Recap: Stacking

- **Idea**
  - Learn $L$ classifiers (based on the training data)
  - Find a meta-classifier that takes as input the output of the $L$ first-level classifiers.
  - Example
    - Learn $L$ classifiers with leave-one-out.
    - Interpret the prediction of the $L$ classifiers as $L$-dimensional feature vector.
    - Learn “level-2” classifier based on the examples generated this way.

- **Example**
  - Learn $L$ classifiers with leave-one-out.
  - Interpret the prediction of the $L$ classifiers as $L$-dimensional feature vector.
  - Learn “level-2” classifier based on the examples generated this way.

Recap: Bayesian Model Averaging

- **Model Averaging**
  - Suppose we have $H$ different models $h = 1, \ldots, H$ with prior probabilities $p(h)$.
  - Construct the marginal distribution over the data set
    \[
    p(X) = \sum_{h=1}^{H} p(X|h)p(h)
    \]
  - **Average error of committee**
    \[
    E_{\text{COM}} = \frac{1}{M} E_{\text{AV}}
    \]
    - This suggests that the average error of a model can be reduced by a factor of $M$ simply by averaging $M$ versions of the model.
    - Unfortunately, this assumes that the errors are all uncorrelated.
    - In practice, they will typically be highly correlated.

Recap: Boosting (Schapire 1989)

- **Algorithm:** (3-component classifier)
  1. Sample $N < N$ training examples (without replacement) from training set $D$ to get set $D_1$.
  2. Sample $N < N$ training examples (without replacement), half of which were misclassified by $C_1$, to get set $D_2$.
  3. Choose all data in $D_1$ on which $C_1$ and $C_2$ disagree to get set $D_3$.
  4. Get the final classifier output by majority voting on $C_1$, $C_2$, and $C_3$.

Course Outline

- **Fundamentals** (2 weeks)
  - Bayes Decision Theory
  - Probability Density Estimation
- **Discriminative Approaches** (4 weeks)
  - Linear Discriminant Functions
  - Statistical Learning Theory & SVMs
  - Ensemble Methods & Boosting
  - Decision Trees & Randomized Trees
- **Generative Models** (4 weeks)
  - Bayesian Networks
  - Markov Random Fields
- **Unifying Perspective** (2 weeks)
Recap: AdaBoost - “Adaptive Boosting”

- **Main idea** [Freund & Schapire, 1996]
  - 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:
    \[
    H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right)
    \]
    
Recap: AdaBoost - Intuition

Consider a 2D feature space with **positive** and **negative** examples.

Each weak classifier splits the training examples with at least 50% accuracy.

Examples misclassified by a previous weak learner are given more emphasis at future rounds.

Recap: AdaBoost - Algorithm

1. **Initialization:** Set \( w_1^{(n)} = \frac{1}{N} \) for \( n = 1, ..., N \).
2. For \( m = 1, ..., M \) iterations
   a. Train a new weak classifier \( h_m(x) \) using the current weighting coefficients \( W^{(m)} \) by minimizing the weighted error function
      \[
      J_m = \sum_{n=1}^{N} w_1^{(n)} I(h_m(x) \neq t_n) \quad I(4) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{else} \end{cases}
      \]
   b. Estimate the weighted error of this classifier on \( X \):
      \[
      \epsilon_m = \frac{1}{N} \sum_{n=1}^{N} w_1^{(n)} I(h_m(x) \neq t_n)
      \]
   c. Calculate a weighting coefficient for \( h_m(x) \):
      \[
      \alpha_m = \ln \left( \frac{1}{\epsilon_m} \right)
      \]
   d. Update the weighting coefficients:
      \[
      w_1^{(n+1)} = w_1^{(n)} \exp \left\{ \alpha_m I[h_m(x_n) \neq t_n] \right\}
      \]

Recap: Comparing Error Functions

- **Ideal misclassification error function**
- "Hinge error" used in SVMs
- **Exponential error function**
  - Continuous approximation to ideal misclassification function.
  - Sequential minimization leads to simple AdaBoost scheme.
  - Disadvantage: exponential penalty for large negative values!
  - Less robust to outliers or misclassified data points!
Recap: Comparing Error Functions

- Ideal misclassification error function
- “Hinge error” used in SVMs
- Exponential error function
- “Cross-entropy error”
  - Similar to exponential error for \( z > 0 \).
  - Only grows linearly with large negative values of \( z \).

\[
E = -\sum\{ t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \}
\]

Ideal misclassification error function

"Hinge error" used in SVMs

Exponential error function

“Cross-entropy error”

Topics of This Lecture

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

- Random Forests
  - Basic idea
  - Bootstrap sampling
  - Randomized attribute selection
  - Applications

Recap: Comparing Error Functions

- 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 \( \in \{\text{red}, \text{green}, \text{blue}\} \) or weather \( \in \{\text{sunny}, \text{rainy}\} \).
      - 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.”

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

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:
    \[
    \text{Outlook} = \text{Sunny} \land \text{Humidity} = \text{Normal}
    \]
    \[
    \lor (\text{Outlook} = \text{Overcast})
    \]
    \[
    \lor (\text{Outlook} = \text{Rain} \land \text{Wind} = \text{Weak})
    \]
**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:

$\Rightarrow$ 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

$$i(N) = 1 - \max_j \frac{|C_j|}{|N|}$$

“Fraction of the training patterns in category $C_j$ that end up in node $N$.”

- Entropy impurity

$$i(N) = -\sum_j \frac{|C_j|}{|N|} \log_2 \frac{|C_j|}{|N|}$$

“Reduction in entropy = gain in information.”
CART – Overfitting Prevention (Pruning)

- 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.

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

- Evaluating candidate splits
  - Nominal attributes: exhaustive search over all possibilities.
  - Real-valued attributes: only need to consider changes in label.
    - Order all data points based on attribute \( x_i \).
    - Only need to test candidate splits where \( \text{label}(x_i) \neq \text{label}(x_{i+1}) \).

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.

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

- Evaluating candidate splits
  - Nominal attributes: exhaustive search over all possibilities.
  - Real-valued attributes: only need to consider changes in label.
    - Order all data points based on attribute \( x_i \).
    - Only need to test candidate splits where \( \text{label}(x_i) \neq \text{label}(x_{i+1}) \).

CART – Picking a Good Splitting Feature

- Application
  - Select the query that decreases impurity the most
    \[
    \Delta i(N) = i(N) - \sum_{k=1}^{K} P_k i(N_k) - (1 - P_k) i(N_R)
    \]

- Multiway generalization (gain ratio impurity):
  - Maximize
    \[
    \Delta i(s) = \frac{1}{2} \left( i(N) - \sum_{k=1}^{K} P_k i(N_k) \right)
    \]
  - where the normalization factor ensures that large \( K \) are not inherently favored:
    \[
    Z = - \sum_{k=1}^{K} P_k \log_2 P_k
    \]

CART – 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.

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

- Evaluating candidate splits
  - Nominal attributes: exhaustive search over all possibilities.
  - Real-valued attributes: only need to consider changes in label.
    - Order all data points based on attribute \( x_i \).
    - Only need to test candidate splits where \( \text{label}(x_i) \neq \text{label}(x_{i+1}) \).

CART – Picking a Good Splitting Feature

- Application
  - Select the query that decreases impurity the most
    \[
    \Delta i(N) = i(N) - \sum_{k=1}^{K} P_k i(N_k) - (1 - P_k) i(N_R)
    \]

- Multiway generalization (gain ratio impurity):
  - Maximize
    \[
    \Delta i(s) = \frac{1}{2} \left( i(N) - \sum_{k=1}^{K} P_k i(N_k) \right)
    \]
  - where the normalization factor ensures that large \( K \) are not inherently favored:
    \[
    Z = - \sum_{k=1}^{K} P_k \log_2 P_k
    \]

CART – 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.

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

- Evaluating candidate splits
  - Nominal attributes: exhaustive search over all possibilities.
  - Real-valued attributes: only need to consider changes in label.
    - Order all data points based on attribute \( x_i \).
    - Only need to test candidate splits where \( \text{label}(x_i) \neq \text{label}(x_{i+1}) \).

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.

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

- Evaluating candidate splits
  - Nominal attributes: exhaustive search over all possibilities.
  - Real-valued attributes: only need to consider changes in label.
    - Order all data points based on attribute \( x_i \).
    - Only need to test candidate splits where \( \text{label}(x_i) \neq \text{label}(x_{i+1}) \).
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 dismissed as likely due to random chance.
    - Chi-squared statistic (one degree of freedom)
      \[
      \chi^2 = \sum_{i=1}^{N} \frac{(n_{i} - \hat{n}_{i})^2}{\hat{n}_{i}}
      \]
      “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.

(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 \( \Rightarrow \) can prune them selectively.

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
  \[
  i(N) = \sum_{ij} \lambda_{ij}p(C_i)p(C_j)
  \]
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!
    - Result of discrete and greedy learning procedure.
  - Expensive learning step
    - Mostly due to costly selection of optimal split.

Topics of This Lecture

- **Decision Trees**
  - CART
  - Impurity measures
  - Stopping criterion
  - Pruning
  - Evaluations
  - Historical development: ID3, C4.5
- **Random Forests**
  - Basic idea
  - Bootstrap sampling
  - Randomized attribute selection
  - Applications
Random Forests (Breiman 2001)

- 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.
    - Allows fitting to small data sets without explicitly holding back any data for testing.

Random Forests - Injecting Randomness (2)

- Random attribute selection
  - For each node, randomly choose subset of \( T \) attributes on which the split is based (typically square root of number available).
  - Evaluate splits only on OOB data (out-of-bag estimate).
  - Very fast training procedure
    - Need to test few attributes.
    - Evaluate only on ~37% of the data.
  - 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.

Big Question

How can this ever possibly work???
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.
  - Very rapid training
  - Also often used for online learning.
  - 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
    - Code + documentation
    - In Fortran 77
  - But also newer version available in Fortran 90!
  - Fast Random Forest implementation for Java (Weka)


Applications

- **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
Application: Fast Keypoint Detection


References and Further Reading

- More information on Decision Trees can be found in Chapters 8.2-8.4 of Duda & Hart.

R.O. Duda, P.E. Hart, D.G. Stork
Pattern Classification
2nd Ed., Wiley-Interscience, 2000

- The original paper for Random Forests: