Topics of This Lecture

- Decision Trees
  - CART
  - Impurity measures, Stopping criterion, Pruning
  - Extensions, Issues
  - Historical development: ID3, C4.5
- Randomized Decision Trees
  - Randomized attribute selection
- Random Forests
  - Bootstrap sampling
  - Ensemble of randomized trees
  - Posterior sum combination
  - Analysis
- Extremely randomized trees
  - Random attribute selection

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  - Origin in the 60s, might seem outdated.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...

Example:

- "Classify Saturday mornings according to whether they’re suitable for playing tennis."
Decision Trees

- Assumption
  - Links must be mutually distinct and exhaustive
    - I.e. one and only one link will be followed at each step.
  - Interpretability
    - Information in a tree can then be rendered as logical expressions.
    - In our example:
      \[
      \begin{align*}
      \text{(Outlook} = \text{Sunny} \land \text{Humidity} = \text{Normal}) \\
      \lor (\text{Outlook} = \text{Overcast}) \\
      \lor (\text{Outlook} = \text{Rain} \land \text{Wind} = \text{Weak})
      \end{align*}
      \]

Training Decision Trees

- Finding the optimal decision tree is NP-hard...
- Common procedure: Greedy top-down growing
  - Start at the root node.
  - Progressively split the training data into smaller and smaller subsets.
  - In each step, pick the best attribute to split the data.
    - If the resulting subsets are pure (only one label) or if no further attribute can be found that splits them, terminate the tree.
    - Else, recursively apply the procedure to the subsets.
- CART framework
  - Classification And Regression Trees (Breiman et al. 1993)
  - Formalization of the different design choices.

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?

CART - 1. Number of Splits

- Each multi-valued tree can be converted into an equivalent binary tree:

\[ \Rightarrow \text{Only consider binary trees here...} \]

CART - 2. Picking a Good Splitting Feature

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

CART - Impurity Measures

- Misclassification impurity
  \[ i(N) = 1 - \max_j P(C_j | N) \]

Problem: discontinuous derivative!
CART - Impurity Measures

- Entropy impurity
  \[ i(N) = - \sum_j p(C_j|N) \log_2 p(C_j|N) \]
  “Reduction in entropy = gain in information.”

- Gini impurity (variance impurity)
  \[ i(N) = \sum_j p(C_j|N)p(C_j|N) = \frac{1}{2} \left[ 1 - \sum_j p^2(C_j|N) \right] \]
  “Expected error rate at node \( N \) if the category label is selected randomly.”

CART - Impurity Measures

- Which impurity measure should we choose?
  - Some problems with misclassification impurity.
    - Discontinuous derivative.
    - Problems when searching over continuous parameter space.
    - Sometimes misclassification impurity does not decrease when Gini impurity would.
  - Both entropy impurity and Gini impurity perform well.
    - No big difference in terms of classifier performance.
    - In practice, stopping criterion and pruning method are often more important.

CART - Picking a Good Splitting Feature

- For efficiency, splits are often based on a single feature
  - “Monothetic decision trees”

CART - 2. Picking a Good Splitting Feature

- Application
  - Select the query that decreases impurity the most
    \[ \Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R) \]
  - Multiway generalization (gain ratio impurity):
    - Maximize
      \[ \Delta i(s) = \frac{1}{Z} \left( i(N) - \sum_{k=1}^K P_k i(N_k) \right) \]
    - where the normalization factor ensures that large \( K \) are not inherently favored:
      \[ Z = - \sum_{k=1}^K P_k \log_2 P_k \]

CART - 3. When to Stop Splitting

- Problem: Overfitting
  - Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization to unseen data.
  - Reasons
    - Noise or errors in the training data.
    - Poor decisions towards the leaves of the tree that are based on very little data.

- Typical behavior
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.
- **Postpruning**: Grow the full tree, then remove subtrees that do not have sufficient evidence.

Label leaf resulting from pruning with the majority class of the remaining data, or a class probability distribution.

\[ C_N = \arg \max_k p(C_k|N) \]

\[ \hat{p}(C_i|N) \]

Benefits
- Avoids the horizon effect.
- Better use of training data (no hold-out set for cross-validation).

During training
- E.g. 3-dimensional data, one point is missing attribute \( x_i \).
  - Compute possible splits on \( x_i \) using all \( N \) points.
  - Compute possible splits on \( x_j \) using all \( N \) points.
  - Compute possible splits on \( x_j \) using \( N-1 \) non-deficient points.
  - Choose split which gives greatest reduction in impurity.

During test
- Cannot handle test patterns that are lacking the decision attribute!
  - In addition to primary split, store an ordered set of surrogate splits that try to approximate the desired outcome based on different attributes.

Decision Trees - Handling Missing Attributes

**CART - 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.
  - **Postpruning**: Grow the full tree, then remove subtrees that do not have sufficient evidence.
- Label leaf resulting from pruning with the majority class of the remaining data, or a class probability distribution.

\[ C_N = \arg \max_k p(C_k|N) \]

\[ \hat{p}(C_i|N) \]

Benefits
- Avoids the horizon effect.
- Better use of training data (no hold-out set for cross-validation).

**CART - Stopping Criterion**
- Determining which subtrees to prune:
  - **Cross-validation**: Reserve some training data as a hold-out set (validation set, tuning set) to evaluate utility of subtrees.
  - **Statistical test**: Determine if any observed regularity can be dismisses as likely due to random chance.
    - Determine the probability that the outcome of a candidate split could have been generated by a random split.
    - Chi-squared statistic (one degree of freedom)
      \[ X^2 = \sum_{i=1}^k \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i} \]
      "expected number from random split"
    - Compare to critical value at certain confidence level (table lookup).
  - **Minimum description length (MDL)**: Determine if the additional complexity of the hypothesis is less complex than just explicitly remembering any exceptions resulting from pruning.

**CART - 4. (Post-)Pruning**
- Stopped splitting often suffers from "horizon effect"
  - Decision for optimal split at node \( N \) is independent of decisions at descendent nodes.
    - Might stop splitting too early.
    - Stopped splitting biases learning algorithm towards trees in which the greatest impurity reduction is near the root node.
- Often better strategy
  - Grow tree fully (until leaf nodes have minimum impurity).
  - Then prune away subtrees whose elimination results only in a small increase in impurity.
- Benefits
  - Avoids the horizon effect.
  - Better use of training data (no hold-out set for cross-validation).

**Post-)Pruning Strategies**
- **Common strategies**
  - **Merging leaf nodes**
    - Consider pairs of neighboring leaf nodes.
    - If their elimination results only in small increase in impurity, prune them.
    - Procedure can be extended to replace entire subtrees with leaf node directly.
  - **Rule-based pruning**
    - Each leaf has an associated rule (conjunction of individual decisions).
    - Full tree can be described by list of rules.
    - Can eliminate irrelevant preconditions to simplify the rules.
    - Can eliminate rules to improve accuracy on validation set.
    - Advantage: can distinguish between the contexts in which the decision rule at a node is used and can prune them selectively.

Decision Trees - Feature Choice
- **Best results if proper features are used**
Decision Trees - Feature Choice

- Best results if proper features are used
  - Preprocessing to find important axes often pays off.

Decision Trees - Non-Uniform Cost

- Incorporating category priors
  - Often desired to incorporate different priors for the categories.
  - Solution: weight samples to correct for the prior frequencies.

- Incorporating non-uniform loss
  - Create loss matrix $\lambda_{ij}$
  - Loss can easily be incorporated into Gini impurity
    $$i(N) = \sum_{ij} \lambda_{ij} p(C_i)p(C_j)$$

Historical Development

- ID3 (Quinlan 1986)
  - One of the first widely used decision tree algorithms.
  - Intended to be used with nominal (unordered) variables
    - Real variables are first binned into discrete intervals.
  - General branching factor
    - Use gain ratio impurity based on entropy (information gain) criterion.

- Algorithm
  - Select attribute $a$ that best classifies examples, assign it to root.
  - For each possible value $v_i$ of $a$,
    - Add new tree branch corresponding to test $a = v_i$.
    - If example_list($v_i$) is empty, add leaf node with most common label in example_list($a$).
    - Else, recursively call ID3 for the subtree with attributes $A \setminus a$.

Decision Trees - Computational Complexity

- Given
  - Data points $[x_1, \ldots, x_N]$
  - 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. $O(DN \log N)$

Summary: Decision Trees

- Properties
  - Simple learning procedure, fast evaluation.
  - Can be applied to metric, nominal, or mixed data.
  - Often yield interpretable results.
Summary: 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!
  - Expensive learning step
    - Mostly due to costly selection of optimal split.

References and Further Reading

- More information on Decision Trees can be found in Chapters 8.2-8.4 of Duda & Hart.
- The original paper for Random Forests:

Topics of This Lecture

- Decision Trees
  - CART
  - Impurity measures, Stopping criterion, Pruning
  - Extensions, Issues
  - Historical development: ID3, C4.5
- Randomized Decision Trees
  - Randomized attribute selection
- Random Forests
  - Bootstrap sampling
  - Ensemble of randomized trees
  - Posterior sum combination
- Analysis
- Extremely randomized trees
  - Random attribute selection

Randomized Decision Trees (Amit & Geman 1997)

- Decision trees: main effort on finding good split
  - Training runtime: \( O(KN^2 \log N) \) with \( K \ll D \).
  - 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):
    \[
    \Delta E = \sum_{k=1}^{K} \left[ S_k - \frac{N}{L} \sum_{j=1}^{L} p_j \log_2(p_j) \right]
    \]

Ensemble Combination

- Ensemble combination
  - Tree leaves \((i,j)\) store posterior probabilities of the target classes.
    \[
    p_{i,j}(C|x)
    \]
  - Combine the output of several trees by averaging their posteriors (Bayesian model combination)
    \[
    p(C|x) = \frac{1}{L} \sum_{i=1}^{L} \sum_{j=1}^{L} p_{i,j}(C|x)
    \]
Applications: Character Recognition

- **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.


Slide adapted from Jan Hosang

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?

**B. Leibe**

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 adapted from Jan Hosang

Topics of This Lecture

- **Decision Trees**
  - CART
  - Impurity measures, Stopping criterion, Pruning
  - Extensions, Issues
  - Historical development: ID3, C4.5

- **Randomized Decision Trees**
  - Randomized attribute selection

- **Random Forests**
  - Bootstrap sampling
  - Ensemble of randomized trees
  - Posterior sum combination
  - Analysis

- **Extremely randomized trees**
  - Random attribute selection

**B. Leibe**
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 reweight 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 \( K = \sqrt{N} \)).
  - 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.

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...

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.

You Can Try It At Home...

- Free implementations available
  - Original RF implementation by Breiman & Cutler
    - http://www.stat.berkeley.edu/users/breiman/RandomForests/
    - Papers, documentation, and code...
      - ...in Fortran 77.
  - But also newer version available in Fortran 90!
  - Fast Random Forest implementation for Java (Weka)
    - http://code.google.com/p/fast-random-forest/


Topics of This Lecture

- Decision Trees
  - CART
  - Impurity measures, Stopping criterion, Pruning
  - Extensions, Issues
  - Historical development: ID3, C4.5

- Randomized Decision Trees
  - Randomized attribute selection

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

- Extremely randomized trees
  - Random attribute selection

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: \( O(N) \)
    - \( \Rightarrow \) 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
  - 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

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


Application: Keypoint Matching

Application: Mobile Augmented Reality

Practical Issues - Selecting the Tests

- For a small number of classes
  - We can try several tests.
  - Retain the best one according to some criterion.
    - E.g. entropy, Gini

- When the number of classes is large
  - Any test does a decent job.

Summary

- We started from full decision trees...
  - Successively simplified the classifiers...

- ...and ended up with very simple randomized versions
  - Ensemble methods: Combination of many simple classifiers
  - Good overall performance
  - Very fast to train and to evaluate

- Common limitations of Randomized Trees and Ferns?
  - Need large amounts of training data!
  - In order to fill the many probability distributions at the leaves.
  - Memory consumption?
    - Linear in the number of trees.
    - Exponential in the tree depth.
    - Linear in the number of classes (histogram at each leaf!)
References and Further Reading

• Very recent topics, not covered sufficiently well in books yet...

• The original papers for Randomized Trees

• The original paper for Random Forests:

• The papers for Ferns: