Lecture Pattern Analysis

Part 04: Random Forests and their Variants

Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg

May 6, 2022

Introduction

- We will look at tree-based sample representations
- Compared to our kernels, tree-based methods
  - do not need to store all samples to respond to a query
  - subdivide the sample space dynamically with an objective function
- We start with Classification and Regression Trees (CART)[1], and
  - go over random forests (RFs)[2] to
  - a variant called density forests[3]
- RFs are still actively used, primarily as light-weight and somewhat explainable alternatives to deep neural networks
- We aim to understand CART and RFs as tools for hierarchical sample space partitioning

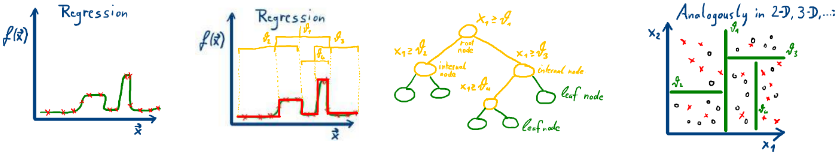
[1] see Hastie/Tibshirani/Friedman Sec. 9.2

[2] see Hastie/Tibshirani/Friedman Sec. 15-15.3.4 and 15.4.3

[3] see paper in studOn by Criminisi/Shotton/Konukoglu

Classification and Regression Trees (CART)

- The idea of CARTs is to
  - recursively subdivide the sample space, and to
  - perform classification or regression locally in each partition
- This subdivision is naturally represented as a binary tree
  - Each internal node encodes a split of the feature domain
  - Left and right successor nodes operate on the “left” and “right” part of the split
  - If the split is sufficiently simple (e.g., a threshold along one of the dimensions), define “left” as below the threshold, and “right” as above
- The leaf nodes perform the actual task on the samples
  - For regression, return the average of the node samples y_i
  - For classification, return the relative frequencies per class



CART Training

- Each node has 2 parameters: one axis-aligned split along one dimension
- The tree has a maximum height, which is a hyper-parameter
- Training steps:
  1. The tree is grown greedily from root to leaves
     - 1.1 At each node, perform an exhaustive search for the best split w.r.t. an objective function $Q_m(T)$ (→ best dimension, best position)
     - 1.2 Stop splitting a node if only few samples are left in this branch
  2. Prune the tree by merging those splits with least cost increase.
     Pruning cost function is tradeoff between model accuracy and model complexity

$$C_\alpha(T) = \sum_{m=1}^{|T|} Q_m(T) + \alpha |T| \ , \tag{1}$$

where T is a subtree, m the index of a region, $|T|$ the number of terminal nodes of T, and α an additional hyperparameter

Objective Functions for CART Training

- Regression:

$$Q_m(T) = \sum_{\mathbf{x}_i \in \mathcal{R}_m} (y_i - \bar{y}_m)^2 \ , \tag{2}$$

where $\mathcal{R}_m$ are the samples in region m, y_i the sample label, and $\bar{y}_m$ the average of all samples in $\mathcal{R}_m$.

- Classification: Negative cross-entropy

$$Q_m(T) = \sum_{k}^{K} p_{mk} \cdot \ln p_{mk} \ , \tag{3}$$

with K as number of classes, and p_{mk} the relative frequency of class k in $\mathcal{R}_m$.

- Classification: Gini index

$$Q_m(T) = \sum_{k}^{K} p_{mk} \cdot (1 - p_{mk}) \tag{4}$$

Decision for the Best Splitting Candidate

- Overall objective: search candidate split with minimum loss

$$\min \frac{|s_l|}{N} Q_l(T) + \frac{|s_r|}{N} Q_r(T) \ , \tag{5}$$

  with l and r as left and right parts of the candidate split, $s_{l/r} = \{\mathbf{x}_i \in \mathcal{R}_{l/r}\}$ as associated samples, and N as total number of samples
- Cross-entropy and Gini index outperform misclassification rate, and are differentiable:

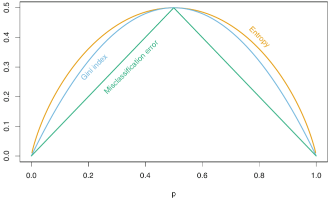


FIGURE 9.3. *Node impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through* $(0.5, 0.5)$.

Prediction in a Leaf Node

- For classification:
  For each class, the prediction is just the relative frequency of samples in the leaf node
- For regression:
  The prediction is just the average of the data points in the class
  (if needed, a more sophisticated interpolation is also possible)

Random Forests (RF)

- CART has a tendency to overfit:
  Trees with high training accuracy oftentimes subdivide the space into too small regions
- Random forests form an ensemble of randomized CARTs to counter overfitting:
  Each individual tree is a “weak learner”, the ensemble is strong
- The ensemble just averages votes from individual trees
- Randomization components:
  - Bagging: train each tree on a random subset of the training data
  - Random subspace projections: per node, search only in *d* random dimensions for best split
  - Also popular is a further randomization option:
    Replace exhaustive search for best split by selecting best split from randomly drawn split candidates (weird class name in sklearn: `ExtraTreesClassifier`)

Rationale behind Randomization

- Randomization adds statistical independence and decorrelates trees:
  - Averaging the votes of N **statistically independent and identically distributed (i.i.d.)** learners with accuracy $> 50\%$ arbitrarily reduces the variance:

$$\sigma^2_{\text{i.i.d. ensemble}} = \frac{1}{N}\sigma^2 \tag{6}$$

  - Note, however, that the i.i.d. assumption is overly optimistic: For example bagging introduces only **identically distributed** variables with correlation ρ, which "only" reduces the variance to

$$\sigma^2_{\text{i.d. ensemble}} = \rho\sigma^2 + \frac{1-\rho}{N}\sigma^2 \quad , \tag{7}$$

    hence the variance is at least $\rho\sigma^2$.
  - The remaining randomization steps aim to reduce the correlation ρ
  - The configuration of the randomization are additional hyperparameters (bagging percentage, subspaces dimension, number of candidate splits)

Engineering Remarks

- Bagging enables cross-validation without a separate validation set: Use **Out-of-bag samples**, i.e., samples not part of the bag of each tree, for validation
- More trees improve the variance, i.e., counter overfitting
- Deeper trees tend to overfit, hence set a depth limit (but too shallow trees increase the bias)
- Parallelization is easy by assigning individual trees to separate compute nodes
- Random Forests can be directly used for **multi-class classification**, which is impossible with many other classifiers such as SVMs
- Decision boundaries of extremely randomized forests are very smooth, which oftentimes improves robustness
- Trees and Forests are oftentimes appraised for **explainable decisions**, by searching for the most important splitting decisions

Random Forests as Adaptive Nearest Neighbors

- RF space partitioning can be seen as a data-driven (adaptive) variant to the nearest neighbor (NN) mechanism
- Figure below: NN classifier, i.e., majority vote among *k* nearest neighbors

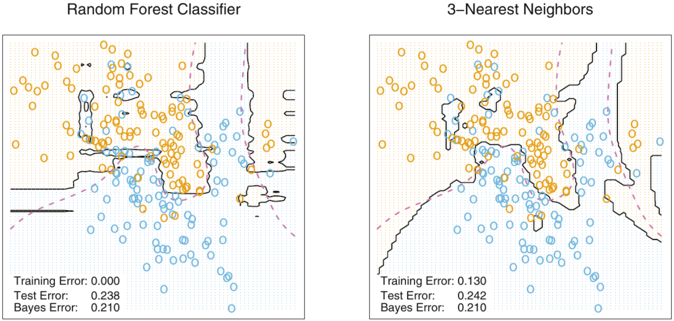


FIGURE 15.11. *Random forests versus 3-NN on the mixture data. The axis-oriented nature of the individual trees in a random forest lead to decision regions with an axis-oriented flavor.*

Density Forests

- We can exchange the **supervised** objective functions with an **unsupervised** function to perform density estimation[4]
- An unsupervised loss can only use the structure of the data
- Criminisi/Shotton/Konukoglu propose to prefer splits that lead to compact Gaussian distributions:
  - Compactness is measured with the **determinant of the Gaussian covariance** that is fitted to the samples in the left and right branch of a candidate split
  - Re-define the entropy in Eqn. 3 as

$$Q_m(T) = \frac{1}{2} \log \left((2\pi e)^D |\Lambda(\mathbf{x}_i \in \mathcal{R}_m)| \right) \tag{8}$$

  where $1/2$ and $(2\pi e)^D$ are constants from the Gaussian (can be ignored), and Λ is the covariance of the samples in $\mathcal{R}_m$.
  - Geometrically, the determinant approximates a rectangular volume around the majority of samples

[4] This is not covered in Hastie/Tibshirani/Friedman, please refer to the paper by Criminisi/Shotton/Konukoglu in studOn, Sec. 5

Full Density, and Sampling from the Density

- A single tree models the density as multiple truncated Gaussians:

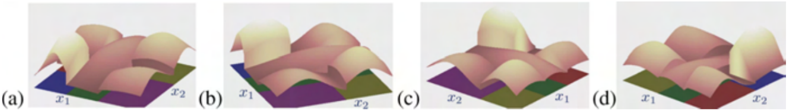


Fig. 5.2 A tree density is piece-wise Gaussian. (a,b,c,d) Different views of a tree density $p_t(\mathbf{v})$ defined over an illustrative 2D feature space. Each individual Gaussian component is defined over a bounded domain. See text for details.

- Again, averaging over the forest smooths out the sharp boundaries
- However, this is not necessary to sample from this density:
  1. Randomly select a tree
  2. Randomly move from root to leaf, descending to the left with probability $\frac{|s_l|}{|s_l|+|s_r|}$ where again $s_{l/r} = \{\mathbf{x}_i \in \mathcal{R}_{l/r}\}$
  3. Randomly sample from the covariance in the leaf
  4. Repeat step 3 until the drawn sample is inside of the leaf region