CS 291A: Deep Learning for NLP

Neural Networks: NN Tips and Seq2Seq

William Wang

UCSB Computer Science

william@cs.ucsb.edu

Slides adapted from V. Chen.

HW2: Fake News Detection Competition

The handout of HW2 was out on Piazza.

The deadline is in about two weeks (Due Monday 02/26, 23:59pm PT):

- You will be given a real-world dataset of short statements.
- **The task is to predict the truthfulness label of each statement**.
- You may design or use any machine learning or NLP algorithms or toolkits.
- However, you are not allowed to use additional training data.
- Use test data for training, or touching / searching / labeling test data is a violation of honor code and academic integrity.
- If you are not sure, ask questions on Piazza.

Agenda Today

1. Additional tips for designing DNNs.
2. Neural Machine Translation
   - sequence-to-sequence models.

Additional tips for designing DNNs

Data Preprocessing

Activation Function

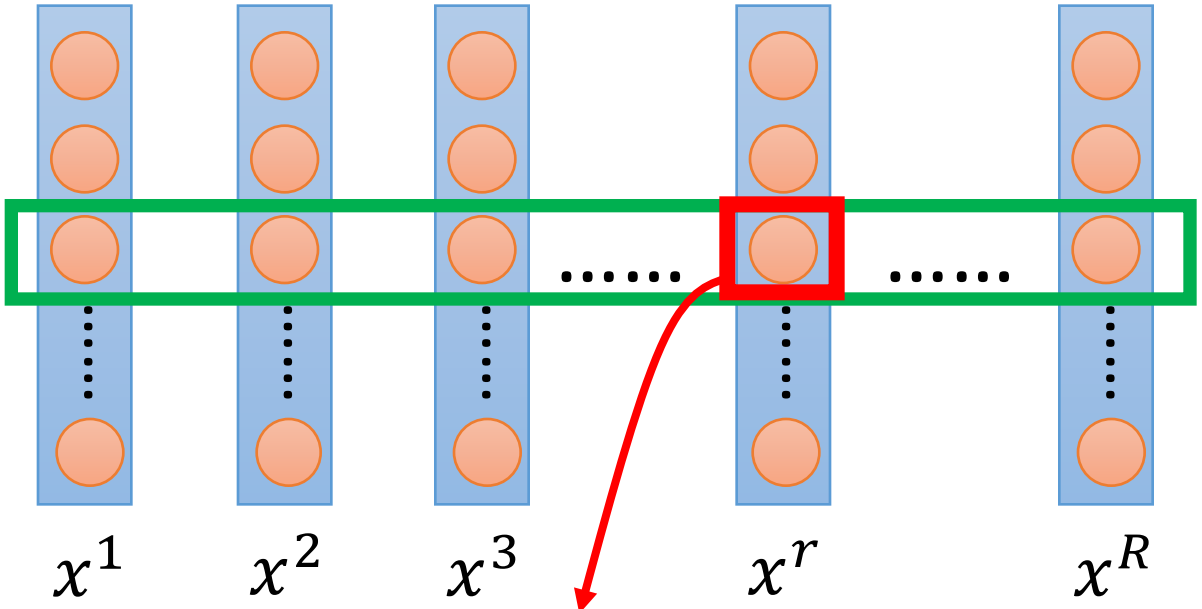
Loss Function

Optimization

Generalization

- Early Stopping
- Regularization
- Dropout

If you are not using embeddings as input… Standardization is common.



For each dimension i:

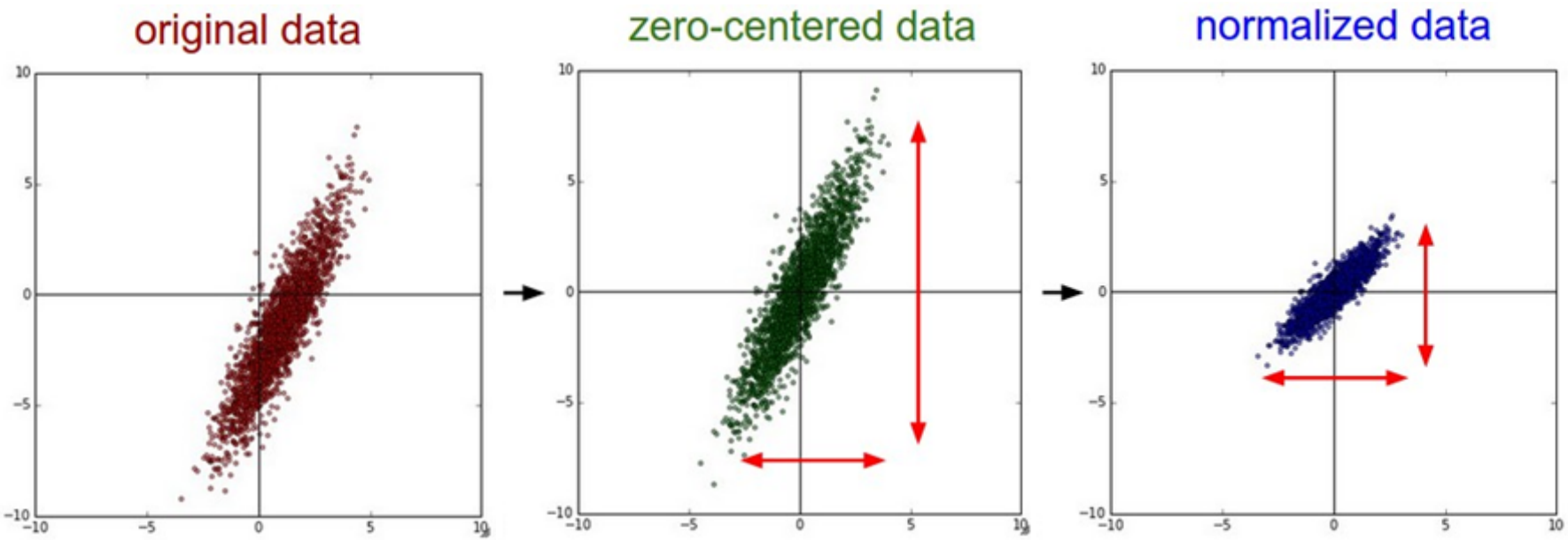
mean: m_i

standard deviation: σ_i

$$x_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

The means of all dimensions are 0, and the variances are all 1

Input Normalization



Normalizing training and testing data in the same way

Source of figure: http://cs231n.github.io/neural-networks-2/

Outline

Data Preprocessing

Activation Function

Loss Function

Optimization

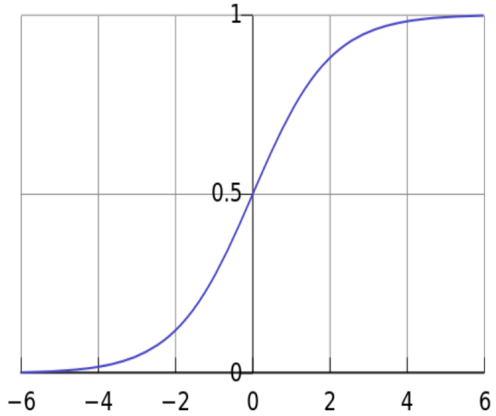
- Adagrad
- Momentum

Generalization

- Early Stopping
- Regularization
- Dropout

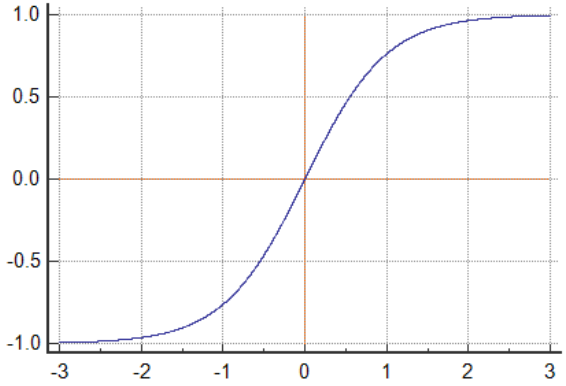
Activation Function

Sigmoid $f(x) = \dfrac{1}{1+e^{-x}}$



$$f'(x) = f(x)(1 - f(x))$$

Tanh $f(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$



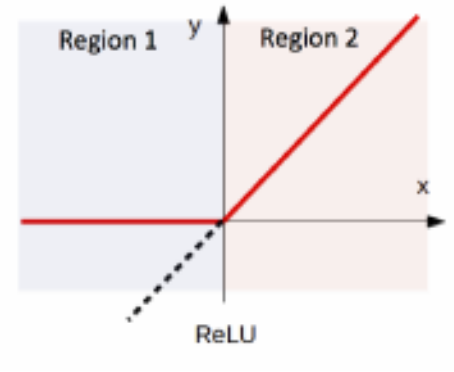
$$f'(x) = 1 - f(x)^2$$

tanh is just a rescaled and shifted sigmoid, but better for many models

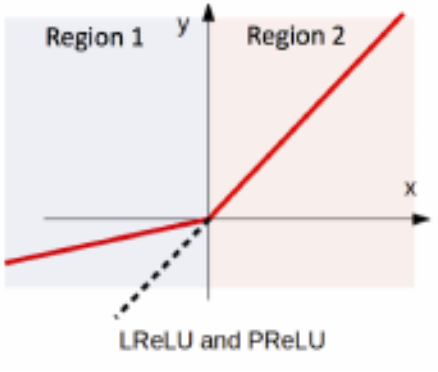
- Initialization: values close to 0
- Convergence: faster in practice
- Nice derivative (similar to sigmoid)

Variants

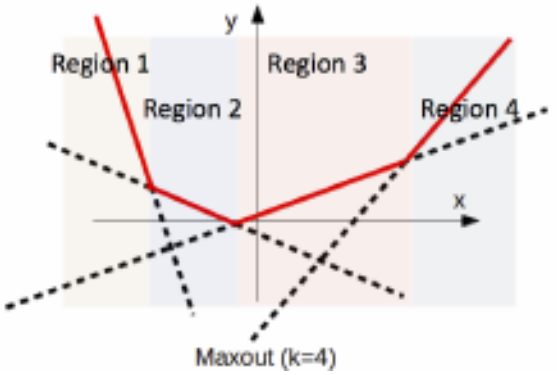
ReLU



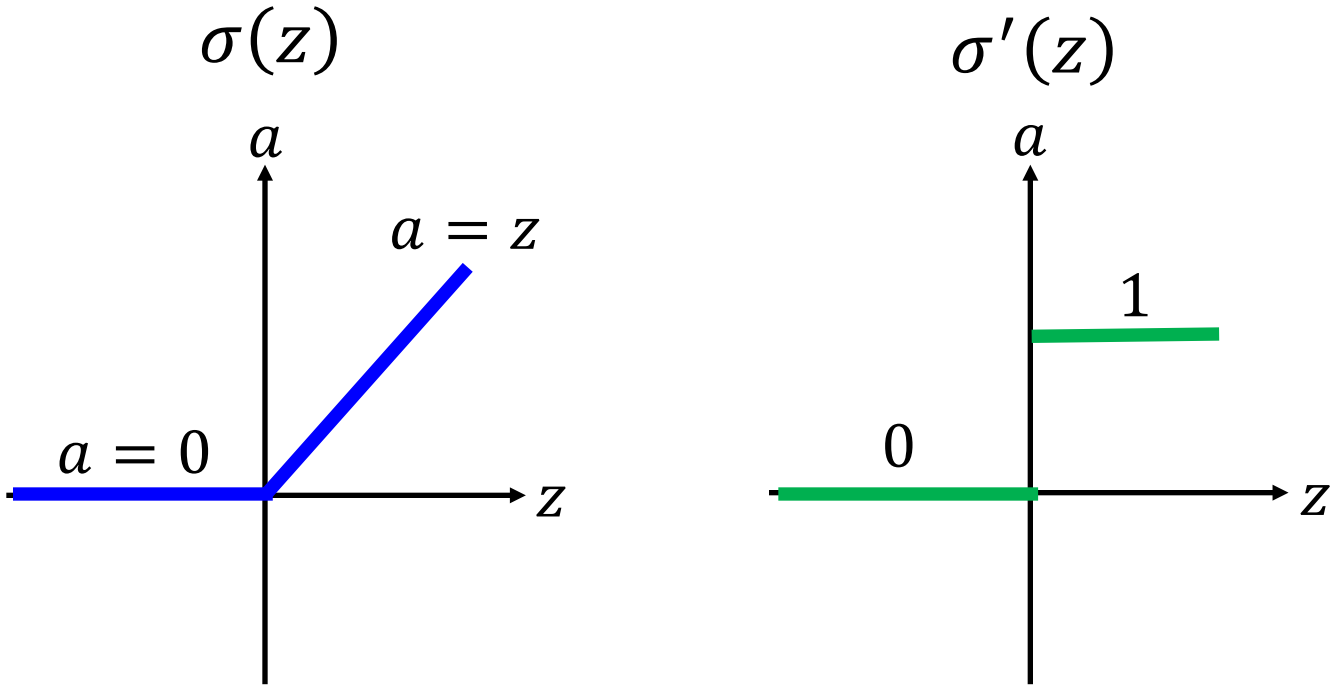
LReLU & PReLU



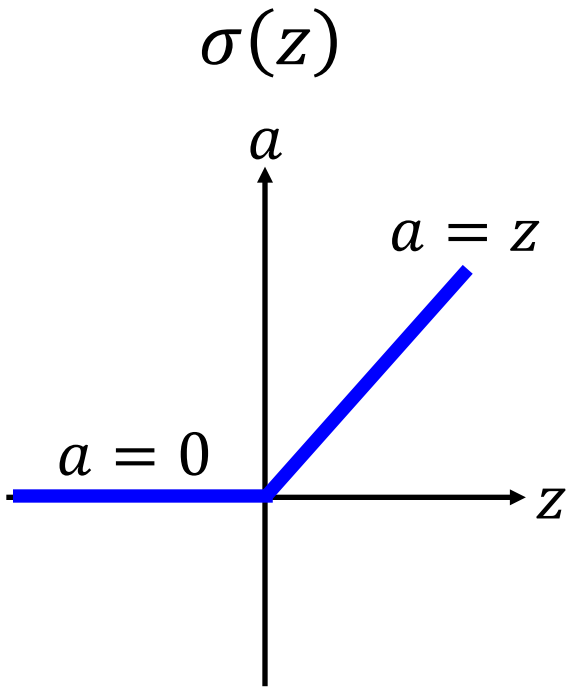
Maxout



Rectified Linear Unit (ReLU)



Rectified Linear Unit (ReLU)



Reason

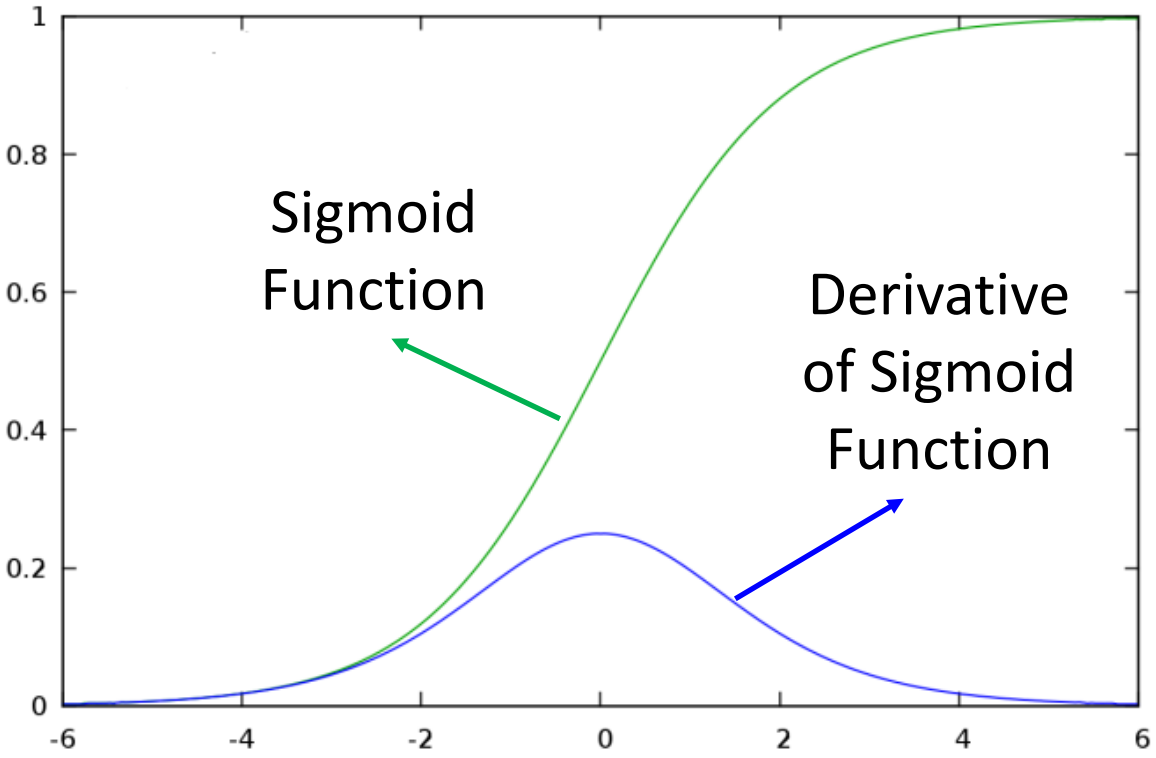
1. Fast to compute
2. Biological reason
3. Solution for vanishing/exploding gradient

Piazza Poll: Sigmoid Issue

Sigmoid activation is more likely to have which of the following issue during DNN training:

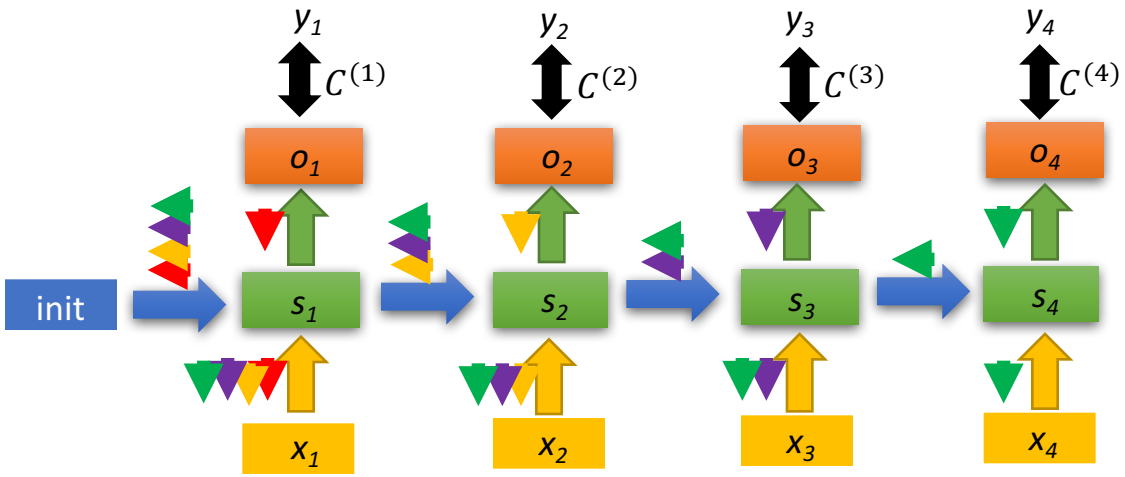
a) Vanishing gradient.
b) Exploding gradient.
c) It depends.
d) Both.

Sigmoid Issue



Derivative of the sigmoid function is always smaller than 1

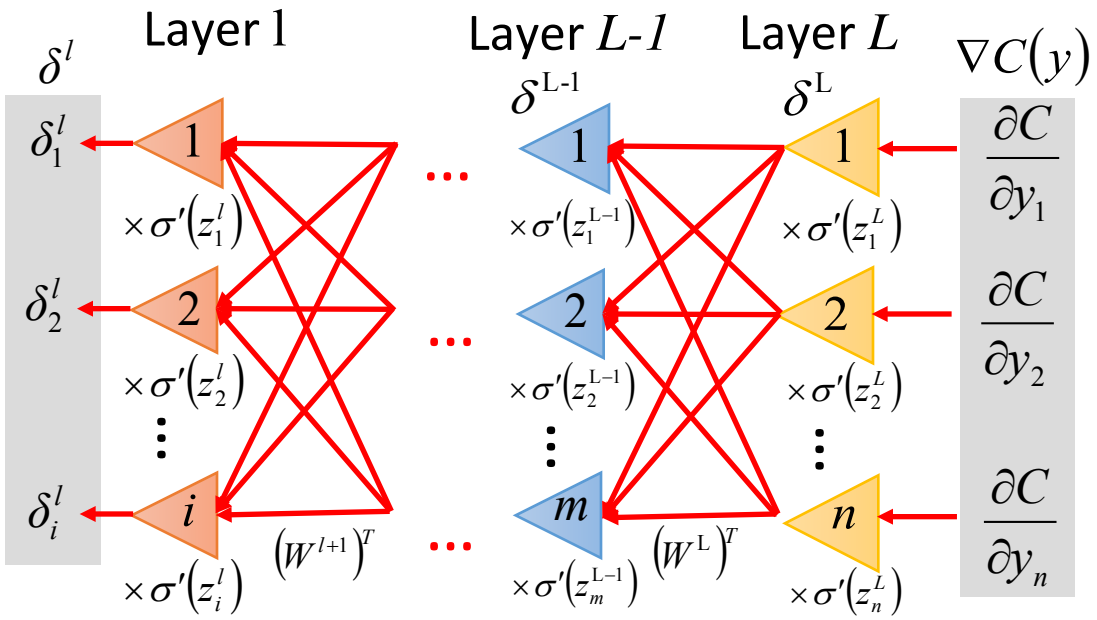
Vanishing Gradient Problem

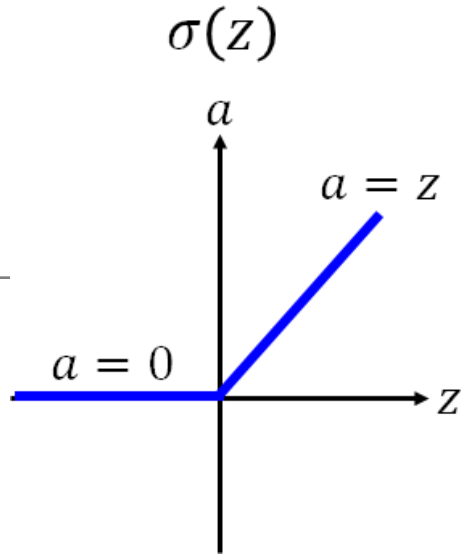


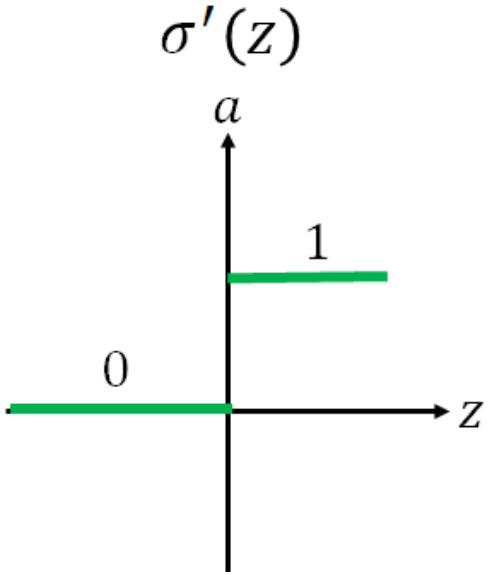
$$\delta^l = \sigma'(z^l) \odot (W^{l+1})^T \delta^{l+1}$$

The error signal is getting smaller and smaller due to $\sigma'(z) < 1$
→ **vanishing gradient**

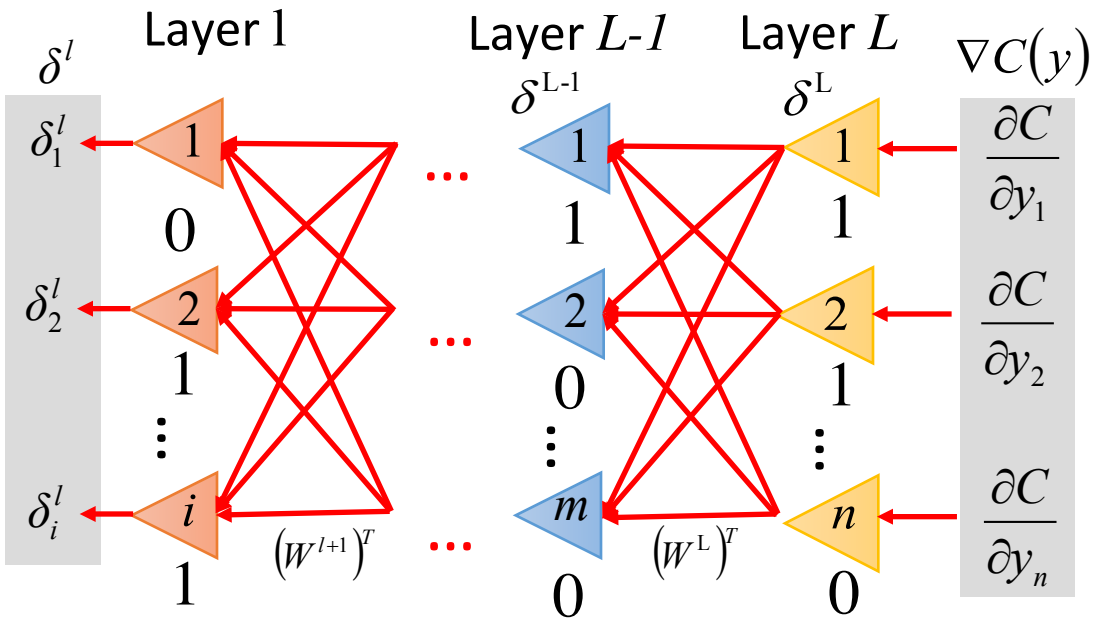
ReLU

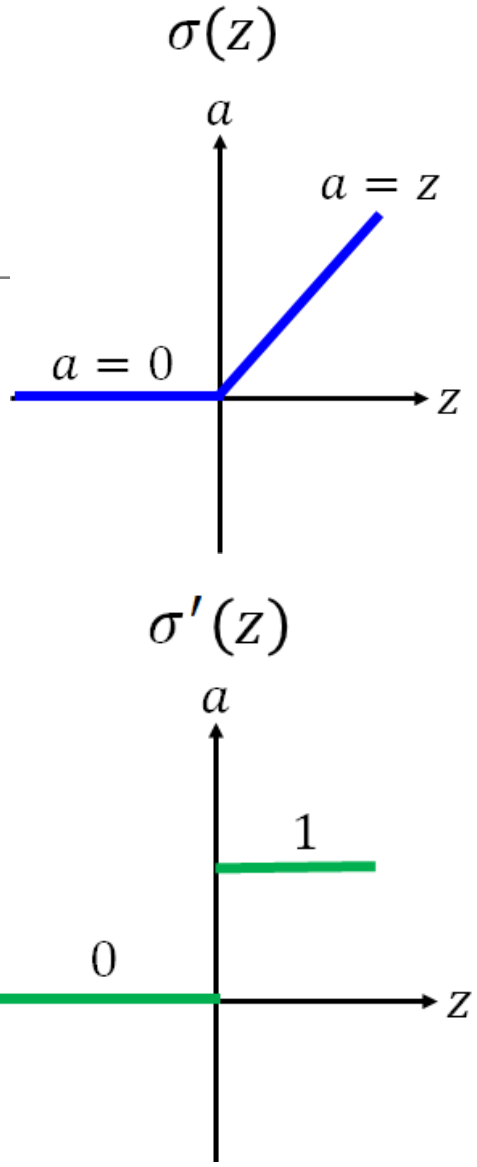

Layer 1
Layer L-1
Layer L
δ^l
δ^{L-1}
δ^L
$\nabla C(y)$
δ_1^l
δ_2^l
δ_i^l
1
2
i
m
n
$\times\sigma'(z_1^l)$
$\times\sigma'(z_2^l)$
$\times\sigma'(z_i^l)$
$\times\sigma'(z_1^{L-1})$
$\times\sigma'(z_2^{L-1})$
$\times\sigma'(z_m^{L-1})$
$\times\sigma'(z_1^L)$
$\times\sigma'(z_2^L)$
$\times\sigma'(z_n^L)$
$(W^{l+1})^T$
$(W^L)^T$
$\frac{\partial C}{\partial y_1}$
$\frac{\partial C}{\partial y_2}$
$\frac{\partial C}{\partial y_n}$


$\sigma(z)$
a
$a = z$
$a = 0$
z

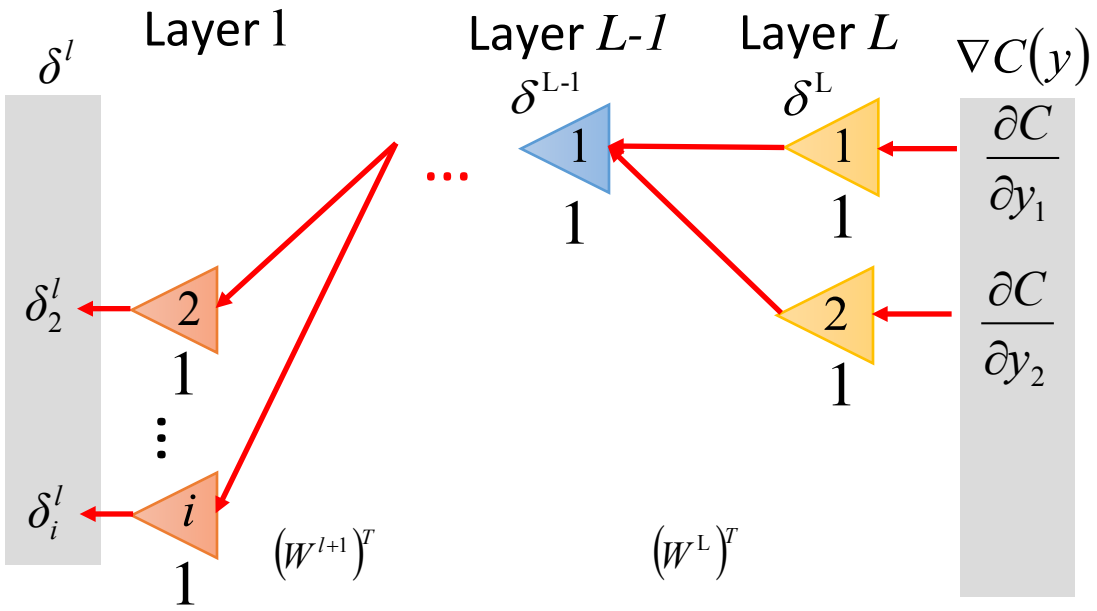

$\sigma'(z)$
a
1
0
z

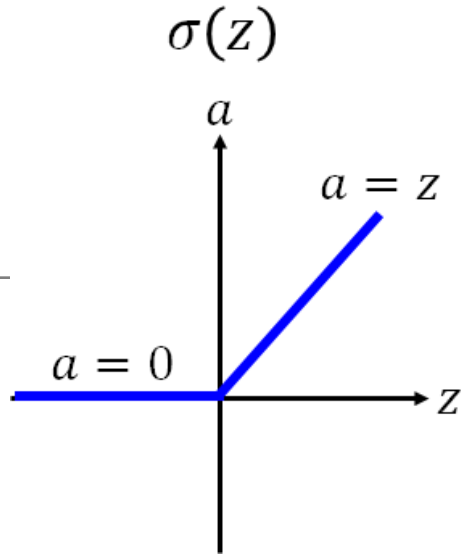
ReLU

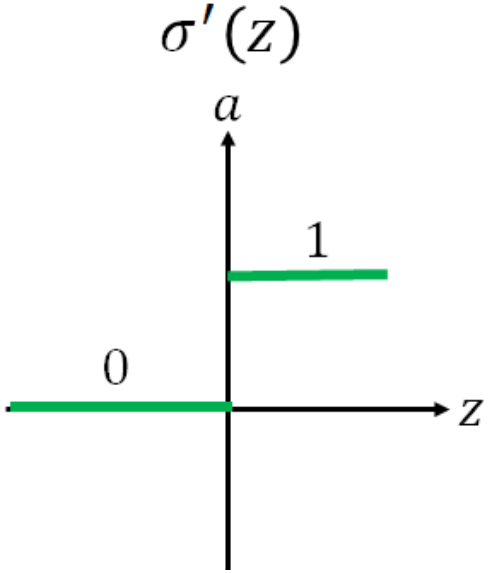

Layer l
Layer L-1
Layer L
δ^l
δ^{L-1}
δ^L
$\nabla C(y)$
δ_1^l
δ_2^l
δ_i^l
$\frac{\partial C}{\partial y_1}$
$\frac{\partial C}{\partial y_2}$
$\frac{\partial C}{\partial y_n}$
$(W^{l+1})^T$
$(W^L)^T$


$\sigma(z)$
a
$a = z$
$a = 0$
z
$\sigma'(z)$
a
1
0
z

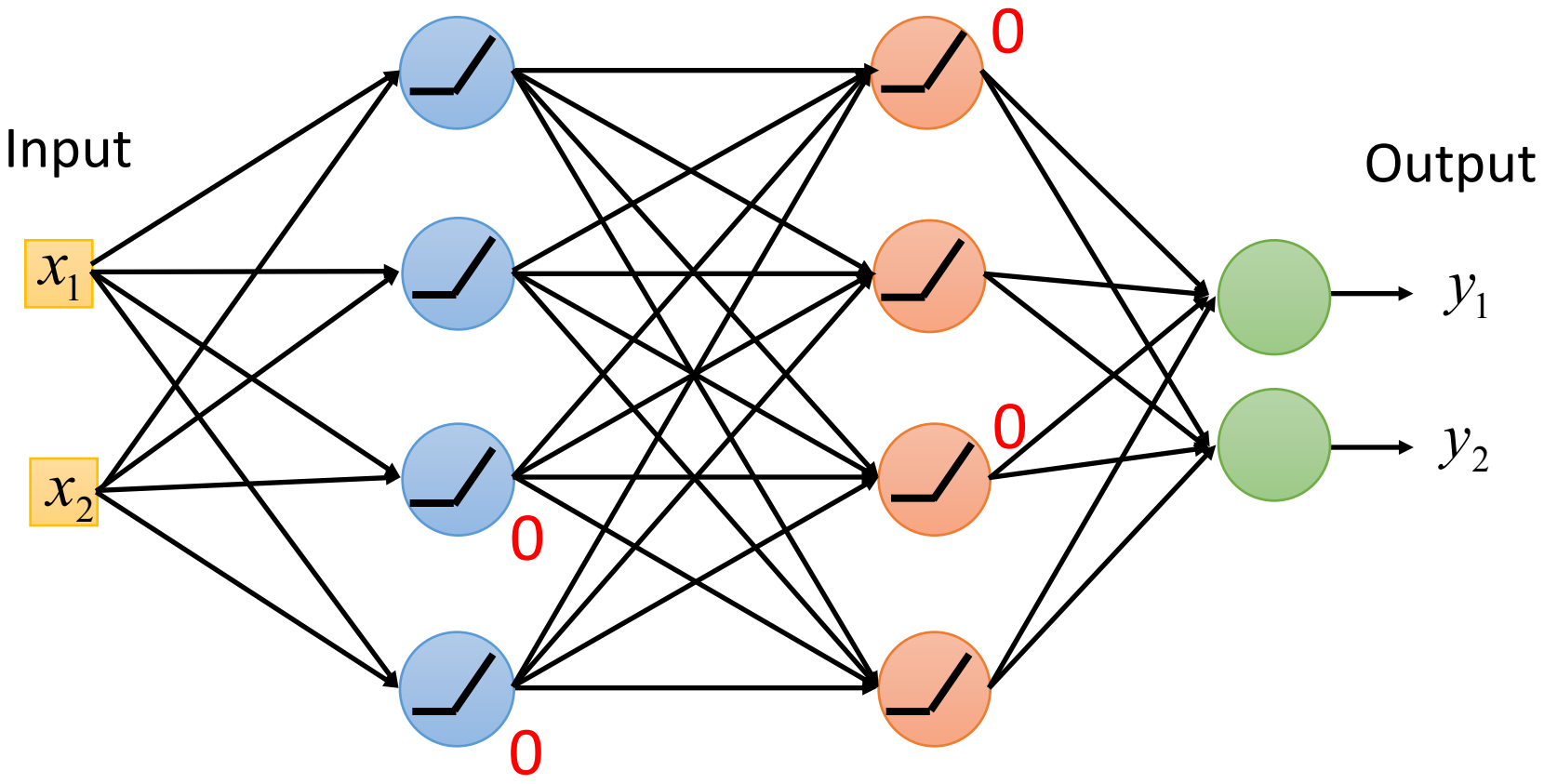
ReLU


Layer 1
Layer L-1
Layer L
δ^l
δ^{L-1}
δ^L
$\nabla C(y)$
δ_2^l
δ_i^l
$\frac{\partial C}{\partial y_1}$
$\frac{\partial C}{\partial y_2}$
$(W^{l+1})^T$
$(W^L)^T$

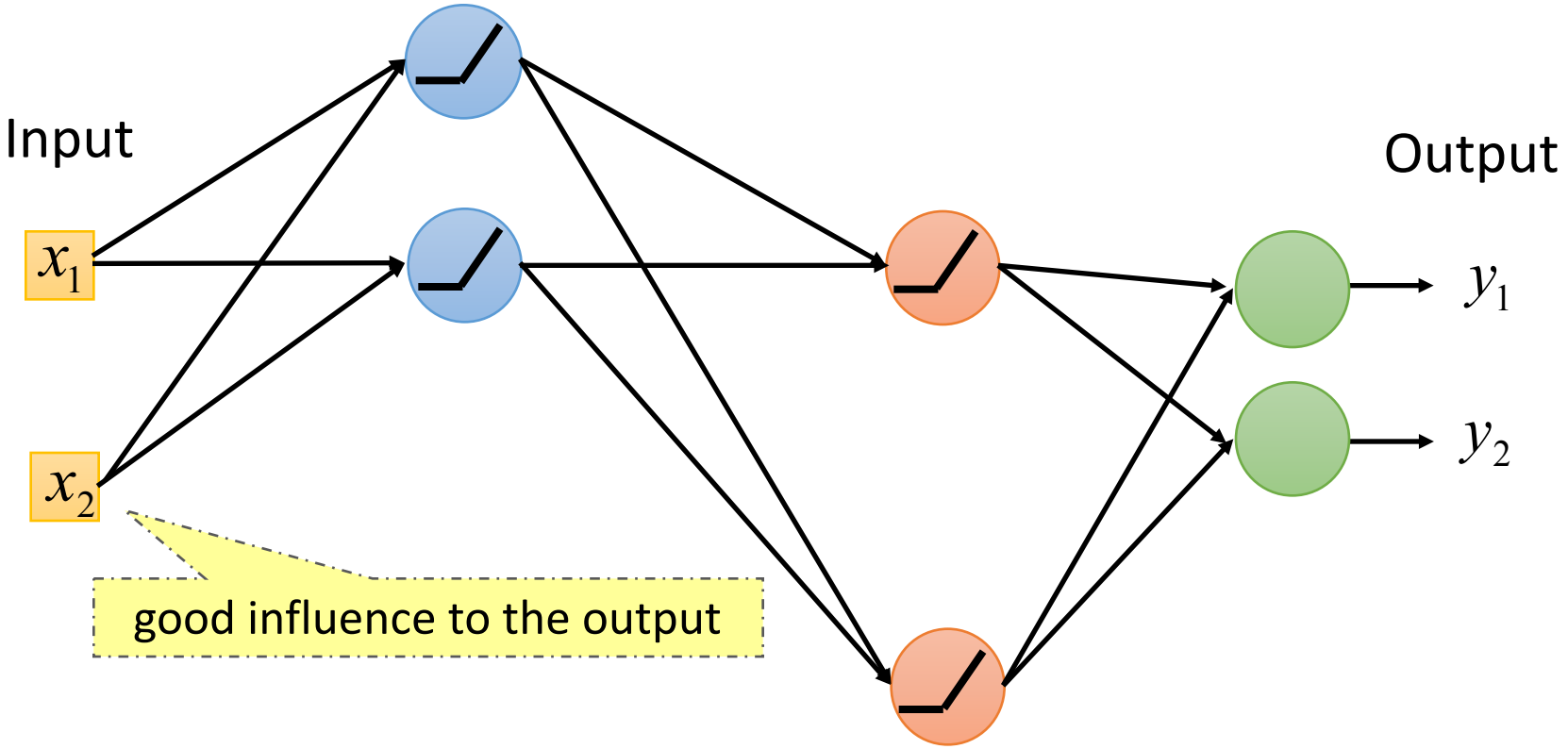

$\sigma(z)$
a
$a = z$
$a = 0$
z


$\sigma'(z)$
a
1
0
z

ReLU – Forward Pass

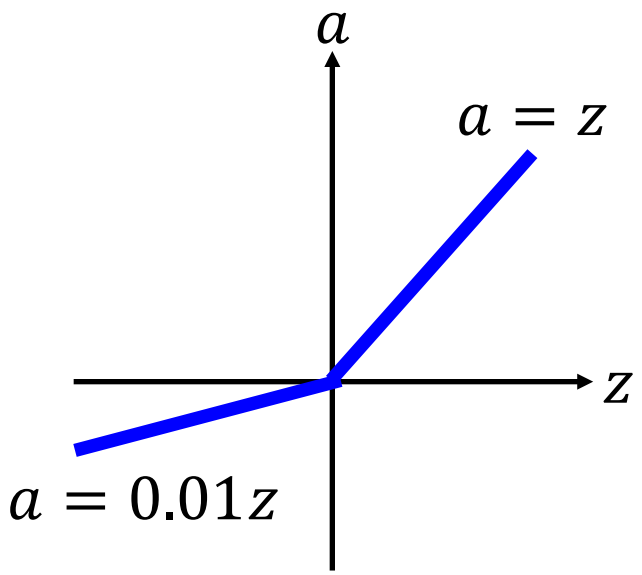


ReLU – Backward Pass

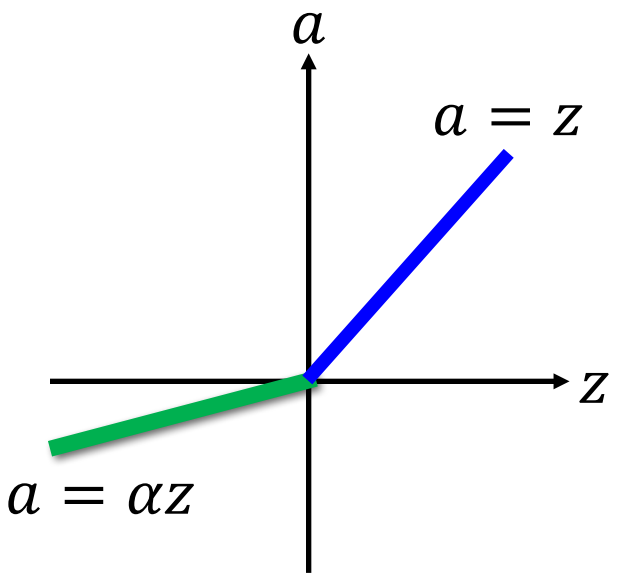


Variant ReLU

Leaky ReLU

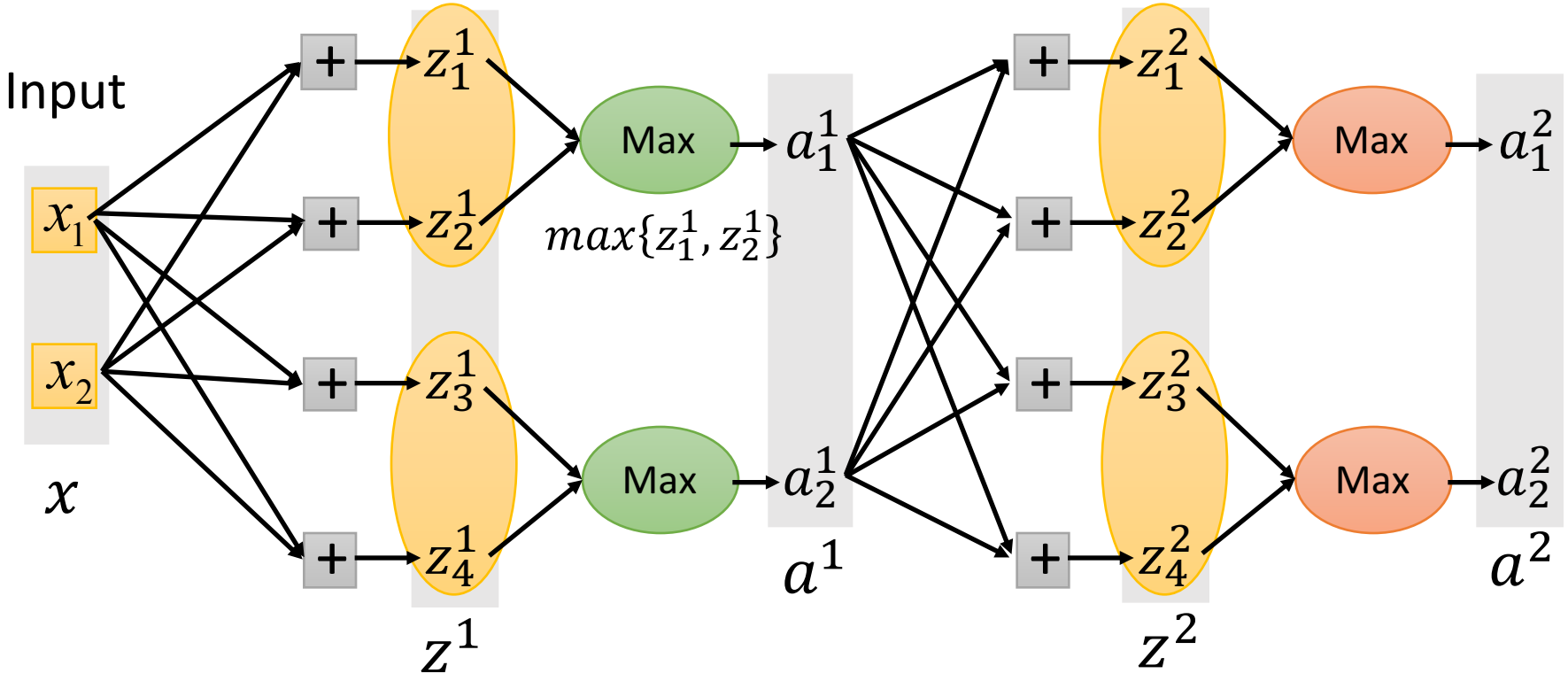


Parametric ReLU

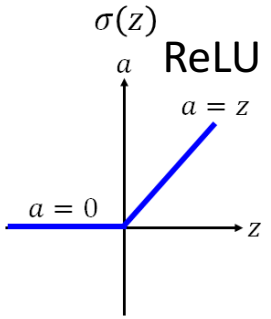


α is also learned by gradient descent

Maxout


Input
x_1
x_2
x
+
z_1^1
z_2^1
z_3^1
z_4^1
z^1
Max
$max\{z_1^1, z_2^1\}$
a_1^1
a_2^1
a^1
z_1^2
z_2^2
z_3^2
z_4^2
z^2
a_1^2
a_2^2
a^2
$z^1 = W^1 x$
$z^2 = W^2 a^1$

Piazza Poll: what's the relationship between ReLU and Maxout?



Input

x_1

x_2

x

z_1^1 z_2^1 z_3^1 z_4^1

z^1

Max

Max

$max\{z_1^1, z_2^1\}$

a_1^1 a_2^1

a^1

z_1^2 z_2^2 z_3^2 z_4^2

z^2

Max

Max

a_1^2 a_2^2

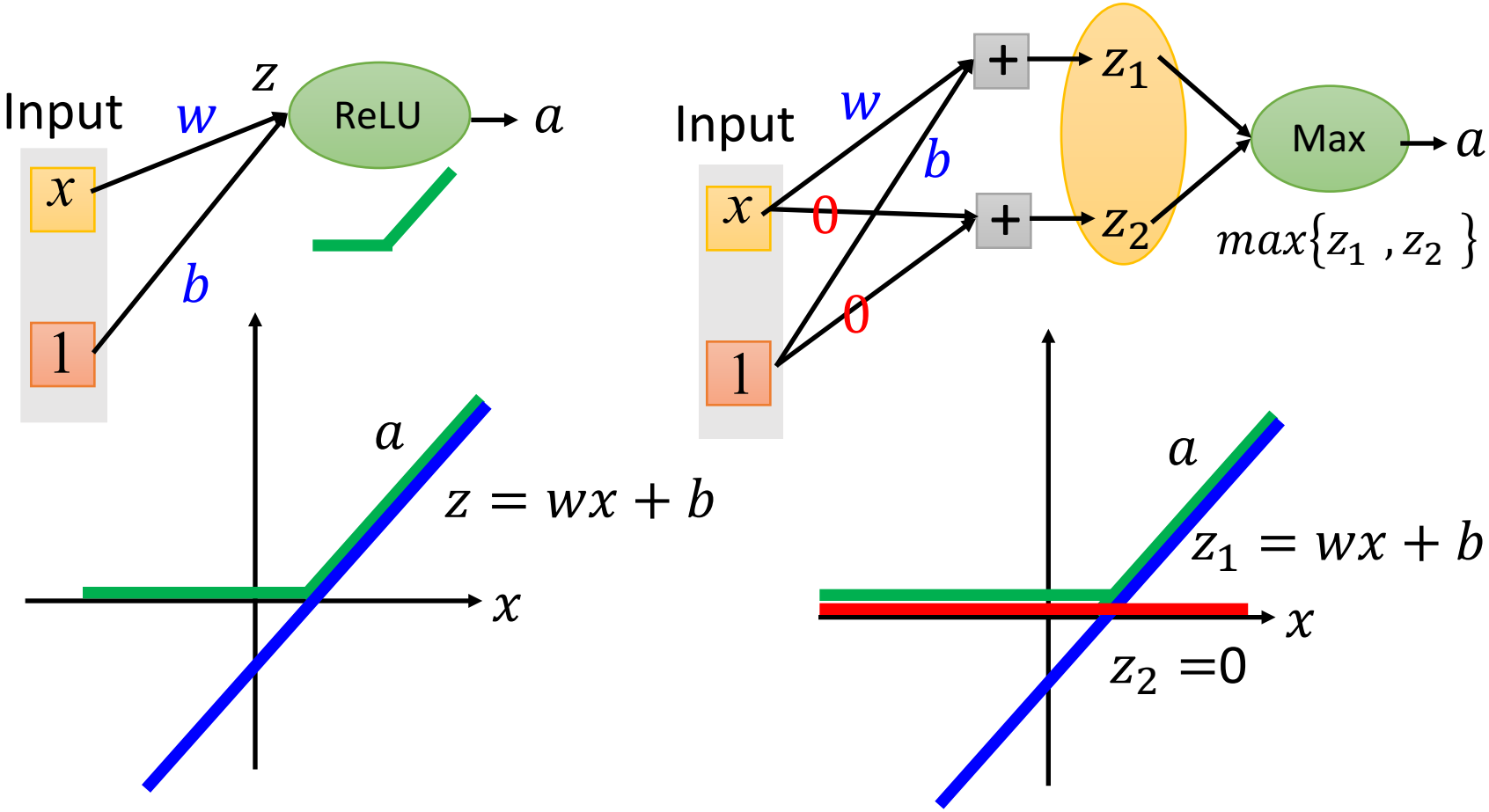
a^2

$z^1 = W^1 x$

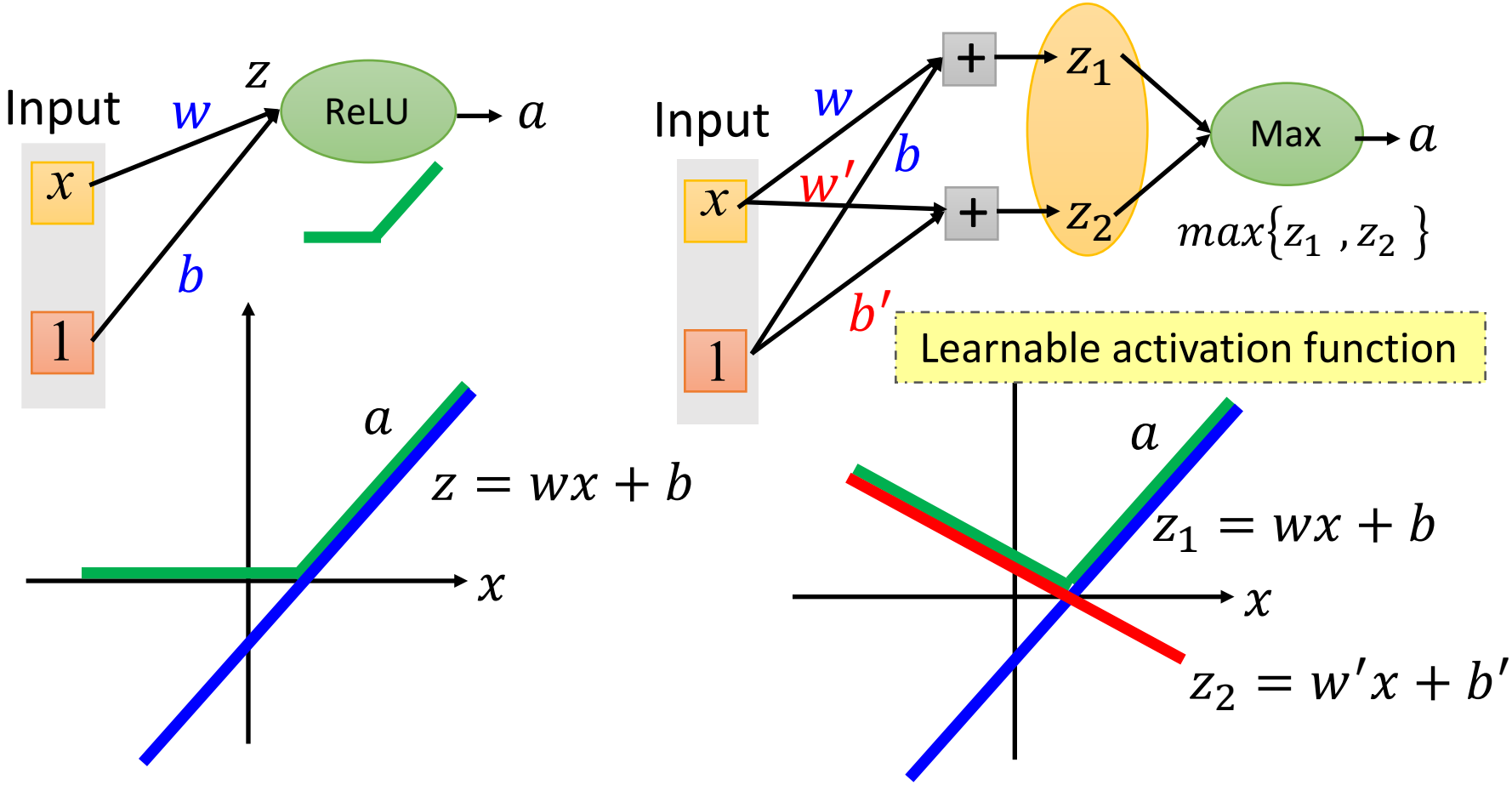
$z^2 = W^2 a^1$

Maxout

ReLU is a special case of Maxout

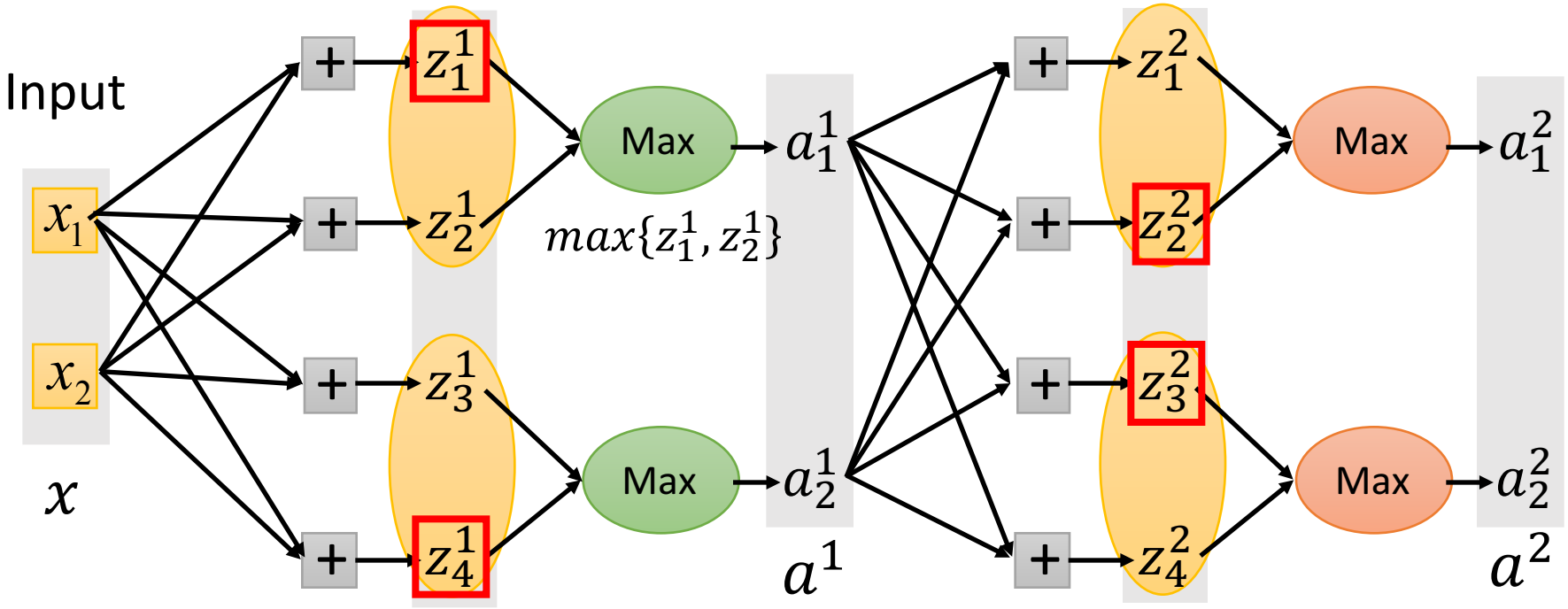


Maxout – ReLU is a special case



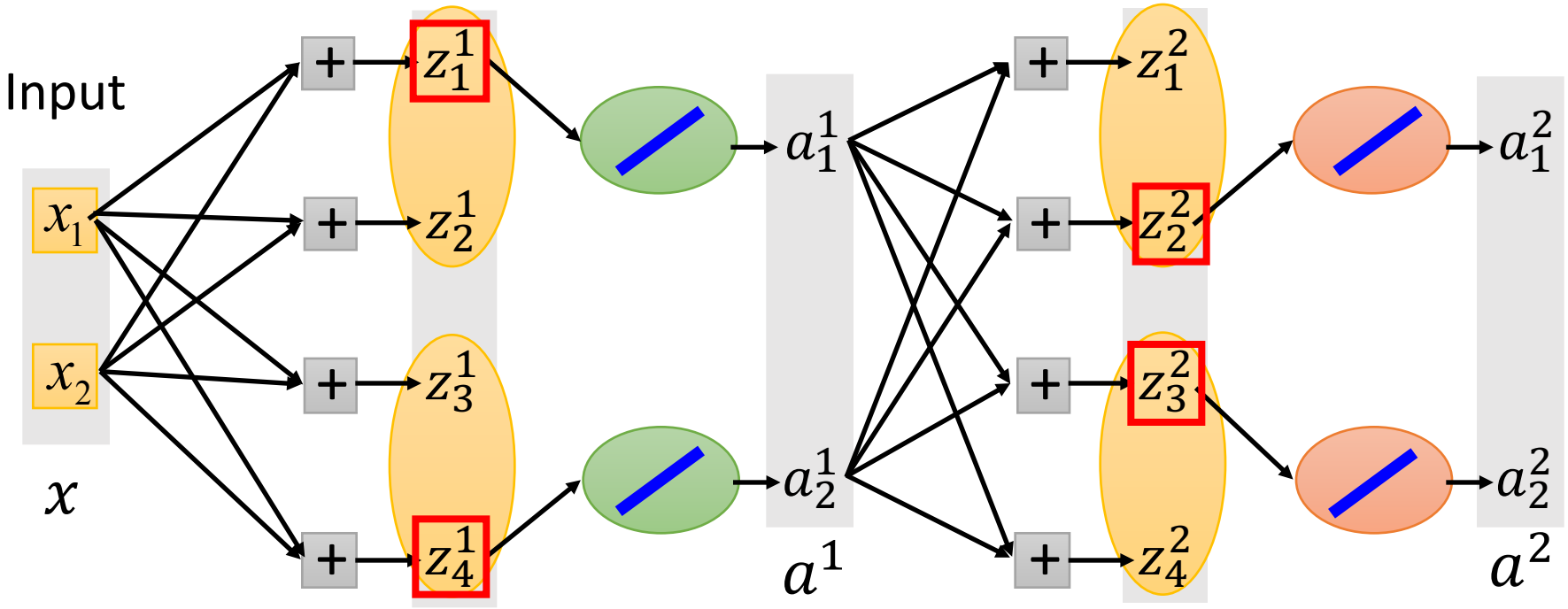
Maxout - Training

Given training data x, we decide z for maxout



Maxout - Training

Given training data x, we decide z for maxout



Outline

Data Preprocessing

Activation Function

Loss Function

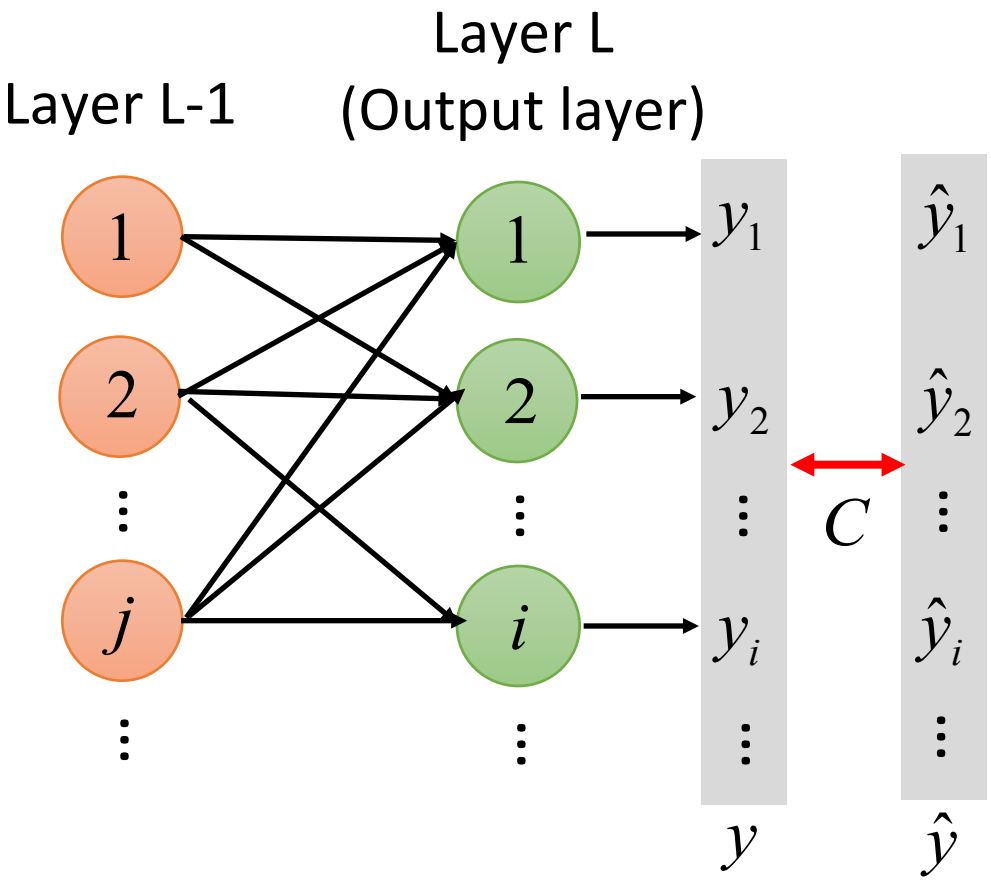
Optimization

- Adagrad
- Momentum

Generalization

- Early Stopping
- Regularization
- Dropout

Loss Function – Square Error



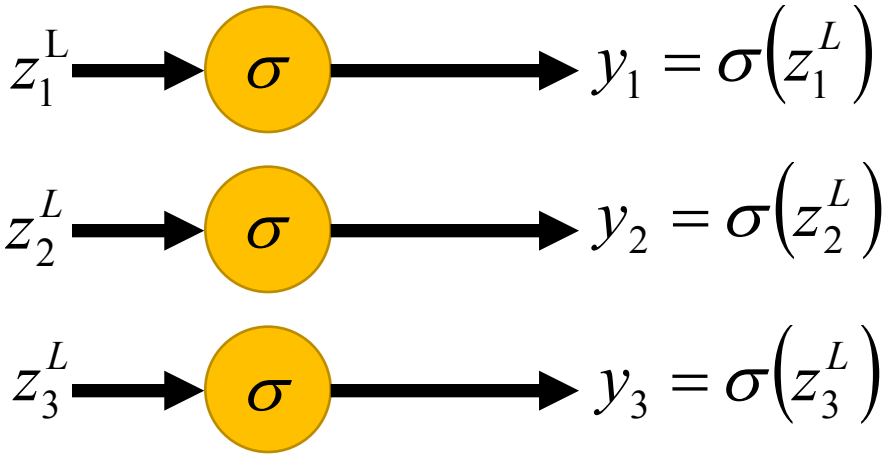
$$C = \frac{1}{2}\left\|y - \hat{y}\right\|^2$$

$$= \frac{1}{2}\sum_n \left(y_n - \hat{y}_n\right)^2$$

Softmax

Softmax layer as the output layer

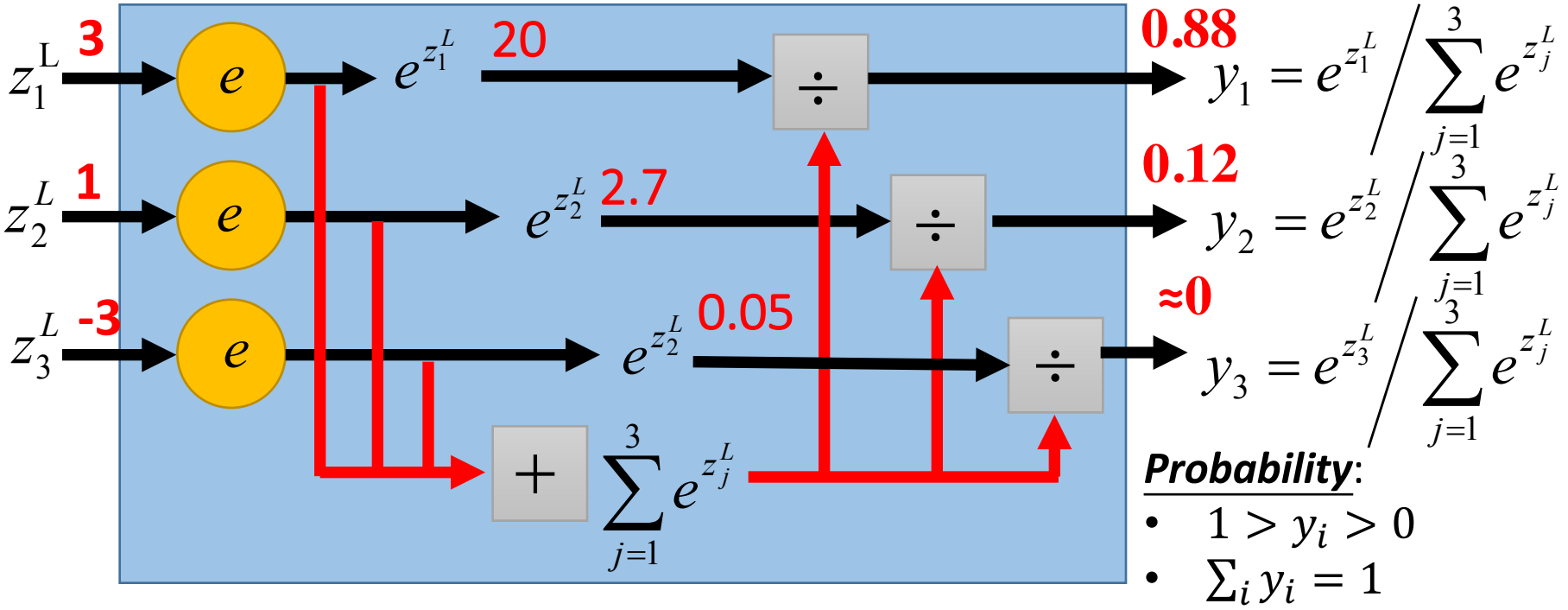
Ordinary Output layer



Softmax

Softmax layer as the output layer

Softmax Layer



Probability:

- $1 > y_i > 0$
- $\sum_i y_i = 1$

Training labels indicate positive and negative samples in stead of the actual values

Outline

Data Preprocessing

Activation Function

Loss Function

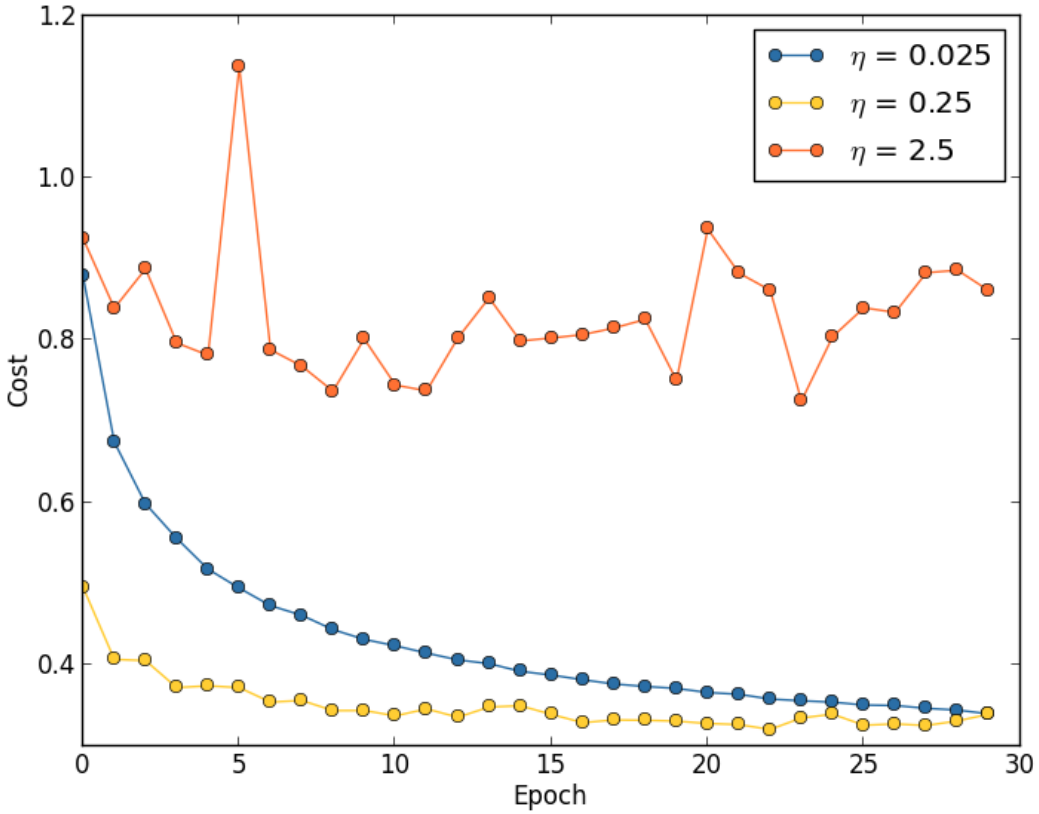
Optimization

- Adagrad
- Momentum

Generalization

- Early Stopping
- Regularization
- Dropout

Learning Rate



The proper learning rate is important to find the optimal point

Source: http://neuralnetworksanddeeplearning.com/chap3.html

Learning Rate

Idea: reduce the learning rate every few epochs

- At the beginning, we are far from the destination, so we use a larger learning rate
- After several epochs, we are close to the destination, so we reduce the learning rate

Manually set learning rate

1) Reduce by 0.5 when validation error stops improving
2) 1/t decay: $\eta^t = \eta/\sqrt{t+1}$ due to theoretical convergence guarantees

Learning rate cannot be one-size-fits-all
→ different parameters have different learning rates

Outline

Data Preprocessing

Activation Function

Loss Function

Optimization

- Adagrad
- Momentum

Generalization

- Early Stopping
- Regularization
- Dropout

Generalization

Practical tricks

1) find the right network structure and implement and optimize it properly
2) prevent overfitting
   - Reduce the model size by lowering the number of units and layers/hyperparameters
   - Early stopping
   - Standard L1 or L2 regularization
   - Sparsity constraint

Outline

Data Preprocessing

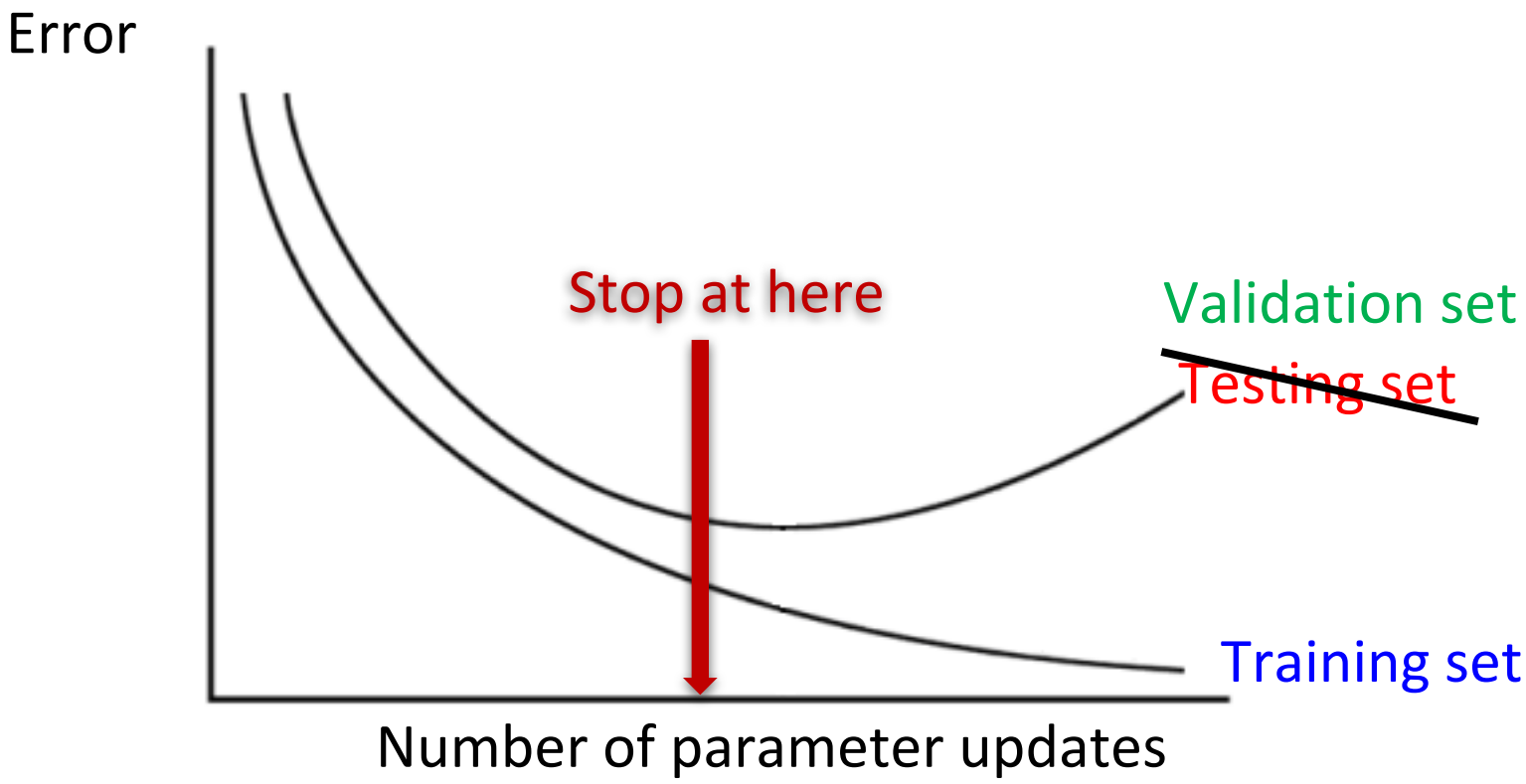
Activation Function

Loss Function

Optimization
- Adagrad
- Momentum

Generalization
- **Early Stopping**
- Regularization
- Dropout

Early Stopping



Check performance on validation set to prevent training too many iterations

Outline

Data Preprocessing

Activation Function

Loss Function

Optimization

- Adagrad
- Momentum

Generalization

- Early Stopping
- **Regularization**
- Dropout

Regularization

Idea: the parameters closer to zero are preferred

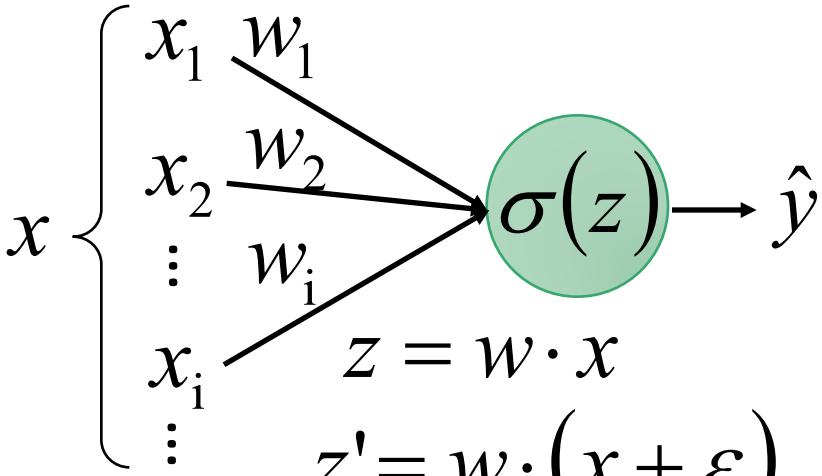
Training data:

$$\{(x, \hat{y}), \ldots\}$$

Testing data:

$$\{(x', \hat{y}), \ldots\}$$

$$x' = x + \varepsilon$$



$$z = w \cdot x$$

$$\begin{aligned} z' &= w \cdot (x + \varepsilon) \\ &= w \cdot x + w \cdot \varepsilon \\ &= z + w \cdot \varepsilon \end{aligned}$$

To minimize the effect of noise, we want w close to zero.

Regularization

Idea: optimize a new cost function to find a set of weight that 1) minimizes original cost and 2) is close to zero

$$C'(\theta) = C(\theta) + \lambda \frac{1}{2} \|\theta\|^2$$

→ regularization term

$$\theta = \{W^1, W^2, \ldots\}$$

original cost

(e.g. minimize square error, cross entropy …)

Regularization

Idea: optimize a new cost function to find a set of weight that 1) minimizes original cost and 2) is close to zero

$$C'(\theta) = C(\theta) + \lambda \frac{1}{2} \|\theta\|^2$$

Gradient: $\frac{\partial C'}{\partial w} = \frac{\partial C}{\partial w} + \lambda w$

Update:

$$w^{t+1} \rightarrow w^t - \eta \frac{\partial C'}{\partial w^t} = w^t - \eta \left(\frac{\partial C}{\partial w^t} + \lambda w^t \right)$$

$$= (1 - \eta\lambda) w^t - \eta \frac{\partial C}{\partial w^t}$$

Smaller and smaller

Outline

Data Preprocessing

Activation Function

Loss Function

Optimization

- Adagrad
- Momentum

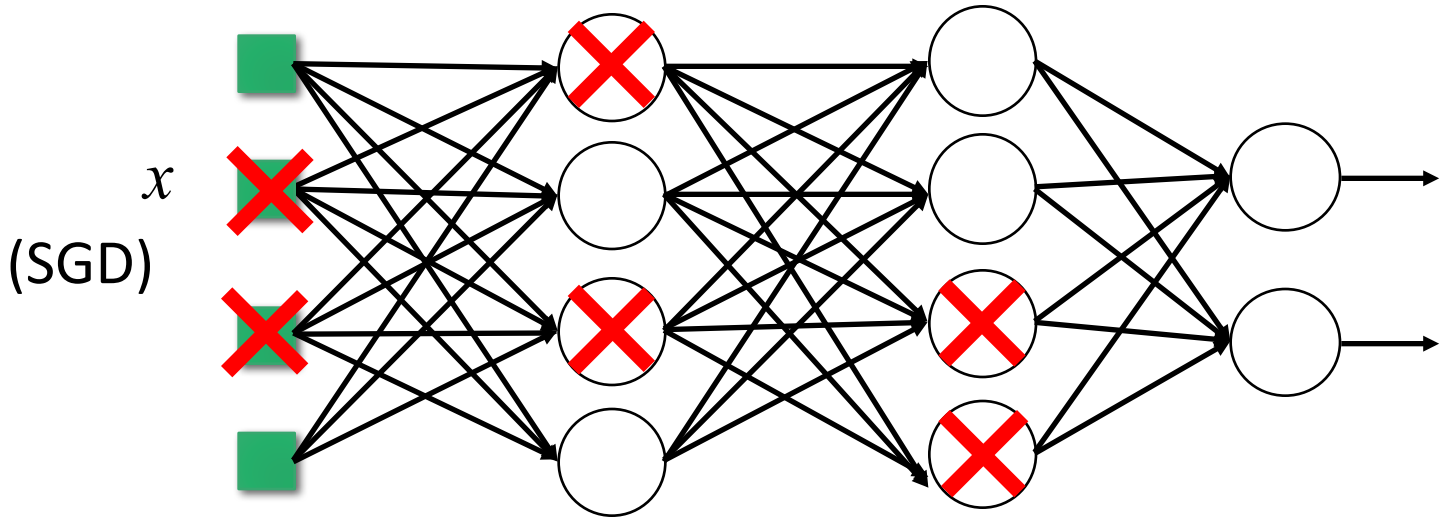
Generalization

- Early Stopping
- Regularization
- **Dropout**

Dropout

For each iteration of training,

- each neuron has $p\%$ to be removed

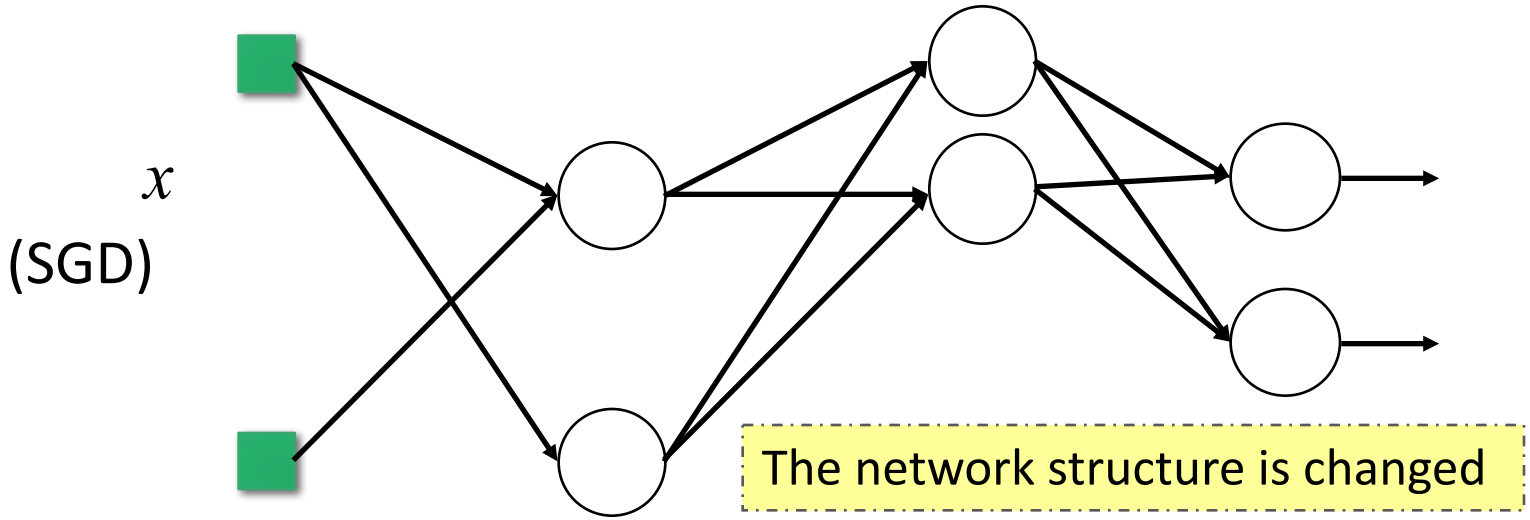


Dropout

For each iteration of training,

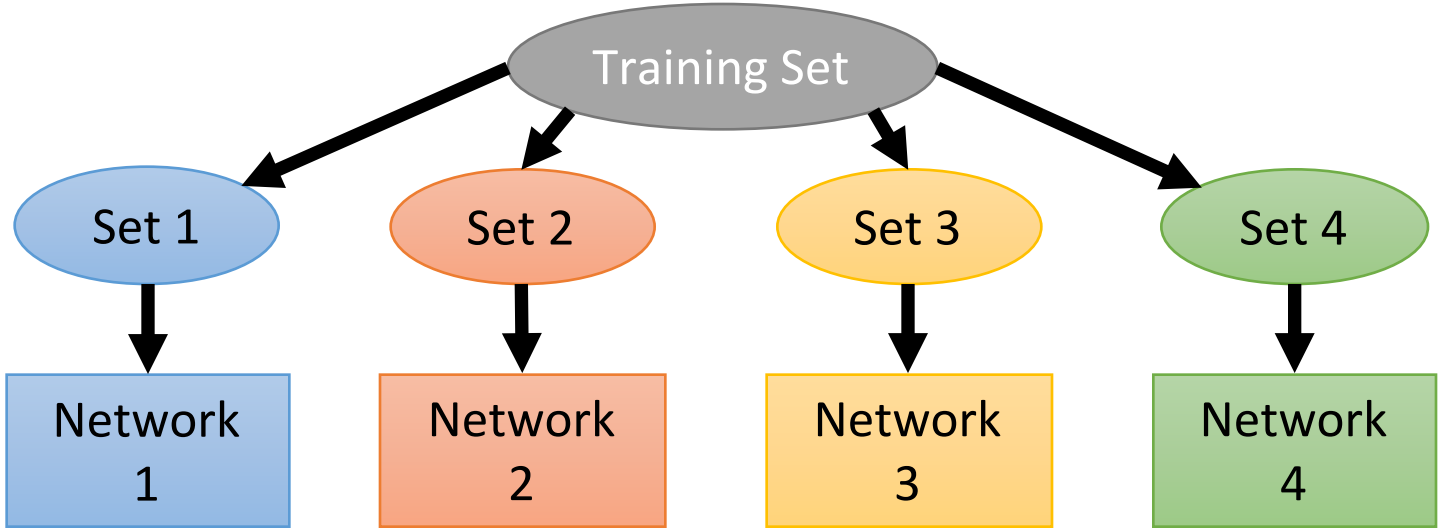
- each neuron has $p\%$ to be removed
- training using the new network

Training: $\theta^{i+1} \leftarrow \theta^{i} - \eta \nabla_{\theta} C(\theta^{i})$



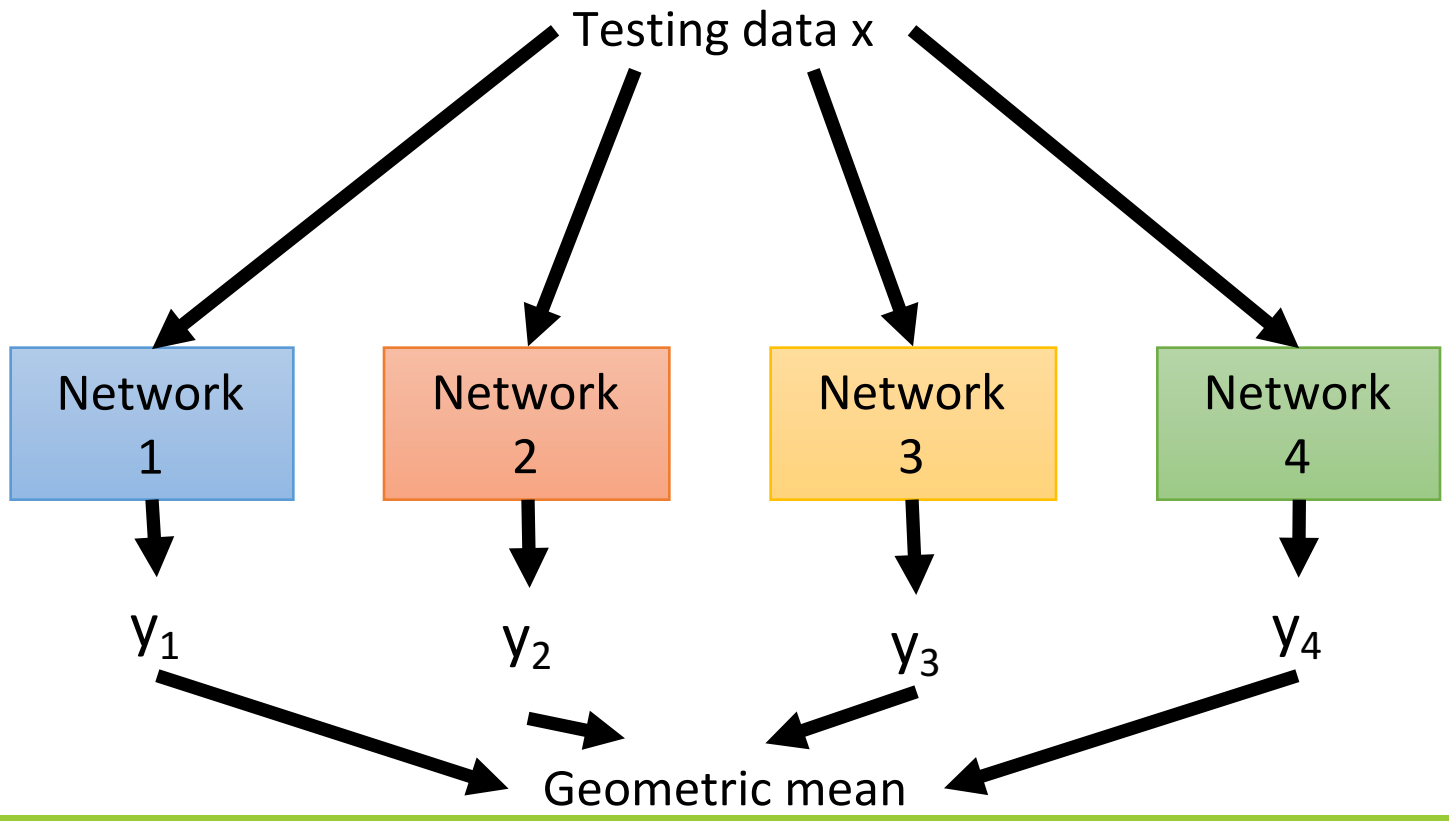
Dropout

Train a bunch of networks with different structures



Dropout – Ensemble

Ensemble



Concluding Remarks

Data Preprocessing: Input Normalization

Activation Function: ReLU, Maxout

Loss Function: Softmax

Optimization

Generalization

- Early Stopping: avoid too many iterations from overfitting
- Regularization: minimize the effect of noise
- Dropout: leverage the benefit of ensemble

Machine Translation

MT in the real world

Machine Translation

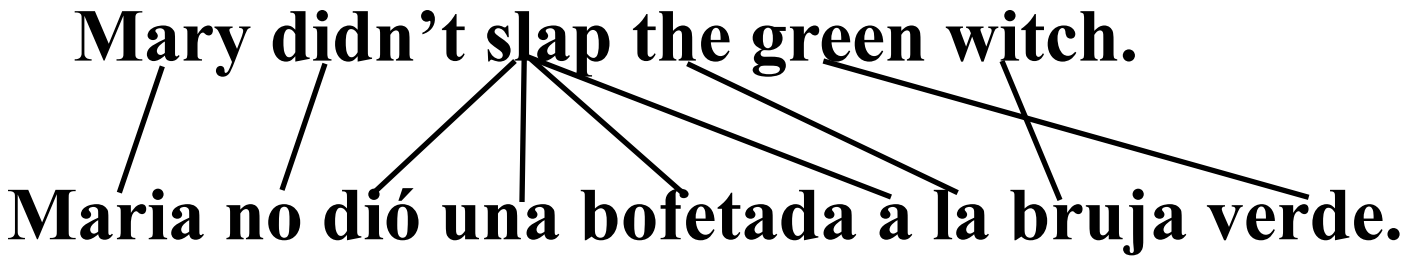
Automatically translate one natural language into another.

Mary didn’t slap the green witch.

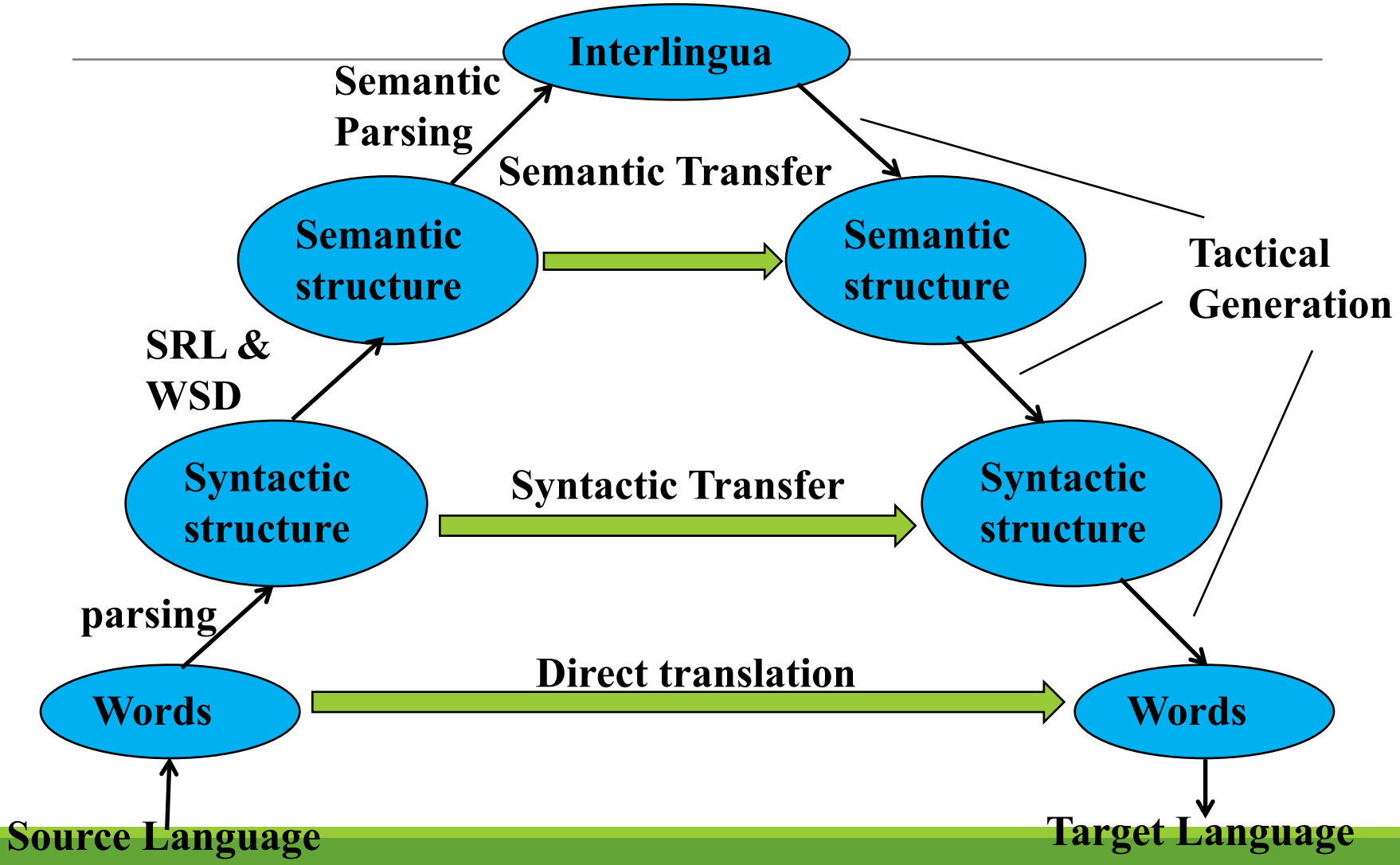
Maria no dió una bofetada a la bruja verde.

Word Alignment

Shows mapping between words in one language and the other.



MT Pyramid


Interlingua
Semantic Parsing
Semantic Transfer
Semantic structure
Semantic structure
Tactical Generation
SRL & WSD
Syntactic structure
Syntactic Transfer
Syntactic structure
parsing
Direct translation
Words
Words
Source Language
Target Language

Statistical MT

Manually encoding comprehensive bilingual lexicons and transfer rules is difficult.

SMT acquires knowledge needed for translation from a ***parallel corpus*** or ***bitext*** that contains the same set of documents in two languages.

The Canadian Hansards (parliamentary proceedings in French and English) is a well-known parallel corpus.

First align the sentences in the corpus based on simple methods that use coarse cues like sentence length to give bilingual sentence pairs.

Picking a Good Translation

A good translation should be ***faithful*** and correctly convey the information and tone of the original source sentence.

A good translation should also be ***fluent***, grammatically well structured and readable in the target language.

Final objective:

$$T_{best} = \underset{T \in \text{Target}}{\text{argmax}}\ \text{faithfulness}(T, S)\ \text{fluency}(T)$$

Noisy Channel Model

Based on analogy to information-theoretic model used to decode messages transmitted via a communication channel that adds errors.

Assume that source sentence was generated by a "noisy" transformation of some target language sentence and then use Bayesian analysis to recover the most likely target sentence that generated it.

Translate foreign language sentence $F=f_1, f_2, \ldots f_m$ to an English sentence $\hat{E} = e_1, e_2, \ldots e_l$ that maximizes $P(E \mid F)$

Bayesian Analysis of Noisy Channel

$$\hat{E} = \underset{E \in English}{\operatorname{argmax}} P(E \mid F)$$

$$= \underset{E \in English}{\operatorname{argmax}} \frac{P(F \mid E)P(E)}{P(F)}$$

$$= \underset{E \in English}{\operatorname{argmax}} P(F \mid E)P(E)$$

Translation Model **Language Model**

A decoder determines the most probable translation $\hat{E}$ given F

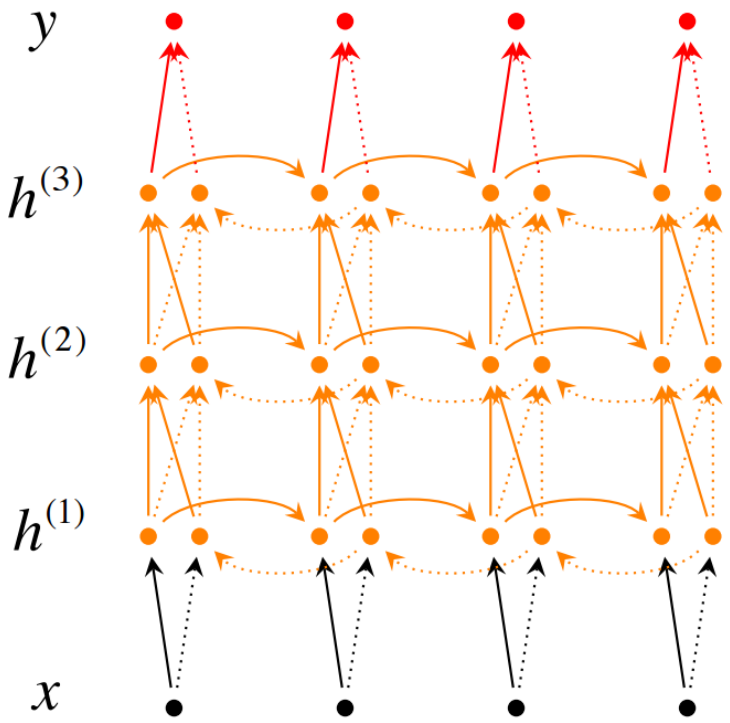
Neural Machine Translation

Let’s forget about designing features and Bayes rules.

- We are going to design a neural network for end-to-end MT.

Piazza Poll:

Can NMT use this Deep Bidirectional RNN?



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

X would be input in foreign language, and Y would be output in English

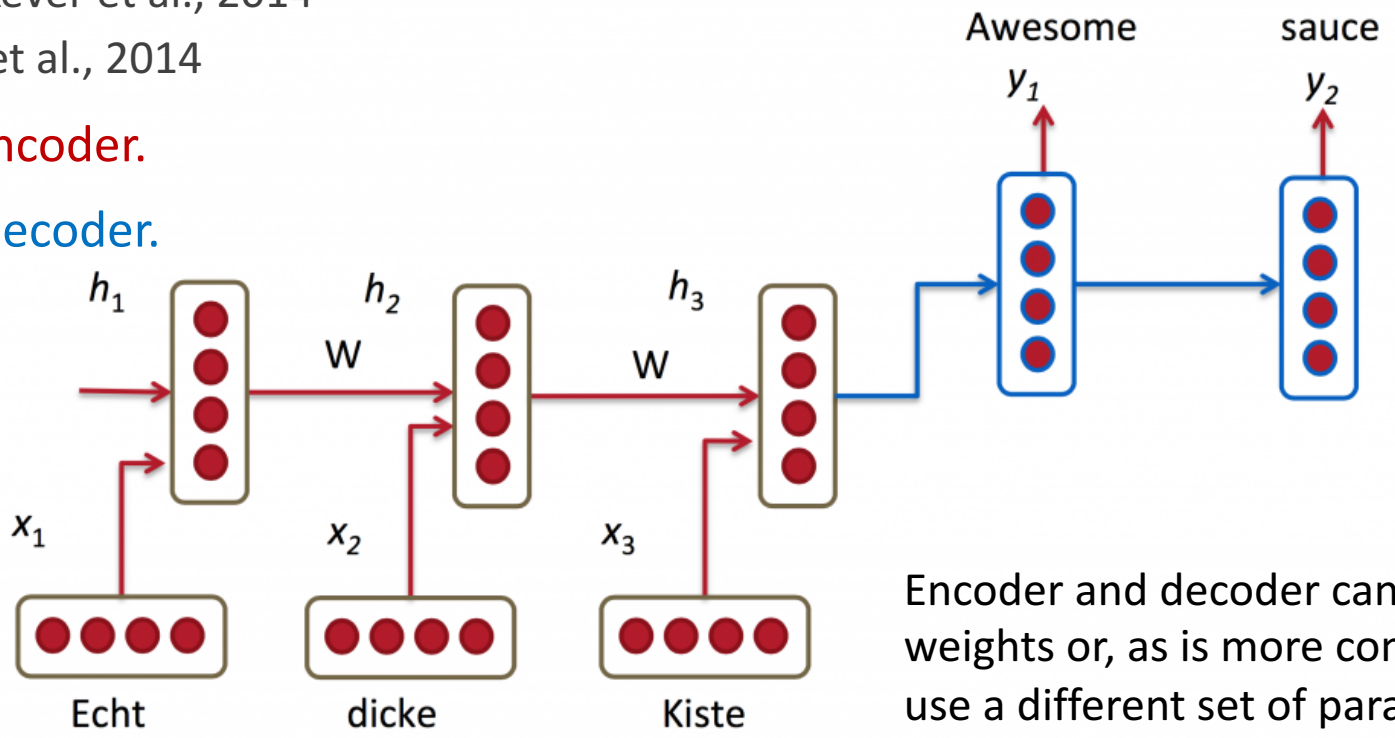
Neural Machine Translation

Sequence-to-sequence model (seq2seq)

- Sutskever et al., 2014
- Cho et al., 2014

Red: encoder.

Blue: decoder.



Encoder and decoder can share weights or, as is more common, use a different set of parameters.

Image Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

Neural Machine Translation

Predict: $$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

Training: optimize LCL.

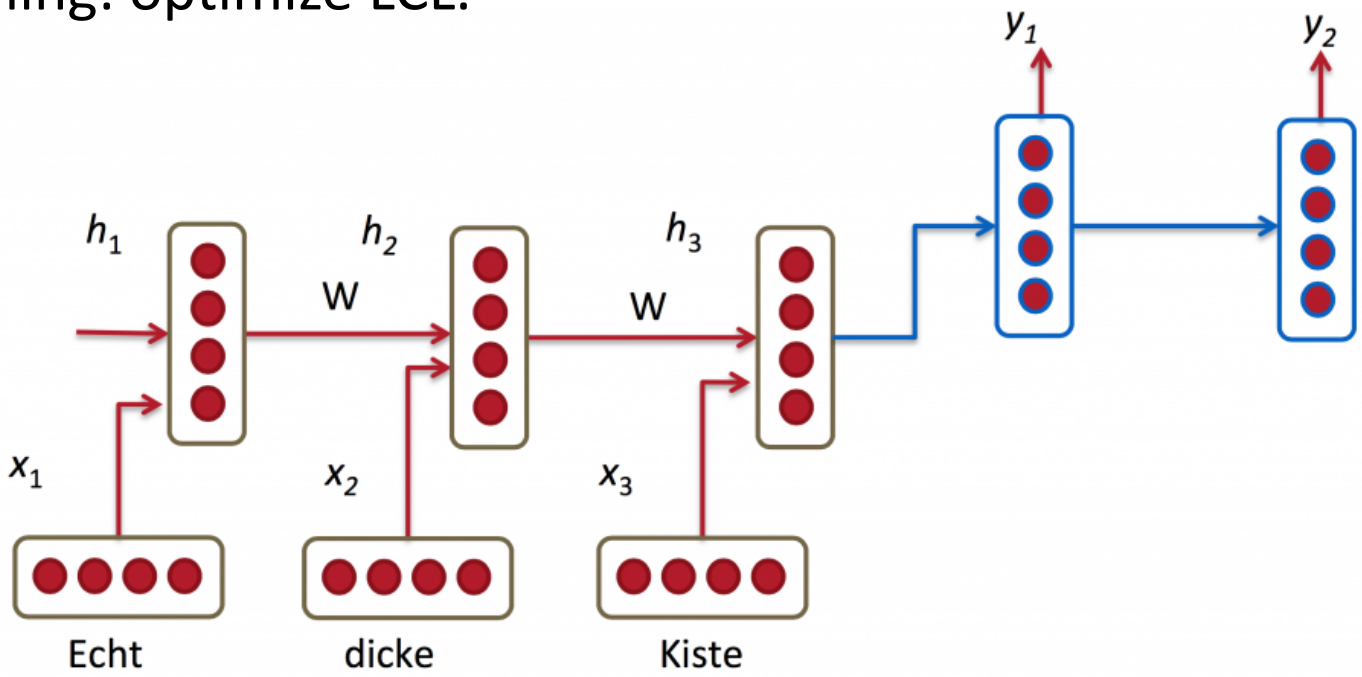


Image Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

Group discussion:

Is there an issue with this plain seq2seq model?

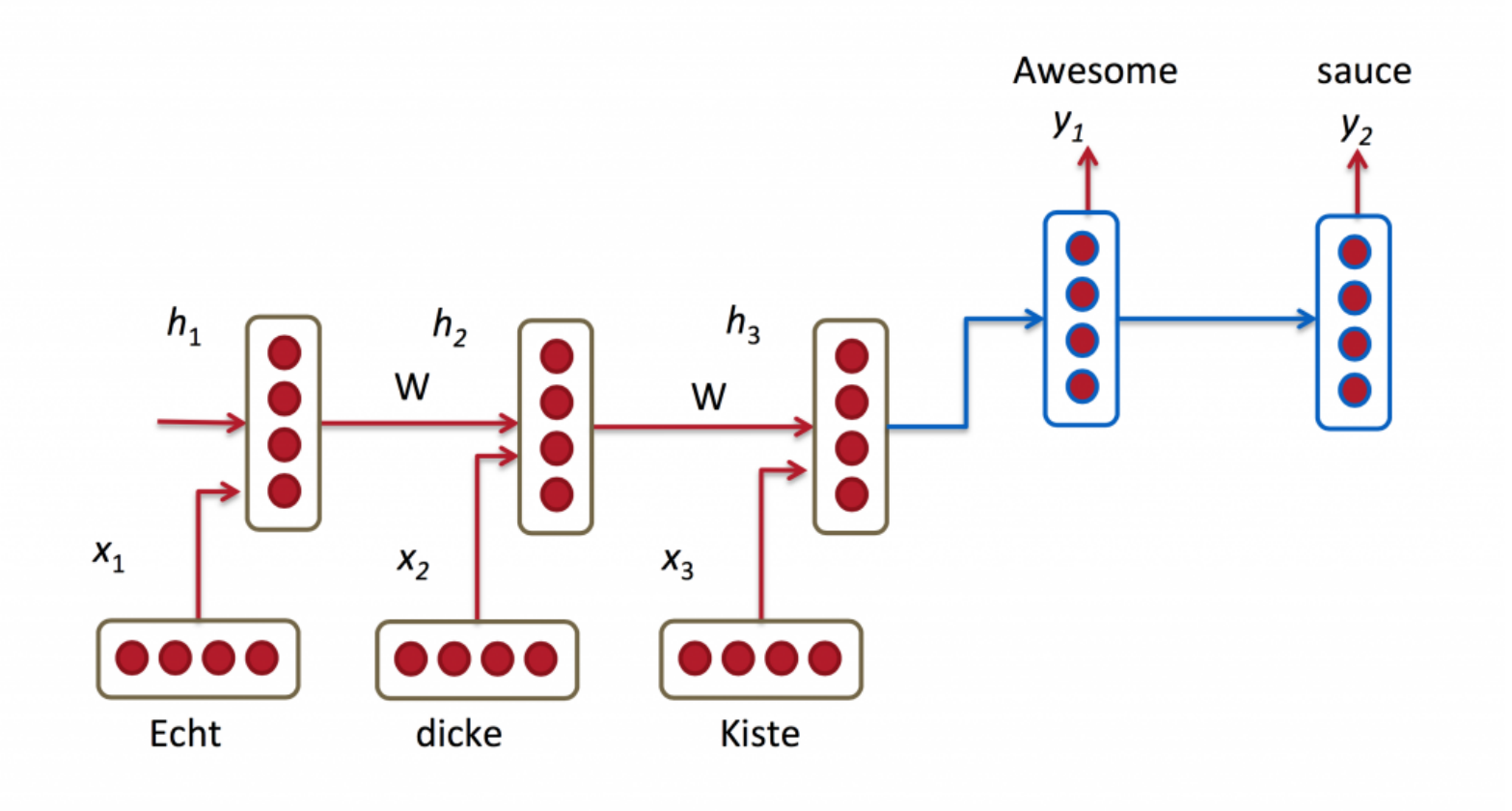


Image Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf