

# MSBA 434: Advanced Workshop on Machine Learning

## Lecture 5: Ensemble Methods

# Agenda

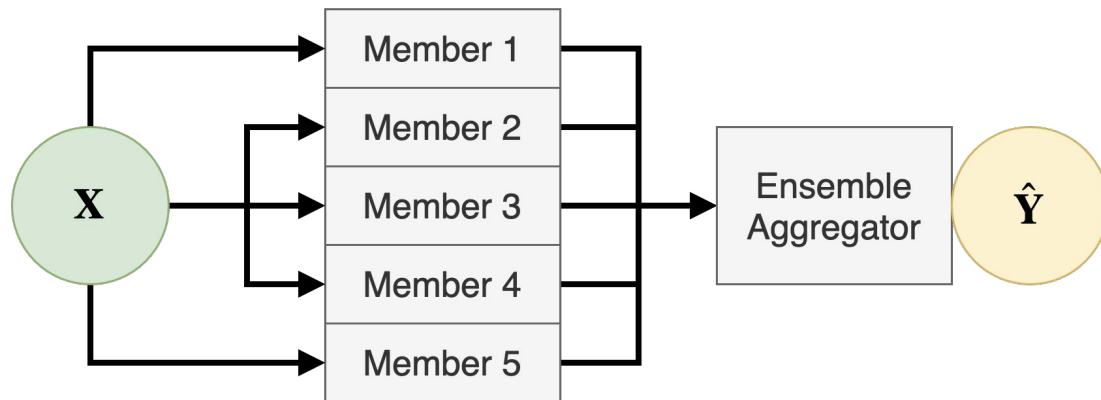
1. Why ensembles work?
2. Ensemble techniques
3. Case Study: Gaussian Circles
4. Final Project
5. Course Summary

# Ensemble Inference

Ensemble is a collection of models whose output is aggregated together.

Under certain conditions, **aggregated output is better than any member output.**

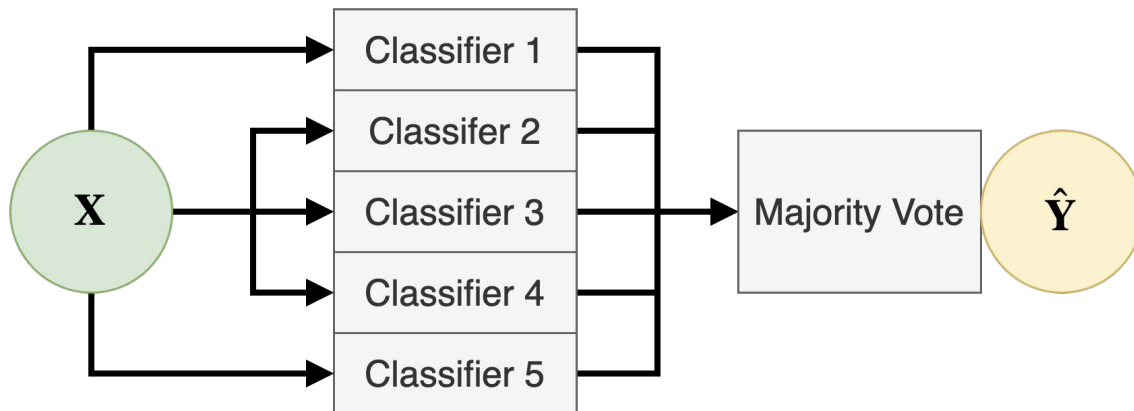
Members may or may not communicate.



# Binary Classification

Ensemble output is the class predicted by the majority of member classifiers.

In case of probabilities, the average probability will be consistent with discrete majority vote.

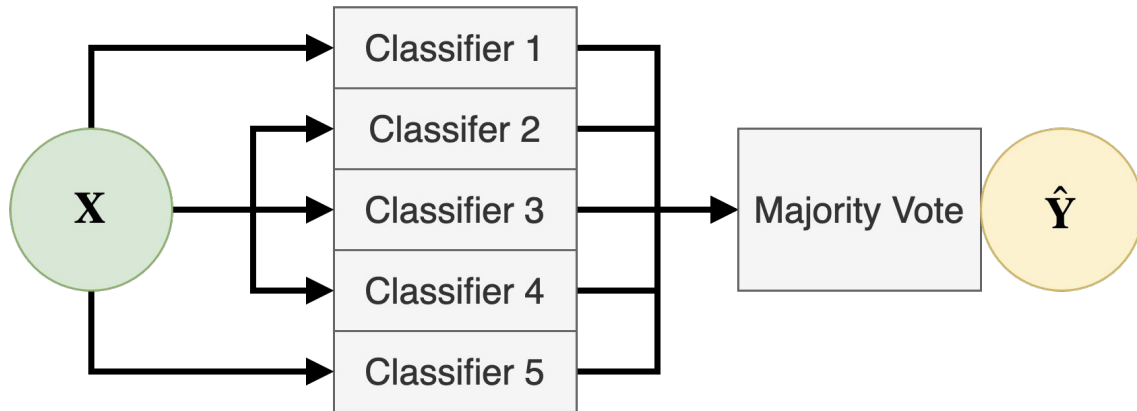


# Ensemble Properties

Ensemble performance depends on:

1. Member strength
2. Ensemble size
3. Member independence

Ensemble performance may or may not be better than its best member.



# Ensemble Performance

How to evaluate ensemble performance as a function of its properties?

## Method 1: Analytical

Mathematical analysis of the statistical distribution and its properties.

## Method 2: Monte Carlo

Approximate the distribution properties using a finite batch of random samples.

$$x \sim p_{data}$$

$$a_i(x) \in \mathbb{R}^n \rightarrow \{0, 1\}$$

$$a_i(x) \sim \text{Bernoulli}(p)$$

$$\mathbb{E}[a_i(x)] = p$$

$$e(x) = \left\lceil \sum_i^s \frac{a_i(x)}{s} \right\rceil$$

$$e(x) \sim \text{Binomial}(p, s) > \frac{s}{2}$$

See [MonteCarlo.ipynb](#) for NumPy implementation.

# Member Strength

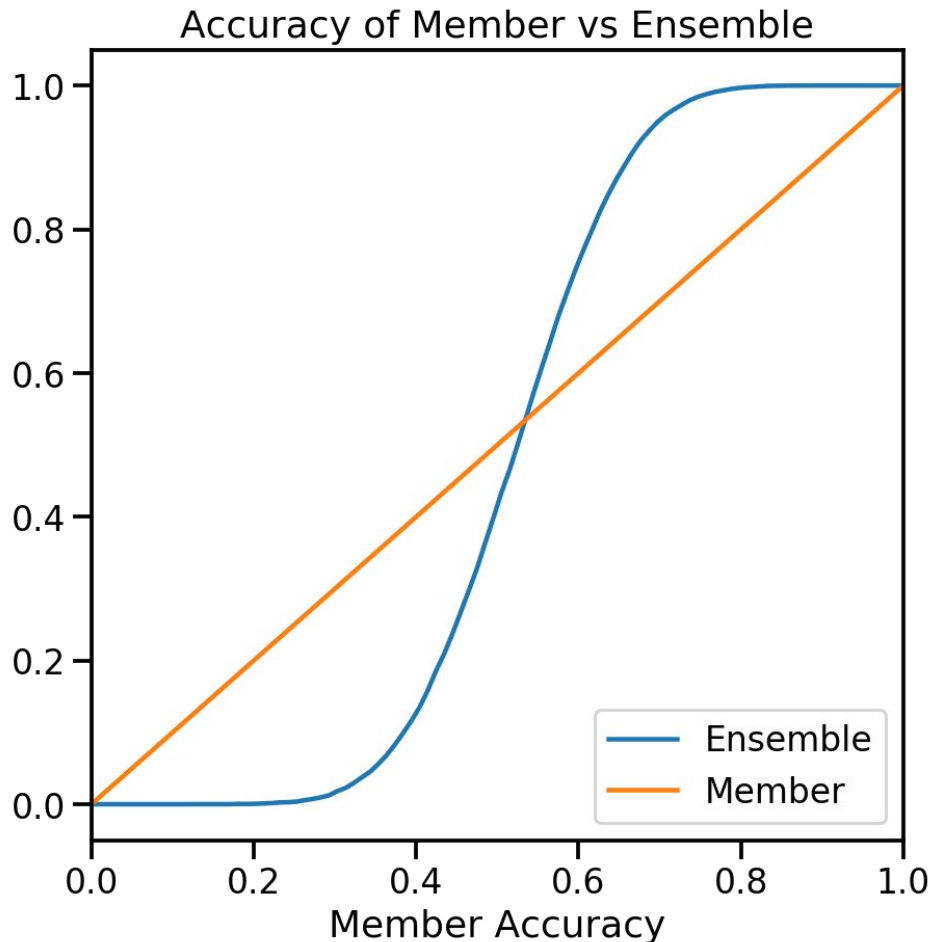
## Simulation

Ensemble of 20 independent binary classifiers.

## Inference

Ensemble strength increases with member strength.

Ensemble members must be better than random.



# Ensemble Size

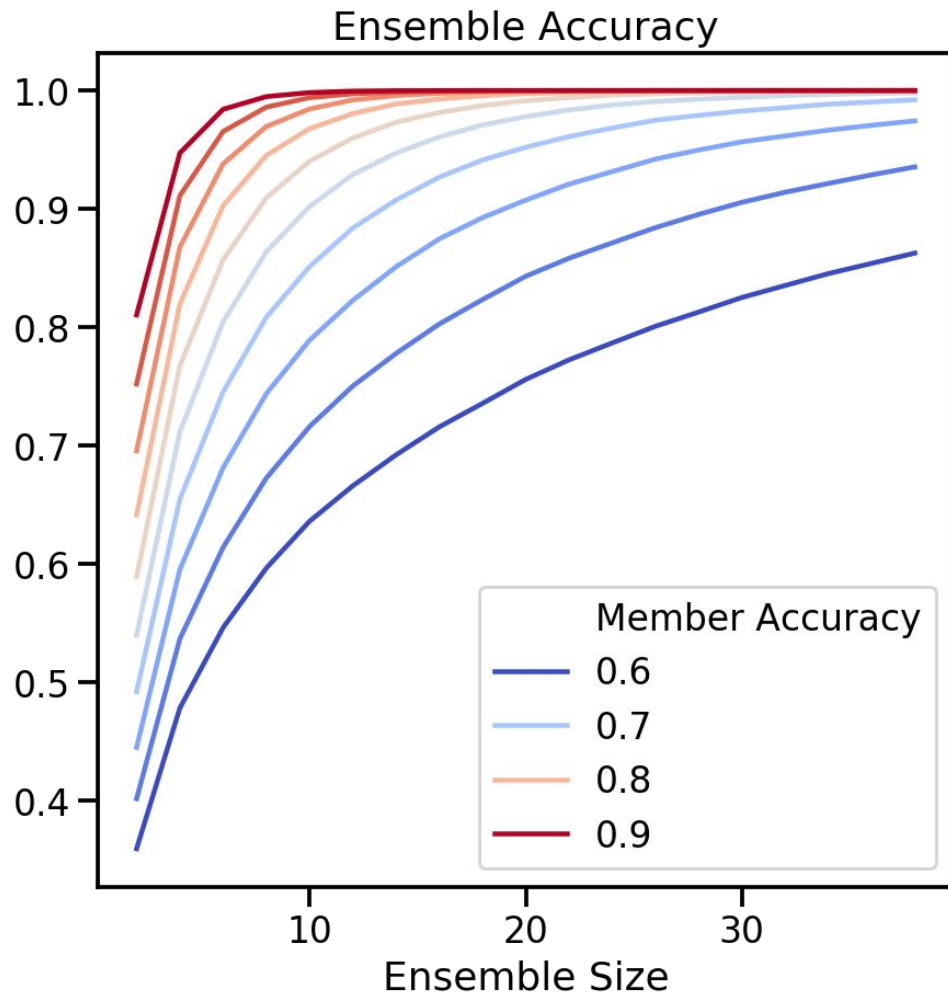
## Simulation

Ensemble of independent binary classifiers.

## Inference

Ensemble strength increases with size.

Marginal value of an additional member decreases with its accuracy and ensemble size.





# Member Independence

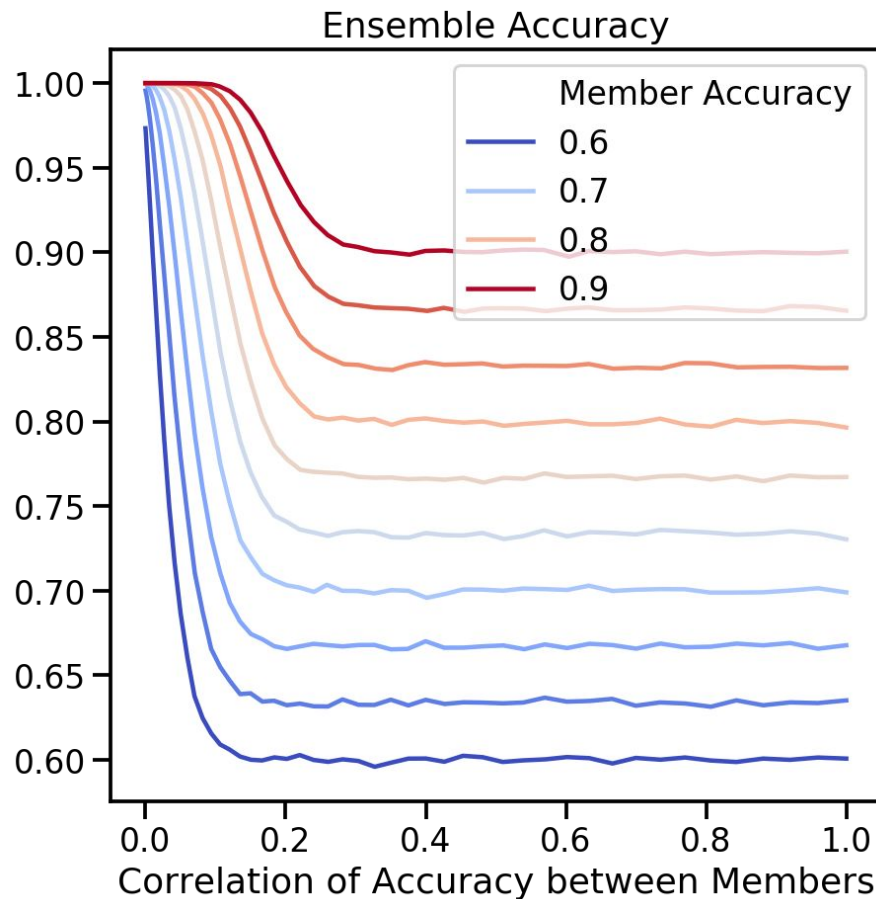
## Simulation

Ensemble of 100 binary classifiers with correlated errors.

## Inference

Ensemble strength decreases with correlation between members.

Sensitivity to correlation decrease with member strength.

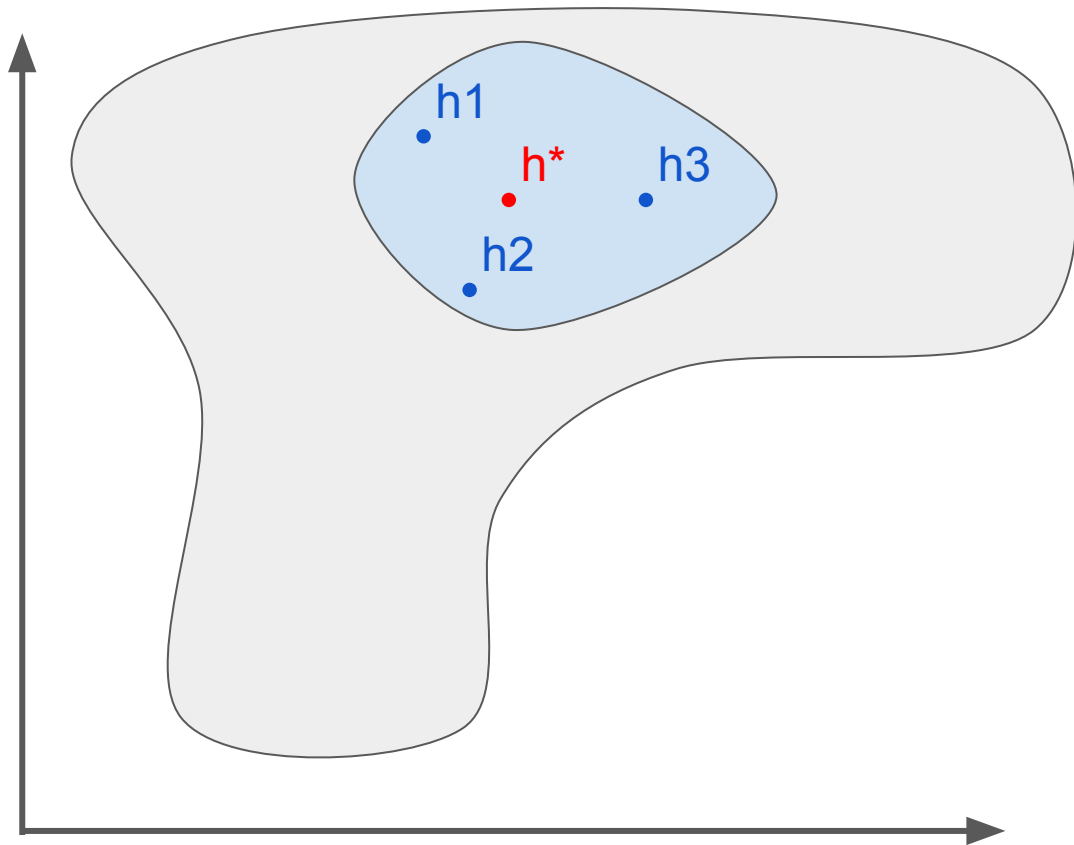


See [MonteCarlo.ipynb](#) for NumPy implementation.

# Statistical

When datasets are small relative to the hypothesis space, **multiple hypotheses may minimize the estimate of the cost.**

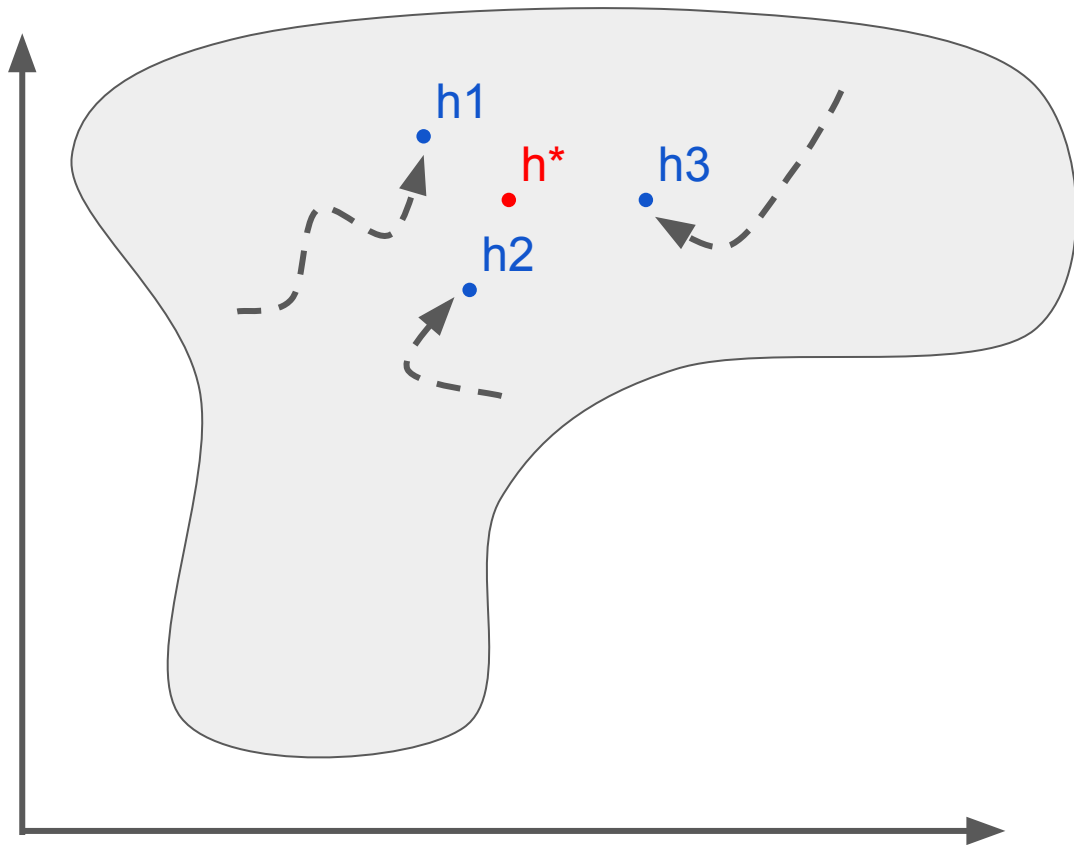
However, under all possible datasets, there should be a single optimal hypothesis.



# Computational

Iterative learning algorithms (e.g. gradient descent) may converge to different local minima.

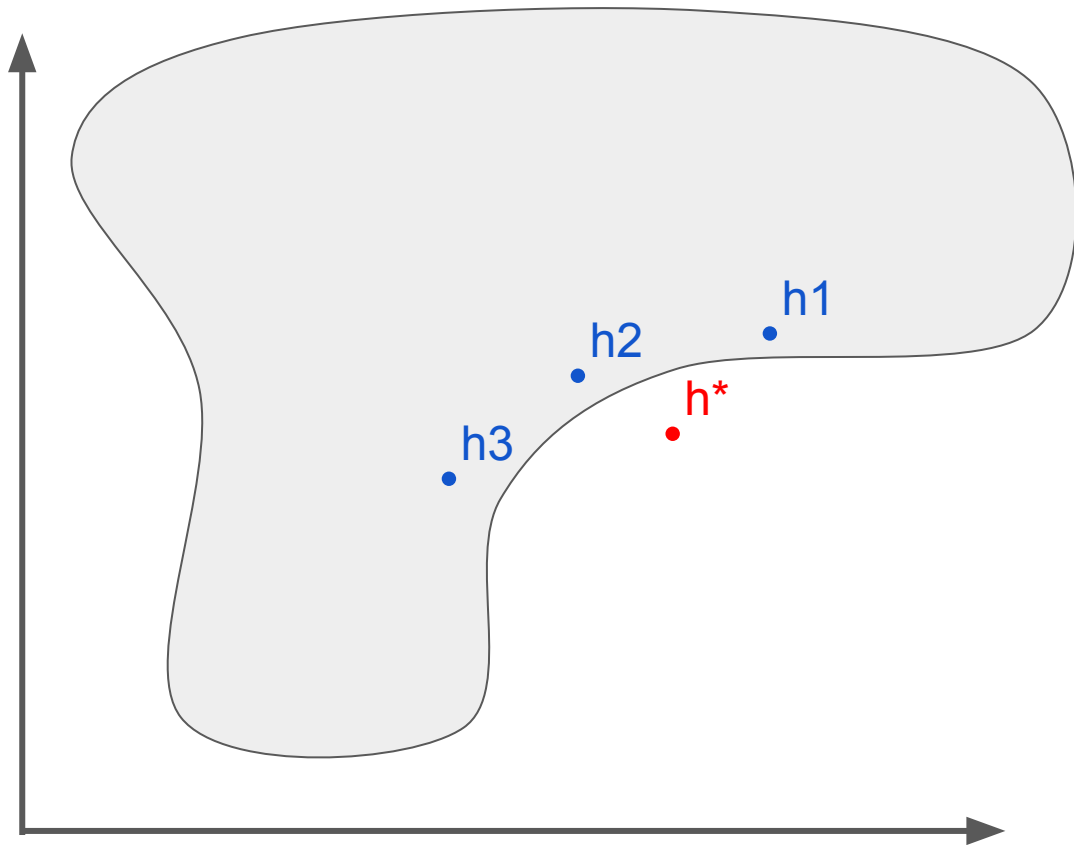
An average of multiple hypotheses may be close to the global minimum.



# Representational

The optimal hypothesis may be outside of the effective hypothesis space that is searched over using a finite dataset.

An average of multiple hypotheses may be outside of the effective hypothesis space and closer to the optimal hypothesis.

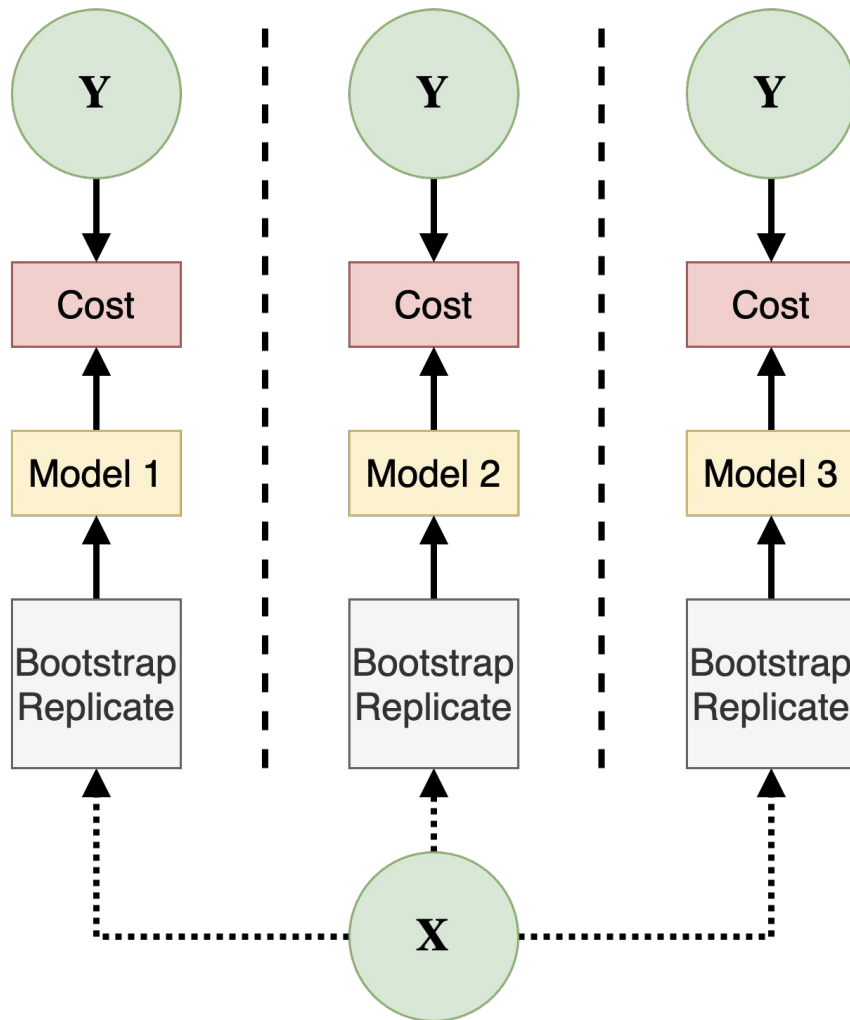


# Bagging

**Bootstrap replicate** =  
random resampling of the  
dataset with replacement.

**Bootstrap aggregation** =  
an ensemble of models  
trained on different  
bootstrap replicates.

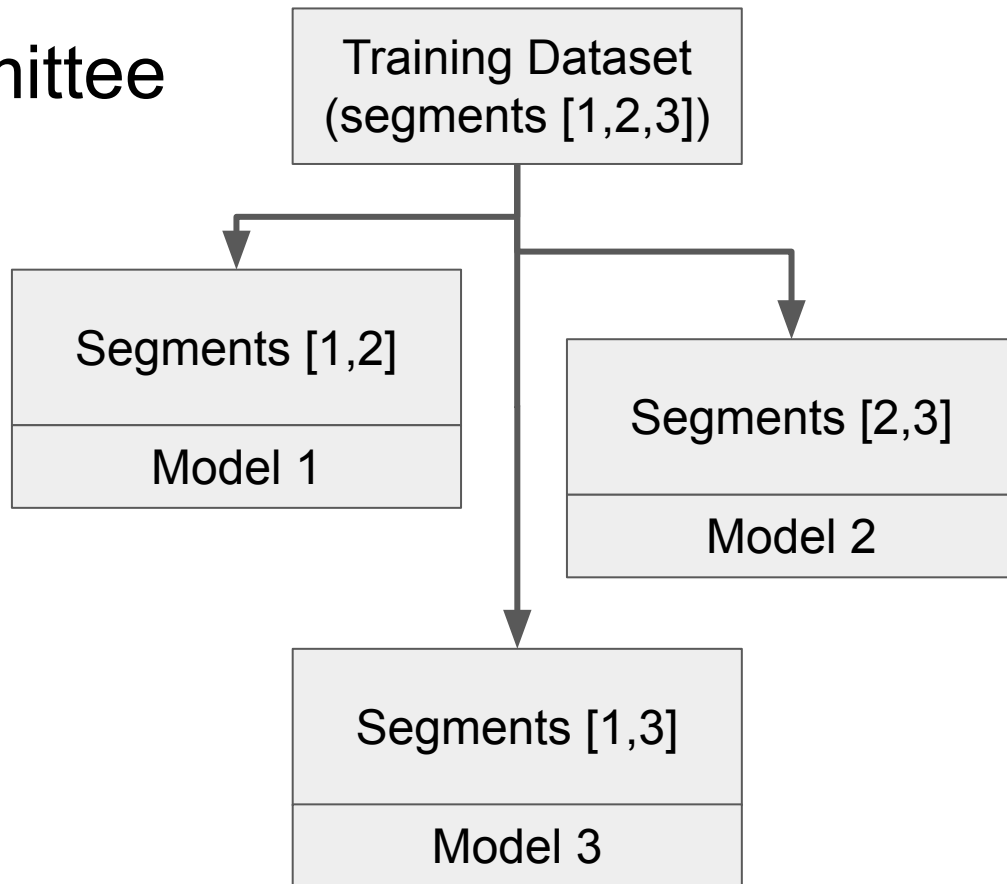
On average ~60% of data  
present in every replicate.



# Cross-validated Committee

**Segments** =  $n$  disjoint subsets of the dataset.

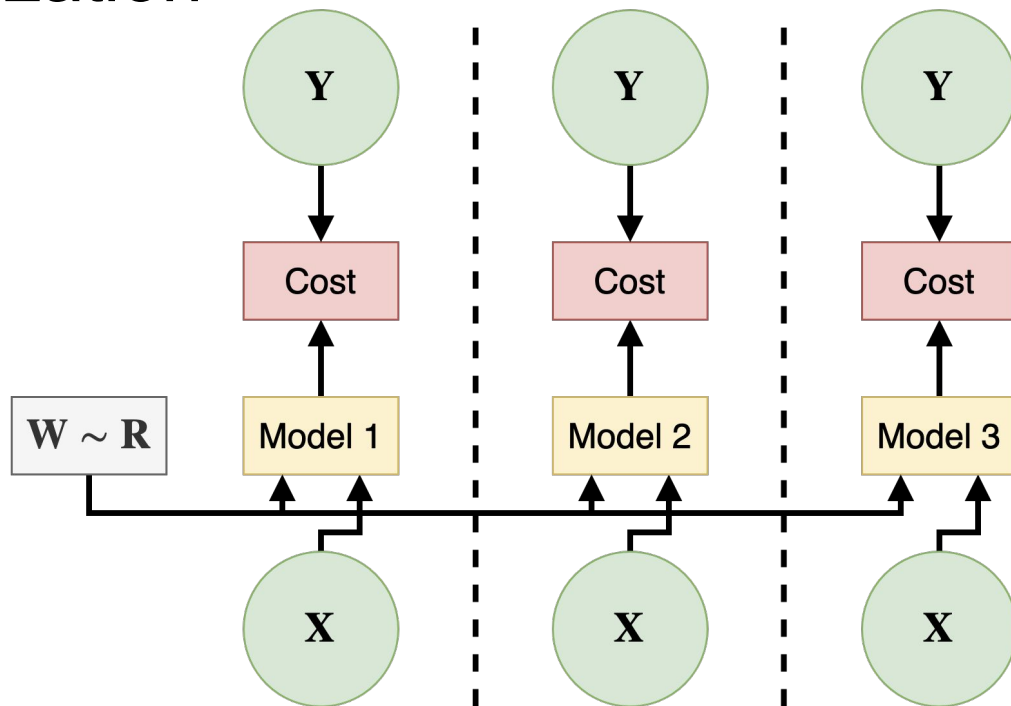
**Cross-validated committee** = an ensemble of  $n-1$  models trained on the dataset with different segments missing.



# Noise Injection: Initialization

Create multiple neural networks with **different initial parameters**.

After training, the networks will arrive at **different hypotheses**.



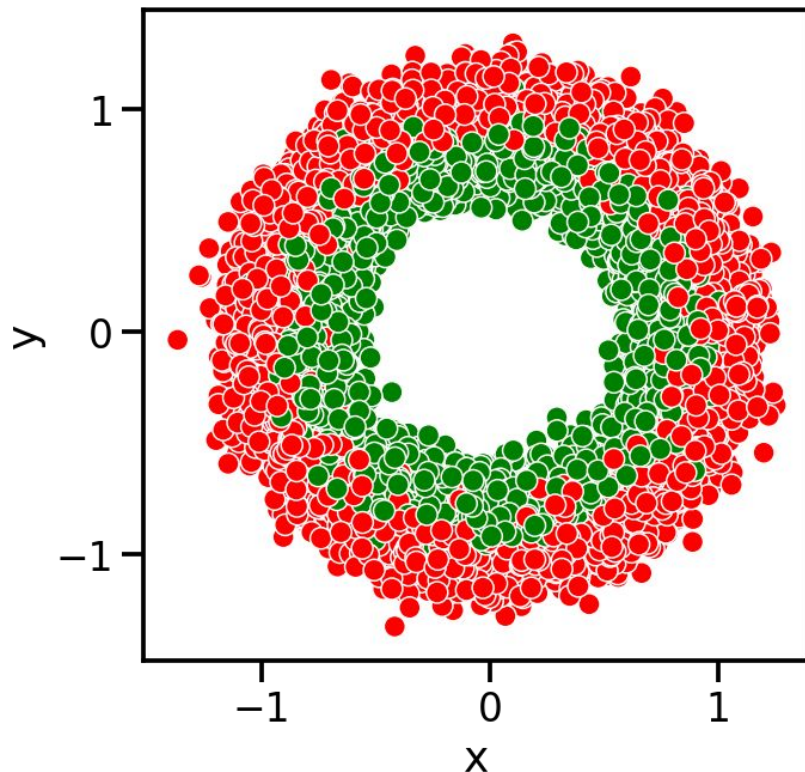
# Case Study: Gaussian Circles

## Dataset

Collection of 10K points in 2D space arranged as two circles with Gaussian noise.

## Task

Classify a point as either green or red given its coordinates.





# Logistic Regression #1

## Features

Coordinates X, Y, and constant

## Target

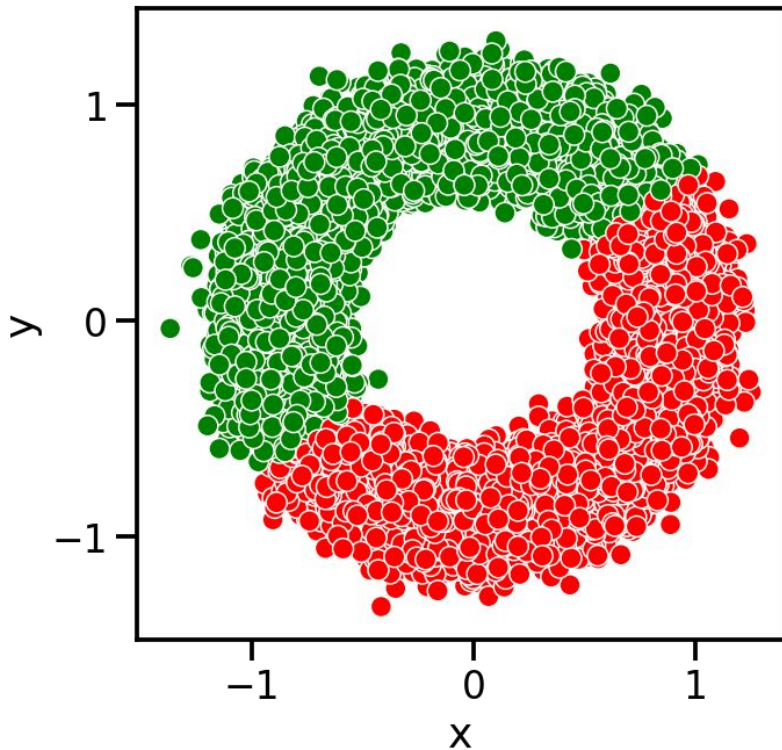
Red = 0, green = 1

## Accuracy

0.50

## Log Loss

0.693

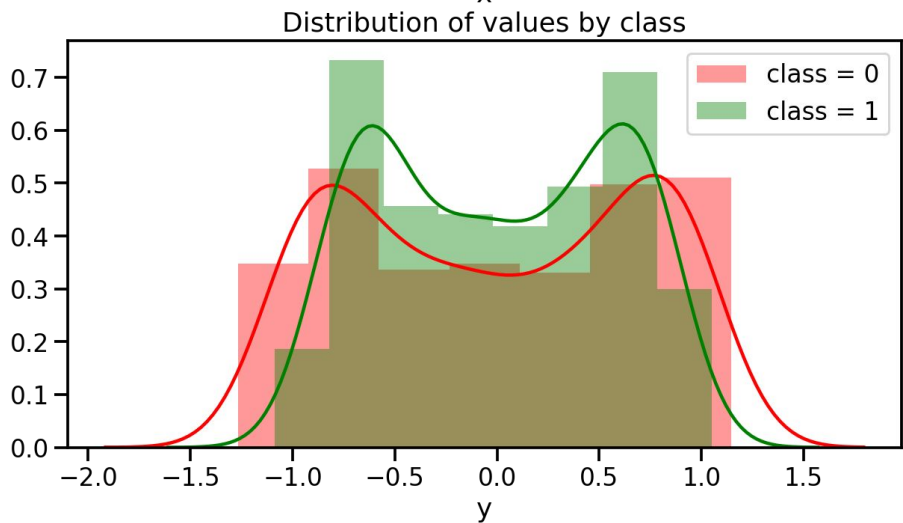
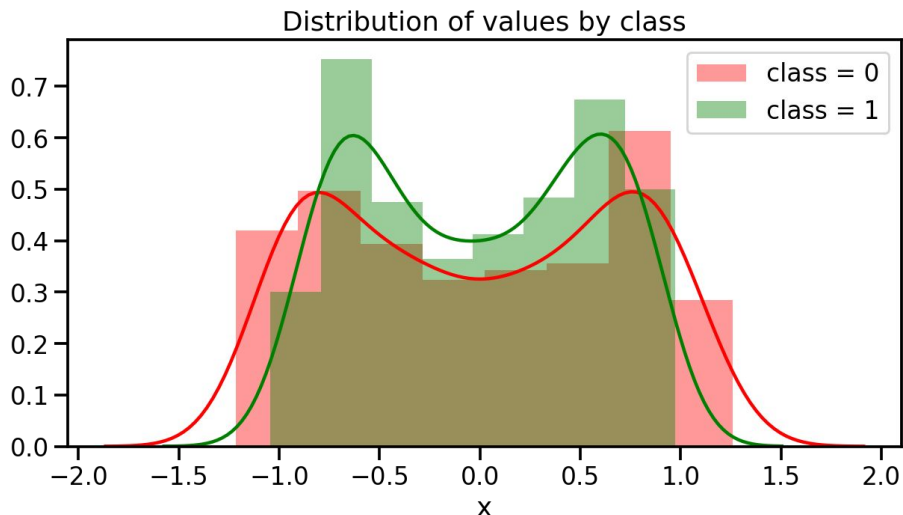


# Distribution by X and Y

## Inference from Distributions

There is no way to separate classes with a straight line in either X or Y space.

How to construct **a feature**  
**which can separate the two**  
**circles** with a line?

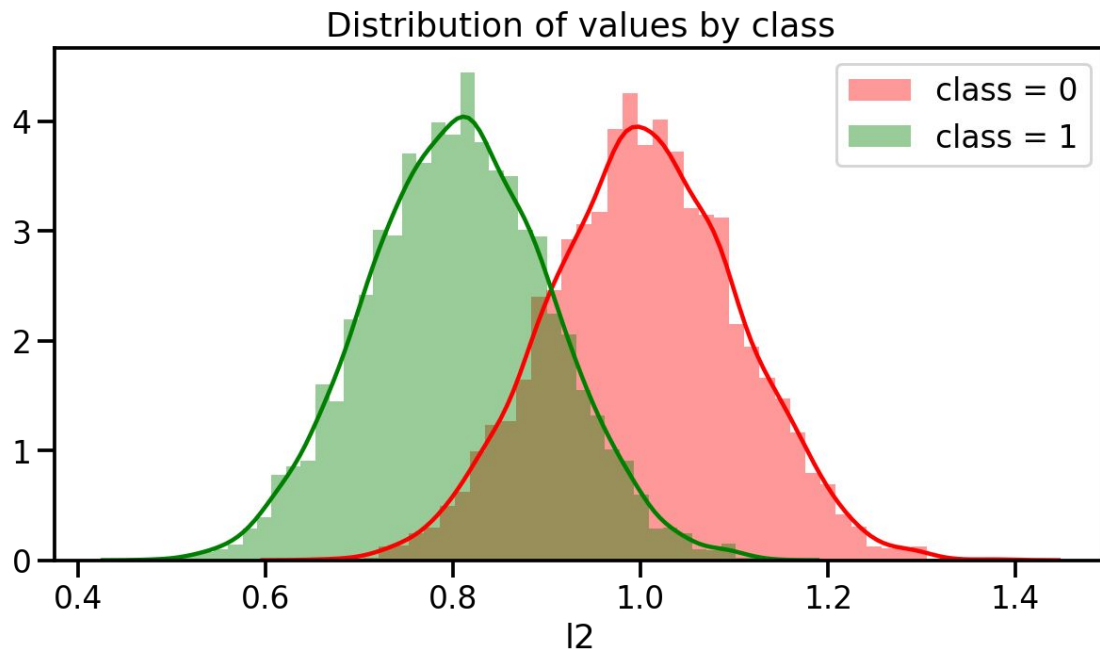


# Distribution by L2 Norm

L2 Norm of X and Y =  
distance from the center.

Distributions **partially**  
**separated** from each other.

Overlapping area implies that  
a **perfectly accurate**  
**classifier is not possible**.



# Logistic Regression #2

## Features

L2 norm, and constant

## Target

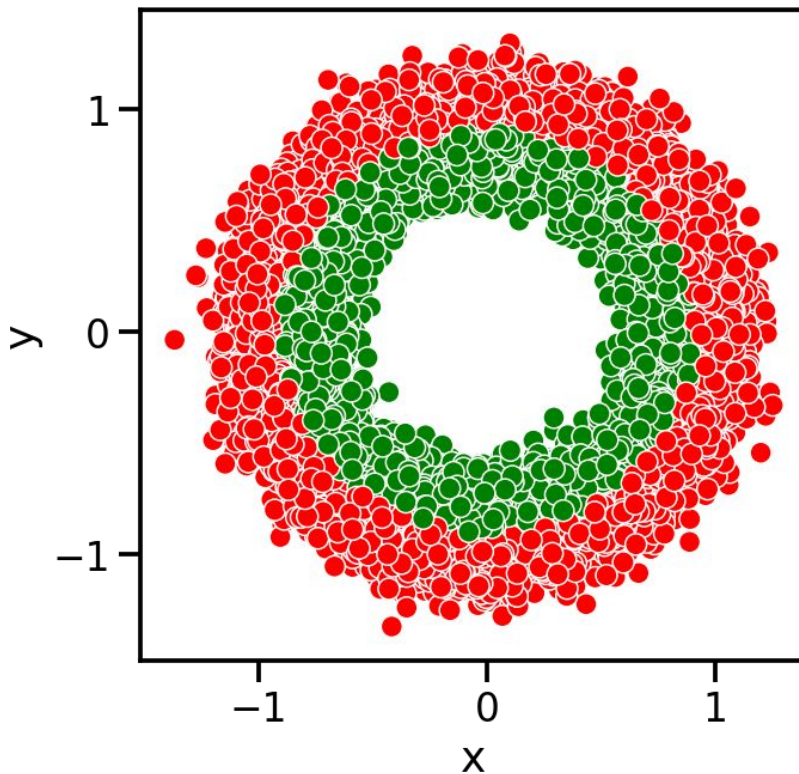
Red = 0, green = 1

## Accuracy

0.84

## Log Loss

0.36



# Neural Network

## Layers

1. FC with 10 nodes, tanh
2. FC with 1 node, sigmoid

## Features

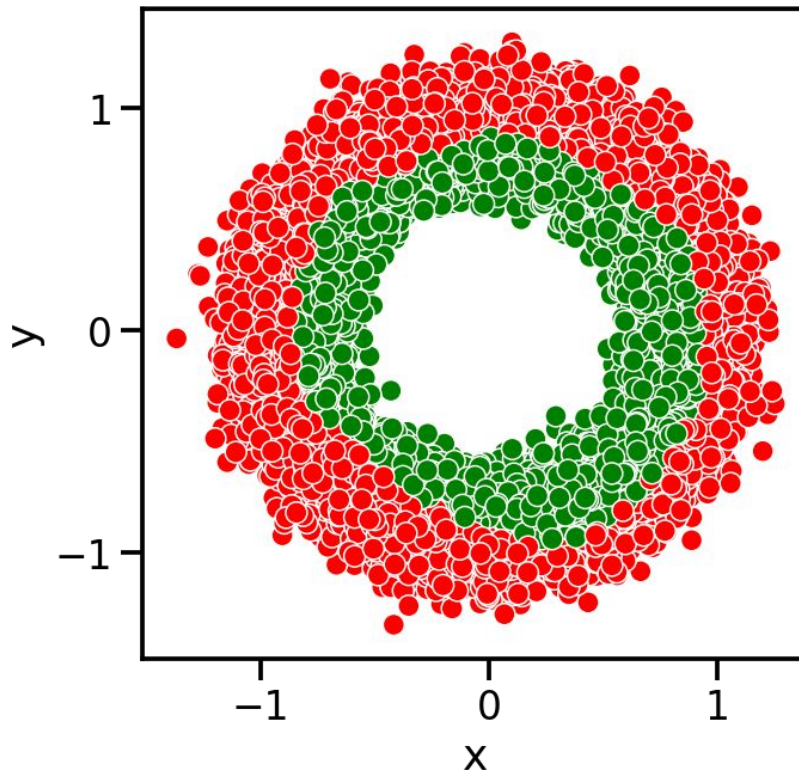
Coordinates of X and Y

## Target

Red = 0, green = 1

## Log Loss

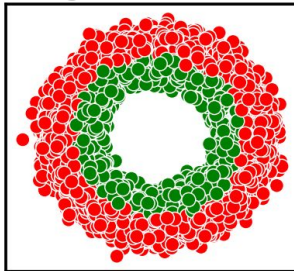
0.435



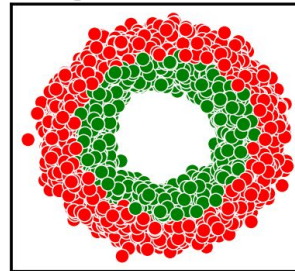
# Example: Initialization

1. Sample 9 different initialization seeds
2. Train each model for 10 epochs
3. Compare performance on the training dataset

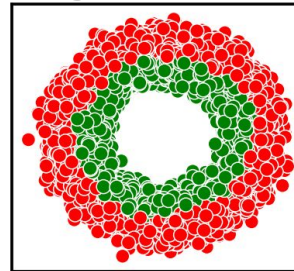
Log Loss = 0.401



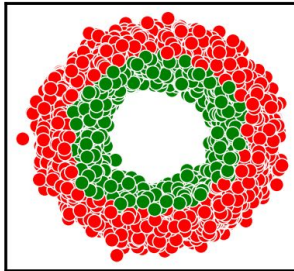
Log Loss = 0.388



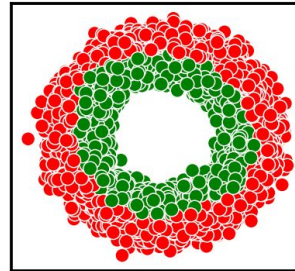
Log Loss = 0.384



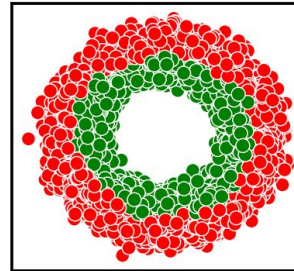
Log Loss = 0.408



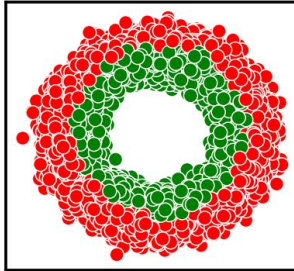
Log Loss = 0.389



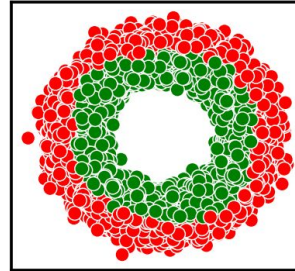
Log Loss = 0.389



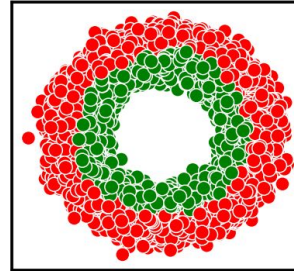
Log Loss = 0.415



Log Loss = 0.446



Log Loss = 0.389





# Ensemble: Initialization

## Members of Ensemble

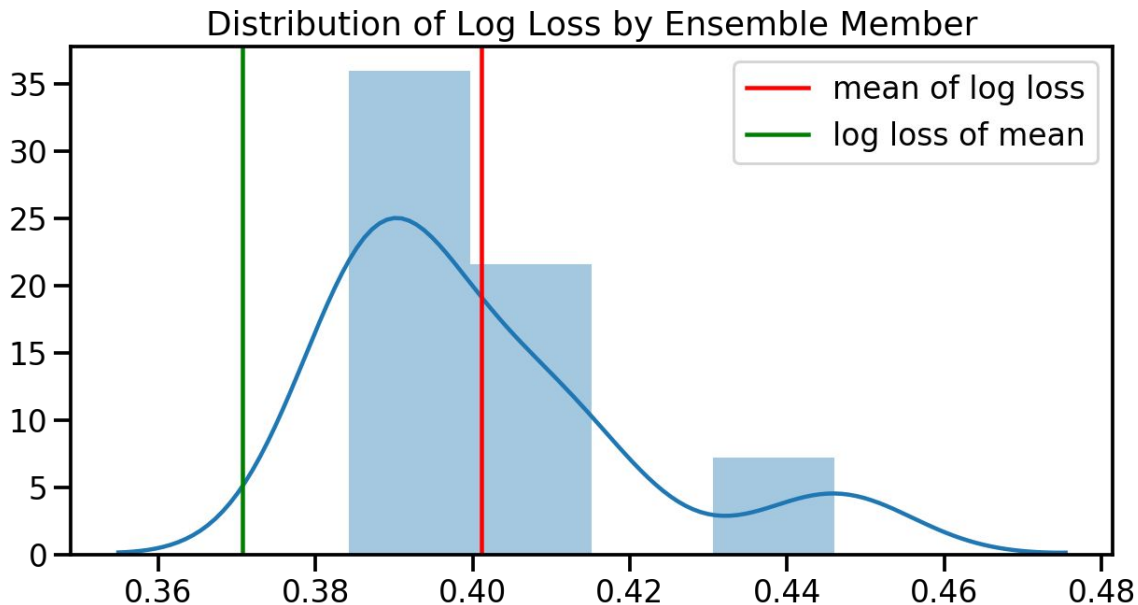
9 neural networks with  
different weight initializations

## Aggregation Method

Average of Probability

## Log Loss of Ensemble

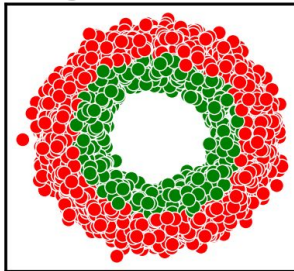
0.371



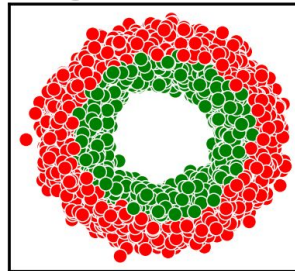
# Example: Bagging

1. Sample 20 different bootstrap replicates of the training dataset
2. Train each model for 10 epochs
3. Compare performance on the training dataset

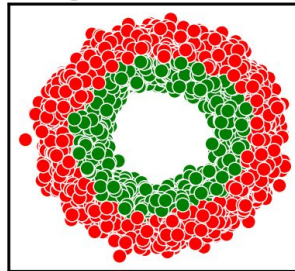
Log Loss = 0.401



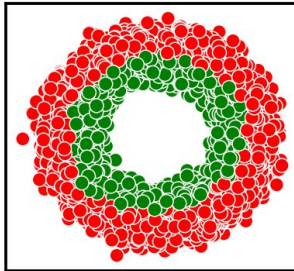
Log Loss = 0.388



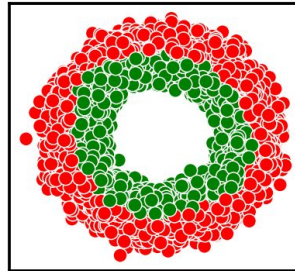
Log Loss = 0.384



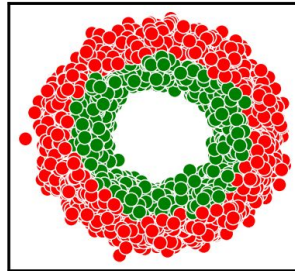
Log Loss = 0.408



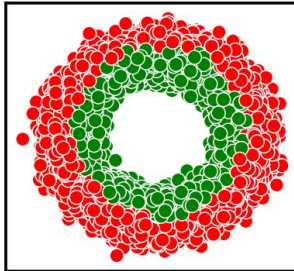
Log Loss = 0.389



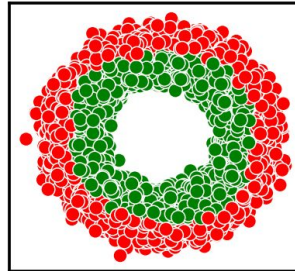
Log Loss = 0.389



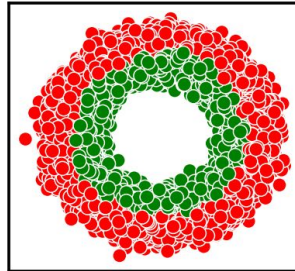
Log Loss = 0.415



Log Loss = 0.446



Log Loss = 0.389





# Ensemble: Bagging

## Members of Ensemble

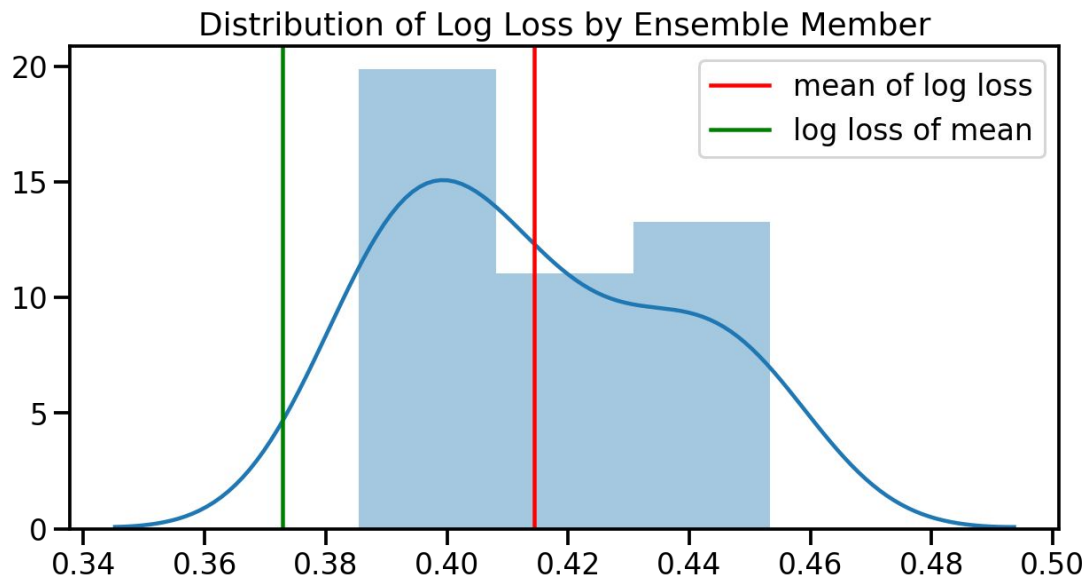
20 neural networks with  
different training bags

## Aggregation Method

Average of Probability

## Log Loss of Ensemble

0.373

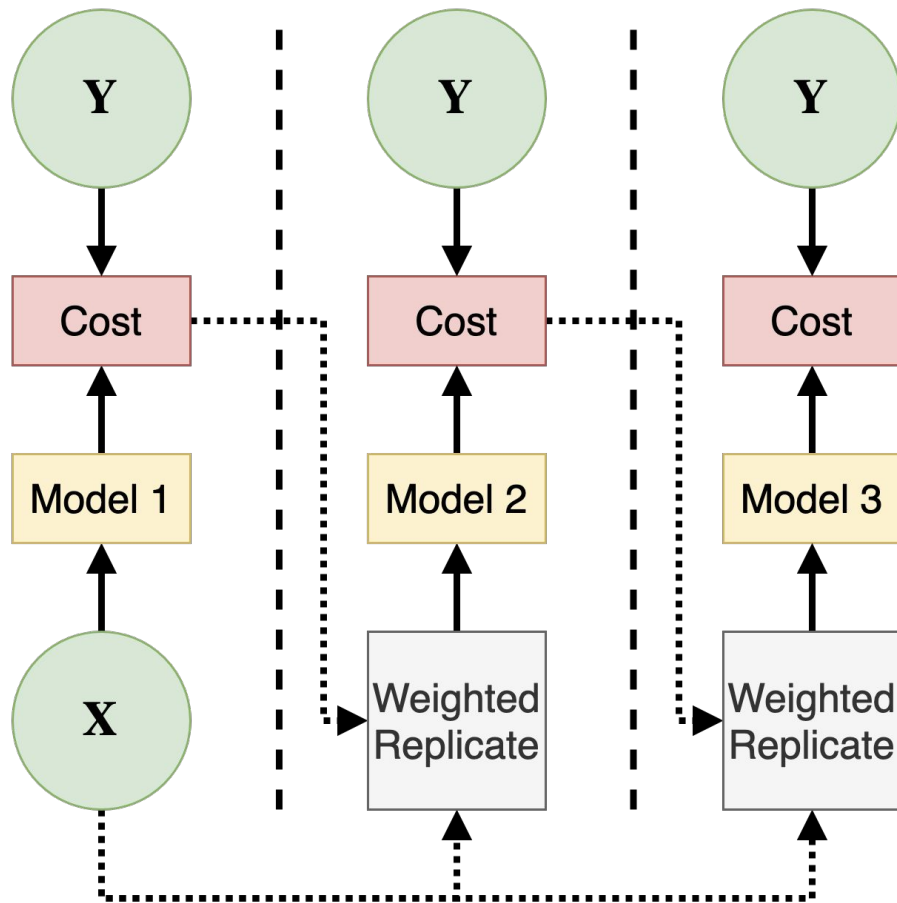


# AdaBoost: Training #1

Sequential model training on different datasets.

Each dataset resampled with probabilities depending on the **probabilities depending on the per-sample cost** of the previous model.

Each subsequent model is trained with higher focus on hard training samples.

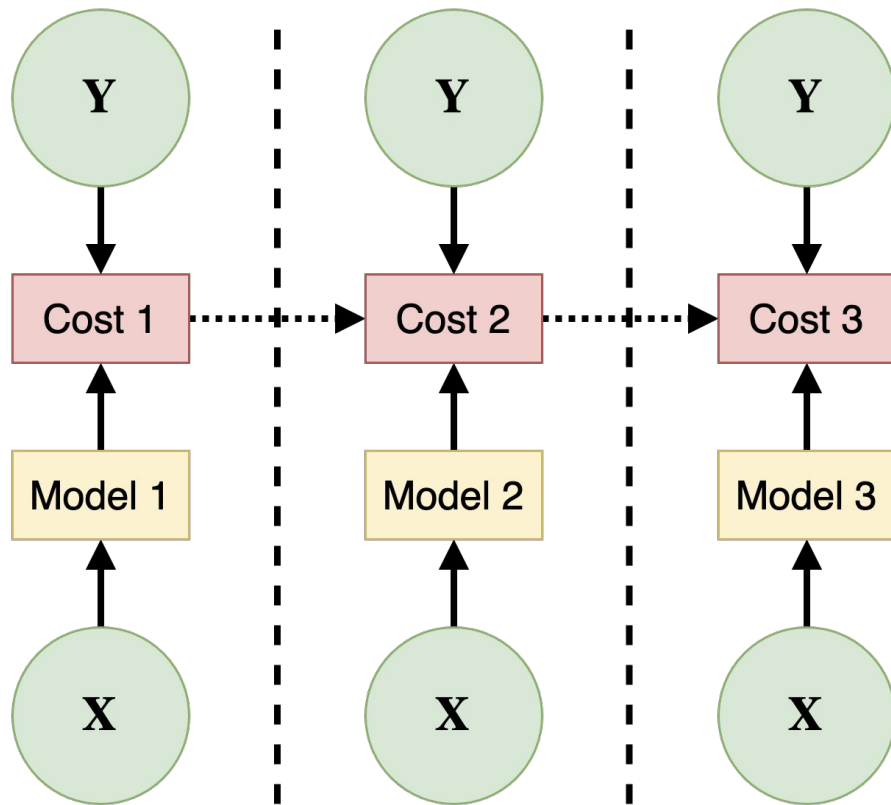


# AdaBoost: Training #2

Sequential model training on the same dataset.

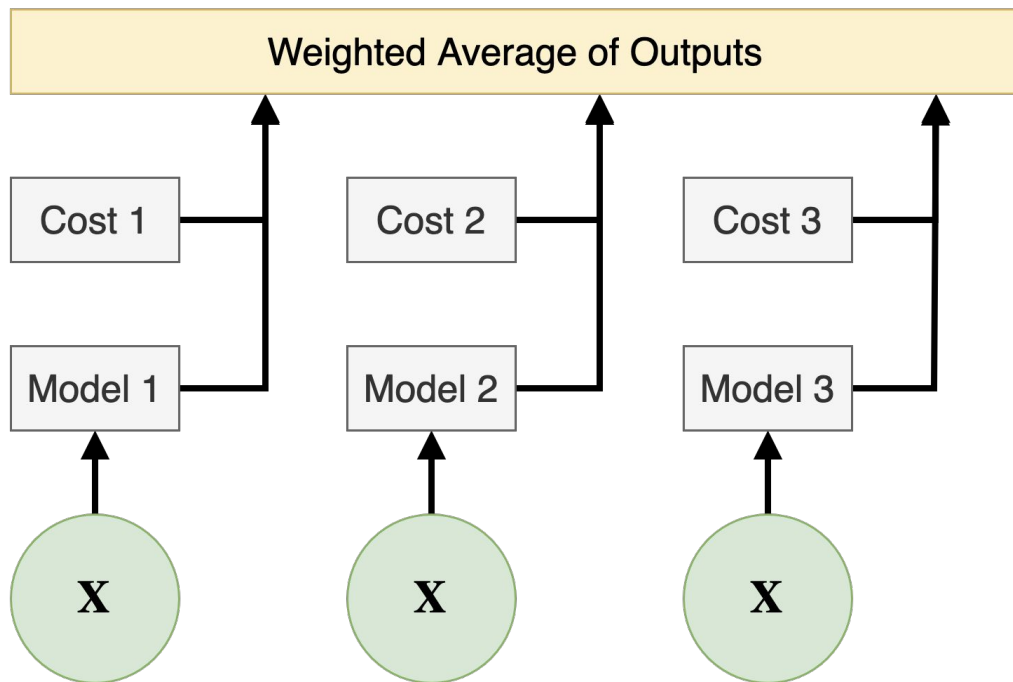
Cost = average of this model's per-example cost weighted by per-example cost from the previous model.

Each subsequent model is trained with higher focus on hard training examples.



# AdaBoost: Inference

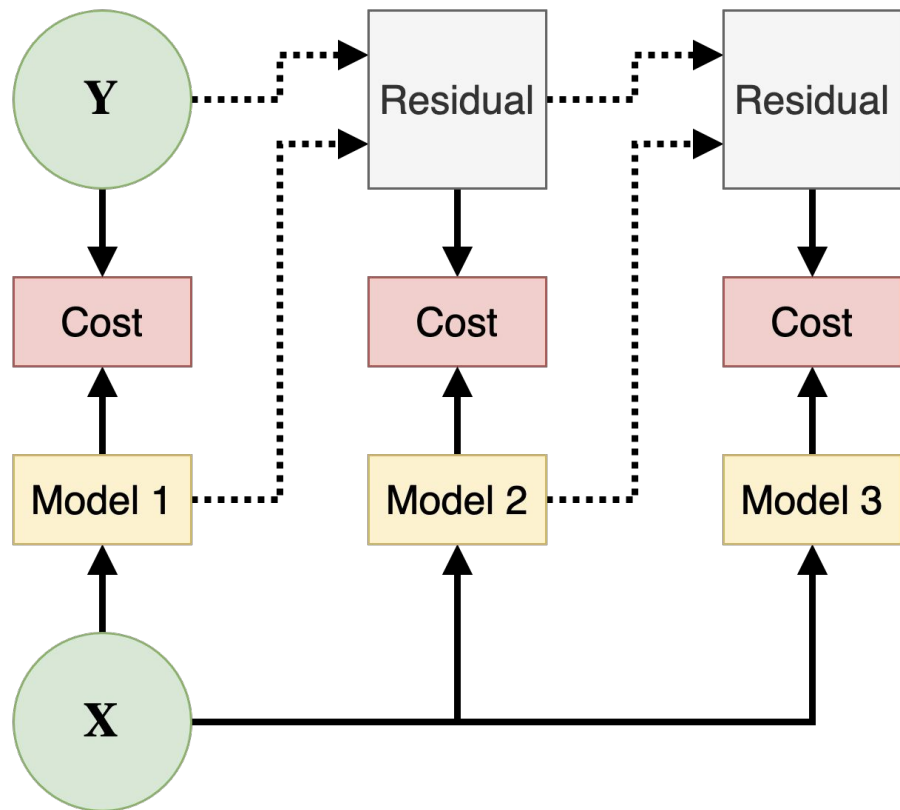
Ensemble output is the **average over each model's output weighted by the performance** on the training dataset.



# Gradient Boosting: Training

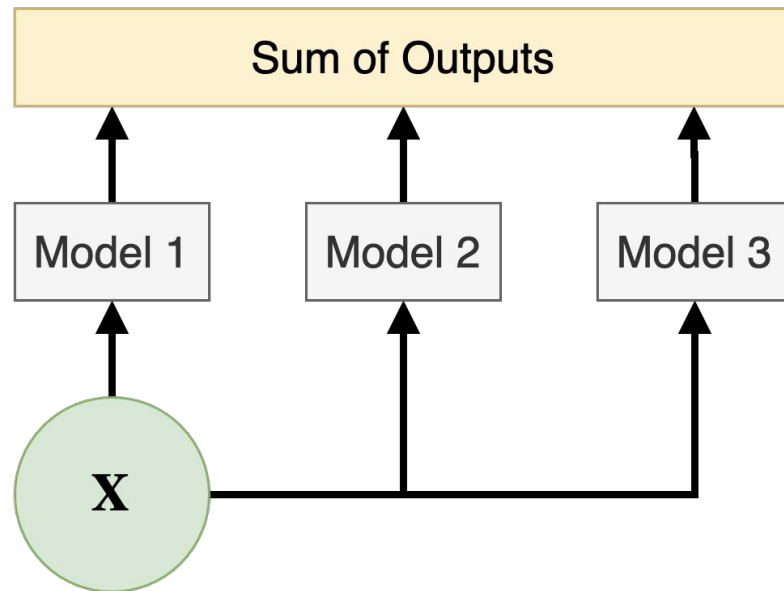
Sequential model training on the same dataset.

Each subsequent model is trained to output the residual of the previous model.



# Gradient Boosting: Inference

Ensemble output is the sum of outputs of each of the models.



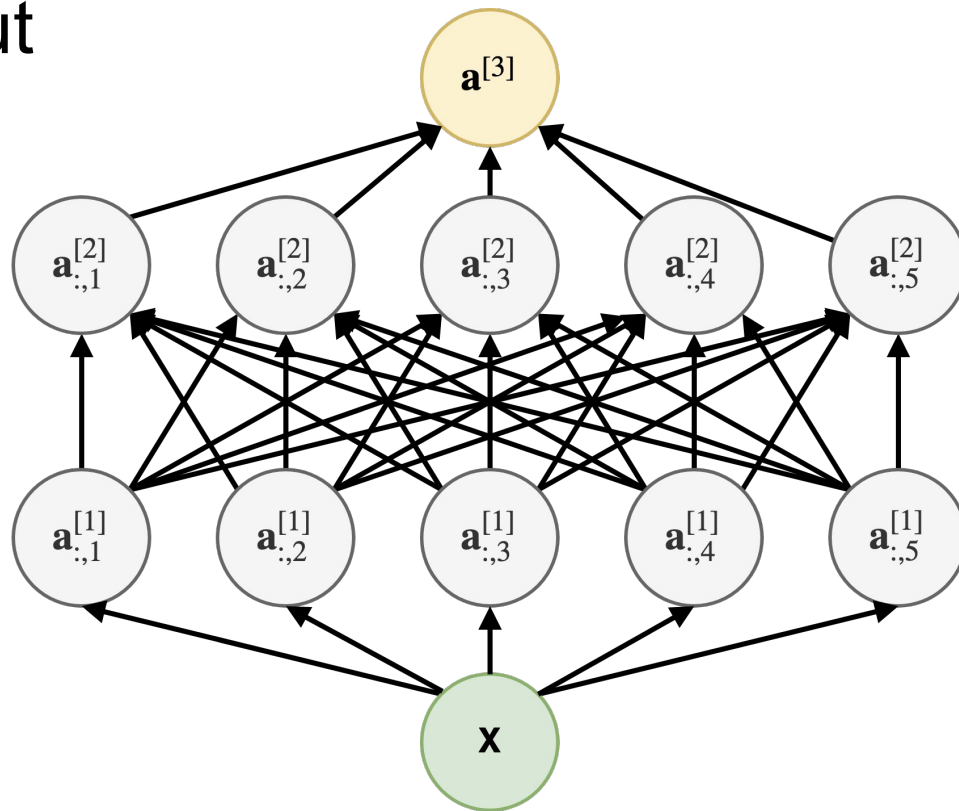
# Noise Injection: Dropout

## During training

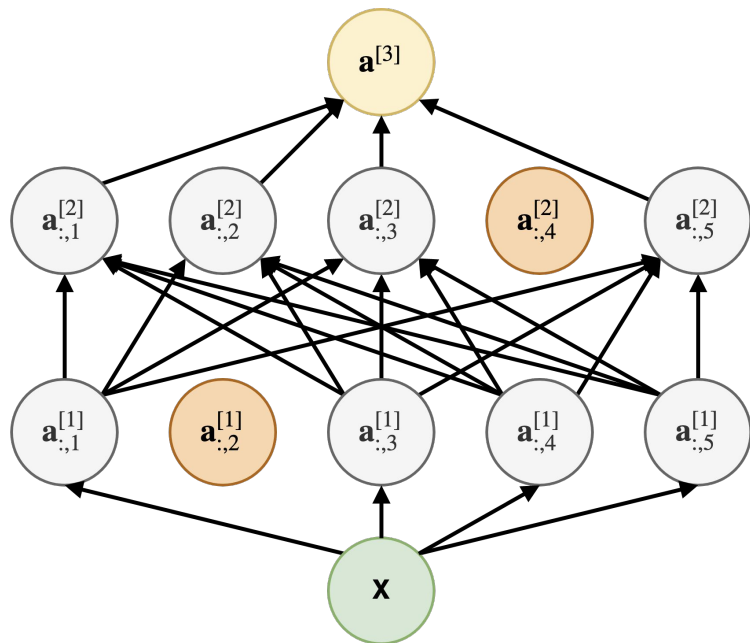
On each iteration, randomly choose and disable nodes with fixed probability.

## During inference

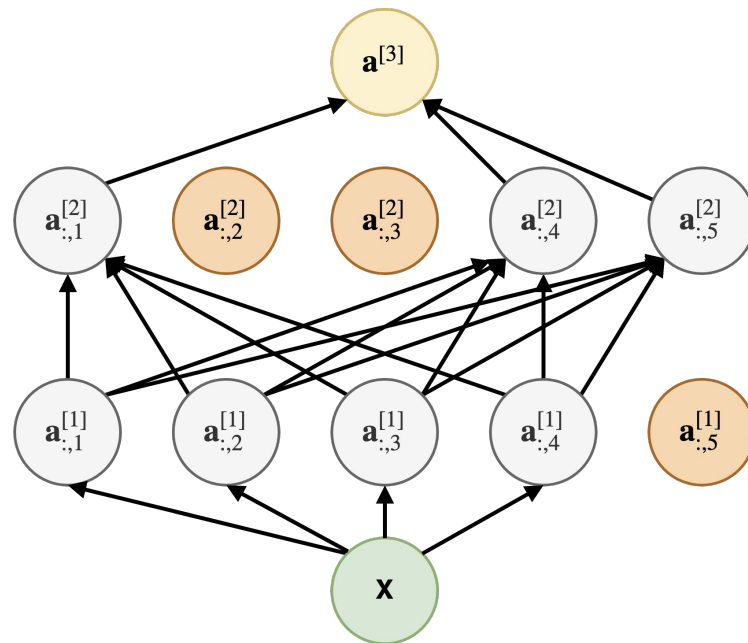
Use all nodes



# Noise Injection: Dropout



Iteration 1



Iteration 2

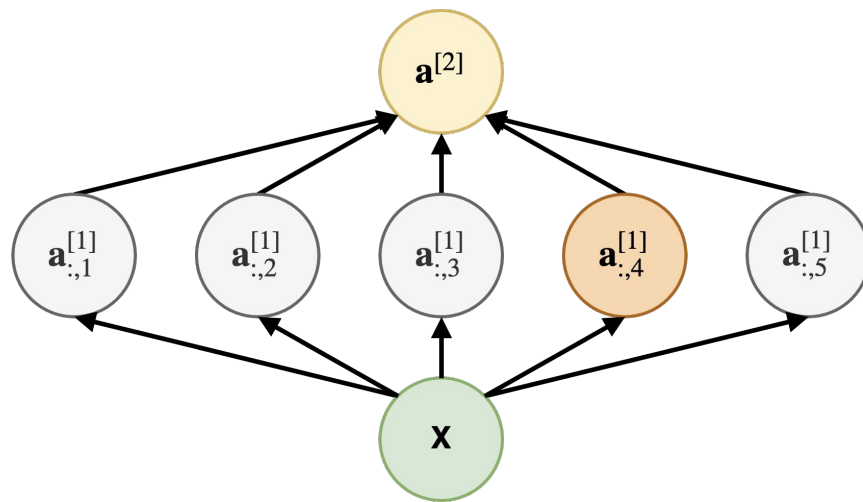


# Noise Injection: Dropout

Let's define:

- $p$  = probability of keeping a node
- $\mathbf{d}$  = vector of random variables  $\sim \text{Bernoulli}(p)$

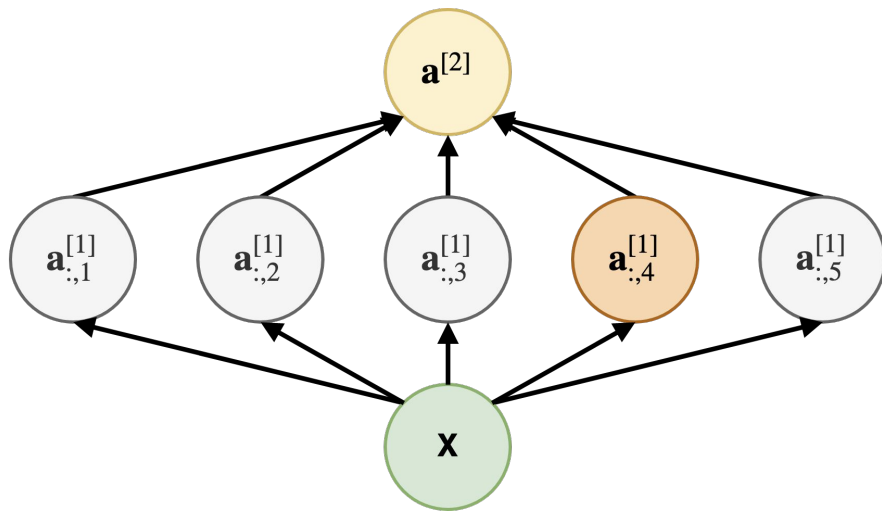
$$\tilde{\mathbf{a}}_{:,m}^{[1]} = \frac{\mathbf{a}_{:,m}^{[1]} * d_m}{p}$$



# Noise Injection: Dropout

$$\tilde{\mathbf{a}}_{:,m}^{[1]} = \frac{\mathbf{a}_{:,m}^{[1]} * d_m}{p}$$

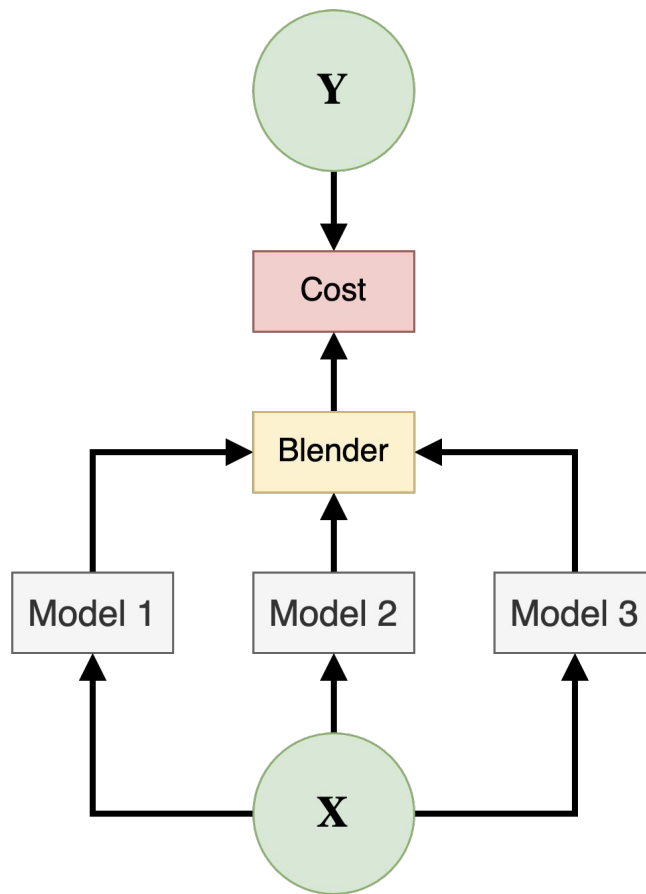
$$\begin{aligned}\mathbb{E}[\tilde{\mathbf{a}}_{s,:}^{[1]}] &= \frac{\mathbb{E}[\mathbf{a}_{s,:}^{[1]}] * \mathbb{E}[\mathbf{d}]}{p} \\ &= \frac{\mathbb{E}[\mathbf{a}_{s,:}^{[1]}] * p}{p} \\ &= \mathbb{E}[\mathbf{a}_{s,:}^{[1]}]\end{aligned}$$



# Stacking: Forward Pass

Instead of taking a simple average, the aggregation step can be an **independent model taking outputs of ensemble members as input**.

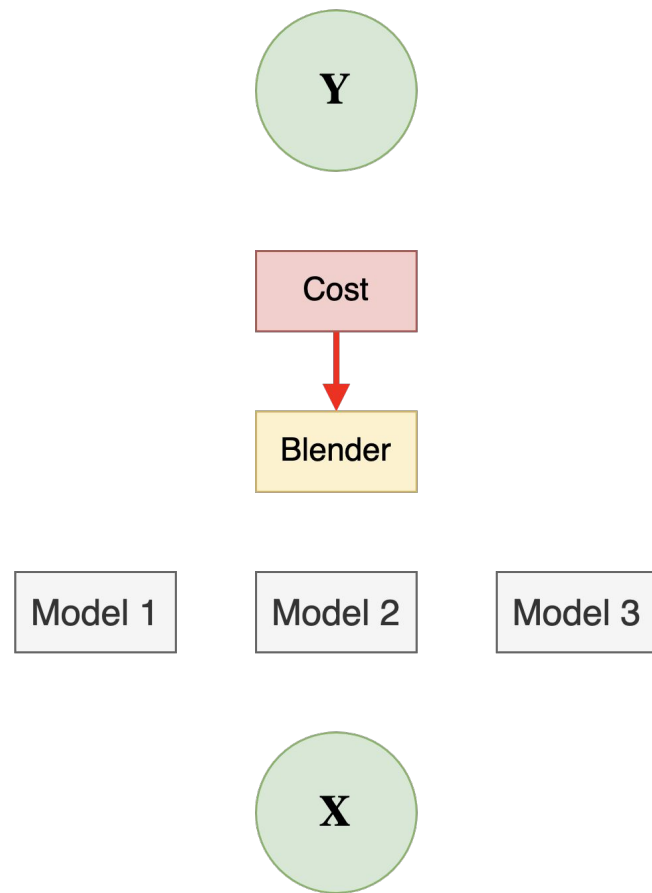
Blender is trained after the ensemble models have been trained.



# Stacking: Backward Pass

Instead of taking a simple average, the aggregation step can be an **independent model taking outputs of ensemble members as input**.

Blender is trained after the ensemble models have been trained.



# Examples from Previous Papers

Model	Diversity Source
GoogLeNet	Dropout + Batch seed
ResNet	Network depth
VGGNet	Dropout + Image size
AlexNet	Dropout + ?

# Final Project

## Dataset

70K images, 28x28 grayscale, 10 classes

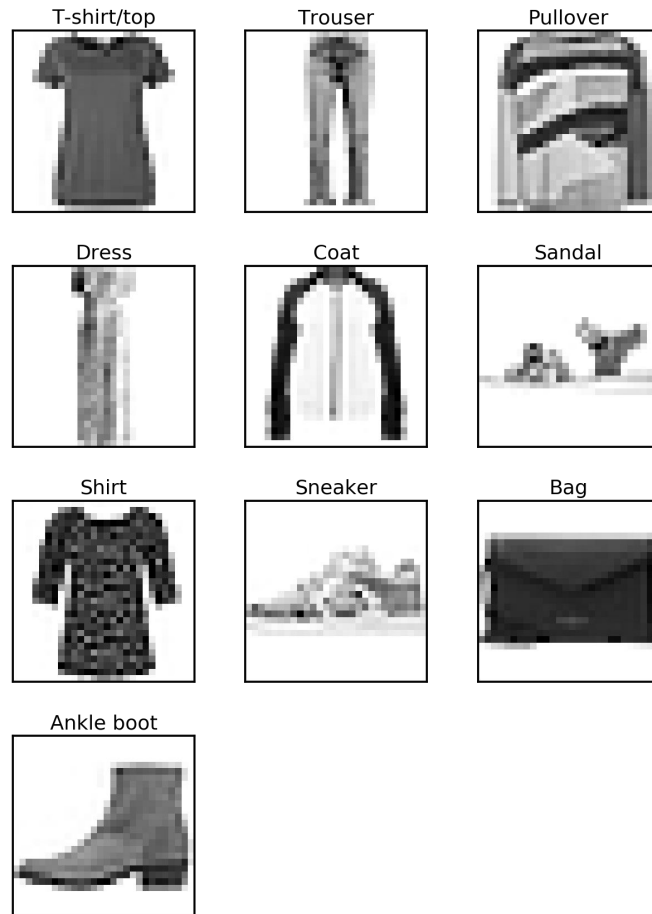
## Task

Image classification

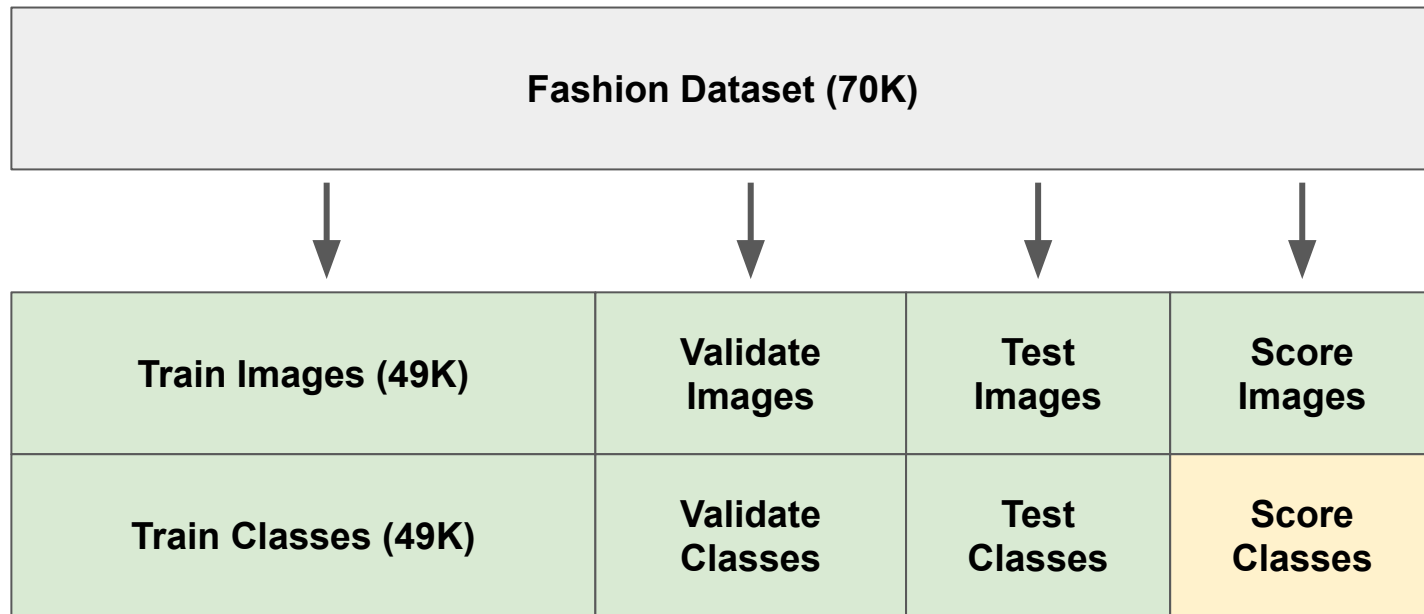
Any architecture, any training procedure,  
and preprocessing, and ensemble method.

## Deadline

EOD November 10th, 2020



# Assignment Dataset



# Demo

Assignment Template



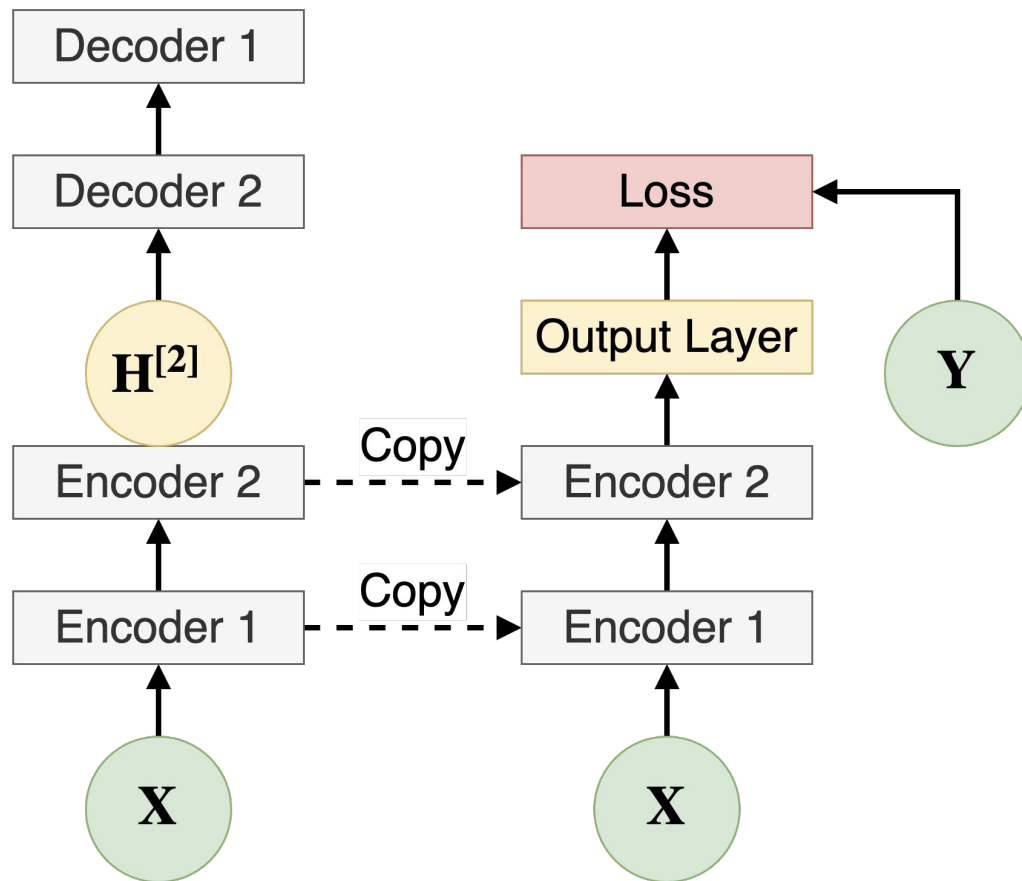
# Demo

BigGAN Solution

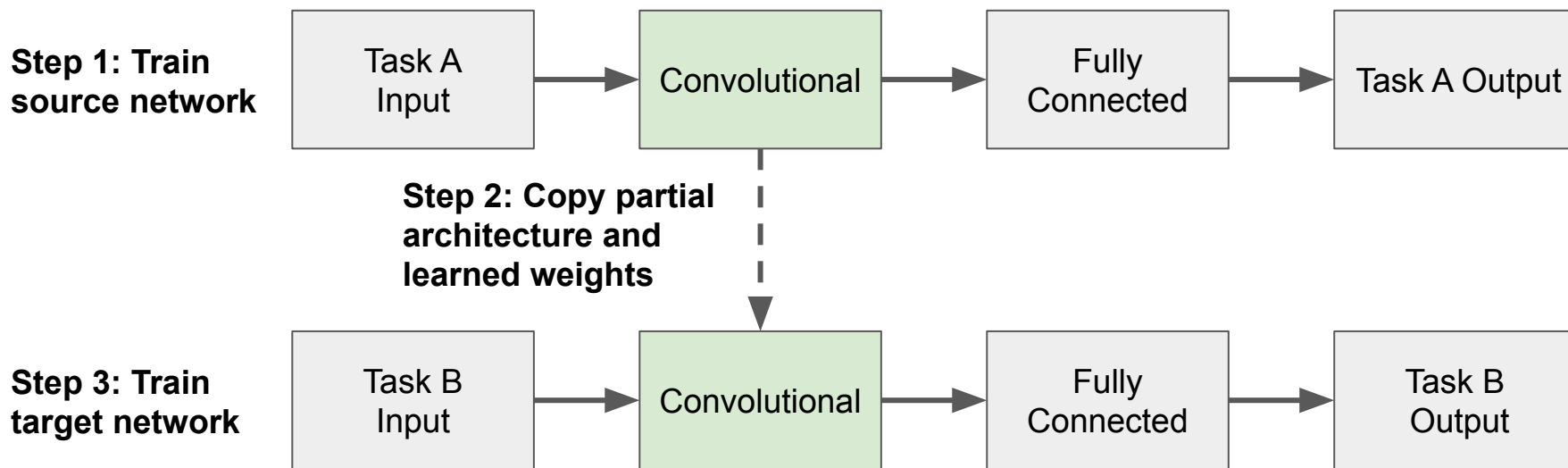
# Managerial Economics of Machine Learning

Project Goal	Project Duration	Training Cost	Required Training
Design new architecture and train from scratch	Months	Expensive	Machine Learning Researcher
Use pre-existing architecture and/or transfer learning	Weeks	Cheaper	Machine Learning Engineer
Find a pre-trained model and/or buy Inference as a Service API	Days	Free	Automation Software Engineer

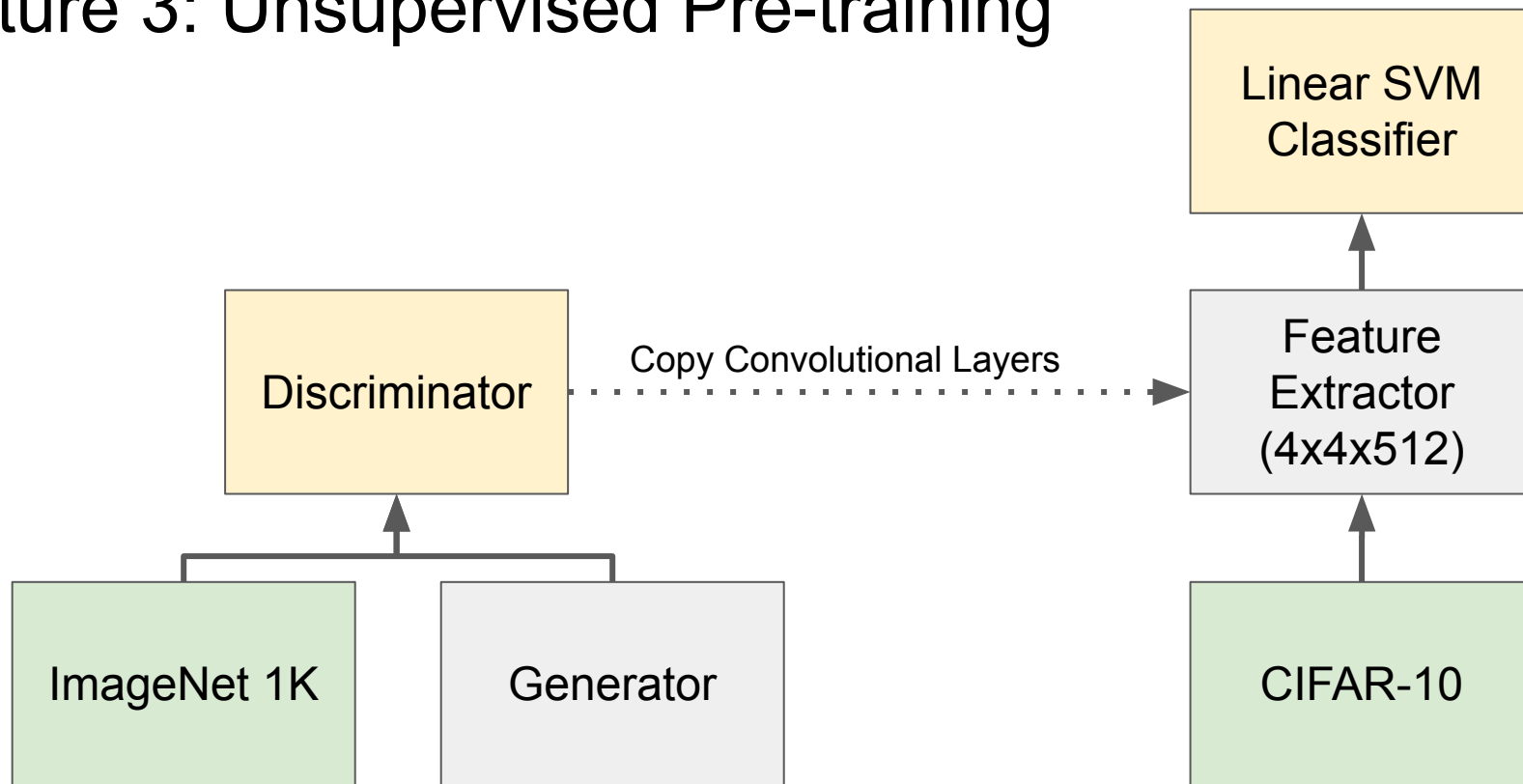
# Lecture 1: Unsupervised Pre-training



# Lecture 2: Transfer of Supervised Learning



# Lecture 3: Unsupervised Pre-training



# AWS Rekognition

## Product

~ \$0.001 / image

Available as a convenient API

## Features

Object/scene detection

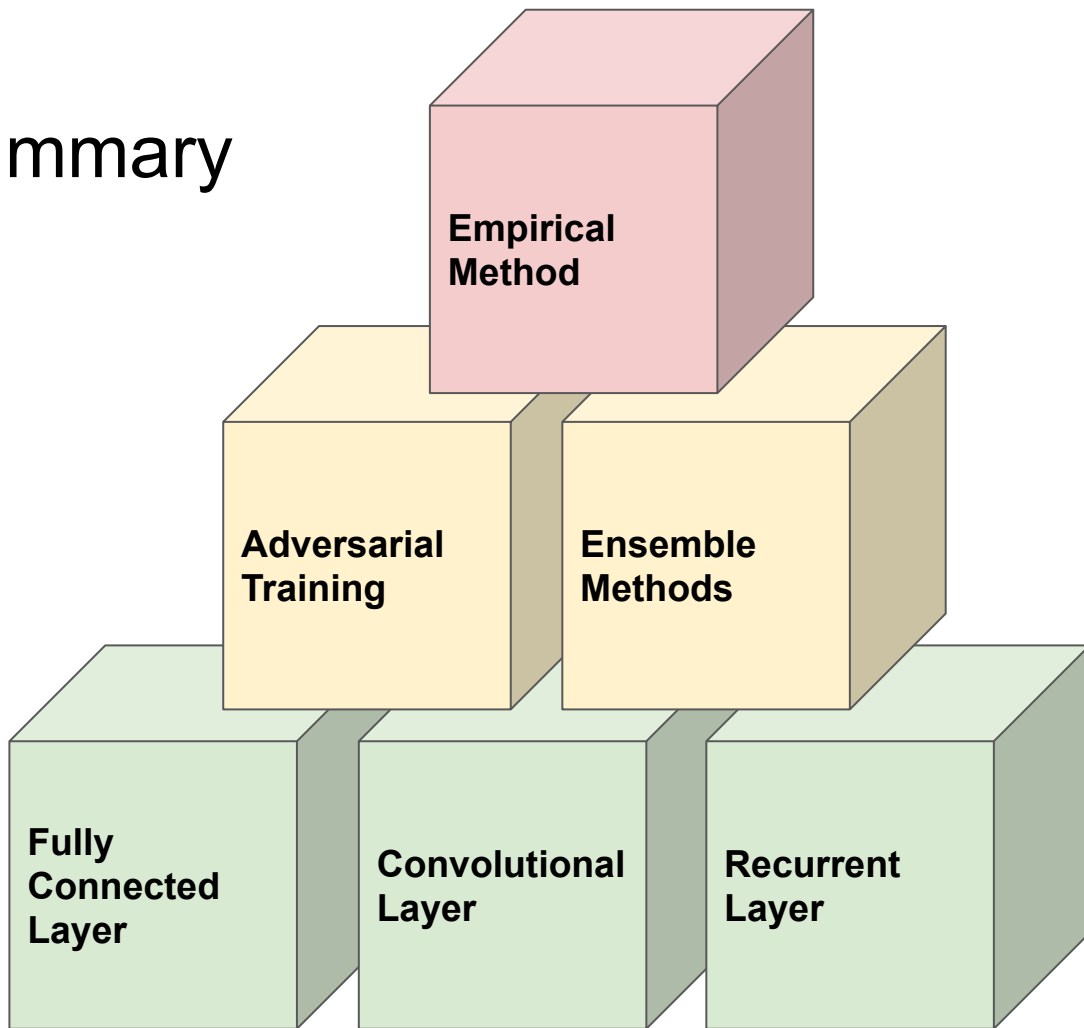
Face recognition and analysis

Both photo and video input

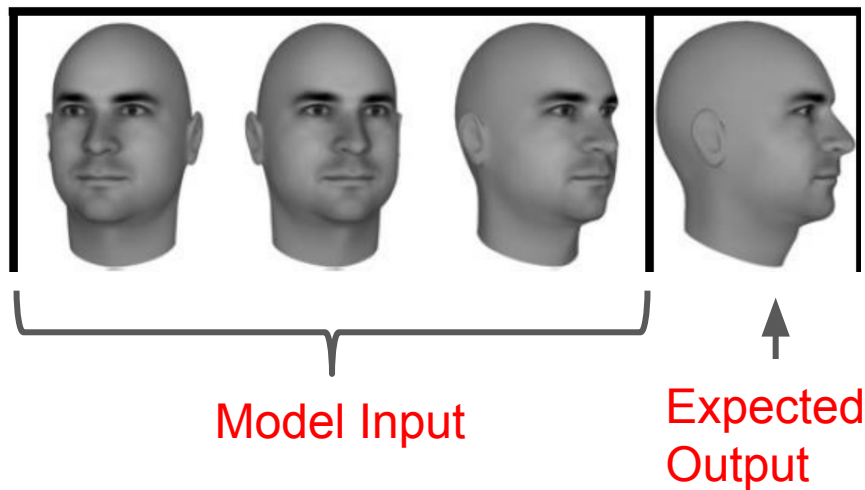
Image to text



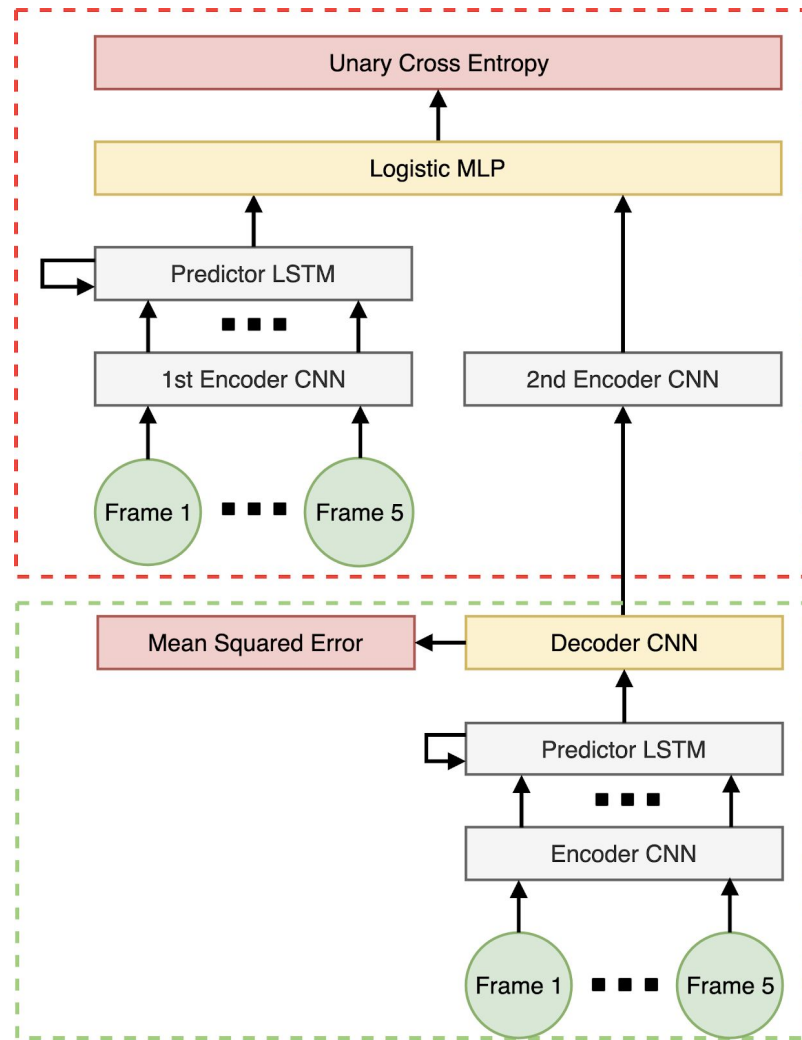
# Course Summary



# Lecture 4: PGN

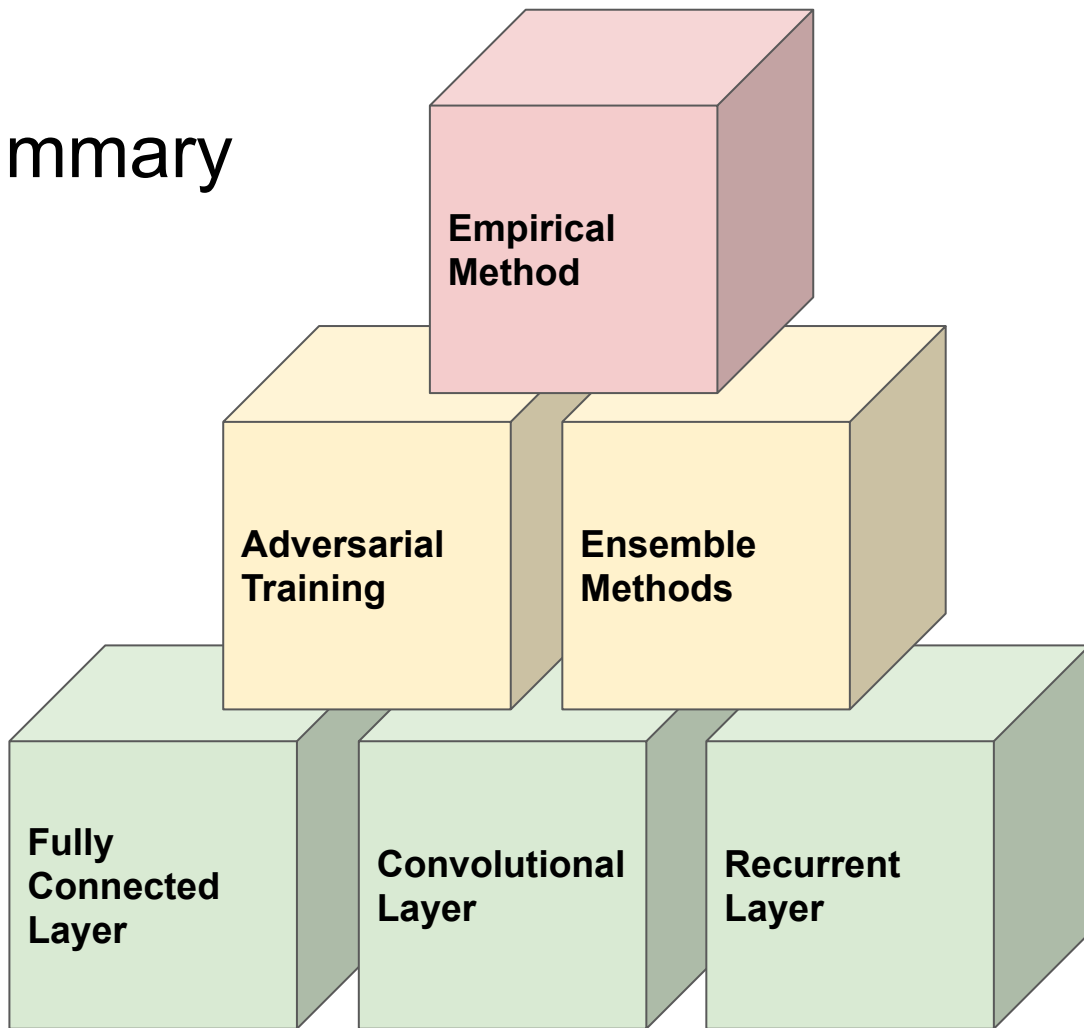


Lotter, W., et al. (2016)





# Course Summary



# References

- A Short Introduction to Boosting (Fruend, Y., and Schapire, R. E., 1999)
- Dropout: A Simple Way to Prevent Neural Networks from Overfitting (Srivastava, N., et al., 2014)
- Ensemble Methods in Machine Learning (Diettrich, T. G., 2000)

# Thank you

(Really) Final Q&A Time