Sampling and Improving Predictive Performance

CS 584: Big Data Analytics

Sampling

- Obtain a subset of individuals from a population to estimate characteristics of the whole population

- Why is it popular?
  - Visualize and explore the data
  - Estimate population characteristics
  - Test your algorithm and find optimal parameters
  - Improve prediction on imbalanced data

Common Sampling Methods

- Simple random sampling: each sample has an equal probability of being selected

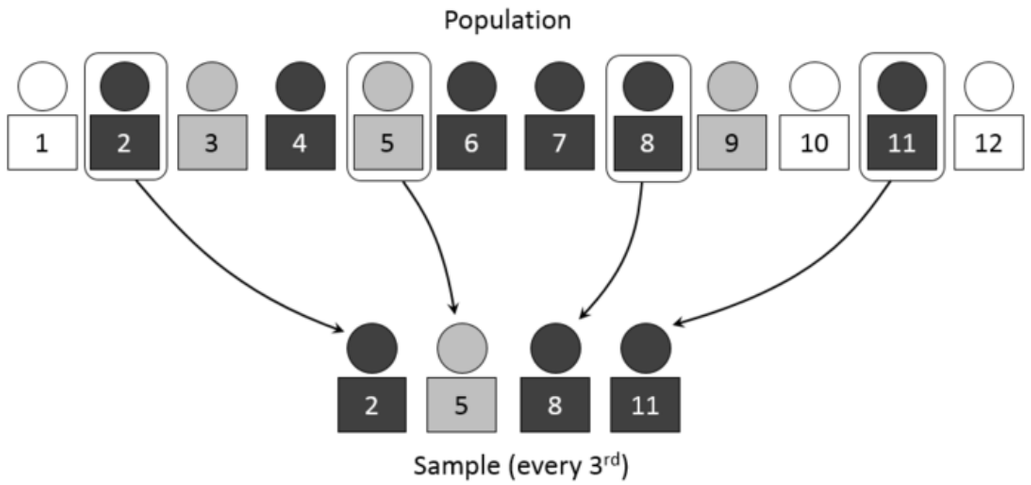
- Weighted sampling: each sample has an associated weight and the probability of choosing is proportional to the weights

- Systematic sampling: order the samples and select elements at regular intervals from the ordered list

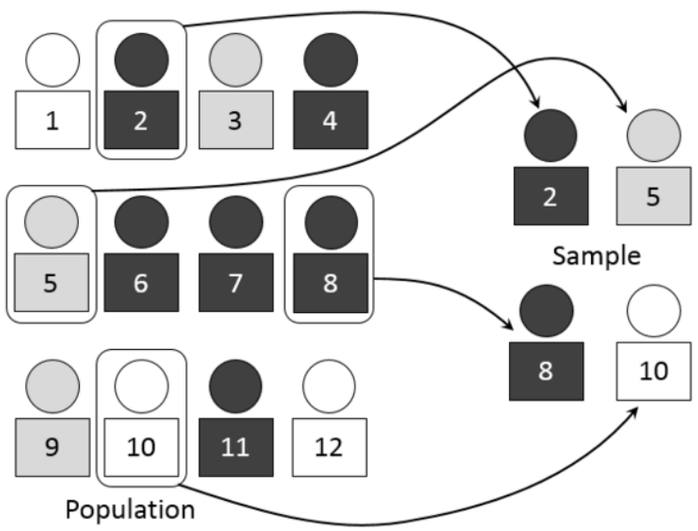
- Stratified sampling: each “strata” (group) is sampled as an independent sub-population such that the strata ratio is maintained in your sample

Common Sampling Methods (2)

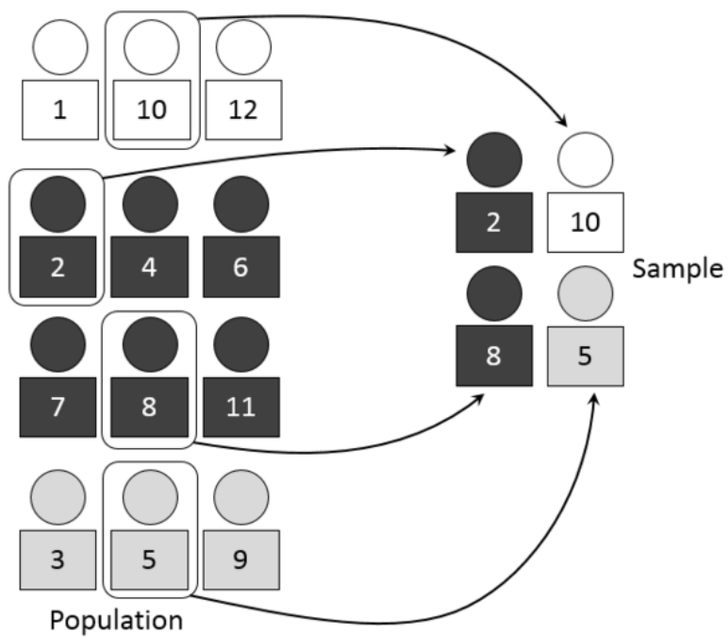
Systematic Sampling



Simple Random Sampling



Stratified Sampling



Sampling Method Properties

- Systematic sampling:
  - Great for streaming applications or no “master” list
  - Bad if population has repeating or cyclical patterns
- Simple random sampling:
  - Easy to design and defend
  - No extra information is required
- Stratified sampling
  - Can avoid taking bad samples
  - Potentially better estimates with smaller standard errors

Measure Uncertainty

- Suppose we have independent samples drawn from some population

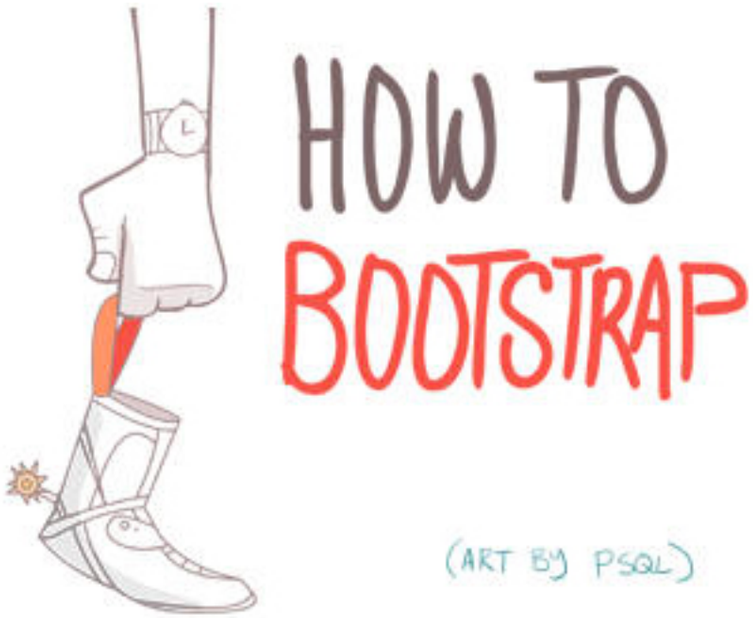
$$x_1, \cdots, x_n \sim P_\theta$$

- We estimate our parameter of interest $\hat{\theta}$
  (e.g., mean of the distribution, median value, etc.)

- We want to know the variance of our parameter or even construct approximate confidence intervals

What if we can't make usual assumptions such as normality?

Bootstrap Method



Metaphor for a “self-sustaining process that proceeds without external help”

http://www.gmw.rug.nl/~huisman/sgs/2012_10_25_Bootstrap.pdf

Bootstrapping (Efron, 1979)

- Fundamental resampling tool in statistics
- General and most widely used tool to estimate measures of uncertainty associated with a given statistical model (e.g., confidence intervals, bias, variance, etc.)
- Resampling technique with replacement
  - "The population is to the sample as the sample is to the bootstrap samples"
- Distribution-independent or non-parametric

Bootstrap Method

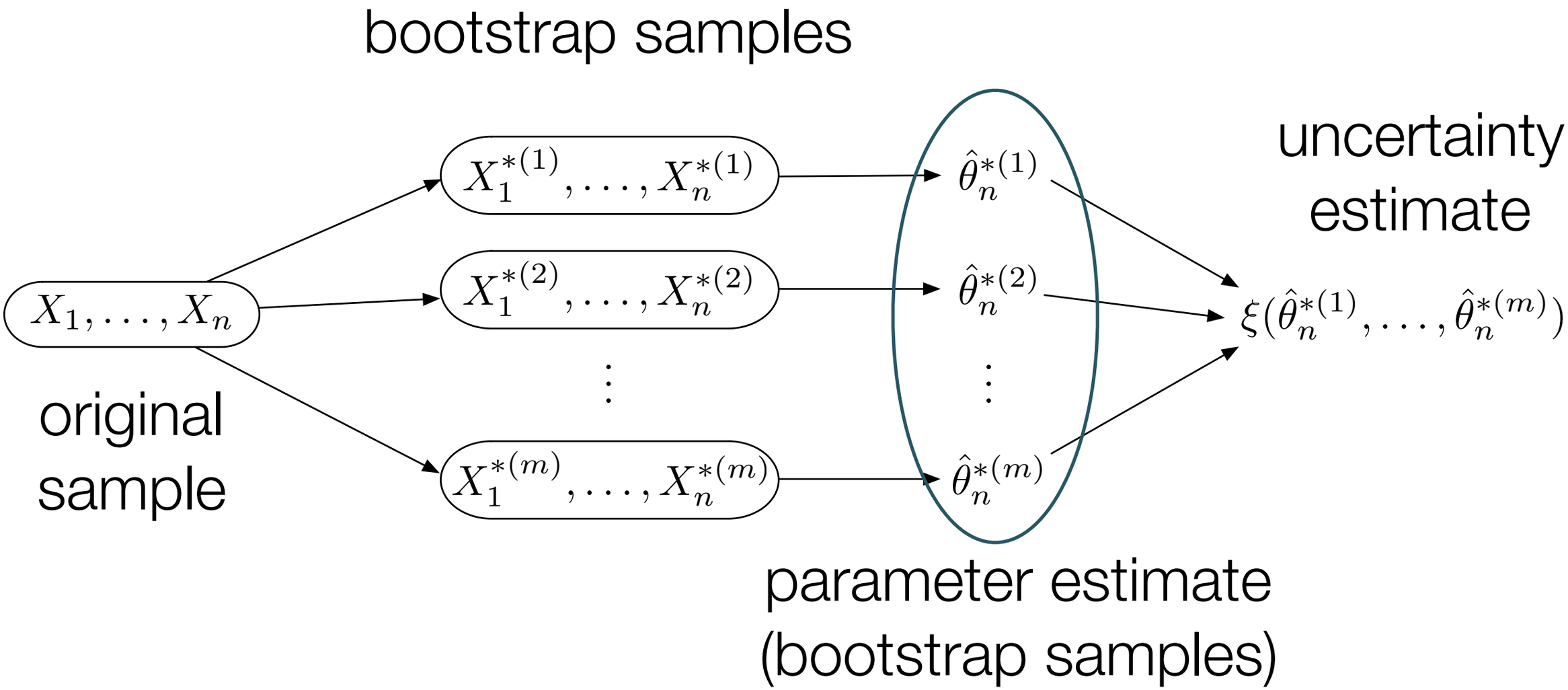
Given a sample of size n

- Draw B samples of size n with replacement from the sample (bootstrap samples)

- Compute for each bootstrap sample the statistic of interest (e.g., the mean)

- Estimate the sample distribution of the statistic method by the bootstrap sample distribution

Bootstrap Method



Bootstrap: Measuring Uncertainty

- Estimating standard errors

$$\mathrm{SE}(\hat{\theta}) = \sqrt{\frac{1}{B}\sum_{b=1}^{B}(\theta_b - \frac{1}{B}\sum_{r=1}^{B}\theta_r)^2}$$

- Estimating bias

$$E(\hat{\theta}) \approx \frac{1}{B}\sum_{b=1}^{B}(\theta_b - \hat{\theta})$$

- Estimating confidence

$$\mathbb{P}(2\hat{\theta} - q_{1-\alpha/2} \leq \theta \leq 2\hat{\theta} - q_{\alpha/2}) = 1 - \alpha$$

Bootstrap Properties

- Simple and straightforward to derive estimates of standard errors and confidence intervals for complex estimators

- Control and check the stability of the results

- Asymptotically consistent (under certain conditions)

- Expected number of distinct points in a bootstrap sample is ~0.632n

Improving Classification Performance

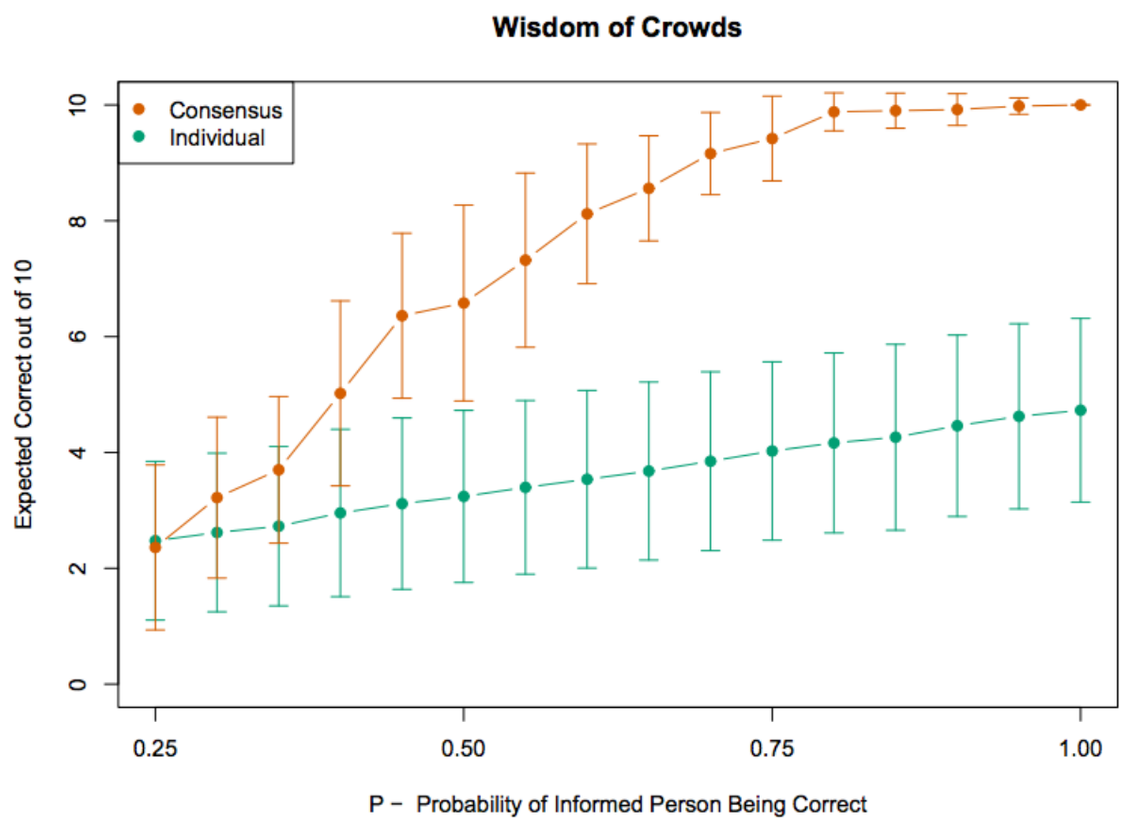
- Ensembles
  - Bagging
  - Boosting
- Oversampling / Undersampling

Motivation for Ensembles

- Different learners have different “inductive bias”
  - Generalize differently from the same training set
- Different properties of models
  - Local vs global
  - Computation time / memory
  - Susceptibility to outliers
- Hope is to use multiple models for better accuracy and better reliability

Wisdom of Crowds

- Concept popularized outside of statistics
- Idea that collection of knowledge of an independent group of people can exceed knowledge of one person individually



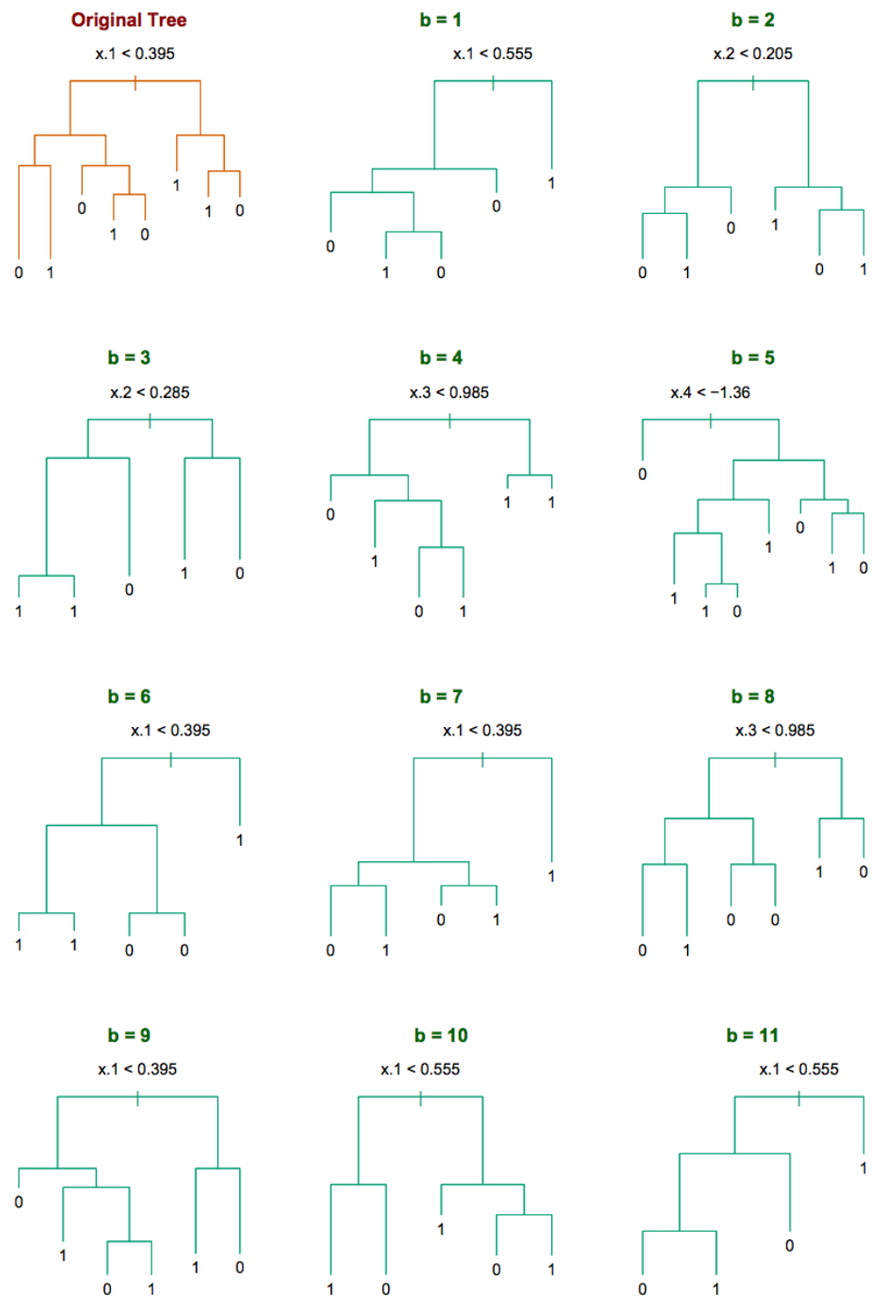
Elements of Statistical Learning Figure 8.11

Bagging (Breiman, 1992)

- Bootstrap Aggregating: variance reduction technique
  - Create bootstrap replicates
  - Fits model to each replicate
    - Combines predictions via averaging or voting
- Stabilizes unstable models (e.g., decision trees, neural nets)
- Easily parallelizable

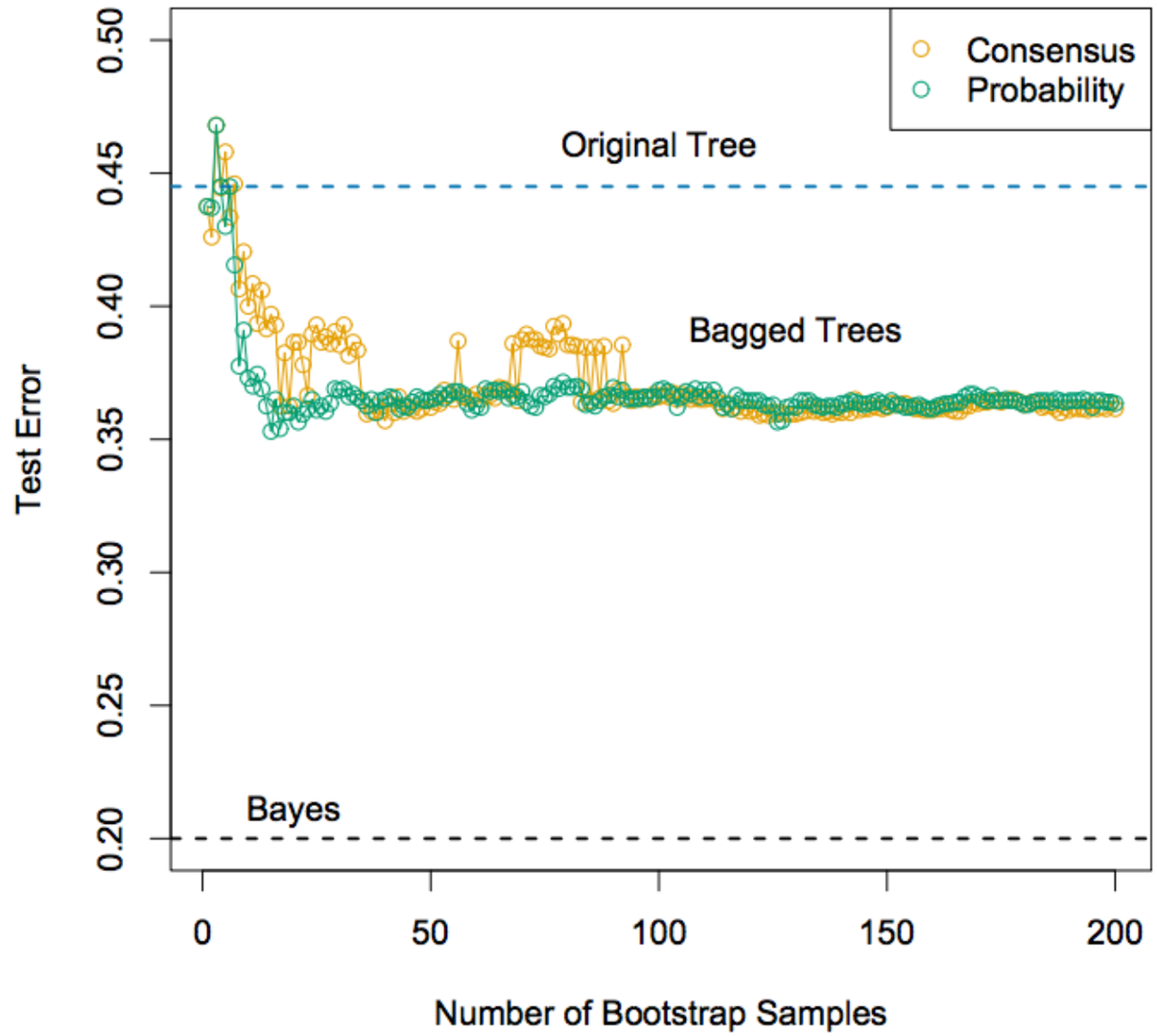
Example: Bagging

Simulated data with n=30, two classes, and 5 features



Elements of Statistical Learning Figure 8.9

Example: Bagging



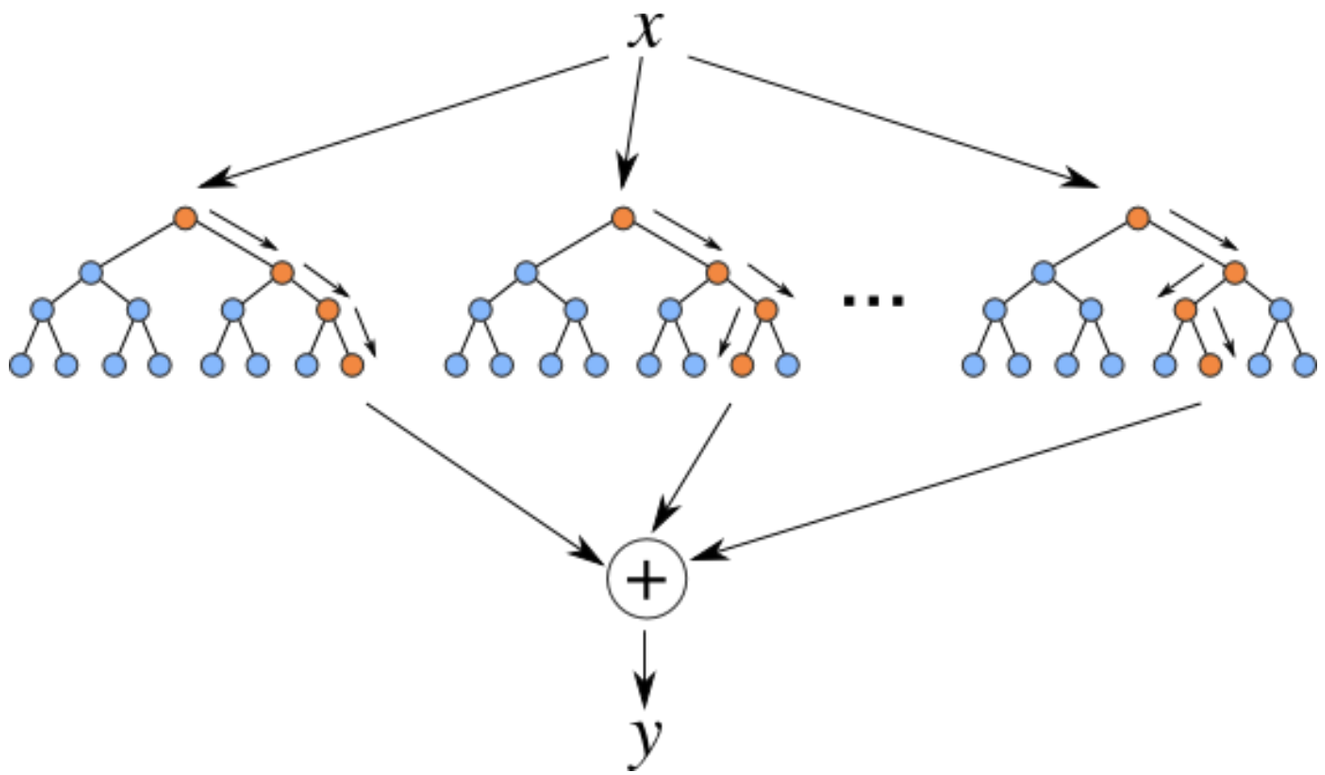
Bagging helps decrease the misclassification rate of the classifier (evaluated on large independent test set)

Elements of Statistical Learning Figure 8.10

Random Forests (Breiman, 2001)

Bagged classifier (ensemble) using decision trees

- Each split only considers a small group of features
- Tree is grown to maximum size without pruning
- Final predictions obtained by aggregating over ensemble



Random Forests Properties

- State of the art method, generally one of the most accurate general-purpose learners available

- Handles a large number of input variables without overfitting

- Easy to train and tune

- Reduces correlation amongst bagged trees by considering only a subset of variables at each split

Random Forest Properties (2)

- Easily parallelized by training
- Robust to errors and outliers
- Can model non-linear boundaries
- Gives variable importance and out of bag error rates
- (Con) Loss of interpretability
- (Con) Difficult to analyze as an algorithm and mathematical properties still largely unknown

Why Does Bagging Work?

- Suppose that for a given input x in a binary classification problem where we have B independent classifiers and each as a misclassification rate e=0.4

- Assume without loss of generality that the true class is 1

$$P(\hat{f}_b(x) = -1) = 0.4$$

- Our bagged classifier

$$\hat{f}(x) = \text{argmax}_{k=-1,1} \sum_b 1\{\hat{f}_b(x) = k\}$$

Why Does Bagging Work? (2)

- Let B_{-1} be the number of votes for class -1, a binomial variable with p=0.4

- Misclassification rate of the bagged classifier:

$$B_{-1} \sim \text{Binom}(B, 0.4)$$

$$P(\hat{f}(x) = -1) = P(B_{-1} \geq B/2)$$

- As B grows larger, our classifier should be perfect in theory

  - This is not the case as this assumes independence and our classifiers are not independent

Bagging Disadvantages

- If the misclassification rate is high, the bagged classifier is perfectly inaccurate as B approaches infinity (degradation in predictive accuracy)

- Loss of interpretability: if the original classifier model was interpretable, final bagged classifier will not be so easy to understand

- Computational complexity: multiply the work of a single classifier by B

- Limited model space: bagging can still not easily represent certain boundaries

Boosting

- Sequentially fit models (weak learners) with later models seeing more of the samples mispredicted by earlier ones (reweighting)

- Combined using weighted average where the weights are determined by the accuracy

- Reduces both bias and variance

- Slow to overfit

Boosting vs Bagging

- Boosting fits the entire training set whereas bagging is just bootstrap samples

- Boosting adaptively adjusts the weight of the observations to encourage better predictions for misclassified points (bagging is equal weights for all observations)

- Boosting tends to have greater accuracy compared to bagging but also risks overfitting

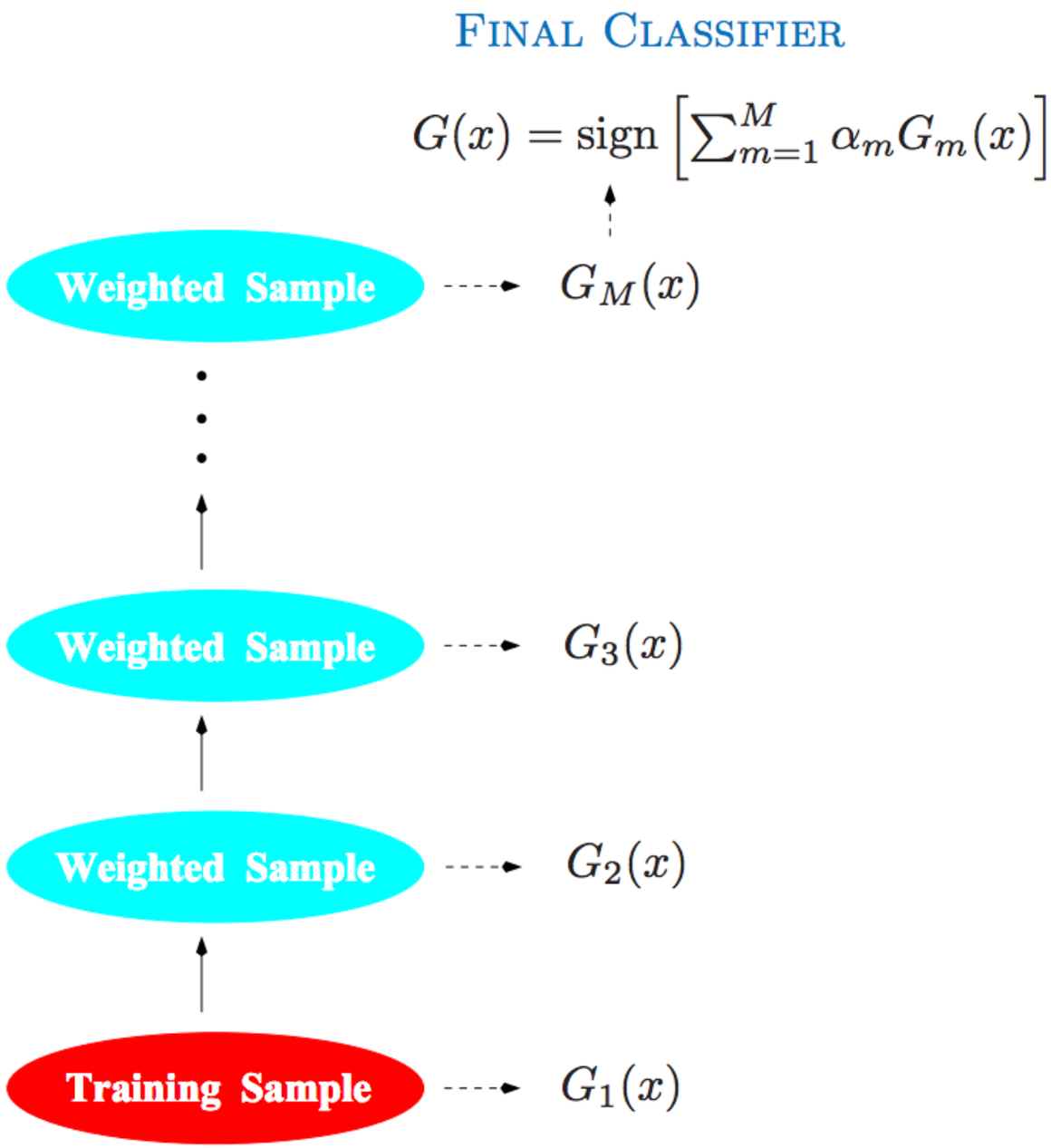
AdaBoost (Fruend & Schapire, 1997)

- Most popular boosting algorithm

- Consider a two-class problem, where the output variable is coded as {+1, -1}

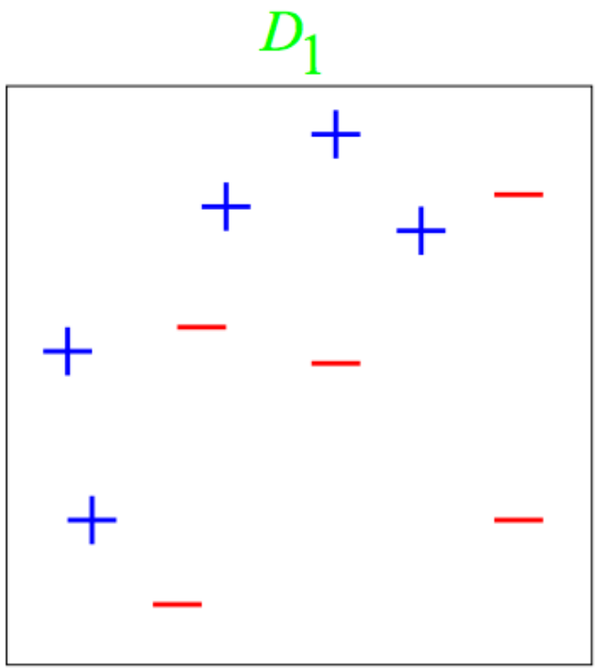
Initialize $w_i = 1/n$
for $b = 1, \cdots, B$ **do**
  Fit model $\hat{f}_b$ to the training data with weights $w_1, \cdots w_n$
  Compute weighted error $e_b = \sum_i w_i 1\{y_i \neq \hat{f}_b(x_i)\} / \sum_i w_i$
  $\alpha_b = \log((1 - e_b)/e_b)$
  Update weights $w_i = w_i \exp(\alpha_b 1\{y_i \neq \hat{f}_b(x_i)\})$
end
$\hat{f}_{\text{boost}}(x) = \text{sign}\left(\sum_b \alpha_b \hat{f}_b(x)\right)$

AdaBoost



Elements of Statistical Learning Figure 10.1

Example: Toy (Simulated) Data

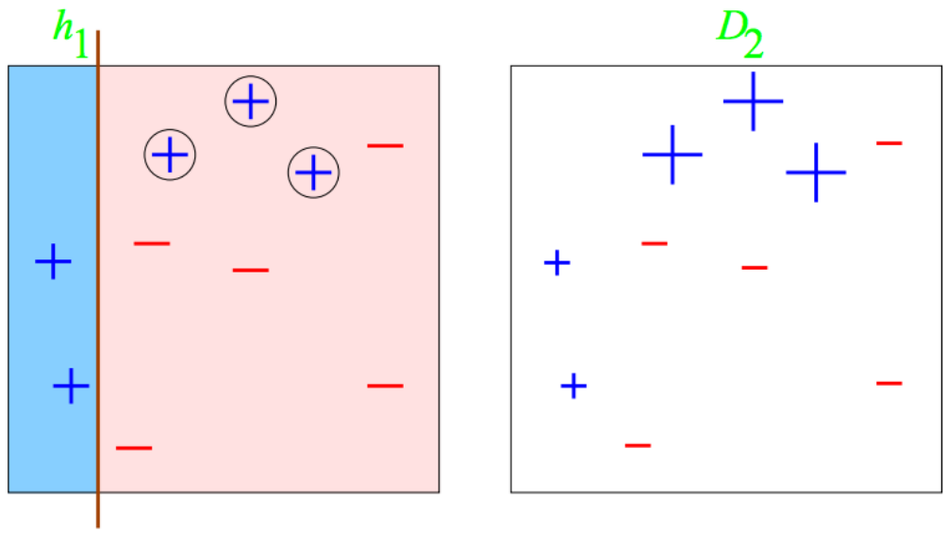


weak classifier: single horizontal or vertical half-plane

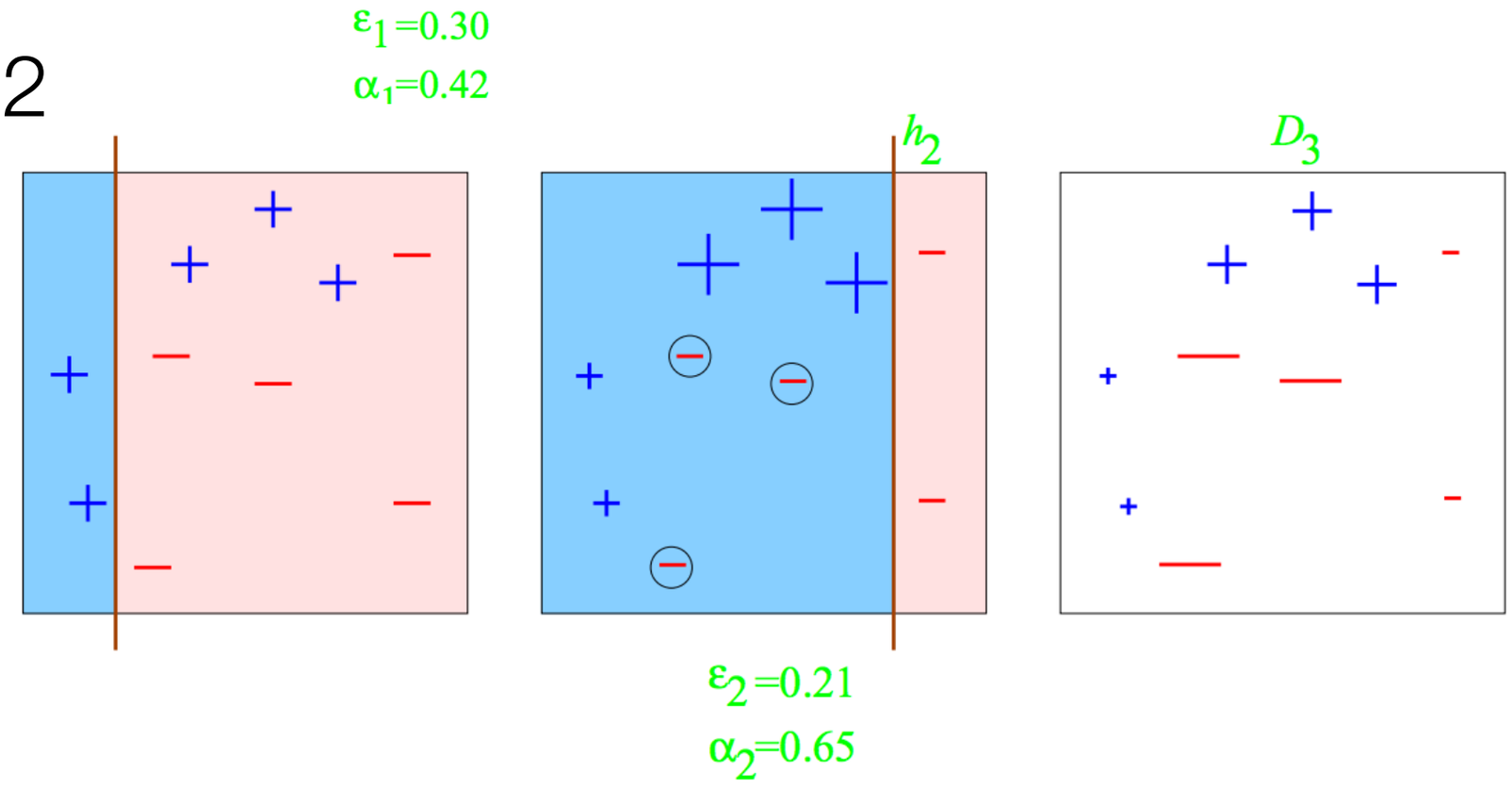
http://media.nips.cc/Conferences/2007/Tutorials/Slides/schapire-NIPS-07-tutorial.pdf

Example: Toy (Simulated) Data (2)

Round 1

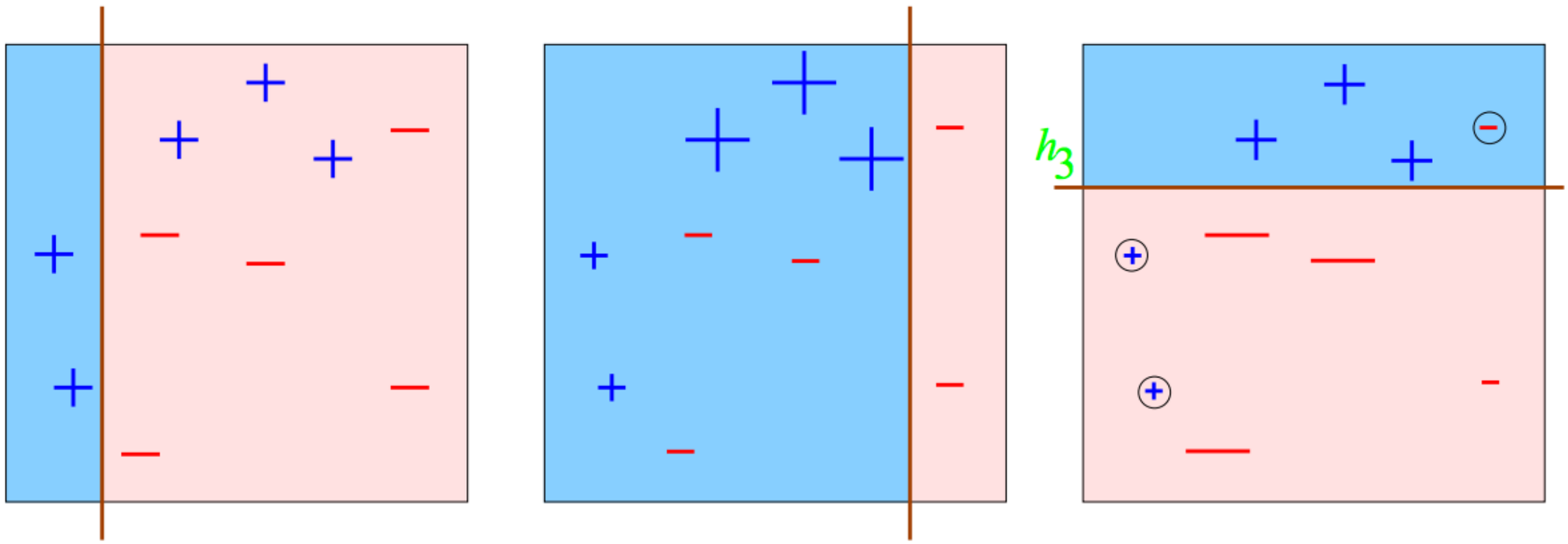


Round 2



Example: Toy (Simulated) Data (3)

Round 3

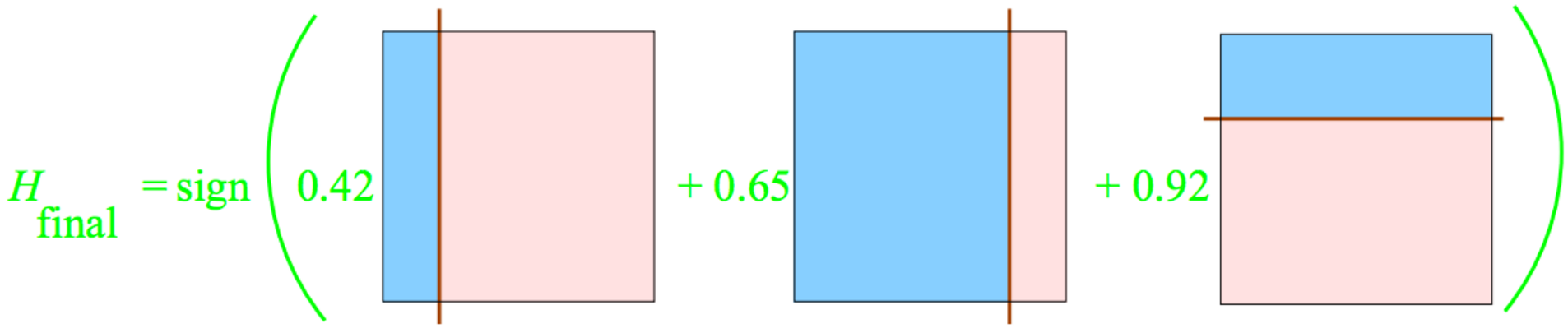


$\varepsilon_3 = 0.14$

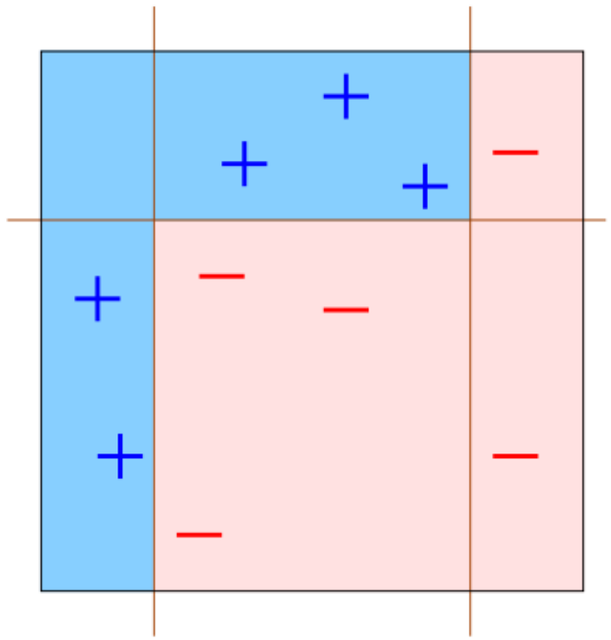
$\alpha_3 = 0.92$

Example: Toy (Simulated) Data (4)

Final Classifier



=

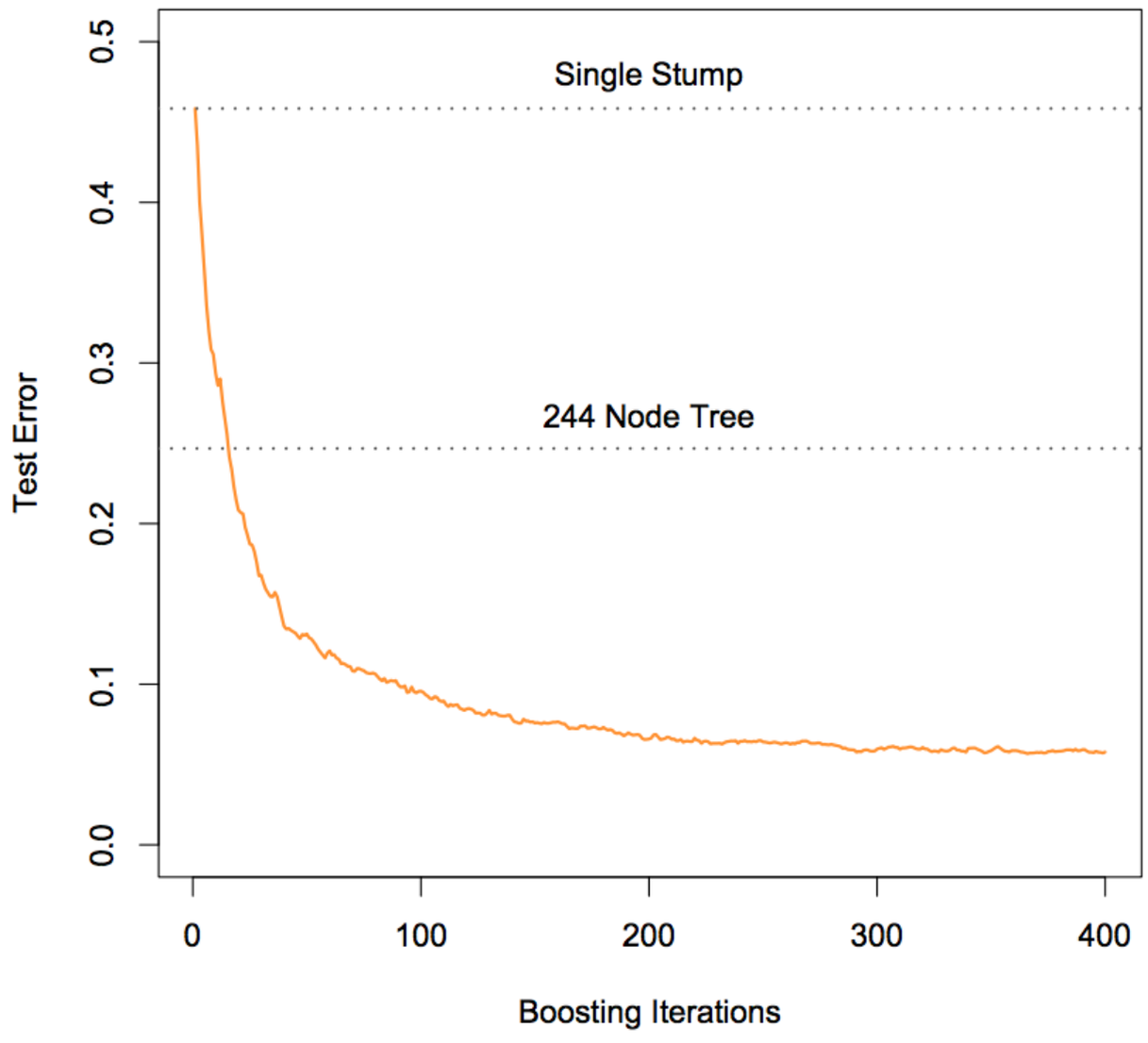


Example: Boosting Stumps

- Simulated data with 1000 points draw from known model

- Classification tree with one split (two leaves)

- Misclassification rate of 45.8% for single tree



Elements of Statistical Learning Figure 10.2

AdaBoost Strengths

- Single parameter to tune (number of rounds)
- Fast, simple and easy to program
- Theoretical guarantees on the training error and test error
- Only need base learning that performs better than random (weak learner)
- Identify outliers (i.e., examples that are mislabeled, inherently ambiguous, or hard to categorize)

AdaBoost Disadvantages

- Boosting can be susceptible to noise
- Actual performance depends on the data and base learner
- Number of outliers can hurt the performance due to emphasis placed on hard examples
  - Resulted in different variations such as Gentle AdaBoost, BrownBoost

Why Does Boosting Work?

- Intuition is simple: misclassified samples are weighed to get properly classified in future iterations

- Connection between boosting and forward stepwise modeling, so each additional model is improving the accuracy of the model

- Many different ways to extend boosting based on different losses and shrinkage (adding a small multiple of a tree) leads boosting to be a general, powerful tool

Boosting Disadvantages

- Loss of interpretability: if the original classifier model was interpretable, final boosted classifier will not be so easy to understand

- Serial computation: each classifier must be built sequentially to get the proper weights for the instances

- Computation can be difficult depending on the boosting algorithm itself (AdaBoost is fairly straightforward)

Sampling for Imbalanced Classes

- Some real-world data sets are dominated by “normal” examples with small percentage of “abnormal” examples

- Standard learners are biased towards majority class

- Main idea: Modify the distribution of events so the rare class is well-represented in the training sample

  - Simple way of biasing the generalization process

Methods for Sampling

- Undersampling
  - Abundance of majority class examples so we can take any random sample
  - Downside: some of the useful instances may not be chosen for training
- Oversampling
  - Replicate events of the minority class
  - Downside: Overfitting for noisy data due to replication

SMOTE: Synthetic Minority Oversampling Technique (Chawla, Hall, & Kegelmeyer 2002)

- Informed oversampling of the minority class with random under sampling of majority class

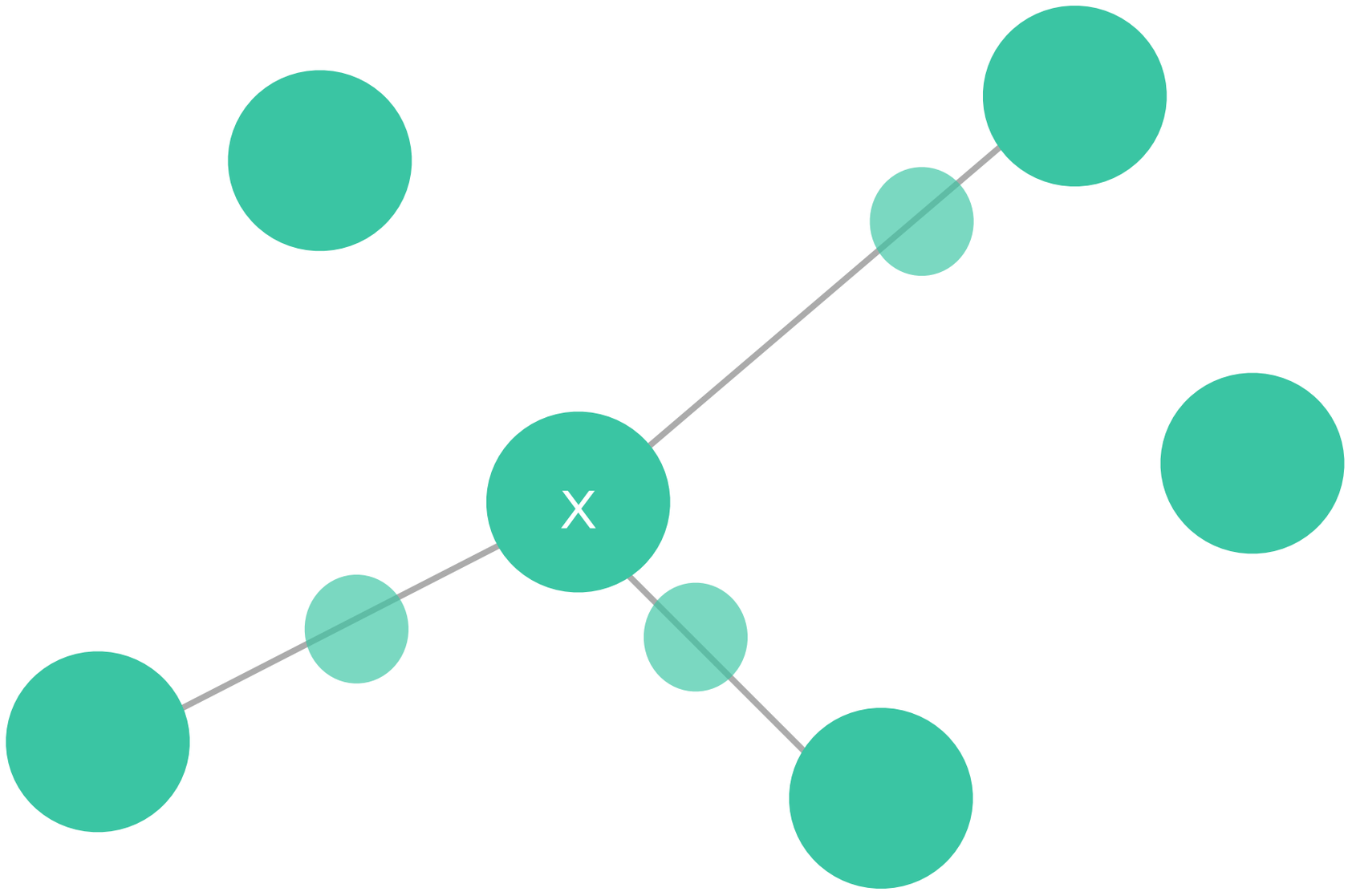
- Informed oversampling procedure to generalize decision region for minority class

- Find its k-nearest minority neighbors

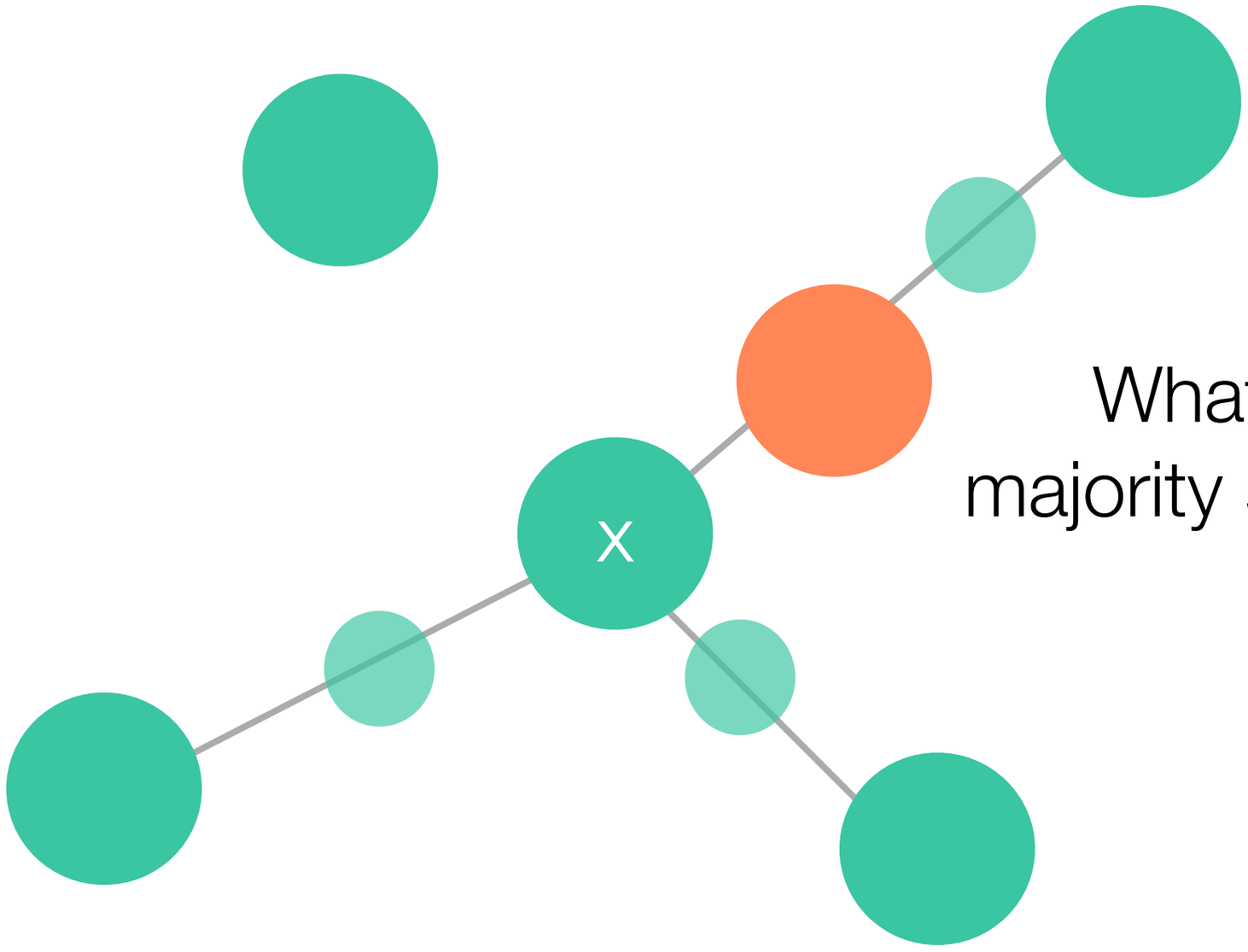
- Randomly select j of these neighbors

- Randomly generate synthetic samples along the liens joining the minority sample and its j selected neighbors

SMOTE: Informed Undersampling



SMOTE: Informed Undersampling



What if there is a majority sample nearby?

SMOTE: Shortcomings

- Overgeneralization of minority class: blindly generalizes minority area without regards to the majority class
    - Problematic for highly skewed class distributions which results in greater chance of class mixture
- Lack of flexibility: number of synthetic samples fixed in advance
- Computational cost is higher than undersampling or oversampling