NNSE 784
Advanced Analytics Methods

Instructor: F Doyle (CESTM L210)

MW 4:30 – 5:50, NFN 203

Slide Set #19
Support Vector Machine
&
Multi-layer Perceptron

Lecture Outline

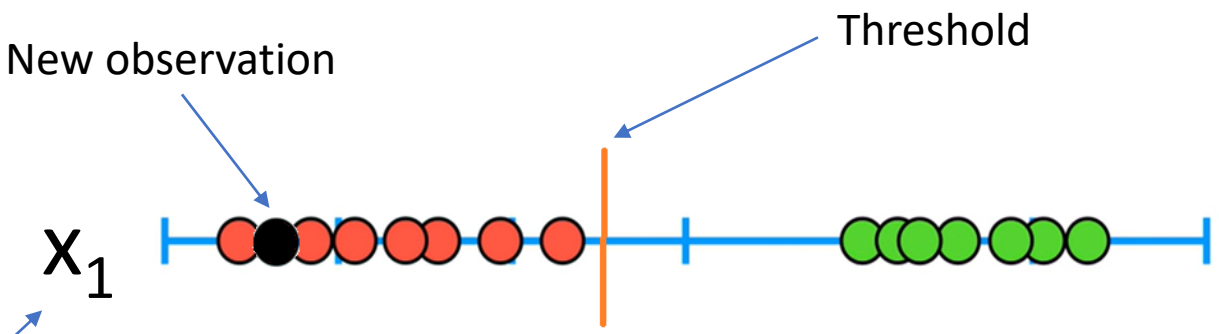
- Support Vector Machine
  - The “gist”
  - Dimensionality Increase
  - Kernel trick
- Multi-layer Perceptron
  - Neurons
  - Neural networks

One Dimensional Classification

Predictor variable (e.g., “fasting glucose level”, “gas flow rate at some PCVD process step”, “income”, etc.)

- Category 1 = (e.g. “will develop diabetes”, “wafer will fail”, “customer will quit service”, etc.)
- Category 2 = (e.g. “won’t develop diabetes”, “wafer will pass”, “customer will keep subscription”, etc.)

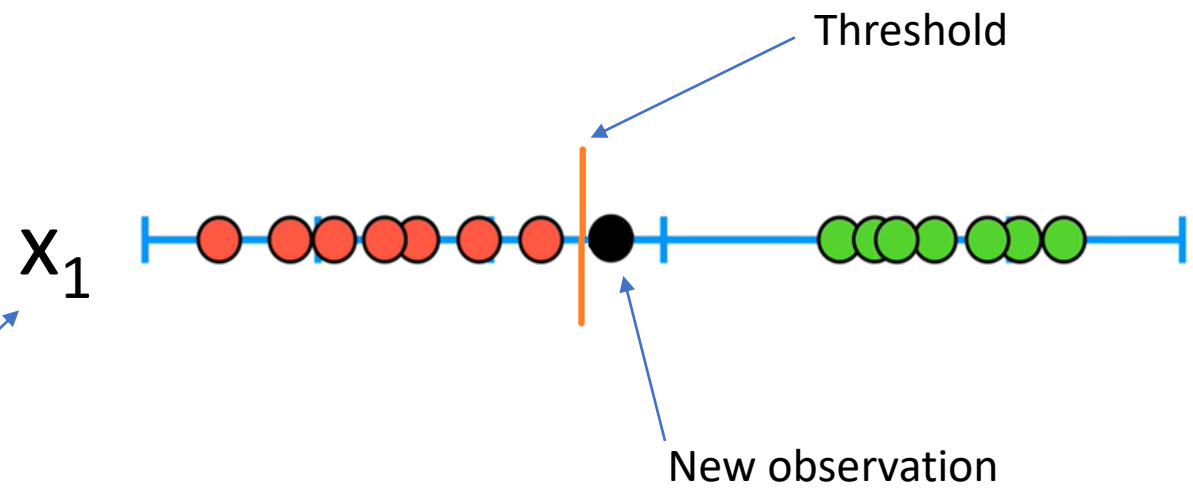
Threshold Choice



Predictor variable (e.g., “fasting glucose level”, “gas flow rate at some PCVD process step”, “income”, etc.)

- Category 1 = (e.g. “will develop diabetes”, “wafer will fail”, “customer will quit service”, etc.)
- Category 2 = (e.g. “won’t develop diabetes”, “wafer will pass”, “customer will keep subscription”, etc.)

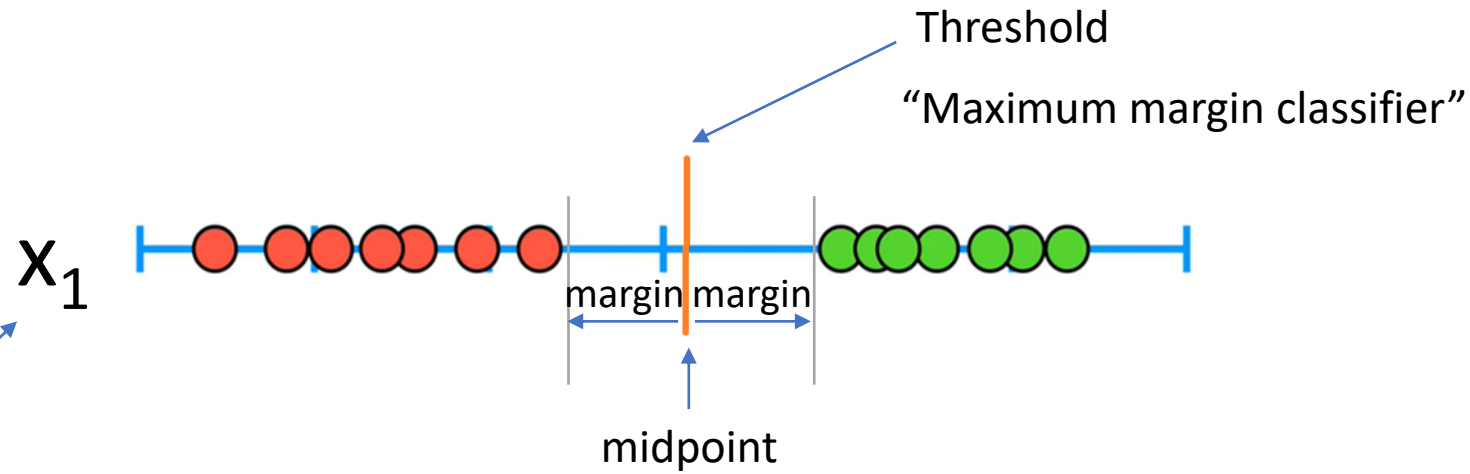
Threshold Choice



Predictor variable (e.g., “fasting glucose level”, “gas flow rate at some PCVD process step”, “income”, etc.)

- Category 1 = (e.g. “will develop diabetes”, “wafer will fail”, “customer will quit service”, etc.)
- Category 2 = (e.g. “won’t develop diabetes”, “wafer will pass”, “customer will keep subscription”, etc.)

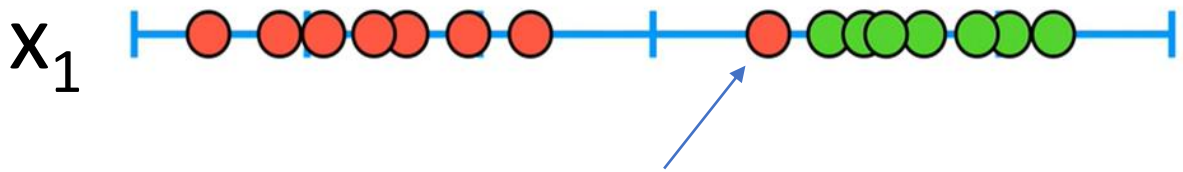
Maximum Margin Classifier



Predictor variable (e.g., “fasting glucose level”, “gas flow rate at some PCVD process step”, “income”, etc.)

- Category 1 = (e.g. “will develop diabetes”, “wafer will fail”, “customer will quit service”, etc.)
- Category 2 = (e.g. “won’t develop diabetes”, “wafer will pass”, “customer will keep subscription”, etc.)

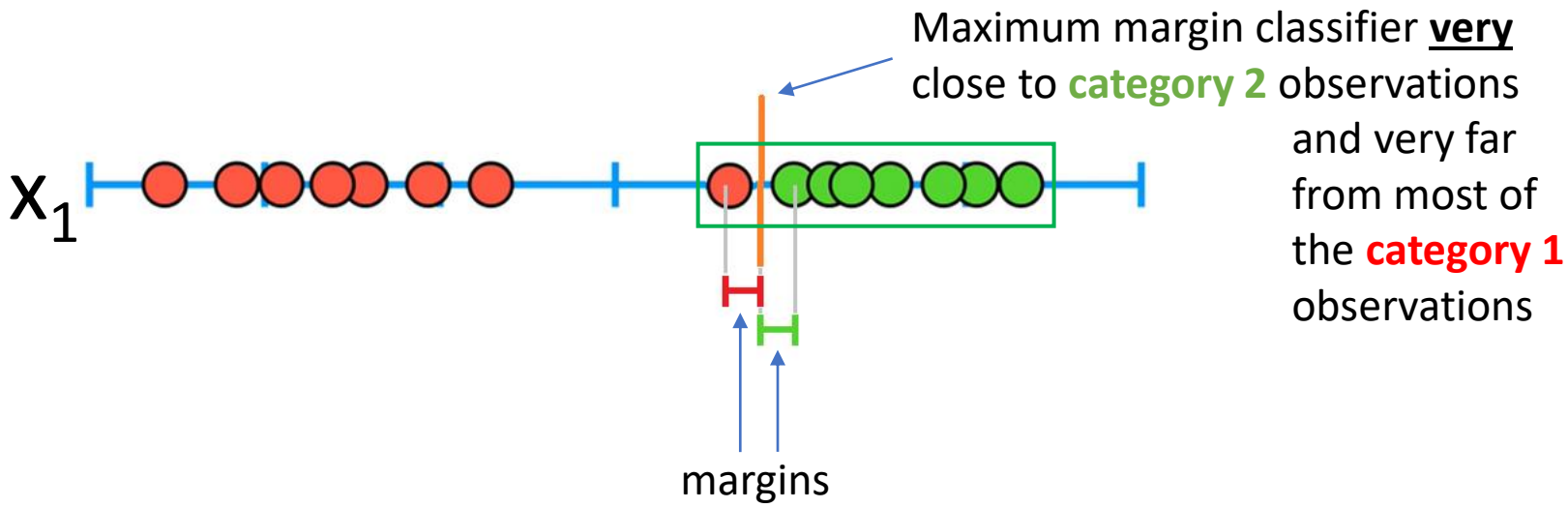
Outlier Effects



What about an outlier?

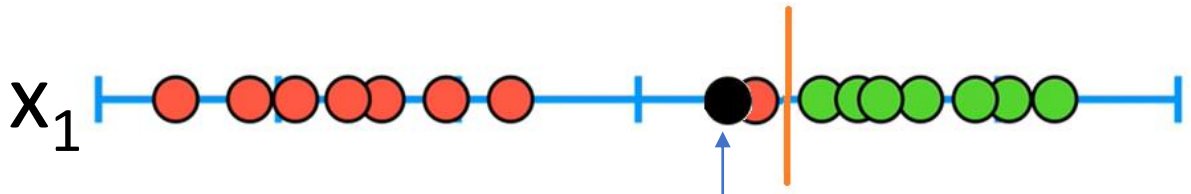
- Category 1 = (e.g. “will develop diabetes”, “wafer will fail”, “customer will quit service”, etc.)
- Category 2 = (e.g. “won’t develop diabetes”, “wafer will pass”, “customer will keep subscription”, etc.)

Outlier Effects - continued



- Category 1 = (e.g. “will develop diabetes”, “wafer will fail”, “customer will quit service”, etc.)
- Category 2 = (e.g. “won’t develop diabetes”, “wafer will pass”, “customer will keep subscription”, etc.)

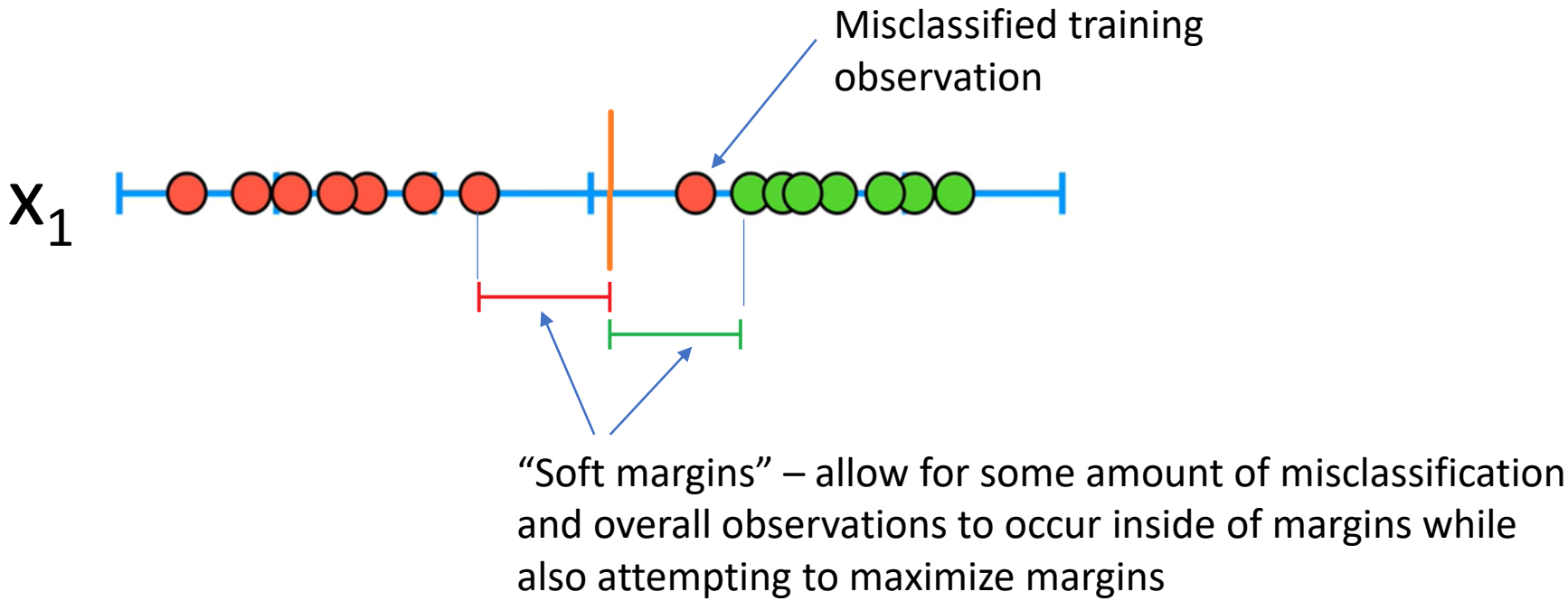
Failure Due to Outlier Effects



New observation at this point would be classified as category 1 even though MOST of the category 1 observations are much farther away than the category 2 observations

- Category 1 = (e.g. “will develop diabetes”, “wafer will fail”, “customer will quit service”, etc.)
- Category 2 = (e.g. “won’t develop diabetes”, “wafer will pass”, “customer will keep subscription”, etc.)

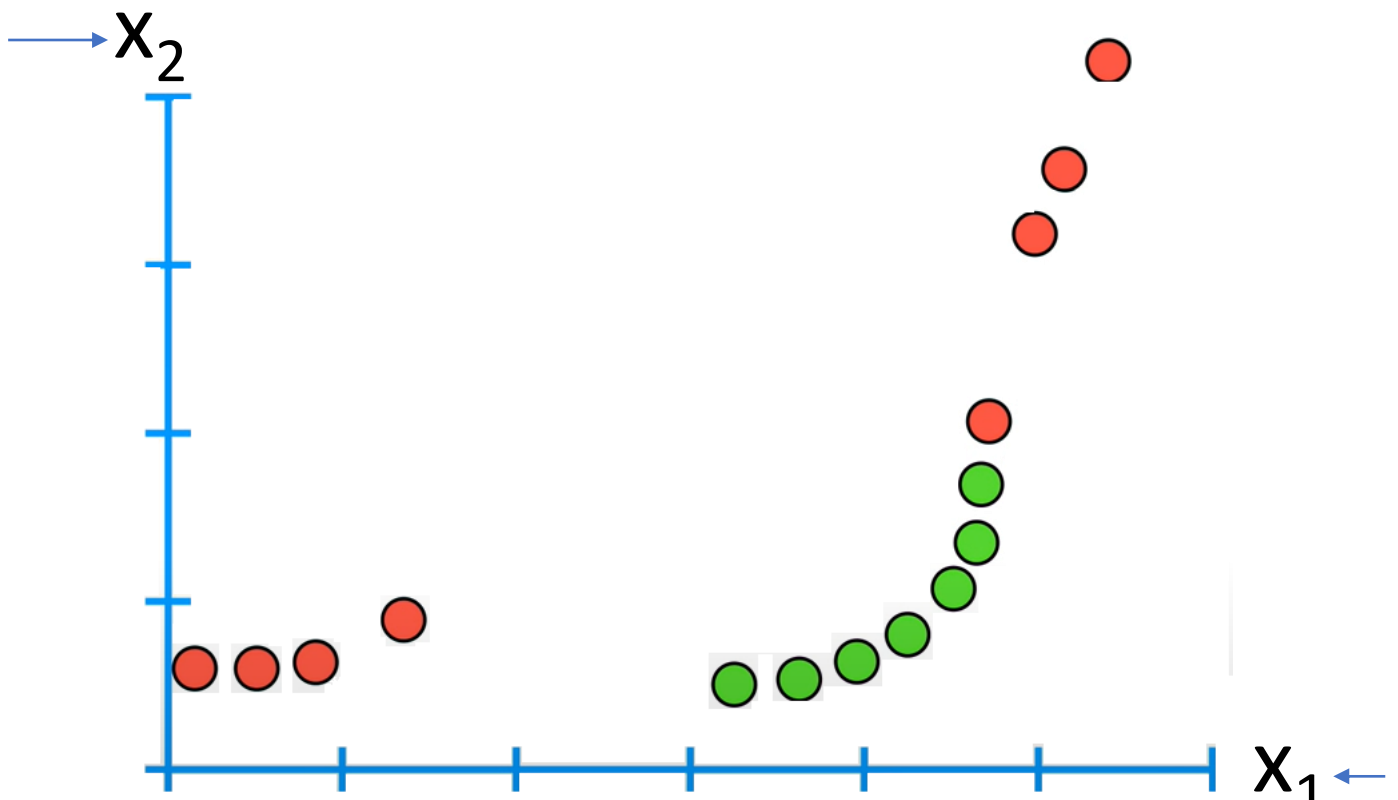
“Soft Margin Classifier”
AKA
“Support Vector Classifier”



- Category 1 = (e.g. “will develop diabetes”, “wafer will fail”, “customer will quit service”, etc.)
- Category 2 = (e.g. “won’t develop diabetes”, “wafer will pass”, “customer will keep subscription”, etc.)

Moving into Two Dimensions

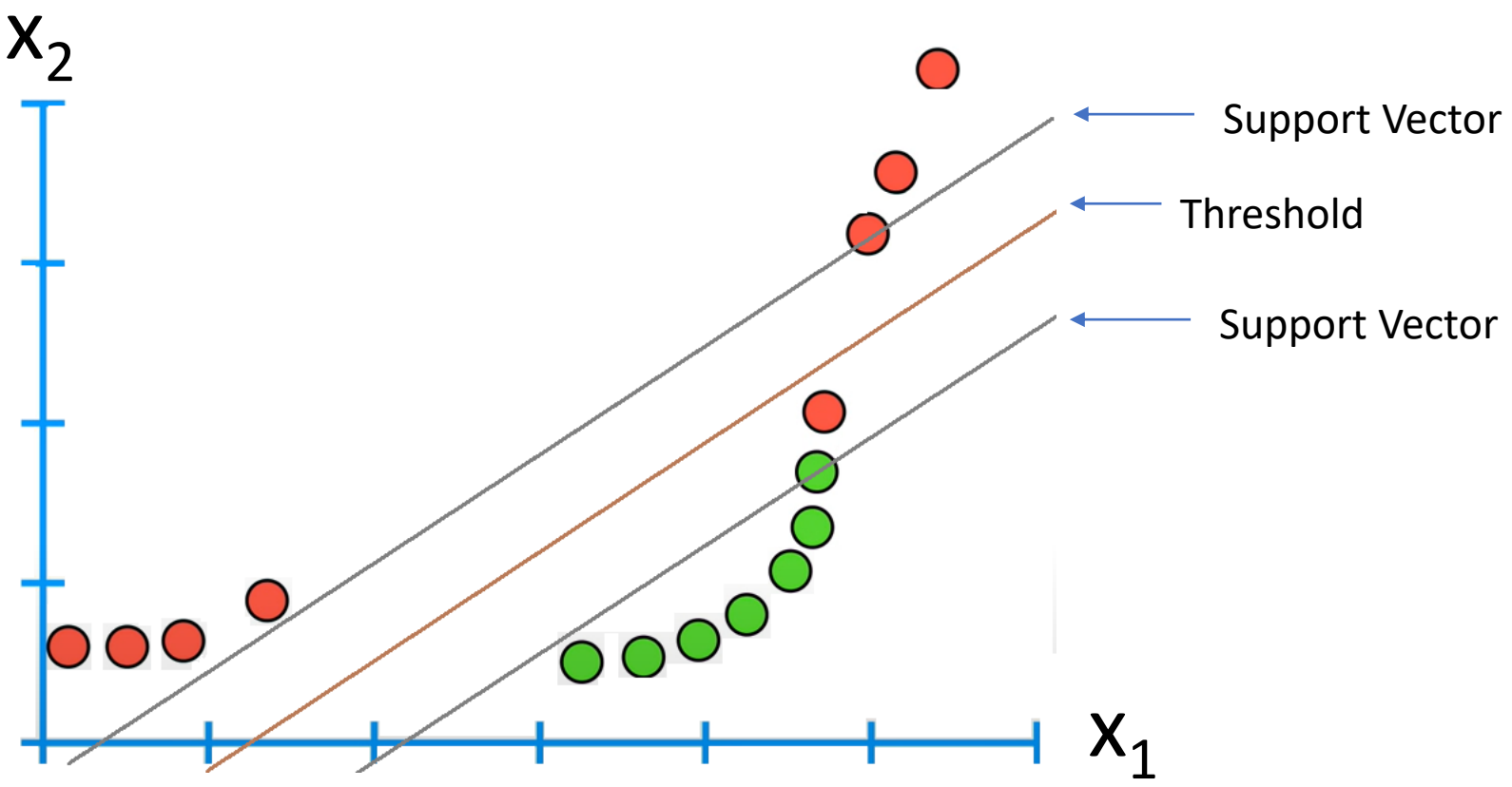
Predictor variable (e.g., "blood pressure", "RF frequency", "location", etc.)



Predictor variable (e.g., "fasting glucose level", "gas flow rate at some PCVD process step", "income", etc.)

- Category 1 = (e.g. "will develop diabetes", "wafer will fail", "customer will quit service", etc.)
- Category 2 = (e.g. "won't develop diabetes", "wafer will pass", "customer will keep subscription", etc.)

SVM Classifier in 2D is a Line

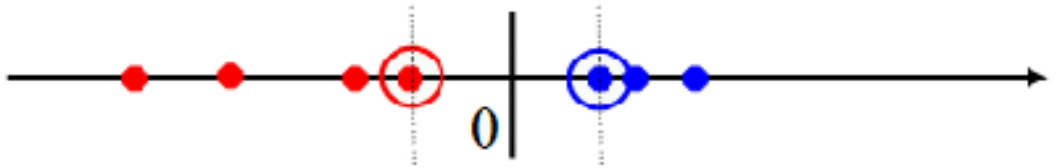


- Category 1 = (e.g. “will develop diabetes”, “wafer will fail”, “customer will quit service”, etc.)
- Category 2 = (e.g. “won’t develop diabetes”, “wafer will pass”, “customer will keep subscription”, etc.)

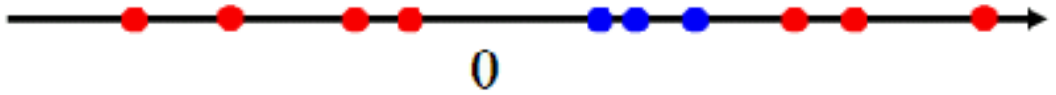
Dimensionality Increase
*(beyond number of features)

The Conundrum

Choice of threshold here is obvious



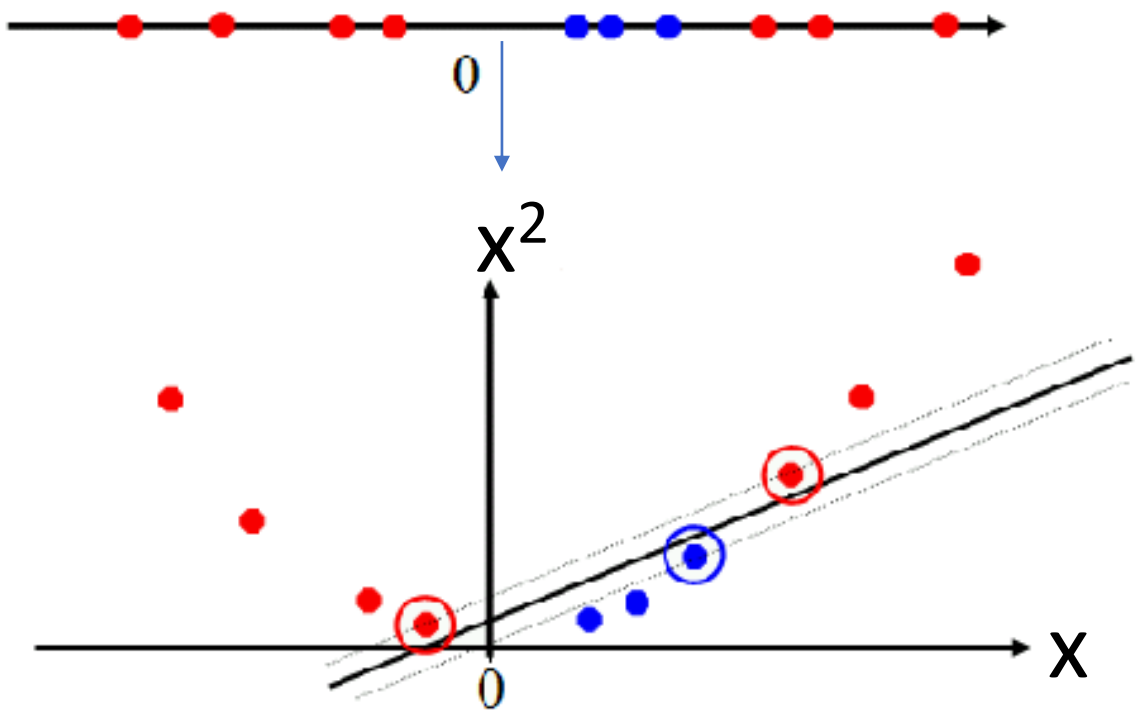
What about here?



Category 1

Category 2

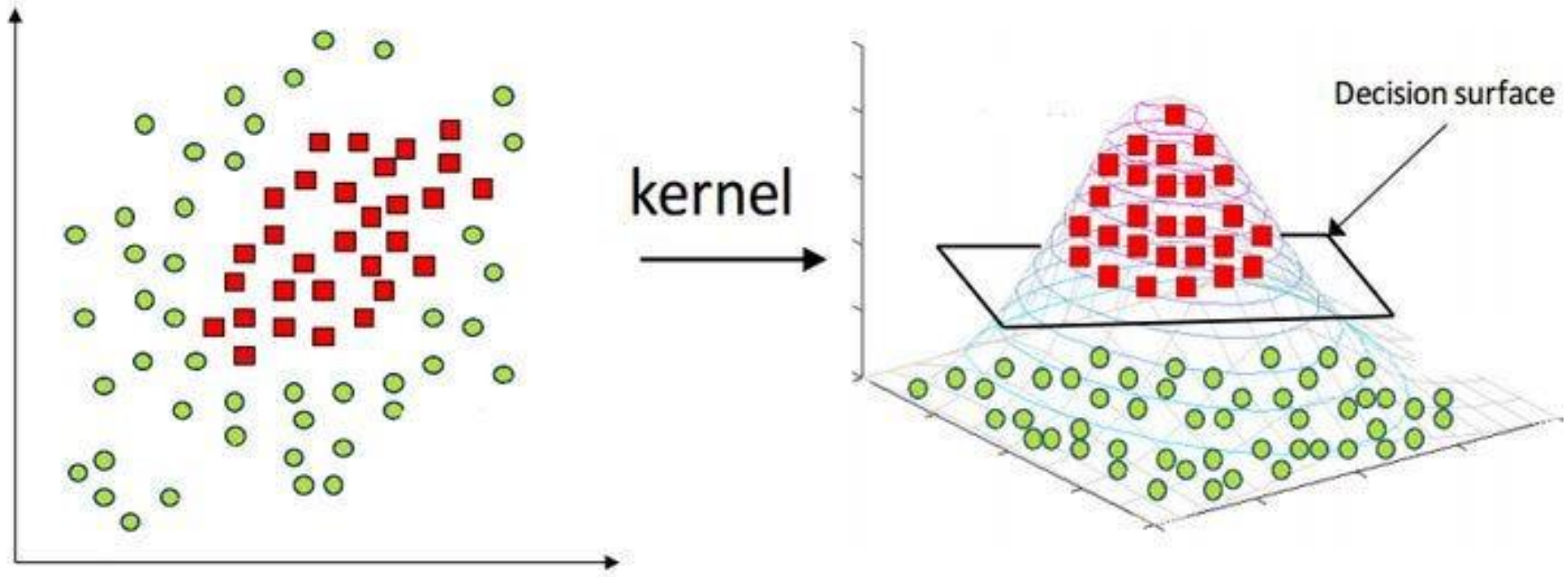
The Solution – Engineer a New Feature



Category 1

Category 2

2D to 3D



Category 1

Category 2

Kernel Trick

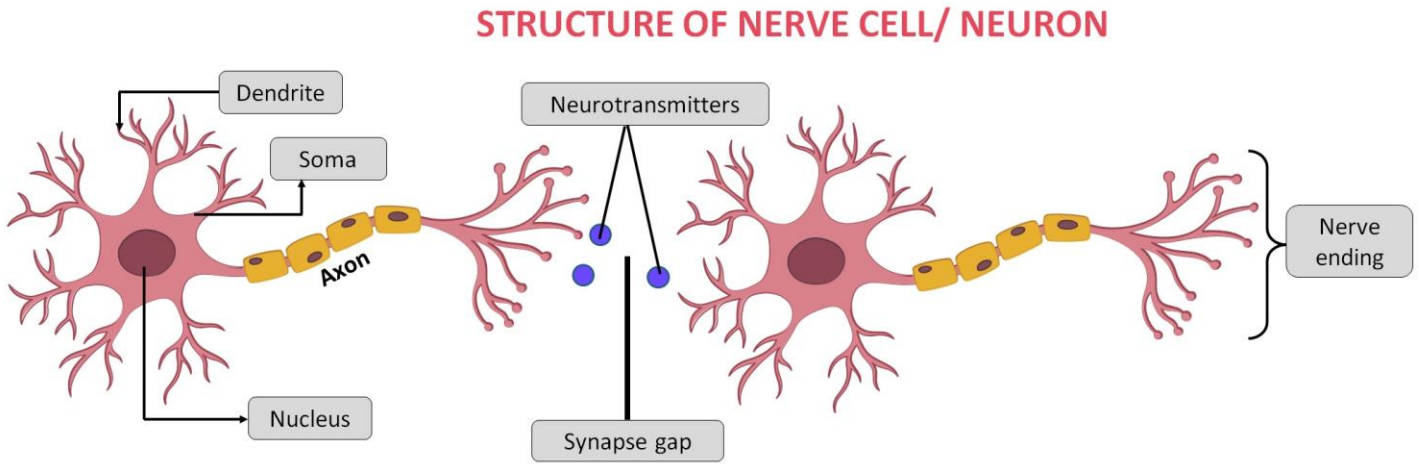
“Kernel Trick”

- Part of the SVM process involves comparing inner (dot) products of all combinations of the observations
- Raising an observation point to higher dimensional space may require numerous additional terms to be calculated (think about the additional features created in polynomial regression)
- The “kernel trick” is the ability to apply the transformation to the dot product of the original observations and achieve the same result as if you applied the transformation to all of the original data and then calculated the dot products
- This can substantially reduce overhead including memory requirements

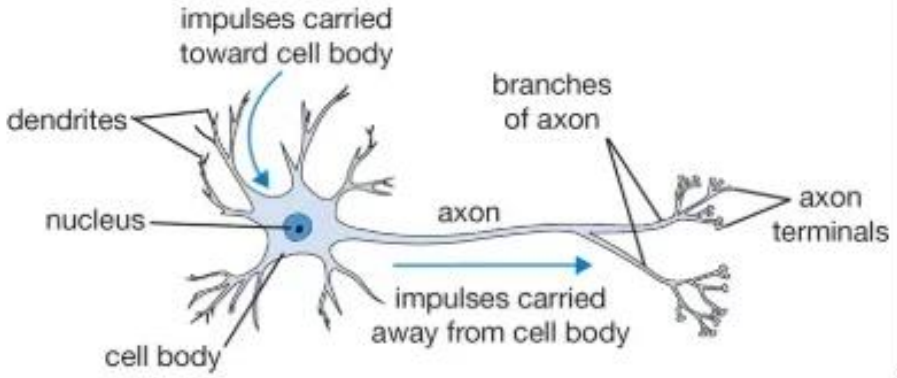
Don’t get hung up on this. There is a substantial amount of underlying math that we do not have time to go into and is beyond the scope of this class. You should understand that with regard to SVM the “kernel” is the function that allows calculation of proximity of observations in higher dimensional space without transforming the observations themselves into that higher dimensional space.

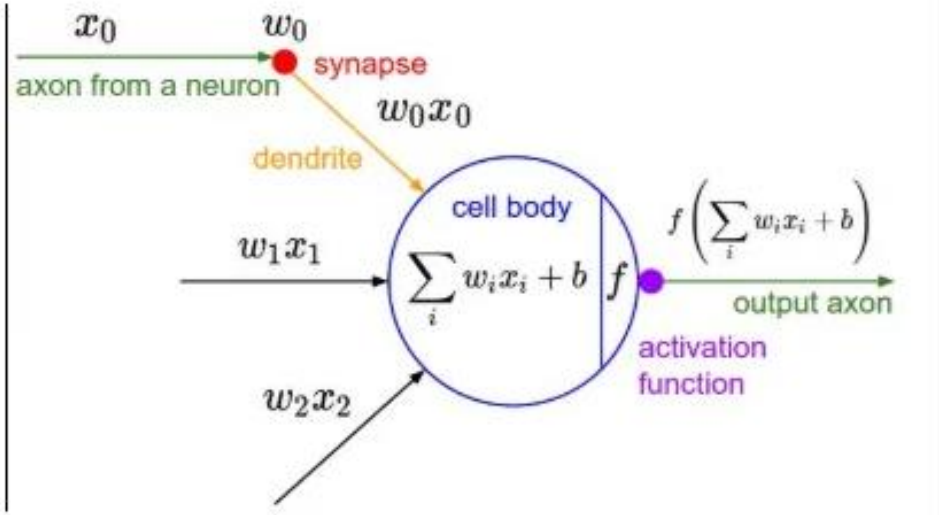
Multilayer Perceptron

Biological Neurons

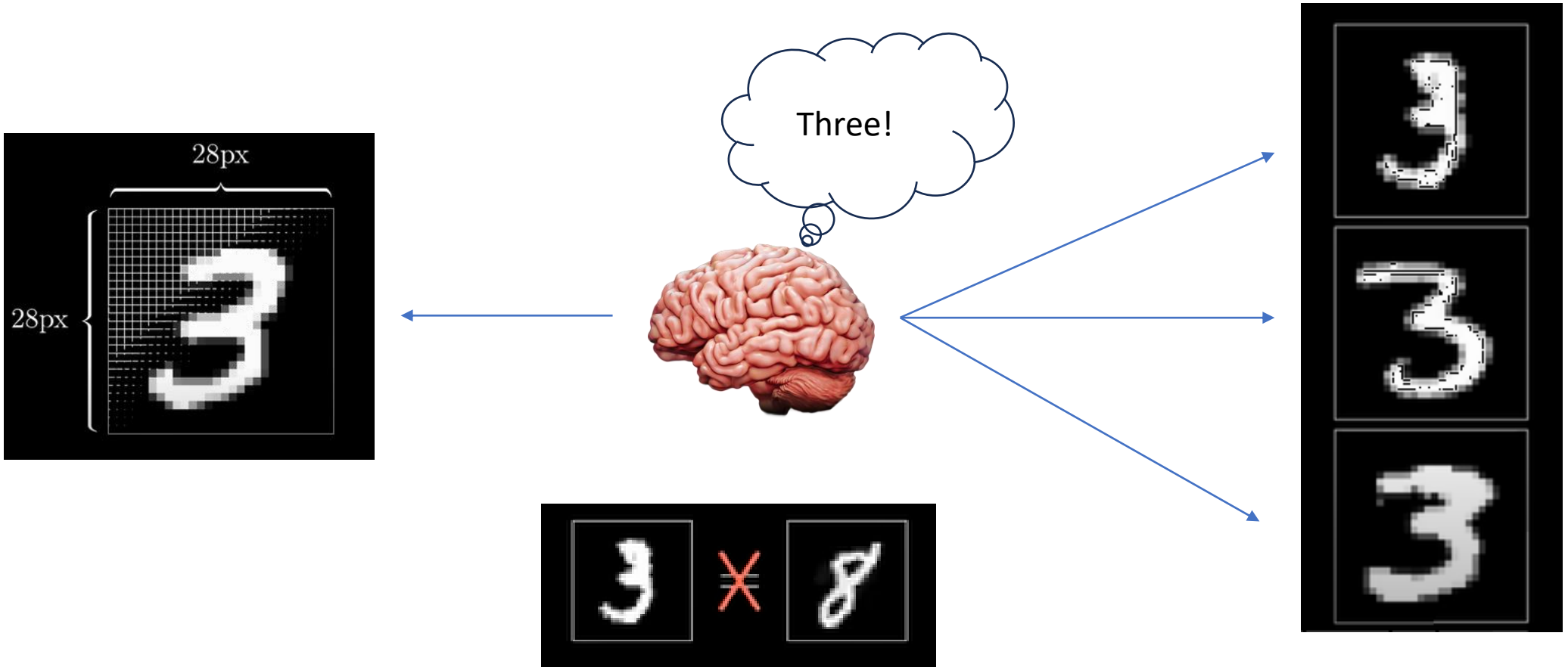


Direction of Signals



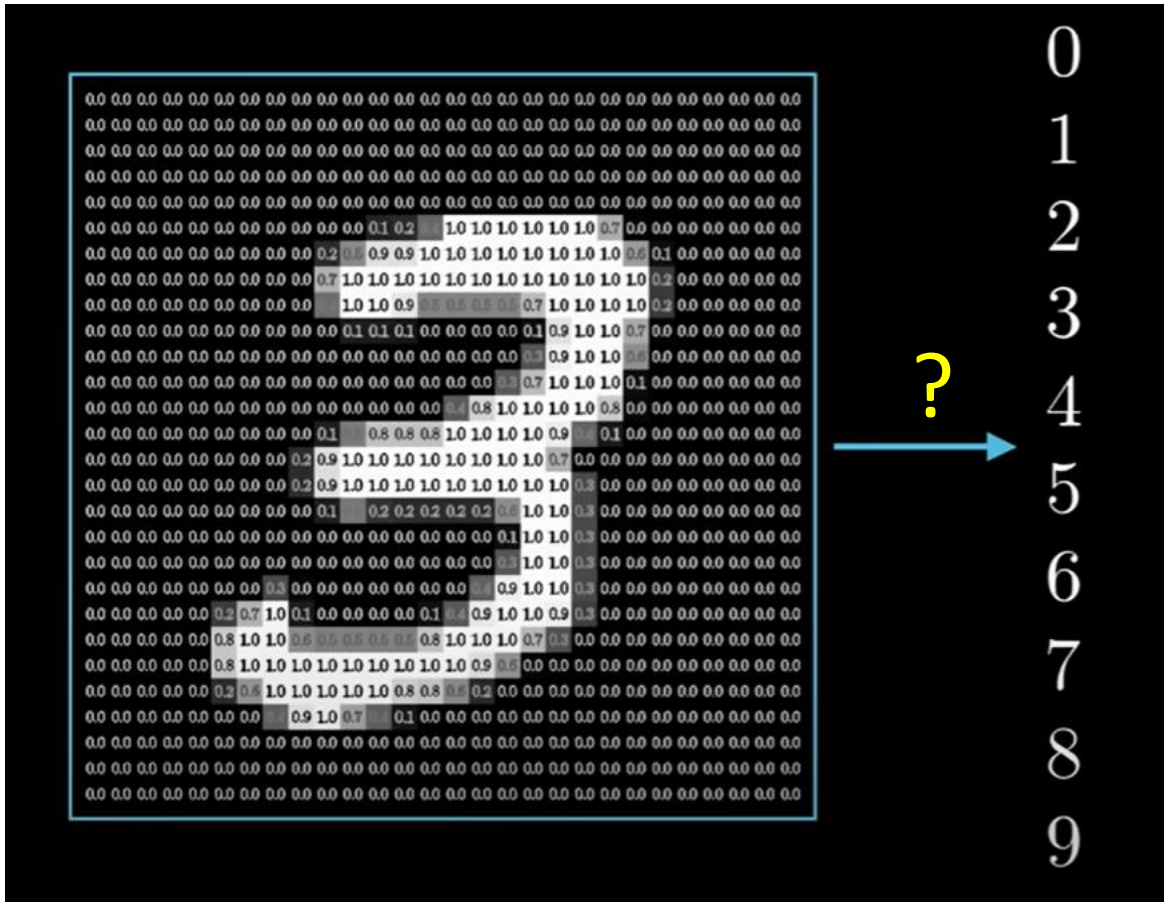


Consider How Easily Your Brain Recognizes Characters

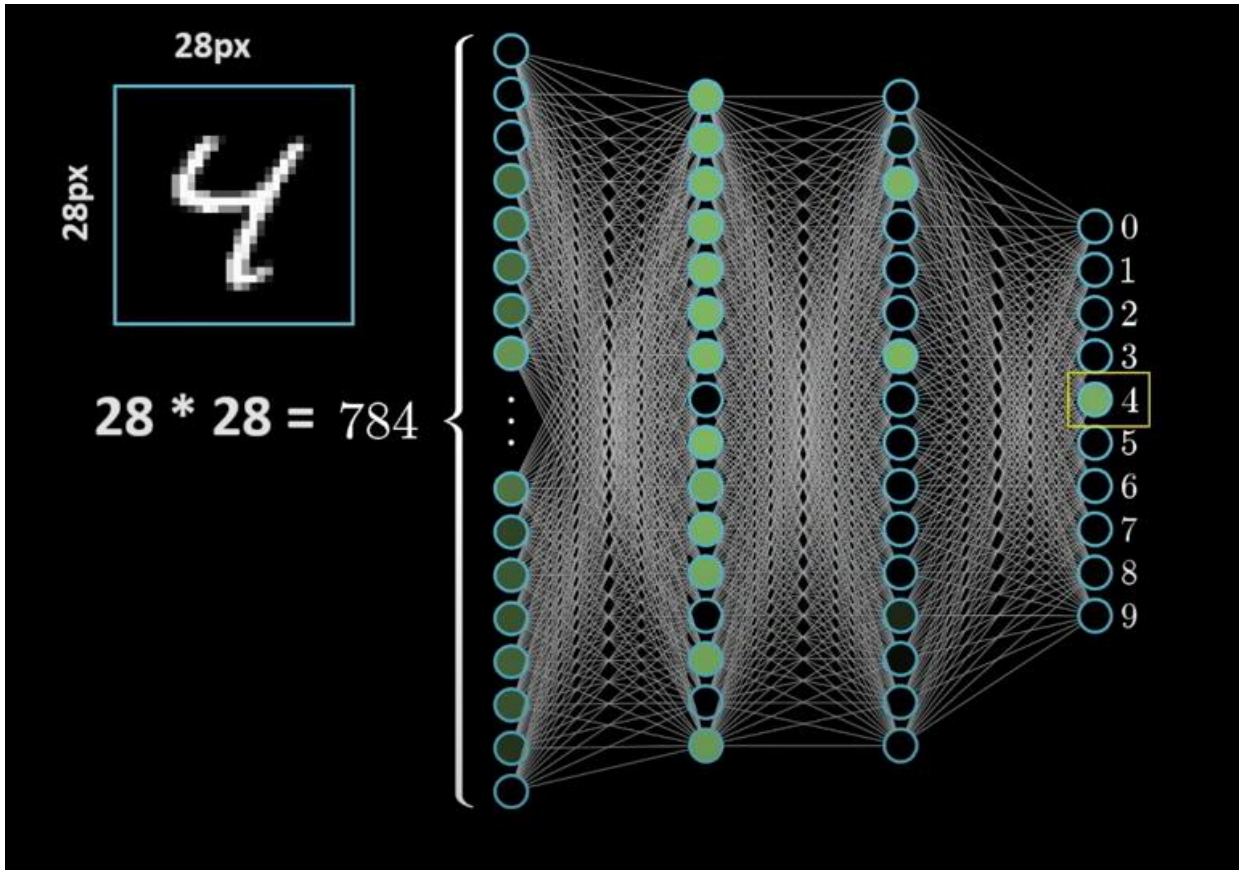


For that matter, consider how (relatively) easy you can determine what this is: 4Ƨ5

How Would You Write A Program To Do This?

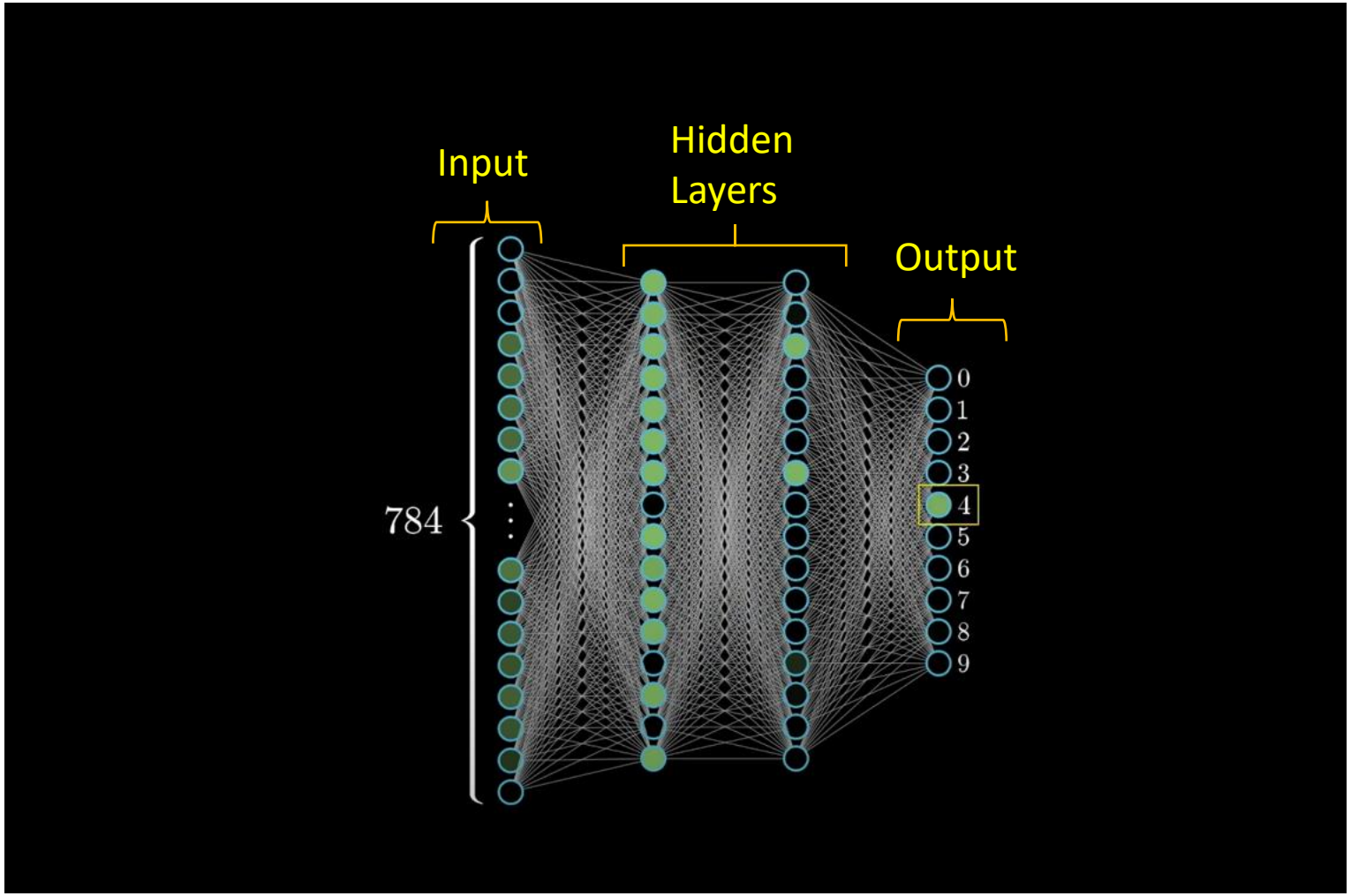


Basic Neural Network – “Multilayer Perceptron”

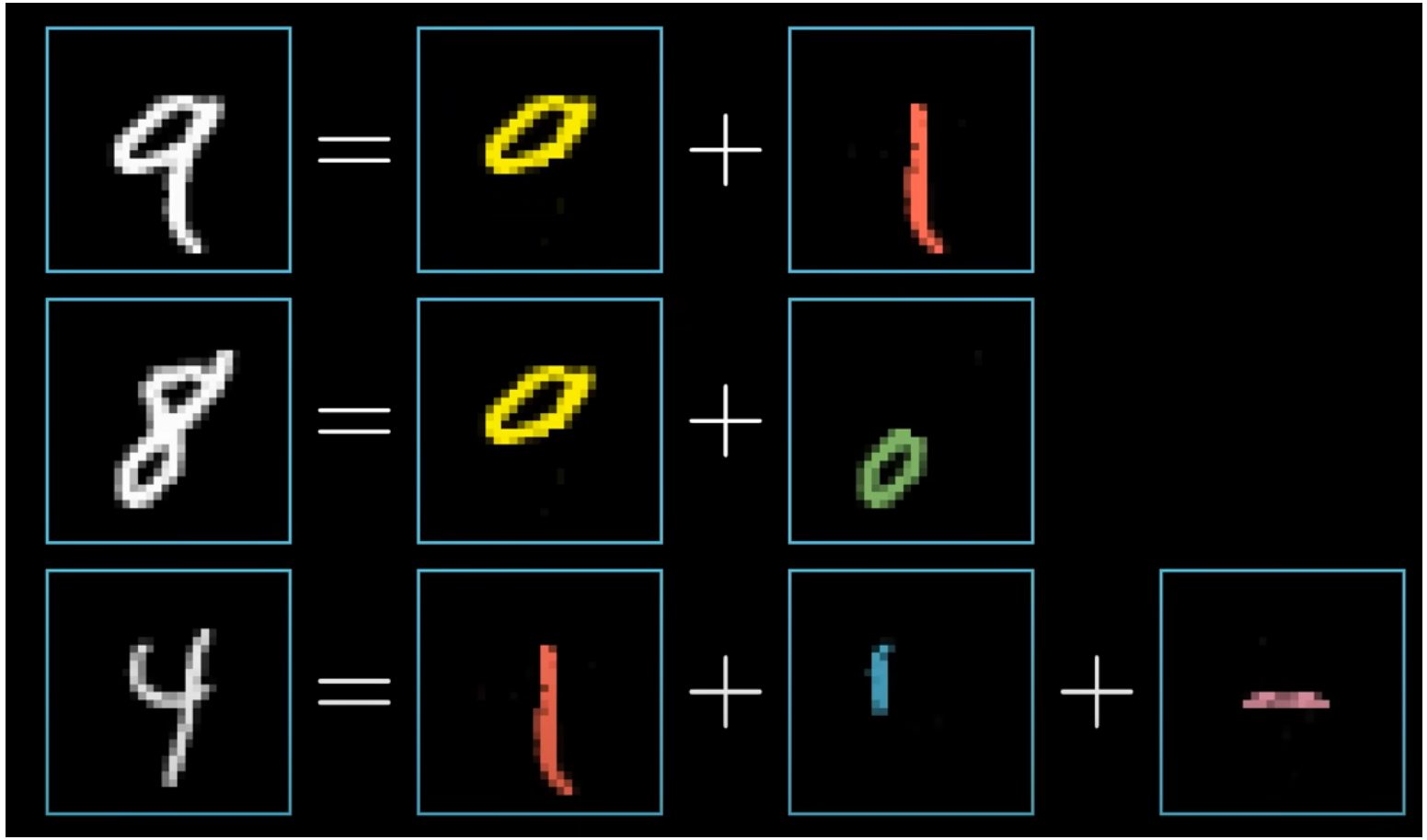


Nodes (circles) represent neurons and the lines represent connections to other neurons in other layers.

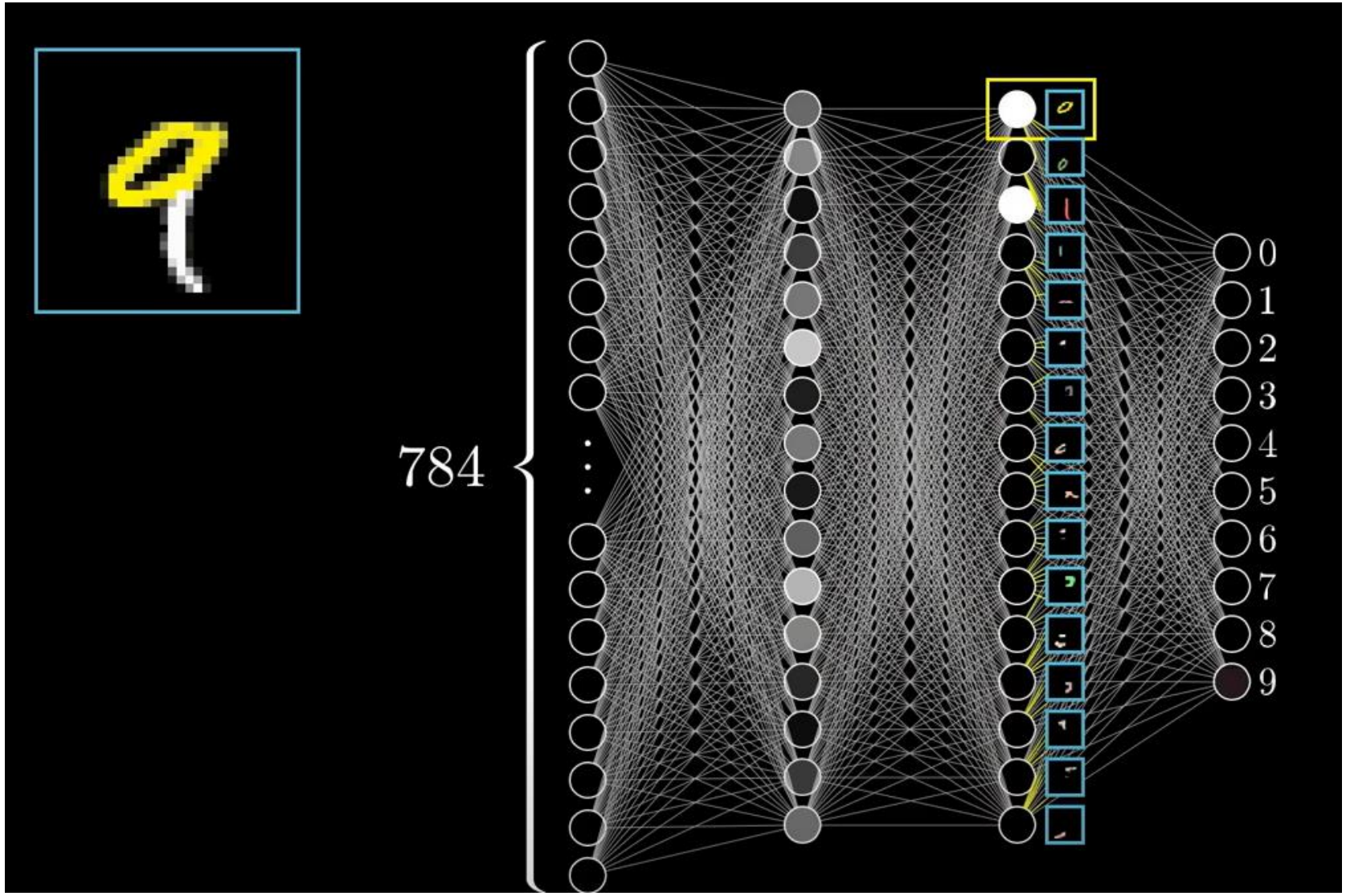
Component Layers of Network



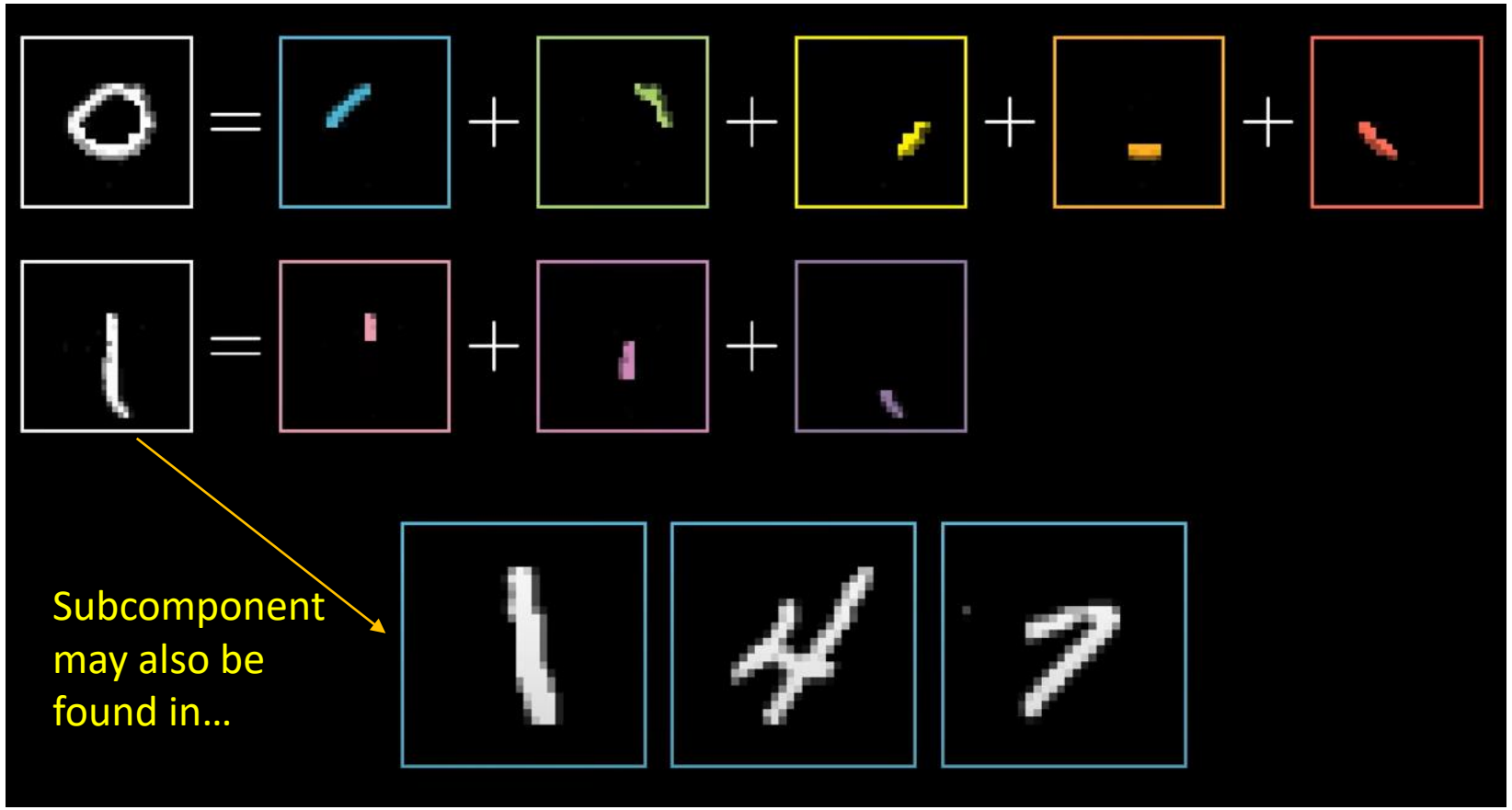
Why Layers?



Second Hidden Layer Represents These Shapes
(maybe… this is one way the network could function)

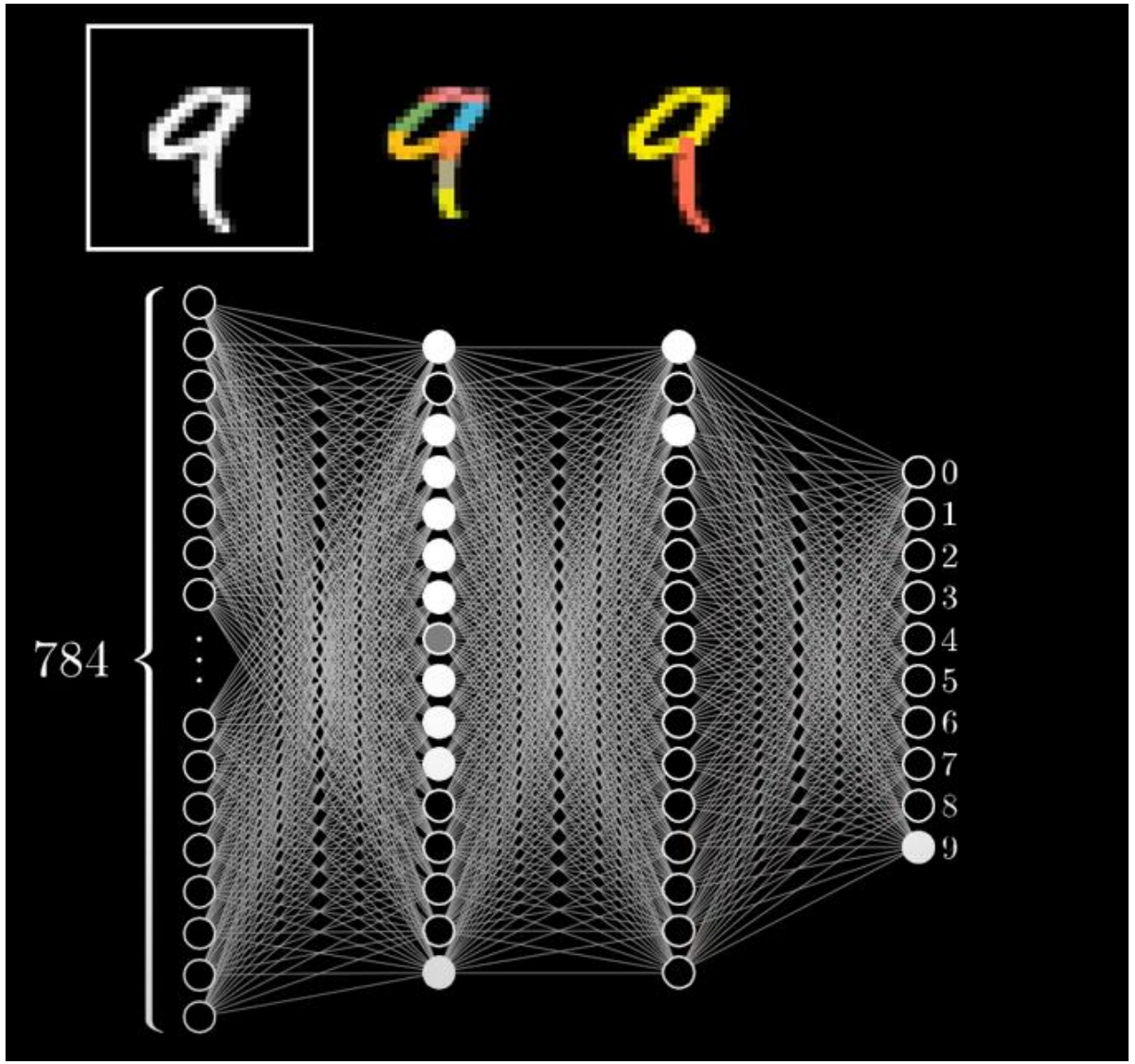


What About First Hidden Layer?
(sub-components of the basic shapes…maybe)

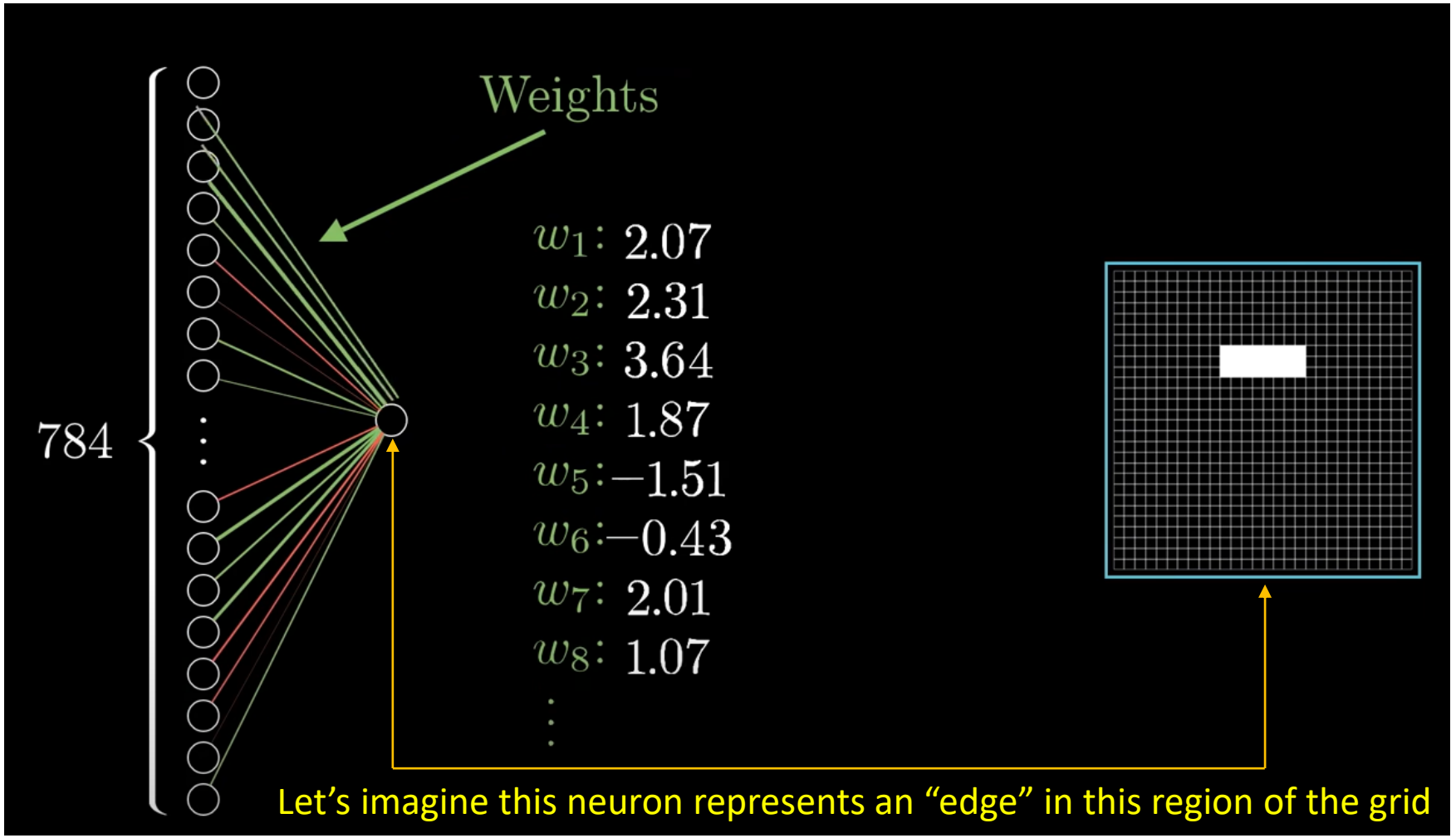


1 4 7

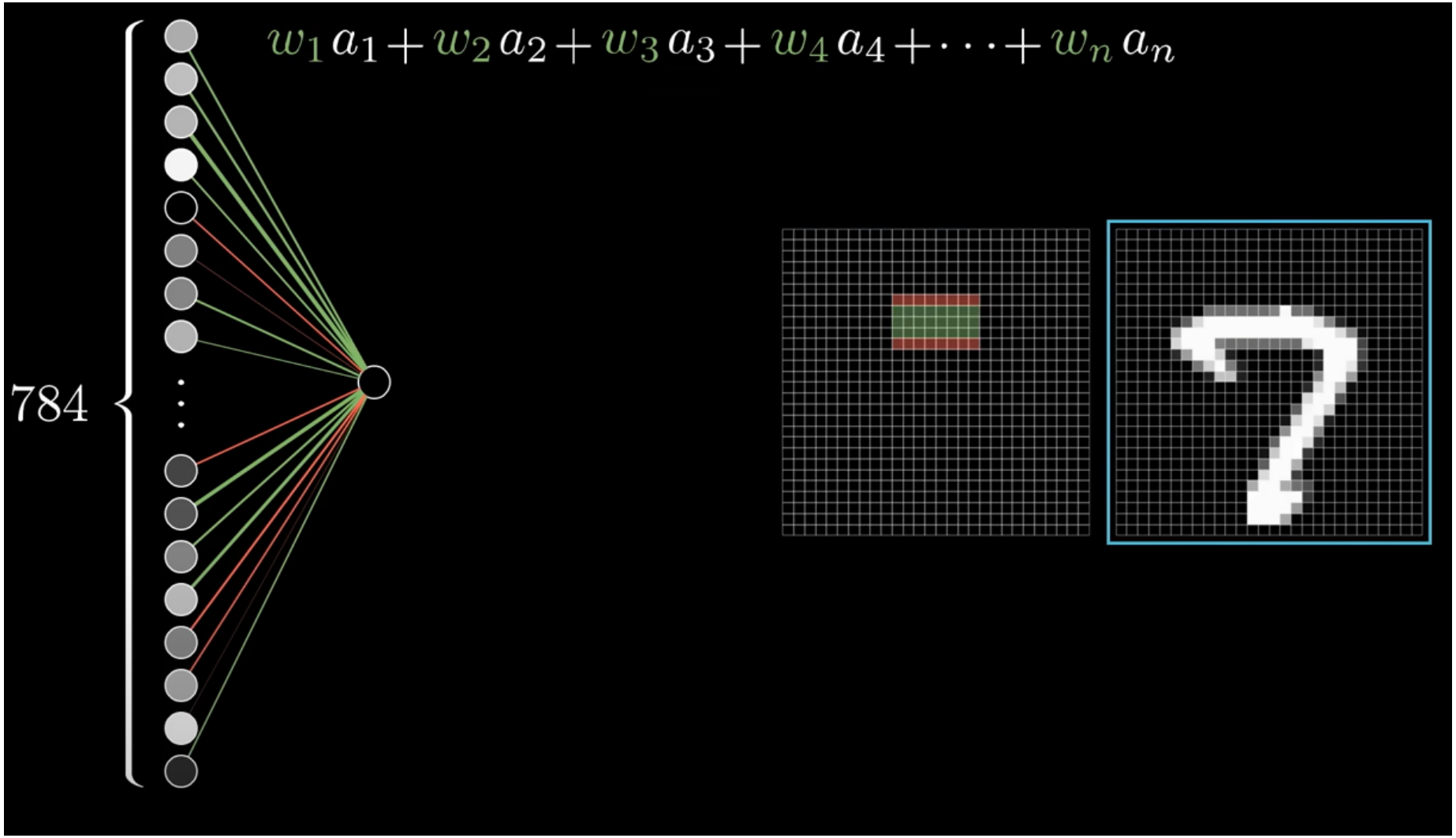
Example of Layer Neurons Activating



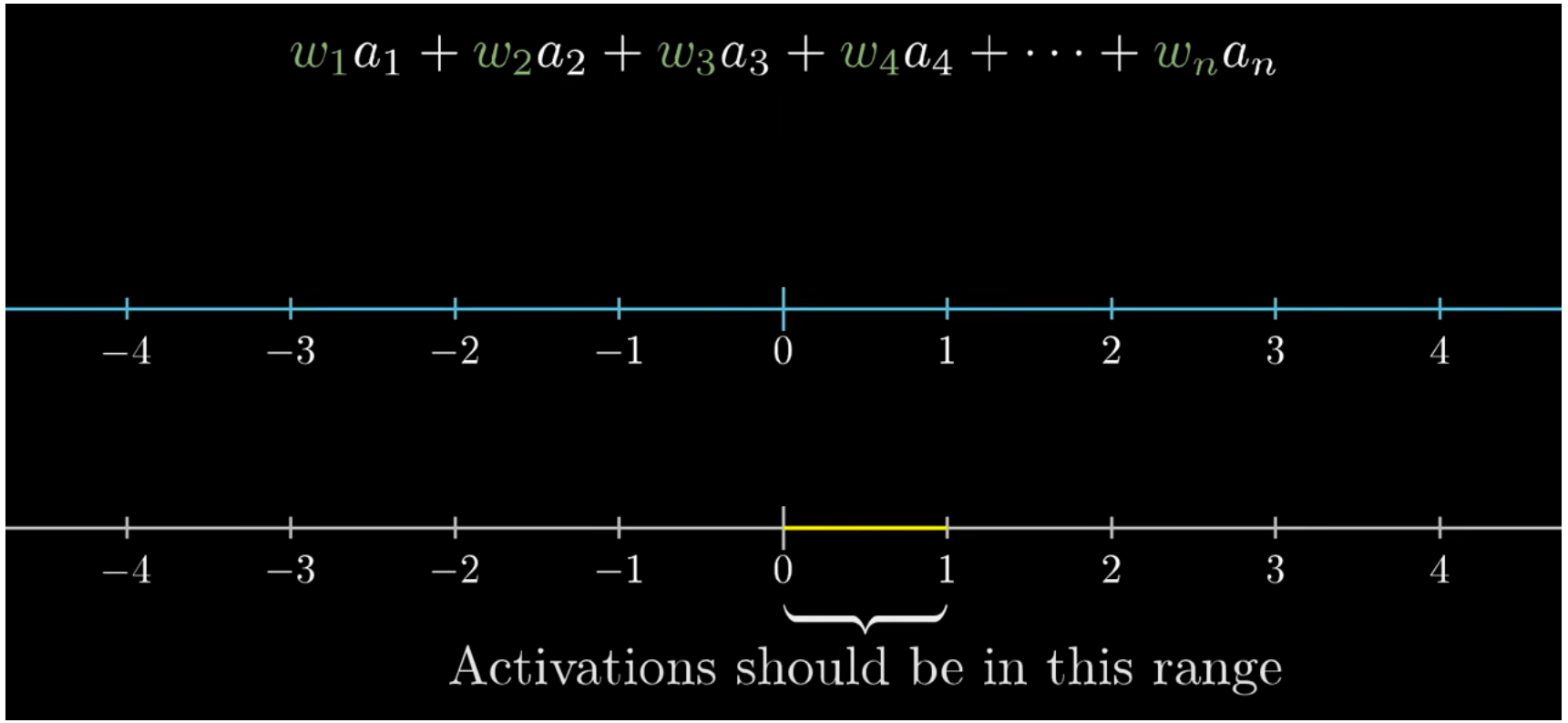
How Does This Work?



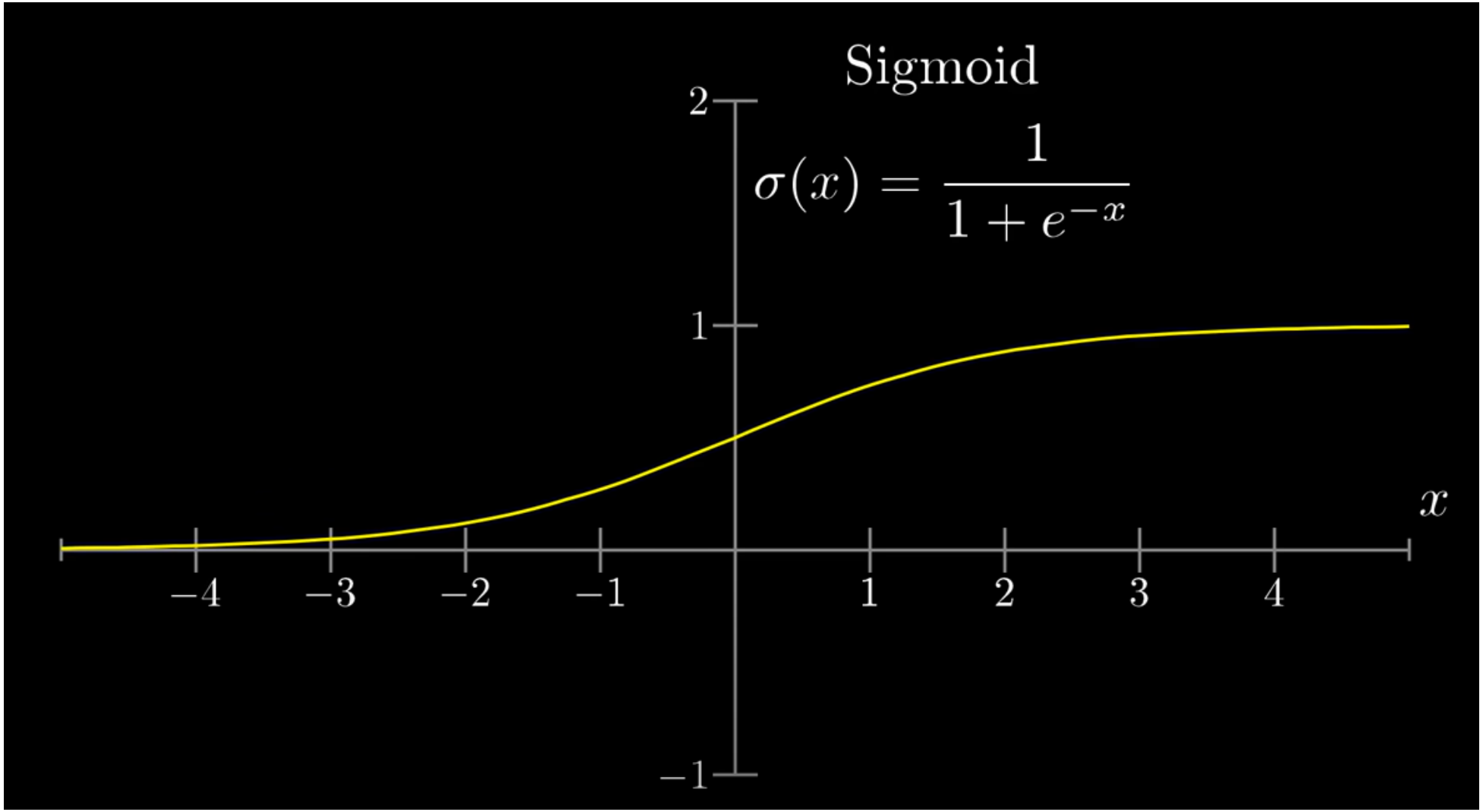
Inhibitory vs Excitatory Signals



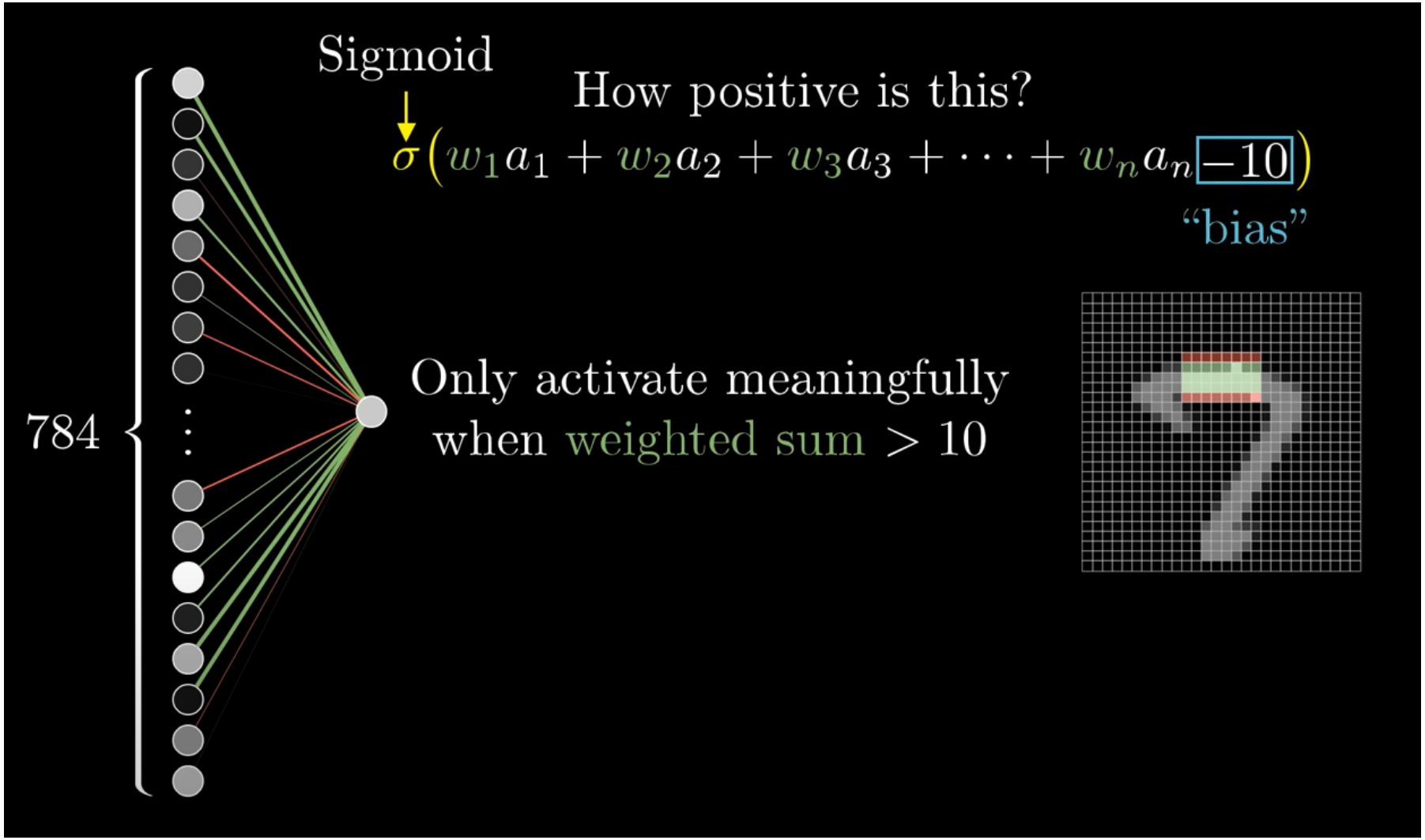
Activation Weights Can Add to Arbitrarily Large or Small Values



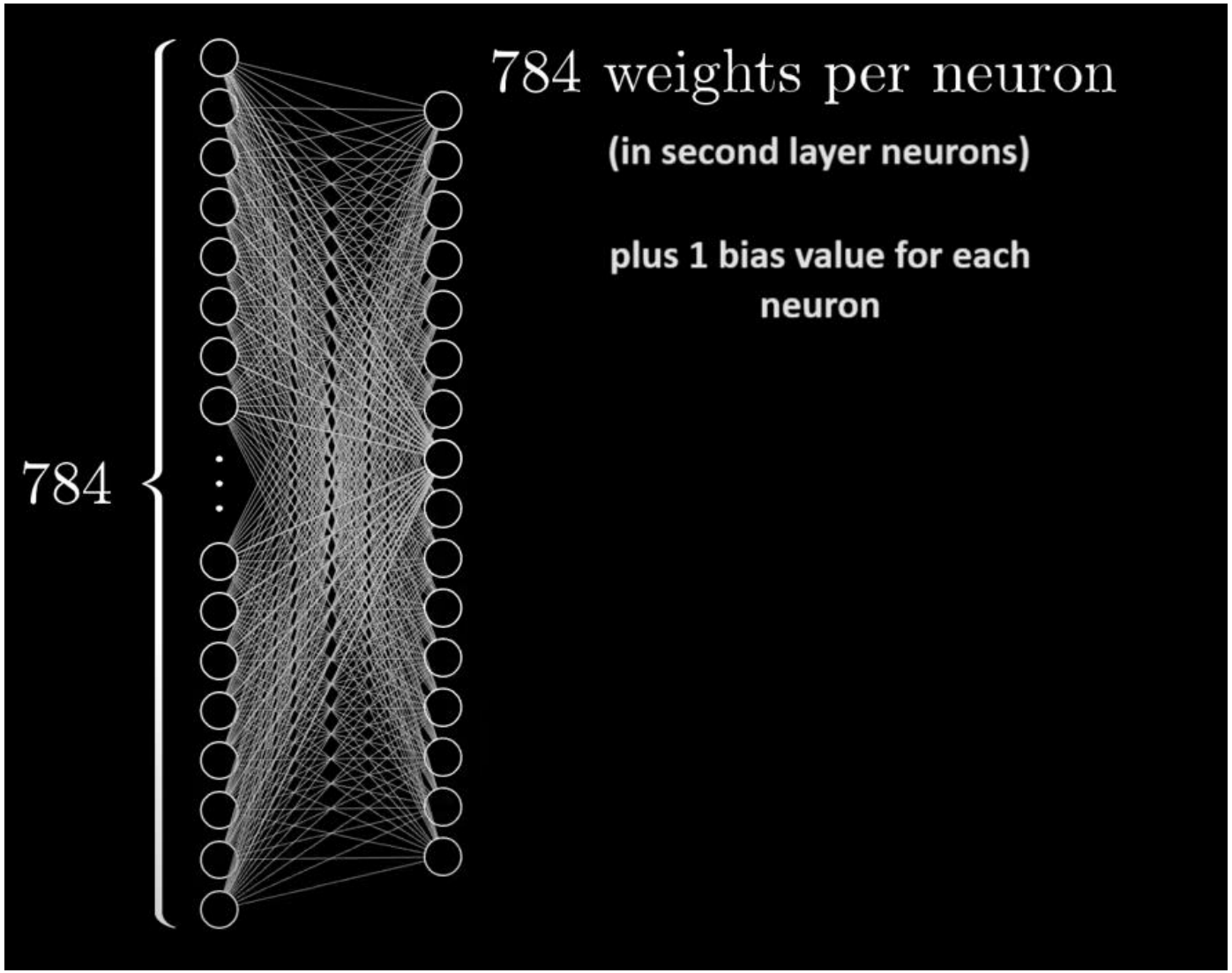
Sigmoid Function (should be familiar)



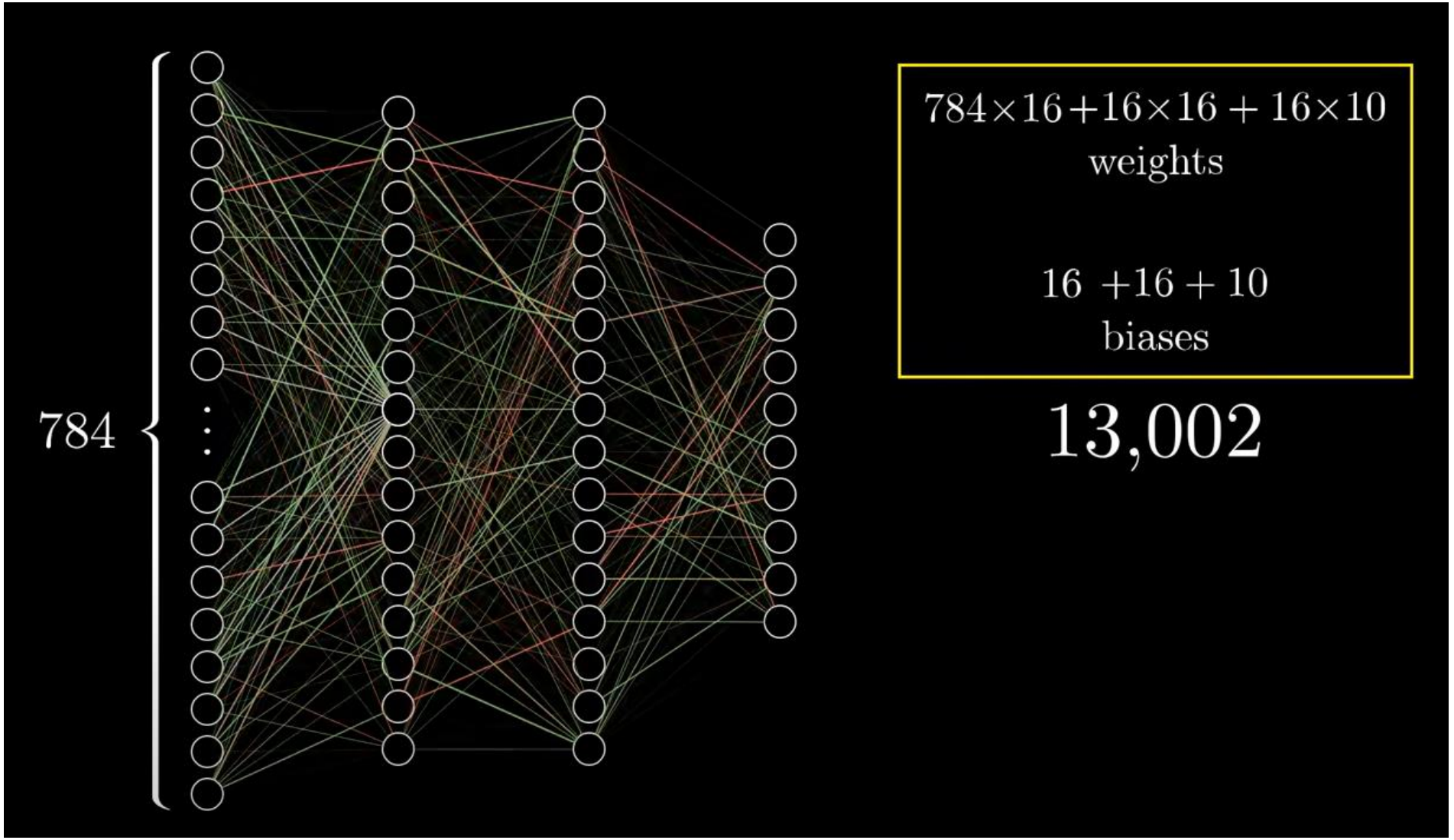
Introducing a Bias



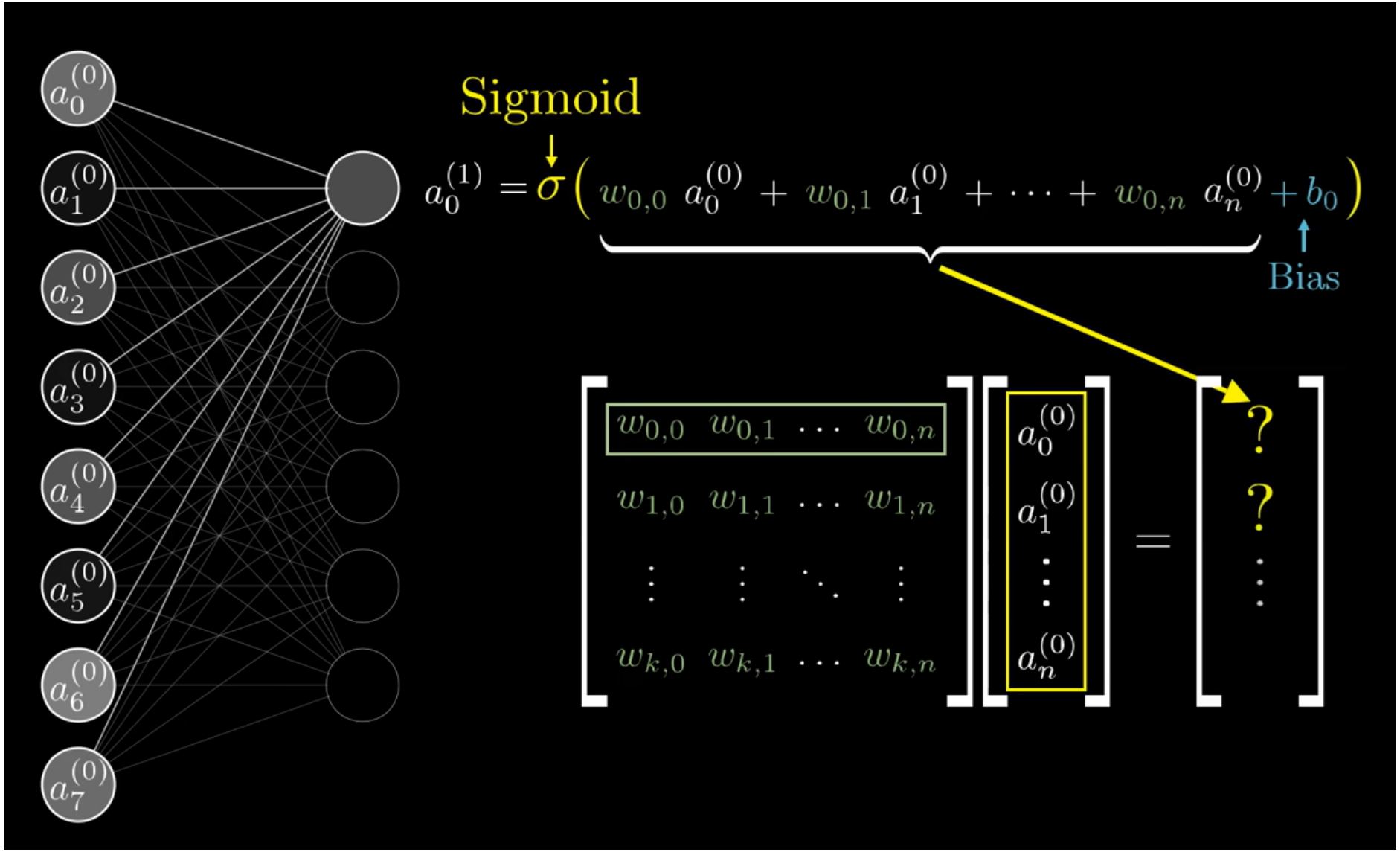
Increasing Complexity Per Layer



Total Complexity



The Importance of Linear Algebra



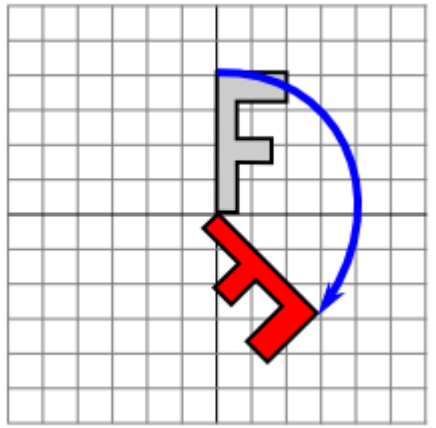
Relationship to Computer Graphics and GPUs

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

Rotation Matrix

(Homogeneous Coordinates Representation)

Rotation →



Reflection example →