Committee of Networks

- Train several networks and form a committee to avoid overtraining

- Approaches
  - Ensemble averaging
  - Boosting
    - Filtering
    - Subsampling
Committee of Networks

INPUT

Network1

Network2

Network3

Combiner

OUTPUT
Learning Methodologies

- **Strong Learning:** PAC learning model (error rate and confidence parameter set arbitrarily close to 0)
- **Weak Learning:** Learn with error rate $\varepsilon < \frac{1}{2}$
- Are notions of strong learning and weak learning equivalent? (Yes)
Filtering: Training with 3 Experts

- First expert trained with \( N_1 \) examples
- Second expert trained with \( N_2 \) examples and use first expert to filter examples
  - Flip fair coin, if heads add misclassified example
  - Otherwise add correctly classified example
  - Continue until have \( N_1 \) examples
- Third expert trained with \( N_3 \) examples add example when first and second experts disagree, continue until additional \( N_1 \) examples drawn
Boosting by Filtering Comments

- Total number of examples needed: $N = N_1 + N_2 + N_3$
- Total computational cost: $3N_1$ examples
- Learn on tougher examples: lose error bound if error on first expert is $\varepsilon < \frac{1}{2}$ then overall error rate is $< 3 \varepsilon^2 - 2 \varepsilon^3$
- To get lower error rate filter more examples
Adaboost (resample)

- Boosting by filtering (takes too many examples)
- Adaboost use same training examples, but change distribution on which you sample
- Adaboost is simple to implement and used on many applications and different types of learning machines
Boosting Algorithm

- **Input**: $S=\{(x(i), d(i)), 1 \leq i \leq m\}$
- **Initialization**: Choose distribution $D_1$ that picks inputs equally likely and let $0 < \gamma < \frac{1}{2}$ be the weak learning rate (i.e. algorithm produces an error rate less than $\frac{1}{2} - \gamma$)
- **Iterate**: for $n=1 \ldots T$
  - Call weak learning algorithm $\mathcal{L}$ with examples chosen from distribution $D_n$
  - Get outputs from $h_n : X \mapsto Y$
  - Calculate error $\varepsilon_n = \frac{1}{2} - \gamma_n \leq \frac{1}{2} - \gamma$
  - Update $D_{n+1}$ based on training errors
- **Produce hypothesis $h$ from $h_1, \ldots, h_T$**
Adaboost algorithm

- Choosing $D_n$

$$D_{n+1}(x(i)) = D_n(x(i))\exp(-\alpha_n h_n(x(i))d(i))/Z_n$$
where $Z_n$ is a normalization constant.

- Output hypothesis

$$h(x) = \text{sgn} \left( \sum_{n=1:T} \alpha_n h_n(x) \right)$$
Empirical error rate of Adaboost

Theorem: Adaboost produces an empirical error rate of
\[ \text{error}(h(S)) \leq \exp (-2\gamma^2 T) \]

Comment: \( D_{T+1}(x(i)) = \exp (- f(x(i))d(i))/(m \prod_{n=1:T} Z_n) \) where
\[ f(x(i)) = \sum_{n=1:T} \alpha_n h_n(x(i)) \]

Lemma 1: \( \text{error}(h(S)) \leq \prod_{n=1:T} Z_n \)

Lemma 2: \( Z_n \leq 2 (\varepsilon_n (1- \varepsilon_n ))^{1/2} \)
Proof or Lemma 1

\[
\text{error}(h(S)) = (1/m) \sum_{i=1:m} 1(d(i) \neq h(x(i))) \\
= (1/m) \sum_{i=1:m} 1(d(i)f(x(i) \leq 0) \\
\leq (1/m) \sum_{i=1:m} \exp(-d(i)f(x(i))) \\
= (1/m) \sum_{i=1:m} D_{T+1}(x(i)) \cdot m \prod_{n=1:T} Z_n \\
= \prod_{n=1:T} Z_n
\]

Inequality is by bounding indicator function by exponential function, next to last equality is from Comment, and last equality is by interchanging product and sum.
Proof of Lemma 2

\[ Z_n = \sum_{i=1}^{m} D_n(x(i)) \exp(-\alpha_n d(i) h_n(x(i))) \]

\[ = \sum_{i:d(i) \neq h_n(x(i))} D_n(x(i)) \exp(\alpha_n) \]
\[ + \sum_{i:d(i) = h_n(x(i))} D_n(x(i)) \exp(-\alpha_n) \]

\[ = \varepsilon_n \exp(\alpha_n) + (1 - \varepsilon_n) \exp(-\alpha_n) \]

Choose optimal value for weighting term \( \alpha_n \) to get that

\[ \alpha_n = \frac{1}{2} \log \left( \frac{(1 - \varepsilon_n)}{\varepsilon_n} \right) \]

which gives us desired inequality that

\[ Z_n \leq 2 \left( \varepsilon_n \left(1 - \varepsilon_n \right) \right)^{\frac{1}{2}} \]
Proof of Theorem

Combine Lemmas 1 and 2 to get that
\[\text{error}(h(S)) \leq \prod_{n=1:T} 2 (\varepsilon_n (1 - \varepsilon_n))^{1/2} = \prod_{n=1:T} (1 - 4 \gamma_n^2)^{1/2} \leq \prod_{n=1:T} \exp(-2 \gamma_n^2) \leq \exp(-2 \gamma^2 T)\]

If we choose \(T = 1/(2 \gamma^2) \log(m)\), then training error rate is 0.

Lemma 2 also chooses the best weighting factor \(\alpha_n = \frac{1}{2} \log((1 - \varepsilon_n)/\varepsilon_n)\).
References