# Lecture:
# Machine Learning

VCU
College of Engineering

# Don’t forget to hit the record button

# Literature Review

## Matrix Examples

(show spreadsheets)

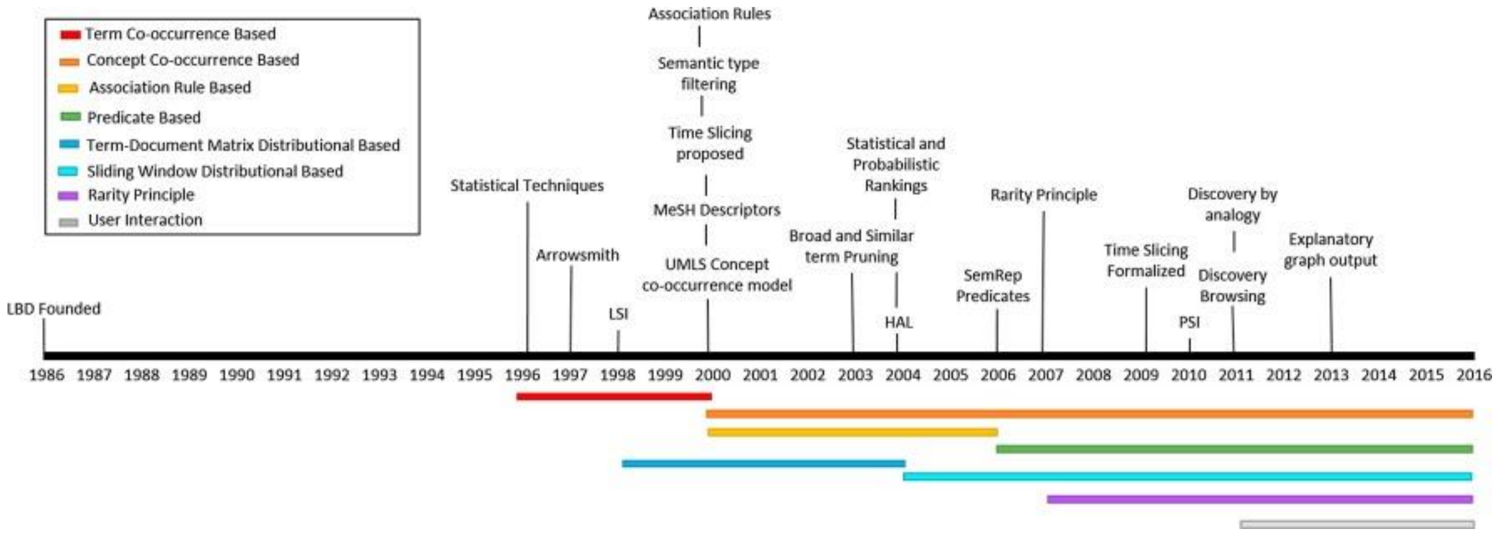
| Author | Discovery replicated | Dates used |
|---|---|---|
| Gordon and Lindsay [32] | Raynaud's disease & fish oil | 1983–1985 |
| Hu et al. [63] | Raynaud's disease & fish oil | 1980–1985 |
| | Migraine & magnesium | 1980–1984 |
| Weeber et al. [27] | Raynaud's disease & fish oil | 1960–1986 |
| | Migraine & magnesium | 1960–1986 |
| | Raynaud's disease & fish oil | 1960–1985 |
| | Somatomedin C & arginine | 1960–1989 |
| | Migraine & magnesium | 1980–1984 |
| Preiss et al. [34] | Magnesium deficiency & neurologic disease | 1966–1994[a] |
| | Alzheimer's disease & indomethacin | 1966–1996 |
| | Alzheimer's disease & estrogen | 1974-June 1995[a] |
| | Schizophrenia & calcium-independent phospholipase A2 | 1960–1997 |

Table 4: End-to-End Clinical Timeline Extraction Systems

| Tool: | MedLEE [106] | Deep NLU [107] | Dehghan [54] | Najafabadipour [53] |
|---|---|---|---|---|
| Year | 2005 | 2011 | 2014 | 2020 |
| TIMEX | Rule-based: Temporal Constraint Structure (TCS), explicit focus | Rule-based: Logic Form Parsing, explicit focus | Hybrid: CRF + Rules | Rule-based: Temporal Tagger |
| EVENT | Rule-based | Rule-based: LF Parsing | Hybrid: CRF + Rules | Rule-based: C-liKES and TNM Annotator |
| TLINK | Rule-based | Rule-based: LF Parsing | Rule-based | Hybrid: UDPipe + Rules |
| Event Normalization | Rule-based: absolute reference chosen over implicit | X | Rule-based | Rule-based: Semantic Similarity + Temporal Rules |
| Event Ordering | Temporal discourse, Simple Temporal Problems (STP) Graph | Assigns intervals to events | PathCluster | ule-based: Selects earliest occurrence of unique events, sorts temporally |
| Visualization | X | Simile Timeline Widget | PathVisualization | X |
| Evaluation | X | X | Precision and Recall | P,R,F1 for components, manual for timeline system |
| Scope | Single Document | Single Document | Multiple Documents | Multiple Documents |

| | |
|---|---|
| *Term co-occurrence* | |
| Gordon and Lindsay [32] | Relative frequency |
| Hristovski et al. [69] | Confidence[a] |
| Hristovski et al. [36] | Support |
| Swanson et al. [70] | Literature cohesiveness (COH) |
| Cole and Bruza [50] | Odds-ratio |
| Stegmann and Grohmann [71] | Equivalence index |
| *Measures of independence* | |
| Yetisgen-Yildiz and Pratt [35], [68] | Z-score |
| Wren et al. [67] | Mutual Information Measure (MIM) |
| Cole and Bruza [50] | Log Likelihood (ll) |
| *Semantic predication* | |
| Hristovski et al. [72] | Predication frequency |
| Wilkowski et al. [53] | Degree centrality |
| Cameron et al. [73] | Intra-cluster predication similarity |
| *Nearest neighbor search* | |
| Gordon and Dumais [30] | Cosine distance |
| Bruza et al. [33] | Euclidean distance |
| Bruza et al. [33] | Information flow |
| *Implicit term* | |
| Hristovski et al. [69] | $X \rightarrow Z$ Support |
| Wren et al. [67] | Average Mutual Information Measure (AMIM) |
| Wren et al. [67] | Minimum Mutual Information Measure (MMIM) |
| Wren et al. [67] | Average Minimum Weight (AMW) |

| Author | Model | Document representation | Term representation | Filters | | | Evaluation | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | S | R | H | R | P | T |
| Kostoff et al. [58] | bibliometric | abstracts,titles,MesH | n-grams/MeSH | X | | | X | X | |
| Gordon and Lindsay [32] | co-occurrence | abstracts,titles | n-grams | | | | X | | |
| Weeber et al. [27] | co-occurrence | abstracts,titles | CUIs | X | | | X | | |
| Wren et al. [79] | co-occurrence | abstracts,titles | CUIs | | | | | X | |
| Hu et al. [63] | co-occurrence | MeSH | CUIs | X | X | X | X | | |
| Hristovski et al. [36] | co-occurrence | MeSH | CUIs | X | | | | | X |
| Srinivasan [28] | co-occurrence | MeSH | CUIs | X | | | X | | |
| Yetisgen-Yildiz [64] | co-occurrence | MeSH | CUIs | X | | X | | X | X |
| Stegmann and Grohmann [71] | co-occurrence | MeSH | CUIs | | | | X | X | |
| Pratt and Yetisgen-Yildiz [60] | co-occurrence | titles | CUIs | X | | X | X | | |
| Swanson and Smalheiser [26] | co-occurrence | titles | unigrams | | | | X | X | |
| Preiss [43] | semantic | abstracts,titles | CUIs | X | | X | X | | |
| van der Eijk et al. [29] | distributional | abstracts | CUIs | | | | | X | |
| Gordon and Dumais [30] | distributional | abstracts,titles,MesH | n-grams | | | | X | | |
| Bruza et al. [33] | distributional | titles | unigrams | | | | X | | |
| Cohen et al. [61] | distributional | SemMedDB | CUIs | X | X | | | X | |
| Wilkowski et al. [53] | interactive | SemMedDB | CUIs | X | X | | | X | |
| Workman et al. [56] | interactive | SemMedDB | CUIs | X | X | | | X | |
| Petric et al. [31] | rarity | PMC Full Text | n-grams | X | | | | X | |



Term Co-occurrence Based
Concept Co-occurrence Based
Association Rule Based
Predicate Based
Term-Document Matrix Distributional Based
Sliding Window Distributional Based
Rarity Principle
User Interaction
LBD Founded
Statistical Techniques
Arrowsmith
LSI
Association Rules
Semantic type filtering
Time Slicing proposed
MeSH Descriptors
UMLS Concept co-occurrence model
Broad and Similar term Pruning
Statistical and Probabilistic Rankings
HAL
SemRep Predicates
Rarity Principle
Time Slicing Formalized
PSI
Discovery by analogy
Discovery Browsing
Explanatory graph output
1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016

# Machine Learning

gives "computers the ability to learn without being explicitly programmed."

– Arthur Samuel 1959

# Types of Machine Learning

- Supervised machine learning
  - Algorithm: "learns" from preannotated data
  - Examples: Naïve Bayes, SVMs, Neural Networks
- Unsupervised machine learning
  - Algorithm: only relies on the distributional characteristics of the data set to make predicitions
  - Examples: k-nearest neighbor, kmeans clustering

# Our focus

- Supervised machine learning
  - The computer “learns” patterns from a given set of examples
  - The computer infers a function from labeled training data

# ML & NLP

Basically this is a method
to solve a classification problem

In NLP: we can cast many problems as
classification problems

# Examples

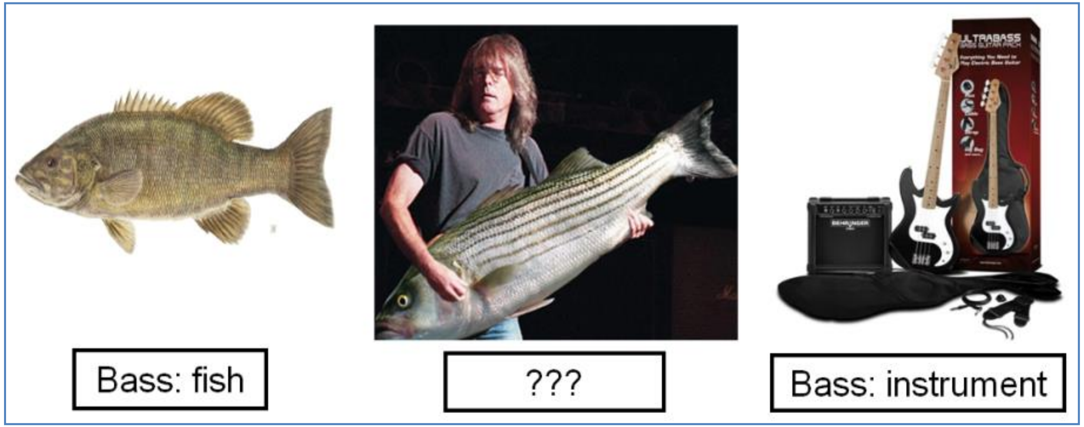
- Document Classification
- Word Sense Disambiguation
- Entity Recognition
- Relationship Extraction
- Sentiment Analysis
- POS Tagging
- Parsing
- etc ….

# Training Data

The algorithm learns from a set of training data

Data that consists of a set of instances that have been tagged with their appropriate label

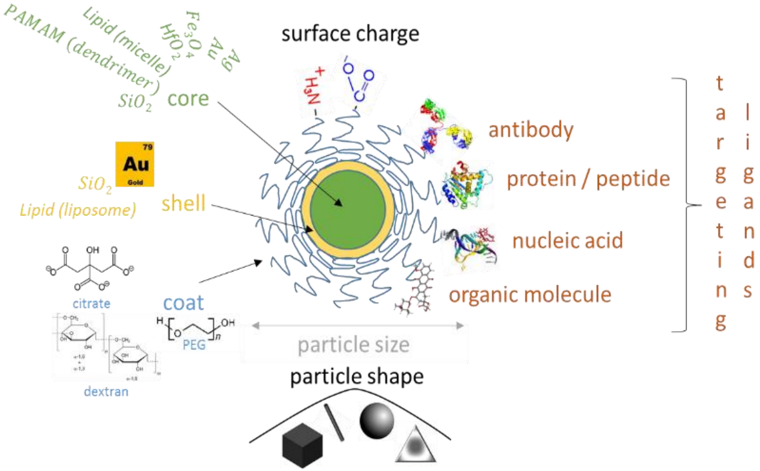
# Training Data: WSD




I went <tw sense="fish">bass</tw> fishing

I played the <tw sense="instrument">bass</tw>

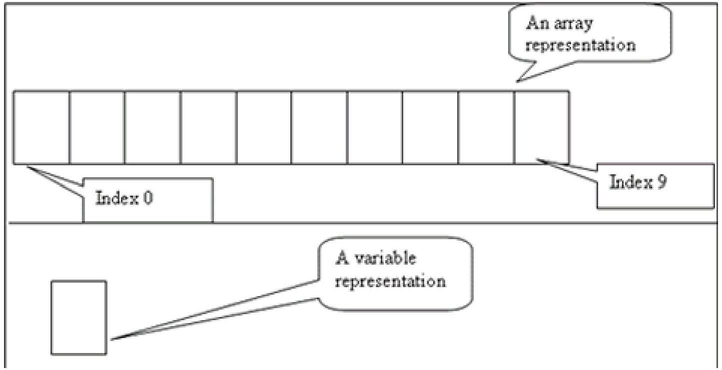
# Training Data: Entity Recognition




The macromolecular complex is <START:surfacecharge> negatively charged <END> at alkaline pH and is present in solution with sodium cations.

# Representing an instance

Feature vectors: an n-dimensional vector of numerical features that represent some object



# Types of features

- Feature-based Representations
  - Morphological
    - Suffix/prefix of the word
  - Lexical
    - Word itself
    - Contextual information
  - Syntatic
    - Part of speech information
  - Semantic
    - Mapped Concepts from a knowledge source
    - Semantic type/group/category
  - Domain specific
    - Is the term a chemical?

- Featureless Representations
  - Word Embeddings
    - Skip Gram
    - CBOW
    - BERT
    - eLMO
  - Character Embeddings

# Feature-based Example

I went <tw sense="fish">bass</tw> fishing

observations

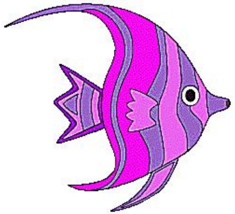
label

data instance

$$\begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} & f_{1,4} & f_{1,5} \\ f_{2,1} & f_{2,2} & f_{2,3} & f_{2,4} & f_{2,5} \\ f_{3,1} & f_{3,2} & f_{3,3} & f_{3,4} & f_{3,5} \\ f_{4,1} & f_{4,2} & f_{4,3} & f_{4,4} & f_{4,5} \end{bmatrix} \rightarrow \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

fishing played bass NOUN ADJECTIVE

[1 0 1 1 0]



# Featureless Example

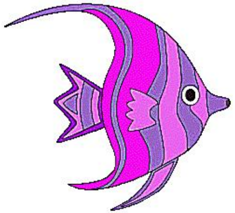
I went <tw sense=“fish”>bass</tw> fishing

observation
s

label

data
instance

$$\begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} & f_{1,4} & f_{1,5} \\ f_{2,1} & f_{2,2} & f_{2,3} & f_{2,4} & f_{2,5} \\ f_{3,1} & f_{3,2} & f_{3,3} & f_{3,4} & f_{3,5} \\ f_{4,1} & f_{4,2} & f_{4,3} & f_{4,4} & f_{4,5} \end{bmatrix} \longrightarrow \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$



```
[ 3.11617279e+00 -2.93757856e-01  2.49324501e-01  2.97784281e+00
  7.16809511e-01 -1.10850143e+00 -3.00553751e+00 -1.39176166e+00
  1.65937781e+00  8.07074010e-01  2.81844378e+00 -1.86554122e+00
```

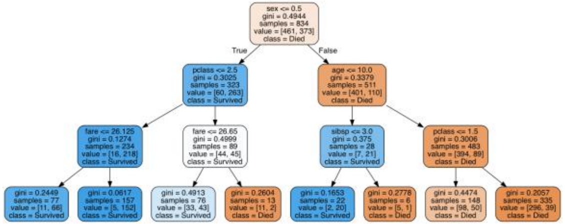
# Basic Idea

$$X \longrightarrow \boxed{\text{Learning Algorithm}} \longrightarrow y$$

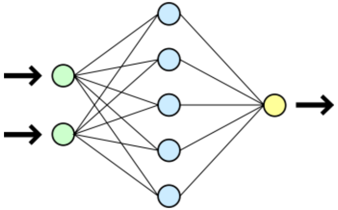
$$\begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} & f_{1,4} & f_{1,5} \\ f_{2,1} & f_{2,2} & f_{2,3} & f_{2,4} & f_{2,5} \\ f_{3,1} & f_{3,2} & f_{3,3} & f_{3,4} & f_{3,5} \\ f_{4,1} & f_{4,2} & f_{4,3} & f_{4,4} & f_{4,5} \end{bmatrix} \begin{matrix} \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \longrightarrow \end{matrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

# Basic Idea

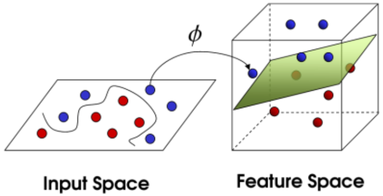
X → Learning Algorithm → y




Decision Trees



Multi-layer
Perceptrons
(Neural Networks)




Support Vector
Machines
(SVM's)

# Types of Supervised Machine Learning Algorithms

- **Discriminative Models**
  - distinguish decision boundaries by inferring knowledge from observed data
  - **Examples:**
    - Neural Networks
    - Decision Trees
    - Support Vector Machines
- **Generative Models**
  - indirect utilizing probability theory
  - **Examples:**
    - Naïve Bayes
    - Hidden Markov Models

# Generative: HMM

- We went over HMMs previously with POS tagging in the Syntactic lecture

# Generative: Naïve Bayes

- Common machine learning algorithm
- Premise
  - choose the best label ($\hat{s}$) given feature vector ($\vec{f}$)

$$\hat{s} = \ argmax_{s \in S} P(s|\vec{f})$$

$$\hat{s} = argmax_{s \in S} P(s|\vec{f})$$

Bayes Rule

Naïve Assumption:

The features are conditionally independent given the word sense

$$\hat{s} = argmax_{s \in S} \frac{P\left(\vec{f}\middle| s\right)P(s)}{P(\vec{f})}$$

Denominator same

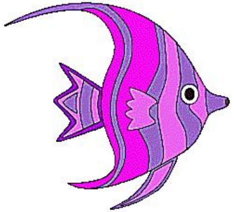
$$\hat{s} = argmax_{s \in S} P\left(\vec{f}\middle| s\right)P(s)$$

$$\hat{s} = argmax_{s \in S} P(s) \prod_{j=1}^{n} P(f_j|s)$$

$$\hat{s} = argmax_{s \in S} P(s) \prod_{j=1}^{n} P(f_j|s)$$

$$P(s_i) = \frac{count(s_i, w_j)}{count(w_j)}$$

How likely is *bass* referring to a *fish* over all the instances of *bass* in your training data



$$\hat{s} = argmax_{s \in S} P(s) \prod_{j=1}^{n} P(f_j|s)$$

$$P(f_j|s) = \frac{count(f_j, s)}{count(s)}$$

So if we have a feature: [$f_j$ = played]

[$f_j$ = played] occurred 3 times for sense $bass^1$

sense $bass^1$ occurred 60 times in the training data

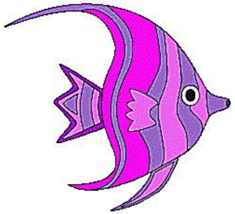
$$P([played]|bass^1) = \frac{count(([played], bass^1)}{count(bass^1)} = \frac{3}{60} = 0.05$$

$$\hat{s} = argmax_{s \in S} P(s) \prod_{j=1}^{n} P(f_j|s) \qquad P(f_j|s) = \frac{count(f_j, s)}{count(s)}$$

*New instance*: She hooked the worm when <tw sense=?>**bass**</tw> fishing

fishing, played, bass, NOUN, ADJECTIVE

[1 0 1 1 0]



$$score^{\,1} = P(bass^1) *$$

$$P([bass]|bass^1) = \frac{count(([bass], bass^1)}{count(bass^1)} *$$

$$P([fishing]|bass^1) = \frac{count(([fishing], bass^1)}{count(bass^1)} *$$

$$P([adjective]|bass^1) = \frac{count(([adjective], bass^1)}{count(bass^1)}$$

$$score^{\,2} = P(bass^2) *$$

$$P([bass]|bass^2) = \frac{count(([bass], bass^2)}{count(bass^2)} *$$

$$P([fishing]|bass^2) = \frac{count(([fishing], bass^2)}{count(bass^2)} *$$

$$P([adjective]|bass^2) = \frac{count(([adjective], bass^2)}{count(bass^2)}$$

The sense with the highest score is assigned to the target word

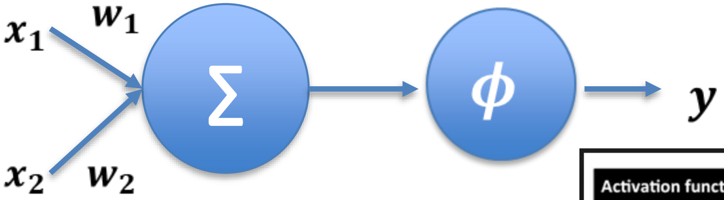
# Generative: Naïve Bayes

The underlying idea is that it is estimating the probability of the label based on seeing that label with the features representing the instance from the training data

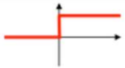
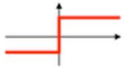
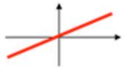
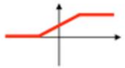
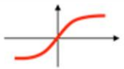
# Discriminative: Neural Network

distinguish decision boundaries by
inferring knowledge from observed data

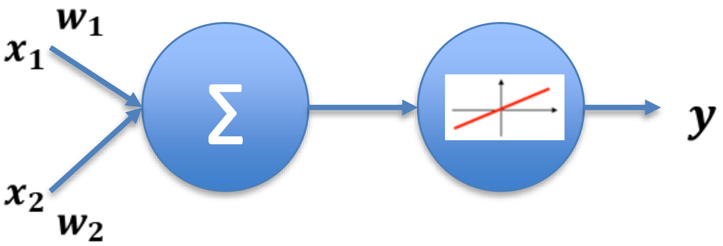
# Perceptron




$$\sum_{i=0}^{n} x_i w_i = x_1 w_1 + x_2 w_2$$

| Activation function | Equation | Example | 1D Graph |
|---|---|---|---|
| Unit step (Heaviside) | $\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$ | Perceptron variant |  |
| Sign (Signum) | $\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$ | Perceptron variant |  |
| Linear | $\phi(z) = z$ | Adaline, linear regression |  |
| Piece-wise linear | $\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$ | Support vector machine |  |
| Logistic (sigmoid) | $\phi(z) = \frac{1}{1 + e^{-z}}$ | Logistic regression, Multi-layer NN |  |
| Hyperbolic tangent | $\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ | Multi-layer NN |  |

# Perceptron



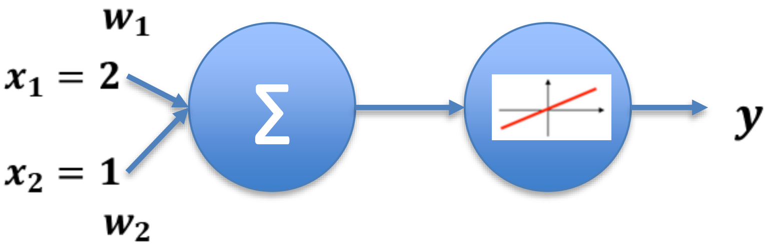

**Training Data**

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| ***label*** | 0 |



$\varepsilon = 0.1$

# Perceptron



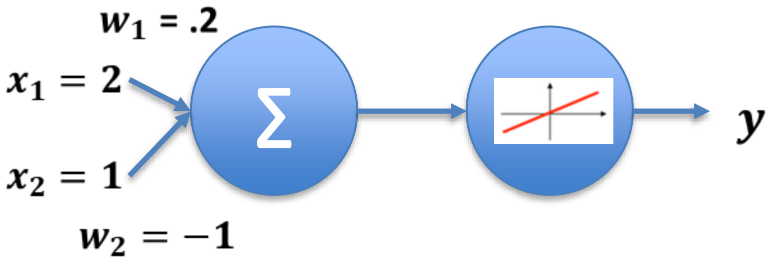

**Training Data**

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| ***label*** | 0 |



$\varepsilon = 0.1$

# Perceptron




**Training Data**

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| *label* | 0 |

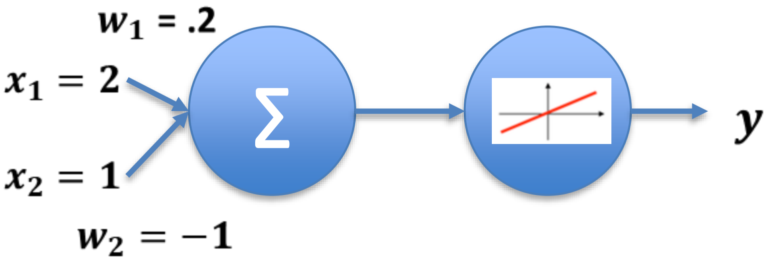
**Random weights**

| | |
|---|---|
| $w_1$ | .2 |
| $w_2$ | -1 |



$\varepsilon = 0.1$

# Perceptron




$$\sum_{i=0}^{n} x_i w_i = 2 * .2 + 1 * -1 = -.6$$

**Training Data**

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| *label* | 0 |

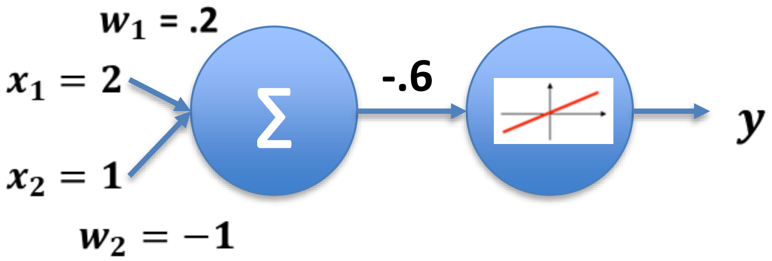
**Random weights**

| | |
|---|---|
| $w_1$ | .2 |
| $w_2$ | -1 |



$\varepsilon = 0.1$

# Perceptron



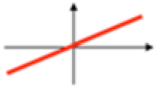

**Training Data**

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| *label* | 0 |

**Random weights**

| | |
|---|---|
| $w_1$ | .2 |
| $w_2$ | -1 |

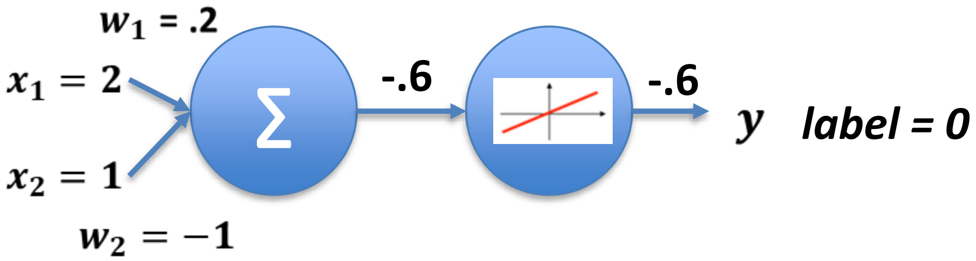
$$\sum_{i=0}^{n} x_i w_i = 2 \ * .2 + 1 \ * -1 = -.6$$

| | | | |
|---|---|---|---|
| Linear | $\phi(z) = z$ | Adaline, linear regression |  |



$\varepsilon = 0.1$

# Perceptron

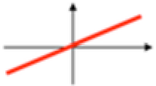



error (E) = label - y

## Training Data

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| *label* | 0 |

$$\sum_{i=0}^{n} x_i w_i = 2 * .2 + 1 * -1 = -.6$$

| | | | |
|---|---|---|---|
| Linear | $\phi(z) = z$ | Adaline, linear regression |  |

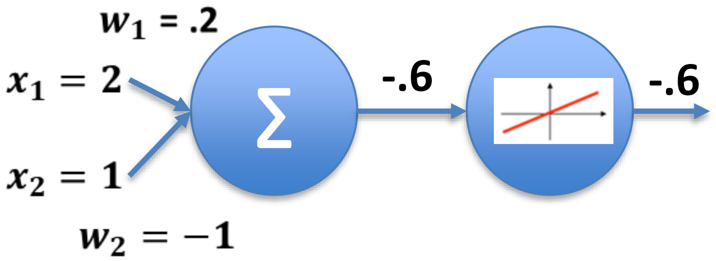
## Random weights

| | |
|---|---|
| $w_1$ | .2 |
| $w_2$ | -1 |



$\varepsilon = 0.1$

# Perceptron



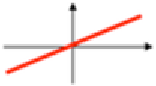

*label = 0*

$Error\ (E)$ = label – y
= 0 – (-.6)
= .6

**Training Data**

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| *label* | 0 |

$$\sum_{i=0}^{n} x_i w_i = 2 * .2 + 1 * -1 = -.6$$

| | | | |
|---|---|---|---|
| Linear | $\phi(z) = z$ | Adaline, linear regression |  |

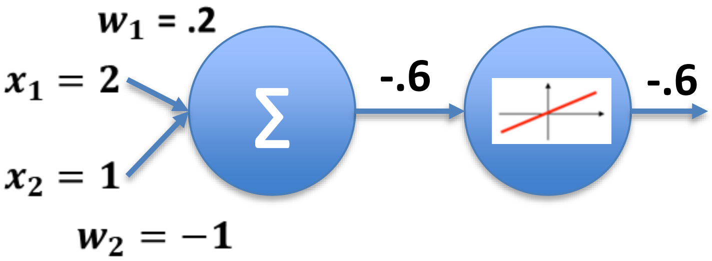
**Random weights**

| | |
|---|---|
| $w_1$ | .2 |
| $w_2$ | -1 |



$\varepsilon = 0.1$

# Perceptron



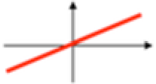

*label = 0*

$Error\ (E)$ = label – y
= 0 – (-.6)
= .6

**Training Data**

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| *label* | 0 |

$$\sum_{i=0}^{n} x_i w_i = 2 * .2 + 1 * -1 = -.6$$

| | | | |
|---|---|---|---|
| Linear | $\phi(z) = z$ | Adaline, linear regression |  |

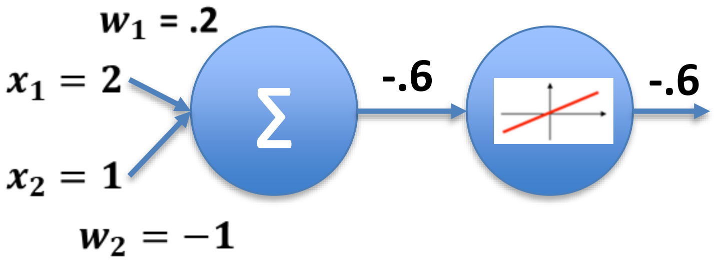
**Random weights**

| | |
|---|---|
| $w_1$ | .2 |
| $w_2$ | -1 |



$\varepsilon = 0.1$

# Perceptron




$y$ *label = 0*

$Error\ (E)$ = label – y
= 0 – (-.6)
= .6

$$\Delta w_i = \frac{\delta E}{\delta w_i}$$

$$w_i = w_i + \Delta w_i$$

**Training Data**

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| *label* | 0 |

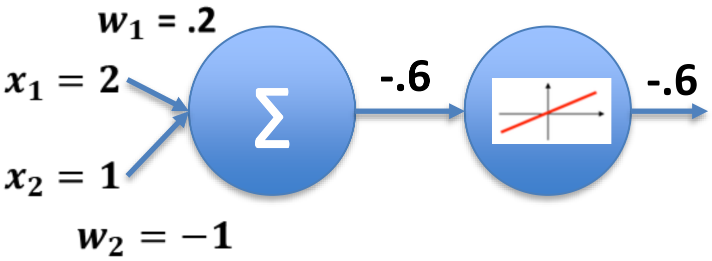
**Random weights**

| | |
|---|---|
| $w_1$ | .2 |
| $w_2$ | -1 |



$\varepsilon = 0.1$

# Perceptron




$y$ ***label = 0***

$Error\ (E)$ = label – y
= 0 – (-.6)
= .6

$$\Delta w_i = \frac{\delta E}{\delta w_i}$$

$$w_i = w_i + \Delta w_i$$

$$w_i = w_i + \epsilon E x_i$$

**Training Data**

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| ***label*** | 0 |

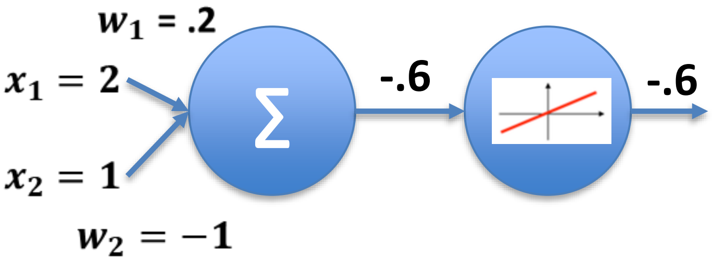
**Random weights**

| | |
|---|---|
| $w_1$ | .2 |
| $w_2$ | -1 |



$\varepsilon = 0.1$

# Perceptron




$y$ *label = 0*

$Error\ (E)$ = label – y
= 0 – (-.6)
= .6

$$\Delta w_i = \frac{\delta E}{\delta w_i}$$

$$w_i = w_i + \Delta w_i$$

$$w_i = w_i + \epsilon E x_i$$

**Training Data**

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| *label* | 0 |

$$w_1 = 0.2 + 0.1 * 0.6 * 0.2 = 0.32$$

$$w_2 = -1 + 0.1 * 0.6 * 1 = -0.40$$

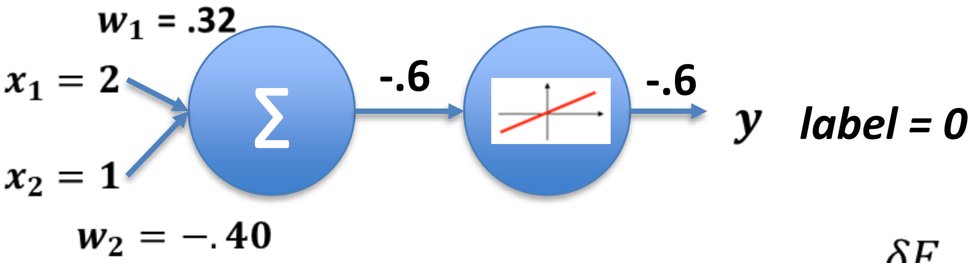
**Random weights**

| | |
|---|---|
| $w_1$ | .2 |
| $w_2$ | -1 |



$\varepsilon = 0.1$

# Perceptron




$Error\ (E)$ = label – y

= 0 – (-.6)

= .6

$$\Delta w_i = \frac{\delta E}{\delta w_i}$$

$$w_i = w_i + \Delta w_i$$

$$w_i = w_i + \epsilon E x_i$$

**Training Data**

| | |
|---|---|
| $x_1$ | 2 |
| $x_2$ | 1 |
| *label* | 0 |

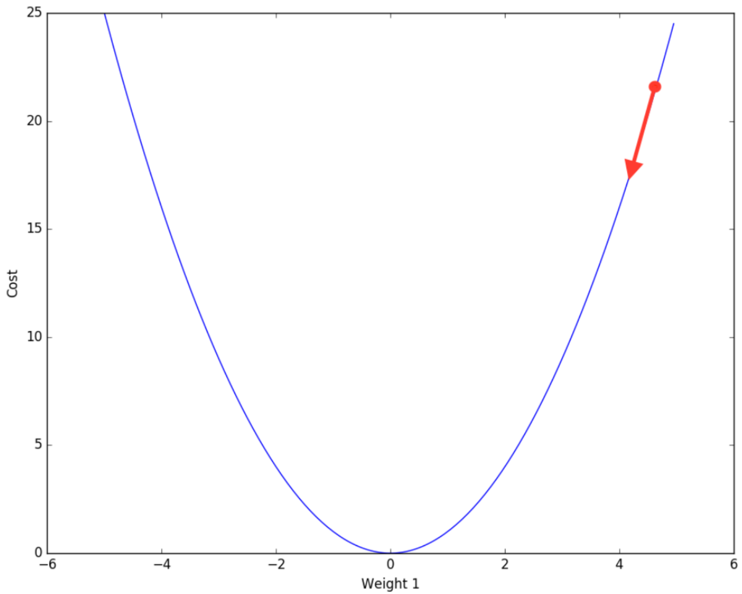
$$w_1 = 0.2 + 0.1 * 0.6 * 0.2 = 0.32$$

$$w_2 = -1 + 0.1 * 0.6 * 1 = -0.40$$



$\varepsilon = 0.1$

# Stochastic Gradient Descent



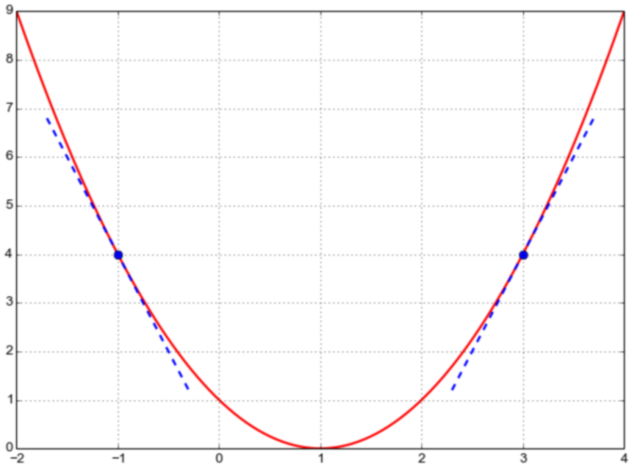

think of the error as a function with respect to the weights

- initializing our weight randomly
- the derivative shows us the slope at this point is a is a positive number
- we want to move closer to the center — so opposite direction of the slope.

$$\Delta w_i = \frac{\delta E}{\delta w_i}$$

# Gradient Descent

Goal: find the minimum of the function.



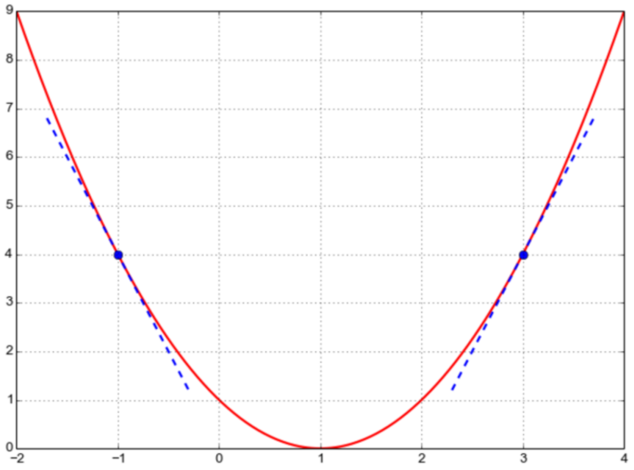
# Gradient Descent

Step 1: Guess an x

Step 2: update x

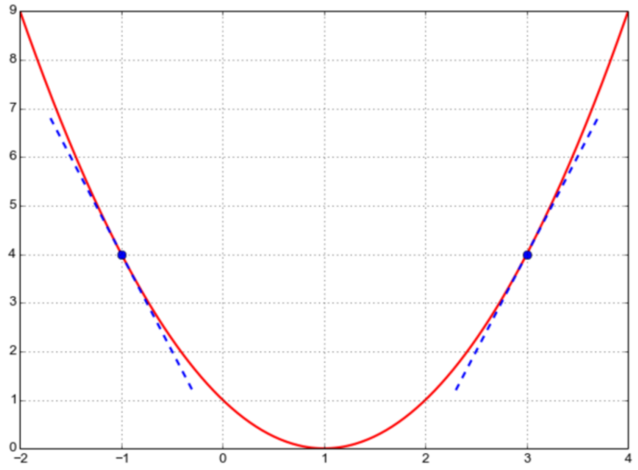
for x, we can go two possible directions:

- increase x
- decrease x



# Gradient Descent

We do this based on the derivative of $f(x)$

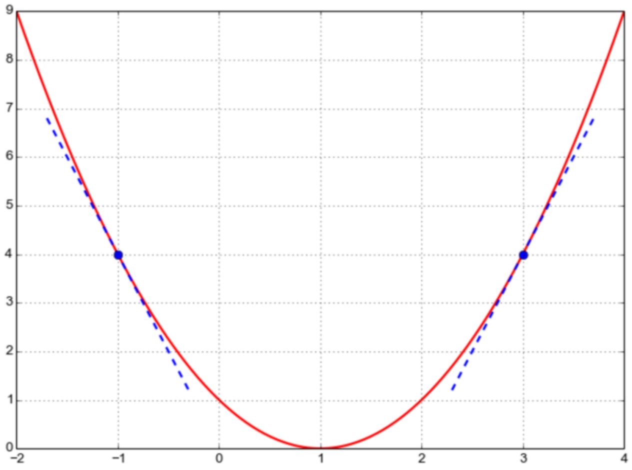



The derivative at some point $x_0$ is defined as:

$$\frac{d}{dx}f(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

# Gradient Descent

This tells us what happens to $f(x)$ when we vary small value to $x$.

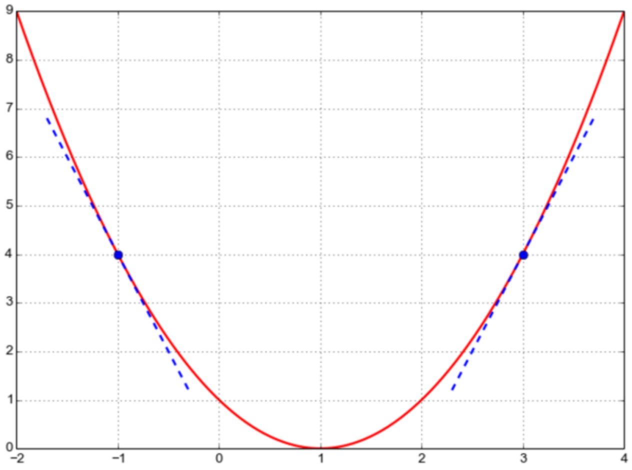



$$\frac{d}{dx} f(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

# Gradient Descent

Example: at $x_0 = 3$

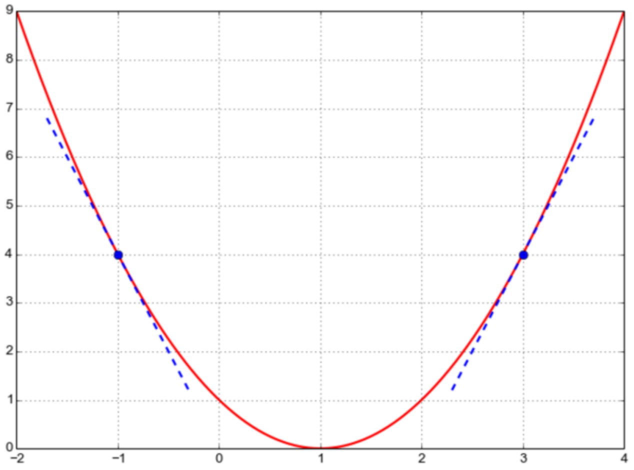
$$\begin{aligned}\frac{d}{dx}f(3) &= \lim_{h\to 0}\frac{f(3+h)-f(3)}{h}\\ &= \lim_{h\to 0}\frac{(3+h-1)^2-(3-1)^2}{h}\\ &= \lim_{h\to 0}\frac{h^2+4h}{h}\\ &= \lim_{h\to 0} h+4 = 4\end{aligned}$$



# Gradient Descent

## Example: at $x_0 = 3$

$$
\begin{aligned}
\frac{d}{dx}f(3) &= \lim_{h\to 0}\frac{f(3+h)-f(3)}{h} \\
&= \lim_{h\to 0}\frac{(3+h-1)^2-(3-1)^2}{h} \\
&= \lim_{h\to 0}\frac{h^2+4h}{h} \\
&= \lim_{h\to 0} h+4 = 4
\end{aligned}
$$

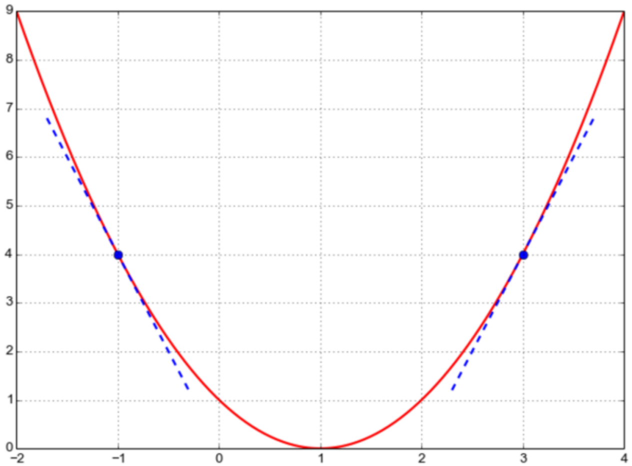



The slope of $\frac{df}{dx}$ at $x_0 = 3$ is 4

# Gradient Descent

Example: at $x_0 = 3$

$$\begin{aligned}\frac{d}{dx}f(3) &= \lim_{h\to 0}\frac{f(3+h)-f(3)}{h}\\ &= \lim_{h\to 0}\frac{(3+h-1)^2-(3-1)^2}{h}\\ &= \lim_{h\to 0}\frac{h^2+4h}{h}\\ &= \lim_{h\to 0} h+4 = 4\end{aligned}$$

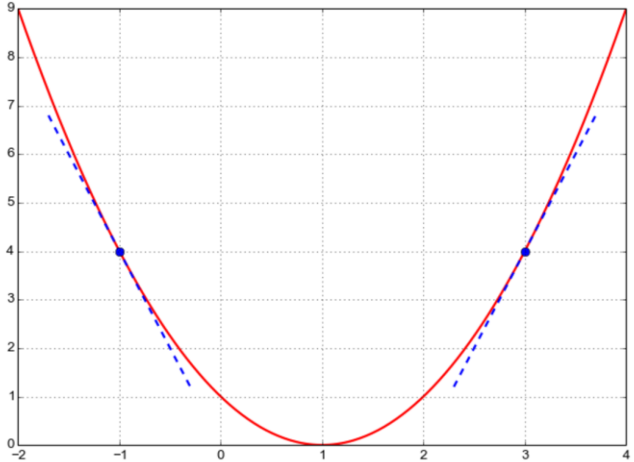



for a small change of $h$ to $x$
the value of $f(x)$ will increase by $4h$

# Gradient Descent

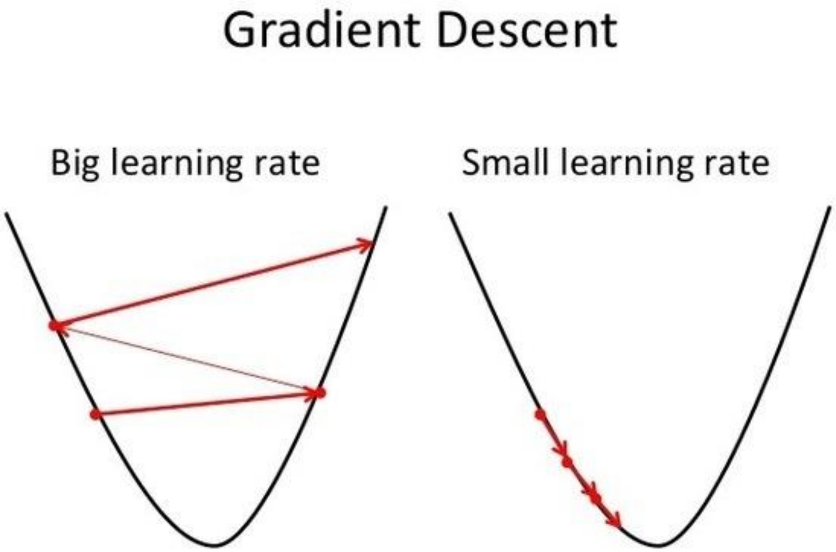
for a small change of $h$ to $x$
the value of $f(x)$ will increase by $4h$

Therefore, to get closer to the minimum of $f(x)$ we should decrease $x_0$ a little bit



$w_i = w_i + \epsilon E x_i$

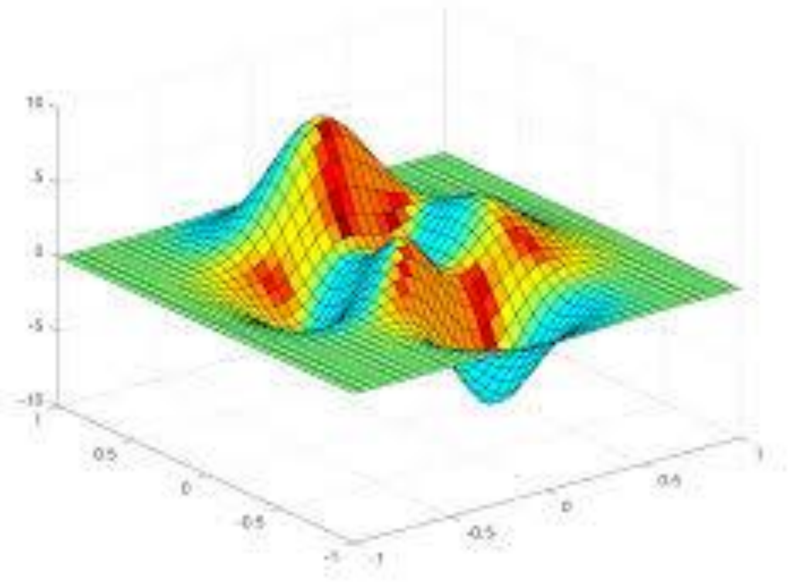
# Learning rate




To big of learning rate and it may take a long time to find the global minimum

To small of learning rate and we may end up in a local minimum thinking it is out global minimum

# Learning rates

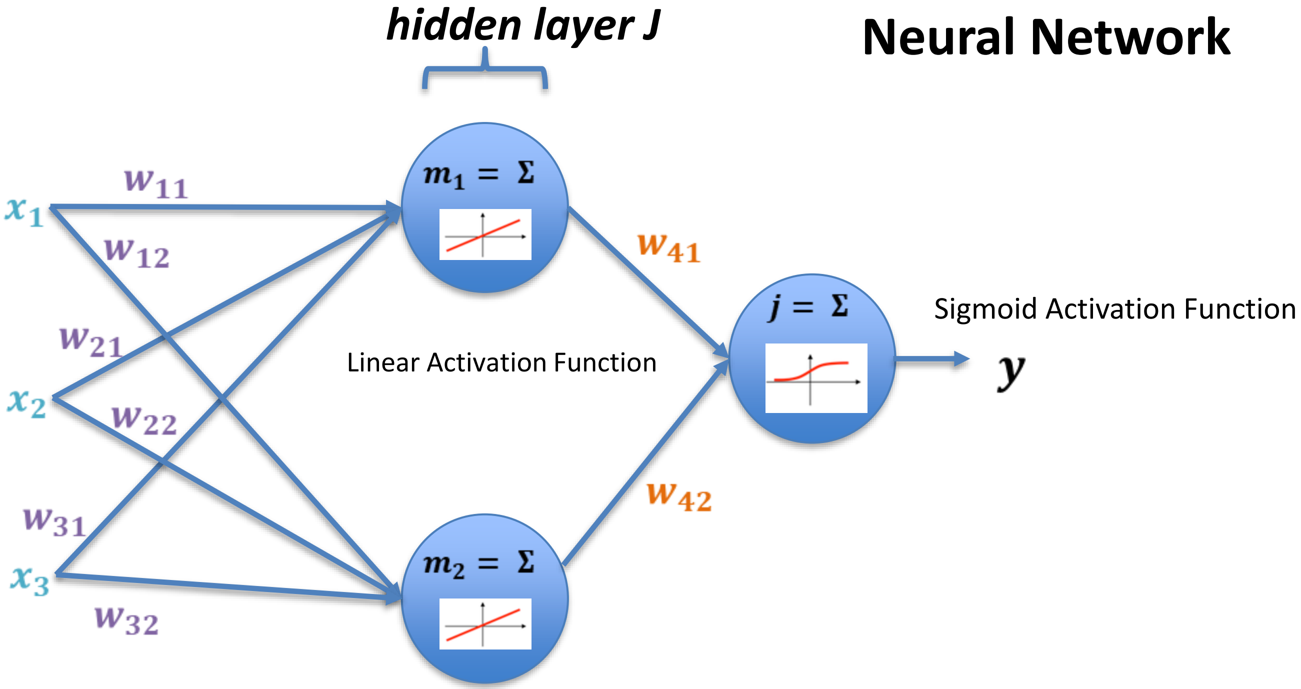


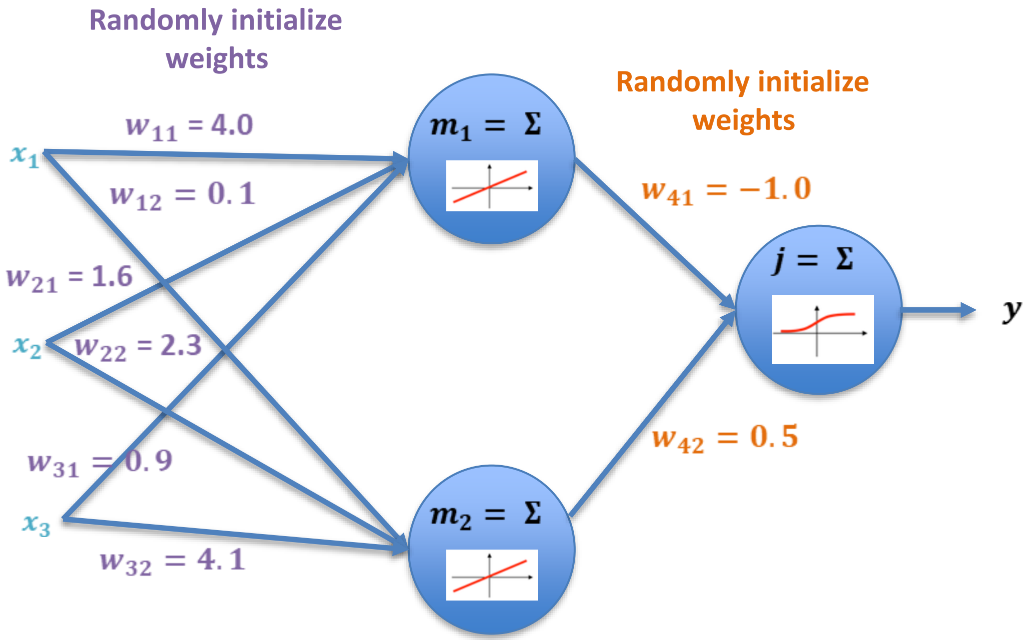
To big of learning rate and it may take a long time to find the global minimum

To small of learning rate and we may end up in a local minimum thinking it is out global minimum

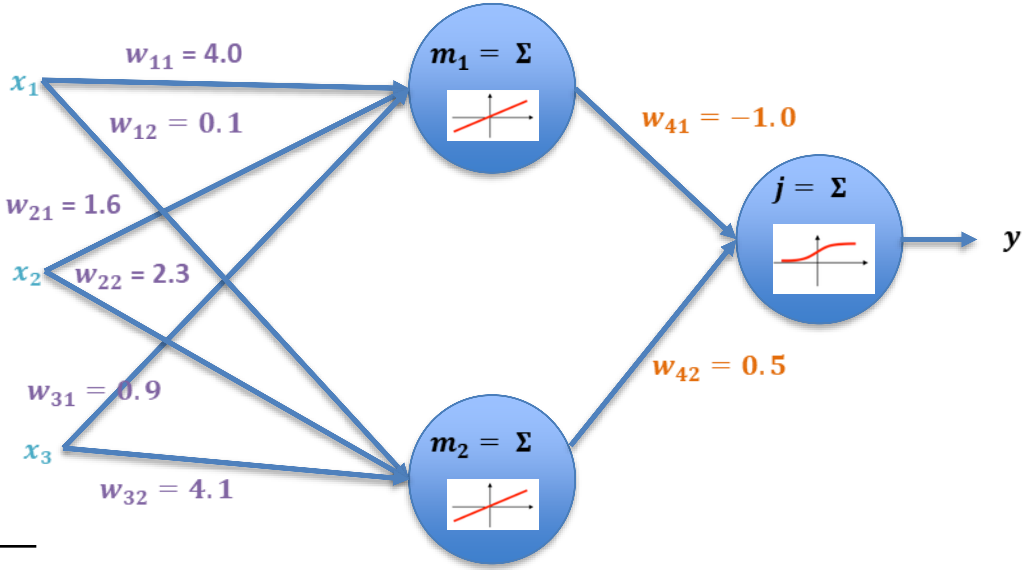
# Does that make sense at a high level?

# Neural Network




Randomly initialize weights
$w_{11} = 4.0$
$x_1$
$w_{12} = 0.1$
$w_{21} = 1.6$
$x_2$
$w_{22} = 2.3$
$w_{31} = 0.9$
$x_3$
$w_{32} = 4.1$
$m_1 = \Sigma$
$m_2 = \Sigma$
Randomly initialize weights
$w_{41} = -1.0$
$w_{42} = 0.5$
$j = \Sigma$
$y$

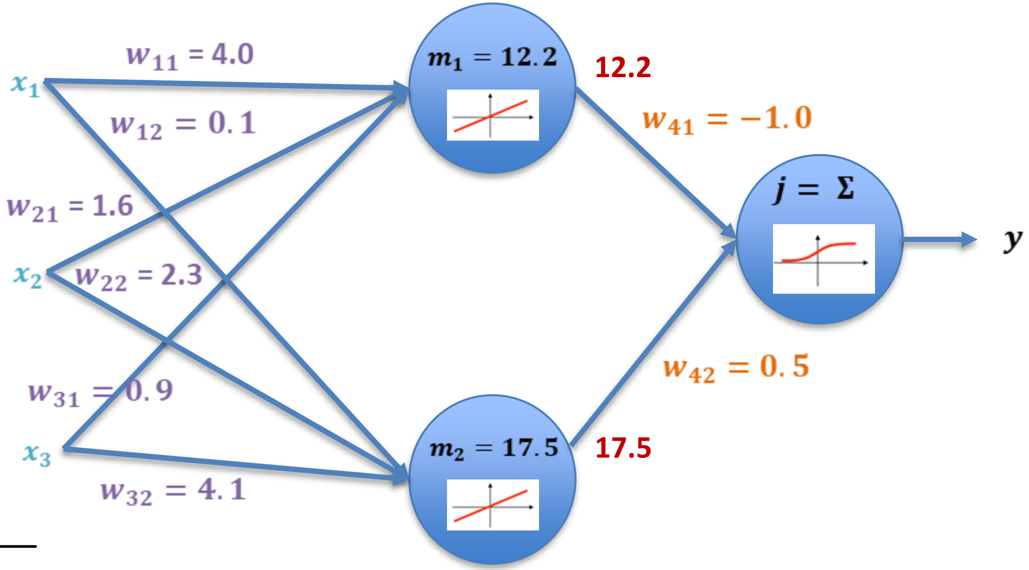
$$\sum_{i=0}^{n} x_i w_i = 1 * 4.0 + 4 * 1.6 + 2 * 0.9 = 12.2$$




| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

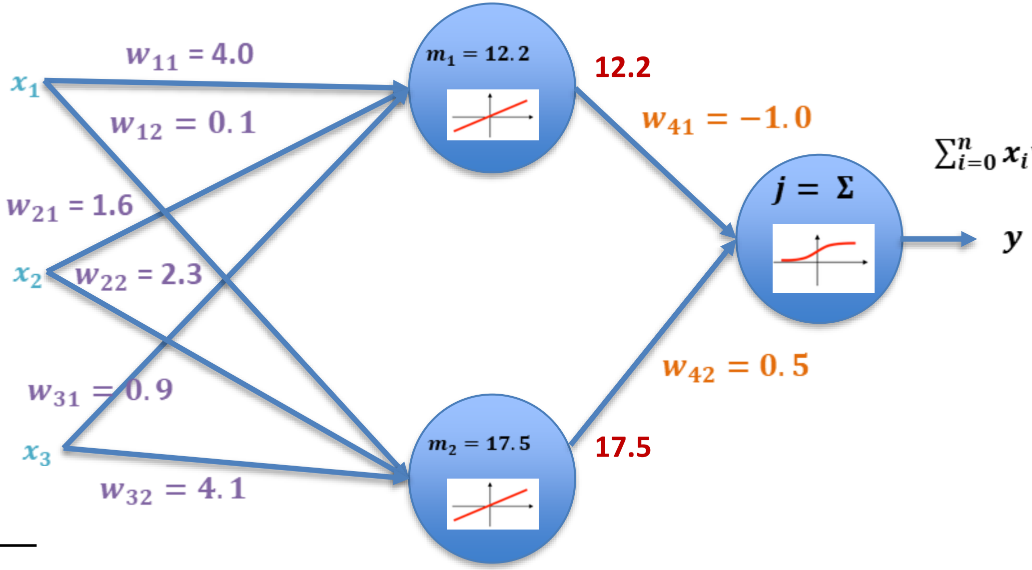
$$\sum_{i=0}^{n} x_i w_i = 1 * 0.1 + 4 * 2.3 + 2 * 4.1 = 17.5$$

$\sum_{i=0}^{n} x_i w_i = 1 * 4.0 + 4 * 1.6 + 2 * 0.9 = 12.2$ $\phi(12.2) = 12.2$




| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$\sum_{i=0}^{n} x_i w_i = 1 * 0.1 + 4 * 2.3 + 2 * 4.1 = 17.5$ $\phi(17.5) = 17.5$

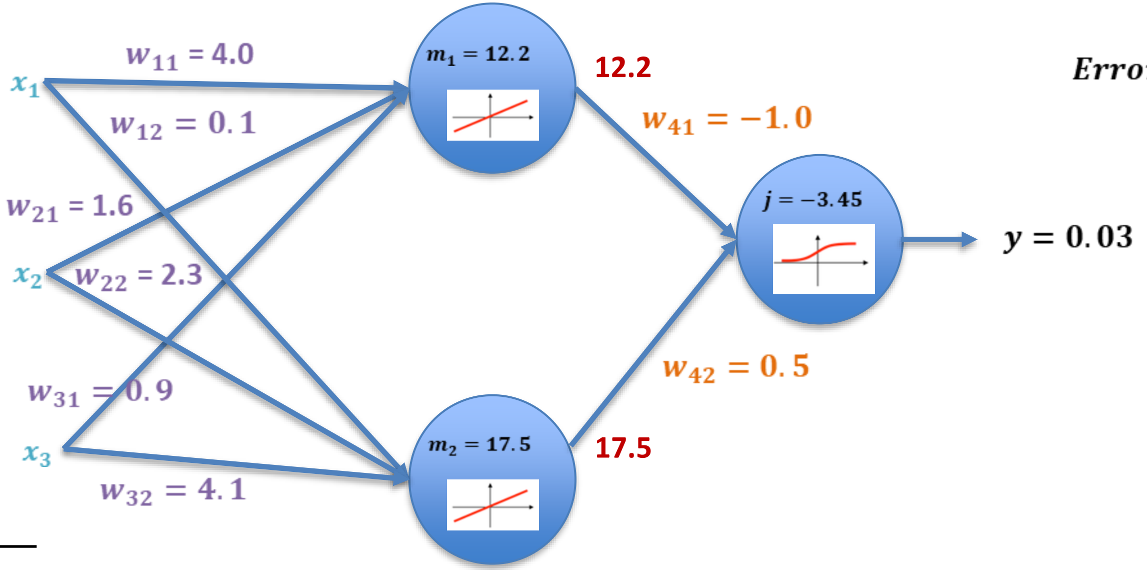



$\sum_{i=0}^{n} x_i w_i = 12.2 * -1.0 + 17.5 * 0.5 = = -3.45$

$\phi(-3.45) = 0.03$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

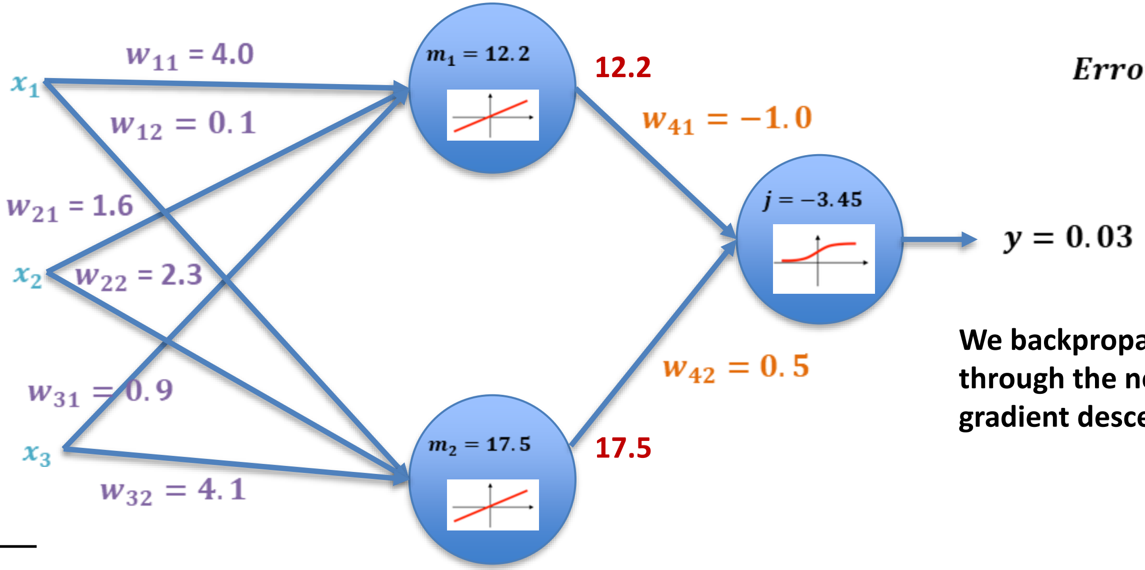
$$\boxed{w_i = w_i + \varepsilon E x_i}$$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\boxed{w_i = w_i + \varepsilon E x_i}$$

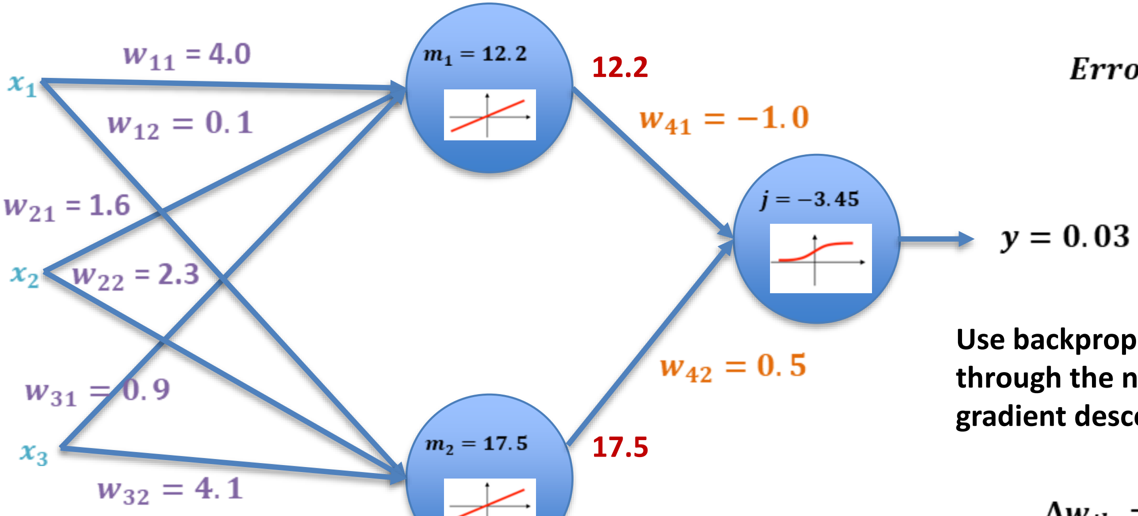



$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

**We backpropagate the error through the network using gradient descent**

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\boxed{w_i = w_i + \varepsilon E x_i}$$



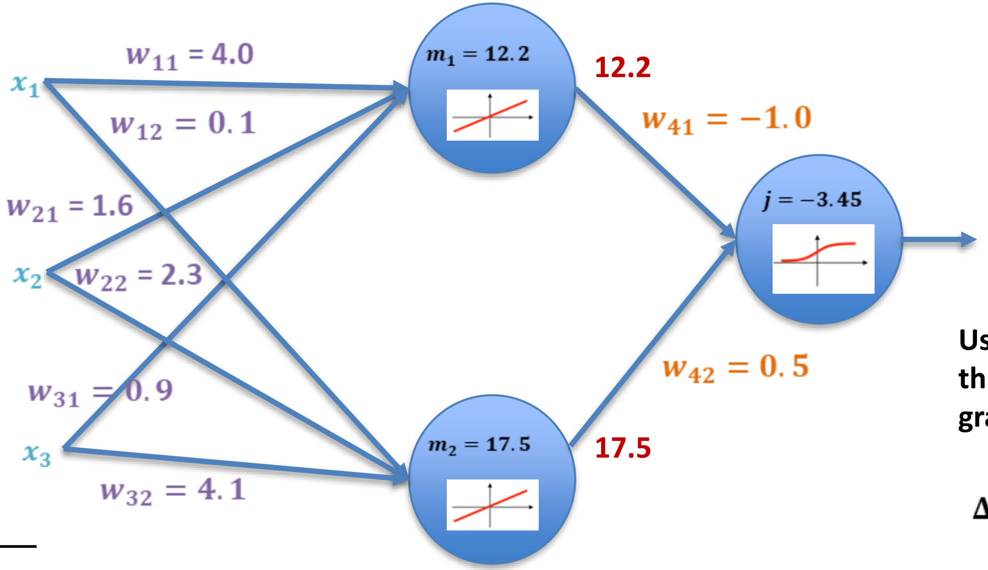

$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

**Use backpropagate the error through the network using gradient descent**

$$\Delta w_{jk} = \frac{\delta E}{\delta w_{jk}}$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\boxed{w_i = w_i + \varepsilon E x_i}$$




$$y = 0.03$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

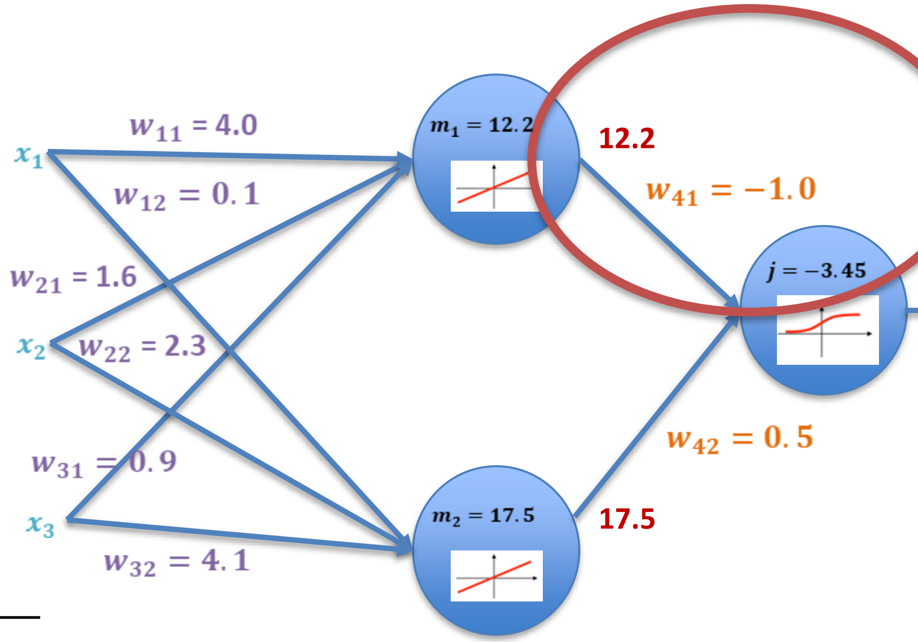
$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

**Use backpropagate the error through the network using gradient descent**

$$\Delta w_{jk} = \frac{\delta E}{\delta w_{jk}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{jk}}$$

**Chain Rule**

$$\boxed{w_i = w_i + \varepsilon E x_i}$$



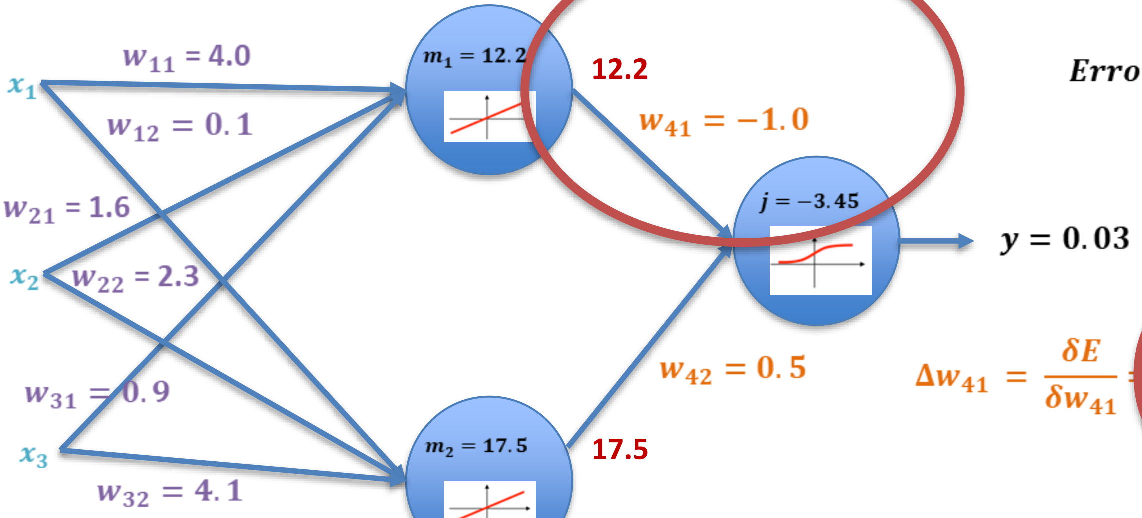

$y = 0.03$

$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{41} = \frac{\delta E}{\delta w_{41}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{41}}$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\boxed{w_i = w_i + \varepsilon E x_i}$$




$Error\ (E)$ = label – y

= 1 – (0.03)

= 0.97

$$\Delta w_{41} = \frac{\delta E}{\delta w_{41}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{41}}$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

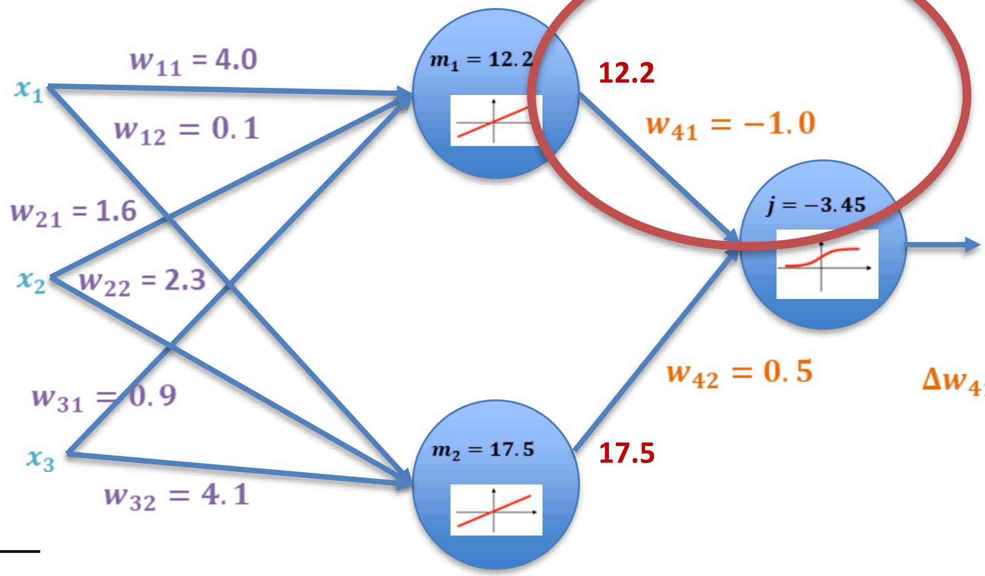
$$\frac{\delta E}{\delta j} = \frac{\delta(\sigma(j))}{\delta j} = \frac{\delta\left(\frac{1}{1+\ e^{-j}}\right)}{\delta j} = \sigma(j) * \left(1 - \sigma(j)\right) * E$$

**derivative of a sigmoid function**

# Derivative of a sigmoid function

$$
\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] \\
&= \frac{d}{dx}\left(1+\mathrm{e}^{-x}\right)^{-1} \\
&= -(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) \\
&= \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) \\
&= \sigma(x) \cdot (1-\sigma(x))
\end{aligned}
$$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$y = 0.03$

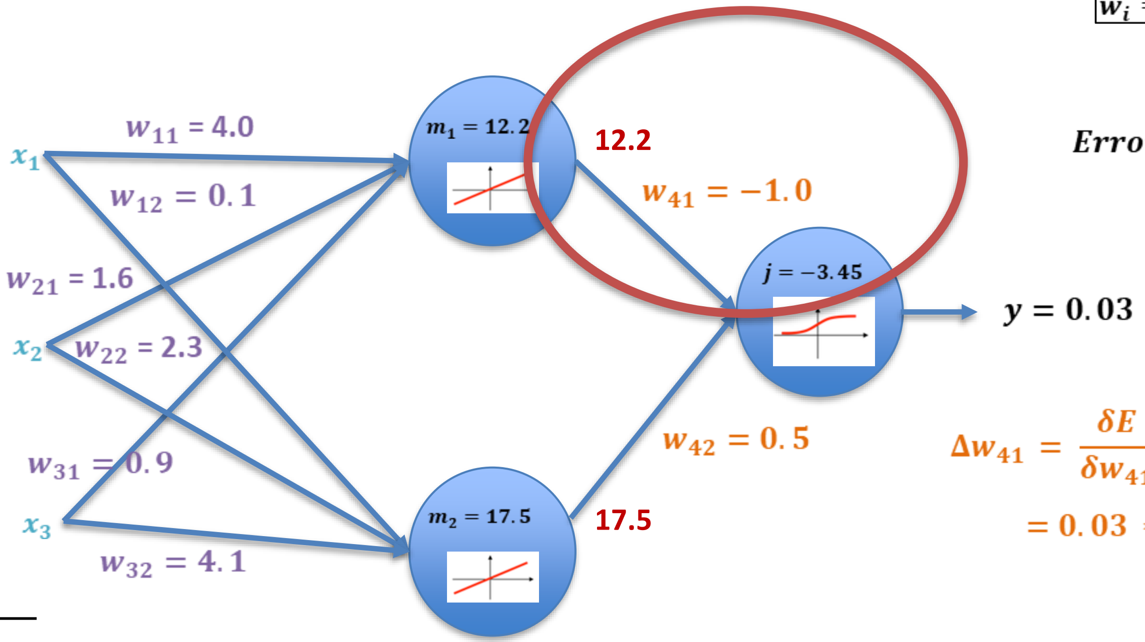
$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{41} = \frac{\delta E}{\delta w_{41}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{41}}$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\frac{\delta E}{\delta j} = \sigma(j) * \left(1 - \sigma(j)\right) * E = \sigma(-3.45) * (1 - \sigma(-3.45) * E$$
$$= 0.03 * (1 - 0.03) * 0.97 = 0.03$$

$$\boxed{w_i = w_i + \varepsilon E x_i}$$



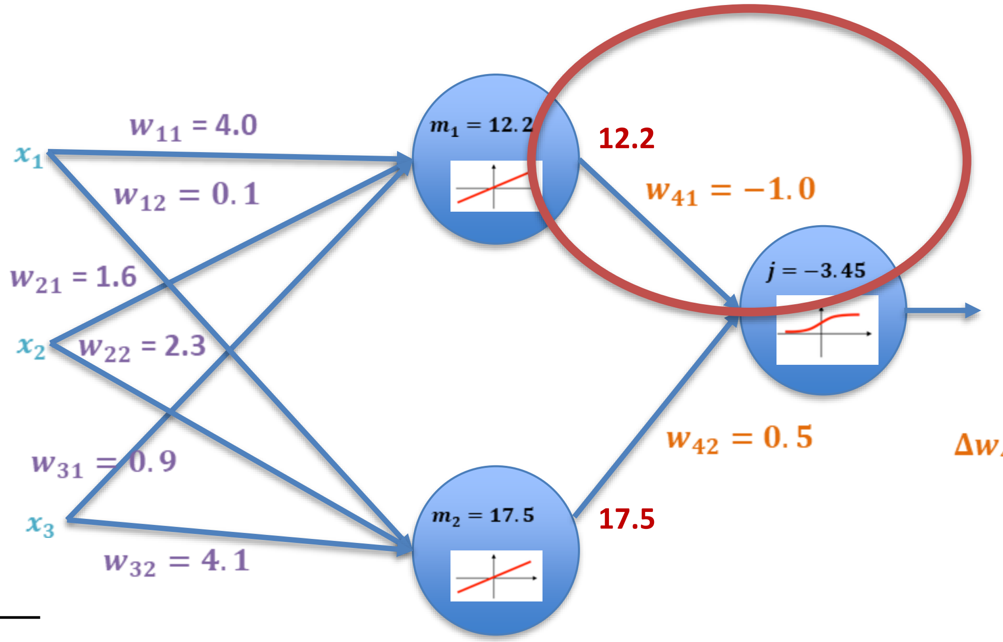

$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{41} = \frac{\delta E}{\delta w_{41}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{41}}$$
$$= 0.03 * \frac{\delta j}{\delta w_{41}}$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\frac{\delta E}{\delta j} = \sigma(j) * \left(1 - \sigma(j)\right) * E = \sigma(-3.45) * (1 - \sigma(-3.45) * E$$
$$= 0.03 * (1 - 0.03) * 0.97 = 0.03$$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$y = 0.03$

$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{41} = \frac{\delta E}{\delta w_{41}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{41}}$$

$$= 0.03 * \frac{\delta j}{\delta w_{41}}$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\frac{\delta j}{\delta w_{41}} = \frac{\delta(m_1 w_{41} + m_2 w_{42})}{\delta w_{41}} = m_1 = 12.2$$

# Derivative of ax+b

**Explanation:**
$f(x) = ax + b$

Take derivative on both sides:

$$\frac{d}{dx}(f(x)) = \frac{d}{dx}(ax + b)$$

Apply the sum/difference rule for derivative which is stated as:

$$\frac{d}{dx}(f + g) = \frac{d}{dx}(f) + \frac{d}{dx}(g)$$

So that we will have:

$$f'(x) = \frac{d}{dx}(ax) + \frac{d}{dx}(b)$$

Remember the derivative of a constant is zero, so that we will have:

$$f'(x) = \frac{d}{dx}(ax) + 0$$

Take the constant out by applying $\frac{d}{dx}(a \cdot f) = a \cdot \frac{d}{dx}(f)$

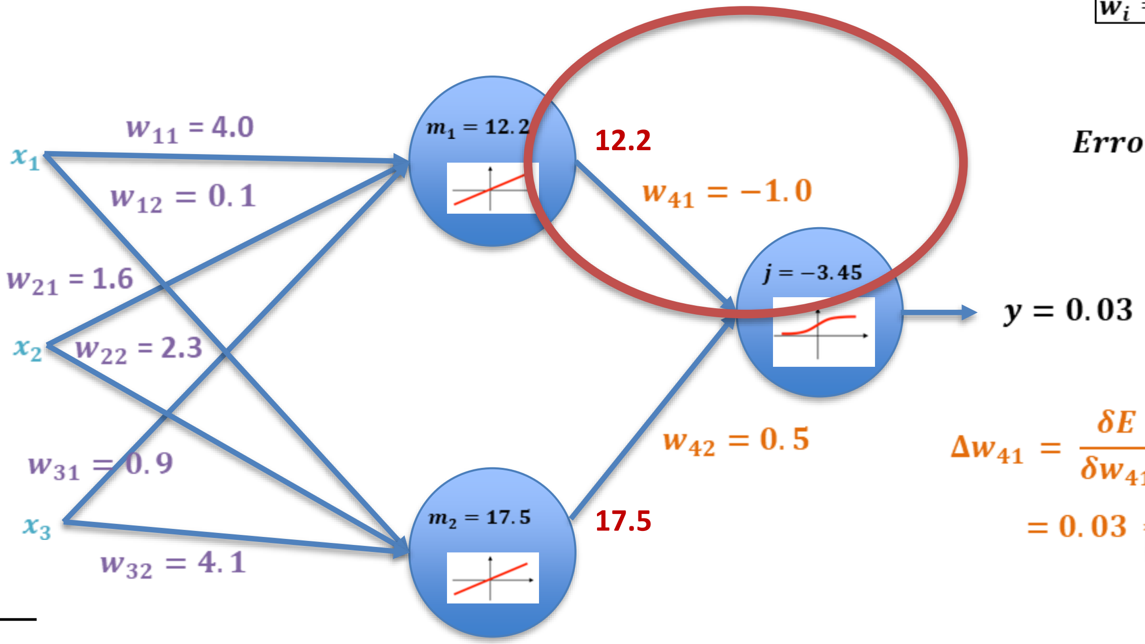
$$f'(x) = a \cdot \frac{d}{dx}(x)$$

Apply the common derivative rule $\frac{d}{dx}(x) = 1$

$$f'(x) = a \cdot 1$$

$$f'(x) = a$$

$$\boxed{w_i = w_i + \varepsilon E x_i}$$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{41} = \frac{\delta E}{\delta w_{41}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{41}}$$

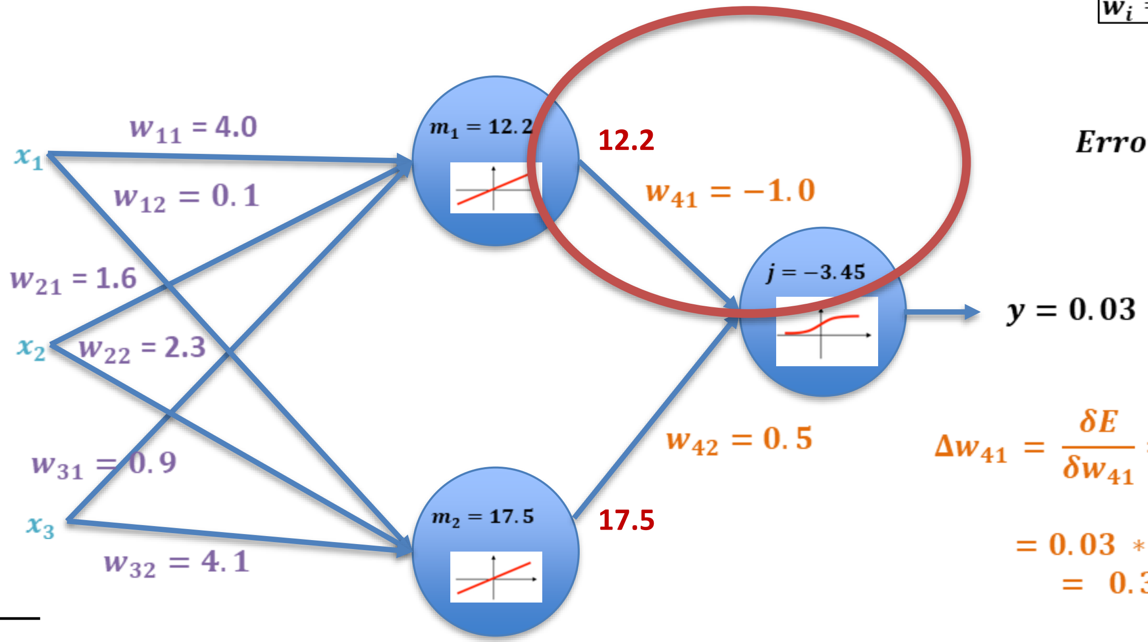
$$= 0.03 * \frac{\delta j}{\delta w_{41}}$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\frac{\delta j}{\delta w_{41}} = \frac{\delta(m_1 w_{41} + m_2 w_{42})}{\delta w_{41}} = m_1 = 12.2$$

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{41} = \frac{\delta E}{\delta w_{41}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{41}}$$

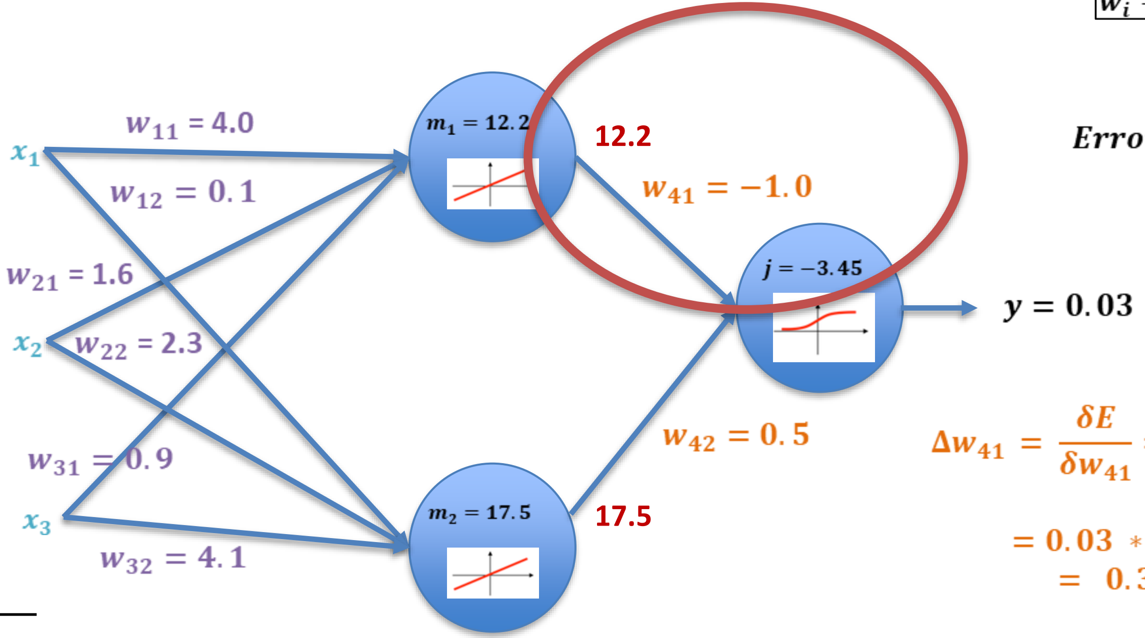
$$= 0.03 * 12.2$$
$$= 0.36$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

VCU College of Engineering

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{41} = \frac{\delta E}{\delta w_{41}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{41}}$$

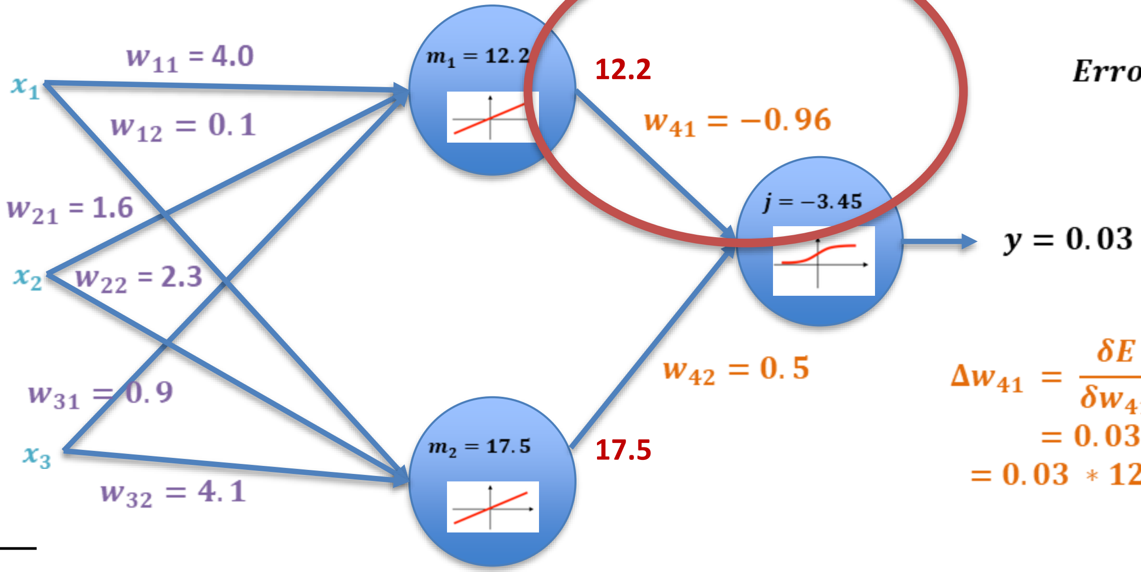
$$= 0.03 * 12.2$$
$$= 0.36$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$w_{41} = w_{41} + \varepsilon \Delta w_{41}$ = -1 + (0.1 * .36) = - .96

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

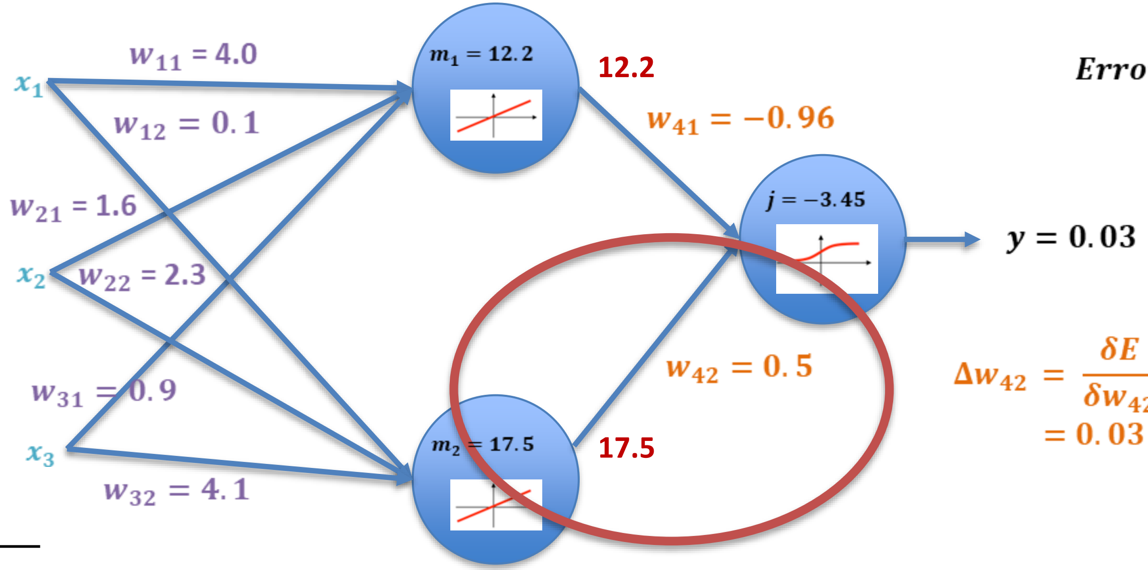
$$\Delta w_{41} = \frac{\delta E}{\delta w_{41}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{41}}$$
$$= 0.03 * m_1$$
$$= 0.03 * 12.2 = 0.36$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$w_{41} = w_{41} + \varepsilon \Delta w_{41}$ = -1 + (0.1 * .36) = - .96

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




*Error* (*E*) = label – y
= 1 – (0.03)
= 0.97

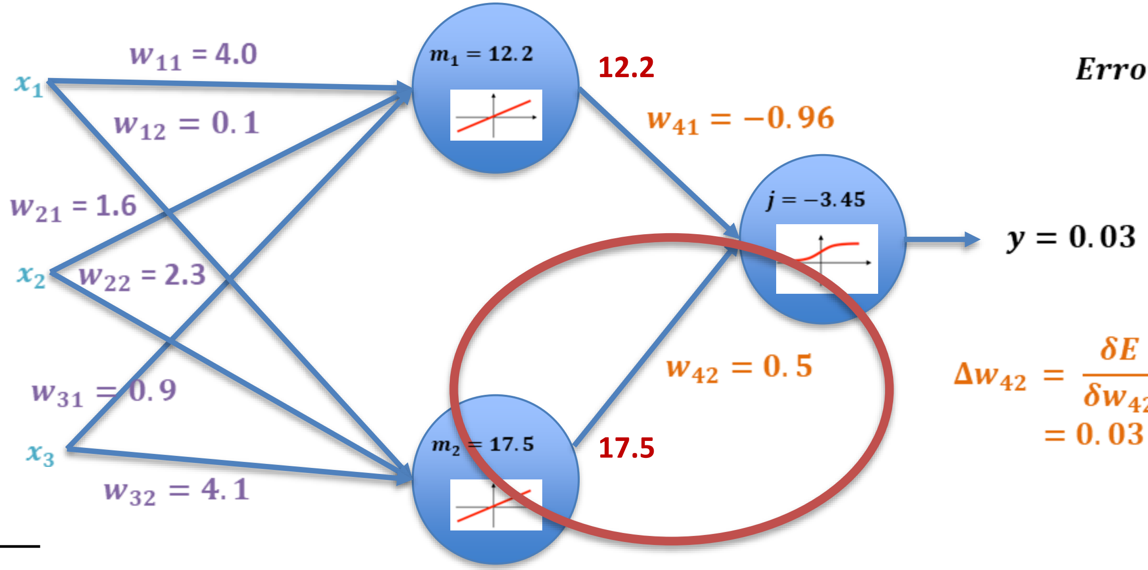
$$\Delta w_{42} = \frac{\delta E}{\delta w_{42}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{42}}$$
$$= 0.03 * m_2$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\frac{\delta E}{\delta j} = \sigma(j) * \left(1 - \sigma(j)\right) * E = \sigma(-3.45) * (1 - \sigma(-3.45) * E$$
$$= 0.03 * (1 - 0.03) * 0.97 = 0.03$$

$\varepsilon = 0.1$

$$\boxed{w_i = w_i + \varepsilon E x_i}$$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

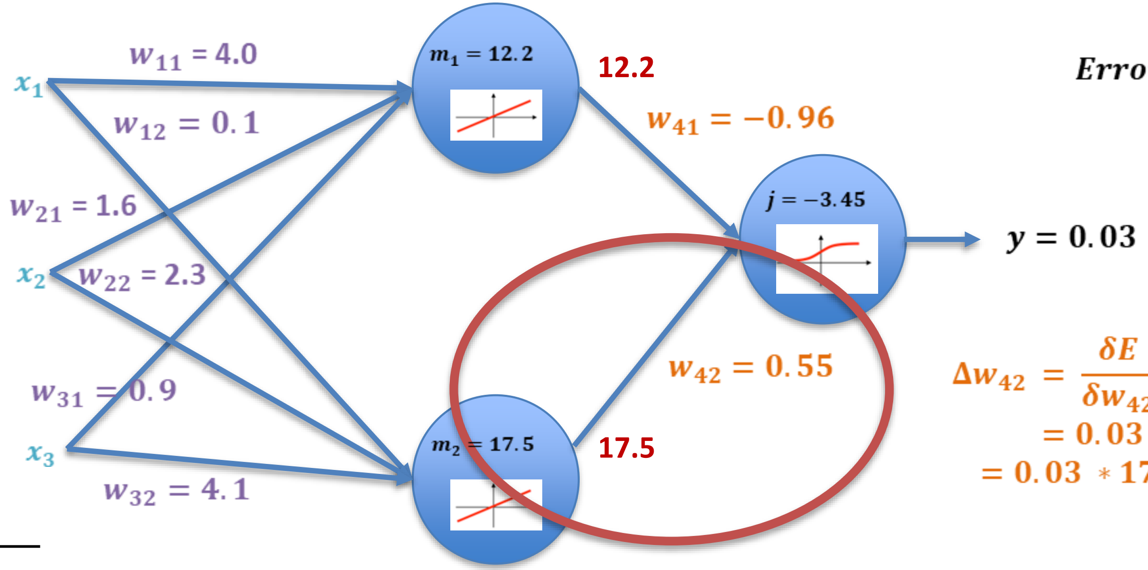
$$\Delta w_{42} = \frac{\delta E}{\delta w_{42}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{42}}$$
$$= 0.03 * m_2$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\frac{\delta j}{\delta w_{42}} = \frac{\delta(m_1 w_{41} + m_2 w_{42})}{\delta w_{42}} = m_2$$

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




*Error* $(E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{42} = \frac{\delta E}{\delta w_{42}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{42}}$$
$$= 0.03 * m_2$$
$$= 0.03 * 17.5 = 0.5$$

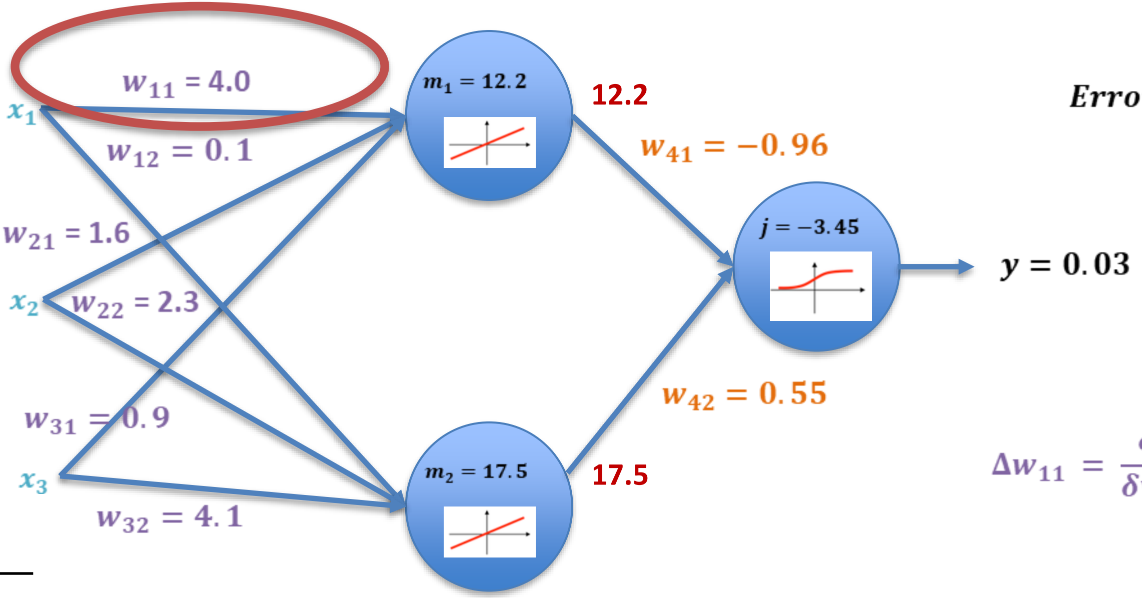
| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\frac{\delta j}{\delta w_{42}} = \frac{\delta(m_1 w_{41} + m_2 w_{42})}{\delta w_{42}} = m_2$$

$w_{42} = w_{42} + \varepsilon \Delta w_{42}$ = 0.5 + (0.1 * .5) = - .55

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$



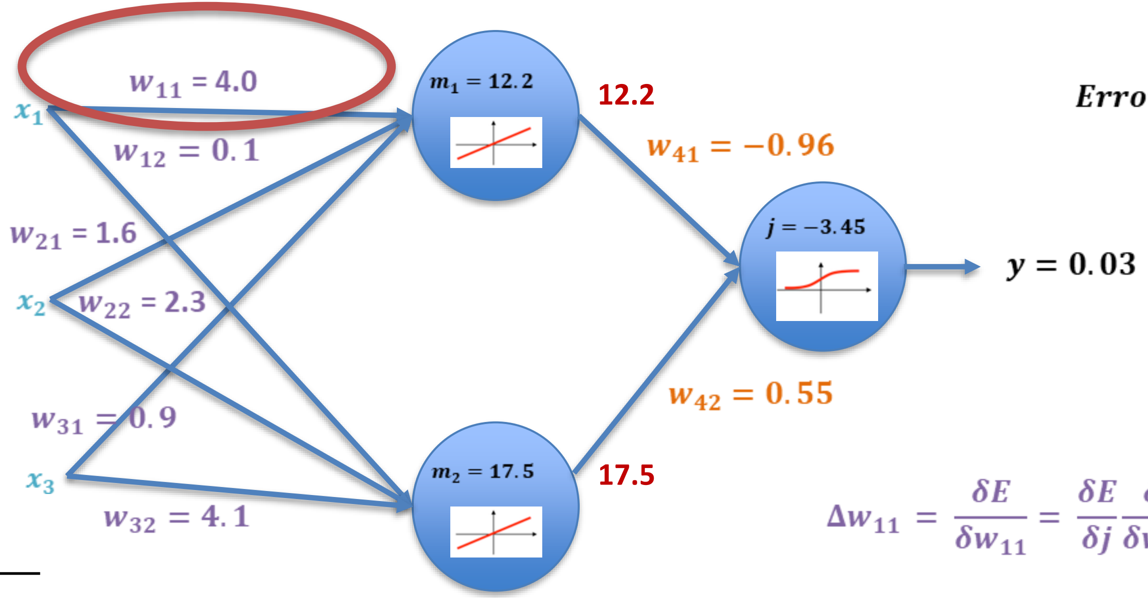

$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}}$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$



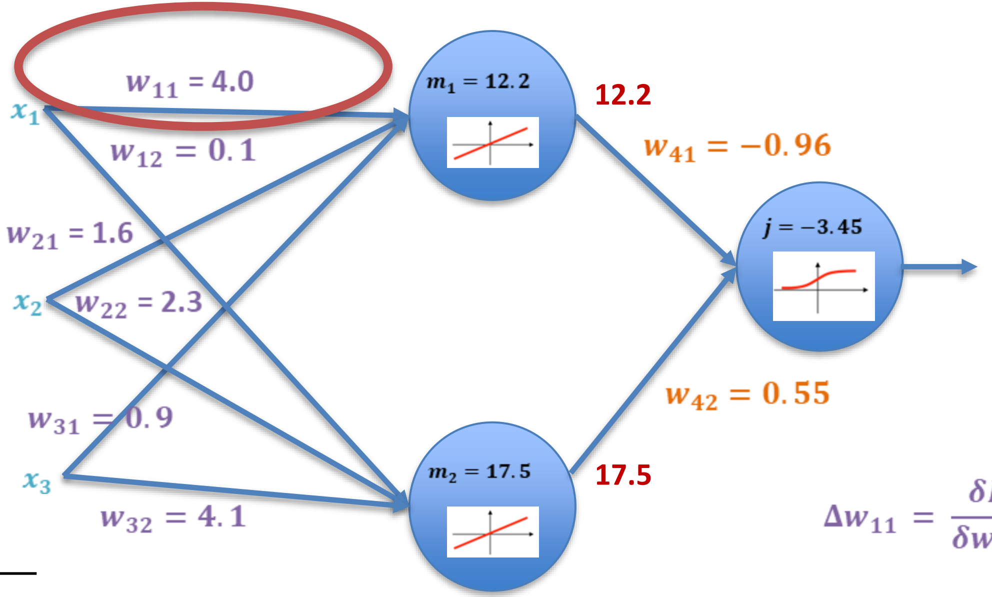

*Error* $(E)$ = label – y
= 1 – (0.03)
= 0.97

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{11}}$$

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$y = 0.03$

$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{11}}$$

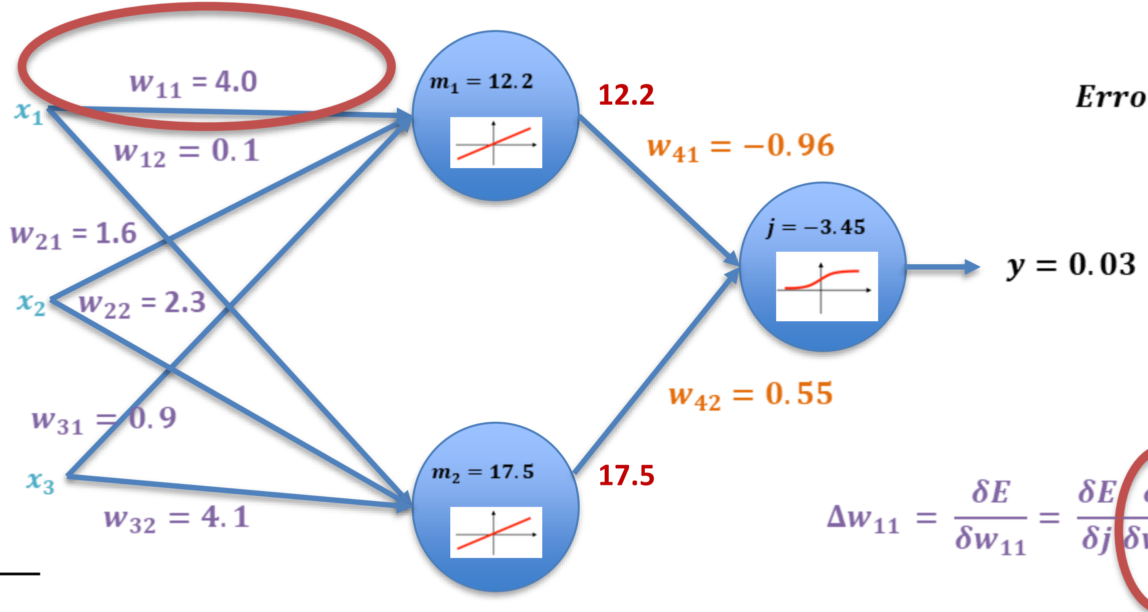
| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\frac{\delta E}{\delta j} = \sigma(j) * \left(1 - \sigma(j)\right) * E = \sigma(-3.45) * (1 - \sigma(-3.45) * E$$
$$= 0.03 * (1 - 0.03) * 0.97 = 0.03$$

**ALREADY CALCULATED THIS PREVIOUSLY**

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$



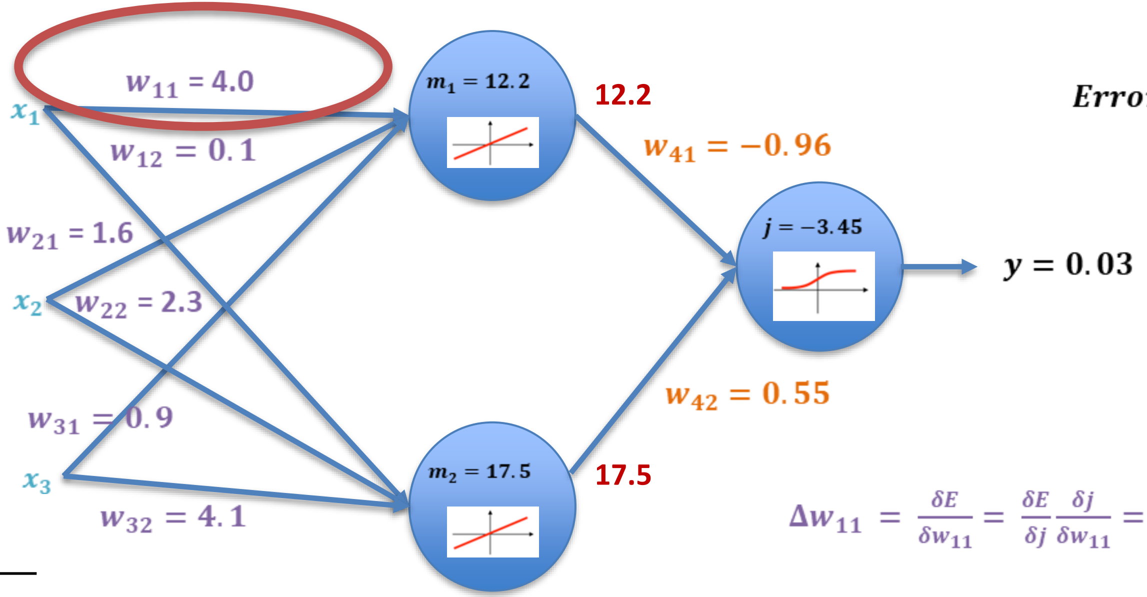

$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{11}}$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




*Error* $(E)$ = label – y
= 1 – (0.03)
= 0.97

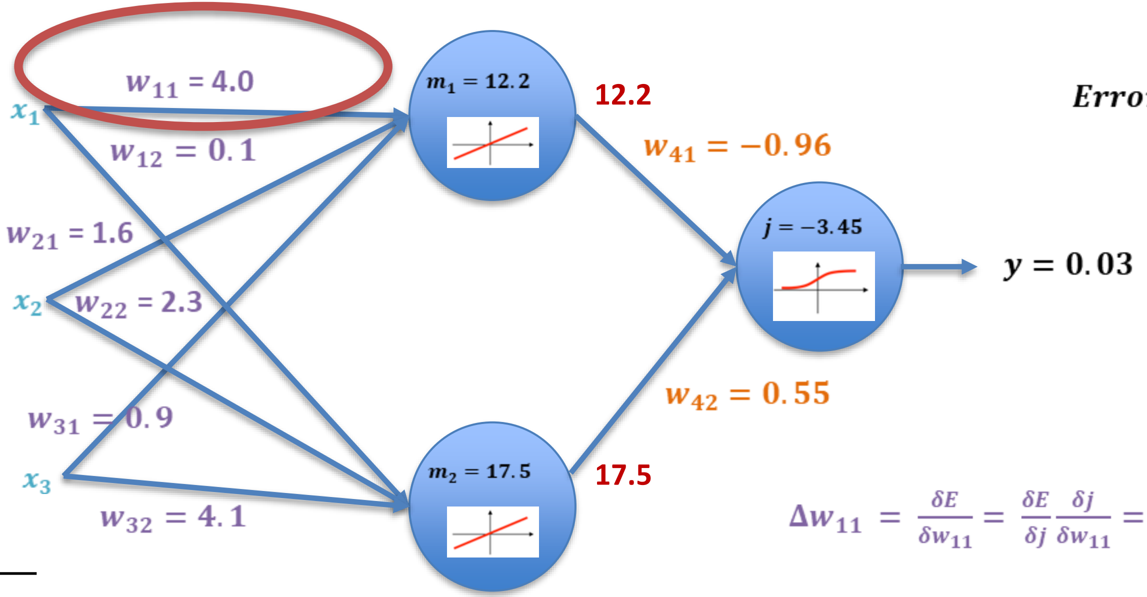
$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta m_1}\frac{\delta m_1}{\delta w_{11}}$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

**Chain Rule again**

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

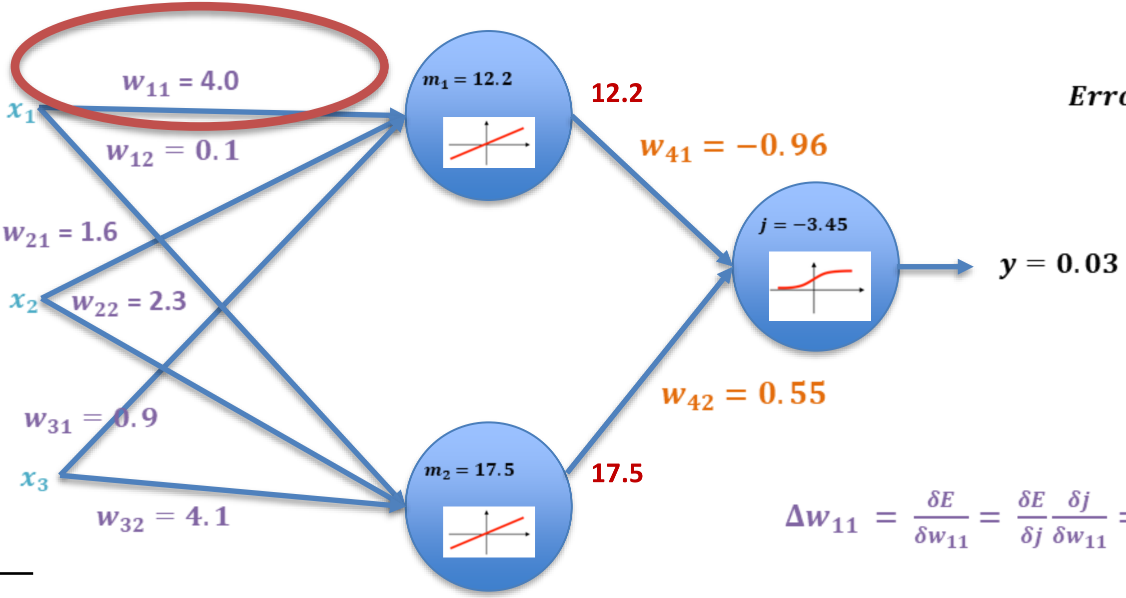
| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta m_1}\frac{\delta m_1}{\delta w_{11}}$$

$$\frac{\delta j}{\delta m_1} = \frac{\delta(m_1 w_{41} + m_2 w_{42})}{\delta m_1} = w_{41} = -0.96$$

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




*Error* $(E)$ = label – y
= 1 – (0.03)
= 0.97

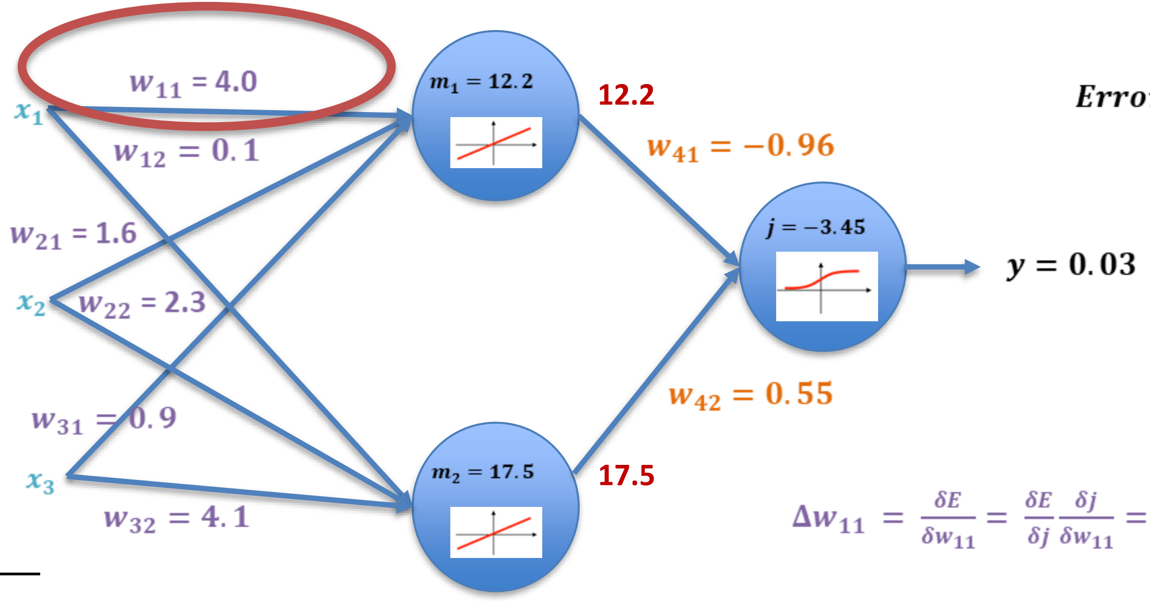
$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta m_1}\frac{\delta m_1}{\delta w_{11}}$$

$$= 0.03 * w_{41} * \frac{\delta m}{\delta w_{11}}$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

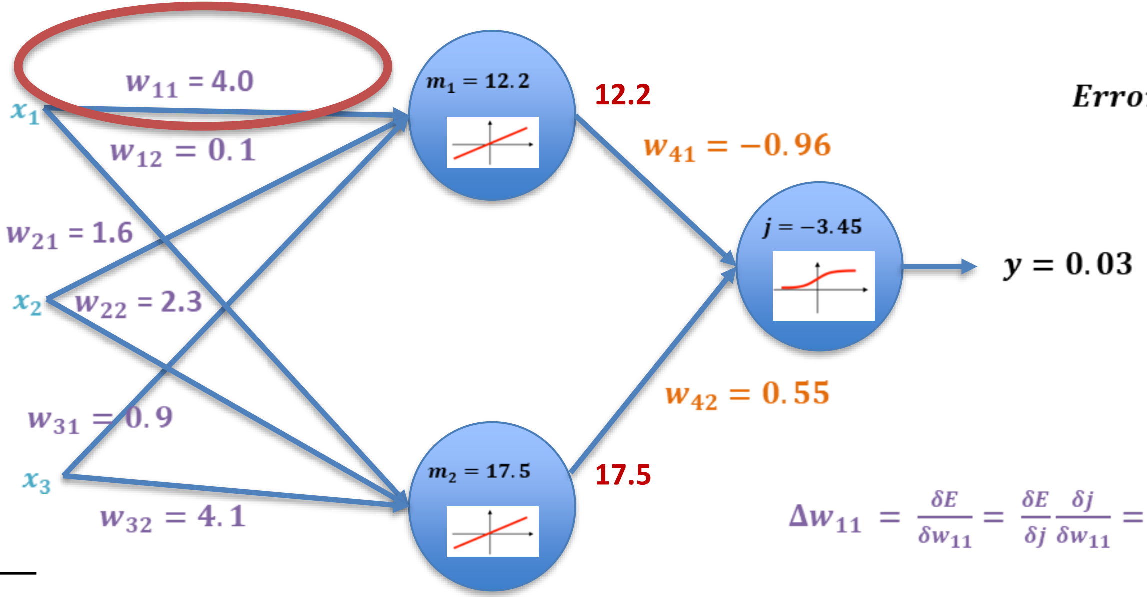
$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta m_1}\frac{\delta m_1}{\delta w_{11}}$$

$$= 0.03 * w_{41} * \frac{\delta m_1}{\delta w_{11}}$$

$$\frac{\delta m_1}{\delta w_{11}} = \frac{\delta(x_1 w_{11} + x_2 w_{21} + x_3 w_{31})}{\delta w_{11}} = x_1 = 1$$

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

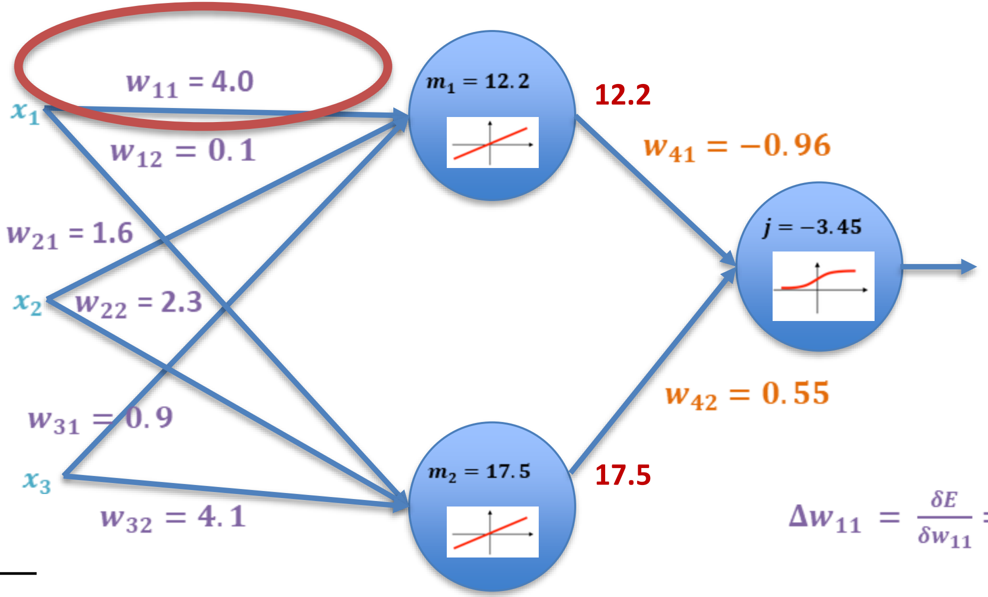
$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta m_1}\frac{\delta m_1}{\delta w_{11}}$$

$$= 0.03 * w_{41} * x_1$$

$$\frac{\delta m_1}{\delta w_{11}} = \frac{\delta(x_1 w_{11} + x_2 w_{21} + x_3 w_{31})}{\delta w_{11}} = x_1 = 1$$

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$y = 0.03$

$Error\ (E)$ = label – y

= 1 – (0.03)

= 0.97

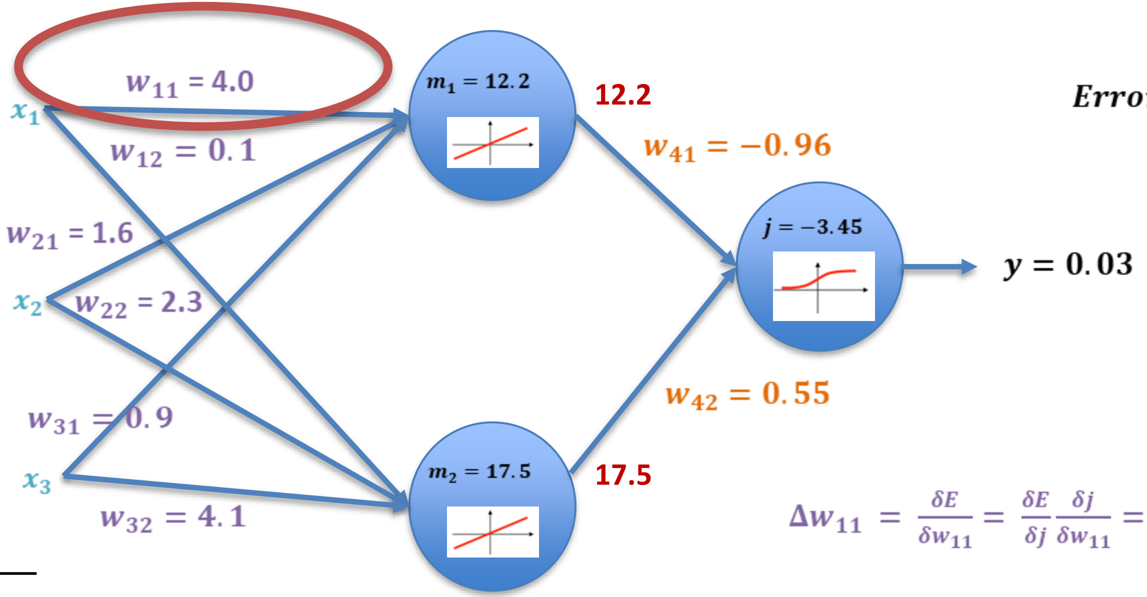
$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta m_1}\frac{\delta m_1}{\delta w_{11}}$$

$$= 0.03 * -1.0 * 1 = -0.03$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| ***label*** | 1 |

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta m_1}\frac{\delta m_1}{\delta w_{11}}$$

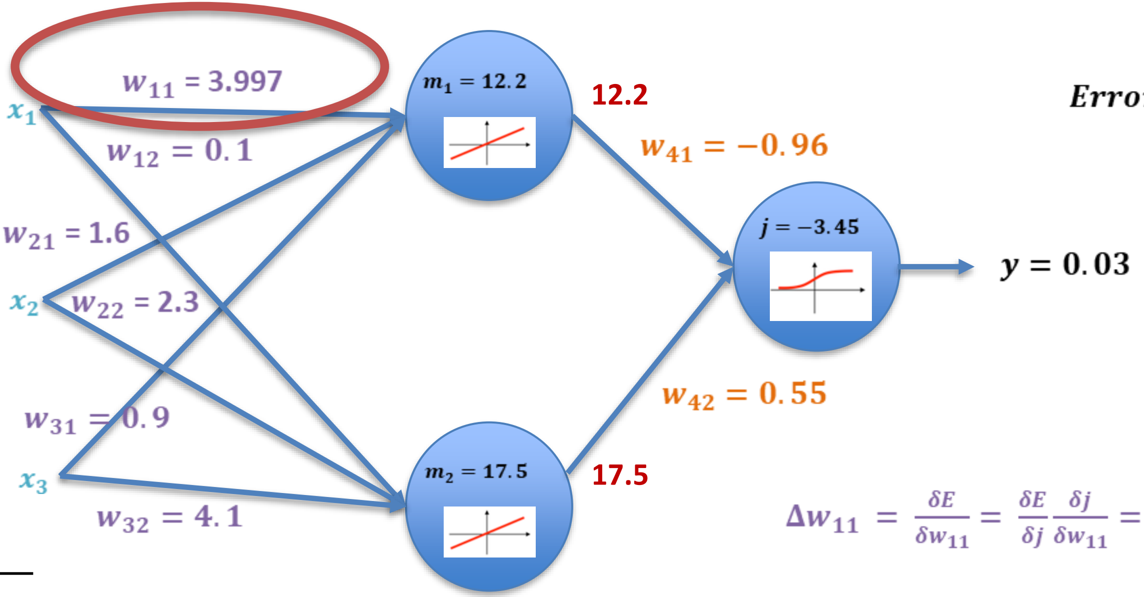
$$= 0.03 * -1.0 * 1 = -0.03$$

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$w_{11} = w_{11} + \varepsilon \Delta w_{11} = 4.0 + (0.1 * -.03) = 3.997$$

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

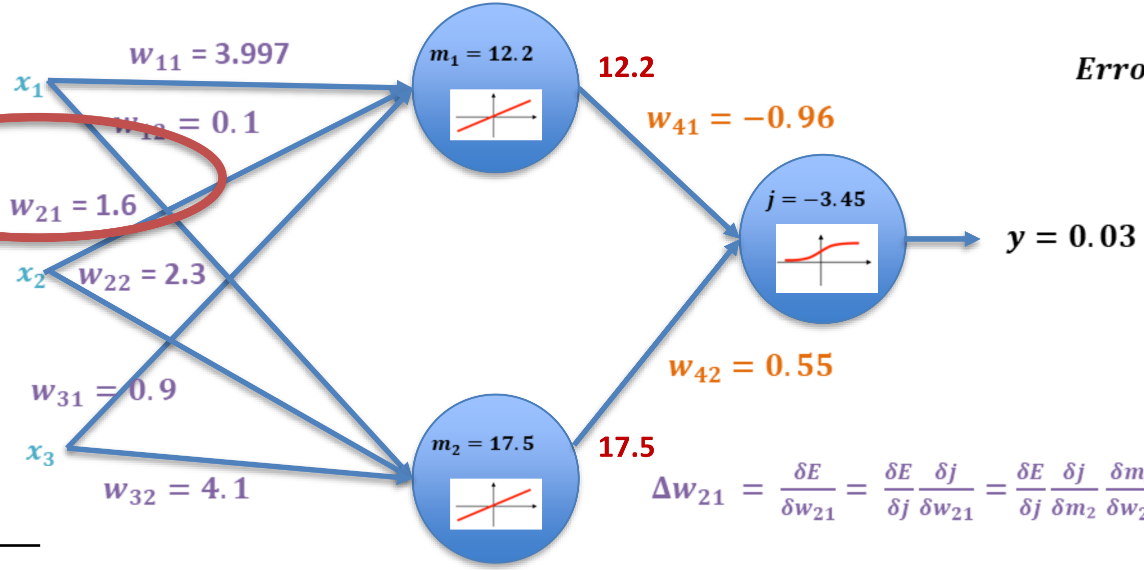
$$\Delta w_{11} = \frac{\delta E}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{11}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta m_1}\frac{\delta m_1}{\delta w_{11}}$$

$$= 0.03 * -1.0 * 1 = -0.03$$

$$w_{11} = w_{11} + \varepsilon \Delta w_{11} = 4.0 + (0.1 * -.03) = 3.997$$

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

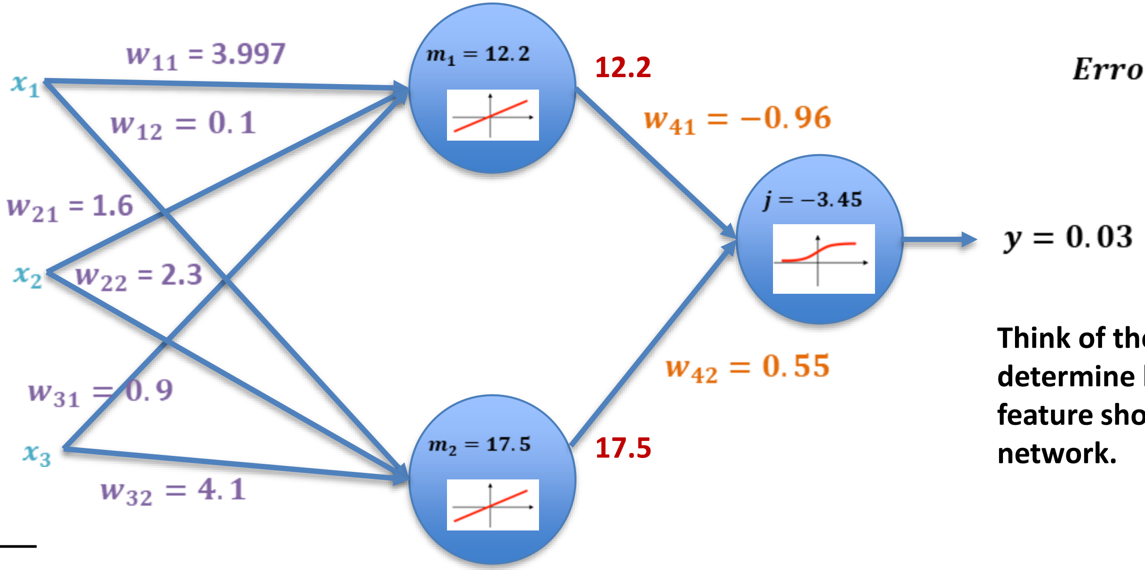
| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$$\Delta w_{21} = \frac{\delta E}{\delta w_{21}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta w_{21}} = \frac{\delta E}{\delta j}\frac{\delta j}{\delta m_2}\frac{\delta m_2}{\delta w_{21}}$$

**Chain Rule again**
**and the process continues …**

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$



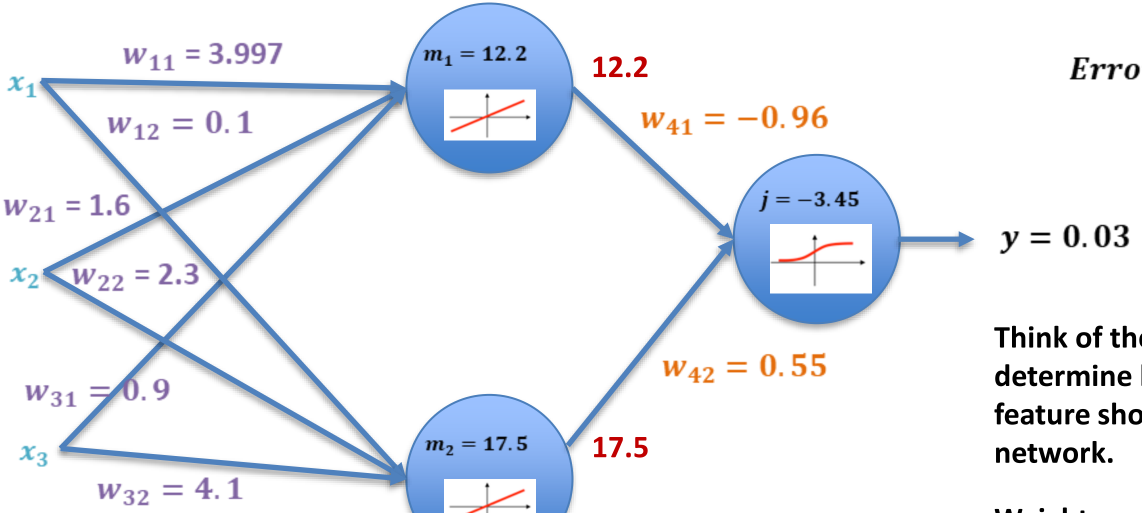

$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

**Think of the activation function to determine how much of an input feature should progress through the network.**

| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

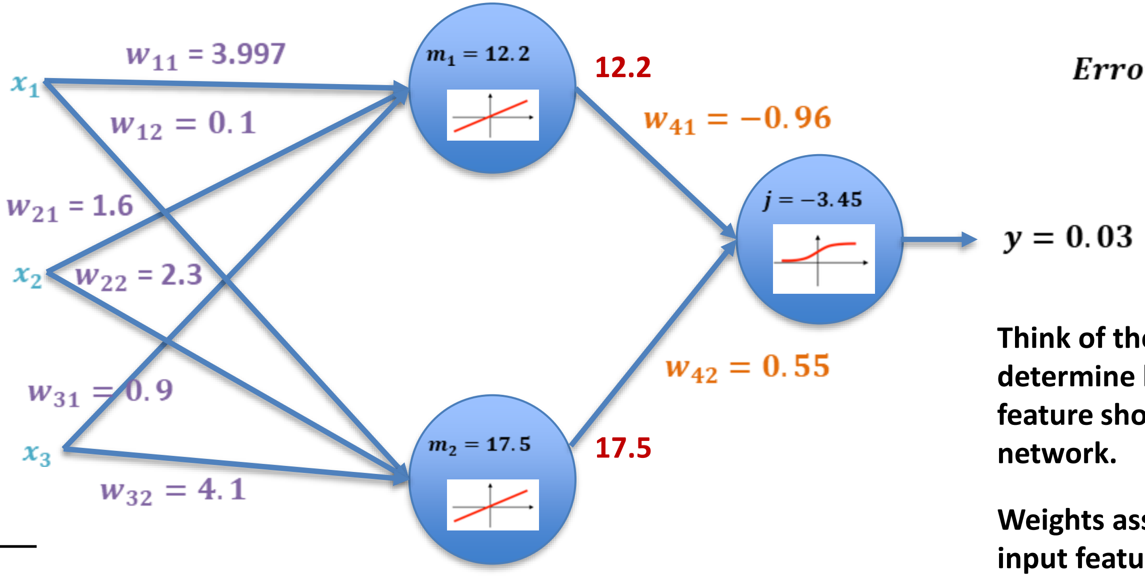
| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

**Think of the activation function to determine how much of an input feature should progress through the network.**

**Weights assign significance to those input features.**

$\varepsilon = 0.1$

$\boxed{w_i = w_i + \varepsilon E x_i}$




$Error\ (E)$ = label – y
= 1 – (0.03)
= 0.97

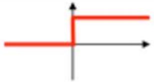
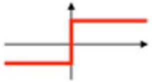
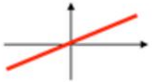
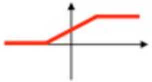
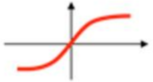
| | |
|---|---|
| $x_1$ | 1 |
| $x_2$ | 4 |
| $x_3$ | 2 |
| *label* | 1 |

**Think of the activation function to determine how much of an input feature should progress through the network.**

**Weights assign significance to those input features.**

**Modify the weights based on the resulting error**

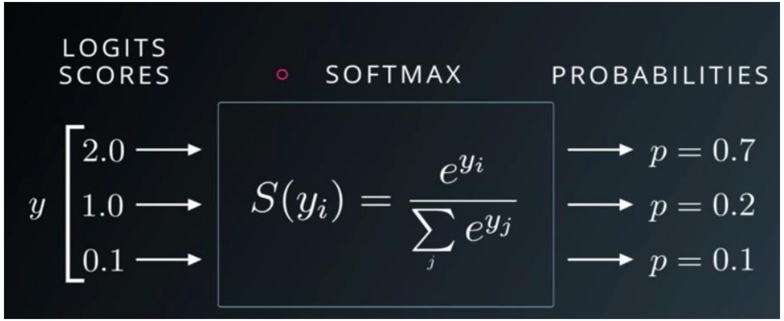
# Activation Functions

| Activation function | Equation | Example | 1D Graph |
|---|---|---|---|
| Unit step (Heaviside) | $\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$ | Perceptron variant |  |
| Sign (Signum) | $\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$ | Perceptron variant |  |
| Linear | $\phi(z) = z$ | Adaline, linear regression |  |
| Piece-wise linear | $\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$ | Support vector machine |  |
| Logistic (sigmoid) | $\phi(z) = \frac{1}{1 + e^{-z}}$ | Logistic regression, Multi-layer NN |  |
| Hyperbolic tangent | $\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ | Multi-layer NN |  |

# SoftMax or Sigmoid

The **sigmoid** function is used for the two-class logistic regression

The **softmax** function is an extension of the sigmoid function to be used for the multiclass logistic regression -- they return a probability distribution over the classes




VCU College of Engineering

# Questions?

# Types of NN

- Recurrent Neural Networks
  - LSTM
  - bi-LSTM
- Convolutional Neural Networks

# Recurrent Neural Networks

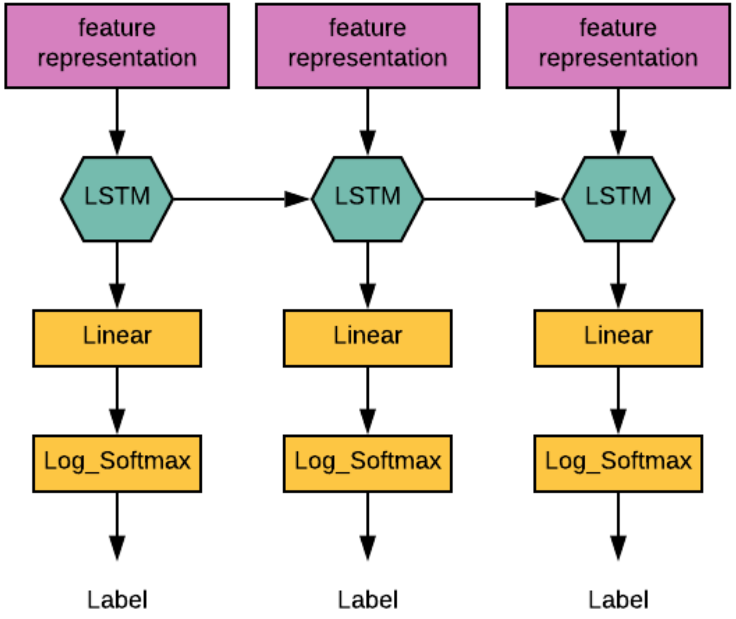
- Often used for sequence classification tasks
- They remember a history

Long Short Term Memory (LSTM) Units



Thank you, Jorge!

# Long Short Term Memory Units




Feature vector

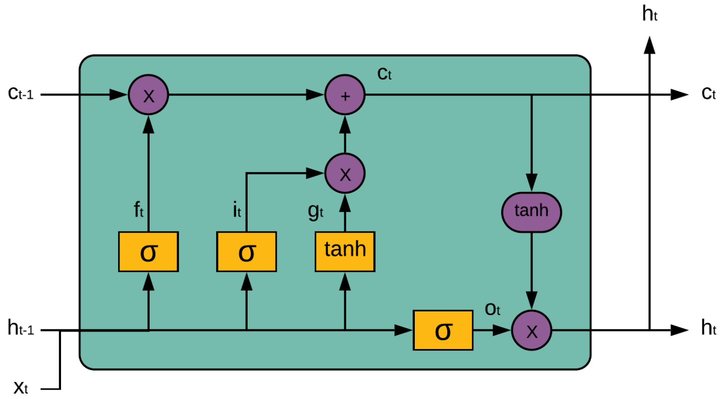
LSTM Layer

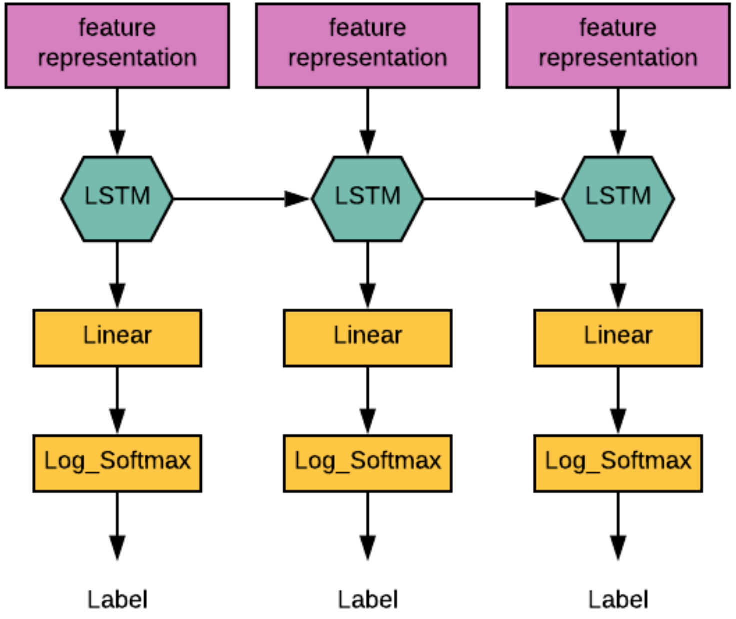
Linear Layer

LogSoftmax

Output

# Long Short Term Memory Units




Feature vector

LSTM Layer

Linear Layer

LogSoftmax

Output

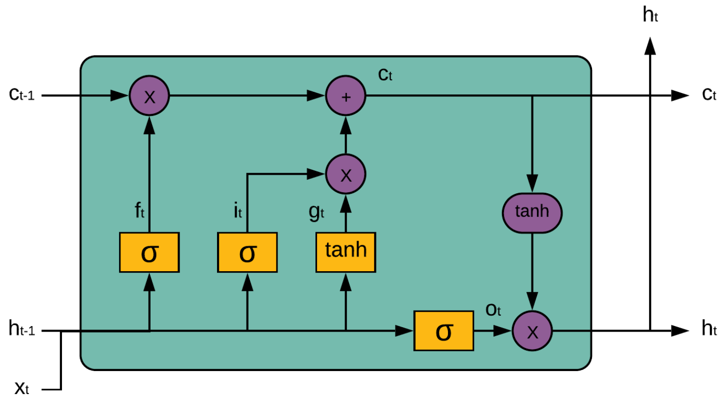
# LSTM Cell

x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

$\boldsymbol{\sigma}$ = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate
*Gates are always Wx + b + Wh + b*




$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$c_t = f_t * c_{(t-1)} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

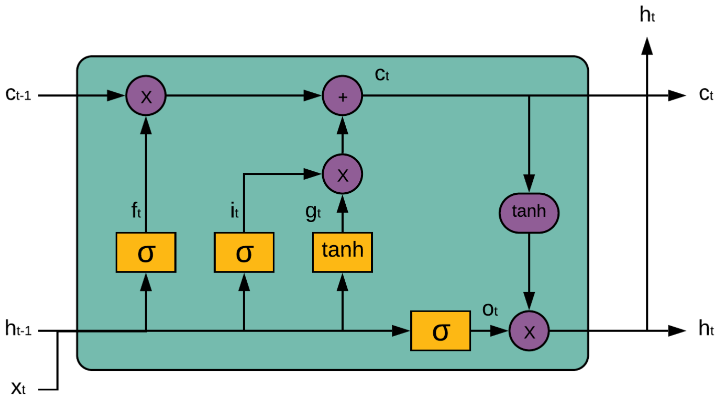
# LSTM Cell

x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

$\boldsymbol{\sigma}$ = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate
*Gates are always Wx + b + Wh + b*




$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t * c_{(t-1)} + i_t * g_t \\
h_t &= o_t * \tanh(c_t)
\end{aligned}
$$

# LSTM Cell

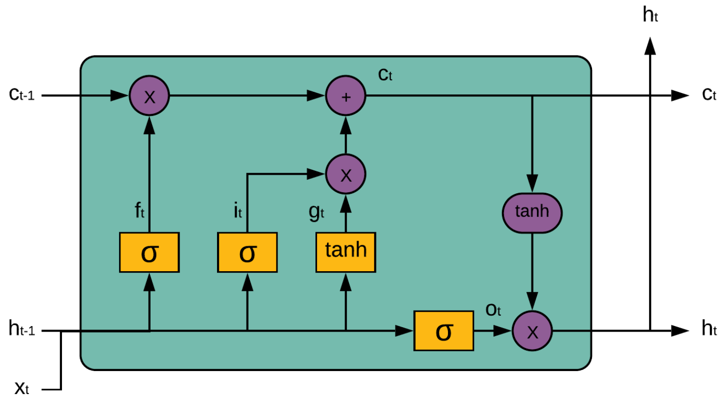
x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

**σ** = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

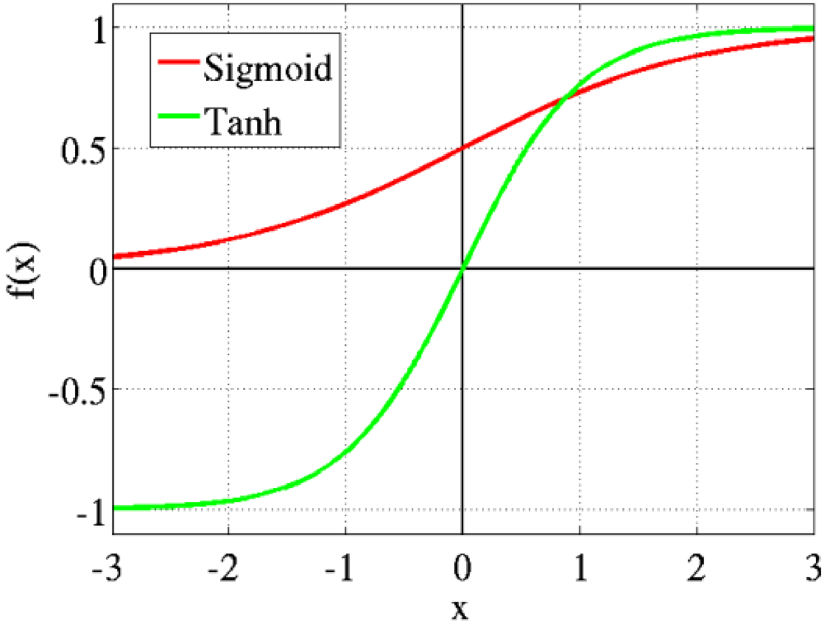
$$\begin{array}{c} G \\ \begin{bmatrix} 3 & 5 & 7 \\ 4 & 9 & 8 \end{bmatrix} \end{array} \circ \begin{array}{c} H \\ \begin{bmatrix} 1 & 6 & 3 \\ 0 & 2 & 9 \end{bmatrix} \end{array} = \begin{array}{c} N \\ \begin{bmatrix} 3\times1 & 5\times6 & 7\times3 \\ 4\times0 & 9\times2 & 8\times9 \end{bmatrix} \end{array}$$

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate
*Gates are always Wx + b + Wh + b*




$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$c_t = f_t * c_{(t-1)} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

# Activation Functions




$$\mathrm{Sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

# Input Gate

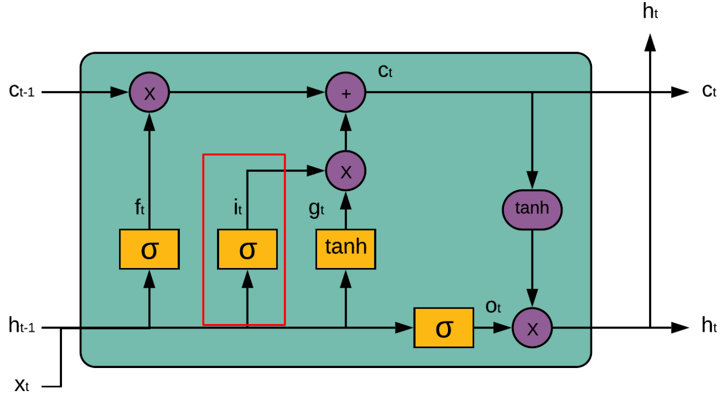
x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

$\boldsymbol{\sigma}$ = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate

*Gates are always W**x** + b + W**h** + b*




$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t * c_{(t-1)} + i_t * g_t \\
h_t &= o_t * \tanh(c_t)
\end{aligned}
$$

# Input Gate

x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

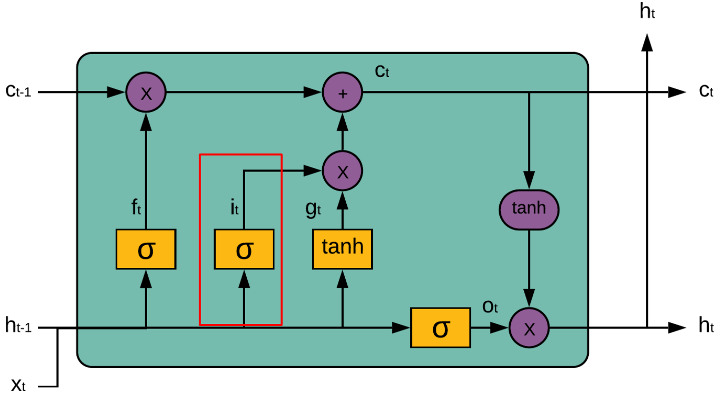
$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

$\boldsymbol{\sigma}$ = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate

This decides the parts of the input that we want to output based on our learned weights

*Gates are always Wx + b + Wh + b*




$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$c_t = f_t * c_{(t-1)} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

# Forget Gate

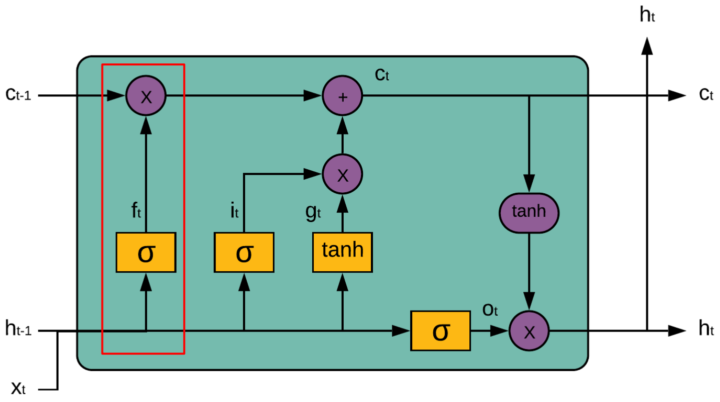
x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

$\boldsymbol{\sigma}$ = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate

*Gates are always Wx + b + Wh + b*




$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$\boxed{f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$c_t = f_t * c_{(t-1)} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

# Forget Gate

x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

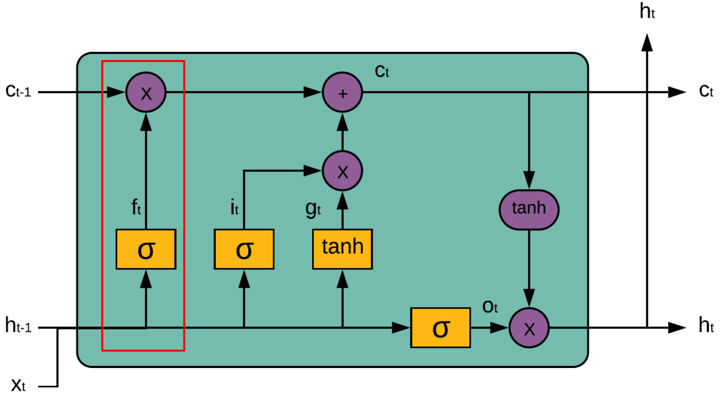
$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

$\boldsymbol{\sigma}$ = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate

1 = completely keep this
0 = completely get rid of this

*Gates are always Wx + b + Wh + b*




$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$\boxed{f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$c_t = f_t * c_{(t-1)} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

# Forget Gate

x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

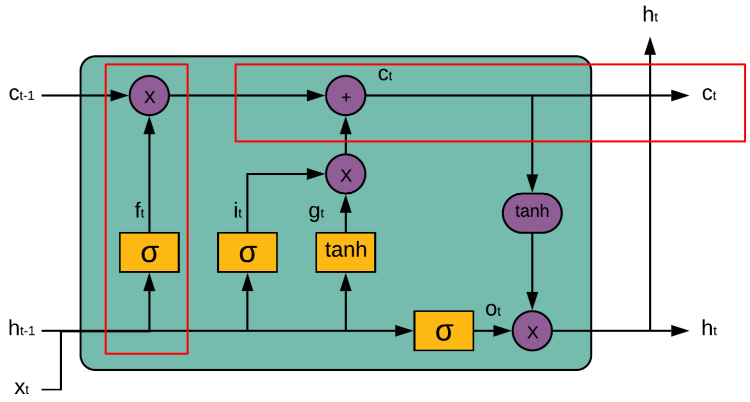
$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

$\boldsymbol{\sigma}$ = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate

This is combined with our input and is saying how much of the input do we want to pass on

*Gates are always Wx + b + Wh + b*




$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$\boxed{f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$\boxed{c_t = f_t * c_{(t-1)} + i_t * g_t}$$
$$h_t = o_t * \tanh(c_t)$$

# Cell Gate

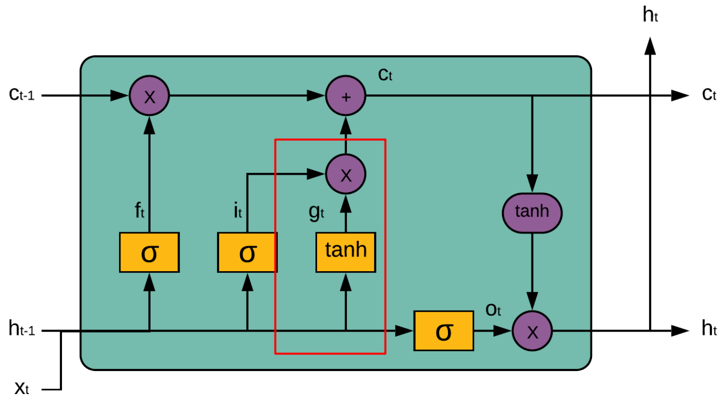
x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

**σ** = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate

*Gates are always W**x** + b + W**h** + b*




$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$
$$\boxed{g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$c_t = f_t * c_{(t-1)} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

# Cell Gate

x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

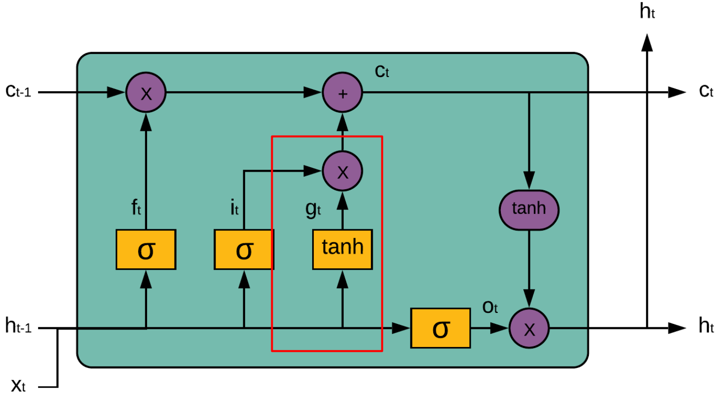
$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

$\boldsymbol{\sigma}$ = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate

This is the cell state that we squash between -1 and 1

*Gates are always Wx + b + Wh + b*




$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$c_t = f_t * c_{(t-1)} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

# Cell Gate

x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

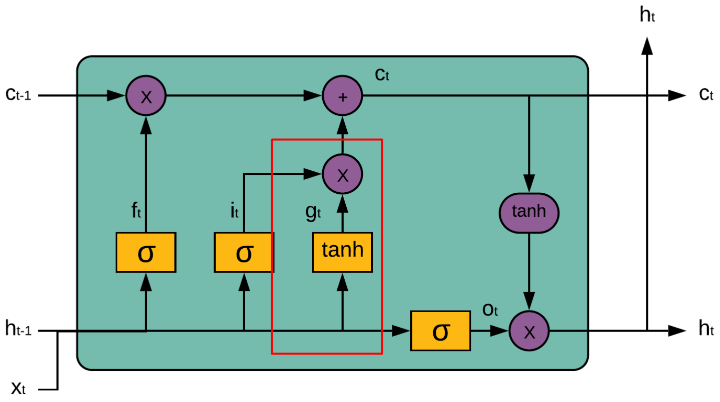
$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

$\boldsymbol{\sigma}$ = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
$\boxed{g \text{ = cell gate}}$
*o* = output gate

Then it is multiplied with our sigmoid gate so we only output the parts of our input that we want to.

*Gates are always W**x** + b + W**h** + b*




$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$
$$\boxed{g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$c_t = f_t * c_{(t-1)} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

# Output Gate

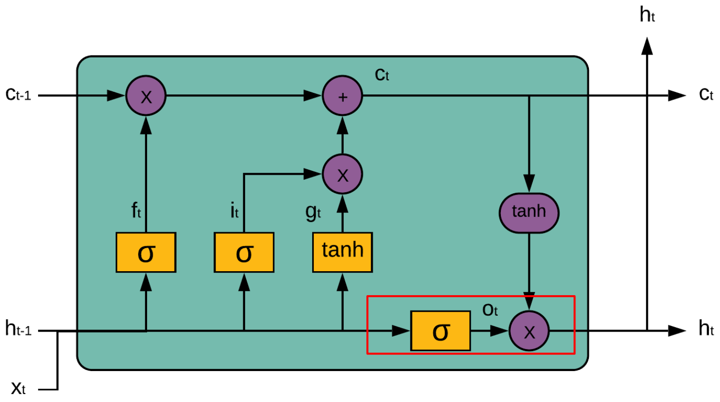
x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

$\boldsymbol{\sigma}$ = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate

*Gates are always W**x** + b + W**h** + b*




$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$c_t = f_t * c_{(t-1)} + i_t * g_t$$
$$h_t = o_t * \tanh(c_t)$$

# new Cell state

x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

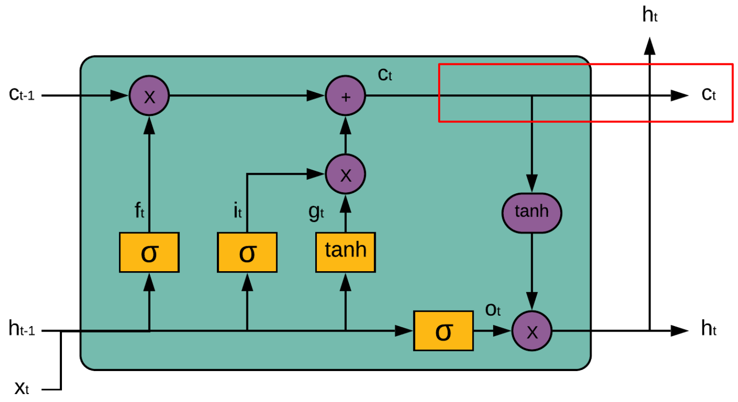
$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

$\boldsymbol{\sigma}$ = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate

This is the information that we want to continue to pass on to the next cell

*Gates are always Wx + b + Wh + b*




$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$\boxed{c_t = f_t * c_{(t-1)} + i_t * g_t}$$
$$h_t = o_t * \tanh(c_t)$$

# new Hidden state

x = input
h = hidden state
c = cell state
*t* = current cell (timestep)

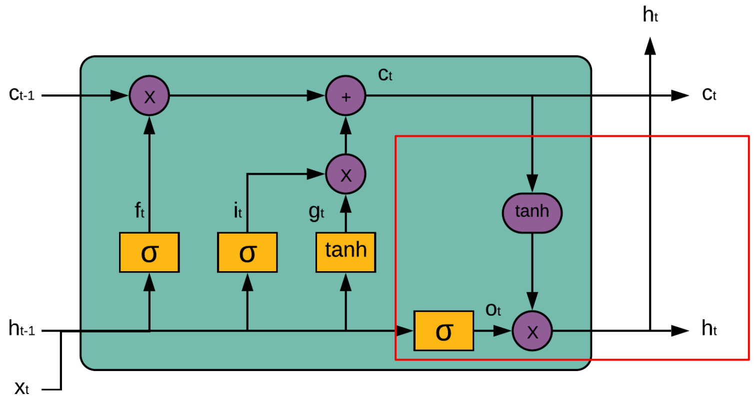
$W_{yz}$ = weights for y at gate z
$b_{yz}$ = bias for y at gate z

**σ** = sigmoid
tanh = hyperbolic tangent
* = Hadamard product (not matrix multiplication)

*i* = input gate
*f* = forget gate
*g* = cell gate
*o* = output gate

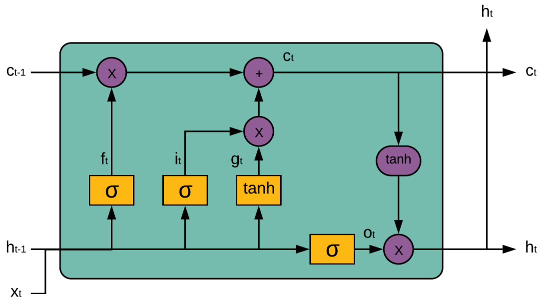
This is the hidden state of the cell (the weights)

*Gates are always W**x** + b + W**h** + b*

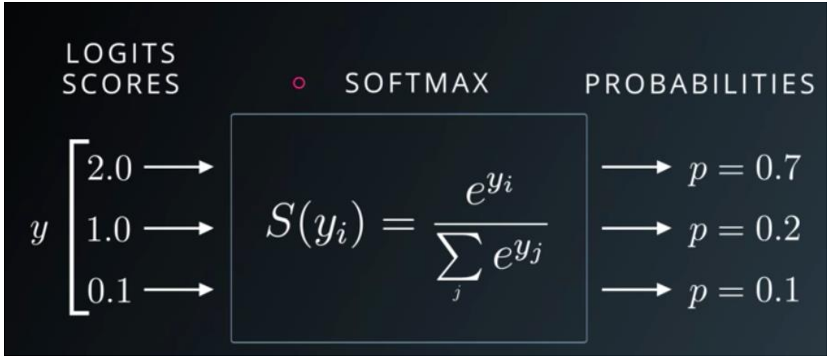


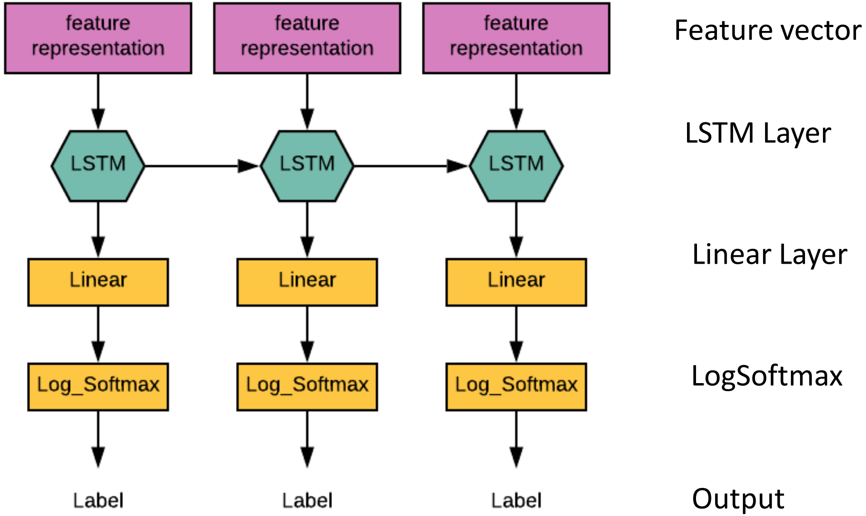

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$
$$c_t = f_t * c_{(t-1)} + i_t * g_t$$
$$\boxed{h_t = o_t * \tanh(c_t)}$$

# LSTM

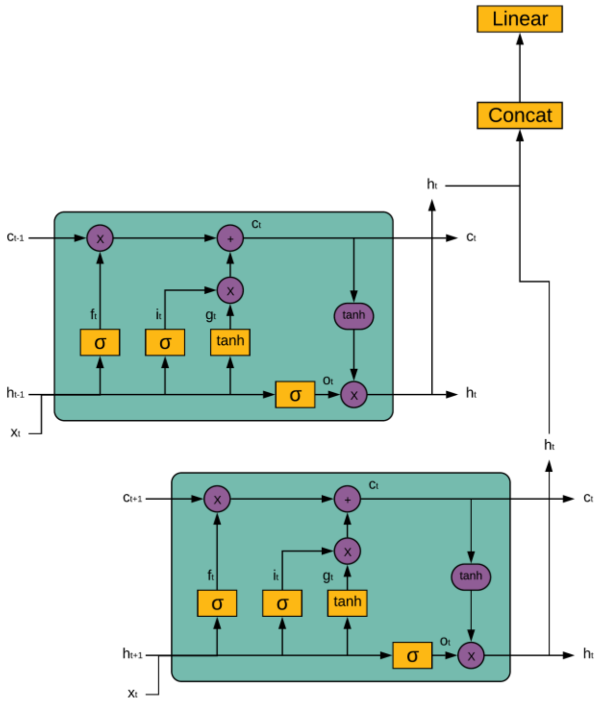


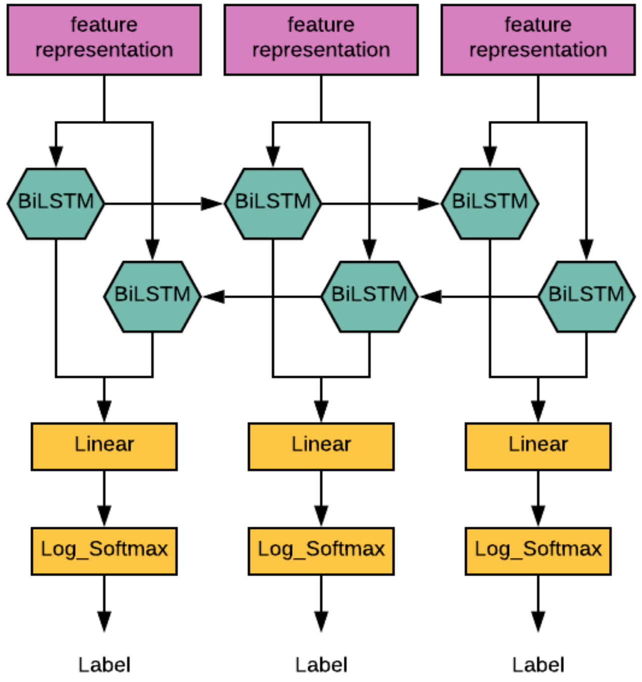

$$\mathrm{LogSoftmax}(x_i) = \log\left(\frac{\exp(x_i)}{\sum_j \exp(x_j)}\right)$$



# Bi-LSTM




Feature vectors

LSTM Layer

Linear Layer

LogSoftmax

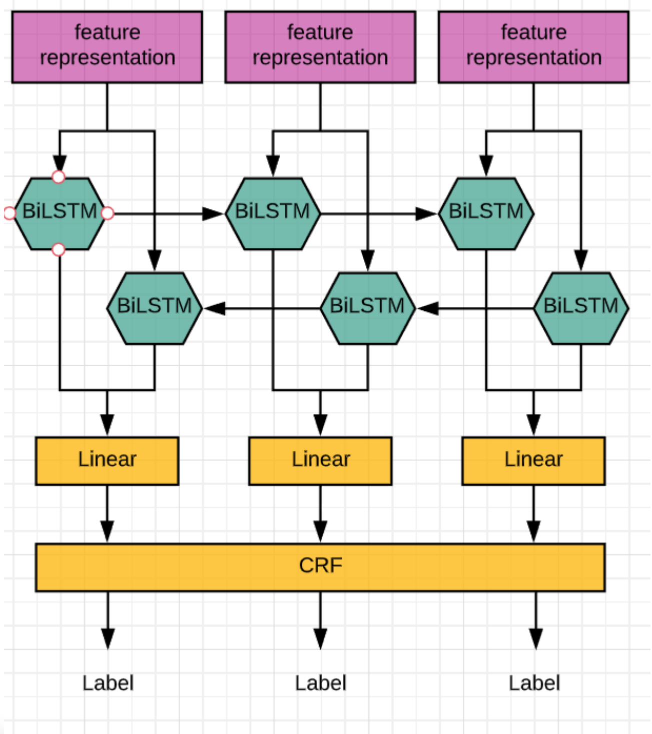
Output

# BiLSTM-CRF

$$P(\mathbf{y} \mid \mathbf{X}) = \prod_{k=1}^{\ell} P(y_k \mid \mathbf{x}_k)$$

$$= \prod_{k=1}^{\ell} \frac{\exp(U(\mathbf{x}_k, y_k))}{Z(\mathbf{x}_k)}$$

$$= \frac{\exp\left(\sum_{k=1}^{\ell} U(\mathbf{x}_k, y_k)\right)}{\prod_{k=1}^{\ell} Z(\mathbf{x}_k)}$$




Feature vectors

LSTM Layer

Linear Layer

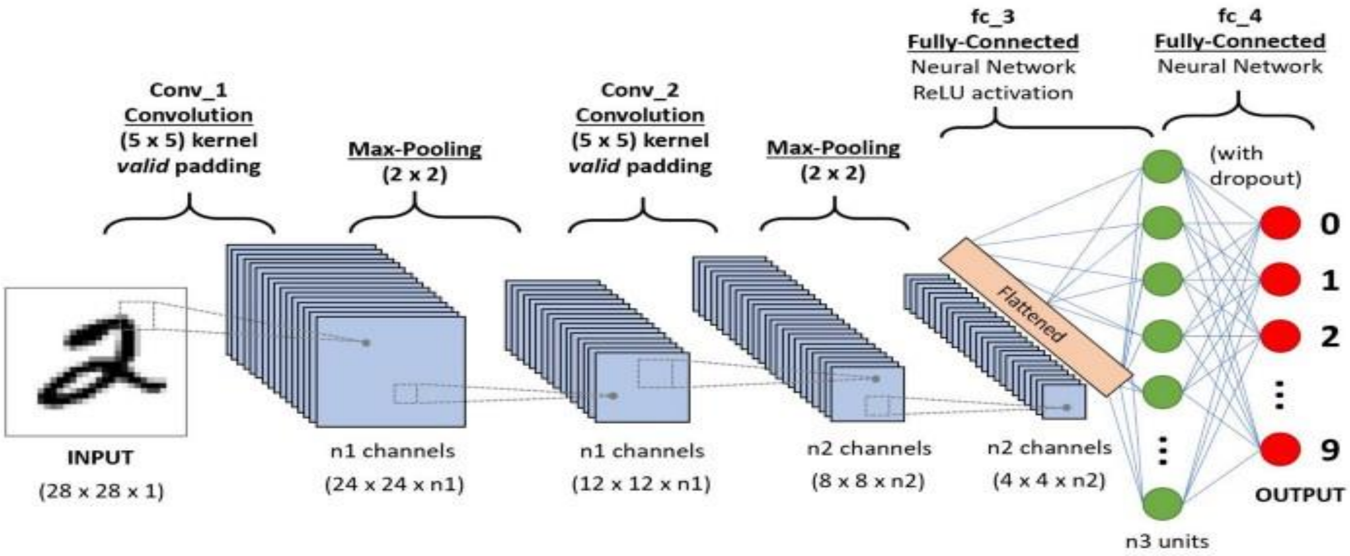
CRF

Output

# Questions?

# Convolutional Neural Network

"convolutional neural network" employ a mathematical operation called convolution

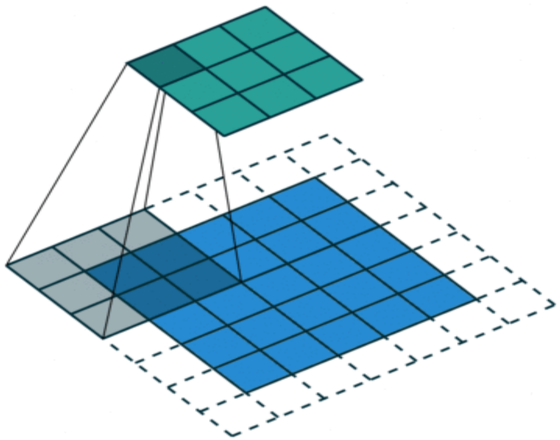


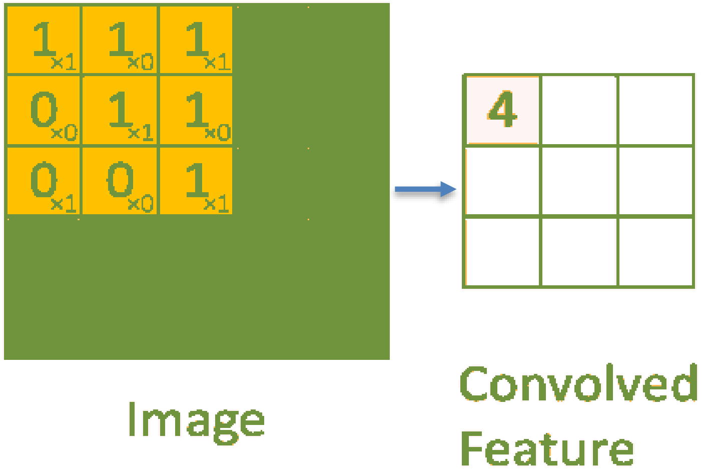



Thank you, Samantha!

# Convolution Layer

- Traverses a filter over the input layer
- Results are summed with the bias to give a squashed one-depth channel Convoluted Feature Output





# Pooling layer

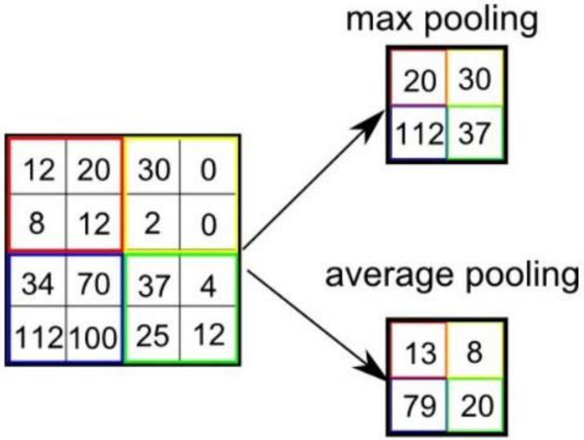
Two types of Pooling :

Max Pooling - returns the maximum value from the portion of the image covered by the Kernel

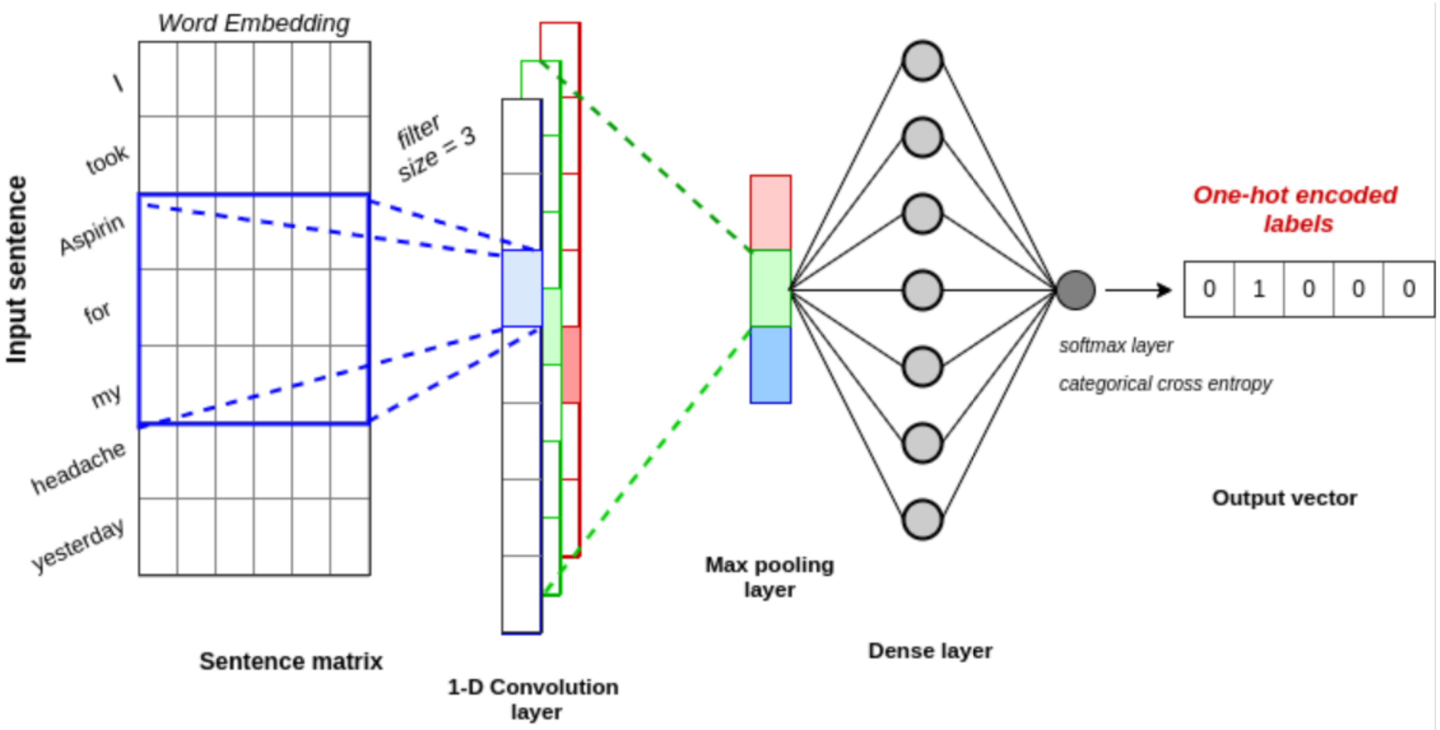
a. discards the noisy activations
b. performs dimensionality reduction

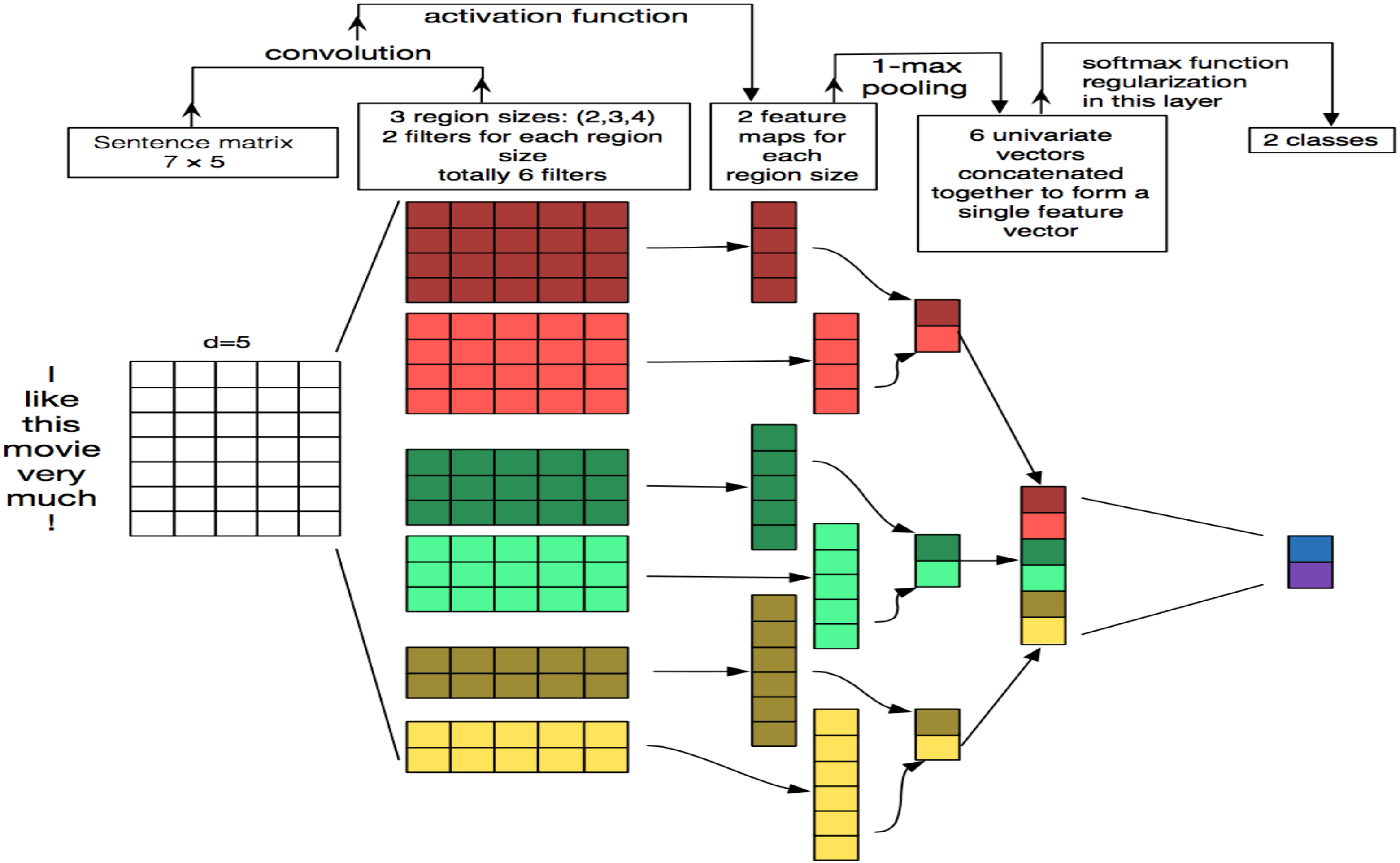
Average Pooling - returns the average of all the values from the portion of the image covered by the Kernel

a. performs dimensionality reduction



# simple CNN applied to NLP




activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 x 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

# Questions?

# Evaluation Metrics

A model's predictive ability on a test set is measured utilizing evaluation metrics.



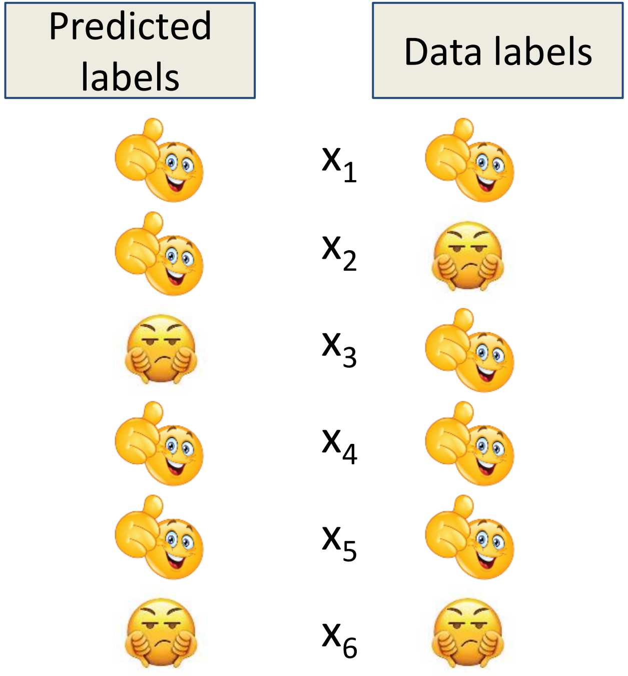
Thank you, Andriy!

# Evaluation Metrics

Data labels

$x_1$
$x_2$
$x_3$
$x_4$
$x_5$
$x_6$

# Evaluation Metrics



# Evaluation Metrics

| | Predicted labels | | Data labels |
|---|---|---|---|
| True Positive |  | $x_1$ |  |
| False Positive |  | $x_2$ |  |
| False Negative |  | $x_3$ |  |
| True Positive |  | $x_4$ |  |
| True Positive |  | $x_5$ |  |
| True Negative |  | $x_6$ |  |

# Evaluation Metrics

| | PP | PN |
|---|---|---|
| P | TP = 3 | FN = 1 |
| N | FP = 1 | TN = 1 |

| Predicted labels | | Data labels |
|---|---|---|
| True Positive | $x_1$ | |
| False Positive | $x_2$ | |
| False Negative | $x_3$ | |
| True Positive | $x_4$ | |
| True Positive | $x_5$ | |
| True Negative | $x_6$ | |

# Evaluation Metrics

| | PP | PN |
|---|---|---|
| P | TP = 3 | FN = 1 |
| N | FP = 1 | TN = 1 |

How relevant are our predictions?

$$\text{Precision} = \frac{tp}{tp + fp}$$

| | Predicted labels | | Data labels |
|---|---|---|---|
| True Positive | | $x_1$ | |
| False Positive | | $x_2$ | |
| False Negative | | $x_3$ | |
| True Positive | | $x_4$ | |
| True Positive | | $x_5$ | |
| True Negative | | $x_6$ | |

# Evaluation Metrics

| | PP | PN |
|---|---|---|
| P | TP = 3 | FN = 1 |
| N | FP = 1 | TN = 1 |

| | Predicted labels | | Data labels |
|---|---|---|---|
| True Positive | | $x_1$ | |
| False Positive | | $x_2$ | |
| False Negative | | $x_3$ | |
| True Positive | | $x_4$ | |
| True Positive | | $x_5$ | |
| True Negative | | $x_6$ | |

How sensitive is our system to the target class?

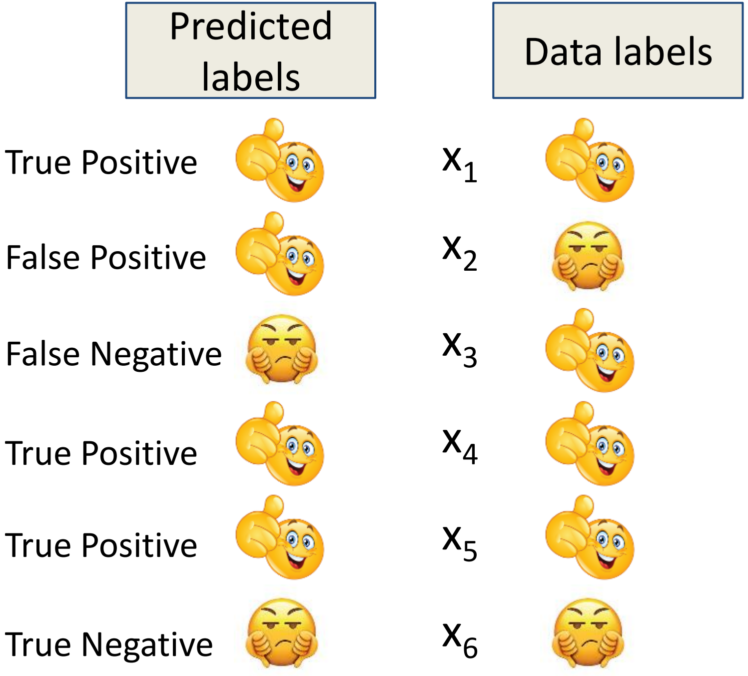
$$\text{Recall} = \frac{tp}{tp + fn} \qquad = 3/4$$

# Evaluation Metrics

| | PP | PN |
|---|---|---|
| P | TP = 3 | FN = 1 |
| N | FP = 1 | TN = 1 |

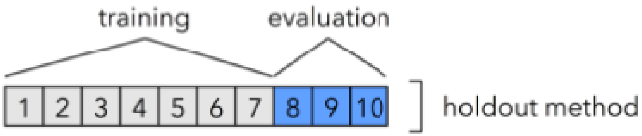
Harmonic mean between precision and recall

$$F_1 = \left( \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} \right) = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} .$$



# Evaluation Methodologies

Hold out validation:




- Don't train on all available data.
- If dataset is small, hold out set may not be a representative sample.

Cross validation:

- Partition data into *n* disjoint subsets
- Use each subset as a testing set and train on the remaining subsets - average metrics.

# Hyper parameter tuning

- Number of parameters that need to be tuned
  - For example
    - Learning rate
    - Epochs
    - Window size

- Conducted over a validation set that is separate from the training and test
  - Tune your hyper parameters over the validation set
  - Evaluate:
    - hold out
    - cross validation
  - Your validation set must be independent of your training and test

# Questions?