Structured Random Forests

Ben Haghi Cody Han Yury Tokpanov

Department of Computing and Mathematical Sciences
California Institute of Technology

CS159, Spring 2017

Outline

Prototypical Types of Tasks

Problems related to automatic analysis of complex data such as text, photographs, videos, and n-dimensional medical images can be categorized into these prototypical tasks:

- **Classification.** Recognizing type or category of a scene captured in a photograph, output is a discrete, categorical label.
- **Regression.** Predicting the price of a house as a function of its distance from a good school; output is a continuous variable.
- **Density estimation.** Learning a probability density function for healthy individual scans to detect abnormalities.
- **Manifold learning.** Capturing intrinsic variability of size and shape of different structures in human brain from MRIs.

Prototypical Types of Tasks

- **Semi-supervised.** Interactive image segmentation, for example, where user's brush strokes define labeled data and other pixels provide already available unlabeled data.
- **Active learning.** Learning a general rule for detecting tumors using only a few manual annotations, input mostly unlabeled.

What follows is a unified model of decision forests that can be used in all of these prototypical learning tasks. Can implement and optimize inference algorithms once and use them in many applications.

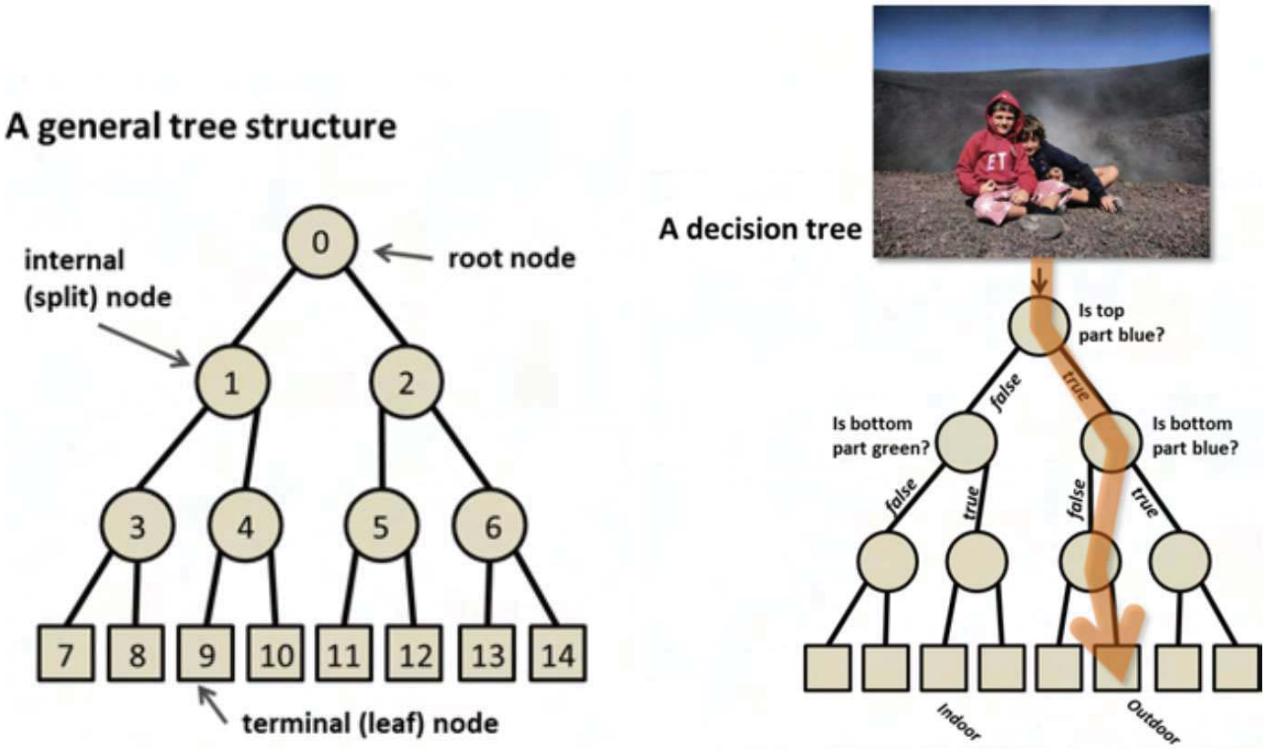
Review: Decision Tree Basics

A decision tree is a set of questions organized in a hierarchical manner and represented graphically as a tree.

- To estimate an unknown property of an input object, the tree asks successive questions about its known properties. Each question depends on the answers to the previous questions — represented as a path through the tree. The terminal node on the path determines the decision.
- Requires (1) tests associated to internal nodes and (2) decision-making predictors associated with each leaf.

In general, decision trees can be compared to splitting complex problems into sets of simpler ones with a hierarchical model. Model parameters can be selected by hand (simple) or learned from data.

Review: Decision Tree Basics



Left: general decision tree structure. Right: a decision tree for determining whether a photo was taken indoors or outdoors.

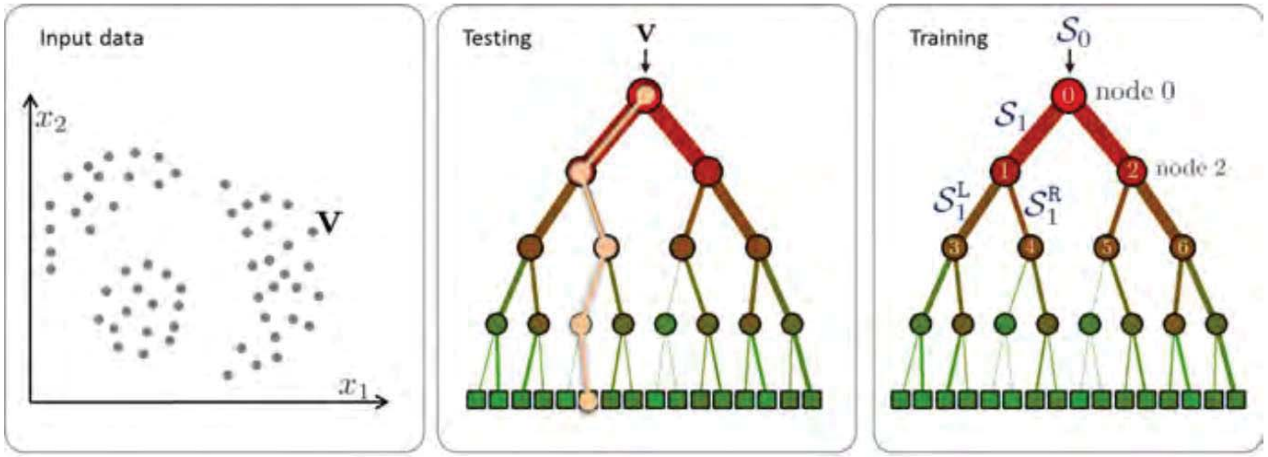
Notation

Definition (Data points and features)

A generic object, called a *data point*, is denoted by a vector $\mathbf{v} = (x_1, x_2, \ldots, x_d) \in \mathcal{F}$ where the components x_i represent some attributes of the data point called *features*.

Features vary between applications; for computer vision $\mathbf{v}$ may correspond to a pixel in an image and x_i may represent responses of a chosen filter bank at that location.

Notation



Notation example. Input is collected in d-dimensional space. Testing is done by each node on $\mathbf{v}$, which is sent to child node. Training involves sending all training data $\mathcal{S}_0$ into the tree and optimizing the split node parameters over an energy function. (Criminsi 2012)

Notation

- Number of features depends on the type of data point and application.
- Dimensionality of feature space $\mathcal{F} = d$ can be large, even infinite, so we extract a small portion of d features as we need them.

Definition (Features of interest)

Define $\phi(\mathbf{v}) = \left(x_{\phi_1}, x_{\phi_2}, \ldots, x_{\phi_{d'}}\right) \in \mathcal{F}^{d'} \subset \mathcal{F}$ as the selected subset of features where d' denotes the dimensionality of the subspace and $\phi_i \in [1, d]$ denote the selected dimensions.

We usually choose a subspace $\mathcal{F}^{d'}$ that is much smaller than the original ($d' \ll d$).

Notation

Test functions, split functions, weak learners

We use the terms "test function," "split function," and "weak learner" interchangeably.

Definition (Test function)

A test function at split node j is a function

$$h(\mathbf{v}, \theta_j) : \mathcal{F} \times \mathcal{T} \rightarrow \{0, 1\},$$

where 0 and 1 can be interpreted as false or true respectively, and $\theta_j \in \mathcal{T}$ denote the parameters of the test function at the jth split node.

A data point $\mathbf{v}$ arriving at j is sent to its left or right child according to the result of j's test function.

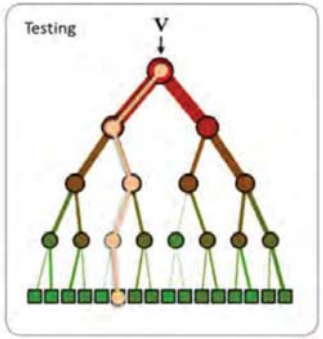
Notation

Training points and training sets

- A *training point* is a data point for which the attributes that we are seeking for are actually known. For example, a set of photos with "indoor" or "outdoor" labels.
- A *training set* $\mathcal{S}_0$ is a collection of different training points.
- In supervised tasks, a training point is a pair $(\mathbf{v}, \mathbf{y})$ where $\mathbf{v}$ is the input feature vector and $\mathbf{y}$ represents a generic, known label.
- In unsupervised tasks, a training point is represnted by its feature response and it has no associated label.

Decision tree testing

Decision trees are separated into an offline training phase and an online testing phase.



(Criminsi 2012)

Testing

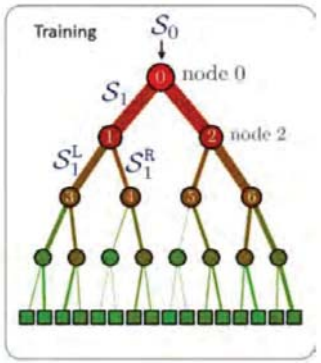
A new data point **v** starts at the root and makes it way down to a leaf node, which contains uses a predictor to construct an output (label or continuous value) for **v**.

Decision tree training

Training

Selects the type and parameters of the test function $h(\mathbf{v}, \theta_j)$ associated with each split node j by optimizing a chosen objective function defined on the training set.

- At each node j with input $\mathcal{S}_j$, learn the function that "best" splits S_j into S_j^R and S_j^L.
- Do this by maximizing an objective function.



(Criminsi 2012)

Training objective function

The objective function at node j, in general, is defined as

$$\theta_j^* = \arg\max_{\theta_j \in \mathcal{T}} I_j \tag{1}$$

where

$$I_j = I(\mathcal{S}_j, \mathcal{S}_j^L, \mathcal{S}_j^R, \theta_j)$$
$$S_j^L = \{(\mathbf{v}, \mathbf{y}) \in \mathcal{S}_j \mid h(\mathbf{v}, \theta) = 0\}$$
$$S_j^R = \{(\mathbf{v}, \mathbf{y}) \in S_j \mid h(\mathbf{v}, \theta_j) = 1\}$$

I_j captures some notion of information gain from the split while the second two equations state that data points on which h evaluates to 0 go left and to 1 go right.

Stopping criteria for training

We need some way to know when to stop the training algorithm.

- Can stop when the tree has some maximum number of levels D.
- Can also stop when we reach some minimum value of $\max_{\theta_j} I_j$, which is when the attributes that we care about in the leaf nodes are similar to one another.
- Can stop growing tree when a node contains too few training points.

Stopping criteria prevent overfitting and loss of generalization power.

Weak Learner Models

Consider a geometric parameterization for split functions, with a weak learner model formulated as $\theta = (\phi, \psi, \tau)$ where:

- ψ defines the geometric primitive used to separate the data (e.g. axis-aligned hyperplane, oblique hyperplane, general surface).
- τ is a parameter vector that captures thresholds for inequalities used in the binary test.
- ϕ is a filter function that selects features from input vector $\mathbf{v}$.

We perform the optimization from equation (1) over these subparameters.

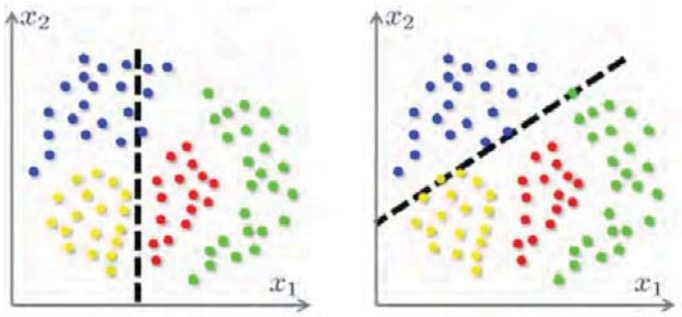
Linear and Nonlinear Data Separation

A simple parameterization is the linear model

$$h(\mathbf{v}, \theta_j) = [\tau_1 > \phi(\mathbf{v}) \cdot \psi > \tau_2] \tag{2}$$

where $[\cdot]$ is the indicator function.

- In 2D, $\phi(\mathbf{v}) = (x_1, x_2, 1)^T$ and $\psi \in \mathbf{R}^3$ denote a generic line in homogenous coordinates.
- Setting $\tau_1 = \infty$ or $\tau_2 = -\infty$ corresponds to using a single inequality test function.
- Lines ψ aligned with an axes of the feature space (e.g. $\psi = (1, 0, \psi_3)$ or $\psi = (0, 1, \psi_3)$) are often used in boosting and are referred to as *stumps*.



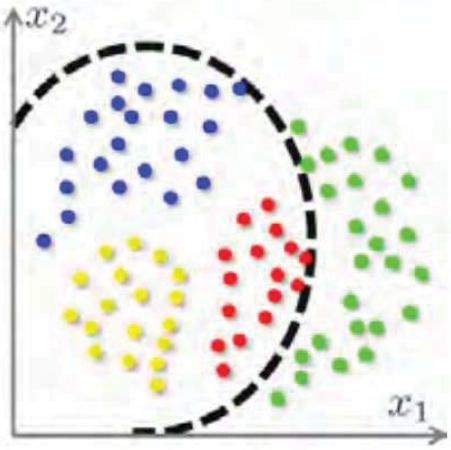
Axis aligned vs general line (Criminsi 2012)

Linear and Nonlinear Data Separation

We can replace hyperplanes with surfaces that have more degrees of freedom. In 2D we can use conic sections with

$$h(\mathbf{v}, \theta_j) = \left[\tau_1 > \phi^T(\mathbf{v})\ \psi\ \phi(\mathbf{v}) > \tau_2\right] \quad (3)$$

with $\psi \in \mathbf{R}^{3\times 3}$ representing the conic section in homogenous coordinates.



Quadratic separation

Remark

Low dimensional weak learners can be used for high dimensional data because the selector ϕ_j can select a small set of features for different nodes.

Energy Models (discrete)

Entropy

By training objective functions, we reduce uncertainty using the concepts of *entropy* and *information gain*. We can quantify the intuition of uncertainty reduction with measures for entropy and information gain.

Shannon entropy

For discrete probability distributions we typically use Shannon entropy.

$$H(\mathcal{S}) = -\sum_{c \in \mathcal{C}} p(c) \log(p(c)) \tag{4}$$

where $\mathcal{S}$ is the set of training points and c indicates the class label. $\mathcal{C}$ denotes the set of all classes and $p(c)$ is the empirical distribution extracted from the training points in $\mathcal{S}$.

Energy Models (discrete)

Information gain

Improvements from decision tree splits can be quantified by measuring information gain

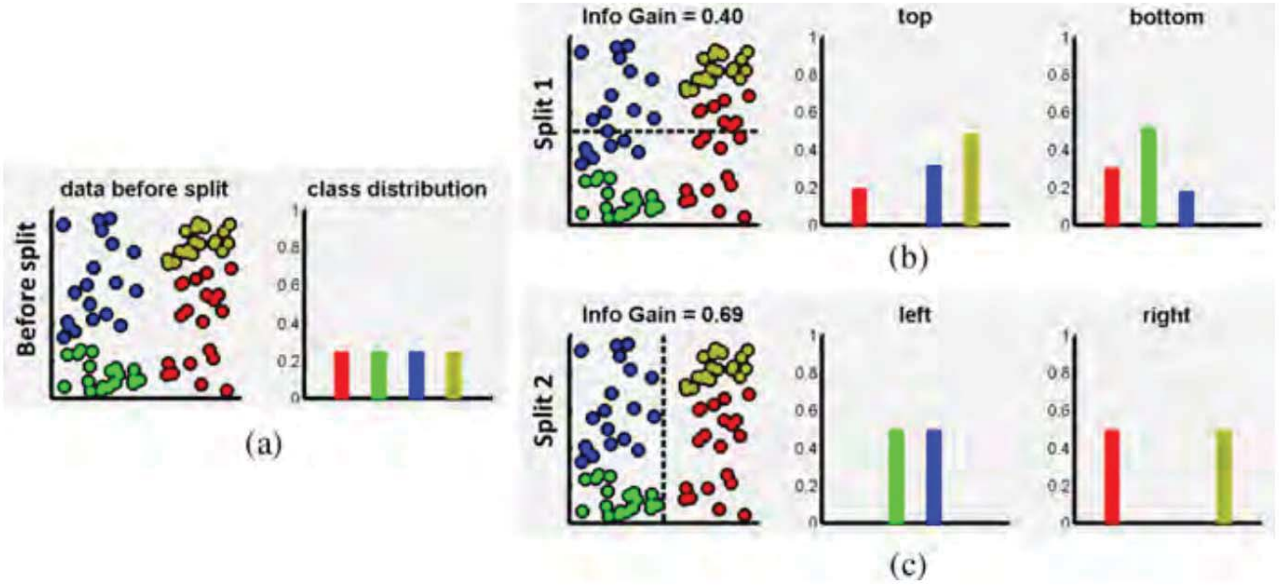
$$I = H(\mathcal{S}) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i) \tag{5}$$

Remark

Maximizing information gain gives us split parameters which produce the highest confidence in the final distributions.

Energy Models (discrete)

Example



Information gain from axis aligned discrete splits (Criminsi 2012)

Energy Models (continuous)

Entropy

We can also define entropy and information gain for continuous labels and distributions.

Differential entropy

In place of the Shannon entropy, we use differential entropy

$$H(\mathcal{S}) = -\int_{y \in \mathcal{Y}} p(y) \log(p(y))\, dy \tag{6}$$

where y is a continuous label and p is the probability density functino estimated from the training points in $\mathcal{S}$.

Energy Models (continuous)

Gaussian-based models are often used to approximate $p(y)$ because of their simplicity. The differential entropy of a d-variate Gaussian is

$$H(\mathcal{S}) = \frac{1}{2} \log \left[(2\pi e)^d \left| \Lambda(\mathcal{S}) \right| \right] \tag{7}$$

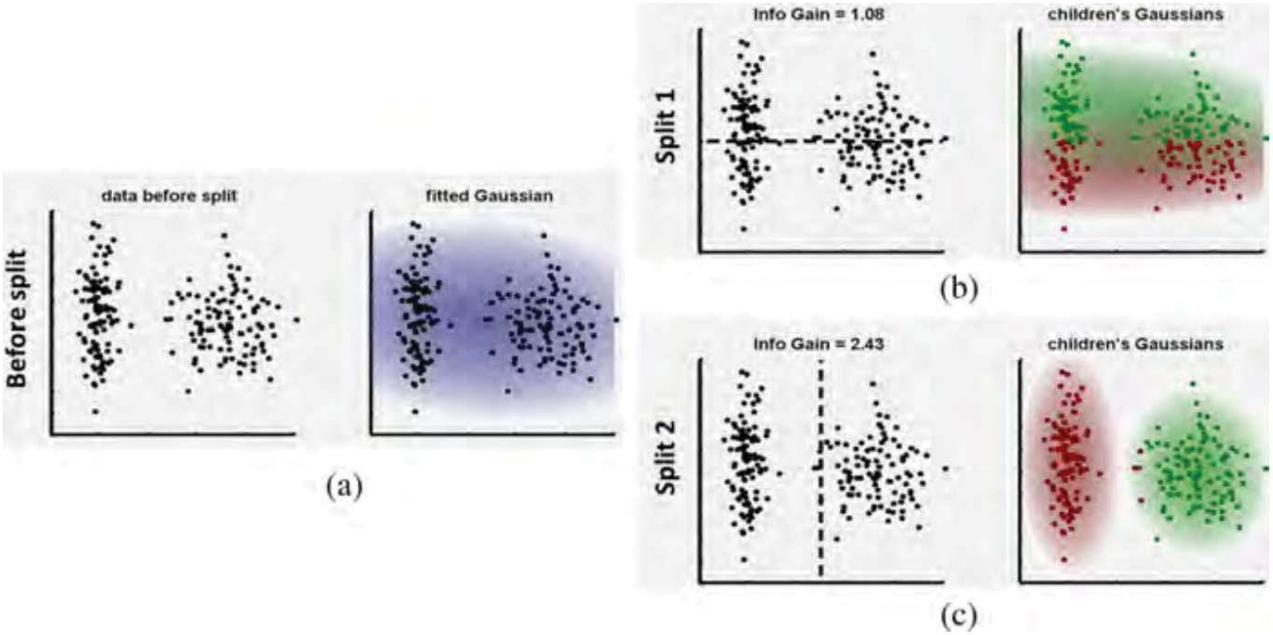
where $\Lambda(\mathcal{S})$ is the covariance matrix of the training set.

Remark

Information gain (5) *can be defined the same way in continuous models as discrete models because of the flexibility of its definition.*

Energy Models (continuous)

Example



Information gain from Gaussian model (Criminsi 2012)

Leaf Prediction Models

After training, each leaf is associated with a subset of the labeled training data.

- A test point traverses the tree until it reaches a leaf, and we generate a label for that point using the test statistics gathered in that leaf.
- In general, leaf statistics can be captured with posterior distributions

$$p(c \mid \mathbf{v}) \quad \text{and} \quad p(y \mid \mathbf{v}) \tag{8}$$

where c and y represent categorical (discrete) or continuous labels and $\mathbf{v}$ is the data point being tested. Conditioning denotes the fact that the distributions depend on the specific leaf note reached by $\mathbf{v}$.

Leaf Prediction Models

MAP predictor

We can do a maximum a posteriori (MAP) estimate with

$$c^* = \arg\max_c p(c \mid \mathbf{v}) \tag{9}$$

in the discrete case. In general, however, keeping the entire distribution around lets us reason about uncertainties better.

Randomness Model

During training, we inject randomness into trees in order to give the trained trees better generalization power. Two popular ways to do this are:

1. Random training set sampling (bagging). This typically yields good training efficiency.
2. Randomized node optimization. This enables us to train trees on the entire training data and yields margin-maximization properties for ensemble models.

These two methods can be used together.

Randomness Model

Randomized node optimization

In the optimization equation (1), we optimize with respect to the entire parameter space $\mathcal{T}$. For large dimensional problems, it can be infeasible to optimize over $\mathcal{T}$.

Idea

At node j, use a small random subset $\mathcal{T}_j \subset \mathcal{T}$ of parameter values, optimizing

$$\theta_j^* = \arg\max_{\theta_j \in \mathcal{T}_j} I_j \tag{10}$$

The amount of randomness is controlled by $|\mathcal{T}_j|/|\mathcal{T}|$. We can define $\rho = |\mathcal{T}_j|$ so that when $\rho = |\mathcal{T}|$ there is no randomness and when $\rho = 1$ we have maximum randomness and no optimization.

Random Decision Forests

A random decision forest is an emsemble of randomly trained decision trees.

Key aspect

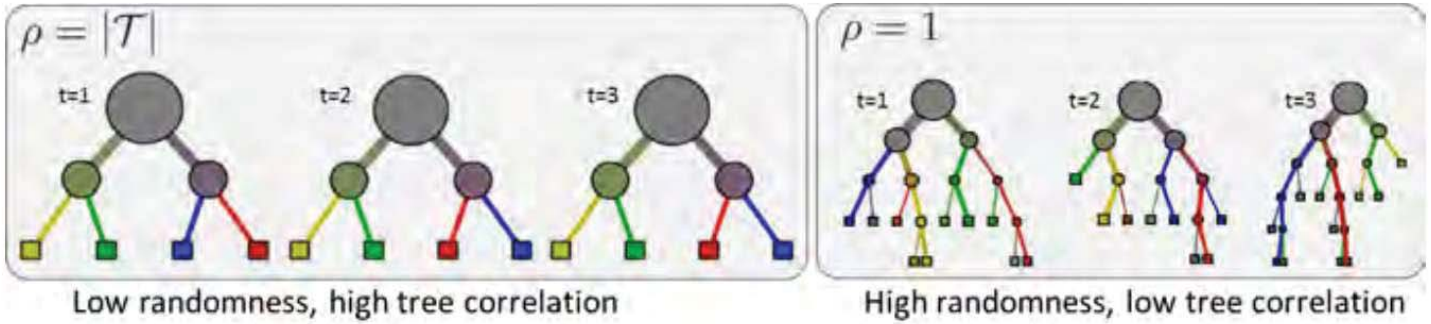
The component trees in a random forest model are all randomly different from one another, which decorrelates individual tree predictions and results in improved generalization and robustness.

Forest properties

The weak learners, energy model, leaf predictors, and types of randomness in its component trees influence the prediction and estimation properties in a forest.

Random Decision Forests

The randomness parameter $\rho = |\mathcal{T}_j|$ controls the correlation between trees in a forest.



Randomness control in forest (Criminsi 2012)

Random Decision Forests
Training and prediction

Notation

In a forest with T trees, we use $t \in 1, \ldots, T$ to index component trees.

- Component trees are trained individually — this can be done in parallel.
- A test point $\mathbf{v}$ is simultaneously passed through all trees until it reaches the leaves — can also be done in parallel.

Random Decision Forests
Training and prediction

We can combine all tree predictions into a forest prediction in multiple ways. For classification, we can use a simple average

$$p(c \mid \mathbf{v}) = \frac{1}{T} \sum_{t=1}^{T} p_t(c \mid \mathbf{v}) \tag{11}$$

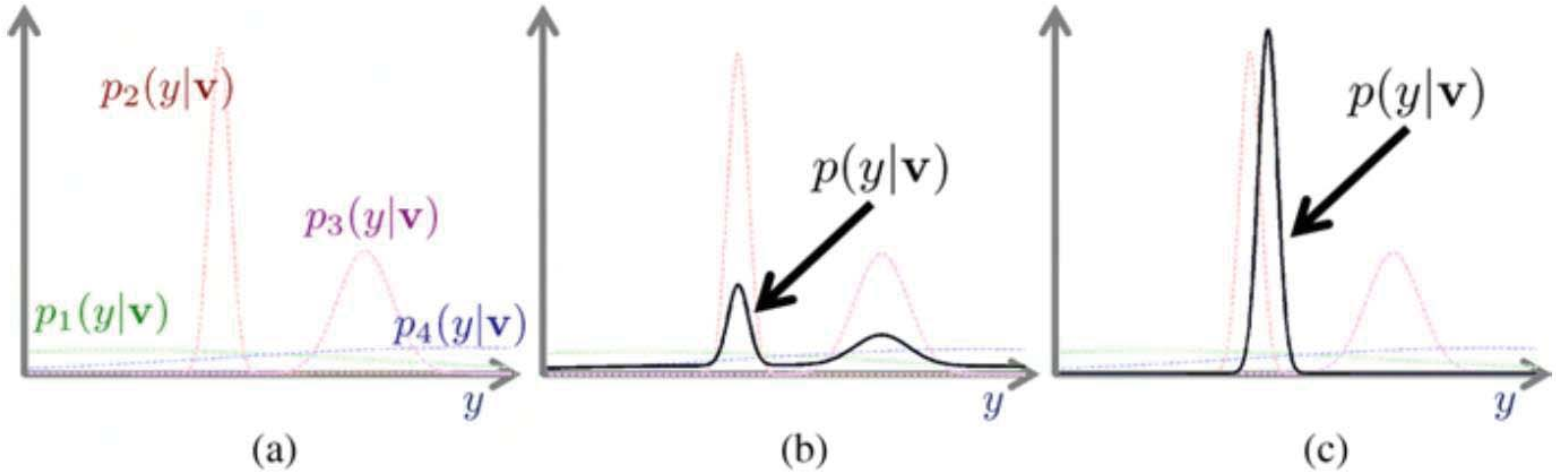
where $p_t(c \mid \mathbf{v})$ represents the posterior distribution found in the leaf of the tth tree. We can also multiply tree outputs together

$$p(c \mid \mathbf{v}) = \frac{1}{Z} \prod_{t=1}^{T} p_t(c \mid \mathbf{v}) \tag{12}$$

where the partition function Z ensures probabilistic normalization.

Random Decision Forests

Example



Forest ensemble model (Criminsi 2012)

Key Parameters

The parameter models that most influence decision forest behavior:

- Maximum allowed tree depth D. Very deep trees can lead to overfitting.
- Amount of randomness (controlled by ρ) and its type. Randomness affects tree correlation and generalization properties.
- Forest size (number of trees) T. Testing accuracy generally increases as T increases, but training time also increases.
- Choice of weak learner model
- Training objective function
- Choice of features in practical applications

Random Forest Specializations

The generic decision forest model we just defined can be applied to various specific tasks:

- Classification forests
- Regression forests
- Density estimation forests
- Manifold forests
- Semi-supervised forests
- Random ferns and other variants

Each of the above models can be defined with slight modifications and specifications on top of the random forest framework. If interested, see the original paper in the references.

For Further Reading I

A. Criminsi, J. Shotton, E. Konukoglu.
Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning.
Foundations and Trends in Computer Graphics and Vision, vol. 7, no. 2-3, pp. 81-227, 2012.