CS 639: Foundation Models
Deep Learning I

Fred Sala

University of Wisconsin-Madison

Jan. 27, 2026

Announcements

- **Midterm: Weds. March 11th**
  - More details as we get closer
- **Resources**
  - https://mlstory.org/ : fun book by Hardt and Recht
- Class roadmap:

Tuesday Jan. 27	Deep Learning I
Thursday Jan. 20	Deep Learning II
Tuesday Feb. 3	Self-Supervised Learning
Thursday Feb. 5	Guest Lecture

Review + Start FMs

Outline

- **Neural Networks**
  - Perceptrons, setup, training, limitations, multilayer perceptrons, loss functions (mostly from last time)
- **Convolutional Neural Networks**
  - Motivation, convolutional layers, CNN architectures
- **Sequence Models**
  - Recurrent neural networks, architecture, LSTMs, alternatives, training tricks

Outline

- **Neural Networks**
  - Perceptrons, setup, training, limitations, multilayer perceptrons, loss functions (mostly from last time)
- **Convolutional Neural Networks**
  - Motivation, convolutional layers, CNN architectures
- **Sequence Models**
  - Recurrent neural networks, architecture, LSTMs, alternatives, training tricks

From Last Time: Perceptron

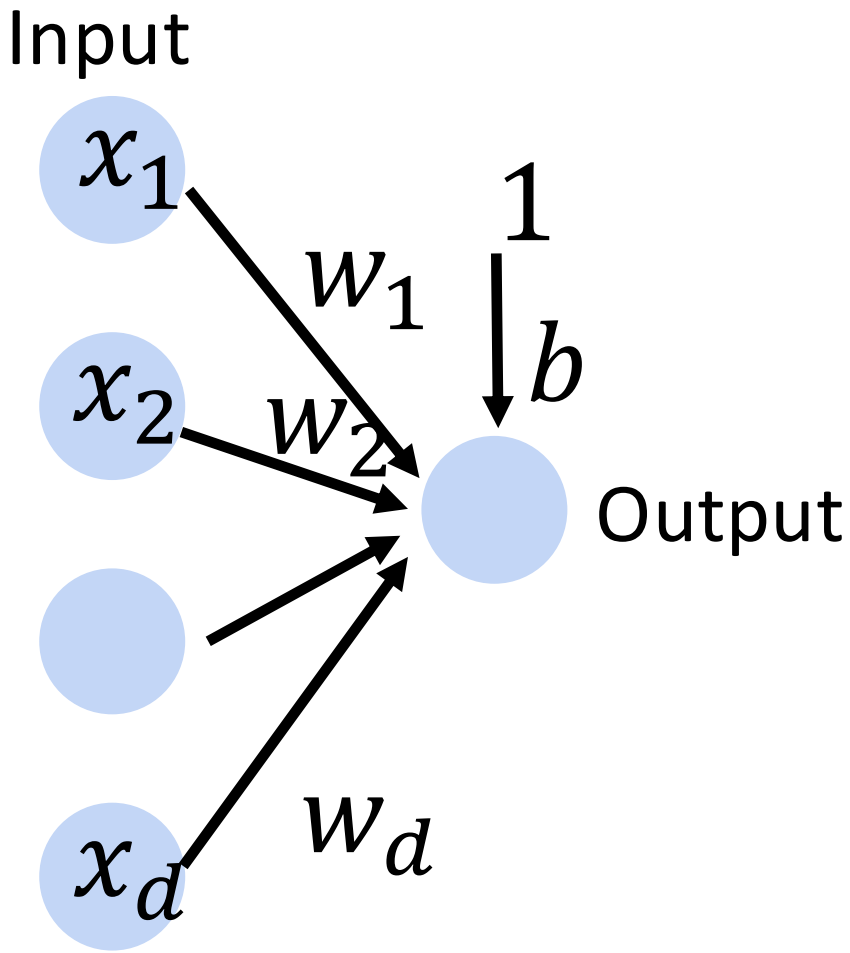
- Given input $\mathbf{x}$, weight $\mathbf{w}$ and bias b, perceptron outputs:

$$o = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

$$\sigma(x) = \begin{cases} 1 & if\ x > 0 \\ 0 & otherwise \end{cases}$$

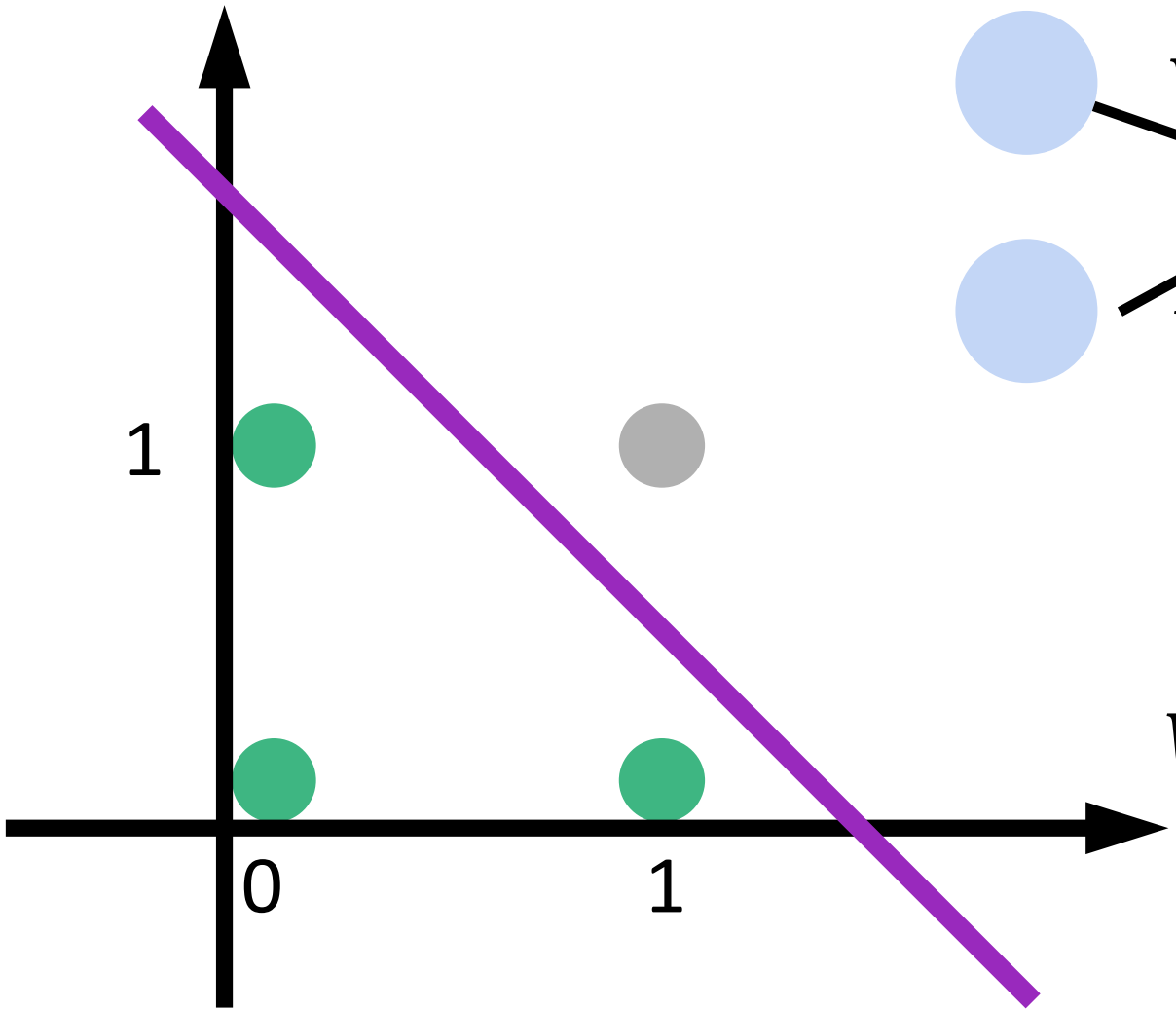
Activation function

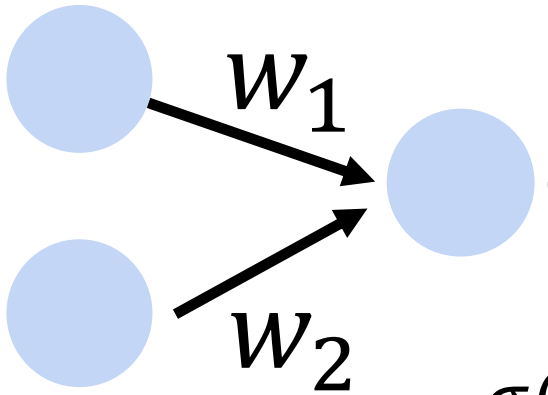
Cats vs. dogs?



Learning **AND** function using perceptron

The perceptron can learn an AND function





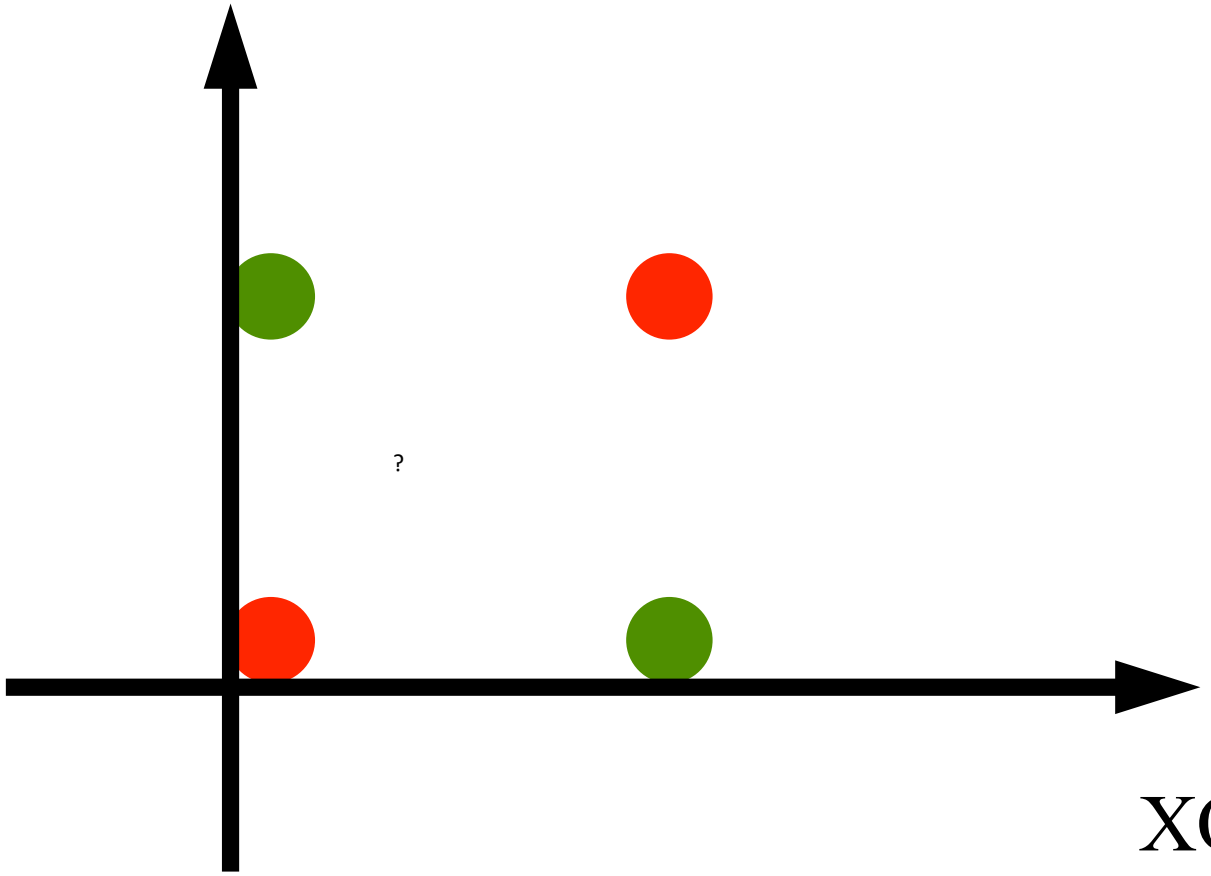
Output $\sigma(x_1 w_1 + x_2 w_2 + b)$

$$\sigma(x) = \begin{cases} 1 & if\ x > 0 \\ 0 & otherwise \end{cases}$$

$$w_1 = 1, w_2 = 1, b = -1.5$$

The limited power of a single neuron

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



$$x_1 = 1, x_2 = 1, y = 0$$

$$x_1 = 1, x_2 = 0, y = 1$$

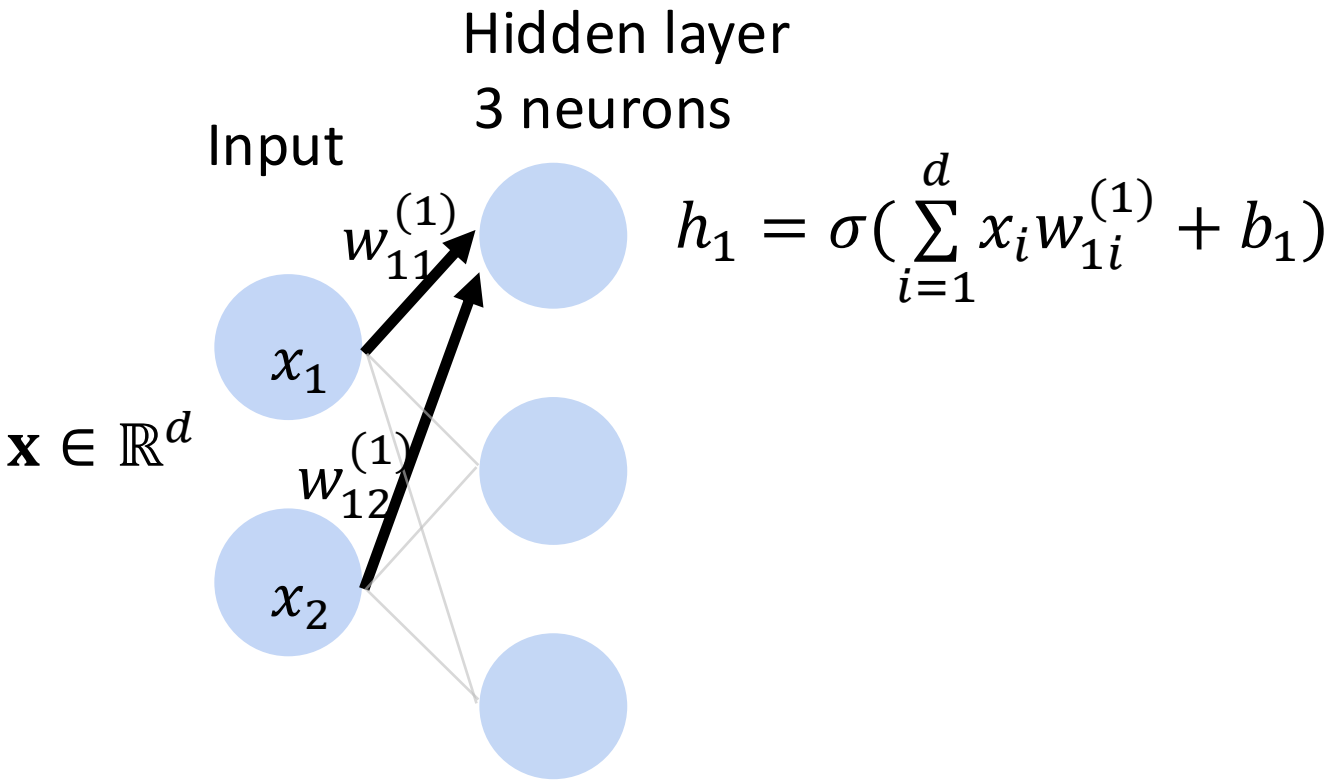
$$x_1 = 0, x_2 = 1, y = 1$$

$$x_1 = 0, x_2 = 0, y = 0$$

$$\mathrm{XOR}(x1,x2) = (x1 \wedge \neg x2) \vee (\neg x1 \wedge x2)$$

Multi-layer perceptron: Example

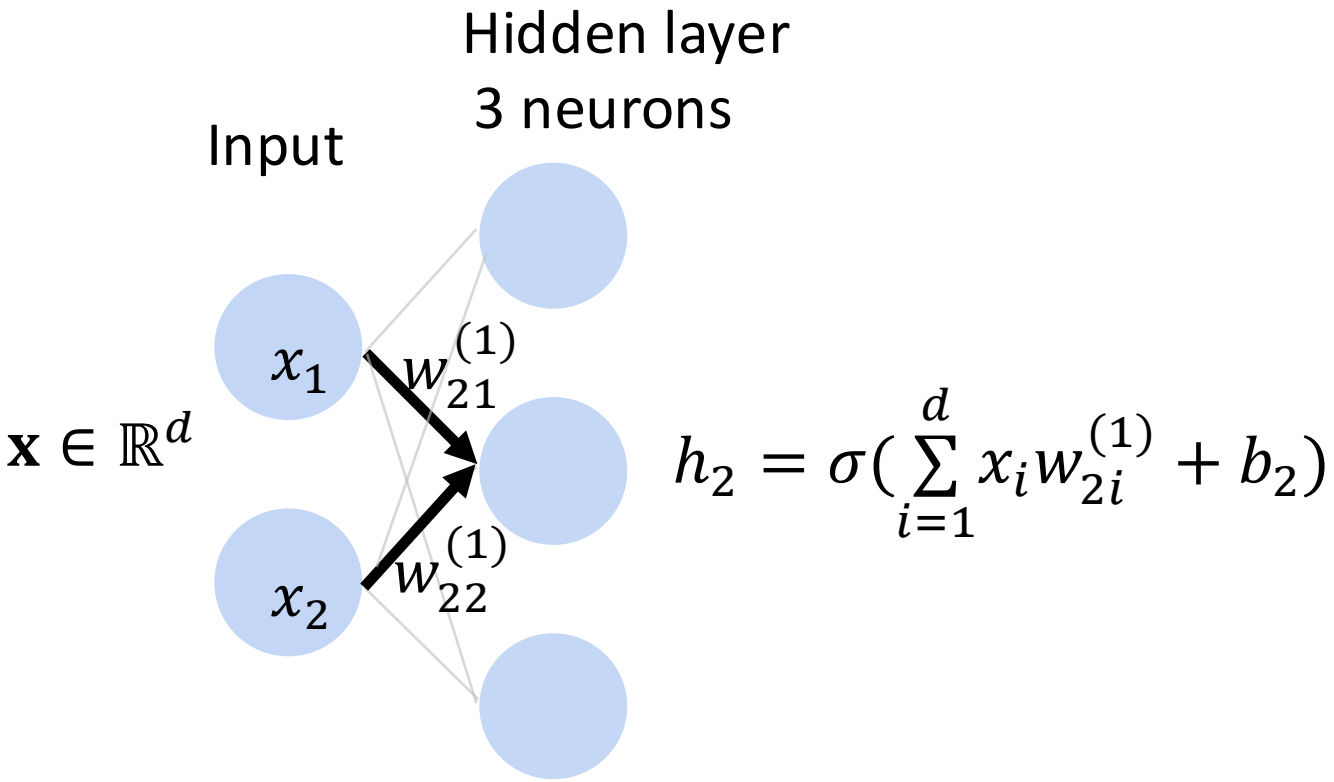
- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



$$h_1 = \sigma(\sum_{i=1}^{d} x_i w_{1i}^{(1)} + b_1)$$

Multi-layer perceptron: Example

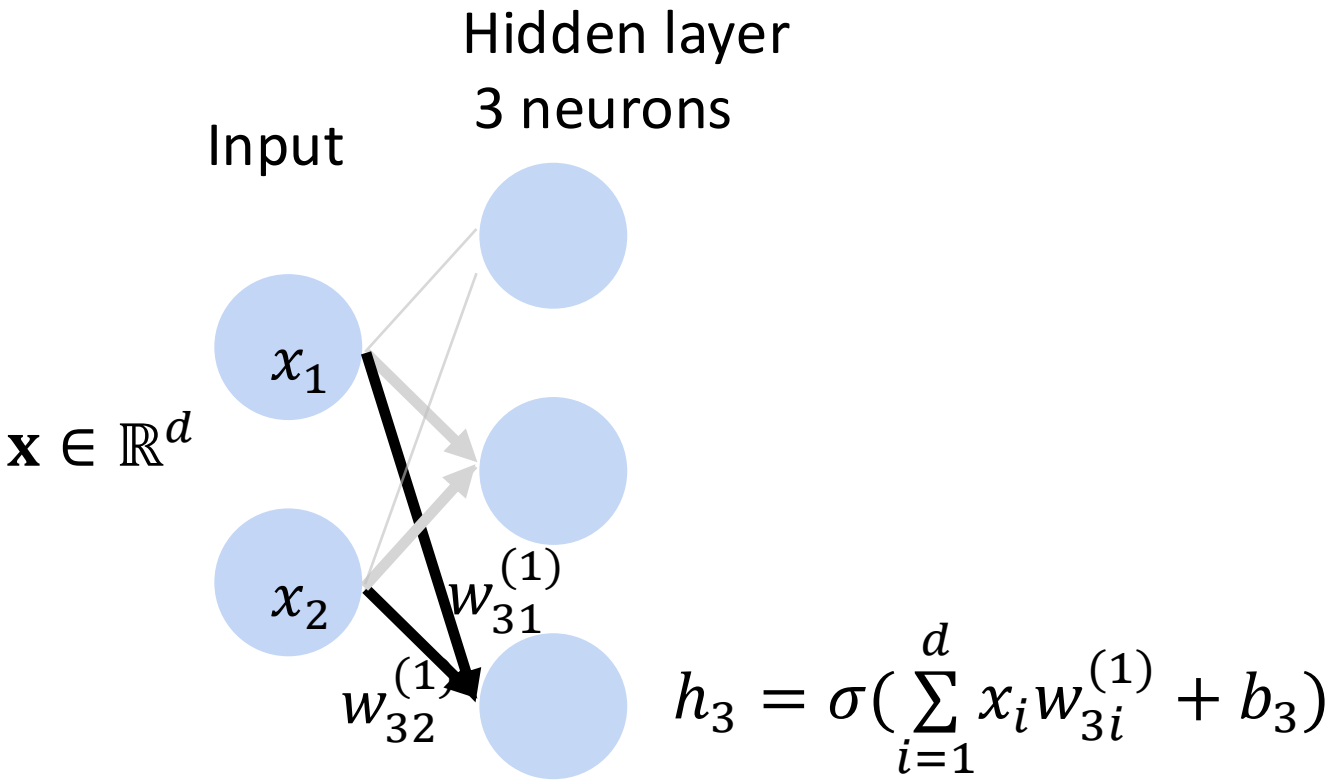
- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



$$h_2 = \sigma\left(\sum_{i=1}^{d} x_i w_{2i}^{(1)} + b_2\right)$$

Multi-layer perceptron: Example

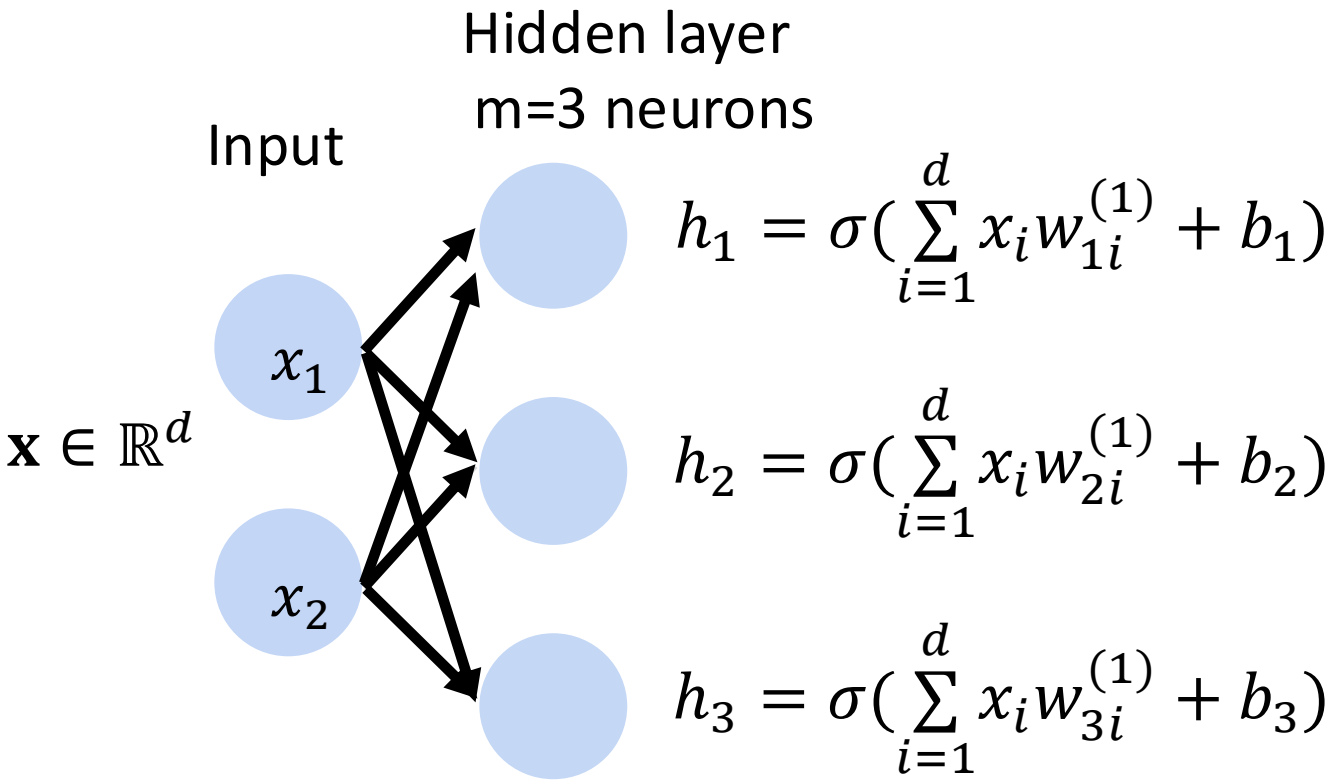
- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



$$h_3 = \sigma\left(\sum_{i=1}^{d} x_i w_{3i}^{(1)} + b_3\right)$$

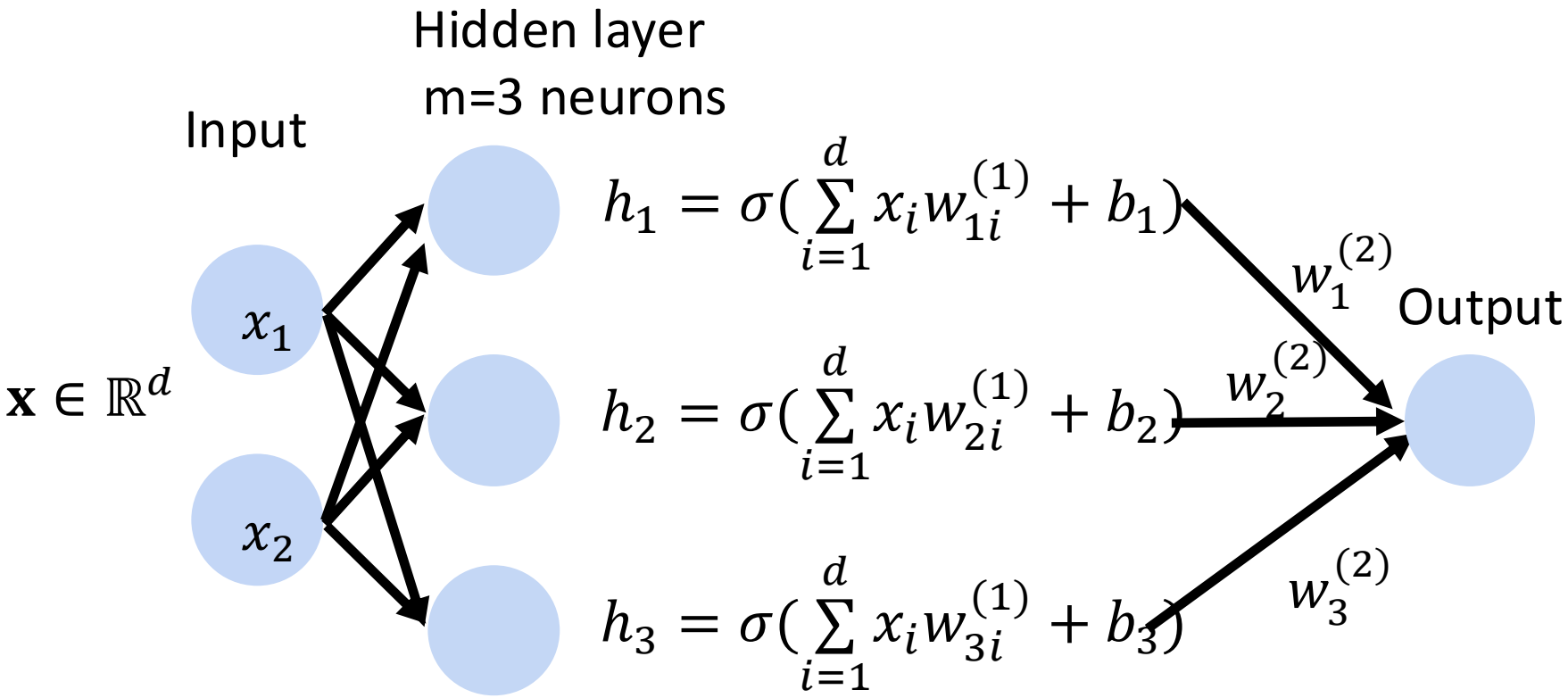
Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



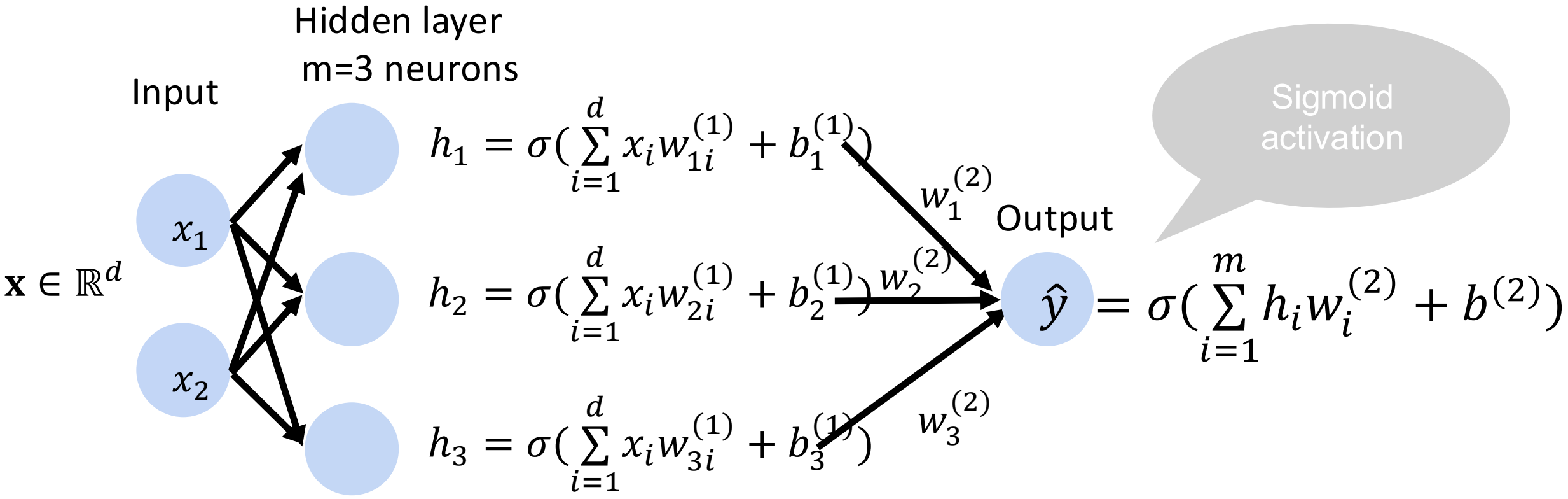
Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2

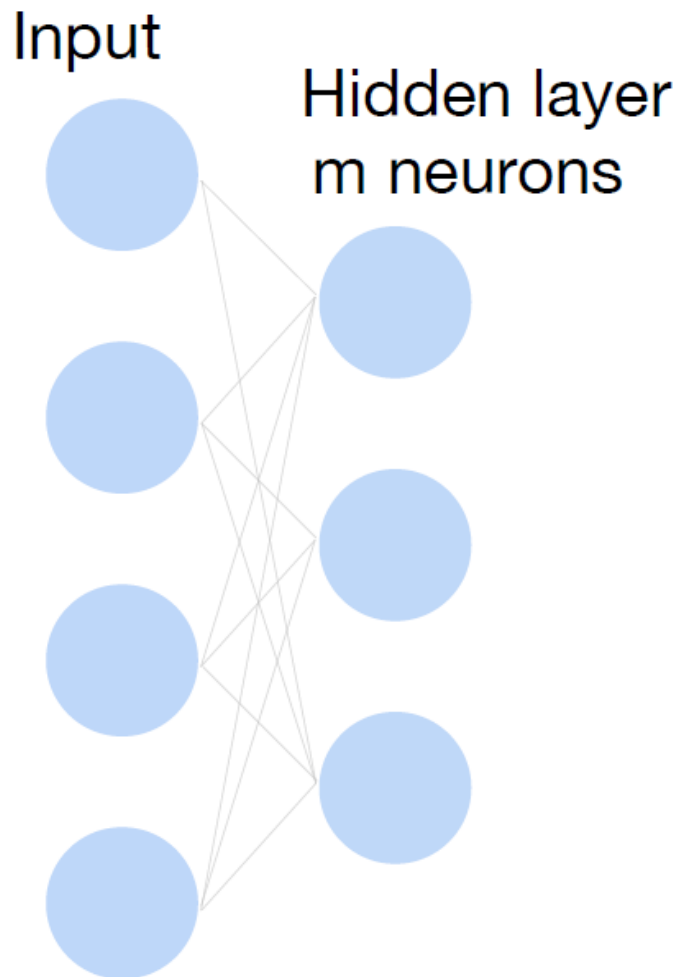


Multi-layer perceptron: Matrix Notation

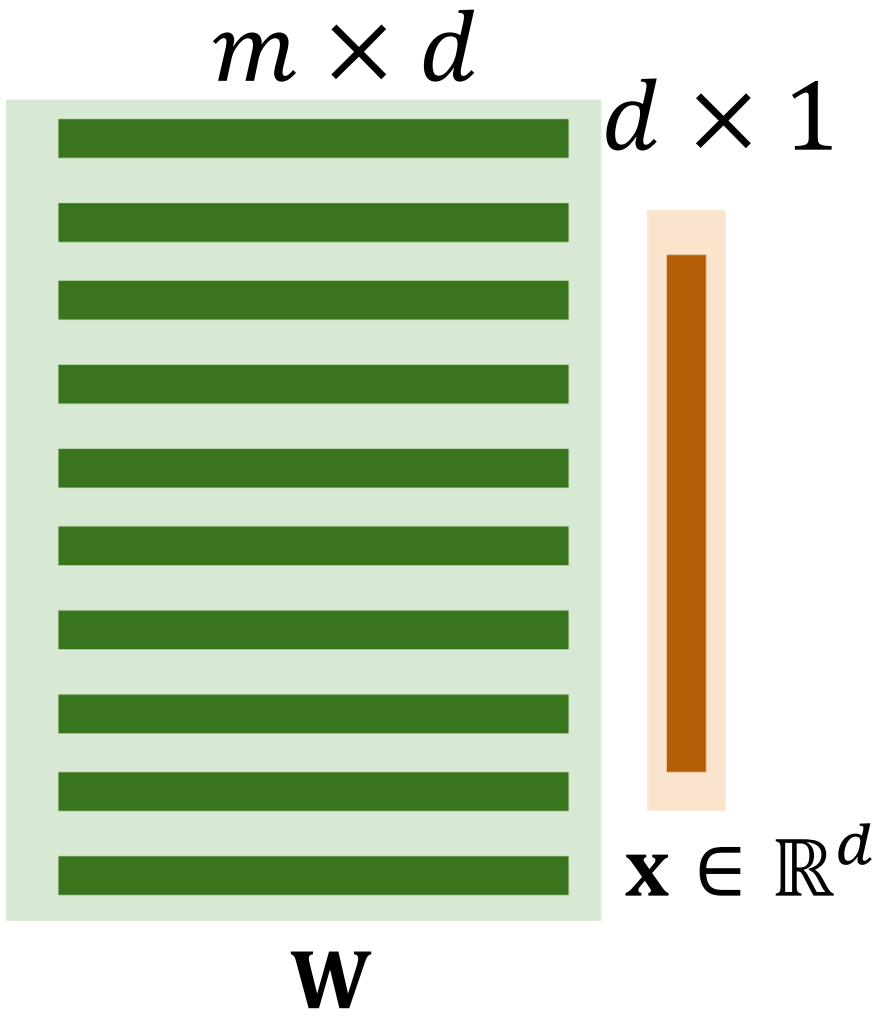
- Input $\mathbf{x} \in \mathbb{R}^d$
- Hidden $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}, \mathbf{b}^{(1)} \in \mathbb{R}^m$
- Intermediate output

$$\mathbf{h} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\mathbf{h} \in \mathbb{R}^m$$

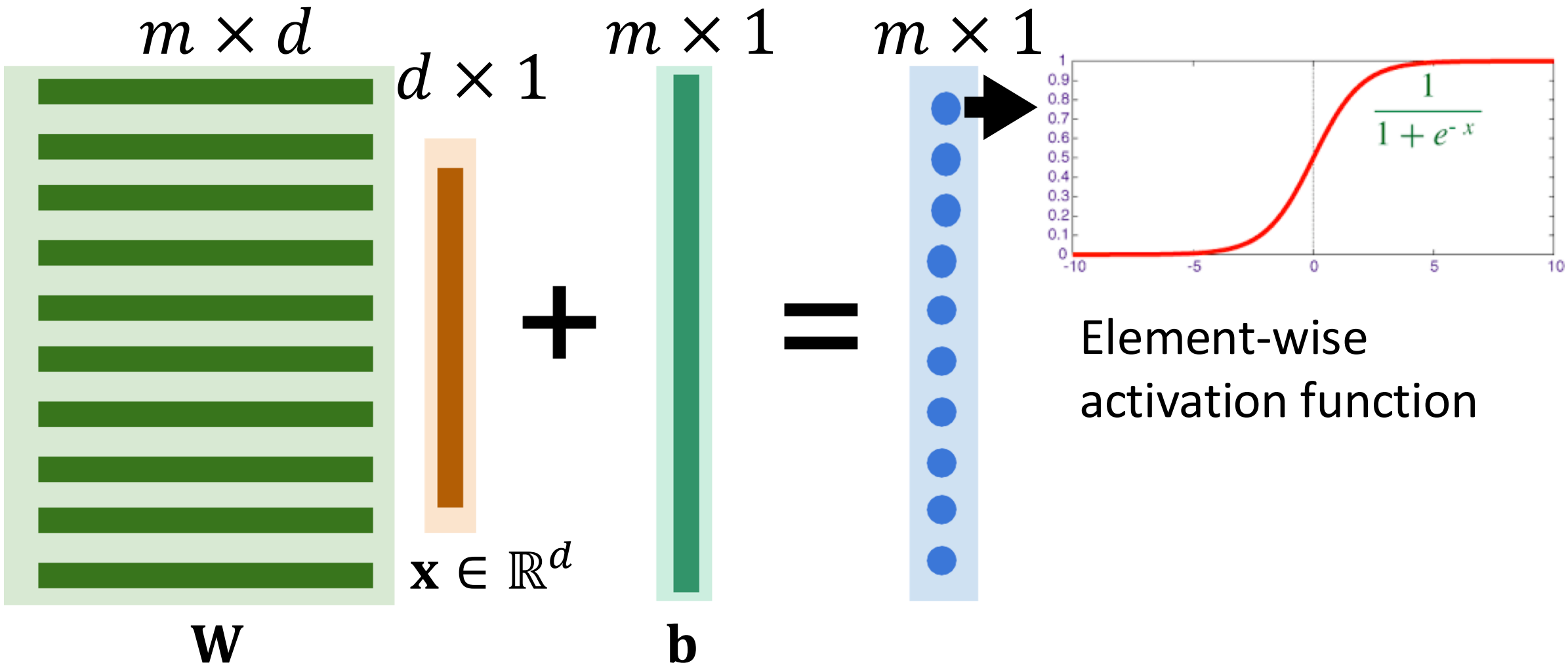


Multi-layer perceptron: **Matrix Notation**



Multi-layer perceptron: **Matrix Notation**

Key elements: linear operations + Nonlinear activations



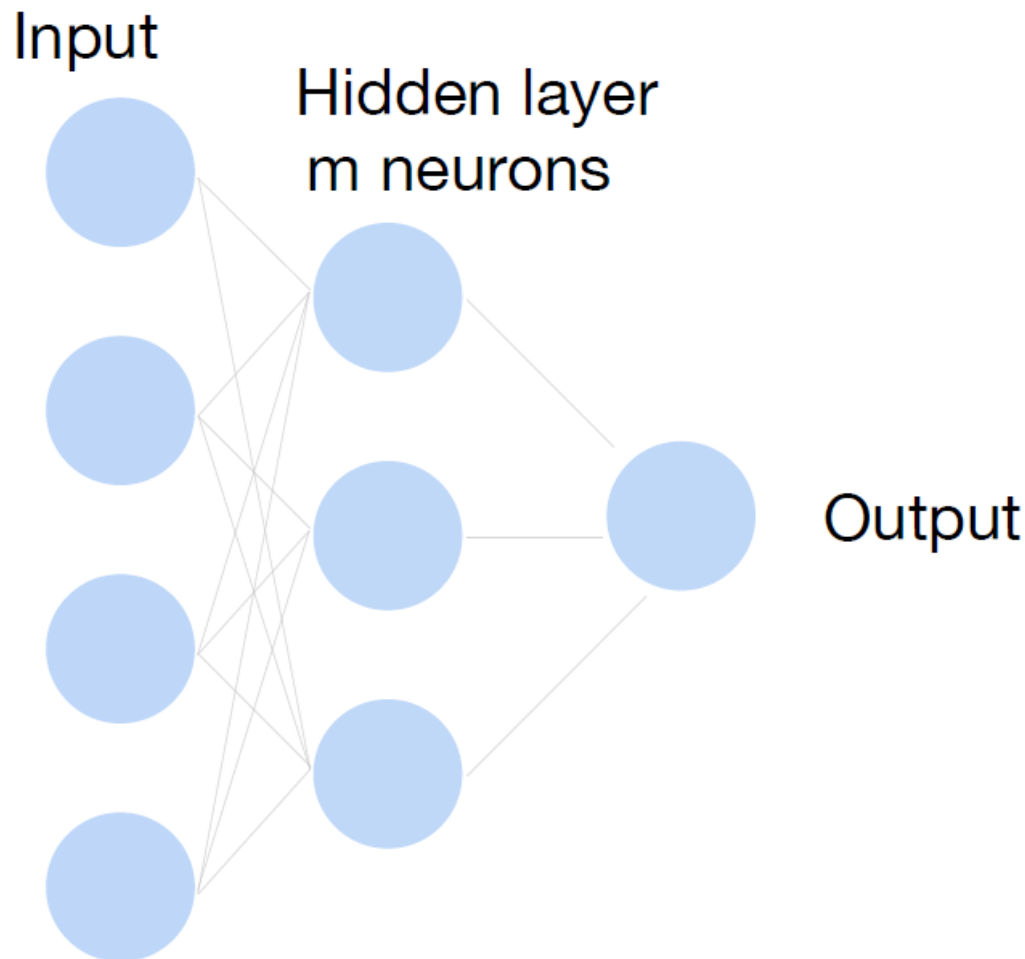
Element-wise activation function

Multi-layer perceptron: Matrix Notation

- Input $\mathbf{x} \in \mathbb{R}^d$
- Hidden $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}, \mathbf{b}^{(1)} \in \mathbb{R}^m$
- Intermediate output

$$\mathbf{h} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

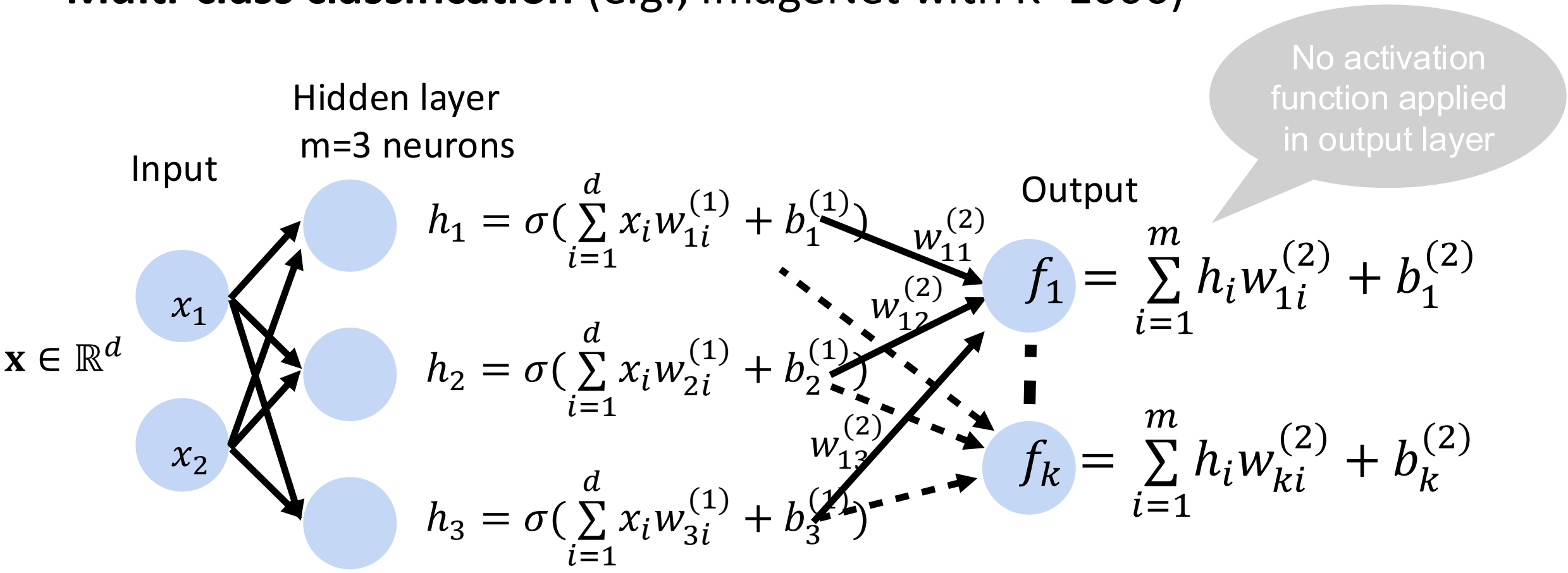
$$\hat{y} = \sigma(\mathbf{w}^{(2)}\mathbf{h} + \mathbf{b}^{(2)})$$



Neural network for k-way classification

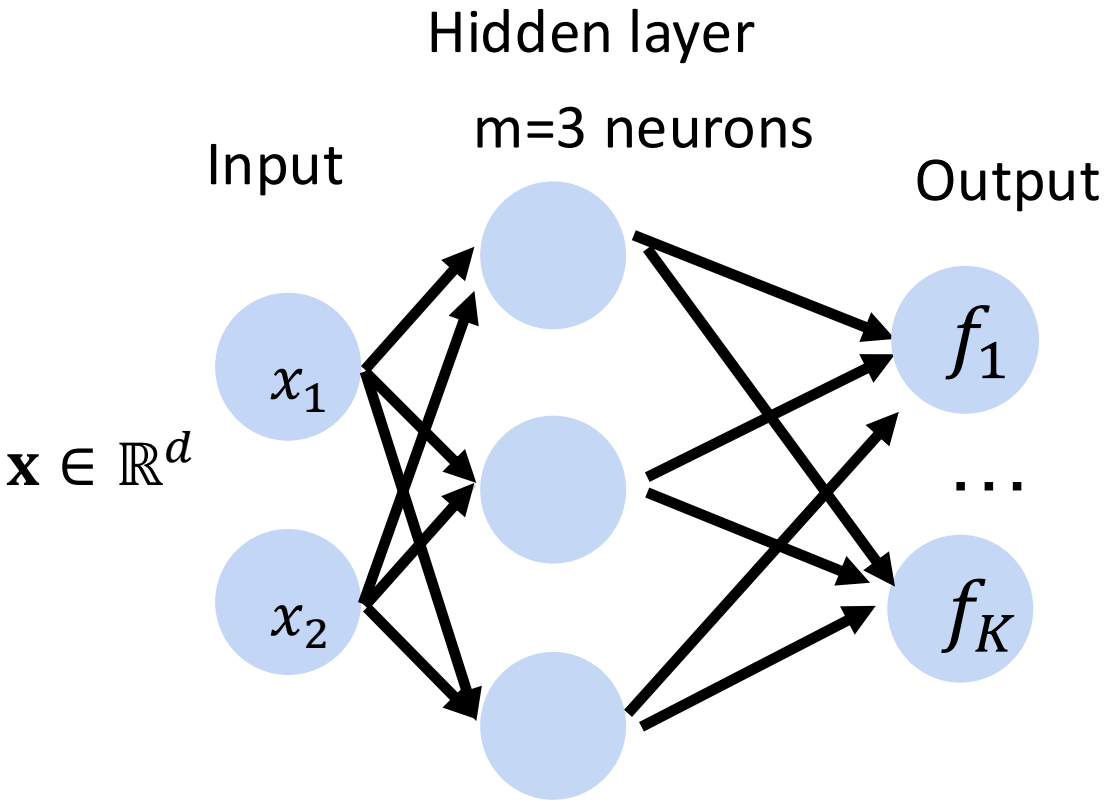
- K outputs in the final layer

Multi-class classification (e.g., ImageNet with K=1000)



Softmax

Turns outputs f into probabilities (sum up to 1 across K classes)

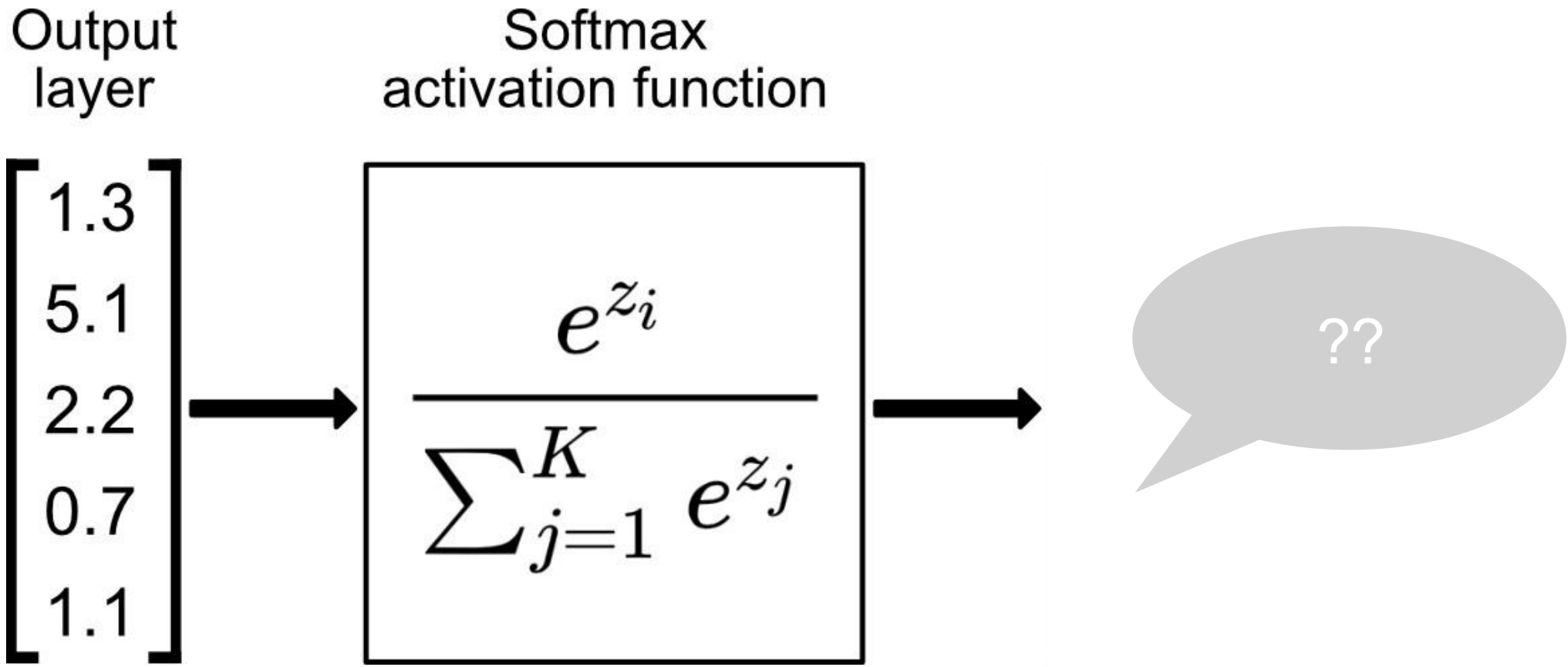


$$p(y|\mathbf{x}) = softmax(f)$$

$$= \frac{\exp(f_y(x))}{\sum_{k=1}^{K} \exp(f_k(x))}$$

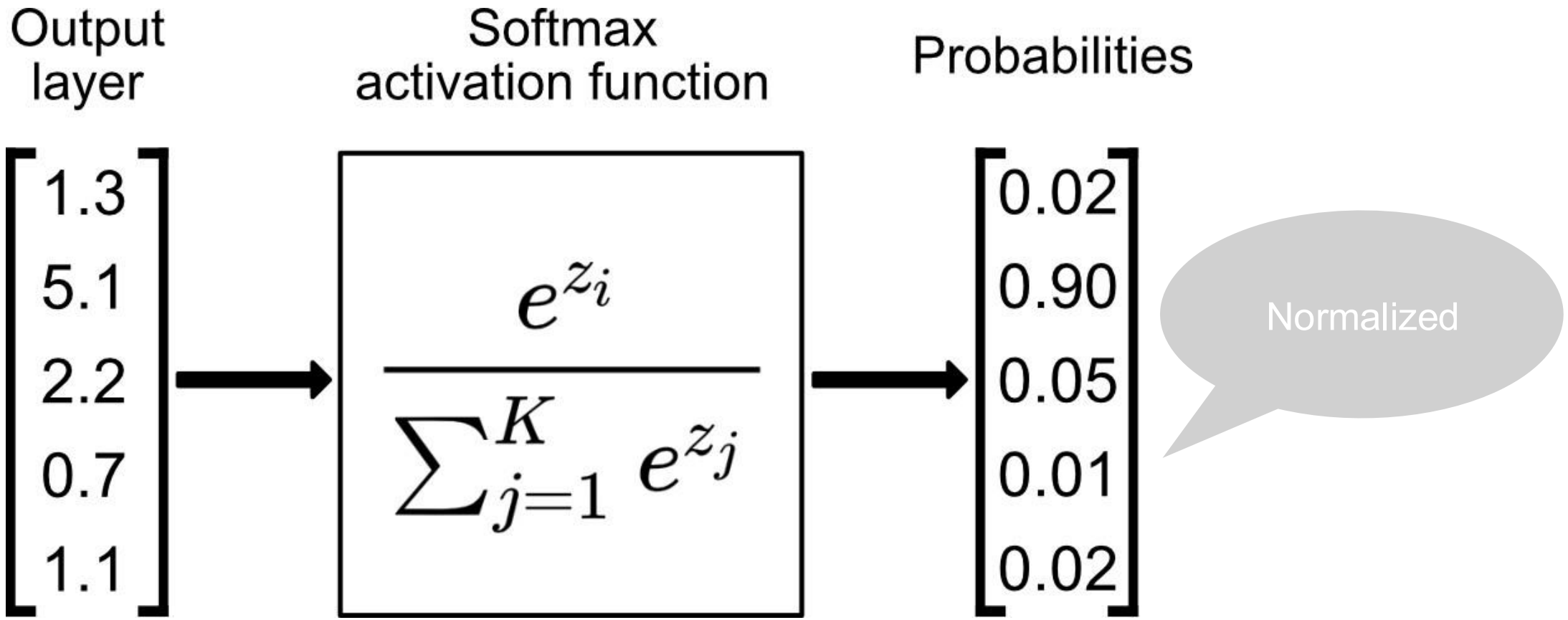
Softmax

Turns outputs f into probabilities (sum up to 1 across K classes)



Softmax

Turns outputs f into probabilities (sum up to 1 across K classes)



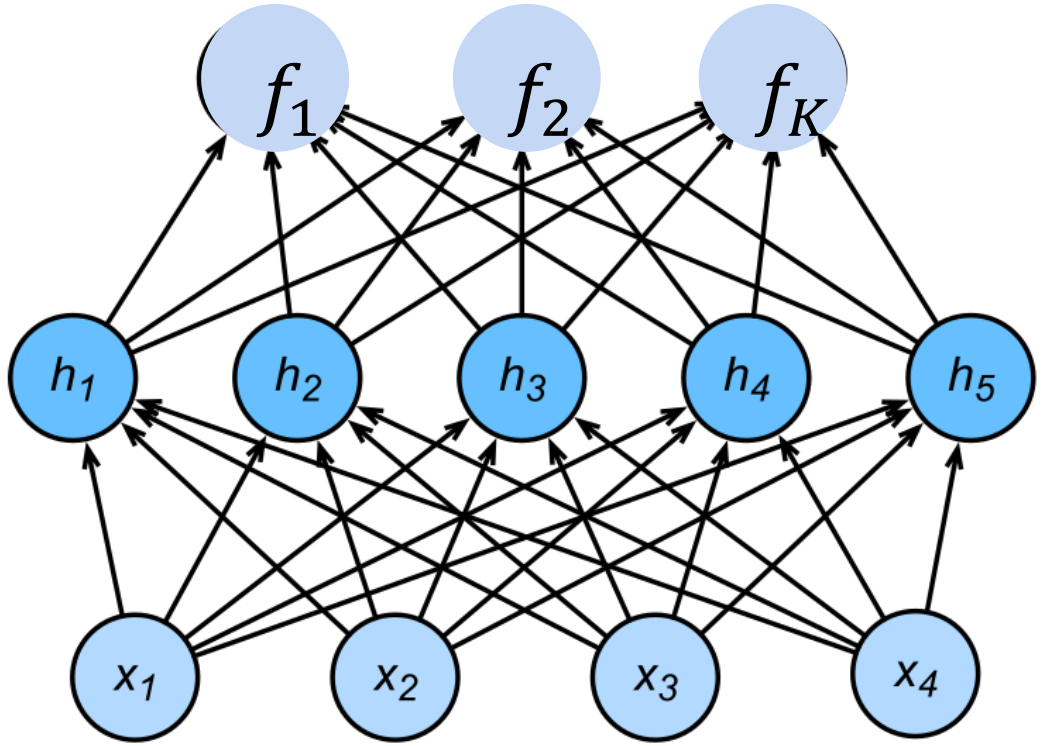
More complicated neural networks

$$p_1, p_2, \ldots, p_K = \text{softmax}(f_1, f_2, \ldots, f_K)$$

Output layer

Hidden layer

Input layer

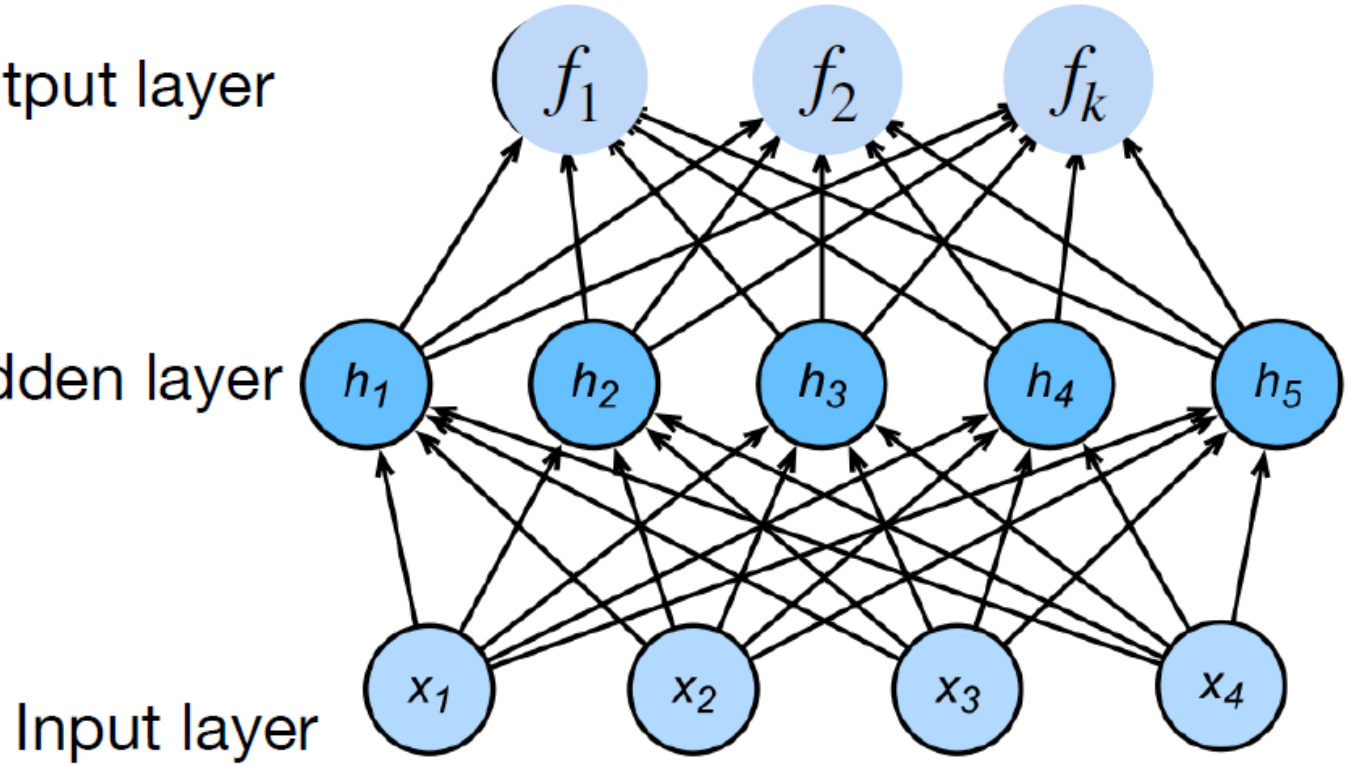


More complicated neural networks

- Input $\mathbf{x} \in \mathbb{R}^d$
- Hidden $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}, \mathbf{b}^{(1)} \in \mathbb{R}^m$

$$p_1, p_2, \ldots, p_K = \text{softmax}(f_1, f_2, \ldots, f_K)$$

$$\mathbf{h} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$
$$\mathbf{f} = \mathbf{W}^{(2)}\mathbf{h} + \mathbf{b}^{(2)}$$
$$\mathbf{p} = \text{softmax}(\mathbf{f})$$



More complicated neural networks: multiple hidden layers

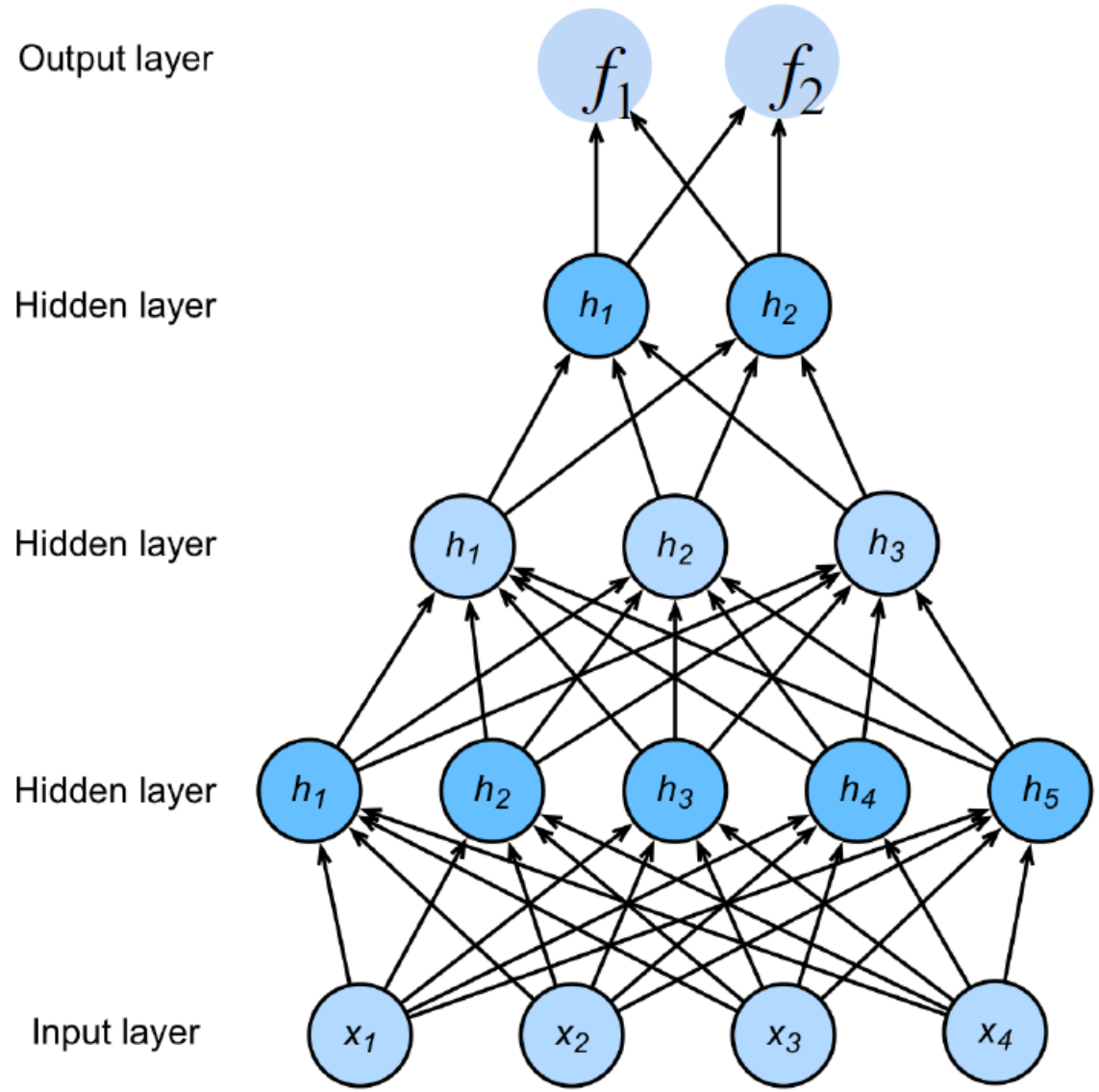
$$\mathbf{h}_1 = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\mathbf{h}_2 = \sigma(\mathbf{W}^{(2)}\mathbf{h}_1 + \mathbf{b}^{(2)})$$

$$\mathbf{h}_3 = \sigma(\mathbf{W}^{(3)}\mathbf{h}_2 + \mathbf{b}^{(3)})$$

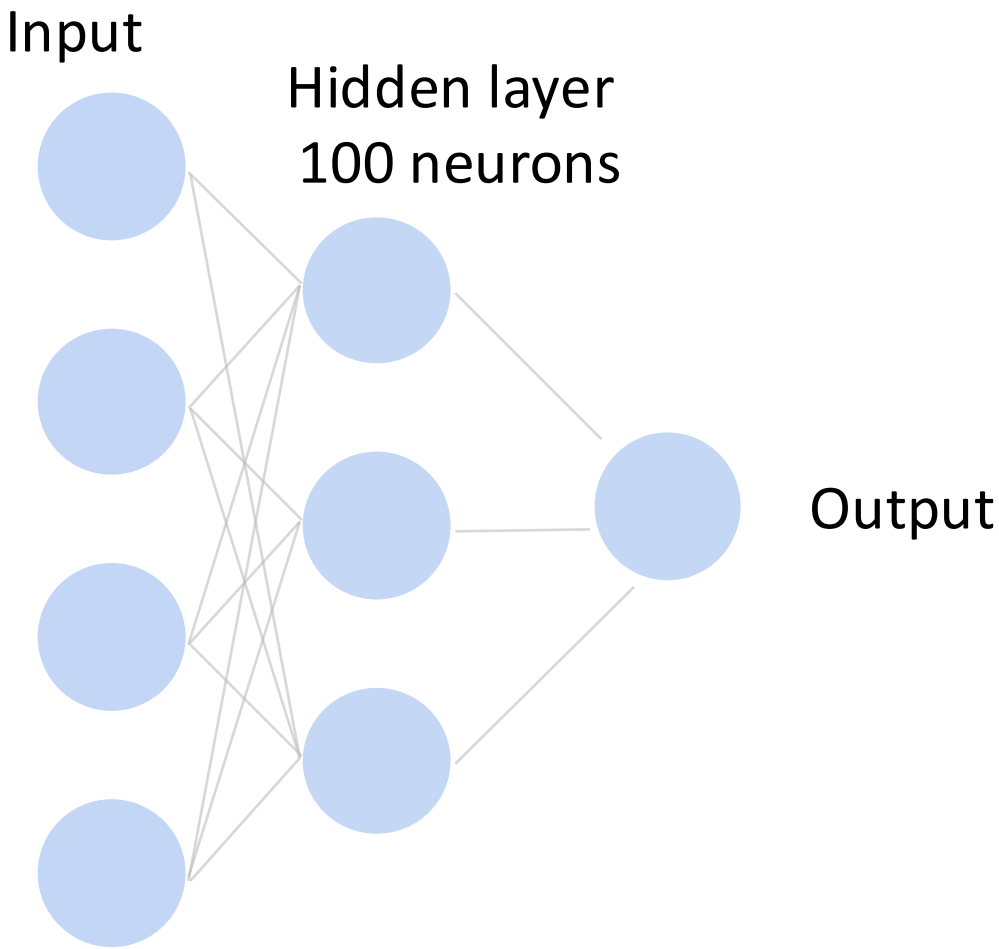
$$\mathbf{f} = \mathbf{W}^{(4)}\mathbf{h}_3 + \mathbf{b}^{(4)}$$

$$\mathbf{p} = \text{softmax}(\mathbf{f})$$



How to train a neural network?

Classify cats vs. dogs



How to train a neural network? Binary classification

$\mathbf{x} \in \mathbb{R}^d$ One training data point in the training set D

$\hat{y} \in [0,1]$ Model output for example **x**

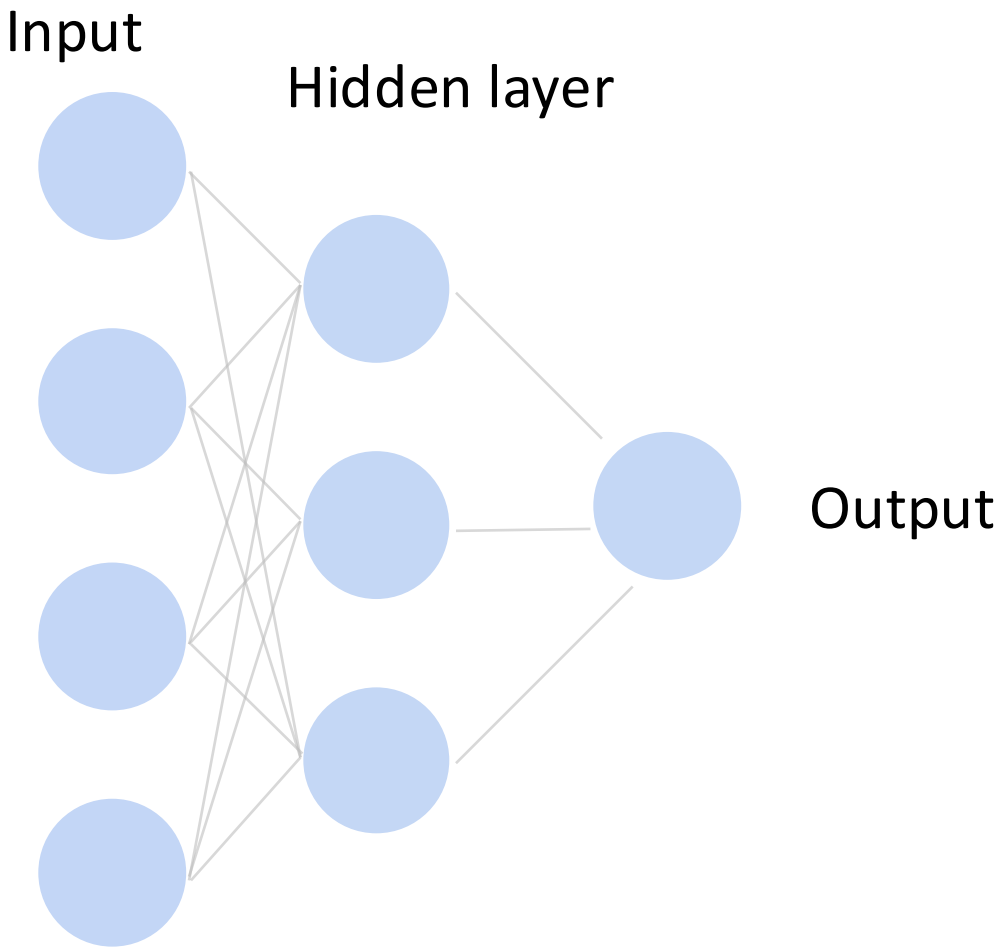
(This is a function of all weights W: $\hat{y} = g(W)$)

y Ground truth label for example **x**

Learning by matching the output to the label

We want $\hat{y} \to 1$ **when** $y = 1$,

and $\hat{y} \to 0$ **when** $y = 0$

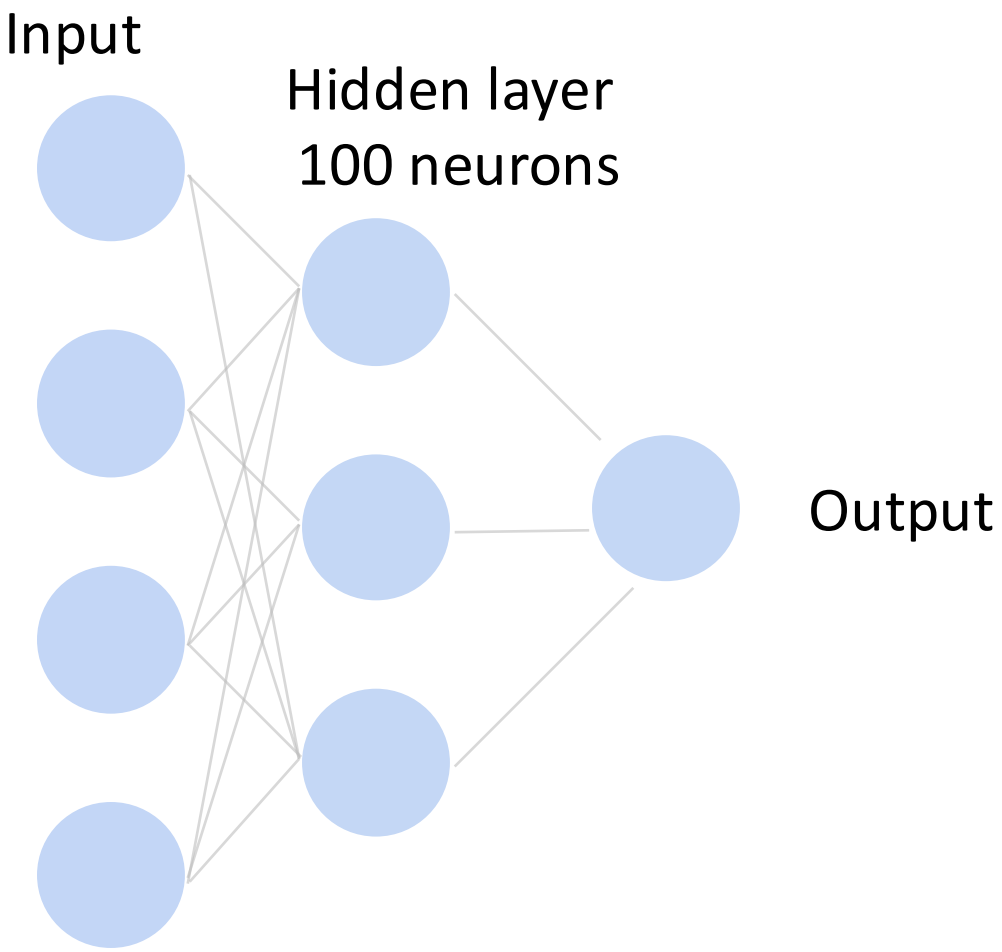


How to train a neural network? Binary classification

Loss function: $\frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \ell(\mathbf{x}, y)$

Per-sample loss:

$$\ell(\mathbf{x}, y) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$



Negative log likelihood

Also known as binary cross-entropy loss

- **We'll use all the time for LLMs**

How to train a neural network? Multiclass

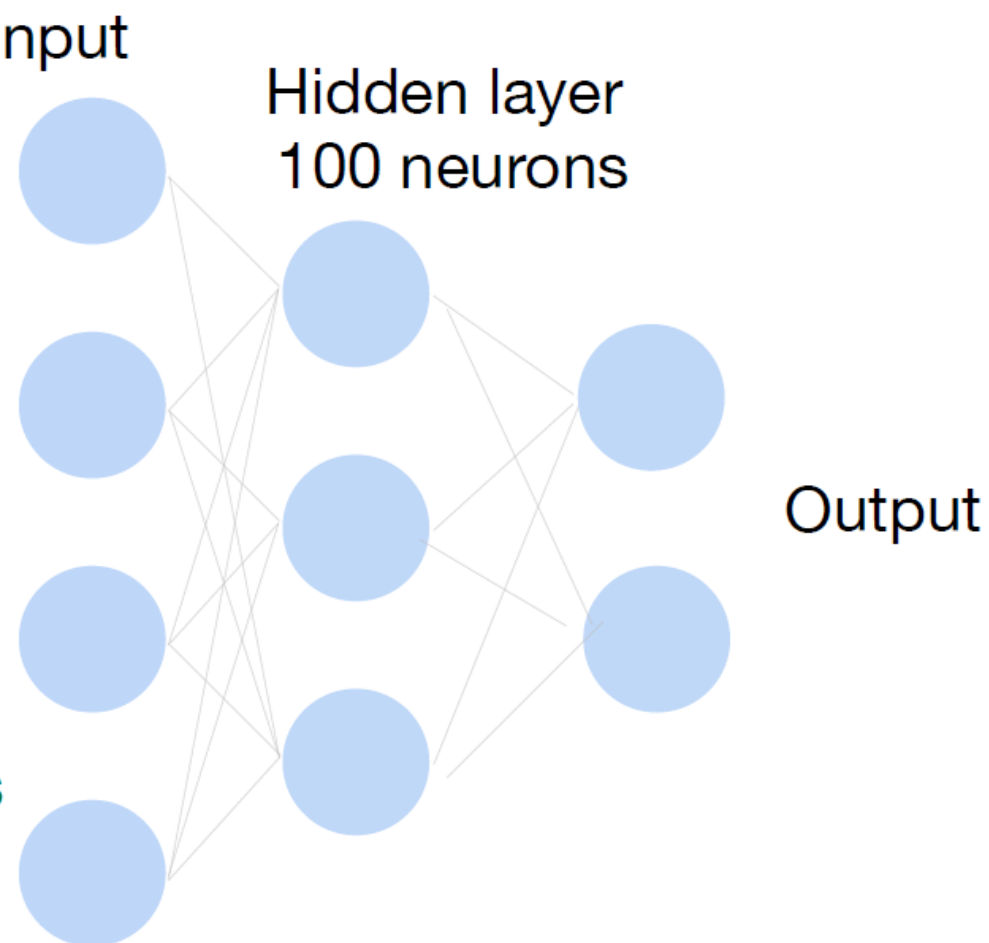
Loss function: $$\frac{1}{|D|}\sum_{(\mathbf{x},y)\in D}\ell(\mathbf{x},y)$$

Per-sample loss:

$$\ell(\mathbf{x},y)=\sum_{k=1}^{K}-Y_k\log p_k=-\log p_y$$

where Y is one-hot encoding of y

Also known as cross-entropy loss or softmax loss



Training Neural Networks

- Algorithm:
  - Get $D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$
  - Initialize weights
  - Until stopping criteria met,
    - For each training point $(x^{(i)}, y^{(i)})$
    - Compute: $f_{\text{network}}(x^{(d)})$ ← **Forward Pass**
    - Compute gradient: $\nabla L^{(i)}(w) = \left[\frac{\partial L^{(d)}}{\partial w_0}, \frac{\partial L^{(d)}}{\partial w_1}, \ldots, \frac{\partial L^{(d)}}{\partial w_m}\right]^T$ ← **Backward Pass**
    - Update weights: $w \leftarrow w - \alpha \nabla L^{(i)}(w)$

Break & Questions

Outline

- **Neural Networks**
  - Perceptrons, setup, training, limitations, multilayer perceptrons, loss functions (mostly from last time)
- **Convolutional Neural Networks**
  - Motivation, convolutional layers, CNN architectures
- **Sequence Models**
  - Recurrent neural networks, architecture, LSTMs, alternatives, training tricks

How to classify

Cats vs. dogs?

Dual

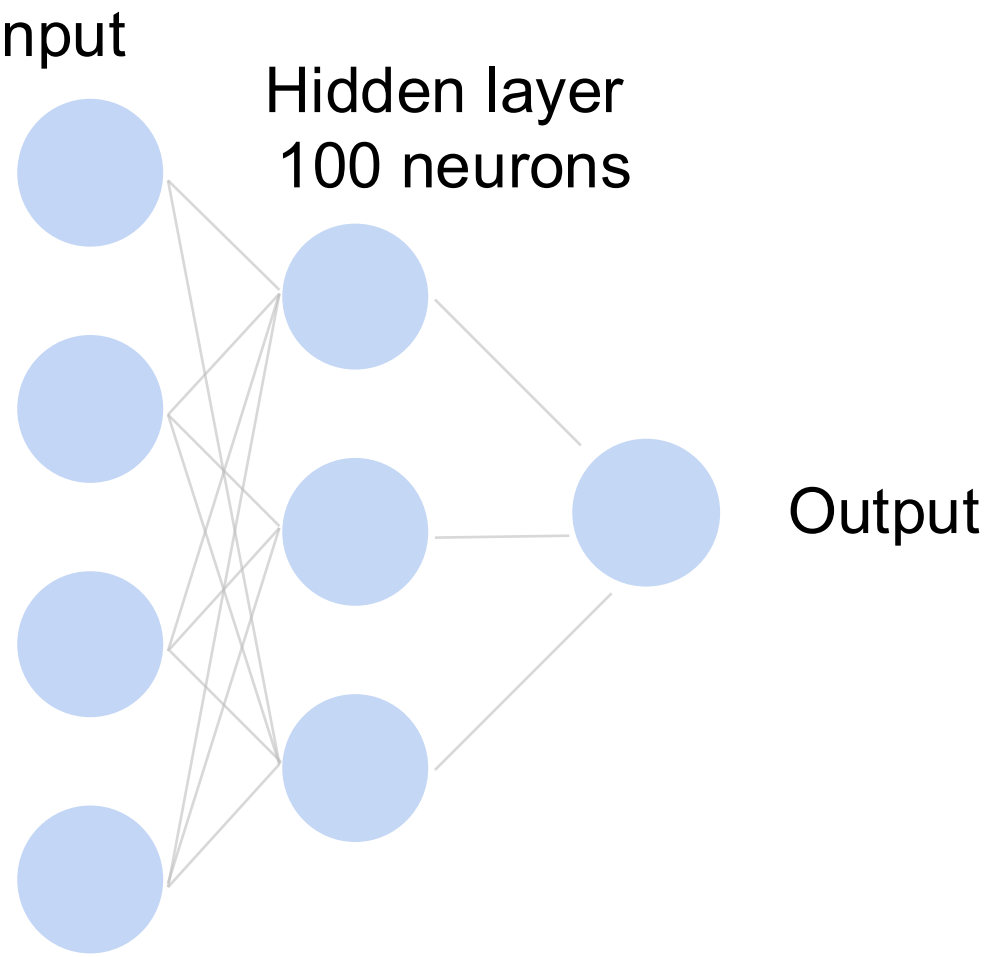
12MP

wide-angle and
telephoto cameras

36M floats in a RGB image!

Fully Connected Networks

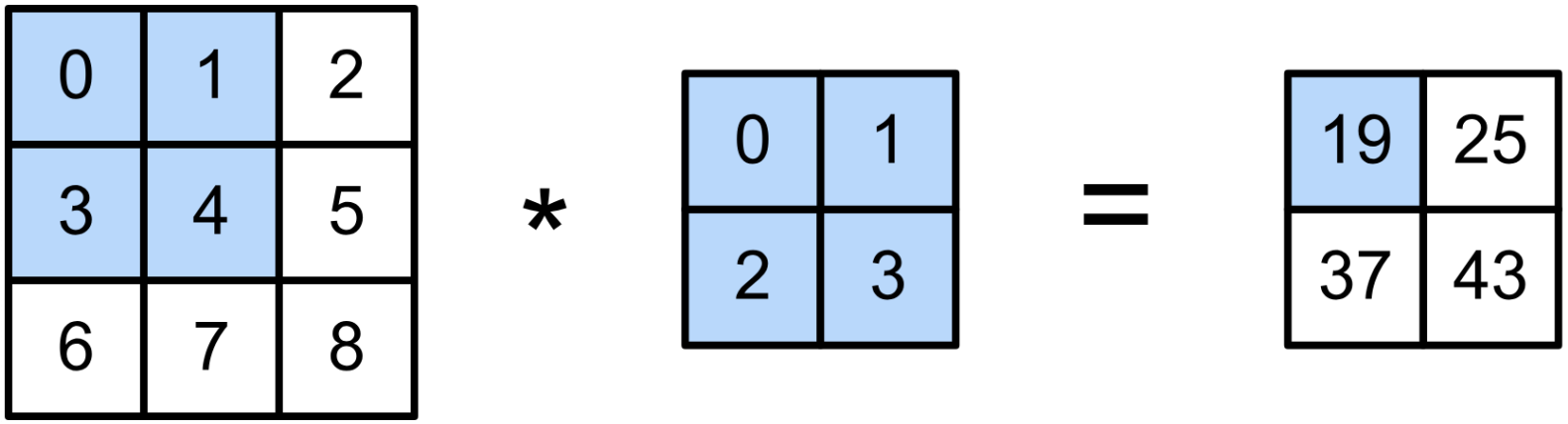
Cats vs. dogs?



~ 36M elements x 100 = **~3.6B** parameters!

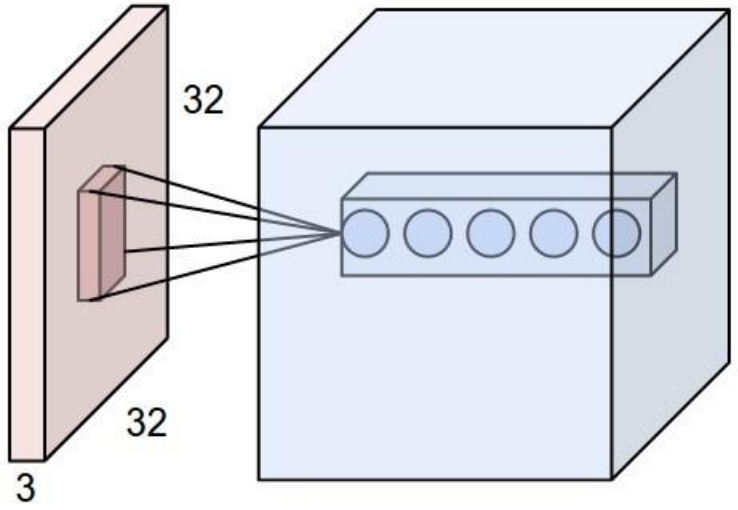
Neural Networks: Convolution Layers

- Notation:
  - *X:* n_h x n_w input matrix
  - *W:* k_h x k_w kernel matrix
  - *b* : bias (a scalar)
  - *Y*: () x () output matrix
- As usual *W*, *b* are learnable parameters



Neural Networks: Convolution NNs

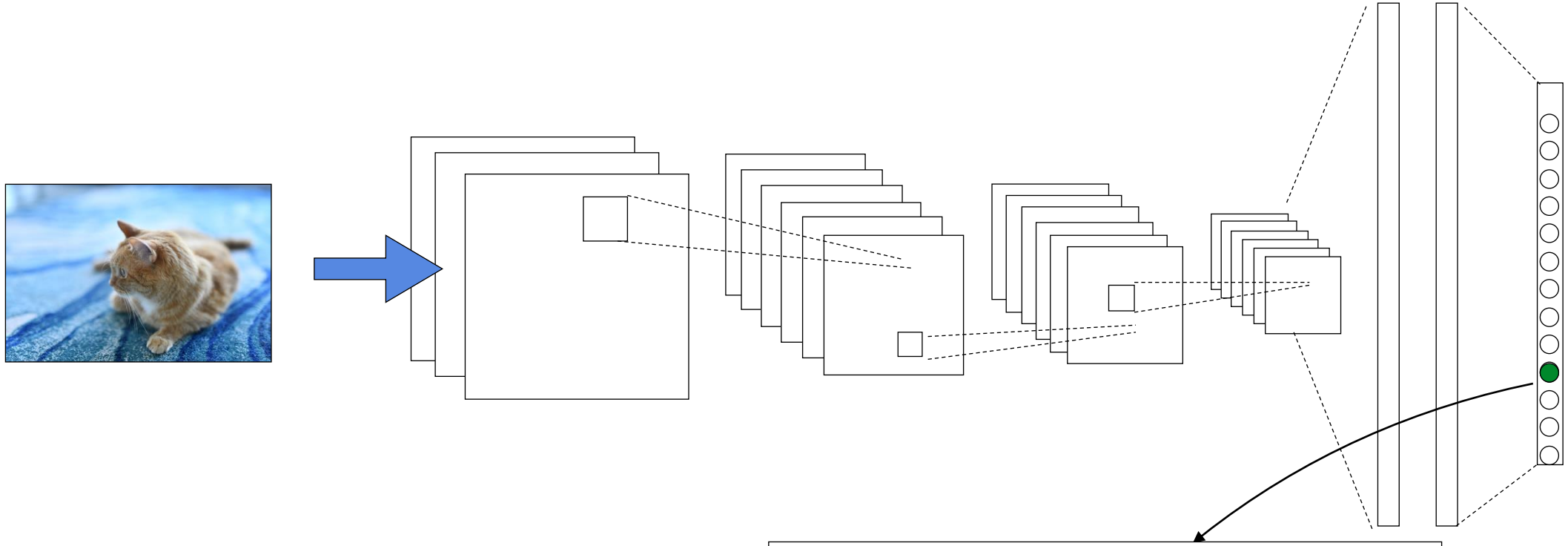
- Properties
  - Input: volume c_i x n_h x n_w (channels x height x width)
  - Hyperparameters: # of kernels/filters c_o, size k_h x k_w, stride s_h x s_w, zero padding p_h x p_w
  - Output: volume c_o x m_h x m_w (channels x height x width)
  - Parameters: k_h x k_w x c_i per filter, total $(k_h$ x k_w x $c_i)$ x c_o



Stanford CS 231n

Training a CNN

- Q: so we have a bunch of layers. How do we train?
- A: same as before. Apply softmax at the end, use backprop.

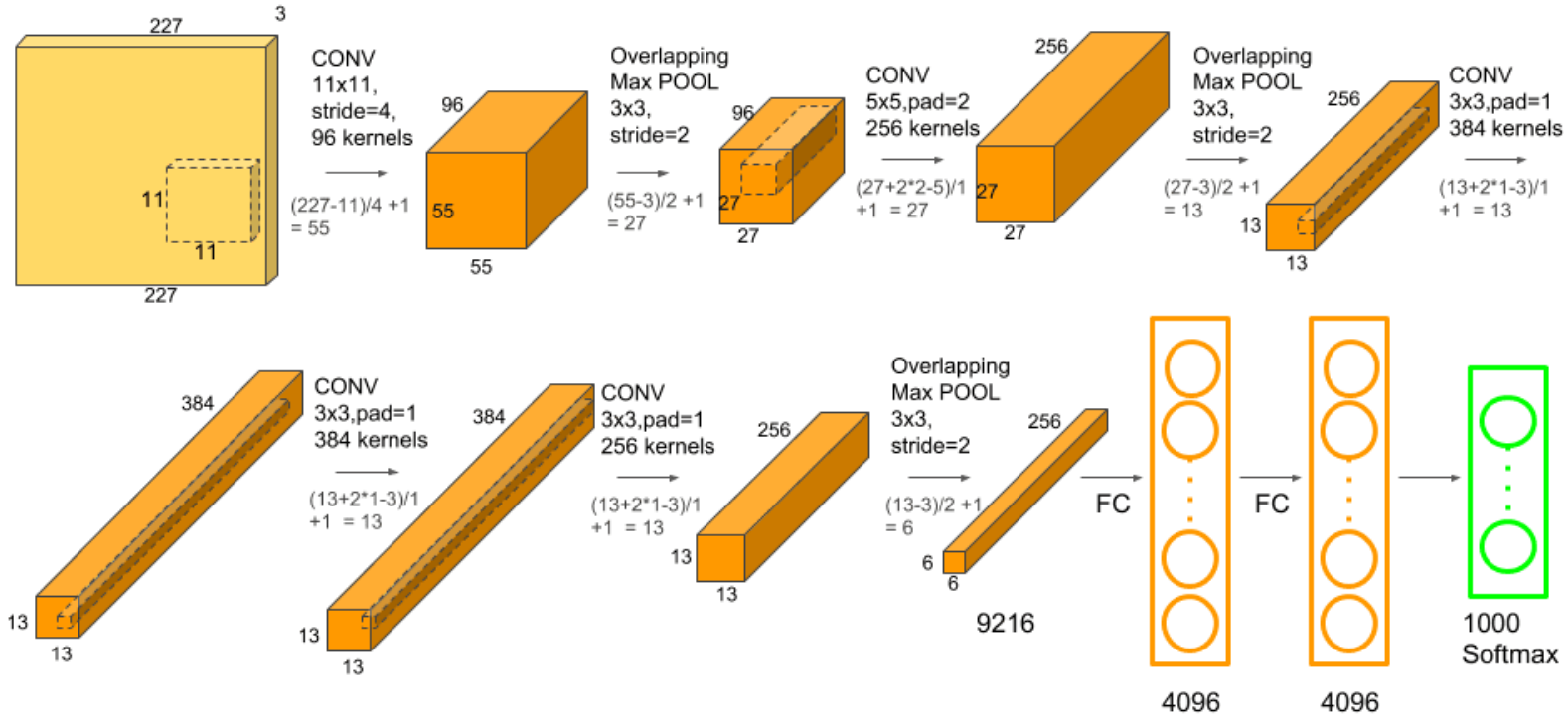


$$p_i(\boldsymbol{x}) = \frac{\exp\left(f_i(\boldsymbol{x})\right)}{\sum_{j=1}^{N} \exp\left(f_j(\boldsymbol{x})\right)},$$

softmax

CNN Architectures: AlexNet

- First of the major advancements: AlexNet
- Wins 2012 ImageNet competition
- Major trends: deeper, bigger LeNet

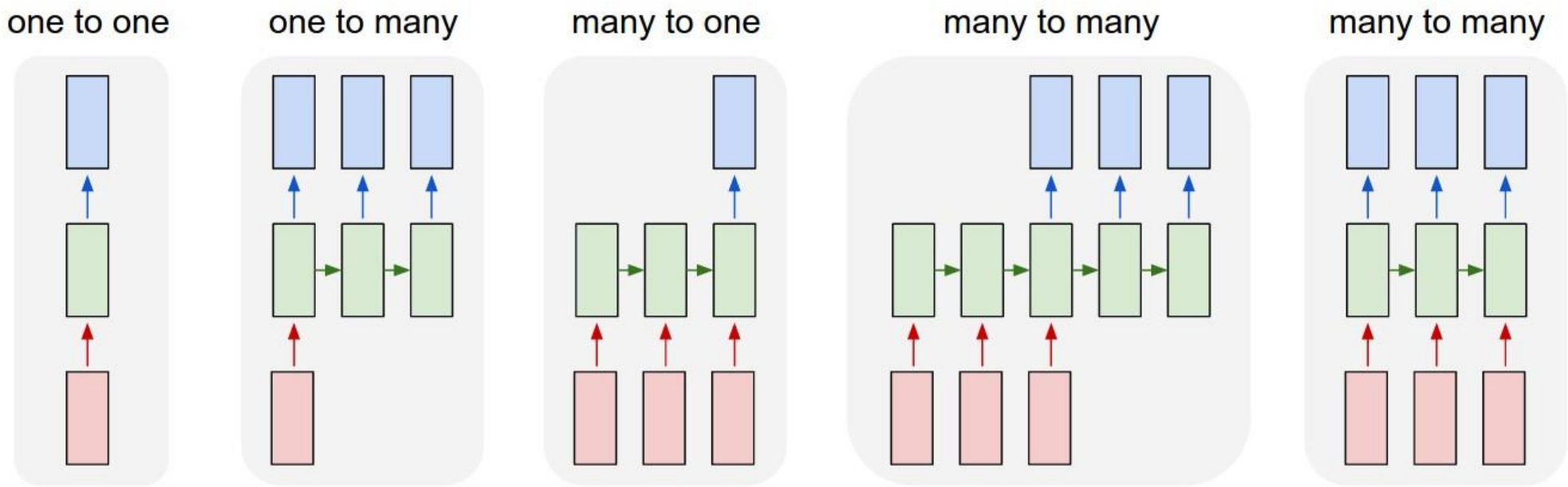


Break & Questions

Outline

- **Neural Networks**
  - Perceptrons, setup, training, limitations, multilayer perceptrons, loss functions (mostly from last time)
- **Convolutional Neural Networks**
  - Motivation, convolutional layers, CNN architectures
- **Sequence Models**
  - Recurrent neural networks, architecture, LSTMs, alternatives, training tricks

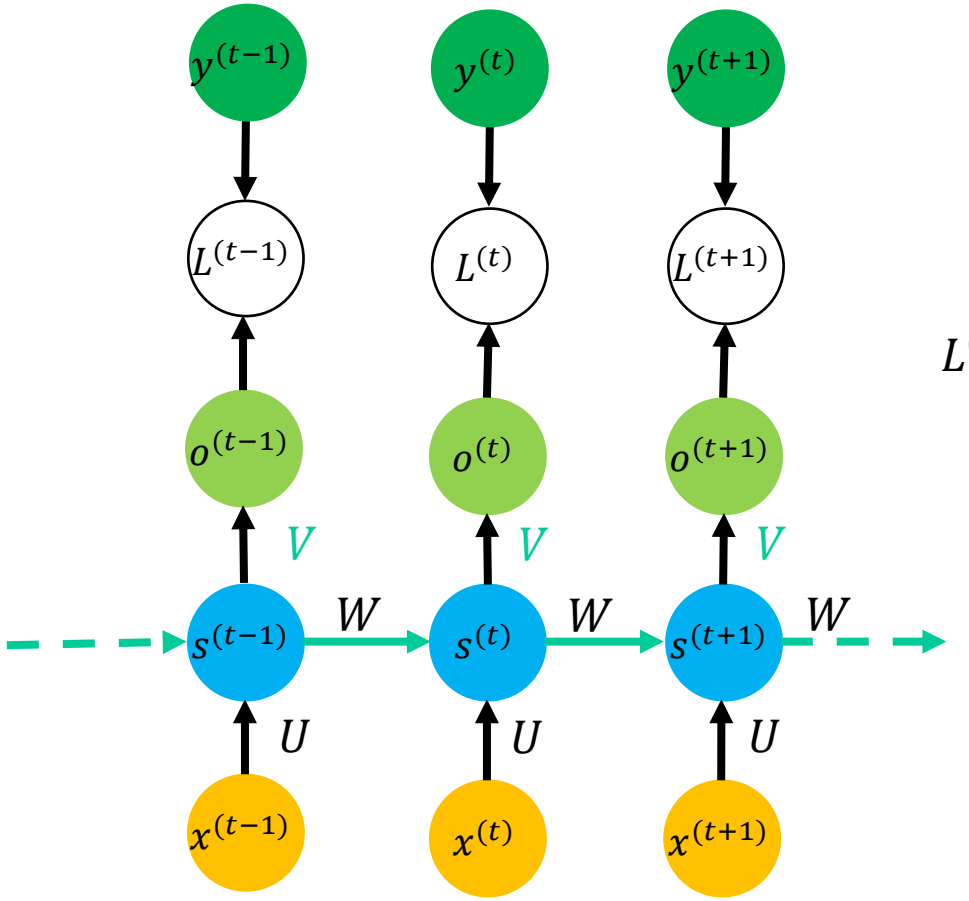
Tasks We Can Handle with NNs?



- Mostly talked about (1) so far
  - Others: need a new kind of model

Neural Networks: Simple RNNs

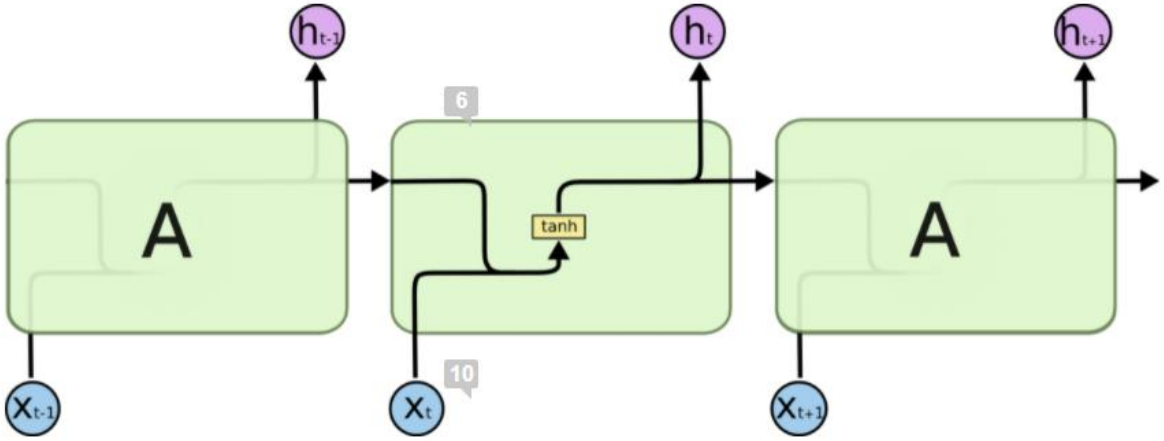
- Classical RNN variant:



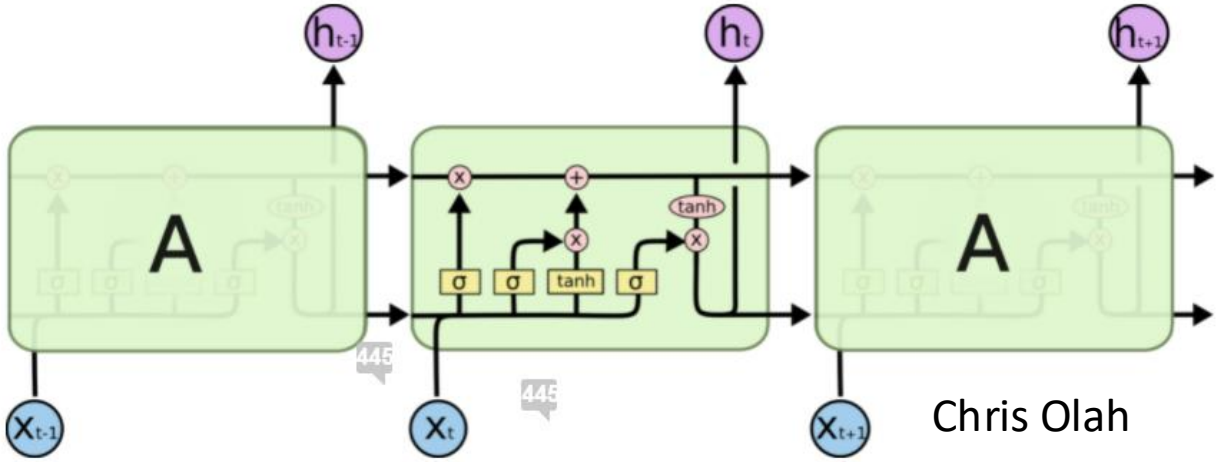
$$a^{(t)} = b + Ws^{(t-1)} + Ux^{(t)}$$
$$s^{(t)} = \tanh\left(a^{(t)}\right)$$
$$o^{(t)} = c + Vs^{(t)}$$
$$\hat{y}^{(t)} = \text{softmax}\left(o^{(t)}\right)$$
$$L^{(t)} = \text{CrossEntropy}(y^{(t)}, \hat{y}^{(t)})$$

Neural Networks: LSTMs

- RNN: can write structure as:



- Long Short-Term Memory: deals with problem. Cell:

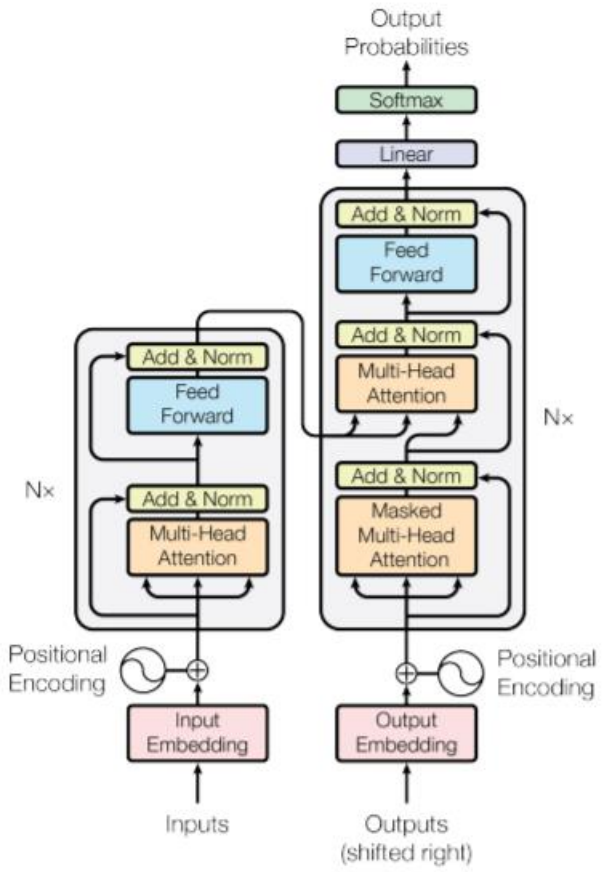


Chris Olah

Neural Networks: Transformers

- Initial goal for an architecture: **encoder-decoder**
  - Get **rid of recurrence**
  - Replace with **self-attention**

- Architecture
  - The famous picture you've seen
  - Centered on self-attention blocks



Vaswani et al. '17

Data Augmentation

Augmentation: transform + add new samples to dataset

- Transformations: based on domain
- Idea: build **invariances** into the model
  - **Ex**: if all images have same alignment, model learns to use it
- Keep the label the same!

Data Augmentation: Examples

Examples of transformations for images

- **Crop** (and zoom)
- **Color** (change contrast/brightness)
- **Rotations+** (translate, stretch, shear, etc)

Many more possibilities. Combine as well!

Q: how to deal with this at **test time**?

- A: transform, test, average