Boosting for Unlabelled Data

Steven Abney
AT&T Labs–Research

Motivation

- Many classification problems in language, much unlabelled data
- Examples
  - Yarowsky: word-sense disambiguation
  - Blum & Mitchell: web page classification
  - Brin: author-title pairs
  - Collins & Singer: named entity classification
  - Hearst: “is-a” pairs
  - Roark & Charniak: cosiblings in taxonomy
- Holy grail: completely unsupervised language learning

AdaBoost

- Like MaxEnt: can use any features
- Don't need constrained ordering
- Don't need independence
- Smoothing is not an issue
- Very resistant to overfitting
- Much more efficient than GIS
- But designed for supervised training

The setting

- Handful of positives, find me more
- Yarowsky: train on seed, label where confident, repeat
- AdaBoost provides confidence scores
- Differs from Yarowsky's, Collins & Singer's setting:
  - Binary
  - Highly skewed distribution
  - Only positives in seed

The Basic Idea

- Assume unlabelled = negative, treat as label noise
- Bayesian image reconstruction
  - Posterior from prior and likelihood (fit)

$$p(\mathbf{y}|\tilde{\mathbf{y}}) \propto p(\mathbf{y}, \tilde{\mathbf{y}}) = p(\mathbf{y})p(\tilde{\mathbf{y}}|\mathbf{y})$$

  - Prior from classifier: $p(\mathbf{y}|\mathbf{x})$
  - Noise: probability u of being mislabelled

$$p(\tilde{\mathbf{y}}|\mathbf{y}) = \prod_i u^{[\![\tilde{y}_i \neq y_i]\!]}(1-u)^{[\![\tilde{y}_i = y_i]\!]}$$

- AdaBoost doesn't give probabilities
- More general: loss combines classifier and fit components

AdaBoost

- Examples $\mathbf{x}$; individual example x_i
- Labels $\mathbf{y}$; individual label y_i
- Initial ("observed") labels $\tilde{\mathbf{y}}$; individual label $\tilde{y}_i$
- Predictors ("weak hypotheses") h_k

$$h_k(x) = \begin{cases} +y & \text{if } P(x) \\ -y & \text{otherwise} \end{cases}$$

AdaBoost

- Prediction for example x_i

$$f(x_i) = \sum_k \alpha_k h_k(x_i)$$

  - Predicted label $= \mathrm{sign}(f(x_i))$
  - Confidence $= |f(x_i)|$

AdaBoost

- Measure difficulty (loss) of examples

$$L_c(x_i) = \left\{ \begin{array}{ll} e^{\text{confidence}} & \text{if prediction is wrong} \\ 1/e^{\text{confidence}} & \text{if prediction is right} \end{array} \right\} = e^{-y_i f(x_i)}$$

- Objective: minimize total loss

$$L_c = \sum_i L_c(x_i)$$

Constructing Classifier

- For each predictor h_k, find optimal weight α_k

$$\alpha_k = \frac{1}{2} \log \frac{A}{B}$$

- Compute what new loss will be

$$\text{NewLoss} = 2\sqrt{AB}$$

- Choose α_k, h_k that minimizes new loss, add it in

$$f(x_i) = \sum_k \alpha_k h_k(x_i)$$

- Repeat

Loss is Upper Bound on Error

- Classifier error: $\mathrm{cerr}(x_i)$

$$\begin{aligned}&\text{if prediction is wrong} \quad L_c(x_i) = e^{\text{confidence}} && \geq 1 \quad = \mathrm{cerr}(x_i)\\ &\text{if prediction is right} \quad L_c(x_i) = 1/e^{\text{confidence}} && \geq 0 \quad = \mathrm{cerr}(x_i)\end{aligned}$$

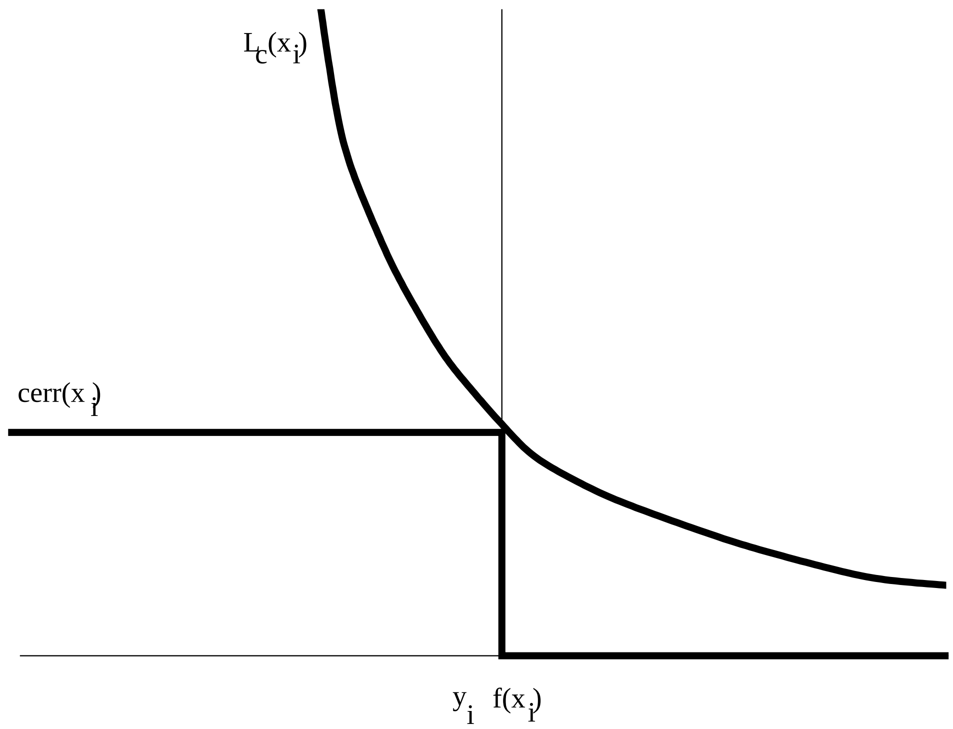
- Loss is upper bound for error

$$L_c(x_i) \geq \mathrm{cerr}(x_i)$$

$$L_c \geq \mathrm{cerr}$$

- AdaBoost minimizes errors by minimizing loss L_c

Loss is Upper Bound on Error



U-Boost

Input: attributes $\mathbf{x}$, observation $\tilde{\mathbf{y}}$, threshold g.

1. At $t = 0$, initialize $\mathbf{y}^{(t)} = \tilde{\mathbf{y}}$
2. Repeat to convergence:
   a. **Boosting Step**
   Use AdaBoost on $(\mathbf{x}, \mathbf{y}^{(t)})$ to choose $\alpha^{(t)}$
   b. **Relabelling Step**
   Define $\mathbf{y}^{(t+1)}$ as:

$$y_i^{(t+1)} = \begin{cases} -\tilde{y}_i & \text{if } -\tilde{y}_i \text{ predicted and } |f(x_i)| > g \\ \tilde{y}_i & \text{otherwise} \end{cases}$$

Relabelling Error

- Relabelling error

$$\mathrm{lerr}(x_i) = [\![y_i \neq \tilde{y}_i]\!]$$

- Relabelling loss

$$L_r(x_i) = \left\{ \begin{array}{ll} e^{\gamma} > 1 & \text{if } y_i \neq \tilde{y}_i \\ 1/e^{\gamma} > 0 & \text{if } y_i = \tilde{y}_i \end{array} \right\} = \mathrm{lerr}(x_i)$$

U-Boost Total Error

- Total error

$$\max(L_c(x_i), L_r(x_i)) \geq \max(\mathrm{cerr}(x_i), \mathrm{lerr}(x_i)) = \mathrm{err}(x_i)$$

- Sum of two positives upper bounds max
- Total loss is upper bound on total error

$$L(x_i) = L_c(x_i) + L_r(x_i) \geq \mathrm{err}(x_i)$$

U-Boost Minimizes Loss

- Loss $L = \sum_i L_c(x_i) + L_r(x_i)$
- In boosting step, labelling unchanged
  - So $L_r(x_i)$ is unchanged
  - AdaBoost decreases $L_c(x_i)$

U-Boost Minimizes Loss

- In relabelling step:

$$L(x_i) = e^{-f(x_i)y_i} + e^{-\gamma y_i \tilde{y}_i}$$

$$\text{If keep label} \quad L(x_i) = e^{-f(x_i)\tilde{y}_i} + e^{-\gamma}$$

$$\text{If flip label} \quad L(x_i) = e^{f(x_i)\tilde{y}_i} + e^{\gamma}$$

U-Boost Minimizes Loss

- So flip label just in case:

$$e^{-f(x_i)\tilde{y}_i} + e^{-\gamma} > e^{f(x_i)\tilde{y}_i} + e^{\gamma}$$

$$e^{-f(x_i)\tilde{y}_i} - e^{f(x_i)\tilde{y}_i} > e^{\gamma} - e^{-\gamma}$$

$$2\sinh(-f(x_i)\tilde{y}_i) > 2\sinh(\gamma)$$

$$-f(x_i)\tilde{y}_i > \gamma$$

- Relabelling step decreases loss, $g = \gamma$

Selecting γ

- γ represents belief about target concept size
- Friedman et al. suggest normalizing boosting loss to get probability

$$p(y_i \neq \tilde{y}_i) = \frac{e^{-\gamma}}{e^{-\gamma} + e^{\gamma}}$$

- If seed set is iid from target

$$p(y_i \neq \tilde{y}_i) = \frac{M - n}{N}$$

- Ergo

$$\gamma = \frac{1}{2} \log(\frac{N}{M - n} - 1)$$

Application to Active Learning

- Choosing examples for humans to annotate
- Choose initial value for γ, choose borderline examples

$$-f(x_i)\tilde{y}_i = \gamma$$

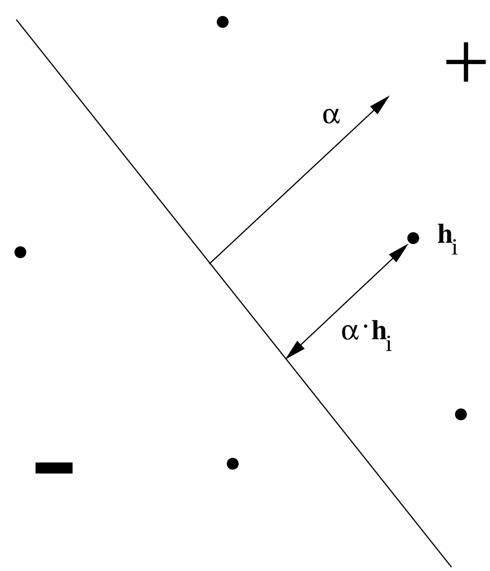
- If too many are negative, increase γ, vice versa

Geometric Interpretation

- AdaBoost loss function

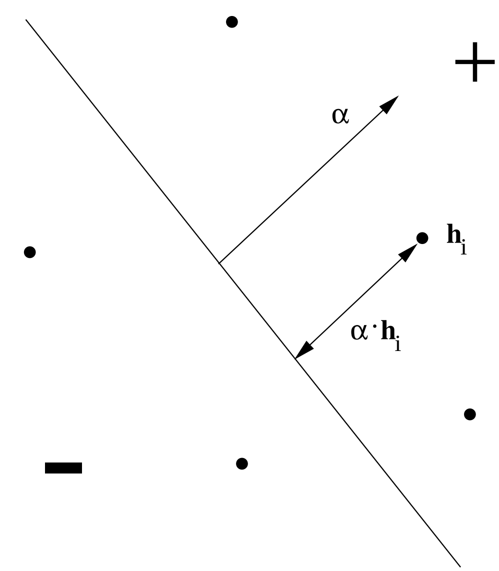
$$L_c = \sum_i e^{-y_i \Sigma_k \alpha_k h_k(x_i)}$$

- Dot product of weight vector $\vec{\alpha}$ and feature vector $\mathbf{h}_i$
- Weight vector defines hyperplane



Geometric Interpretation

- Dot product is distance; negative means negative side
- Multiplying by y_i changes sign: negative means on wrong side
- Margin $y_i \vec{\alpha} \cdot \mathbf{h}_i$



Geometric Interpretation: U-Boost

- AdaBoost (boosting step) minimizes error by maximizing margin
- Relabelling step relabels examples deepest in wrong half-plane
- Allows hyperplane to move in next boosting step
- Seeks "fissure" that allows largest possible margin
- Allow negative γ: keeps hyperplane moving even if separable
- Annealing