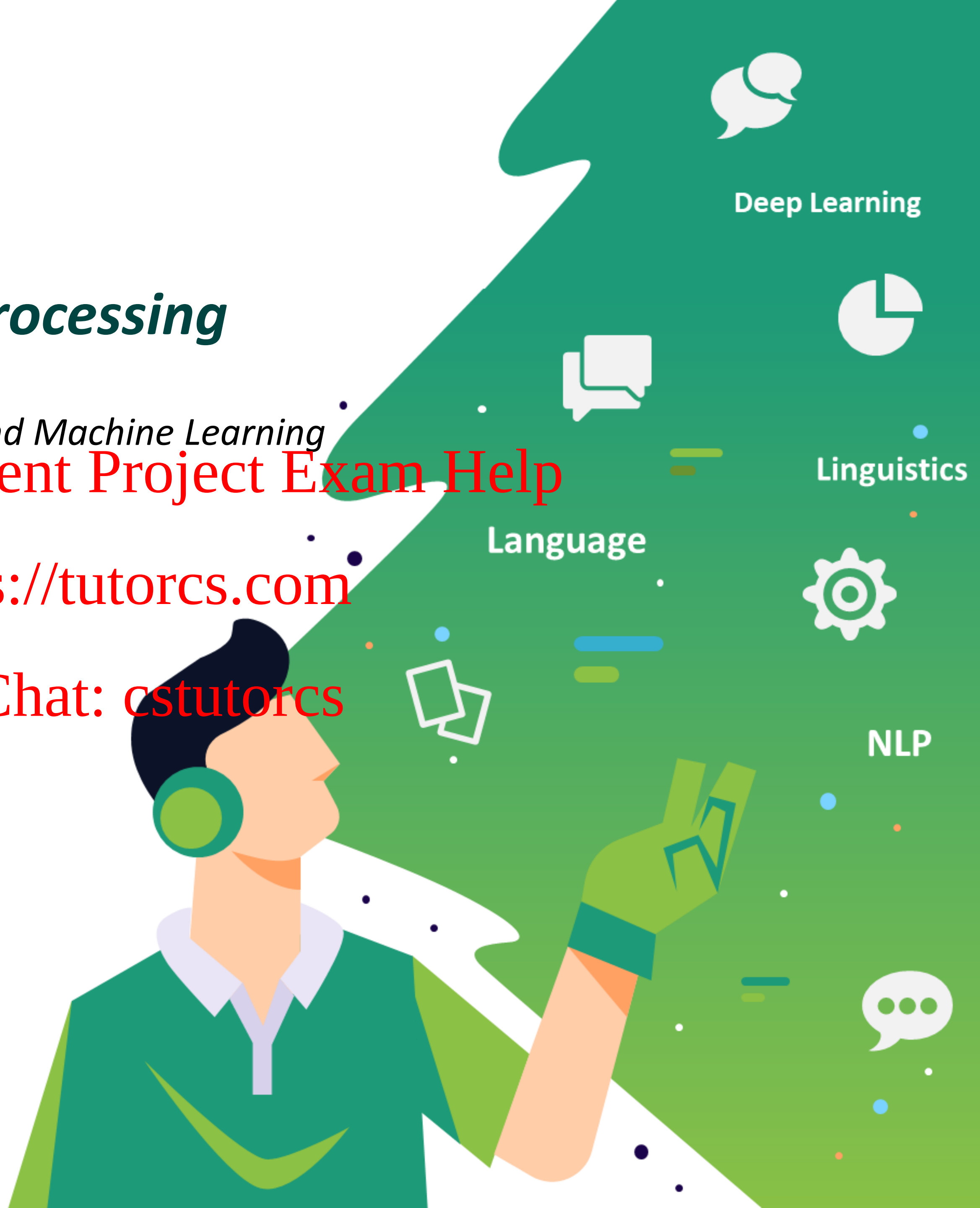
COMP5046

Natural Language Processing

Lecture 3: Word Classification and Machine Learning



Dr. Caren Han

Semester 1, 2021
School of Computer Science,
University of Sydney

0 LECTURE PLAN

Lecture 3: Word Classification and Machine Learning

1. **Previous Lecture: Word Embedding Review**
2. Word Embedding Evaluation
3. Deep Neural Network for Natural Language Processing
   1. Perceptron and Neural Network (NN)
   2. Multilayer Perceptron
   3. Applications
4. Next Week Preview

   See how the Deep Learning can be used for NLP

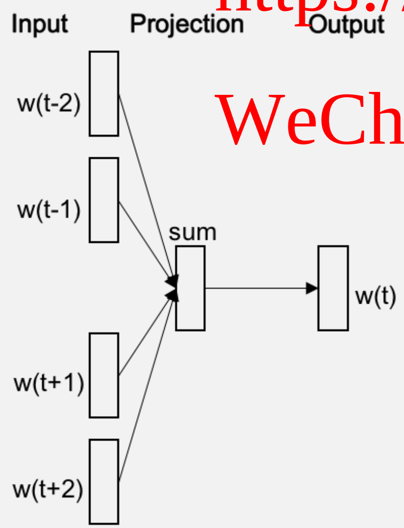
   - Text Classification, etc.

Previous Lecture Review

Word2Vec Models

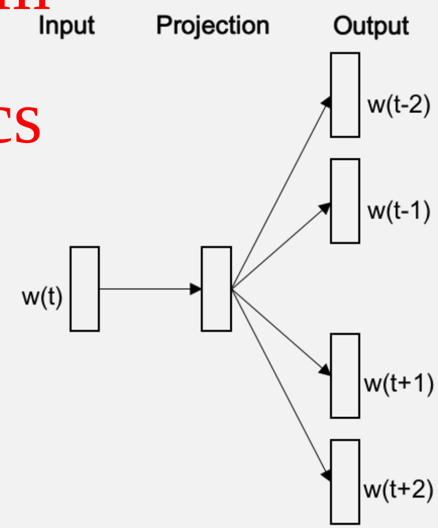
CBOW

Predict center word from (bag of) context words



Skip-gram

Predict context words given center word



Previous Lecture Review

1

Word2Vec with Continuous Bag of Words (CBOW)

Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"

Using window slicing, develop the training data

Center word	Context ("outside") word	
[1,0,0,0,0,0,0]	[0,1,0,0,0,0,0], [0,0,1,0,0,0,0]	Sydney is the state capital of NSW
[0,1,0,0,0,0,0]	[1,0,0,0,0,0,0], [0,0,1,0,0,0,0], [0,0,0,1,0,0,0]	Sydney is the state capital of NSW
[0,0,1,0,0,0,0]	[1,0,0,0,0,0,0], [0,1,0,0,0,0,0] [0,0,0,1,0,0,0], [0,0,0,0,1,0,0]	Sydney is the state capital of NSW
[0,0,0,1,0,0,0]	[0,1,0,0,0,0,0], [0,0,1,0,0,0,0] [0,0,0,0,1,0,0], [0,0,0,0,0,1,0]	Sydney is the state capital of NSW
[0,0,0,0,1,0,0]	[0,0,1,0,0,0,0], [0,0,0,1,0,0,0] [0,0,0,0,0,1,0], [0,0,0,0,0,0,1]	Sydney is the state capital of NSW
[0,0,0,0,0,1,0]	[0,0,0,1,0,0,0], [0,0,0,0,1,0,0] [0,0,0,0,0,0,1]	Sydney is the state capital of NSW
[0,0,0,0,0,0,1]	[0,0,0,0,1,0,0], [0,0,0,0,0,1,0]	Sydney is the state capital of NSW

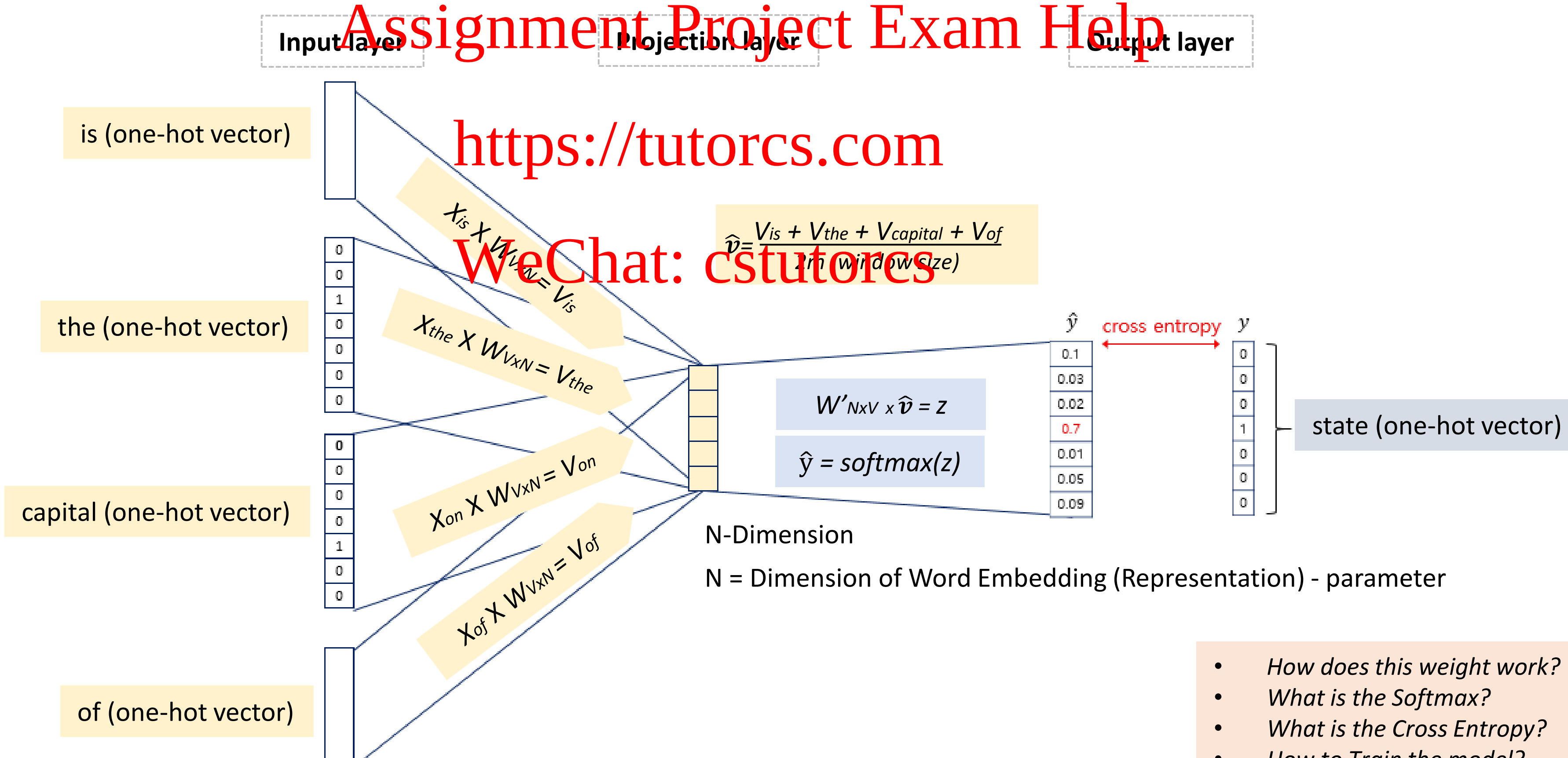
Center word

Context ("outside") word

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"



N = Dimension of Word Embedding (Representation) - parameter

- *How does this weight work?*
- *What is the Softmax?*
- *What is the Cross Entropy?*
- *How to Train the model?*

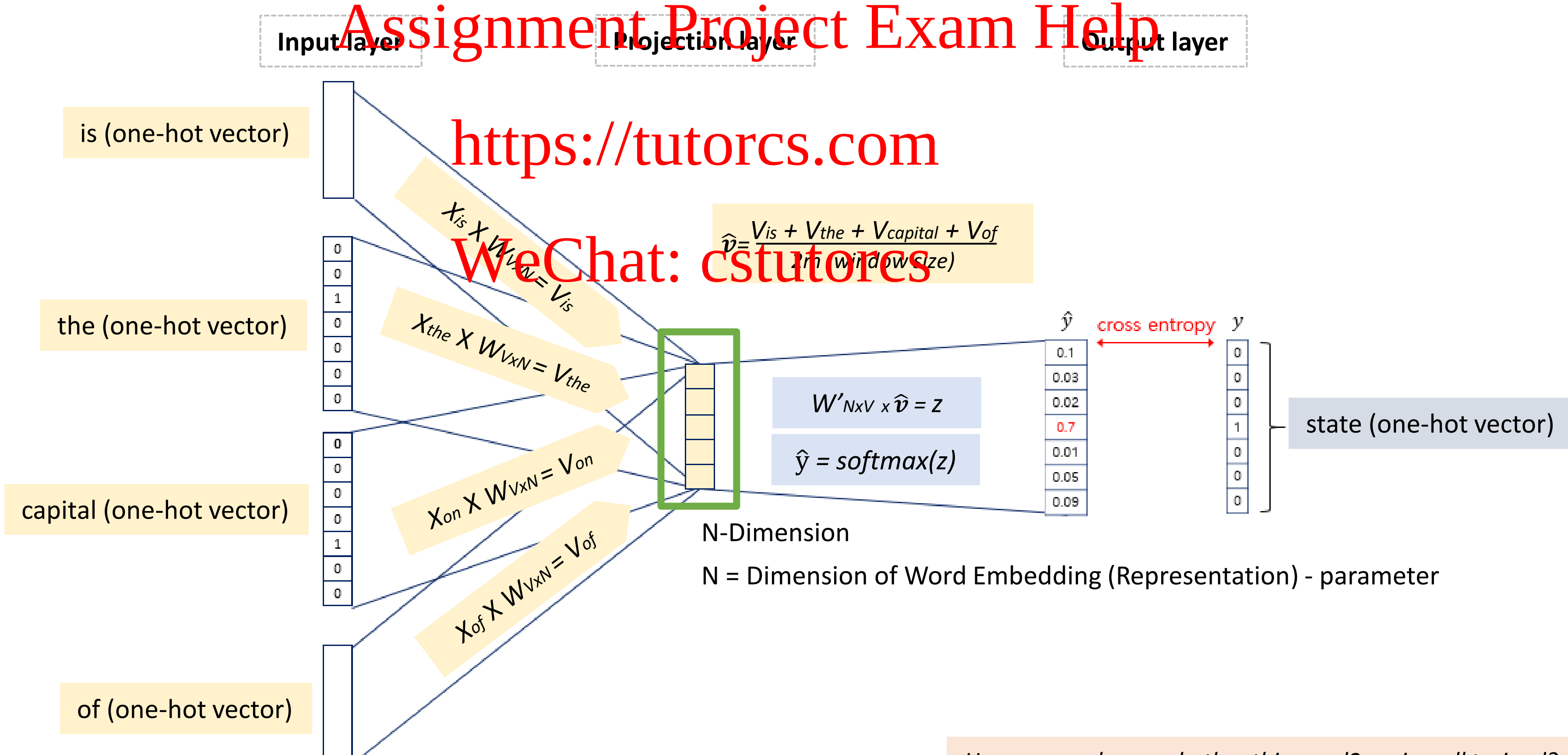
1

Previous Lecture Review

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"



How can we know whether this word2vec is well trained?

0 LECTURE PLAN

Lecture 3: Word Classification and Machine Learning

1. Previous Lecture: Word Embedding Review
2. **Word Embedding Evaluation**
3. Deep Neural Network for Natural Language Processing
   1. Perceptron and Neural Network (NN)
   2. Multilayer Perceptron
   3. Applications
4. Next Week Preview

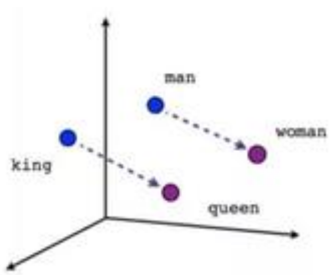
   See how the Deep Learning can be used for NLP

   - Text Classification, etc.

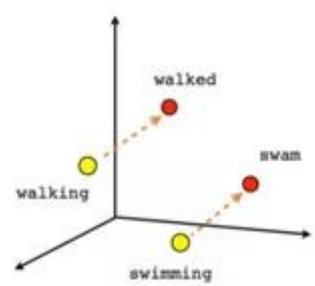
2 Word Embedding Evaluation

How to evaluate word vectors?

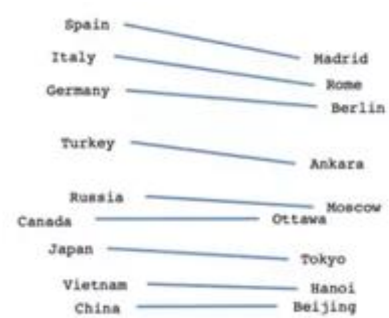
Type	How to work / Benefit
Intrinsic	Evaluation on a specific/intermediate subtask • Fast to compute • Helps to understand that system • Not clear if really helpful unless correlation to real task is established
Extrinsic	Evaluation on a real task • Can take a long time to compute accuracy • Unclear if the subsystem is the problem or its interaction or other subsystems



Male-Female



Verb tense



Country-Capital

2 Word Embedding Evaluation

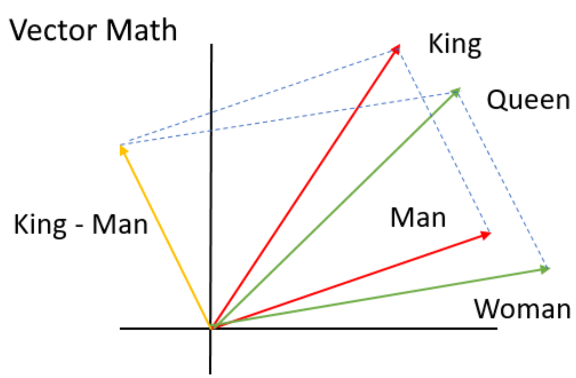
Intrinsic word vector evaluation

Word Vector Analogies

$$a \leftrightarrow b :: c \leftrightarrow ???$$

$$man \leftrightarrow women :: king \leftrightarrow ???$$

- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions



Word Embedding Evaluation

Intrinsic word vector evaluation

Word Vector Analogies

King – Man + Woman = ?

No	Training Dataset	Type	Result
1	TED Script	word2vec CBOW	President
2		word2vec Skip-gram	Luther
3		fastText CBOW	Kidding
4		fastText Skip-gram	Jarring
5	Google News	word2vec CBOW	queen
6		word2vec Skip-gram	queen

2 Word Embedding Evaluation

Intrinsic word vector evaluation

Evaluation Result Comparison

The Semantic-Syntactic word relationship tests for understanding of a wide variety of relationships as shown below.

Using 640-dimensional word vectors, a skip-gram trained model achieved 55% semantic accuracy and 59% syntactic accuracy.

Table 3: *Comparison of architectures using models trained on the same data, with 640-dimensional word vectors. The accuracies are reported on our Semantic-Syntactic Word Relationship test set, and on the syntactic relationship test set of [20]*

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

(Original Word2vec Paper - Mikolov et al.2013)

2 **Word Embedding Evaluation**

Intrinsic word vector evaluation

Evaluation Result Comparison

The Semantic-Syntactic word relationship tests for understanding of a wide variety of relationships as shown below.

Table 2: Results on the word analogy task, given as percent accuracy. Underlined scores are best within groups of similarly-sized models; bold scores are best overall. HPCA vectors are publicly available[2]; (i)vLBL results are from (Mnih et al., 2013); skip-gram (SG) and CBOW results are from (Mikolov et al., 2013a,b); we trained SG† and CBOW† using the `word2vec` tool[3]. See text for details and a description of the SVD models.

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

(Original Glove Paper - Pennington et al.2014)

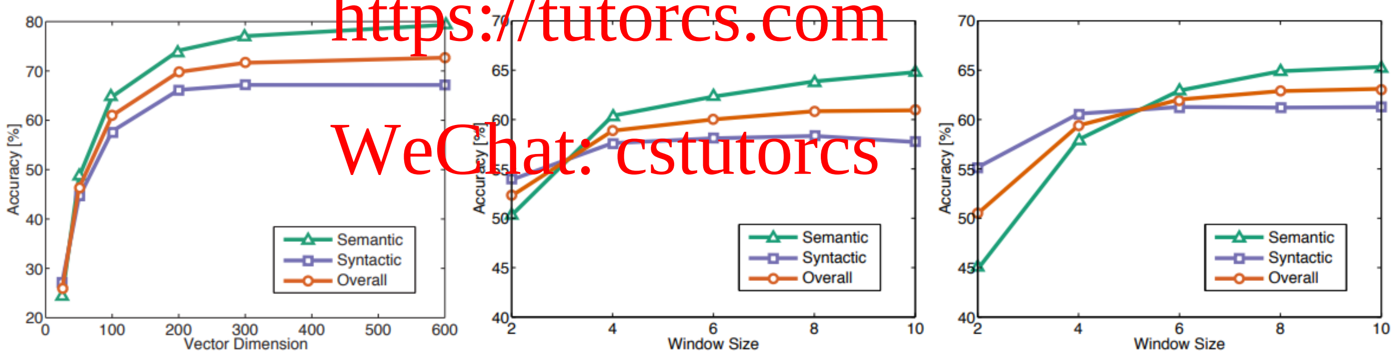
Word Embedding Evaluation

Intrinsic word vector evaluation

Evaluation Result Comparison

The Semantic-Syntactic word relationship tests for understanding of a wide variety of relationships as shown below.

Window-Size (m) and Vector Dimension (N)



(a) Symmetric context (b) Symmetric context (c) Asymmetric context

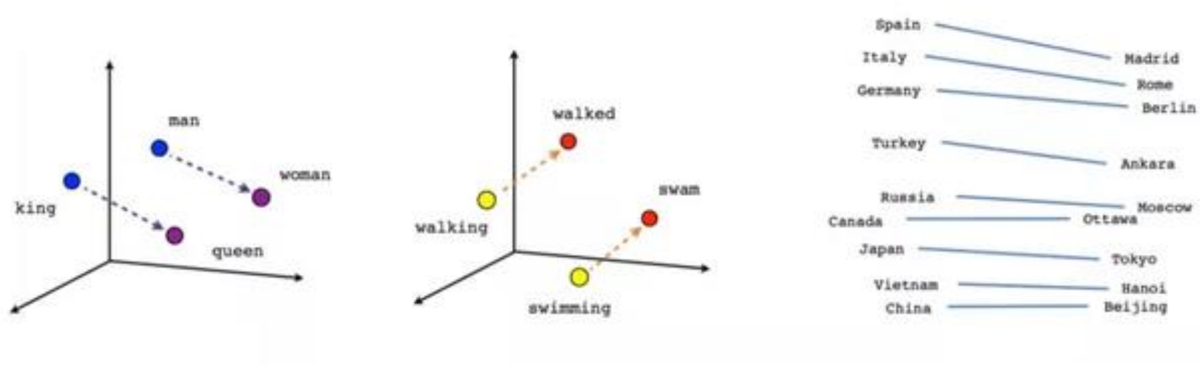
Figure 2: Accuracy on the analogy task as function of vector size and window size/type. All models are trained on the 6 billion token corpus. In (a), the window size is 10. In (b) and (c), the vector size is 100.

(Original Glove Paper - Pennington et al.2014)

Word Embedding Evaluation

How to evaluate word vectors?

Type	How to work / Benefit
Intrinsic	Evaluation on a specific/intermediate subtask • Fast to compute • Helps to understand that system • Not clear if really helpful unless correlation to real task is established
Extrinsic	**Evaluation on a real task** • **Can take a long time to compute accuracy** • **Unclear if the subsystem is the problem or its interaction or other subsystems**



Male-Female Verb tense Country-Capital

0

LECTURE PLAN

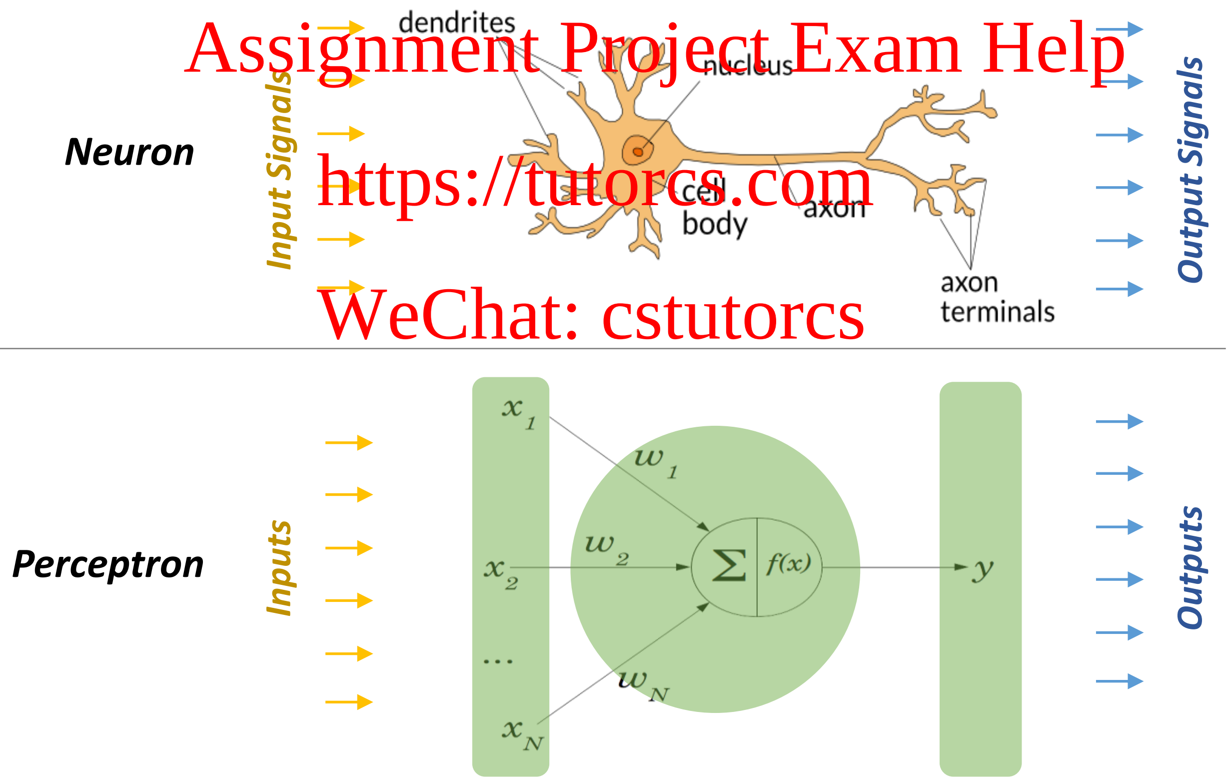
Lecture 3: Word Classification and Machine Learning

1. Previous Lecture: Word Embedding Review
2. Word Embedding Evaluation
3. **Deep Neural Network for Natural Language Processing**
    1. Perceptron and Neural Network (NN)
    2. Multilayer Perceptron
    3. Applications
4. Next Week Preview

    See how the Deep Learning can be used for NLP
    - Text Classification, etc.

Deep Learning with Neural Network

Neuron and Perceptron

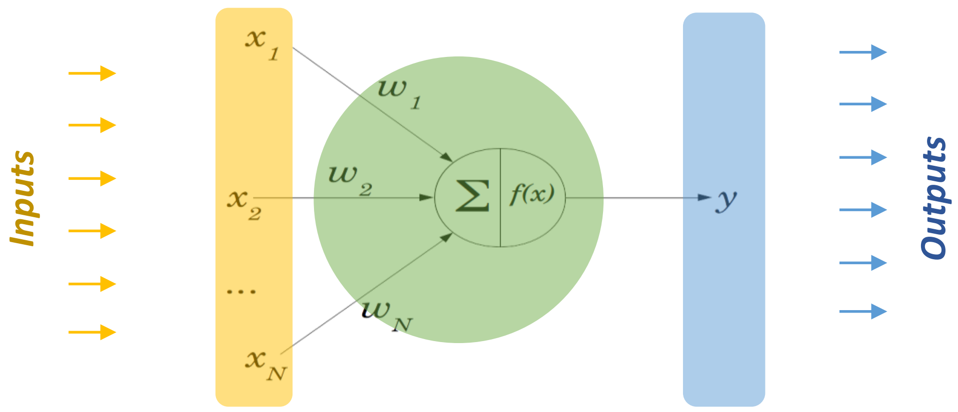


Deep Learning with Neural Network

Inputs and Outputs (Labels) for Natural Language Processing

x_i	Inputs	**Features** words (indices or vectors!), context windows, sentences, documents, etc.
y_i	Outputs (labels)	**What we try to predict/classify** • E.g. word meaning, sentiment, name entity

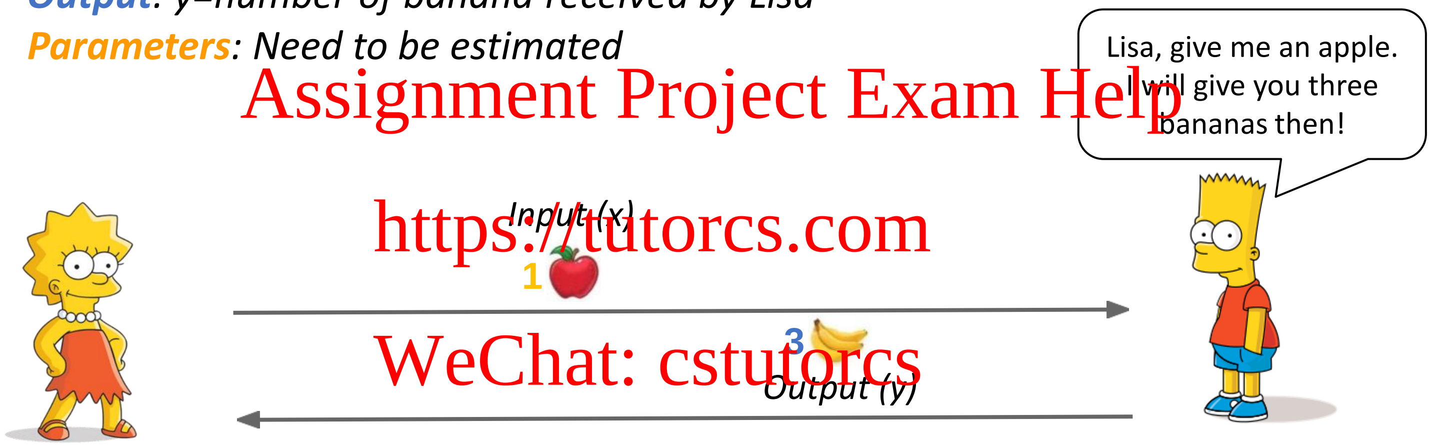
Perceptron



3 Deep Learning for NLP

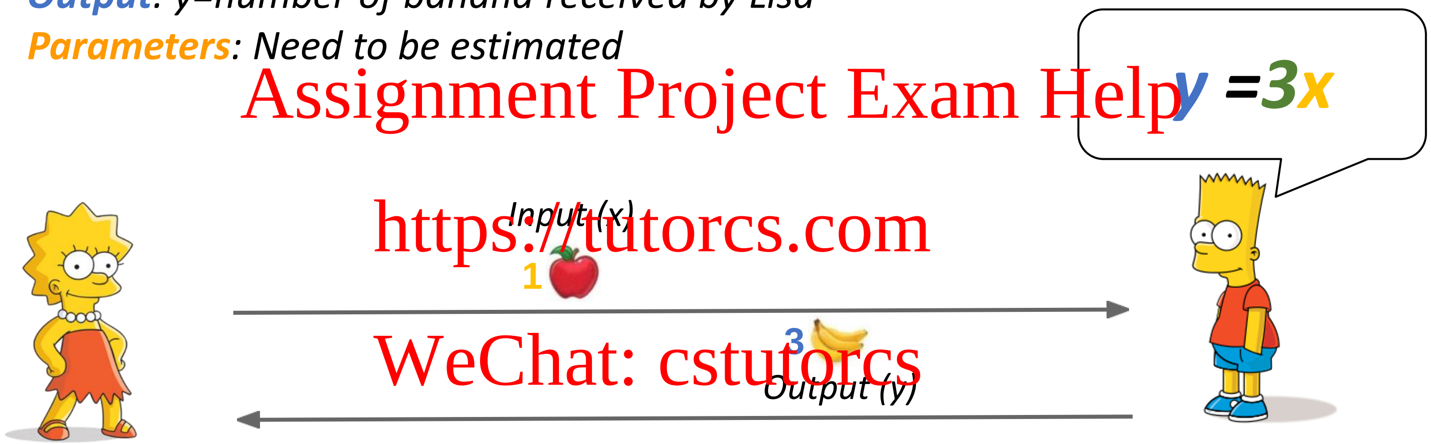
Deep Learning with Neural Network

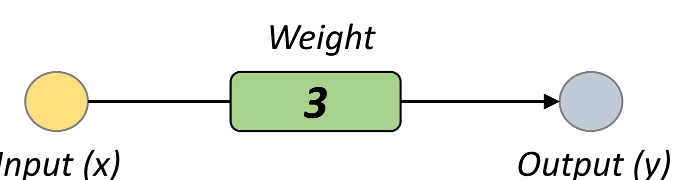
Input: *x=number of apple given by Lisa*
Output: *y=number of banana received by Lisa*
Parameters: *Need to be estimated*



Deep Learning with Neural Network - Model

Input: *x=number of apple given by Lisa*
Output: *y=number of banana received by Lisa*
Parameters: *Need to be estimated*





$$y = W x$$

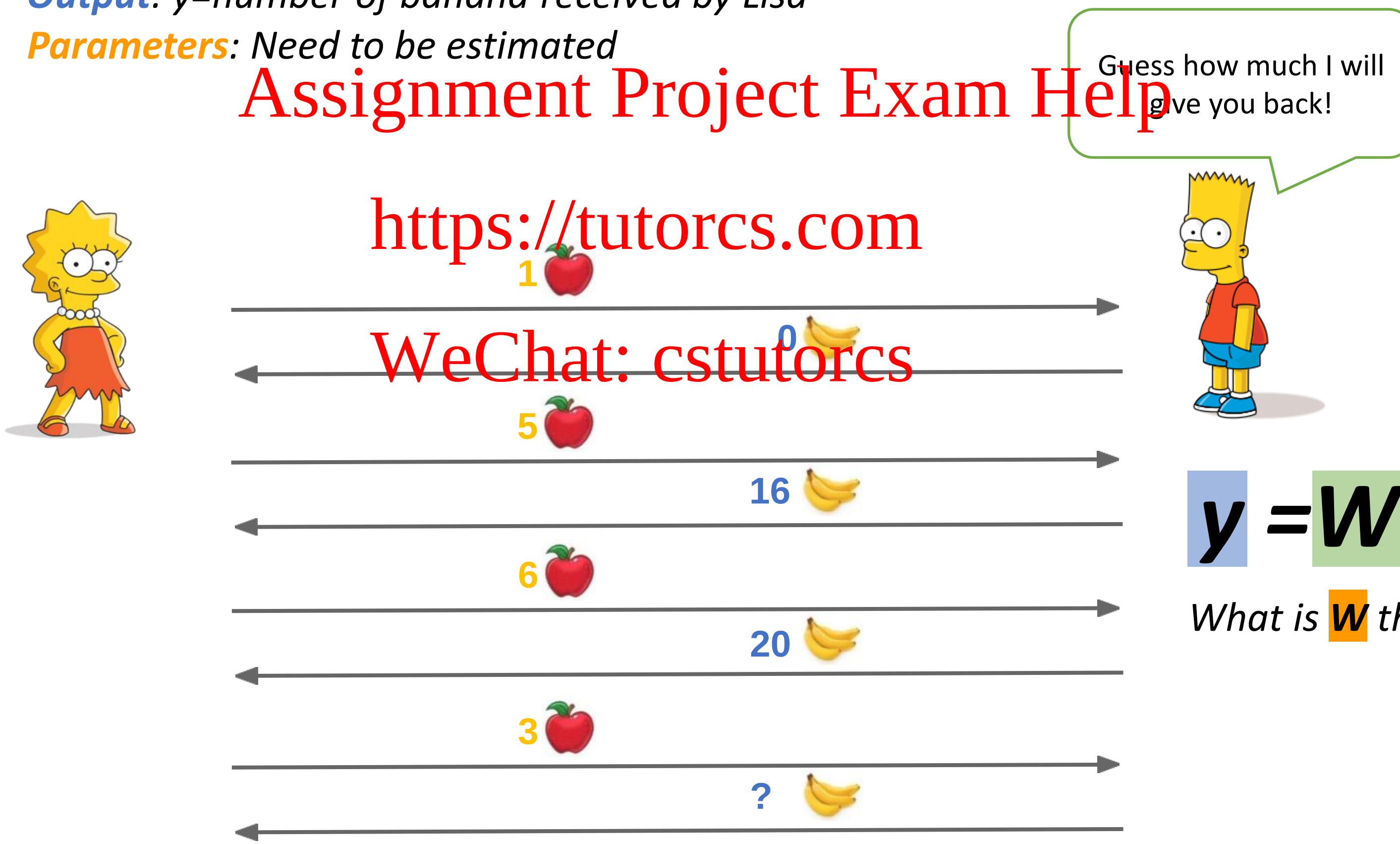
3 Deep Learning for NLP

Deep Learning with Neural Network - Model

Input: x=number of apple given by Lisa
Output: y=number of banana received by Lisa
Parameters: Need to be estimated

Deep Learning with Neural Network - Parameter

Input: *x=number of apple given by Lisa*
Output: *y=number of banana received by Lisa*
Parameters: *Need to be estimated*

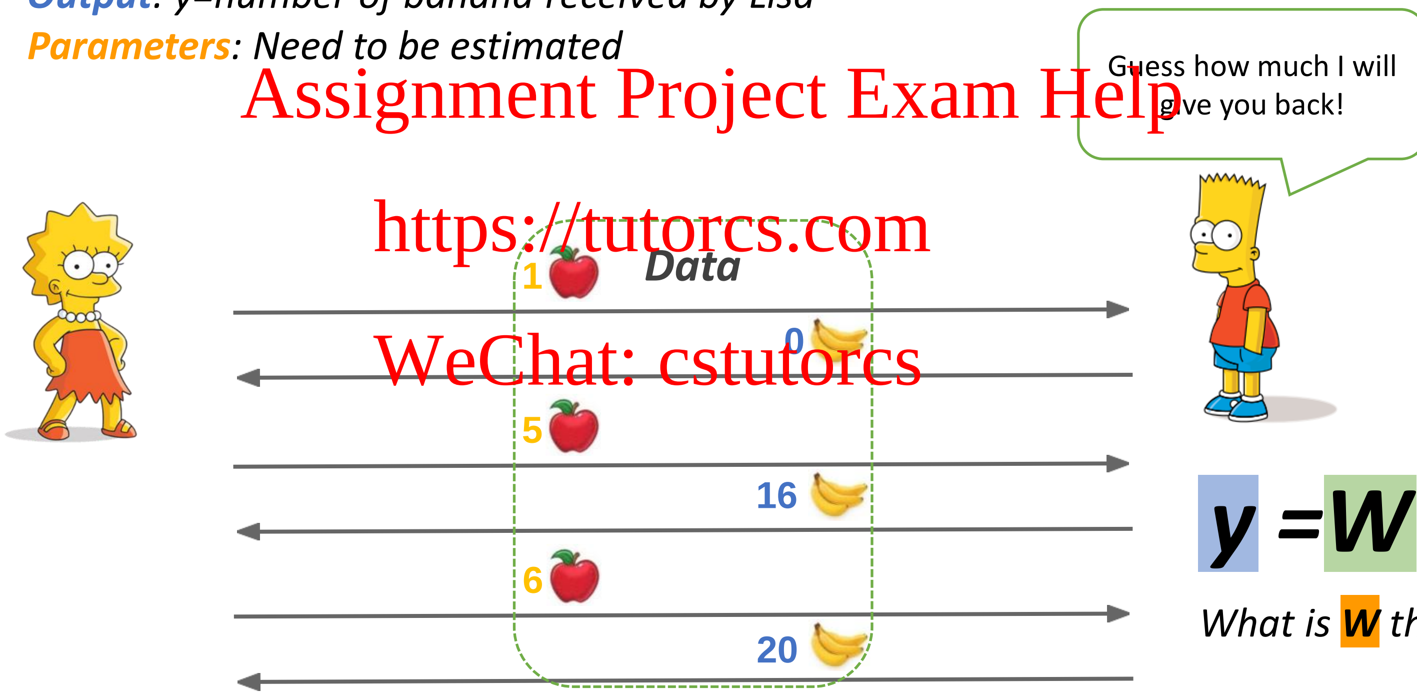


$$y = W x$$

What is ***W*** *then?*

Deep Learning with Neural Network - Parameter

Input: x=number of apple given by Lisa
Output: y=number of banana received by Lisa
Parameters: Need to be estimated



$y = W x$

*What is **W** then?*

Deep Learning with Neural Network - Parameter

Input: *x=number of apple given by Lisa*
Output: *y=number of banana received by Lisa*
Parameters: *Need to be estimated*

Data

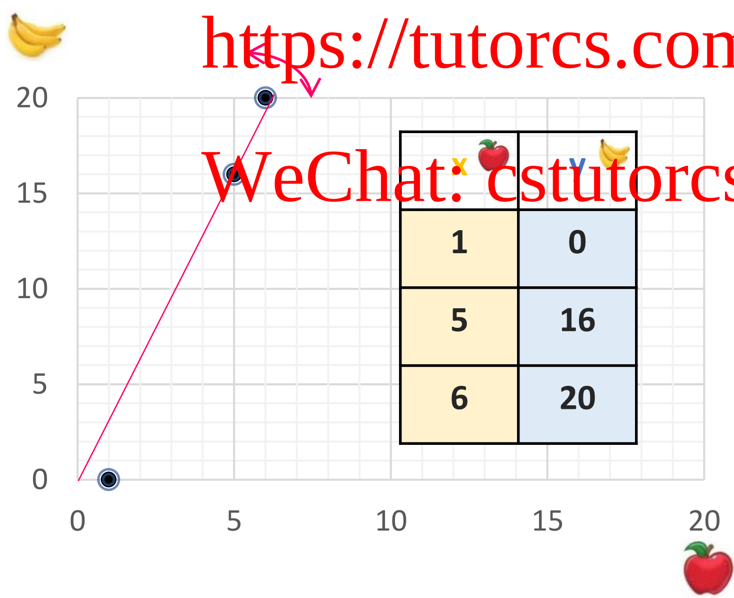
x	y
1	0
5	16
6	20

$$y = Wx$$

What is ***W*** *then?*

Deep Learning with Neural Network - Parameter

Input: x=number of apple given by Lisa
Output: y=number of banana received by Lisa
Parameters: Need to be estimated



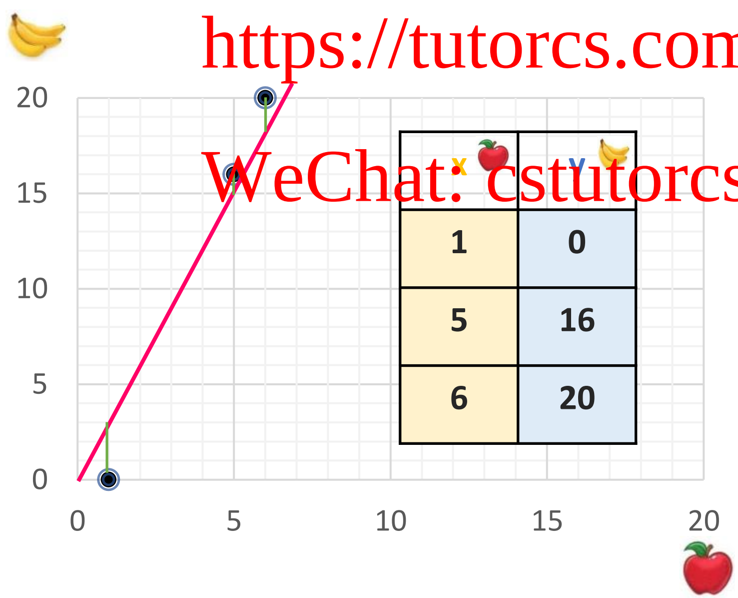
x	y
1	0
5	16
6	20

$$y = W x$$

*What is **W** then?*

Deep Learning with Neural Network - Parameter

Input: x=number of apple given by Lisa
Output: y=number of banana received by Lisa
Parameters: Need to be estimated



x 🍎	y 🍌
1	0
5	16
6	20

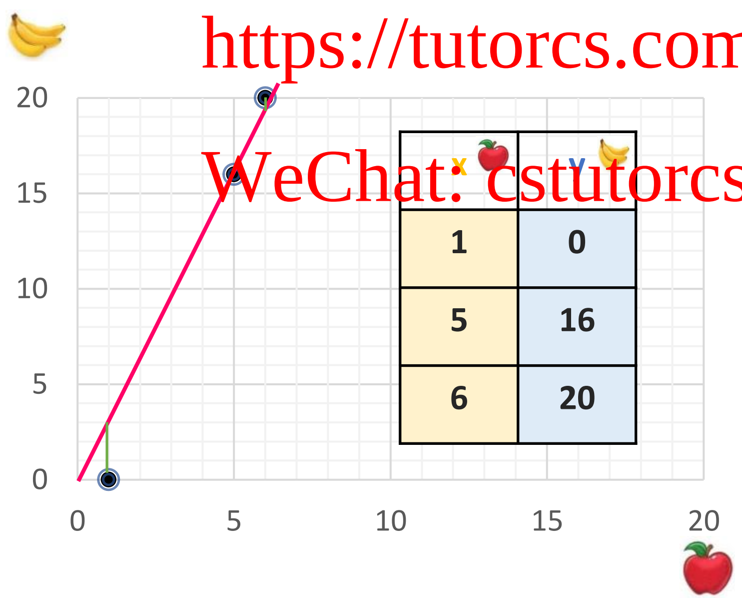
$$y = W x$$

*What if W is **3**?*

3 = 3 x 1
15 = 3 x 5
20 = 3 x 6

Deep Learning with Neural Network - Parameter

Input: x=number of apple given by Lisa
Output: y=number of banana received by Lisa
Parameters: Need to be estimated



x	y
1	0
5	16
6	20

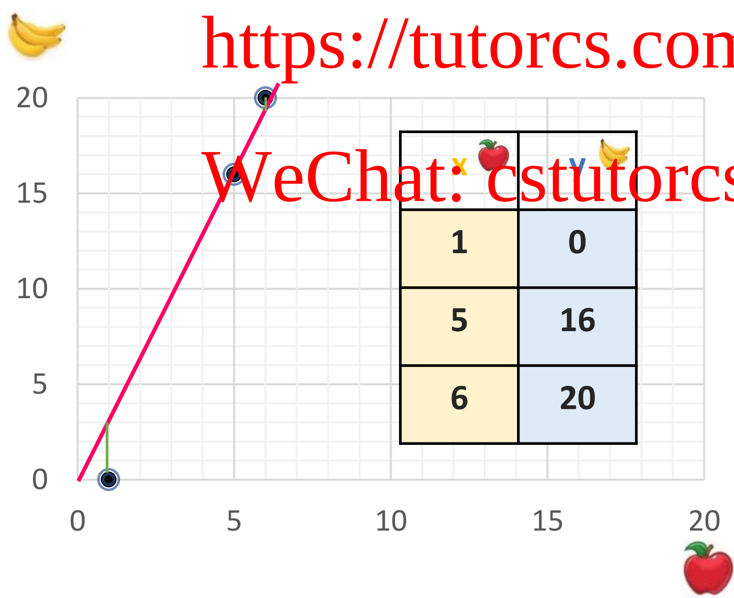
$$y = W x$$

What if W is ***3.2****?*

3.2 = 3.2 x 1
16 = 3.2 x 5
19.2 = 3.2 x 6

Deep Learning with Neural Network - Parameter

Input: *x=number of apple given by Lisa*
Output: *y=number of banana received by Lisa*
Parameters: *Need to be estimated*



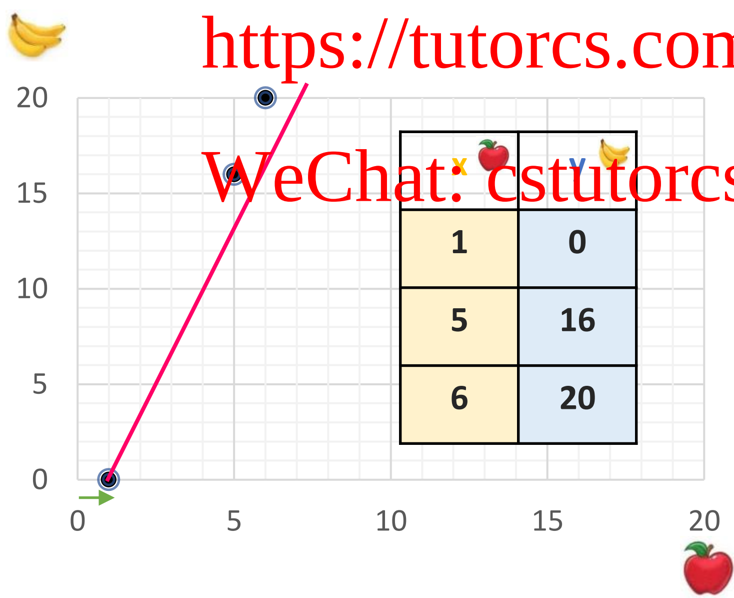
x	y
1	0
5	16
6	20

$$y = \underbrace{W}_{\text{weight}} x + \underbrace{b}_{\text{bias}}$$

Weight is not enough…

Deep Learning with Neural Network - Parameter

Input: x=number of apple given by Lisa
Output: y=number of banana received by Lisa
Parameters: Need to be estimated



x	y
1	0
5	16
6	20

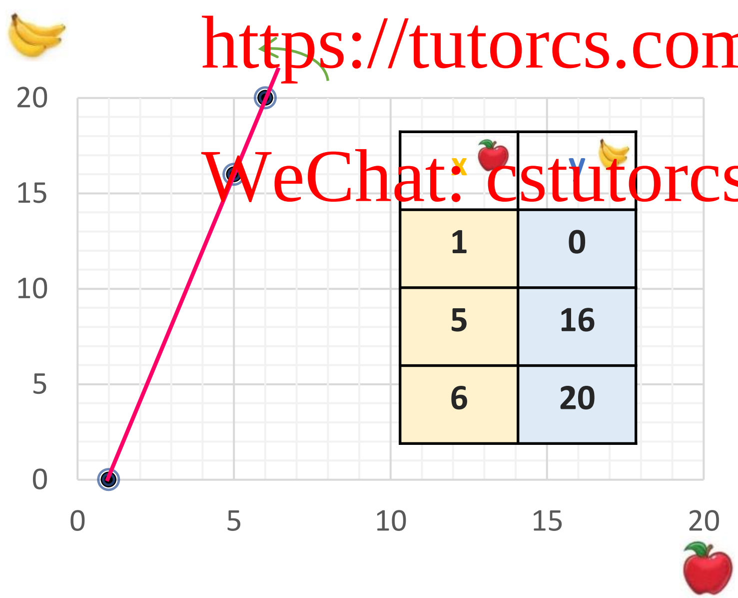
weight, bias

$$y = W x + b$$

How can we find the parameters, ***w*** *and* ***b****?*

Deep Learning with Neural Network - Parameter

Input: x=number of apple given by Lisa
Output: y=number of banana received by Lisa
Parameters: Need to be estimated



x 🍎	y 🍌
1	0
5	16
6	20

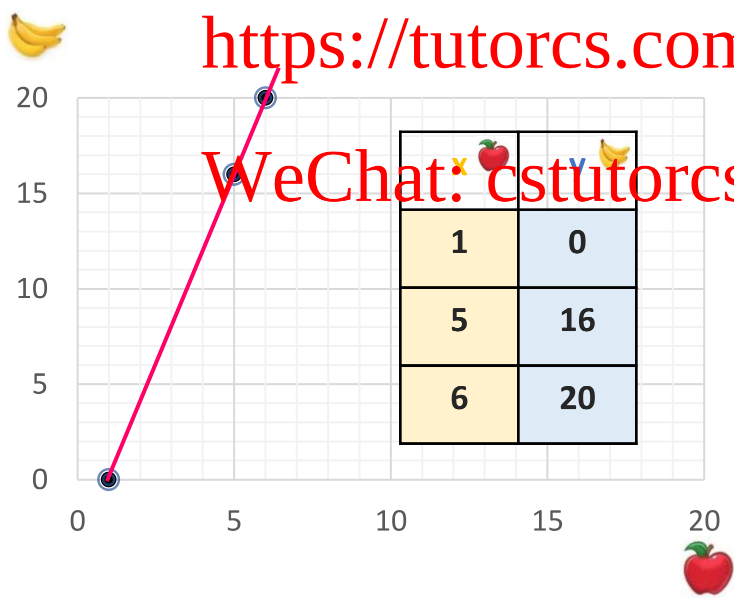
$$y = \underbrace{W}_{\text{weight}} x + \underbrace{b}_{\text{bias}}$$

*How can we find the parameters, **w** and **b**?*

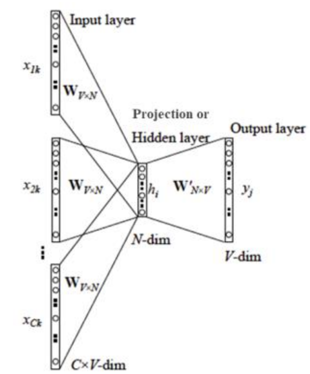
THE UNIVERSITY OF SYDNEY

Deep Learning with Neural Network - Parameter

Input: *x=number of apple given by Lisa*
Output: *y=number of banana received by Lisa*
Parameters: *Need to be estimated*



x	y
1	0
5	16
6	20



Model $y = \overbrace{W}^{weight} x + \overbrace{b}^{bias}$

*How can we find the parameters, **w** and **b**?*

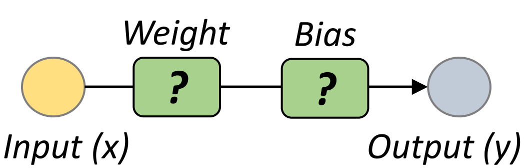
3 Deep Learning for NLP

Deep Learning with Neural Network - Cost

Actual Data

$$\underbrace{y}_{} = \overset{\text{weight}}{?}\, x + \overset{\text{bias}}{?}$$

x	y
1	0
5	16
6	20



Model Ex#1

$$\hat{y} = \overset{\text{weight}}{1}\, x + \overset{\text{bias}}{0}$$

x	$\hat{y}$ (predicted)	y (actual)
1	1	0
5	5	16
6	6	20

Model Ex#2

$$\hat{y} = \overset{\text{weight}}{2}\, x + \overset{\text{bias}}{2}$$

x	$\hat{y}$ (predicted)	y (actual)
1	4	0
5	12	16
6	14	20

Which one is closer?

3 Deep Learning for NLP

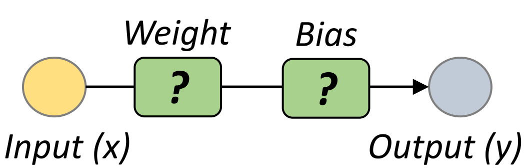
Deep Learning with Neural Network – Cost (loss)

Actual Data

weight bias

$$y = ?\,x + ?$$

x	y
1	0
5	16
6	20



Model Ex#1

weight bias

$$\hat{y} = 1\,x + 0$$

x	ŷ (predicted)	y (actual)	$(y-\hat{y})^2$ (cost)
1	1	0	
5	5	16	
6	6	20	

Model Ex#2

weight bias

$$\hat{y} = 2\,x + 2$$

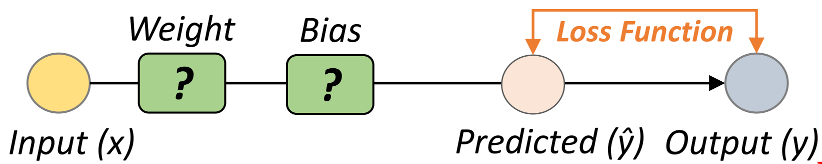
x	ŷ (predicted)	y (actual)	$(y-\hat{y})^2$ (cost)
1	4	0	
5	12	16	
6	14	20	

Let's calculate the cost(loss)!

Mean Squared Error (MSE) $$C(w,b) = \sum(y_n-\hat{y}_n)$$

$n\in\{0,1,2\}$

WAIT! Loss Function? Cost Calculation?

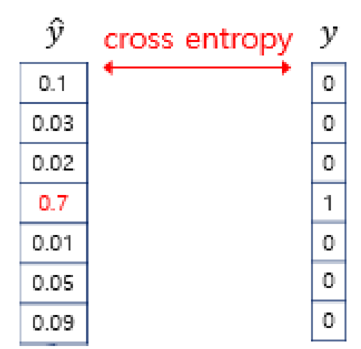


1) Mean Squared Error (MSE): measures the average of the squares of the errors

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

2) Cross Entropy: calculating the difference between two probability distributions

$$L_{\text{cross-entropy}}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i y_i \log(\hat{y}_i)$$



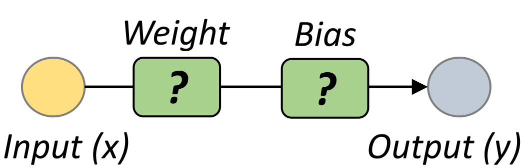
Deep Learning with Neural Network - Cost (loss)

Actual Data

weight bias

$y = ?\ x + ?$

x 🍎	y 🍌
1	0
5	16
6	20



Model Ex#1

weight bias

$\hat{y} = 1\ x + 0$

x 🍎	predicted $\hat{y}$	actual y 🍌	cost $(y-\hat{y})^2$
1	1	0	1
5	5	16	121
6	6	20	196

C(1,0) = 318

Model Ex#2

weight bias

$\hat{y} = 2\ x + 2$

x 🍎	predicted $\hat{y}$	actual y 🍌	cost $(y-\hat{y})^2$
1	4	0	16
5	12	16	16
6	14	20	36

C(2,2) = 68 Winner

😎 ***Let's calculate the cost!***

$$C(w,b) = \sum(y_n - \hat{y}_n)$$

$n \in \{0,1,2\}$

3 Deep Learning for NLP

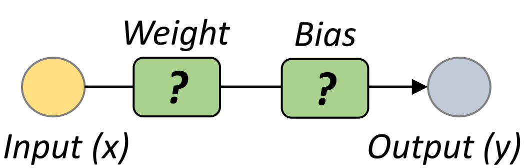
Deep Learning with Neural Network - Cost (loss)

Actual Data

weight *bias*

$y = ?\ x + ?$

x	y
1	0
5	16
6	20



Model Ex#1

weight *bias*

$\hat{y} = 1\ x + 0$

x	$\hat{y}$ (predicted)	y (actual)	$(y-\hat{y})^2$
1	1	0	1
5	5	16	121
6	6	20	196

C(1,0) = 318

Model Ex#2

weight *bias*

$\hat{y} = 2\ x + 2$

x	$\hat{y}$ (predicted)	y (actual)	$(y-\hat{y})^2$
1	4	0	16
5	12	16	16
6	14	20	36

C(2,2) = 68 Winner

Let's calculate the costs and get the lowest one!

arg min C(w,b)

w,b∈[-∞,∞]

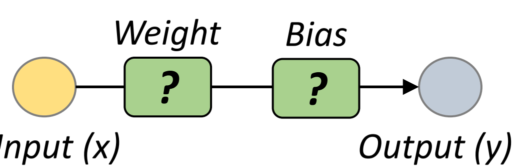
Deep Learning with Neural Network - Optimizer

Actual Data

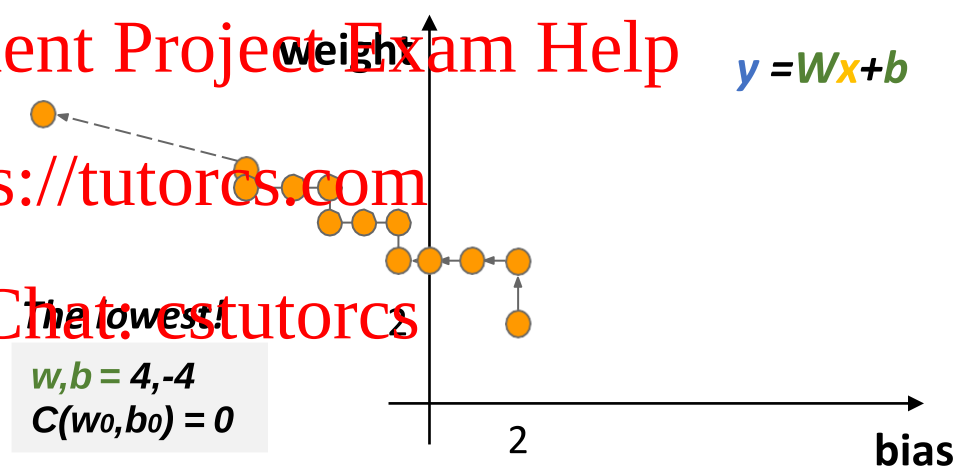
weight *bias*

$$y = ?\ x + ?$$

x 🍎	y 🍌
1	0
5	16
6	20



$y = Wx + b$



😎 ***Let's calculate the costs and get the lowest one!***

$$\underset{w,b \in [-\infty,\infty]}{\arg\min}\ C(w,b)$$

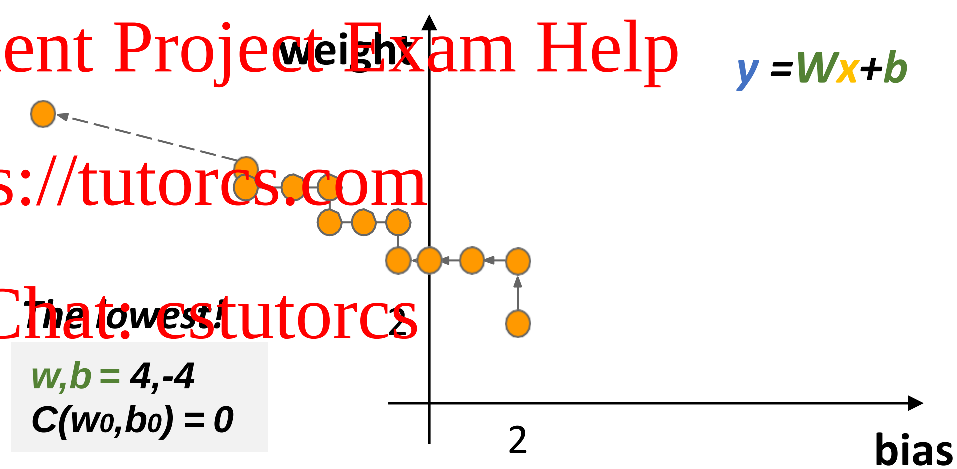
Deep Learning with Neural Network - Optimizer

Backpropagation (weight update)

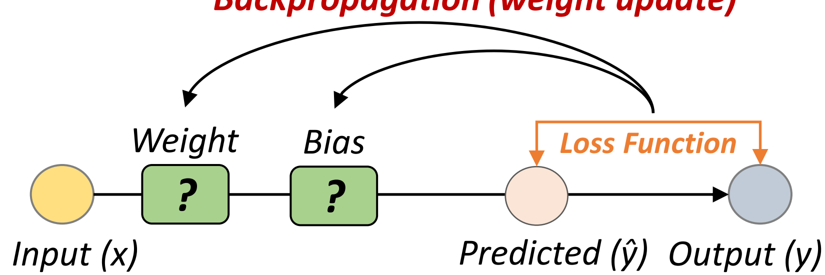
weight bias

$$y = 4x - 4$$

x 🍎	y 🍌
1	0
5	16
6	20



$y = Wx + b$



$$\arg\min C(w,b)$$

$w,b \in [-\infty, \infty]$

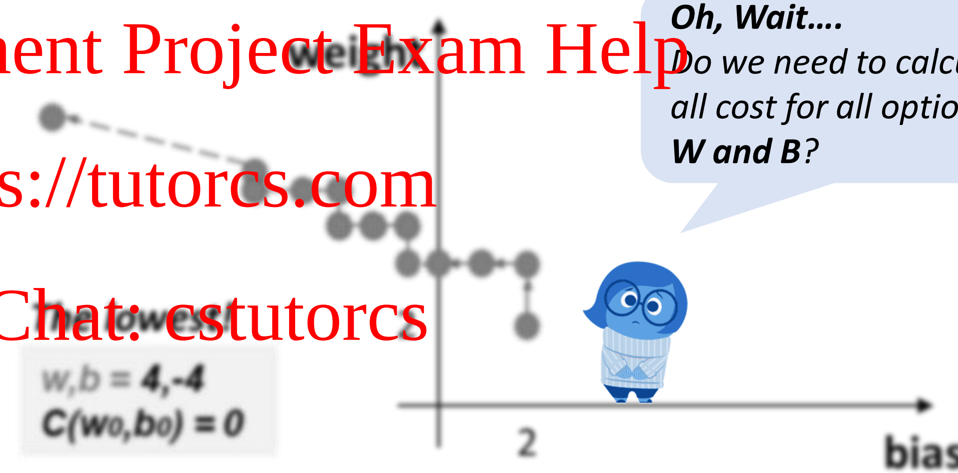
Backpropagation (weight update)

Deep Learning with Neural Network - Optimizer

weight *bias*

$$y = 4\,x - 4$$

x 🍎	y 🍌
1	0
5	16
6	20



Oh, Wait….
*Do we need to calculate all cost for all options of **W and B**?*

Expensive to compute (hours or days)

arg min C(w,b)

w,b∈[-∞,∞]

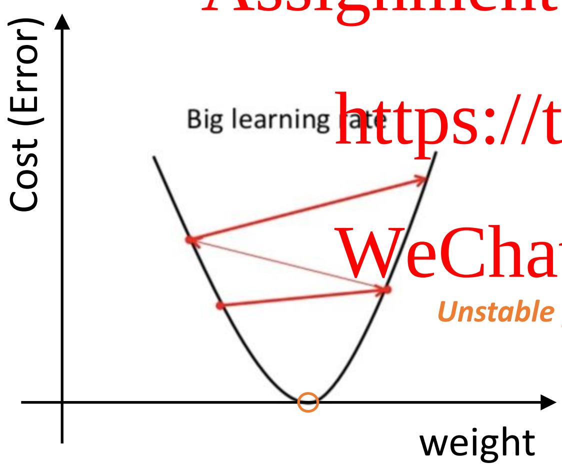
Finding the Optimal weight and bias – Gradient Descent

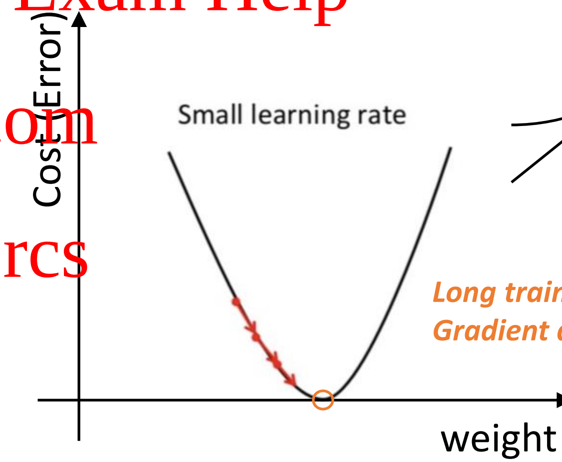
There are different types of Gradient descent optimization algorithms:
Batch Gradient Descent, Stochastic Gradient Descent, Momentum, Adam, etc.

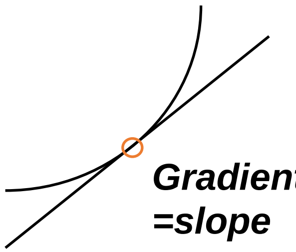
Deep Learning for NLP

Choose the optimal Learning Rate!

Learning Rate: *a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated.*





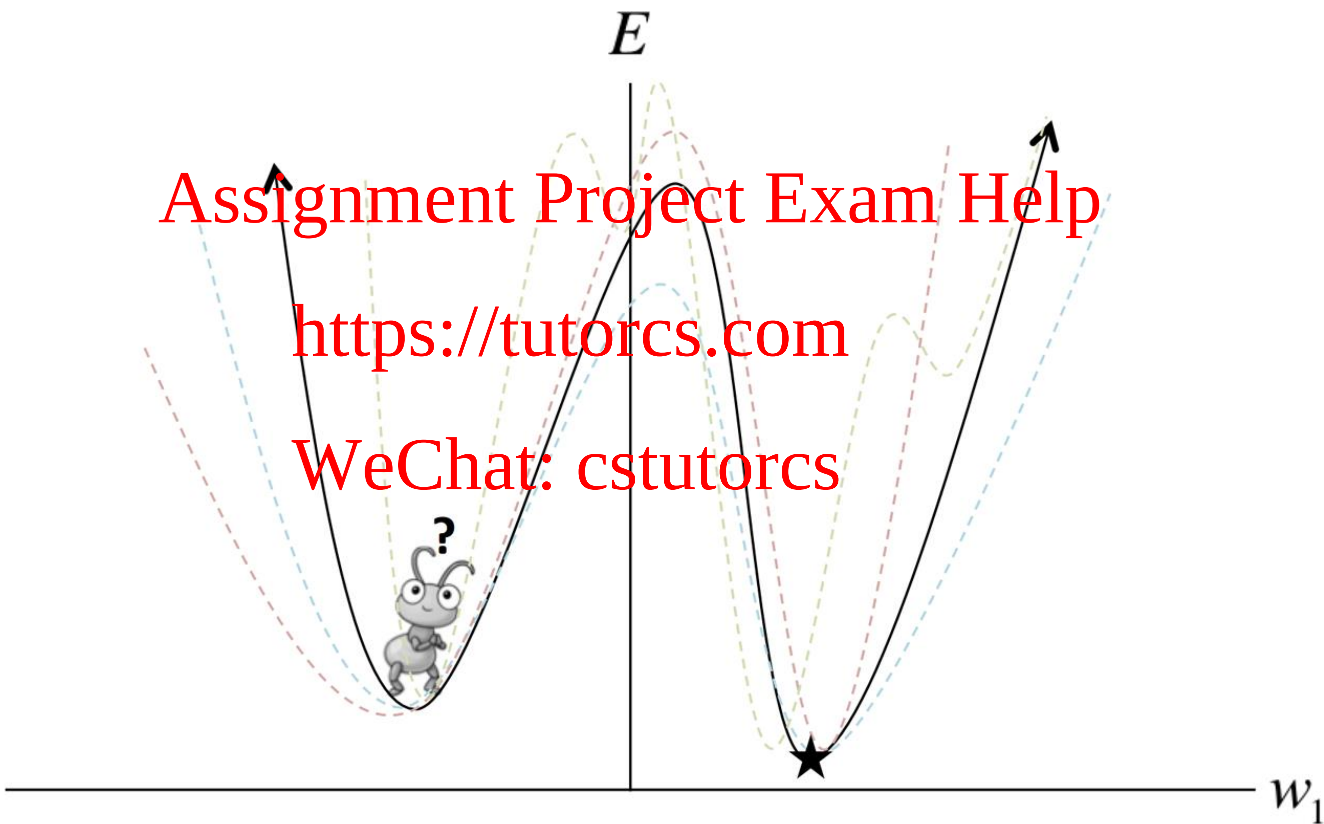


*new_weight = existing_weight — learning_rate * gradient*

*new_weight = existing_weight — learning_rate * (current_output – desired output) *gradient(current output) * existing_input*

3 Deep Learning for NLP

Finding the Optimal weight and bias – Gradient Descent



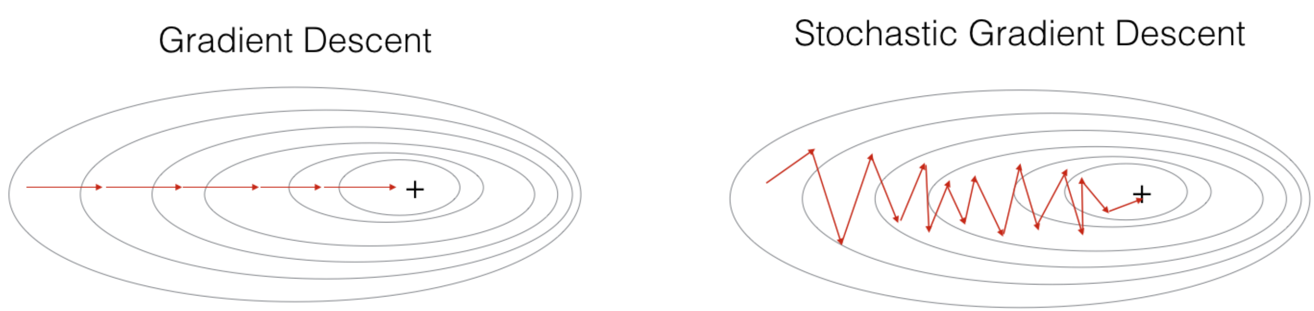
There are different types of Gradient descent optimization algorithms:
Batch Gradient Descent, Stochastic Gradient Descent, Momentum, Adam, etc.

Stochastic Gradient Descent

***The cost would be very expensive** if we calculate it for all windows in the corpus!*
You would wait a very long time before making a single update!

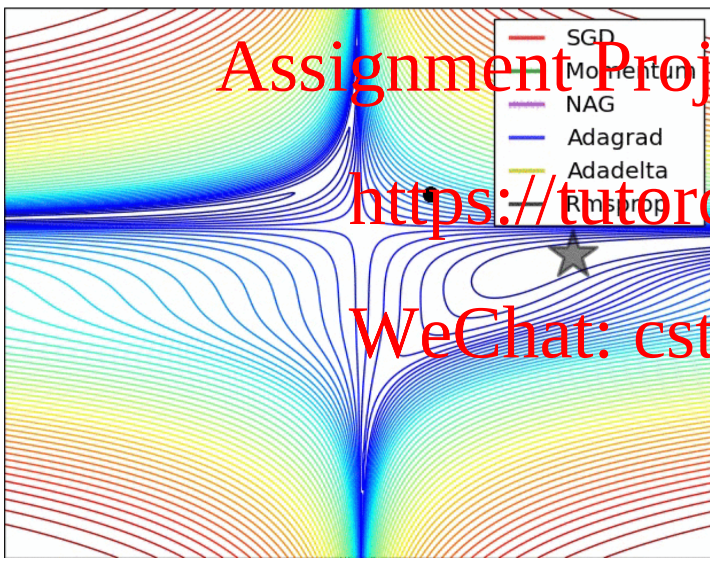
The Solution can be used different Gradient Descent Method.
*The most common – "**Stochastic Gradient Descent (SGD)**"*

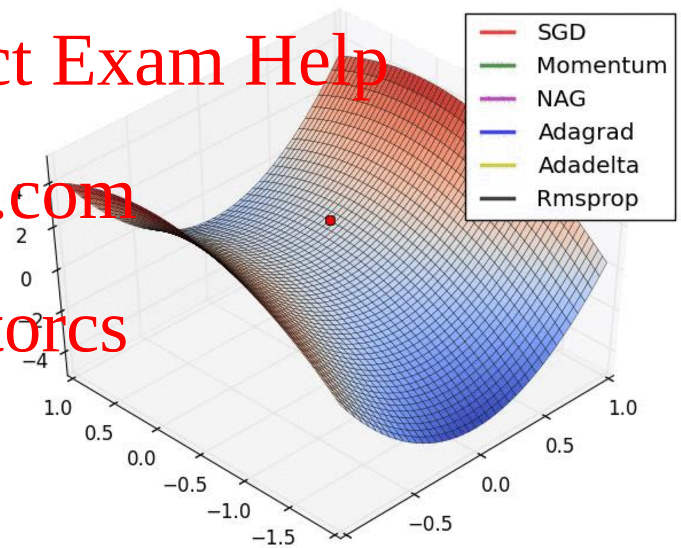
*Vanilla (Batch) gradient descent performs redundant computations for large datasets, as it **recomputes gradients for similar examples before each parameter update**. SGD does away with **this redundancy by performing one update at a time.** It is therefore usually much faster and can also be used to learn online.*



3 Deep Learning for NLP

Finding the Optimal weight and bias – Gradient Descent





There are different types of Gradient descent optimization algorithms:
Batch Gradient Descent, Stochastic Gradient Descent, Momentum, Adam, etc.

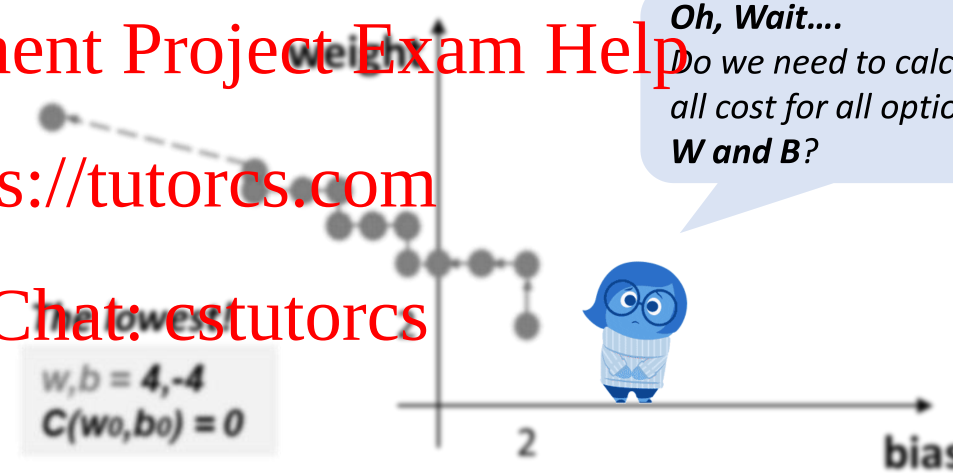
Backpropagation (weight update)

Deep Learning with Neural Network - Optimizer

weight *bias*

$$y = 4\,x\,-4$$

x 🍎	y 🍌
1	0
5	16
6	20



Oh, Wait….
*Do we need to calculate all cost for all options of **W and B**?*

<u>***Expensive to compute (hours or days)***</u>

arg min C(w,b)

w,b∈[-∞,∞]

Gradient!

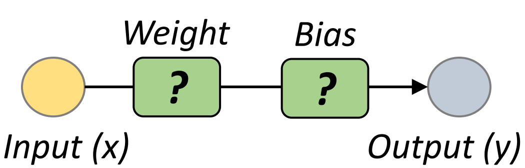
Deep Learning with Neural Network - Optimizer

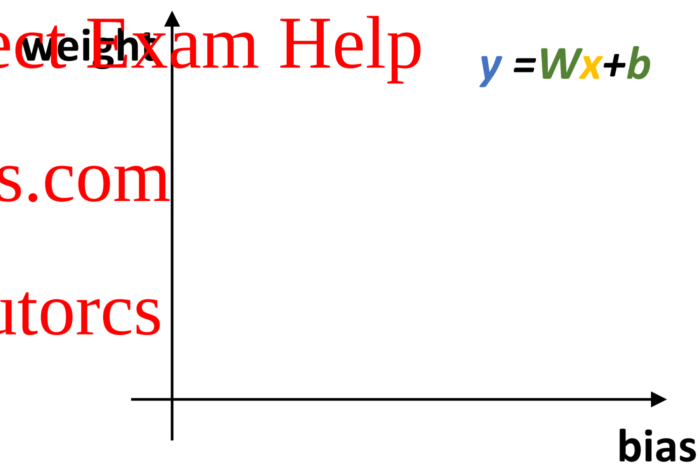
Actual Data

weight *bias*

$$y = ?\ x + ?$$

x 🍎	y 🍌
1	0
5	16
6	20





$y = Wx + b$

Should be used sparingly

$$\underset{w,b \in [-\infty,\infty]}{\arg\min}\ C(w,b)$$

Gradient!

Deep Learning with Neural Network - Optimizer

Actual Data

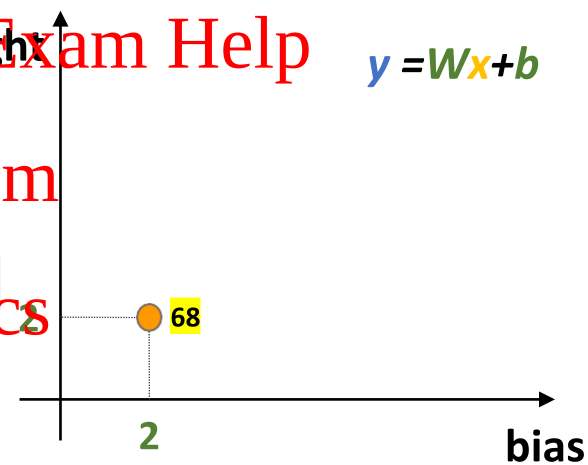
weight *bias*

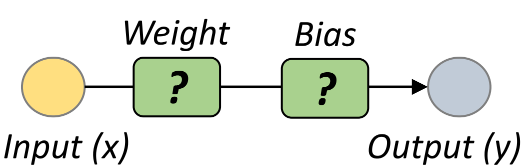
$$y = ?\ x + ?$$

x 🍎	y 🍌
1	0
5	16
6	20

$$\underset{w,b \in [-\infty,\infty]}{\arg\min}\ C(w,b)$$

$w,b = 2,2$ $C(w_0,b_0) = 68$





THE UNIVERSITY OF SYDNEY

Gradient!

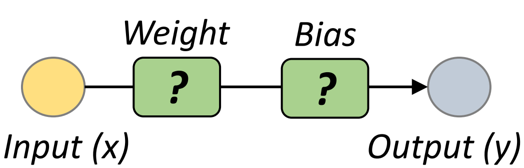
Deep Learning with Neural Network - Optimizer

Actual Data

weight *bias*

$$y = ?\, x + ?$$

x 🍎	y 🍌
1	0
5	16
6	20



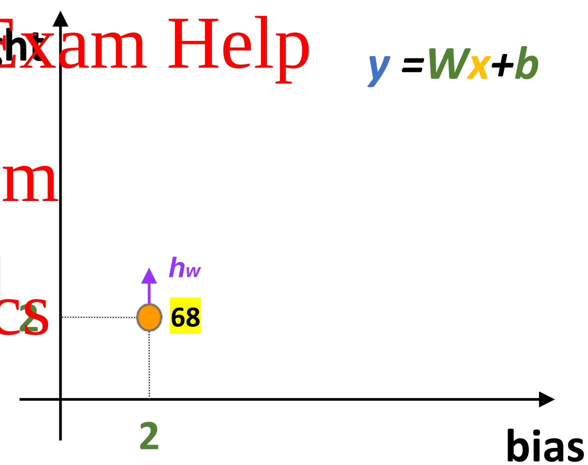
$$\underset{w,b\in[-\infty,\infty]}{\arg\min}\ C(w,b)$$

$w,b = 2,2$ $C(w_0,b_0) = 68$

$h_w = 1$

$C(w_0+h_w, b_0) = C(3,2) = 26$

$y = Wx + b$



THE UNIVERSITY OF SYDNEY

Gradient!

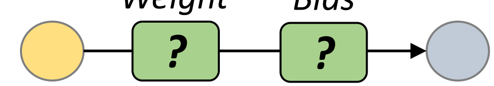
Deep Learning with Neural Network - Optimizer

Actual Data

weight, bias

$y = ?\, x + ?$

x 🍎	y 🍌
1	0
5	16
6	20



$$\underset{w,b \in [-\infty,\infty]}{\arg\min}\ C(w,b)$$

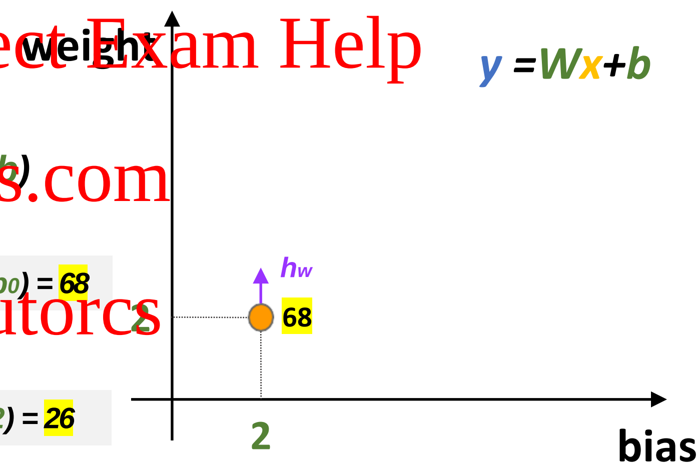
$w,b = 2,2 \quad C(w_0,b_0) = 68$

$h_w = 1$

$C(w_0 + h_w, b_0) = C(3,2) = 26$

$$r = \frac{(C(w_0+h_w, b_0) - C(w_0,b_0))}{1}$$

$$r = \frac{(C(2+1,2) - C(2,2)) = -42}{1}$$



$y = Wx + b$

3 Deep Learning for NLP

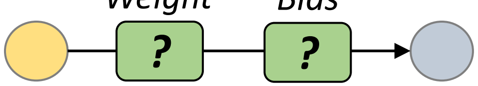
Gradient!

Deep Learning with Neural Network - Optimizer

Actual Data

$$\underbrace{y}_{} = \overset{weight}{?}\, x + \overset{bias}{?}$$

x 🍎	y 🍌
1	0
5	16
6	20

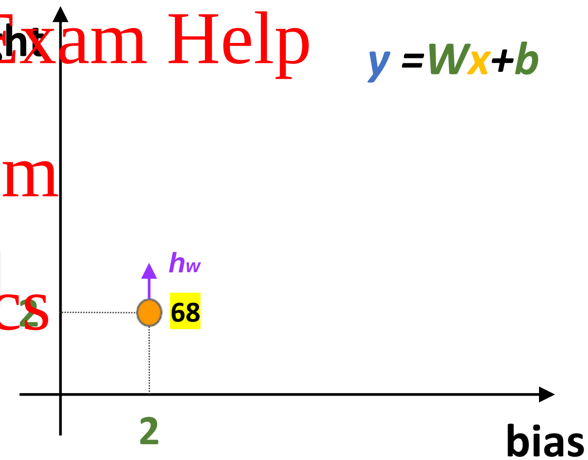


$$\underset{w,b\in[-\infty,\infty]}{\arg\min}\ C(w,b)$$

$w,b = 2,2$ $C(w_0,b_0) = 68$

$h_w = 1$, $r = -42$
$h_w = 0.1$, $r = -98$
$h_w = 0.01$, $r = -104$
$h_w = 0.001$, $r = -104$

$y = Wx + b$



Gradient!

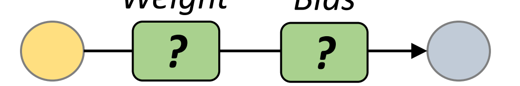
Deep Learning with Neural Network - Optimizer

Actual Data

weight *bias*

$$y = ?\,x + ?$$

x 🍎	y 🍌
1	0
5	16
6	20

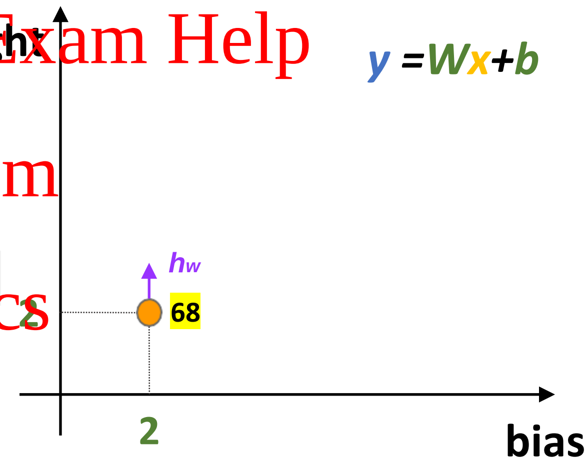


$$\underset{w,b \in [-\infty,\infty]}{\arg\min}\, C(w,b)$$

$w,b = 2,2$ $C(w_0,b_0) = 68$

$h_w = 1$, $r = -42$
$h_w = 0.1$, $r = -98$
$h_w = 0.01$, $r = -104$
$h_w = 0.001$, $r = -104$

$$h_w \rightarrow 0 , \quad r = \frac{\partial C}{\partial w}(w_0,b_0)$$



$y = Wx + b$

THE UNIVERSITY OF SYDNEY

Deep Learning with Neural Network - Optimizer

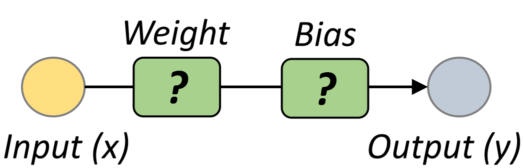
Gradient!

Actual Data

weight *bias*

$$y = ?\ x + ?$$

x 🍎	y 🍌
1	0
5	16
6	20

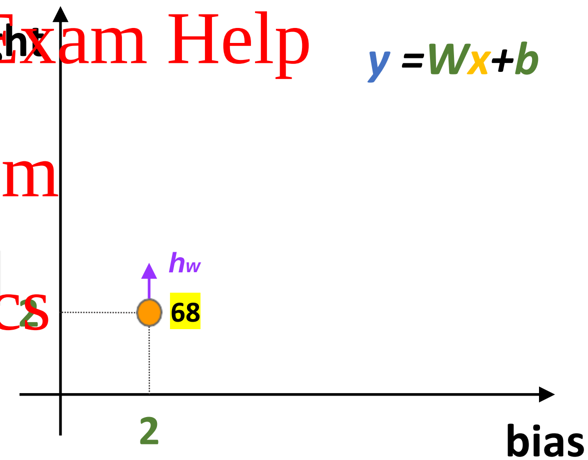


$$\underset{w,b\in[-\infty,\infty]}{\arg\min}\ C(w,b)$$

$w,b = 2,2$ $C(w_0,b_0) = 68$

$$\frac{\partial C}{\partial w} = \frac{\partial \sum_n (y_n-\hat{y}_n)^2}{\partial w}$$

$y = Wx+b$



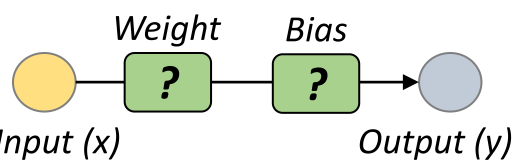
Deep Learning with Neural Network - Optimizer

Gradient!

Actual Data

$$y = \underset{\text{weight}}{?}\; x + \underset{\text{bias}}{?}$$

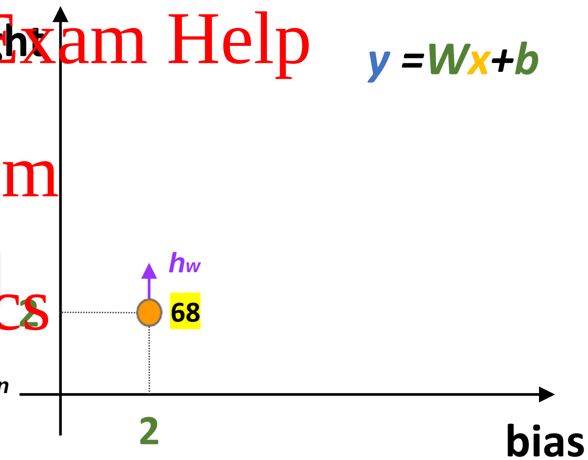
x 🍎	y 🍌
1	0
5	16
6	20



$$\underset{w,b \in [-\infty, \infty]}{\arg\min}\; C(w,b)$$

$w,b = 2,2 \qquad C(w_0,b_0) = 68$

$$\frac{\partial C}{\partial w} = \frac{\partial \sum_n (y_n - \hat{y}_n)^2}{\partial w} = \sum_n 2(y_n - \hat{y}_n)x_n$$



THE UNIVERSITY OF SYDNEY

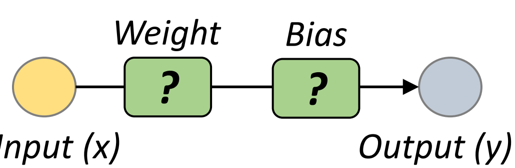
Deep Learning with Neural Network - Optimizer

Gradient!

Actual Data

$$y = \underbrace{?}_{\text{weight}}\, x + \underbrace{?}_{\text{bias}}$$

x 🍎	y 🍌
1	0
5	16
6	20

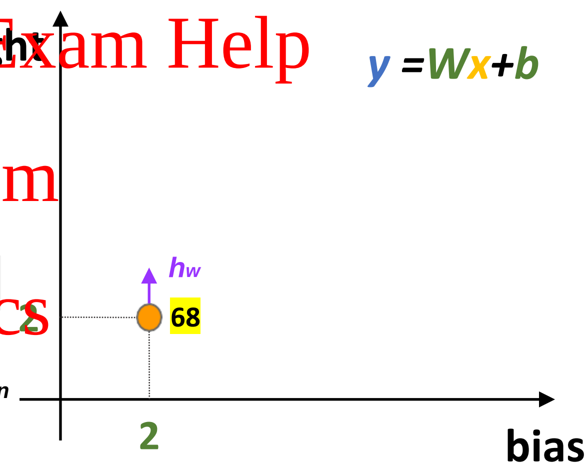


$$\underset{w,b\in[-\infty,\infty]}{\arg\min}\; C(w,b)$$

$w,b = 2,2 \qquad C(w_0,b_0) = 68$

$$\frac{\partial C}{\partial w} = \frac{\partial \sum_n (y_n-\hat{y}_n)^2}{\partial w} = \sum_n 2(y_n-\hat{y}_n)x_n$$

$$h_w \rightarrow 0\,,\; r = \frac{\partial C}{\partial w}(w_0,b_0)$$



Deep Learning with Neural Network - Optimizer

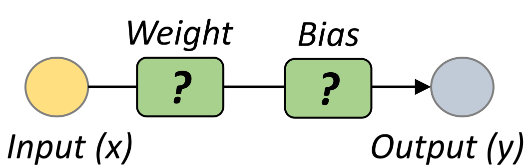
Gradient!

Actual Data

weight *bias*

$$y = ?\ x + ?$$

x 🍎	y 🍌
1	0
5	16
6	20



$$\underset{w,b\in[-\infty,\infty]}{\arg\min}\ C(w,b)$$

$w,b = 2,2$ $C(w_0,b_0) = 68$

$$\frac{\partial C}{\partial w} = \frac{\partial \sum_n (y_n-\hat{y}_n)^2}{\partial w} = \sum_n 2(y_n-\hat{y}_n)x_n$$

$$h_w \rightarrow 0,\ r = \frac{\partial C}{\partial w}(w_0,b_0) = 104$$

weight *bias*

$$y = 2\ x + 2$$

x 🍎	ŷ (*predicted*)	y 🍌 (*actual*)	(y−ŷ)	2(y−ŷ)x
1	4	0	-4	-8
5	12	16	4	40
6	14	20	6	72

THE UNIVERSITY OF SYDNEY

Gradient!

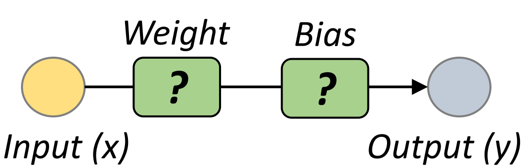
Deep Learning with Neural Network - Optimizer

Actual Data

weight *bias*

$y = ?\ x + ?$

x 🍎	y 🍌
1	0
5	16
6	20

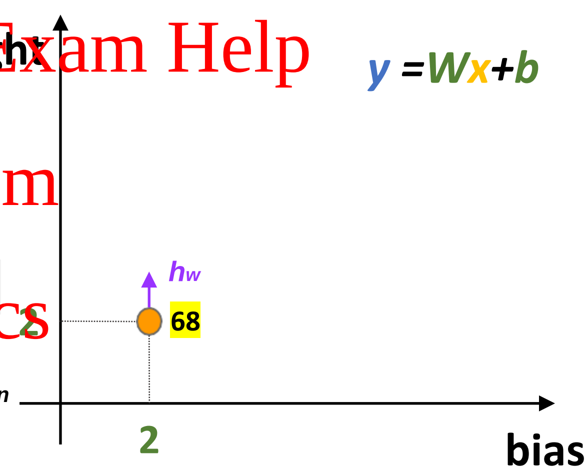


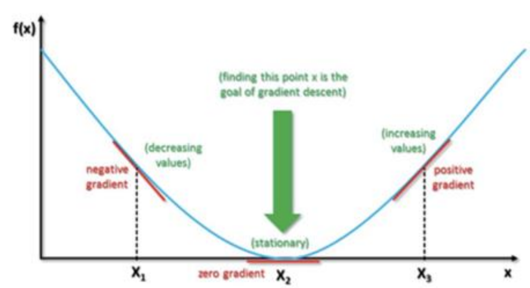
$$\underset{w,b\in[-\infty,\infty]}{\arg\min}\ C(w,b)$$

$w,b = 2,2 \qquad C(w_0,b_0) = 68$

$$\frac{\partial C}{\partial w} = \frac{\partial \sum_n (y_n-\hat{y}_n)^2}{\partial w} = \sum_n 2(y_n-\hat{y}_n)x_n$$

$$\frac{\partial C}{\partial b} = \frac{\partial \sum_n (y_n-\hat{y}_n)^2}{\partial b} = \sum_n 2(y_n-\hat{y}_n)$$





3 Deep Learning for NLP

Gradient!

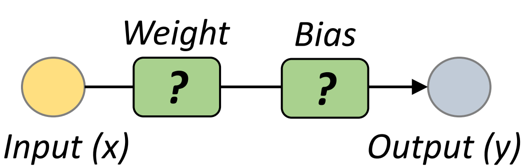
Deep Learning with Neural Network - Optimizer

Actual Data

weight *bias*

$y = ?\ x + ?$

x 🍎	y 🍌
1	0
5	16
6	20



$$\underset{w,b \in [-\infty,\infty]}{\arg\min}\ C(w,b)$$

$w,b = 2,2 \qquad C(w_0,b_0) = 68$

$h_w \rightarrow 0\ ,\ r = \frac{\partial C}{\partial w}(w_0,b_0) = 104$

$h_b \rightarrow 0\ ,\ r = \frac{\partial C}{\partial b}(w_0,b_0) = 12$

weight *bias*

$y = 2\ x + 2$

x 🍎	ŷ (predicted)	y (actual) 🍌	(y−ŷ)	2(y−ŷ)
1	4	0	-4	-8
5	12	16	4	8
6	14	20	6	12

Gradient!

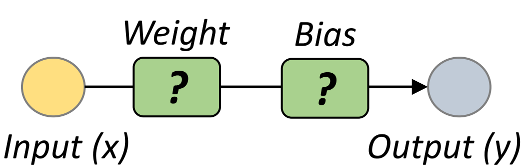
Deep Learning with Neural Network - Optimizer

Actual Data

weight *bias*

$y = ?\ x + ?$

x 🍎	y 🍌
1	0
5	16
6	20



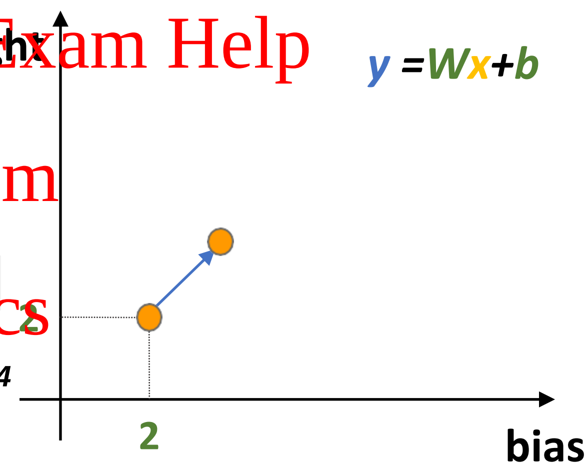
$$\underset{w,b \in [-\infty,\infty]}{\arg\min}\ C(w,b)$$

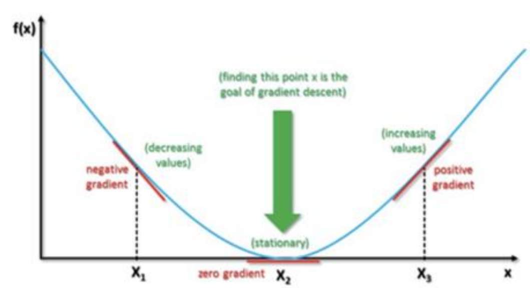
$w,b = 2,2$ $C(w_0,b_0) = 68$

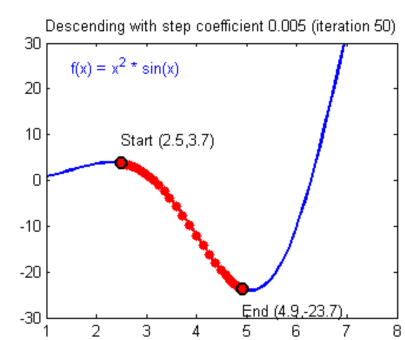
$h_w \rightarrow 0\ ,\ r = \frac{\partial C}{\partial w}(w_0,b_0) = 104$

$h_b \rightarrow 0\ ,\ r = \frac{\partial C}{\partial b}(w_0,b_0) = 12$

$y = Wx + b$







Deep Learning with Neural Network

Data

x 🍎	y 🍌
1	0
5	16
6	20

Model

$$y = \underbrace{?}_{\text{weight}}\, x + \underbrace{?}_{\text{bias}}$$

Cost

$$C(w,b) = \sum_{n \in \{0,1,2\}} (y_n - \hat{y}_n)$$

Optimizer

$$\underset{w,b \in [-\infty,\infty]}{\arg\min}\, C(w,b)$$

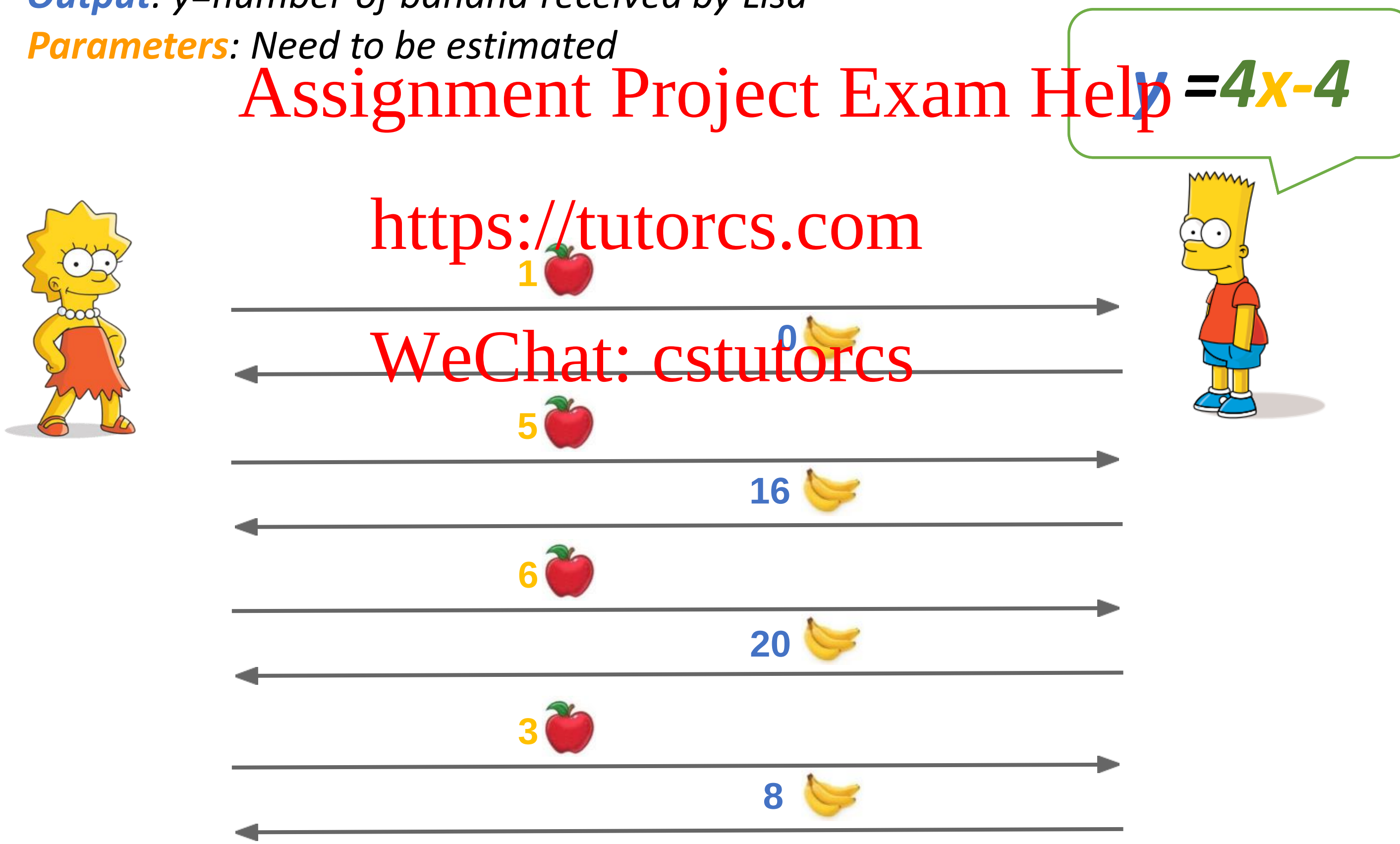
System

$$y = \underbrace{4}_{\text{weight}}\, x \underbrace{- 4}_{\text{bias}}$$

3 Deep Learning for NLP

Deep Learning with Neural Network

Input: *x=number of apple given by Lisa*
Output: *y=number of banana received by Lisa*
Parameters: *Need to be estimated*



Deep Learning with Neural Network

$$y = w_1 \underset{}{\overset{pixel(1,1)}{x_1}} + w_2 \overset{pixel(1,2)}{x_2} + w_3x_3 + w_4x_4 + \ldots + w_nx_n + b$$

Data

Millions of ***Parameters***
Millions of ***Samples***

Deep Learning with Neural Network

$$y = w_1\overset{\text{Vector1}}{x_1} + w_2\overset{\text{Vector2}}{x_2} + w_3x_3 + w_4x_4 + \ldots + w_nx_n + b$$

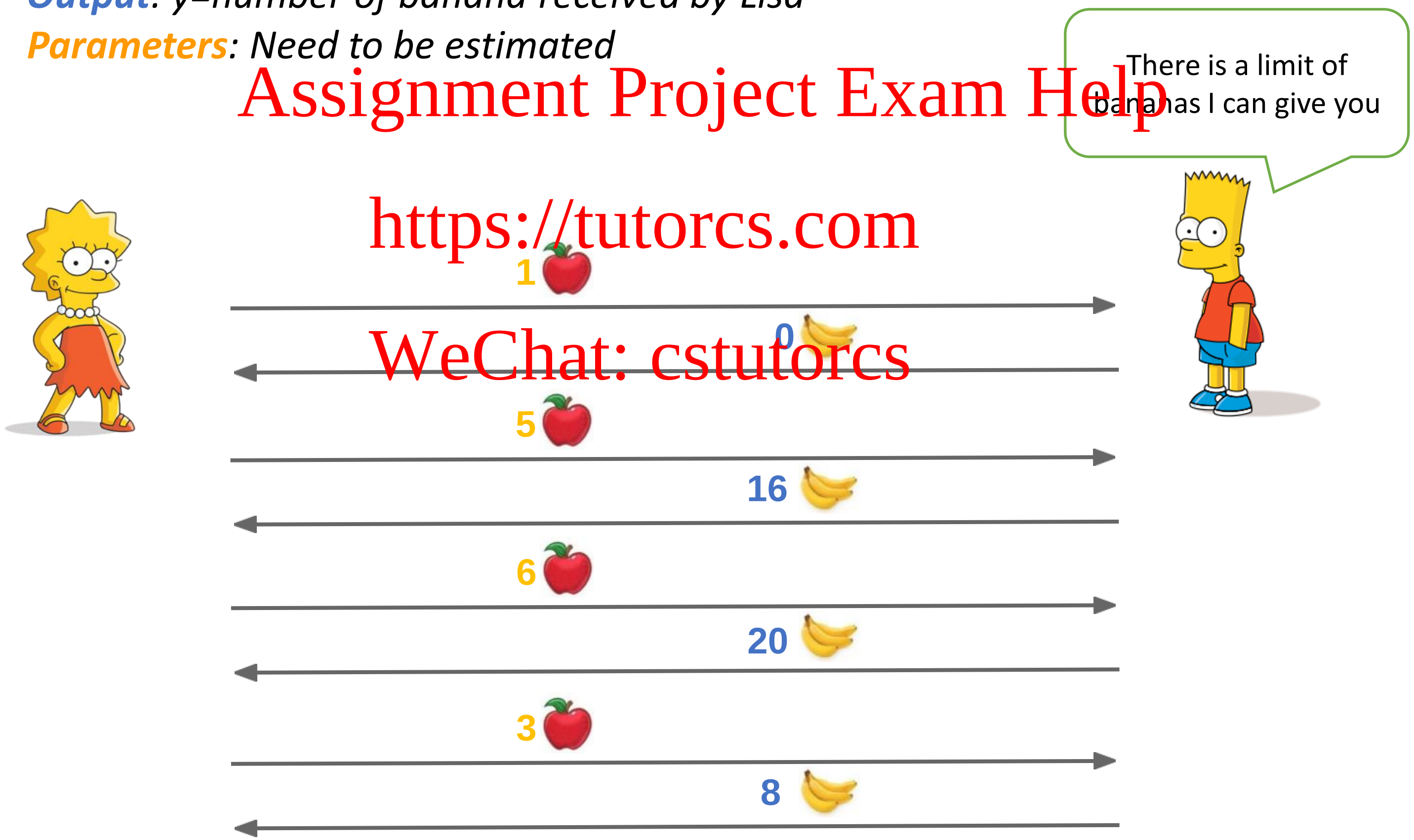
Millions of ***Parameters***

Millions of ***Samples***

THE UNIVERSITY OF SYDNEY

Deep Learning with Neural Network

Input: *x=number of apple given by Lisa*
Output: *y=number of banana received by Lisa*
Parameters: *Need to be estimated*



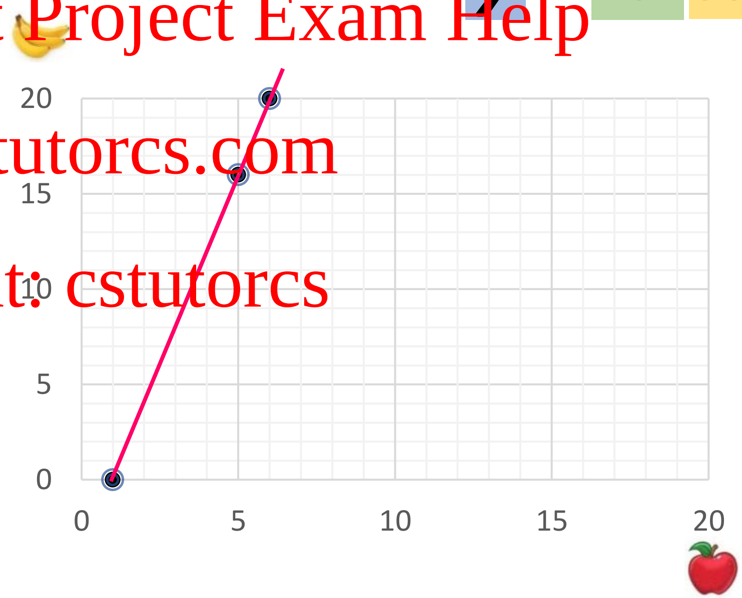
Deep Learning with Neural Network

Nonlinear Neural Network

$$y = \underbrace{4}_{\text{weight}}\, x \underbrace{- 4}_{\text{bias}}$$

Data

x	y
1	0
5	16
6	20



Deep Learning with Neural Network

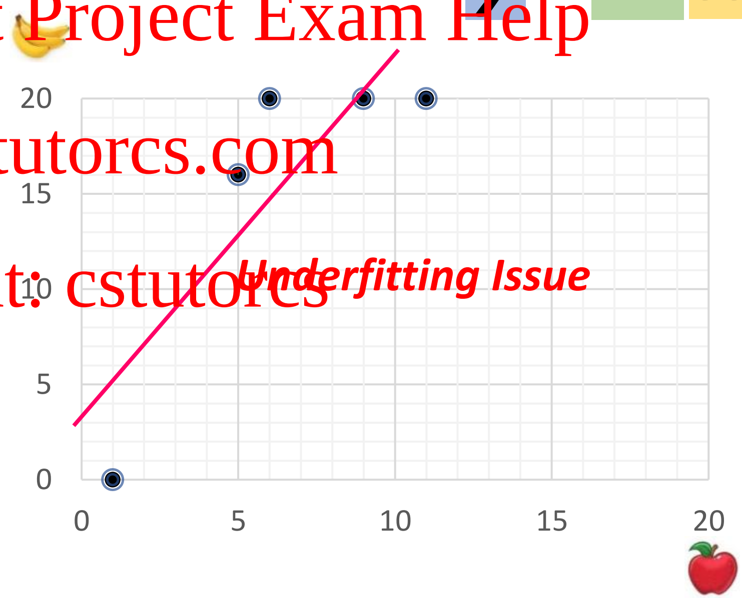
Nonlinear Neural Network

weight *bias*

$$y = 2x + 3$$

Data

x 🍎	y 🍌
1	0
5	16
6	20
9	20
11	20



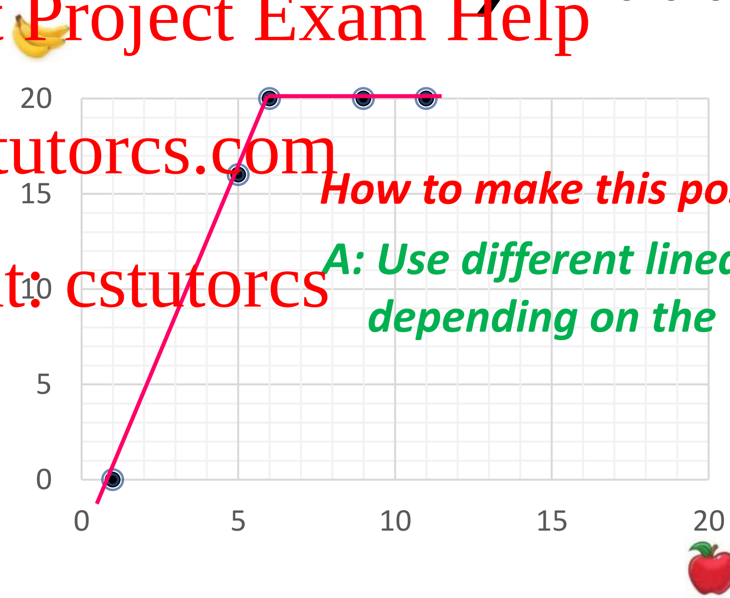
Deep Learning with Neural Network

Nonlinear Neural Network

y = ???

Data

x	y
1	0
5	16
6	20
9	20
11	20



How to make this possible?

A: Use different linear functions depending on the value of x

THE UNIVERSITY OF SYDNEY

Deep Learning with Neural Network

Nonlinear Neural Network

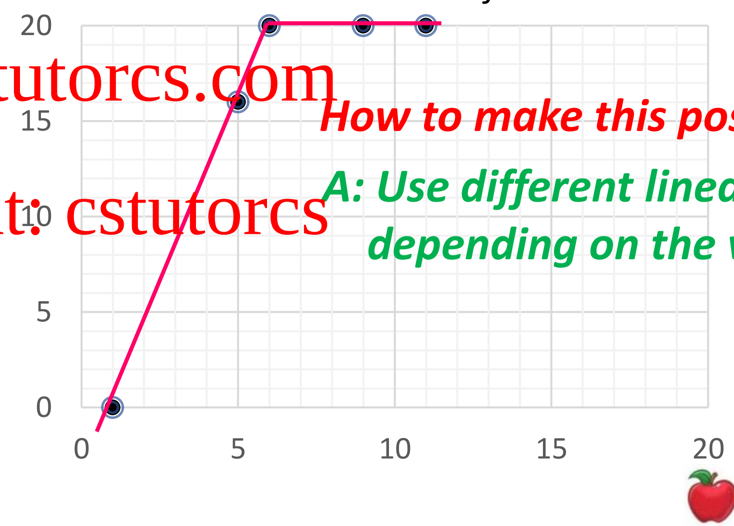
weight bias weight bias

$$y = (w_1 x + b_1)s_1 + (w_2 x + b_2)s_2$$

If x < 6 and 0 *If x >= 6 and 0*

Data

x 🍎	y 🍌
1	0
5	16
6	20
9	20
11	20



How to make this possible?

A: Use different linear functions depending on the value of x

Deep Learning with Neural Network

Nonlinear Neural Network

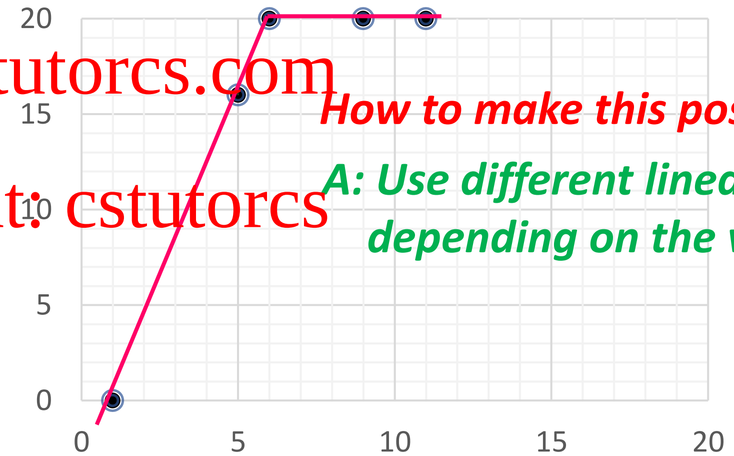
weight *bias* *weight* *bias*

$$y = (w_1 x + b_1)s_1 + (w_2 x + b_2)s_2$$

If x < 6 and 0 *If x >= 6 and 0*

Data

x	y
1	0
5	16
6	20
9	20
11	20



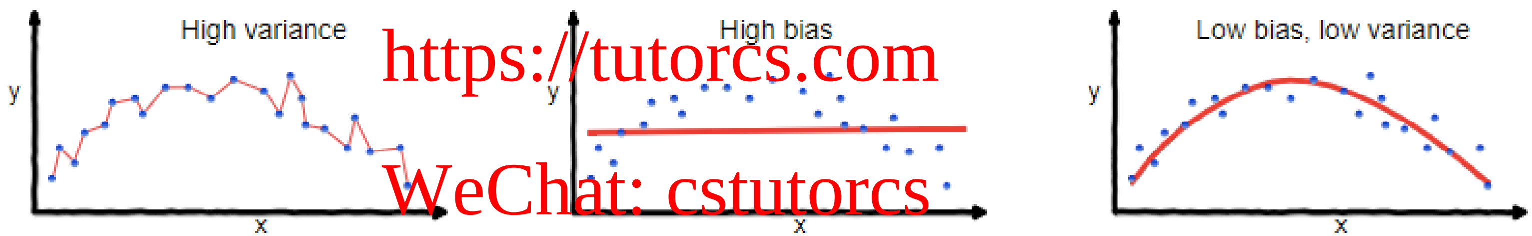
How to make this possible?

A: Use different linear functions depending on the value of x

$W_1 = 4$, $b_1 = -4$, $W_2 = 0$, $b_2 = 20$

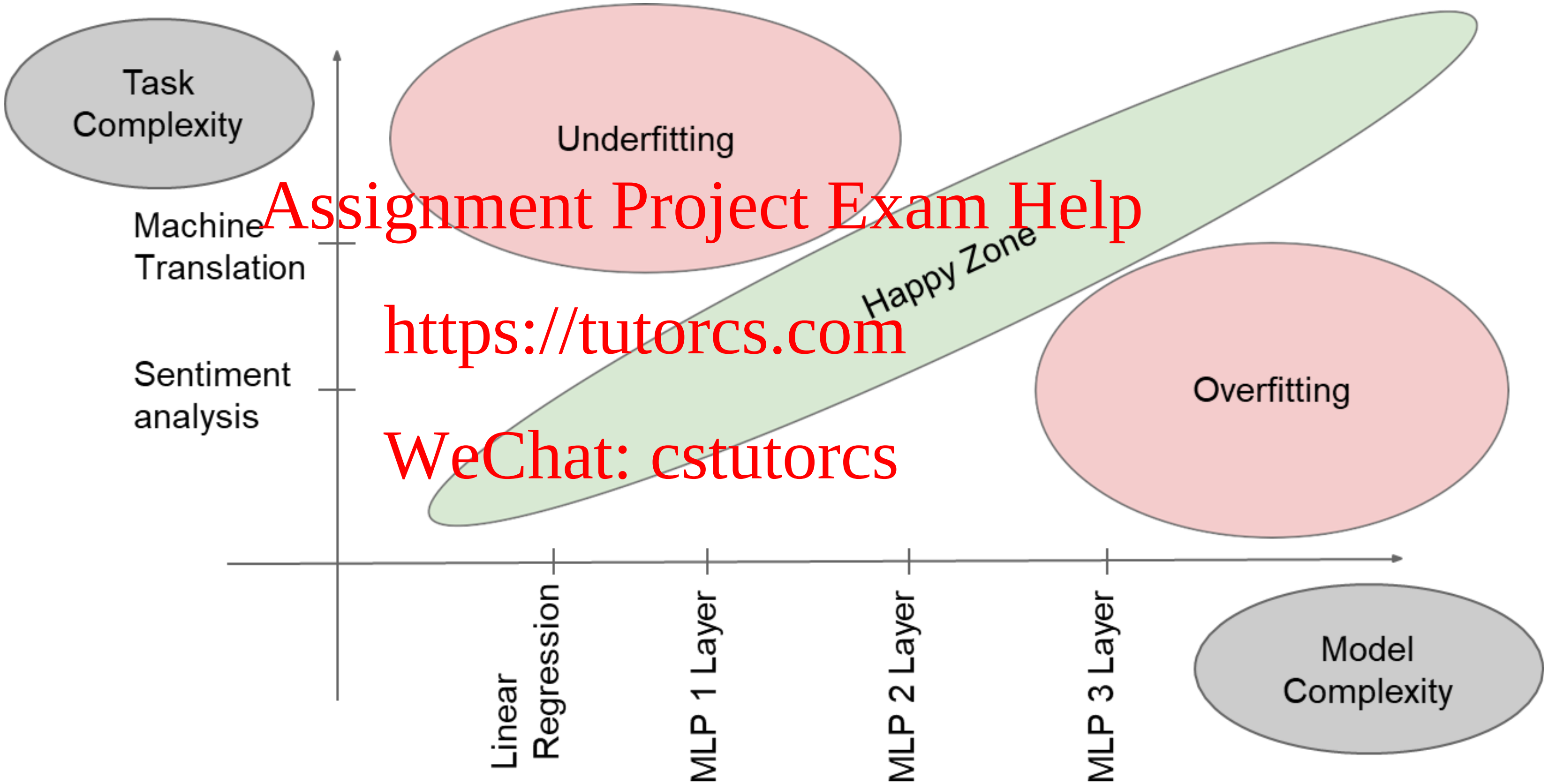
3 Deep Learning for NLP

Deep Learning with Neural Network

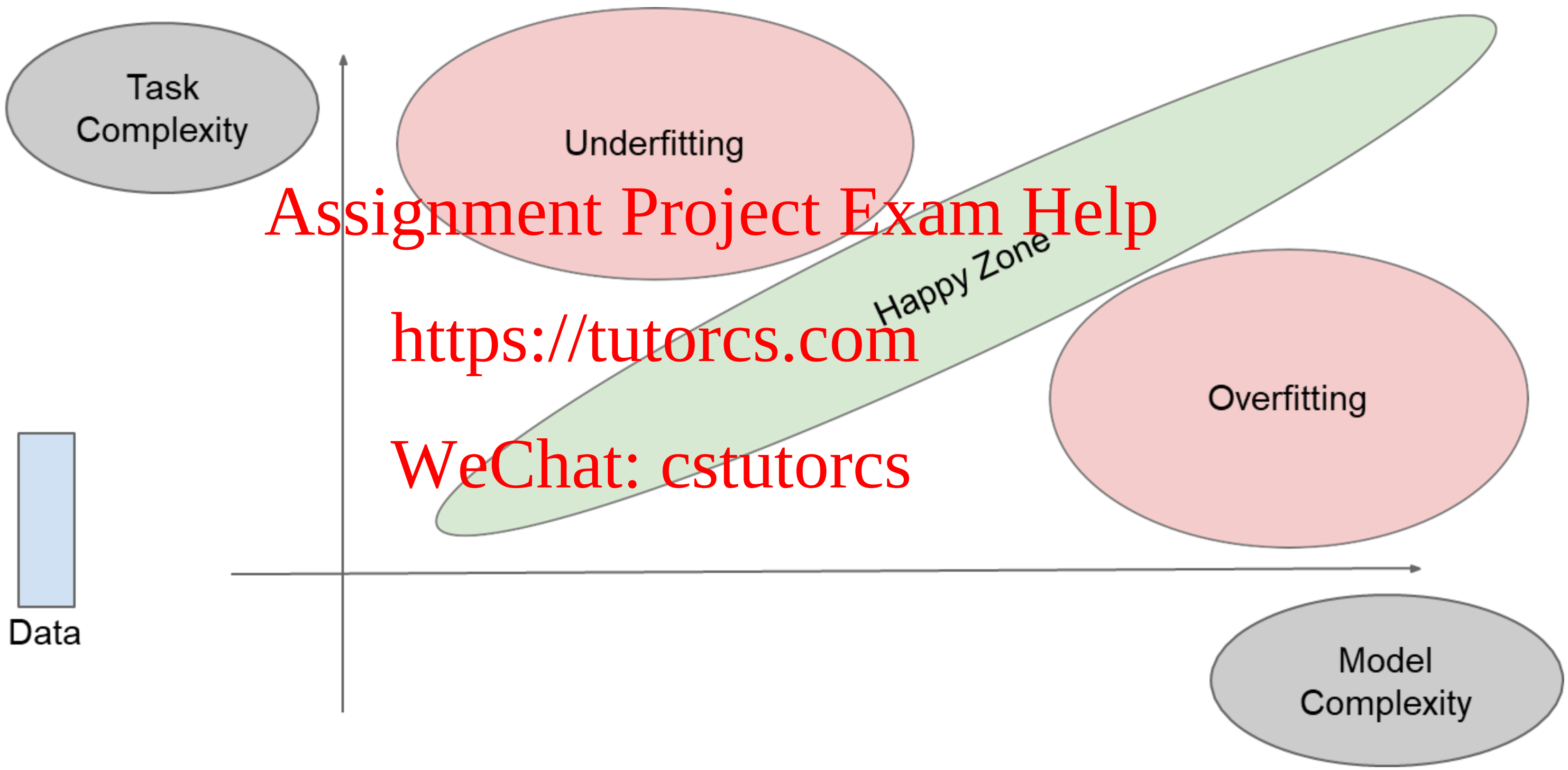


overfitting underfitting Good balance

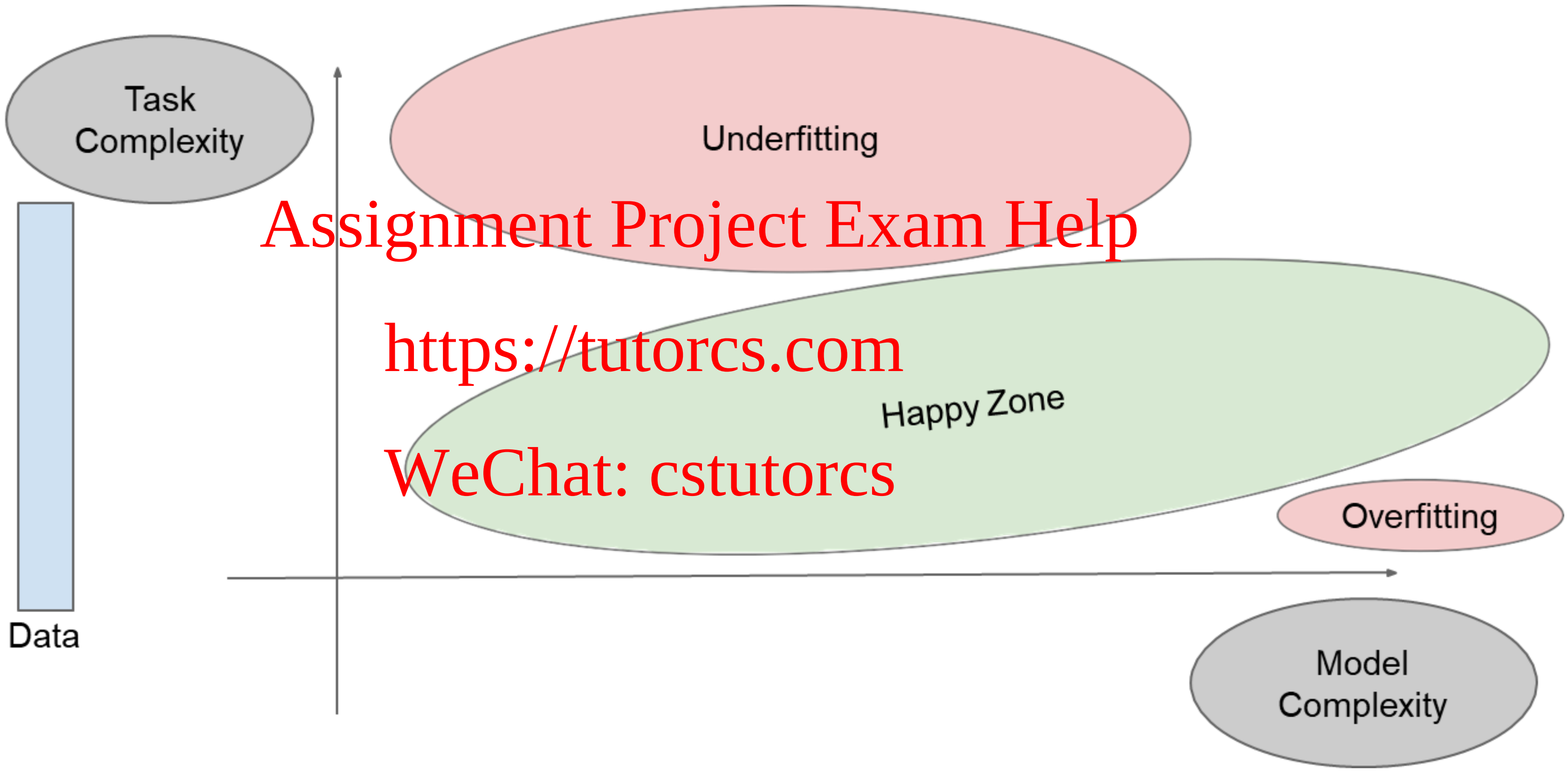
3 Deep Learning for NLP



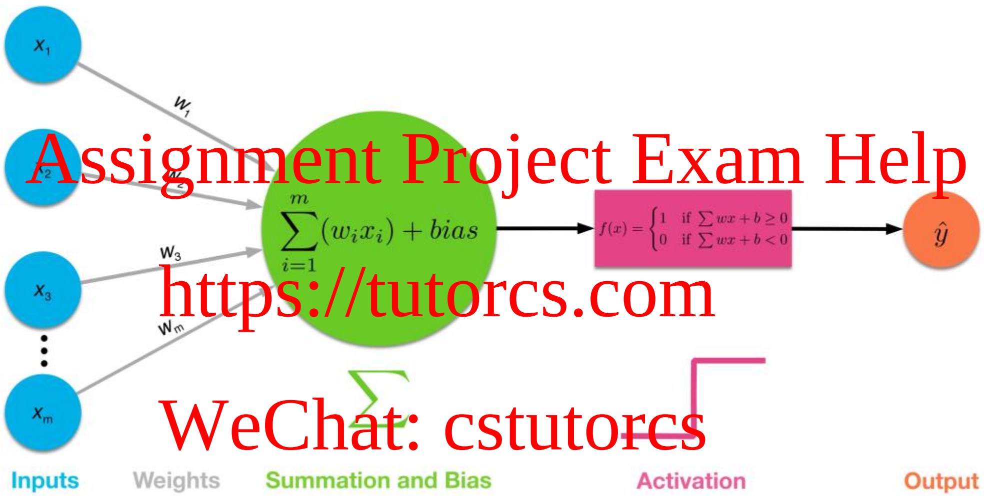
3 Deep Learning for NLP

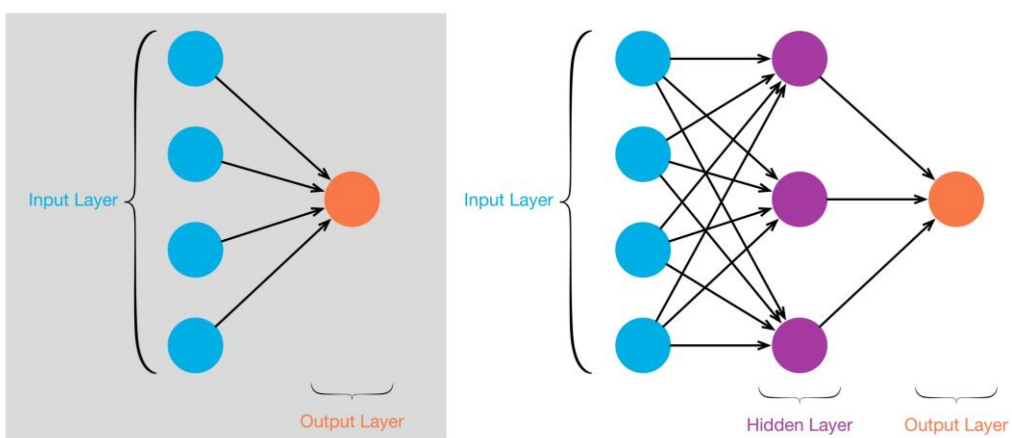


3 Deep Learning for NLP



Single Neuron VS Multilayer



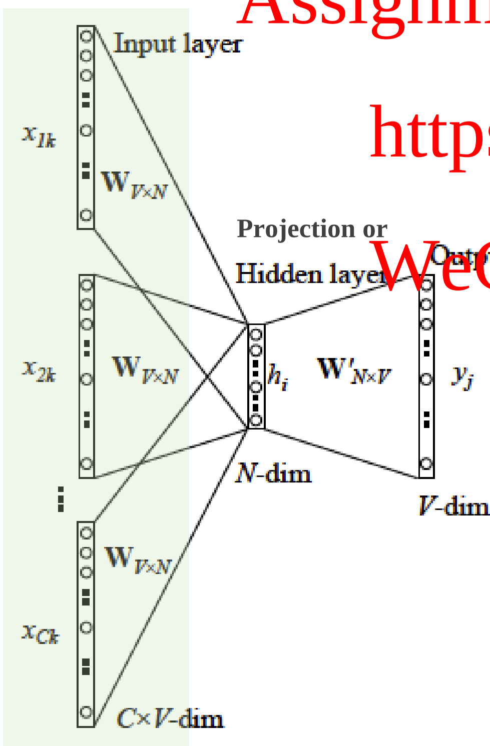


THE UNIVERSITY OF SYDNEY

CBOW – Neural Network Architecture (ReCAP with Optimizer)

Predict center word from (bag of) context words.

Summary of CBOW Training (Review your understanding with equations)



1. Initialise each word in a one-hot vector form.

$\boldsymbol{x_k} = [0,...,0,1,0,...,0]$

2. Use context words (**2m**, based on window size =m) as input of the Word2Vec-CBOW model.

$(\boldsymbol{x^{c-m}}, \boldsymbol{x^{c-m+1}}, \ldots, \boldsymbol{x^{c-1}}, \boldsymbol{x^{c+1}}, \ldots, \boldsymbol{x^{c+m-1}}, \boldsymbol{x^{c+m}}) \in \mathbb{R}^{|V|}$

3. Has two Parameter Matrices:

1) Parameter Matrix (from Input Layer to **Hidden/Projection Layer**)

$\mathbf{W} \in \mathbb{R}^{V\text{x}N}$

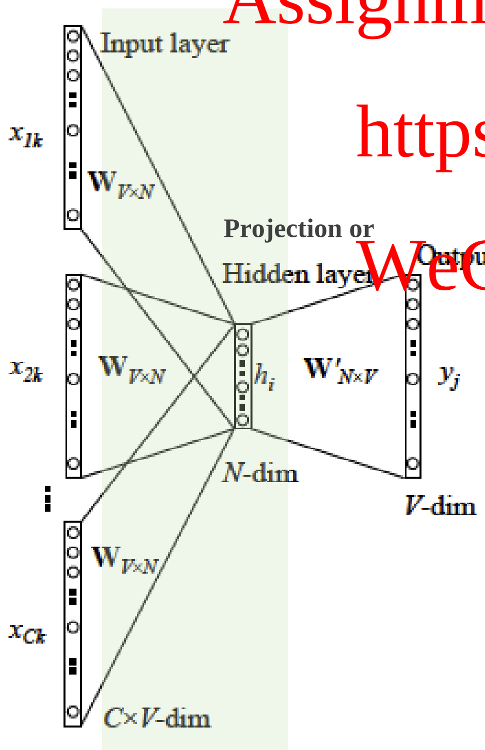
2) Parameter Matrix (to Output Layer)

$\mathbf{W'} \in \mathbb{R}^{N\text{x}V}$

CBOW – Neural Network Architecture (ReCAP with Optimizer)

Predict center word from (bag of) context words.

Summary of CBOW Training (Review your understanding with equations)



4. Initial words are represented in one hot vector so multiplying a **one hot vector** with $\mathbf{W_{V\times N}}$ will give you a 1 x N (embedded word) vector.

e.g. $[0\ \ 1\ \ 0\ \ 0] \times \begin{bmatrix} 10 & 2 & 18 \\ 15 & 22 & 3 \\ 25 & 11 & 19 \\ 4 & 7 & 22 \end{bmatrix} = [15\ \ 22\ \ 3]$

$$(\boldsymbol{v_{c-m}} = \mathrm{W}x^{c-m}, \ldots, \boldsymbol{v_{c+m}} = \mathrm{W}x^{c+m}) \in \mathbb{R}^{\boldsymbol{n}}$$

5. Average those **2m** embedded vectors to calculate the value of the Hidden Layer.

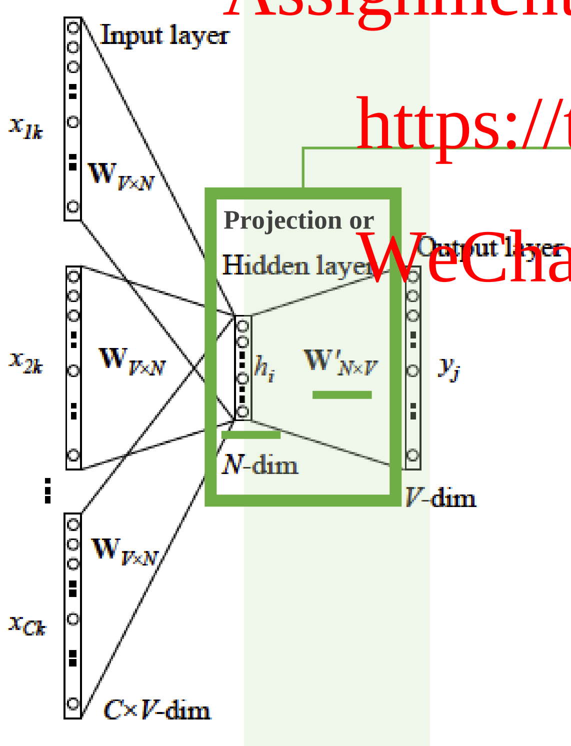
$$\hat{v} = \frac{\boldsymbol{v_{c-m}} + \boldsymbol{v_{c-m+1}} + \ \ldots + \boldsymbol{v_{c+m}}}{2m}$$

Deep Learning for NLP

CBOW – Neural Network Architecture (ReCAP with Optimizer)

Predict center word from (bag of) context words.

Summary of CBOW Training (Review your understanding with equations)



6. Calculate the score value for the output layer. The higher score is produced when words are closer.

$$\boldsymbol{z} = \mathbf{W}' \times \hat{v} \in \mathbb{R}^{|V|}$$

7. Calculate the probability using softmax

$$\hat{y} = softmax\,(\boldsymbol{z}) \in \mathbb{R}^{|V|}$$

8. Train the parameter matrix using **objective function.**

$$\mathrm{H}(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \, log(\hat{y}_j)$$

* Focus on minimising the value

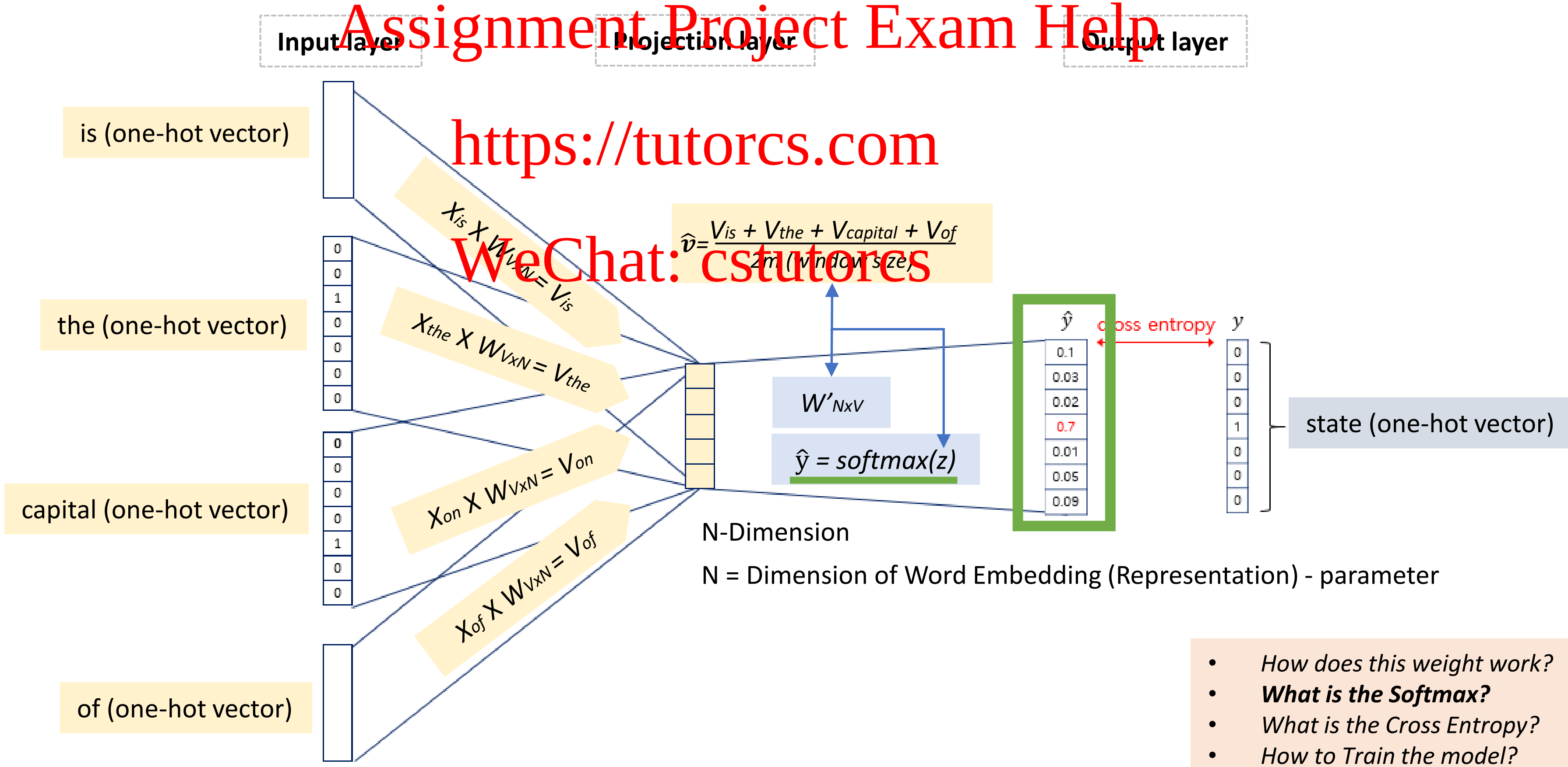
We use an one-hot vector (one 1, the rest 0) so it will be calculated in only one.

$$\mathrm{H}(\hat{y}, y) = -y_j \, log(\hat{y}_j)$$

CBOW – Neural Network Architecture (ReCAP with Optimizer)

Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"



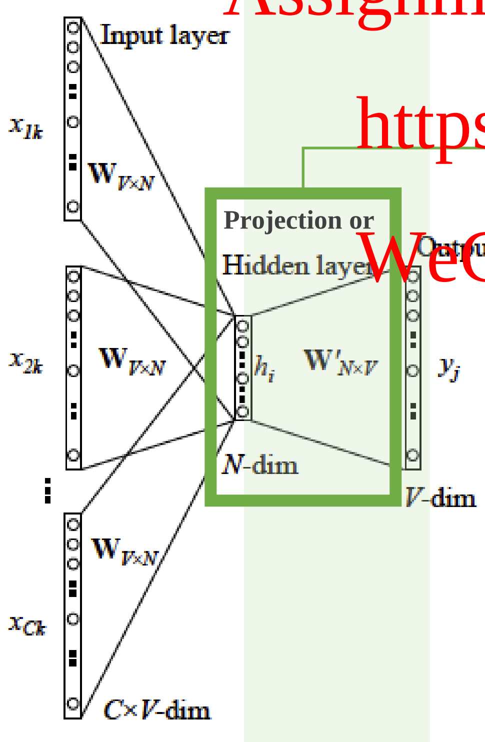
N = Dimension of Word Embedding (Representation) - parameter

- *How does this weight work?*
- ***What is the Softmax?***
- *What is the Cross Entropy?*
- *How to Train the model?*

CBOW – Neural Network Architecture (ReCAP with Optimizer)

Predict center word from (bag of) context words.

Summary of CBOW Training (Review your understanding with equations)



6. Calculate the score value for the output layer. The higher score is produced when words are closer.

$z = \mathbf{W}' \times \hat{v} \in \mathbb{R}^{|V|}$

7. Calculate the probability using softmax

$\hat{y} = softmax\ (\boldsymbol{z}) \in \mathbb{R}^{|V|}$

The softmax is an operator that will be used frequently. It transforms a vector into a vector whose i-th component is:

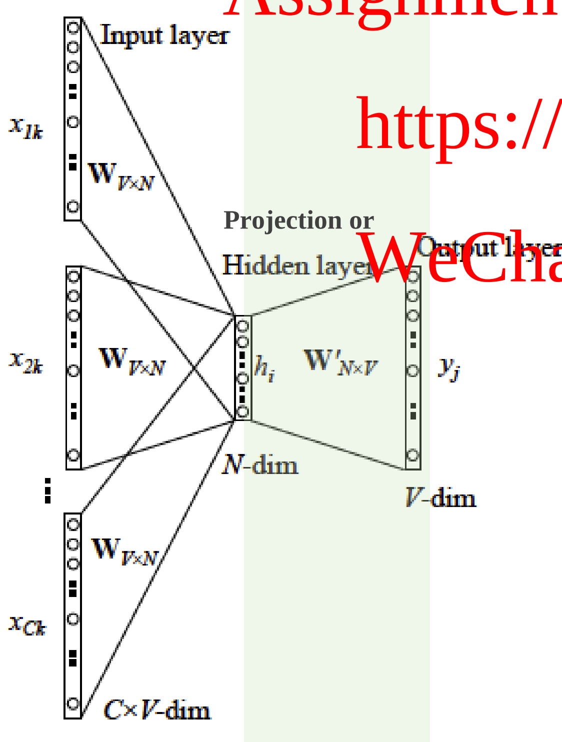
$$\frac{e^{\hat{y}_i}}{\sum_{j=1}^{|V|} e^{\hat{y}_j}}$$

- Exponentiate to make positive
- Dividing by $\sum_{j=1}^{|V|} e^{\hat{y}_j}$ normalizes the vector ($\sum_{j=1}^{n} \hat{y}_j = 1$) to give probability

CBOW – Neural Network Architecture (ReCAP with Optimizer)

Predict center word from (bag of) context words.

Summary of CBOW Training (Review your understanding with equations)



6. Calculate the score value for the output layer. The higher score is produced when words are closer.

$$\boldsymbol{z} = \mathbf{W}' \times \hat{v} \in \mathbb{R}^{|V|}$$

7. Calculate the probability using softmax

$$\hat{y} = softmax\ (\boldsymbol{z}) \in \mathbb{R}^{|V|}$$

8. Train the parameter matrix using **objective function.**

$$\mathrm{H}(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

Cross Entropy

* Focus on minimising the value

We use an one-hot vector (one 1, the rest 0) so it will be calculated in only one.

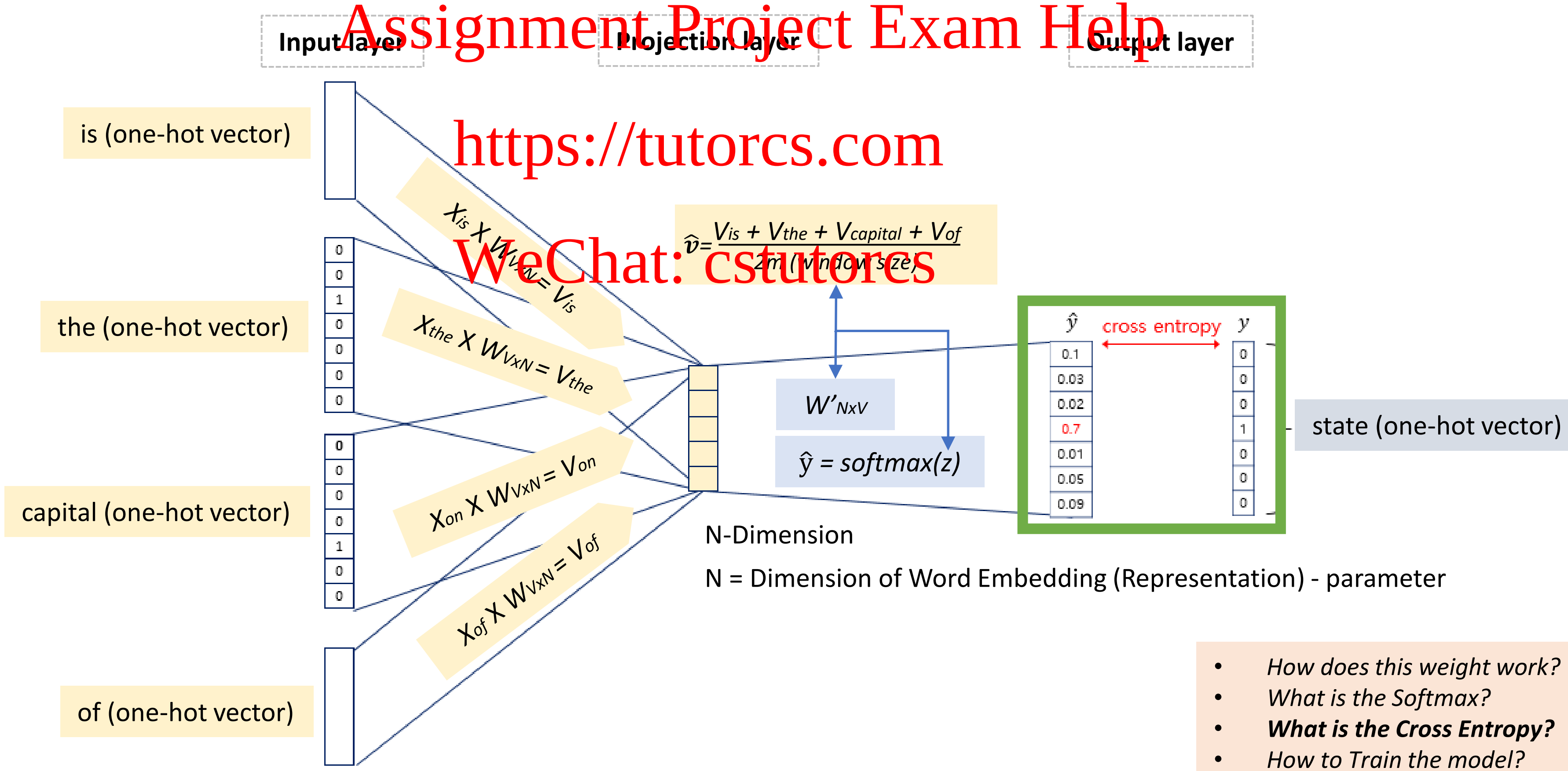
$$\mathrm{H}(\hat{y}, y) = -y_j \log(\hat{y}_j)$$

Deep Learning for NLP

CBOW – Neural Network Architecture (ReCAP with Optimizer)

Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"

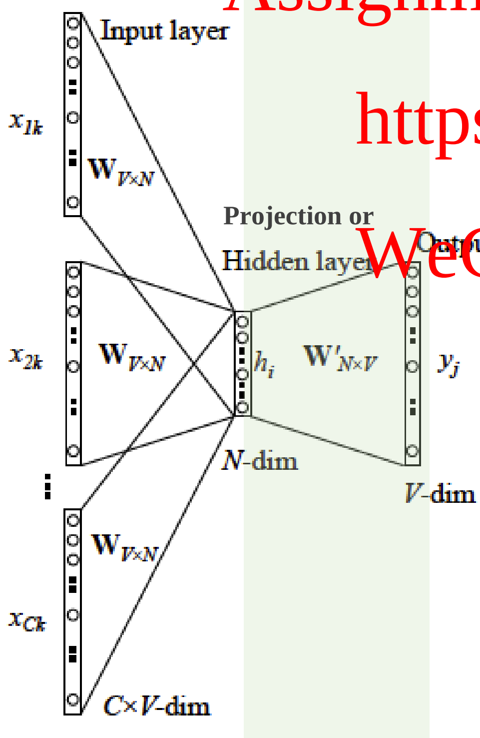


- *How does this weight work?*
- *What is the Softmax?*
- ***What is the Cross Entropy?***
- *How to Train the model?*

CBOW – Neural Network Architecture (ReCAP with Optimizer)

Predict center word from (bag of) context words.

Summary of CBOW Training (Review your understanding with equations)



8-1. Optimization Objective Function can be presented:

$$\begin{aligned}
\text{minimize } J &= -\log P(w_c \mid w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m}) \\
&= -\log P(u_c \mid \hat{v}) \\
&= -\log \frac{\exp(u_c^T \hat{v})}{\sum_{j=1}^{|V|} \exp(u_j^T \hat{v})} \\
&= -u_c^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v})
\end{aligned}$$

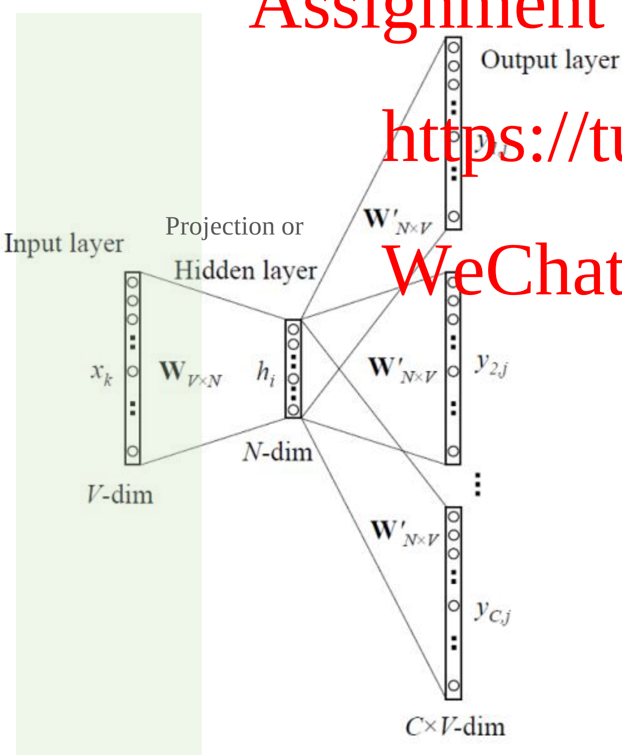
u_i=the output vector representation of word w_i

3 Prediction based Word representation

Skip Gram – Neural Network Architecture

Predict context (“outside”) words (position independent) given center word

Summary of Skip Gram Training (Review your understanding with equations)



1. Initialise the centre word in a one-hot vector form.

$\boldsymbol{x_k} = [0,...,0,1,0,...,0]$

$\boldsymbol{x} \in \mathbb{R}^{|V|}$

2. Has two Parameter Matrices:

1) Parameter Matrix (from Input Layer to Hidden/Projection Layer)

$\mathbf{W} \in \mathbb{R}^{V\text{x}N}$

2) Parameter Matrix (to Output Layer)

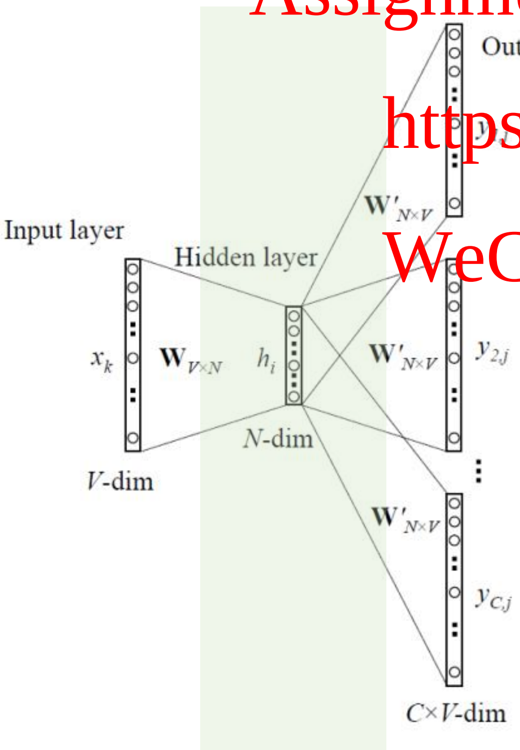
$\mathbf{W}' \in \mathbb{R}^{N\text{x}V}$

Prediction based Word representation

Skip Gram – Neural Network Architecture

Predict context ("outside") words (position independent) given center word

Summary of Skip Gram Training (Review your understanding with equations)



3. Initial words are represented in one hot vector so multiplying a **one hot vector** with $\mathbf{W}_{\mathrm{VxN}}$ will give you a 1 x N (embedded word) vector.

e.g. $[0 \;\; 1 \;\; 0 \;\; 0] \times \begin{bmatrix} 10 & 2 & 18 \\ 15 & 22 & 3 \\ 25 & 11 & 19 \\ 4 & 7 & 22 \end{bmatrix} = [15 \;\; 22 \;\; 3]$

$\boldsymbol{v_c} = \mathbf{W}_{\boldsymbol{x}} \in \mathbb{R}^{\boldsymbol{n}}$ (as there is only one input)

4. Calculate the score value for the output layer by multiplying the parameter matrix **W'**

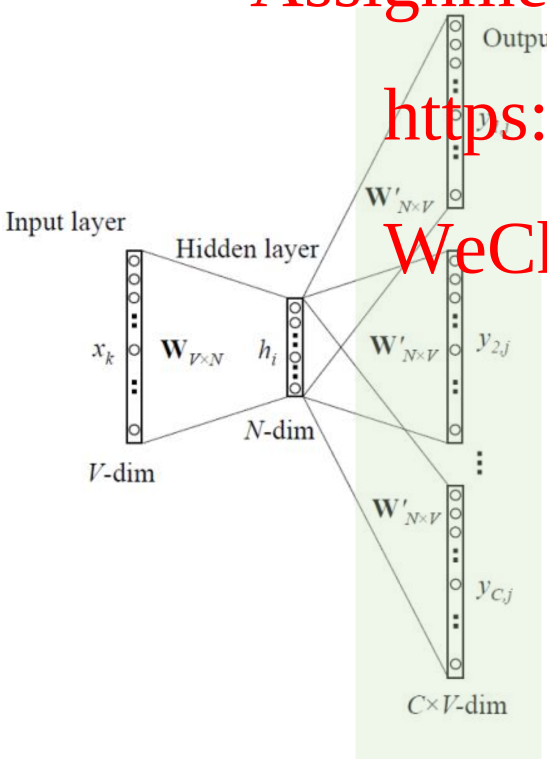
$\boldsymbol{z} = \mathbf{W}'_{\boldsymbol{v_c}}$

Prediction based Word representation

Skip Gram – Neural Network Architecture

Predict context (“outside”) words (position independent) given center word

Summary of Skip Gram Training (Review your understanding with equations)



5. Calculate the probability using softmax

$\hat{y} = softmax\ (\boldsymbol{z})$

6. Calculate 2m probabilities as we need to predict **2m** context words.

$\hat{y}_{\boldsymbol{c-m}}, \dots, \hat{y}_{\boldsymbol{c-1}}, \hat{y}_{\boldsymbol{c+1}}, \dots, \hat{y}_{\boldsymbol{c+m}}$

and compare with the ground truth (one-hot vector)

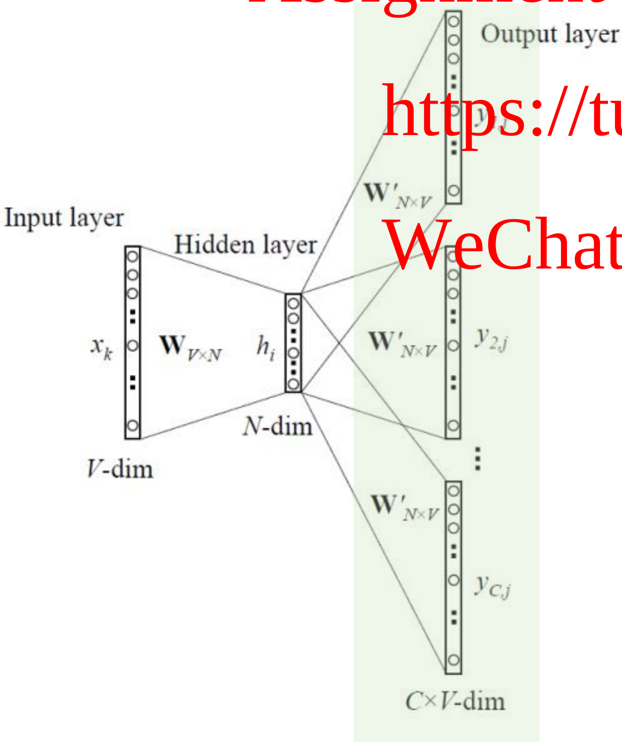
$y^{(c-m)}, \dots, y^{(c-1)}, y^{(c+1)}, \dots, y^{(c+m)}$

Prediction based Word representation

Skip Gram – Neural Network Architecture

Predict context ("outside") words (position independent) given center word

Summary of Skip Gram Training (Review your understanding with equations)



8. As in CBOW, use an objective function for us to evaluate the model. A key difference here is that we invoke a Naïve Bayes assumption to break out the probabilities. It is a strong naïve conditional independence assumption. Given the centre word, all output words are completely independent.

$$\begin{aligned}
\text{minimize } J &= -\log P(w_{c-m},\ldots,w_{c-1},w_{c+1},\ldots,w_{c+m}|w_c) \\
&= -\log \prod_{j=0,j\neq m}^{2m} P(w_{c-m+j}|w_c) \\
&= -\log \prod_{j=0,j\neq m}^{2m} \frac{\exp(u_{c-m+j}^{\top} v_c)}{\sum_{k=1}^{|V|}\exp(u_k^{\top} v_c)} \\
&= -\sum_{j=0,j\neq m}^{2m} u_{c-m+j}^{\top} v_c + 2m\log\sum_{k=1}^{|V|}\exp(u_k^{\top} v_c)
\end{aligned}$$

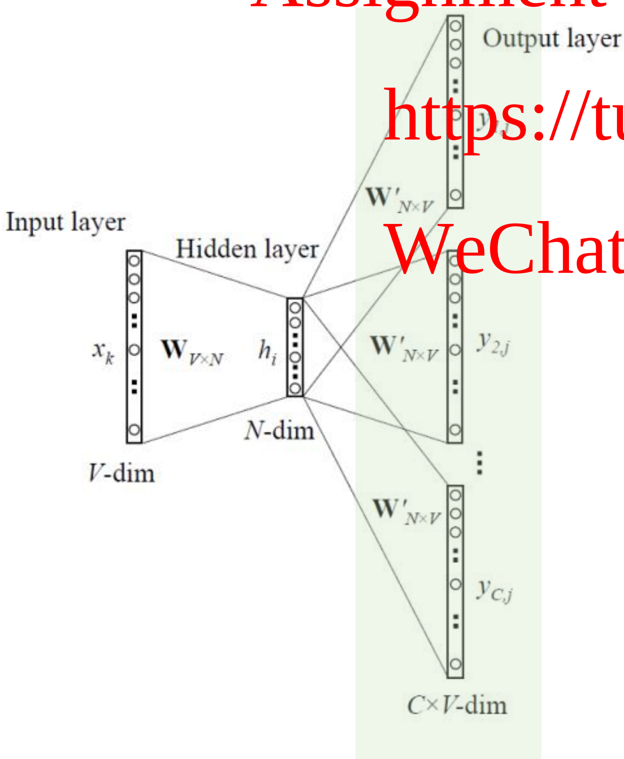
u_i=the output vector representation of word w_i

3 Prediction based Word representation

Skip Gram – Neural Network Architecture

Predict context ("outside") words (position independent) given center word

Summary of Skip Gram Training (Review your understanding with equations)



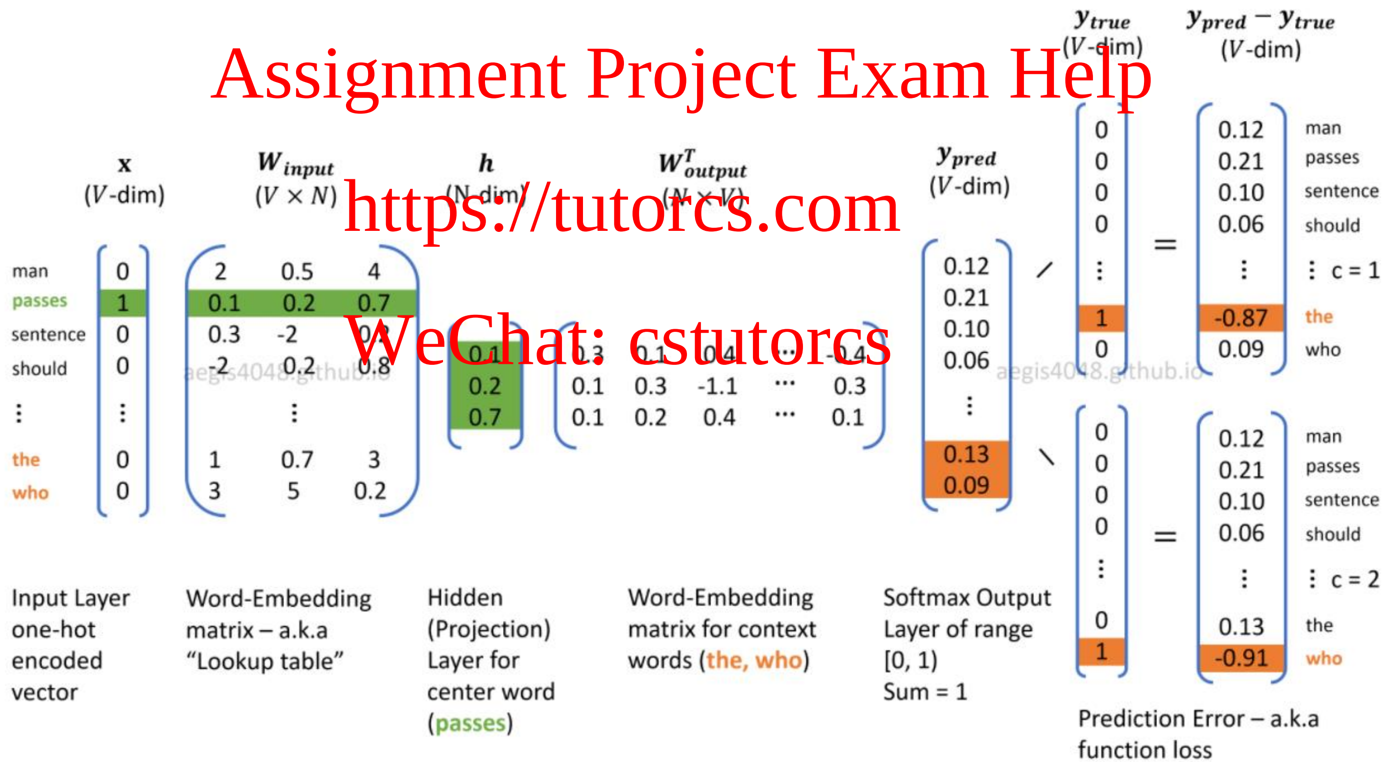
8-1. With this objective function, we can compute the gradients with respect to the unknown parameters and at each iteration update them via Stochastic Gradient Descent

$$J = -\sum_{j=0, j\neq m}^{2m} \log P(u_{c-m+j} | v_c)$$

$$= \sum_{j=0, j\neq m}^{2m} H(\hat{y}, y_{c-m+j})$$

Word2Vec-SkipGram Overview

With a simple diagram



https://aegis4048.github.io/optimize_computational_efficiency_of_skip-gram_with_negative_sampling

Key Parameter (2) for Training methods: Negative Samples (From lecture 2)

The number of negative samples is another factor of the training process.

Negative samples to our dataset – samples of words that are not neighbors

Negative sample: 2

Input word	*Output word*	*Target*
eat	mango	1
eat	exam	0
eat	tobacco	0

**1= Appeared, 0=Not Appeared*

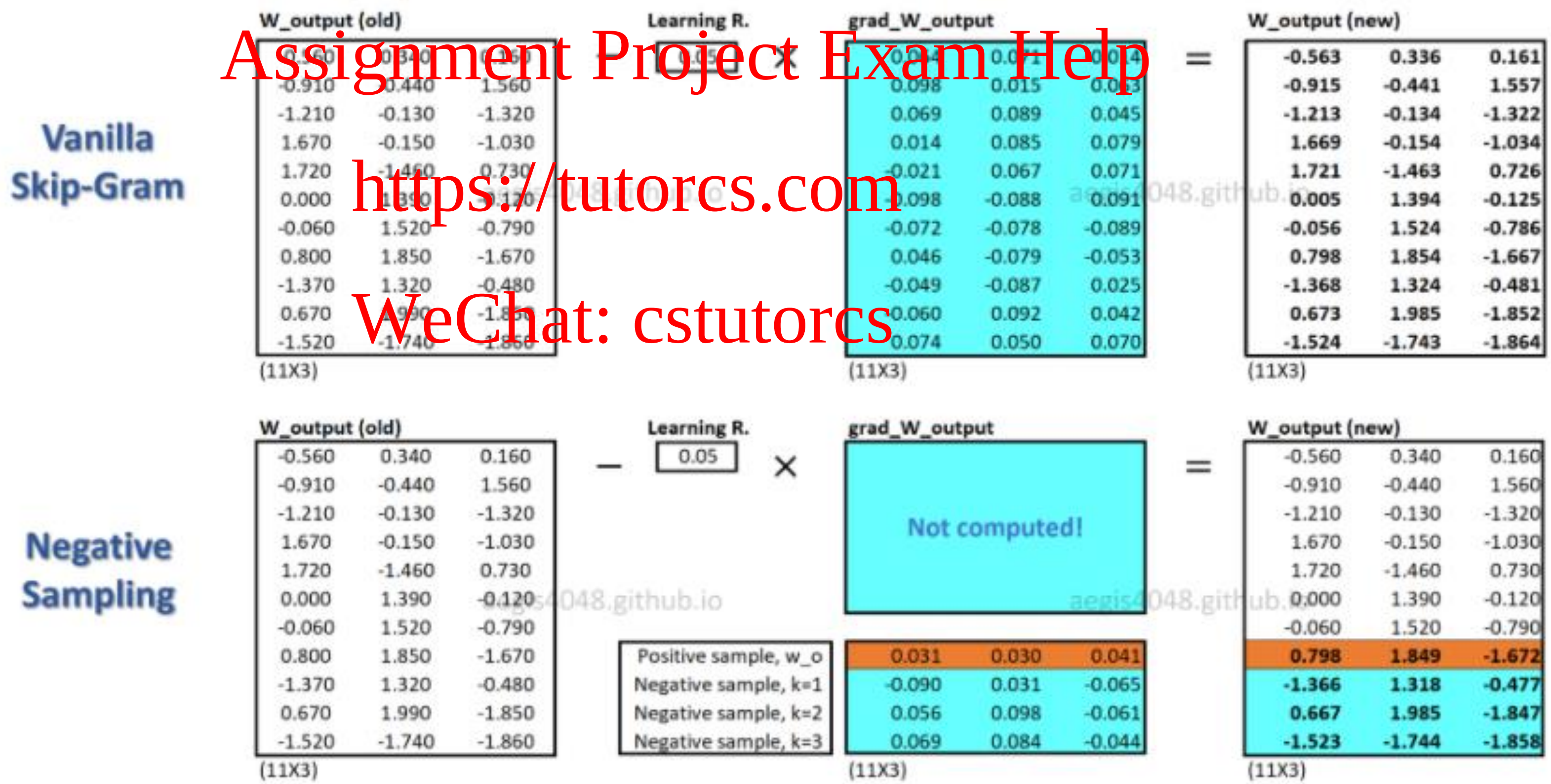
Negative sample: 5

Input word	*Output word*	*Target*
eat	mango	1
eat	exam	0
eat	tobacco	0
eat	pool	0
eat	supervisor	0

The original paper prescribes **5-20 as being a good number of negative samples**. It also states that **2-5 seems to be enough when you have a large enough dataset**.

Word2Vec-SkipGram Overview – negative sampling

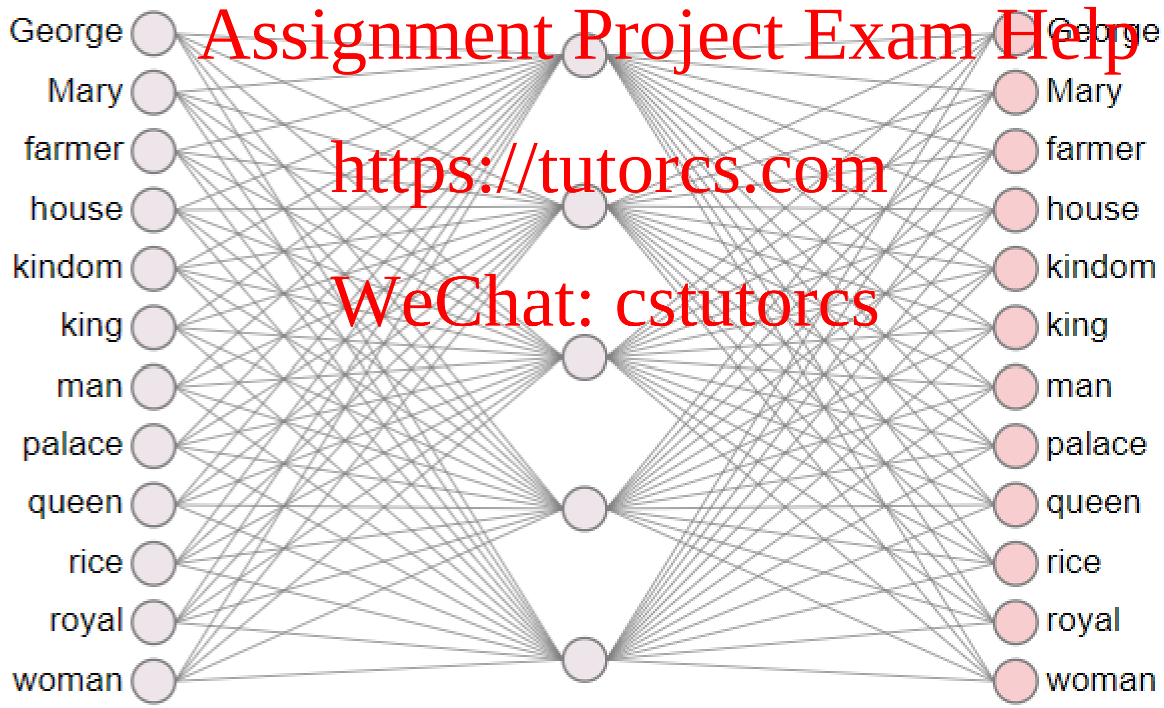
With a simple diagram



https://aegis4048.github.io/optimize_computational_efficiency_of_skip-gram_with_negative_sampling

Application

Application #1: Embedding Pretraining



http://ronxin.github.io/wevi/

0 LECTURE PLAN

THE UNIVERSITY OF SYDNEY

Lecture 3: Word Classification and Machine Learning

1. Previous Lecture: Word Embedding Review
2. Word Embedding Evaluation
3. Deep Neural Network for Natural Language Processing
   1. Perceptron and Neural Network (NN)
   2. Multilayer Perceptron
   3. Applications
4. **Next Week Preview**

   See how the Deep Learning can be used for NLP

   - Text Classification, etc.

Reference

Reference for this lecture

- Deng, L., & Liu, Y. (Eds.). (2018). Deep Learning in Natural Language Processing. Springer.
- Rao, D., & McMahan, B. (2019). Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning. " O'Reilly Media, Inc.".
- Manning, C. D., Manning, C. D., & Schütze, H. (1999). Foundations of statistical natural language processing. MIT press.
- Blunsom, P 2017, Deep Natural Language Processing, lecture notes, Oxford University
- Manning, C 2017, Natural Language Processing with Deep Learning, lecture notes, Stanford University