Ensemble Methods

An ensemble method builds a model by ensembling a pool of (weaker) models. The pool is a committee: each member is a base model. If the committee has identical models (but probably with different parameters), it is a homogeneous committee; if some models are different (e.g., SVM + GDA), it is a heterogeneous committee. We will focus on homogeneous committee. Ensembled models can provide nonlinear decision boundary, even if each base model is linear. There are two common approaches to ensemble models: bagging and boosting.

Bagging

Let \( f_1, \ldots, f_m \) be \( m \) base models. Bagging ensembles them by averaging their predictions, i.e.,

\[
f(x) := \frac{1}{m} \sum_{k=1}^{m} f_k(x).
\]

Bagging (separately) learns each \( f_k \) from a bootstrap sample, which is obtained by randomly sampling a subset of training sample (with replacement).

[Discussion] Exemplify three bootstrap samples of size 3 from set \{a, b, c, d, e\}.

[Discussion] Why does bagging learn \( f_k \) from bootstrap sample instead of the entire sample?

Random Forest is bagging of decision trees with an additional constraint that only a subset of features are used for every node split. This reduces the correlation between base trees.

[Discussion] How does bagging improve learning?

Bagging can reduce prediction error under certain conditions. Let \( f_s \) be the true model. Each base model can be expressed as

\[
f_k(x) = f_s(x) + \epsilon_k(x),
\]

where \( \epsilon_k(x) \) is a noise (induced from bootstrapping). The expected prediction error of \( f_k \) is

\[
er(f_k) = E[(f_k(x) - f_s(x))^2] = E[\epsilon_k(x)]^2,
\]

where \( E \) is taken over the randomness of \( x \). The averaged error of all base models is

\[
er_m = \frac{1}{m} \sum_{i=1}^{m} er(f_k) = \frac{1}{m} \sum_{k=1}^{m} E[\epsilon_k(x)]^2.
\]
On the other hand, the expected prediction error of ensembled model $f$ is

$$er(f) = E[f(x) - f_*(x)]^2 = E \left[ \frac{1}{m} \sum_{k=1}^{m} f_k(x) - f_*(x) \right]^2$$

$$= E \left[ \frac{1}{m} \sum_{k=1}^{m} (f_k(x) - f_*(x)) \right]^2$$

$$= \frac{1}{m^2} E \left[ \sum_{k=1}^{m} \epsilon_k(x) \right]^2. \quad (5)$$

If we assume errors are uncorrelated i.e. $E[\epsilon_k(x)\epsilon_{k'}(x)] = 0$ whenever $k \neq k'$, then

$$E \left[ \sum_{k=1}^{m} \epsilon_k(x) \right]^2 = E \left[ \sum_{k=1}^{m} \epsilon_k^2(x) + 2 \sum_{k=1}^{m} \sum_{k' \neq k} \epsilon_k(x)\epsilon_{k'}(x) \right]$$

$$= E \left[ \sum_{k=1}^{m} \epsilon_k^2(x) \right] + E \left[ 2 \sum_{k=1}^{m} \sum_{k' \neq k} \epsilon_k(x)\epsilon_{k'}(x) \right]$$

$$= \sum_{k=1}^{m} E \left[ \epsilon_k^2(x) \right] + 2 \sum_{k=1}^{m} \sum_{k' \neq k} E \left[ \epsilon_k(x)\epsilon_{k'}(x) \right]$$

$$= \sum_{k=1}^{m} E \left[ \epsilon_k^2(x) \right]. \quad (6)$$

Combining (4), (5) and (6), we have

$$er(f) = \frac{1}{m^2} \cdot E \left[ \sum_{k=1}^{m} \epsilon_k(x) \right]^2 = \frac{1}{m} \cdot \frac{1}{m} \cdot \sum_{k=1}^{m} E[\epsilon_k(x)]^2 = \frac{1}{m} er_m(f). \quad (7)$$

Assuming $er_m(f)$ is bounded, we see $er(f)$ decreases as $m$ increases. In other words, prediction error of the ensembled model reduces as more base models are added to the committee. But note a limitation of this justification is the strong assumption that model errors are uncorrelated.

**Boosting**

Boosting learns base models sequentially and average them with weights. A popular algorithm is AdaBoost. It builds a model of the form

$$f(x) := \sum_{k=1}^{m} \alpha_k f_k(x), \quad (8)$$

where $\alpha_k$ is larger if $f_k(x)$ is more accurate, and $f_{k+1}(x)$ is learned on a weighted sample where instances have higher weights if they are misclassified by the previously learned $k$ models. The specific ways of computing $\alpha_i$ and instance weight are shown in Algorithm 1 – if $f_k$ is accurate, $\epsilon_k$ is small and $\alpha_k$ is big; if $(x_i, y_i)$ is misclassified, $\ell(f(x_i), y_i)$ is big and $w_i$ becomes big.

The designs of $\alpha_k$ and $w_i$ are not arbitrary. They are derived assuming an AdaBoost model is minimizing exponential loss on training sample. We will show this derivation in the following.
Algorithm 1 AdaBoost

**Input:** training sample \( S = \{x_1, \ldots, x_n\} \), committee size \( m \)

**Initialize:** weight \( w_i = 1/n \) for instance \( x_i \)

**for** \( k = 1, \ldots, m \) **do**

1. train base model \( f_k \) on \( S \) by minimizing the following weighted loss

\[
J(f_k) = \sum_{i=1}^{n} w_i \cdot 1_{f_k(x_i) \neq y_i}.
\]

2. compute model weight

\[
\alpha_k = \ln \left\{ \frac{1 - \epsilon_k}{\epsilon_k} \right\},
\]

where

\[
\epsilon_k = \frac{\sum_{i=1}^{n} w_i \cdot 1_{f_k(x_i) \neq y_i}}{\sum_{i=1}^{n} w_i}.
\]

3. update instance weight

\[
w_i = w_i \cdot \exp \{ \alpha_k \cdot 1_{f_k(x_i) \neq y_i} \}.
\]

**end for**

**Output:** an ensembled model

\[
f(x) := \sum_{k=1}^{m} \alpha_k f_k(x).
\]

Let \( Y = \{-1, +1\} \) be the label set. Let the ensemble model be\(^1\)

\[
f_{[m]}(x) = \frac{1}{2} \sum_{k=1}^{m} \alpha_k f_k(x).
\]

Assume the previous \( m - 1 \) models \( f_1, \ldots, f_{m-1} \) are learned. Our goal is to learn \( f_m \) so that the ensemble model can minimize the following exponential loss on training sample

\[
L_n = \sum_{i=1}^{n} \exp[-y_i f_{[m]}(x_i)].
\]

To do so, first isolate \( f_m \) in the loss, i.e.,

\[
L_n = \sum_{i=1}^{n} \exp \left\{ -y_i \cdot (f_{[m-1]}(x_i) + \frac{1}{2} \alpha_m f_m(x_i)) \right\}
\]

\[
= \sum_{i=1}^{n} \exp \left\{ -y_i f_{[m-1]}(x_i) - \frac{1}{2} \alpha_m y_i f_m(x_i) \right\}
\]

\[
= \sum_{i=1}^{n} \exp \left\{ -y_i f_{[m-1]}(x_i) \right\} \cdot \exp \left\{ -\frac{1}{2} \alpha_m y_i f_m(x) \right\}
\]

\[
= \sum_{i=1}^{n} w_i \cdot \exp \left\{ -\frac{1}{2} \alpha_m y_i f_m(x) \right\},
\]

where \( w_i = \exp \left\{ -y_i f_{[m-1]}(x_i) \right\} \) depends on previously learned models and thus can be treated as a constant when optimizing \( f_m \) and \( \alpha_m \).

\(^1\)The constant \( \frac{1}{2} \) there to facilitate discussion; it can be absorbed by \( \alpha \).
Next, isolate misclassified instances in the loss. Let $I_{\text{cor}}$ and $I_{\text{mis}}$ be the index sets of correctly classified and misclassified instances, respectively. Note $y_i f_m(x_i)$ equals -1 if $x_i$, $y_i$ is misclassified and equals 1 otherwise. Then

$$L_n = \sum_{i=1}^{n} w_i \cdot \exp \left[ -\frac{1}{2} \alpha_m y_i f_m(x) \right],$$

$$= \sum_{i \in I_{\text{cor}}} w_i \cdot \exp \left[ -\frac{1}{2} \alpha_m y_i f_m(x) \right] + \sum_{i \in I_{\text{mis}}} w_i \cdot \exp \left[ -\frac{1}{2} \alpha_m y_i f_m(x) \right]$$

$$= \sum_{i \in I_{\text{cor}}} w_i \cdot \exp \left[ -\frac{1}{2} \alpha_m \right] + \sum_{i \in I_{\text{mis}}} w_i \cdot \exp \left[ \frac{1}{2} \alpha_m \right]$$

$$= \left( \exp \left[ \frac{1}{2} \alpha_m \right] - \exp \left[ -\frac{1}{2} \alpha_m \right] \right) \sum_{i=1}^{n} w_i \cdot 1_{f_m(x_i) \neq y_i} + \exp \left[ -\frac{1}{2} \alpha_m \right] \sum_{i=1}^{n} w_i.$$ (17)

[Exercise] Verify the last equation of (17).

From (17), we see minimizing $L_n$ w.r.t. $f_m$ is equivalent to minimizing $\sum_{i=1}^{n} w_i \cdot 1_{f_m(x_i) \neq y_i}$, which gives (9) in Algorithm 1. We also see minimizing $L_n$ w.r.t. $\alpha_m$ gives (10) (11) in Algorithm 1, because applying the critical point method gives

$$\alpha_m = \ln \frac{\sum_{i=1}^{n} w_i - \sum_{i=1}^{n} w_i \cdot 1_{f_m(x_i) \neq y_i}}{\sum_{i=1}^{n} w_i}.$$ (18)

[Exercise] Verify (18) is equivalent to (10) (11).

To derive (12) in Algorithm 1, consider learning the $(m+1)_{\text{th}}$ model. The loss is

$$L(f_{m+1}) = \sum_{i=1}^{n} \exp \left[ -y_i f_{m+1}(x_i) \right]$$

$$= \sum_{i=1}^{n} \exp \left[ -y_i \left( f_m(x_i) + \frac{1}{2} \alpha_m f_{m+1}(x_i) \right) \right]$$

$$= \sum_{i=1}^{n} \exp \left[ -y_i f_m(x_i) \right] \cdot \exp \left[ -\frac{1}{2} y_i \alpha_m f_{m+1}(x_i) \right]$$

$$= \sum_{i=1}^{n} \exp \left[ -y_i f_{m-1}(x_i) \right] \cdot \exp \left[ -\frac{1}{2} y_i \alpha_m f_m(x_i) \right] \cdot \exp \left[ -\frac{1}{2} y_i \alpha_m f_{m+1}(x_i) \right]$$

$$= \sum_{i=1}^{n} w_i \cdot \exp \left[ -\frac{1}{2} y_i \alpha_m f_{m+1}(x_i) \right] \cdot \exp \left[ -\frac{1}{2} y_i \alpha_m f_{m+1}(x_i) \right]$$

$$= \sum_{i=1}^{n} w_i' \cdot \exp \left[ -\frac{1}{2} y_i \alpha_m f_{m+1}(x_i) \right],$$ (19)

where the new instance weight for learning $f_{m+1}$ is updated as

$$w_i' = w_i \cdot \exp \left[ -\frac{1}{2} y_i \alpha_m f_m(x_i) \right].$$ (20)

Because of the trick that

$$y_i f_m(x_i) = 1 - 2 \cdot 1_{f_m(x_i) \neq y_i},$$ (21)
we have
\[ w'_i = w_i \cdot \exp \left[ -\frac{1}{2} \alpha_m (1 - 2 \cdot 1_{f_m(x_i) \neq y_i}) \right] = \exp \left[ -\frac{1}{2} \alpha_m \right] \cdot w_i \cdot \exp \left[ \alpha_m 1_{f_m(x_i) \neq y_i} \right], \quad (22) \]
where the first term does not depend on \( x_i \) and the last two terms give (12) in Algorithm 1.

[Exercise] Verify (21) and (22).