Harmony in Complexity: Unveiling Mathematical Unity Across Logistic Regression, Artificial Neural Networks and Computer Vision

Liliang Chen

Freddie Mac

chenliliang@gmail.com

ODSC EAST 2024

BOSTON | APRIL 23–25

Agenda

- Basics of Logistic Regression
- Understand $w^T x$ from the Sum of Vectors
- Geometric Interpretation of a Logistic Regression
- Goal of an Activation Function
- Logistic Regression vs. Neural Network: One-Layer, Two Layer, Hidden Layer
- Back-propagation of Error vs. Forward Activation
- Application of Convolutional Neural Network (CNN) in Computer Vision
- CNN vs. Neural Network
- CNN’s Back-propagation

Basics of Logistic Regression

Binary Classification: $y \in \{ 0, 1 \}$

Predict the probability of being in a particular class: $P(y = 1 \mid x; w)$

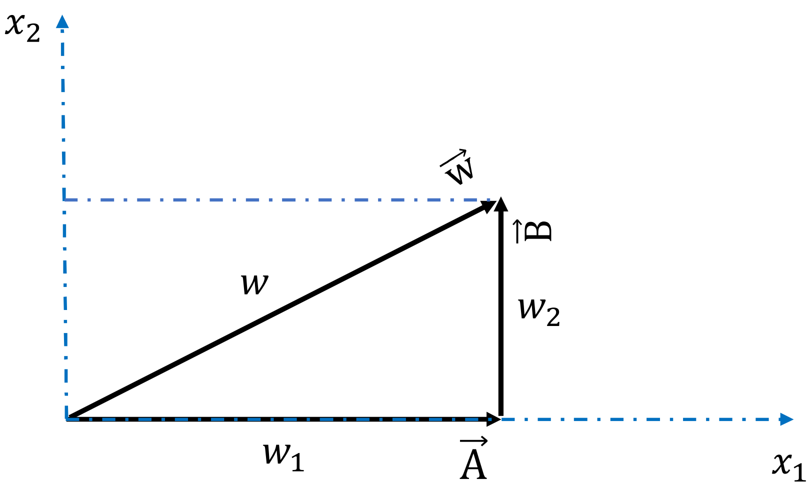
Could fit a linear model: $f(x, w) = w^T x$

Use the sigmoid function to force the output to lie in [0,1] range:

$$f(x, w) = Activation(w^T x) = \frac{1}{1+e^{-w^T x}}$$

Understand $w^T x$ from Geometry (the Sum of Vectors)

Vector addition in 2D

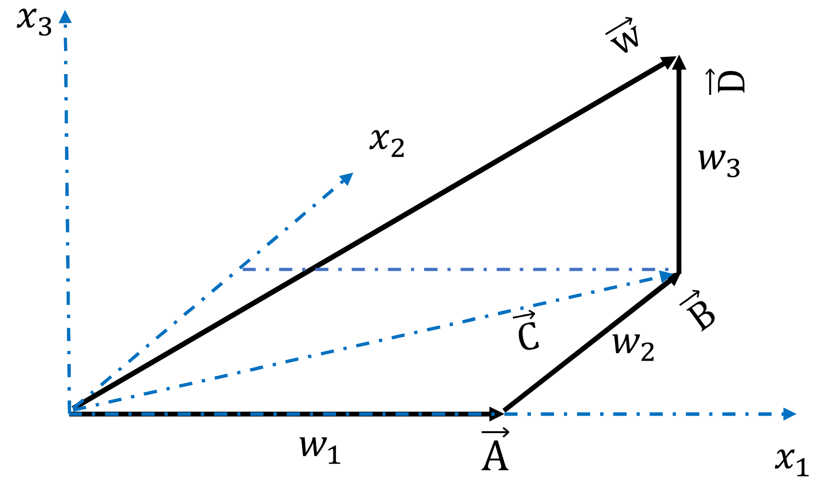


$$\vec{w} = \vec{A} + \vec{B} = w_1\overrightarrow{x_1} + w_2\overrightarrow{x_2} = w\vec{X}$$

$$\vec{w} = < w_1, w_2 >$$

$$\|w\| = \sqrt{w_1^2 + w_2^2}$$

Vector addition in 3D



$$\vec{C} = \vec{A} + \vec{B} = w_1\overrightarrow{x_1} + w_2\overrightarrow{x_2} = w_c\overrightarrow{x_c}$$

$$\vec{w} = \vec{C} + \vec{D} = w_c\overrightarrow{x_c} + w_3\overrightarrow{x_3} = w_1\overrightarrow{x_1} + w_2\overrightarrow{x_2} + w_3\overrightarrow{x_3} = w\vec{X}$$

$$\vec{w} = < w_1, w_2, w_3 >$$

$$\|w\| = \sqrt{w_1^2 + w_2^2 + w_3^2}$$

We can think of $\text{w}^\text{T}\text{x}$ as tranforming from multiple independent vectors into a single vector with a deterministic direction and magnitude

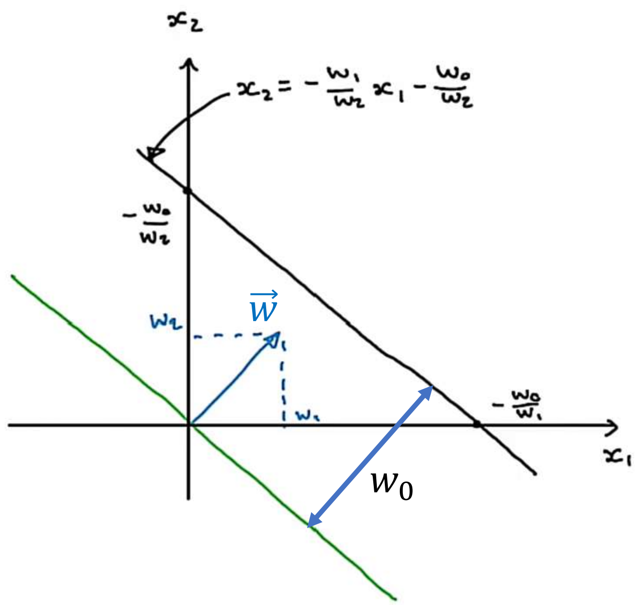
Understanding the linear model shape $w^T x = 0$ from geometry

$w^T X = 0$ defines a line in a two-dimension space vs a hyper-plane in a three-dimension space

The unit vector normal to the line/plane has the same direction as the sum of vectors based on individual vectors.

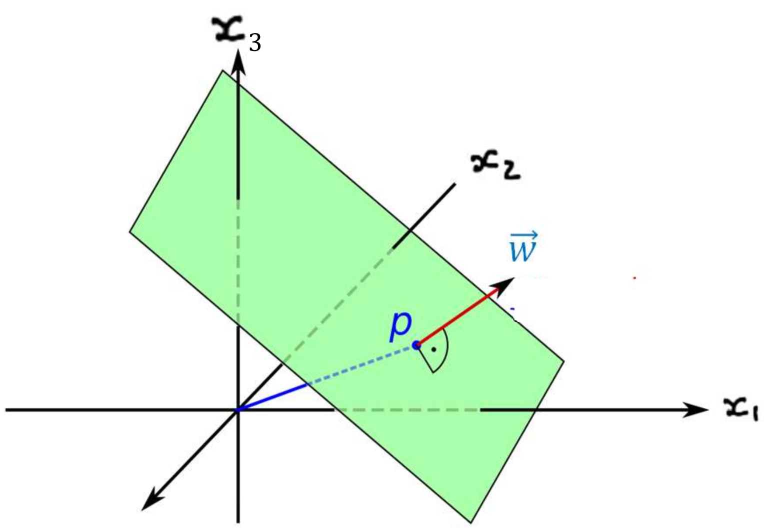
$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

The unit vector normal to this line is $\dfrac{< w_1, w_2 >}{\|w\|}$



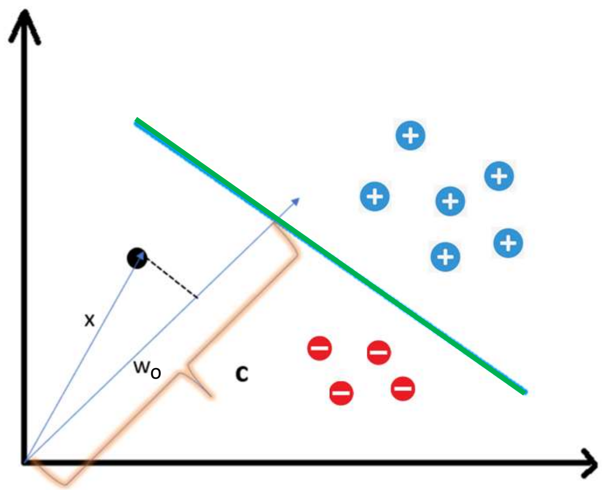
$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 = 0$$

The unit vector normal to this plane is $\dfrac{< w_1, w_2, w_3 >}{\|w\|}$

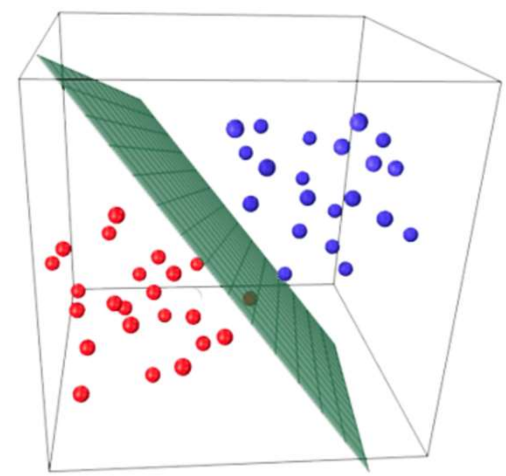


Geometric Interpretation of a Logistic Regression

Decision Boundary as a line in 2D



Decision Boundary as a hyperplane in 3D



Logistic regression seeks the decision boundary to perfect linear separate positive and negative points;
Classification depends on comparing relative distance from the origin to the data points vs. the decision boundary.

$$w \cdot \vec{X} = c \ (\textit{the point lies on the decision boundary})$$

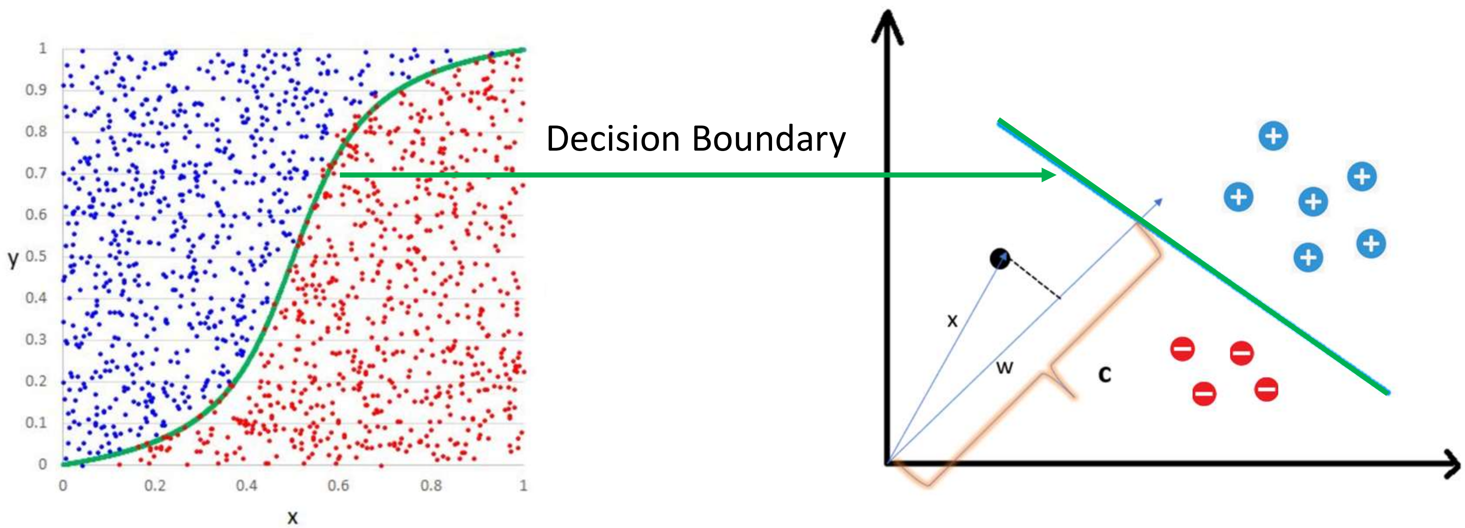
$$w \cdot \vec{X} > c \ (\textit{positive classifcation})$$

$$w \cdot \vec{X} < c \ (\textit{negative classifcation})$$

Goal of an Activation Function

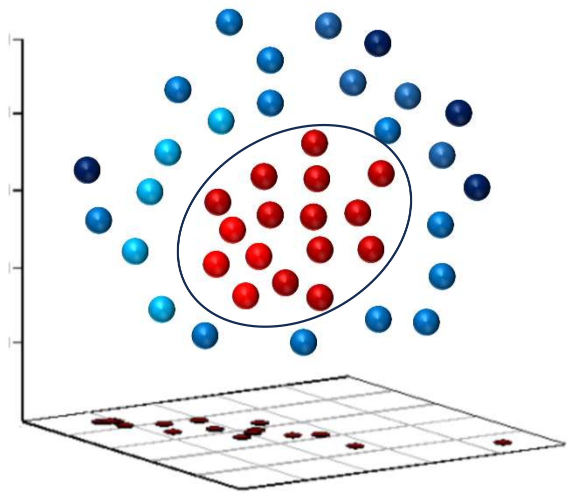
Activation function translates a line into a nonlinear decision boundary in a 2D space;
Logistic regression normally choose sigmoid function as its activation function:

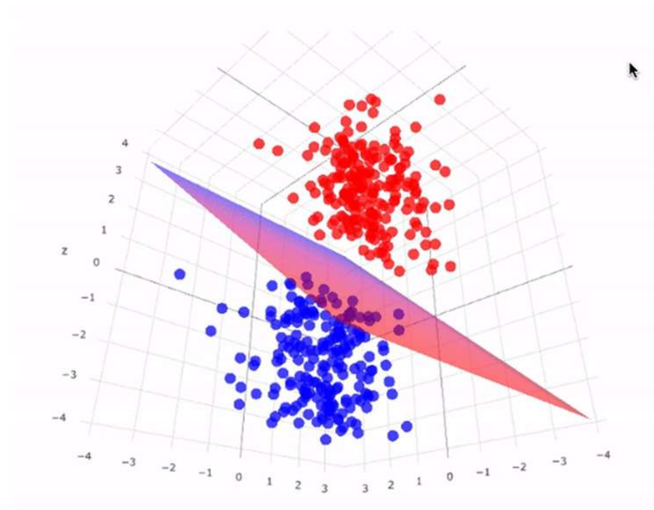
$$Activation(z) = \frac{1}{1 + e^{-z}}$$



Decision Boundary in a Hyper-dimension Space

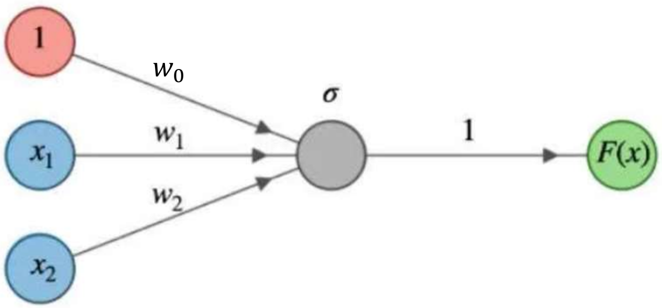
Activation function translates a hyperplane into a complex decision boundary in a higher dimension space





*Source: Deep Learning: Feed Forward Neural Networks (FFNNS) ---- Medium

Understand logistic Regression from a One-layer Neural Network



One-layer neural network can be formulated as

$$\mathrm{F}(x) = Activation(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)$$

Rewrite using vectorized form, we get

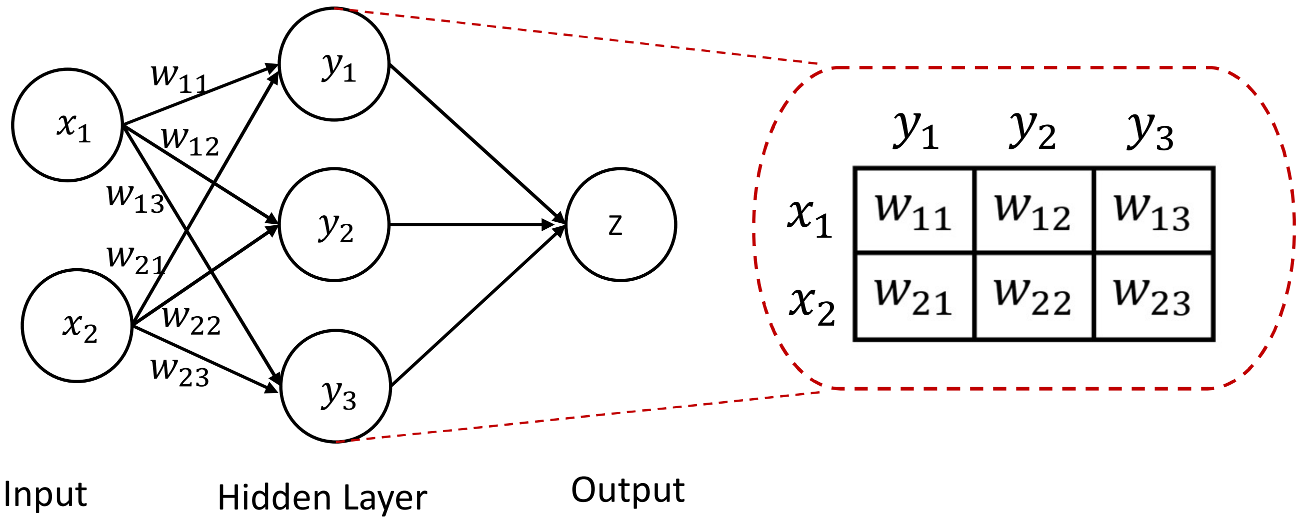
$$\mathrm{F}(x) = Activation\left(w^T X\right)$$

which has the same mathematical formula as a logistic regression.
Logistic regression uses below sigmoid activation function:

$$Activation(z) = \frac{1}{1 + e^{-z}}$$

while a neural network can have more activation function variation.
A logistic regression can be thought of a special case of one-layer neural network.

Two-layer Neural Networks



$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{11}x_1 + w_{21}x_2 \\ w_{12}x_1 + w_{22}x_2 \\ w_{13}x_1 + w_{23}x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$Y = Activation\left(w_x^T X\right)$$

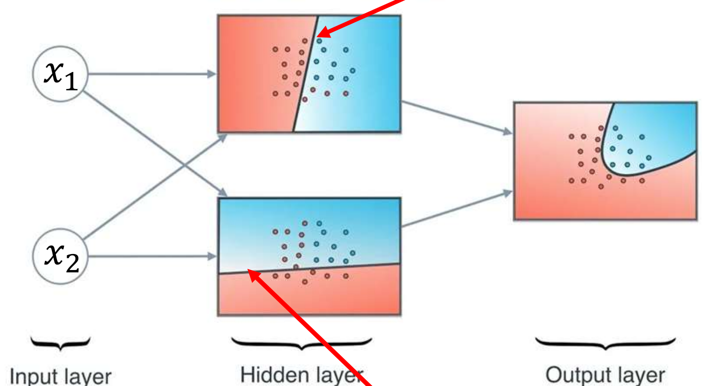
$$Z = Activation\left(w_y^T Y\right)$$

Adding a layer will add the complexity of the networks, but the general forward activation form is the same

The Necessity of One Hidden Layer

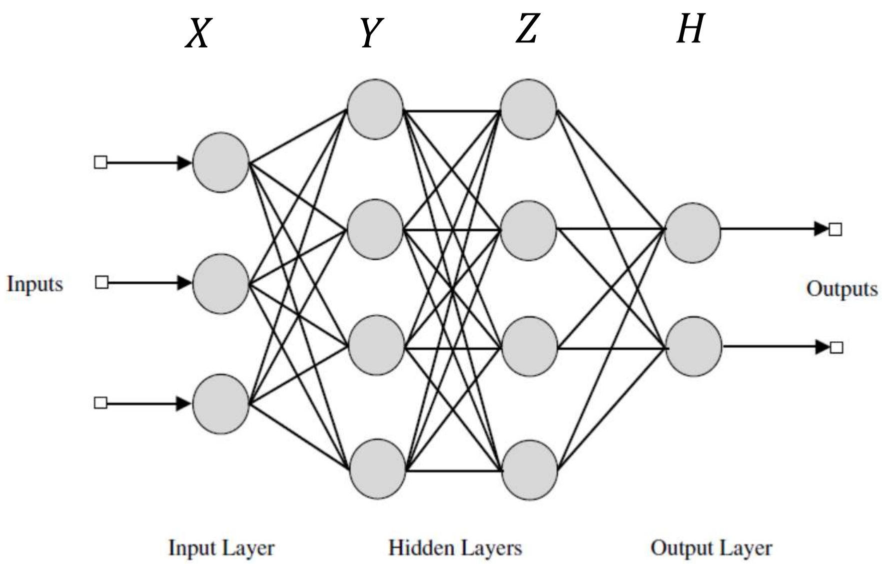
A hidden layer is needed since the decision boundaries could be nonlinear;
We use the combination of different decision boundaries to construct the final decision boundary;
Different decision boundaries require us to adopt hidden layers with different feature space.

$$w_{11}x_1 + w_{21}x_2 + w_{01} = 0$$



$$w_{21}x_1 + w_{22}x_2 + w_{02} = 0$$

Two Hidden-layer Neural Networks



$$X = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \quad w_x^T = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}$$

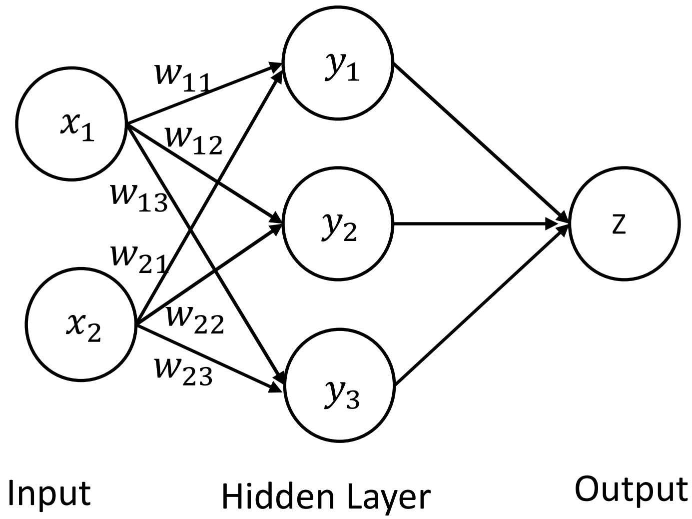
$$Y = Activation\left(w_x^T X\right)$$

$$Z = Activation\left(w_y^T Y\right)$$

$$H = Activation\left(w_z^T T\right)$$

With more hidden layers, we have more freedom to transform between different spaces.

Compare Similarity of back-propagation of error with the Forward Activation



The error of each neuron is proportional to its weight:

$$E(x_1) = \frac{w_{11}}{w_{11}+w_{21}}E(y_1) + \frac{w_{12}}{w_{12}+w_{22}}E(y_2) + \frac{w_{13}}{w_{13}+w_{23}}E(y_3)$$

$$E(x_2) = \frac{w_{21}}{w_{11}+w_{21}}E(y_1) + \frac{w_{22}}{w_{12}+w_{22}}E(y_2) + \frac{w_{23}}{w_{13}+w_{23}}E(y_3)$$

$$\Longrightarrow \begin{bmatrix} E(x_1) \\ E(x_2) \end{bmatrix} = \begin{bmatrix} \frac{w_{11}}{w_{11}+w_{21}} & \frac{w_{12}}{w_{12}+w_{22}} & \frac{w_{13}}{w_{13}+w_{23}} \\ \frac{w_{21}}{w_{11}+w_{21}} & \frac{w_{22}}{w_{12}+w_{22}} & \frac{w_{23}}{w_{13}+w_{23}} \end{bmatrix} \begin{bmatrix} E(y_1) \\ E(y_2) \\ E(y_3) \end{bmatrix}$$

Normalize above equation to simplify it as:

$$\begin{bmatrix} E(x_1) \\ E(x_2) \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \begin{bmatrix} E(y_1) \\ E(y_2) \\ E(y_3) \end{bmatrix}$$

Forward activation has same multiplier

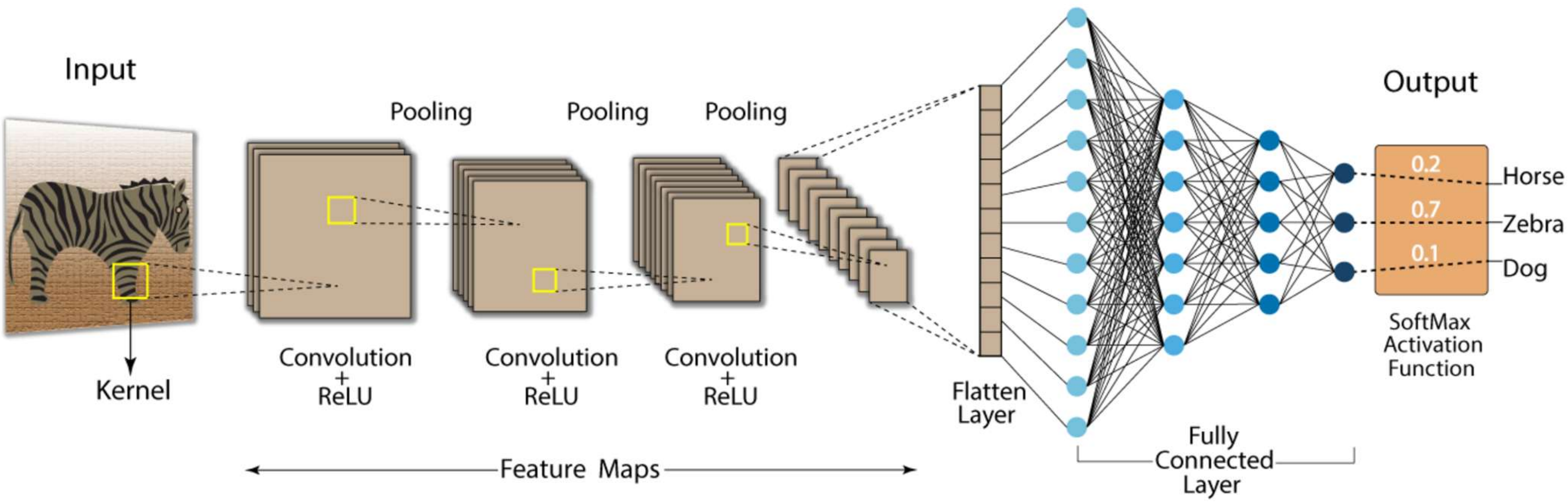
Below is a general equation of the back-propagation algorithm

$$Y = w^T X \longrightarrow E_n = w^T E_{n+1}$$

Adjust weight based on

$$w_i^n = w_i^n - \alpha \frac{\partial E_n}{\partial w_i^n}$$

Application of Convolutional Neural Network (CNN) in Computer Vision



- Convolutional Neural Networks (CNN) are a type of neural network
- widely used in images recognition, images classifications, and objects detections
- Computer reads images as pixels and expresses them as matrix
- Three basic components: Convolution layer, Flatten layer, Fully connected layer
- Filter (Kernel) is worked as the weight

Source: What is Convolutional Neural Network – CNN (Deep Learning)

Convolutional Neural Network

Input Image

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

$*$

Filter

w_1	w_2
w_3	w_4

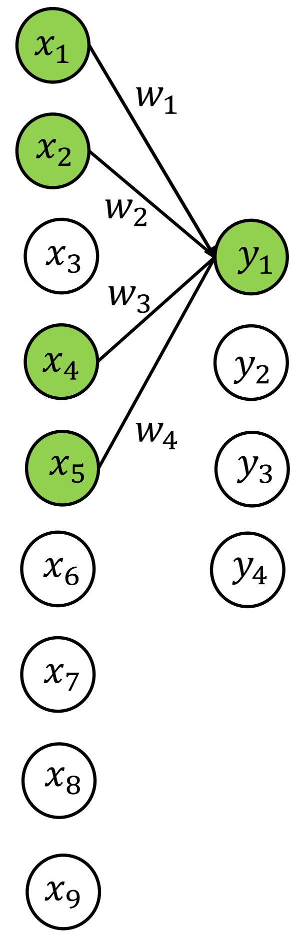
=

Feature Map

$\mathbf{y}_1$	$\mathbf{y}_2$
$\mathbf{y}_3$	y_4

The forward pass of the convolutional layer

$$y_1 = w_1 x_1 + w_2 x_2 + w_3 x_4 + w_4 x_5$$



Convolutional Neural Network

Input Image

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

*

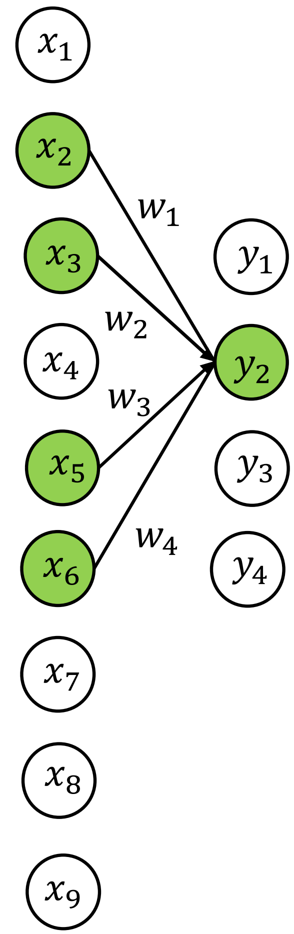
Kernel

w_1	w_2
w_3	w_4

=

Feature Map

$\mathbf{y}_1$	$\mathbf{y}_2$
$\mathbf{y}_3$	y_4



The forward pass of the convolutional layer

$$y_2 = w_1\, x_2 + w_2 x_3 + w_3 x_5 + w_4 x_6$$

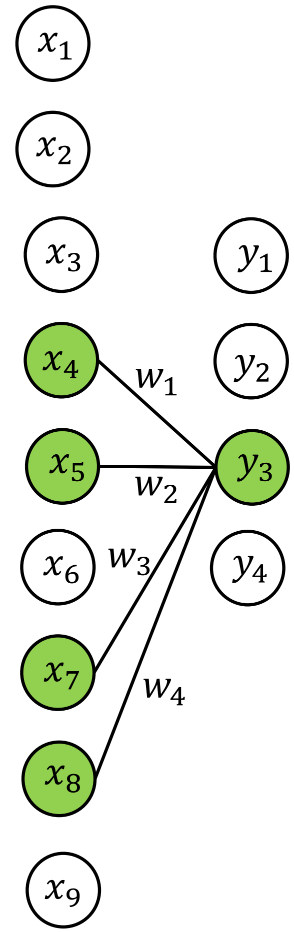
Convolutional Neural Network

Input Image Kernel Feature Map

$$\begin{array}{|c|c|c|}\hline x_1 & x_2 & x_3 \\ \hline x_4 & x_5 & x_6 \\ \hline x_7 & x_8 & x_9 \\ \hline \end{array} * \begin{array}{|c|c|}\hline w_1 & w_2 \\ \hline w_3 & w_4 \\ \hline \end{array} = \begin{array}{|c|c|}\hline \mathbf{y}_1 & \mathbf{y}_2 \\ \hline \mathbf{y}_3 & y_4 \\ \hline \end{array}$$

The forward pass of the convolutional layer

$$y_3 = w_1\ x_4 + w_2 x_5 + w_3 x_7 + w_4 x_8$$



Convolutional Neural Network

Input Image, Kernel, Feature Map

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix} * \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 \\ \mathbf{y}_3 & y_4 \end{bmatrix}$$

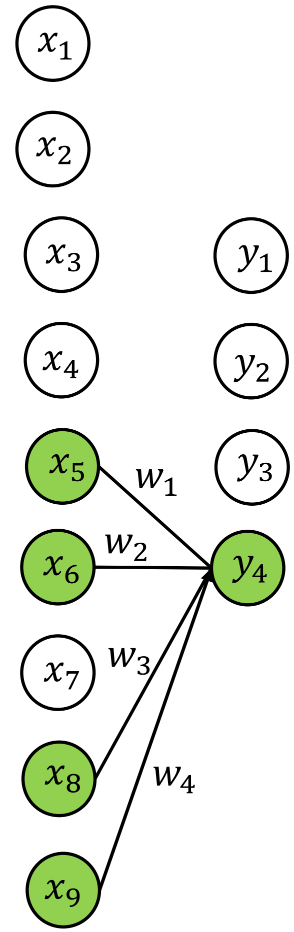
The forward pass of the convolutional layer

$$y_4 = w_1\ x_5 + w_2 x_6 + w_3 x_8 + w_4 x_9$$

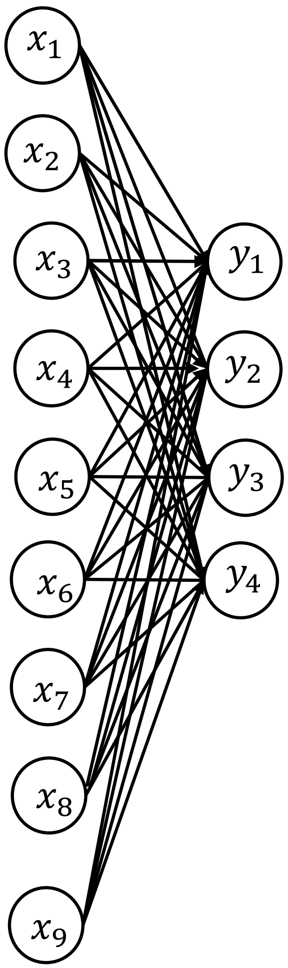
If we generalize the forward pass:

$$\mathrm{Y} = conv(w * X)$$

Convolutional Multiply

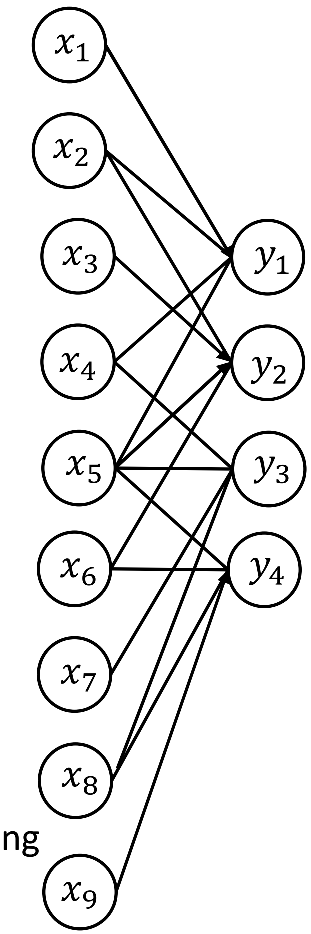


Comparison of Neural Networks vs Convolutional Neural Networks



	y_1	y_2	y_3	y_4
x_1	w_{11}	w_{12}	w_{13}	w_{14}
x_2	w_{21}	w_{22}	w_{23}	w_{24}
x_3	w_{31}	w_{32}	w_{33}	w_{34}
x_4	w_{41}	w_{42}	w_{43}	w_{44}
x_5	w_{51}	w_{52}	w_{53}	w_{54}
x_6	w_{61}	w_{62}	w_{63}	w_{64}
x_7	w_{71}	w_{72}	w_{73}	w_{74}
x_8	w_{81}	w_{82}	w_{83}	w_{84}
x_9	w_{91}	w_{92}	w_{93}	w_{94}

y_1	y_2	y_3	y_4
w_1			
w_2	w_1		
0	w_2		
w_3	0	w_1	
w_4	w_3	w_2	w_1
	w_4	0	w_2
		w_3	0
		w_4	w_3
			w_4



$Y = Activation(w^T X)$

- Difficult to understand
- More weight parameter
- Global impact from input

$\mathrm{Y} = conv(w * X)$

- Can be visually understandable
- Fewer weight due to weight sharing
- Localized impact from input

Convolutional Neural Network Back-propagation

Input Image		
x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

$*$

Kernel	
w_1	w_2
w_3	w_4

$=$

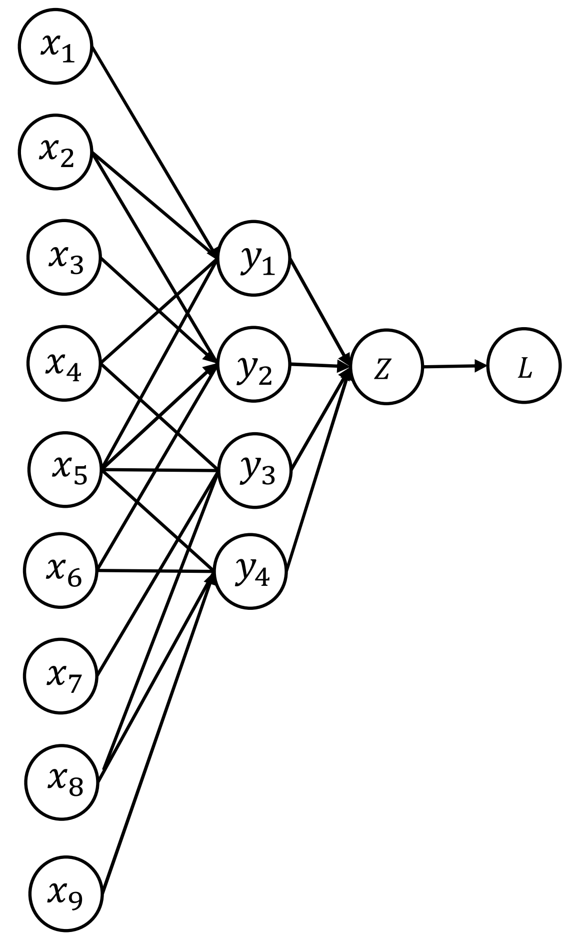
Feature Map	
$\mathbf{y}_1$	$\mathbf{y}_2$
$\mathbf{y}_3$	y_4

Weight adjustment is:

$$w_i^n = w_i^{n-1} - \alpha \frac{\partial L}{\partial w_i}$$

The loss gradient based on the chain rule

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial w_i} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial w_i} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial w_i} + \frac{\partial L}{\partial y_3}\frac{\partial y_3}{\partial w_i} + \frac{\partial L}{\partial y_4}\frac{\partial y_4}{\partial w_i}$$



Convolutional Neural Network Back-propagation

Convolutional function of forward pass

$$y_1 = w_1\, x_1 + w_2 x_2 + w_3 x_4 + w_4 x_5$$

$$y_2 = w_1\, x_2 + w_2 x_3 + w_3 x_5 + w_4 x_6$$

$$y_3 = w_1\, x_4 + w_2 x_5 + w_3 x_7 + w_4 x_8$$

$$y_4 = w_1\, x_5 + w_2 x_6 + w_3 x_8 + w_4 x_9$$

Local gradient w.r.t the filter

$\frac{\partial y_1}{\partial w_1}=x_1$	$\frac{\partial y_1}{\partial w_2}=x_2$	$\frac{\partial y_1}{\partial w_3}=x_4$	$\frac{\partial y_1}{\partial w_4}=x_5$
$\frac{\partial y_2}{\partial w_1}=x_2$	$\frac{\partial y_2}{\partial w_2}=x_3$	$\frac{\partial y_2}{\partial w_3}=x_5$	$\frac{\partial y_2}{\partial w_4}=x_6$
$\frac{\partial y_3}{\partial w_1}=x_4$	$\frac{\partial y_3}{\partial w_2}=x_5$	$\frac{\partial y_3}{\partial w_3}=x_7$	$\frac{\partial y_3}{\partial w_4}=x_8$
$\frac{\partial y_4}{\partial w_1}=x_5$	$\frac{\partial y_4}{\partial w_2}=x_6$	$\frac{\partial y_4}{\partial w_3}=x_8$	$\frac{\partial y_4}{\partial w_4}=x_9$

Loss gradient w.r.t the filter

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_1}x_1 + \frac{\partial L}{\partial y_2}x_2 + \frac{\partial L}{\partial y_3}x_4 + \frac{\partial L}{\partial y_4}x_5$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial y_1}x_1 + \frac{\partial L}{\partial y_2}x_3 + \frac{\partial L}{\partial y_3}x_5 + \frac{\partial L}{\partial y_4}x_6$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial y_1}x_4 + \frac{\partial L}{\partial y_2}x_5 + \frac{\partial L}{\partial y_3}x_7 + \frac{\partial L}{\partial y_4}x_8$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial y_1}x_5 + \frac{\partial L}{\partial y_2}x_6 + \frac{\partial L}{\partial y_3}x_8 + \frac{\partial L}{\partial y_4}x_9$$

Rewrite using convolutional operator

$$\begin{array}{|c|c|}\hline \frac{\partial L}{\partial w_1} & \frac{\partial L}{\partial w_2} \\ \hline \frac{\partial L}{\partial w_3} & \frac{\partial L}{\partial w_4} \\ \hline \end{array} = \text{Conv} \left(\begin{array}{|c|c|c|}\hline x_1 & x_2 & x_3 \\ \hline x_4 & x_5 & x_6 \\ \hline x_7 & x_8 & x_9 \\ \hline \end{array} * \begin{array}{|c|c|}\hline \frac{\partial L}{\partial y_1} & \frac{\partial L}{\partial y_2} \\ \hline \frac{\partial L}{\partial y_3} & \frac{\partial L}{\partial y_4} \\ \hline \end{array} \right)$$

$$\frac{\partial L}{\partial w} = \text{Conv}\ \left(\frac{\partial L}{\partial Y} * \text{X} \right)$$

Conclusion

	Similarity	
Logistic regression vs. Neural Network	Logistic Regression F$(x, w) = Activation\left(w^T x\right) = \frac{1}{1 + e^{-w^T x}}$	Neural Network F$(x) = Activation\left(w^T x\right)$
Neural Network back-propagation vs. forward activation	back-propagation $E_{n-1} = w^T E_n$	forward activation Y= $w^T X$
Neural Network vs. CNN	Neural Network F$(x) = Activation\left(w^T X\right)$	CNN forward activation Y $= conv(w * X)$
CNN back-propagation vs. forward activation	CNN back -propagation $\frac{\partial L}{\partial w}$ = Conv ($\frac{\partial L}{\partial Y}$ *X)	CNN forward activation Y $= conv(w * X)$

Thanks!