

Machine Learning

Randomized Trees, Forests, and Ferns - Lecture IX

Videos for This Lecture

- Repetition Video (Part 0)
- Decision Tree (Part 1)
- Randomized Decision Trees (Part 2)
- Random Forests (Part 3)
- Extremely randomized trees (Part 4)
- Ferns (Part 5)

Course Outline

Basic Concepts

- Parametric Method,
- Bayesian Learning and Nonparametrics Methods
- Clustering and Mixture of Gaussians

Classification Approaches

- Linear Discriminants
- Ensemble Methods and Boosting
- Randomized Trees, Forest

Reinforcement Learning

- Classical Reinforcement Learning
- Deep Reinforcement Learning

Deep Learning

- Foundations
- Optimization

Today's Topics

Decision Trees

- Main concepts

Randomized Decision Trees

- Randomized attribute selection

Forests

- Bootstrap sampling
- Ensemble of randomized trees
- Posterior sum combination
- Analysis

Extremely randomized trees

- Random attribute selection

Ferns

- Fern structure
- Semi-Naïve Bayes combination

Recap: AdaBoost – “Adaptive Boosting”

Main idea

Instead of resampling, reweight misclassified training examples.

Increase the chance of being selected in a sampled training set.

Or increase the misclassification cost when training on the full set.

Components

$h_m(x)$: “weak” or base classifier

– Condition: <50% training error over any distribution

$H(x)$: “strong” or final classifier

AdaBoost:

Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:

$$H(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^M \alpha_m h_m(\mathbf{x}) \right)$$

AdaBoost – Final Algorithm

1. Initialization: Set $w_n^{(1)} = \frac{1}{N}$ for $n = 1, \dots, N$

2. For $m = 1, \dots, M$ **iterations**

a) Train a new weak classifier $h_m(\mathbf{x})$ using the current weighting coefficients $\mathbf{W}^{(m)}$ by minimizing the weighted error function

$$J_m = \sum_{n=1}^N w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)$$
$$I(A) = \begin{cases} 1 & \text{if } A \text{ is true} \\ 0 & \text{else} \end{cases}$$

b) Estimate the weighted error of this classifier on \mathbf{X} :

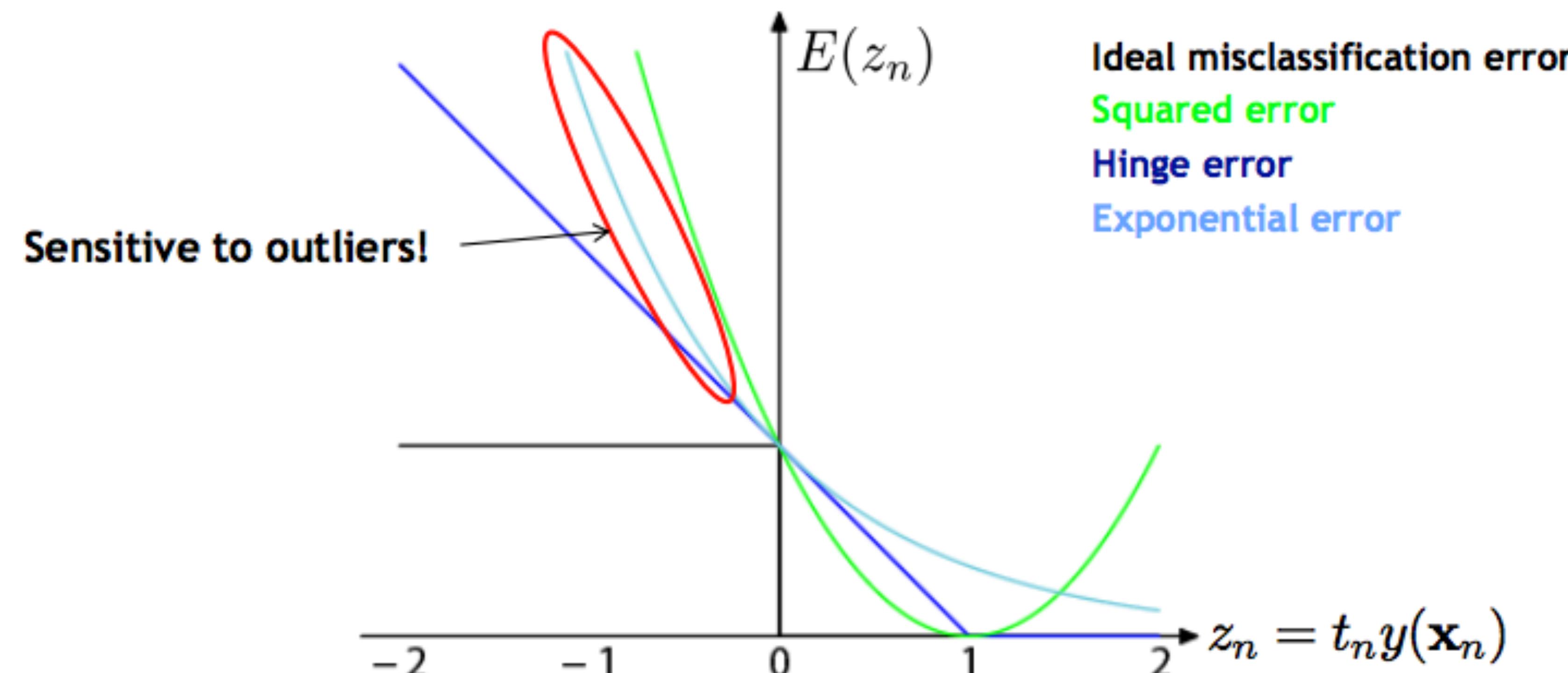
$$\epsilon_m = \frac{\sum_{n=1}^N w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)}{\sum_{n=1}^N w_n^{(m)}}$$

c) Calculate a weighting coefficient for $h_m(\mathbf{x})$: $\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}$

d) Update the weighting coefficients:

$$w_n^{(m+1)} = w_n^{(m)} \exp \{ \alpha_m I(h_m(\mathbf{x}_n) \neq t_n) \}$$

Discussion: AdaBoost Error Functions



Exponential error used in AdaBoost

- No penalty for too correct data points, fast convergence.
- Disadvantage: exponential penalty for large negative values!
=> Less robust to outliers or misclassified data points!?

Part 1, Video RandomForest_p1

Decision Trees

Today's Topics

Decision Trees

- Main concepts

Randomized Decision Trees

- Randomized attribute selection

Forests

- Bootstrap sampling
- Ensemble of randomized trees
- Posterior sum combination
- Analysis

Extremely randomized trees

- Random attribute selection

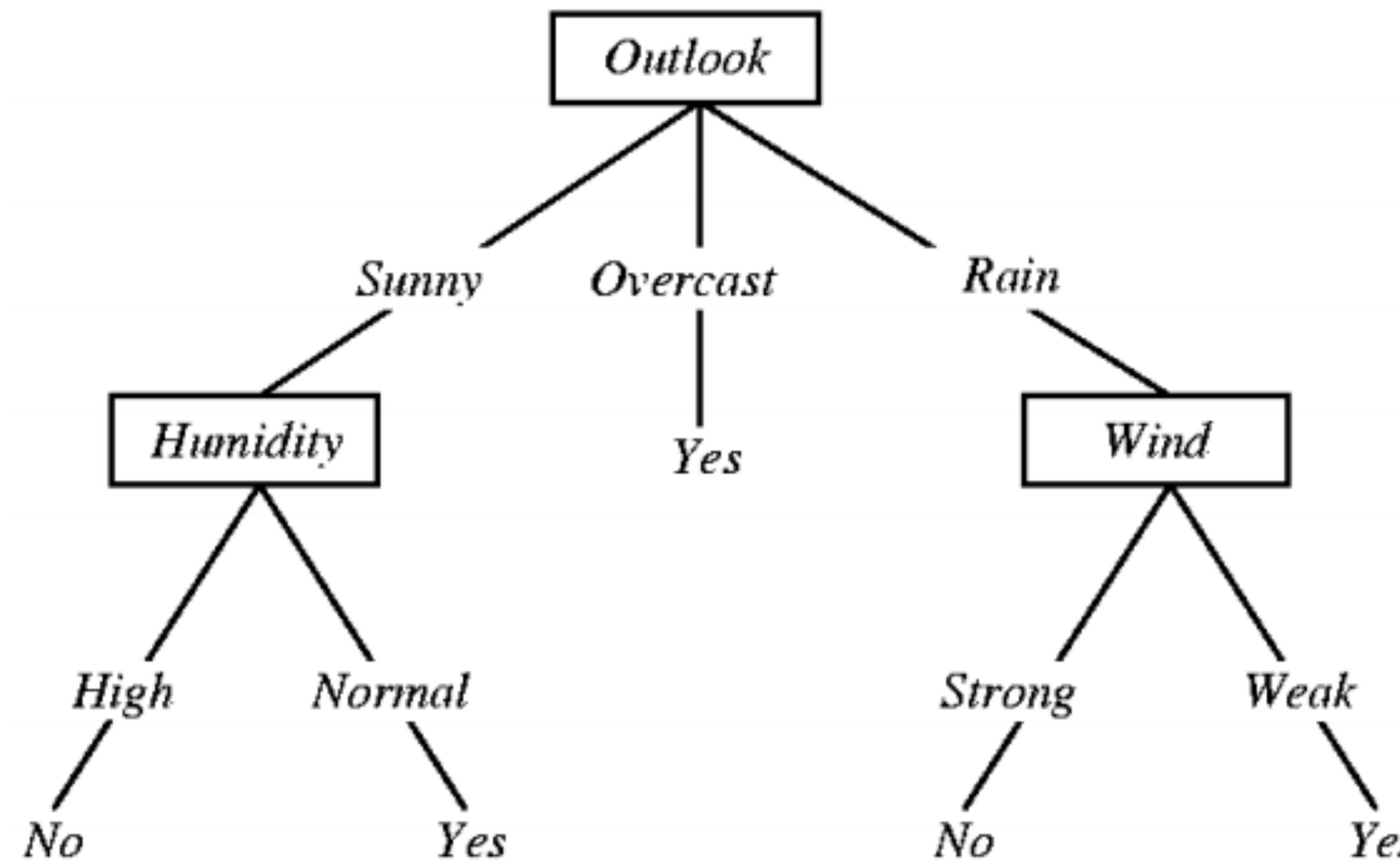
Ferns

- Fern structure
- Semi-Naïve Bayes combination

Decision Trees

- Very old technique
 - Origin in the 60s, might seem outdated.
- But...
 - Can be used for problems with nominal data
 - E.g. attributes color {red, green, blue} or weather {sunny, rainy}.
 - Discrete values, no notion of similarity or even ordering.
 - Interpretable results
 - Learned trees can be written as sets of if-then rules.
 - Methods developed for handling missing feature values.
 - Successfully applied to broad range of tasks
 - E.g. Medical diagnosis.
 - E.g. Credit risk assessment of loan applicants
 - Some interesting novel developments building on top of them...

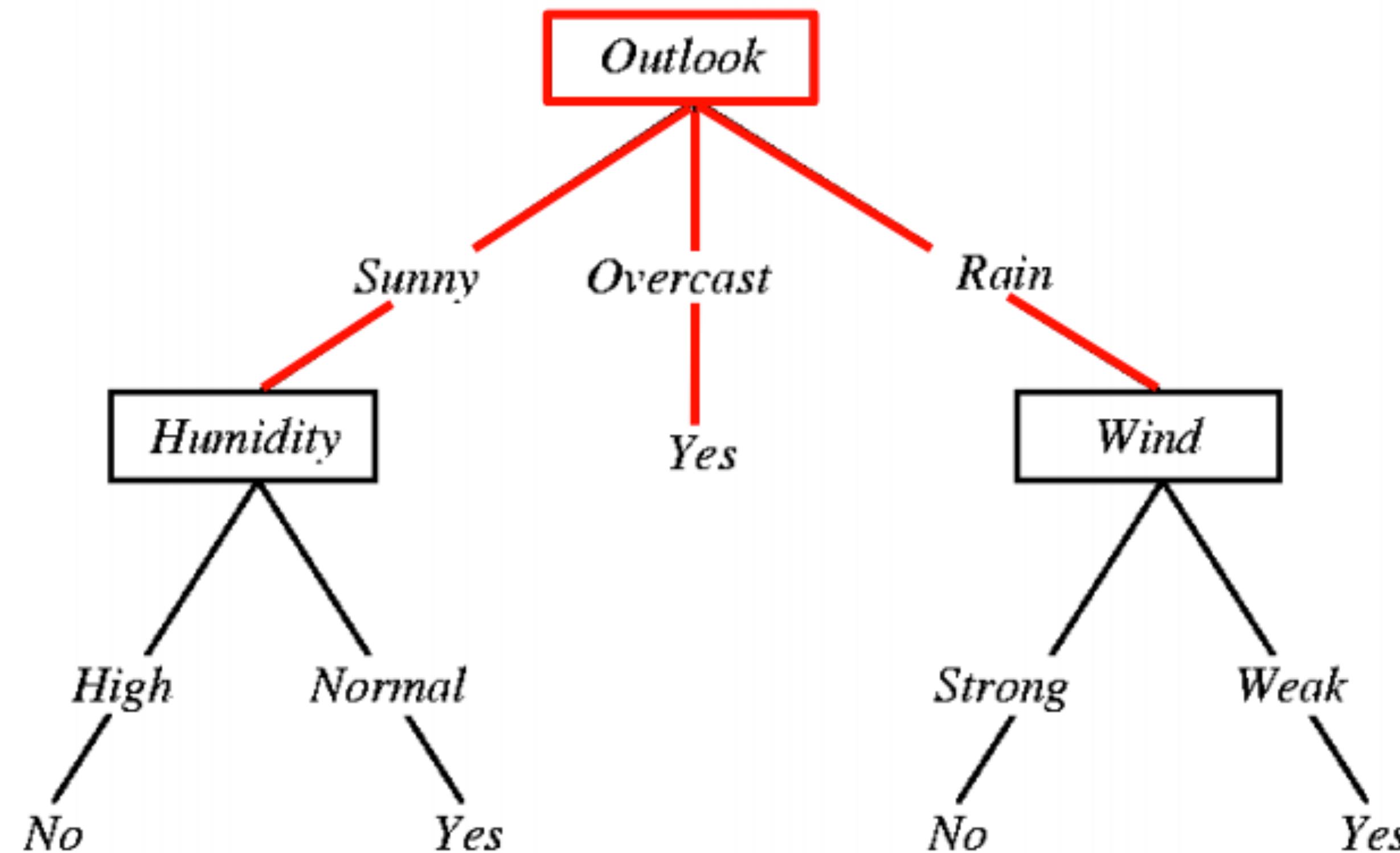
Decision Trees



Example:

- “Classify Saturday mornings according to whether they’re suitable for playing tennis.”

Decision Trees



Elements

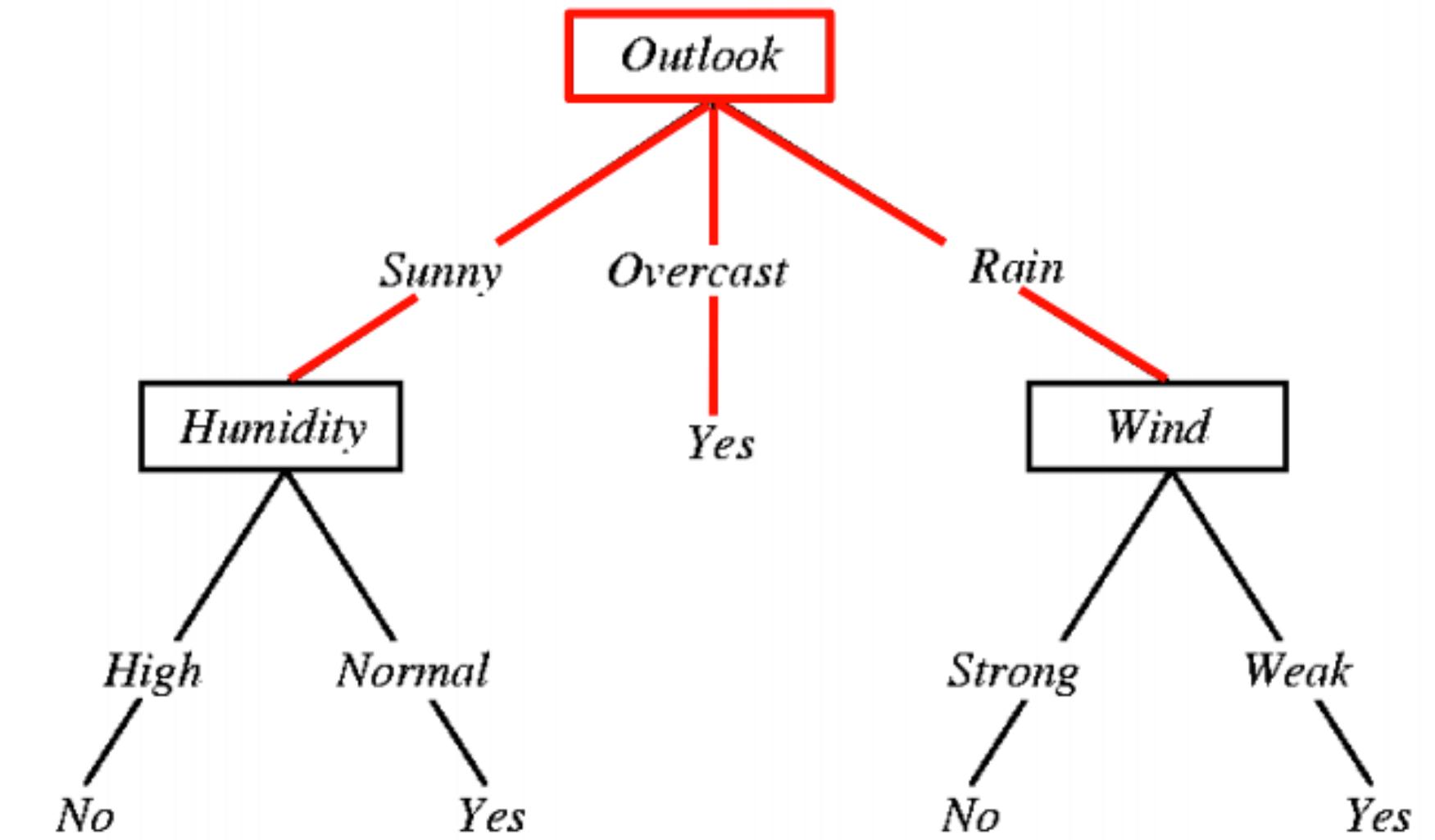
- Each node specifies a test for some attribute.
- Each branch corresponds to a possible value of the attribute.

CART Framework

Six general questions

1. **Binary or multi-valued** problem?
 - I.e. how many splits should there be at each node?
2. Which **property** should be tested at a node?
 - I.e. how to select the query attribute?
3. When should a node be declared a **leaf**?
 - I.e. when to stop growing the tree?
4. How can a grown tree be simplified or **pruned**?
 - Goal: reduce overfitting.
5. How to deal with **impure nodes**?
 - I.e. when the data itself is ambiguous.
6. How should **missing attributes** be handled?

This will be
our focus!



CART – 2. Picking a Good Splitting Feature

Goal

- Want a tree that is simple/small as possible (Occam's razor).
- But: Finding a minimal tree is an NP-hard optimisation problem.

Greedy top-down search

- Efficient, but not guaranteed to find the smallest tree.
- Seek a property T at each node N that makes the data in the child nodes as **pure** as possible.
- For formal reasons more convenient to define **impurity** $i(N)$.
- Several possible definitions explored.

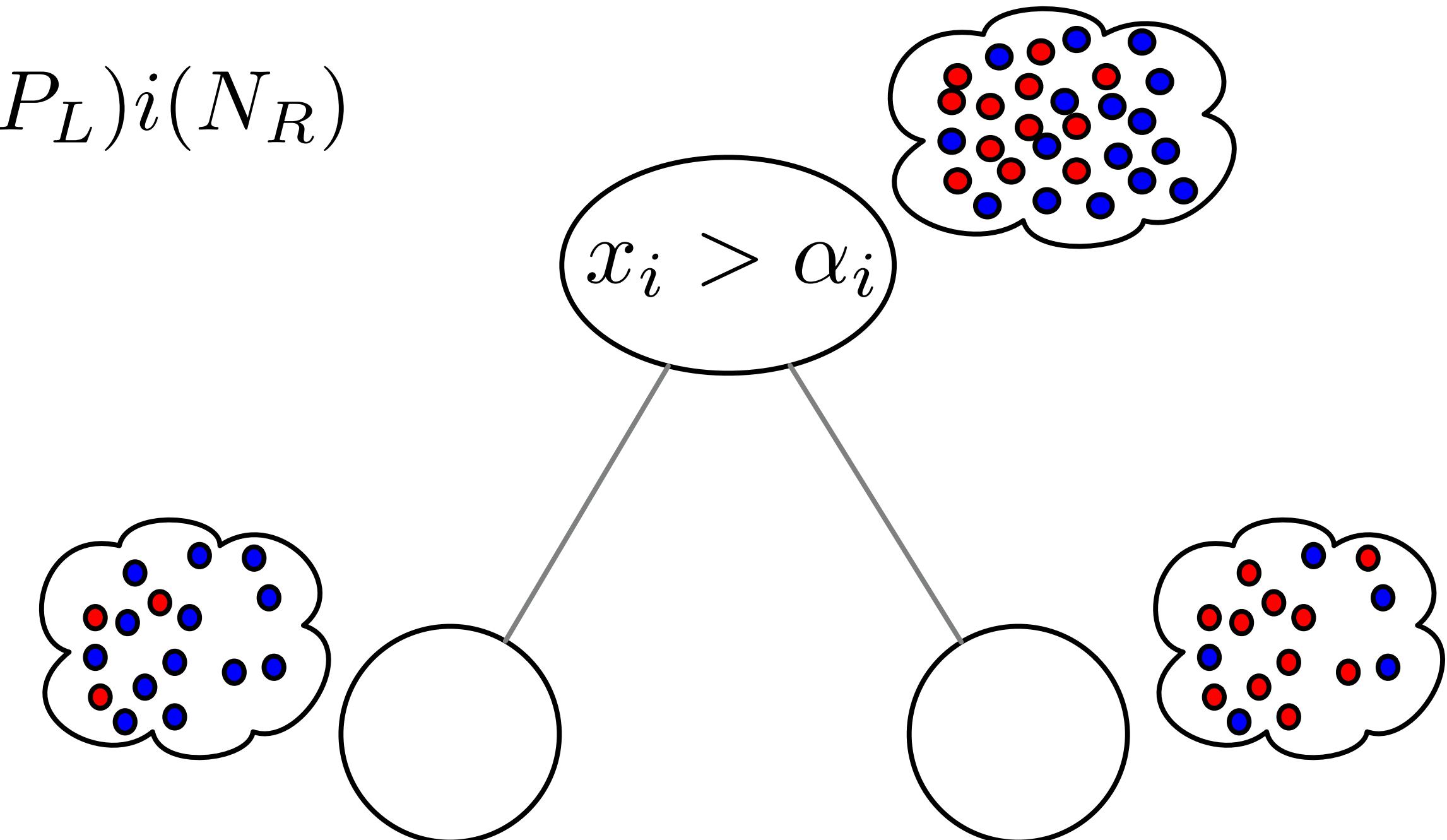
Picking a Good Splitting Feature

Goal

- Select the query (=split) that decreases impurity the most.

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$

Fraction of points
in left child node



Picking a Good Splitting Feature

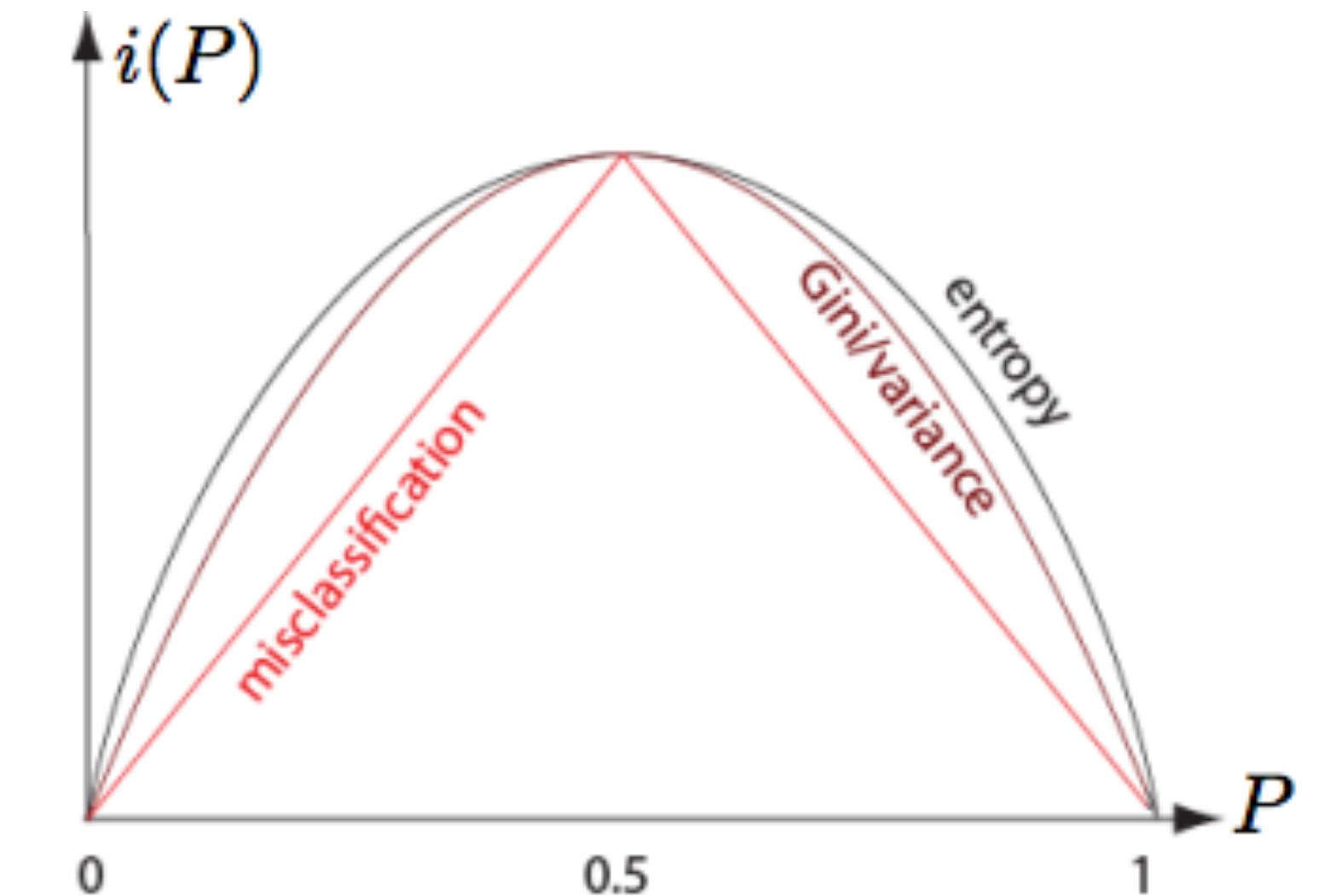
Impurity measures

- Entropy impurity (information gain):

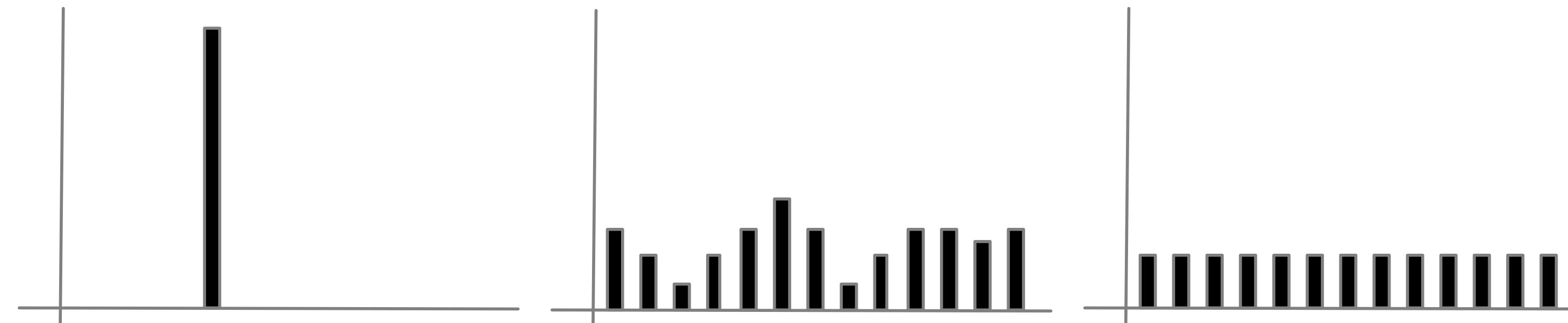
$$i(N) = - \sum_j p(C_j|N) \log_2 p(C_j|N)$$

- Gini impurity:

$$i(N) = - \sum_{i \neq j} p(C_i|N)p(C_j|N) = \frac{1}{2} [1 - \sum_j p^2(C_j|N)]$$



Entropy concept:



Decision Trees – Properties

- **Simple learning procedure, fast evaluation.**
- **Can be applied to metric, nominal, or mixed data.**
- **Often yield interpretable results.**

Decision Trees – Limitations

Often produce noisy (bushy) or weak (stunted) classifiers.

Do not generalize too well.

Training data fragmentation:

- As tree progresses, splits are selected based on less and less data.

Overtraining and undertraining:

- Deep trees: fit the training data well, will not generalize well to new test data.
- Shallow trees: not sufficiently refined.

Stability

- Trees can be very sensitive to details of the training points.
- If a single data point is only slightly shifted, a radically different tree may come out!
=> Result of discrete and greedy learning procedure.

Expensive learning step

- Mostly due to costly selection of optimal split

Part 2, Video RandomForest_p2

Randomized Decision Trees

- Randomized attribute selection

Today's Topics

Decision Trees

- Main concepts

Randomized Decision Trees

- Randomized attribute selection

Forests

- Bootstrap sampling
- Ensemble of randomized trees
- Posterior sum combination
- Analysis

Extremely randomized trees

- Random attribute selection

Ferns

- Fern structure
- Semi-Naïve Bayes combination

Decision Trees – Computational Complexity

Given

- Data points $\{x_1, \dots, x_N\}$
- Dimensionality D

Complexity

- Storage: $O(N)$
- Test runtime: $O(\log N)$
- Training runtime: $O(DN^2 \log N)$
 - Most expensive part.
 - Critical step: selecting the optimal splitting point.
 - Need to check D dimensions, for each need to sort N data points $O(DN \log N)$

Randomized Decision Trees (Amit & Geman 1997)

Decision trees: main effort on finding good split

- Training runtime: $O(DN^2 \log N)$
- This is what takes most effort in practice.
- Especially cumbersome with many attributes (large D).

Idea: randomize attribute selection

- No longer look for globally optimal split.
- Instead randomly use subset of K attributes on which to base the split.
- Choose best splitting attribute e.g. by maximizing the information gain (= reducing entropy):

$$\Delta E = \sum_{k=1}^K \frac{|S_k|}{|S|} \sum_{j=1}^N p_j \log_2(p_j)$$

Randomized Decision Trees

Randomized splitting

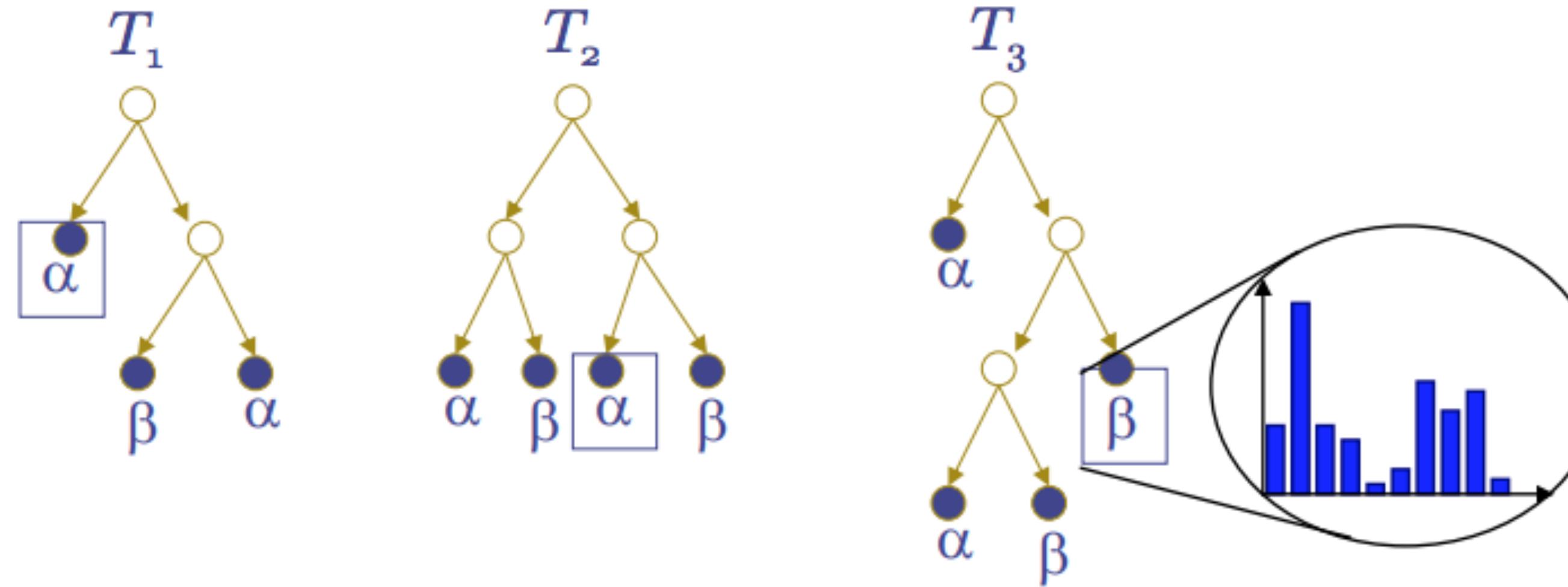
Training runtime:

- Faster training: $O(KN^2 \log N)$ with $K \ll D$
- Use very simple binary feature tests.
- Typical choice
 - $K = 10$ for root node.
 - $K = 100d$ for node at level d .

Effect of random split

- Of course, the tree is no longer as powerful as a single classifier...
- But we can compensate by building several trees.

Ensemble Combination



Ensemble combination

- Tree leaves (l, η) store posterior probabilities of the target classes. $p_{l,\eta}(C|x)$
- Combine the output of several trees by averaging their posteriors (Bayesian model combination)

$$p(C|x) = \frac{1}{L} \sum_{l=1}^L p_{l,\eta}(C|x)$$

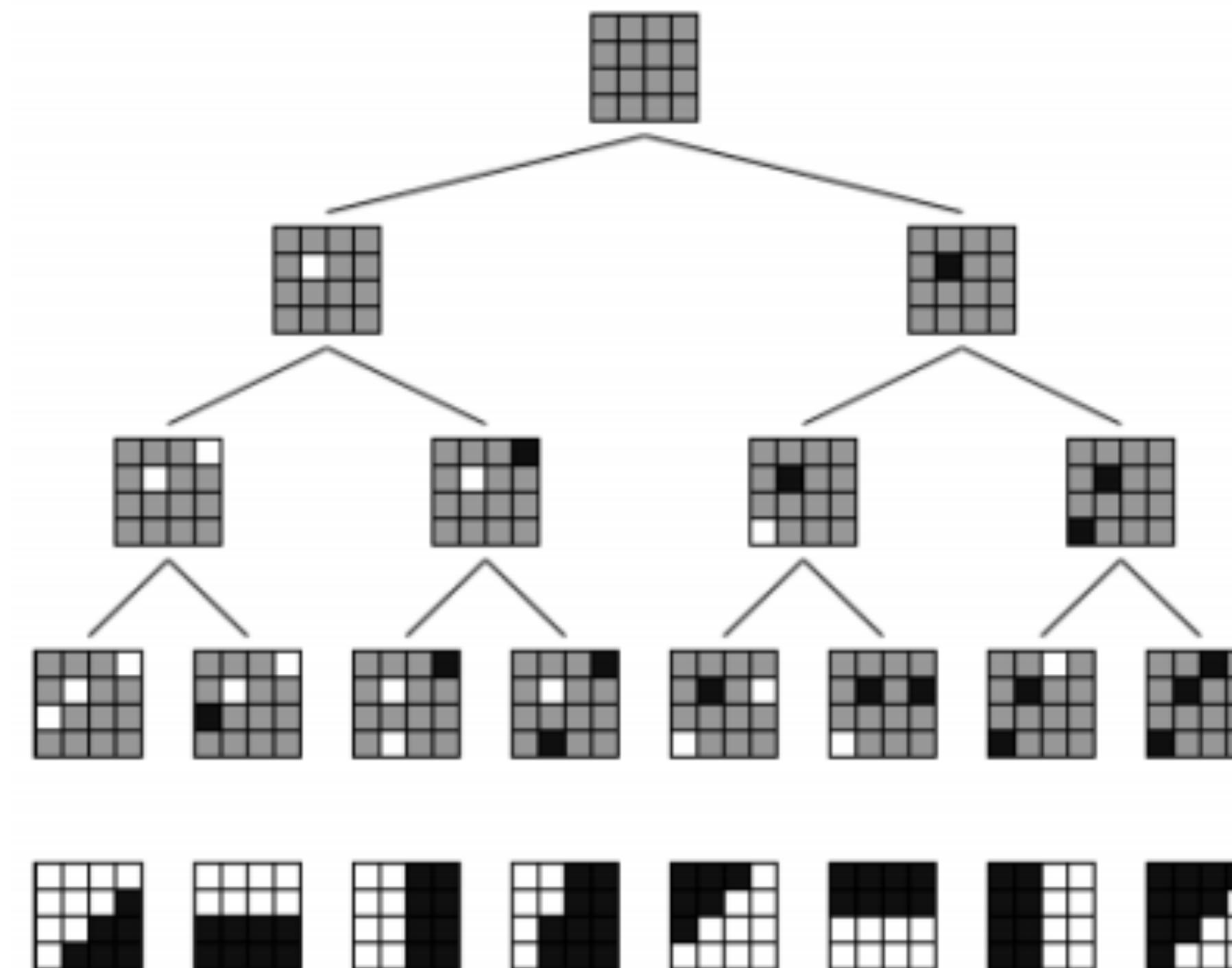
Ensemble Combination

Computer Vision: Optical character recognition

- Classify small (14x20) images of hand-written characters/digits into one of 10 or 26 classes.

Simple binary features

- Tests for individual binary pixel values.
- Organized in randomized tree.

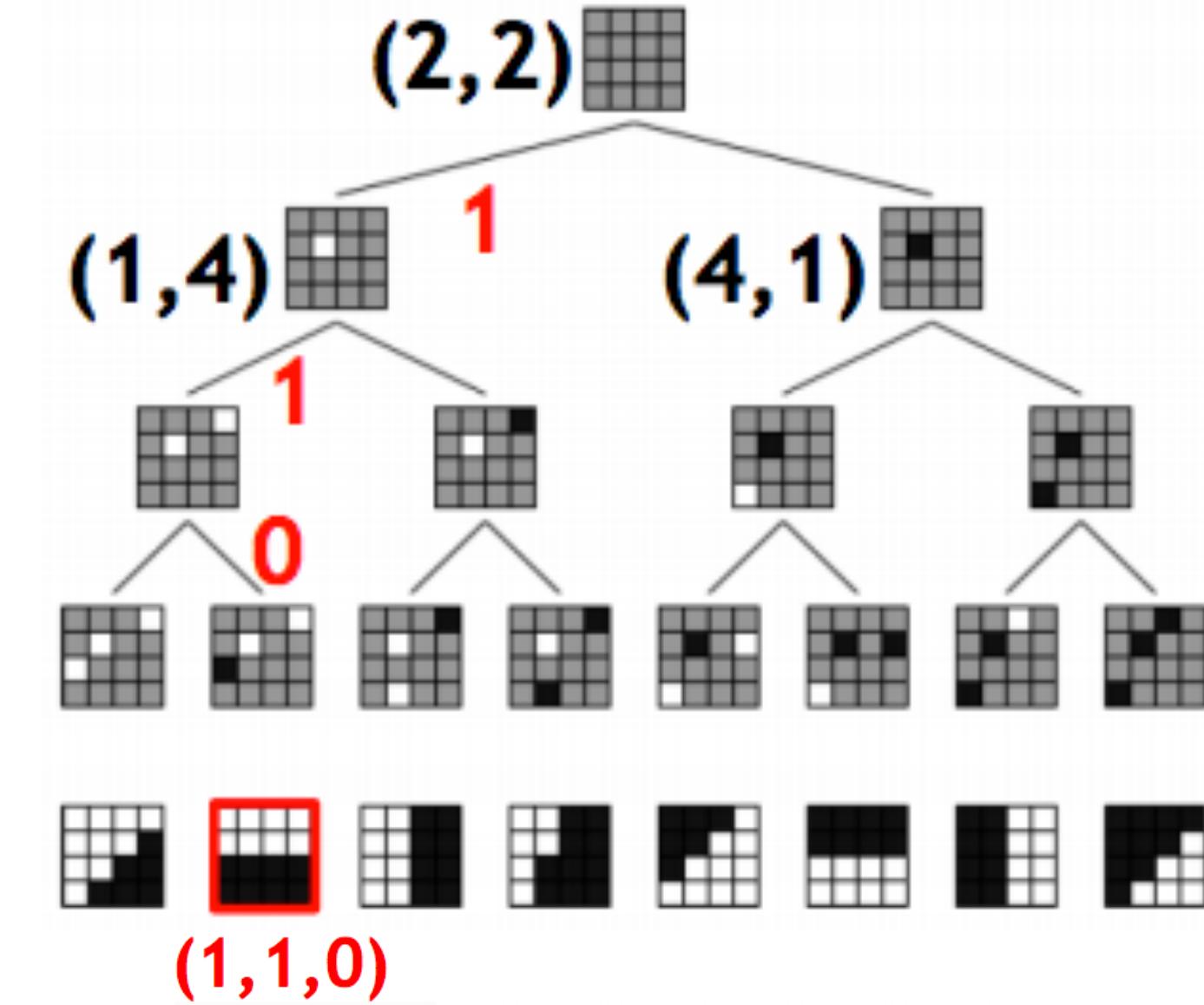


Y. Amit, D. Geman, Shape Quantization and Recognition with Randomized Trees, Neural Computation, Vol. 9(7), pp. 1545-1588, 1997.

Applications: Character Recognition

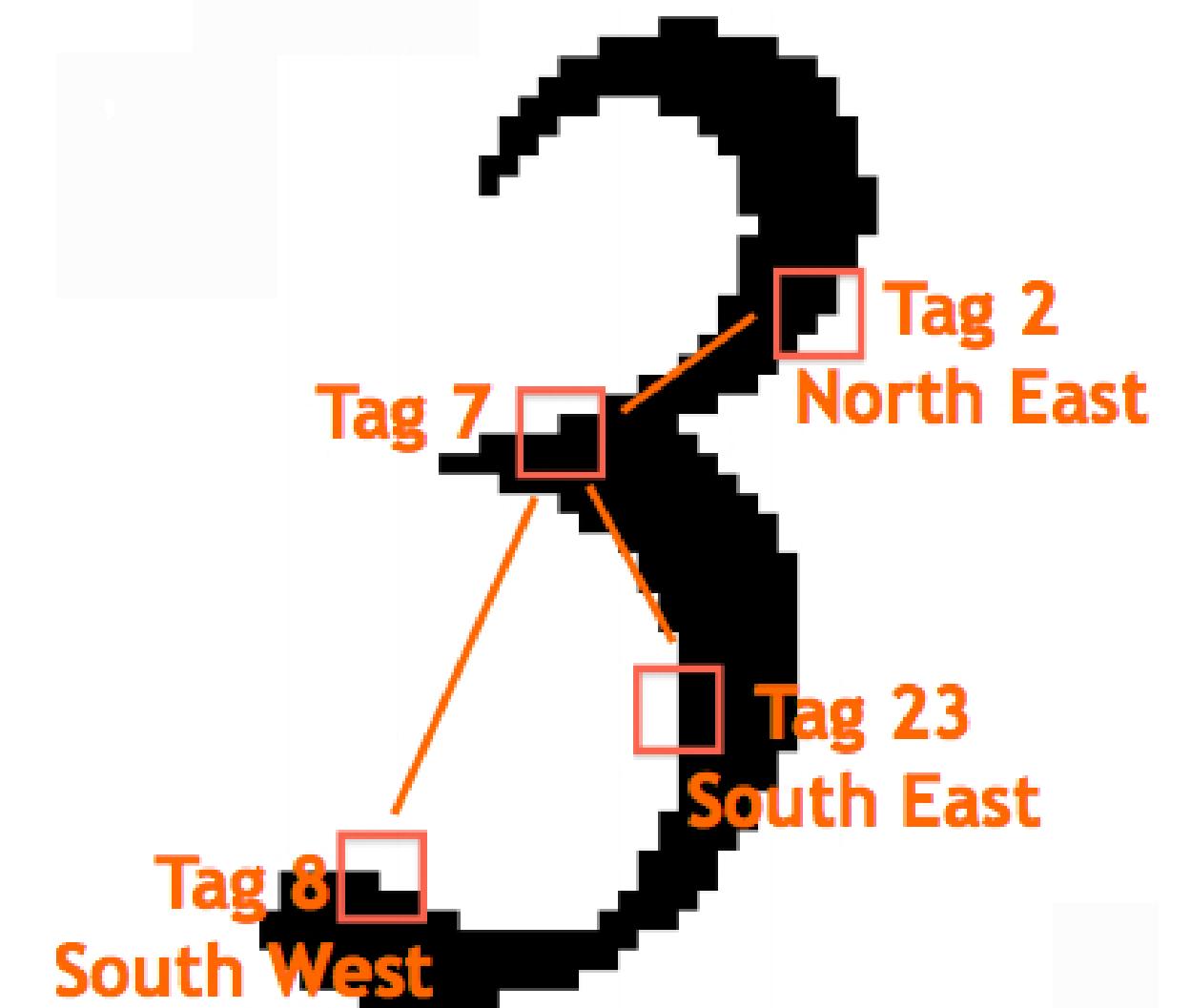
Image patches (“Tags”)

- Randomly sampled 4x4 patches
- Construct a randomized tree based on binary single-pixel tests
- Each leaf node corresponds to a “patch class” and produces a tag



Representation of digits (“Queries”)

- Specific spatial arrangements of tags
- An image answers “yes” if any such structure is found anywhere
- How do we know which spatial arrangements to look for?



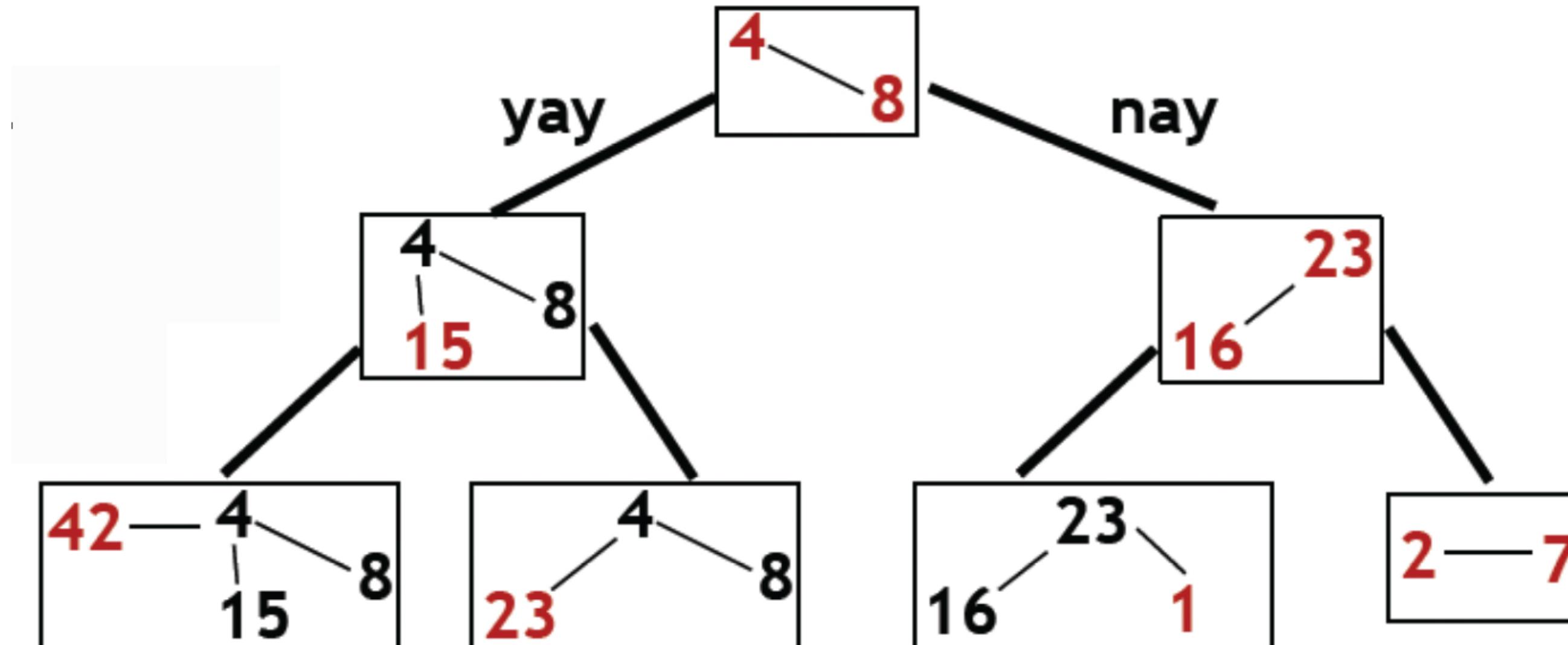
Applications: Character Recognition

Answer: Create a second-level decision tree!

- Start with two tags connected by an arc
- Search through extensions of confirmed queries (or rather through a subset of them, there are lots!)
- Select query with best information gain
- Recurse...

Classification

Average estimated posterior distributions stored in the leaves



Applications: Fast Keypoint Detection

Computer Vision: fast keypoint detection

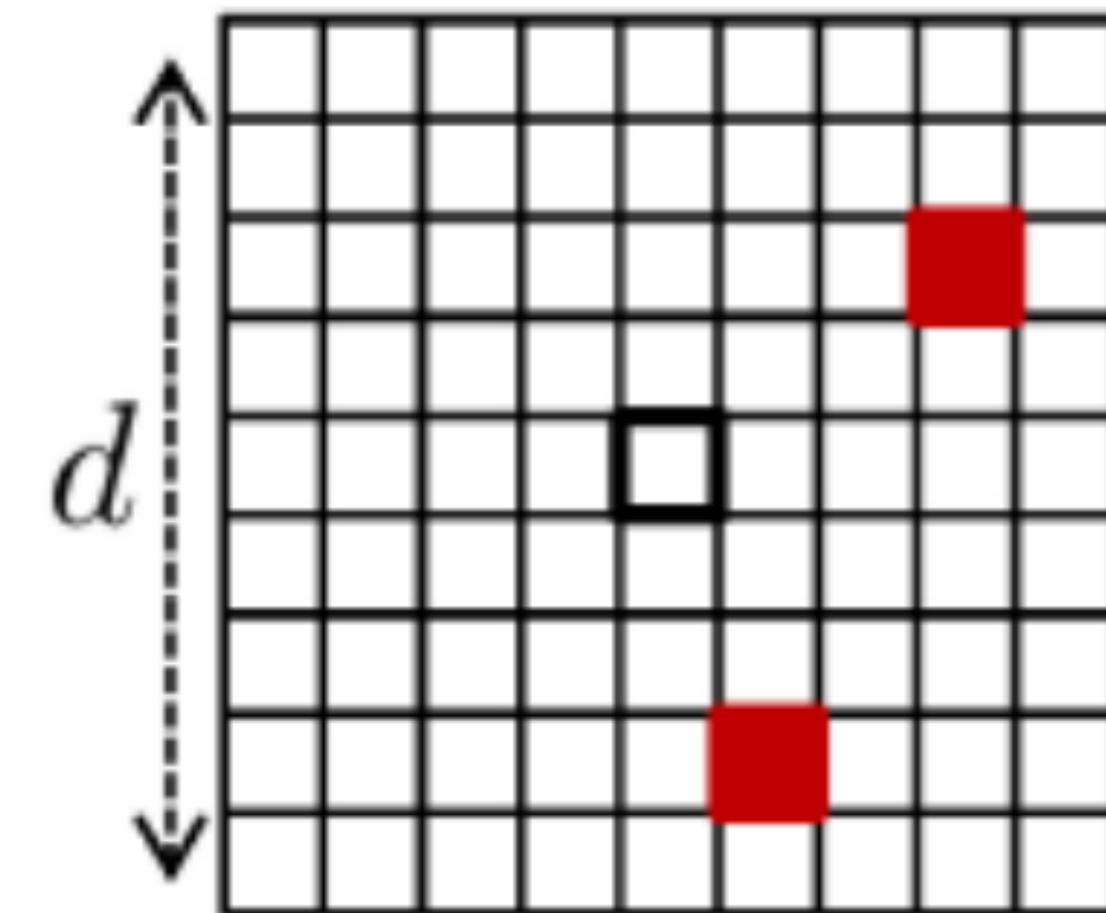
- Detect keypoints: small patches in the image used for matching
- Classify into one of ~200 categories (visual words)

Extremely simple features

- E.g. pixel value in a color channel (CIELab)
- E.g. sum of two points in the patch
- E.g. difference of two points in the patch
- E.g. absolute difference of two points

Create forest of randomized decision trees

- Each leaf node contains probability distribution over 200 classes
- Can be updated and re-normalized incrementally.



<https://www.youtube.com/watch?v=izwL5TrD3xU>

Applications: Fast Keypoint Detection



<https://www.youtube.com/watch?v=izwL5TrD3xU>

Slide source: B. Leibe Slides ML, 2015

Part 3, Video RandomForest_p3

Forests

- Bootstrap sampling
- Ensemble of randomized trees
- Posterior sum combination
- Analysis

Today's Topics

Decision Trees

- Main concepts

Randomized Decision Trees

- Randomized attribute selection

Forests

- Bootstrap sampling
- Ensemble of randomized trees
- Posterior sum combination
- Analysis

Extremely randomized trees

- Random attribute selection

Ferns

- Fern structure
- Semi-Naïve Bayes combination

Random Forests (Breiman 2001)

General ensemble method

- Idea: Create ensemble of many (very simple) trees

Empirically very good results

- Often as good as SVMs (and sometimes better)!
- Often as good as Boosting (and sometimes better)!

Standard decision trees: main effort on finding good split

- Random Forests trees put very little effort in this.
- CART algorithm with Gini coefficient, no pruning.
- Each split is only made based on a random subset of the available attributes.
- Trees are grown fully (important!).

Main secret

- Injecting the “right kind of randomness”

Random Forests – Algorithmic Goals

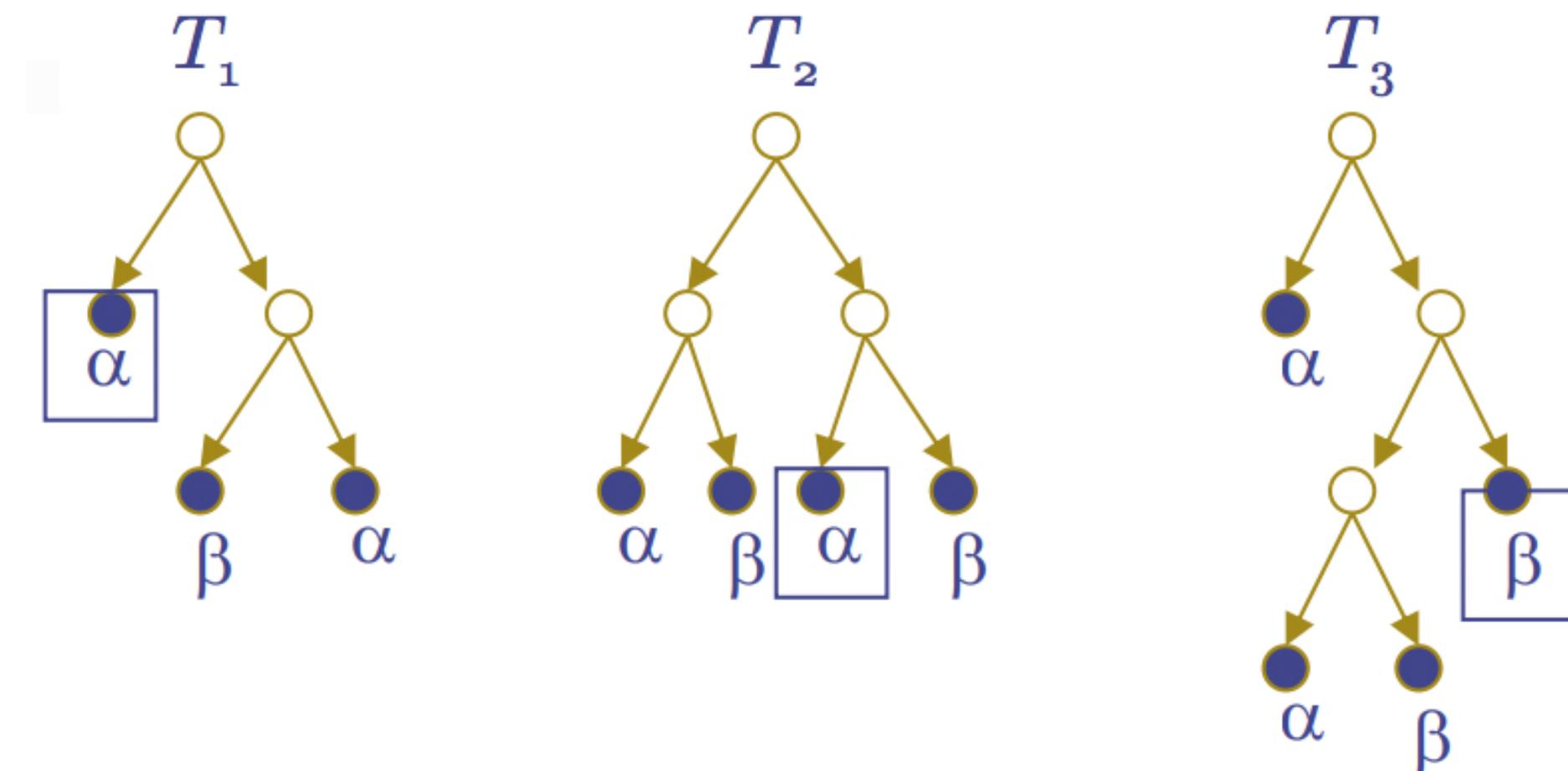
Create many trees (50 – 1,000)

Inject randomness into trees such that

- Each tree has maximal strength
 - I.e. a fairly good model on its own
- Each tree has minimum correlation with the other trees.
 - I.e. the errors tend to cancel out.

Ensemble of trees votes for final result

- Simple majority vote for category.
- Alternative (Friedman)
 - Optimally re-weight the trees via regularized regression (lasso).



Random Forests – Injecting Randomness (1)

Bootstrap sampling process

Select a training set by choosing N times with replacement from all N available training examples.

=> On average, each tree is grown on only ~63% of the original training data.

Remaining 37% “out-of-bag” (OOB) data used for validation.

- Provides ongoing assessment of model performance in the current tree.
- Allows fitting to small data sets without explicitly holding back any data for testing.
- Error estimate is unbiased and behaves as if we had an independent test sample of the same size as the training sample.

Random Forests – Injecting Randomness (2)

Random attribute selection

For each node, randomly choose subset of K attributes on which the split is based
(typically $K = \sqrt{N_f}$).

=> Faster training procedure

- Need to test only few attributes.

Minimizes inter-tree dependence

- Reduce correlation between different trees.

Each tree is grown to maximal size and is left unpruned

Trees are deliberately overfit

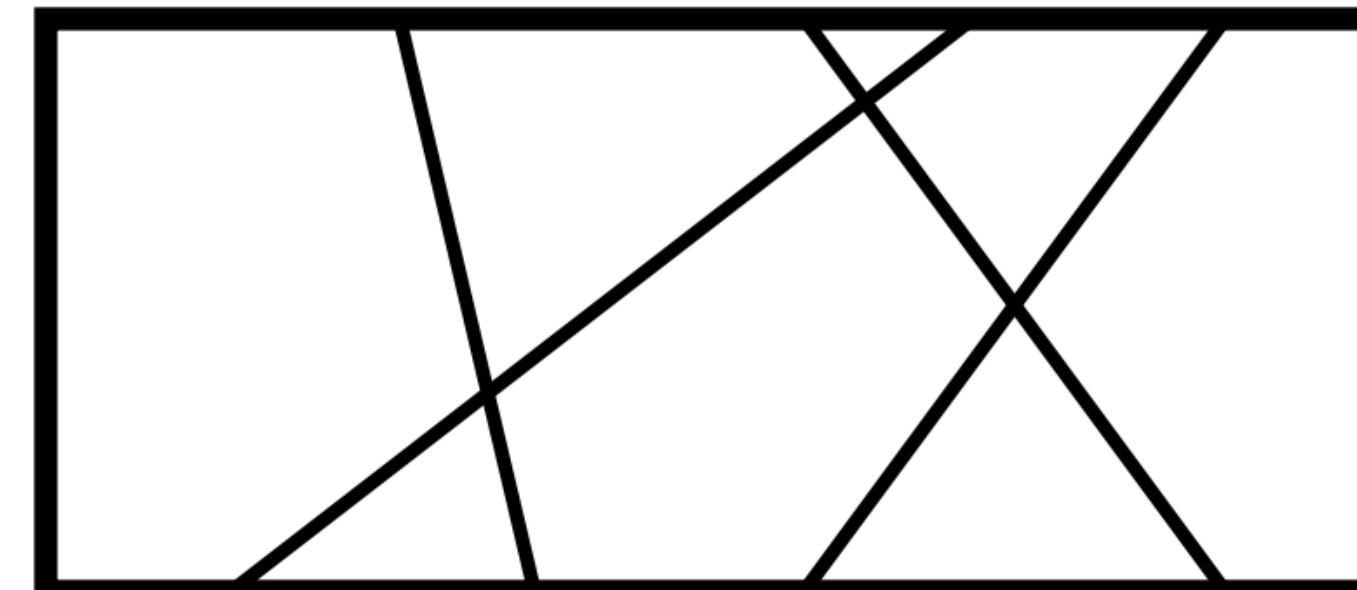
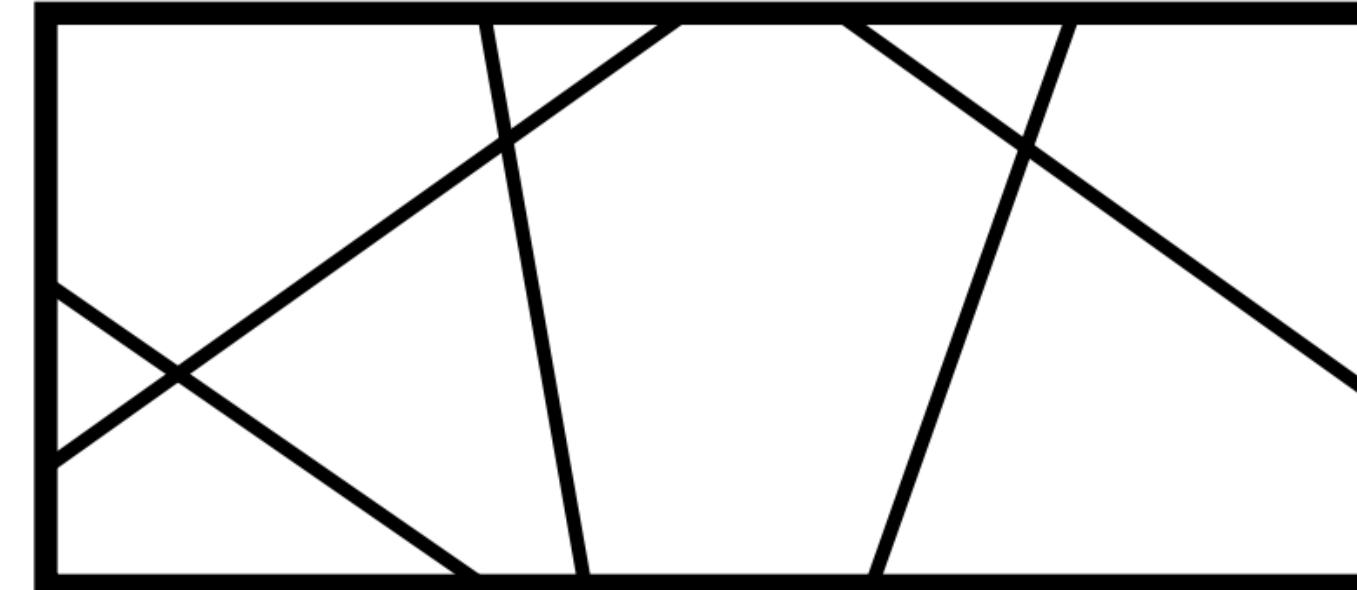
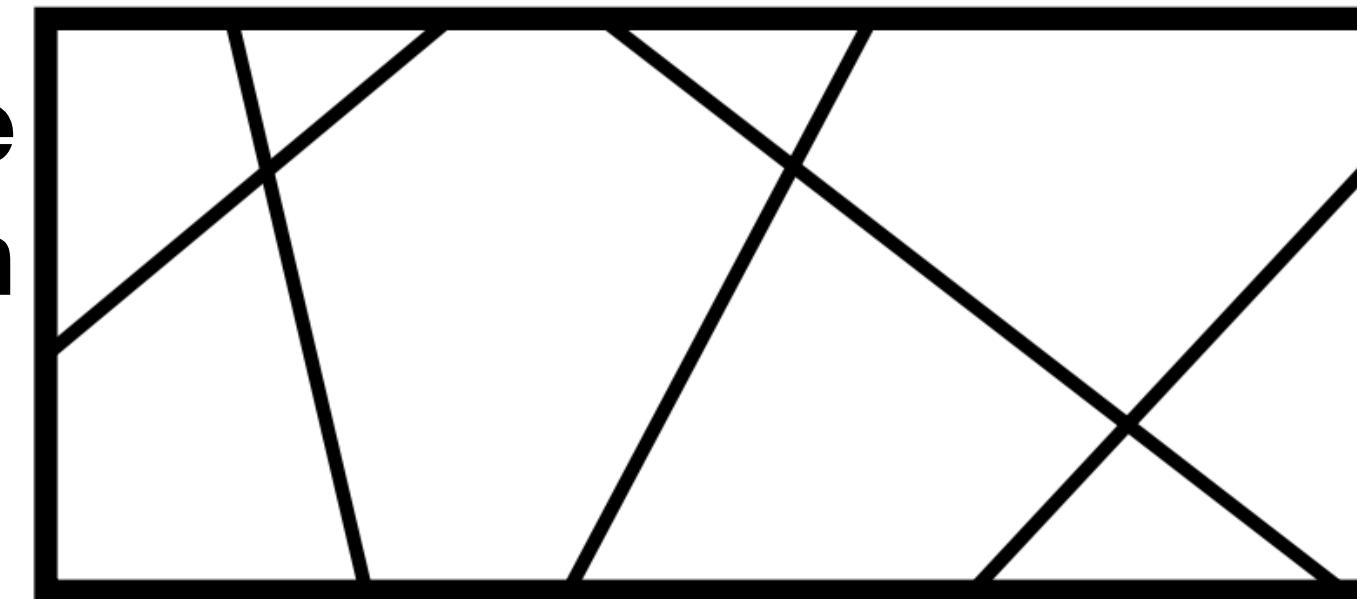
=> Become some form of nearest-neighbor predictor.

Bet You're Asking...

How can this possibly **ever** work???

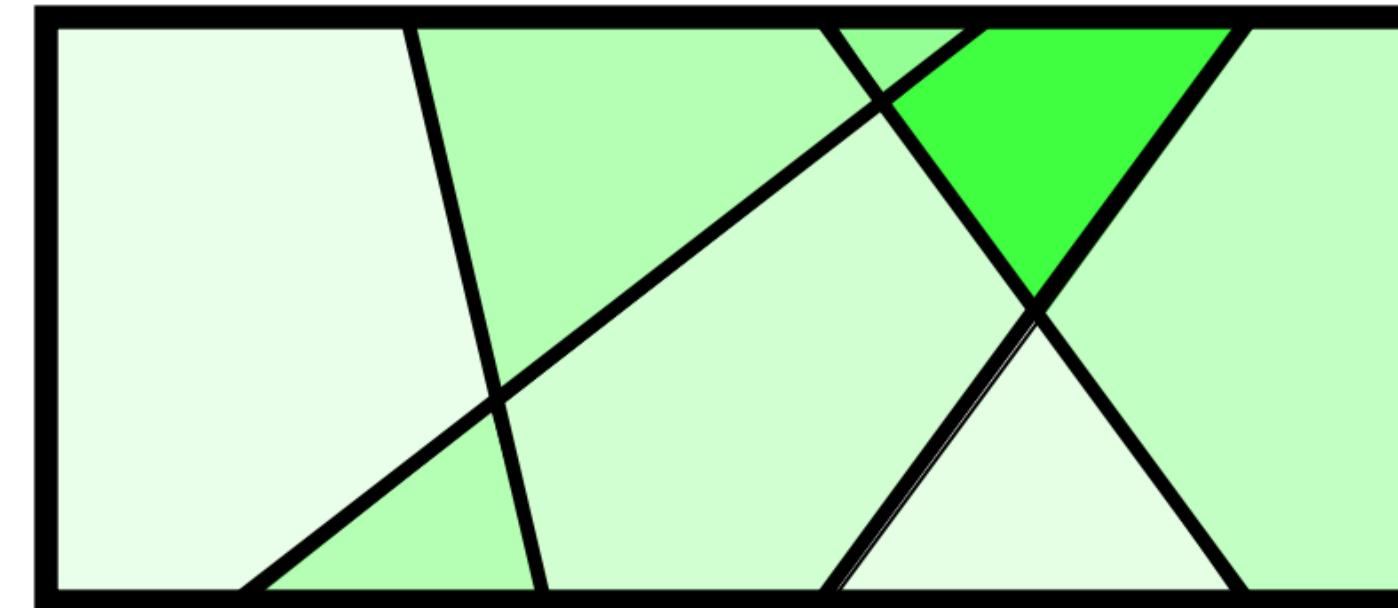
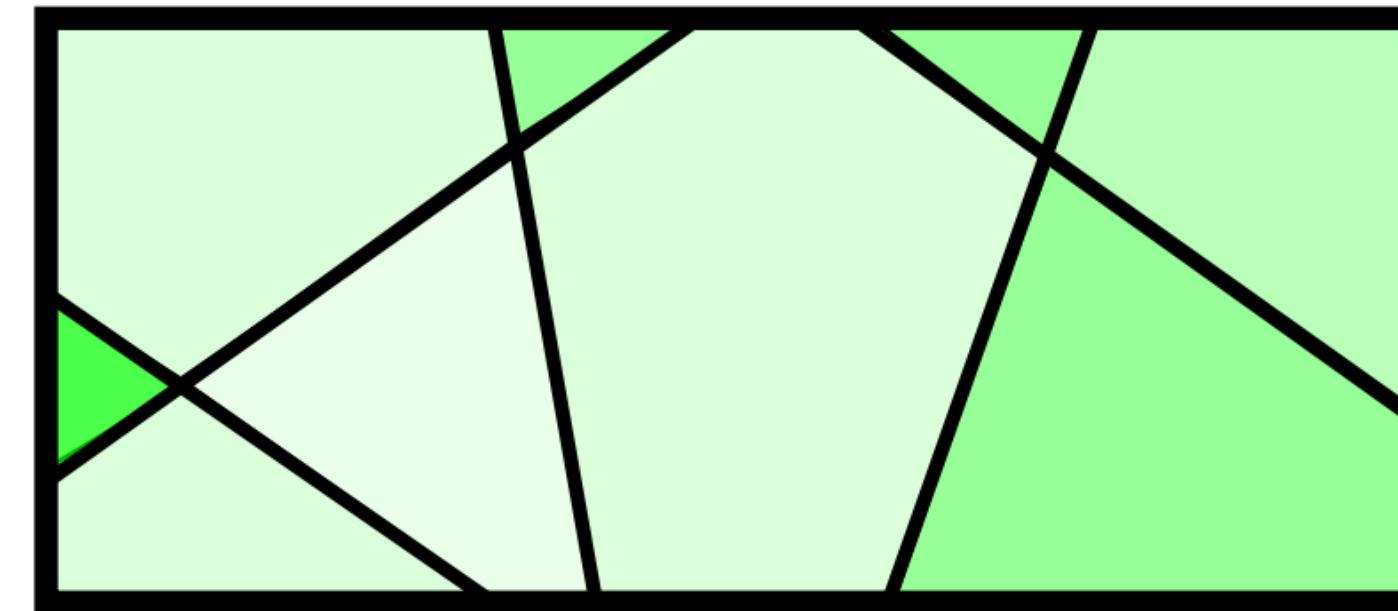
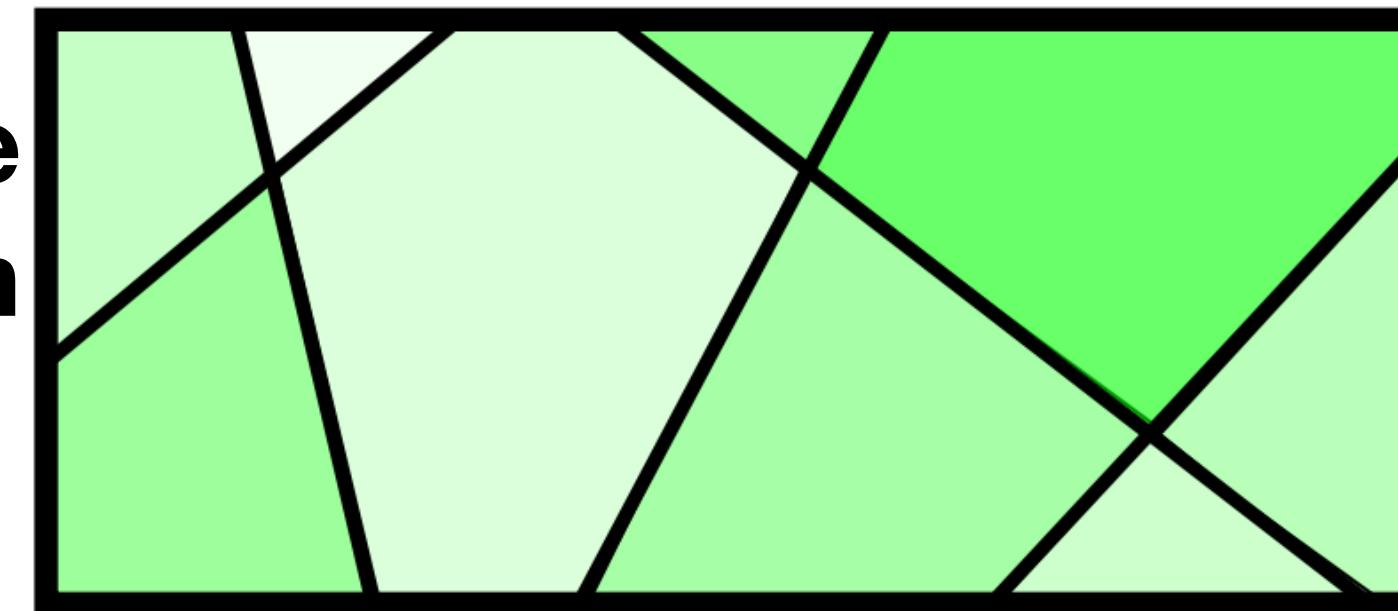
A Graphical Interpretation

Different trees induce different partitions on the data.



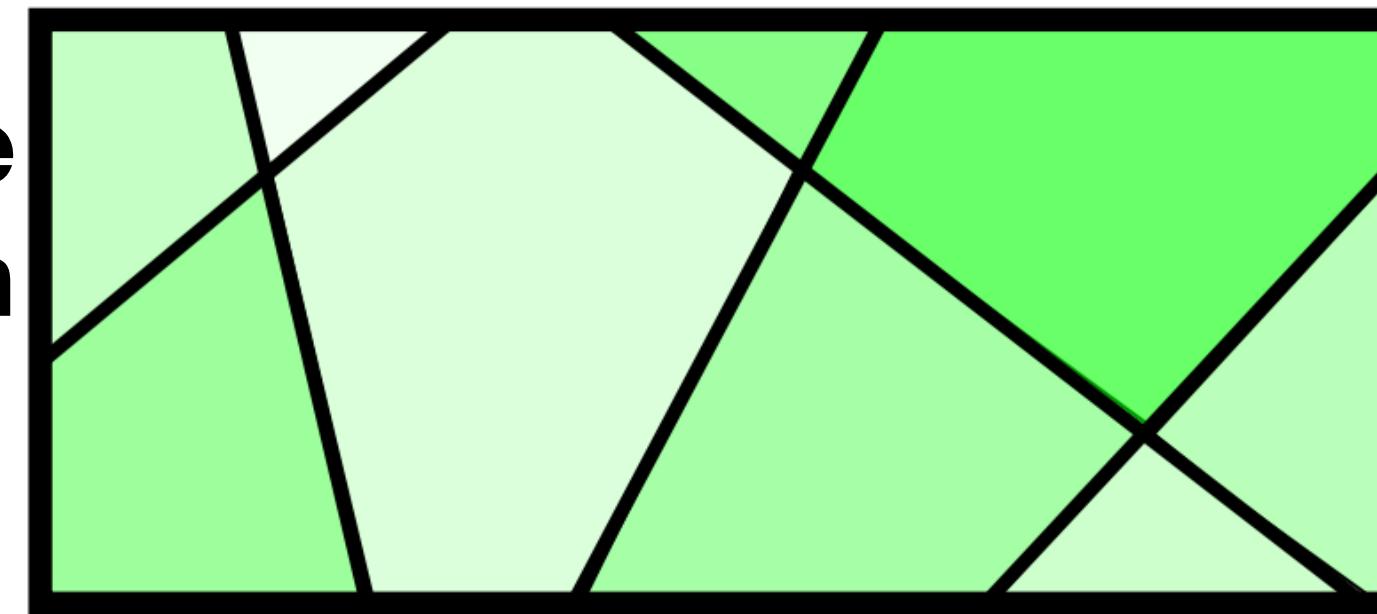
A Graphical Interpretation

Different trees induce different partitions on the data.

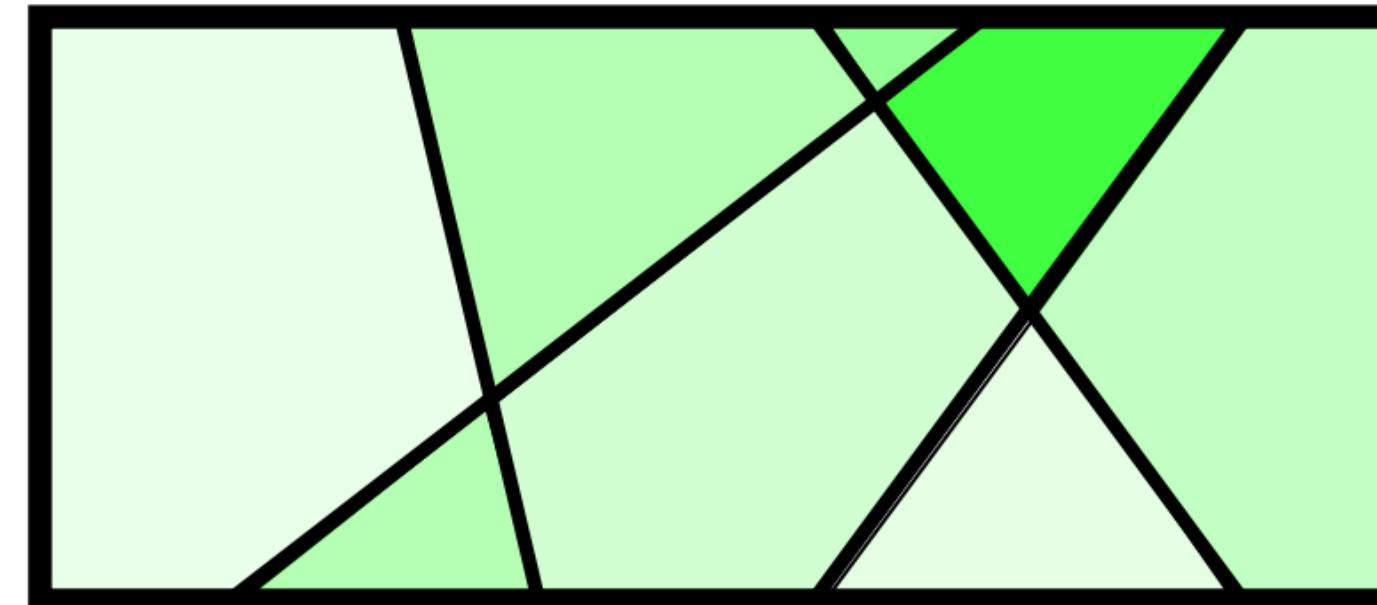
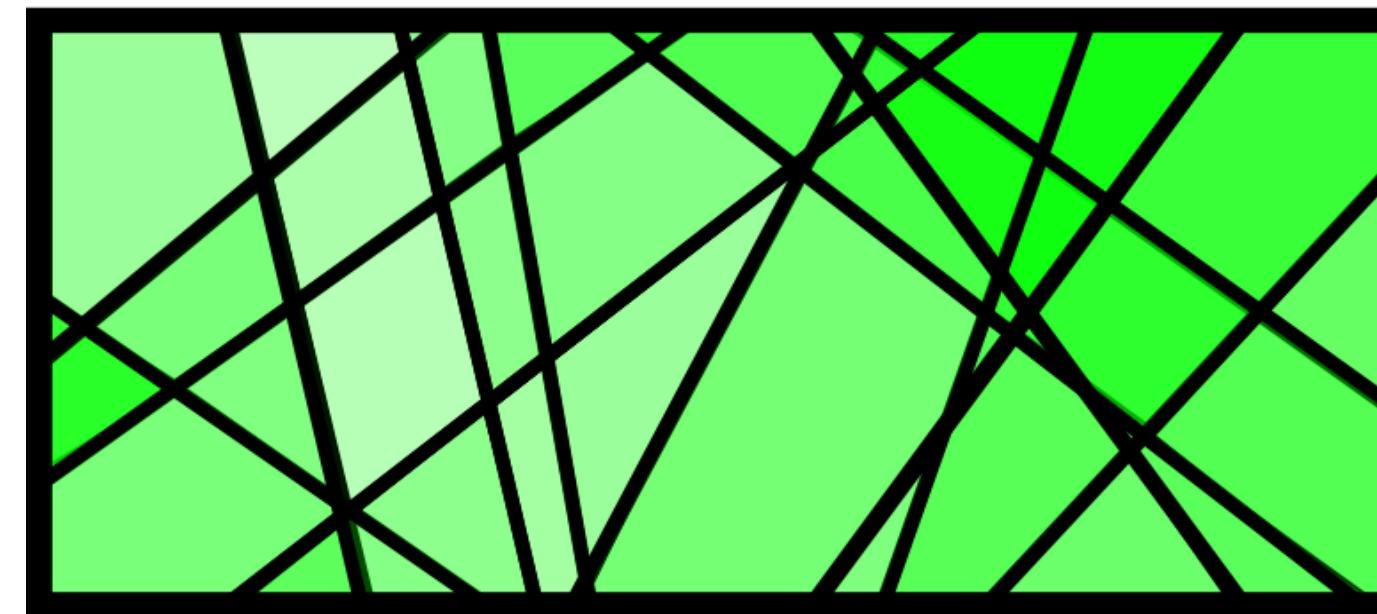


A Graphical Interpretation

Different trees induce different partitions on the data.



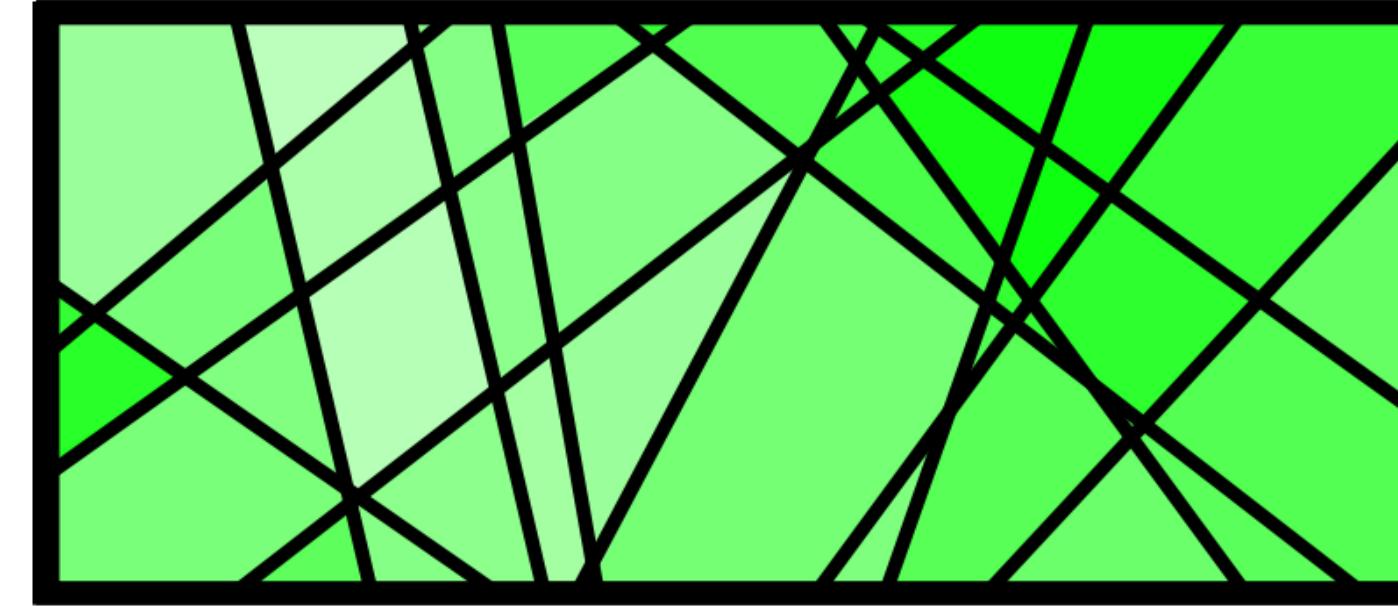
By combining them, we obtain a finer subdivision of the feature space...



A Graphical Interpretation

Different trees induce different partitions on the data.

By combining them, we obtain a finer subdivision of the feature space...



...which at the same time also better reflects the uncertainty due to the bootstrapped sampling.

Summary: Random Forests

Properties

- Very simple algorithm.
- Resistant to overfitting – generalizes well to new data.
- Faster training
- Extensions available for clustering, distance learning, etc.

Limitations

- Memory consumption
 - Decision tree construction uses much more memory.
- Well-suited for problems with little training data
 - Little performance gain when training data is really large.

Part 4, Video RandomForest_p4

Extremely randomized trees

- Random attribute selection

Today's Topics

Decision Trees

- Main concepts

Randomized Decision Trees

- Randomized attribute selection

Forests

- Bootstrap sampling
- Ensemble of randomized trees
- Posterior sum combination
- Analysis

Extremely randomized trees

- Random attribute selection

Ferns

- Fern structure
- Semi-Naïve Bayes combination

A Case Study in Deconstructivism...

What we've done so far

- Take the original decision tree idea.
- Throw out all the complicated bits (pruning, etc.).
- Learn on ***random subset*** of training data (bootstrapping/bagging).
- Select splits based on ***random choice*** of candidate queries.
 - So as to maximize information gain.
 - Complexity: $O(KN^2\log N)$

=> Ensemble of weaker classifiers.

How can we further simplify that?

- Main effort still comes from selecting the optimal split (from reduced set of options)...
- Simply choose a ***random query*** at each node.
 - Complexity: $O(N)$

=> ***Extremely randomized decision trees***

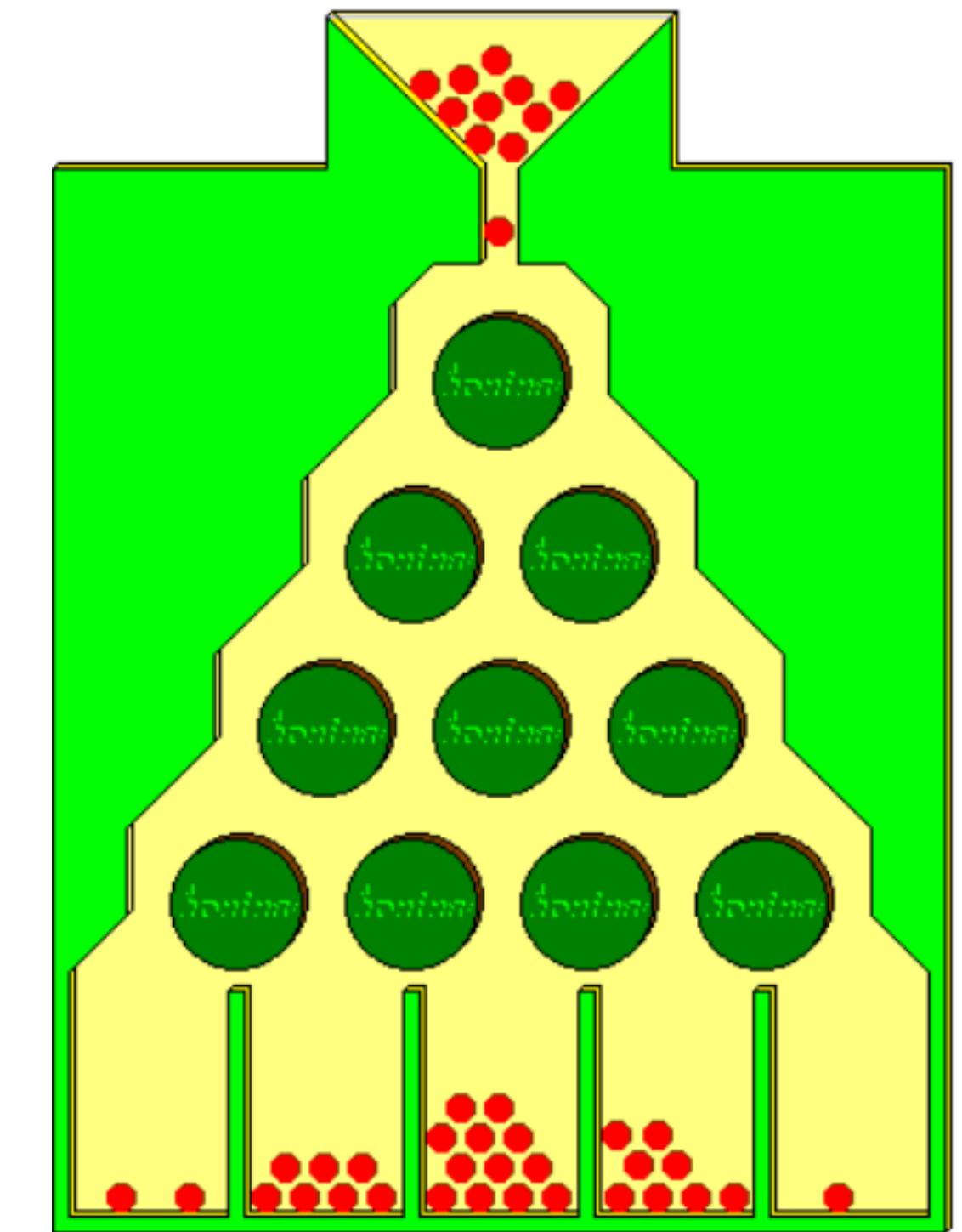
Extremely Randomized Decision Trees

Random queries at each node...

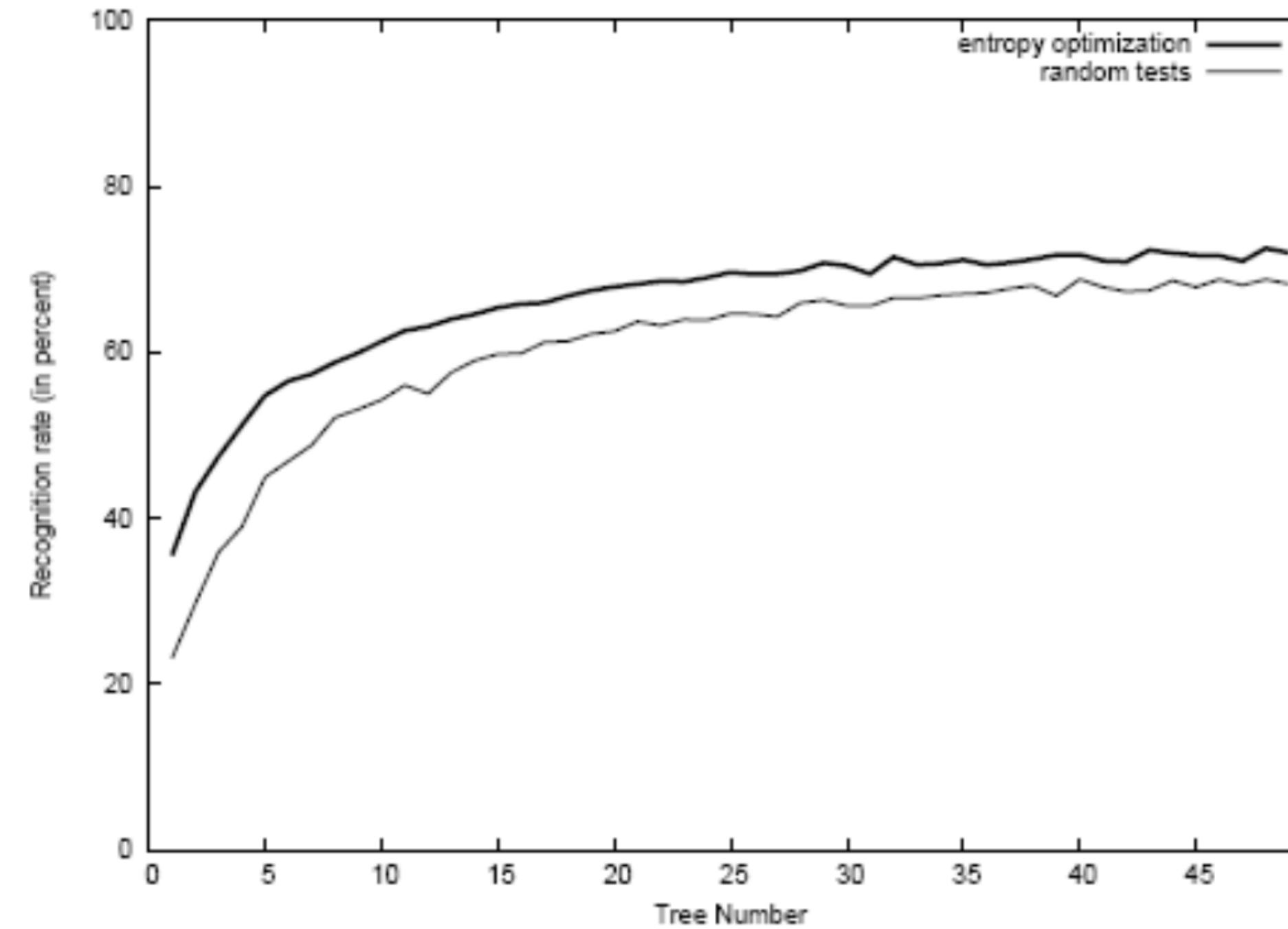
- Tree gradually develops from a classifier to a flexible container structure.
- Node queries define (randomly selected) structure.
- Each leaf node stores posterior probabilities

Learning (e.g. for keypoint detection)

- Patches are “dropped down” the trees.
 - Only pairwise pixel comparisons at each node.
 - Directly update posterior distributions at leaves
- => Very fast procedure, only few pixel-wise comparisons
- => No need to store the original patches!



Performance Comparison



**Keypoint
detection task**

Results

- Almost equal performance for random tests when a sufficient number of trees is available (and much faster to train!).

V. Lepetit, P. Fua, Keypoint Recognition using Randomized Trees, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 28(9), pp. 1465–1479, 2006.

Part 5, Video RandomForest_p5

Ferns

- Fern structure
- Semi-Naïve Bayes combination
- Applications

Today's Topics

Decision Trees

- Main concepts

Randomized Decision Trees

- Randomized attribute selection

Forests

- Bootstrap sampling
- Ensemble of randomized trees
- Posterior sum combination
- Analysis

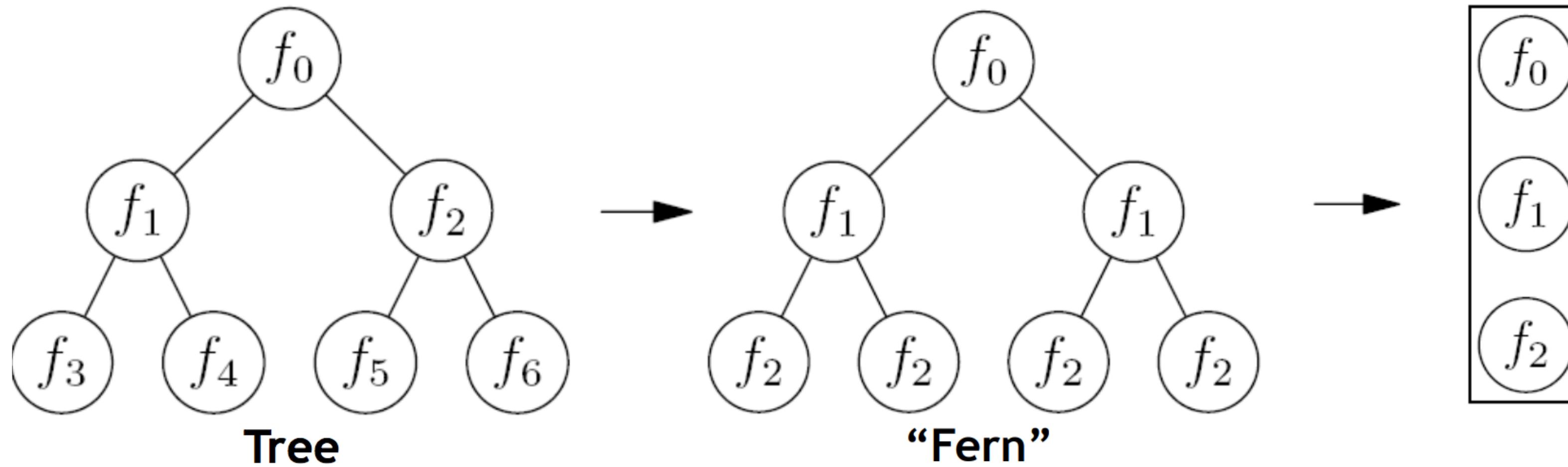
Extremely randomized trees

- Random attribute selection

Ferns

- Fern structure
- Semi-Naïve Bayes combination

From Trees to Ferns...



Observation

- If we select the node queries randomly anyway, what is the point of choosing different ones for each node?
=> Keep the same query for all nodes at a certain level.
- This effectively enumerates all $2M$ possible outcomes of the M tree queries.
- Tree can be collapsed into a fern-like structure.

What Does This Mean?

Interpretation of the decision tree

- We model the class conditional probabilities of a large number of binary features (the node queries).

- Notation
 - f_i : **Binary feature**
 - N_f : **Total number of features in the model**
 - \mathcal{C}_k : **Target class**

- Given f_1, \dots, f_{N_f} , we want to select class C_k such that

$$k = \arg \max_k p(C_k | f_1, \dots, f_{N_f})$$

- Assuming a uniform prior over classes, this is equal to

$$k = \arg \max_k p(f_1, \dots, f_{N_f} | C_k)$$

- Main issue: How do we model the joint distribution?

Modeling the Joint Distribution

Full Joint

- Model all correlations between features

$$p(f_1, \dots, f_{N_f} | C_k)$$

=> Model with 2^{N_f} parameter, not feasible to learn.

Naïve Bayes classifier

- Assumption: all features are independent.

$$p(f_1, \dots, f_{N_f} | C_k) = \prod_{i=1}^{N_f} p(f_i | C_k)$$

=> Too simplistic, assumption does not really hold!

=> Naïve Bayes model ignores correlation between features.

Modeling the Joint Distribution

Decision tree

- Each path from the root to a leaf corresponds to a specific combination of feature outcomes, e.g.

$$p_{leaf_m}(\mathcal{C}_k) = p(f_{m1} = 1, f_{m2} = 0, \dots, f_{md} = 1 | \mathcal{C}_k)$$

- Those path outcomes are independent, therefore

$$p(f_1, \dots, f_{N_f} | \mathcal{C}_k) \approx \prod_{m=1}^M p_{leaf_m}(\mathcal{C}_k)$$

- But not all feature outcomes are represented here...

Modeling the Joint Distribution

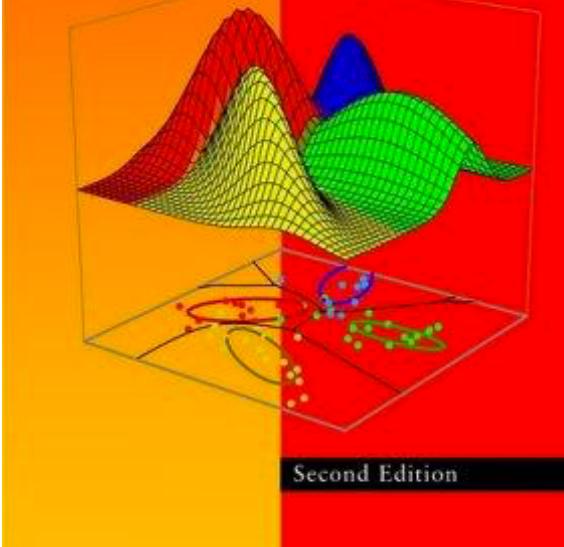
Ferns

- A fern F is defined as a set of S binary features $\{f_l, \dots, f_{l+S}\}$
- M : number of ferns, $N_f = S \cdot M$
- This represents a compromise:

$$\begin{aligned} p(f_1, \dots, f_{N_f} | \mathcal{C}_k) &\approx \prod_{j=1}^M p(F_j | \mathcal{C}_k) \\ &= \underbrace{p(f_1, \dots, f_S | \mathcal{C}_k)}_{\text{Full joint inside fern}} \cdot p(f_{S+1}, \dots, f_{2S} | \mathcal{C}_k) \cdot \dots \end{aligned}$$

=> Model with $M \cdot 2^S$ parameters (“Semi-Naïve”).

=> Flexible solution that allows complexity/performance tuning.



Readings

Duda and Hart's book (Decision Trees)

Chapter 8.2-8.4

The original papers for Randomized Trees

Y. Amit, D. Geman, Shape Quantization and Recognition with Randomized Trees, *Neural Computation*, Vol. 9(7), pp. 1545-1588, 1997.

V. Lepetit, P. Fua, KeypointRecognition using Randomized Trees, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 28(9), pp. 1465 – 1479, 2006.

The original paper for Random Forests:

L. Breiman, Random Forests, *Machine Learning*, Vol. 45(1), pp. 5-32, 2001.

The papers for Ferns:

M. Ozuysal, M. Calonder, V. Lepetit, P. Fua, Fast Keypoint Recognition using Random Ferns. In *IEEE. Trans. Pattern Analysis and Machine Intelligence*, 2009.

D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, D. Schmalstieg, Pose Tracking from Natural Features on Mobile Phones. In *ISMAR 2008*.

You should be able to answer ...

- What is Decision tree?
- What is CART?
- Properties and limitations of decision trees.
- What are difference between decision tree and randomised decision tree?
- Random Forests: Injecting Randomness.
- Properties and limitations of random forest.
- What is extremely randomized decision trees?
- What is Ferns?