MATH 567: Mathematical Techniques in Data Science
Random forest

Dominique Guillot

Departments of Mathematical Sciences
University of Delaware

April 24, 2017
We saw before that decision trees often overfit the data. We will now discuss techniques that can be used to mitigate that problem.

**The bootstrap**

- General statistical method that relies on resampling data with replacement.
- Idea: Given data \((y_i, x_i)\), \(i = 1, \ldots, n\), construct bootstrap samples by sampling \(n\) of the observations with replacement (i.e., allow repetitions):

<table>
<thead>
<tr>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>((y_{i1}, x_{i1}))</td>
<td>((y_{j1}, x_{j1}))</td>
<td>((y_{k1}, x_{k1}))</td>
</tr>
<tr>
<td>((y_{i2}, x_{i2}))</td>
<td>((y_{j2}, x_{j2}))</td>
<td>((y_{k2}, x_{k2}))</td>
</tr>
<tr>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>((y_{in}, x_{in}))</td>
<td>((y_{jn}, x_{jn}))</td>
<td>((y_{kn}, x_{kn}))</td>
</tr>
</tbody>
</table>

Each bootstrap sample mimics the statistical properties of the original data. Often used to estimate parameter variability (or uncertainty).
The bootstrap

- We saw before that decision trees often overfit the data.
- We will now discuss techniques that can be used to mitigate that problem.

**Bootstrapping**: General statistical method that relies on resampling data with replacement.
The bootstrap

- We saw before that decision trees often overfit the data.
- We will now discuss techniques that can be used to mitigate that problem.

**Bootstrapping:** General statistical method that relies on resampling data with replacement.

Idea: Given data \((y_i, x_i), i = 1, \ldots, n\), construct *bootstrap samples* by sampling \(n\) of the observations *with replacement* (i.e., allow repetitions):

<table>
<thead>
<tr>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>((y_{i_1}, x_{i_1}))</td>
<td>((y_{j_1}, x_{j_1}))</td>
<td>((y_{k_1}, x_{k_1}))</td>
</tr>
<tr>
<td>((y_{i_2}, x_{i_2}))</td>
<td>((y_{j_2}, x_{j_2}))</td>
<td>((y_{k_2}, x_{k_2}))</td>
</tr>
<tr>
<td>(\vdots)</td>
<td>(\vdots)</td>
<td>(\vdots)</td>
</tr>
<tr>
<td>((y_{i_n}, x_{i_n}))</td>
<td>((y_{j_n}, x_{j_n}))</td>
<td>((y_{k_n}, x_{k_n}))</td>
</tr>
</tbody>
</table>
We saw before that decision trees often overfit the data.
We will now discuss techniques that can be used to mitigate that problem.

**Bootstrapping**: General statistical method that relies on resampling data with replacement.

Idea: Given data \((y_i, x_i), i = 1, \ldots, n\), construct *bootstrap samples* by sampling \(n\) of the observations *with replacement* (i.e., allow repetitions):

<table>
<thead>
<tr>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>((y_{i_1}, x_{i_1}))</td>
<td>((y_{j_1}, x_{j_1}))</td>
<td>((y_{k_1}, x_{k_1}))</td>
</tr>
<tr>
<td>((y_{i_2}, x_{i_2}))</td>
<td>((y_{j_2}, x_{j_2}))</td>
<td>((y_{k_2}, x_{k_2}))</td>
</tr>
<tr>
<td>(\vdots)</td>
<td>(\vdots)</td>
<td>(\vdots)</td>
</tr>
<tr>
<td>((y_{i_n}, x_{i_n}))</td>
<td>((y_{j_n}, x_{j_n}))</td>
<td>((y_{k_n}, x_{k_n}))</td>
</tr>
</tbody>
</table>

- Each bootstrap sample mimics the statistical properties of the original data.
- Often used to estimate parameter variability (or uncertainty).
Bagging (bootstrap aggregation) Suppose we have a model $y \approx \hat{f}(x)$ for data $(y_i, x_i) \in \mathbb{R}^{p+1}$.

1. Construct $B \in \mathbb{N}$ bootstrap samples.
2. Train the method on the $b$-th bootstrap sample to get $\hat{f}^*_{b}(x)$.
3. Compute the average of the estimators:
   \[
   \hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{i=1}^{B} \hat{f}^*_{b}(x).
   \]

Bagging is often used with regression trees. Can improve estimators significantly.

Note: Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes. The bagged estimate is the average prediction at $x$ from these $B$ trees.

For classification: Use a majority vote from the $B$ trees.
Bagging:

(bootstrap aggregation) Suppose we have a model $y \approx \hat{f}(x)$ for data $(y_i, x_i) \in \mathbb{R}^{p+1}$.

1. Construct $B \in \mathbb{N}$ bootstrap samples.
Bagging: (bootstrap aggregation) Suppose we have a model $y \approx \hat{f}(x)$ for data $(y_i, x_i) \in \mathbb{R}^{p+1}$.

1. Construct $B \in \mathbb{N}$ bootstrap samples.
2. Train the method on the $b$-th bootstrap sample to get $\hat{f}^b(x)$. 

Bagging is often used with regression trees. Can improve estimators significantly. Note: Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes. The bagged estimate is the average prediction at $x$ from these $B$ trees.

For classification: Use a majority vote from the $B$ trees.
Bagging: (bootstrap aggregation) Suppose we have a model $y \approx \hat{f}(x)$ for data $(y_i, x_i) \in \mathbb{R}^{p+1}$.

1. Construct $B \in \mathbb{N}$ bootstrap samples.
2. Train the method on the $b$-th bootstrap sample to get $\hat{f}^b(x)$.
3. Compute the average of the estimators:

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

Bagging is often used with regression trees. It can improve estimators significantly.

Note: Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes.

The bagged estimate is the average prediction at $x$ from these $B$ trees.

For classification: Use a majority vote from the $B$ trees.
Bagging: (bootstrap aggregation) Suppose we have a model $y \approx \hat{f}(x)$ for data $(y_i, x_i) \in \mathbb{R}^{p+1}$.

1. Construct $B \in \mathbb{N}$ bootstrap samples.
2. Train the method on the $b$-th bootstrap sample to get $\hat{f}^*_b(x)$.
3. Compute the average of the estimators:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{i=1}^{B} \hat{f}^*_b(x).$$

- Bagging is often used with regression trees.
- Can improve estimators significantly.
**Bagging:** (bootstrap aggregation) Suppose we have a model $y \approx \hat{f}(x)$ for data $(y_i, x_i) \in \mathbb{R}^{p+1}$.

1. Construct $B \in \mathbb{N}$ bootstrap samples.
2. Train the method on the $b$-th bootstrap sample to get $\hat{f}^*_b(x)$.
3. Compute the average of the estimators:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{i=1}^{B} \hat{f}^*_b(x).$$

- Bagging is often used with regression trees.
- Can improve estimators significantly.

Note: Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes.
Bagging: (bootstrap aggregation) Suppose we have a model \( y \approx \hat{f}(x) \) for data \( (y_i, x_i) \in \mathbb{R}^{p+1} \).

1. Construct \( B \in \mathbb{N} \) bootstrap samples.
2. Train the method on the \( b \)-th bootstrap sample to get \( \hat{f}^b(x) \).
3. Compute the average of the estimators:

\[
\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{i=1}^{B} \hat{f}^b(x).
\]

- Bagging is often used with regression trees.
- Can improve estimators significantly.

Note: Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes.

The bagged estimate is the average prediction at \( x \) from these \( B \) trees.
Bagging: (bootstrap aggregation) Suppose we have a model
\[ y \approx \hat{f}(x) \] for data \( (y_i, x_i) \in \mathbb{R}^{p+1} \).

1. Construct \( B \in \mathbb{N} \) bootstrap samples.
2. Train the method on the \( b \)-th bootstrap sample to get \( \hat{f}^* (x) \).
3. Compute the average of the estimators:

\[
\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{i=1}^{B} \hat{f}^* (x).
\]

- Bagging is often used with regression trees.
- Can improve estimators significantly.

Note: Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes.

The bagged estimate is the average prediction at \( x \) from these \( B \) trees.

For classification: Use a majority vote from the \( B \) trees.
Simulation:

- $N = 30$ samples with $p = 5$ features.
- Features from a standard Gaussian distribution with pairwise correlation 0.95.
- $Y$ generated according to

$$P(Y = 1|X_1 \leq 0.5) = 0.2$$
$$P(Y = 1|X_1 > 0.5) = 0.8.$$
Simulation:

- $N = 30$ samples with $p = 5$ features.
- Features from a standard Gaussian distribution with pairwise correlation 0.95.
- $Y$ generated according to

$$P(Y = 1 | X_1 \leq 0.5) = 0.2$$
$$P(Y = 1 | X_1 > 0.5) = 0.8.$$ 

- A test sample of size 2,000 was also generated using the same model.
- The test error for the original tree and the bagged tree are reported.
Bootstrap trees:

Original Tree

\[ x_1 < 0.395 \]

\[ 0 \quad 1 \quad 0 \quad 1 \quad 0 \]

\[ 0 \quad 1 \]

\[ b = 1 \]

\[ x_1 < 0.555 \]

\[ 0 \quad 0 \quad 1 \quad 1 \]

\[ b = 2 \]

\[ x_2 < 0.205 \]

\[ 0 \quad 1 \quad 0 \quad 0 \]

\[ b = 3 \]

\[ x_2 < 0.285 \]

\[ 0 \quad 0 \quad 1 \quad 1 \]

\[ b = 4 \]

\[ x_3 < 0.985 \]

\[ 0 \quad 1 \quad 1 \]

\[ b = 5 \]

\[ x_4 < -1.36 \]

\[ 0 \quad 1 \quad 0 \quad 1 \]

\[ b = 6 \]

\[ x_1 < 0.395 \]

\[ 1 \quad 1 \quad 0 \quad 0 \]

\[ b = 7 \]

\[ x_1 < 0.555 \]

\[ 0 \quad 1 \quad 0 \quad 1 \]

\[ b = 8 \]

\[ x_3 < 0.985 \]

\[ 0 \quad 0 \quad 1 \]

\[ b = 9 \]

\[ x_1 < 0.395 \]

\[ 1 \quad 0 \quad 1 \quad 0 \]

\[ b = 10 \]

\[ x_1 < 0.555 \]

\[ 1 \quad 0 \quad 1 \]

\[ b = 11 \]

\[ x_1 < 0.555 \]

\[ 0 \quad 1 \quad 0 \]

ESL, Figure 8.9.
Test error:

Errors for the bagging example. (ESL, Figure 8.10.)

The orange points correspond to the consensus vote, while the green points average the probabilities.
**Test error:**

![Graph showing test error with bagged trees and original tree](image)

Errors for the bagging example. (ESL, Figure 8.10.)

The orange points correspond to the consensus vote, while the green points average the probabilities.

**Out-of-bag error:** Mean prediction error on each training sample $x_i$, using only the trees that did not have $x_i$ in their bootstrap sample.

Can be used to approximate the prediction error.
Random forests

- Idea of bagging: average many noisy but approximately unbiased models, and hence reduce the variance.

- However, the bootstrap trees are generally correlated.

- Random forests improve the variance reduction of bagging by reducing the correlation between the trees.

- Achieved in the tree-growing process through random selection of the input variables.

- Popular method.
Random forests (cont.): Each time a split in a tree is considered, a random selection of $m$ predictors is chosen as split candidates from the full set of $p$ predictors.
Random forests: Each time a split in a tree is considered, a random selection of $m$ predictors is chosen as split candidates from the full set of $p$ predictors.

- Typical value for $m$ is $\sqrt{p}$. 
Random forests: Each time a split in a tree is considered, a random selection of \( m \) predictors is chosen as split candidates from the full set of \( p \) predictors.

- Typical value for \( m \) is \( \sqrt{p} \).
- We construct \( T_1, \ldots, T_B \) trees using that method on bootstrap samples. The random forest (regression) predictor is

\[
\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x).
\]

For classification: use majority vote.
Example (Izenman, 2013)

Diagnostic classification of four childhood tumors (Khan et al., 2001):

- Small, round, blue-cell tumors (SRBCTs) of childhood.
- Four types of SRBCTs (EWS, BL, NB, RMS).
- Tumors have a similar appearance.
- Getting the diagnosis correct impacts directly upon the type of treatment, therapy, and prognosis the patient receives.
- Currently, no single clinical test that can discriminate between these cancers.

Data:

- 83 cases (29 EWS, 11 BL, 18 NB, 25 RMS).
- Gene expression data for 567 genes, reduced to 308 by requiring a minimum intensity.

A random forest was applied to these data using 500 fully grown trees with \( m = 25 \) variables at each split. Able to get a 0% Out-of-bag misclassification rate.

research.nhgri.nih.gov/microarray/Supplement
Diagnostic classification of four childhood tumors (Khan et al., 2001):

- Small, round, blue-cell tumors (SRBCTs) of childhood.
- Four types of SRBCTs (EWS, BL, NB, RMS).
- Tumors have a similar appearance.
- Getting the diagnosis correct impacts directly upon the type of treatment, therapy, and prognosis the patient receives.
- Currently, no single clinical test that can discriminate between these cancers.

Data:

- 83 cases (29 EWS, 11 BL, 18 NB, 25 RMS).
- Gene expression data for 6,567 genes, reduced to 2,308 by requiring a minimum intensity.
Diagnostic classification of four childhood tumors (Khan et al., 2001):

- Small, round, blue-cell tumors (SRBCTs) of childhood.
- Four types of SRBCTs (EWS, BL, NB, RMS).
- Tumors have a similar appearance.
- Getting the diagnosis correct impacts directly upon the type of treatment, therapy, and prognosis the patient receives.
- Currently, no single clinical test that can discriminate between these cancers.

Data:

- 83 cases (29 EWS, 11 BL, 18 NB, 25 RMS).
- Gene expression data for 6,567 genes, reduced to 2,308 by requiring a minimum intensity.
- A random forest was applied to these data using 500 fully grown trees with $m = 25$ variables at each split.
Diagnostic classification of four childhood tumors (Khan et al., 2001):

- Small, round, blue-cell tumors (SRBCTs) of childhood.
- Four types of SRBCTs (EWS, BL, NB, RMS).
- Tumors have a similar appearance.
- Getting the diagnosis correct impacts directly upon the type of treatment, therapy, and prognosis the patient receives.
- Currently, no single clinical test that can discriminate between these cancers.

Data:

- 83 cases (29 EWS, 11 BL, 18 NB, 25 RMS).
- Gene expression data for 6,567 genes, reduced to 2,308 by requiring a minimum intensity.
- A random forest was applied to these data using 500 fully grown trees with $m = 25$ variables at each split.
- Able to get a 0% Out-of-bag misclassification rate.
Like bagging, boosting is a general approach that can be applied to many models. *Combines weak learners into a single strong learner.*

---

**Boosting**


def boost_with_trees(X, y, B):
    # Initialize
    f = np.zeros(len(X))
    residuals = y
    
    for b in range(1, B + 1):
        # Fit a tree
        tree = fit_tree(X, residuals)
        
        # Update the estimator
        f += lambda_ * tree.predict(X)
        
        # Update the residuals
        residuals = y - lambda_ * tree.predict(X)

    return f

---

Note: $\lambda > 0$ is a learning rate.
Like bagging, boosting is a general approach that can be applied to many models. *Combines weak learners into a single strong learner.*

**Boosting:** Recursively fit trees to residuals. (Compensate the shortcoming of previous model.)
Like bagging, boosting is a general approach that can be applied to many models. *Combines weak learners into a single strong learner.*

**Boosting:** Recursively fit trees to residuals. (Compensate the shortcoming of previous model.)

**Input:** \((y_i, x_i) \in \mathbb{R}^{p+1}, i = 1, \ldots, n.\)
Like bagging, boosting is a general approach that can be applied to many models. *Combines weak learners into a single strong learner.*

**Boosting:** Recursively fit trees to residuals. (Compensate the shortcoming of previous model.)

**Input:** \((y_i, x_i) \in \mathbb{R}^{p+1}, i = 1, \ldots, n.\) Initialize \(\hat{f}(x) = 0, r_i = y_i.\)
Like bagging, boosting is a general approach that can be applied to many models. *Combines weak learners into a single strong learner.*

**Boosting:** Recursively fit trees to residuals. (Compensate the shortcoming of previous model.)

**Input:** \((y_i, x_i) \in \mathbb{R}^{p+1}, i = 1, \ldots, n\). Initialize \( \hat{f}(x) = 0, r_i = y_i \).

For \( b = 1, \ldots, B \):

1. Fit a tree estimator \( \hat{f}^b \) with \( d \) splits to the training data.
2. Update the estimator using:
   \[
   \hat{f}(x) \leftarrow \hat{f}(x) + \lambda \cdot \hat{f}^b(x).
   \]
3. Update the residuals:
   \[
   r_i \leftarrow r_i - \lambda \cdot \hat{f}^b(x_i).
   \]
Like bagging, boosting is a general approach that can be applied to many models. *Combines weak learners into a single strong learner.*

**Boosting:** Recursively fit trees to residuals. (Compensate the shortcoming of previous model.)

**Input:** \((y_i, x_i) \in \mathbb{R}^{p+1}, i = 1, \ldots, n.\) Initialize \(\hat{f}(x) = 0, r_i = y_i.\)

For \(b = 1, \ldots, B:\)

1. Fit a tree estimator \(\hat{f}^b\) with \(d\) splits to the training data.
2. Update the estimator using:
   \[
   \hat{f}(x) \leftarrow \hat{f}(x) + \lambda \cdot \hat{f}^b(x).
   \]
3. Update the residuals:
   \[
   r_i \leftarrow r_i - \lambda \cdot \hat{f}^b(x_i).
   \]

**Output:** Boosted tree:

\[
\hat{f}(x) = \sum_{i=1}^{B} \lambda \hat{f}^b(x).
\]

Note: \(\lambda > 0\) is a *learning rate.*
Can use many small trees (by choosing $d$ small) and learn slowly ($\lambda$ small) to avoid overfitting.
Can use many small trees (by choosing \( d \) small) and learn slowly (\( \lambda \) small) to avoid overfitting.

**Choosing the parameters:**

1. Number of trees \( B \): choose by cross-validation.
2. Number of splits: can use a small value (e.g. \( d = 1 \)).
3. Learning rate: can use 0.01, 0.001. Note: A small \( \lambda \) will generally require a larger \( B \)…
Can use many small trees (by choosing $d$ small) and learn slowly ($\lambda$ small) to avoid overfitting.

**Choosing the parameters:**

1. **Number of trees $B$:** choose by cross-validation.
2. **Number of splits:** can use a small value (e.g. $d = 1$).
3. **Learning rate:** can use 0.01, 0.001. Note: A small $\lambda$ will generally require a larger $B$.

**Gradient boosting:** More generally, one can work with a general loss function (instead of sum of squares) and replace the residuals with the (negative) of the gradient of the loss function.
The previous methodologies can improve decision trees considerably. However, we lose the nice interpretability of decision trees.
Relative importance of predictor variables

- The previous methodologies can improve decision trees considerably.
- However, we lose the nice interpretability of decision trees. A *relative importance* of each predictor can be computed to help understand a model with multiple trees.
Relative importance of predictor variables

- The previous methodologies can improve decision trees considerably.
- However, we lose the nice interpretability of decision trees. A *relative importance* of each predictor can be computed to help understand a model with multiple trees.
  - Let $T$ be a (binary) decision tree with $J - 1$ internal nodes.
Relative importance of predictor variables

- The previous methodologies can improve decision trees considerably.
- However, we lose the nice interpretability of decision trees. A relative importance of each predictor can be computed to help understand a model with multiple trees.

  - Let $T$ be a (binary) decision tree with $J - 1$ internal nodes.
  - At each internal node $t$, a variable $X_{v(t)}$ is split, resulting in an improvement $\hat{\iota}_t^2$ in squared error.
Relative importance of predictor variables

- The previous methodologies can improve decision trees considerably.
- However, we lose the nice interpretability of decision trees. A relative importance of each predictor can be computed to help understand a model with multiple trees.

Let $T$ be a (binary) decision tree with $J - 1$ internal nodes.
- At each internal node $t$, a variable $X_{v(t)}$ is split, resulting in an improvement $\hat{\iota}_t^2$ in squared error.
- We define a measure of relevance of $X_l$ by

$$I^2_l(T) := \sum_{t=1}^{J-1} \hat{\iota}_t^2 \cdot I(v(t) = l).$$

In other words, we add-up the improvements at the nodes where $X_l$ is split.
Similarly, in a model obtained from $M$ trees (e.g. bagging, random forest), we use:

$$I^2_{l} = \frac{1}{M} \sum_{m=1}^{M} I^2_l(T_m).$$
Similarly, in a model obtained from $M$ trees (e.g. bagging, random forest), we use:

$$\mathcal{I}_l^2 = \frac{1}{M} \sum_{m=1}^{M} \mathcal{I}_l^2(T_m).$$

Taking the square root of the relevance measure, we obtain the *relevance* of $X_l$. 
Similarly, in a model obtained from $M$ trees (e.g. bagging, random forest), we use:

$$\mathcal{I}_l^2 = \frac{1}{M} \sum_{m=1}^{M} \mathcal{I}_l^2(T_m).$$

Taking the square root of the relevance measure, we obtain the relevance of $X_l$.

Typically, we do not report the actual relevance of a variable. We rather report the percentage of relevance of a given variable with respect to the variable with the largest relevance.
Relative importance of predictor for the spam data

ESL, Figure 10.6.