Chapter 9

Object recognition – Random Forests

9.9 Random forests

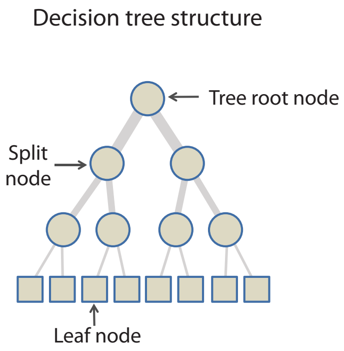
Random forests — a classification approach that is especially well suited for problems with many classes when large datasets are available for training.

They naturally deal with more than two classes, provide probabilistic outputs, offer excellent unseen data generalization, and are inherently parallel.

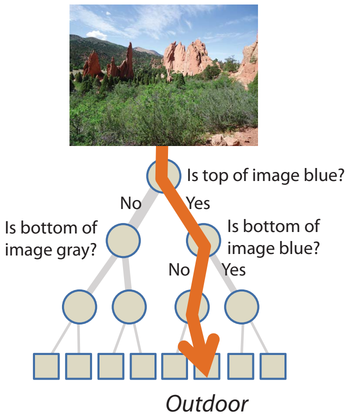
- in 1993, Quinlan offered an approach called the C4.5 algorithm to train decision trees optimally [Quinlan, 1993]
- a single decision tree concept was extended to multiple such trees in a randomized fashion, forming random forests
- some aspects of random forests resemble the boosting strategy since weak classifiers are associated with individual tree nodes and the entire forest yields a strong classification decision

- Two main decision making tasks
  - *classification*
  - *regression*
- In classification (e.g., when classifying images into categories denoting types of captured scenes – beach, road, person, etc.), the decision-making output is a class label
- In non-linear regression (e.g., predicting severity of flu season from – possibly multi-dimensional – social network data), the outcome is a continuous numeric value

- a *decision tree* consists of internal (or split) nodes and terminal (or leaf) nodes (see Figure 9.1)
- arriving image patterns are evaluated in respective nodes of the tree and – based on the pattern properties—are passed to either left or right child nodes
- leafs L store the statistics of the patterns that arrived at a particular node during training
  - when a decision tree $\mathcal{T}_t$ is used for classification, the stored statistical information contains the probability of each class ω_r, $r \in 1, ..., R$ or $p_t(\omega_r|L)$
  - if used for regression, the statistical information contains a distribution over the continuous parameter that is being estimated
  - for a combined *classification–regression* task, both kinds of statistics are collected
- a random forest $\mathcal{T}$ then consists of a set of T such trees and each tree $\mathcal{T}_t$, $t \in \{1, ..., T\}$, is trained on a randomly sampled subset of the training data
- ensembles of slightly different trees (differences resulting, e.g., from training on random training subsets) produce much higher accuracy and better noise insensitivity compared to single trees when applied to previously unseen data, demonstrating excellent generalization capabilities



(a)



(b)

Figure 9.1: Decision tree. (a) Decision trees contain one root node, internal or split nodes (circles), and terminal or leaf nodes (squares). (b) A pattern arrives at the root and is sequentially passed to one of two children of each split node according to the node-based split function until it reaches a leaf node. Each leaf node is associated with a probability of a specific decision, for example associating a pattern with a class label. [*Based on [Criminisi et al., 2011]*] *A color version of this figure may be seen in the color inset—Plate 1.*

- Once a decision tree is trained, predefined binary tests are associated with each internal node and unseen data patterns are passed from the tree root to one of the leaf nodes.
- The exact path is decided based on the outcome of the internal-node tests, each of which determines whether the data pattern is passed to one or the other child node.
- The process of binary decisions is repeated until the data pattern reaches a leaf node.
- Each of the leaf nodes contains a *predictor*, i.e., a classifier or a regressor, which associates the pattern with a desired output (classification label, regression value).
- If a forest of many trees is employed, the individual tree leaf predictors are combined to form a single prediction. In this sense, the decision-making process based on the node-associated binary predictors is fully deterministic.

9.9.1 Random forest training

- decision-making capabilities of the individual tree nodes depend on the predefined binary tests associated with each internal node and on the leaf predictors
- parameters of the binary tests can be either expert-designed or result from training
  - S_i — subset of training data reaching node i
  - S_i^L and S_i^R — subsets of training data reaching the left or right child nodes of node i
- decisions at each node are binary ...

$$S_i = S_i^L \cup S_i^R \,, \qquad S_i^L \cap S_i^R = \emptyset \,. \tag{9.1}$$

- training process constructs a decision tree for which parameters of each binary test were chosen to minimize some objective function
- to stop construction of tree children at a certain node of a certain branch, tree-growth stopping criteria are applied
- if the forest contains T trees, each tree $\mathcal{T}_t$ is trained independently of the others using a randomly selected subset of the training set per tree

- 4-class classification problem, the same number of 2D patterns belong to each class (Figure 9.2)
- comparing two of many ways in which the feature space may be split—say, using a half-way horizontal or half-way vertical split line—both yield more homogeneous subsets (higher similarity of subset-member patterns) and result in a lower entropy of the subsets than was the case prior to the splits
- change in entropy, called the *information gain* I, is

$$I = H(S) - \sum_{i \in \{1,2\}} \frac{|S^i|}{|S|} H(S^i) \,. \tag{9.2}$$

- note that the vertical split in Figure 9.2 separates the classes much better than the horizontal split and this observation is reflected in the differences in the information gain
- parameters of the internal-node binary decision elements can be set so that the information gain achieved on the training set by each split is maximized
- forest training is based on this paradigm

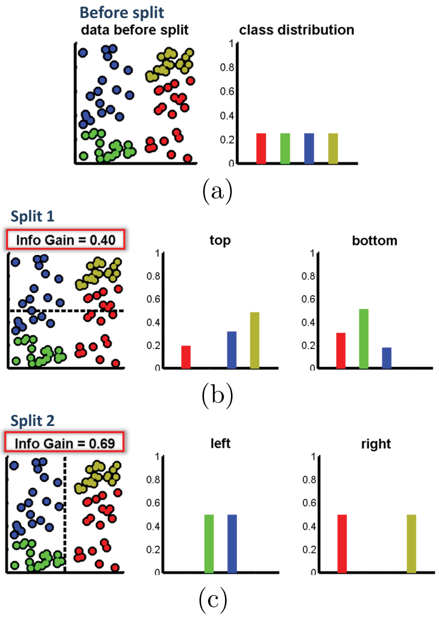


Figure 9.2: Information gain resulting from a split. (a) Class distributions prior to the split. (b) Distributions after a horizontal split. (c) Distributions after a vertical split. Note that both yield more homogeneous subsets and that the entropy of both subsets is decreased as a result of these splits. [*Based on [Criminisi et al., 2011]*] *A color version of*

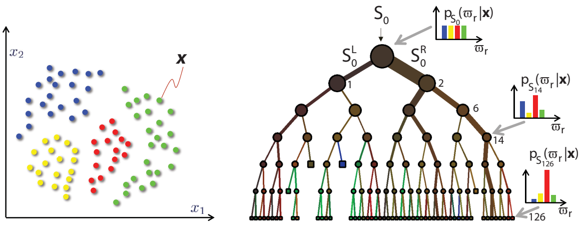

x_2
$\mathbf{x}$
x_1
S_0
S_0^L
S_0^R
$p_{S_0}(\varpi_r|\mathbf{x})$
ϖ_r
1
2
6
14
$p_{S_{14}}(\varpi_r|\mathbf{x})$
ϖ_r
$p_{S_{126}}(\varpi_r|\mathbf{x})$
ϖ_r
126

Figure 9.3: Tree training. Distribution of two-dimensional feature patterns in the feature space is reflected by the class distribution at the root-node level. Here class labels are color-coded and each class includes an identical number of patterns. As a result of training, binary decision functions associated with each split node are optimized—note the increased selectivity of class distributions at nodes more distant from the root (reflecting decreasing entropy). Relative numbers of training patterns passing through individual tree branches are depicted by their thickness. The branch colors correspond to the distribution of class labels. [*Based on [Criminisi et al., 2011]*] *A color version of this figure may be seen in the color inset—Plate 3.*

- binary *split function* associated with a node j

$$h(\mathbf{x}, \theta_j) \in \{0, 1\} \tag{9.3}$$

directs the patterns $\mathbf{x}$ arriving at node j to either the left or the right child (0 or 1 decision – Figure 9.3

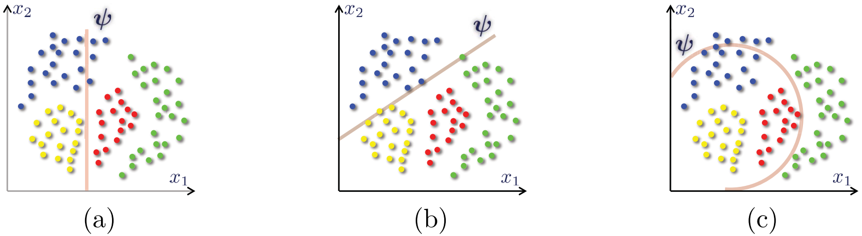


Figure 9.4: Weak learners can use a variety of binary discrimination functions. (a) Axis-aligned hyperplane. (b) General hyperplane. (c) General hypersurface. [*Based on [Criminisi et al., 2011]*] *A color version of this figure may be seen in the color inset—Plate 4.*

- Figure 9.4 ...
  these node-associated split functions play the role of weak classifiers
- the weak learner at node j is characterized by
  parameters $\theta_j = (\phi_j, \psi_j, \tau_j)$ defining the feature selection function ϕ (specifying which features from the full feature set are used in the split function associated with node j),
  data separation function ψ (which hypersurface type is used to split the data, e.g., axis-aligned hyperplane, oblique hyperplane, general surface, etc.
- threshold τ driving the binary decision.

- parameters θ_j must be optimized for all tree nodes j during training, yielding optimized parameters θ_j^*
- one way to optimize the split function parameters is to maximize the information gain objective function
$$\theta_j^* = \underset{\theta_j}{\operatorname{argmax}}\, I_j \ , \tag{9.4}$$
where $I_j = I(S_j, S_j^L, S_j^R, \theta_j)$ and S_j, S_j^L, S_j^R represent training data before and after the left/right split at node j

The decision tree is constructed during the training and a *stopping criterion* is needed for each tree node to determine whether child-nodes should be formed, or tree-branch construction terminated.

Meaningful criteria include:

- defining a maximum allowed tree depth D (this is very popular)
- allowing the node to form child nodes only if a pre-specified minimum information gain is achieved by a split during training
- not allowing child-node construction if a node is not on a frequented data path, i.e., if a node processes less than a pre-defined number of training patterns

After training, tree leaf nodes contain information that will be used during the decision making process.

- If the goal is *classification*, each leaf node L stores the empirical distribution over the classes associated with a training subset that has reached that particular leaf node.
- Considering R classes, a probabilistic predictor model for pattern $\mathbf{x}$ and tree $\mathcal{T}_t$ is
$$p_t(\omega_r|\mathbf{x}) \, , \ \text{where } r \in \{1, ..., R\} \, . \tag{9.5}$$
- If a **regression** task is considered, the output is a continuous variable. The leaf predictor model yields a posterior over the desired continuous variable.

A forest $\mathcal{T}$ consists of T trees that are randomly different from each other.

- The randomness is introduced during training, for example by employing
  - a random subset of the training data for each tree of the forest (so called *bagging*),
  - using randomized node optimization,
  - or other approaches.

9.9.2 Random forest decision making

The behavior of the random forest decision process depends on

- number T of trees forming the forest
- maximum allowed tree depth D
- parameters of the randomness
- choice of the weak learner model
- objective function used for tree training
- choice of features representing the underlying data

Training described above is performed independently for each tree and may be performed in parallel

- when using a forest consisting of T trees, the testing-set pattern $\mathbf{x}$ is simultaneously provided to all T roots of T trained trees
- tree-level processing can also be performed in parallel until the pattern reaches the tree leaves, each of which is providing a leaf-specific prediction

All tree predictions must be combined to form a single forest prediction that represents the forest output

- forest prediction can be obtained in several ways
- e.g., by averaging all tree predictions
- or multiplying the tree outputs together

$$p(\omega_r|\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} p_t(\omega_r|\mathbf{x}) \qquad \text{or} \qquad p(\omega_r|\mathbf{x}) = \frac{1}{Z}\prod_{t=1}^{T} p_t(\omega_r|\mathbf{x})\,, \tag{9.6}$$

where $1/Z$ provides probabilistic normalization

Tree **ensemble models** combine contributions from many trees into a single forest-level output.

- Figure 9.5 — outputs from three randomly trained decision trees used for a classification task, these outputs are combined according to equation (9.6).

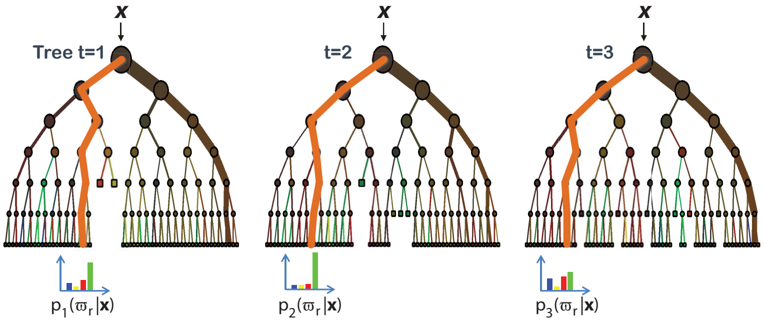

x
Tree t=1
t=2
t=3
$p_1(\varpi_r|\mathbf{x})$
$p_2(\varpi_r|\mathbf{x})$
$p_3(\varpi_r|\mathbf{x})$

Figure 9.5: Decision making process using classification random forests—the same unseen pattern $\mathbf{x}$ arrives simultaneously at the roots of all trees forming the forest. At each node at which the pattern arrives, a test optimized during training is employed to direct the pattern to one of the two children. Due to the random character of training, each tree of the forest includes different split functions and the same pattern follows a different path through the tree nodes until a leaf is reached. Once the leaf is reached in each tree, the tree-specific class posteriors $p_t(\omega_r|\mathbf{x})$ are averaged (or multiplied) to yield a forest posterior $p(\omega_r|\mathbf{x})$. Note that there is one and only one path how a specific pattern $\mathbf{x}$ passes through the tree during decision making. Consequently, the probability $p_t(\omega_r|\mathbf{x})$, while linked to a single leaf node, represents the posterior associated with the entire tree. [*Based on [Criminisi et al., 2011]*] *A color version of this figure may be seen in the color inset—Plate 5.*

- Figure 9.6 — prediction outputs from multiple trees ($T = 4$) can be combined for a regression-task random forest
- *Let the posterior outputs representing prediction of the desired continuous variable y for all four individual trees for pattern* $\mathbf{x}$ *be* $p_t(y|\mathbf{x})$, $t \in \{1, .., 4\}$.
  - Figure 9.6c shows the forest output obtained via averaging of individual tree outputs
  - Figure 9.6d gives the output obtained by multiplying tree outputs
  - in both panels, the forest output benefits from combining the outputs and is influenced more strongly by the more confident portions of the individual tree outputs
- individual tree contributions may be noisy and averaging many tree posteriors decreases noise sensitivity
- product-based ensemble models are more sensitive to noise

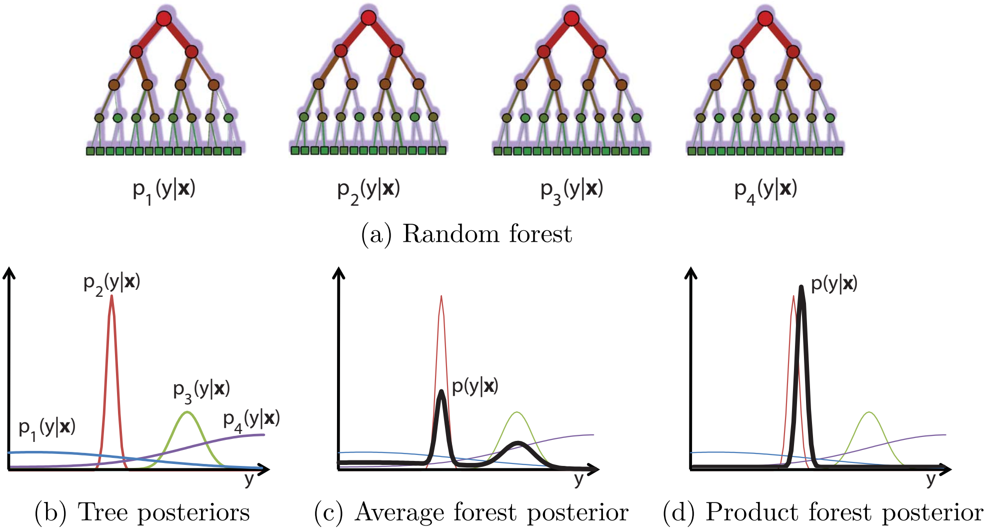


Figure 9.6: Random forest ensemble model for predicting continuous variable y. (a) Random forest consisting of 4 trees. (b) Posteriors of individual trees $p_t(y|\mathbf{x})$. (c) Forest posterior resulting from average-based ensemble model. (d) Forest posterior resulting from multiplication-based ensemble model. Note the strongest influence of the more informative trees on the forest posterior. [*Based on [Criminisi et al., 2011]*] *A color version of this figure may be seen in the color inset—Plate 6.*

9.10 References

Criminisi A., Shotton J., and Konukoglu E. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. Technical Report MSR-TR-2011-114, Microsoft Research, Ltd., Cambridge, UK, 2011.

Quinlan J. R. *C4.5: Programs for machine learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.