Learning and Adaptivity

Randomized Trees, Forests, and Ferns - Lecture V

Course Outline

Basic Concepts

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

Supervised Learning, Classification Approaches

- Ensemble Methods and Boosting
- Randomized Trees, Forest

Unsupervised Learning

- Dimensionality Reduction and Manifold Learning (PCA, SNE/t-SNE, MDS, umap)
- Uncertainty Estimation

Reinforcement Learning

- Classical Reinforcement Learning
- Deep Reinforcement Learning

Today's Topics

Decision Trees

Randomized Decision Trees

- Randomized attribute selectionRandom

Forests

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

Extremely randomized trees

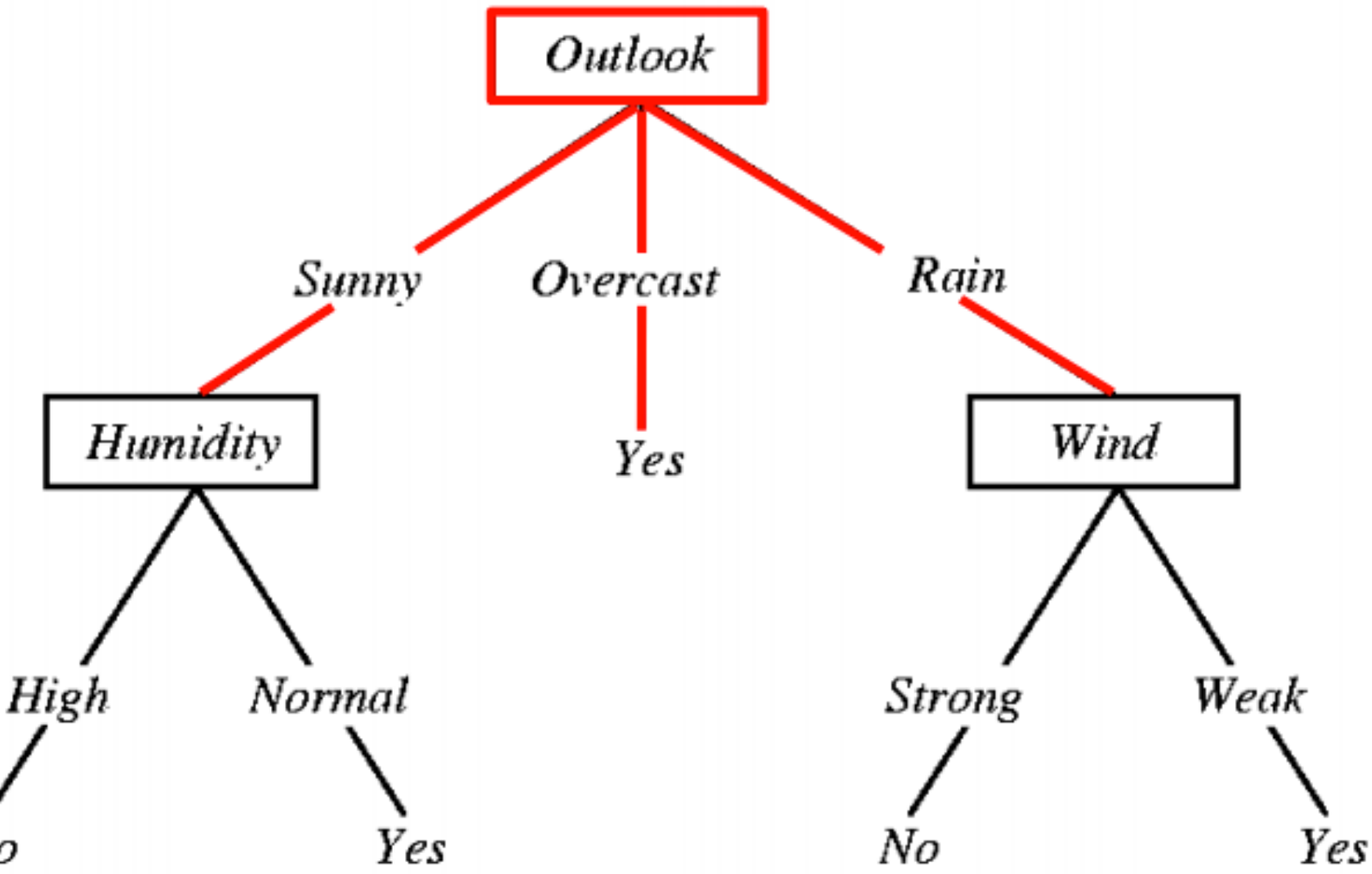
- Random attribute selection

Ferns

- Fern structure
- Semi-Naïve Bayes combination
- Applications

Slide source: B. Leibe Slides ML, 2015

Decision Trees

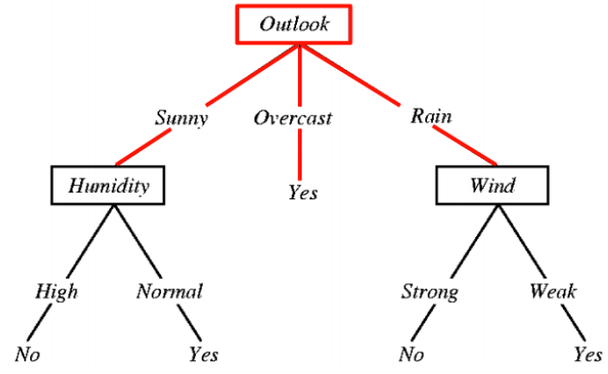


Elements

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

Image source: T. Mitchell, 1997

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!

Slide source: B. Leibe Slides ML, 2015

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.

Slide source: B. Leibe Slides ML, 2015

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

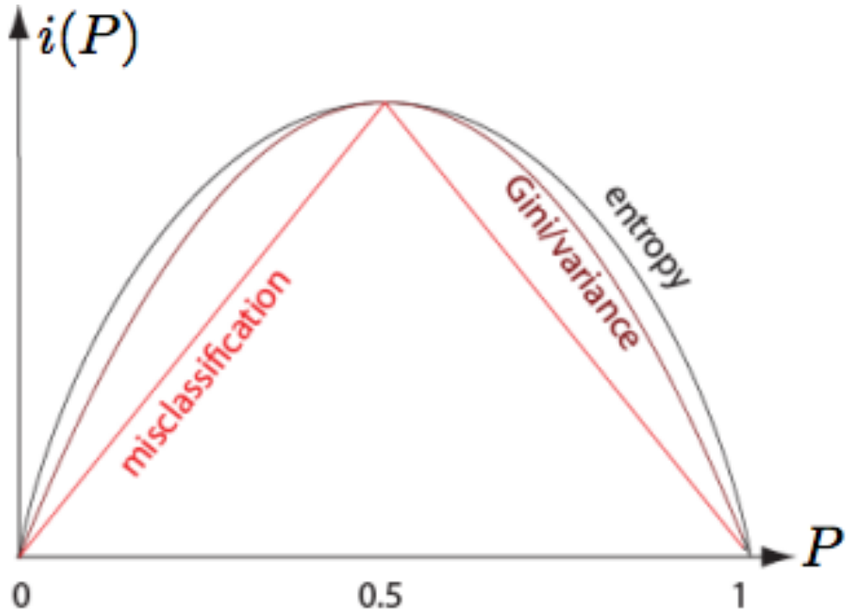
Impurity measures

- Entropy impurity (information gain):

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

- Gini impurity:

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



Slide source: B. Leibe Slides ML, 2015

Overfitting Prevention (Pruning)

Two basic approaches for decision trees

- **Prepruning**: Stop growing tree as some point during top-down construction when there is no longer sufficient data to make reliable decisions.
  - Cross-validation
  - Chi-square test
  - MDL

- **Postpruning**: Grow the full tree, then remove subtrees that do not have sufficient evidence.
  - Merging nodes
  - Rule-based pruning

In practice often preferable to apply post-pruning.

Slide source: B. Leibe Slides ML, 2015

Decision Trees – Properties

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

Slide source: B. Leibe Slides ML, 2015

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

Slide source: B. Leibe Slides ML, 2015

Decision Trees – Computational Complexity

Given

– Data points $\{\mathrm{x}_1, ..., \mathrm{x}_N\}$

– 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 $O(DN \log N)$

Slide source: B. Leibe Slides ML, 2015

Today's Topics

Decision Trees

Randomized Decision Trees

- Randomized attribute selectionRandom

Forests

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

Extremely randomized trees

- Random attribute selection

Ferns

- Fern structure
- Semi-Naïve Bayes combination
- Applications

Slide source: B. Leibe Slides ML, 2015

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):

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

Slide source: B. Leibe Slides ML, 2015

Randomized Decision Trees

Randomized splittingTraining runtime:

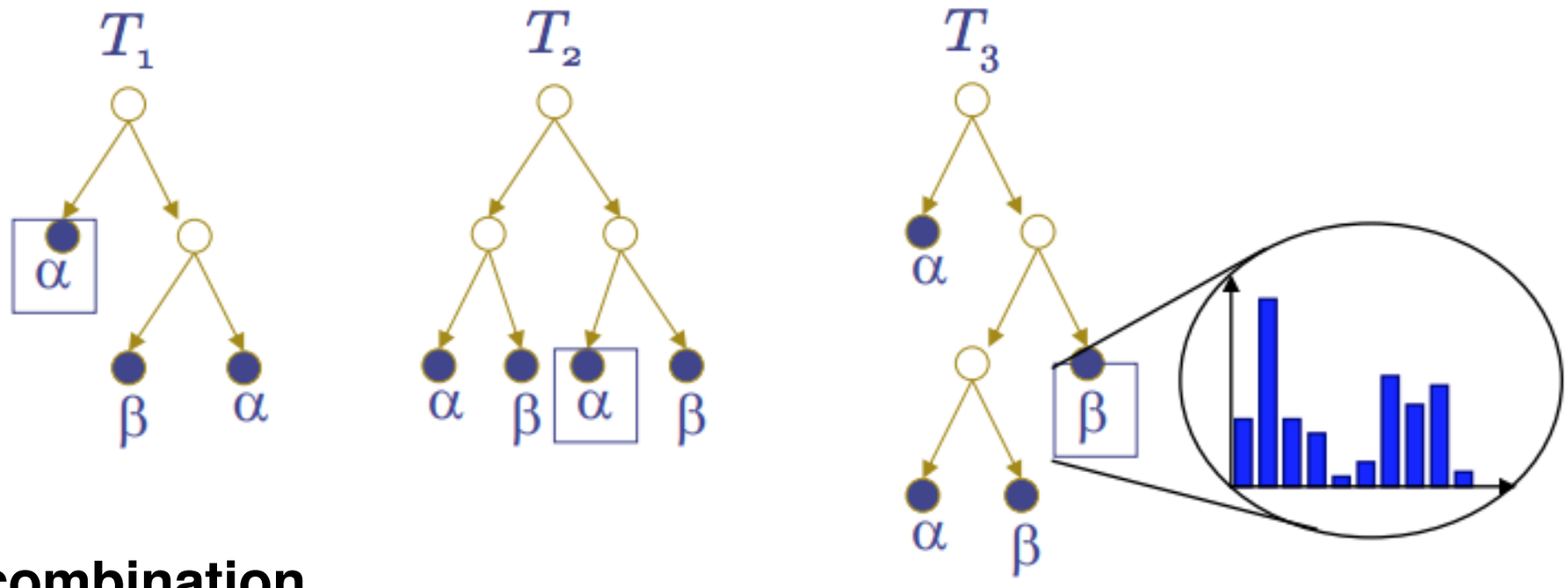
- Faster training: $O(KN^2 \log N)$ with $K \ll D$
- Use very simple binary feature tests.
- Typical choice
  - K = 10 for root node.
  - K = 100d for node at level d.

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.

Slide source: B. Leibe Slides ML, 2015

Ensemble Combination



Ensemble combination

- **Tree leaves (l, η) store posterior probabilities of the target classes.**

$$p_{l,\eta}(C|\mathrm{x})$$

- **Combine the output of several trees by averaging their posteriors (Bayesian model combination)**

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

Slide source: B. Leibe Slides ML, 2015

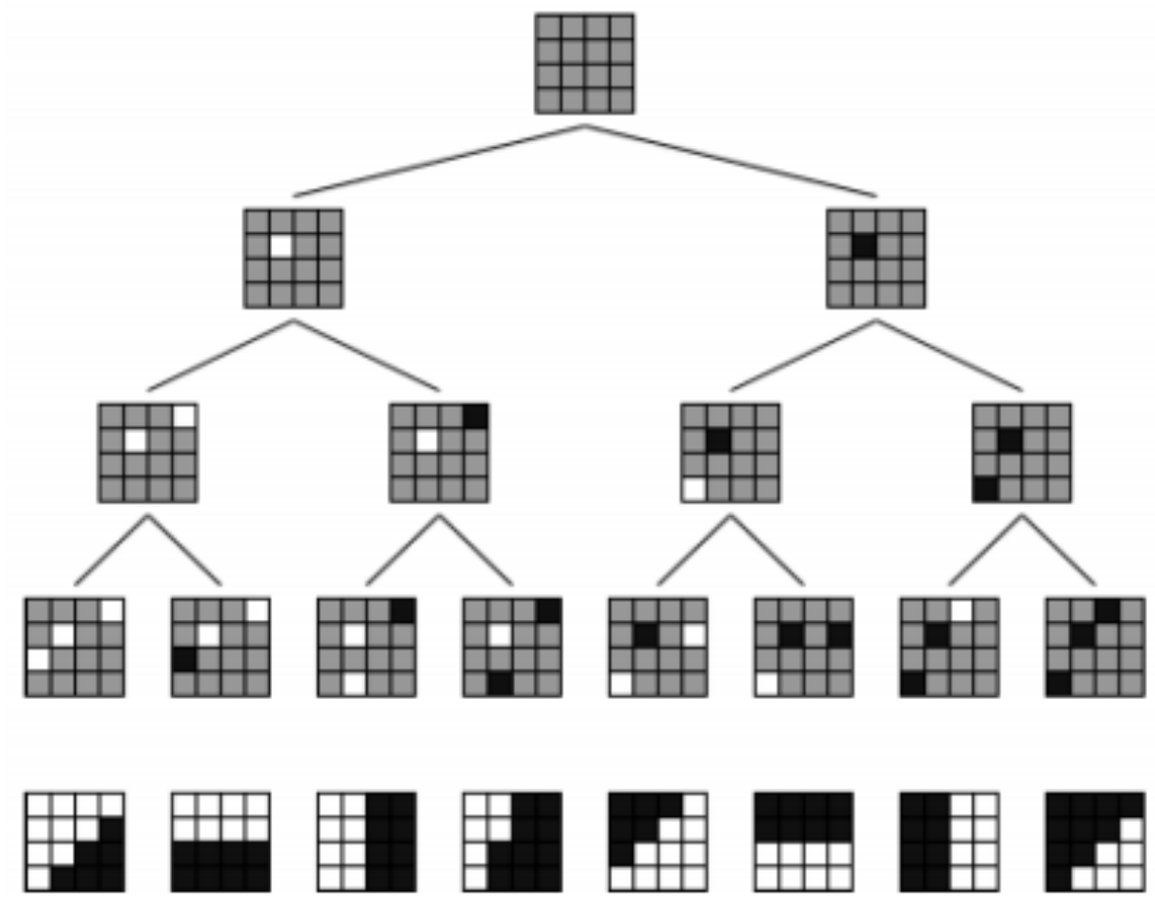
Ensemble Combination

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.



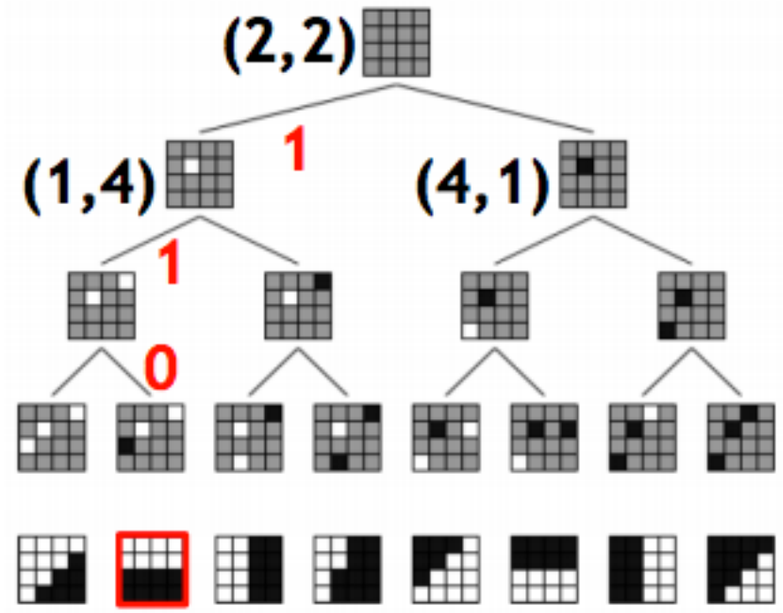
Y. Amit, D. Geman, Shape Quantization and Recognition with Randomized Trees, Neural Computation, Vol. 9(7), pp. 1545-1588, 1997.

Slide source: B. Leibe Slides ML, 2015

Applications: Character Recognition

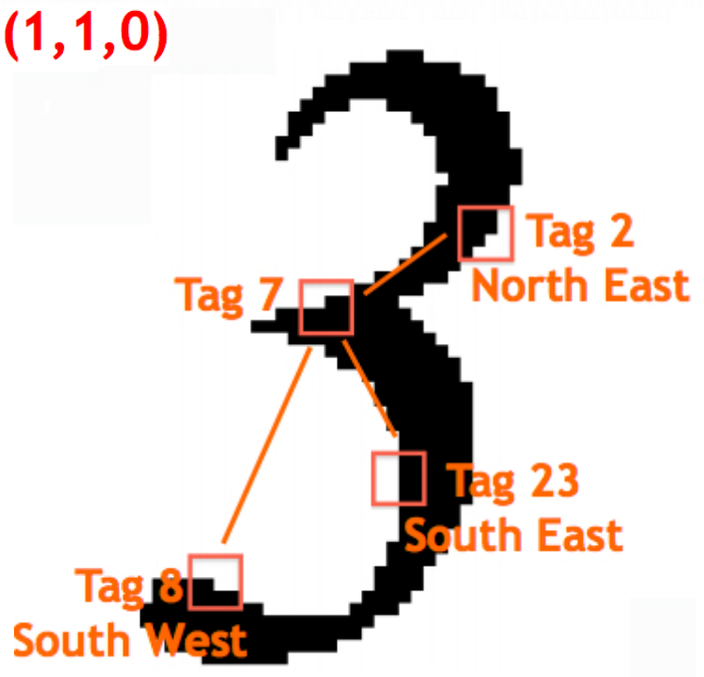
Image patches (“Tags”)

- 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



Slide source: B. Leibe Slides ML, 2015

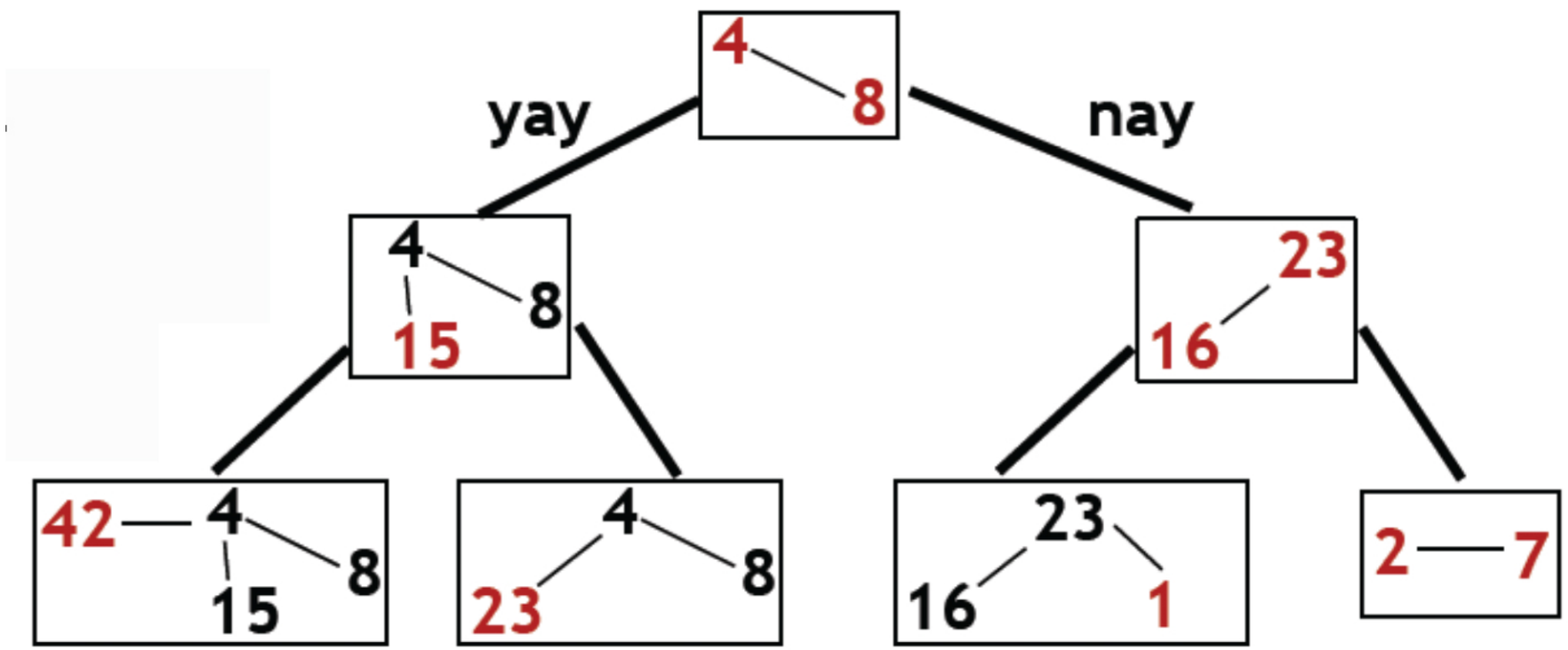
Applications: Character Recognition

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

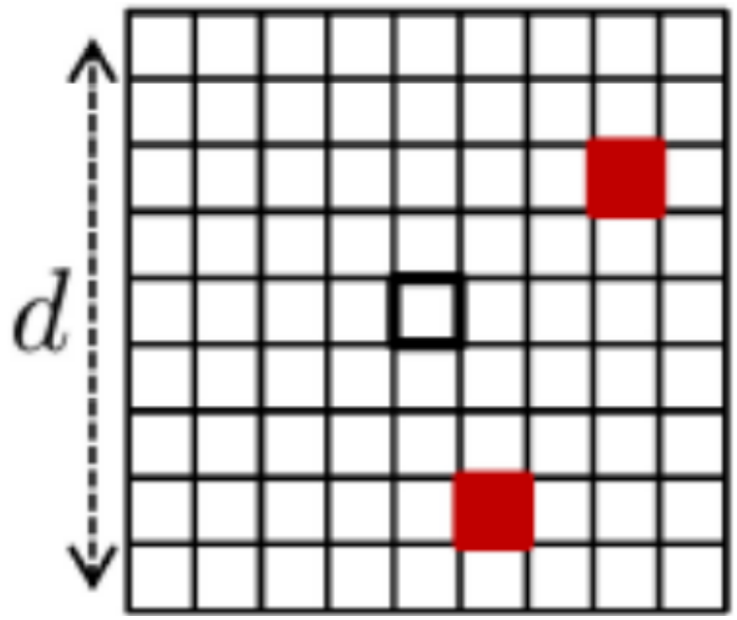


Slide source: B. Leibe Slides ML, 2015

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.

Slide source: B. Leibe Slides ML, 2015

Today's Topics

Decision Trees

Randomized Decision Trees

- Randomized attribute selectionRandom

Forests

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

Extremely randomized trees

- Random attribute selection

Ferns

- Fern structure
- Semi-Naïve Bayes combination
- Applications

Slide source: B. Leibe Slides ML, 2015

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”

Slide source: B. Leibe Slides ML, 2015

Random Forests – Algorithmic Goals

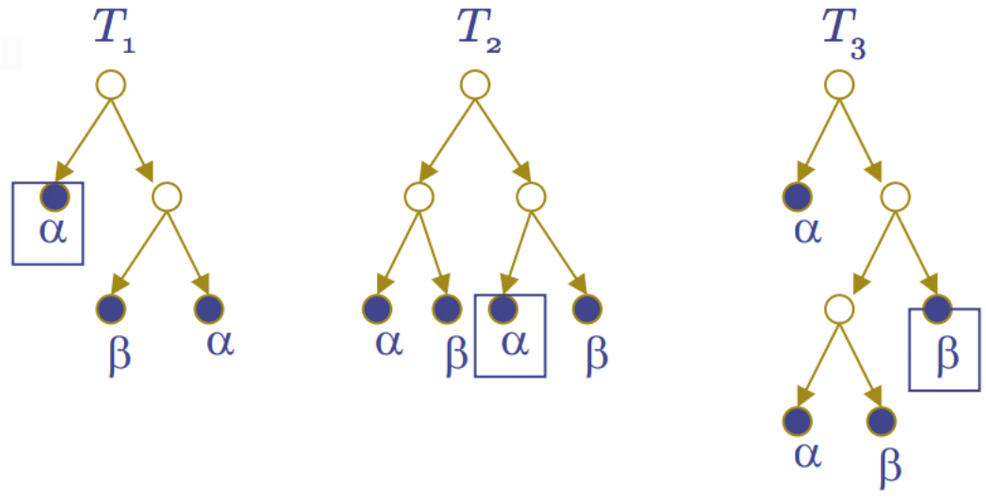
Create many trees (50 – 1,000)

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).

Slide source: B. Leibe Slides ML, 2015

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.

Slide source: B. Leibe Slides ML, 2015

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.

Slide source: B. Leibe Slides ML, 2015

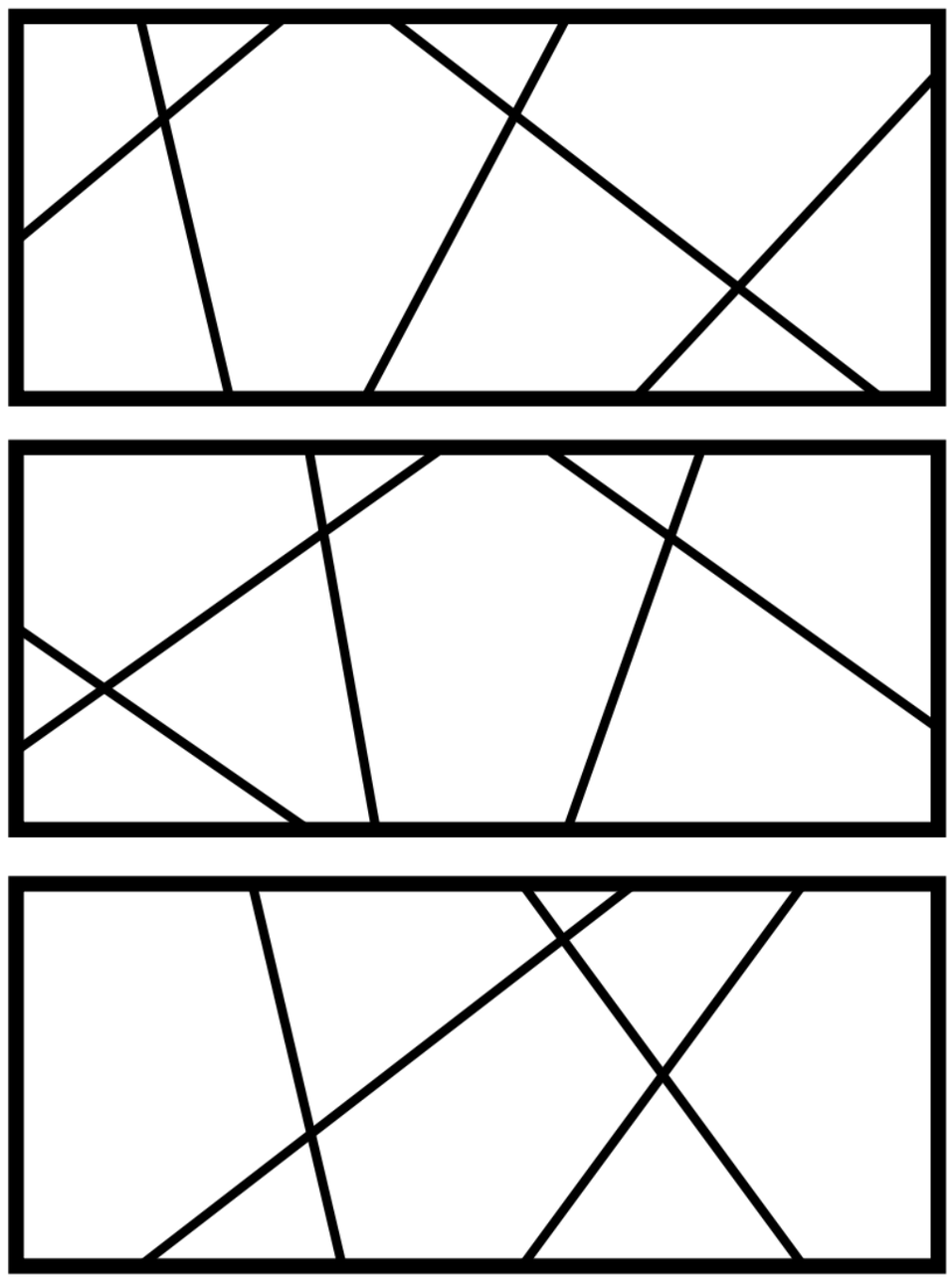
Bet You're Asking…

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

Slide source: B. Leibe Slides ML, 2015

A Graphical Interpretation

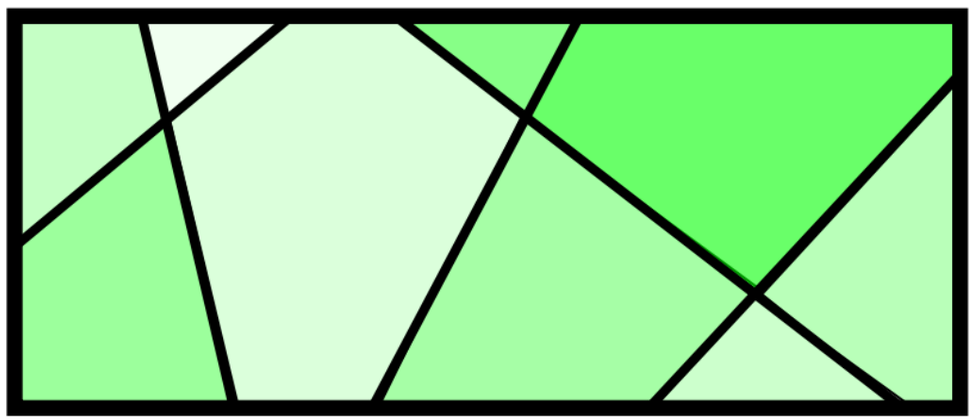
Different trees induce different partitions on the data.

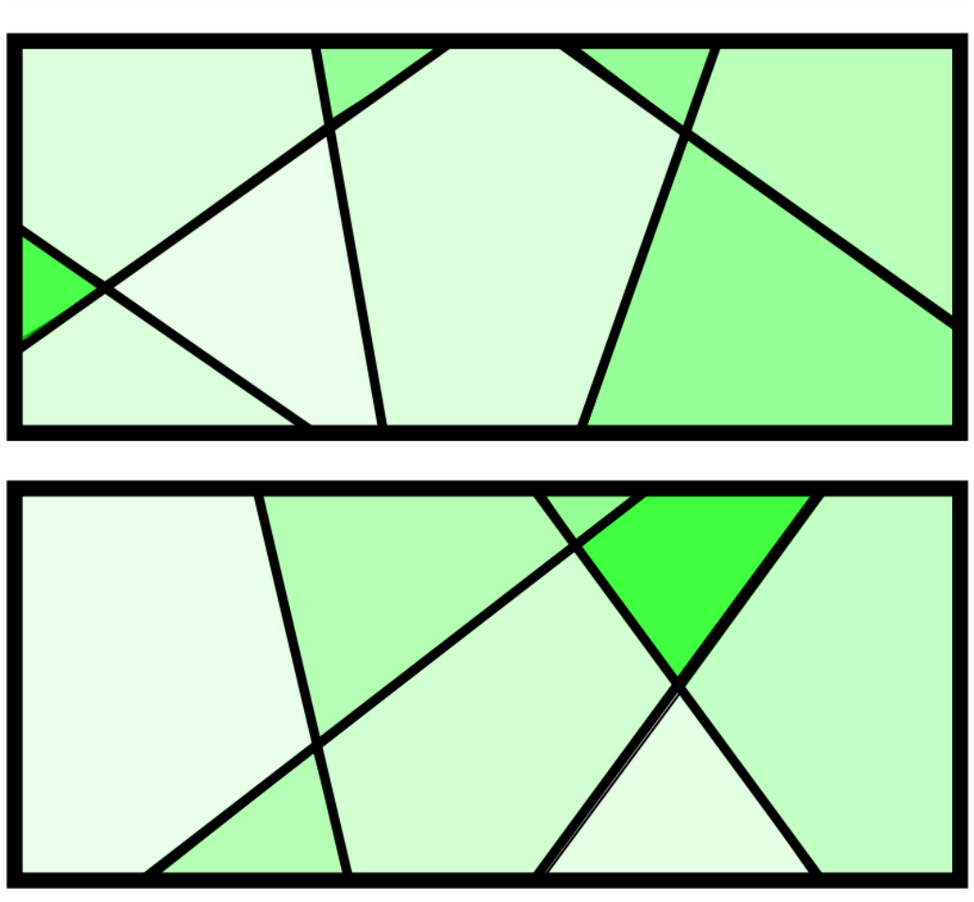


Slide source: B. Leibe Slides ML, 2015

A Graphical Interpretation

Different trees induce different partitions on the data.

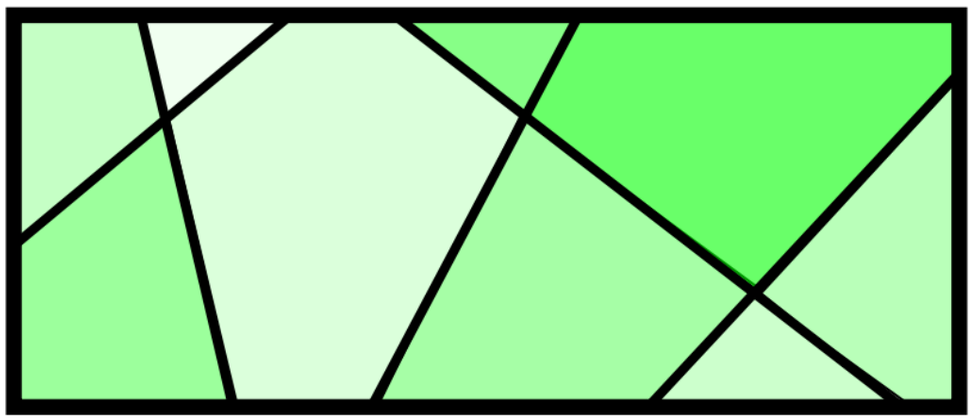




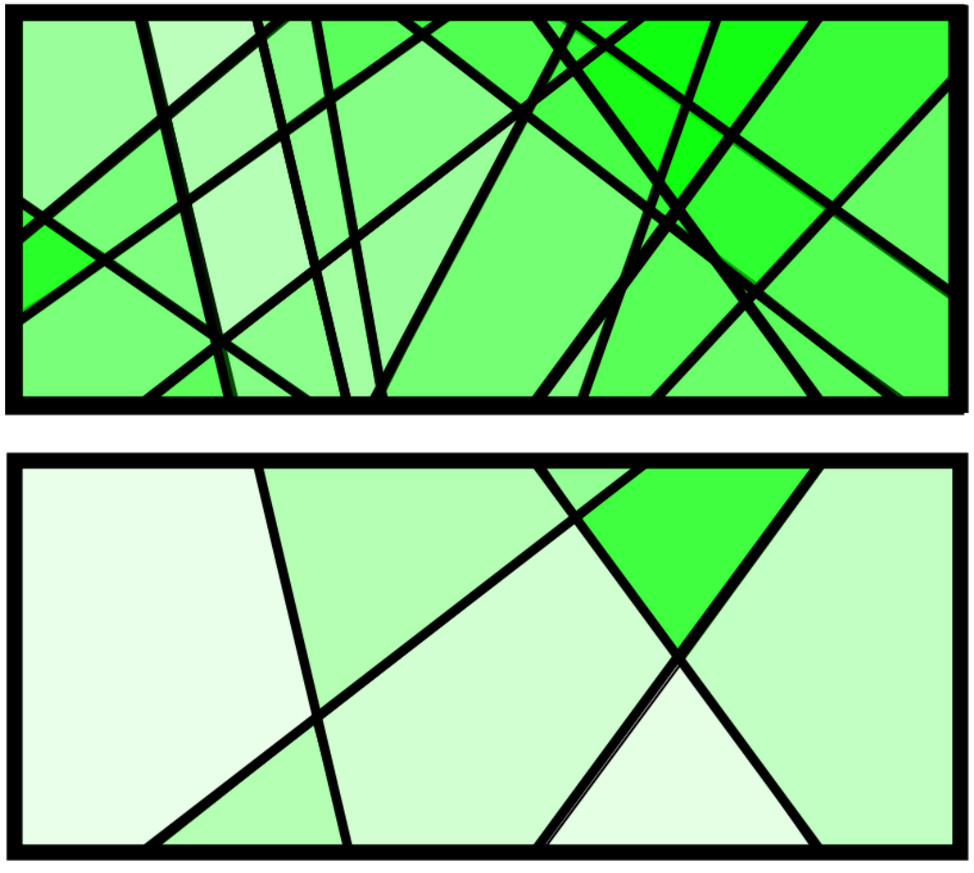
Slide source: B. Leibe Slides ML, 2015

A Graphical Interpretation

Different trees induce different partitions on the data.



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

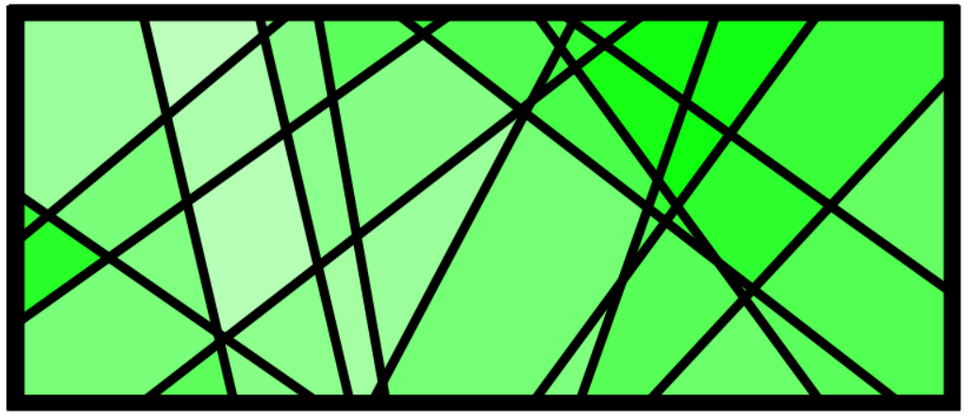


Slide source: B. Leibe Slides ML, 2015

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.

Slide source: B. Leibe Slides ML, 2015

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.

Slide source: B. Leibe Slides ML, 2015

Random Forests - Implementations

Original RF implementation by Breiman & Cutler

– http://www.stat.berkeley.edu/users/breiman/RandomForests/

– Papers, documentation, and code…

– …in Fortran 77.

Newer version available in Fortran 90!

– http://www.irb.hr/en/research/projects/it/2004/2004-111/

Fast Random Forest implementation for Java (Weka)

– http://code.google.com/p/fast-random-forest/

Random Forest implementation in Python

– library: sklearn

– functions: RandomForestClassifier, RandomForestRegressor

Slide source: B. Leibe Slides ML, 2015

Today's Topics

Decision Trees

Randomized Decision Trees

- Randomized attribute selectionRandom

Forests

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

Extremely randomized trees

- Random attribute selection

Ferns

- Fern structure
- Semi-Naïve Bayes combination
- Applications

Slide source: B. Leibe Slides ML, 2015

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: $O(N)$

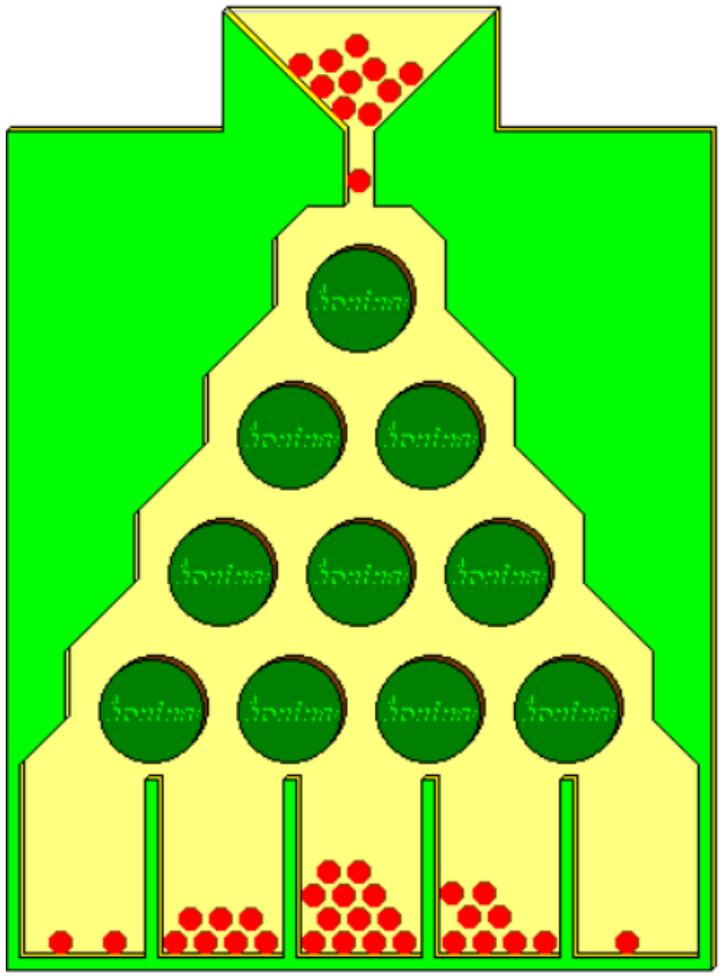
=> ***Extremely randomized decision trees***

Slide source: B. Leibe Slides ML, 2015

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)

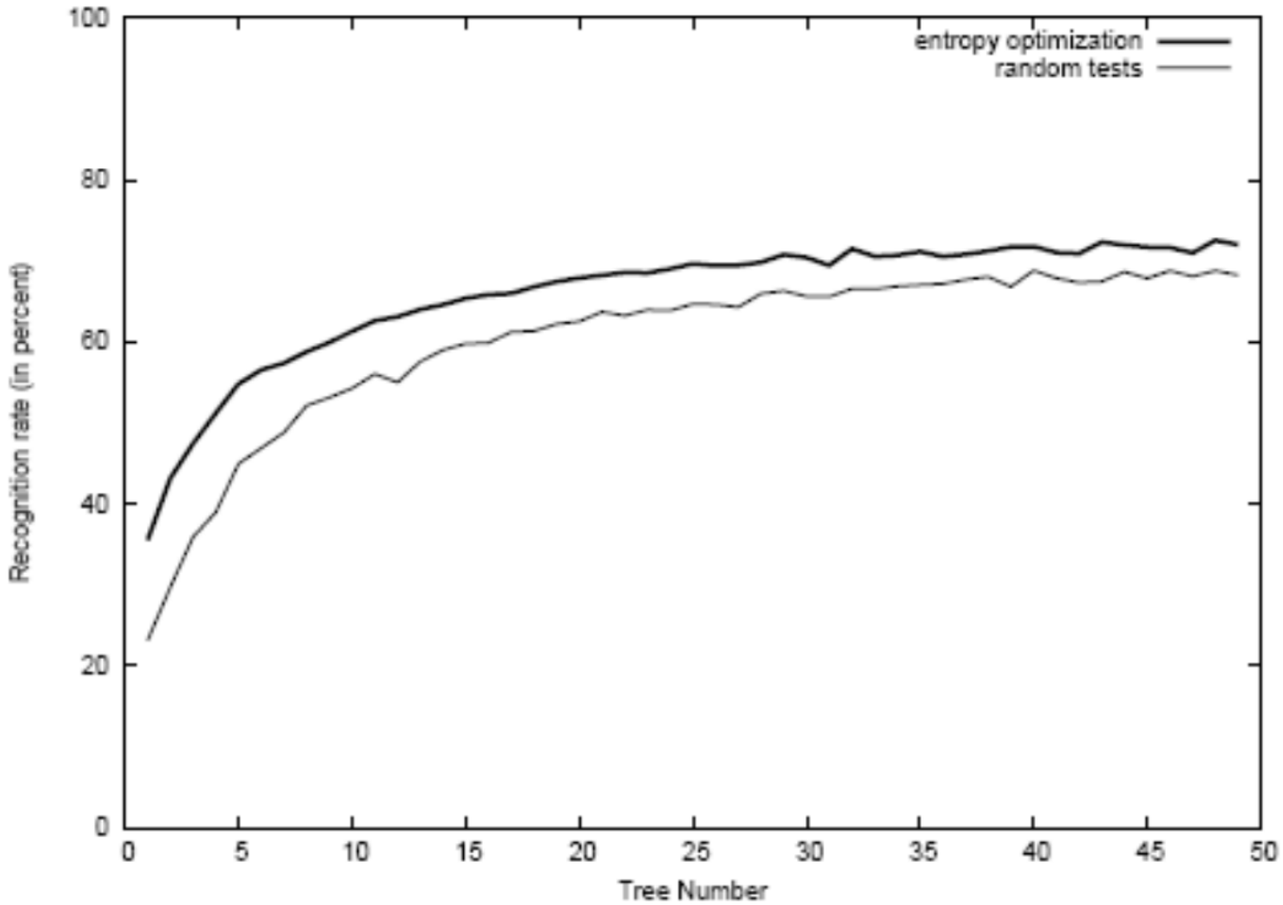
- Patches are “dropped down” the trees.
  - Only pairwise pixel comparisons at each node.
  - Directly update posterior distributions at leaves

=> Very fast procedure, only few pixel-wise comparisons

=> No need to store the original patches!

Slide source: B. Leibe Slides ML, 2015

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.

Slide source: B. Leibe Slides ML, 2015

Today's Topics

Decision Trees

Randomized Decision Trees

- Randomized attribute selectionRandom

Forests

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

Extremely randomized trees

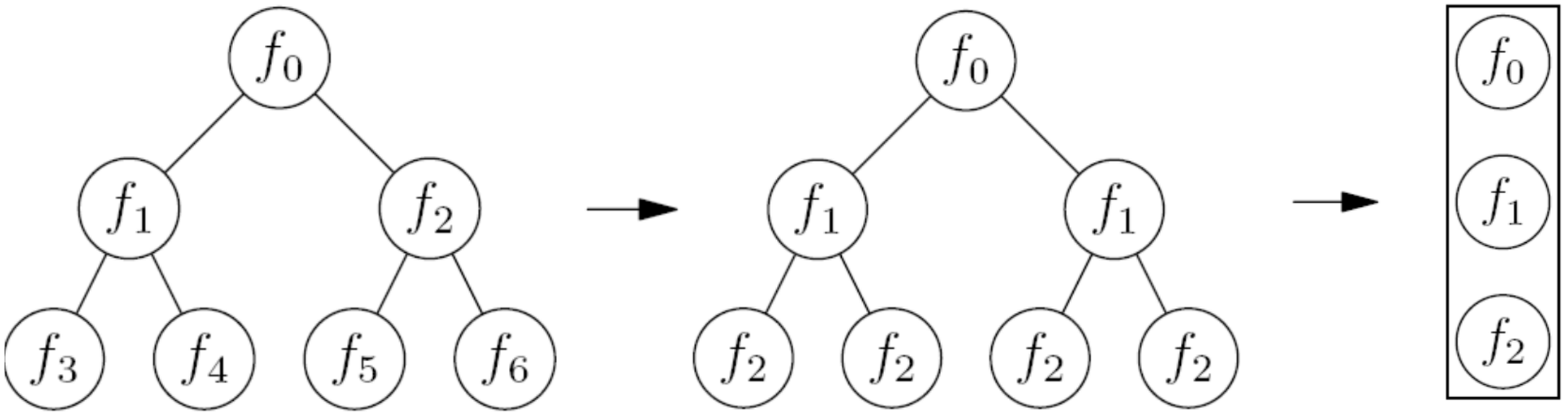
- Random attribute selection

Ferns

- Fern structure
- Semi-Naïve Bayes combination
- Applications

Slide source: B. Leibe Slides ML, 2015

From Trees to Ferns…



Tree **“Fern”**

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.

Slide source: B. Leibe Slides ML, 2015

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 $f_1, ..., f_{Nf}$, we want to select class C_k such that

$$k = \arg\max_k p(\mathcal{C}_k | f_1, \ldots, f_{N_f})$$

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

$$k = \arg\max_k p(f_1, \ldots, f_{N_f} | \mathcal{C}_k)$$

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

Slide source: B. Leibe Slides ML, 2015

Modeling the Joint Distribution

Full Joint

- Model all correlations between features

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

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

Naïve Bayes classifier

- Assumption: all features are independent.

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

=> Too simplistic, assumption does not really hold!

=> Naïve Bayes model ignores correlation between features.

Slide source: B. Leibe Slides ML, 2015

Modeling the Joint Distribution

Decision tree

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

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

- Those path outcomes are independent, therefore

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

- But not all feature outcomes are represented here…

Slide source: B. Leibe Slides ML, 2015

Modeling the Joint Distribution

Ferns

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

$$p(f_1,\ldots,f_{N_f}|C_k) \approx \prod_{j=1}^{M} p(F_j|C_k)$$

$$= \underbrace{p(f_1,\ldots,f_S|C_k)}_{\text{Full joint inside fern}} \cdot \overbrace{p(f_{S+1},\ldots,f_{2S}|C_k)}^{\text{Naïve Bayes between ferns}} \cdot \ldots$$

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

=> Flexible solution that allows complexity/performance tuning.

Slide source: B. Leibe Slides ML, 2015

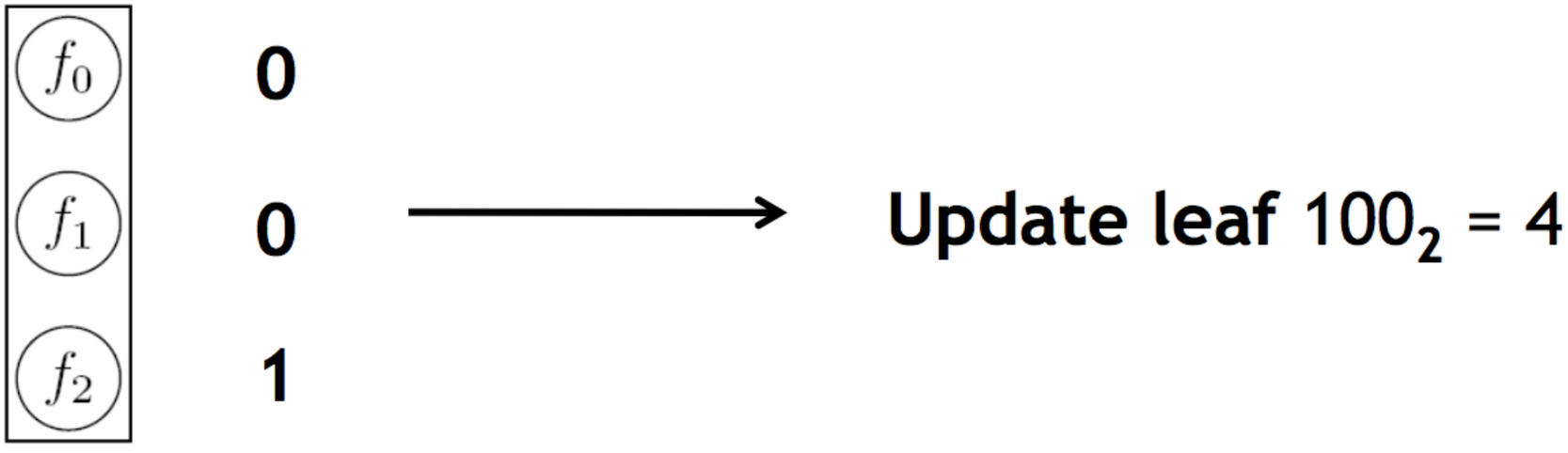
Modeling the Joint Distribution

Ferns

- Ferns are thus semi-naïve Bayes classifiers.
- They assume independence between sets of features (between the ferns)…
- …and enumerate all possible outcomes inside each set.

Interpretation

- Combine the tests $f_i, \ldots, f_{i+S}$ into a binary number.
- Update the "fern leaf" corresponding to that number.



Slide source: B. Leibe Slides ML, 2015