Ensemble Methods: Classification by Committee
Ensemble methods perform classification by "pooling" or "aggregating" the results of several classifiers.

**Setting 1: Hard decisions**

Suppose $f_1, \ldots, f_m$ are classifiers

$$f_k : \mathbb{R}^d \rightarrow \{ -1, 1 \}$$

that output "hard" decisions, -1 or 1.

The classifiers may be combined by taking a (weighted) majority vote

$$f_{\text{combined}} (x) = \text{sign} \left\{ \sum_{k=1}^{M} w_k f_k (x) \right\}$$
Setting 2: Soft decisions

Write

\[ f_k(x) = \text{sign}(g_k(x)) \]

where \( g_k : \mathbb{R}^d \rightarrow \mathbb{R} \) provides a "soft" decision.

Examples:

- **Linear classifier**: \( g(x) = \mathbf{w}^T x + b \)
- **Kernel classifier**: \( g(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) \)

These soft decisions may be combined as

\[ f_{\text{combined}}(x) = \text{sign}\left\{ \sum_{k=1}^{M} w_k g_k(x) \right\} \]
Why do this?

1. $f_1, \ldots, f_n$ are too simple to be good classifiers by themselves. But if they were somehow organized to work in unison, the combined classifiers could perform very well.

2. Variance reduction:

   Suppose a classification algorithm has a high variance, meaning it is highly sensitive to slight perturbations of the training data. If each $f_k$ is produced by the same algorithm, but on different variations (or reweightings) of the training data, the combined classifier may have smaller variance.
Example 1: Histograms

Consider a $d=2$ dimensional setting. A regular histogram classifier with bin width 1 looks like

The partition is fixed, irrespective of the data. As the data gradually shift upward, the classifier changes abruptly.

In contrast, a linear classifier would transition smoothly as the data shifts.
Example 1 Decision trees.

A decision tree is based on a tree-structured hierarchy of classifiers.

Often the "nodes" of the tree are simple univariate splits, e.g.
In 2-d:

\[
\begin{array}{c|cc|}
 & -1 & +1 \\
-1 &   & +1 \\
-1 & -1 & +1 \\
\end{array}
\]

Like histograms, most algorithms for constructing decision trees have a high variance.

**Averaged Shifted Histograms**

An ensemble approach can be used to reduce the variance of the histogram classifier based on a fixed partition.

**Idea:**

- Generate $M$ small shifts of the data
- For each shift, form the histogram classifier
- Average the resulting classifiers
In detail

- For $k = 1, \ldots, M$, generate
  
  \[ \mathbf{e}_k = (e_{k1}, e_{k2}, \ldots, e_{kd})^T \]

  For example, if the histogram binwidth is $h$, consider shifts of the form
  
  \[ (0, \ldots, 0, \pm \frac{h}{2}, 0, \ldots, 0)^T \]

- Let
  
  \[ X_0 = \left\{ x_1, \ldots, x_n \right\}, \quad x_i \in \mathbb{R}^d \]

  denote the training data.

Define

\[ X_k = X_0 + \mathbf{e}_k \]

\[ = \left\{ x_i + e_{ki}, \ldots, x_n + e_{kd} \right\} \]

- Let $f_{X_k}(x)$ be the histogram classifier based on the data $X_k$
Majority vote:

\[ f_{\text{combined}}(x) = \frac{1}{M} \sum_{k=1}^{M} f_{X_k}(x) \]

The result is a classifier with much lower variance.

- Single histogram
- Average of many histograms

true decision boundary
$n = 100$ points, $\text{bin width} = \frac{1}{3}$
# of votes = 1

5

11

21

5 realizations of data

n = 1000 points, bin width = $\frac{1}{3}$
Bagging is short for bootstrap aggregation

**Definition** Let $X_0 = \{x_1, \ldots, x_n\}$ be a training sample. Let $X^* = \{x_1^*, \ldots, x_n^*\}$ be obtained by sampling with replacement from $X_0$. Then $X^*$ is called a bootstrap sample.

**Idea:**

- Generate $B$ bootstrap samples $X_1^*, \ldots, X_B^*$.
- Let $f_{x_b}^* (x)$ be the classifier trained on $X_b^*$.
- Vote:

$$f_{\text{combined}} (x) = \frac{1}{B} \sum_{b=1}^{B} f_{x_b}^* (x)$$
Both the averaged shifted histograms and bagging combine resampling with a majority vote. Many other resampling schemes are conceivable.

E.g., random convex combinations:

- Given \( X_0 = \{ x_1, \ldots, x_n \} \), generate \( X^* = \{ x^*_1, \ldots, x^*_n \} \), where \( x^*_i \) is obtained as

\[
x^*_i = \lambda x_a + (1-\lambda) x_b
\]

where \( a, b \) are random, \( \lambda \in [-\delta, 1+\delta] \) is uniform.
Recall there were two reasons that motivated ensemble rules: 1) combining simple rules into a complex rule; 2) variance reduction. Thus far we have only discussed the latter point.

Boosting is an ensemble rule that achieves both. It is based on the notion of a base learner.

Definition | A base learner is any classification rule such that, given any training sample \((x_1, y_1), \ldots, (x_n, y_n)\), and weights \(w_1, \ldots, w_n \ (w_i > 0, \sum w_i = 1)\), it produces a classifier \(f\) such that

\[
\sum_{i=1}^{n} w_i \mathbb{I}_{\{f(x_i) \neq y_i\}}
\]

is small
In short, a base learner can learn a classifier that respects any possible weighting of the training error.

The Boosting principle

Choose an initial weighting \( W^{(1)} \)

- Given a weighting \( W^{(t)} \), apply the base learner to generate a classifier \( f_t \)

- Upweight \( w_{i}^{(t)} \) if \( f(x_i) \neq y_i \)

- Downweight \( w_{i}^{(t)} \) if \( f(x_i) = y_i \)

Repeat while \( t \leq T \).

Output

\[
f(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t f_t(x) \right)
\]

Where \( \alpha_t > 0 \) reflects the confidence in \( f_t \).
Examples of base learners

- Decision trees. As an averaging procedure, boosting will reduce their variance.

- Decision stumps: trees consisting of a single split

  \[ f(x) = \text{sign} \left\{ x_j \geq c \right\} \]

- Radial basis functions

  \[ f(x) = \text{sign} \left\{ k(x, x_i) + b \right\} \]

Exercise Describe an algorithm (a base learner) that chooses the decision stump with minimal weighted training error. Ditto for RBFs.
Solution

A similar strategy applies to RBFs. If $\sigma$ is fixed (choosing $\sigma$ is a separate problem), the total number of classifiers to consider is ___.

Adaboost

The first successful boosting algorithm was introduced by Yoav Freund and Robert Schapire, called Adaboost.
Given \((x_i, y_i), \ldots, (x_n, y_n)\), \(y_i \in \{-1, +1\}\)

Initialize \(w_i^1 = \frac{1}{n}\).

For \(t = 1, \ldots, T\)

- Apply base learner with weights \(w_t^t\) to produce classifier \(f_t\)

- Set \(r_t = \sum_{i=1}^{n} w_i^t I\{f_t(x_i) \neq y_i\}\)

- Set \(\alpha_t = \frac{1}{2} \ln \left( \frac{1 - r_t}{r_t} \right)\)

- Update \(w_i^{t+1} = w_i^t \cdot \exp \left\{ -\alpha_t y_i f_t(x_i) \right\} / Z_t\)

where \(Z_t\) is a normalization constant

End

Output \(f(x) = \text{sign} \left\{ \sum_{t=1}^{T} \alpha_t f_t(x) \right\}\)
The success of AdaBoost is reflected in the following result.

Theorem. Suppose $R_t = \frac{1}{2} - \delta_t$, $\delta_t \geq 0$, for each $t$. Then

$$\frac{1}{n} \sum_{i=1}^{n} I\{f(x_i) \neq y_i\} \leq \exp\left(-2 \sum_{t=1}^{T} \delta^2_t\right)$$

In particular, if $\delta_t \geq \delta > 0$ for all $t$, then

$$\frac{1}{n} \sum_{i=1}^{n} I\{f(x_i) \neq y_i\} \leq \exp\left(-2\delta^2 \cdot T\right)$$

The assumption $\delta_t \geq \delta > 0 \ \forall t$ is sometimes called the weak learning hypothesis, and the base learner is called a weak learner.

In words, the theorem tells us if our base learner does slightly better than random guessing, the final combined classifier can separate the training data perfectly for $T$ large enough. In fact, the error goes to zero exponentially fast!
Adaboost details and comments

1. If $r_t = 0$, then $\alpha_t = 0$ and the algorithm breaks down. On the other hand, if $r_t = 0$, then $f_t$ classifies every point perfectly and there is no need to boost.

2. $T$ must be set in some manner. Unfortunately, no satisfactory theory or method for setting $T$ is known. In practice a couple of options are

   - Set $T$ by cross-validation
   - Let $T_0$ be the number of iterations until the training error is zero.
   
     Set $T = (1.1) \times T_0$.

3. Empirical evidence suggests that Adaboost using decision trees for base learners is one of the best "off-the-shelf" methods for classification.
Proof of Theorem. The proof is broken down into some lemmas.

Lemma
\[ \frac{1}{n} \sum_{i=1}^{n} I\{f(x_i) \neq y_i \} \leq \prod_{t=1}^{T} Z_t \]

Proof. By unraveling the update rule we find

\[ W_{iT+1} = \frac{W_i^T \exp \left( -\alpha_T y_i f_T(x_i) \right)}{Z_T} \]
\[ = \frac{w_i^{T-1} \exp \left( -y_i [\alpha_{T-1} f_{T-1}(x_i) + \alpha_T f_T(x_i)] \right)}{Z_{T-1} \cdot Z_T} \]
\[ \vdots \]
\[ = \frac{1}{n} \cdot \exp \left( -y_i \sum_{t=1}^{T} \alpha_t f_t(x_i) \right) \]
\[ \frac{Z_1 \cdot Z_2 \cdots Z_T}{Z_1 \cdot Z_2 \cdots Z_T} \]
\[ = \frac{\exp \left( -y_i F_T(x_i) \right)}{n \prod_{t=1}^{T} Z_t} \]

where \[ F_t = \sum_{s=1}^{t} \alpha_s f_s \]

and \[ f(x) = \text{sign} \left\{ F_T(x) \right\} \]
Now use the bound

\[ I_{\{f(x_i) \neq y_i\}} = I_{\{y_i F_T(x_i) < 0\}} \leq \exp\left(-y_i F_T(x_i)\right) \]

Then

\[ 1 = \sum_{i=1}^{n} w_i^{T+1} \]

\[ = \sum_{i=1}^{n} \frac{\exp\left(-y_i F_T(x_i)\right)}{n \cdot (\pi Z_t)} \]

\[ \geq \frac{1}{(\pi Z_t)} \cdot \frac{1}{n} \sum_{i=1}^{n} \ I_{\{f(x_i) \neq y_i\}} \]

and the lemma follows.
Lemma \[ Z_t = \sqrt{1 - \gamma_t^2} \]

Proof \[ Z_t = \sum_{i=1}^{n} w_i^t \exp (-\alpha_t y_i f_t(x_i)) \]
\[ = \sum_{i=1}^{n} w_i^t e^{-\alpha_t} + \sum_{i=1}^{n} w_i^t e^{\alpha_t} \]
\[ = (1 - r_t) e^{-\alpha_t} + r_t e^{\alpha_t} \]

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

Then \[ Z_t = (1 - r_t) \sqrt{\frac{r_t}{1 - r_t}} + r_t \sqrt{\frac{1 - r_t}{r_t}} \]
\[ = 2 \sqrt{r_t (1 - r_t)} \]

Now substitute \[ r_t = \frac{1}{2} - \gamma_t \]
\[ z_t = 2 \sqrt{\left( \frac{1}{2} - \delta_t \right) \left( \frac{1}{2} + \delta_t \right)} \]

\[ = 2 \sqrt{\frac{1}{4} - \delta_t^2} \]

\[ = \sqrt{1 - 4\delta_t^2} \]

**Lemma**

\[ \sqrt{1 - x} \leq e^{-\frac{1}{2}x} \]

**Proof**
Formally, \( \sqrt{1 - x} \) is concave, \( e^{-\frac{1}{2}x} \) is convex, so it suffices to show their slopes (derivatives) are both \( = -\frac{1}{2} \) at \( 0 \).

Putting it all together, we obtain

\[
\frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} f(x_i) + y_i^2 \leq \frac{1}{n} \sum_{i=1}^{n} \exp \left( -y_i E_t(x_i) \right)
\]

\[
= \prod_{t=1}^{T} Z_t
\]

\[
= \prod_{t=1}^{T} \sqrt{1 - 4\chi_t^2}
\]

\[
\leq e^{-2 \sum_{t=1}^{T} \chi_t^2}
\]

Exercise | View \( Z_t \) as a function of \( x_t \), and find the value of \( x_t \) that minimizes \( Z_t \).
**Solution**  

Earlier we showed

\[ Z_t = (1 - r_t) e^{-\alpha_t} + r_t e^{\alpha_t}. \]

This is a convex, differentiable function of \( \alpha_t \).

It is minimized by setting

\[ 0 = \frac{\partial Z_t}{\partial \alpha_t} = -(1 - r_t) e^{-\alpha_t} + r_t e^{\alpha_t} \]

\[ \Rightarrow e^{2\alpha_t} = \frac{1 - r_t}{r_t} \]

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

In conclusion, each \( \alpha_t \) is chosen to minimize the corresponding term \( Z_t \) in the bound \( \prod_{t=1}^{T} Z_t \).

That is, the bound is minimized incrementally (not globally).
Alternative Loss Functions

AdaBoost uses the loss function

\[ \phi(u) = e^{-u} \]

as a convex, differentiable upper bound on \[ \mathcal{J}(u < 0) \]. However, other loss functions are possible.

For example, the "logistic loss" \[ \phi(u) = \log(1 + e^{-u}) \] doesn't work as hard on misclassified points, and therefore may be less susceptible to overfitting.
To generalize AdaBoost to other loss functions, recall

\[ F_t(x) = \sum_{s=1}^{t} \alpha_t f_s(x) \]

On the \( t \)th iteration of boosting, we have the upper bound

\[ \frac{1}{n} \sum_{i=1}^{n} I_{y_i F_t(x_i)} < 0 \overset{?}{\leq} \frac{1}{n} \sum_{i=1}^{n} \phi(y_i F_t(x_i)) \]

View this bound as an objective to be minimized over function space. That is, view the function \( F_t \) as a variable being optimized.

Boosting can be seen as functional gradient descent.

Given that \( f_1, \ldots, f_{t-1} \) have been learned, view \( f_t \) as the direction of the next step in a gradient descent minimization of the upper bound.

\[ \alpha_1 f_1, \alpha_2 f_2, \alpha_3 f_3 \]

\[ F_1 \rightarrow F_2 \rightarrow F_3 \]
We seek $f_t$ that minimizes the slope of $B_t$ at $F_{t-1}$.

Writing

$$B_t(\alpha_t) = \frac{1}{n} \sum_{i=1}^{n} \phi \left( y_i F_{t-1}(x_i) + y_i \alpha_t f_t(x_i) \right)$$

the slope of $B_t$ in the direction $f_t$ is

$$\frac{\partial B_t}{\partial \alpha_t} \bigg|_{\alpha_t=0} = \frac{1}{n} \sum_{i=1}^{n} y_i f_t(x_i) \phi'(y_i F_{t-1}(x_i))$$

Minimizing this is equivalent to minimizing

$$-\sum_{i=1}^{n} y_i f_t(x_i) \frac{\phi'(y_i F_{t-1}(x_i))}{\sum_{j=1}^{n} \phi'(y_j F_{t-1}(x_j))} =: W_{i}^{t}$$

$$= \sum_{i: y_i \neq f_t(x_i)} W_{i}^{t} - \sum_{i: y_i = f_t(x_i)} W_{i}^{t}$$

$$= 2 \left( \sum_{i: y_i \neq f_t(x_i)} W_{i}^{t} \right) - 1$$

$$\Rightarrow$$ We can use the base learner to find $f_t$
Once the direction $f_t$ is established, the next step is to determine the optimal step-size $\alpha_t$.

This is achieved by minimizing $B_t(\alpha_t)$ with respect to $\alpha_t$.

The advantage of the exponential loss is computational:

- the weight update has a nice recursive formula since

$$\phi'(a + b) = \phi'(a) \cdot \phi'(b)$$

- $B_t'(\alpha_t) = 0$ has a closed form solution.

However, using other convex losses is not much worse from a computational perspective:

- $\phi'(y_i F_{t-1}(x_i))$ is easy to compute
- $\alpha_t$ is the solution of a univariate, convex optimization problem.
Generalized Boosting Algorithm

Given \((x_1, y_1), \ldots, (x_n, y_n)\), \(y_i \in \{-1, 1\}\), convex loss \(\phi\)

Initialize \(W_i^0 = \frac{1}{n}\)

For \(t = 1, \ldots, T\)

- Apply base learner with weights \(w^t\)
  to produce classifiers \(f_t\)

- Set
  \[
  \alpha_t = \arg \min_{\alpha} \frac{1}{n} \sum_{i=1}^{n} \phi \left( y_i f_{t-1}(x_i) + y_i \alpha f_t(x_i) \right)
  \]

- Update
  \[
  W_i^{t+1} = \frac{\phi'(y_i f_t(x_i))}{\sum_{j=1}^{n} \phi'(y_j f_t(x_j))}
  \]

End

Output

\[
  f(x) = \text{sign} \left\{ \sum_{t=1}^{T} \alpha_t f_t(x) \right\} = \text{sign} \left\{ \sum_{t=1}^{T} \alpha_t f_t(x) \right\}
\]
Exercise: Verify that when \( \phi(u) = e^{-u} \), the algorithm reduces to AdaBoost.
Solution. If $\phi(u) = e^{-u}$, then $\phi''(u) = -e^{-u}$, and

$$w_i^t \propto -\phi'(y_i f_t(x_i)) = e^{-y_i f_t(x_i)}$$

$$= \frac{1}{T} e^{-y_i \sum_{s=1}^{T} f(x_i)} \quad \checkmark$$

To see that $\alpha_t$ is the same as for AdaBoost, apply the same argument used to show $\alpha_t$ minimized $E_t$.

Remark. When $\phi(u) = \log_2 (1 + e^{-2u})$, and $\alpha_t$ is estimated by a single step of a Newton-Raphson algorithm, the algorithm is called LogitBoost. This is the other common boosting algorithm besides AdaBoost.