Prediction or Regression Trees (RT)

- Just like decision trees, greedily fits piecewise constant fits to partitions of the input space
- Many packages automatically switch between RT and DT depending on independent variable. (Essentially only splitting criteria changes).
- At every node of tree T, choose a binary split such that it minimizes Sum of Squared Errors for the data.

$$S = \sum_{c \in Leaves(T)} \sum_{i \in c} (y_i - m_c)^2, \text{where } m_c = \frac{1}{n_c} \sum_{i \in c} y_i \tag{1}$$

- Stop when number of data-points at the current (leaf) node is less than a threshold q or reduction in error by adding new nodes is less than δ

From HTF Fig 9.2

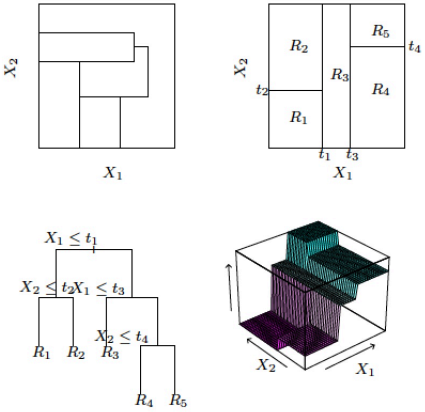


FIGURE 9.2. *Partitions and CART. Top right panel shows a partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the tree corresponding to the partition in the top right panel, and a perspective plot*

E.g. - Baseball Players Salaries

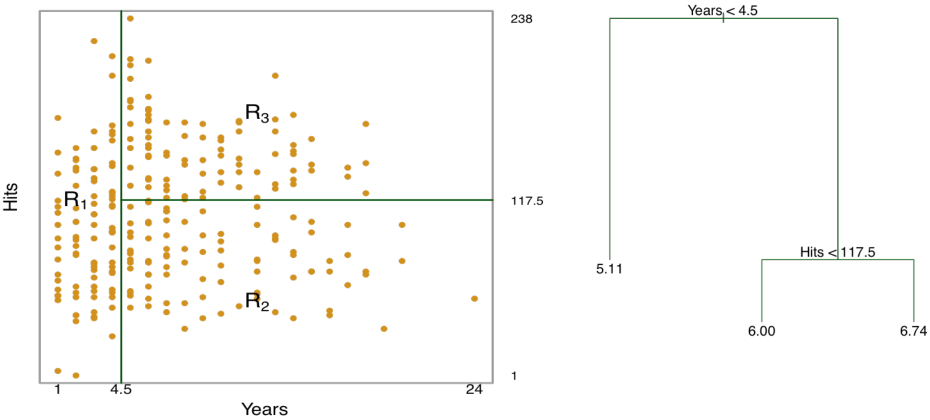
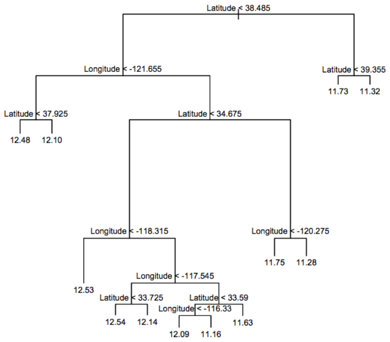
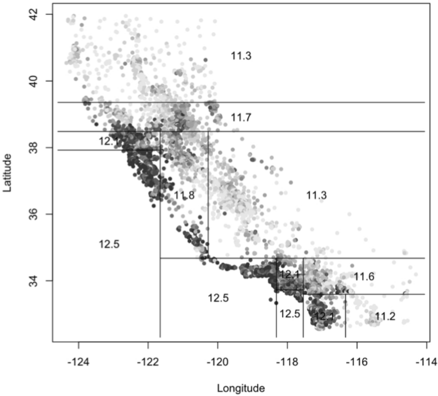


Figure: Regression Tree for Baseball Salaries (in '000, log transformed, so 6 means $402,834

E.g. - California Median House-Prices






```
Regression tree:
tree(formula = log(MedianHouseValue) ~ Longitude + Latitude,
    data = calif)
Number of terminal nodes:  12
Residual mean deviance:  0.1662 = 3429 / 20630
Distribution of residuals:
      Min.    1st Qu.     Median       Mean    3rd Qu.      Max.
-2.759e+00 -2.608e-01 -1.359e-02 -5.050e-15  2.631e-01 1.841e+00
```

Pruning, Cross-Validation and Uncertainty of data

- Avoiding Overfitting
  - By itself, the above training procedure over-fits
  - Use Cross-Validation to prune from bottom up as need be.
  - `prune.tree`, `cv.tree` in `tree` `package`.
- Uncertainty consists of two parts
  - Assuming grown tree is right, prediction is still uncertain
  - The tree itself could change if data changes
- REFERENCE: http://www.stat.cmu.edu/~cshalizi/350/lectures/22/lecture-22.pdf

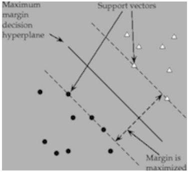
Why Regression Trees?

- Predictions are fast (just traverse the tree!); Interpretation is easy
- In case of missing data, we can still traverse to a sub-tree!
- Model can handle non-smooth regression functions by nature
- Lets us look at different regions at adaptive resolutions
- Ensembles of Trees, (Forests) known to work well in practice

Disadvantages?

- Since splits at any node depend on previous splits, errors are propogated
- Hard to capture higher order interactions
- Small change in dataset could mean a huge change in the tree

SVMs Vs. Support Vector Regression (SVR)*

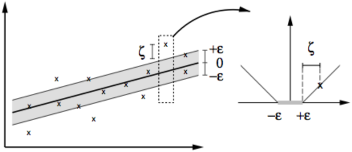


- http://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-the-linearly-separable-case-1.html

- Discriminative Classifier
- Objective is penalized by data-points on the wrong side of the margin
- Leads to sparse solutions depending only on support vectors

$$minimize \ \frac{\|w\|^2}{2} + C\sum_{i=1}^{n}\xi_i \tag{2}$$

$$s.t. \ y_i(w.x_i + b) \geq 1 - \xi_i, \xi_i > 0$$



- SVR: SVM for Function Estimation

$$\hat{y} = \langle w, x\rangle + b \text{ where } w \in \mathbb{R}^d \tag{3}$$

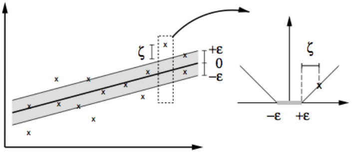
- Objective Function penalizes data-points except those within the ϵ-tube

$$\begin{aligned} & minimize \ \frac{\|w\|^2}{2} + C\sum_{i=1}^{n}(\xi_i + \xi_i{}^*) \\ & s.t. \begin{cases} \langle w, x_i\rangle + b - y_i \leq \epsilon + \xi_i{}^* \\ y_i - \langle w, x_i\rangle - b \leq \epsilon + \xi_i \\ \xi_i, \xi_i{}^* \geq 0 \end{cases} \end{aligned} \tag{4}$$

Optimization of SVR Objective Function

(Linear) SVR is like MLR except (i) cost function is different, (ii) procedure to obtain coefficients is different.

$$\text{Cost}(\epsilon) := \begin{cases} 0 & \text{if } |\xi| \leq \epsilon \\ |\xi| - \epsilon & \text{elsewhere} \end{cases} \tag{5}$$



- Optimization more easily solved in convex dual space
- Solution: $\widehat{y}(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i{}^*)\langle x, x_i \rangle + b$, where x_is are support vectors αs are dual variables (Lagrange Multipliers).
- ϵ and slack penalty (C) obtained through Cross-Validation.

See: Alex J. Smola and Bernhard Scholkopf. "A tutorial on support vector regression", *Statistics and Computing, 2004 for details. http://eprints.pascal-network.org/archive/00000856/01/fulltext.pdf*
R Packages: LinearizedSVR: Linearized Support Vector Regression, http://cran.fhcrc.org/web/packages/LinearizedSVR/index.html; also part of `e1071 package`
`See http://www.jstatsoft.org/v15/i09/paper for other choices and a comparison.`

Advantages of SVRs

- Like SVMs data-points can be Kernelized to achieve non-linear regression
- Formulation remains the same with different loss functions
- Is not affected by dimensionality of data but depends on the number of support vectors
- Online versions for learning based on Stochastic Gradient Descent or Primal-Dual Optimization available