Intro to ML

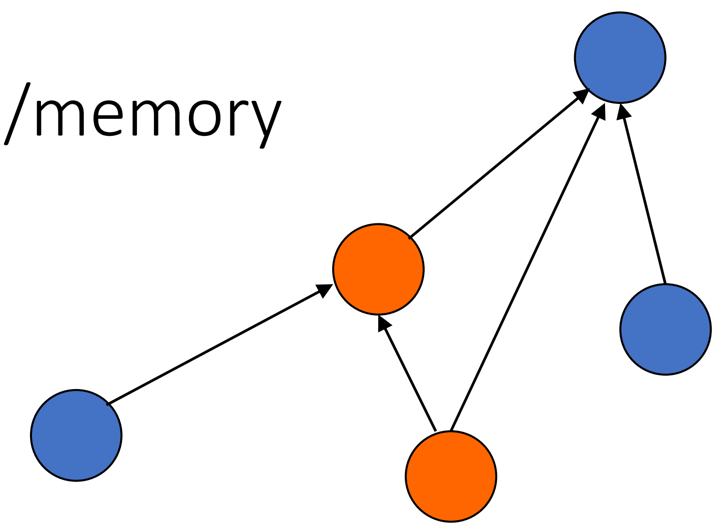
November 29th, 2021

CHAPTER 11:

Multilayer Perceptrons

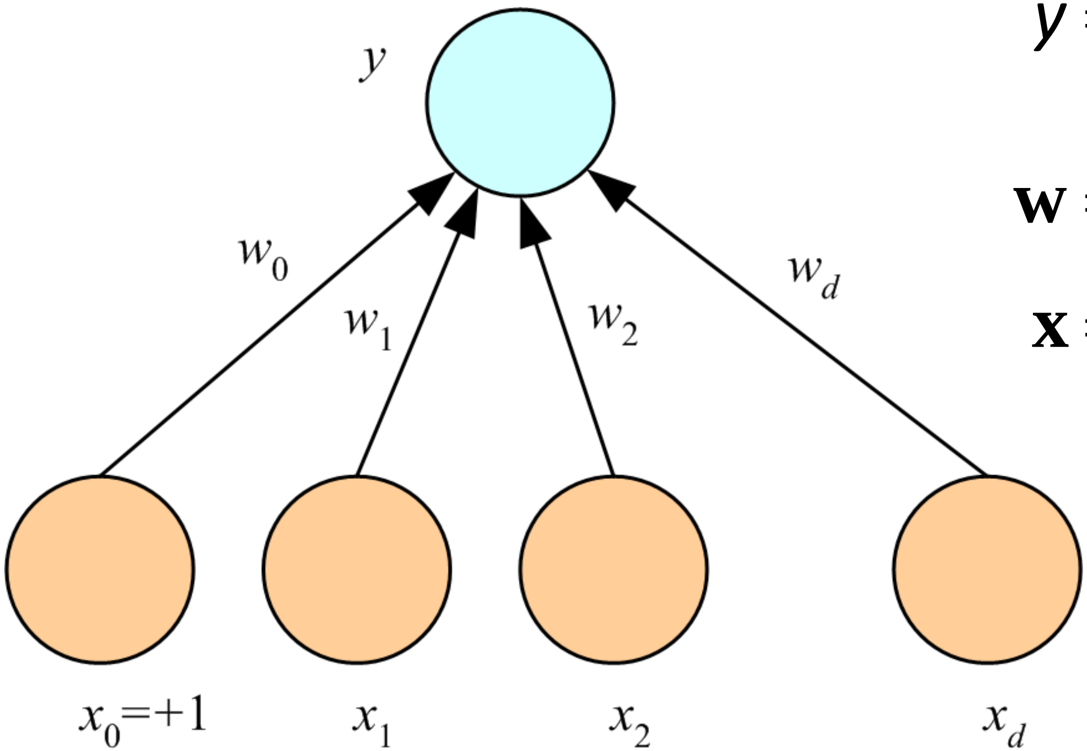
Neural Networks (human brain)

- Networks of processing units (neurons) with connections (synapses) between them
- Large number of neurons: 10^{10}
- Large connectitivity: 10^5
- Parallel processing
- Distributed computation/memory
- Robust to noise, failures



Perceptron

Basic processing unit



$$y = \sum_{j=1}^{d} w_j x_j + w_0 = \mathbf{w}^T \mathbf{x}$$

$$\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$$

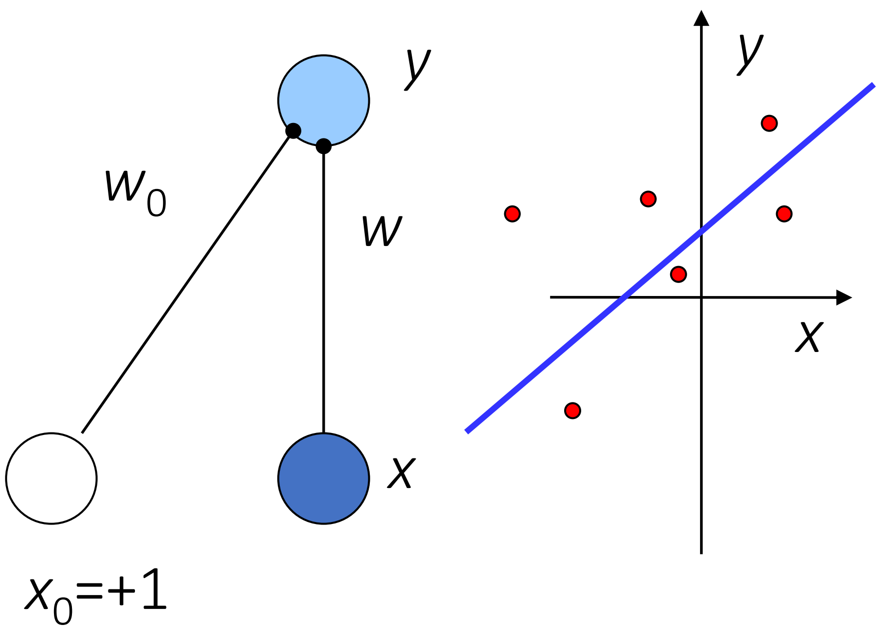
$$\mathbf{x} = [1, x_1, \ldots, x_d]^T$$

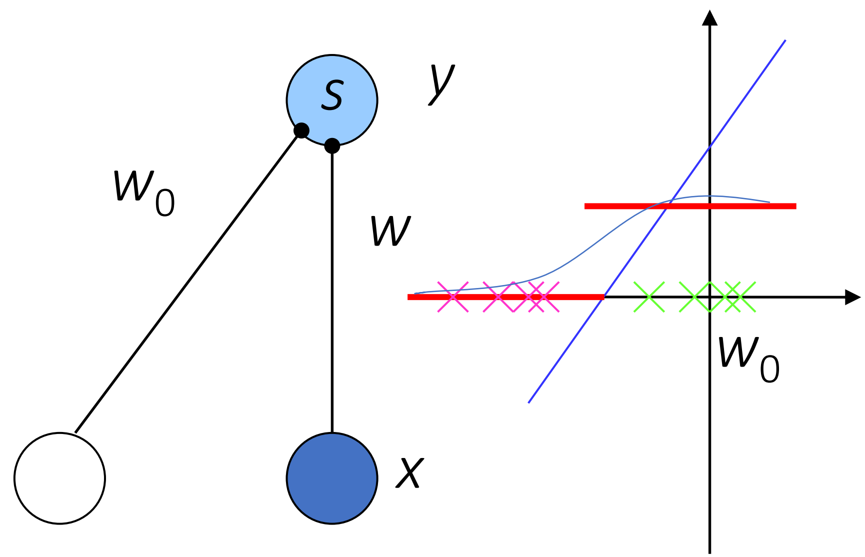
(Rosenblatt, 1962)

What a Perceptron Does

Simple discriminant

- Regression: $y = wx + w_0$
- Classification: $y = 1(wx + w_0 > 0)$

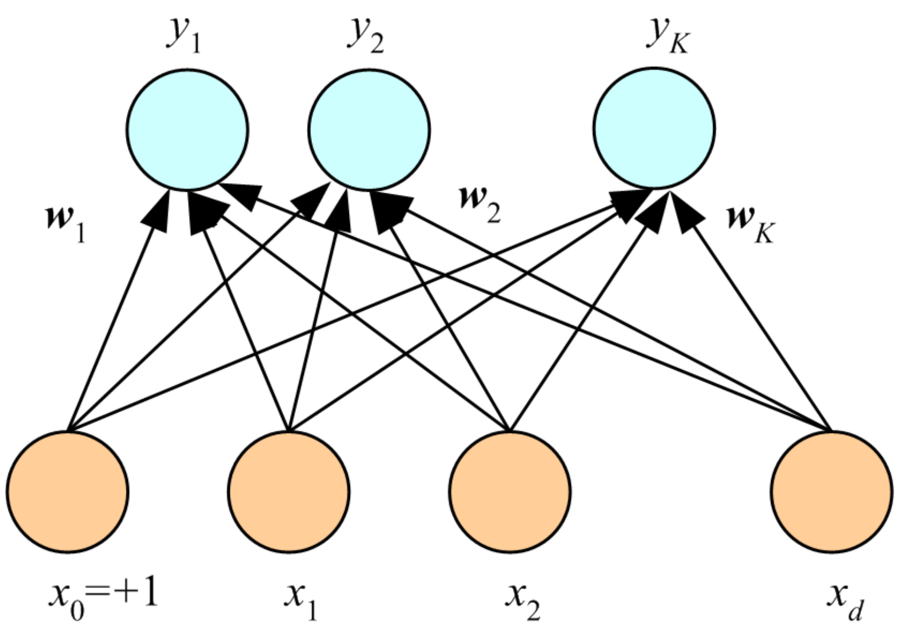




posterior

$$y = \text{sigmoid}(o) = \frac{1}{1 + \exp\left[-\mathbf{w}^T \mathbf{x}\right]}$$

K Outputs



Changing from d dimensions to K dimensions

Regression:

$$y_i = \sum_{j=1}^{d} w_{ij} x_j + w_{i0} = \mathbf{w}_i^T \mathbf{x}$$

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

Classification:

$$o_i = \mathbf{w}_i^T \mathbf{x}$$

$$y_i = \frac{\exp o_i}{\sum_k \exp o_k}$$

choose C_i

if $y_i = \max_k y_k$

Imagine this as two steps:

1. Weighted sum
2. Softmax

Training

- Online (instances seen one by one) vs batch (whole sample) learning:
  - No need to store the whole sample
  - Problem may change in time
  - Wear and degradation in system components
- Stochastic gradient-descent: Update after a single pattern
- Generic update rule (LMS rule):

Update rule for each training sample

$$\Delta w_{ij}^t = \eta\left(r_i^t - y_i^t\right)x_j^t$$

$$\text{Update} = \text{LearningFactor} \cdot \left(\text{DesiredOutput} - \text{ActualOutput}\right) \cdot \text{Input}$$

Training a Perceptron: Regression

- Regression (Linear output):

$$E^t\left(\mathbf{w} \mid \mathbf{x}^t, r^t\right) = \frac{1}{2}\left(r^t - y^t\right)^2 = \frac{1}{2}\left[r^t - \left(\mathbf{w}^T \mathbf{x}^t\right)\right]^2$$

$$\Delta w_j^t = \eta\left(r^t - y^t\right)x_j^t$$

Update one by one
Stochastic gradient descent

Classification

- Single sigmoid output

$$y^t = \text{sigmoid}\left(\mathbf{w}^T\mathbf{x}^t\right)$$
$$E^t\left(\mathbf{w} \mid \mathbf{x}^t, \mathbf{r}^t\right) = -r^t \log y^t - \left(1 - r^t\right)\log\left(1 - y^t\right)$$
$$\Delta w_j^t = \eta\left(r^t - y^t\right)x_j^t$$

The same as previous chapters (logistic discrimination)

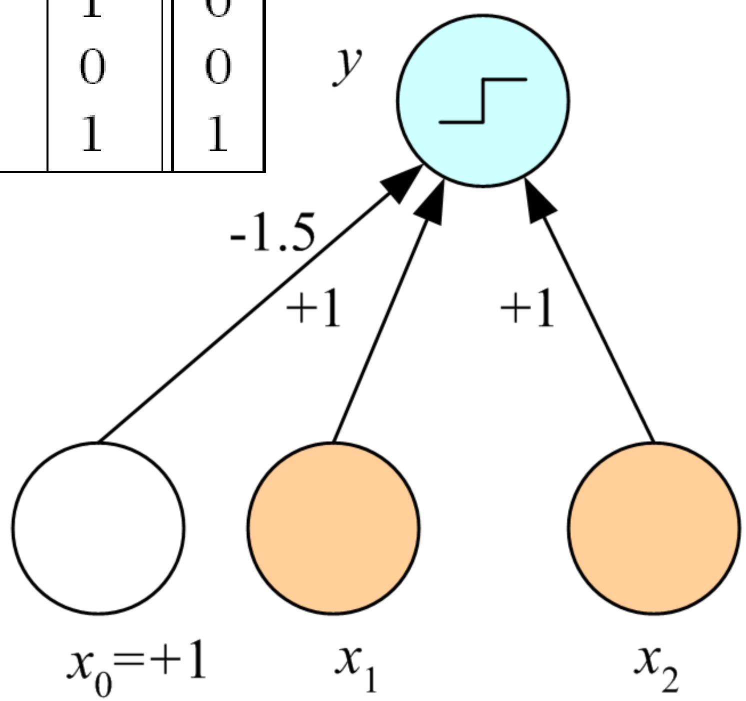
Just for each sample

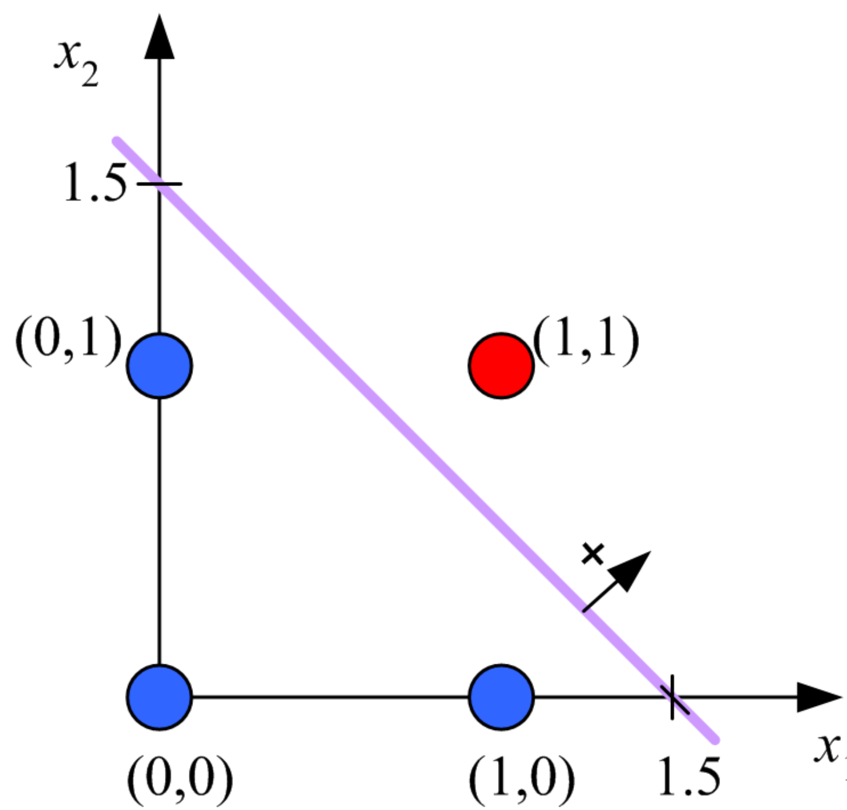
- K>2 softmax outputs

$$y^t = \frac{\exp \mathbf{w}_i^T\mathbf{x}^t}{\sum_k \exp \mathbf{w}_k^T\mathbf{x}^t} \qquad E^t\left(\{\mathbf{w}_i\}_i \mid \mathbf{x}^t, \mathbf{r}^t\right) = -\sum_i r_i^t \log y_i^t$$
$$\Delta w_{ij}^t = \eta\left(r_i^t - y_i^t\right)x_j^t$$

Learning Boolean AND

x_1	x_2	r
0	0	0
0	1	0
1	0	0
1	1	1





XOR

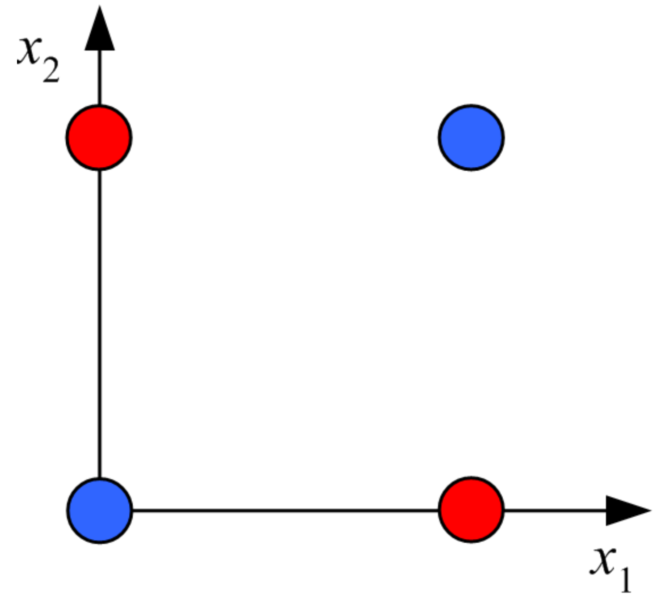
Not linearly separable

Cant be solve by having a line as the perceptron

x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0

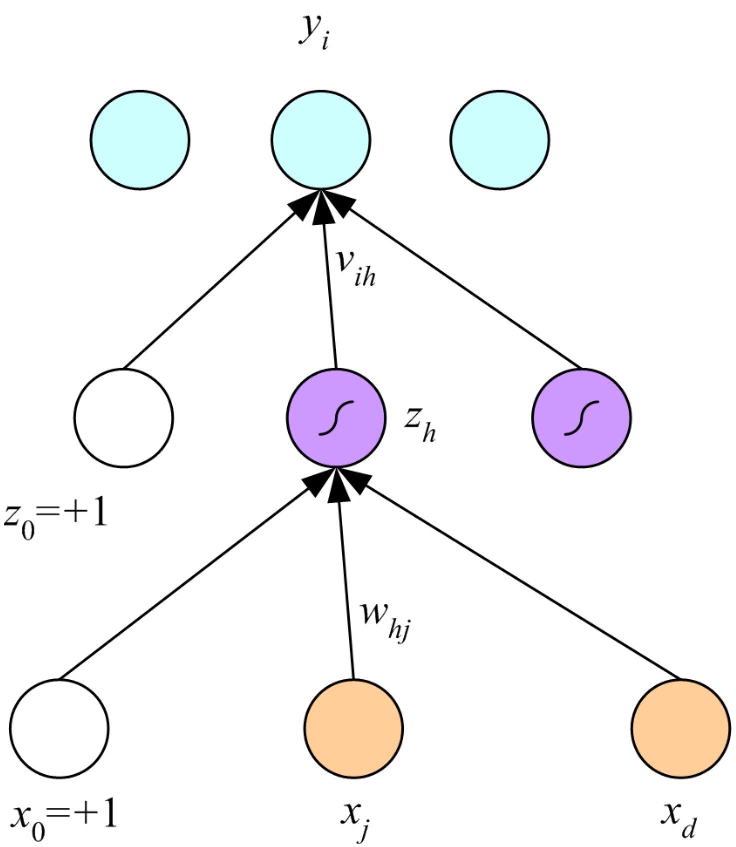
- No w_0, w_1, w_2 satisfy:

$$
\begin{aligned}
w_0 &\leq 0 \\
w_2 + w_0 &> 0 \\
w_1 + w_0 &> 0 \\
w_1 + w_2 + w_0 &\leq 0
\end{aligned}
$$



(Minsky and Papert, 1969)

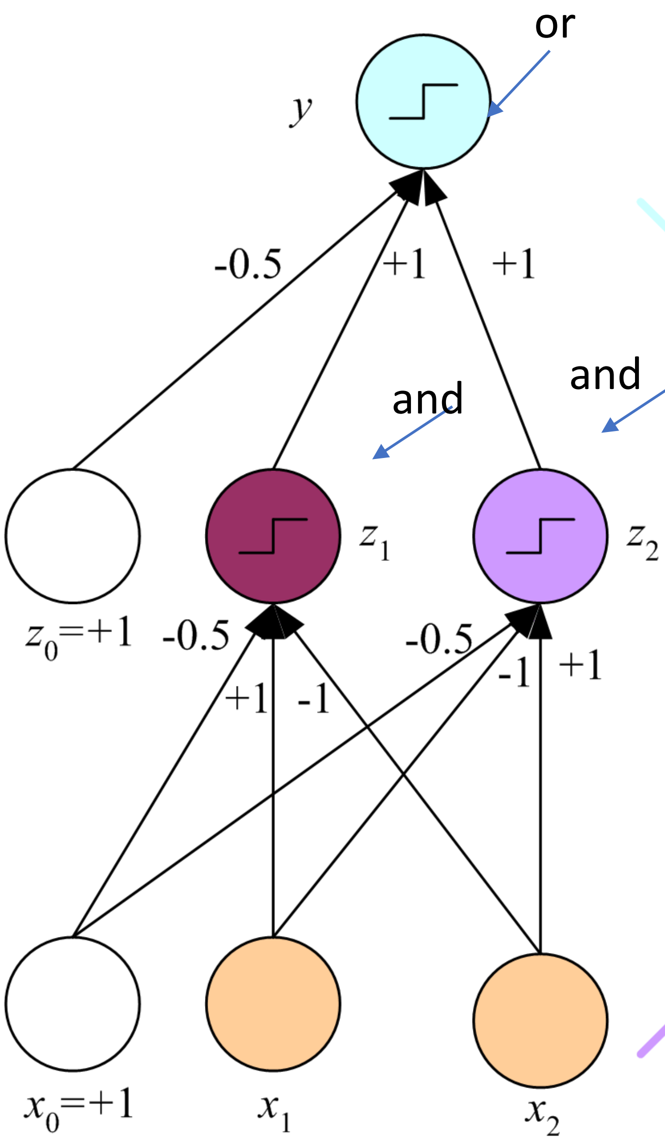
Multilayer Perceptrons

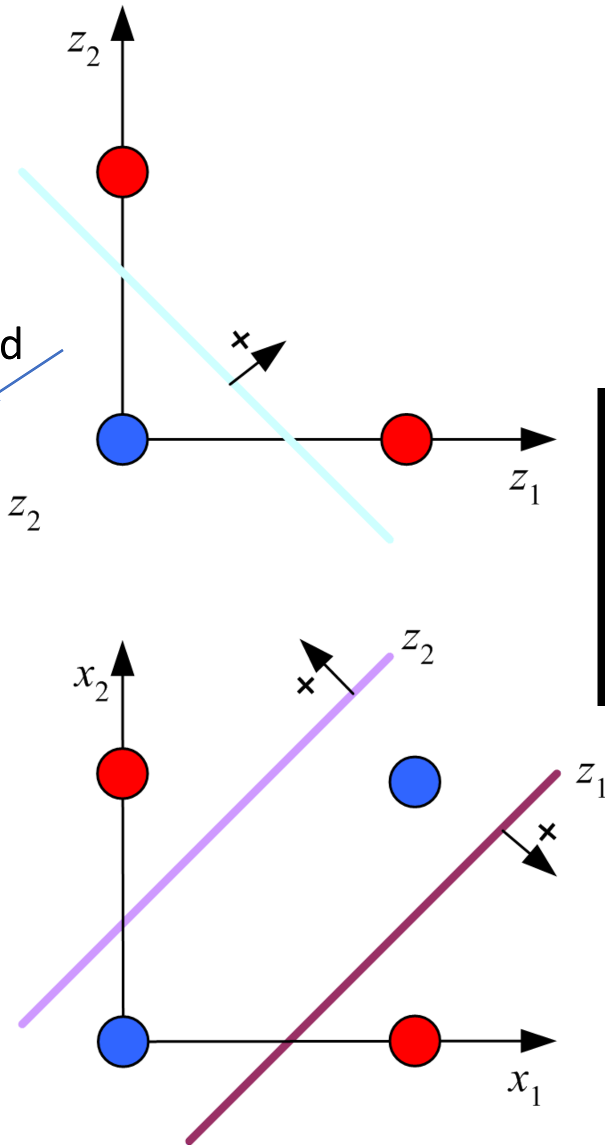


$$y_i = \mathbf{v}_i^T \mathbf{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

$$z_h = \text{sigmoid}\left(\mathbf{w}_h^T \mathbf{x}\right) = \frac{1}{1 + \exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}$$

(Rumelhart et al., 1986)



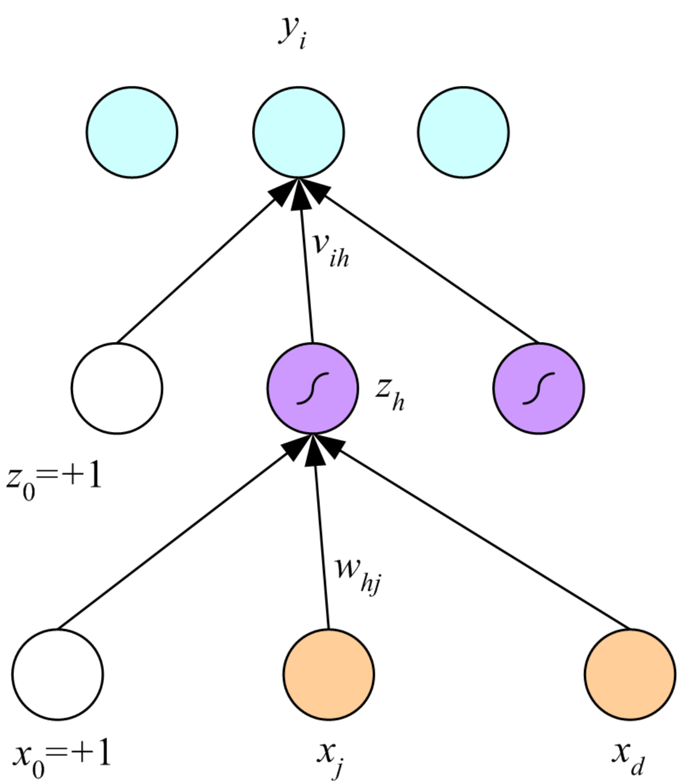


x_1	x_2	z_1	z_2	y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

This implements xor

x_1 XOR x_2 = (x_1 AND ~x_2) OR (~x_1 AND x_2)

Backpropagation



$$y_i = \mathbf{v}_i^T \mathbf{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

$$z_h = \text{sigmoid}\left(\mathbf{w}_h^T \mathbf{x}\right)$$

$$= \frac{1}{1+\exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}$$

$$\boxed{\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial z_h} \frac{\partial z_h}{\partial w_{hj}}}$$

Regression

$$E(\mathbf{W},\mathbf{v}\mid\mathcal{X})=\frac{1}{2}\sum_t\left(r^t-y^t\right)^2$$

$$y^t=\sum_{h=1}^{H}v_h z_h^t+v_0$$

$$\Delta v_h=\sum_t\left(r^t-y^t\right)z_h^t$$

Backward

Forward

$$z_h=\text{sigmoid}\left(\mathbf{w}_h^T\mathbf{x}\right)$$

$$\begin{aligned}\Delta w_{hj}&=-\eta\frac{\partial E}{\partial w_{hj}}\\&=-\eta\sum_t\frac{\partial E}{\partial y^t}\frac{\partial y^t}{\partial z_h^t}\frac{\partial z_h^t}{\partial w_{hj}}\\&=-\eta\sum_t-\left(r^t-y^t\right)v_h z_h^t\left(1-z_h^t\right)x_j^t\\&=\eta\sum_t\left(r^t-y^t\right)v_h z_h^t\left(1-z_h^t\right)x_j^t\end{aligned}$$

Update of w
Through
backpropagate
From h layer

x

Regression with Multiple Outputs

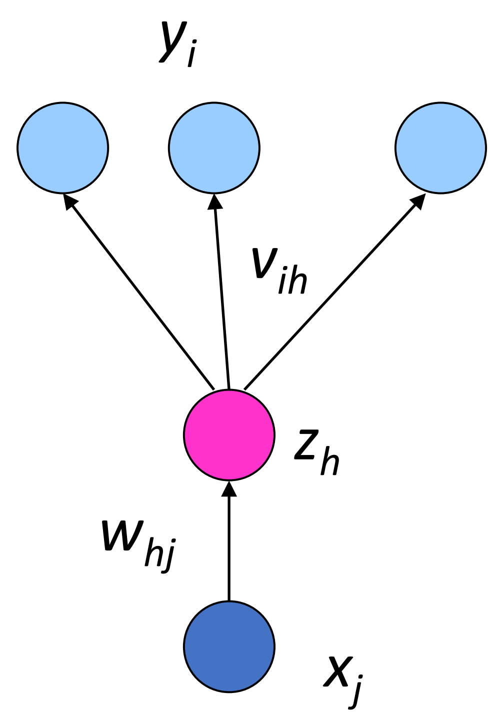
$$E(\mathbf{W},\mathbf{V}|\mathcal{X})=\frac{1}{2}\sum_t\sum_i\left(r_i^t-y_i^t\right)^2$$

$$y_i^t=\sum_{h=1}^{H}v_{ih}z_h^t+v_{i0}$$

$$\Delta v_{ih}=\eta\sum_t\left(r_i^t-y_i^t\right)z_h^t$$

$$\Delta w_{hj}=\eta\sum_t\left[\sum_i\left(r_i^t-y_i^t\right)v_{ih}\right]z_h^t\left(1-z_h^t\right)x_j^t$$

Accumulated backpropagated error of hidden unit h from all outputs



Initialize all v_{ih} and w_{hj} to rand$(-0.01, 0.01)$

Repeat

    For all $(\boldsymbol{x}^t, r^t) \in \mathcal{X}$ in random order

        For $h = 1, \ldots, H$

            $z_h \leftarrow \mathrm{sigmoid}(\boldsymbol{w}_h^T \boldsymbol{x}^t)$

        For $i = 1, \ldots, K$

            $y_i = \boldsymbol{v}_i^T \boldsymbol{z}$

        For $i = 1, \ldots, K$

            $\Delta \boldsymbol{v}_i = \eta (r_i^t - y_i^t) \boldsymbol{z}$

        For $h = 1, \ldots, H$

            $\Delta \boldsymbol{w}_h = \eta (\sum_i (r_i^t - y_i^t) v_{ih}) z_h (1 - z_h) \boldsymbol{x}^t$

        For $i = 1, \ldots, K$

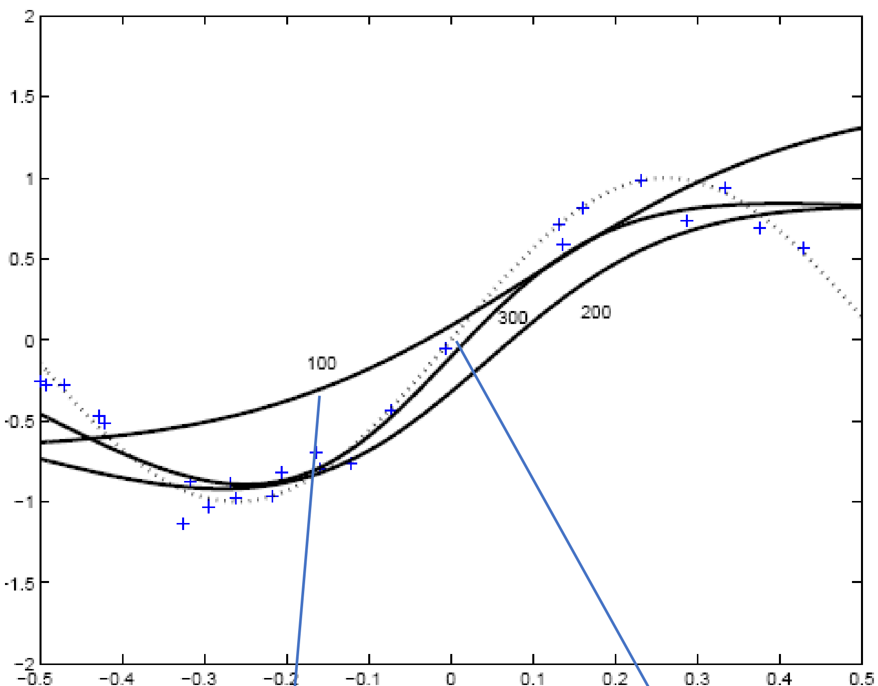
            $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \Delta \boldsymbol{v}_i$

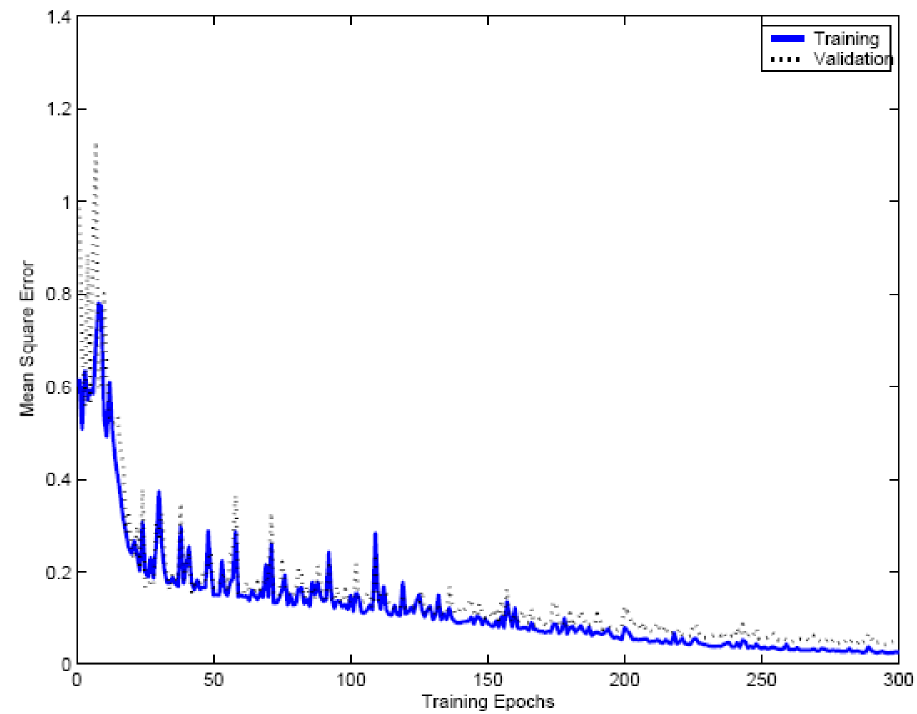
        For $h = 1, \ldots, H$

            $\boldsymbol{w}_h \leftarrow \boldsymbol{w}_h + \Delta \boldsymbol{w}_h$

Until convergence

1d Regression: Convergence



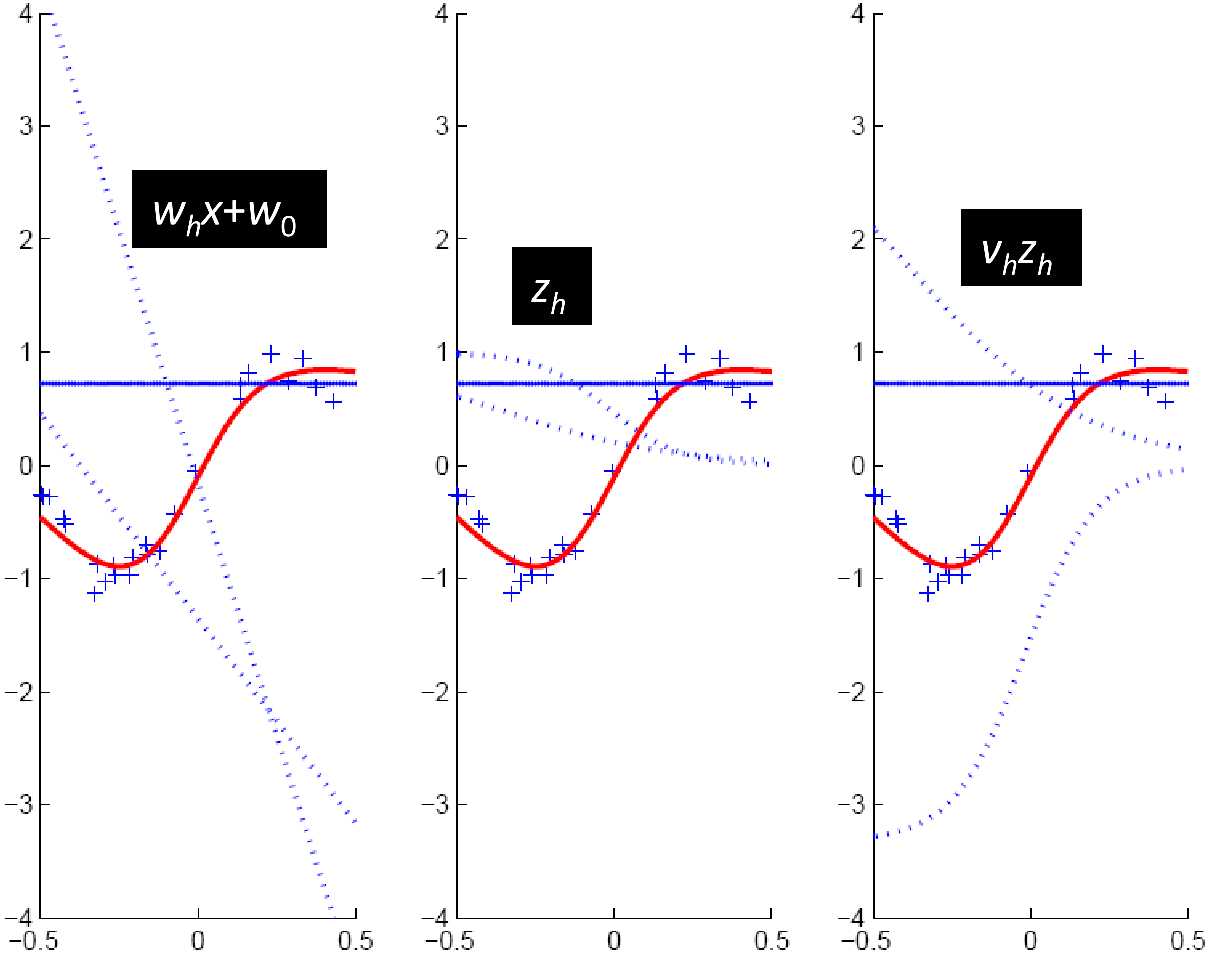


2 hidden units
100 epochs..

$y^t=f(x^t) + N(0,0.1)$, $f(x) = \sin(6x)$

Epoch:
A complete pass over all training samples
Minibatch:
Say 32 samples a time

1d Regression: What hidden units do



Two-Class Discrimination

- One sigmoid output y^t for $P(\mathrm{C}_1|\boldsymbol{x}^t)$ and $P(\mathrm{C}_2|\boldsymbol{x}^t) \equiv 1\text{-}y^t$

$$y^t = \text{sigmoid}\left(\sum_{h=1}^{H} v_h z_h^t + v_0\right)$$

$$E(\mathbf{W},\mathbf{v}|\mathcal{X}) = -\sum_t r^t \log y^t + \left(1-r^t\right)\log\left(1-y^t\right)$$

$$\Delta v_h = \eta \sum_t \left(r^t - y^t\right) z_h^t$$

$$\Delta w_{hj} = \eta \sum_t \left(r^t - y^t\right) v_h z_h^t \left(1 - z_h^t\right) x_j^t$$

K>2 Classes

softmax

$$o_i^t = \sum_{h=1}^{H} v_{ih} z_h^t + v_{i0} \qquad y_i^t = \frac{\exp o_i^t}{\sum_k \exp o_k^t} \equiv P(C_i \mid \mathbf{x}^t)$$

$$E(\mathbf{W}, \mathbf{v} \mid \mathcal{X}) = -\sum_t \sum_i r_i^t \log y_i^t$$

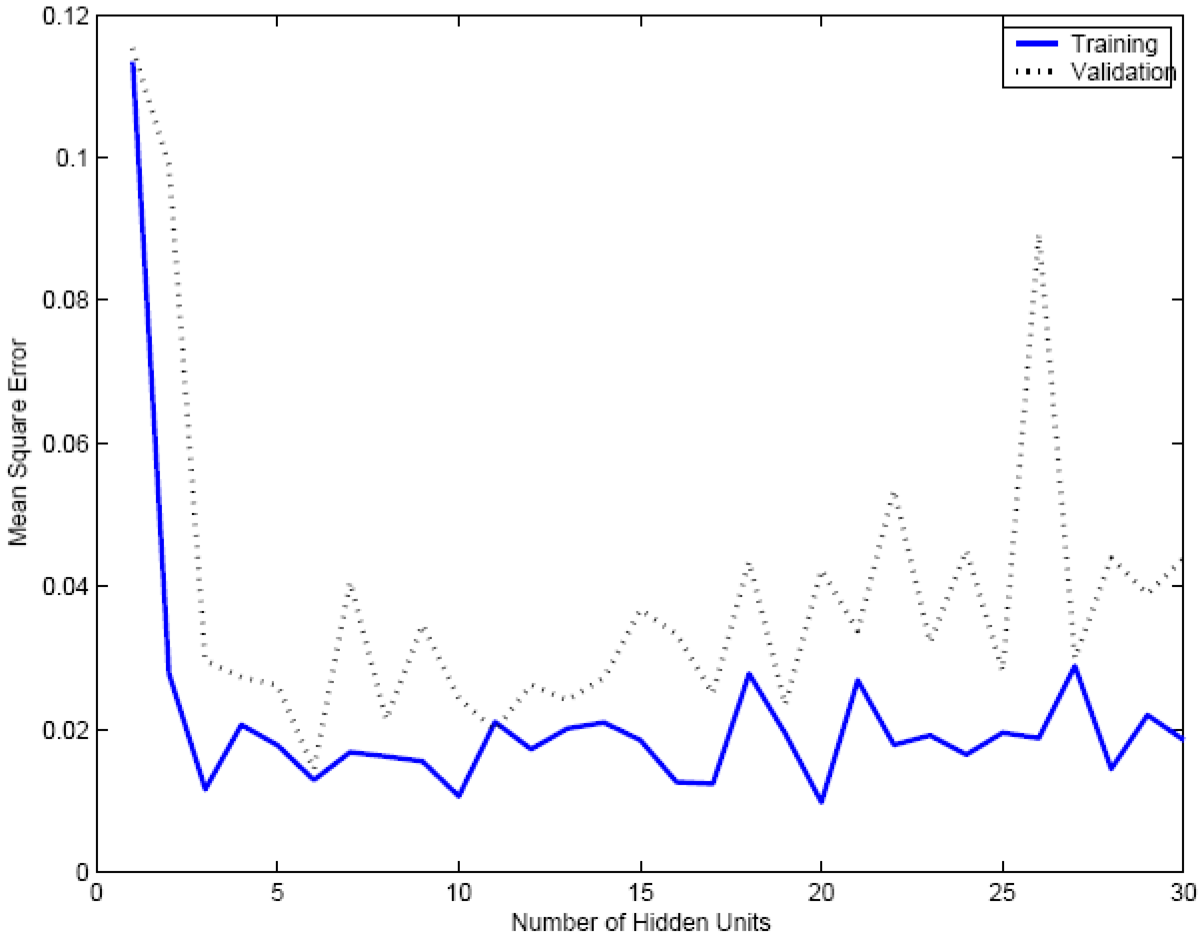
$$\Delta v_{ih} = \eta \sum_t (r_i^t - y_i^t) z_h^t$$

$$\Delta w_{hj} = \eta \sum_t \left[\sum_i (r_i^t - y_i^t) v_{ih} \right] z_h^t (1 - z_h^t) x_j^t$$

Overfitting

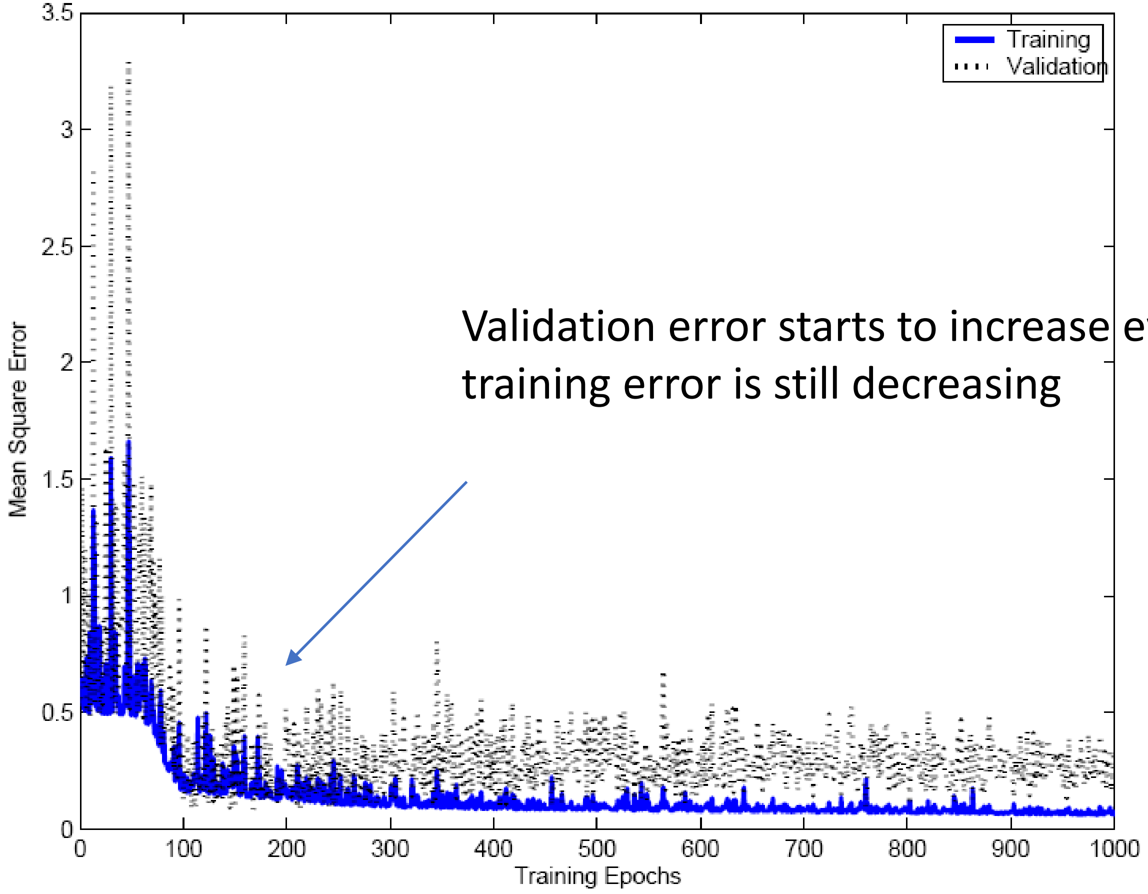
MLP:

d inputs
H hidden units
K outputs



Number of weights:
$H(d+1)+(H+1)K$

Overtraining



Validation error starts to increase even when training error is still decreasing

Most weights starts close to zero
Few jumps get updated
If keep training, those close to zeros will also be updated
(adding complexity to the model)

Learning Hidden Representations

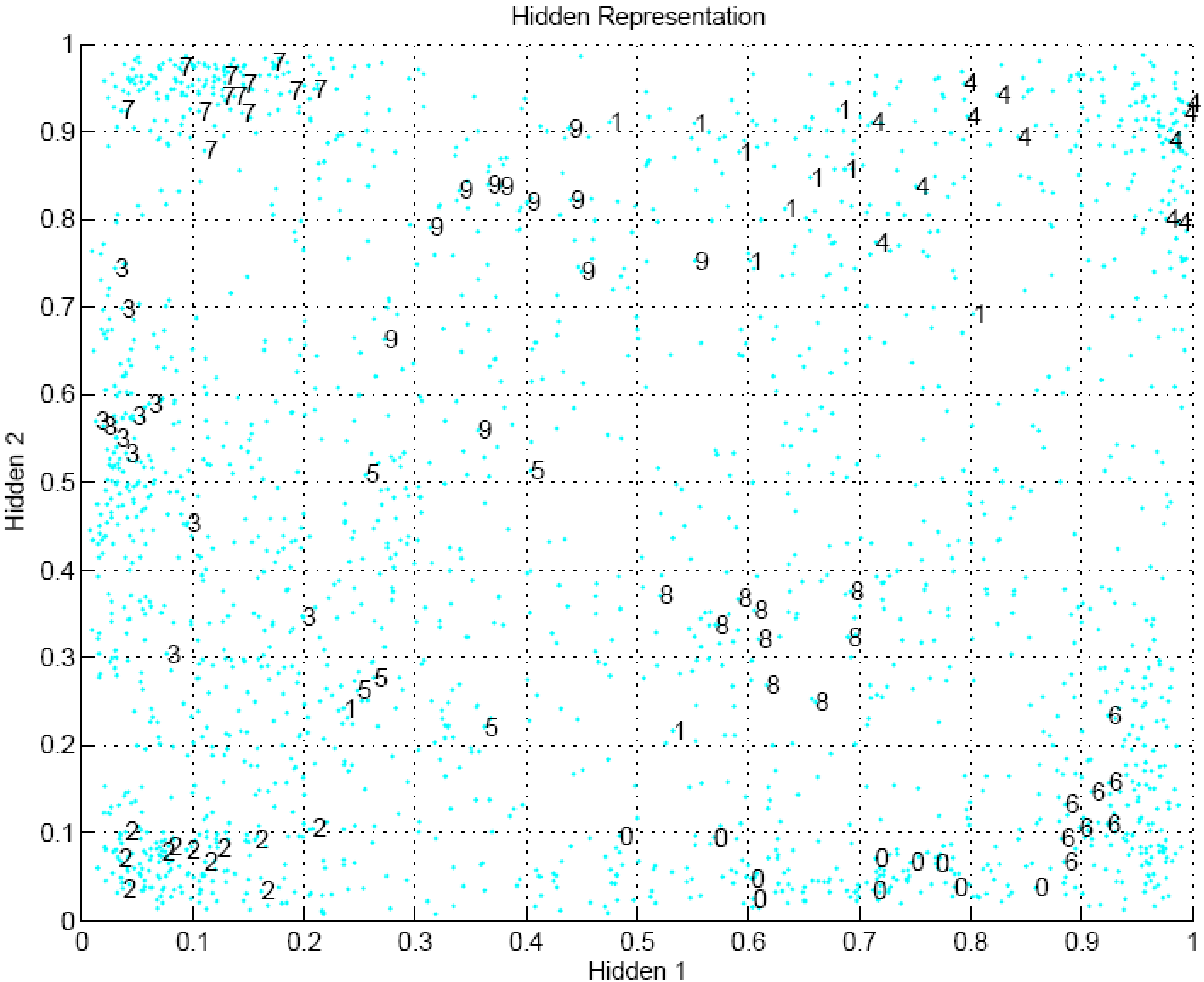
- MLP is a generalized linear model where hidden units are the nonlinear basis functions:

$$y = \sum_{h=1}^{H} v_h \phi(\boldsymbol{x} | \boldsymbol{w}_h)$$

where

$$\phi(\boldsymbol{x} | \boldsymbol{w}_h) \equiv \text{sigmoid}(\boldsymbol{w}_h^T \boldsymbol{x})$$

- The advantage is that the basis function parameters can also be learned from data.
- The hidden units, z_h, learn a *code/embedding*, a representation in the hidden space
  - If H space is < original d dimension: dimension reduction
- Transfer learning: Use code in another task
- Semisupervised learning: Transfer from an unsupervised to supervised problem



100 data points

64 inputs
2 hidden units

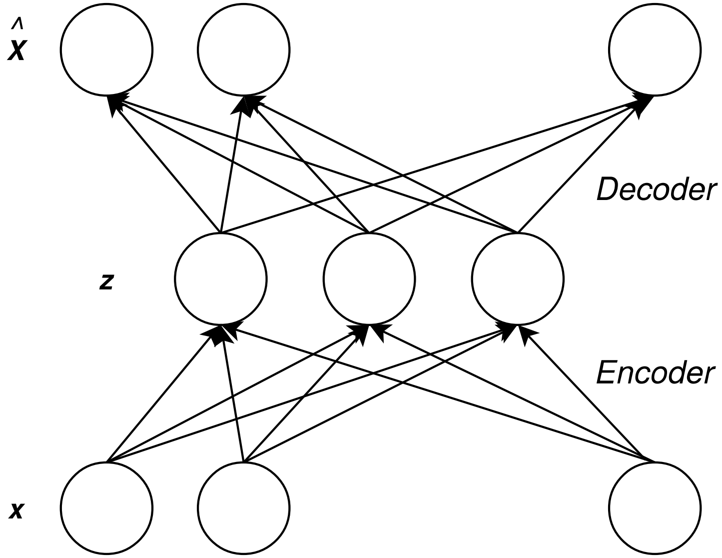
10 output

Autoencoders

An MLP structure

Equal number of input and ouput

Dimension reduction



$$\hat{\mathbf{x}}^t = \mathrm{Dec}(\mathbf{z}^t|\mathbf{V})$$

$$\mathbf{z}^t = \mathrm{Enc}(\mathbf{x}^t|\mathbf{W})$$

$$E(\mathbf{W},\mathbf{V}|\mathcal{X}) = \sum_t \|\mathbf{x}^t - \hat{\mathbf{x}}^t\|^2 = \sum_t \|\mathbf{x}^t - \mathrm{Dec}(\mathrm{Enc}(\mathbf{x}^t|\mathbf{W})|\mathbf{V})\|^2$$

- Variants: Denoising, sparse autoencoders