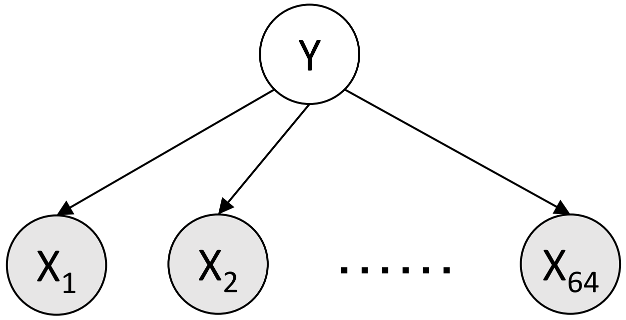
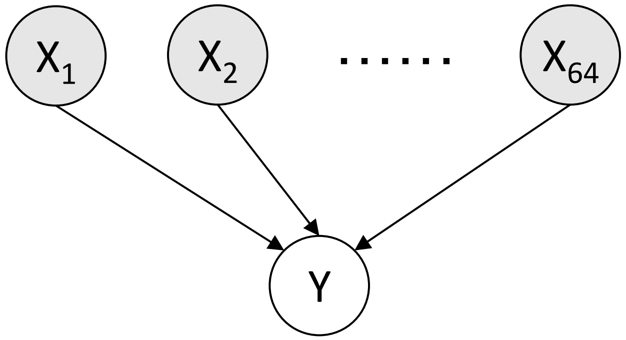
Neural Network

Chen-Yu Wei

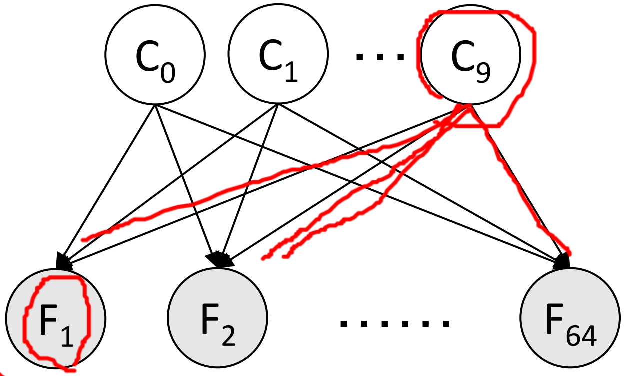
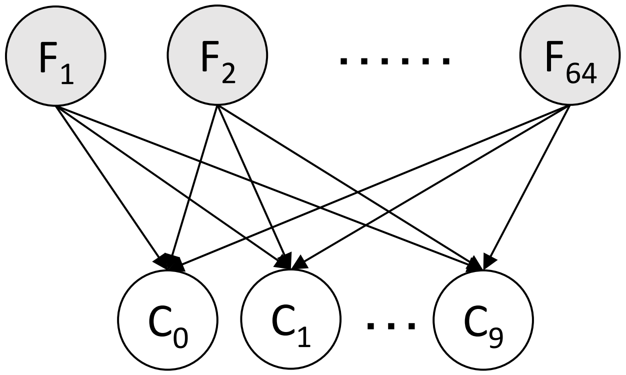
Naïve Bayes and Logistic Regression

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

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

Naïve Bayes and Logistic Regression

W_{ij}: the weight between F_i and C_j

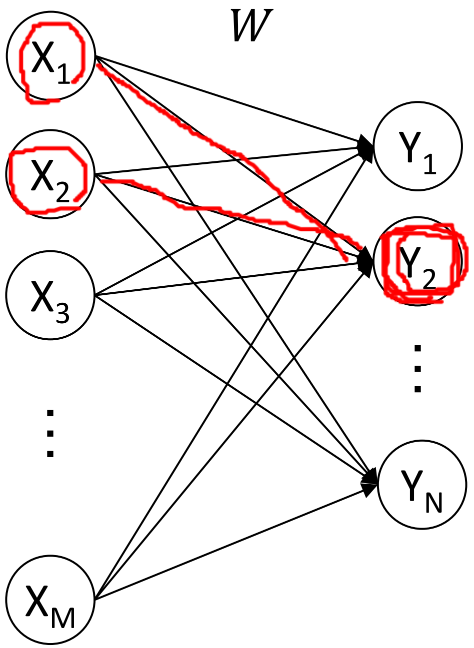
	Naïve Bayes	Logistic Regression
"Neural Net" representation $W_{ij} = p(x_i = 1 \mid y = j)$	 $F_i = \sum_{j=0}^{9} W_{ij} C_j$ $= W_{ij}$	 $C_j = \sum_{i=1}^{64} W_{ij} F_i$ $W_i^{(j)}$
The meaning of C_j	Class $= j \Leftrightarrow (C_0, \dots, C_9) = (0, \dots, 1, \dots, 0)$	The score between class 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

Input

$X \in \mathbb{R}^M$



Output

$Y \in \mathbb{R}^N$

This neural network describes the relation

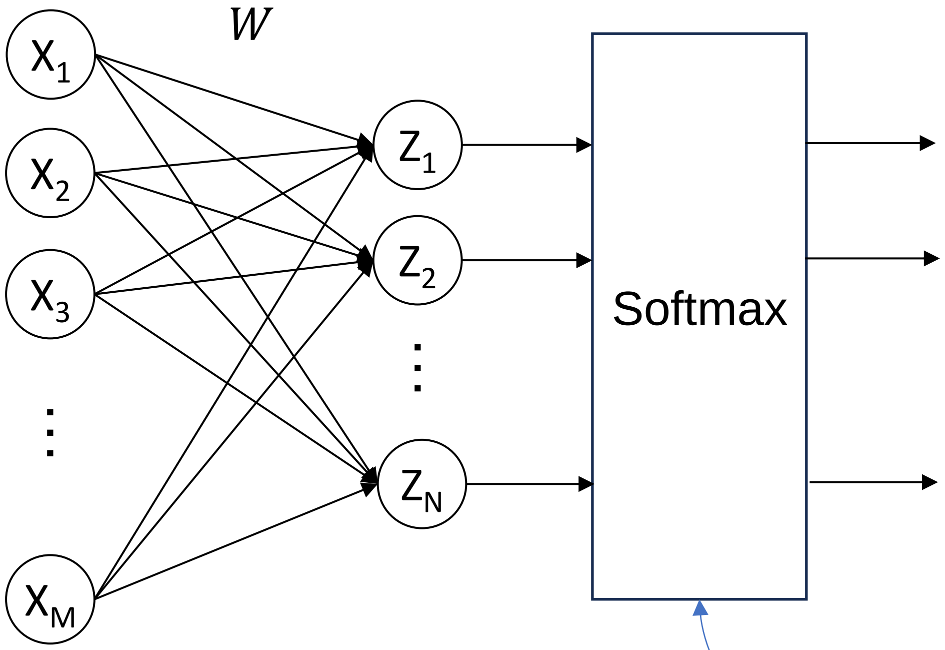
$$Y_i = \sum_{j=1}^{M} W_{ij} X_j \qquad \forall i = 1, \ldots, N$$

or, more succinctly, $\overset{N}{Y} = \underset{N \times M}{W} \overset{M}{X}$

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)



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

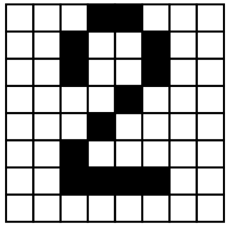
$$\frac{\exp(Z_2)}{\sum_{j=1}^{N}\exp(Z_j)} = P_W(Y = 2 \mid X)$$

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

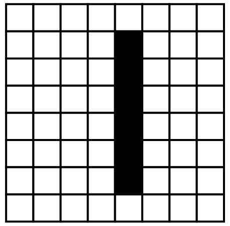
Additional operation to fulfill the restriction on the final output (e.g., here we want the output to be a distribution)

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

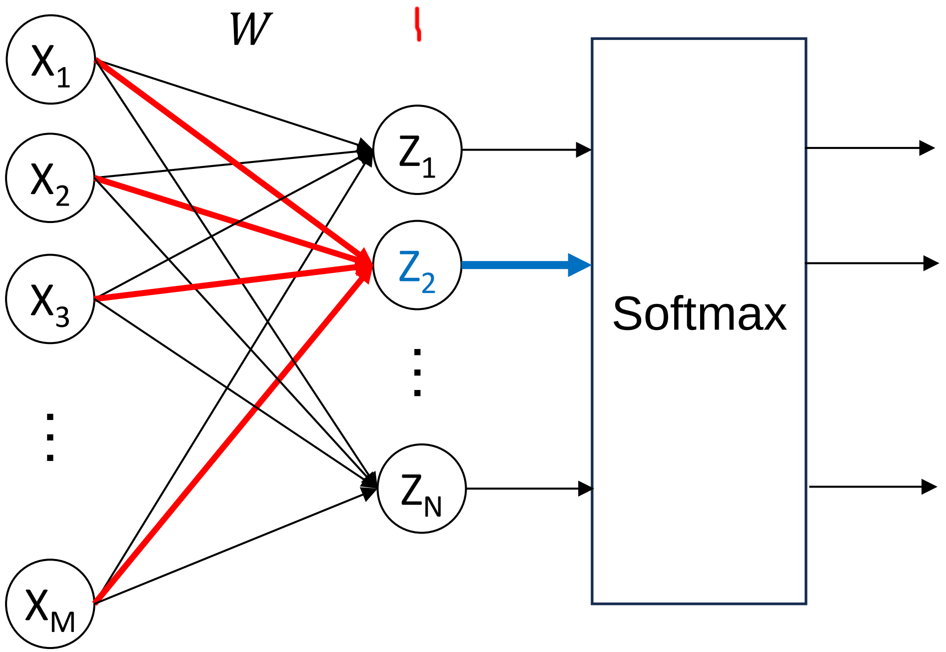
Logistic Regression (1-Layer NN for Classification)



Higher Z_2



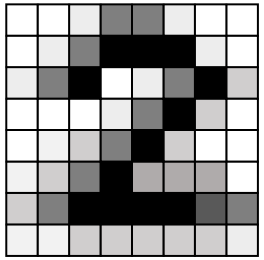
Lower Z_2



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

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

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

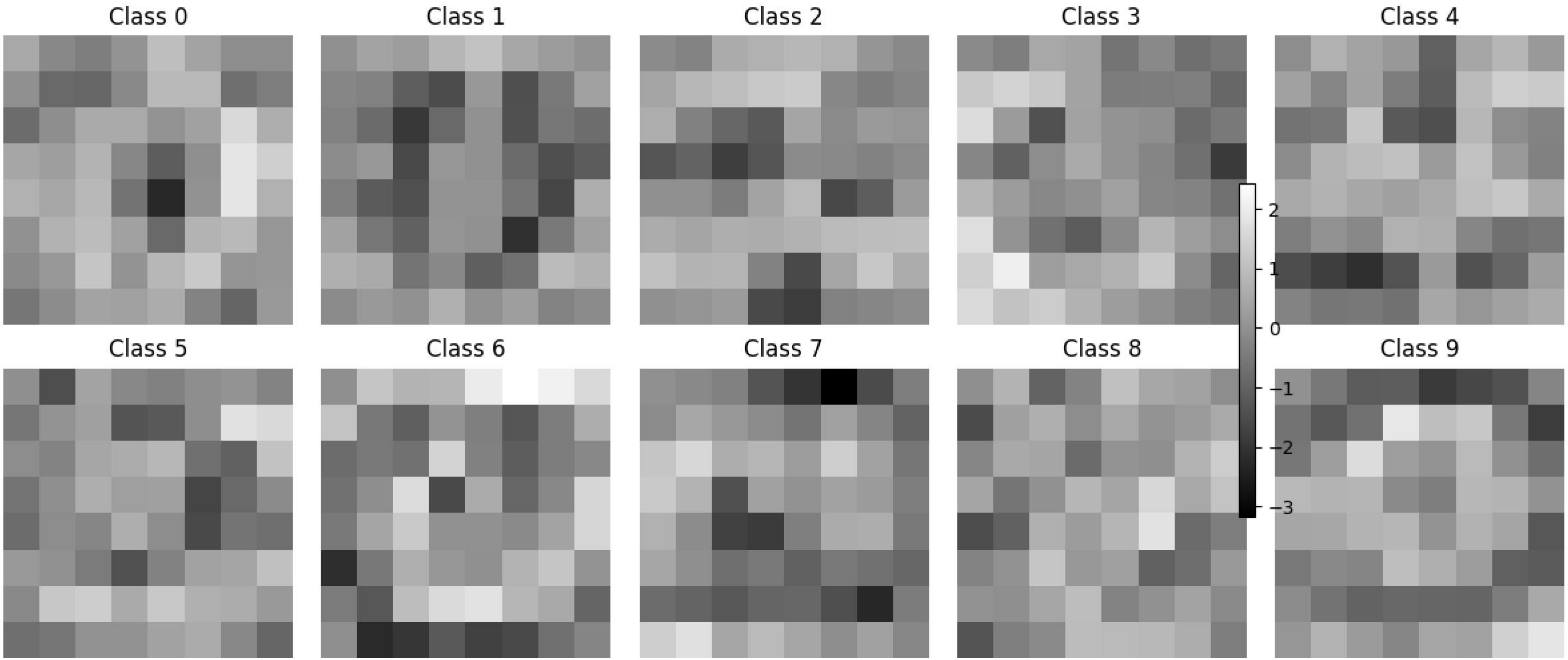


$$W_2 = (W_{2,1}, W_{2,2,\ldots,}, W_{2,64})$$

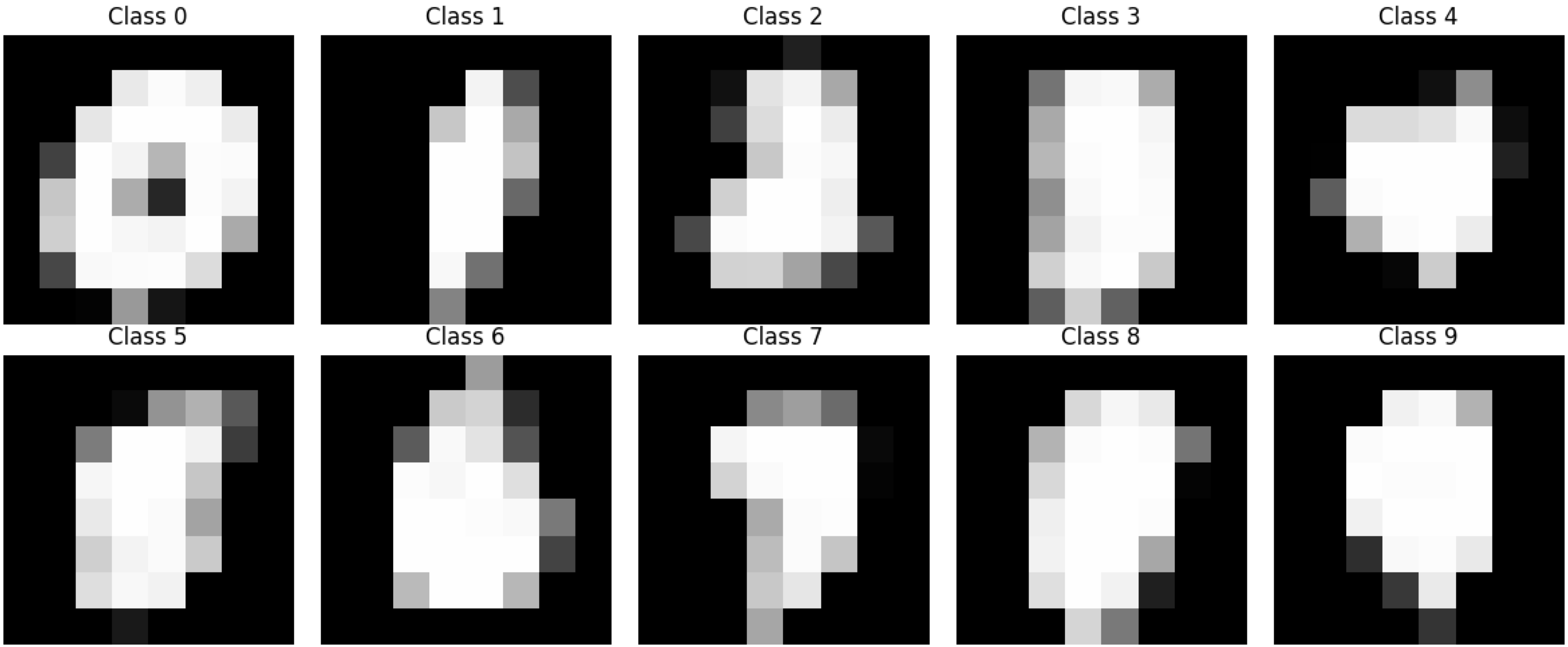
Z_2 will be high if the input pattern X matches W_2 (i.e., $X \cdot W_2$ is large)

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

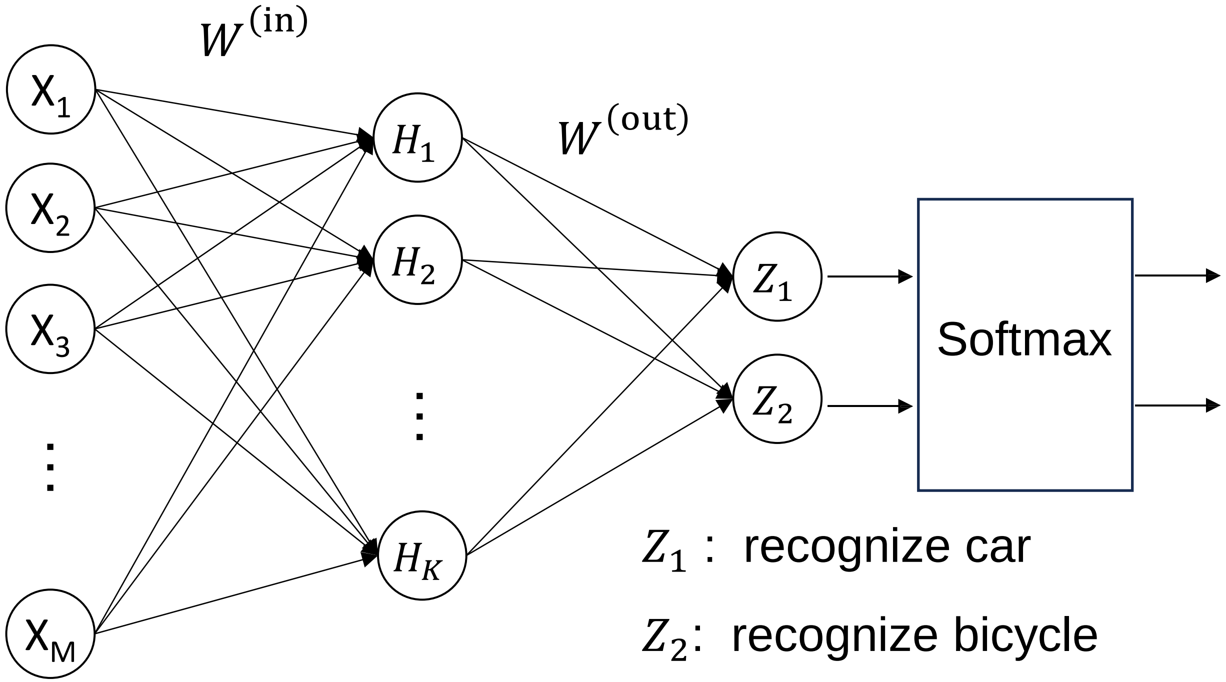
The Weights Produced by Logistic Regression



The Weights Produced by Naïve Bayes



2-Layer NN for Classification



Z_1 : recognize car

Z_2: recognize bicycle

H_1 : recognize wheels

H_2 : recognize windows

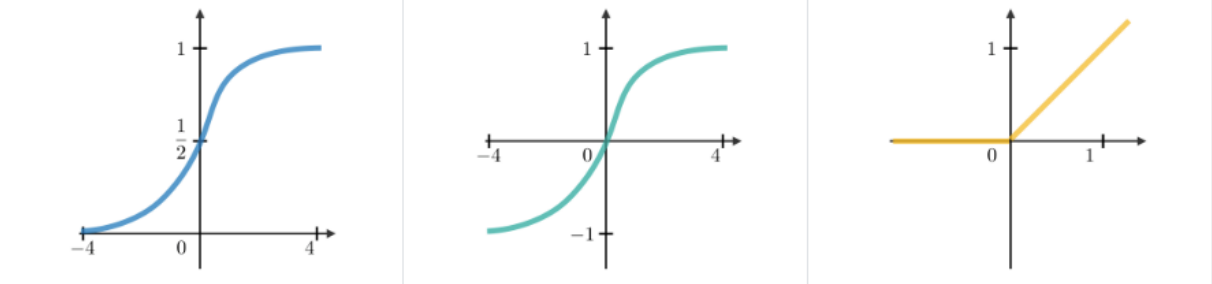
H_3 : recognize handlebar

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

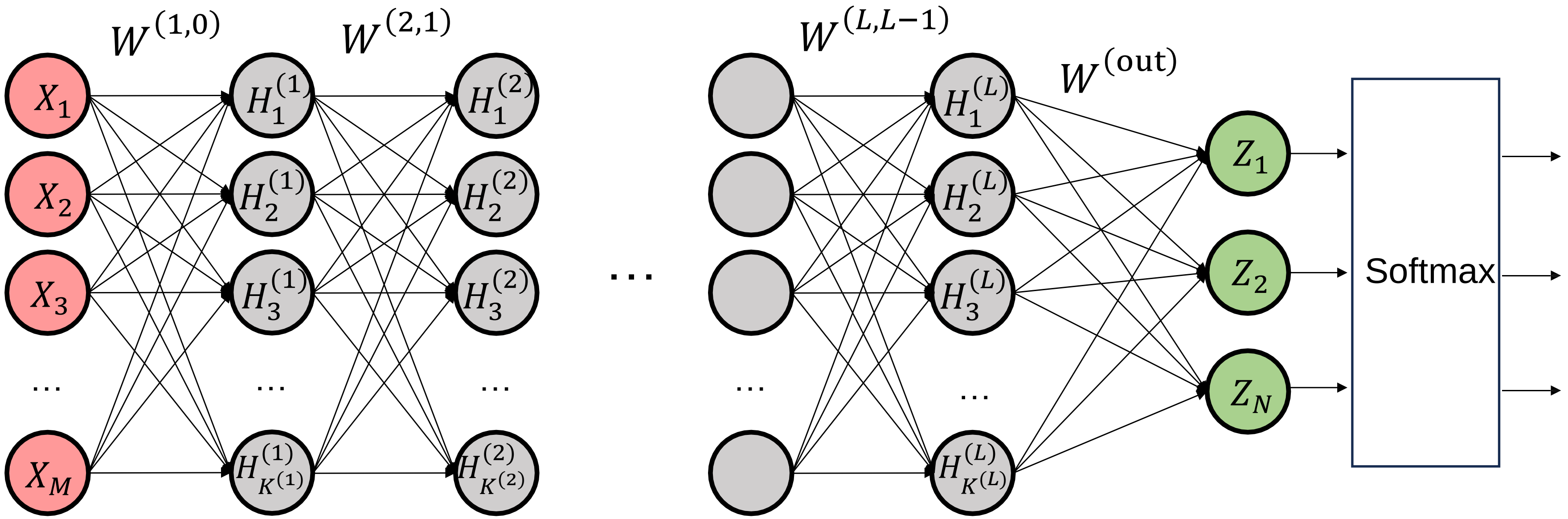
g = nonlinear **activation function**

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

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



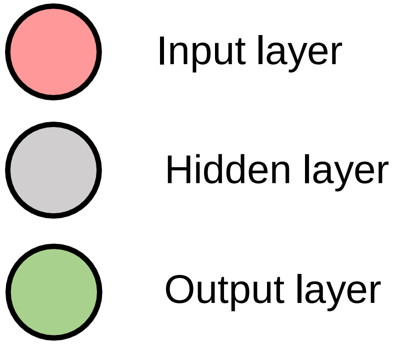
Multi-Layer NN for Classification



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

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

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



Training Multi-Layer Neural Network

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

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

$$\begin{aligned} f_W(x,y) &= e_y^\top W^{(\text{out})} H^{(L)} \\ &= e_y^\top W^{(\text{out})} g\left(W^{(L,L-1)} H^{(L-1)}\right) \\ &= e_y^\top W^{(\text{out})} g\left(W^{(L,L-1)} g\left(W^{(L-1,L-2)} H^{(L-2)}\right)\right) \\ &= \underbrace{e_y^\top W^{(\text{out})} g\left(W^{(L,L-1)} g\left(W^{(L-1,L-2)} g\left(\dots . g\left(W^{(1,0)} x\right)\right)\right)\right)}_{\text{A quite complicated non-linear function of } W = (W^{(\text{out})}, W^{(L,L-1)}, \dots, W^{(1,0)})} \end{aligned}$$

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

- 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.

hidden in W