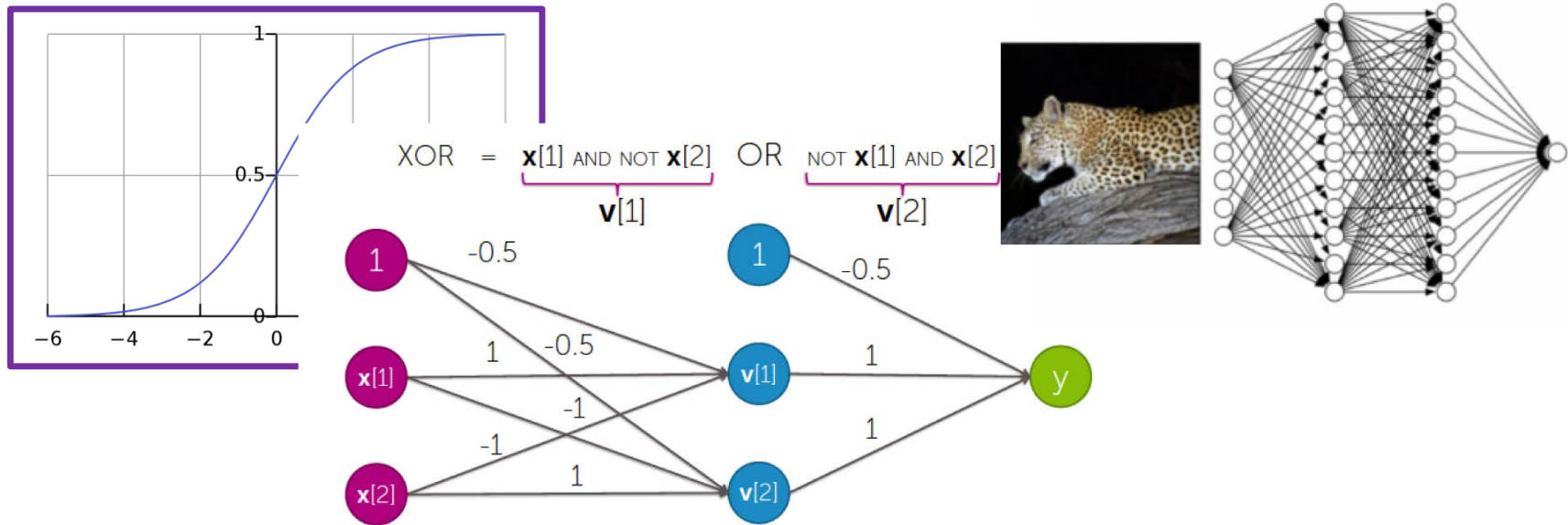


Tufts COMP 135: Introduction to Machine Learning

<https://www.cs.tufts.edu/comp/135/2019s/>

Neural Networks



Many slides attributable to:

Erik Sudderth (UCI), Emily Fox (UW),

Finale Doshi-Velez (Harvard)

James, Witten, Hastie, Tibshirani (ISL/ESL books)

Prof. Mike Hughes

Logistics

- Project 1: Keep going!
- Recitation next Monday
 - Hands-on intro to neural nets
 - With automatic gradient computation

Objectives Today:

Neural Networks Unit 1/2

- How to **learn** feature representations
 - Feed-forward neural nets (MLPs)
 - Universal approximation
 - Activation functions
- The Rise of Deep Learning:
 - Success stories on Images and Language
- Preview: Training via gradient descent
 - Back-propagation = gradient descent + chain rule

What will we learn?

Supervised
Learning

Unsupervised
Learning

Reinforcement
Learning

Training

Data, Label Pairs

$$\{x_n, y_n\}_{n=1}^N$$

Performance
measure

Task

data
 x



label
 y



Prediction

Evaluation

Task: Binary Classification

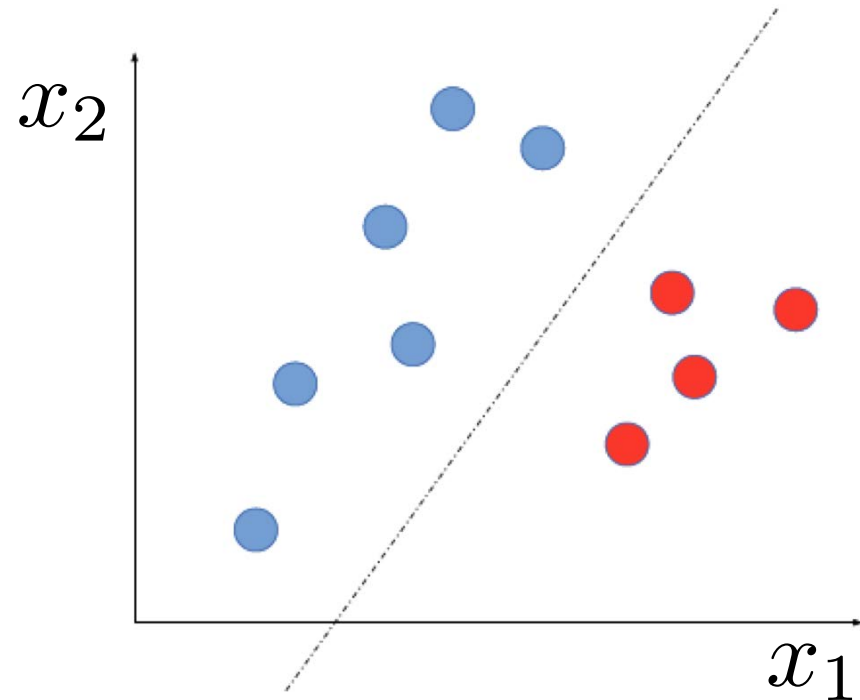
Supervised
Learning

binary
classification

Unsupervised
Learning

Reinforcement
Learning

y is a binary variable
(red or blue)



Example: Hotdog or Not



<https://www.theverge.com/tldr/2017/5/14/15639784/hbo-silicon-valley-not-hotdog-app-download>

Text Sentiment Classification

Sample review:

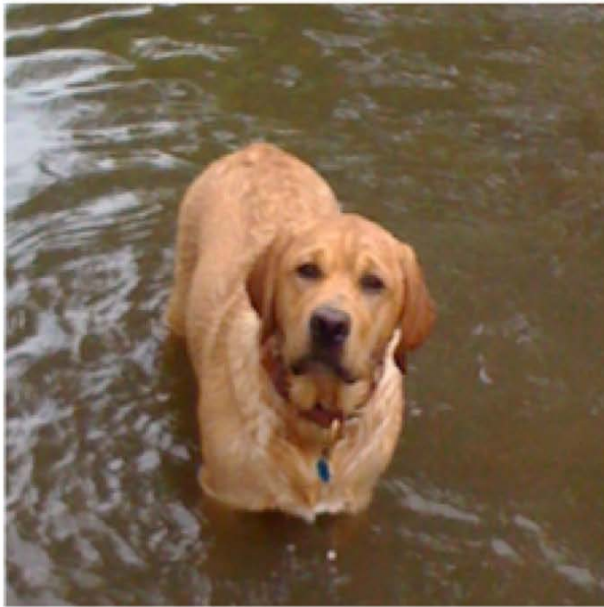
Watching the chefs create incredible edible art made the experience very unique.

My wife tried their ramen and it was pretty forgettable.

All the sushi was delicious!
Easily best sushi in Seattle.



Image Classification

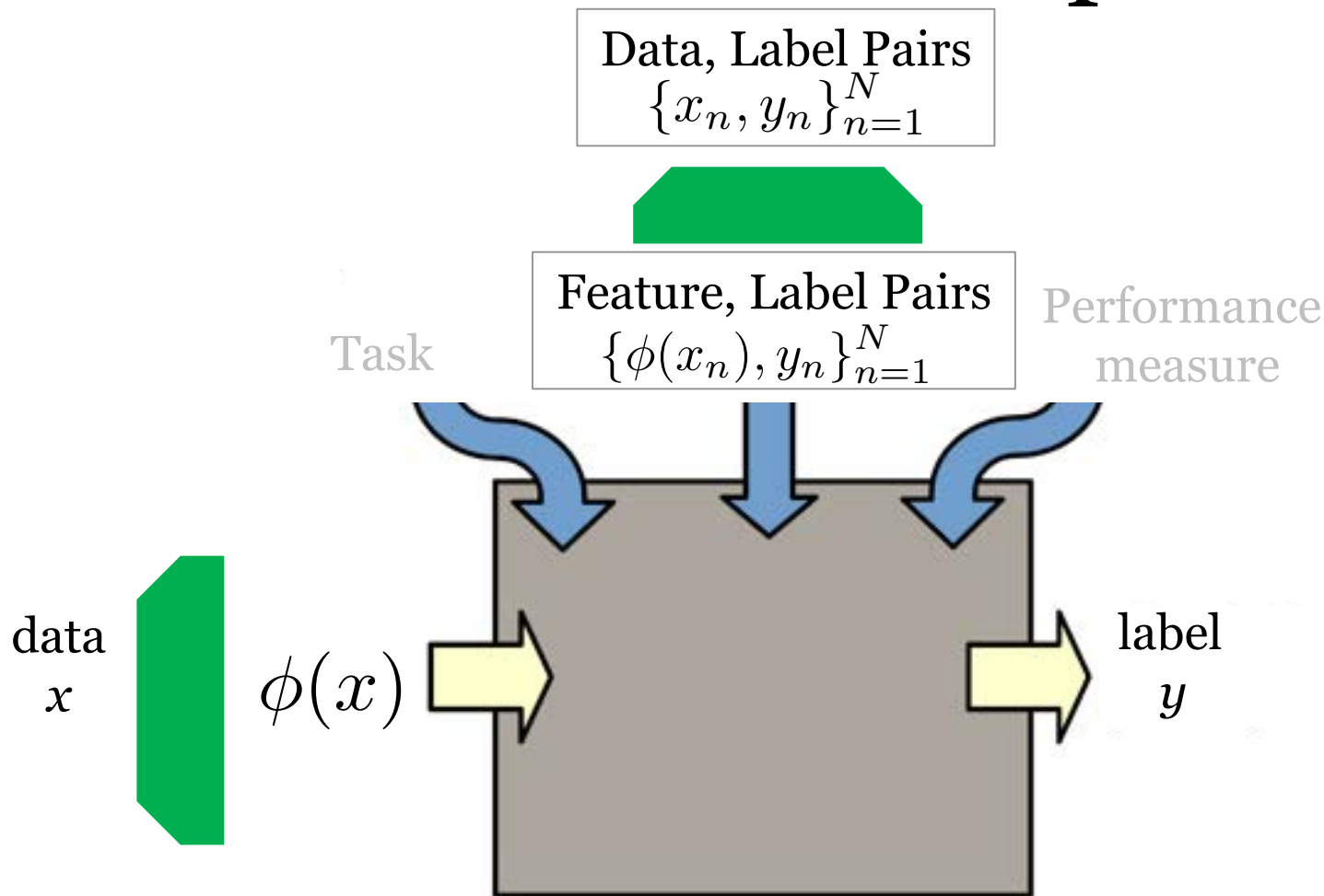


Input: x
Image pixels

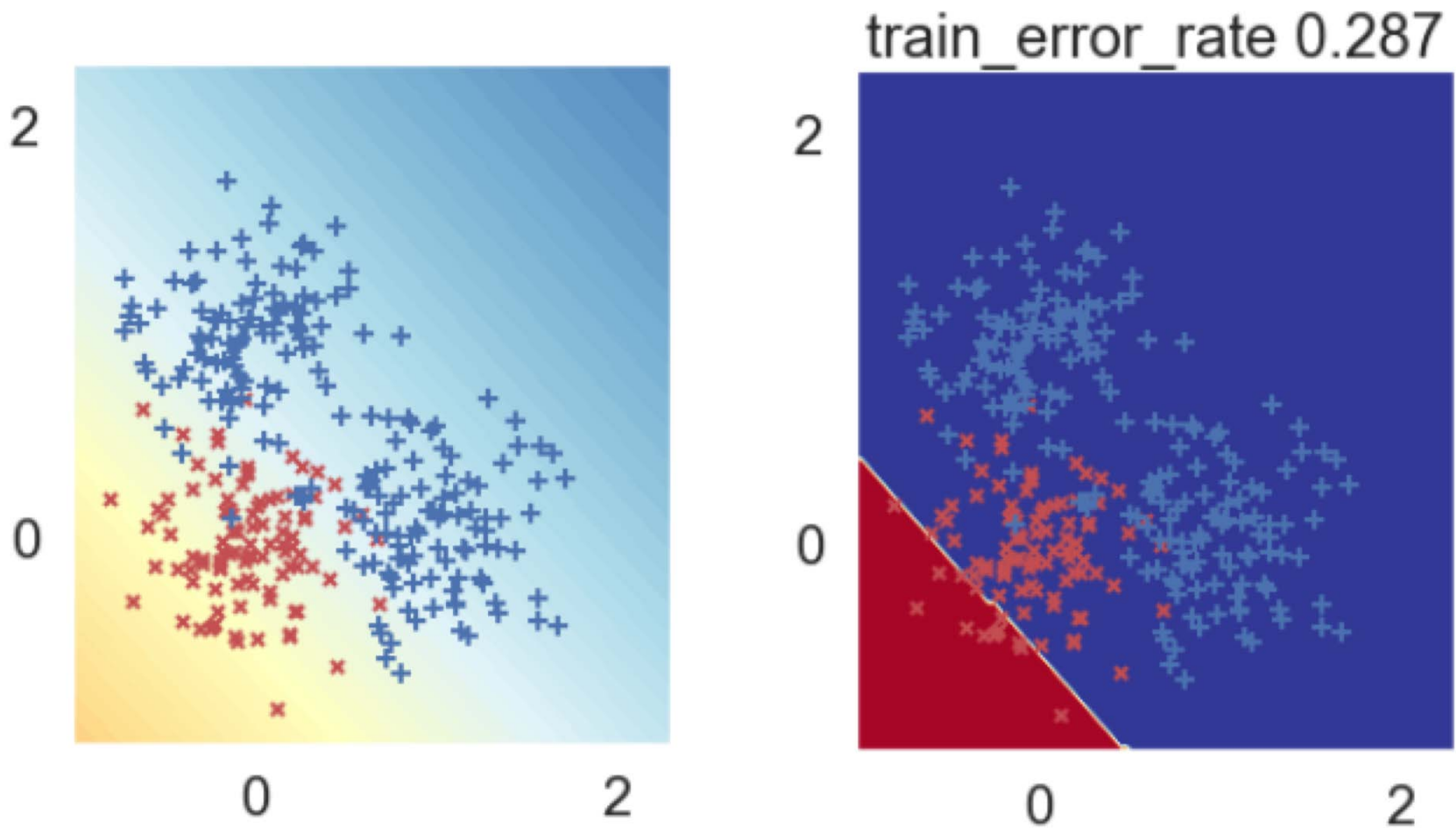


Output: y
Predicted object

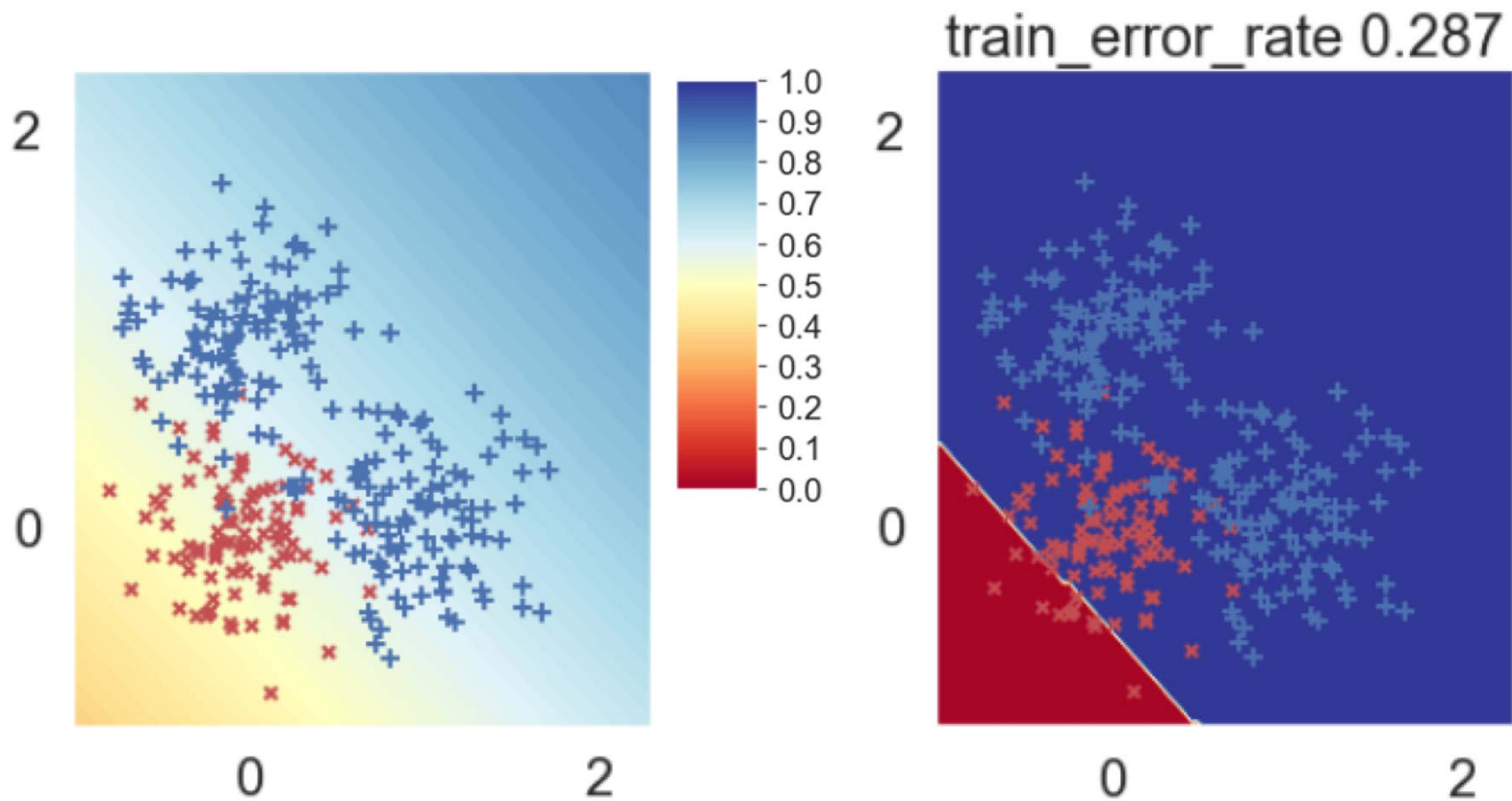
Feature Transform Pipeline



Predicted Probas vs Binary Labels

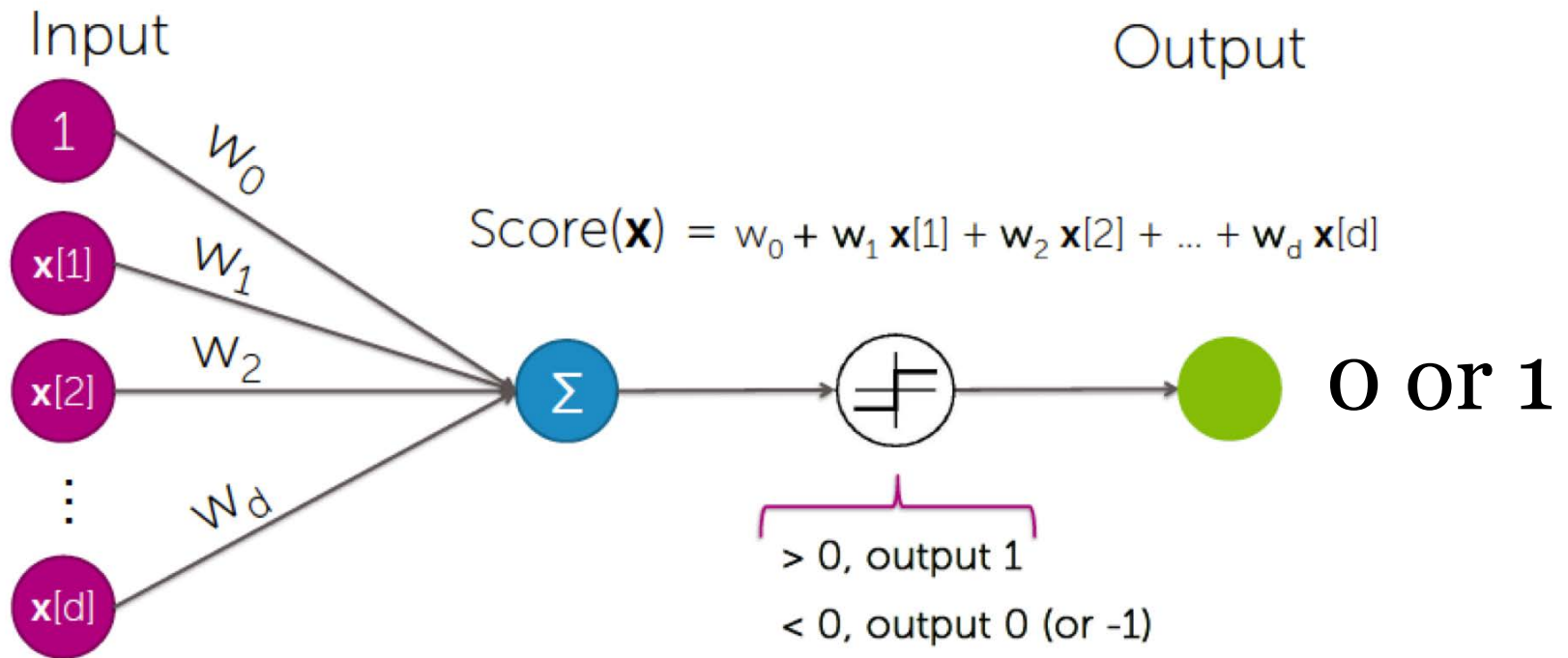


Decision Boundary is **Linear**



$$\{x \in \mathbb{R}^2 : \sigma(w^T \tilde{x}) = 0.5\} \longleftrightarrow \{x \in \mathbb{R}^2 : w^T \tilde{x} = 0\}$$

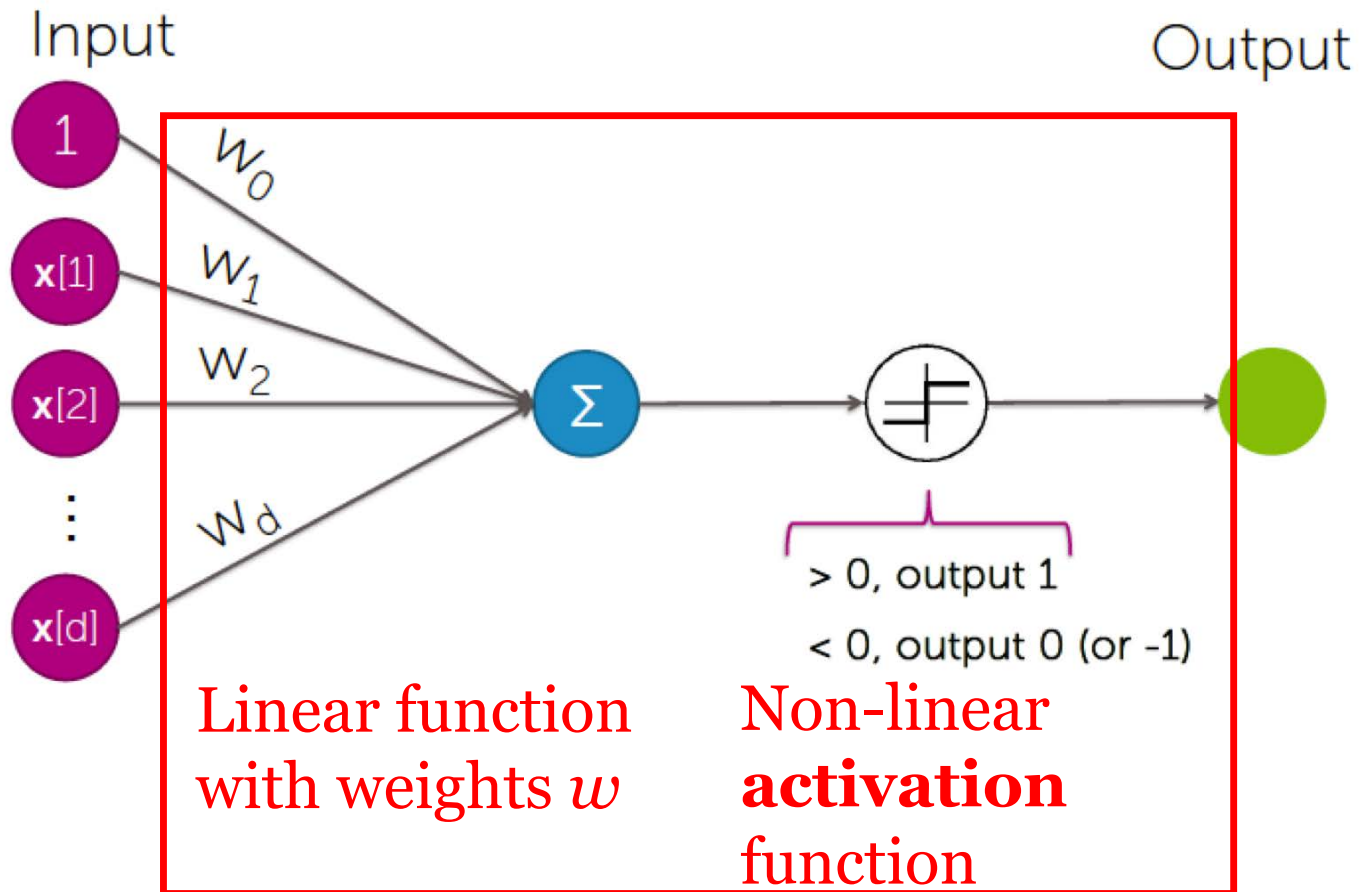
Logistic Regr. Network Diagram



Credit: Emily Fox (UW)

<https://courses.cs.washington.edu/courses/cse416/18sp/slides/>

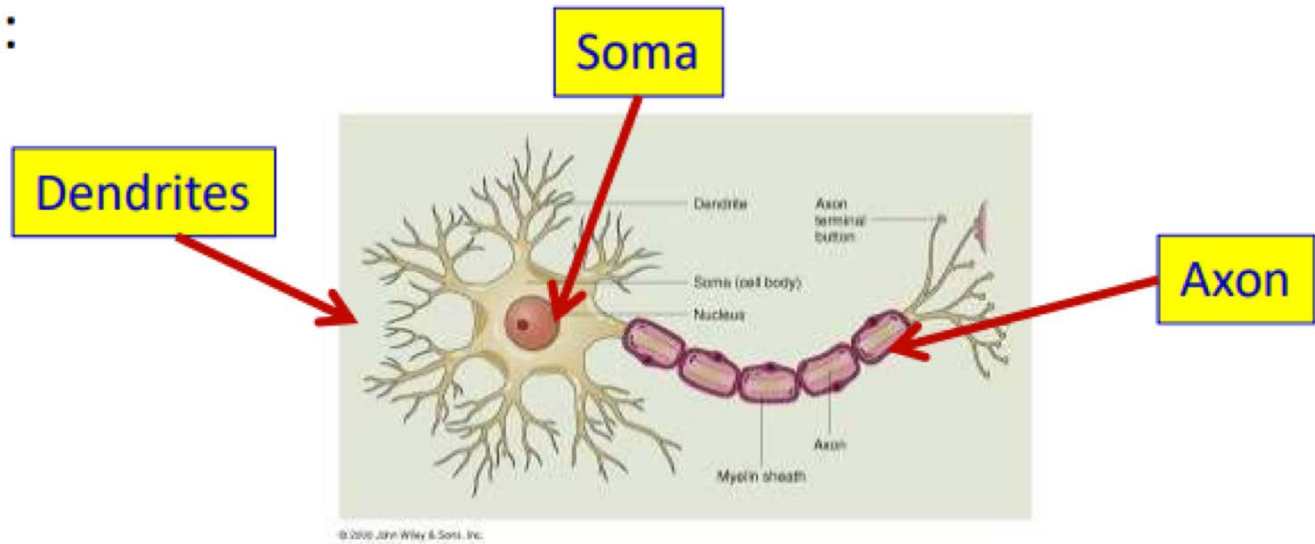
A “Neuron” or “Perceptron” Unit



Credit: Emily Fox (UW)

“Inspired” by brain biology

A neuron:



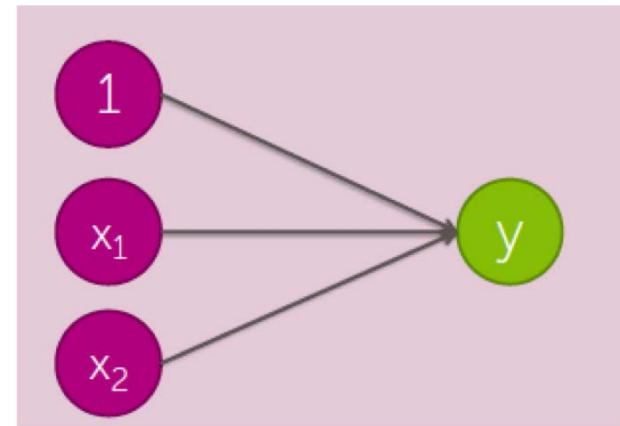
Signals come in through the dendrites into the Soma

A signal goes out via the axon to other neurons

- Only one axon per neuron

Slide Credit: Bhiksha Raj (CMU)

Challenge: Find w for these functions



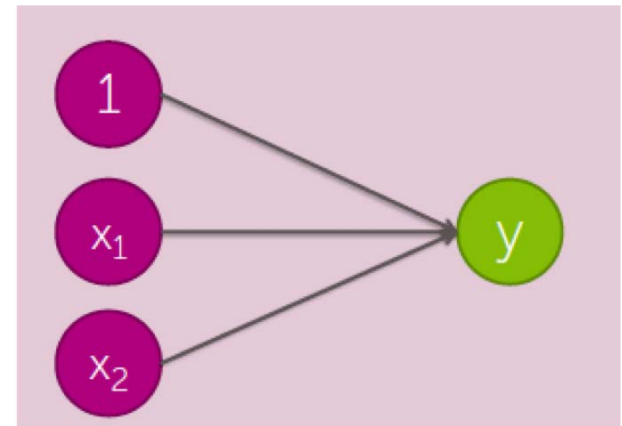
x_1 OR x_2

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

x_1 AND x_2

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Challenge: Find w for these functions



x_1 OR x_2

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

$$-0.5 + x[1] + x[2]$$

x_1 AND x_2

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

$$-1.5 + x[1] + x[2]$$

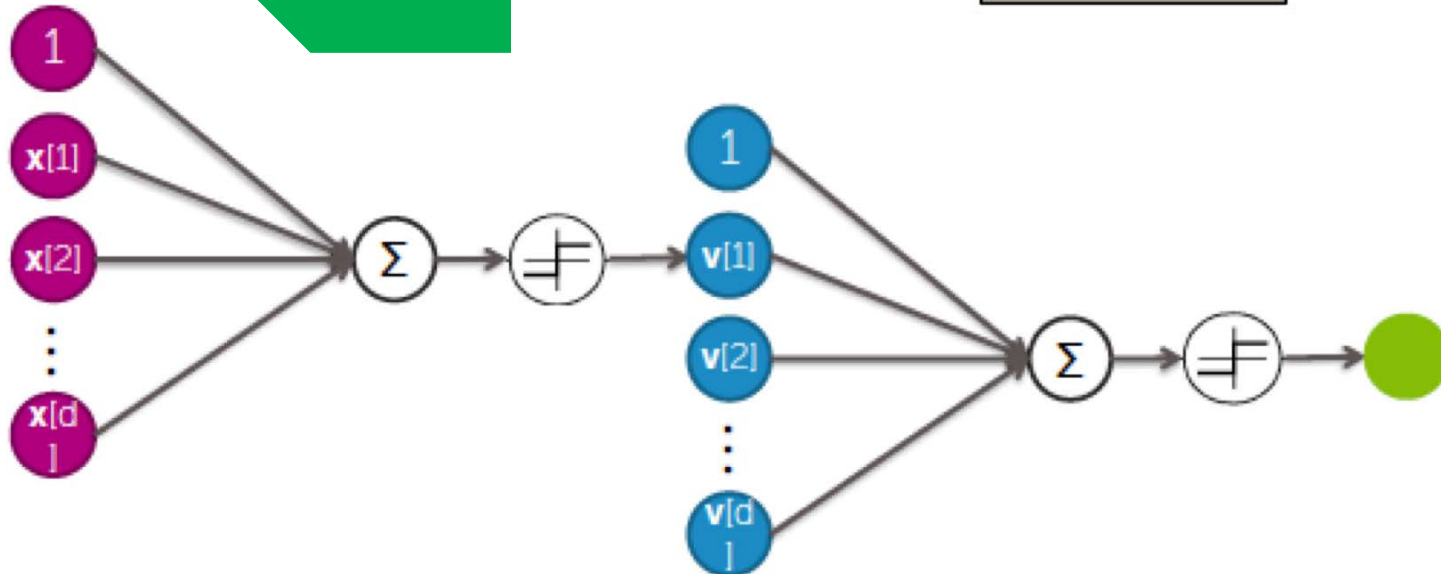
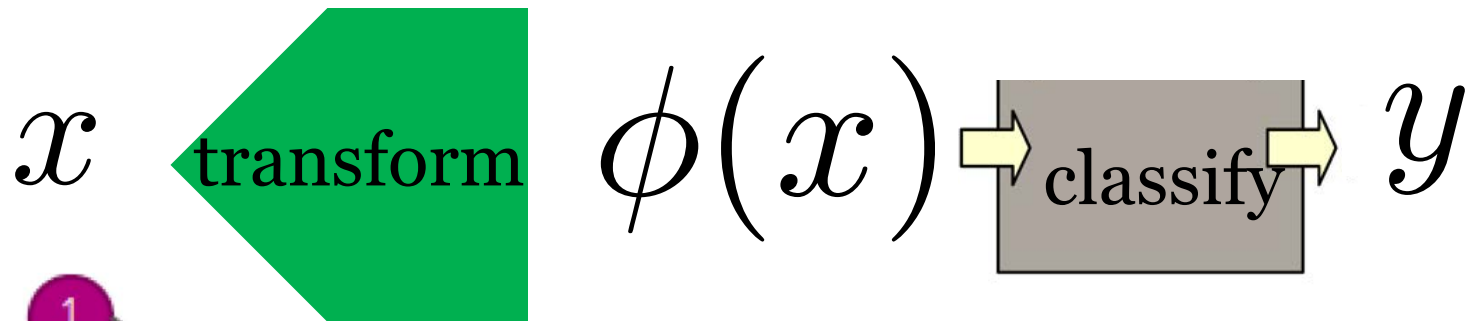
What we can't do with linear decision boundary classifiers



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

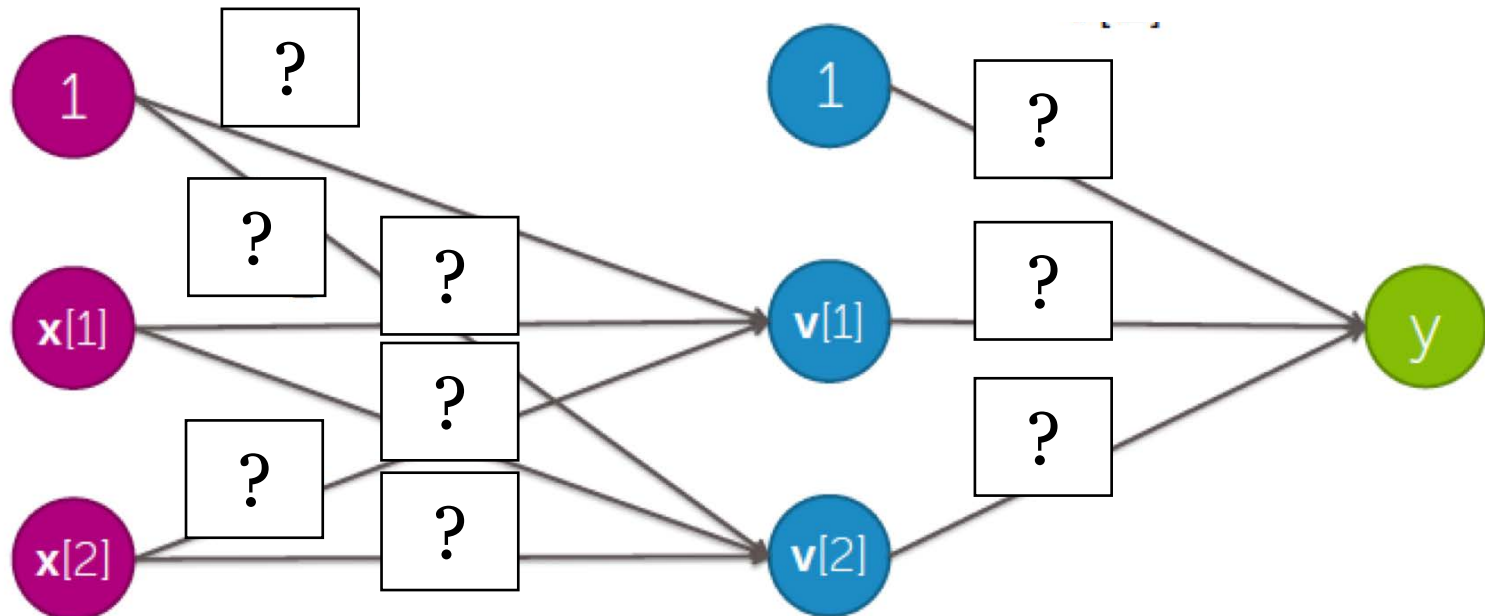
$$\text{XOR} = \mathbf{x}[1] \text{ AND NOT } \mathbf{x}[2] \text{ OR NOT } \mathbf{x}[1] \text{ AND } \mathbf{x}[2]$$

Idea: Compose Neurons together!

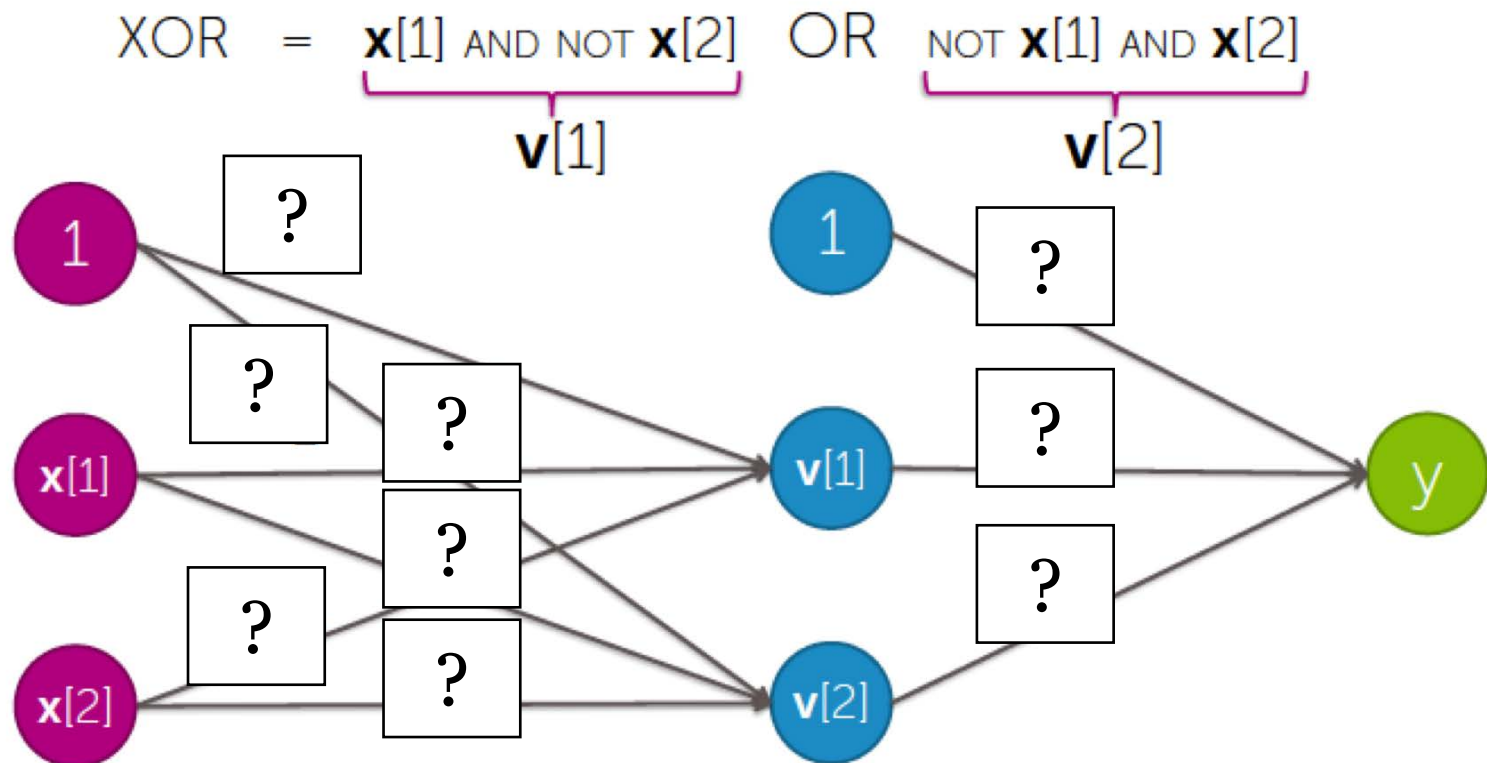


Can you find w to solve XOR?

$$\text{XOR} = \boxed{?} \text{ AND/OR } \boxed{?}$$

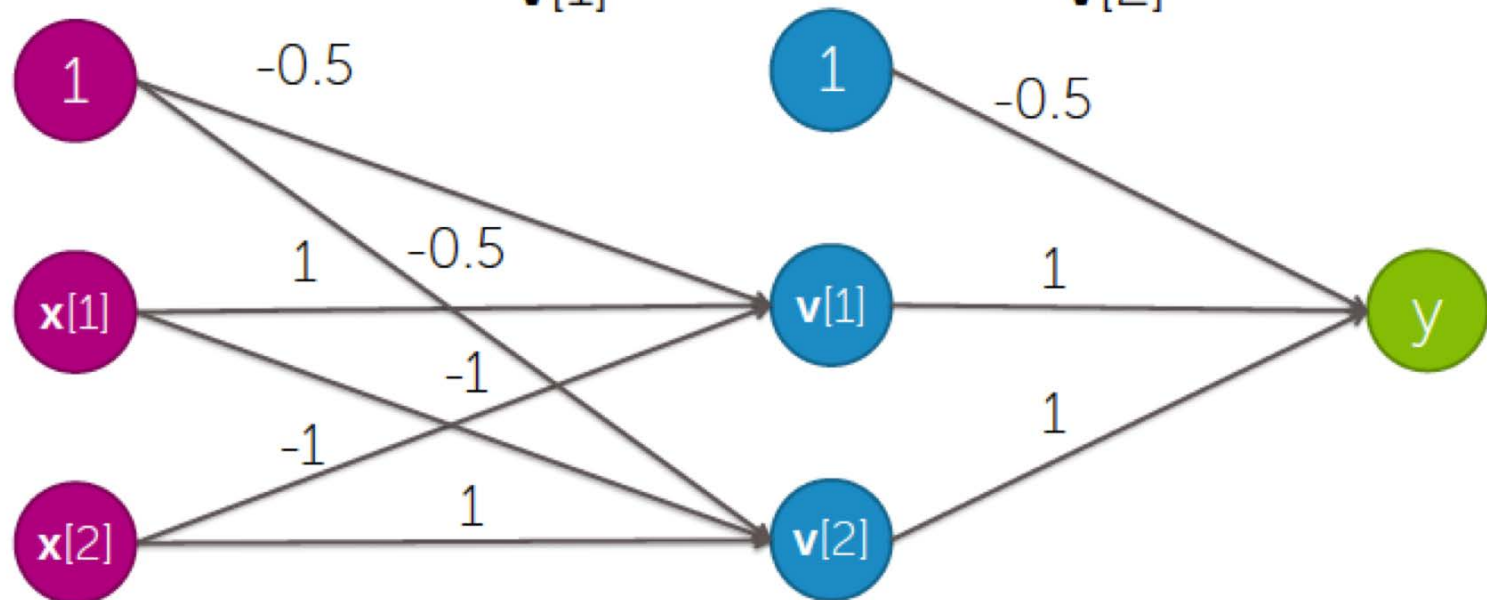


Can you find w to solve XOR?

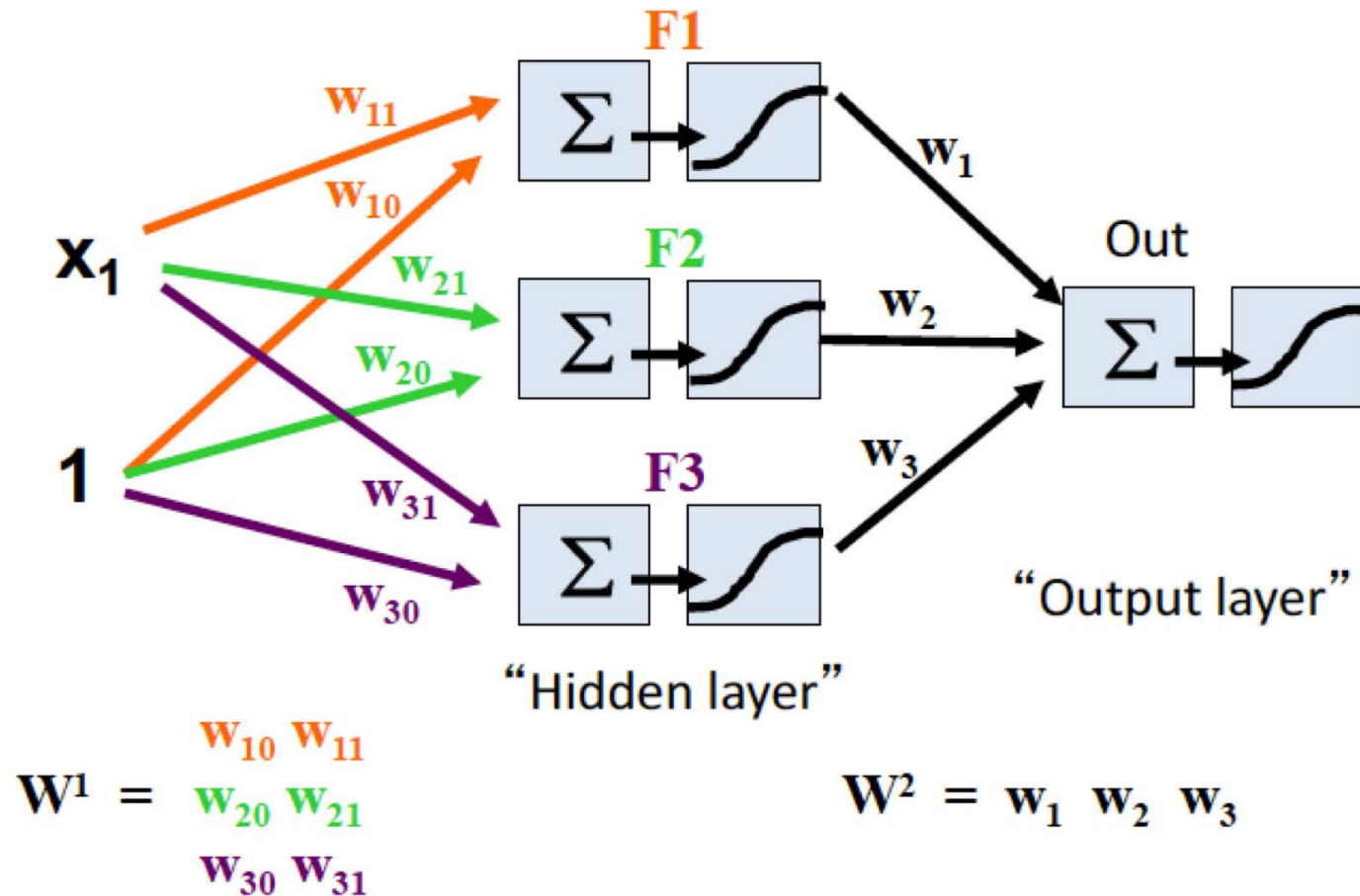


Can you find w to solve XOR?

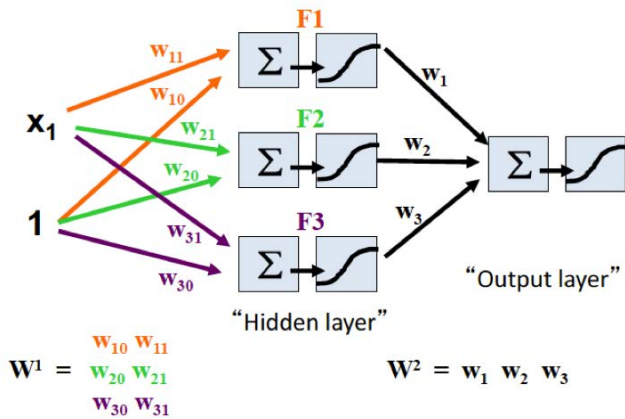
$$\text{XOR} = \underbrace{\mathbf{x}[1] \text{ AND NOT } \mathbf{x}[2]}_{\mathbf{v}[1]} \text{ OR } \underbrace{\text{NOT } \mathbf{x}[1] \text{ AND } \mathbf{x}[2]}_{\mathbf{v}[2]}$$



1D Input + 3 hidden units



1D Input + 3 hidden units

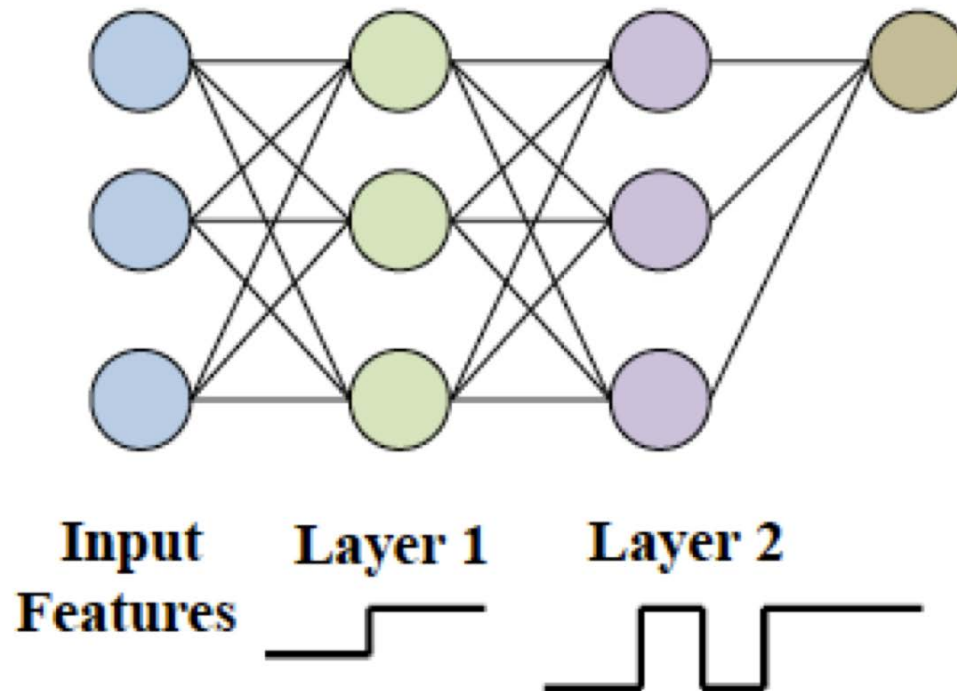


Example functions
(before final threshold)



Intuition: Piece-wise step function
Partitioning input space into regions

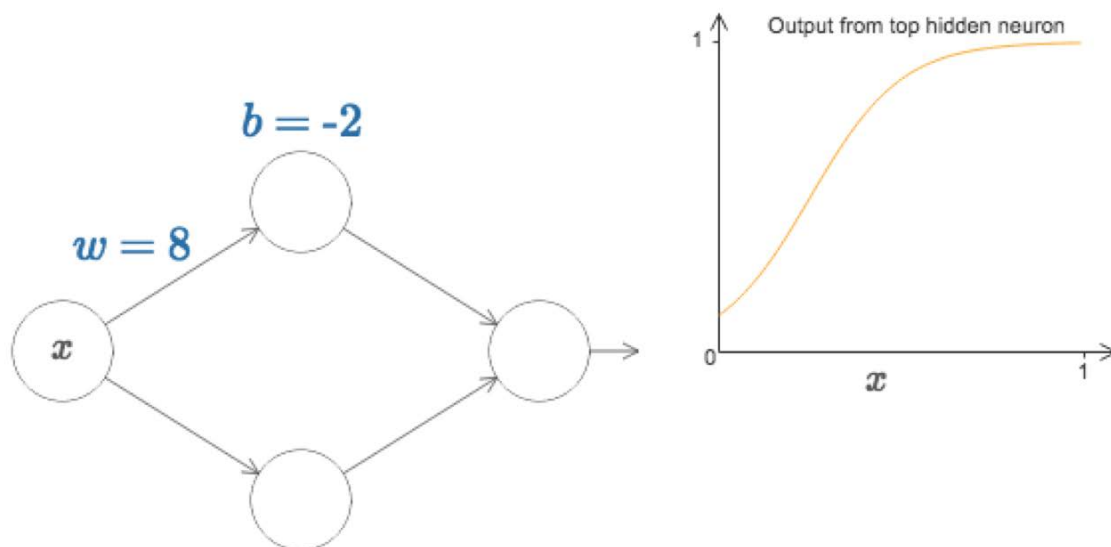
More layers = more partitions



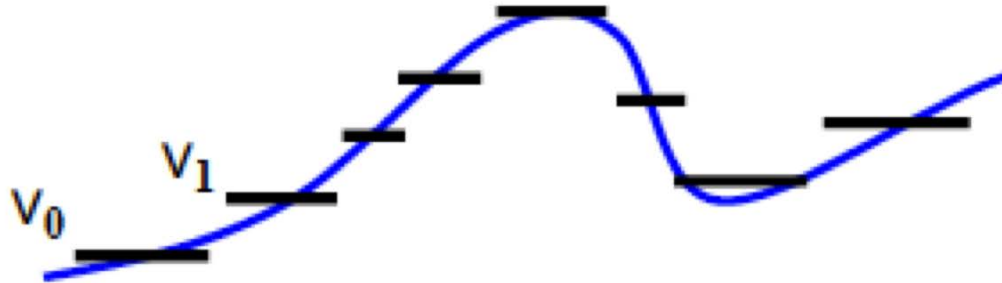
Interactive Demo

[http://neuralnetworksanddeeplearning.com/chap4.html](http://neuralnetworksanddeeplearning.com/chap4.html#universality)
#universality with one input and one output

Credit: Michael Nielson



MLPs can approximate any functions with enough hidden units!



Universal approximation theorem

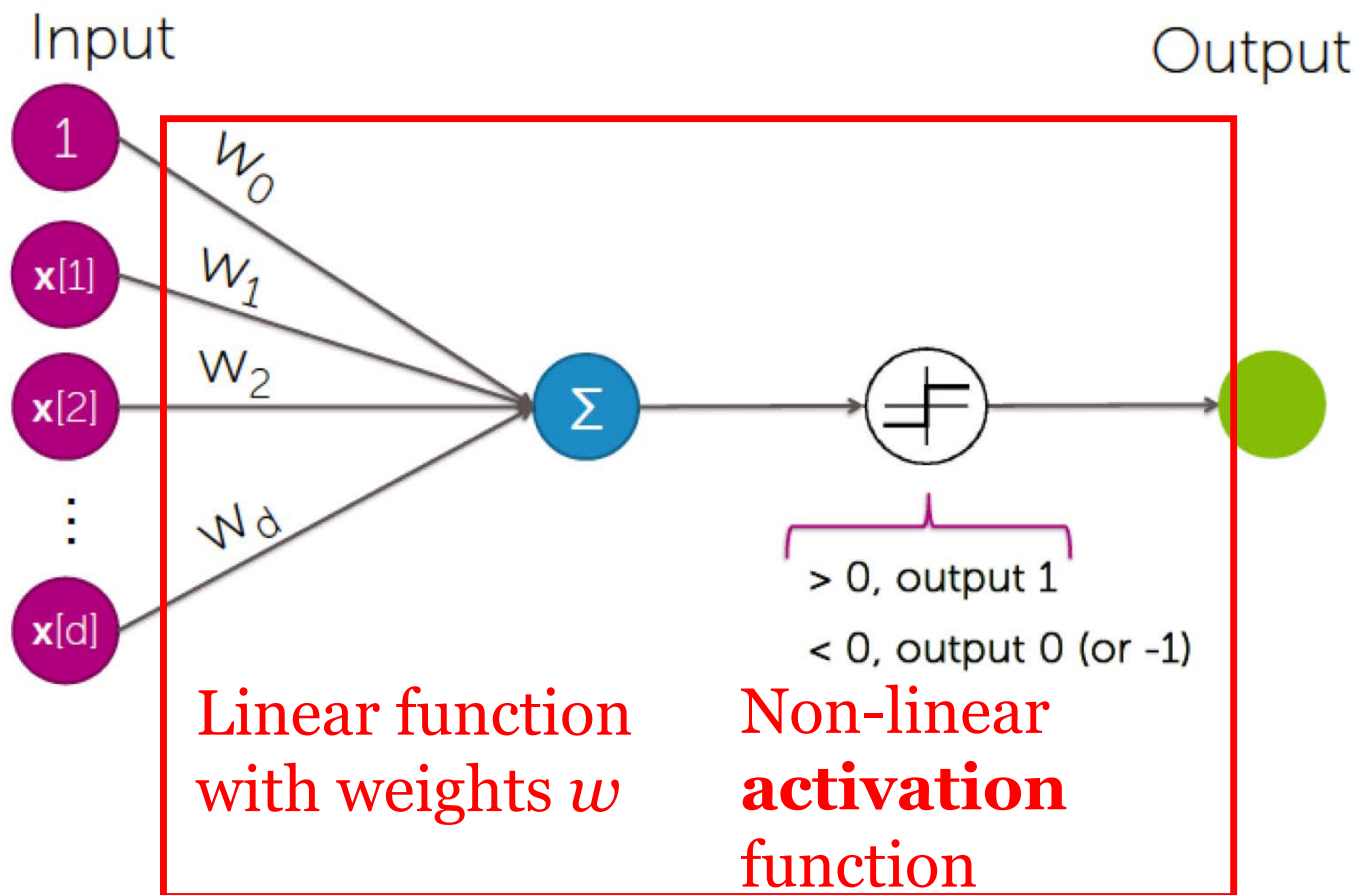
From Wikipedia, the free encyclopedia

In the [mathematical](#) theory of [artificial neural networks](#), the **universal approximation theorem** states^[1] that a [feed-forward](#) network with a single hidden layer containing a finite number of [neurons](#) can approximate [continuous functions](#) on [compact subsets](#) of \mathbf{R}^n , under mild assumptions on the activation function. The theorem thus states that simple neural networks can *represent* a wide variety of interesting functions when given appropriate parameters; however, it does not touch upon the algorithmic [learnability](#) of those parameters.

One of the first versions of the [theorem](#) was proved by [George Cybenko](#) in 1989 for [sigmoid](#) activation functions.^[2]

Neuron Design

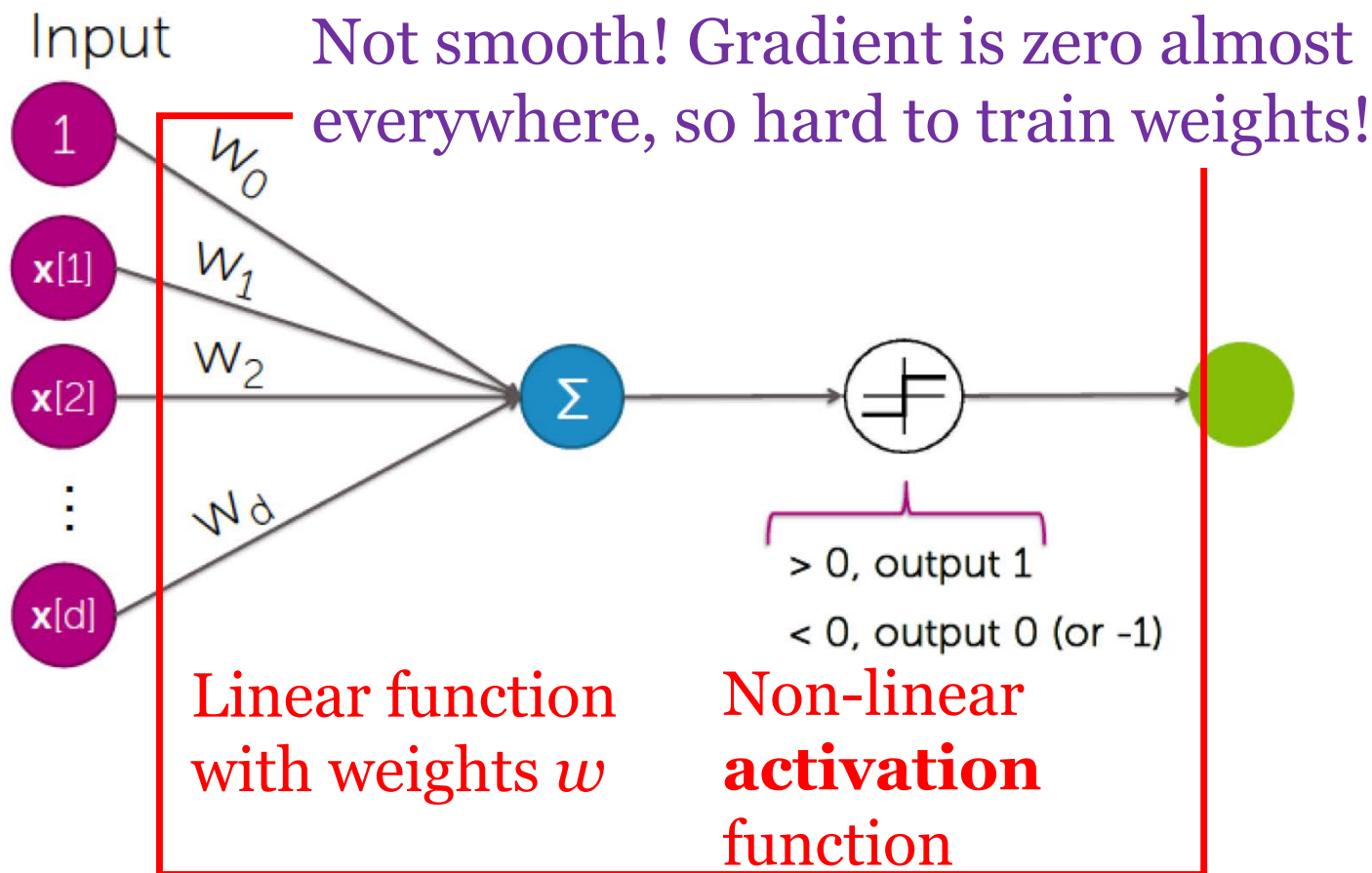
What's wrong with hard step activation function?



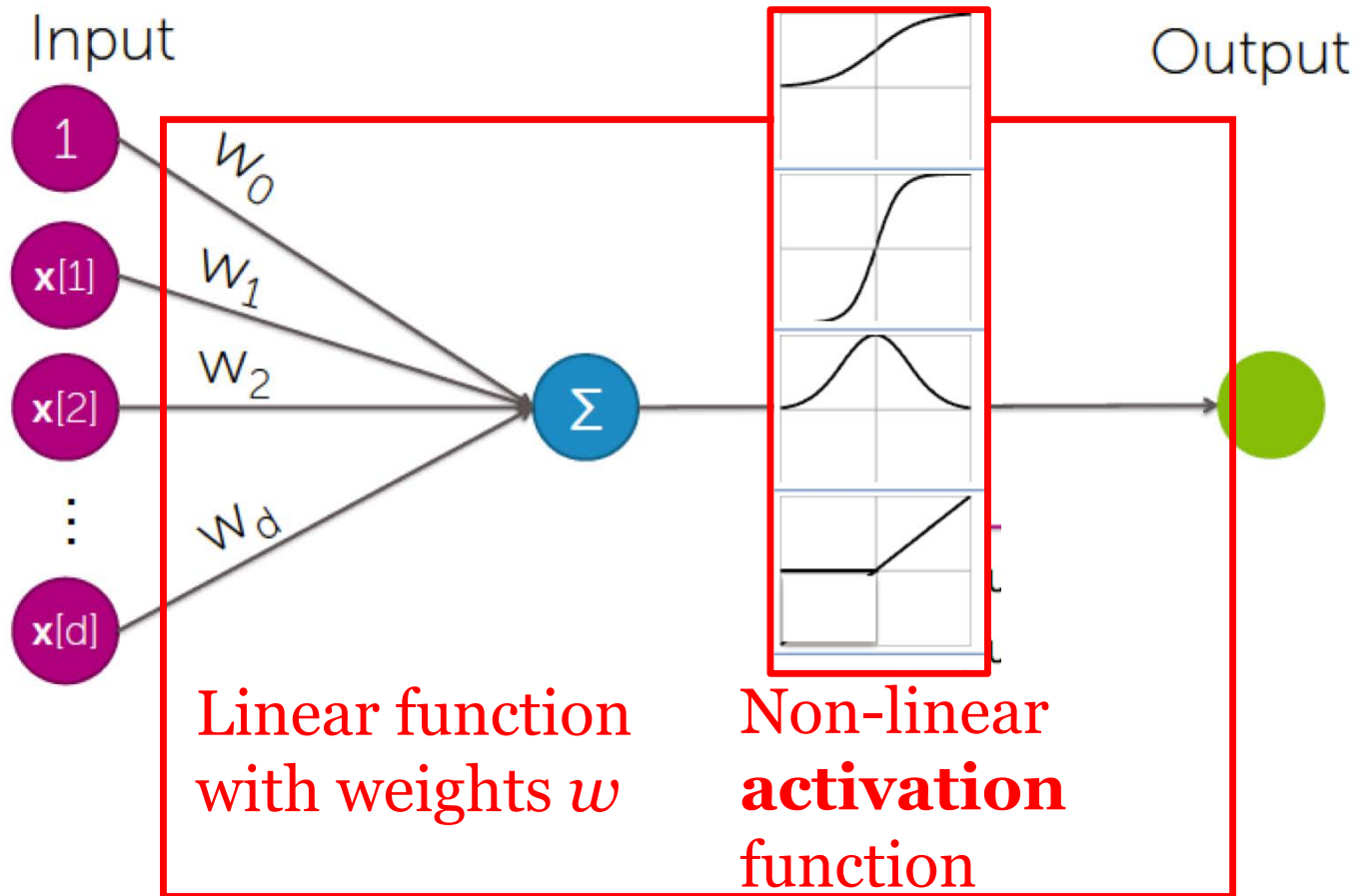
Credit: Emily Fox (UW)

Neuron Design


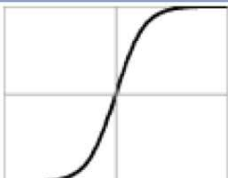
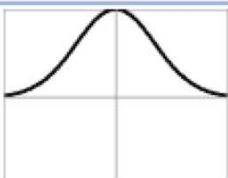

What's wrong with hard step activation function?



Which Activation Function?



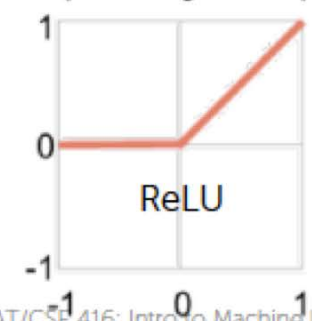
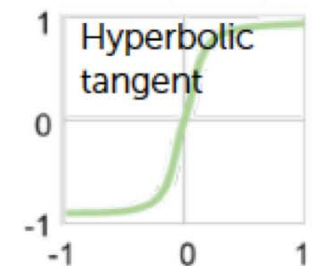
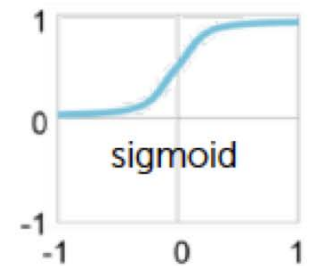
Activation Functions

Logistic	$\sigma(z) = \frac{1}{1 + \exp(-z)}$		$\frac{\partial \sigma}{\partial z}(z) = \sigma(z)(1 - \sigma(z))$
Hyperbolic Tangent	$\sigma(z) = \frac{1 - \exp(-2z)}{1 + \exp(-2z)}$		$\frac{\partial \sigma}{\partial z}(z) = 1 - (\sigma(z))^2$
Gaussian	$\sigma(z) = \exp(-z^2/2)$		$\frac{\partial \sigma}{\partial z}(z) = -z\sigma(z)$
ReLU (rectified linear)	$\sigma(z) = \max(0, z)$		$\frac{\partial \sigma}{\partial z}(z) = \mathbb{1}[z > 0]$
Linear	$\sigma(z) = z$		and many others...

Activation Function Advice

Credit: Emily Fox (UW)

- **Sigmoid**
 - Historically popular, but (mostly) fallen out of favor
 - Neuron's activation saturates (weights get very large \rightarrow gradients get small)
 - Not zero-centered \rightarrow other issues in the gradient steps
 - When put on the output layer, called "softmax" because interpreted as class probability (soft assignment)
- **Hyperbolic tangent** $g(x) = \tanh(x)$
 - Saturates like sigmoid unit, but zero-centered
- **Rectified linear unit (ReLU)** $g(x) = x^+ = \max(0, x)$
 - Most popular choice these days
 - Fragile during training and neurons can "die off"... be careful about learning rates
 - "Noisy" or "leaky" variants
- **Softplus** $g(x) = \log(1 + \exp(x))$
 - Smooth approximation to rectifier activation

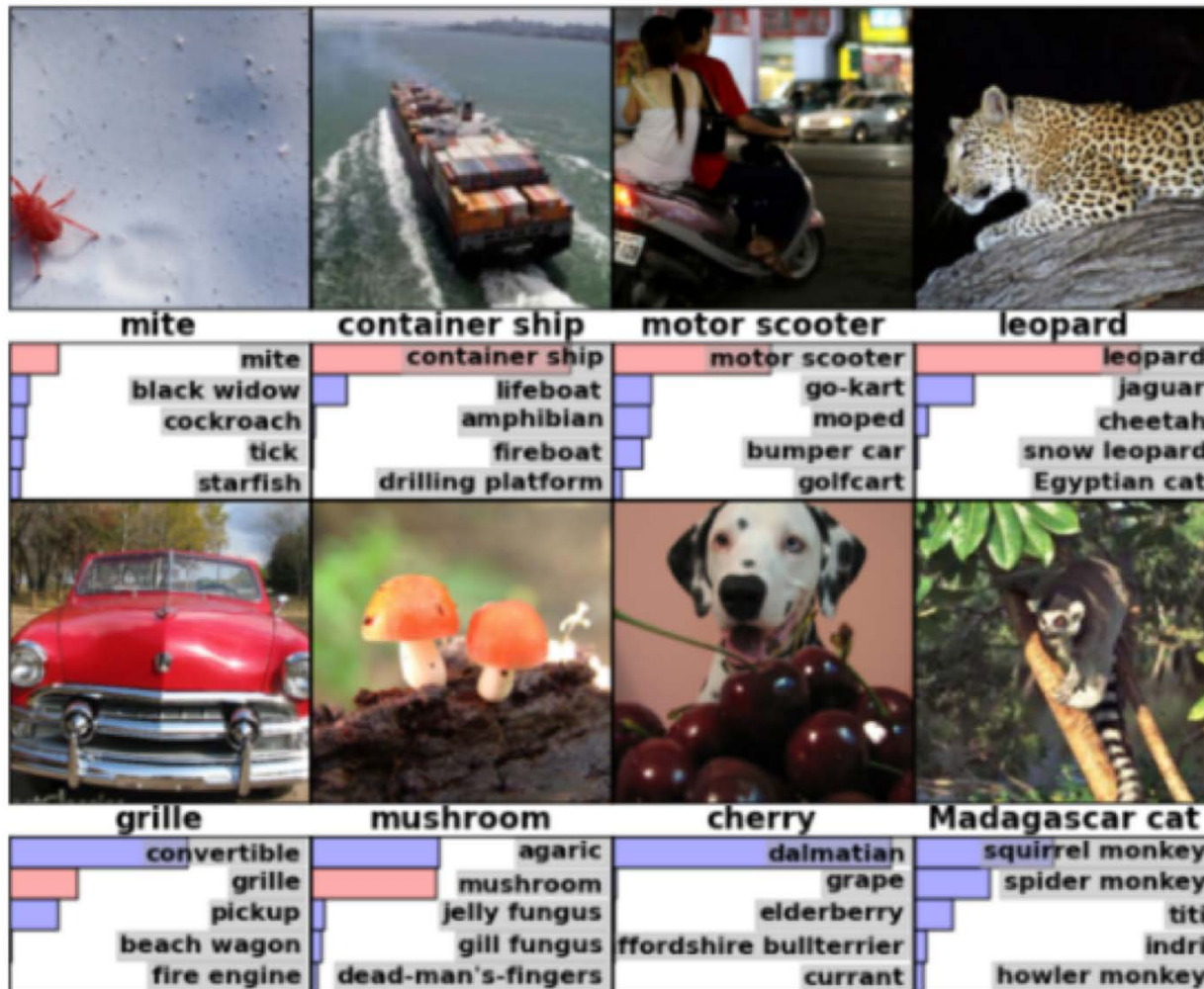


©2018 Emily Fox

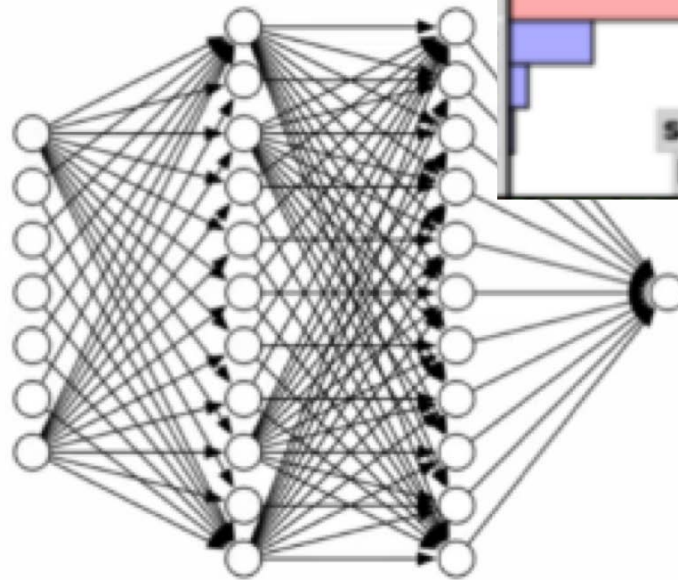
STAT/CSE 416: Intro to Machine Learning

Exciting Applications: Computer Vision

Object Recognition from Images



Deep Neural Networks for Object Recognition



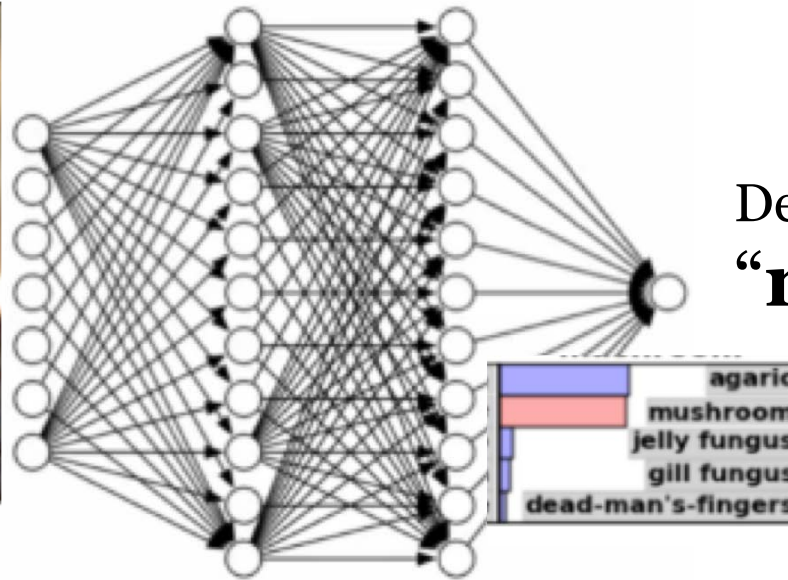
Scores for each
possible object category



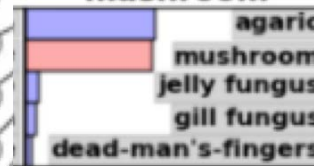
Decision:
“leopard”

Deep Neural Networks for Object Recognition

Scores for each
possible object category



Decision:
“mushroom”



Each Layer Extracts “Higher Level” Features



Layer 1



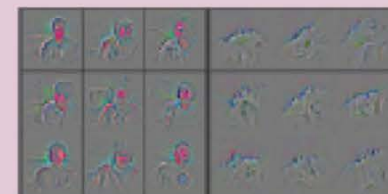
Layer 2



Layer 3



Example
detectors
learned



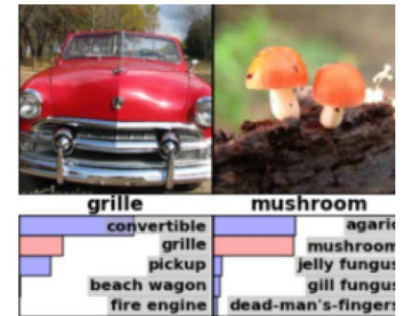
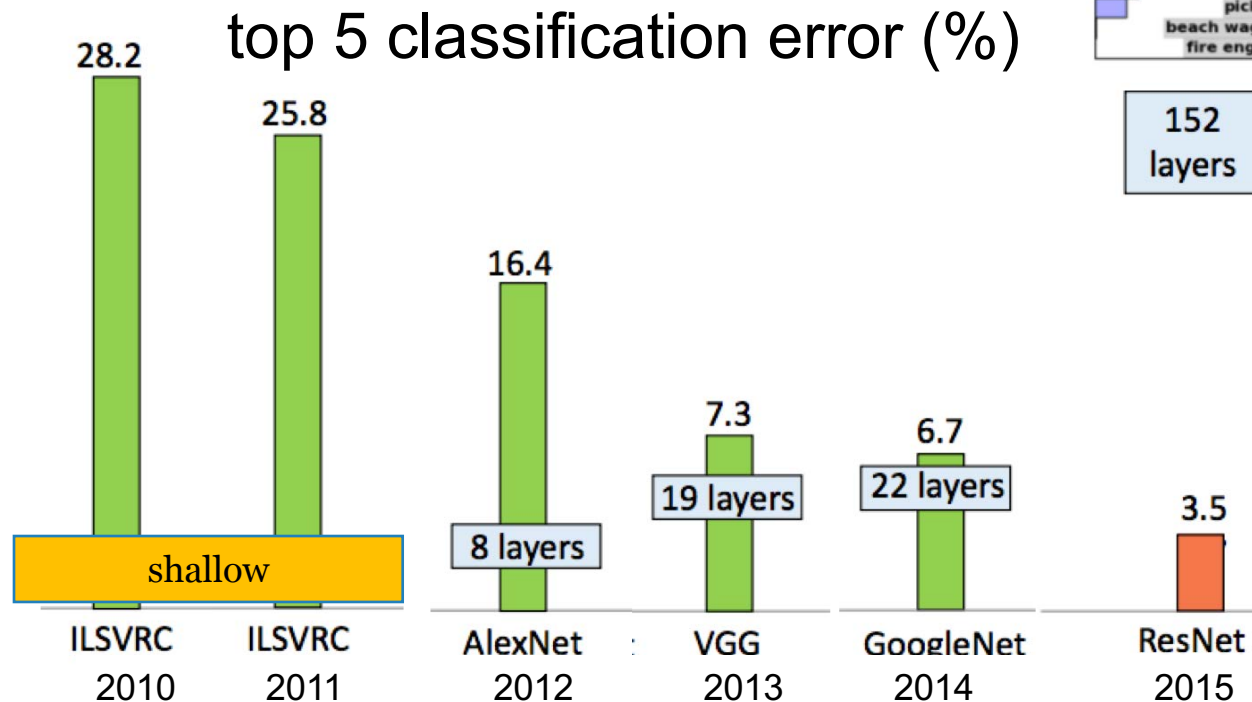
Example
interest
points
detected



More layers = less error!

ImageNet challenge

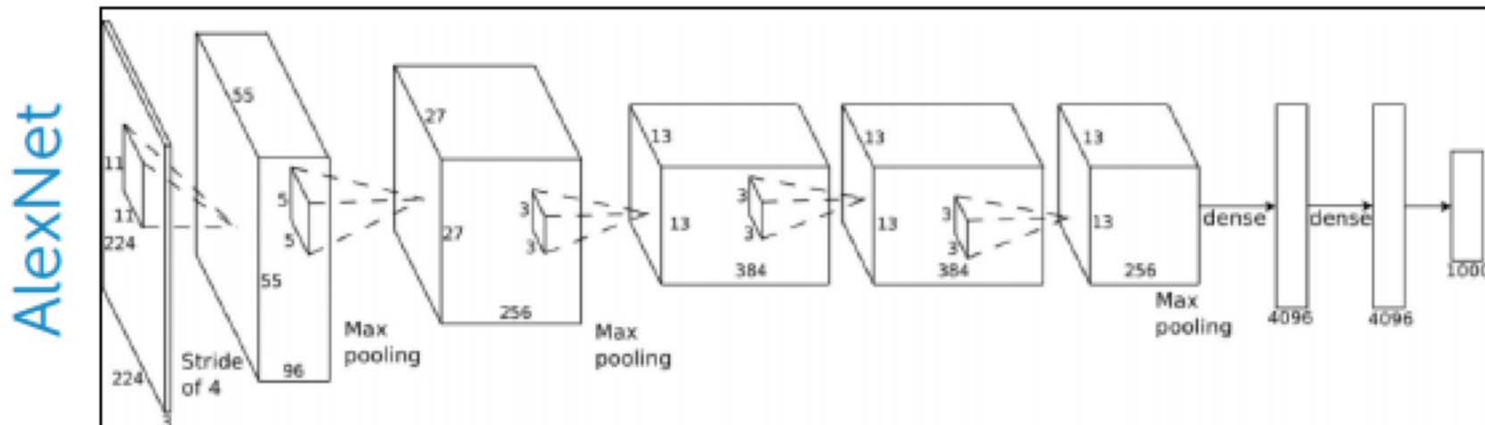
1000 categories, 1.2 million images in training set



Credit: KDD Tutorial by Sun, Xiao, & Choi: <http://dl4health.org/>
Figure idea originally from He et. al., CVPR 2016

2012 ImageNet Challenge Winner

8 layers, 60M parameters [Krizhevsky et al. '12]



Achieving these amazing results required:

- New learning algorithms
- GPU implementation

State of the art Results

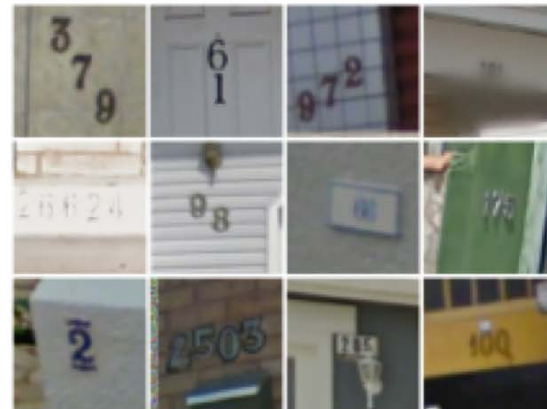
German traffic sign
recognition benchmark

- 99.5% accuracy (IDSIA team)

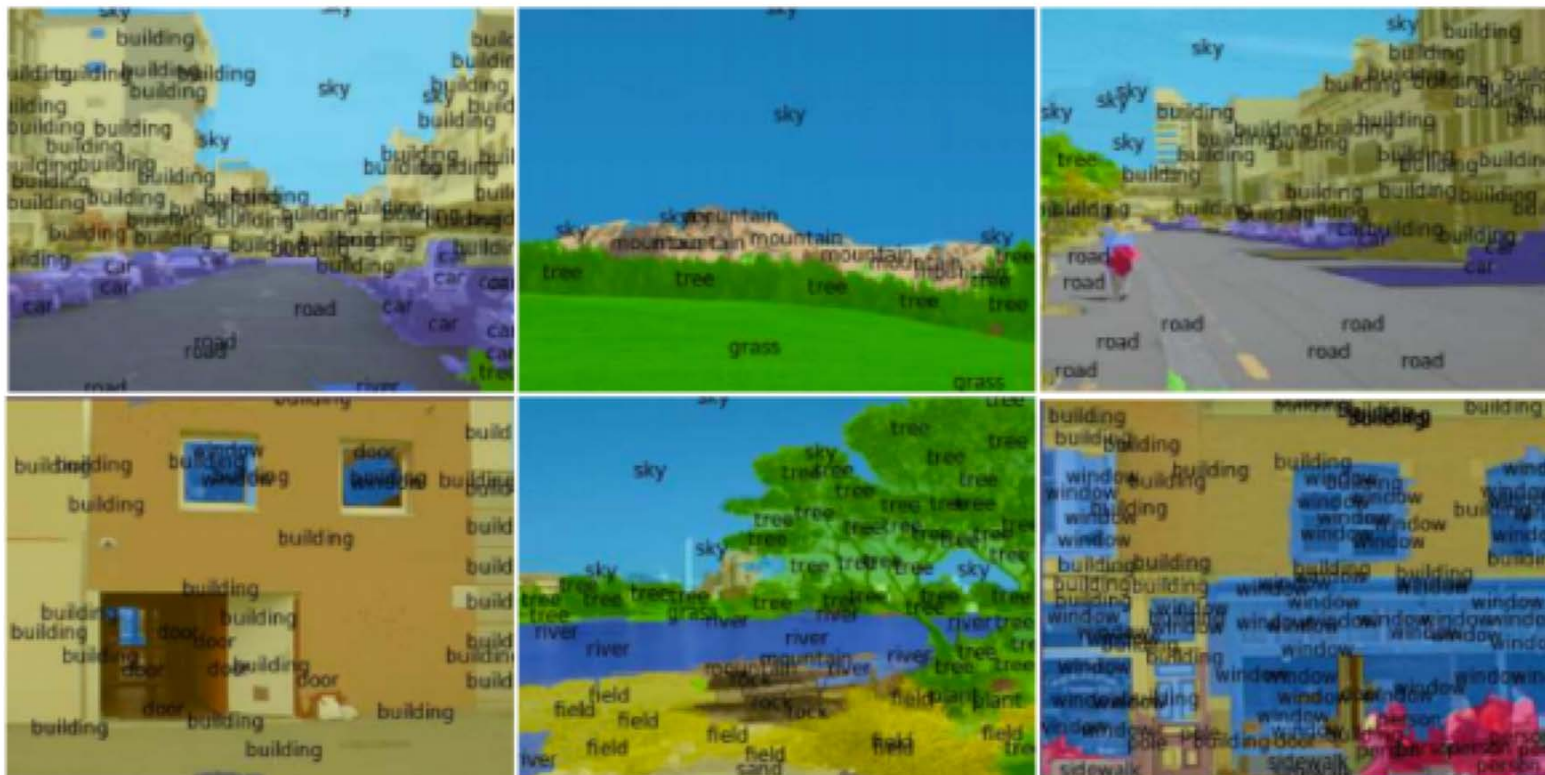


House number recognition

- 97.8% accuracy per character
[Goodfellow et al. '13]

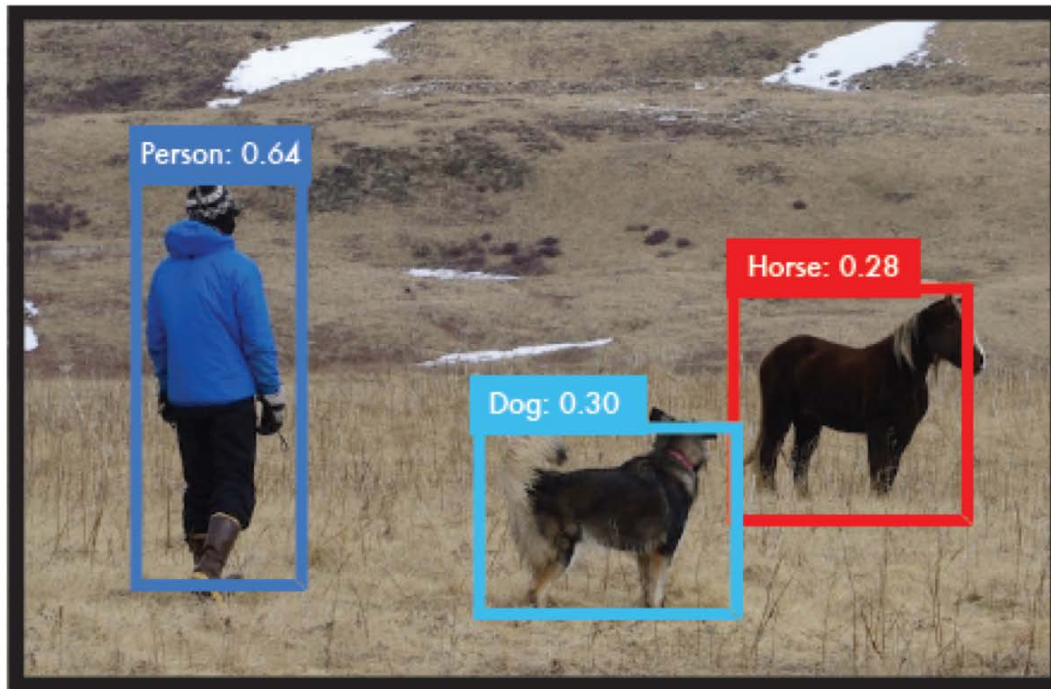


Semantic Segmentation



[Farabet et al. '13]

Object Detection

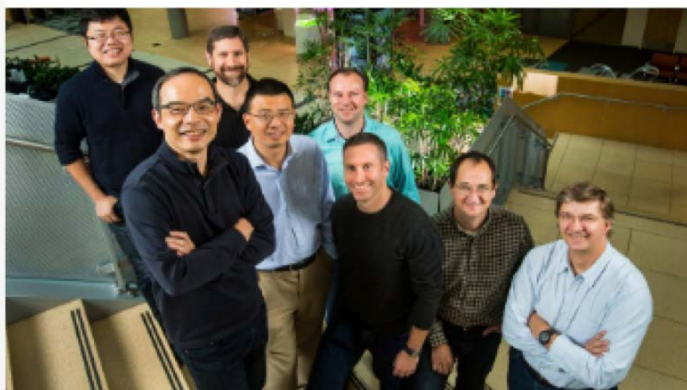


Redmon et al. 2015
<http://pjreddie.com/yolo/>

Exciting Applications: Natural Language (Spoken and Written)

Reaching Human Performance in Speech-to-Text

Historic Achievement: Microsoft researchers reach human parity in conversational speech recognition
October 18, 2016 | [Allison Linn](#)



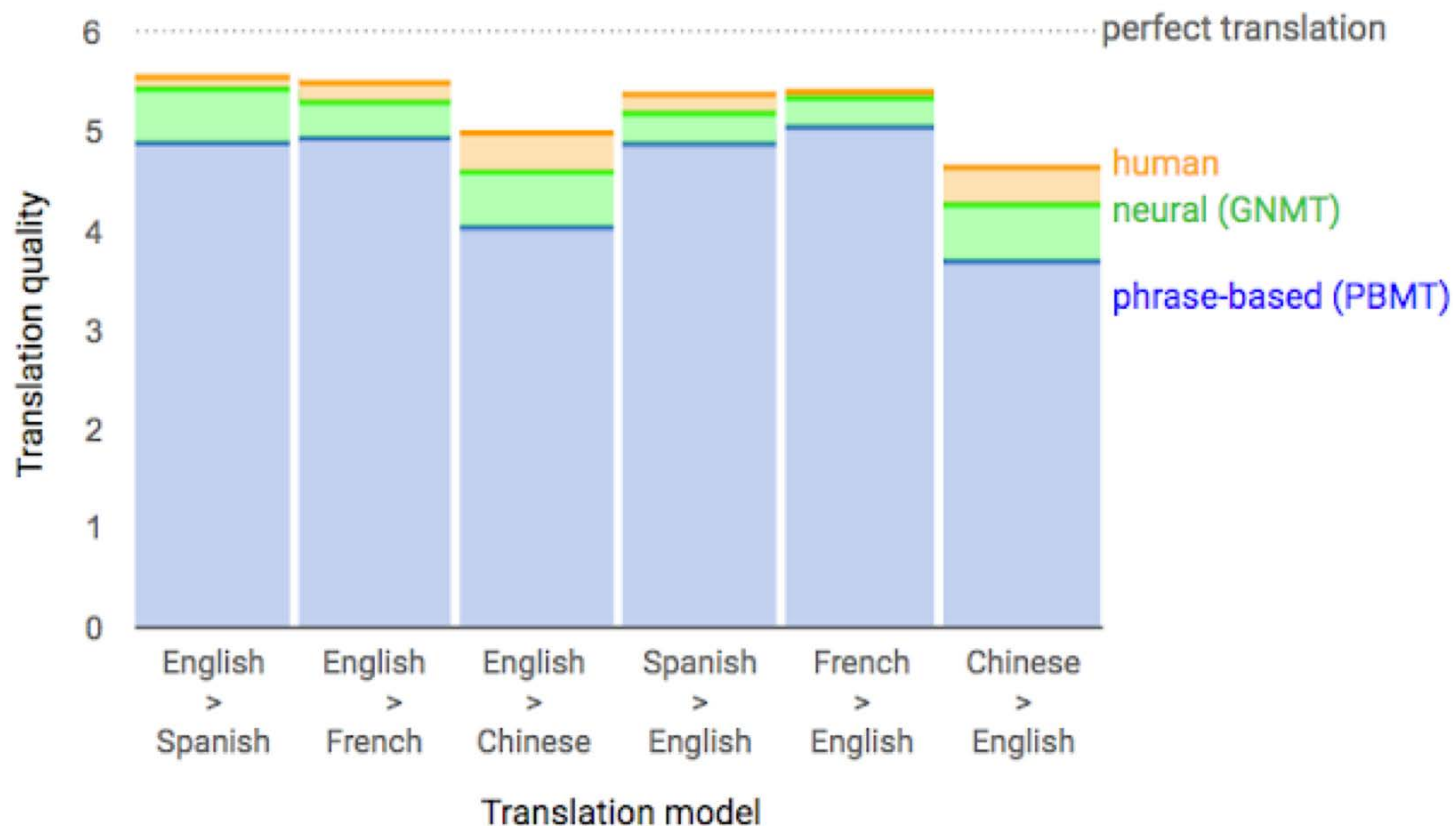
 Microsoft | The AI Blog

<https://arxiv.org/pdf/1610.05256.pdf>

In a paper [published Monday](#), a team of researchers and engineers in Microsoft Artificial Intelligence and Research reported a speech recognition system that makes the same or fewer errors than professional transcriptionists. The researchers reported a word error rate (WER) of 5.9 percent, down from the 6.3 percent WER the team [reported](#) just last month.

To reach the human parity milestone, the team used [Microsoft Cognitive Toolkit](#), a homegrown system for deep learning that the research team has made available on [GitHub](#) via an open source license.

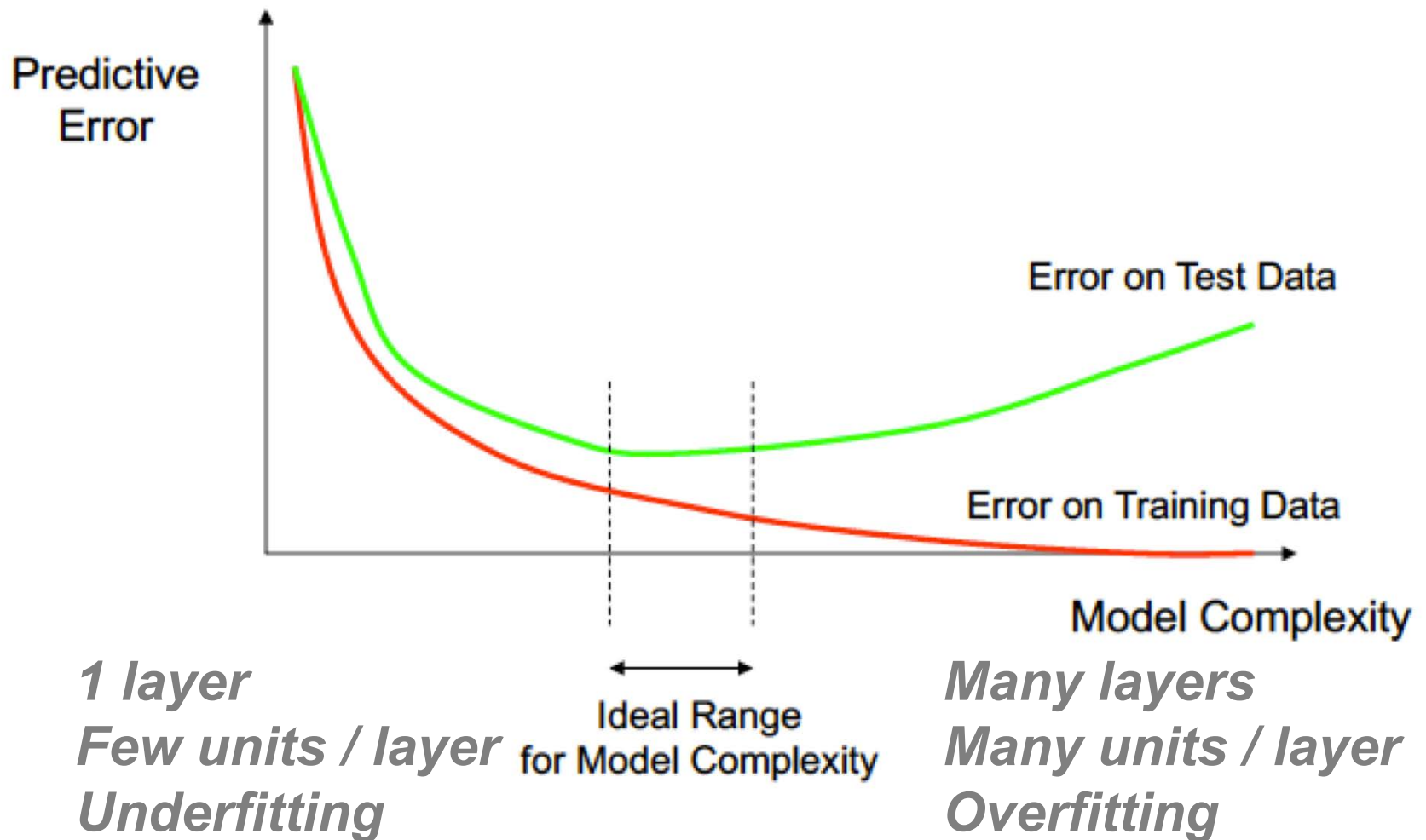
Gains in Translation Quality



<https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html>

Any Disadvantages?

Deep Neural Networks can overfit!



Ways to avoid overfitting

- More training data!
- L2 / L1 penalties on weights
- More tricks next time....
 - Early stopping
 - Dropout

Objectives Today:

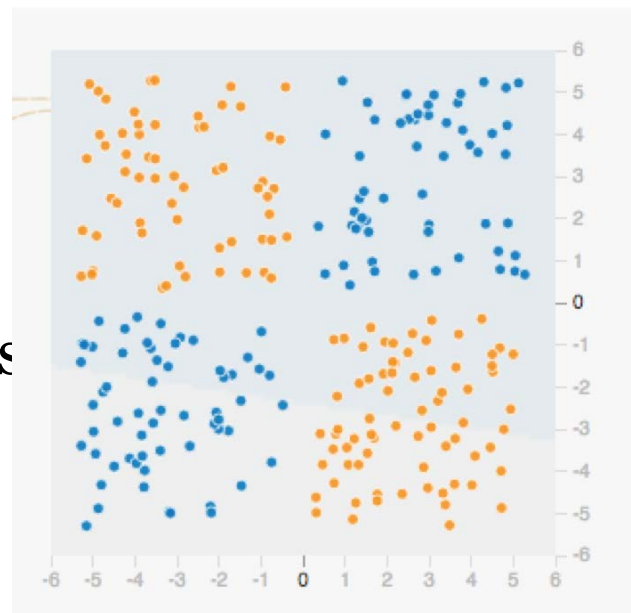
Neural Networks Unit 1/2

- How to **learn** feature representations
 - Feed-forward neural nets (MLPs)
 - Universal approximation
 - Activation functions
- The Rise of Deep Learning:
 - Success stories on Images and Language
- Preview: Training via gradient descent
 - Back-propagation = gradient descent + chain rule

Interactive Demo

- <https://playground.tensorflow.org>

- Use XOR dataset
- Explore impact of:
 - Number of hidden units
 - Activation function



How to train Neural Nets?
Just like logistic regression
Set up a loss function
Apply Gradient Descent!

Review: LR notation

- Feature vector with first entry constant

$$\tilde{x}_n = [1 \ x_{n1} \ x_{n2} \ \dots \ x_{nF}]$$

- Weight vector (first entry is the “bias”)

$$w = [w_0 \ w_1 \ w_2 \ \dots \ w_F]$$

- “Score” value z (real number, -inf to +inf)

$$z_n = w^T \tilde{x}_n$$

Review: Gradient of LR

$$z_n = w^T \tilde{x}_n$$

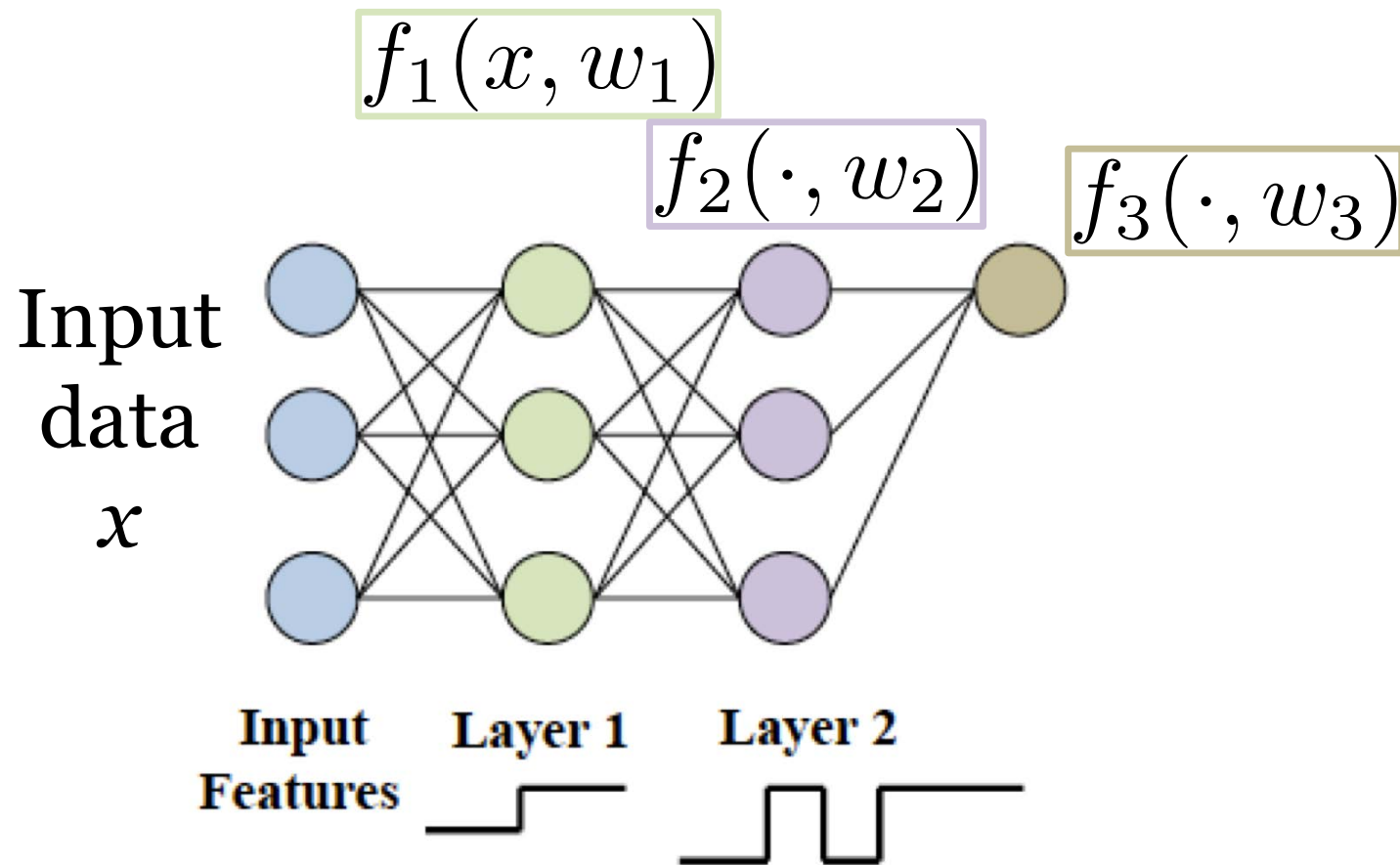
Log likelihood

$$J(z_n(w)) = y_n z_n - \log(1 + e^{z_n})$$

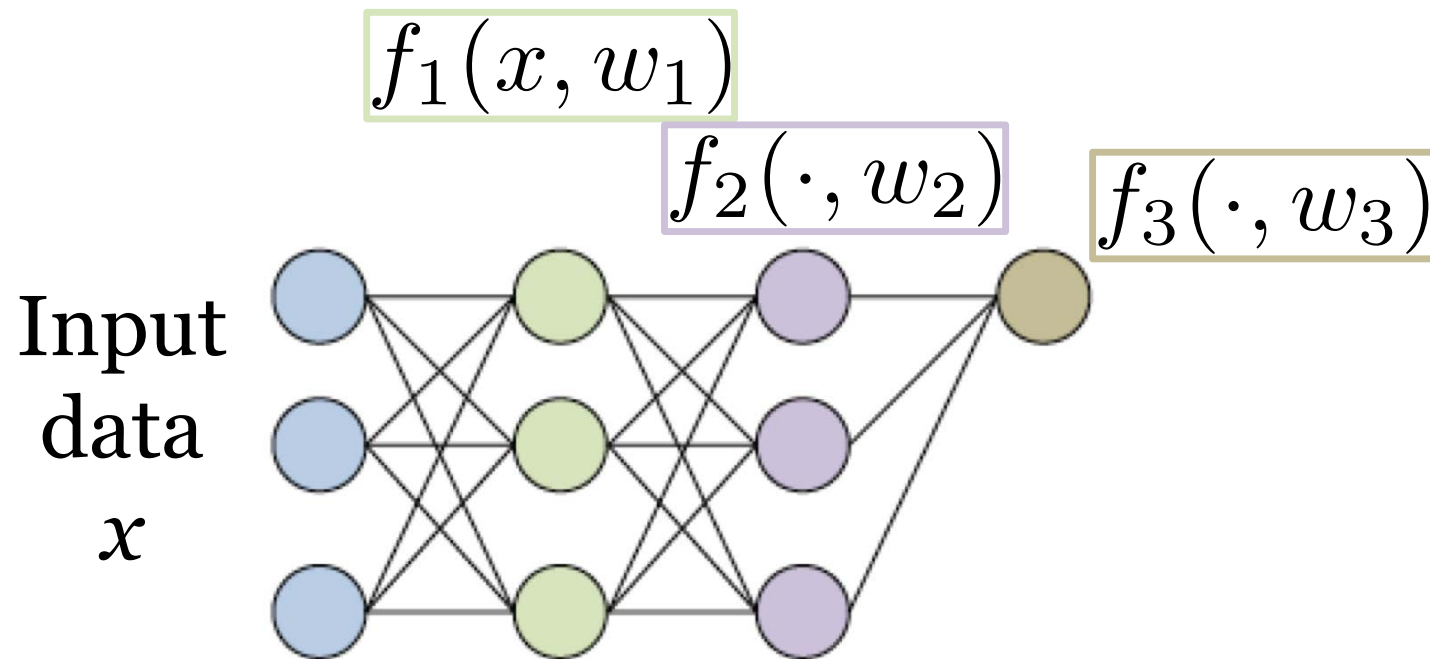
Gradient w.r.t. weight on feature f

$$\frac{d}{dw_f} J(z_n(w)) = \frac{d}{dz_n} J(z_n) \cdot \frac{d}{dw_f} z(w)$$

MLP : Composable Functions



Output as function of x



$$f_3(f_2(f_1(x, w_1), w_2), w_3)$$

Minimizing loss for composable functions

$$\min_{w_1, w_2, w_3} \sum_{n=1}^N \text{loss}(y_n, f_3(f_2(f_1(x_n, w_1), w_2), w_3))$$

Loss can be:

- Squared error for regression problems
- Log loss for binary classification problems
- ... many other possibilities!