Deep Learning Crash Course

www.deeplearningcrashcourse.org

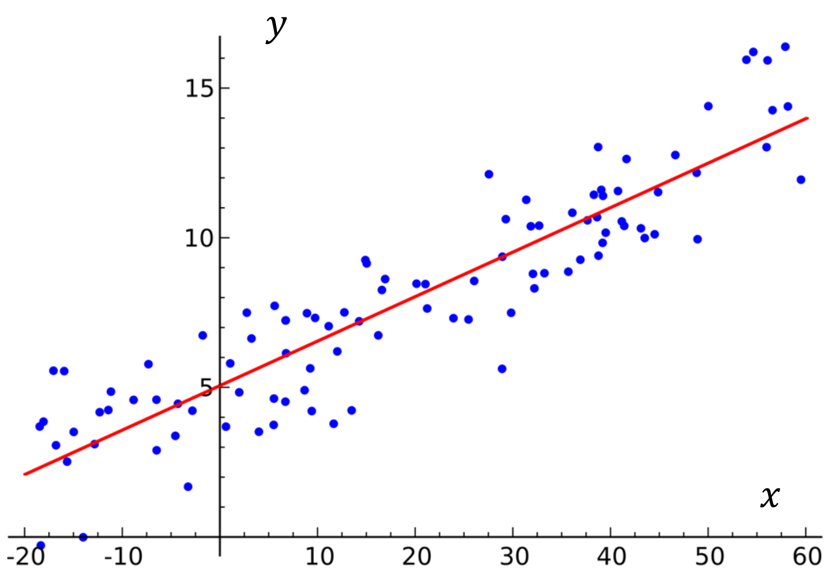
Hui Xue

Fall 2022

Lecture 1

- Deep learning as a data driven approach
- Binary and multi-class classification
- Multi-layer perceptron (MLP)

Linear regression



https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_regression.svg

Model:

$$y = wx + b$$

Loss:

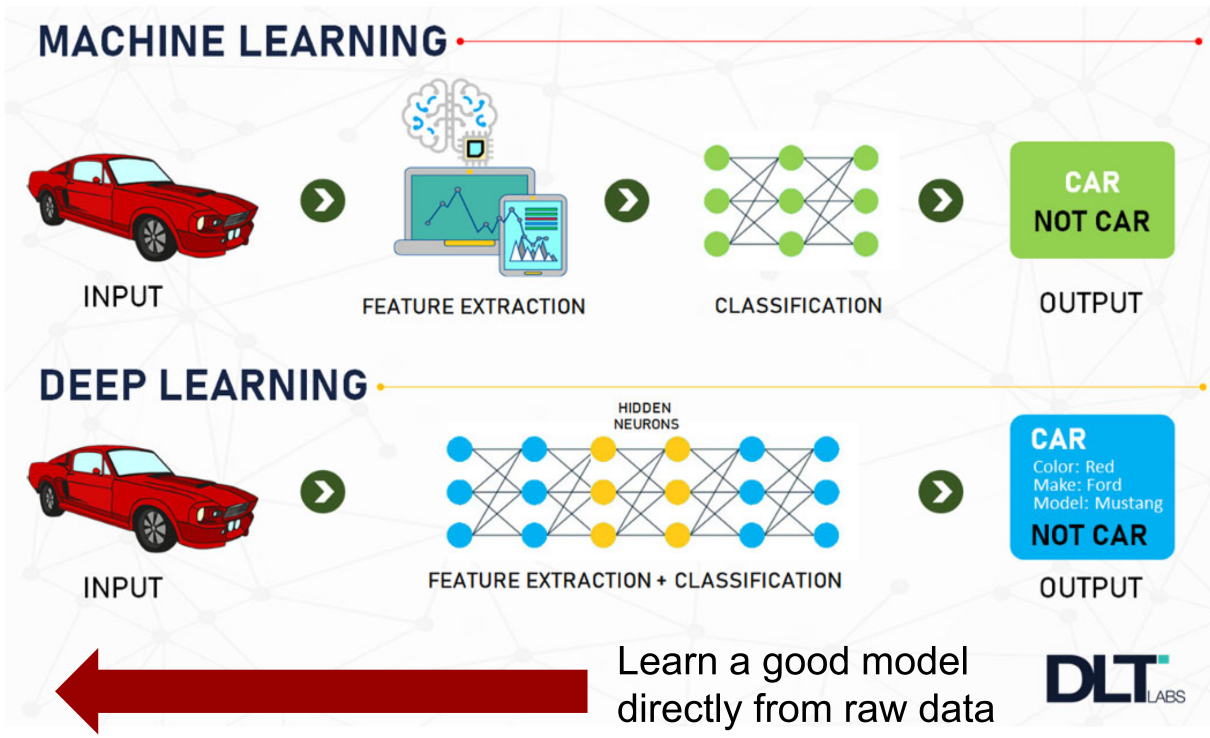
$$\ell(w, b) = \sum_{i=0}^{N-1} [y^{(i)} - (wx^{(i)} + b)]^2$$

Optimization:

$$\min_{w,b} \ell(w, b)$$

Minimize the empirical risk, computed on the measured data sample $(x^{(i)}, y^{(i)})$

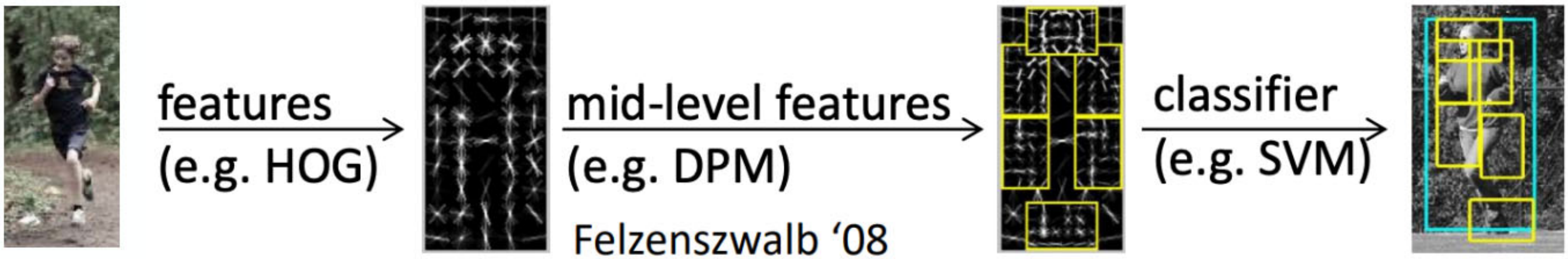
Deep Learning model is data-driven



Learn a good model directly from raw data

https://dltlabs.medium.com/understanding-machine-learning-deep-learning-f5aa95264d61

Feature engineering



histogram of oriented gradients

- Human design and build feature extractor – very hard to scale, e.g. to 2000 object classes
- Model works on the feature vectors, instead of original data
- Limited by the human insights and amount of data

Object Detection with Discriminatively Trained Part Based Models. https://cs.brown.edu/people/pfelzens/papers/lsvm-pami.pdf

Data driven detection

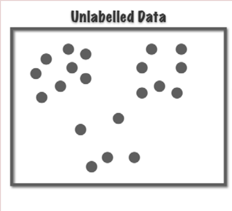
https://youtu.be/VOC3huqHrss

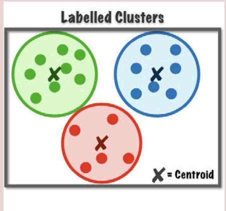
- Start from image to object detection (class and bounding box)
- End-to-end training, without human crafted features
- So powerful, can be done in real-time

Deep Learning Landscapes

Unsupervised Learning

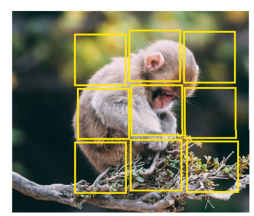
- Unlabeled X
- Learn representation in data

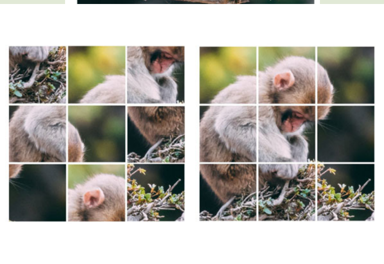




Self-supervised Learning

- Unlabeled X
- Generate self-supervisory signal
- Learn effective representation of data to help downstream apps





Supervised Learning

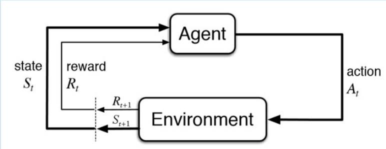
- Labelled X and Y
- Learn model: X->Y
- Make prediction

Is a cat?

1

Reinforcement Learning

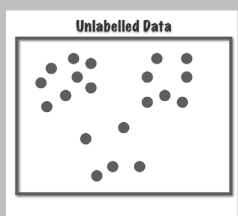
- Learn actions to reach the goal given an environment

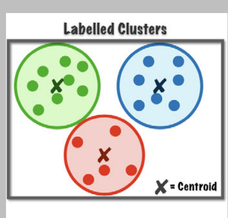


Part 1

Unsupervised Learning

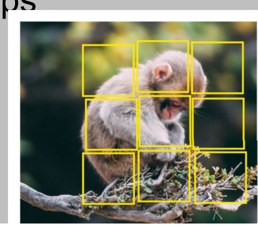
- Unlabeled X
- Learn representation in data

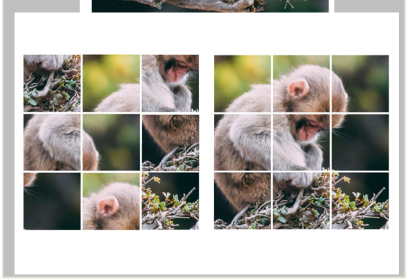




Self-supervised Learning

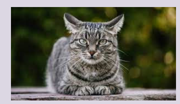
- Unlabeled X
- Generate self-supervisory signal
- Learn effective representation of data to help downstream apps





Supervised Learning

- Labelled X and Y
- Learn model: X->Y
- Make prediction

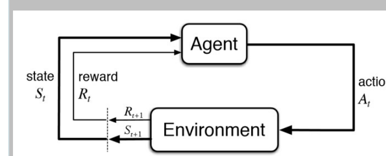


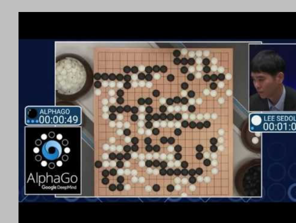
Is a cat?

1

Reinforcement Learning

- Learn actions to reach the goal given an environment





Heart finding problem

A running example : heart finding from MRI images

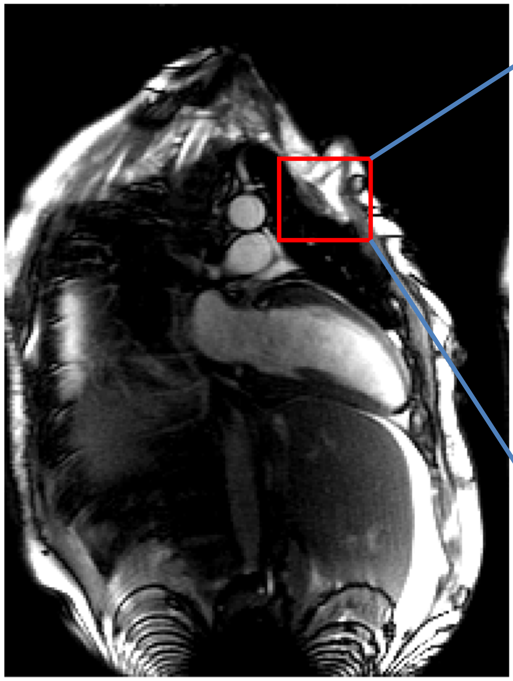
$$\boldsymbol{X} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} =$$

$\boldsymbol{Y} = 1$, if imaging the heart

$\boldsymbol{Y} = \mathbf{0}$

Computer sees the number, not content

What computer sees in this image:



967	965	967	975	984	993	1004	1015	1027	1041
1003	985	975	988	995	1001	1011	1021	1031	1042
838	944	1025	993	1001	1017	1017	1027	1037	1043
685	891	1053	994	1004	1026	1019	1029	1037	1042
737	912	1050	998	1006	1024	1019	1027	1033	1040
776	951	1074	1002	1006	1025	1018	1025	1030	1042
785	882	994	1010	1018	1024	1024	1027	1033	1055
825	789	886	1052	1043	1021	1040	1037	1046	1083
870	829	928	1092	1069	1041	1062	1059	1077	1131
937	932	956	1030	1099	1111	1099	1109	1143	1214
1068	1063	1014	1032	1179	1225	1193	1214	1262	1351
1240	1235	1195	1213	1330	1364	1336	1360	1419	1524
1390	1385	1350	1373	1481	1496	1462	1494	1567	1690
1528	1519	1488	1489	1546	1584	1593	1627	1713	1860
1708	1693	1681	1655	1625	1692	1769	1799	1892	2060
1921	1905	1892	1864	1830	1888	1964	2003	2107	2281
2106	2090	2072	2044	2017	2071	2145	2199	2317	2495
2256	2240	2219	2194	2177	2227	2303	2378	2510	2688
2404	2390	2355	2347	2385	2416	2459	2558	2702	2873
2555	2541	2501	2499	2561	2600	2642	2749	2896	3062
2724	2707	2672	2665	2700	2746	2825	2954	3102	3257
2895	2876	2846	2837	2850	2887	2993	3156	3303	3442
3044	3024	2995	2992	3014	3063	3176	3337	3480	3611
3202	3181	3153	3159	3196	3258	3372	3525	3656	3773
3358	3337	3313	3325	3367	3434	3545	3687	3815	3926
3491	3471	3452	3467	3514	3593	3689	3804	3944	4062
3635	3615	3598	3617	3670	3753	3843	3945	4082	4195
3771	3752	3739	3764	3821	3904	3996	4097	4220	4320
3881	3866	3860	3893	3955	4039	4129	4226	4343	4436
3996	3987	3991	4030	4095	4179	4266	4359	4471	4556
4121	4118	4132	4176	4241	4322	4409	4498	4591	4668
4248	4252	4273	4319	4384	4462	4548	4631	4705	4775
4376	4386	4414	4461	4525	4600	4682	4760	4829	4893
4499	4514	4548	4597	4660	4734	4811	4886	4952	5009

Model as a mapping function

First idea : design a mapping function to map image pixel values to the probability

$$y = \boldsymbol{f}(\boldsymbol{X})$$

y : a single floating value

lower case, regular

X : Nx1 vector

N=HxW, number of pixels

upper case, bold

A scalar for the probability where heart was imaged

Linear mapping, weights and bias

Start with a linear mapping

$$z = \boldsymbol{W}\boldsymbol{X} + b$$

z : 1x1 scalar

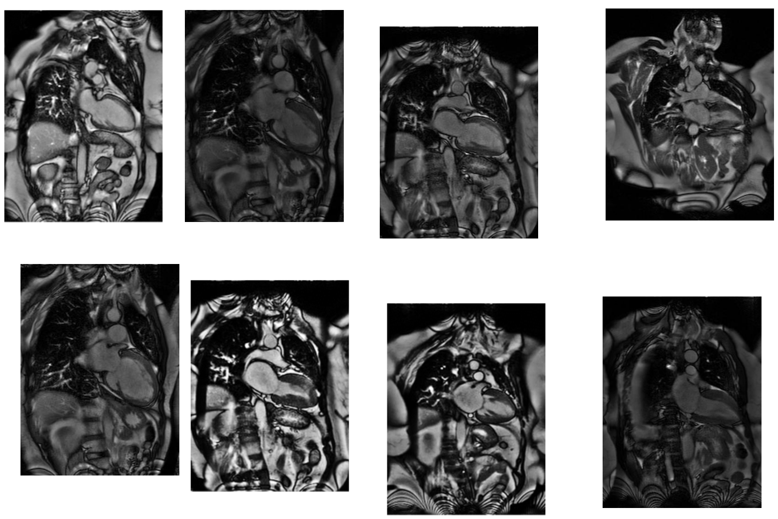
W : 1xN **weights** matrix

b : 1x1 **bias**

Process a ***Batch***

We can process multiple images in one pass

B=8, a **batch**



W and b are the same to all images

X : NxB

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{X} + b$$

Z : 1xB scalar

score

W : 1xN **weights** matrix

b : 1x1 **bias**

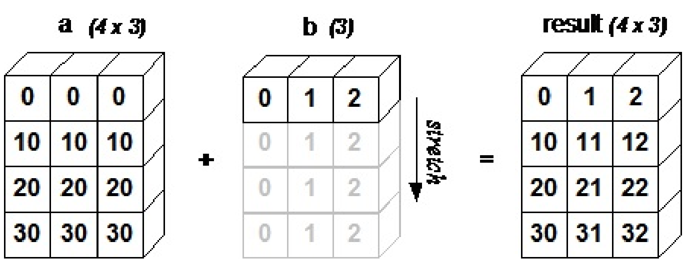
Broadcasting

$$Z = WX \quad + \quad b$$

W : 1xN
X : NxB

1x1 scalar

So Z is 1xB



Add same b to every element of **Z**

Broadcasting : repeat the smaller array to make its shape compatible to larger array

- starts with the trailing (i.e. rightmost) dimensions and works its way left
- **Repeat any dimension of 1**

A (4d array): 8 x 1 x 6 x 1

B (3d array): 7 x 1 x 5

Result (4d array): 8 x 7 x 6 x 5

https://numpy.org/doc/stable/user/basics.broadcasting.html

Enforce probability

$$z = WX + b$$

z is not a value [0 1], so it is not a probability

$$a = f(z) = f(WX + b)$$

f is applied element-wise

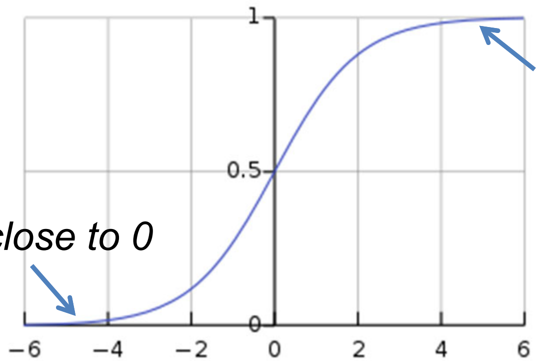
A choice of f is the sigmoid function

Sigmoid function $\sigma(x) = \dfrac{1}{1 + exp(-x)}$

$$\frac{d\sigma(x)}{dx} = \sigma(x)[1 - \sigma(x)]$$

Active range -5 to 5

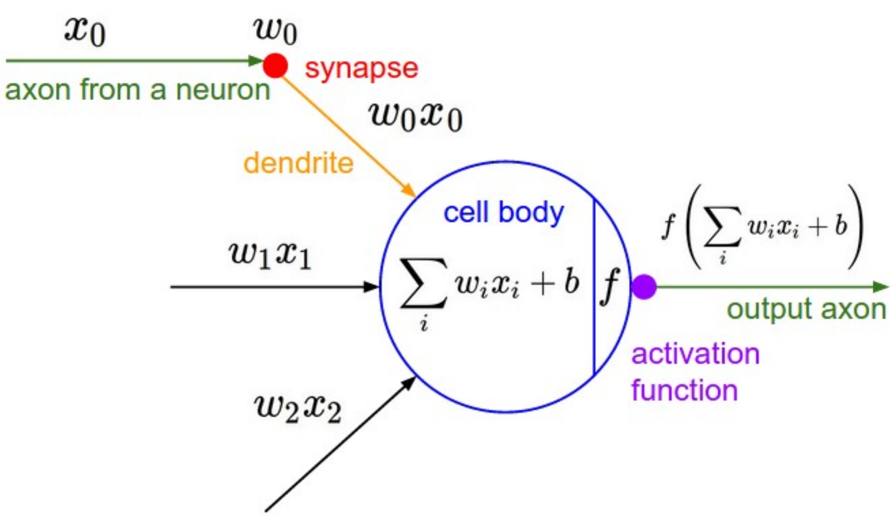
Accept inputs from -∞ to + ∞



>5, close to 1

<-5, close to 0

This is a "neuron" in neural network



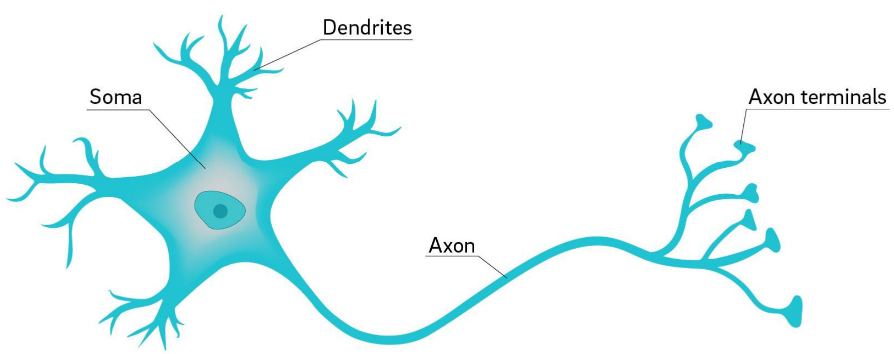
Neuron = linear mapping + nonlinear activation

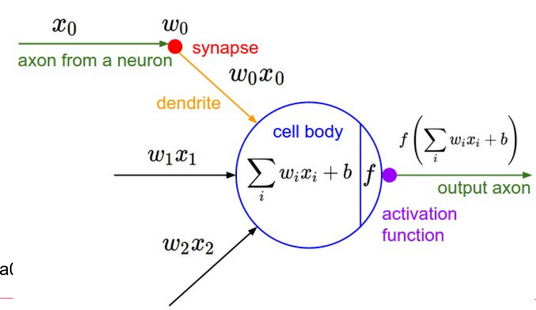
- Historically, inspired by biological neuron
- Building block of DL
- **Logistic regression** if sigmoid nonlinear activation is used

Picture is from https://cs231n.github.io/neural-networks-1/

This concept was invented in 1958

Neuron





https://medium.datadriveninvestor.com/introduction-to-neural-networks-a(

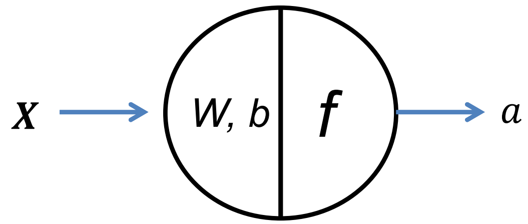
The New Yorker, December 6, 1958 P. 44

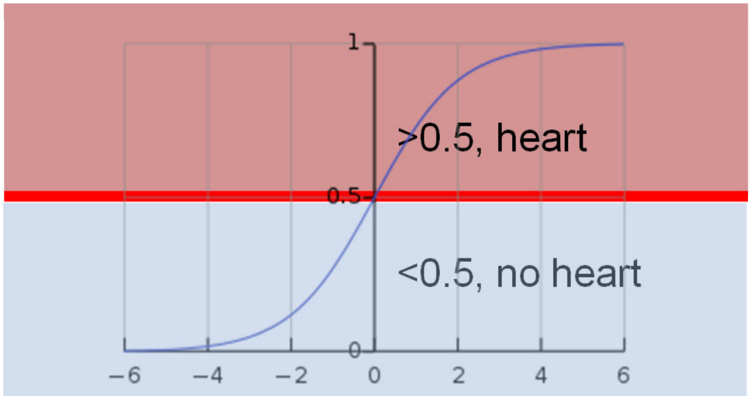
Talk story about the perceptron, a new electronic brain which hasn't been built, but which has been successfully simulated on the I.B.M. 704. Talk with Dr. Frank Rosenblatt, of the Cornell Aeronautical Laboratory, who is one of the two men who developed the prodigy; the other man is Dr. Marshall C. Yovits, of the Office of Naval Research, in Washington. Dr. Rosenblatt defined the perceptron as the first non-biological object which will achieve an organization o its external environment in a meaningful way. It interacts with its environment, forming concepts that have not been made ready for it by a human agent. If a triangle is held up, the perceptron's eye picks up the image & conveys it along a random succession of lines to the response units, where the image is registered. It can tell the difference betw. a cat and a dog,

https://www.newyorker.com/magazine/1958/12/06/rival-2

Binary classification

$$a = f(z) = f(\boldsymbol{WX} + b)$$

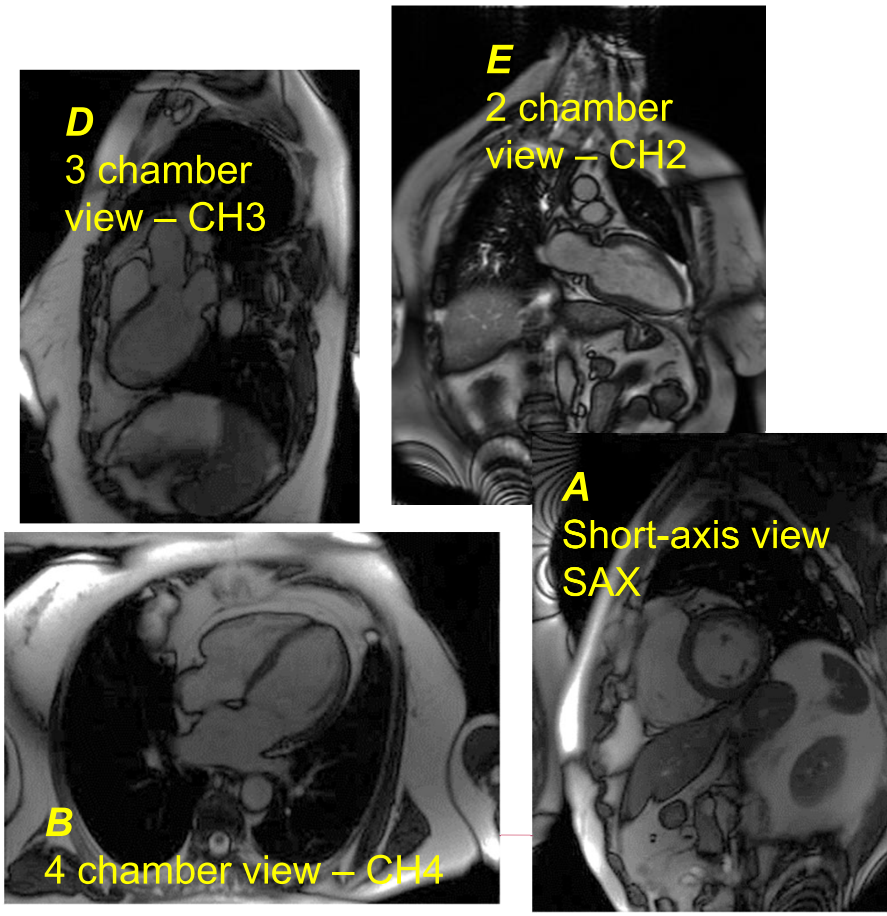


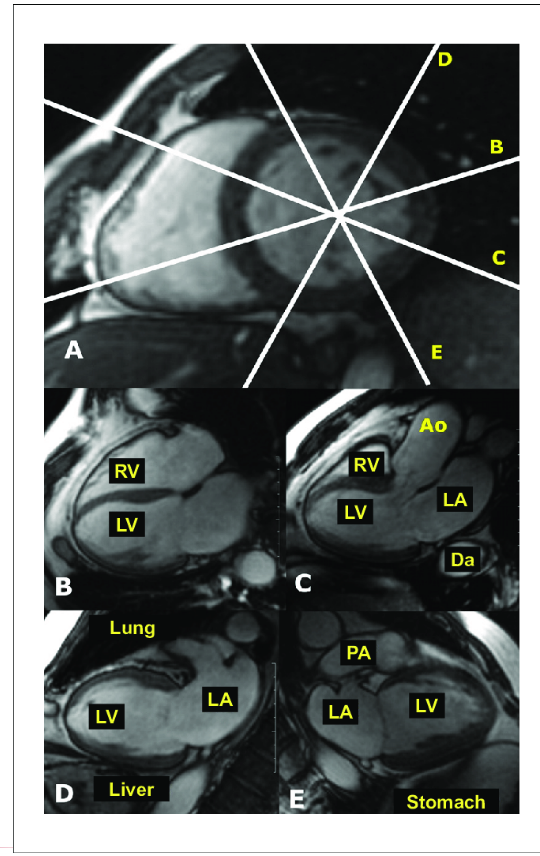


- Only two output status “Yes” or “No”
- One scalar is needed
- Binary decision

Multi-class classification

Cardiac view detection

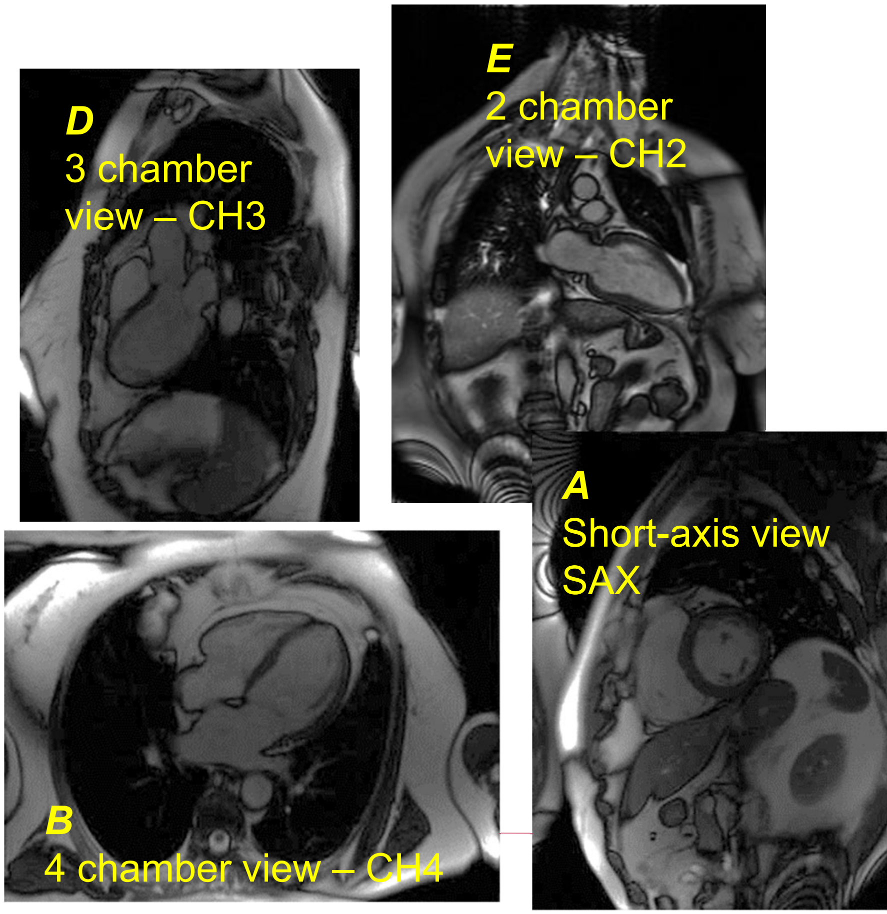




https://www.scielo.br/scielo.php?script=sci_arttext&pid=S0066-782X2010001600014&lng=pt&nrm=iso&tlng=pt

Multi-class classification

Cardiac view detection



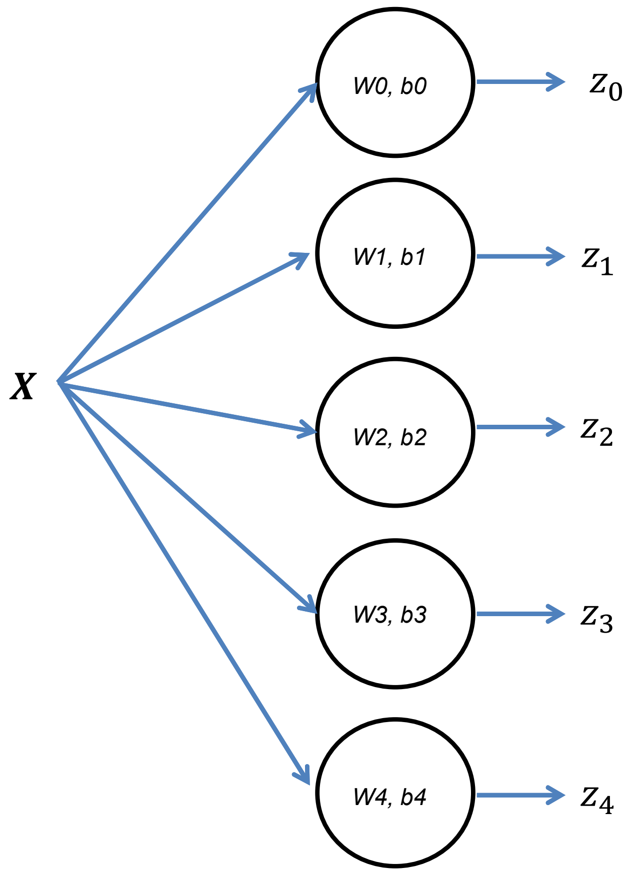
Given an image, decide which imaging view it is

$$Y = \begin{cases} CH3, & 1 \quad 0 \quad 0 \quad 0 \quad 0 \\ CH2, & 0 \quad 1 \quad 0 \quad 0 \quad 0 \\ CH4, & 0 \quad 0 \quad 1 \quad 0 \quad 0 \\ SAX, & 0 \quad 0 \quad 0 \quad 1 \quad 0 \\ Other, & 0 \quad 0 \quad 0 \quad 0 \quad 1 \end{cases}$$

Now we need 5 numbers for K=5 classes

This type of encoding y is “one-hot encoding”

Multi-class classification



Generate 5 numbers as the score for each class

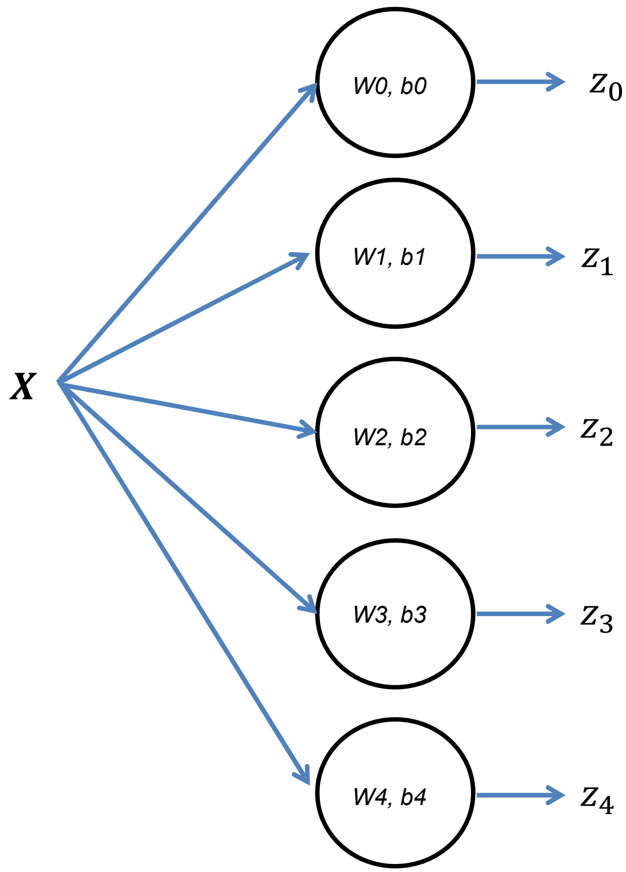
$$\boldsymbol{Z} = [z_0, z_1, z_2, z_3, z_4]$$

X : NxB, images

W0, W1, …, W4: 1XN weights

b0, b1, …, b4: 1x1 bias

Multi-class classification



X : NxB, images

W0, W1, …, W4: 1XN weights

b0, b1, …, b4: 1x1 bias

We can put all weights in one matrix and bias

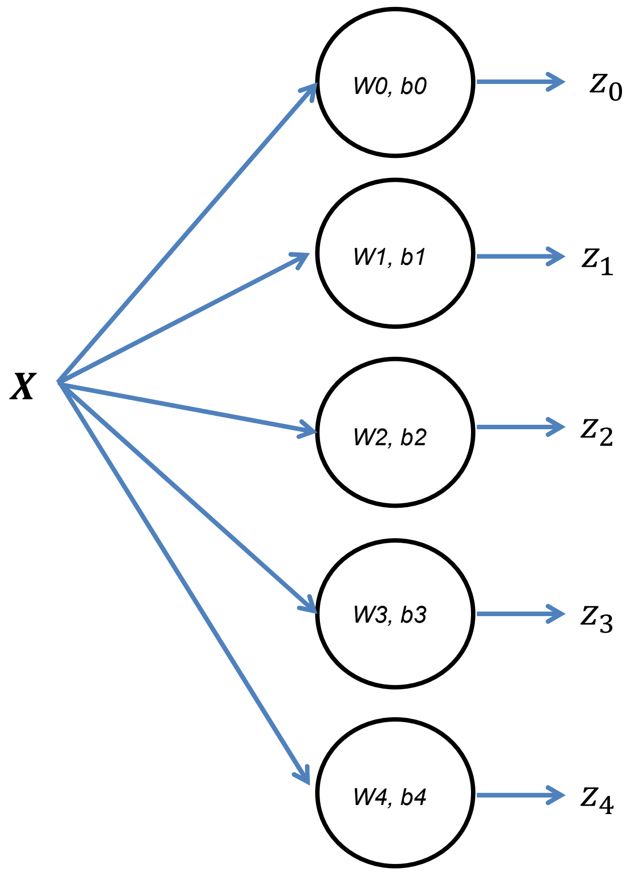
$$\boldsymbol{W} = \begin{bmatrix} - & W0 & - \\ - & W1 & - \\ - & W2 & - \\ - & W3 & - \\ - & W4 & - \end{bmatrix}$$

KxN matrix, every row for each class

$$\boldsymbol{b} = \begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \\ b4 \end{bmatrix}$$

Kx1 vector

Multi-class classification



Matrix representation:

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{X} + \boldsymbol{b}$$

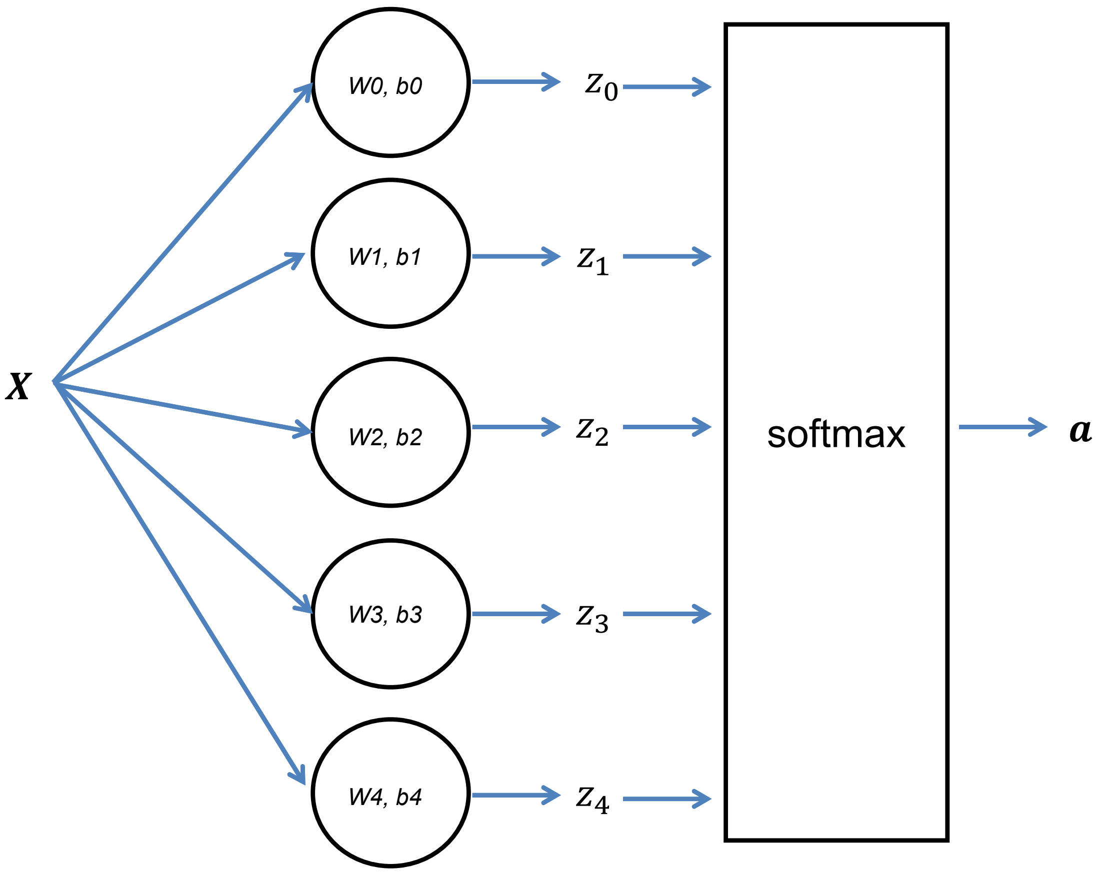
X: NxB, input images

W: KxN, weights

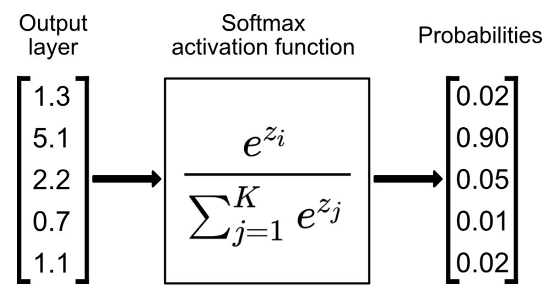
b: Kx1, bias

Z: KxB, score

Softmax



$$softmax(\boldsymbol{Z}) = \frac{\exp(z_i)}{\sum_{j=0}^{K-1} \exp(z_j)}$$



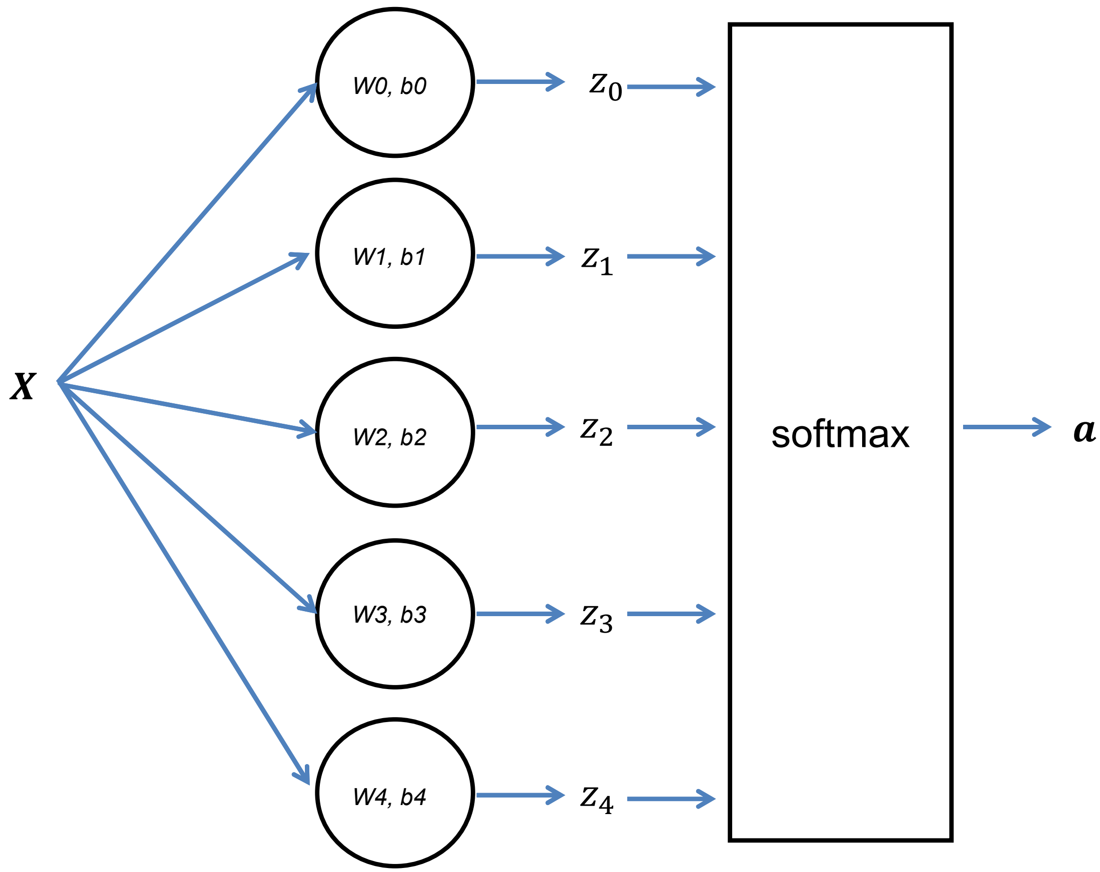
Softmax trick:

$$softmax(\boldsymbol{Z} - c) = softmax(\boldsymbol{Z})$$

compute $softmax(\boldsymbol{Z})$ ***as*** $softmax(\boldsymbol{Z} - max(\boldsymbol{Z}))$

Better numeric stability

Logits



When score ***Z*** is used as inputs to softmax, it is also called “logits”

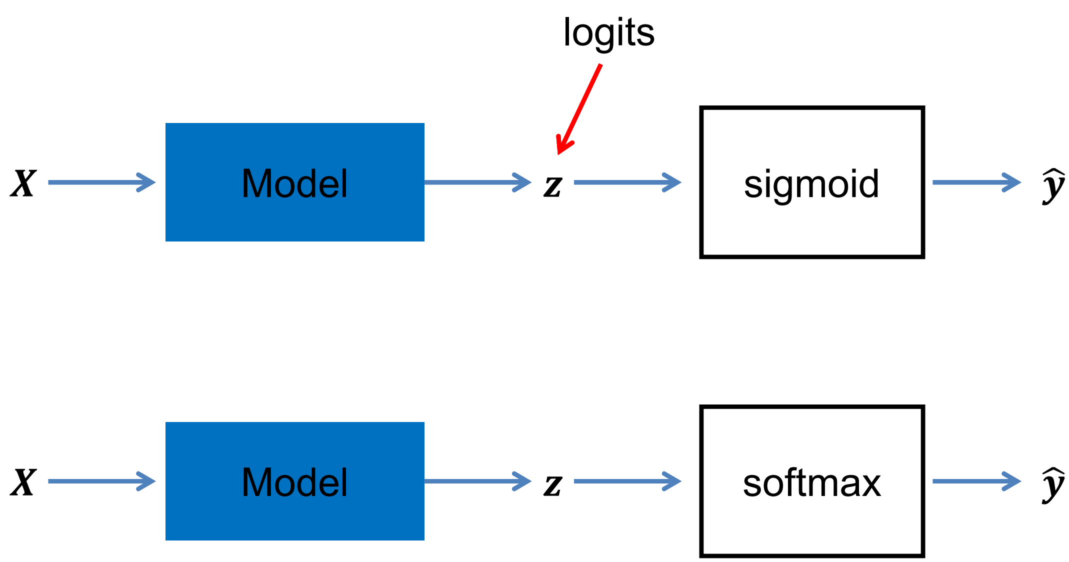
Lq#wkh#p dfklqh#ohduqlqj/#orjlw#fdq#eh#
ghilqhg#dv#d#yhfwru#ri#udz#suhglfwlrqv#
wkdw#duh#lqsxwwhg#lqwr#qrupdol}dwlrq#
ixqfwlrq#wr#jhqhudwh#suredelolwlhv1

$$z = logits(\boldsymbol{a}) = \log(\frac{a}{1-a})$$

$$a \in [0,1], logits \in (-\infty, +\infty)$$

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

Logits



- The term "logits" is often used in DL frameworks
- Same concept for binary classification
- When models are illustrated, often the sigmoid/softmax are included in the model plotting
- Loss is computed on $L(\mathbf{y}, \hat{\boldsymbol{y}})$; every sample is $(\boldsymbol{x}, \boldsymbol{y})$; model outputs $\hat{\boldsymbol{y}}$

Binary vs. multi-class classification

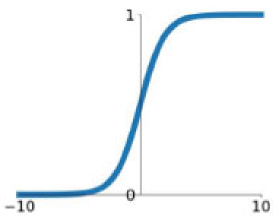
Binary	Multi-class
$\boldsymbol{a} = \boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{X} + b)$	$\boldsymbol{a} = \boldsymbol{softmax}(\boldsymbol{W}\boldsymbol{X} + b)$
X : NxB W : 1xN b : 1x1 a : 1xB	X : NxB W : KxN b : Kx1 a : KxB
Apply sigmoid element-wise	Apply softmax along class K

We used one layer of linear mapping and one nonlinear activation function

Activation functions

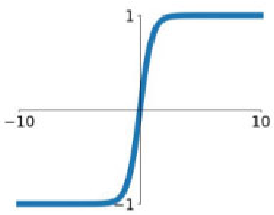
Sigmoid

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



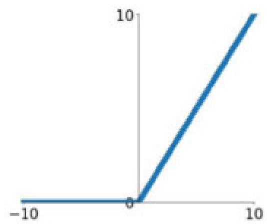
tanh

$\tanh(x)$



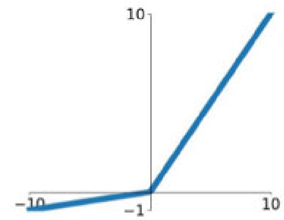
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

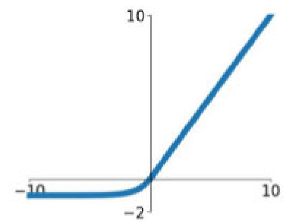


Maxout

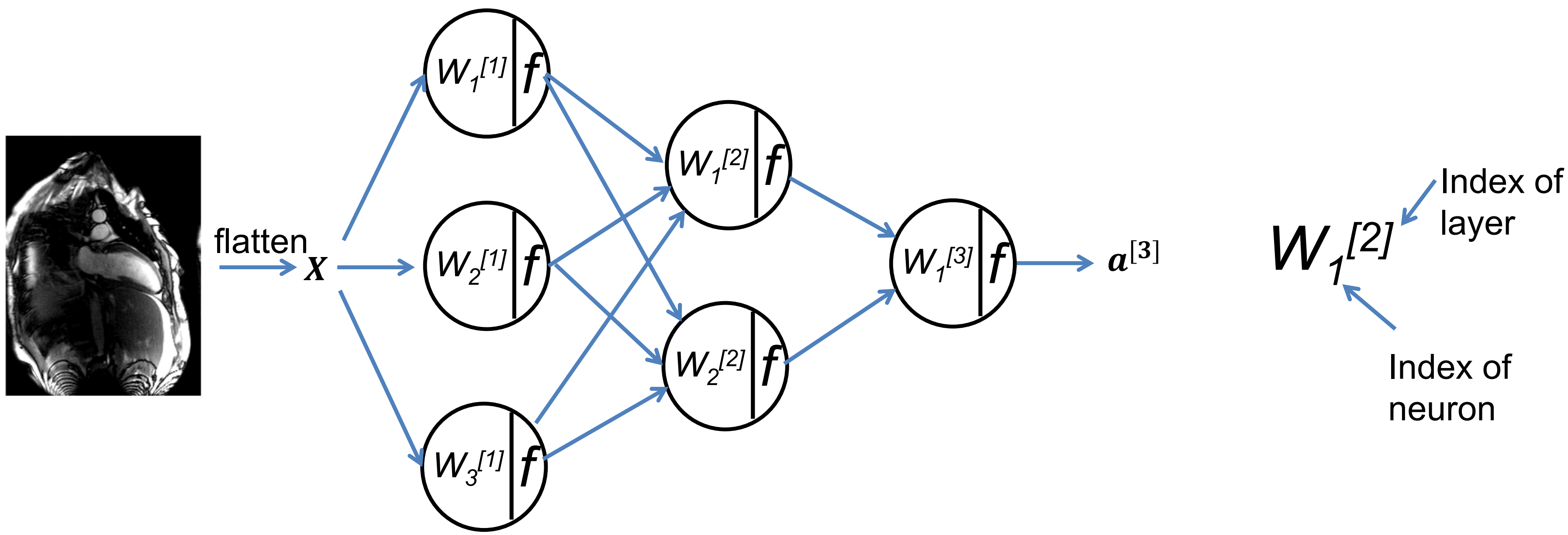
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

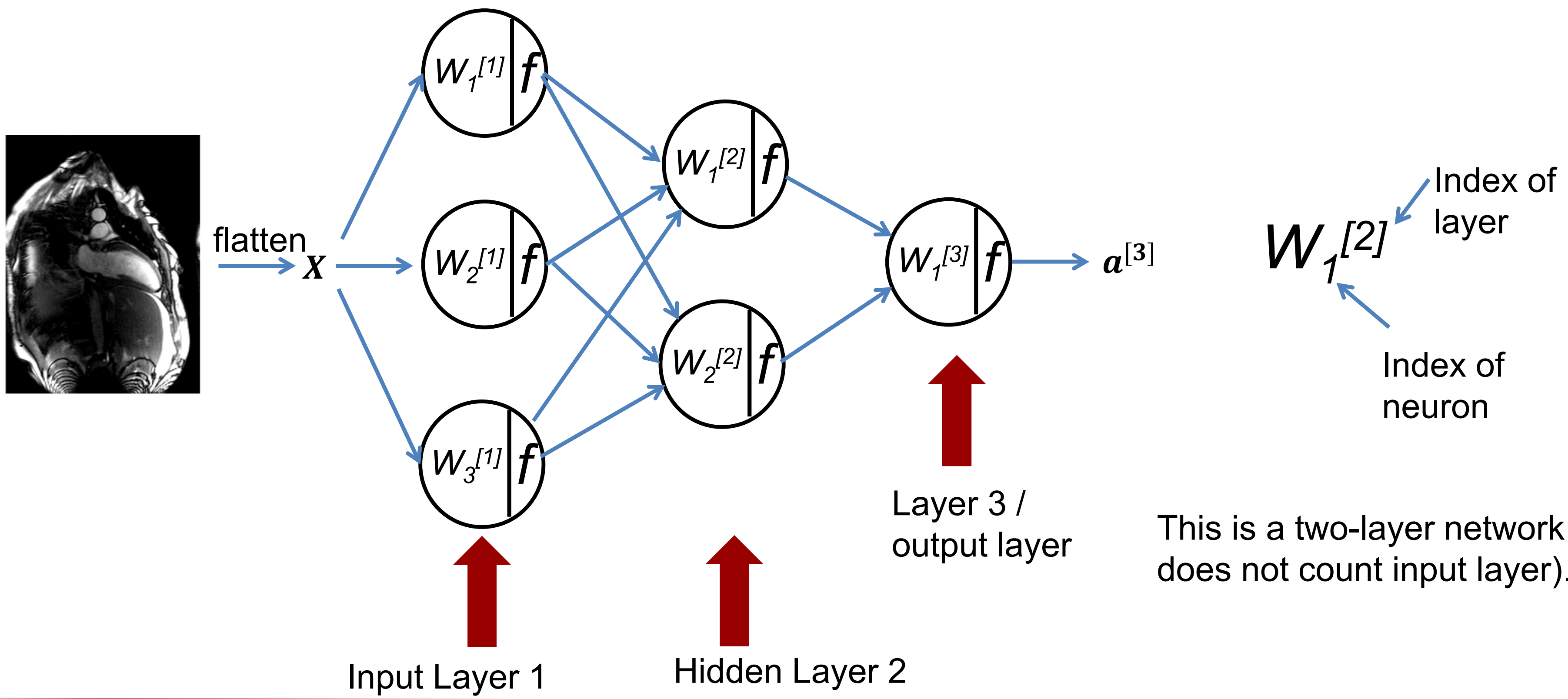
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Multi-layer Perceptron (MLP)

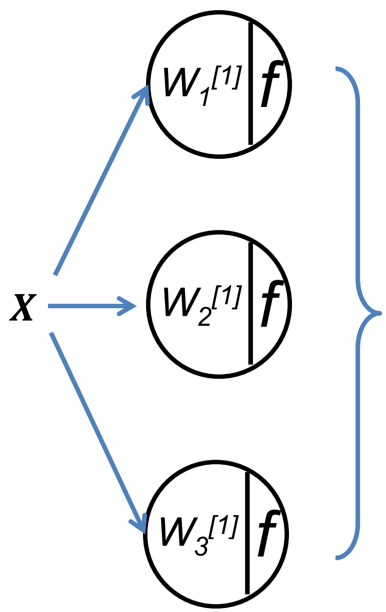


Multi-layer Perceptron (MLP)



This is a two-layer network (often does not count input layer).

Multi-layer Perceptron (MLP)



$$\boldsymbol{Z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}$$
$$\boldsymbol{a}^{[1]} = \boldsymbol{f}(\boldsymbol{Z}^{[1]})$$

X : N x1

$W^{[1]}$: 3xN

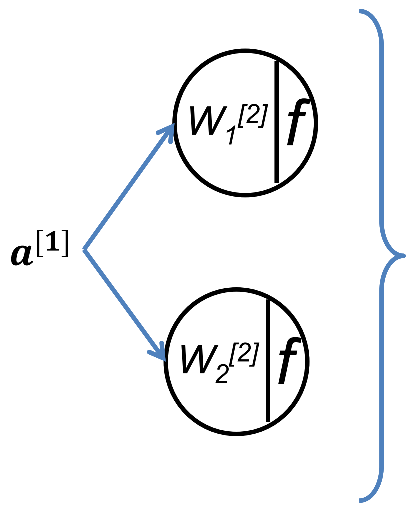
$b^{[1]}$: 3x1

$Z^{[1]}$: 3x1, score

$a^{[1]}$: 3x1, activation

f : applied element-wise

Multi-layer Perceptron (MLP)



$$\boldsymbol{Z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

$$\boldsymbol{a}^{[2]} = \boldsymbol{f}(\boldsymbol{Z}^{[2]})$$

$a^{[1]}$: 3x1

$W^{[2]}$: 2x3

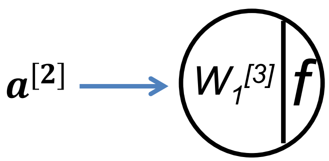
$b^{[2]}$: 2x1

$Z^{[2]}$: 2x1

$a^{[2]}$: 2x1

f : applied element-wise

Multi-layer Perceptron (MLP)



$$\boldsymbol{Z}^{[3]} = \boldsymbol{W}^{[3]}\boldsymbol{a}^{[2]} + \boldsymbol{b}^{[3]}$$
$$\boldsymbol{a}^{[3]} = \boldsymbol{f}(\boldsymbol{Z}^{[3]})$$

$a^{[2]}$: 2x1

$W^{[3]}$: 1x2

$b^{[3]}$: 1x1

$Z^{[3]}$: 1x1

$a^{[3]}$: 1x1

f : applied element-wise

Forward Pass

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$a^{[1]} = f(Z^{[1]})$$

$$Z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = f(Z^{[2]})$$

$$Z^{[3]} = W^{[3]}a^{[2]} + b^{[3]}$$

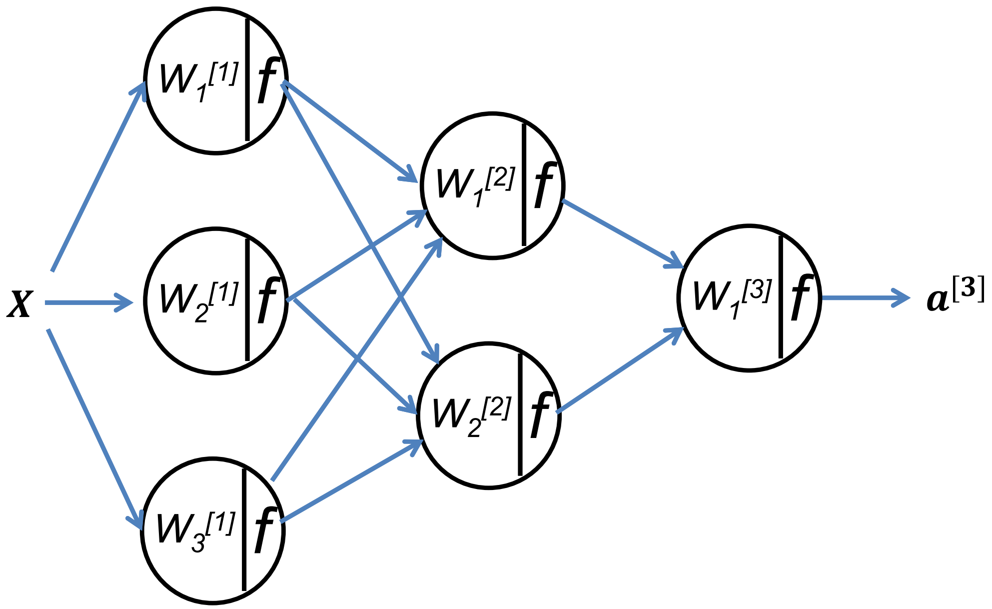
$$\hat{y} = a^{[3]} = f(Z^{[3]})$$

Forward pass: from input X to output $\hat{y}$

For every layer,

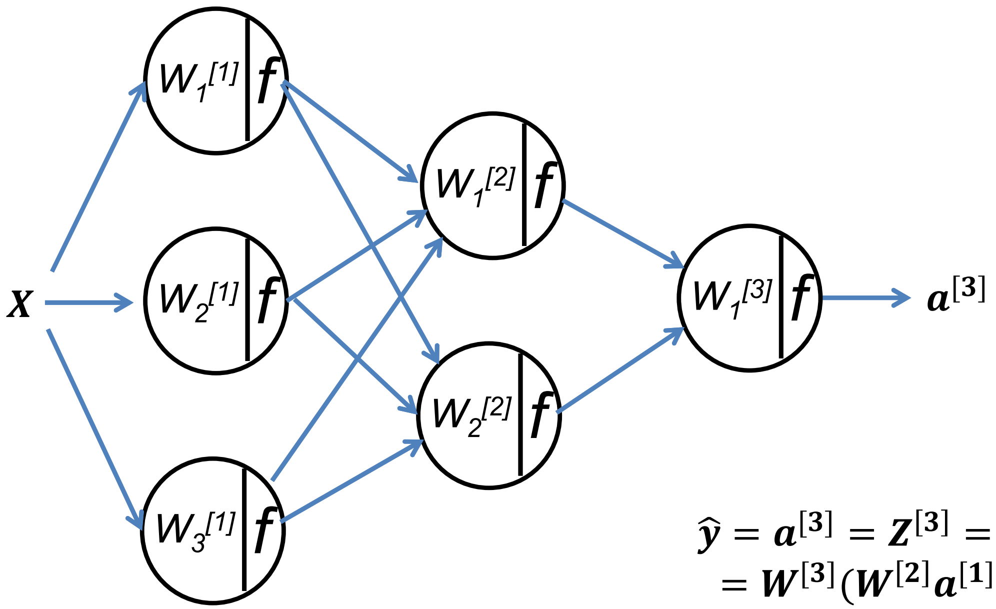
W.shape is [number of output, number of input]
b.shape is [number of output, 1]

Number of parameters?



$Z^{[1]} = W^{[1]}X + b^{[1]}$	3N+3
$a^{[1]} = f(Z^{[1]})$	0
$Z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$	2x3+2=8
$a^{[2]} = f(Z^{[2]})$	0
$Z^{[3]} = W^{[3]}a^{[2]} + b^{[3]}$	1x2+1=3
$\hat{y} = a^{[3]} = f(Z^{[3]})$	0
	3N+14

Nonlinear activation is necessary



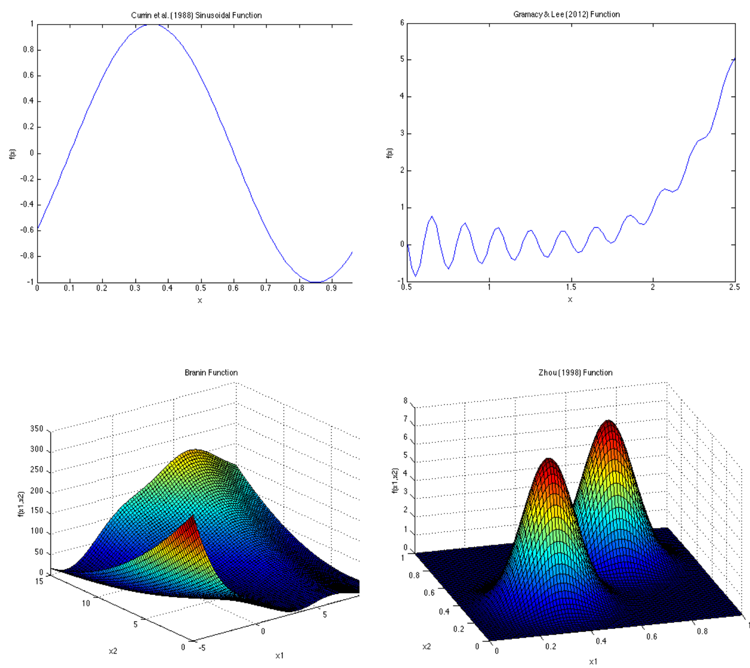
$if\ \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{x}$

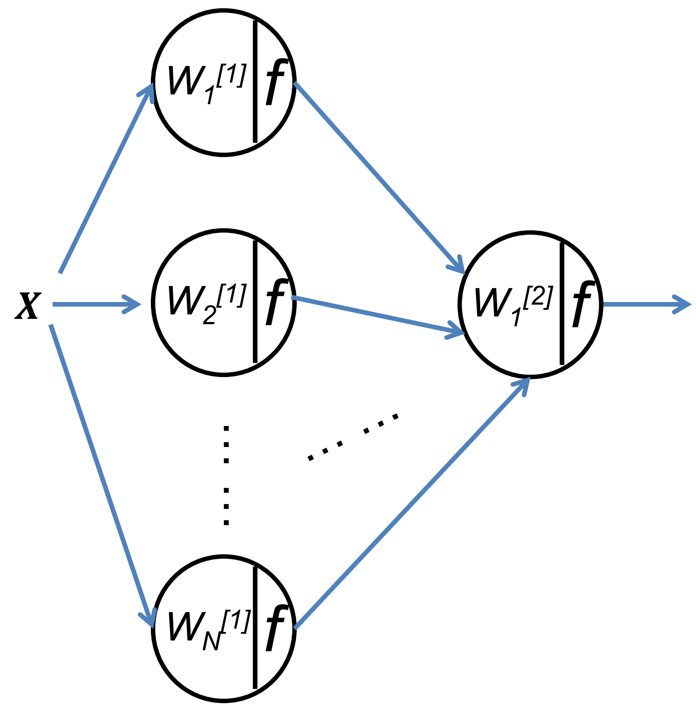
$$\boldsymbol{Z}^{[1]} = \boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}$$
$$\boldsymbol{a}^{[1]} = \boldsymbol{Z}^{[1]}$$
$$\boldsymbol{Z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$
$$\boldsymbol{a}^{[2]} = \boldsymbol{Z}^{[2]}$$
$$\boldsymbol{Z}^{[3]} = \boldsymbol{W}^{[3]}\boldsymbol{a}^{[2]} + \boldsymbol{b}^{[3]}$$

$$\begin{aligned}\hat{\boldsymbol{y}} = \boldsymbol{a}^{[3]} = \boldsymbol{Z}^{[3]} &= \boldsymbol{W}^{[3]}\boldsymbol{a}^{[2]} + \boldsymbol{b}^{[3]} \\ &= \boldsymbol{W}^{[3]}(\boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}) + \boldsymbol{b}^{[3]} \\ &= \boldsymbol{W}^{[3]}(\boldsymbol{W}^{[2]}(\boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{b}^{[1]}) + \boldsymbol{b}^{[2]}) + \boldsymbol{b}^{[3]} \\ &= \boldsymbol{W}^{[3]}\boldsymbol{W}^{[2]}\boldsymbol{W}^{[1]}\boldsymbol{X} + \boldsymbol{W}^{[3]}\boldsymbol{W}^{[2]}\boldsymbol{b}^{[1]} + \boldsymbol{W}^{[3]}\boldsymbol{b}^{[2]} + \boldsymbol{b}^{[3]}\end{aligned}$$

A linear model …

Universal Approximation



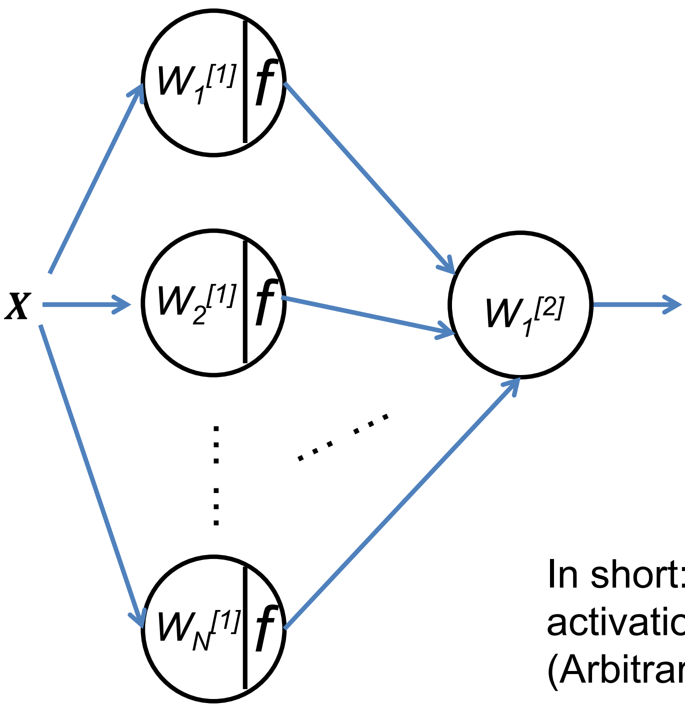


Can a MLP approximate any functions?

https://www.sfu.ca/~ssurjano

Universal Approximation

Can a MLP approximate any functions?



Approximation Capabilities of Multilayer Feedforward Networks

KURT HORNIK

Technische Universität Wien, Vienna, Austria

(*Received* 30 *January* 1990; *revised and accepted* 25 *October* 1990)

Abstract—*We show that standard multilayer feedforward networks with as few as a single hidden layer and arbitrary bounded and nonconstant activation function are universal approximators with respect to $L^p(\mu)$ performance criteria, for arbitrary finite input environment measures μ, provided only that sufficiently many hidden units are available. If the activation function is continuous, bounded and nonconstant, then continuous mappings can be learned uniformly over compact input sets. We also give very general conditions ensuring that networks with sufficiently smooth activation functions are capable of arbitrarily accurate approximation to a function and its derivatives.*

https://doi.org/10.1016/0893-6080(91)90009-T

In short: A MLP with at least one hidden layer and nonlinear activation function can approximate any continuous function (Arbitrary Width).

Similar conclusions exist for arbitrary depth, CNN and other conditions

https://en.wikipedia.org/wiki/Universal_approximation_theorem