ECE595 / STAT598: Machine Learning I

Lecture 18.1: Multi-Layer Perceptron

Spring 2020

Stanley Chan

School of Electrical and Computer Engineering
Purdue University

PURDUE
UNIVERSITY

Outline

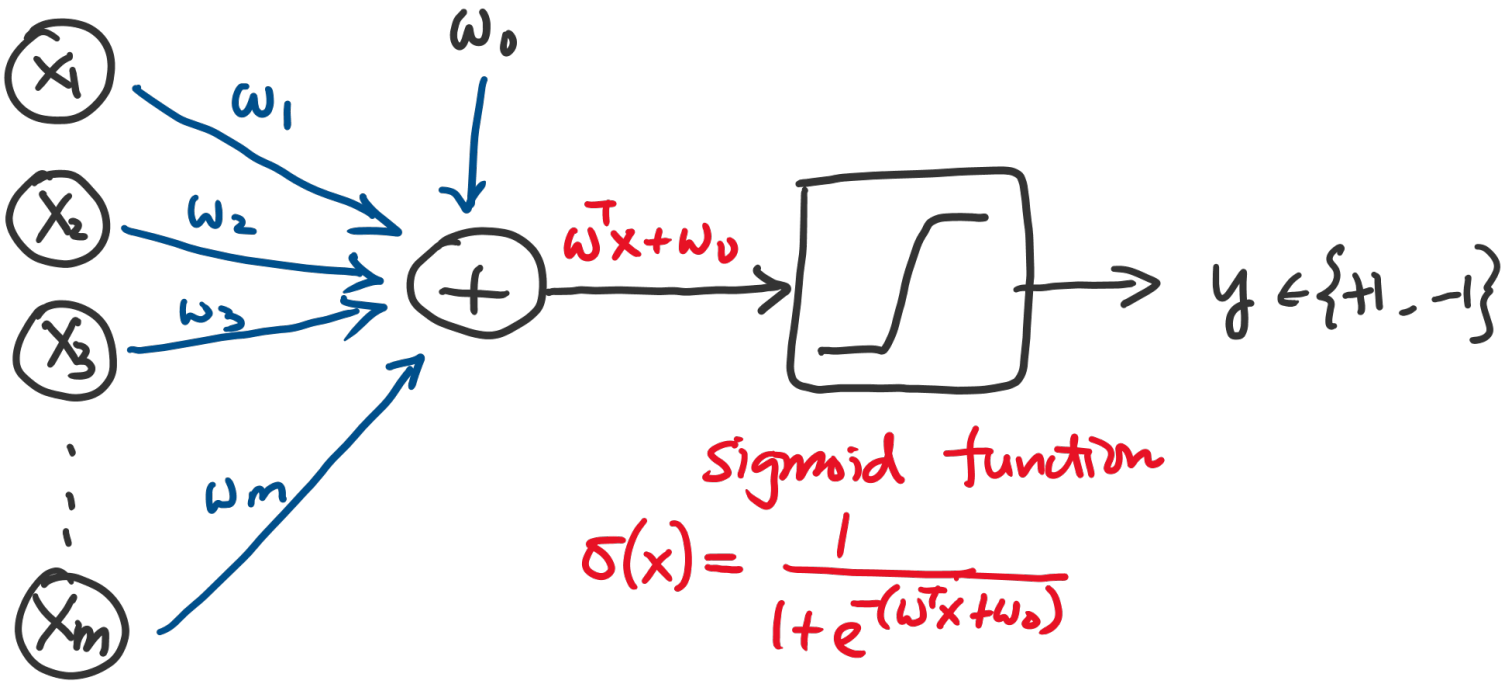
Discriminative Approaches

- Lecture 16 Perceptron 1: Definition and Basic Concepts
- Lecture 17 Perceptron 2: Algorithm and Property
- Lecture 18 Multi-Layer Perceptron: Back Propagation

This lecture: Multi-Layer Perceptron: Back Propagation

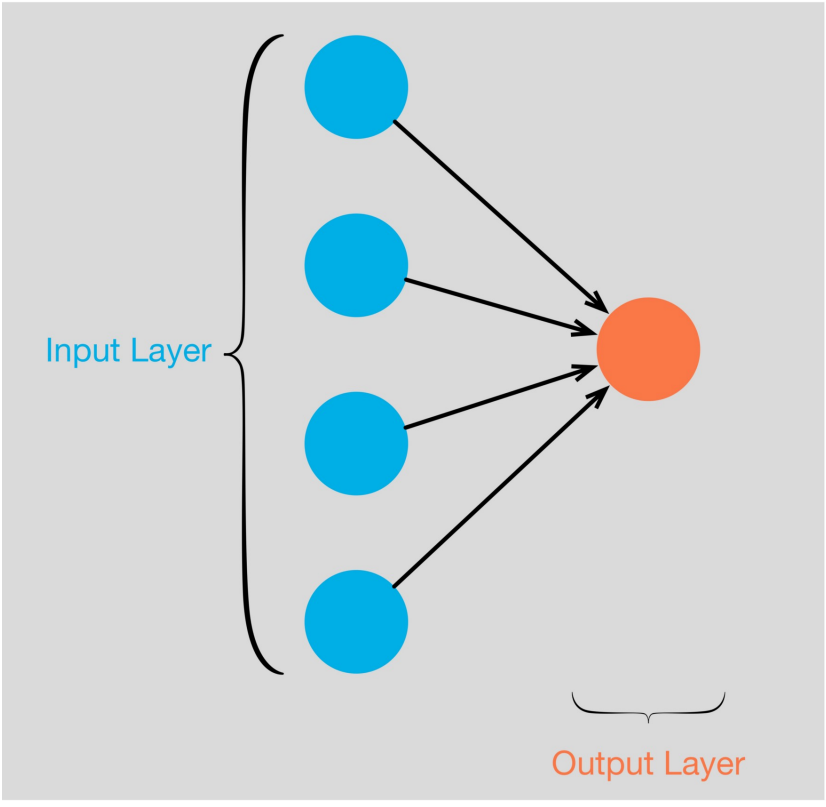
- Multi-Layer Perceptron
  - Hidden Layer
  - Matrix Representation
- Back Propagation
  - Chain Rule
  - 4 Fundamental Equations
  - Algorithm
  - Interpretation

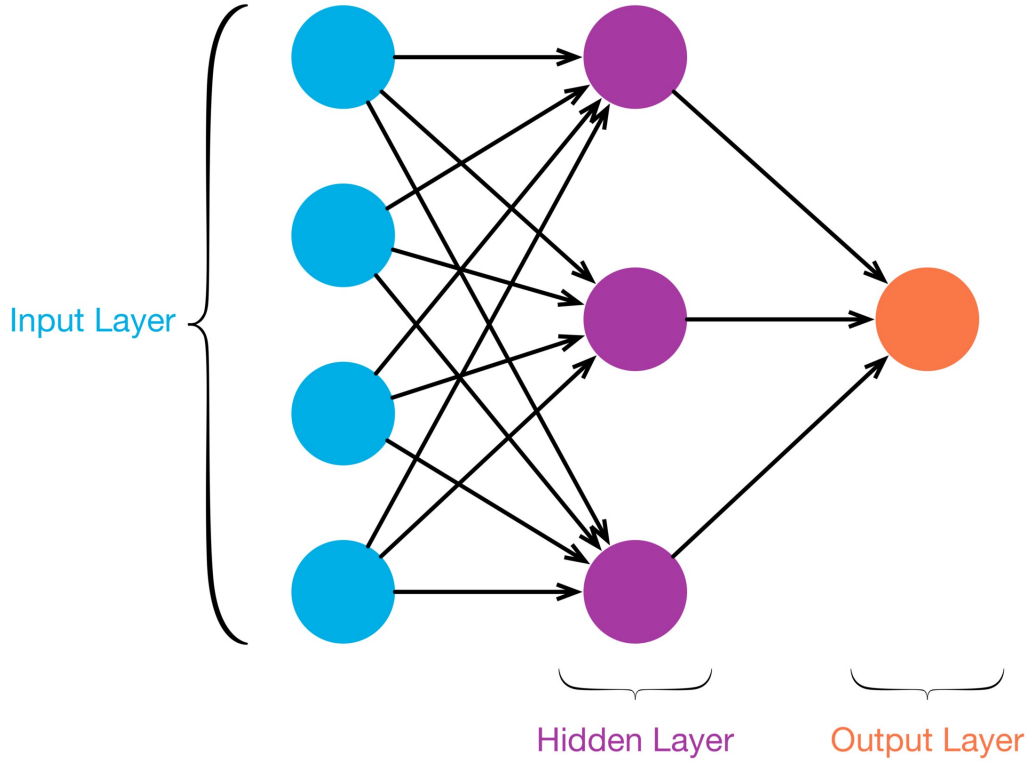
Single-Layer Perceptron



- Input neurons $\boldsymbol{x}$
- Weights $\boldsymbol{w}$
- Predicted label $= \sigma(\boldsymbol{w}^T \boldsymbol{x} + w_0)$.

Multi-Layer Network

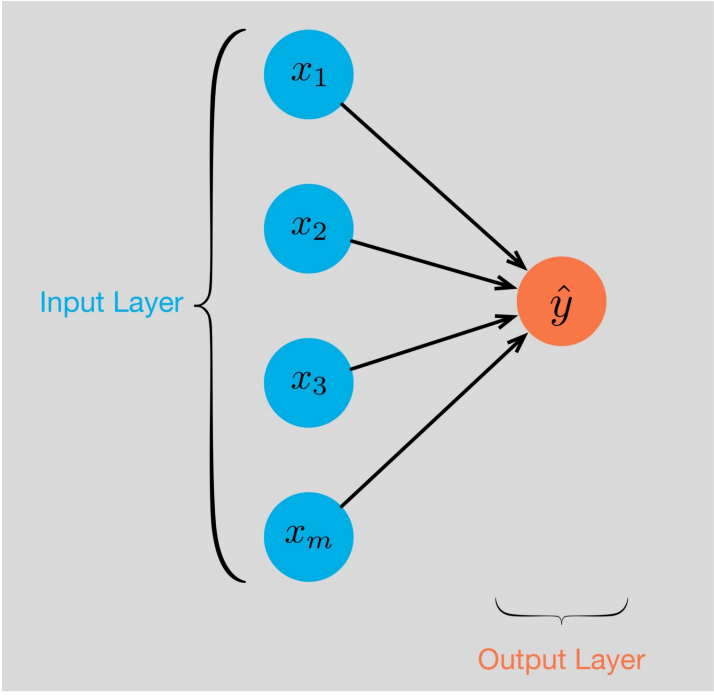


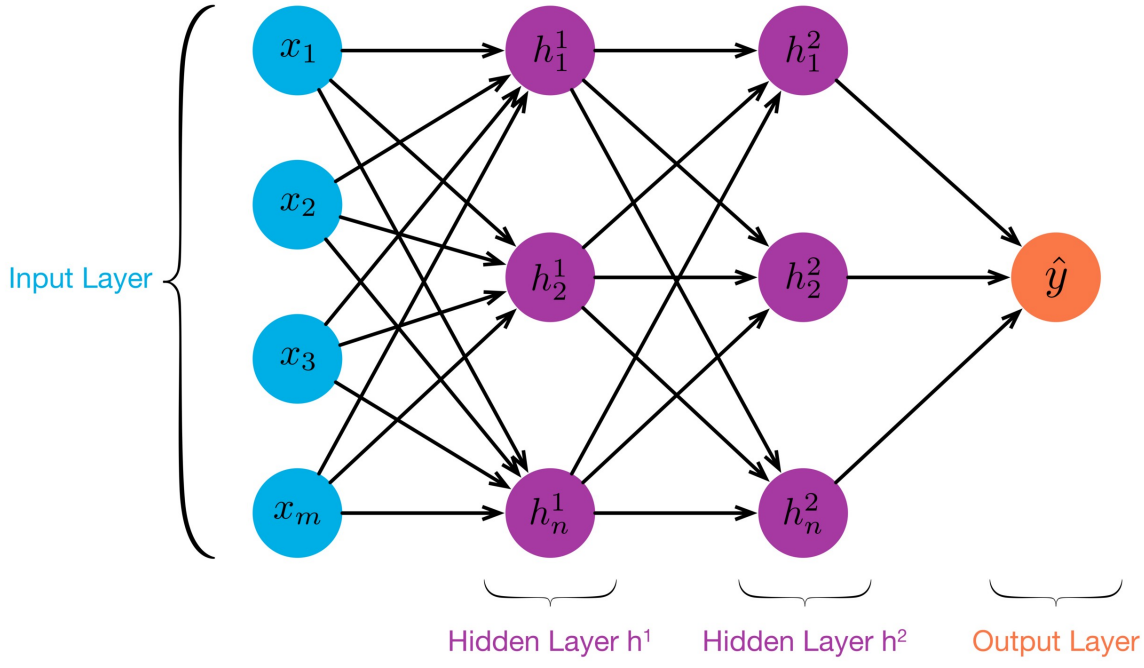


https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

- Introduce a layer of **hidden** neurons
- So now you have two sets of weights: from input to hidden, and from hidden to output

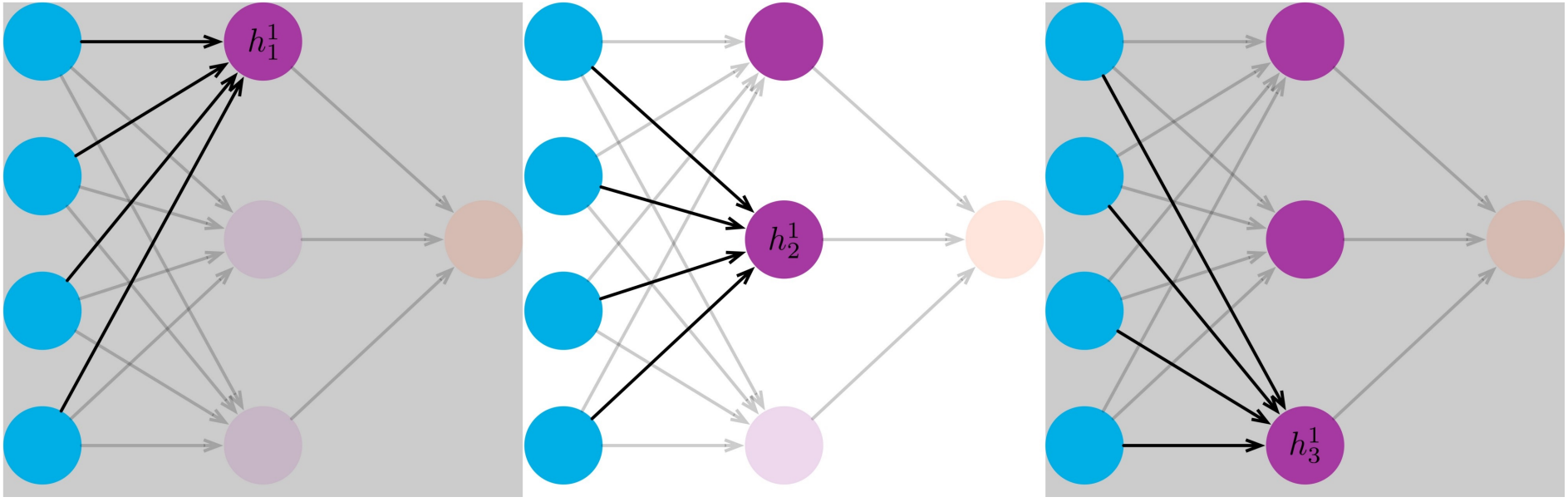
Many Hidden Layers





- You can introduce as many hidden layers as you want.
- Every time you add a hidden layer, you add a set of weights.

Understanding the Weights



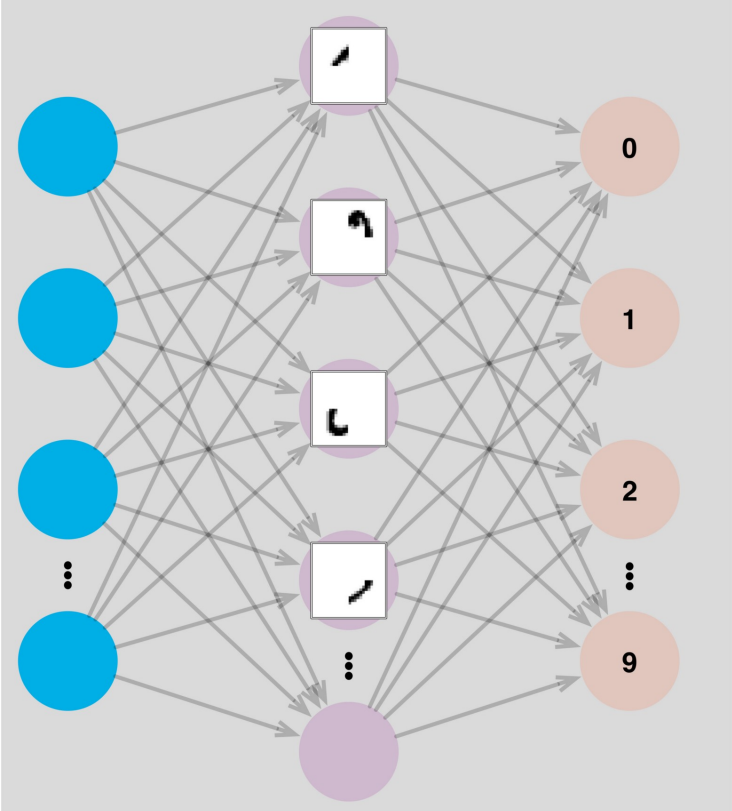
- Each hidden neuron is an **output** of a perceptron
- So you will have

$$\begin{bmatrix} h_1^1 \\ h_2^1 \\ \dots \\ h_n^1 \end{bmatrix} = \begin{bmatrix} w_{11}^1 & w_{12}^1 & \dots & w_{1n}^1 \\ w_{21}^1 & w_{22}^1 & \dots & w_{2n}^1 \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1}^1 & w_{m2}^1 & \dots & w_{mn}^1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

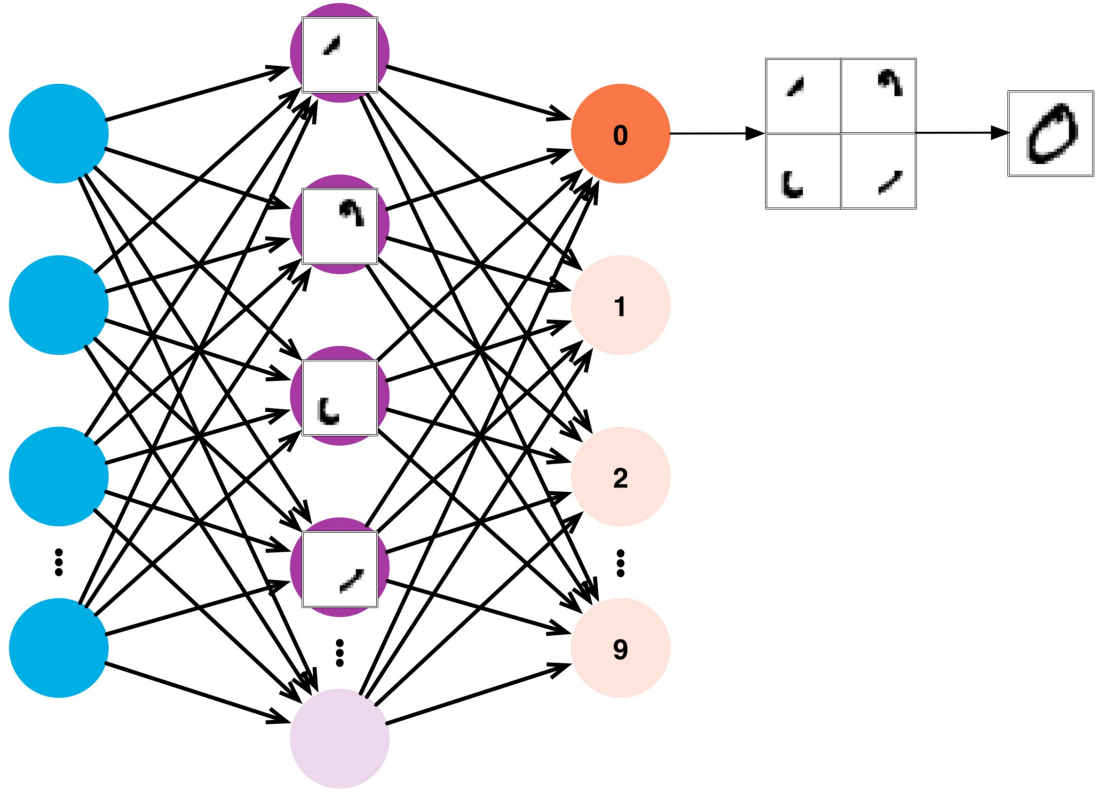
Progression to DEEP (Linear) Neural Networks

- Single-layer:
$$h = \boldsymbol{w}^T \boldsymbol{x}$$
- Hidden-layer:
$$\boldsymbol{h} = \boldsymbol{W}^T \boldsymbol{x}$$
- Two Hidden Layers:
$$\boldsymbol{h} = \boldsymbol{W}_2^T \boldsymbol{W}_1^T \boldsymbol{x}$$
- Three Hidden Layers:
$$\boldsymbol{h} = \boldsymbol{W}_3^T \boldsymbol{W}_2^T \boldsymbol{W}_1^T \boldsymbol{x}$$
- A LOT of Hidden Layers:
$$\boldsymbol{h} = \boldsymbol{W}_N^T \ldots \boldsymbol{W}_2^T \boldsymbol{W}_1^T \boldsymbol{x}$$

Interpreting the Hidden Layer



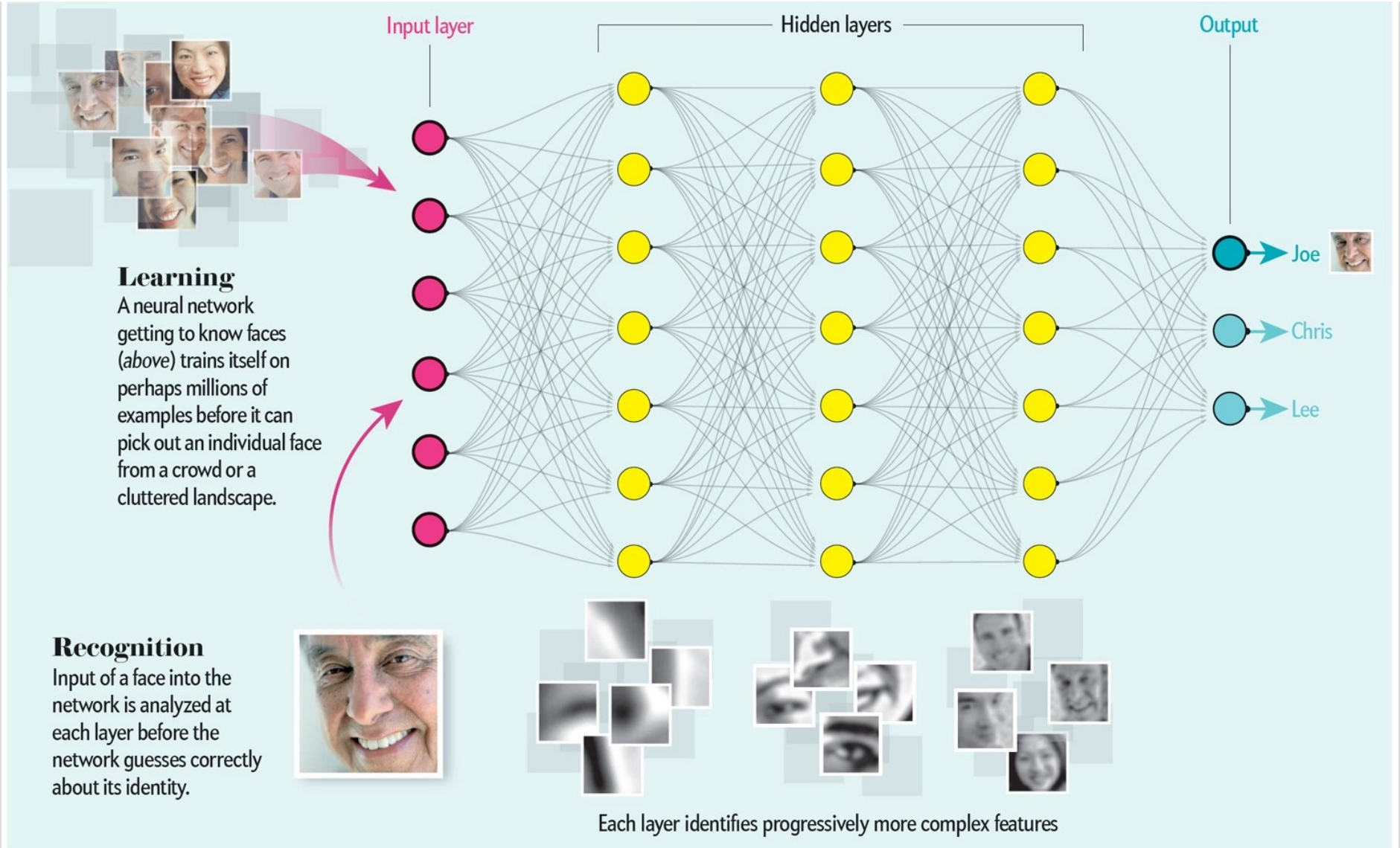
A Trained Neural Network in which each hidden neuron classifies certain pattern



Feeding a handwritten digit of 0 should trigger the 4 hidden layer neurons, and then the first output neuron

- Each hidden neuron is responsible for certain features.
- Given an object, the network identifies the most likely features.

Interpreting the Hidden Layer



https://www.scientificamerican.com/article/springtime-for-ai-the-rise-of-deep-learning/

Two Questions about Multi-Layer Network

- How do we **efficiently** learn the weights?
  - Ultimately we need to minimize the loss

$$J(\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L) = \sum_{i=1}^{N} \| \boldsymbol{W}_L^T \ldots \boldsymbol{W}_2^T \boldsymbol{W}_1^T \boldsymbol{x}_i - \boldsymbol{y}_i \|^2$$

  - One layer: Gradient descent. Multi-layer: Also gradient descent, also known as Back propagation (BP) by Rumelhart, Hinton and Williams (1986)
  - Back propagation = Very careful book-keeping and chain rule
- What is the optimization **landscape**?
  - Convex? Global minimum? Saddle point?
  - Two-layer case is proved by Baldi and Hornik (1989)
  - All local minima are global.
  - A critical point is either a saddle point or global minimum.
  - L-layer case is proved by Kawaguchi (2016). Also proved L-layer nonlinear network (with sigmoid between adjacent layers.)