RF out-of-bag samples

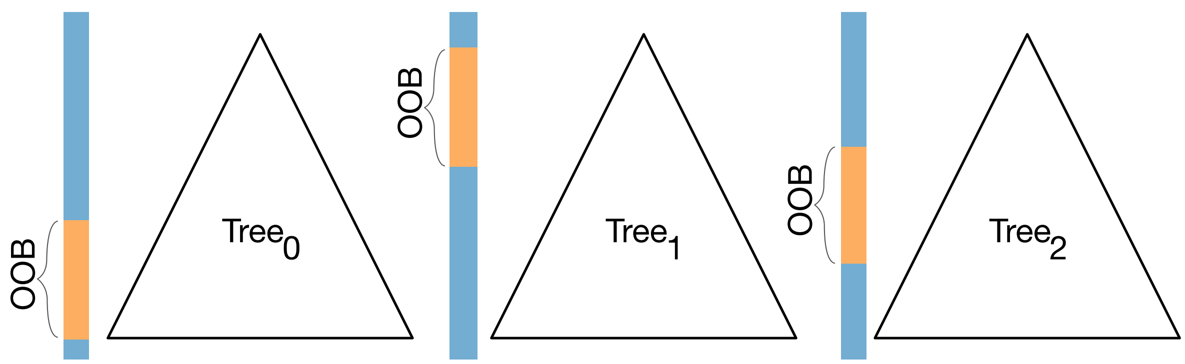
Validation sets for free!

Terence Parr
MSDS program
University of San Francisco

UNIVERSITY OF SAN FRANCISCO

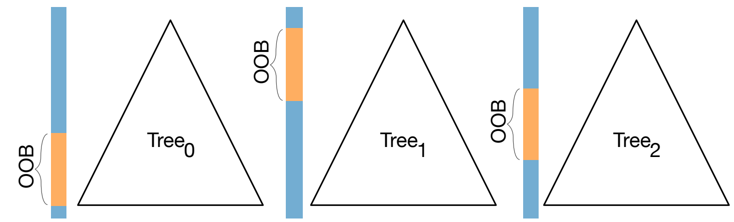
RF's have built-in out-of-bag validation set

- RFs have a major advantage over other models: OOB metrics
- Each tree is trained on 63% of data, leaving 37% OOB
- OOB record subsets available to each tree is different
- It’s an excellent estimate of the validation error
- Stick with OOB unless default sklearn score() is not suitable
- Not having to process training and validation sets separately is a huge productivity win (assuming significant feature engineering)

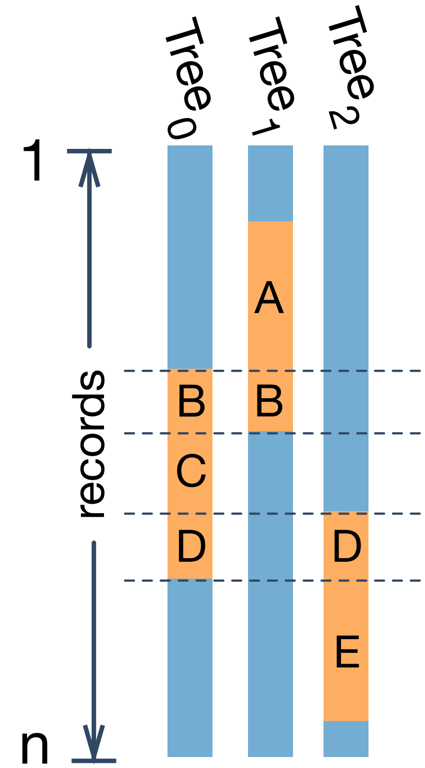


UNIVERSITY OF SAN FRANCISCO

Computing OOB predictions



- Get $\hat{y}^{(i)}$ by averaging estimates from trees not trained with $(x^{(i)}, y^{(i)})$
  - Image to right; blue is training set, OOB orange
  - Trees from same labeled OOB region of $x^{(i)}$ used to get $\hat{y}^{(i)}$
  - Must find all trees not trained on $x^{(i)}$
  - E.g., compute $\hat{y}^{(i)}$ for **B** region using Trees 0, 1 but not 2
  - No OOB error estimate is possible for unlabeled regions
- Do not make predictions for OOB and compute error per tree!
- Each tree has high bias, so OOB scores from one tree would be very high
- Average OOB predictions to get $\hat{y}$ then compute metric on predicted $\hat{y}$ vector as usual
- Algorithms for regression and classification later



OOB continued

- OOB error might slightly overestimate test set error. Why?
  - OOB samples are not predicted with all trees in forest whereas test set uses whole forest, which presumably has lower bias/variation [1]
- Some research suggests OOB overestimates error for binary classification https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0201904
- OOB metrics don't affect training, just gives metric
- OOB not to be used with time-sensitive data sets. Why not? Validation set for time-sensitive data can't be split randomly

[1] For n<<p case, see paper https://file.scirp.org/Html/9-1240025_8072.htm

When OOB error is lower than validation

- Maybe the validation set is drawn from a different distribution than the training set or it's a time-sensitive data set
- Or, the model is overfit to the data in the training set, focusing on relationships that are not relevant to the test set
  - E.g., dropping SalesID transaction ID from training set improved our RF model as SalesID never seen in valid set but predictive in training set
  - E.g., if we add (predictive) feature and OOB is still ok, but the validation set is worse, the validation set is not good
- (Sometimes the validation score is a bit better or worse than the OOB score, due to random fluctuations caused by the inherent randomness of RF construction)

Extremely randomized trees (Geurts *et al* 2006)

- The variable/value pair is highly sensitive to the training set, and responsible for much of error rate
- "*The optimal cut-point was shown to depend very strongly on the particular learning sample used…this cut-point variance appeared to be responsible for a significant part of the error rates of tree-based methods.*" https://link.springer.com/article/10.1007/s10994-006-6226-1
- Geurts wondered if more randomness could reduce variance further
- Pick random split value in min(X[:,j]) .. max(X[:,j]), ignoring y!
- Like RF, select $m \leq p$ variables and choose var/value with lowest loss
- Fits using entire X training set, not bootstrap and not subsample (trying to reduce bias)
- Our use of just 11 (not n) X candidate values in the project is similar (an effort to reduce variance and increase speed)

OOB regression scoring

Algorithm: $oob\_score_{regr}(RF,\ X,\ y)$

Let $oob\_counts[i] = 0\ \forall$ records $i = 1..|X|$ (*Num obs. in all leaves reached by X[i]*)

Let $oob\_preds[i] = 0\ \forall$ records $i = 1..|X|$ (*Predictions for X[i] weighted by leaf size*)

foreach $t \in RF$ **do**

$leafsizes = |t.leaf(X[t.oob])|$ (*Num samples in leaf reached by each X*)

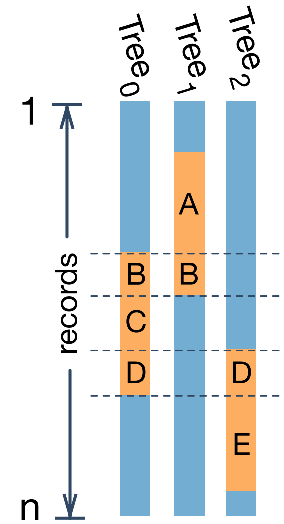
$oob\_preds[t.oob] \mathrel{+}= leafsizes \otimes t.predict(X[t.oob])$

$oob\_counts[t.oob] \mathrel{+}= leafsizes$

end

$$oob\_avg\_preds = \frac{oob\_preds[oob\_counts{>}0]}{oob\_counts[oob\_counts{>}0]}$$

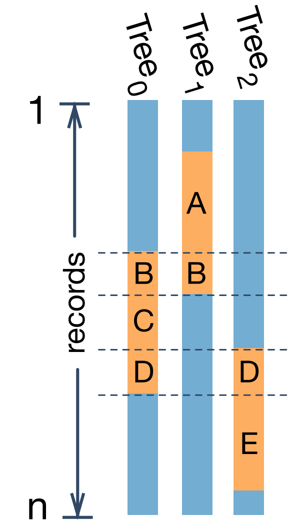
return R^2 *score for* $(y[oob\_counts > 0], oob\_avg\_preds)$



Assumes each tree collects OOB sample indexes during fit()

UNIVERSITY OF SAN FRANCISCO

OOB classification scoring



Algorithm: $oob\_score_{class}(RF,\ X,\ y)$

Let $oob\_counts[i] = 0\ \forall$ records $i = 1..|X|$ (*Num trees w/predictions for X[i]*)

(*Create 2D matrix tracking vote counts per class for each X[i]*):

Let $oob\_preds[i,k] = 0\ \forall$ records $i = 1..|X|, k = 1..|unique(y)|$

foreach $t \in RF$ **do**

  $leafsizes = |t.leaf(X[t.oob])|$ (*Num samples in leaf reached by each OOB X*)

  $tpred = t.predict(X[t.oob])$

  $oob\_preds[t.oob, tpred] \mathrel{+}= leafsizes$ (*count weighted class votes*)

  $oob\_counts[t.oob] \mathrel{+}= 1$ (*track num trees used for each OOB X*)

end

for i *such that* $oob\_counts[i] > 0$ **do**

  $oob\_votes[i] = \arg\max_k oob\_preds[i,k]$

end

return $accuracy\ of\ y[oob\_counts > 0] = oob\_votes$

Assumes each tree collects OOB sample indexes during fit()