732A99/TDDE01/732A68 Machine Learning

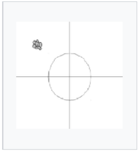
Lecture 1a Block 2: Ensemble Methods

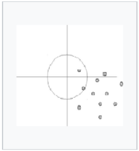
Jose M. Peña
IDA, Linköping University, Sweden

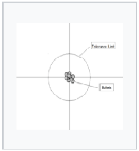
Contents and Literature

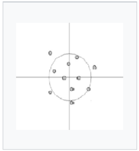
- Content
  - Bias-Variance Decomposition
  - Bagging
  - Random Forests
  - AdaBoost
  - Gradient Boosting
  - Summary
- Literature
  - Lindholm, A., Wahlström, N., Lindsten, F. and Schön, T. B. *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022. Chapter 7.

Bias-Variance Decomposition


High bias, low variance


High bias, high variance


Low bias, low variance


Low bias, high variance

Let us first make a short reminder of the general concepts of bias and variance. Consider an experiment with an unknown constant z_0, which we would like to estimate. To our help for estimating z_0 we have a random variable z. Think, for example, of z_0 as being the (true) position of an object, and z of being noisy GPS measurements of that position. Since z is a random variable, it has some mean $\mathbb{E}[z]$ which we denote by $\bar{z}$. We now define

$$\text{Bias: } \bar{z} - z_0 \tag{4.12a}$$

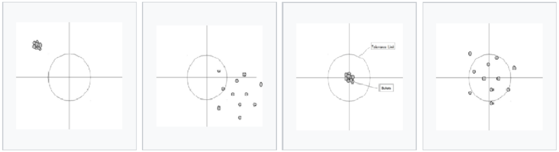
$$\text{Variance: } \mathbb{E}\left[(z-\bar{z})^2\right] = \mathbb{E}\left[z^2\right] - \bar{z}^2. \tag{4.12b}$$

The *variance* describes how much the experiment varies each time we perform it (the amount of noise in the GPS measurements), whereas the *bias* describes the systematic error in z that remains no matter how many times we repeat the experiment (a possible shift or offset in the GPS measurements). If we consider the expected squared error between z and z_0 as a metric of how good the estimator z is, we can re-write it in terms of the variance and the squared bias,

$$\begin{aligned} \mathbb{E}\left[(z-z_0)^2\right] &= \mathbb{E}\left[\left((z-\bar{z}) + (\bar{z}-z_0)\right)^2\right] = \\ &= \underbrace{\mathbb{E}\left[(z-\bar{z})^2\right]}_{\text{Variance}} + 2\underbrace{(\mathbb{E}[z]-\bar{z})}_{0}(\bar{z}-z_0) + \underbrace{(\bar{z}-z_0)^2}_{\text{bias}^2}. \end{aligned} \tag{4.13}$$

In words, the average squared error between z and z_0 is the sum of the squared bias and the variance. The main point here is that to obtain a small expected squared error, we have to consider both the bias *and* the variance. Only a small bias *or* little variance in the estimator is not enough, but both aspects are important.

Bias-Variance Decomposition



High bias, low variance   High bias, high variance   Low bias, low variance   Low bias, high variance

Let us first make a short reminder of the general concepts of bias and variance. Consider an experiment with an unknown constant z_0, which we would like to estimate. To our help for estimating z_0 we have a random variable z. Think, for example, of z_0 as being the (true) position of an object, and z of being noisy GPS measurements of that position. Since z is a random variable, it has some mean $\mathbb{E}[z]$ which we denote by $\bar{z}$. We now define

$$\text{Bias: } \bar{z} - z_0 \tag{4.12a}$$

$$\text{Variance: } \mathbb{E}\left[(z-\bar{z})^2\right] = \mathbb{E}\left[z^2\right] - \bar{z}^2. \tag{4.12b}$$

The *variance* describes how much the experiment varies each time we perform it (the amount of noise in the GPS measurements), whereas the *bias* describes the systematic error in z that remains no matter how many times we repeat the experiment (a possible shift or offset in the GPS measurements). If we consider the expected squared error between z and z_0 as a metric of how good the estimator z is, we can re-write it in terms of the variance and the squared bias,

$$\begin{aligned}\mathbb{E}\left[(z-z_0)^2\right] &= \mathbb{E}\left[\left((z-\bar{z}) + (\bar{z}-z_0)\right)^2\right] = \\ &= \underbrace{\mathbb{E}\left[(z-\bar{z})^2\right]}_{\text{Variance}} + 2\underbrace{(\mathbb{E}[z]-\bar{z})}_{0}(\bar{z}-z_0) + \underbrace{(\bar{z}-z_0)^2}_{\text{bias}^2}.\end{aligned} \tag{4.13}$$

In words, the average squared error between z and z_0 is the sum of the squared bias and the variance. The main point here is that to obtain a small expected squared error, we have to consider both the bias *and* the variance. Only a small bias *or* little variance in the estimator is not enough, but both aspects are important.

- In the regression setting, $z_0 = f(x_*)$ and $z = \hat{y}(x_*; \mathcal{T})$ for test point x_* and training data $\mathcal{T}$, and the expectation is over $\mathcal{T}$.
- The reasoning carries over to classification.

Wisdom of Crowds and Boosting Aggregation (Bagging)

- “The collective knowledge of a diverse and **independent** body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting”.

Wisdom of Crowds and Boosting Aggregation (Bagging)

- "The collective knowledge of a diverse and **independent** body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting".

Example 7.1: Using bagging for a regression problem

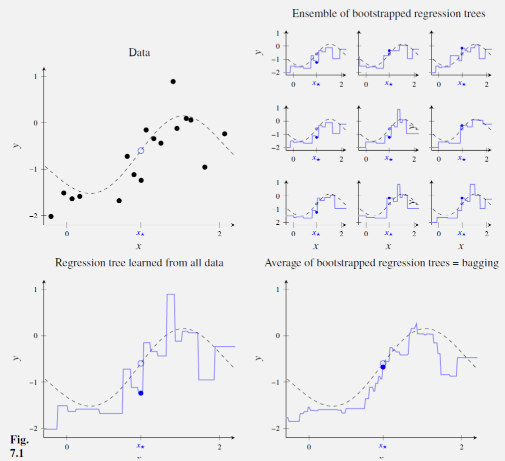


Fig. 7.1

Wisdom of Crowds and Boosting Aggregation (Bagging)

- "The collective knowledge of a diverse and **independent** body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting".

Example 7.1: Using bagging for a regression problem

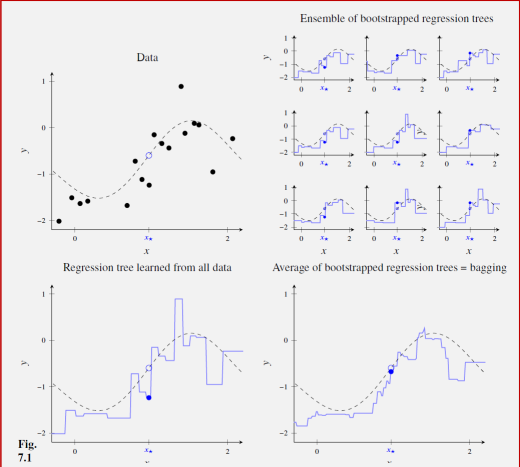


Fig. 7.1

- To reduce the risk of **overfitting**.

Bagging

Algorithm 7.1: The bootstrap.

```
Data: Training dataset $\mathcal{T} = \{\mathbf{x}_i, y_i\}_{i=1}^n$
Result: Bootstrapped data $\widetilde{\mathcal{T}} = \{\widetilde{\mathbf{x}}_i, \widetilde{y}_i\}_{i=1}^n$
for $i = 1, \dots, n$ do
    Sample $\ell$ uniformly on the set of integers $\{1, \dots, n\}$
    Set $\widetilde{\mathbf{x}}_i = \mathbf{x}_\ell$ and $\widetilde{y}_i = y_\ell$
end
```

Learn all base models

```
Data: Training dataset $\mathcal{T} = \{\mathbf{x}_i, y_i\}_{i=1}^n$
Result: $B$ base models
for $b = 1, \dots, B$ do
    Run Algorithm 7.1 to obtain a bootstrapped training dataset $\widetilde{\mathcal{T}}^{(b)}$
    Learn a base model from $\widetilde{\mathcal{T}}^{(b)}$
end
Obtain $\widehat{y}_{\text{bag}}(\mathbf{x}_\star)$ or $\mathbf{g}_{\text{bag}}(\mathbf{x}_\star)$ by averaging (7.1).
```

Predict with the base models

```
Data: $B$ base models and test input $\mathbf{x}_\star$
Result: A prediction $\widehat{y}_{\text{bag}}(\mathbf{x}_\star)$ or $\mathbf{g}_{\text{bag}}(\mathbf{x}_\star)$
for $b = 1, \dots, B$ do
    Use base model $b$ to predict $\widetilde{y}^{(b)}(\mathbf{x}_\star)$ or $\widetilde{\mathbf{g}}^{(b)}(\mathbf{x}_\star)$
end
Obtain $\widehat{y}_{\text{bag}}(\mathbf{x}_\star)$ or $\mathbf{g}_{\text{bag}}(\mathbf{x}_\star)$ by averaging (7.1).
```

$$\widehat{y}_{\text{bag}}(\mathbf{x}_\star) = \frac{1}{B}\sum_{b=1}^{B} \widetilde{y}^{(b)}(\mathbf{x}_\star) \quad \text{or} \quad \mathbf{g}_{\text{bag}}(\mathbf{x}_\star) = \frac{1}{B}\sum_{b=1}^{B} \widetilde{\mathbf{g}}^{(b)}(\mathbf{x}_\star), \tag{7.1}$$

Method 7.1: Bagging

Bagging

Algorithm 7.1: The bootstrap.

```
Data: Training dataset $\mathcal{T} = \{\mathbf{x}_i, y_i\}_{i=1}^n$
Result: Bootstrapped data $\widetilde{\mathcal{T}} = \{\widetilde{\mathbf{x}}_i, \widetilde{y}_i\}_{i=1}^n$
1 for $i = 1, \ldots, n$ do
2     Sample $\ell$ uniformly on the set of integers $\{1, \ldots, n\}$
3     Set $\widetilde{\mathbf{x}}_i = \mathbf{x}_\ell$ and $\widetilde{y}_i = y_\ell$
4 end
```

Learn all base models

```
Data: Training dataset $\mathcal{T} = \{\mathbf{x}_i, y_i\}_{i=1}^n$
Result: $B$ base models
1 for $b = 1, \ldots, B$ do
2     Run Algorithm 7.1 to obtain a bootstrapped training dataset $\widetilde{\mathcal{T}}^{(b)}$
3     Learn a base model from $\widetilde{\mathcal{T}}^{(b)}$
4 end
5 Obtain $\widehat{y}_{\text{bag}}(\mathbf{x}_\star)$ or $\mathbf{g}_{\text{bag}}(\mathbf{x}_\star)$ by averaging (7.1).
```

Predict with the base models

```
Data: $B$ base models and test input $\mathbf{x}_\star$
Result: A prediction $\widehat{y}_{\text{bag}}(\mathbf{x}_\star)$ or $\mathbf{g}_{\text{bag}}(\mathbf{x}_\star)$
1 for $b = 1, \ldots, B$ do
2     Use base model $b$ to predict $\widetilde{y}^{(b)}(\mathbf{x}_\star)$ or $\widetilde{\mathbf{g}}^{(b)}(\mathbf{x}_\star)$
3 end
4 Obtain $\widehat{y}_{\text{bag}}(\mathbf{x}_\star)$ or $\mathbf{g}_{\text{bag}}(\mathbf{x}_\star)$ by averaging (7.1).
```

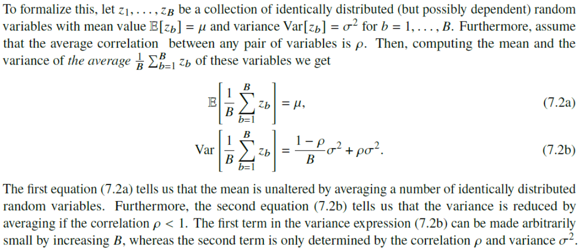
$$\widehat{y}_{\text{bag}}(\mathbf{x}_\star) = \frac{1}{B}\sum_{b=1}^{B} \widetilde{y}^{(b)}(\mathbf{x}_\star) \quad \text{or} \quad \mathbf{g}_{\text{bag}}(\mathbf{x}_\star) = \frac{1}{B}\sum_{b=1}^{B} \widetilde{\mathbf{g}}^{(b)}(\mathbf{x}_\star), \tag{7.1}$$

Method 7.1: Bagging

- Use majority voting in Equation 7.1 if the base models return class labels instead of class probabilities.

Bagging

- Averaging random variables reduces the variance.



To formalize this, let $z_1, \dots, z_B$ be a collection of identically distributed (but possibly dependent) random variables with mean value $\mathbb{E}[z_b] = \mu$ and variance $\mathrm{Var}[z_b] = \sigma^2$ for $b = 1, \dots, B$. Furthermore, assume that the average correlation between any pair of variables is ρ. Then, computing the mean and the variance of *the average* $\frac{1}{B}\sum_{b=1}^{B} z_b$ of these variables we get

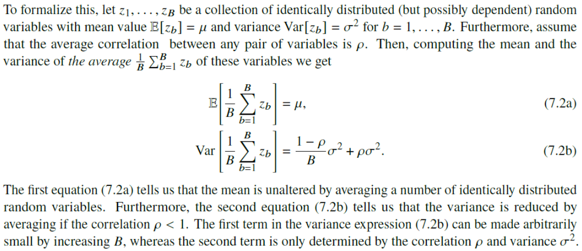
$$\mathbb{E}\left[\frac{1}{B}\sum_{b=1}^{B} z_b\right] = \mu, \tag{7.2a}$$

$$\mathrm{Var}\left[\frac{1}{B}\sum_{b=1}^{B} z_b\right] = \frac{1-\rho}{B}\sigma^2 + \rho\sigma^2. \tag{7.2b}$$

The first equation (7.2a) tells us that the mean is unaltered by averaging a number of identically distributed random variables. Furthermore, the second equation (7.2b) tells us that the variance is reduced by averaging if the correlation $\rho < 1$. The first term in the variance expression (7.2b) can be made arbitrarily small by increasing B, whereas the second term is only determined by the correlation ρ and variance σ^2.

Bagging

- Averaging random variables reduces the variance.



To formalize this, let $z_1, \dots, z_B$ be a collection of identically distributed (but possibly dependent) random variables with mean value $\mathbb{E}[z_b] = \mu$ and variance $\mathrm{Var}[z_b] = \sigma^2$ for $b = 1, \dots, B$. Furthermore, assume that the average correlation between any pair of variables is ρ. Then, computing the mean and the variance of *the average* $\frac{1}{B}\sum_{b=1}^{B} z_b$ of these variables we get

$$\mathbb{E}\left[\frac{1}{B}\sum_{b=1}^{B} z_b\right] = \mu, \tag{7.2a}$$

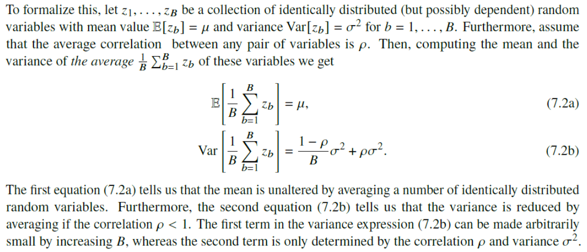
$$\mathrm{Var}\left[\frac{1}{B}\sum_{b=1}^{B} z_b\right] = \frac{1-\rho}{B}\sigma^2 + \rho\sigma^2. \tag{7.2b}$$

The first equation (7.2a) tells us that the mean is unaltered by averaging a number of identically distributed random variables. Furthermore, the second equation (7.2b) tells us that the variance is reduced by averaging if the correlation $\rho < 1$. The first term in the variance expression (7.2b) can be made arbitrarily small by increasing B, whereas the second term is only determined by the correlation ρ and variance σ^2.

- The base models' predictions can be seen as random variables. Since they originate from the same data via bootstrapping, they are identically distributed but correlated.

Bagging

- Averaging random variables reduces the variance.



To formalize this, let $z_1, \dots, z_B$ be a collection of identically distributed (but possibly dependent) random variables with mean value $\mathbb{E}[z_b] = \mu$ and variance $\mathrm{Var}[z_b] = \sigma^2$ for $b = 1, \dots, B$. Furthermore, assume that the average correlation between any pair of variables is ρ. Then, computing the mean and the variance of *the average* $\frac{1}{B}\sum_{b=1}^{B} z_b$ of these variables we get

$$\mathbb{E}\left[\frac{1}{B}\sum_{b=1}^{B} z_b\right] = \mu, \tag{7.2a}$$

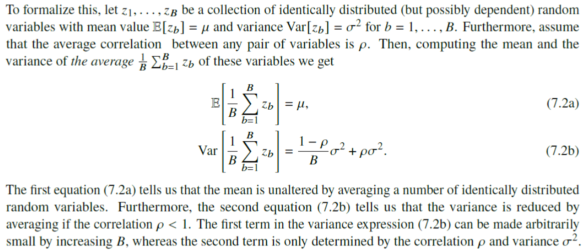
$$\mathrm{Var}\left[\frac{1}{B}\sum_{b=1}^{B} z_b\right] = \frac{1-\rho}{B}\sigma^2 + \rho\sigma^2. \tag{7.2b}$$

The first equation (7.2a) tells us that the mean is unaltered by averaging a number of identically distributed random variables. Furthermore, the second equation (7.2b) tells us that the variance is reduced by averaging if the correlation $\rho < 1$. The first term in the variance expression (7.2b) can be made arbitrarily small by increasing B, whereas the second term is only determined by the correlation ρ and variance σ^2.

- The base models' predictions can be seen as random variables. Since they originate from the same data via bootstrapping, they are identically distributed but correlated.
- Thus, bagging reduces the variance of the base model's predictions without increasing the bias. Thus, bagging reduces the risk of overfitting, because **a too large variance indicates risk of overfitting** to the bootstrapped training data.

Bagging

- Averaging random variables reduces the variance.



To formalize this, let $z_1, \ldots, z_B$ be a collection of identically distributed (but possibly dependent) random variables with mean value $\mathbb{E}[z_b] = \mu$ and variance $\mathrm{Var}[z_b] = \sigma^2$ for $b = 1, \ldots, B$. Furthermore, assume that the average correlation between any pair of variables is ρ. Then, computing the mean and the variance of *the average* $\frac{1}{B}\sum_{b=1}^{B} z_b$ of these variables we get

$$\mathbb{E}\left[\frac{1}{B}\sum_{b=1}^{B} z_b\right] = \mu, \tag{7.2a}$$

$$\mathrm{Var}\left[\frac{1}{B}\sum_{b=1}^{B} z_b\right] = \frac{1-\rho}{B}\sigma^2 + \rho\sigma^2. \tag{7.2b}$$

The first equation (7.2a) tells us that the mean is unaltered by averaging a number of identically distributed random variables. Furthermore, the second equation (7.2b) tells us that the variance is reduced by averaging if the correlation $\rho < 1$. The first term in the variance expression (7.2b) can be made arbitrarily small by increasing B, whereas the second term is only determined by the correlation ρ and variance σ^2.

- The base models' predictions can be seen as random variables. Since they originate from the same data via bootstrapping, they are identically distributed but correlated.
- Thus, bagging reduces the variance of the base model's predictions without increasing the bias. Thus, bagging reduces the risk of overfitting, because **a too large variance indicates risk of overfitting** to the bootstrapped training data.
- Bagging improves the base model trained on the bootstrapped data. Does it improve the base model trained on the original data? Yes, because they usually perform similarly.

Bagging

Example 7.3: Bagging for regression (cont.)

We consider the problem from Example 7.1 again, and explore how the number of base models B affects the result. We measure the squared error between the “true” function value at $x_\star$ and the predicted $\widehat{y}^{\text{bag}}(x_\star)$ when using different B. (Because of the bootstrap, there is a certain amount of randomness in the bagging algorithm itself. To avoid that “noise”, we average the result over multiple runs of the bagging algorithm.)

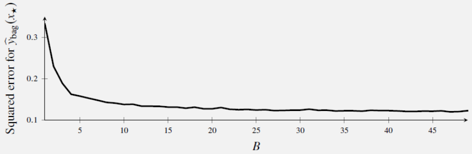


Fig. 7.4

What we see in Figure 7.4 is that the squared error eventually reaches a plateau as $B \to \infty$. Had there been an overfitting issue with $B \to \infty$, the squared error would have started to increase again for some large value of B.

Bagging

Example 7.3: Bagging for regression (cont.)

We consider the problem from Example 7.1 again, and explore how the number of base models B affects the result. We measure the squared error between the “true” function value at $x_\star$ and the predicted $\widehat{y}^{\text{bag}}(x_\star)$ when using different B. (Because of the bootstrap, there is a certain amount of randomness in the bagging algorithm itself. To avoid that “noise”, we average the result over multiple runs of the bagging algorithm.)

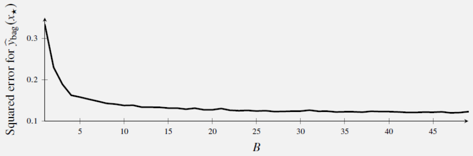


Fig. 7.4

What we see in Figure 7.4 is that the squared error eventually reaches a plateau as $B \to \infty$. Had there been an overfitting issue with $B \to \infty$, the squared error would have started to increase again for some large value of B.

- Out-of-bag error estimation:
  - For each training data point $\{\boldsymbol{x}_i, y_i\}$, do the following
    - Consider the ensemble's base models that did not see $\{\boldsymbol{x}_i, y_i\}$ during training. These are known to be roughly $B/3$.
    - Build an ensemble with them. This is called the i-th out-of-bag ensemble.
    - Compute the error when the i-th out-of-bag ensemble predicts $\{\boldsymbol{x}_i, y_i\}$.
  - Average the errors, and denote it as E_{OOB}.

Bagging

Example 7.3: Bagging for regression (cont.)

We consider the problem from Example 7.1 again, and explore how the number of base models B affects the result. We measure the squared error between the “true” function value at $x_\star$ and the predicted $\hat{y}^{\text{bag}}(x_\star)$ when using different B. (Because of the bootstrap, there is a certain amount of randomness in the bagging algorithm itself. To avoid that “noise”, we average the result over multiple runs of the bagging algorithm.)

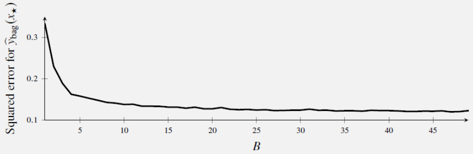


Fig. 7.4

What we see in Figure 7.4 is that the squared error eventually reaches a plateau as $B \to \infty$. Had there been an overfitting issue with $B \to \infty$, the squared error would have started to increase again for some large value of B.

- Out-of-bag error estimation:
  - For each training data point $\{\mathbf{x}_i, y_i\}$, do the following
    - Consider the ensemble’s base models that did not see $\{\mathbf{x}_i, y_i\}$ during training. These are known to be roughly $B/3$.
    - Build an ensemble with them. This is called the i-th out-of-bag ensemble.
    - Compute the error when the i-th out-of-bag ensemble predicts $\{\mathbf{x}_i, y_i\}$.
  - Average the errors, and denote it as E_{OOB}.
- E_{OOB} is an estimate of the expected new data error E_{new} for an ensemble with $B/3$ base models. As seen above, if B is large then ensembles with B and $B/3$ base models usually perform similarly.

Bagging

Example 7.3: Bagging for regression (cont.)

We consider the problem from Example 7.1 again, and explore how the number of base models B affects the result. We measure the squared error between the "true" function value at $x_\star$ and the predicted $\widehat{y}^{\text{bag}}(x_\star)$ when using different B. (Because of the bootstrap, there is a certain amount of randomness in the bagging algorithm itself. To avoid that "noise", we average the result over multiple runs of the bagging algorithm.)

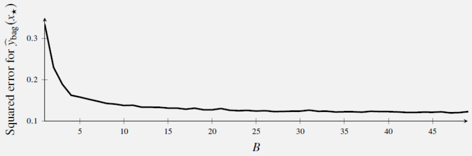


Fig. 7.4

What we see in Figure 7.4 is that the squared error eventually reaches a plateau as $B \to \infty$. Had there been an overfitting issue with $B \to \infty$, the squared error would have started to increase again for some large value of B.

- Out-of-bag error estimation:
  - For each training data point $\{\mathbf{x}_i, y_i\}$, do the following
    - Consider the ensemble's base models that did not see $\{\mathbf{x}_i, y_i\}$ during training. These are known to be roughly $B/3$.
    - Build an ensemble with them. This is called the i-th out-of-bag ensemble.
    - Compute the error when the i-th out-of-bag ensemble predicts $\{\mathbf{x}_i, y_i\}$.
  - Average the errors, and denote it as E_{OOB}.
- E_{OOB} is an estimate of the expected new data error E_{new} for an ensemble with $B/3$ base models. As seen above, if B is large then ensembles with B and $B/3$ base models usually perform similarly.
- Computing E_{OOB} is much cheaper than using cross-validation, as no re-training is required.

Random Forests

- In bagging, the **variance reduction is limited by the average correlation** of the ensemble's base models' predictions (see the second term in Equation 7.2b and note that the first term can be made arbitrarily small by increasing B).

Random Forests

- In bagging, the **variance reduction is limited by the average correlation** of the ensemble's base models' predictions (see the second term in Equation 7.2b and note that the first term can be made arbitrarily small by increasing B).
- Thus, we can improve bagging by **decorrelating** the ensemble's base models' predictions. This can easily be done when the base models are decision trees. Specifically, every time a split variable is about to be selected,
  - select q of the p input variables at random, and
  - consider only the q selected variables as possible split variables.

Random Forests

- In bagging, the **variance reduction is limited by the average correlation** of the ensemble's base models' predictions (see the second term in Equation 7.2b and note that the first term can be made arbitrarily small by increasing B).
- Thus, we can improve bagging by **decorrelating** the ensemble's base models' predictions. This can easily be done when the base models are decision trees. Specifically, every time a split variable is about to be selected,
  - select q of the p input variables at random, and
  - consider only the q selected variables as possible split variables.
- The base decision trees use different sets of split variables, which reduces the correlation of their predictions but increases their variance. In other words, in terms of Equation 7.2b, random forests decrease ρ but increase σ^2. Experience has shown that the former is the dominant effect.

Random Forests

- In bagging, the **variance reduction is limited by the average correlation** of the ensemble's base models' predictions (see the second term in Equation 7.2b and note that the first term can be made arbitrarily small by increasing B).
- Thus, we can improve bagging by **decorrelating** the ensemble's base models' predictions. This can easily be done when the base models are decision trees. Specifically, every time a split variable is about to be selected,
  - select q of the p input variables at random, and
  - consider only the q selected variables as possible split variables.
- The base decision trees use different sets of split variables, which reduces the correlation of their predictions but increases their variance. In other words, in terms of Equation 7.2b, random forests decrease ρ but increase σ^2. Experience has shown that the former is the dominant effect.
- As a rule-of-thumb, set $q = \sqrt{p}$ for classification and $q = p/3$ for regression. A more principled approach is to use cross-validation or E_{OOB}.

Random Forests

- In bagging, the **variance reduction is limited by the average correlation** of the ensemble's base models' predictions (see the second term in Equation 7.2b and note that the first term can be made arbitrarily small by increasing B).
- Thus, we can improve bagging by **decorrelating** the ensemble's base models' predictions. This can easily be done when the base models are decision trees. Specifically, every time a split variable is about to be selected,
  - select q of the p input variables at random, and
  - consider only the q selected variables as possible split variables.
- The base decision trees use different sets of split variables, which reduces the correlation of their predictions but increases their variance. In other words, in terms of Equation 7.2b, random forests decrease ρ but increase σ^2. Experience has shown that the former is the dominant effect.
- As a rule-of-thumb, set $q = \sqrt{p}$ for classification and $q = p/3$ for regression. A more principled approach is to use cross-validation or E_{OOB}.
- In summary, random forests $=$ decision trees $+$ bagging $+$ decorrelation.

Random Forests

Example 7.4: Random forests and bagging for a binary classification problem

The most apparent difference is the higher individual variation of the random forest ensemble members compared to bagging. Roughly half of the random forest ensemble members have been forced to make the first split along the horizontal axis, which has lead to an increased variance and a decreased correlation, compared to bagging where all ensemble members make the first split along the vertical axis.

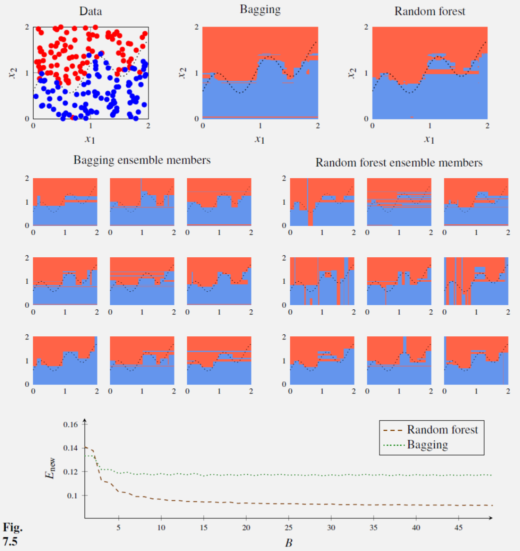


Fig. 7.5

Boosting

- To reduce the risk of **underfitting** by reducing the bias of the base model's predictions (cf. bagging).

Example 7.5: Boosting illustration

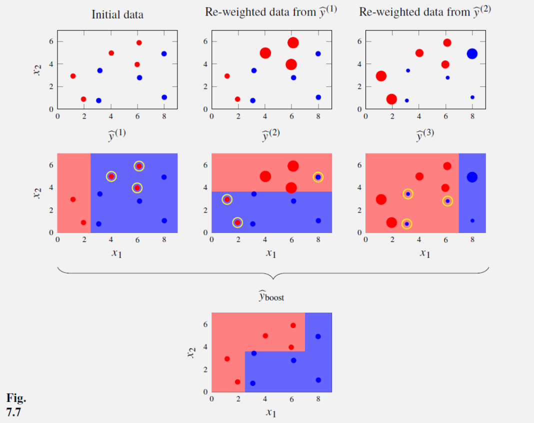


Fig. 7.7

Adaptive Boosting (AdaBoost)

- For binary ($y \in \{-1, +1\}$) classification.

```
Learn an AdaBoost classifier
  Data: Training data $\mathcal{T} = \{\mathbf{x}_i, y_i\}_{i=1}^n$
  Result: $B$ weak classifiers
1 Assign weights $w_i^{(1)} = 1/n$ to all data points.
2 for $b = 1, \dots, B$ do
3     Train a weak classifier $\widehat{y}^{(b)}(\mathbf{x})$ on the weighted training data $\{(\mathbf{x}_i, y_i, w_i^{(b)})\}_{i=1}^n$.
4     Compute $E_{\text{train}}^{(b)} = \sum_{i=1}^n w_i^{(b)} \mathbb{I}\{y_i \neq \widehat{y}^{(b)}(\mathbf{x}_i)\}$.
5     Compute $\alpha^{(b)} = 0.5 \ln((1 - E_{\text{train}}^{(b)})/E_{\text{train}}^{(b)})$.
6     Compute $w_i^{(b+1)} = w_i^{(b)} \exp(-\alpha^{(b)} y_i \widehat{y}^{(b)}(\mathbf{x}_i)), i = 1, \dots, n$.
7     Set $w_i^{(b+1)} \leftarrow w_i^{(b+1)} / \sum_{j=1}^n w_j^{(b+1)}$, for $i = 1, \dots, n$.
8 end

Predict with the AdaBoost classifier
  Data: $B$ weak classifiers with confidence values $\{\widehat{y}^{(b)}(\mathbf{x}), \alpha^{(b)}\}_{b=1}^B$ and test input $\mathbf{x}_\star$
  Result: Prediction $\widehat{y}_{\text{boost}}^{(B)}(\mathbf{x}_\star)$
1 Output $\widehat{y}_{\text{boost}}^{(B)}(\mathbf{x}_\star) = \text{sign}\left\{\sum_{b=1}^B \alpha^{(b)} \widehat{y}^{(b)}(\mathbf{x}_\star)\right\}$.
```

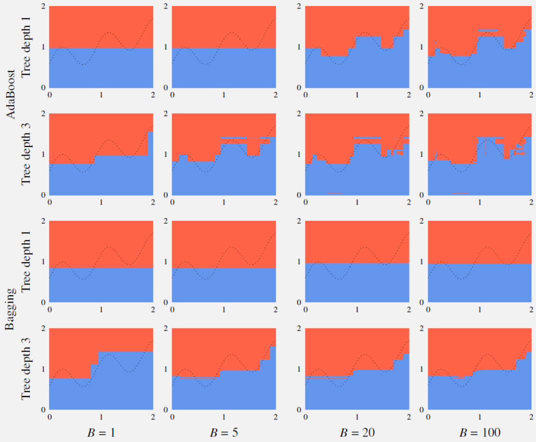
Method 7.2: AdaBoost

- Steps 3-4: Training a weighted classifier is simple for most base classifiers, as the cost function is typically a sum whose terms only need to be weighted.
- Step 5: Note that $\alpha^{(b)}$ can be seen as the confidence in the b-th member's predictions, i.e. if $E_{train}^{(a)} \leq E_{train}^{(b)}$ then $\alpha^{(a)} \geq \alpha^{(b)}$.
- Step 6-7: **The weight increases for a misclassified training point** and decreases otherwise.

AdaBoost

Example 7.6: AdaBoost and bagging for a binary classification example

The decision boundaries for each method with $B = 1, 5, 20$ and 100 ensemble members are shown in Figure 7.8. Despite using quite weak ensemble members (a shallow tree has high bias), AdaBoost adapts quite well to the data. This is in contrast to bagging, where the decision boundary does not become much more flexible despite using many ensemble members. In other words, AdaBoost reduces the bias of the base model, whereas bagging only has minor effect on the bias.



AdaBoost

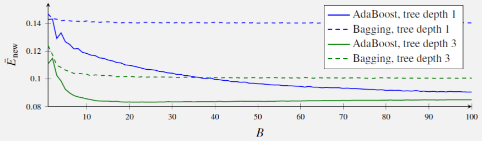


Fig. 7.8

We also numerically compute $\bar{E}_{\text{new}}$ for this problem, as a function of B, which is shown in the bottom of Figure 7.8. Remember that $\bar{E}_{\text{new}}$ depends on both the bias and the variance. As discussed, the main effect of bagging is variance reduction, but that does not help much since the base model already is quite low-variance (but high-bias). Boosting, on the other hand, reduces bias, which has a much bigger effect in this case. Furthermore, bagging does not overfit as $B \to \infty$, but that is *not* the case for boosting! We can indeed see that for trees of depth 3, the smallest $\bar{E}_{\text{new}}$ is obtained for $B \approx 25$, and there is actually a slight increase in $\bar{E}_{\text{new}}$ for larger values of B. Hence, AdaBoost with depth-3 trees suffers from a (minor) overfit as $B \gtrsim 25$ in this problem.

AdaBoost

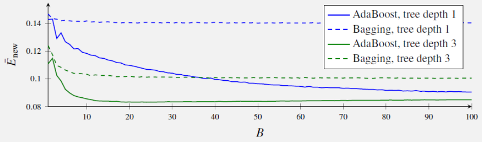


Fig. 7.8

We also numerically compute $\bar{E}_{\text{new}}$ for this problem, as a function of B, which is shown in the bottom of Figure 7.8. Remember that $\bar{E}_{\text{new}}$ depends on both the bias and the variance. As discussed, the main effect of bagging is variance reduction, but that does not help much since the base model already is quite low-variance (but high-bias). Boosting, on the other hand, reduces bias, which has a much bigger effect in this case. Furthermore, bagging does not overfit as $B \to \infty$, but that is *not* the case for boosting! We can indeed see that for trees of depth 3, the smallest $\bar{E}_{\text{new}}$ is obtained for $B \approx 25$, and there is actually a slight increase in $\bar{E}_{\text{new}}$ for larger values of B. Hence, AdaBoost with depth-3 trees suffers from a (minor) overfit as $B \gtrsim 25$ in this problem.

- Shallow decision trees are typically used as base models in AdaBoost. They can be trained quickly. That they are biased is not a problem, since AdaBoost is designed to address this precise problem.

AdaBoost

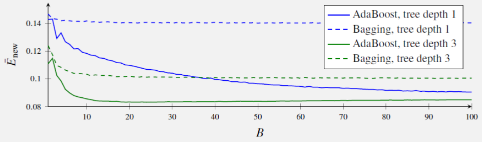


Fig. 7.8

We also numerically compute $\bar{E}_{\text{new}}$ for this problem, as a function of B, which is shown in the bottom of Figure 7.8. Remember that $\bar{E}_{\text{new}}$ depends on both the bias and the variance. As discussed, the main effect of bagging is variance reduction, but that does not help much since the base model already is quite low-variance (but high-bias). Boosting, on the other hand, reduces bias, which has a much bigger effect in this case. Furthermore, bagging does not overfit as $B \to \infty$, but that is *not* the case for boosting! We can indeed see that for trees of depth 3, the smallest $\bar{E}_{\text{new}}$ is obtained for $B \approx 25$, and there is actually a slight increase in $\bar{E}_{\text{new}}$ for larger values of B. Hence, AdaBoost with depth-3 trees suffers from a (minor) overfit as $B \gtrsim 25$ in this problem.

- Shallow decision trees are typically used as base models in AdaBoost. They can be trained quickly. That they are biased is not a problem, since AdaBoost is designed to address this precise problem.
- Each iteration of AdaBoost adds a new member to the ensemble to classify correctly the previously misclassified points. So, the ensemble becomes more flexible with the number of iterations. Thus, **the risk of overfitting increases**, as seen above (cf. bagging). Experience has shown that this risk increases slowly. Nevertheless, using early stopping is advised.

Gradient Boosting

- Boosting can be described as learning a sequence of weak predictors, where each tries to correct the mistakes made by the previous ones.

Gradient Boosting

- Boosting can be described as learning a sequence of weak predictors, where each tries to correct the mistakes made by the previous ones.
- More generally, boosting can be described as learning an additive model, i.e. an additive expansion of some basis functions. **For classification, the basis functions do not need to be weak classifiers**.

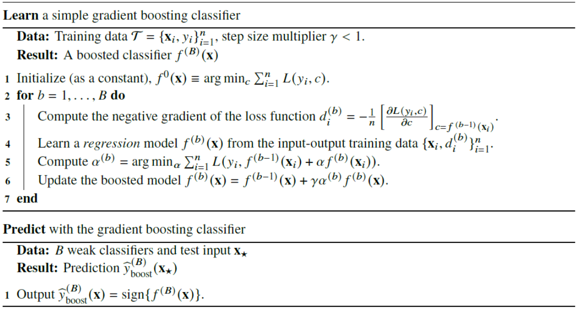
Gradient Boosting

- Boosting can be described as learning a sequence of weak predictors, where each tries to correct the mistakes made by the previous ones.
- More generally, boosting can be described as learning an additive model, i.e. an additive expansion of some basis functions. **For classification, the basis functions do not need to be weak classifiers**.
- The differences with other additive models are that boosting learns the additive model sequentially, and that the basis functions are learned from data.

Gradient Boosting

- Boosting can be described as learning a sequence of weak predictors, where each tries to correct the mistakes made by the previous ones.
- More generally, boosting can be described as learning an additive model, i.e. an additive expansion of some basis functions. **For classification, the basis functions do not need to be weak classifiers**.
- The differences with other additive models are that boosting learns the additive model sequentially, and that the basis functions are learned from data.
- Akin to gradient descent, each iteration of boosting adds to the additive model a basis function that is **close to the negative gradient of some cost function**.

Gradient Boosting



```
Learn a simple gradient boosting classifier
  Data: Training data $\mathcal{T} = \{\mathbf{x}_i, y_i\}_{i=1}^n$, step size multiplier $\gamma < 1$.
  Result: A boosted classifier $f^{(B)}(\mathbf{x})$
1 Initialize (as a constant), $f^0(\mathbf{x}) \equiv \arg\min_c \sum_{i=1}^n L(y_i, c)$.
2 for $b = 1, \dots, B$ do
3     Compute the negative gradient of the loss function $d_i^{(b)} = -\frac{1}{n}\left[\frac{\partial L(y_i,c)}{\partial c}\right]_{c=f^{(b-1)}(\mathbf{x}_i)}$.
4     Learn a regression model $f^{(b)}(\mathbf{x})$ from the input-output training data $\{\mathbf{x}_i, d_i^{(b)}\}_{i=1}^n$.
5     Compute $\alpha^{(b)} = \arg\min_\alpha \sum_{i=1}^n L(y_i, f^{(b-1)}(\mathbf{x}_i) + \alpha f^{(b)}(\mathbf{x}_i))$.
6     Update the boosted model $f^{(b)}(\mathbf{x}) = f^{(b-1)}(\mathbf{x}) + \gamma\alpha^{(b)} f^{(b)}(\mathbf{x})$.
7 end

Predict with the gradient boosting classifier
  Data: $B$ weak classifiers and test input $\mathbf{x}_\star$
  Result: Prediction $\widehat{y}_{\text{boost}}^{(B)}(\mathbf{x}_\star)$
1 Output $\widehat{y}_{\text{boost}}^{(B)}(\mathbf{x}) = \text{sign}\{f^{(B)}(\mathbf{x})\}$.
```

Method 7.3: A simple gradient boosting algorithm

- Step 3: Easy sometimes, e.g. if $L(y_i, c) = (y_i - c)^2$ then $\partial L(y_i, c)/\partial c = -2(y_i - c)$.
- Steps 3-4: The function $f^{(b)}(\boldsymbol{x})$ is not exactly the negative gradient because the basis functions are restricted to some functional form, e.g. decision trees of a certain depth. Moreover, $f^{(b)}(\boldsymbol{x})$ is a regressor (even for a classification problem) because the gradient is real-valued.
- Step 5: $\alpha^{(b)}$ can be seen as the learning rate, which is learned from data as opposed to the user-defined learning rate in gradient descent.
- Step 6: Experience has shown that multiplying the learning rate with a constant $\gamma < 1$ introduces some beneficial regularization.

Summary

- Bias-Variance Decomposition
- Bagging
- Random Forests
- AdaBoost
- Gradient Boosting