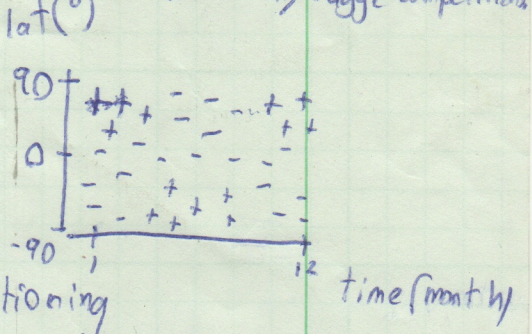
Decision Trees + Ensemble Methods

Decision Trees are good to use with ensembles. Tree ensembles win many Kaggle competitions

Decision Trees



Non-linear (vs. SVMs)

Ski example - linear model bad
- Split into regions. How?

Generally intractable

Picking Regions

Greedy, Top-Down, Recursive Partitioning

- 20 questions, draw out tree, branch vs. leaf
- Formally, given node p (parent node), covering region R_p, can define a split s_p on j-th feature with threshold t as

$$s_p(j, t) = \left(\underbrace{\{X \mid X_j < t, X \in R_p\}}_{R_1}, \underbrace{\{X \mid X_j \geq t, X \in R_p\}}_{R_2} \right)$$

How to choose splits?

- Intuitively, trying to isolate + and -
- Useful to define $L(R)$ on region
- For now define $L(R)$ as misclassification loss

  Given C total classes, define $\hat{p}_c$ as proportion of examples in R that are of class c

  $$L_{misclass} = 1 - \max_c (\hat{p}_c)$$

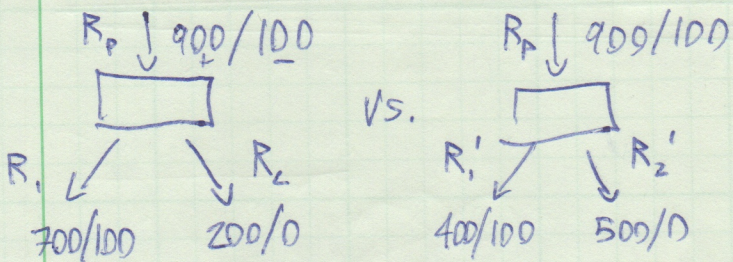
- Loss of parent node is: $L(R_p)$

  of children: $L(R_1) + L(R_2)$

- Want to maximize reduction in loss $\left(L(R_p) - [L(R_1) + L(R_2)] \right)$

  ↳ choose j, t for s_p to do ≥ 0

(**) Misclassification Loss Has Problems



$L(R_1)+L(R_2) = 100$

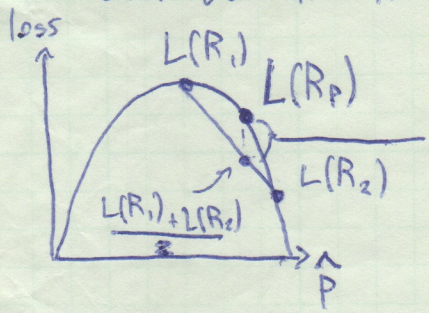
$L(R_1')+L(R_2') = 100$

In fact, $L(R_p) = 100$!

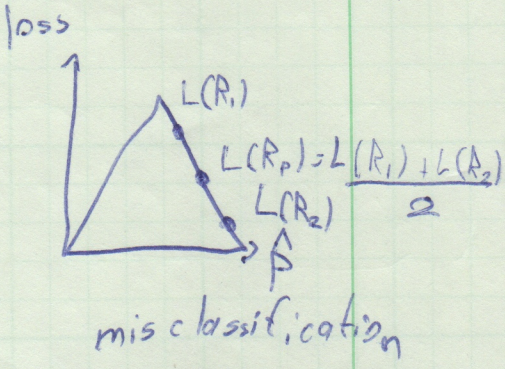
(*) Instead, define cross-entropy loss

$$L_{cross} = -\sum_c \hat{p}_c \log_2 \hat{p}_c \quad (\hat{p}\log_2\hat{p} \equiv 0 \text{ if } \hat{p}=0)$$

Number of bits needed to specify outcome given distribution is known.



Change in loss (Information Gain in cross-entropy terms)



Gini loss has similar shape $L_{gini} = \sum_c \hat{p}_c(1-\hat{p}_c)$

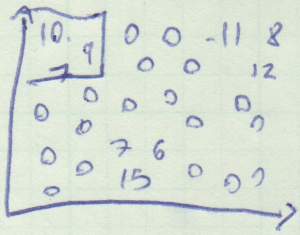
Now, let's cover some extensions of DTs...

(**) Regression Trees

Instead of predicting majority class, predict mean of values in R_m : $\hat{y}_m = \frac{\sum_{i \in R_m} y_i}{|R_m|}$

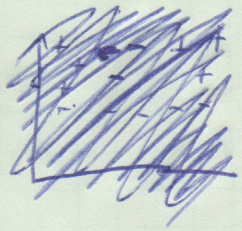
(*) Minimize squared loss

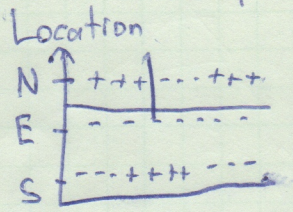
$$L_{squared} = \frac{\sum_{i \in R_m} (y_i - \hat{y}_m)^2}{|R_m|}$$

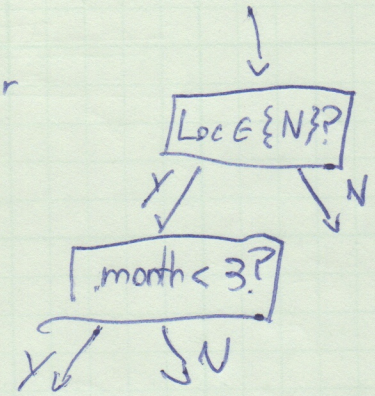


Categorical Variables

North, South Hemisphere, Equator







- For q categories, 2^q splits possible so does not scale well, except for binary classification case.

Regularization

- Decision trees if fully grown are high variance, low bias.

Use heuristics for regularization

1) Min leaf size
2) Max depth
3) Max # nodes
4) Min decrease in loss → dangerous since might be higher order interactions!
5) Pruning → better use validation set, measure misclassification or squared error

Runtime - Binary Classification

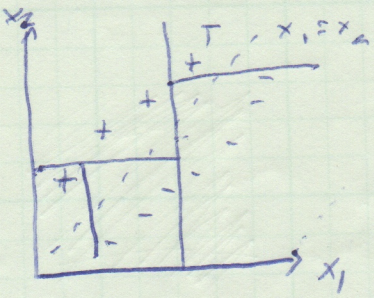
n examples, f features, d depth

- Test time is $O(d)$, if balanced $O(\log n)$

- Train time, each point in $O(d)$ nodes, costs $O(f)$ at each node

So $O(nfd)$ nf is size of design matrix and d often $\log n$

No additive structure



Linear models on other hand are great at additive structure!

Recap

⊕ Easy to explain
⊕ Interpretable
⊕ Categorical Var
⊕ Fast

⊖ High Variance
⊖ Bad at Additive
⊖ Low Predictive Accuracy

Will solve these issues via ... ensembling!

Ensembling

Take independent, identically distributed (iid) random variables (RV) X_i. Say $Var(X_i) = \sigma^2$

$$Var(\bar{X}) = Var\left(\frac{1}{n}\sum X_i\right) = \frac{\sigma^2}{n}$$

Now drop independence assumption, so only id,

Say X_i's are correlated by ρ

Then $Var(\bar{X}) = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2$ as $n \to \infty$ $Var(\bar{X}) = \rho\sigma^2$

Ways to ensemble

1) Use different algorithms
2) Use different traing sets
3) Bagging (RF)
4) Boosting (Adaboost, xgboost)

Bagging

Stands for bootstrap aggregation, variance reduction method

- Bootstrap - method from statistics to measure uncertainty
- Have true population P, which training set S was drawn from. Write $S \sim P$.

Ideally, have $S_1, S_2, \ldots$ all drawn from P.

Now, assume $S = P$.

- Can now draw new samples Z from S! $Z \sim S$

(sample w/ replacement, $|Z| = |S|$) $Z_1, Z_2, \ldots Z_M$

- Train model G_m on Z_m. Can then look at variability in predictions

However, can also define new aggregate predictor

$$G(x) = \sum_m \frac{G_m(x)}{M}$$

Bias-Variance Analysis

- Recall $\text{Var}(\bar{X}) = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2$ for M predictors.
- Bagging creates less correlated predictors, $\downarrow \rho$, thereby reducing Variance.

Note that Bias increases a bit due to reduced sampling), but $\downarrow$ Var typically outweighs $\uparrow$ Bias

Note too that $\uparrow M$ can't ~~hurt~~ make overfit more, since just decrease Var

Decision Trees + Bagging

Recall DT are high variance, low bias → ideal fit for variance reduction of bagging!

Note bagging has another benefit for DT:

If a feature is missing, simply don't use trees in ensemble that contain that feature!

Missing feat → Ignore Gm's splitting on it!

Out-of-bag estimation

Additional benefit of bagging: free validation set.

On average, Z will contain 2/3 of S. Use other 1/3 to estimate error, call OOB error. In limit as $M \to \infty$ OOB gives equivalent results to LOOCV.

Variable Importance Measure

Do lose some interpretability.

For each feature, find each split that uses it in the ensemble. Measure decrease in loss, average.

Note doesn't measure degradation in perf if didn't have feature, since other features might be able to substitute.

Random Forest

Intuition: one very strong predictor → correlated trees

Instead, at each split only allow subset of features to be used (ex: $u = \sqrt{f}$). Decreases Var, slight increase in Bias

Again $\text{Var}(\bar{x}) = p\sigma^2 + \frac{1-p}{M}\sigma^2$ as $\downarrow p \Rightarrow \downarrow \text{Var}$ (assuming M large)

Just what RF does! Works even better for missing values..

Recap - Bagging

~~Bagging ↓ Var, ↑ Bias, tends for ~~

⊕ ↓ Var (even more so for RF)

⊕ Better accuracy

⊕ Deal with missing values

⊖ ↑ bias (even more so for RF)

⊖ Harder to interpret

⊖ Still not additive

⊖ More expensive

Boosting

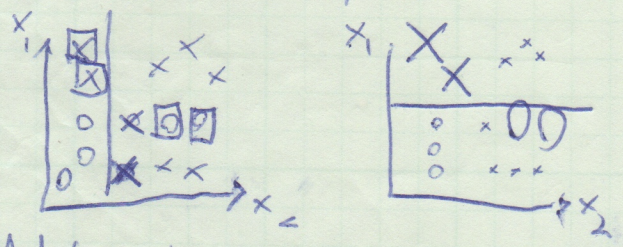
Bagging was variance reducing, boosting bias reducing

~~Boosting additive~~

Since reducing bias, want high bias, low variance models, weak learners

In terms of DT, just use decision stumps

Intuition via Example



Boost weight of negatives

Adaboost

Training data $(x_1, y_1) \dots (x_N, y_N)$

1) Set $w_i = 1/N$ for $i = 1 \dots N$

2) for $m = 1 \dots M$:

a) Fit weak classifier G_m to weighted training data

b) Compute weighted $err_m = \frac{\sum_i w_i I(y_i \neq G_m(x_i))}{\sum w_i}$

c) Compute weight $\alpha_m = \log\left(\frac{1 - err_m}{err_m}\right)$

d) set $w_i = w_i \cdot \exp\left[\alpha_m I(y_i \neq G_m(x_i))\right]$

3) Return $f(x) = \text{sign}\left[\sum_m \alpha_m G_m\right]$

Adaboost - cont'd

Additive - combine together prediction of many weak models.
Oftentimes beats single strong model.
No longer independent of previous models in sequence
↳ can overfit.

More general framework: Forward Stagewise Additive Modeling (FSAM)

Though derived on its own, Adaboost can be thought of as special case of FSAM.

Still want to create ensemble classifier f_M

1) Initialize $f_0(x) = 0$
2) For $m = 1 \ldots M$
   a) Compute $(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$
   b) Set $f_m(x) = f_{m-1} + \beta_m b(x; \gamma_m)$
3) Return $f(x) = f_m(x)$

- Adaboost is FSAM with exponential loss and 2-class classification $L(y, f(x)) = \exp(-y f(x))$. Proof in ESL.
- For squared loss for regression, FSAM is same as fitting individual tree to residual $y_i - f_{m-1}(x_i)$ $L(y, f(x)) = (y - f(x))^2$

Gradient Boosting

basis of xgboost

For more general losses, can't always write out nice closed-form solution to minimization problem. Turn to numerical optimization.
One way:
Take derivative, do gradient descent.
But restricted to taking steps that are in model class.

Compute gradient at each training point i wrt current predictor f

(*) $$g_i = -\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}$$

New predictor simply tries to train regression tree to match this gradient. In FSAM:

(*) $\gamma = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} (g_i - b(x_i; \gamma))^2$. Add in, is your gradient step.

(*)(*) Recap - Boosting

⊕ ↓ bias
⊕ Very good accuracy
⊖ ↑ var.
⊖ Prone to overfitting