Vector Semantics

Lizhen Qu

Recap

- Language model.

$$P(x_1, x_2, ..., x_l) = P(x_1) \prod_{i=2}^{l} P(x_i | x_{i-1})$$

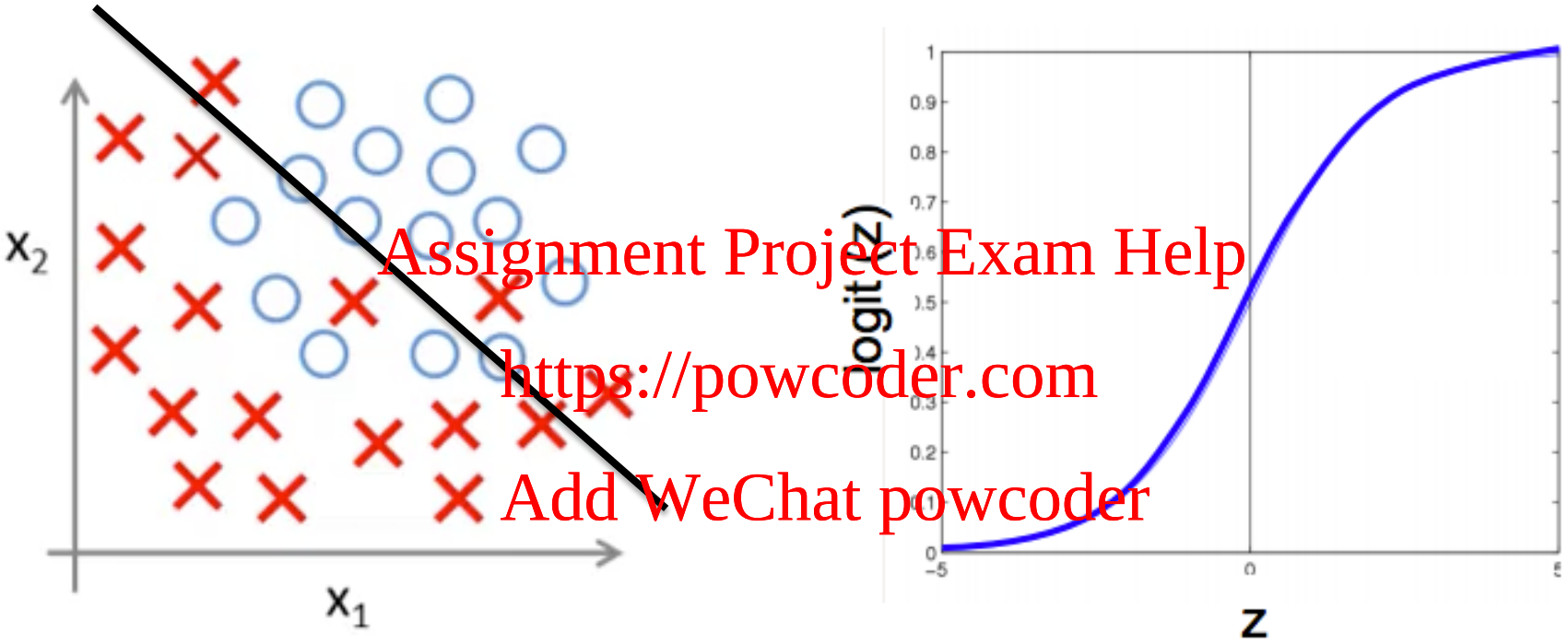
  - Kneser-Ney smoothing:
  - Stupid Backoff:

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4 S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

Overview of the NLP Lectures

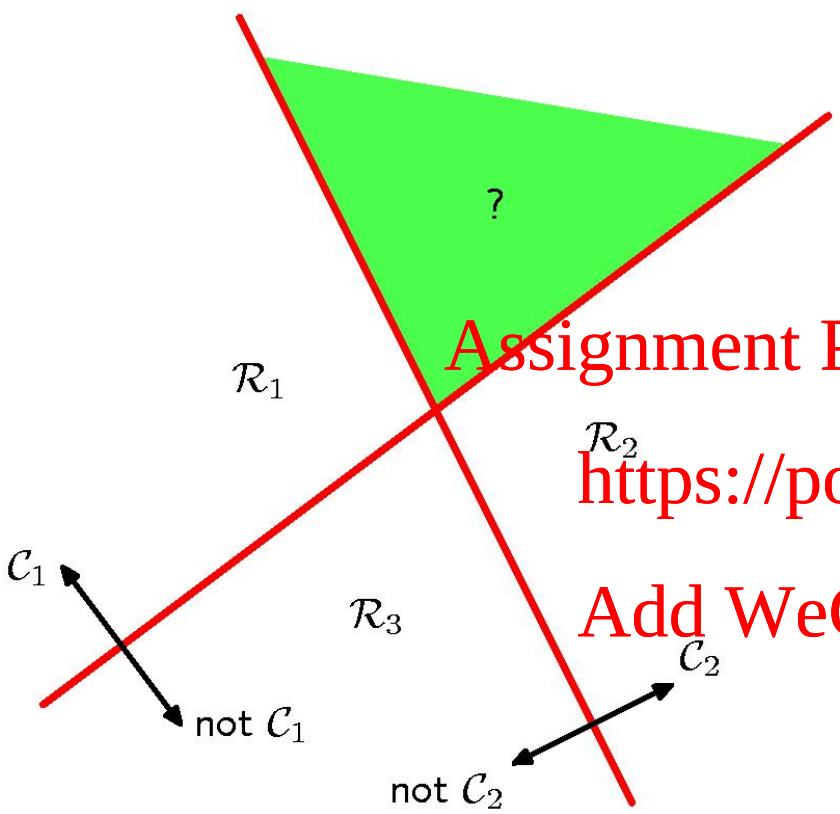
- Introduction to natural language processing (NLP).
- Regular expressions, sentence splitting, tokenization, part-of-speech tagging.
- Language models.
- Vector semantics.
  - Multiclass logistic regression.
- Parsing.
- Compositional semantics..

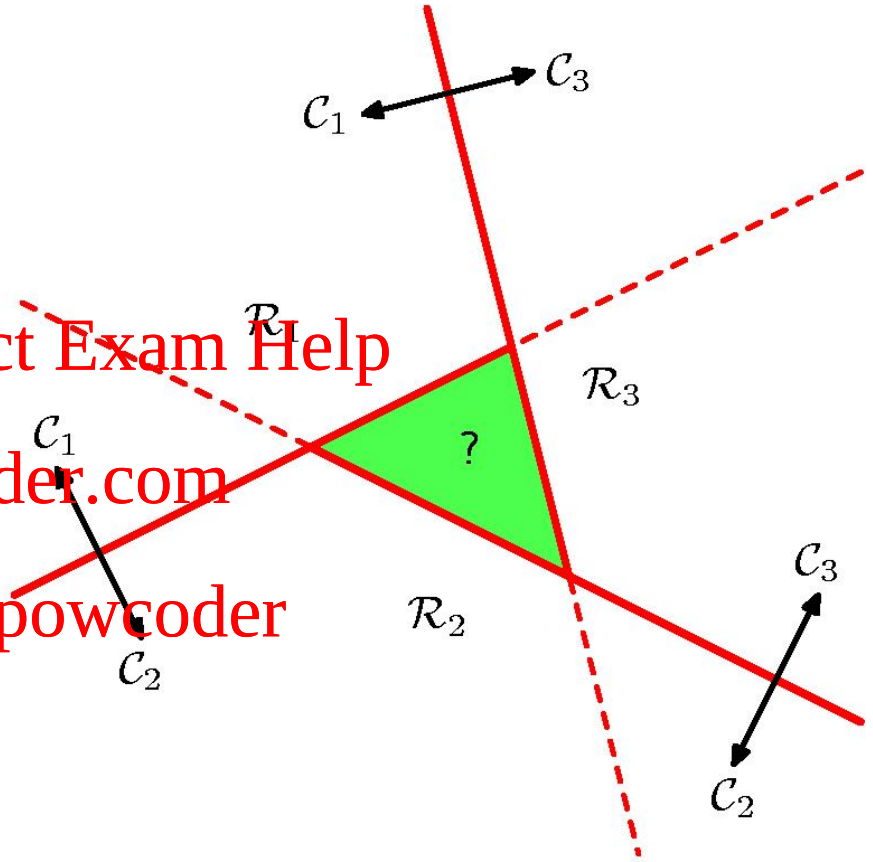
Logistic Regression for Binary Classification



$$P(Y = 1|X = \mathbf{x}) = \frac{1}{1 + \exp(-(\alpha + \sum_i \beta_i x_i))}$$

Classification of Multi-classes



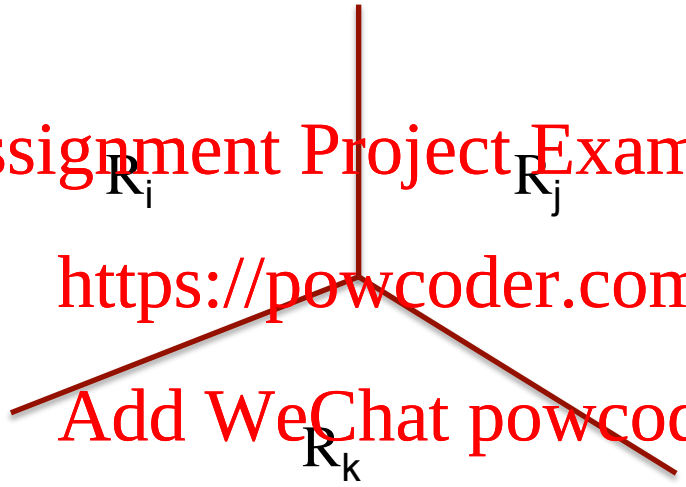


One versus rest

One versus One

When there is only

- Key idea: Use K discriminant functions $g_m(\mathbf{x})$ and pick the max.



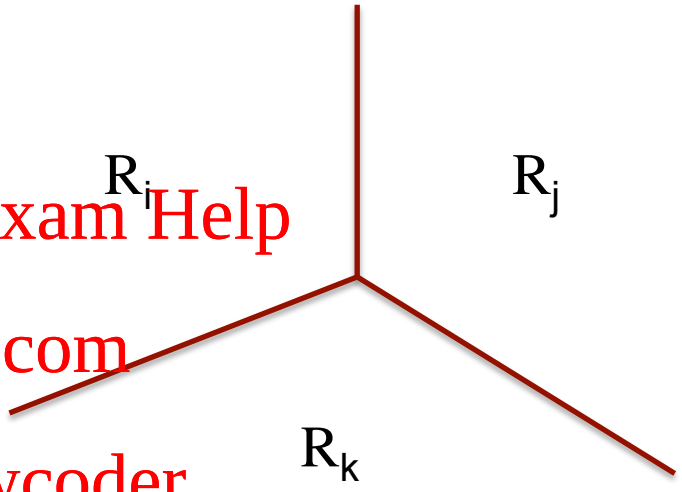
$$g_k(\mathbf{x}) > g_i(\mathbf{x}) \text{ and } g_k(\mathbf{x}) > g_j(\mathbf{x})$$

Softmax for Classification

- Definition:

$$P(Y = j|X = \mathbf{x}_i) = \frac{\exp(z_j)}{\sum_{j'=1}^{K} \exp(z_{j'})}$$

where $z_m = g_m(\mathbf{x}_i)$



$g_m(\mathbf{x})$ is a linear function so that $g_m(\mathbf{x}) = \alpha_m + \sum_k \beta_k^m x_k$.

Training of Multiclass LR

- Training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$.

$$\hat{\theta} = \arg\max_{\theta} \log[\prod_{j=1}^{N} P(y_j|\mathbf{x}_j;\theta)]$$

Rewrite each y_i with 1-of-K encoding:

if $y_1 = 3$, then $\mathbf{t}_1 = (0,0,1,0,0)$
if $y_2 = 1$, then $\mathbf{t}_2 = (1,0,0,0,0)$

Training of Multiclass LR

- Training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$.

$$\hat{\theta} = \arg\max_{\theta} \log[\prod_{j=1}^{N} P(y_j|\mathbf{x}_j;\theta)]$$

$$= \arg\max_{\theta} \log[\prod_{j=1}^{N} \prod_{k=1}^{K} P(Y=k|\mathbf{x}_j;\theta)^{t_{jk}}]$$

Training of Multiclass LR

- Training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$.

$$\hat{\theta} = \arg\max_\theta \log[\prod_{j=1}^N P(y_j|\mathbf{x}_j;\theta)]$$

$$= \arg\max_\theta \log[\prod_{j=1}^N \prod_{k=1}^K P(Y=k|\mathbf{x}_j;\theta)^{t_{jk}}]$$

$$= \arg\max_\theta \sum_{j=1}^N \sum_{k=1}^K t_{jk} \log P(Y=k|\mathbf{x}_j;\theta)$$

Cross Entropy

Loss Functions (Classification)

Given: $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$

Loss:

$$L_D(\theta) = -\log P(D|\theta)$$

$$\arg\min_\theta L_D(\theta) = \arg\max_\theta P(D|\theta)$$

Usually minimize negative log-likelihood.

Parameter Learning

Have a parametric loss function $L(w_0, w_1, ..., w_n)$

where $\theta = (w_0, w_1, ..., w_n)$

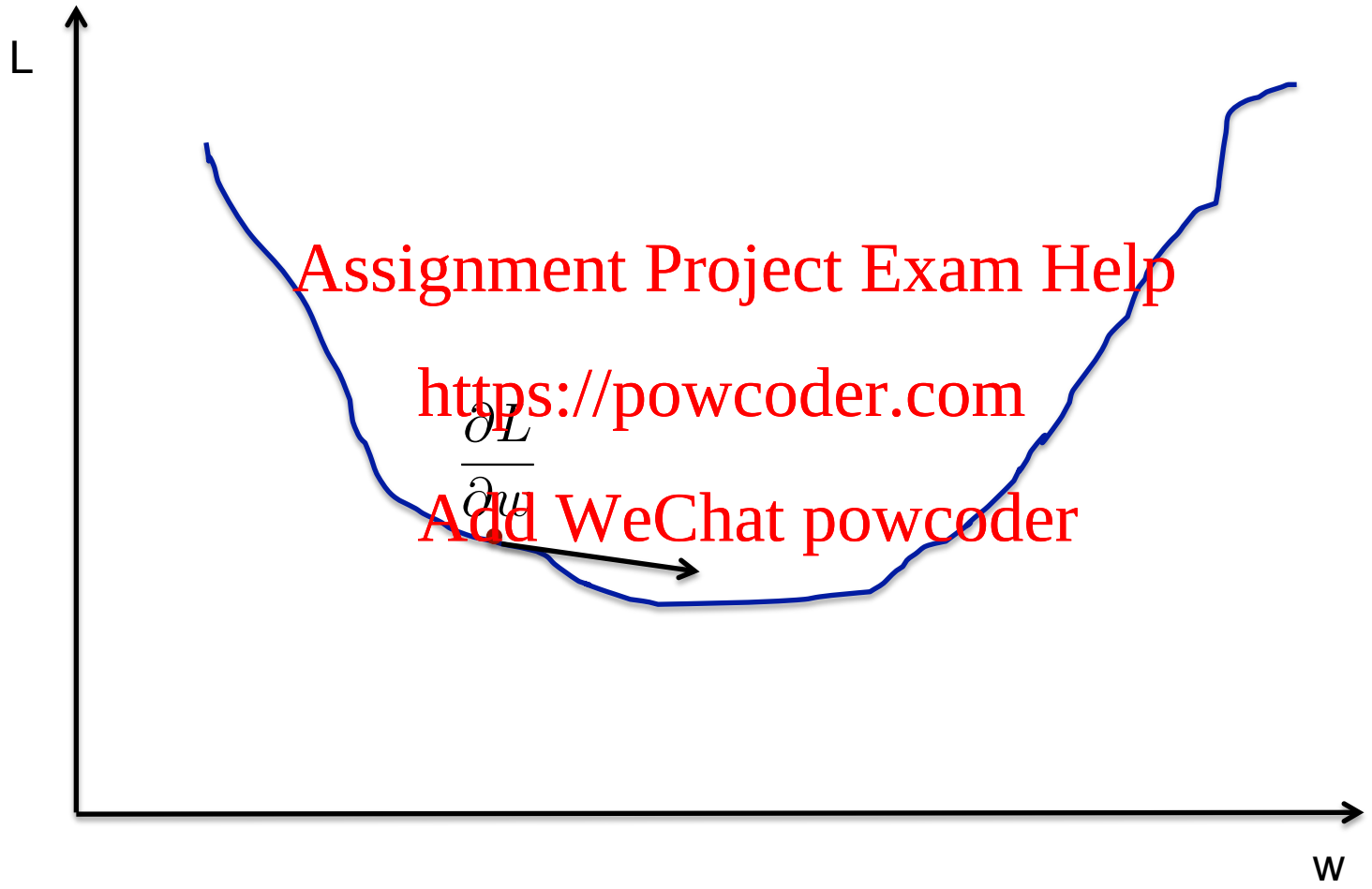
Aim to $\min_{\mathbf{w}} L(w_0, w_1, ..., w_n)$

Family of Stochastic gradient descent (SGD):

1. Start with some $w_0, w_1, ..., w_n$.
2. **Repeat**: update $w_0, w_1, ..., w_n$ to reduce $L(w_0, ..., w_n)$

   Until: reach a local minimum.

Gradient Descent



Descent Methods

Procedure:

1. Start with some $\mathbf{w}^0$.

2. Repeat: update $\mathbf{w}$ by

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \alpha^t \Delta \mathbf{w}^t$$

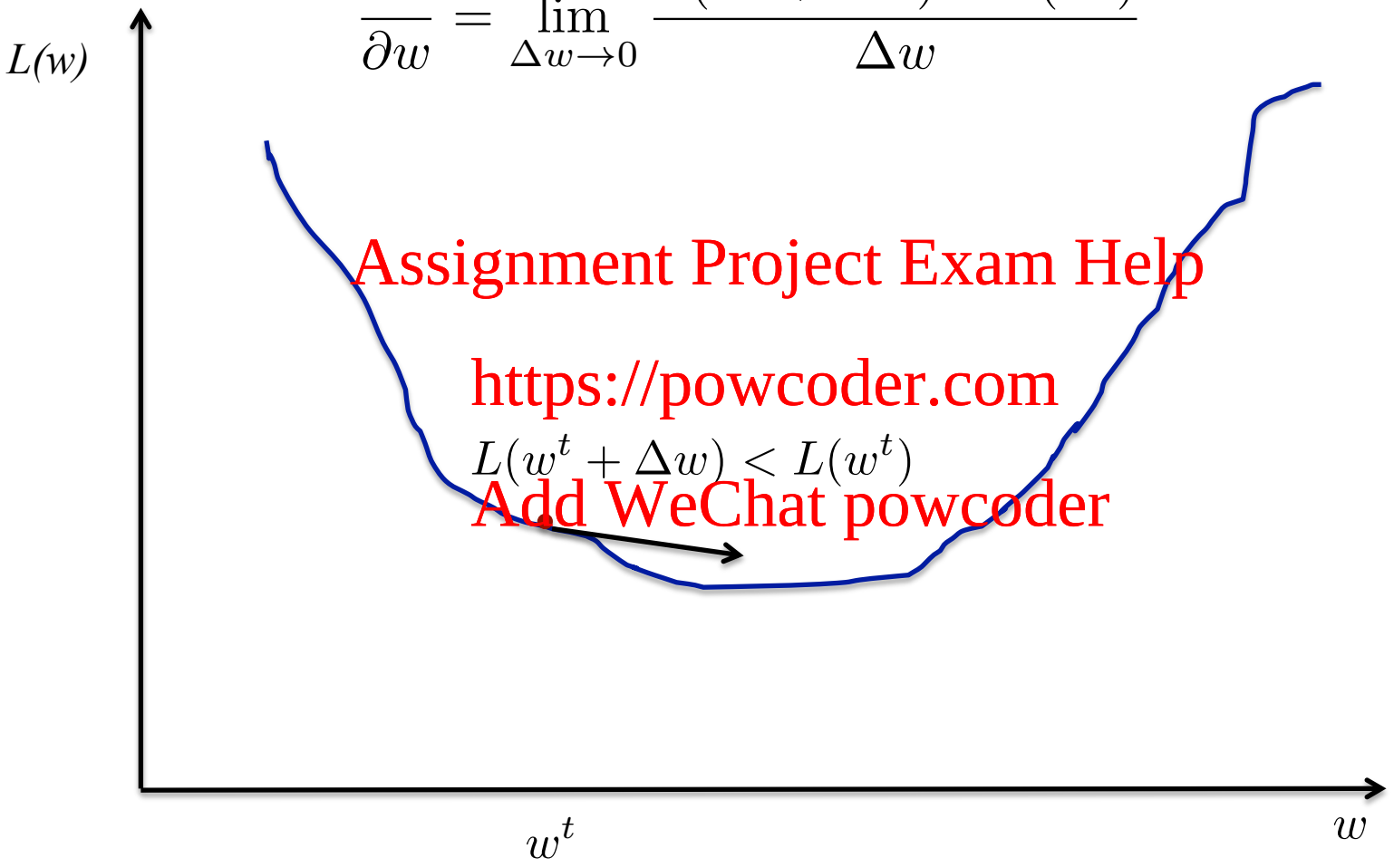
search direction

Until convergence.

step size

Gradient Descent (Univariate)

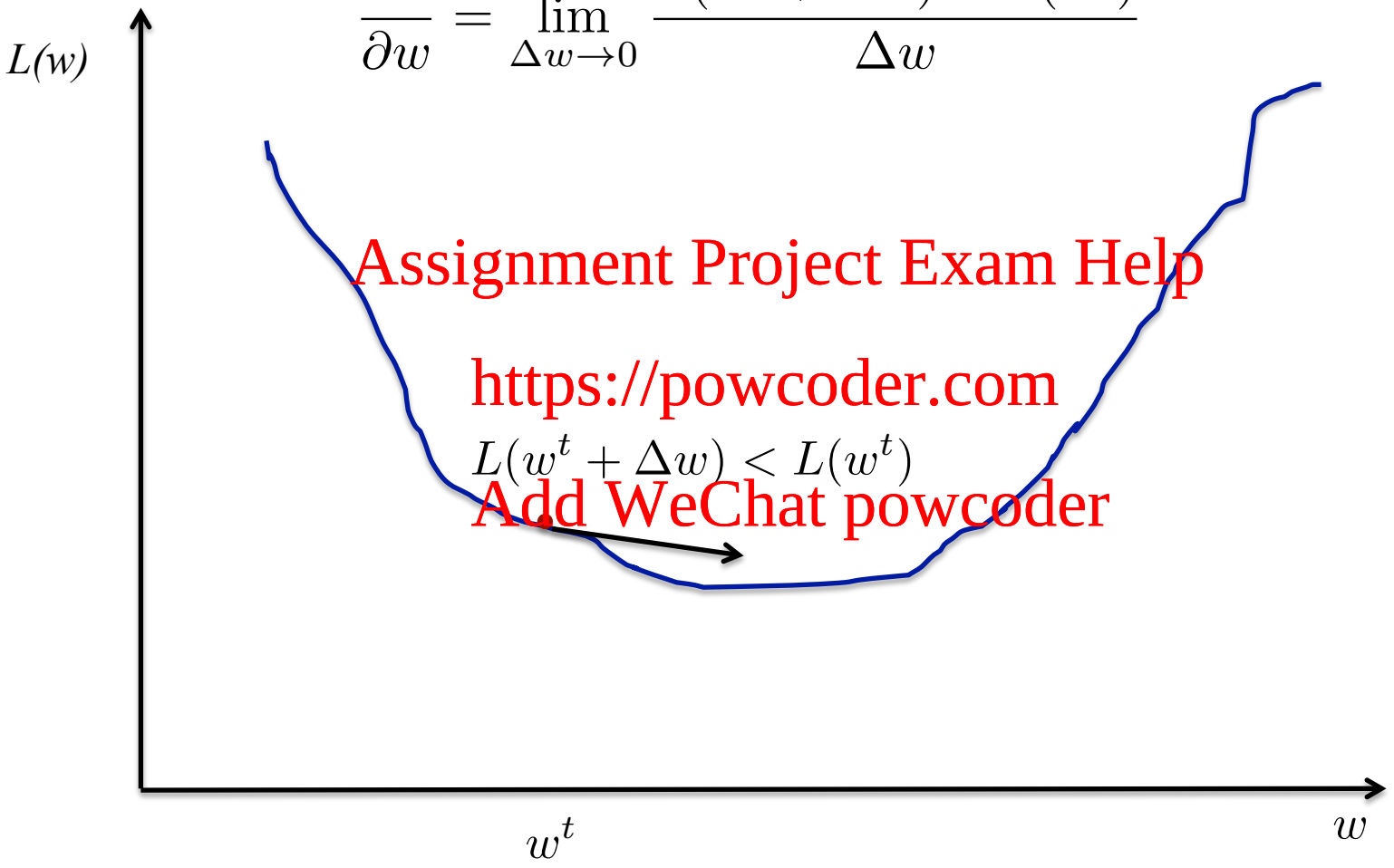
$$\frac{\partial L}{\partial w} = \lim_{\Delta w \to 0} \frac{L(w^t + \Delta w) - L(w^t)}{\Delta w}$$



$L(w^t + \Delta w) < L(w^t)$

Gradient Descent (Univariate)

$$\frac{\partial L}{\partial w} = \lim_{\Delta w \to 0} \frac{L(w^t + \Delta w) - L(w^t)}{\Delta w}$$

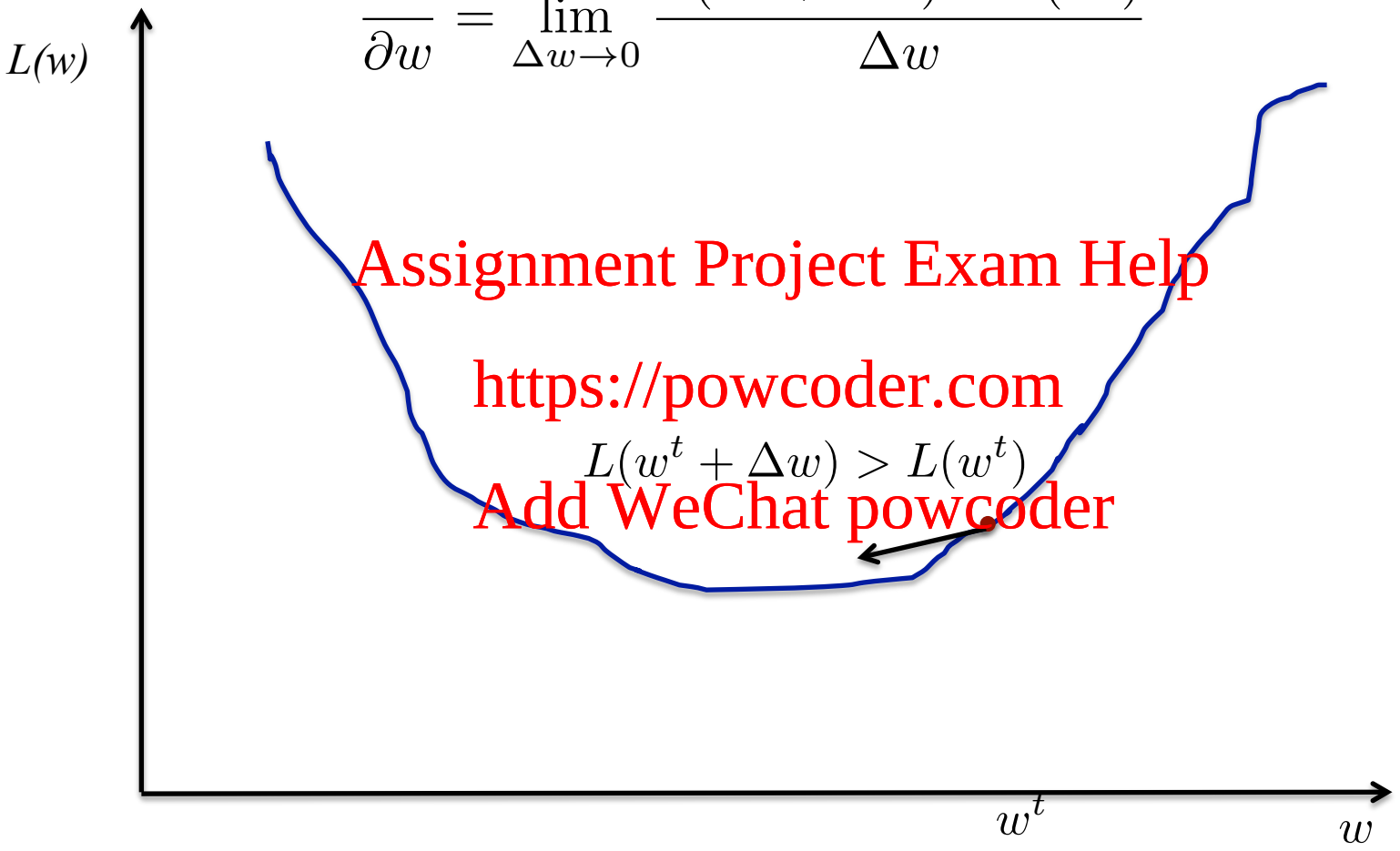


$L(w^t + \Delta w) < L(w^t)$

$$w^t - \alpha(\frac{\partial L}{\partial w}) > w^t$$

Gradient Descent (Univariate)

$$\frac{\partial L}{\partial w} = \lim_{\Delta w \to 0} \frac{L(w^t + \Delta w) - L(w^t)}{\Delta w}$$



$$L(w^t + \Delta w) > L(w^t)$$

$$w^t - \alpha(\frac{\partial L}{\partial w}) < w^t$$

Gradient Descent

Procedure:

1. Start with some $\mathbf{w}^0$.
2. Repeat: update $\mathbf{w}$ by

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha^t \frac{\partial L(\mathbf{w}^t)}{\partial \mathbf{w}}$$

Until convergence.

Gradient Descent

Procedure:

1. Start with some $\mathbf{w}^0$.

2. Repeat: update $\mathbf{w}$ by

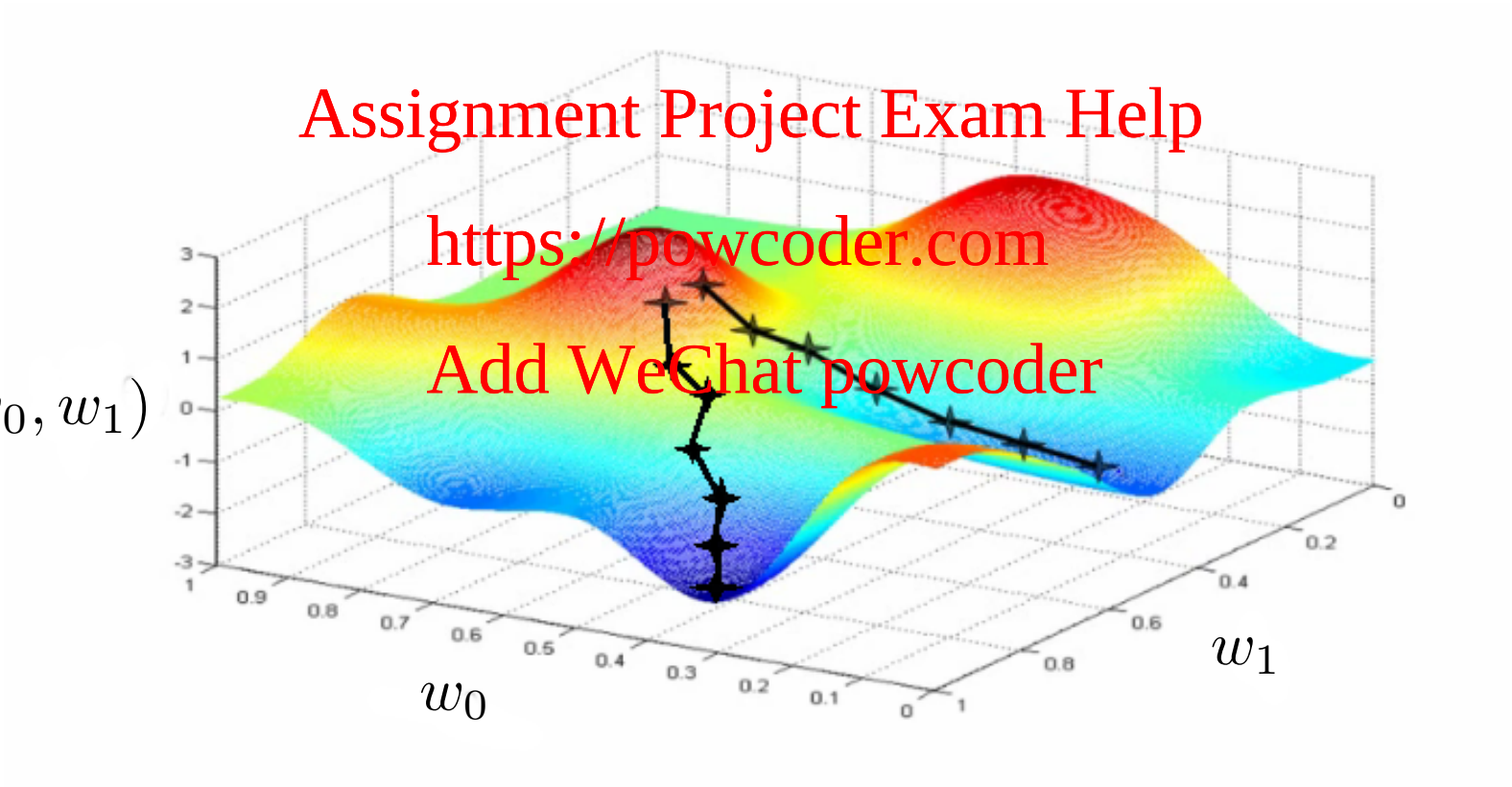
$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha^t \frac{\partial L(\mathbf{w}^t)}{\partial \mathbf{w}}$$

Until convergence.

Example: $\frac{L(\mathbf{w}^t)}{\mathbf{w}} = \frac{1}{N}\sum_{i=1}^{N}((\mathbf{w}^t)^{\mathrm{T}}\mathbf{x}_i - y_i)\mathbf{x}_i$

Search for Optimal Parameters

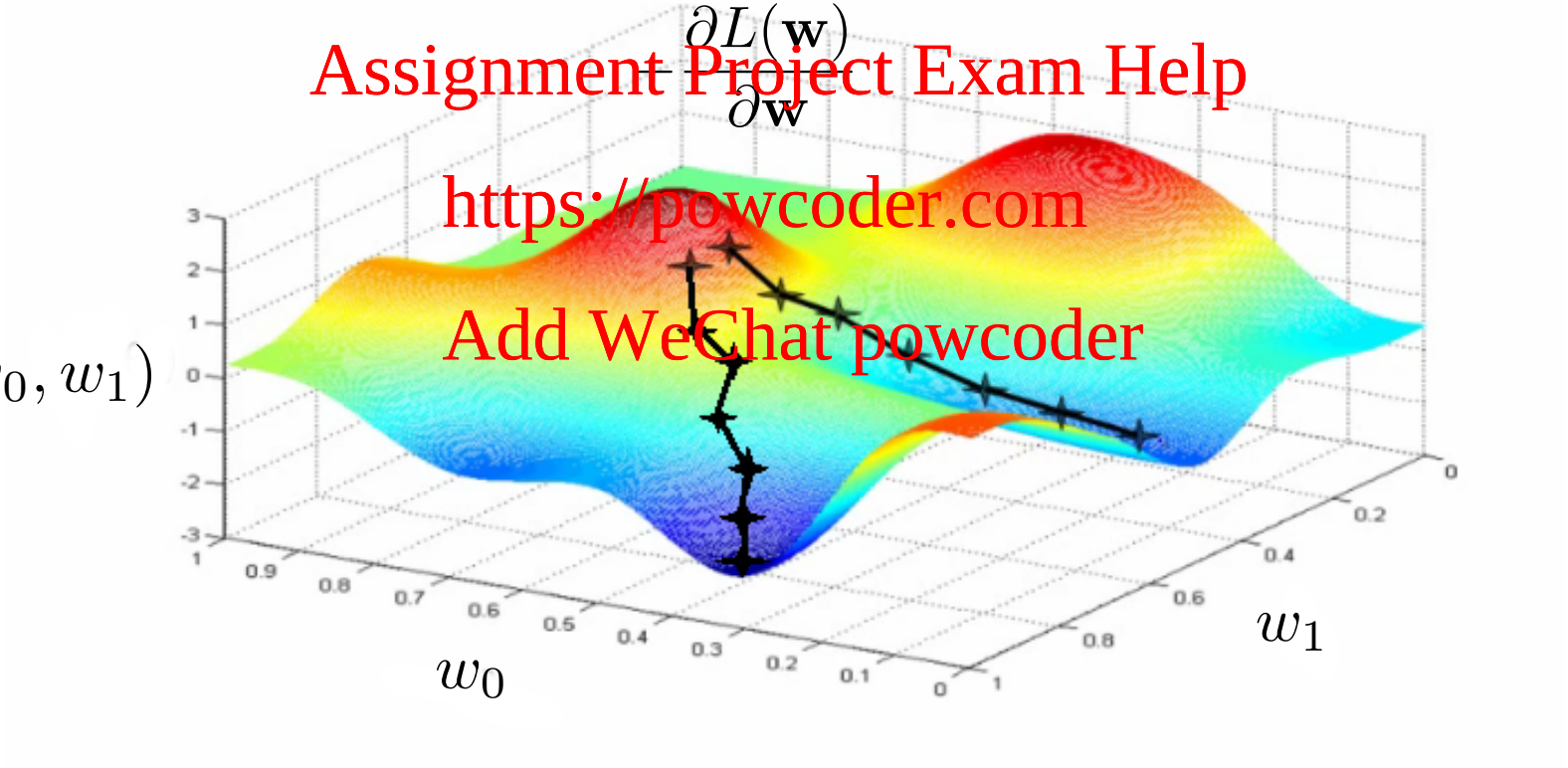
$$\min_{\mathbf{w}} L(\mathbf{w})$$



Gradient Descent

$$\min_{\mathbf{w}} L(\mathbf{w})$$

$$-\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$$



SGD with Mini-Batch

- Use mini-batch:

REPEAT k epochs:

SHUFFLE training dataset

FOR each batch in training dataset:

$$w_j^{t+1} = w_j^t - \alpha \frac{1}{|B|} \sum_{x_i \in B} \frac{\partial}{\partial w_j} L(\theta)$$

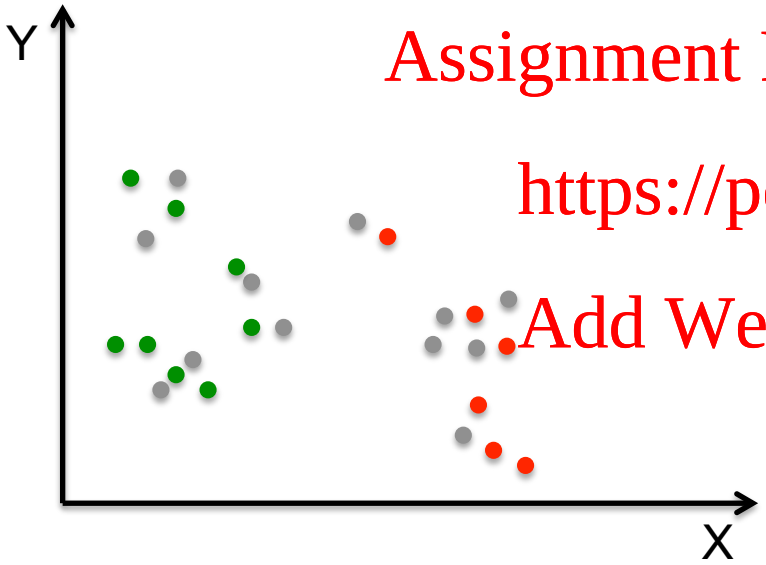
early stopping evaluated on validation set

Overview of the NLP Lectures

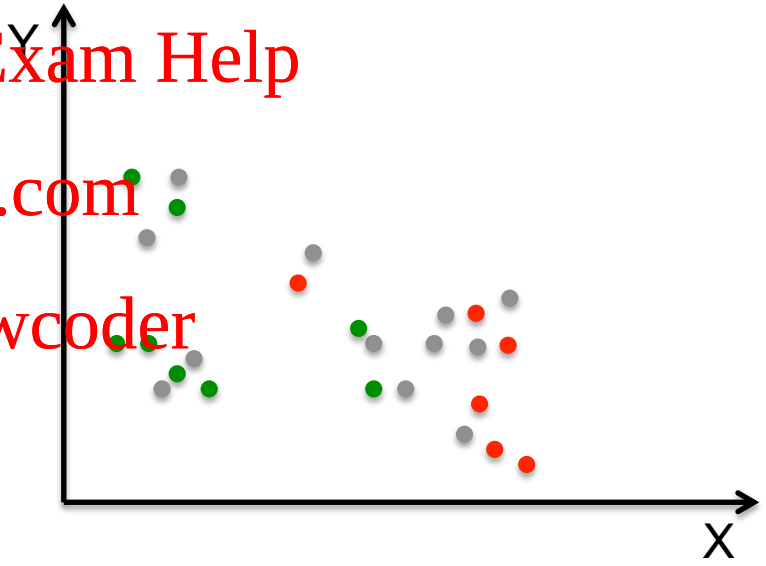
- Introduction to natural language processing (NLP).
- Regular expressions, sentence splitting, tokenization, part-of-speech tagging.
- Language models.
- Vector semantics.
  - Multiclass logistic regression.
  - Feedforward neural networks.
- Parsing.
- Compositional semantics.

Nonlinearity

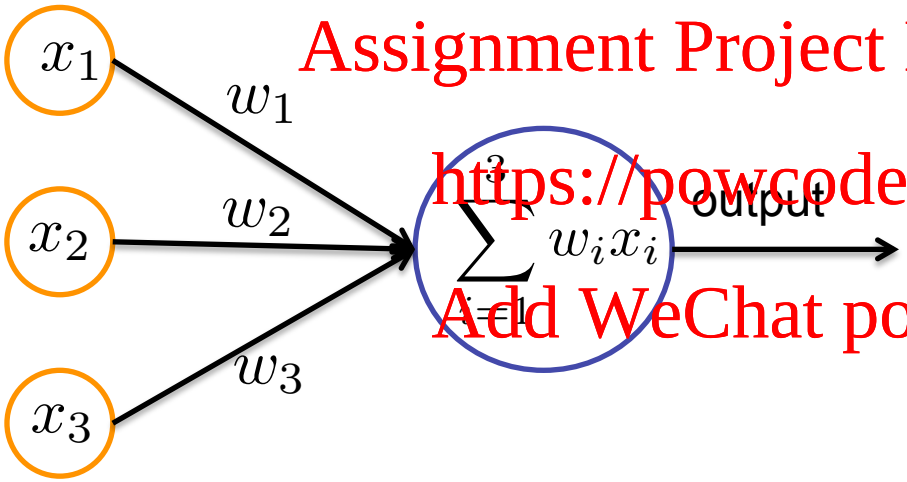
Linear Classification



Nonlinear Classification

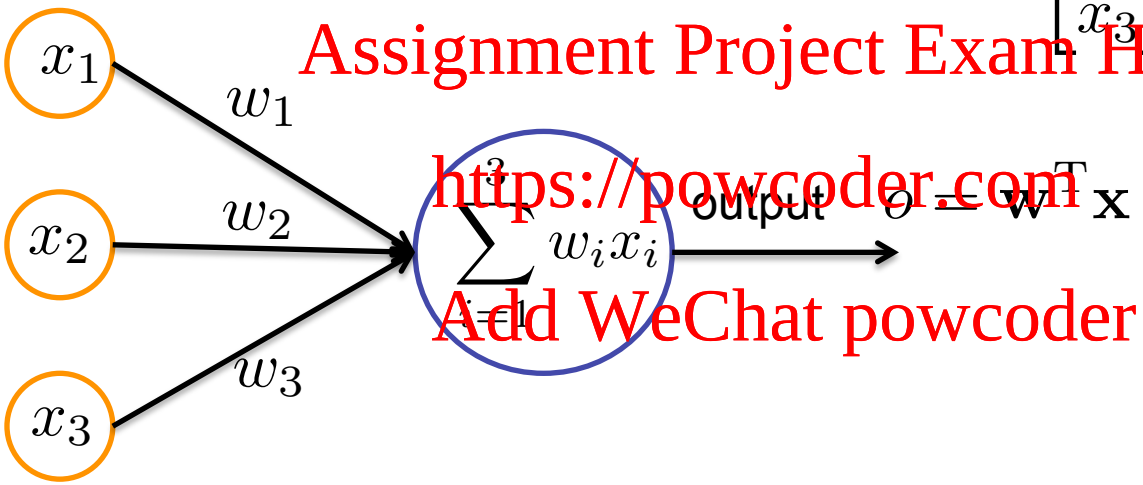


Artificial Neuron



Artificial Neuron

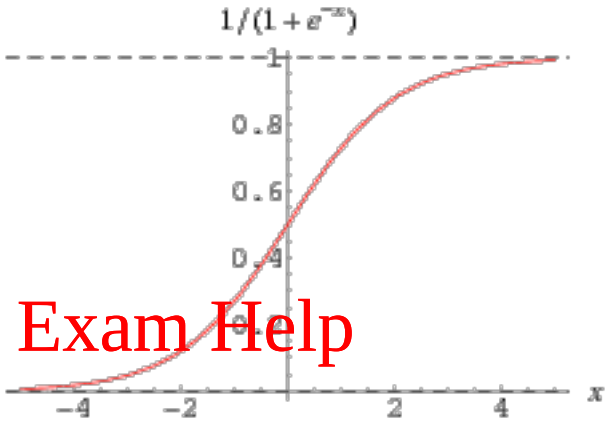
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

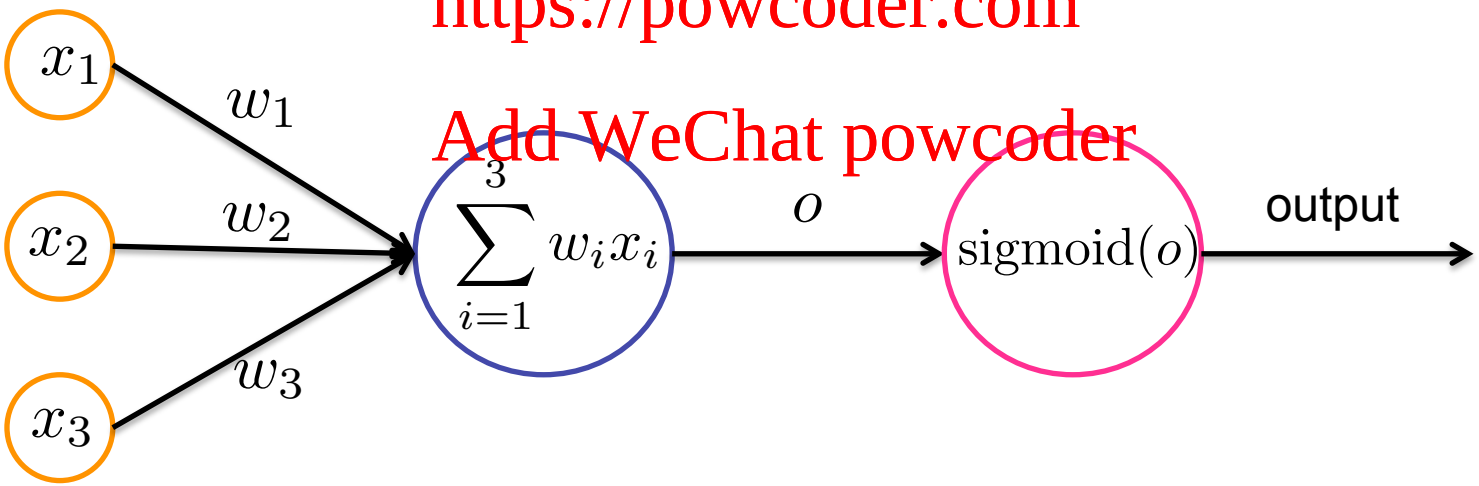


Activation Function

- Sigmoid function

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$

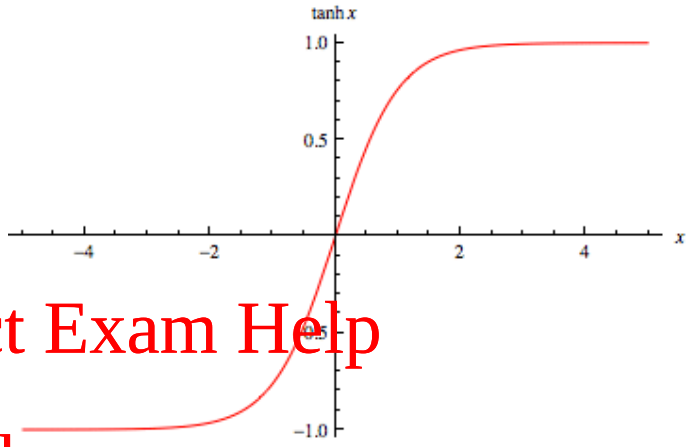




Activation Function

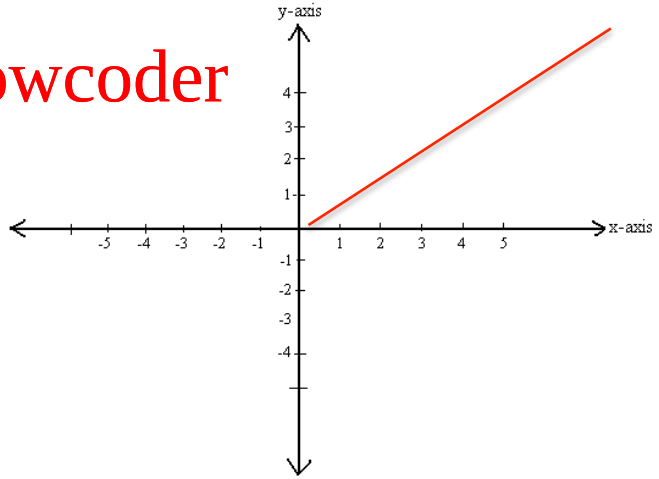
- Hyperbolic tangent

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



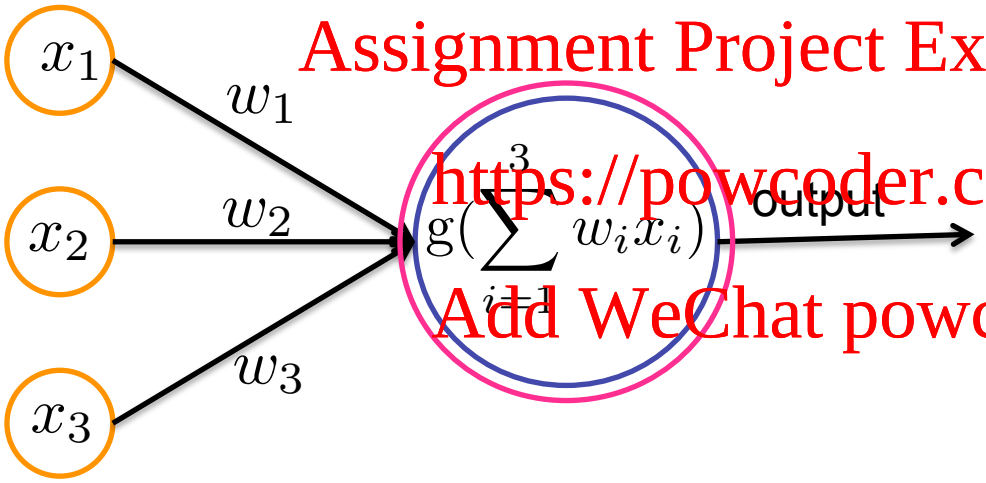
- Linear rectifier

$$\text{rectifier}(x) = \max(0, x)$$

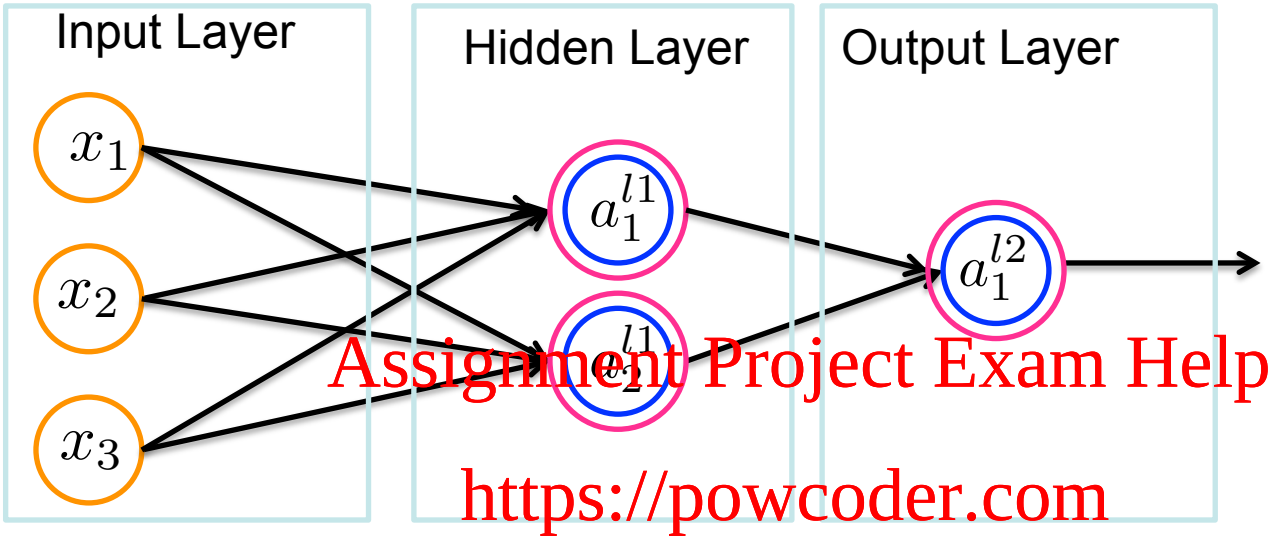


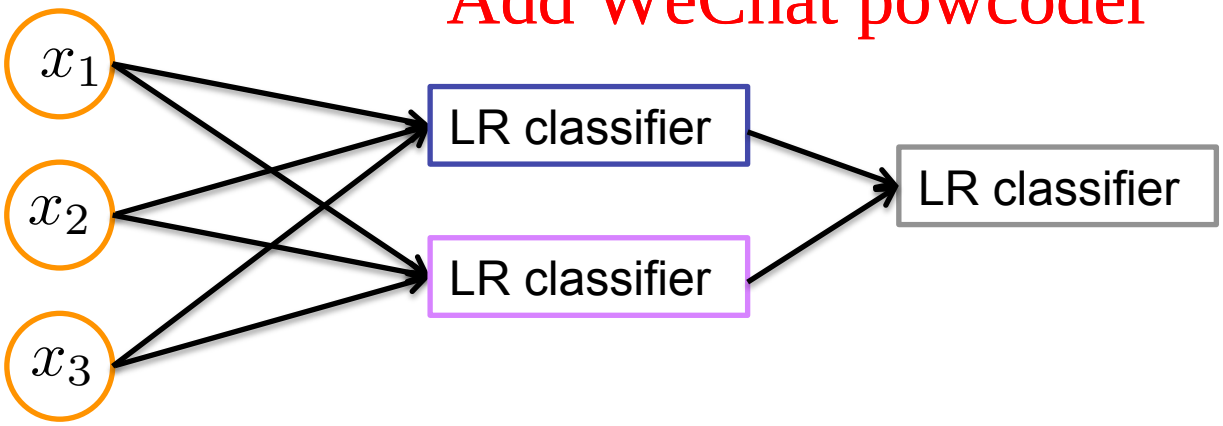
Artificial Neuron (Compact form)

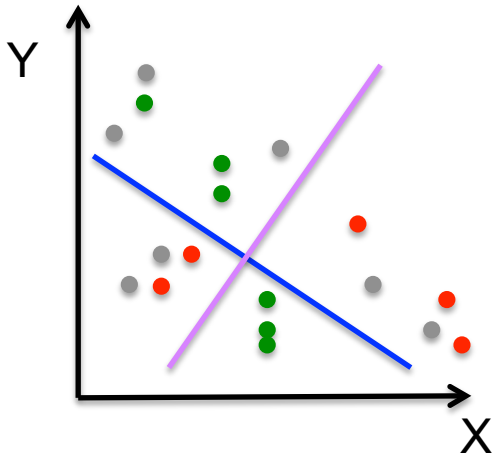
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$



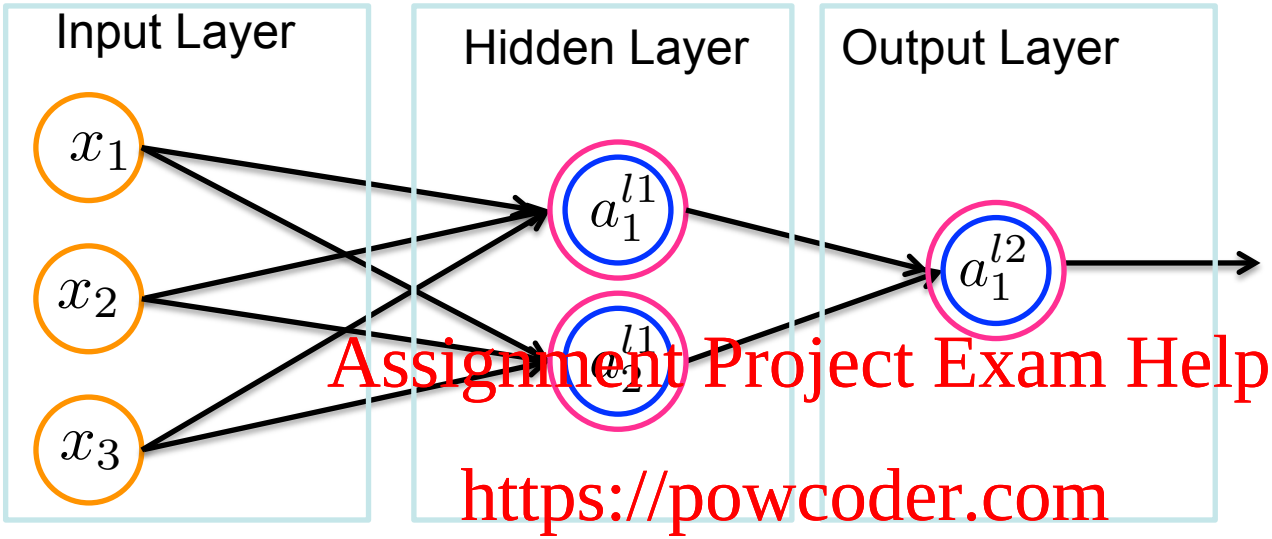
Feedforward Networks







Forward Propagation



$$a_1^{l1} = g(w_{11}^{l0}x_1 + w_{21}^{l0}x_2 + w_{31}^{l0}x_3)$$

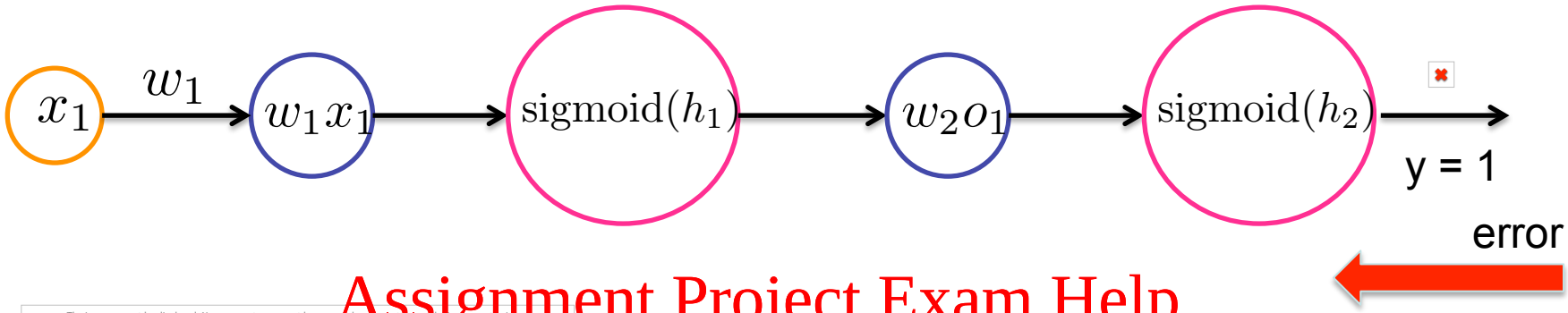
$$a_2^{l1} = g(w_{12}^{l0}x_1 + w_{22}^{l0}x_2 + w_{32}^{l0}x_3)$$

$$\Rightarrow \quad a_1^{l2} = g(w_{12}^{l1}a_1^{l1} + w_{22}^{l1}a_2^{l1})$$

Composite Function: $\mathbf{a}_1^{l2} = g(\mathbf{W}^{l1}g(\mathbf{W}^{l0}\mathbf{x}))$

Forward propagation

Computing Derivatives



Chain Rule
BackPropagation

Overview of the NLP Lectures

- Introduction to natural language processing (NLP).
- Regular expressions, sentence splitting, tokenization, part-of-speech tagging.
- Language models.
- Vector semantics.
  - Multiclass logistic regression.
  - Feedforward neural networks.
  - Word embeddings.
- Parsing.
- Compositional semantics and NLP applications.

Weaknesses of Discrete Representations

- Missing new words.
- Hard to compute word similarity.
  - Similarity between *Canberra* and *Paris*?
- Require human labor to create and maintain.

Distributional Similarity

"You shall know a word by the company it keeps"

(J. R. Firth 1957: 11)

'study in the united states'
'live in New Zealand',
'study in the january 10',
'stay in the New England',
'live in the US',
'study in the United Kingdom'

'work in the United States',
'work in Australia.',
'its meeting on 05 January',
'annual meeting on 01 December',
'a meeting on January NUM',
'ordinary meeting on 9 December',
'regular meeting of February 10'

Word Co-occurrence Counts

	get	see	use	hear	eat	kill
knife	51	20	84	0	3	0
cat	52	58	4	4	6	26
dog	115	83	10	42	33	17
boat	59	39	23	4	0	0
cup	98	14	6	2	1	0
pig	12	17	3	2	9	27
banana	11	2	2	0	18	0

Neural Word Representations

- Using continuous-value vectors.
- Learned from unlabeled data.
- Capture distributional similarity.

"You shall know a word by the company it keeps"

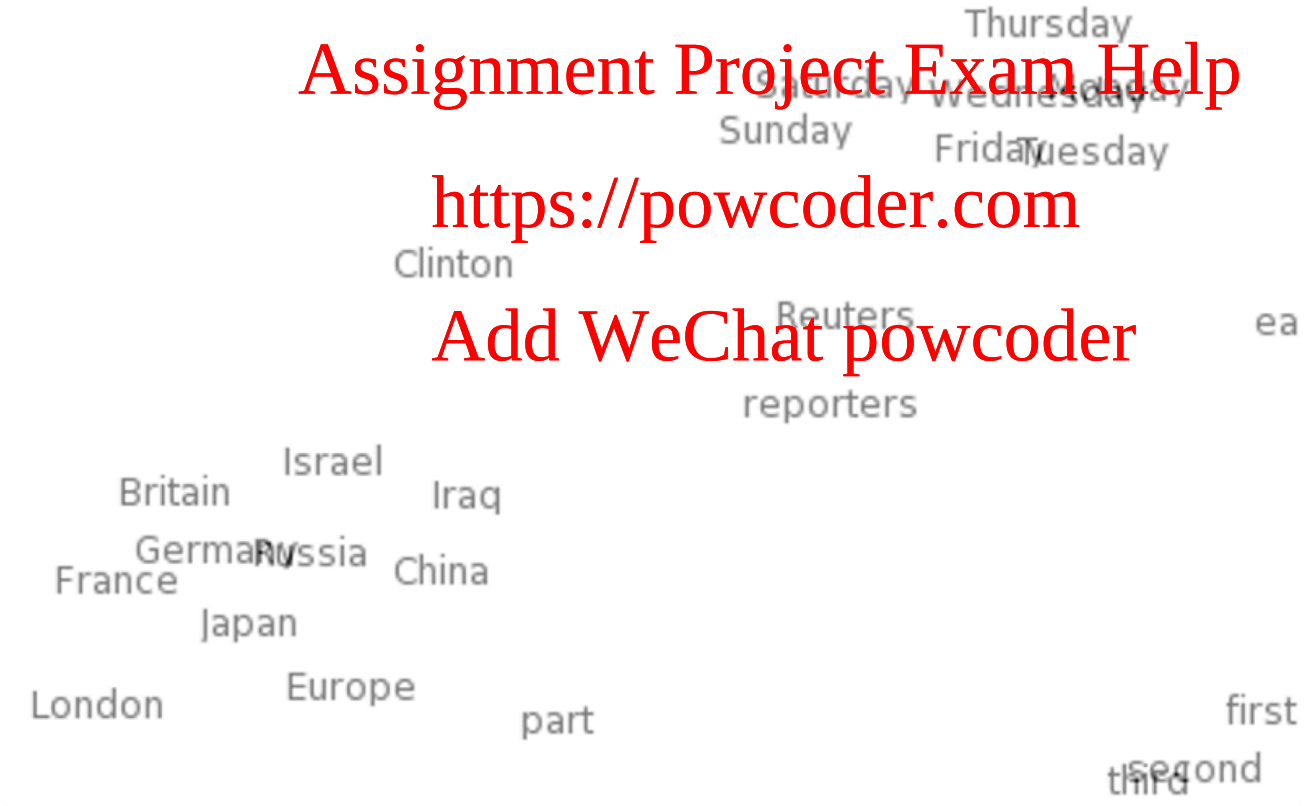
(J. R. Firth 1957: 11)

linguistics =

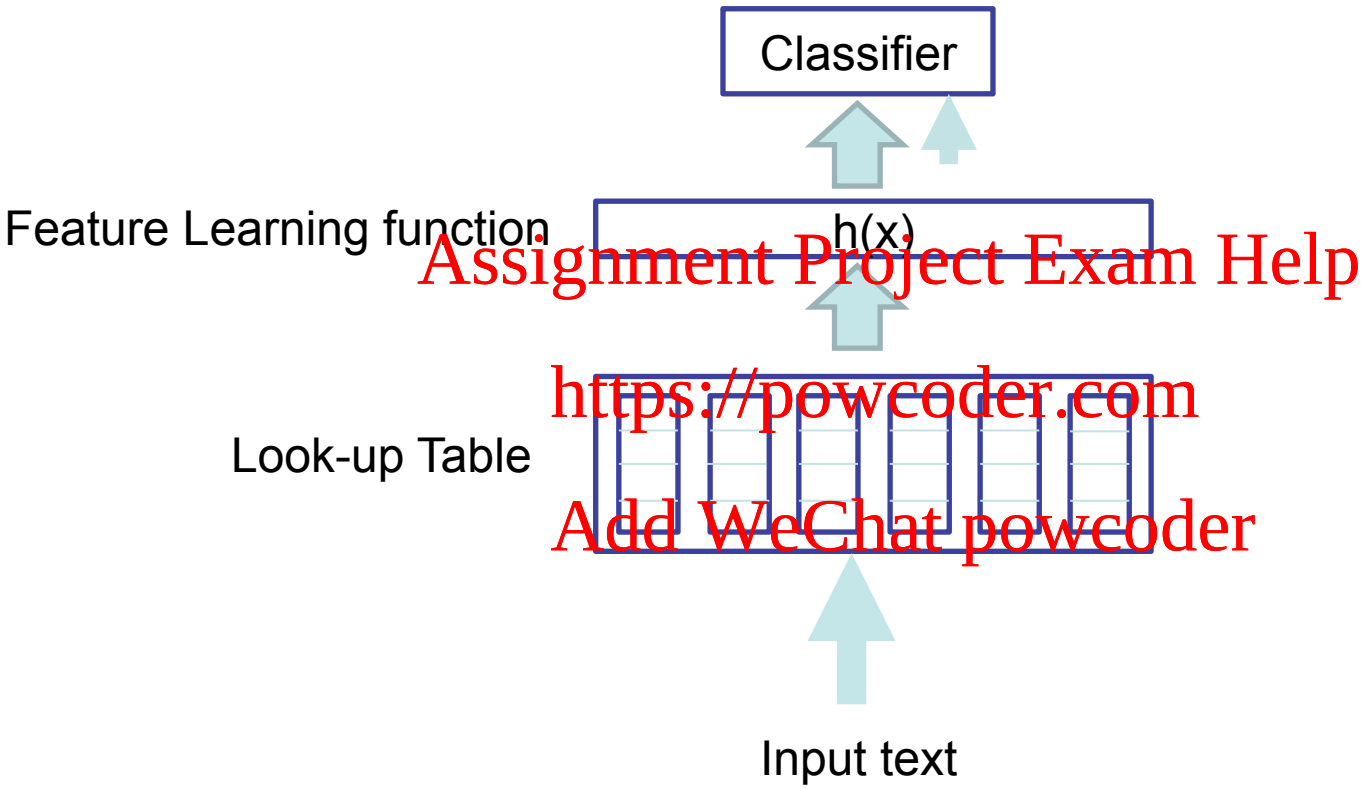
$$\begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Learning Word Representation

Paris – Fance + Italy = Rome
Android – Google + Microsoft = Windows



Neural Networks with Word Embeddings



Learning Word Embeddings

- Predict the word in the middle given context words.

$$\prod_{i=1}^{N} P(x_i | x_{i-k}, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{i+k})$$

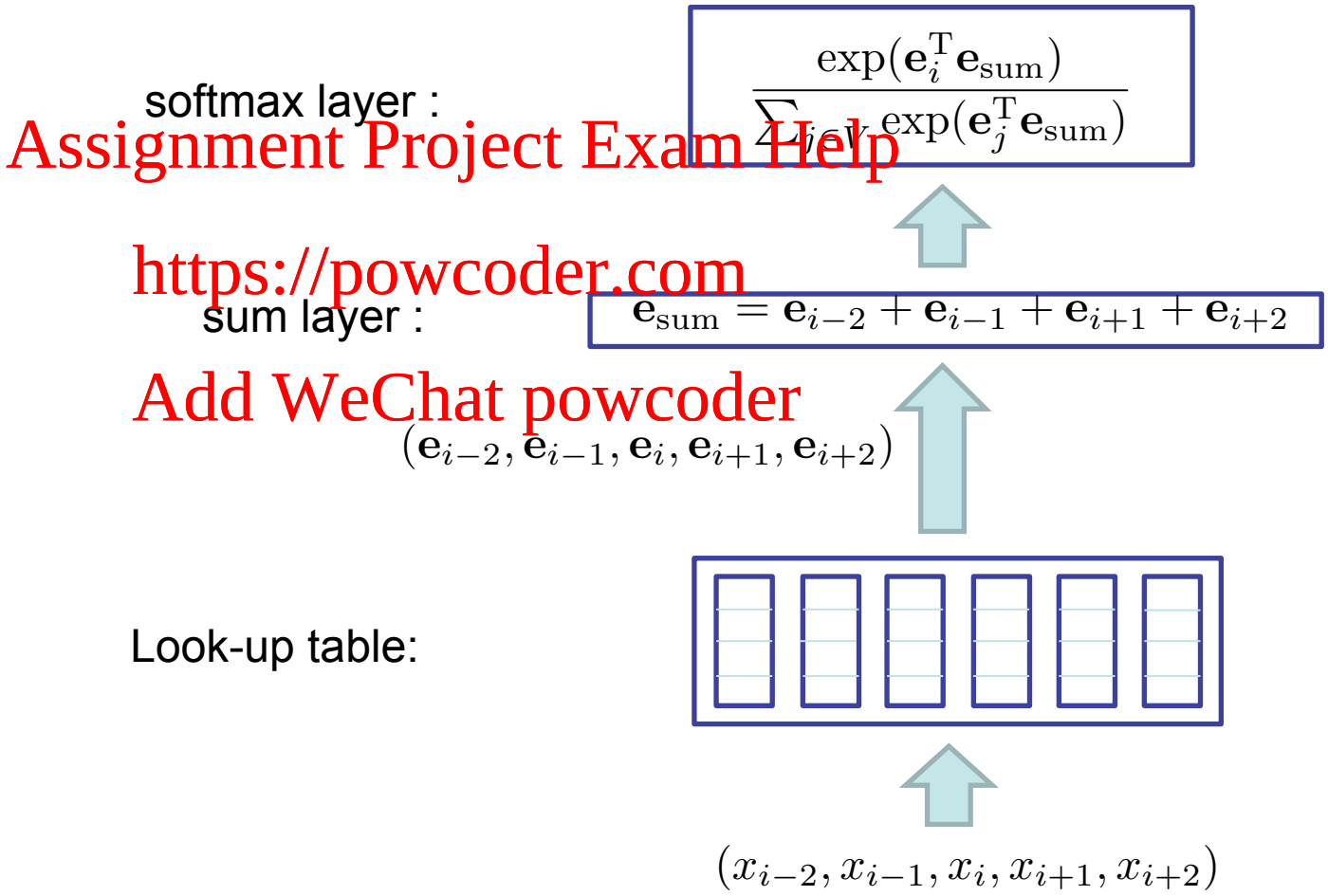
k is the size of the context window.

Continuous Bag-of-Words Model (CBOW) [1,2]

- Basic form: $P(x_i|x_{i-c},...,x_{i-1},x_{i+1},...,x_{i+c}) = \frac{\exp(\mathbf{e}_i^{\mathrm{T}}\mathbf{e}_{\mathrm{sum}})}{\sum_{j\in V}\exp(\mathbf{e}_j^{\mathrm{T}}\mathbf{e}_{\mathrm{sum}})}$

where c is the context size and V is the vocabulary.

Let $c = 2$



High Scalability with Negative Sampling

- Approximate the softmax loss with a set of binary classification losses.
  - Positive example: the loss of seeing word i given context words.
  - Negative examples : the loss of not observing $|M|$ word j given context words, where M is a set of randomly picked words from vocabulary.

$$L_{\text{neg\_sam}}(x_i) = -\log \sigma(\mathbf{e}_i^{\mathrm{T}} \mathbf{e}_{\text{sum}}) - \sum_{u \in M} \log(1 - \sigma(\mathbf{e}_u^{\mathrm{T}} \mathbf{e}_{\text{sum}}))$$

Positive example

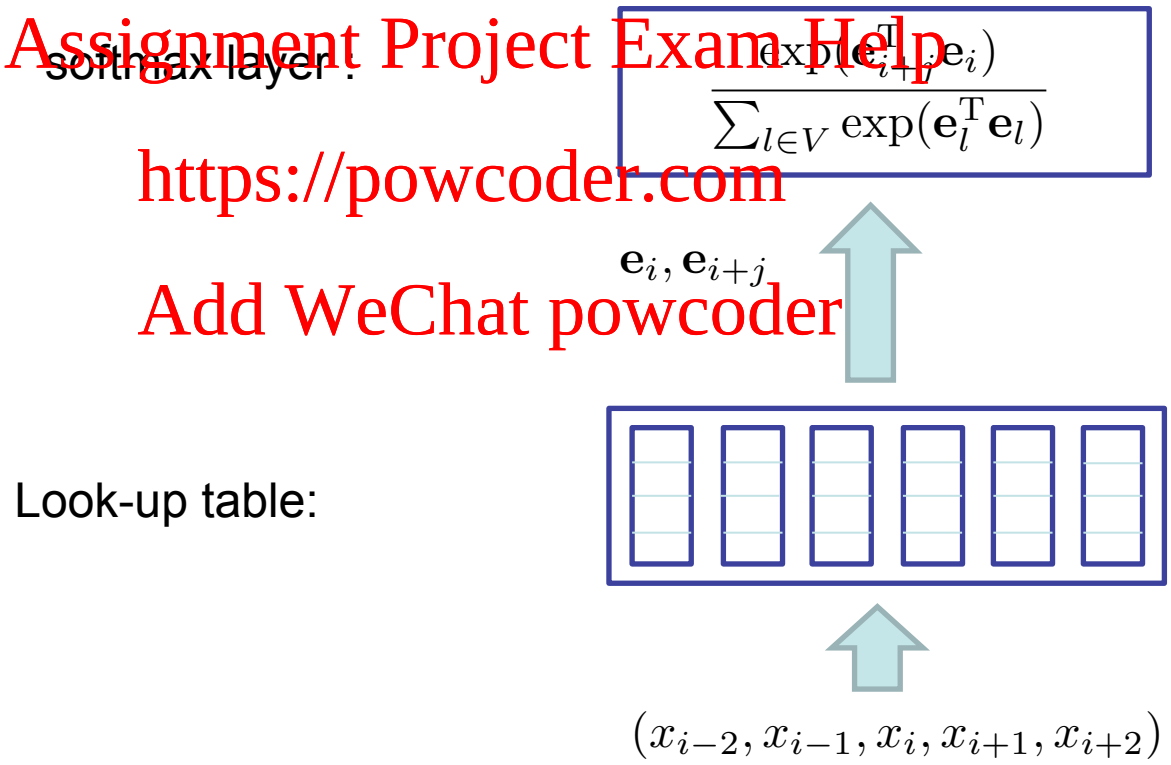
negative examples

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

Skip-Gram [1,2]

- Basic form: $\prod_{-c \le j \le c, j \neq 0} P(x_{i+j}|x_i) = \prod_{-c \le j \le c, j \neq 0} \frac{\exp(\mathbf{e}_{i+j}^{\mathrm{T}} \mathbf{e}_i)}{\sum_{l \in V} \exp(\mathbf{e}_l^{\mathrm{T}} \mathbf{e}_i)}$

where c is the context size and V is the vocabulary.



Skip-gram with Negative Sampling

$$L_{\text{neg\_sam}}(x_i) = \sum_{-c \le j \le c, j \neq 0} \Big[-\log \sigma(\mathbf{e}_{i+j}^{\mathrm{T}} \mathbf{e}_i) - \sum_{u \in M_j} \log(1 - \sigma(\mathbf{e}_u^{\mathrm{T}} \mathbf{e}_i)) \Big]$$

Positive example

negative examples

where $\sigma(z) = \frac{1}{1+\exp(-z)}$

Evaluation with Word Analogy [1]

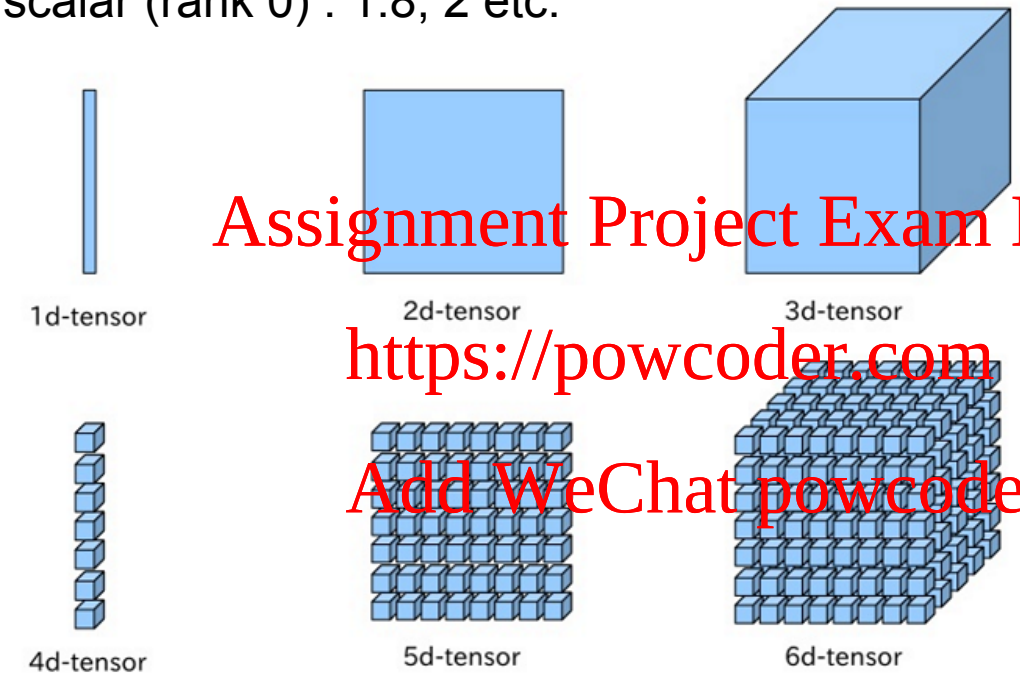
$$\mathbf{e}_? = \mathbf{e}_{\text{Athens}} - \mathbf{e}_{\text{Greece}} + \mathbf{e}_{\text{Norway}}$$

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

demo: http://bionlp-www.utu.fi/wv_demo/

TensorFlow

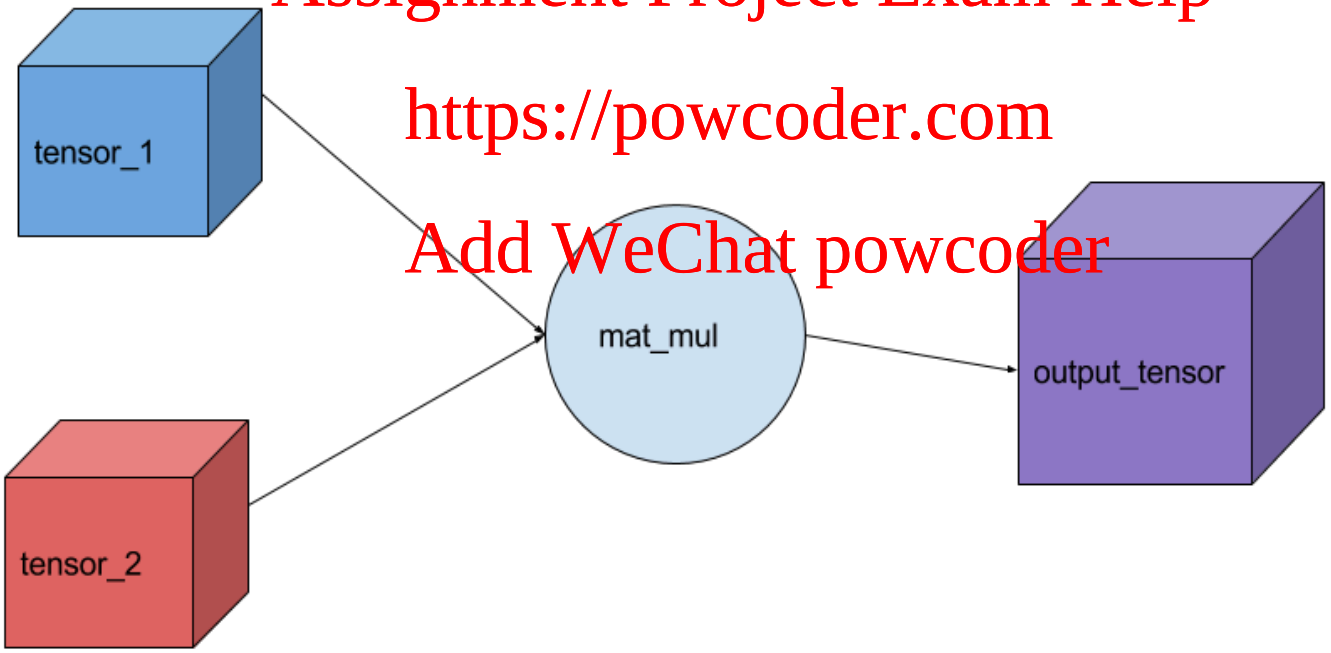
- Tensors.
  - scalar (rank 0) : 1.8, 2 etc.
  -



- Define functions on tensors.
- Auto-differentiation.

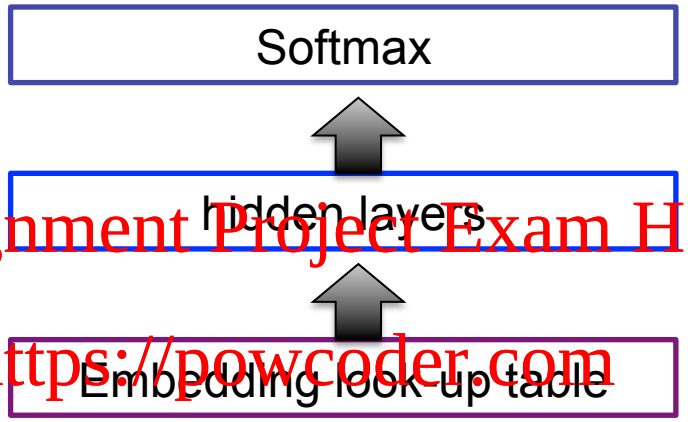
Computation Graph

- Nodes
  - tensors.
  - tensor operations.

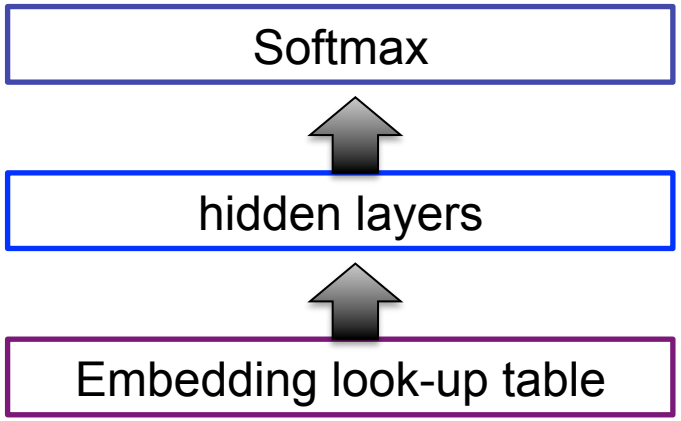


Applications

- Neural Language Model [4].



- Sentiment analysis [5, 6].



REFERENCES

- [1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.
- [2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.
- [3] https://code.google.com/archive/p/word2vec/
- [4] Yoshua Bengio, Rejean Ducharme, Pascal Vincent and Christian Jauvin. A Neural Probabilistic Language Model. JMLR, 2003.
- [5] Yoon Kim. Convolutional Neural Networks for Sentence Classification.
- [6] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank.
- [7] Understanding Word Vectors. https://medium.com/explorations-in-language-and-learning/understanding-word-vectors-f5f9e9fdef98