1 Ensemble Learning

If a single model cannot give satisfactory performance, one can ensemble a committee of models to build a more powerful combined model. This is the idea of ensemble learning.

We will introduce two ensemble learning methods: bagging and boosting.

1.1 Bagging

Suppose a committee is made of multiple identically-structured logistic regression models. If these models are trained on the same training set, they will converge to the same optimal model. This is no different from single-model learning.

To fully exploit the power of ensemble learning, we need some diversity in the committee.

One way to introduce diversity is to apply bootstrapping. This approach randomly samples a subset of training data (with replacement), and learns each committee model from one subset.

Let \( f_1, \ldots, f_m \) be \( m \) committee models learned from bootstrapped samples. Their predictions on any instance \( x \) can be combined as

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

The above technique is called bagging.

[Exercise] Bagging decision tree.

Theoretical Justification of Bagging.

Let’s see how bagging can improve learning from a theoretical perspective.

Let \( f_* \) be the true model. Each committee model can be expressed as

\[
f_k(x) = f_*(x) + \epsilon_k(x),
\]

where \( \epsilon_k(x) \) is the noise (induced from the random subset sampled to train \( f_k \)).

The expected squared prediction error of each model is

\[
E[f_k(x) - f_*(x)]^2 = E[\epsilon_k(x)]^2,
\]

where both expectations are taken over the randomness of the population.

The averaged squared prediction error of all models is

\[
\text{er}_{av} = \frac{1}{m} \sum_{k=1}^{m} E[f_k(x) - f_*(x)]^2 = \frac{1}{m} \sum_{k=1}^{m} E[\epsilon_k(x)]^2.
\]

\(^1\)This means different models have the same set of hyper-parameters and learning algorithms.

\(^2\)Bagging of decision tree is called random forest.
Similarly, the expected squared prediction error of the committee model (1) is

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

\[ = E \left[ \frac{1}{m} \sum_{k=1}^{m} (f_k(x) - f_*(x)) \right]^2 \] (5)

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

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

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

\[ = E \left[ \sum_{k=1}^{m} \epsilon_k^2(x) \right] + E \left[ 2 \sum_{k=1}^{m} \sum_{k' \neq k} \epsilon_k(x)\epsilon_{k'}(x) \right] \] (6)

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

Plugging (6) back to (5), we have

\[ e_{\text{com}} = \frac{1}{m^2} E \left[ \sum_{k=1}^{m} \epsilon_k(x) \right]^2 = \frac{1}{m^2} E \left[ \sum_{k=1}^{m} \epsilon_k^2(x) \right] = \frac{1}{m^2} \sum_{k=1}^{m} E \left[ \epsilon_k^2(x) \right]. \] (7)

Combining (4) and (7), we have

\[ e_{\text{com}} = \frac{1}{m} \cdot e_{\text{ave}}. \] (8)

Assume \( e_{\text{ave}} \) is bounded. We see as more committee models are aggregated (i.e. \( m \) increases), our committee model \( f_{\text{com}} \) becomes more accurate (i.e. \( e_{\text{com}} \) decreases).

Above is a theoretical justification of bagging. But note it assumes model errors are uncorrelated, which is rarely true in reality. So the improvement of bagging may be limited in practice.

[Reading] PRML, Chapter 14.2.

### 1.2 Boosting

Boosting constructs committee models sequentially and with weights.

A popular boosting algorithm is AdaBoost. It assigns weights to both instances and committee models. When use instances to learn a model, it assigns higher weights to instances mis-classified by previous models; it also assigns higher weights to models with higher prediction accuracy. Its procedure is described in Algorithm 1.
Algorithm 1 AdaBoost Algorithm.

**Input:** training set $S = \{x_1, \ldots, x_n\}$; weight $w_i = 1/n$ for $x_i$; $m$ number of committee models

**for** $k = 1, \ldots, m$ **do**

1: train committee model $f_k$ on $S$ by minimizing the weighted loss

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

(9)

2: evaluate quantities

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

and then

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

(11)

3: update weights by

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

(12)

**end for**

**Output:** an aggregated prediction model

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

(13)

Algorithm 1 can be interpreted as follows:

- (9) learns a committee model with weighted instances
- (11) assigns weight to a model based on its prediction loss\(^3\)
- (12) assigns weight to an instance based on loss of its prediction\(^4\)
- (13) combines all committee models based on their weights

**Theoretical Justification of AdaBoost.**

We will show AdaBoost finds a minimizer of an aggregated exponential loss.

Assume binary classification problem and the label set is $\{-1, +1\}$.

Let’s denote an aggregated model consisting of $m$ committee members as

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

(14)

Assume $f_1, \ldots, f_{m-1}$ and their corresponding coefficients $\alpha$’s are fixed. Our goal is to find the last member $f_m$ and its coefficient $\alpha_m$ that minimize the following exponential loss

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

(15)

\(^3\)Higher loss gives lower weight.
\(^4\)Higher loss gives larger weight.
Let’s do the optimization.

First, we should isolate $f_m$ and $\alpha_m$ in $L_n$. We have

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

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

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

where $w_i = \exp \left[ -y_i f_{[m-1]}(x_i) \right]$ is a factor contributed from previous model $f_{[m-1]}$ and thus can be treated as a constant when optimizing $f_m$ and $\alpha_m$.

Next, let’s separate training instances that are correctly classified by $f_m$ from those that are misclassified by $f_m$. Let $I_{cor}$ and $I_{mis}$ be the index sets of the two types of instances, respectively.

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

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

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

[Exercise] Verify (17).

To minimize $L_n$ w.r.t. $f_m$, we can minimize

$$\sum_{i=1}^{n} w_i \cdot 1_{f_m(x_i) \neq y_i}.$$  \hspace{1cm} (18)

Note this is exactly step (9) in AdaBoost.

To minimize $L_n$ w.r.t. $\alpha_m$, we apply critical point method. The result is

$$\alpha_m = \ln \frac{\sum_{i=1}^{n} w_i - \sum_{i=1}^{n} w_i \cdot 1_{f_m(x_i) \neq y_i}}{\sum_{i=1}^{n} w_i \cdot 1_{f_m(x_i) \neq y_i}}.$$  \hspace{1cm} (19)

Note this is step (11) in AdaBoost.

[Exercise] Verify (19) is equivalent to step (11) in AdaBoost.
Finally, let’s derive the update rule for weight $w_i$. From (14), we see

$$f_{(m+1)}(x) = \frac{1}{2} \sum_{k=1}^{m+1} \alpha_k f_k(x)$$

$$= \frac{1}{2} \sum_{k=1}^{m} \alpha_k f_k(x) + \frac{1}{2} \alpha_{m+1} f_{m+1}(x)$$

$$= f_{[m]}(x) + \frac{1}{2} \alpha_{m+1} f_{m+1}(x).$$

(20)

Based on this, we have

$$\exp[-y_i f_{[m+1]}(x_i)] = \exp \left[ -y_i \left( f_{[m]}(x_i) + \frac{1}{2} \alpha_{m+1} f_{m+1}(x_i) \right) \right]$$

$$= \exp \left[ -y_i f_{[m]}(x_i) \right] \cdot \exp \left[ -\frac{1}{2} y_i \alpha_{m+1} f_{m+1}(x_i) \right].$$

(21)

Applying above to (15), the loss of model $f_{[m+1]}$ is

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

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

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

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

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

where the new weight is

$$w_i' = w_i \cdot \exp \left[ -\frac{1}{2} y_i \alpha_{m} f_{m}(x_i) \right].$$

(23)

This gives an update rule for weight of each model.

Now, we will show the above weight update rule is equivalent to step (12) in AdaBoost. To see this, we need the following trick

$$y_i f_{m}(x_i) = 1 - 2 \cdot 1_{f_{m}(x_i) \neq y_i}.$$  

(24)

[Exercise] Verify (24).\(^5\)

Plugging (24) back to (23), we have

$$w_i' = w_i \cdot \exp \left[ -\frac{1}{2} \alpha_{m} (1 - 2 \cdot 1_{f_{m}(x_i) \neq y_i}) \right] = w_i \cdot \exp \left[ -\frac{1}{2} \alpha_{m} \right] \cdot \exp \left[ \alpha_{m} 1_{f_{m}(x_i) \neq y_i} \right].$$

(25)

Note the term $\exp \left[ -\frac{1}{2} \alpha_{m} \right]$ does not depend on any $x_i$. So we can ignore it when updating $w_i'$ for each instance. Then (25) recovers step (12) in AdaBoost.

[Reading] PRML, Chapter 14.3.

\(^5\)Remember $y \in \{-1,+1\}$.  

5