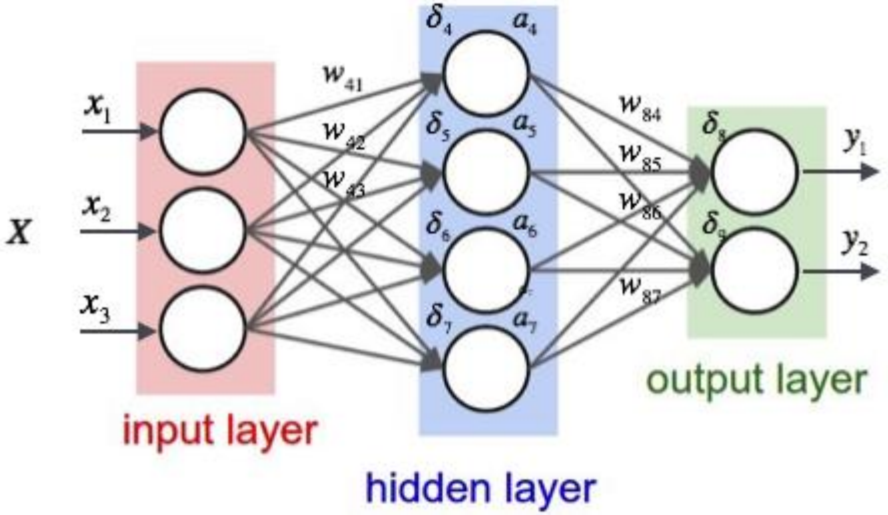
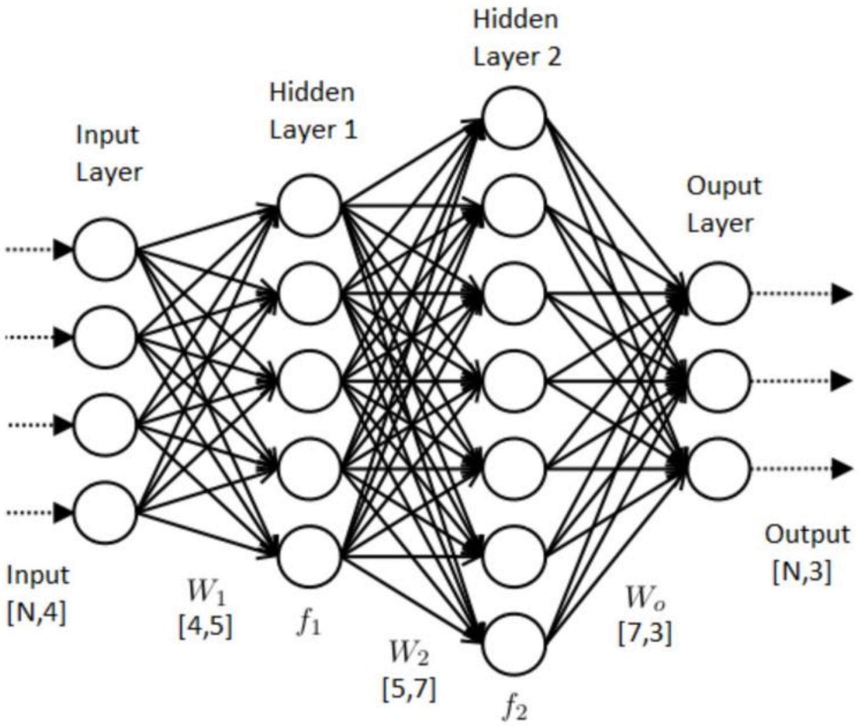
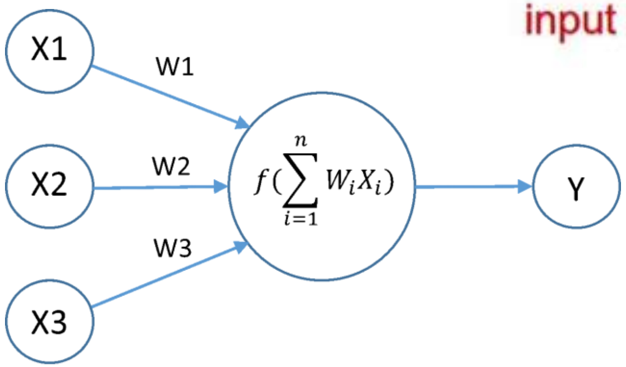
Recurrent Neural Networks

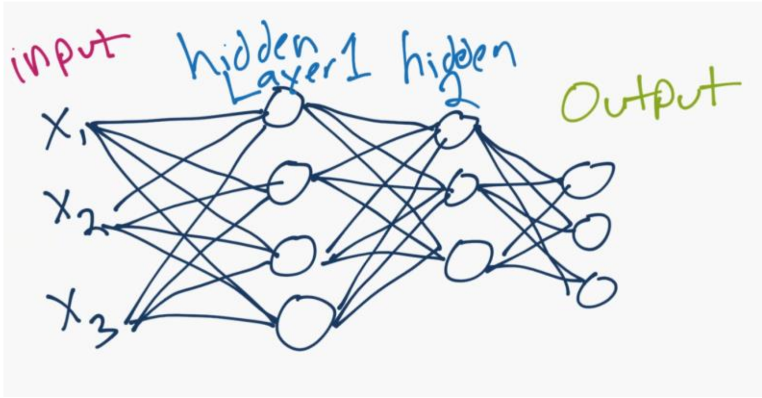
DATA SCIENCE & MACHINE LEARNING

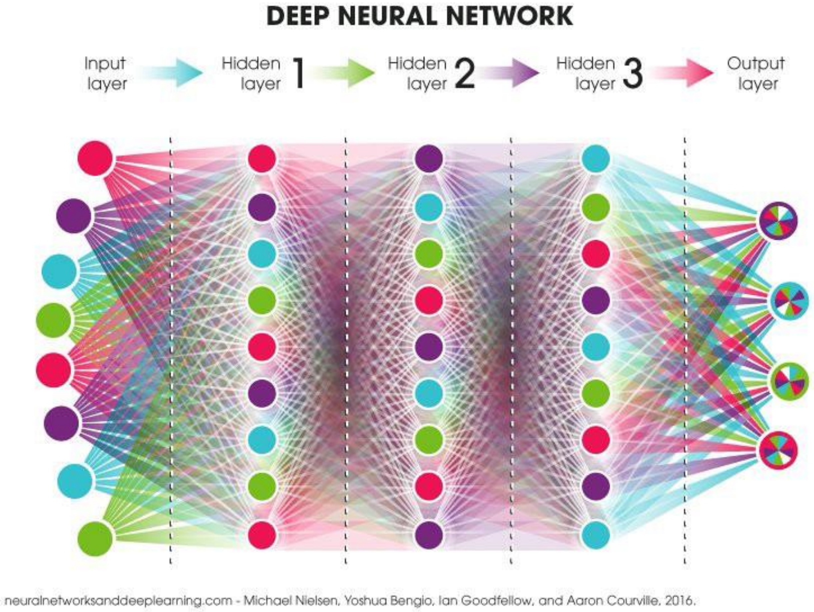
Neural Networks





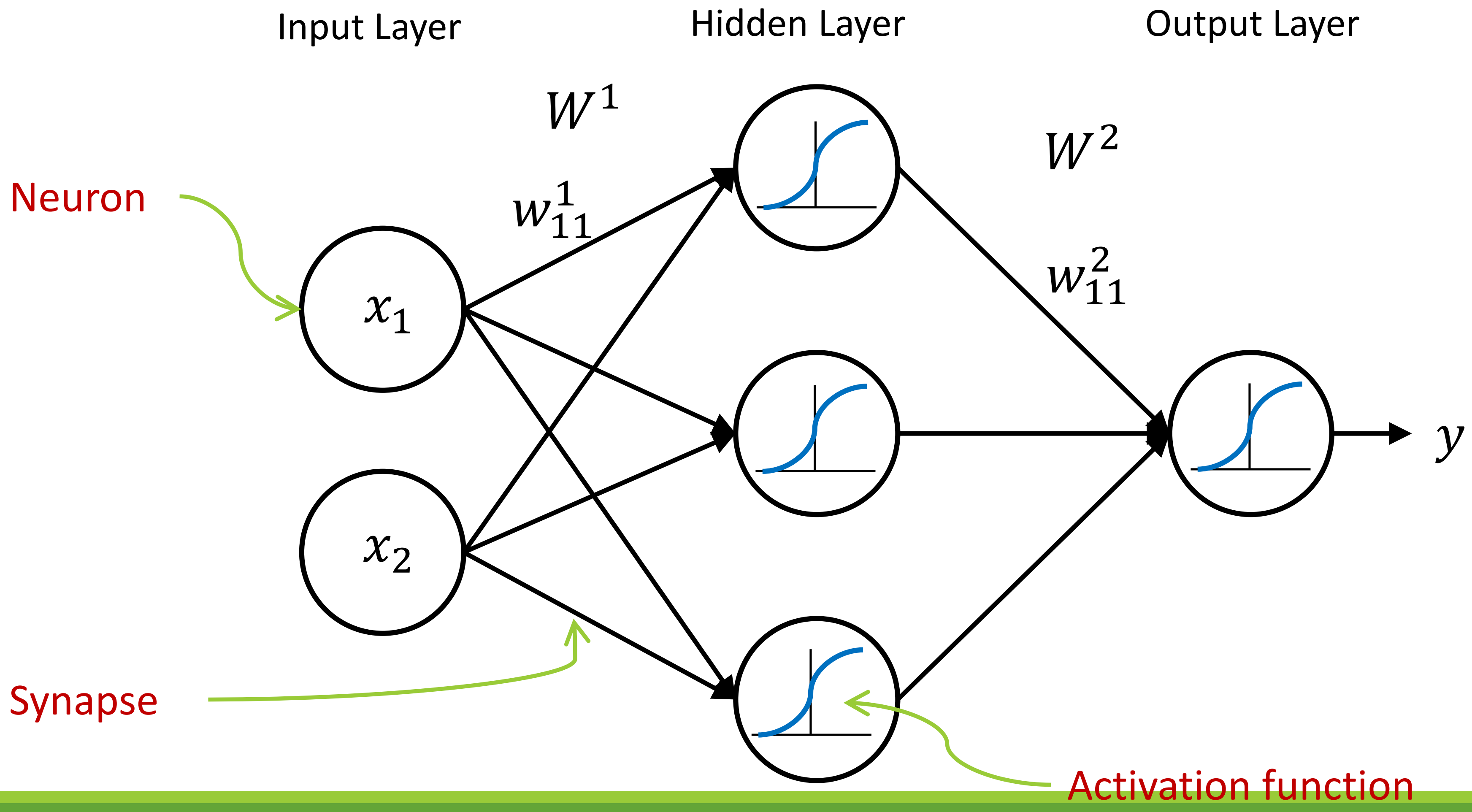






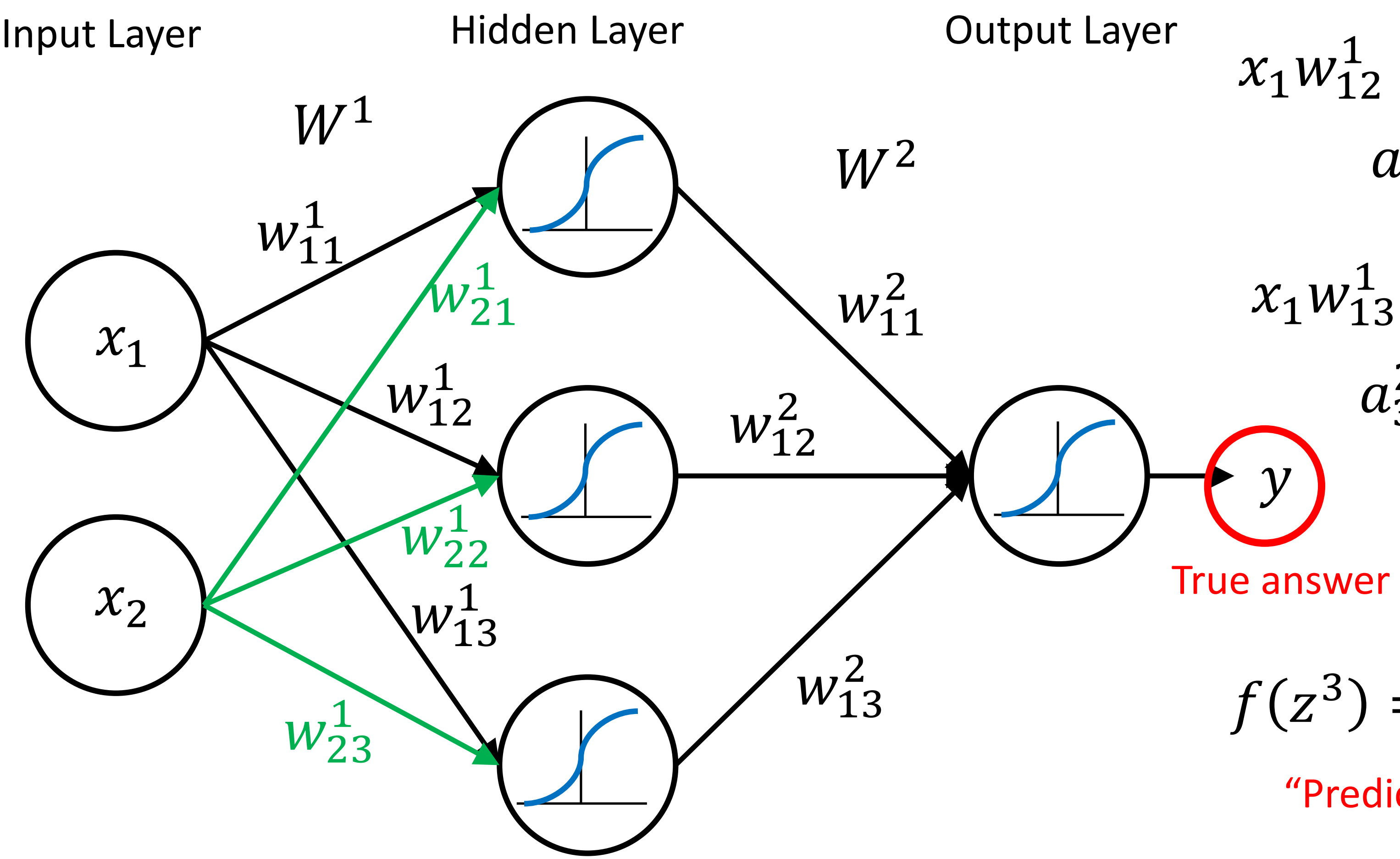
Click on image for image source

Neural Networks



Neural Networks

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



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

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

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

"Predicted" answer

Training Errors/Loss

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

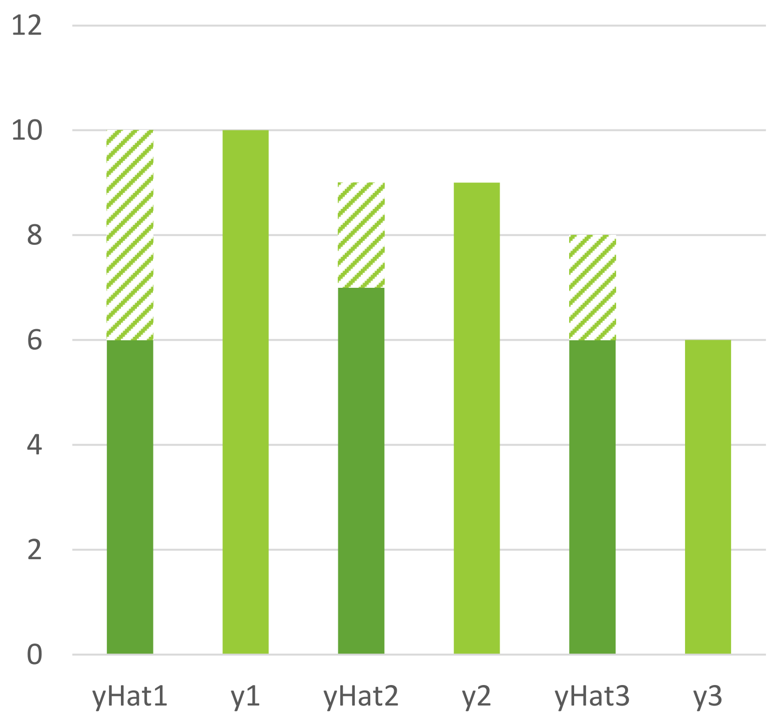
That is, the difference between the true answer and the predicted answer.

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

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

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

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

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

Neural Networks

y

True answer

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

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

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

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

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

"Predicted" answer

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

$$=$$

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

$$XW^1 = z^2$$

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

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

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!!

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

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

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

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

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

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.

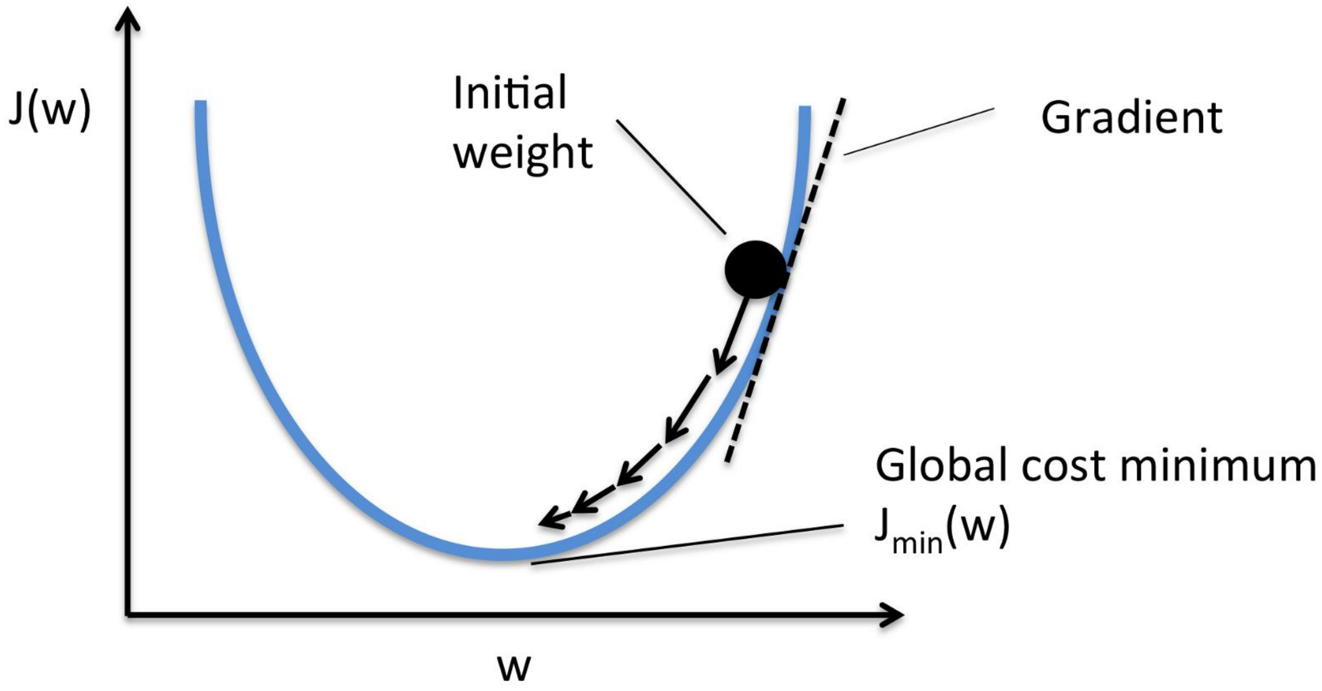
How do we Minimize training error/loss???

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

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.



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

How do we Minimize training error/loss???

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

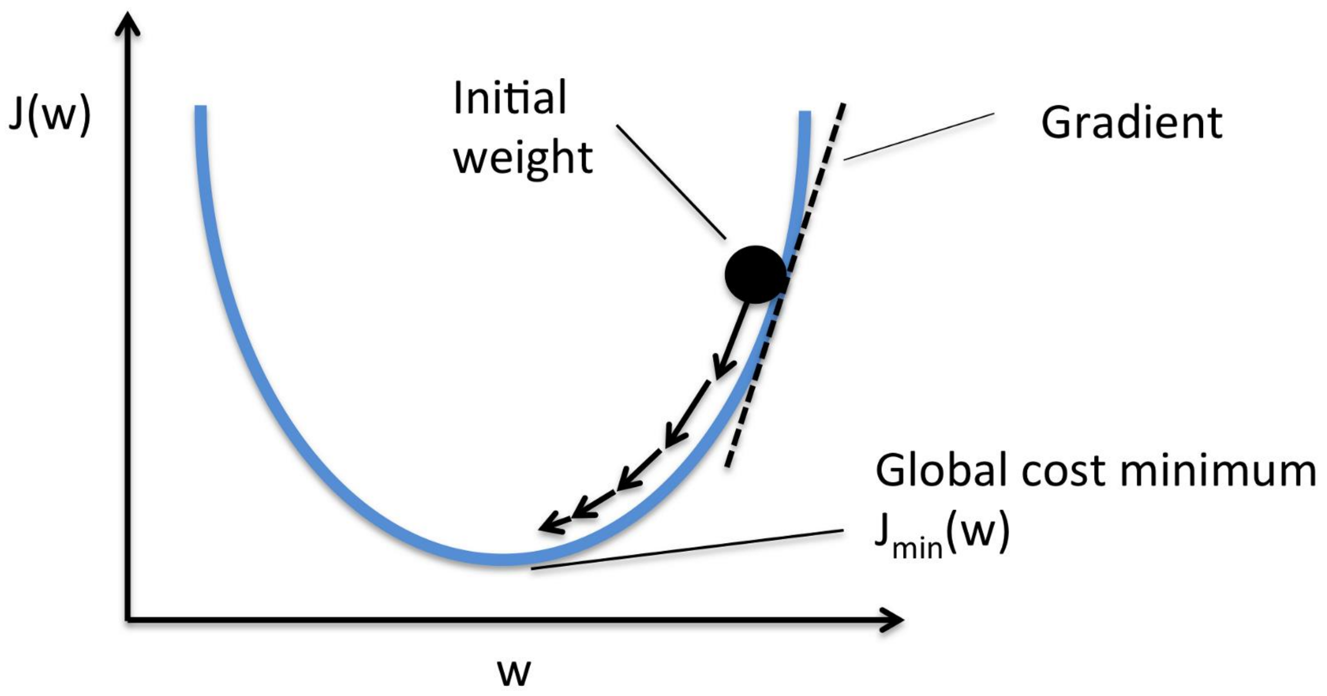
Gradient Descent

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



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

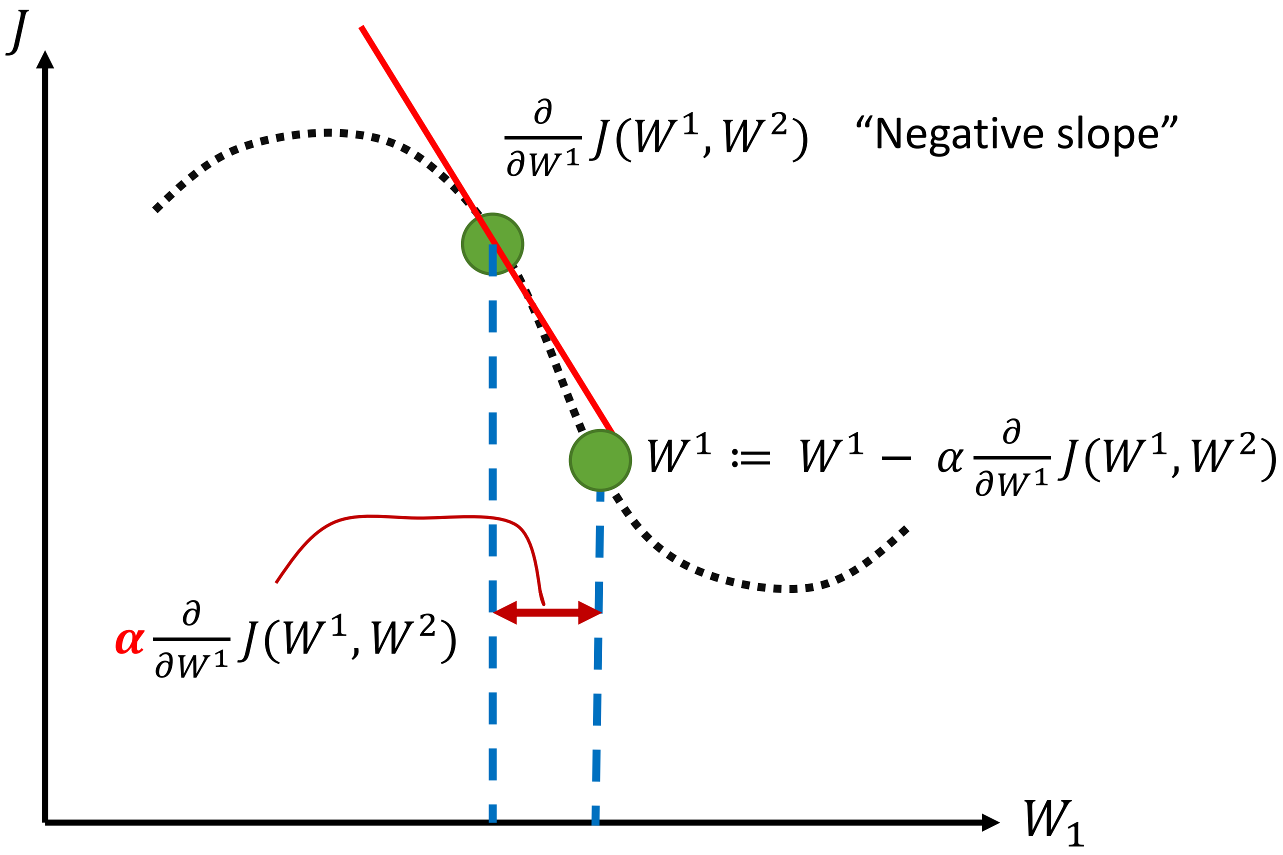
$$J = \frac{1}{m}\sum(f(f(XW^1)W^2) - y)^2$$

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

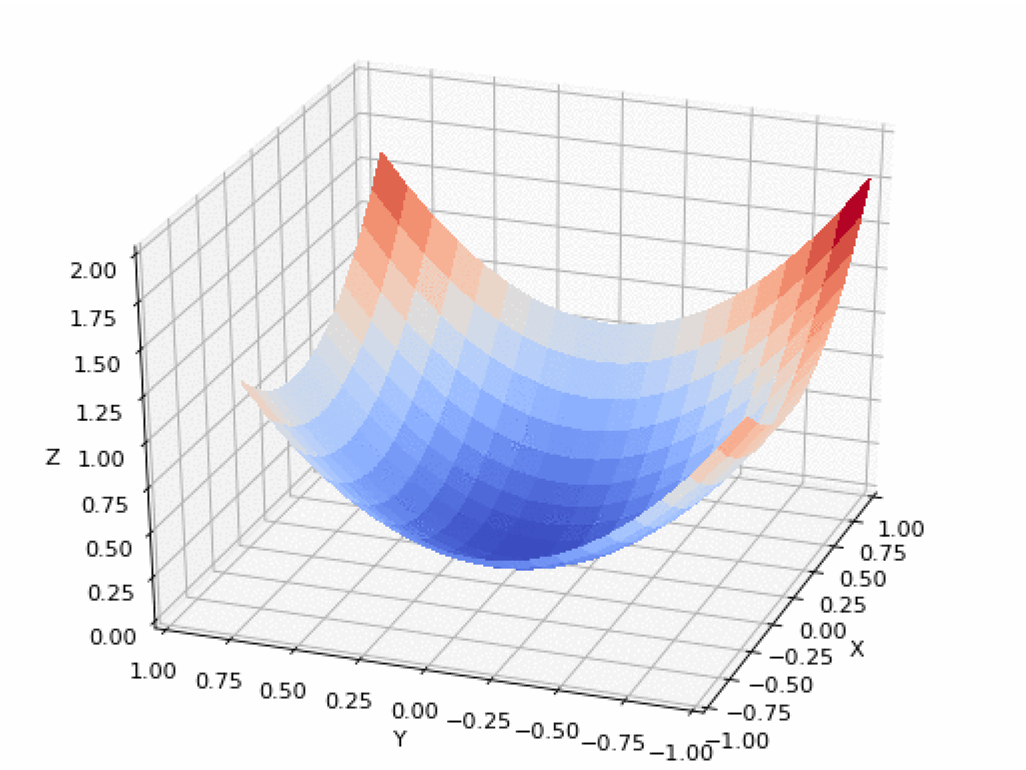
Update rule: $W^1 := W^1 - \alpha \frac{\partial}{\partial W^1} J(W^1, W^2)$

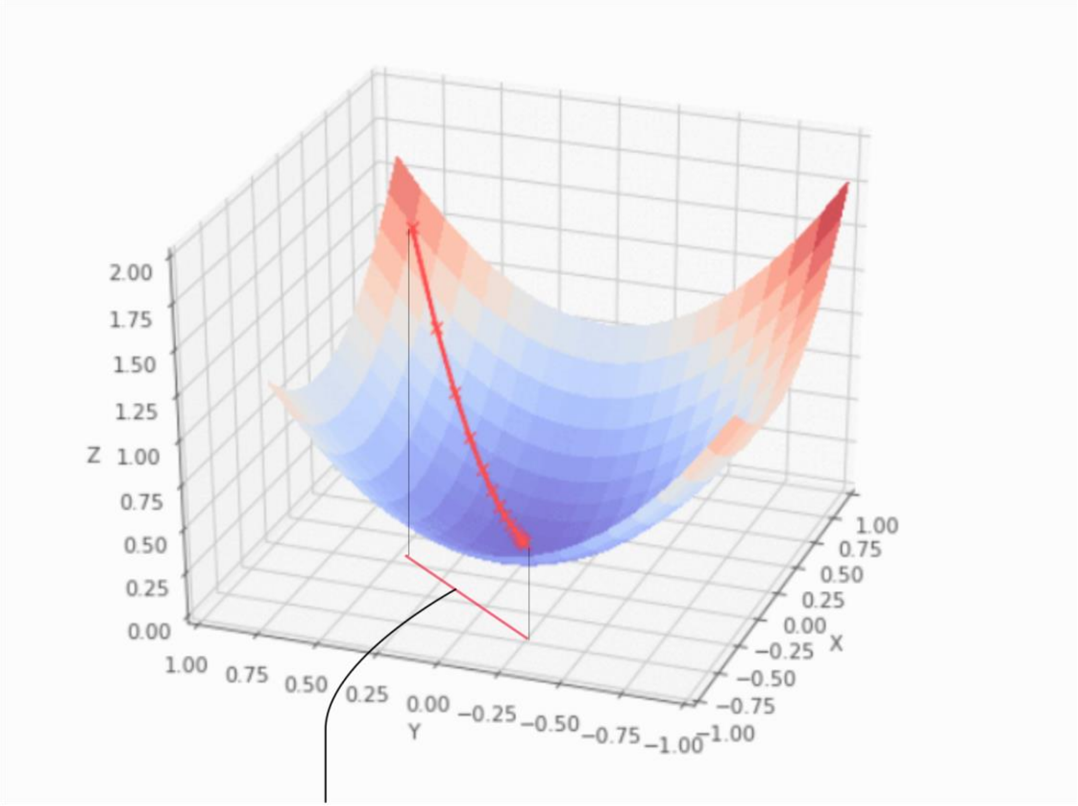
$W^2 := W^2 - \alpha \frac{\partial}{\partial W^2} J(W^1, W^2)$

α is learning rate.



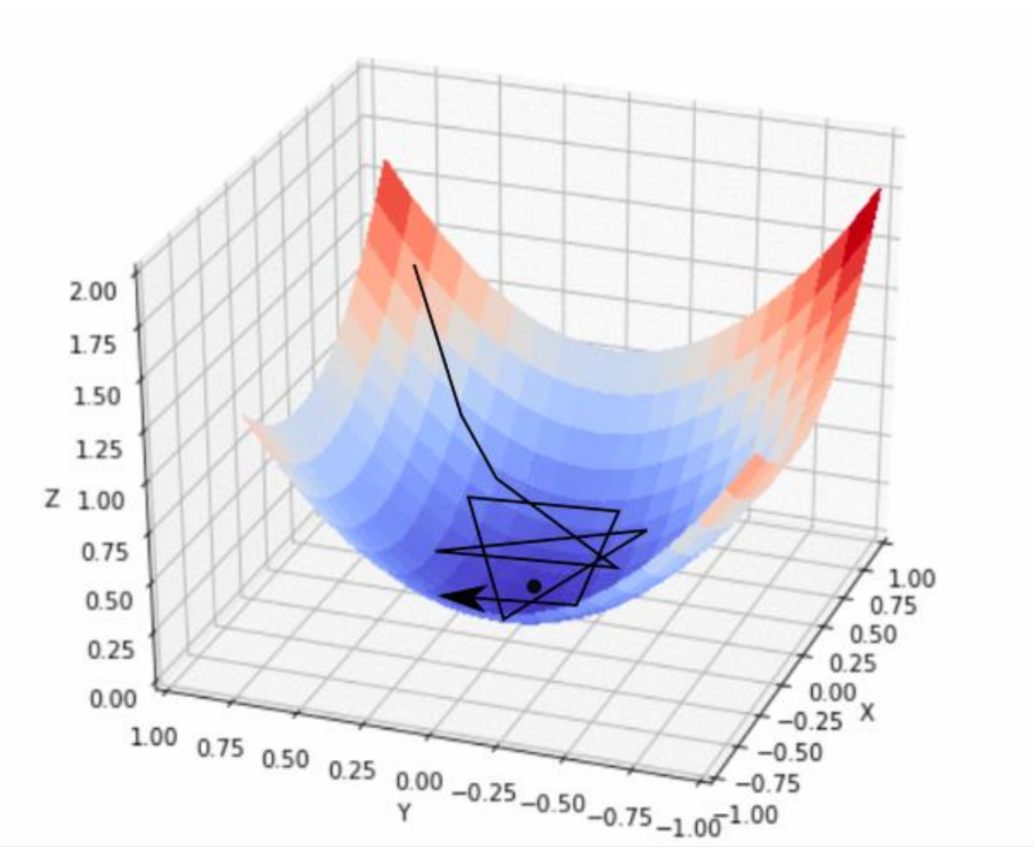
Gradient Descent





Real Trajectory of G.D.

large learning rates…



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

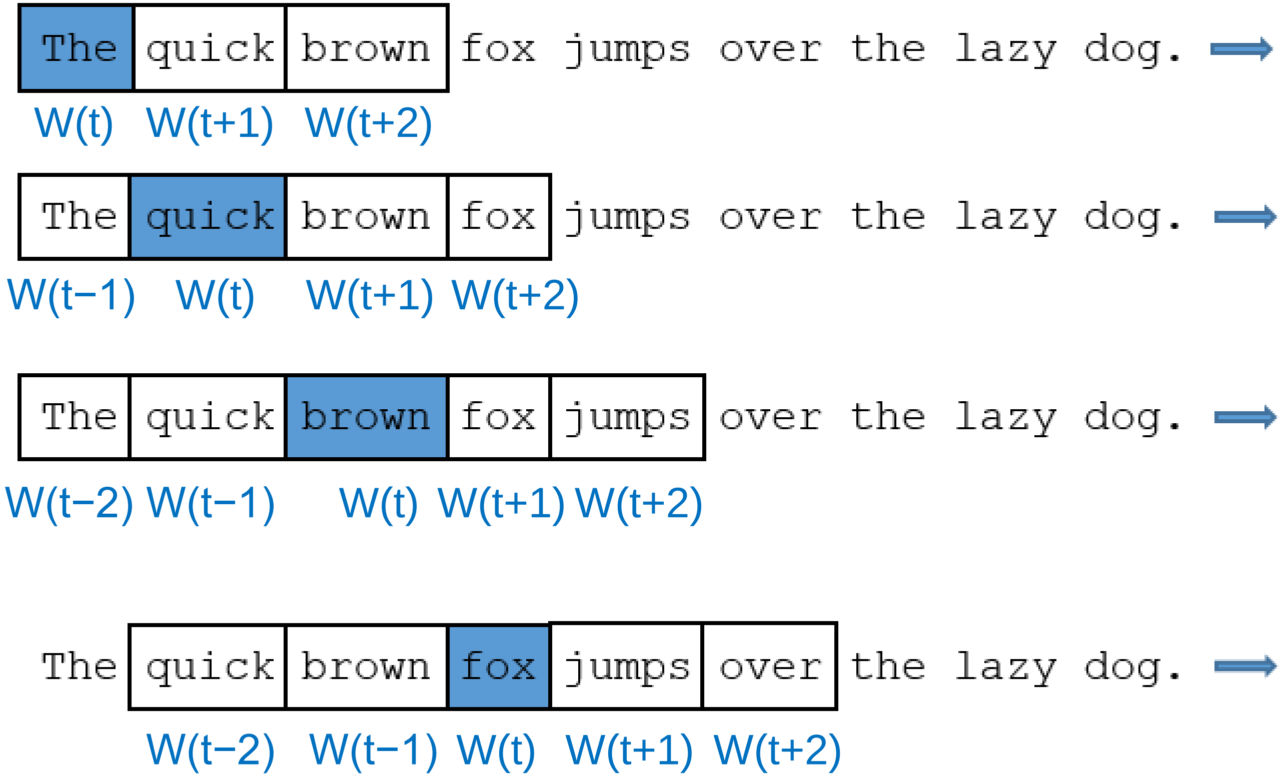
Word2Vec, Skip-Gram

Source Text	Training Samples
The quick brown fox jumps over the lazy dog. ⟹	(the, quick) (the, brown)
The **quick** brown fox jumps over the lazy dog. ⟹	(quick, the) (quick, brown) (quick, fox)
The quick **brown** fox jumps over the lazy dog. ⟹	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown **fox** jumps over the lazy dog. ⟹	(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

Source Text



Training Samples

(the, quick)
(the, brown)

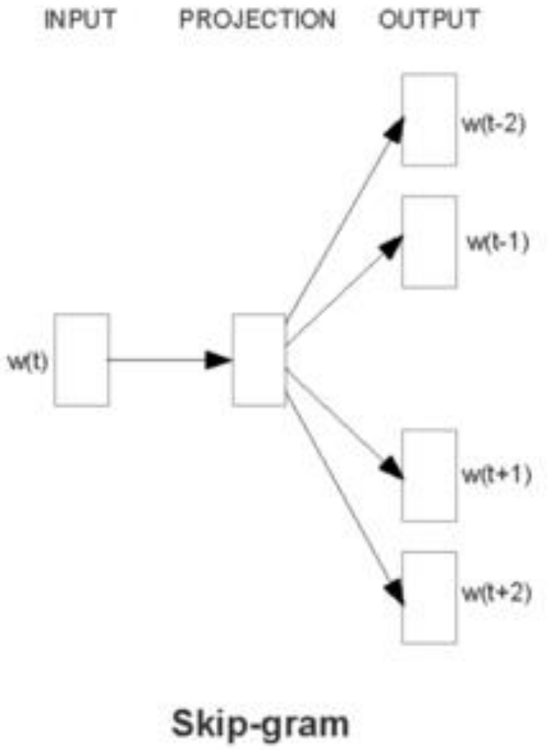
(quick, the)
(quick, brown)
(quick, fox)

(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)

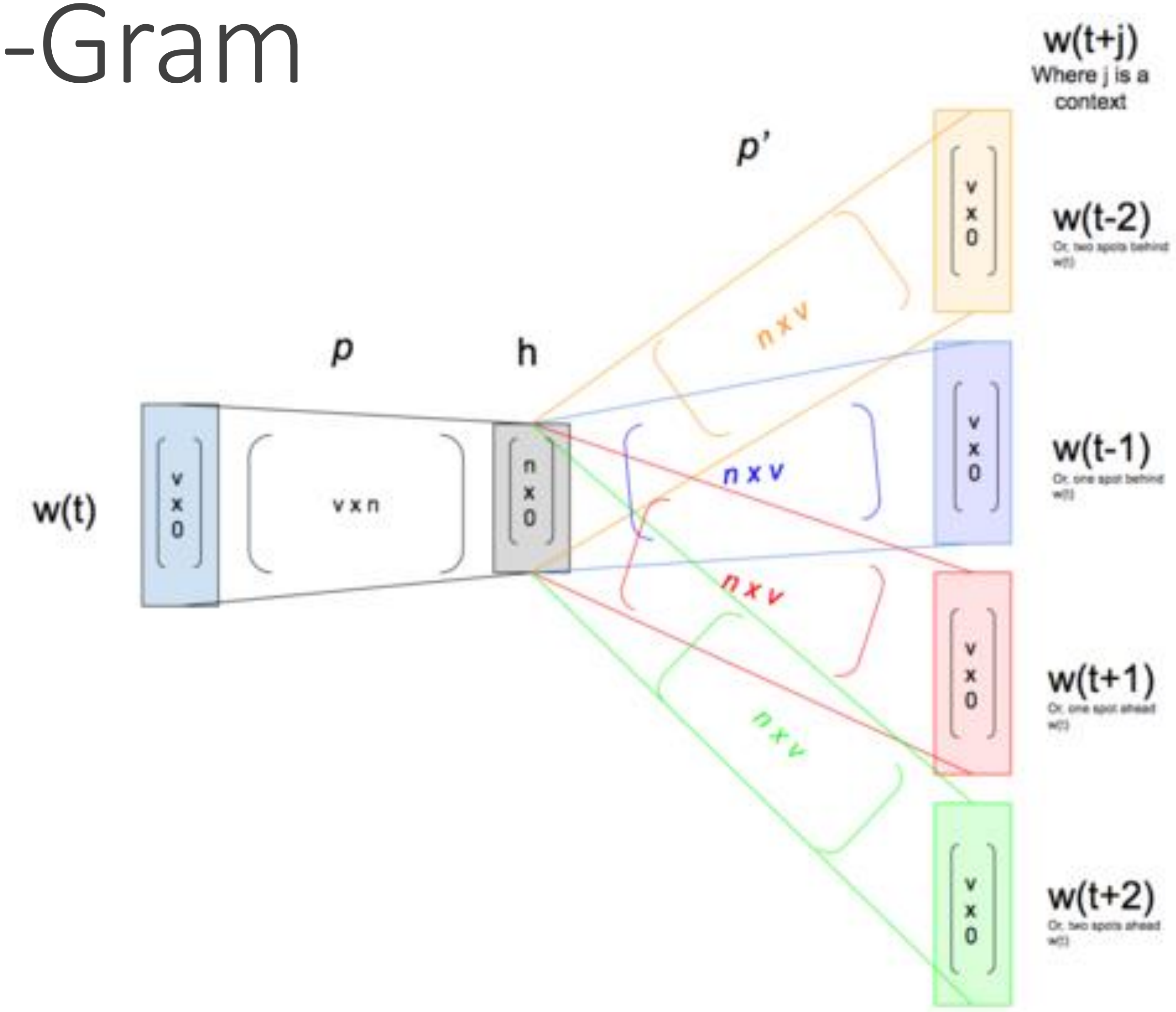
(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)

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

Word2Vec, Skip-Gram



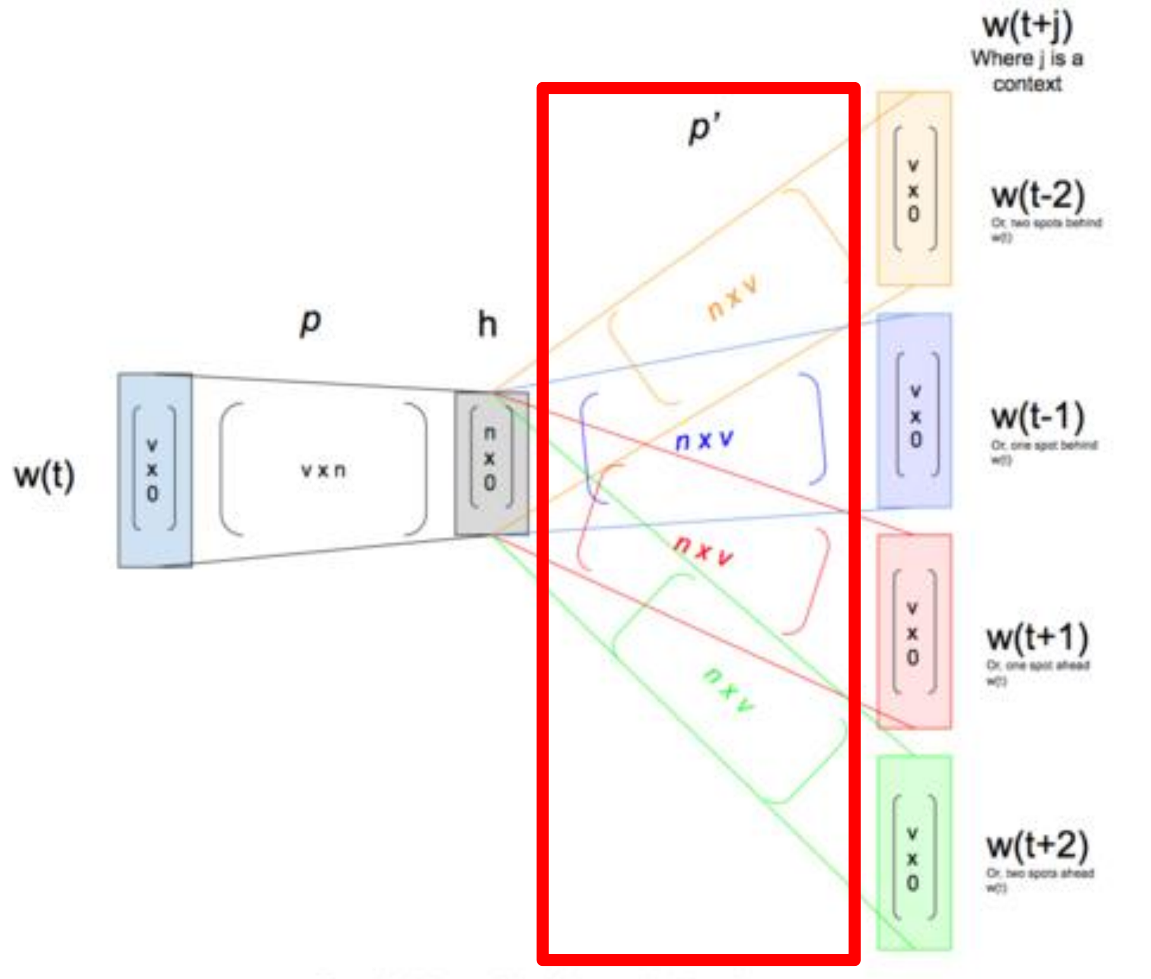
Original diagram from Mikolov et al (2013)



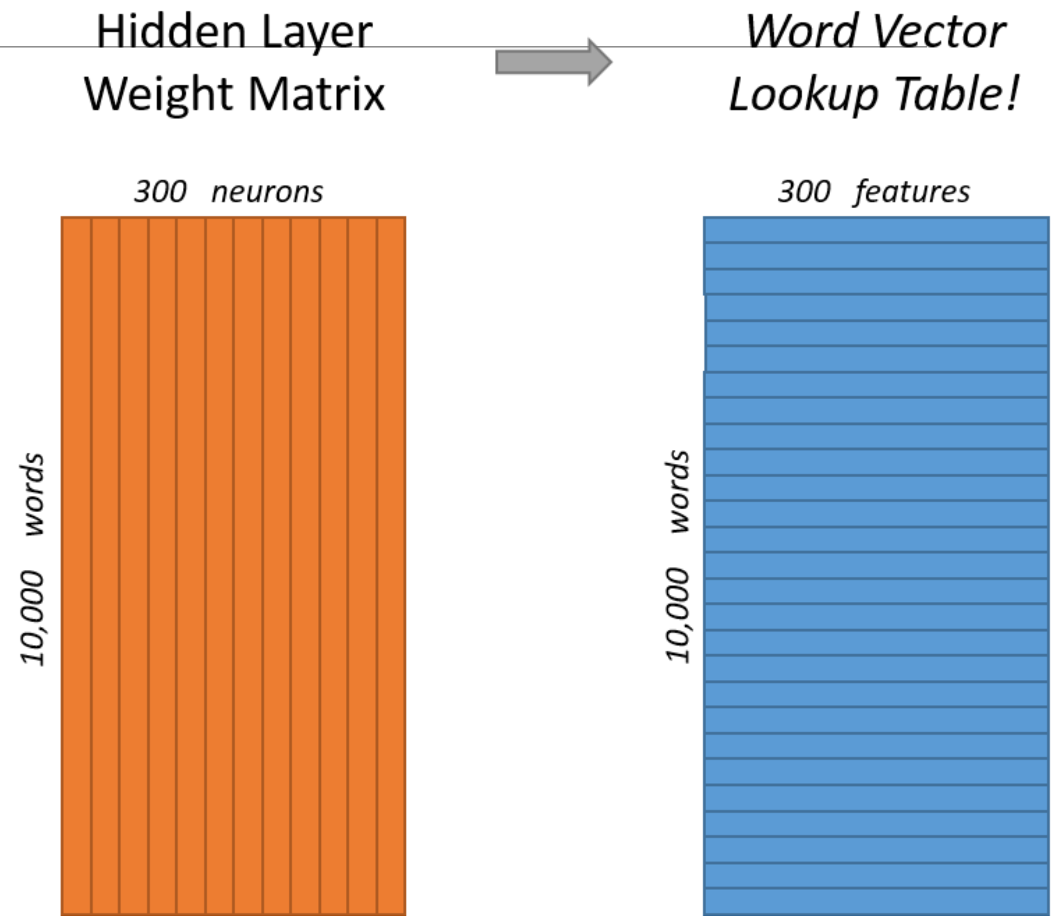
Extended diagram identifying matrix dimensions

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

Word2Vec, Skip-Gram



Extended diagram identifying matrix dimensions



Word2Vec, Skip-gram

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

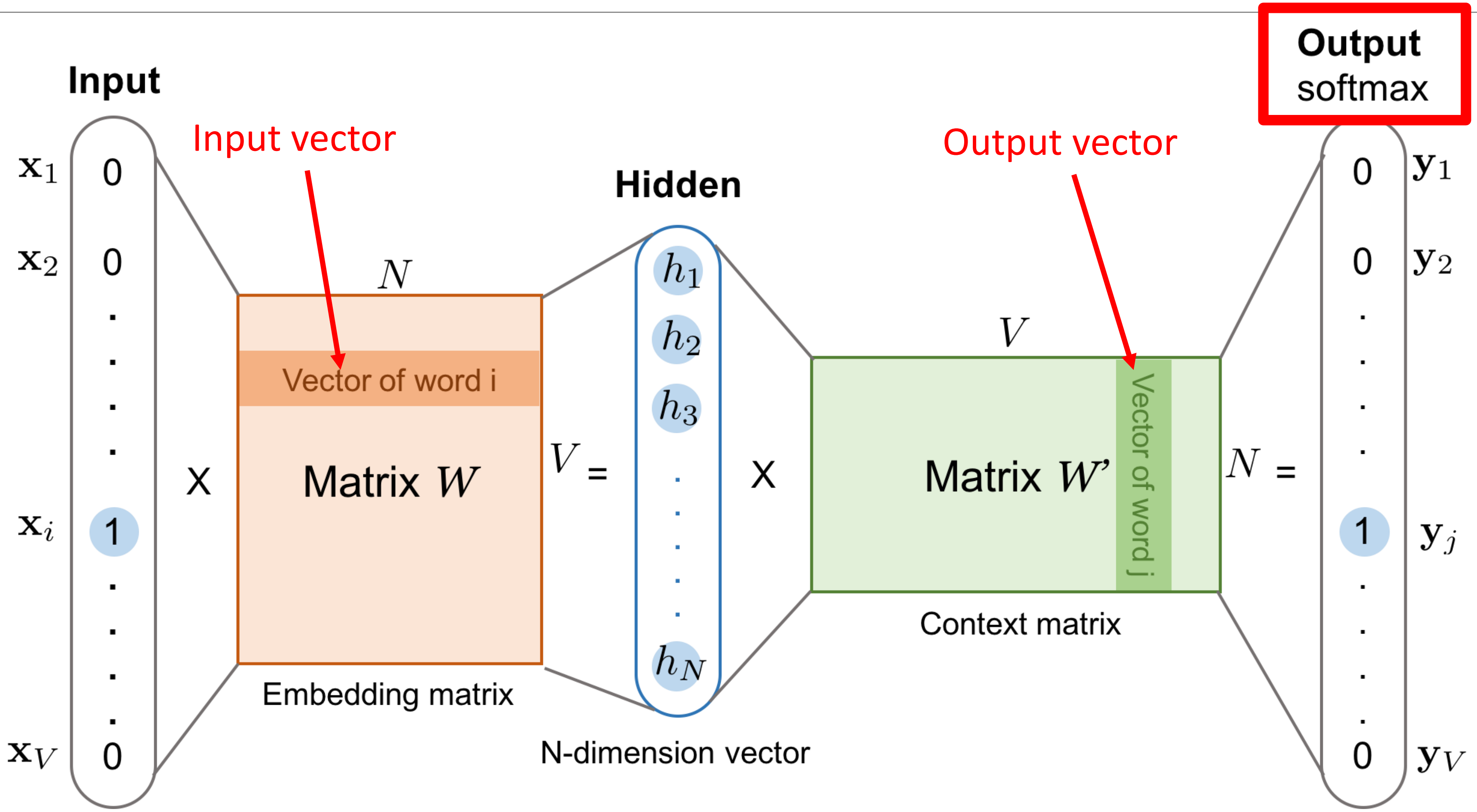


Image source: https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

RNN: Recurrent Neural Networks

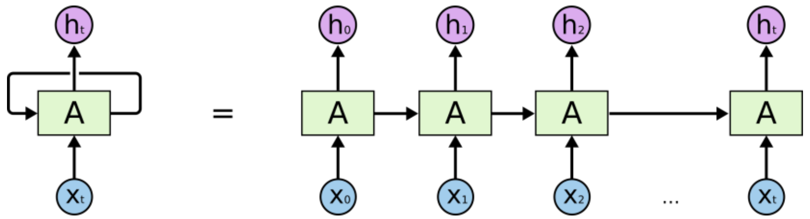
https://youtu.be/xCGidAeyS4M

https://youtu.be/6niqTuYFZLQ

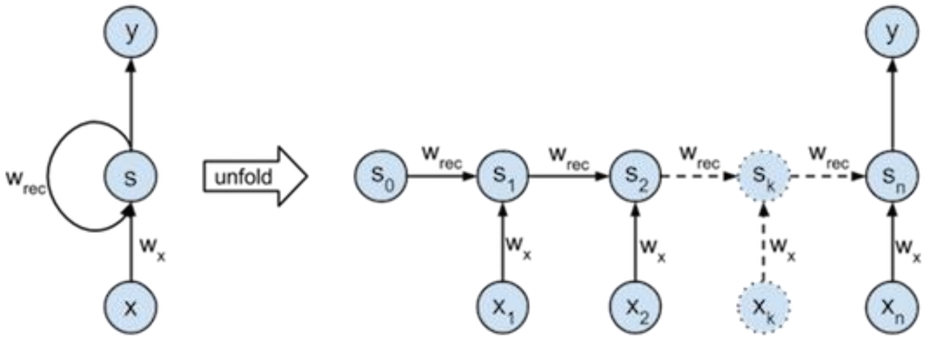
https://youtu.be/rTqmWlnwz_0

資管系 柯士文 George Ke

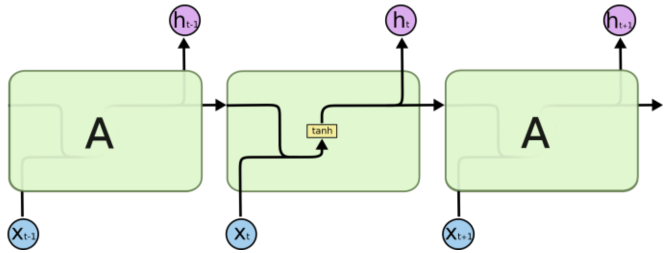
RNN



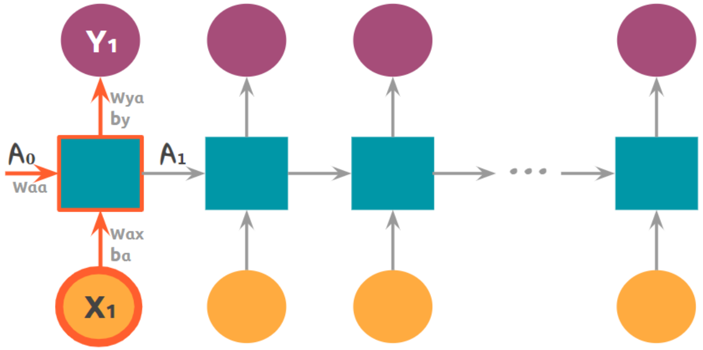
http://colah.github.io/posts/2015-08-Understanding-LSTMs/



https://peterroelants.github.io/posts/rnn-implementation-part01/



http://colah.github.io/posts/2015-08-Understanding-LSTMs/



- When it was ANN,

$Z_1 = W_1 \cdot X + b_1$

$A_1 = g(Z_1)$

→

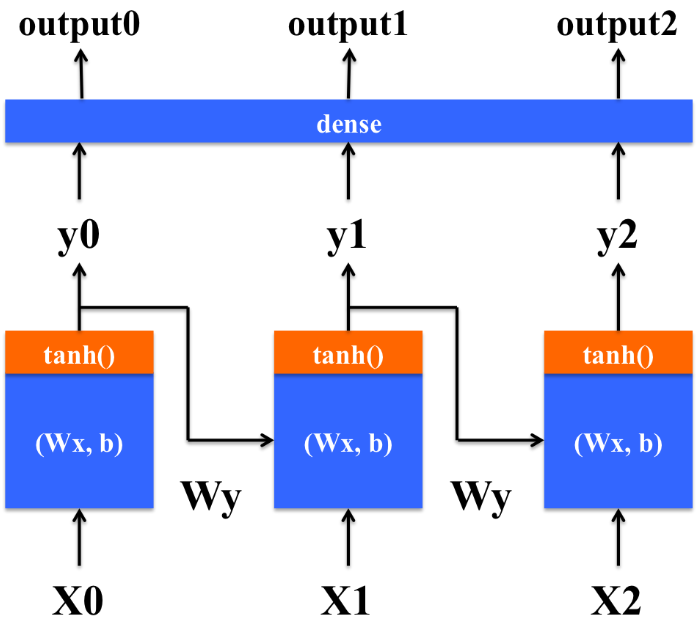
- Now in RNN,

$Z_1 = W_{ax} \cdot X_1 + W_{aa} \cdot A_0 + b_a$ ···· ①

$A_1 = g(Z_1)$ ···· ②

$Y_1 = g(W_{ya} \cdot A_1 + b_y)$ ···· ③

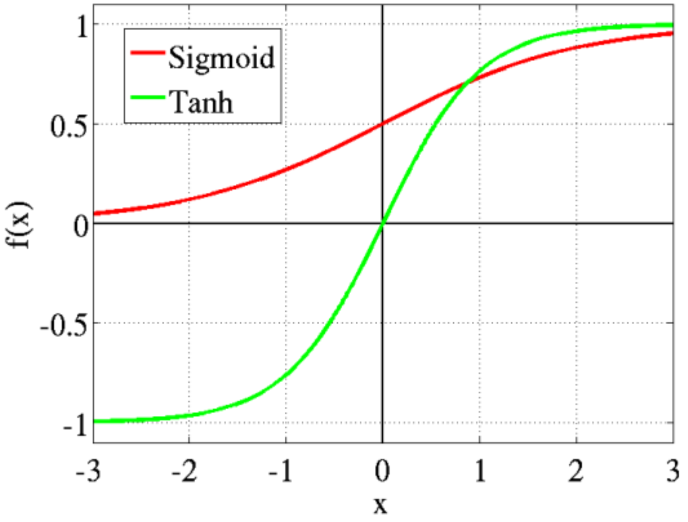
https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-recurrent-neural-network-873c29da73c7



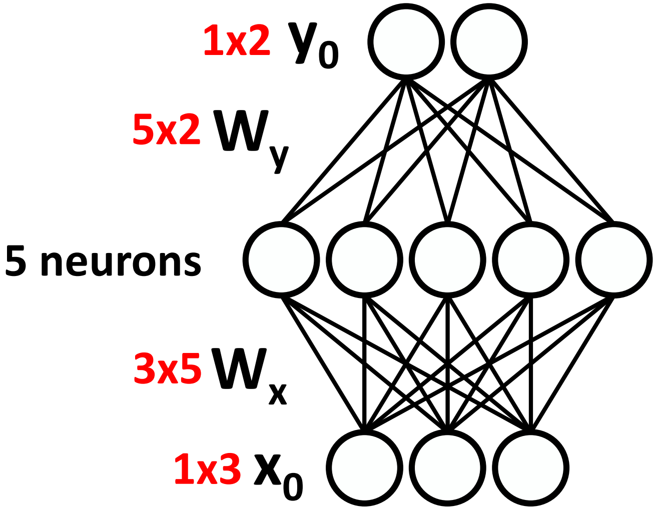
https://medium.com/machine-learning-algorithms/basic-recurrent-neural-network-tutorial-5ea479ac6f82

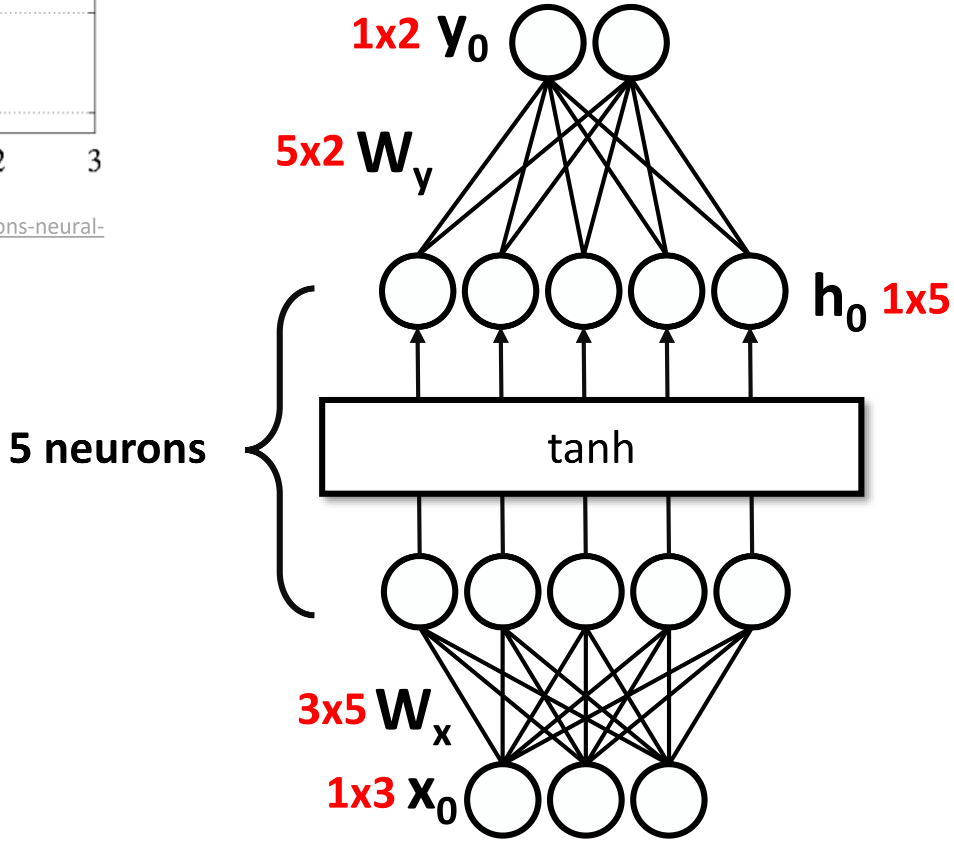
資管系 柯士文 George Ke

RNN



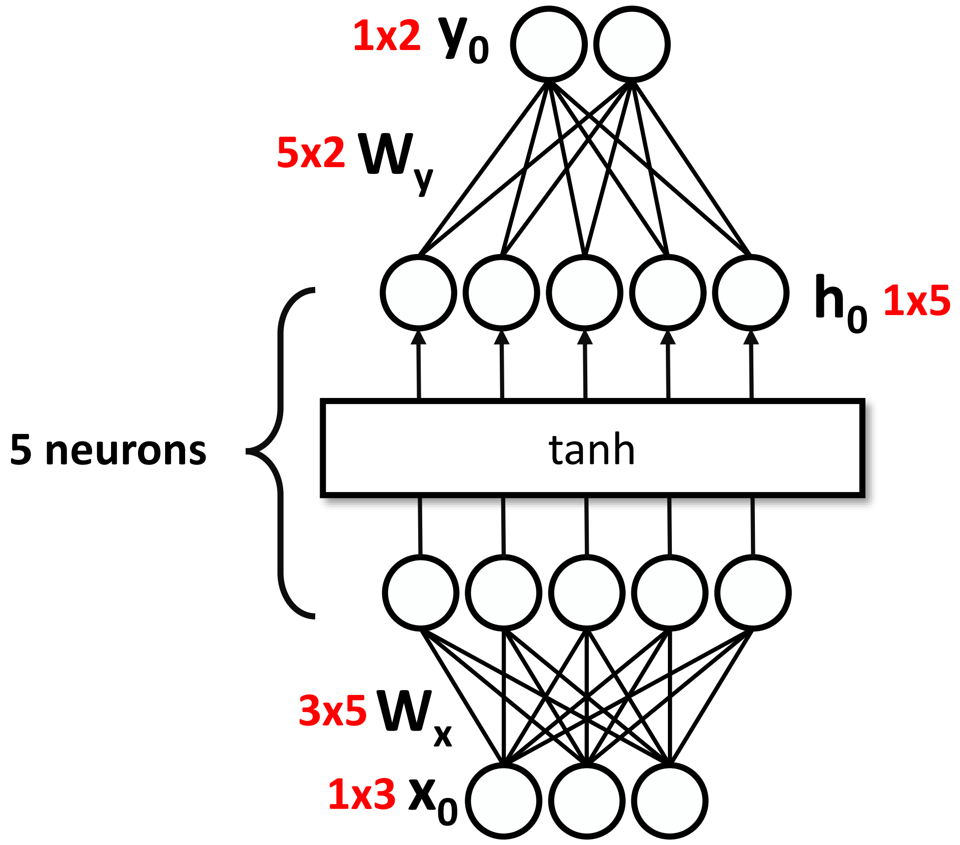
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6





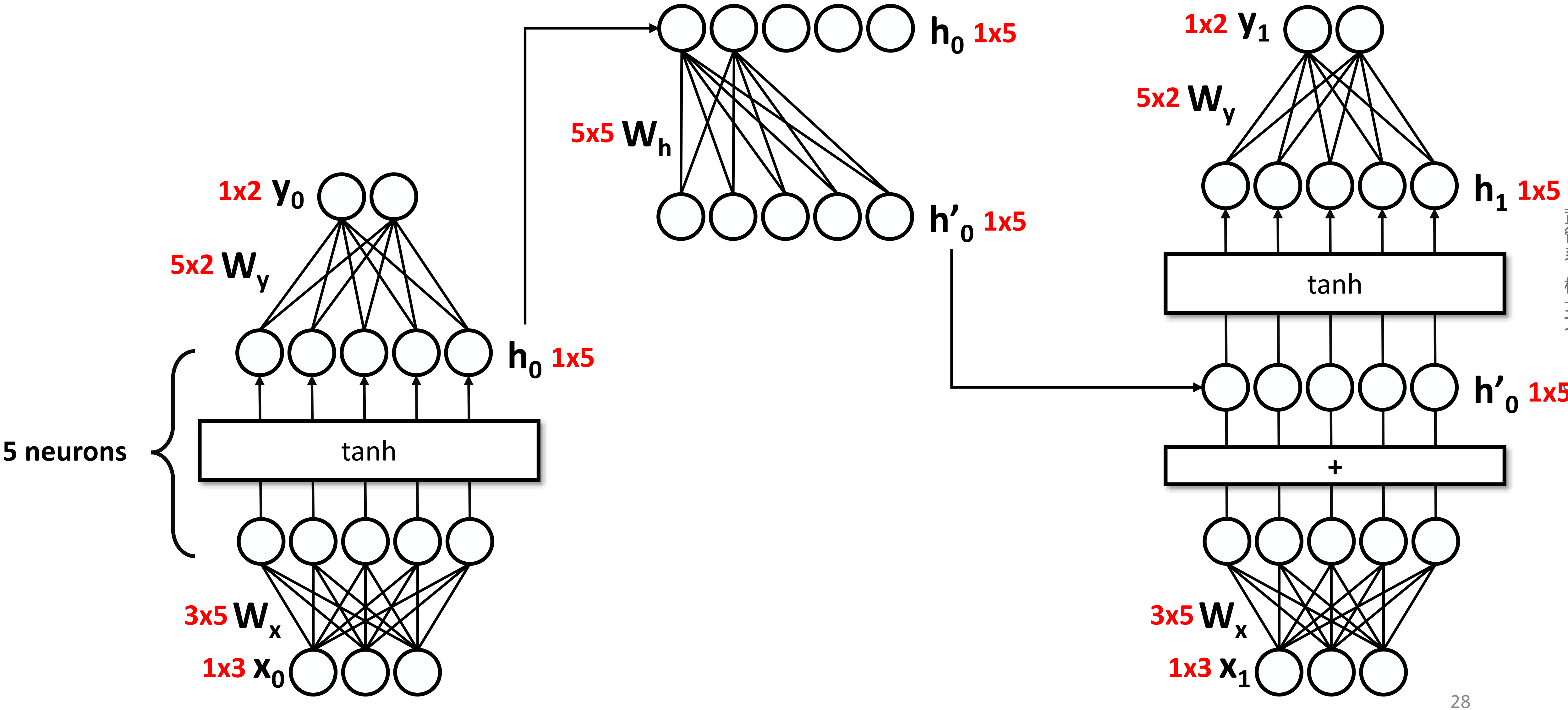
資管系 柯士文 George Ke

RNN



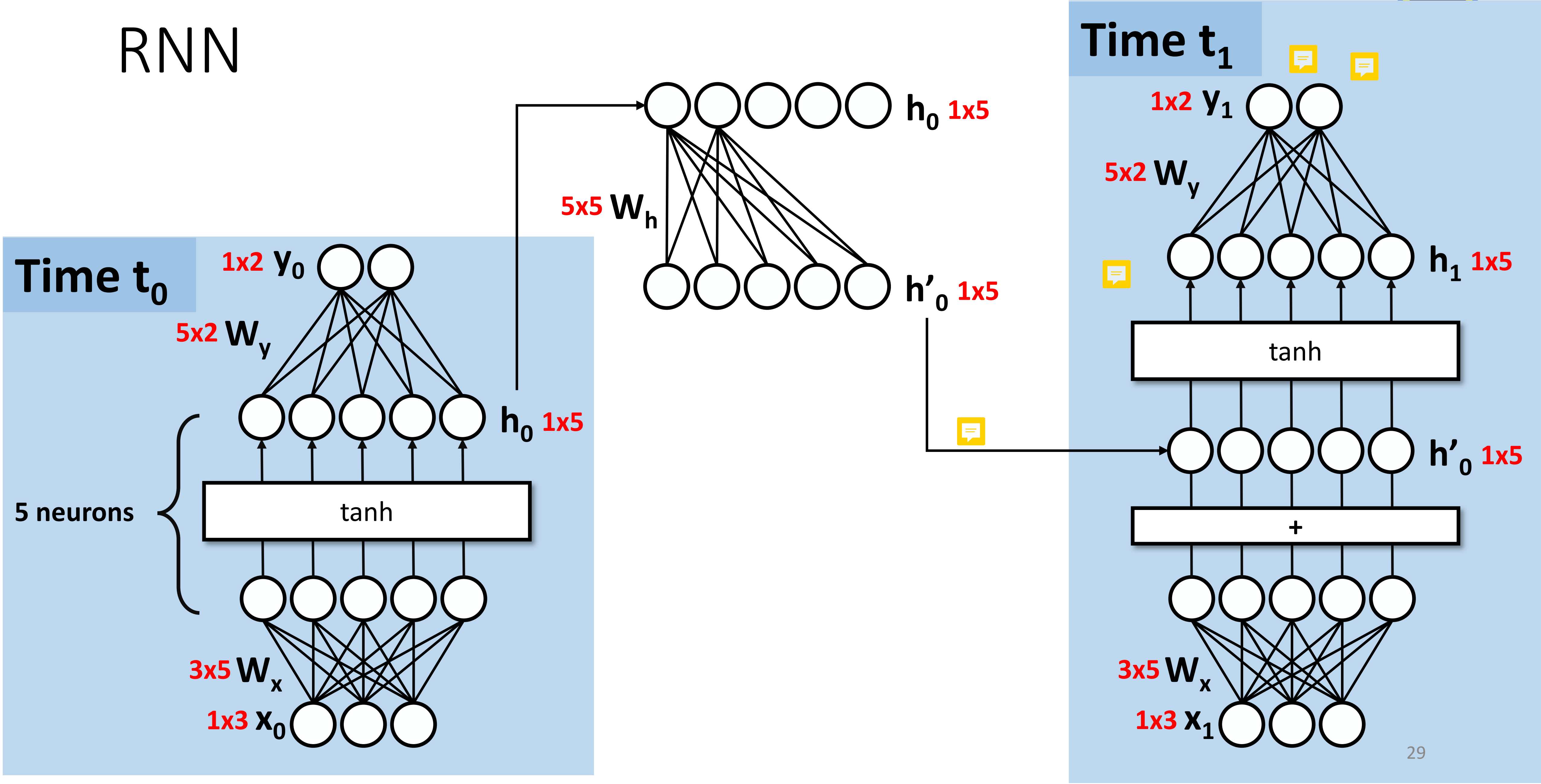
資管系 柯士文 George Ke

RNN

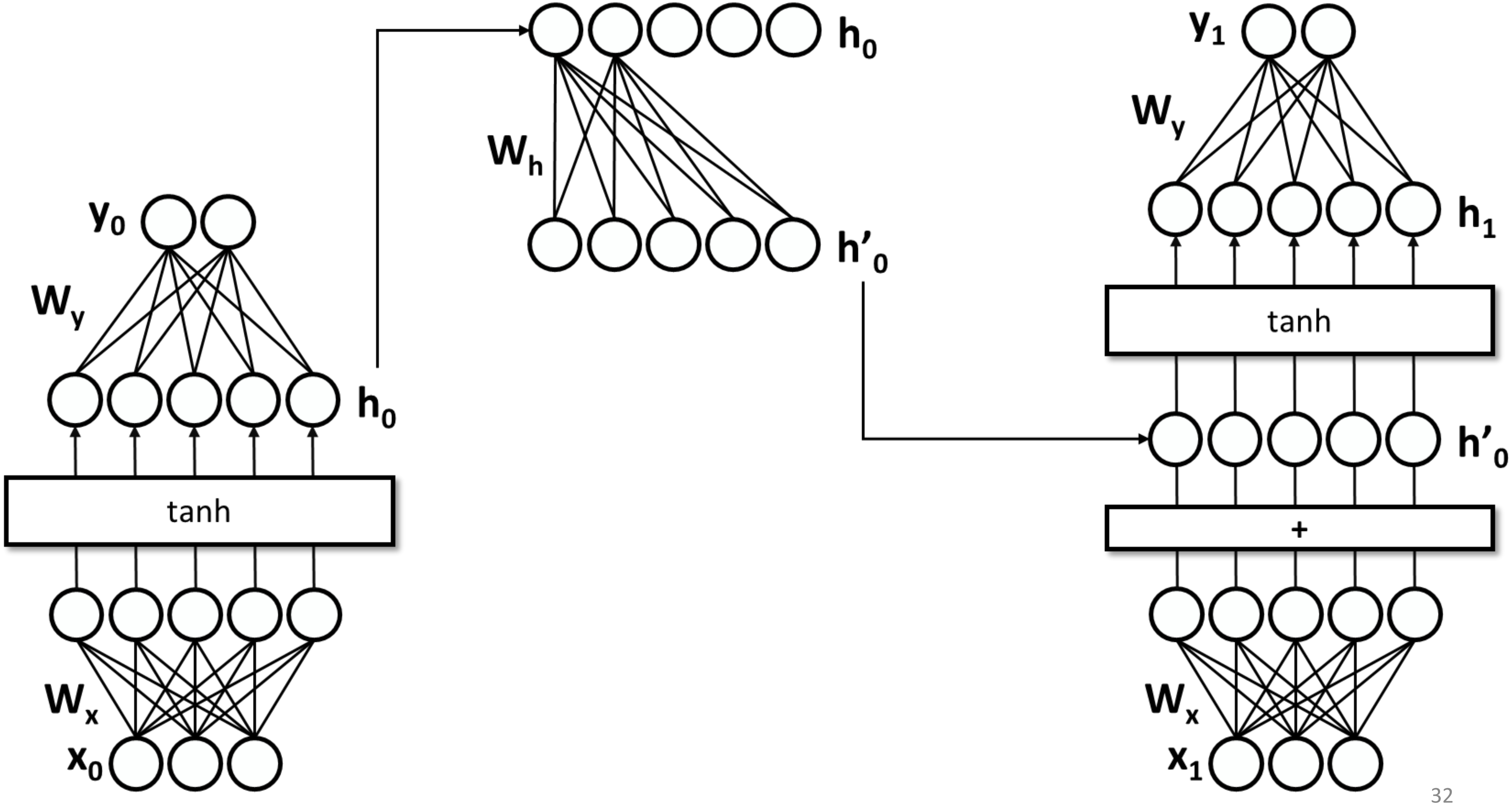


資管系 柯士文 George Ke

RNN

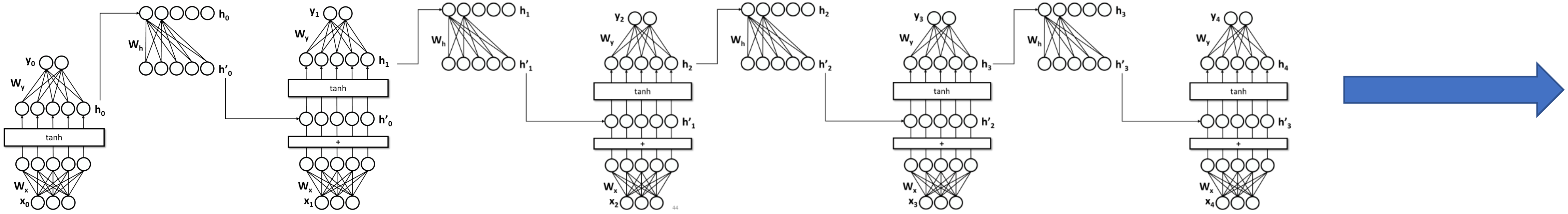

Time t_0
1x2 y_0
5x2 W_y
h_0 1x5
5 neurons
tanh
3x5 W_x
1x3 x_0
h_0 1x5
5x5 W_h
h'_0 1x5
Time t_1
1x2 y_1
5x2 W_y
h_1 1x5
tanh
h'_0 1x5
+
3x5 W_x
1x3 x_1

RNN



資管系 柯士文 George Ke

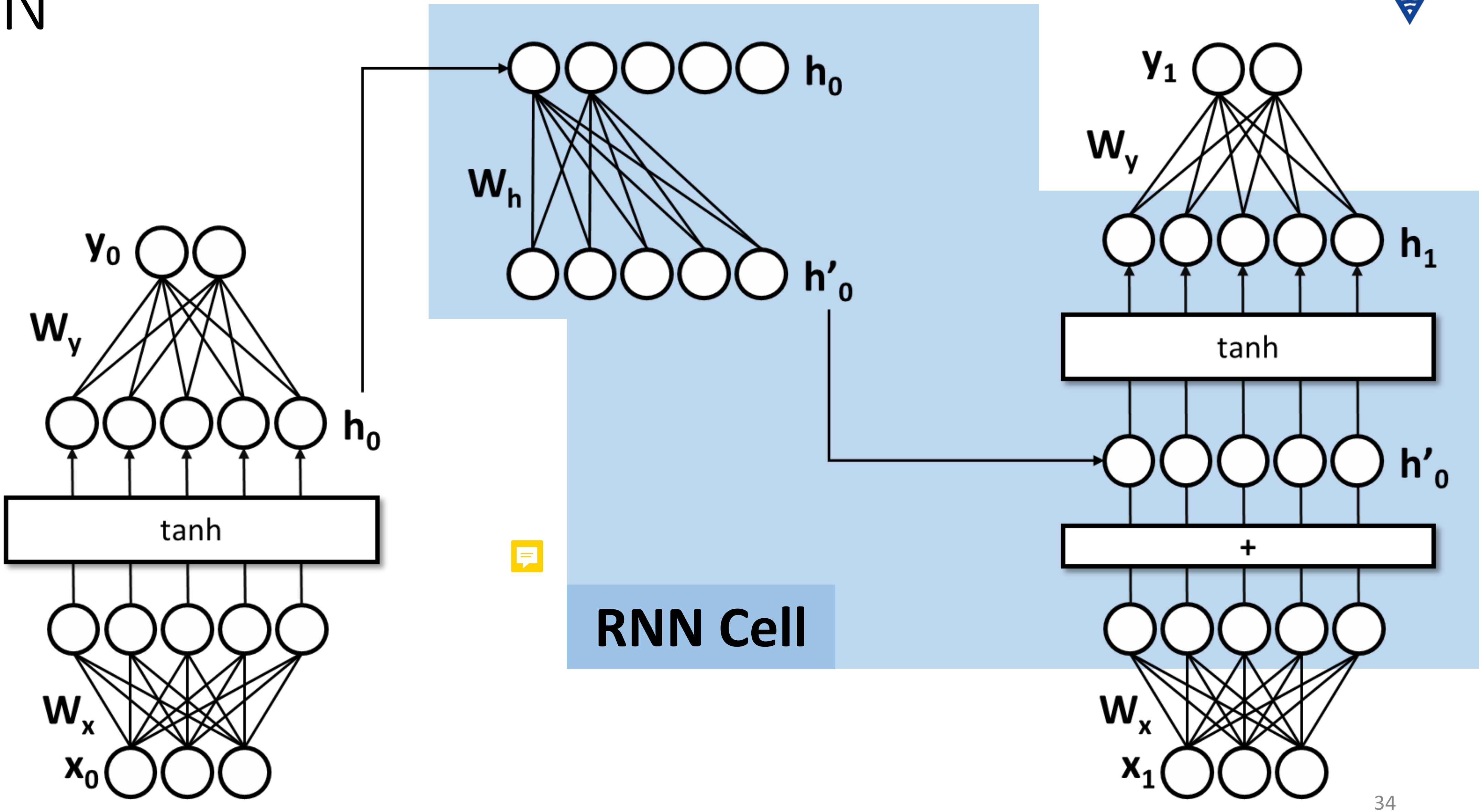
RNN



Time t_0 **Time t_1** **Time t_2** **Time t_3** **Time t_4**

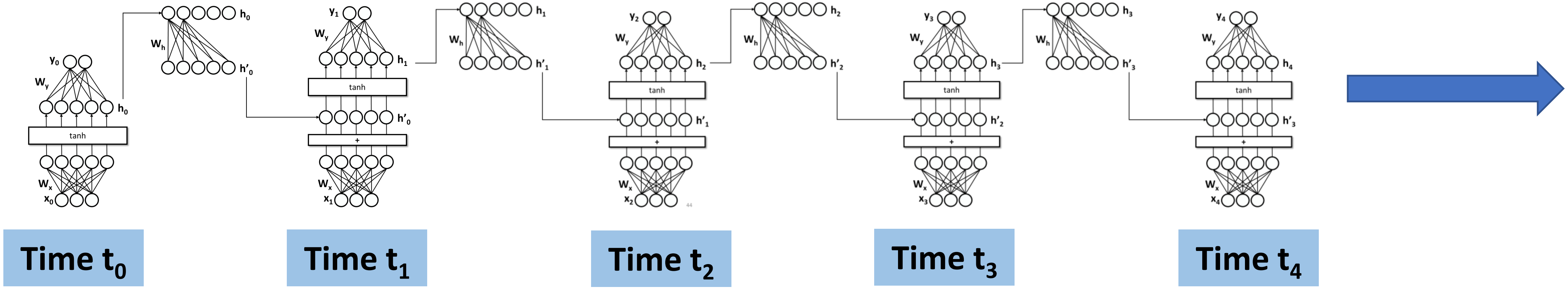
資管系 柯士文 George Ke

RNN

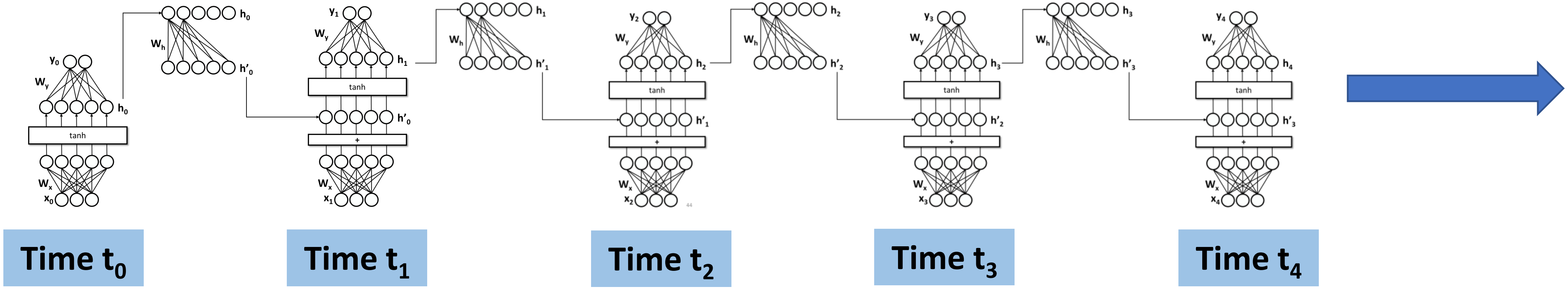


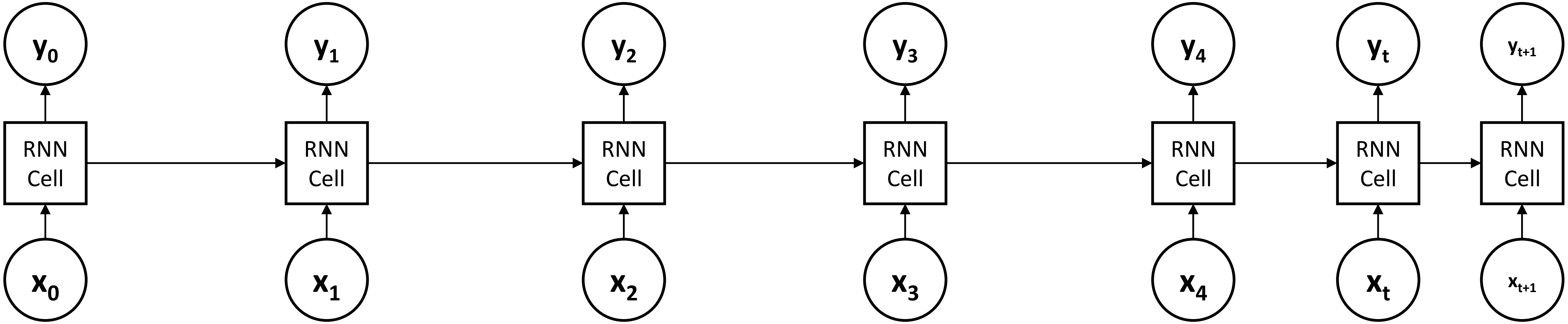
資管系 柯士文 George Ke

RNN



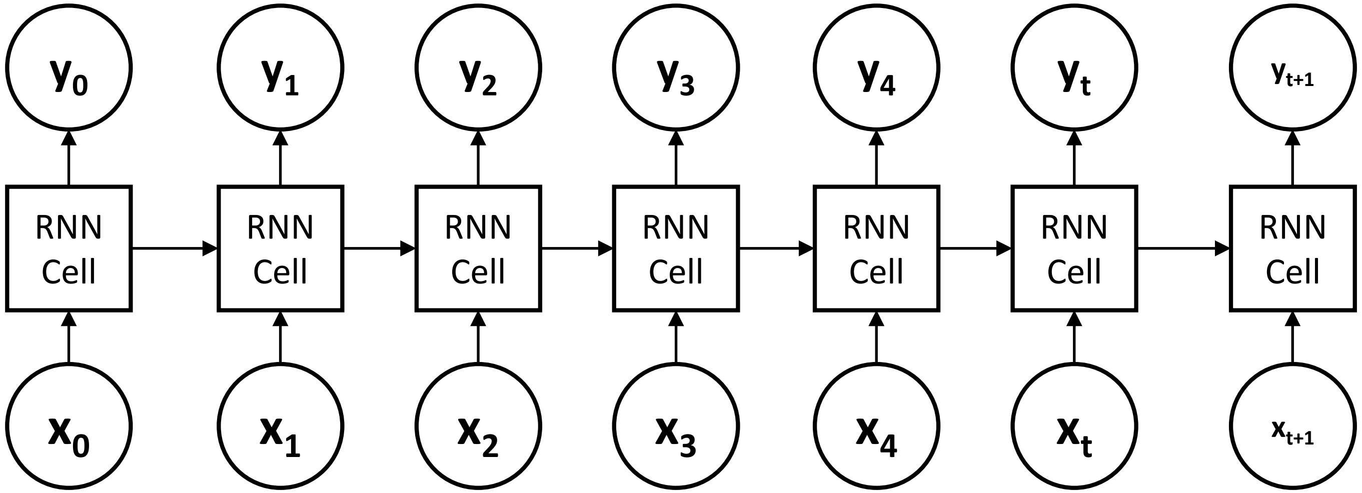
RNN





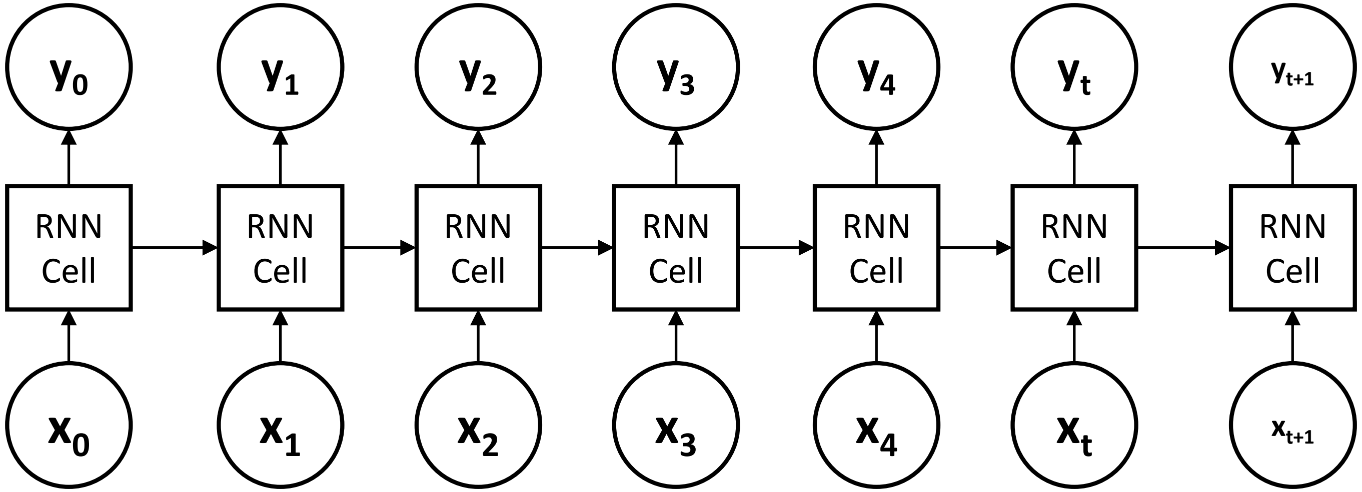
資管系 柯士文 George Ke

RNN



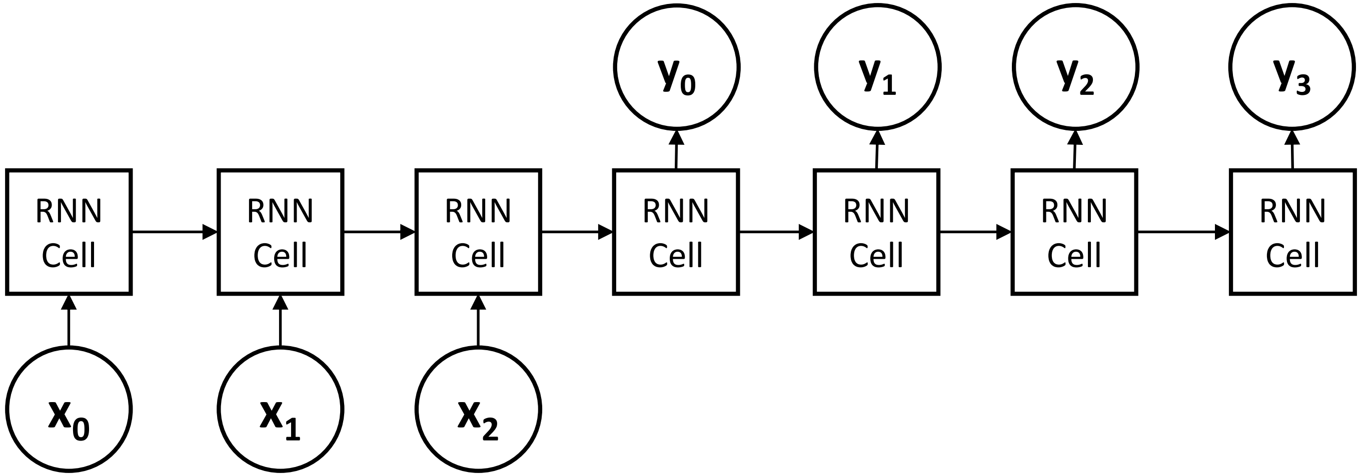
y_0
y_1
y_2
y_3
y_4
y_t
y_{t+1}
RNN Cell
RNN Cell
RNN Cell
RNN Cell
RNN Cell
RNN Cell
RNN Cell
x_0
x_1
x_2
x_3
x_4
x_t
x_{t+1}

RNN



Many to many

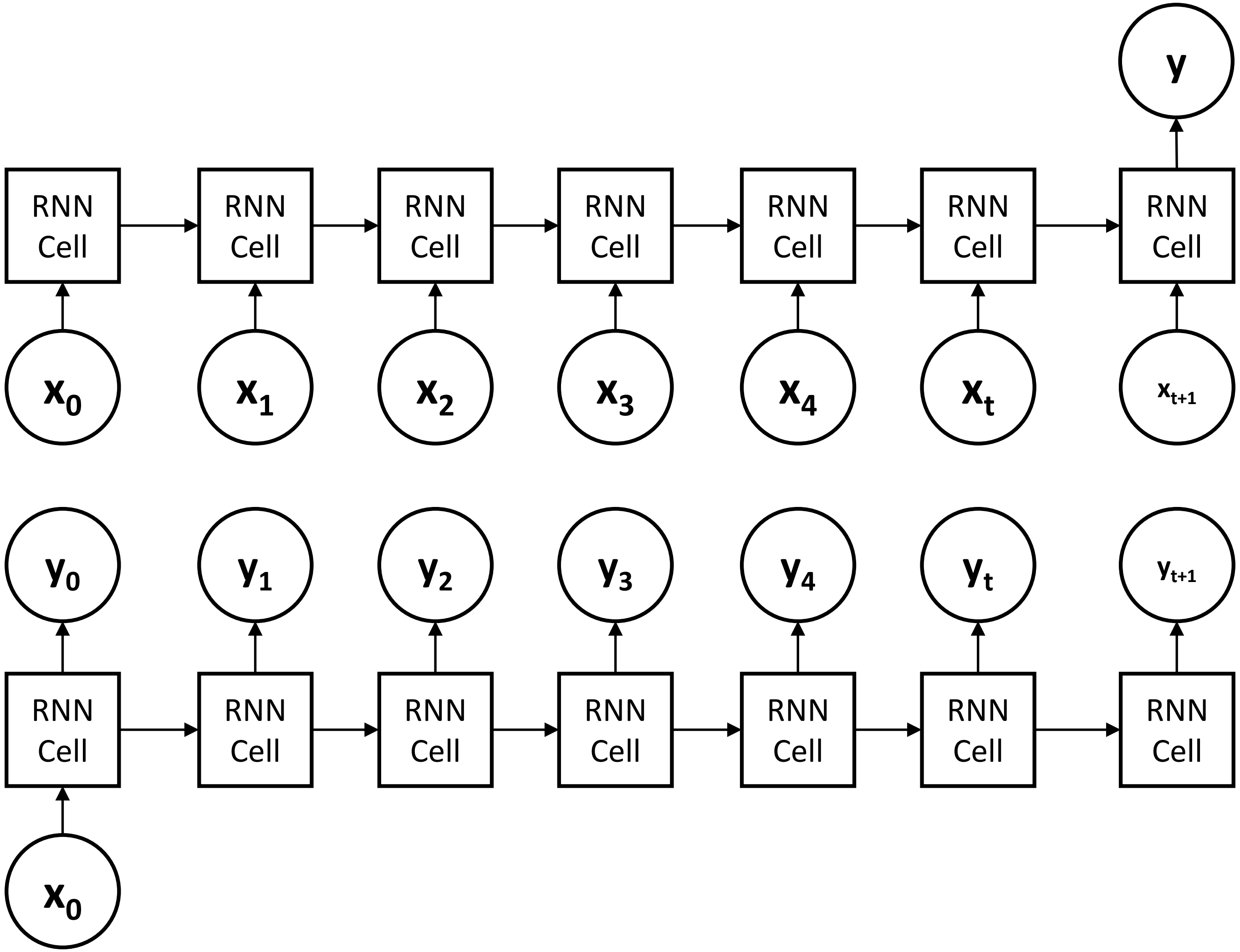
Video classification on frame level
Audio classification on frame level



Many to many

Machine translation
我很帥 → I am very handsome
Paraphrasing
我很帥 → 我很緣投 / 我超厲害

RNN



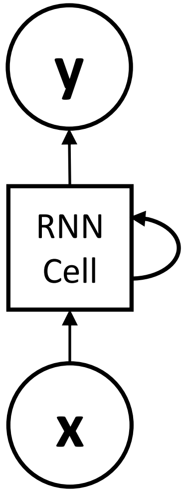
Many to one

Text classification
Word sequence → Class label

One to many

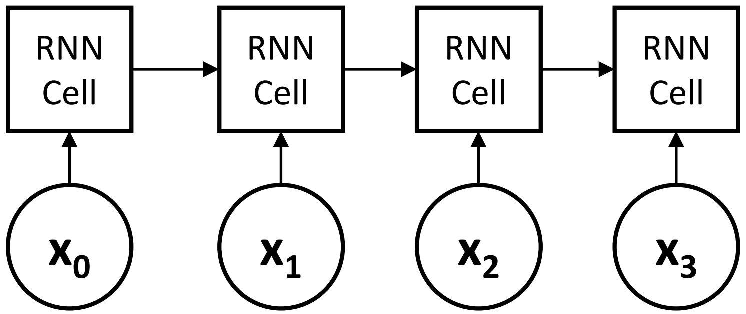
Image captioning
Image → Word sequence

RNN



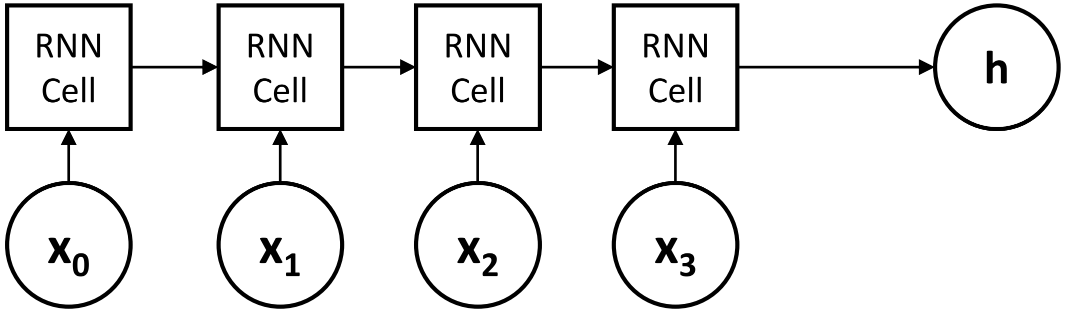
資管系 柯士文 George Ke

Basic Sequence-to-Sequence



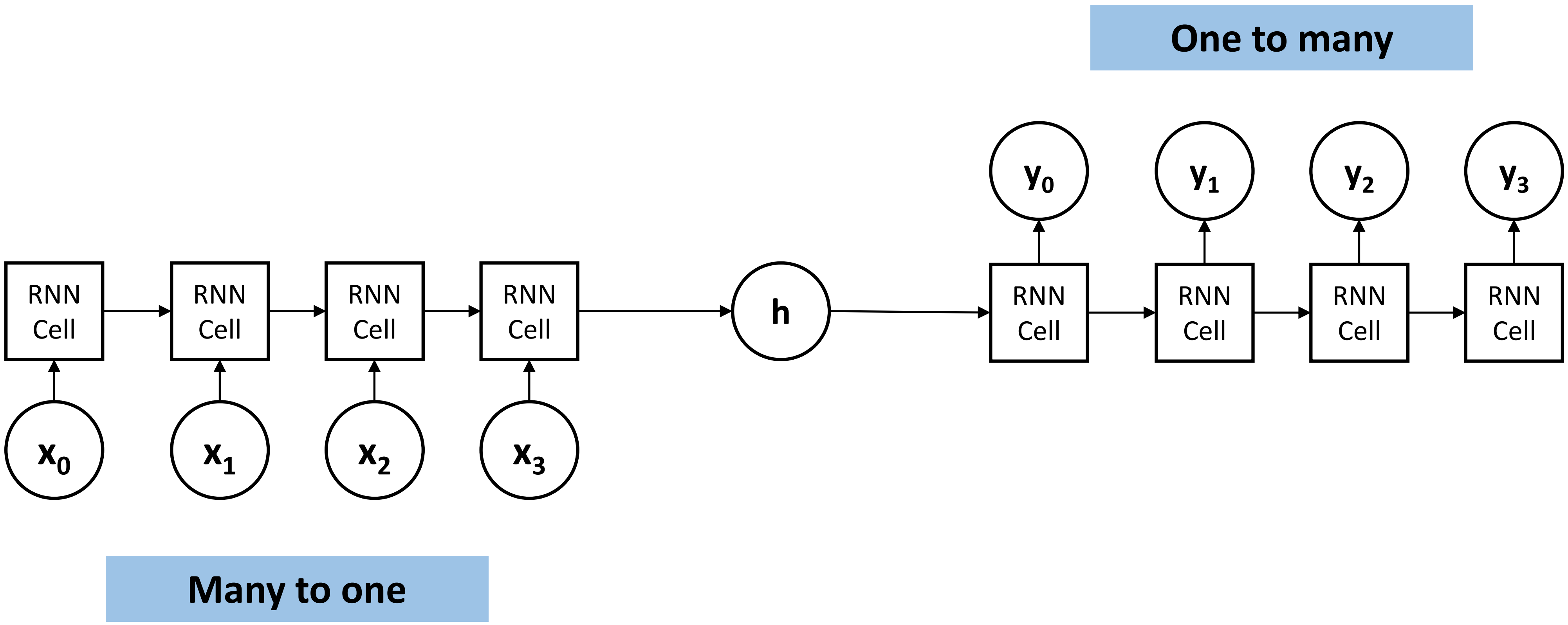
資管系 柯士文 George Ke

Basic Sequence-to-Sequence



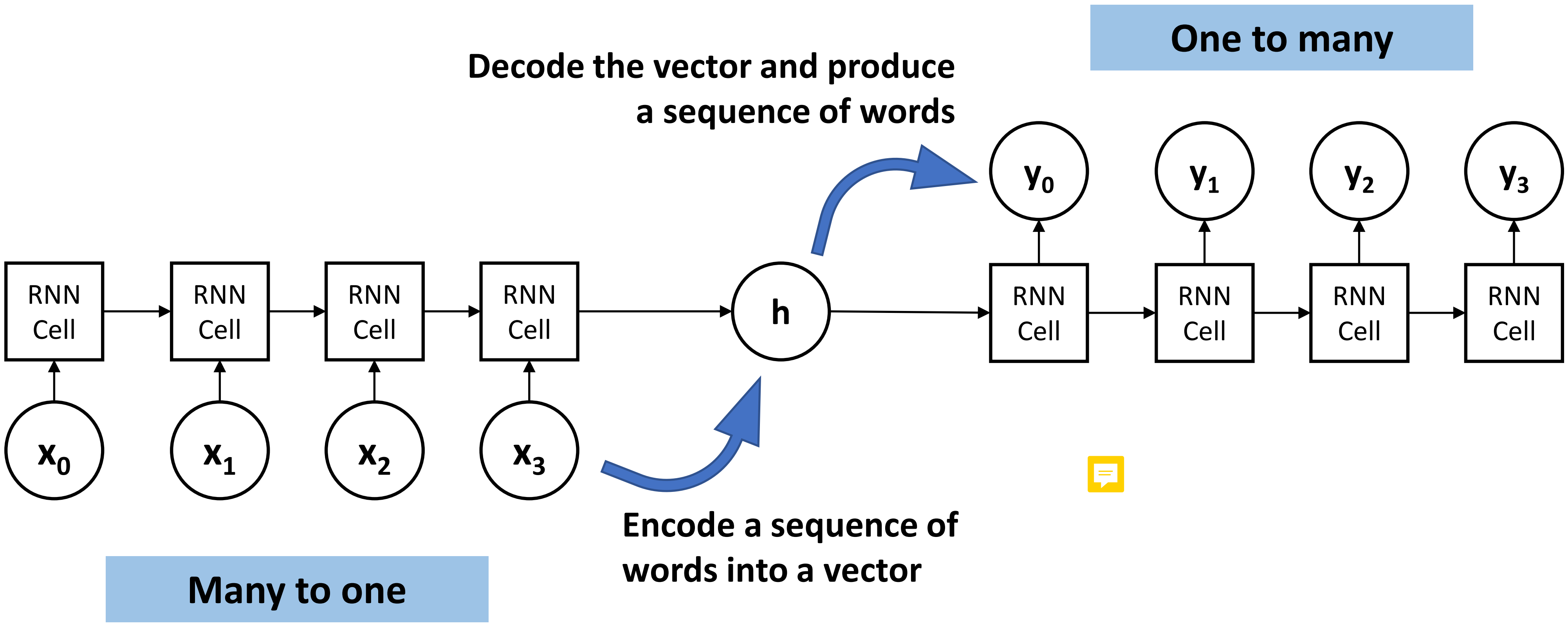
Many to one

Basic Sequence-to-Sequence



資管系 柯士文 George Ke

Basic Sequence-to-Sequence



資管系 柯士文 George Ke

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