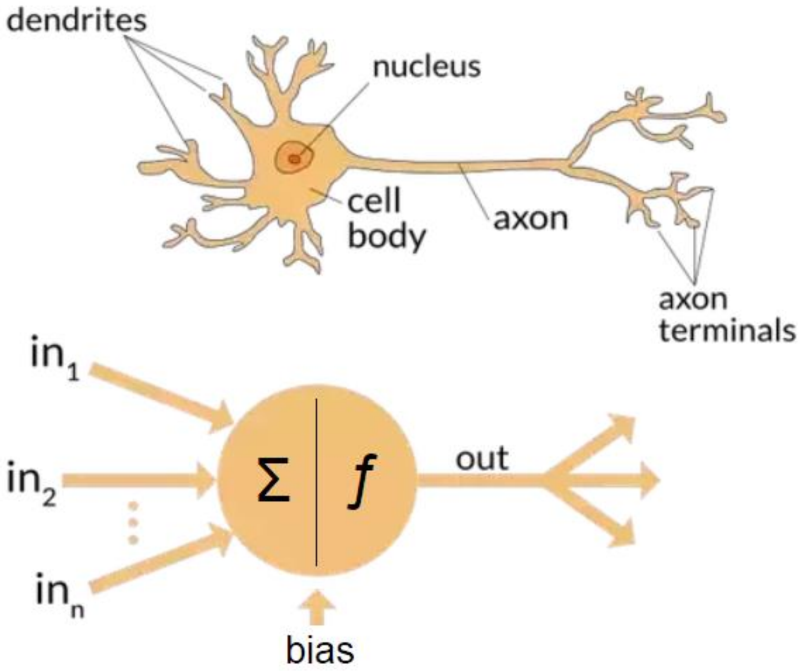
Artificial Neural Networks

DATA SCIENCE & MACHINE LEARNING

ANN or NN

While the term “Neural Networks” is commonly used in DS, ML and AI communities, “Artificial Neural Networks” is probably the more correct term to use to distinguish ANN from biological neural networks.



HTTPS://WWW.QUORA.COM/WHAT-IS-THE-DIFFERENCES-BETWEEN-ARTIFICIAL-NEURAL-NETWORK-COMPUTER-SCIENCE-AND-BIOLOGICAL-NEURAL-NETWORK

Some History

Warren McCulloch and Walter Pitts(1943) created a computational model for neural networks based on mathematics and algorithms called threshold logic. This model paved the way for neural network research to split into two approaches. One approach focused on biological processes in the brain while the other focused on the application of neural networks to artificial intelligence. This work led to work on nerve networks and their link to finite automata.

The first artificial neuron was produced in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pits. But the technology available at that time did not allow them to do too much.

SOURCE:
HTTPS://EN.WIKIPEDIA.ORG/WIKI/ARTIFICIAL_NEURAL_NETWORK
HTTPS://WWW.DOC.IC.AC.UK/~ND/SURPRISE_96/JOURNAL/VOL4/CS11/REPORT.HTML

Inspired by Biology

Ant colony optimization

- https://www.youtube.com/watch?v=eVKAIufSrHs

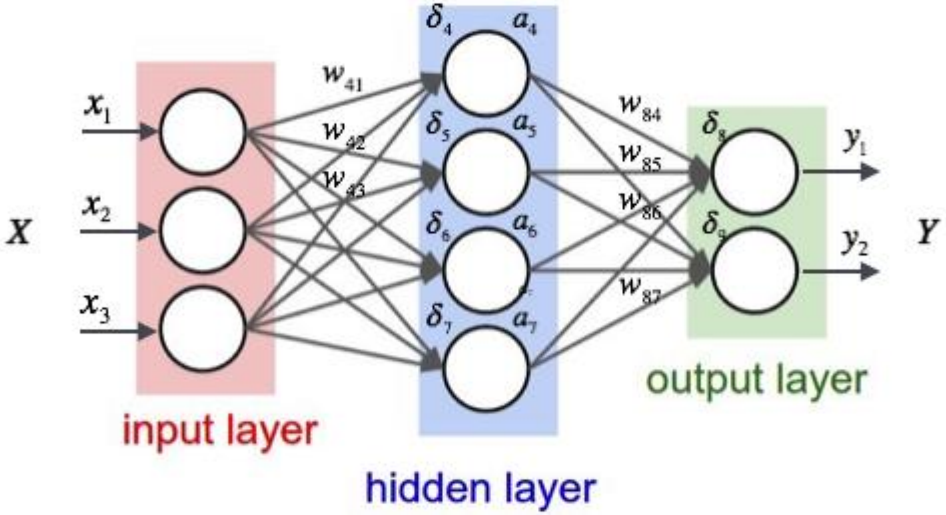
Bees algorithm

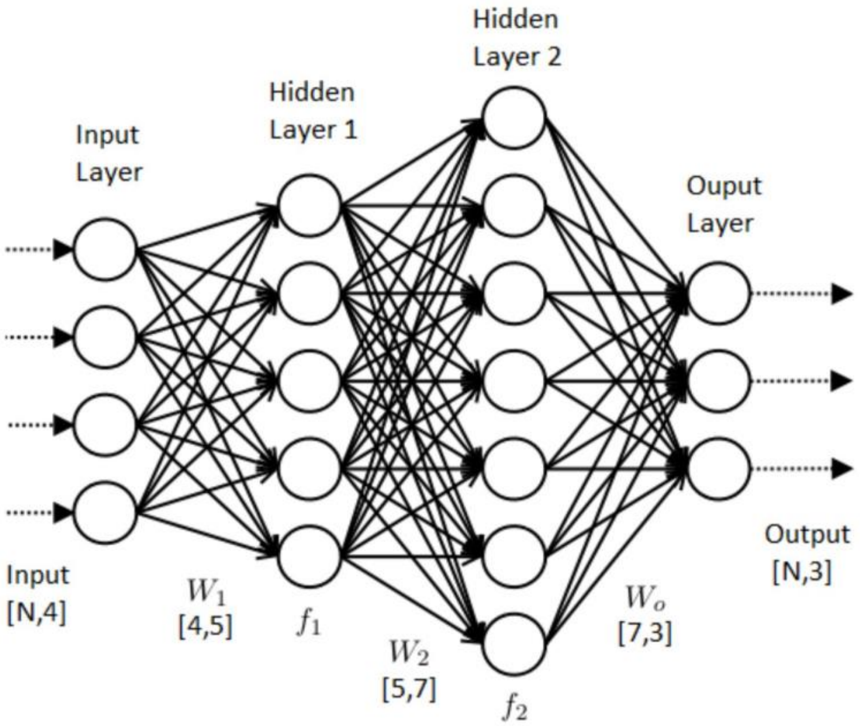
- https://www.youtube.com/watch?v=yQFXGtQwSQI
- https://www.youtube.com/watch?v=zxcb6ZBj5PE

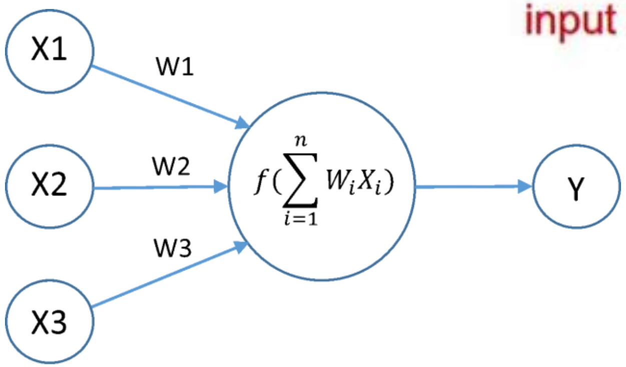
Genetic algorithm

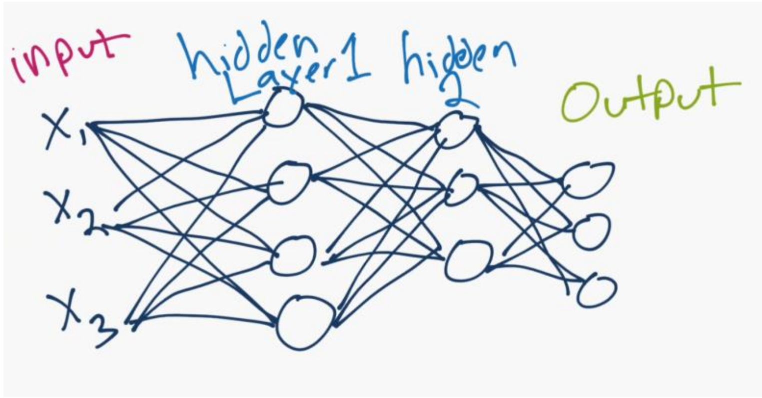
- https://www.youtube.com/watch?v=XcinBPhgT7M

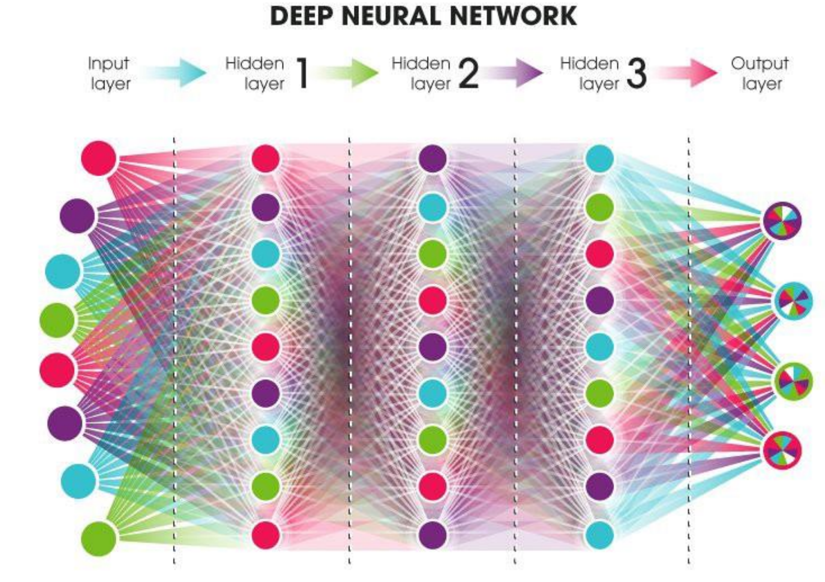
Neural Networks







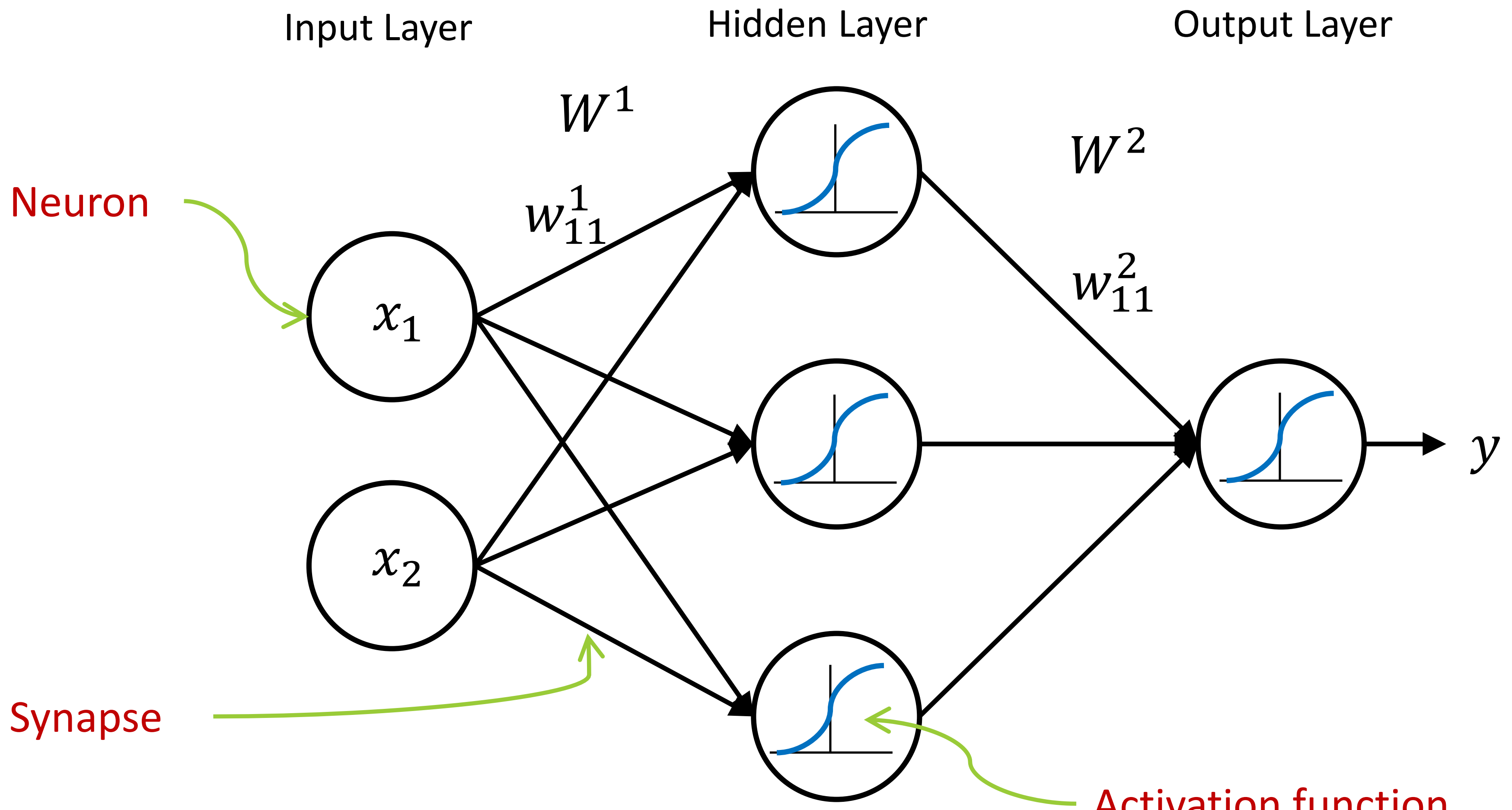




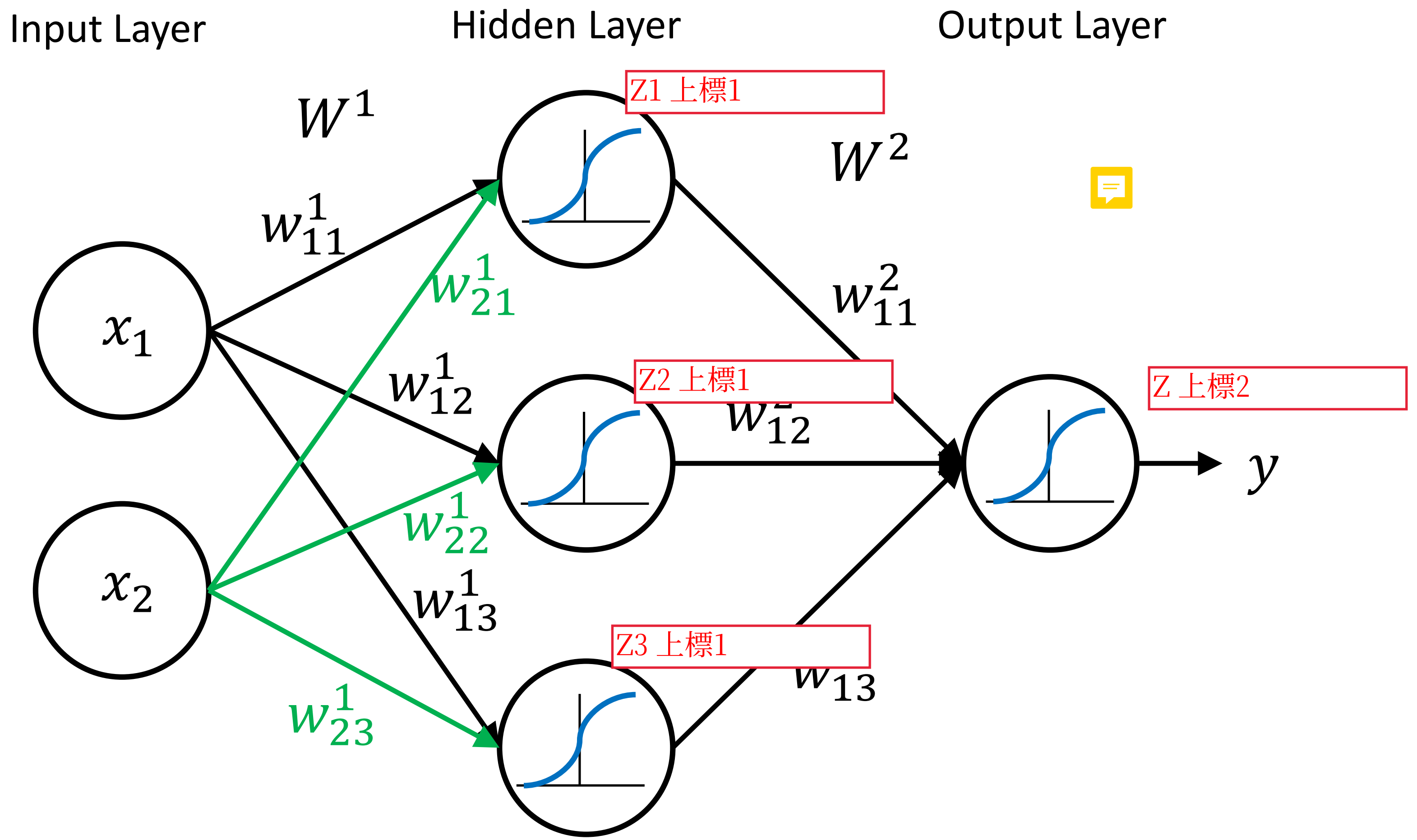
neuralnetworksanddeeplearning.com - Michael Nielsen, Yoshua Bengio, Ian Goodfellow, and Aaron Courville, 2016.

Click on image for image source

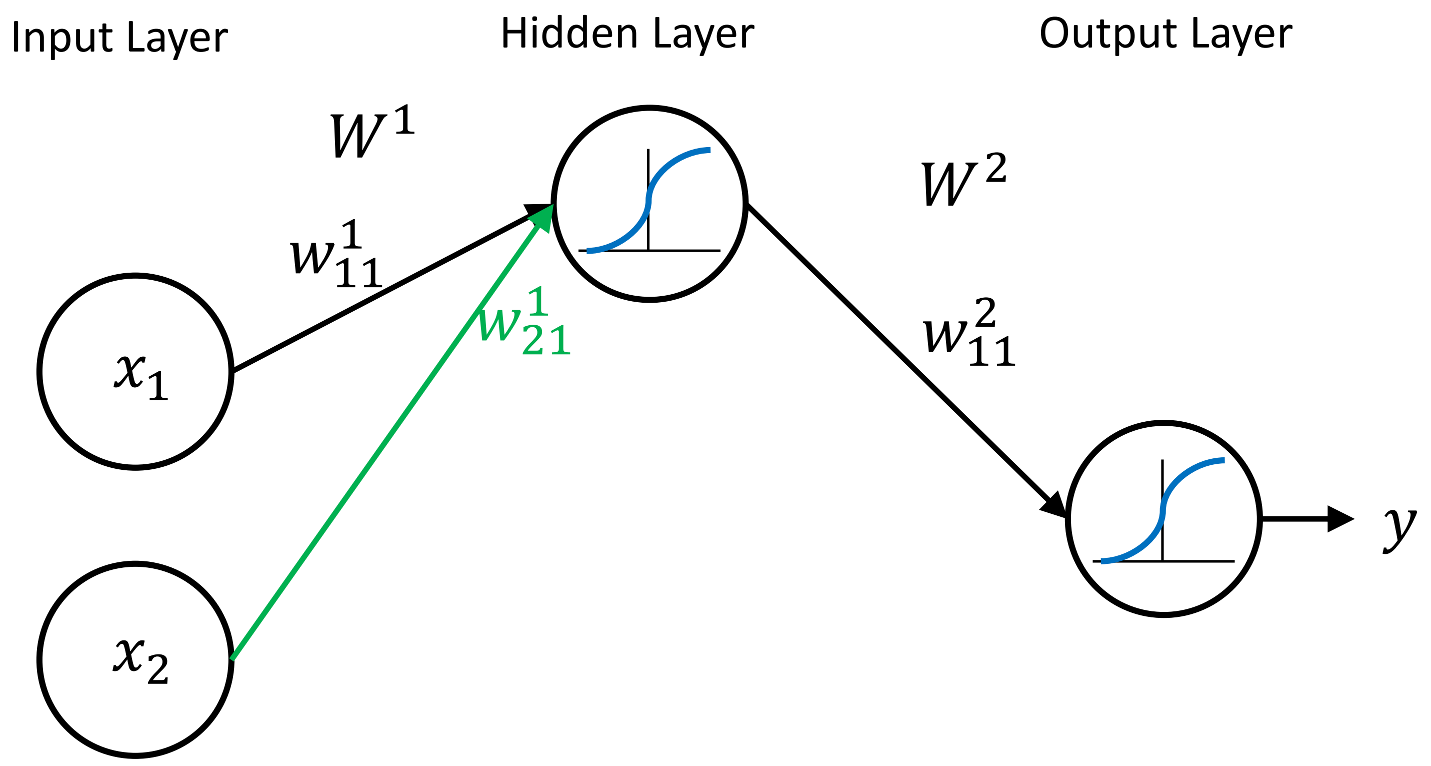
Neural Networks



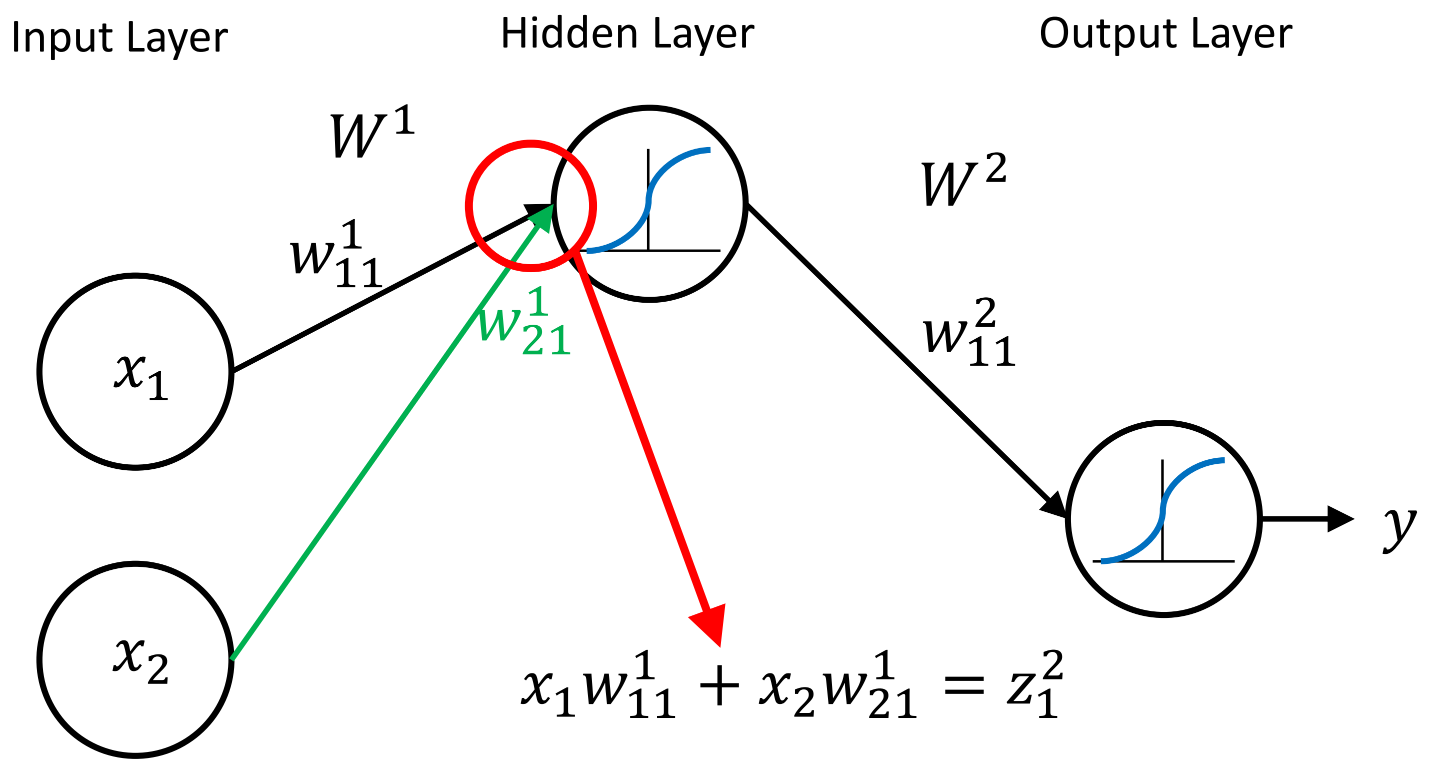
Neural Networks



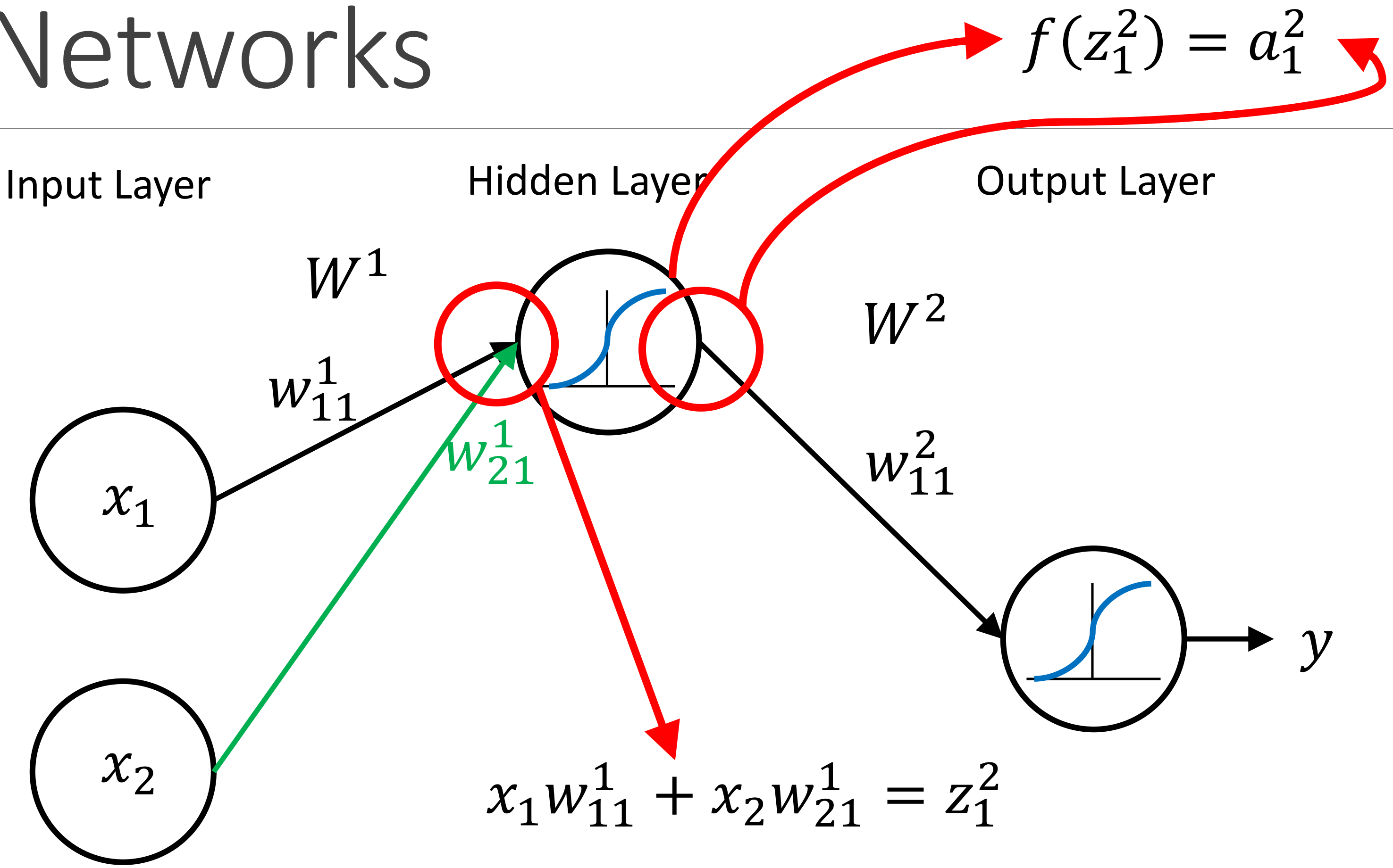
Neural Networks



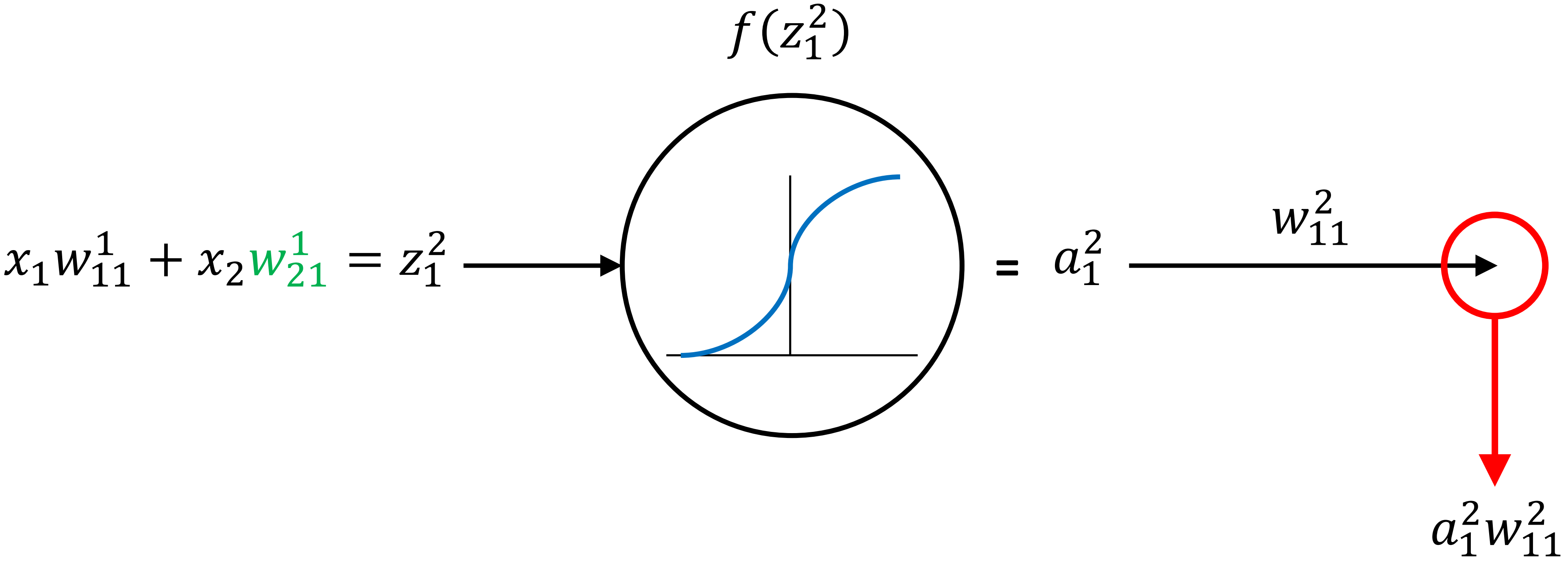
Neural Networks



Neural Networks



Neural Networks

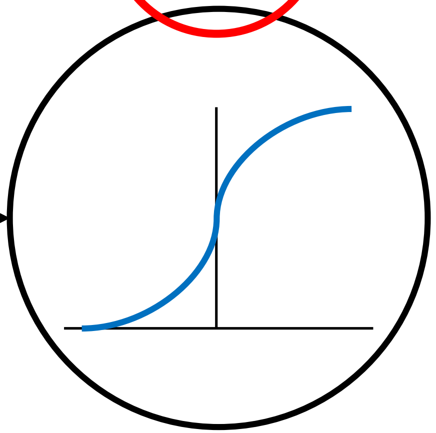


Neural Networks

$f(z_1^2)$

$f(x)$ Activation function

$$x_1 w_{11}^1 + x_2 w_{21}^1 = z_1^2$$



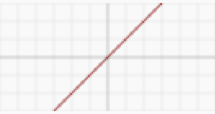
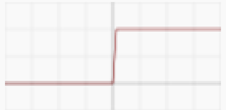
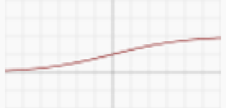
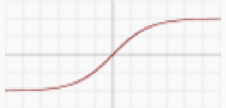
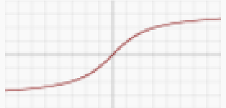
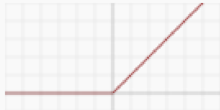
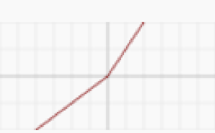
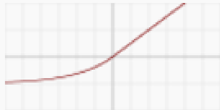
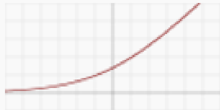
$= a_1^2$

w_{11}^2

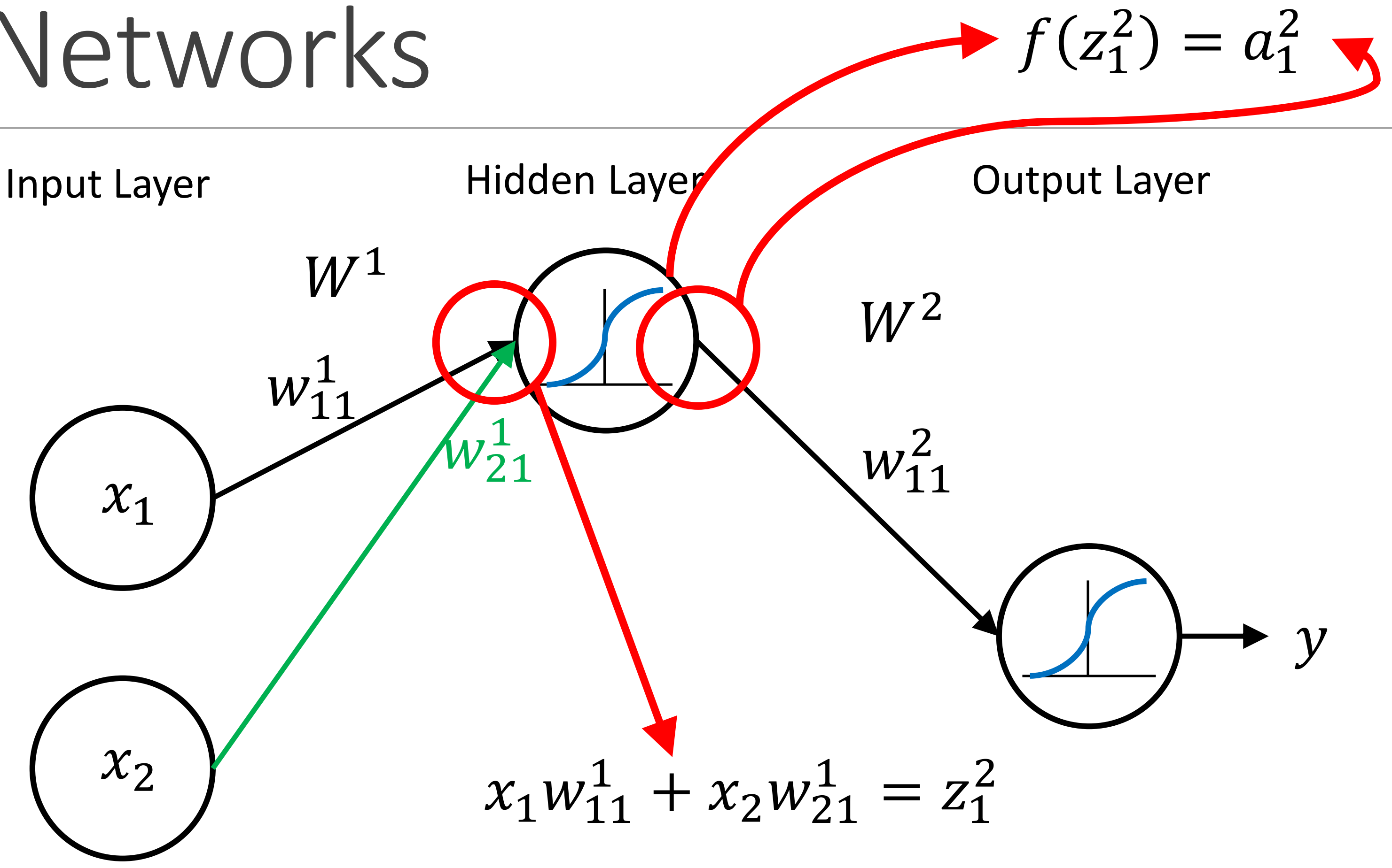
$a_1^2 w_{11}^2$

Source: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

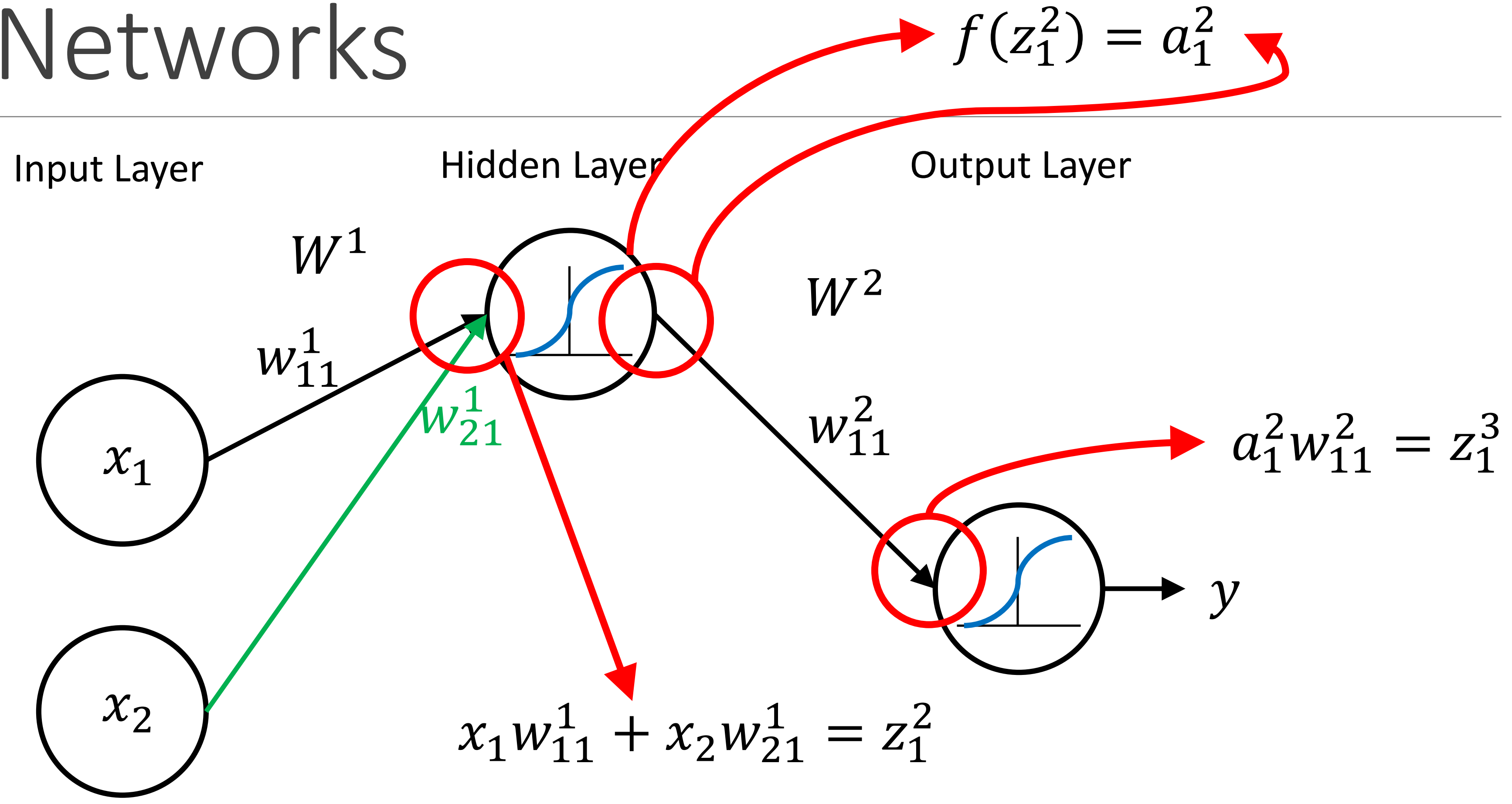
Sigmoid

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
$\boxed{\text{Logistic (a.k.a Soft step)}}$		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Neural Networks



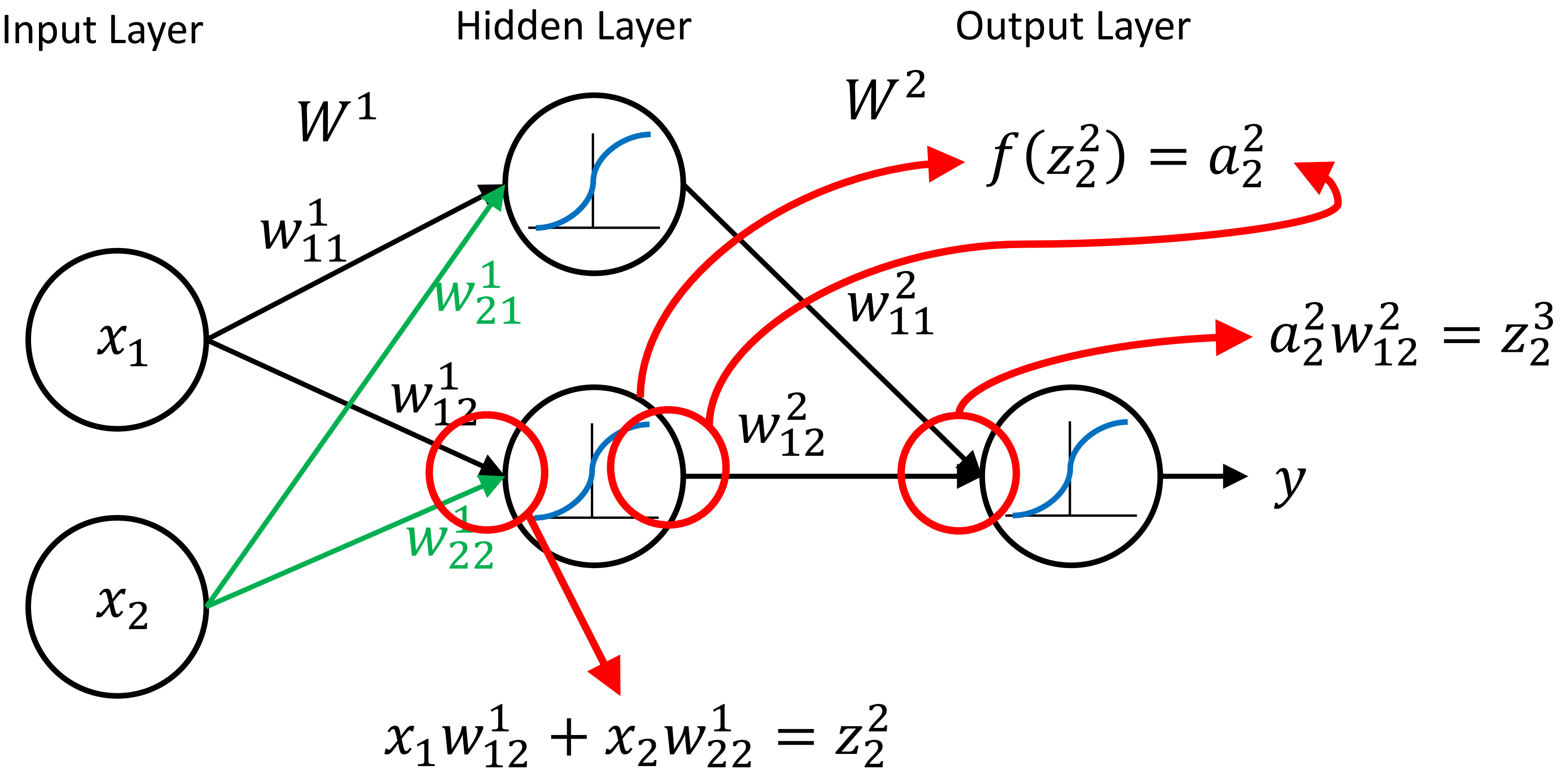
Neural Networks



Neural Networks

$$x_1 w_{11}^1 + x_2 w_{21}^1 = z_1^2$$

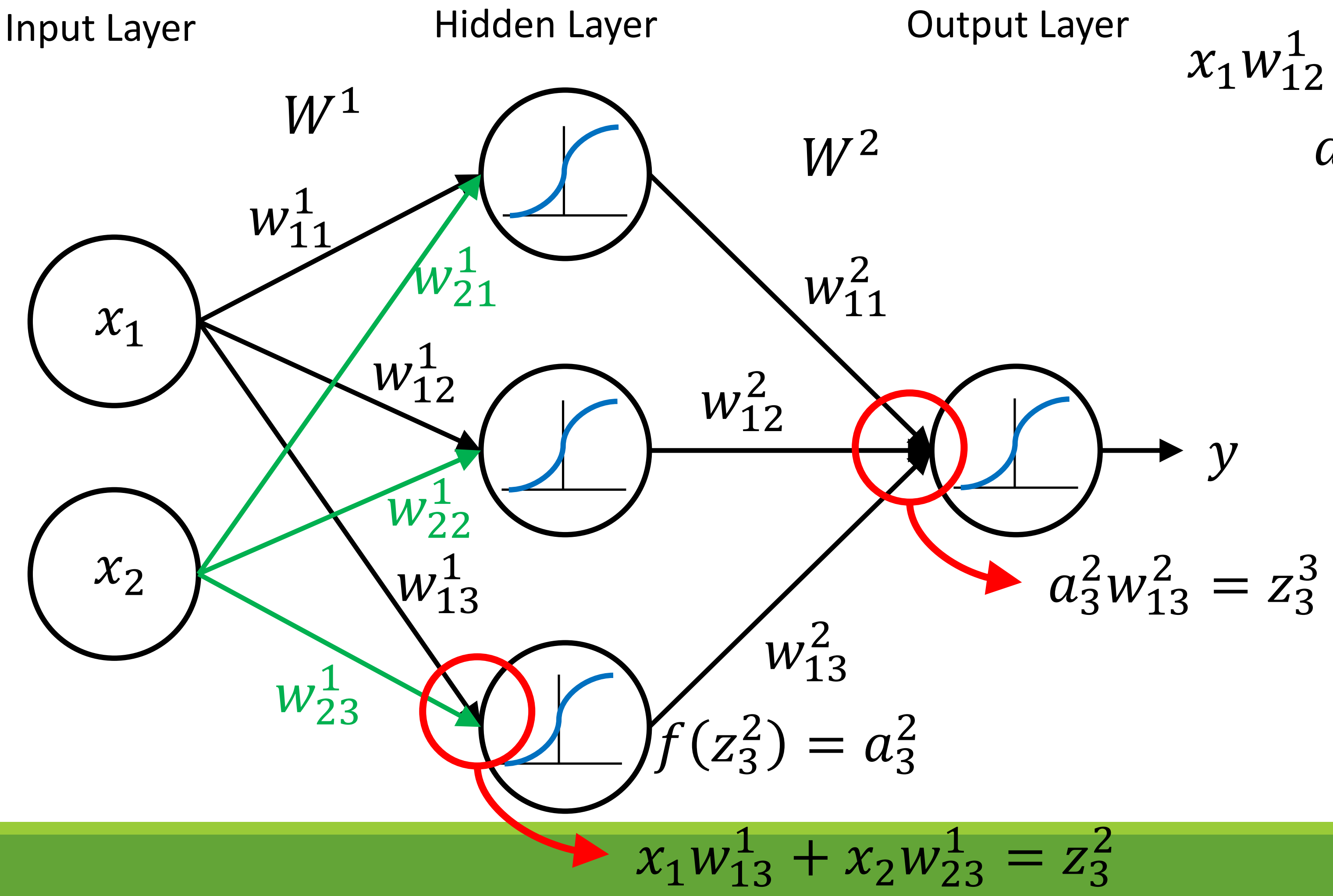
$$a_1^2 w_{11}^2 = z_1^3$$



Neural Networks

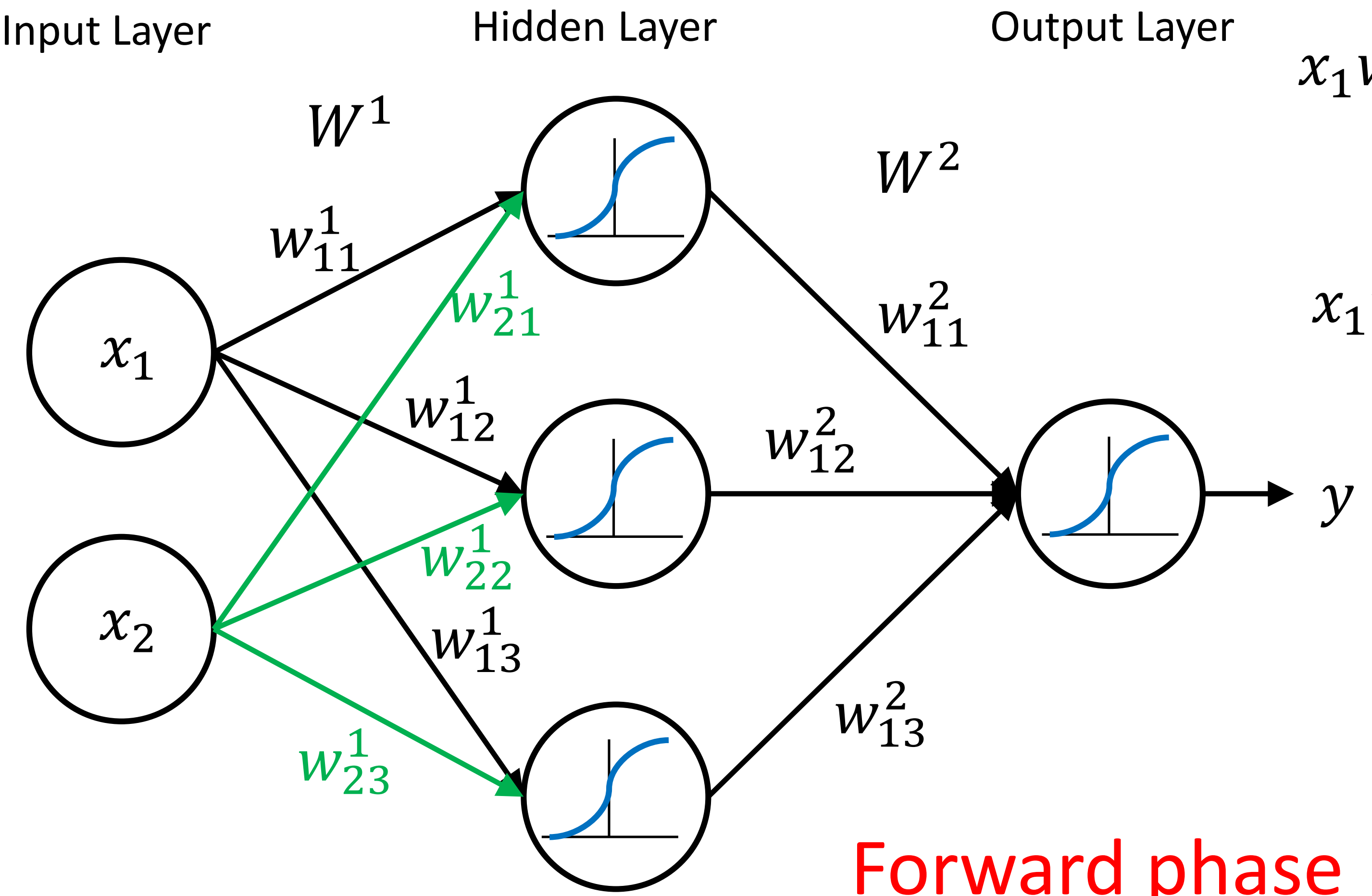
$$x_1 w_{11}^1 + x_2 w_{21}^1 = z_1^2$$
$$a_1^2 w_{11}^2 = z_1^3$$

$$x_1 w_{12}^1 + x_2 w_{22}^1 = z_2^2$$
$$a_2^2 w_{12}^2 = z_2^3$$



Neural Networks

$$x_1 w_{11}^1 + x_2 w_{21}^1 = z_1^2$$
$$a_1^2 w_{11}^2 = z_1^3$$



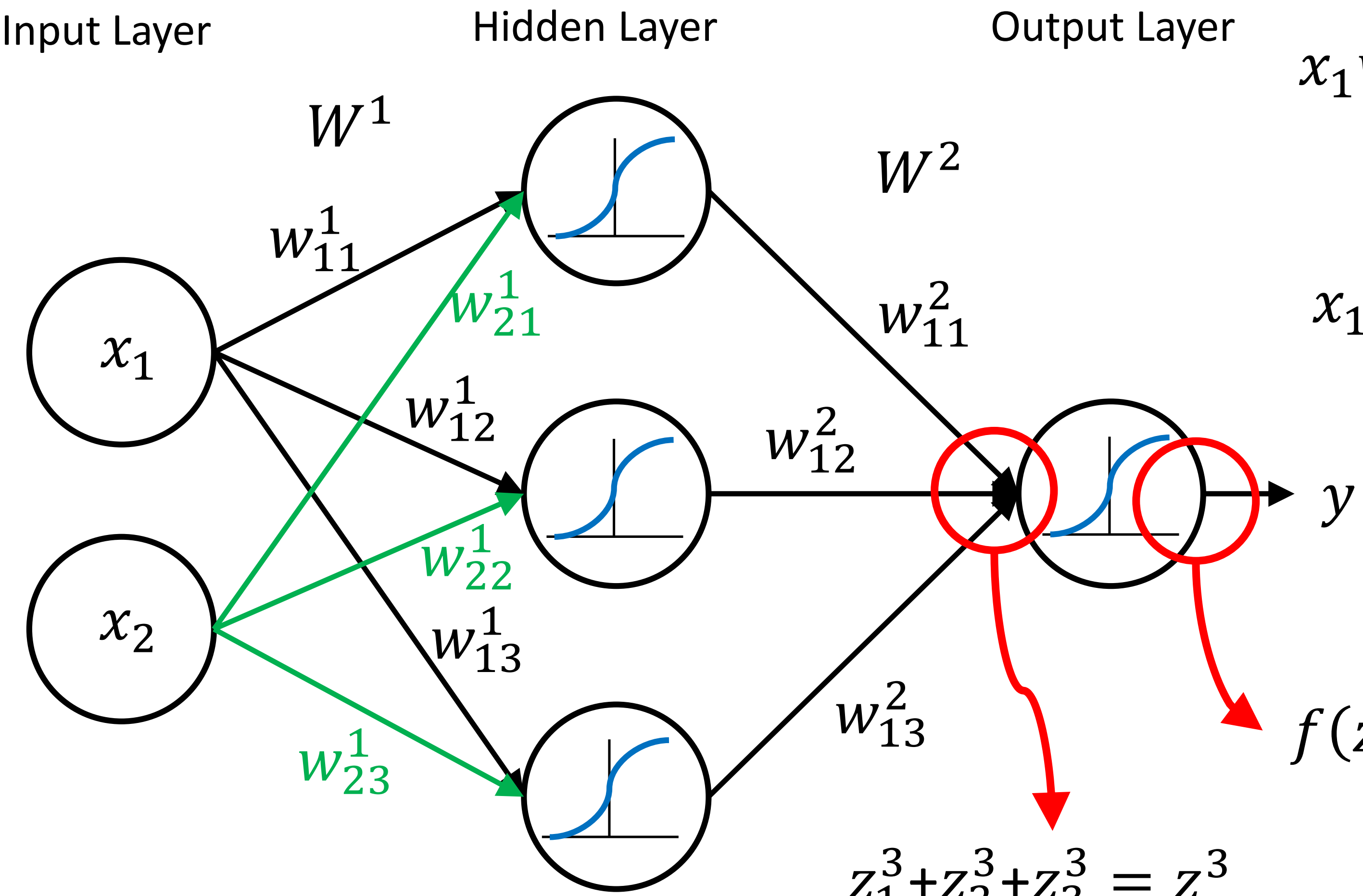
$$x_1 w_{12}^1 + x_2 w_{22}^1 = z_2^2$$
$$a_2^2 w_{12}^2 = z_2^3$$

$$x_1 w_{13}^1 + x_2 w_{23}^1 = z_3^2$$
$$a_3^2 w_{13}^2 = z_3^3$$

Forward phase

Neural Networks

$$x_1 w_{11}^1 + x_2 w_{21}^1 = z_1^2$$
$$a_1^2 w_{11}^2 = z_1^3$$



$$x_1 w_{12}^1 + x_2 w_{22}^1 = z_2^2$$
$$a_2^2 w_{12}^2 = z_2^3$$

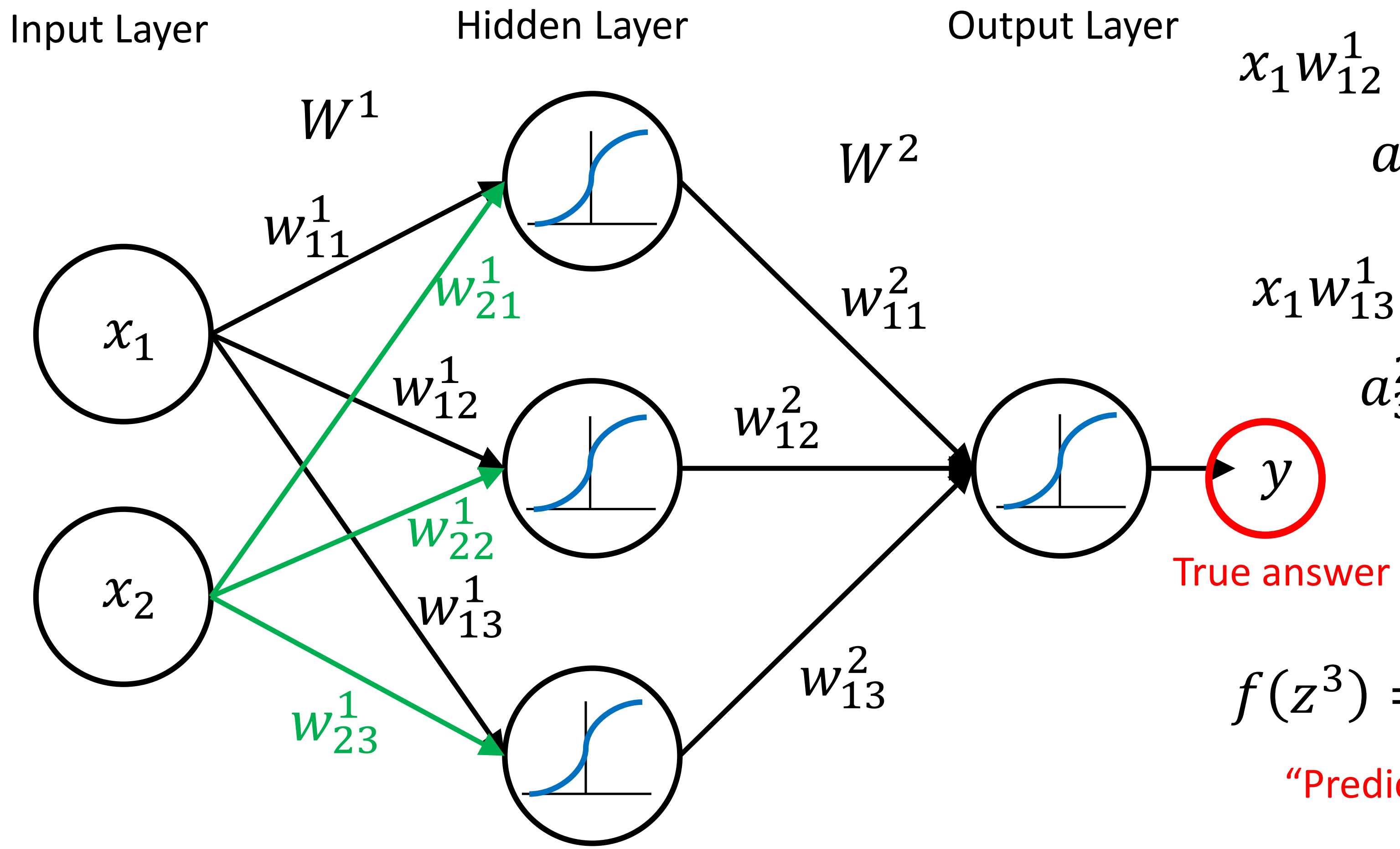
$$x_1 w_{13}^1 + x_2 w_{23}^1 = z_3^2$$
$$a_3^2 w_{13}^2 = z_3^3$$

$$f(z^3) = \hat{y}$$

$$z_1^3 + z_2^3 + z_3^3 = z^3$$

Neural Networks

$$x_1 w_{11}^1 + x_2 w_{21}^1 = z_1^2$$
$$a_1^2 w_{11}^2 = z_1^3$$



$$x_1 w_{12}^1 + x_2 w_{22}^1 = z_2^2$$
$$a_2^2 w_{12}^2 = z_2^3$$

$$x_1 w_{13}^1 + x_2 w_{23}^1 = z_3^2$$
$$a_3^2 w_{13}^2 = z_3^3$$

$$f(z^3) = \hat{y}$$

"Predicted" answer

Neural Networks

Input Layer

Hidden Layer

Output Layer

$$x_1 w_{11}^1 + x_2 w_{21}^1 = z_1^2 \qquad f(z_1^2) = a_1^2 \qquad a_1^2 w_{11}^2 = z_1^3$$

y

x_1

$$x_1 w_{12}^1 + x_2 w_{22}^1 = z_2^2 \qquad f(z_2^2) = a_2^2 \qquad a_2^2 w_{12}^2 = z_2^3$$

$$z_1^3 + z_2^3 + z_3^3$$

$$f(z^3) = \hat{y}$$

x_2

$$x_1 w_{13}^1 + x_2 w_{23}^1 = z_3^2 \qquad f(z_3^2) = a_3^2 \qquad a_3^2 w_{13}^2 = z_3^3$$

Wait…

Matrix Multiplication

$$\overset{\textbf{A}}{\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}} \overset{\textbf{B}}{\begin{pmatrix} 6 & 3 \\ 5 & 2 \\ 4 & 1 \end{pmatrix}} = \overset{\textbf{A} * \textbf{B}}{\begin{pmatrix} 1*6+2*5+3*4 & 1*3+2*2+3*1 \\ 4*6+5*5+6*4 & 4*3+5*2+6*1 \end{pmatrix}}$$

n by m × m by p = n by p

Neural Networks

Input Layer

Hidden Layer

Output Layer

$$x_1 w_{11}^1 + x_2 w_{21}^1 = z_1^2 \qquad f(z_1^2) = a_1^2 \qquad a_1^2 w_{11}^2 = z_1^3$$

y

x_1

$$x_1 w_{12}^1 + x_2 w_{22}^1 = z_2^2 \qquad f(z_2^2) = a_2^2 \qquad a_2^2 w_{12}^2 = z_2^3$$

$$z_1^3 + z_2^3 + z_3^3$$

$$f(z^3) = \hat{y}$$

x_2

$$x_1 w_{13}^1 + x_2 w_{23}^1 = z_3^2 \qquad f(z_3^2) = a_3^2 \qquad a_3^2 w_{13}^2 = z_3^3$$

y

Neural Networks

$$x_1 w_{11}^1 + x_2 w_{21}^1 = z_1^2$$
$$x_1 w_{12}^1 + x_2 w_{22}^1 = z_2^2$$
$$x_1 w_{13}^1 + x_2 w_{23}^1 = z_3^2$$

$$f(z_1^2) = a_1^2$$
$$f(z_2^2) = a_2^2$$
$$f(z_3^2) = a_3^2$$

$$a_1^2 w_{11}^2 = z_1^3$$
$$a_2^2 w_{12}^2 = z_2^3$$
$$a_3^2 w_{13}^2 = z_3^3$$

$$z_1^3 + z_2^3 + z_3^3$$

$$f(z^3) = \hat{y}$$

$$[x_1 \quad x_2] \begin{bmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \end{bmatrix}$$

$$=$$

$$[z_1^2 \quad z_2^2 \quad z_3^2]$$

$$XW^1 = z^2$$

$$f(Z^2) = a^2$$

$$a^2 W^2 = z^3$$

y

True answer

Neural Networks

$$x_1 w_{11}^1 + x_2 w_{21}^1 = z_1^2$$
$$x_1 w_{12}^1 + x_2 w_{22}^1 = z_2^2$$
$$x_1 w_{13}^1 + x_2 w_{23}^1 = z_3^2$$

$$f(z_1^2) = a_1^2$$
$$f(z_2^2) = a_2^2$$
$$f(z_3^2) = a_3^2$$

$$a_1^2 w_{11}^2 = z_1^3$$
$$a_2^2 w_{12}^2 = z_2^3$$
$$a_3^2 w_{13}^2 = z_3^3$$

$$z_1^3 + z_2^3 + z_3^3$$

$$f(z^3) = \hat{y}$$

"Predicted" answer

$$[x_1 \quad x_2] \begin{bmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \end{bmatrix}$$

$$\shortparallel$$

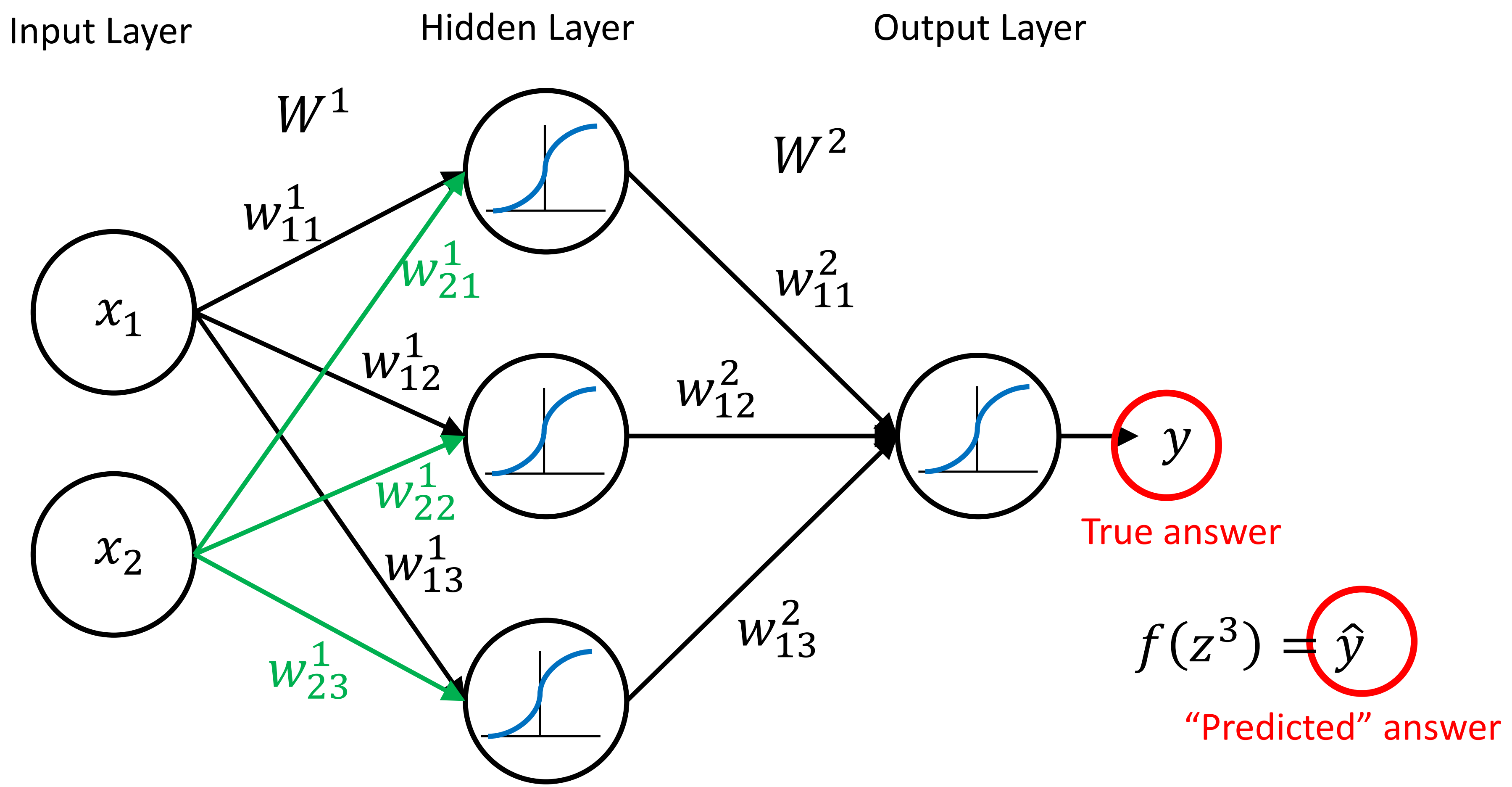
$$[z_1^2 \quad z_2^2 \quad z_3^2]$$

$$XW^1 = z^2$$

$$f(Z^2) = a^2 \qquad a^2 W^2 = z^3$$

Forward phase

Neural Networks



Training Errors/Loss

Difference between y and $\hat{y}$.

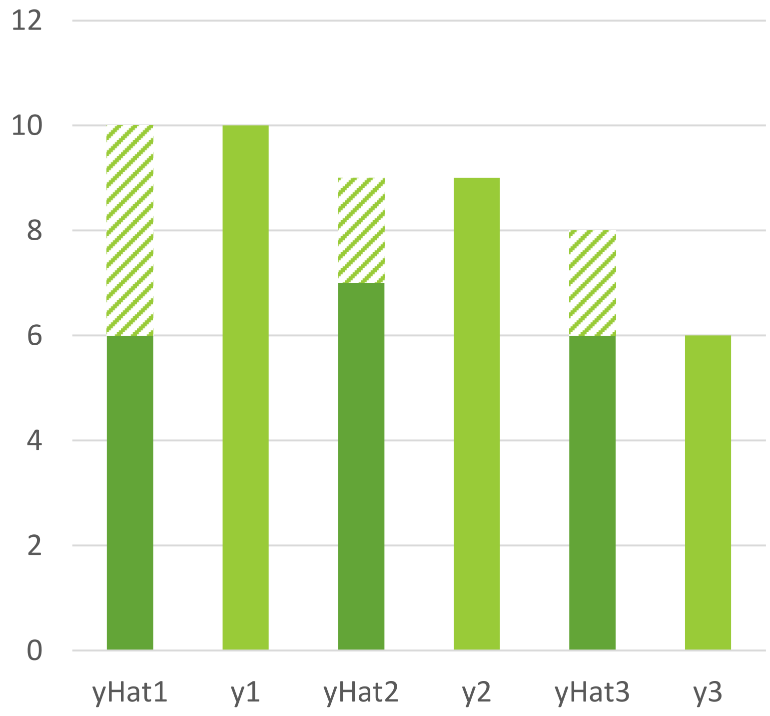
That is, the difference between the true answer and the predicted answer.

- We need a way to "quantify" how big the error $\boldsymbol{e}$ is.
- A Cost Function $\boldsymbol{C}$ is used to quantify our errors.
- One simple cost function is *mean square error:*

$$C = \frac{1}{m} \sum_{j} \left(\hat{y}_j - y_j \right)^2$$

- where j is the j^{th} true answer y and the j^{th} predicted answer $\hat{y}$.

Training Errors/Loss



Training error = $$\frac{1}{3}\left((\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + (\hat{y}_3 - y_3)^2\right)$$

$$= \frac{1}{3}\left((6 - 10)^2 + (7 - 9)^2 + (9 - 6)^2\right)$$

$$= \frac{1}{3}\left((-4)^2 + (-2)^2 + (3)^2\right)$$

$$= \frac{1}{3}(16 + 4 + 9)$$

$$= \frac{29}{3}$$

y

True answer

Neural Networks

$$x_1 w_{11}^1 + x_2 w_{21}^1 = z_1^2$$
$$x_1 w_{12}^1 + x_2 w_{22}^1 = z_2^2$$
$$x_1 w_{13}^1 + x_2 w_{23}^1 = z_3^2$$

$$f(z_1^2) = a_1^2$$
$$f(z_2^2) = a_2^2$$
$$f(z_3^2) = a_3^2$$

$$a_1^2 w_{11}^2 = z_1^3$$
$$a_2^2 w_{12}^2 = z_2^3$$
$$a_3^2 w_{13}^2 = z_3^3$$

$$z_1^3 + z_2^3 + z_3^3$$

$$f(z^3) = \hat{y}$$

"Predicted" answer

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \end{bmatrix}$$

$$\shortparallel$$

$$\begin{bmatrix} z_1^2 & z_2^2 & z_3^2 \end{bmatrix}$$

$$XW^1 = z^2$$

$$f(Z^2) = a^2 \qquad a^2 W^2 = z^3$$

Forward phase

Neural Networks

$$XW^1 = z^2$$

$$f(Z^2) = a^2$$

$$a^2W^2 = z^3$$

$$f(z^3) = \hat{y}$$

$$J = \frac{1}{m}\sum(\hat{y} - y)^2$$

Minimize this!!

Neural Networks

$$XW^1 = z^2$$

$$f(Z^2) = a^2$$

$$a^2W^2 = z^3$$

$$f(z^3) = \hat{y}$$

$$J = \frac{1}{m}\sum(\hat{y} - y)^2$$

Minimize this!!

$$J = \frac{1}{m}\sum(\hat{y} - y)^2$$

$$J = \frac{1}{m}\sum(f(z^3) - y)^2$$

$$J = \frac{1}{m}\sum(f(a^2W^2) - y)^2$$

$$J = \frac{1}{m}\sum(f(f(Z^2)W^2) - y)^2$$

$$J = \frac{1}{m}\sum(f(f(XW^1)W^2) - y)^2$$

Neural Networks

$$XW^1 = z^2$$

$$f(Z^2) = a^2$$

$$a^2W^2 = z^3$$

$$f(z^3) = \hat{y}$$

$$J = \frac{1}{m}\sum(\hat{y} - y)^2$$

Minimize this!!

$$J = \frac{1}{m}\sum(\hat{y} - y)^2$$

$$J = \frac{1}{m}\sum(f(z^3) - y)^2$$

$$J = \frac{1}{m}\sum(f(a^2W^2) - y)^2$$

$$J = \frac{1}{m}\sum(f(f(Z^2)W^2) - y)^2$$

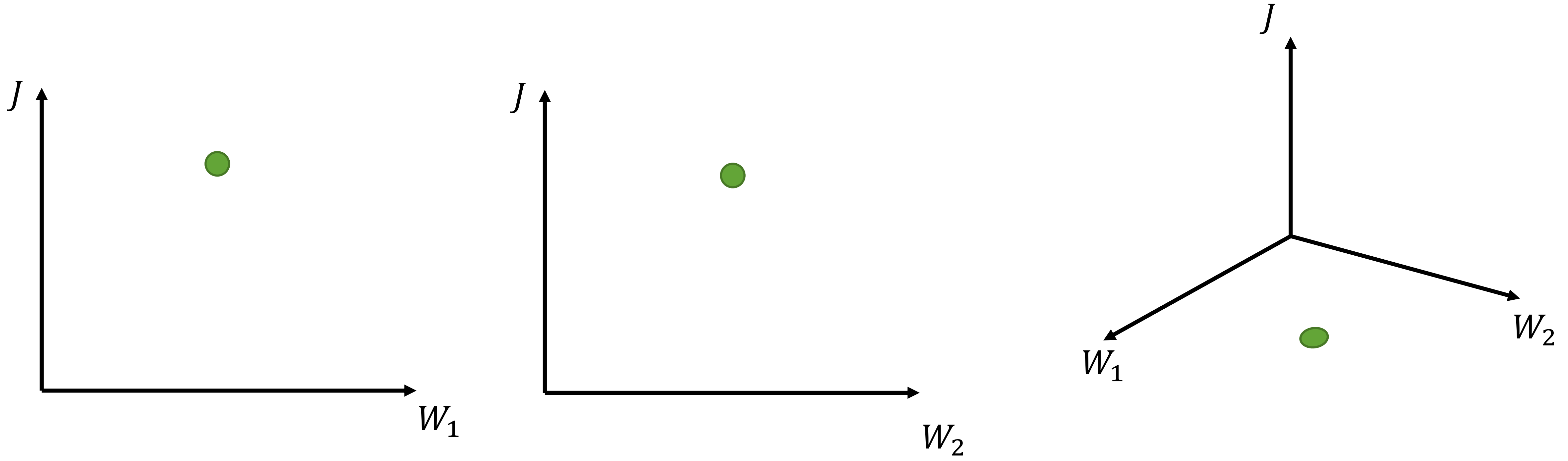
$$J = \frac{1}{m}\sum(f(f(XW^1)W^2) - y)^2$$

X and y fixed.

So the objective is to find the set of W_1 and W_2 that yield the smallest J, the error/loss.

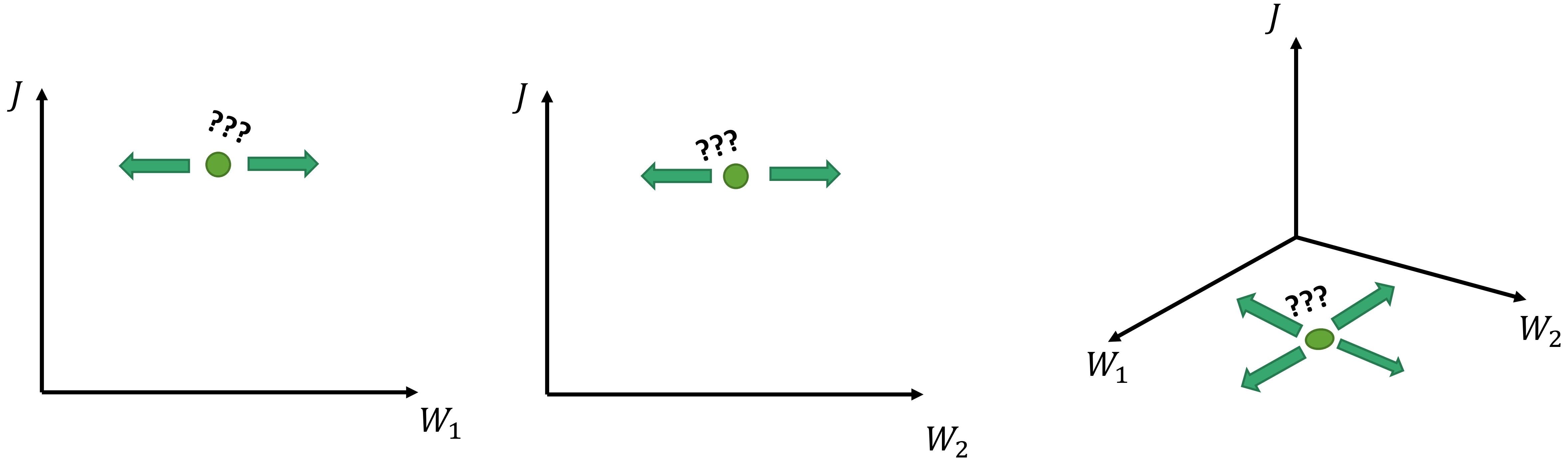
How do we minimize training error/loss??

The objective is to find the set of W_1 and W_2 that yield the smallest J, the error/loss.



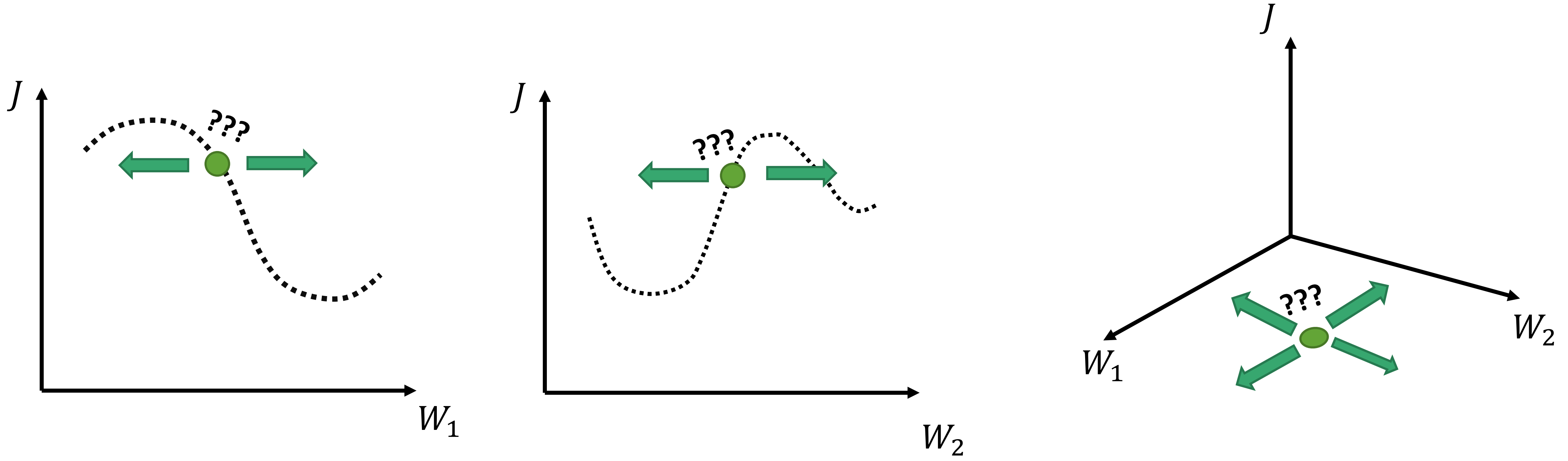
How do we minimize training error/loss??

The objective is to find the set of W_1 and W_2 that yield the smallest J, the error/loss.



How do we minimize training error/loss??

The objective is to find the set of W_1 and W_2 that yield the smallest J , the error/loss.



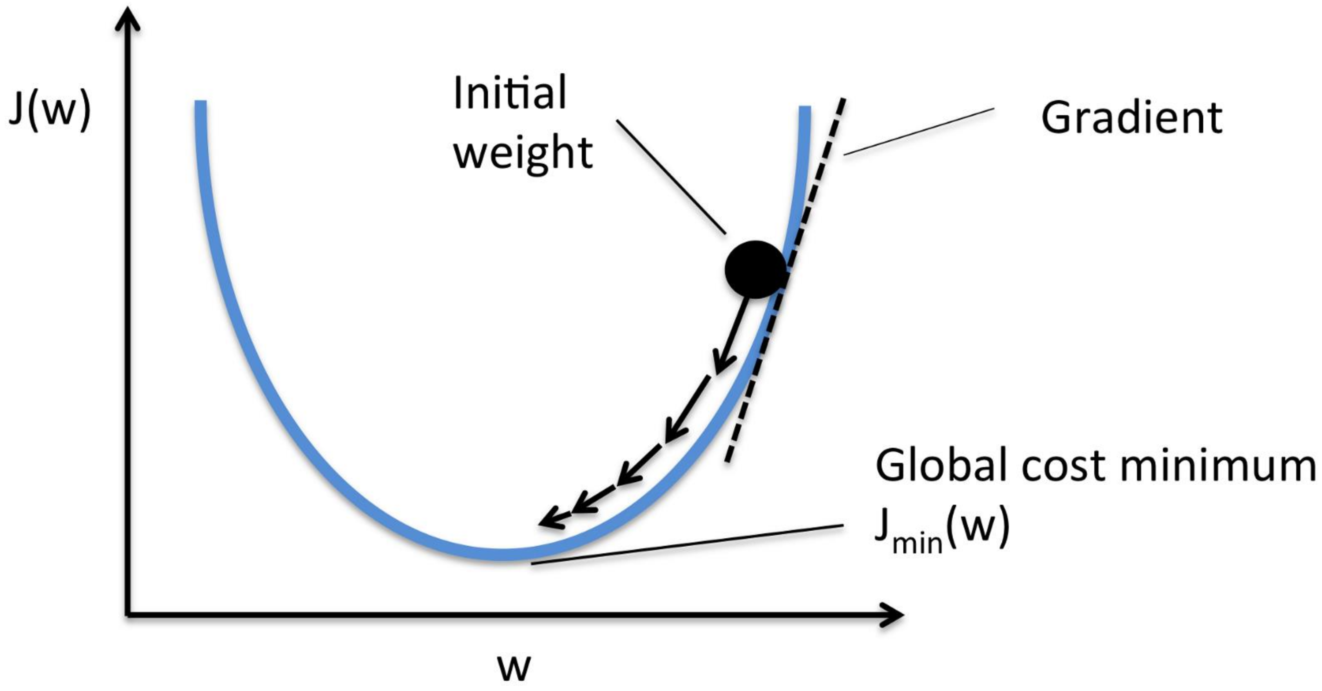
How do we Minimize training error/loss???

Gradient Descent

The goal of gradient descent is to minimize the cost function, i.e. error/loss.

Minimizing the cost function is viewed as a convex problem where there is only one minimum.

$$J = \frac{1}{m}\sum(f(f(XW^1)W^2) - y)^2$$



https://hackernoon.com/gradient-descent-aynk-7cbe95a778da

How do we Minimize training error/loss???

$$J = \frac{1}{m}\sum(f(f(XW^1)W^2) - y)^2$$

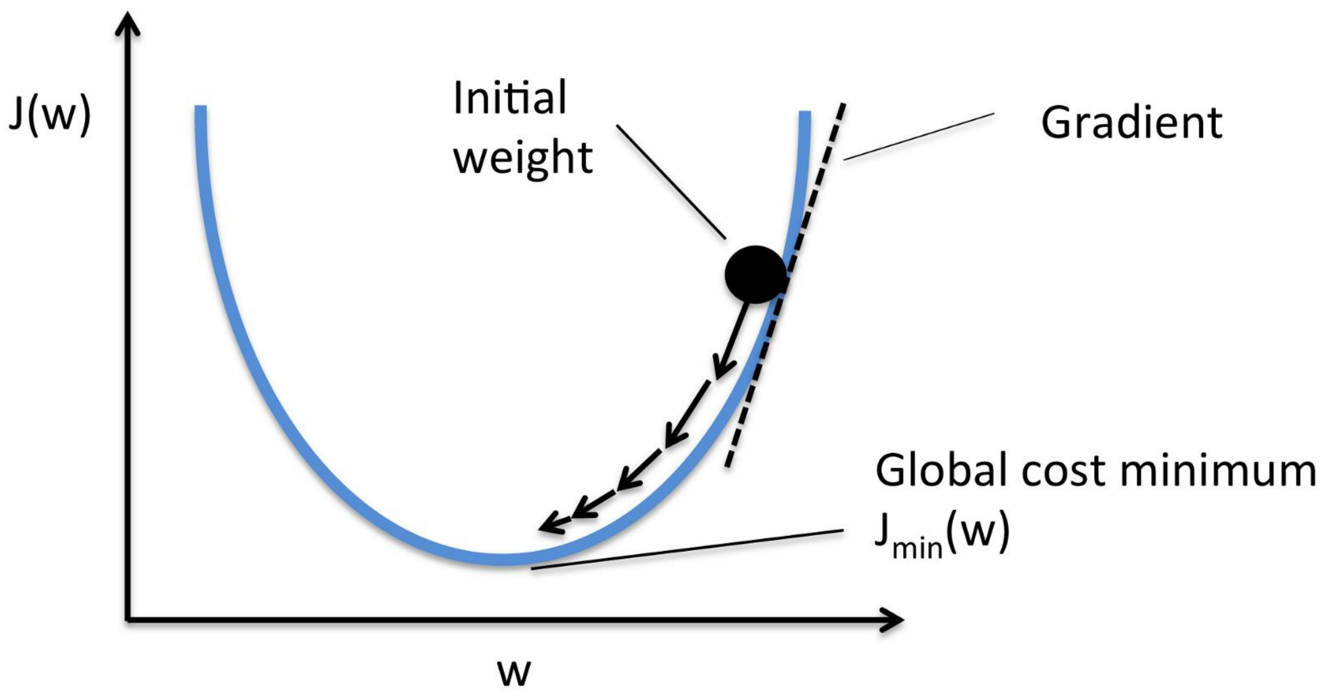
Gradient Descent

https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent

Learning rates

Batch size

- Batch
- Mini batch
- Stochastic GD
- https://hackernoon.com/gradient-descent-aynk-7cbe95a778da



https://hackernoon.com/gradient-descent-aynk-7cbe95a778da

Partial Derivatives

$$J = \frac{1}{m}\sum(f(f(XW^1)W^2) - y)^2$$

Objective: $\min_{W^1,W^2} J(W^1, W^2)$

Update rule: $W^1 \coloneqq W^1 - \alpha \frac{\partial}{\partial W^1} J(W^1, W^2)$

$$W^2 \coloneqq W^2 - \alpha \frac{\partial}{\partial W^2} J(W^1, W^2)$$

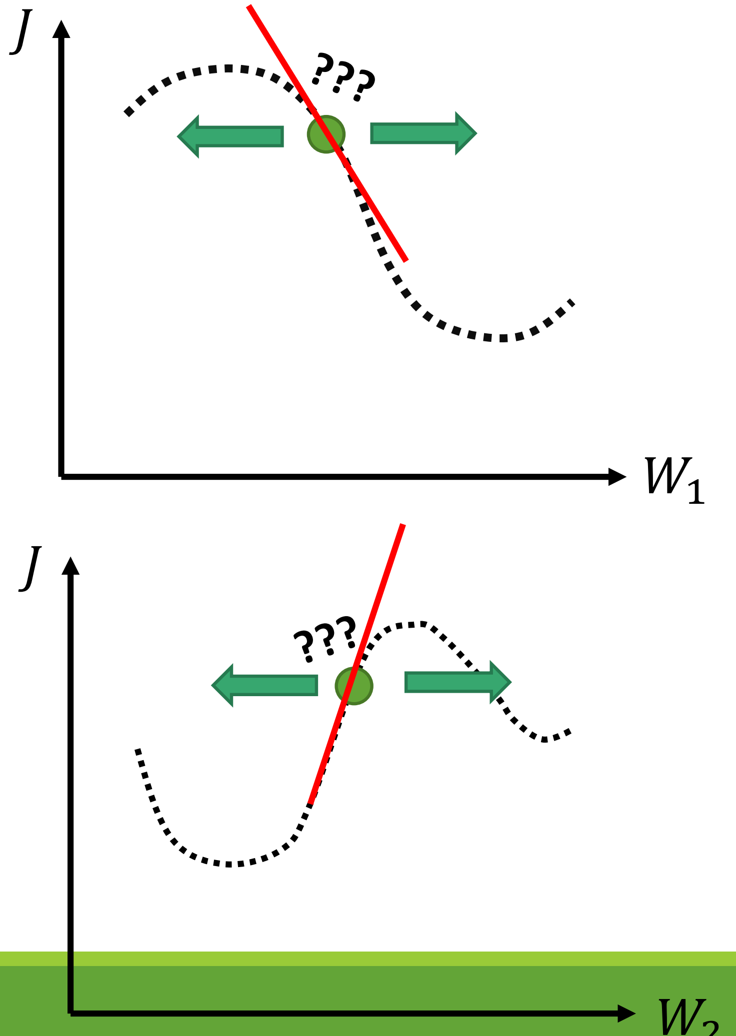
Partial Derivatives

$$J = \frac{1}{m}\sum(f(f(XW^1)W^2) - y)^2$$

Objective: $\min_{W^1,W^2} J(W^1, W^2)$

Update rule: $W^1 := W^1 - \alpha \frac{\partial}{\partial W^1} J(W^1, W^2)$

$W^2 := W^2 - \alpha \frac{\partial}{\partial W^2} J(W^1, W^2)$



Partial Derivatives

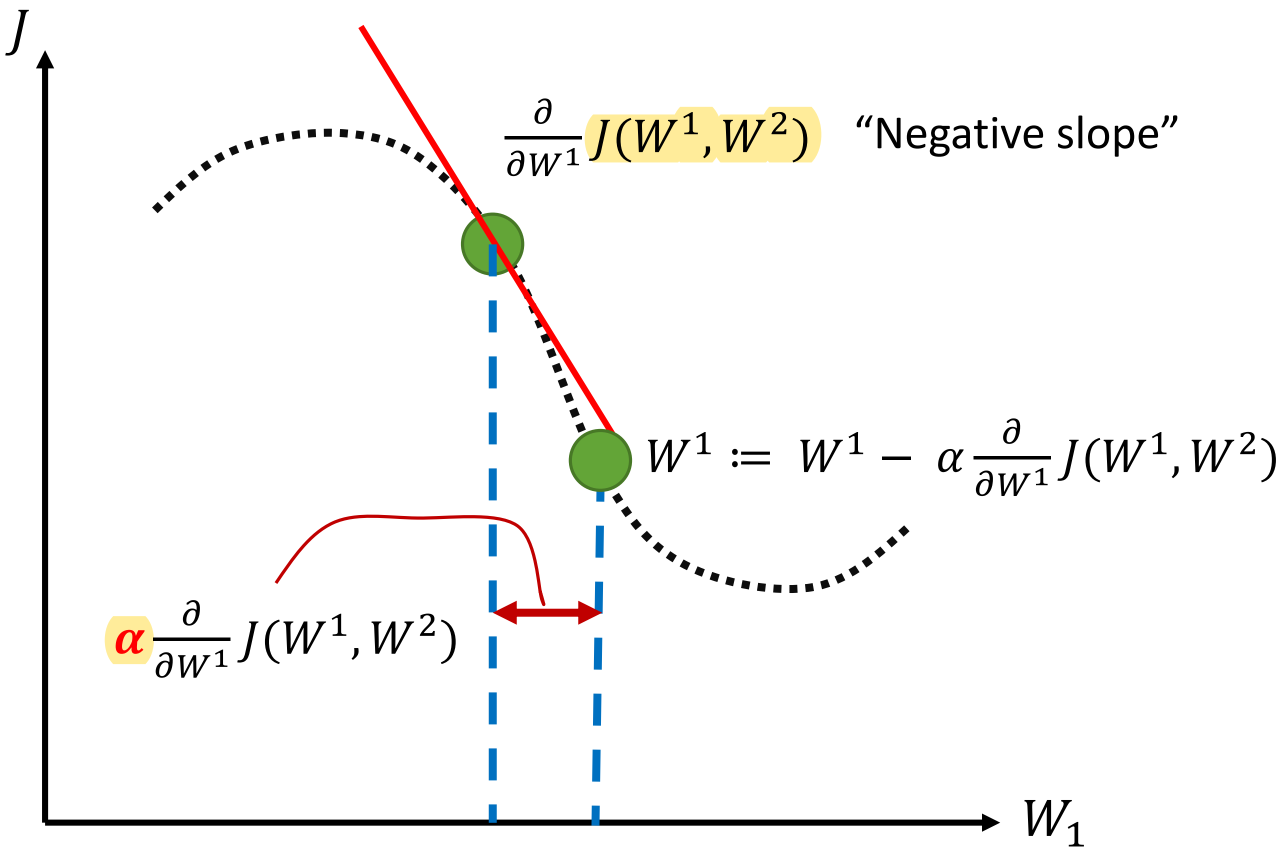
$$J = \frac{1}{m}\sum (f(f(XW^1)W^2) - y)^2$$

Objective: $\min_{W^1,W^2} J(W^1, W^2)$

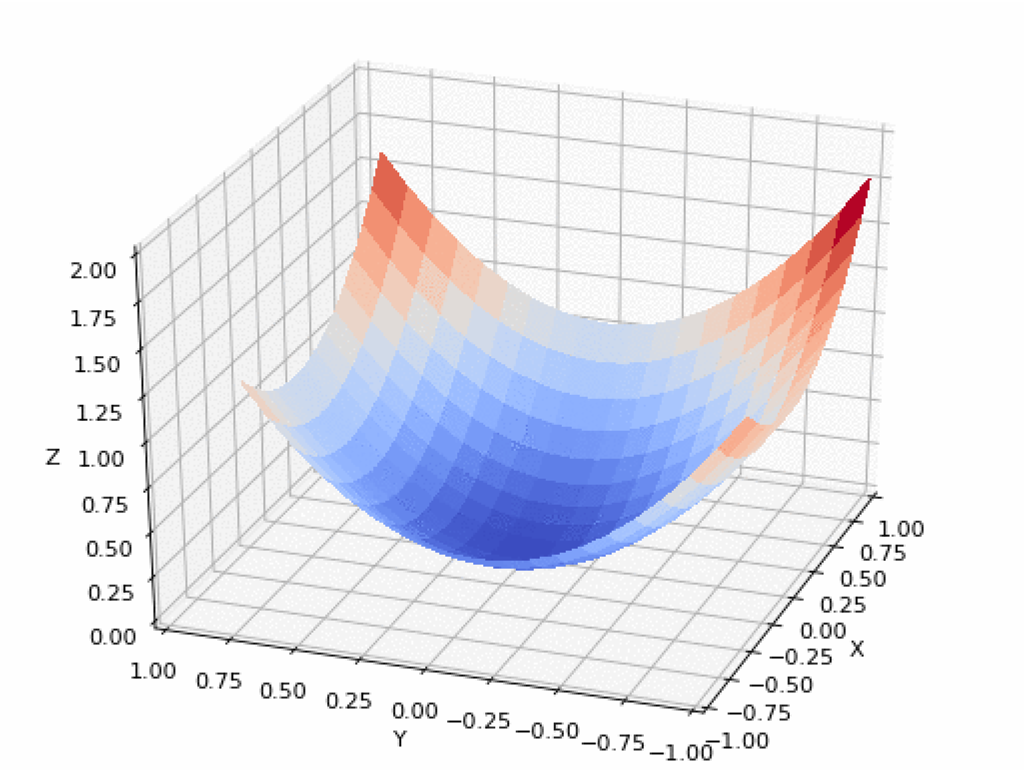
Update rule: $W^1 := W^1 - \alpha \frac{\partial}{\partial W^1} J(W^1, W^2)$

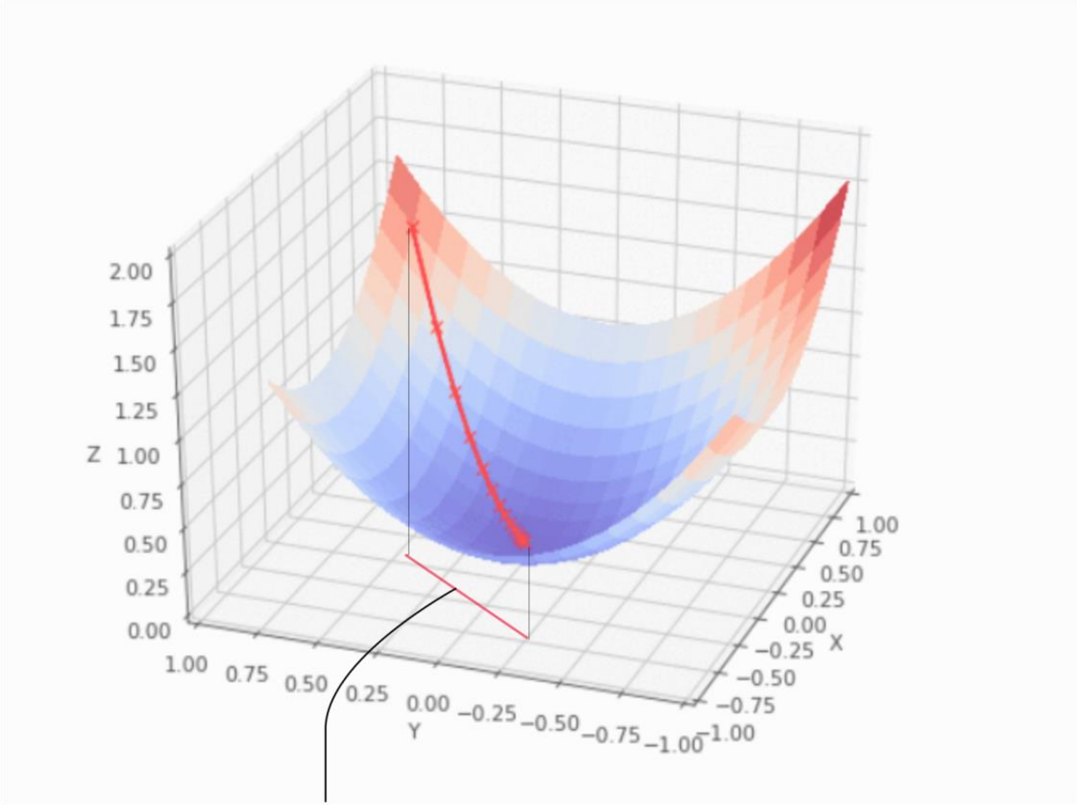
$W^2 := W^2 - \alpha \frac{\partial}{\partial W^2} J(W^1, W^2)$

α is learning rate.



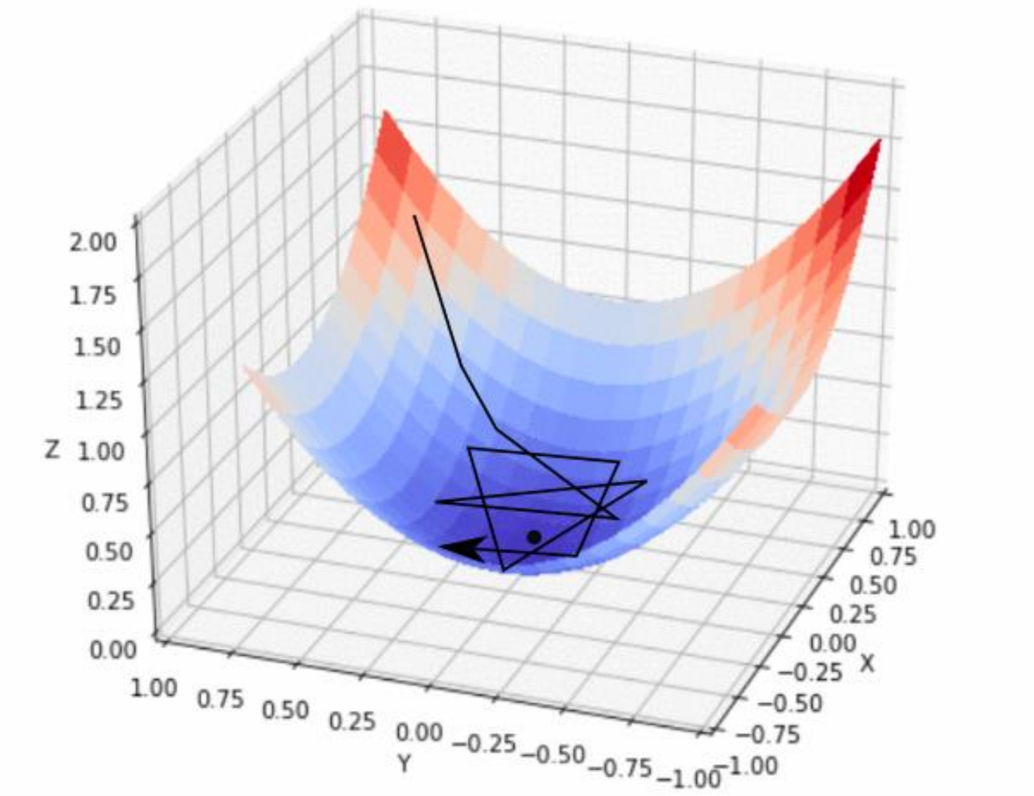
Gradient Descent





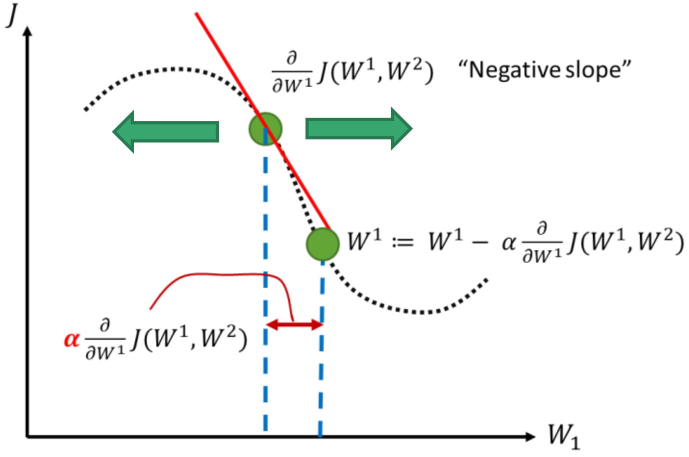
Real Trajectory of G.D.

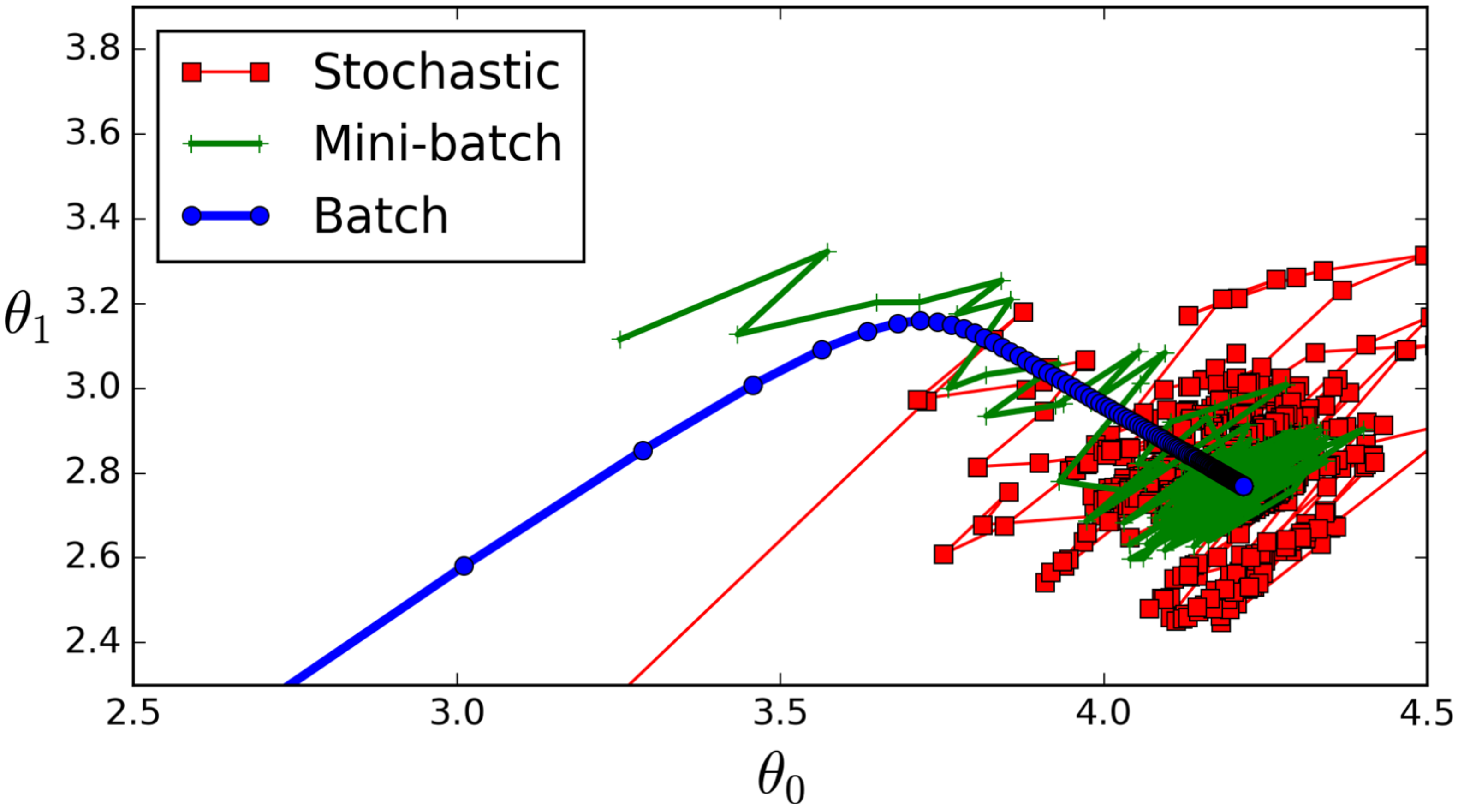
large learning rates…



Batch size

- Batch size
  - Batch
  - Mini batch
  - Stochastic GD
  - https://hackernoon.com/gradient-descent-aynk-7cbe95a778da





Gradient Descent

https://jed-ai.github.io/py1_gd_animation/

https://xavierbourretsicotte.github.io/animation_ridge.html

https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

Machine Learning & Text

Word as feature/dimension.

Texts are represented by “vectors”.

I love you.

I really like you.

I have feelings for you.

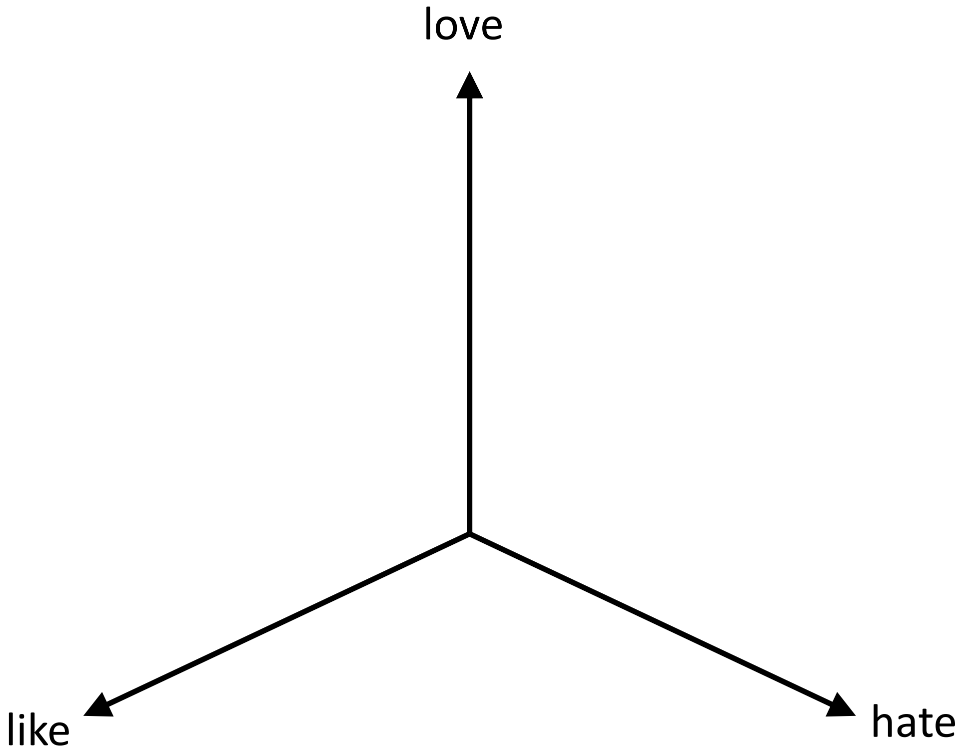
I really hate you.

feelings	for	hate	have	I	like	love	really	you
0	0	0	0	1	0	1	0	1
0	0	0	0	1	1	0	1	1
1	1	0	1	1	0	0	0	1
0	0	1	0	1	0	0	0	1

https://colab.research.google.com/drive/1TGaZosIeIGxtpLv-6OYS63ACl_qKiuJd

https://colab.research.google.com/drive/1f7FYwe-HZNnXD7DPx1bLjpGIaBOde5nx

Machine Learning & Text

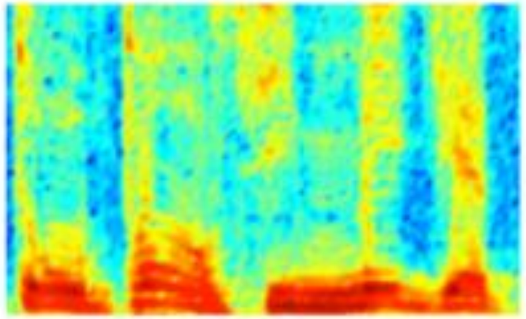


I don’t like him. I hate him.

I don’t hate him. I like him.

Word Embedding

AUDIO



Audio Spectrogram

DENSE

IMAGES

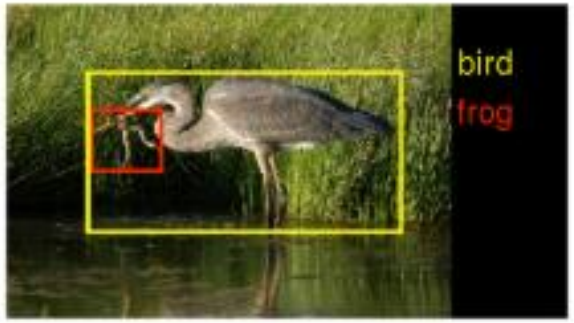


Image pixels

DENSE

TEXT

0	0	0	0.2	0	0.7	0	0	0	...	...

Word, context, or document vectors

SPARSE

Image source: https://www.tensorflow.org/tutorials/representation/word2vec

Word Embedding: Word2Vec

Representing a word as a vector.

Latent Semantic Analysis

GloVe

Word2Vec

- CBOW: Continuous Bag-of-Words
- **Skip-Gram**

Word2Vec, Skip-Gram

the quick brown $\boxed{\text{fox}}$ jumps over the lazy dog

Window size = 2

Window size = 2

center word

Word2Vec, Skip-Gram

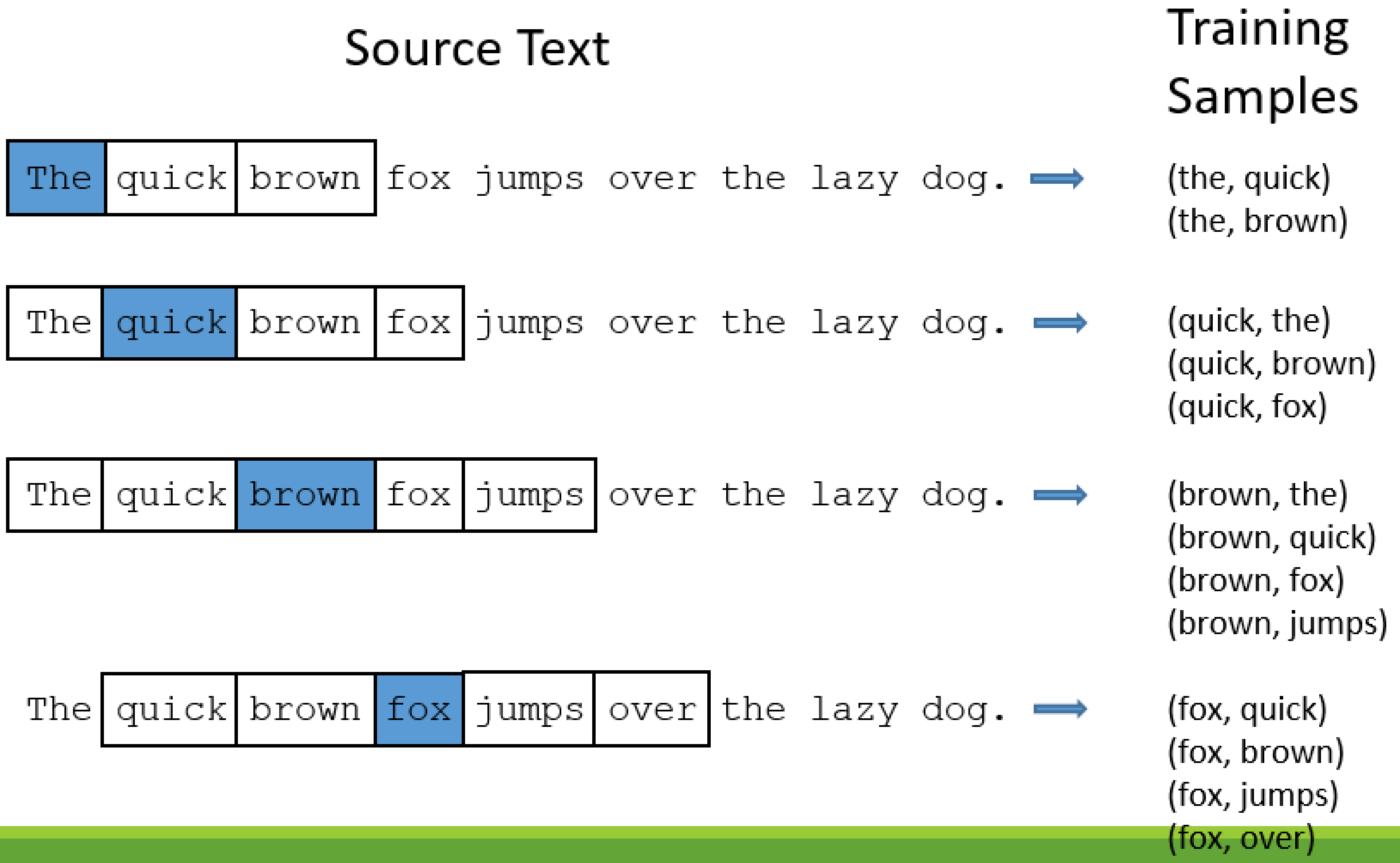


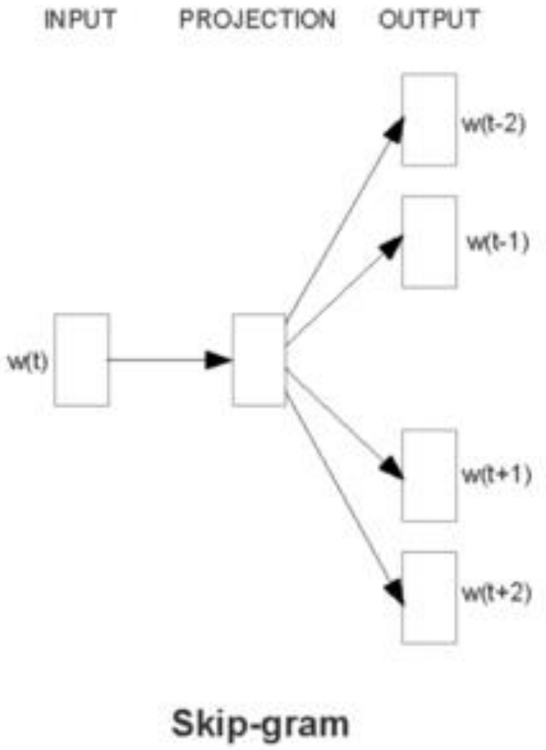
Image source: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word2Vec, Skip-Gram

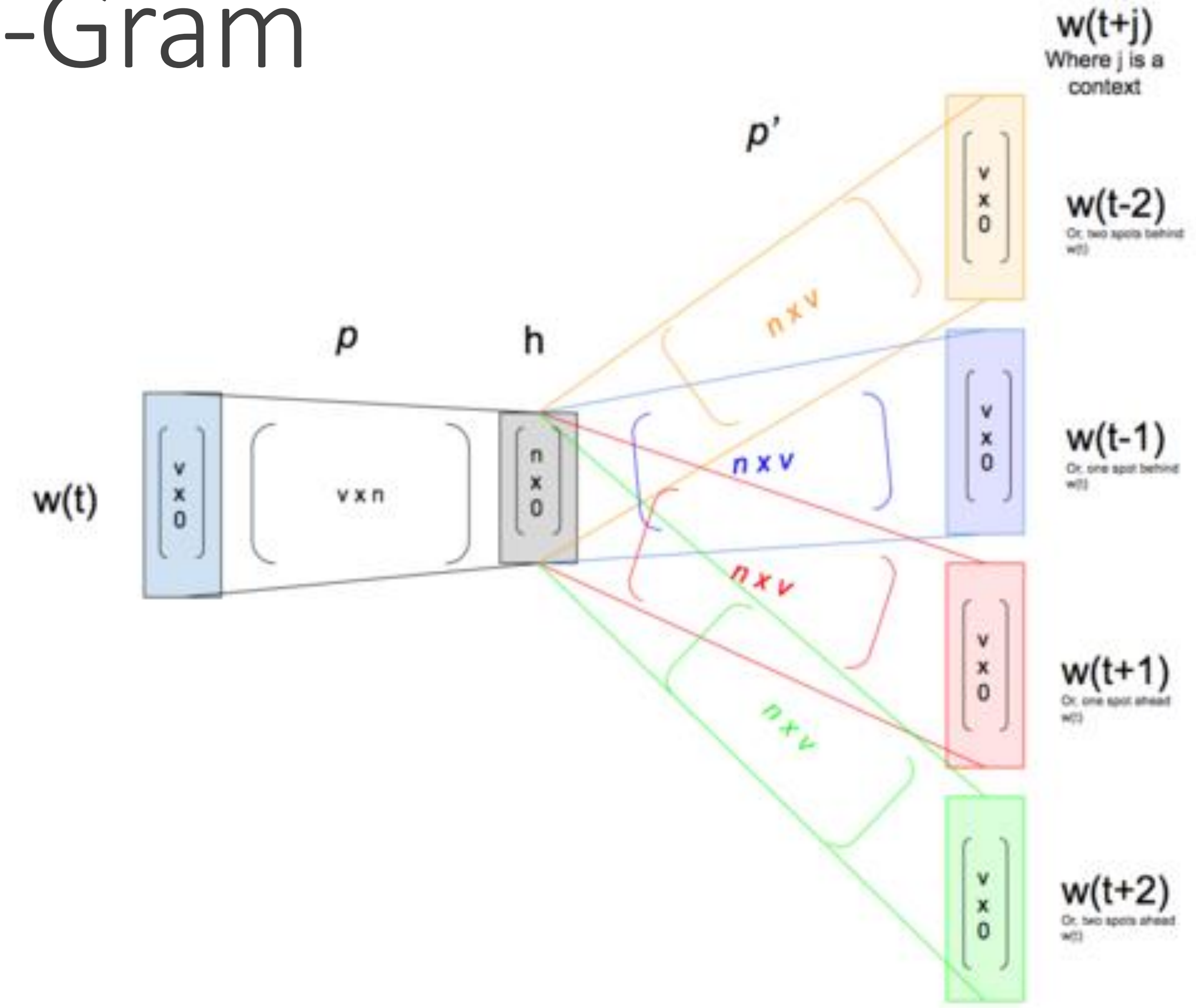
Source Text	Training Samples
The quick brown fox jumps over the lazy dog. ⟹ W(t) W(t+1) W(t+2)	(the, quick) (the, brown)
The **quick** brown fox jumps over the lazy dog. ⟹ W(t−1) W(t) W(t+1) W(t+2)	(quick, the) (quick, brown) (quick, fox)
The quick **brown** fox jumps over the lazy dog. ⟹ W(t−2) W(t−1) W(t) W(t+1) W(t+2)	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown **fox** jumps over the lazy dog. ⟹ W(t−2) W(t−1) W(t) W(t+1) W(t+2)	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

Image source: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word2Vec, Skip-Gram



Original diagram from Mikolov et al (2013)



Extended diagram identifying matrix dimensions

Image source: https://medium.com/district-data-labs/forward-propagation-building-a-skip-gram-net-from-the-ground-up-9578814b221

Neural Networks!! Yes, Again!!

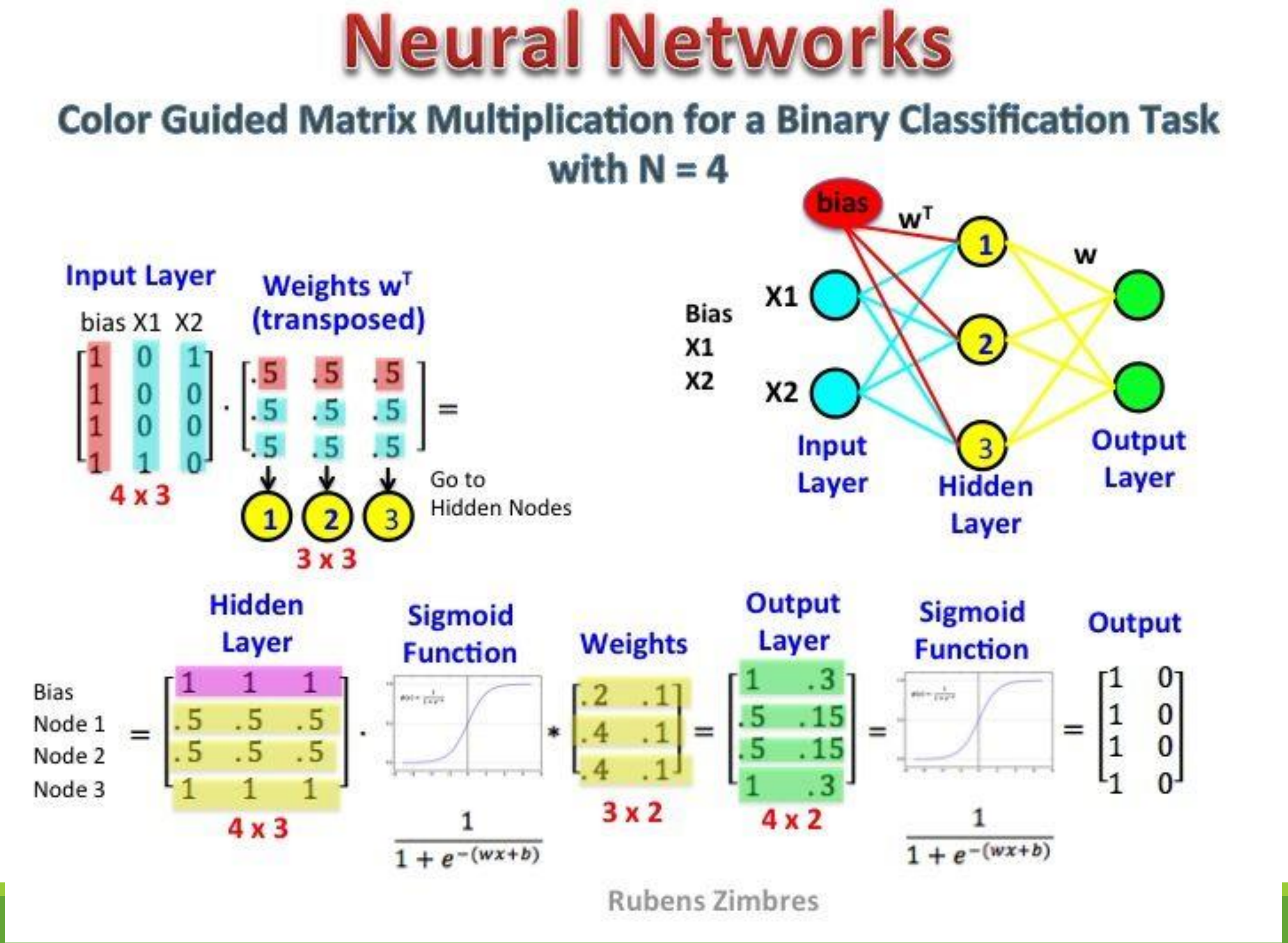
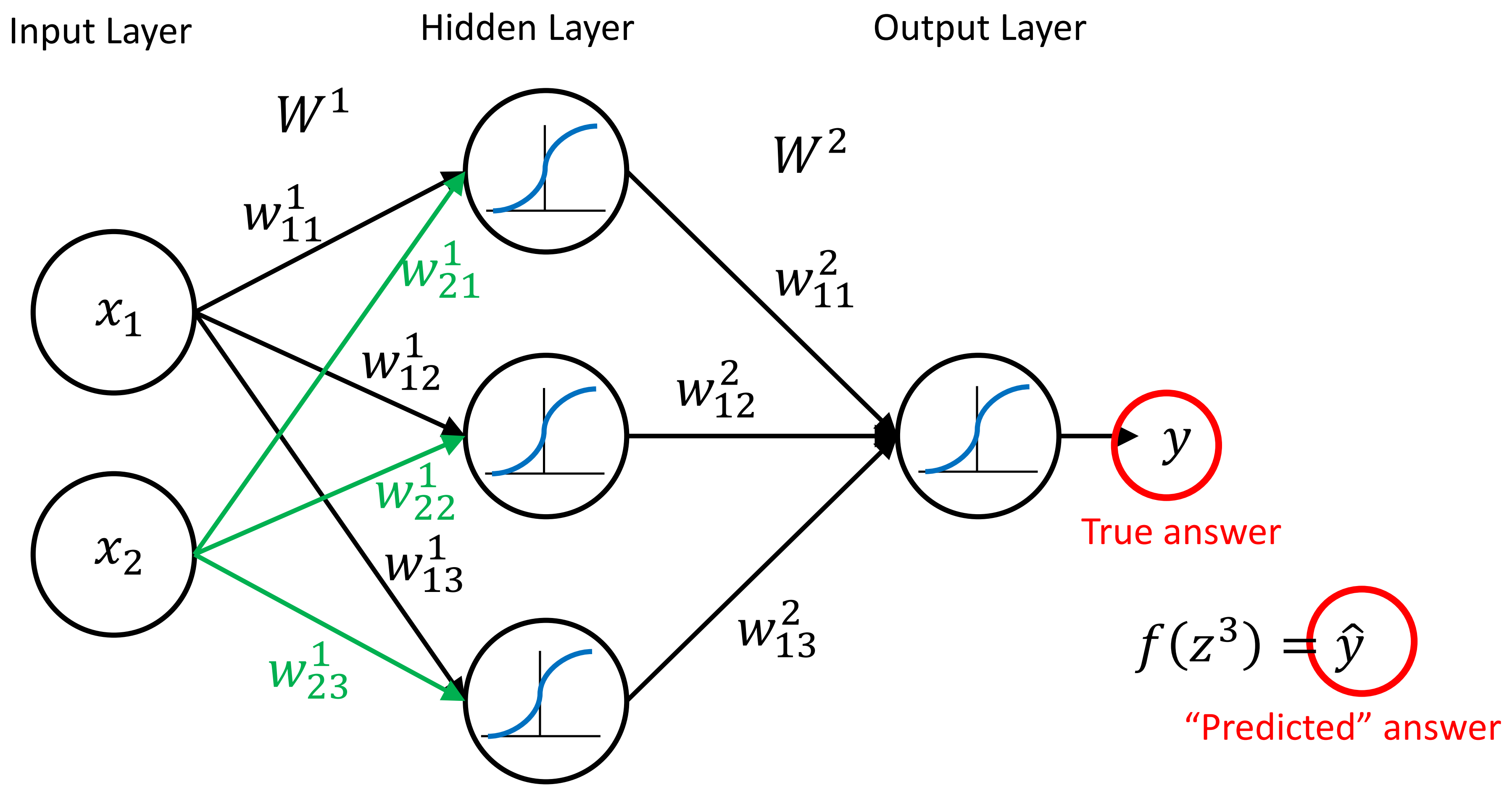
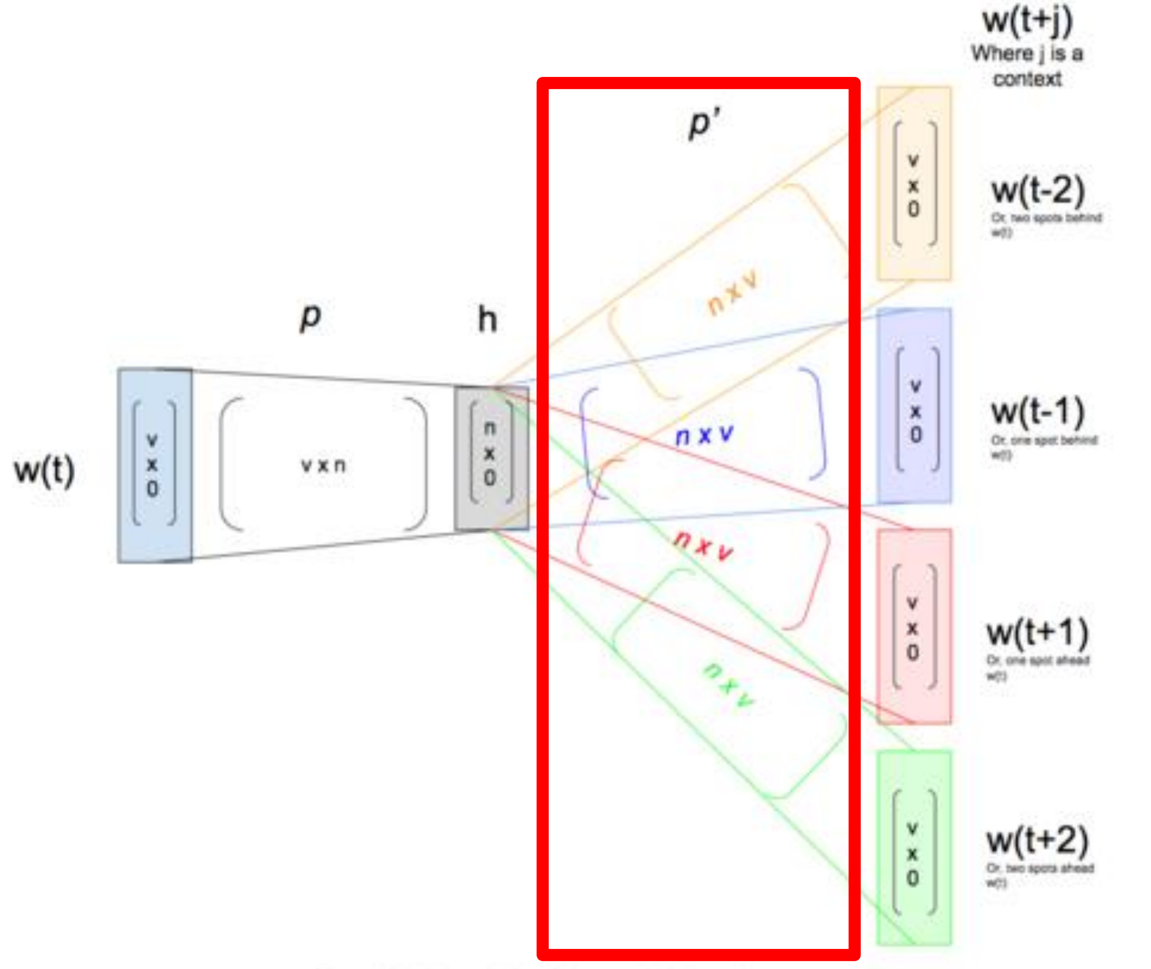


Image source: https://www.datasciencecentral.com/profiles/blogs/matrix-multiplication-in-neural-networks

Neural Networks

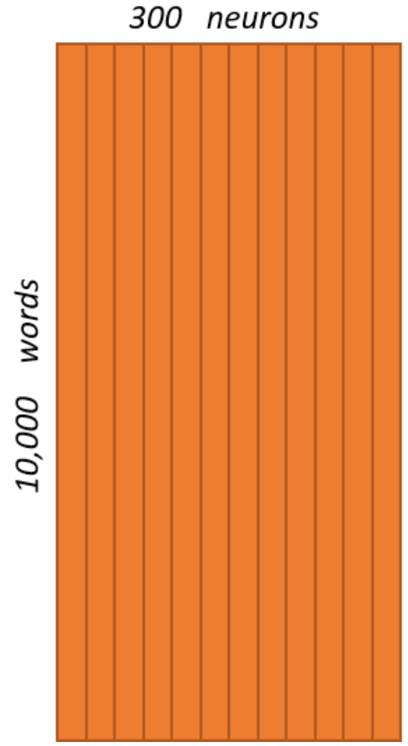


Word2Vec, Skip-Gram



Extended diagram identifying matrix dimensions

Hidden Layer
Weight Matrix



Word Vector
Lookup Table!

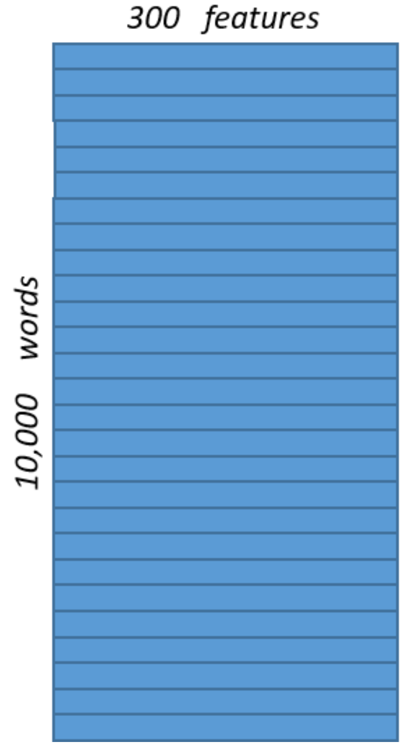


Image source: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word2Vec, Skip-gram

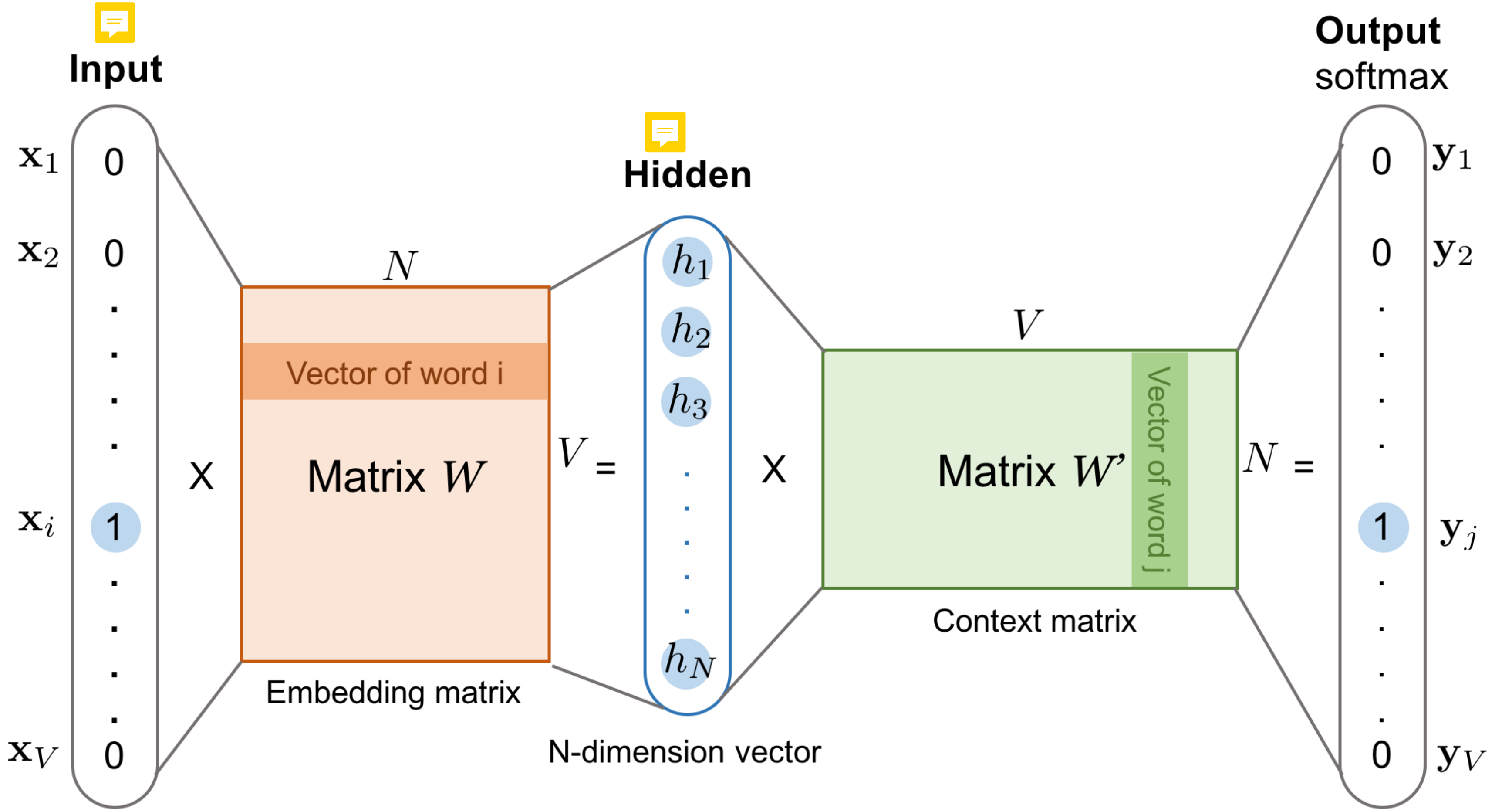


Image source: https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

Word2Vec, Skip-gram

https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d

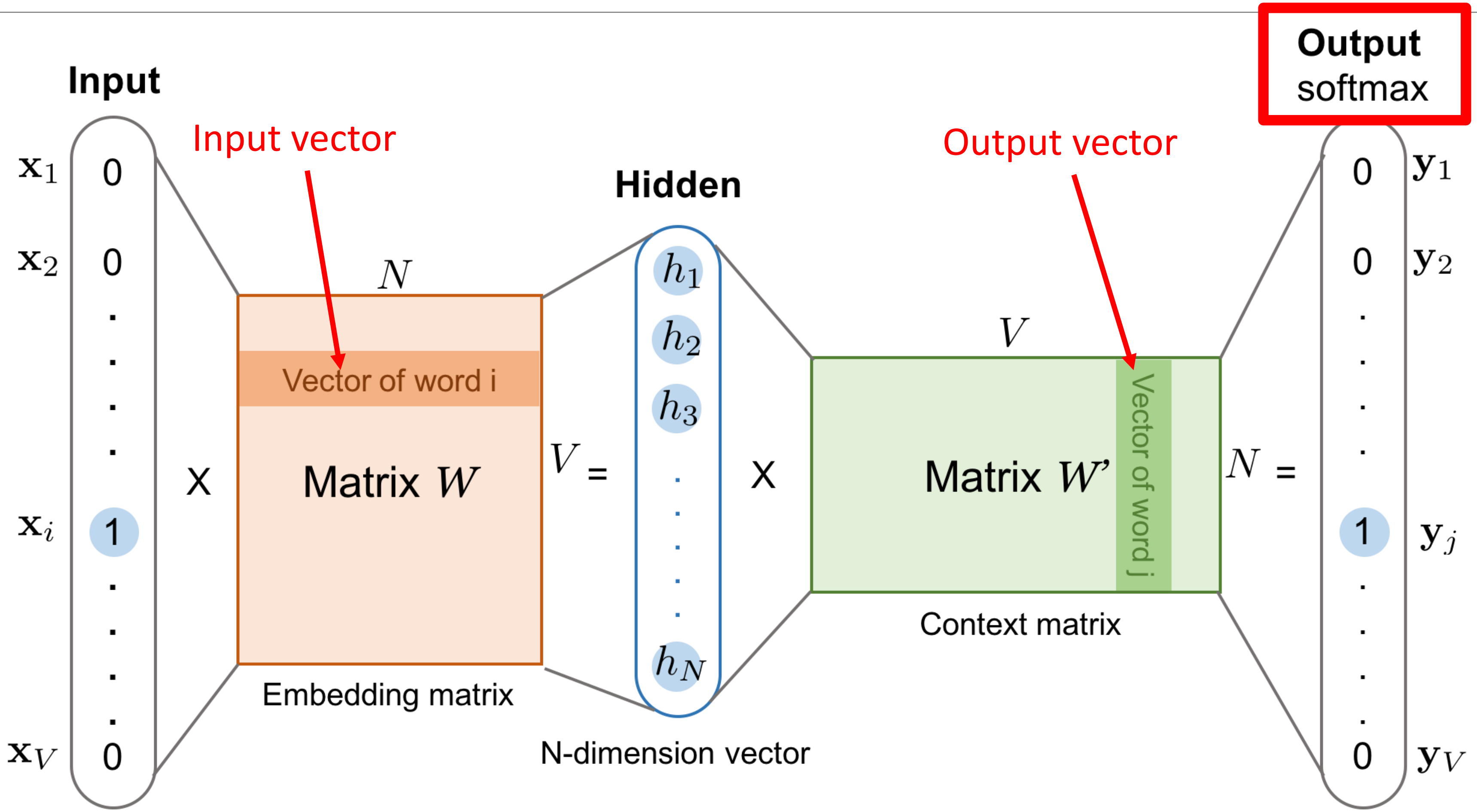


Image source: https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

Resources

https://github.com/stephencwelch/Neural-Networks-Demystified

https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU

https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

https://dashee87.github.io/deep%20learning/visualising-activation-functions-in-neural-networks/

https://blog.paperspace.com/vanishing-gradients-activation-function/

https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6

http://iamtrask.github.io/2015/07/12/basic-python-network/

Resources

https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

https://adventuresinmachinelearning.com/stochastic-gradient-descent/

https://medium.com/coinmonks/stochastic-vs-mini-batch-training-in-machine-learning-using-tensorflow-and-python-7f9709143ee2

https://scikit-learn.org/stable/modules/neural_networks_supervised.html

https://stats.stackexchange.com/questions/164876/tradeoff-batch-size-vs-number-of-iterations-to-train-a-neural-network

https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network