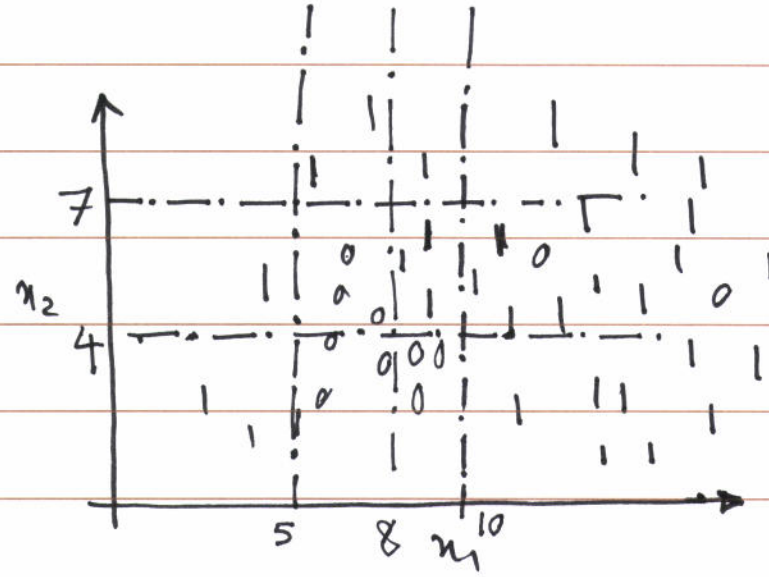
Discussion 7 (10/3/18)

- Useful techniques for projects
  - Decision Tree Reg./Classification
  - Model Averaging < Bagging, AdaBoost
  - Random Forest
  - Dimension Reduction Tech.

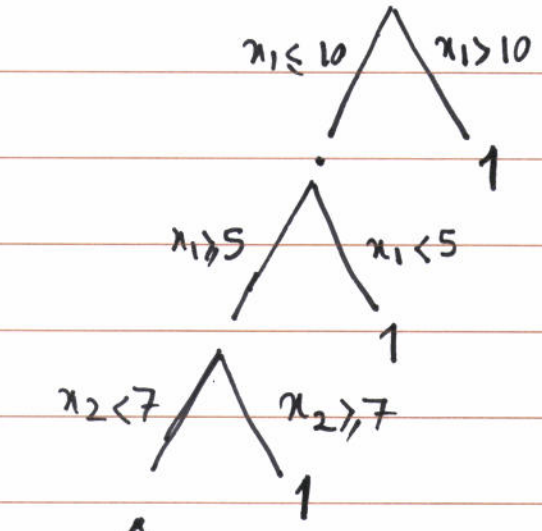
Assessing the size of training:

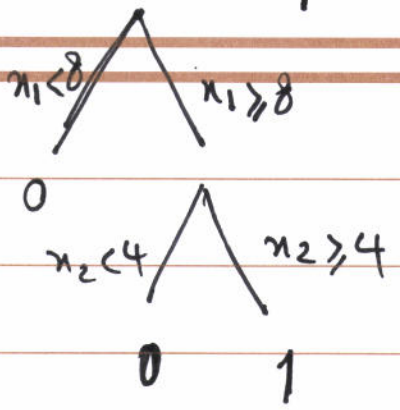
Rule of thumb: 3-10 times larger than degrees of freedom

Decision Tree:



Goal: find a decision tree based classifier s.t. error minimized.

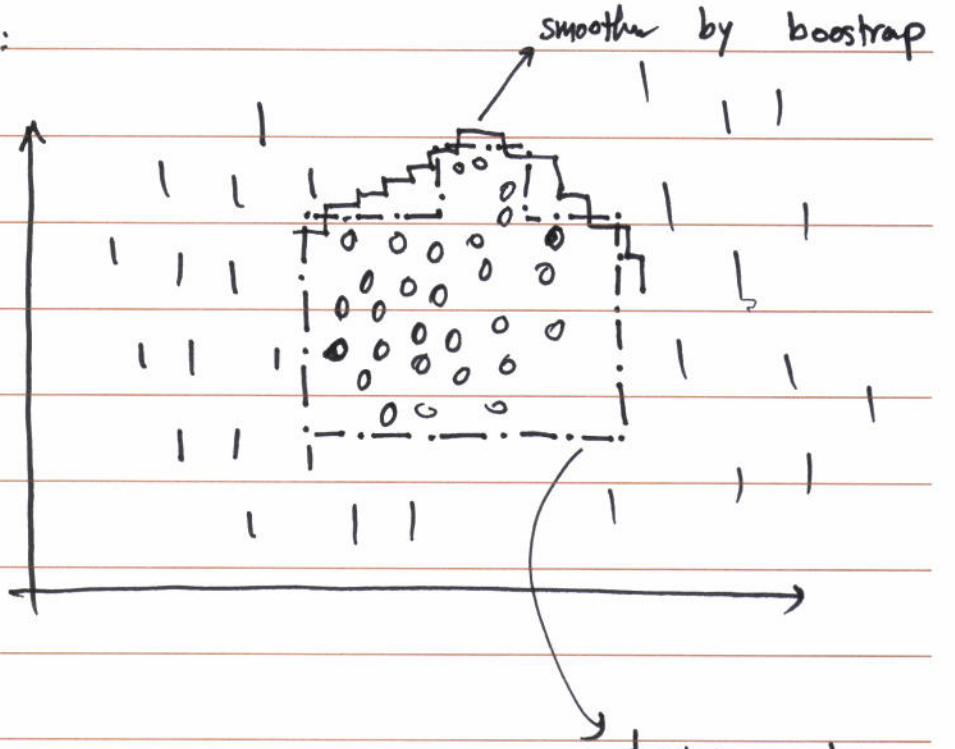




Tech. to overcome overfitting is pruning.

- If the tree is too big, the lower branches are modeling noise ⇒ overfitting
- The paradigm is to grow the tree large and then "prune" back unnecessary splits

Example:



drawback

* main → Decision tree is having high variance.

To overcome this issue, we can use Model Averaging technique.

Model Averaging

~~classifi~~ classificatia tree is simple, but produce noisy & weak classifier.

- Bagging (Breiman):

  Fit many large trees to boostrap-resampled versions of training data and classify by majority pol.

- Boosting : Fit many large/small trees to reweighted versions of the training data. Classify by weighted majority poll.

(1) Bagging :

Bagging / Boostrap aggregation averages a given procedure over many samples, to reduce variance.

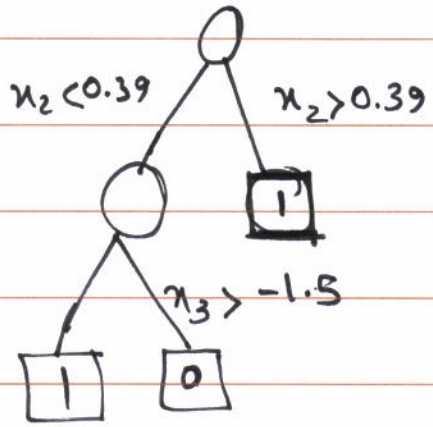
Suppose $G(X, x)$ is a classifier, such as tree producing a predicted class label at point x.

To bag G, we draw boostrap samples $X^{*1}, \ldots, X^{*B}$ each of size N with replacement from the training data. Then

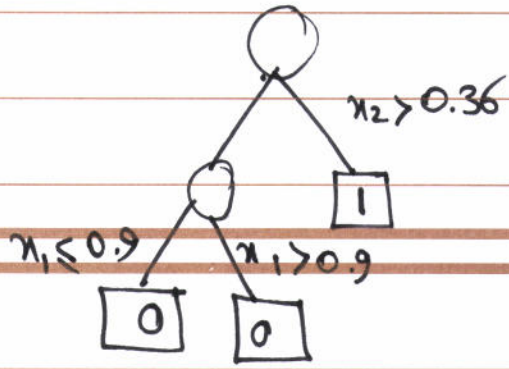
$$\hat{G}_{bag}(x) = \text{Majority Vote} \left\{ G(X^{*i}, x) \right\}_{i=1}^{B}$$

example:

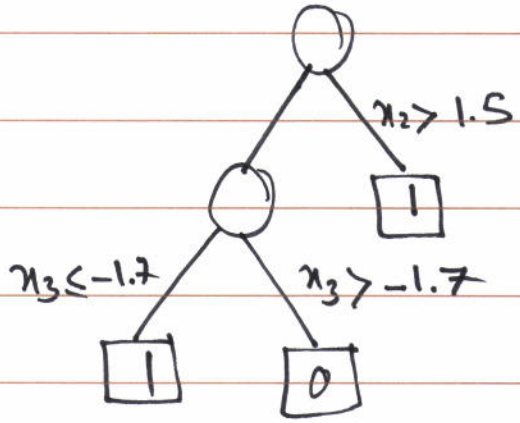
Original Tree



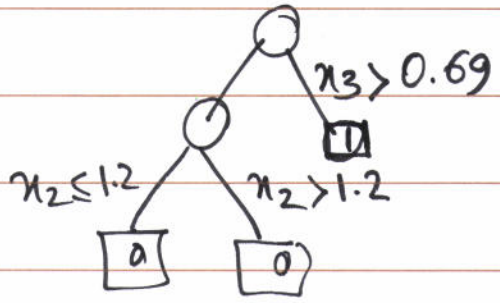
Boostrap Tree 1



Boostrap Tree 2



Boostrap Tree 3:



~~Boosting~~ Taking majority poll for final answer

⋮

Bootstrap ⟶ $G_2(x)$

Training Sample ⟶ $G_1(x)$

⟹ Final Classifier $G(x) =$ majority poll result on all models $\{G_m\}_{m=1}^{M}$

AdaBoost : (Algorithm)

1. Initialize the observation weights

$$w_i = \frac{1}{N} \qquad i = 1, \cdots, N$$

2. For $m=1$ to M repeat the following

(a) fit a classifier $G_m(x)$ to training data using weights w_i

(b) compute error

$$err_m = \frac{\sum_{i=1}^{N} w_i I_i (y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$

(c) Compute α_m

$$\alpha_m = \log \frac{1-err_m}{err_m}$$

(d) Updated weights for $i \in \{1, \ldots, N\}$

$$w_i \leftarrow w_i \left[e^{\alpha_m I(y_i \neq G_m(x_i))} \right]$$

and renormalize to w_i to sum to 1.

3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x) \right]$

Boosting

:

weighted sample 2 $\longrightarrow G_3(x)$

weighted sample 1 $\longrightarrow G_2(x)$

Training Sample $\longrightarrow G_1(x)$

Final classifier

$$G(x) = \text{Sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x) \right)$$

Random Forest :

- Grow a ~~for~~ forest of many trees (say like 500)
- Grow each tree on an independent boostrap sample from training ~~dat~~ points.

At each node :

1) Select m variables at random out of M possible variables

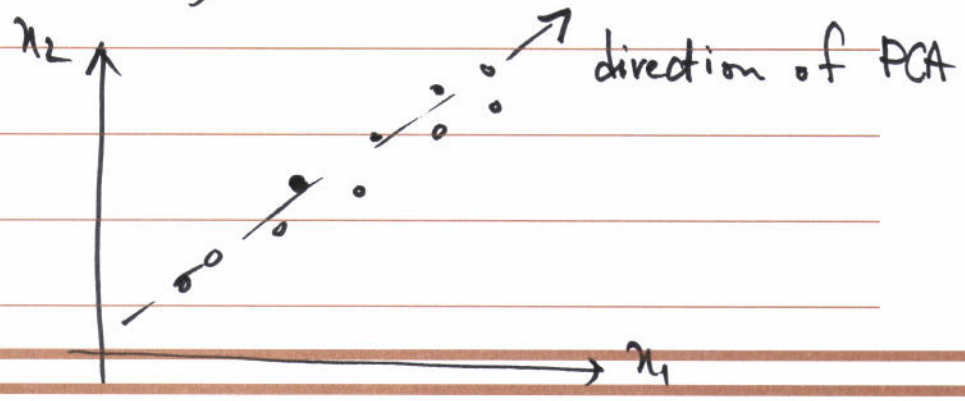
2) find the best split on the selected m variable.

- Grow the trees to maximum depth (for classification)
- Vote / average the trees to get prediction for new data point

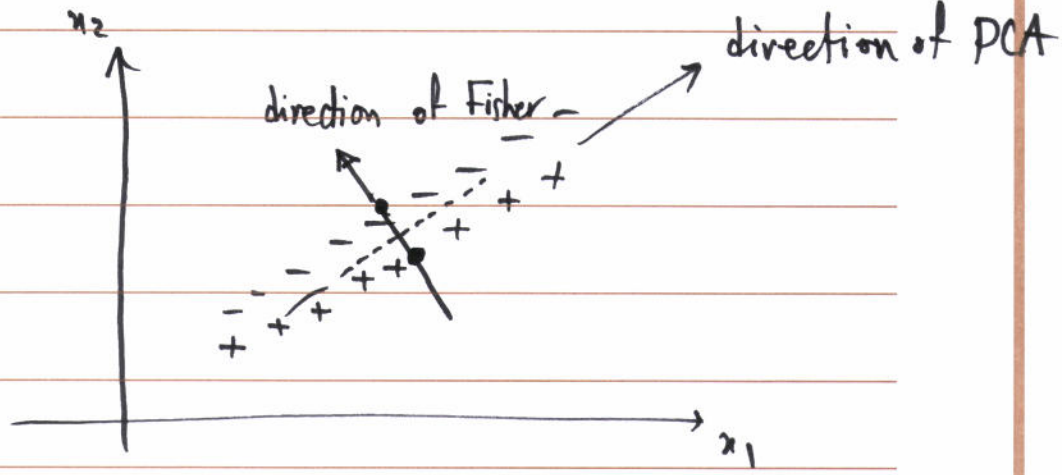
Techniques for dimension reduction:

- PCA

→ look for direction that has highest variance



- Fisher Discriminant Analysis



Techniques for overcoming overfitting

- Regularization

  → add $\lambda \|\underline{w}\|_2^2$ term to loss function

  or $\lambda \|w\|_1$

what difference they have?

if we want sparse weights, we need to use $\|w\|_1$