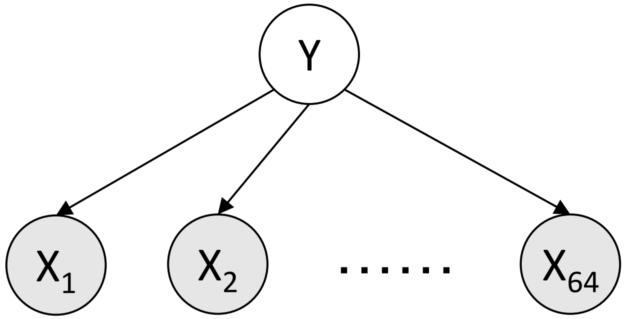
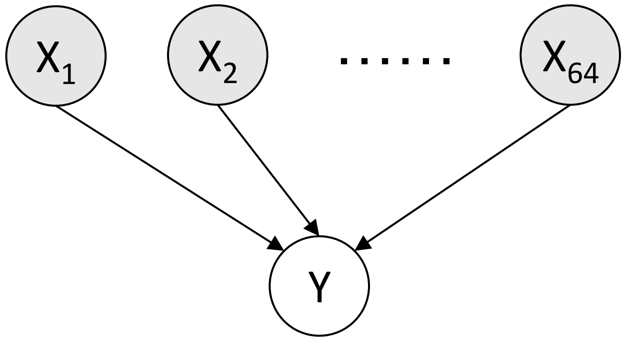
Neural Network

Chen-Yu Wei

Naïve Bayes and Logistic Regression

$Y \in \{0, \dots, 9\}$: class
$X_1, \dots, X_{64}$: features

<table>
<tr><th></th><th>Naïve Bayes</th><th>Logistic Regression</th></tr>
<tr><td>Bayes Net representation</td><td>
</td><td>
</td></tr>
<tr><td>Modeling</td><td>$P(X_1, \dots, X_{64} \mid Y)$ $P(Y)$</td><td>$P(Y \mid X_1, \dots, X_{64})$ $\cancel{P(X_1, \dots, X_{64})}$</td></tr>
<tr><td>Assumption</td><td>$P(X_1, \dots, X_{64} \mid Y)$
$= P(X_1|Y)P(X_2|Y) \dots P(X_{64}|Y)$</td><td>$P(Y \mid X_1, \dots, X_{64})$
$\propto \exp\big(f_w(X, Y)\big) = \exp\big(w^{(Y)} \cdot X\big)$</td></tr>
<tr><td>Type of model</td><td>Generative model</td><td>Discriminative model</td></tr>
</table>

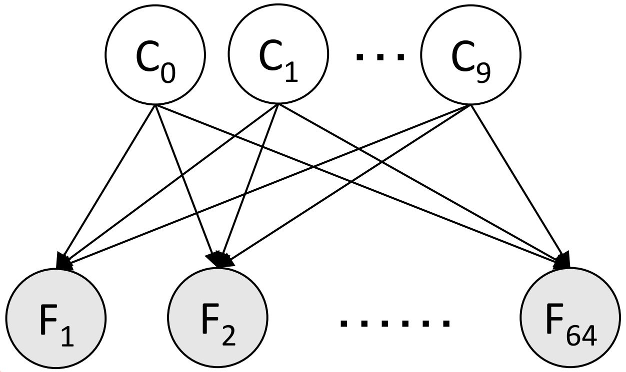
Naïve Bayes and Logistic Regression

W_{ij}: the weight between F_i and C_j

"Neural Net" representation

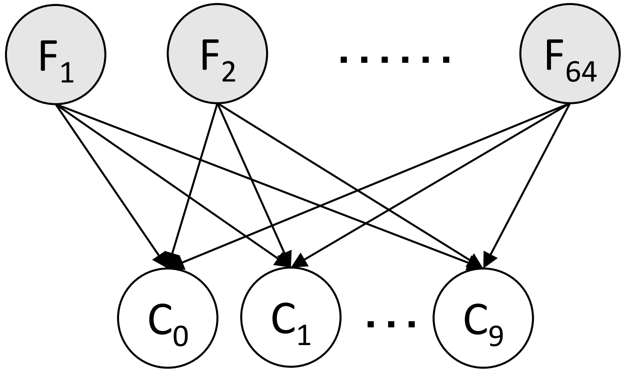
$W_{ij} = p(X_i = 1 \mid Y = j)$

Naïve Bayes



$$F_i = \sum_{j=0}^{9} W_{ij} C_j$$

Logistic Regression

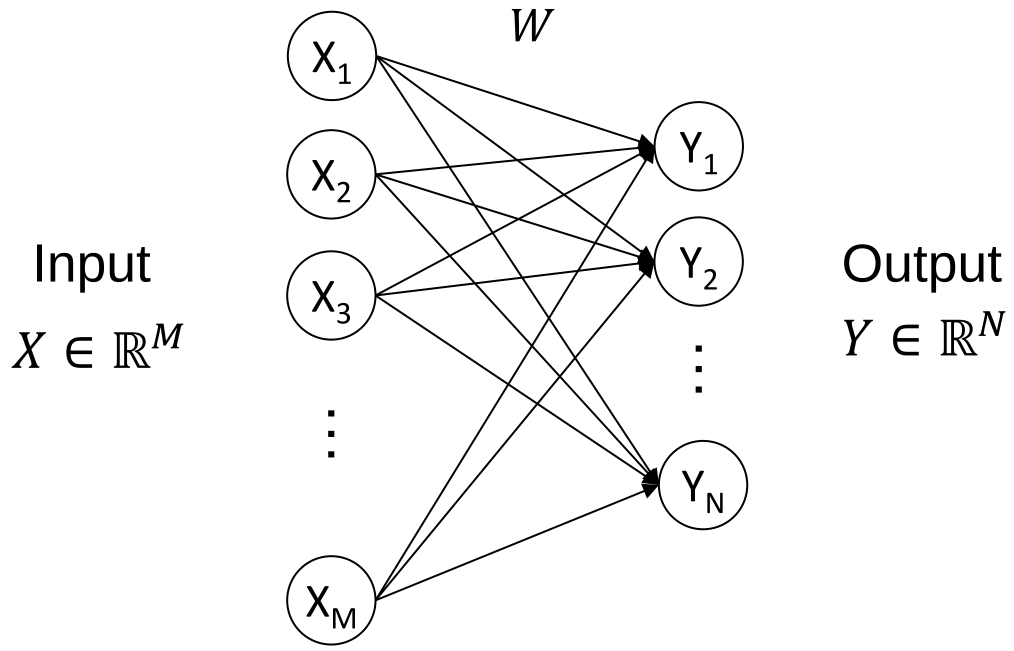


$$C_j = \sum_{i=1}^{64} W_{ij} F_i$$

	Naïve Bayes	Logistic Regression
The meaning of C_j	Class $= j \iff (C_0, \dots, C_9) = (0, \dots, 1, \dots, 0)$	The score between class j and the input features
The meaning of F_i	The expected value of i-th feature given the input class	The i-th feature

Neural network (NN)

A general tool to model the relation between two real-valued vectors



This neural network describes the relation

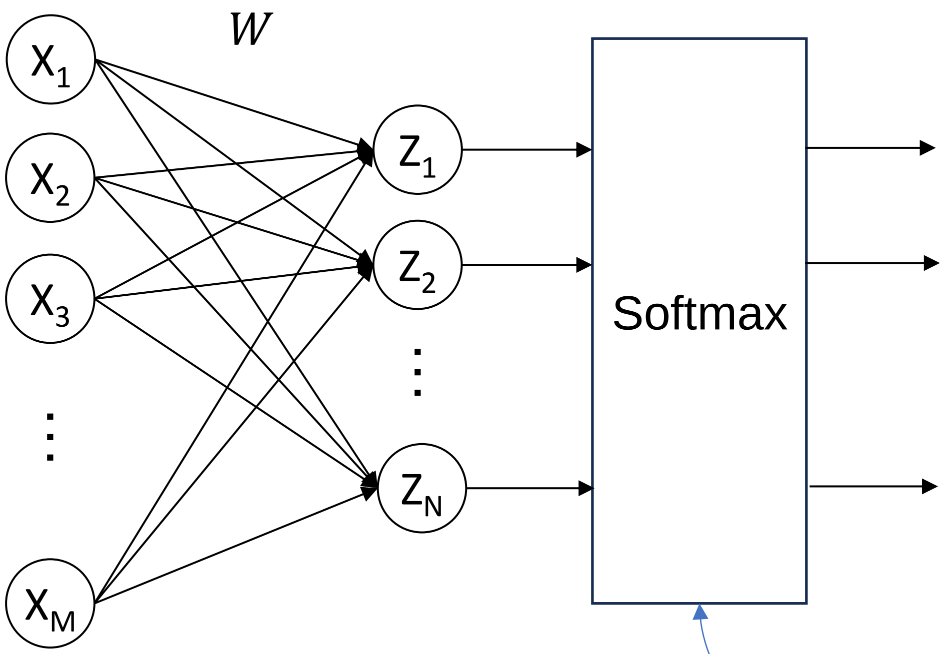
$$Y_i = \sum_{j=1}^{M} W_{ij} X_j \qquad \forall i = 1, \dots, N$$

or, more succinctly, $Y = WX$

X	Y	
Pixel values	Scores	(LR)
Digit label in one-hot representation	Expected pixel value (if pixels value ∈ {0,1})	(NB)
Digit label in one-hot representation	Pixel value (if pixels value ∈ [0,1])	
Spam/ham in one-hot representation	Word frequency	(NB)

X, Y here are general vectors and do not need to correspond to feature and label

Logistic Regression (1-Layer NN for Classification)



$$\frac{\exp(Z_1)}{\sum_{j=1}^{N} \exp(Z_j)} = P_W(Y = 1 \mid X)$$

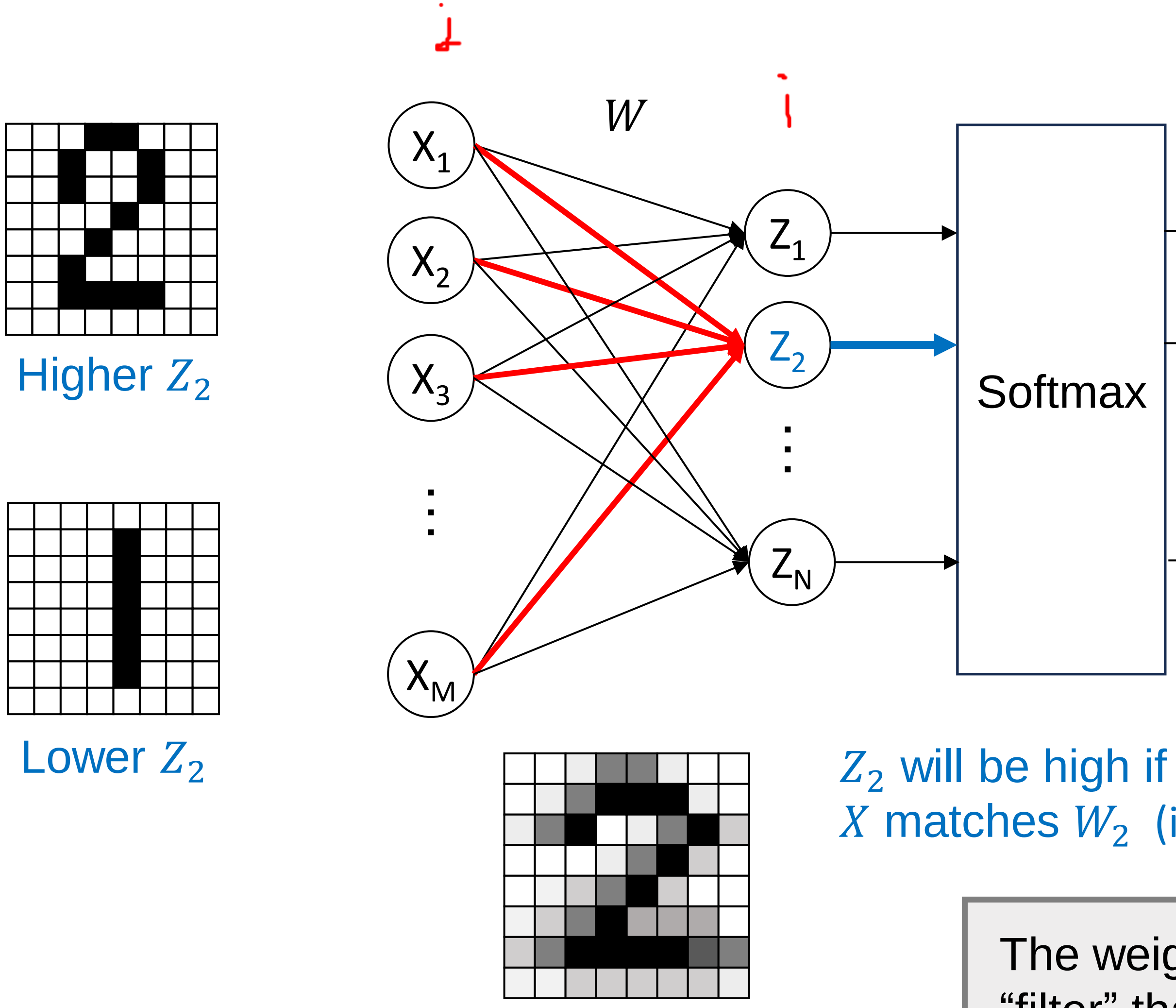
$$\frac{\exp(Z_2)}{\sum_{j=1}^{N} \exp(Z_j)} = P_W(Y = 2 \mid X)$$

$$\frac{\exp(Z_N)}{\sum_{j=1}^{N} \exp(Z_j)} = P_W(Y = N \mid X)$$

Additional operation to fulfill the restriction on the final output (e.g., here we want the output to be a distribution)

Find W that minimizes $\sum_{s=1}^{|S|} -\log P_W(y_s \mid x_s)$ using Stochastic Gradient Descent

Logistic Regression (1-Layer NN for Classification)



$$\frac{\exp(Z_1)}{\sum_{j=1}^{N}\exp(Z_j)} = P_W(Y = 1 \mid X)$$

$$\frac{\exp(Z_2)}{\sum_{j=1}^{N}\exp(Z_j)} = P_W(Y = 2 \mid X)$$

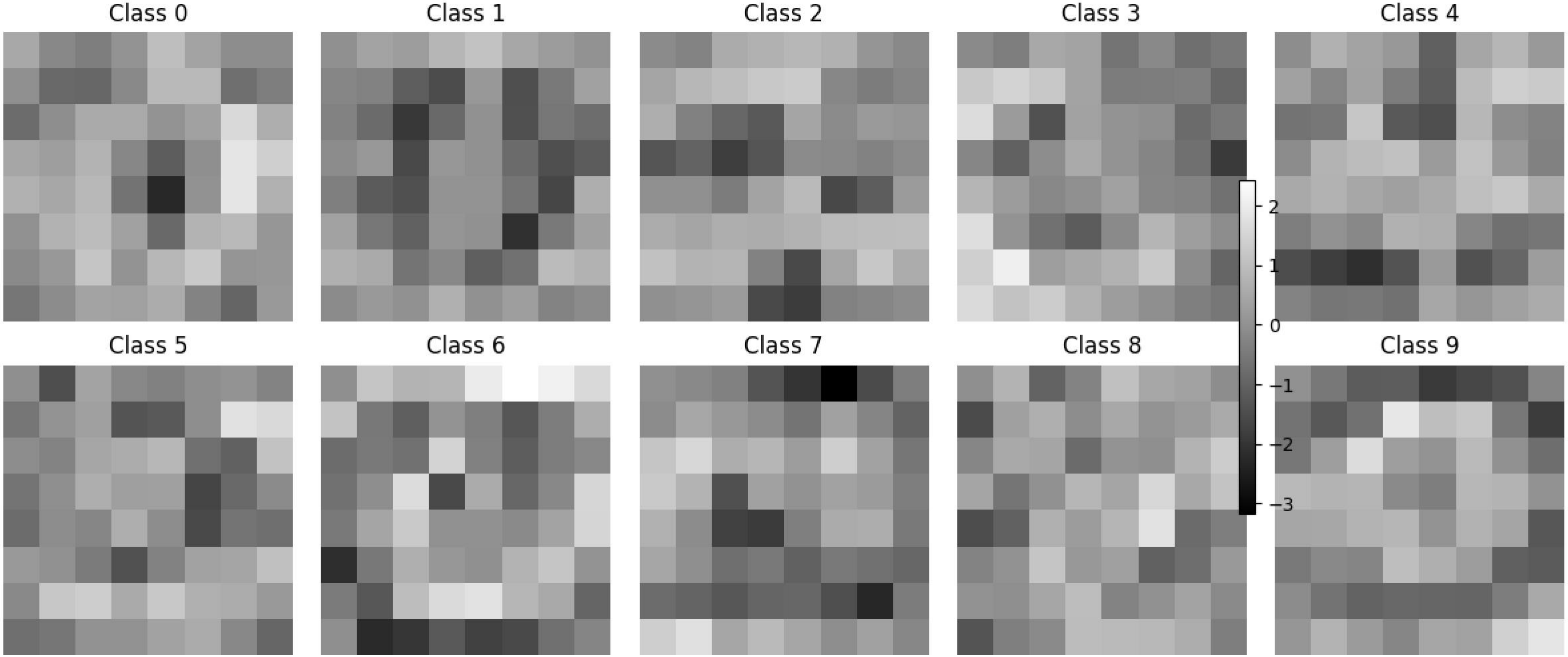
$$\frac{\exp(Z_N)}{\sum_{j=1}^{N}\exp(Z_j)} = P_W(Y = N \mid X)$$

Z_2 will be high if the input pattern X matches W_2 (i.e., $X \cdot W_2$ is large)

$$W_2 = (W_{2,1}, W_{2,2,\dots,}, W_{2,64})$$

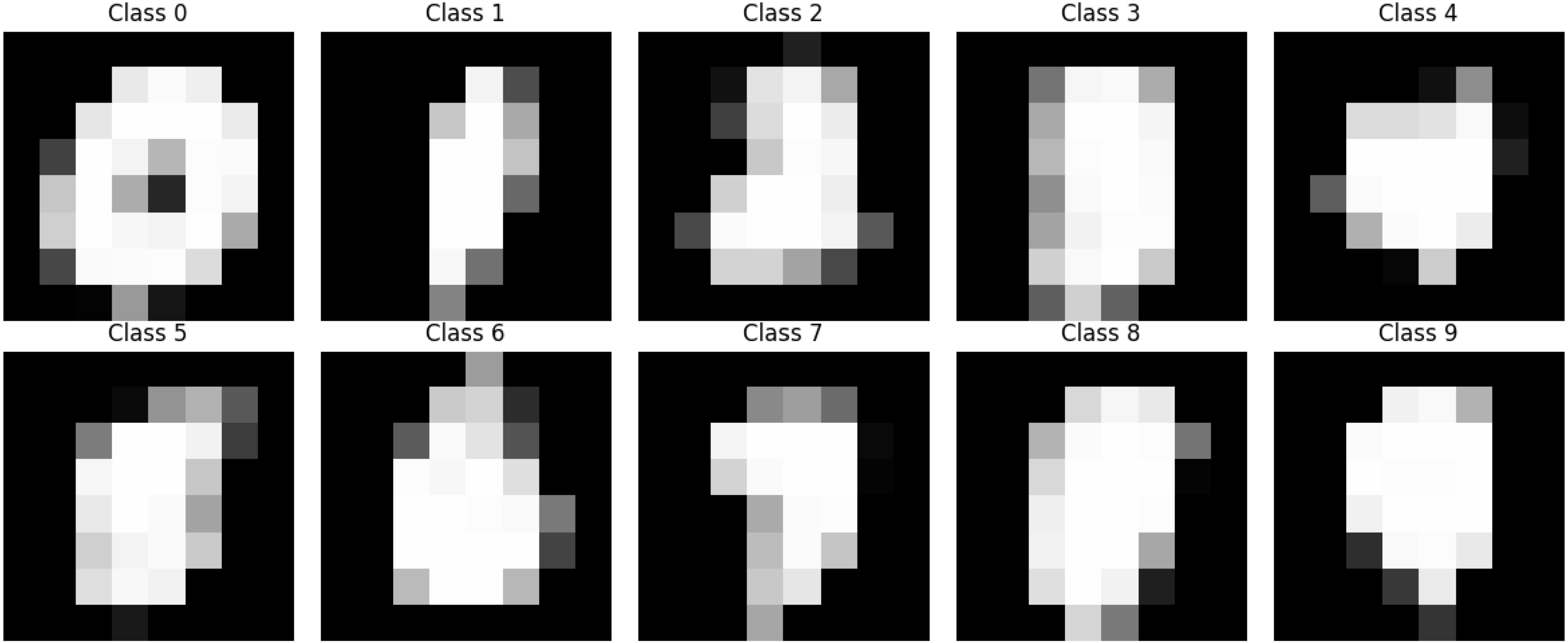
The weight associated with an output node acts like a "filter" that recognizes a particular pattern on the input.

The Weights Produced by Logistic Regression



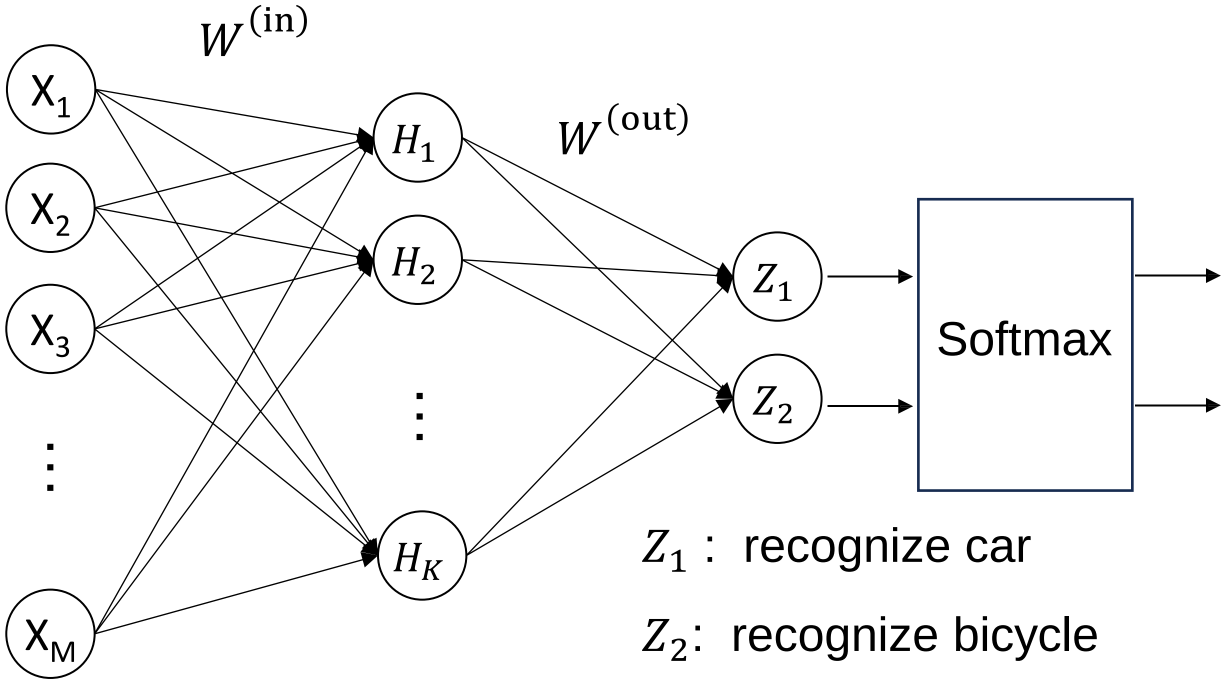
Classification (testing) accuracy: 78.25%

The Weights Produced by Naïve Bayes



Classification (testing) accuracy: 67.30%

2-Layer NN for Classification



Z_1 : recognize car

Z_2: recognize bicycle

H_1 : recognize wheels

H_2 : recognize windows

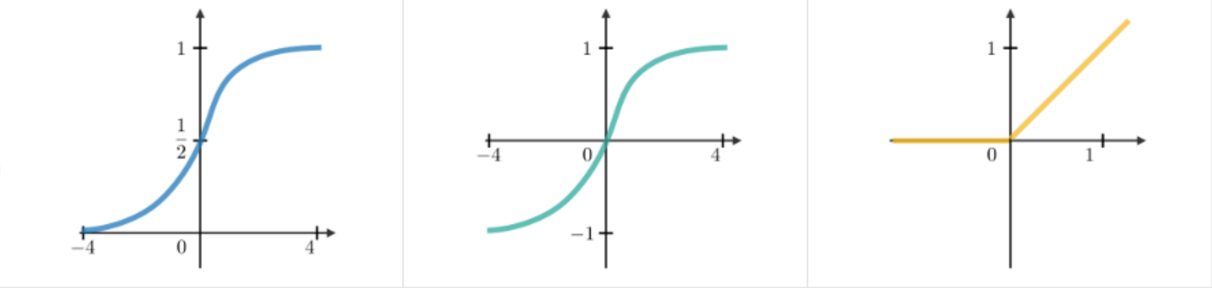
H_3 : recognize handlebar

$$H_i = g\left(\sum_j W_{ij}^{(\text{in})} X_j\right) \qquad H = g(W^{(\text{in})} X)$$

g = nonlinear **activation function**

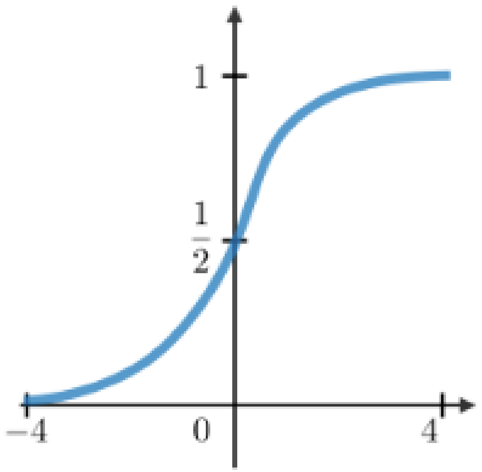
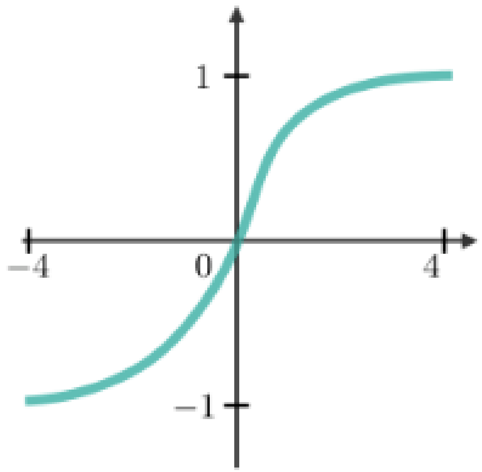
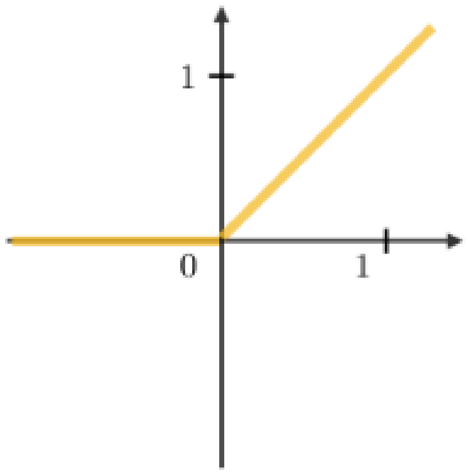
$$Z_i = \sum_j W_{ij}^{(\text{out})} H_j \qquad Z = W^{(\text{out})} H$$

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$

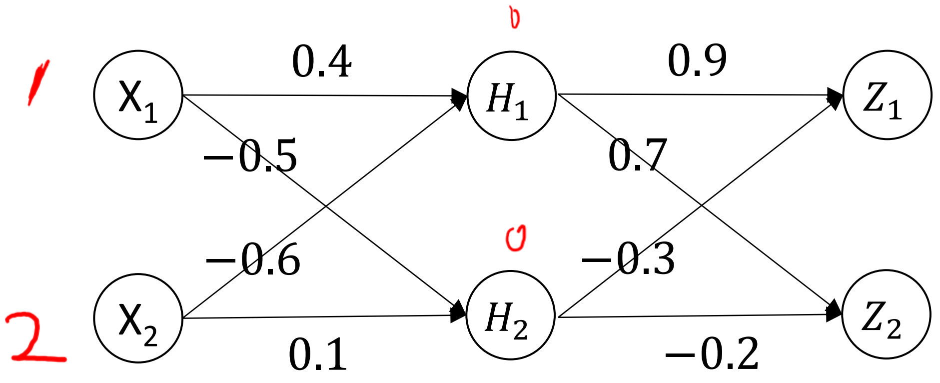


Activation Functions

Rectified Linear Unit

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

Exercise: 2-Layer NN with Activation Function



If we use the ReLU activation function

What's Z given input X = (1,2)?

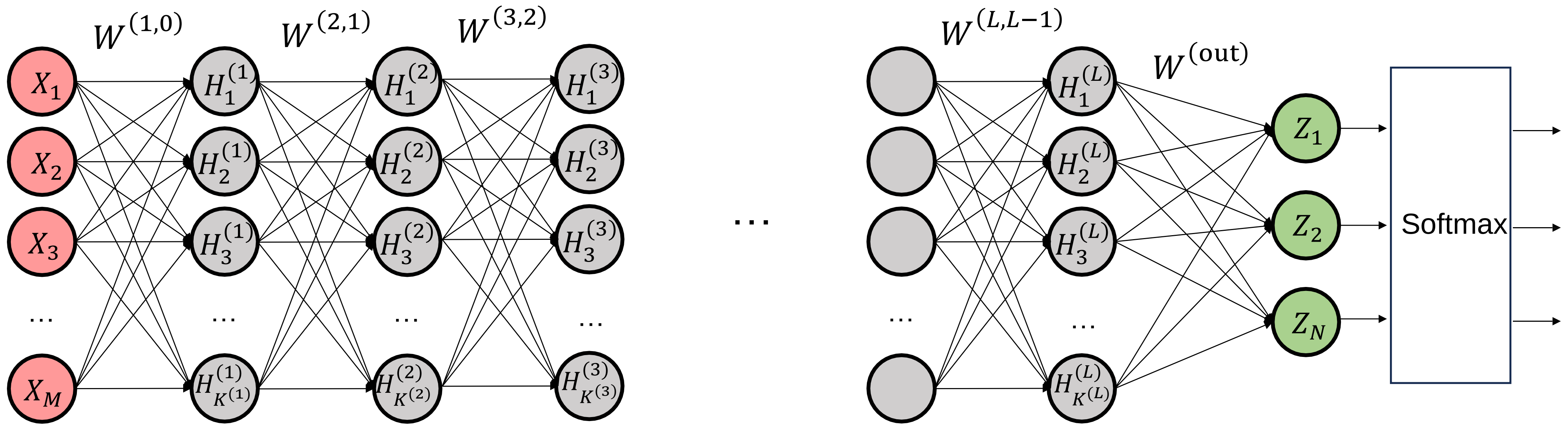
$H_1 = ReLU(0.4 - 1.2) = 0$

$H_2 = ReLU(-0.5 + 0.2) = 0$

$Z_1 = ReLU(\cdots) = 0$

$Z_2 = ReLU(\cdots) = 0$

Multi-Layer NN for Classification

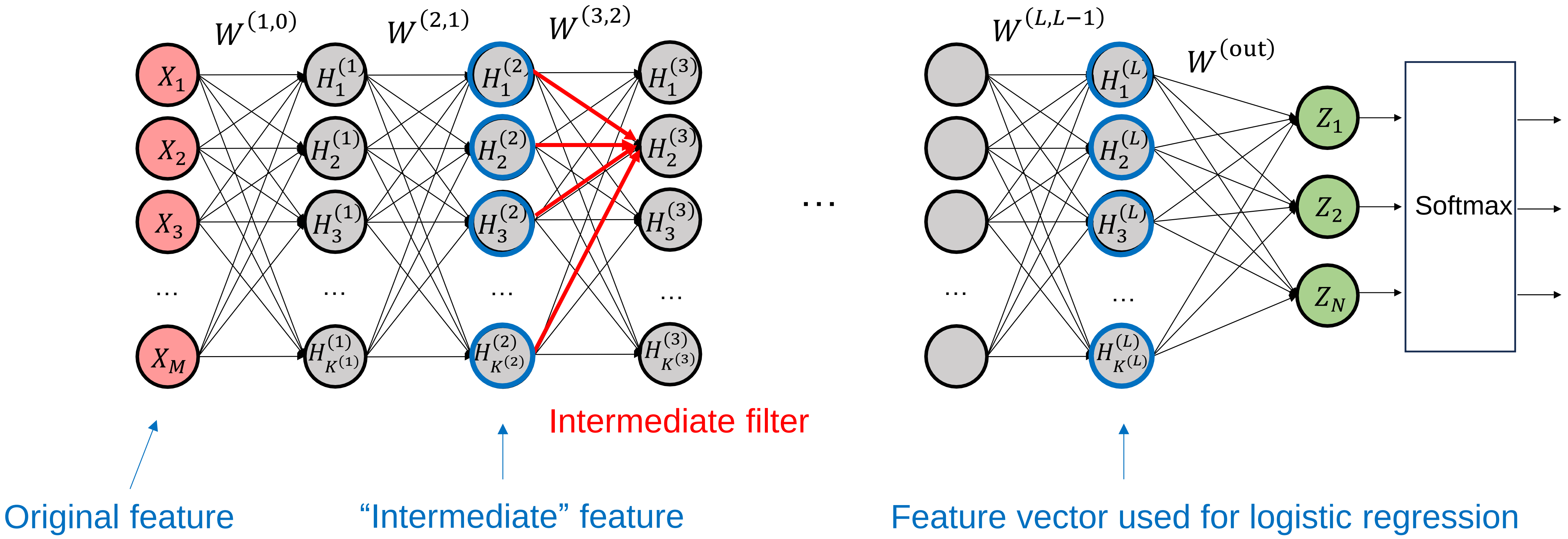


$$H_i^{(0)} \coloneqq X_i \qquad H^{(0)} \coloneqq X$$

$$H_i^{(\ell)} = g\left(\sum_j W_{ij}^{(\ell,\ell-1)} H_j^{(\ell-1)}\right) \quad \forall \ell = 1, \dots, L \qquad H^{(\ell)} = g(W^{(\ell,\ell-1)} H^{(\ell-1)})$$

$$Z_i = \sum_j W_{ij}^{(\mathrm{out})} H_j^{(L)} \qquad Z = W^{(\mathrm{out})} H^{(L)}$$

Multi-Layer NN for Classification



Multi-layer neural network enables successive feature transformations (e.g., from low-level feature to high-level feature)

These transformations (through W) is learned automatically from data

→ **Representation learning**

Training Multi-Layer Neural Network

$$P_W(y_s|x_s) = \left. \frac{\exp(Z_{y_s})}{\sum_y \exp(Z_y)} \right|_{\text{input}=x_i} = \frac{\exp(f_W(x_s,y_s))}{\sum_y \exp(f_W(x_s,y))}$$

[0 0, ... 1, 0 0 0]

$W^{(y)} \cdot x = e_y^T W x$

We can expand $f_W(x,y)$ as

$$f_W(x,y) = e_y^\top W^{(\text{out})} H^{(L)}$$

$$= e_y^\top W^{(\text{out})} \boxed{g\left(W^{(L,L-1)} H^{(L-1)}\right)}$$

$$= e_y^\top W^{(\text{out})} g\left(W^{(L,L-1)} g\left(W^{(L-1,L-2)} H^{(L-2)}\right)\right)$$

$$= \underbrace{e_y^\top W^{(\text{out})} g\left(W^{(L,L-1)} g\left(W^{(L-1,L-2)} g\left(\dots g\left(W^{(1,0)} x\right)\right)\right)\right)}$$

A quite complicated **non-linear** function of $W = (W^{(\text{out})}, W^{(L,L-1)}, \dots, W^{(1,0)})$

Nevertheless, we use the same idea (Maximum Likelihood + Stochastic Gradient Descent) to find a good W

Training Multi-Layer Neural Network

$$P_W(y_s|x_s) = \left.\frac{\exp(Z_{y_s})}{\sum_y \exp(Z_y)}\right|_{\text{input}=x_i} = \frac{\exp(f_W(x_s, y_s))}{\sum_y \exp(f_W(x_s, y))}$$

$$f_W(x_s, y_s) = e_{y_s}^{\top} W^{(\text{out})} g\left(W^{(L,L-1)} g\left(W^{(L-1,L-2)} g\left(\dots g\left(W^{(1,0)} x_s\right)\right)\right)\right)$$

Parameters of the model

$$\nabla_W L(W) = \begin{bmatrix} \frac{\partial}{\partial W_1} L(W) \\ \frac{\partial}{\partial W_2} L(W) \\ \vdots \end{bmatrix}$$

$$\sum_s -\log P_W(y_s|x_s) = L(W)$$

The process of calculating the gradient in NNs through chain rule is called **Backpropagation.**

Training Multi-Layer Neural Network

- Get dataset consisting of (X, Y) pairs: $(x_1, y_1), (x_2, y_2), \dots, (x_S, y_S) \in \mathbb{R}^d \times \{1, 2, \dots, C\}$

- Define the **objective function / loss function**:

$$\frac{1}{S}\sum_{s=1}^{S} -\log P_W(y_s|x_s) = \frac{1}{S}\sum_{s=1}^{S} \underbrace{-\log\left(\frac{\exp(f_W(x_s, y_s))}{\sum_y \exp(f_W(x_s, y))}\right)}_{L_s(W)}$$

- Use stochastic gradient descent to minimize the loss

For $t = 1, 2, \dots$

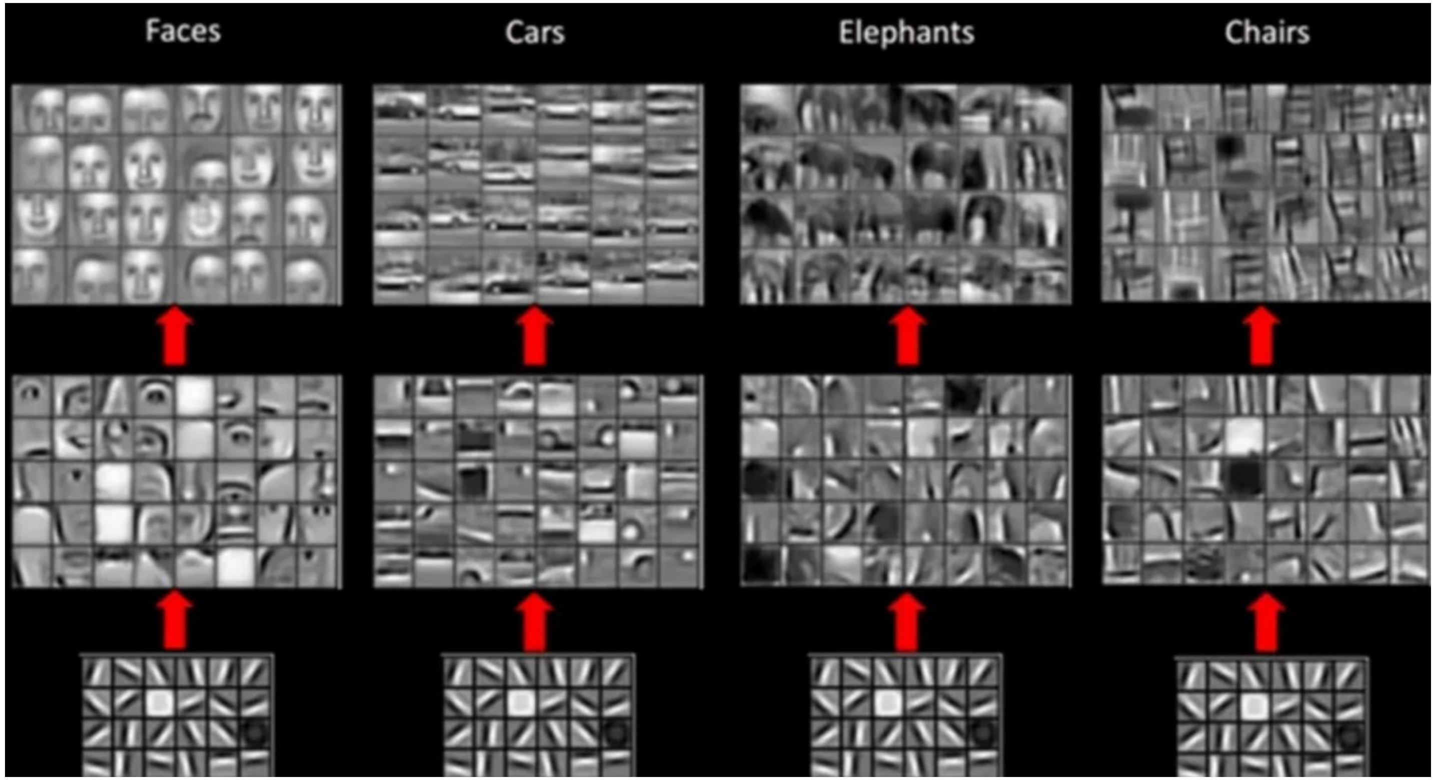
Randomly sample a minibatch $B \subset \{(x_1, y_1), (x_2, y_2), \dots, (x_S, y_S)\}$ of size $|B| = b$

$$W_t = W_{t-1} - \eta \cdot \frac{1}{b} \sum_{(x_s, y_s) \in B} \nabla L_s(W_{t-1})$$

Multi-Layer Pattern Recognition

The machine can automatically discover **useful patterns** through maximum likelihood / loss minimization training.

hidden in W



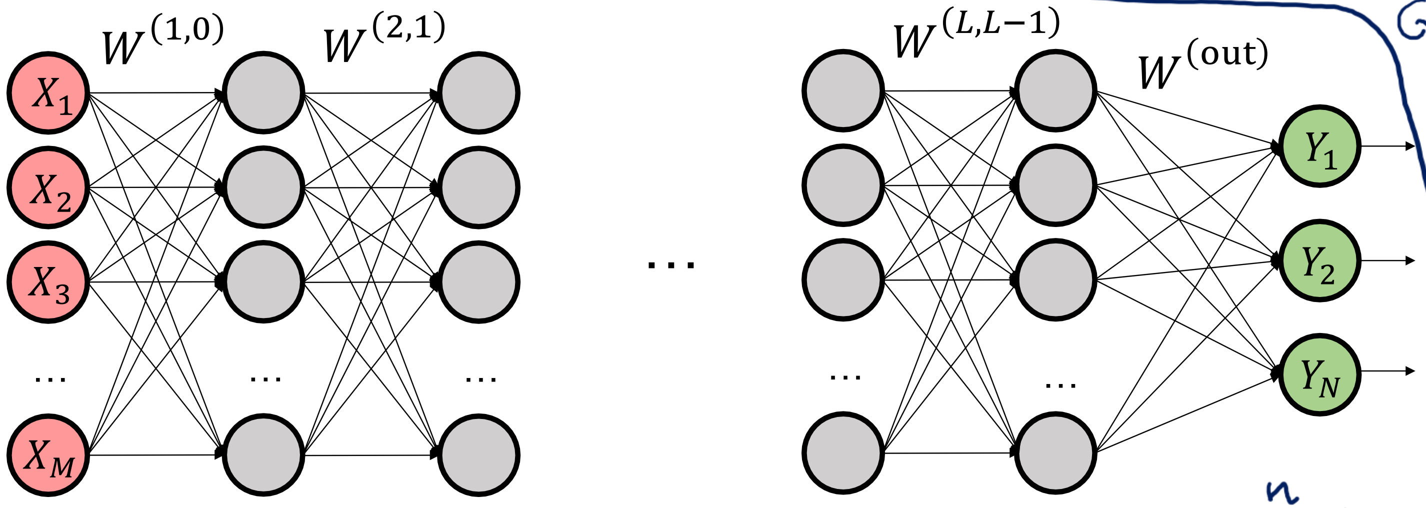
Neural Network for Regression Problems

- Neural network is a general tool to model the relation between two vectors. Besides classification (where output is a distribution over classes), it can also solve **regression** problems (where output is simply a real valued vector).
- For example,

$\{(x_i, y_i)\}$ $\quad x_i \in \mathbb{R}^M, \; y_i \in \mathbb{R}^N$

Goal: find a function such that

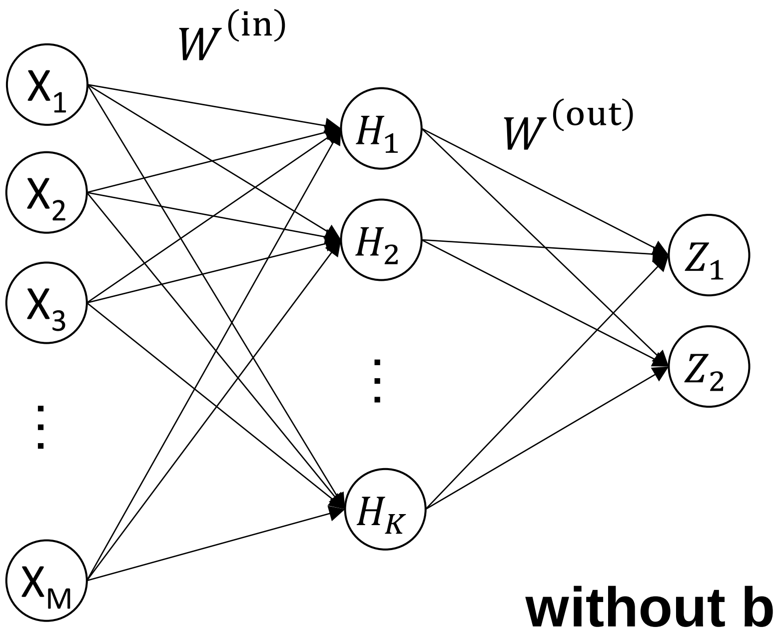
$$y_i \approx f(x_i)$$



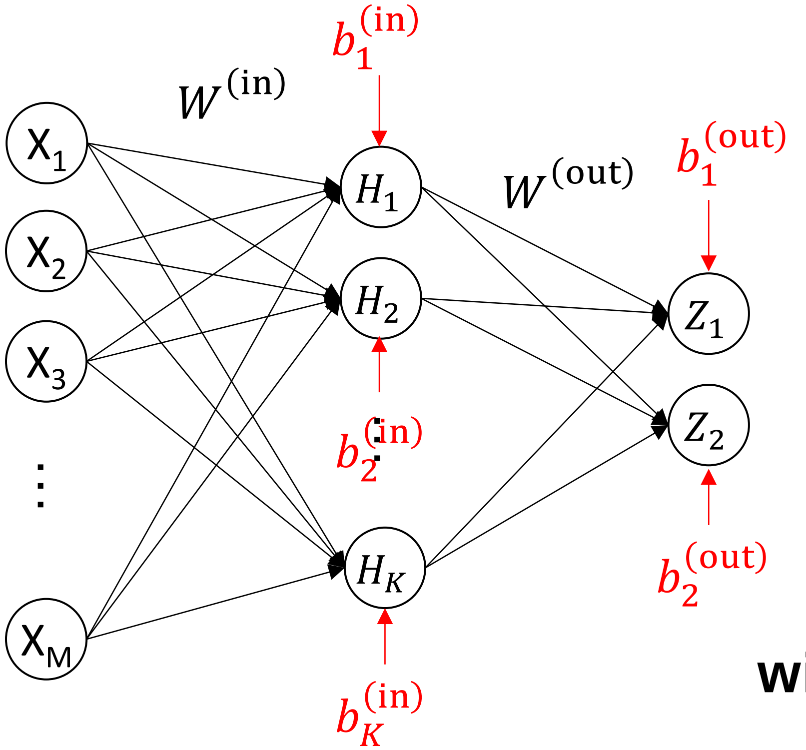
$$L(w) = \frac{1}{n} \sum_{i=1}^{n} \| y_i - f_w(x_i) \|^2$$

Biases

- Besides weights, we usually include **biases**.



without bias



with bias

$$H = g\left(W^{(\text{in})}X\right)$$
$$Z = W^{(\text{in})}H$$

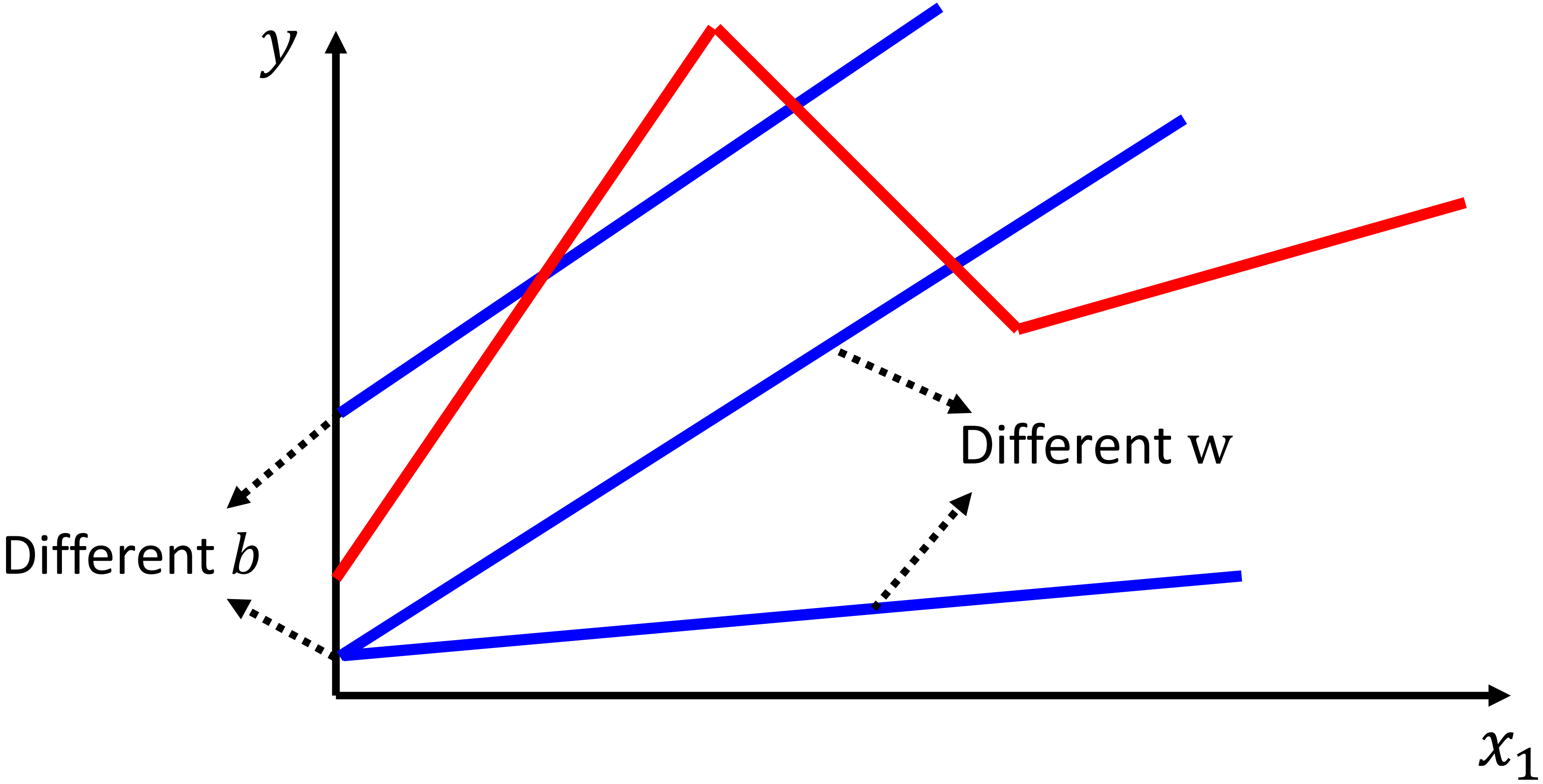
$$H = g\left(W^{(\text{in})}X + b^{(\text{in})}\right)$$
$$Z = W^{(\text{in})}H + b^{(\text{out})}$$

Why Activation Function?

- To make the model more expressive

Explanation by prof. Hung-Yi Lee

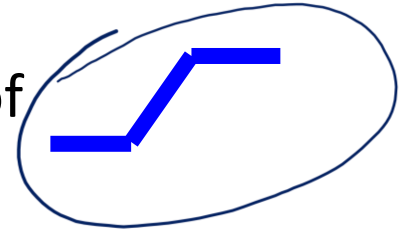
Linear models are too simple … we need more sophisticated modes.

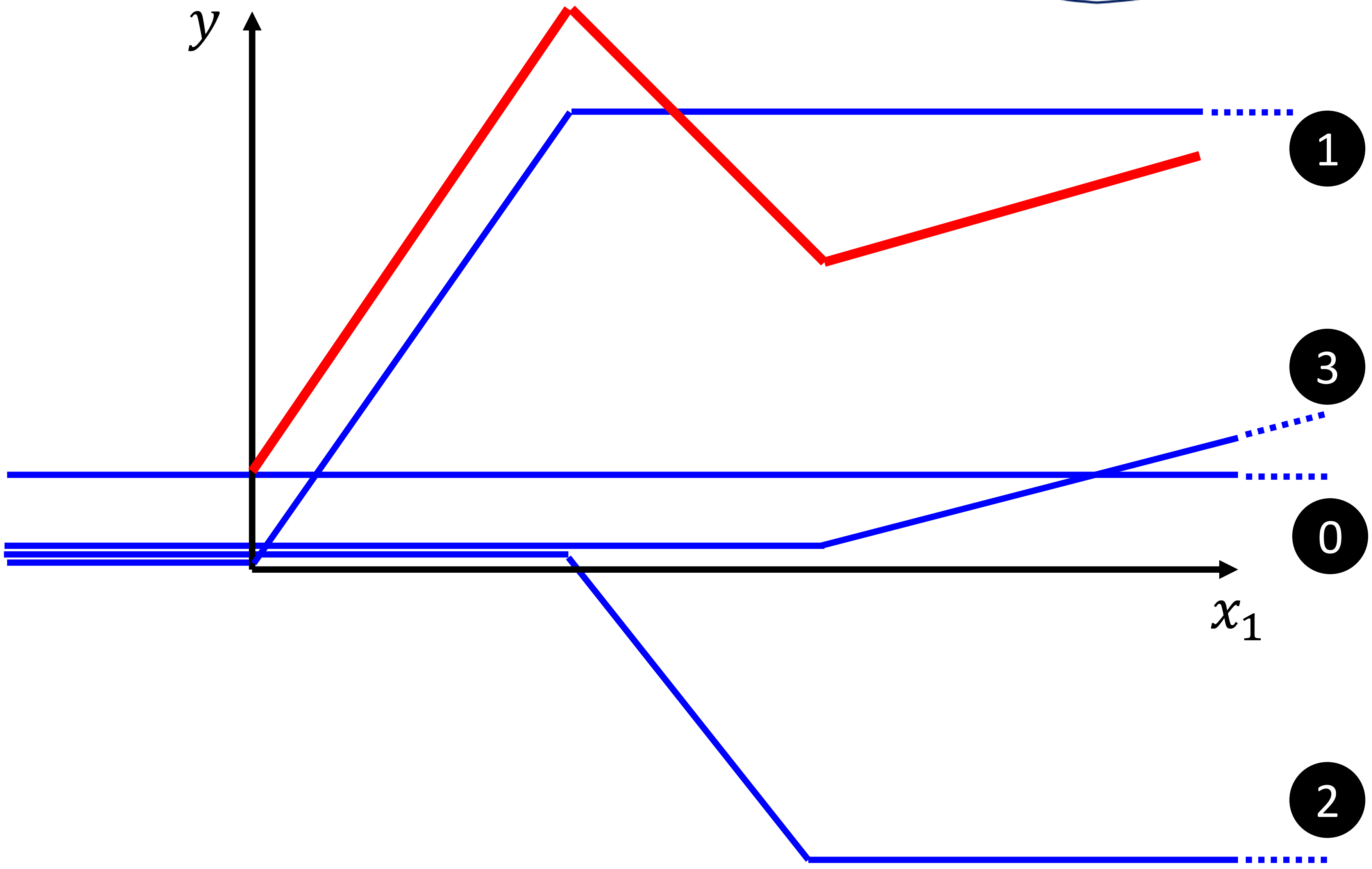


Linear models have severe limitation. ***Model Bias***

We need a more flexible model!

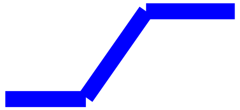
Explanation by prof. Hung-Yi Lee

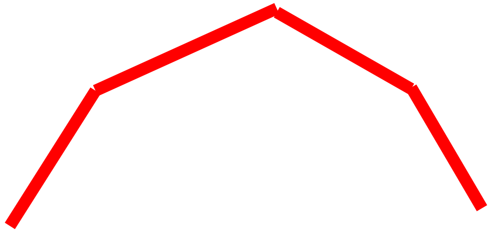
red curve = constant + sum of a set of 

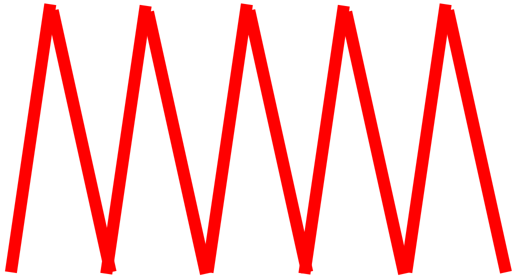


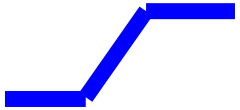
Explanation by prof. Hung-Yi Lee

All Piecewise Linear Curves

= constant + sum of a set of 



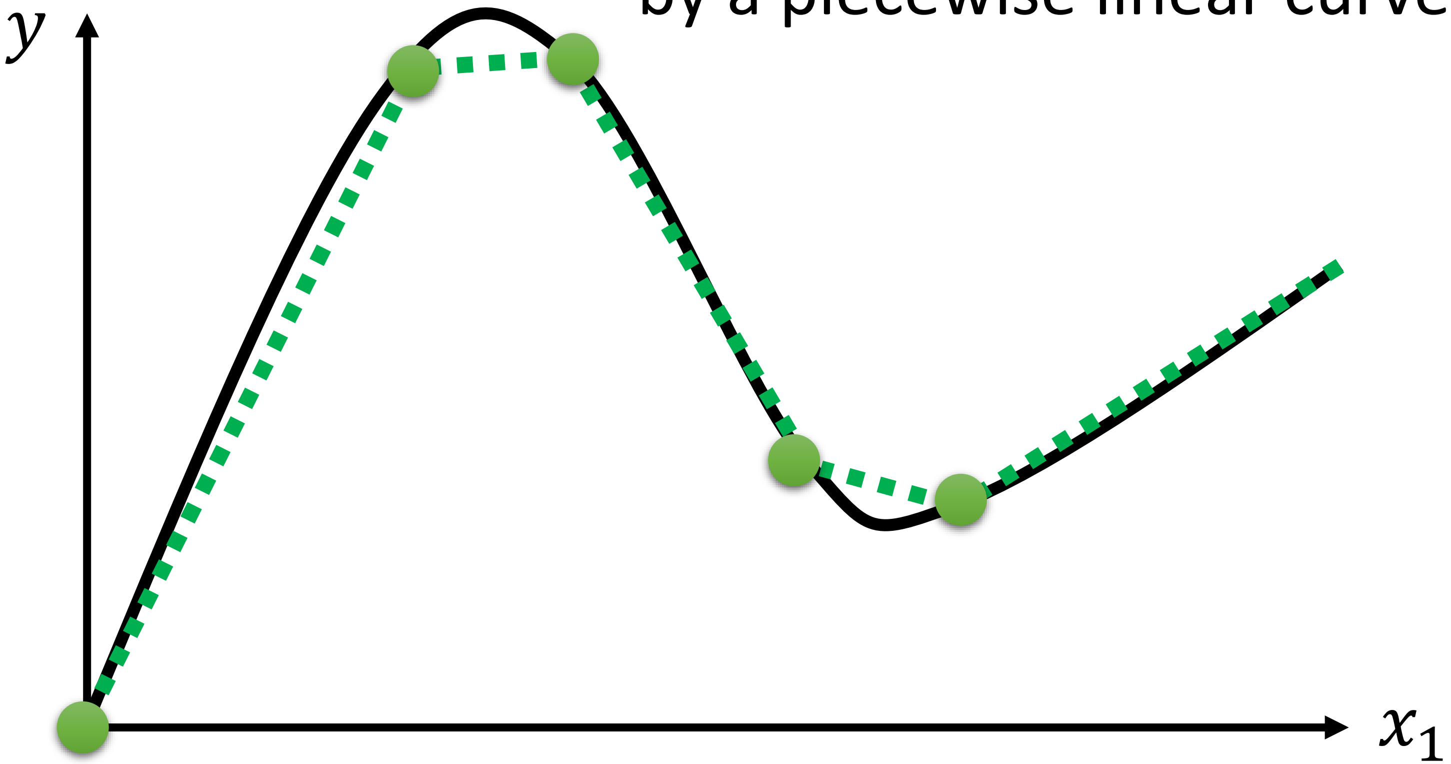


More pieces require more 

Explanation by prof. Hung-Yi Lee

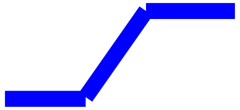
Beyond Piecewise Linear?

Approximate continuous curve by a piecewise linear curve.

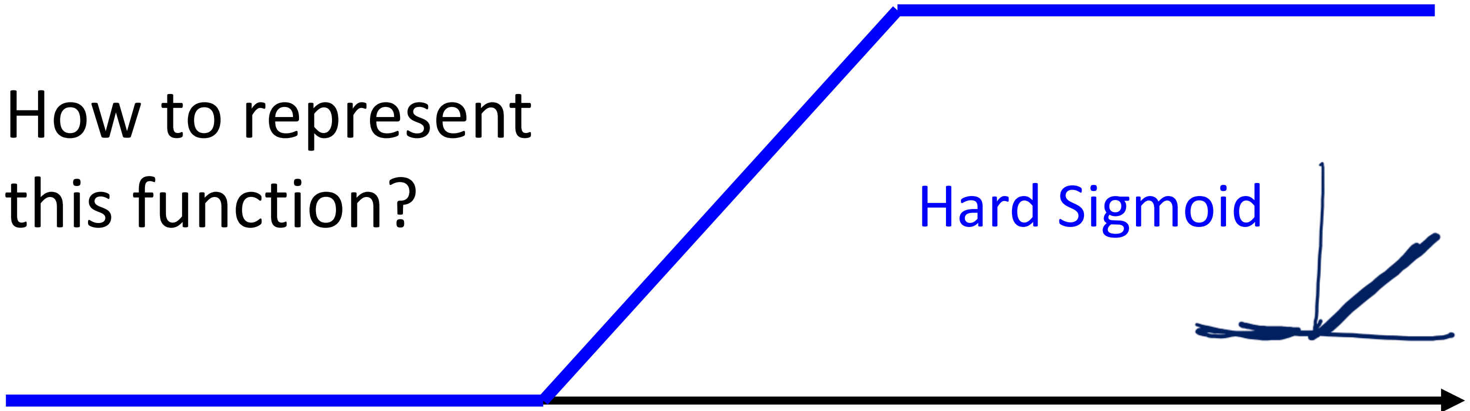


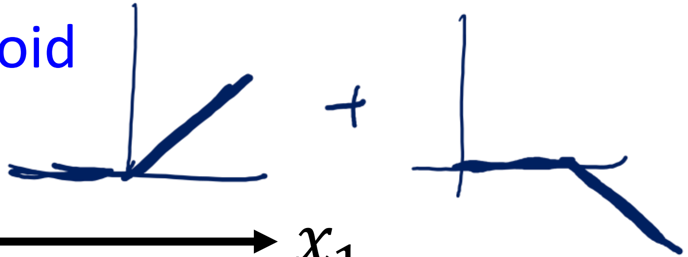
To have good approximation, we need sufficient pieces.

Explanation by prof. Hung-Yi Lee

red curve = constant + sum of a set of 

How to represent this function?

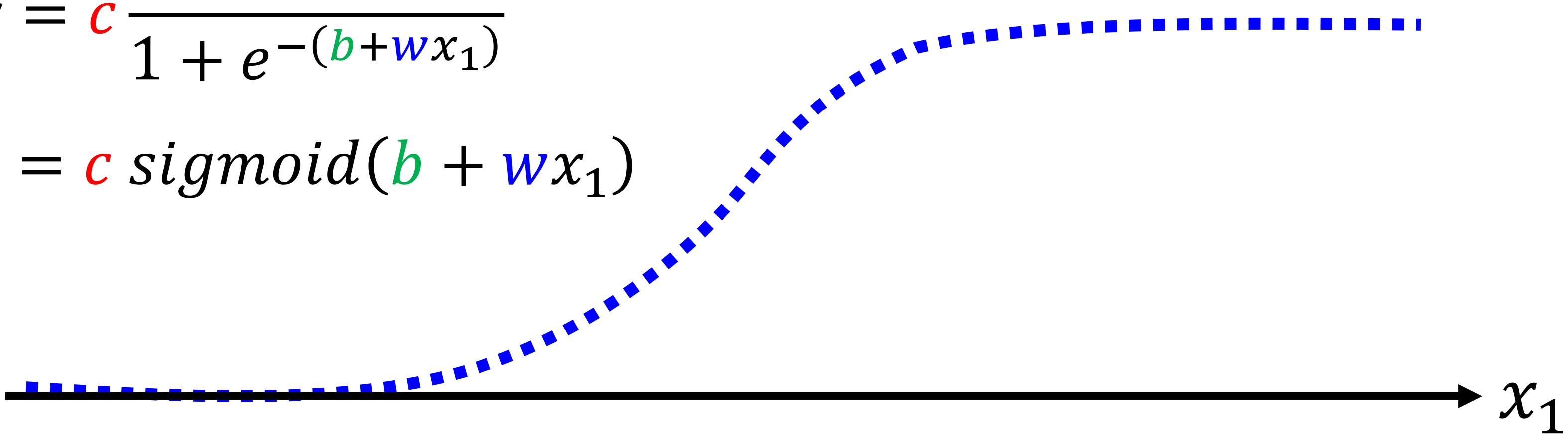


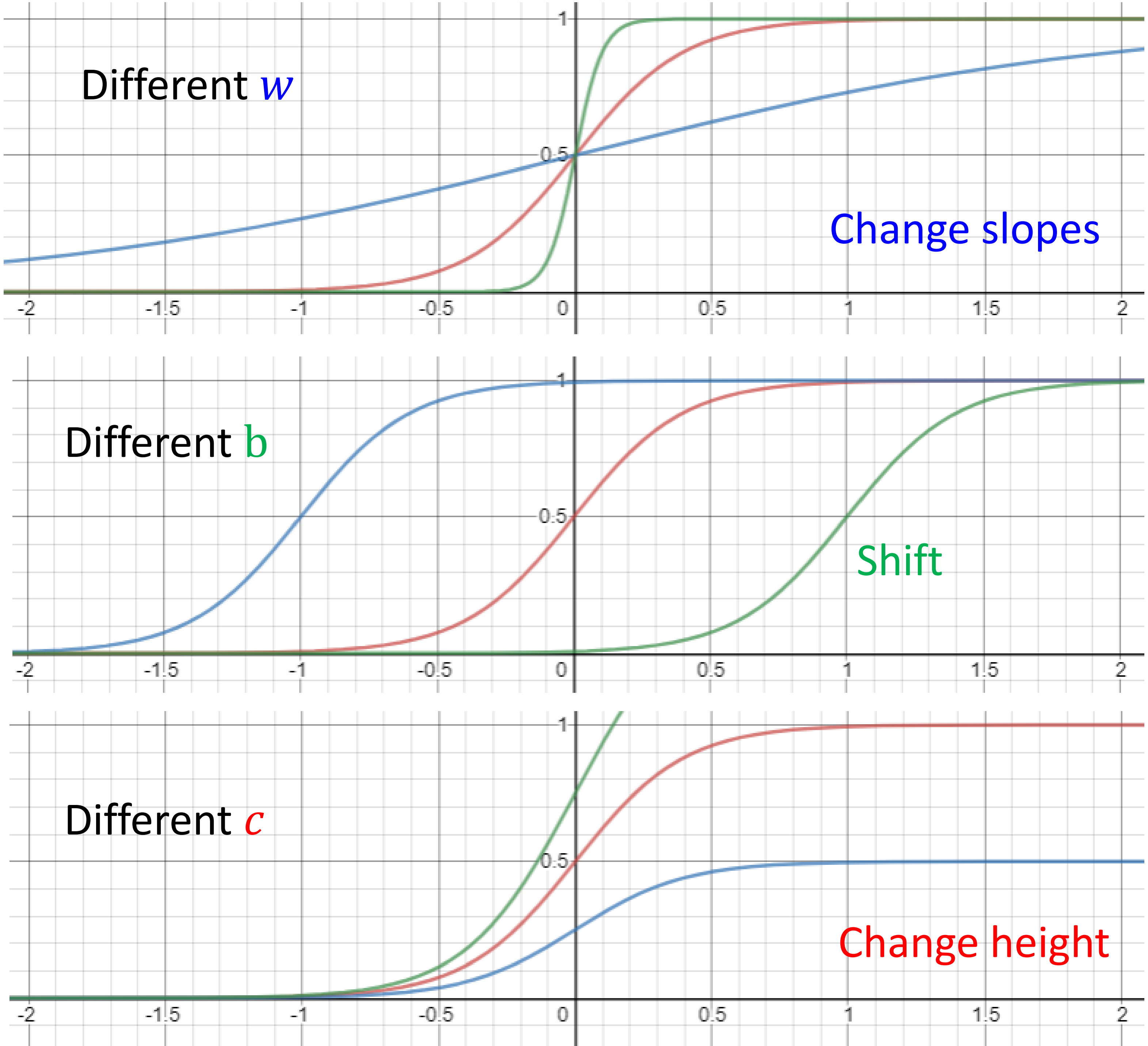


Sigmoid Function

$$y = c \frac{1}{1 + e^{-(b + w x_1)}}$$

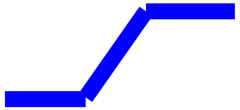
$$= c \; sigmoid(b + w x_1)$$

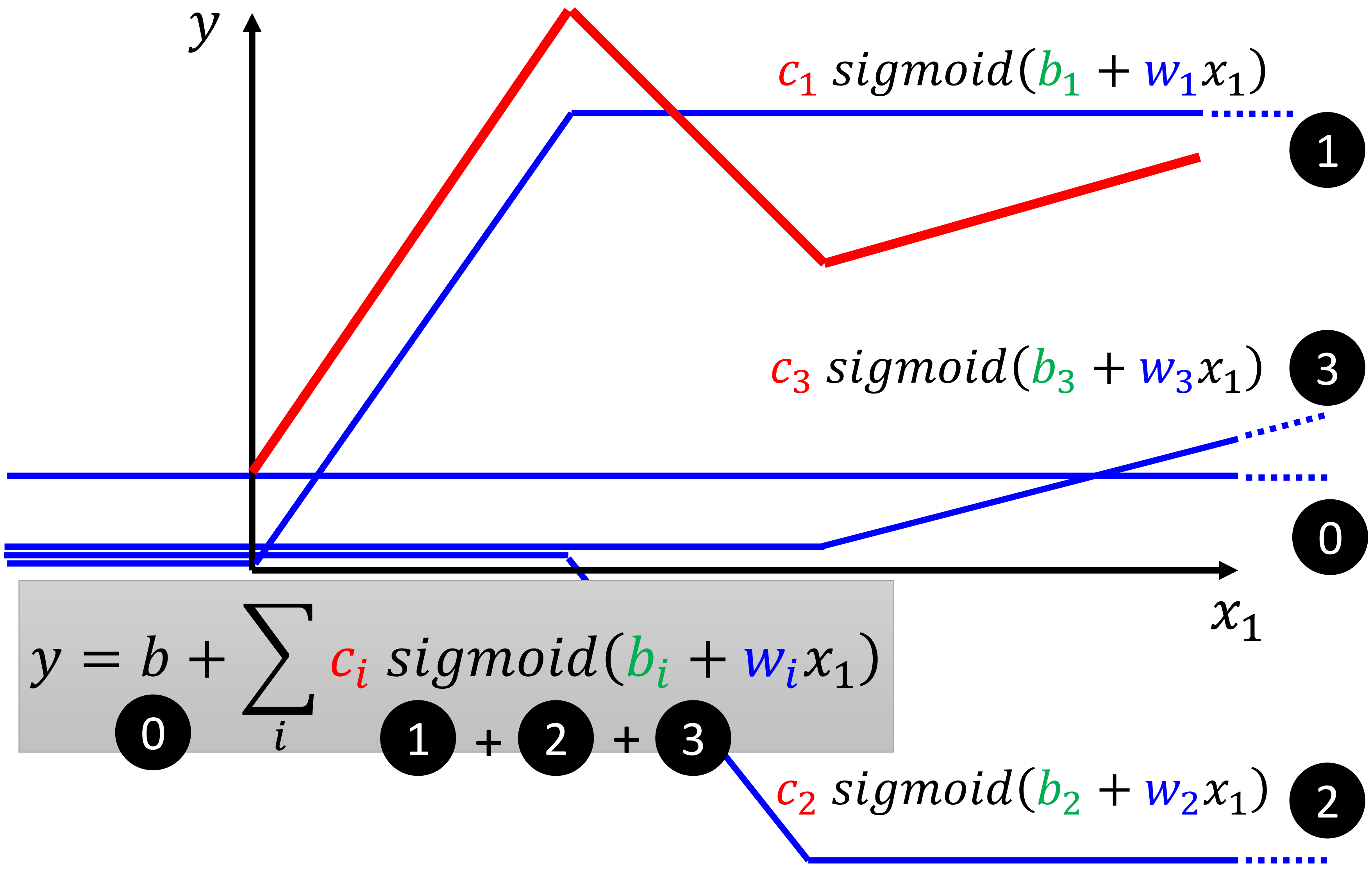




Different w
Change slopes
Different b
Shift
Different c
Change height

Explanation by prof. Hung-Yi Lee

red curve = sum of a set of  + constant

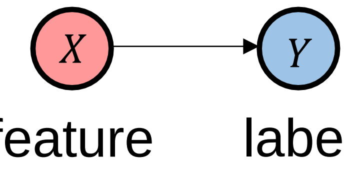


$$y = b + \sum_i c_i\ sigmoid(b_i + w_i x_1)$$

0 1 + 2 + 3

Recap: Neural Network for Classification (Discriminative Model)

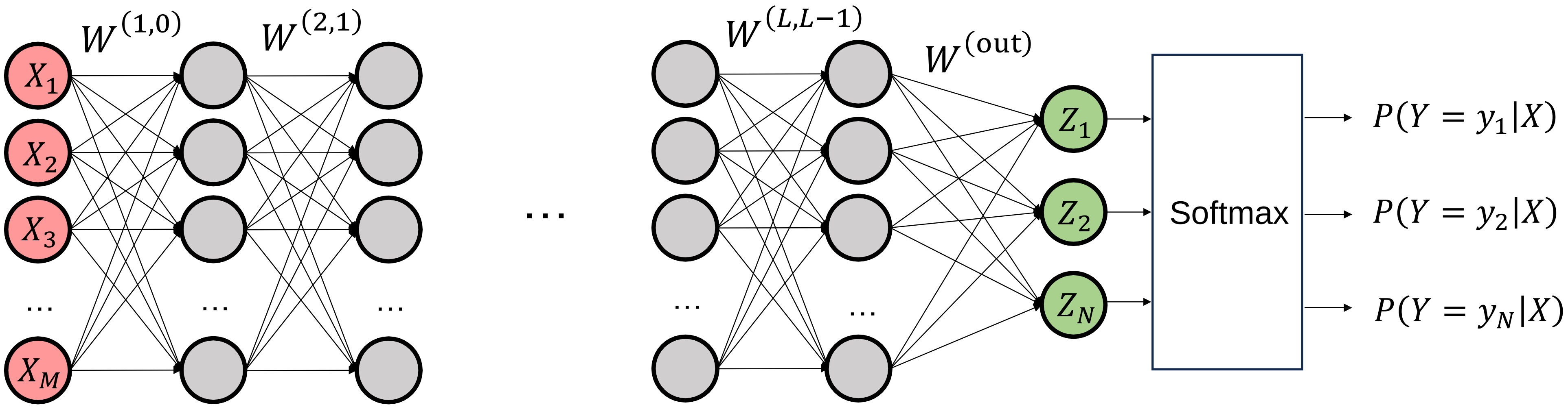
BN representation



Parameters in the most general case = $N \prod_{i=1}^{M} |X_i|$

X_i can take $|X_i|$ different values

NN modeling



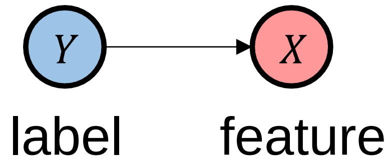
Parameters = $MK^{(1)} + K^{(1)}K^{(2)} + \cdots + K^{(L-1)}K^{(L)} + K^{(L)}N$

Transform the features layer by layer and perform logistic regression at the end

Maximum-likelihood training finds the transformations automatically

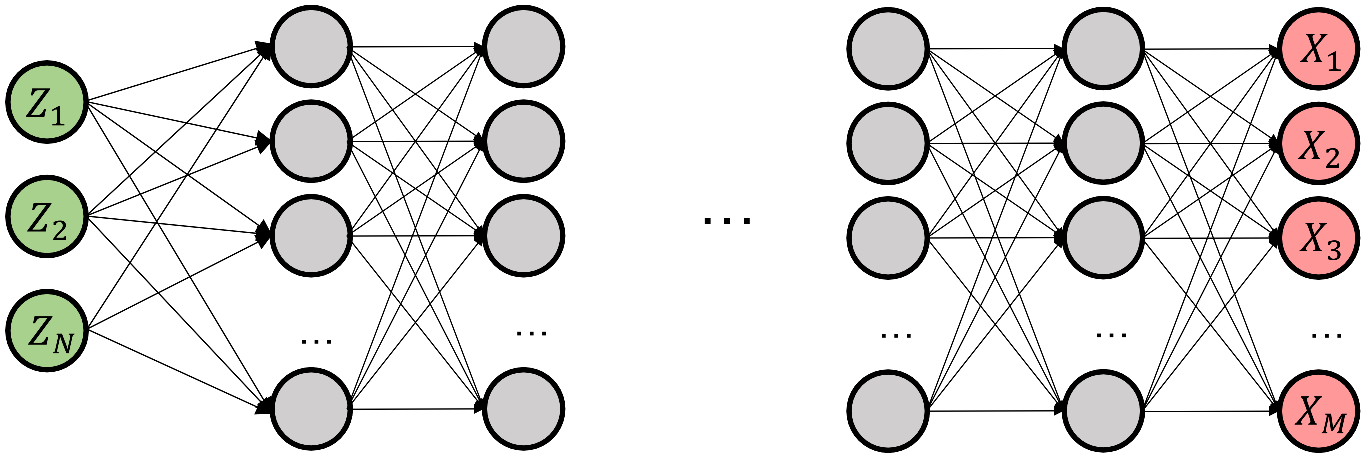
Recap: Neural Network can also implement Generative Models

BN representation



Let Z be the one-hot encoding of the label

NN modeling

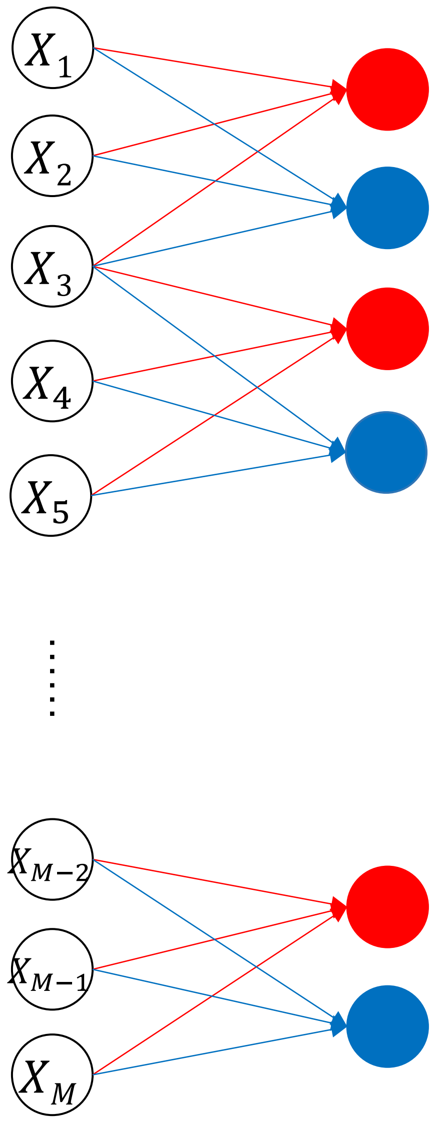


We less use this structure to perform classification, except for Naïve Bayes (a single-layer version). However, such structure is useful in **generating new samples**.

Generative Models lead to important applications in modern AI. For example, generating images or natural languages (will see more examples in the next week).

Neural Networks with Special Structures

Convolutional Neural Network (CNN)



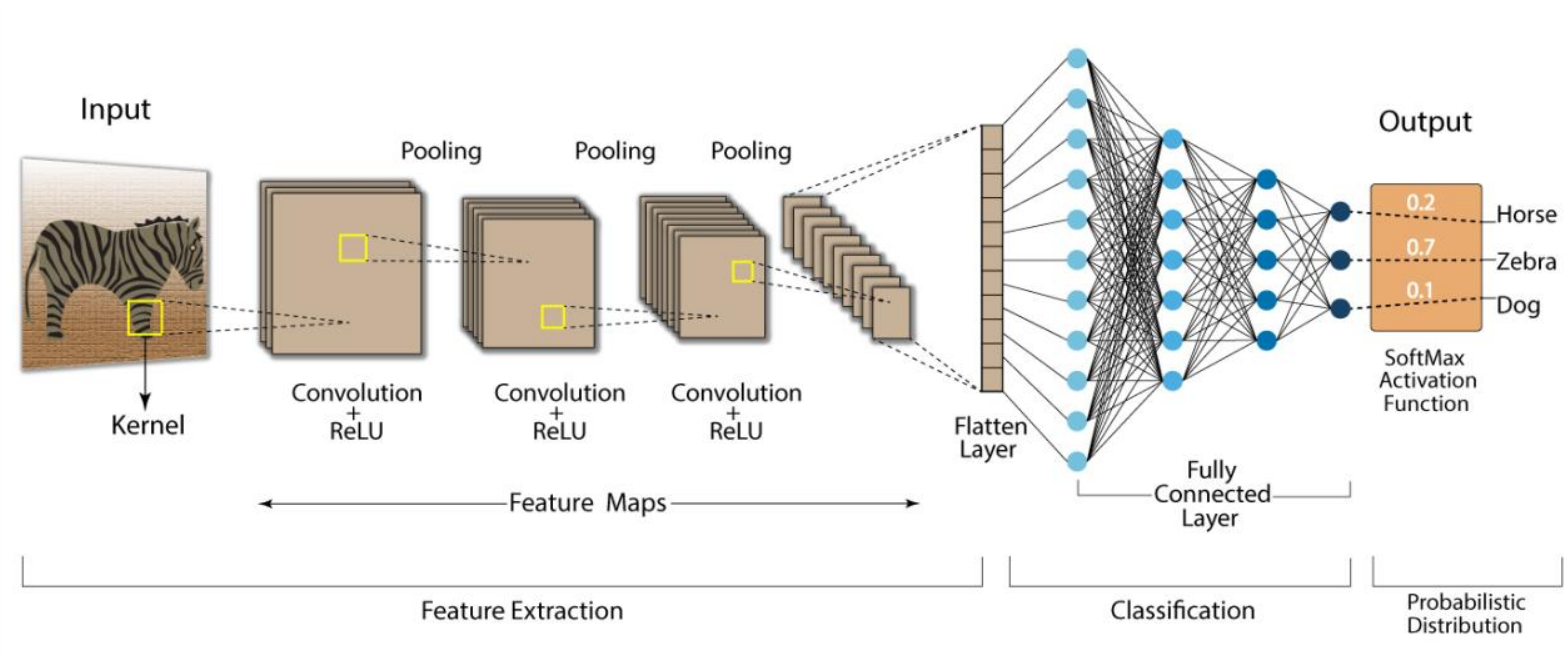
Useful in **image recognition**

Difference with the *fully connected* NN discussed previously:

- Each filter only covers a **local region** in the previous layer.
  This allows the NN to recognize local patterns.
  e.g., recognize wheels, windows in the image

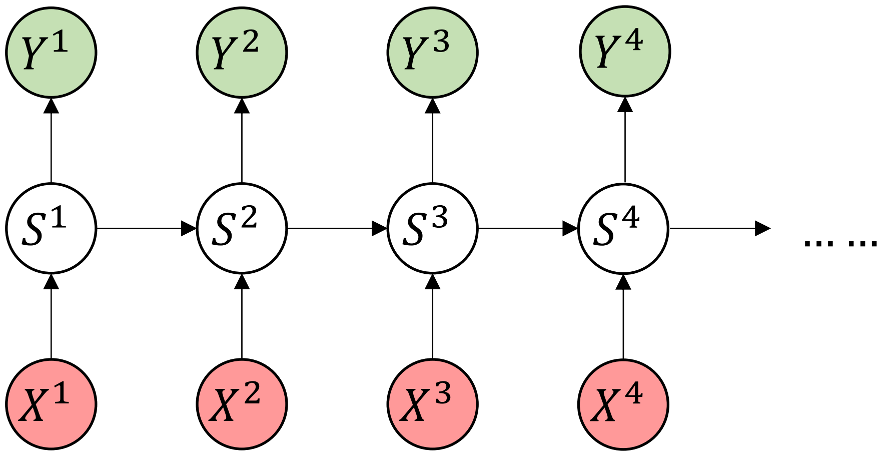
- Filters of the same color **shares the weights**.
  e.g., the red filters are able to recognize wheels everywhere in the image, and the blue filters recognize windows.

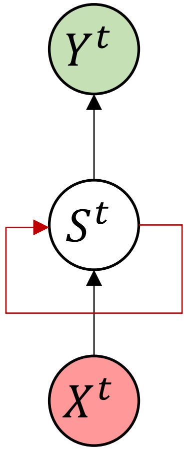
Convolutional Neural Network (CNN) for Computer Vision



https://www.linkedin.com/pulse/what-cnn-logesh-s-nxhfc/

Recurrent Neural Network (RNN)





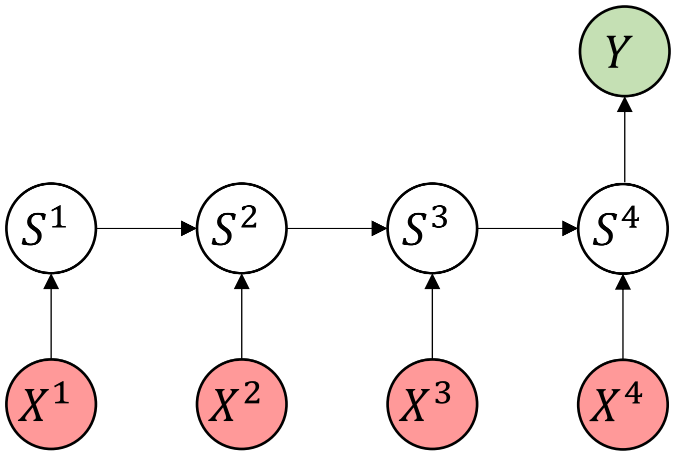
Restrictions (Markov property):

The parameters (i.e., neural network weights) for the mappings $X^t \to S^t$, $S^t \to Y^t$, and $S^t \to S^{t+1}$ are independent of t

Useful in modeling

- Sequence-to-sequence mapping
  - Translate English to French
  - Convert speech to English
  - Question and answering
- Language generation
  - $X^t = Y^{t-1}$

Recurrent Neural Network (RNN)



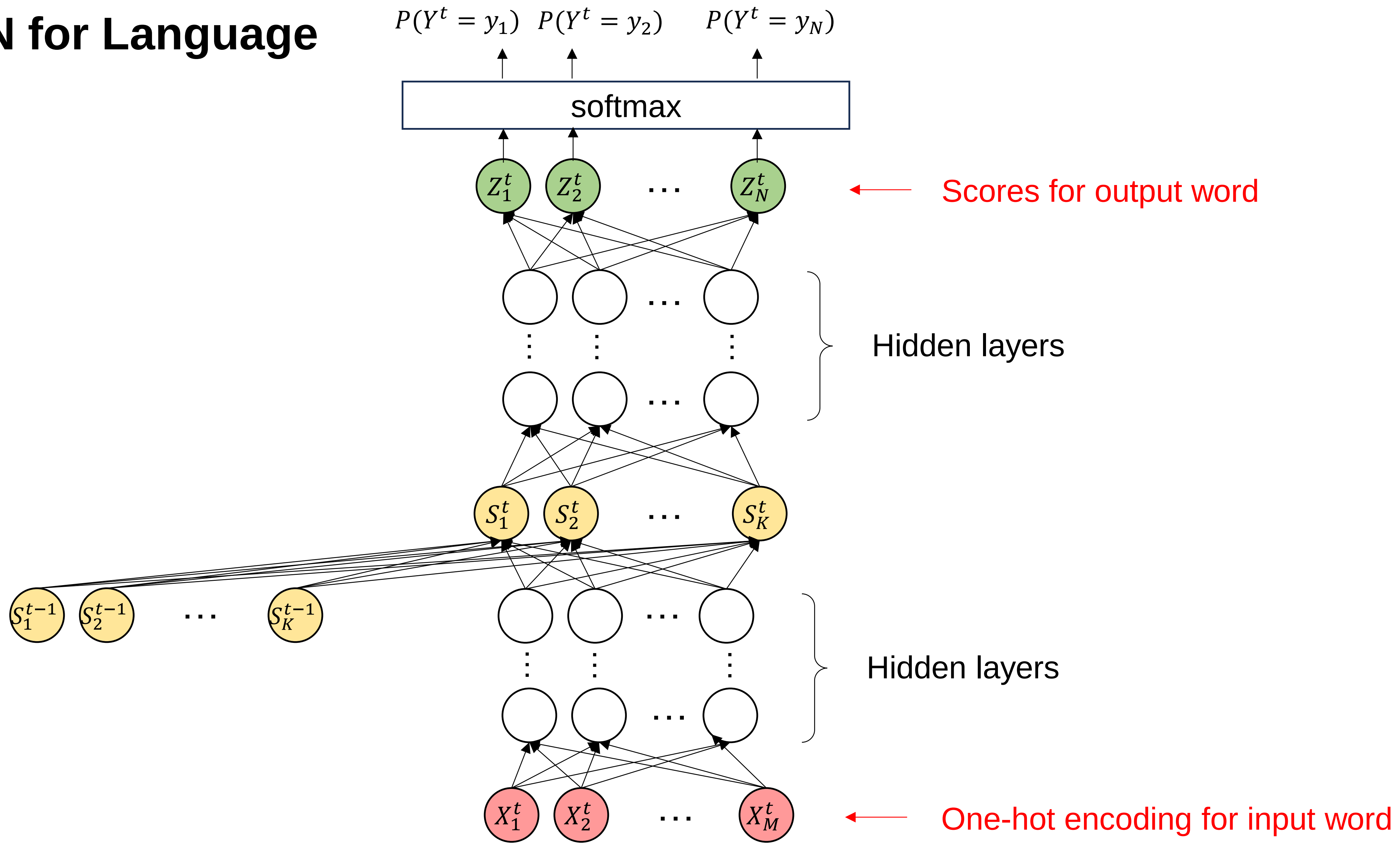
Restrictions (Markov property):

The parameters (i.e., neural network weights) for the mappings $X^t \rightarrow S^t$, $S^t \rightarrow Y^t$, and $S^t \rightarrow S^{t+1}$ are independent of t

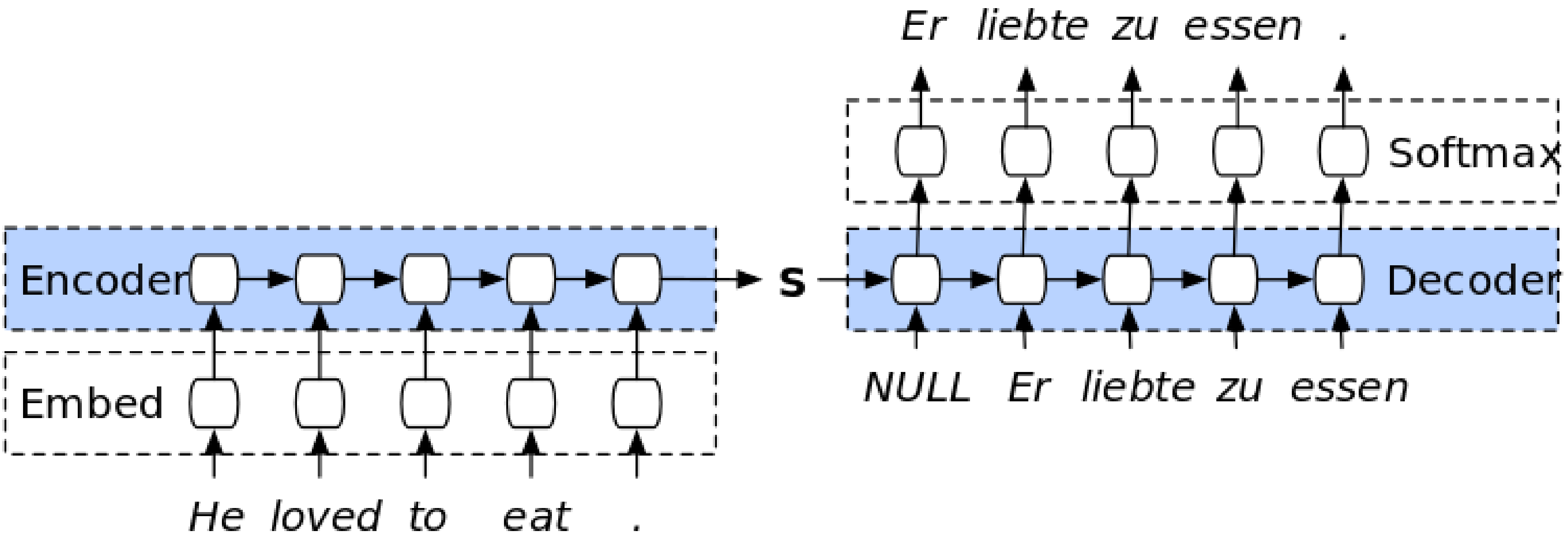
Useful in modeling

- Sequence classification: language detection, sentiment analysis

RNN for Language



RNN for Language



https://www.nbshare.io/notebook/313339236/English-to-German-Translation-using-Seq2Seq-Models-In-PyTorch/

Homework 5

Deadline: December 2

Homework 5

1. Choice Questions (10 points)
    a. 10 questions.
    b. Answer directly on Gradescope
    c. The same requirements as the last time.

1. Program Questions (Machine Learning) (19 points)
    - We skipped Question 1 (Perceptron). So we have q2, q3, q4
    - We use the original (NumPy) version not PyTorch version.
        - Libraries needed:
            - Numpy
            - Matplotlib (for 2D plotting)

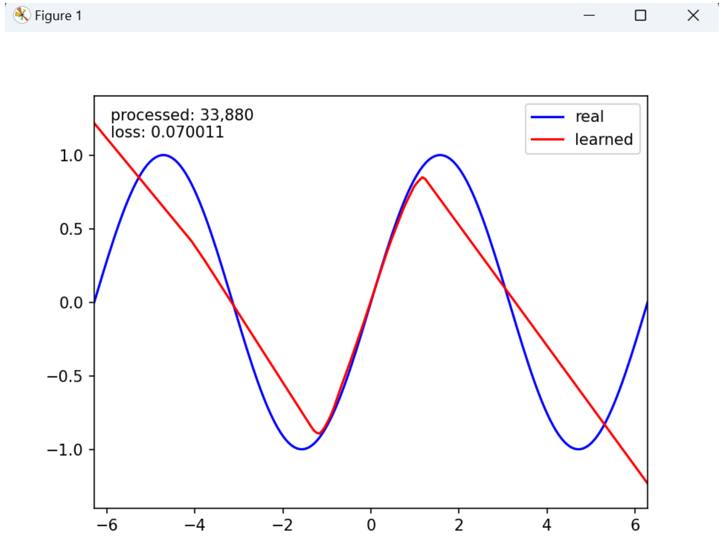
Introduction of Project 5: Q2 Non-linear Regression

Primary Task:

1. For this question, you will train a neural network to approximate sin(x) over [-2pi, 2pi]
2. Backward and dataset loader already implemented.
3. You need Implement *model* with:
   a. Initialization
   b. Run (model forward)
   c. Get_loss (return a loss for a given input and target)
   d. Train (train the model using gradient-based updates)

   (Similar for the other questions)

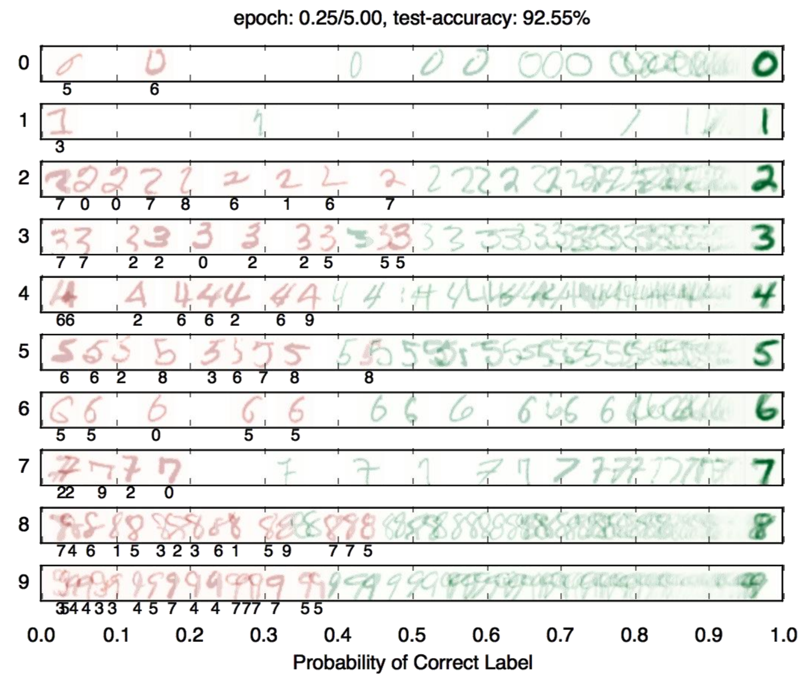
1. Receive full points if it gets a loss of 0.02 or lower.



Introduction of Project 5: Q3 Digit Classification

Primary Task:

1. Train a network to classify handwritten digits from the MNIST dataset.
2. Each digit is of size 28 by 28 pixels and flat to 784 dimensional vector.
3. Achieve an accuracy of at least 97% to get the full score.



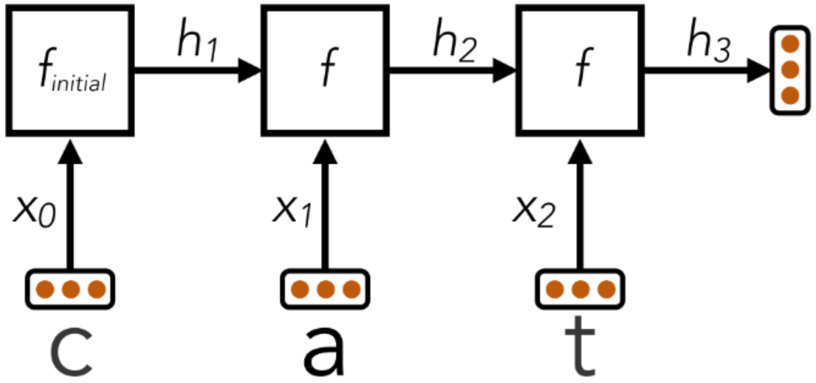
Introduction of Project 5: Q4 Digit Classification

Primary Task:

1. Figure out, given a piece of text, what language the text is written in.

   For example:

   - discussed → English
   - eternidad → Spanish
1. Use RNN to handle variable-length inputs.
2. Achieve an accuracy of at least 81% to get the full score.



Lectures Next Week

- Tuesday: applying deep learning to **computer vision** by Prof. Zezhou Cheng
  - Recognition: image classification, object detection, etc.
  - Generation: image generation, video generation, etc.
  - Reconstruction: 2D to 3D, VR/AR, etc.

- Thursday: applying deep learning to **natural languages** by Prof. Yu Meng
  - Sequence-to-sequence learning: language translation, etc.
  - Language generation: large language model (ChatGPT), etc.