Deep Learning

Fundamentals

Machine Learning

Dongsuk Yook
Artificial Intelligence Laboratory
Korea University

Contents

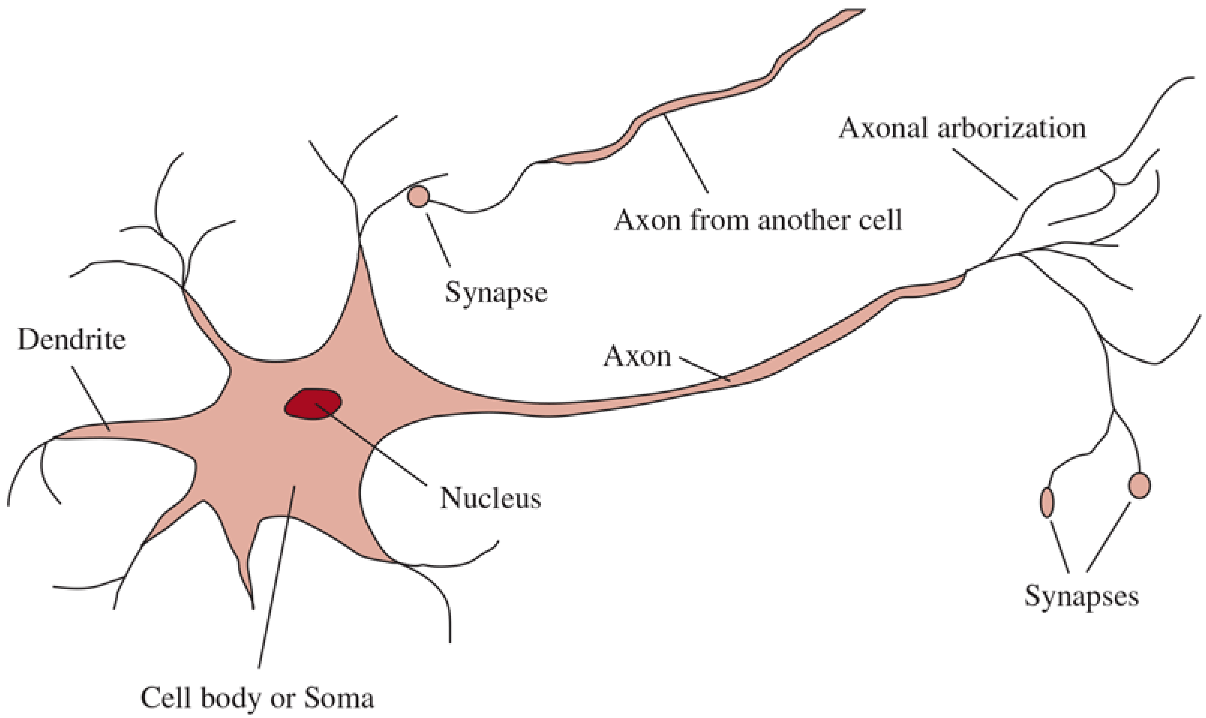
Class Objectives

- Understanding the fundamentals of deep learning
- Being able to build a basic deep neural network for a given application

Contents

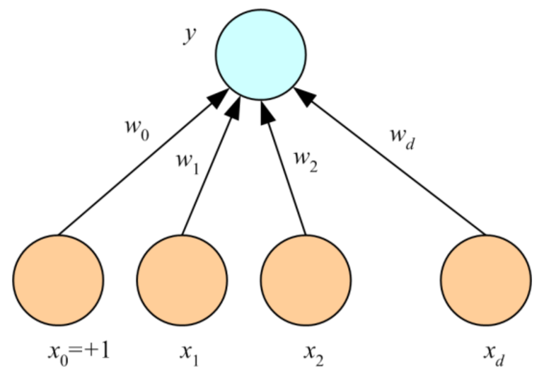
Neurons

- Biological neural networks
  - Approximately 10^{11} interconnected *neurons*, each connected to 10–100,000 others through *synapses*
  - More than 10^{14} synaptic connections
  - Highly complex and massively parallel



Perceptrons

- Perceptron [McCulloch and Pitts 1943, Rosenblatt 1957]
  - Input x_j
  - Output y
  - Connection *weight* (or synaptic weight) w_j

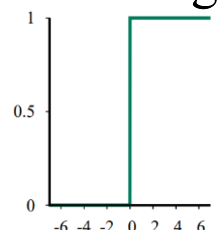


- Linear regression
  - $y = \sum_j w_j x_j + w_0 = \boldsymbol{w}^\top \boldsymbol{x}$

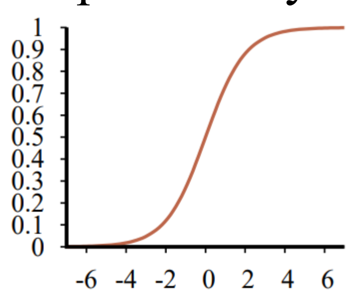
$; \boldsymbol{x} \equiv [1, x_1, \cdots, x_d]$

w_0: bias

- Binary classification
  - Linear discriminant function using a *hard threshold* function
  - $y = \begin{cases} 1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} > 0 \\ 0 & \text{otherwise} \end{cases}$



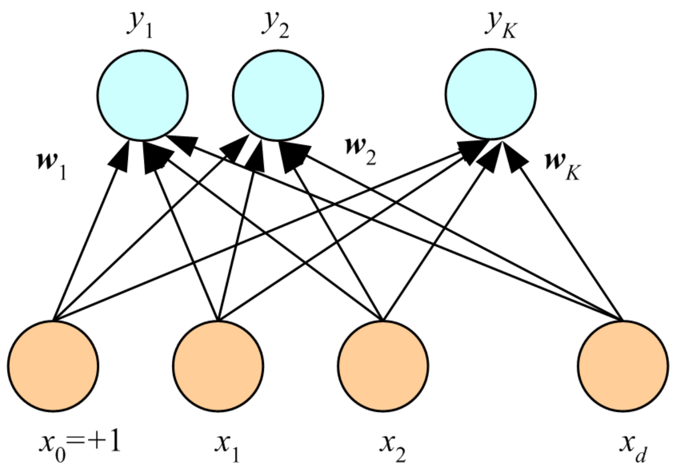
- *Sigmoid* function for posterior probability
  - $y = \frac{1}{1+\exp(-\boldsymbol{w}^\top \boldsymbol{x})}$



$; \exp(-\boldsymbol{w}^\top \boldsymbol{x}) \equiv e^{-\boldsymbol{w}^\top \boldsymbol{x}}$

Multiple Perceptrons

- Multiple perceptrons
  - Input x_j
  - Output y_i
  - Weight w_{ij} (between input x_j and output y_i)



$; \boldsymbol{x} \equiv [1, x_1, \cdots, x_d]$

- Multivariate linear regression
  - $y_i = \sum_j w_{ij} x_j + w_{i0} = \boldsymbol{w}_i^\top \boldsymbol{x}$
    $\boldsymbol{y} = \mathbf{W}\boldsymbol{x}$

- Multiclass classification
  - $\boldsymbol{y} = \mathbf{W}\boldsymbol{x}$
    $\boldsymbol{x} \in c_k$ if $k = \arg\max_i y_i$

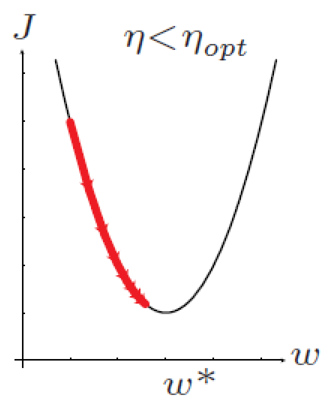
- *Softmax* function for posterior probability
  - $s_i = \sum_j w_{ij} x_j + w_{i0} = \boldsymbol{w}_i^\top \boldsymbol{x}$
    $y_i = \dfrac{\exp(s_i)}{\sum_i \exp(s_i)}$

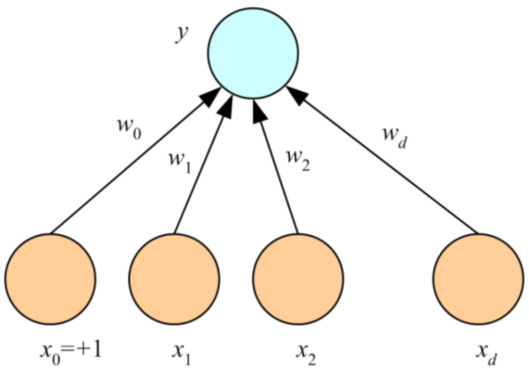
Contents

Learning for a Linear Output Node

- *Stochastic gradient descent* (SGD) for a linear output node
  - $E(\boldsymbol{w}|\boldsymbol{x}, r) \equiv \frac{1}{2}(r-y)^2$; *mean squared error* (MSE)
  - $\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w_j}$; $y = \boldsymbol{w}^\mathsf{T}\boldsymbol{x} = \sum_j w_j x_j + w_0$

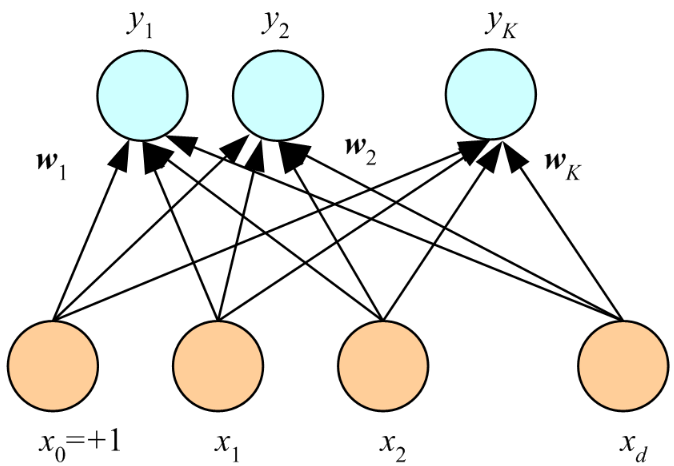
    $= \underbrace{-(r-y)}_{\frac{\partial E}{\partial y}} \underbrace{x_j}_{\frac{\partial y}{\partial w_j}}$
  - $w_j \leftarrow w_j + \eta(r-y)x_j$; η: *learning rate* or *step size*



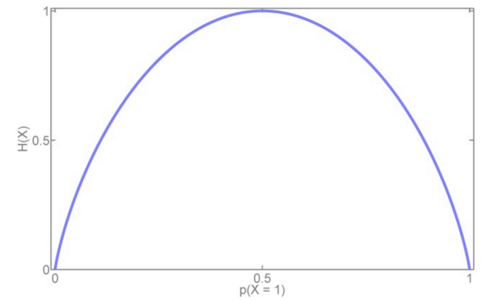


Learning for Multiple Linear Output Nodes

- Stochastic gradient descent for multiple linear output nodes
  - $E(\mathbf{W}|\boldsymbol{x},\boldsymbol{r}) \equiv \sum_k \underbrace{\frac{1}{2}(r_k - y_k)^2}_{E_k}$; $y_i = \boldsymbol{w}_i^\top \boldsymbol{x} = \sum_j w_{ij}x_j + w_{i0}$
  - $\frac{\partial E}{\partial w_{ij}} = \sum_k \frac{\partial E_k}{\partial y_i}\frac{\partial y_i}{\partial w_{ij}}$
    $= \underbrace{-(r_i - y_i)}_{\frac{\partial E_i}{\partial y_i}} \underbrace{x_j}_{\frac{\partial y_i}{\partial w_{ij}}}$
  - $w_{ij} \leftarrow w_{ij} + \eta(r_i - y_i)x_j$



Entropy



- **Entropy**
  - $\mathbb{H}(p) \equiv -\sum_x p(x)\log_2 p(x)$; $\mathbb{H}(x)$
  - e.g., $-[\theta\log_2\theta + (1-\theta)\log_2(1-\theta)]$

- **Cross entropy**
  - $\mathbb{H}(p,q) \equiv -\sum_x p(x)\log_2 q(x)$; minimum when $p = q$ ($\because$ Jensen's inequality)

- **Kullback-Leibler (KL) divergence** (also called *relative entropy*)
  - $$\mathbb{KL}(p||q) \equiv \mathbb{H}\left(p(x), \frac{q(x)}{p(x)}\right) = -\sum_x p(x)\log_2\frac{q(x)}{p(x)}$$
    $$= -\sum_x p(x)\log_2 q(x) + \sum_x p(x)\log_2 p(x) = \mathbb{H}(p,q) - \mathbb{H}(p)$$

- **Conditional entropy**
  - $$\mathbb{H}(x|y) \equiv \mathbb{H}\big(p(x,y), p(x|y)\big) = -\sum_{x,y}\overbrace{p(x,y)}^{p(y)p(x|y)}\log_2 p(x|y)$$
    $$= -\sum_y\sum_x p(y)p(x|y)\log_2 p(x|y) = -\sum_y p(y)\,\mathbb{H}\big(p(x|y)\big)$$

- **Mutual information** (MI, also called *information gain*)
  - $\mathbb{I}(x,y) \equiv \mathbb{KL}\big(p(x,y)||p(x)p(y)\big) = \mathbb{H}(x) - \mathbb{H}(x|y) = \mathbb{H}(y) - \mathbb{H}(y|x)$

Learning for a Sigmoid Output Node

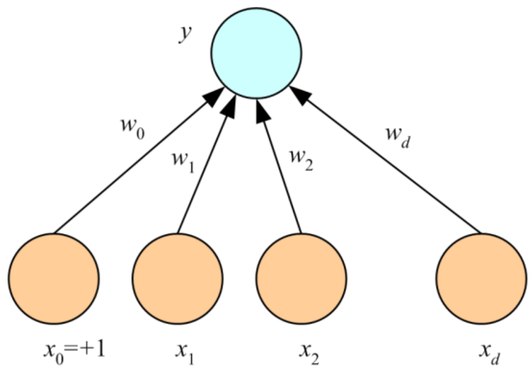
- Stochastic gradient descent for a sigmoid output node
  - $E(\boldsymbol{w}|\boldsymbol{x}, r) \equiv -r \log y - (1-r)\log(1-y)$; *cross entropy*
  - $\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial s}\frac{\partial s}{\partial w_i}$; $y = \frac{1}{1+\exp(-s)}$, $s = \boldsymbol{w}^\top \boldsymbol{x}$

$$= \underbrace{\left(-\frac{r}{y} + \frac{1-r}{1-y}\right)}_{\frac{\partial E}{\partial y}} \underbrace{y(1-y)}_{\frac{\partial y}{\partial s}} \underbrace{x_i}_{\frac{\partial s}{\partial w_i}}$$

$$= -(r-y)x_i$$

$$\begin{aligned} y' &= -\frac{1}{(1+\exp(-s))^2}\exp(-s)\,(-1) \\ &= \frac{1}{1+\exp(-s)}\frac{\exp(-s)}{1+\exp(-s)} \\ &= \frac{1}{1+\exp(-s)}\left(\frac{1+\exp(-s)}{1+\exp(-s)} - \frac{1}{1+\exp(-s)}\right) \\ &= y(1-y) \end{aligned}$$

  - $w_i \leftarrow w_i + \eta(r-y)x_i$



Learning for Softmax Output Nodes

- Stochastic gradient descent for a softmax output node
  - $E(\mathbf{W}|\boldsymbol{x},\boldsymbol{r}) \equiv -\sum_k \underbrace{r_k \log y_k}_{E_k}$; $y_k = \frac{\exp(s_k)}{\sum_{\acute{k}} \exp(s_{\acute{k}})}$, $s_i = \boldsymbol{w}_i^{\top} \boldsymbol{x}$
  - $\frac{\partial E}{\partial w_{ij}} = -\sum_k \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial s_i} \frac{\partial s_i}{\partial w_{ij}}$

$$= -\left(\sum_{k\neq i} \underbrace{\frac{r_k}{y_k}}_{\frac{\partial E_k}{\partial y_k}} \underbrace{\left(-\frac{\exp(s_k)\exp(s_i)}{\left(\sum_{\acute{k}}\exp(s_{\acute{k}})\right)^2}\right)}_{\frac{\partial y_k}{\partial s_i}} + \underbrace{\frac{r_i}{y_i}}_{\frac{\partial E_i}{\partial y_i}} \underbrace{\left(\frac{\exp(s_i)}{\sum_{\acute{i}}\exp(s_{\acute{i}})} - \frac{\exp(s_i)\exp(s_i)}{\left(\sum_{\acute{i}}\exp(s_{\acute{i}})\right)^2}\right)}_{\frac{\partial y_i}{\partial s_i}}\right) \underbrace{x_j}_{\frac{\partial s_i}{\partial w_{ij}}}$$

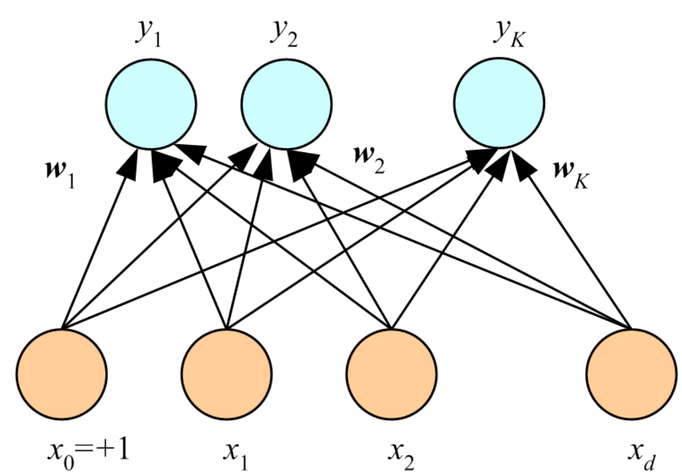
$$= -\left(\sum_{k\neq i} \frac{r_k}{y_k}(-y_k y_i) + \frac{r_i}{y_i}\left(y_i - y_i^2\right)\right) x_j$$

$$= -\left(\sum_{k\neq i} r_k(-y_i) + r_i(1 - y_i)\right)x_j$$

$$= -\left(\sum_k r_k(-y_i) + r_i\right)x_j$$

$$= -(r_i - y_i)x_j$$

  - $w_{ij} \leftarrow w_{ij} + \eta(r_i - y_i)x_j$

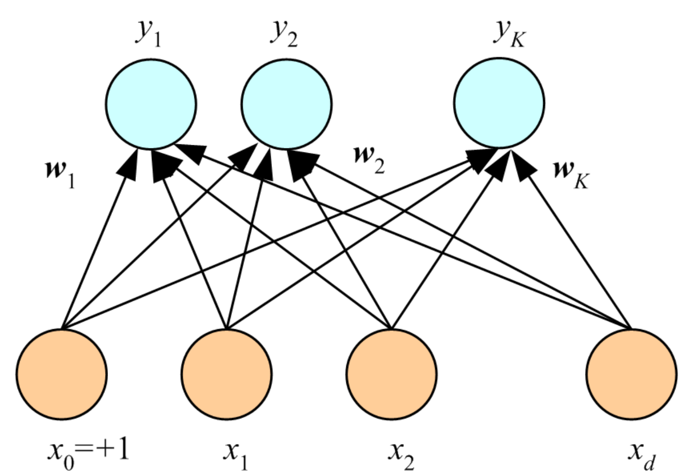


Stochastic Gradient Descent for Perceptrons

- Input: $\mathcal{D} = \{\boldsymbol{x}^t, \boldsymbol{r}^t\}_{t=1}^{N}$

Initialize **W** with small random numbers.
repeat ; epoch
    for each $(\boldsymbol{x}^t, \boldsymbol{r}^t) \in \mathcal{D}$ in random order
        for each output node index i
            $s_i \leftarrow \boldsymbol{w}_i^\top \boldsymbol{x}^t$
        end
        for each output node index i
            $y_i \leftarrow \exp(s_i)/\sum_i \exp(s_i)$; regression: $y_i \leftarrow s_i$
        end
        for each output node index i sigmoid: $y \leftarrow 1/(1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x}))$
            for each input node index j Hard threshold: $y \leftarrow \begin{cases} 1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} > 0 \\ 0 & \text{otherwise} \end{cases}$
                $w_{ij} \leftarrow w_{ij} + \eta(r_i^t - y_i)x_j^t$
            end
        end
    end
until convergence

- Output: **W**



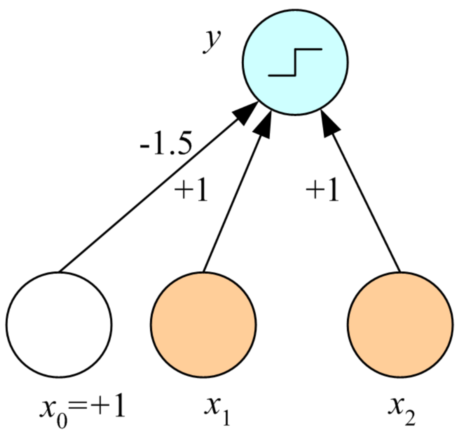
Contents

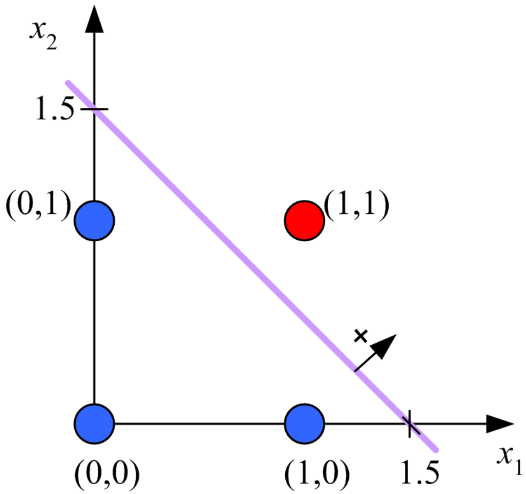
Boolean AND

- Boolean AND operation
  - $y = x_1 \text{ AND } x_2$
  - $y = \begin{cases} 1 & \text{if } \boldsymbol{w}^\mathsf{T}\boldsymbol{x} > 0 \\ 0 & \text{otherwise} \end{cases}$
  - $\boldsymbol{w}^\mathsf{T}\boldsymbol{x} = w_0 + w_1 x_1 + w_2 x_2 > 0$

    $\Rightarrow x_2 > -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$

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

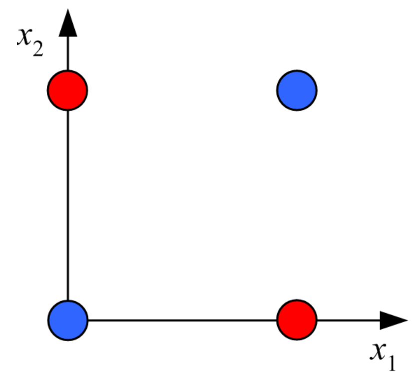




Boolean XOR

- Boolean XOR operation
  - $y = x_1 \text{ XOR } x_2$
  - $y = \begin{cases} 1 & \text{if } \boldsymbol{w}^\mathsf{T}\boldsymbol{x} > 0 \\ 0 & \text{otherwise} \end{cases}$
  - $\boldsymbol{w}^\mathsf{T}\boldsymbol{x} = w_0 + w_1 x_1 + w_2 x_2 > 0$
    $\Rightarrow x_2 > -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$

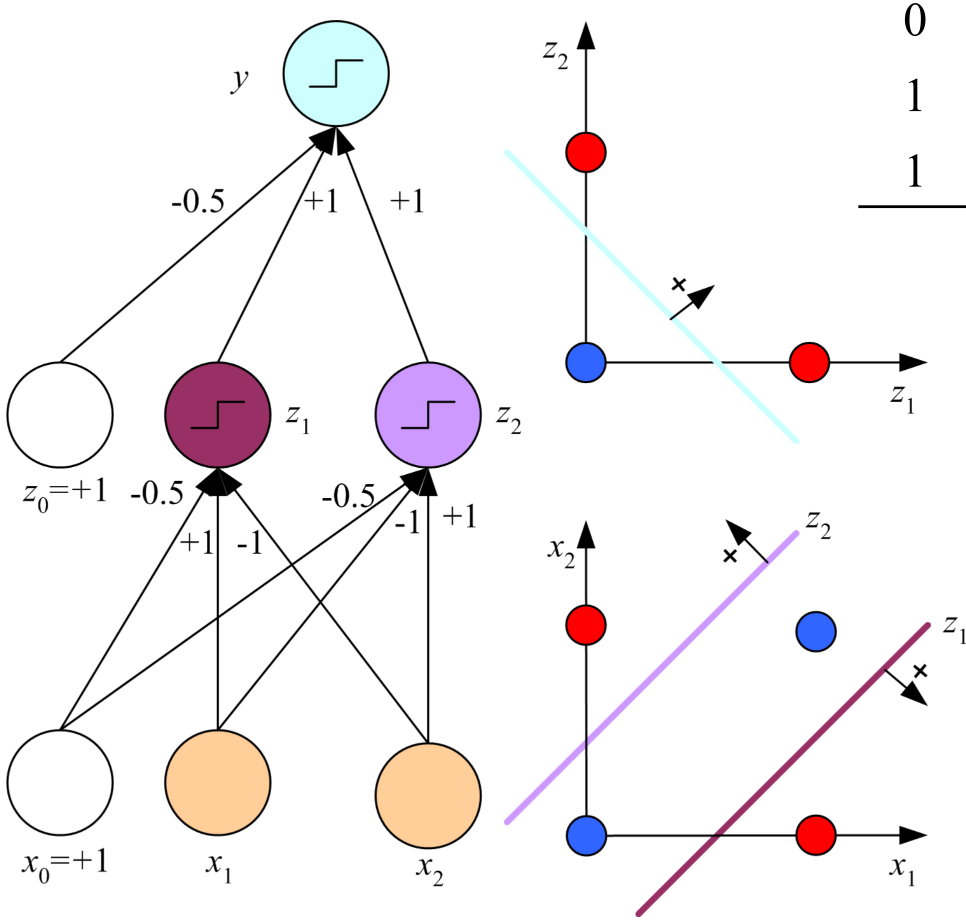
x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0



Boolean XOR

- Boolean XOR function as a disjunction of conjunctions
  - x_1 XOR $x_2 = (x_1$ AND $\neg x_2)$ OR $(\neg x_1$ AND $x_2)$

x_1	x_2	z_1	z_2	r
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0



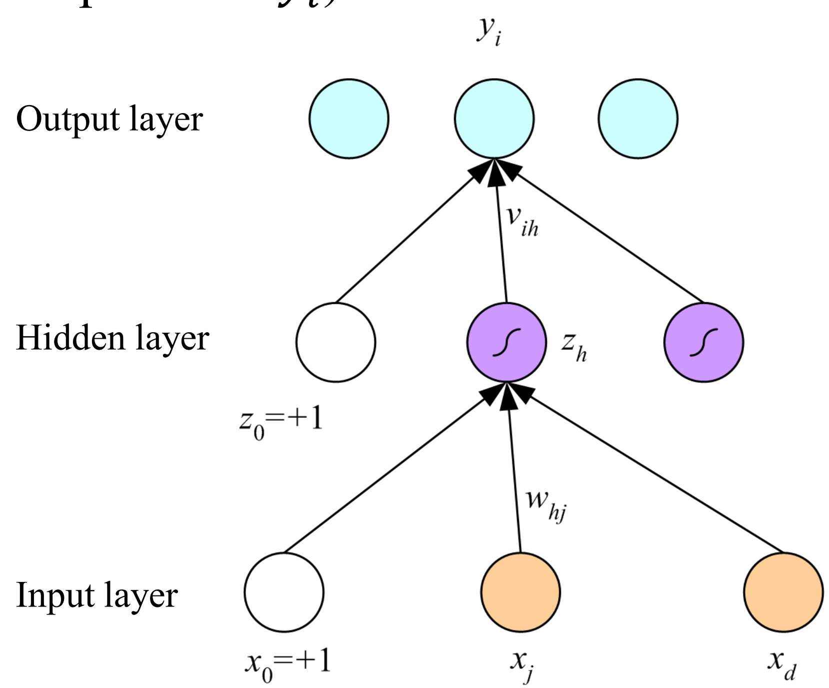
Two-layer perceptrons

Contents

Multilayer Perceptrons

- Multilayer perceptrons (MLP)
  - Input x_j
  - Hidden node $z_i = \frac{1}{1+\exp\left(-\boldsymbol{w}_{z,i}^{\top}\boldsymbol{x}\right)}$; nonlinear activation function
  - Weight $w_{z,ij}$ (between input x_j and hidden node z_i)
  - Output $y_i = \begin{cases} \boldsymbol{w}_{o,i}^{\top}\boldsymbol{z} & ; z_0 \equiv 1 \\ \frac{\exp(s_i)}{\sum_i \exp(s_i)} & ; s_i = \boldsymbol{w}_{o,i}^{\top}\boldsymbol{z} \end{cases}$
  - Weight $w_{o,ij}$ (between hidden node z_j and output node y_i)

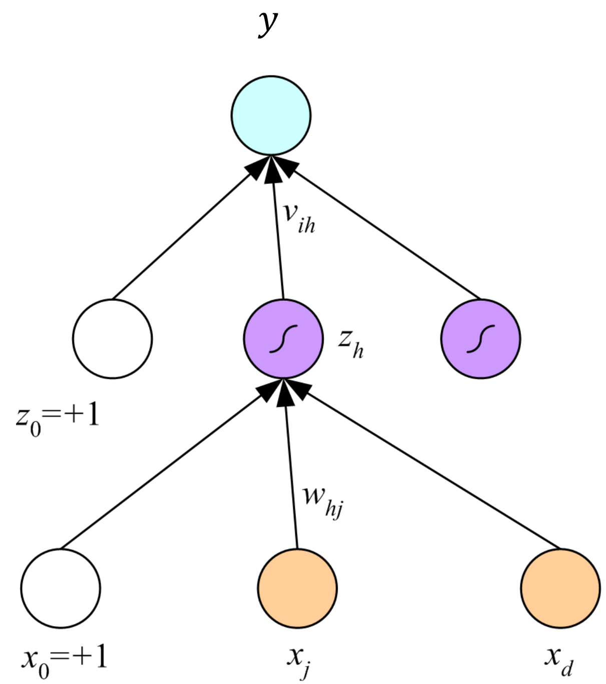
- Importance of non-linearity in hidden layers
  - $\boldsymbol{z} = \mathbf{W}\boldsymbol{x}$
    $\boldsymbol{y} = \mathbf{V}\boldsymbol{z}$
    $\boldsymbol{y} = \mathbf{V}\mathbf{W}\boldsymbol{x} = \mathbf{U}\boldsymbol{x}$



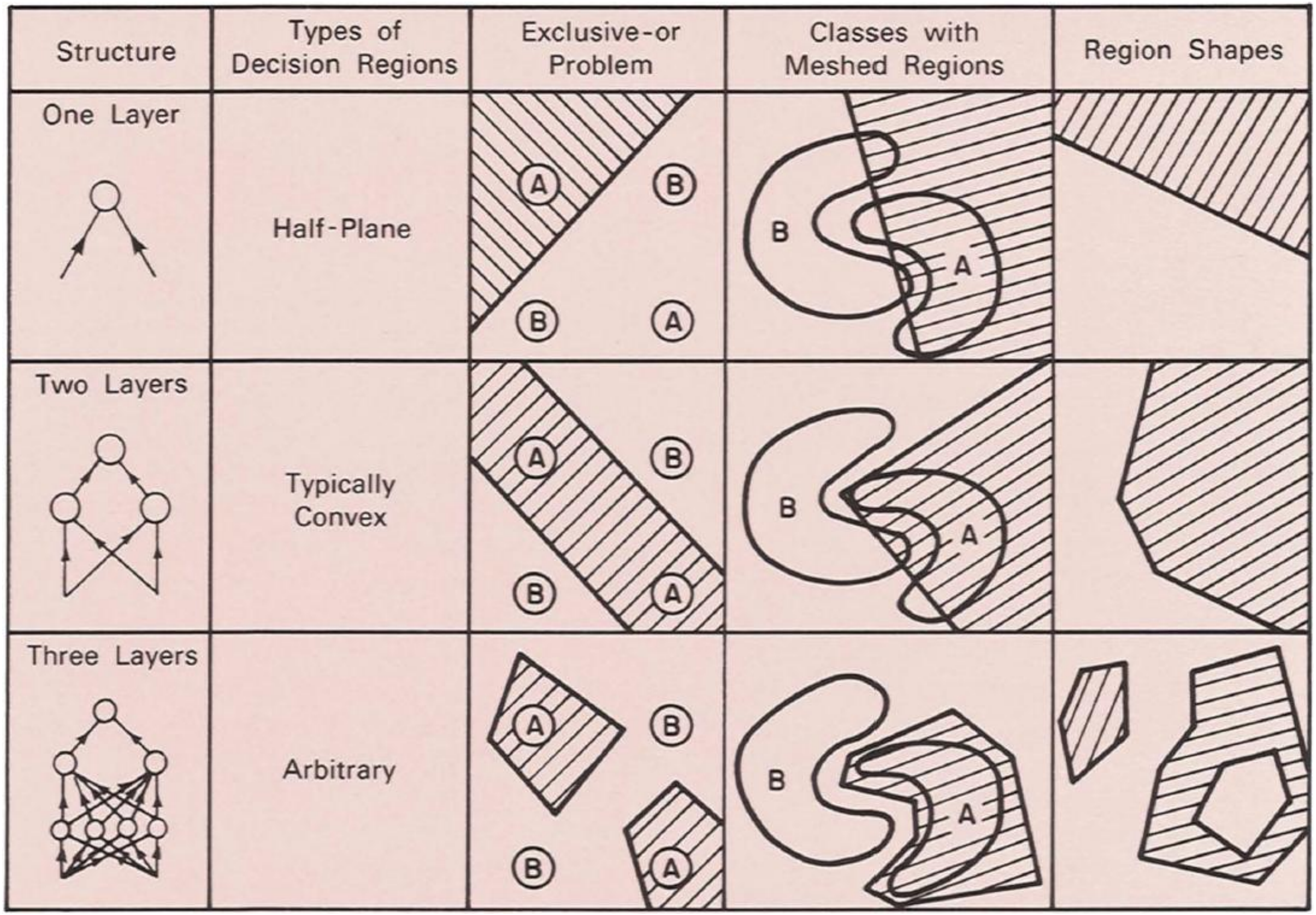
Universal Approximation Theorem

- Any arbitrary function with continuous input and output can be approximated arbitrarily closely by a two-layer network given sufficient number of hidden units, proper nonlinearities, and weights.

- Proof
  - Fourier theorem



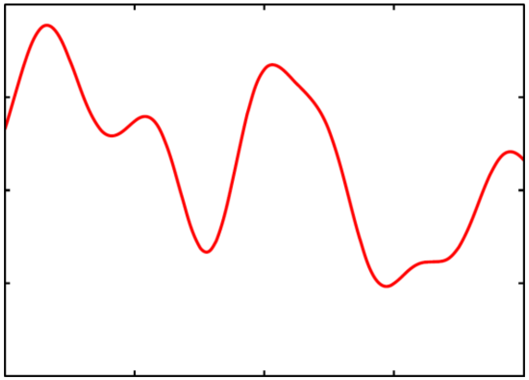
Decision Regions of MLPs

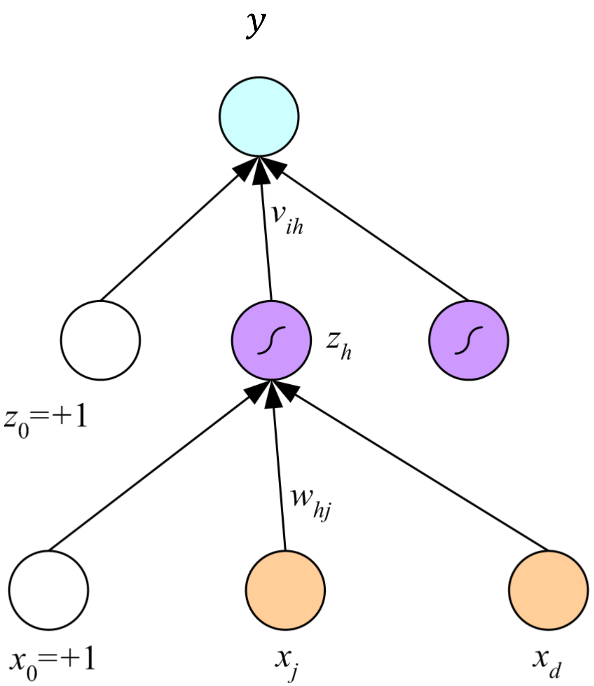


Contents

Error Backpropagation

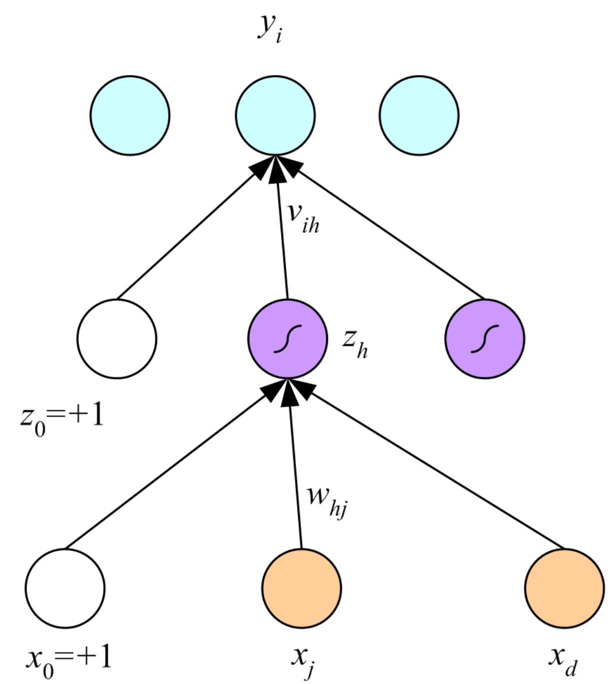
- Gradient descent
  - $E(\theta|\boldsymbol{x}, \boldsymbol{r}) \equiv \frac{1}{2}\sum_i (r_i - y_i)^2$
  - $E(\theta|\boldsymbol{x}, \boldsymbol{r}) \equiv -\sum_i r_i \log y_i$
  - $\theta \leftarrow \theta - \eta \nabla E$





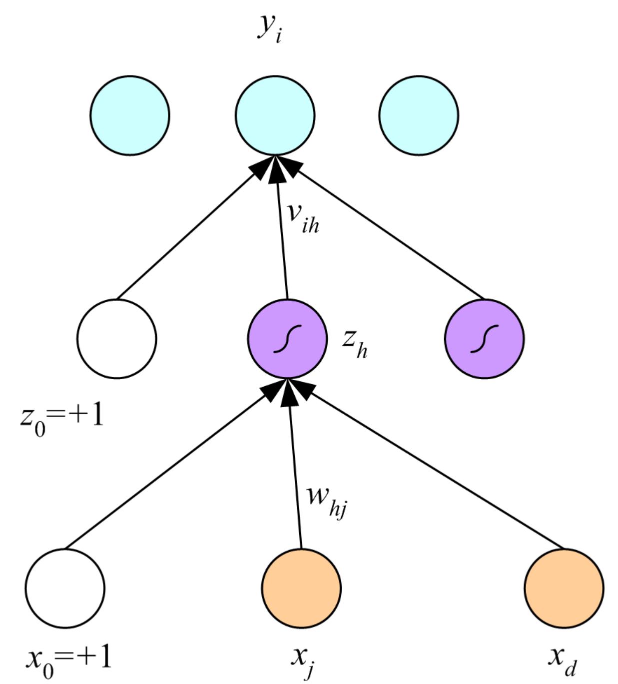
Learning for Multiple Linear Output Nodes

- Weight update for multiple linear output nodes
  - $E(\mathbf{W}_z, \mathbf{W}_o | \boldsymbol{x}, \boldsymbol{r}) \equiv \sum_k \underbrace{\frac{1}{2}(r_k - y_k)^2}_{E_k}$; $y_k = \boldsymbol{w}_{o,k}^{\top}\boldsymbol{z}$, $z_i = \frac{1}{1+\exp\left(-\boldsymbol{w}_{z,i}^{\top}\boldsymbol{x}\right)}$
  - $$\frac{\partial E}{\partial w_{o,ij}} = \sum_k \frac{\partial E_k}{\partial y_i} \frac{\partial y_i}{\partial w_{o,ij}} = \underbrace{-(r_i - y_i)}_{\frac{\partial E_i}{\partial y_i}} \underbrace{z_j}_{\frac{\partial y_i}{\partial w_{o,ij}}}$$
  - $w_{o,ij} \leftarrow w_{o,ij} + \eta(r_i - y_i)z_j$



Learning for Hidden Nodes

- Weight update for a hidden node connected to multiple linear output nodes
  - $E(\mathbf{W}_z, \mathbf{W}_o | \boldsymbol{x}, \boldsymbol{r}) \equiv \sum_k \underbrace{\frac{1}{2}(r_k - y_k)^2}_{E_k}$; $y_k = \boldsymbol{w}_{o,k}^\top \boldsymbol{z}$, $z_i = \frac{1}{1+\exp\left(-\boldsymbol{w}_{z,i}^\top \boldsymbol{x}\right)}$
  - $$\frac{\partial E}{\partial w_{z,ij}} = \sum_k \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial z_i} \frac{\partial z_i}{\partial w_{z,ij}}$$
    $$= \sum_k \underbrace{-(r_k - y_k)}_{\frac{\partial E_k}{\partial y_k}} \underbrace{w_{o,ki}}_{\frac{\partial y_k}{\partial z_i}} \underbrace{z_i(1 - z_i)x_j}_{\frac{\partial z_i}{\partial w_{z,ij}}}$$
  - $w_{z,ij} \leftarrow w_{z,ij} + \eta \sum_k (r_k - y_k) w_{o,ki}\, z_i(1 - z_i) x_j$



Contents

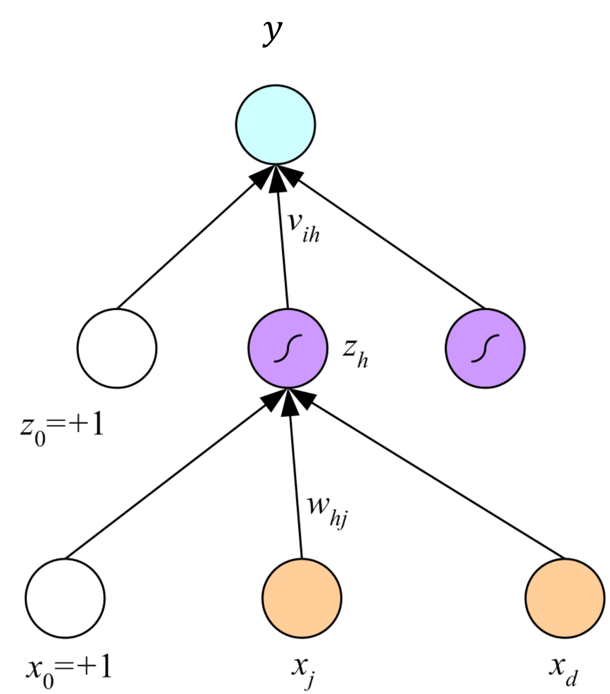
Learning for a Sigmoid Output Node

- Weight update for a single sigmoid output node
  - $E(\mathbf{W}_z, \boldsymbol{w}_o | \boldsymbol{x}, r) \equiv -r \log y - (1-r)\log(1-y) \quad ; y = \frac{1}{1+\exp(-s)}, \; s = \boldsymbol{w}_o^\top \boldsymbol{z}$
  - $\frac{\partial E}{\partial w_{o,i}} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial s}\frac{\partial s}{\partial w_{o,i}} \qquad z_i = \frac{1}{1+\exp\left(-\boldsymbol{w}_{z,i}^\top \boldsymbol{x}\right)}$

$$= \underbrace{\left(-\frac{r}{y} + \frac{1-r}{1-y}\right)}_{\frac{\partial E}{\partial y}} \underbrace{y(1-y)}_{\frac{\partial y}{\partial s}} \underbrace{z_i}_{\frac{\partial s}{\partial w_{o,i}}}$$

$$= -(r-y)z_i$$

  - $w_{o,i} \leftarrow w_{o,i} + \eta(r-y)z_i$



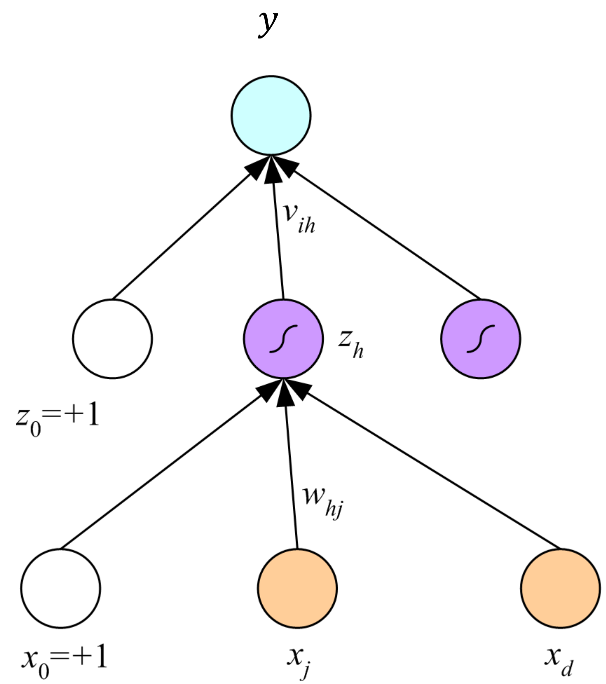
Learning for Hidden Nodes

- Weight update for a hidden node connected to a single sigmoid output node
  - $E(\mathbf{W}_z, \boldsymbol{w}_o | \boldsymbol{x}, r) \equiv -r \log y - (1-r)\log(1-y)$; $y = \dfrac{1}{1+\exp(-\boldsymbol{w}_o^\top \boldsymbol{z})}$
  - $\dfrac{\partial E}{\partial w_{z,ij}} = \dfrac{\partial E}{\partial y}\dfrac{\partial y}{\partial z_i}\dfrac{\partial z_i}{\partial w_{z,ij}}$; $z_i = \dfrac{1}{1+\exp\left(-\boldsymbol{w}_{z,i}^\top \boldsymbol{x}\right)}$

$$= \underbrace{\left(-\frac{r}{y} + \frac{1-r}{1-y}\right)}_{\frac{\partial E}{\partial y}} \underbrace{y(1-y)w_{o,i}}_{\frac{\partial y}{\partial z_i}} \underbrace{z_i(1-z_i)x_j}_{\frac{\partial z_i}{\partial w_{z,ij}}}$$

$$= -(r-y)w_{o,i}z_i(1-z_i)x_j$$

  - $w_{z,ij} \leftarrow w_{z,ij} + \eta(r-y)w_{o,i}z_i(1-z_i)x_j$



Learning for Softmax Output Nodes

- Weight update for a softmax output node
  - $E(\mathbf{W}_z, \mathbf{W}_o | \boldsymbol{x}, \boldsymbol{r}) \equiv -\sum_k \underbrace{r_k \log y_k}_{E_k}$; $y_k = \frac{\exp(s_k)}{\sum_{\acute{k}} \exp(s_{\acute{k}})}$, $s_i = \boldsymbol{w}_{o,i}^\top \boldsymbol{z}$
  - $\frac{\partial E}{\partial w_{o,ij}} = -\sum_k \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial s_i} \frac{\partial s_i}{\partial w_{o,ij}}$; $z_i = \frac{1}{1+\exp\left(-\boldsymbol{w}_{z,i}^\top \boldsymbol{x}\right)}$

$$= -\left(\sum_{k\neq i} \underbrace{\frac{r_k}{y_k}}_{\frac{\partial E_k}{\partial y_k}} \underbrace{\left(-\frac{\exp(s_k)\exp(s_i)}{\left(\sum_{\acute{k}} \exp(s_{\acute{k}})\right)^2}\right)}_{\frac{\partial y_k}{\partial s_i}} + \underbrace{\frac{r_i}{y_i}}_{\frac{\partial E_i}{\partial y_i}} \underbrace{\left(\frac{\exp(s_i)}{\sum_{\acute{k}} \exp(s_{\acute{k}})} - \frac{\exp(s_i)\exp(s_i)}{\left(\sum_{\acute{k}} \exp(s_{\acute{k}})\right)^2}\right)}_{\frac{\partial y_i}{\partial s_i}}\right) \underbrace{z_j}_{\frac{\partial s_i}{\partial w_{o,ij}}}$$

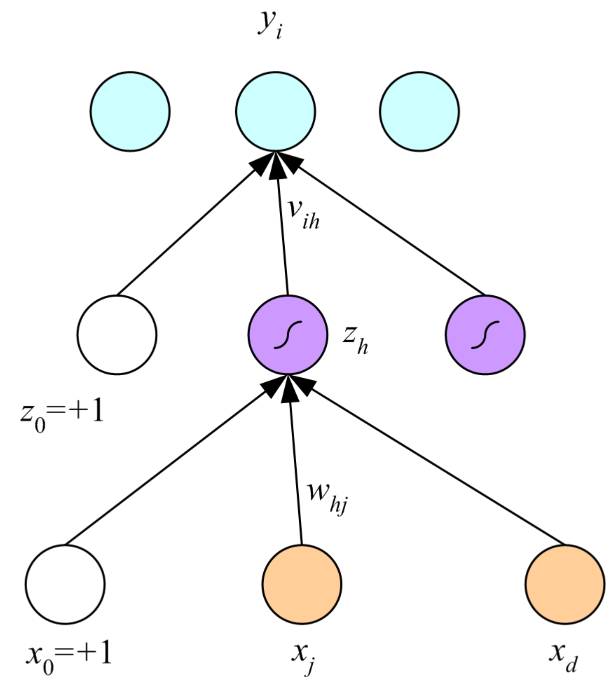
$$= -\left(\sum_{k\neq i} \frac{r_k}{y_k}(-y_k y_i) + \frac{r_i}{y_i}\left(y_i - y_i^2\right)\right) z_j$$

$$= -\left(\sum_{k\neq i} r_k(-y_i) + r_i(1-y_i)\right) z_j$$

$$= -\left(\sum_k r_k(-y_i) + r_i\right) z_j$$

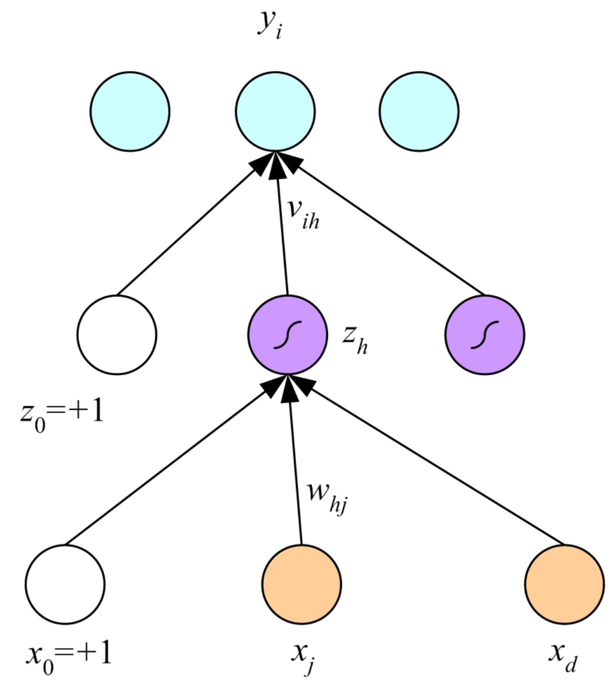
$$= -(r_i - y_i) z_j$$

  - $w_{o,ij} \leftarrow w_{o,ij} + \eta(r_i - y_i) z_j$



Learning for Hidden Nodes

- Weight update for a hidden node connected to multiple softmax output nodes
  - $E(\mathbf{W}_z, \mathbf{W}_o | \boldsymbol{x}, \boldsymbol{r}) \equiv -\sum_k \underbrace{r_k \log y_k}_{E_k}$; $y_k = \frac{\exp(s_k)}{\sum_{\acute{k}} \exp(s_{\acute{k}})}$, $s_i = \boldsymbol{w}_{o,i}^{\top} \boldsymbol{z}$
  - $\frac{\partial E}{\partial w_{z,ij}} = -\sum_k \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial z_i} \frac{\partial z_i}{\partial w_{z,ij}}$; $z_i = \frac{1}{1+\exp\left(-\boldsymbol{w}_{z,i}^{\top} \boldsymbol{x}\right)}$
  - $w_{z,ij} \leftarrow w_{z,ij} + \eta \sum_k (r_k - y_k) w_{o,ki}\, z_i (1 - z_i) x_j$

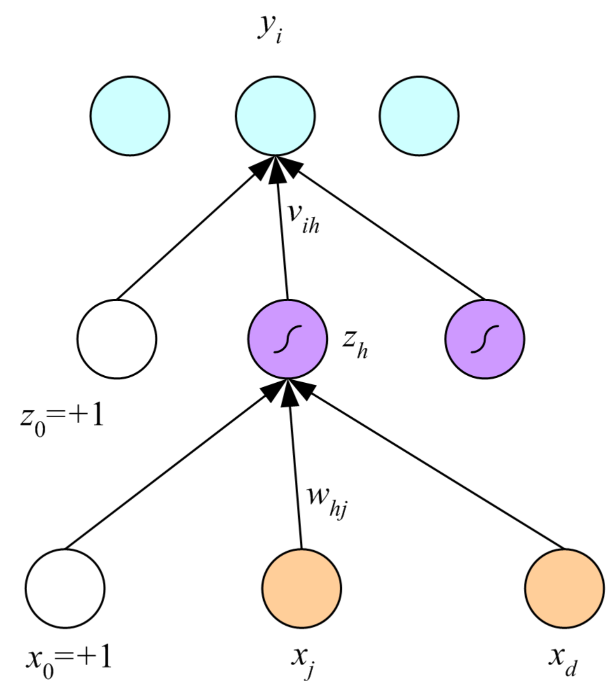


Learning for Output Nodes

- Weight update for a single sigmoid output node
  - $w_{o,i} \leftarrow w_{o,i} + \eta(r - y)z_i$; $y = \frac{1}{1+\exp(-\boldsymbol{w}_o^\top \boldsymbol{z})}$, $z_i = \frac{1}{1+\exp\left(-\boldsymbol{w}_{z,i}^\top \boldsymbol{x}\right)}$

- Weight update for multiple softmax output nodes
  - $w_{o,ij} \leftarrow w_{o,ij} + \eta(r_i - y_i)z_j$; $y_i = \frac{\exp(s_i)}{\sum_i \exp(s_i)}$, $s_i = \boldsymbol{w}_{o,i}^\top \boldsymbol{z}$

- Weight update for multiple linear output nodes
  - $w_{o,ij} \leftarrow w_{o,ij} + \eta(r_i - y_i)z_j$; $y_i = \boldsymbol{w}_{o,i}^\top \boldsymbol{z}$

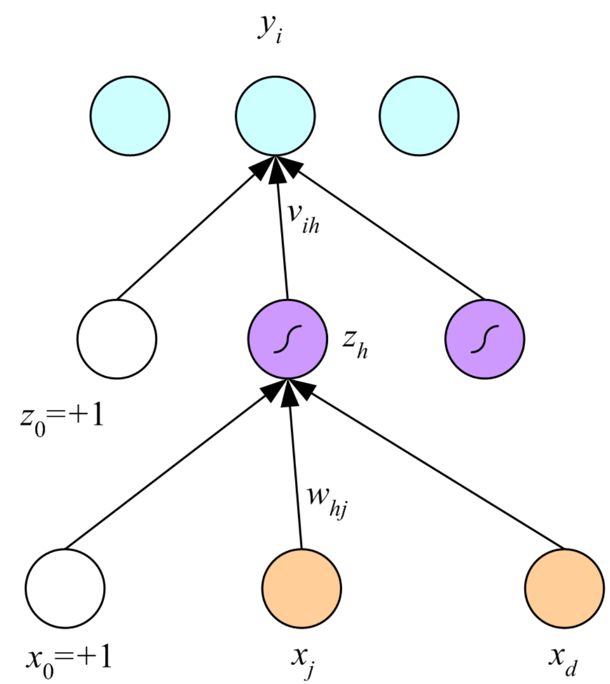


Learning for Hidden Nodes

- Weight update for a hidden node connected to a single sigmoid output node
  - $w_{z,ij} \leftarrow w_{z,ij} + \eta(r - y)w_{o,i}z_i(1 - z_i)x_j$

- Weight update for a hidden node connected to multiple softmax output nodes
  - $w_{z,ij} \leftarrow w_{z,ij} + \eta \sum_k (r_k - y_k)w_{o,ki}\, z_i(1 - z_i)x_j$

- Weight update for a hidden node connected to multiple linear output nodes
  - $w_{z,ij} \leftarrow w_{z,ij} + \eta \sum_k (r_k - y_k)w_{o,ki}\, z_i(1 - z_i)x_j$



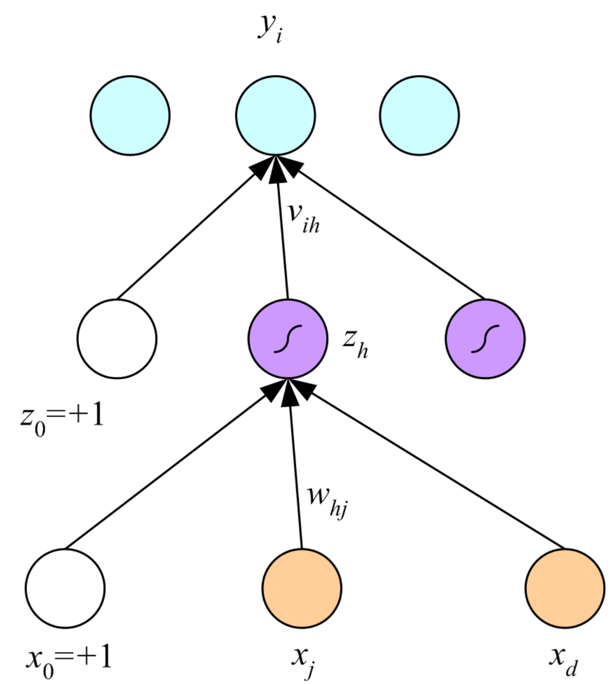
Stochastic Gradient Descent for MLPs

- Input: $\mathcal{D} = \{\boldsymbol{x}^t, \boldsymbol{r}^t\}_{t=1}^{N}$

Initialize $\mathbf{W}_z$ and $\mathbf{W}_o$ with small random numbers.
repeat
    for each $(\boldsymbol{x}^t, \boldsymbol{r}^t) \in \mathcal{D}$ in random order
        for each hidden node index i
            $z_i \leftarrow 1/\left(1 + \exp\left(-\boldsymbol{w}_{z,i}^{\top} \boldsymbol{x}^t\right)\right)$
        end
        for each output node index i
            $s_i \leftarrow \boldsymbol{w}_{o,i}^{\top} \boldsymbol{z}$
        end
        for each output node index i
            $y_i \leftarrow \exp(s_i) / \sum_i \exp(s_i)$
        end
        for each hidden node index i
            for each input node index j
                $w_{z,ij} \leftarrow w_{z,ij} + \eta \sum_k \left(r_k^t - y_k\right) w_{o,ki}\, z_i (1 - z_i) x_j^t$
            end
        end
        for each output node index i
            for each hidden node index j
                $w_{o,ij} \leftarrow w_{o,ij} + \eta \left(r_i^t - y_i\right) z_j$
            end
        end
    end
until convergence

; sigmoid: $y \leftarrow 1/(1 + \exp(-\boldsymbol{w}^{\top} \boldsymbol{x}))$

- Output: $\mathbf{W}_z$ and $\mathbf{W}_o$

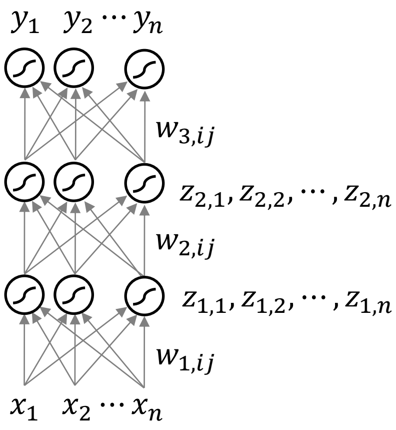


Contents

Deep Neural Networks

- Deep neural network (DNN)
  - Multiple hidden layers
    - "deep and narrow" vs. "shallow and wide"
  - Representation learning ; more abstract concept
    - e.g., Image recognition: pixels → edges → corners → digits

- Deep learning

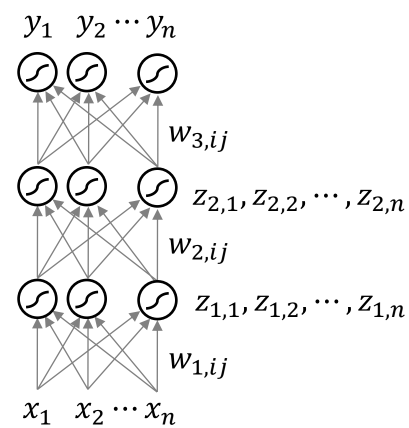


Learning for Multiple Hidden Layers

- Weight update for a hidden node of the first hidden layer in a two hidden layer neural network with multiple linear output nodes
  - $E(\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 | \boldsymbol{x}, \boldsymbol{r}) \equiv \sum_k \underbrace{\frac{1}{2}(r_k - y_k)^2}_{E_k}$; $y_k = \sum_m w_{3,km} z_{2,m}$
  - $\frac{\partial E}{\partial w_{1,ij}} = \sum_k \frac{\partial E_k}{\partial y_k} \underbrace{\frac{\partial y_k}{\partial z_{1,i}}}_{\frac{\partial \sum_m w_{3,km} z_{2,m}}{\partial z_{1,i}} = \sum_m \frac{\partial w_{3,km} z_{2,m}}{\partial z_{2,m}} \frac{\partial z_{2,m}}{\partial z_{1,i}}} \frac{\partial z_{1,i}}{\partial w_{1,ij}}$; $z_{2,m} = \frac{1}{1+\exp\left(-\sum_i w_{2,mi} z_{1,i}\right)}$; $z_{1,i} = \frac{1}{1+\exp\left(-\sum_j w_{1,ij} x_j\right)}$

$$= \sum_k \underbrace{-(r_k - y_k)}_{\frac{\partial E_k}{\partial y_k}} \sum_m \underbrace{w_{3,km}}_{\frac{\partial w_{3,km} z_{2,m}}{\partial z_{2,m}}} \underbrace{z_{2,m}\left(1 - z_{2,m}\right) w_{2,mi}}_{\frac{\partial z_{2,m}}{\partial z_{1,i}}} \underbrace{z_{1,i}\left(1 - z_{1,i}\right) x_j}_{\frac{\partial z_{1,i}}{\partial w_{1,ij}}}$$

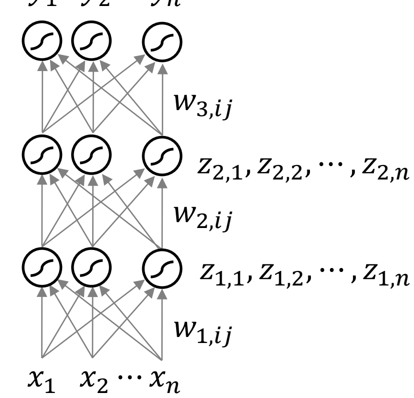
  - $w_{1,ij} \leftarrow w_{1,ij} + \eta \sum_k (r_k - y_k) \sum_m w_{3,km} z_{2,m}\left(1 - z_{2,m}\right) w_{2,mi}\, z_{1,i}\left(1 - z_{1,i}\right) x_j$



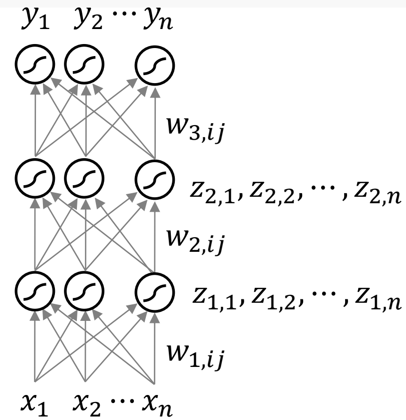
Learning for Multiple Hidden Layers

- Weight update for a hidden node of the first hidden layer in a two hidden layer neural network with multiple softmax output nodes
  - $E(\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3|\boldsymbol{x}, \boldsymbol{r}) \equiv -\sum_k \underbrace{r_k \log y_k}_{E_k}$; $y_k = \frac{\exp(s_k)}{\sum_{\acute{k}} \exp(s_{\acute{k}})}$, $s_k = \sum_m w_{3,km} z_{2,m}$
  ; $z_{2,m} = \frac{1}{1+\exp(-\sum_i w_{2,mi} z_{1,i})}$
  - $\frac{\partial E}{\partial w_{1,ij}} = -\sum_k \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial z_{1,i}} \frac{\partial z_{1,i}}{\partial w_{1,ij}}$; $z_{1,i} = \frac{1}{1+\exp\left(-\sum_j w_{1,ij} x_j\right)}$
  - $w_{1,ij} \leftarrow w_{1,ij} + \eta \sum_k (r_k - y_k) \sum_m w_{3,km} z_{2,m} \left(1 - z_{2,m}\right) w_{2,mi}\, z_{1,i} \left(1 - z_{1,i}\right) x_j$

- Time complexity
  - $\mathcal{O}\left(n^L\right)$



Learning for DNNs

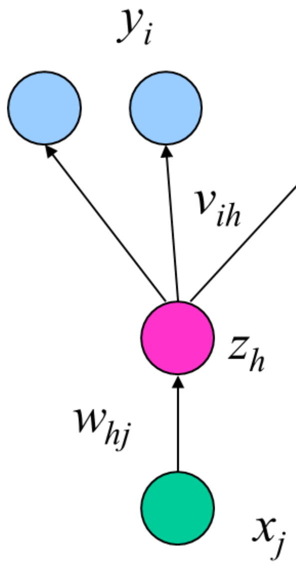


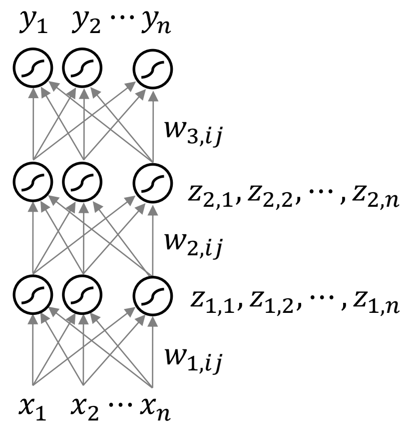
- Weight update for an output node
  - $\frac{\partial E}{\partial w_{3,ij}} = \sum_k \frac{\partial E_k}{\partial y_i} \frac{\partial y_i}{\partial w_{3,ij}} = \sum_k \frac{\partial E_k}{\partial y_i} \frac{\partial y_i}{\partial s_{3,i}} \frac{\partial s_{3,i}}{\partial w_{3,ij}}$
  - $w_{3,ij} \leftarrow w_{3,ij} + \eta \underbrace{(r_i - y_i)} z_{2,j}$; $s_{l,i} \equiv \sum_j w_{l,ij} z_{l-1,j}$
  - $w_{3,ij} \leftarrow w_{3,ij} + \eta \delta_{3,i} z_{2,j}$; $\delta_{3,i} \equiv r_i - y_i = \frac{\partial E}{\partial s_{3,i}}$
- Weight update for a hidden node connected to multiple output nodes
  - $\frac{\partial E}{\partial w_{2,ij}} = \sum_k \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial z_{2,i}} \frac{\partial z_{2,i}}{\partial w_{2,ij}} = \sum_k \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial z_{2,i}} \frac{\partial z_{2,i}}{\partial s_{2,i}} \frac{\partial s_{2,i}}{\partial w_{2,ij}}$
  - $w_{2,ij} \leftarrow w_{2,ij} + \eta \underbrace{\sum_k (r_k - y_k) w_{3,ki} z'_{2,i}} z_{1,j}$
  - $w_{2,ij} \leftarrow w_{2,ij} + \eta \delta_{2,i} z_{1,j}$; $\delta_{2,i} \equiv \sum_k \delta_{3,k} w_{3,ki} z'_{2,i} = \frac{\partial E}{\partial s_{2,i}}$
- Weight update for a hidden node of the first hidden layer in a two hidden layer neural network with multiple output nodes
  - $\frac{\partial E}{\partial w_{1,ij}} = \sum_k \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial z_{1,i}} \frac{\partial z_{1,i}}{\partial w_{1,ij}} = \sum_k \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial z_{1,i}} \frac{\partial z_{1,i}}{\partial s_{1,i}} \frac{\partial s_{1,i}}{\partial w_{1,ij}}$
  - $w_{1,ij} \leftarrow w_{1,ij} + \eta \sum_k (r_k - y_k) \sum_m w_{3,km} z'_{2,m} w_{2,mi} z'_{1,i} x_j$
  - $w_{1,ij} \leftarrow w_{1,ij} + \eta \underbrace{\sum_m \sum_k (r_k - y_k) w_{3,km} z'_{2,m} w_{2,mi} z'_{1,i}} x_j$
  - $w_{1,ij} \leftarrow w_{1,ij} + \eta \delta_{1,i} x_j$; $\delta_{1,i} \equiv \sum_m \delta_{2,m} w_{2,mi} z'_{1,i} = \frac{\partial E}{\partial s_{1,i}}$

Learning for DNNs

- Weight update for an output node
  - $w_{L,ij} \leftarrow w_{L,ij} + \eta \underbrace{\delta_{L,i}}_{\frac{\partial E}{\partial s_{L,i}}} \underbrace{z_{L-1,j}}_{\frac{\partial s_{L,i}}{\partial w_{L,ij}}}$; $\delta_{L,i} \equiv r_i - y_i$, $\partial s_{l,i} \equiv \sum_j w_{l,ij} z_{l-1,j}$

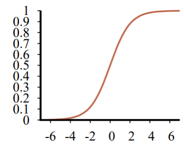
- Weight update for a hidden node of the l-th hidden layer in a multiple hidden layer neural network
  - $w_{l,ij} \leftarrow w_{l,ij} + \eta \underbrace{\delta_{l,i}}_{\frac{\partial E}{\partial s_{l,i}}} \underbrace{z_{l-1,j}}_{\frac{\partial s_{l,i}}{\partial w_{l,ij}}}$; $\delta_{l,i} \equiv \sum_m \delta_{l+1,m} w_{l+1,mi} z'_{l,i}$

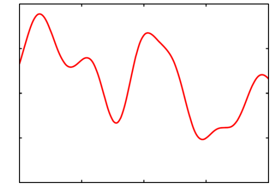




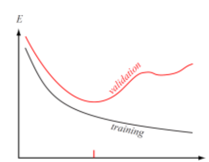
Stochastic Gradient Descent for DNNs

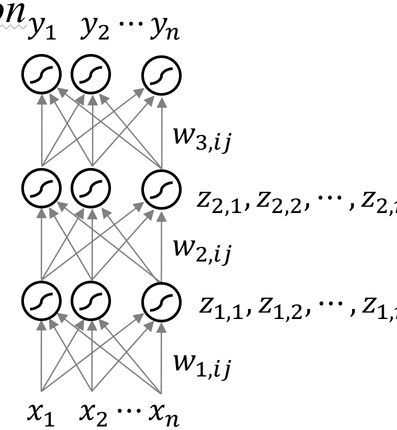
- Input: $\mathcal{D} = \{\boldsymbol{x}^t, \boldsymbol{r}^t\}_{t=1}^{N}$





Initialize $\{\mathbf{W}_l\}_{l=1}^{L}$ with small random numbers.
repeat
  for each $(\boldsymbol{x}^t, \boldsymbol{r}^t) \in \mathcal{D}$ in random order
    for each hidden layer index l from 1 to $L-1$; forward process: feed forward
      for each node index i in the l-th layer
        $z_{l,i} \leftarrow 1/\left(1+\exp\left(-\boldsymbol{w}_{l,i}^{\mathsf{T}} \boldsymbol{z}_{l-1}\right)\right)$; $\boldsymbol{z}_0 \equiv \boldsymbol{x}^t$
      end
    end
    for each output node index i
      $y_i \leftarrow \exp\left(\boldsymbol{w}_{L,i}^{\mathsf{T}} \boldsymbol{z}_{L-1}\right) / \sum_i \exp\left(\boldsymbol{w}_{L,i}^{\mathsf{T}} \boldsymbol{z}_{L-1}\right)$; $y_i \leftarrow \boldsymbol{w}_{L,i}^{\mathsf{T}} \boldsymbol{z}_{L-1}$
    end $y \leftarrow 1/(1+\exp(-\boldsymbol{w}_L^{\mathsf{T}} \boldsymbol{z}_{L-1}))$
    for each output node index i
      $\delta_{L,i} \leftarrow r_i^t - y_i$
    end
    for each layer index l from L to 1 ; backward process
      for each node index j in the $(l-1)$-th layer
        if $l \neq 1$ **then** $\delta_{l-1,j} \leftarrow \sum_i \delta_{l,i} w_{l,ij}\, z'_{l-1,j}$; *error back-propagation*
        for each node index i in the l-th layer
          $w_{l,ij} \leftarrow w_{l,ij} + \eta \delta_{l,i} z_{l-1,j}$
        end
      end
    end
  end
until convergence



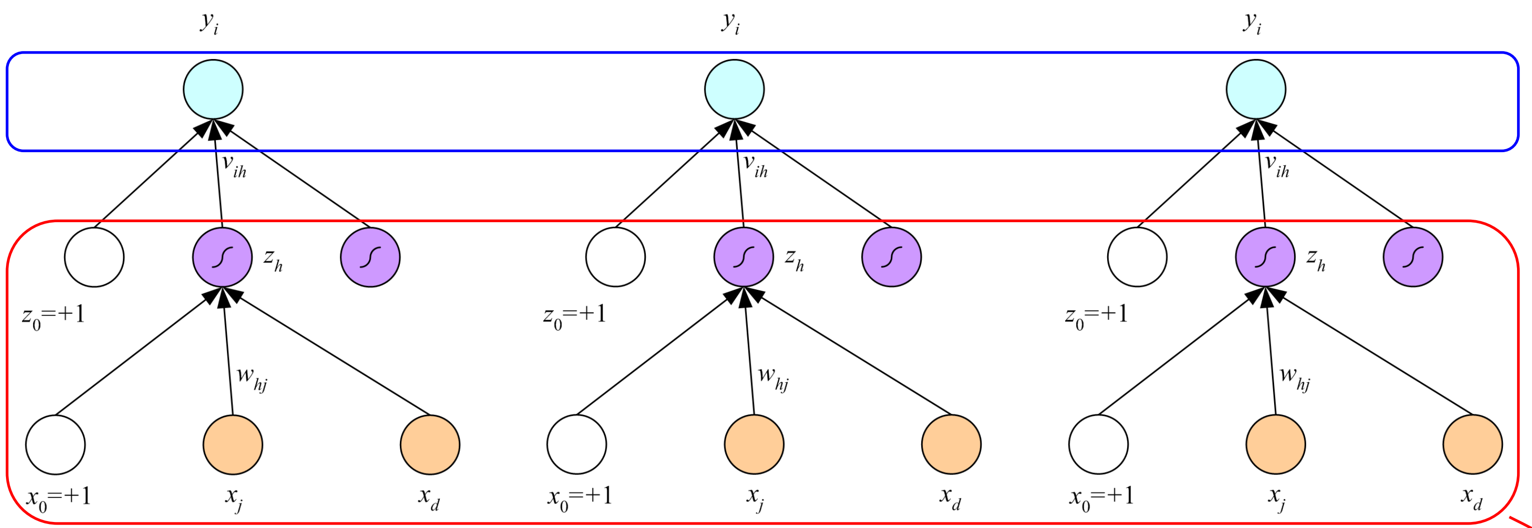


- Output: all $\{\mathbf{W}_l\}_{l=1}^{L}$

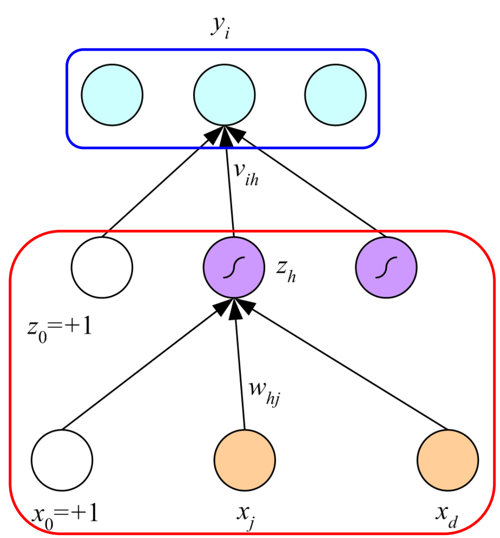
Contents

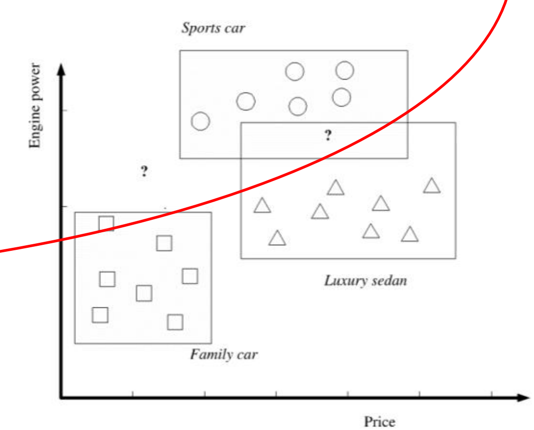
Multilabel Classification

- K independent two-class classification



- Multitask learning





Learning for Multilabel Classification

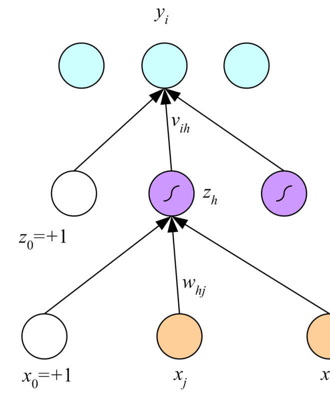
- K dependent two-class classification
  - $E(\mathbf{W}_z, \mathbf{W}_o | \boldsymbol{x}, \boldsymbol{r}) \equiv \sum_k \underbrace{[-r_k \log y_k - (1 - r_k) \log(1 - y_k)]}_{E_k}$

$$r_k = \begin{cases} 1 & \text{if } x \text{ has label } k \\ 0 & \text{otherwise} \end{cases}$$

$$y_k = \frac{1}{1+\exp\left(-\boldsymbol{w}_{o,k}^{\mathsf{T}} \boldsymbol{z}\right)}$$

$$z_i = \frac{1}{1+\exp\left(-\boldsymbol{w}_{z,i}^{\mathsf{T}} \boldsymbol{x}\right)}$$

  - Weight update for K sigmoid output nodes
    - $\frac{\partial E}{\partial w_{o,ij}} = \sum_k \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial w_{o,ij}}$
      $= \underbrace{\left(-\frac{r_i}{y_i} + \frac{1-r_i}{1-y_i}\right)}_{\frac{\partial E_i}{\partial y_i}} \underbrace{y_i(1 - y_i) z_j}_{\frac{\partial y_i}{\partial w_{o,ij}}}$
      $= -(r_i - y_i) z_j$
    - $w_{o,ij} \leftarrow w_{o,ij} + \eta(r_i - y_i) z_j$
  - Weight update for a hidden node connected to K sigmoid output nodes
    - $\frac{\partial E}{\partial w_{z,ij}} = \sum_k \frac{\partial E_k}{\partial y_k} \frac{\partial y_k}{\partial z_i} \frac{\partial z_i}{\partial w_{z,ij}}$
      $= \sum_k \underbrace{\left(-\frac{r_k}{y_k} + \frac{1-r_k}{1-y_k}\right)}_{\frac{\partial E_k}{\partial y_k}} \underbrace{y_k(1 - y_k) w_{o,ki}}_{\frac{\partial y_k}{\partial z_i}} \underbrace{z_i(1 - z_i) x_j}_{\frac{\partial z_i}{\partial w_{z,ij}}}$
      $= -\sum_k (r_k - y_k) w_{o,ki}\, z_i(1 - z_i) x_j$
    - $w_{z,ij} \leftarrow w_{z,ij} + \eta \sum_k (r_k - y_k) w_{o,ki}\, z_i(1 - z_i) x_j$



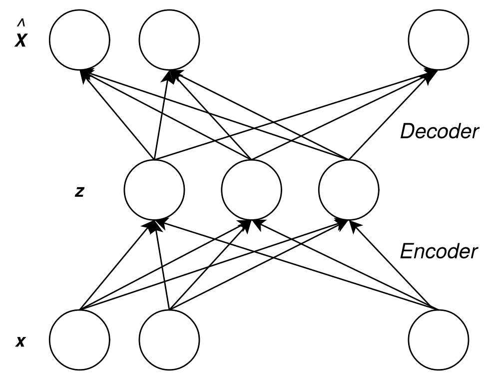
Contents

Autoencoders

- Autoencoders [Cottrell, Munro, and Zipser 1987]
  - $\boldsymbol{z} = \text{Encode}(\boldsymbol{x}|\mathbf{W})$
  - $\tilde{\boldsymbol{x}} = \text{Decode}(\boldsymbol{z}|\mathbf{V})$

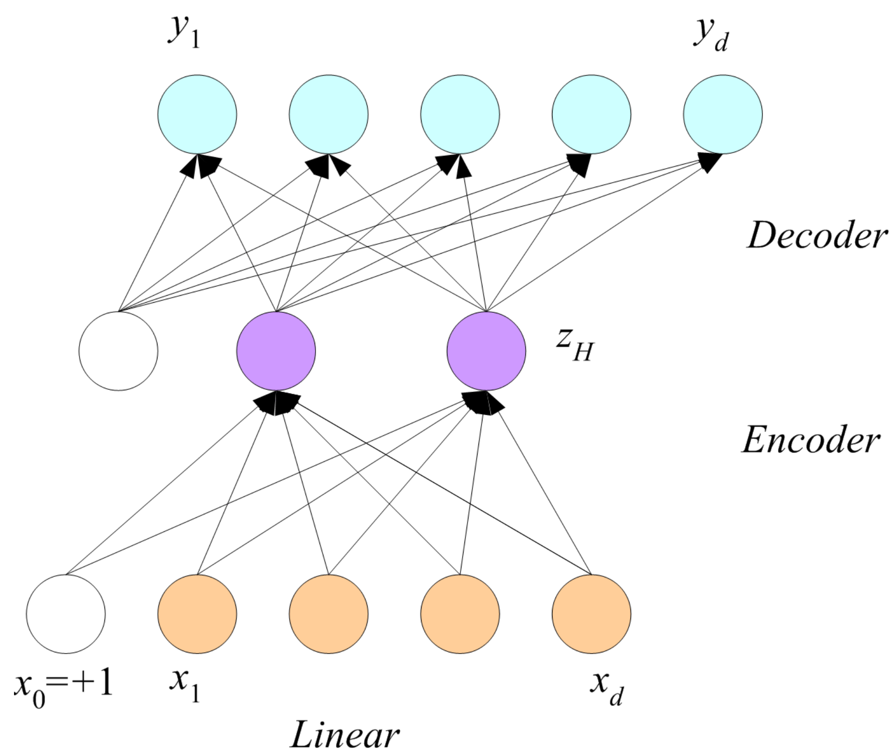
- Reconstruction error of an autoencoder
  - $E(\mathbf{W}, \mathbf{V}|\mathcal{D}) = \frac{1}{N}\sum_t \|\boldsymbol{x}^t - \tilde{\boldsymbol{x}}^t\|^2$; $\|\boldsymbol{x}\|_2 \equiv \sqrt{\sum_i x_i^2}$ (Euclidean norm or ℓ^2 norm)

    $= \frac{1}{N}\sum_t \|\boldsymbol{x}^t - \text{Decode}(\text{Encode}(\boldsymbol{x}^t|\mathbf{W})|\mathbf{V})\|^2$

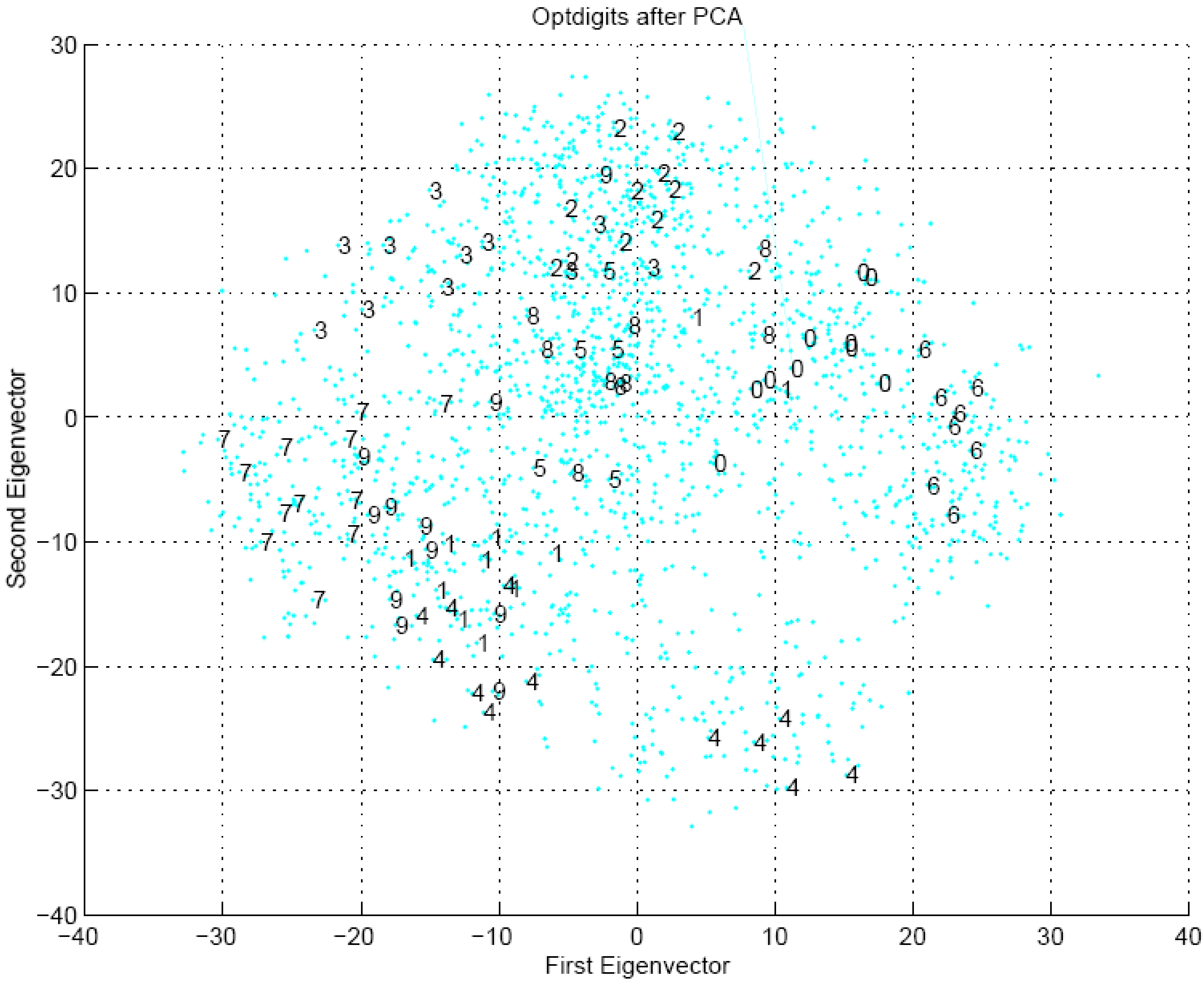


Dimensionality Reduction Using Autoencoders

- Dimensionality reduction using linear autoencoders



Dimensionality Reduction Example

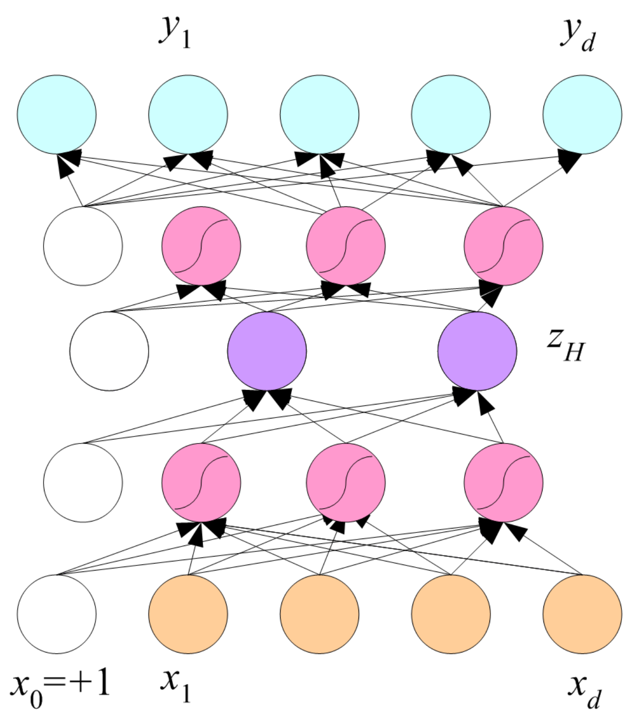


Variants of Autoencoders

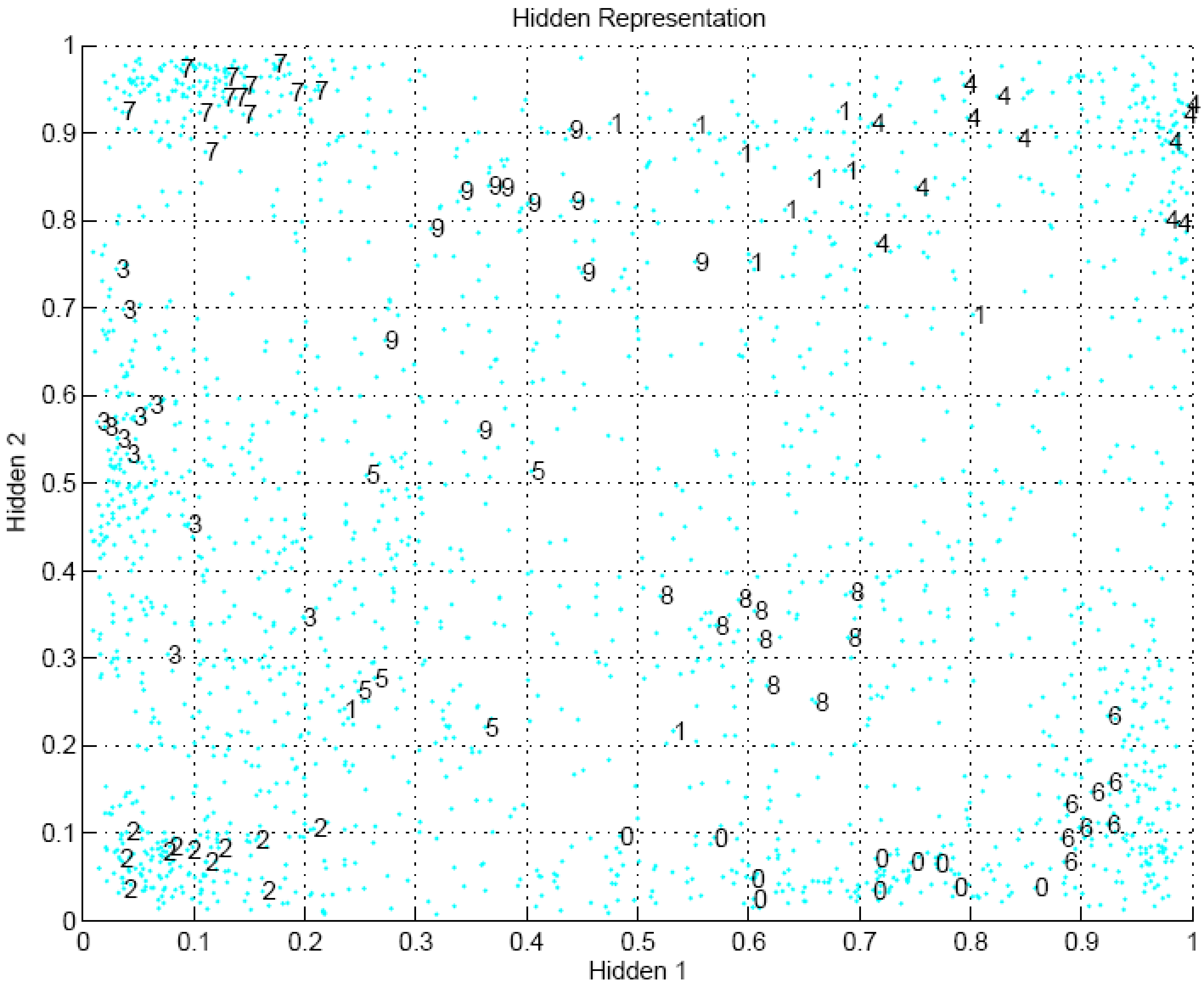
- Denoising autoencoder [Vincent *et al*. 2008]
  - Noiseless and noisy versions of the same instance with the same desired output

- Sparse autoencoder [Ranzato *et al*. 2007]
  - dimensionality of input < dimensionality of hidden layer

- Deep autoencoder and stacked autoencoder



Dimensionality Reduction Example

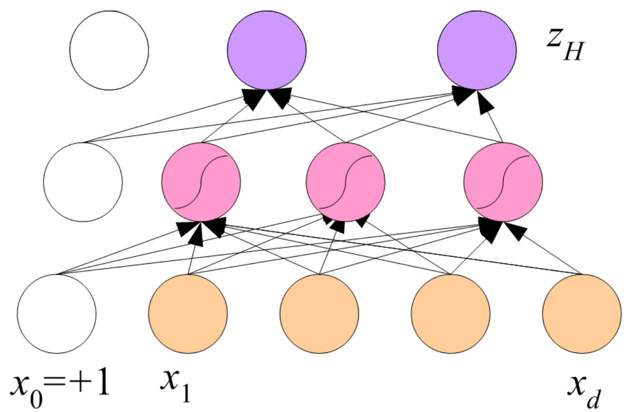


Multidimensional Scaling

- Multidimensional scaling (MDS)
  - high-dimensional space $\Rightarrow$ low-dimensional space
  - Trying to keep the inter-point distances unchanged
  - Dimensionality reduction

- Sammon mapping
  - $\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x}|\theta)$ where $\boldsymbol{x} \in \mathcal{R}^d$, $\boldsymbol{z} \in \mathcal{R}^k$, and $k < d$
  - Sammon stress: $\sum_{t,s} \frac{\left(\|\boldsymbol{g}(\boldsymbol{x}^t|\theta) - \boldsymbol{g}(\boldsymbol{x}^s|\theta)\| - \|\boldsymbol{x}^t - \boldsymbol{x}^s\|\right)^2}{\|\boldsymbol{x}^t - \boldsymbol{x}^s\|^2} = \sum_{t,s} \frac{\left(\|\boldsymbol{z}^t - \boldsymbol{z}^s\| - \|\boldsymbol{x}^t - \boldsymbol{x}^s\|\right)^2}{\|\boldsymbol{x}^t - \boldsymbol{x}^s\|^2}$

- An MLP with d inputs, hidden units, and $k < d$ output units



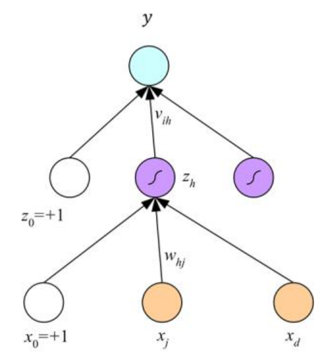
Contents

- Perceptrons
- Multilayer Perceptrons
- Error Backpropagation Algorithm
- Applications
  - Multilabel Classification
  - Dimensionality Reduction
  - Representation Learning

Representation Learning

- Linear model and nonlinear model
  - $y = \sum_i v_i x_i + v_0$
  - $y = \sum_i v_i \phi_i(\boldsymbol{x})$; ϕ_i: basis function e.g., polynomial basis function

- Embedding (also called code or *representation*)
  - $y = \sum_i v_i \phi(\boldsymbol{x}|\boldsymbol{w}_i)$; $\phi(\boldsymbol{x}|\boldsymbol{w}_i) \equiv \frac{1}{1+\exp(-\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x})}$
  - Learnable basis function
  - A new space where the problem becomes linear, e.g., XOR
  - Unsupervised learning, e.g., autoencoder



- Dimensionality reduction: dimensionality of embedding < dimensionality of input

- Transfer learning: using the embedding trained on one task as input to another task

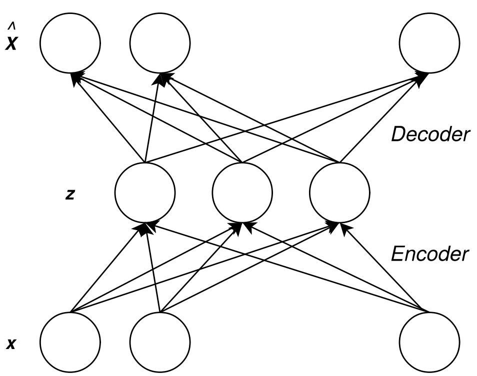
- Multitask learning: learning similar tasks with shared hidden layers

- Self-supervised learning and finetuning: using the embedding trained on a large unlabeled data set as the input in supervised learning with a small labeled data set

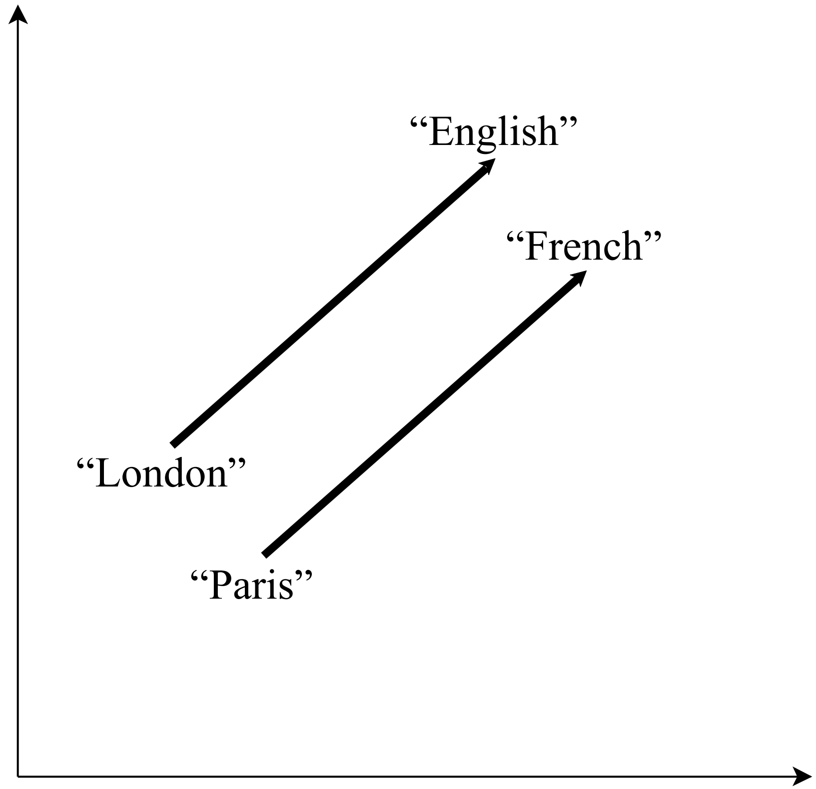
Word Embedding

- Binary vector representation of words
  - d-dimensional one-hot vector ; bag of words
  - High dimensional
  - No syntactic/semantic relationship

- Continuous vector representation of words (WORD2VEC, Mikolov *et al*. 2013)
  - Input: context (windows of words)
    - Each word in the window at a time (SKIP-GRAM)
      e.g., “Protesters in Paris clash with the French police.”
  - Hidden layer: less than d hidden units
  - Output: center word (d-dimensional one-hot vector)



Word Embedding Example



$$\textsc{vec}(\text{“French”}) - \textsc{vec}(\text{“Paris”}) + \textsc{vec}(\text{“London”}) = \textsc{vec}(\text{“English”})$$

Summary and Preview

- Perceptrons
  - Introduction
  - Learning for Perceptrons
- Multilayer Perceptrons
  - Boolean Functions
  - Multilayer Perceptrons
- Error Backpropagation Algorithm
  - Regression
  - Classification
  - Deep Neural Networks
- Applications
  - Multilabel Classification
  - Dimensionality Reduction
  - Representation Learning

- Architecture of DNNs