



Ensemble Models Demystified

@KevinLemagnen



Ensembles: Why do we care?

- Good performance
- General purpose
- Usually easier to train than other fancy techniques
- Really popular in industry and ML competitions

Theory



CAMBRIDGE SPARK

1

[illegible]



Agenda

1. Intuition
2. Weak learner (Decision Tree)
3. Bagging (Random Forest)
4. Boosting (Gradient Boosting)
5. Other boosting libraries

1. Intuition

What are ensemble models?

- Combining multiple simple models (weak learners) into a larger one (ensemble)
- Two popular techniques:
 - Bagging
 - Boosting
 - Usually with decision trees as weak learner

Intuition



Accuracy = 60%



Accuracy = 75%

Both say you have **X**... what's the likelihood that you really have **X**?

Two big assumptions

- **Weak Learner:** “Experts” need to be more right than wrong on average
 - > 50% accuracy
- **Diversity:** “Experts” need to make different errors

2. Weak Learners

Decision Trees

Why are they good?

- Can capture complex relationships in the data
 - We'll often be able to get our $> 50\%$ accuracy!
- Overfits easily
 - We can use that to create diverse models!

How do we control them?

No constraints = one leaf per sample = **massive overfitting**

Some good constraints:

- Pick a **maximum depth**
- Pick a **minimum number of samples** needed in a new node/leaf

3. Bagging

Random Forest

How do we build **diverse** trees?

- Each one is trained on a subsample of **observations** [Bootstrapping]

How do we build **diverse** trees?

- Each one is trained on a subsample of **observations** [Bootstrapping]
- Each one is trained on a subsample of **features**

How do we build **diverse** trees?

- Each one is trained on a subsample of **observations** [Bootstrapping]
- Each one is trained on a subsample of **features**
- Loosen your constraints to let your trees overfit

How do we build **diverse** trees?

- Each one is trained on a subsample of **observations** [Bootstrapping]
- Each one is trained on a subsample of **features**
- Loosen your constraints to let your trees overfit

Don't overdo it... We still need:

- Good performance per tree (no underfitting)

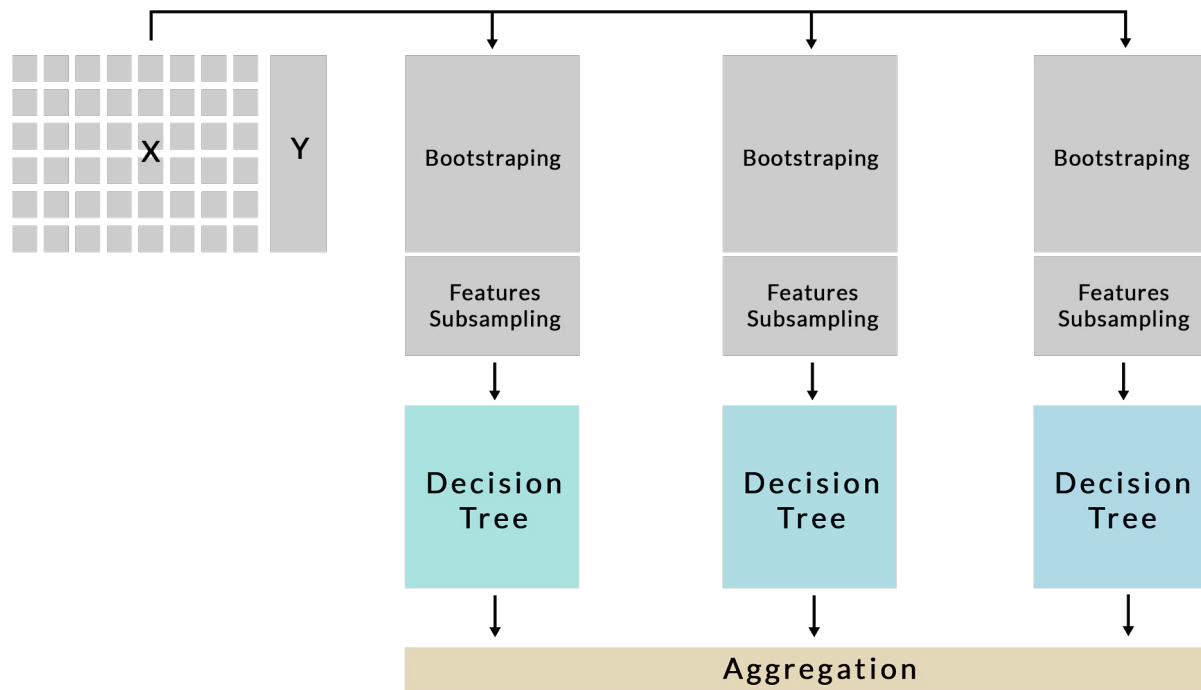
How do we build **diverse** trees?

- Each one is trained on a subsample of **observations** [Bootstrapping]
- Each one is trained on a subsample of **features**
- Loosen your constraints to let your trees overfit

Don't overdo it... We still need:

- Good performance per tree (no underfitting)
- Able to generalise (no overfitting)

Bagging - Random Forest



Some pros and cons

- + Easy to run in **parallel**

Some pros and cons

- + Easy to run in **parallel**
- + Decision Trees = we can get **feature importance**

Some pros and cons

- + Easy to run in **parallel**
- + Decision Trees = we can get **feature importance**
- Models remain **correlated** (similar data)

Some pros and cons

- + Easy to run in **parallel**
- + Decision Trees = we can get **feature importance**
- Models remain **correlated** (similar data)
- Hard to **interpret**

Some pros and cons

- + Easy to run in **parallel**
- + Decision Trees = we can get **feature importance**
- Models remain **correlated** (similar data)
- Hard to **interpret**
- ? **Outliers** likely to be ignored by most weak learners

4. Boosting

Gradient Boosting

Boosting - Intuition

We want to build weak learners that **actively compensate** each others' errors.

Boosting - Intuition

We want to build weak learners that **actively compensate** each others' errors.

Let's focus on one sample: (\mathbf{x}, y) with $y = 100$

Boosting - Intuition

We want to build weak learners that **actively compensate** each others' errors.

Let's focus on one sample: (\mathbf{X}, y) with $y = 100$

1. Train DT1 on (\mathbf{X}, y)

$$DT1(X) = 95$$

Boosting - Intuition

We want to build weak learners that **actively compensate** each others' errors.

Let's focus on one sample: (\mathbf{X}, y) with $y = 100$

1. Train DT1 on (\mathbf{X}, y)
2. Compute residuals

$$DT1(X) = 95$$

$$r = 100 - 95 = 5$$

Boosting - Intuition

We want to build weak learners that **actively compensate** each others' errors.

Let's focus on one sample: (\mathbf{X}, y) with $y = 100$

1. Train DT1 on (\mathbf{X}, y)
2. Compute residuals
3. Train DT2 on $(\mathbf{X}, 5)$

$$DT1(X) = 95$$

$$r = 100 - 95 = 5$$

$$DT2(X) = 4$$

Boosting - Intuition

We want to build weak learners that **actively compensate** each others' errors.

Let's focus on one sample: (\mathbf{X}, y) with $y = 100$

1. Train DT1 on (\mathbf{X}, y)
2. Compute residuals
3. Train DT2 on $(\mathbf{X}, 5)$
4. Aggregate DT1 and DT2

$$DT1(X) = 95$$

$$r = 100 - 95 = 5$$

$$DT2(X) = 4$$

$$DT1(X) + DT2(X) = 95 + 4 = 99$$

Boosting - Intuition

We want to build weak learners that **actively compensate** each others' errors.

Let's focus on one sample: (\mathbf{X}, y) with $y = 100$

1. Train DT1 on (\mathbf{X}, y)
2. Compute residuals
3. Train DT2 on $(\mathbf{X}, 5)$
4. Aggregate DT1 and DT2
5. Repeat

$$DT1(X) = 95$$

$$r = 100 - 95 = 5$$

$$DT2(X) = 4$$

$$DT1(X) + DT2(X) = 95 + 4 = 99$$

Boosting - Intuition

We want to build weak learners that **actively compensate** each others' errors.

Let's focus on one sample: (\mathbf{X}, y) with $y = 100$

1. Train DT1 on (\mathbf{X}, y)

$$DT1(X) = 95$$

2. Compute residuals

$$r = 100 - 95 = 5$$

3. Train DT2 on $(\mathbf{X}, 5)$

$$DT2(X) = 4$$

4. Aggregate DT1 and DT2

$$DT1(X) + DT2(X) = 95 + 4 = 99$$

5. Repeat

Weak learners increasingly focus on "hard points"

Boosting - Gradient Boosting

too many stages **OR** too complex trees = overfit to noise

Boosting - Gradient Boosting

too many stages **OR** too complex trees = overfit to noise

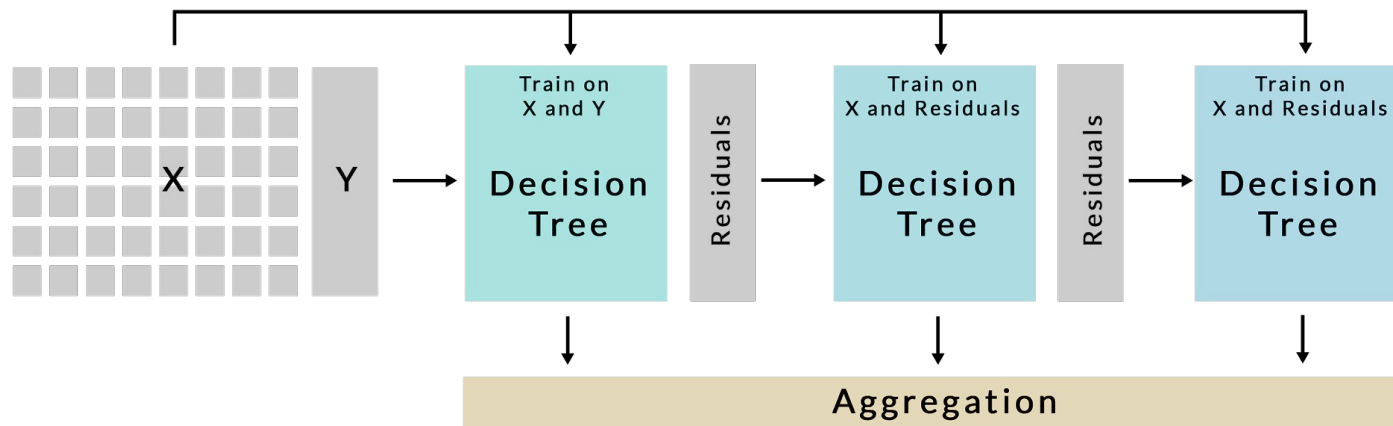
- Getting the number of stages right is **extremely** important

Boosting - Gradient Boosting

too many stages **OR** too complex trees = overfit to noise

- Getting the number of stages right is **extremely** important
- We need to build **small, constrained** trees

Boosting - Gradient Boosting



Some pros and cons

- + Great **performance** (usually)

Some pros and cons

- + Great **performance** (usually)
- + Decision Trees = we can get **feature importance**

Some pros and cons

- + Great **performance** (usually)
- + Decision Trees = we can get **feature importance**
- Hard to run in **parallel**

Some pros and cons

- + Great **performance** (usually)
- + Decision Trees = we can get **feature importance**
- Hard to run in **parallel**
- Hard to **interpret**

Some pros and cons

- + Great **performance** (usually)
- + Decision Trees = we can get **feature importance**
- Hard to run in **parallel**
- Hard to **interpret**
- Can easily **overfit**

Any questions?