An introduction to random forests

Eric Debreuve / Team Morpheme

Institutions: University Nice Sophia Antipolis / CNRS / Inria
Labs: I3S / Inria CRI SA-M / iBV
Outline

• Machine learning
• Decision tree
  • Bagging
  • Random decision trees
• Kernel-Induced Random Forest (KIRF)
• Byproducts
  • Out-of-bag error
  • Variable importance
Machine learning

• Learning/training: build a classification or regression rule from a set of samples

• Prediction: assign a class or value to new samples
(Un)Supervised learning

- **Supervised**
  - Learning set = \{ (sample [acquisition], class [expert]) \}

- **Unsupervised**
  - Learning set = unlabeled samples

- **Semi-supervised**
  - Learning set = some labeled samples + many unlabeled samples
Ensemble learning

• Combining weak classifiers (of the same type)...
  • ... in order to produce a strong classifier
    • Condition: diversity among the weak classifiers

• Example: Boosting
  • Train each new weak classifier focusing on samples misclassified by previous ones

  • Popular implementation: AdaBoost
    • Weak classifiers: only need to be better than random guess
Outline

• Machine learning

▷ • Decision tree
  • Random forest
    • Bagging
    • Random decision trees
  • Kernel-Induced Random Forest (KIRF)

• Byproducts
  • Out-of-bag error
  • Variable importance
Decision tree

- **Root node**
  - Entry point to a collection of data

- **Inner nodes (among which the root node)**
  - A question is asked about data
  - One child node per possible answer

- **Leaf nodes**
  - Correspond to the decision to take (or conclusion to make) if reached

- **Example: CART – Classification and Regression Tree**
  - Labeled sample
    - Vector of variable/feature values + class label
  - Binary decision tree
    - Top–down, greedy building...
    - ... by recursively partitioning the feature space into hyper–rectangles
  - Similarity with weighted kNN

- ** Normally, pruning**
  - To avoid over–fitting of learning data
  - To achieve a trade–off between prediction accuracy and complexity
Decision tree > CART > Building

• All labeled samples initially assigned to root node
• \( N \leftarrow \) root node
• With node \( N \) do
  • Find the feature \( F + \) threshold value \( T \)...
    • ... that split the samples assigned to \( N \) into 2 subsets \( S_{\text{left}} \) and \( S_{\text{right}} \)...
    • ... so as to maximize the label purity within these subsets
  • Assign \((F,T)\) to \( N \)
  • If \( S_{\text{left}} \) and \( S_{\text{right}} \) too small to be splitted
    • Attach child leaf nodes \( L_{\text{left}} \) and \( L_{\text{right}} \) to \( N \)
    • Tag the leaves with the most present label in \( S_{\text{left}} \) and \( S_{\text{right}} \), resp.
  • else
    • Attach child nodes \( N_{\text{left}} \) and \( N_{\text{right}} \) to \( N \)
    • Assign \( S_{\text{left}} \) and \( S_{\text{right}} \) to them, resp.
    • Repeat procedure for \( N = N_{\text{left}} \) and \( N = N_{\text{right}} \)
(Im)Purity
- Quality measure applied to each subset $S_{left}$ and $S_{right}$
- Combination of the measures (e.g., weighted average)

Examples
- Gini index = $\sum_{l=1}^{L} f_l (1 - f_l)$
- Entropy = $- \sum_{l=1}^{L} f_l \log_2 f_l$
- Misclassification error = $1 - \max_{l \in [1..L]} f_l$
### Decision tree > CART > Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>CART</th>
<th>kNN</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intrinsically multiclass</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
<tr>
<td>Handles Apple and Orange features</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Robustness to outliers</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
<tr>
<td>Works w/ &quot;small&quot; learning set</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
</tr>
<tr>
<td>Scalability (large learning set)</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
<tr>
<td>Prediction accuracy</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
</tr>
<tr>
<td>Parameter tuning</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Outline

• Machine learning

• Decision tree

• Random forest
  • Bagging
  • Random decision trees

• Kernel-Induced Random Forest (KIRF)

• Byproducts
  • Out-of-bag error
  • Variable importance
Random forest

• **Definition**
  - Collection of unpruned CARTs
  - Rule to combine individual tree decisions

• **Purpose**
  - Improve prediction accuracy

• **Principle**
  - Encouraging diversity among the tree

• **Solution: randomness**
  - Bagging
  - Random decision trees (rCART)
Random forest > Bagging

• Bagging: Bootstrap aggregation

• Technique of ensemble learning...
  • ... to avoid over-fitting
  • Important since trees are unpruned
  • ... to improve stability and accuracy

• Two steps
  • Bootstrap sample set
  • Aggregation
Random forest > Bagging > Bootstrap

• L: original learning set composed of p samples

• Generate K learning sets $L_k$...
  • ... composed of q samples, $q \leq p$...
  • ... obtained by uniform sampling with replacement from L
  • In consequence, $L_k$ may contain repeated samples

• Random forest: $q = p$
  • Asymptotic proportion of unique samples in $L_k = 100 \times (1 - 1/e) \approx 63\%$
  • → The remaining samples can be used for testing
Random forest > Bagging > Aggregation

- **Learning**
  - For each $L_k$, one classifier $C_k$ (rCART) is learned

- **Prediction**
  - $S$: a new sample
  - Aggregation = majority vote among the $K$ predictions/votes $C_k(S)$
Random forest > Random decision tree

• All labeled samples initially assigned to root node
• \( N \leftarrow \text{root node} \)

• With node \( N \) do
  • Find the feature \( F \) among a random subset of features + threshold value \( T \)...
    • ... that split the samples assigned to \( N \) into 2 subsets \( S_{\text{left}} \) and \( S_{\text{right}} \)...
    • ... so as to maximize the label purity within these subsets
  • Assign \((F,T)\) to \( N \)
  • If \( S_{\text{left}} \) and \( S_{\text{right}} \) too small to be splitted
    • Attach child leaf nodes \( L_{\text{left}} \) and \( L_{\text{right}} \) to \( N \)
    • Tag the leaves with the most present label in \( S_{\text{left}} \) and \( S_{\text{right}} \), resp.
  • else
    • Attach child nodes \( N_{\text{left}} \) and \( N_{\text{right}} \) to \( N \)
    • Assign \( S_{\text{left}} \) and \( S_{\text{right}} \) to them, resp.
    • Repeat procedure for \( N = N_{\text{left}} \) and \( N = N_{\text{right}} \)

• Random subset of features
  • Random drawing repeated at each node
  • For \( D\)-dimensional samples, typical subset size = \( \text{round}(\sqrt{D}) \) (also \( \text{round}(\log_2(x)) \))
  • \( \rightarrow \) Increases diversity among the rCARTs + reduces computational load

• Typical purity: Gini index
Random forest > Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>RF</th>
<th>CART</th>
<th>kNN</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intrinsically multiclass</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Handles Apple and Orange features</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Robustness to outliers</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Works w/ &quot;small&quot; learning set</td>
<td></td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Scalability (large learning set)</td>
<td></td>
<td></td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Prediction accuracy</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Parameter tuning</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
</tbody>
</table>
Random forest > Illustration

1 rCART

10 rCARTs

100 rCARTs

500 rCARTs
Random forest > Limitations

• Oblique/curved frontiers
  • Staircase effect
  • Many pieces of hyperplanes

• Fundamentally discrete
  • Functional data? (Example: curves)
Outline

• Machine learning

• Decision tree

• Random forest
  • Bagging
  • Random decision trees

▷ • Kernel-Induced Random Forest (KIRF)

• Byproducts
  • Out-of-bag error
  • Variable importance
Kernel–Induced Random Forest (KIRF)

• **Random forest**
  - Sample $S$ is a vector
  - Features of $S = \text{components of } S$

• **Kernel–induced features**
  - Learning set $L = \{ S_i, i \in [1..N] \}$
  - Kernel $K(x,y)$
    - Features of sample $S = \{ K_i(S) = K(S_i, S), i \in [1..N] \}$
    - Samples $S$ and $S_i$ can be vectors or functional data
Kernel > Kernel trick

• **Kernel trick**
  • Maps samples into an inner product space...
  • ... usually of higher dimension (possibly infinite)...
  • ... in which classification (or regression) is easier
    • Typically linear

• **Kernel \( K(x, y) \)**
  • Symmetric
  • Positive semi–definite (Mercer's condition):
    \[
    \int \int f(x) K(x, y) f(y) \, dx \, dy \geq 0
    \]
  • \( K(x, y) = \langle \varphi(x), \varphi(y) \rangle \)

  • Note: mapping needs not to be known (might not even have an explicit representation; e.g., Gaussian kernel)
Kernel > Examples

• Polynomial (homogeneous): \( K(x, y) = (x \cdot y)^d \)

• Polynomial (inhomogeneous): \( K(x, y) = (x \cdot y + 1)^d \)

• Hyperbolic tangent: \( K(x, y) = \tanh(\alpha x \cdot y + \beta) \)

• Gaussian: \( K(x, y) = \exp(-\gamma |x - y|^2) \)

  • Function of the distance between samples
  • Straightforward application to functional data of a metric space
    • E.g., curves
KIRF > Illustration

- **Gaussian kernel**
  - Some similarity with vantage-point tree

Reminder: RF w/ 100 rCARTs
KIRF > Limitations

• **Which kernel?**
  - Which kernel parameters?

• **No “orange and apple” handling anymore**
  - \((x \cdot y \text{ or } (x - y)^2)\)

• **Computational load (kernel evaluations)**
  - Especially during learning

• **Needs to store samples**
  - (Instead of feature indices in Random forest)
Outline

• Machine learning

• Decision tree

• Random forest
  • Bagging
  • Random decision trees

• Kernel-Induced Random Forest (KIRF)

• Byproducts
  • Out-of-bag error
  • Variable importance
Byproduct > Reminder

• **To grow one rCART**
  • Bootstrap sample set from learning set L
  • Remaining samples
    • Called out-of-bag samples
    • Can be used for testing

• **Two points of view**
  • For one rCART, out-of-bag samples = L \ Bootstrap samples
    • Used for variable importance
  • For one sample S of L, set of rCARTs for which S was out-of-bag
    • Used for out-of-bag error
• For each sample $S$ of the learning set
  • Look for all the rCARTs for which $S$ was out-of-bag
  • Build the corresponding sub-forest
  • Predict the class of $S$ with it
  • Error = is prediction correct?

• Out-of-bag error = average over all samples of $S$
  • Note: predictions not made using the whole forest...
  • ... but with some aggregation

• Provides an estimation of the generalization error
  • Can be used to decide when to stop adding trees to the forest
Byproduct > Variable importance

• For each rCART
  • Compute out-of-bag error $OOB_{\text{original}}$
    • Fraction of misclassified out-of-bag samples
  • Consider the $i^{th}$ feature/variable of the samples
  • Randomly permute its values among the out-of-bag samples
  • Re-compute out-of-bag error $OOB_{\text{permutation}}$
  • $r$CART-level importance$(i) = OOB_{\text{permutation}} - OOB_{\text{original}}$

• Variable importance$(i) =$ average over all rCARTs
  • Note: rCART-based errors (no aggregation)
    • Avoid attenuation of individual errors
An introduction to random forests

Thank you for your attention