Randomised Decision Forests for Regression

Tae-Kyun (T-K) Kim

Senior Lecturer

https://labicvl.github.io/

References:

A. Criminisi et al., Decision forests, an unified framework, Foundations and Trends in Computer Graphics and Vision, 7:2-3.

Breiman L, Random Forests. Machine Learning, 45 (1), pp 5-32, 2001.

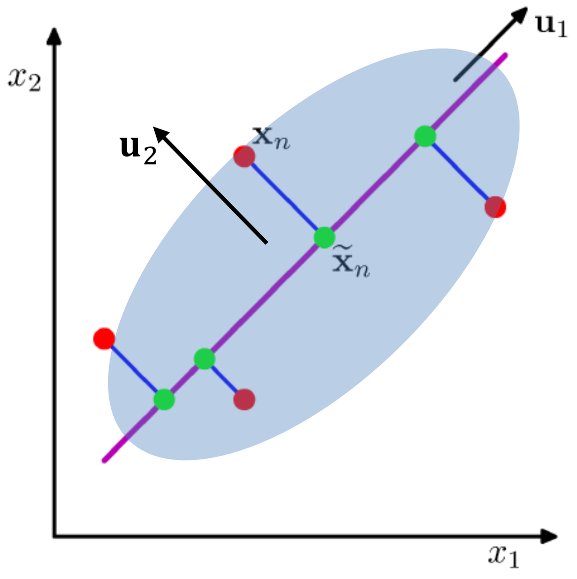
Regression forests

- We discuss the use of random decision forests for the probabilistic estimation of continuous variables.
- Regression forests are used for the nonlinear regression of dependent variables given independent input.
- Both input and output may be multi-dimensional.
- The output can be a point estimate or a full probability density function.
- Regression forests are less popular than their classification counterpart.
- The main difference is that the output label to be associated with an input data is continuous.
- Therefore, the training labels are continuous.
- Consequently the objective function has to be adapted appropriately.
- Regression forests share many of the advantages of classification forests such as efficiency and flexibility.

Nonlinear regression

- Given a set of noisy input data and associated continuous measurements, least squares techniques (closely related to principal component analysis) can be used to fit a linear regressor which minimizes some error computed over all training points.
- Under this model, given a new test input the corresponding output can be efficiently estimated.
- The limitation of this model is in its linear nature, when we know that most natural phenomena have nonlinear behaviour.
- Another well known issue with linear regression techniques is their sensitivity to input noise.
- In geometric computer vision, a popular technique for achieving robust regression via randomization is RANSAC.

select few reliable in-layer data points for modelling.



PCA (see Pattern Recognition lecture notes)

split in recursions

Notations

Input data vector: $\mathbf{v} = (x_1, \cdots, x_d) \in \mathbb{R}^d$

Output/label: $\mathbf{y} \in \mathcal{Y} \subset \mathbb{R}^n$

Subset of training points reaching node j: $\mathcal{S}_j$

Subset of points going to the left child node: $\mathcal{S}_j^{\mathrm{L}}$

Subset of points going to the right child node: $\mathcal{S}_j^{\mathrm{R}}$, s.t. $\mathcal{S}_j = \mathcal{S}_j^{\mathrm{L}} \cup \mathcal{S}_j^{\mathrm{R}}$

Node test parameters: $\boldsymbol{\theta} \in \mathcal{T}$

Node weak learner: $h(\mathbf{v}, \boldsymbol{\theta}_j) \in \{\mathtt{true}, \mathtt{false}\}$

Node objective function: $I = I(\mathcal{S}_j, \boldsymbol{\theta})$

Stopping criteria: e.g. max tree depth = *D*

Leaf predictor model: $p(y|\mathbf{v})$

Forest size i.e. number of trees: T

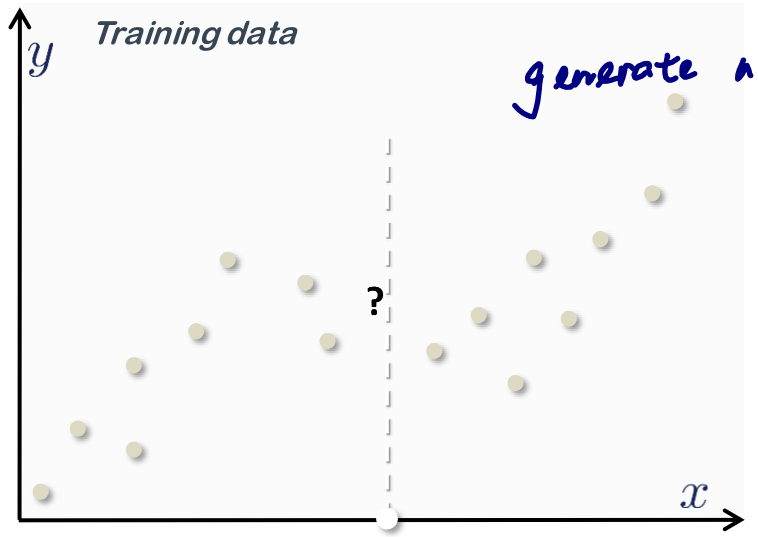
Ensemble model: $p(y|\mathbf{v}) = \dfrac{1}{T}\sum_t^T p_t(y|\mathbf{v})$

Decision forest model for regression

- The regression task: Given a labelled training set learn a general mapping which associates previously unseen independent test data with their correct continuous prediction.
- Like classification the regression task is inductive, with the main difference being the continuous nature of the output.
- In general, a training point is denoted as a labelled pair (v,y). A previously unseen test input (unavailable during training) is shown as a light gray circle on the x axis.

regression: supervised learning (given training sets)

generate new points from existing

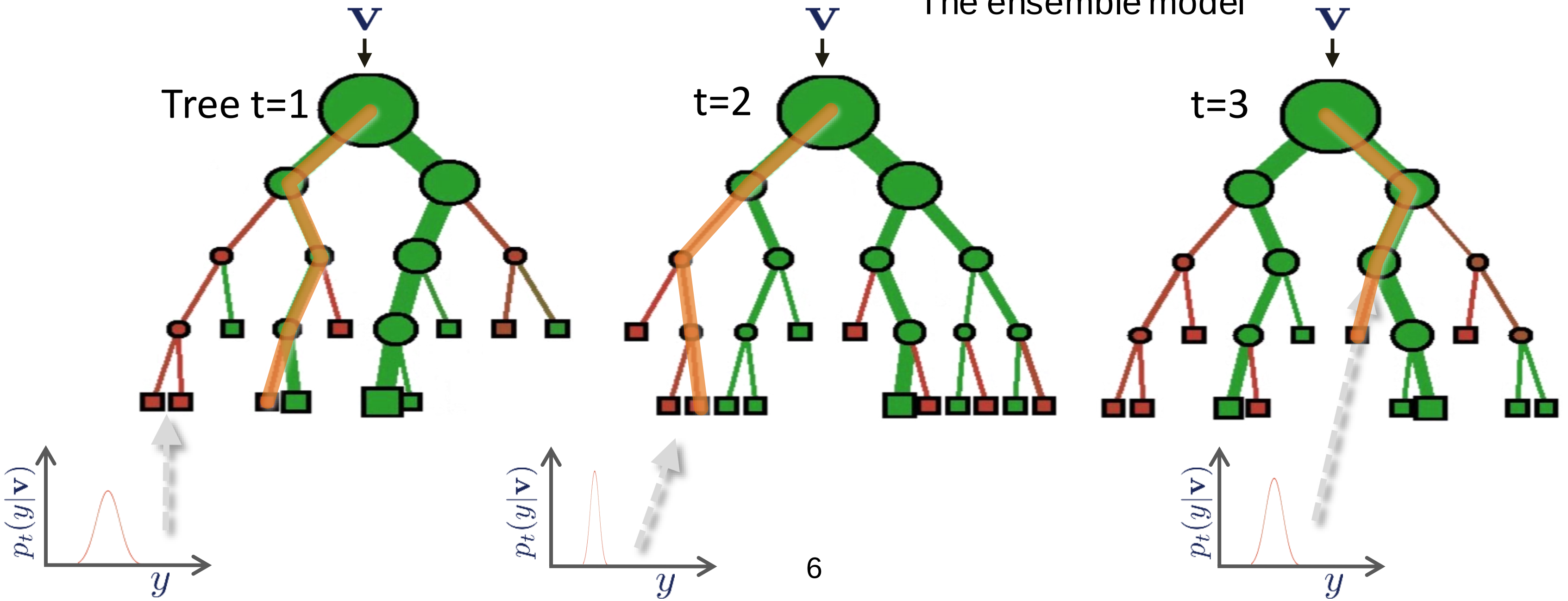


An example of training data and associated continuous ground-truth labels.

Decision forest model for regression

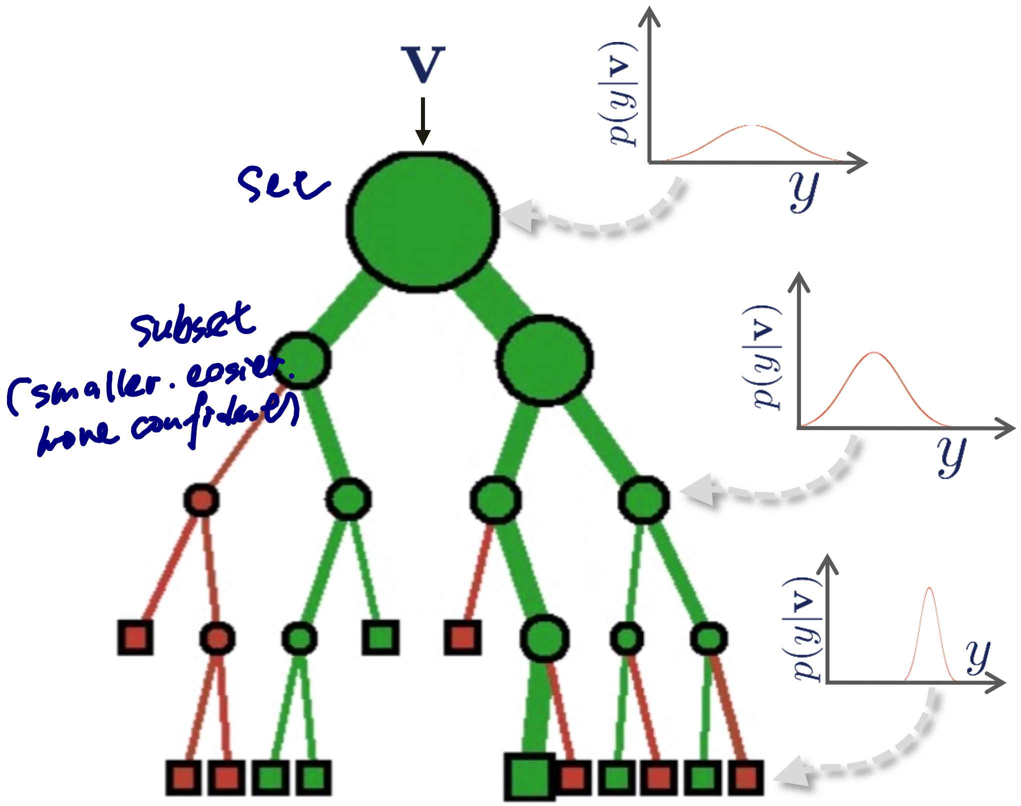
- Formally, given a multi-variate input v we wish to associate a continuous multi-variate label $\mathbf{y} \in \mathcal{Y} \subset \mathbb{R}^n$
- More generally, we wish to estimate the probability density function p(y|v).
- As usual the input is represented as a multi-dimensional feature response vector $\mathbf{v} = (x_1, \cdots, x_d) \in \mathbb{R}^d$
- A regression forest is a collection of randomly trained regression trees.
- Like in classification it can be shown that a forest generalizes better than a single over-trained tree. single tree: overfitting

The ensemble model



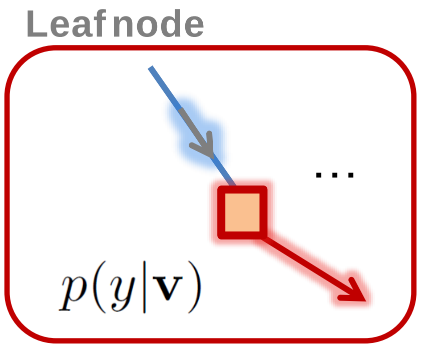
Decision forest model for regression

– A regression tree splits a complex nonlinear regression problem into a set of smaller problems which can be more easily handled by simpler models (e.g., linear ones).



The prediction model

- The first job of a decision tree is to decide which branch to direct the incoming data to.
- When the data reaches a terminal node then that leaf needs to make a prediction.
- The actual form of the prediction depends on the prediction model.
- In classification we have used the pre-stored empirical class posterior as model.
- In regression forests we have a few alternatives.

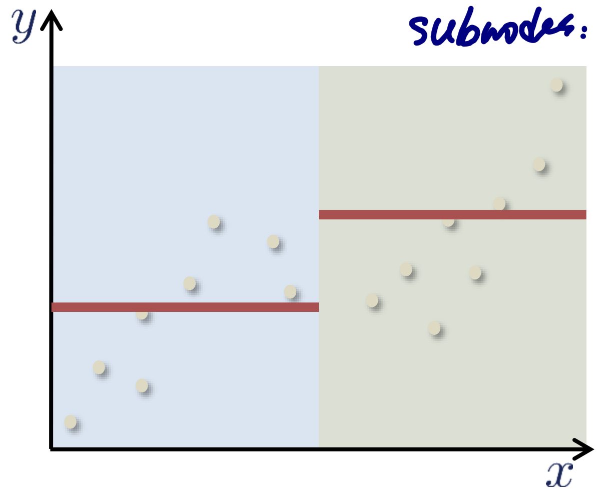


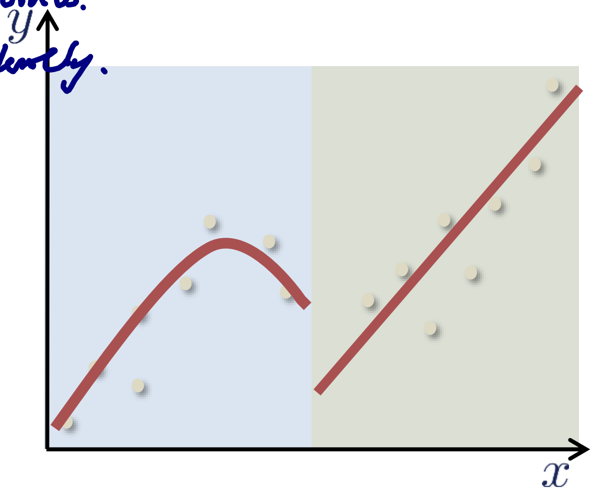
The prediction model

- For instance we could use a polynomial function of a subspace of the input v.
- In the low dimensional example a generic polynomial model is given below.
- This simple model captures both the linear and constant models.

Examples of leaf (predictor) models

subnodes: typical data points modelled independently.





Predictor model: constant

$$y = const$$

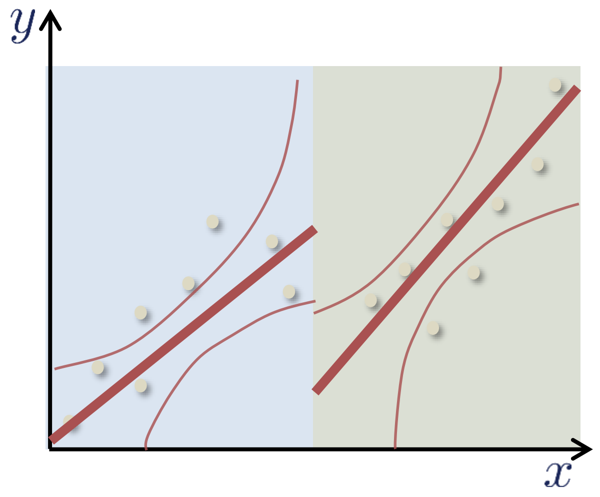
Predictor model: polynomial ((linear/nonlinear))

$$y = \sum_{i=0}^{n} w_i x^i$$

(note: linear for n=1, constant for n=0)

The prediction model

- In this we are interested in output confidence as well as its actual value.
- Thus for prediction we can use a probability density function over the continuous variable y.
- So, given the t-th tree in a forest and an input point v, the associated leaf output takes the form $p_t(y|\mathbf{v})$.
- In the low-dimensional example, we assume an underlying linear model and each leaf yields the conditional p(y|x).



Predictor model: probabilistic-linear

$p(y|x)$

The ensemble model

- Like in classification, the forest output is the average of all tree outputs

$$p(\mathbf{y}|\mathbf{v}) = \frac{1}{T}\sum_{\underset{\text{trees}}{t}}^{T} \underset{\text{var.}}{p_t}(\mathbf{y}|\underset{\text{input}}{\mathbf{v}})$$

- Randomness model:
  - As in classification here we use a randomized node optimization model.
  - Therefore, the amount of randomness is controlled during training by the parameter $\rho = |\mathcal{T}_j|$.
  - The random subsets of split parameters $\mathcal{T}_j$ can be generated on the fly when training the j-th node.

The training objective function

- Forest training happens by optimizing an energy over a training set $\mathcal{S}_0$ of data and associated continuous labels.
- Training a split node j happens by optimizing the parameters of its weak learner:

$$\boldsymbol{\theta}_j^* = \arg \max_{\boldsymbol{\theta}_j \in \mathcal{T}_j} I_j$$

- Now, the main difference between classification and regression forest is in the form of the objective function I.
- We employ a continuous formulation of information gain.

Information gain on continuous distribution

– Entropy and information gain can also be defined for continuous-valued labels and distributions.

– The definition of the information gain remains the same but this time, instead of using the Shannon entropy, the differential entropy is used

summation → intergration

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

Here *y* is a continuous label and *p* is the probability density function estimated from the training points in the set *S*.

– From a practical point of view, in the discrete case, the distribution *p*(*c*) was defined as the empirical distribution (i.e. class histogram) computed from the training set.

– Similarly in the continuous distribution *p*(*y*) can be defined either using parametric distributions or non-parametric methods.

eg. Gaussian (sum=1)

Information gain on continuous distribution

– One of the most popular choice in various applications is to use Gaussian-based models to approximate the density *p*(*y*) due to their simplicity.

– The differential entropy of a *d*-variate Gaussian is defined analytically as

Peak: more containing / lower entropy

dim of data vec.

differentiable: need gradient.

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

where $\Lambda(\mathcal{S})$ is the data covariance matrix.

Gaussian: avoid high complexity in differentiations

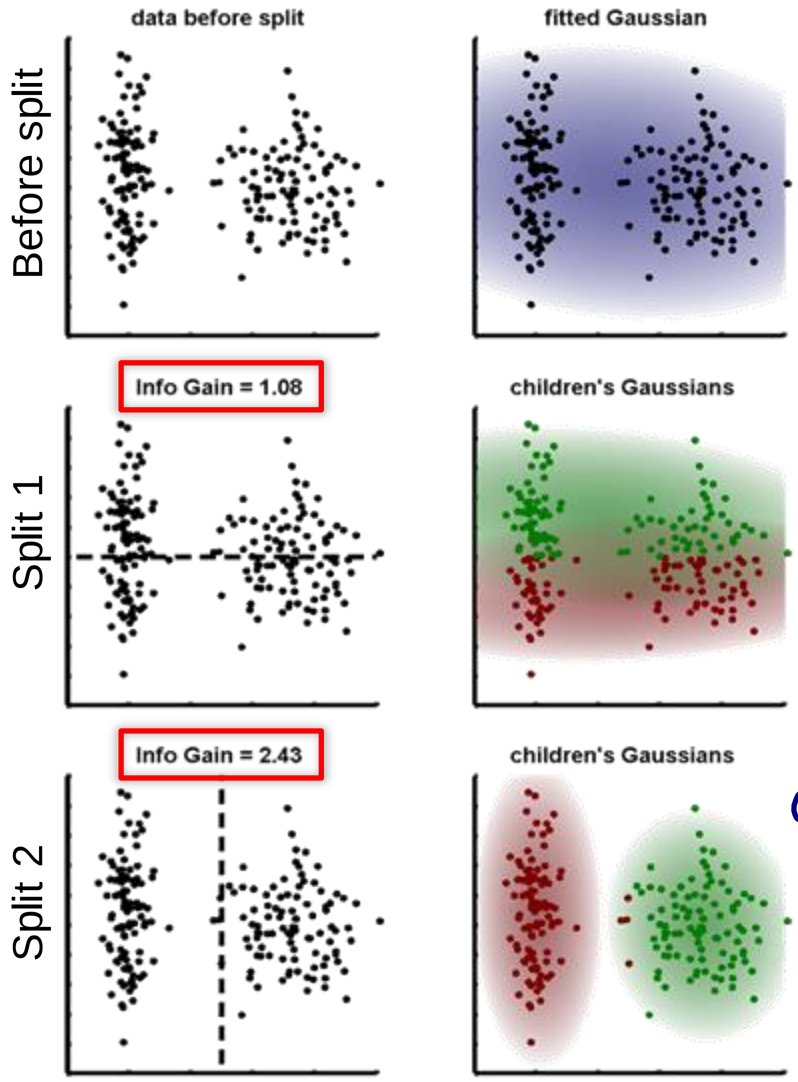
– A toy example in the next slide illustrates the role of the continuous information gain in training.

– Information gain for discrete categorical distribution is for classification, continuous distribution of y for regression, continuous distribution of x for clustering.

– This time we wish to cluster similar points according to their features (again, depicted as the coordinates of a 2D space).

– Given an arbitrary input data point we wish the tree to predict its associated cluster.

A toy example, Information gain on continuous, parametric densities



compact Gaussian

A toy example, Information gain on continuous, parametric densities

- Fitting a Gaussian to the entire initial set *S* produces the density shown in blue, which has a high differential entropy.
- Splitting the data horizontally produces two largely overlapping and slightly smaller Gaussians (in red and green).
- The large overlap indicates a suboptimal separation and is associated with a relatively low information gain ($I = 1.08$).
- Splitting the data points vertically yields better separation, with peakier Gaussians and a correspondingly higher value of information gain ($I = 2.43$).

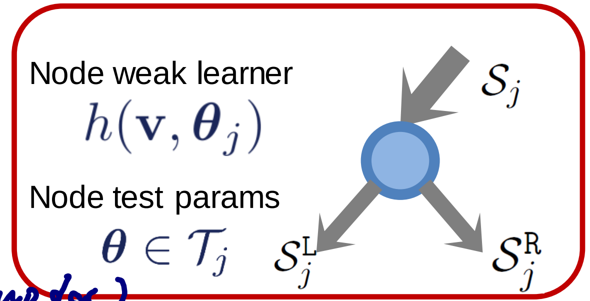
The training objective function

– The following definition of information gain is used:

$$I_j = \underbrace{\sum_{\mathbf{v}\in\mathcal{S}_j} \log(|\Lambda_{\mathbf{y}}(\mathbf{v})|)}_{\text{fixed}} - \underbrace{\sum_{i\in\{\mathrm{L},\mathrm{R}\}} \left(\sum_{\mathbf{v}\in\mathcal{S}_j^i} \log(|\Lambda_{\mathbf{y}}(\mathbf{v})|) \right)}_{\text{minimise (for children nodes)}}$$

Where Λy the covariance matrix computed from probabilistic fitting.

Splitting data at node j


Node weak learner
$h(\mathbf{v}, \boldsymbol{\theta}_j)$
Node test params
$\boldsymbol{\theta} \in \mathcal{T}_j$
$\mathcal{S}_j$
$\mathcal{S}_j^{\mathrm{L}}$
$\mathcal{S}_j^{\mathrm{R}}$

The training objective function

– The error or fit objective function for single-variate output *y* is:

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

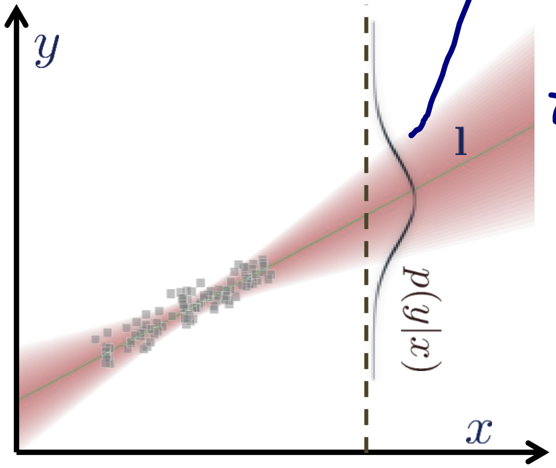
$$H(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{x\in\mathcal{S}} \underbrace{\int_y p(y|x) \log p(y|x)\, dy}$$

$$p(y|x) \sim N\left(y; \overline{y}, \sigma_y^2(x)\right)$$

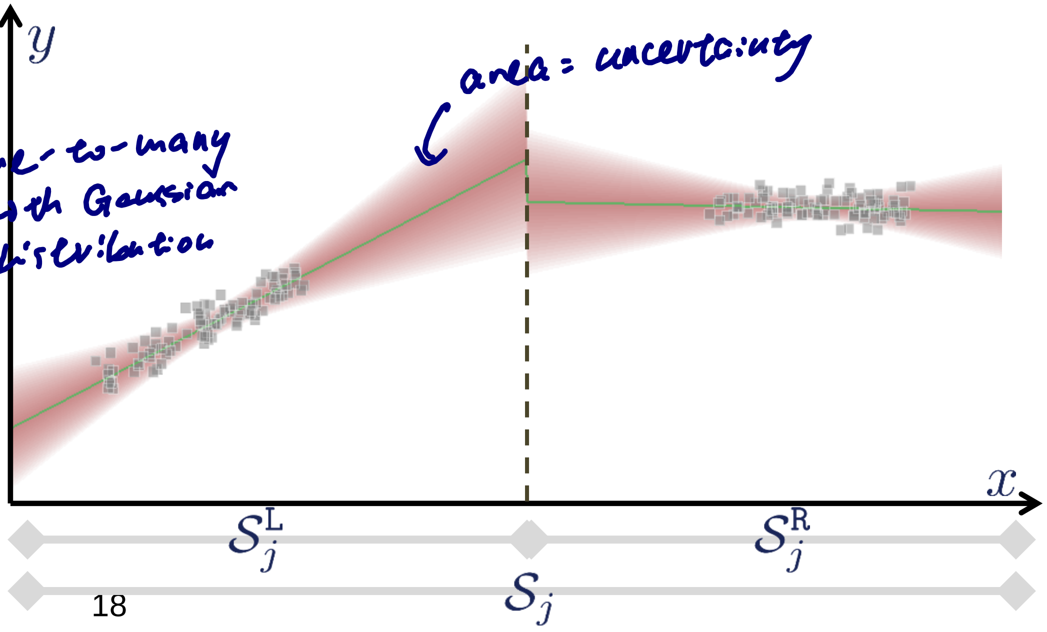
$$H(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{x\in\mathcal{S}} \frac{1}{2} \log\left((2\pi e)^2 \sigma_y^2(x)\right)$$

$$I = \sum_{(x,y)\in\mathcal{S}_j} \log\left(\sigma_y(x)\right) - \sum_{i\in\{\mathtt{L},\mathtt{R}\}} \left(\sum_{(x,y)\in\mathcal{S}_j^i} \log\left(\sigma_y(x)\right) \right)$$

minimise data variance



x→y : one-to-many with Gaussian distribution



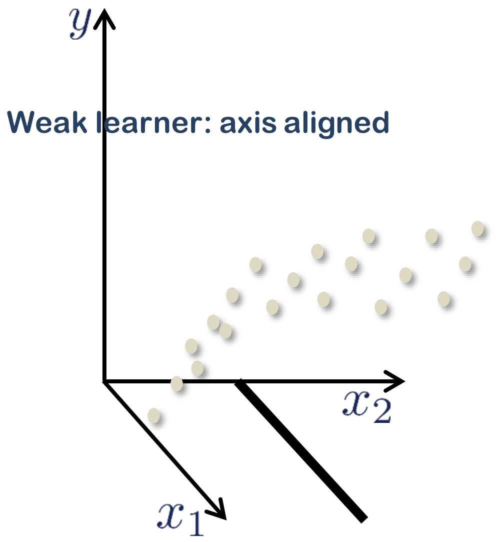
The weak learner model

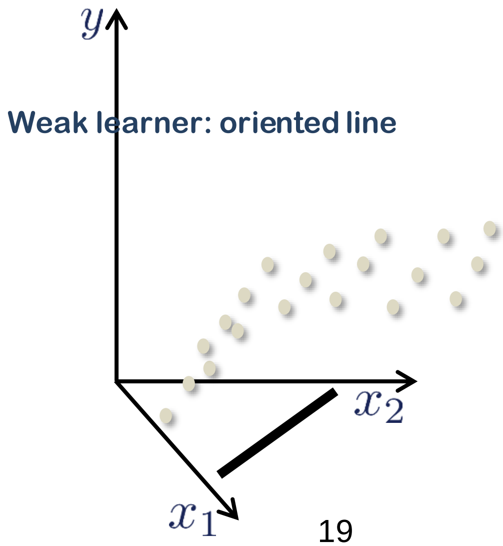
– As usual, the data arriving at a split node *j* is separated into its left or right children according to a binary weak learner stored in an internal node, of the following general form:

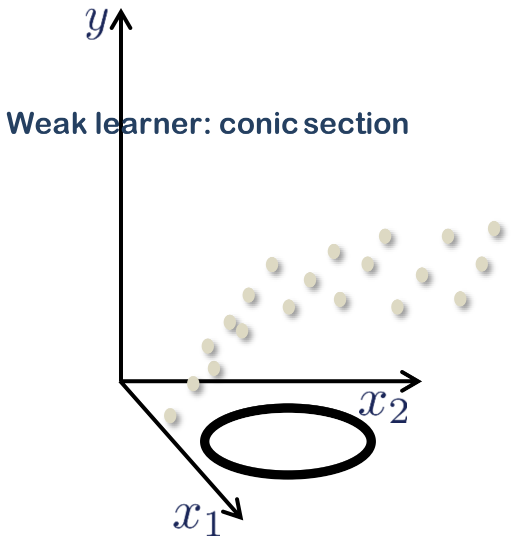
$$h(\mathbf{v},\theta_j) = \{0 \text{ or} 1\}$$

where 0 and 1 can be interpreted as “false” and “true” respectively.

– Like in classification here we consider three types of weak learners: (i) axis-aligned, (ii) oriented hyperplane, (iii) quadratic (for an illustration on 2D→1D regression).

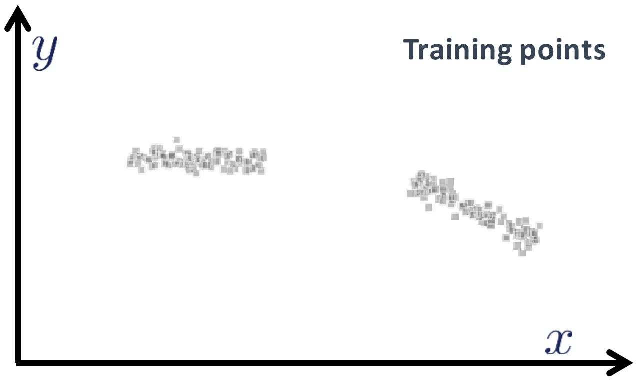




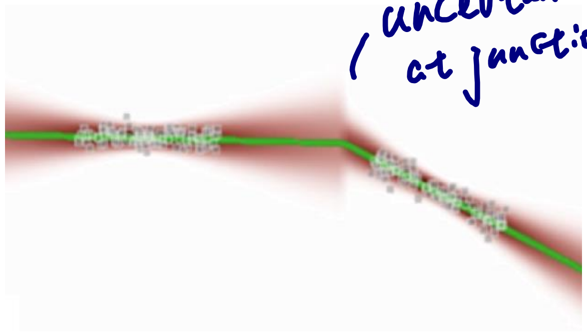


Effect of the forest size

- A forest of shallow trees ($D = 2$) and varying size T is trained.
- We use axis-aligned weak learners, and probabilistic linear predictor models.
- As the number of trees increases both the prediction mean and its uncertainty become smoother.

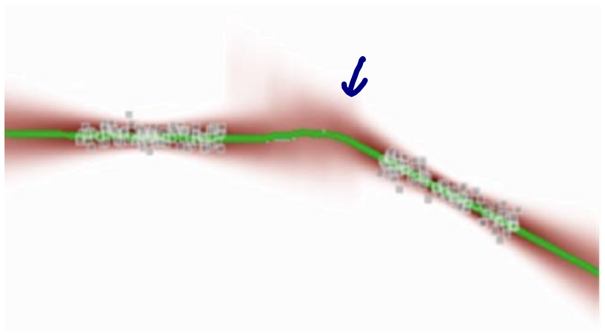


T=1

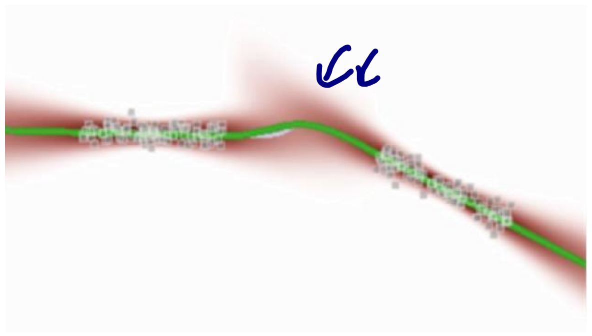


(uncertainty at junctions

T=8



T=400

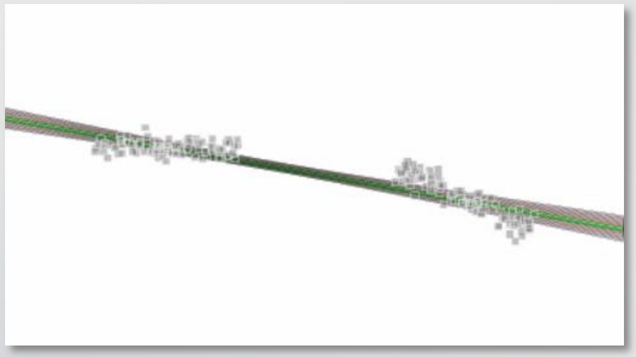
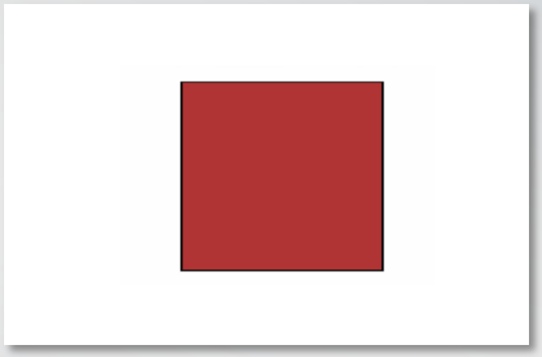
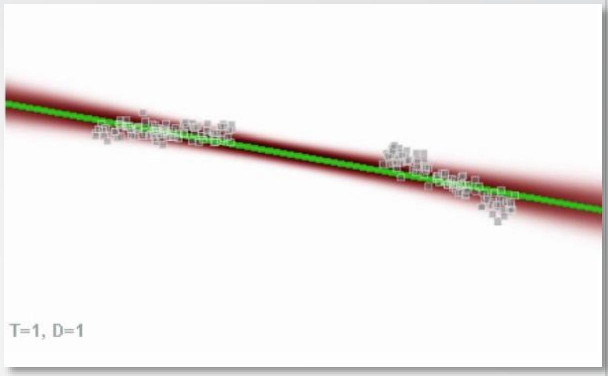


Effect of the tree depth

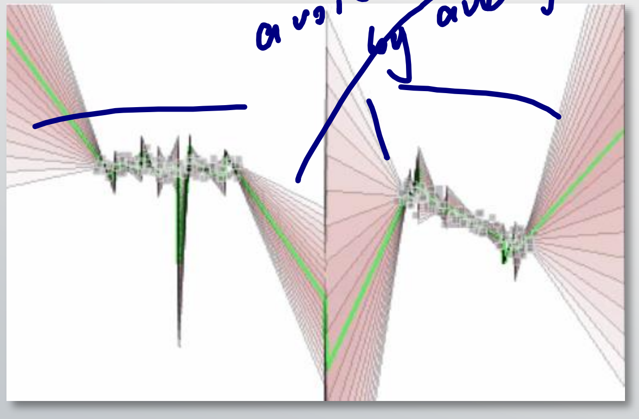
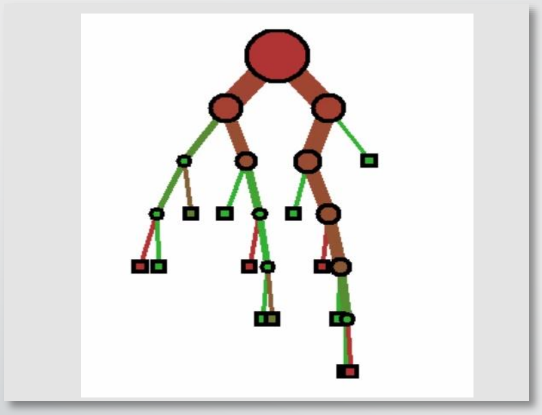
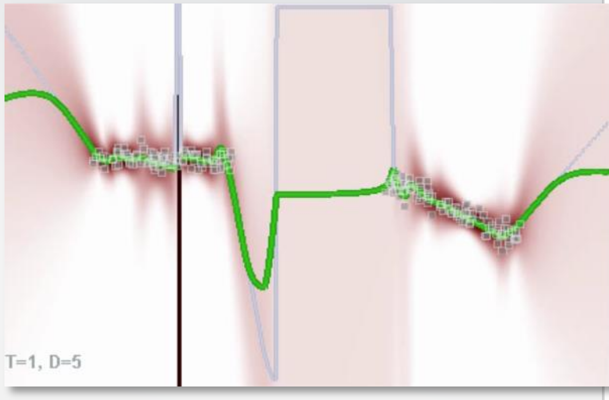
- The effect of varying the maximum allowed tree depth D on the same training set is shown.
- A regression forest with $D = 1$ (top row in figure) corresponds to conventional linear regression (with additional confidence estimation).
- In this case the training data is more complex than a single line and thus such a degenerate forest under-fits.
- In contrast, a forest of depth $D = 5$ (bottom row in figure) yields over-fitting. This is highlighted in the figure by the high frequency variations in the prediction confidence and the mean of $y(x)$.

Effect of the tree depth

	Training	Testing
underfitting	max tree depth D = 1	max tree depth D = 1
overfitting	max tree depth D = 5	max tree depth D = 5

avoid overfitting by averaging weak trees.

(6 videos in this page)

Parameters: T=400, w.l. = aligned, l.m. = prob. line