CSCE 478/878 Lecture 7: Bagging and Boosting

Stephen Scott

(Adapted from Ethem Alpaydin and Rob Schapire and Yoav Freund)

sscott@cse.unl.edu

Introduction

- Sometimes a single classifier (e.g., neural network, decision tree) won't perform well, but a **weighted combination** of them will
- When asked to predict the label for a new example, each classifier (inferred from a **base learner**) makes its own prediction, and then the **master algorithm** (or **meta-learner**) combines them using the weights for its own prediction
- If the classifiers themselves cannot learn (e.g., heuristics) then the best we can do is to learn a good set of weights (e.g., **Weighted Majority**)
- If we are using a learning algorithm (e.g., ANN, dec. tree), then we can rerun the algorithm on different subsamples of the training set and set the classifiers' weights during training

Outline

- Bagging
- Boosting

Bagging
[Breiman, ML Journal, 1996]

Bagging = Bootstrap aggregating

Bootstrap sampling: given a set $\mathcal{X}$ containing N training examples:

- Create $\mathcal{X}_j$ by drawing N examples uniformly at random **with replacement** from $\mathcal{X}$
- Expect $\mathcal{X}_j$ to omit $\approx 37\%$ of examples from $\mathcal{X}$

Bagging:

- Create L bootstrap samples $\mathcal{X}_1, \ldots, \mathcal{X}_L$
- Train classifier d_j on $\mathcal{X}_j$
- Classify new instance $\mathbf{x}$ by majority vote of learned classifiers (equal weights)

Result: An **ensemble** of classifiers

Bagging Experiment
[Breiman, ML Journal, 1996]

Given sample $\mathcal{X}$ of labeled data, Breiman did the following 100 times and reported avg:

1. Divide $\mathcal{X}$ randomly into test set T (10%) and train set D (90%)
2. Learn decision tree from D and let e_S be error rate on T
3. Do 50 times: Create bootstrap set $\mathcal{X}_j$ and learn decision tree (so ensemble size = 50). Then let e_B be the error of a majority vote of the trees on T

Bagging Experiment

Results

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
waveform	29.0	19.4	33%
heart	10.0	5.3	47%
breast cancer	6.0	4.2	30%
ionosphere	11.2	8.6	23%
diabetes	23.4	18.8	20%
glass	32.0	24.9	27%
soybean	14.5	10.6	27%

Bagging Experiment
(cont'd)

Same experiment, but using a nearest neighbor classifier, where prediction of new example $\mathbf{x}$'s label is that of $\mathbf{x}$'s nearest neighbor in training set, where distance is e.g., Euclidean distance

Results

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
waveform	26.1	26.1	0%
heart	6.3	6.3	0%
breast cancer	4.9	4.9	0%
ionosphere	35.7	35.7	0%
diabetes	16.4	16.4	0%
glass	16.4	16.4	0%

What happened?

When Does Bagging Help?

When learner is **unstable**, i.e., if small change in training set causes large change in hypothesis produced

- Decision trees, neural networks
- **Not** nearest neighbor

Experimentally, bagging can help substantially for unstable learners; can somewhat degrade results for stable learners

Boosting
[Schapire & Freund Book]

Similar to bagging, but don't always sample uniformly; instead adjust resampling distribution $\mathbf{p}_j$ over $\mathcal{X}$ to focus attention on previously misclassified examples

Final classifier weights learned classifiers, but not uniform; instead weight of classifier d_j depends on its performance on data it was trained on

Final classifier is weighted combination of $d_1, \ldots, d_L$, where d_j's weight depends on its error on $\mathcal{X}$ w.r.t. $\mathbf{p}_j$

Boosting

Algorithm Idea [$\mathbf{p}_j \leftrightarrow D_j$; $d_j \leftrightarrow h_j$]

Repeat for $j = 1, \ldots, L$:

1. Run learning algorithm on examples randomly drawn from training set $\mathcal{X}$ according to distribution $\mathbf{p}_j$ ($\mathbf{p}_1 =$ uniform)
   - Can sample $\mathcal{X}$ according to $\mathbf{p}_j$ and train normally, or directly minimize error on $\mathcal{X}$ w.r.t. $\mathbf{p}_j$
2. Output of learner is binary hypothesis d_j
3. Compute $error_{\mathbf{p}_j}(d_j)$ = error of d_j on examples from $\mathcal{X}$ drawn according to $\mathbf{p}_j$ (can compute exactly)
4. Create $\mathbf{p}_{j+1}$ from $\mathbf{p}_j$ by decreasing weight of instances that d_j predicts correctly

Boosting

Algorithm Pseudocode (Fig 17.2)

Training:

For all $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$, initialize $p_1^t = 1/N$

For all base-learners $j = 1, \ldots, L$

Randomly draw $\mathcal{X}_j$ from $\mathcal{X}$ with probabilities p_j^t

Train d_j using $\mathcal{X}_j$

For each (x^t, r^t), calculate $y_j^t \leftarrow d_j(x^t)$

Calculate error rate: $\epsilon_j \leftarrow \sum_t p_j^t \cdot 1(y_j^t \neq r^t)$

If $\epsilon_j > 1/2$, then $L \leftarrow j - 1$; stop

$\beta_j \leftarrow \epsilon_j / (1 - \epsilon_j)$

For each (x^t, r^t), decrease probabilities if correct:

If $y_j^t = r^t$, then $p_{j+1}^t \leftarrow \beta_j p_j^t$ Else $p_{j+1}^t \leftarrow p_j^t$

Normalize probabilities:

$Z_j \leftarrow \sum_t p_{j+1}^t$; $p_{j+1}^t \leftarrow p_{j+1}^t / Z_j$

Testing:

Given x, calculate $d_j(x), j = 1, \ldots, L$

Calculate class outputs, $i = 1, \ldots, K$:

$$y_i = \sum_{j=1}^L \left(\log \frac{1}{\beta_j}\right) d_{ji}(x)$$

Boosting

Algorithm Pseudocode (Schapire & Freund)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t to minimalize the weighted error:

$$\epsilon_t \doteq \mathbf{Pr}_{i \sim D_t}[h_t(x_i) \neq y_i] \,.$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

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

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

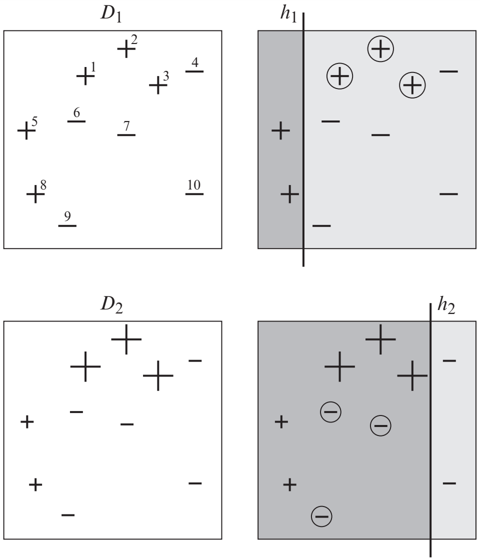
Output the final hypothesis:

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

Boosting

Schapire & Freund Example: Decision Stumps

$D_j = \mathbf{p}_j;\ h_j = d_j;\ \alpha_j = \frac{1}{2}\ln(1/\beta_j) = \frac{1}{2}\ln\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$



Boosting

Schapire & Freund Example: Decision Stumps

$D_j = \mathbf{p}_j;\ h_j = d_j;\ \alpha_j = \frac{1}{2}\ln(1/\beta_j) = \frac{1}{2}\ln\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$

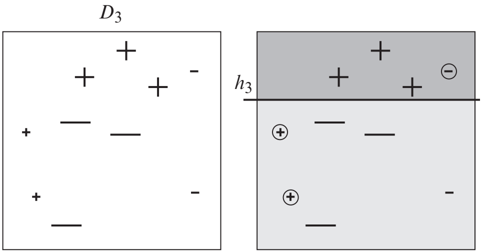
	1	2	3	4	5	6	7	8	9	10	
$D_1(i)$	$\underline{0.10}$	$\underline{0.10}$	$\underline{0.10}$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	$\epsilon_1 = 0.30, \alpha_1 \approx 0.42$
$e^{-\alpha_1 y_i h_1(x_i)}$	1.53	1.53	1.53	0.65	0.65	0.65	0.65	0.65	0.65	0.65	
$D_1(i)\ e^{-\alpha_1 y_i h_1(x_i)}$	0.15	0.15	0.15	0.07	0.07	0.07	0.07	0.07	0.07	0.07	$Z_1 \approx 0.92$
$D_2(i)$	0.17	0.17	0.17	0.07	0.07	$\underline{0.07}$	$\underline{0.07}$	0.07	$\underline{0.07}$	0.07	$\epsilon_2 \approx 0.21, \alpha_2 \approx 0.65$
$e^{-\alpha_2 y_i h_2(x_i)}$	0.52	0.52	0.52	0.52	0.52	1.91	1.91	0.52	1.91	0.52	
$D_2(i)\ e^{-\alpha_2 y_i h_2(x_i)}$	0.09	0.09	0.09	0.04	0.04	0.14	0.14	0.04	0.14	0.04	$Z_2 \approx 0.82$
$D_3(i)$	0.11	0.11	0.11	$\underline{0.05}$	$\underline{0.05}$	0.17	0.17	$\underline{0.05}$	0.17	0.05	$\epsilon_3 \approx 0.14, \alpha_3 \approx 0.92$
$e^{-\alpha_3 y_i h_3(x_i)}$	0.40	0.40	0.40	2.52	2.52	0.40	0.40	2.52	0.40	0.40	
$D_3(i)\ e^{-\alpha_3 y_i h_3(x_i)}$	0.04	0.04	0.04	0.11	0.11	0.07	0.07	0.11	0.07	0.02	$Z_3 \approx 0.69$

Calculations are shown for the ten examples as numbered in the figure. Examples on which hypothesis h_t makes a mistake are indicated by underlined figures in the rows marked D_t.

Boosting

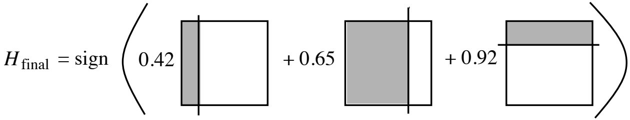
Schapire & Freund Example: Decision Stumps

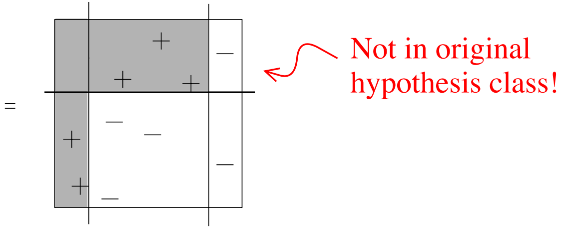
$D_j = \mathbf{p}_j$; $h_j = d_j$; $\alpha_j = \frac{1}{2}\ln(1/\beta_j) = \frac{1}{2}\ln\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$



Boosting

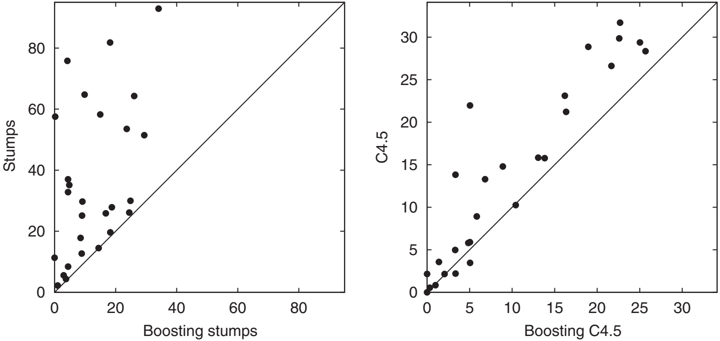
Example (cont'd)





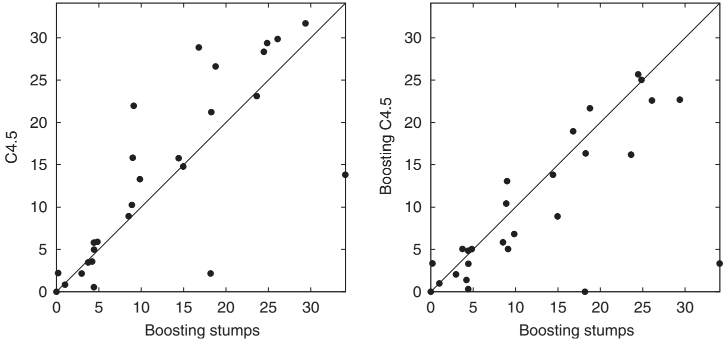
In this case, need at least two of the three hypotheses to predict $+1$ for weighted sum to exceed 0.

Scatter plot: Percent classification error of non-boosted vs boosted on 27 learning tasks



Boosting

Experimental Results (cont'd)



- If each $\epsilon_j < 1/2 - \gamma_j$, error of ensemble on $\mathcal{X}$ drops exponentially in $\sum_{j=1}^{L} \gamma_j$
- Can also bound generalization error of ensemble
- Very successful empirically
  - Generalization sometimes improves if training continues after ensemble's error on $\mathcal{X}$ drops to 0
    - Contrary to intuition: would expect overfitting
    - Related to increasing the combined classifier's **margin**
- Useful even with very simple base learners, e.g., **decision stumps**