Majority Voting

- A randomly chosen hyperplane has an expected **error of 0.5**.
- Many random hyperplanes combined by **majority vote** will still be random.
- Suppose we have m classifiers, performing slightly better than random, that is ***error*** **= 0.5-ε**.
- Combine: make a decision based on majority vote?
- What if we combined these m **slightly-better-than-random** classifiers? Would majority vote be a good choice?

Condorcet's Jury Theorem

Marquis de Condorcet Application of Analysis to the Probability of Majority Decisions. 1785.

Assumptions:

1. Each individual makes the right choice with a probability p.
2. The votes are independent.

If $p > 0.5$, then adding more voters increases the probability that the majority decision is correct. if $p < 0.5$, then adding more voters makes things worse.

Ensemble Methods

- An **Ensemble Method** combines the predictions of many individual classifiers by majority voting.
- Such individual classifiers, called **weak learners**, are required to perform slightly better than random.

How do we produce independent weak learners using the same training data?

- Use a **strategy** to obtain relatively independent weak learners!
- Different methods:
  1. Boosting → 自适应加权，串行
  2. Bagging → 随机加权，并行
  3. Random Forests

Boosting

- First ensemble method.
- One of the most powerful Machine Learning methods.
- Popular algorithm: AdaBoost.
- Simple algorithm.
- Weak learners can be trees, perceptrons, decision stumps, etc.
- **Idea:**

  Train the weak learners on weighted training examples.

- The predictions from all of the G_m, $m \in \{1, ..., M\}$ are combined with a weighted majority voting.
- α_m is the contribution of each weak learner G_m.
- Computed by the boosting algorithm to give a weighted importance to the classifiers in the sequence.
- The decision of a highly-performing classier in the sequence should weight more than less important classifiers in the sequence.
- This is captured in:

$$G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$$

The error rate on the training sample:

$$err := \frac{\sum_{i=1}^{n} 1\{y_i \neq G(x_i)\}}{n}$$

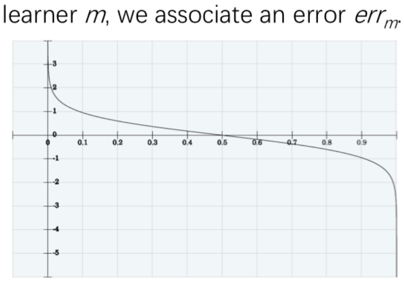
The error rate on each weak learner:

$$err_m := \frac{\sum_{i=1}^{n} w_i \; 1\{y_i \neq G_m(x_i)\}}{\sum_{i=1}^{n} w_i}$$

Intuition:

- Give large weights for hard examples.
- Give small weights for easy examples.

For each weak learner m, we associate an error err_m



$$\alpha_m = \frac{1}{2} log(\frac{1-err_m}{err_m})$$

AdaBoost

1. Initialize the example weights, $w_i = 1/n$, $i = 1, \ldots, n$.
2. For $m = 1$ to M (number of weak learners)

   (a) Fit a classier $G_m(x)$ to training data using the weights w_i

   (b) Compute

   $$err_m := \frac{\sum_{i=1}^{n} w_i \; 1\{y_i \neq G_m(x_i)\}}{\sum_{i=1}^{n} w_i}$$

   (c) Compute

   $$\alpha_m = \frac{1}{2} \, log(\frac{1-err_m}{err_m})$$

   (d) Compute

   $$w_i \leftarrow w_i . exp[-\alpha_m y_i G_m(x_i)] \quad for \; i = 1, \cdots, n.$$

3. Output

   $$G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$$

Digression: Decision Stumps

This is an example of very weak classier

A simple 2-terminal node decision tree for binary classification.

$$f(\mathbf{x}) = s(x_k > c)$$

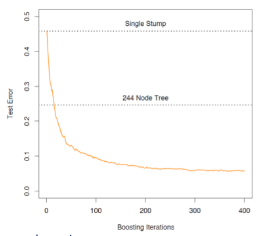
Where $c \in \mathbb{R}$, $k \in \{1, \ldots, d\}$, $s \in \{-1, 1\}$.

A decision stump is often trained by brute force: discretize the real numbers from the smallest to the largest value in the training set, enumerate all possible classifiers, and pick the one with the lowest training error.

Example: A dataset with 10 features, 2,000 examples training and 10,000 testing.

AdaBoost Performance

AdaBoost Performance



Error rates:

- Random: 50%.
- Stump: 45.8%.
- Large classification tree: 24.7%.
- AdaBoost with stumps: 5.8% after 400 iterations!

AdaBoost with Decision stumps lead to a form of: **feature selection**

Bagging & Bootstrapping

- Bootstrap is a re-sampling technique ≡ sampling from the empirical distribution.
- Aims to improve the quality of estimators.
- Bagging and Boosting are based on bootstrapping.
- Both use re-sampling to generate weak learners for classification.
- **Strategy:** Randomly distort data by re-sampling.
- Train weak learners on re-sampled training sets.
- **B**ootstrap **agg**regation ≡ Bagging.

Bagging

Training

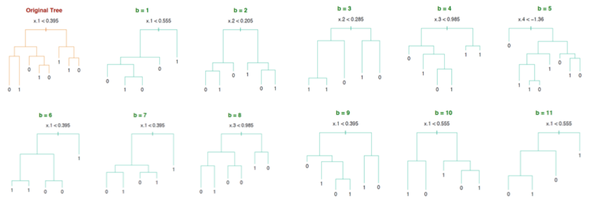
For $b = 1, \ldots, B$

1. Draw a bootstrap sample B_b of size L from training data.
2. Train a classifier f_b on B_b.

Classification: Classify by majority vote among the B classifiers:

$$favg := \frac{1}{B} \sum_{b=1}^{B} f_b(x)$$

Bagging works well for trees:



Random Forests

1. Random forests: modifies bagging with trees to reduce correlation between trees.
2. Tree training optimizes each split over all dimensions.
3. But for Random forests, **choose a different subset of dimensions at each split**. Number of dimensions chosen m.
4. Optimal split is chosen within the subset.
5. The subset is chosen at random out of all dimensions $1, \ldots, d$.
6. Recommended $m = \sqrt{d}$ or smaller.