Farhana Zulkernine

Non-Linear Problems

- Perceptrons are good for data that have linear separability.
- Can they be used for problems where data are NOT linearly separable?
  - We saw two learning techniques in Perceptron.
  1. Simple Feedback Learning – Reduce the # of points that are wrongly classified.
  2. Error Correction Learning – Reduce the amount of total error represented as *Mean Squared Error*.

Error Measures

- One error value is associated with each combination of weight values. Of course $w_i \in \mathbb{R}$, and w may have many components, so we can't try all possibilities to find the minimum.
- For this, we introduce the idea of an error surface.



Mean Squared Error

- Error $e = d - y$
- For *n* iterations the mean squared error will be:

  $MSE = 1/n * (d - y)^2$
- Since MSE is a quadratic function, its derivative exists everywhere.
- If $y = f(\text{net}) = a$, a linear output function then

  $MSE = 1/n * (d - a)^2$ and

$$a = \sum_{i=1}^{n} x_i w_i$$

Adaline

(Widrow, 1959)

- An **Ada**ptive **Lin**ear **E**lement tries to minimize the **amount of error** instead of aiming to reduce the number of misclassifications.
- The Adaline, proposed by Widrow (1959,1960), modifies weight in such a way to diminish the mean square error (MSE) at every iteration.

  $E = (1/n)e^2$

  where $e = (d - a)$ and $a = \sum_{i=1}^{n} x_i w_i$

Adaline (cont…)

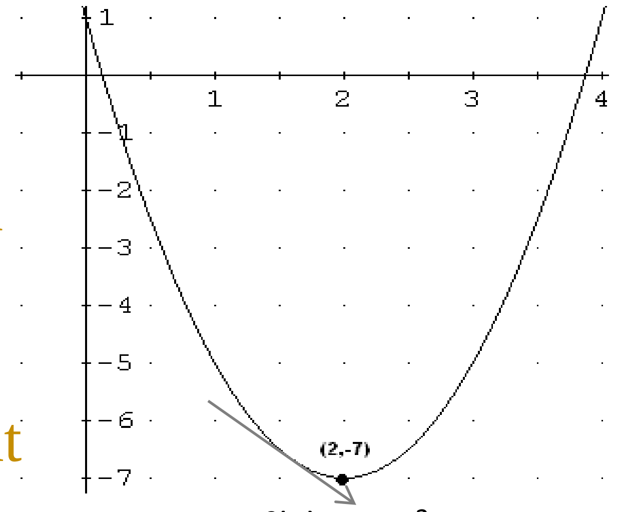
(Widrow, 1959)

- Instead of assuming a binary classification, Adalines consider real values as desired outputs.
  - The training examples are of form (x, d) where $d \in \mathbb{R}$.
  - To produce an output y where $y \in \mathbb{R}$, the network just outputs the activation a.
    - Called linear output function.

Learning Using Gradient Descent

- Adaline uses gradient descent to reduce error
  - If $f(x_1, \ldots, x_n)$ is a *differentiable, scalar-valued function of several variables*, its **gradient** is the *vector* whose components are the *n* partial derivatives of *f*.

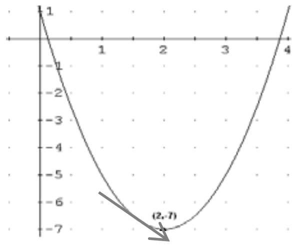
Gradient represents the **slope of the tangent of the graph** of the function and points in the direction of the greatest rate of increase of the function and its magnitude is the slope of the graph in that direction.



$f(x) = 2x^2 - 8x + 1$

Learning Using Gradient Descent

- Adaline uses gradient descent to reduce error
  - **Gradient descent** is a first-order optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the *negative* of the gradient (or of the approximate gradient) of the function at the current point.
- The error surface in Adaline is quadratic.
- Thus it has a single global minimum obtainable from every point on the surface.

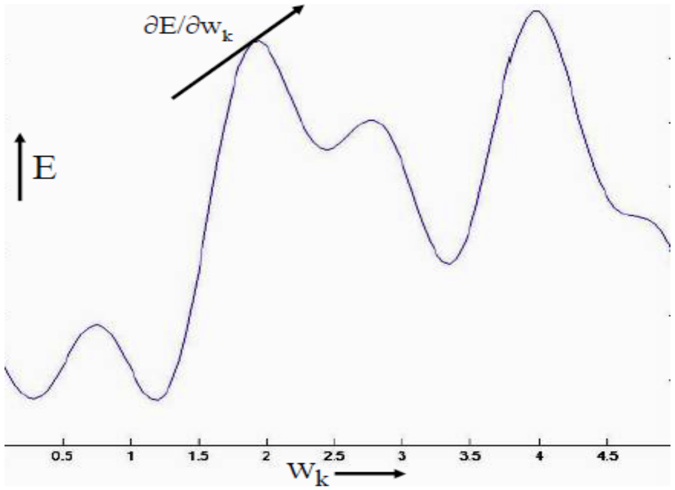


$f(x) = 2x^2 - 8x + 1$

Learning in Adaline

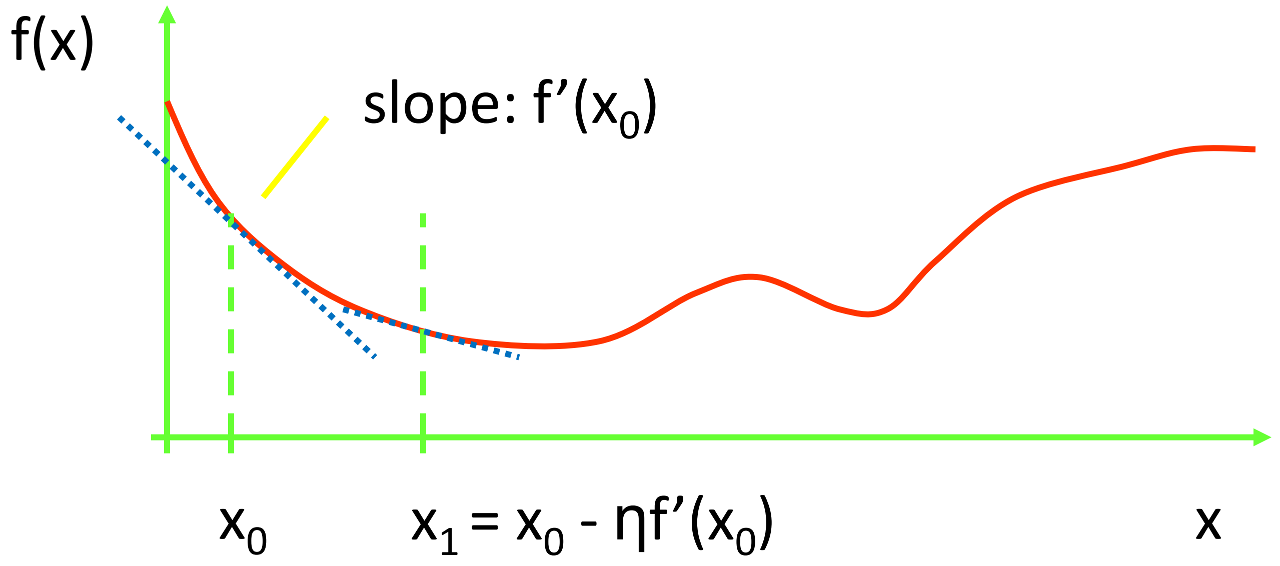
- Adaline itself did not generate much interest but its learning technique got popular and more complex versions of that are used in some popular ANNs.
- Error is minimized by adjusting w.
- Consider the projection of a single weight w_k on the error surface.
- $\partial E/\partial w_k$ is the direction of steepest ascent, and so

  $-\partial E/\partial w_k$ is the direction of steepest descent, which minimizes error.



Gradient Descent Example

- Finding the absolute minimum of a one-dimensional error function f(x):

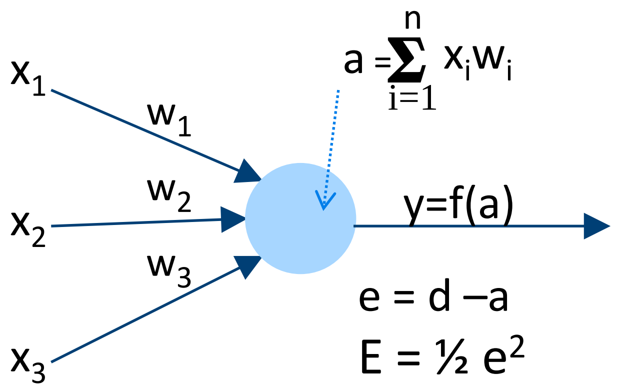


Repeat this iteratively until for some x_i, $f'(x_i)$ is sufficiently close to 0.

Computation of Error

- So, considering two dimensional input vector and $E = \frac{1}{2}e^2 = (d - a)^2$ and $a = \sum_{i=1}^{n} x_i w_i$, w_k should be changed in the direction of $-\partial E/\partial w_k$.
- Applying the *chain rule* to solve the derivative,

$$
\begin{aligned}
-\partial E/\partial w_k &= -\partial E/\partial e.\ \partial e/\partial a.\ \partial a/\partial w_k \\
&= -(d - a)\ .\ (-1)\ .\ x_k \\
&= (d - a)\ x_k
\end{aligned}
$$



Computation of Error

- Any multiple of $(d - a)x_k$ will move down the error surface. We use

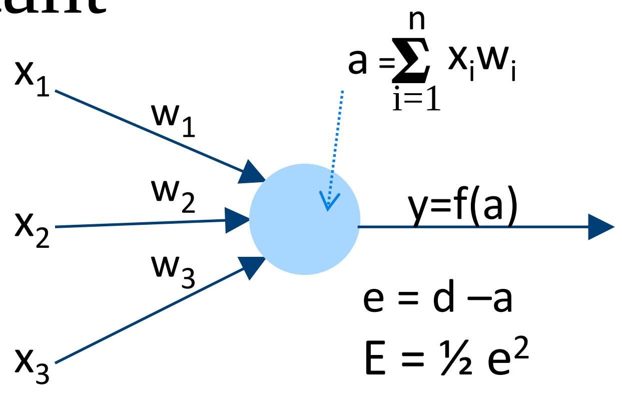
  $\Delta w_k = (d - a)cx_k$

  where $k = 1,.., n$ and

  c = a small +ve constant

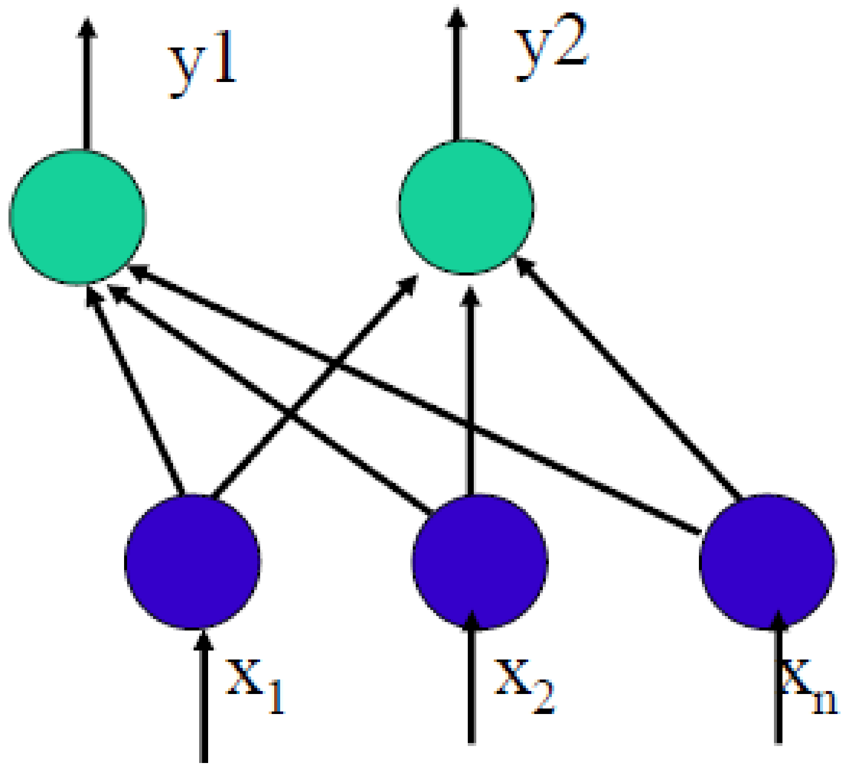
  that represents the

  learning rate



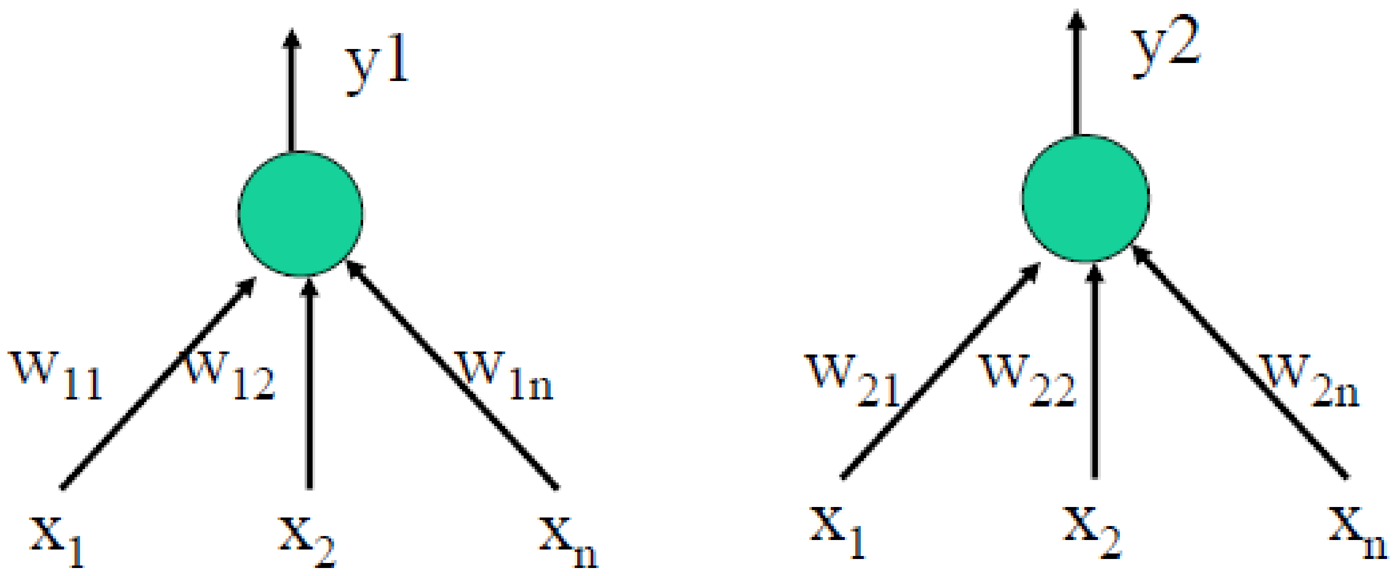
Multiple Output Nodes

- What if we have more than a single output?
- E.g. Classify cells based on dimensions – each class is represented by a single output.



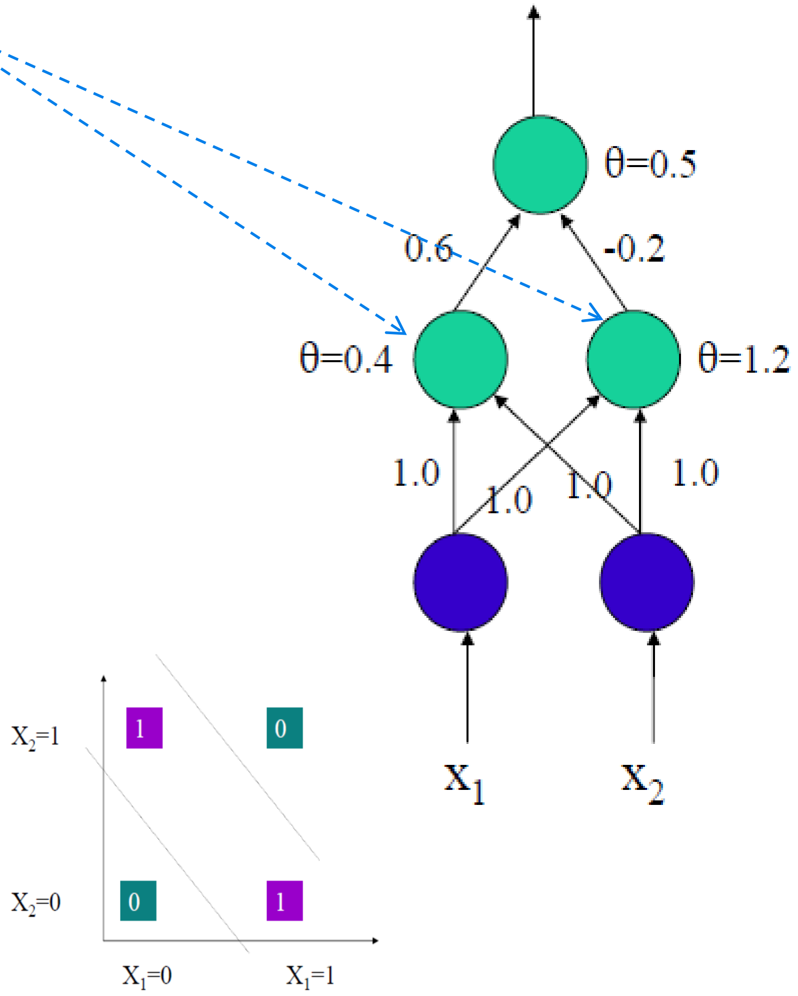
Multiple Output Nodes

- Then it is really like each processor operating on its own with the same inputs.
- What about output values that are not numbers?



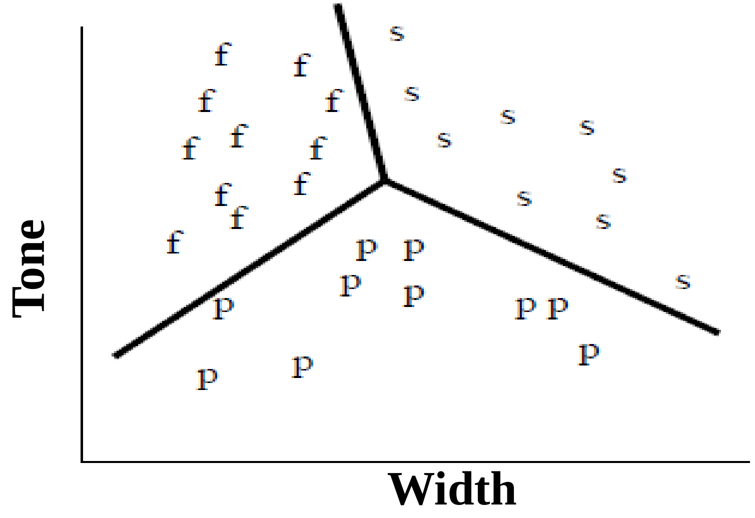
Back to the XOR Problem

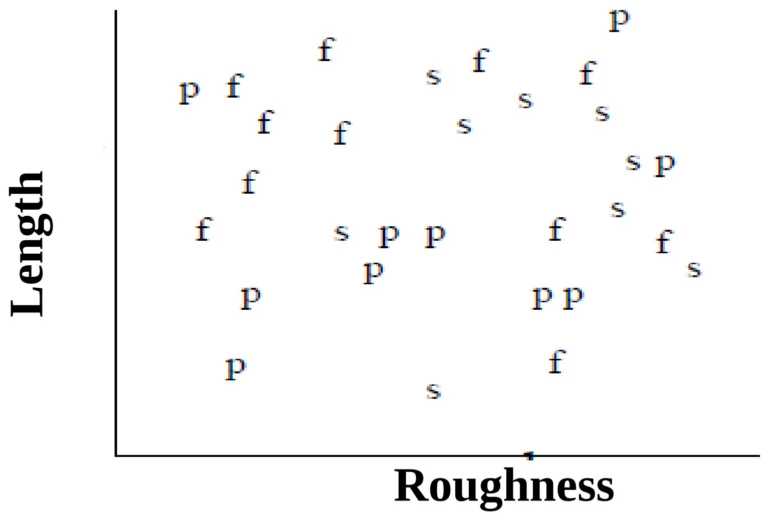
- Can be solved with 2 perceptron processors (one for each decision boundary), and a third that resolves their outputs.
- Different weight and threshold values can be used to solve the same problem.



Feature Selection

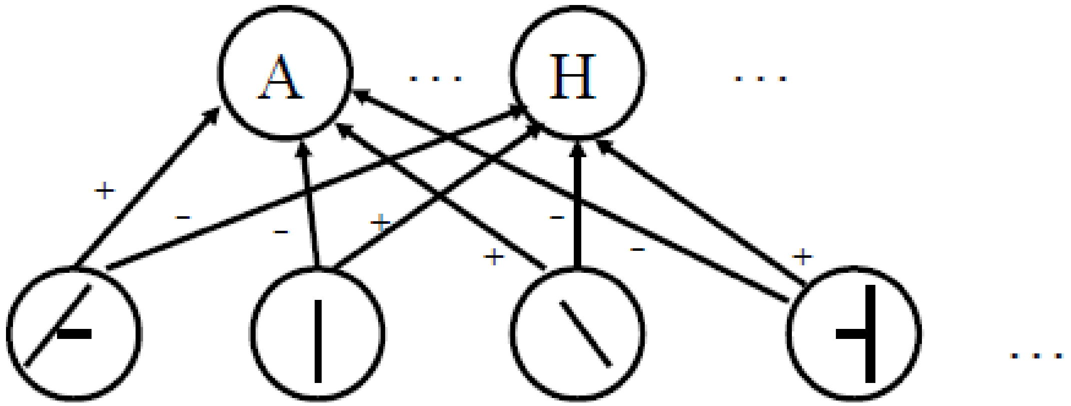
- Classification using **single layer networks** requires the use of correct features. If we try to use features that do not distinguish the classes well, we will fail regardless of how well the network can operate. Consider the following features for classifying wood logs as fir, spruce, or pine.





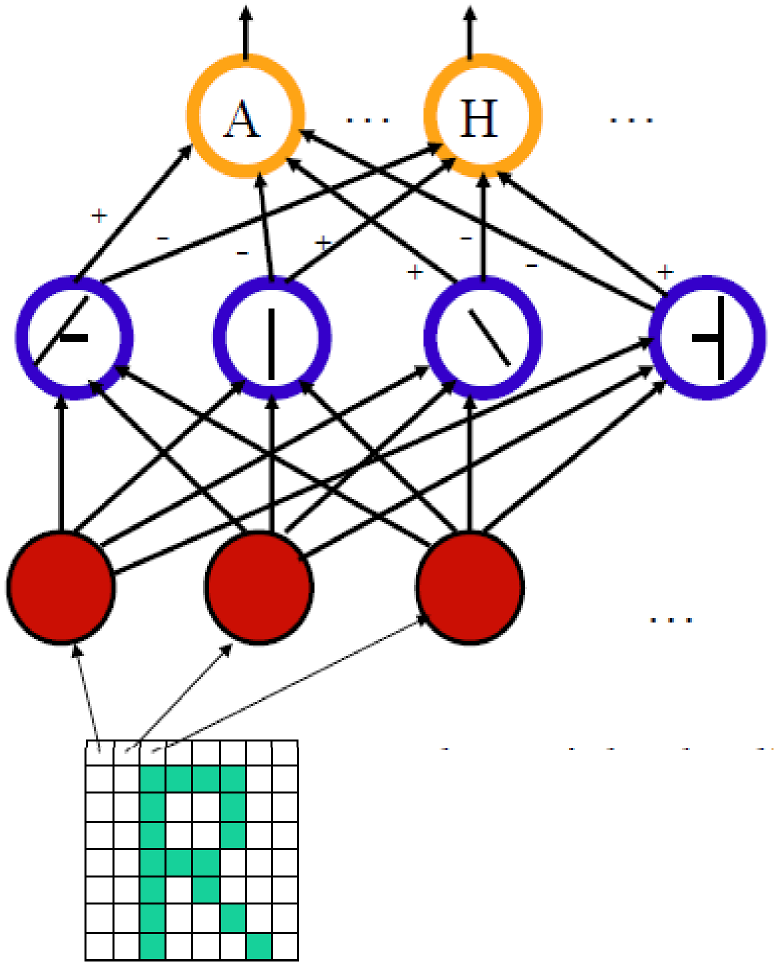
Features: Letter Classification

- All the *single-layer techniques*, as well as *statistical pattern recognition techniques* need someone to know in advance what set of features will work so that it can be incorporated into the system.



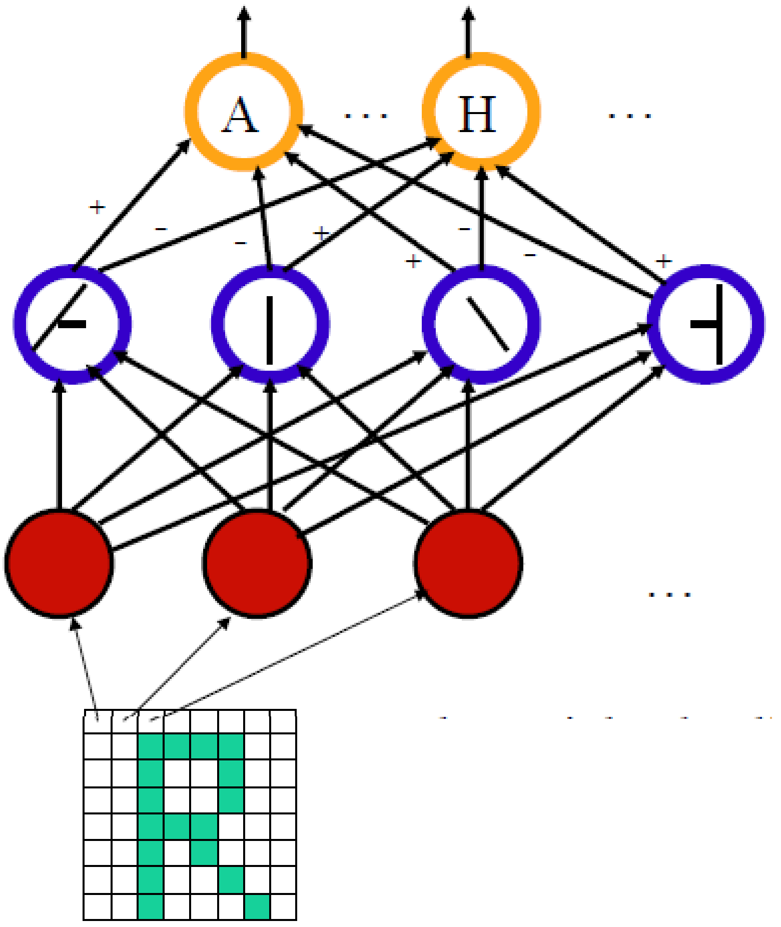
Features (cont…)

- Suppose we added another layer whose inputs are the pixel values of the character image.
    - Could the middle layer "learn" to represent an effective set of features, so that the top layer could perform the desired classification?



Features (cont…)

- The weights leading to the top layer could be trained with examples, but in the 1960's there were *no algorithms for training the weights leading to the middle layer.*



Perceptron and Linear Separability

- In general, networks of perceptron-like processors can solve most non-linearly separable tasks by using more than one layer of processors.
- Rosenblatt (and others) realized this in the 1960's, but did not have a learning rule that would work effectively with more than one level (it wasn't invented until the mid 1970's).

ANN ARCHITECTURE

Fully Connected Networks: Asymmetric

- Fully Connected Networks: Every node is connected to every other node.
- **Asymmetric**: Each connection can have different weight.
- Connections may be
  - Excitatory (positive),
  - Inhibitory (negative), or
- This is the most general neural net architecture of which others are special cases.

Symmetric FCN

- Symmetric fully connected networks: The strength of a connection from one node to another is identical to that of the reverse connection.

ANN Architectures

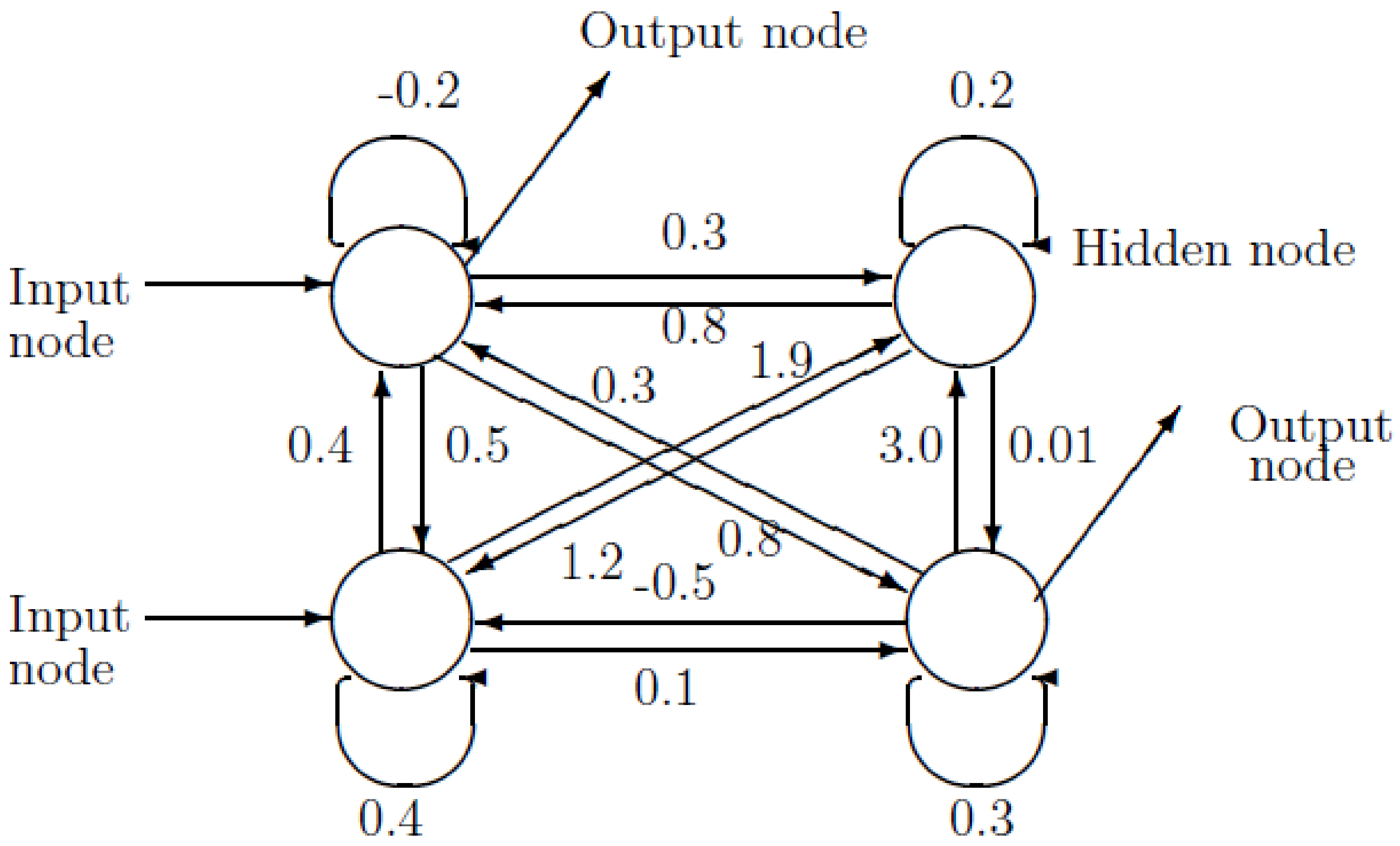


Figure 1.10: Fully Connected Asymmetric Nettwork

Layered Network

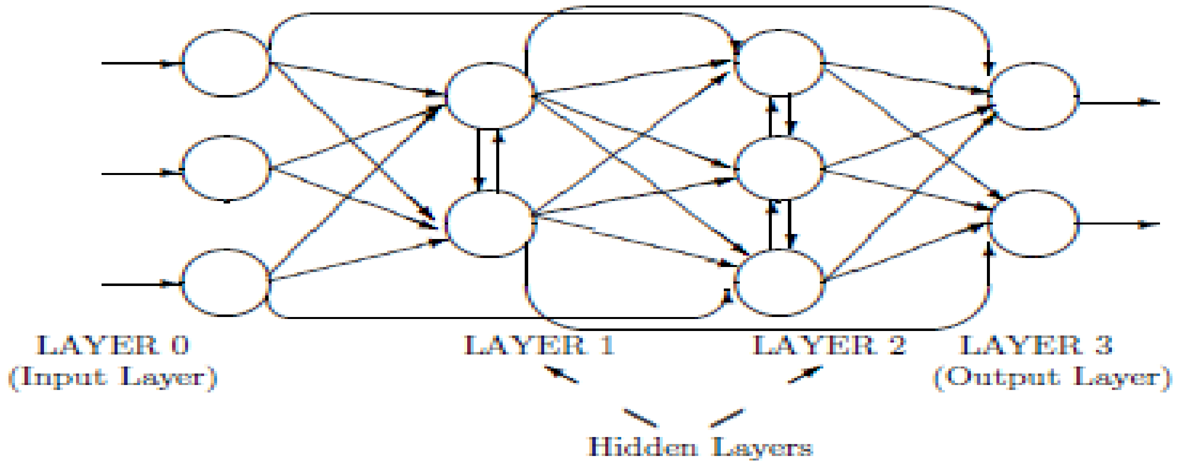


Figure 1.11: Layered Networks

- When nodes are partitioned or organized into subsets called **layers**, with connections only leading from layer k to layer j where j > k, no reverse connection from j to k.
- Inputs arrive at "input layer" or "layer 0", where no real processing occurs.
- Intra-layer connections exist.

Acyclic Networks

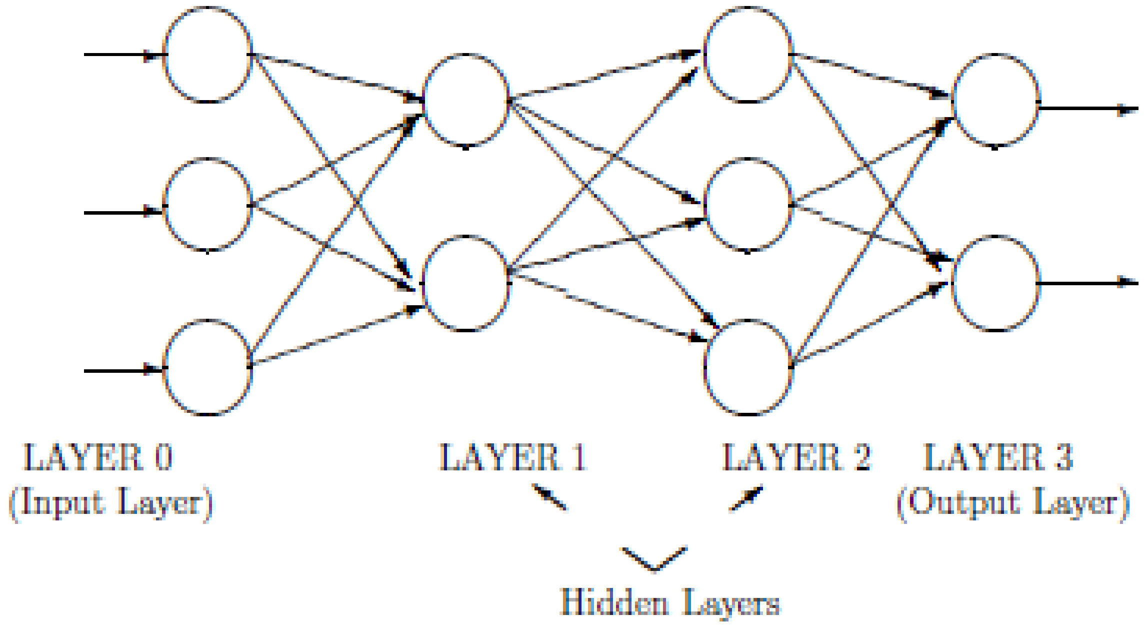


Figure 1.12: Feed Froward 3-2-3-2 Network

- **Acyclic Networks:** Layered networks **without intra-layer connections**.

Feedforward Networks

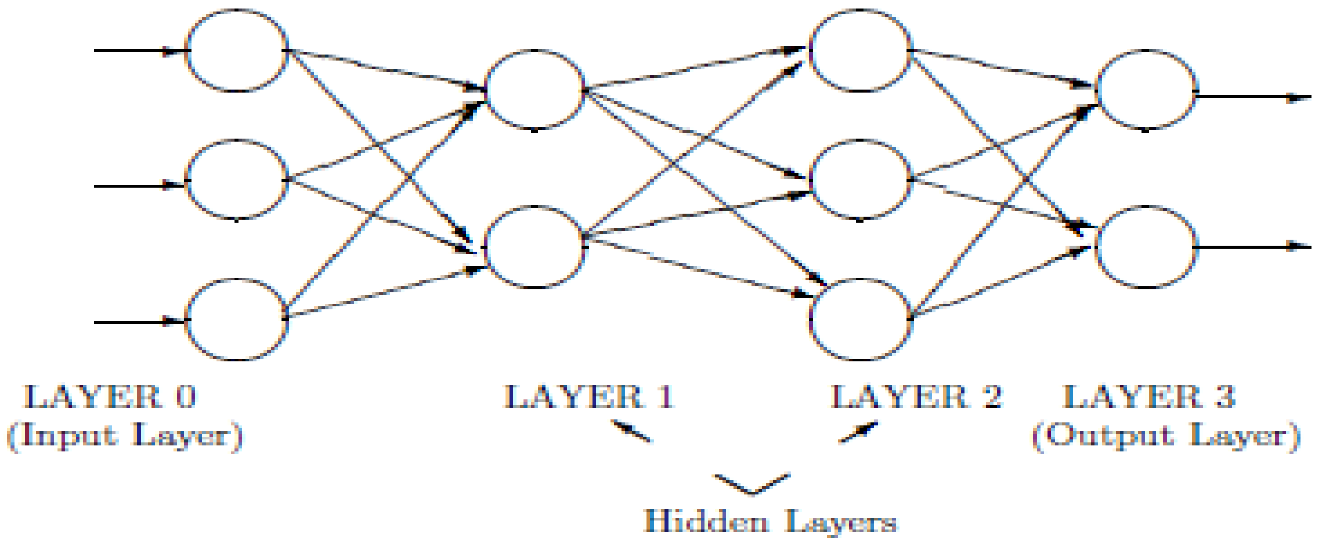


Figure 1.12: Feed Froward 3-2-3-2 Network

- **Feedforward Networks:** A connection is allowed from a node in layer i only to nodes in layer i + 1.
  - Feedforward networks are succinctly described by a **sequence of numbers** indicating the number of nodes in each layer.

Feedforward Networks (cont…)

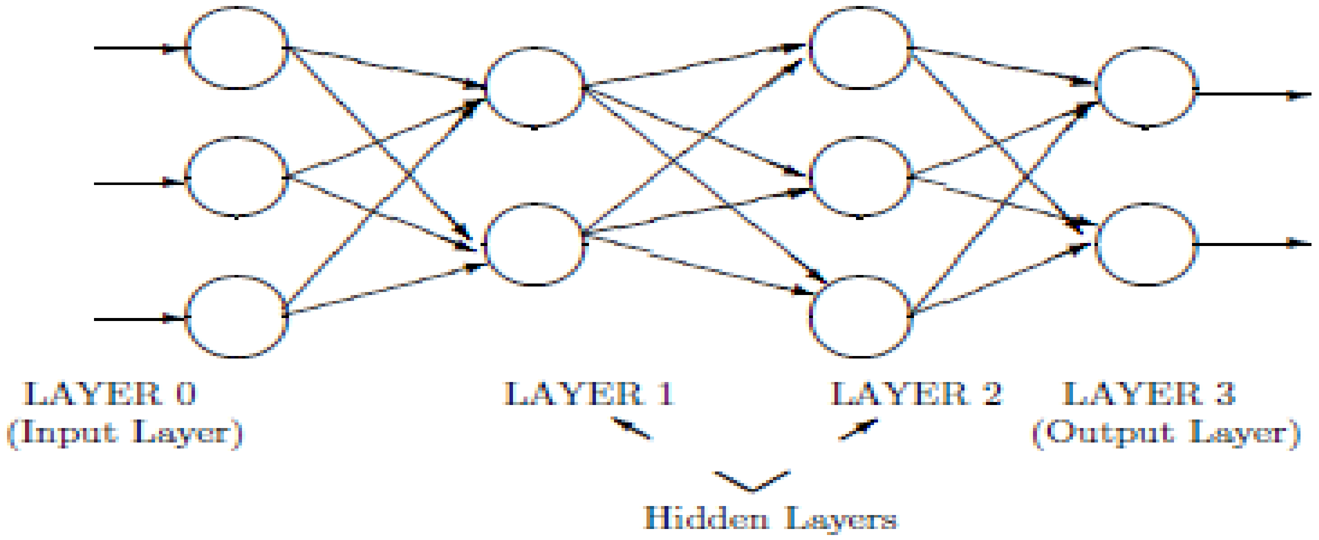


Figure 1.12: Feed Froward 3-2-3-2 Network

- **Feedforward Networks:** These networks, generally with *no more than 4 hidden layers*, are among the **most common** neural nets in use. Conceptually, nodes in successively higher layers abstract successively higher level features from preceding layers.

Modular Network

- Outputs from the 2nd layer are grouped into modules.
- Different modules have no connection between them.
- Used when different processing needs to be applied to different groups of inputs.