10601 Machine Learning

Boosting
Fighting the bias-variance tradeoff

• **Simple (a.k.a. weak) learners are good**
  – e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
  – Low variance, don’t usually overfit

• **Simple (a.k.a. weak) learners are bad**
  – High bias, can’t solve hard learning problems

• Can we make weak learners always good???
  – No!!!
  – But often yes...
Voting (Ensemble Methods)

• Instead of learning a single (weak) classifier, learn many weak classifiers that are good at different parts of the input space

• Output class: (Weighted) vote of each classifier
  – Classifiers that are most “sure” will vote with more conviction
  – Classifiers will be most “sure” about a particular part of the space
  – On average, do better than single classifier!

• But how do you ???
  – force classifiers to learn about different parts of the input space?
  – weigh the votes of different classifiers?
Boosting [Schapire, 1989]

- Idea: given a weak learner, run it multiple times on (reweighted) training data, then let the learned classifiers vote

- On each iteration $t$:
  - weight each training example by how incorrectly it was classified
  - Learn a hypothesis $- h_t$
  - A strength for this hypothesis $- \alpha_t$

- Final classifier:
  - A linear combination of the votes of the different classifiers weighted by their strength

- Practically useful
- Theoretically interesting
Learning from weighted data

- Sometimes not all data points are equal
  - Some data points are more equal than others

- Consider a weighted dataset
  - $D(i)$ – weight of $i$th training example $(x^i, y^i)$
  - Interpretations:
    - $i$th training example counts as $D(i)$ examples
    - If I were to “resample” data, I would get more samples of “heavier” data points

- Now, in all calculations, whenever used, $i$th training example counts as $D(i)$ “examples”
  - e.g., MLE for Naïve Bayes, redefine $\text{Count}(Y=y)$ to be weighted count
Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\)

Initialize \(D_1(i) = 1/m\).

For \(t = 1, \ldots, T\):

- Train weak learner using distribution \(D_t\).
- Get weak classifier \(h_t : X \rightarrow \mathbb{R}\).
- Choose \(\alpha_t \in \mathbb{R}\).
- Update:

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

where \(Z_t\) is a normalization factor

\[
Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))
\]

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right).
\]

Figure 1: The boosting algorithm AdaBoost.
What $\alpha_t$ to choose for hypothesis $h_t$?

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i))$$

Where $f(x) = \sum_{t} \alpha_t h_t(x)$; $H(x) = \text{sign}(f(x))$
What $\alpha_t$ to choose for hypothesis $h_t$?

[Schapire, 1989]

Training error of final classifier is bounded by:

$$
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i)) = \prod_t Z_t
$$

Where

$$
f(x) = \sum_t \alpha_t h_t(x); \quad H(x) = \text{sign}(f(x))
$$

$$
Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))
$$
What $\alpha_t$ to choose for hypothesis $h_t$?

[Schapire, 1989]

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i} \exp(-y_i f(x_i)) = \prod_t Z_t$$

Where

$$f(x) = \sum_t \alpha_t h_t(x); H(x) = \text{sign}(f(x))$$

If we minimize $\prod_t Z_t$, we minimize our training error

We can tighten this bound greedily, by choosing $\alpha_t$ and $h_t$ on each iteration to minimize $Z_t$.

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$
What $\alpha_t$ to choose for hypothesis $h_t$?

We can minimize this bound by choosing $\alpha_t$ on each iteration to minimize $Z_t$.

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

Define

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) \delta(h_t(x_i) \neq y_i)$$

We can show that:

$$Z_t = (1 - \epsilon_t) \exp^{-\alpha_t} + \epsilon_t \exp^{\alpha_t}$$
What $\alpha_t$ to choose for hypothesis $h_t$?

We can minimize this bound by choosing $\alpha_t$ on each iteration to minimize $Z_t$.

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

For boolean target function, this is accomplished by [Freund & Schapire '97]:

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Where:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) \delta(h_t(x_i) \neq y_i)$$

[Schapire, 1989]
Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\) 
Initialize \(D_1(i) = 1/m\).
For \(t = 1, \ldots, T\):

- Train base learner using distribution \(D_t\).
- Get base classifier \(h_t : X \rightarrow \mathbb{R}\).
- Choose \(\alpha_t \in \mathbb{R}\).
- Update:

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

\[
\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)
\]
Strong, weak classifiers

• If each classifier is (at least slightly) better than random
  – \( \varepsilon_t < 0.5 \)

• With a few extra steps it can be shown that AdaBoost will achieve zero *training error* (exponentially fast):

\[
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \prod_{t} Z_t \leq \exp \left( -2 \sum_{t=1}^{T} \left( \frac{1}{2} - \varepsilon_t \right)^2 \right)
\]
Boosting results – Digit recognition

[Schapire, 1989]

- Boosting often
  - Robust to overfitting
  - Test set error decreases even after training error is zero
Boosting: Experimental Results

Comparison of C4.5, Boosting C4.5, Boosting decision stumps (depth 1 trees), 27 benchmark datasets

[Freund & Schapire, 1996]
AdaBoost and AdaBoost.MH on Train (left) and Test (right) data from Irvine repository. [Schapire and Singer, ML 1999]
Boosting and Logistic Regression

Logistic regression assumes:

\[ P(Y = 1|X) = \frac{1}{1 + \exp(f(x))} \]

And tries to maximize data likelihood:

\[ P(D|H) = \prod_{i=1}^{m} \frac{1}{1 + \exp(-y_if(x_i))} \]

Equivalent to minimizing log loss

\[ \sum_{i=1}^{m} \ln(1 + \exp(-y_if(x_i))) \]
Boosting and Logistic Regression

Logistic regression equivalent to minimizing log loss

\[ \sum_{i=1}^{m} \ln(1 + \exp(-y_if(x_i))) \]

Boosting minimizes similar loss function!!

\[ \frac{1}{m} \sum_{i} \exp(-y_if(x_i)) = \prod_{t} Z_t \]

Both smooth approximations of 0/1 loss!
Logistic regression and Boosting

Logistic regression:
• Minimize loss fn
\[ \sum_{i=1}^{m} \ln(1 + \exp(-y_if(x_i))) \]
• Define
\[ f(x) = \sum_{j} w_j x_j \]
where \( x_j \) predefined

Boosting:
• Minimize loss fn
\[ \sum_{i=1}^{m} \exp(-y_if(x_i)) \]
• Define
\[ f(x) = \sum_{t} \alpha_t h_t(x) \]
where \( h_t(x_i) \) defined dynamically to fit data
(not a linear classifier)
• Weights \( \alpha_j \) learned incrementally
What you need to know about Boosting

- Combine weak classifiers to obtain very strong classifier
  - Weak classifier – slightly better than random on training data
  - Resulting very strong classifier – can eventually provide zero training error
- AdaBoost algorithm
- Boosting v. Logistic Regression
  - Similar loss functions
  - Single optimization (LR) v. Incrementally improving classification (B)
- Most popular application of Boosting:
  - Boosted decision stumps!
  - Very simple to implement, very effective classifier