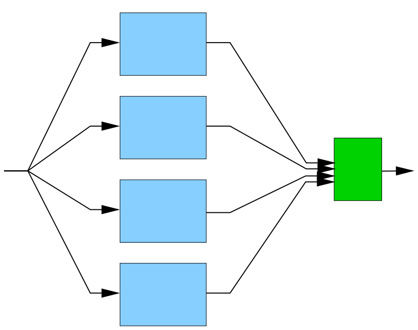
Boosting

Ensemble Method

Combining hypotheses from several learners



Terminology

Weak Learner

Learning algorithm capable of always producing hypotheses that perform *better than chance*.

Strong Learner

Learning algorithm capable of producing hypotheses that perform *arbitrarily well*.

A *strong learner* can always be constructed by combining multiple instances of any *weak learner*.

- Instead of inventing very good learning algorithms we can use multiple simple ones

Revival of simple learning algorithms

- Single layer perceptrons
- Limited height trees ("Stumping")
- Naïve Bayesian classifiers

Bagging — Bootstrap Aggregating

- Given a set of training examples D
- Form new sample sets D_i by randomly sampling D
- Use any weak classifier to get hypotheses: $D_i \to h_i$
- Create an aggregated classifier $h^\star$ which delegates to all h_i and returns the majority vote

Boosting

- Evaluate each learner on the training data
- Force next learner to concentrate on hard examples
- Weighted majority vote, based on performance

AdaBoost — Adaptive Boosting

- Most common variant of boosting
- Assign a weight w_i to each training example
- Use a weak learner which pays more attention to high-weight examples
- Increase w_i for examples which are incorrectly classified

1. Initialize weights $w_i = \frac{1}{N} \quad \forall$ examples i
2. Repeat until $t = T \vee \epsilon_t \geq 0.5$
   1. Train weak classifier: $\{D, \vec{w}\} \to h_t$
   2. Evaluate the performance of h_t:

      $$\epsilon_t \leftarrow \sum_{i:h(x_i)\neq t_i} w_i \qquad \text{set } \alpha_t \leftarrow \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$$

   3. Update weights:

      $$w_i \leftarrow \begin{cases} w_i e^{-\alpha_t} & \text{for correctly classified points} \\ w_i e^{\alpha_t} & \text{for incorrectly classified points} \end{cases}$$

   4. Normalize weights: $w_i \leftarrow \frac{w_i}{\sum_j w_j}$
3. Final classifier: $h^\star(x) \equiv \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

- Adding more learners reduces training error
- Risk of overlearning
- In practice, this does not happen!
- Recent theoretical finding: AdaBoost tends to maximize margins