Ensembles of Classifiers

- **Traditional approach**: Use one classifier
- **Can one do better?**
- **Approaches**:
  - Cross-validated committees
  - Bagging
  - Boosting
  - Stacking

---

Ensembles of Classifiers

- **Assume**
  - Errors are independent (suppose 30% error)
  - Majority vote
- **Probability that majority is wrong...**
  - Area under binomial distribution
  - If individual area is 0.3
  - Area under curve for ≥11 wrong is 0.026
  - Order of magnitude improvement!

---

Bagging Example

- **BAGGing = Bootstrap AGGregation**
  - (Breiman, 1996)
  - For \( m = 1, 2, ..., M \):
    - \( B_m \leftarrow \) randomly select \( N \) training instances with replacement
    - \( C_m \leftarrow \) learn\( (B_m) \) [ID3, NB, kNN, neural net, ...]
  - Combine the \( C_m \) together
    - Uniform voting \( (\alpha_m = 1/K \) for all \( i \))

---

\[ \text{CART decision boundary} \]
**Ensemble Creation III**

**Boosting – Incorrect Version**

- Maintain prob distribution over set of training ex
- Create M sets of training data iteratively:
  - On iteration \( m \)
    - Draw \( m \) examples randomly (like bagging)
    - But use probability distribution to bias selection
    - Train classifier number \( i \) on this training set
    - Modify distribution: increase \( P \) of each error example
    - Assign confidence to classifier \( I = f(error) \)
- Create harder and harder learning problems...
- "Bagging with optimized choice of examples"

**Bagging vs Boosting**

**Bias, Variance, and Noise**

- **Variance:** \( E[ (h(x*) - \hat{h}(x*))^2 ] \)
  Describes how much \( h(x*) \) varies from one training set \( S \) to another
- **Bias:** \( [h(x*) - f(x*)] \)
  Describes the average error of \( h(x*) \).
- **Noise:** \( E[ (y* - f(x*))^2 ] = E[\epsilon^2] = \sigma^2 \)
  Describes how much \( y* \) varies from \( f(x*) \)

© Daniel S. Weld 10

© Daniel S. Weld 11
Bias / Variance Tradeoff

Decreasing bias increases variance
Want the best compromise
**1st Order Polynomial**

- Image of a 1st order polynomial graph.

**9th Order Polynomial**

- Image of a 9th order polynomial graph.

---

**Regularization**

\[
\overline{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2} ||\mathbf{w}||^2
\]

Penalize large coefficient values

Increasing \( \lambda \) trades bias for variance

---

**Regularization: \( \ln \lambda = -18 \)**

- Image of a graph showing regularization with \( \ln \lambda = -18 \).

**Regularization: \( \ln \lambda = 0 \)**

- Image of a graph showing regularization with \( \ln \lambda = 0 \).

**Regularization: \( E_{RMS} vs. \ln \lambda \)**

- Image of a graph showing \( E_{RMS} \) vs. \( \ln \lambda \).
Punchline

- Ensembles trade bias & variance
- Bagging reduces variance (bias almost unchanged)
  - Use with low bias learner, eg full decision tree
  - (Bagging decision stumps performs poorly)
- Boosting can reduce both
  - Often used with high bias learners, eg decision stumps

Boosting

Idea: run weak learner multiple times on (reweighted!) training data; weight learned classifiers \( \propto \) their accuracy

On each iteration \( t \):
- Learn a hypothesis, \( h_t \), using distribution to weight examples
- Compute a strength for this hypothesis \( \alpha_t \)
- Reweight training examples by how well they were classified

Final classifier:

\[
\hat{h}(x) = \text{sign} \left( \sum_{t} \alpha_t h_t(x) \right)
\]

- Practically useful
- Theoretically interesting

Boosting Applet

http://cseweb.ucsd.edu/~yfreund/adaboost/index.html
Learning from weighted data

- Consider a weighted dataset
  - $D(j)$ – weight of $j$th training example $(x,y)$
  - Interpretations:
    - $j$th training example counts as if it occurred $D(j)$ times
    - If I were to “resample” data, I would get more samples of “heavier” data points
  - Now, always do weighted calculations:
    - e.g., MLE for Naïve Bayes, redefine $\text{Count}(Y=y)$ to be weighted count:
      \[
      \text{Count}(Y = y) = \sum_{j=1}^{n} D(j) \delta(Y^j = y)
      \]
      where $\delta$ is the Kronecker delta function.

\[
\text{Final Result: linear sum of “base” or “weak” classifier outputs.}
\]
What \( \alpha_t \) to choose for hypothesis \( h_t \)?

Idea: choose \( \alpha_t \) to minimize a bound on training error!

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

Where

\[
f(x) = \sum_{t} \alpha_t h_t(x); H(x) = \text{sign}(f(x))
\]

This equality isn’t obvious! Can be shown with algebra (telescoping sums).

If we minimize \( \prod Z_t \), we minimize our training error!!!

- We can tighten this bound greedily, by choosing \( \alpha_t \) and \( h_t \) on each iteration to minimize \( Z_t \).
- \( h_t \) is estimated as a black box, but can we solve for \( \alpha_t \)?

Summary: choose \( \alpha_t \) to minimize error bound

We can squeeze this bound by choosing \( \alpha_t \) on each iteration to minimize \( Z_t \):

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

For boolean \( Y \): differentiate, set equal to 0, there is a closed form solution! [Freund & Schapire ’97]:

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

Why Does Boosting Work So Well?

- On each iteration:
  - Learn a classifier, \( h_t \), using distribution to weight examples
  - Compute a strength for this classifier – \( \alpha_t \)
  - Reweight training examples by how well they were classified

- Final classifier:

\[
h(x) = \text{sign} \left( \sum_{t} \alpha_t h_t(x) \right)
\]

- Look familiar?
  - Another linear model, except...
  - Not in terms of original features
  - Creates new features (classifiers, \( h_t \)) while it learns weights

Strong, weak classifiers

- If each classifier is (at least slightly) better than random: \( \epsilon_t < 0.5 \)
- Another bound on error:

\[
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \prod_{i=1}^{t} Z_i \leq \exp \left( 2 \sum_{i=1}^{t} (1/2 - \epsilon_i)^2 \right)
\]

- What does this imply about the training error?
  - Will reach zero!
  - Will get there exponentially fast!
Boosting results – Digit recognition
[Schapire, 1989]

- Boosting:
  - Seems to be robust to overfitting
  - Test error can decrease even after training error is zero!!

Boosting: Experimental Results
[Freund & Schapire, 1996]

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

What you need to know about Boosting

- Combine weak classifiers to get very strong classifier
  - Weak classifier – slightly better than random on training data
  - Resulting very strong classifier – can get zero training error
- AdaBoost algorithm
- Boosting v. Logistic Regression
  - Both linear model, boosting “learns” features
  - 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