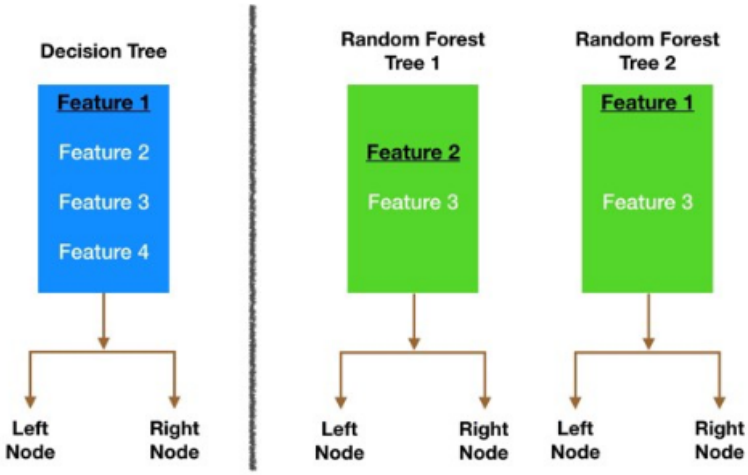
수식 없이 random forest 이해하기

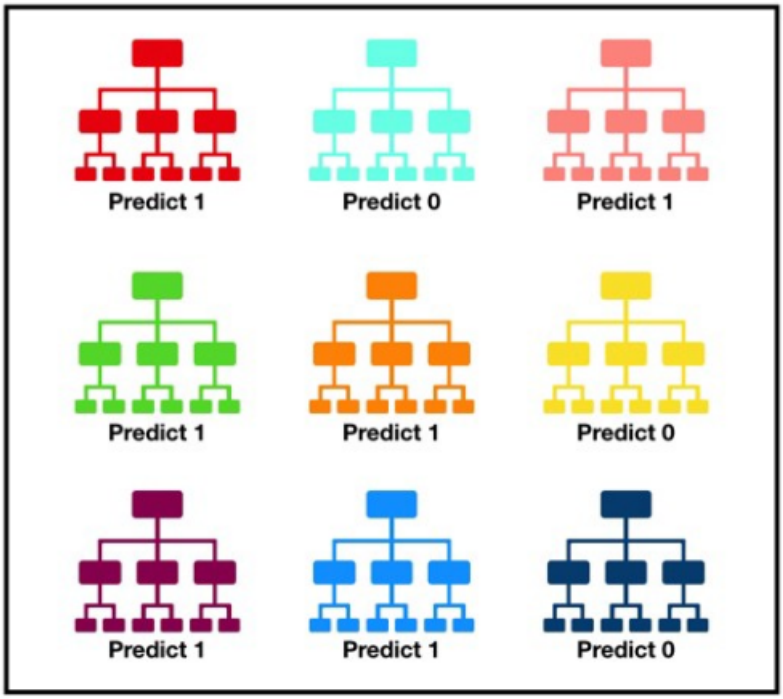
random forest는 decision tree에 기반한 머신러닝 알고리즘이다. CART (Classification and Regression Trees)는 Breiman(1984)이 제안하였으며 분류, 회귀 문제에서 모두 사용되는 모형이다. decision tree는 모형이 굉장히 직관적이고 훈련하기에 쉽다는 장점이 있지만 훈련 데이터에 과적합될 우려가 있다. 마지막 노드에 하나의 데이터가 있을 때까지 tree를 split한다면 물론 훈련 데이터에는 아주 높은 정확도를 보이겠지만 unseen 데이터에 대해서는 성능이 좋지 못하다.
과적합이 발생하는 decision tree의 단점을 보완하기 위해, 많은 연구가 제안되었다. 그 중 대표적인 것이, Efron(1994)이 제안한 bootstrap을 이용하여 여러개의 tree를 키운 후에, 이를 average하는 bagging (Breiman, 1996)하는 방법이다.이 방법은 tree를 키울 때 모든 변수를 사용한다. 즉 각 tree에서 사용하는 변수는 모두 동일하다.
random forest는 여러개의 tree를 키울 때, 각 tree에서 모든 변수를 사용하는 것이 아니라 그 중 일부를 사용한다. 이를 통해서 random forest는 아래와 같이 작동한다.

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

위 표현이 random forest를 한 문장으로 표현한다고 해도 과언이 아니다. 우선 large number of tree라는 것은 많은 수의 tree키운다는 뜻이다. 근데 여기서 ***uncorrelated*** 라는 말에 주목하자. 만약 이 말이 없다면, 모든 변수를 사용하여 각 tree를 키우는 것인데, 그것이 아니라 일부 변수를 사용함으로써 tree가 서로 *uncorrelated* 되는 것이다. 아래 그림을 보자.



왼쪽에는 decision tree, 오른쪽에는 두 개의 tree를 키운 random forest tree가 있다. 만약 데이터가 4개의 feature로 구성된다면, decision tree는 모든 feature를 사용하는 한 개의 tree만 키운다. 훈련 데이터에 대해서 한 개의 tree만 키우므로 테스트 데이터에는 성능이 좋지 않은 것이다. 오른쪽을 보자. 동일한 데이터에 두 개의 tree를 키운 random forest의 경우이다. 각 tree에는 4개의 변수를 모두 사용하는 것이 아니라 그 중 random하게 두 개를 뽑았다. (보통 분류 문제에서는 $\sqrt{p}$개 만큼 뽑고, 회귀 문제에서는 $p/3$개만큼 뽑는다.) 바로 feature를 random하게 뽑음으로써 각 tree가 서로 uncorrelate하게 되는 것이다.
이렇게 각 tree를 키우고, 각 tree마다 결과를 average한다. 분류 문제라면 class label일 것이고 회귀 문제라면 연속형 값이다. average한다는 뜻은, 분류 문제에서 입력 값 x에 대해 각 majority vote를 하여 가장 많은 투표를 받은 class가 배정되고, 회귀 문제에서는 각 tree에서 나온 결과 값의 평균이 입력 값 x의 최종 $\hat{Y}$가 된다. 그림으로 보면 아래와 같다.



알고리즘을 formal하게 적으면 아래와 같다. (ESL에서 참조)

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Out of Bag Error Estimation

bagging에서는 cross-validation 없이 test error를 추정할 수 있는 직관적인 방법이있다. 바로 out of bag samples을 이용하는 것이다.
bagging은 bootstrap과 함께 등장한다. 그러면 n개의 데이터가 있을 때, 데이터를 with replacement로 뽑을 확률은 무엇일까? empirical distribution을 사용한다면, 이는 $1/n$이다. 두 번째, 세 번째 bootstrap을 뽑을 확률도 모두 $1/n$으로 동일한데, 이는 우리가 샘플링을 with replacement로 하기 때문이다. 그럼 반대로, bootstrap sample을 뽑을 때, j번째 데이터가 bootstrap에 있지 않을 확률은 $1 - 1/n$이다. 총 n개의 데이터가 있고 bootstrap도 n개만큼 뽑으므로, n개로 구성된 bootstrap samples이 j번째 데이터를 포함하지 않을 확률은

$$(1-1/n) \times \cdots \times (1-1/n) = (1-1/n)^n = e^{-1} \approx 0.3674 \approx 1/3$$

따라서 평균적으로, 각각의 bagged tree는 전체 데이터의 $2/3$만큼 사용하고 나머지 $1/3$은 사용하지 않는다고 할 수 있다. 바로 이 나머지 $1/3$ 샘플을 *out-of-bag* (OOB) 관측치라고 부른다.
i번째 관측치의 response를 예측하는 상황을 생각해보자. B개의 tree가 있다면, 각 tree마다 bootstrap을 통해 사용한 데이터와 사용하지 않은 데이터 (OOB)가 있을 것이다. 각 i번째 관측치를 포함하지 않은 tree, 즉 i번째 관측치가 OOB에 있는 tree를 골라낸다. 아마도 대략 $B/3$개의 tree가 나올 것이다. $B/3$개의 grown tree를 이용해서, OOB에 있는 i번째 데이터의 response 값을 예측한다. 총 $B/3$개가 나오므로, 회귀 문제라면 평균을 통해, 분류 문제라면 다수결 투표를 통해 최종 1개의 값을 정한다.
이와 같이 random forest에서 OOB 에러를 이용하면, cross-validation 없이도 test error을 추정할 수 있다. 특히, 데이터의 갯수가 매우 많아서 cross-validation의 계산이 클 것으로 예상될 때, OOB 접근법은 매우 유용하다고 할 수 있다.

Variable Importance Measures

random forest을 돌리고 나면, 각 변수별로 '중요도'를 뽑아볼 수가 있다. 이 중요도는 어떻게 계산이 되는 것일까? 변수 중요도는 tree를 split할 때 사용하는 기준과 관련이 있다. 분류 tree에서 일반적으로 사용되는 Gini index를 예로 들어보자. 분류 tree에서는 split할 때 각 변수별로 줄어드는 Gini index을 추적해서, 가장 많은 Gini index을 감소시키는 변수를 기준으로 tree를 split한다. 이 감소하는 정도를 모든 tree에서 기록한다. 만약 중요한 변수라면, 이 변수가 많이 선택되고 그만큼 감소되는 Gini index도 클 것이다. 이를 모든 변수에 대해 기록하여 가장 많은 Gini index를 감소시킨 변수부터 나열할 수 있다. 바로 이 순서를 변수 중요도를 나타낸다고 보는 것이다.

Number of Features to Choose

random forest에서 각 tree를 키울 때, 모든 변수를 사용하는 것이 아니라 일부 변수만 사용함을 살펴보았고 회귀 문제에서는 $p/3$개, 분류 문제에서는 $\sqrt{p}$개의 변수를 랜덤하게 선택한다. 이는 Breiman이 제시한 갯수로, 이에 대한 일반적인 증명은 존재하지 않는다. 그러나 Hastie (ESL)에 따르면 변수의 수는 일종의 tuning parameter이므로, 모델링 이전에 tune 되어야 한다고 말한다.
또 한 가지 기억해야할 점은, Breiman에 따르면 범주형 변수가 많다면 선택하는 변수의 갯수를 늘리는 것도 전략이라고 한다. 이때는 $int(log_2 M + 1)$로 선택하는 것이 좋은 성능을 보장한다고 한다.

Comparison with Boosting

tree을 aggregate하는 또 다른 머신러닝 기법으로는 boosting이 있다. boosting과 random forest 모두 여러 개의 tree를 사용함으로써 한 개의 tree를 사용하는 것보다 더 나은 결과를 기대한다. 둘 사이의 가장 큰 차이점은 다음과 같다. random forest에서 각 tree는 서로 독립이라고 볼 수 있다. 즉, 각 tree를 형성할 때 bootstrap을 통해서 bootstrap samples을 구성하고, 각 tree를 독립적으로 키운다. 하나의 tree가 다른 tree의 성장에 영향을 미치지 않는다. 각 tree는 *uncorrelate*되어 있는 것이다.
하지만 boosting에서는 각 tree가 *correlate*되어 있다. boosting은 이전 tree가 잘 학습하지 못한 부분을 다음 tree가 보완하려고 한다. 즉, tree를 키울 때, 다른 tree에 영향을 받는다는 것이다.