CS 260: Foundations of Data Science

Prof. Thao Nguyen

Fall 2025

HAVERFORD
COLLEGE

Materials by Sara Mathieson

Admin

- **Lab 8** grades & feedback posted on Moodle
- **End-of-semester survey** (link on Piazza)

Outline for today

- Gaussian Mixture Models (GMMs)
- Kernel Density Estimation (KDE)
- Missing data
- Neural networks

Outline for today

- Gaussian Mixture Models (GMMs)
- Kernel Density Estimation (KDE)
- Missing data
- Neural networks

Problems with K-means

- Does not account for different cluster sizes, variances, and shapes
- Does not allow points to belong to multiple clusters
- Not generative (cannot create a new data point)

Discriminative vs. Generative Algorithms

- Discriminative: finds a decision boundary
  - Logistic regression, K-means
- Generative: estimates probability distributions
  - Naïve Bayes, Gaussian Mixture Models

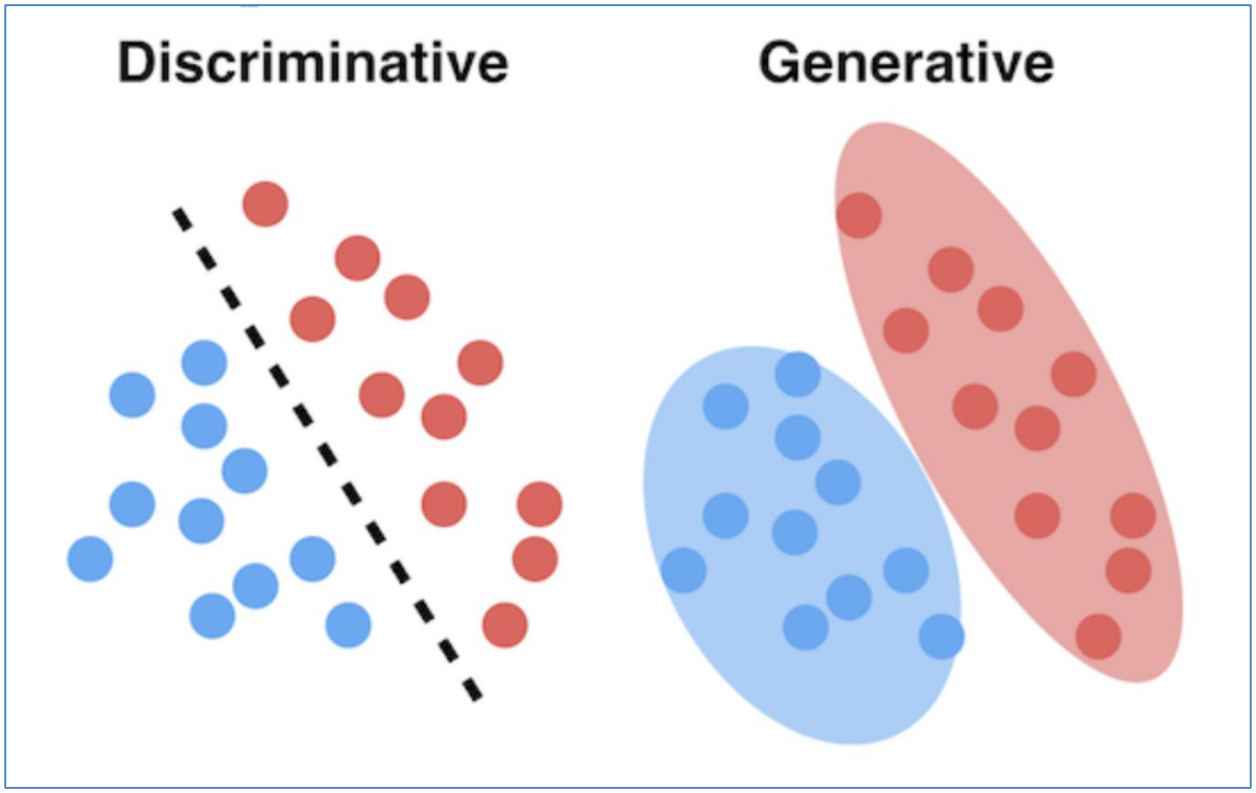


Figure: Ameet Soni

Gaussian Mixture Models (GMMs)

$$p(\vec{x}_i) = \sum_{k=1}^{K} p(\vec{x}_i, k) = \sum_{k=1}^{K} p(k)p(\vec{x}_i|k) = \sum_{k=1}^{K} \pi_k N\left(\vec{x}_i \middle| \vec{\mu}_k, \sigma_k^2\right)$$

prior over cluster sizes

cluster membership

assume Gaussian distribution

- Maximize likelihood:

$$L(X) = \prod_{i=1}^{n} p(\vec{x}_i) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k N\left(\vec{x}_i \middle| \vec{\mu}_k, \sigma_k^2\right)$$

Model parameters

Gaussian Mixture Models (GMMs)

- Initialization step: for each cluster
  - Probability $\pi_k = 1/K$ (uniform prior)
  - Mean $\vec{\mu}_k$ = choose random point
  - Variance σ_k^2 = sample variance of all points closest to each mean

- E-step: "soft" assignment

$$w_{ik} = p(k|\vec{x}_i) = \frac{p(k)p(\vec{x}_i|k)}{p(\vec{x}_i)} = \frac{\pi_k N(\vec{x}_i|\vec{\mu}_k, \sigma_k^2)}{\sum_{j=1}^{K} \pi_j N(\vec{x}_i|\vec{\mu}_j, \sigma_j^2)}$$

probability that $\vec{x}_i$ came from cluster k

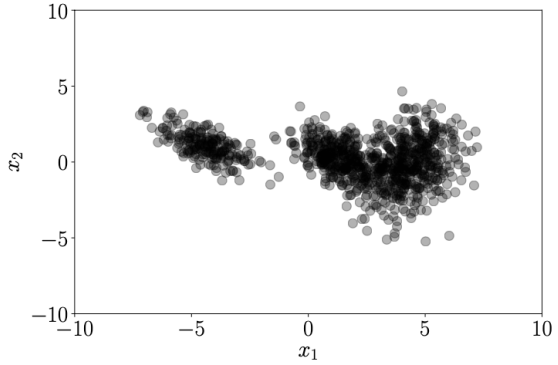
Gaussian Mixture Models (GMMs)

- M-step: parameter update

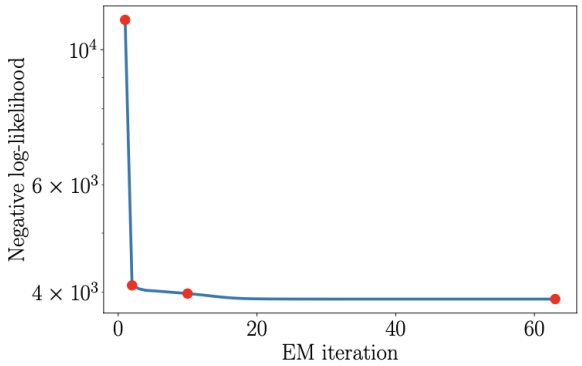
  $N_k = \sum_{i=1}^{n} w_{ik}$ (# of points assigned to cluster k)

  - $\pi_k = \frac{N_k}{n}$
  - $\vec{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{n} w_{ik} \vec{x}_i$
  - $\sigma_k^2 = \frac{1}{N_k} \sum_{i=1}^{n} w_{ik} \left(\vec{x}_i - \vec{\mu}_k \right)^2$

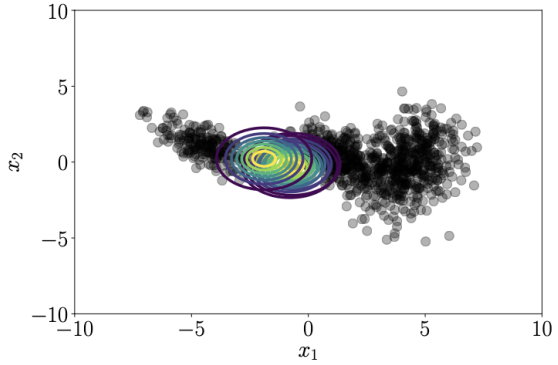
use updated mean



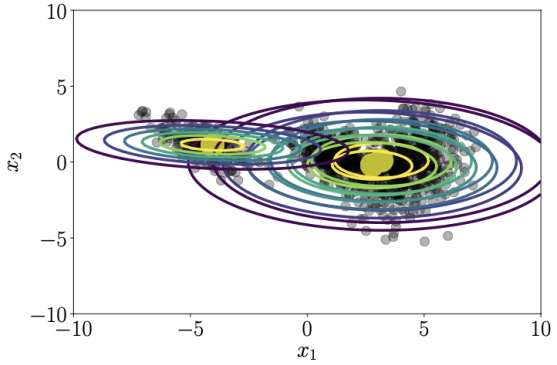
(a) Dataset.



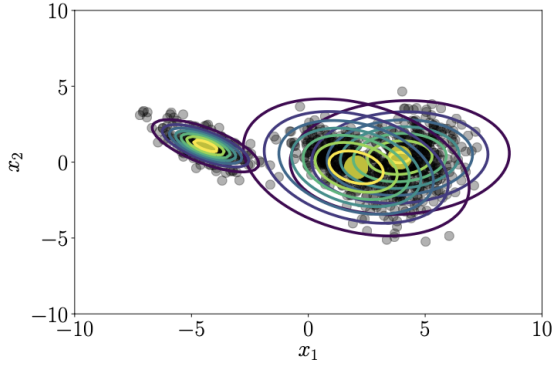
(b) Negative log-likelihood.



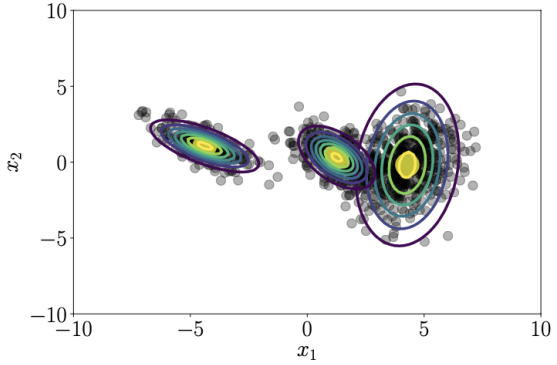
(c) EM initialization.



(d) EM after one iteration.



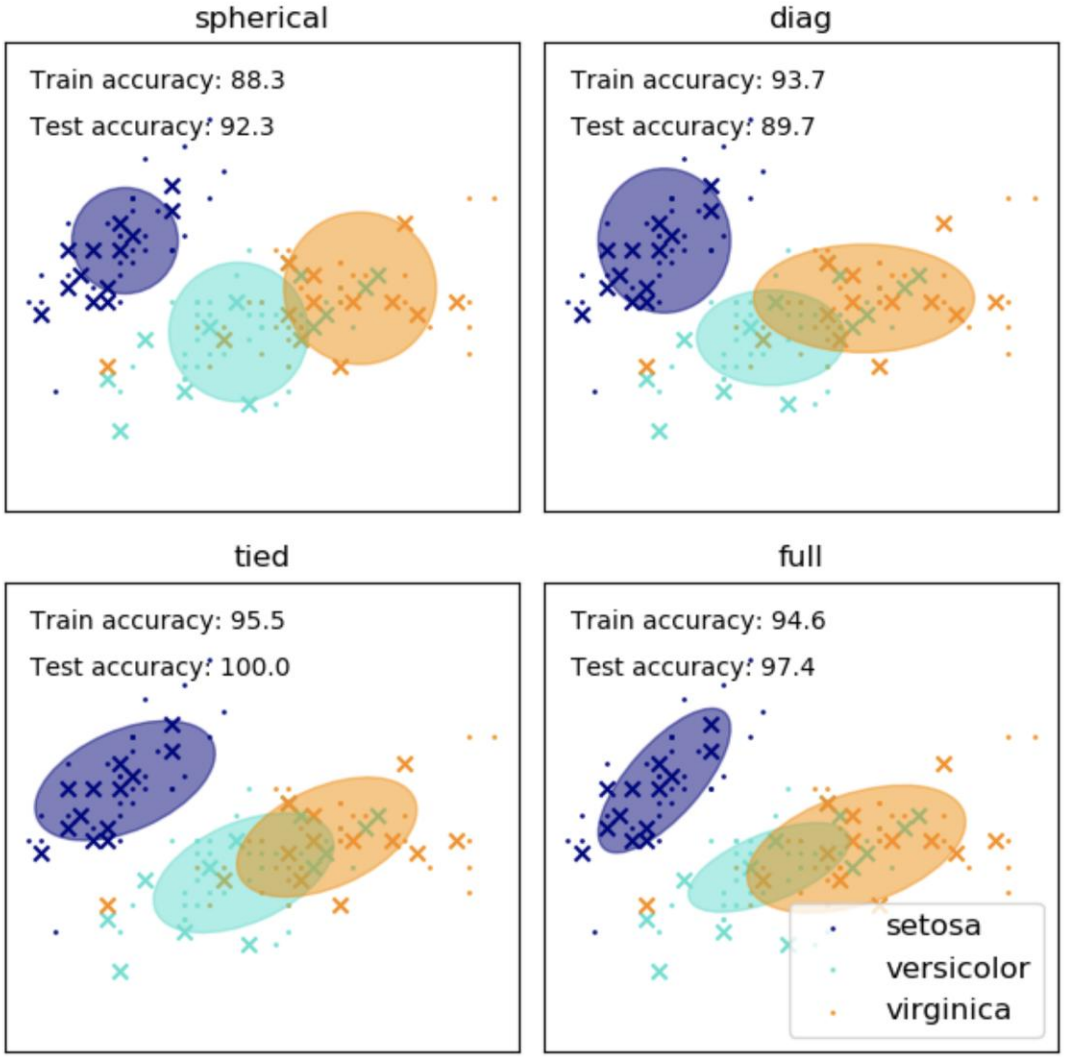
(e) EM after 10 iterations.



(f) EM after 62 iterations.

Figure 11.7 from MML textbook

Example of GMMs with different covariance constraints on the Iris flower data



https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_covariances.html#sphx-glr-auto-examples-mixture-plot-gmm-covariances-py

Generative Process

- Sample cluster k using $[\pi_1, \pi_2, \ldots, \pi_k]$
- Sample x from $N(\vec{\mu}_k, \sigma_k^2)$

Outline for today

- Gaussian Mixture Models (GMMs)
- Kernel Density Estimation (KDE)
- Missing data
- Neural networks

KDE (Kernel Density Estimation)

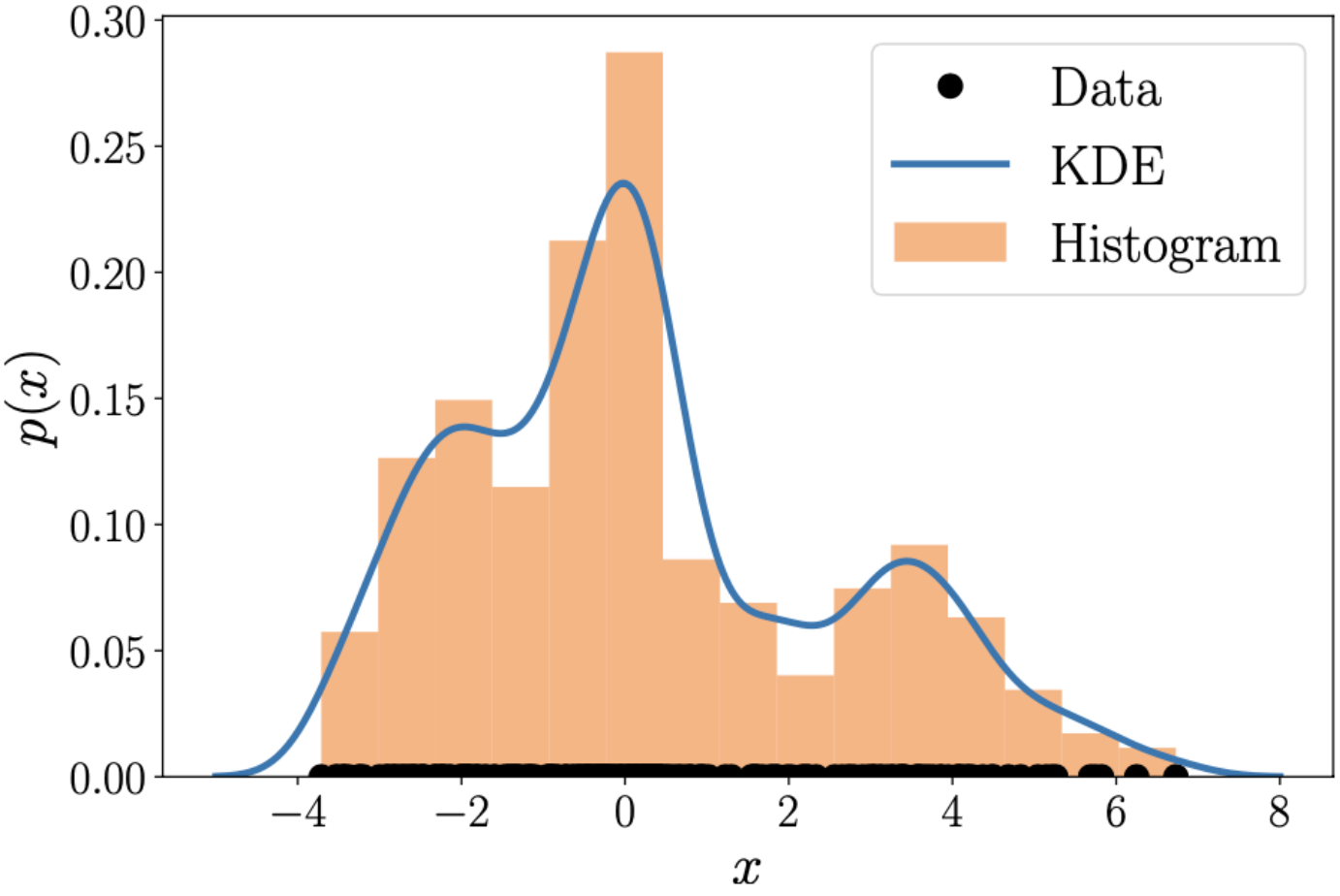


Figure 11.9 from MML textbook

KDE (Kernel Density Estimation)

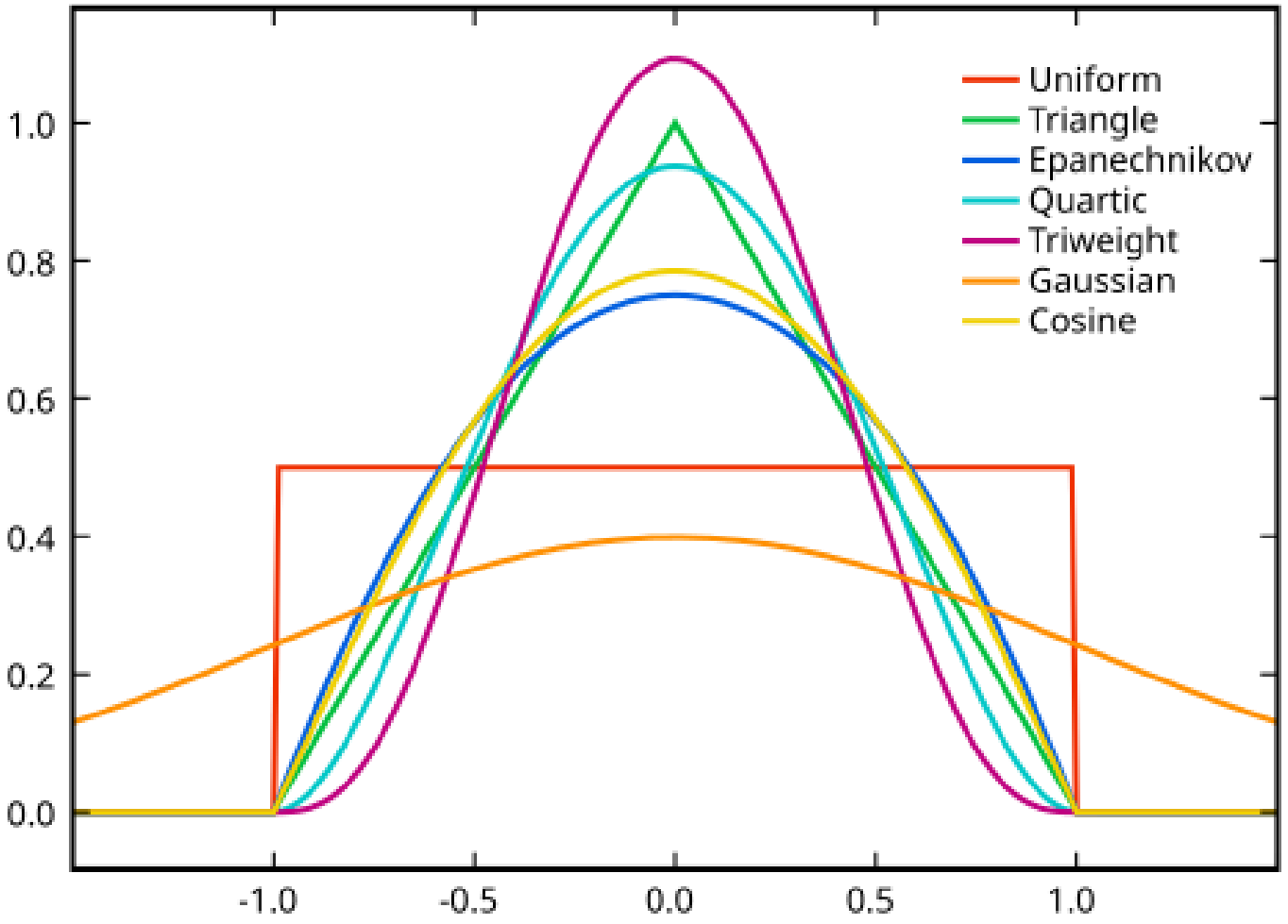
$$p(x) = \frac{1}{nh}\sum_{i=1}^{n} k\left(\frac{x - x_i}{h}\right)$$

given point

kernel: weighting function

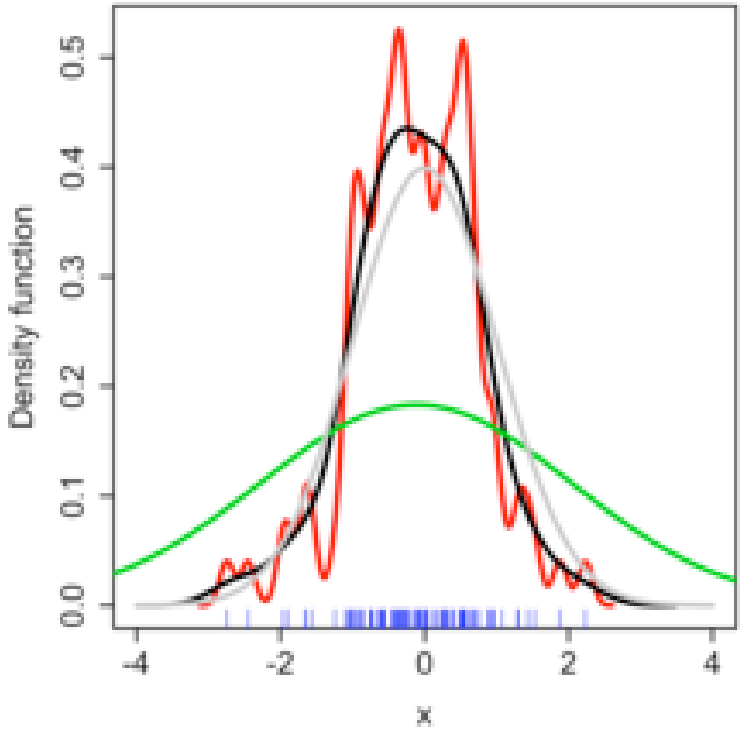
width: smoothing parameter

Commonly used kernel functions



Wikipedia

Width selection



Kernel density estimate (KDE) with different bandwidths of a random sample of 100 points from a standard normal distribution. Grey: true density (standard normal). Red: KDE with h=0.05. Black: KDE with h=0.337. Green: KDE with h=2.

Wikipedia

Outline for today

- Gaussian Mixture Models (GMMs)
- Kernel Density Estimation (KDE)
- Missing data
- Neural networks

Types of missing data

- MCAR: Missing Completely At Random. Not related to:
  - Specific values
  - Observed responses
- MAR: Missing At Random. Not related to:
  - Specific values
- MNAR: Missing Not At Random

Reference: “The prevention and handling of the missing data” Kang (2013)

Techniques for handling missing data

- Try to prevent the problem in the first place
  - Careful study design, follow-up with participants, etc
- Omit rows with missing data (reduces *n*)
- Omit only when value is needed
  - e.g. Naïve Bayes, per-feature estimates
- Mean substitution (per feature)

Reference: "The prevention and handling of the missing data" Kang (2013)

Techniques for handling missing data

- Imputation
  - Use similar examples to guess the missing values
  - Can be done locally or globally
- Last observation carried forward
  - Useful for time-series data

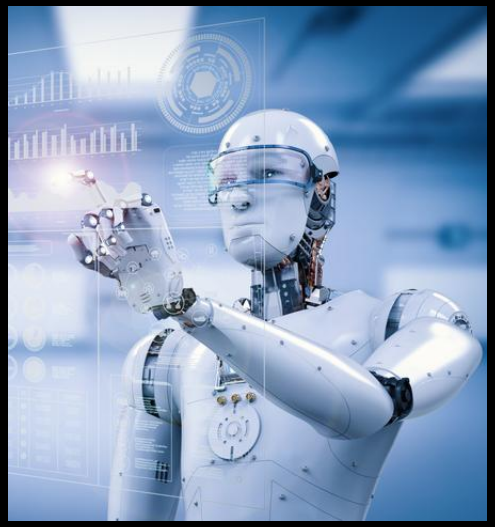
Reference: “The prevention and handling of the missing data” Kang (2013)

Outline for today

- Gaussian Mixture Models (GMMs)
- Kernel Density Estimation (KDE)
- Missing data
- Neural networks

Adapted from: "Know Your Meme"

MACHINE LEARNING



What society thinks I do

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0$$

This implies that

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}.$$

As for the derivative with respect to b, we obtain

$$\frac{\partial}{\partial b} \mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0.$$

If we take the definition of w in Equation (9) and plug that back into the Lagrangian (Equation 8), and simplify, we get

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - b \sum_{i=1}^{m} \alpha_i y^{(i)}.$$

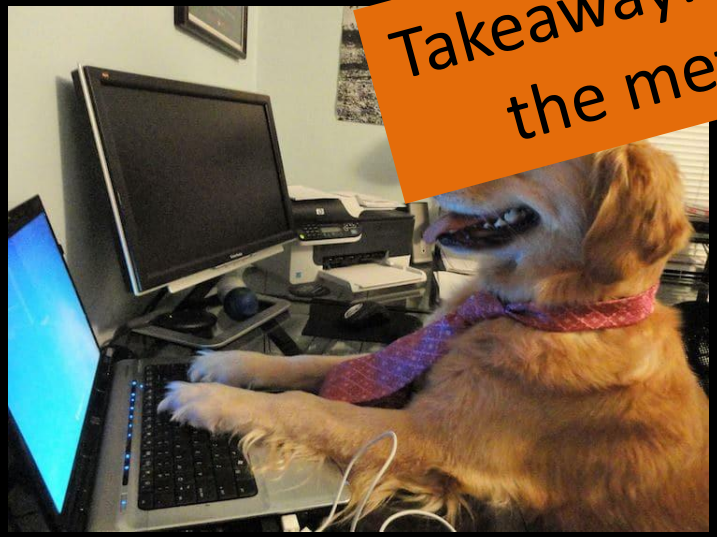
But from Equation (10), the last term must be zero, so we obtain

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_{(}$$

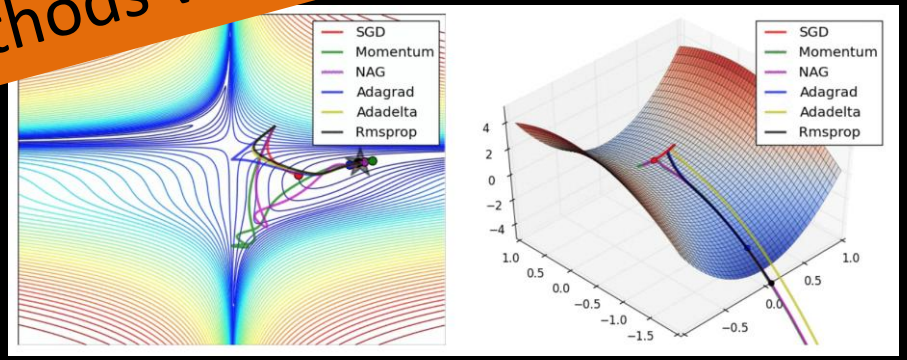
Wha

t other computer scientists think I do

Takeaway: we should understand the methods we are using!



What mathematicians think I do

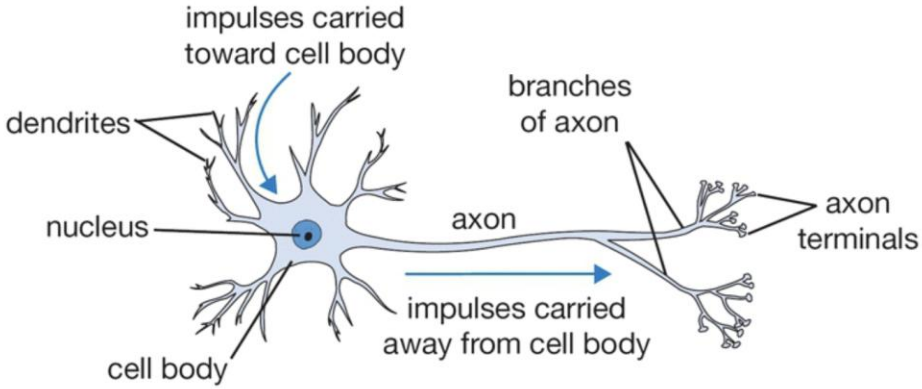


What I think I do

```
>>> from sklearn import svm
>>> import tensorflow as tf
```

What I really do

Biological Inspiration for Neural Networks



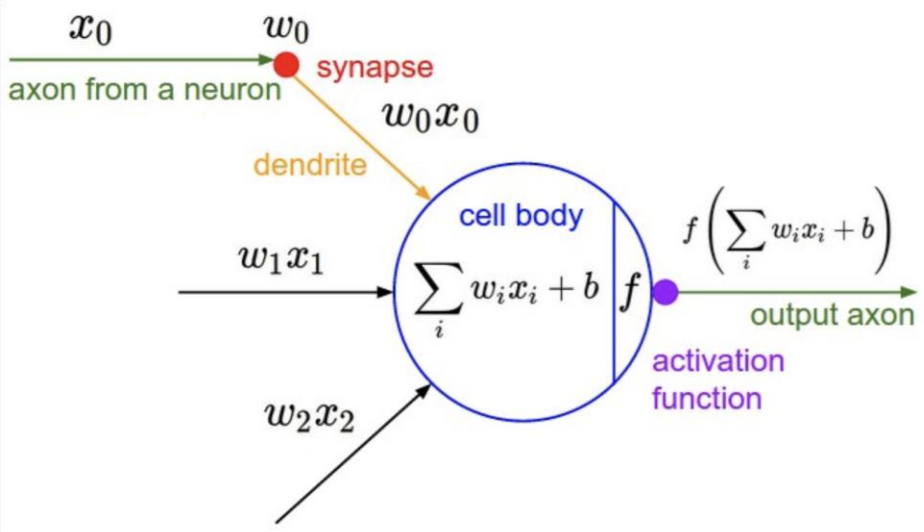


Figure: Stanford CS231n http://cs231n.github.io/neural-networks-1/

Goal: learn from complicated inputs

X_1

X_2

X_3

X_4

X_5

X_6

input data

?

Y_1 glasses?

Y_2 smiling?

Y_3 identity?

parameters

Image: Labeled Faces in the Wild (UMass)

Idea: transform data into lower dimension

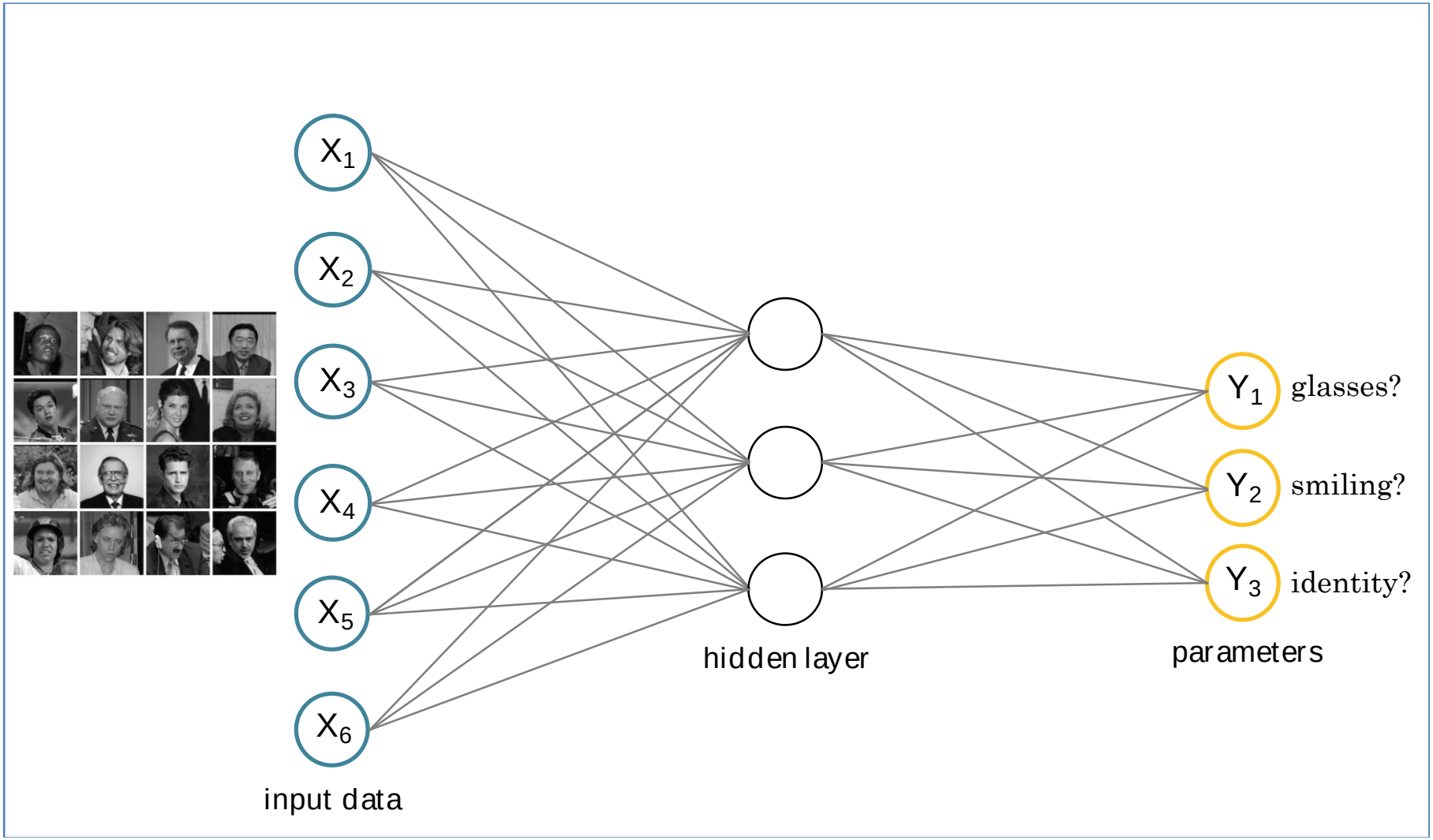


Image: Labeled Faces in the Wild (UMass)

Multi-layer networks = “deep learning”

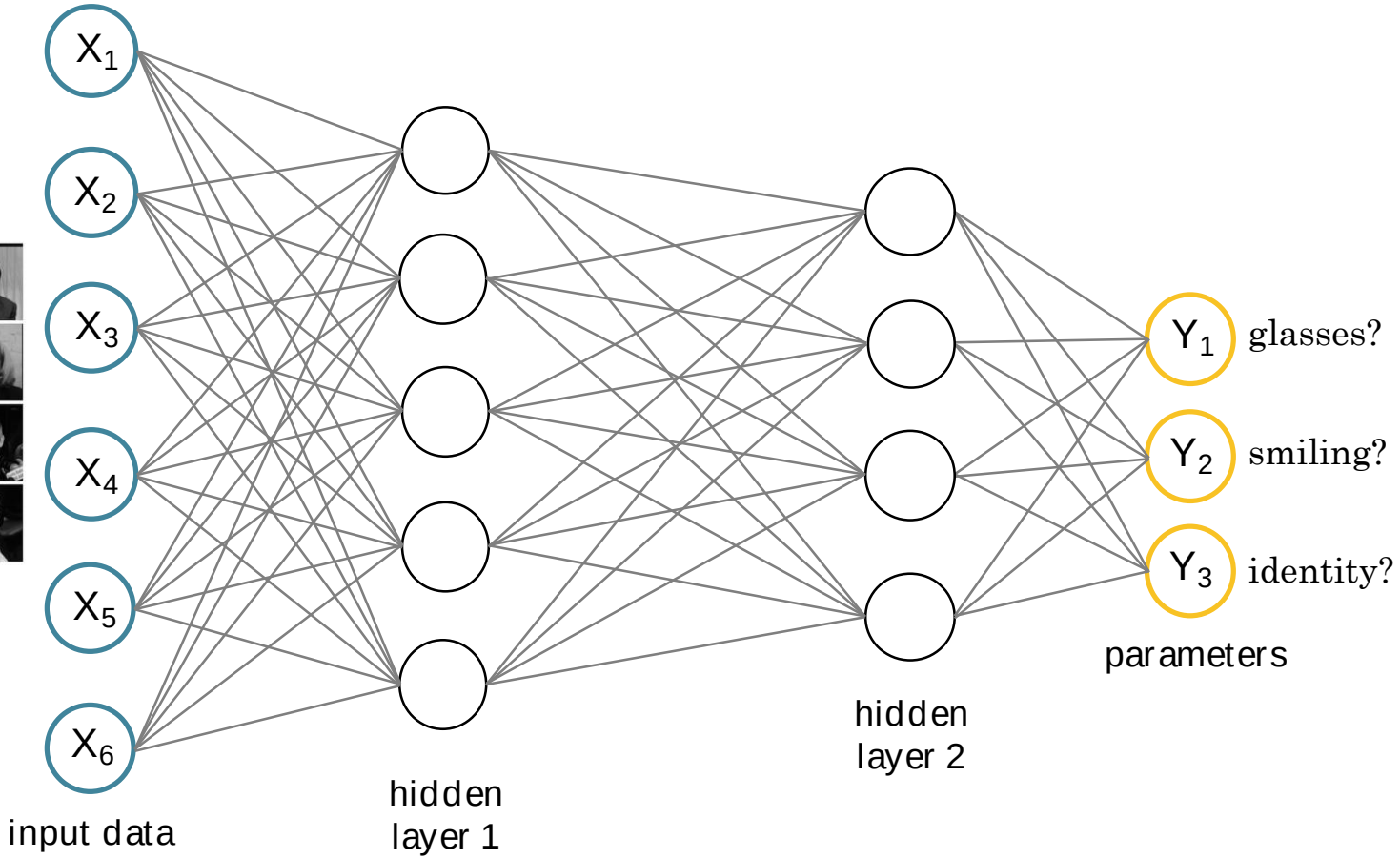
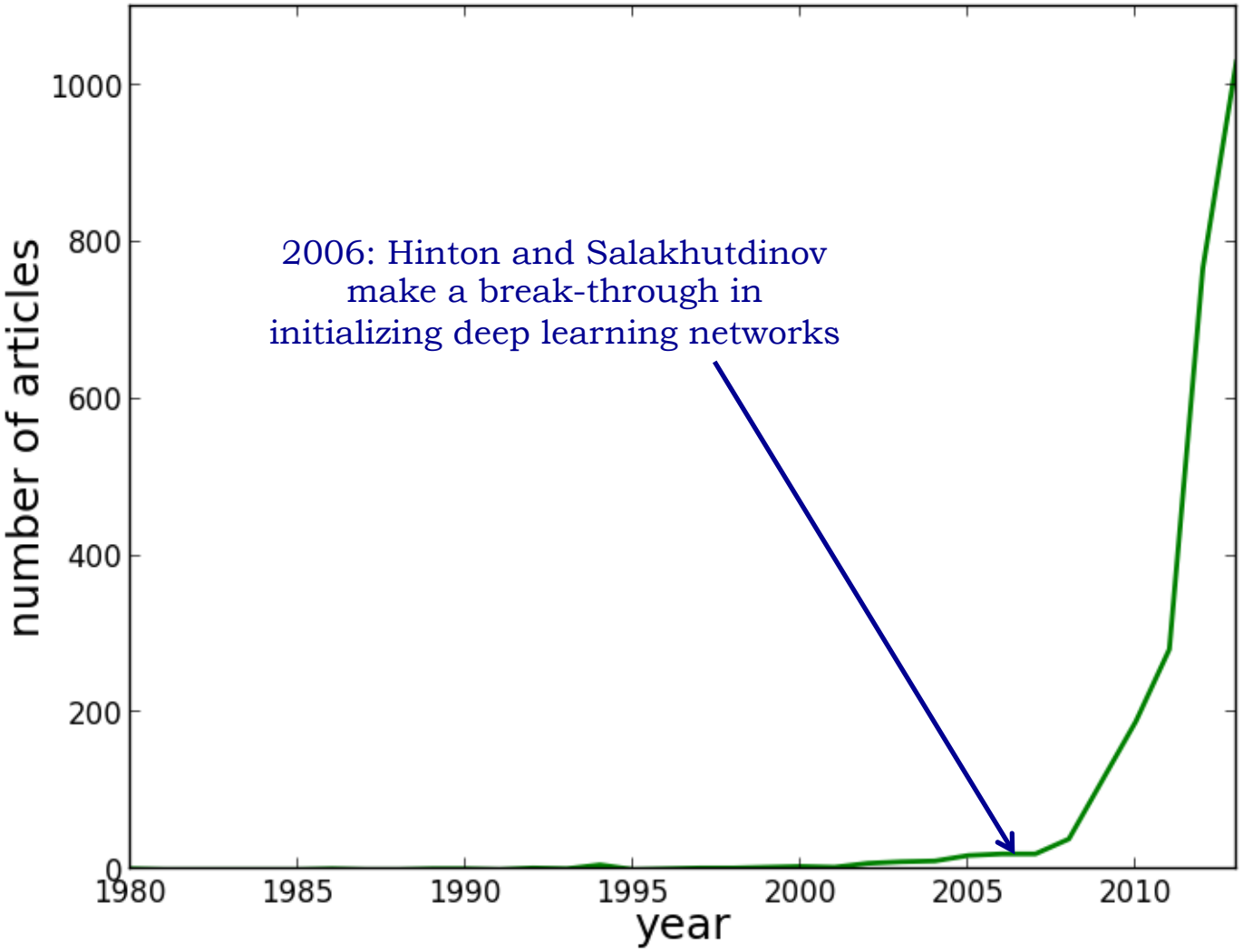


Image: Labeled Faces in the Wild (UMass)

History of Neural Networks

- Perceptron can be interpreted as a simple neural network
- Misconceptions about the weaknesses of perceptrons contributed to declining funding for NN research
- Difficulty of training multi-layer NNs contributed to second setback
- Mid 2000's: breakthroughs in NN training contribute to rise of "deep learning"

Number of papers that mention "deep learning" over time



Big picture for today

- Neural networks can approximate any function!

Big picture for today

- Neural networks can approximate any function!

- For our purposes in ML, we want to use them to approximate a function from our inputs to our outputs

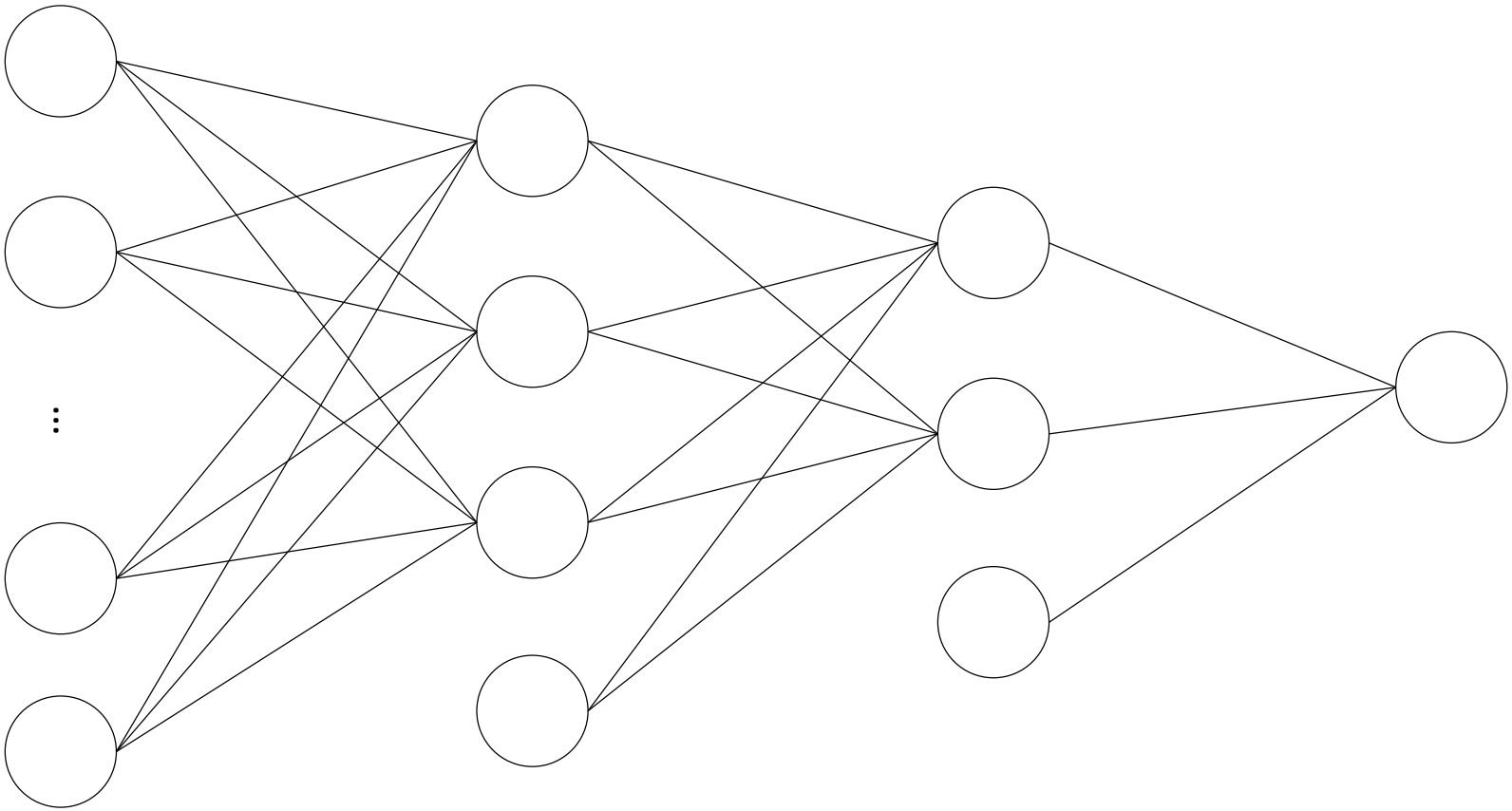
Big picture for today

- Neural networks can approximate any function!
- For our purposes in ML, we want to use them to approximate a function from our inputs to our outputs
- We will train our network by asking it to minimize the loss between its output and the true output

Big picture for today

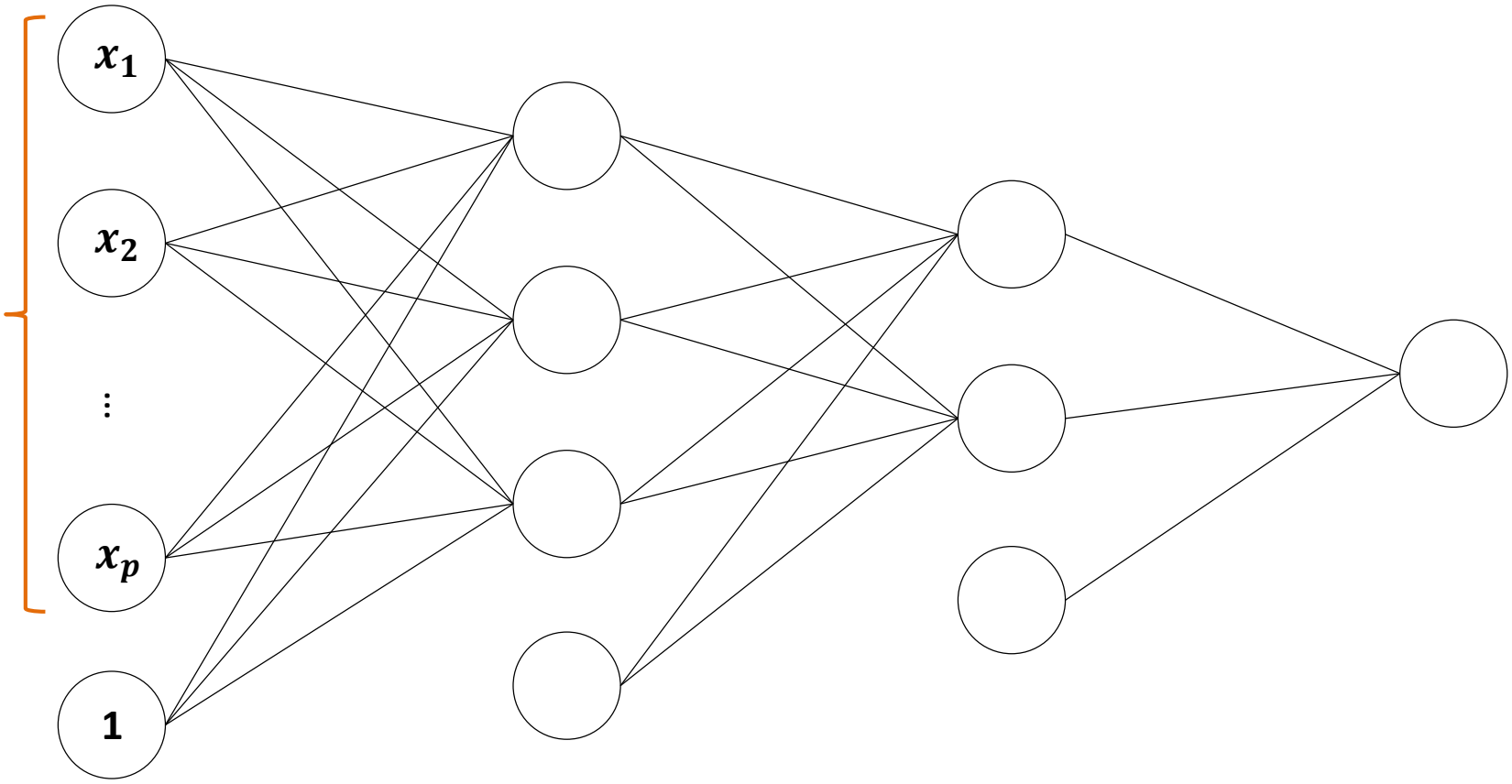
- Neural networks can approximate any function!
- For our purposes in ML, we want to use them to approximate a function from our inputs to our outputs
- We will train our network by asking it to minimize the loss between its output and the true output
- We will use SGD-like approaches to minimize loss

Fully Connected Neural Network Architecture



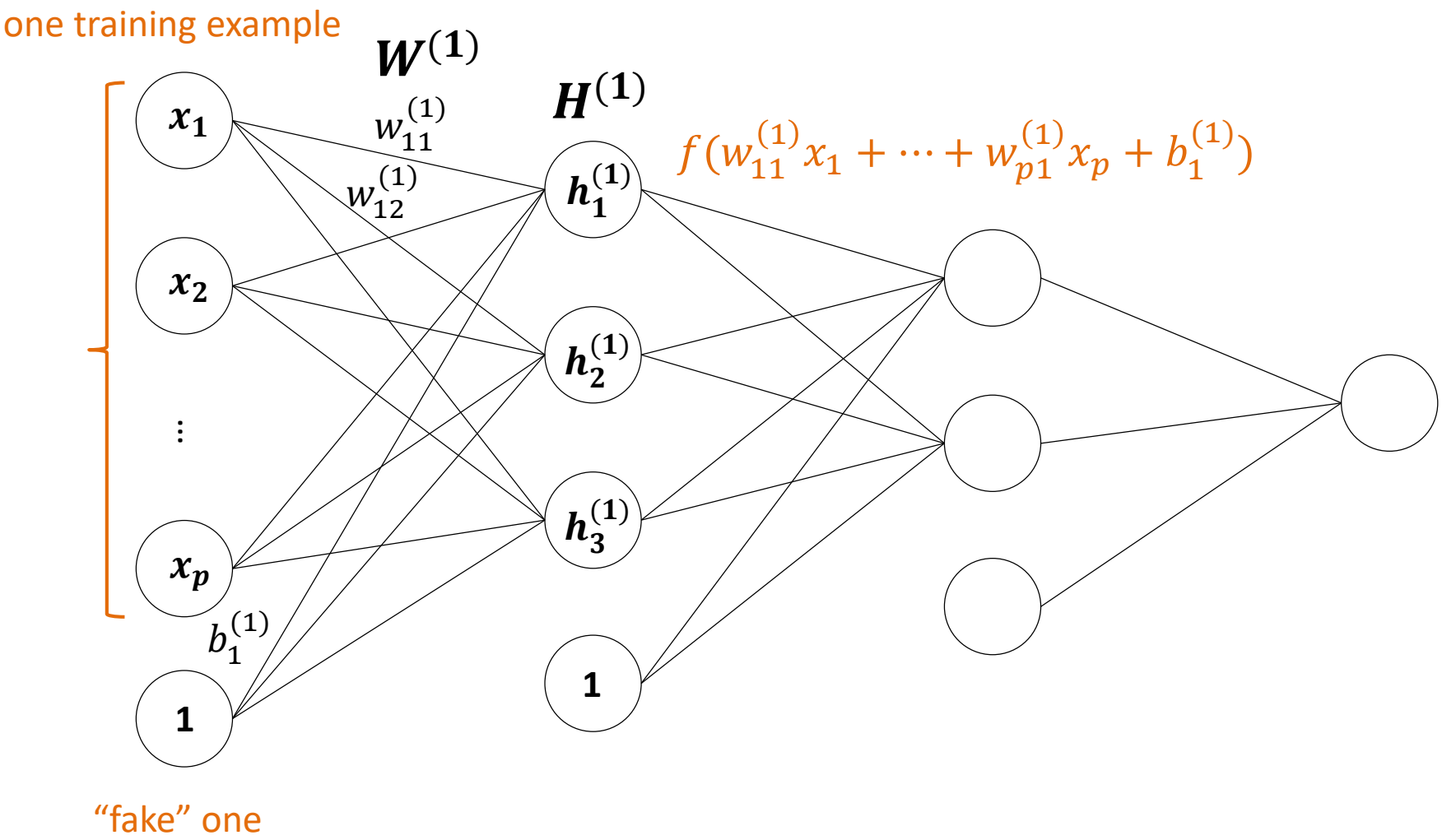
Fully Connected Neural Network Architecture

one training example

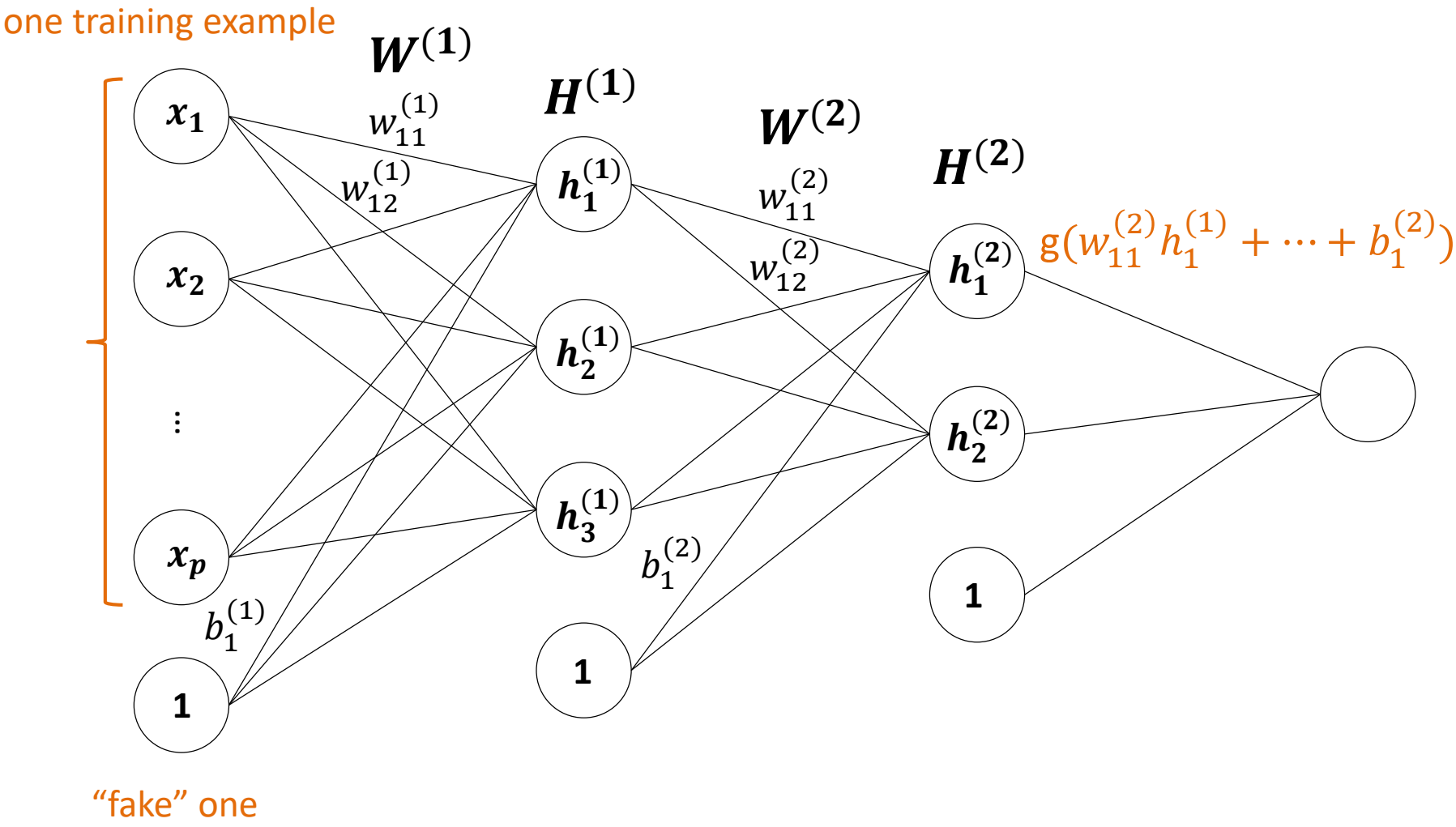


"fake" one

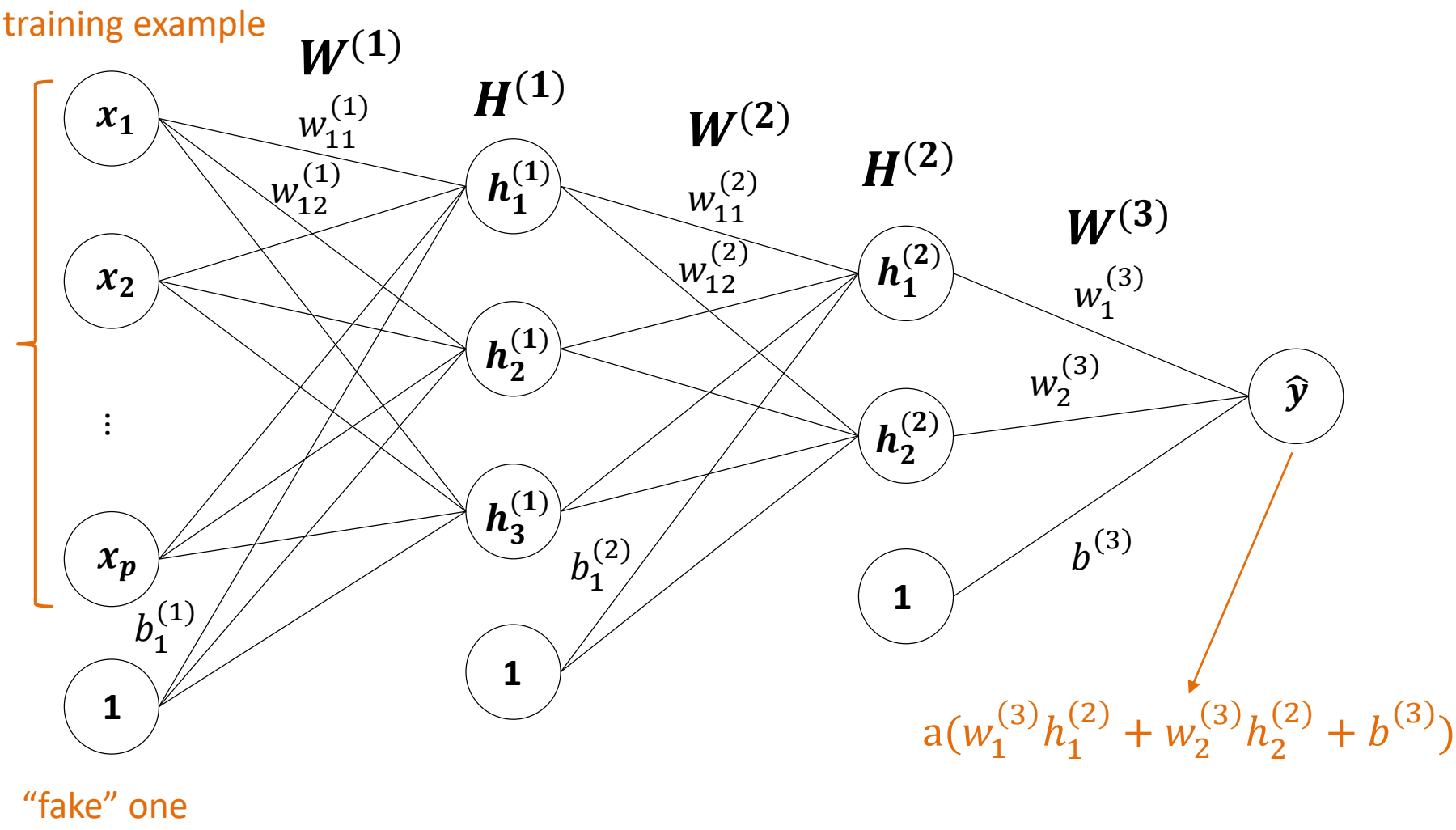
Fully Connected Neural Network Architecture



Fully Connected Neural Network Architecture



Fully Connected Neural Network Architecture



Layer Output

- $H^{(1)} = a\left(W^{(1)}X^T + B^{(1)}\right)$

p_1 = # of nodes in layer 1

activation function

$p_1 \times p$ $p \times n$ $p_1 \times n$

$p_1 \times n$

- $H^{(2)} = a\left(W^{(2)}H^{(1)} + B^{(2)}\right)$

- $\hat{y} = a\left(W^{(3)}H^{(2)} + \vec{b}^{(3)}\right)$