Decision Trees, Ensembles

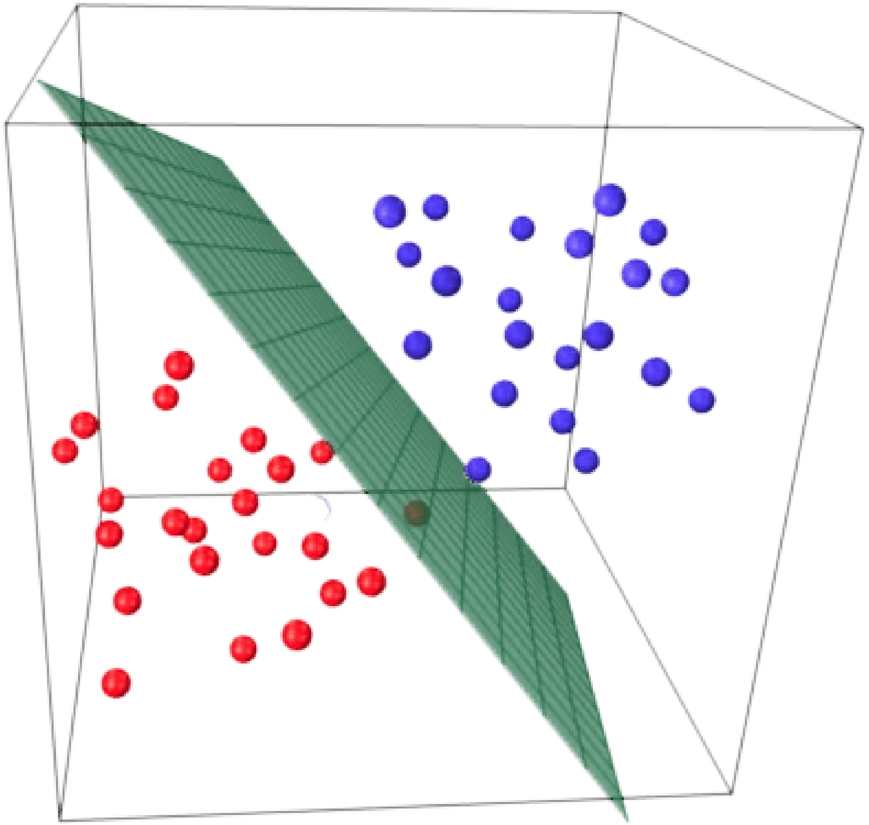
Robin Jia
USC CSCI 467, Spring 2024
March 5, 2024

Previously: Reliance on Linear Layers

- Linear models
  - Linear regression, logistic regression, softmax regression
  - Classification: Decision boundary is defined by

    $$w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b = 0$$

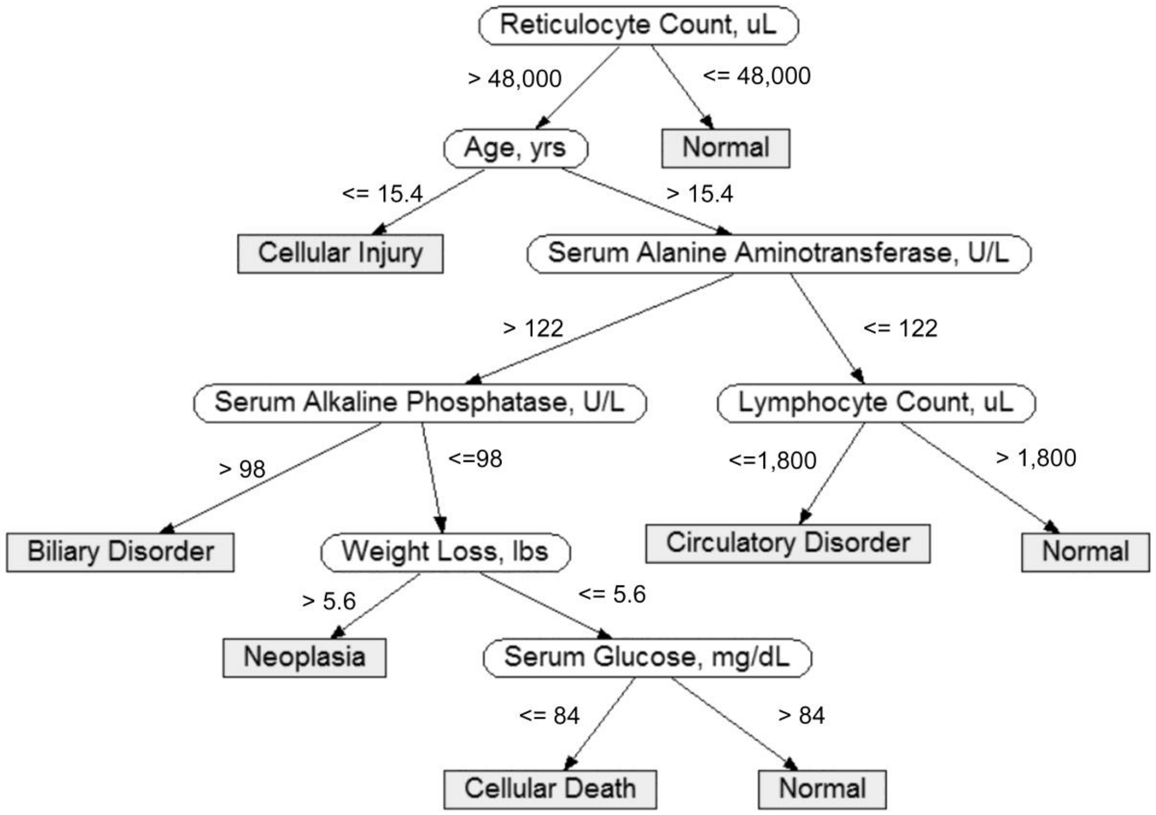
  - Note: Combination of ***every*** feature x_i
    - Not necessarily how humans make decisions
    - Can be hard to understand why a prediction was made
- Neural networks
  - Linear layers are core building blocks
  - Final decision boundary is linear function of learned features



Modeling decision making

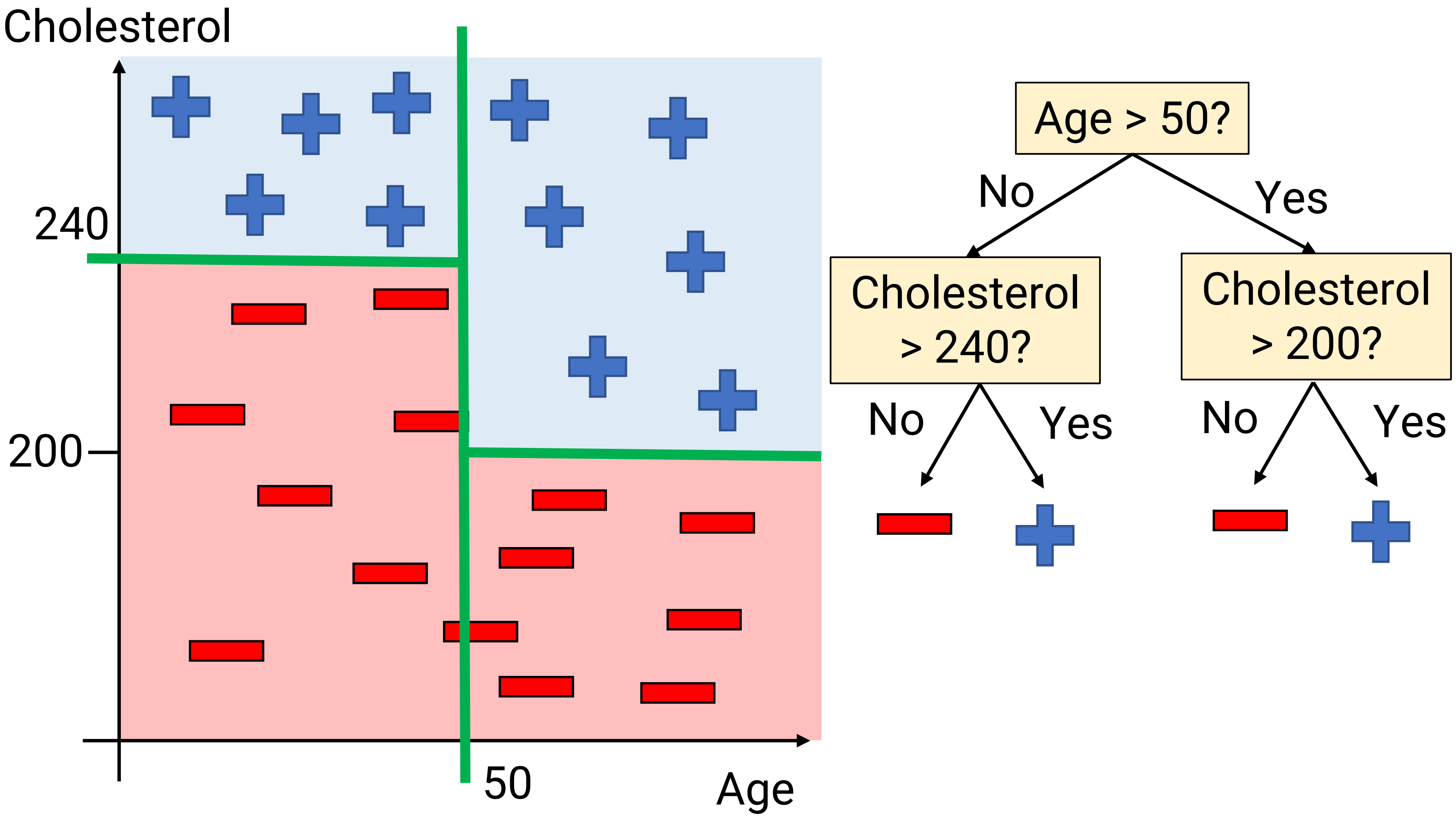
- Human experts make complex decisions and predictions every day
  - E.g., Given observations about a patient, what disease do they have?
- Doesn't really look like a linear function; more like a flow chart
- Can we build models that emulate the human decision-making process?

Hepatic Disorders Decision Tree



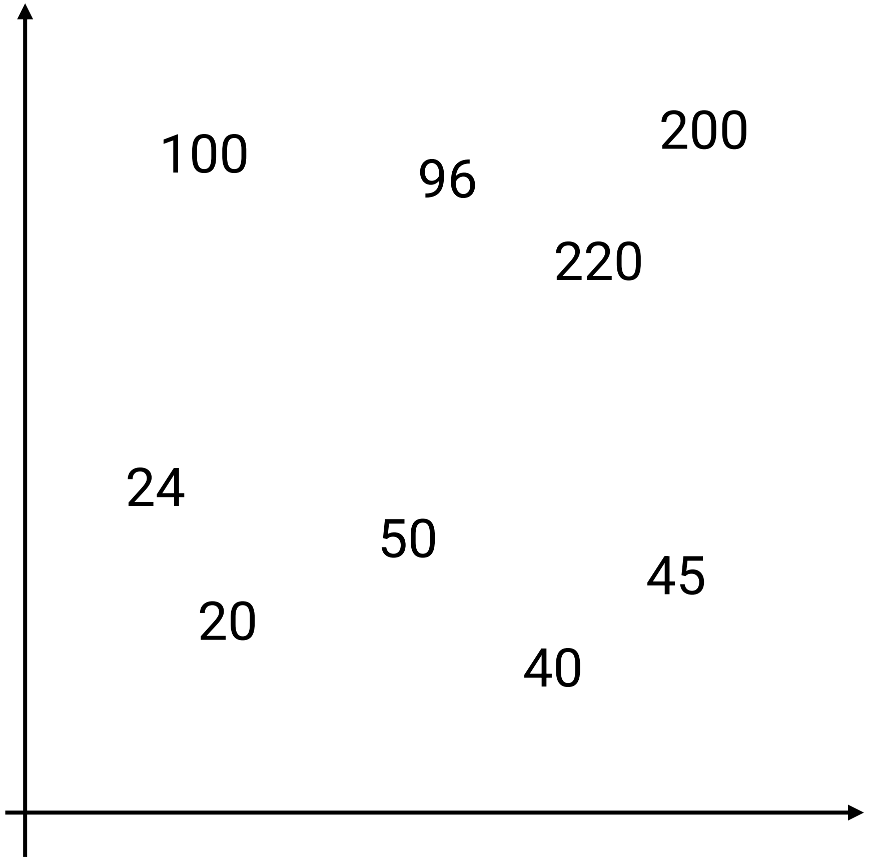
Decision Trees

- At each node, split on one feature
- Remember the best output at each leaf node
    - Classification: Majority class
    - Regression: Mean within node
- Given new example, find which leaf node it belongs to and predict the associated output
- Interpretable!



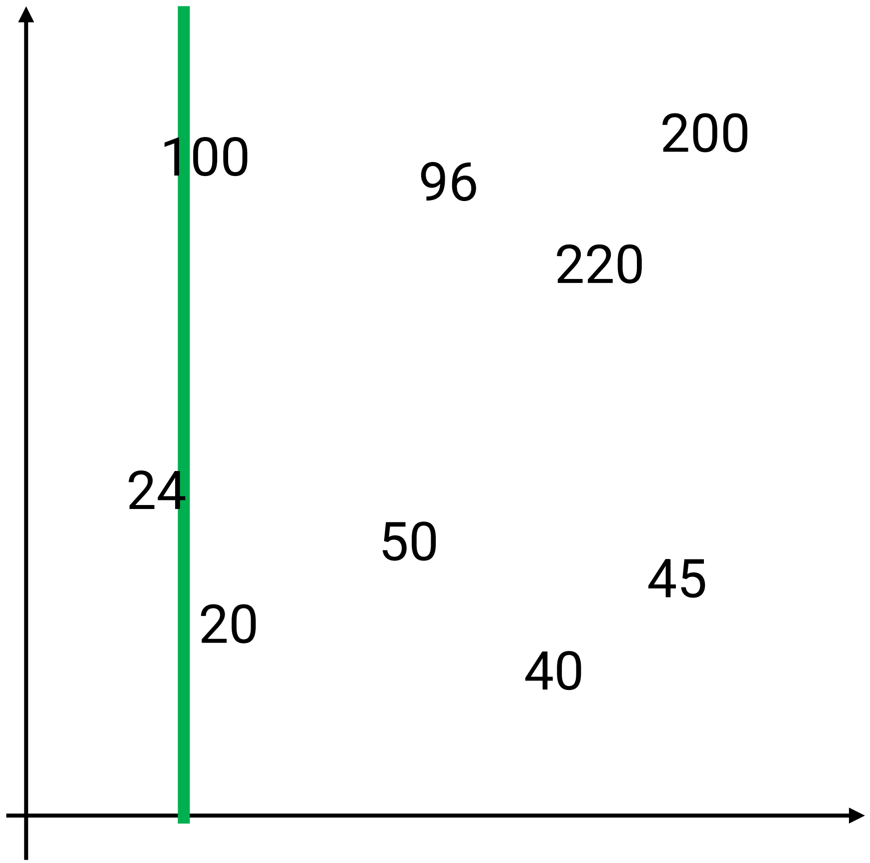
Learning Decision Trees for Regression

- At each node, decide:
  - Which feature to use
  - Which threshold to split on
- Strategy
  - Try each feature and all possible splits
  - Greedily choose split that minimizes error
    - For regression: Best prediction will be the mean on each side of the split, measure error of that relative to actual values



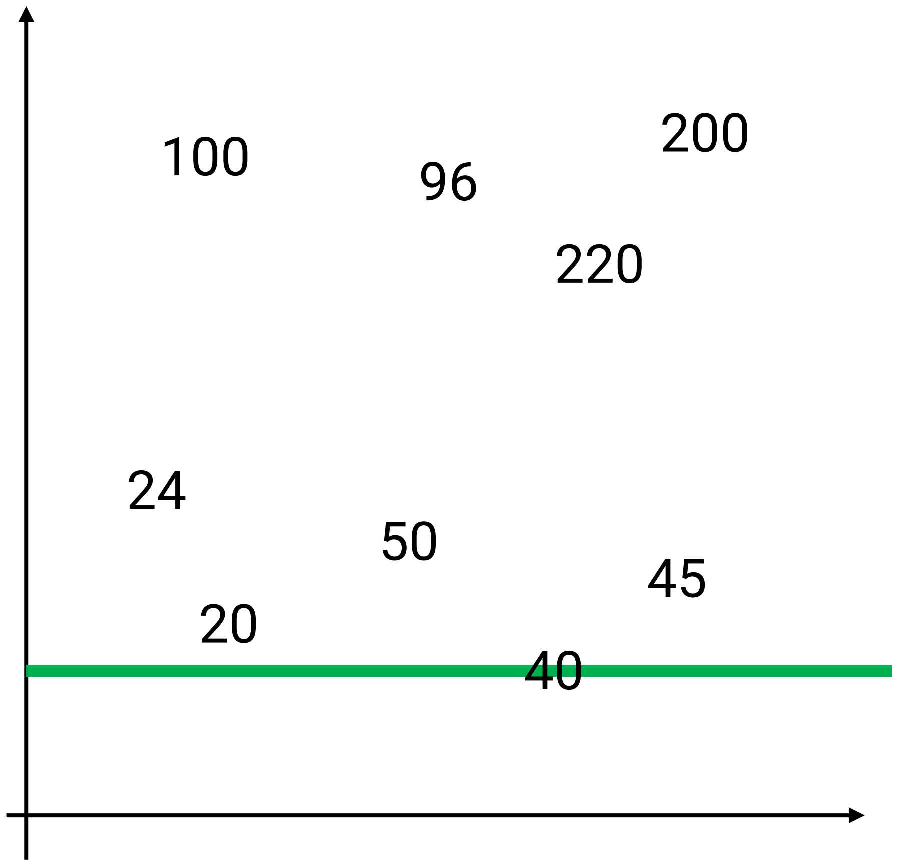
Learning Decision Trees for Regression

- At each node, decide:
  - Which feature to use
  - Which threshold to split on
- Strategy
  - Try each feature and all possible splits
  - Greedily choose split that minimizes error
    - For regression: Best prediction will be the mean on each side of the split, measure error of that relative to actual values



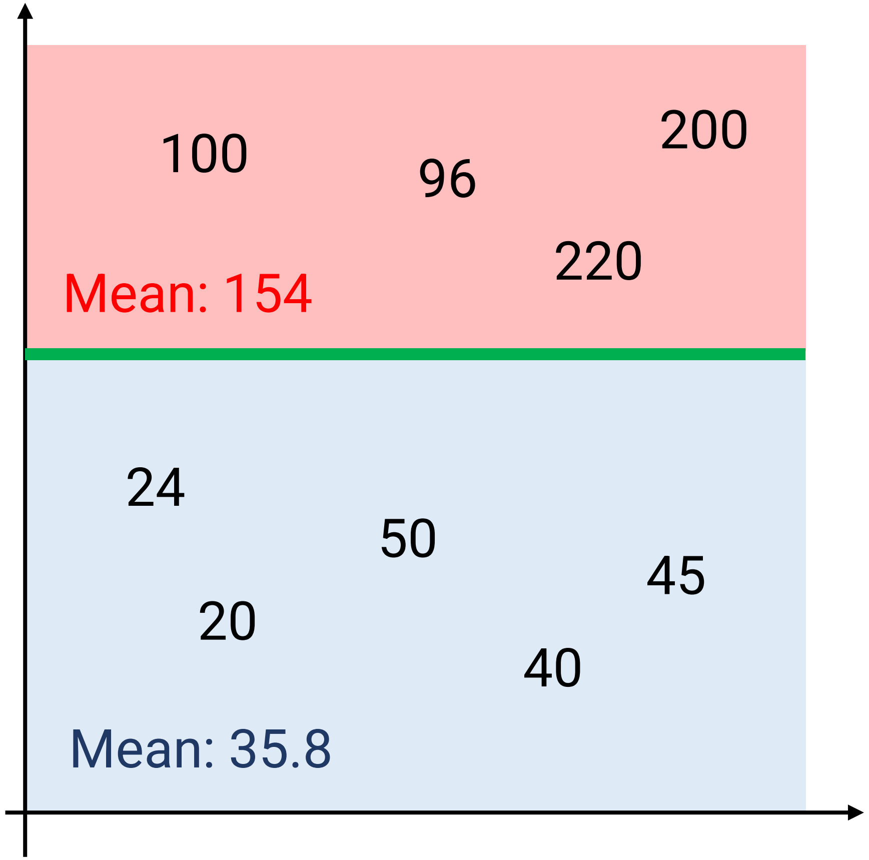
Learning Decision Trees for Regression

- At each node, decide:
  - Which feature to use
  - Which threshold to split on
- Strategy
  - Try each feature and all possible splits
  - Greedily choose split that minimizes error
    - For regression: Best prediction will be the mean on each side of the split, measure error of that relative to actual values



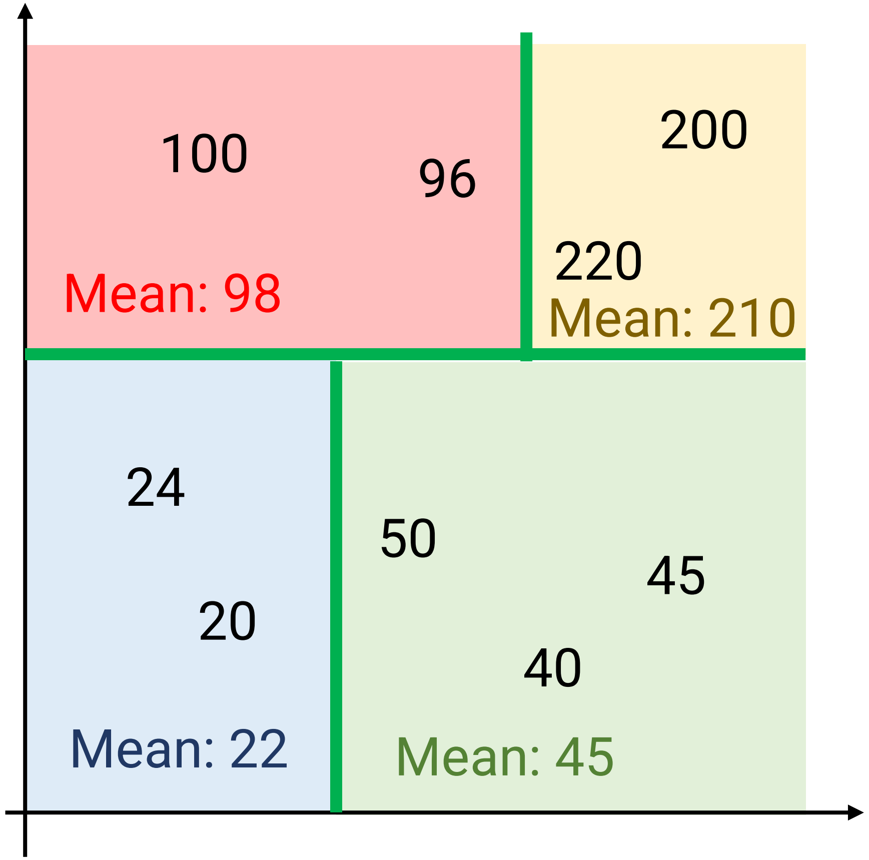
Learning Decision Trees for Regression

- At each node, decide:
  - Which feature to use
  - Which threshold to split on
- Strategy
  - Try each feature and all possible splits
  - Greedily choose split that minimizes error
    - For regression: Best prediction will be the mean on each side of the split, measure error of that relative to actual values



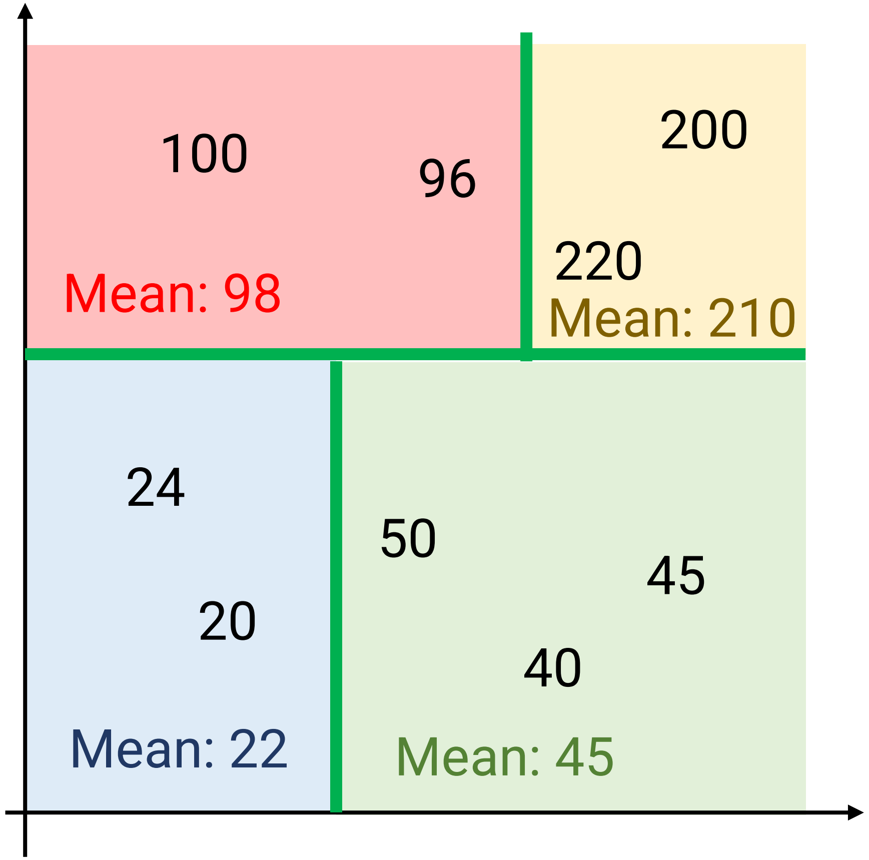
Learning Decision Trees for Regression

- At each node, decide:
  - Which feature to use
  - Which threshold to split on
- Strategy
  - Try each feature and all possible splits
  - Greedily choose split that minimizes error



Learning Decision Trees for Regression

- When do we stop splitting?
  - If we split forever to nodes of size 1, we overfit
  - Heuristic stopping criteria
    - Minimum number of examples per node
    - Maximum depth of tree
  - Can go back afterwards and "prune" tree (i.e., merge nodes back together)

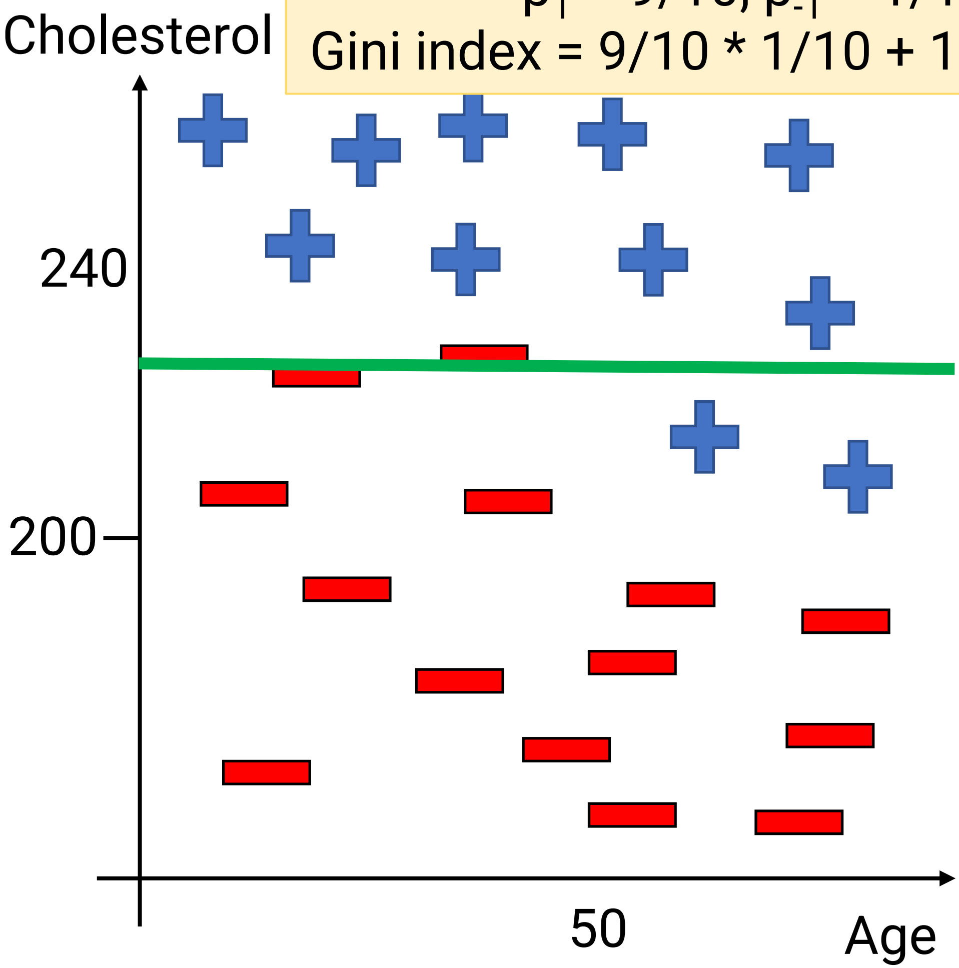


Learning decision trees for classification

- Basic idea is the same
- But how do we measure the goodness of a split?
  - Option 1: Accuracy of majority classifier
  - Option 2: Gini index $\sum_{c=1}^{C} p_c(1-p_c)$
    - p_c = Empirical probability of class c within the current node
    - Equals expected number of errors if you classify with the empirical distribution

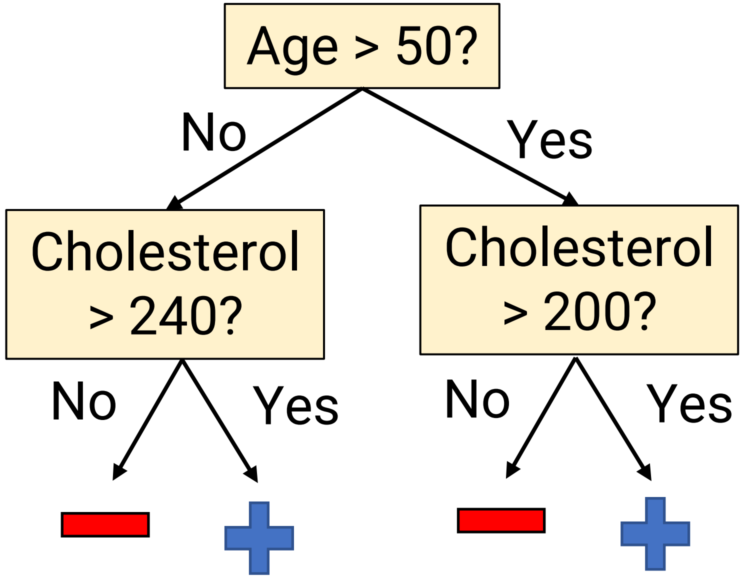
$p_1 = 9/10, p_{-1} = 1/10$

Gini index = 9/10 * 1/10 + 1/10 * 9/10

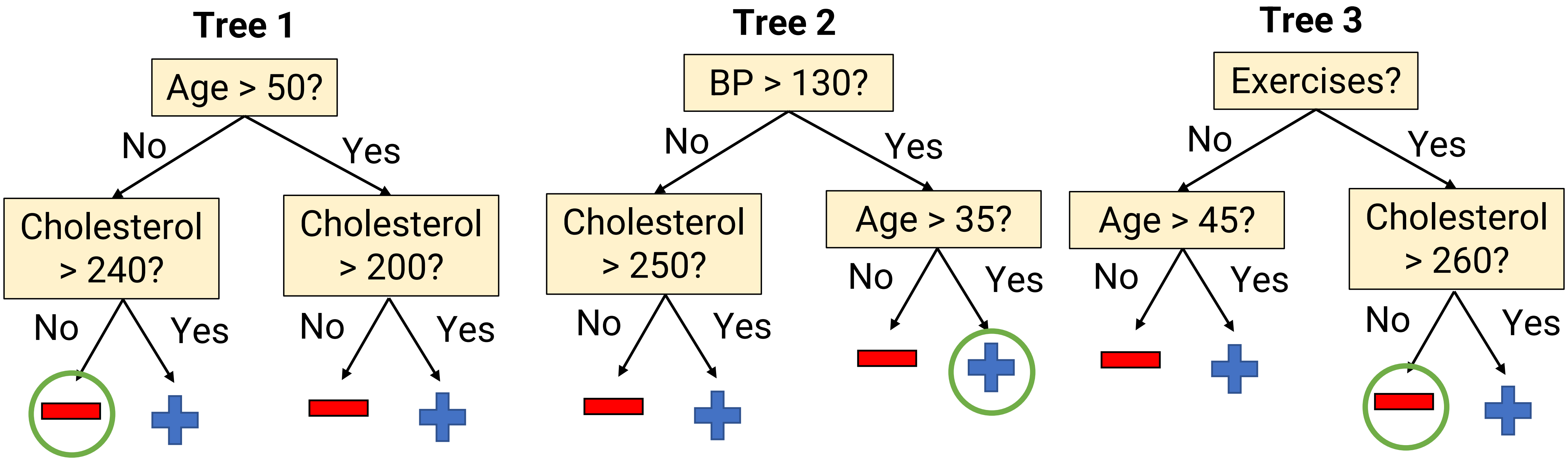


Handling Missing Features

- Some examples may be missing some features
  - E.g., For some patients, you didn't measure cholesterol level
  - What to do at a node where you split on cholesterol?
- Idea: Surrogate variables
  - During training, at each node, check which features act as **surrogates** of the feature you're using (i.e., lead to similar splits)
  - If original feature is missing, use a surrogate feature
  - E.g., If "blood pressure > 130" is correlated with "Cholesterol > 240", use blood pressure as surrogate for patients without cholesterol measurement

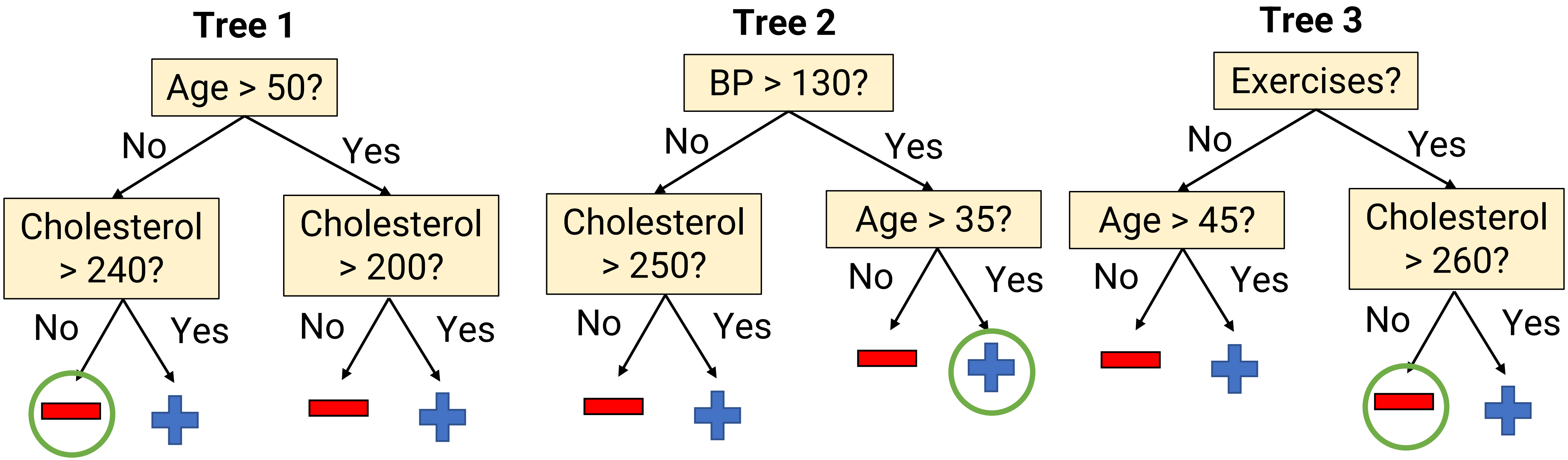


Ensembling



- Create an “ensemble” of multiple models (e.g., multiple trees)
- Make final prediction by averaging/majority vote

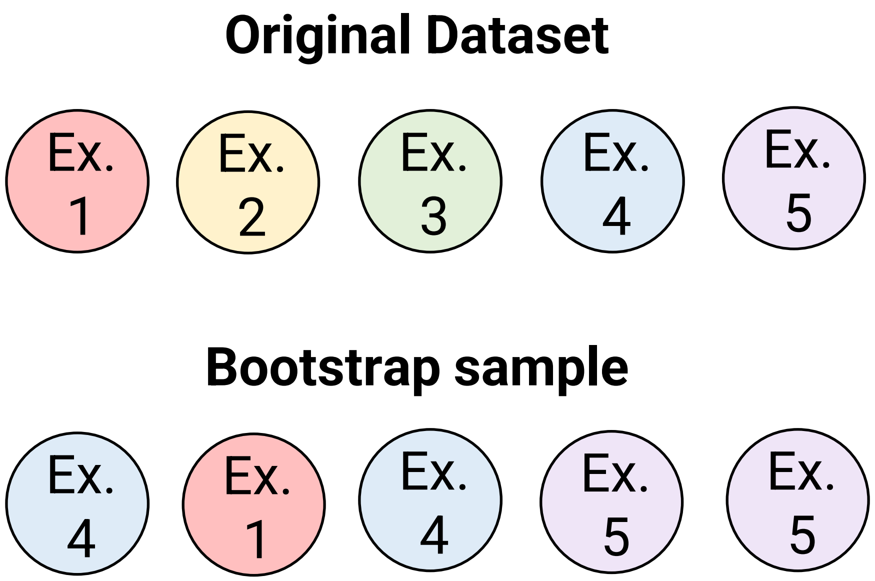
Ensembling and Trees



- An individual tree can capture complex patterns, but should not be too deep to avoid overfitting
- Thus it can only depend on a handful of features
- An ensemble of trees can leverage more features

Bagging

- How do you learn different trees from the same dataset?
- Idea: Randomly resample the dataset!
    - Given dataset with n examples, sample a new dataset of n examples **with replacement**
        - Also known as “Bootstrapping”
    - In expectation, each new dataset contains 63% of the original dataset, with some examples duplicated
    - Learn a tree on each resampled dataset

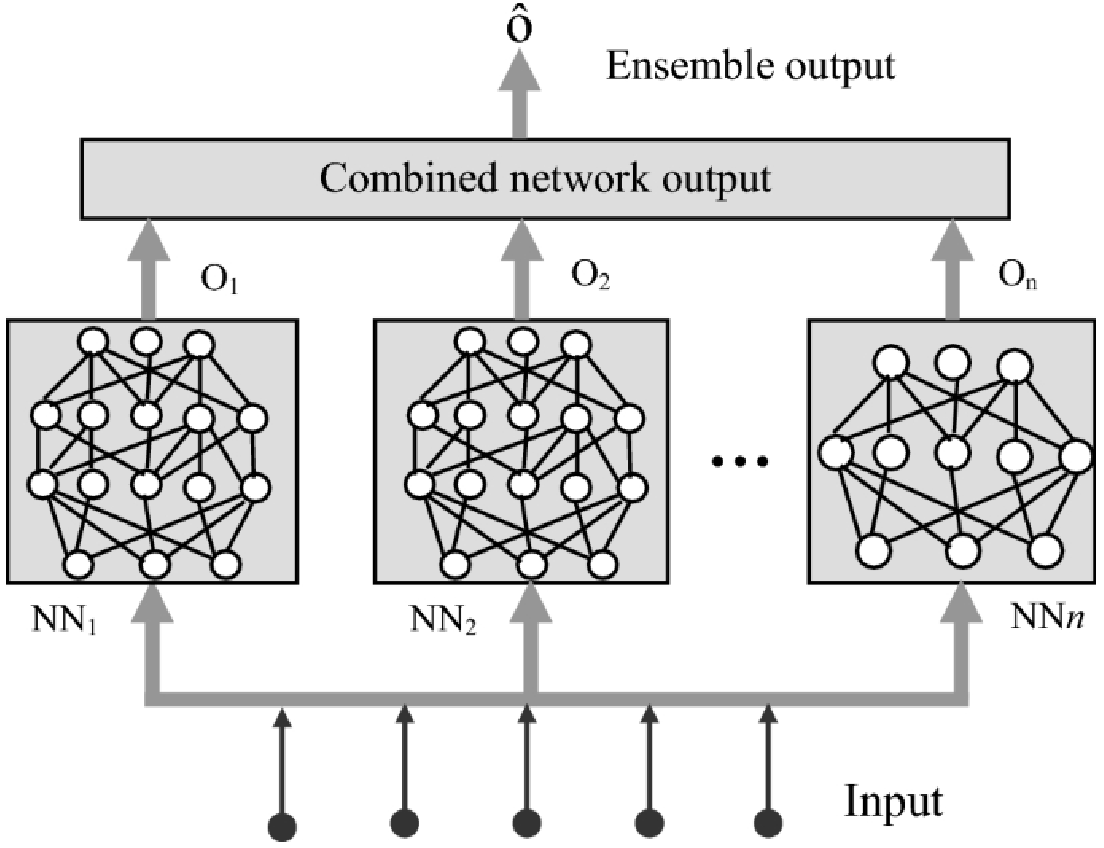


Random Forests

- Goal: Make the individual trees in the ensemble more different
  - Thus, all elements of the ensemble are complementary
- Simple strategy: Before each split, choose a random subset of features as candidates for splitting
  - Something like $\sqrt{d}$ features if *d* total features
  - Can even be randomly choosing 1 feature
- Very good general-purpose learners in practice!

Ensembles and neural networks

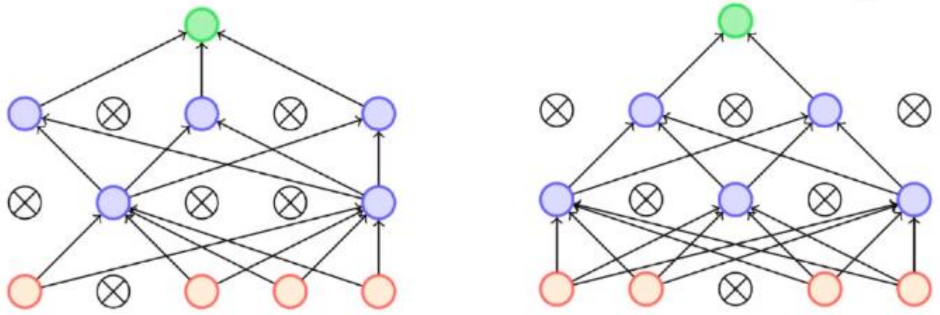
- Random Forest: Each member of ensemble differs due to random resampling of data & feature choice
- Neural Networks: Already have randomness
  - Initialization
  - Order of examples for SGD
  - Dropout
  - So, bagging is not necessary
- In practice: Very common to ensemble neural networks!
  - Compute vs. accuracy trade-off
  - Rumor: GPT-4 is an ensemble of 8 language models with 220 billion parameters each



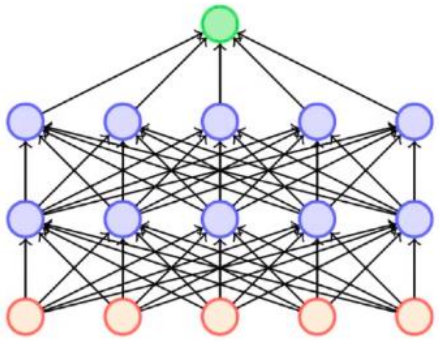
Dropout as an Ensemble

- Why does Dropout work? One explanation: **It learns a sort of ensemble**
- Training time
  - At each iteration, randomly drop out each neuron with probability *p*
  - Each iteration trains a weaker "subnetwork" instead of full network
- Test time
  - All neurons are active
  - Result is an average/ensemble of all the subnetworks
  - Note: Not exactly an ensemble in the usual sense because different subnetworks share parameters

Training time: Many "subnetworks"



Test time: Full network is average/ensemble of all subnetworks



Announcements

- Midterm exam: Thursday March 7, SLH 100
  - One 8.5” x 11” sheet of notes allowed, can be typed or handwritten
  - Please write in pen
  - Practice exams & lecture videos posted
- HW2 solutions posted
- No Section Friday
- After Spring Break: Project Midterm Reports due Tuesday, March 26