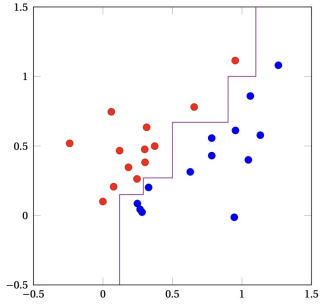
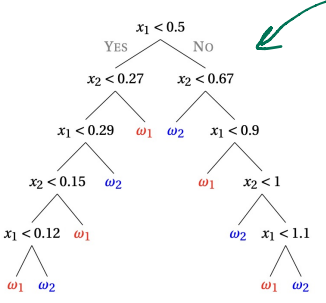
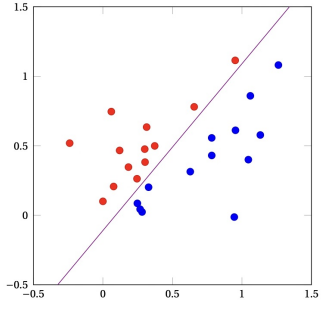
Decision Tree Variations

- Can use multivariate splits (more than 1 feature)
  - non-axis-aligned splits w/ other classification algorithms or by generating them randomly.

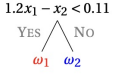




axis-aligned splits



linear split, not axis aligned



both features at the same time

↳ find split w/ SVM or other classifier.

Figure 15.1: Comparison between a vanilla decision tree and a decision tree with a multivariate split

decision tree w/ linear classifier → nonlinear decision boundaries

nonlinear classifier → piece-wise linear boundaries

may gain better classifier at cost of:
- worse interpretability (e.g. using all features at node)
- speed

Usually, decision trees very fast
- looking at less features as you go

most decision trees in practice only use 1 feature at a time
↳ assumed on final exam

not so common

Solution: at each node, look at at most $\sqrt{p}$ features
- limit # features per split: forward stepwise selection or Lasso

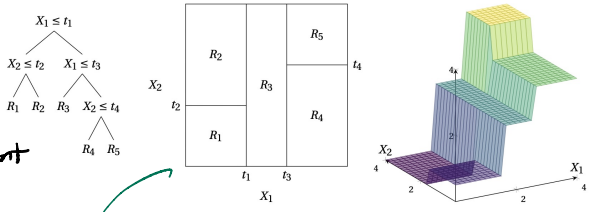


Figure 15.2: Example of regression using a decision tree

Decision Tree Regression:

- creates a decision fn. that's piecewise constant
- each leaf stores a label $\mu_S = \frac{1}{|S|} \sum_{i \in S} y_i$

average of all labels in index set

constant within each rectangle

Cost: $J(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \mu_S)^2$

↳ variance of all labels of pts. in node

$\mathrm{Var}(\{y_i : i \in S\})$

choose split that minimizes this

* entropy for classification, not so good w/ regression (continuously-valued labels)

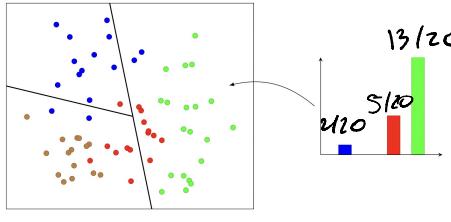
If all training pts are identical, cost = 0, positive o/w

Stopping Early

- don't always have to keep going until every leaf is pure (i.e. only contains 1 class)

Why stop early?

- limit tree depth (for speed)
- limit tree size (for big datasets)
- complete tree more prone to overfitting (trying to classify every single pt.)
  - data has noise or isn't separable/overlapping distribution
    - purity of leaves is counterproductive in this case (better to estimate posteriors)



posterior probabilities

Figure 15.3: Example of a decision tree that has been stopped early, with the corresponding posterior probability distribution for one leaf

- leaves w/ multiple points, return:
  - (classification) majority vote
  - (classification) posterior probabilities for each class
  - an average (regression)

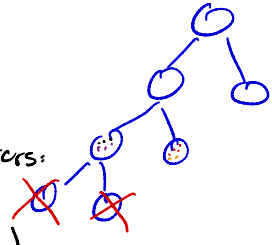
common for split to have no info. gain, but next split makes much more progress.

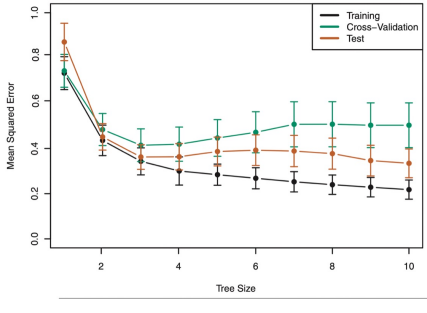
When do we stop decision tree?

- when split doesn't reduce entropy/error enough (this is dangerous, pruning better)
- when most of node's points (e.g. 95%) have the same class (prevent overfitting)
- when nodes contain too few pts. (e.g. ≤ 10)
- when cell's edges are too tiny
- when depth is too large (risky if still lots of pts. in a cell)
- use validation to compare errors
  - ↳ slowest but most effective way to know when to stop
  - ↳ to avoid overfitting - grow tree large
    - use validation to prune tree back.

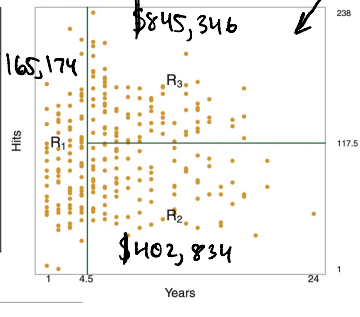
Pruning: grow tree too large then greedily remove each split whose removal improves validation

- empirically more reliable than stopping early
- don't prune node w/ no children
  - ↳ work bottom-up
- when comparing validation errors, compare 2 errors:
  - errors of two children
  - error of parent (if we prune children)





3 nodes



* decision trees are fast, simple, interpretable, easy to explain, invariant under scaling/translation, robust to bad features

- but they're also not the best at prediction (compared to prev. methods)
- also high variance (depending on splits)

Ensemble Learning: combine diff. algorithms to compute collective answer
idea: reduce variance by taking the avg. of a bunch of decision trees

Weak learning algo: does better than guessing

→ take avg. output of:
- different algorithms
- same learning algorithm on different training sets (more common)

most common:
- **bagging**: same learning algo. on random subsets of one training set (if low data)
- **random forest**: random decision trees on random subsamples of data

* averaging isn't specific to decision trees but it works particularly well for trees

for regression algos: median or mean output
for classification: majority vote or average posterior probabilities

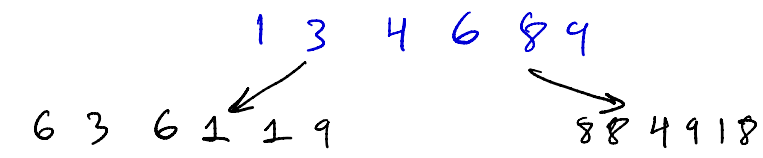
- use learners w/ low bias (e.g. deep decision trees)
  - averaging reduces variance, not necessarily bias
- high variance & some overfitting okay since averaging reduces variance
- averaging sometimes reduces bias & increases flexibility, but not reliably

e.g. averaging nonlinear decision boundaries from linear classifiers

- hyperparameter settings usually different than those for 1 learner
- # trees is a hyperparameter
  ↳ variance vs. runtime tradeoff

Bagging: randomized method for creating different learners from same dataset
- works well w/ many diff. learning algorithms (except k-nearest neighbors)

given n-point training sample, generate random subsample of size n' by Sampling w/ replacement
↳ some samples chosen multiple times
↳ some won't be chosen at all



- If $n' = n$, ~63.2% are chosen

Build learner. Points chosen j times have greater weight in learning algorithm
- decision trees: j-time pt. has j x weight in entropy
- SVMs: pt. incurs j x penalty if it violates the margin (slack var.)
- regression: pt. incurs j x loss

repeat until T learners made

Metalearner: takes test pt., feeds it into all T classifiers, returns average/majority output of learners

Random Forests: random sampling w/ random classifiers
- random sampling isn't random enough

- one really strong predictor → same feature split at top of every tree
- even more randomness → different splits, especially at root node

Idea: - Choose random sample m of d features
- choose best split from m features
- * different random sample of m features at each tree node
  $m \approx \sqrt{d}$ good for classification m is a hyperparam
  $m \approx d/3$ good for regression

Smaller m → more randomness in trees, less tree correlation, more bias

Why does this work well?
- if ∃ really strong predictor, only a fraction of trees can choose that predictor ($^m/_d$)
  ↳ decorrelates trees, taking avg → less variance than single tree

* averaging works best when we have very strong learners that are also diverse
  ↳ be careful not to dumb down trees too much in order to decorrelate

disadvantage of random forest: - slow
- loses interpretability/inference

advantage: - more accurate than a single tree