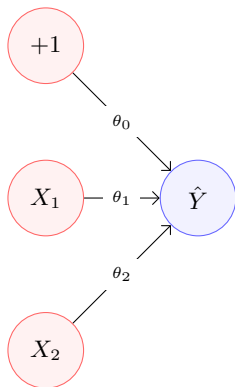


# Artificial Neural Networks

Linda Mawhinney

# Logistic Regression



$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} \\ \vdots & \vdots & \vdots \\ 1 & x_{m,1} & x_{m,2} \end{bmatrix} \in \mathbb{R}^{m \times 3}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \in \mathbb{R}^3$$

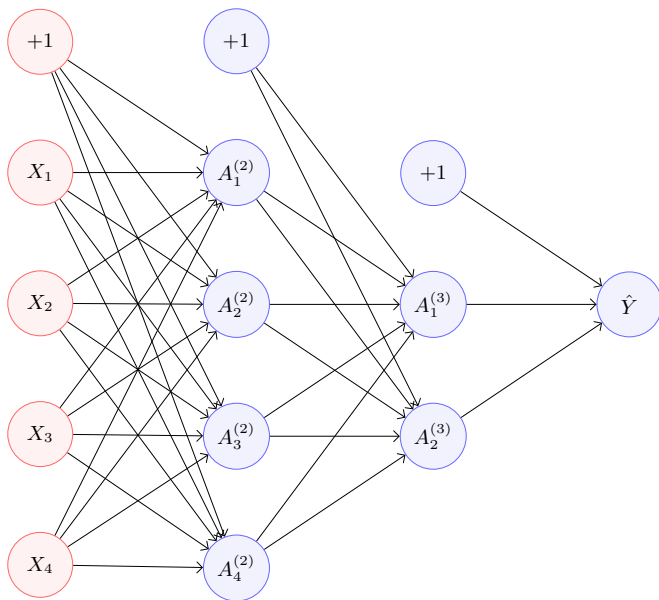
$$\hat{Y} = \sigma(X\theta) \in [0, 1]^m$$

$$\text{where } \sigma(z) = \frac{1}{1+e^{-z}}$$

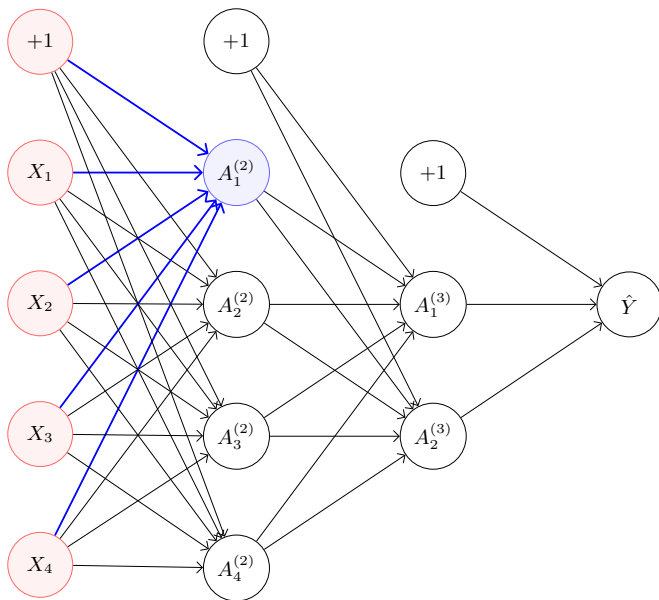
Cost function:

$$J(\theta) = \sum (y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$$

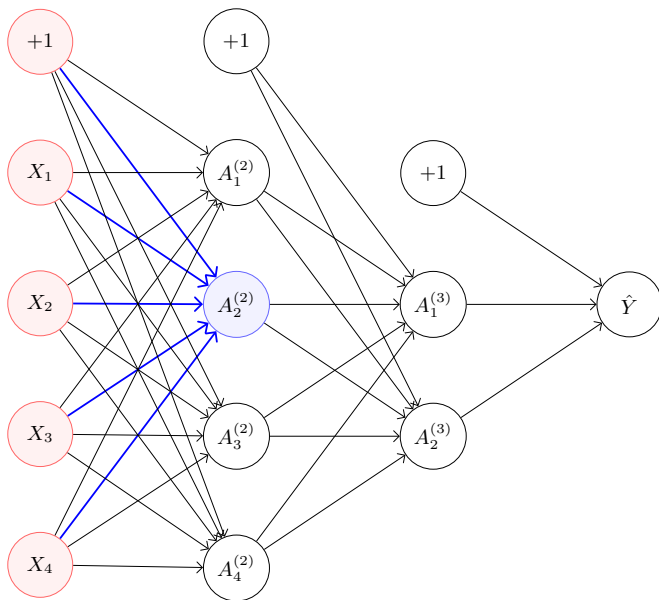
# Hierarchy of Logistic Regressions



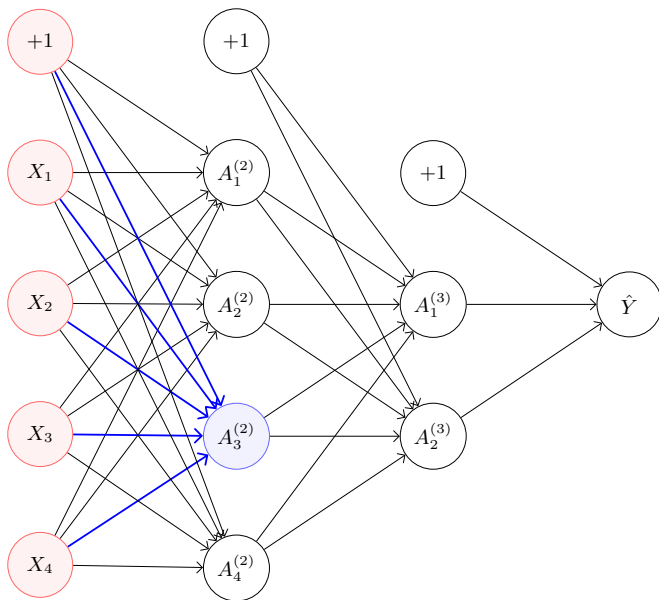
## Hierarchy of Logistic Regressions



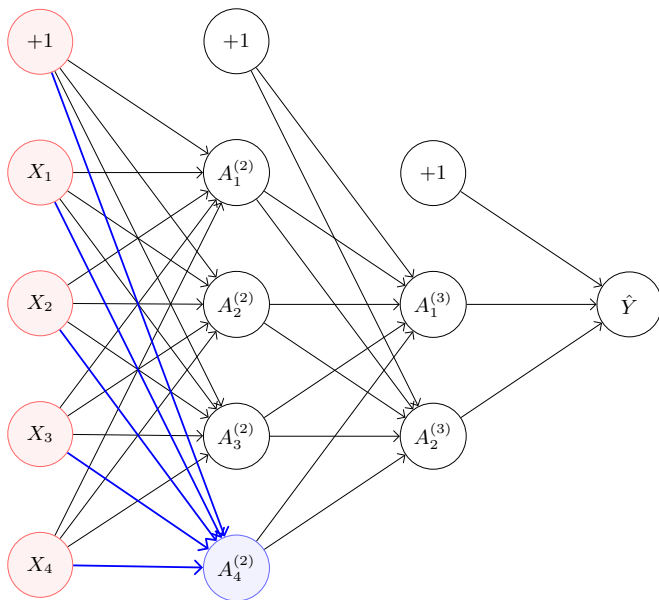
## Hierarchy of Logistic Regressions



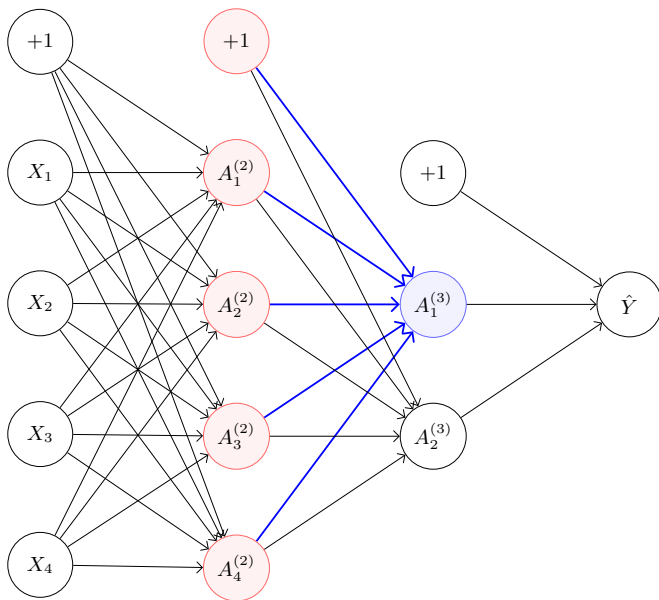
## Hierarchy of Logistic Regressions



## Hierarchy of Logistic Regressions

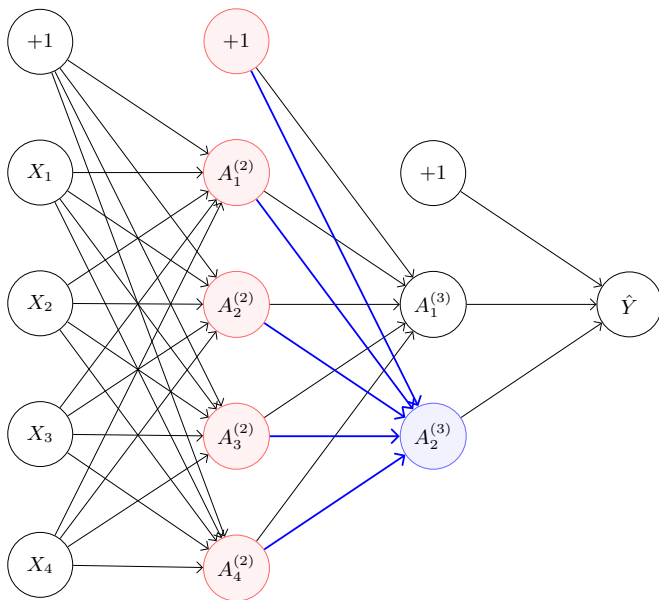


## Hierarchy of Logistic Regressions

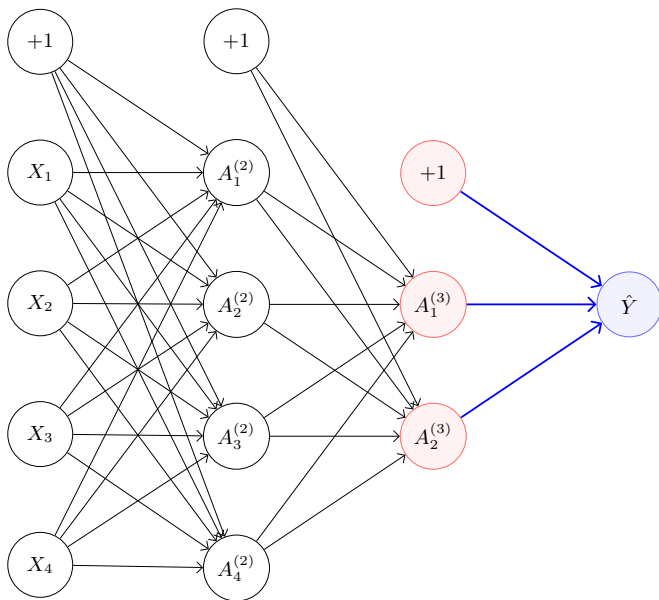




## Hierarchy of Logistic Regressions

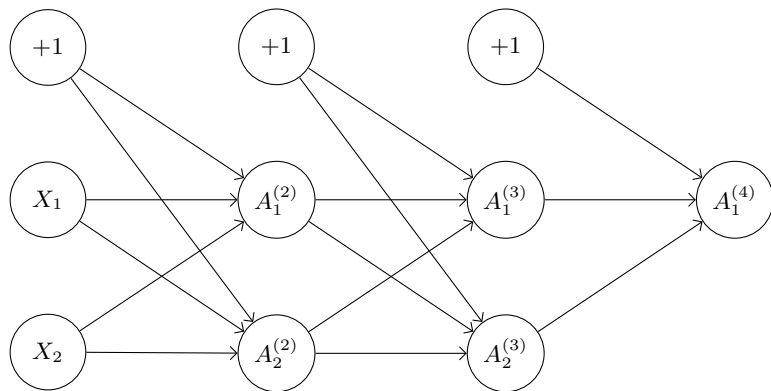


## Hierarchy of Logistic Regressions

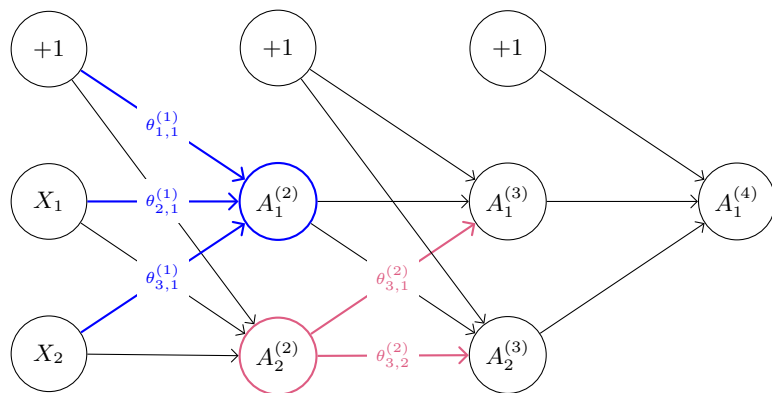


# Learning

## Algorithm: 2 Hidden Layers, 2 Hidden Nodes



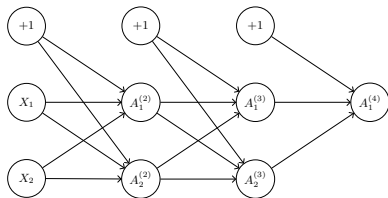
## Step 1. Initialize Weights



$$\theta^{(1)} = \begin{bmatrix} \theta_{1,1}^{(1)} & \theta_{1,2}^{(1)} \\ \theta_{2,1}^{(1)} & \theta_{2,2}^{(1)} \\ \theta_{3,1}^{(1)} & \theta_{3,2}^{(1)} \end{bmatrix}, \quad \theta^{(2)} = \begin{bmatrix} \theta_{1,1}^{(2)} & \theta_{1,2}^{(2)} \\ \theta_{2,1}^{(2)} & \theta_{2,2}^{(2)} \\ \theta_{3,1}^{(2)} & \theta_{3,2}^{(2)} \end{bmatrix}, \quad \theta^{(3)} = \begin{bmatrix} \theta_{1,1}^{(3)} \\ \theta_{2,1}^{(3)} \\ \theta_{3,1}^{(3)} \end{bmatrix}$$

## Step 2. Feed Forward

- ▶  $A^{(1)} \leftarrow [1 \ X]$   
 $Z^{(2)} = A^{(1)}\theta^{(1)}$   
 $A^{(2)} = \sigma(Z^{(2)})$
- ▶  $A^{(2)} \leftarrow [1 \ A^{(2)}]$   
 $Z^{(3)} = A^{(2)}\theta^{(2)}$   
 $A^{(3)} = \sigma(Z^{(3)})$
- ▶  $A^{(3)} \leftarrow [1 \ A^{(3)}]$   
 $Z^{(4)} = A^{(3)}\theta^{(3)}$   
 $A^{(4)} = \sigma(Z^{(4)})$

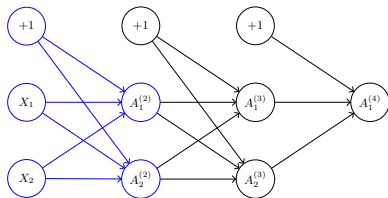


## Step 2. Feed Forward

$$\begin{aligned} \blacktriangleright A^{(1)} &\leftarrow [1 \quad X] \\ Z^{(2)} &= A^{(1)}\theta^{(1)} \\ A^{(2)} &= \sigma(Z^{(2)}) \end{aligned}$$

$$\begin{aligned} \blacktriangleright A^{(2)} &\leftarrow [1 \quad A^{(2)}] \\ Z^{(3)} &= A^{(2)}\theta^{(2)} \\ A^{(3)} &= \sigma(Z^{(3)}) \end{aligned}$$

$$\begin{aligned} \blacktriangleright A^{(3)} &\leftarrow [1 \quad A^{(3)}] \\ Z^{(4)} &= A^{(3)}\theta^{(3)} \\ A^{(4)} &= \sigma(Z^{(4)}) \end{aligned}$$

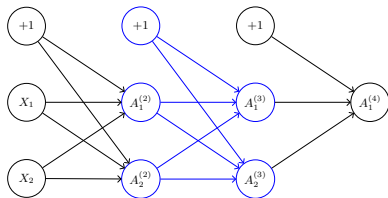


## Step 2. Feed Forward

$$\begin{aligned} \blacktriangleright A^{(1)} &\leftarrow \begin{bmatrix} 1 & X \end{bmatrix} \\ Z^{(2)} &= A^{(1)}\theta^{(1)} \\ A^{(2)} &= \sigma(Z^{(2)}) \end{aligned}$$

$$\begin{aligned} \blacktriangleright A^{(2)} &\leftarrow \begin{bmatrix} 1 & A^{(2)} \end{bmatrix} \\ Z^{(3)} &= A^{(2)}\theta^{(2)} \\ A^{(3)} &= \sigma(Z^{(3)}) \end{aligned}$$

$$\begin{aligned} \blacktriangleright A^{(3)} &\leftarrow \begin{bmatrix} 1 & A^{(3)} \end{bmatrix} \\ Z^{(4)} &= A^{(3)}\theta^{(3)} \\ A^{(4)} &= \sigma(Z^{(4)}) \end{aligned}$$



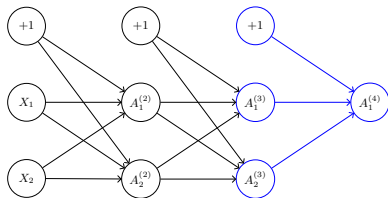


## Step 2. Feed Forward

$$\begin{aligned} \blacktriangleright A^{(1)} &\leftarrow \begin{bmatrix} 1 & X \end{bmatrix} \\ Z^{(2)} &= A^{(1)}\theta^{(1)} \\ A^{(2)} &= \sigma(Z^{(2)}) \end{aligned}$$

$$\begin{aligned} \blacktriangleright A^{(2)} &\leftarrow \begin{bmatrix} 1 & A^{(2)} \end{bmatrix} \\ Z^{(3)} &= A^{(2)}\theta^{(2)} \\ A^{(3)} &= \sigma(Z^{(3)}) \end{aligned}$$

$$\begin{aligned} \blacktriangleright A^{(3)} &\leftarrow \begin{bmatrix} 1 & A^{(3)} \end{bmatrix} \\ Z^{(4)} &= A^{(3)}\theta^{(3)} \\ A^{(4)} &= \sigma(Z^{(4)}) \end{aligned}$$

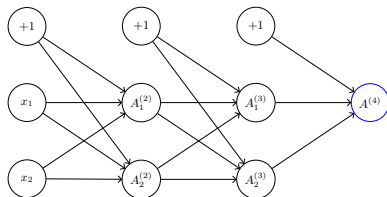


### Step 3. Back Propagation

►  $\delta^{(4)} = A^{(4)} - Y$

►  $\delta^{(3)} = (\delta^{(4)} \dot{\theta}^{(3)\top}) * \sigma'(Z^{(3)})$

►  $\delta^{(2)} = (\delta^{(3)} \dot{\theta}^{(2)\top}) * \sigma'(Z^{(2)})$



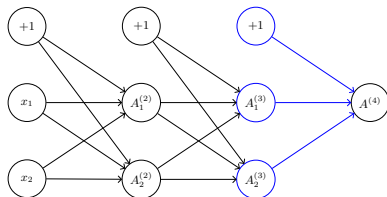
where  $\dot{\theta}^{(j)}$  is all but first row of  $\theta^{(j)}$ ,  $*$  is the entry-wise product and  $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ .

### Step 3. Back Propagation

►  $\delta^{(4)} = A^{(4)} - Y$

►  $\delta^{(3)} = (\delta^{(4)} \dot{\theta}^{(3)\top}) * \sigma'(Z^{(3)})$

►  $\delta^{(2)} = (\delta^{(3)} \dot{\theta}^{(2)\top}) * \sigma'(Z^{(2)})$



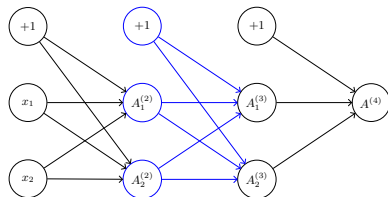
where  $\dot{\theta}^{(j)}$  is all but first row of  $\theta^{(j)}$ ,  $*$  is the entry-wise product and  $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ .

### Step 3. Back Propagation

►  $\delta^{(4)} = A^{(4)} - Y$

►  $\delta^{(3)} = (\delta^{(4)} \dot{\theta}^{(3)\top}) * \sigma'(Z^{(3)})$

►  $\delta^{(2)} = (\delta^{(3)} \dot{\theta}^{(2)\top}) * \sigma'(Z^{(2)})$



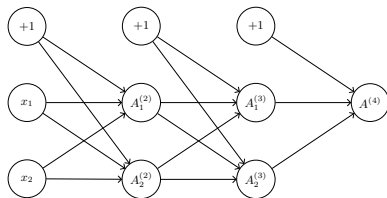
where  $\dot{\theta}^{(j)}$  is all but first row of  $\theta^{(j)}$ ,  $*$  is the entry-wise product and  $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ .

### Step 3. Back Propagation

$$\blacktriangleright \delta^{(4)} = A^{(4)} - Y$$

$$\blacktriangleright \delta^{(3)} = (\delta^{(4)} \dot{\theta}^{(3)\top}) * \sigma'(Z^{(3)})$$

$$\blacktriangleright \delta^{(2)} = (\delta^{(3)} \dot{\theta}^{(2)\top}) * \sigma'(Z^{(2)})$$



where  $\dot{\theta}^{(j)}$  is all but first row of  $\theta^{(j)}$ ,  $*$  is the entry-wise product and  $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ .

$$\frac{\partial J}{\partial \theta^{(1)}} = A^{(1)\top} \delta^{(2)}$$

$$\frac{\partial J}{\partial \theta^{(2)}} = A^{(2)\top} \delta^{(3)}$$

$$\frac{\partial J}{\partial \theta^{(3)}} = A^{(3)\top} \delta^{(4)}$$

# Coding in R

## R Code

```
initialize.weights <- function(n.hidden.nodes,  
                                n.hidden.layers,  
                                n.in.nodes,  
                                n.out.nodes) {  
  
  W <- list()  
  
  W[[1]] <- random.normal.matrix(n.in.nodes+1, n.hidden.nodes)  
  
  W[[n.hidden.layers+1]] <- random.normal.matrix(n.hidden.nodes+1, n.out.nodes)  
  
  if (n.hidden.layers > 1){  
    for (i in 2:n.hidden.layers) {  
      W[[i]] <- random.normal.matrix(n.hidden.nodes+1, n.hidden.nodes)  
    }  
  }  
  
  W  
}
```

## R Code

```
feed.forward <- function(W, X){  
  n.hidden.layers <- length(W) - 1  
  A <- list(add.bias.column(X))  
  Z <- list(NULL)  
  for(i in 1:n.hidden.layers){  
    Z[[i+1]] <- A[[i]] %*% W[[i]]  
    A[[i+1]] <- add.bias.column(sigmoid(Z[[i+1]]))  
  }  
  Z[[n.hidden.layers+2]] <- A[[n.hidden.layers+1]] %*% W[[n.hidden.layers+1]]  
  A[[n.hidden.layers+2]] <- sigmoid(Z[[n.hidden.layers+2]])  
  list(A=A, Z=Z)  
}
```



## R Code

```
back.prop <- function(A,
                      W,
                      Z,
                      y,
                      n.hidden.layers) {

  delta <- list()

  D <- list()

  delta[[n.hidden.layers+2]] <- A[[n.hidden.layers+2]] - y

  if (n.hidden.layers >= 1){

    for (i in (n.hidden.layers+1):2){
      delta[[i]] <- (delta[[i+1]] %*% t(submatrix(W[[i]], 2))) * sigmoid.grad(Z[[i]])
    }

    for (i in 1:(n.hidden.layers+1)){
      D[[i]] <- t(A[[i]]) %*% delta[[i+1]]
    }

  }

  D

}
```

## R Code

```
nn <- function(train.X,  
               train.y,  
               n.hidden.layers,  
               n.hidden.nodes,  
               alpha=0.0001,  
               tolerance=0.0000001,  
               maxiter=1000000) {  
  
  unroll.y <- unroll.matrix(train.y)  
  
  n.in.nodes <- ncol(train.X)  
  n.out.nodes <- ncol(unroll.y)  
  n <- nrow(train.X)  
  
  W <- initialize.weights(n.hidden.nodes, n.hidden.layers, n.in.nodes, n.out.nodes)  
  
  new <- feed.forward(W, train.X)  
  A <- new$A  
  Z <- new$Z  
  
  loss <- log.loss(A[[n.hidden.layers+2]], unroll.y)
```

## R Code

```
num.iterations <- 0
converged <- FALSE

while (!converged && num.iterations < maxiter) {

  num.iterations <- num.iterations + 1

  D <- back.prop(A, W, Z, unroll.y, n.hidden.layers)

  for (i in 1:(n.hidden.layers+1)) {
    W[[i]] <- W[[i]] - alpha * D[[i]]
  }

  new <- feed.forward(W, train.X)
  A <- new$A
  Z <- new$Z

  new.loss <- log.loss(A[[n.hidden.layers+2]], unroll.y)
  converged <- abs((new.loss - loss)/loss) < tolerance
  loss <- new.loss

}

list(W=W, iterations=num.iterations, loss=loss)

}
```

# Classifying Handwritten Digits

## MNIST data set: Classifying hand written digits

Data from images of hand-drawn digits from zero through nine.



← 28 pixels →

- ▶ Each image is 784 pixels in total.
- ▶ Each pixel has a single pixel-value associated to it.
- ▶ Higher numbers mean darker.
- ▶ The pixel value is an integer between 0 and 225 inclusive.

# MNIST data set

Visually, the pixels make up the image as follows:

000	001	002	003	...	027
028	029	030	031	...	055
056	057	058	059	...	083
⋮	⋮	⋮	⋮	...	⋮
728	729	730	731	...	755
756	757	758	759	...	783

## 12,000 images

- ▶ 10,000 train
- ▶ 2,000 test

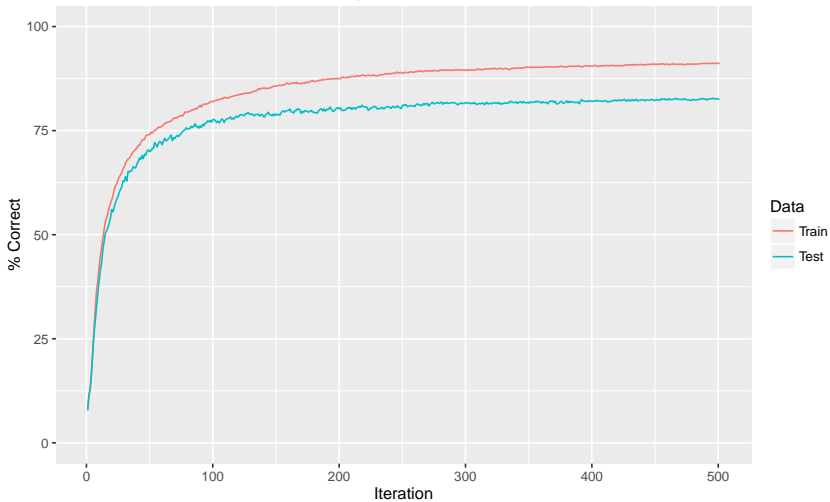
## 784 columns

- ▶ Pixel 0 - Pixel 783
- ▶ Pixel-value

## Examples

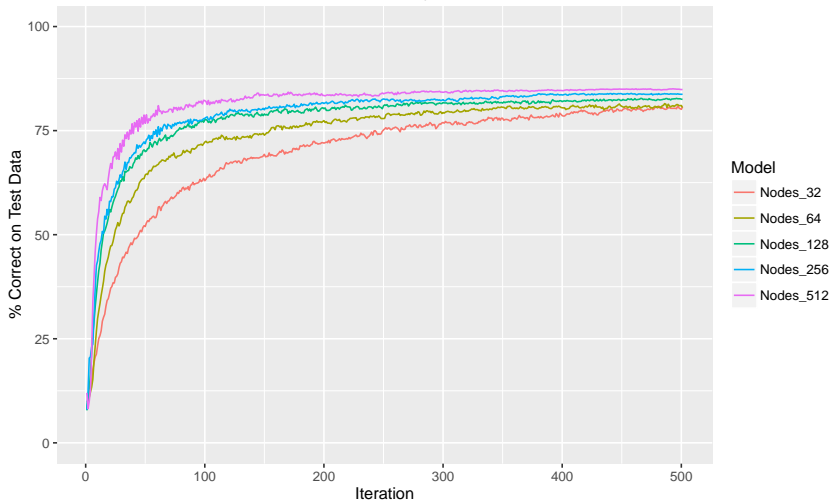


1 Hidden Layer, 128 Hidden Nodes

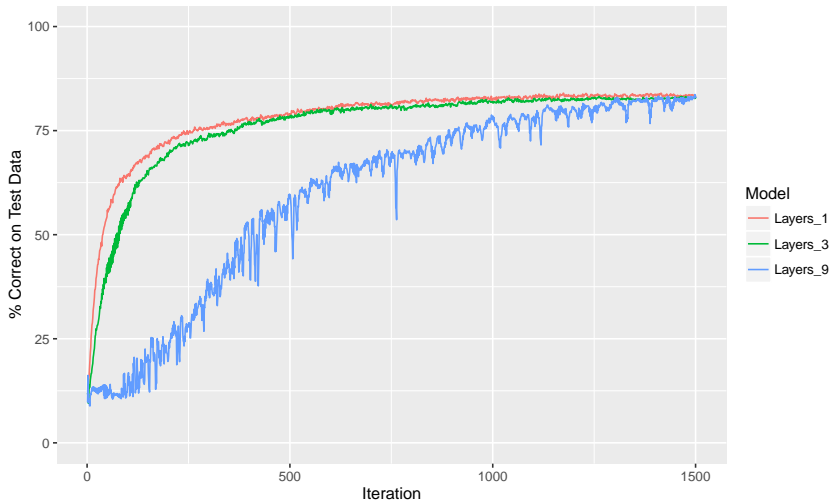




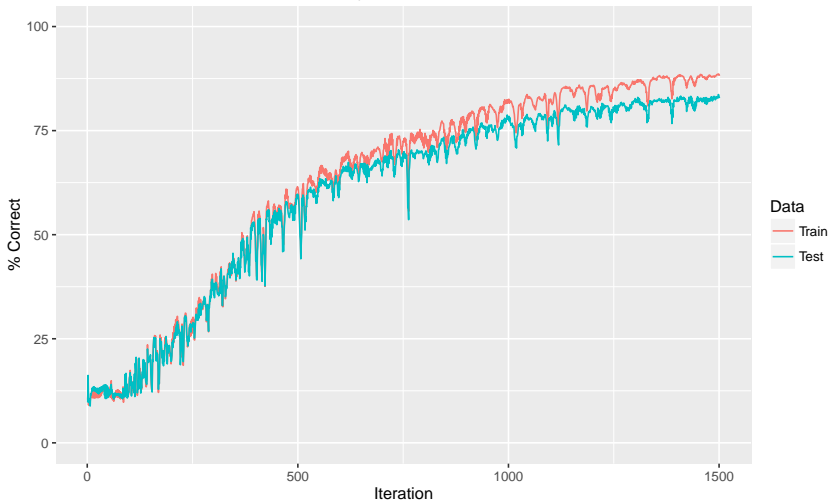
## 1 Hidden Layer



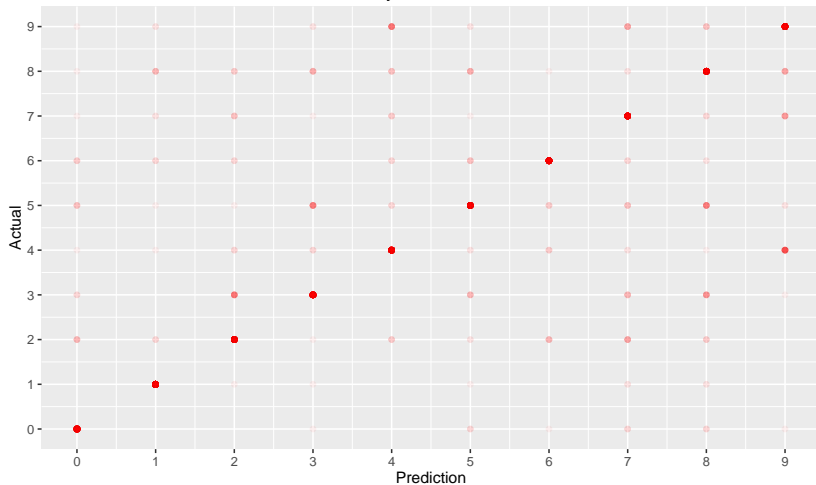
### 32 Hidden Nodes



9 Hidden Layers, 32 Hidden Nodes



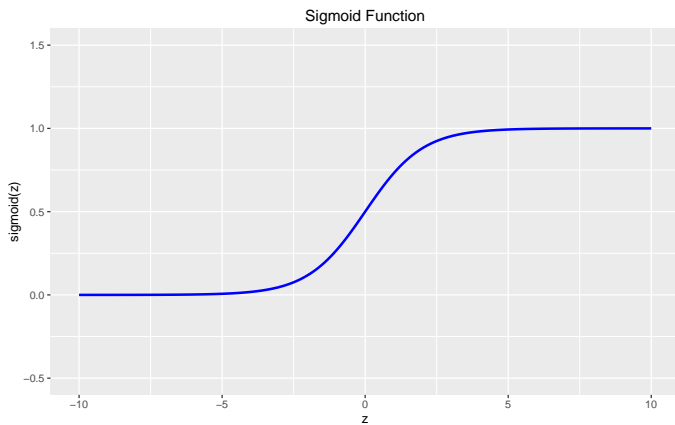
1 Hidden Layer, 512 Hidden Nodes



# More on Neural Networks

# Activation Functions

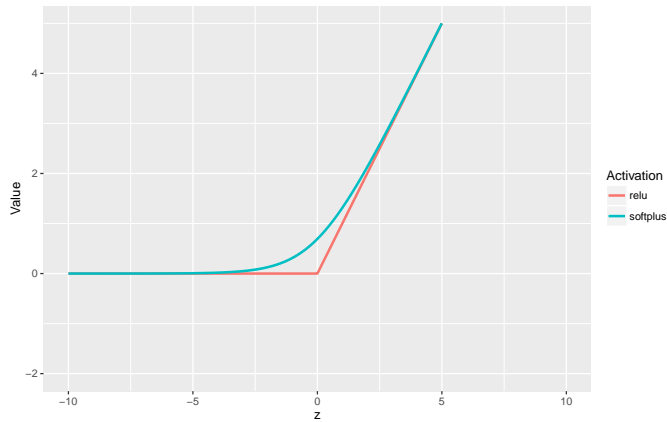
$$\sigma(z) = \frac{1}{1+e^{-z}}$$



# Activation Functions

$$\text{ReLU}(z) = \max(z, 0)$$

$$\text{softplus}(z) = \ln(1 + e^z)$$



# Some Issues in Training Neural Networks

- ▶ Symmetry breaking
- ▶ Choosing architecture
- ▶ Overfitting
- ▶ Error function is non-convex



# References

- [1] T. Hastie and R. Tibshirani and J. Freidman, *The Elements of Statistical Learning*, Springer, 2009
- [2] A. Ng, *An Introduction to Machine Learning*, Coursera, 2016
- [3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11):2278-2324, November 1998
- [4] M. Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015