Let's Build a (really, really, really) Simple Neural Network

Hana Lee
Zagaku
6 February 2018

What is a neural network?

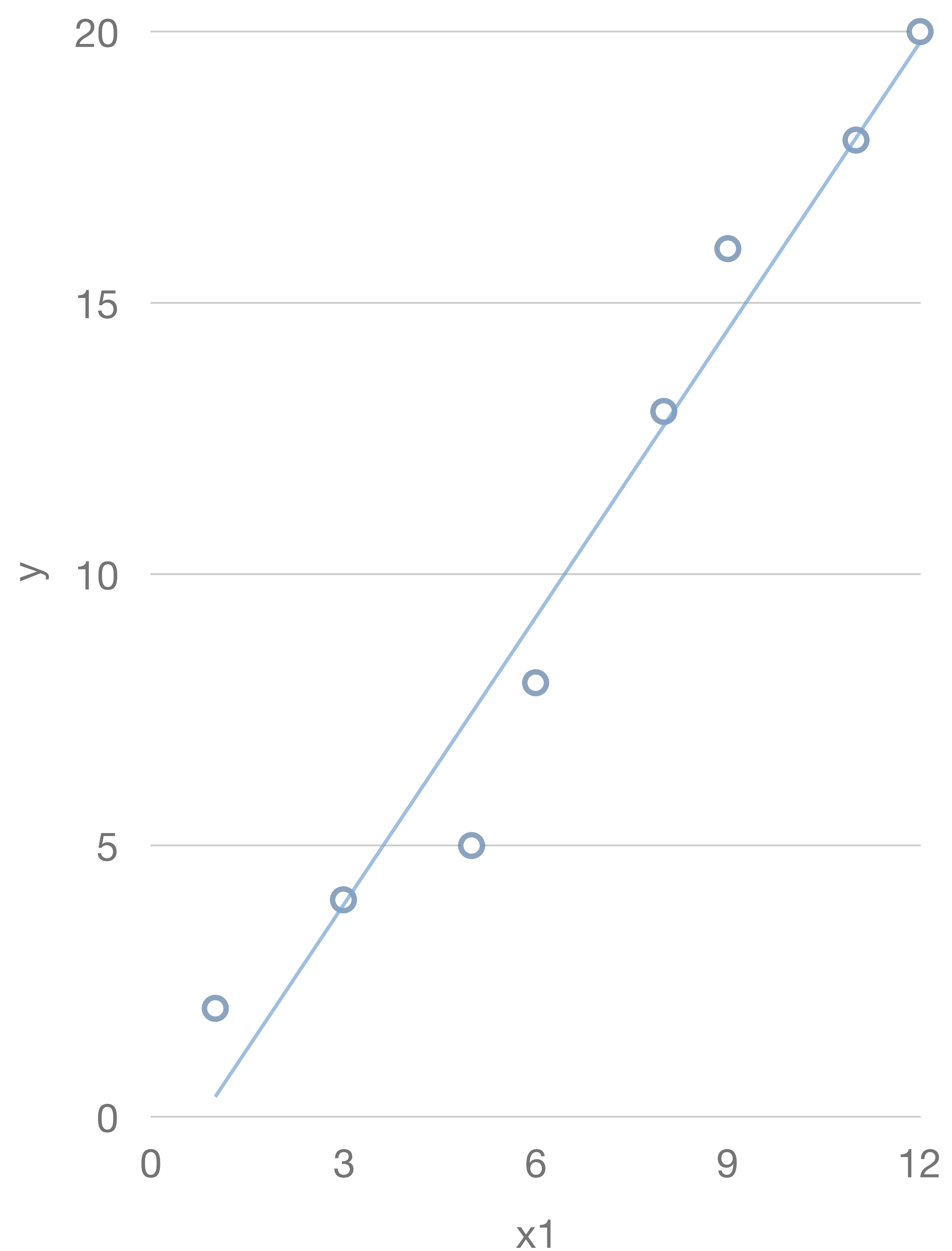
~~What is a neural network?~~

What is a model?

What is a model?

- A **model** is a mathematical function that takes input variables and tries to predict the value of an output variable
- Simplest example:

$$y = a_0 + a_1 x_1$$



$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots + a_n x_n$$

weights

output

inputs

Output can be a vector: $Y = [\, y_1, y_2, y_3, \ldots, y_n \,]$

Inputs can also be vectors: $X_1 = [\, x_1, x_2, x_3, \ldots, x_n \,]$

Polynomial rather than linear: $a_{11}x_1 + a_{12}x_1^2 + a_{13}x_1^3 + \ldots + a_{1n}x_1^n$

Interactions between inputs: $a_{10}x_1 + a_{11}x_1x_2 + a_{01}x_2$

Cost function

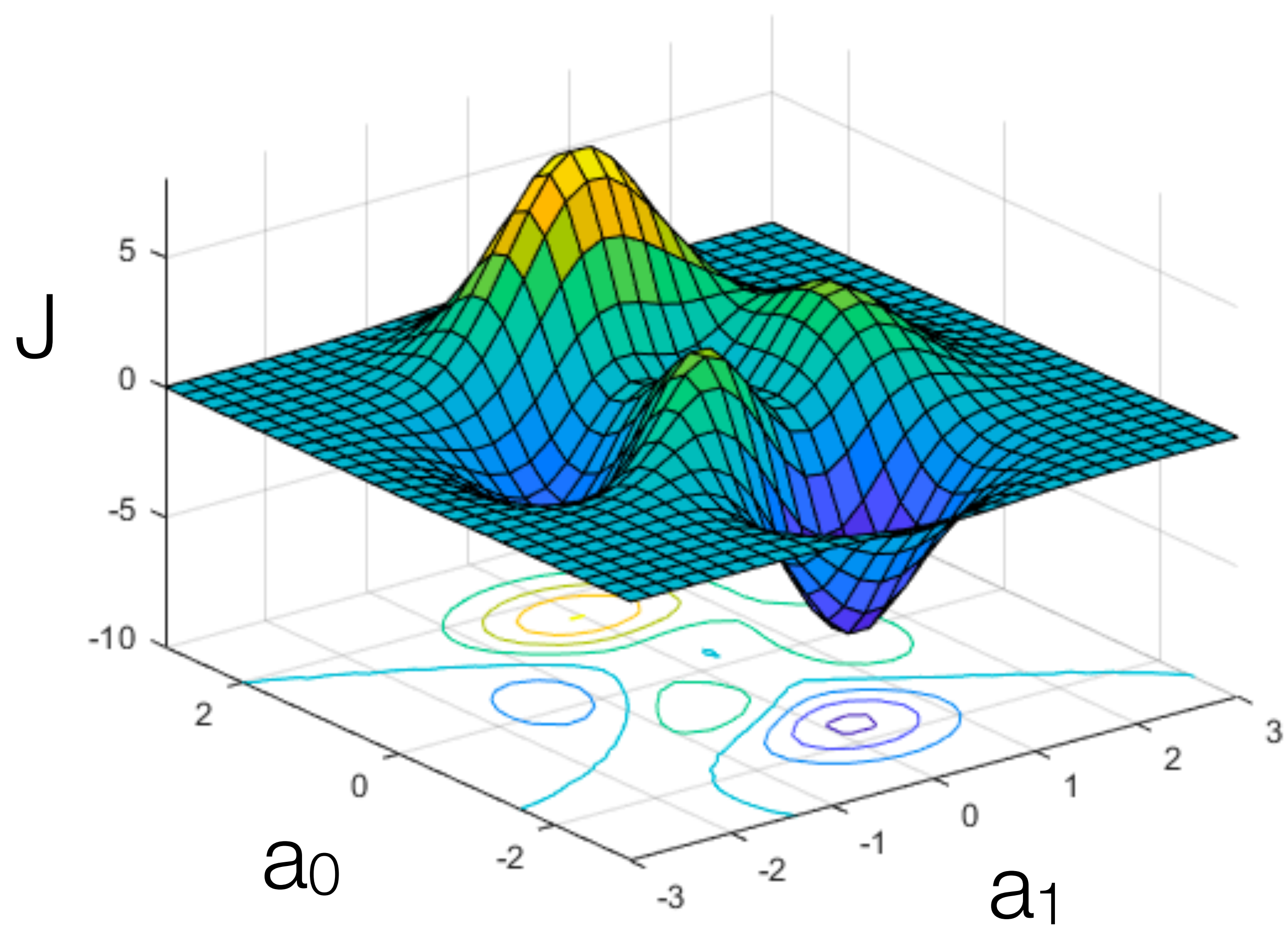
- Used to evaluate how well a model fits the data
- Many possible methods:
  - root mean squared error (RMSE),
  - log-loss
  - area under receiver operating characteristic (ROC) curve
  - …and many more
- Mental shortcut: **reality - prediction**

Training a model

- Requires **training set** data with output (i.e. “reality”) associated with given inputs
- Find the set of **weights** for the **model** that minimizes value of the **cost function**
- An optimization algorithm (just like minimax)

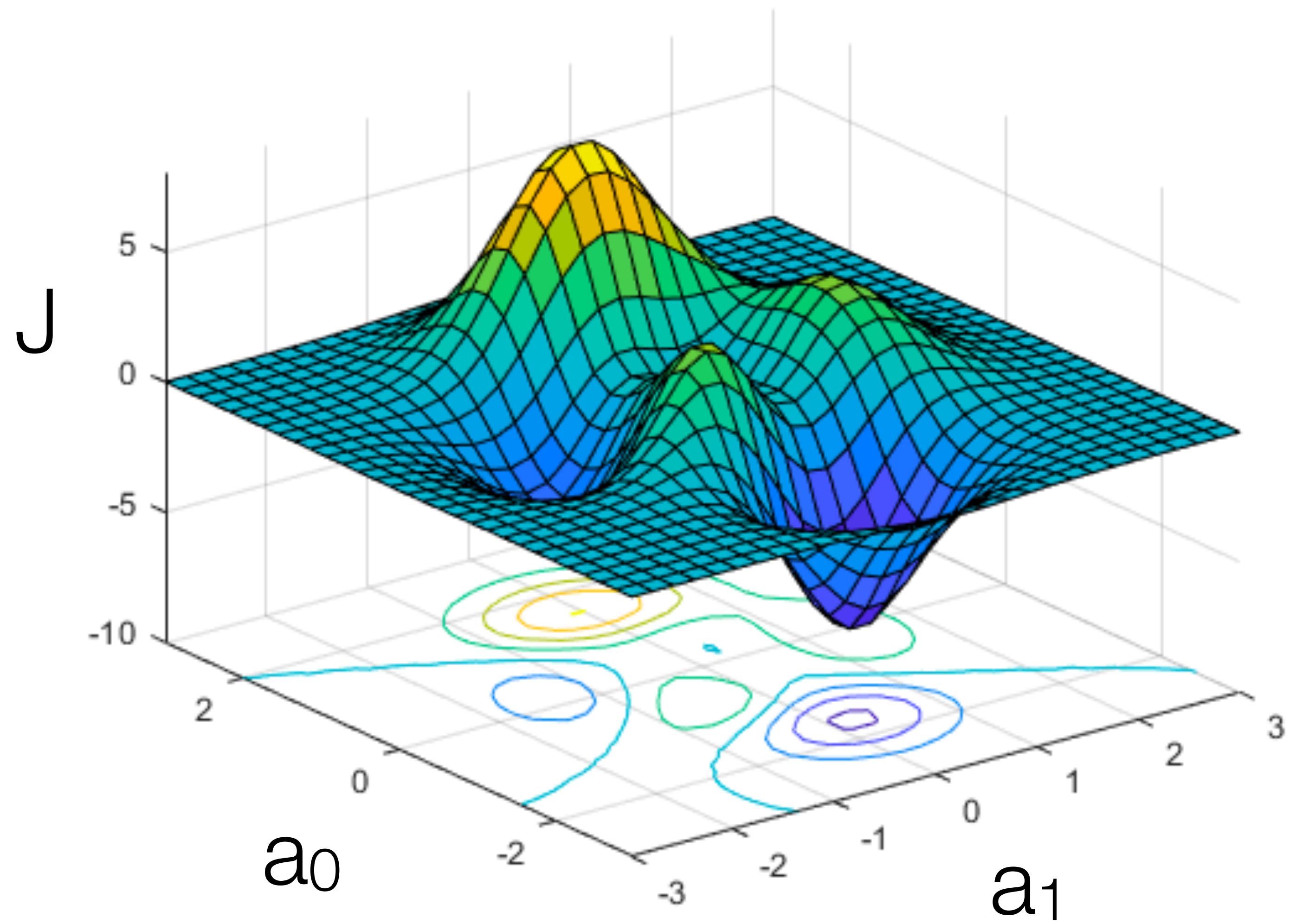
Minimizing the cost function

- Model: $y = a_0 + a_1 x_1$
- Cost function: $J = \sqrt{[(y_{actual} - y_{predicted})^2]}$



Stochastic gradient descent

- Initial weights are randomly selected
- **Learning rate** determines size of "steps"



Cross-validation

- Put aside portion of training set that is not used to train the model
- After training, apply model to this **test set** data and calculate cost function to evaluate model's performance
- In practice, more complicated methods of splitting up training and test data

Neural network

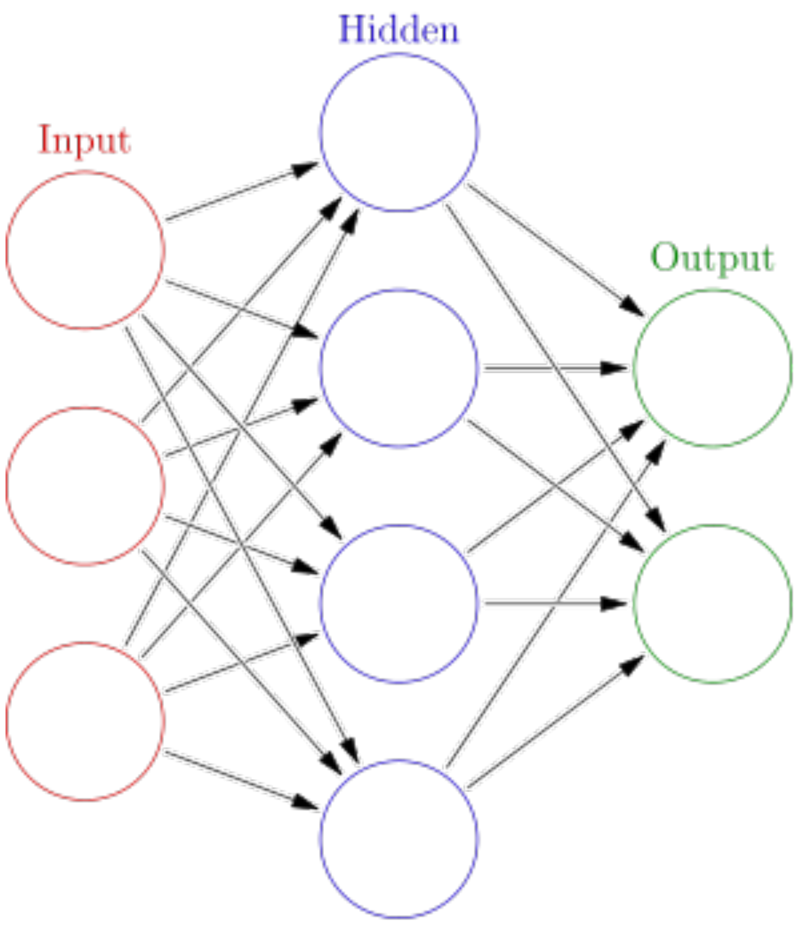
- Each hidden and output **neuron** is calculated through a function:

$$y = \begin{cases} 0 & \text{if } a_0 + a_1x_1 + \ldots + a_nx_n \leq 0 \\ 1 & \text{if } a_0 + a_1x_1 + \ldots + a_nx_n > 0 \end{cases}$$

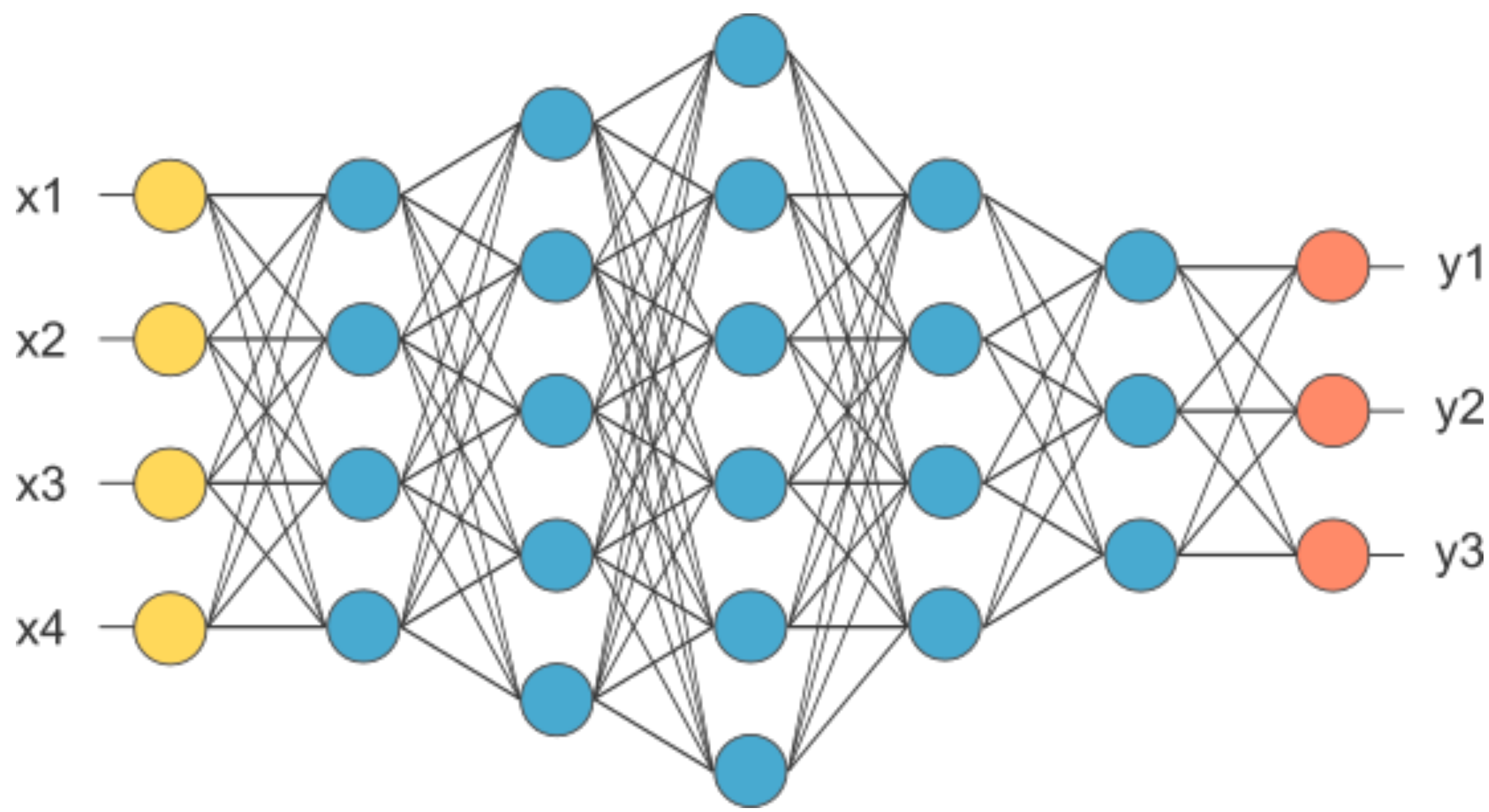
- Need to fit all a_i

$$(3 + 1) \cdot (4 + 1) \cdot 2 = 40$$

- In practice, the above can be more complicated than just linear



Deep learning



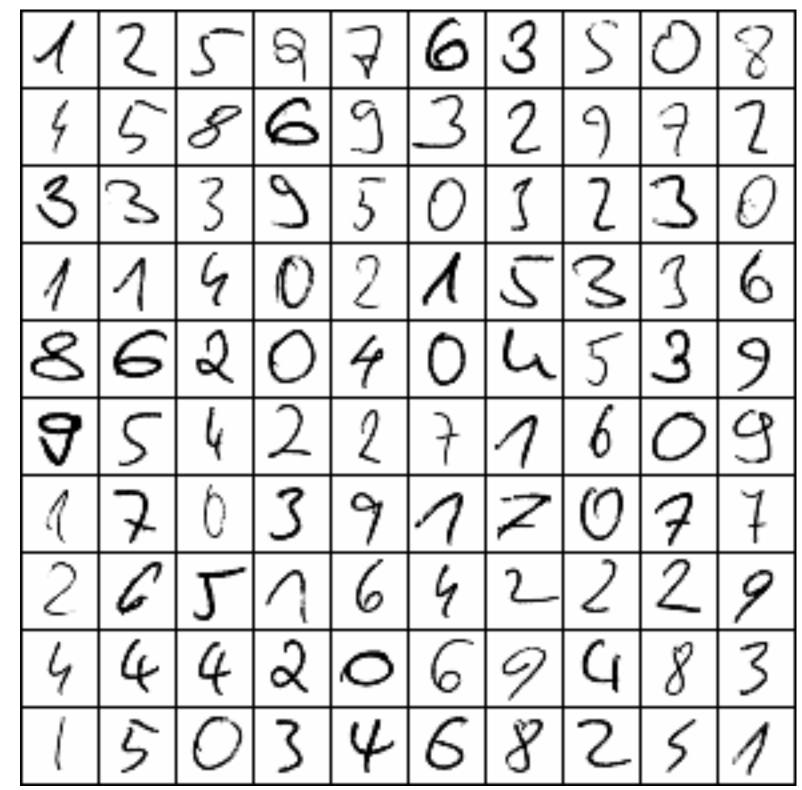
Why neural networks?

- Only recently have computers become powerful enough to fit complex neural networks
- Optimization often requires parallel computations, hence GPUs
- Proven particularly powerful for unstructured data (images, audio, video) and nonlinear behavior

Let’s build a neural network!

The data

- MNIST database of handwritten digits
- Each image is grayscale scan with 28x28 pixels
- Each row has 784 inputs representing value at each pixel
- Classic data set from machine learning research



The model

- Using scikit-learn's MLPClassifier
- Default settings: 1 hidden layer of 100 neurons
- In practice, MLPClassifier can't handle large-scale data, but will work for a relatively small data set

Resources for learning more

- Andrew Ng’s Coursera courses
    - Machine Learning: https://www.coursera.org/learn/machine-learning
    - Deep Learning: https://www.coursera.org/specializations/deep-learning
- *Neural Networks: A Systematic Introduction*, Raul Rojas: https://page.mi.fu-berlin.de/rojas/neural/
- TensorFlow: https://www.tensorflow.org/
- Keras: https://keras.io/
- Kaggle: https://www.kaggle.com/