CSCE 970 Lecture 6: System Evaluation and Combining Classifiers

Stephen D. Scott
(Adapted partially from Tom Mitchell's slides)

March 13, 2003

Introduction

- Once features generated/selected and classifier built, need to assess its performance on new data
- Assume all data drawn i.i.d. according to some prob. distribution $\mathcal{D}$ and try to estimate classifier's prediction error on new data drawn according to $\mathcal{D}$
- If error estimate unacceptable, need to select/gen. new features and/or build new classifier
  - Change features used
  - Change size/structure of neural network
  - Change assumptions in Bayesian classifier
  - Choose new learning method, e.g. decision tree

Introduction
(cont'd)

- Can't use error on training set to estimate ability to generalize, because it's too optimistic
- So use independent testing set to estimate error
- Can use statistical hypothesis testing techniques to:
  - Give confidence intervals for error estimate
  - Contrast performance of two classifiers (see if the difference in their error estimates is statistically significant)
- Sometimes need to train and test with a small data set
- Will also look at improving a classifier's performance

Outline

- Sample error vs. true error
- Confidence intervals for observed hypothesis error
- Estimators
- Binomial distribution, Normal distribution, Central Limit Theorem
- Paired t tests
- Comparing learning methods
- Combining classifiers to improve performance: Weighted Majority, Bagging, Boosting

Two Definitions of Error

- Denote the learned classifier by the hypothesis h and the target function (that labels examples) by f

- The true error of hypothesis h with respect to target function f and distribution $\mathcal{D}$ is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$.

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[f(x) \neq h(x)]$$

- The sample error of h with respect to target function f and data sample S is the proportion of examples h misclassifies

$$error_S(h) \equiv \frac{1}{|S|} \sum_{x \in S} \delta(f(x) \neq h(x))$$

Where $\delta(f(x) \neq h(x))$ is 1 if $f(x) \neq h(x)$, and 0 otherwise.

- How well does $error_S(h)$ estimate $error_{\mathcal{D}}(h)$?

Problems Estimating Error

- Bias: If S is training set, $error_S(h)$ is optimistically biased

$$bias \equiv E[error_S(h)] - error_{\mathcal{D}}(h)$$

For unbiased estimate, h and S must be chosen independently

- Variance: Even with unbiased S, $error_S(h)$ may still *vary* from $error_{\mathcal{D}}(h)$

Estimators

Experiment:

1. Choose sample S of size n according to distribution $\mathcal{D}$

2. Measure $error_S(h)$

$error_S(h)$ is a random variable (i.e., result of an experiment)

$error_S(h)$ is an unbiased estimator for $error_{\mathcal{D}}(h)$

Given observed $error_S(h)$, what can we conclude about $error_{\mathcal{D}}(h)$?

Confidence Intervals

If

- S contains n examples (feature vectors), drawn independently of h and each other
- $n \geq 30$

Then

- With approximately 95% probability, $error_{\mathcal{D}}(h)$ lies in interval

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

E.g. hypothesis h misclassifies 12 of the 40 examples in test set S:

$$error_S(h) = \frac{12}{40} = 0.30$$

Then with approx. 95% confidence, $error_{\mathcal{D}}(h) \in [0.158, 0.442]$

Confidence Intervals
(cont'd)

If

- S contains n examples, drawn independently of h and each other
- $n \geq 30$

Then

- With approximately N% probability, $error_{\mathcal{D}}(h)$ lies in interval

$$error_S(h) \pm z_N \sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

where

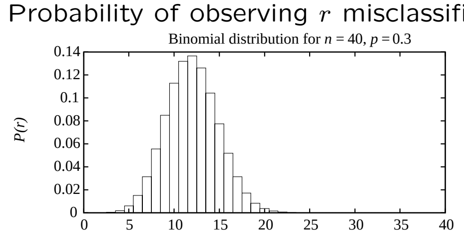
N%:	50%	68%	80%	90%	95%	98%	99%
z_N:	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Why?

$error_S(h)$ is a Random Variable

Repeatedly run the experiment, each with different randomly drawn S (each of size n)

Probability of observing r misclassified examples:

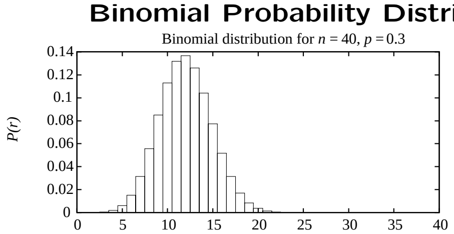


$$P(r) = \binom{n}{r} error_{\mathcal{D}}(h)^r (1 - error_{\mathcal{D}}(h))^{n-r}$$

I.e. let $error_{\mathcal{D}}(h)$ be probability of heads in biased coin, the $P(r) =$ prob. of getting r heads out of n flips

What kind of distribution is this?

Binomial Probability Distribution



$$P(r) = \binom{n}{r} p^r (1-p)^{n-r} = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

Probability $P(r)$ of r heads in n coin flips, if $p = \Pr(heads)$

- Expected, or mean value of X, $E[X]$, is

$$E[X] \equiv \sum_{i=0}^{n} iP(i) = np$$

- Variance of X is

$$Var(X) \equiv E[(X - E[X])^2] = np(1-p)$$

- Standard deviation of X, σ_X, is

$$\sigma_X \equiv \sqrt{E[(X - E[X])^2]} = \sqrt{np(1-p)}$$

Approximate Binomial Dist. with Normal

$error_S(h) = r/n$ is binomially distributed, with

- mean $\mu_{error_S(h)} = error_{\mathcal{D}}(h)$ (i.e. unbiased est.)
- standard deviation $\sigma_{error_S(h)}$

$$\sigma_{error_S(h)} = \sqrt{\frac{error_{\mathcal{D}}(h)(1 - error_{\mathcal{D}}(h))}{n}}$$

(i.e. increasing n decreases variance)

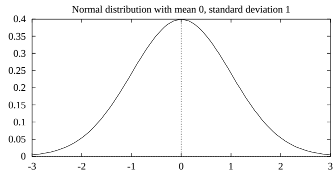
Want to compute confidence interval = interval centered at $error_{\mathcal{D}}(h)$ containing N% of the weight under the distribution (difficult for binomial)

Approximate binomial by normal (Gaussian) dist:

- mean $\mu_{error_S(h)} = error_{\mathcal{D}}(h)$
- standard deviation $\sigma_{error_S(h)}$

$$\sigma_{error_S(h)} \approx \sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

Normal Probability Distribution



$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \, \text{exp}\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

- Defined completely by μ and σ
- The probability that X will fall into the interval (a, b) is given by

$$\int_a^b p(x)dx$$

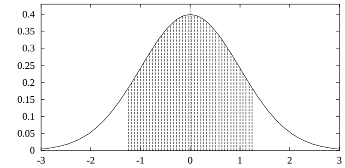
- Expected, or mean value of X, $E[X]$, is

$$E[X] = \mu$$

- Variance of X is $Var(X) = \sigma^2$
- Standard deviation of X, σ_X, is

$$\sigma_X = \sigma$$

Normal Probability Distribution
(cont'd)

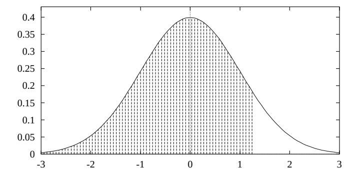


80% of area (probability) lies in $\mu \pm 1.28\sigma$

N% of area (probability) lies in $\mu \pm z_N \sigma$

N%:	50%	68%	80%	90%	95%	98%	99%
z_N:	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Can also have one-sided bounds:



N% of area lies $< \mu + z'_N \sigma$ or $> \mu - z'_N \sigma$, where $z'_N = z_{100-2(100-N)}$

N%:	50%	68%	80%	90%	95%	98%	99%
z'_N:	0.0	0.47	0.84	1.28	1.64	2.05	2.33

Confidence Intervals Revisited

If

- S contains n examples, drawn independently of h and each other
- $n \geq 30$

Then

- With approximately 95% probability, $error_S(h)$ lies in interval

$$error_{\mathcal{D}}(h) \pm 1.96\sqrt{\frac{error_{\mathcal{D}}(h)(1-error_{\mathcal{D}}(h))}{n}}$$

Equivalently, $error_{\mathcal{D}}(h)$ lies in interval

$$error_S(h) \pm 1.96\sqrt{\frac{error_{\mathcal{D}}(h)(1-error_{\mathcal{D}}(h))}{n}}$$

which is approximately

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

(One-sided bounds yield upper or lower error bounds)

Central Limit Theorem

How can we justify approximation?

Consider a set of independent, identically distributed random variables $Y_1 \ldots Y_n$, all governed by an arbitrary probability distribution with mean μ and finite variance σ^2. Define the sample mean,

$$\bar{Y} \equiv \frac{1}{n}\sum_{i=1}^{n} Y_i$$

Note that $\bar{Y}$ is itself a random variable, i.e. the result of an experiment (e.g. $error_S(h) = r/n$)

Central Limit Theorem: As $n \rightarrow \infty$, the distribution governing $\bar{Y}$ approaches a Normal distribution, with mean μ and variance σ^2/n

Thus the distribution of $error_S(h)$ is approximately normal for large n, and its expected value is $error_{\mathcal{D}}(h)$

(Rule of thumb: $n \geq 30$ when estimator's distribution is binomial, might need to be larger for other distributions)

Calculating Confidence Intervals

1. Pick parameter p to estimate
   - $error_{\mathcal{D}}(h)$
2. Choose an estimator
   - $error_S(h)$
3. Determine probability distribution that governs estimator
   - $error_S(h)$ governed by binomial distribution, approximated by normal when $n \geq 30$
4. Find interval (L, U) such that $N\%$ of probability mass falls in the interval
   - Could have $L = -\infty$ or $U = \infty$
   - Use table of z_N or z'_N values

Difference Between Hypotheses

Test h_1 on sample S_1, test h_2 on S_2

1. Pick parameter to estimate

$$d \equiv error_{\mathcal{D}}(h_1) - error_{\mathcal{D}}(h_2)$$

2. Choose an estimator

$$\hat{d} \equiv error_{S_1}(h_1) - error_{S_2}(h_2)$$

(unbiased)

3. Determine probability distribution that governs estimator (difference between two normals is also normal, variances add)

$$\sigma_{\hat{d}} \approx \sqrt{\frac{error_{S_1}(h_1)(1 - error_{S_1}(h_1))}{n_1} + \frac{error_{S_2}(h_2)(1 - error_{S_2}(h_2))}{n_2}}$$

4. Find interval (L, U) such that $N\%$ of prob. mass falls in the interval: $\hat{d} \pm Z_n \sigma_{\hat{d}}$

Paired t test to compare h_A,h_B

1. Partition data into k disjoint test sets $T_1, T_2, \ldots, T_k$ of equal size, where this size is at least 30.
2. For i from 1 to k, do

$$\delta_i \leftarrow error_{T_i}(h_A) - error_{T_i}(h_B)$$

3. Return the value $\bar{\delta}$, where

$$\bar{\delta} \equiv \frac{1}{k} \sum_{i=1}^{k} \delta_i$$

$N\%$ confidence interval estimate for d:

$$\bar{\delta} \pm t_{N,k-1}\ s_{\bar{\delta}}$$

$$s_{\bar{\delta}} \equiv \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} \left(\delta_i - \bar{\delta}\right)^2}$$

t (student's t dist. with $k-1$ degrees of freedom) plays role of z, s plays role of σ

t test gives more accurate results since std. deviation approximated and test sets not independent

Comparing Learning Algorithms L_A and L_B

What we'd like to estimate:

$$E_{S \subset \mathcal{D}}[error_{\mathcal{D}}(L_A(S)) - error_{\mathcal{D}}(L_B(S))]$$

where $L(S)$ is the hypothesis output by learner L using training set S

I.e., the expected difference in true error between hypotheses output by learners L_A and L_B, when trained using randomly selected training sets S drawn according to distribution $\mathcal{D}$

But, given limited data D_0, what is a good estimator?

- Could partition D_0 into training set S_0 and testing set T_0, and measure

$$error_{T_0}(L_A(S_0)) - error_{T_0}(L_B(S_0))$$

- Even better, repeat this many times and average the results (next slide)

Comparing learning algorithms L_A and L_B (cont'd)

1. Partition data D_0 into k disjoint test sets $T_1, T_2, \ldots, T_k$ of equal size, where this size is at least 30.
2. For i from 1 to k, do

   (use T_i for the test set, and the remaining data for training set S_i)

   - $S_i \leftarrow \{D_0 - T_i\}$
   - $h_A \leftarrow L_A(S_i)$
   - $h_B \leftarrow L_B(S_i)$
   - $\delta_i \leftarrow error_{T_i}(h_A) - error_{T_i}(h_B)$
3. Return the value $\bar{\delta}$, where

$$\bar{\delta} \equiv \frac{1}{k} \sum_{i=1}^{k} \delta_i$$

Comparing learning algorithms L_A and L_B (cont'd)

- Notice we'd like to use the paired t test on $\bar{\delta}$ to obtain a confidence interval
- Not really correct, because the training sets in this algorithm are not independent (they overlap!)
- More correct to view algorithm as producing an estimate of

$$E_{S \subset D_0}[error_{\mathcal{D}}(L_A(S)) - error_{\mathcal{D}}(L_B(S))]$$

instead of

$$E_{S \subset \mathcal{D}}[error_{\mathcal{D}}(L_A(S)) - error_{\mathcal{D}}(L_B(S))]$$

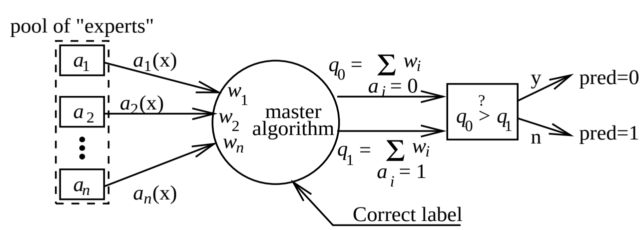
- But even this approximation is better than no comparison

Combining Classifiers

- Sometimes a single classifier (e.g. neural network, decision tree) won't perform well, but a weighted combination of them will
- Each classifier (or expert) in the pool has its own weight
- When asked to predict the label for a new example, each expert makes its own prediction, and then the master algorithm combines them using the weights for its own prediction (i.e. the "official" one)
- If the classifiers themselves cannot learn (e.g. heuristics) then the best we can do is to learn a good set of weights
- If we are using a learning algorithm (e.g. NN, dec. tree), then we can rerun the algorithm on different subsamples of the training set and set the classifiers' weights during training

Weighted Majority Algorithm (WM)

[Mitchell, Sec. 7.5.4]



Weighted Majority Algorithm (WM) (cont'd)

a_i is ith pred. algorithm in pool A of algs; each alg is arbitrary function from X to $\{0,1\}$ or $\{-1,1\}$

w_i is weight the master alg associates with a_i

$\beta \in [0,1)$ is parameter

- $\forall i$ set $w_i \leftarrow 1$
- For each training example (or trial) $\langle x, c(x) \rangle$
  - Set $q_0 \leftarrow q_1 \leftarrow 0$
  - For each algorithm a_i
    - If $a_i(x) = 0$ then $q_0 \leftarrow q_0 + w_i$ else $q_1 \leftarrow q_1 + w_i$
    - If $q_1 > q_0$ then predict 1 for $c(x)$, else predict 0 (case for $q_1 = q_0$ is arbitrary)
    - For each $a_i \in A$
      - If $a_i(x) \neq c(x)$ then $w_i \leftarrow \beta w_i$

Setting $\beta = 0$ yields Halving algorithm over A

Weighted Majority Mistake Bound (On-Line Model)

- Let $a_{opt} \in A$ be expert that makes fewest mistakes on arbitrary sequence S of exs; let $k =$ its number of mistakes
- Let $\beta = 1/2$ and $W_t = \sum_{i=1}^{n} w_{i,t} =$ sum of wts at trial t ($W_0 = n$)
- On trial t such that WM makes a mistake, the total weight reduced is

$$W_t^{mis} = \sum_{a_i(x_t) \neq c(x_t)} w_i \geq W_t/2$$

so

$$W_{t+1} = \left(W_t - W_t^{mis}\right) + W_t^{mis}/2 = W_t - W_t^{mis}/2 \leq 3W_t/4$$

- After seeing all of S, $w_{opt,|S|} = (1/2)^k$ and $W_{|S|} \leq n(3/4)^M$ where $M =$ total number of mistakes, yielding

$$\left(\frac{1}{2}\right)^k \leq n\left(\frac{3}{4}\right)^M,$$

so

$$\boxed{M \leq \frac{k + \log_2 n}{-\log_2(3/4)} \leq 2.41\,(k + \log_2 n)}$$

Weighted Majority Mistake Bound (cont'd)

- Thus for any arbitrary sequence of examples, WM guaranteed to not perform much worse than best expert in pool plus log of number of experts
  - Implicitly agnostic
- Other results:
  - Bounds hold for general values of $\beta \in [0,1)$
  - Better bounds hold for more sophisticated algorithms, but only better by a constant factor (worst-case lower bound: $\Omega\,(k + \log n)$)
  - Get bounds for real-valued labels and predictions
  - Can track shifting concept, i.e. where best expert can suddenly change in S; key: don't let any weight get too low relative to other weights, i.e. don't overcommit

Bagging Classifiers [Breiman, ML Journal, '96]

Bagging = Bootstrap aggregating

Bootstrap sampling: given a set D containing m training examples:

- Create D_i by drawing m examples uniformly at random with replacement from D
- Expect D_i to omit $\approx 37\%$ of examples from D

Bagging:

- Create k bootstrap samples $D_1, \ldots, D_k$
- Train a classifier on each D_i
- Classify new instance $x \in X$ by majority vote of learned classifiers (equal weights)

Bagging Experiment
[Breiman, ML Journal, '96]

Given sample S of labeled data, Breiman did the following 100 times and reported avg:

1. Divide S randomly into test set T (10%) and training set D (90%)
2. Learn decision tree from D and let e_S be its error rate on T
3. Do 50 times: Create bootstrap set D_i, learn decision tree and let e_B be the error of a majority vote of the trees on T

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 feature vector $\mathbf{x}$'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 Classifiers
[Freund & Schapire, ICML '96; many more]

Similar to bagging, but don't always sample uniformly; instead adjust resampling distribution over D to focus attention on previously misclassified examples

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

Repeat for $t = 1, \ldots, T$:

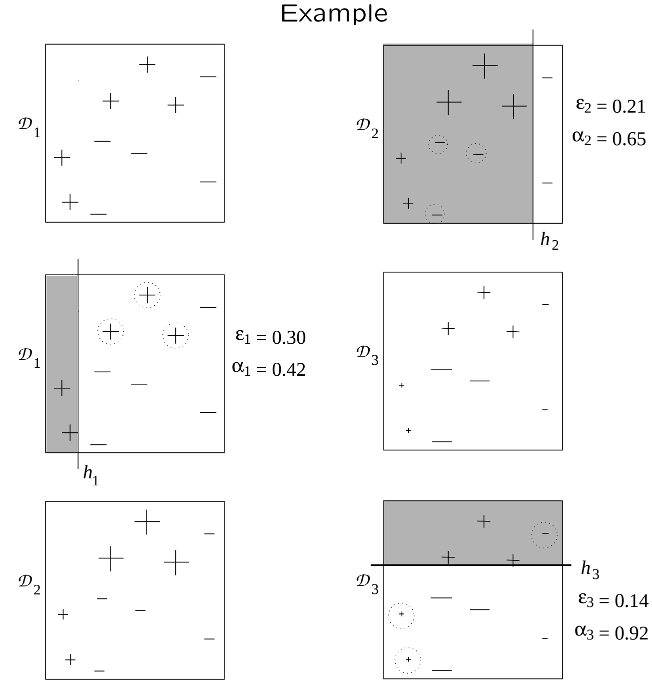
1. Run learning algorithm on examples randomly drawn from training set D according to distribution $\mathcal{D}_t$ ($\mathcal{D}_1 =$ uniform)
2. Output of learner is hypothesis $h_t : X \to \{-1, +1\}$
3. Compute expected error of h_t on examples drawn according to $\mathcal{D}_t$ (can compute exactly)
4. Create $\mathcal{D}_{t+1}$ from $\mathcal{D}_t$ by increasing weight of examples that h_t mispredicts

Final classifier is weighted combination of $h_1, \ldots, h_T$, where h_t's weight depends on its error w.r.t. $\mathcal{D}_t$

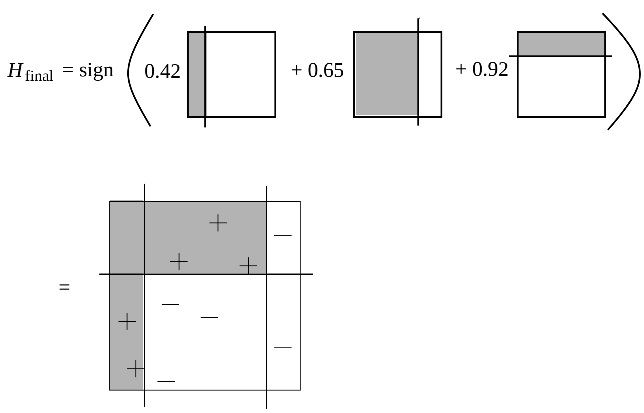
Boosting
(cont'd)

- Preliminaries: $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, $y_i \in \{-1, +1\}$, $\mathcal{D}_t(i) =$ weight of $(\mathbf{x}_i, y_i)$ under $\mathcal{D}_t$
- Initialization: $\mathcal{D}_1(i) = 1/m$
- Error Computation: $\epsilon_t = \Pr_{\mathcal{D}_t}[h_t(\mathbf{x}_i) \neq y_i]$
  (easy to do since we know $\mathcal{D}_t$)
- If $\epsilon_t > 1/2$ then halt; else:
- Weighting Factor: $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$
  (grows as ϵ_t decreases)
- Update: $\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i) \exp(-\alpha_t \, y_i \, h_t(\mathbf{x}_i))}{\underbrace{Z_t}_{\text{normalization factor}}}$
  (increase wt. of mispredicted exs, decr. wt of correctly pred.)
- Final Hypothesis: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \, h_t(\mathbf{x}) \right)$
  (ϵ_t large $\Rightarrow$ flip h_t's prediction strongly)

Boosting
Example



Boosting
Example (cont'd)



Boosting
Miscellany

- If each $\epsilon_t < 1/2 - \gamma_t$, error of $H(\cdot)$ on D drops exponentially in $\sum_{t=1}^{T} \gamma_t$ (what does γ correspond to?)
- Can also bound generalization error of $H(\cdot)$ independent of T
- Also successful empirically on neural network and decision tree learners
  - Empirically, generalization sometimes improves if training continues after $H(\cdot)$'s error on D drops to 0
  - Contrary to intuition; expect overfitting
  - Related to increasing the combined classifier's margin (confidence in prediction)

Topic summary due in 1 week!