Decision Forests for Computer Vision and Medical Image Analysis Chapters 5 & 6

1 Regression Forests

Regression forests are used for nonlinear regression of multi dimensional dependent variables given multi dimensional independent input. The primary difference between regression and classification forests is that the output label to be associated with an input data point is continuous in regression.

Problem Statement: Given a labeled set of training data, learn a general mapping which associates previously unseen, independent test data points with their dependent, continuous output prediction.

- A regression forest is a collection of randomly trained regression trees
- given a multivariate input $\mathbf{v} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ we wish to associate a continuous multivariate label $\mathbf{y} \in \mathbb{R}^n$. More generally, we want to estimate the probability density function $p(\mathbf{y}|\mathbf{v})$
- **Predicition Model**: we can use a probability density function over the continuous variable $\mathbf{y}$. Given the tth tree, the leaf output takes the form $p_t(\mathbf{y}|\mathbf{v})$

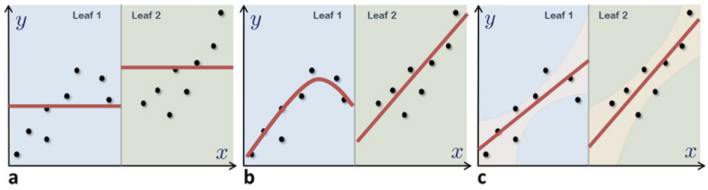

Fig. 5.3 Example predictor models. (**a**) Constant function. (**b**) Polynomial and linear functions. (**c**) Probabilistic linear model, which returns the conditional distribution $p(y|x)$. The confidence of the prediction is denoted with the *shaded area*

- **Ensemble Model**: The regression forest posterior is the average of all individual tree posteriors:

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

- **Randomness Model**: Using RNO, the amount of randomness is controlled during training by the parameter $\rho = |\mathcal{T}_j|$ and the random subsets of split parameters $\mathcal{T}_j$ can be generated when training the jth node
- **Training Objective Function** Again, a split node j is optimized as

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

  however the form of the objective function, I, differs from classification. The general regression information gain is

$$I(\mathcal{S}_j, \boldsymbol{\theta}) = \sum_{\mathbf{v}\in\mathcal{S}_j} \log(|\Lambda_{\mathbf{y}}(\mathbf{v})|) - \sum_{i\in\{\mathrm{L,R}\}} \left(\sum_{\mathbf{v}\in\mathcal{S}_j^i} \log(|\Lambda_{\mathbf{y}}(\mathbf{v})|) \right)$$

  with $\Lambda_{\mathbf{y}}$ the conditional covariance matrix computed from probabilistic linear fitting:

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

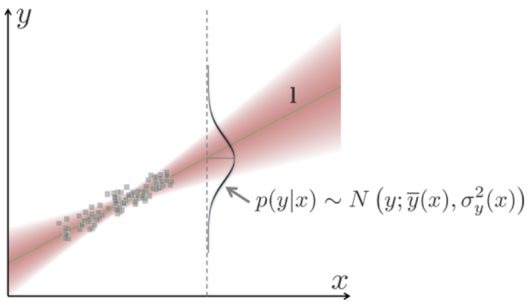

Fig. 5.4 Probabilistic line fitting. Given a set of training points we can fit a line $\mathbf{l}$ to them, *e.g.* by least squares or RANSAC. In this example $\mathbf{l} \in \mathbb{R}^2$. Matrix perturbation theory enables us to estimate a probabilistic model of $\mathbf{l}$ from where we can derive $p(y|x)$ (modeled here as a Gaussian) [80]. Training a regression tree involves minimizing the uncertainty of the prediction $p(y|x)$ over the training set. Therefore, the training objective is a function of σ_y^2 evaluated at the training points

- **Weak Learner Model**: Data arriving at a split node j are separated into left and right children according to a binary weak learner:

$$h(\mathbf{v}, \boldsymbol{\theta}_j) \in \{0, 1\}$$

  like classification, there are axis aligned hyperplanes, oriented hyperplanes, and conic sections

1.1 Effect of Model Parameters

examples use axis aligned weak learners, probabilistic linear predictor models

- **Forest Size** as the number of trees increases both the prediction mean curve and its uncertainty become smoother. Thus smoothness of the interpolating curve is controlled here by the forest size T
- **Tree Depth** A regression forest with $D = 1$ corresponds to linear regression and underfits, as tree depth increases leads to overfitting
- **Spatial Smoothness and Testing Uncertainty** uncertainty increases away from training data and in general, the larger the forest size the smoother the mean predicition curve.

2 Density Forests

- A density forest is a collection of randomly trained clustering trees with leaves that contain simple prediction models such as Gaussians
- **Problem Statement:** Given a set of unlabeled observations we wish to estimate the probability density function from which such data have been generated
- **Training Objective Function**: Given a collection of points (no labels) $\mathcal{S}_0 = \{\mathbf{v}\}$ train each individual tree independently. Optimizing the jth split node:

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

  with generic information gain I:

$$I(\mathcal{S}_j, \boldsymbol{\theta}) = H(\mathcal{S}_j) - \frac{|\mathcal{S}_j^L|}{|\mathcal{S}_j|} H(\mathcal{S}_j^L) + \frac{|\mathcal{S}_j^R|}{|\mathcal{S}_j|} H(\mathcal{S}_j^R)$$

  with unsupervised/continuous entropy:

$$H(\mathcal{S}) = \frac{1}{2} \log\big((2\pi e)^d |\Lambda(\mathcal{S})|\big)$$

with Λ the associated $d \times d$ covariance matrix. The information gain then reduces to

$$I(\mathcal{S}_j, \boldsymbol{\theta}) = \log(|\Lambda(\mathcal{S}_j)|) - \sum_{i \in \mathrm{L,R}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|} \log\big(|\Lambda(\mathcal{S}_j^i)|\big)$$

with $|\cdot|$ indicating a determinant for matrices or cardinality for sets. by maximizing, the tree training procedure tends to split the original dataset $\mathcal{S}_0$ into a number of compact clusters with centers that tend to be placed in areas of high data density, while the separating surfaces are placed along regions of low density.

- **Prediciton model**: set of leaves in the tth tree defines a partition of the data such that

$$l(\mathbf{v}) : \mathbb{R}^d \to \mathcal{L} \subset \mathbb{N}$$

where l denotes the leaf reached by the input $\mathbf{v}$ and $\mathcal{L}$ the set of all leaves in a given tree. The statistics of all training points arriving at each leaf node are summarized by a single multivariate Gaussian distribution, then the output of the tth tree is

$$p_t(\mathbf{v}) = \frac{\pi_{l(\mathbf{v})}}{Z_t} \mathcal{N}(\mathbf{v}; \boldsymbol{\mu}_{l(\mathbf{v})}, \Lambda_{l(\mathbf{v})})$$

Here $\boldsymbol{\mu}_l$ is the mean of all points reaching the leaf l, the scalar π_l is the proportion of all training points that reach the leaf l ($\pi_l = \frac{|\mathcal{S}_l|}{|\mathcal{S}_0|}$) The partition function Z_t is

$$Z_t = \int_{\mathbf{v}} \pi_{l(\mathbf{v})} \mathcal{N}(\mathbf{v}; \boldsymbol{\mu}_{l(\mathbf{v})}, \Lambda_{l(\mathbf{v})}) d\mathbf{v}$$

the partition function Z_t is the sum of all the volumes subtended by each Gaussian cropped by its associated partition cell

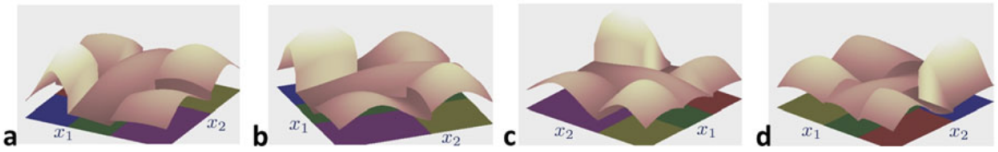

Fig. 6.2 A tree density is piece-wise Gaussian. (**a**, **b**, **c**, **d**) Different views of a tree density $p_t(\mathbf{v})$ defined over an illustrative 2D feature space. Each individual Gaussian component is defined over the bounded partition cell associated with the corresponding tree leaf. See text for details

- **Ensemble model**: again, average of all tree densities

$$p(\mathbf{v}) = \frac{1}{T} \sum_t^T p_t(\mathbf{v})$$

2.1 Effect of Model Parameters

- **Tree depth**: deep trees lead to overfitting by further splitting/smaller Gaussians
- **Forest size**: Increase in forest size improves generalization