Introduction to Deep Learning

1. Neural Networks 101

Haibin Lin and Leonard Lausen

gluon-nlp.mxnet.io

8:30-9:00	Continental Breakfast
9:00-9:45	Introduction and Setup
9:45-10:30	Neural Networks 101
10:30-10:45	Break
10:45-11:15	Machine Learning Basics
11:15-11:45	Context-free Representations for Language
11:45-12:15	Convolutional Neural Networks
12:15-13:15	Lunch Break
13:15-14:00	Recurrent Neural Networks
14:00-14:45	Attention Mechanism and Transformer
14:45-15:00	Coffee Break
15:00-16:15	Contextual Representations for Language
16:15-17:00	Language Generation

Outline

- **Linear Model**
  - Single layer network
  - XOR is hard
- **Multilayer Perceptron**
  - Layers
  - Nonlinearities
  - Computational Cost

House Buying 101

- Pick a house, take a tour, and read facts
- Estimate its price, bid



Listing price from agent

$5,498,000	7	5	4,865 Sq. Ft.
Price	Beds	Baths	$1130 / Sq. Ft.

Redfin Estimate: $5,390,037 On Redfin: 15 days

Predicted sale price

Virtual Tour

- Branded Virtual Tour
- Virtual Tour (External Link)

Parking Information

- Garage (Minimum): 2
- Garage (Maximum): 2
- Parking Description: Attached Garage, On Street
- Garage Spaces: 2

Multi-Unit Information

- # of Stories: 2

School Information

- Elementary School: El Carmelo El
- Elementary School District: Palo
- Middle School: Jane Lathrop Stan
- High School: Palo Alto High
- High School District: Palo Alto Un

Interior Features

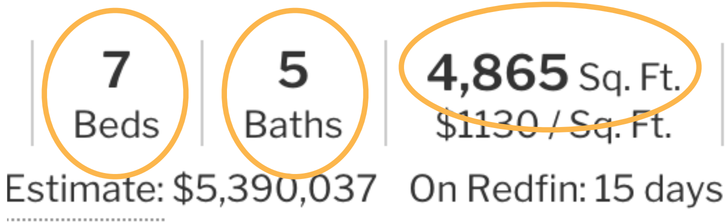
Bedroom Information

- # of Bedrooms (Minimum): 7

- Kitchen Description: Countertop Dishwasher, Garbage Disposal, H Island with Sink, Microwave, Over

gluon-nlp.mxnet.io

A Simplified Model



- **Assumption 1**
  The key factors impacting the prices are #Beds, #Baths, Living sqft, denoted by x_1, x_2, x_3
- **Assumption 2**
  The sale price is a weighted sum over the key factors

$$y = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

Weights and bias are determined later

Linear Model

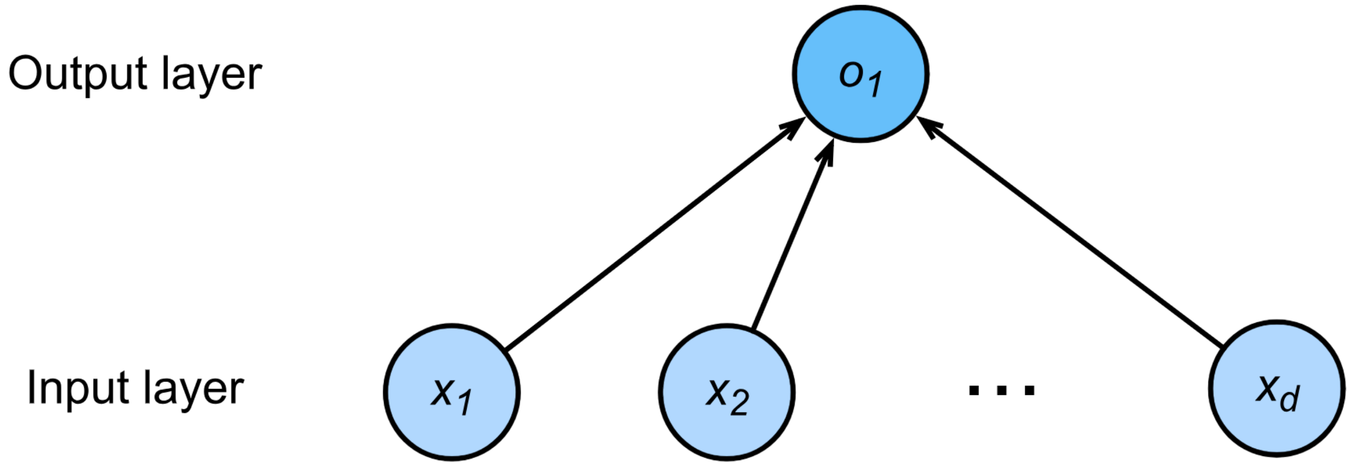
$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$$

$$\mathbf{w} = [w_1, w_2, \ldots, w_n]^T, \quad b$$

$$y = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

$$y = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

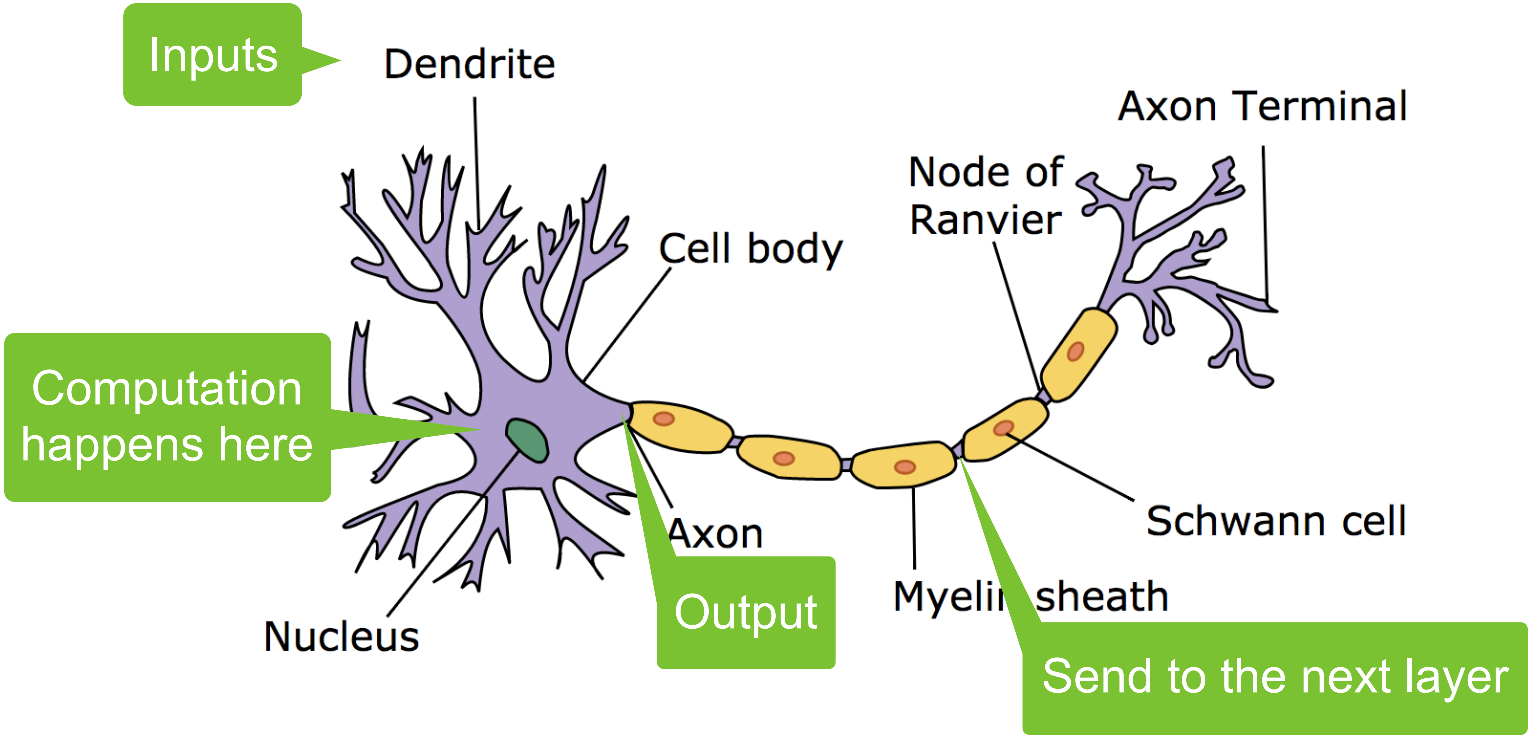
Linear Model as a Single-layer Neural Network



We can stack multiple layers to get deep neural networks

Neural Networks Derive from Neuroscience

The real neuron



Measure Estimation Quality

- Compare the true value vs the estimated value
  Real sale price vs estimated house price
- Let y the true value, and $\hat{y}$ the estimated value, we can compare the **squared loss**

$$\ell(y, \hat{y}) = \left(y - \hat{y}\right)^2$$

Learn Parameters

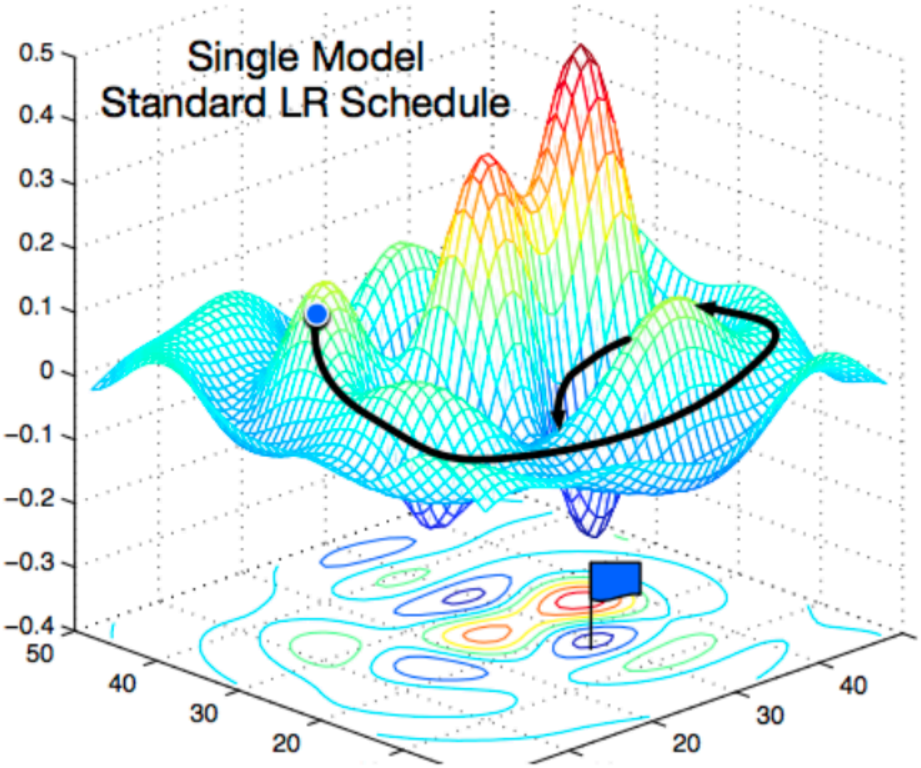
- Training loss

$$\ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \langle \mathbf{x}_i, \mathbf{w} \rangle - b\right)^2 = \frac{1}{n} \; \| \; \mathbf{y} - \mathbf{X}\mathbf{w} - b \; \| \; ^2$$

- Minimize loss to learn parameters

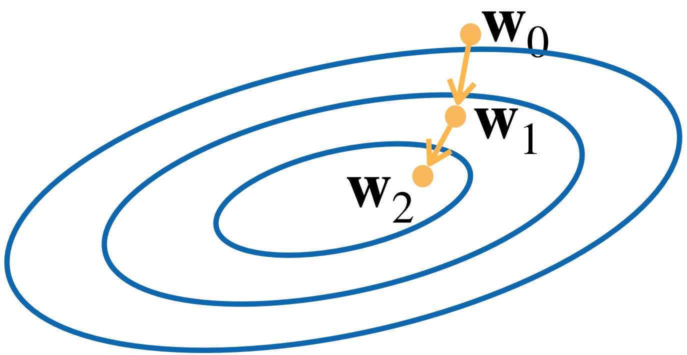
$$\mathbf{w}^*, \mathbf{b}^* = \arg\min_{\mathbf{w}, b} \ell(\mathbf{X}, \mathbf{y}, \mathbf{w}, b)$$

Basic Optimization



Gradient Descent

- Choose a staring point $\mathbf{w}_0$
- Repeat to update the weight *t=1,2,3*

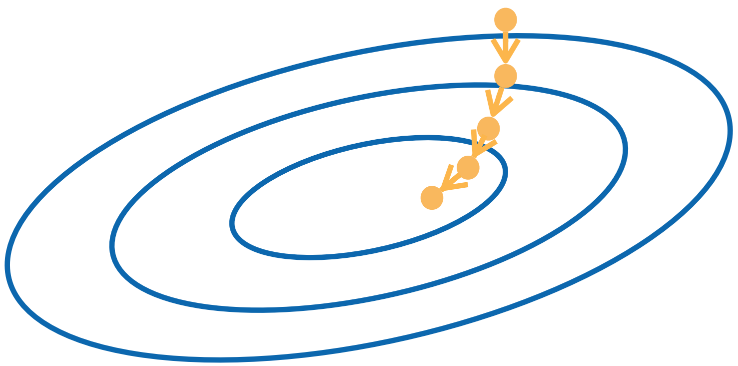


$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \frac{\partial \ell}{\partial \mathbf{w}_{t-1}}$$

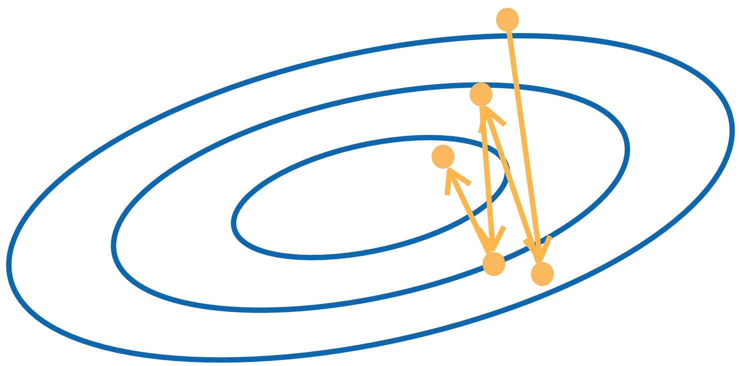
  - Gradient: a direction that increases the value
  - Learning rate: a hyper-parameter specifies the step length

Choose a Learning Rate

Not too small



Not too big



Mini-batch Stochastic Gradient Descent (SGD)

- Computing the gradient over the whole training data is too expensive
  - Takes minutes to hours for DNN models
- Randomly sample *b* examples $i_1, i_2, \ldots, i_b$ to approximate the loss

$$\frac{1}{b}\sum_{i\in I_b}\ell(\mathbf{x}_i, y_i, \mathbf{w})$$

  - *b* is the batch size, another important hyper-parameters

Choose a Batch Size

Not too small

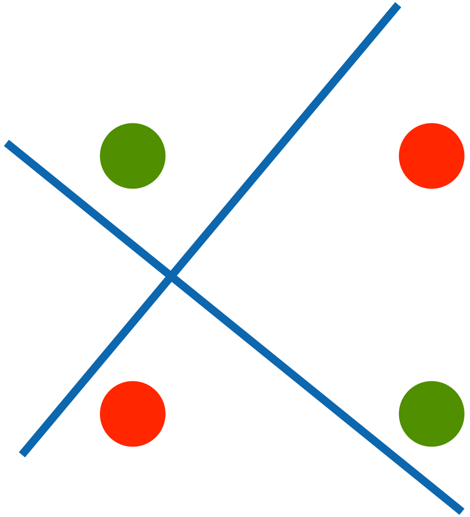
Workload is too small, hard to fully utilize computation resources

Not too big

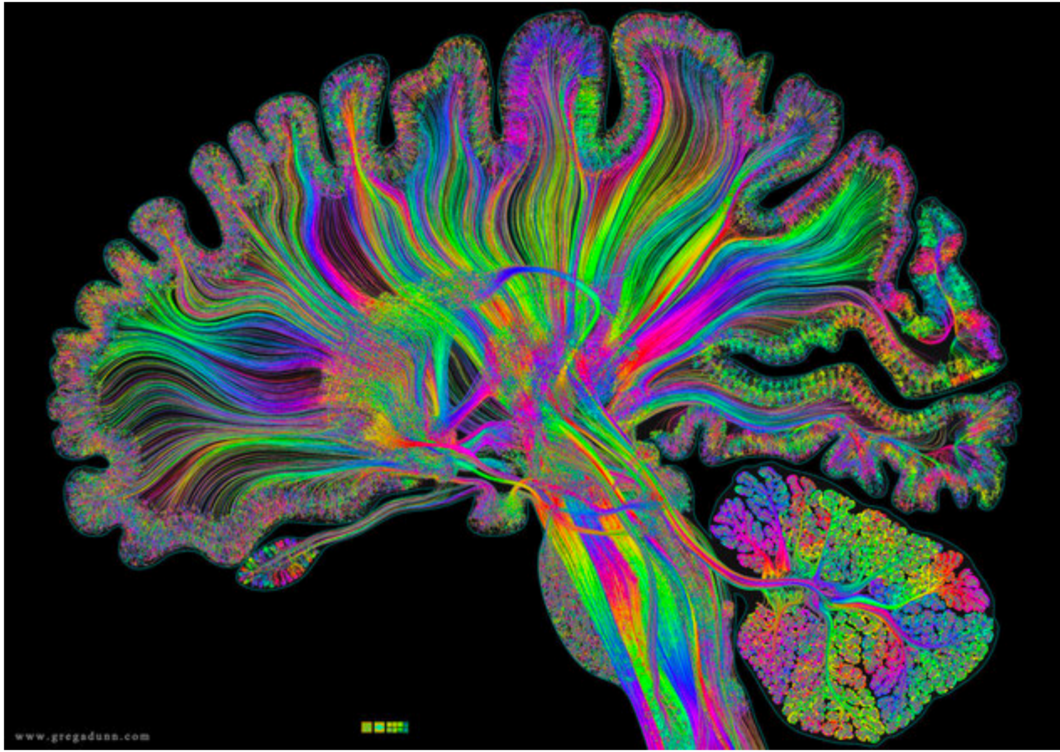
Memory issue
Waste computation, e.g. when all x_i are identical

XOR Problem (Minsky & Papert, 1969)

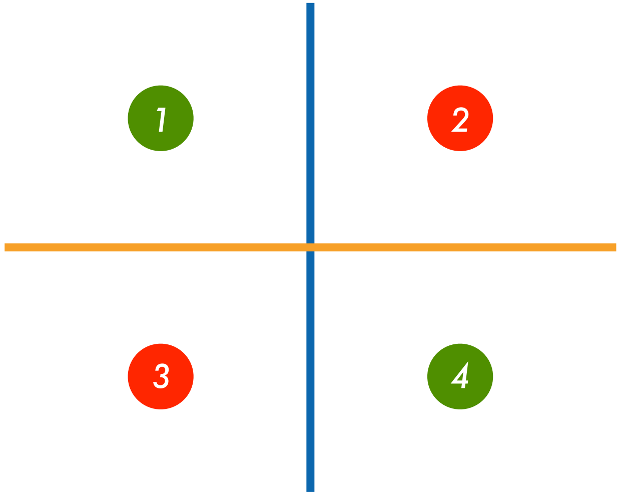
The perceptron cannot learn an XOR function
(neurons can only generate linear separators)



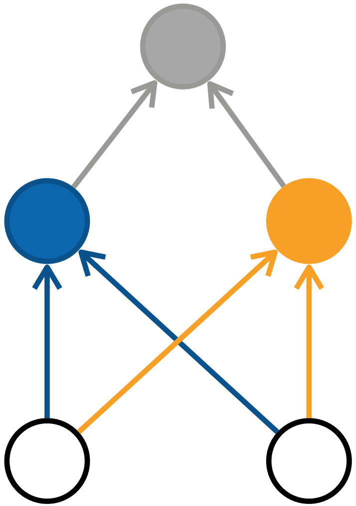
Multilayer Perceptron



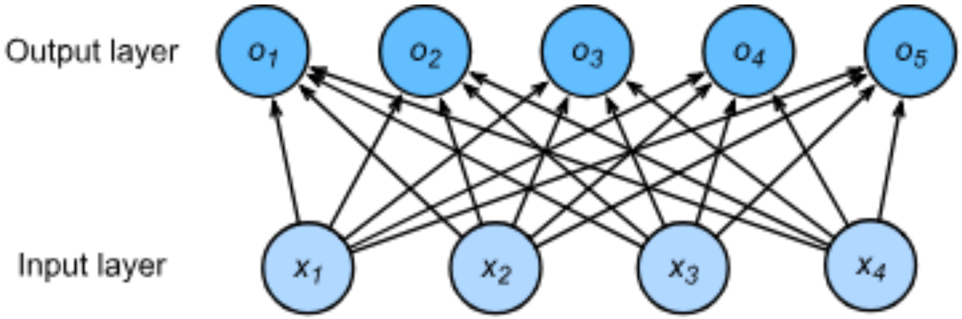
Learning XOR



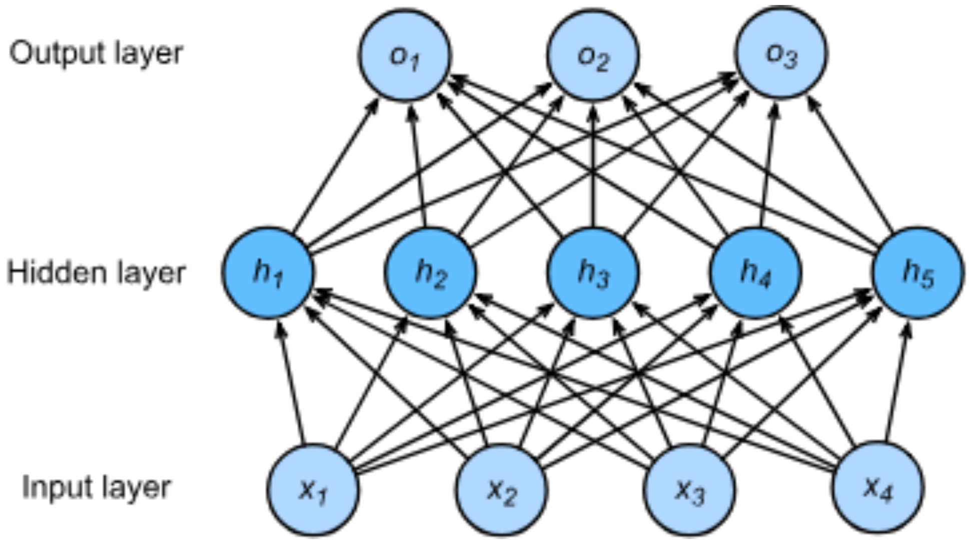
	1	2	3	4
	+	-	+	-
	+	+	-	-
product	+	-	-	+



Single Hidden Layer



Single Hidden Layer



Hyperparameter - size m of hidden layer

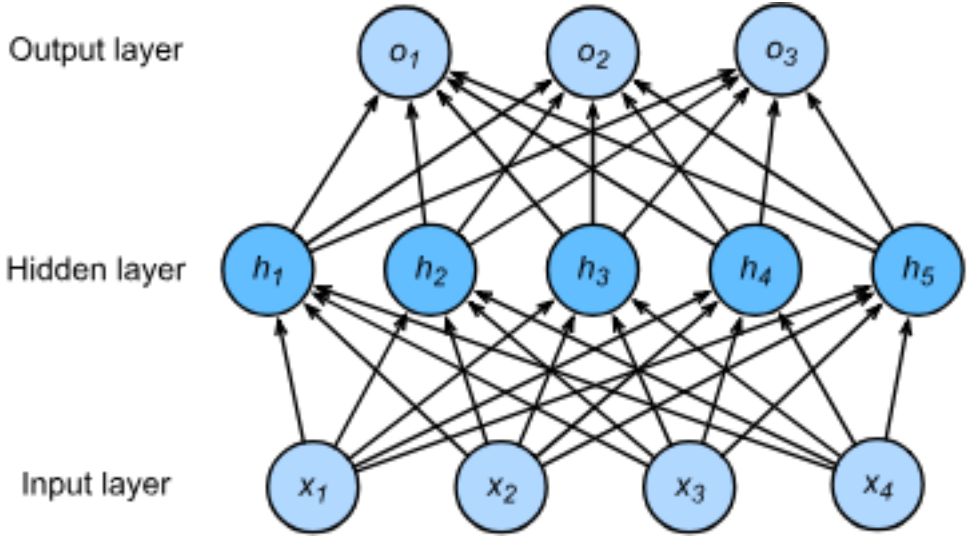
Single Hidden Layer

- Input $\mathbf{x} \in \mathbb{R}^n$
- Hidden $\mathbf{W}_1 \in \mathbb{R}^{m \times n}, \mathbf{b}_1 \in \mathbb{R}^m$
- Output $\mathbf{w}_2 \in \mathbb{R}^m, b_2 \in \mathbb{R}$

$$\mathbf{h} = \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{o} = \mathbf{w}_2^T \mathbf{h} + b_2$$

σ is an element-wise activation function



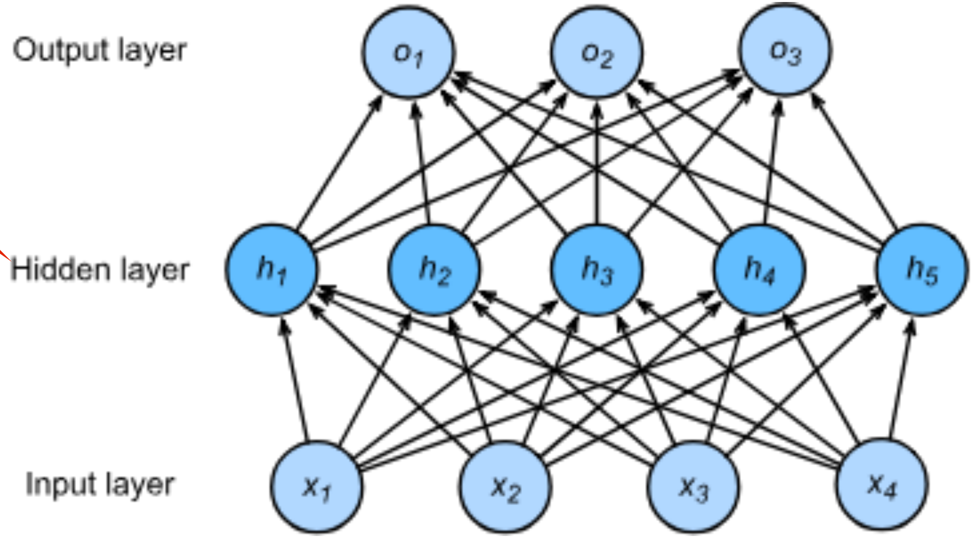
Single Hidden Layer

Why do we need an a nonlinear activation?

$$\mathbf{h} = \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{o} = \mathbf{w}_2^T \mathbf{h} + b_2$$

σ is an element-wise activation function



Single Hidden Layer

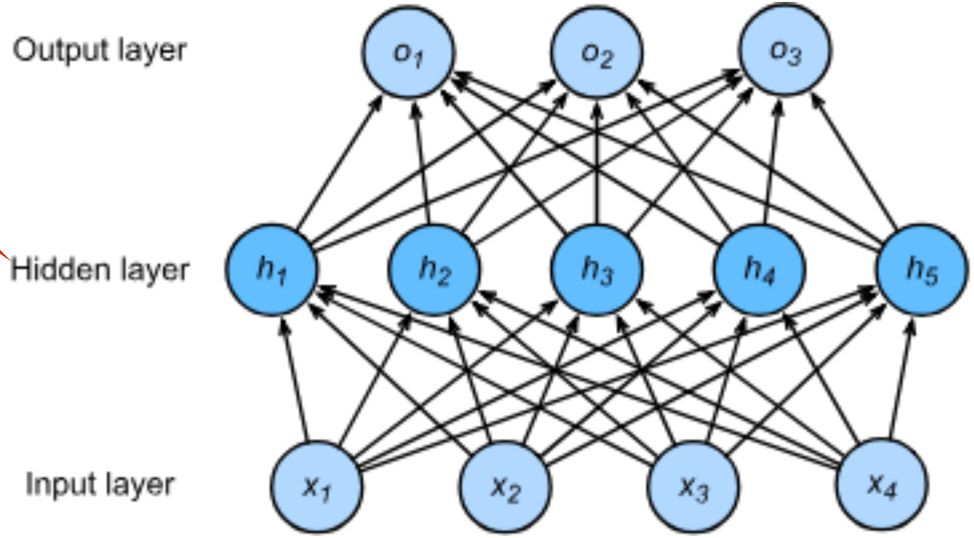
Why do we need an a nonlinear activation?

$$\mathbf{h} = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1$$

$$\mathbf{o} = \mathbf{w}_2^T\mathbf{h} + b_2$$

hence $o = \mathbf{w}_2^\top \mathbf{W}_1 \mathbf{x} + b'$

Linear ...



From Regression to Multi-class Classification

Calibrated Scale

- Output matches probabilities (nonnegative, sums to 1)

$$p(y \mid o) = \text{softmax}(o)$$
$$= \frac{\exp(o_y)}{\sum_i \exp(o_i)}$$

- Negative log-likelihood

$$-\log p(y \mid y) = \log \sum_i \exp(o_i) - o_y$$

Classification

- Multiple classes, typically multiple outputs
- Score *should* reflect confidence …