Introduction to Neural Networks (NN)

Goals

- Application of NN
- Software Ecosystem
- Logistic / Linear Regression as example of NN
- Multi Hidden Layer NN

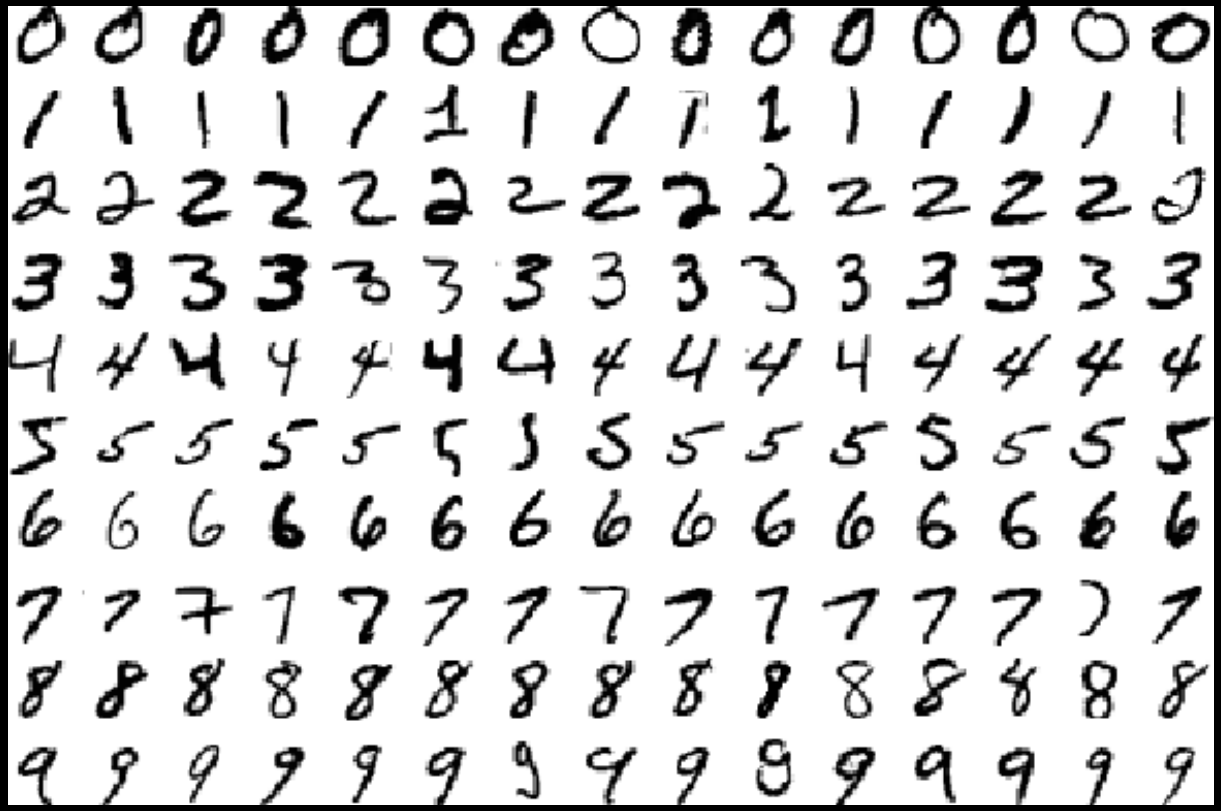
Why Neural Net

- Superior to computer vision techniques
- Auto feature engineering
- No need for manual inspection

Why Neural Net 2

- Able to recognize patterns of complex / unexpected inputs
- Image / Text / Audio / Video Search

MNIST Dataset



Why Neural Net 2

Classifier	Test Error (%)	Reference
Large/deep conv. net	**99.77**	**Multi-column Deep Neural Networks for Image Classification**
Virtual SVM, deg-9 poly, 2-pixel jittered	99.44	Training Invariant Support Vector Machines
product of stumps on Haar f.	99.13	Boosting products of base classifiers

Why not Neural Net

- Hard to tune
- Computationally intensive
- **Lack of feature introspection**
  - ★ arxiv.org/abs/1503.02531

Ecosystem (May, 2015)

- **Python**
  - **Theano (Low level + GPU)**
    - deeplearning.net/software/theano/
  - **Lasagne (Abstraction of Theano)**
    - lasagne.readthedocs.org/en/latest/index.html
  - **Others:** https://www.cbinsights.com/blog/python-tools-machine-learning/
- **Java / Scala**
  - **DL4J (Run on distributed systems)**: deeplearning4j.org/
- **C++**
  - **Caffe (GPU):** caffe.berkeleyvision.org/

Terminology

Neural Network == **Artificial Neural Network**

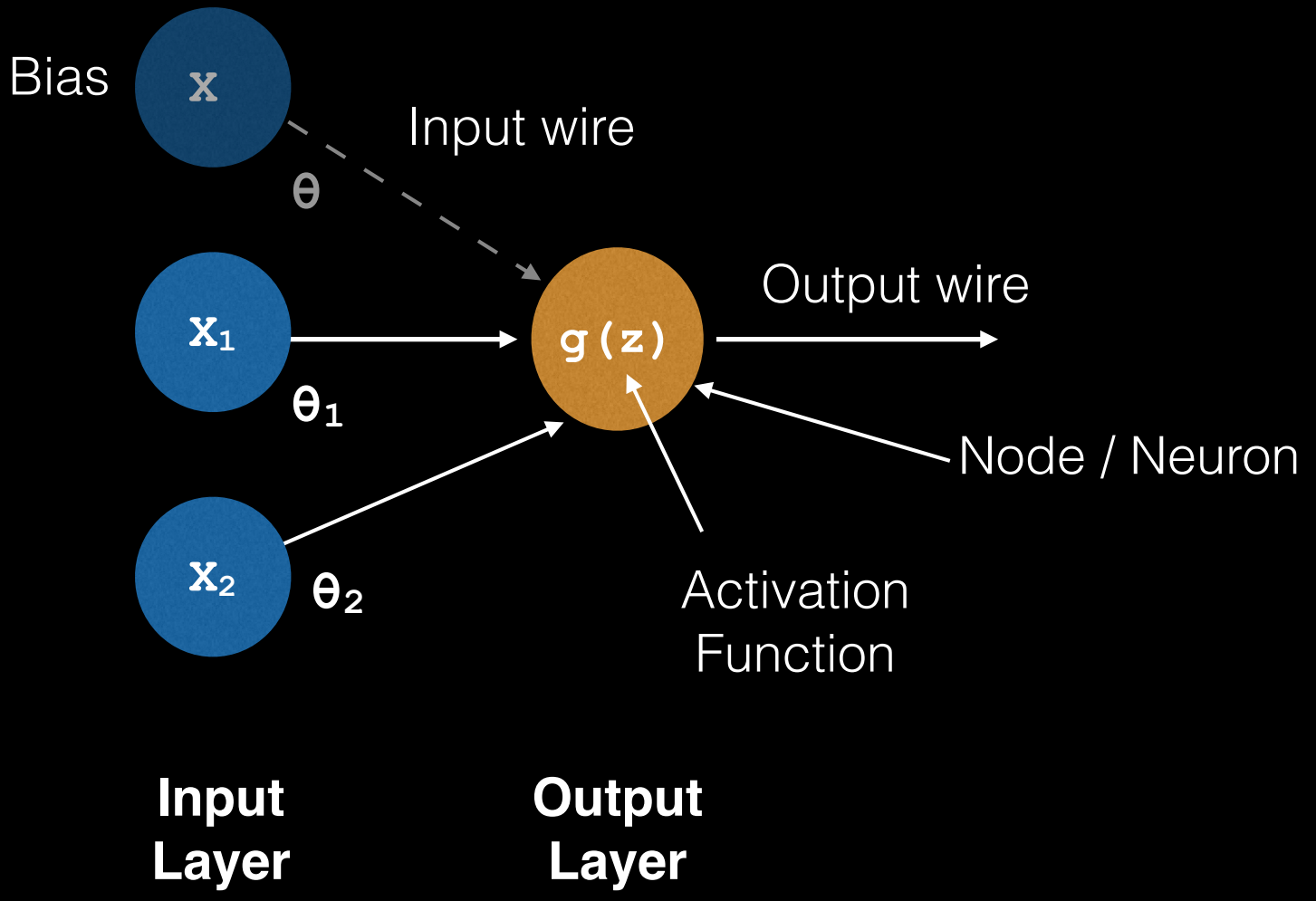
Deep Learning

- 3 or more hidden layers in a NN

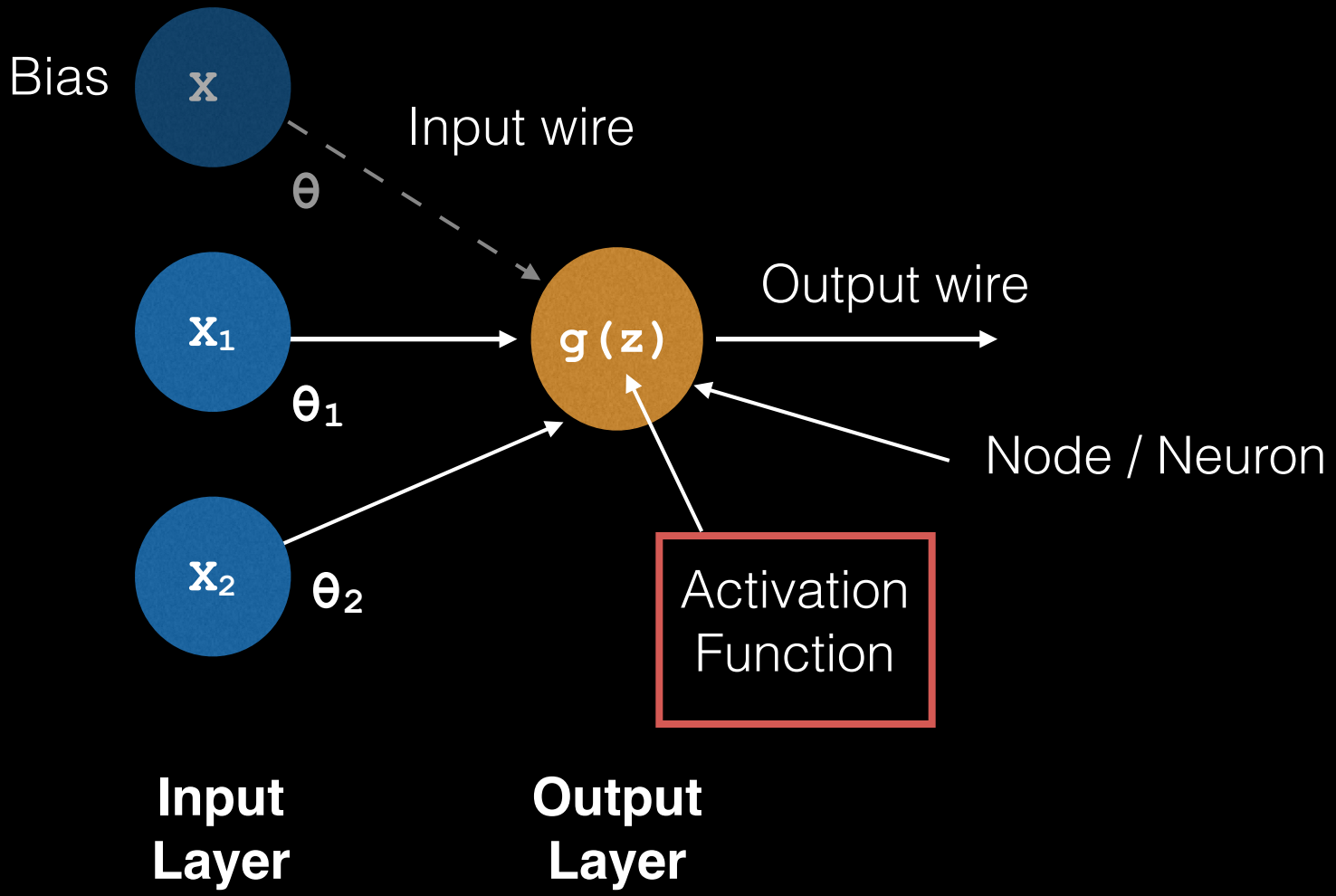
Feed-Forward NN

- Information moves in **1 direction**
- Input **—>** Output

Components of NN (2-layer)

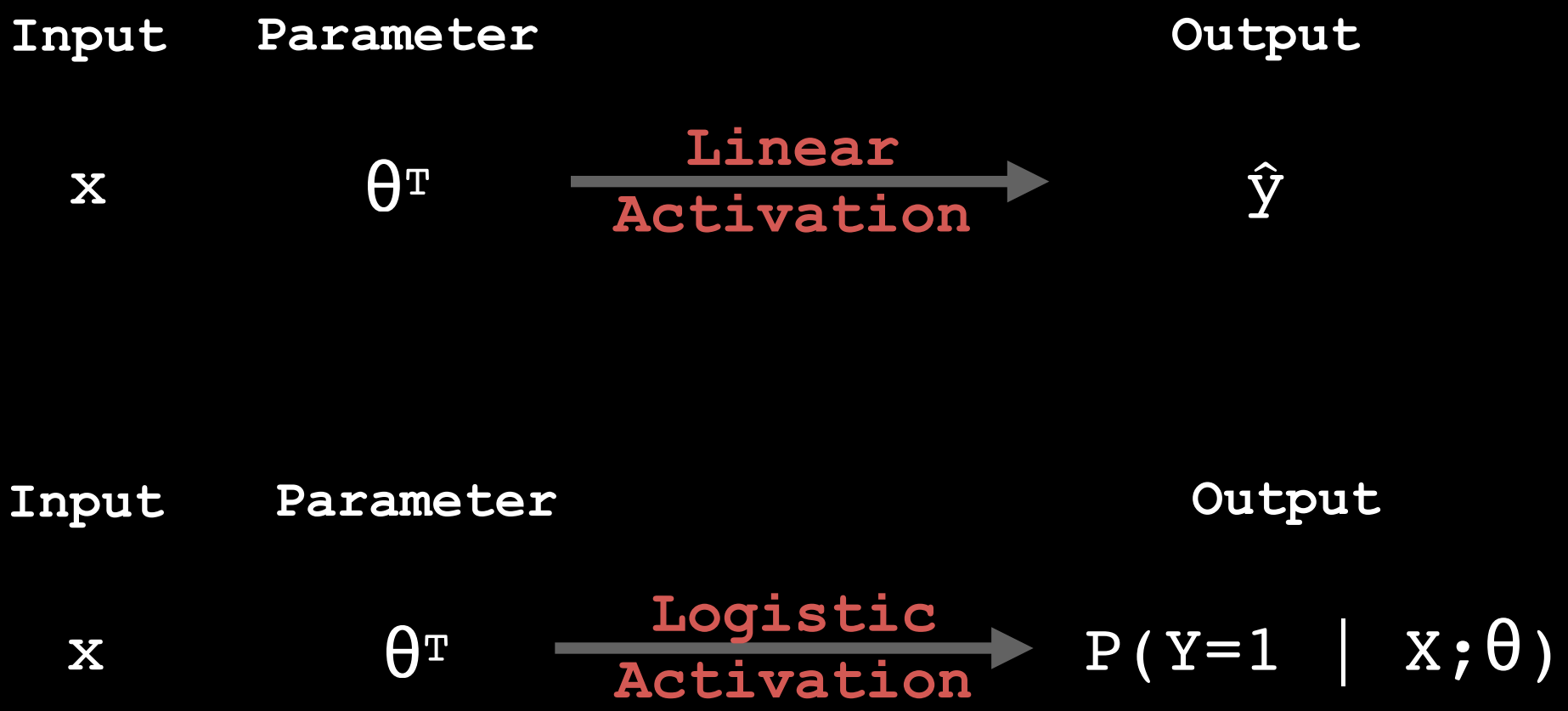


Components of NN (2-layer)



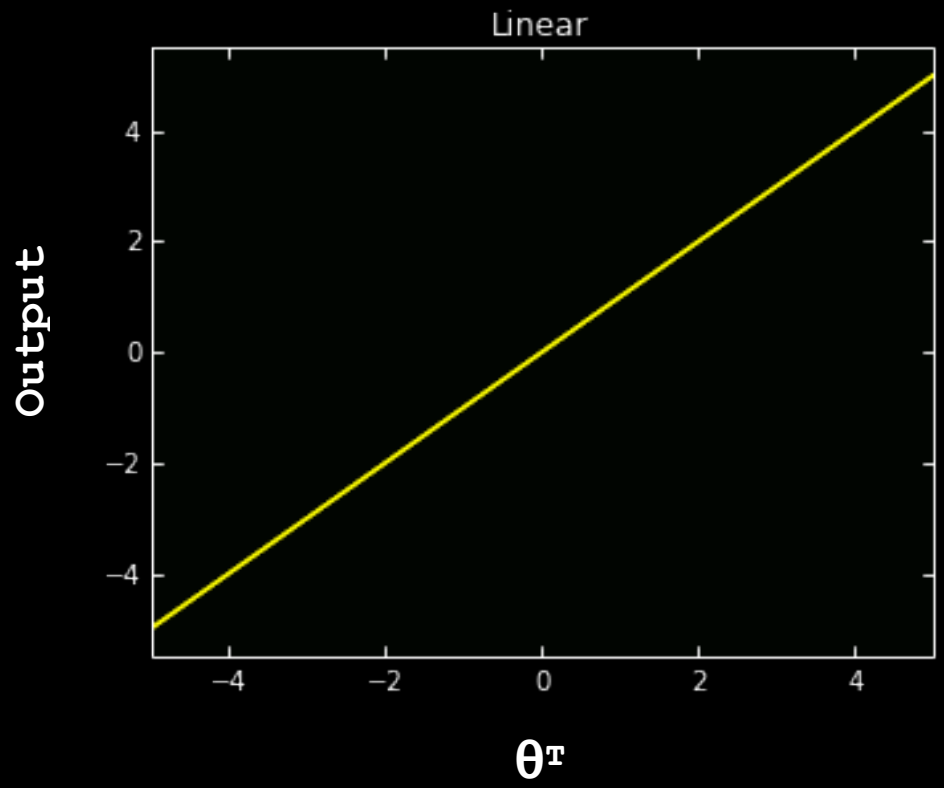
Activation Function

- **Transforms** certain input to an other domain

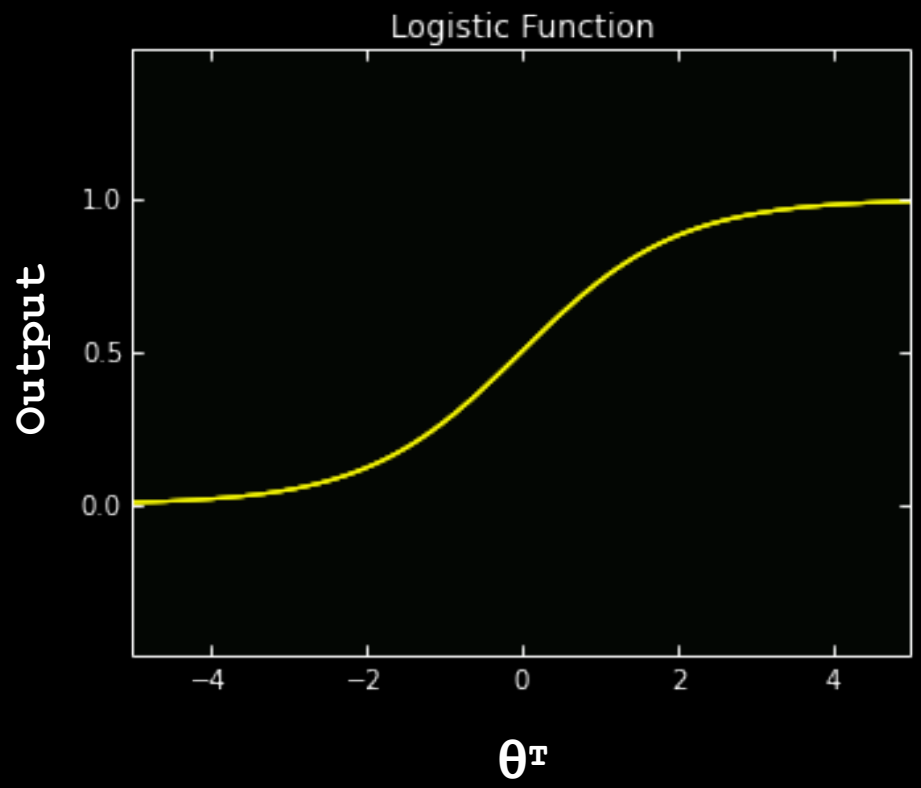

Input
Parameter
Output
x
θ^T
Linear
Activation
ŷ
Input
Parameter
Output
x
θ^T
Logistic
Activation
P(Y=1 | X;θ)

Output Layer Activation Function

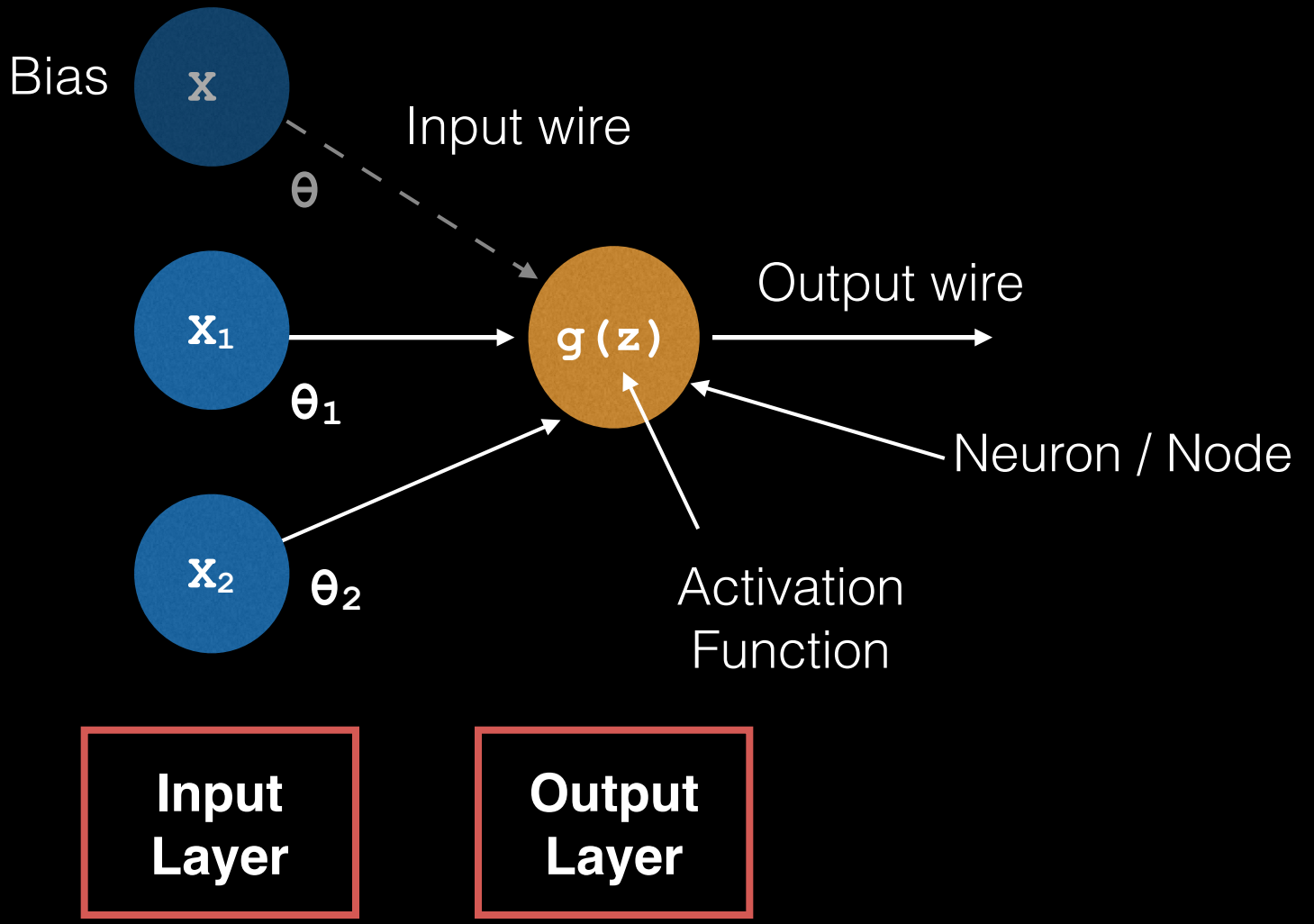
(Regression)



(Classification)



Components of NN (2-layer)



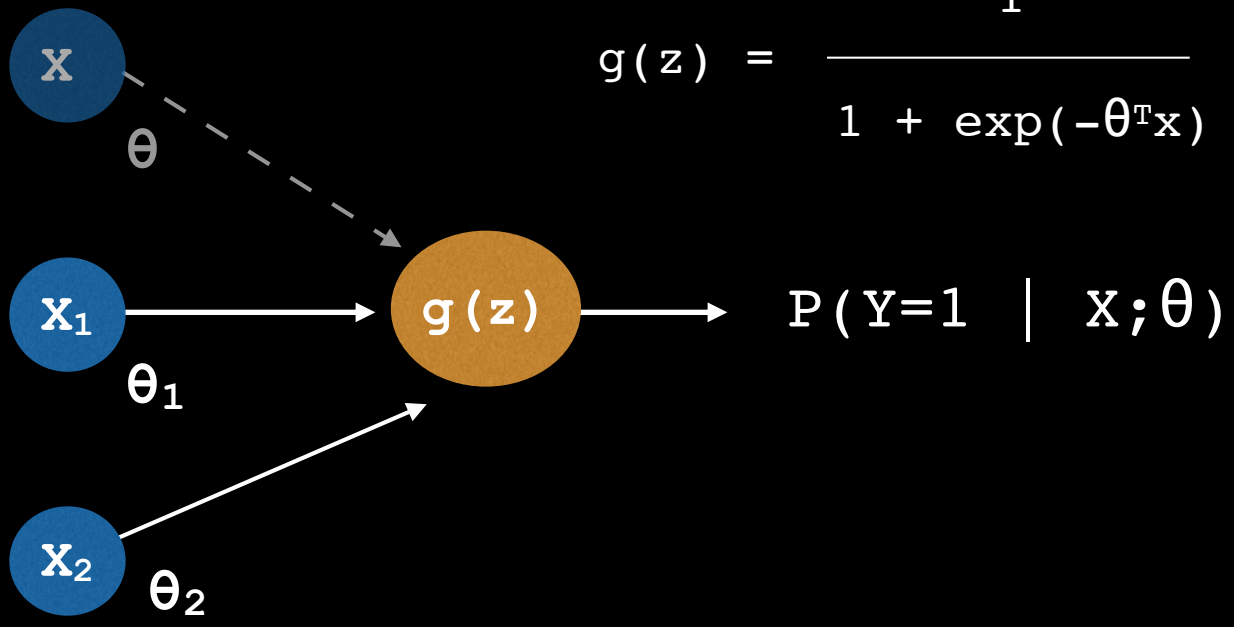
How many input nodes

- Number of features (x_i) + 1 bias term

How many output nodes

- **Continuous response (Regression)**
  - ⋆ 1 output node
- **Discrete response (Classification)**
  - ⋆ Number of unique responses

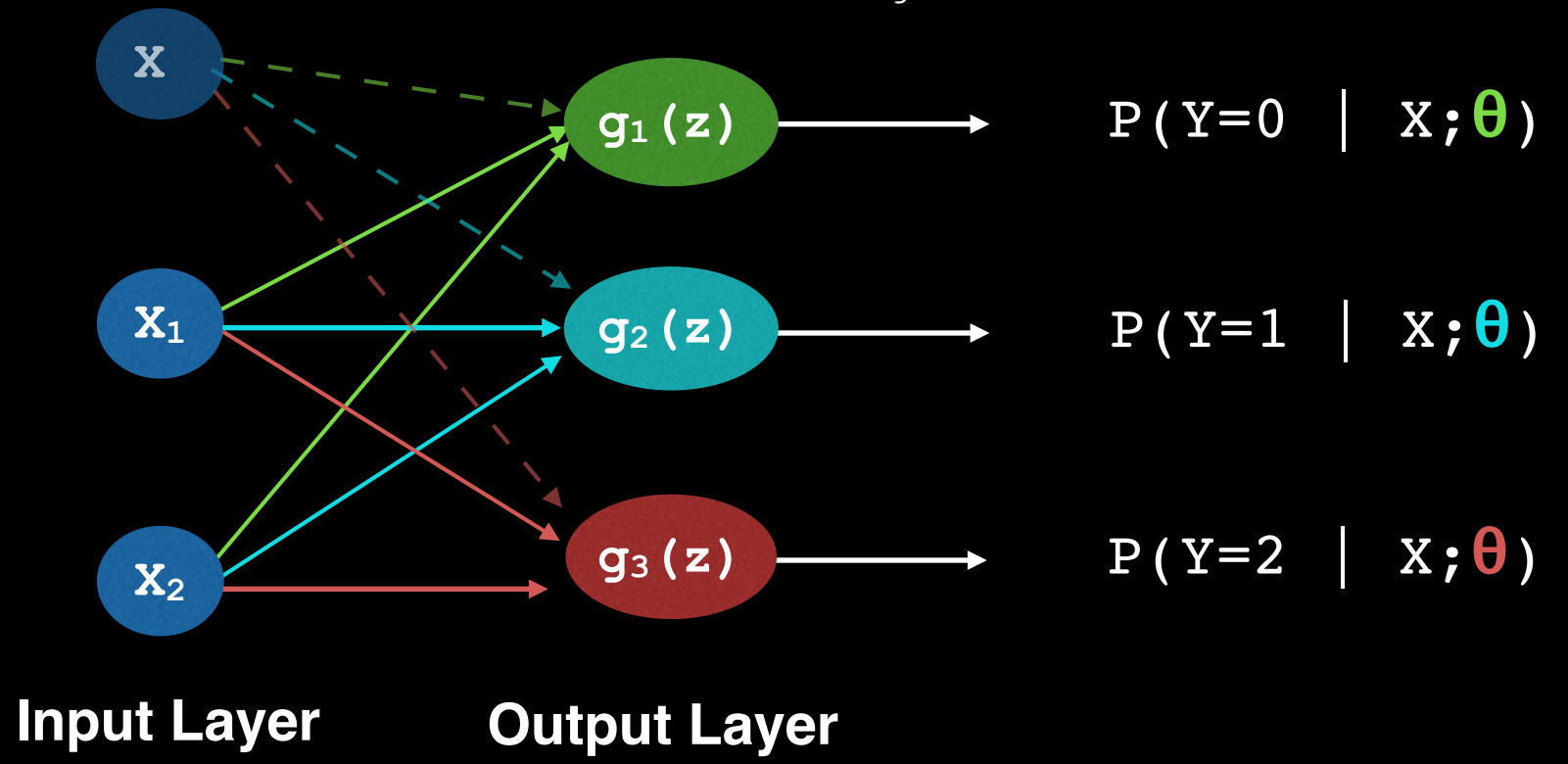
Logistic Regression (2 classes)



$$g(z) = \frac{1}{1 + \exp(-\theta^T x)}$$

Logistic Regression (3 classes - Softmax)

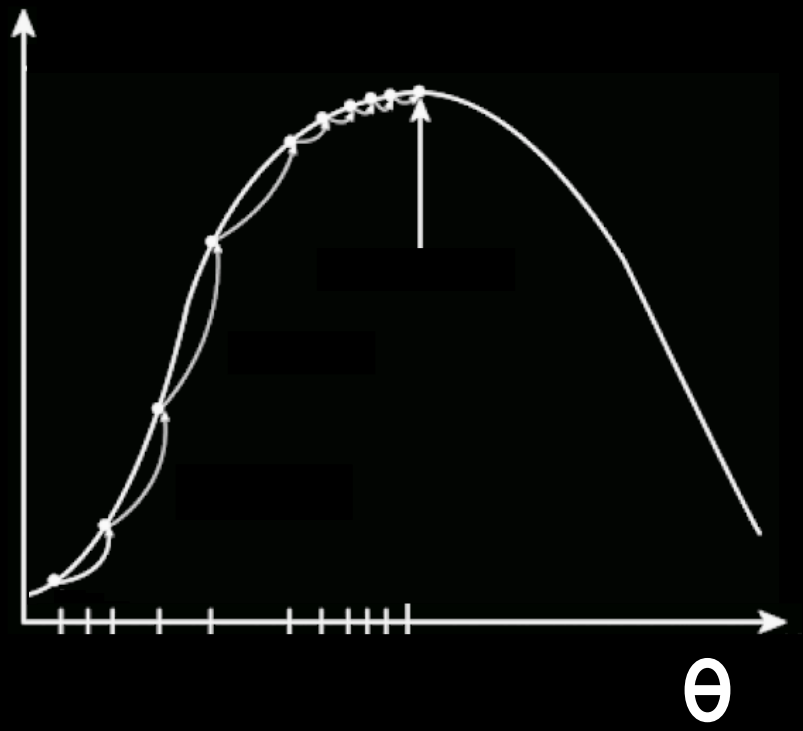
$$g_i(z) = \frac{\exp(\theta_i^T x)}{\sum_{j=1}^{k} \exp(\theta_j^T x)}$$



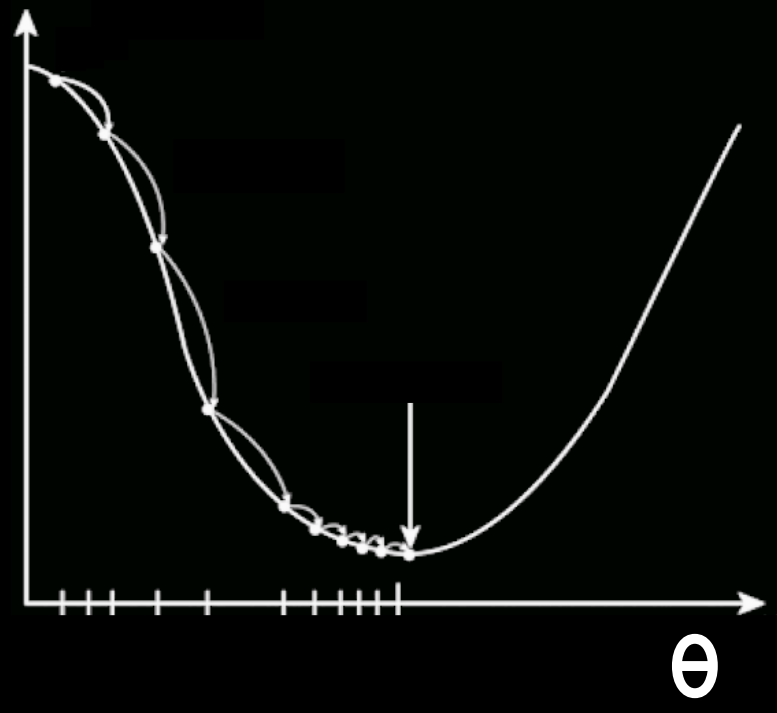
Optimization

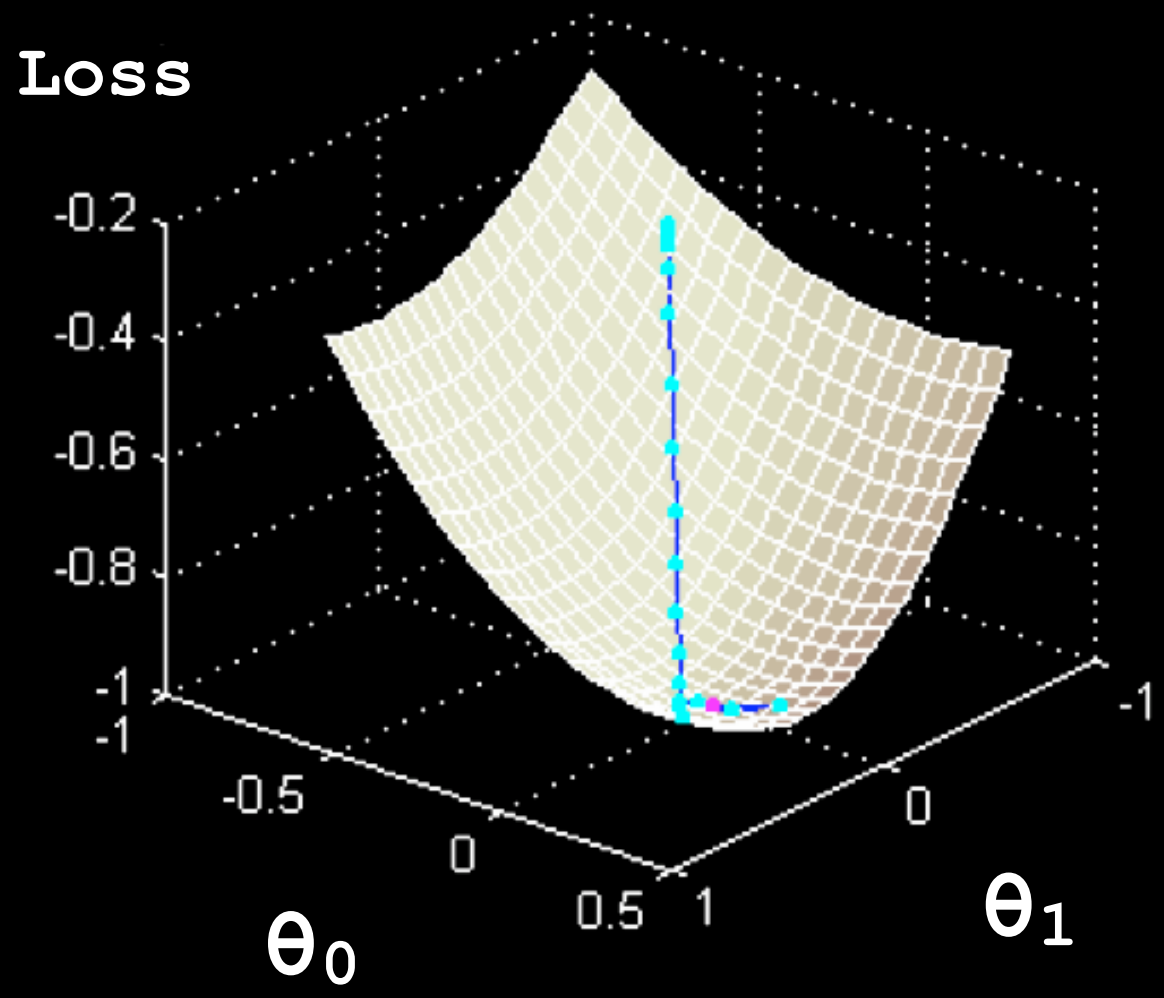
- Optimize $\boldsymbol{\theta}$
- Minimize loss function / Maximize Likelihood
- Gradient descent / ascent optimization
- Incrementally adjust $\boldsymbol{\theta}$ to min. loss / max. likelihood

Likelihood

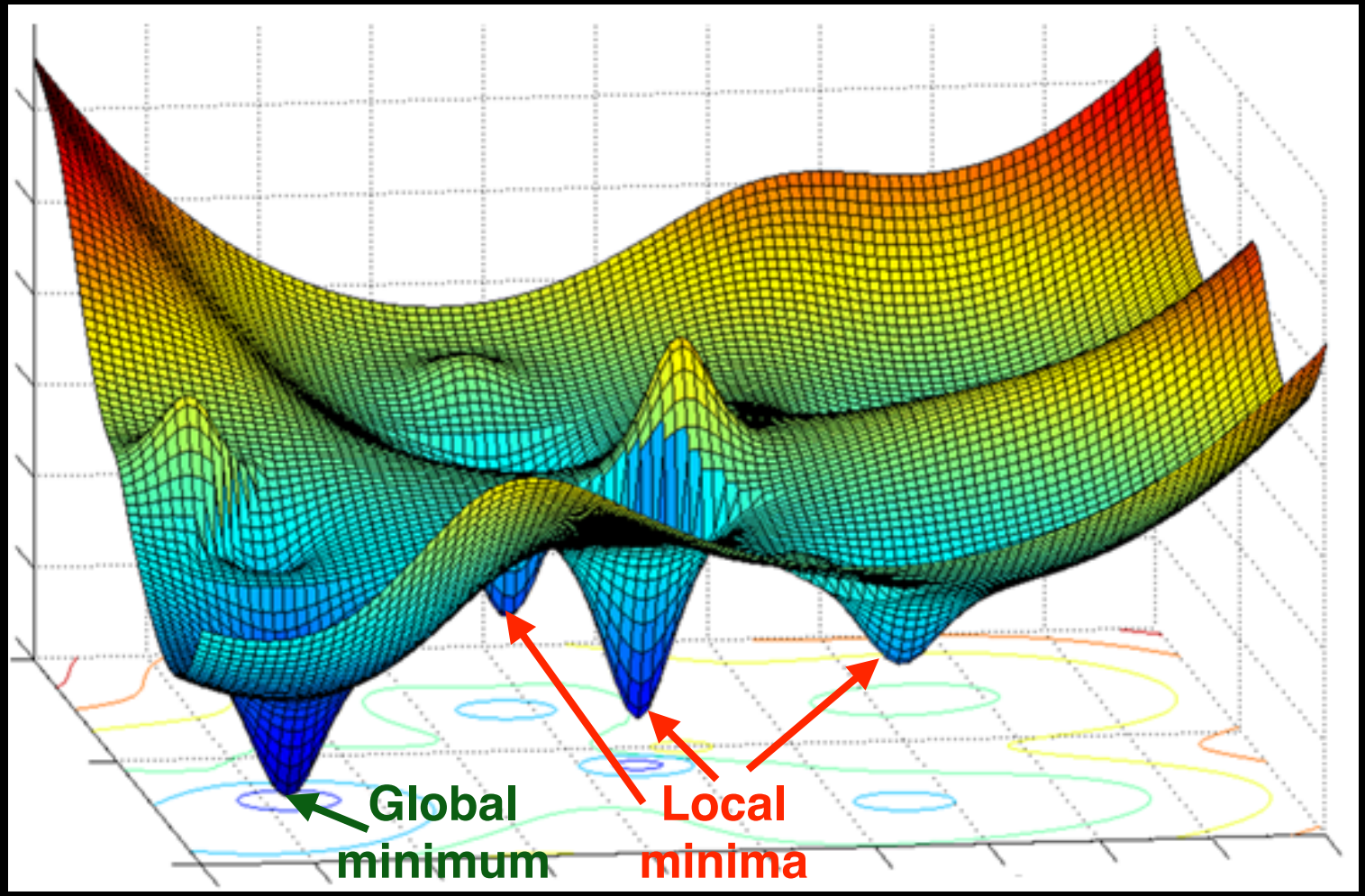


Loss




Loss
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
θ0
0
-1
θ1

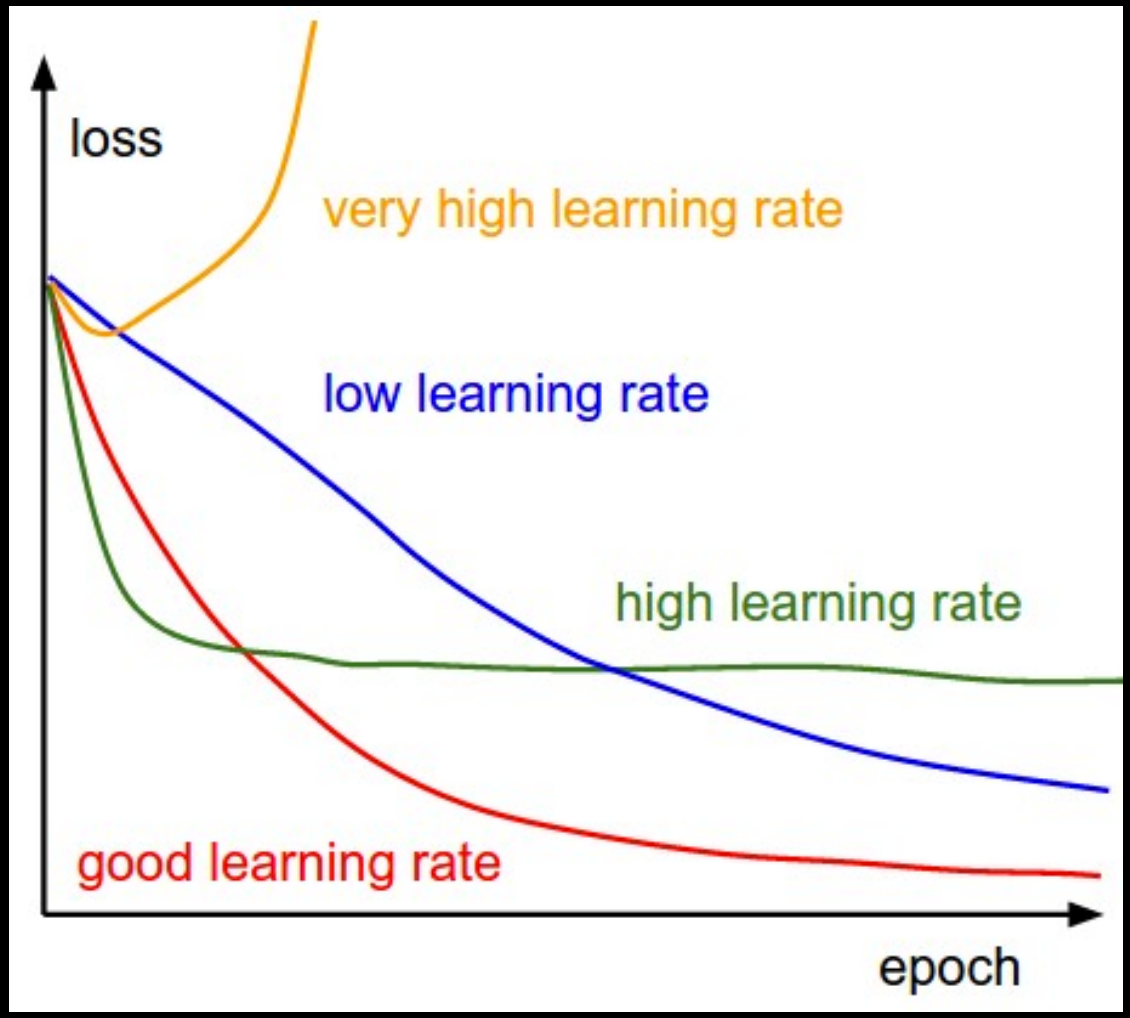
Local minima



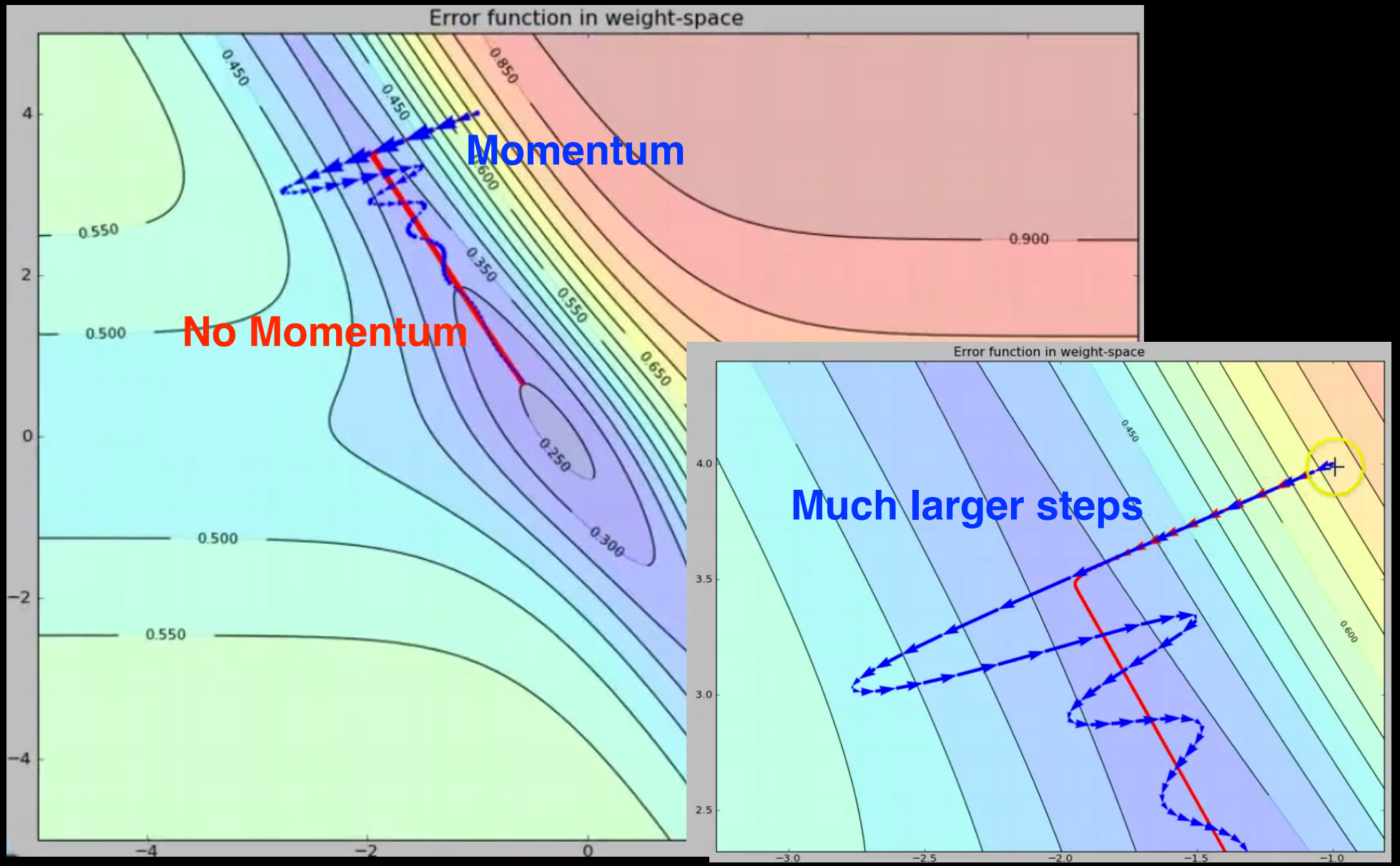
Hyper-parameters

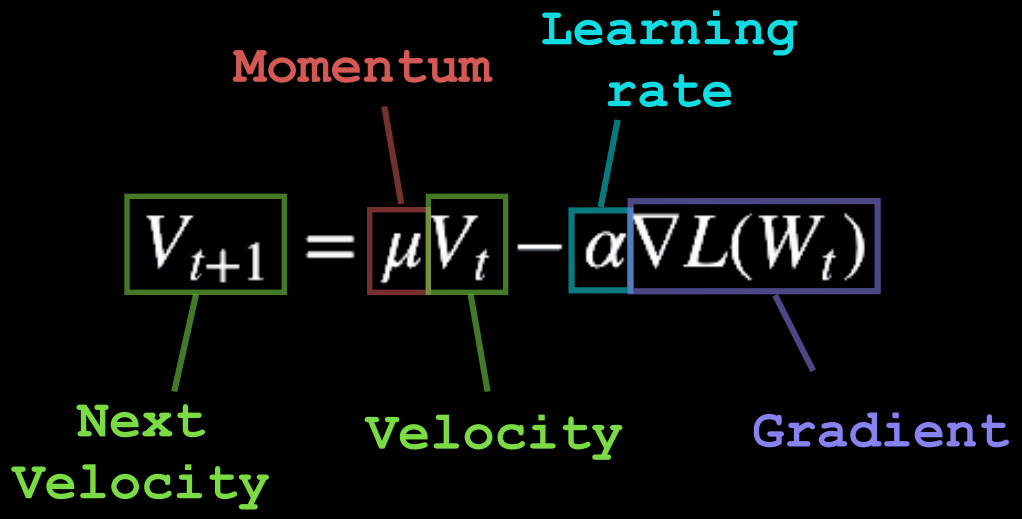
- **Learning Rate**
  - ⋆ Update a fraction of the gradient
- **Momentum**
  - ⋆ Update **θ** incorporating the last “gradient”
- **Epoch**
  - ⋆ Times going through the whole set of data

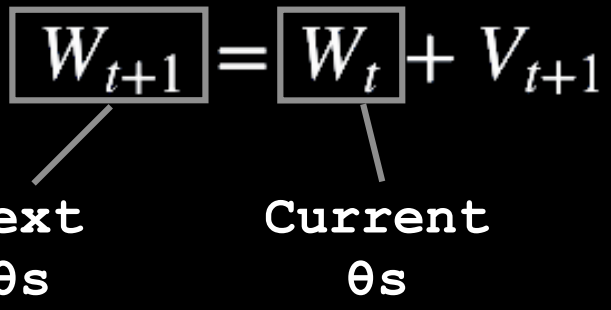
Learning Rate



Momentum


Error function in weight-space
Momentum
No Momentum
Error function in weight-space
Much larger steps


Momentum
Learning rate
$V_{t+1} = \mu V_t - \alpha \nabla L(W_t)$
Next Velocity
Velocity
Gradient


$W_{t+1} = W_t + V_{t+1}$
Next θs
Current θs

Epoch vs Iteration

- An **iteration** is an update of θ
- An **epoch** contains **iterations** through the whole data set

Mini-batch Gradient Descent

- Compute gradient with few data points
- Shuffle data and loop through mini-batches
- Repeat until convergence
- Generally faster than batch GD
  (More frequent gradient updates)

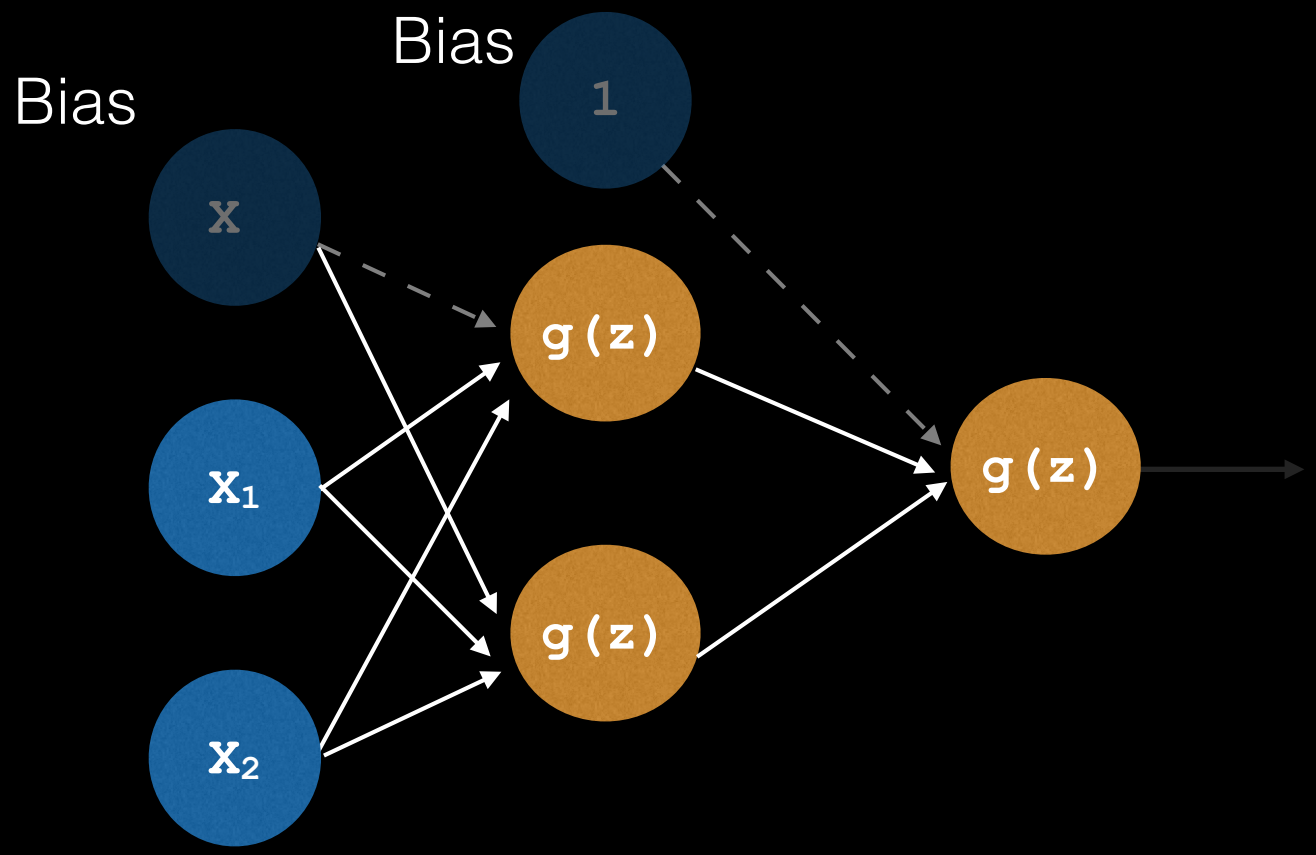
Review

- Basic architecture of feed-forward NN
- Activation functions
- Gradient Descent Recap
- Different hyper-parameters to optimize parameters
- **Tips and Tricks**
  - ⋆ research.microsoft.com/pubs/192769/tricks-2012.pdf
- Mini-batch GD

Break

Multi Hidden Layer NN

3-layer NN



Input Layer **Hidden Layer** **Output Layer**

Hidden Layer

- **Hidden:** You do not observe the output
- Can have multiple hidden layers
- 1 bias term per hidden layer

How many hidden layers

- **Usually ≤ 3**
- More layers for more complex relationships
- More hidden layers = Harder to optimize
- Longer to train and tune hyper-parameters

How many Neurons

- **Per layer:**

$$\#\text{ of outputs} \leq x \leq \#\text{ of inputs}$$

(High # of inputs)

- **Overall:**

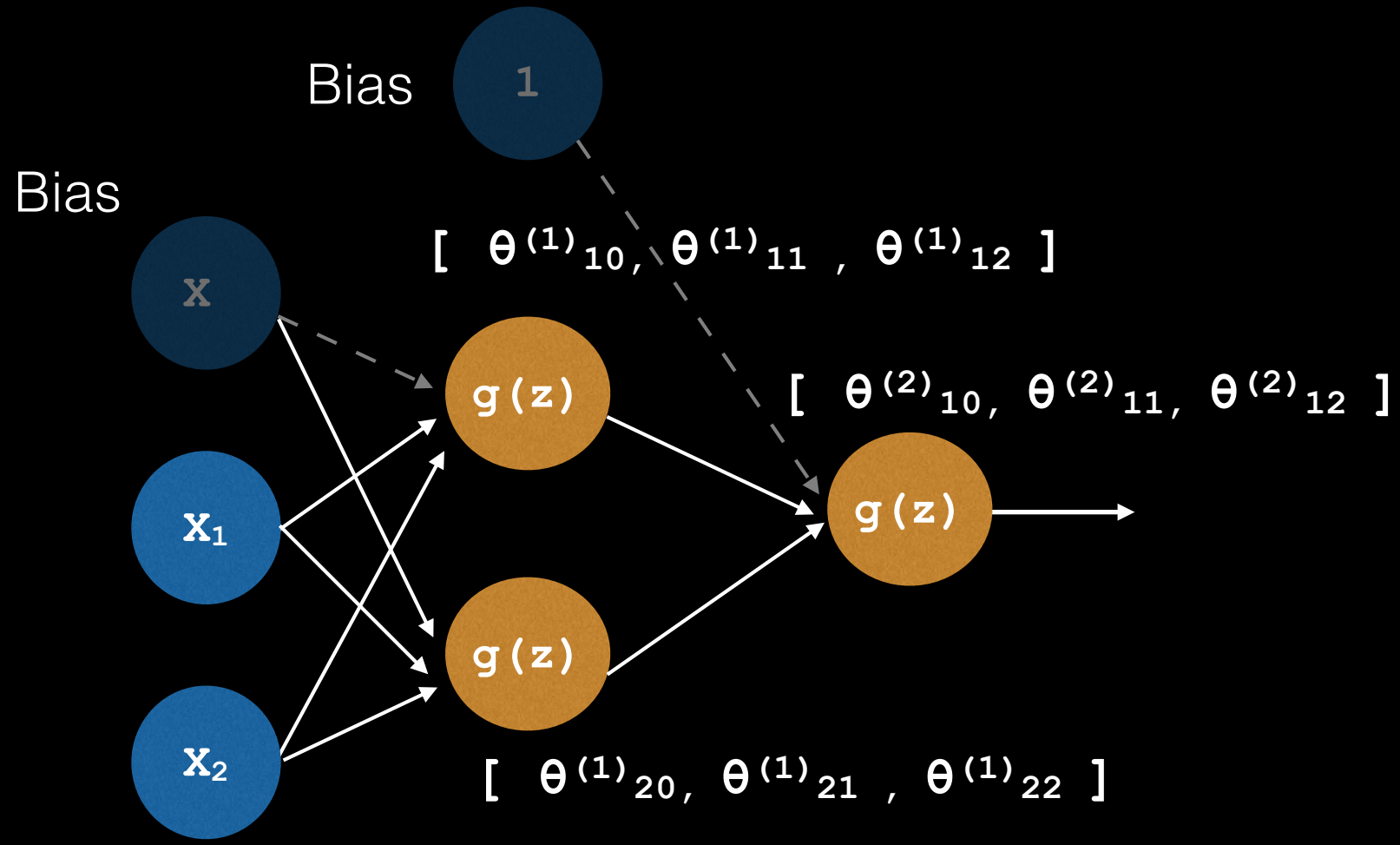
$$\#\text{ of data points} \geq \frac{\#\text{ of }\theta\text{s}}{30}$$

(Noisy data)

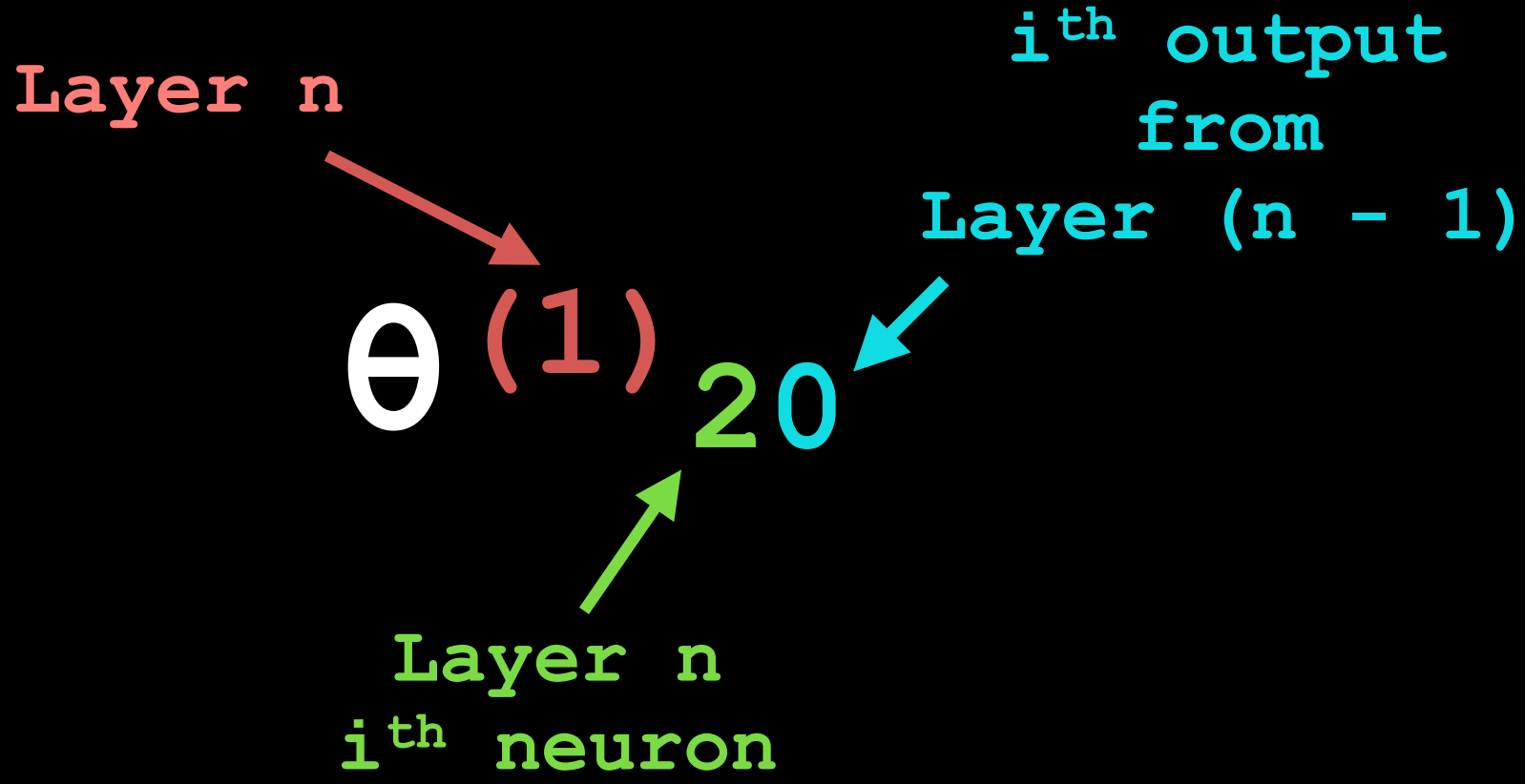
$$\#\text{ of data points} \geq \frac{\#\text{ of }\theta\text{s}}{2}$$

(Completely noise free)

How many θs

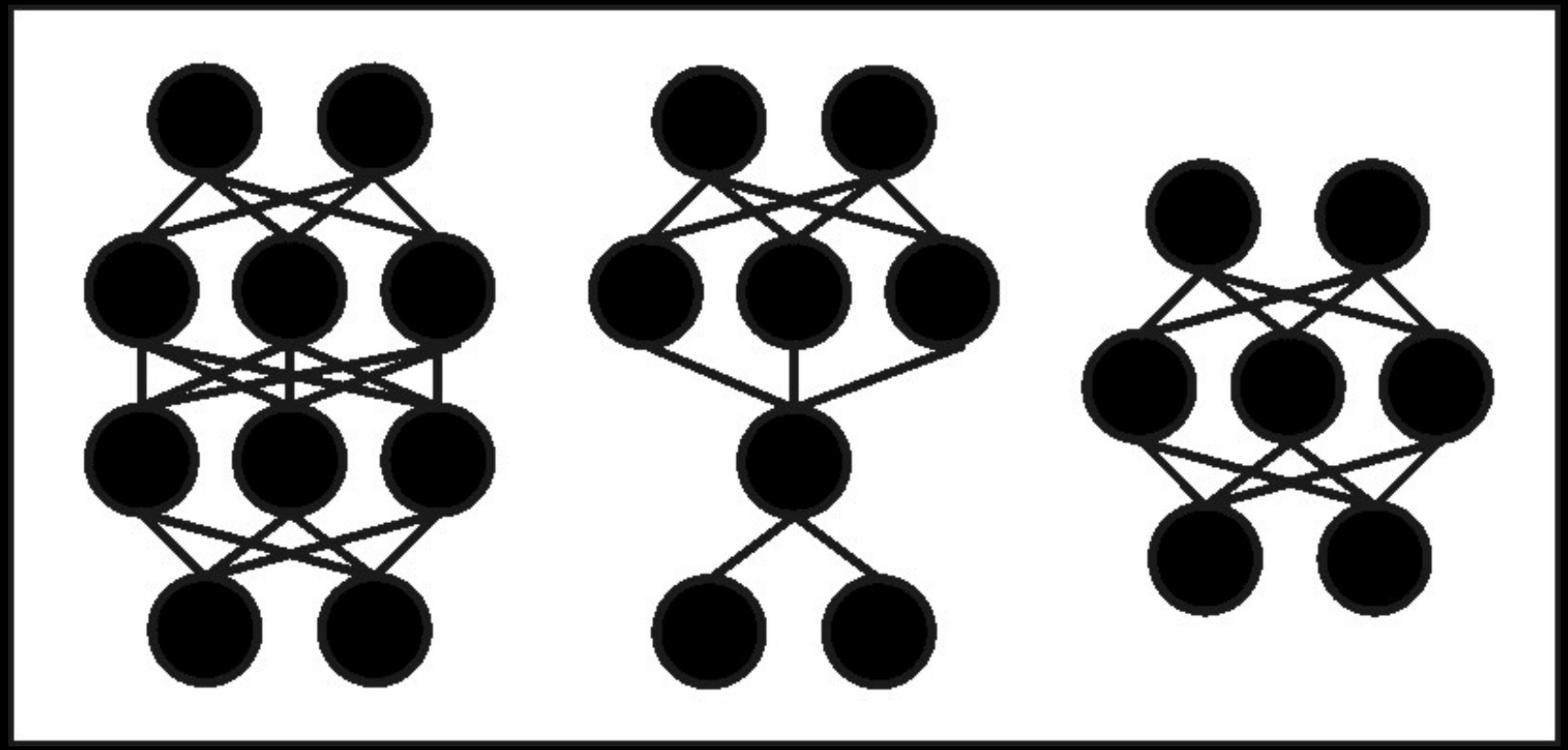

Bias
1
Bias
x
x_1
x_2
g(z)
g(z)
g(z)
$[\ \theta^{(1)}_{10}, \theta^{(1)}_{11}, \theta^{(1)}_{12}\ ]$
$[\ \theta^{(2)}_{10}, \theta^{(2)}_{11}, \theta^{(2)}_{12}\ ]$
$[\ \theta^{(1)}_{20}, \theta^{(1)}_{21}, \theta^{(1)}_{22}\ ]$

θ Notation

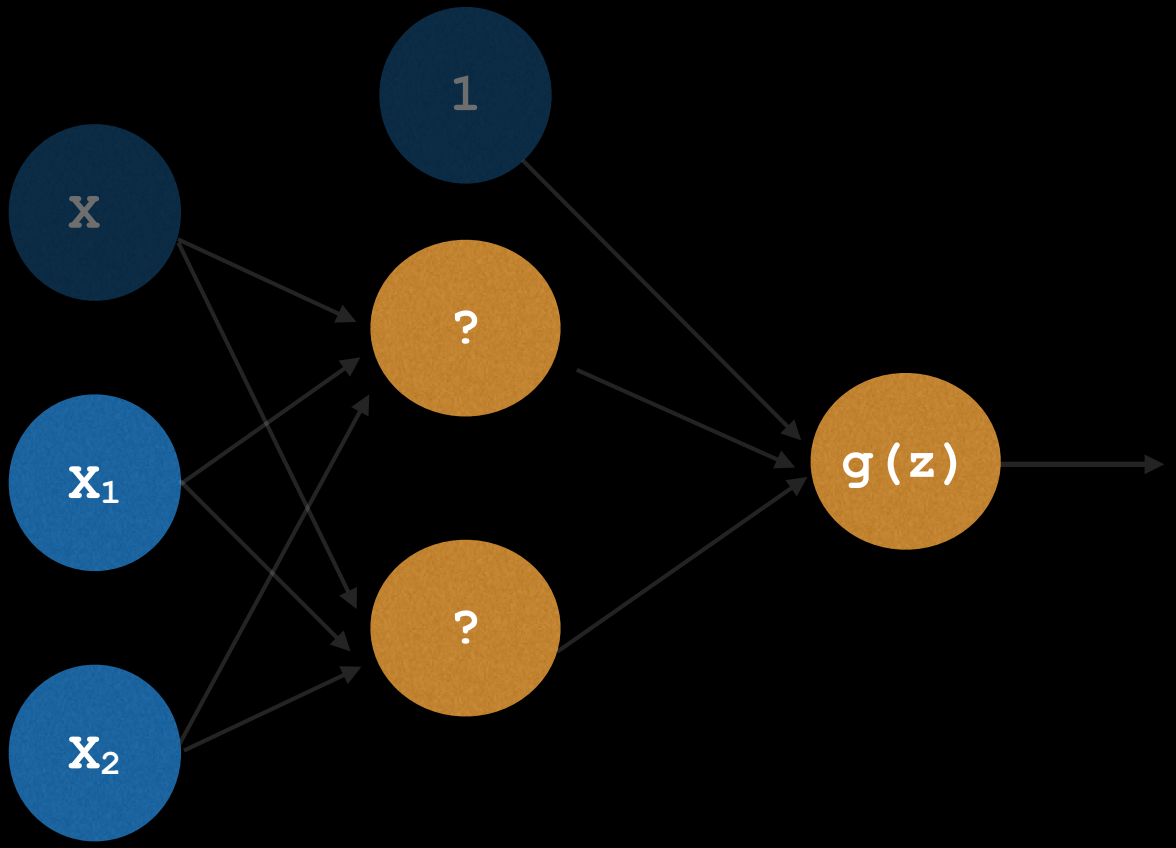


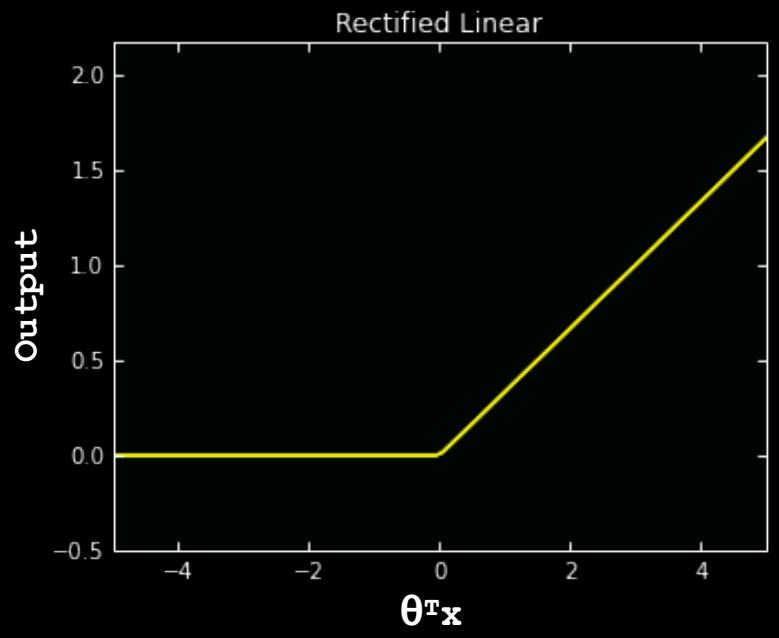
- `# of layers` , `# of neurons` are guidelines
- Much debate around
  1. **Network configuration**
  2. **Hyper-parameter tuning**
- Mostly trial-and-error

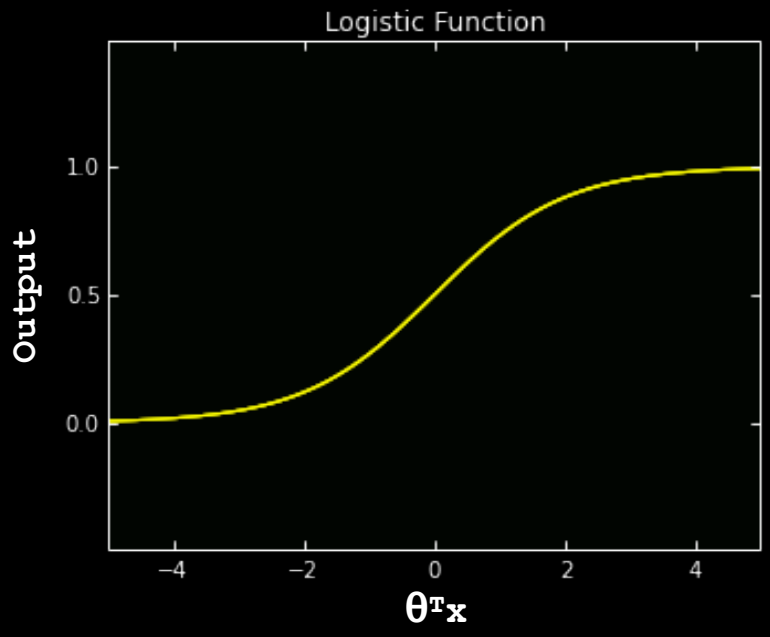
Network Topology

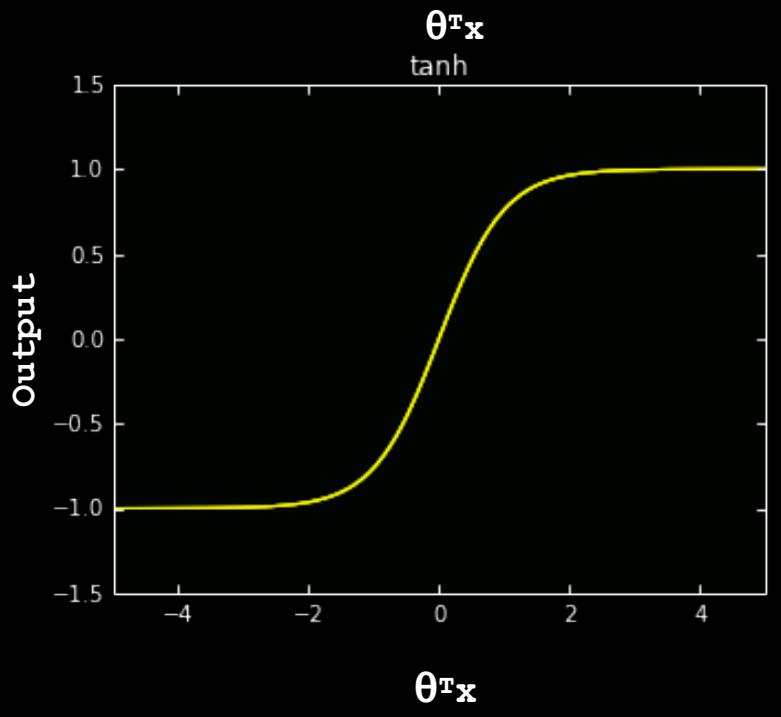


Activation Function (Hidden Layer)




Rectified Linear
Output
2.0
1.5
1.0
0.5
0.0
−0.5
−4
−2
0
2
4
θᵀx


Logistic Function
Output
1.0
0.5
0.0
−4
−2
0
2
4
θᵀx


θᵀx
tanh
Output
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−4
−2
0
2
4
θᵀx

Rectify Linear (ReL/ ReLU)

- Most common activation function
- Quicker to optimize (Fewer epochs)

Rectify Linear Cont.

- Zero out data where $\theta^T x < 0$
- Emphasize data where $\theta^T x > 0$ (Without bounds)
- Sparsity / Overfitting

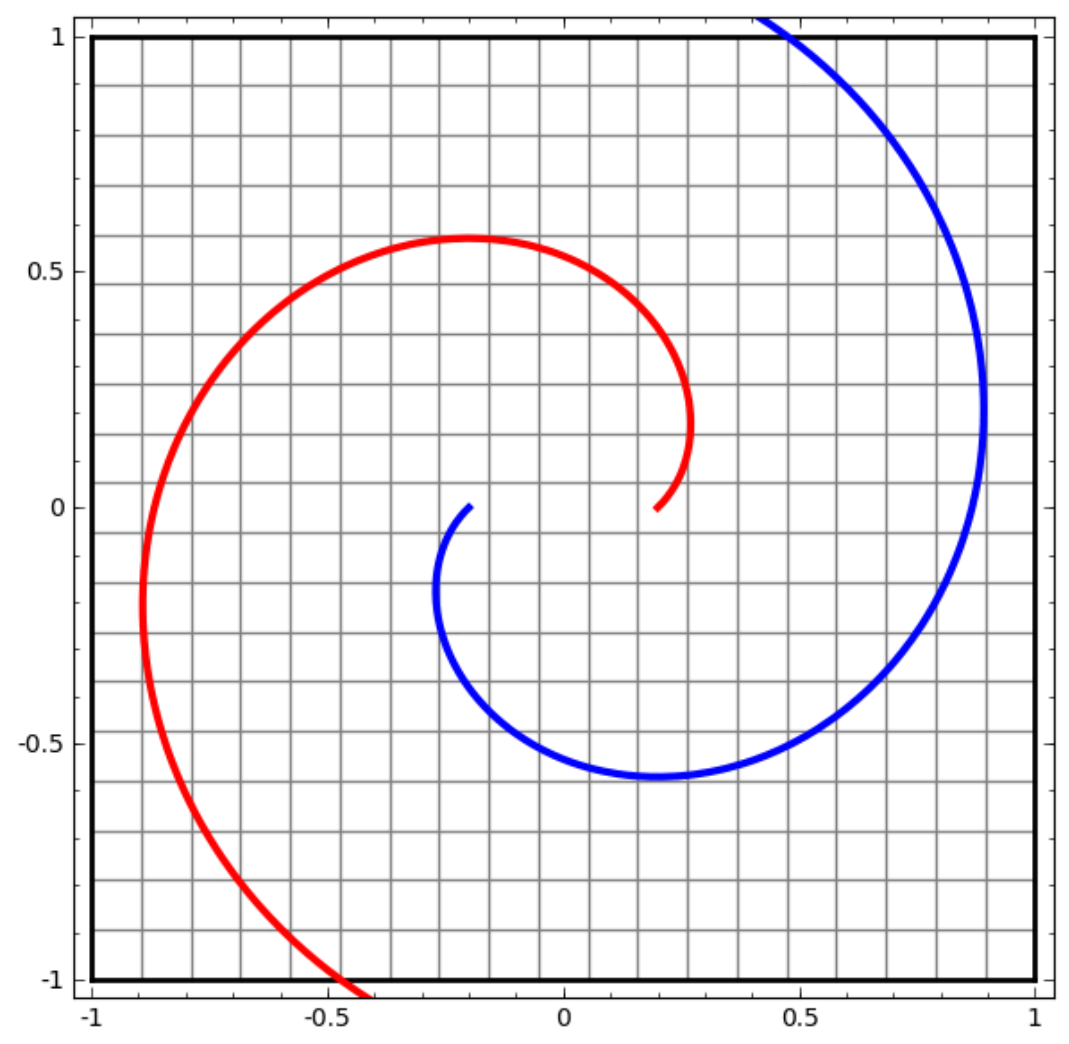
Tanh

- Similar to sigmoid, except over range `(-1, 1)`
- Used if you features are continuous and has -ve values
- Quicker to optimize than sigmoid

Sigmoid / Logistic

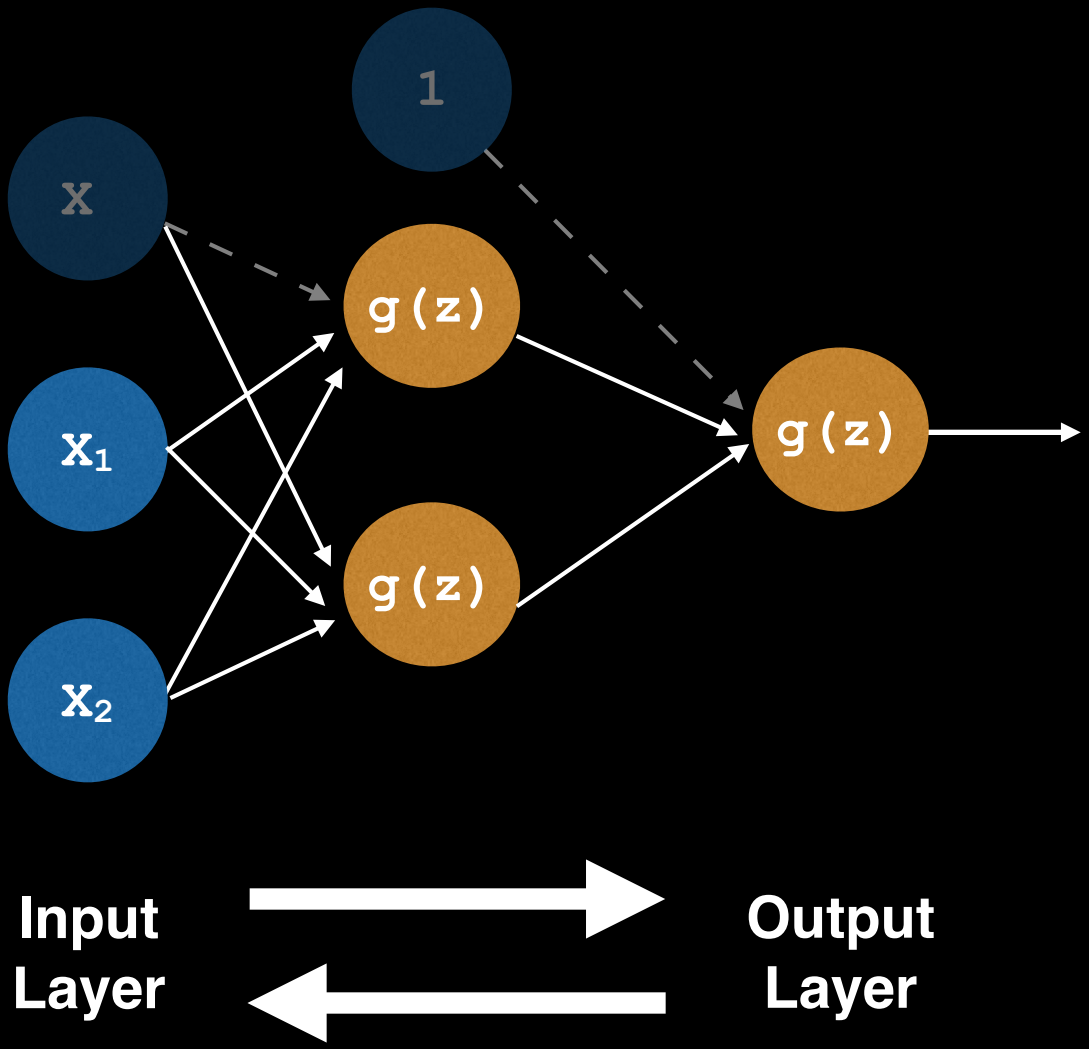
- Over the range `(0, 1)`
- Used if you features are binary

Activation Function Transformation



Almost there …

Forward / Backward Propagation



Next Steps

Convolutional Neural Network

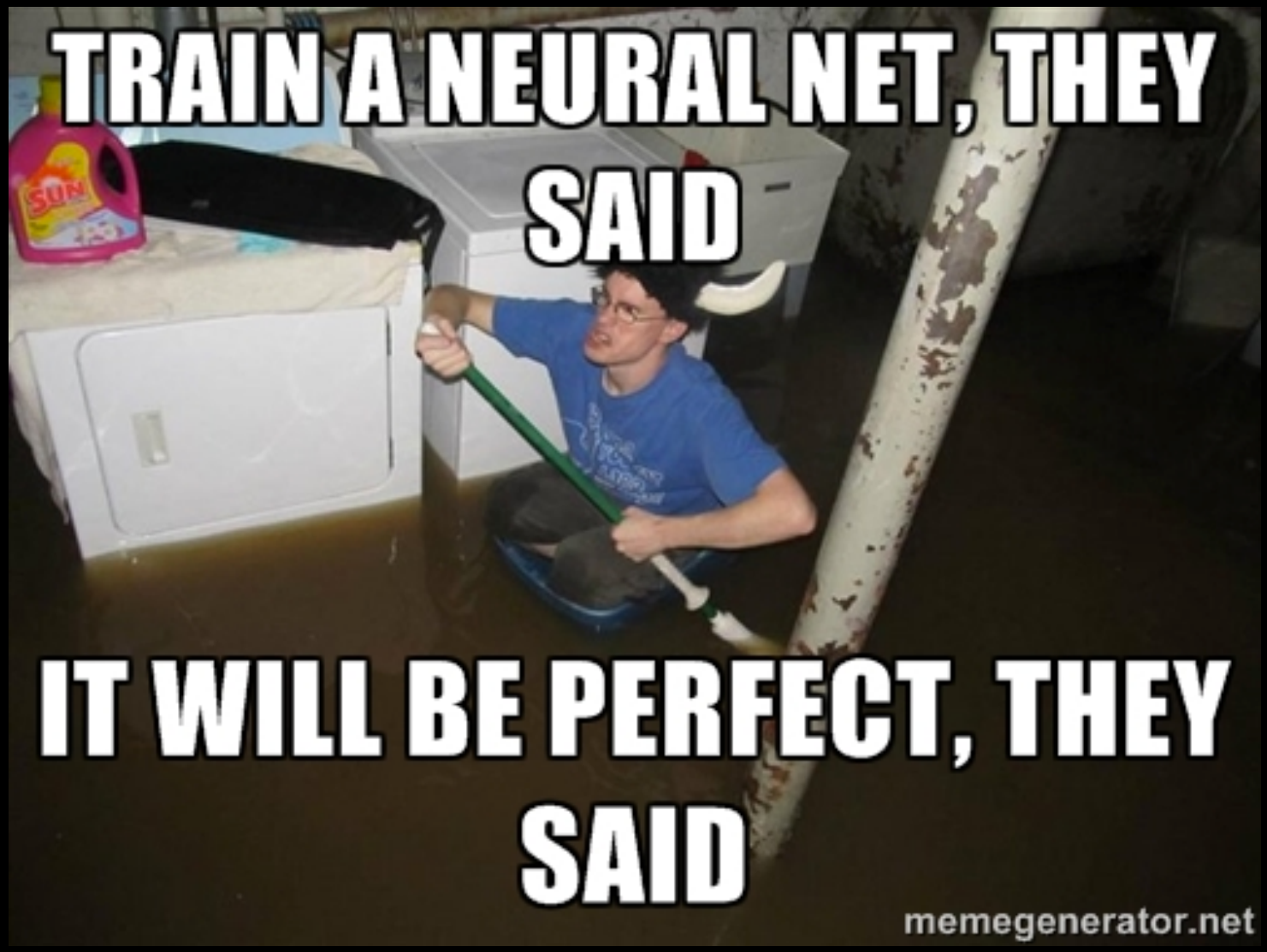
- NN with a convolution layer(s) as the first layer(s)

Recurrent Neural Network

- NN where information moves in a directed cycle

Next Step Resources

- **CS231n: Convolutional Neural Networks for Visual Recognition**
  - ⋆ cs231n.github.io/
- **More Mathematical Treatment**
  - ⋆ deeplearning.net/tutorial/lenet.html


TRAIN A NEURAL NET, THEY SAID
IT WILL BE PERFECT, THEY SAID
memegenerator.net