Lecture 6 – Tree-based methods, Bagging and Random Forests

Fredrik Lindsten
Division of Systems and Control
Department of Information Technology
Uppsala University.

Email: fredrik.lindsten@it.uu.se
Summary of Lecture 5 (I/IV)

When choosing models/methods, we are interested in how well they will perform when faced with new unseen data.

The new data error

\[ E_{\text{new}} \triangleq \mathbb{E}_* [E(\hat{y}(x_*; \mathcal{T}), y_*))] \]

describes how well a method (which is trained using data set \( \mathcal{T} \)) will perform “in production”.

\( E \) is for instance mean squared error (regression) or misclassification (classification).

The overall goal in supervised machine learning is to achieve small \( E_{\text{new}} \).

\[ E_{\text{train}} \triangleq \frac{1}{n} \sum_{i=1}^{n} E(\hat{y}(x_i; \mathcal{T}), y_i) \]

is the training data error.  
Not a good estimate of \( E_{\text{new}} \).
Two methods for estimating $E_{\text{new}}$:

1. **Test data approach**: Randomly split the data into a training set and a test set. Learn the model using the training set. Estimate $E_{\text{new}}$ using the test set.

2. **$c$-fold cross-validation**: Randomly split the data into $c$ parts (or folds) of roughly equal size.
   
   a) The first fold is kept aside as a validation set and the model is learned using only the remaining $c - 1$ folds. $E_{\text{new}}$ is estimated on the validation set.
   
   b) The procedure is repeated $c$ times, each time a different fold is treated as the validation set.
   
   c) The average of all $c$ estimates is taken as the final estimate of $E_{\text{new}}$. 
Summary of Lecture 5 (III/IV)

\[ \tilde{E}_{\text{new}} = \mathbb{E}_* \left[ (g(x_*) - f(x_*))^2 \right] + \mathbb{E}_* \left[ \mathbb{E}_T \left[ (\hat{y}(x_*; T))^2 \right] - (g(x_*))^2 \right] + \sigma^2 \]

- **Bias**: The inability of a method to describe the complicated patterns we would like it to describe.
- **Variance**: How sensitive a method is to the training data.

The more prone a model is to adapt to complicated pattern in the data, the higher the **model complexity** (or model flexibility).
Finding a balanced fit (neither over- nor underfit) is called the **bias-variance tradeoff**.
Contents – Lecture 6

1. Classification and regression trees (CART)
2. Bagging – a general variance reduction technique
3. Random forests
The idea behind tree-based methods

In both regression and classification settings we seek a function \( \hat{y}(x) \) which maps the input \( x \) into a prediction.

One **flexible** way of designing this function is to partition the input space into disjoint regions and fit a simple model in each region.

- **Classification**: Majority vote within the region.
- **Regression**: Mean of training data within the region.
Finding the partition

The key challenge in using this strategy is to find a good partition.

Even if we restrict our attention to seemingly simple regions (e.g. “boxes”), finding an optimal partition w.r.t. minimizing the training error is computationally infeasible!

Instead, we use a “greedy” approach: recursive binary splitting.

1. Select one of the inputs $x_j$ and a cut-point $s$. Partition the input space into two half-spaces,

$$\{ x : x_j < s \} \quad \text{and} \quad \{ x : x_j \geq s \}.$$

2. Repeat this splitting for each region until some stopping criterion is met (e.g., no region contains more than 5 training data points).
Recursive binary splitting

Partitioning of input space

Tree representation
Classification trees

Classification trees are constructed similarly to regression trees, but with two differences.

Firstly, the class prediction for each region is based on the proportion of data points from each class in that region. Let

\[ \hat{\pi}_{mk} = \frac{1}{n_m} \sum_{i: x_i \in R_m} \mathbb{I}\{y_i = k\} \]

be the proportion of training observations in the \( m \)th region that belong to the \( k \)th class. Then we approximate

\[ p(y = k \mid x) \approx \sum_{m=1}^{M} \hat{\pi}_{mk} \mathbb{I}\{x \in R_m\} \]
**Classification trees**

**Secondly,** the squared loss used to construct the tree needs to be replaced by a measure suitable to qualitative outputs.

Three common error measures are,

- **Misclassification error:** \( 1 - \max_k \hat{\pi}_{mk} \)

- **Entropy/deviance:** \(- \sum_{k=1}^{K} \hat{\pi}_{mk} \log \hat{\pi}_{mk}\)

- **Gini index:** \( \sum_{k=1}^{K} \hat{\pi}_{mk}(1 - \hat{\pi}_{mk})\)
Classification error measures

For a binary classification problem ($K = 2$)

- **Misclassification**
- **Gini index**
- **Deviance**

Class-1 probability, $\hat{\pi}_{m1}$

Loss

Fredrik Ljungberg

fredrik.lindsten@it.uu.se
ex) Spam data

Classification tree for spam data:

<table>
<thead>
<tr>
<th>Test error:</th>
<th>Tree</th>
<th>LDA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11.3 %</td>
<td>10.9 %</td>
</tr>
</tbody>
</table>
Improving CART

The flexibility/complexity of classification and regression trees (CART) is decided by the tree depth.

! To obtain a small bias the tree need to be grown deep,
! but this results in a high variance!

The performance of (simple) CARTs is often unsatisfactory!

To improve the practical performance:

▲ **Pruning** – grow a deep tree (small bias) which is then pruned into a smaller one (reduce variance).
▲ **Ensemble methods** – average or combine multiple trees.
  • Bagging and Random Forests (this lecture)
  • Boosted trees (next lecture)
Predicting US Supreme Court behavior

Random forest classifier built on SCDB data\textsuperscript{1} to predict the votes of Supreme Court justices:

\[ Y \in \{ \text{affirm, reverse, other} \} \]

Result: 70\% correct classifications
Not only have random forests proven to be “unreasonably effective” in a wide array of supervised learning contexts, but in our testing, random forests outperformed other common approaches including support vector machines and feedforward artificial neural network models such as multi-layer perceptron.

— Katz, Bommarito II and Blackman (arXiv:1612.03473v2)
Random forests

Bagging can drastically improve the performance of CART!

However, the $B$ bootstrapped dataset are *correlated* ⇒ the variance reduction due to averaging is diminished.

**Idea:** De-correlate the $B$ trees by randomly perturbing each tree.

A *random forest* is constructed by bagging, but for each split in each tree only a *random subset* of $q \leq p$ inputs are considered as splitting variables.

Rule of thumb: $q = \sqrt{p}$ for classification trees and $q = p/3$ for regression trees.\(^2\)

\(^2\)Proposed by Leo Breiman, inventor of random forests.

fredrik.lindsten@it.uu.se
Random forest pseudo-code

**Algorithm** Random forest for regression

1. For $b = 1$ to $B$ (*can run in parallel*)
   
   (a) Draw a bootstrap data set $\tilde{T}$ of size $n$ from $T$.

   (b) Grow a regression tree by repeating the following steps until a minimum node size is reached:
      
      i. Select $q$ out of the $p$ input variables uniformly at random.

      ii. Find the variable $x_j$ among the $q$ selected, and the corresponding split point $s$, that
          minimizes the squared error.

      iii. Split the node into two children with $\{x_j \leq s\}$ and $\{x_j > s\}$.

2. Final model is the average the $B$ ensemble members,

$$\hat{y}_*^{rf} = \frac{1}{B} \sum_{b=1}^{B} \tilde{y}_*^b.$$
Random forests

Recall: For i.d. random variables \( \{Z_i\}_{i=1}^n \)

\[
\text{Var} \left( \frac{1}{n} \sum_{i=1}^n Z_i \right) = \frac{1}{n} \rho \sigma^2 + \rho \sigma^2
\]

The random input selection used in random forests:

▼ increases the bias, but often very slowly
▼ adds to the variance \( (\sigma^2) \) of each tree
▲ reduces the correlation \( (\rho) \) of the trees

The reduction in correlation is typically the dominant effect \( \Rightarrow \) there is an overall reduction in MSE!
ex) Toy regression model

For the toy model previously considered...

![Graph showing test error vs number of trees for Random Forest, Bagging, and Single regression tree methods.](image)
The complexity of a bagging/random forest model increases with an increasing number of trees $B$.

**Will this lead to overfitting as $B$ increases?**

No – more ensemble members *does not* increase the *flexibility* of the model!

**Regression case:**

\[
\hat{y}_{rf}^* = \frac{1}{B} \sum_{b=1}^{B} \tilde{y}_{*,b} \rightarrow \mathbb{E}[\tilde{y}_{*} | T], \quad \text{as } B \to \infty,
\]

where the expectation is w.r.t. the randomness in the data bootstrapping and input selection.
Advantages of random forests

Random forests have several **computational advantages**:

▲ Embarrassingly parallelizable!
▲ Using $q < p$ potential split variables reduces the computational cost of each split.
▲ We *could* bootstrap fewer than $n$, say $\sqrt{n}$, data points when creating $\tilde{T}^b$ — very useful for “big data” problems.

... and they also come with some other benefits:

▲ Often works well off-the-shelf – few tuning parameters
▲ Requires little or no input preparation
▲ Implicit input selection
**ex) Automatic music generation**

**ALYSIA**: automated music generation using random forests.

- User specifies the lyrics
- ALYSIA generates accompanying music via
  - rythm model
  - melody model
- Trained on a corpus of pop songs.

http://www.cs.sjsu.edu/~ackerman/ALYSIA_songs.html

A few concepts to summarize lecture 6

**CART:** Classification and regression trees. A class of nonparametric methods based on partitioning the input space into regions and fitting a simple model for each region.

**Recursive binary splitting:** A greedy method for partitioning the input space into “boxes” aligned with the coordinate axes.

**Gini index and deviance:** Commonly used error measures for constructing classification trees.

**Ensemble methods:** Umbrella term for methods that average or combine multiple models.

**Bagging:** Bootstrap aggregating. An ensemble method based on the statistical bootstrap.

**Random forests:** Bagging of trees, combined with random feature selection for further variance reduction (and computational gains).