Decision Trees and Boosting

Ron Parr
CPS 271

Decision Trees

- Decision trees try to construct small, consistent hypothesis
- Suppose our concept is “blue cube”
Facts About Decision Trees

• If the concept has $d$ conjuncts, there will be a decision tree for this concept with depth $d$.

• Decision trees are very bad for some functions:
  – Parity function
  – Majority function

• For errorless data, you can always construct a decision tree that correctly labels every element of the training set, but it may be exponential in size.

Decision Tree Algorithms

• Aim for:
  – Small decision trees
  – Robustness to misclassification

• Finding shortest decision tree is NP-hard

• Standard approaches are greedy

• Classical approach is to split tree using an information-theoretic criterion
Growing Decision Trees

Repeat until (no good leaves)
Pick leaf
Split = choose_variable(variables – all_parents(leaf))
For val in domain(split)
    new_leaf = new_leaf(split=val)
    new_leaf.instances=leaf.stances s.t. split=val
For leaf in tree
    classification(leaf)=majority_classification(leaf)

Implementation details:
• Implementation of (no good leaves)
• Implementation of choose_variable

Information Theory

• Roughly speaking, information theory measures the expected number of bits needed to communicate information from one person to another

• Suppose person1 is flipping a coin with bias p

• Person1 wants to tell person2 the sequence of results

• Expected number of bits person 1 will send to person 2?

• Note relation to compression
Information Content

\[ I(p_1, \ldots, p_n) = E(\# \text{bits}) = \sum_{i=1}^{n} -p_i \log_2 (p_i) \]

For an unbiased coin, the information content is 1.
For a totally biased coin, the information content is 0.
Information Content of a Leaf

\[ I(\frac{p}{p+n}, \frac{n}{p+n}) = -\frac{p}{p+n} \log_2 p + n - \frac{n}{p+n} \log_2 n \]

Information gain of a split:

\[ I(\frac{p}{p+n}, \frac{n}{p+n}) - \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}) \]

Gain Example

- Suppose we have seen:
  - Red tetrahedron(f), Blue sphere(t), Blue cone(t),
  green cone(f)
- Is it better to split on shape or color?
- Information of original set is: 1
- Information gain of splitting on cone:
- Information gain of splitting on blue:
Favoring Small Examples

• Information gain (and other splitting criteria)
  – Are greedy
  – Favor small trees
• This makes representation an issue yet again
• Suppose you want to learn “parity(+) and blue”
• Hard to learn with decision trees, but
  – If we treat parity like a state variable, then it’s easy
  – Call these derived variables features or attributes

Decision Tree Conclusion

• Simple method
• Works surprisingly well in many cases
• Can be generalized to continuous values

• Issues:
  – Missing values
  – Expressive power
Boosting

Contents adapted from slides by Sebastian Thrun, Vandi Verma, and Rob Schapire

How to Make $$$ In Horse Races?

• Ask professional.
• Suppose:
  – Professional cannot give single highly accurate rule
  – But presented with a set of races, can always generate better-than-random rules

• Can you get rich?
Idea

• Ask expert for rule-of-thumb
• Assemble set of cases where rule-of-thumb fails (hard cases, e.g., imaginary races)
• Ask expert again for selected set of hard cases
• And so on...
• Combine all rules-of-thumb
• Expert could be “weak” learning algorithm

Questions

• **How to choose** races on each round?
  – concentrate on “hardest” races
    (those most often misclassified by previous rules of thumb)

• **How to combine** rules of thumb into single prediction rule?
  – take (weighted) majority vote of rules of thumb
Boosting

• **boosting** = general method of converting rough rules of thumb into highly accurate prediction rule

• more technically:
  – given “weak” learning alg. that can consistently find hypothesis (classifier) w/error $\leq 1/2-\gamma$
  – a boosting algorithm can **provably** construct single hypothesis with error $\leq \varepsilon$
  – theory & practice suggest good generalization

Background

• [Valiant’84]
  theoretical PAC model for studying machine learning

• [Kearns&Valiant’88]
  open problem of finding a boosting algorithm

• [Schapire’89], [Freund’90]
  first polynomial-time boosting algorithms

• [Drucker, Schapire&Simard ’92]
  first experiments using boosting
Background (cont.)

• [Freund & Schapire ‘95]
  – introduced AdaBoost algorithm
  – strong practical advantages over previous boosting algorithms

• experiments using AdaBoost:
  [Drucker & Cortes ‘95] [Schapire & Singer ‘98]
  [Jackson & Cravon ‘96] [Maclin & Opitz ‘97]
  [Freund & Schapire ‘96] [Bauer & Kohavi ‘97]
  [Quinlan ‘96] [Schwenk & Bengio ‘98]
  [Breiman ‘96] [Dietterich ‘98]

• continuing development of theory & algorithms:
  [Schapire, Freund, Bartlett & Lee ‘97] [Schapire & Singer ‘98]
  [Breiman ‘97] [Mason, Bartlett & Baxter ‘98]
  [Grove and Schuurmans ‘98] [Friedman, Hastie & Tibshirani ‘98]

Algorithmic View of Boosting

• Given training set \( X = \{ (x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)}) \} \)
• \( y^{(i)} \in \{-1, +1\} \) correct label of instance \( x^{(i)} \in X \)

• for \( t = 1, \ldots, T \):
  • construct distribution \( D_t \) on \( \{1, \ldots, m\} \)
  • Find weak hypothesis (“rule of thumb”)
    \( h_t : X \rightarrow \{-1, +1\} \)
    with small error \( \varepsilon_t \) on \( D_t \):
    \[ \varepsilon_t = \Pr_{D_t} [ h_t(x_i) \neq y_i ] \]
  • output final hypothesis \( H_{\text{final}} \)
AdaBoost [Freund&Schapire ’97]

• constructing $D_t$:
  - $D_t(i) = \frac{1}{m}$

• given $D_t$ and $h_t$:

$$
D_{t+1} = \frac{D_t \cdot e^{-\alpha_t}}{Z_t} \begin{cases} 
  e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\
  e^{\alpha_t} & \text{if } y_i \neq h_t(x_i)
\end{cases}
$$

where: $Z_t = \text{normalization constant}$

$$
\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_t}{\varepsilon_t} \right) > 0
$$

• final hypothesis:

$$
H_{final}(x) = \text{sgn} \left( \sum_t \alpha_t h_t(x) \right)
$$

AdaBoost Magnified

\[ \alpha_t = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_t}{\varepsilon_t} \right) > 0 \]

\[ D_{t+1} = \frac{D_t \cdot e^{-\alpha_t}}{Z_t} \begin{cases} 
  e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\
  e^{\alpha_t} & \text{if } y_i \neq h_t(x_i)
\end{cases} \]

\[ H_{final}(x) = \text{sgn} \left( \sum_t \alpha_t h_t(x) \right) \]
Digression: Weights

• Q: What if our learning algorithm doesn’t handle weighted inputs?

• A: Can always simulate weights by:
  – Duplicating inputs
  – Sampling

Digression: Weak Learners

• How hard is it to make a weak learner?
• Easier than you might think:
  – Planes (in continuous space)
  – Stumps (decision trees with just one level)
  – Haar wavelet-like features for vision

• Can always turn a reliably bad learner into a reliably good one (at least for binary classification)
Round 2

\[ e_2 = 0.21 \]
\[ \alpha_2 = 0.65 \]

Round 3

\[ e_3 = 0.14 \]
\[ \alpha_3 = 0.92 \]
Final Hypothesis

\[ H_{\text{final}} = \text{sign} \left( 0.42 \cdot +0.65 \cdot +0.92 \right) \]

UCI Benchmarks

Comparison with
- C4.5 (Quinlan’s Decision Tree Algorithm)
- Decision Stumps (only single attribute)

\[
\begin{align*}
\text{eye color = brown?} \\
\text{predict} & \quad \text{yes} \quad \text{predict} \\
+1 & \quad -1
\end{align*}
\]

\[
\begin{align*}
\text{height > 5 feet?} \\
\text{predict} & \quad \text{yes} \quad \text{predict} \\
-1 & \quad +1
\end{align*}
\]
Understanding Boosting

- Many interpretations of boosting
- Game theoretic interpretation
- Gradient descent in margin space (more later)
- With large set of stumps:
  - Constructs a perceptron
  - Adds features one a time
Analyzing the Training Error

- Theorem [Freund & Schapire ’97]:
  
  Write $\varepsilon_t$ as $\frac{1}{2} \cdot \gamma_t$
  
  Then training error($H_{final}$) $\leq \exp\left(-2 \sum_t \gamma_t^2\right)$

  So if $\forall t: \gamma_t \geq \gamma > 0$ then

  training error($H_{final}$) $\leq e^{-2\gamma^2 T}$

- Key point: Error gets crushed exponentially

- Remark: AdaBoost is adaptive:
  
  - does not need to know $\gamma$ or $T$ a priori
  - can exploit $\gamma_t >> \gamma$

Proof Intuition (Rob’s)

- on round $t$:
  
  Increase weight of examples incorrectly classified by $h_t$

- if $x'$ incorrectly classified by $h_t$
  
  Then $x'$ incorrectly classified by weighted majority of $h_t$’s
  
  Then $x'$ must have “large” weight under final dist. $D_{T+1}$

- since total weight $\leq 1$:
  
  Number of incorrectly classified examples “small”
Overfitting: What “should” happen

- training error to continue to drop (or reach zero)
- test error to increase when $H_{\text{final}}$ becomes “too complex” (Occam’s razor)

What often happens

- Test error does not increase even after 1,000 rounds
- Test error continues to drop after training error is zero!
- Boosting can be interpreted as increasing the “margin” of the classifier, a concept we’ll revisit later in the context of support vector machines
Extensions: Multiclass Problems

- Reduce to binary problem by creating several binary questions for each example:
  
  - “does or does not example $x$ belong to class 1?”
  - “does or does not example $x$ belong to class 2?”
  - “does or does not example $x$ belong to class 3?”
  - ...

Text Categorization

- Decision stumps: presence of word or short phrase. Example:

  “If the word Clinton appears in the document predict document is about politics”

![Graphs showing error rates for different number of classes](database: AP, database: Reuters)
Extensions: Confidences and Probabilities

- Prediction of hypothesis $h_t$: $\text{sgn}(h_t(x))$
- Confidence of hypothesis $h_t$: $|h_t(x)|$
- Probability of $H_{\text{final}}$: $\Pr_h[y = +1|x] = \frac{e^{f(x)}}{e^{f(x)} + e^{-f(x)}}$

[Schapire&Singer '98], [Friedman, Hastie & Tibshirani '98]

Practical Advantages of AdaBoost

- (quite) fast
- simple + easy to program
- only a single parameter to tune ($T$)
- no prior knowledge
- flexible: can be combined with any classifier (neuro net, C4.5, ...)
- provably effective (assuming weak learner)
  - shift in mind set: goal now is merely to find hypotheses that are better than random guessing
- finds outliers
Caveats

• performance depends on data & weak learner
• AdaBoost can fail if
  – weak hypothesis too complex (overfitting)
  – weak hypothesis too weak ($\gamma_t \rightarrow 0$ too quickly),
    • underfitting
    • Low margins $\rightarrow$ overfitting
• empirically, AdaBoost seems especially susceptible to noise

Conclusion

• boosting useful tool for classification problems
  • grounded in rich theory
  • performs well experimentally
  • often (but not always) resistant to overfitting
  • many applications
• but
  • slower classifiers
  • Results possibly less comprehensible
  • sometime susceptible to noise
Final Thoughts

- Decision trees and variants are a classical method from the early days of (statistical) machine learning
- Developed independently in machine learning community and statistics community
- Many software packages the implement some form of decision trees (with varying stopping and splitting criteria) are available, including continuous variable extensions

- Boosting is a powerful extension to decision trees (even stumps) or any basic machine learning method
- Boosting resists overfitting and can be viewed as a feature selection method