1 Decision Forests

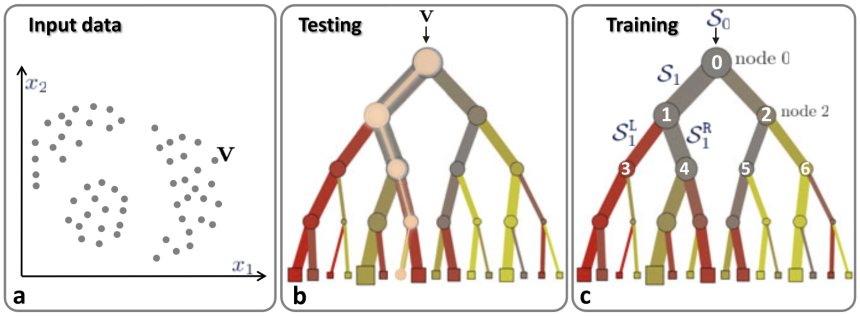
From *Decision Forests for Computer Vision and Medical Image Analysis*[1]

1.1 The Decision Forest Model (chapter 3)

Popularity of decision forests is mostly due to success in classification tasks; however, forests can be used for classification, regression, density estimation, manifold learning, semi-supervised learning, and active learning. Generalization can be achieved by ensambles of slightly different trees.

1.1.1 Data Structure & Notation: Decision Trees

Nodes are **internal** (circles) or **terminal** (squares). Each internal node stores a test function to be applied to incoming data and each leaf stores the final answer/predictor



- Input data are represented as a collection of points in the d-dimensional space defined by their feature responses
- During testing, an internal node applies a test to the input data $\mathbf{v}$ and outputs to the left or right child
- This process is repeated until a leaf node is reached
- Training a decision tree involves sending the entire training set S_0 into the tree and optimizing the parameters of the internal nodes.
- to function well, must establish
  1. the tests at each internal node
  2. decision making predictors at each leaf

1.2 Mathematical Notation

1.2.1 Data Points

- A data point is denoted by a vector
$$\mathbf{v} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$$
  where x_i represents a feature.
- In many cases the dimensionality of the feature space d is very large and we are instead interested in a only a subset of features:
$$\boldsymbol{\phi}(\mathbf{v}) = (x_{\phi_1}, x_{\phi_2}, \dots, x_{\phi_{d'}}) \in \mathbb{R}^{d'}$$
- d' is the dimensionality of the subspace and $\phi_1 \in [1, d]$ is the selected dimensions. In general,
$$d' \ll d$$

[1] https://link.springer.com/book/10.1007/978-1-4471-4929-3

1.2.2 Test/Split Functions

Each internal node j has a different associated test/split function with binary outputs:

$$h(\mathbf{v}, \boldsymbol{\theta}_{\mathbf{j}}) : \mathbb{R}^d \times \mathcal{T} \to \{0, 1\}$$

- 0=False, 1=True
- $\mathcal{T}$ is the space of all split parameters ($\boldsymbol{\theta}_j \in \mathcal{T}$)

1.2.3 Training Points/Sets

- In supervised learning a **training point** is a pair $(\mathbf{v}, \mathbf{y})$ where $\mathbf{v}$ is input data point and $\mathbf{y}$ is a label
- A **training set** $\mathcal{S}_0$ is a collection of different training points
- nodes are ordered in breadth-first oder starting from the root (0)
- $\mathcal{S}_1^L$ is the subset to the left of node 1. similarly, $\mathcal{S}_1^R$ is the subset to the left of node 1
- properties for each split node j:

$$\mathcal{S}_j = \mathcal{S}_j^L \cup \mathcal{S}_j^R \qquad \mathcal{S}_j^L \cap \mathcal{S}_j^R = \emptyset \qquad \mathcal{S}_j^L = \mathcal{S}_{2j+1} \qquad \mathcal{S}_j^R = \mathcal{S}_{2j+2}$$

1.3 Randomly Trained Decision Trees

1.3.1 Tree Testing (on-line) vs Tree training (off-line)

- **testing:** starting at the root, each test node applies a test function $h(\cdot, \cdot)$ to $\mathbf{v}$, the data point $\mathbf{v}$ is sent to either the left or right child and this process is repeated until a leaf node (predictor/estimator) is reached.
- **training:** The training phase takes care of selecting the type and parameters of the test function $h(\mathbf{v}, \boldsymbol{\theta})$ associated with each test node by optimizing a chosen objective function
  - At each node, j, we learn the function that best splits $\mathcal{S}_j$ into $\mathcal{S}_j^R$ and $\mathcal{S}_j^L$. Objective function:

    $$\boldsymbol{\theta} = \underset{\boldsymbol{\theta} \in \mathcal{T}}{\arg\max}\, I(\mathcal{S}_j, \boldsymbol{\theta})$$

  - Given the set $\mathcal{S}_j$ and the split parameters $\boldsymbol{\theta}$, the left and right sets are uniquely determind:

    $$\mathcal{S}_j^L(\mathcal{S}_j, \boldsymbol{\theta}) = \{(\mathbf{v}, \cdot) \in \mathcal{S}_j | h(\mathbf{v}, \boldsymbol{\theta}) = 0\}$$

    $$\mathcal{S}_j^R(\mathcal{S}_j, \boldsymbol{\theta}) = \{(\mathbf{v}, \cdot) \in \mathcal{S}_j | h(\mathbf{v}, \boldsymbol{\theta}) = 1\}$$

    with $\cdot$ denoting $\mathbf{y}$ for continuous lablels used in regression or discrete c in classification
  - since $\mathcal{S}_j^L, \mathcal{S}_j^R$ are functions of $\mathcal{S}_j$ the objective function $I(\mathcal{S}_j, \boldsymbol{\theta})$ takes as input the parent set and the splitting parameters.
  - The tree structuree depends on how and when we decide to stop growing branches. There are many ways including stopping when a max number of levels D has been reached
  - At the end of training we have :
    1. the optimum split functions associated with each node
    2. a learned tree structure
    3. a different set of training points at each leaf

1.3.2 Weak Learner Models

- **Parameters** of the weak learner model: $\boldsymbol{\theta} = (\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\tau})$
  - $\boldsymbol{\phi}(\mathbf{v})$ selects features of choice out of the vector $\mathbf{v}$
  - $\boldsymbol{\psi}$ defines the geometric primitive to separate the data (hyperplane, general surface, etc)
  - $\boldsymbol{\tau}$ captures thresholds for the inequalities used in the binary test
- **Linear data separation:**

  $$h(\mathbf{v}, \boldsymbol{\theta}) = [\tau_1 > \boldsymbol{\phi}(\boldsymbol{v}) \cdot \boldsymbol{\psi} > \tau_2]$$

  where $[\cdot]$ is the indicator function (1=true, 0=false)
- **nonlinear data separation:** more complex weak learners replace hyperplanes with higher degree of freedom surfaces. for example, in 2d one could use conic sections:

  $$h(\mathbf{v}, \boldsymbol{\theta}) = [\tau_1 > \boldsymbol{\phi}^{\mathrm{T}}(\mathbf{v})\boldsymbol{\psi}\boldsymbol{\phi}(\mathbf{v}) > \tau_2]$$

  with $\boldsymbol{\psi} \in \mathbb{R}^{3\times 3}$ a matrix representing the conic section

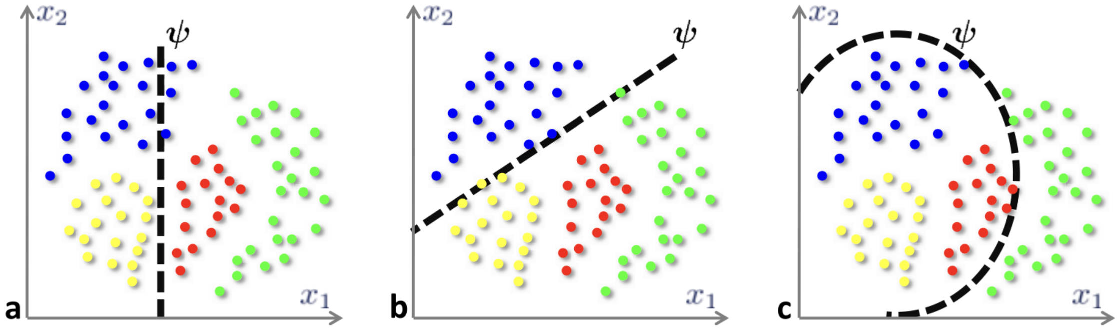


Fig. 3.4 Example weak learners. In this illustration the colors attached to each data point (*circles*) indicate different classes. (**a**) Axis-aligned hyperplane weak learner. (**b**) General oriented hyperplane. (**c**) Quadratic surface (conic in 2D). For ease of visualization here we have $\mathbf{v} = (x_1\ x_2)^\top \in \mathbb{R}^2$ and $\boldsymbol{\phi}(\mathbf{v}) = (x_1\ x_2\ 1)^\top$ in homogeneous coordinates. In general, a data point $\mathbf{v}$ may have a much higher dimensionality and $\boldsymbol{\phi}(\mathbf{v})$ still a dimensionality ≤ 2

1.3.3 Energy Models

- Information gain associated with a tree split node is defined as the reduction in uncertainty achieved y splitting the training data arriving at the node into multiple child subsets

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

  where H is entropy: a measure of the uncertainty associated with the random variable we wish to predict and $|\cdot|$ indicates weighting the entropy by the cardinality of the child sets to avoid splitting off children containing few points

1.3.4 Leaf Prediction Models

In the most general sense leaf stats can be captured using the conditional distributions

$$p(c|\mathbf{v})$$

for categorical labels or

$$p(\mathbf{y}|\mathbf{v})$$

for continuous labels.

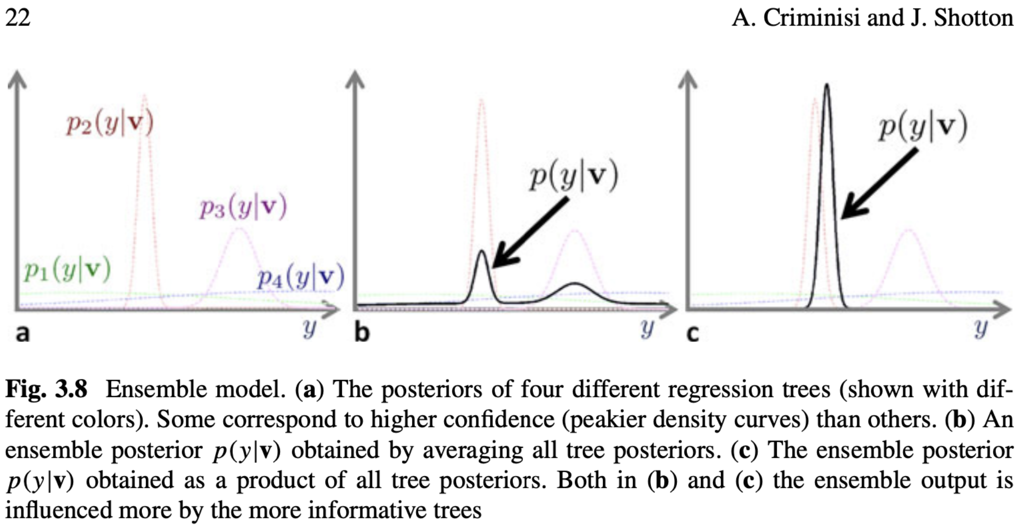
1.3.5 The Randomness Model

- Randomness is injected during the training phase through bagging and randomized node optimization.
- **Bagging:** train each tree in a forest on a different training subset, sampled at random from the same labeled dataset to reduce overfitting/improve generalization
- **Randomized Node Optimization (RNO)** to improve efficiency, when training the jth node only make available a subset $\mathcal{T}_j \subset \mathcal{T}$ of parameter values. ie optimizing each splite node j as

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}_j} I(\mathcal{S}_j, \boldsymbol{\theta})$$

- In cases $|\mathcal{T}| = \infty$, we can use the parameter $\rho = |\mathcal{T}_j|$. ρ controls the degree of randomness in a tree and usually has a fixed value for all nodes. $\rho = 1$ each split node taks a single randomly chosen set of values for $\boldsymbol{\theta}$ (max randomness)
- bagging and RNO can be used together

1.4 Forest Ensamble



22 A. Criminisi and J. Shotton

Fig. 3.8 Ensemble model. (**a**) The posteriors of four different regression trees (shown with different colors). Some correspond to higher confidence (peakier density curves) than others. (**b**) An ensemble posterior $p(y|\mathbf{v})$ obtained by averaging all tree posteriors. (**c**) The ensemble posterior $p(y|\mathbf{v})$ obtained as a product of all tree posteriors. Both in (**b**) and (**c**) the ensemble output is influenced more by the more informative trees

1.4.1 Combining Trees into a Forest

- A random decision forest is an ensamble of randomly trained decision trees. All trees are trained independently
- In a forest with T trees, the variable $t \in \{1, \ldots, T\}$ to index the component trees
- combining all tree predicitions into asingle forest prediction may be done by simple averaging

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

1.4.2 Key Model Parameters

- D: max tree depth
- ρ: amount of randomness and its type
- T: Forest size
- the choice of weak learner model, the training objective function, the choice of features in practical applications