CSCI 5521: Introduction to Machine Learning (Spring 2020)

Multilayer Perceptron

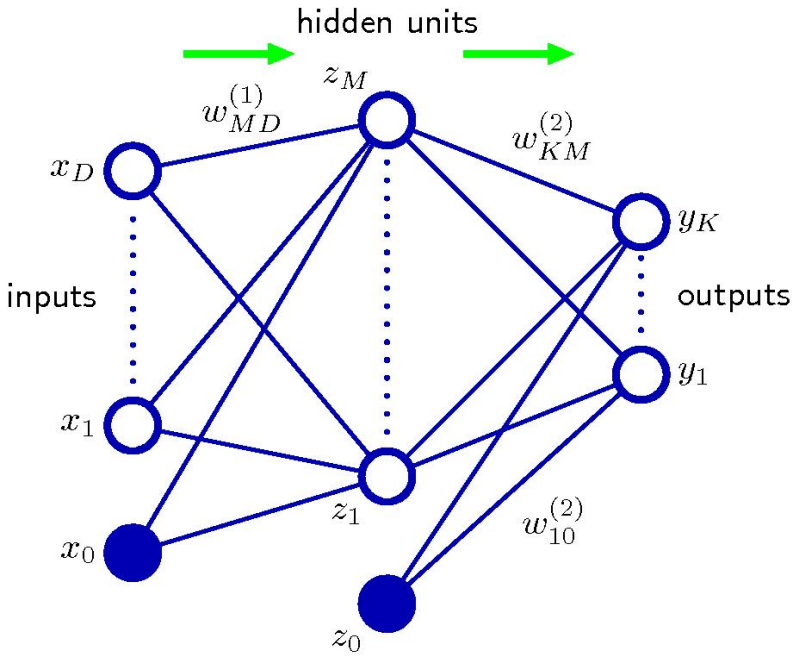
Rui Kuang

Department of Computer Science and Engineering

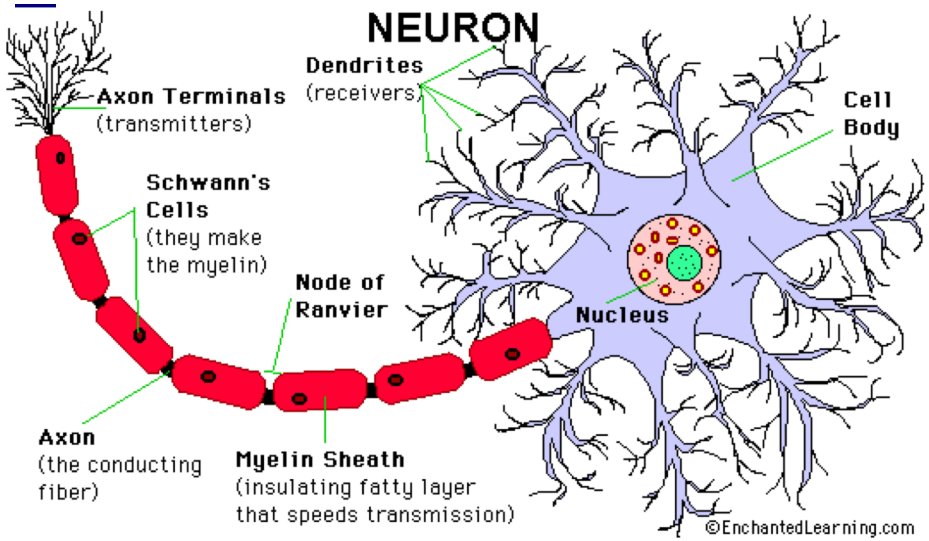
University of Minnesota

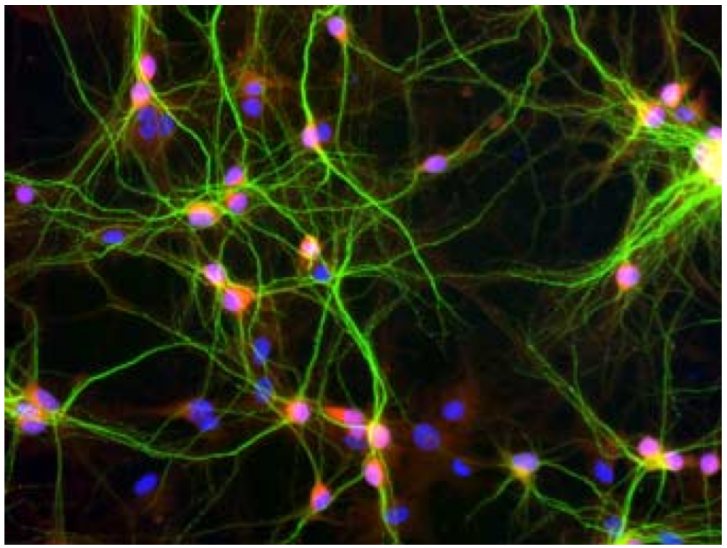
Artificial Neural Networks

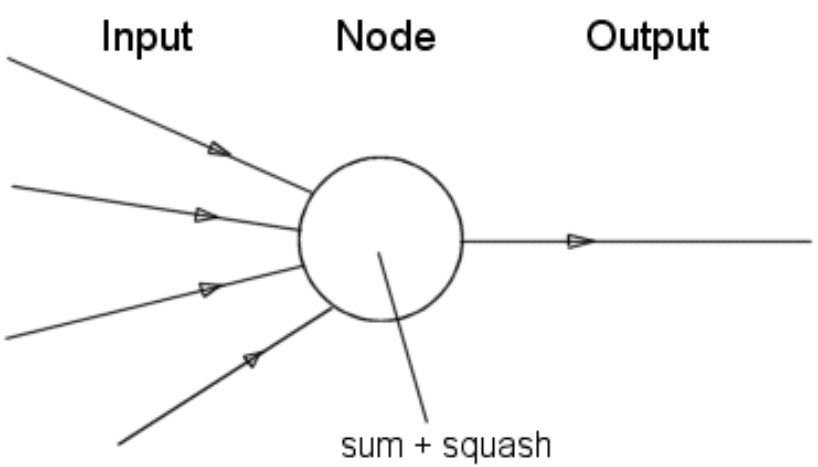
- Network models with processing units (variables) with connections (linear coefficients) between them.
- Motivated by human brain for information processing.
- Structures of hidden units incorporte prior knowledge.
- Basic units are each perceptron.

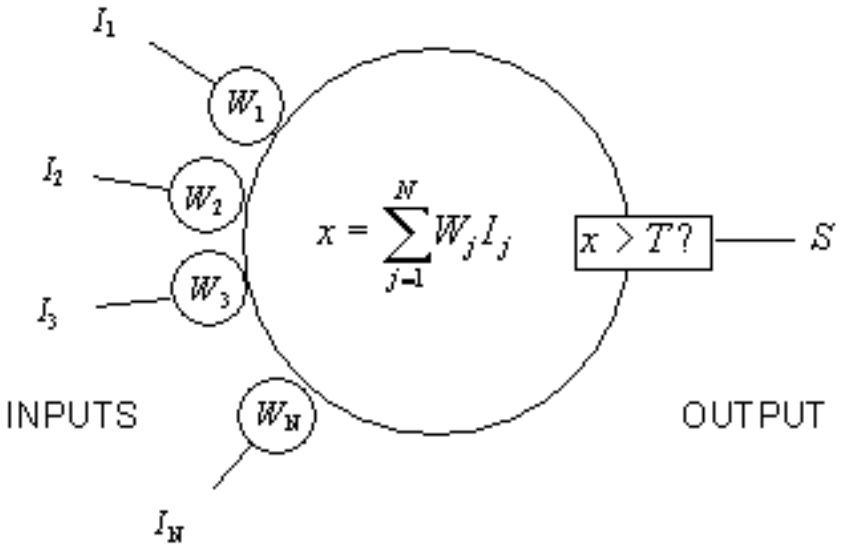


Neurone vs. Node

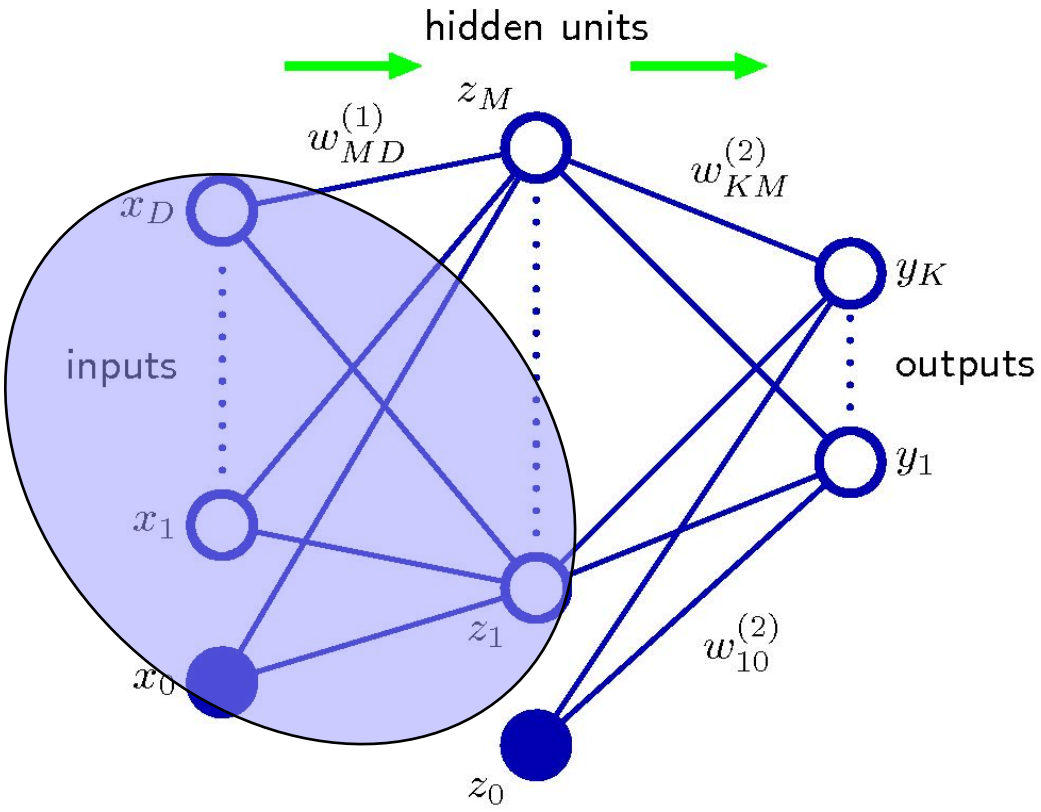






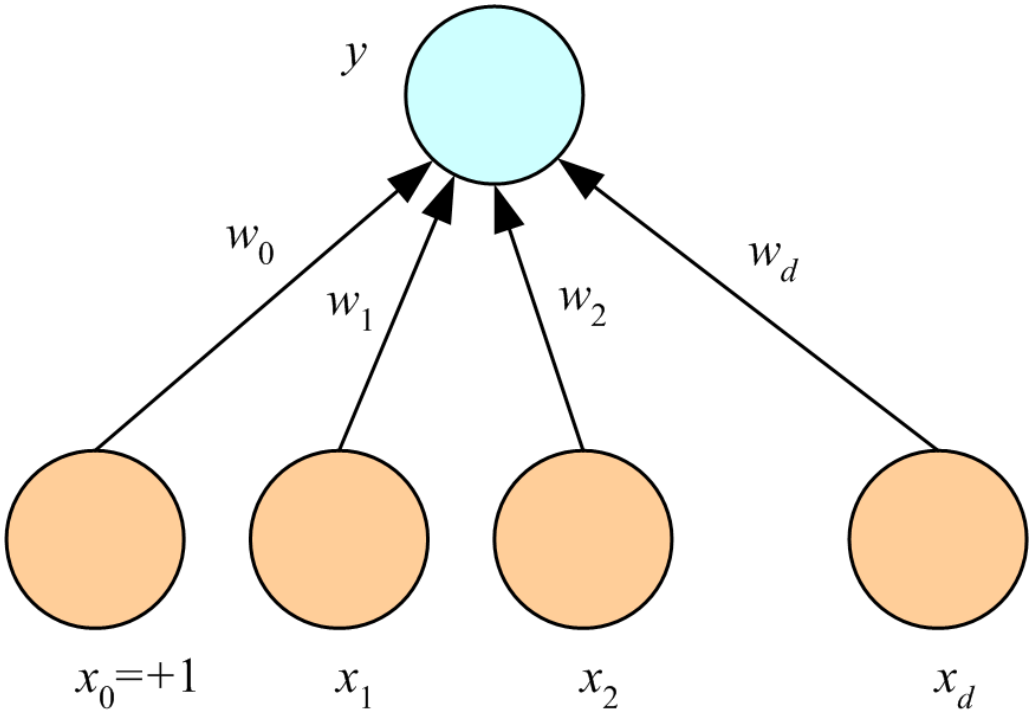


Artificial Neural Networks



- Basic processing units are called perceptrons.
- The perceptrons are connected to form multilayers.
- When evaluating the likelihood, errors are propagated through the network.

Perceptron



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

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

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

(Rosenblatt, 1962)

Perceptron vs Gradient Descent

- The same update rule in batch model or online model.

Gradient Decent :

$\mathbf{w}=0$;

Repeat until converge

$$y^t = sigmoid(\mathbf{w}^T\mathbf{x}^t);$$

$$\mathbf{w} = \mathbf{w} + \eta\mathbf{X}\left(\mathbf{r}-\mathbf{y}\right)^T;$$

Perceptron :

$\mathbf{w}=0$;

Repeat until converge

for each example $(\mathbf{x}^t, \mathbf{r}^t)$:

$$y^t = sigmoid(\mathbf{w}^T\mathbf{x}^t);$$

$$\mathbf{w} = \mathbf{w} + \eta\left(r^t - y^t\right)\mathbf{x}^t;$$

Perceptron Training

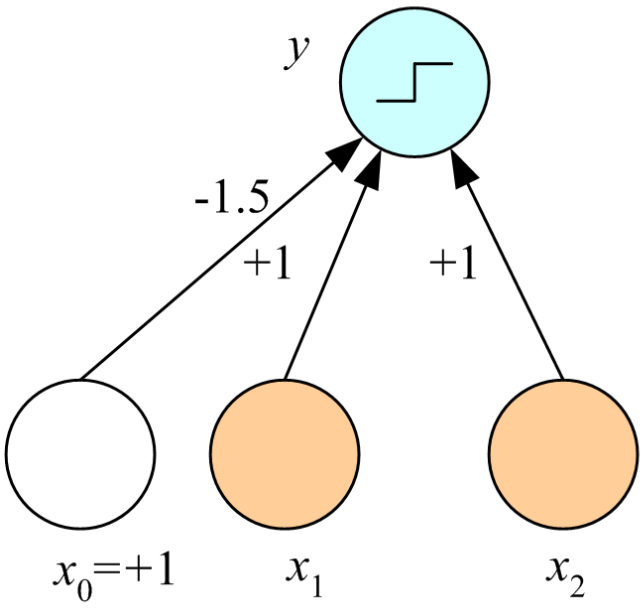
- Online training: instances seen one by one. (vs batch learning):
  - No need to store the whole sample
  - Problem may change in time
  - Wear and degradation in system components
- Stochastic gradient-descent: Update after a single pattern (or a subset of samples)

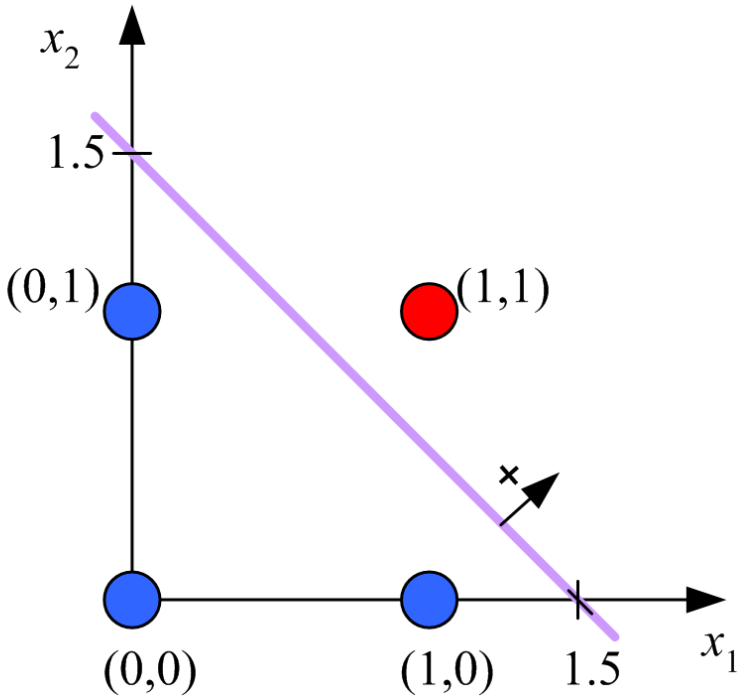
$$\Delta \mathbf{w}^t = \eta\left(r^t - y^t\right)\mathbf{x}^t$$

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

Learning Boolean AND

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



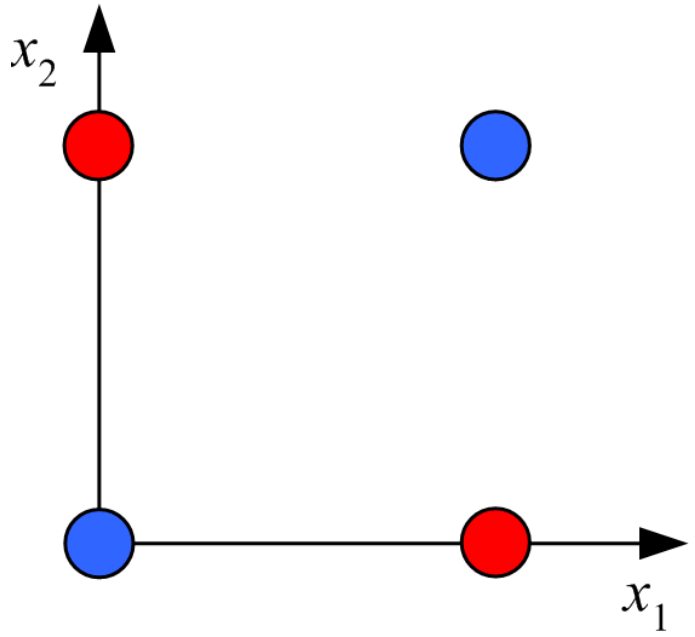


$$y = step(w^T x) = step(w_0 + w_1 x_1 + w_2 x_2)$$

$$step(a) = \begin{cases} 1 & a > 0 \\ 0 & a \le 0 \end{cases}$$

XOR

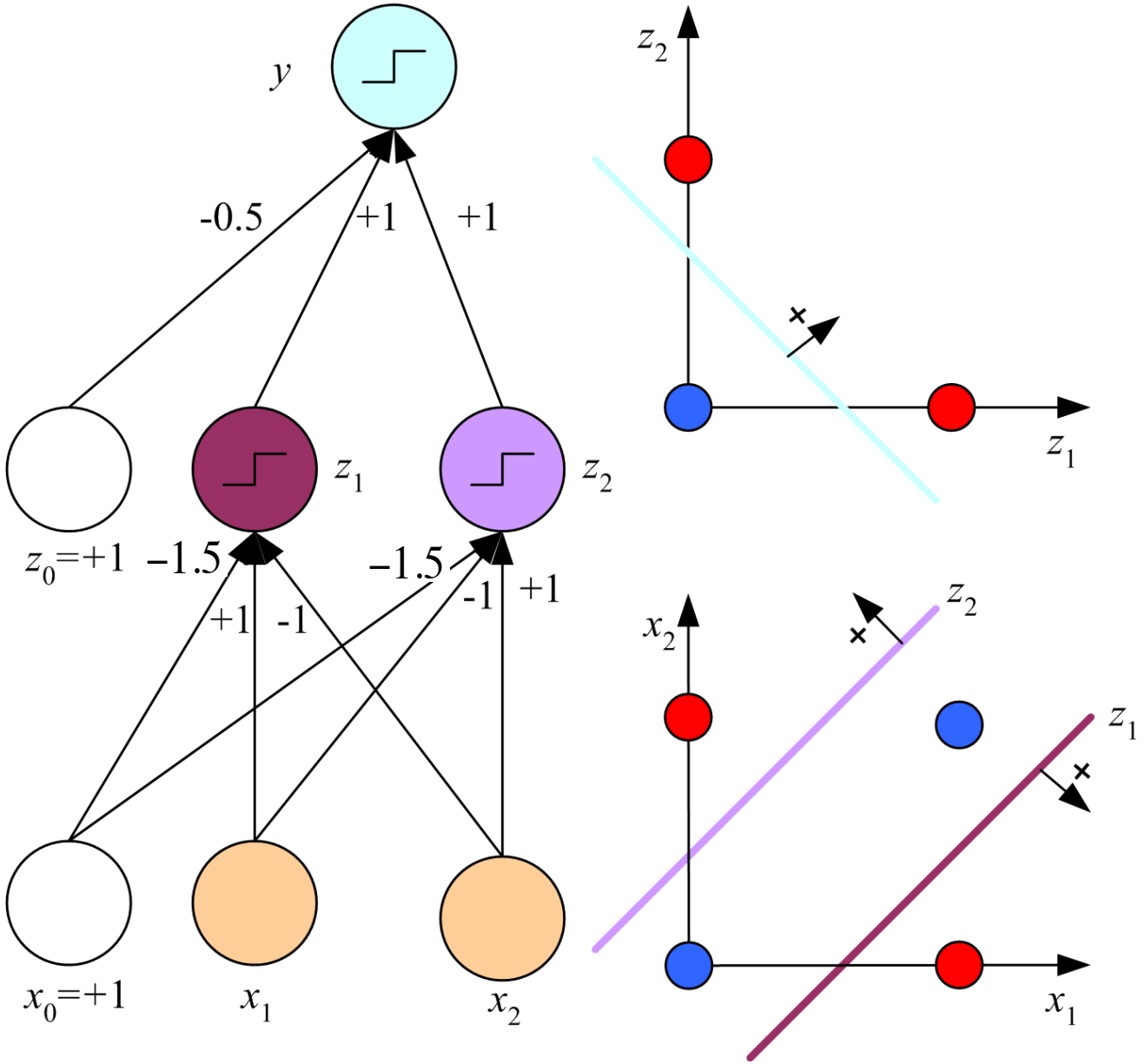
x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0



- No w_0, w_1, w_2 satisfy:

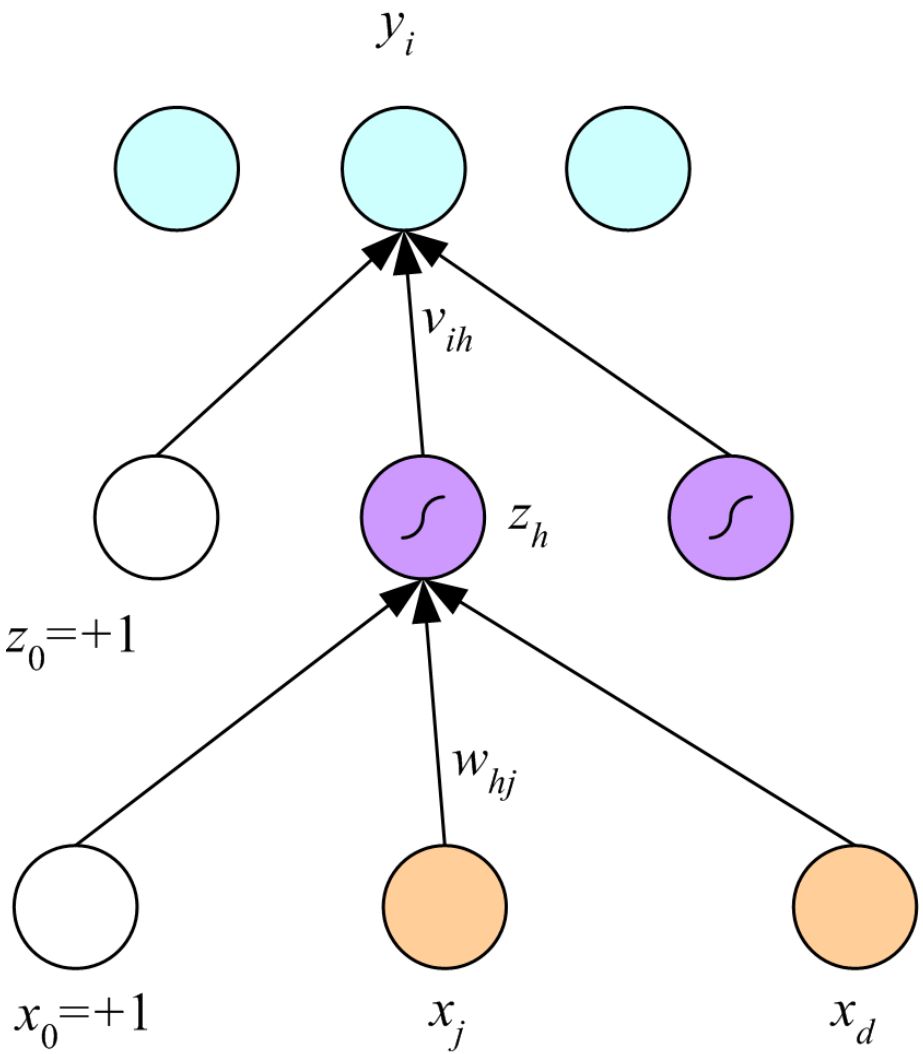
$$\begin{aligned} & & w_0 & \le 0 \\ & w_2 + & w_0 & > 0 \\ w_1 + & & w_0 & > 0 \\ w_1 + & w_2 + & w_0 & \le 0 \end{aligned}$$

No w is a solution!



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

Multilayer Perceptrons

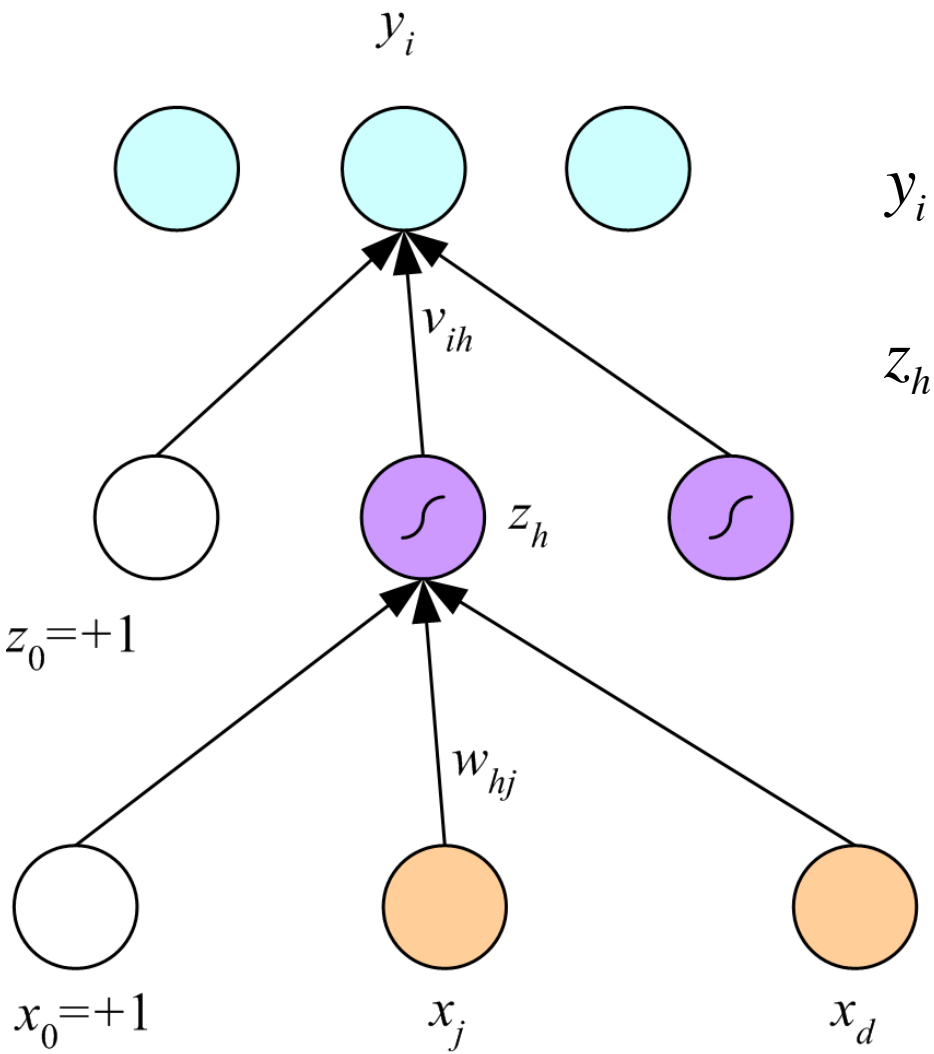


$$y_i = \mathbf{v}_i^T \mathbf{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

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

(Rumelhart et al., 1986)

Backpropagation



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

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

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

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

Regression

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

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

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

Backward

Forward

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

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

$\boldsymbol{x}$

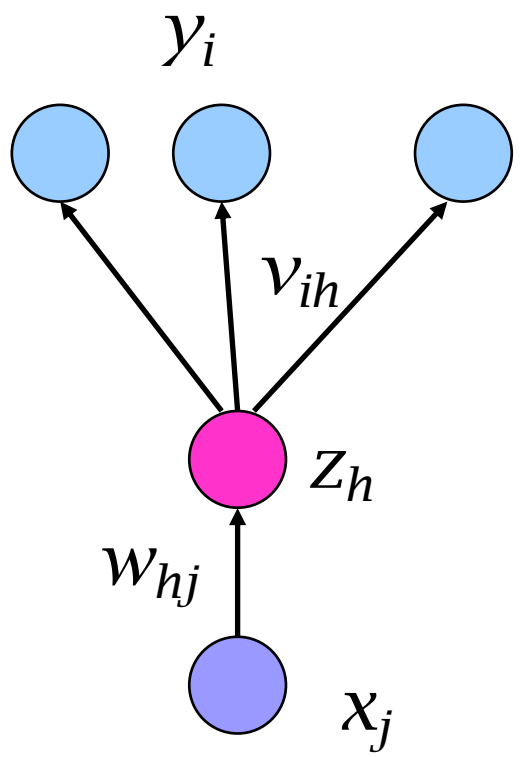
Regression with Multiple Outputs

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

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

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

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



Multi-regression

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

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

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

Backward

Forward

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

$$\mathbf{x}$$

$$\begin{aligned}
\Delta w_{hj} &= -\eta \frac{\partial E}{\partial w_{hj}} \\
&= -\eta \sum_t \sum_i \frac{\partial E}{\partial y_i^t} \frac{\partial y_i^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}} \\
&= -\eta \sum_t \sum_i -\left(r_i^t - y_i^t\right) v_{ih} z_h^t \left(1 - z_h^t\right) x_j^t \\
&= \eta \sum_t \sum_i \left(r_i^t - y_i^t\right) v_{ih} z_h^t \left(1 - z_h^t\right) x_j^t
\end{aligned}$$

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 $h = 1, \ldots, H$

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

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 $h = 1, \ldots, H$

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

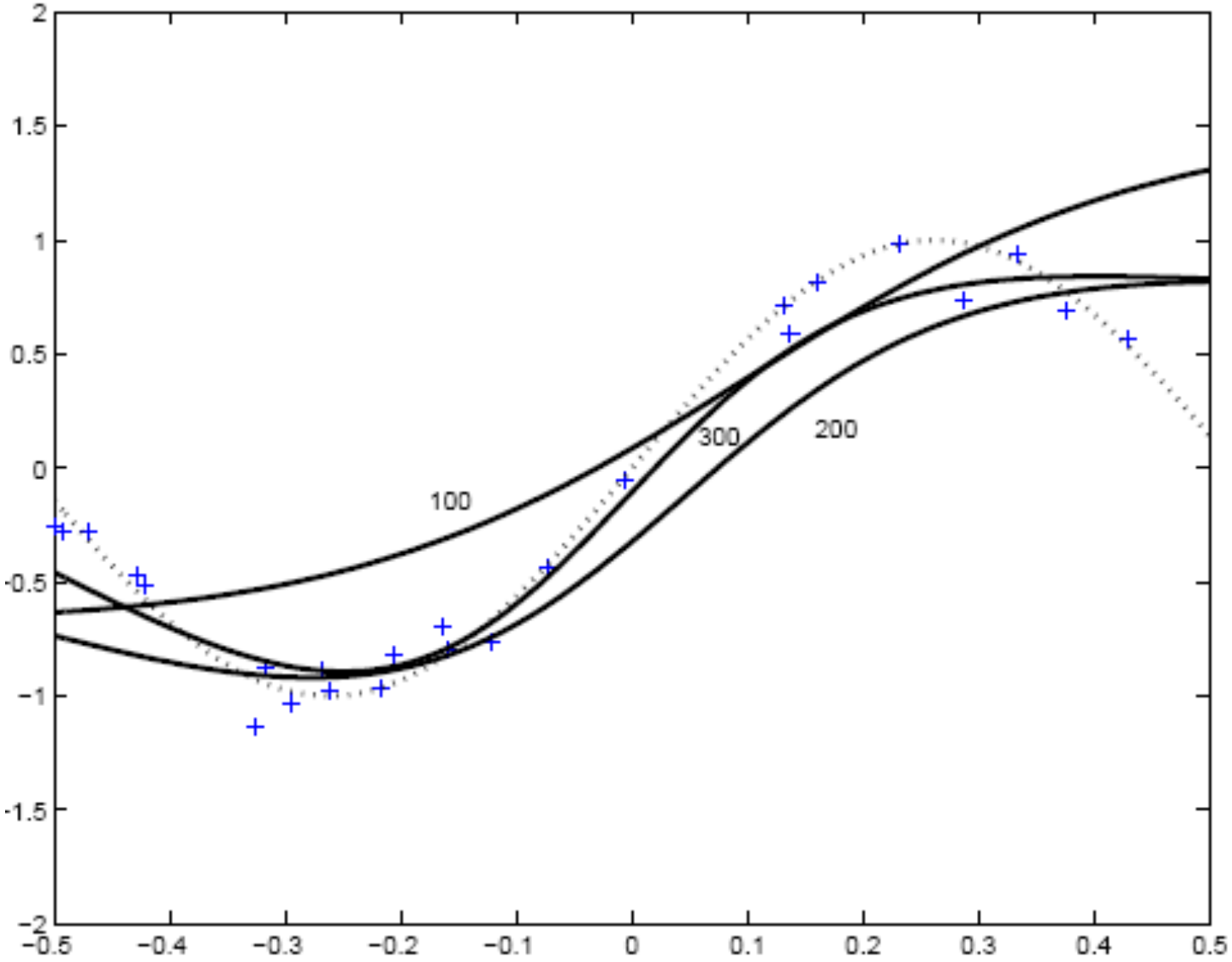
For $i = 1, \ldots, K$

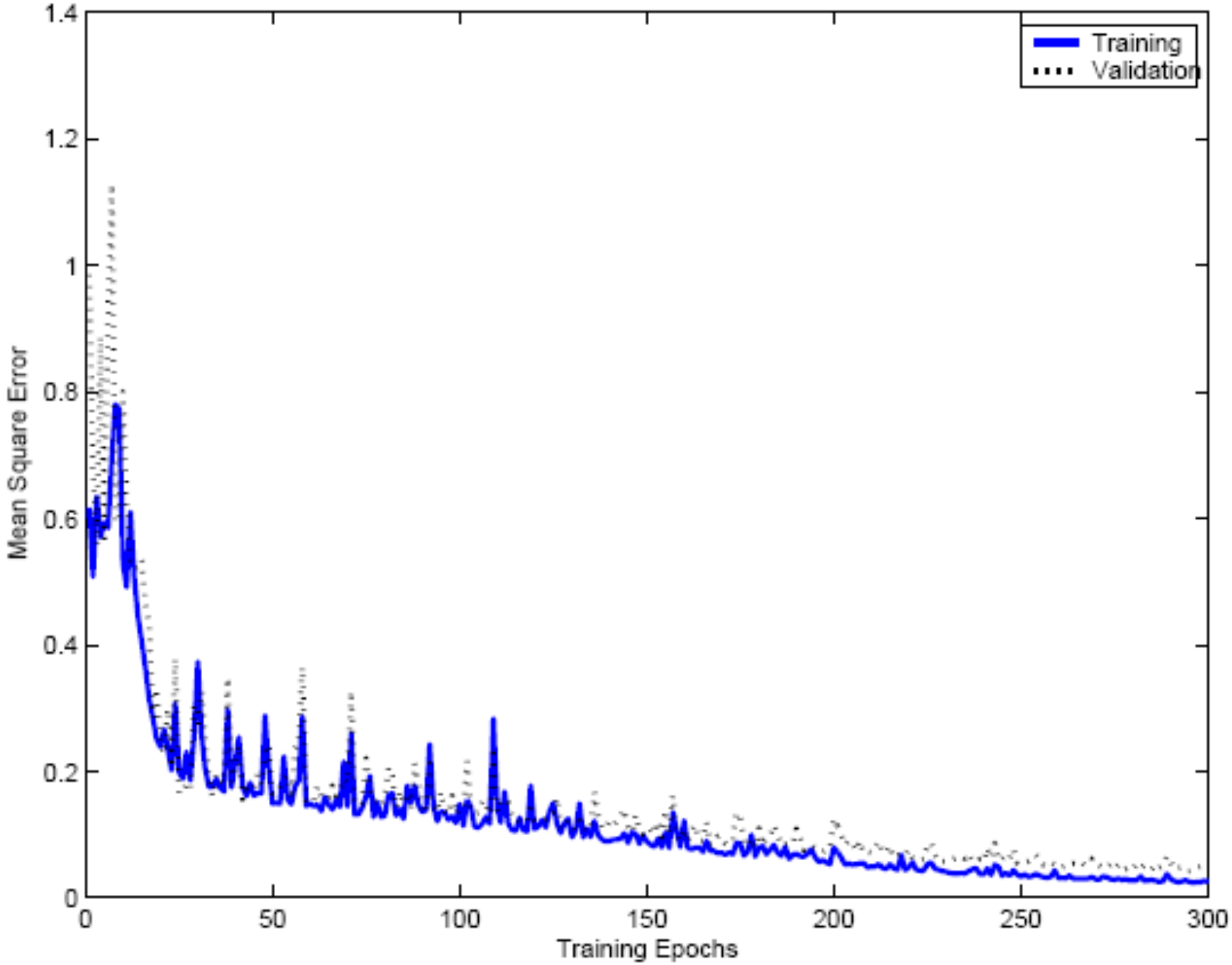
$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \Delta \boldsymbol{v}_i$

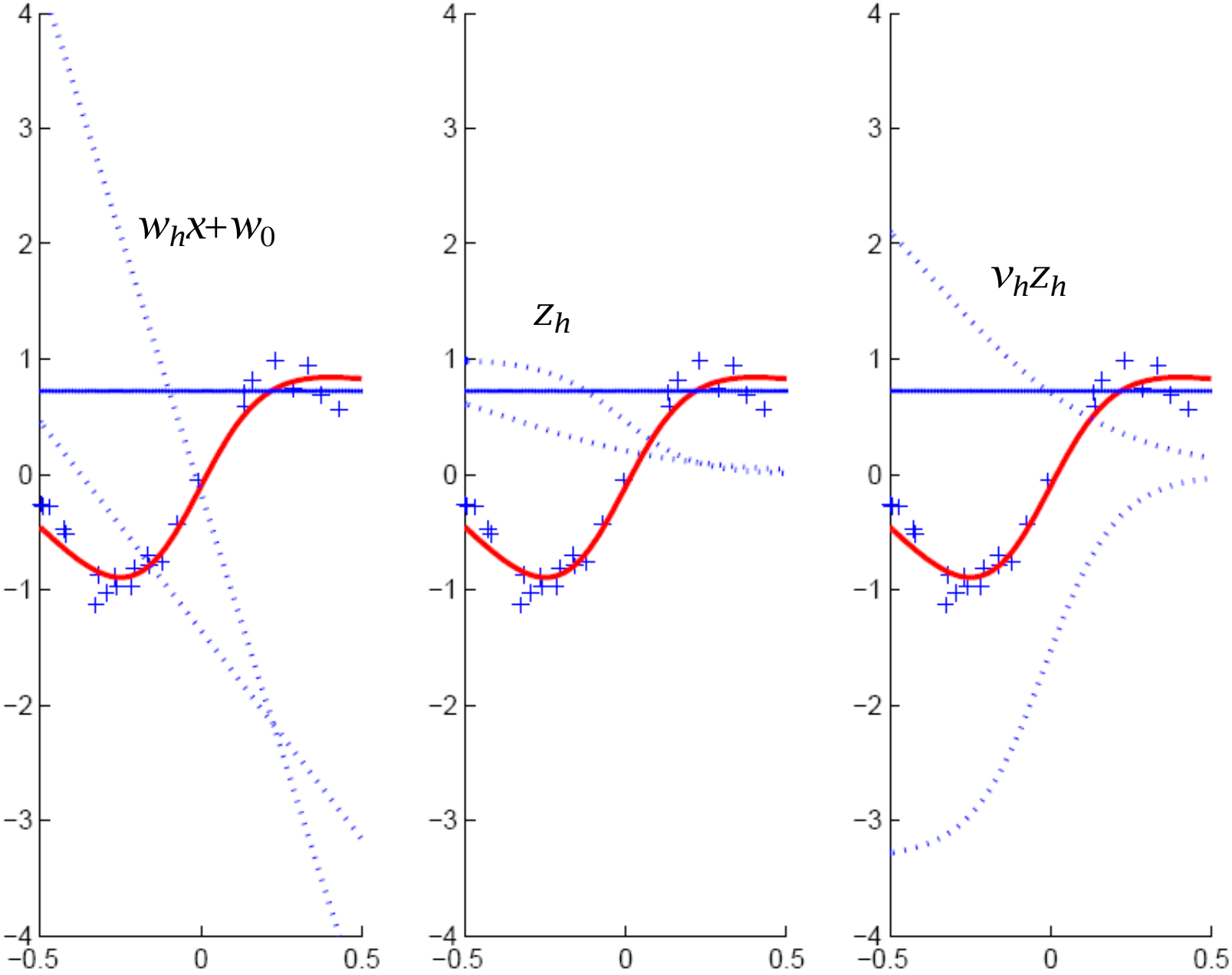
For $h = 1, \ldots, H$

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

Until convergence

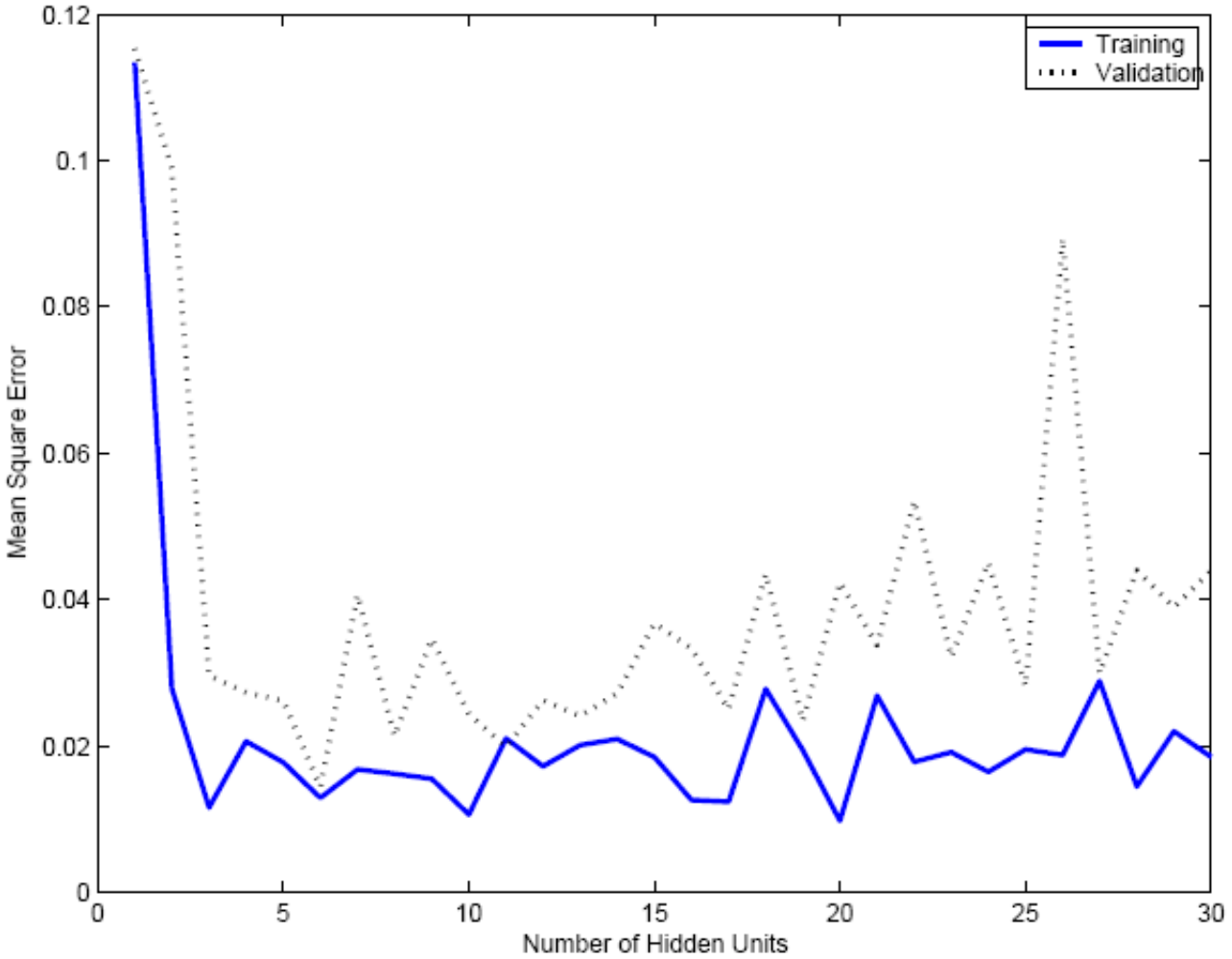

100
300
200

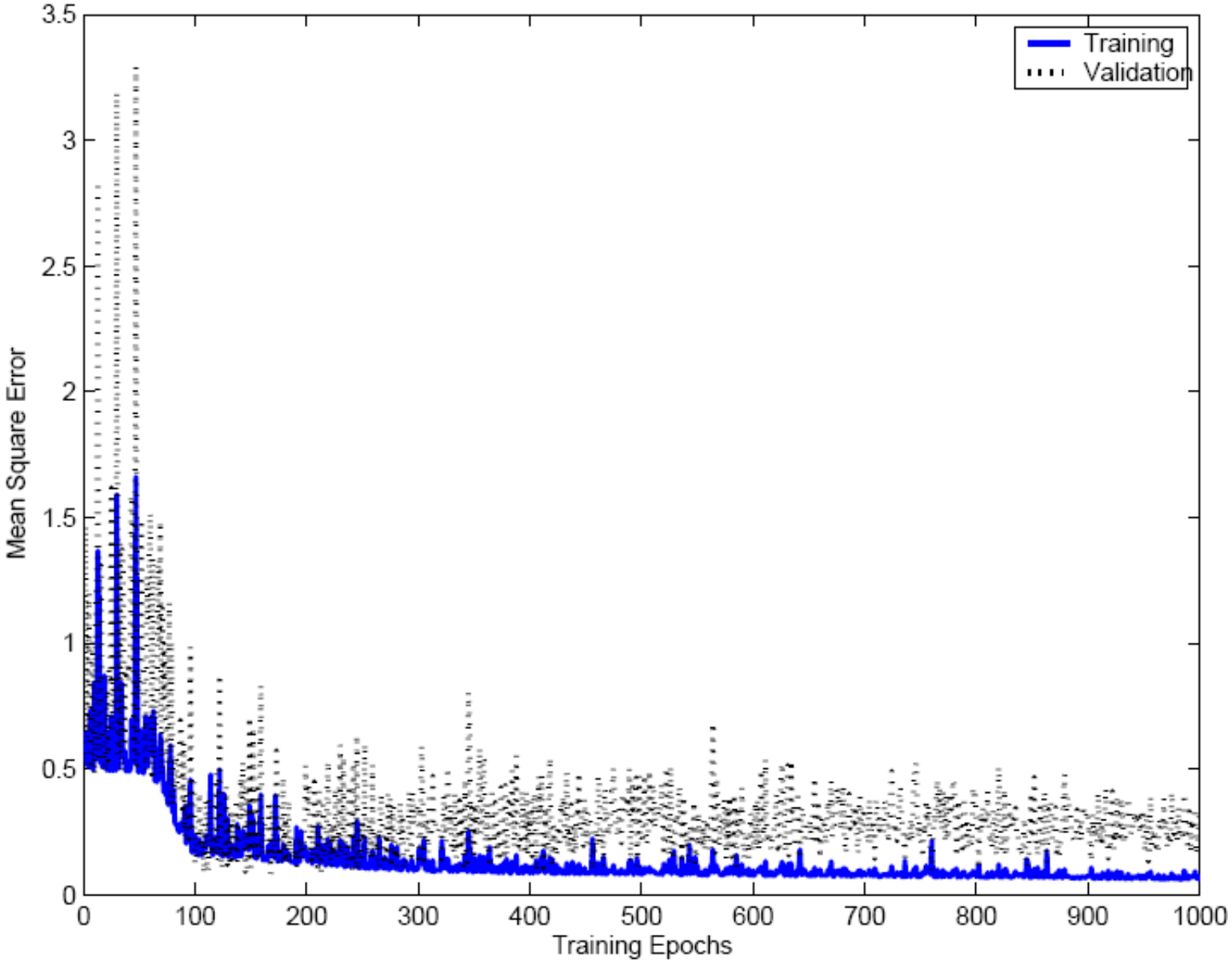

Training
Validation
Mean Square Error
Training Epochs
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
50
100
150
200
250
300


$w_h x + w_0$
z_h
$v_h z_h$
4
3
2
1
0
−1
−2
−3
−4
−0.5
0
0.5

Overfitting/Overtraining

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




Training
Validation
Mean Square Error
Training Epochs
0
0.5
1
1.5
2
2.5
3
3.5
0
100
200
300
400
500
600
700
800
900
1000

Two-Class Classification

- One sigmoid output y^t for $P(C_1|\boldsymbol{x}^t)$ and $P(C_2|\boldsymbol{x}^t) \equiv 1-y^t$

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

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

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

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

Classification

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

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

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

Backward

Forward

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

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

$$= -\eta \sum_t \frac{\partial E}{\partial y^t} \frac{\partial y^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}}$$

$$= -\eta \sum_t -\left(r^t - y^t\right) v_h z_h^t \left(1 - z_h^t\right) x_j^t$$

$$= \eta \sum_t \left(r^t - y^t\right) v_h z_h^t \left(1 - z_h^t\right) x_j^t$$

$\boldsymbol{x}$

K>2 Classes

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

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

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

(See softmax logistic regression)

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

Multi-classification

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

$$o_i^t = \sum_{h=1}^{H} v_{ih} z_h^t + v_{i0} \qquad y_i^t = \frac{\exp o_i^t}{\sum_k \exp o_k^t}$$

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

Backward

Forward

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

$$\mathbf{x}$$

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

$$= -\eta \sum_t \sum_i \frac{\partial E}{\partial y_i^t} \frac{\partial y_i^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}}$$

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

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

Multiple Hidden Layers

- MLP with one hidden layer is a universal approximator (Hornik et al., 1989), but using multiple layers may lead to simpler networks

$$z_{1h} = \text{sigmoid}\left(\mathbf{w}_{1h}^T \mathbf{x}\right) = \text{sigmoid}\left(\sum_{j=1}^{d} w_{1hj} x_j + w_{1h0}\right), h = 1,\ldots,H_1$$

$$z_{2l} = \text{sigmoid}\left(\mathbf{w}_{2l}^T \mathbf{z}_1\right) = \text{sigmoid}\left(\sum_{h=1}^{H_1} w_{2lh} z_{1h} + w_{2l0}\right), l = 1,\ldots,H_2$$

$$y = \mathbf{v}^T \mathbf{z}_2 = \sum_{l=1}^{H_2} v_l z_{2l} + v_0$$

Improving Convergence

- Momentum

$$\Delta w_i^t = -\eta \frac{\partial E^t}{\partial w_i} + \alpha \Delta w_i^{t-1}$$

- Adaptive learning rate

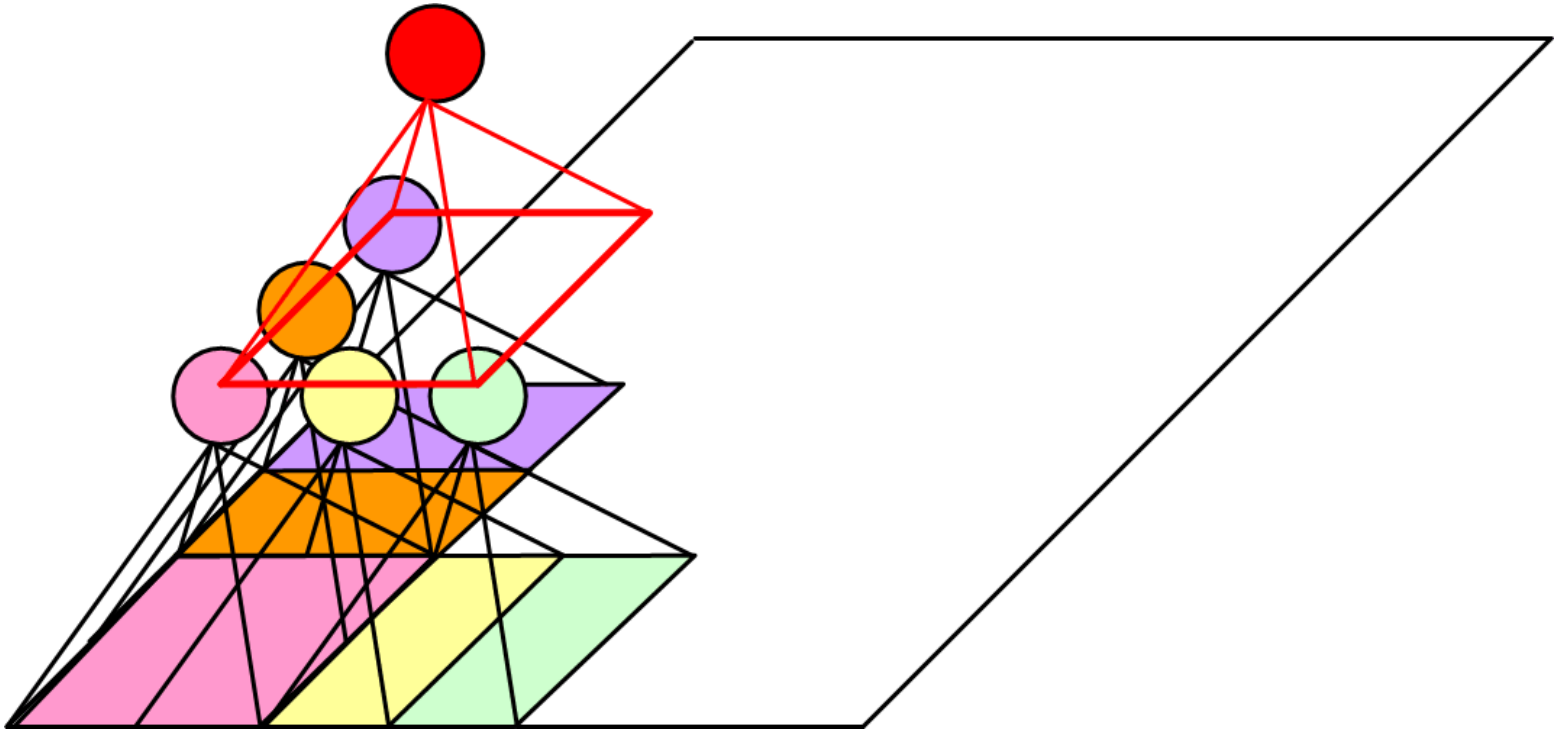
$$\Delta \eta = \begin{cases} +a & \text{if } E^{t+\tau} < E^t \\ -b\eta & \text{otherwise} \end{cases}$$

Go faster!

Oscillating!
Slow down!

Convolutional Neural Network

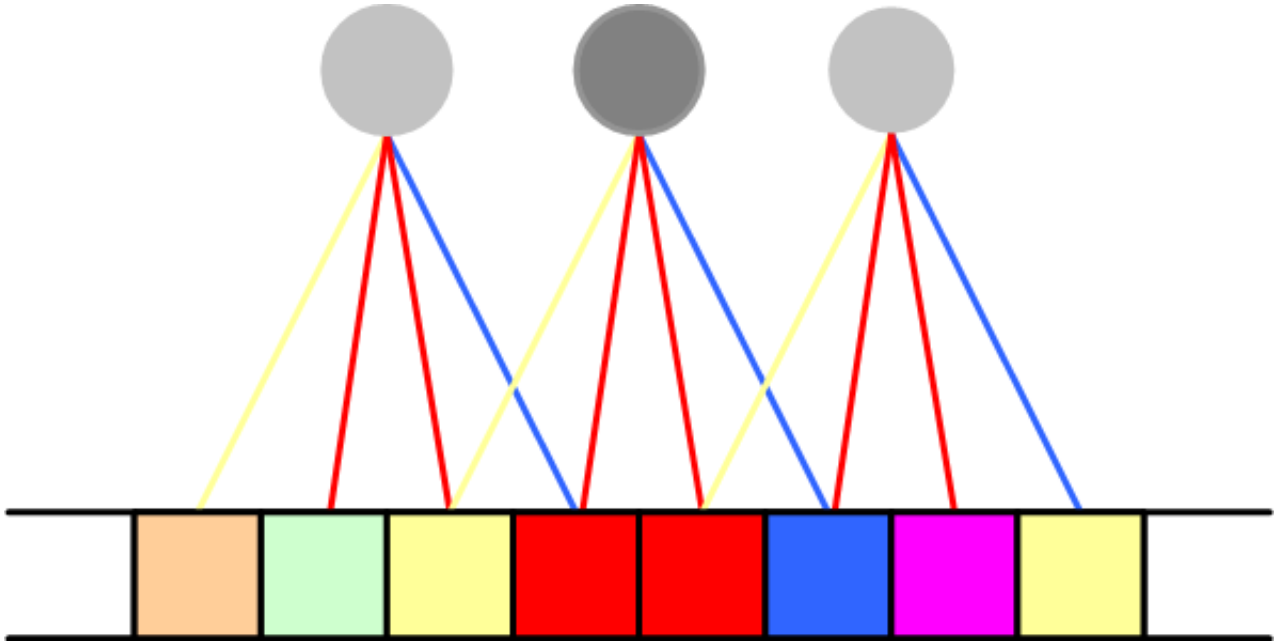
- Hidden units are connected to subsets of inputs



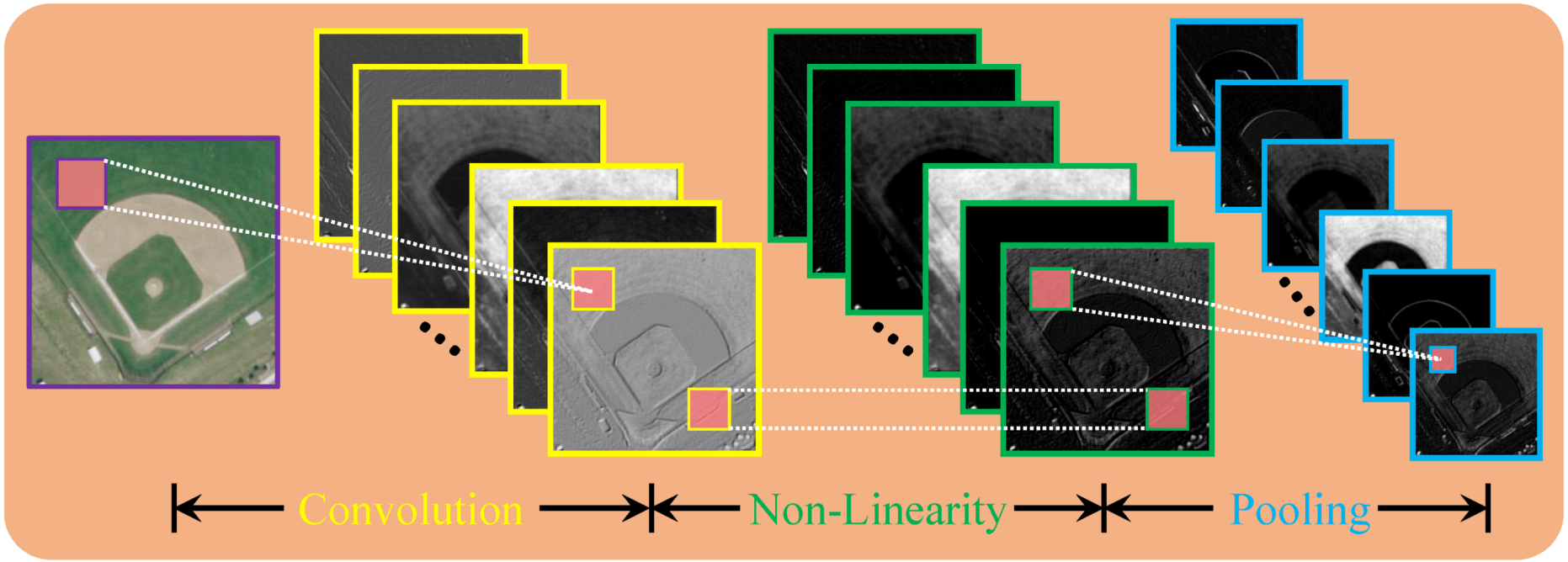
(Le Cun et al, 1989)

Weight Sharing

- Some of the weights are shared among several units.

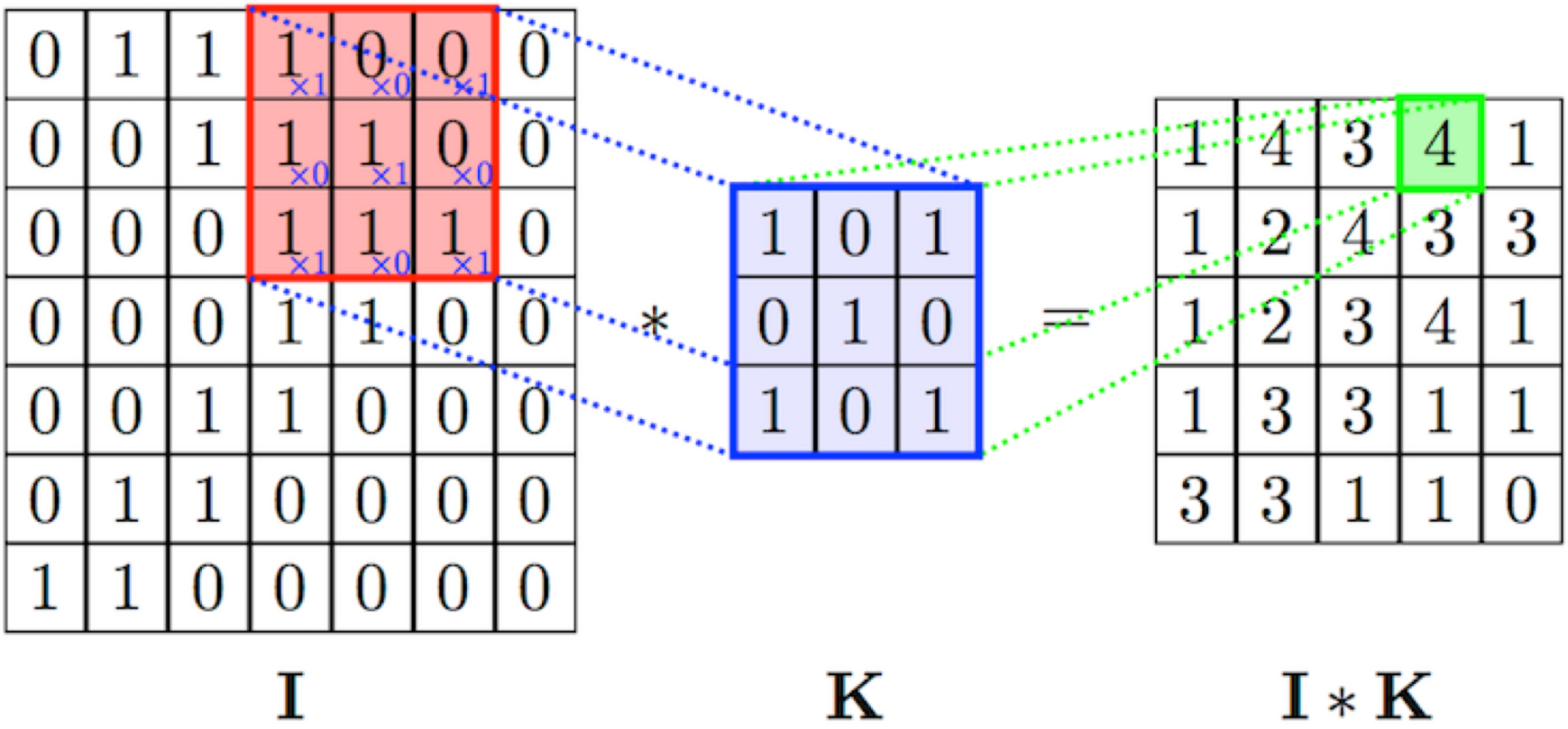


Convolution and Pooling



http://www.mdpi.com/remotesensing/remotesensing-07-14680/article_deploy/html/images/remotesensing-07-14680-g001-1024.png

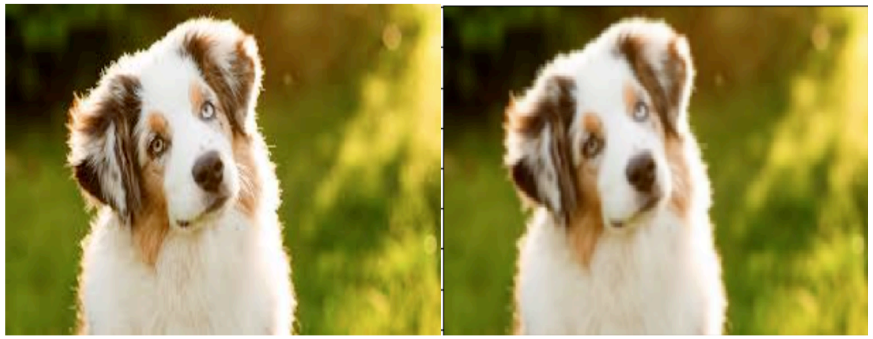
Convolution



https://deeplearning4j.org/convolutionalnets

More on Filters

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



Blurry

0	-1	0
-1	5	-1
0	-1	0

Sharpen

https://www.saama.com/blog/different-kinds-convolutional-filters/

More on Filters

-1	-2	-1
0	0	0
1	2	1

Horizontal

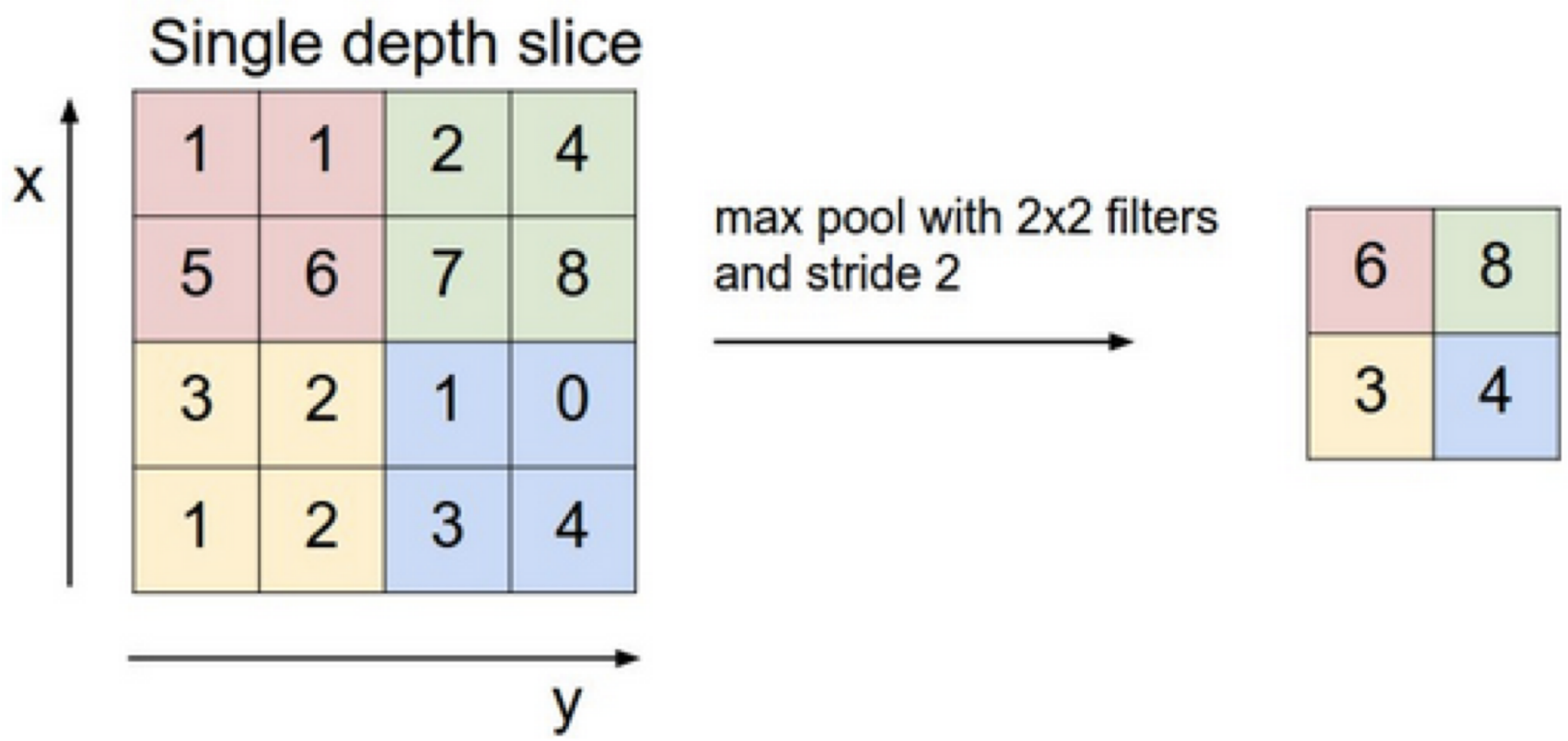
-1	0	1
-2	0	2
-1	0	1

Vertical

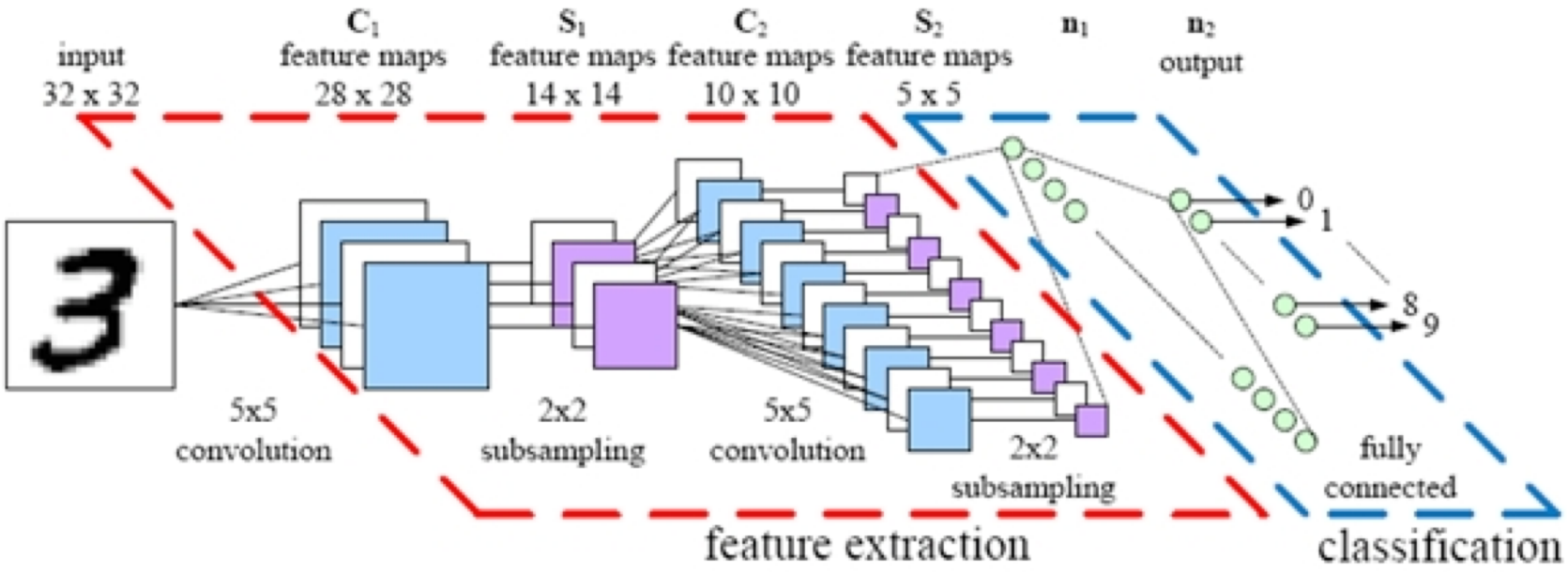
Edge detection

https://www.saama.com/blog/different-kinds-convolutional-filters/

Pooling (subsampling)



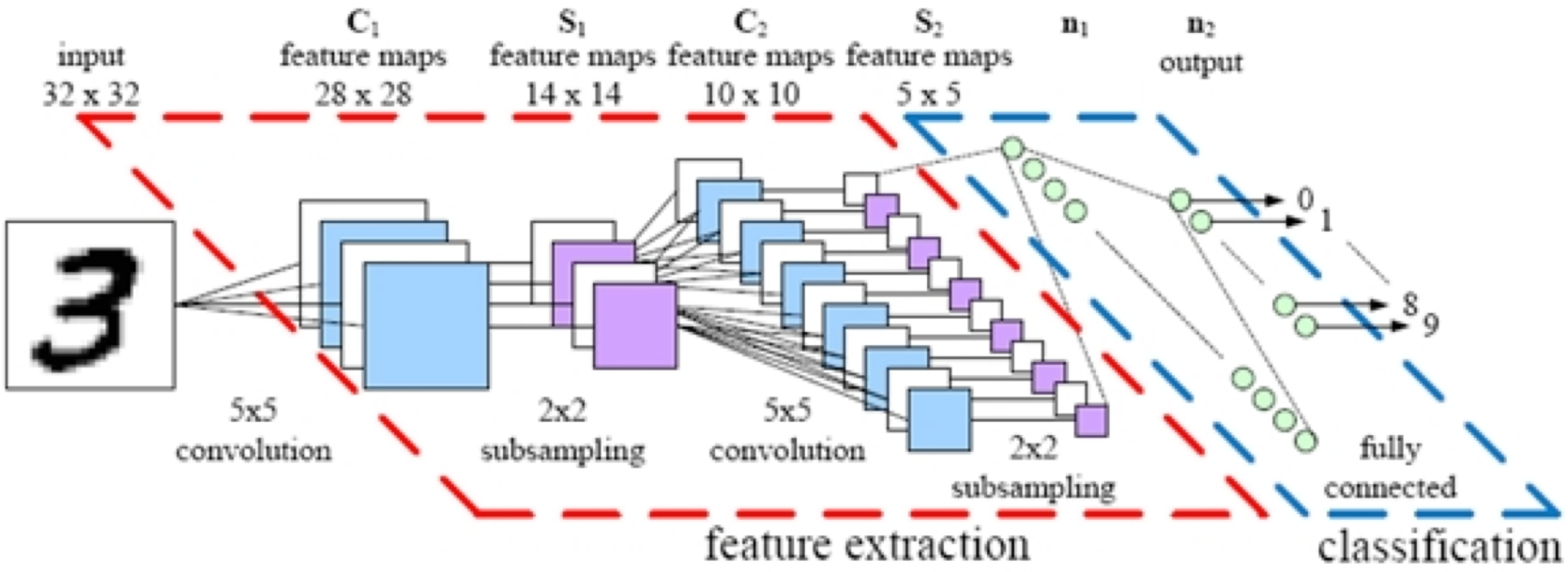
CNN Example



- Convolutional layer
  - Each unit takes subregion as input.
  - Shared weights across all the units in a feature map.
  - There are multiple feature maps in the convolutional layer.

http://www.pyimagesearch.com/2014/06/09/get-deep-learning-bandwagon-get-perspective/

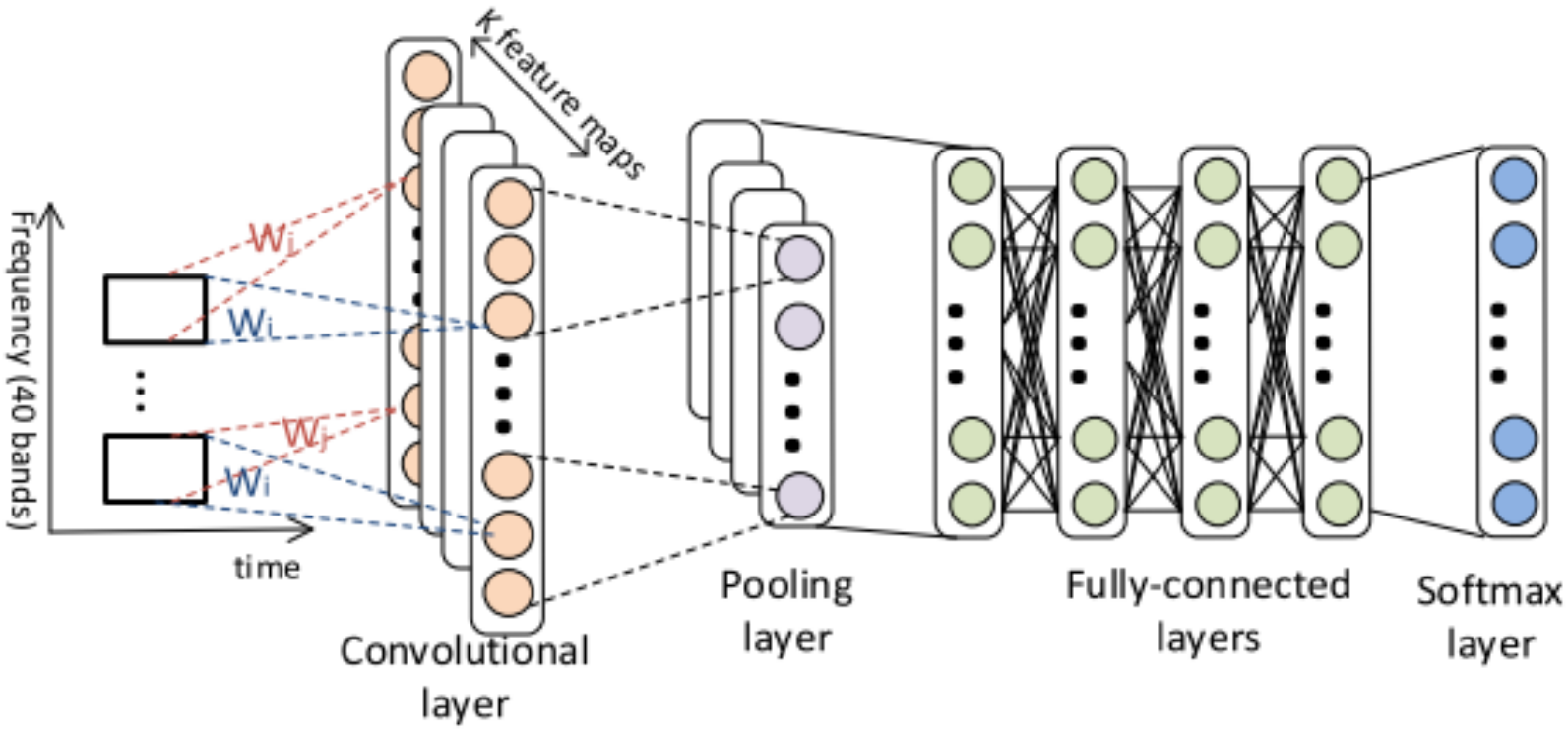
CNN Example



- Nonoverlaping subsampling layer.
  - Units in one plane connect to 2x2 region (receptive field) in a feature map.
  - Taking the mean or max of the inputs in the region.
- Final layers are fully connected NN.

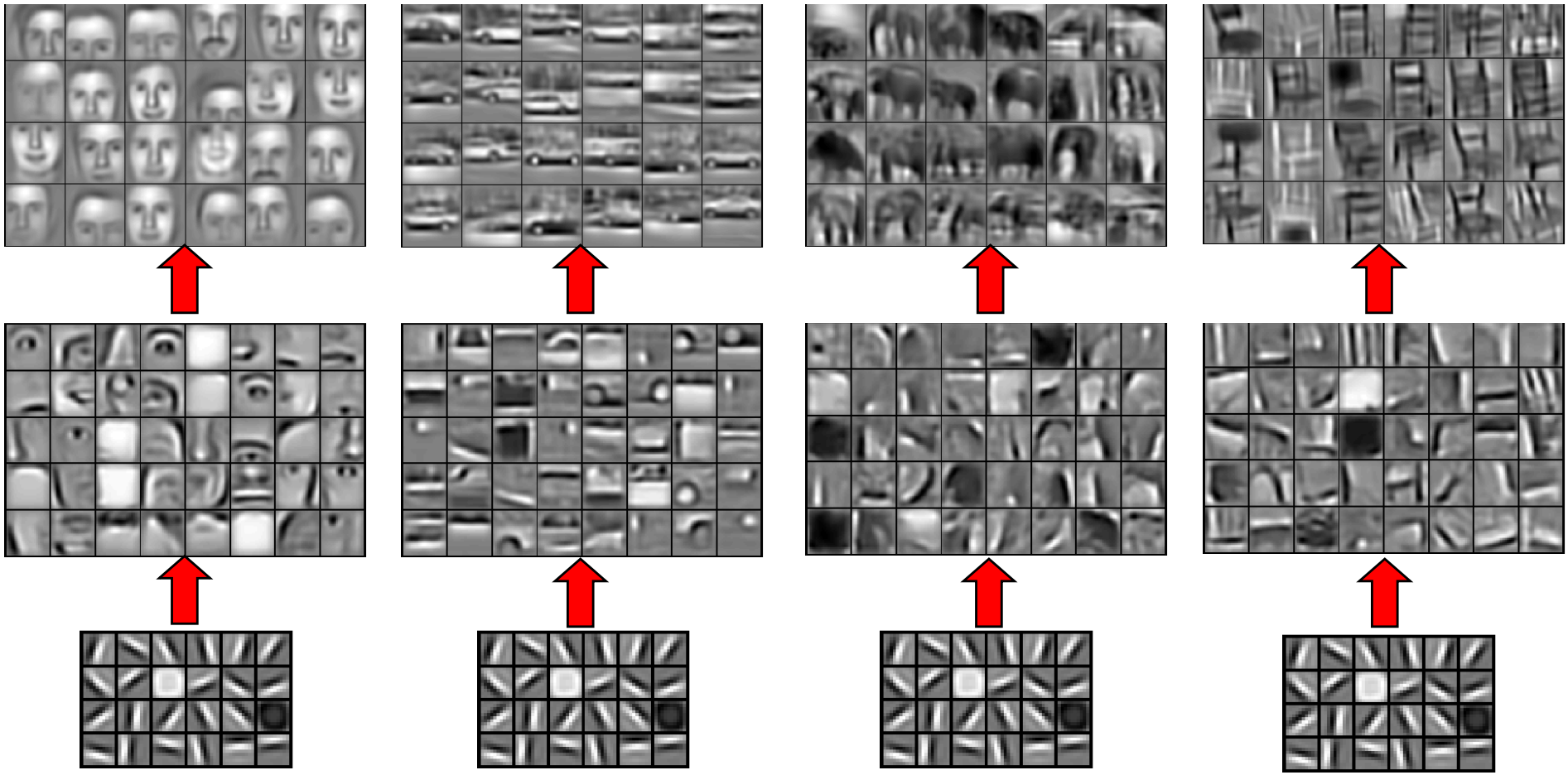
http://www.pyimagesearch.com/2014/06/09/get-deep-learning-bandwagon-get-perspective/

CNN Example



AN ANALYSIS OF CONVOLUTIONAL NEURAL NETWORKS FOR SPEECH RECOGNITION, Huang et al.

Learning of Object Parts



http://ufldl.stanford.edu/eccv10-tutorial/

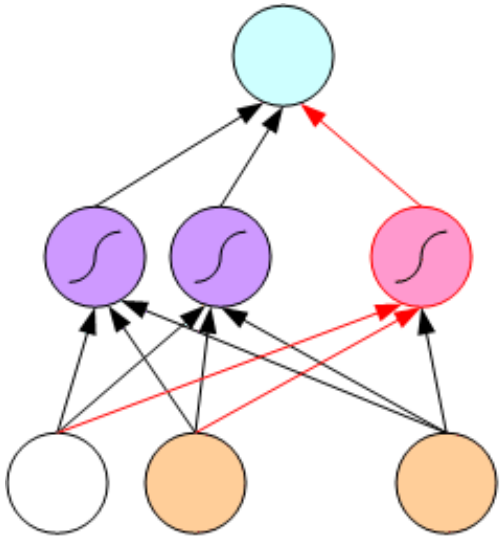
Tuning the Network Size

- Destructive
  - Weight decay: penalizing non-zero parameters
  - The same as adding additional constaints

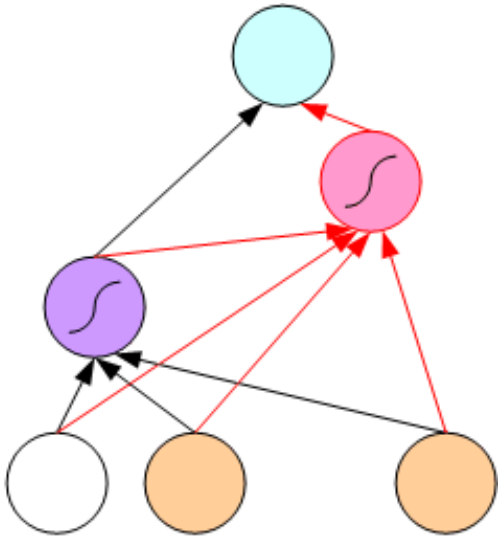
$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} - \lambda w_i$$

$$E' = E + \frac{\lambda}{2} \sum_i w_i^2$$

- Constructive
  - Growing networks: if training error is high, add more hidden units.

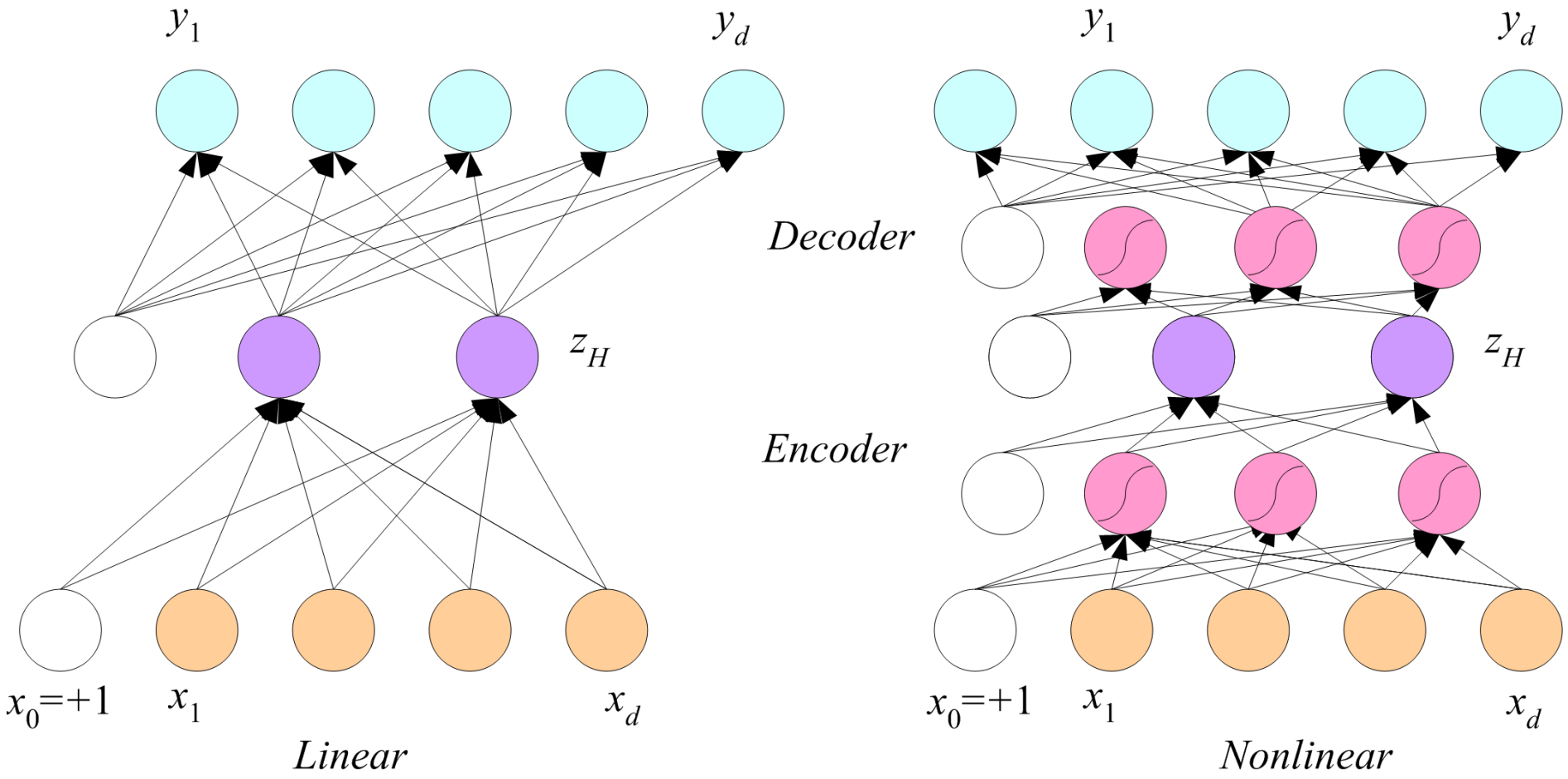


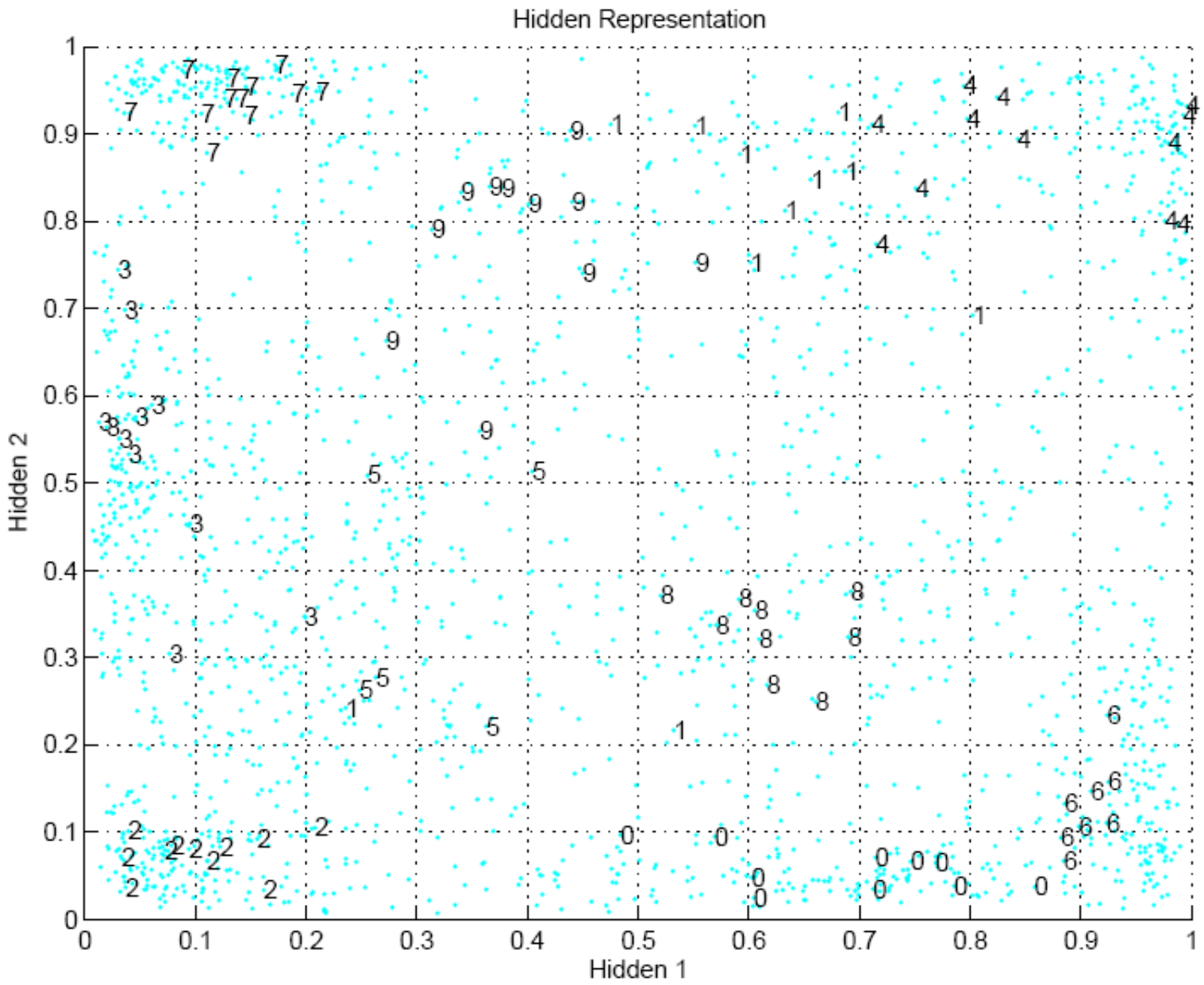
Dynamic Node Creation



Cascade Correlation

Autoencoder: Dimension Reduction

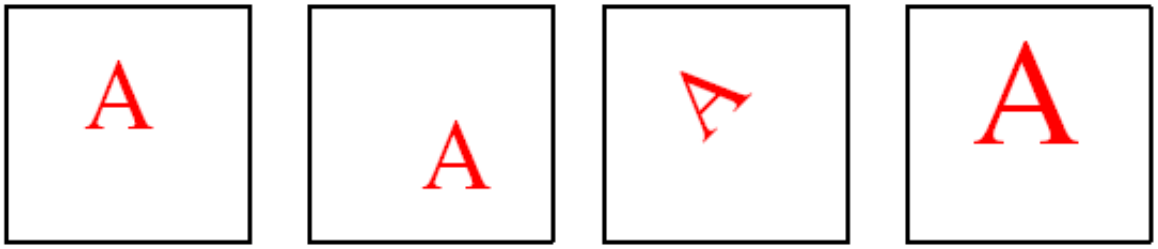



Hidden Representation
Hidden 2
Hidden 1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Hints

(Abu-Mostafa, 1995)

- Invariance to translation, rotation, size



- Virtual examples
- Augmented error: $E' = E + \lambda_h E_h$

If $\boldsymbol{x}'$ and $\boldsymbol{x}$ are the "same": $E_h = [g(x|\theta) - g(x'|\theta)]^2$