Ensemble Methods

COMP9417, 22T2

1. Ensemble Methods

2. Quick Recap: Bias and Variance of Estimators

3. Bias-Variance Tradeoff

4. Bagging

5. Random Forests

6. Boosting

Ensemble Methods

Ensemble Methods

Arguably the most powerful non *deep-learning* methods, coming close to the performance of neural networks and still winning Kaggle competitions.

Ensemble Methods

Arguably the most powerful non *deep-learning* methods, coming close to the performance of neural networks and still winning Kaggle competitions.

Why?

Quick Recap: Bias and Variance of Estimators

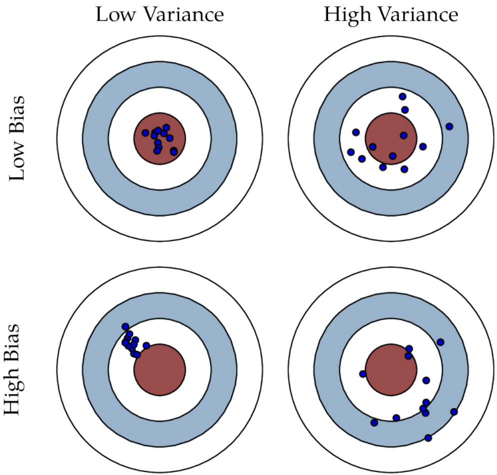
Quick Recap: Bias and Variance of Estimators

Recall the bias of an estimator $\hat{\theta}$ is defined as:

$$\text{bias}(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$$

And its variance is defined as:

$$\text{var}(\hat{\theta}) = \mathbb{E}\left[(\theta - \mathbb{E}[\hat{\theta}])^2\right]$$


Low Variance
High Variance
Low Bias
High Bias

Bias-Variance Tradeoff

Bias-Variance Tradeoff

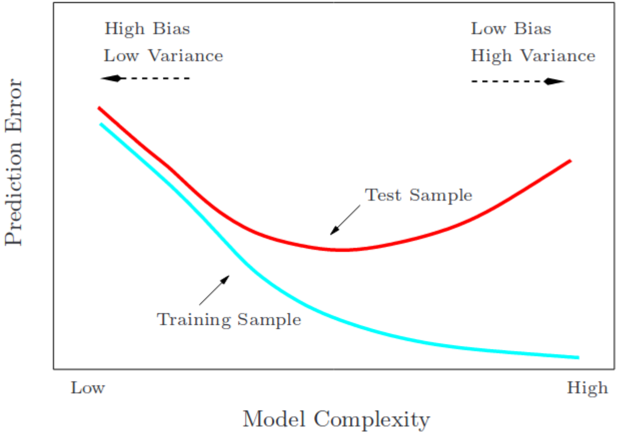
Recall the bias-variance decomposition of the MSE for an estimator $\hat{\theta}$:

$$\mathsf{MSE}(\hat{\theta}) = \mathsf{var}(\hat{\theta}) + \mathsf{bias}(\hat{\theta})^2$$

obviously for the best estimator we need to minimise the variance and minimise the bias.

However, if we try and minimise the bias, we typically also increase variance.

However, if we try and minimise the bias, we typically also increase variance.



Bagging

Bagging

Bagging or **B**oostrap **Agg**regation is an ensemble method we can apply to reduce the variance of our model.

Bagging

Bagging or **B**oostrap **Agg**regation is an ensemble method we can apply to reduce the variance of our model.

We typically take models which are easy to train and suffer from high variance (i.e decision trees), fit their basic forms on different parts of our dataset and aggregate them into a *committee*.

Bagging

Bagging or **B**oostrap **Agg**regation is an ensemble method we can apply to reduce the variance of our model.

We typically take models which are easy to train and suffer from high variance (i.e decision trees), fit their basic forms on different parts of our dataset and aggregate them into a *committee*.

For example, if we have a dataset $D = (x_i, y_i)$ for $i \in [1, n]$, we might train 4 decision trees on m points (where $m = n/4$) randomly picked from our dataset.

Bagging

Bagging or **B**oostrap **Agg**regation is an ensemble method we can apply to reduce the variance of our model.

We typically take models which are easy to train and suffer from high variance (i.e decision trees), fit their basic forms on different parts of our dataset and aggregate them into a *committee*.

For example, if we have a dataset $D = (x_i, y_i)$ for $i \in [1, n]$, we might train 4 decision trees on m points (where $m = n/4$) randomly picked from our dataset. We then have a committee of four trees with distinct knowledge on the dataset, which we can then average for our final prediction.

Generally, if we take B separate training sets from data D, our bootstrapped models will be:

$$\hat{f}^1(D_1), \hat{f}^2(D_2), \ldots, \hat{f}^B(D_B)$$

Generally, if we take B separate training sets from data D, our bootstrapped models will be:

$$\hat{f}^1(D_1), \hat{f}^2(D_2), \dots, \hat{f}^B(D_B)$$

and the final prediction for a point x is

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x)$$

Why does this work in reducing variance?

Why does this work in reducing variance?

If we consider a statistical learning problem, where we have iid. data $X_1, \ldots, X_n \sim N(\mu, \sigma^2)$ and we try finding an estimator $\hat{\mu}$ for the mean μ.

Why does this work in reducing variance?

If we consider a statistical learning problem, where we have iid. data $X_1, \ldots, X_n \sim N(\mu, \sigma^2)$ and we try finding an estimator $\hat{\mu}$ for the mean μ.

Consider an averaging estimator, where

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

Why does this work in reducing variance?

If we consider a statistical learning problem, where we have iid. data $X_1, \dots, X_n \sim N(\mu, \sigma^2)$ and we try finding an estimator $\hat{\mu}$ for the mean μ.

Consider an averaging estimator, where

$$\mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n}\mathbb{E}\left(\sum_{i=1}^{n} X_i\right)$$

Why does this work in reducing variance?

If we consider a statistical learning problem, where we have iid. data $X_1, \ldots, X_n \sim N(\mu, \sigma^2)$ and we try finding an estimator $\hat{\mu}$ for the mean μ.

Consider an averaging estimator, where

$$\mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \mu$$

Why does this work in reducing variance?

If we consider a statistical learning problem, where we have iid. data $X_1, \ldots, X_n \sim N(\mu, \sigma^2)$ and we try finding an estimator $\hat{\mu}$ for the mean μ.

Consider an averaging estimator, where

$$\mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \mu$$

$$\text{var}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\text{var}\left(\sum_{i=1}^{n} X_i\right)$$

Why does this work in reducing variance?

If we consider a statistical learning problem, where we have iid. data $X_1, \ldots, X_n \sim N(\mu, \sigma^2)$ and we try finding an estimator $\hat{\mu}$ for the mean μ.

Consider an averaging estimator, where

$$\mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \mu$$

$$\text{var}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{\sigma^2}{n}$$

Random Forests

Random Forests

In bootstrap aggregation, the trees we generate may be *correlated*. To combat this we introduce *random forests* where:

- Randomly pick bootstrap samples

Random Forests

In bootstrap aggregation, the trees we generate may be *correlated*. To combat this we introduce *random forests* where:

- Randomly pick bootstrap samples
- At every step of tree learning, randomise what features the tree splits on
  - Typically we pick $m \approx \sqrt{p}$ features for the trees to split on

Random Forests

In bootstrap aggregation, the trees we generate may be *correlated*. To combat this we introduce *random forests* where:

- Randomly pick bootstrap samples
- At every step of tree learning, randomise what features the tree splits on
    - Typically we pick $m \approx \sqrt{p}$ features for the trees to split on

Rationale: if we have strong predictors/features in our dataset, bagged trees will all typically pick the same features, leading to highly correlated predictions within the committee. This methods reduces this correlation and therefore the variance.

Boosting

Boosting

In boosting, we use a weak learner and improve it incrementally by adding more weak learners to make up for its mistakes.

Boosting

In boosting, we use a weak learner and improve it incrementally by adding more weak learners to make up for its mistakes. So we'll have a final model in the form,

$$C_m(X) = \alpha_1 h_1(X) + \alpha_2 h_2(X) + \ldots + \alpha_m h_m(X)$$

where α_i signifies the influence/weighting we give a model h_i for the final decision.

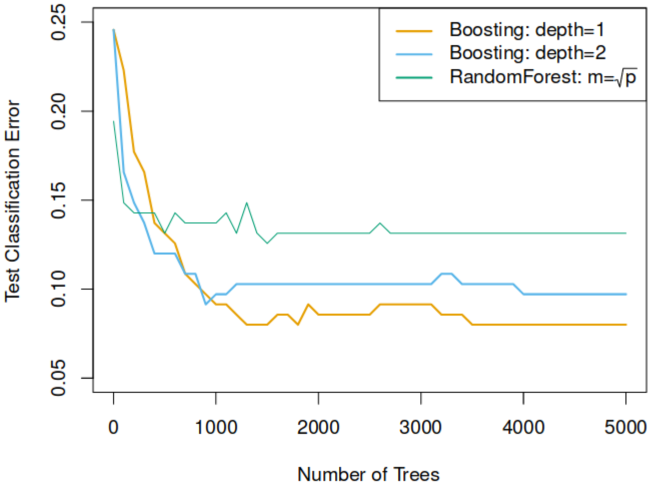
Boosting

In boosting, we use a weak learner and improve it incrementally by adding more weak learners to make up for its mistakes. So we'll have a final model in the form,

$$C_m(X) = \alpha_1 h_1(X) + \alpha_2 h_2(X) + \ldots + \alpha_m h_m(X)$$

where α_i signifies the influence/weighting we give a model h_i for the final decision.

We also define a w_i for each iteration, signifying the weighting of each point. As each subsequent model needs to be an improvement on the last, we use these weights to signify which point the previous model misclassified.


Boosting: depth=1
Boosting: depth=2
RandomForest: $m=\sqrt{p}$
Test Classification Error
0.05
0.10
0.15
0.20
0.25
0
1000
2000
3000
4000
5000
Number of Trees

Adaboost

Let's take a look at the **Ada**ptive **Boost**ing algorithm.

Adaboost

Let's take a look at the **Ada**ptive **Boost**ing algorithm.

For a binary classification problem, we'll define the exponential loss as:

$$L(h(x_i), y_i) = e^{-y_i h(x_i)}$$

this loss typically isn't used in practice, but gives us a way of *weighting* how good a model performs on a dataset.

Recall, our boosted model takes the form:
$C_m(X) = \alpha_1 h_1(X) + \alpha_2 h_2(X) + \ldots + \alpha_m h_m(X)$.

So, our total loss will be:

$$L(C_m(X), Y) = \sum_{i=1}^{n} e^{-y_i C_m(x_i)}$$

Recall, our boosted model takes the form:
$C_m(X) = \alpha_1 h_1(X) + \alpha_2 h_2(X) + \ldots + \alpha_m h_m(X)$.

So, our total loss will be:

$$\begin{aligned} L(C_m(X), Y) &= \sum_{i=1}^{n} e^{-y_i C_m(x_i)} \\ &= \sum_{i=1}^{n} e^{-y_i C_m(x_i)} \end{aligned}$$

Recall, our boosted model takes the form:
$C_m(X) = \alpha_1 h_1(X) + \alpha_2 h_2(X) + \ldots + \alpha_m h_m(X)$.

So, our total loss will be:

$$\begin{aligned} L(C_m(X), Y) &= \sum_{i=1}^{n} e^{-y_i C_m(x_i)} \\ &= \sum_{i=1}^{n} e^{-y_i (C_{m-1}(x_i) + \alpha_m h_m(x_i))} \end{aligned}$$

Recall, our boosted model takes the form:
$C_m(X) = \alpha_1 h_1(X) + \alpha_2 h_2(X) + \ldots + \alpha_m h_m(X)$.

So, our total loss will be:

$$\begin{aligned} L(C_m(X), Y) &= \sum_{i=1}^{n} e^{-y_i C_m(x_i)} \\ &= \sum_{i=1}^{n} e^{-y_i (C_{m-1}(x_i) + \alpha_m h_m(x_i))} \\ &= \sum_{i=1}^{n} e^{-y_i C_{m-1}(x_i)} e^{-y_i \alpha_m h_m(x_i)} \end{aligned}$$

Recall, our boosted model takes the form:
$C_m(X) = \alpha_1 h_1(X) + \alpha_2 h_2(X) + \ldots + \alpha_m h_m(X)$.

So, our total loss will be:

$$
\begin{aligned}
L(C_m(X), Y) &= \sum_{i=1}^{n} e^{-y_i C_m(x_i)} \\
&= \sum_{i=1}^{n} e^{-y_i (C_{m-1}(x_i) + \alpha_m h_m(x_i))} \\
&= \sum_{i=1}^{n} e^{-y_i C_{m-1}(x_i)} e^{-y_i \alpha_m h_m(x_i)} \\
&= \sum_{i=1}^{n} w_i^m e^{-y_i \alpha_m h_m(x_i)}
\end{aligned}
$$

$$L(C_m(X), Y) = \sum_{i=1}^{n} w_i^m e^{-y_i \alpha_m h_m(x_i)}$$

$$L(C_m(X), Y) = \sum_{y_i = h_m(x_i)} w_i^m e^{-\alpha_m} + \sum_{y_i \neq h_m(x_i)} w_i^m e^{\alpha_m}$$

$$L(C_m(X), Y) = \sum_{y_i = h_m(x_i)} w_i^m e^{-\alpha_m} + \sum_{y_i \neq h_m(x_i)} w_i^m e^{\alpha_m}$$

So, our problem is essentially,

$$\arg\min_{\alpha, h} \left(\sum_{y_i = h_m(x_i)} w_i^m e^{-\alpha_m} + \sum_{y_i \neq h_m(x_i)} w_i^m e^{\alpha_m} \right)$$

$$\frac{\partial L}{\partial \alpha} = -e^{-\alpha_m} \sum_{y_i = h_m(x_i)} w_i^m + e^{\alpha_m} \sum_{y_i \neq h_m(x_i)} w_i^m$$

$$\frac{\partial L}{\partial \alpha} = -e^{-\alpha_m} \sum_{y_i = h_m(x_i)} w_i^m + e^{\alpha_m} \sum_{y_i \neq h_m(x_i)} w_i^m$$

At the minimum:

$$-e^{-\alpha_m} \sum_{y_i = h_m(x_i)} w_i^m + e^{\alpha_m} \sum_{y_i \neq h_m(x_i)} w_i^m = 0$$

$$\frac{\partial L}{\partial \alpha} = -e^{-\alpha_m} \sum_{y_i = h_m(x_i)} w_i^m + e^{\alpha_m} \sum_{y_i \neq h_m(x_i)} w_i^m$$

At the minimum:

$$e^{2\alpha_m} = \frac{\sum_{y_i = h_m(x_i)} w_i^m}{\sum_{y_i \neq h_m(x_i)} w_i^m}$$

$$\frac{\partial L}{\partial \alpha} = -e^{-\alpha_m} \sum_{y_i = h_m(x_i)} w_i^m + e^{\alpha_m} \sum_{y_i \neq h_m(x_i)} w_i^m$$

At the minimum:

$$e^{2\alpha_m} = \frac{\sum_{y_i = h_m(x_i)} w_i^m}{\sum_{y_i \neq h_m(x_i)} w_i^m}$$

$$\alpha_m = \frac{1}{2} \log \left(\frac{\sum_{y_i = h_m(x_i)} w_i^m}{\sum_{y_i \neq h_m(x_i)} w_i^m} \right)$$

If we let

$$\epsilon_m = \frac{\sum_{y_i \neq h_m(x_i)} w_i^m}{\sum_{i=1}^{n} w_i^m}$$

We can redefine α_m as:

$$\alpha_m = \frac{1}{2} \log \left(\frac{1 - \epsilon_m}{\epsilon_m} \right)$$

To actually get a form for $w_i^{(m)}$, we can apply the same trick of recursion,

$$\begin{aligned} w_i^{(m)} &= e^{-y_i C_{m-1}(x_i)} \\ &= e^{-y_i (C_{m-2}(x_i) + \alpha_{m-1} h_{m-1}(x_i))} \\ &= w_i^{(m-1)} e^{-y_i \alpha_{m-1} h_{m-1}(x_i)} \end{aligned}$$

To actually get a form for $w_i^{(m)}$, we can apply the same trick of recursion,

$$\begin{aligned} w_i^{(m)} &= e^{-y_i C_{m-1}(x_i)} \\ &= e^{-y_i (C_{m-2}(x_i) + \alpha_{m-1} h_{m-1}(x_i))} \\ &= w_i^{(m-1)} e^{-y_i \alpha_{m-1} h_{m-1}(x_i)} \end{aligned}$$

So, when $y_i = h_{m-1}(x_i)$:

$$w_i^{(m)} = w_i^{(m-1)} e^{-\alpha_{m-1}}$$

To actually get a form for $w_i^{(m)}$, we can apply the same trick of recursion,

$$
\begin{aligned}
w_i^{(m)} &= e^{-y_i C_{m-1}(x_i)} \\
&= e^{-y_i (C_{m-2}(x_i) + \alpha_{m-1} h_{m-1}(x_i))} \\
&= w_i^{(m-1)} e^{-y_i \alpha_{m-1} h_{m-1}(x_i)}
\end{aligned}
$$

So, when $y_i = h_{m-1}(x_i)$:

$$w_i^{(m)} = w_i^{(m-1)} e^{-\alpha_{m-1}}$$

When $y_i \neq h_{m-1}(x_i)$:

$$w_i^{(m)} = w_i^{(m-1)} e^{\alpha_{m-1}}$$

Now we have our definitions, we can define the **Adaboost** algorithm.

If we have a dataset $D = (X, y)$ where $X \in \mathbb{R}^{n \times p}$ and $y \in \mathbb{R}^n$. Where T is our ensemble size and we have a learning algorithm A.

```
$w^{(1)} \leftarrow \frac{1}{n}$
**for** $t = 1, \ldots, T$ **do**
    $M_t \leftarrow A(X, w^{(t)})$
    $\alpha_t \leftarrow \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
    $w_i^{(t+1)} \leftarrow w_j^{(t)} \exp(\alpha_t)$    $j$ where $y_j \neq M_t(x_j)$
    $w_j^{(t+1)} \leftarrow w_j^{(t)} \exp(-\alpha_t)$    $j$ where $y_j = M_t(x_j)$
**end forreturn** $M(X) = \text{sgn}\left(\sum_{t=1}^{T} \alpha_t M_t(X)\right)$
```