CS 383: Machine Learning

Prof Adam Poliak

Fall 2024

11/11/2024

Lecture 23

Announcements – Remaining Assignments

HW06: extending deadline to Friday 11/15

HW07: due Friday 11/22

HW08: due Friday 12/06

Project Proposal due Thursday 11/14

Agenda/Outline

- SVMs
- Cross-Validation
- Neural Networks

Dual form

$$\max W(\vec{\alpha}) = \sum_{i}^{n} \alpha_i - \frac{1}{2}\sum_{i}^{n}\sum_{j}^{n} y_i y_j \alpha_i \alpha_j \boxed{\vec{x_i}\,\vec{x_j}}$$

$$s.t.\, \alpha_i > 0\ \forall i\ \&\ \sum_{i}^{n} \alpha_i y_i = 0$$

Kernel Idea

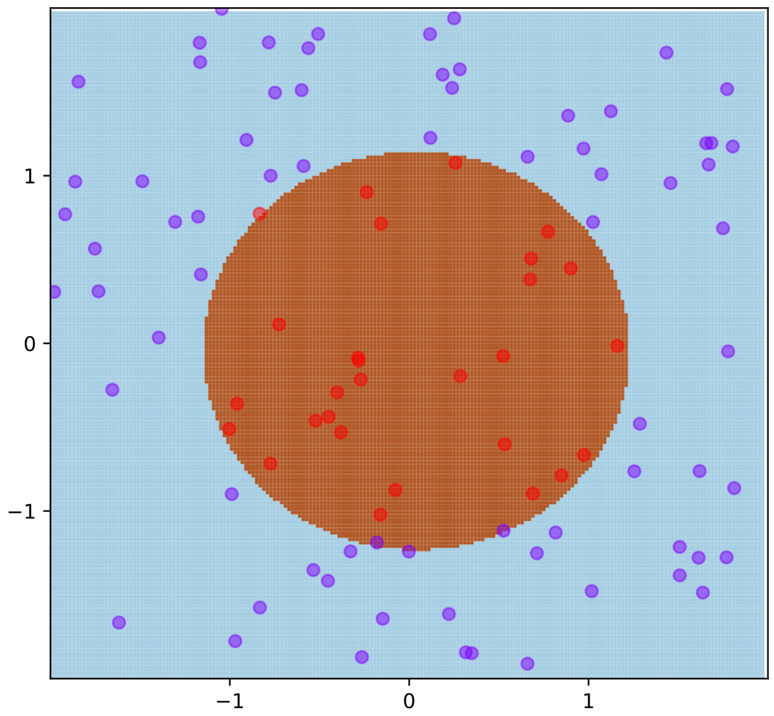
By solving the dual form of the problem, we have seen how all computations can be done in terms of inner products between examples

One example of an inner product is the dot product, which is the linear version of SVMs

But there are many others!

Intuition: if points are close together, their kernel function will have a large value (measure of similarity)

Kernel Trick example

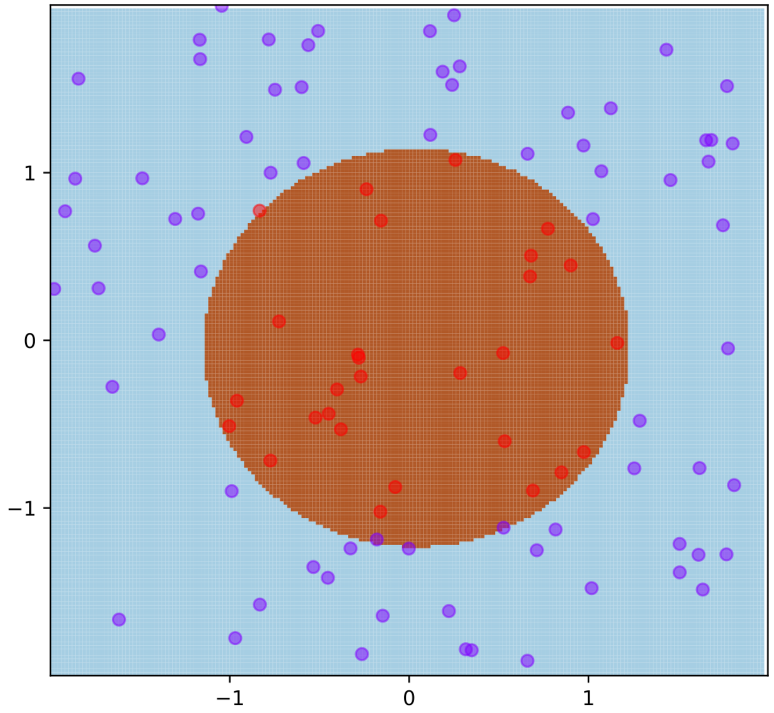


Original feature space

Image: Shiyu Ji (wikipedia)

Kernel Trick example

Feature mapping: $\varphi(\boldsymbol{x}) = (x_1,\ x_2,\ x_1^2 + x_2^2)$

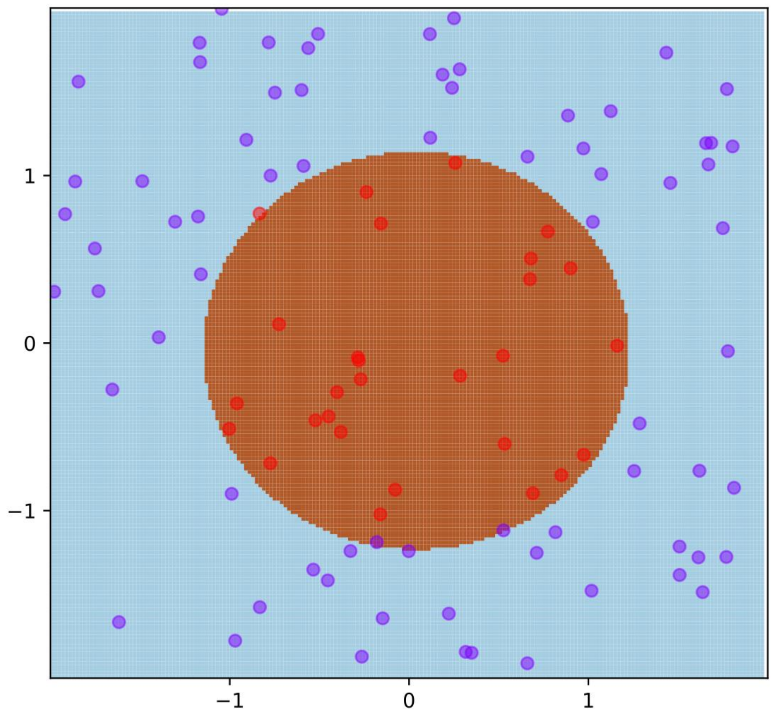


Original feature space

Image: Shiyu Ji (wikipedia)

Kernel Trick example

Feature mapping: $\varphi(\boldsymbol{x}) = (x_1,\ x_2,\ x_1^2 + x_2^2)$



Original feature space

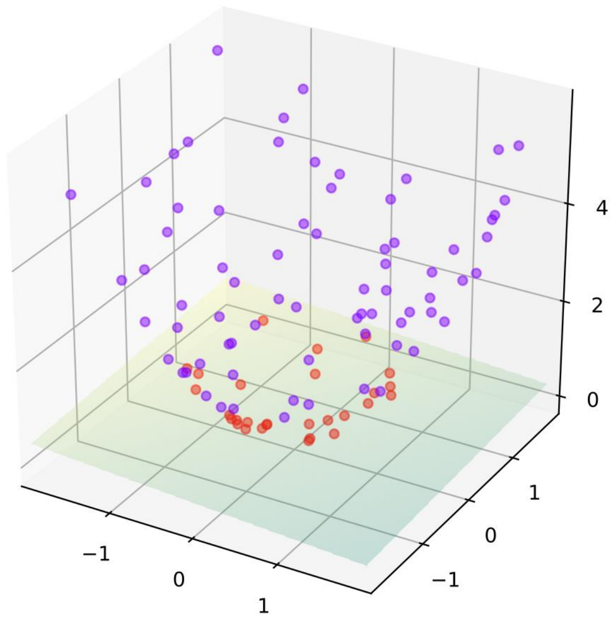
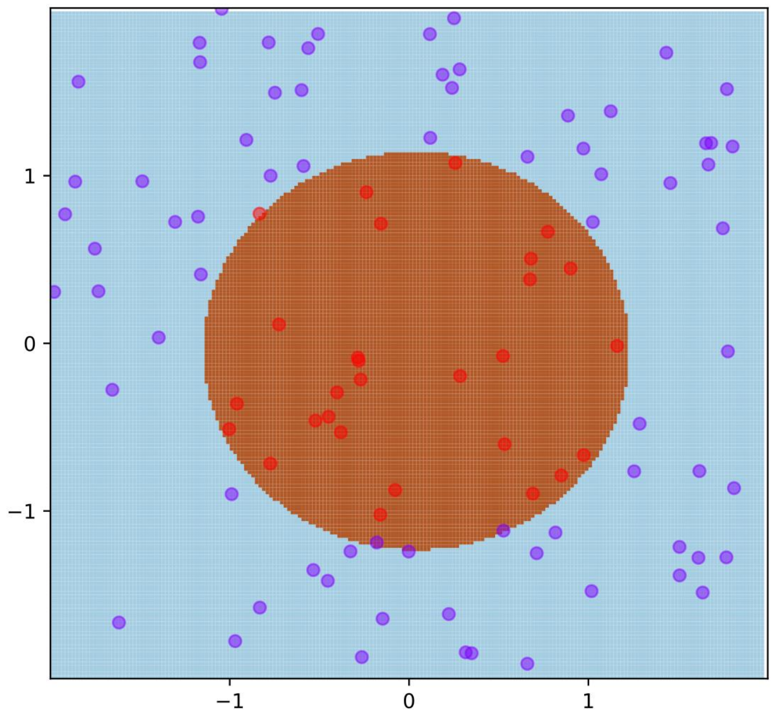


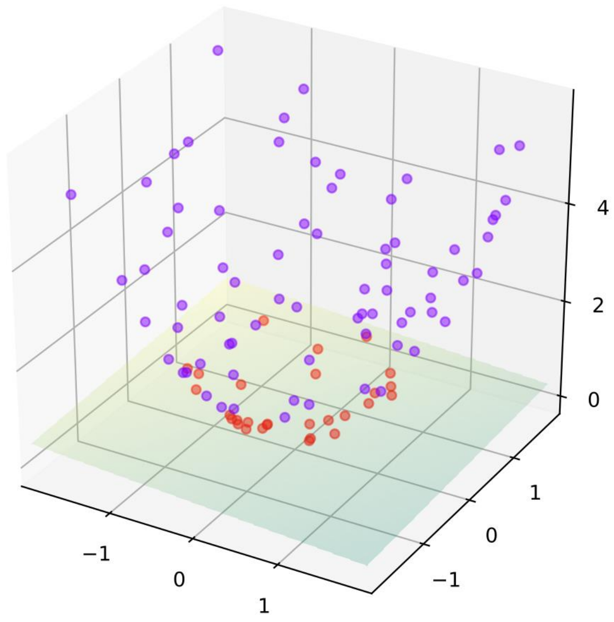
Image: Shiyu Ji (wikipedia)

Kernel Trick example

Feature mapping: $\varphi(\boldsymbol{x}) = (x_1,\ x_2,\ \ x_1^2 + x_2^2)$



Original feature space

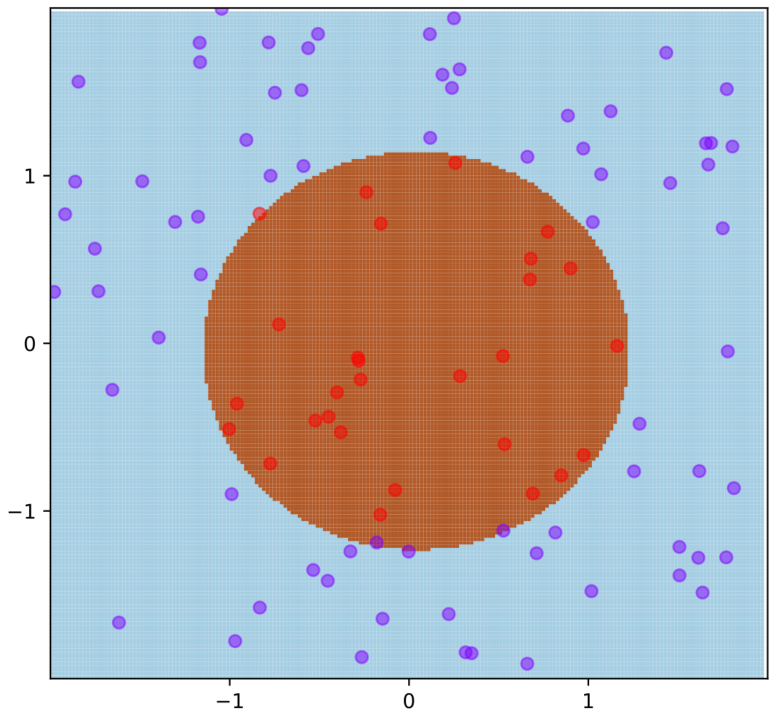


Mapping after applying kernel (can now find a hyperplane)

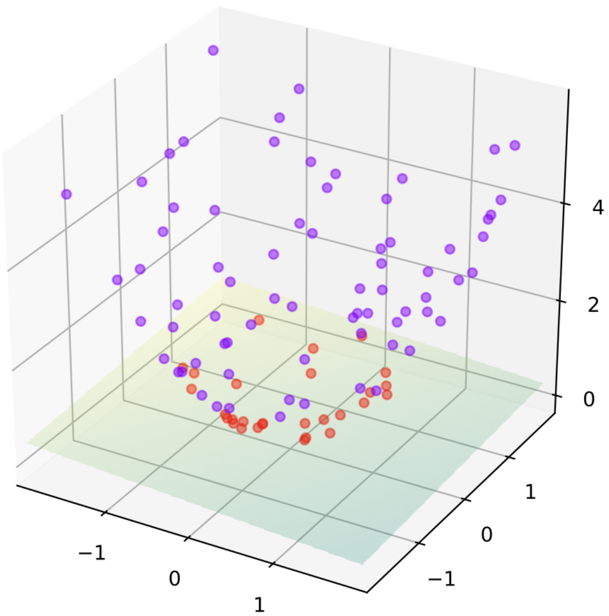
Image: Shiyu Ji (wikipedia)

Kernel Trick example

Feature mapping: $\varphi(\boldsymbol{x}) = (x_1,\ x_2,\ x_1^2 + x_2^2)$



Original feature space



Mapping after applying kernel (can now find a hyperplane)

Kernel function: $K(\boldsymbol{x},\ \boldsymbol{z}) = \boldsymbol{x} \bullet \boldsymbol{z} + ||\boldsymbol{x}||^2\ ||\boldsymbol{z}||^2$

Image: Shiyu Ji (wikipedia)

Gaussian Kernel

- Gaussian kernel is near 0 when points are far apart and near 1 when they are similar
- Also called Radial Basis Function (RBF) kernel

$$K(\vec{x}, \vec{z}) = \exp\left(-\frac{\|\vec{x} - \vec{z}\|^2}{2\sigma^2}\right)$$

Gaussian Kernel

- Gaussian kernel is near 0 when points are far apart and near 1 when they are similar
- Also called Radial Basis Function (RBF) kernel

$$K(\vec{x}, \vec{z}) = \exp\left(-\frac{||\vec{x} - \vec{z}||^2}{2\sigma^2}\right)$$

Often re-parametrized by gamma (different gamma!)

$$\boxed{\gamma = \frac{1}{2\sigma^2}}$$

$$K(\vec{x}, \vec{z}) = \exp\left(-\gamma||\vec{x} - \vec{z}||^2\right)$$

We will tune gamma as part of Homework 7

Soft-margin SVMs (non-separable case)

- Idea: we will use regularization to add a cost for each point being incorrectly classified by the hyperplane

- Hopefully many costs will be 0, but we can accommodate a few outliers

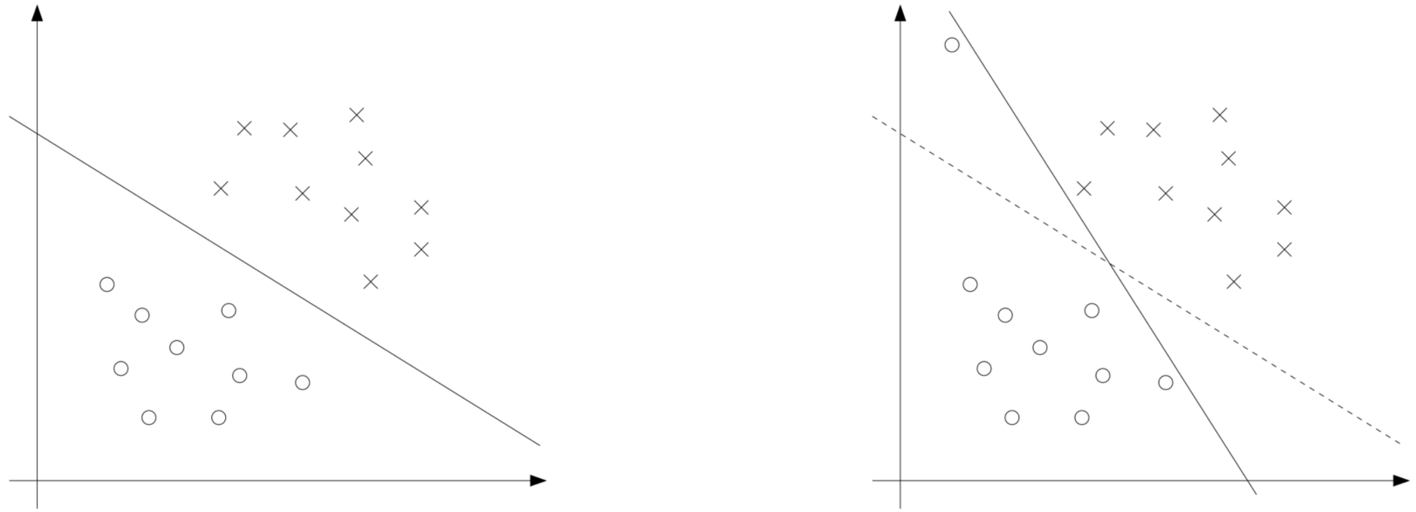


Figure: Andrew Ng

Soft-margin SVMs (non-separable case)

- New optimization problem with regularization
- We will tune the C parameter as part of Homework 7

$$\min_{\xi, \vec{w}, b} \quad \frac{1}{2}\|\vec{w}\|^2 + C \sum_{i=1}^{n} \xi_i$$

"flexible margin"

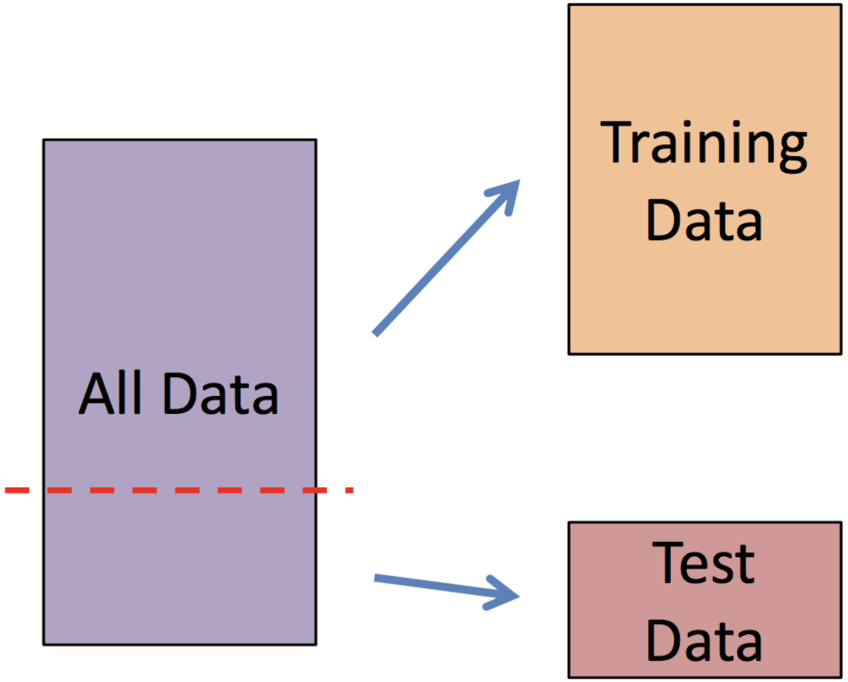
$$\text{s.t.} \quad y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \cdots, n$$

$$\text{and} \quad \xi_i \geq 0, \quad i = 1, \cdots, n$$

Agenda/Outline

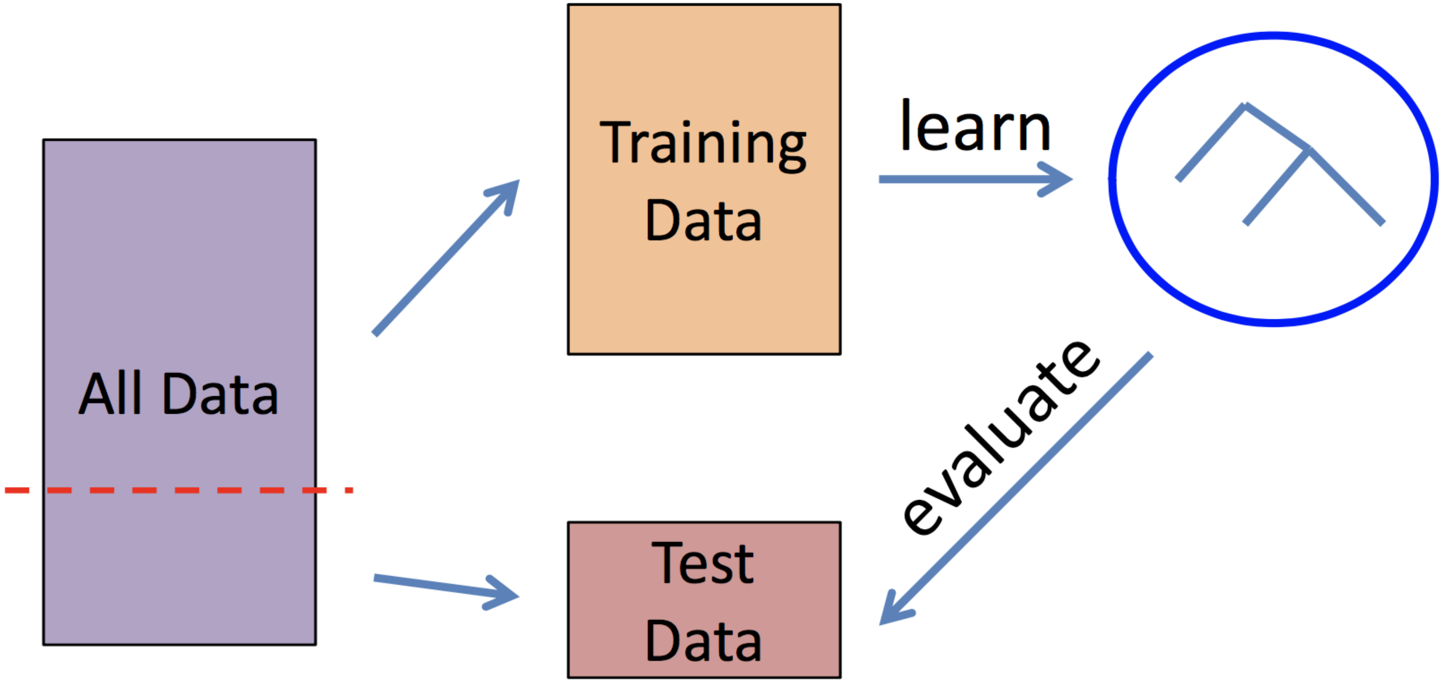
- SVMs
- **Cross-Validation**
- Neural Networks

Evaluation in Practice



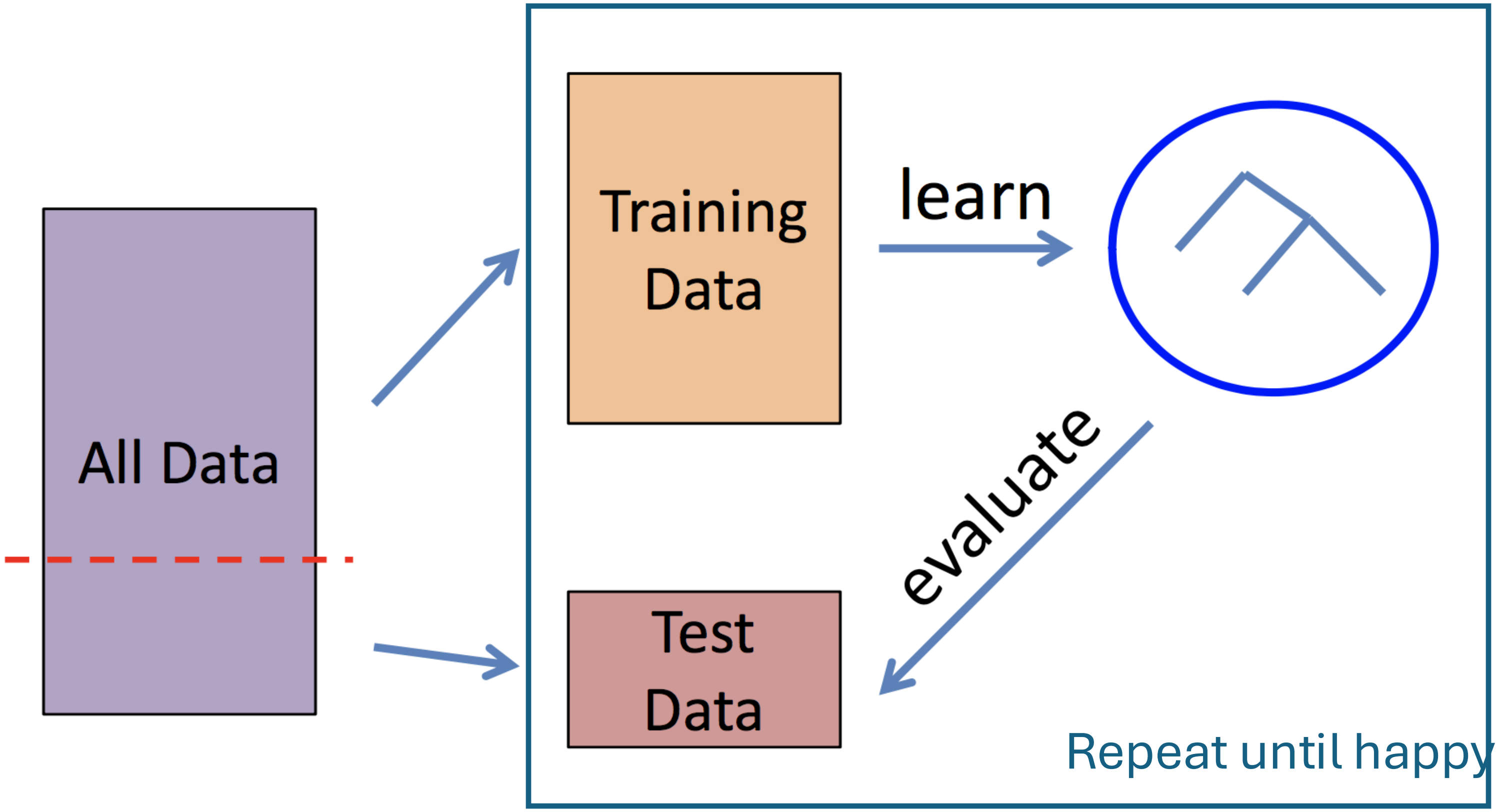
Modified from Jessica Wu

Evaluation in Practice



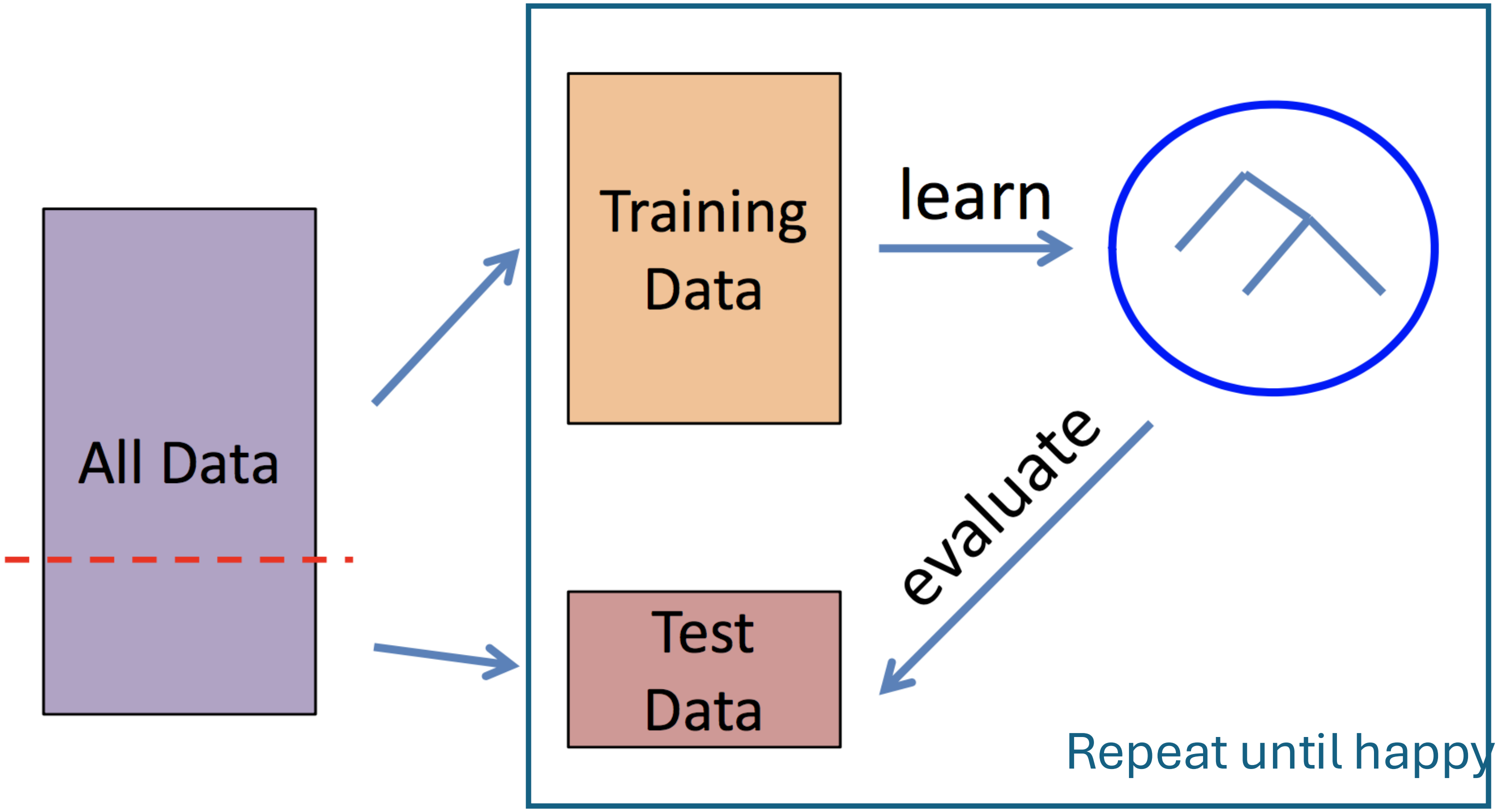
Modified from Jessica Wu

Evaluation in Practice



Modified from Jessica Wu

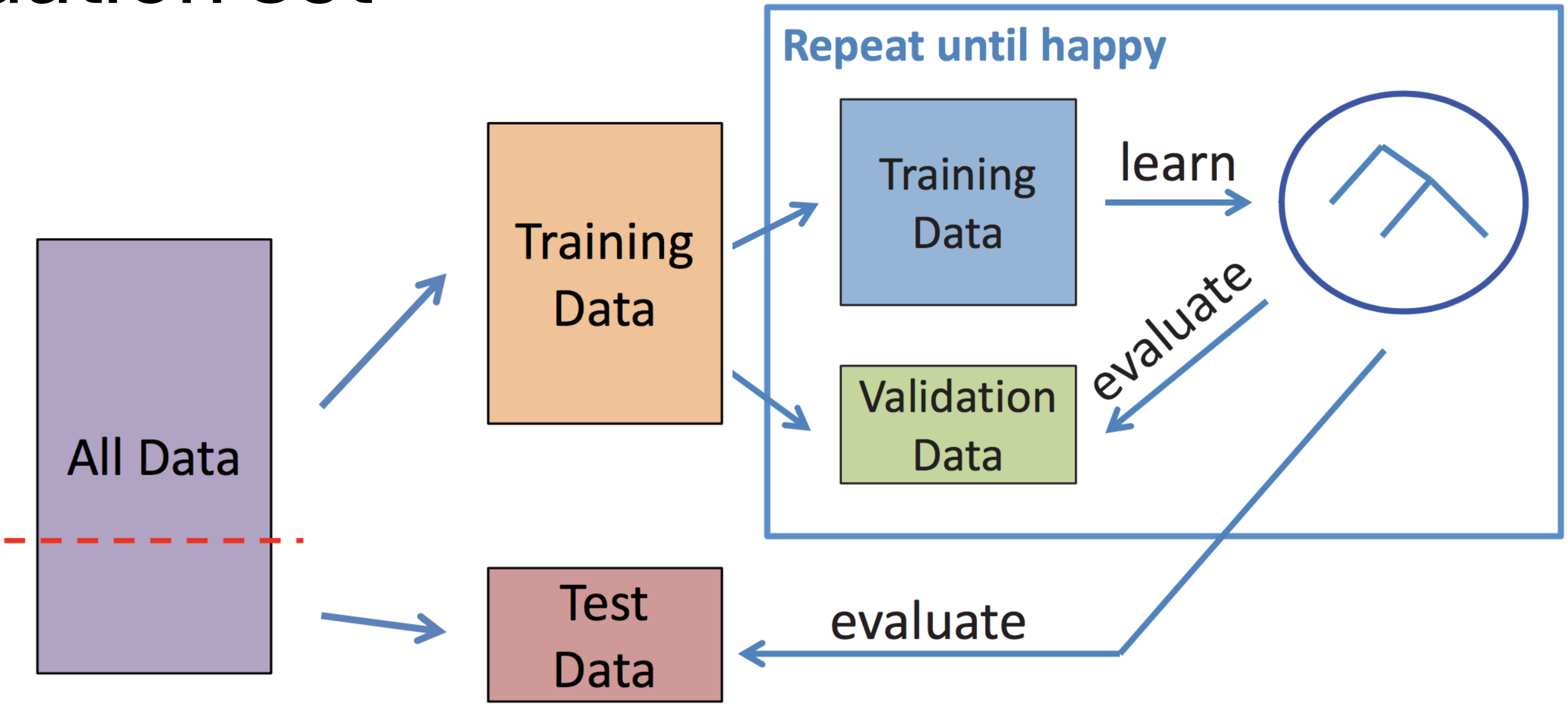
Evaluation in Practice



NO! Using test data as part of the model selection process

Modified from Jessica Wu

Validation set



Modified from Jessica Wu

k-fold Cross Validation

- Why just choose one particular "split" of data?
  - in principle, we should do this multiple times since performance may be different for each split

Slide: Jessica Wu

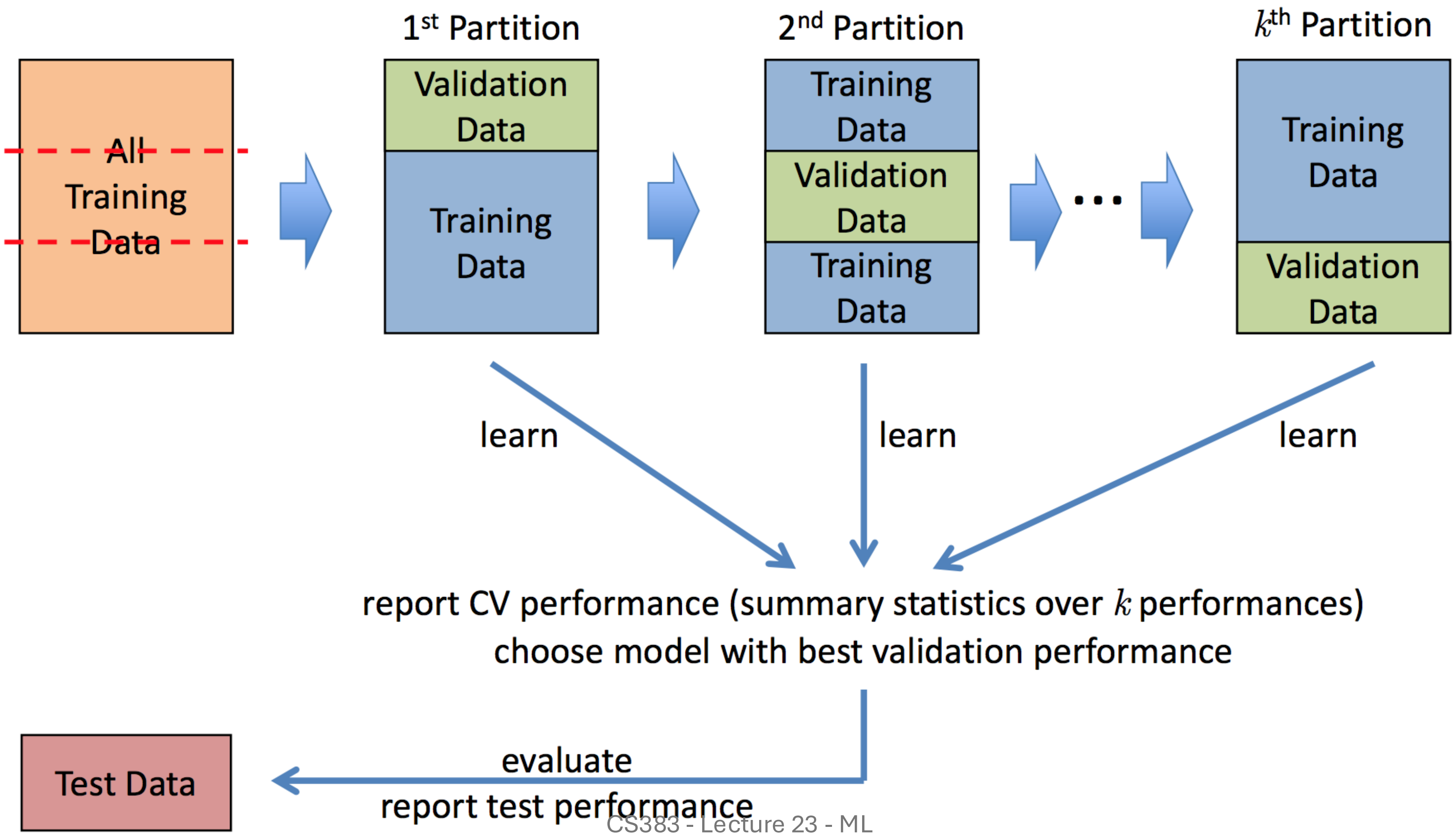
k-fold Cross Validation

- Why just choose one particular "split" of data?
  - in principle, we should do this multiple times since performance may be different for each split

- k-Fold Cross-Validation (e.g., $k = 10$)
  - randomly partition full data set of n instances into $\boldsymbol{k}$ **disjoint subsets** (each roughly of size n/k)
  - choose each fold in turn as validation set; train model on the other $k - 1$ folds and evaluate
  - compute statistics over k test performances, or choose best of k models

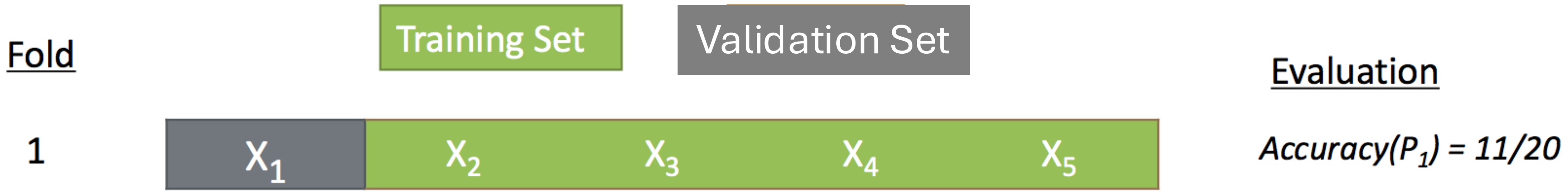
Slide: Jessica Wu

k-fold Cross Validation



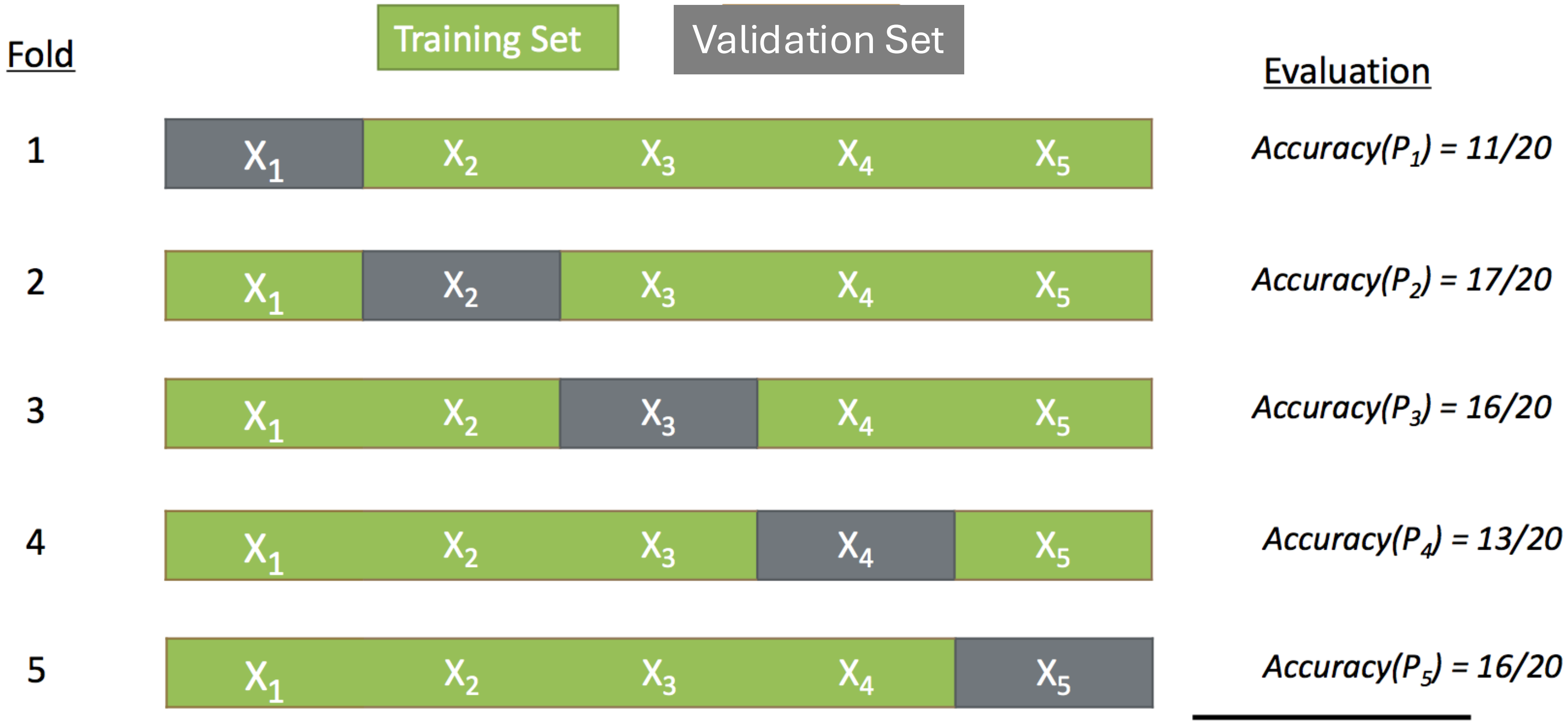
Slide: Jessica Wu

k-fold Cross Validation



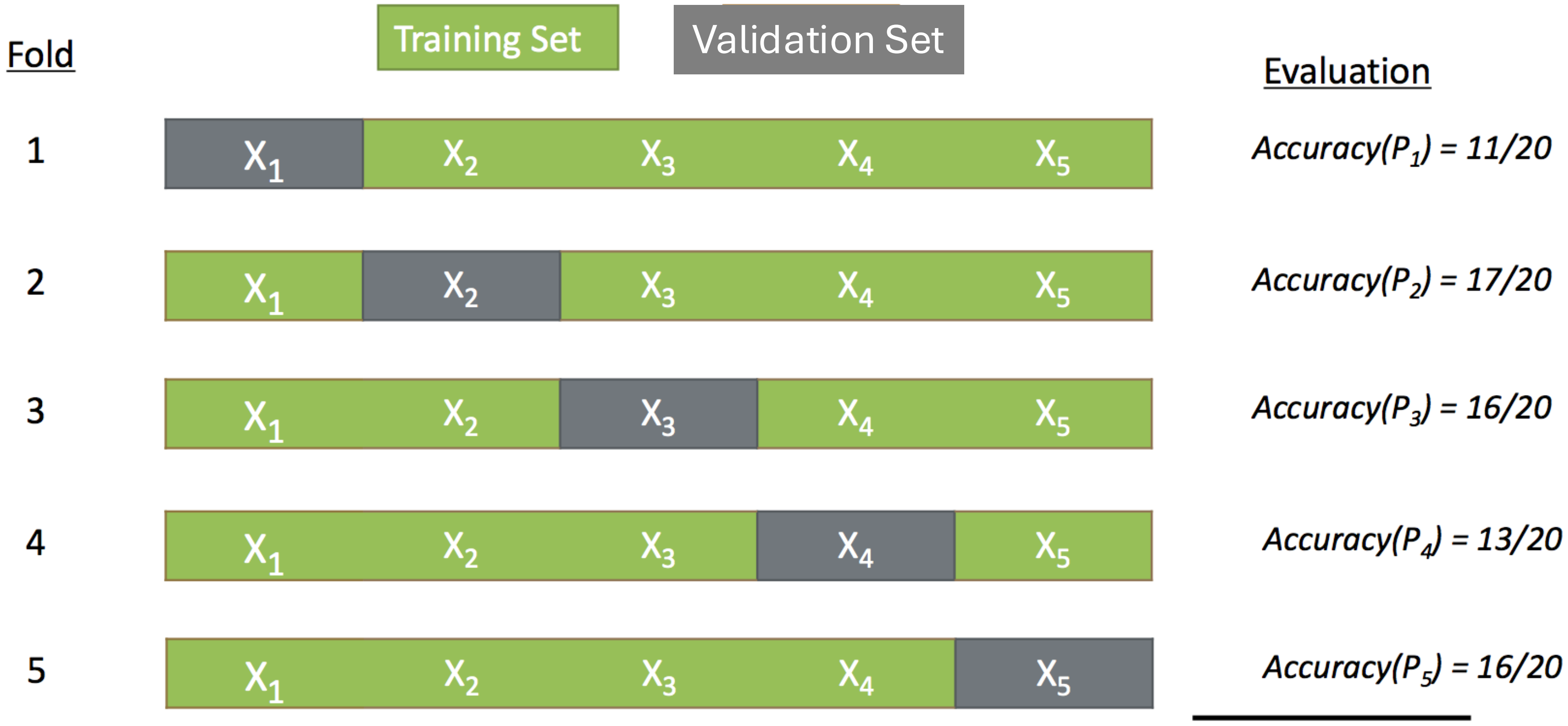
Slide: Ameet Soni

k-fold Cross Validation



Slide: Ameet Soni

k-fold Cross Validation



Generalization: average accuracy across all folds = 73/100 = 73%

Slide: Ameet Soni

Discussion

1) What are the costs of *k*-fold cross validation?

2) Pros and cons of no longer having one model?

3) How to choose *k*?

Discussion

1) What are the costs of k-fold cross validation?
    - Computational, especially if training takes a long time

2) Pros and cons of no longer having one model?
    - Con: might be hard to interpret
    - Pro: might be able to average results

3) How to choose k?
    - Large k can be good for small datasets (i.e. where n is small)
    - Tradeoff between computation and reducing variance
    - Many choose k=10 in practice :)

Cross Validation: other considerations

- Can use cross-validation to choose hyper parameters

- Leave-one-out cross validation (LOOCV)
  - Special case of $k=n$
  - Train using $n-1$ examples, evaluate on remaining
  - Repeat n times

- Can do multiple trials of CV

The Short Way
(that Many People Actually Use)

- Split into only training data + validation data
- Train on training data, evaluate on validation data
- Report cross-validation performance
  - possibly also training performance

Slide: Jessica Wu

The Short Way
(that Many People Actually Use)

- Split into only training data + validation data
- Train on training data, evaluate on validation data
- Report cross-validation performance
  - possibly also training performance

- Why is this used?
  - might not be enough data to create held-out test set
  - you cannot trust that authors did not peek at test data anyway =P

Slide: Jessica Wu

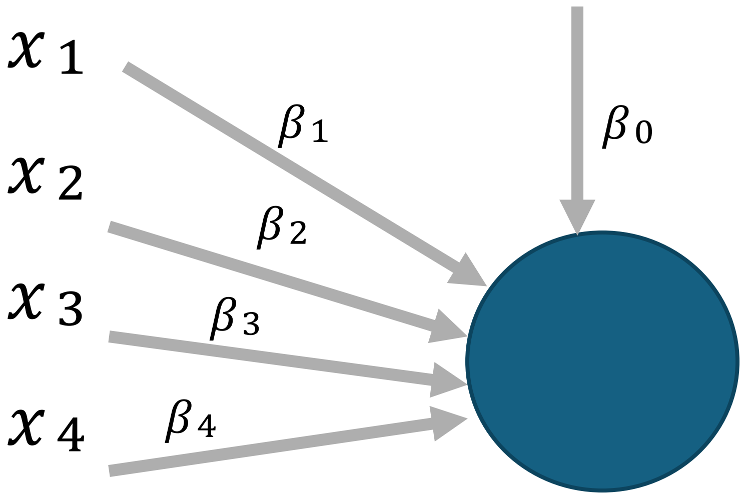
Agenda/Outline

- SVMs
- Cross-Validation
- **Neural Networks**
    - **Perceptron/Logistic Regression as 0-hidden layer NN**
    - Feed forward
    - Non-linear Activation Functions
    - Computation Graph
    - Back-propagation

Logistic Regression

We make a prediction by taking the dot product of the features (covariates) and weights (coefficients)

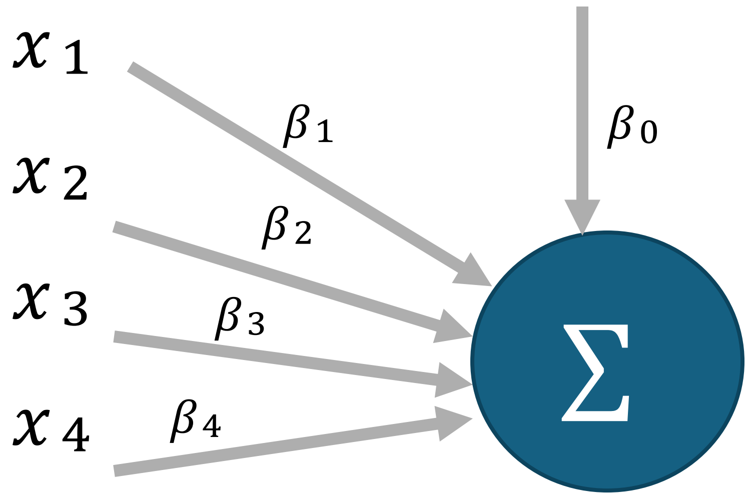
$$\sigma(\boldsymbol{\beta} \cdot \boldsymbol{x}) = \sum_i^j \sigma(\beta_i \cdot x_i)$$



Logistic Regression

We make a prediction by taking the doc product of the features (covariates) and weights (coefficients)

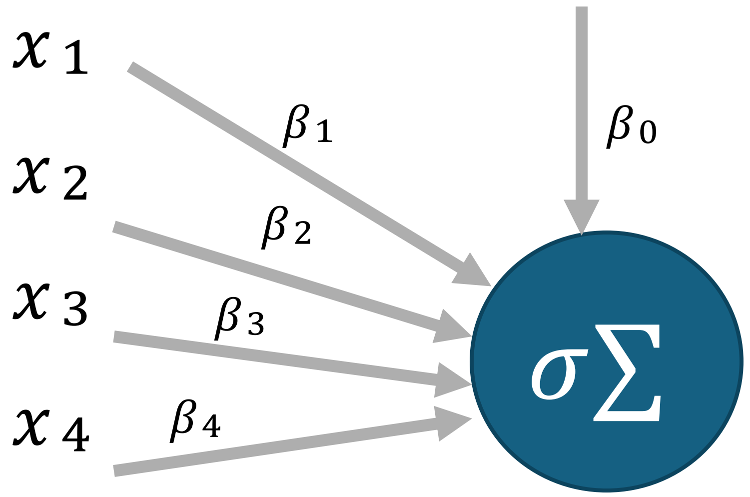
$$\sigma(\boldsymbol{\beta} \cdot \boldsymbol{x}) = \sum_i^j \sigma(\beta_i \cdot x_i)$$



Logistic Regression

We make a prediction by taking the doc product of the features (covariates) and weights (coefficients)

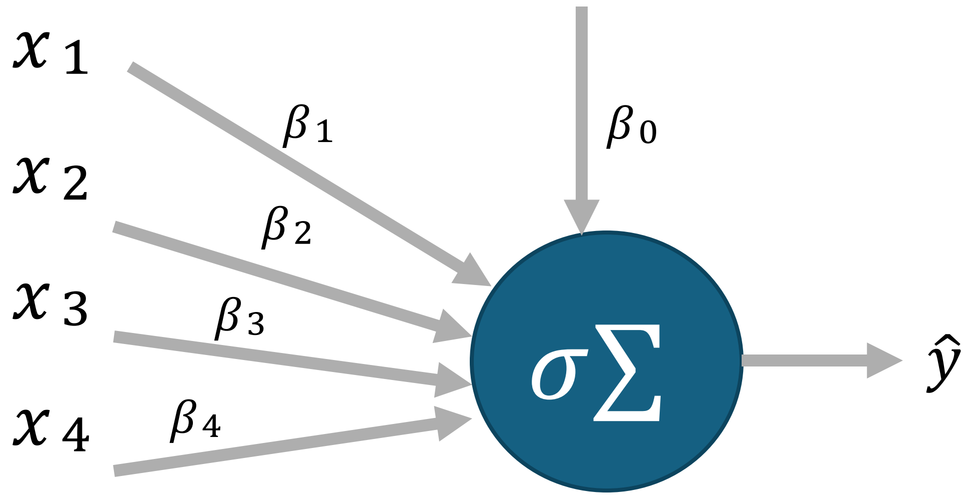
$$\sigma(\boldsymbol{\beta} \cdot \boldsymbol{x}) = \sum_i^j \sigma(\beta_i \cdot x_i)$$



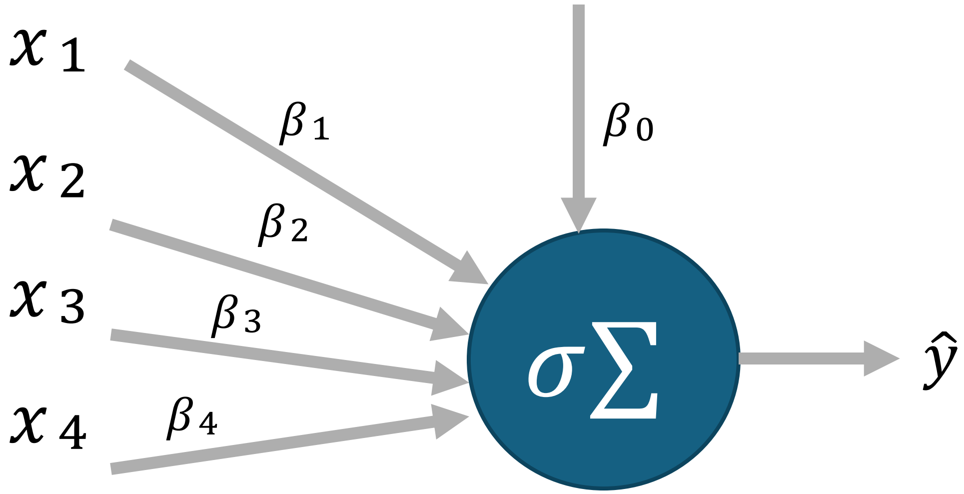
Logistic Regression

We make a prediction by taking the doc product of the features (covariates) and weights (coefficients)

$$\sigma(\boldsymbol{\beta} \cdot \boldsymbol{x}) = \sum_i^j \sigma(\beta_i \cdot x_i)$$



A neuron



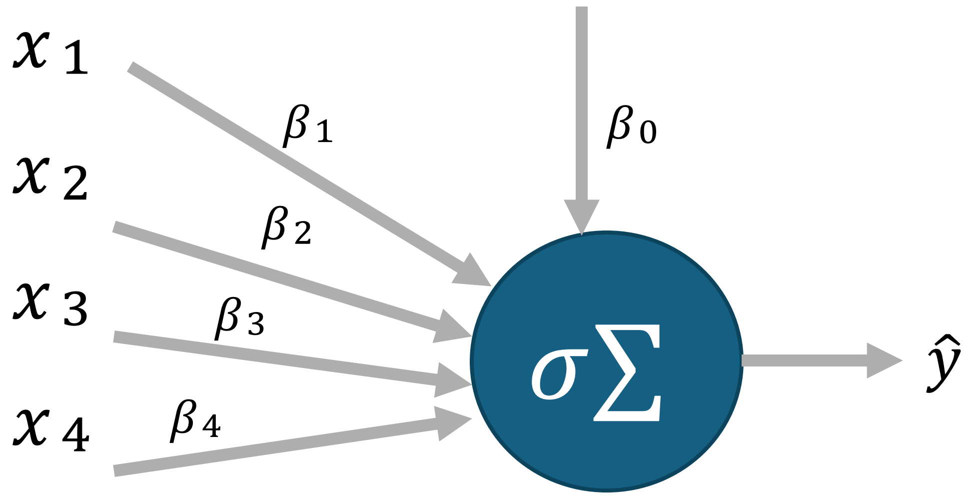
Logistic Regression as NN

A single layer neural network

Input layer: features

Output layer: prediction

We can pass the output of the neuron to another neuron



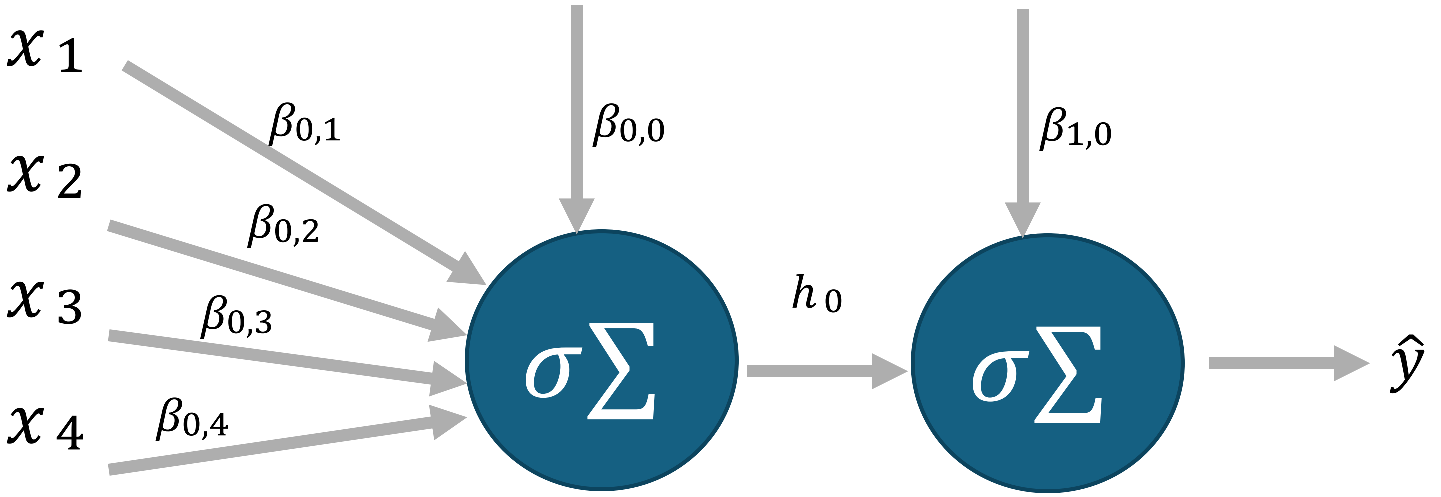
A two layered network

Input layer: features

Output layer: prediction

Hidden layer: h_0

We can add more hidden layers and more neurons at each layer
https://playground.tensorflow.org/



Feed forward NN

x_1

x_2

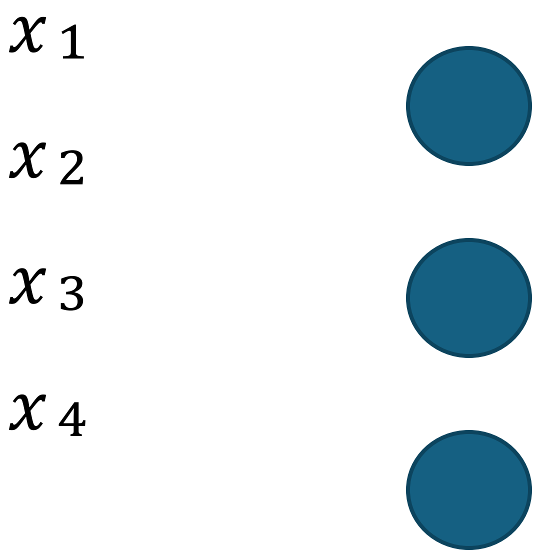
x_3

x_4

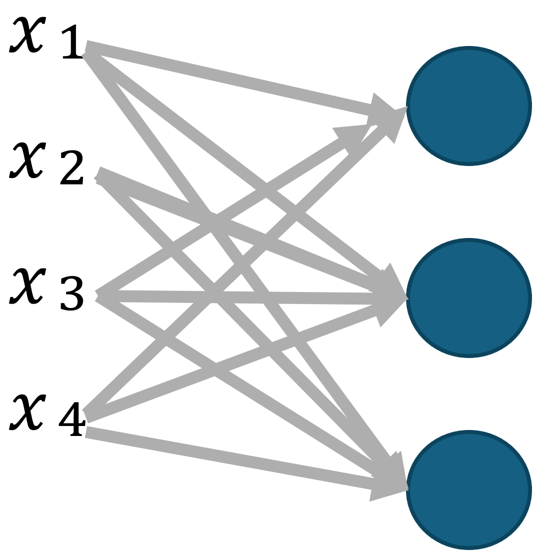
Agenda/Outline

- SVMs
- Cross-Validation
- **Neural Networks**
  - Perceptron/Logistic Regression as 0-hidden layer NN
  - **Feed forward**
  - Non-linear Activation Functions
  - Computation Graph
  - Back-propagation

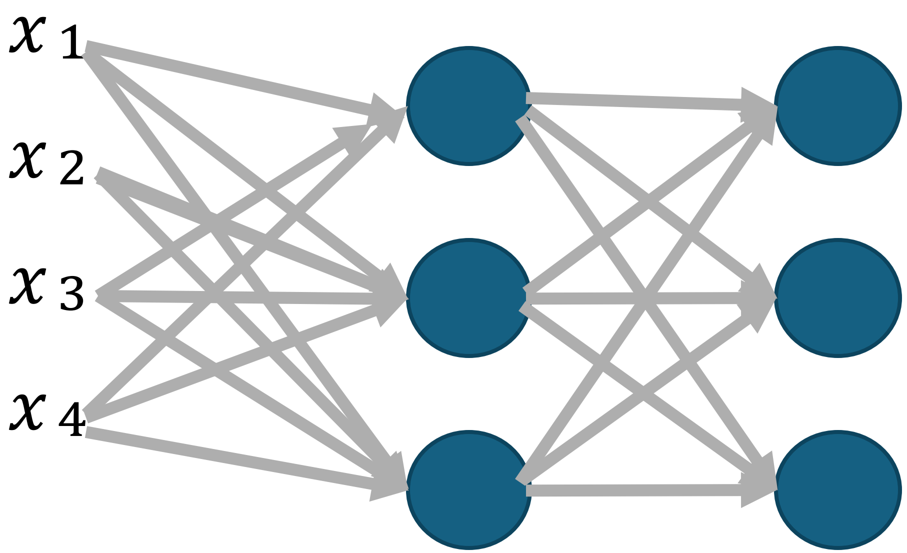
Feed forward NN



Feed forward NN



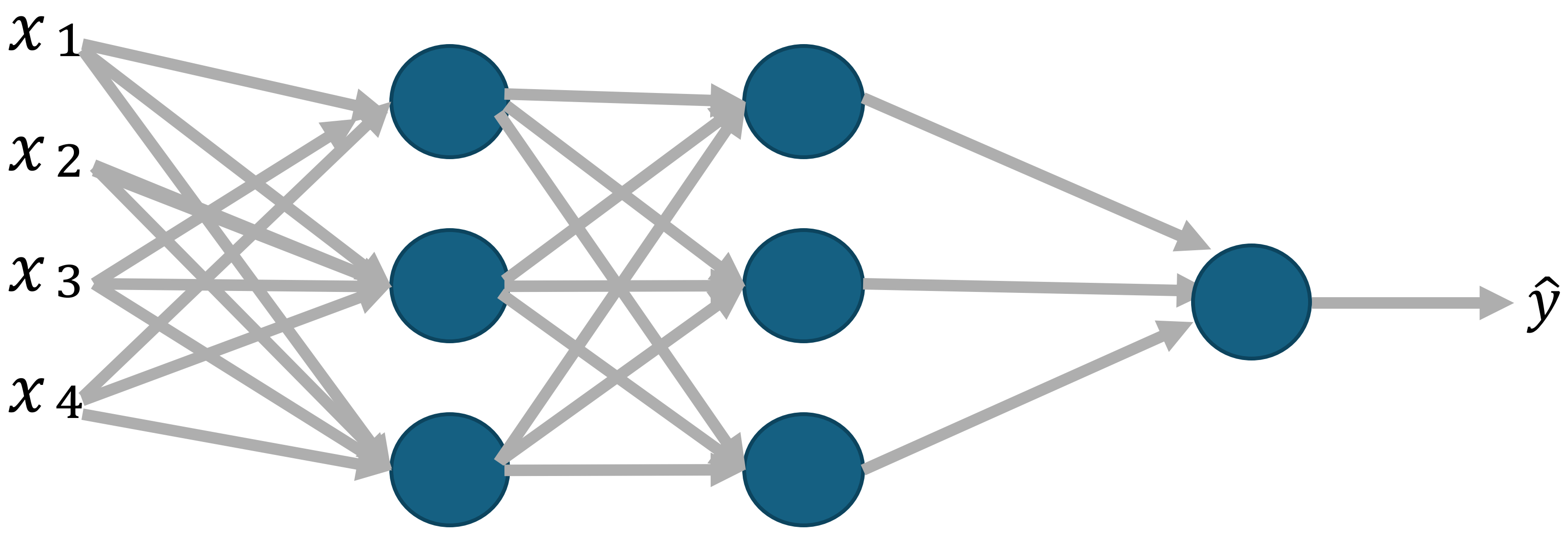
Feed forward NN



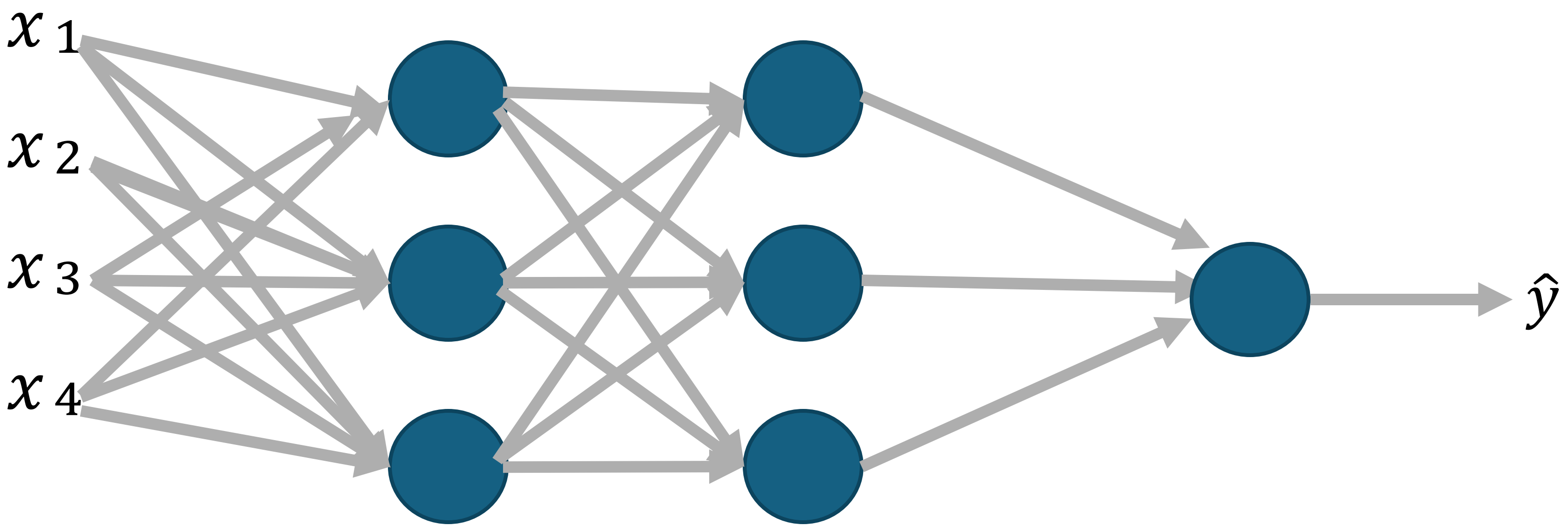
Feed forward NN

All nodes in between each layer are connected

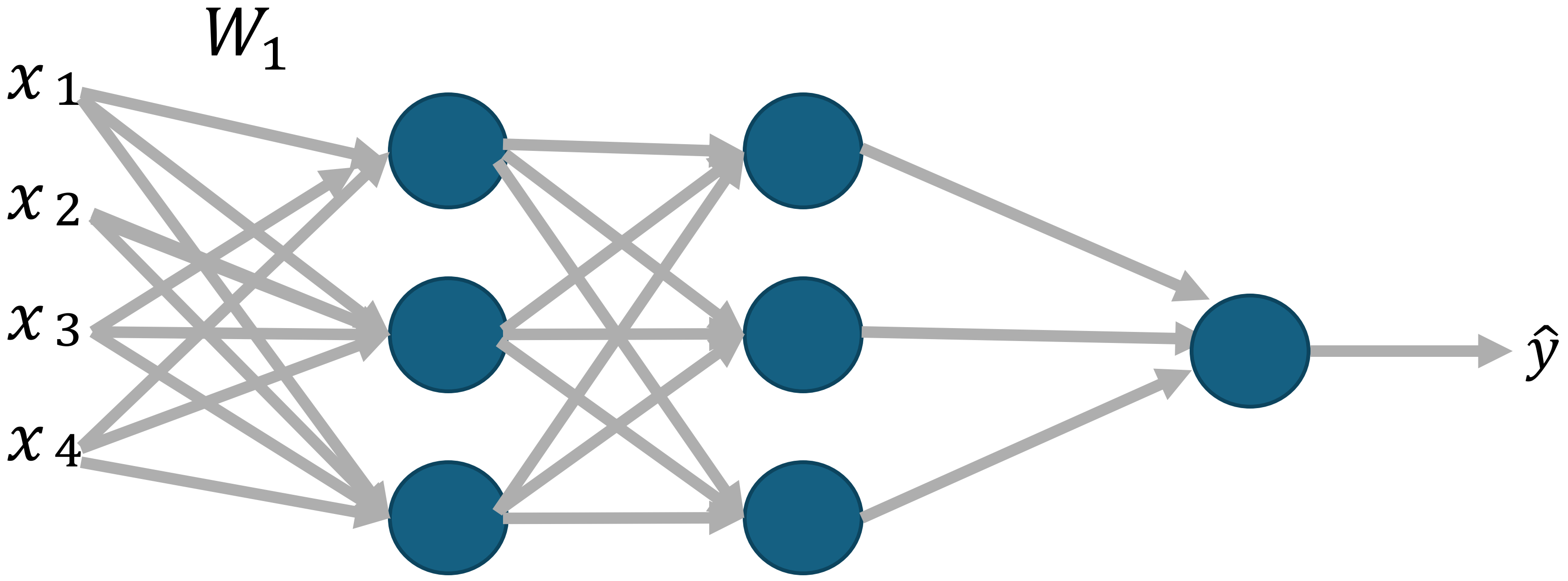
Input layer: features, Output layer: prediction, 2 Hidden layers



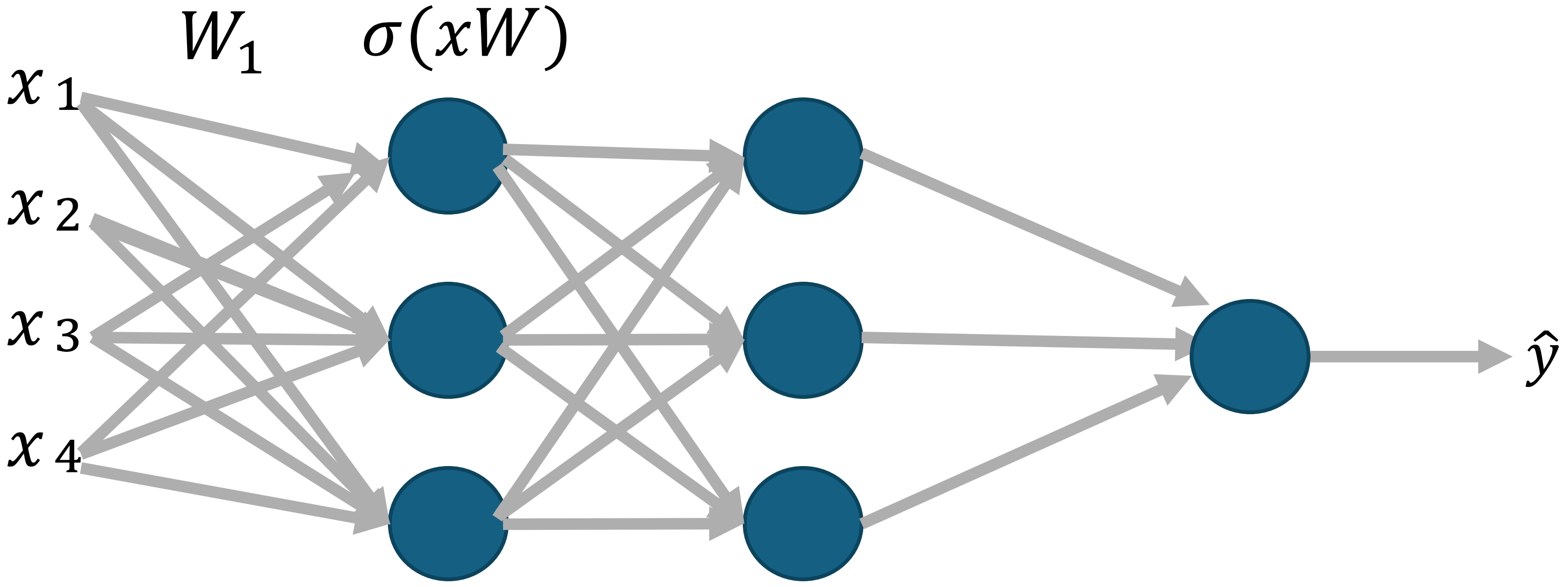
Feed forward NN
Making predictions



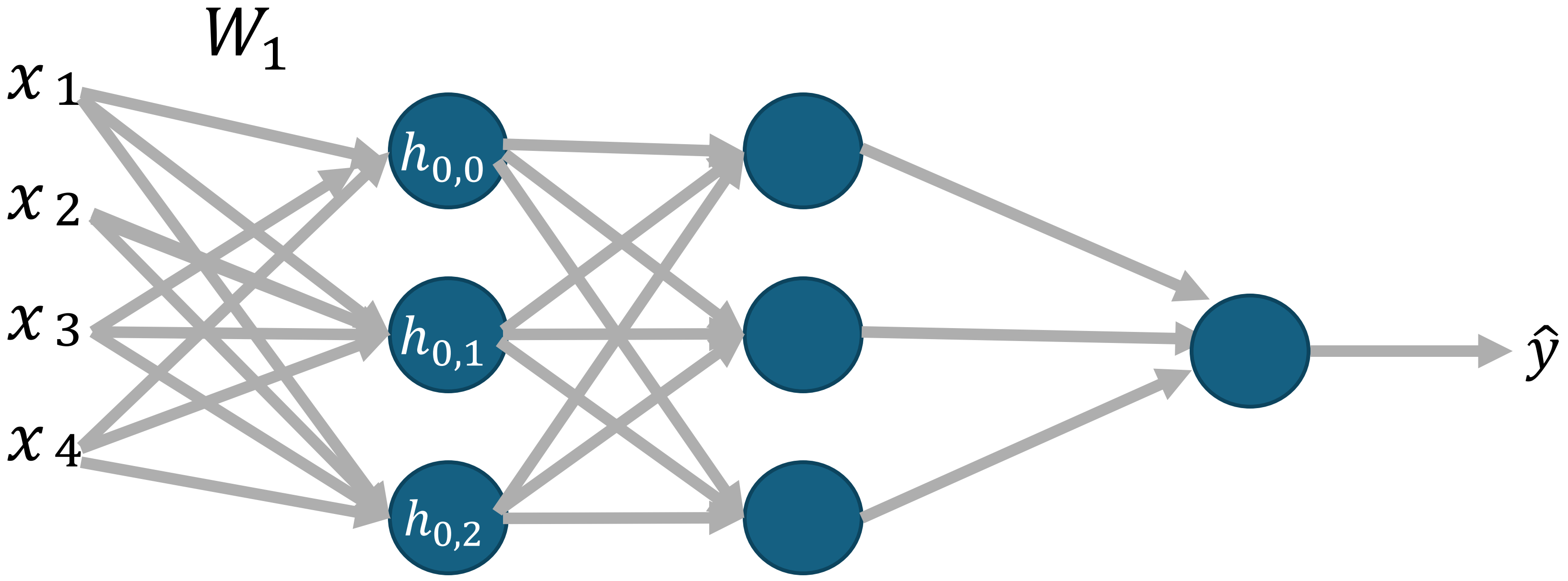
Feed forward NN
Making predictions



Feed forward NN
Making predictions

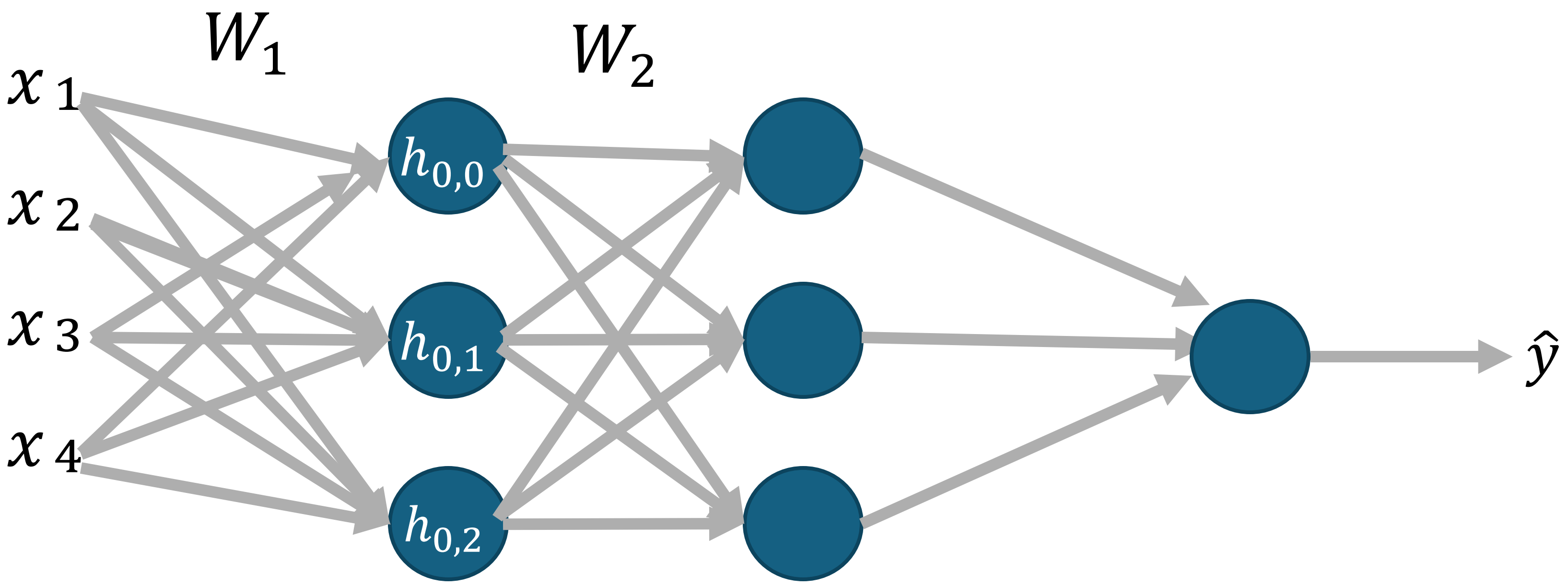


Feed forward NN
Making predictions



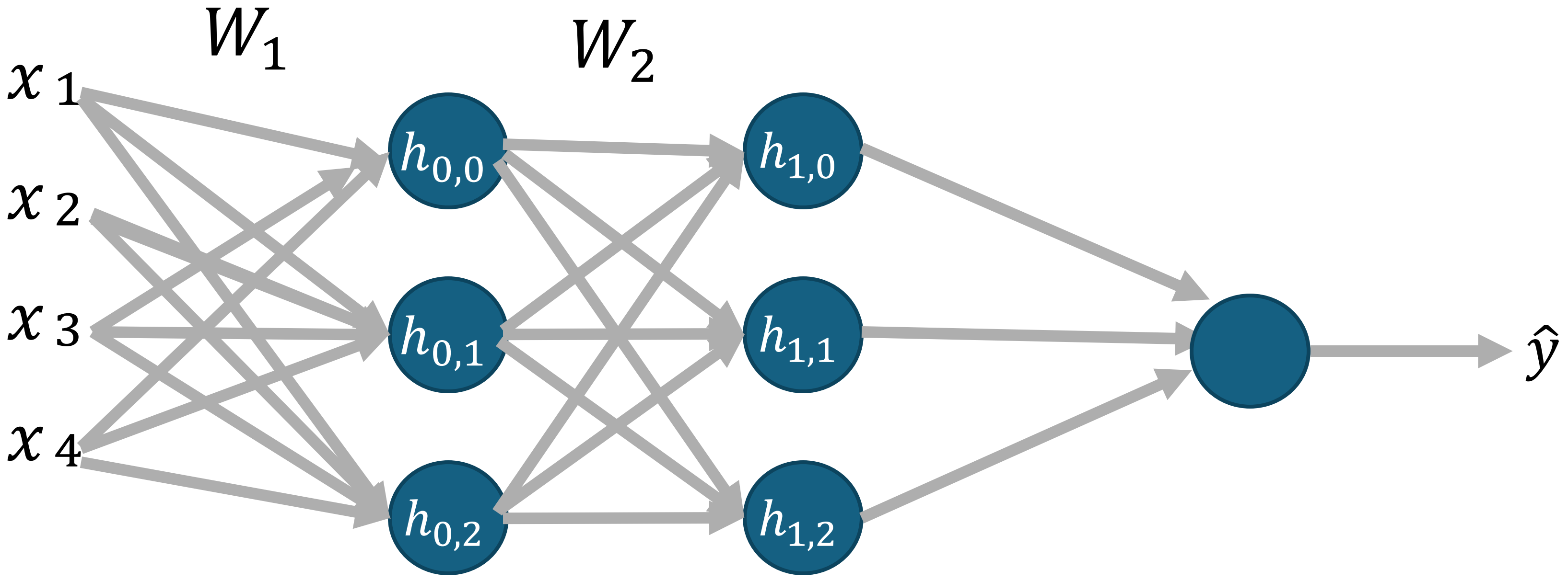
$$\boldsymbol{h}_0 = \sigma(xW_1)$$

Feed forward NN
Making predictions



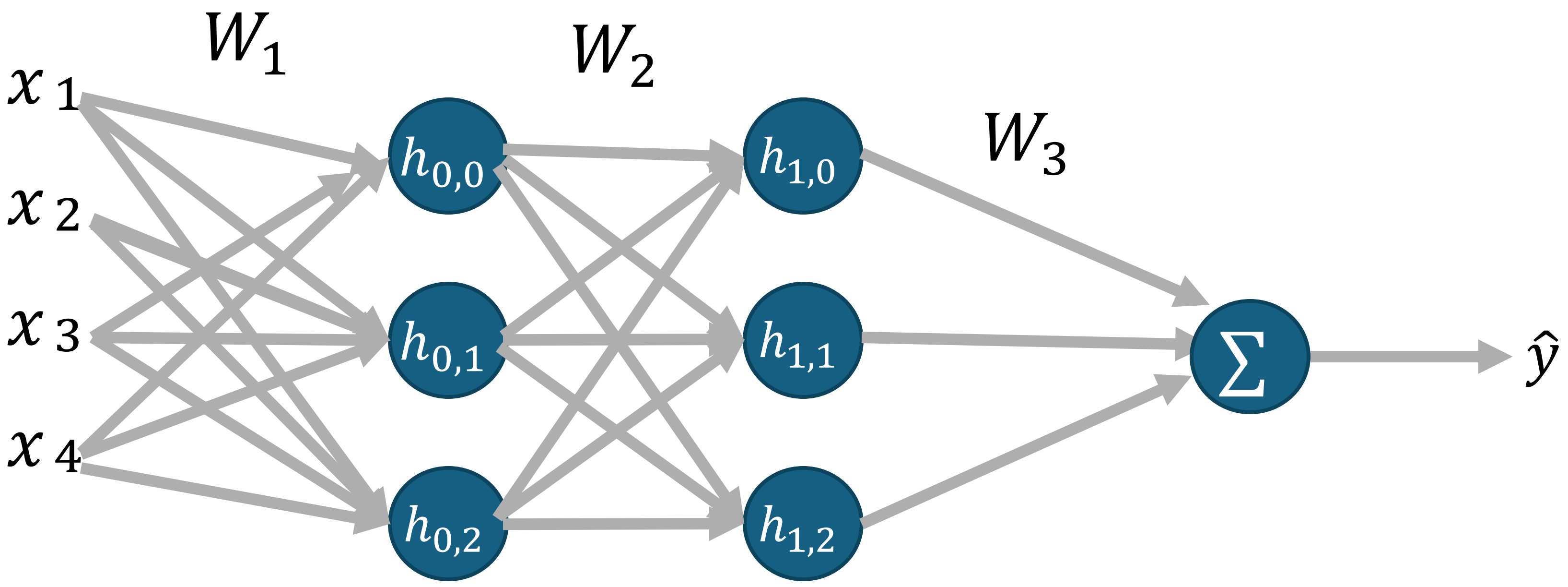
$$\boldsymbol{h}_0 = \sigma(xW_1)$$

Feed forward NN
Making predictions



$$\boldsymbol{h}_0 = \sigma(xW_1)$$

Feed forward NN
Making predictions



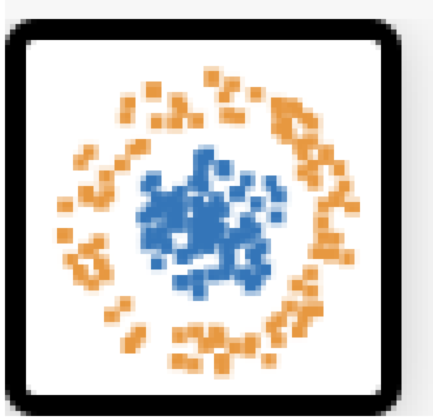
$$\boldsymbol{h}_0 = \sigma(xW_1)$$

Agenda/Outline

- SVMs
- Cross-Validation
- **Neural Networks**
  - Perceptron/Logistic Regression as 0-hidden layer NN
  - Feed forward
  - **Non-linear Activation Functions**
  - Computation Graph
  - Back-propagation

- https://playground.tensorflow.org/

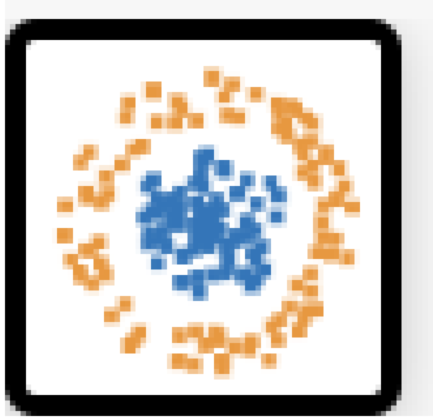
Playground: Example 1

- Data distribution: 
- No hidden layers
- Activation function: Linear

- Goal: create a network to fit this data

Playground: Example 2

- Data distribution:



- No hidden layers
- Activation function: Sigmoid

- Goal: create a network to fit this data