The Boosting Approach to Machine Learning

CS 536: Machine Learning
Littman (Wu, TA)

Example: Spam Filtering

- problem: filter out spam (junk email)
- gather large collection of examples of spam and non-spam:

From: yoav@att.com
From: xa412@hotmail.com

... Rob, can you review a paper... Earn money without working!!!

- main observation:
  - easy to find “rules of thumb” that are “often” correct (If “buy now” appears, predict spam)
  - hard to find a single rule that is very highly accurate

The Boosting Approach

- devise computer program for deriving rough rules of thumb
- apply procedure to subset of emails
- obtain rule of thumb
- apply to 2nd subset of emails
- obtain 2nd rule of thumb
- repeat T times

Source

Notes adapted from Rob Schapire
www.cs.princeton.edu/~schapire
Details

- how to *choose examples* on each round?
  - concentrate on “hardest” examples (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 algorithm that can consistently find classifier with error no more than $1/2 - \gamma$
  - a boosting algorithm can *provably* construct single classifier with no more than $\varepsilon$ error ($\varepsilon, \gamma$ small)

This Talk

- introduction to AdaBoost
- analysis of training error
- analysis of generalization error based on theory of margins
- applications, experiments and extensions

Background

- [Valiant 84] – introduced 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
More Background

- [Freund & Schapire 95]
  - introduced "AdaBoost" algorithm
  - strong practical advantages over previous boosting algorithms
- AdaBoost experiments, applications
  - [Drucker & Cortes 96], [Jackson & Craven 96], [Freund & Schapire 96], [Quinlan '96], [Breiman 96], [Maclin & Opitz 97], [Bauer & Kohavi 97], [Schwenk & Bengio '98], [Schapire, Singer & Singhal '98], [Alony, Schapire & Singer '99], [Cohen & Singer '99], [Dieleman, Marquez & Rigau '00].
  - [Tieu & Viola 00], [Walker, Rambow & Rogati 01], [Rochery, Schapire, Rahim & Gupta 01], [Merler, Furlanello, Larcher & Sboner 01], ...
- continued algorithm/theory development
  - [Breiman 98, 99], [Schapire, Freund, Bartlett & Lee 98], [Grove & Schuurmans 98], [Mason, Bartlett & Baxter 98], [Schapire & Singer 99], [Cohen & Singer 99], [Freund & Mason 99], [Domingo & Watanabe 99], [Mason, Baxter, Bartlett & Frean 99, 00], [Duffy & Heimbold 99, 02], [Freund & Mason 99], [Ridgeway, Madigan & Richardson 99], [Kivinen & Warmuth 99], [Friedman, Hastie & Tibshirani 00], [Ratsch, Onoda & Muller 00], [Ratsch, Warmuth, Mika, Onoda, Lemm & Muller 00], [Allwein, Schapire & Singer 00], [Friedman 01], [Koltchinskii, Panchenko & Lozano 01], [Collins, Schapire & Singer 02], [Deniz, Bennett & Shawe-Taylor 02], [Lebanon & Lafferty 02], ...

A Formal Description

- given training set \((x_1, y_1), \ldots, (x_m, y_m)\)
- \(y_i \in \{-1, +1\}\) correct label of instance \(x_i\) in \(X\)
- for \(t = 1, \ldots, T\):
  - construct distribution \(D_t\) on \(\{1, \ldots, m\}\)
  - find weak classifier ("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 classifier \(H_{\text{final}}\)

AdaBoost (Schapire & Freund)

- constructing \(D_t\)
  - \(D_1(i) = 1/m\)
  - given \(D_t\) and \(h_t\):
    \[D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}\]
    \[= \frac{D_t(i) \exp(-\alpha_t \cdot y_i \cdot h_t(x_i))}{Z_t}\]
  - where \(Z_t\) = normalization constant
    \[\alpha_t = 1/2 \ln \left( \frac{1 - \varepsilon_t}{\varepsilon_t} \right) > 0\]
- final classifier
  - \(H_{\text{final}}(x) = \text{sign}(\Sigma_t \text{ at } h_t(x))\)

Toy Example

- weak classifiers = vertical or horizontal half-planes
Round 1

Round 2

Round 3

Final Classifier
Analyzing the Training Error

- **Theorem**
  - write $\epsilon_t$ as $1/2 - \gamma_t$
  - then
    
    \[
    \text{training error}(H_{\text{final}}) \leq \exp(-2 \sum \gamma_t^2)
    \]

- so: if for all $t$: $\gamma_t \geq \gamma > 0$
  then training error($H_{\text{final}}$) $\leq \exp(-2 \gamma_t^2 T)$

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

Proof

- **Step 1:** unwrapping recurrence:
  \[
  D_{T+1}(i) = \frac{1}{m} \exp(-y_i h_t(x_i)) \prod_{i} Z_t
  \]
  where $h(x) = \sum_{i} \alpha_i h_t(x)$

- **Step 2:**
  \[
  \text{training error}(H_{\text{final}}) = \frac{1}{m} \sum \begin{cases} 
  1 & \text{if } y_i \neq H_{\text{final}}(x_i) \\
  0 & \text{else}
  \end{cases} \\
  = \frac{1}{m} \sum \begin{cases} 
  1 & \text{if } y_i h_t(x_i) \leq 0 \\
  0 & \text{else}
  \end{cases} \\
  \leq \frac{1}{m} \sum \exp(-y_i h_t(x_i)) \\
  = \sum \frac{D_{T+1}(i) y_i}{Z_t} \\
  = \prod Z_t
  \]

- **Step 3:**
  \[
  Z_t = 2(\epsilon_t(1-\epsilon_t)) = 1 - \epsilon_t^2 \leq e^{-2\gamma_t^2}
  \]

Proof Intuition

- on round $t$:
  - increase weight of examples incorrectly classified by $h_t$
  - if $x_i$ incorrectly classified by $H_{\text{final}}$
    then $x_i$ incorrectly classified by (weighted) majority of $h_t$'s
  - therefore, if $x_i$ incorrectly classified by $H_{\text{final}}$
    then $x_i$ must have “large” weight under final distribution $D_{T+1}$
  - number of incorrectly classified examples “small” (since total weight no more than 1)

Generalization Error, Part 1

- expect:
  - training error to continue to drop (or reach zero)
  - test error to *increase* when $H_{\text{final}}$ becomes “too complex”
    - “Occam’s razor”
    - overfitting
      - hard to know when to stop training
**Actual Typical Run**

- Test error does *not* increase, even after 1000 rounds — (total size > 2M nodes)
- Test error continues to drop even after training error is zero!
- Occam’s razor *wrongly* predicts “simpler” rule is better

**Better Story: Margins**

- **Key idea** (Schapire, Freund, Bartlett, Lee)
  - Training error only measures whether classifications are right or wrong
  - Should also consider *confidence* of classifications
- Can write $H_{\text{final}}(x) = \text{sign}(f(x))$
  where
  $$f(x) = \frac{\sum \alpha_h(x)}{\sum \alpha_h} \in [-1, +1]$$
- Define *margin* of example $(x,y)$ to be $y f(x) =$ measure of confidence of classifications

**Empirical Evidence**

- *Margin distribution* = cumulative distribution of margins of training examples

**Theoretical Evidence**

- If all training examples have large margins, then can approximate final classifier by a much *small* classifier
  - (similar to how polls can predict outcome of a not-too-close election)
- Can use this fact to prove that large margins imply better test error, regardless of number of weak classifiers
- Can also prove that *boosting tends to increase margins* of training examples by concentrating on those with smallest margin
- So: although final classifier is getting larger, margins are likely to be increasing, so final classifier is actually getting close to a simpler classifier, driving down the test error
Practical Advantages
• fast
• simple and easy to program
• no parameters to tune (except T)
• flexible--- can combine with any learning algorithm
• no prior knowledge needed about weak learner
• provably effective, provided can consistently find rough rules of thumb
  - shift in mindset: goal now is merely to find classifiers barely better than random guessing
• versatile
  - can use with data that is textual, numeric, discrete, etc.
  - has been extended to learning problems well beyond binary classification

Caveats
• performance of AdaBoost depends on data and weak learner
  • consistent with theory, AdaBoost can fail if
    – weak classifier too complex
      • overfitting
    – weak classifiers too weak ($\gamma_t \rightarrow 0$ too quickly)
      • underfitting
      • low margins $\rightarrow$ overfitting
• empirically, AdaBoost seems especially susceptible to uniform noise

UCI Experiments
• tested AdaBoost on UCI benchmarks
• used: (Schapire and Freund)
  – C4.5 (Quinlan’s decision tree algorithm)
  – “decision stumps”: very simple rules of thumb that test on single attributes

UCI Results
• error, error plots
Multiclass Problems

• most direct extension effective only if all weak classifiers have error no more than half
  – difficult to achieve for “weak” weak learners

• instead, 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?”
  – ...

Application: Text Categorization

• weak classifiers are decision stumps
  – test for presence of word or short phrase in document, for example,
    • “if the word Clinton appears in the document, predict document is about politics”

• Schapire & Singer found it consistently beat or tied tested competitors

Confidence-rated Predictions

• useful to allow weak classifiers to express confidences about predictions

• formally, allow $h_t : X \rightarrow R$
  $\text{sign}(h_t(x)) = \text{prediction}$
  $|h_t(x)| = \text{“confidence”}$

• proposed general principle for:
  – modifying AdaBoost
  – designing weak learners to find (confidence-rated) $h_t$s

• sometimes makes learning faster since removes need to undo under-confident predictions of earlier weak classifiers

Can Help A Lot
Human–computer Dialogue

- phone “helpdesk” for AT&T Natural Voices text-to-speech business
- NLU’s job: classify caller utterances into 24 categories (demo, sales rep, pricing info, yes, no, etc.)

Using Human Knowledge

- boosting is data-driven
  - so works best with lots of data
- for rapid deployment, can’t wait to gather lots of data
  - want to compensate with human knowledge
- idea: balance fit to data against fit to prior, human-built model

Results

Bidding Agents

- trading agent competition (TAC)
- 8 agents in each game
- must purchase flights, hotel rooms and entertainment tickets for 8 clients in complicated, interacting auctions
- value of one good depends on price of others
  - example, need both incoming and outgoing flights
- so: need to predict prices, especially of hotel rooms (Schapire, Stone, Csirik, Littman, McAllester)
  - used boosting
- second place in tournament using straight scores
  - first place with “handicapped” scores)
Predicting Hotel Prices

- predicting real numbers (prices)
- want to estimate entire distribution of prices, given current conditions
- main ideas:
  - reduce to multiple binary classification problems:
    “is price above or below $100?”
    “is price above or below $150?”
  - extract probabilities using modification of boosting for logistic regression
- can be applied to any conditional density estimation problem

Conclusions

- boosting is a useful new tool for classification and other learning problems
  - grounded in rich theory
  - performs well experimentally
  - often (but not always!) resistant to overfitting
  - many applications and extensions
- other stuff:
  - theoretical connections to: game theory and linear programming, SVMs, logistic regression, convex analysis and Bregman distances
  - tool for data cleaning: very effective at finding outliers (mislabeled or ambiguously labeled examples)

Demos

- Face detector
  http://www1.cs.columbia.edu/~freund/talks/violamovie.mpg
- How Can I Help You?
  http://www.cs.princeton.edu/courses/archive/spring04/cos511/demo.au