TEXAS
The University of Texas at Austin

TOPIC 4 NONLINEAR PROGRAMMING

Outline

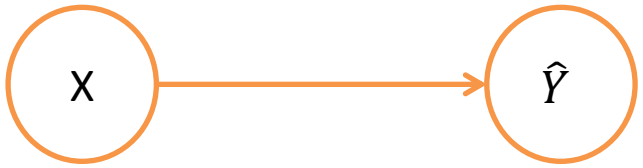
- Neural Networks
  - Multi-Level Perceptrons
  - Convolutional Neural Networks

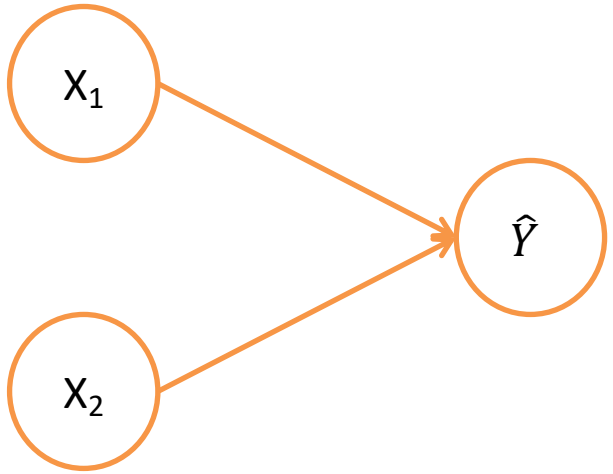
- In 2015 Google released an open-source tool for training neural nets called TensorFlow for python
- TensorFlow does all the backend work
- Keras is a modeling package that makes it easy to formulate neural nets and then passes them to TensorFlow for training

External Resources

- I don’t know a great (free) text resource that explains this content very well
- I did find some pretty good youtube videos though
- https://www.3blue1brown.com/videos
  - Scroll down to the ‘Neural networks’ section
  - There is a 4-part video series that explains multi-level perceptrons well
- https://www.youtube.com/watch?v=FmpDIaiMIeA
  - This does an OK job at explaining convolutional neural networks

Neural Networks



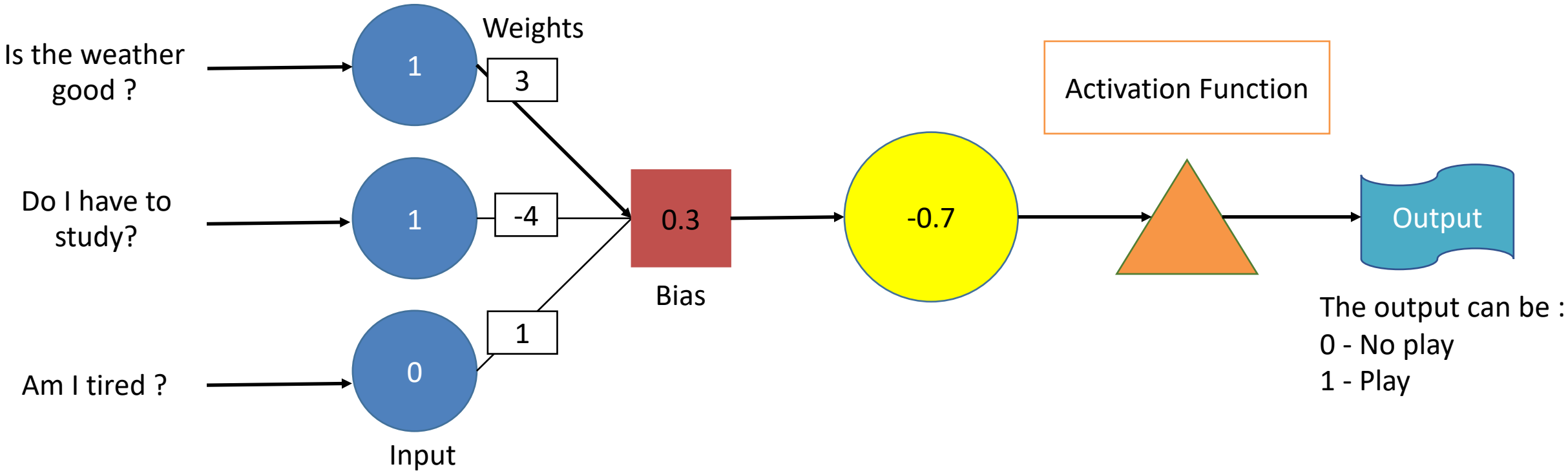


- Weights and biases of a neural network are like the betas of a regression

Activation function

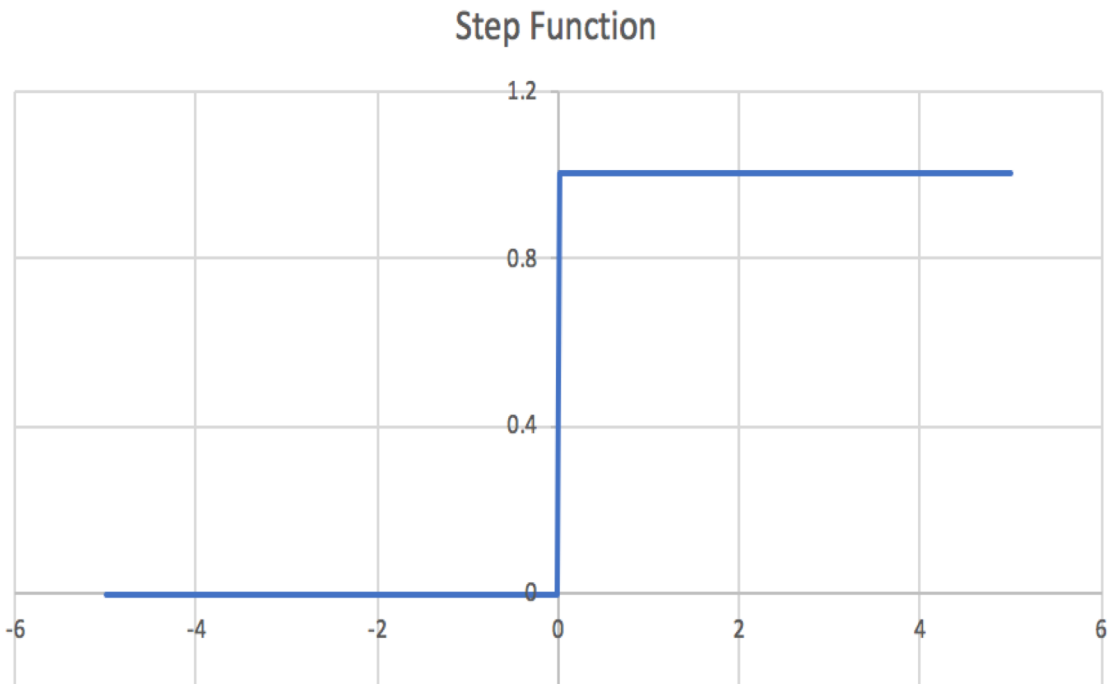
The activation function determines whether a neuron should be activated or not by calculating a weighted sum and then adding bias to it.

If the total signal is positive, the output is 1. If the total signal is negative, the output is 0.



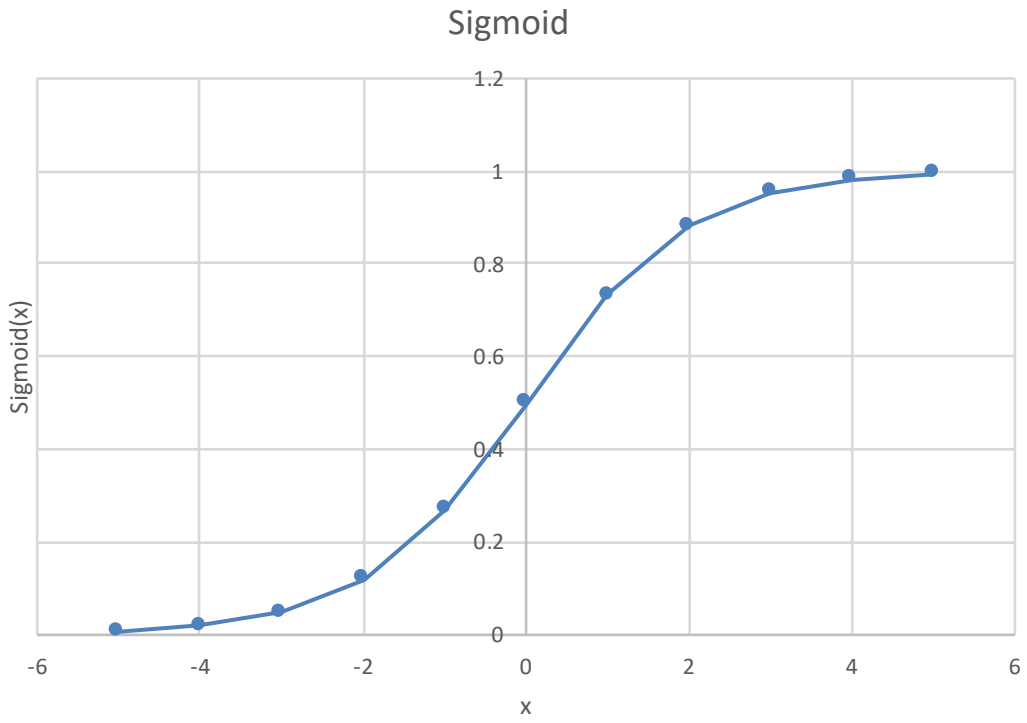
Activation function

Step function activation



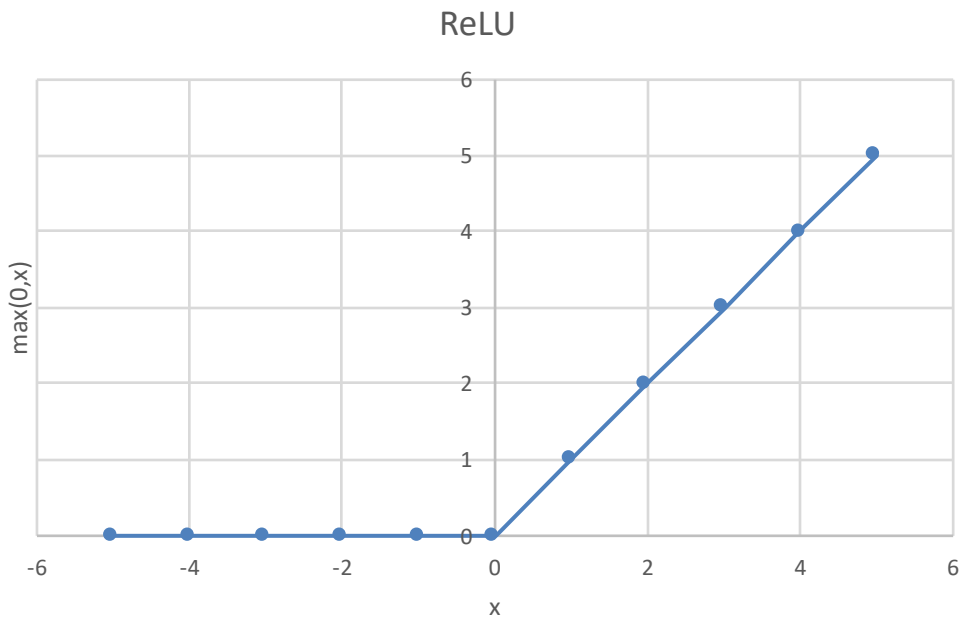
- Step function activation
- Everything left of zero is zero
- Everything right of zero is 1

Sigmoid Activation Function



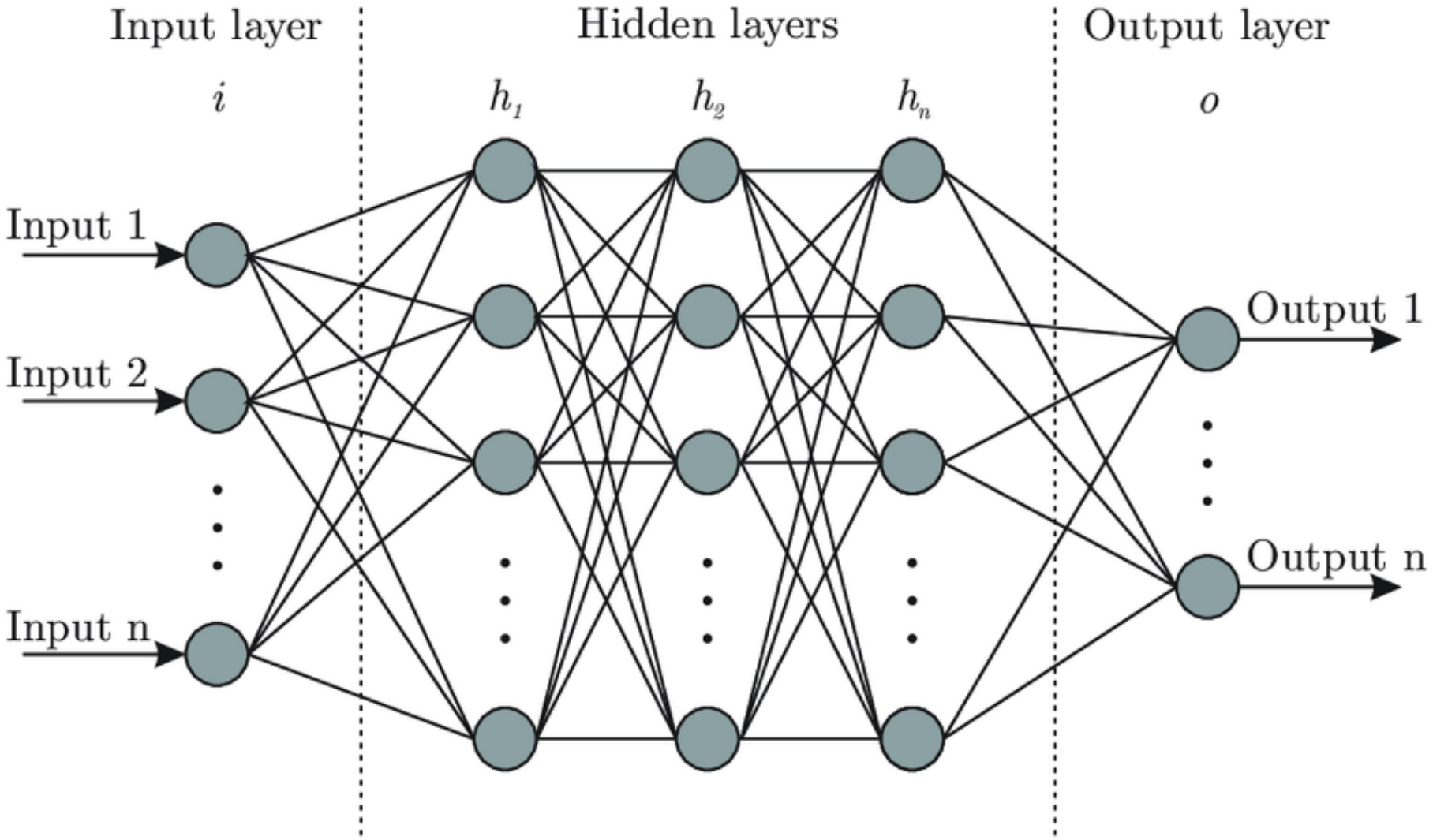
- Also known as the logistic function
  - $\frac{1}{1+e^{-x}}$
- Sigmoid function values never fall below 0 and never exceed 1

ReLU Activation Function



- Rectified Linear Unit
- Will output 0 if the input values are negative

Neural Networks



Neural Networks

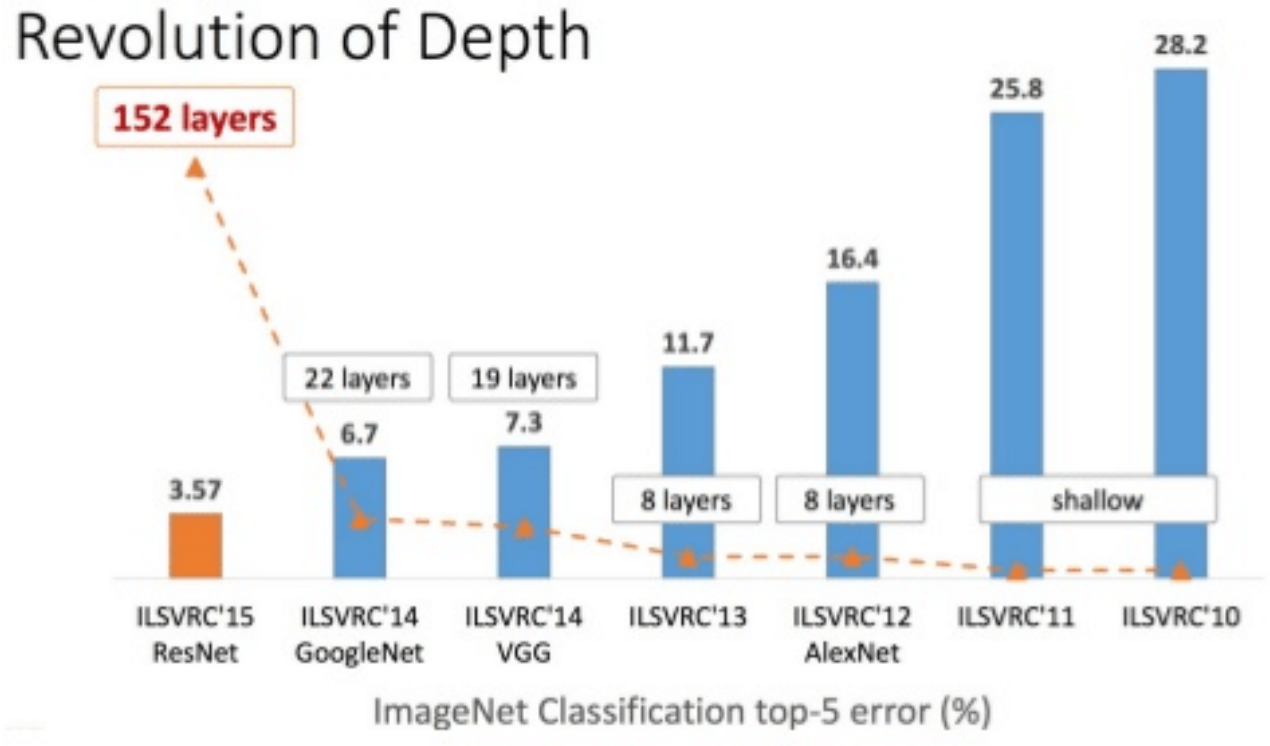
- Our job is to find all the weights and biases
  - Assume network structure and activation functions are given
- The number of neurons in the first layer is the dimension of the input data (number of regressors: m)
- No limit to the number of hidden layers or neurons
  - More layers and neurons means more parameters
- The number of neurons in the last layer is the dimension of the output data
  - Usually 1 for quantitative variables, like regression
  - Number of categories for classification
    - Output is “probability” of each category

Neural Networks

- How many hidden layers?
- How many nodes per hidden layer?
- What activation functions?

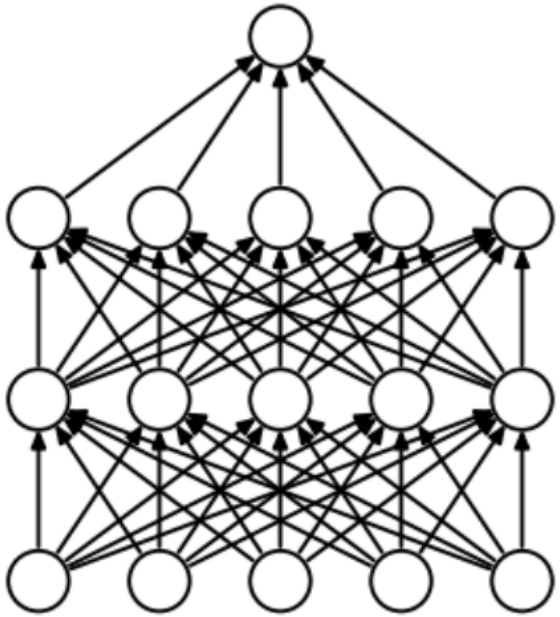
Depth Revolution

- In the early 2010's teams participating in the ImageNet competition started using deep neural networks
  - They drastically changed the study of NN's

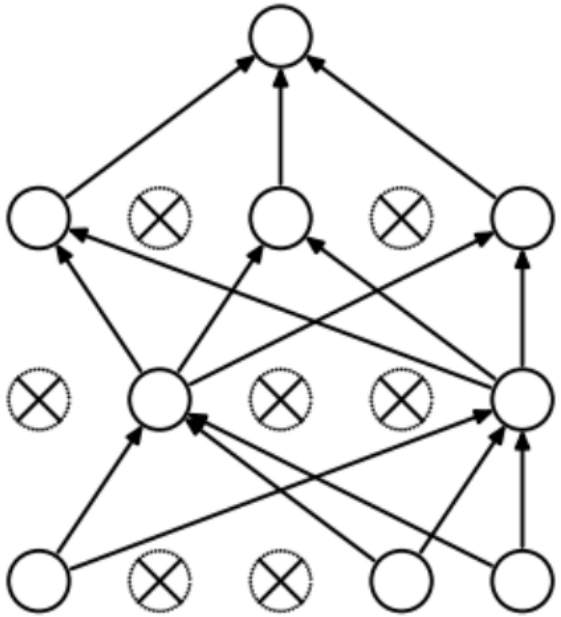


Neural Networks

- A popular way to prevent overfitting NN's is called dropout
  - On a step of SGD just randomly set some neurons equal to zero
  - On another step set a different group of neurons equal to zero



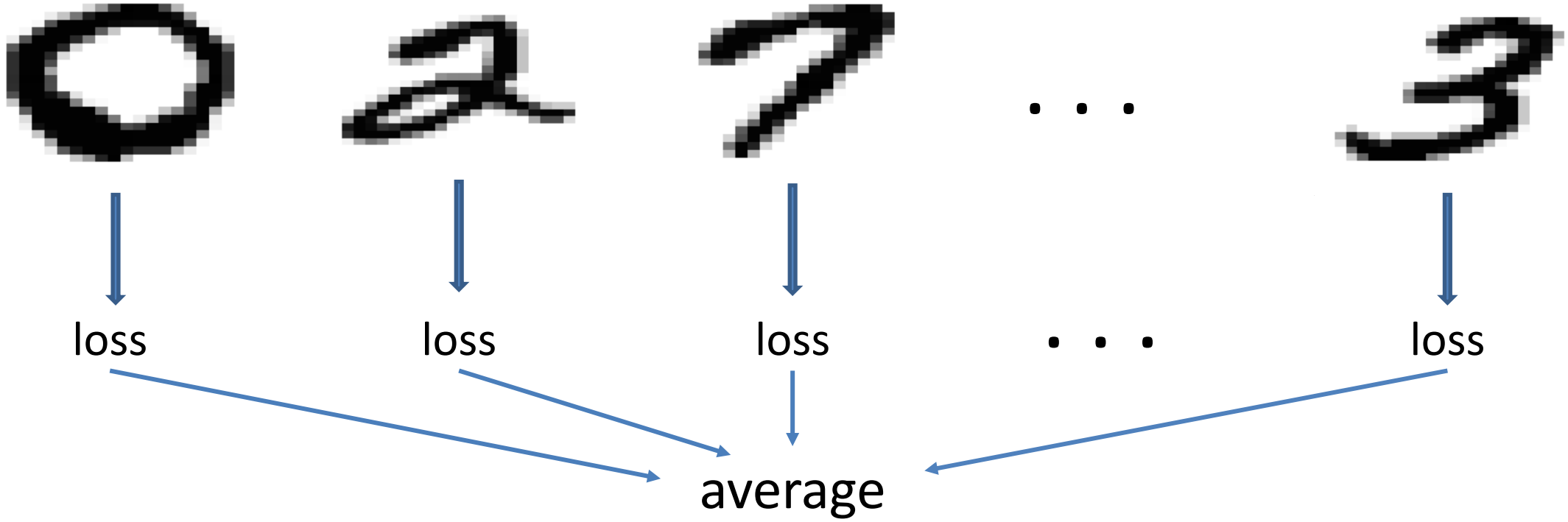
(a) Standard Neural Net



(b) After applying dropout.

Training a Neural Network – Loss

- To train a neural network we show it LOTS of images of characters that are already labeled
- Then calculate the loss for each character
- Average all these losses
- We want to find the weights and biases that makes this average loss as small as possible!



Training a Neural Network

- Training a neural network to find the weights and biases that lead to the minimum average loss is the hardest part!
- We need to find the gradient
  - This is DIFFICULT – Calculus and linear algebra!
- The algorithm we use to calculate these derivatives is called backpropogation
- Once we calculate the derivatives, we use them to find the best weights and biases!

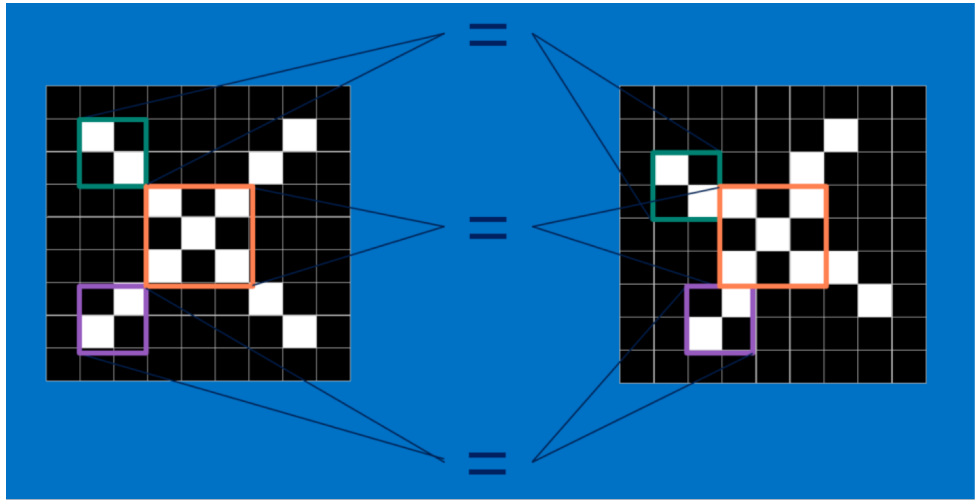
Let's try it!

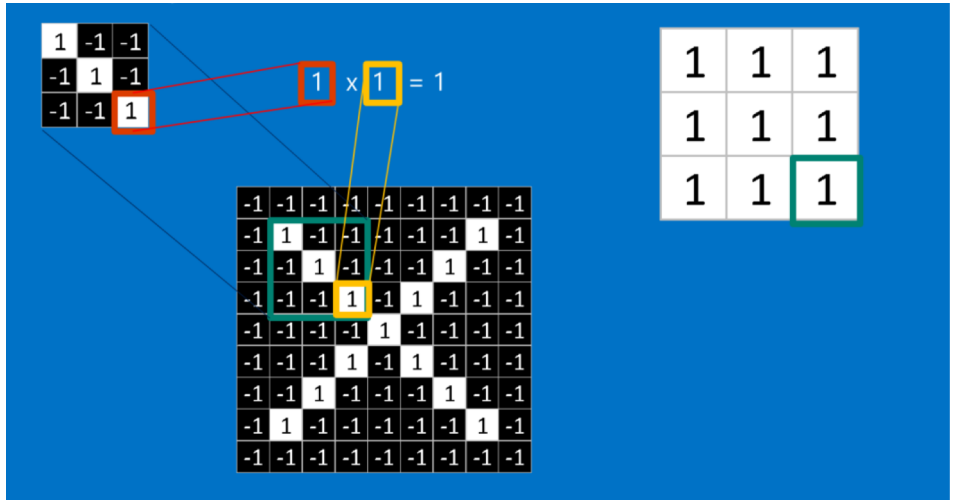
- Let's fit a neural network using TensorFlow

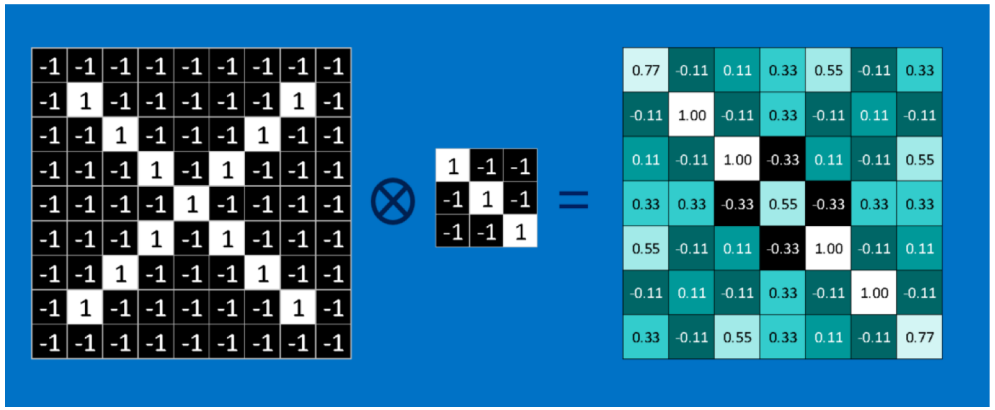
Neural Networks

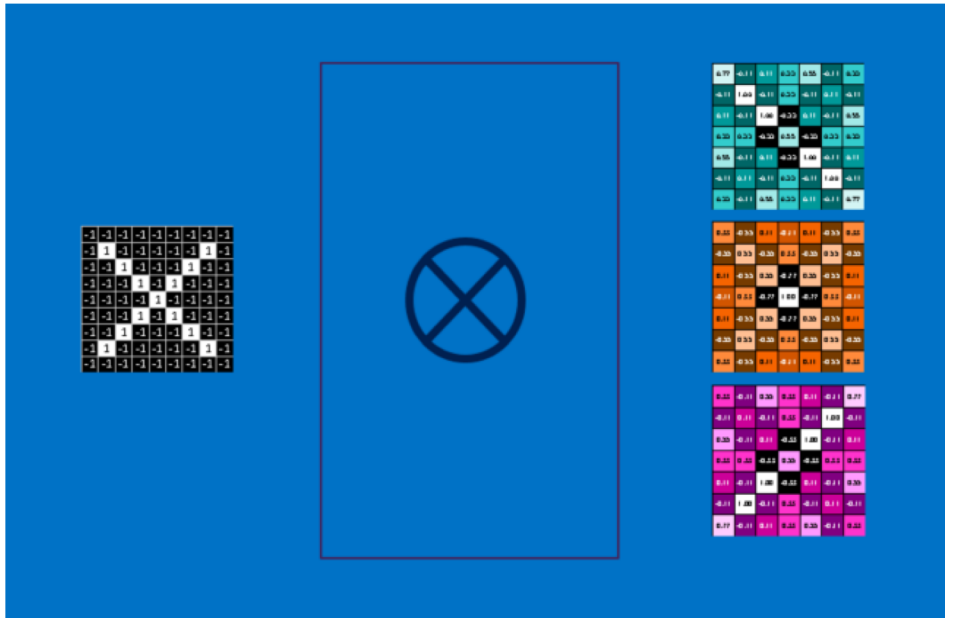
- There are other types of layers besides dense
- A layer used for images is called a convolutional layer
  - Apply a set of filters to an image
  - Take the output of the filters as the input of a dense neural network
- A filter is just a really small image, 3x3, 4x4, … pixels
- We apply the filter by going over every 3x3 set of pixels in the original image and seeing how close the original image is to the filter in that region
  - Record the closeness score everywhere
  - This is then the output of the layer

Neural Networks





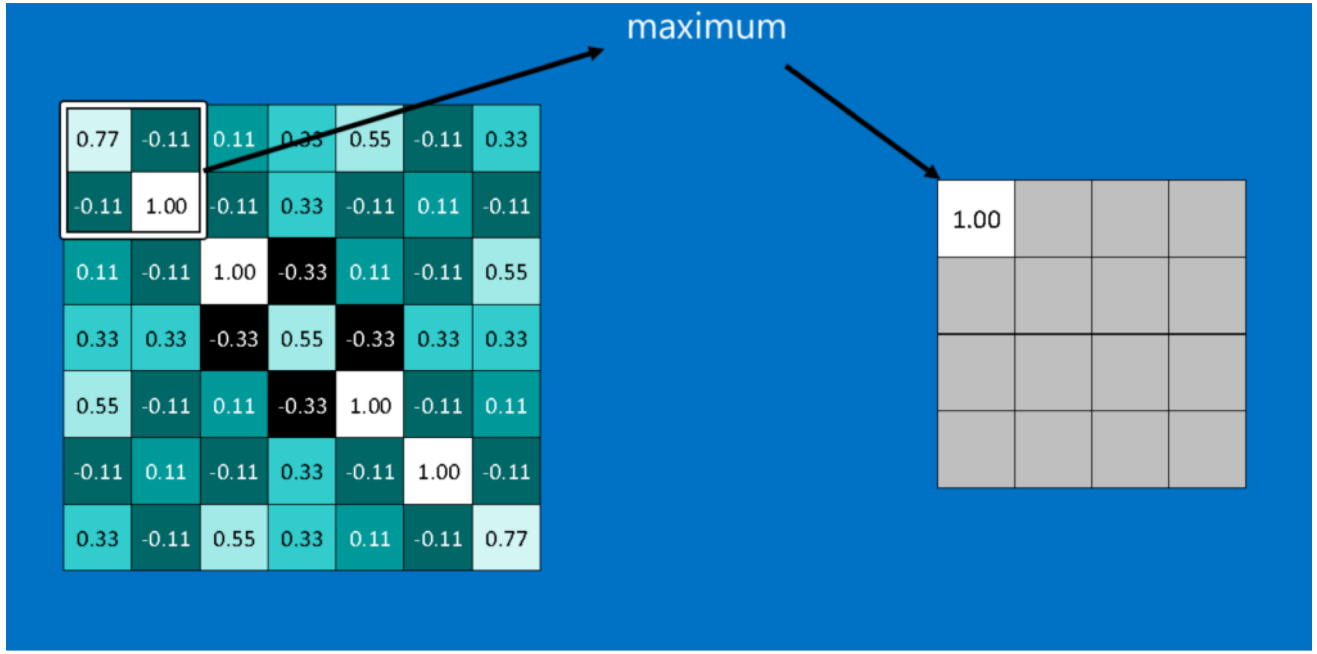


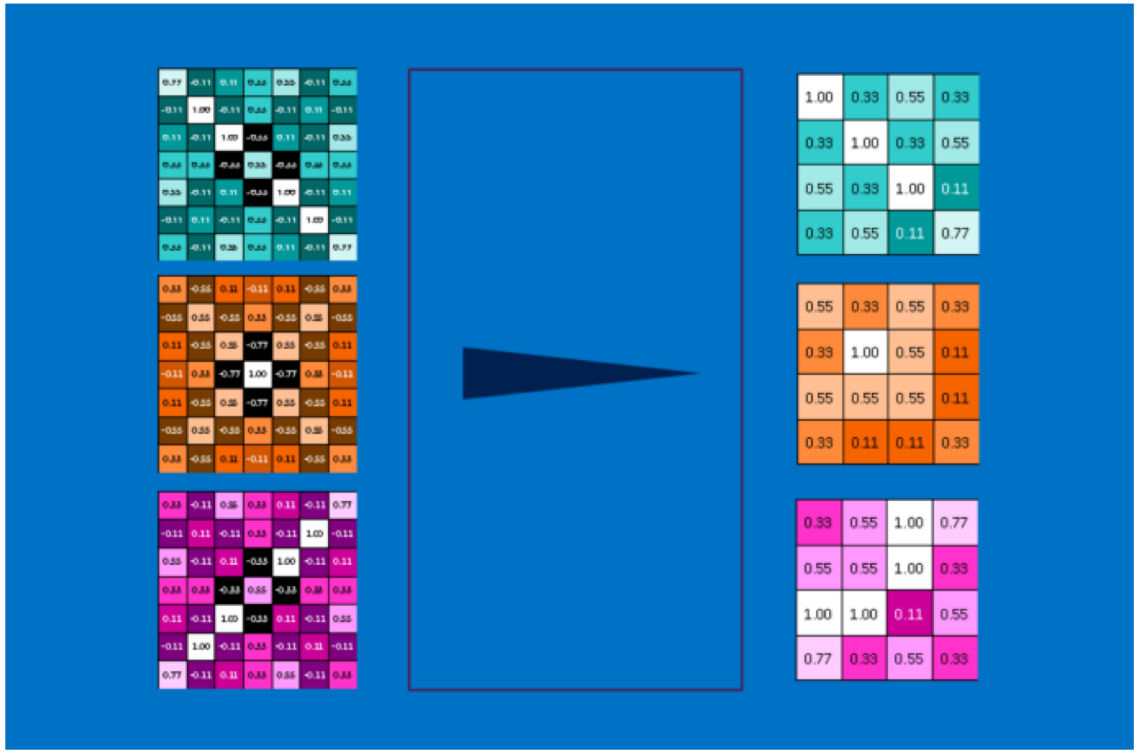


Neural Networks

- By applying 3 filters we have tripled the amount of data we have!
- This means in our full network we'll need a lot of neurons on our dense layer
- A common way to fix this is max pooling
- We take a small box and cycle it through the output of the filters
- Everywhere the box goes, we just remember the largest number in the box
  - Throw everything else away
  - Don't let the boxes overlap

Max Pooling


maximum
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
1.00


1.00 0.33 0.55 0.33
0.33 1.00 0.33 0.55
0.55 0.33 1.00 0.11
0.33 0.55 0.11 0.77
0.55 0.33 0.55 0.33
0.33 1.00 0.55 0.11
0.55 0.55 0.55 0.11
0.33 0.11 0.11 0.33
0.33 0.55 1.00 0.77
0.55 0.55 1.00 0.33
1.00 1.00 0.11 0.55
0.77 0.33 0.55 0.33

Neural Networks

- For the convolutional step, where did we find the filters?????
- The neural network treats the entries in each pixel of the filters as parameters to be learned
- Back propagation then learns what the filters are
- We still have to pick the size and number of filters