



# Welcome to Hands-on Deep Learning (HODL)!

# Lecture 1:

# Introduction to Neural Networks and Deep Learning



15.S04: Hands-on Deep Learning  
Spring 2024  
**Farias, Ramakrishnan**

# Prerequisites



- Familiarity with Python at this level
- Familiarity with fundamental machine learning concepts (such as training/validation/testing, overfitting/underfitting, and regularization).
- If you have taken 15.071/15.072(or will be taking it concurrently) OR if you have other relevant coursework or work experience, you should be fine. If you aren't sure, email us.

The screenshot shows the 'Python' course page on kaggle.com/learn/python. The page has a dark header with a search bar and navigation icons. The main title 'Python' is displayed in large letters, followed by the subtitle 'Learn the most important language for data science.' A prominent 'Begin Course' button is visible, along with a timer indicating '5 hours to go'. Below this, there are tabs for 'Course' (which is selected) and 'Discussion'. The 'Lessons' section lists three lessons: 'Hello, Python', 'Functions and Getting Help', and 'Booleans and Conditionals'. Each lesson card includes a brief description and icons for 'Tutorial' and 'Exercise'.

# Grading

## Grading

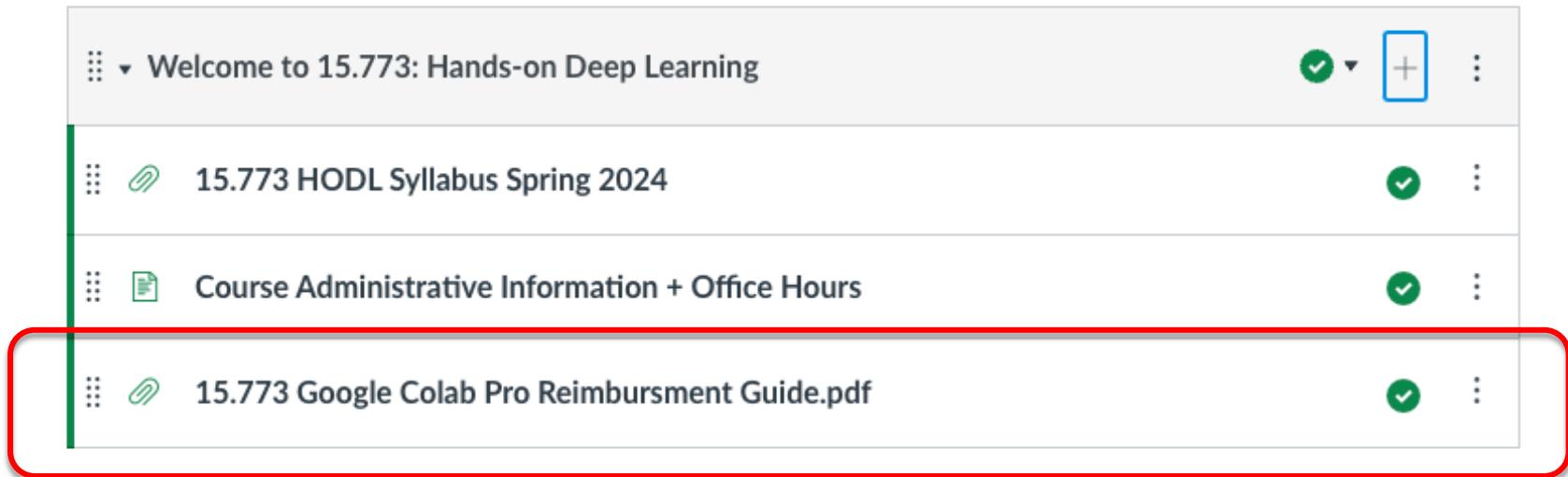
Your course grade will be based on 2 homework assignments, a final project, and class participation:

Class participation	10%
Homework assignments (25% x 2)	50%
Final project	40%

# Waitlist (Section A)

- *Registered students*
  - If you decide to drop HODL, please let the TA know ASAP so we can offer a slot to the waitlist students
- *Waitlist students*
  - Will send out an update by EOD Wednesday based on drops.
  - If you are graduating in 2024, you are welcome to Add the course as a Listener.
  - You have been added to the Canvas site so that you have access to the livestream and other course materials, while you figure out your class schedule. *Please note that this access will automatically disappear in a week or so.*
- *If you are a Listener or on the Waitlist, please plan on watching the livestream. We are at classroom physical capacity.*

# You can sign up for Google Colab Pro and get reimbursed for it



Welcome to 15.773: Hands-on Deep Learning

- 15.773 HODL Syllabus Spring 2024
- Course Administrative Information + Office Hours
- 15.773 Google Colab Pro Reimbursement Guide.pdf

# “Housekeeping” (Section A)

- Class starts at 8:35am and ends at 9:55am
- Lecture slides & links to Colabs will be posted to Canvas after class
- No mobile phones in class. OK to take notes on a tablet/laptop
- If I pronounce your name wrong, please correct me immediately
- An absence is considered excused only if it is due to medical or religious reasons. Otherwise, we will mark it as unexcused

# TA – Evan Yao

- PhD student at the Operations Research Center working with Prof. Retsef Levi
- B.A. Harvard University (2017), Computer Science and Mathematics
- Full-time work experience in tech (Meta) and internship experience in finance (Citadel)
- Research Interests: machine learning, data science, operations management, design and analysis of algorithms
- Singer in the MIT Chamber Choir, avid skier and public transit enthusiast



# Rama Ramakrishnan

Professor of the Practice, AI/ML

## How to reach me outside of class:

- <https://calendly.com/ramamit/> for 1-on-1 mtgs
- Please email me ([ramar@mit.edu](mailto:ramar@mit.edu)) or my assistant Ahmed Abusharkh ([ahmedabu@mit.edu](mailto:ahmedabu@mit.edu)) if the Calendly times don't work

**Education:** Ph.D. and M.S. (Operations Research; MIT), B.Tech. (Engineering; Indian Institute of Technology)

## Industry Experience

McKinsey followed by 4 data science/machine learning startups (in asset management, transportation, retail and ecommerce). Exits to Oracle, Demandware and Salesforce

Post acquisition, Chief Analytics Officer at Oracle Retail; SVP of Data Science at Salesforce

My most recent startup – CQuotient – is now Salesforce Einstein AI for Commerce and is one of the top personalization engines in the world (live on ~10,000 e-commerce sites worldwide)

**Interests:** Applying AI/ML to business problems (especially shortest-path-to-human-impact applications e.g., healthcare, drug development)

**Outside Activities:** Active angel investor, AI advisor to venture firms, several startups and a few large companies

# Why HODL?

# Why did we create HODL?



- DL is one of the most exciting and profound technology developments of our lifetimes
- It is important for Sloanies to understand how to use DL to transform businesses and create exciting new products/services
- While MIT has other (excellent) DL courses, we wanted one that was a better fit for Sloan

# HODL's “philosophy”

- Focus on the key concepts that underlie DL
- Skip the math\* (but we are happy to geek out in office hours and/or suggest readings for those who are interested)

---

\*If you are looking for a ‘mathy’ DL course, this course won’t be a good fit and you may want to consider dropping it.

# HODL's “philosophy”

- Focus on the key concepts that underlie DL
- Skip the math (but we are happy to geek out in office hours and/or suggest readings for those who are interested)
- Focus on coding DL models ...
  - It is the only way to develop a visceral (not just intellectual) understanding of how this stuff works
  - Successful new products and services are often inspired by hands-on tinkering

# HODL's “philosophy”

- Focus on the key concepts that underlie DL
- Skip the math (but we are happy to geek out in office hours and/or suggest readings for those who are interested)
- Focus on coding DL models ...
  - It is the only way to develop a visceral (not just intellectual) understanding of how this stuff works
  - Successful new products and services are often inspired by hands-on tinkering
- ... but only up to a point
  - We aren't trying to teach you how to be ML engineers
  - **But we want you to be able to build a V1.0 DL model by yourself without looking for a Data Scientist or ML Engineer to help you out**

## SCHEDULE (subject to change)

Mon	05 Feb	Introduction to Neural Networks and Deep Learning (motivation, smart representations, layers, activations, architectures); Training Deep NNs (loss functions, gradient descent, regularization)	
Wed	07 Feb	Training Deep NNs (continued); Introduction to Keras/Tensorflow (tensors, functional API, training, evaluation); Application to tabular data	
Mon	12 Feb	Deep Learning for Computer Vision – Building Convolutional Neural Networks from scratch	HW Assignment #1 posted
Wed	14 Feb	Deep Learning for Computer Vision – Transfer Learning and Fine-tuning; Introduction to HuggingFace	
Mon	19 Feb	<i>HOLIDAY (President's Day)</i>	
Tue	20 Feb	Deep Learning for Natural Language – The Basics (Tokenization, N-grams, Bag-of-Words)	HW Assignment #1 due
Wed	21 Feb	Deep Learning for Natural Language – Embeddings	HW Assignment #2 posted
Mon	26 Feb	Deep Learning for Natural Language – Transformers	Project Proposal Due
Wed	28 Feb	Deep Learning for Natural Language – Transformers, Self-Supervised Learning	
Mon	04 Mar	Generative AI - Large Language Models (LLMs) and Retrieval Augmented Generation (RAG)	HW Assignment #2 due
Wed	06 Mar	Generative AI – Adapting LLMs with Parameter-Efficient Fine-Tuning	
Mon	11 Mar	Generative AI – Text-to-Image Models	Projects due
Wed	13 Mar	Project Presentations	

We will start with a very quick introduction  
to the relationship between

- Artificial Intelligence (AI)
- Machine Learning (ML)
- Deep Learning (DL)
- Generative AI

# The field of Artificial Intelligence originated in 1956



In the back row from left to right are Oliver Selfridge, Nathaniel Rochester, Marvin Minsky, and John McCarthy. In front on the left is Ray Solomonoff; on the right, Claude Shannon. The identity of the person between Solomonoff and Shannon remained a mystery for some time. THE MINSKY FAMILY

<https://spectrum.ieee.org/dartmouth-ai-workshop>

# MIT was well-represented 👍



Marvin  
Minsky

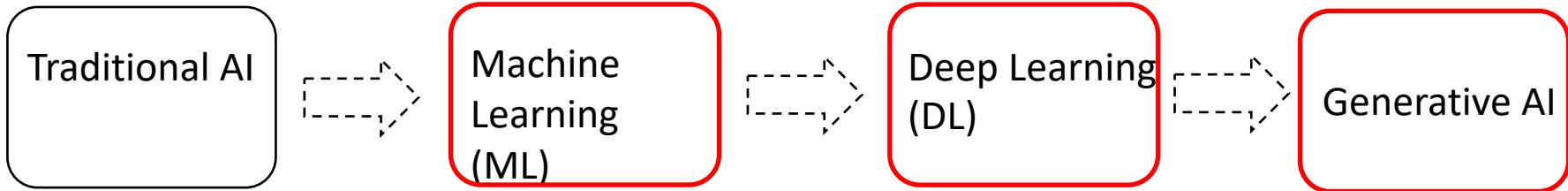
John McCarthy

Claude Shannon

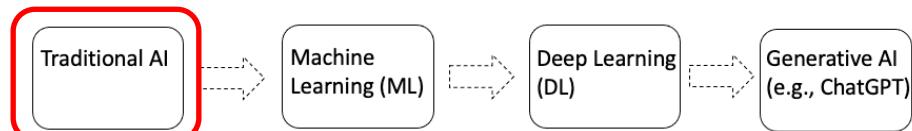
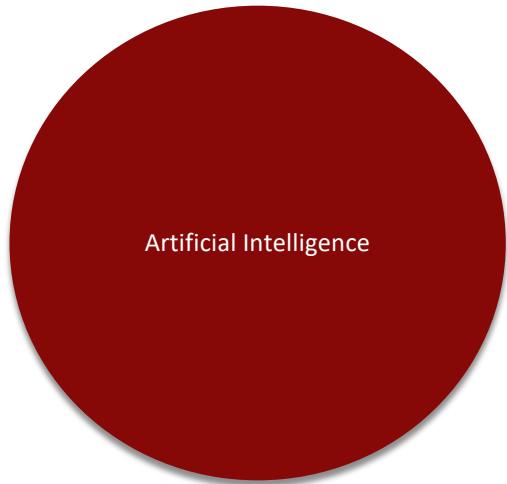
In the back row from left to right are Oliver Selfridge, Nathaniel Rochester, Marvin Minsky, and John McCarthy. In front on the left is Ray Solomonoff; on the right, Claude Shannon. The identity of the person between Solomonoff and Shannon remained a mystery for some time. THE MINSKY FAMILY

<https://spectrum.ieee.org/dartmouth-ai-workshop>

# In the decades since its founding, it has gone through several “breakthroughs”



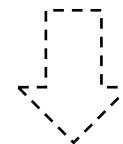
# The traditional approach to AI



*The Goal: Give computers the ability  
to do tasks that traditionally only  
humans have been able to do*

Traditional approach:

Ask human experts how they do it, write it down as IF-THEN rules, explicitly program these rules into the computer



Success in only a few areas

# Why is this so difficult?



- “We know more than we can tell” (Polanyi’s Paradox)
  - We can do lots of things easily but find it very hard to describe how exactly we do them
- We can’t write down if-then rules to cover all situations, edge cases etc. (i.e., we can’t generalize to new situations)

# To address this problem, a different approach was developed

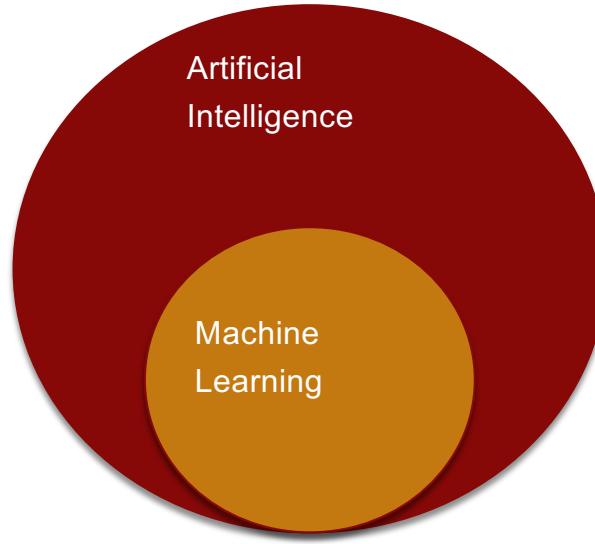
Instead of explicitly telling the computer what to do ...

# To address this problem, a different approach was developed

Instead of explicitly telling the computer what to do ...

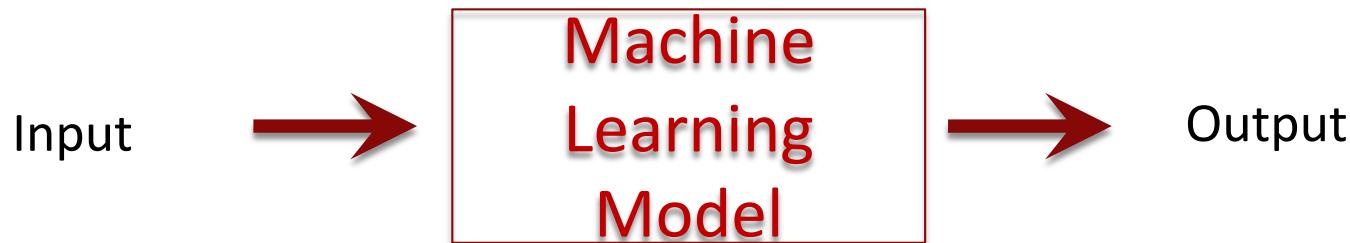
*Provide the computer with lots of examples of inputs-and-outputs and use statistical techniques to learn the relationship between inputs and outputs*

# This is Machine Learning



“learn from input-output examples using statistical techniques\*”

# There are numerous ways\* to create Machine Learning models



- Linear Regression
- Logistic Regression
- Classification and Regression Trees
- Support Vector Machines
- Random Forests
- Gradient Boosted Machines
- Neural Networks
- .....

---

\*Covered in detail in courses like 15.071 The Analytics Edge

Machine Learning has had tremendous impact and is used worldwide across numerous applications (e.g., credit scoring, loan granting, disease prediction, demand forecasting, ....) where the input data is structured

---

# Structured input data = data that can be “numericalized” into a spreadsheet\*

INPUT						OUTPUT
Age	Smoker	Exercise	Cholesterol	Family History	Blood Pressure	Cardiac Arrest
30	No	120	190	Yes	120/80	No
45	Yes	30	220	No	130/90	Yes
50	No	60	210	Yes	125/85	No
35	Yes	45	230	No	135/88	Yes
40	No	150	180	Yes	118/78	No
55	Yes	10	240	Yes	140/92	Yes
28	No	180	170	No	115/75	No
60	Yes	20	250	Yes	145/95	Yes
48	No	90	200	No	128/82	No
53	Yes	35	235	Yes	133/89	Yes

\*informal definition

# But the situation is different for unstructured input data (images, videos, text, audio, ...)

**Images**



**Text**

*Four score and seven years  
ago our fathers brought forth,  
upon this continent, ...*

**Audio**

...

---

# The reason: The “raw form” of unstructured data has no intrinsic meaning



	Red											Green											Blue										
	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]		[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]		[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	
[1,]	147	131	138	144	131	134	144	135	133	145		186	171	179	185	171	172	180	171	168	180		251	232	233	237	230	243	255	255	250	246	
[2,]	140	131	141	149	138	138	143	132	136	146		177	169	180	188	176	175	178	167	169	180		248	234	239	245	238	246	255	251	246	243	
[3,]												175	169	174	176	169	173	179	172	171	182		255	241	238	236	236	229	241	253	249	238	234
[4,]																							255	252	243	233	228	237	242	234	218	205	
[5,]																							255	255	249	231	228	231	224	215	204	166	
[6,]																							255	255	230	192	189	202	205	205	204	147	
[7,]																							231	231	188	140	138	152	156	159	177	136	
[8,]																							155	172	149	114	113	111	93	82	119	115	
[9,]																							107	130	108	93	113	100	67	66	81	95	
[10,]																							84	104	90	69	69	61	52	63	59	46	

To use ML on unstructured data, we had to manually create a better representation of the data first\*

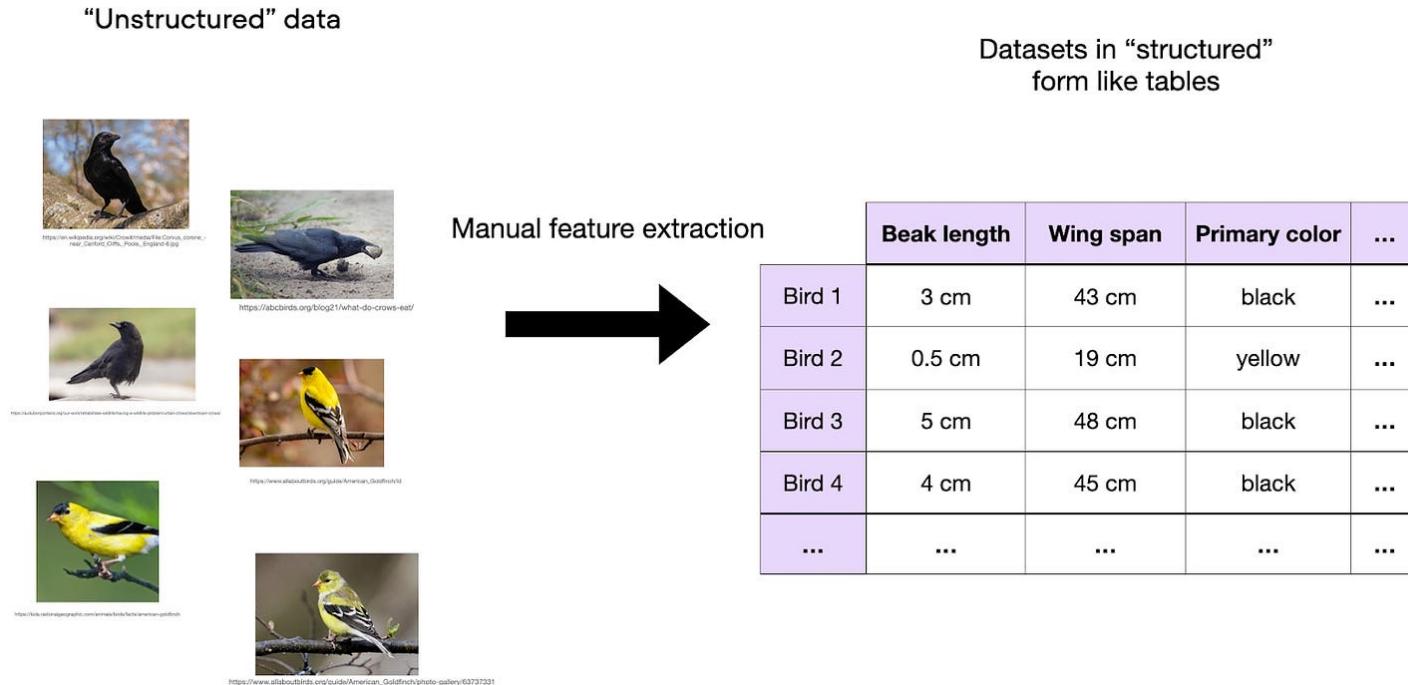


Image source: <https://magazine.sebastianraschka.com/p/ai-and-open-source-in-2023>

\*sometimes referred to as feature extraction

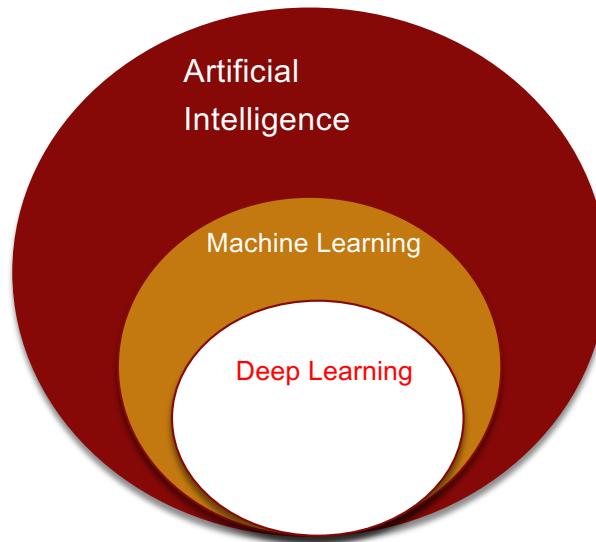
# Learning effective representations is vital

- The raw data has to somehow be transformed into a different representation
- Historically, researchers **manually** developed these representations and then fed them to traditional machine learning algorithms (often just linear/logistic regression!)
- But this required massive human effort and thus sharply limited the reach of Machine Learning

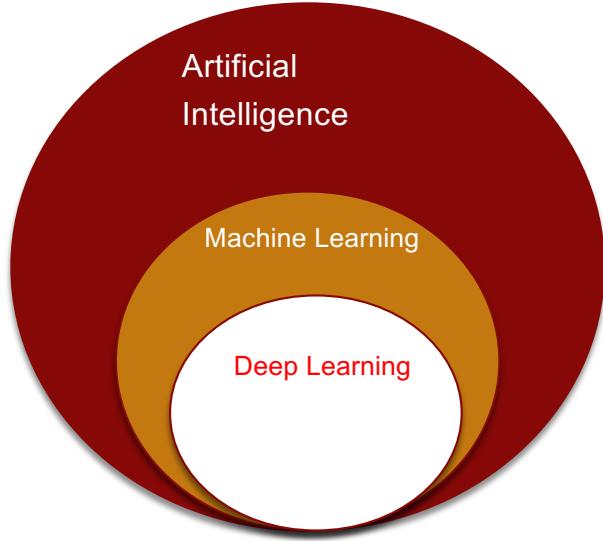


But developing good representations (before ML could be used) required massive human effort and this “human bottleneck” sharply limited the reach of Machine Learning

To address this problem, a different approach was developed – Deep Learning



# Deep Learning can handle unstructured input data without upfront manual preprocessing!!



Structured *and* **Unstructured** data → Deep Learning → Prediction

v	cc0	w_01	rw	gg_04	v	v
0	367	4.51	682	3.51	1	0
0	162	4.93	712	33.67	1	0
0	103	4.91	667	4.74	0	0
0	1212	5.17	722	50.81	0	0
1	1394	4.65	667	3.84	0	1
1	131	4.78	722	24.22	0	0
0	87	4.95	682	69.91	1	0
0	84	4.43	707	5.63	1	0
0	360	4.53	677	13.85	2	1
0	254	5.14	662	5.12	2	0
0	316	4.75	767	6.07	0	0
0	93	5	747	3.02	0	0



*Four score and seven years ago our fathers brought forth, upon this continent, ...*

# What can Deep Learning do that traditional Machine Learning can't?

---

- It can automatically extract smart representations from raw, unstructured data.
- We can simply feed these smart representations to traditional models like linear regression or logistic regression and achieve amazing performance



This demolishes the “human bottleneck” for using  
Machine Learning with unstructured data

---

# Deep Learning



The breakthrough came  
from the confluence of  
three forces ...

- New algorithmic ideas
- Unprecedented amounts of data  
(due to the digitization of  
everything)
- Compute power (from the use of  
powerful Graphics Processing  
Units (GPUs))

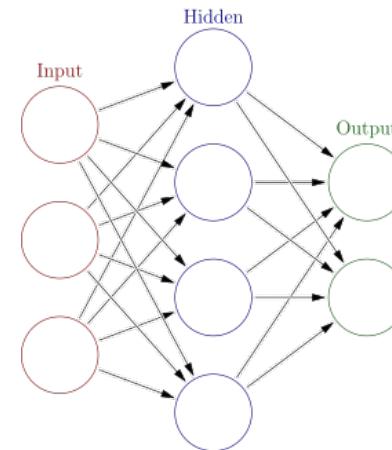
# Deep Learning

The breakthrough came from the confluence of three forces ...

- New algorithmic ideas
- Unprecedented amounts of data (due to the digitization of everything)
- Compute power (from the use of powerful Graphics Processing Units (GPUs))



... applied to an old ML idea:  
**Neural Networks**



[https://en.wikipedia.org/wiki/Artificial\\_neural\\_network](https://en.wikipedia.org/wiki/Artificial_neural_network)



What is the *immediate* application of Deep Learning?



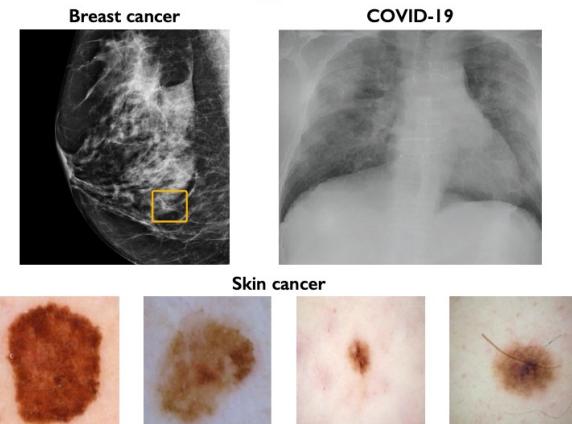
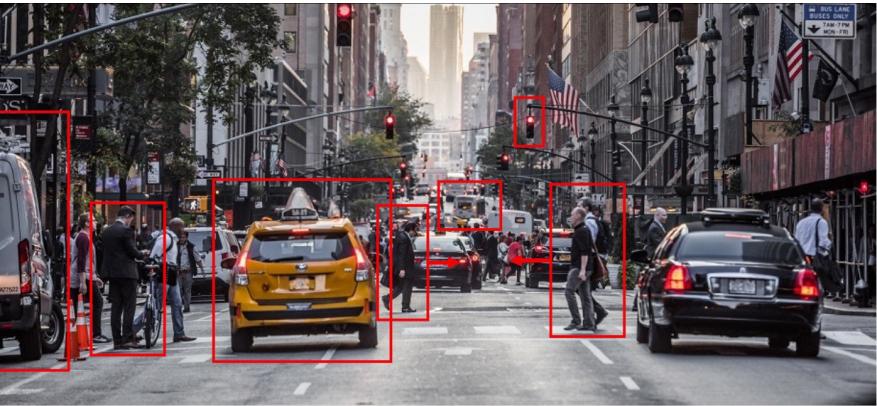
*Every “sensor” can be given the ability to detect, recognize and classify what it is sensing*

---

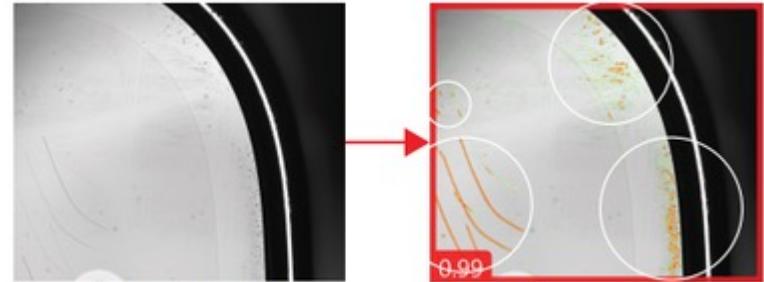
# Examples

## Use Face ID on your iPhone or iPad Pro

Face ID lets you securely unlock your iPhone or iPad, authenticate purchases, sign in to apps, and more — all with just a glance.



6.S191 Introduction to Deep Learning  
[introtodeeplearning.com](http://introtodeeplearning.com) [@MITDeepLearning](https://twitter.com/MITDeepLearning)

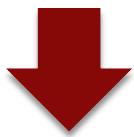


Smartphone glass defect

<https://dwfritz.com/smart-cosmetic-defect-detection-increases-productivity/>

Detected by system

*Every “sensor” can be given the ability to detect, recognize and classify what it is sensing*



*You can create dramatically better products and services by “attaching” DL to sensors*

---

# (Spotted last week!) Binoculars + DL = Smart Binoculars

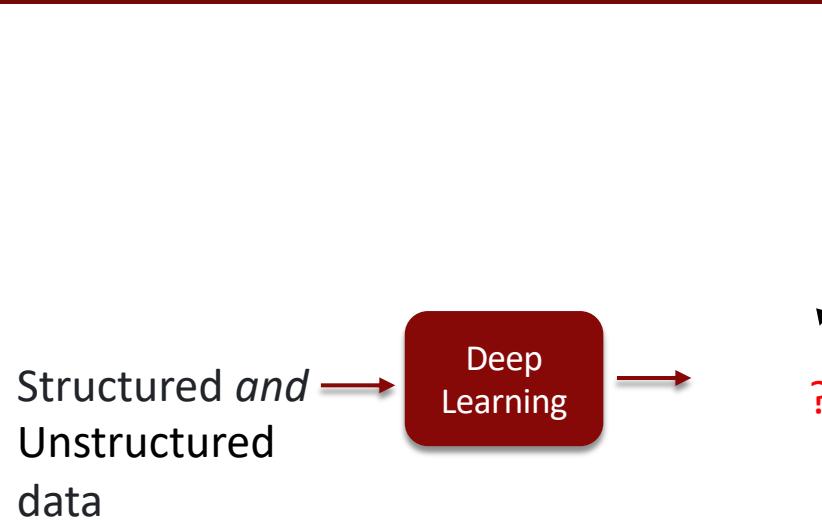




+ DL =



# Now, let's turn our attention to the output



# With Deep Learning, we could predict/generate structured outputs easily

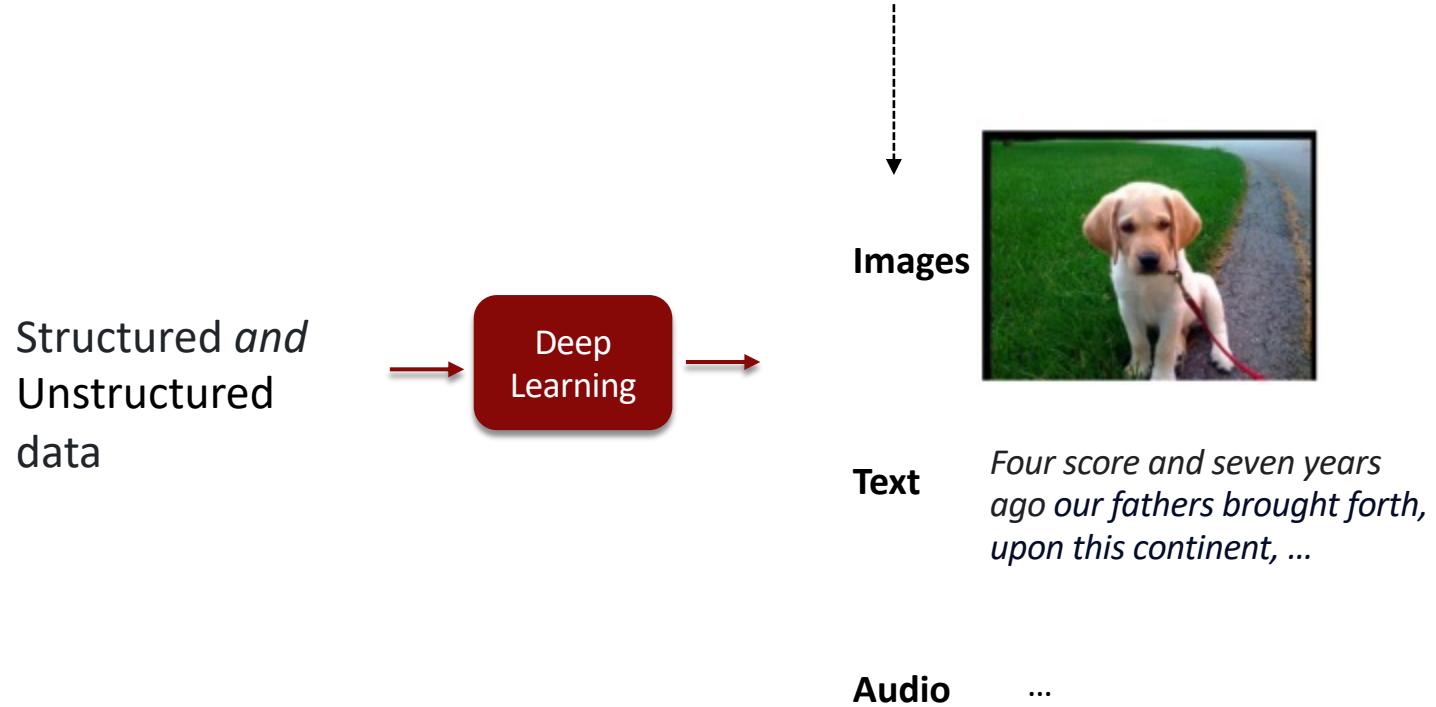
Structured and Unstructured data



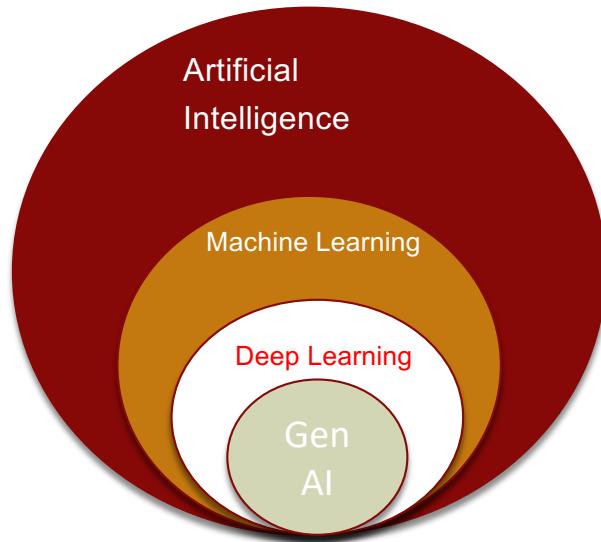
Examples:

- A single number
  - The probability that a borrower will repay a loan
  - The demand for a product next week
- A few numbers
  - The 4 probabilities that an image contains a chair, stool, table or sofa
  - The two GPS coordinates of a taxi

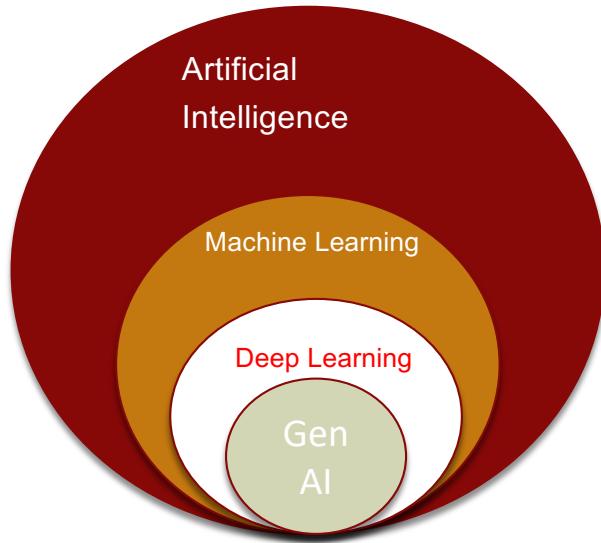
# But we couldn't generate unstructured output very well!



# And then Generative AI happened



# Gen AI can produce unstructured outputs

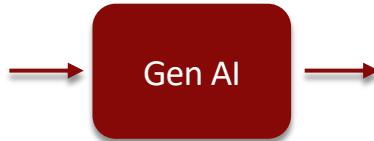


Unstructured input → Gen AI → Unstructured output

"generate a picture of a cute labrador retriever puppy"



# Image-to-text

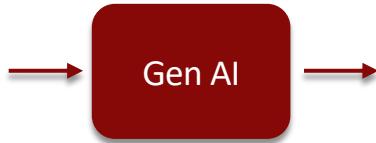


*two cats laying on  
a pink blanket  
with remotes*

<https://huggingface.co/spaces/nielsr/comparing-captioning-models>

# Text-to-image

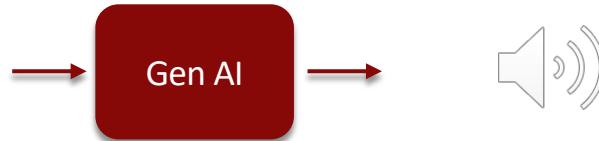
temple in ruins, forest, stairs, columns, cinematic, detailed, atmospheric, epic, concept art, Matte painting, background, mist, photo-realistic, concept art, volumetric light, cinematic epic + rule of thirds octane render, 8k, corona render, movie concept art, octane render, cinematic, trending on artstation, movie concept art, cinematic composition , ultra-detailed, realistic , hyper-realistic , volumetric lighting, 8k –ar 2:3 –test –uplight



<https://mpost.io/best-100-stable-diffusion-prompts-the-most-beautiful-ai-text-to-image-prompts/>

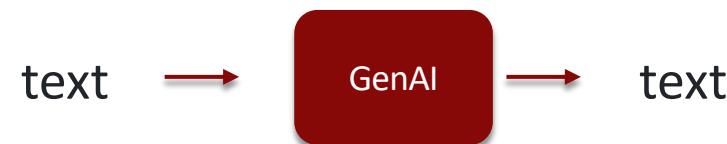
# Text-to-music

The main soundtrack of an arcade game. It is fast-paced and upbeat, with a catchy electric guitar riff. The music is repetitive and easy to remember, but with unexpected sounds, like cymbal crashes or drum rolls.



<https://google-research.github.io/seanet/musiclm/examples/>

# Text-to-text



# Multi-modal



# A fun multi-modal example

[text,

image]

→ ChatGPT

→



It's Wednesday at 4 pm. Can I park at this spot right now? Tell me in 1 line.

# A fun multi-modal example

[text,

image]



text

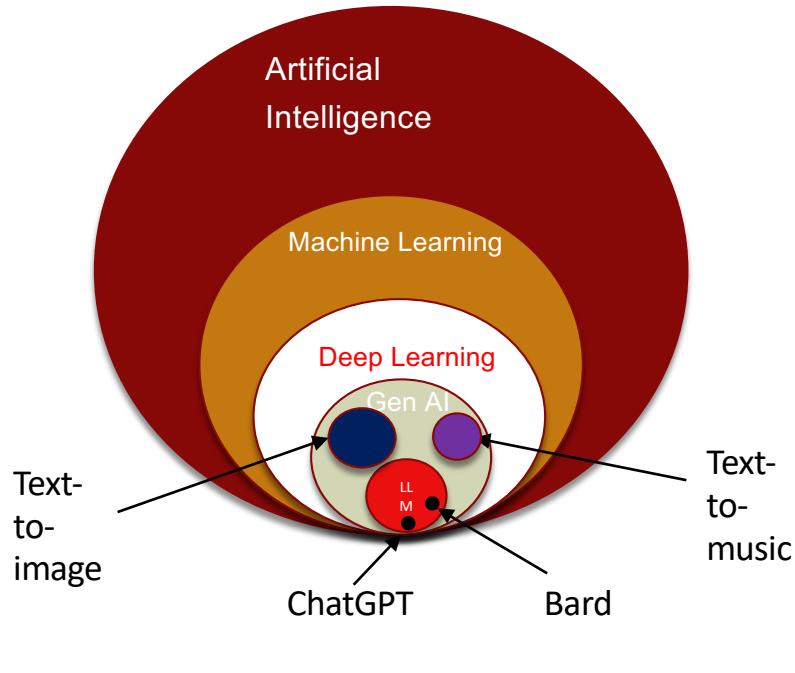


Yes, you can park for up to 1 hour starting at 4 pm.

It's Wednesday at 4 pm. Can I park at this spot right now? Tell me in 1 line.

<https://twitter.com/petergyang/status/1707169696049668472>

# The landscape



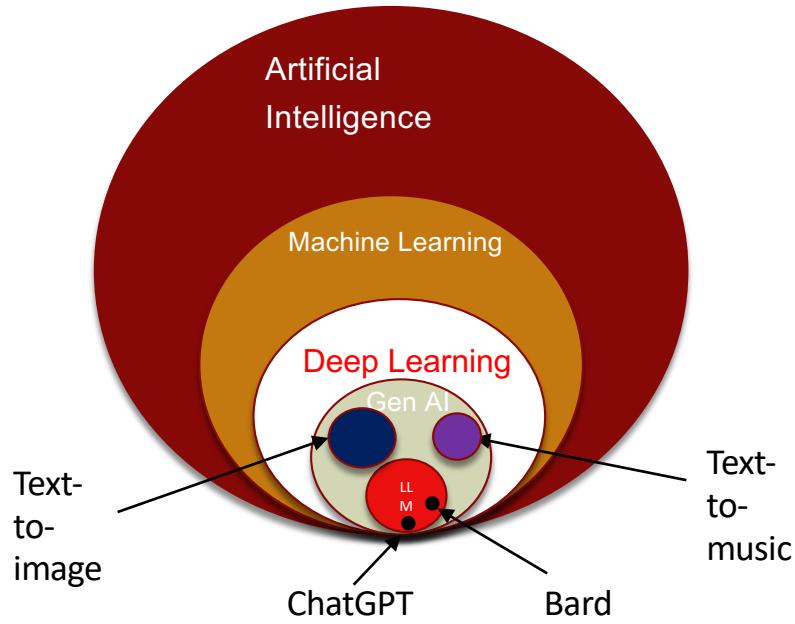
... and the  
circles inside  
GenAI are  
merging!

# Summary: X-to-Y



X and Y can be  
*anything and it can  
be multi-modal!*

# Note that ALL the AI excitement is due to the success of Deep Learning



If you understand Deep Learning,  
everything becomes possible! 😊

OK, let's start at the beginning.  
What's a Neural Network?

# Recall Logistic Regression



$$Pr(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)}}$$

# Recall Logistic Regression

$$Pr(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)}}$$

Let us look at it through the lens of a “network” of mathematical operations.

# We will work with this simple classification example

- Given independent variables ...
  - GPA
  - Experience
- ... predict who will be called for a job interview

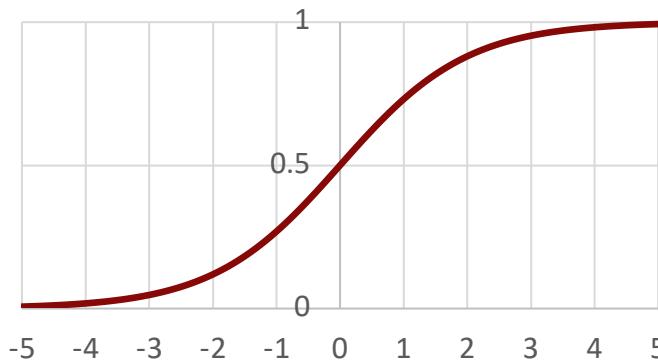
	Interview	GPA	Experience
1	0	3.27	1.93
2	0	3.37	0.07
3	0	3.57	1.91
4	0	3.91	4.35
5	0	3.20	1.70
6	1	3.90	2.41
7	1	3.94	3.00
8	0	3.66	2.47
9	0	3.63	0.93
10	0	3.06	4.14
11	0	3.21	3.34
12	0	3.18	3.97
13	0	3.69	0.54
14	0	3.38	3.62
15	1	3.77	2.06
16	0	3.50	4.10
...	...	...	...
25	1	3.31	3.46
26	0	3.78	0.29
27	0	3.87	1.21
28	0	4.00	0.49
29	1	3.87	2.11

# We will work with this simple classification example

	Interview	GPA	Experience
1		0 3.27	1.93
2		0 3.37	0.07
3		0 3.57	1.91
4		0 3.91	4.35
5		0 3.20	1.70
6	1	3.90	2.41
7	1	3.94	3.00
8		0 3.66	2.47
9		0 3.63	0.93
10		0 3.06	4.14
11		0 3.21	3.34
12		0 3.18	3.97
13		0 3.69	0.54
14		0 3.38	3.62
15	1	3.77	2.06
16		0 3.50	4.10
...	...	...	...
25		1 3.31	3.46
26		0 3.78	0.29
27		0 3.87	1.21
28		0 4.00	0.49
29	1	3.87	2.11

We can estimate this logistic regression model and find values for the coefficients:

$$P(Y = 1) = \frac{1}{1 + e^{-(0.4 + 0.2 \cdot \text{GPA} + 0.5 \cdot \text{Experience})}}$$



We can re-write this formula as a **network** with input data flowing through two functions that have been connected

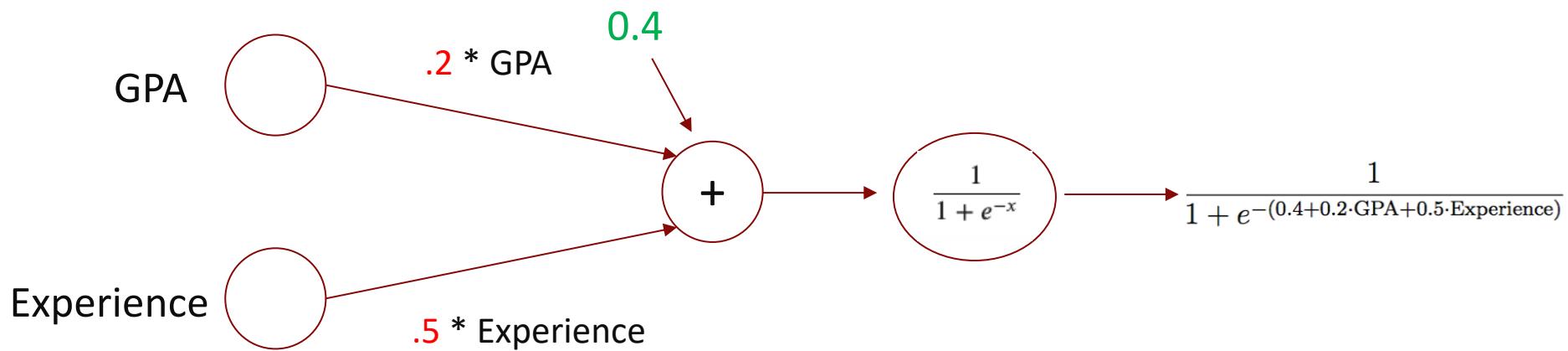
Model equation:

$$P(Y = 1) = \frac{1}{1 + e^{-(0.4 + 0.2 \cdot \text{GPA} + 0.5 \cdot \text{Experience})}}$$

We can re-write this formula as a **network** with input data flowing through two functions that have been connected

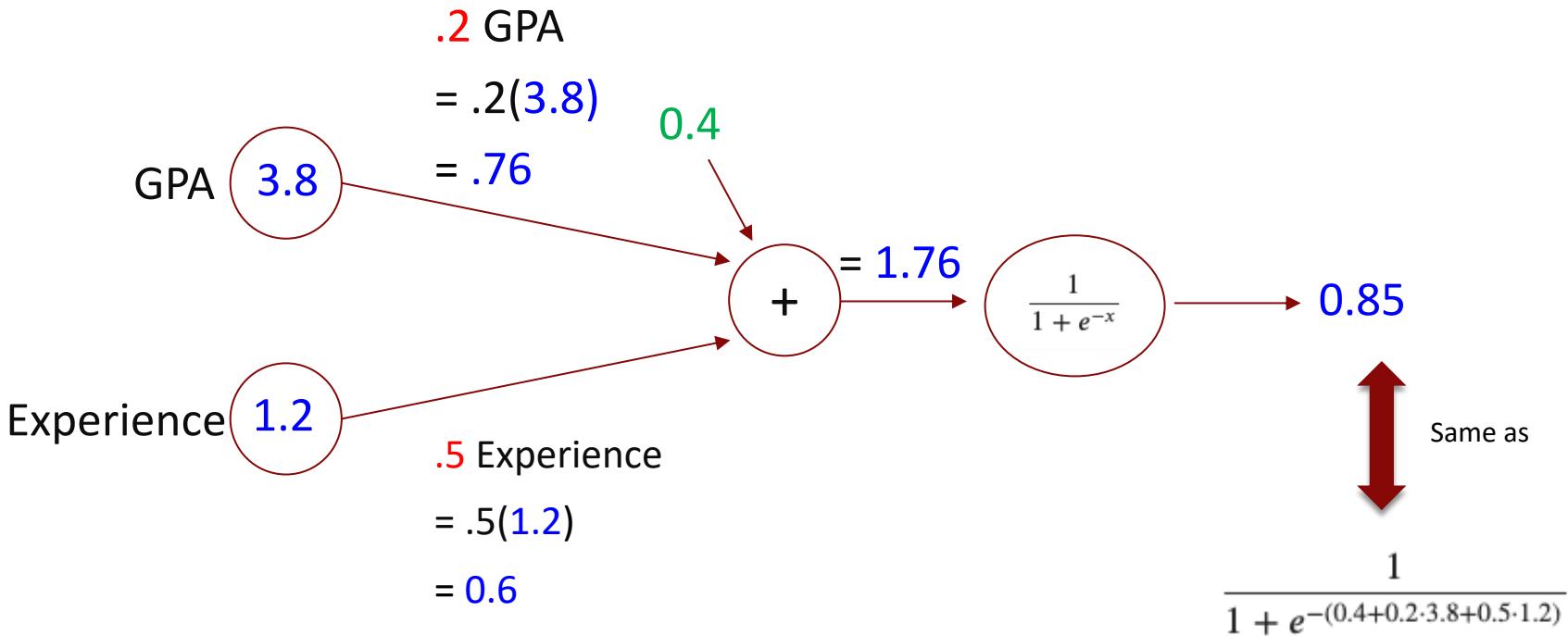
Model equation:

$$P(Y = 1) = \frac{1}{1 + e^{-(0.4 + 0.2 \cdot \text{GPA} + 0.5 \cdot \text{Experience})}}$$



# Let's make a prediction with this “network”

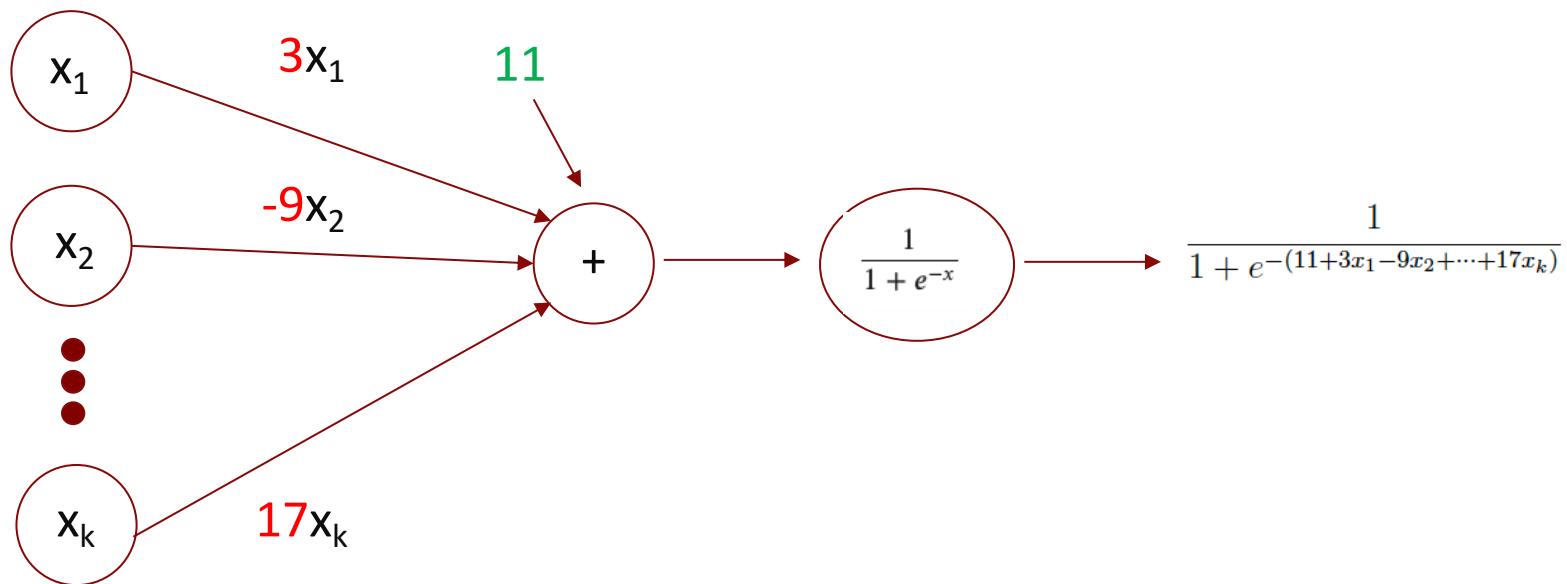
Consider a job applicant with a GPA of 3.8 and 1.2 years of experience.



# The general logistic regression model viewed through a network lens

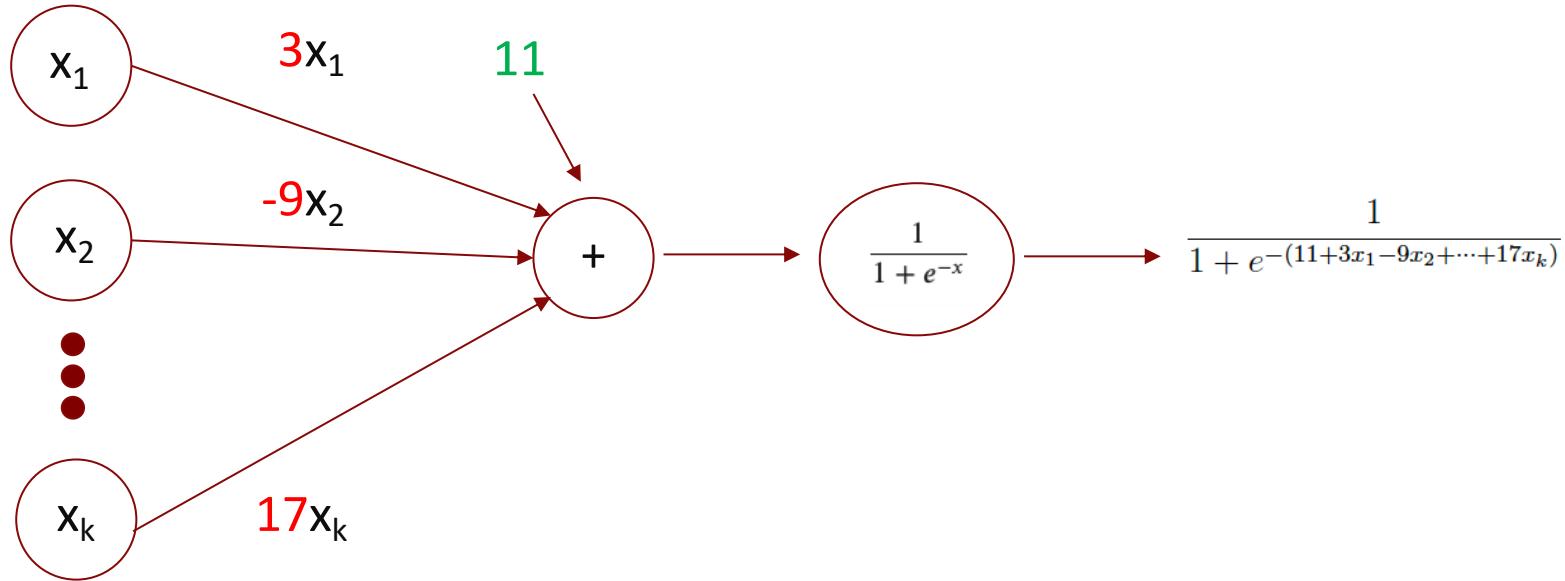
Model equation:

$$P(Y = 1) = \frac{1}{1 + e^{-(11 + 3x_1 - 9x_2 + \dots + 17x_k)}}$$



Notice how the data flows through the network from left to right

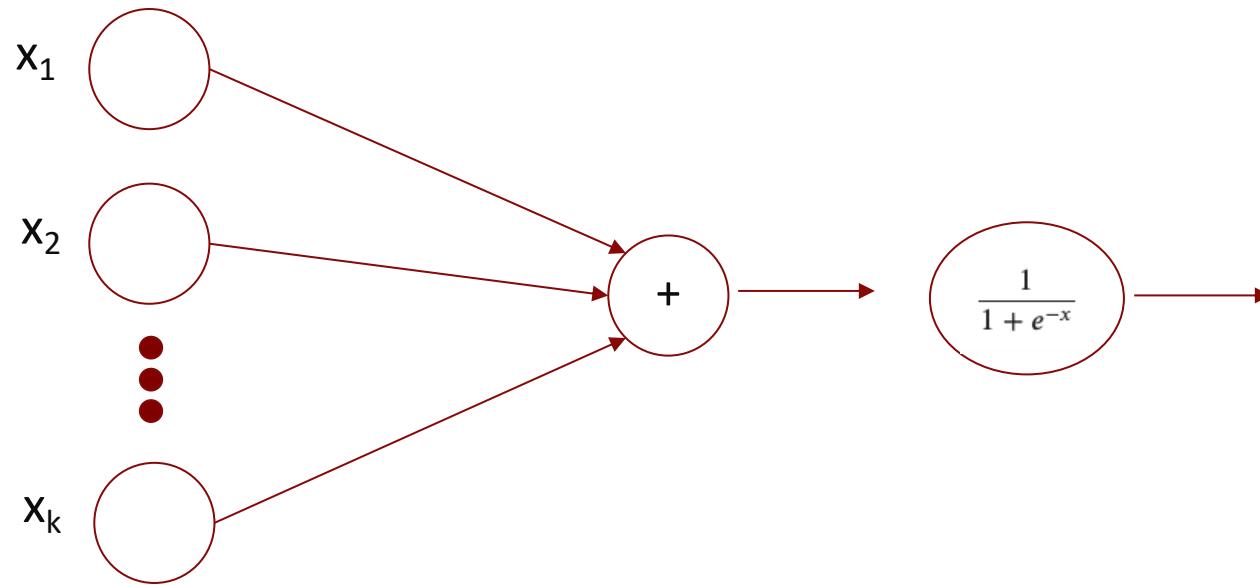
# Terminology



- Multipliers on values from each node = coefficients = **weights**
- Intercept = **bias**

What's the advantage of viewing through a network “lens”?

Recall the notion of learning smart representations of the input data



\*I am not showing the weights and biases to avoid clutter

To learn smart representations, we would like to transform the inputs one or more times before we do the prediction



To learn smart representations, we would like to transform the inputs one or more times before we do the prediction

$x_1$  

$x_2$  

$\vdots$

$x_k$  

What is the  
simplest thing we  
can do here?



To learn smart representations, we would like to transform the inputs one or more times before we do the prediction

$x_1$  

$x_2$  

$\vdots$

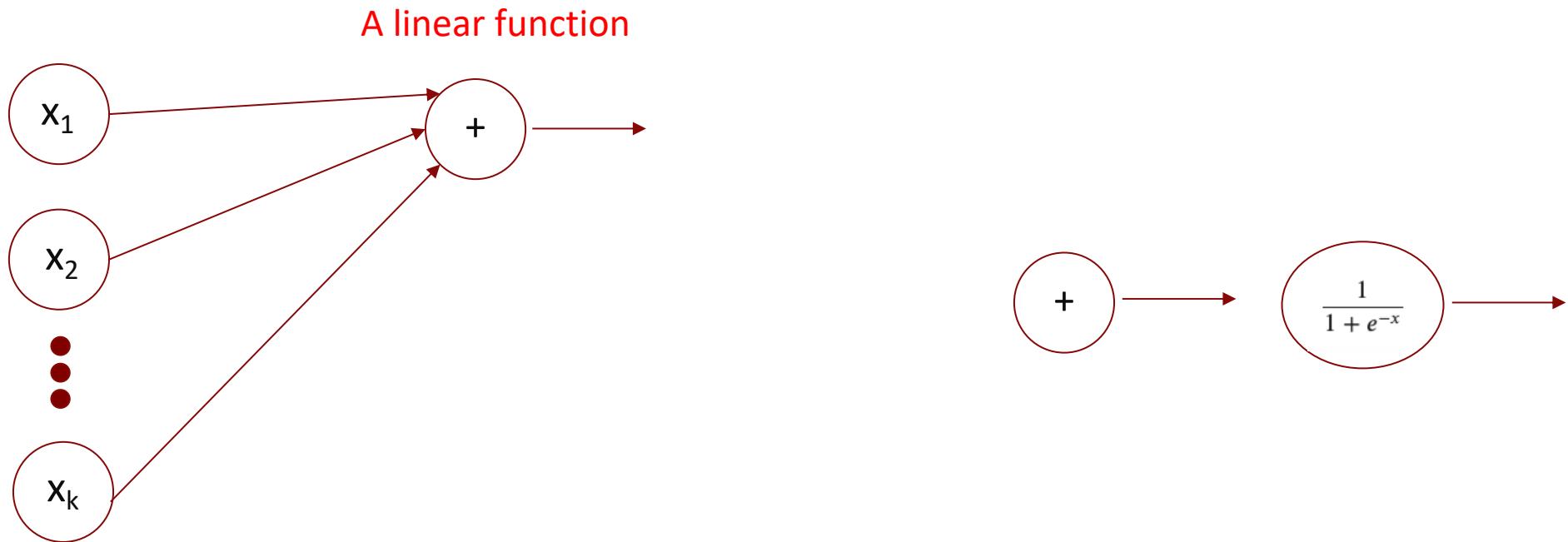
$x_k$  

What is the  
simplest thing we  
can do here?

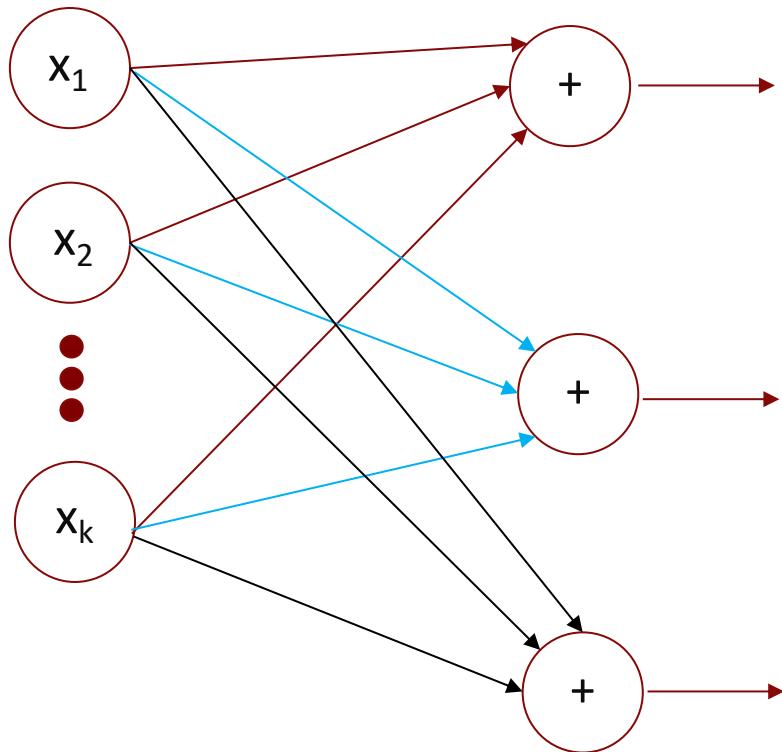
We can insert  
linear functions



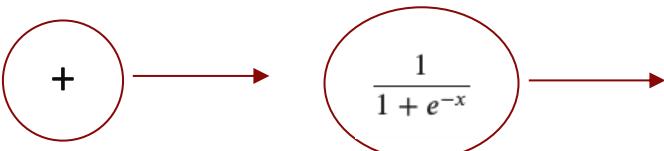
To learn smart representations, we would like to transform the inputs one or more times before we do the prediction



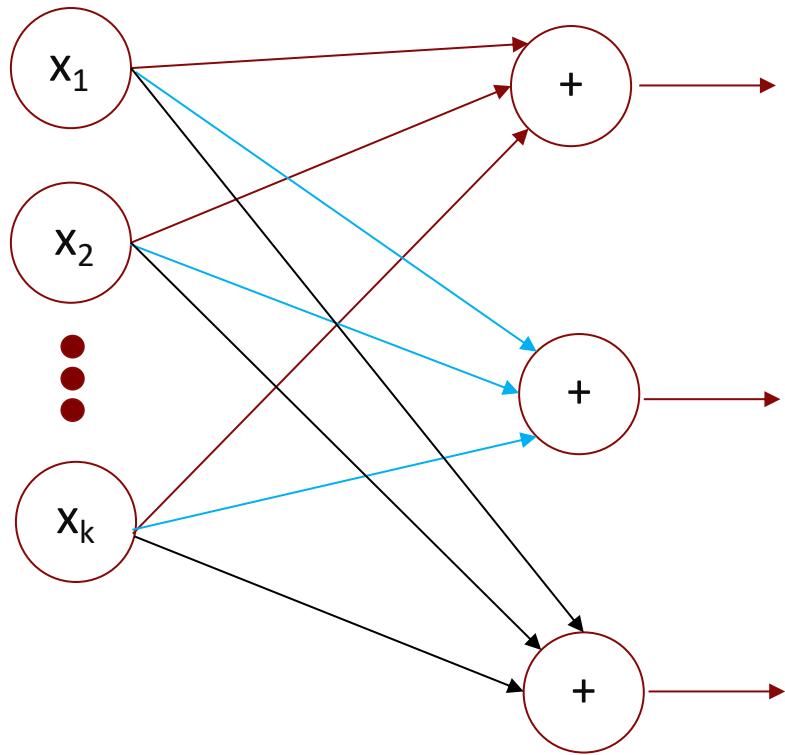
To learn smart representations, we would like to transform the inputs one or more times before we do the prediction



We can "stack" as many linear functions as we want



To learn smart representations, we would like to transform the inputs one or more times before we do the prediction

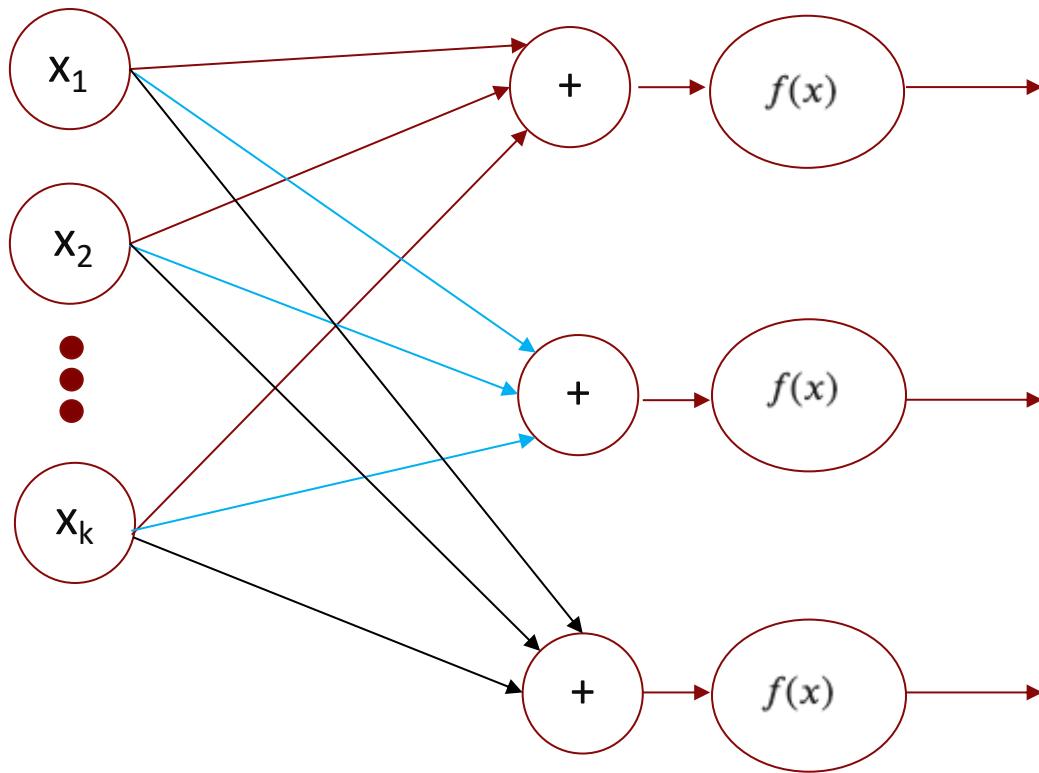


Notice that we have transformed a  $k$ -dimensional input to a 3-dimensional vector

The diagram shows the final step of the transformation. A red arrow points from the output of the third addition node to a circle containing the logistic function formula:  $\frac{1}{1 + e^{-x}}$ . Another red arrow points from this circle to the right.

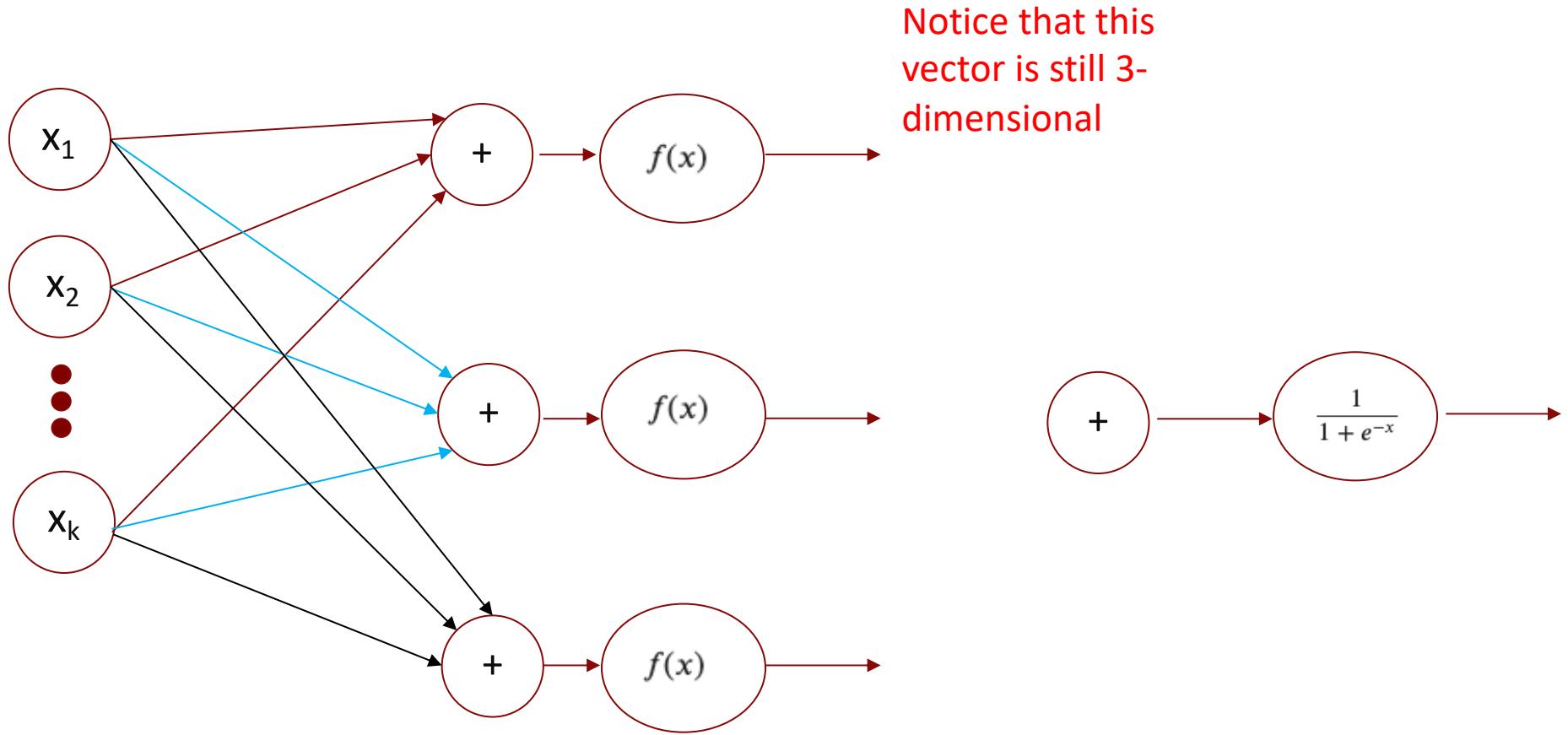
To learn smart representations, we would like to transform the inputs one or more times before we do the prediction

We can “flow” this 3-dimensional vector through another function



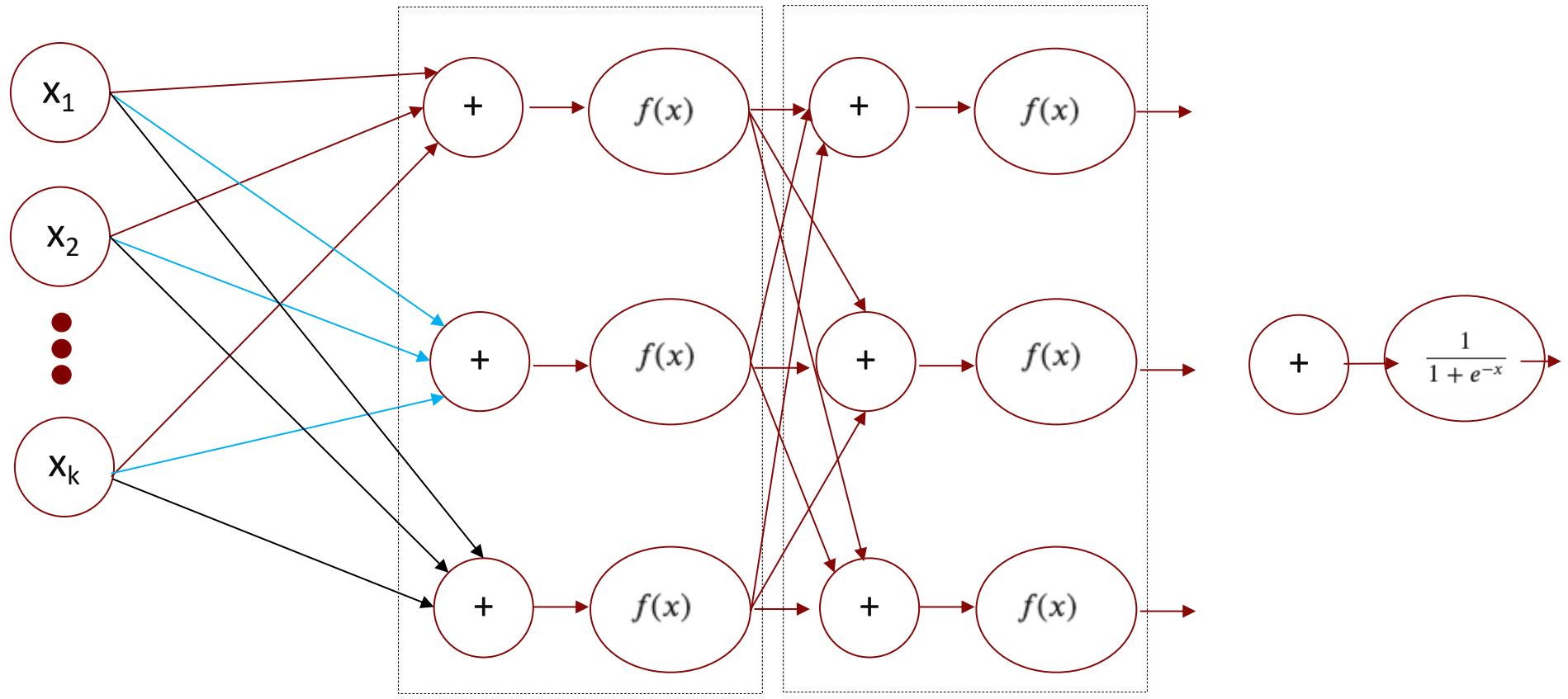
A detailed view of a single neuron's computation. It shows a summation node (circle with a plus sign) followed by a function node containing the expression  $\frac{1}{1 + e^{-x}}$ . An arrow points from the function node to the right, indicating the final output.

To learn smart representations, we would like to transform the inputs one or more times before we do the prediction

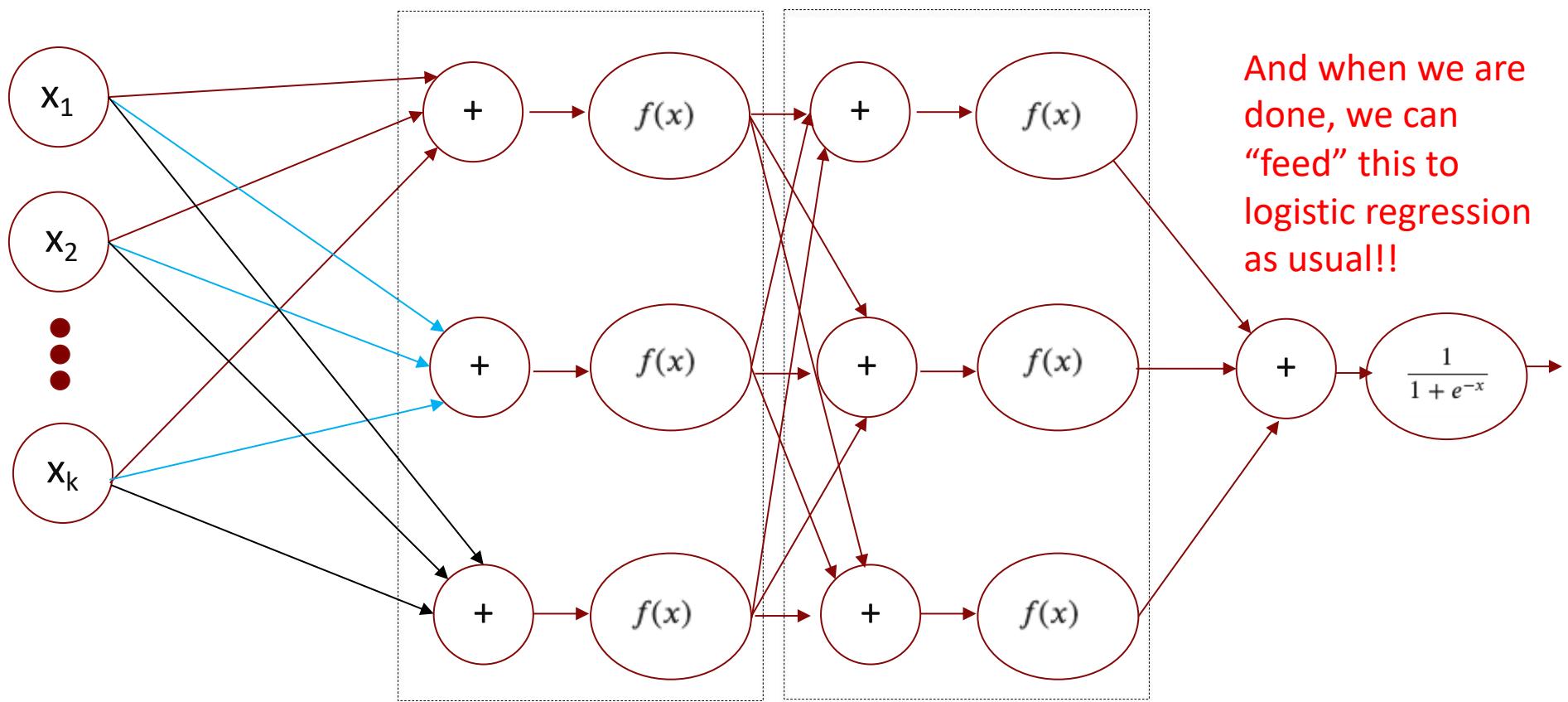


To learn smart representations, we would like to transform the inputs one or more times before we do the prediction

We can do this repeatedly

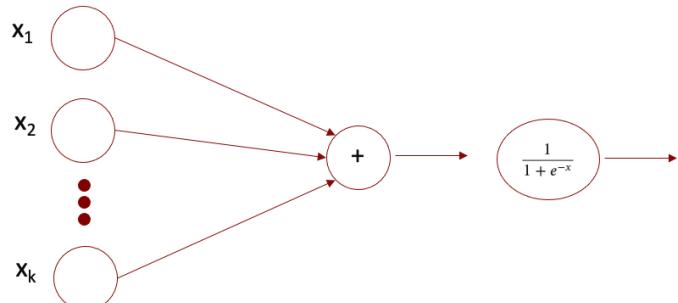


To learn smart representations, we would like to transform the inputs one or more times before we do the prediction

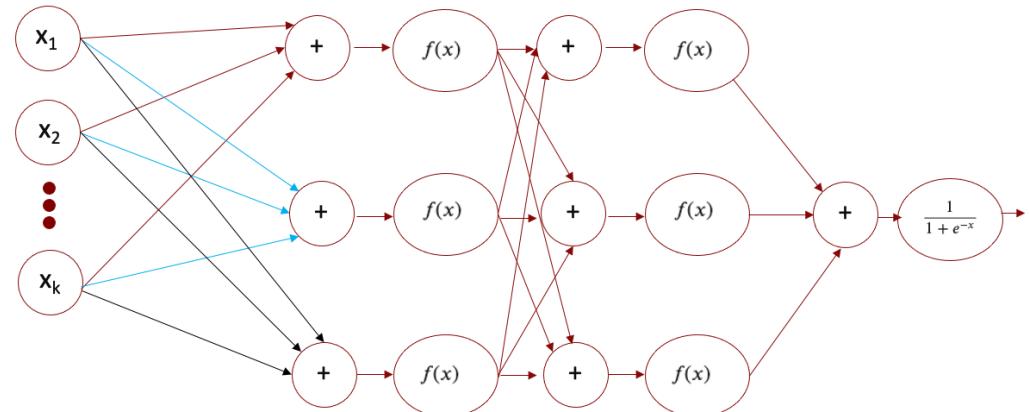


Key Takeaway: Instead of feeding the "raw" input to logistic regression, we feed a repeatedly transformed input

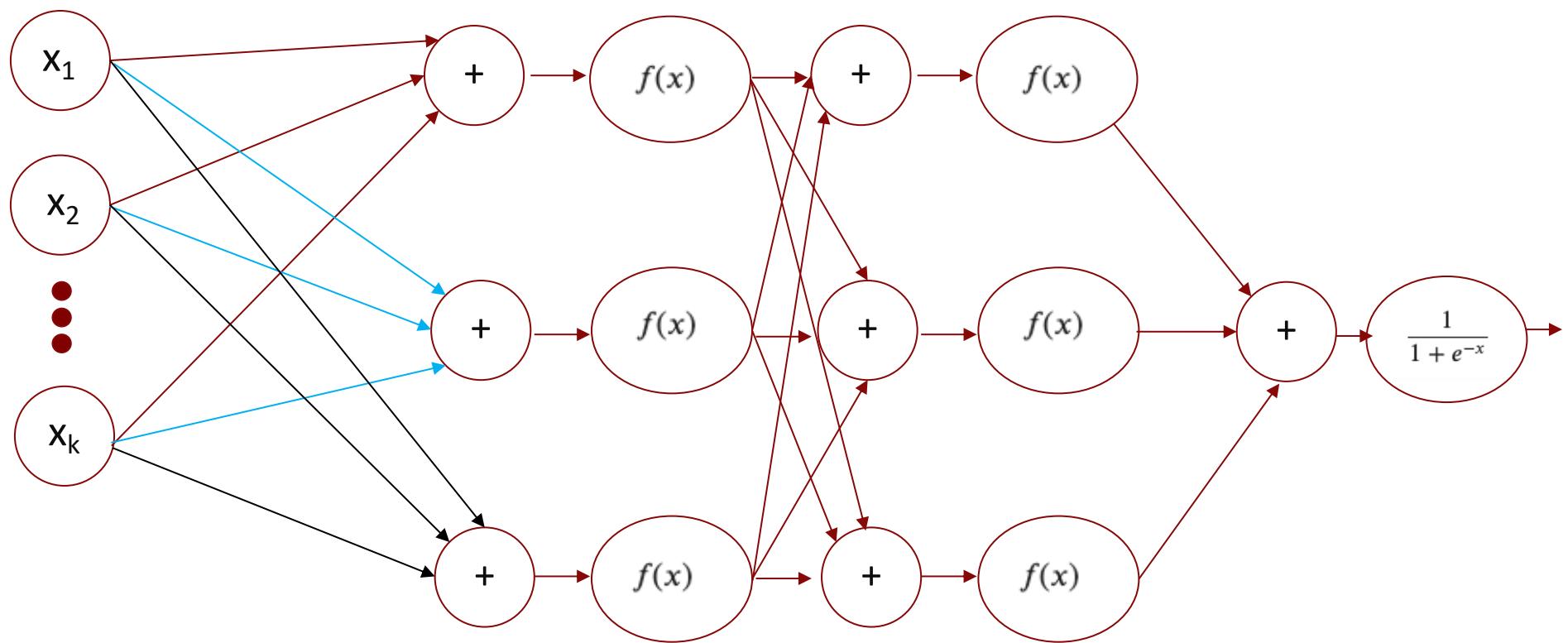
Before



After

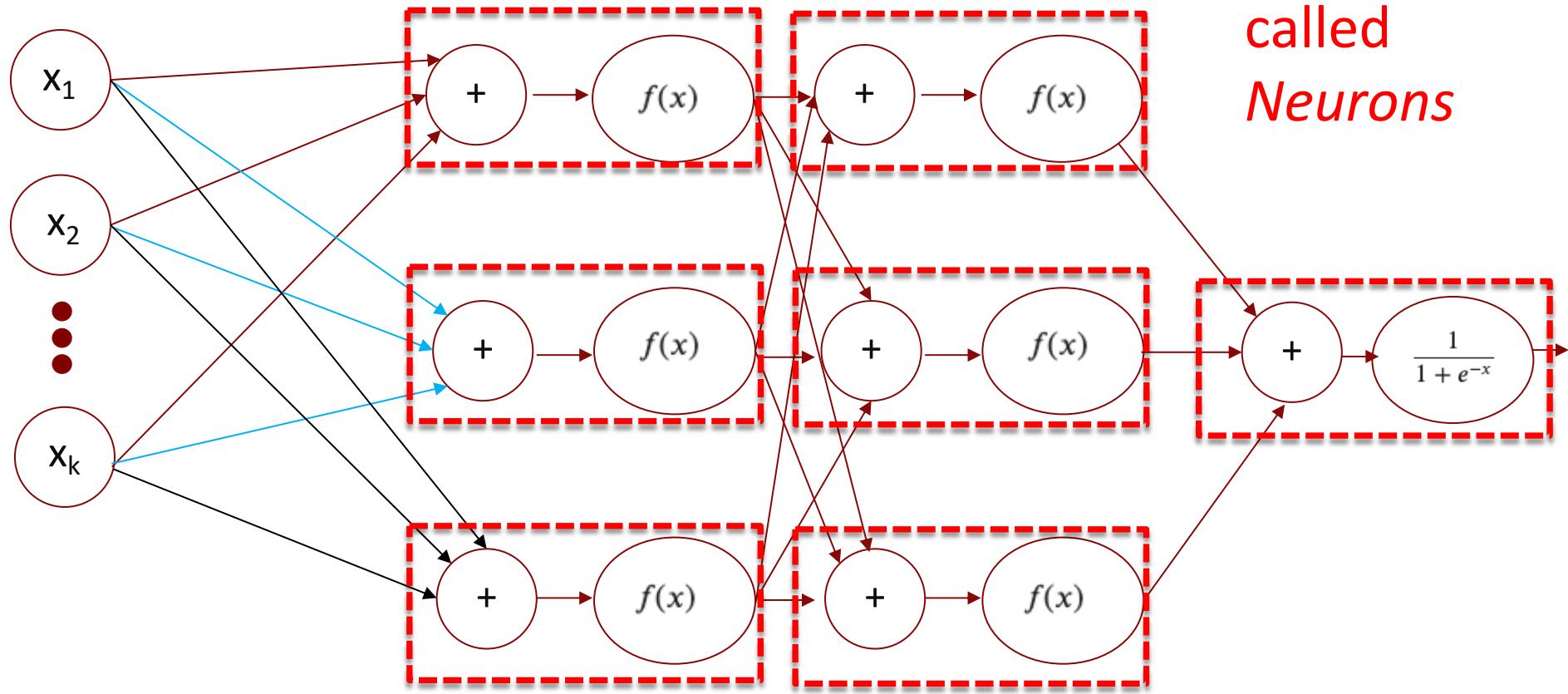


# This is a Neural Network!

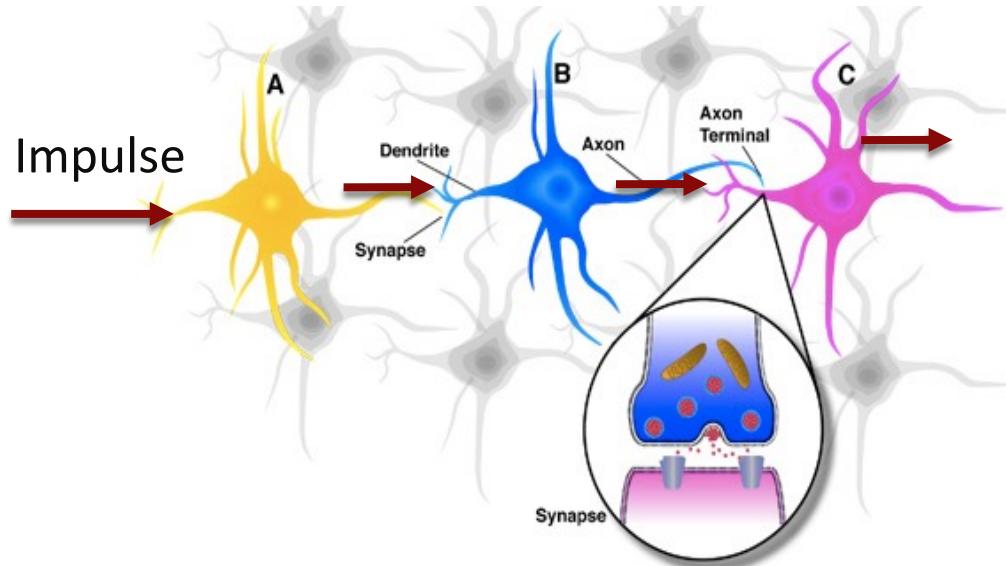


# Terminology

These are  
called  
*Neurons*

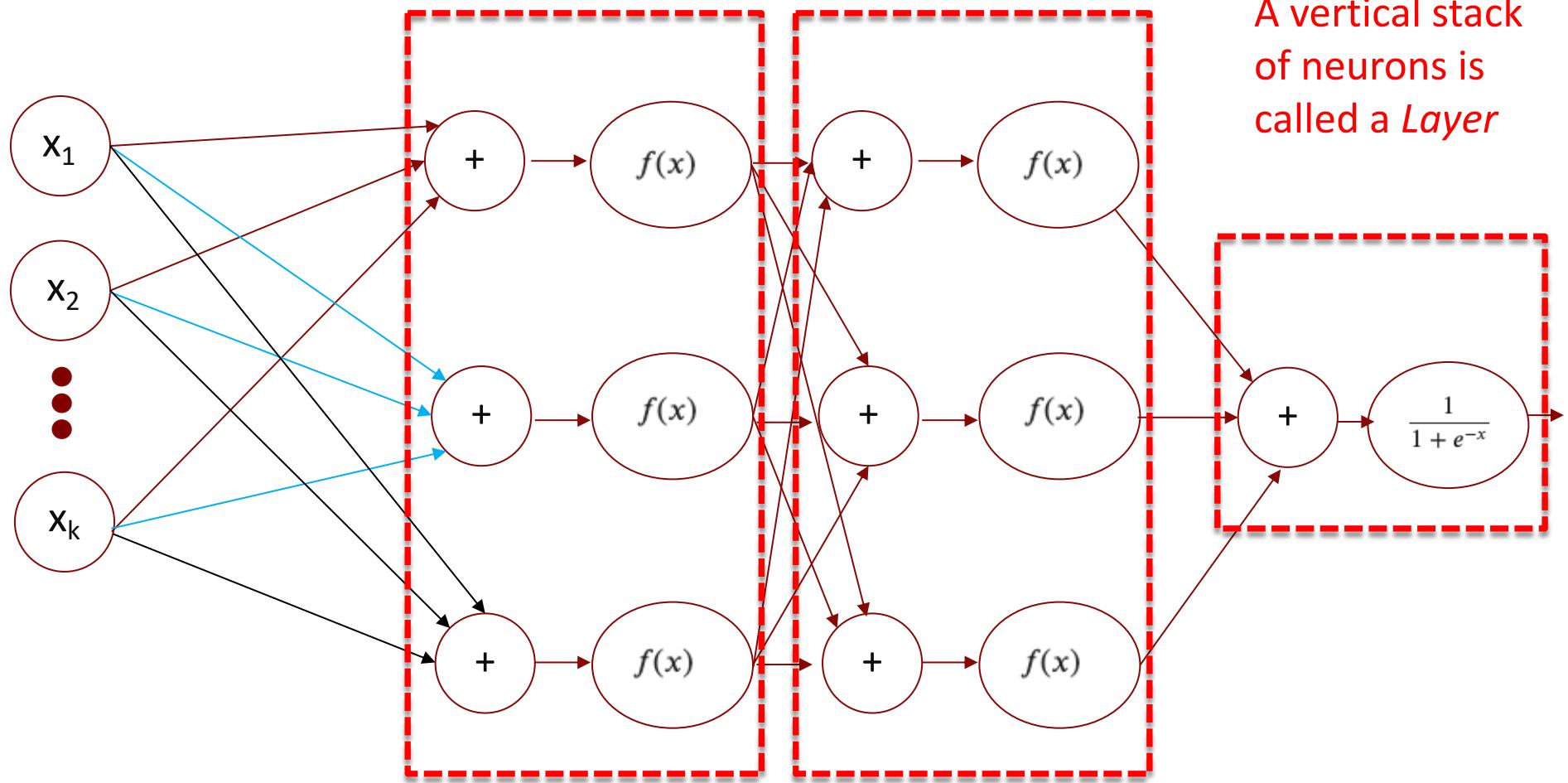


# Aside: The “neural” connection



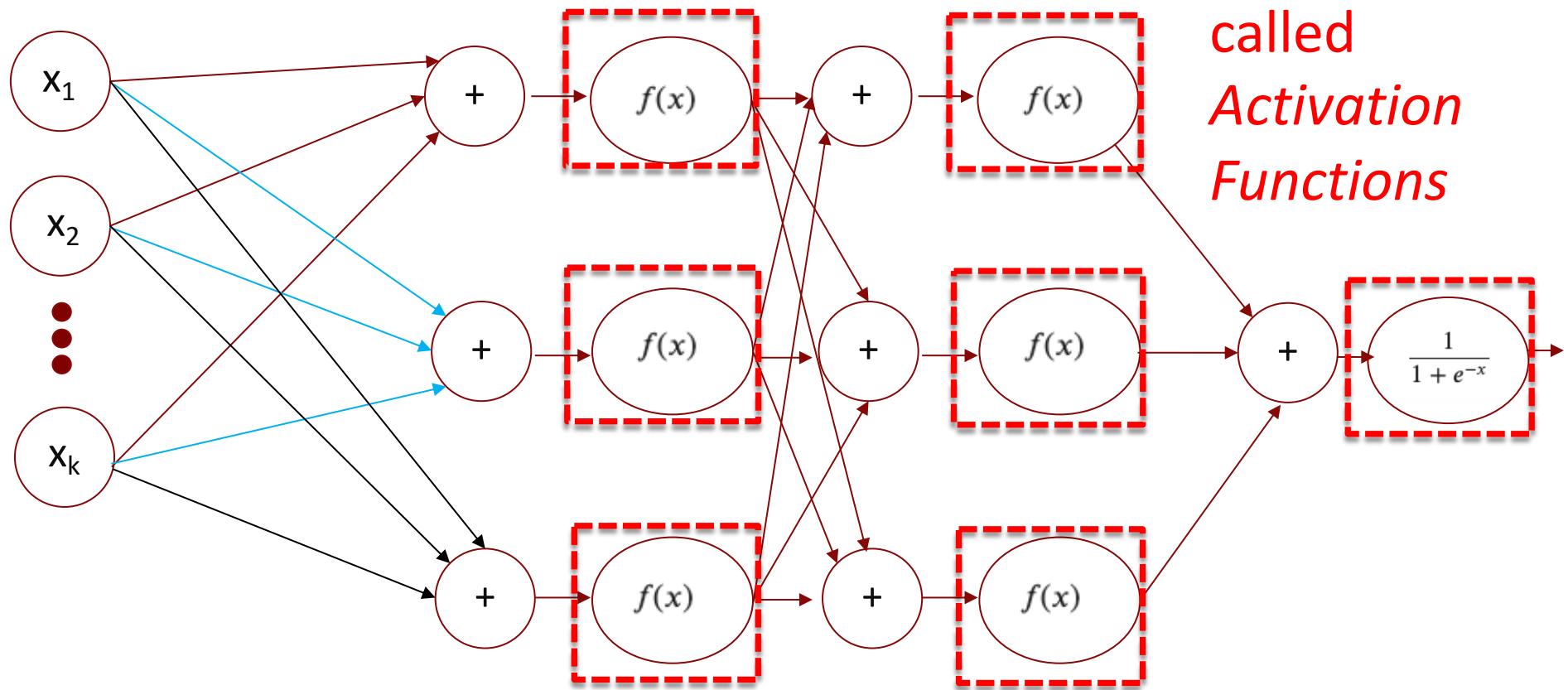
In the brain, the dendrites of one neuron connect to the axons of others at synapses, forming a network

# Terminology

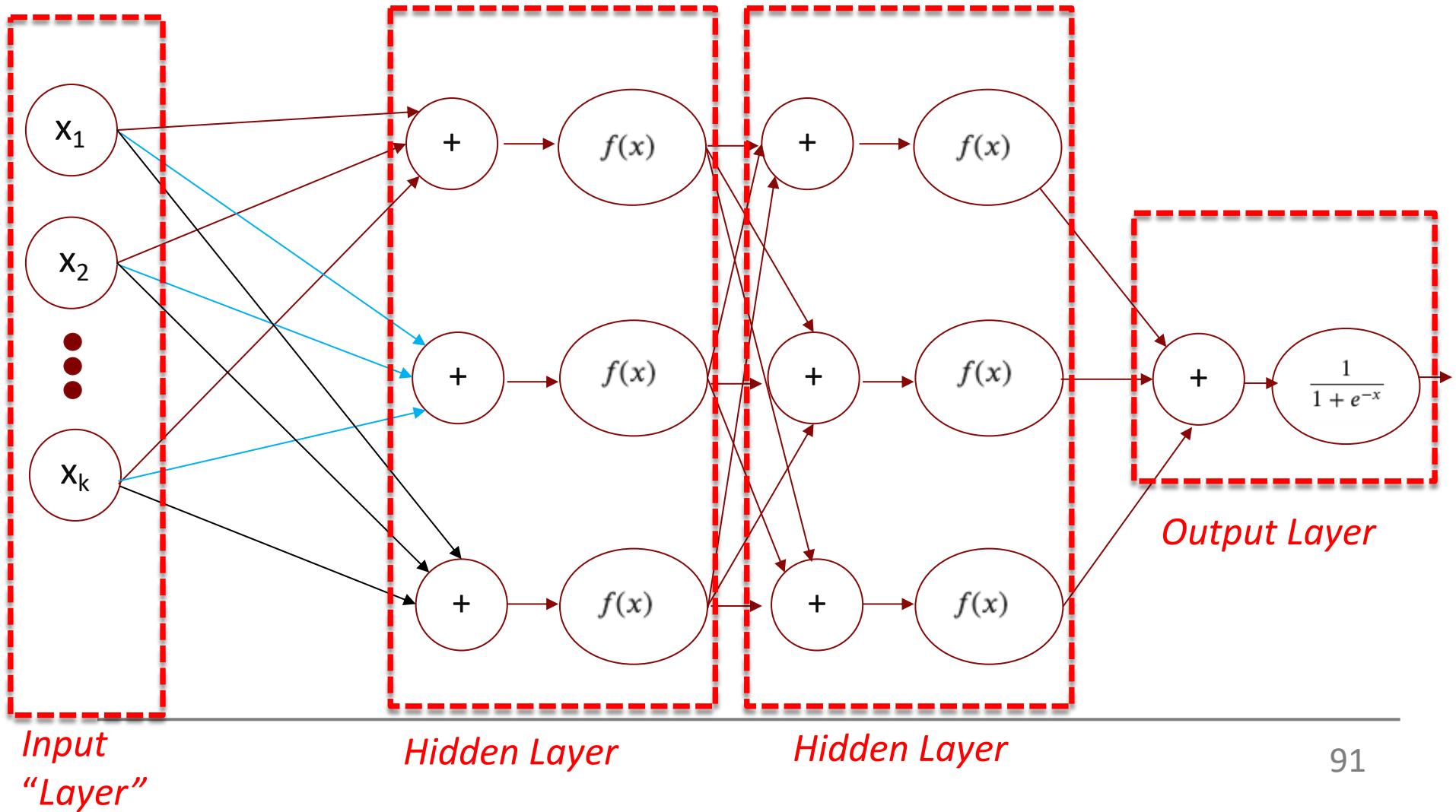


# Terminology

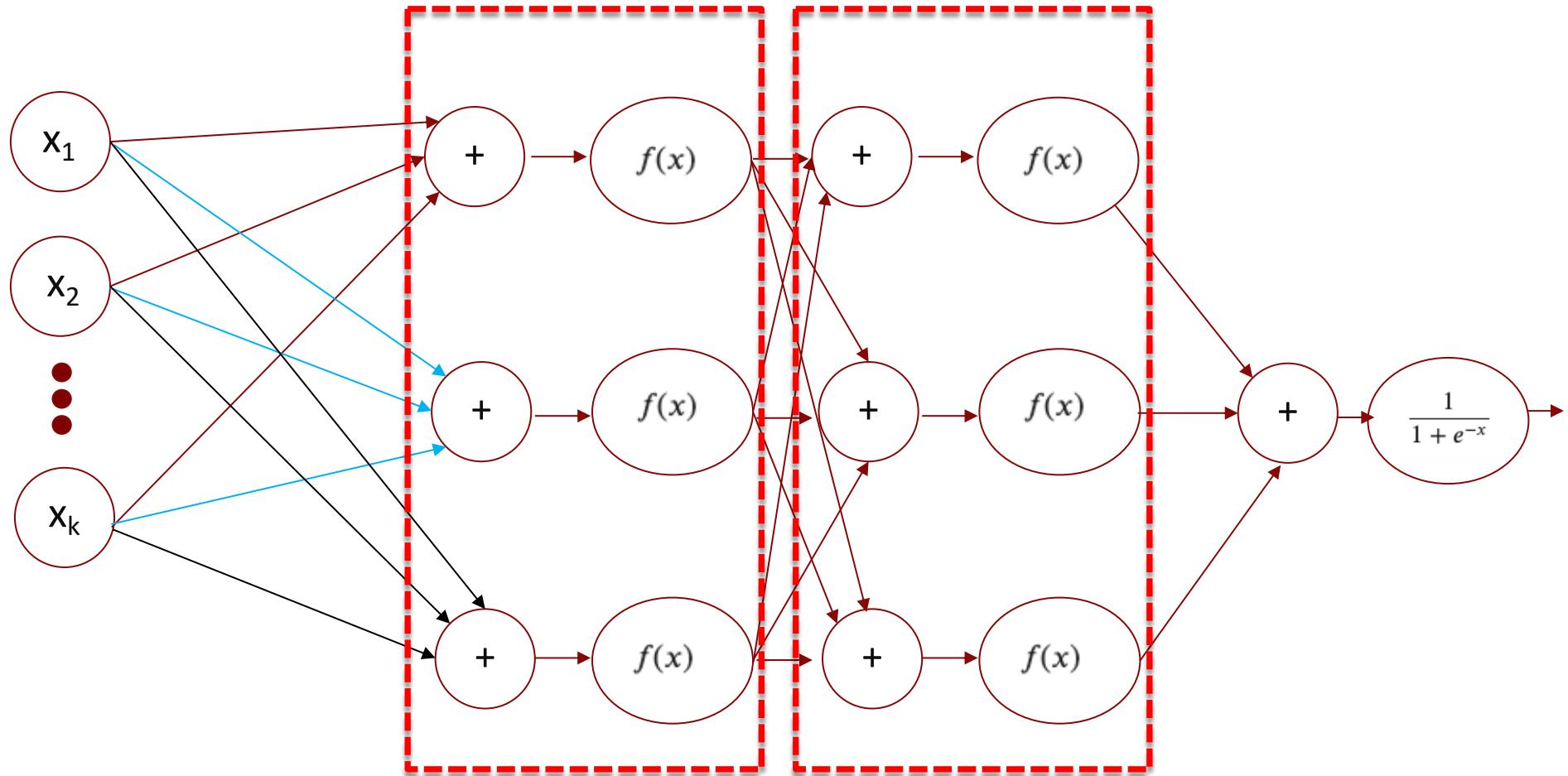
These are  
called  
*Activation  
Functions*



# Terminology

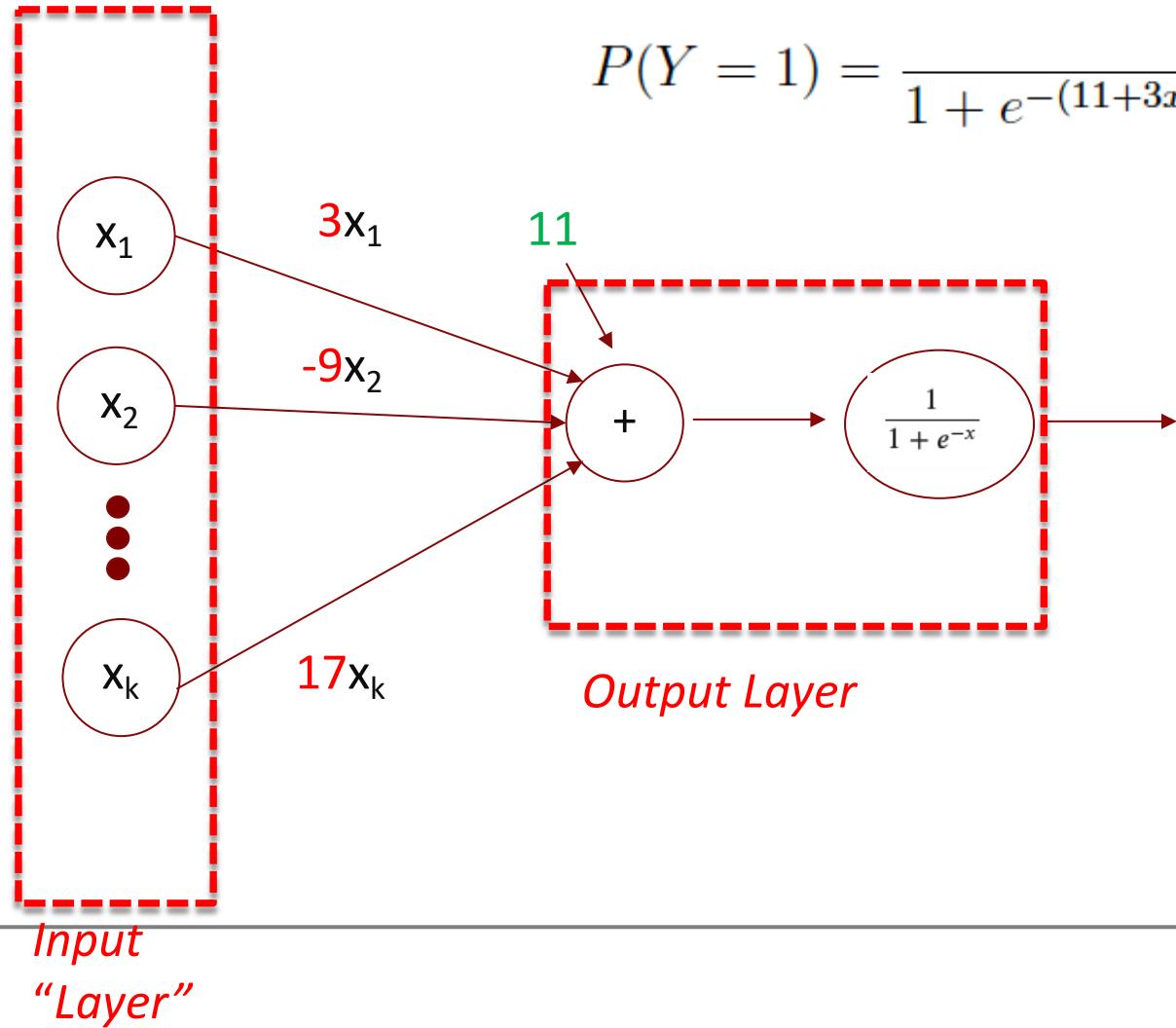


# Terminology



When every neuron in a layer is connected to every neuron in the next layer, it is called Dense or Fully Connected 92

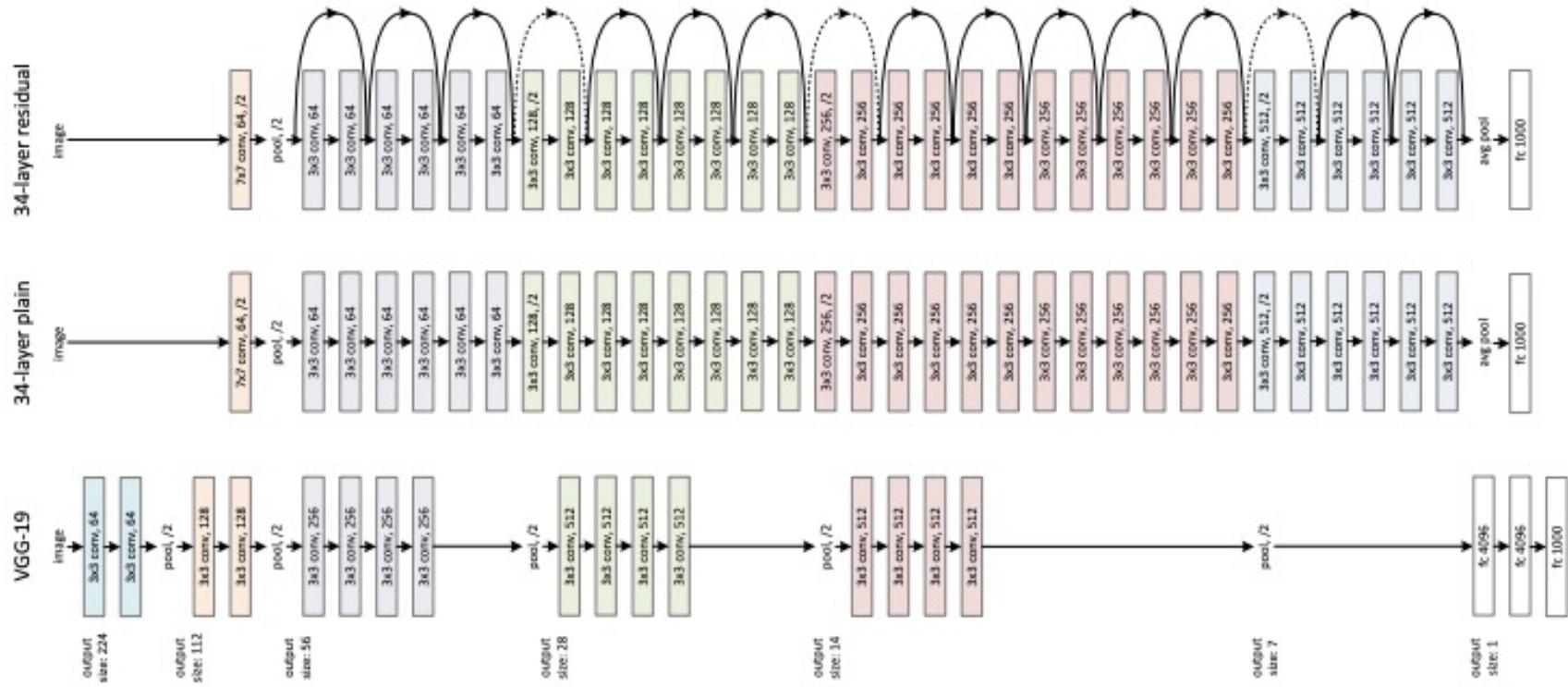
The general logistic regression model is an NN  
but a simple one since it has no hidden layers



$$P(Y = 1) = \frac{1}{1 + e^{-(11 + 3x_1 - 9x_2 + \dots + 17x_k)}}$$

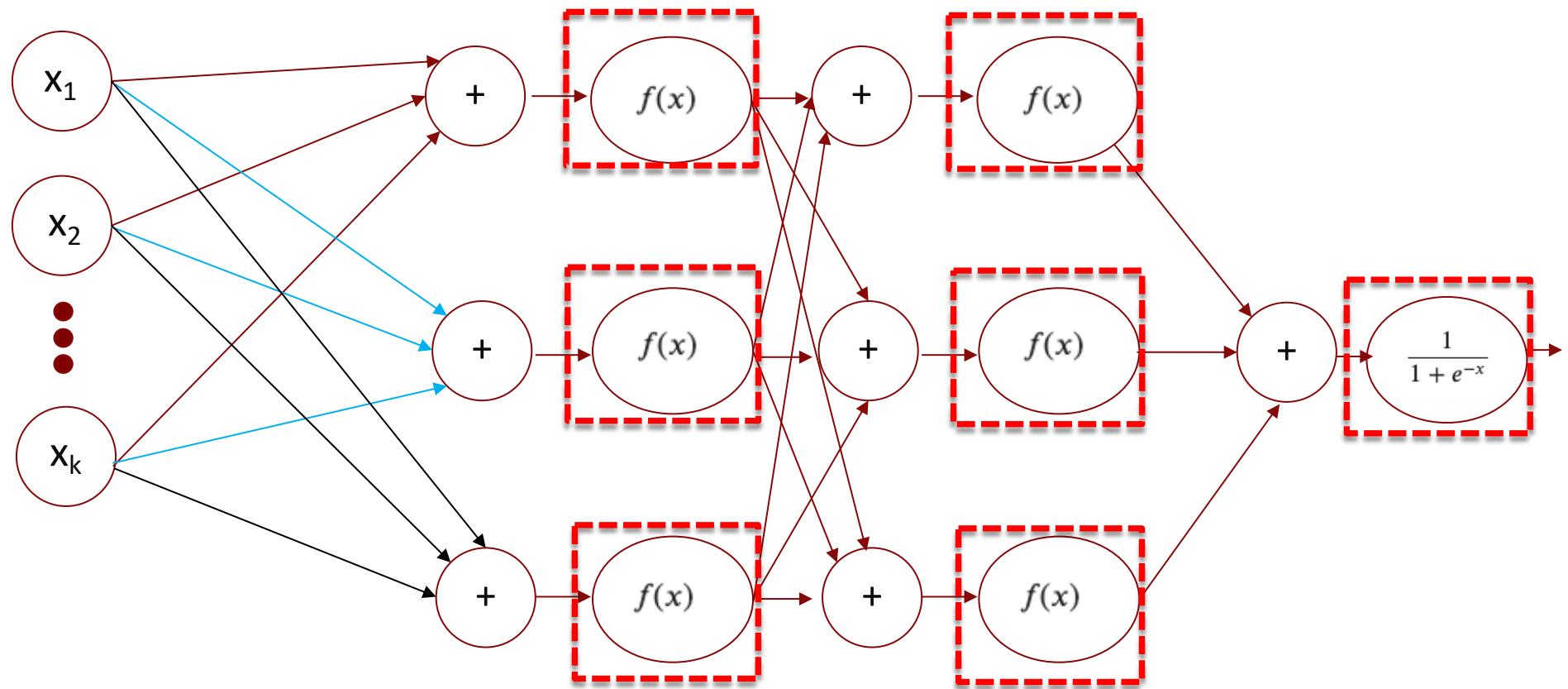
Deep Learning is *just* neural networks with lots and lots of ...

# Deep Learning is just neural networks with lots and lots of hidden layers



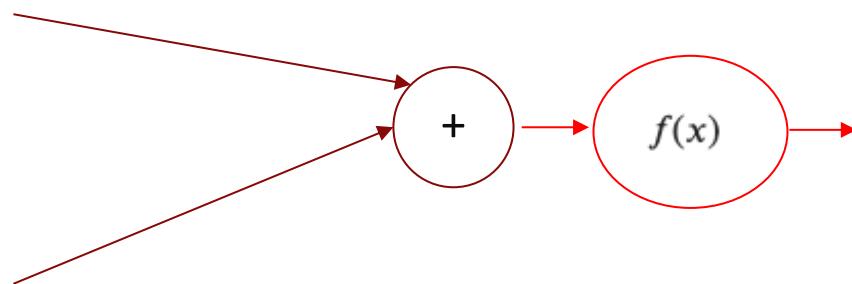
<https://arxiv.org/pdf/1512.03385.pdf>

# Let's now turn to Activation Functions



# Activation functions

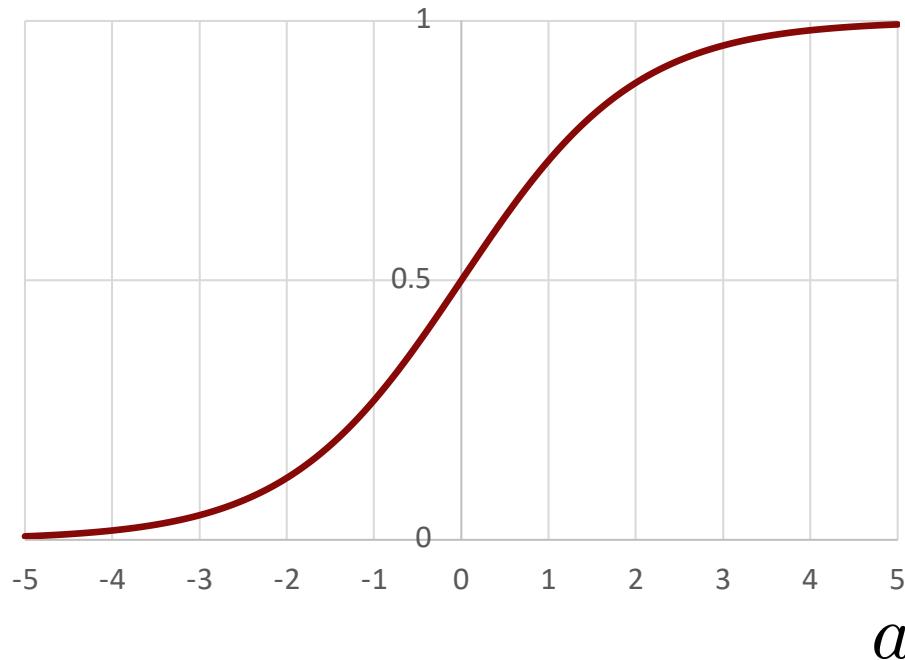
The activation function of a node is just a function that receives a single number and outputs a single number (i.e., scalar in -> scalar out)



# Common Activation Functions

Sigmoid activation function:

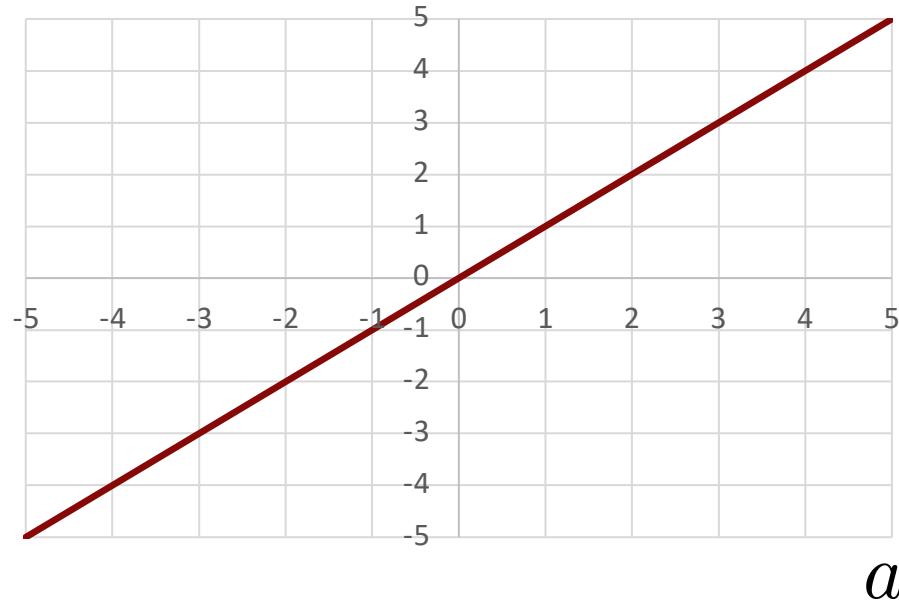
$$\sigma(a) = \frac{1}{1 + e^{-a}}$$



# Common Activation Functions

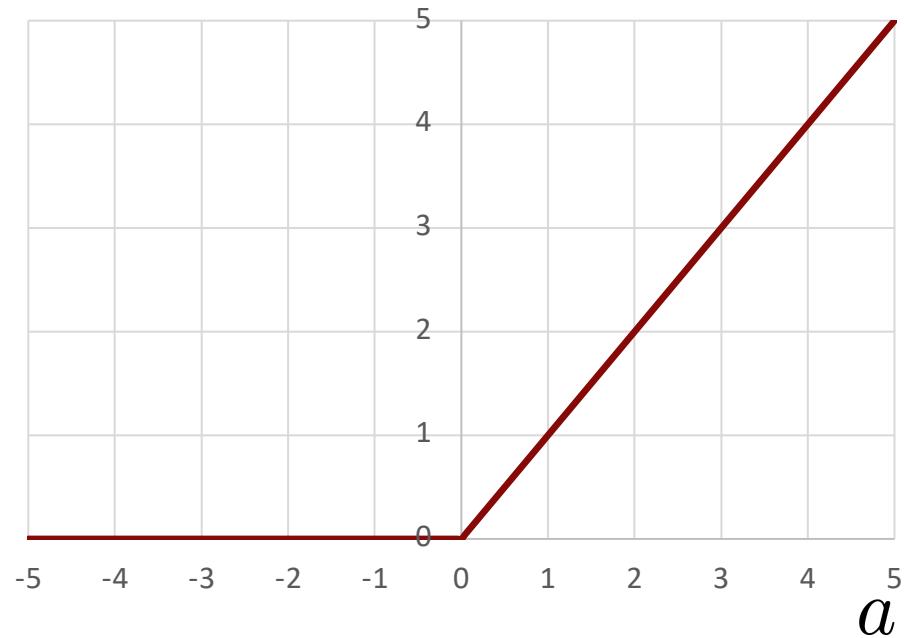
Linear activation function:

$$f(a) = a$$

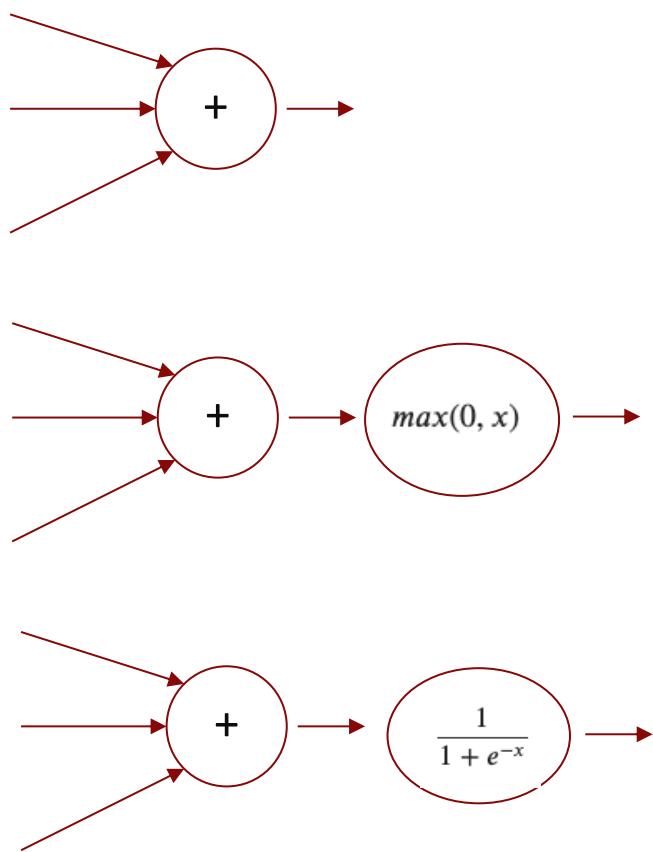


# Common Activation Functions

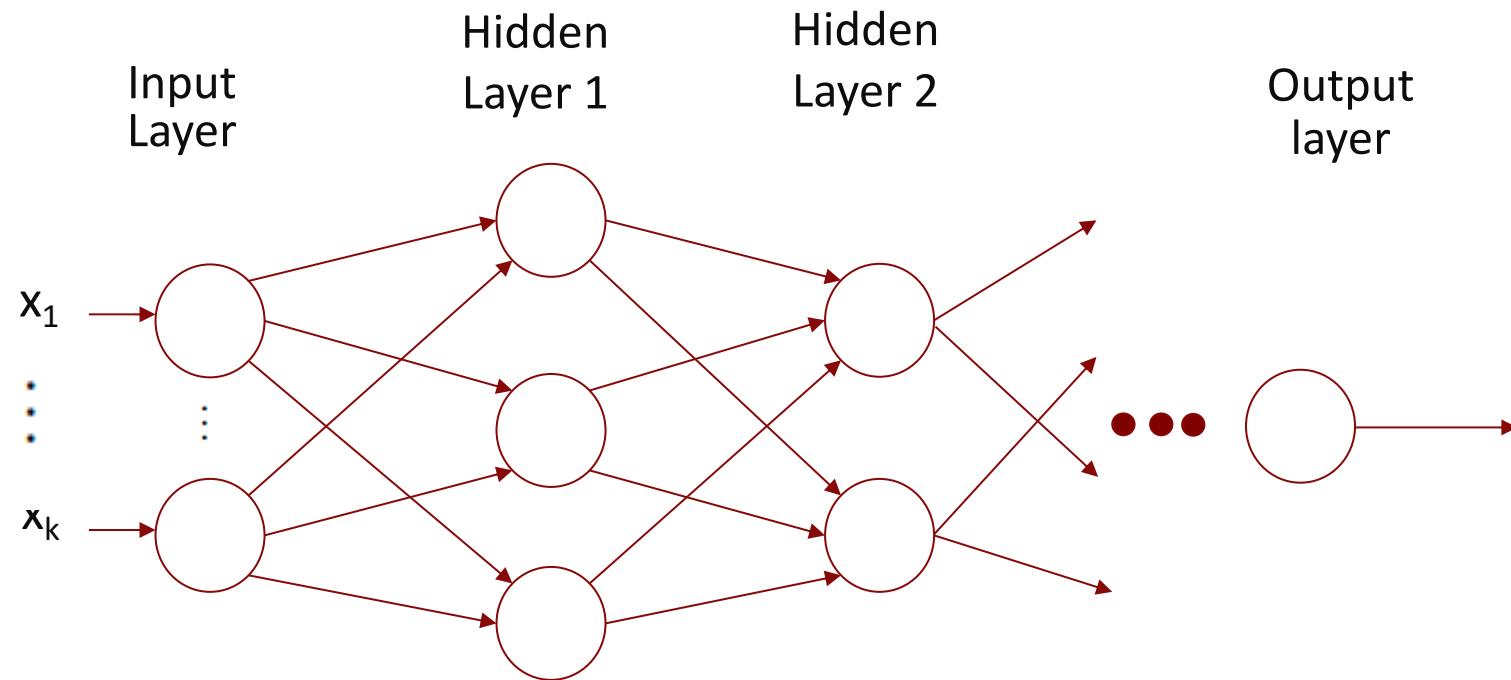
ReLU: 
$$g(a) = \max(0, a)$$
  
("Rectified Linear Unit")



# We will use this “visual shorthand” from now on



# Recap: Designing a DNN



User chooses the # of hidden layers, # units in each layer, the activation function(s) for the hidden layers and for the output layer

# Let's use a DNN for our interview classifier

- Recall the problem:
  - Two input variables (i.e., GPA and experience)
  - An output variable that should be between 0 and 1

User chooses the # of hidden layers, # units in each layer, the activation function(s) for the hidden layers and for the output layer

# Let's use a DNN for our interview classifier

- Recall the problem:
  - Two input variables (i.e., GPA and experience)
  - An output variable that should be between 0 and 1
- Design choices
  - We will use **one hidden layer** with **3 neurons (ReLU)**
  - Since the output is constrained to be in  $(0,1)$ , we will use the **sigmoid for the output layer**

User chooses the # of hidden layers, # units in each layer, the activation function(s) for the hidden layers and for the output layer

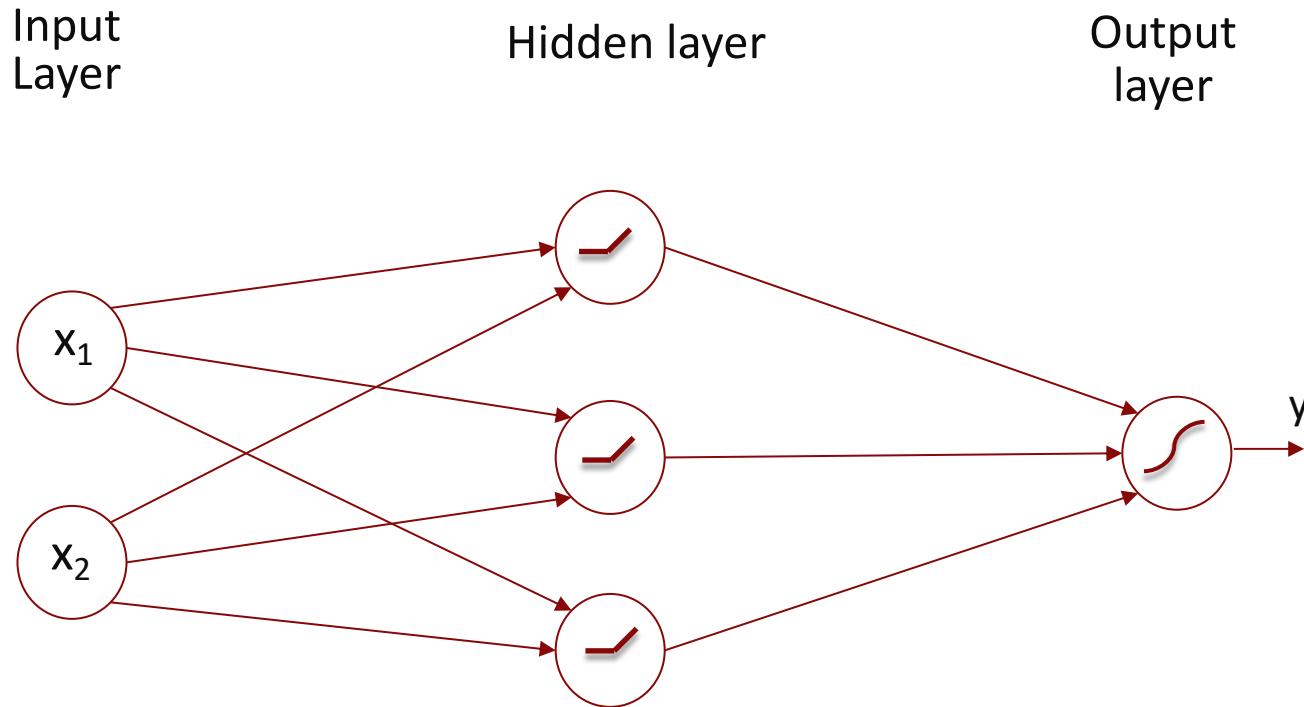
# Let's practice setting up a simple NN

Input  
Layer

Hidden layer

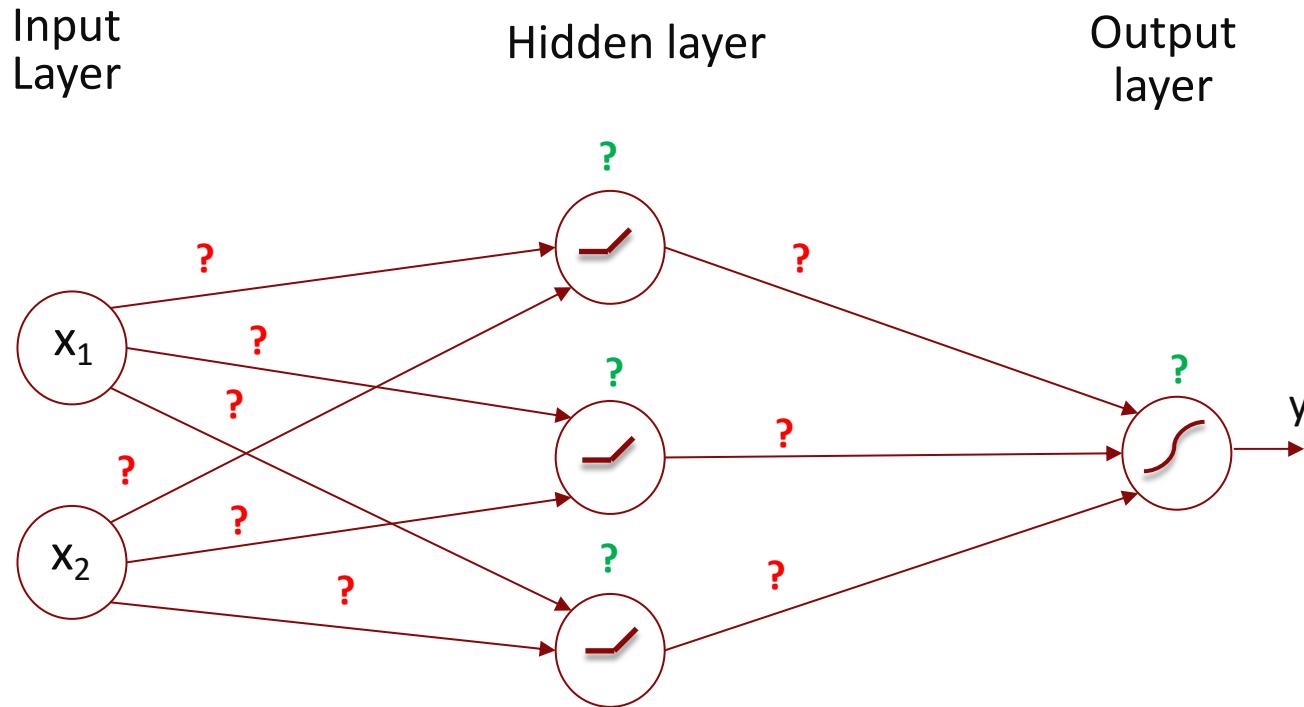
Output  
layer

# Let's practice setting up a simple NN



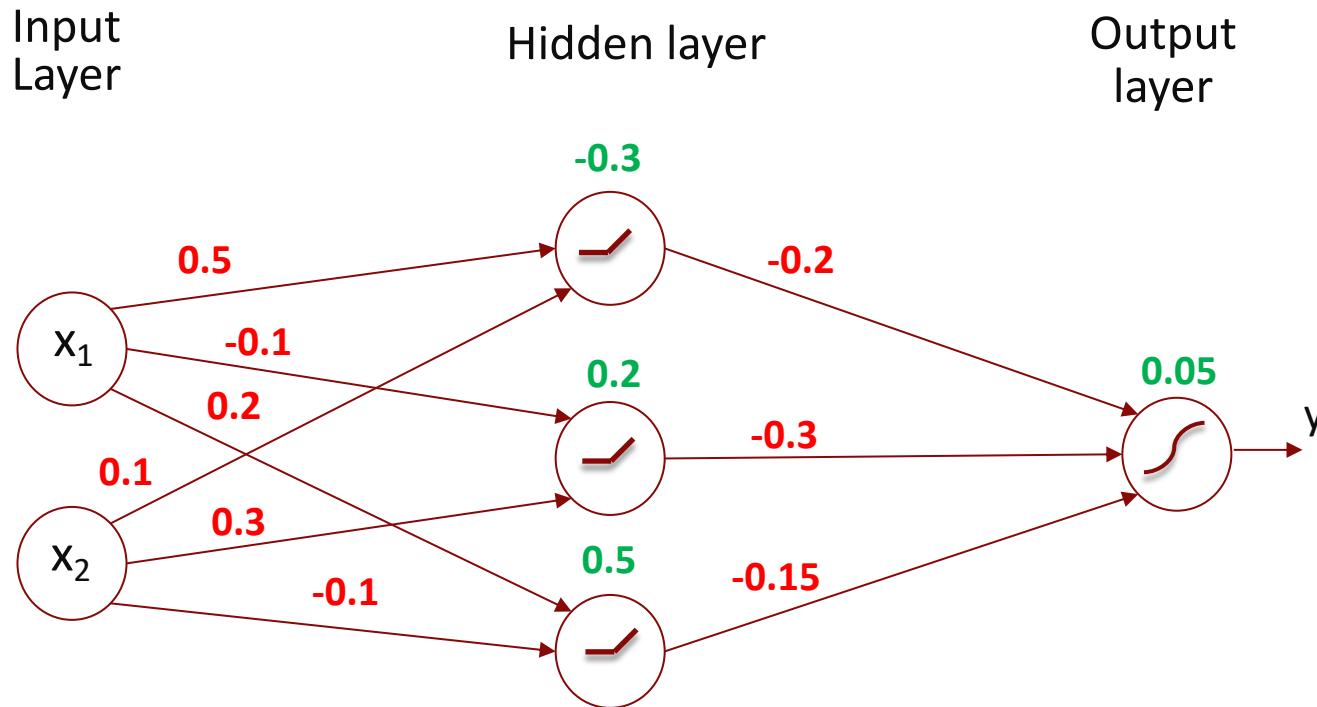
How many parameters (i.e., weights and biases) does this network have?

# Let's practice setting up a simple NN



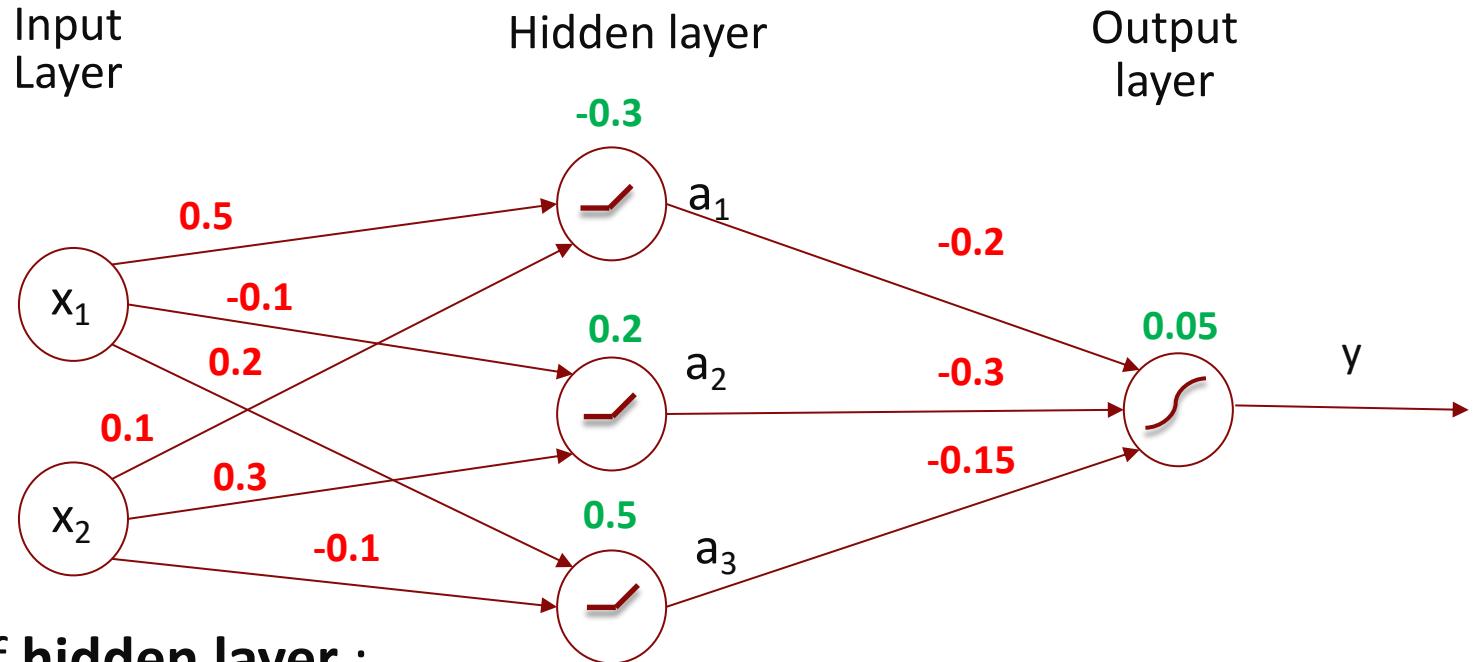
How many parameters (i.e., weights and biases) does this network have? **13**

Let's assume that we have trained\* this network on data and have found these values for the parameters



\*details in the next class

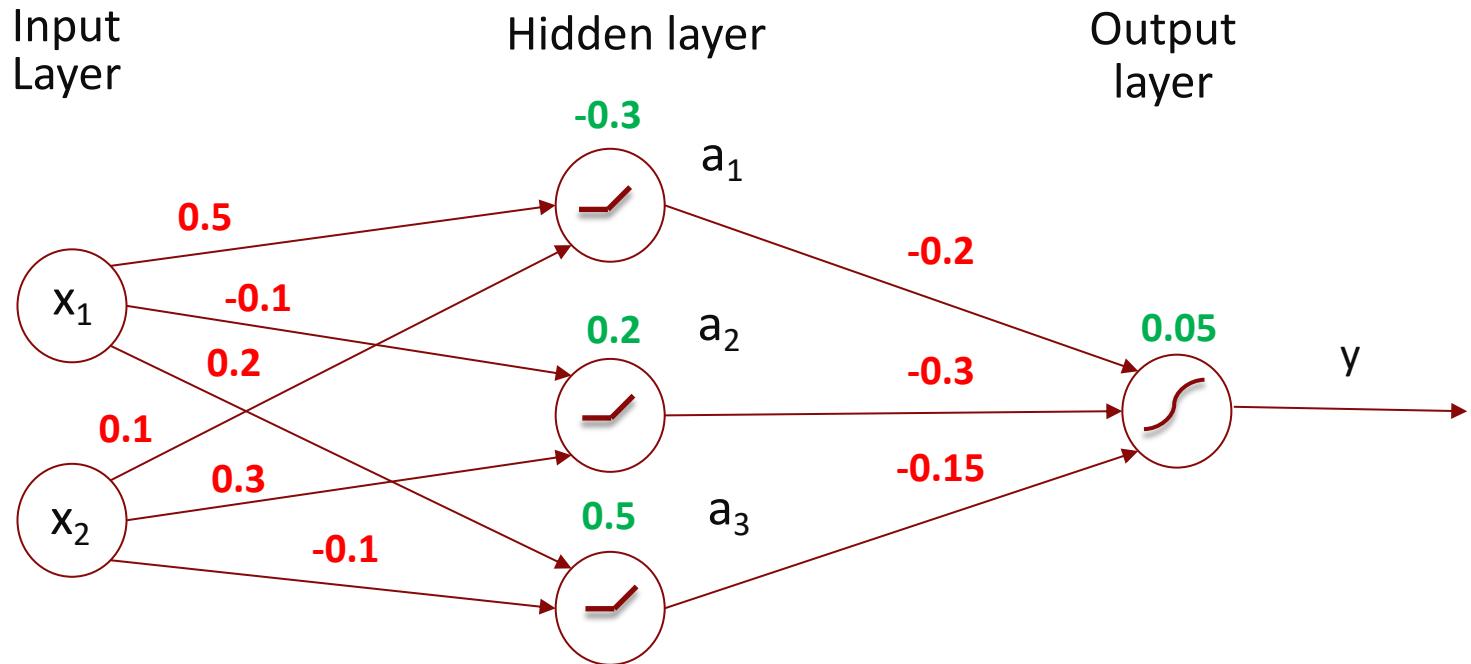
# Predicting with the NN



**Output of hidden layer :**

- Top node:  $\max(0, -0.3 + 0.5x_1 + 0.1x_2) = a_1$
- Middle node:  $\max(0, 0.2 - 0.1x_1 + 0.3x_2) = a_2$
- Bottom node:  $\max(0, 0.5 + 0.2x_1 - 0.1x_2) = a_3$

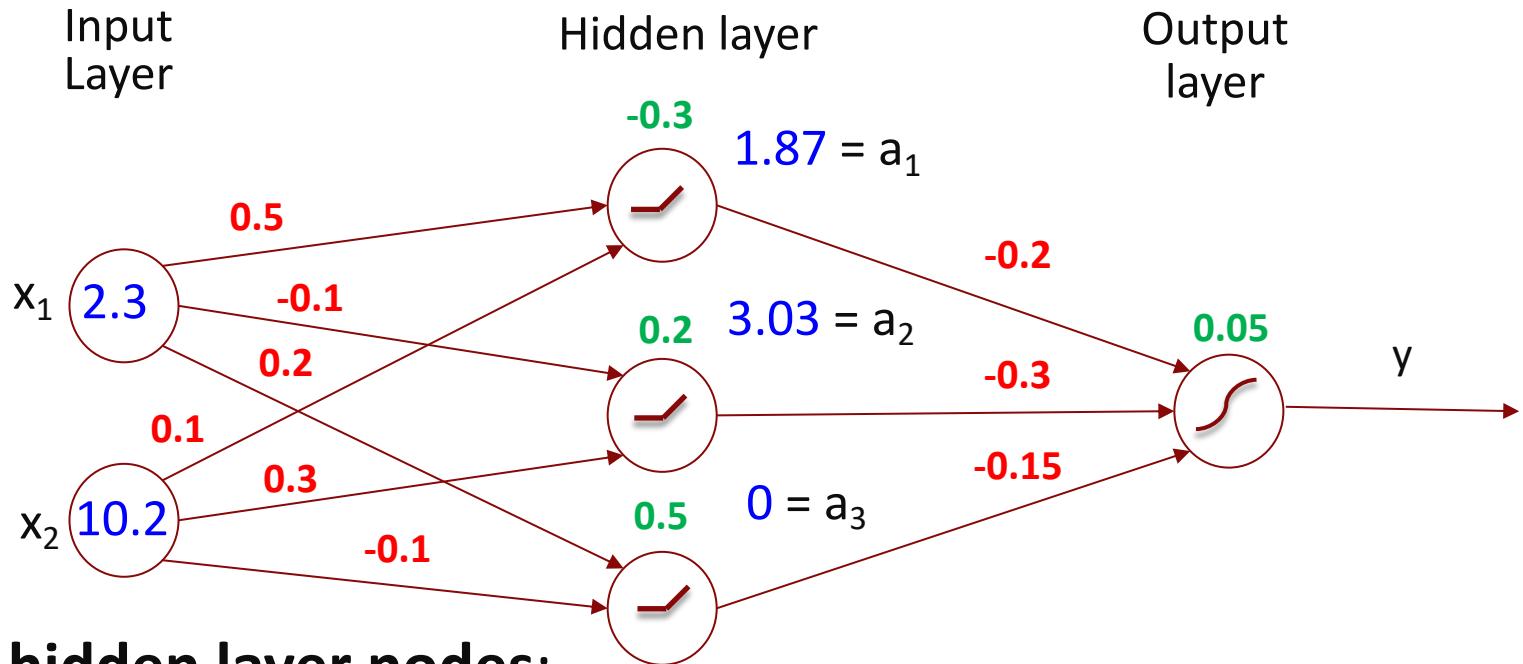
# Predicting with the NN



- Recall  $a_1$ ,  $a_2$ , and  $a_3$  are the output of the hidden layer nodes
- Output of **output layer node**:

$$\frac{1}{1 + e^{-(0.05 - 0.2a_1 - 0.3a_2 - 0.15a_3)}}$$

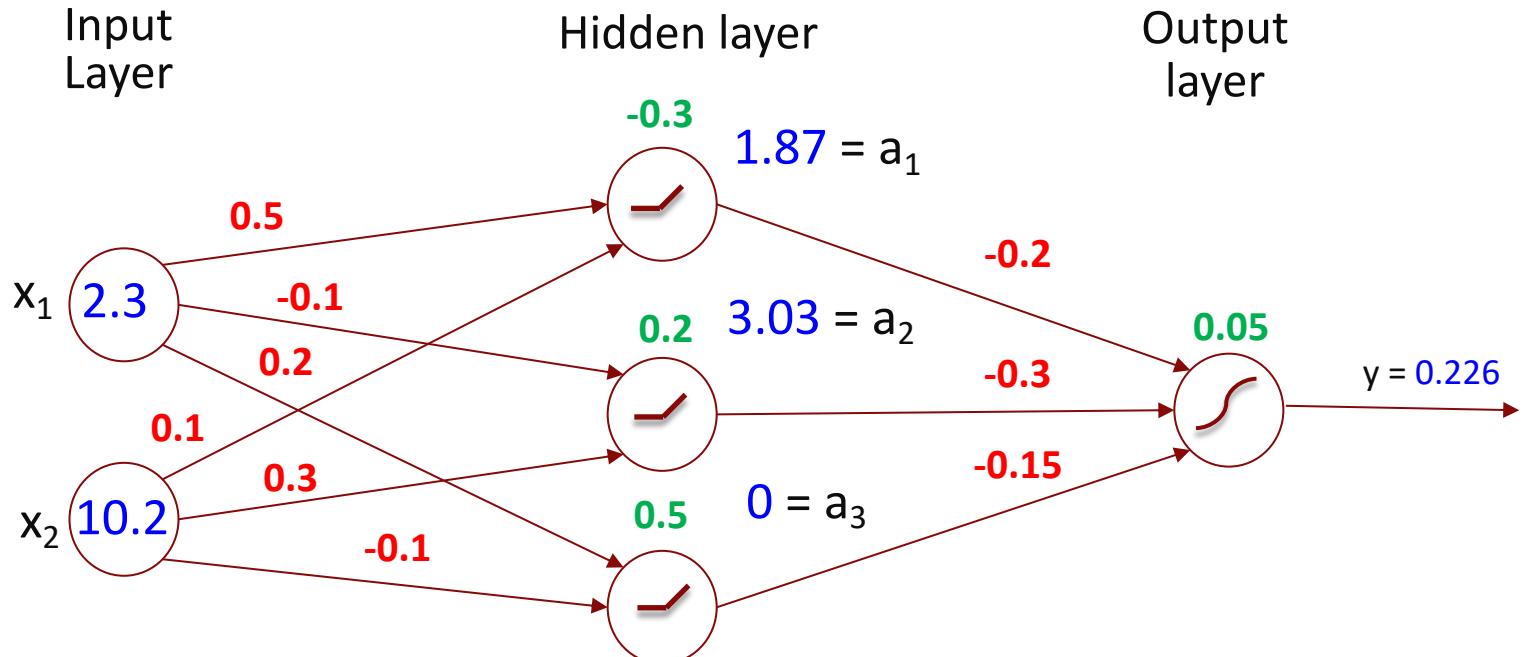
# Predicting with the NN



**Output of hidden layer nodes:**

- Top node:  $\max(0, -0.3 + 0.5*2.3 + 0.1*10.2) = 1.87 = a_1$
- Middle node:  $\max(0, 0.2 - 0.1*2.3 + 0.3*10.2) = 3.03 = a_2$
- Bottom node:  $\max(0, 0.5 + 0.2*2.3 - 0.1*10.2) = 0 = a_3$

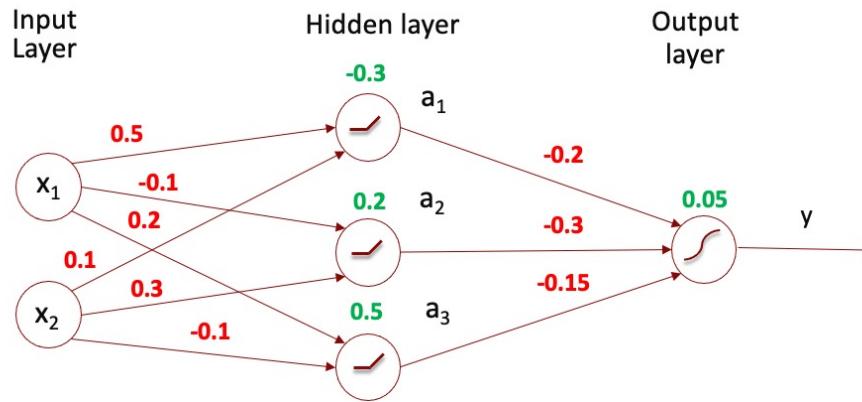
# Predicting with the NN



Output of **output layer node**:

$$\frac{1}{1 + e^{-(0.05 - 0.2 \cdot 1.87 - 0.3 \cdot 3.03 - 0.15 \cdot 0)}} = 0.226$$

# The Network can be written as this function

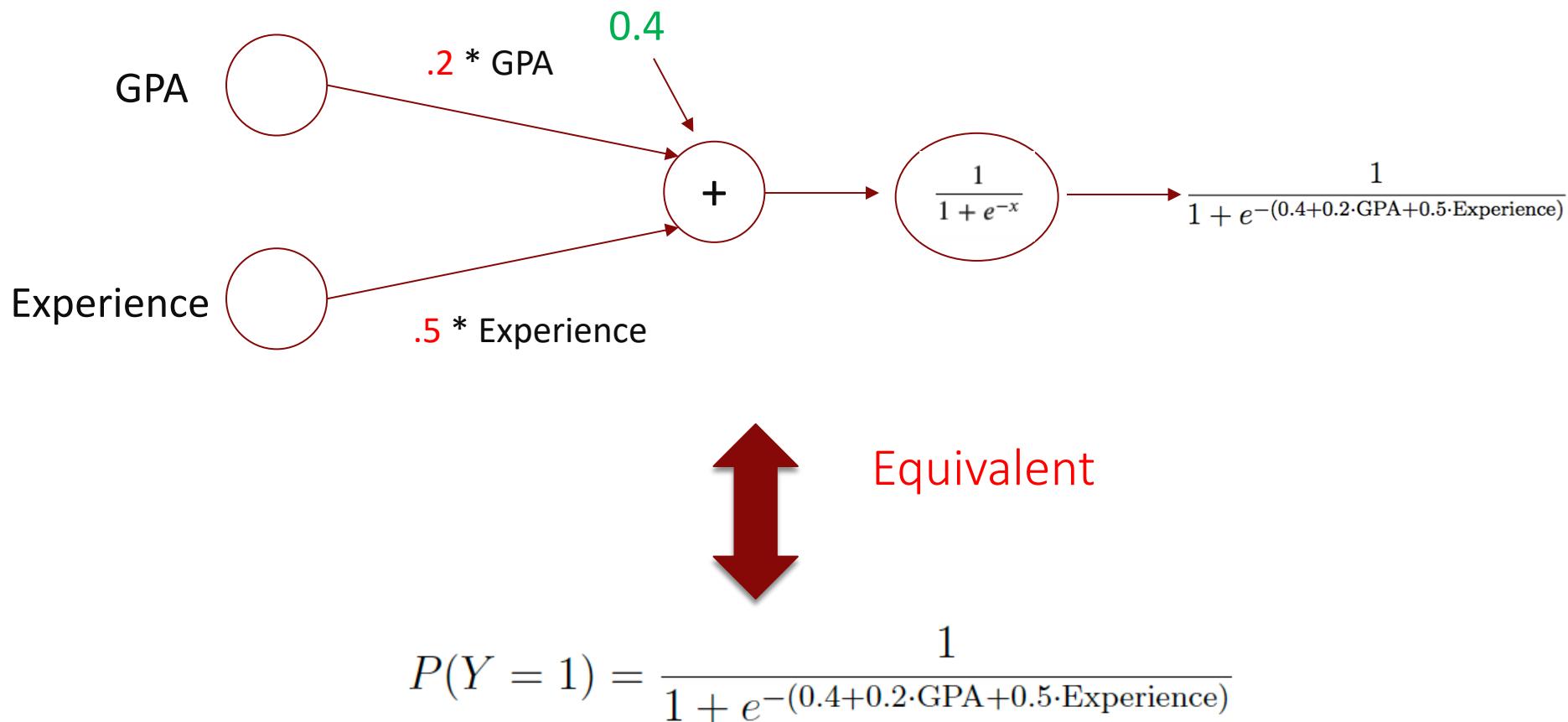


Equivalent

1

$$y = \frac{1}{1 + e^{-(.05 - 0.2(\max(0, -0.3 + 0.5x_1 + 0.1x_2)) - 0.3(\max(0, 0.2 - 0.1x_1 + 0.3x_2)) - 0.15(\max(0.5 + 0.2x_1 - 0.1x_2)))}}$$

# Contrast with the Logistic Regression model from before



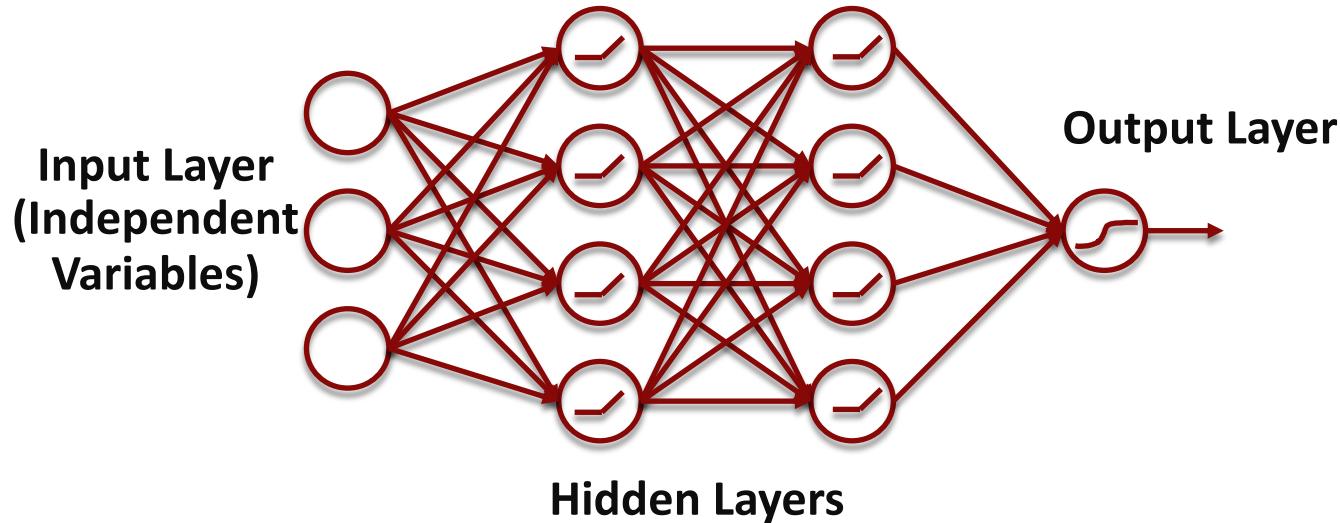
Note the complexity of even this simple network compared to the logistic regression model

$$y = \frac{1}{1 + e^{-(.05 - 0.2(\max(0, -0.3 + 0.5x_1 + 0.1x_2)) - 0.3(\max(0, 0.2 - 0.1x_1 + 0.3x_2)) - 0.15(\max(0.5 + 0.2x_1 - 0.1x_2)))}}$$

$$y = \frac{1}{1 + e^{-(0.4 + 0.2x_1 + 0.5x_2)}}$$

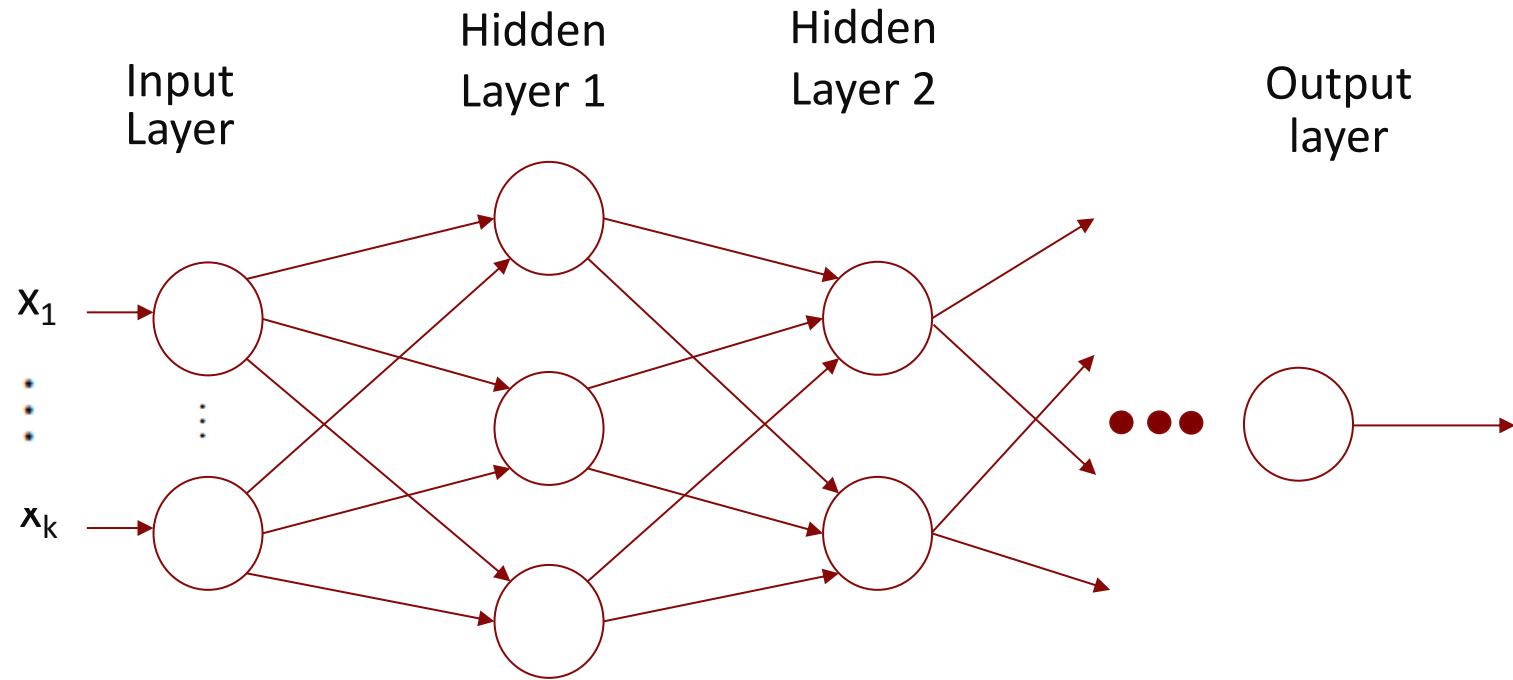
$y$  is a much more complex function of its inputs  $x_1$  and  $x_2$  compared to (say)  $\frac{1}{1 + e^{-(0.4 + 0.2 \cdot \text{GPA} + 0.5 \cdot \text{Experience})}}$  and it can capture more complex relationships between  $x$  and  $y$ .

# Summary: A Deep Neural Network



- This is a **feedforward** (or **vanilla**) neural network
- In general, the arrangement of neurons into layers, the activation functions, and the connections between layers are referred to as the network's **architecture**

# Summary: Designing a DNN



User chooses the # of hidden layers, # units in each layer, the activation function(s) for the hidden layers and for the output layer