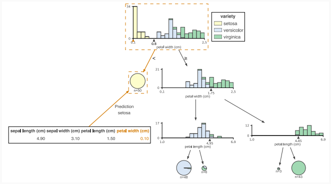
Ensembles of trees: Random Forests and GBMs

Dmitry Adamskiy

February 3, 2020

Outline

Intro: Decision trees



(visualisation by dtreeviz)

- +
  - Very interpretable and 'human-like'
  - Don't need much preprocessing of the data
  - Handle numerical and categorical features well
- -
  - Shallow trees are week predictors
  - Deep trees are prone to overfitting
  - Building optimal trees is hard (thus the greedy approaches)
  - Extrapolating issues (see example).

Bagging: Random Forests

Bagging intuition

- Suppose that you have N predictors that make errors independently

Bagging intuition

- Suppose that you have N predictors that make errors independently :

$$\hat{y}_i^n = y_i + \varepsilon_i^n$$

- Exercise: what would happen if you average their predictions?

Bagging intuition

- Suppose that you have N predictors that make errors independently :

$$\hat{y}_i^n = y_i + \varepsilon_i^n$$

- Exercise: what would happen if you average their predictions?

$$\hat{y}_i = 1/N \sum_{n=1}^{N} \hat{y}_i^n$$

- This leads to the idea of bagging: let's try to train many models and try to decorrelate them.

Random forest

Random Forests (TM) were introduced by Leo Breiman [1]. The two main ideas combined there are:

Random forest

Random Forests (TM) were introduced by Leo Breiman [1]. The two main ideas combined there are:

- Let’s train a number of trees, where each tree is trained on a sample of the same size as the original dataset sampled with replacement.

Random forest

Random Forests (TM) were introduced by Leo Breiman [1]. The two main ideas combined there are:

- Let’s train a number of trees, where each tree is trained on a sample of the same size as the original dataset sampled with replacement.
- At each splitting point let’s restrict a search to a subspace of features.
- Wonderfully parallelisable procedure!

OOB error

Another nice feature of Random Forests is that they provide an estimation of generalisation error for free, without validation set.

- This is called Out-Of-Bag error.
- For each example you compute the prediction using only the trees that were built without this example.

Feature importance

- Random forests are less interpretable than a single decision tree.
- In many applications we still want some interpretability.
- Breiman suggested several techniques to measure feature importance in RF.
- Here is an interesting discussion on why the default one in sklearn is not great. Also, see [3], a recent paper with another tree-ensemble-specific proposal.

Random Forests hyperparameters

- Number of samples you give to each tree (not necessarily sampling with replacement).
- Stopping criteria (like minimum number of samples in the leaf)
- Maximum number of features to sample at each node

How to build a Decision Tree?

- The greedy algorithms typically consider all possible features (from a subspace) and all possible splits.
- Isn't that expensive?
- Here's the derivation for MSE showing how to find a best split in $O(DN \log N)$. Sketch: presort examples by each feature, for each split level n keep $\bar{y}_n = 1/n \sum_{i=1}^{n}(y_i)$ and $s_n = 1/n \sum_{i=1}^{n}(y_i - \bar{y}_n)^2$.
- Update from n to $n + 1$ can be done in constant time.

Proof – 1

- After we make a split, a resulting predictor will make constant predictions $\bar{y}_L$ and $\bar{y}_R$ in the left and right leaves, where, $\bar{y}_L = \frac{1}{N_L}\sum_{i \in L} y_i$ and $\bar{y}_R = \frac{1}{N_R}\sum_{i \in R} y_i$, $N_R + N_L = N$.
- The MSE of a predictor after the split is a weighted average of the MSEs of the left and right children:

$$\text{MSE} = \text{MSE}_L + \text{MSE}_R = \frac{N_L}{N}\sum_{i \in L}(y_i - \bar{y}_L)^2 + \frac{N_R}{N}\sum_{i \in R}(y_i - \bar{y}_R)^2$$

- If we presort the elements based on the given feature and make a split such that the first n elements are in L and the remaining $N - n$ are in R, MSE_L becomes

$$\text{MSE}_L = \frac{N_L}{N} s_n$$

  so if we can update s_n to s_{n+1} in constant time we can compute MSE_L for all the possible splits in one pass over the sorted data points.

Proof – 2

The updates for $\bar{y}_n$ are trivial:

$$\bar{y}_{n+1} = \frac{1}{n+1}(n\bar{y}_n + y_{n+1})$$

Things are tiny bit more involved for s_n:

$$(n+1)s_{n+1} = \sum_{i=1}^{n+1}(y_i - \bar{y}_{n+1})^2 = \sum_{i=1}^{n+1}(y_i - \bar{y}_n + \bar{y}_n - \bar{y}_{n+1})^2$$

$$= ns_n + \cancel{2\sum_{i=1}^{t}(y_i - \bar{y}_n)(\bar{y}_n - \bar{y}_{n+1})} + n(\bar{y}_n - \bar{y}_{n+1})^2 + (y_{n+1} - \bar{y}_{n+1})^2$$

This allows us to update from s_n to s_{n+1} in constant time.

Boosting: GBMs

Gradient Boosting

Gradient Boosting is another type on ensembling…introduced by Friedman [2]

- Instead of building lots of independent predictors, we build an ensemble in stagewise fashion.

Gradient Boosting

Gradient Boosting is another type on ensembling. . . introduced by Friedman [2]

- Instead of building lots of independent predictors, we build an ensemble in stagewise fashion.
- Each new element of the ensemble tried to correct the errors of the previous ones.

Gradient Boosting

Gradient Boosting is another type on ensembling…introduced by Friedman [2]

- Instead of building lots of independent predictors, we build an ensemble in stagewise fashion.
- Each new element of the ensemble tried to correct the errors of the previous ones.
- More than one way of doing this.

Regression example

For regression and MSE the model looks like this. We start with predicting mean of the data $F_0(x) = \bar{y}$ (this is the best constant prediction for MSE!). Then for each iteration.

1. Compute the residuals $r_i^k = y_i - F_k(x_i)$
2. Fit a new model $h_k(x)$ to the residuals.
3. Alter the original model $F_{k+1}(x) = F_k(x) + \alpha h_k(x)$

A few decisions to make:

- How to choose the family for h? How to fit them? Normally (shallow) trees.
- How to choose α? (Line search or fixed hyperparameter). When to stop?

why Gradient?

What is the connection with Gradient Descent?

- Let's have a look at the loss function $L(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
- What is the gradient with respect to the predictions $\hat{y}_i$?

$$\frac{\partial}{\partial \hat{y}_i} L(y, \hat{y}) = -\frac{2}{N}(y_i - \hat{y}_i) = -cr_i$$

- So we are changing our predictions in such a way to move our function towards loss-function minimum, doing gradient descent (in prediction space).

Gradient Descent

We can plug in any differentiable loss function and get pseudo-residuals

$$r^k = \frac{\partial}{\partial \hat{y}} L(y, \hat{y}) \Big|_{\hat{y}=F_{k-1}(X)}$$

Then we can fit a regression tree to these pseudo-residuals using squared loss. Then we can find the best values in the leaves using the original loss.

Implementations

- Scikit-Learn
- XGBoost
- LightGBM
- CatBoost
- ...

References i

L. Breiman.
Random forests.
Mach. Learn., 45(1):5–32, Oct. 2001.

J. H. Friedman.
Greedy function approximation: A gradient boosting machine.
Annals of Statistics, 29:1189–1232, 2000.

S. M. Lundberg, G. G. Erion, and S. Lee.
Consistent individualized feature attribution for tree ensembles.
CoRR, abs/1802.03888, 2018.