Ensemble Methods and Boosting
Ensemble Methods:

• Use set of hypothesis
Ensemble Methods

• Use “ensemble” or group of hypotheses
• Diversity important
  – ensemble of “yes-men” is useless
• Get diverse hypotheses by:
  – Using different data
  – Using different algorithms
  – Using different hyper-parameters
Averaging reduces Variance

• Ensemble more stable than individual
• Consider biased coin (1/3, 2/3)
• Variance of number of heads $H$:
  – $\text{Variance} = E[H - E[H]]^2 = E[H^2] - E[H]^2$
  – of 1 flip $= 1/3 - 1/9 = 2/9$
  – sum of 2 flips $= 8/9 - 4/9 = 4/9$
  – Ave of 2 flips $= 2/9 - 1/9 = 1/9$
• Ensembles often dependent
How to get different data?

- Use different training sets
  - Bootstrap sample: pick $m$ examples from sample with replacement
  - Cross-validation sampling
  - Re-weight data

- Use different features
  - Random forests “hide” some features
How does Ensemble predict?

• Dictator – why have ensemble?
• Unweighted vote
• Weighted vote
  – Pay more attention to better predictors
• Cascade

• Other structures…
Mixture of Experts

Voting where weights are input-dependent (gating)

\[ y = \sum_{j=1}^{L} w_j d_j \]

(Jacobs et al., 1991)
Stacking

- Combiner $f()$ is another learner (Wolpert, 1992)
Boosting is:

• A meta learning technique, it “boosts” the performance of another learning algorithm

• An ensemble technique, it creates a set of classifiers and predicts with a weighted vote

• Ensembles used before to reduce variance (e.g. bagging)
Generic Boosting algorithm

Uses training sample S (±1 labels), learning algorithm L

1. for t from 1 to T do
   1. Create distribution $D_t$ (or $p_t$ in book) on S
   2. Call L with $D_t$ on S to get hypothesis $h_t$ ($d_t$ in book)
   3. Calculate weight $\alpha_t$ for $h_t$

2. Final hypothesis is $H(x) = \sum_t \alpha_t h_t(x)$
   or $H(x) = \arg\max_y \sum_t: h_t(x) = y \alpha_t$

The trick is picking $D_t$ and $w_t$
Example:

- Learn with axis-parallel half-spaces

- Majority of three hypotheses is perfect
AdaBoost (Freund&Schapire ‘97)

• Very powerful algorithm coming out of machine learning theory
• (Un-normalized) **margin**, $m_i$, of an example $(x_i,y_i) \in S$ is $y_i H(x_i) = y_i \sum_t \alpha_t h_t(x_i)$
• AdaBoost uses exponential “potential”:
  $$\text{potential} = \sum_{i \in S} e^{-m_i}$$
• This potential is smooth overestimate on number of mistakes
• Also use margins at each iteration
AdaBoost

Generate a sequence of base-learners each focusing on previous one’s errors (Freund and Schapire, 1996)

Training:
For all \( \{x^t, r^t\}_{t=1}^N \in \mathcal{X} \), initialize \( p^t_1 = 1/N \)
For all base-learners \( j = 1, \ldots, L \)
  Randomly draw \( \mathcal{X}_j \) from \( \mathcal{X} \) with probabilities \( p^t_j \)
  Train \( d_j \) using \( \mathcal{X}_j \)
For each \( (x^t, r^t) \), calculate \( y^t_j \leftarrow d_j(x^t) \)
Calculate error rate: \( \varepsilon_j \leftarrow \sum_t p^t_j \cdot 1(y^t_j \neq r^t) \)
If \( \varepsilon_j > 1/2 \), then \( L \leftarrow j - 1; \) stop
\( \beta_j \leftarrow \varepsilon_j/(1 - \varepsilon_j) \)
For each \( (x^t, r^t) \), decrease probabilities if correct:
  If \( y^t_j = r^t \) \( p^t_{j+1} \leftarrow \beta_j p^t_j \)
  Else \( p^t_{j+1} \leftarrow p^t_j \)
Normalize probabilities:
\( Z_j \leftarrow \sum_t p^t_{j+1} \)
\( p^t_{j+1} \leftarrow p^t_{j+1}/Z_j \)

Testing:
Given \( x \), calculate \( d_j(x), j = 1, \ldots, L \)
Calculate class outputs, \( i = 1, \ldots, K \):
\[
y_i = \sum_{j=1}^L \left( \log \frac{1}{\beta_j} \right) d_{ji}(x)
\]
Analysis of Training Error

• (Use overhead)
AdaBoost training error

- Let \( \gamma_t = \frac{1}{2} - \varepsilon_t \), \( \gamma_t \) is the edge of \( h_t \) over random guessing
- Let \( N \) be the number of examples in the sample
- The number of mistakes of the final hypothesis on the sample is at most: \( N \exp(-2\sum_t \gamma_t) \)
- If \( \gamma_t > \text{const} \), this decreases exponentially, and \( O(\log(N)) \) iterations to perfection (on training set)

- Generalization error bound (approximately): training error + \( O((Td/N)^{1/2}) \)
  where \( T = \# \) boosting rounds, \( d = \text{VC-dim of } h_t \)'s
• AdaBoost does constrained gradient descent on potential: at each iteration
  – $D_t$ chosen to be (proportional to) negative gradient of potential with respect to current margins: $D_t(x_i)$ is how much an increase in $x_i$’s margin helps the potential decrease
  – $D_t(x_i)$ set prop.to. $e^{-m_i}$ (current margins)
  – $w_t$ for $h_t$ is how much of $h_t$ to add in to minimize potential,
  – error $\varepsilon_t = \mathbb{P}_{i \sim D_t} [h_t(x_i) \neq y_i]$, $\alpha_t = \ln((1-\varepsilon_t)/\varepsilon_t)/2$
Contours are where \( \sum(e^{-m_i}) \) is constant
Boosting Overviews at:
http://www.cs.princeton.edu/~schapire/boost.html

Boosting Performance (vs decision tree algorithm) on UCI benchmarks - each dot is one benchmark
AdaBoost and Overfitting

- AdaBoost somewhat resists overfitting
- Test error even improves after perfect on sample - big surprise!

Boosting C4.5 on letter dataset (see Schapire’02 overview on his web page)
Generalization error

- **Normalized margin** is $m'_i = m_i / \sum_t w_t$
- Generalization error rate is, for all $\theta > 0$, (approximately) at most:
  - fraction of sample with $m_i < \theta$
  - Plus $O((d/N)^{1/2} / \theta)$
- Independent of $T$ (no over-fitting!)
- But… bounds loose, over-fitting can happen
Margin explanation

Margin distribution
After 5, 100, and 1000 (solid) Iterations.
Boosting and noise

- Exponential weighting $e^{-m_i}$ can put exponential emphasis on noisy examples
- One Solution: logitboost bounds weights: $D_t(x_i)$ prop to $1/(1 + e^{m_i})$
Boosting Successes

- Many: OCR (boosted nets), Image retrieval, Natural language processing, (see Schapire’02)

- Text classification

![Graph showing error rate versus number of classes for different methods in text classification. The methods include AdaBoost, Sleeping-experts, Rocchio, Naive-Bayes, and PrTFIDF. The graph shows a trend where error rate generally increases with the number of classes. The x-axis represents the number of classes ranging from 3 to 6, and the y-axis represents the error rate ranging from 0 to 16. The labels for the methods are indicated near the legend at the bottom of the graph.](image-url)
Boosting Variants

- Boosting by reweighting, resampling, filtering
- Logitboost (Friedman, Hastie, Tibshirani)
- Totally corrective boosting (adjust all votes)
- LP boost, dual methods (e.g. Manfred’s work)
- Info Boost (Aslam)
- Many, many others
Other Points:

- Boosting connection to game theory
- Boosting by filtering
- Boosting for outlier detection?
- Boosting for multiple classes - three ways
  - AdaBoost.m1 requires really good weak learner
  - AdaBoost.m2 more reasonable
  - ECOC techniques transforms multi-class to binary
Boosting and Regression

- Can’t just re-weight examples
- Convert to classification in AdaBoost paper
- Re-label points with residual of error
- Re-label points with signs of residual and weight with gradient of two-sided potential: \( \exp(-r) + \exp(r) - 2 \)
  (see Duffy and Helmbold)
Boosting Summary

• Model: Linear threshold of base hypotheses - depends on underlying learning algorithm
• Data: depends on the underlying learning algorithm
• Interpretable? Yes if decision stumps, but usually not
• Missing values? Depends on underlying learning alg
• Noise/outliers? AdaBoost bad, logitboost a bit better
• Irrelevant features? Depends on the underlying alg
• Comp. efficiency? Good, if underlying alg good
Interpreting AdaBoost hypotheses (Decision Stumps) on a LIDAR classification problem (2 of 5 features)

Blue = buildings, Green = trees, Brown = road, Yellow = Grass