Classification

Neural Networks 1

Neural networks

- Topics
  - Perceptrons
    - structure
    - training
    - expressiveness
  - Multilayer networks
    - possible structures
      - activation functions
    - training with gradient descent and backpropagation
    - expressiveness

Connectionist models

- Consider humans:
  - Neuron switching time ~ 0.001 second
  - Number of neurons ~ 10^{10}
  - Connections per neuron ~ 10^{4-5}
  - Scene recognition time ~ 0.1 second
  - 100 inference steps doesn't seem like enough

$\Rightarrow$ Massively parallel computation

Neural networks

- Properties:
  - Many neuron-like threshold switching units
  - Many weighted interconnections among units
  - Highly parallel, distributed process
  - Emphasis on tuning weights automatically

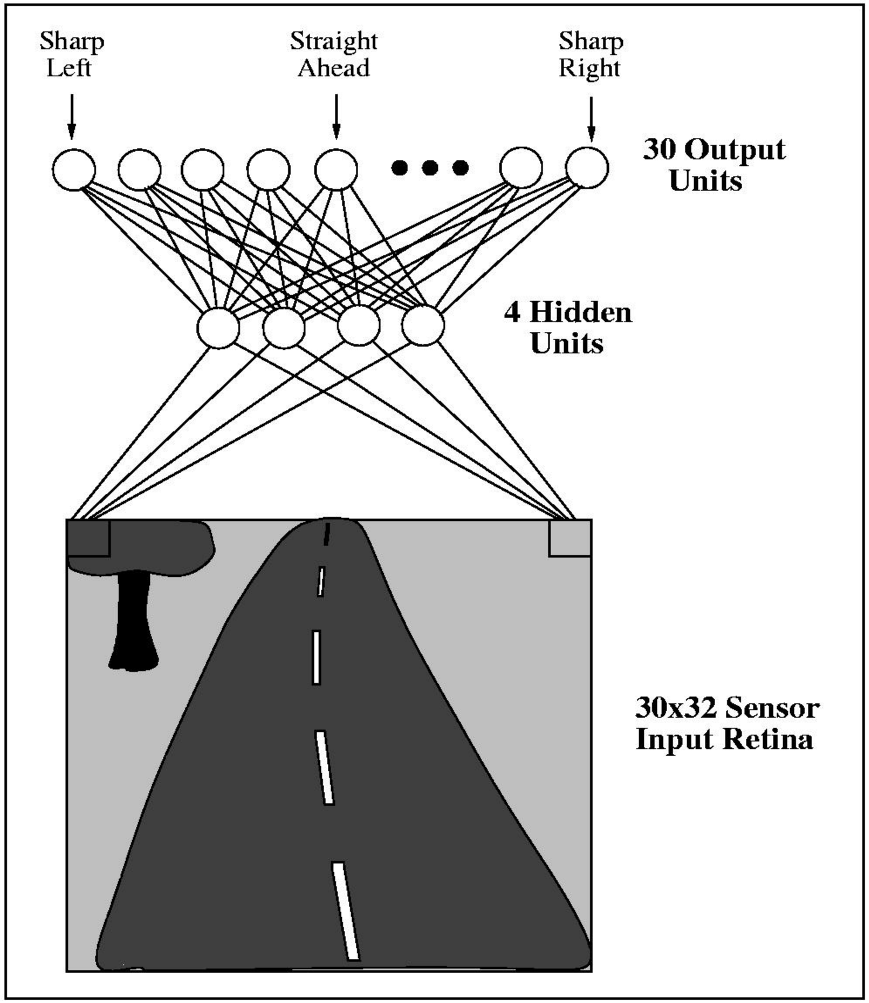
Neural network application

ALVINN: An Autonomous Land Vehicle In a Neural Network

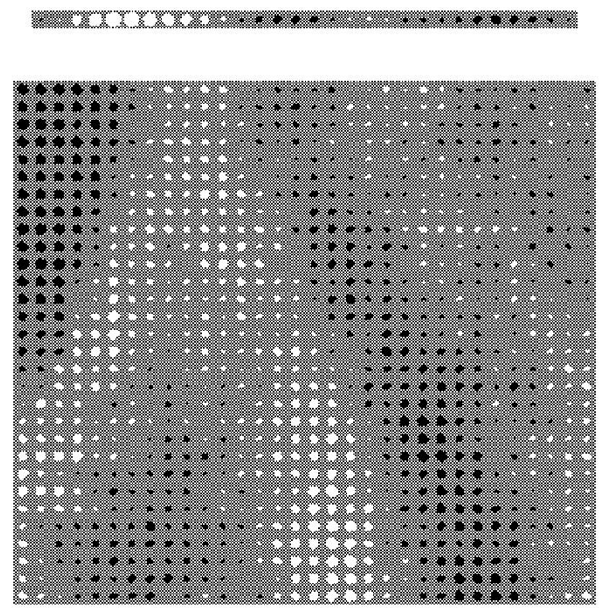
(Carnegie Mellon University Robotics Institute, 1989-1997)

ALVINN is a perception system which learns to control the NAVLAB vehicles by watching a person drive. ALVINN's architecture consists of a single hidden layer back-propagation network. The input layer of the network is a 30x32 unit two dimensional "retina" which receives input from the vehicles video camera. Each input unit is fully connected to a layer of five hidden units which are in turn fully connected to a layer of 30 output units. The output layer is a linear representation of the direction the vehicle should travel in order to keep the vehicle on the road.

Neural network application

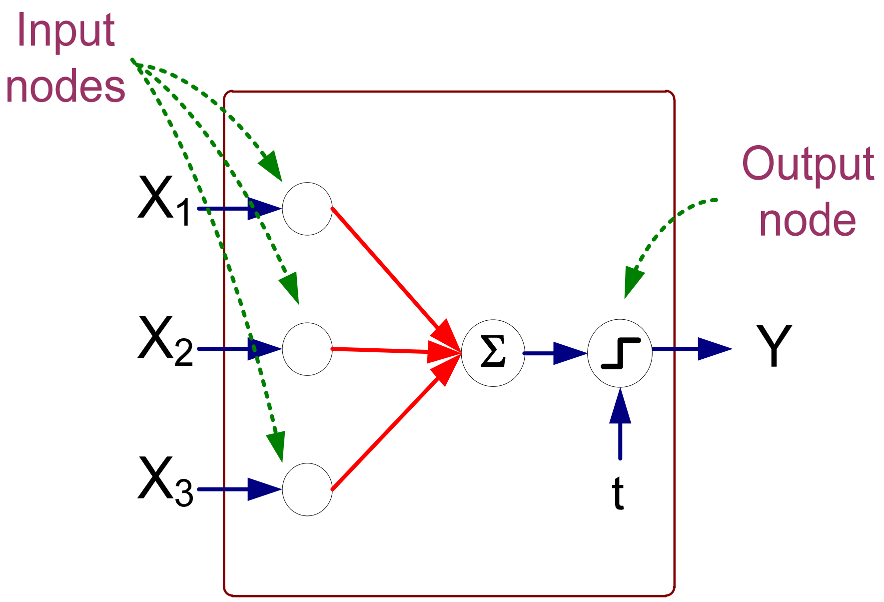


ALVINN drives 70 mph on highways!



Perceptron structure

- Model is an assembly of nodes connected by weighted links
- Output node sums up its input values according to the weights of their links
- Output node sum then compared against some threshold *t*

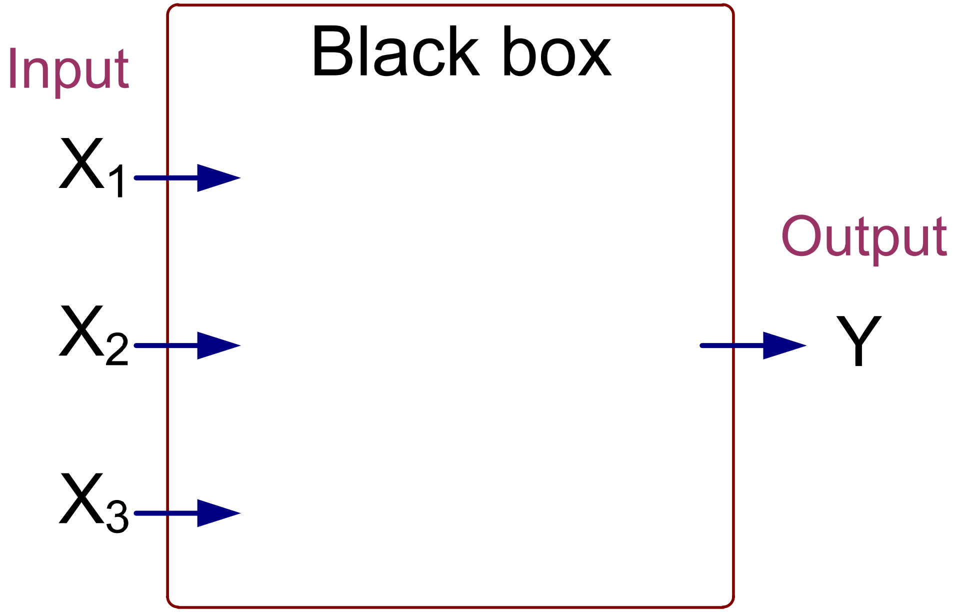


$$y = I(\sum_j w_j x_j - t) \quad \text{or}$$

$$y = sign(\sum_j w_j x_j - t)$$

Example: modeling a Boolean function

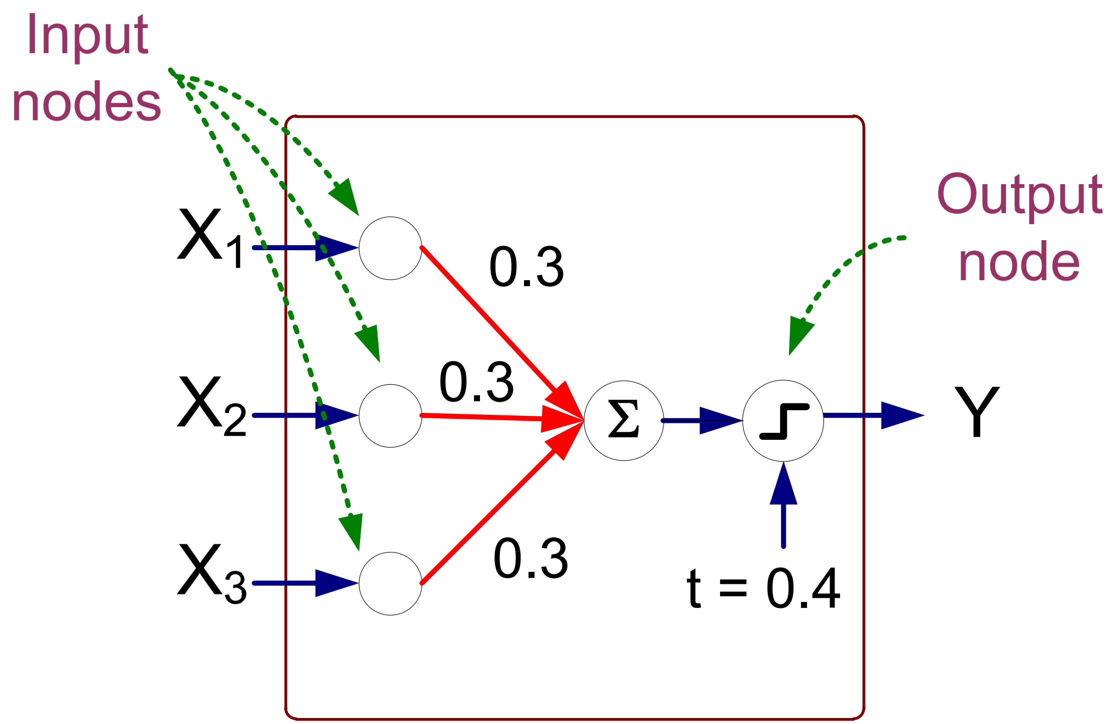
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

Perceptron model

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

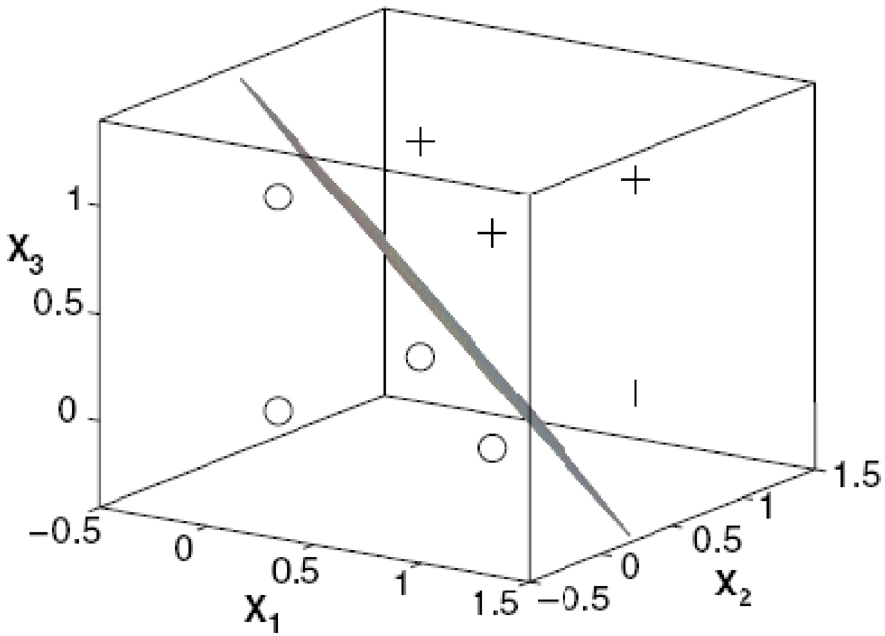


$$y = I(0.3x_1 + 0.3x_2 + 0.3x_3 > 0.4)$$

$$\text{where} \quad I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Perceptron decision boundary

Perceptron decision boundaries are linear
(hyperplanes in higher dimensions)

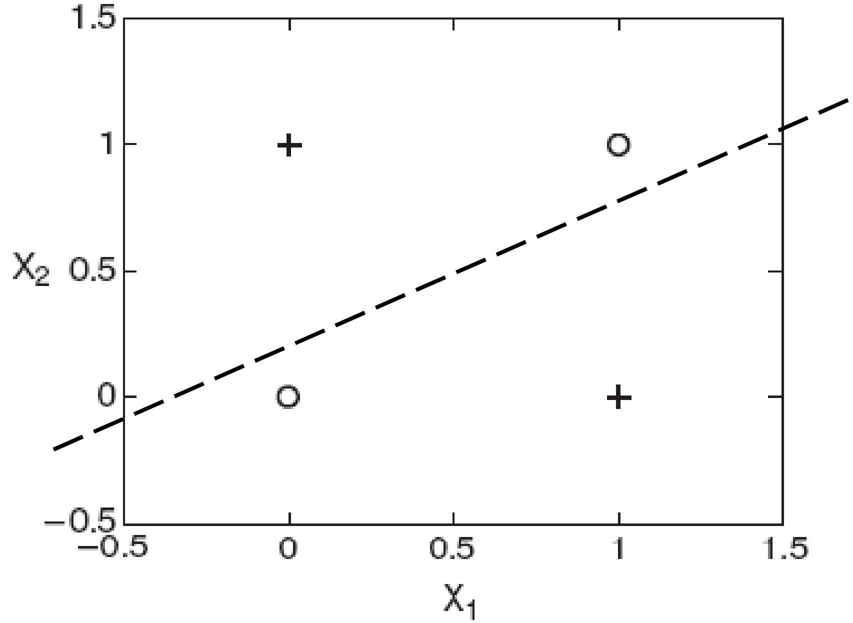


Example: decision surface for Boolean function on preceding slides

Expressiveness of perceptrons

- Can model any function where positive and negative examples are linearly separable
  - Examples: Boolean AND, OR, NAND, NOR
- Cannot (fully) model functions which are not linearly separable.
  - Example: Boolean XOR

X_1	X_2	y
0	0	–1
1	0	1
0	1	1
1	1	–1

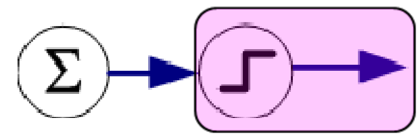


Perceptron training process

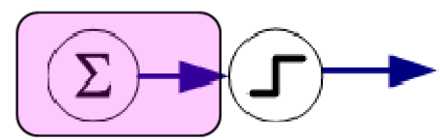
1. Initialize weights with random values.
2. Do
   a. Apply perceptron to each training example.
   b. If example is misclassified, modify weights.
3. Until all examples are correctly classified, or process has converged.

Perceptron training process

- Two rules for modifying weights during training:
  - Perceptron training rule



    - train on thresholded outputs
    - driven by *binary* differences between correct and predicted outputs
    - modify weights with incremental updates
  - Delta rule



    - train on unthresholded outputs
    - driven by *continuous* differences between correct and predicted outputs
    - modify weights via gradient descent

Perceptron training rule

1. Initialize weights with random values.
2. Do
   a. Apply perceptron to each training sample x_i.
   b. If sample x_i is misclassified, modify all weights w_j.

$$w_j \leftarrow w_j + \eta(y_i - \hat{y}_i)x_{ij} \quad \text{where}$$

x_{ij} is input j from sample x_i

y_i is target (correct) output for sample x_i (0 or 1)

$\hat{y}_i$ is thresholded perceptron output (0 or 1)

η is learning rate (a small constant)

3. Until all samples are correctly classified.

Perceptron training rule

b. If sample x_i is misclassified, modify all weights w_j.

$$w_j \leftarrow w_j + \eta(y_i - \hat{y}_i)x_{ij} \quad \text{where}$$

x_{ij} is input j from sample x_i

y_i is target (correct) output for sample x_i (0 or 1)

$\hat{y}_i$ is thresholded perceptron output (0 or 1)

η is learning rate (a small constant)

Examples:

$y_i = \hat{y}_i$ — no update

$y_i - \hat{y}_i = 1$; x_{ij} small, positive — w_j increased by small amount

$y_i - \hat{y}_i = 1$; x_{ij} large, negative — w_j decreased by large amount

$y_i - \hat{y}_i = -1$; x_{ij} large, negative — w_j increased by large amount

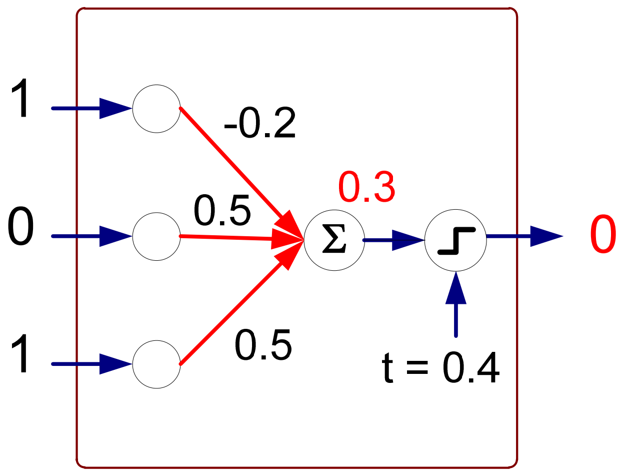
Perceptron training rule

- Example of processing one sample

X_1	X_2	X_3	Y
1	0	1	1

$$\eta = 0.1$$

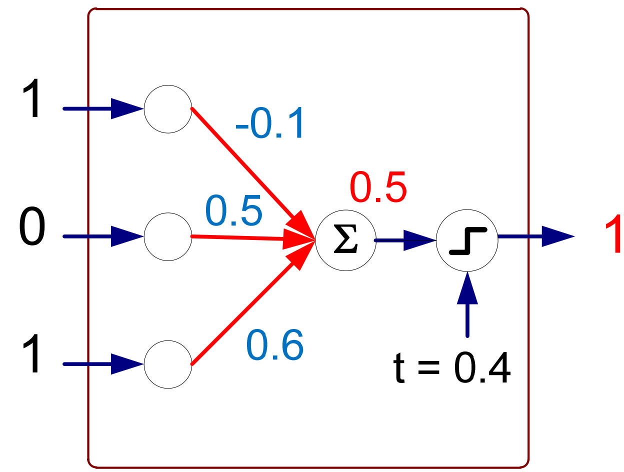
$$y_i - \hat{y}_i = 1$$



$$\eta(y_i - \hat{y}_i)x_{i1} = 0.1$$

$$\eta(y_i - \hat{y}_i)x_{i2} = 0.0$$

$$\eta(y_i - \hat{y}_i)x_{i3} = 0.1$$



Delta training rule

- Based on squared error function for weight vector:

$$E(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \hat{y}_i)^2 = \frac{1}{2}\sum_i (y_i - \mathbf{w}\cdot\mathbf{x}_i)^2$$

Note that error is difference between correct output and <u>unthresholded</u> sum of inputs, a continuous quantity (rather than *binary* difference between correct output and thresholded output).

- Weights are modified by descending gradient of error function.

Squared error function for weight vector w

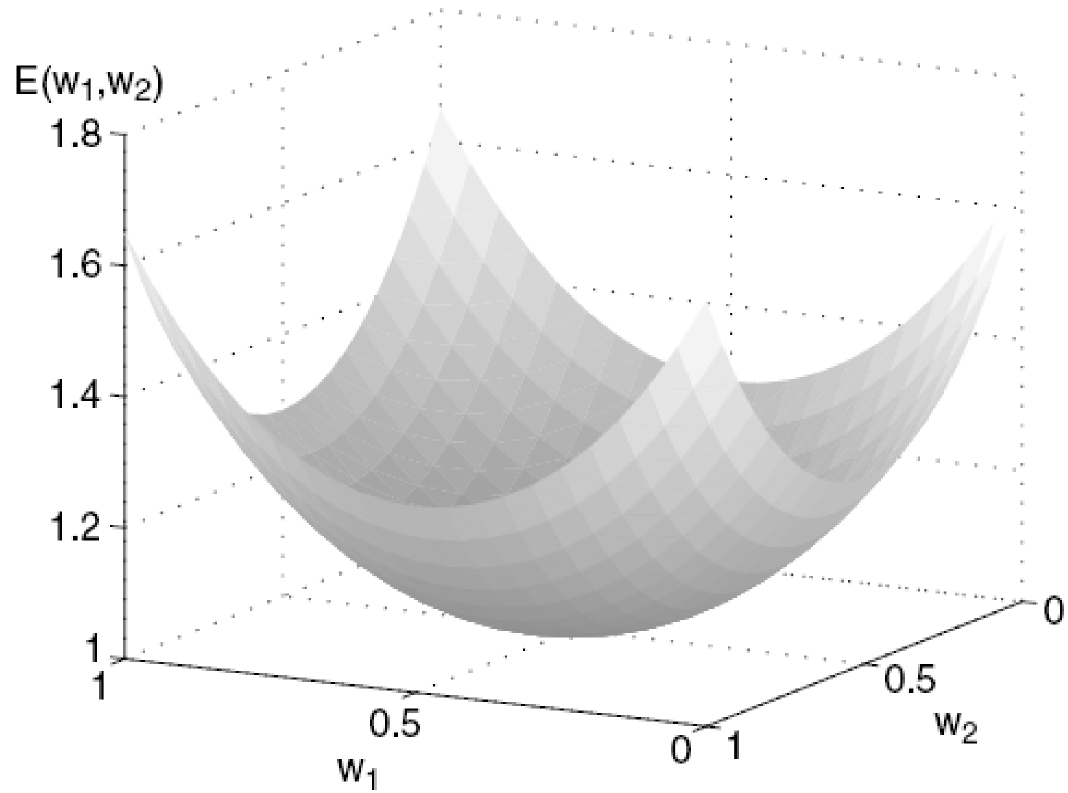


Figure 5.20. Error surface $E(w_1, w_2)$ for a two-parameter model.

Gradient of error function

Gradient :

$$\nabla E(\mathbf{w}) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_d} \right]$$

Training rule for **w** :

$$\Delta \mathbf{w} = -\eta \nabla E(\mathbf{w})$$

Training rule for individual weight :

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j}$$

Gradient of squared error function

$$\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i (y_i - \hat{y}_i)^2$$

$$= \frac{1}{2} \frac{\partial}{\partial w_j} \sum_i (y_i - \hat{y}_i)^2$$

$$= \frac{1}{2} \sum_i 2(y_i - \hat{y}_i) \frac{\partial}{\partial w_j} (y_i - \hat{y}_i)$$

$$= \sum_i (y_i - \hat{y}_i) \frac{\partial}{\partial w_j} (y_i - \mathbf{w} \cdot \mathbf{x}_i)$$

$$\frac{\partial E}{\partial w_j} = \sum_i (y_i - \hat{y}_i)(-x_{ij})$$

Delta training rule

1. Initialize weights with random values.
2. Do
   a. Apply perceptron to each training sample x_i.
   b. If sample x_i is misclassified, modify all weights w_j.

$$w_j \leftarrow w_j + \eta(y_i - \hat{y}_i)x_{ij} \quad \text{where}$$

x_{ij} is input j from sample x_i

y_i is target (correct) output for sample x_i (0 or 1)

$\hat{y}_i$ is unthresholded perceptron output (continuous)

η is learning rate (a small constant)

3. Until all samples are correctly classified, or process converges.

Gradient descent: batch vs. incremental

- Incremental mode (illustrated on preceding slides)
  - Compute error and weight updates for a single sample.
  - Apply updates to weights before processing next sample.
- Batch mode
  - Compute errors and weight updates for a block of samples (maybe all samples).
  - Apply all updates simultaneously to weights.

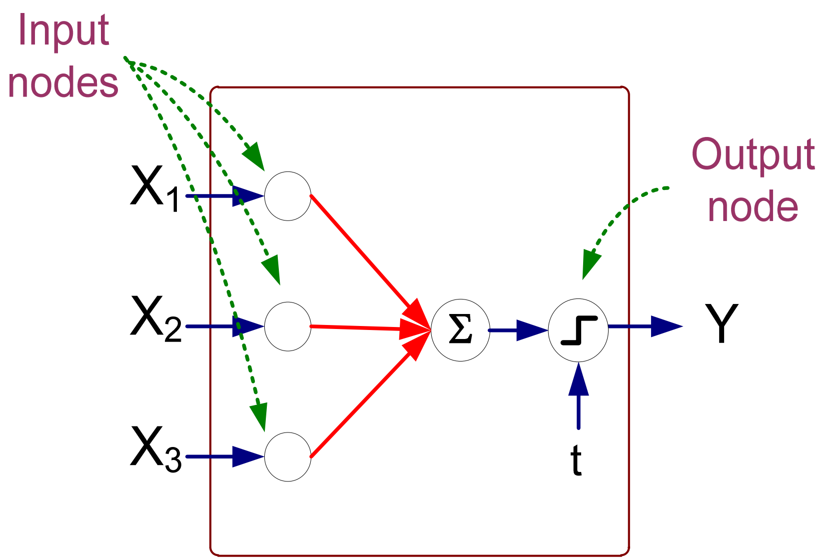
Perceptron training rule vs. delta rule

- Perceptron training rule guaranteed to correctly classify all training samples if:
  - Samples are linearly separable.
  - Learning rate η is sufficiently small.

- Delta rule uses gradient descent. Guaranteed to converge to hypothesis with minimum squared error if:
  - Learning rate η is sufficiently small.

  Even when:
  - Training data contains noise.
  - Training data not linearly separable.

Equivalence of perceptron and linear models



$$y = I(\sum_j w_j x_j - t)$$

$\equiv$

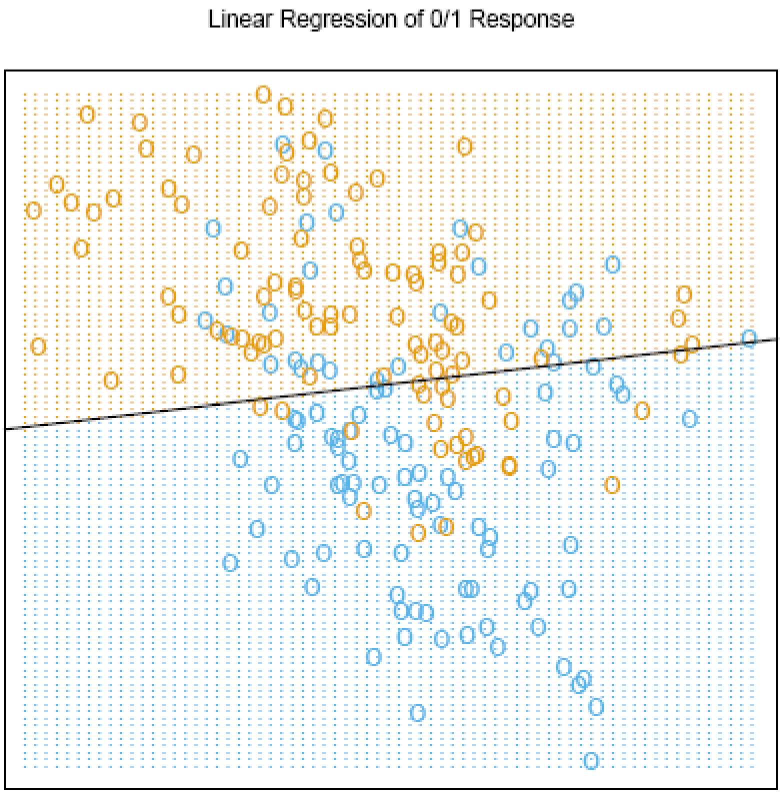


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (BLUE $= 0$, ORANGE $= 1$), and then fit by linear regression. The line is the decision boundary defined by $x^T\hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.*