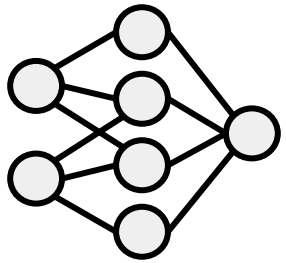
Introduction to machine learning



Look familiar?

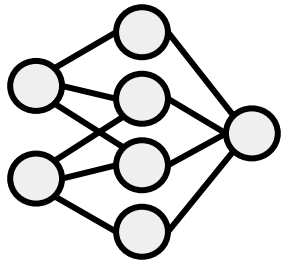
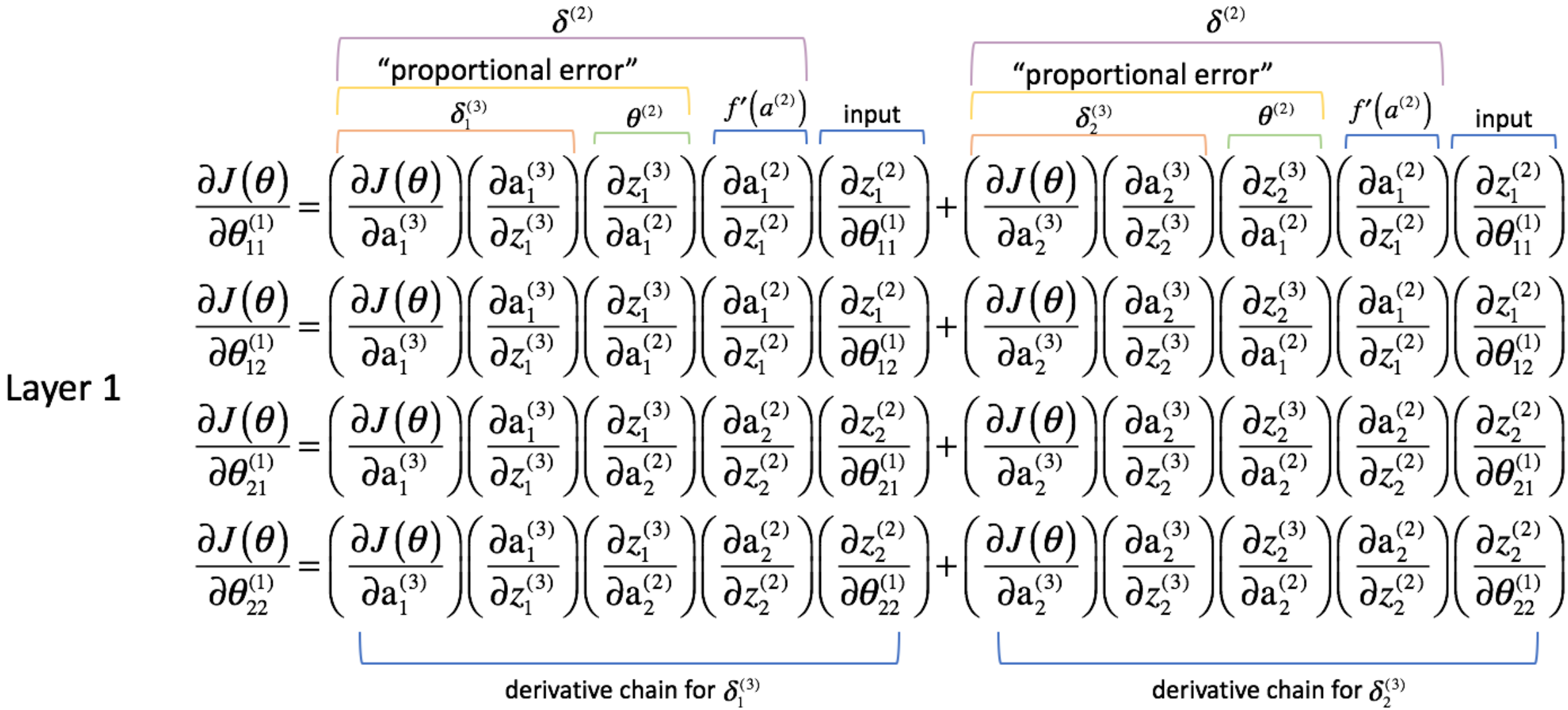


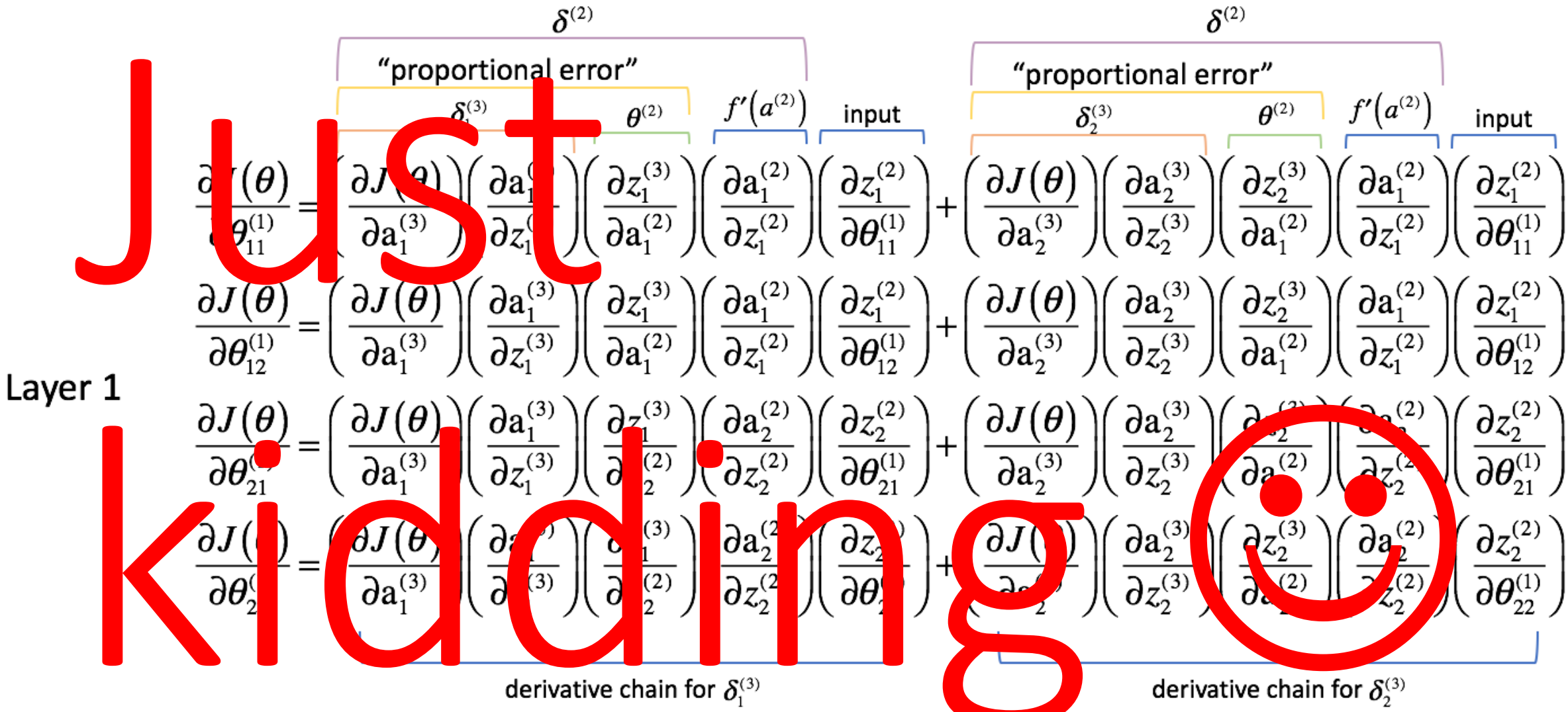
Figure from Sergey Ovchinnikov,
Boston Protein Design and Modeling Club

Let’s start by deciphering this…



https://www.jeremyjordan.me/neural-networks-training/

Let's start by deciphering this…



Just kidding ☺

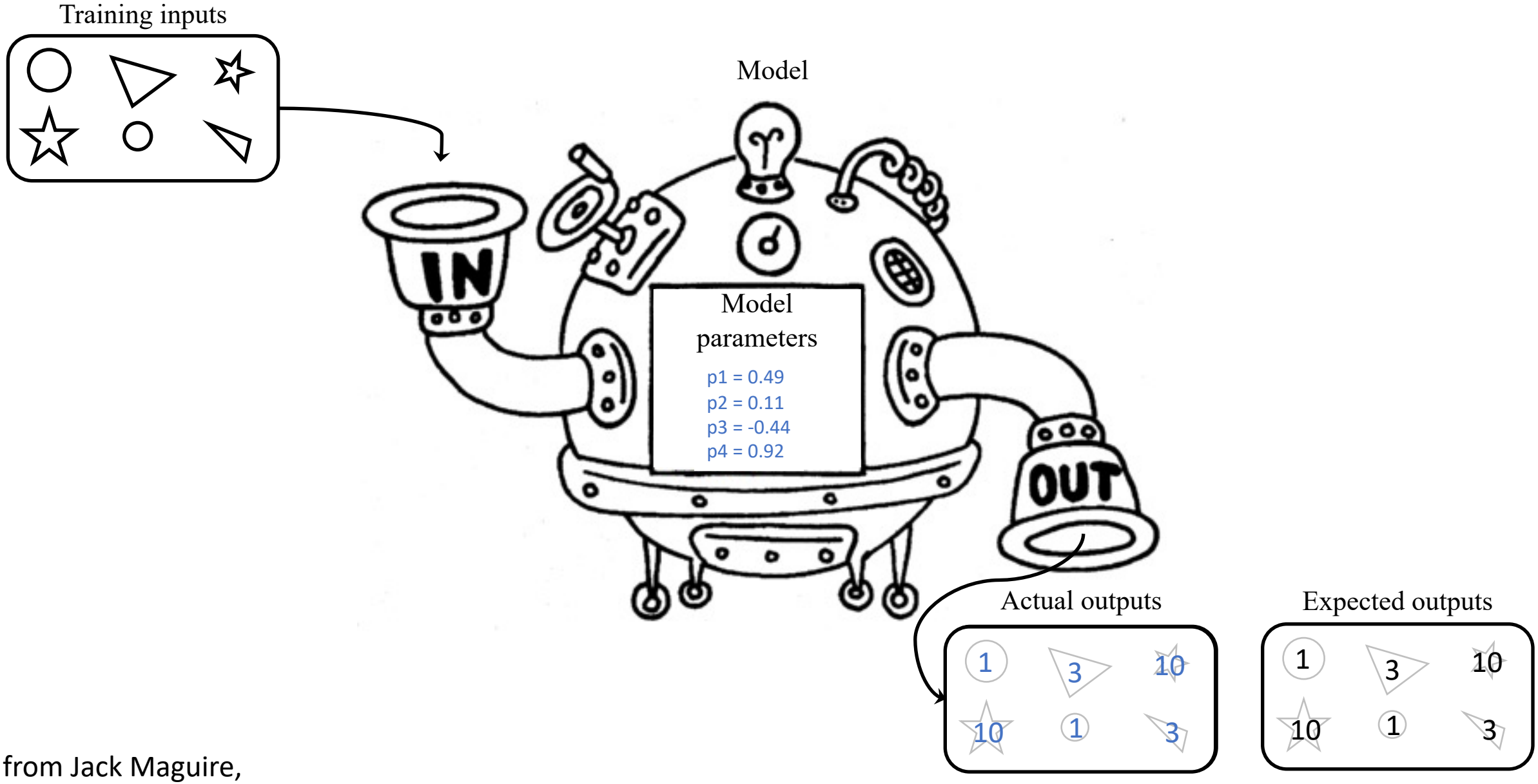
https://www.jeremyjordan.me/neural-networks-training/

First, we will learn machine learning basics

Main concepts that we will cover before structure prediction via ML

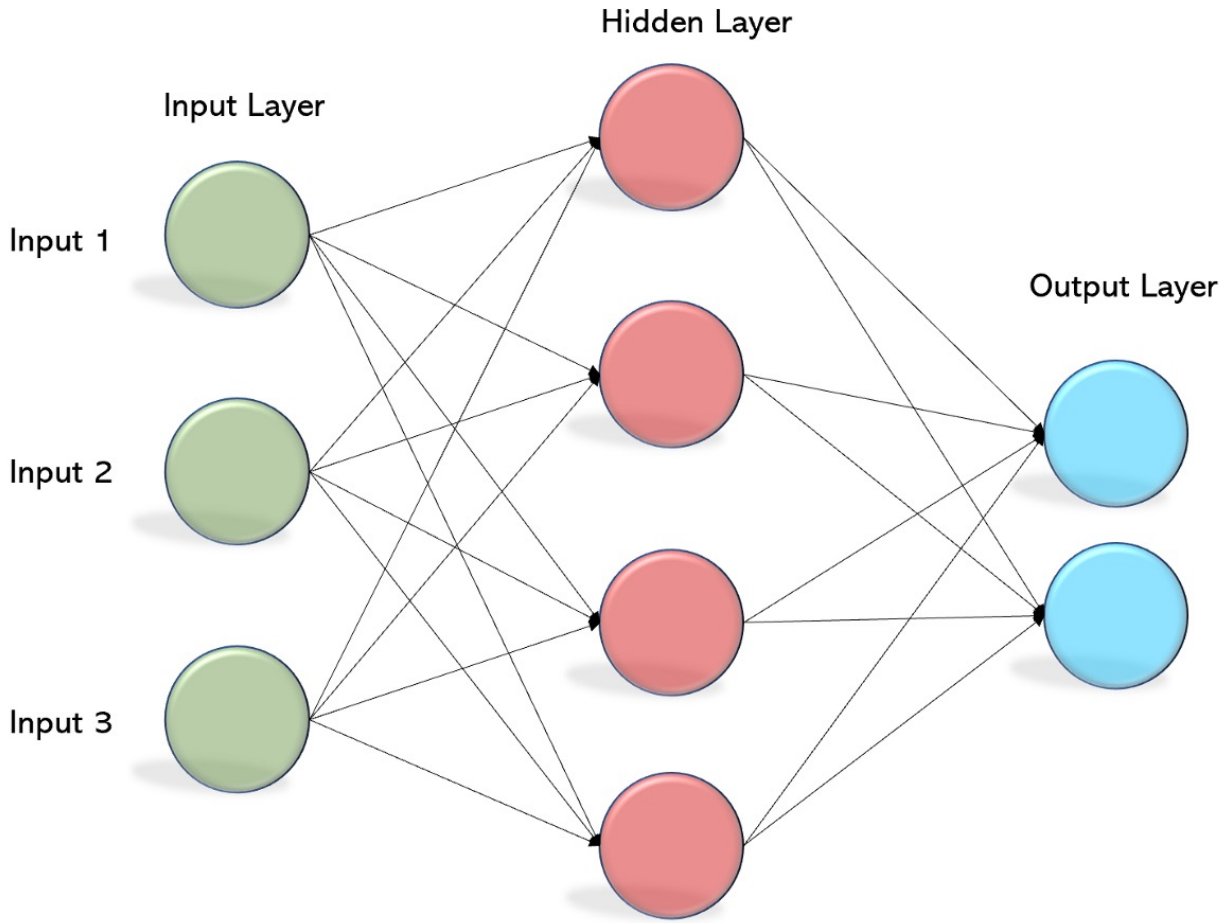
- Loss function (Mean Squared Error, Cross entropy)
- Optimization techniques (Gradient descent)
- Activation function (ReLU)
- Normalization
- Learning rate
- Training
- Backpropagation

What is supervised machine learning?



Slide from Jack Maguire, VLMMLR Spring 2021

Basic Deep Neural Networks (DNNs) a.k.a **Multilayer Perceptrons** (MLPs)



https://becominghuman.ai/

Loss functions

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2$$

$$CE = -\sum_{i=1}^{n} Y_i \cdot \log \widehat{Y}_i$$

Loss functions

$$MSE = \frac{1}{n}\sum_{n=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2$$

$$CE = -\sum_{n=1}^{n} p(x)\log q(x)$$

Optimization via Gradient Descent

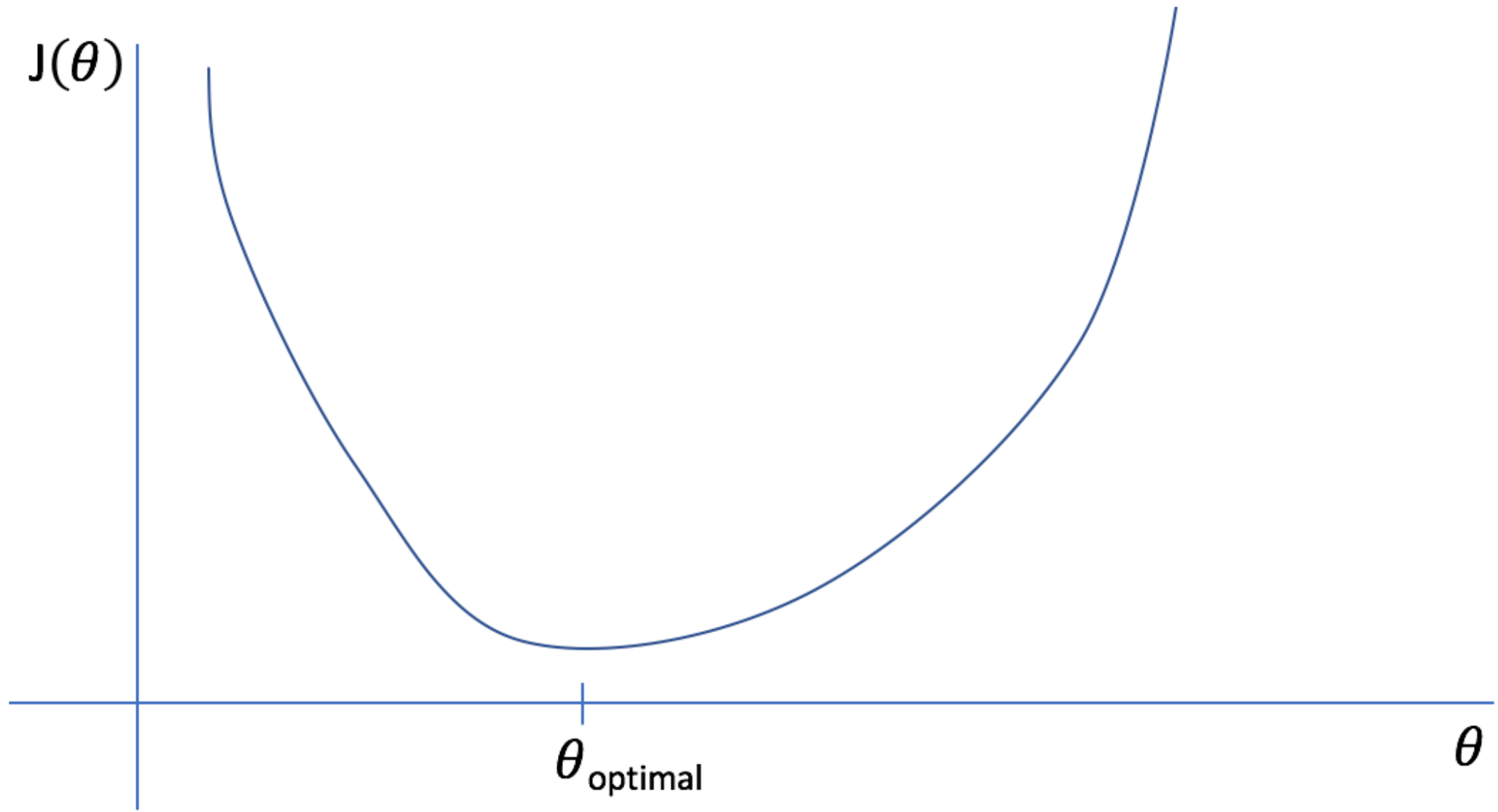
$$\theta_i := \theta_i + \Delta\theta_i$$

where

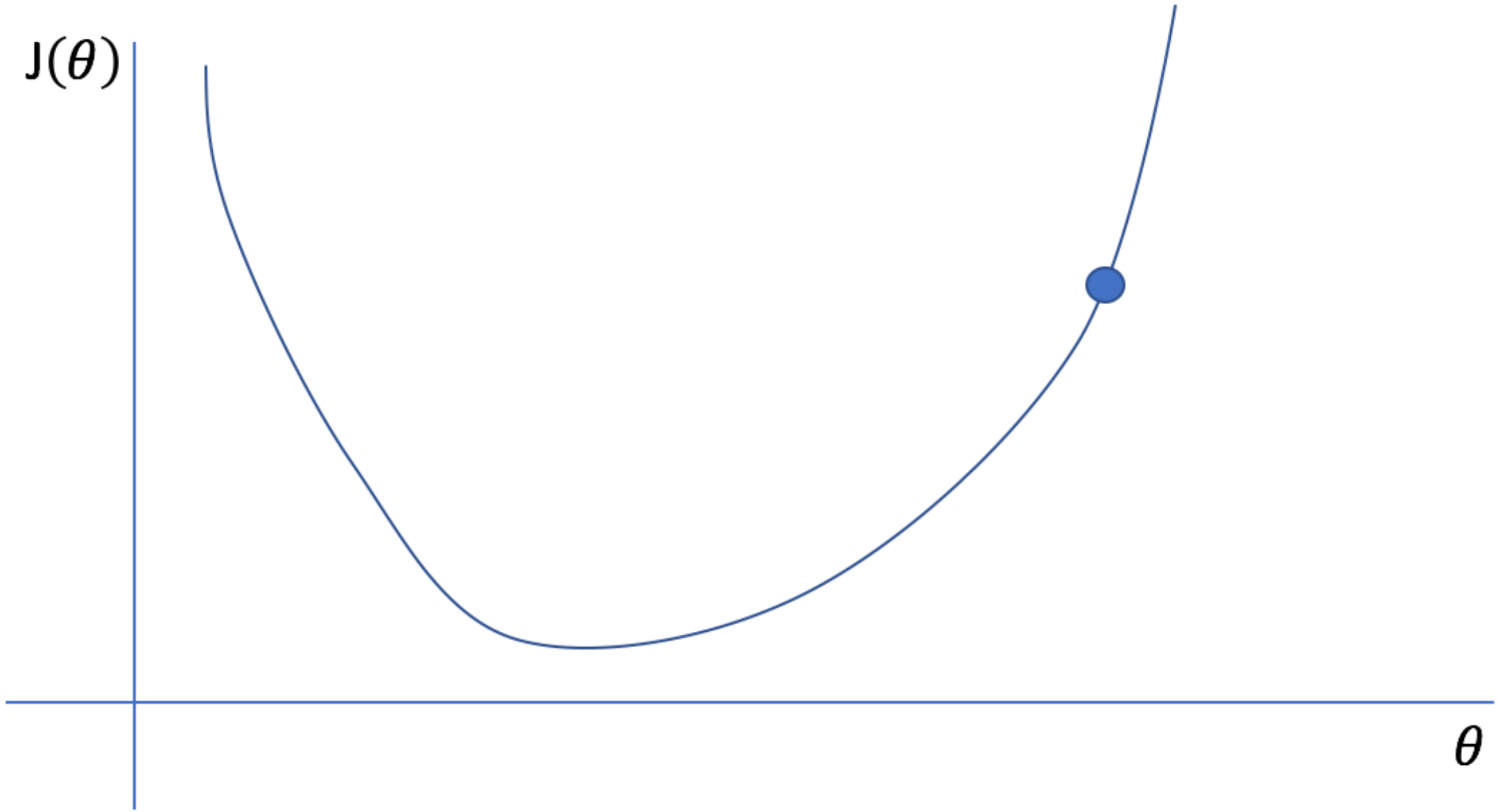
$\theta \rightarrow$ parameter

$\alpha \rightarrow$ learning rate

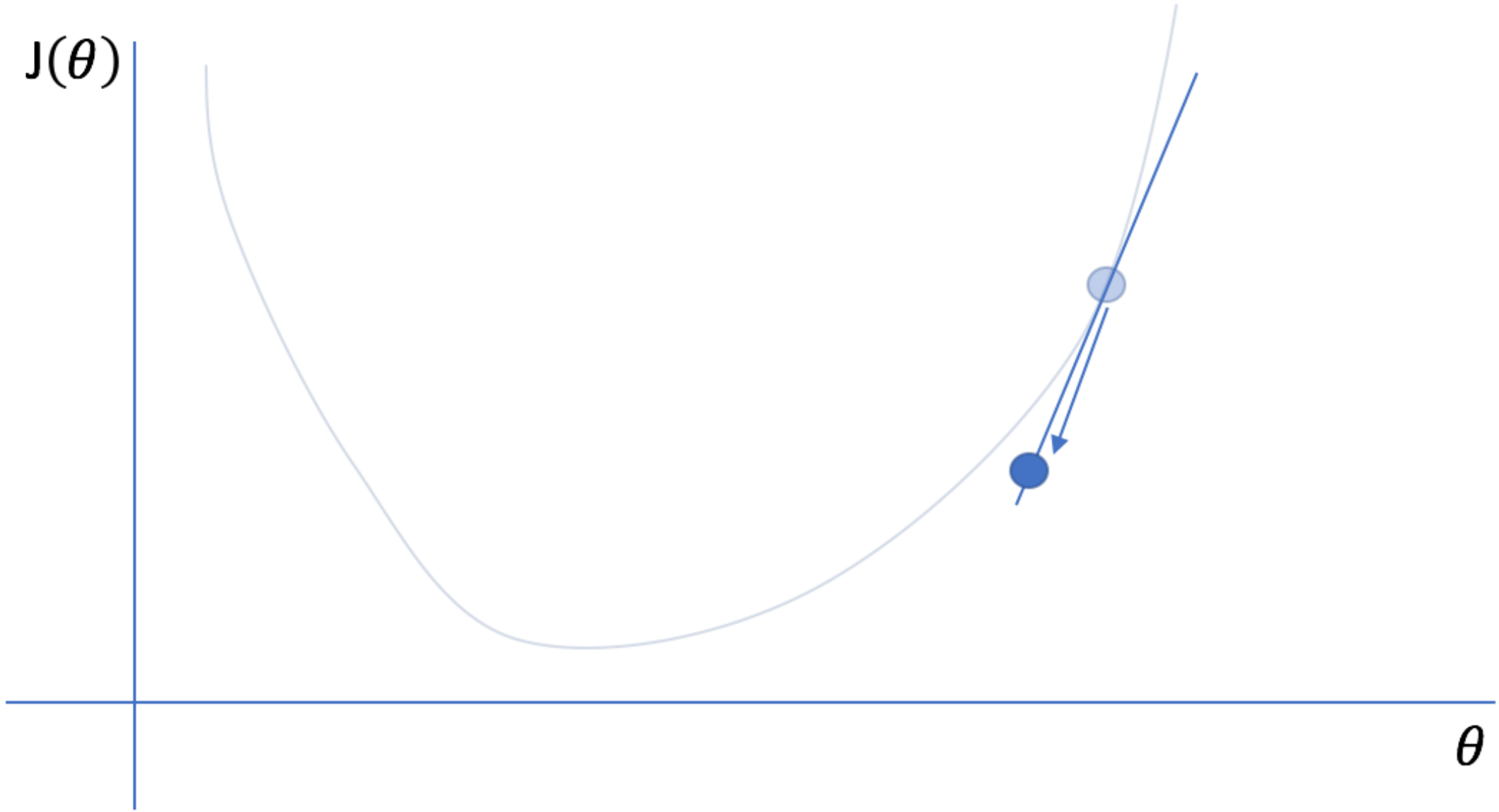
$$\Delta\theta_i = -\alpha \frac{\partial J(\theta)}{\partial \theta_i}$$



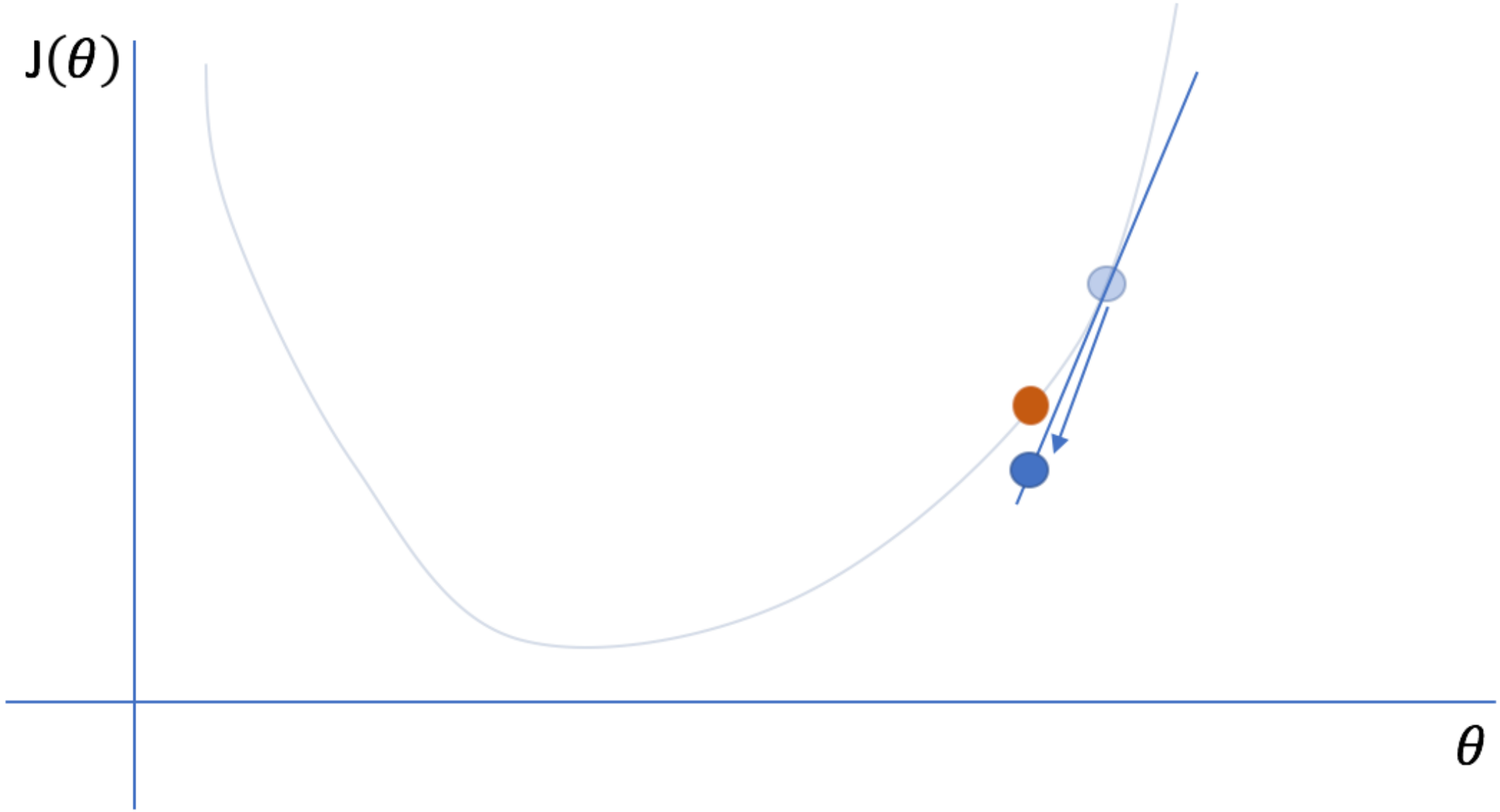
https://www.jeremyjordan.me



https://www.jeremyjordan.me



https://www.jeremyjordan.me

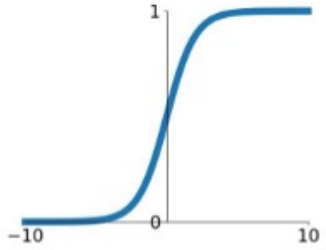


https://www.jeremyjordan.me

Activation functions

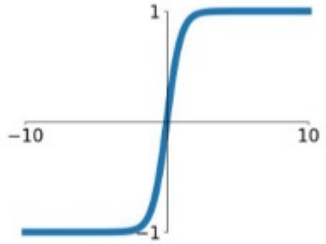
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



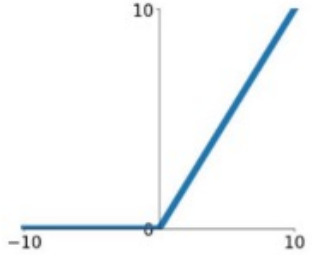
tanh

$$\tanh(x)$$



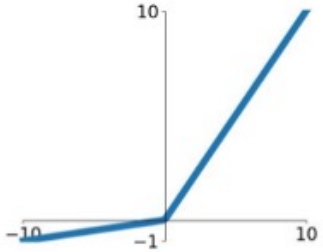
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

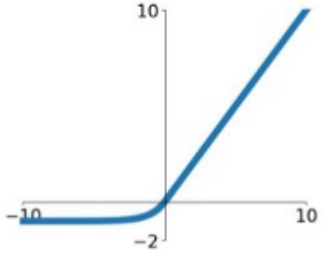


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

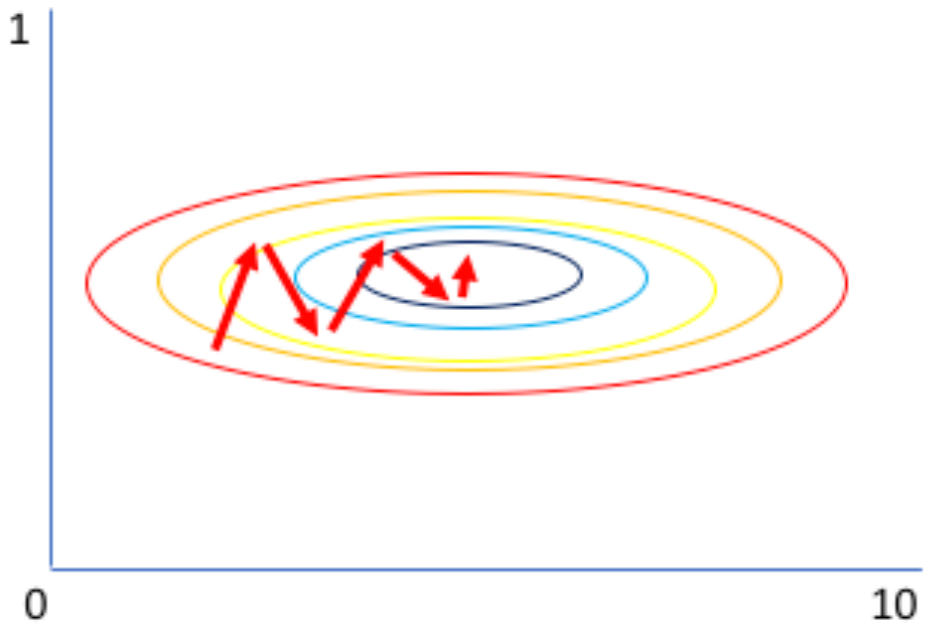
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

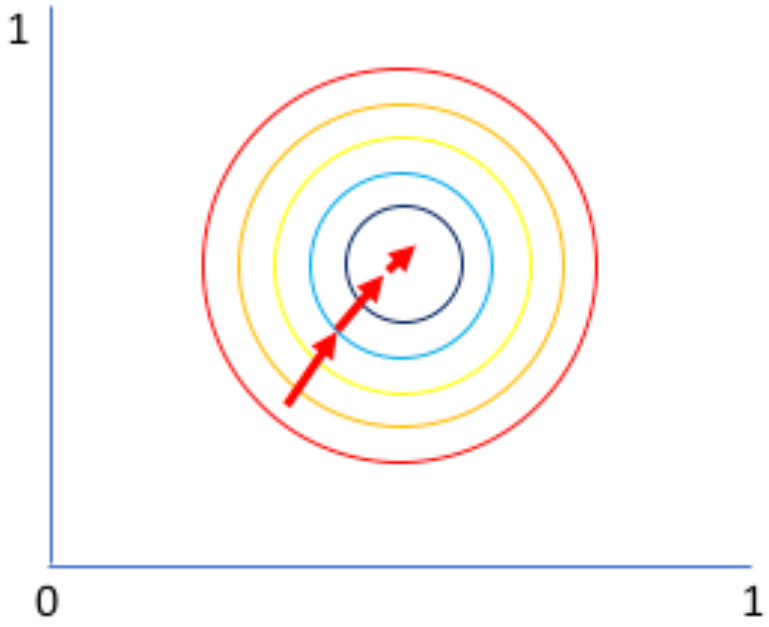


From Jadon S. (2018) Introduction to Different Activation Functions for Deep Learning. https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092.

Normalization



Gradient of larger parameter dominates the update

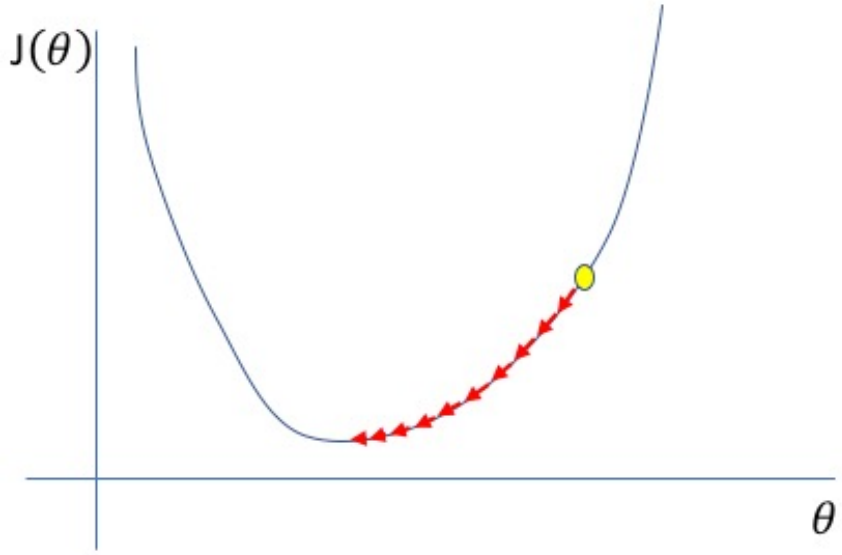


Both parameters can be updated in equal proportions

https://www.jeremyjordan.me

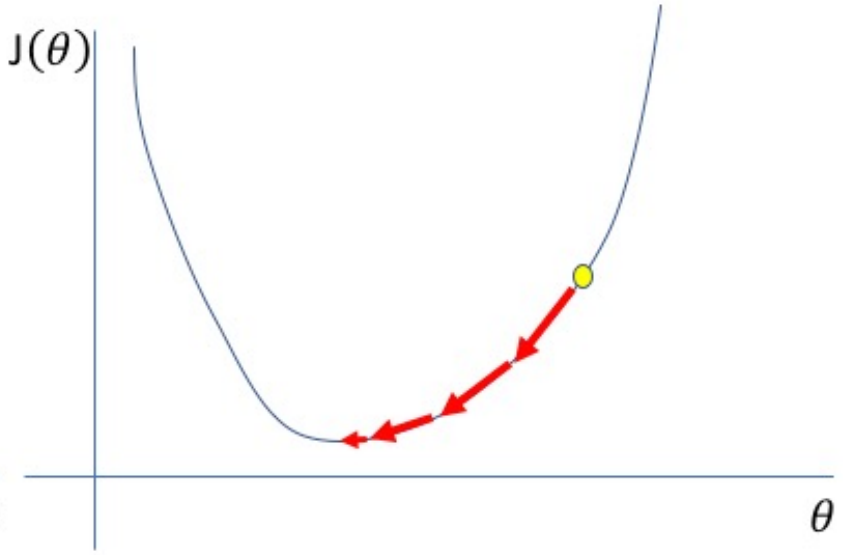
Learning rate

Too low



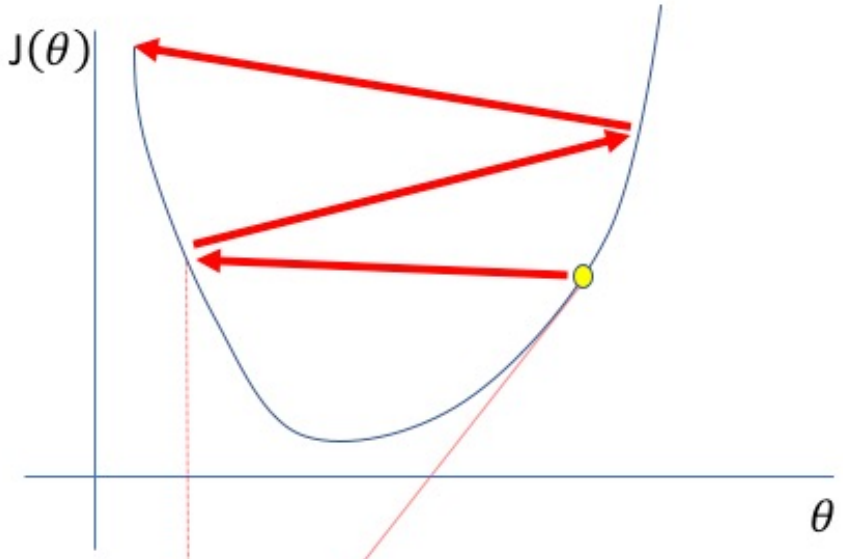
A small learning rate requires many updates before reaching the minimum point

Just right



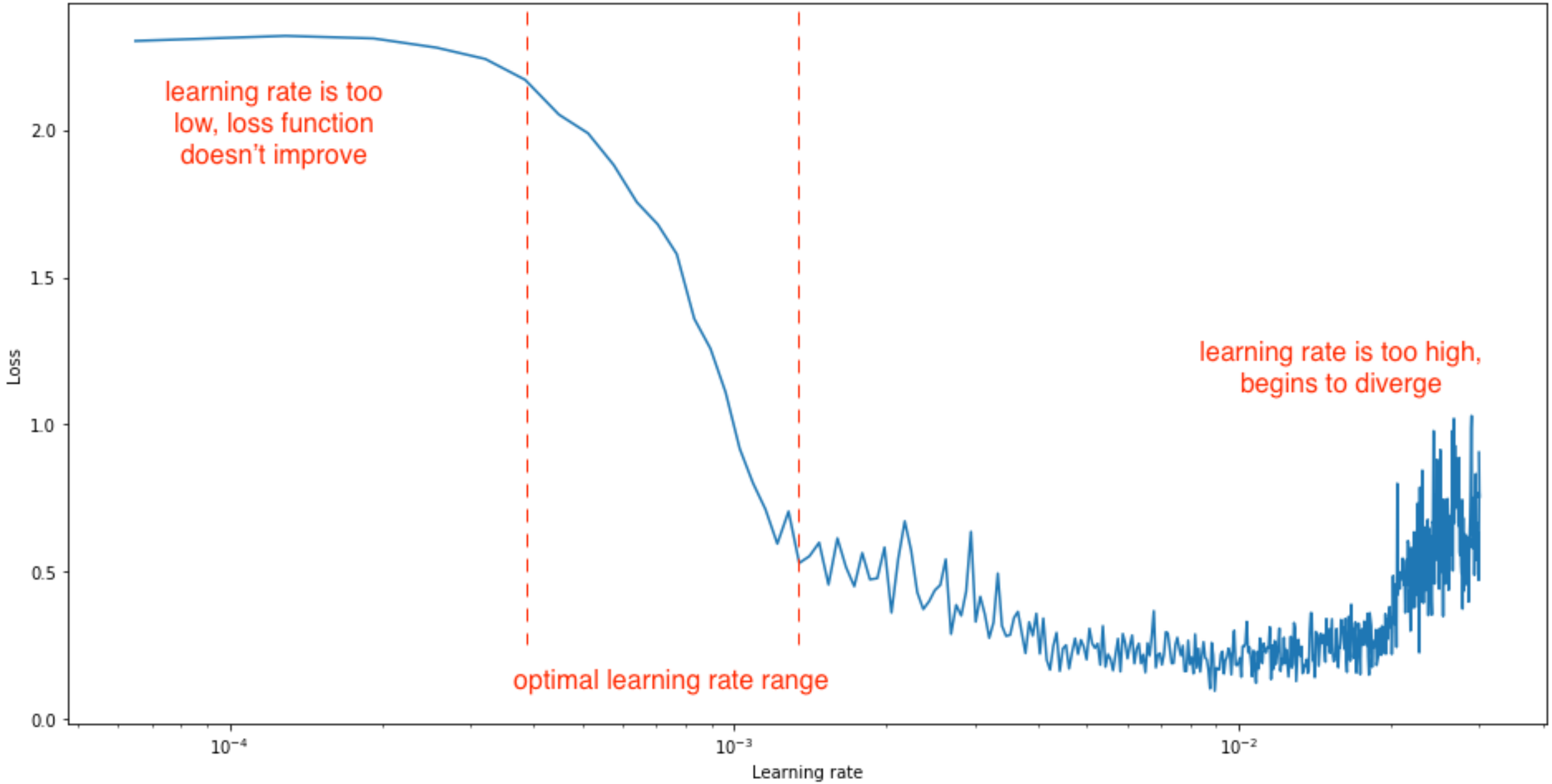
The optimal learning rate swiftly reaches the minimum point

Too high



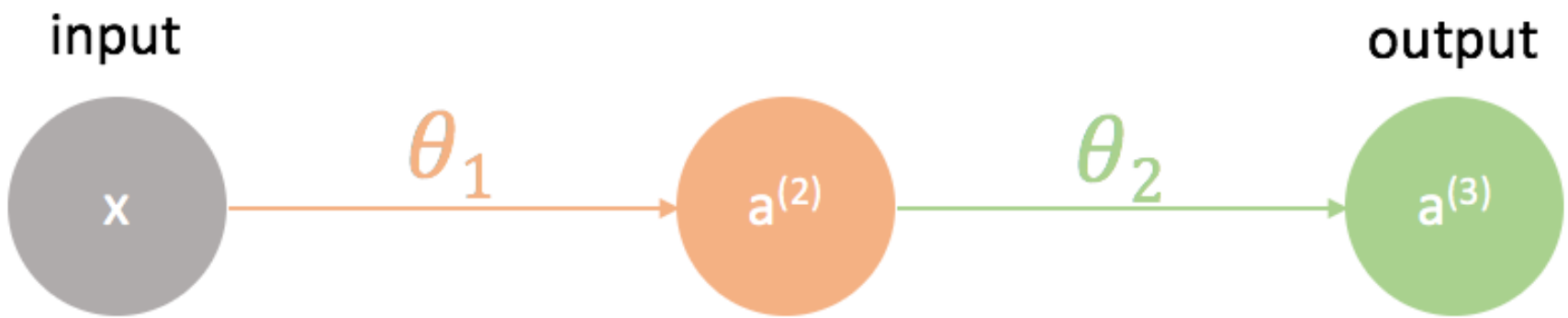
Too large of a learning rate causes drastic updates which lead to divergent behaviors

https://www.jeremyjordan.me



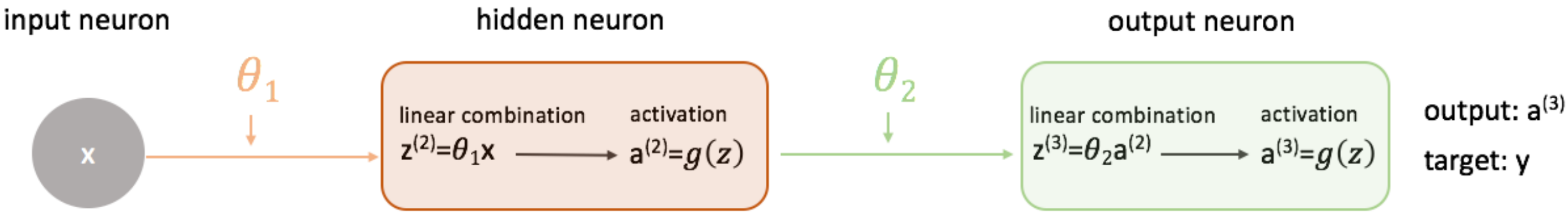
https://www.jeremyjordan.me

Training with backpropagation

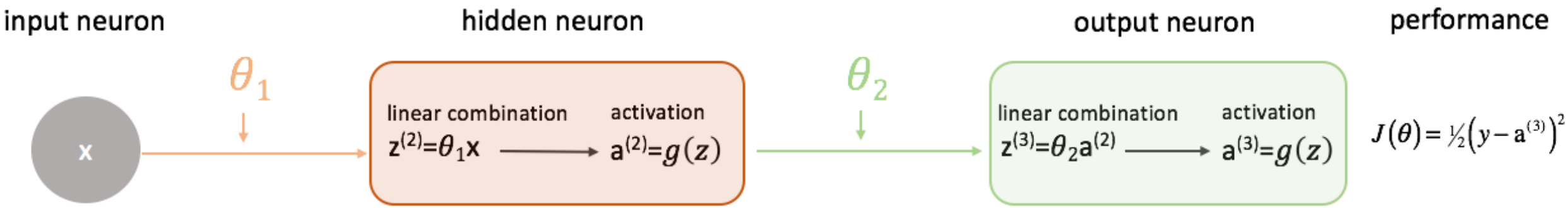


$$a^{(2)} = g(\theta_1 x)$$

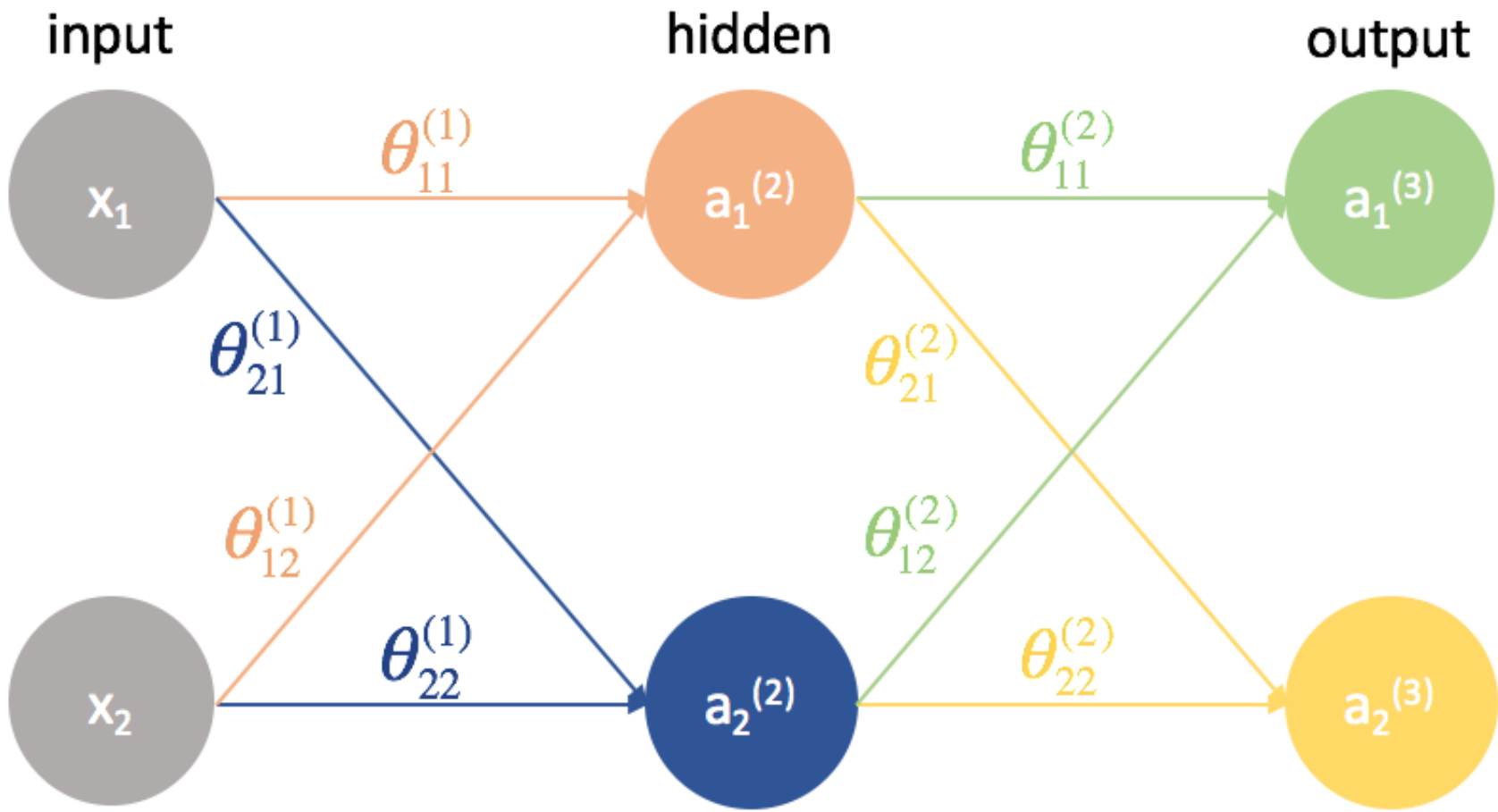
$$a^{(3)} = g(\theta_2 a^{(2)})$$



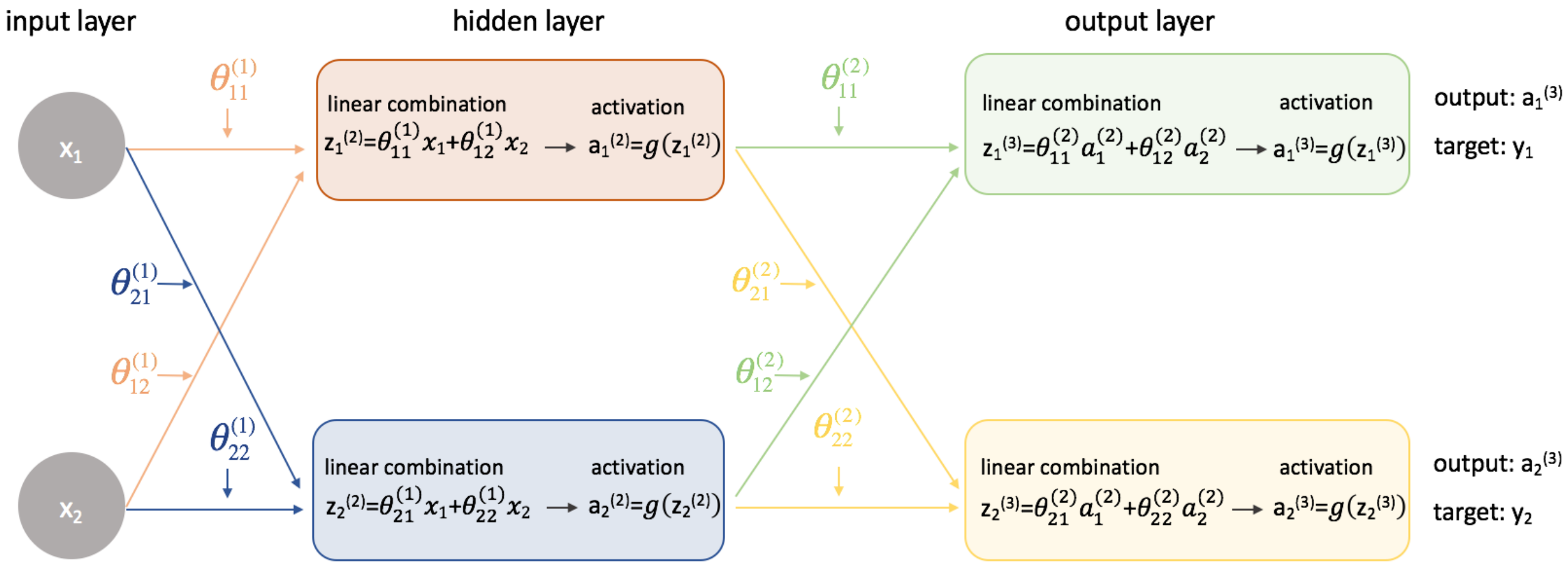
https://www.jeremyjordan.me



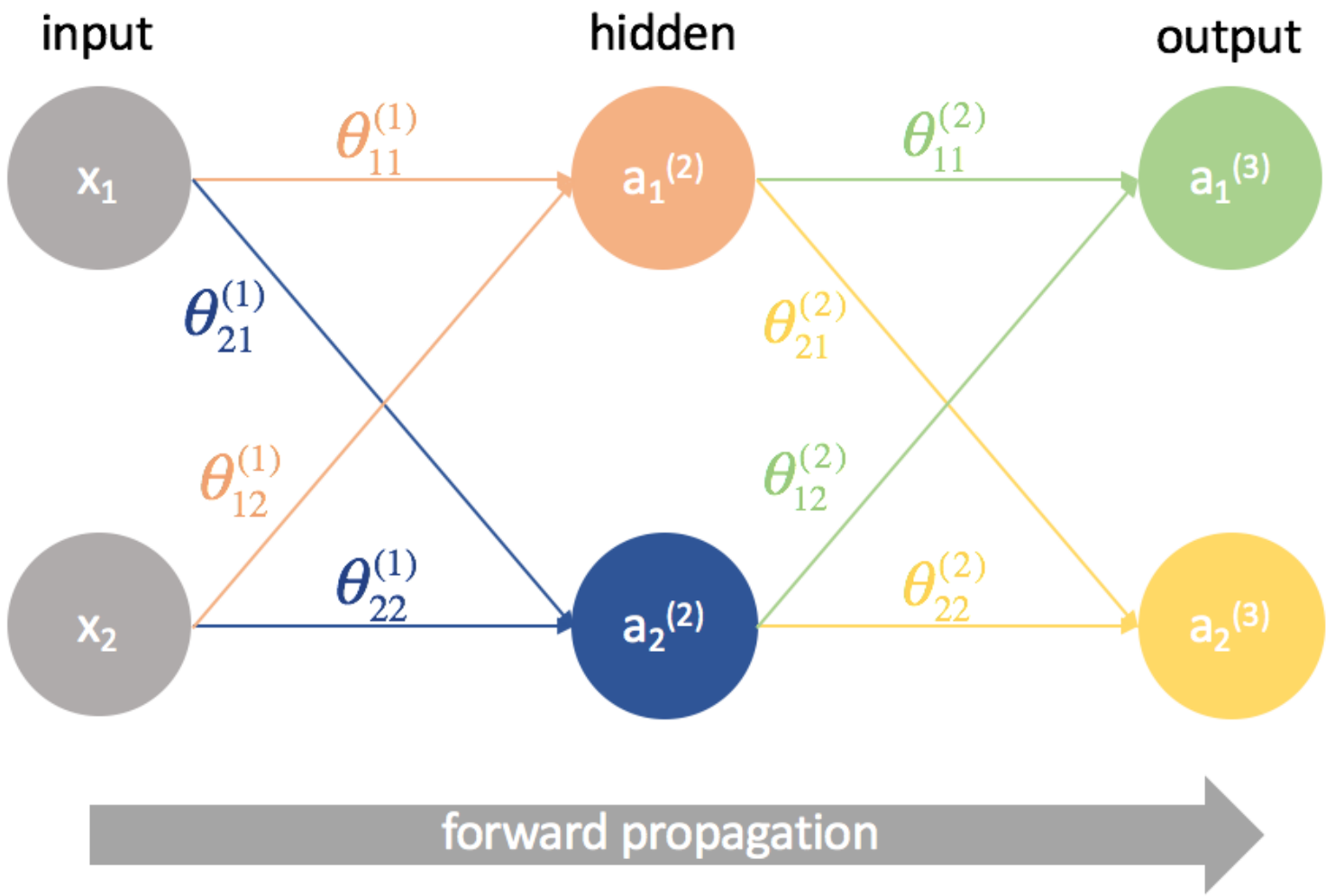
https://www.jeremyjordan.me



https://www.jeremyjordan.me

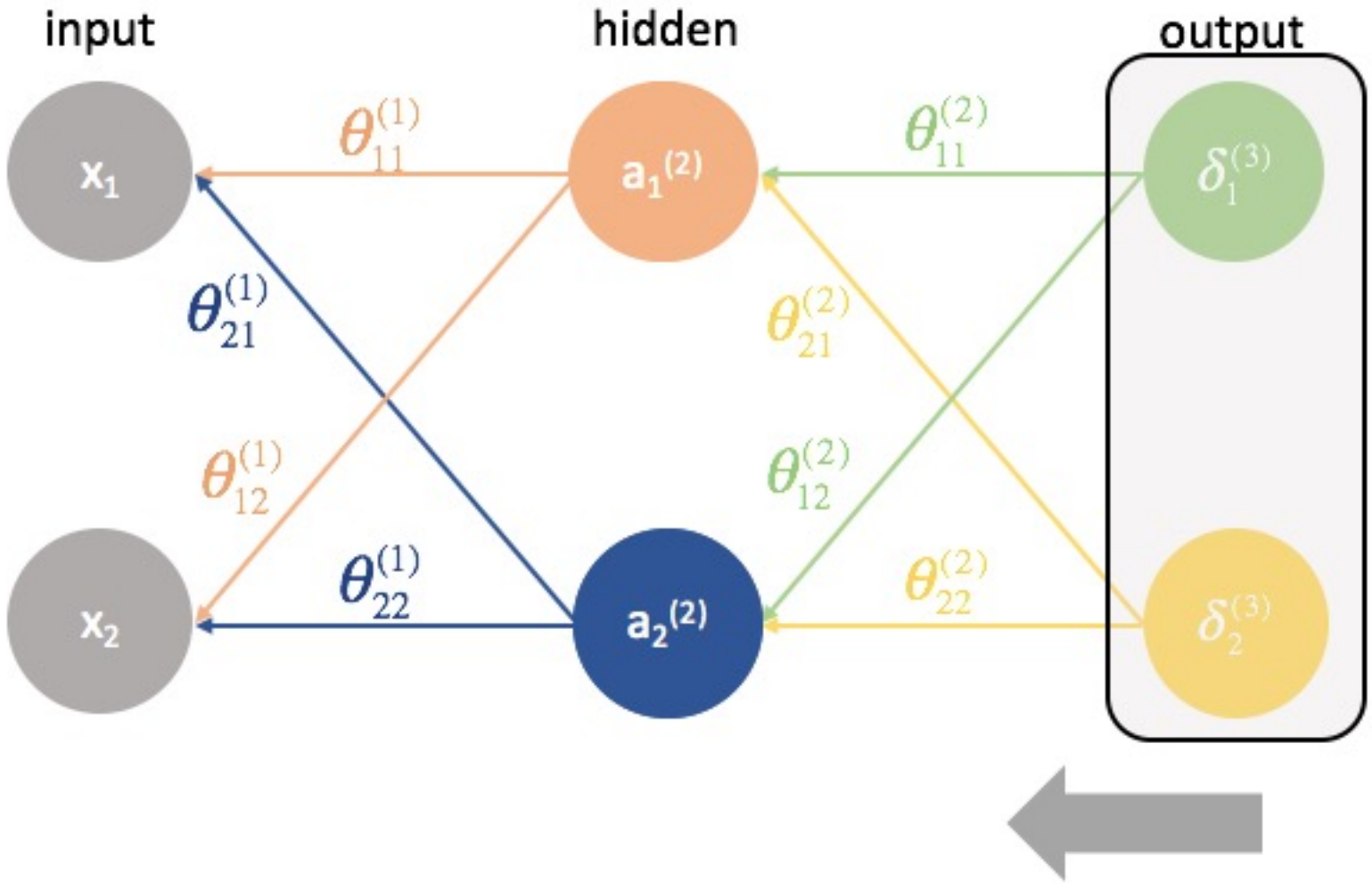


https://www.jeremyjordan.me

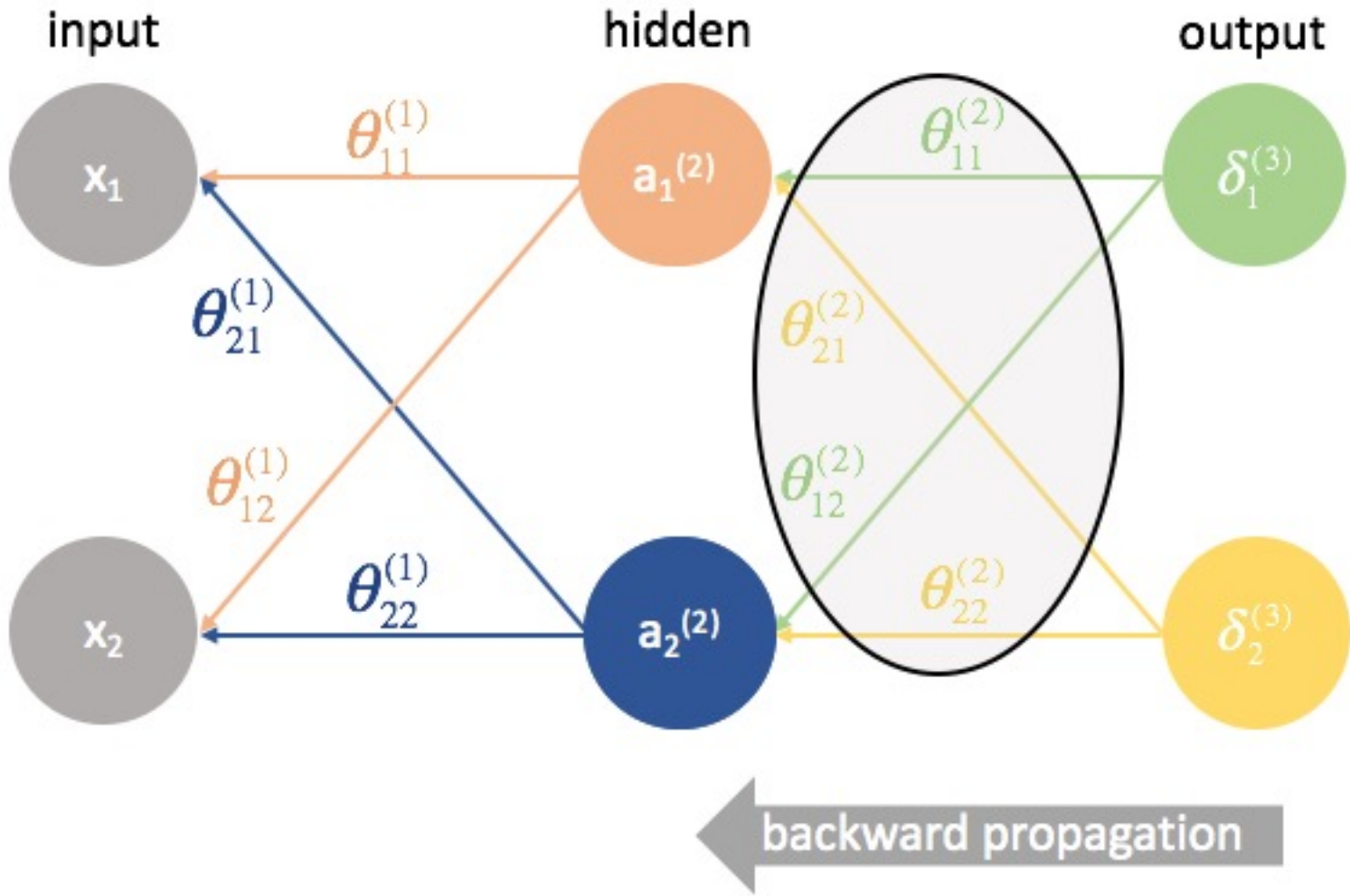


https://www.jeremyjordan.me

$$\delta^{(3)} = \frac{1}{m}\left(y - a^{(3)}\right) f'\left(a^{(3)}\right)$$

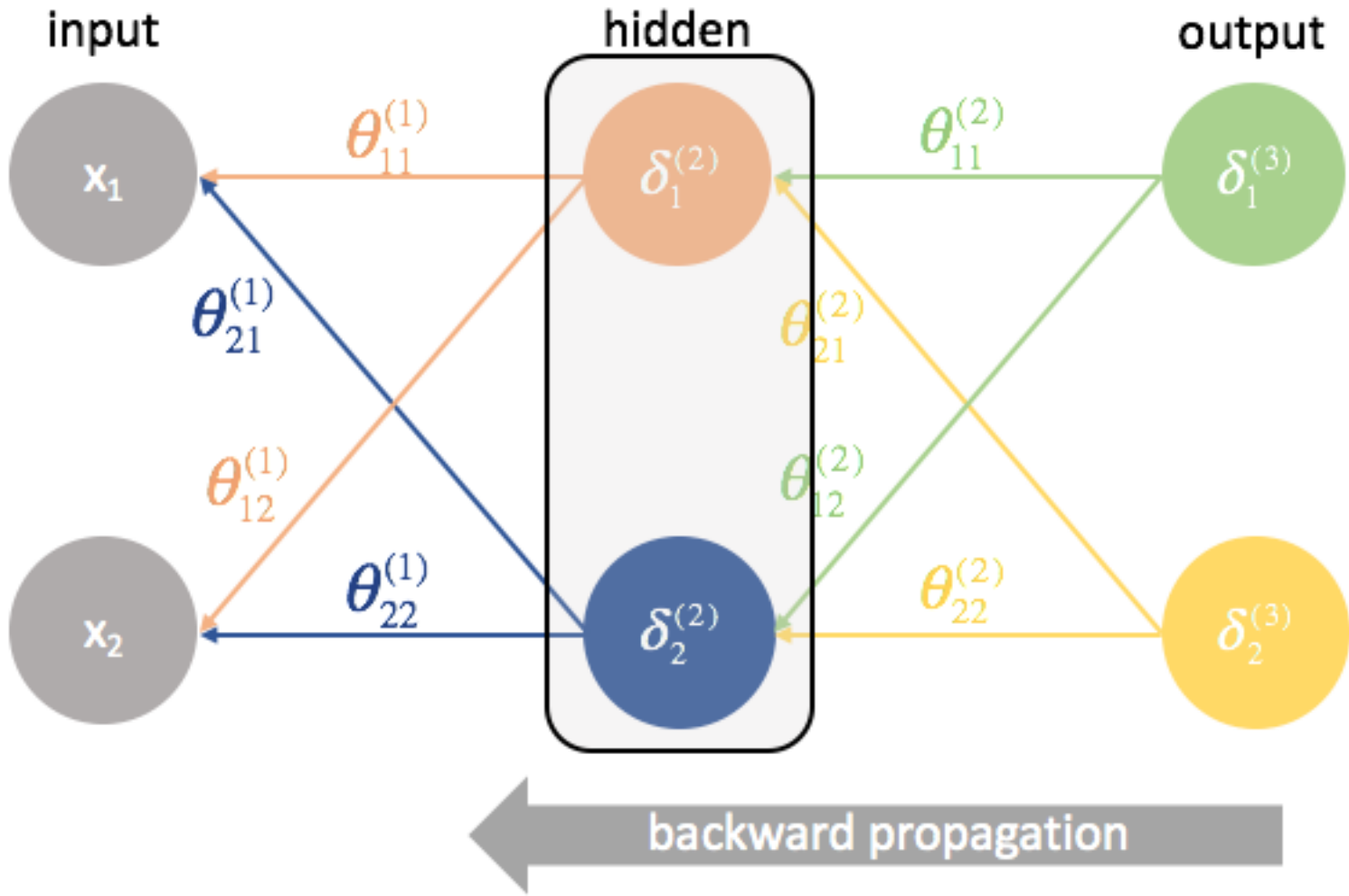


$$\frac{\partial J(\theta)}{\partial \theta_{ij}^{(2)}} = \left(\delta^{(3)}\right)^T a^{(2)}$$



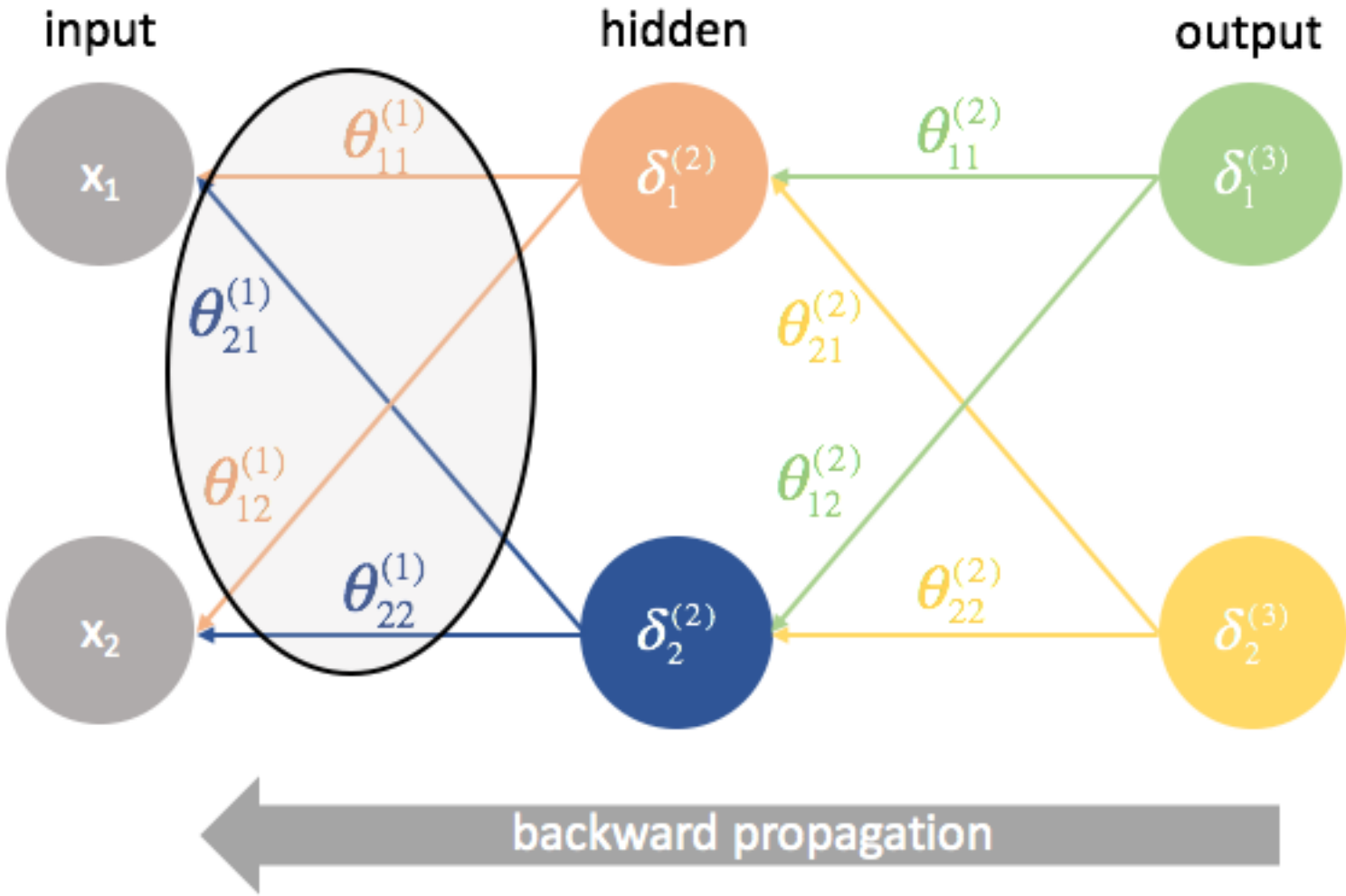
https://www.jeremyjordan.me

$$\delta^{(2)} = \delta^{(3)}\Theta^{(2)} f'\left(a^{(2)}\right)$$



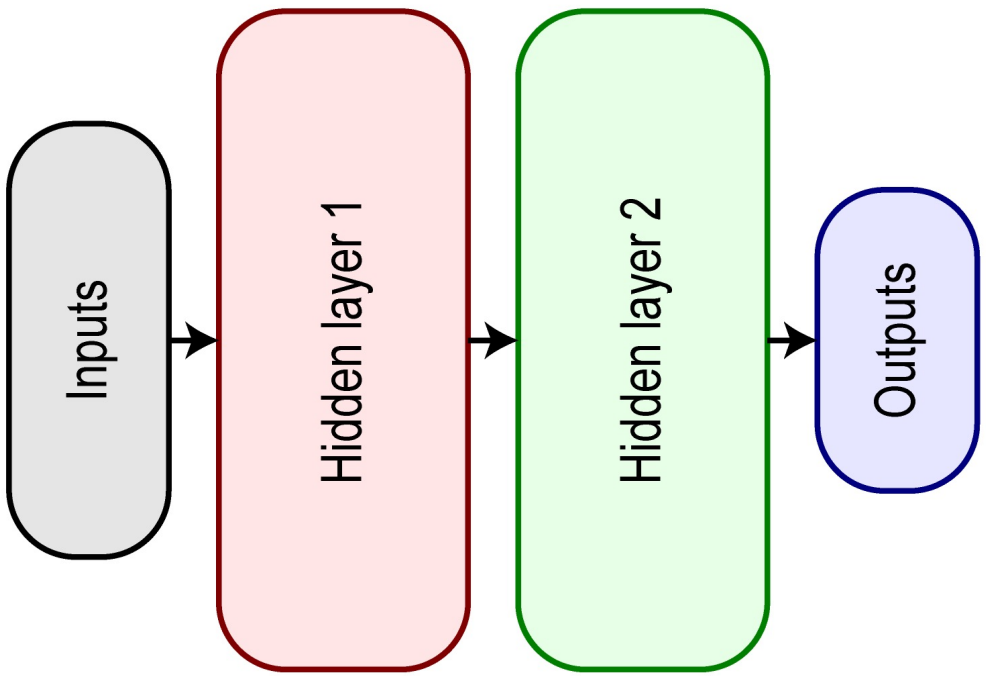
https://www.jeremyjordan.me

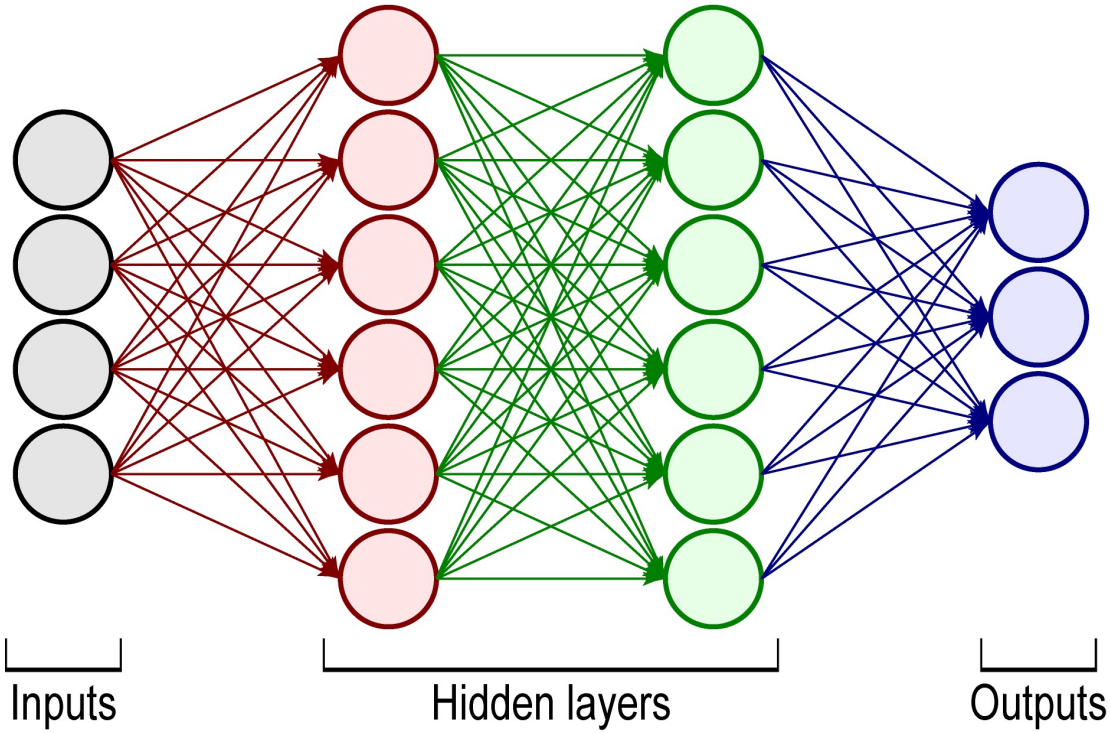
$$\frac{\partial J(\theta)}{\partial \theta_{ij}^{(1)}} = \left(\delta^{(2)}\right)^T x$$



https://www.jeremyjordan.me

Basic deep neural networks (a.k.a. multilayer perceptrons/MLPs)

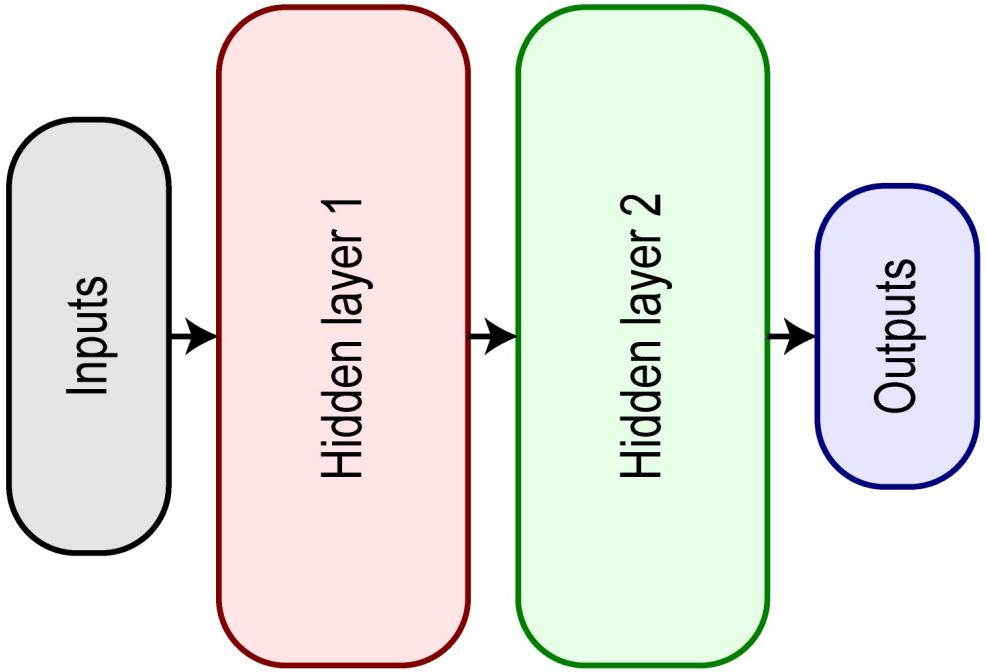


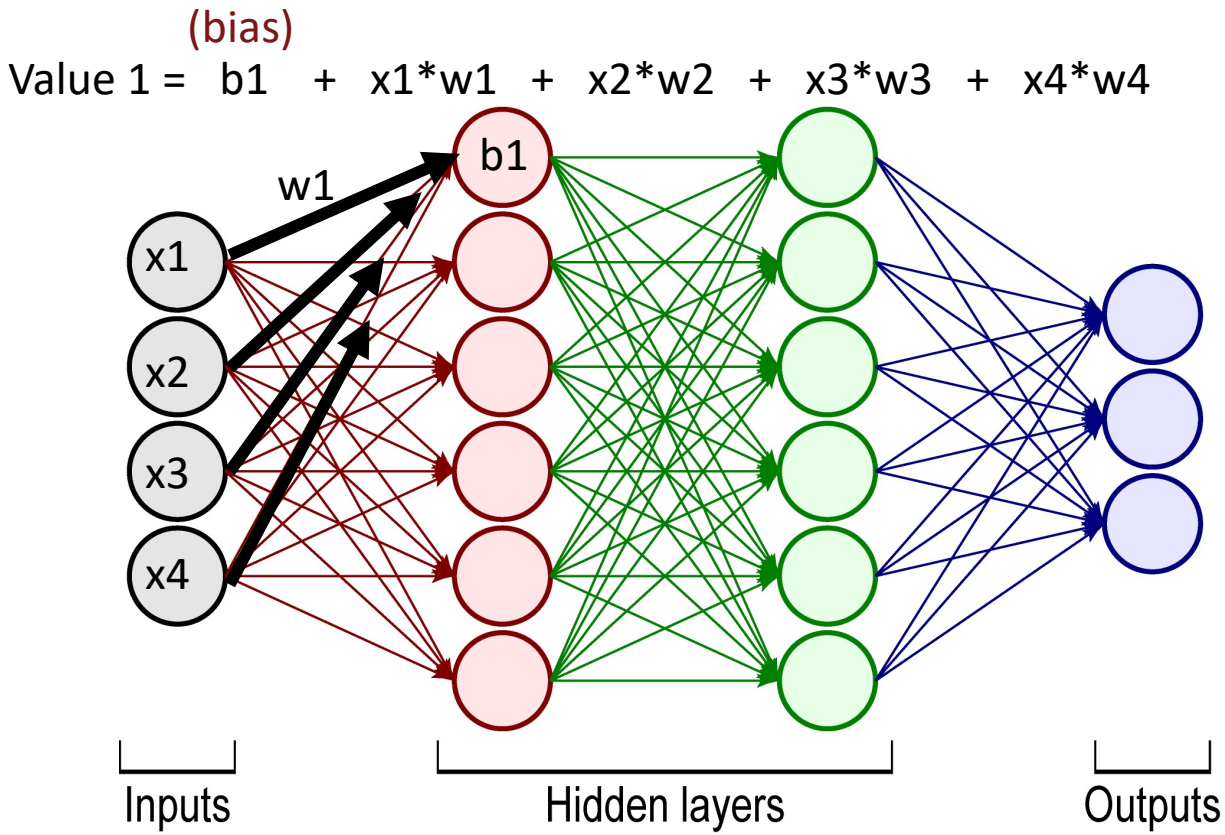


$$f_o\left(f_b\left(f_a\left([x_1 \cdots x_4]\begin{bmatrix} w_{a,11} & \cdots & w_{a,16} \\ \vdots & \ddots & \vdots \\ w_{a,41} & \cdots & w_{a,46} \end{bmatrix} + [c_{a,1} \cdots c_{a,6}]\right)\begin{bmatrix} w_{b,11} & \cdots & w_{b,16} \\ \vdots & \ddots & \vdots \\ w_{b,61} & \cdots & w_{b,66} \end{bmatrix} + [c_{b,1} \cdots c_{b,6}]\right)\begin{bmatrix} w_{o,11} & \cdots & w_{o,13} \\ \vdots & \ddots & \vdots \\ w_{o,61} & \cdots & w_{o,63} \end{bmatrix} + [c_{o,1} \quad c_{o,2} \quad c_{o,3}]\right) = [y_1 \quad y_2 \quad y_3]$$

Slide from Jack Maguire, VLMMLR Spring 2021

Basic deep neural networks (a.k.a. multilayer perceptrons/MLPs)





$$f_o\left(f_b\left(f_a\left([x_1 \cdots x_4]\begin{bmatrix} w_{a,11} & \cdots & w_{a,16} \\ \vdots & \ddots & \vdots \\ w_{a,41} & \cdots & w_{a,46} \end{bmatrix} + [c_{a,1} \cdots c_{a,6}]\right)\begin{bmatrix} w_{b,11} & \cdots & w_{b,16} \\ \vdots & \ddots & \vdots \\ w_{b,61} & \cdots & w_{b,66} \end{bmatrix} + [c_{b,1} \cdots c_{b,6}]\right)\begin{bmatrix} w_{o,11} & \cdots & w_{o,13} \\ \vdots & \ddots & \vdots \\ w_{o,61} & \cdots & w_{o,63} \end{bmatrix} + [c_{o,1} \quad c_{o,2} \quad c_{o,3}]\right) = [y_1 \quad y_2 \quad y_3]$$

Slide from Jack Maguire, VLMMLR Spring 2021

(adapted from original slide by Vikram Mulligan)

How do we figure out these weights?

- Two families:
  - Gradient-based minimization
    - This will work for us 99% of the time
    - Very similar to Rosetta's MinMover protocol
  - Derivative free optimization
    - "reinforcement learning"
    - AlphaGo, DeepStack, etc.
    - Learn to play video games, where "loss" is harder to calculate
    - Very similar to our docking, folding methods
      - Monte Carlo & genetic algorithms

Slide from Jack Maguire, VLMMLR Spring 2021

Summary

- Neural networks are chains of parameters
- If our system is differentiable, we can use gradient-based minimization to optimize these parameters
  - 99% use case
  - Can be tricky to ensure a healthy gradient
    - Normalize all input data
    - Use batch normalization for large models
  - Be aware of your learning rate!
- Otherwise, we must use derivative-free optimization
  - Less common
  - “reinforcement learning”
  - *Much* slower

Slide from Jack Maguire, VLMMLR Spring 2021

Resources

Math Overview: https://www.jeremyjordan.me/neural-networks-training/

Normalization: https://www.jeremyjordan.me/batch-normalization/

Learning Rate: https://www.jeremyjordan.me/nn-learning-rate/

Interactive Demos

- http://neuralnetworksanddeeplearning.com/chap4.html
- https://playground.tensorflow.org/

Slide from Jack Maguire, VLMMLR Spring 2021

It’s time for a quick tensorflow playground NN demo

https://playground.tensorflow.org/

Data tools and techniques

- Basic Data Manipulation and Analysis
  Performing well-defined computations or asking well-defined questions (“queries”)
- Data Mining
  Looking for patterns in data
- Machine Learning
  Using data to build models and make predictions
- Data Visualization
  Graphical depiction of data
- Data Collection and Preparation

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

Regression

Using data to build models and make predictions

- Supervised
- Training data, each example:
  - Set of predictor values - “independent variables”
  - Numerical output value - “dependent variable”
- Model is function from predictors to output
  - Use model to predict output value for new predictor values
- Example
  - Predictors: mother height, father height, current age
  - Output: height

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

Classification

Using data to build models and make predictions

- Supervised
- Training data, each example:
  - Set of feature values – numeric or categorical
  - Categorical output value - "label"
- Model is method from feature values to label
  - Use model to predict label for new feature values
- Example
  - Feature values: age, gender, income, profession
  - Label: buyer, non-buyer

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

Other examples

Medical diagnosis

- Feature values: age, gender, history, symptom1-severity, symptom2-severity, test-result1, test-result2
- Label: disease

Email spam detection

- Feature values: sender-domain, length, #images, keyword_1, keyword_2, …, keyword_n
- Label: spam or not-spam

Credit card fraud detection

- Feature values: user, location, item, price
- Label: fraud or okay

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

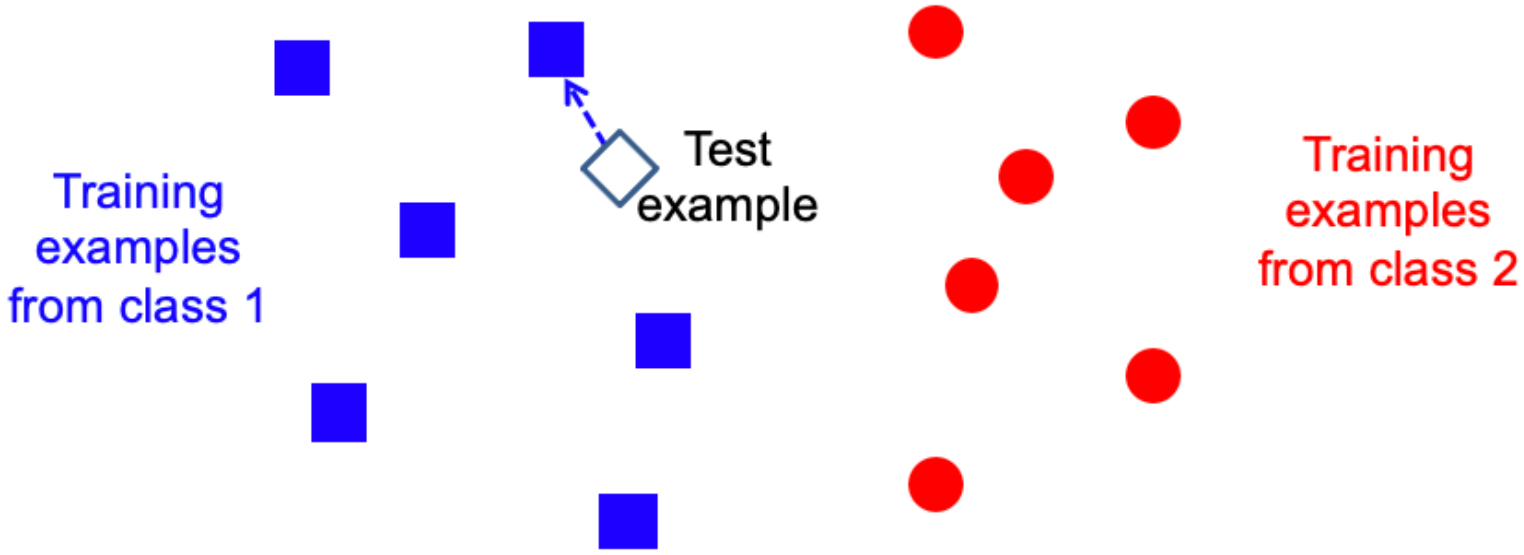
Algorithms for classification

Despite similarity of problem statement to regression, non-numerical nature of classification leads to completely different approaches

- K-nearest neighbors
- Decision trees
- Naïve Bayes
- Deep neural networks
- … and others

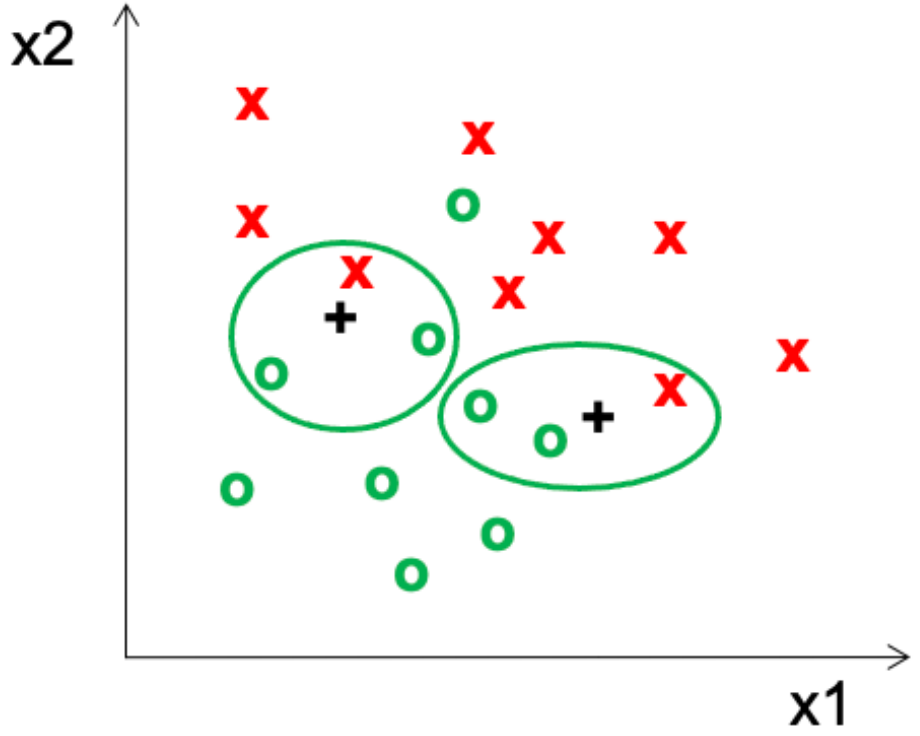
https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

K-Nearest Neighbors (KNN)



https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

K-Nearest Neighbors (KNN)



https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

K-Nearest Neighbors (KNN)

For any pair of data items i_1 and i_2, from their feature values compute *distance*(i_1,i_2)

Example:

Features - gender, profession, age, income, postal-code

person$_1$ = (male, teacher, 47, $25K, 94305)

person$_2$ = (female, teacher, 43, $28K, 94309)

distance(person$_1$, person$_2$)

distance() can be defined as inverse of *similarity*()

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

K-Nearest Neighbors (KNN)

Features - gender, profession, age, income, postal-code

$person_1$ = (male, teacher, 47, $25K, 94305) buyer

$person_2$ = (female, teacher, 43, $28K, 94309) non-buyer

Remember training data has labels

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

K-Nearest Neighbors (KNN)

Features - gender, profession, age, income, postal-code

$person_1$ = (male, teacher, 47, $25K, 94305) buyer

$person_2$ = (female, teacher, 43, $28K, 94309) non-buyer

Remember training data has labels

To classify a new item *i* : In the labeled data find the K closest items to *i*, assign most frequent label

$person_3$ = (female, doctor, 40, $40K, 95123)

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

KNN example

- City temperatures – France and Germany
- Features: longitude, latitude
- Distance is Euclidean distance

  $distance([o_1,a_1],[o_2,a_2]) = sqrt((o_1-o_2)^2 + (a_1-a_2)^2)$

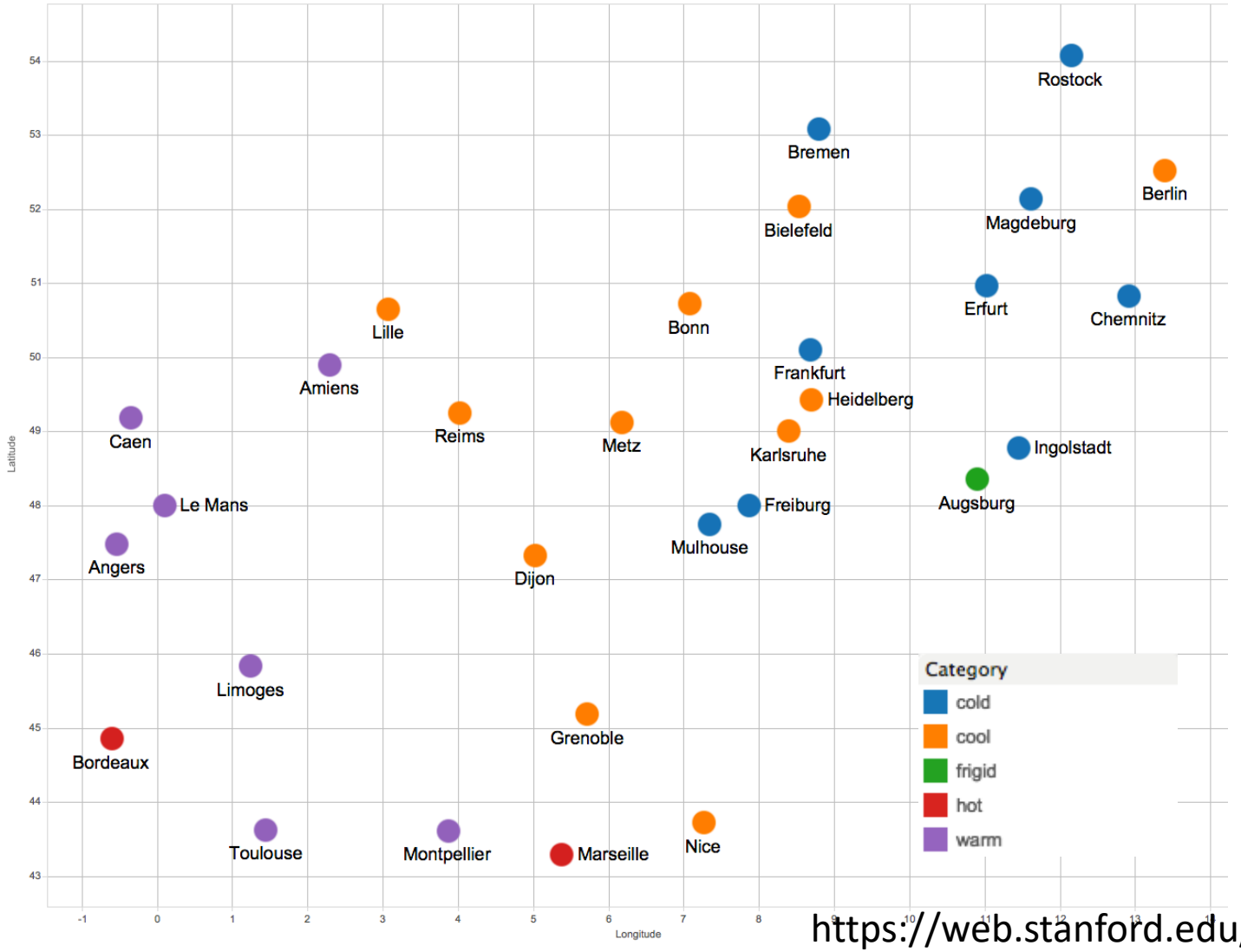
  = actual distance in x-y plane
- Labels: frigid, cold, cool, warm, hot

Nice (7.27, 43.72) cool
Toulouse (1.45, 43.62) warm
Frankfurt (8.68, 50.1) cold
……

Predict temperature category from longitude and latitude

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

KNN example



https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

KNN summary

To classify a new item *i*: find K closest items to *i* in the labeled data, assign most frequent label

- No hidden complicated math!
- Once distance function is defined, rest is easy
- Though not necessarily efficient

  Real examples often have thousands of features

  - Medical diagnosis: symptoms (yes/no), test results
  - Email spam detection: words (frequency)

  Database of labeled items might be enormous

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

Regression using KNN

Features - gender, profession, age, income, postal-code

$person_1$ = (male, teacher, 47, $25K, 94305) $250

$person_2$ = (female, teacher, 43, $28K, 94309) $100

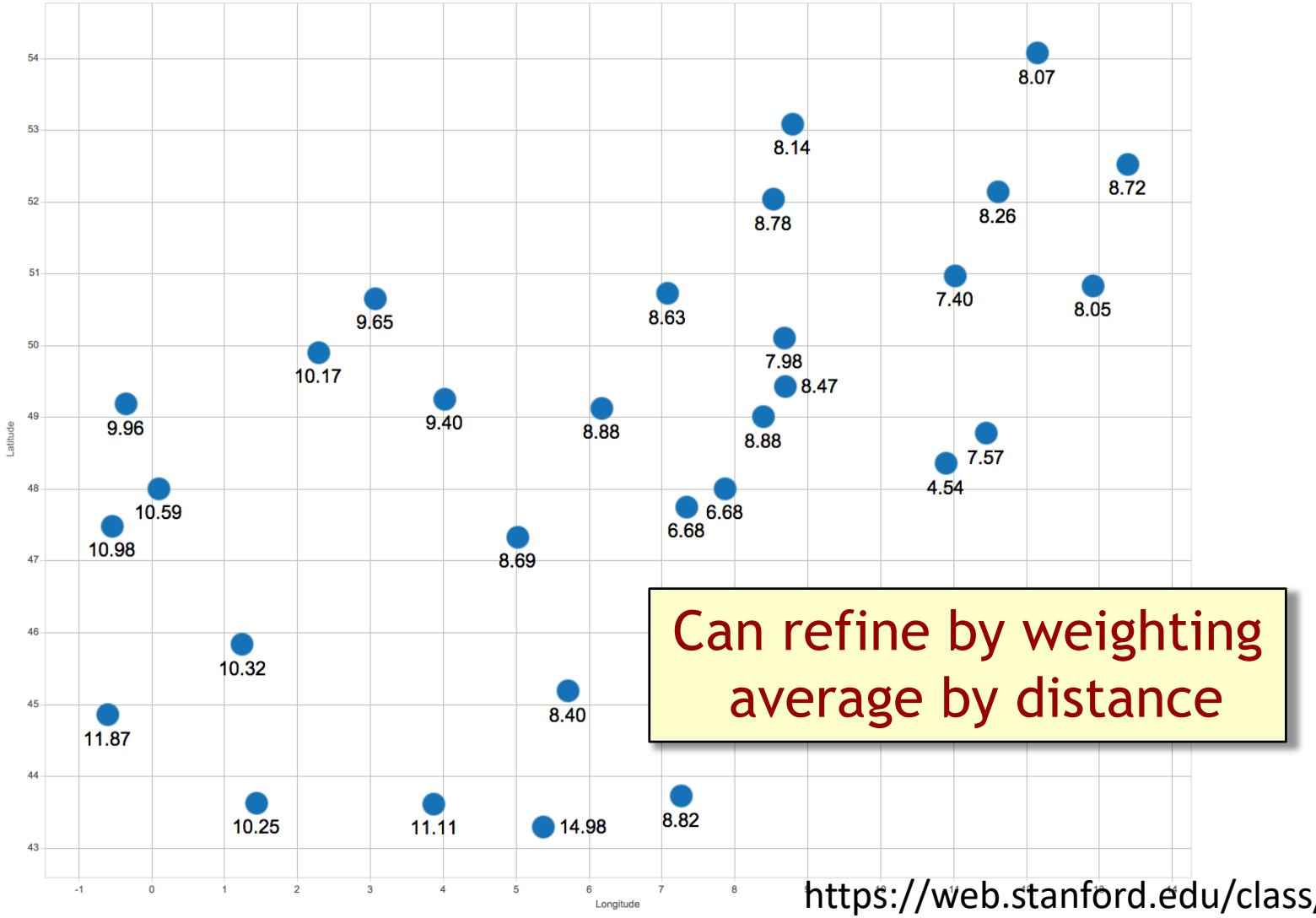
Remember training data has labels

To classify a new item *i*, find K closest items to *i* in the labeled data, assign average value of labels

$person_3$ = (female, doctor, 40, $40K, 95123)

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

Regression using KNN



Can refine by weighting average by distance

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

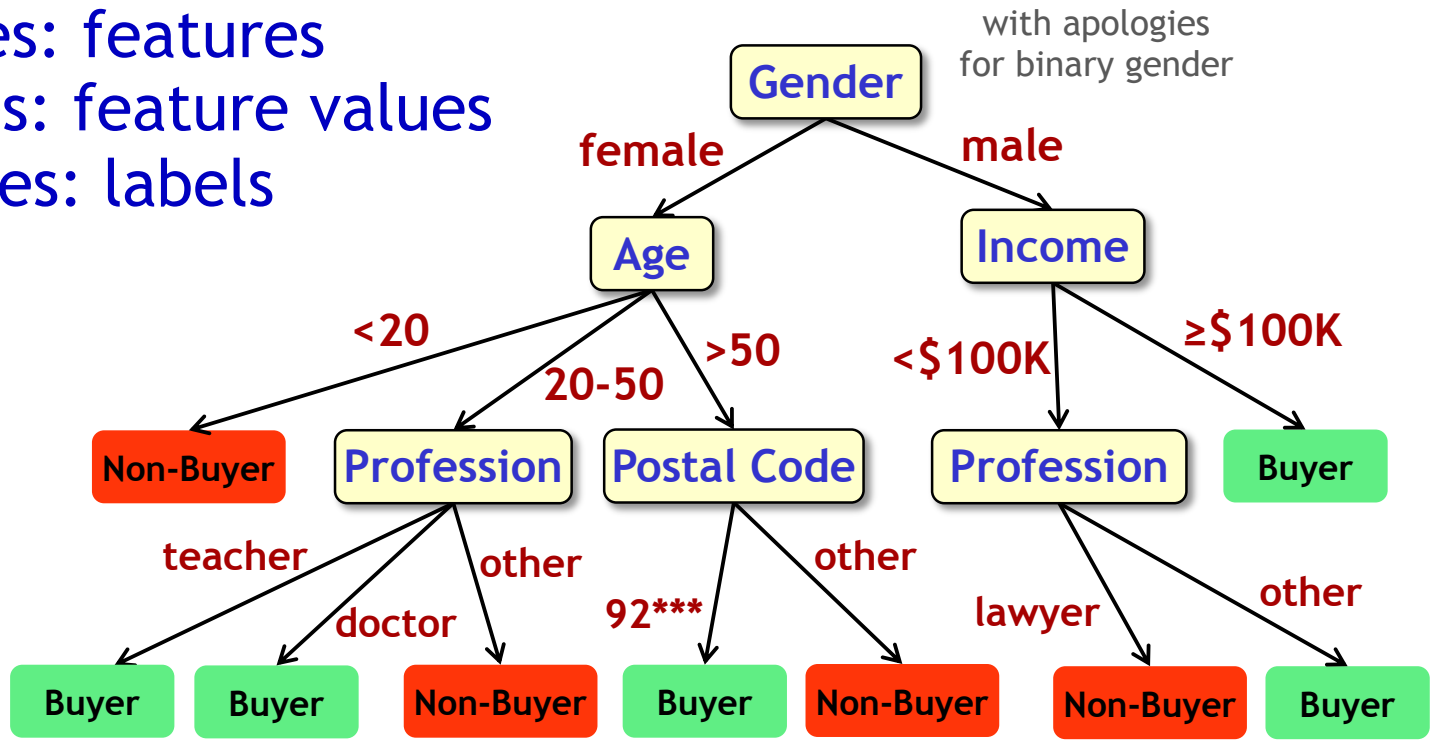
Decision trees

- Use the training data to construct a decision tree
- Use the decision tree to classify new data

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

Decision trees

Nodes: features
Edges: feature values
Leaves: labels



New data item to classify:
Navigate tree based on feature values

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

Decision trees

Primary challenge is building good decision trees from training data

- Which features and feature values to use at each choice point
- HUGE number of possible trees even with small number of features and values

Common approach: “forest” of many trees, combine the results

- Still impossible to consider all trees

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

Deep Neural Networks (DNNs)

Neural Networks

- Machine learning method modeled loosely after connected neurons in brain
- Invented decades ago but not successful
- Recent resurgence enabled by:
  - Powerful computing that allows for many layers (making the network “deep”)
  - Massive data for effective training

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

Deep Neural Networks (DNNs)

= Deep Learning

- Huge breakthrough in effectiveness and reach of machine learning
- Accurate predictions across many domains
- Big plus: Automatically identifies features in unstructured data
  (e.g., images, videos, text)

https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

Other common methods

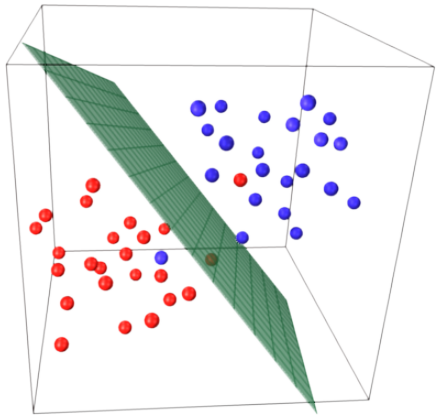
Logistic Regression

- Typically for two labels only ("binary classifier")
- Recall regression model is function *f* from predictor values to numeric output value
- Labels L_1 + L_2, from training data obtain function:

  f(*feature-values*) = *probability of item having label* L_1

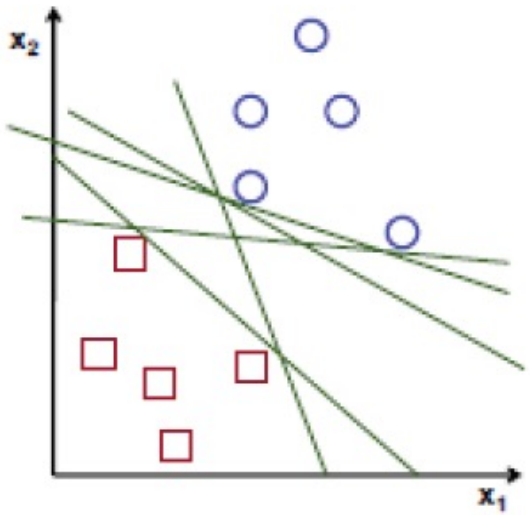
Support Vector Machine

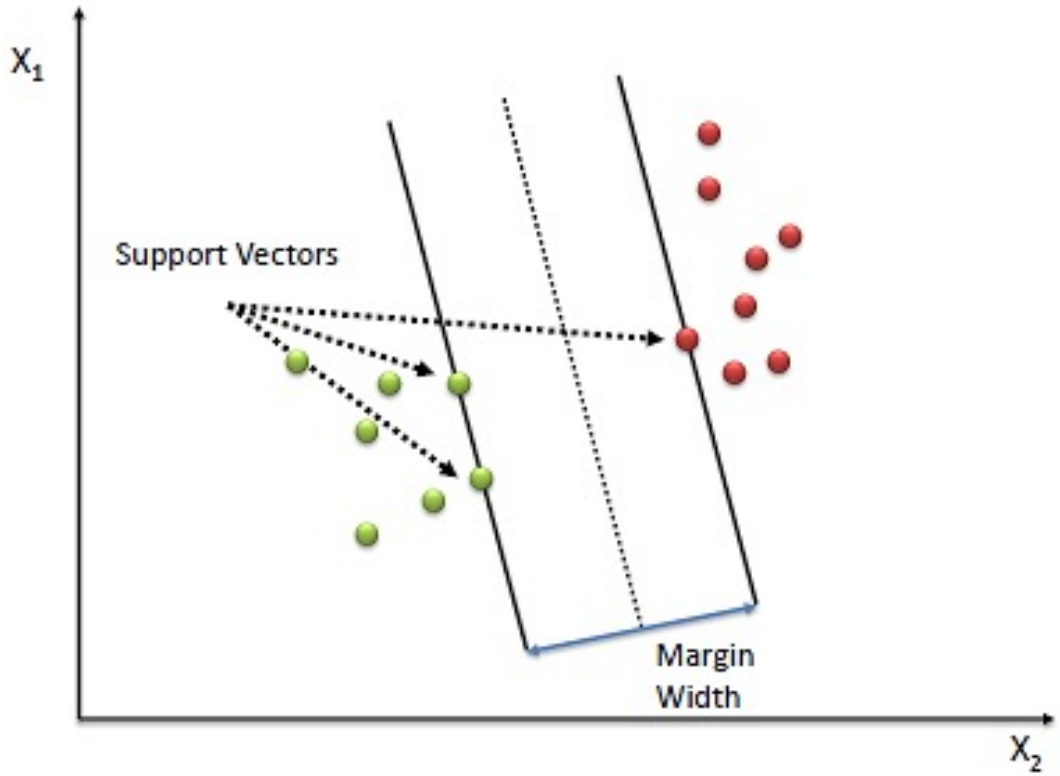
- Also for binary classification
- Features = multidimensional space
- From training data SVM finds hyper-plane that best divides space according to labels



https://web.stanford.edu/class/cs102/lectureslides/ClassificationSlides.pdf

Support vector machines (SVMs)





Slide Credit

Deniz Akpinaroglu