CS 5806 Machine Learning II

Lecture 12 - Ensemble Learning 1
October 2nd, 2023
Hoda Eldardiry

Recommended Reading [1] Ch. 11, [5] Ch. 15-16, [6] Ch. 14, [7] Sec. 18.10, [8] Sec. 16.2.5

Ensemble Learning

- **Definition**
  — A special paradigm for learning
  — Instead of generating a single hypothesis based on a single algorithm
  — Combine multiple algorithms & multiple hypotheses to obtain better results

- **Examples**
  — Bagging
  — Boosting

Supervised Learning

- Multiple approaches
- Multiple options

Which Technique to Pick?

Which method will produce better results?
— We don’t know
— It’s a process of trial & error
— Theory: some techniques do better in some situations
— However: theory does not hold in practice

Do we need to pick a technique?

Can we just combine techniques?
— Would combining techniques do better?

No Perfect Technique

- In practice
  — Different learning techniques yield different hypotheses
  — None of those hypotheses is perfect

- Typically
  — One technique gives you a hypothesis that is good for one type of data
  — Another technique gives you a different hypothesis that is better for a different type of data

- In practice
  — Your data may include both types of data
  => You don’t have a perfect hypothesis
  => No hypothesis works best for all types of data

- **Can we combine several imperfect hypotheses into a better hypothesis?**

Ensemble Learning

- **Similar to things we often do as a society ..**
  — Elections combine voters' choices to pick a good candidate
  — Committees combine experts' opinion to make better decisions

- **Ensemble Learning**
  — Algorithms are voters of predictions that we combine
  — Different algorithms better suited for different types of data
  — When combined obtain a better overall solution

- **Intuition**

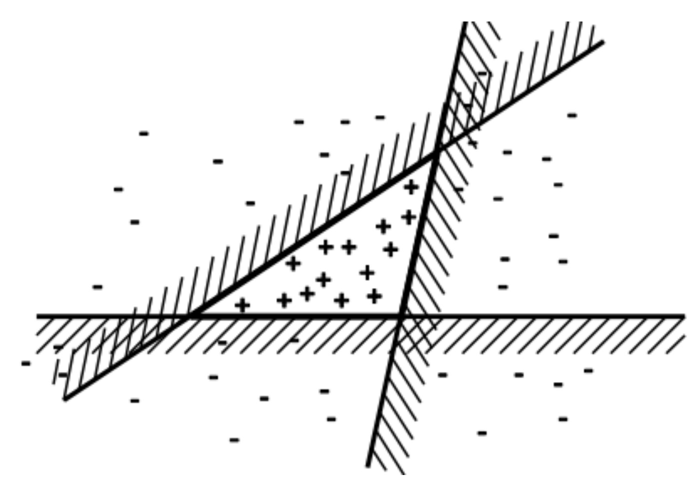
	Individual	Majority
Mistakes	Likely	Less likely
Knowledge	Partial	Complementary
Preference	Individual-specific	More widely suitable for society
Decision		More robust

Ensemble Learning

- **Definition**
  — Method to select & combine an ensemble of hypotheses into a (hopefully) better hypothesis

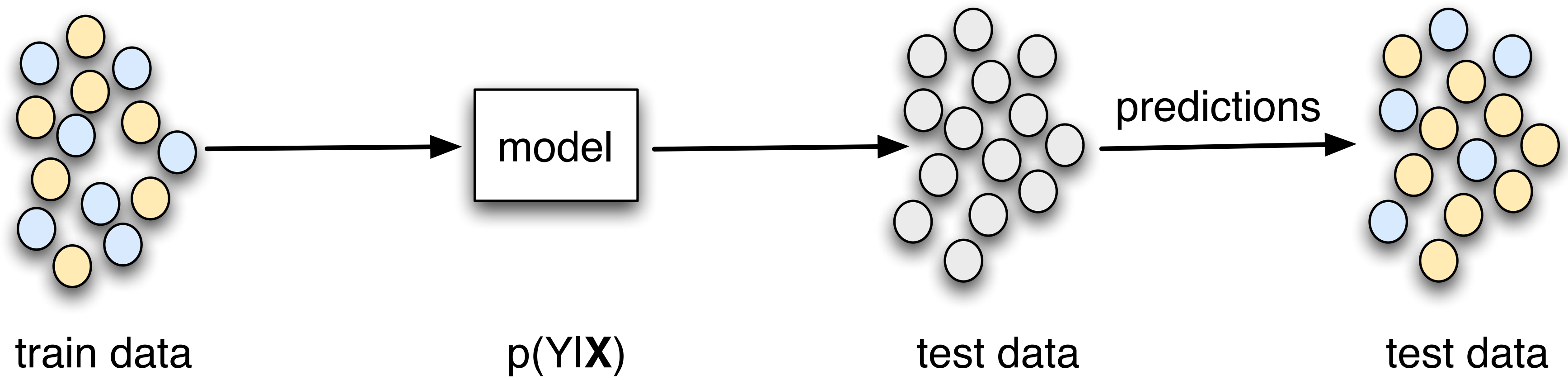
- **Properties**
  — Not always better
  — With right type of algorithm
  — Generally obtain better results

- **Example**
  — Combine multiple linear separators
  — Improve classification on data that is not linearly separable



- **Intuition**
  — Enlarge hypothesis space

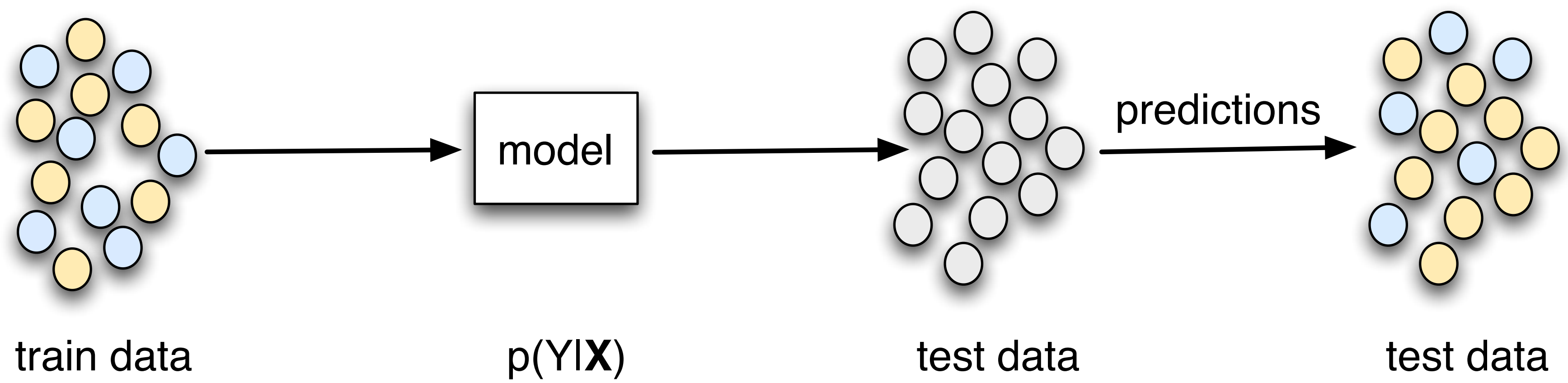
Supervised Classification



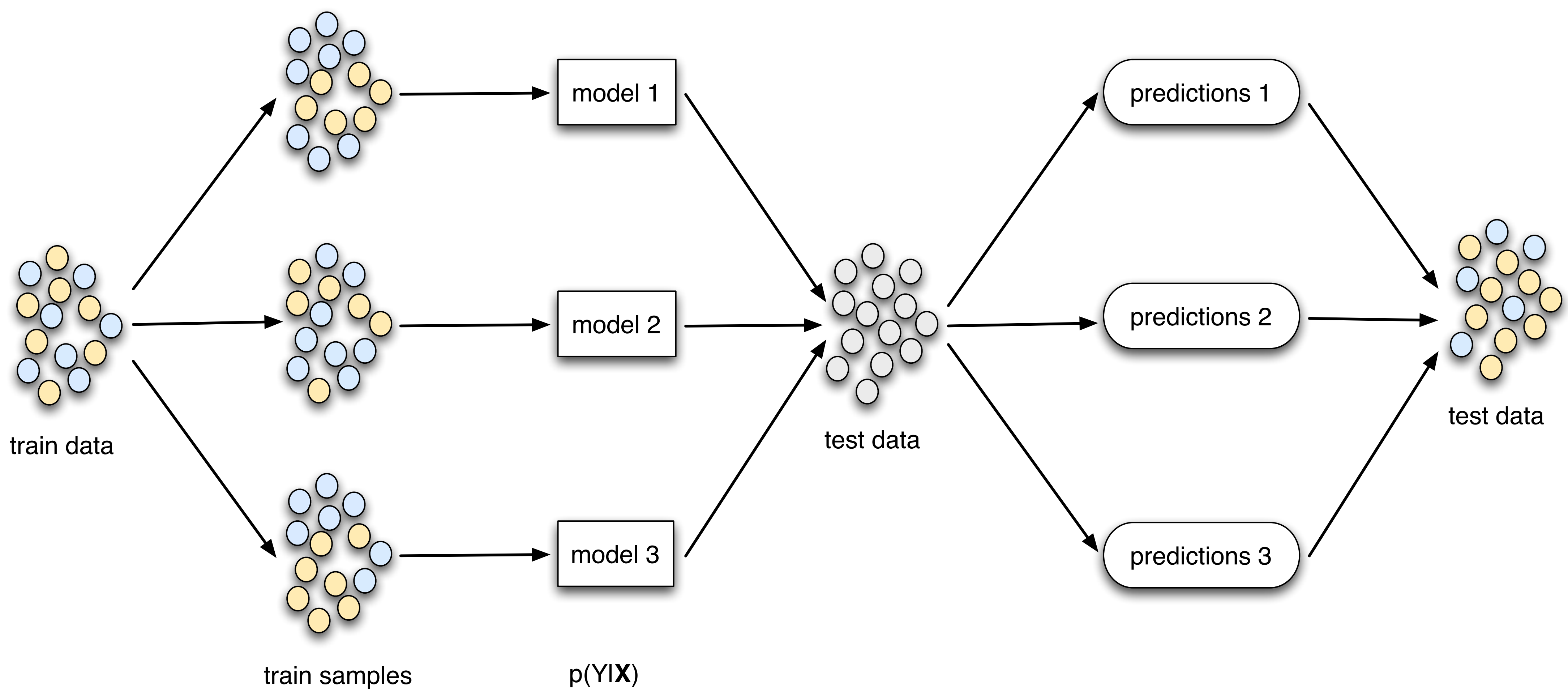
X: attributes

Y: class label

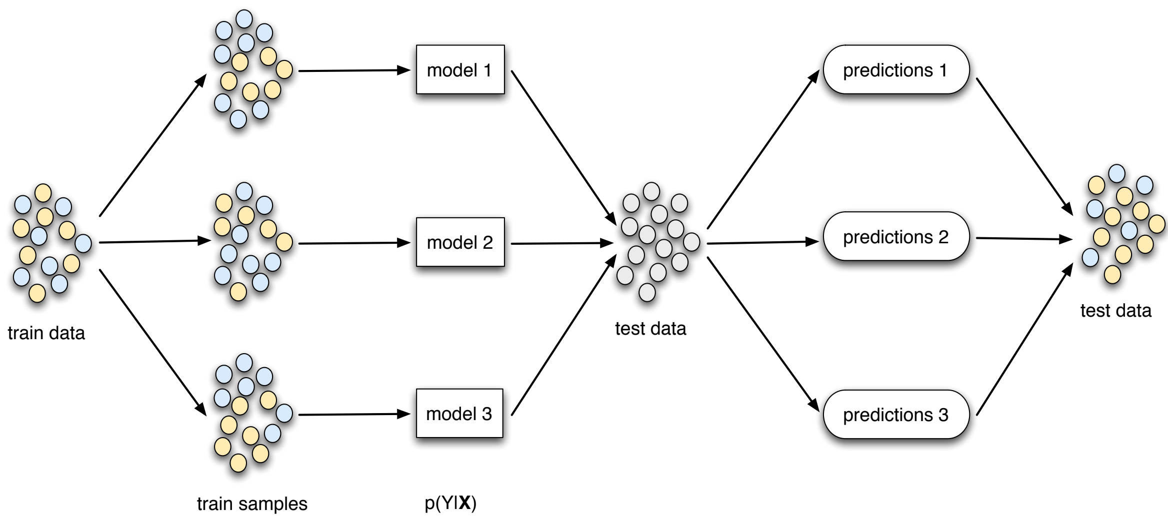
Ensemble Learning



X: attributes
Y: class label

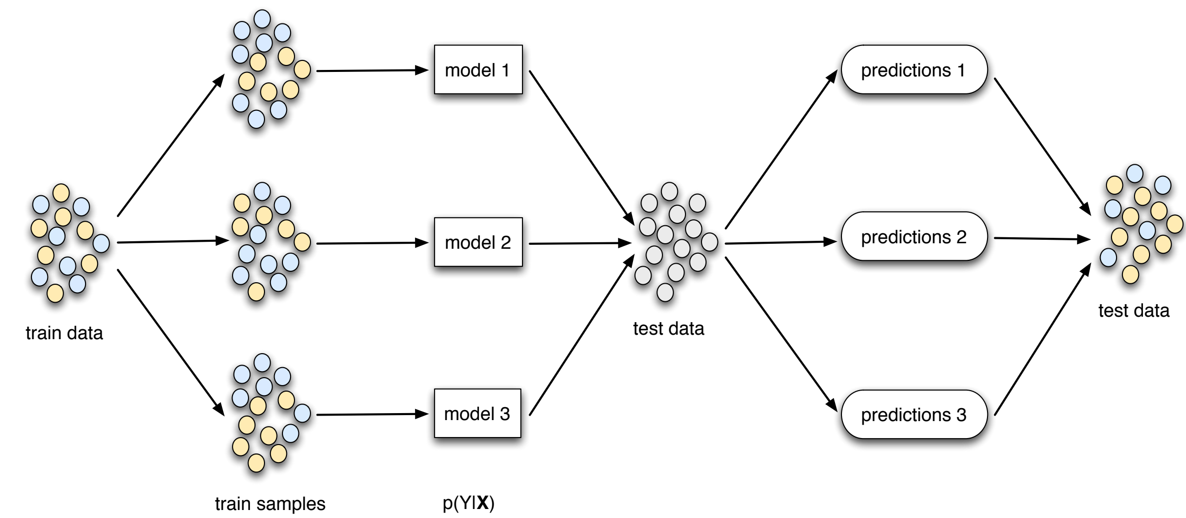


Bagging Mechanism



- **Mechanism**
  — Majority Voting (inspired by voting)
- **Given**
  — Ensemble of hypotheses h1, h2, h3, h4, h5
  — Hypotheses might be produced by different algorithms or different data
- **Goal**
  — Make a classification based on different hypotheses
- **To make a prediction for instance x**
  — Allow each hypothesis to return a predicted class for x (vote)
  hi(x): predicted class by hypothesis hi
  — Return majority(h1(x),h2(x),h3(x),h4(x),h5(x))
  <most popular predicted class>

Misclassification



- For ensemble (majority classification) to be wrong, at least 3 out of 5 hypotheses must be wrong

- The larger the ensemble,
  — More hypotheses must be wrong for the majority to be wrong
  — Under the right assumptions, we can show that this is unlikely

Bagging Assumptions

- Each hi makes error with probability p
- Hypotheses are independent
  — Probability of some error is independent
  — Type of error is independent
  — One makes a mistake does not imply others will too
    — Majority is robust

Bagging Assumptions

- **Majority voting of n hypotheses**
  — pr(k hypotheses make an error) $\binom{n}{k} p^k(1-p)^{n-k}$
  — pr(majority makes an error) $\Sigma_{k>n/2} \binom{n}{k} p^k(1-p)^{n-k}$

for each possible subset k

k out of n will make error with probability k

remaining n-k compute correct class with probability 1-p

consider all k's that are at least n/2

sum the same expression for k > n/2

- **Example** n=5, p=0.1
  — According to above formula above, probability of err(majority) < 0.01
  — Exercise: plug in formula: n = 5, k = 3,4,5, p=0.1
  — Combine hypotheses that make mistakes 10% of the time
  — Resulting ensemble makes mistakes less than 1% of the time
  — When at least 3 out 5 make a mistake

- Big performance boost by just an ensemble of size 5!

Weighted Majority

- **Assumptions do not hold in practice**
  — Hypotheses rarely independent
  — Some hypotheses have less errors than others
  — Probability of making a mistake is not the same

- **Solution**
  — Take a **weighted majority**

- **Intuition**
  — Decrease weight of correlated hypotheses
  (Decrease weights of algorithms that generate similar hypotheses)
  — Increase weight of good hypotheses
  => improve overall accuracy

- Ensemble technique that uses **weighted** majority ??

Boosting

- Popular ensemble technique

- Computes a weighted majority vote

- To obtain hypotheses
  — Use a base learning technique that produces classifiers
  — Base learner can be a weak learner (not very good learning technique)
  — Pooling a lot of its resulting hypotheses
  — Obtain higher accuracy

- “Boost” a “weak learner”
  — As long as the weak learner performs slightly better than random

- Operates on a weighted training set
  — Need reweight training set

Boosting

- Dataset => base learner => hypothesis
- Perturb dataset by changing weights of data instances
- Perturbed dataset => base learner => different hypothesis

- **Intuition**
  — Select instances that are misclassified by one hypothesis
  — Increase their weight => make them more important
  — So next time, algorithm has a better chance of generating a hypothesis that classifies those instances correctly

Weighted Training Set

- Boosting uses learning algorithms that can work with weighted training set
  — All supervised learning techniques can be adjusted to work with weighted training set

- **Examples of how to take weights into account**

- Weights: integers
  — Simply duplicate data instances corresponding to integer weights

- Weights: positive real numbers
  — Change objective by introducing a weight for each data point
  — Change loss function so penalty is proportional to weight
  — Normally, optimization: loss function (objective) computes loss for every data point
  — In this case: multiply loss by weight of data point
  — Example: weighted squared error
    — The goal is to have squared loss be smaller for some data points
    — So we take a weighted combination of squared loss of every data point

- Usually after that
  — There is a normalization step that normalizes the weights so they all sum to one

Weighted Training Set

- It is the relative weight magnitude that matters

- Increasing weight of misclassified points
  Automatically decreases weight of correctly classified ones

- Algorithm wants to minimize loss
  — Those with higher weights contribute larger loss
  — So they are more important to optimize for (minimize their loss)

- Bias algorithm to correctly classify instances that cost more when misclassified

Learning with a Weighted Training Set

- **Learning with a weighted training set**
  — Supervised learning => minimize training error
  — Bias algorithm to correctly learn instances with big weights

- **Boosting Idea**
  — When instance is misclassified by hypothesis
  — Increase its weight
  — So next hypothesis is more likely to classify it correctly

Boosting Framework

- Set all instance weights w_x to 1

Learning algorithm: a base learner that works with weighted datasets

Data weights

- **Repeat**
  — h_i <= learn(dataset, weights)
  — Increase w_x of misclassified instances x

- **Until** sufficient number of hypotheses

Hypothesis weights

- **Ensemble hypothesis:** weighted majority of h_i's with weights w_i proportional to accuracy of h_i

Boosting Framework

- Base learner can be any algorithm
  — Perceptron, SVM, Neural Network, Decision Tree
  — Simple & quick base learner
    — Since we generate multiple hypotheses: don’t use complex ones
    — Also, complex hypotheses usually don’t need boosting

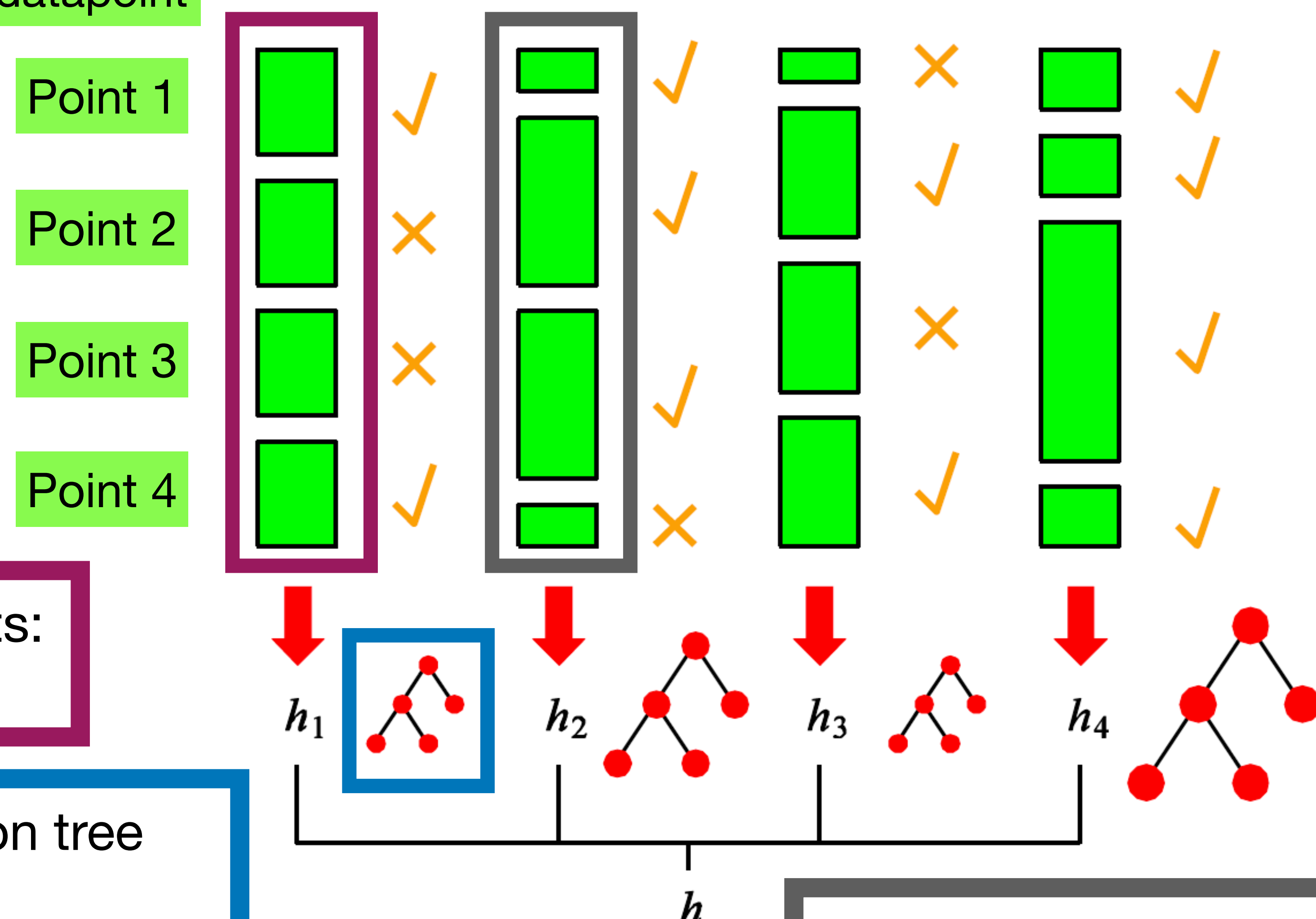
- Weight data
  — To increase penalty of misclassification

- If we get perfect hypothesis w.r.t. training
  — It may not be perfect on test set
  — To avoid overfitting
    — Combine with less perfect hypotheses
    — And give it a higher weight

Boosting Framework

Small dataset: 4 data points

Rectangle size ==> weight of datapoint



Start with uniform weights:
Equal size rectangles

Base learner: decision tree
(can be anything)
Base learner generates h1

Check: correct classification
Cross: incorrect
h1 training accuracy: 50%

Take weighted combination of hypotheses
PREDICTIONS

Increase weights of misclassified points
Normalize weights =>
Correctly classified get smaller weights

AdaBoost: Adaptive Boosting

— A boosting algorithm that made boosting popular
— Theory behind chosen expressions => proven bounds on accuracy & on accuracy improvement

- w_j <= 1/N Forall$_j$
- For m=1 to M do

  h_m <= learn(dataset, w)
  err <= 0
  For each (x_j, y_j) in dataset do
  If $h_m(x_j)$!= y_j then err <= err + w_j
  For each (x_j, y_j) in dataset do
  If $h_m(x_j)$ = y_j then w_j <= w_j err / (1-err)
  w <= normalize(w)
  z_m <= log [(1-err) / err]

- Return weighted-majority(h, z)

Initialize to uniform weights
w: vector of N instance weights

z: vector of M hypotheses weights
Loop, every iteration: base learner

given data & weights, generate h

evaluate h: measure its error rate
totalErr += weight of misclassified point

reduce weight of correctly classified

normalize data weights
compute h's weight

What can we boost?

- **Boosting**
  — Most effective when you work with weak learners that are far from perfect
- **Weak learner**
  — Algorithm produces hypotheses at least as good as random classifier
  — (Slightly better than random)
  — If accuracy is less than random, flip labels, then it would do slightly better than random
- **Not worth using a sophisticated base learner**
  — Because we generate multiple hypotheses
  — So it needs to be simple & fast
- **Examples**
  — Rules of thumb (manually specified)
  — Decision stumps (decision trees of one node)
  — Perceptrons
  — Naïve Bayes models

Boosting Paradigm

- **Advantages**
  — No need to learn a perfect hypothesis
  — Can boost any weak learning algorithm
  — Boosting is very simple to program
  — Good generalization
  => Robust to overfitting because it uses multiple hypotheses
  => Does not pick best, which might lead to overfitting

- **Can think of boosting as a tractable approximation to Bayesian Learning**
  — Generate multiple hypotheses with some weights and take weighted combination
  — Similar to bayesian learning: we have posterior distribution over hypotheses
  & we take weighted combination to make a prediction
  — Difference: boosting uses a specific base learner to generate hypotheses
  — While Bayesian Learning: considers entire space of hypotheses

- **Paradigm shift**
  — Don’t try to learn a perfect hypothesis
  — Just learn simple rules of thumbs and boost them

Boosting Paradigm

- When we already have a bunch of hypotheses
  — Boosting provides a principled approach to combine them

- **Useful for**
  — Sensor fusion
  — Combining experts
  — Where multiple heuristics have been proposed
  — Some may be weak with no strong backing theory
  — Boosting provides a way to combine different classifiers
  or different predictors, perhaps based on different sensors

Applications

- Any supervised learning task
  - Collaborative filtering (Netflix challenge - movie recommendation)
    => ensemble learning is key
  - Body part recognition (Kinect by Microsoft)
    => ensemble of decision trees
  - Spam filtering
  - Speech recognition/natural language processing
  - Data mining
  - Etc.

Summary

- Ways to improve predictions
  - **Ensemble Learning** — combine models

Discussion

- What model are you considering for your project?
- Can you consider an ensemble of models?
- How would you do it?
- Start a discussion topic on canvas
  if you like to brainstorm further with others