M10. Combined models and Ensemble methods

Manikandan Narayanan

Week 16 (Nov 10- 2025)

PRML Jul-Nov 2025 (Grads Section)

Acknowledgment of Sources

- Slides based on content from related
  - Courses:
    - IITM – Profs. Arun/Harish/Chandra's PRML offerings (slides, quizzes, notes, etc.), Prof. Ravi's "Intro to ML" slides – cited respectively as [AR], **[HR]**, [CC], [BR] in the bottom right of a slide.
    - India – NPTEL PR course by IISc Prof. PS. Sastry (slides, etc.) – cited as [PSS] in the bottom right of a slide.
  - Books:
    - PRML by Bishop. (content, figures, slides, etc.) – cited as **[CMB]**
    - Pattern Classification by Duda, Hart and Stork. (content, figures, etc.) – [DHS]
    - Mathematics for ML by Deisenroth, Faisal and Ong. (content, figures, etc.) – [DFO]
    - Information Theory, Inference and Learning Algorithms by David JC MacKay – [DJM]

Outline for Module M10

- M10. Combined models and Ensemble methods
  - **M10.0 Introduction/Motivation**
  - M10.1 Combined models
    - Conditional mixture models
    - Decision trees
  - M10.2 Ensemble methods
    - Parallel ensemble methods (bagging)
    - Sequential ensemble methods (boosting)
  - M10.3 Concluding thoughts

Machine Learning in Practice

- Real world machine learning problems rarely have a unique and single best solution.
- Multiple thought processes and teams and approaches often yield equally valid but completely different solutions.
- The set of methods for combining many such solutions (classifiers, regressors etc.) into one solution are known as “Ensemble methods”

[Mostly based on Harish’s notes/slides; HR]

The Netflix Challenge

- Data set of ~100M star ratings that ~500K users gave to ~18K movies.

- Training data: *<user, movie, date of grade, grade>*

- The grand prize of $1,000,000, to be given to a team which beat Netflix's rating prediction algorithm by 10%

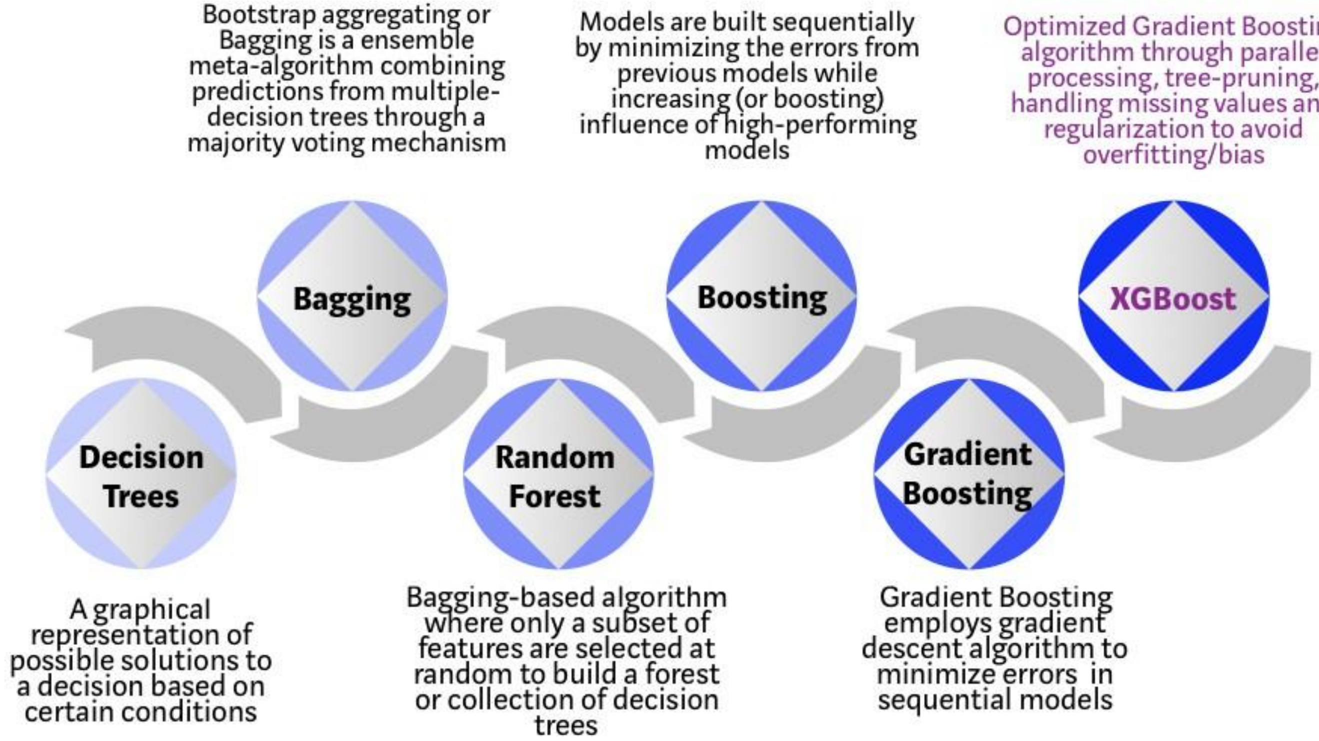
[HR]

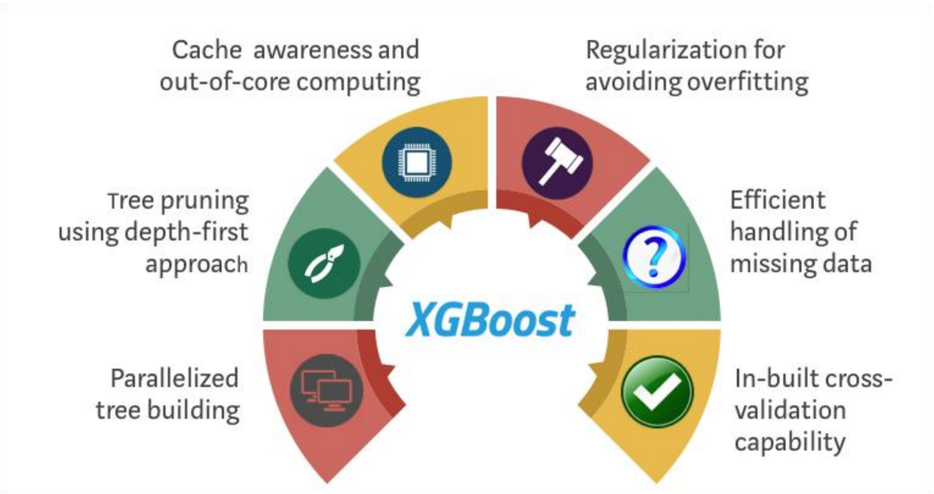
Netflix challenge: Winning Solution

- The winning team — "BellKor's Pragmatic Chaos" (itself a merger of several teams) combined a total of **107** separate prediction models!!
- The methods used various approaches — factor models, regression models, neighbourhood models, etc.
- Most other teams solutions also included large numbers of disparate models combined together.
- Ensemble methods are almost always the state of the art in any large scale Machine learning problem.

[HR]

“XGBoost Algorithm: Long May She Reign!”*



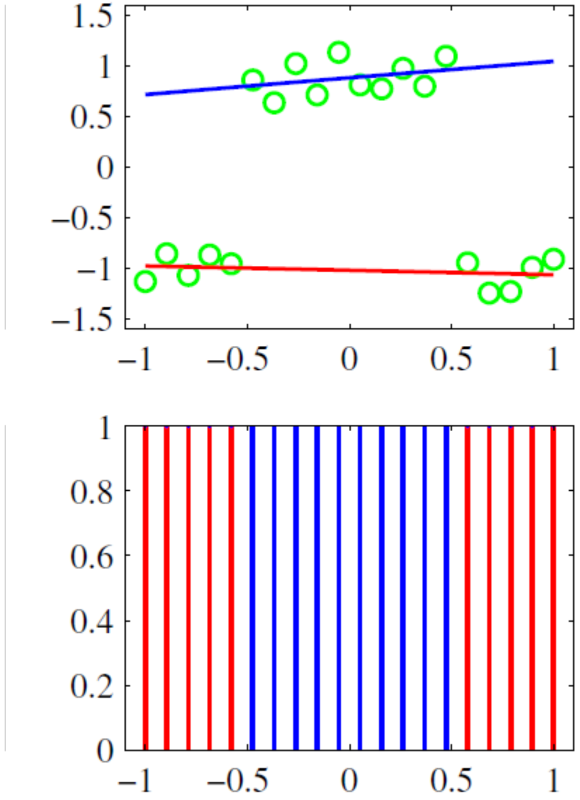


*by Vishal Morde [https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d]

Outline for Module M10

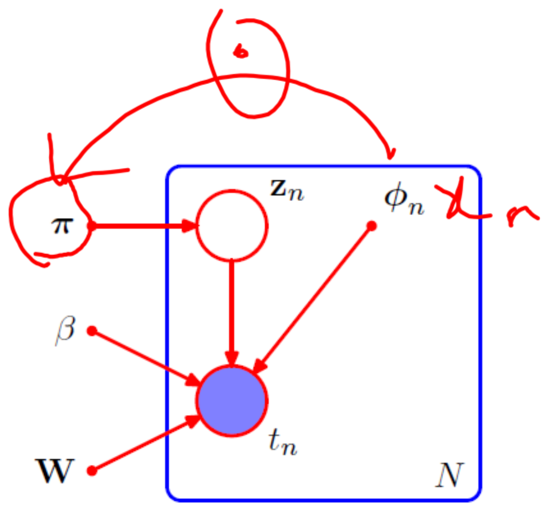
- M10. Combined models and Ensemble methods
  - M10.0 Introduction/Motivation
  - **M10.1 Combined models**
    - **Conditional mixture models**
    - Decision trees
  - M10.2 Ensemble methods
    - Parallel ensemble methods (bagging)
    - Sequential ensemble methods (boosting)
  - M10.3 Concluding thoughts

Example: Mixtures of linear regression models (mixture of experts)



[CMB]

Other conditional mixture models



- Mixture of linear classifiers (logistic regression instead of linear regression models): $p(t\,|x) = \sum_k \pi_k p_k(t|x)$

- Mixture of experts:
  - Key idea: allow mixing coefficients to be a function of the input x:

$$p(\mathbf{t}|\mathbf{x}) = \sum_{k=1}^{K} \pi_k(\mathbf{x}) p_k(\mathbf{t}|\mathbf{x}).$$

$\mathcal{N}(t|\mu_k(x), \Sigma)$ $\quad \mu_k(x) = w_k^T x$

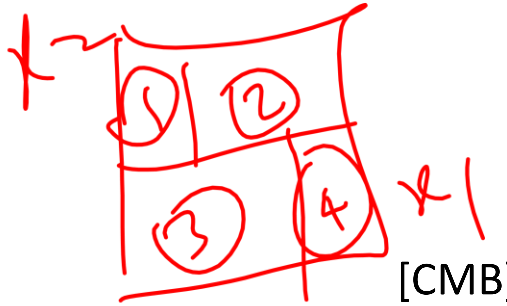
  - Compare with MDN (Mixture Density Network) where all parameters can be input-dependent!

ANN

$$p(\mathbf{t}|\mathbf{x}) = \sum_{k=1}^{K} \pi_k(\mathbf{x}) \mathcal{N}\left(\mathbf{t}|\boldsymbol{\mu}_k(\mathbf{x}), \sigma_k^2(\mathbf{x})\right).$$

x $\quad \theta(x)$

  - Hierarchical mixture of experts also possible, though we will look at **decision trees** as its hard (non-probab.) version of combining different models:



[CMB]

Outline for Module M10

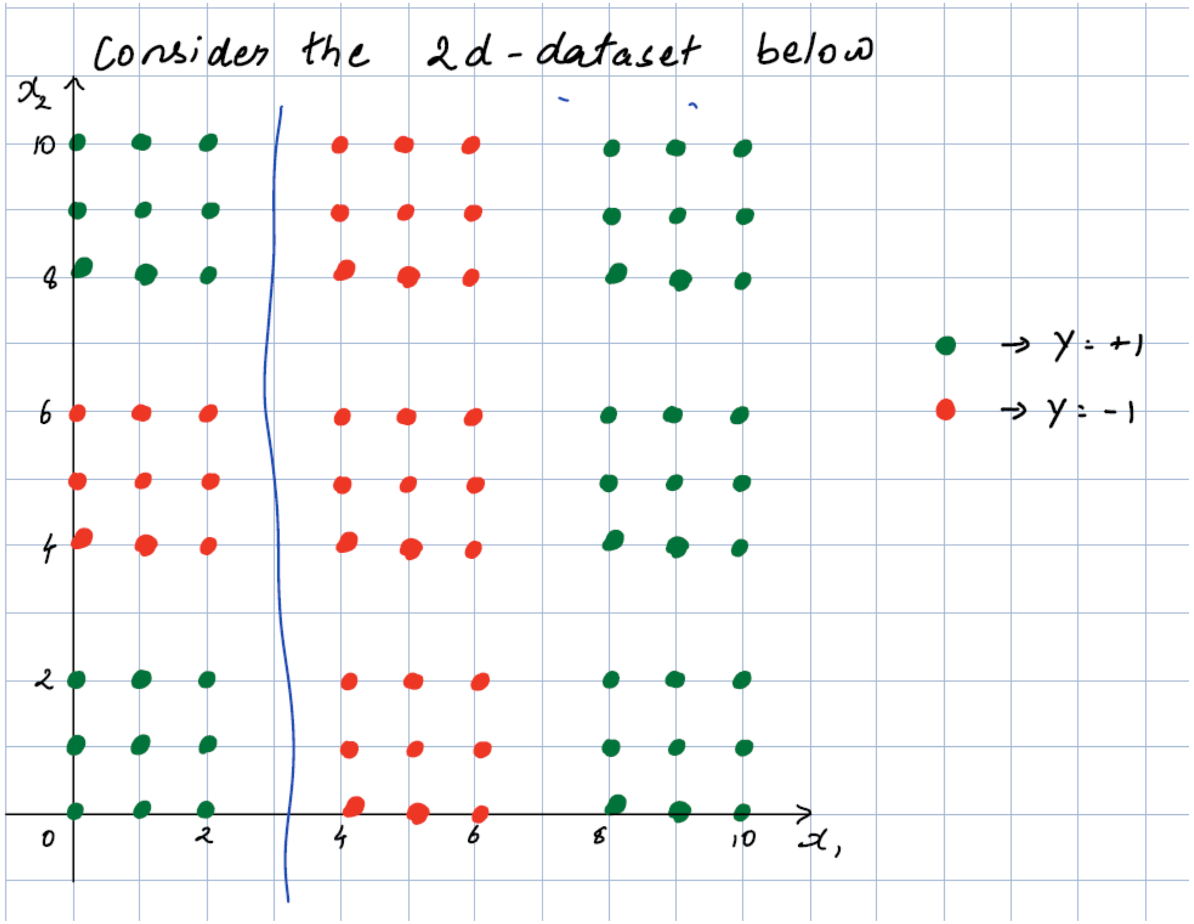
- M10. Combined models and Ensemble methods
  - M10.0 Introduction/Motivation
  - **M10.1 Combined models**
    - Conditional mixture models
    - **Decision trees**
  - M10.2 Ensemble methods
    - Parallel ensemble methods (bagging)
    - Sequential ensemble methods (boosting)
  - M10.3 Concluding thoughts

A visual understanding of decision trees

- http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

Learning a decision tree – toy example

Consider the 2d-dataset below



1.) What should the Root node be?

· Evaluate all classifiers of the form on "Entire" Training.

$$h(x) = \begin{cases} +1 & \text{if } x_1 \geq a \\ -1 & \text{if } x_1 < a \end{cases}$$

and

$$h(x) = \begin{cases} +1 & \text{if } x_2 \geq a \\ -1 & \text{if } x_2 < a \end{cases}$$

and their negations.

Toy example – evaluating all "feature X threshold" combinations at the root node!

Accuracy

For simplicity we will evaluate only 4 such classifiers:

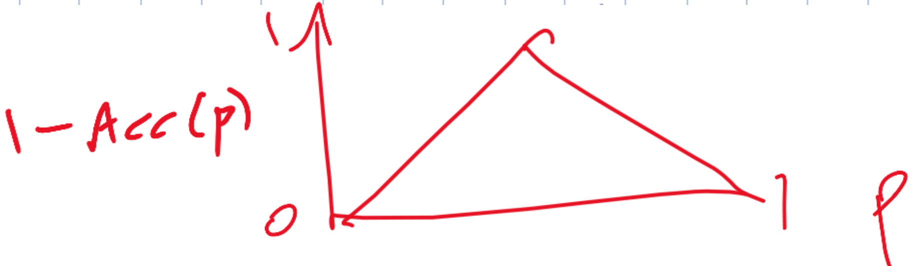
(a) $h(x) = +1$ if $x_1 \geq 7$
$\quad = -1$ if $x_1 < 7$

$$\text{Accuracy} = \frac{3+4}{9} = \frac{7}{9}$$

(a') $h(x) = -1$ if $x_1 \geq 7$
$\quad = +1$ $\quad x_1 \leq 7$

$$\text{Accuracy} = \frac{0+2}{9} = \frac{2}{9}$$

Let $L : x_1 < 7$
$\quad R : x_1 \geq 7$



Entropy

Define $H(p) = p \log \frac{1}{p} + (1-p) \log\left(\frac{1}{1-p}\right)$. (logarithm base is 2)

Avg. Entropy of split (a) $= P_L H_L + P_R H_R$.

P_L : Fraction of points on the left. e.g. above $P_L = \frac{54}{81}$

$H_L = H(q_L)$ Where q_L = Fraction of positive points on the left. e.g. above $q_L = \frac{18}{54}$

$\therefore$ Entropy of split (a) above

$$= \frac{54}{81} H\left(\frac{18}{54}\right) + \frac{27}{81} H(1)$$

$$= \frac{2}{3} H\left(\frac{1}{3}\right) + \frac{1}{3} H(1) = \frac{2}{3} H\left(\frac{1}{3}\right)$$

$$= \frac{2}{3}\left(\frac{1}{3}\log 3 + \frac{2}{3}\log\frac{3}{2}\right) = \frac{2}{3}\left(\log 3 - \frac{2}{3}\right)$$

[HR]

Exercise: try for 3 other splits (& their negations) to find the best root node split.

- E.g.,

(b) $h(x) = +1$ if $x_1 \geq 3$
$= -1$ if $x_1 < 3$

Accuracy $= \frac{3+1}{9} = \frac{4}{9}$

Ill'ly accuracy of (b') which is the negation of above $= 1 - \frac{4}{9} = \frac{5}{9}$

L: $x_1 < 3$
R: $x_1 \geq 3$

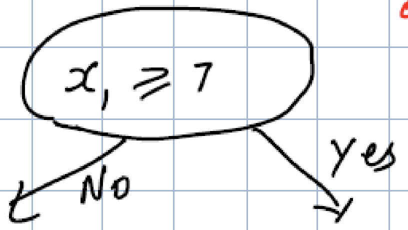
Entropy of split b : $P_L H_L + P_R H_R$

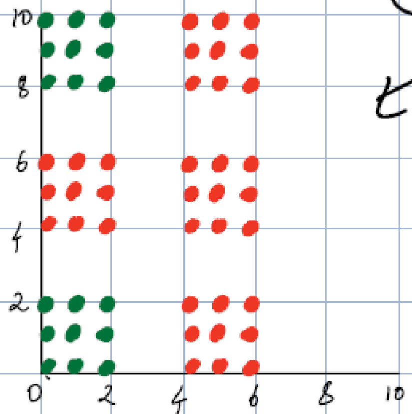
$P_L = \frac{1}{3}$ $\quad q_L = \frac{2}{3}$ $\quad$: $\quad$ Entropy : $\frac{1}{3} H\left(\frac{2}{3}\right) + \frac{2}{3} H\left(\frac{1}{2}\right)$

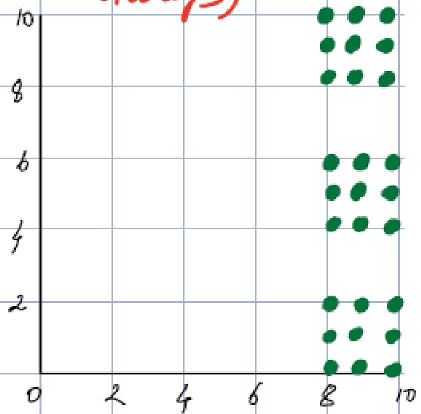
$P_R = \frac{2}{3}$ $\quad q_R = \frac{1}{2}$

Clearly Option (a) is best for root node:

(In this case using either max accuracy or min entropy gives same answer. But not always)





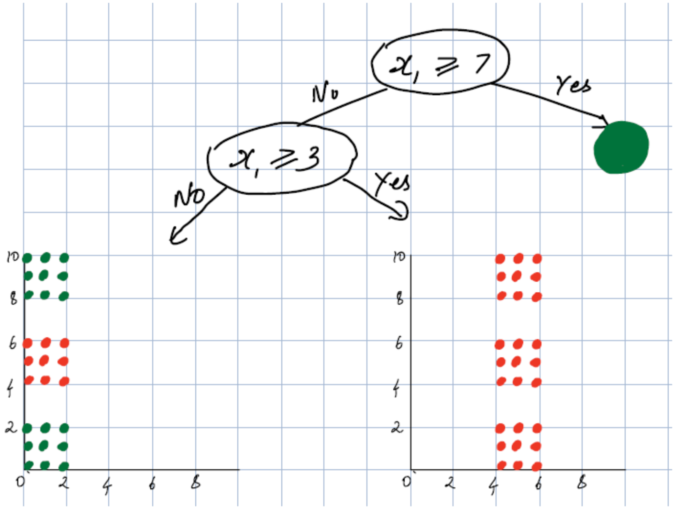


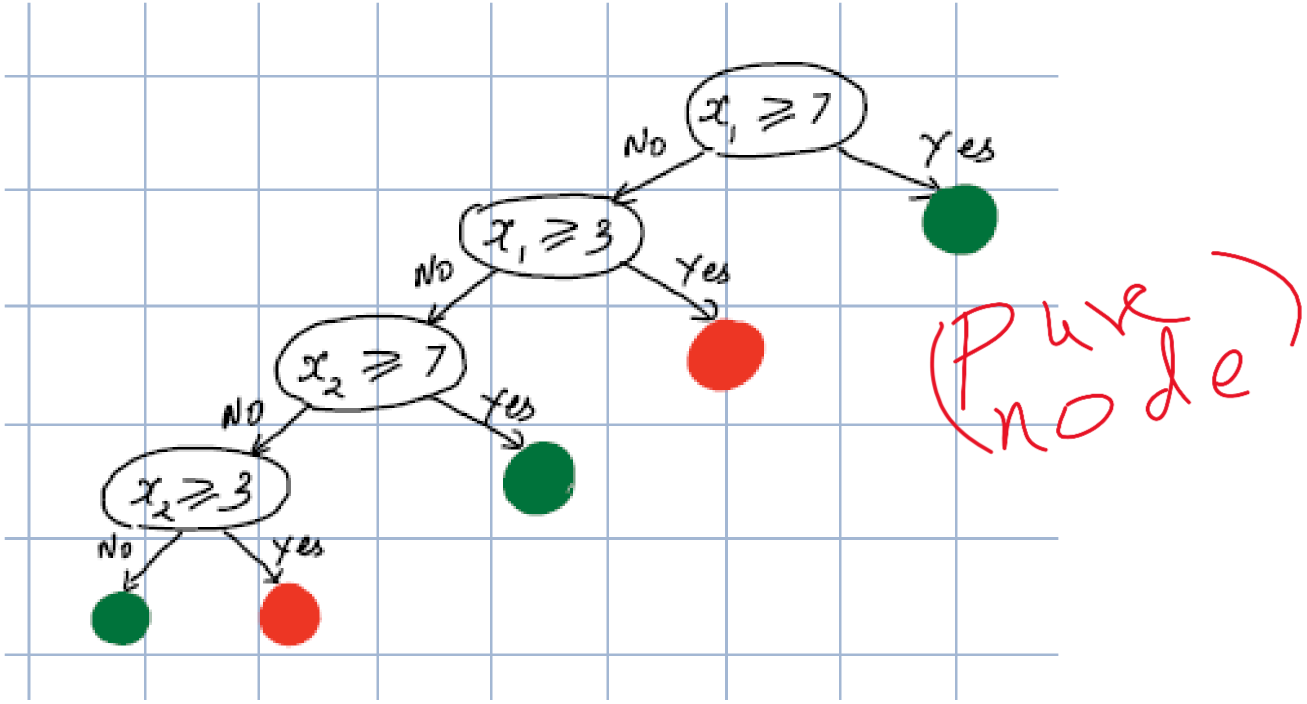
Recurse

Stop

[HR]

Exercise: Recursively solve subproblems on left and right to derive this final tree





final tree

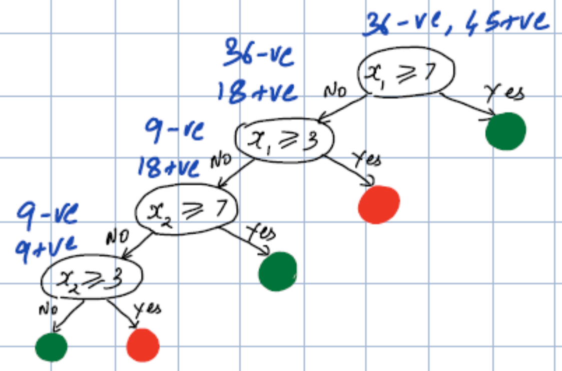
[HR]

Regularization of a decision tree

- Goal: Bring down the # of nodes in the decision tree without losing too much on accuracy.

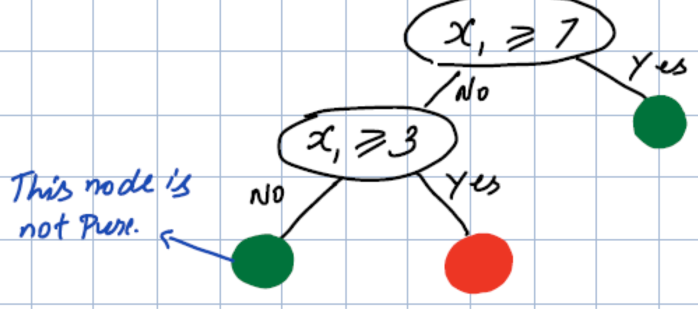
- Solution: Stop early…
  - i) …at a certain depth of the tree, or
  - ii) …when number of training points is less than some number.

preferred



For example if we stopped splitting at 30 ~~nodes~~ training points we would not have $x_2 \geq 7$ node.

The final tree would have been:



[HR]

Outline for Module M10

- M10. Combined models and Ensemble methods
    - M10.0 Introduction/Motivation
    - M10.1 Combined models
        - Conditional mixture models
        - Decision trees
    - **M10.2 Ensemble methods or Committee models**
        - **Parallel ensemble methods (bagging)**
        - **Sequential ensemble methods (boosting)**
    - M10.3 Concluding thoughts

Committees

- Committees are ensemble methods that average the predictions of many individual learners

- Two very different approaches:
  - Bagging – average of **parallel**y/separately-trained **high-capacity** learners
  - Boosting – average of **sequential**ly/adaptively-trained **weak** learners

- Bias-variance decomposition helps in understanding certain aspects of bagging, and computational/statistical learning theory helps derive certain performance bounds on boosting.

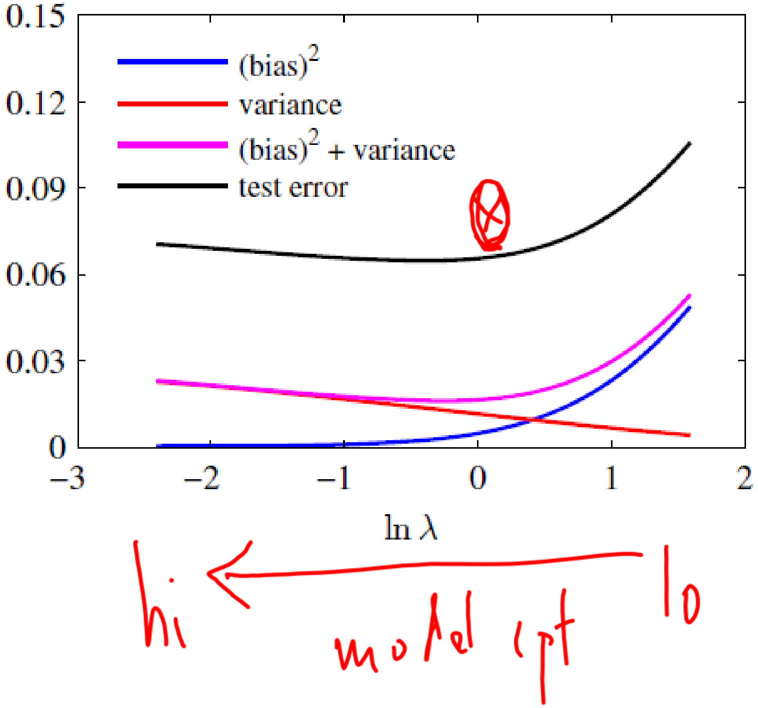
[CMB]

Recall: Bias-variance analysis summary

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

where

$$\begin{aligned}
(\text{bias})^2 &= \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
\text{variance} &= \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
\text{noise} &= \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t
\end{aligned}$$



[CMB]

Recall: Bias-variance in pictures

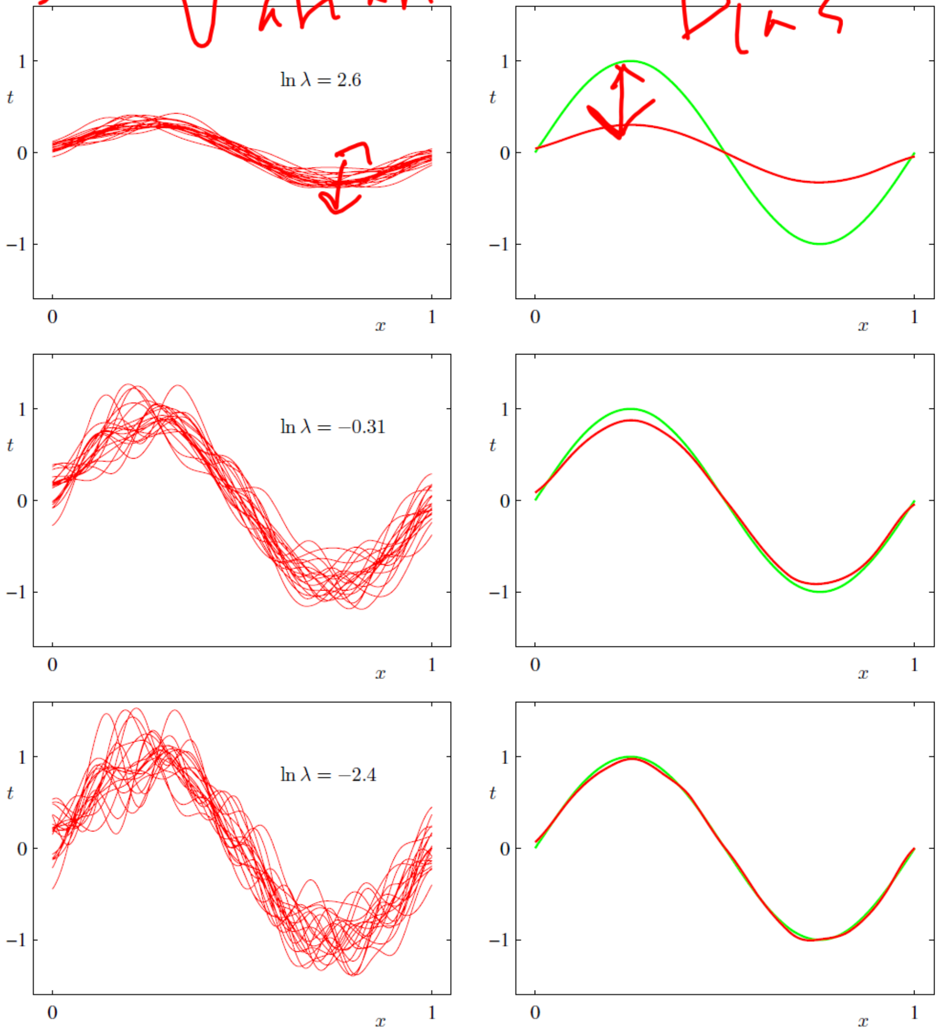
cplxty

Variance Bias

lo

int.

hi



[CMB]

Switch to Harish’s slides on Ensemble methods

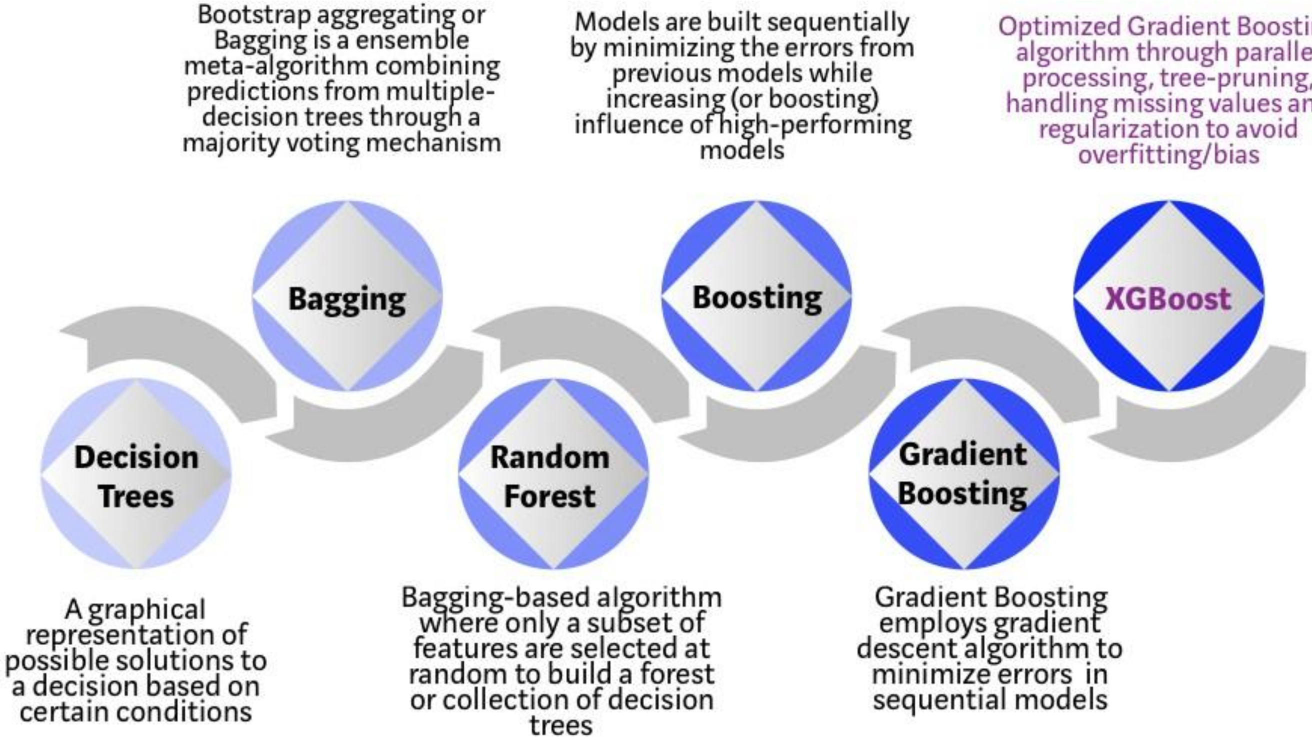
[https://drive.google.com/file/d/1KUy2mziZ1pz-A0NeqlAV_qTP5Rq47Kkt/view?usp=sharing]

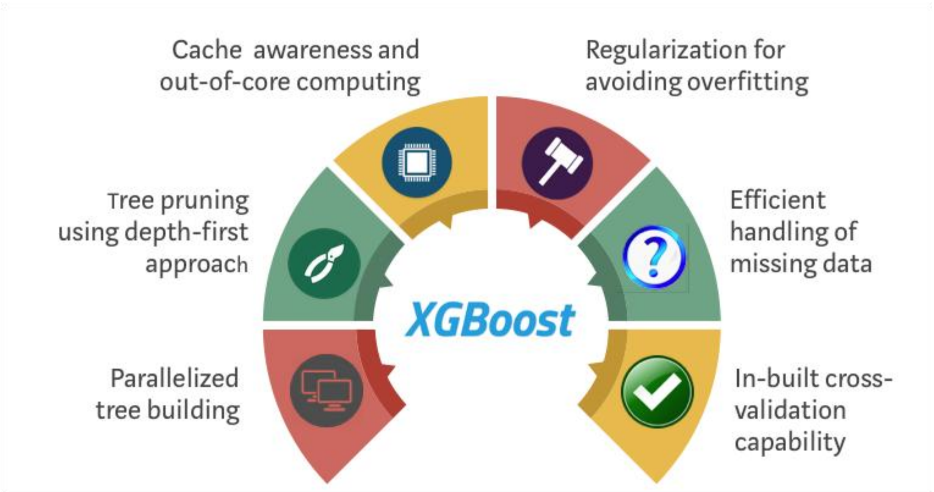
See annotated version of above called “N11_ensemble_methods.ExtraNotes.pdf” in moodle.

Outline for Module M10

- M10. Combined models and Ensemble methods
  - M10.0 Introduction/Motivation
  - M10.1 Combined models
    - Conditional mixture models
    - Decision trees
  - M10.2 Ensemble methods
    - Parallel ensemble methods (bagging)
    - Sequential ensemble methods (boosting)
  - **M10.3 Concluding thoughts**

“XGBoost Algorithm: Long May She Reign!”*





*by Vishal Morde [https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d]

Concluding thoughts & next steps

Supervised learning: Fun starts when you can take the different classification or regression models you've learned from this course, and apply your knowledge to choose the right model or right method of combining models in a systematic rather than brute-force fashion!

- Unified view of different models helps towards above goal – fixed vs. selective (SVM) vs. adaptive (ANN) basis functions; loss fn. view of different classifiers, etc.
- Understanding conceptual/mathematical foundations of different methods – beyond popular "blog" descriptions -- also helps towards above goal.

similar also applies for **Unsupervised learning:** spectral clustering and dimensionality reduction (PCA) both involved understanding the eigenspectrum of a matrix (optimizing $u^T A u$).

Thank you!