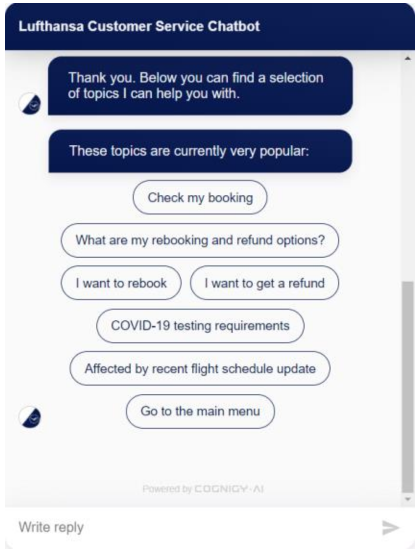
CSCE 689: Special Topics in Trustworthy NLP

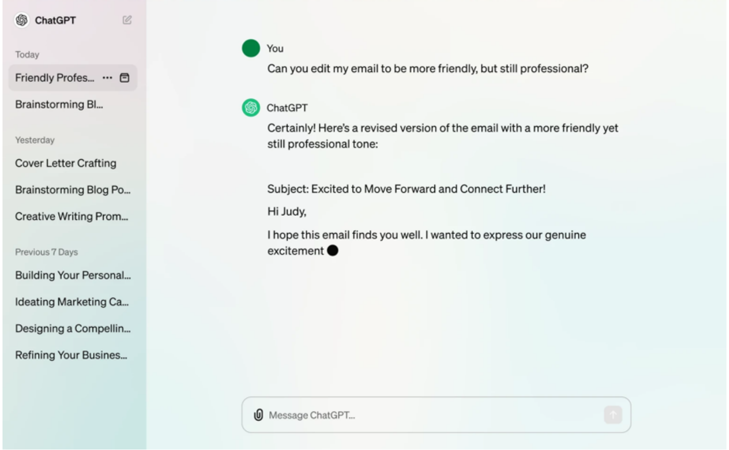
Lecture 2: Machine Learning Basics, Word Representations

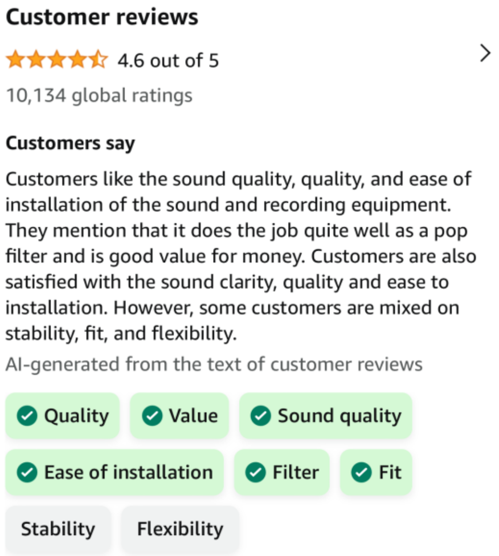
Kuan-Hao Huang
khhuang@tamu.edu

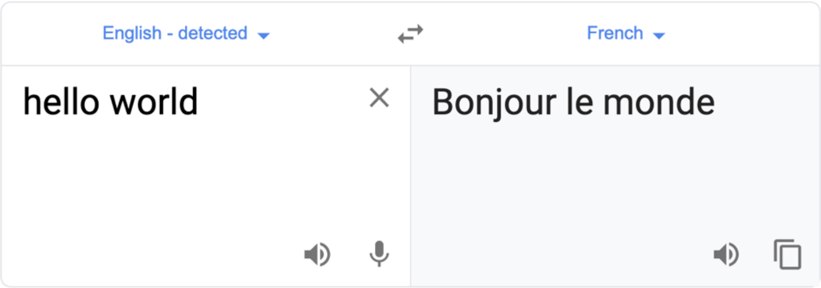
(Some slides adapted from Chris Manning, Dan Jurafsky, Danqi Chen, and Karthik Narasimhan)

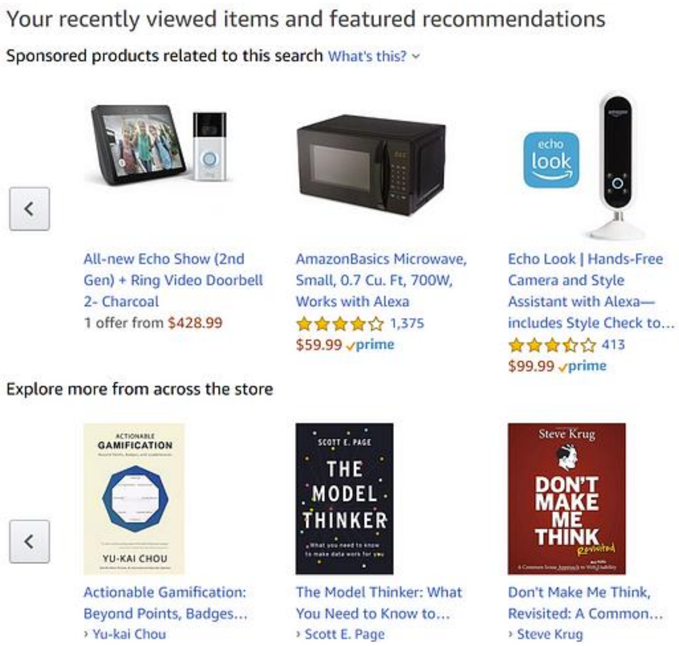
NLP Applications











How to formulate those problems?

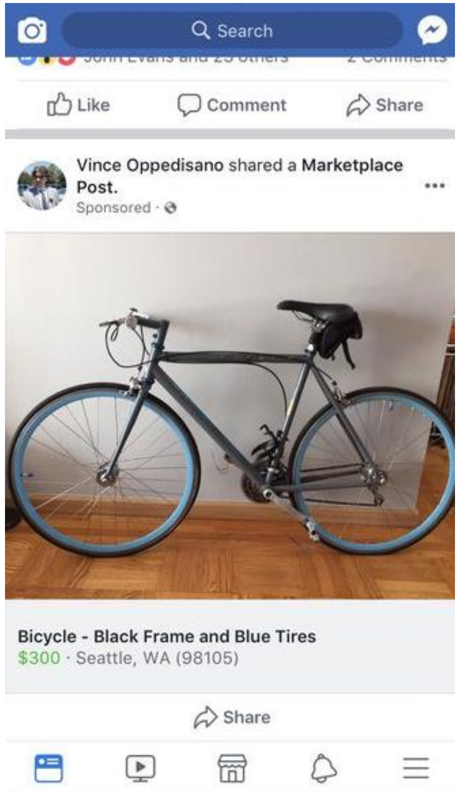
Formulation

- Build an NLP model to learn the association between input x and output y
- Input x: a sequence of symbols
  - What's the temperature now?
  - I like this restaurant.
- Output y: label
  - Category
  - Structure
  - Text
  - …

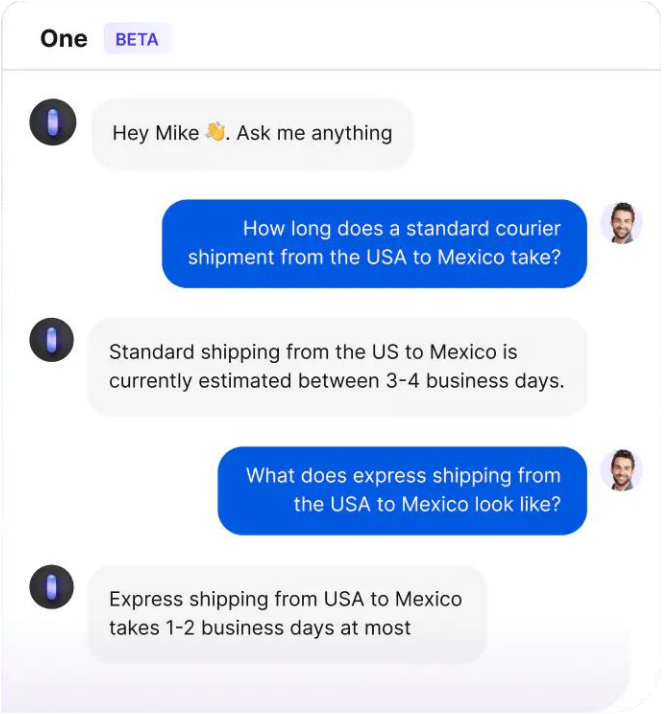
Text Classification

- Input x → Output y (category)

$y \in \{pos, neg\}$



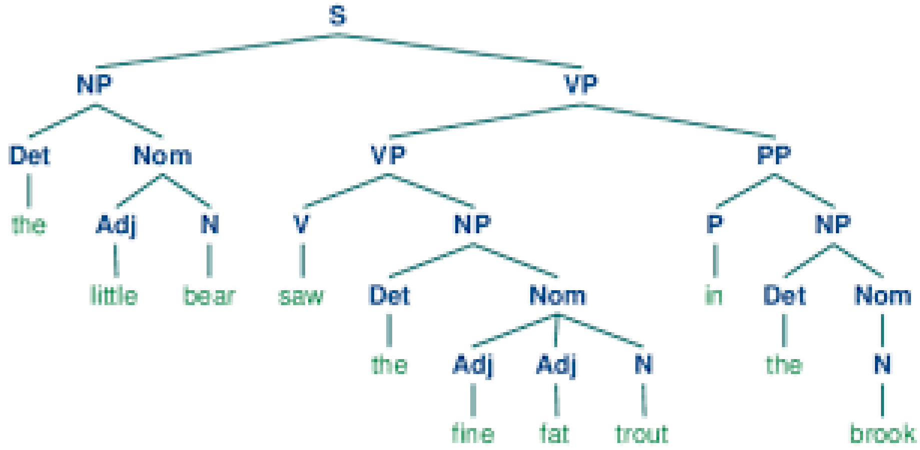
$y \in \{normal, fraud\}$

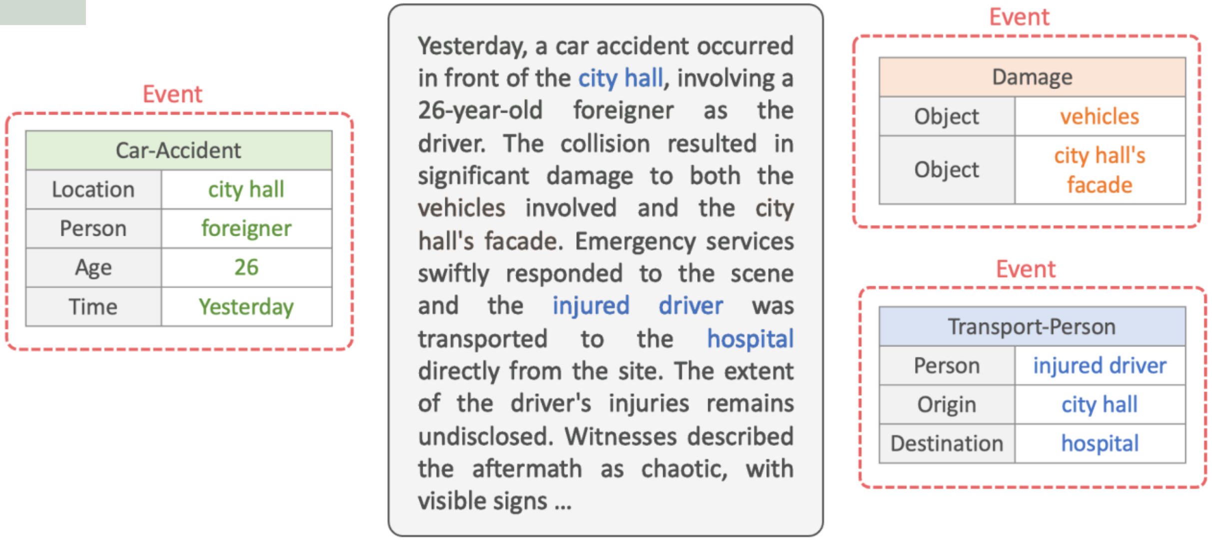


$y \in \{engineer, business, marketing, IT\ service\}$

Structured Classification

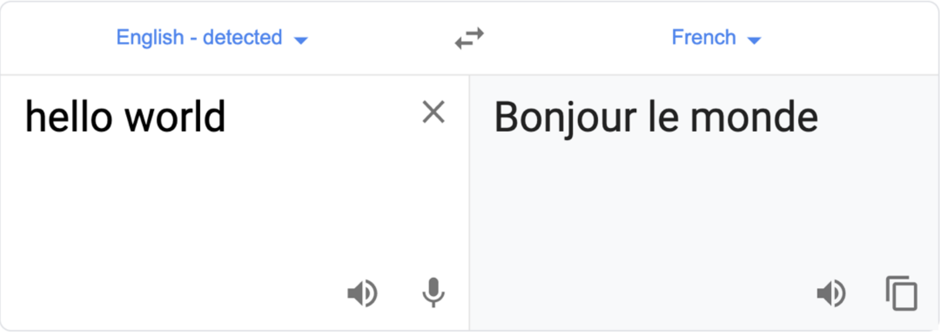
- Input x → Output y (structure)
  - Multiple labels with dependency

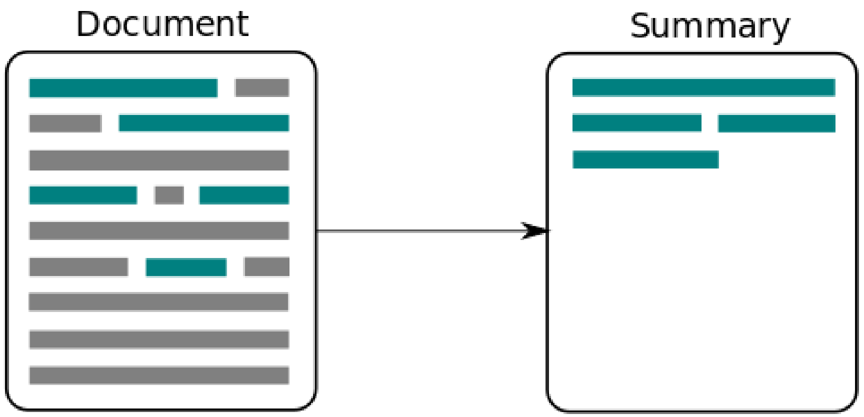




Generation

- Input x → Output y (text)
  - Also called sequence-to-sequence tasks






The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.

How did some suspect that Polo learned about China instead of by actually visiting it?

Answer: through contact with Persian traders

Classification vs. Generation

- There is no clear boundary between classification and generation
- Generation = Structured Token Classification

The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.

How did some suspect that Polo learned about China instead of by actually visiting it?

Answer: through

$y \in \{all\ possible\ words\}$ $\quad$ $y \in \{all\ possible\ words\}$

Classification vs. Generation

- There is no clear boundary between classification and generation
- Classification problems can be solved by generation

What's the sentiment of the following text: I very like this restaurant.

↓

The sentiment is positive.

Supervised Learning

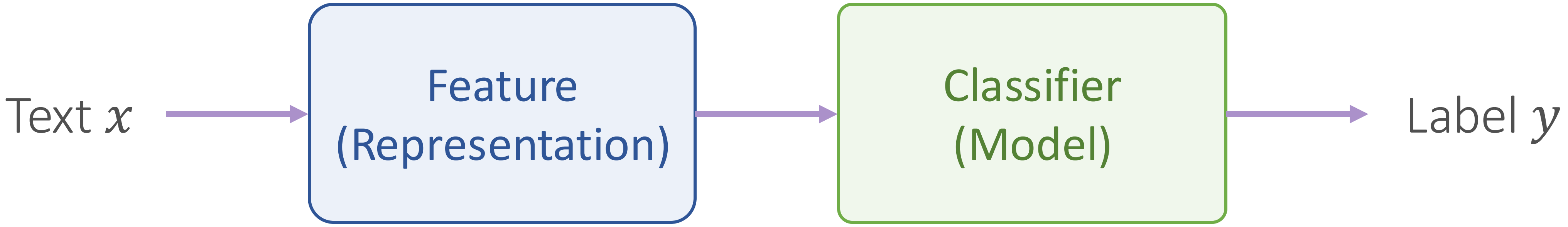
Training Stage

- Training data $\mathcal{D}_{train} = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
  - Example $x_i \in \mathcal{X}$, label $y_i \in C$
- Train a classifier(model) $f: \mathcal{X} \rightarrow C$ ← How to train?

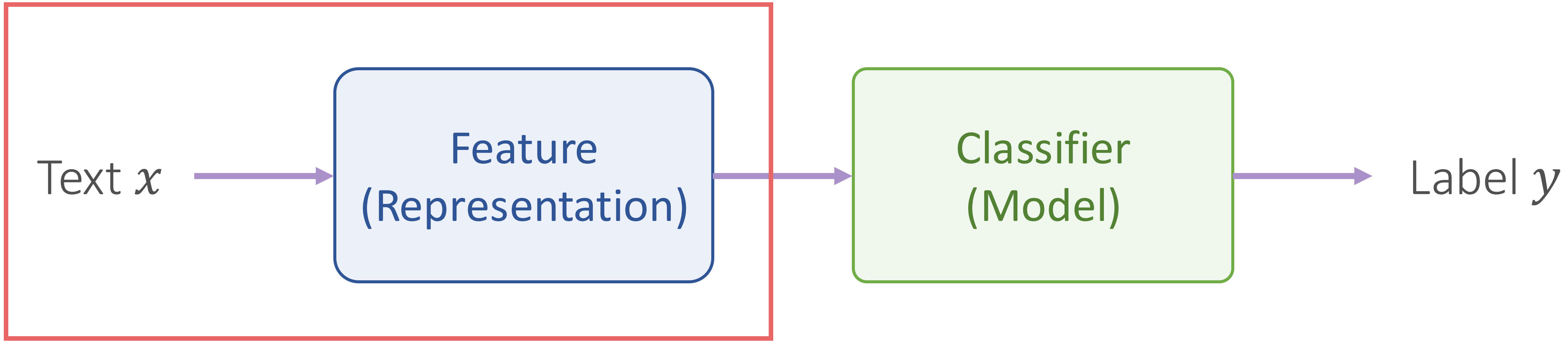
Testing Stage

- Testing data $\mathcal{D}_{test} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
- Make predictions $\tilde{y}_i = f(x_i)$
- Evaluate performance $\frac{1}{n} \sum_i S(y_i, \tilde{y}_i)$ ← Accuracy, F1 Score, etc.

A General Framework for Text Classification



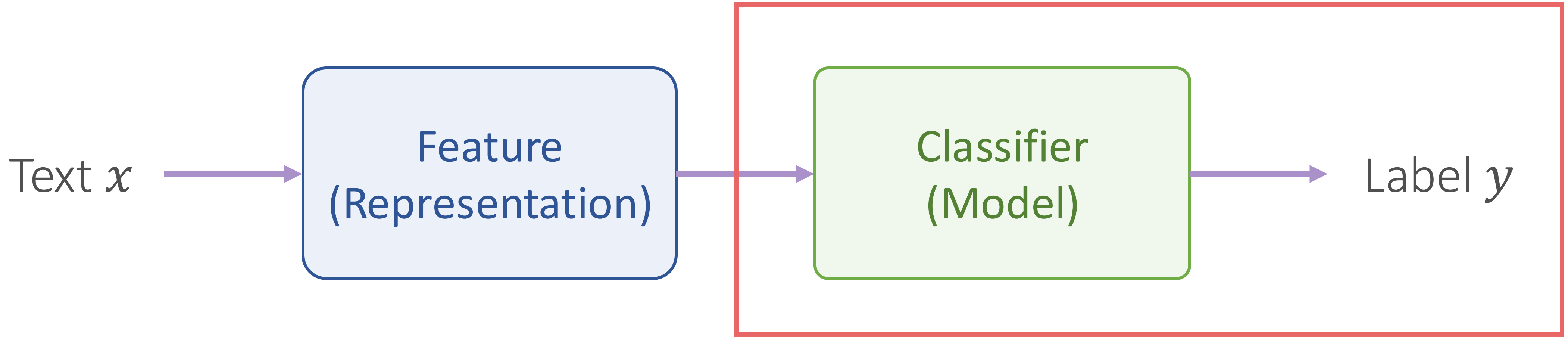
A General Framework for Text Classification



- Teach the model how to understand example x
- Convert the text to a mathematical form
  - The mathematical form captures essential characteristics of the text
- Bag-of-words, n-grams, word embeddings, etc.

We will talk about them later!

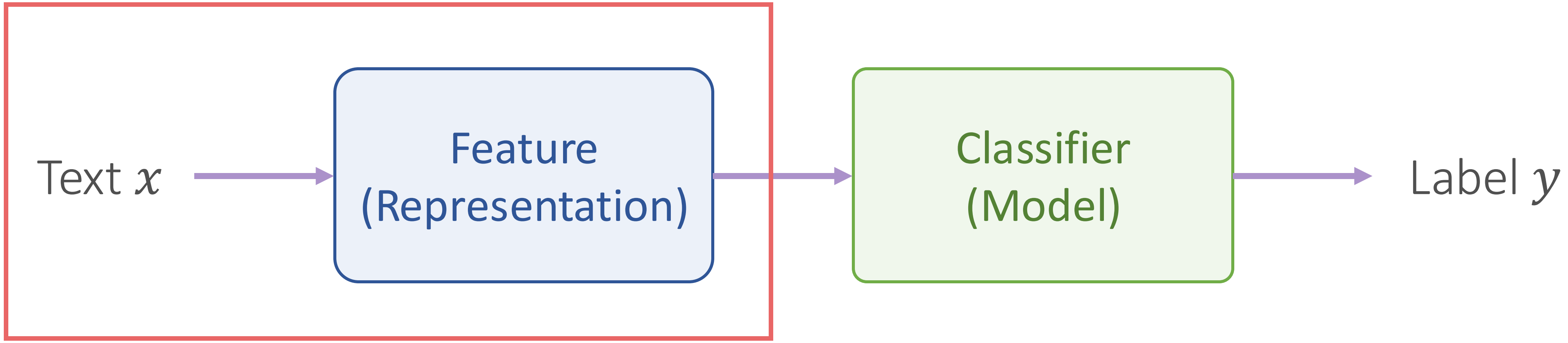
A General Framework for Text Classification



- Teach the model how to make prediction y
- Logistic regression, neural networks, CNN, RNN, LSTM, Transformers

We will talk about them later!

Bag-of-Words (BoW)



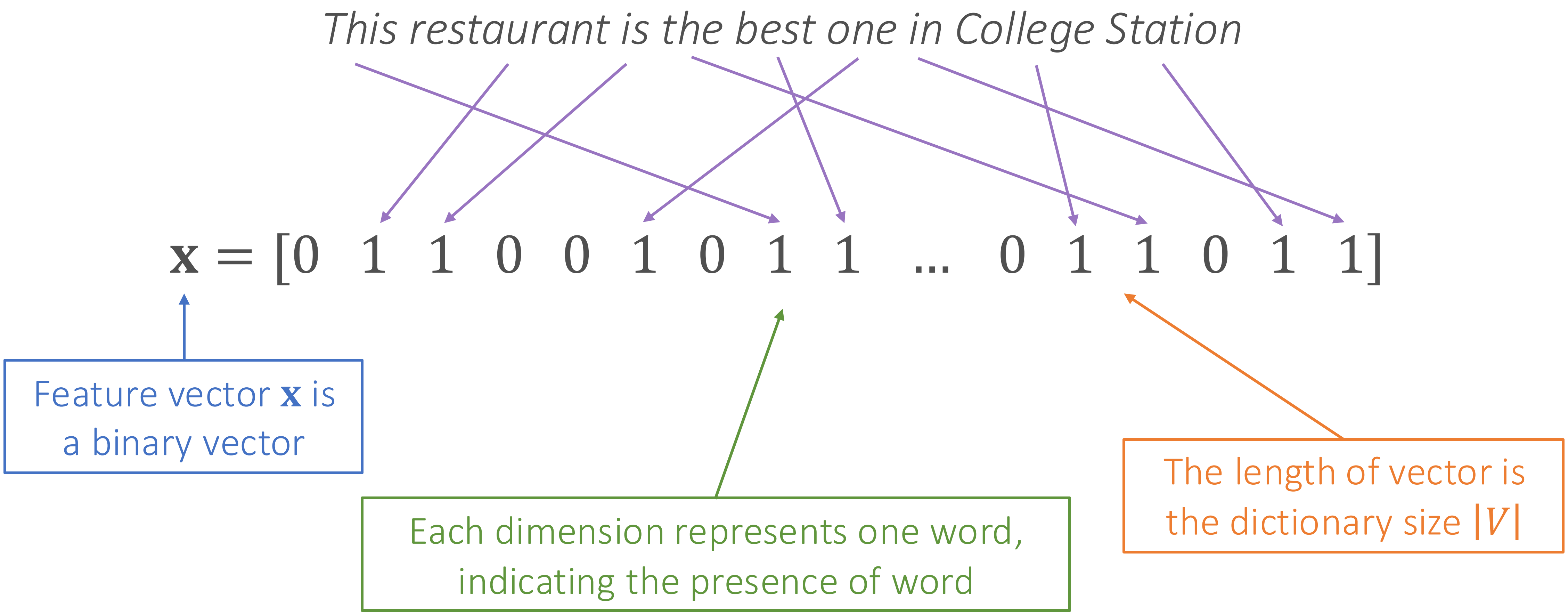
- Bag-of-Words (BoW)
  - Consider text as a set of words
- Easy, no effort required

Bag-of-Words (BoW)

This restaurant is the best one in College Station

I study natural language processing everyday

Bag-of-Words (BoW)



Advantages and disadvantages?

Bag-of-Words (BoW)

Bob likes Alice very much

Alice likes Bob very much

They will have the same BoW vector!

$$\mathbf{x} = [0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad \ldots \quad 0 \quad 1]$$

BoW fails to capture sentential structure

Any solutions?

N-Grams

Bob likes Alice very much

Unigram	*{Bob, likes, Alice, very, much}*
Bigram	*{Bob likes, likes Alice, Alice very, very much}*
Trigram	*{Bob likes Alice, likes Alice very, Alice very much}*
4-gram	*{Bob likes Alice very, likes Alice very much}*

Bag-of-N-Grams

Bob likes Alice very much

$$\mathbf{x} = [0\ \ 1\ \ \ldots\ \ 0\ \ 1\ \ 1\ \ 0\ \ \ldots\ \ 0\ \ 1]$$

Alice likes Bob very much

$$\mathbf{x} = [0\ \ 1\ \ \ldots\ \ 0\ \ 0\ \ 0\ \ 1\ \ \ldots\ \ 1\ \ 1]$$

BoW (unigram) features

Bigram features

We can consider trigrams, 4-grams, …

N-gram features capture more sentential structure

Other Variants

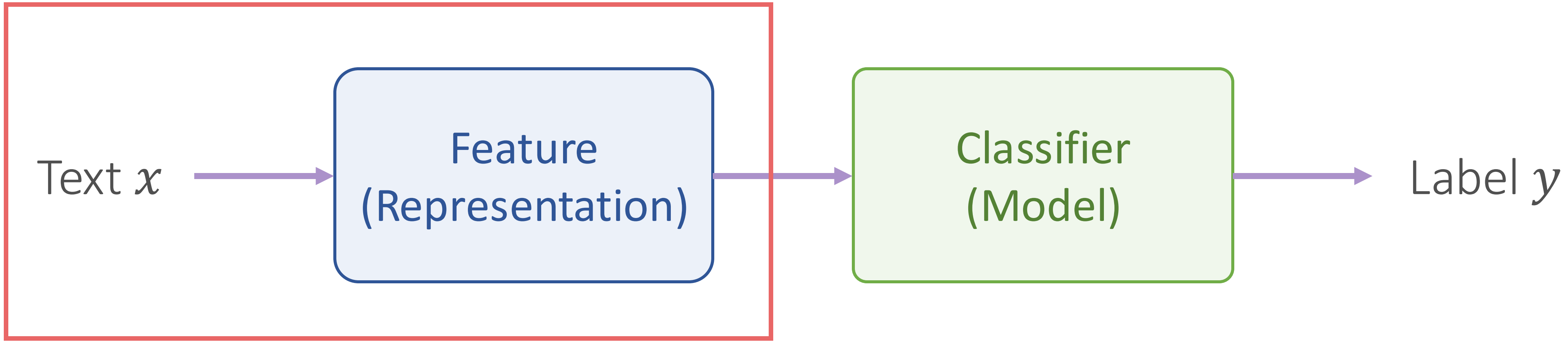
Binary BoW	$\mathbf{x} = [0 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; \dots \;\; 0 \;\; 1]$
Word Count	$\mathbf{x} = [0 \;\; 2 \;\; 1 \;\; 0 \;\; 0 \;\; 4 \;\; \dots \;\; 0 \;\; 3]$
Word Frequency	$\mathbf{x} = [0 \;\; 0.16 \;\; 0.08 \;\; 0 \;\; 0 \;\; 0.32 \;\; \dots \;\; 0 \;\; 0.24]$
TF-IDF	$\mathbf{x} = [0 \;\; 0.48 \;\; 0.02 \;\; 0 \;\; 0 \;\; 0.15 \;\; \dots \;\; 0 \;\; 0.88]$

$$f_w \cdot \log \frac{N}{n_t}$$

Term Frequency (TF)

Inverse Document Frequency (IDF)

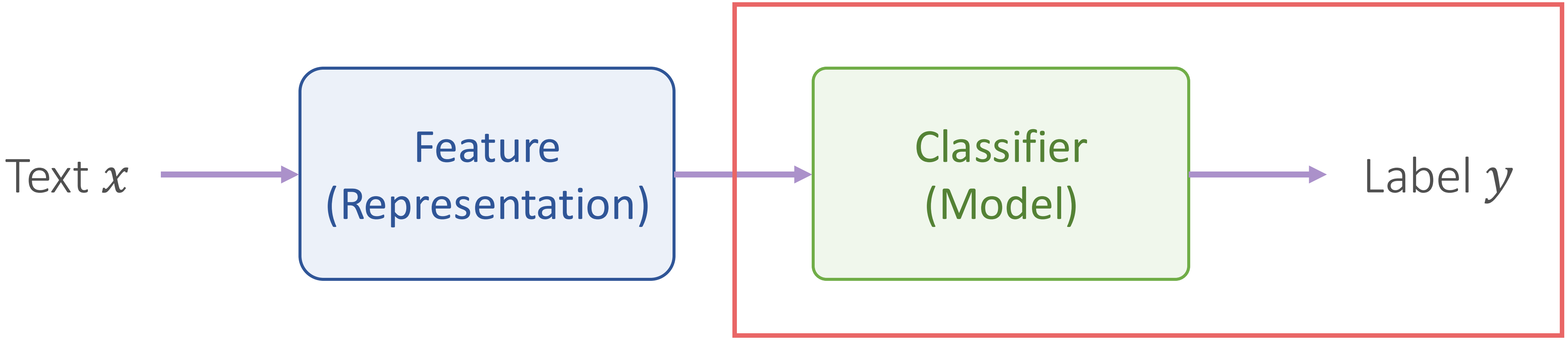
Bag-of-Words and Bag-of-N-Grams



- Bag-of-Words (BoW)
  - A set of words
- Bag-of-N-Grams
  - A set of n-grams

We will discuss "learnable" features later!

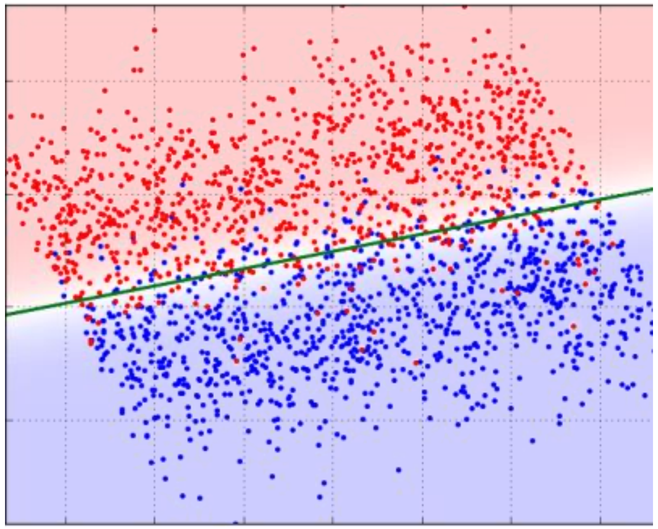
Logistic Regression

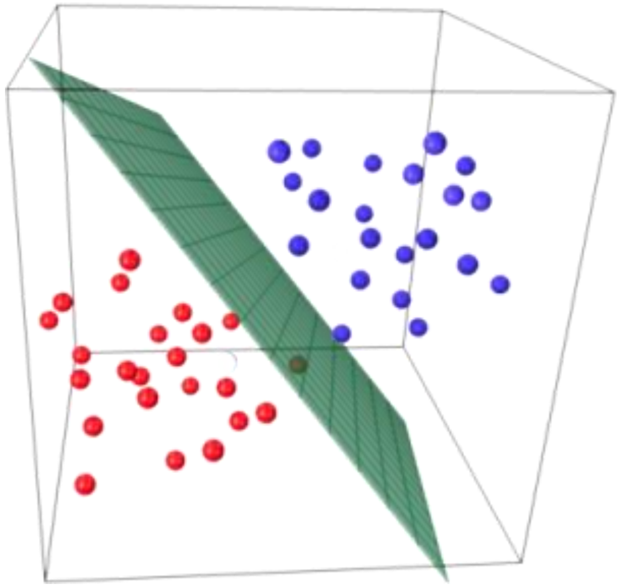


- Logistic regression
  - Find linear weights to map feature vector $\mathbf{x}$ to label y

Logistic Regression

- Let's start from binary classification
  - Input: feature vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_d]$
  - Output: label $y \in \{0, 1\}$
- Find a linear decision boundary to classify $\mathbf{x}$ into $\{0, 1\}$





https://blog.bigml.com/2016/09/28/logistic-regression-versus-decision-trees/
https://codesachin.wordpress.com/2015/08/16/logistic-regression-for-dummies/

Logistic Regression

Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_d]$ Label $y = 0 \text{ or } 1$

Weight Vector $\mathbf{w} = [w_1, w_2, w_3, \dots, w_d]$ Bias b

Learnable parameters

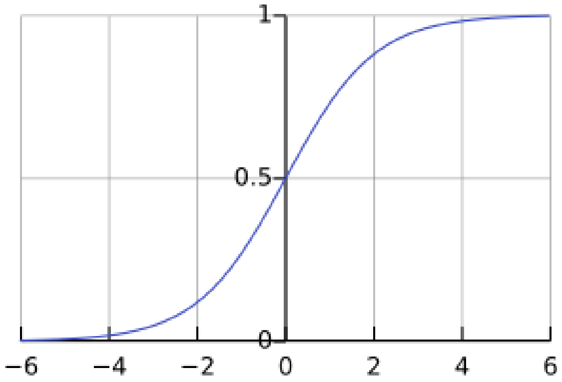
$$z = \mathbf{w} \cdot \mathbf{x} + b$$

$$\tilde{y} = P(y = 1 | \mathbf{x}) = \sigma(z)$$

Convert to probability

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Sigmoid Function



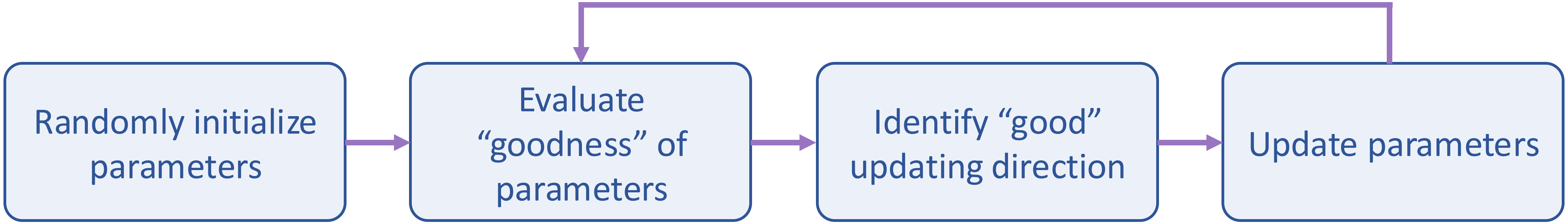
Decision boundary: $= \begin{cases} 1 & \text{If } \tilde{y} \geq 0.5 \\ 0 & \text{If } \tilde{y} < 0.5 \end{cases}$

How to Find The Best Parameters?

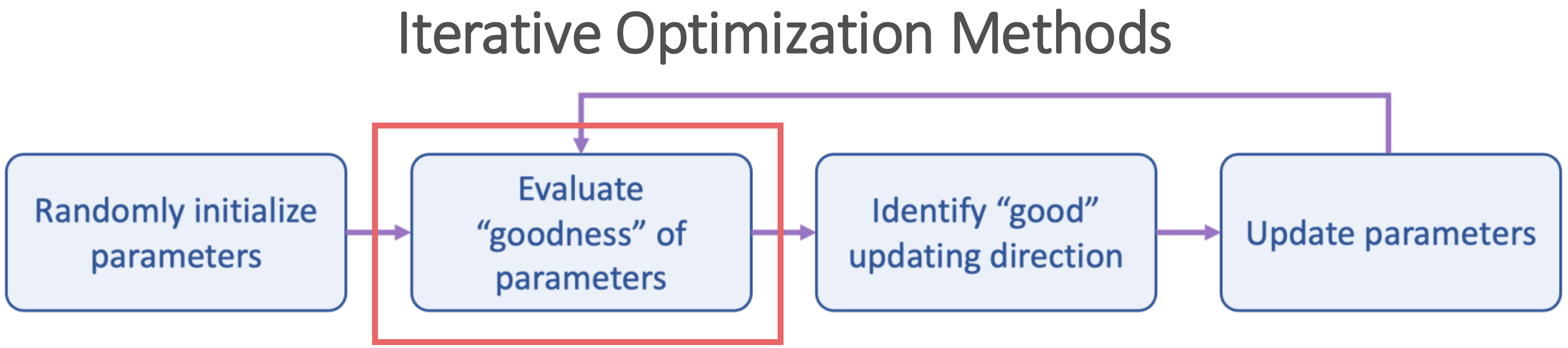
Weight Vector $\mathbf{w} = [w_1, w_2, w_3, \dots, w_d]$ Bias b

Learnable parameters

Iterative Optimization Methods



Loss Function



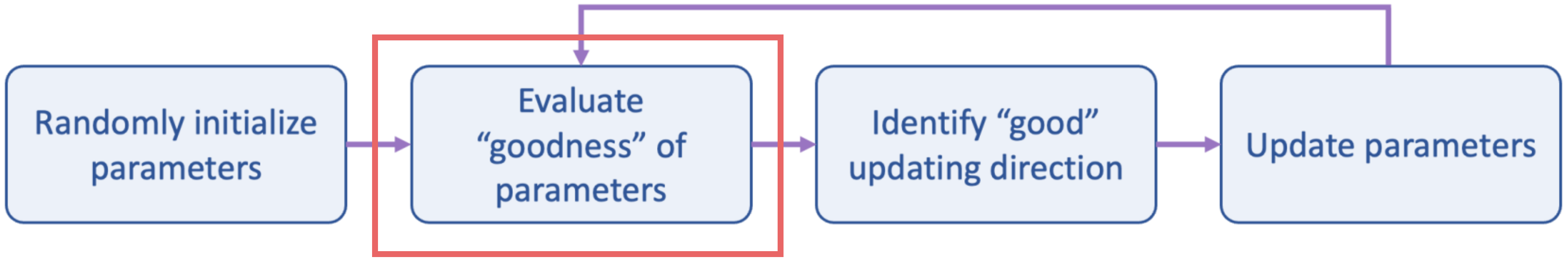
- For each training example (x, y)
- Output label probability is $\tilde{y} = P(y = 1|\,\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$

Cross Entropy Loss

$$\mathcal{L}_{CE}(y, \tilde{y}) = -[y \log \tilde{y} + (1 - y) \log(1 - \tilde{y})]$$

Loss Function

Iterative Optimization Methods



Cross Entropy Loss

$$\mathcal{L}_{CE}(y, \tilde{y}) = -[y \log \tilde{y} + (1 - y) \log(1 - \tilde{y})]$$

$y = 1$ and $\tilde{y} = 0.9$ $\quad \mathcal{L}_{CE} = -[1 \cdot \log 0.9 + 0 \cdot \log(0.1)] = -\log 0.9 \approx 0.105$

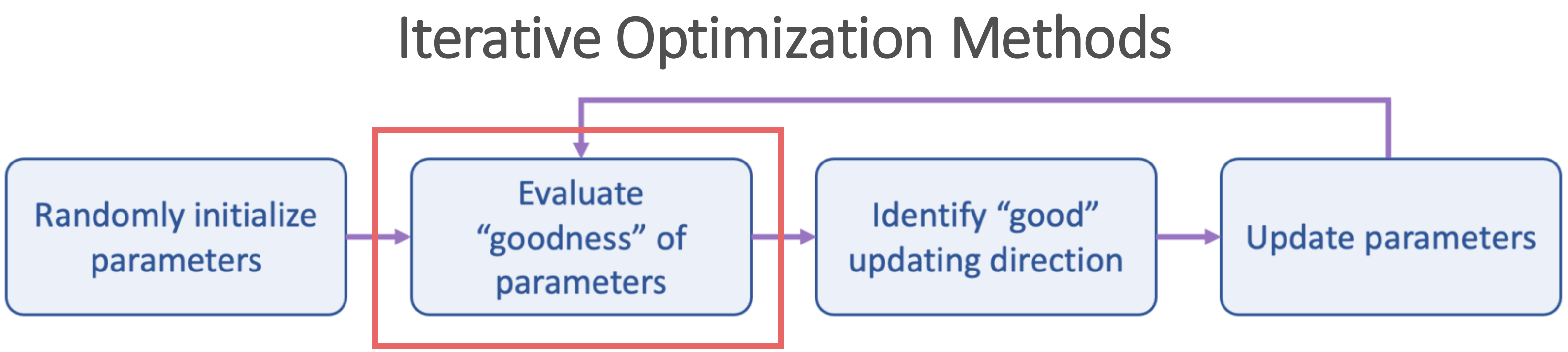
$y = 1$ and $\tilde{y} = 0.1$ $\quad \mathcal{L}_{CE} = -[1 \cdot \log 0.1 + 0 \cdot \log(0.9)] = -\log 0.1 \approx 2.302$

$y = 0$ and $\tilde{y} = 0.9$ $\quad \mathcal{L}_{CE} = -[0 \cdot \log 0.9 + 1 \cdot \log(0.1)] = -\log 0.1 \approx 2.302$

$y = 0$ and $\tilde{y} = 0.1$ $\quad \mathcal{L}_{CE} = -[0 \cdot \log 0.1 + 1 \cdot \log(0.9)] = -\log 0.9 \approx 0.105$

The lower the loss is, the more accurate the output probability is

Loss Function



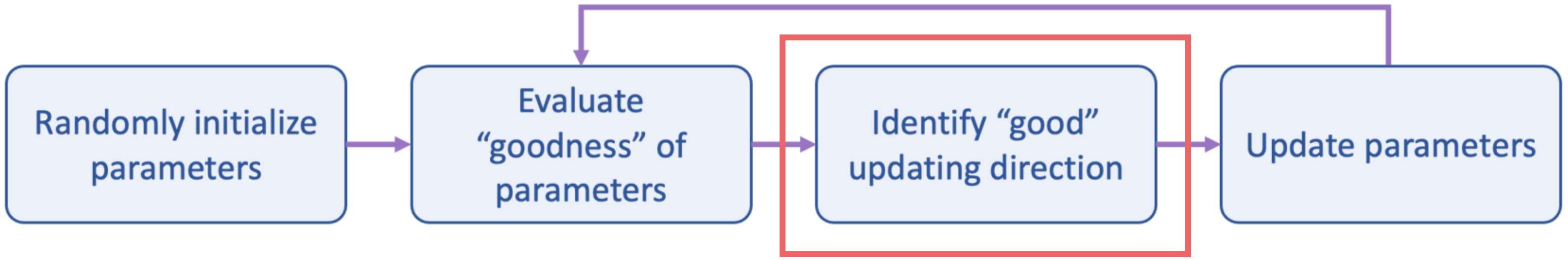
- Training data $\mathcal{D}_{train} = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
- Output labels probabilities $\tilde{y}_1, \tilde{y}_2, \dots, \tilde{y}_m$

Cross Entropy Loss

$$\mathcal{L}_{total} = -\frac{1}{m}\sum_i \mathcal{L}_{CE}(y_i, \tilde{y}_i) = -\frac{1}{m}\sum_i [y_i \log \tilde{y}_i + (1 - y_i)\log(1 - \tilde{y}_i)]$$

Optimization Objective

Iterative Optimization Methods



Cross Entropy Loss

$$\mathcal{L}_{total} = -\frac{1}{m}\sum_{i} \mathcal{L}_{CE}(y_i, \tilde{y_i})$$

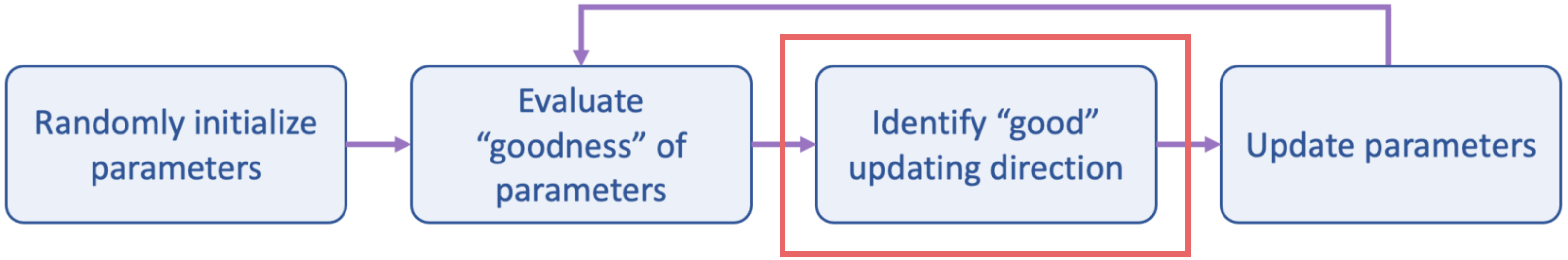
Parameters θ = Weight Vector $\mathbf{w} = [w_1, w_2, w_3, \ldots, w_d]$ Bias b

$$[\mathbf{w}^*; b^*] = \theta^* = \arg\min_{\theta} \mathcal{L}_{total}$$

Optimization objective

Gradient

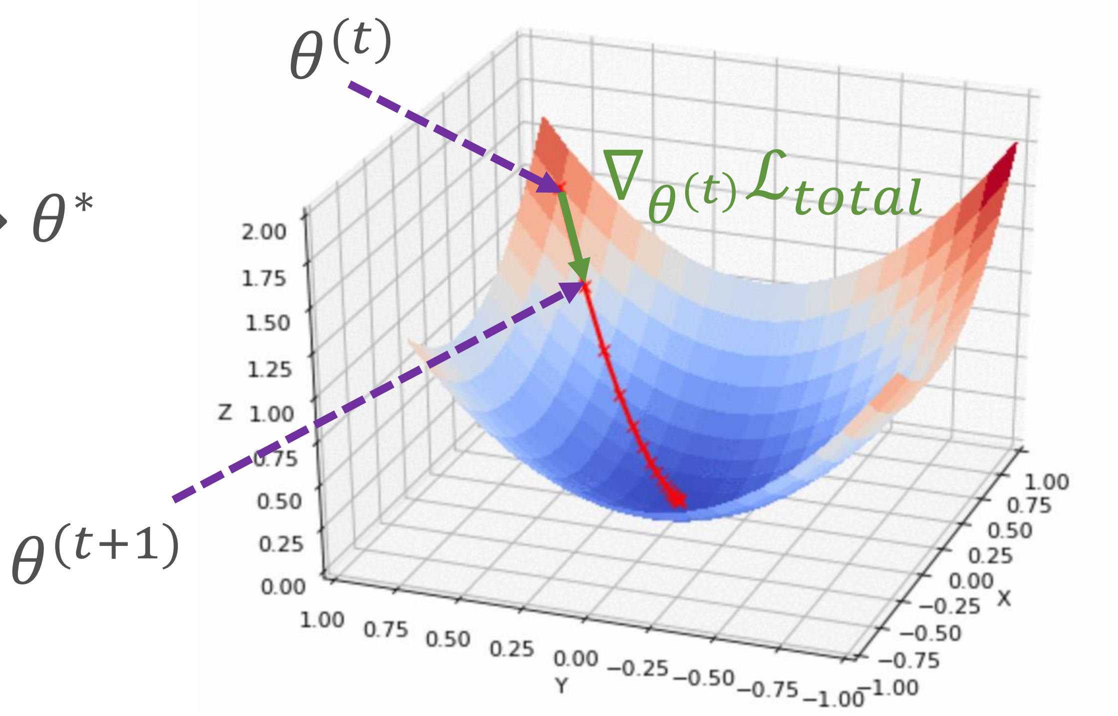
Iterative Optimization Methods



$$\theta^* = \arg\min_{\theta} \mathcal{L}_{total}$$

$$\theta^{(0)} \rightarrow \theta^{(1)} \rightarrow \theta^{(2)} \rightarrow \cdots \rightarrow \theta^{(k)} \rightarrow \cdots \rightarrow \theta^*$$

$\nabla_{\theta^{(t)}} \mathcal{L}_{total}$ is a "good" direction to minimize the objective



https://www.digitalocean.com/community/tutorials/intro-to-optimization-in-deep-learning-gradient-descent

Gradient

$$\nabla_\theta \mathcal{L}_{total} \qquad \frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}} \quad \frac{\partial \mathcal{L}_{total}}{\partial b}$$

$$\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}_j} = \frac{\partial \left(-\frac{1}{m}\sum_i [y_i \log \tilde{y}_i + (1-y_i)\log(1-\tilde{y}_i)]\right)}{\partial \mathbf{w}_j}$$

$$= \frac{\partial \left(-\frac{1}{m}\sum_i [y_i \log \sigma(z_i) + (1-y_i)\log(1-\sigma(z_i))]\right)}{\partial \mathbf{w}_j}$$

$$\tilde{y}_i = \sigma(z_i)$$
$$z_i = \mathbf{w} \cdot \mathbf{x}_i + b$$

$$= -\frac{1}{m}\sum_i \left[y_i \frac{\partial \log \sigma(z_i)}{\partial \mathbf{w}_j} + (1-y_i) \frac{\partial \log(1-\sigma(z_i))}{\partial \mathbf{w}_j} \right]$$

Gradient

$$\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}_j} = -\frac{1}{m}\sum_i \left[y_i \frac{\partial \log \sigma(z_i)}{\partial \mathbf{w}_j} + (1-y_i)\frac{\partial \log(1-\sigma(z_i))}{\partial \mathbf{w}_j} \right]$$

$$\frac{\partial \log \sigma(z_i)}{\partial \mathbf{w}_j} = \frac{1}{\sigma(z_i)} \cdot [\sigma(z_i)(1-\sigma(z_i))] \cdot \mathbf{x}_{i,j} = (1-\sigma(z_i))\mathbf{x}_{i,j}$$

$$\boxed{\sigma'(z) = \sigma(z)(1-\sigma(z))}$$

$$\frac{\partial \log(1-\sigma(z_i))}{\partial \mathbf{w}_j} = \frac{1}{1-\sigma(z_i)} \cdot [-\sigma(z_i)(1-\sigma(z_i))] \cdot \mathbf{x}_{i,j} = -\sigma(z_i)\mathbf{x}_{i,j}$$

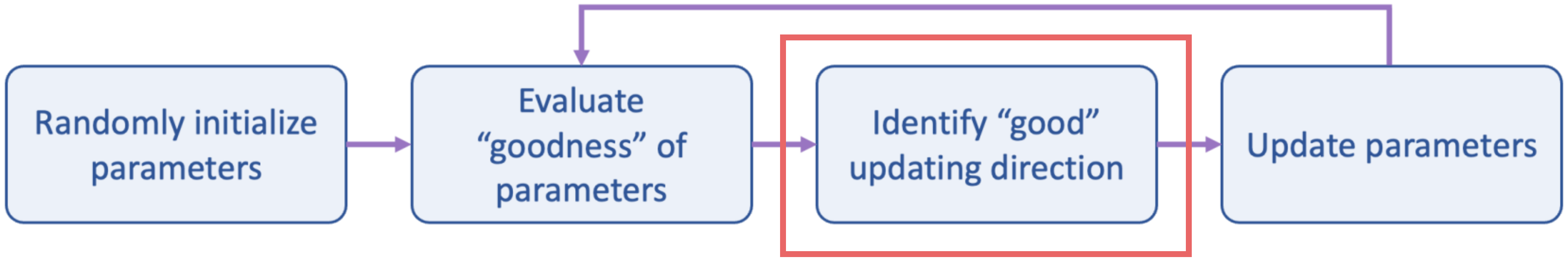
$$\boxed{(1-\sigma(z))' = -\sigma(z)(1-\sigma(z))}$$

$$\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}_j} = -\frac{1}{m}\sum_i [y_i(1-\sigma(z_i))\mathbf{x}_{i,j} + (1-y_i)(-\sigma(z_i)\mathbf{x}_{i,j})]$$

$$= -\frac{1}{m}\sum_i (y_i - \sigma(z_i))\mathbf{x}_{i,j} = \frac{1}{m}\sum_i (\tilde{y}_i - y_i)\mathbf{x}_{i,j}$$

Gradient

Iterative Optimization Methods

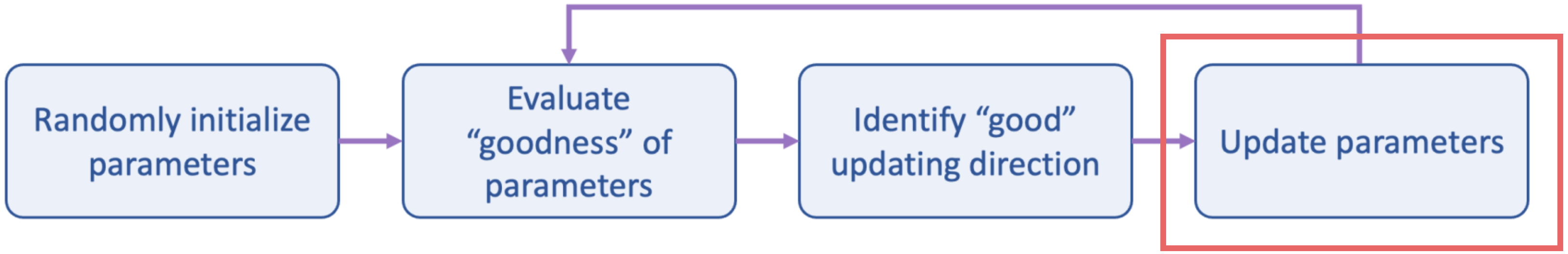


$$\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}} = \sum_{i=1}^{m} (\tilde{y}_i - y_i)\mathbf{x}_i$$

$$\frac{\partial \mathcal{L}_{total}}{\partial b} = \sum_{i=1}^{m} (\tilde{y}_i - y_i)$$

Gradient Descent

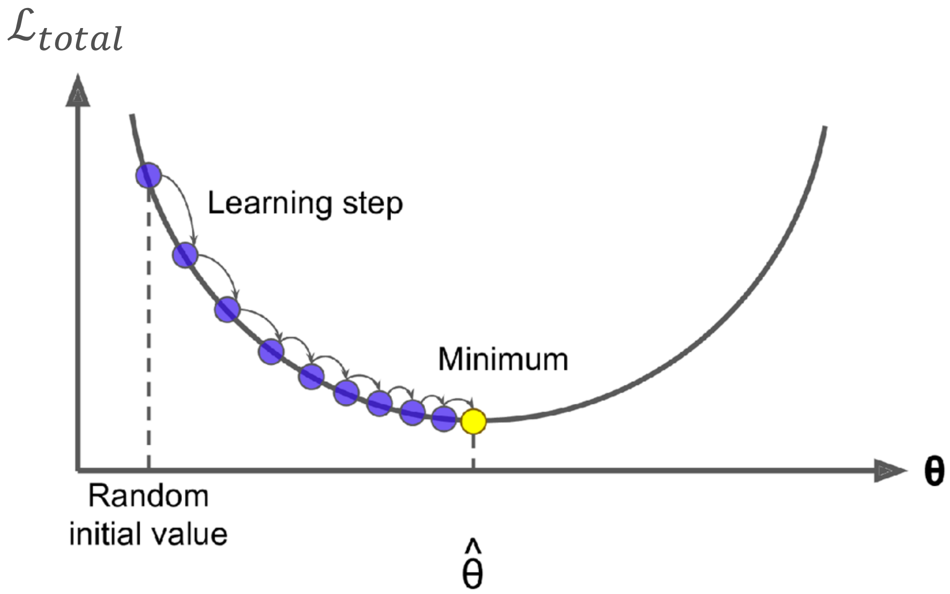
Iterative Optimization Methods



$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \, \nabla_{\mathbf{w}} \mathcal{L}_{total}$$

$$b^{(t+1)} = b^{(t)} - \eta \, \nabla_{b} \mathcal{L}_{total}$$

Learning step



https://insightfultscript.com/collections/programming/machine-learning/sgd/

Training Process

Iterative Optimization Methods

Randomly initialize parameters → Evaluate "goodness" of parameters → Identify "good" updating direction → Update parameters

Cross Entropy Loss

$$\mathcal{L}_{total} = -\frac{1}{m}\sum_{i}\mathcal{L}_{CE}\left(y_i, \tilde{y}_i; \mathbf{w}^{(t)}, b^{(t)}\right)$$

$$\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}^{(t)}} = \sum_{i=1}^{m}(\tilde{y}_i - y_i)\mathbf{x}_i$$

$$\frac{\partial \mathcal{L}_{total}}{\partial b^{(t)}} = \sum_{i=1}^{m}(\tilde{y}_i - y_i)$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta\,\nabla_{\mathbf{w}}\mathcal{L}_{total}$$

$$b^{(t+1)} = b^{(t)} - \eta\,\nabla_{b}\mathcal{L}_{total}$$

From Binary to Multiclass Classification

- Logistic Regression for binary classification

Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_d]$ Label $y = 0 \; or \; 1$

Weight Vector $\mathbf{w} = [w_1, w_2, w_3, \dots, w_d]$ Bias b

Learnable Parameters

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

$$P(y = 1 | \mathbf{x}) = \sigma(z)$$

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Sigmoid Function

$$\text{Prediction} = \begin{cases} 1 & \text{If } P(y = 1 | \mathbf{x}) \geq 0.5 \\ 0 & \text{If } P(y = 1 | \mathbf{x}) < 0.5 \end{cases}$$

From Binary to Multiclass Classification

- Logistic Regression for multiclass classification

Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_d]$ Label $y = 0, 1, \dots, C-1$

Weight Vectors $\mathbf{w}_c = [w_{c,1}, w_{c,2}, w_{c,3}, \dots, w_{c,d}]$ Bias b_c

Learnable Parameters

$$z_c = \mathbf{w}_c \cdot \mathbf{x} + b_c$$

$$P(y = c | \mathbf{x}) = \mathrm{softmax}(z_c)$$

$$\mathrm{softmax}(z_c) = \frac{e^{z_c}}{\sum_t e^{z_t}}$$

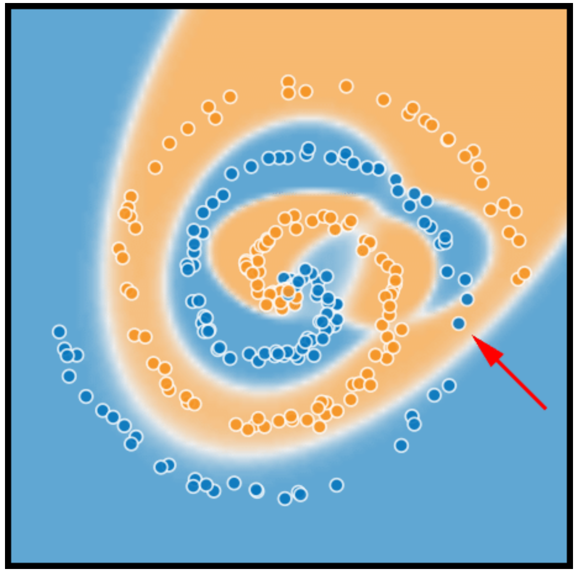
Softmax Function

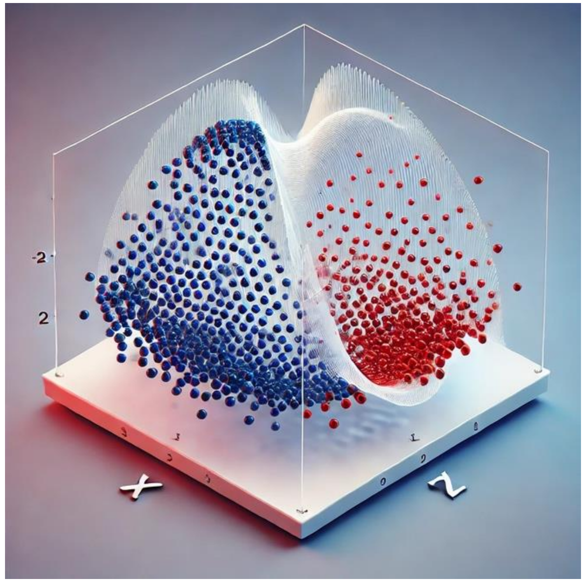
Prediction $= \arg\max_c P(y = c | \mathbf{x})$

Logistic Regression

- Logistic regression
  - Find linear weights to map feature vector $\mathbf{x}$ to label y

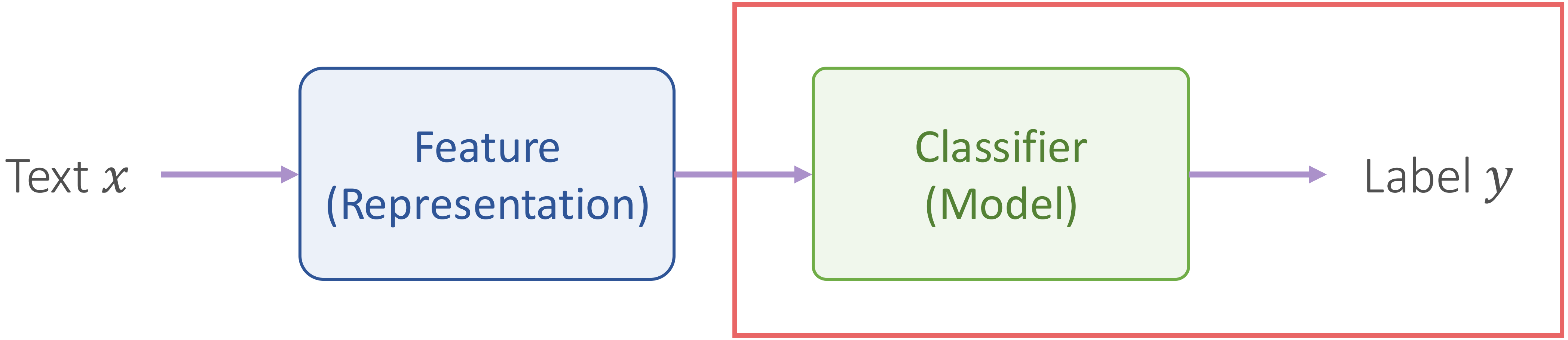
What if linear weights are not powerful enough?





https://www.researchgate.net/publication/344636757_CQNN_Convolutional_Quadratic_Neural_Networks
https://chatgpt.com/

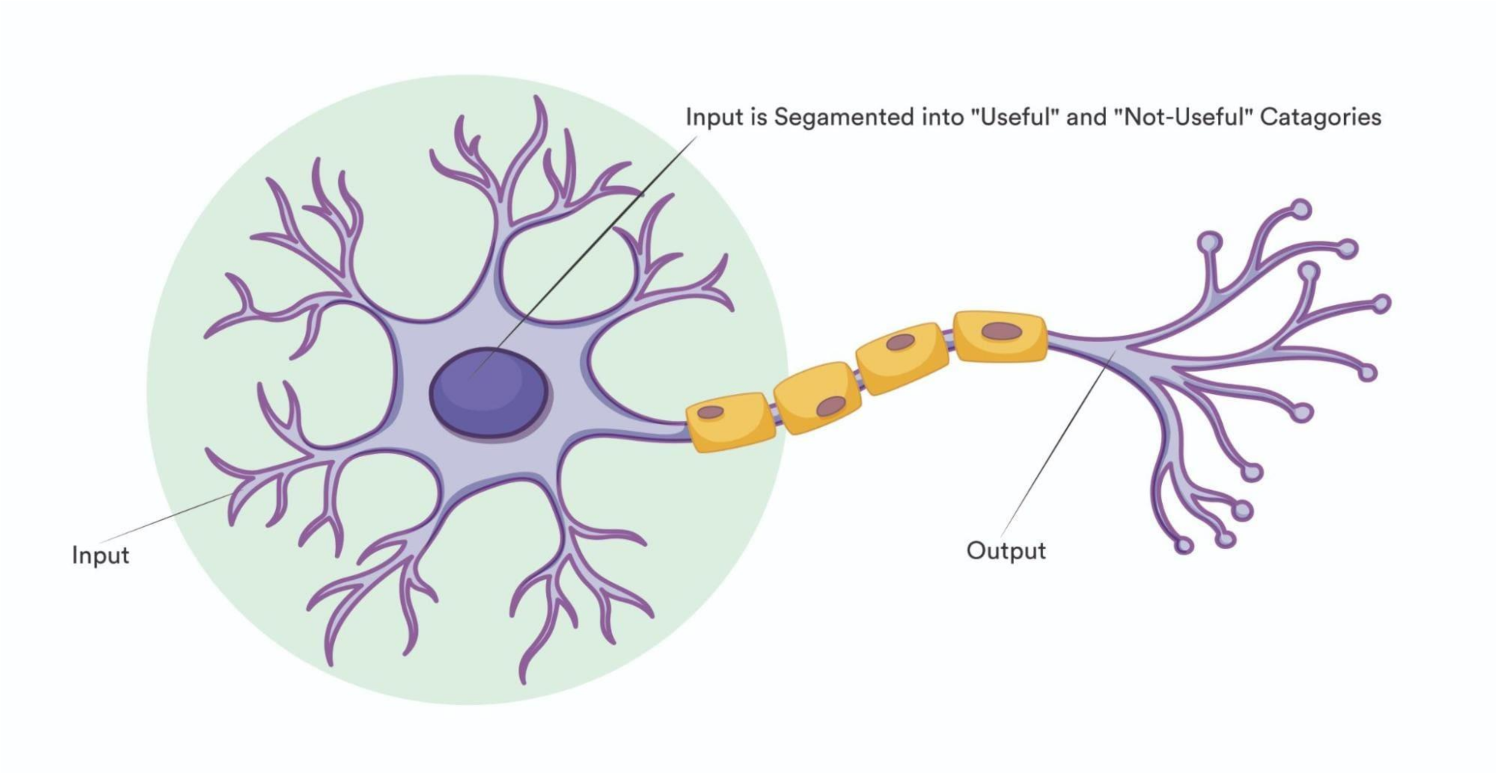
Neural Networks



- Neural Networks
  - Find a non-linear decision boundary to map feature vector $\mathbf{x}$ to label y

Biological Neurons

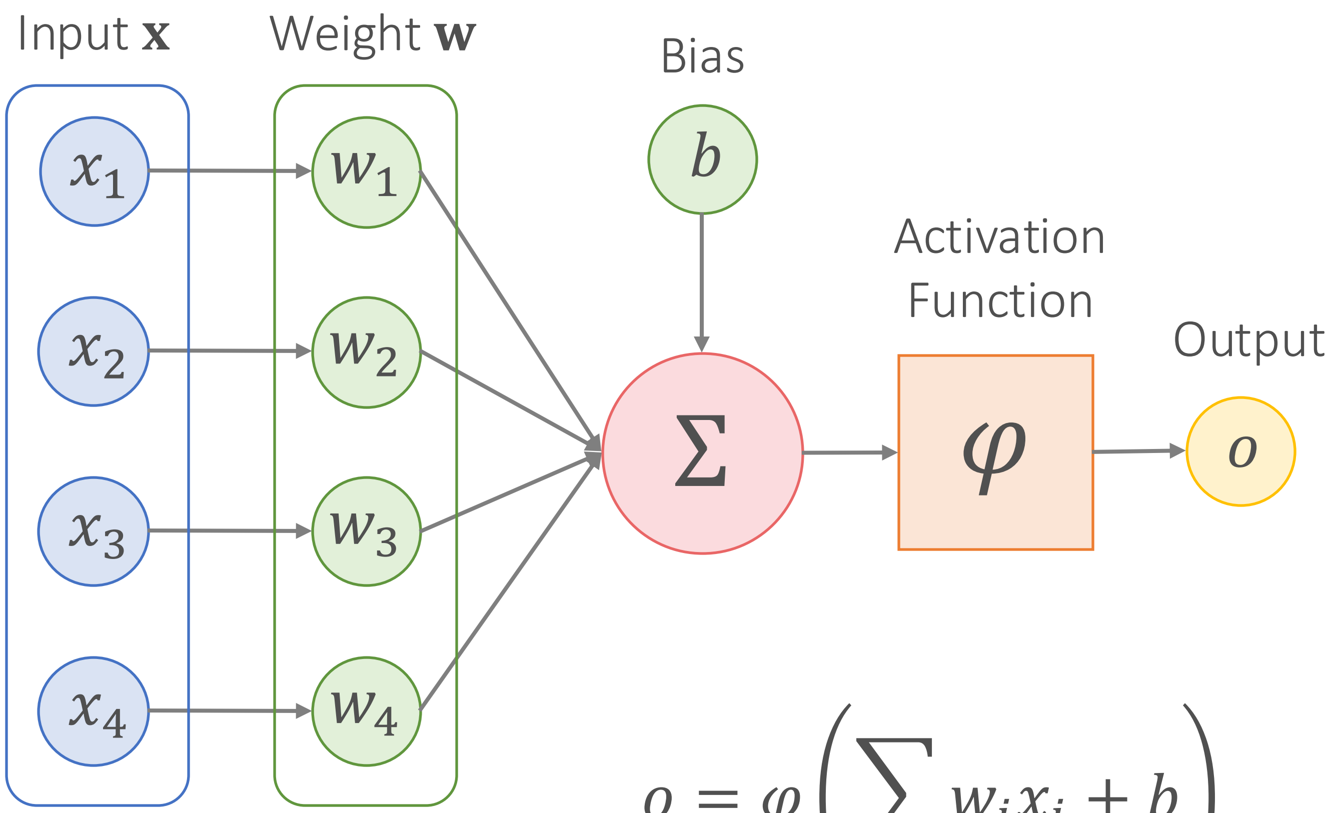
Neuron activation: A neuron becomes active to transmit information when it receives sufficient input from other neurons



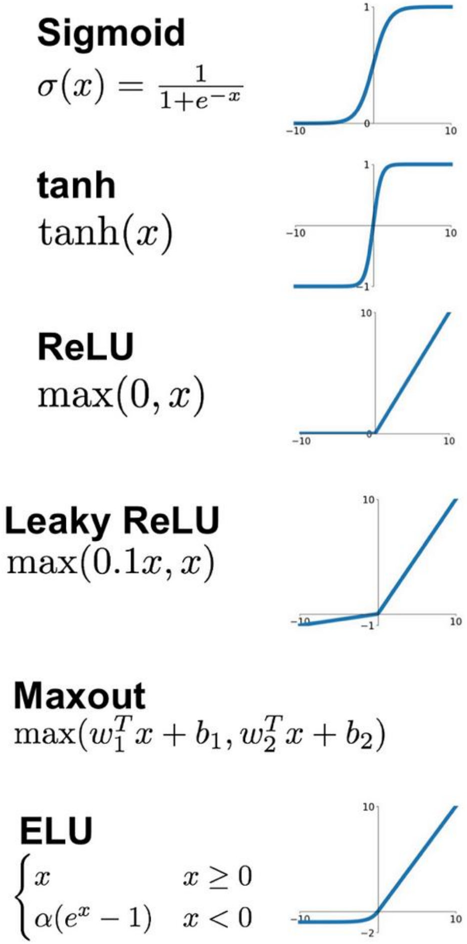
https://www.kdnuggets.com/2022/06/activation-functions-work-deep-learning.html

Neurons in Neural Networks

Mimic the behavior of neurons to transmit information

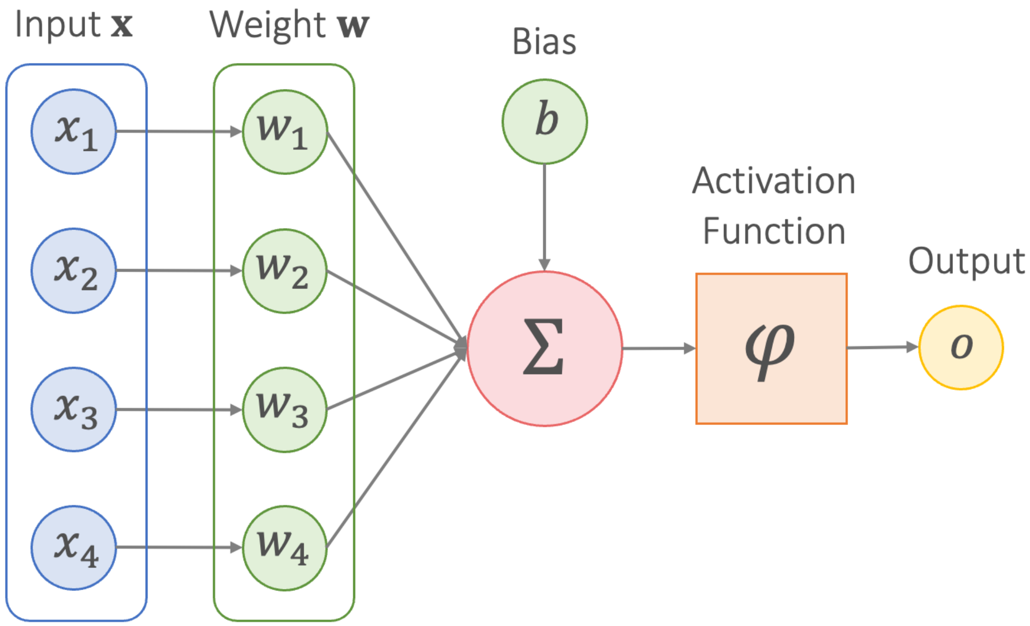


$$o = \varphi\left(\sum_i w_i x_i + b\right)$$



https://blog.devops.dev/exploring-activation-functions-in-deep-learning-properties-derivatives-and-impact-on-model-7585aad8a757

Neurons vs. Logistic Regression



Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \ldots, x_d]$

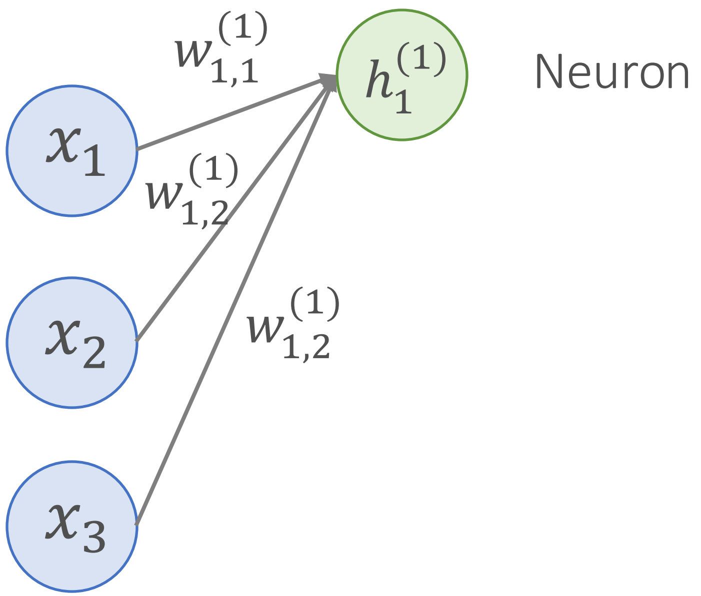
Weight Vector $\mathbf{w} = [w_1, w_2, w_3, \ldots, w_d]$

Bias b

$$o = \varphi\left(\sum_i w_i x_i + b\right)$$

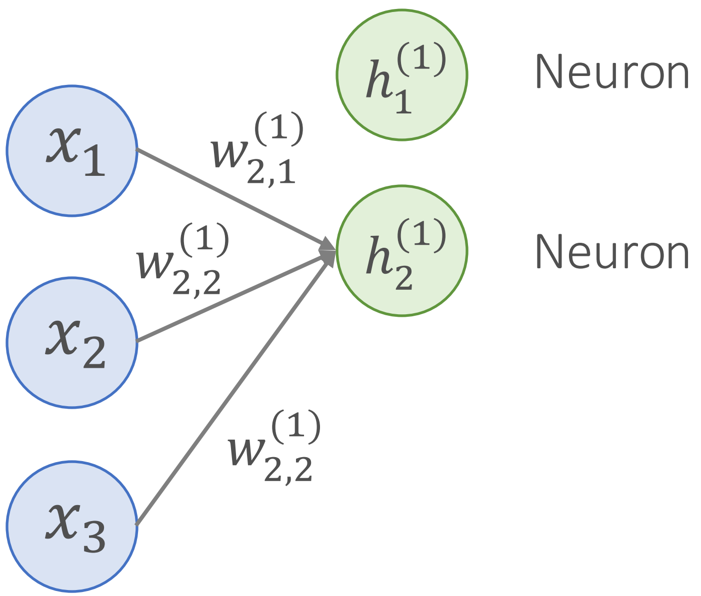
$$\tilde{y} = \sigma\left(\sum_i w_i x_i + b\right)$$

Multilayer Perceptron (MLP)



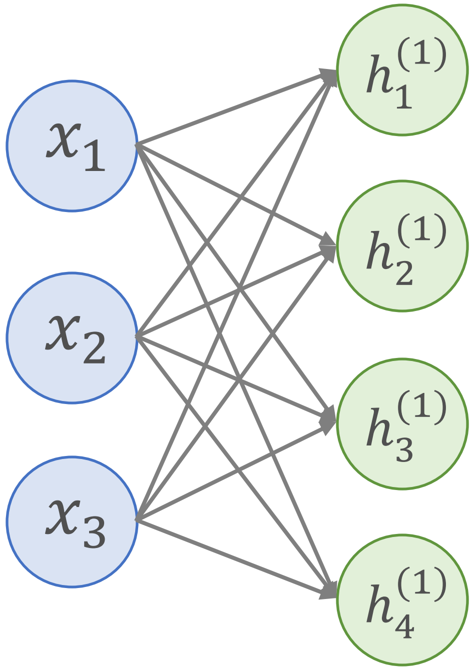
$$h_1^{(1)} = \varphi\left(\sum_i w_{1,i}^{(1)} x_i + b\right) = \varphi\left(\mathbf{w}_1^{(1)} \cdot \mathbf{x} + b\right)$$

Multilayer Perceptron (MLP)



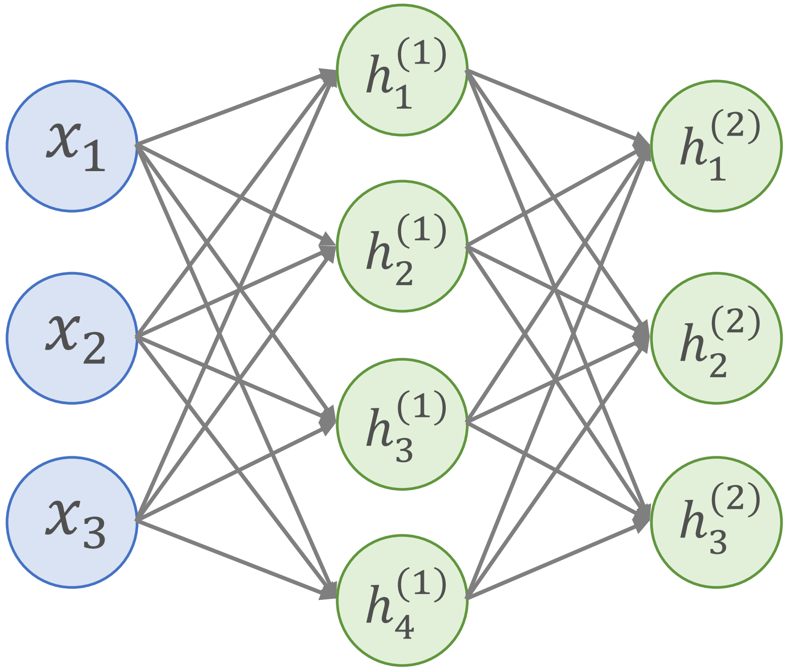
$$h_2^{(1)} = \varphi\left(\sum_i w_{2,i}^{(1)} x_i + b\right) = \varphi\left(\mathbf{w}_2^{(1)} \cdot \mathbf{x} + b\right)$$

Multilayer Perceptron (MLP)



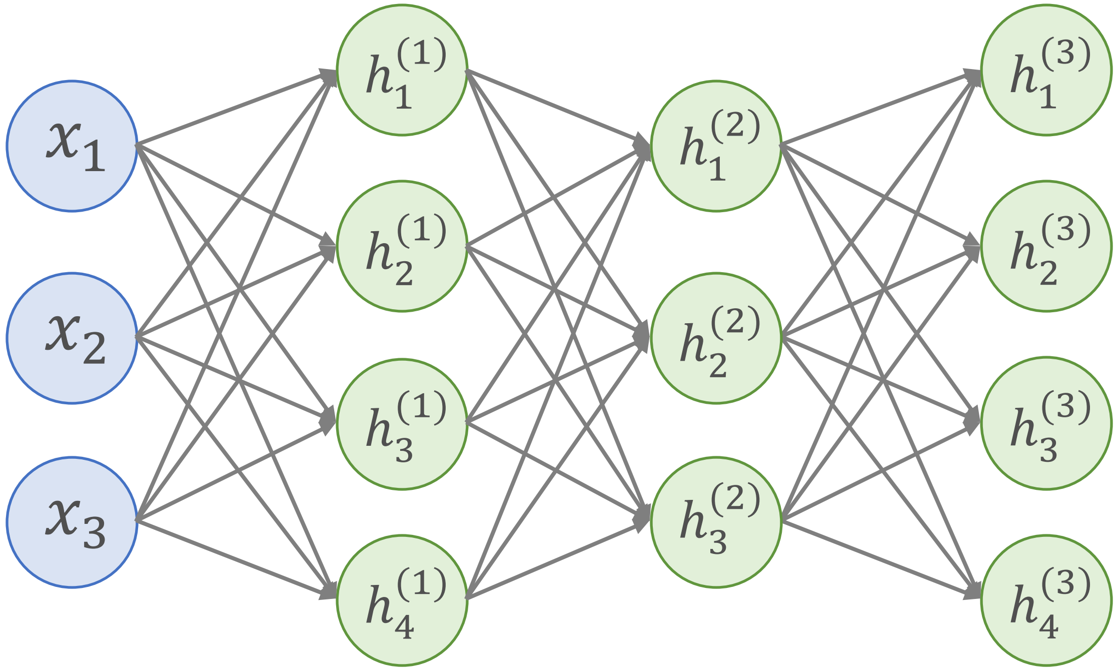
$$\mathbf{h}^{(1)} = \varphi\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right)$$

Multilayer Perceptron (MLP)



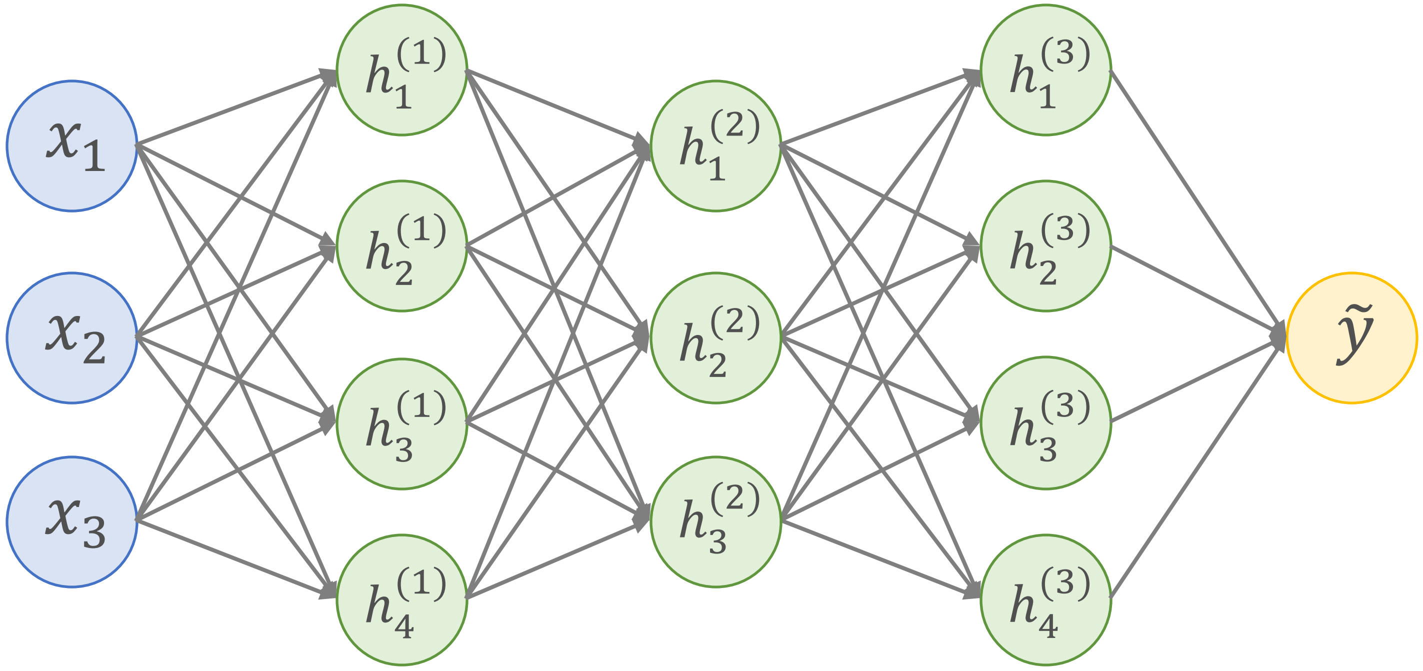
$$\mathbf{h}^{(2)} = \varphi\left(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}\right)$$

Multilayer Perceptron (MLP)



$$\mathbf{h}^{(3)} = \varphi\left(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)}\right)$$

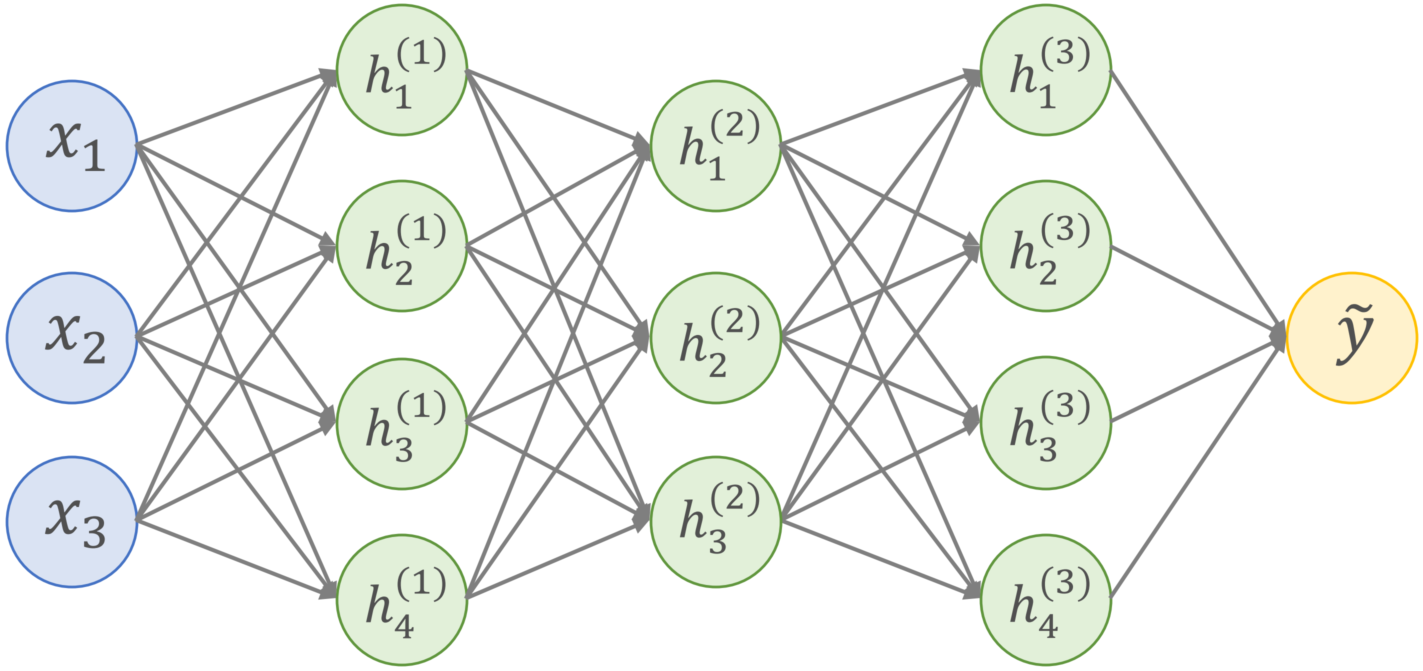
Multilayer Perceptron (MLP)



Decision boundary: $= \begin{cases} 1 & \text{If } \tilde{y} \geq 0.5 \\ 0 & \text{If } \tilde{y} < 0.5 \end{cases}$

$$\tilde{y} = \sigma\left(\mathbf{W}^{(o)}\mathbf{h}^{(3)} + \mathbf{b}^{(o)}\right)$$

Optimization Objective



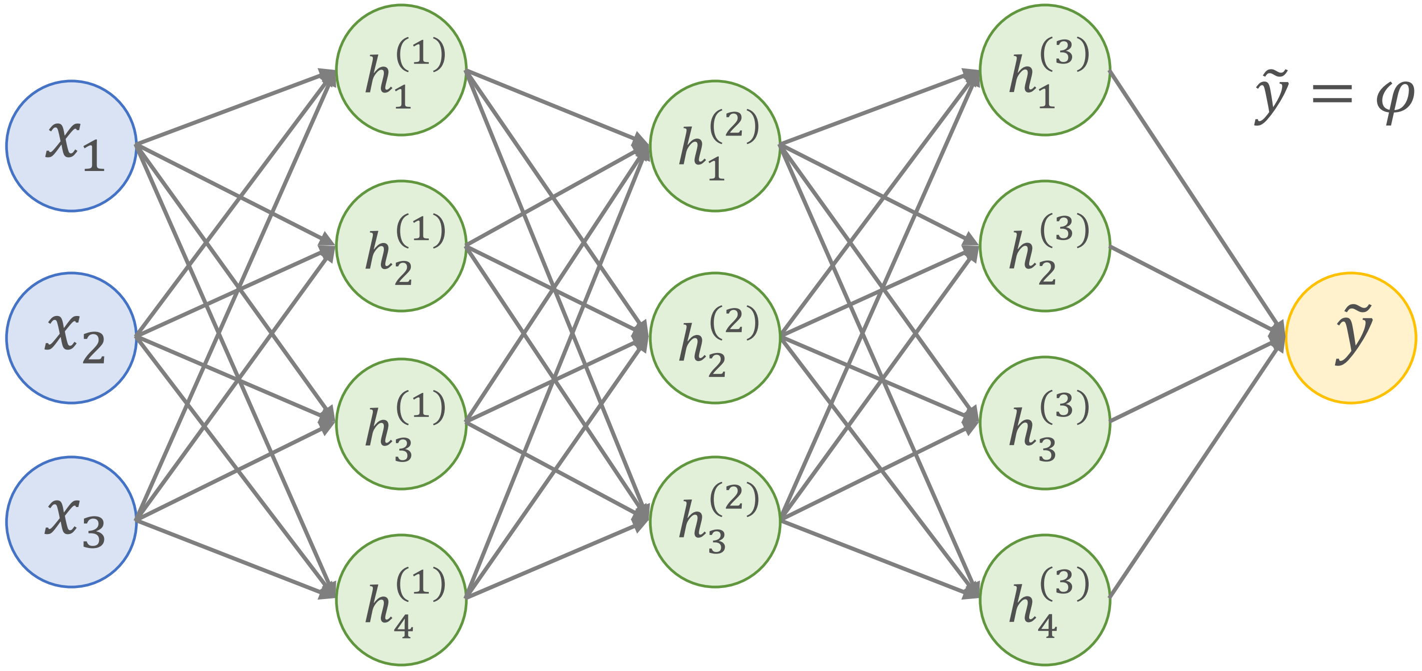
Cross Entropy Loss

$$\mathcal{L}_{total} = -\frac{1}{m}\sum_i \mathcal{L}_{CE}(y_i, \tilde{y}_i)$$

Parameters $\theta = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{W}^{(3)}, \mathbf{W}^{(o)}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \mathbf{b}^{(3)}, \mathbf{b}^{(o)}\},$

$$\theta^* = \arg\min_\theta \mathcal{L}_{total}$$

Back-Propagation



$$\tilde{y} = \varphi\left(\mathbf{W}^{(o)}\mathbf{h}^{(3)} + \mathbf{b}^{(o)}\right)$$

$$\frac{\partial \mathcal{L}}{\partial \tilde{y}}$$

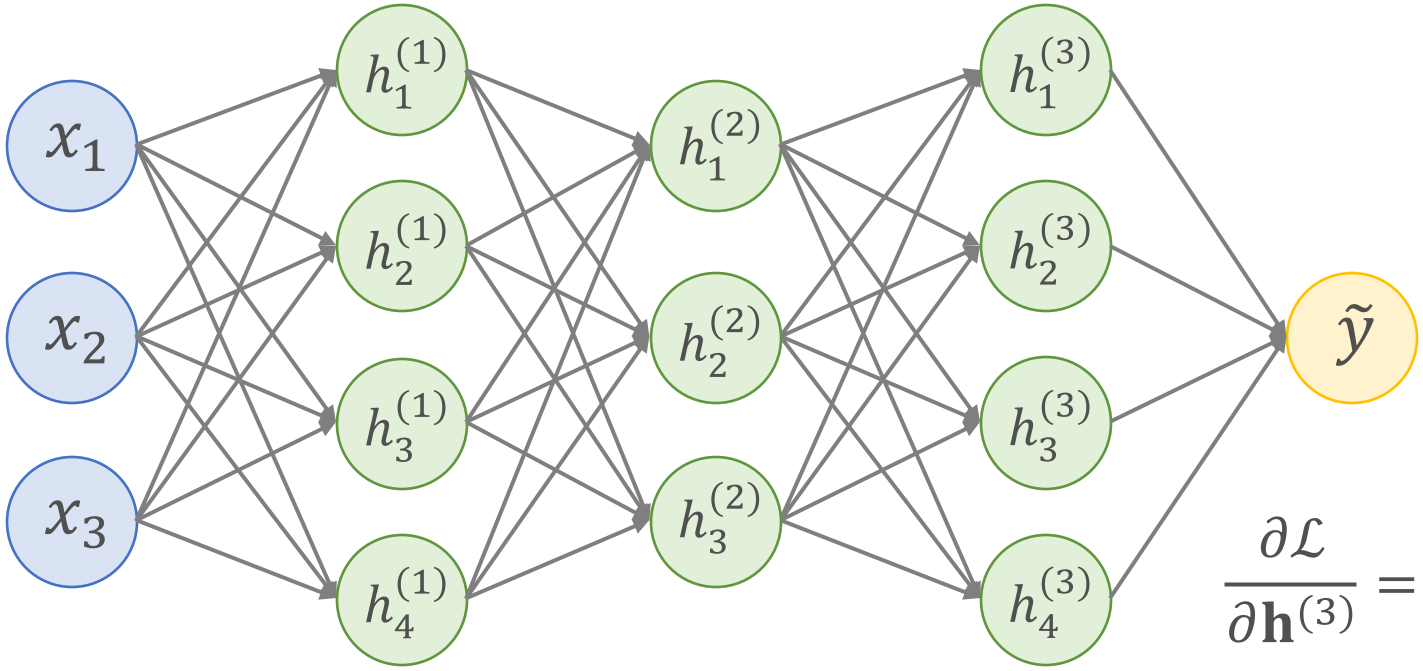
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(o)}} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial \mathbf{W}^{(o)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial \mathbf{h}^{(3)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(o)}} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial \mathbf{b}^{(o)}}$$

Back-Propagation

$$\mathbf{h}^{(3)} = \varphi\left(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)}\right)$$



$$\frac{\partial \mathcal{L}}{\partial \tilde{y}}$$

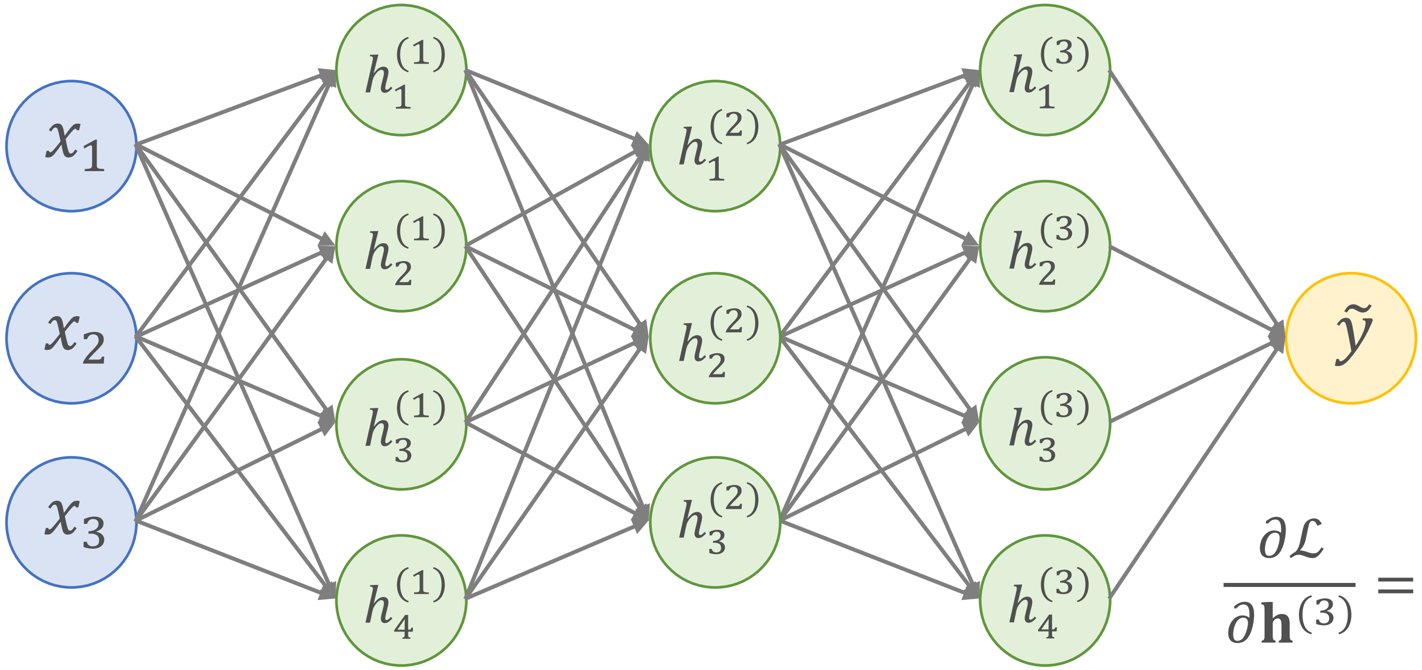
$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial \mathbf{h}^{(3)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(3)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} \cdot \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{W}^{(3)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(2)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} \cdot \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(3)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} \cdot \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{b}^{(3)}}$$

Back-Propagation



$$\frac{\partial \mathcal{L}}{\partial \tilde{y}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial \mathbf{h}^{(3)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(2)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} \cdot \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}}$$

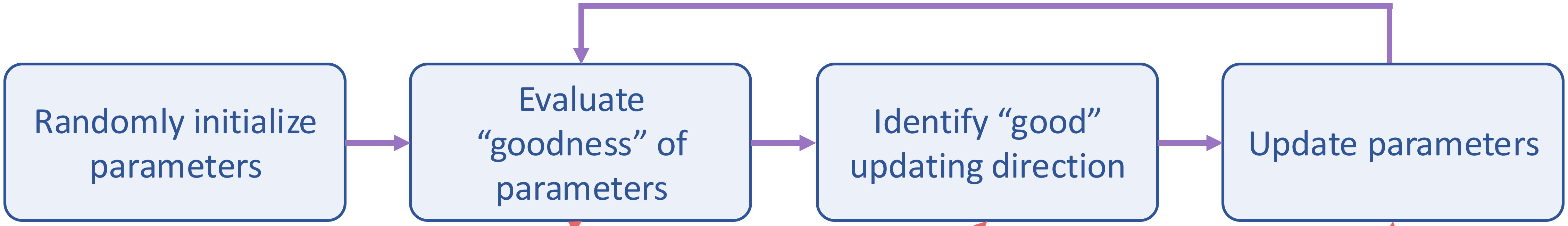
$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(1)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(2)}} \cdot \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(1)}} \cdot \frac{\partial \mathbf{h}^{(1)}}{\partial \mathbf{W}^{(1)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(1)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(1)}} \cdot \frac{\partial \mathbf{h}^{(1)}}{\partial \mathbf{b}^{(1)}}$$

Training Process

Iterative Optimization Methods



Cross Entropy Loss

$$\mathcal{L}_{total} = -\frac{1}{m}\sum_i \mathcal{L}_{CE}\left(y_i, \tilde{y}_i; \mathbf{w}^{(t)}, b^{(t)}\right)$$

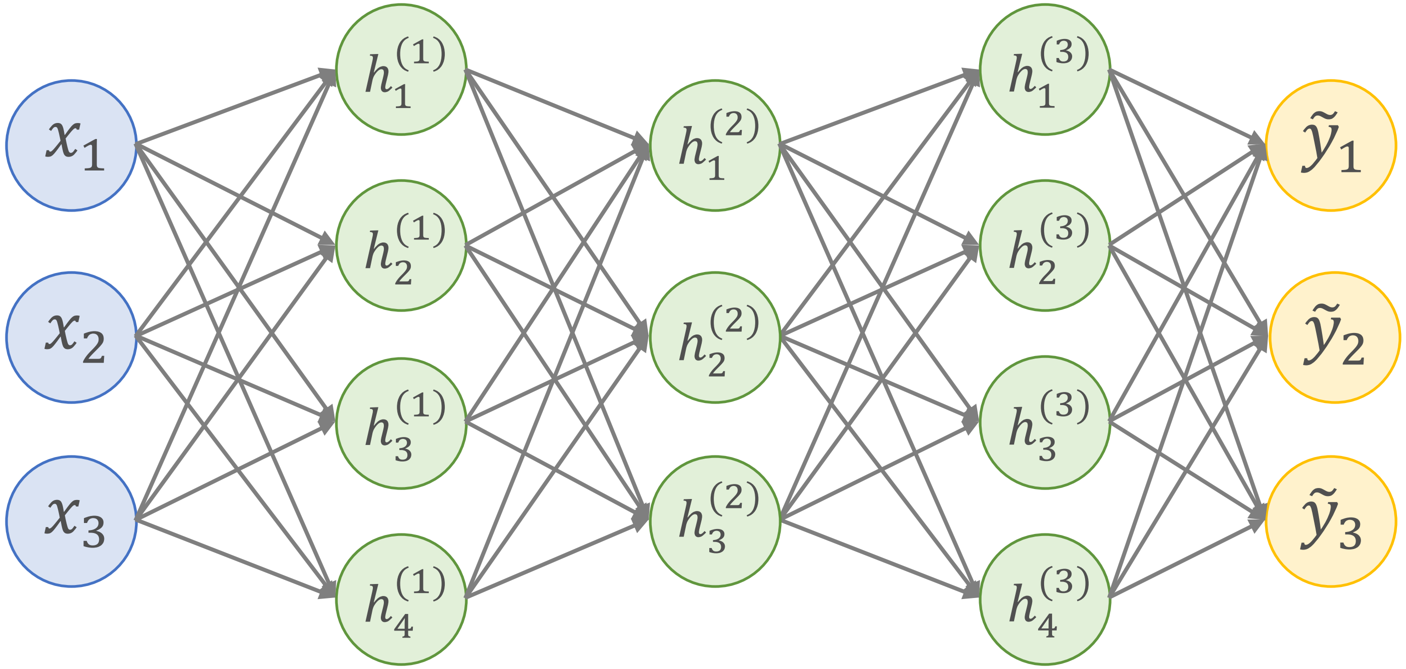
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}}, \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(2)}}, \dots, \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(o)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(1)}}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(2)}}, \dots, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(o)}}$$

$$\mathbf{W}^{(1)} \leftarrow \mathbf{W}^{(1)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}}$$

$$\mathbf{b}^{(1)} \leftarrow \mathbf{b}^{(1)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(1)}}$$

From Binary to Multiclass Classification

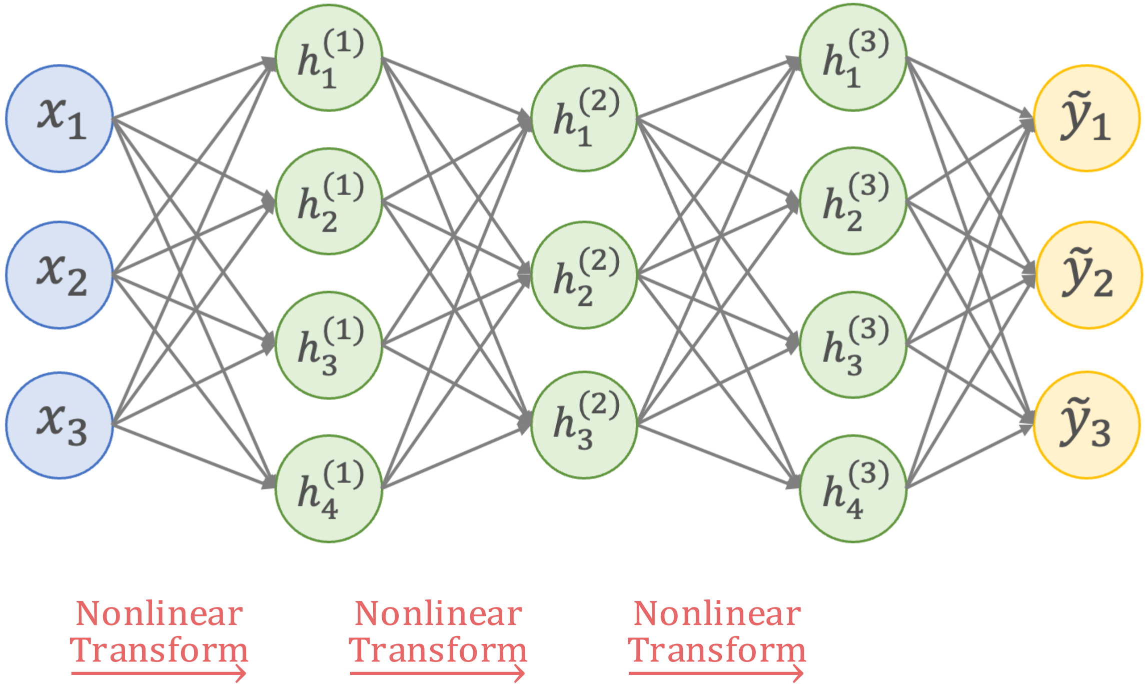


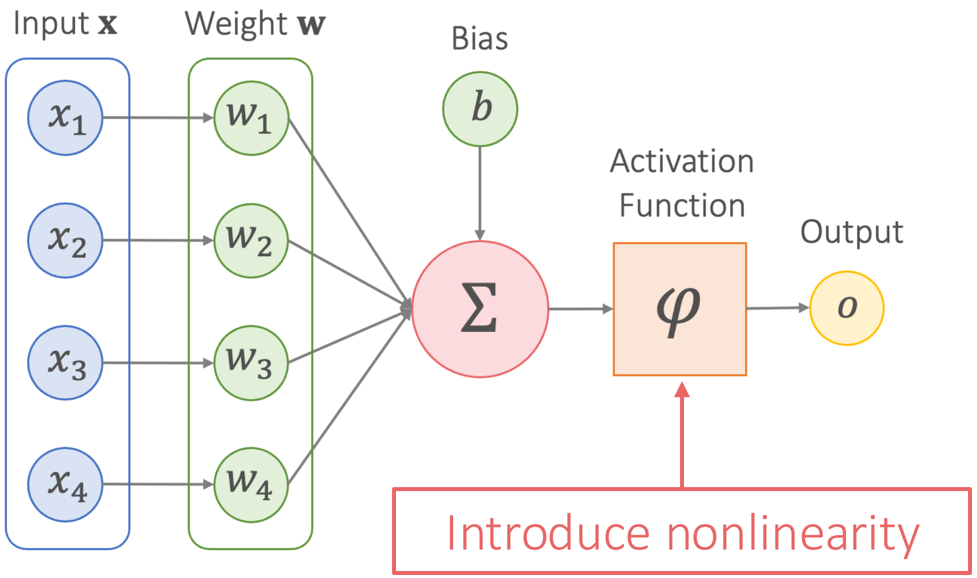
$$\text{Prediction} = \arg\max_{\mathbf{c}} \tilde{y}_c$$

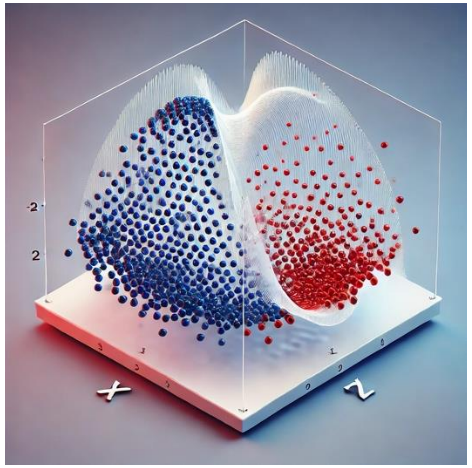
Multiclass Cross Entropy Loss

$$\mathcal{L}_{CE}(y, \tilde{y}) = -\sum_{c=0}^{C} y_c \log P(y = c|\, \mathbf{x})$$

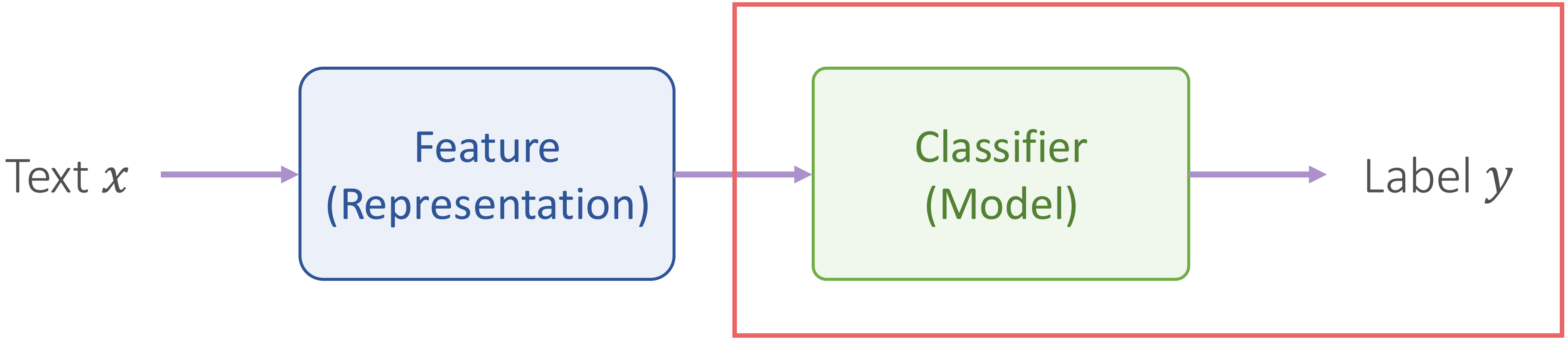
What Makes Neural Networks Powerful?





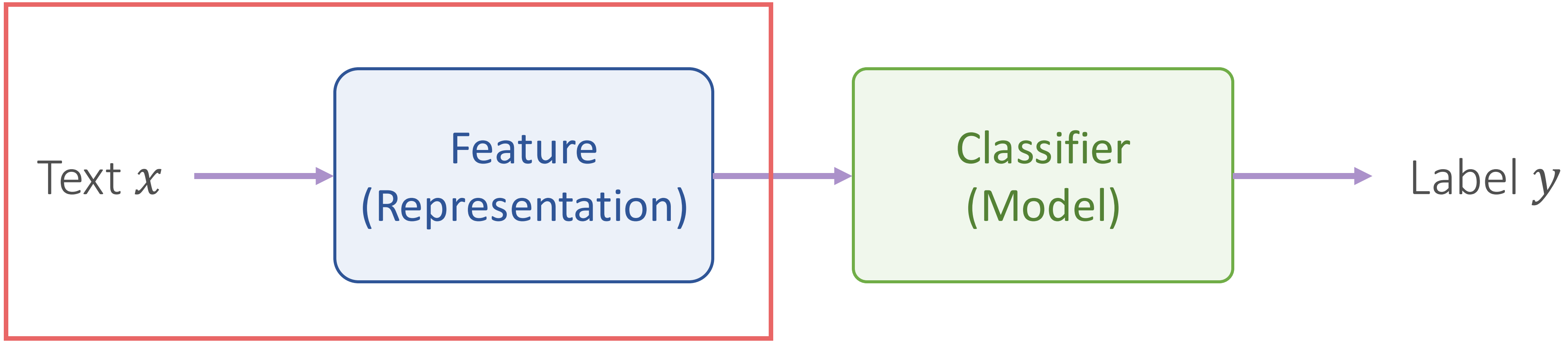


Neural Networks



- Neural Networks
  - Find a non-linear decision boundary to map feature vector $\mathbf{x}$ to label y

Recap: Bag-of-Words and N-Grams



- Teach the model how to understand example x
- Convert the text to a mathematical form
  - The mathematical form captures essential characteristics of the text
- Bag-of-words and n-grams

We will discuss "learnable" features today!

Bag-of-Words and N-Gram Features

Bob likes Alice very much

$\mathbf{x} = [0\ \ 1\ \ \ldots\ \ 0\ \ 1\ \ 1\ \ 0\ \ \ldots\ \ 0\ \ 1]$

Alice likes Bob very much

$\mathbf{x} = [0\ \ 1\ \ \ldots\ \ 0\ \ 0\ \ 0\ \ 1\ \ \ldots\ \ 1\ \ 1]$

BoW (unigram) features

Bigram features

We can consider trigrams, 4-grams, …

Encode a text to ***one vector***

Words as Vectors

Bob *likes* *Alice* *very* *much*

$$W = \begin{bmatrix} | & | & | & | & | \\ w_{bob} & w_{likes} & w_{Alice} & w_{very} & w_{much} \\ | & | & | & | & | \end{bmatrix}$$

Use ***one vector*** to represent ***each word***

Text = A list of vectors

Advantages?

Representing Words by Their Contexts

Distributional hypothesis: words that occur in similar contexts tend to have similar meanings

J.R.Firth 1957

- "You shall know a word by the company it keeps"
- One of the most successful ideas of modern statistical NLP!

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent banking

Distributional Hypothesis

C1: A bottle of ___ is on the table.

C2: Everybody likes ___.

C3: Don’t have ___ before you drive.

C4: I bought ___ yesterday.

	C1	C2	C3	C4
juice	1	1	0	1
loud	0	0	0	0
motor-oil	1	0	0	1
chips	0	1	0	1
choices	0	1	0	0
wine	1	1	1	1

Words that occur in similar contexts tend to have similar meanings

Word Vectors from Word-Word Co-Occurrence Matrix

- Main idea: Similar contexts → Similar word co-occurrence
- Collect a bunch of texts and compute co-occurrence matrix
- Words can be represented by row vectors

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$$

Word Vector

High cosine similarity!

	shark	computer	data	eat	result	sugar
apple	0	0	0	8	0	2
bread	0	0	0	9	0	1
digital	0	6	5	0	2	0
information	0	4	10	0	2	0

Low cosine similarity!

Most entries are 0s → sparse vectors

Issues with Word-Word Co-Occurrence Matrix

- Using raw frequency counts is not always very good (why?)
  - Some frequent words (e.g., the, it, or they) can have large counts

	shark	computer	data	eat	result	sugar	the	it
apple	0	0	0	8	0	2	104	67
bread	0	0	0	9	0	1	95	76
digital	0	6	5	0	2	0	101	65

Similarity(apple, bread) ≈ 0.994710

Similarity(apple, digital) ≈ 0.995545

Similarity is dominated by frequent words

Solution: use a ***weighted function*** instead of raw counts

Pointwise Mutual Information

Pointwise Mutual Information (PMI)

Do events x and y co-occur more or less than if they were independent?

$$\mathrm{PMI}(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

- PMI = 0 → x and y occur independently → co-occurrence is as expected
- PMI > 0 → x and y co-occur more often than expected
- PMI < 0 → x and y co-occur less often than expected

Co-Occurrence Matrix with Positive PMI

Positive Pointwise Mutual Information (PPMI)

$$\text{PPMI}(x, y) = \max\left(\log_2 \frac{P(x, y)}{P(x)P(y)}, 0\right)$$

	shark	computer	data	eat	result	sugar	the	it
apple	0	0	0	1.80	0	0.35	0.08	0
bread	0	0	0	1.54	0	0.29	0	0.14
digital	0	1.47	1.22	0	0.61	0	0.10	0.06

Similarity(apple, bread) ≈ 0.995069

Similarity(apple, digital) ≈ 0.010795

Sparse Vectors vs. Dense Vectors

- The vectors in the word-word occurrence matrix are
  - **Long**: vocabulary size
  - **Sparse**: most are 0's
- Can we have short short (50-300 dimensional) and dense (real-valued) vectors?
  - Short vectors are easier to use as features in ML systems
  - Dense vectors may generalize better than explicit counts
  - Sparse vectors can't capture high-order co-occurrence
    - w_1 co-occurs with "car", w_2 co-occurs with "automobile"
    - They should be similar, but they aren't, because "car" and "automobile" are distinct dimensions
  - In practice, they work better!

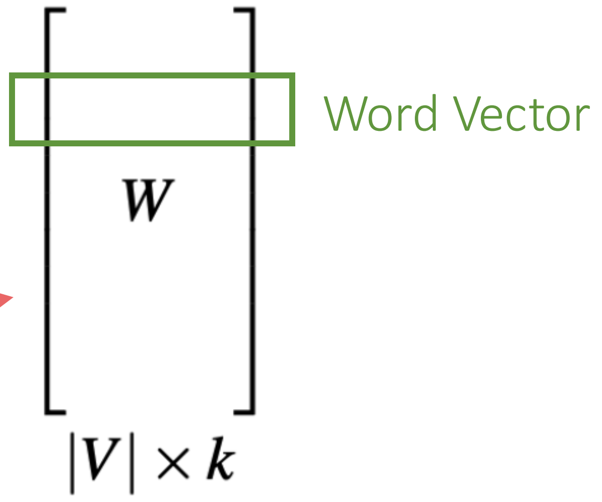
How to Get Dense Vectors?

- Singular value decomposition (SVD) of PPMI weighted co-occurrence matrix

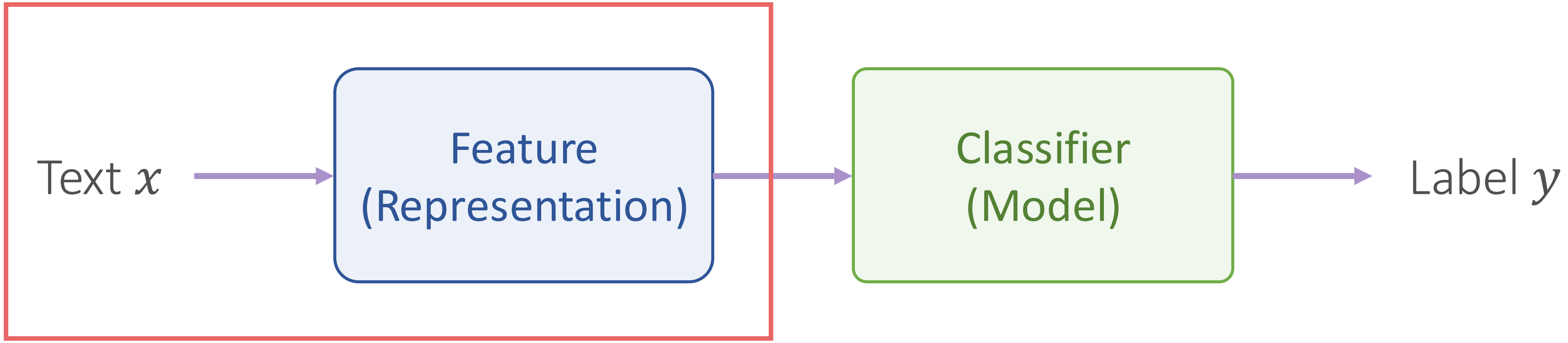
$$\underset{|V|\times|V|}{\left[\; X \;\right]} = \underset{|V|\times|V|}{\left[\; W \;\right]} \underset{|V|\times|V|}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix}} \underset{|V|\times|V|}{\left[\; C \;\right]}$$

Only keep the top k singular values

$$\underset{|V|\times|V|}{\left[\; X \;\right]} = \underset{|V|\times k}{\left[\; W \;\right]} \underset{k\times k}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix}} \underset{k\times|V|}{\left[\; C \;\right]}$$



Count-Based Word Vectors



- Use one vector to represent each word
- Get word vectors by singular value decomposition (SVD) of PPMI weighted co-occurrence matrix