Lecture 5: Decision Trees, Random Forests and the Bias-Variance Tradeoff

Machine Learning Decal

Hosted by Machine Learning at Berkeley

Agenda

Background

Decision Tree Setup

Decision Tree Algorithm and Justification

Random Forests

Boosting

The Bias-Variance Tradeoff

Questions

Background

Let's Play A Game

The game is 20 Questions. Let's review the objective and rules:

Let's Play A Game

The game is 20 Questions. Let's review the objective and rules:

- Player 1 thinks of a particular thing in a broad category e.g. People, Animals, Vegetables, etc.

Let's Play A Game

The game is 20 Questions. Let's review the objective and rules:

- Player 1 thinks of a particular thing in a broad category e.g. People, Animals, Vegetables, etc.
- Player 2 must determine what Player 1 is thinking of.

Let's Play A Game

The game is 20 Questions. Let's review the objective and rules:

- Player 1 thinks of a particular thing in a broad category e.g. People, Animals, Vegetables, etc.
- Player 2 must determine what Player 1 is thinking of.
- Player 2 can ask up to 20 **Yes or No** questions of Player 1, and he is able to correctly guess the thing, then Player 2 wins.

Let's Play The Game

Alright. I'm Player 1, and you are going to be Player 2. I'm thinking of a Person. Which of the following questions are you most likely to **ask me first?** Why?

Let's Play The Game

Alright. I'm Player 1, and you are going to be Player 2. I'm thinking of a Person. Which of the following questions are you most likely to **ask me first?** Why?

- "Is the person from Hawaii?"

Let's Play The Game

Alright. I'm Player 1, and you are going to be Player 2. I'm thinking of a Person. Which of the following questions are you most likely to **ask me first?** Why?

- "Is the person from Hawaii?"
- "Is your person male?"

Dissection

You probably chose **"Is your person male?"** Why is this obviously the best choice?

Dissection

You probably chose **"Is your person male?"** Why is this obviously the best choice?

- You can automatically **eliminate half** of the candidates for Player 1's choice.

Dissection

You probably chose **"Is your person male?"** Why is this obviously the best choice?

- You can automatically **eliminate half** of the candidates for Player 1's choice.
- Compared to asking the Hawaii question:

Dissection

You probably chose **"Is your person male?"** Why is this obviously the best choice?

- You can automatically **eliminate half** of the candidates for Player 1's choice.
- Compared to asking the Hawaii question:
  - If from Hawaii: You have awesomely good luck. You'll likely win soon.

Dissection

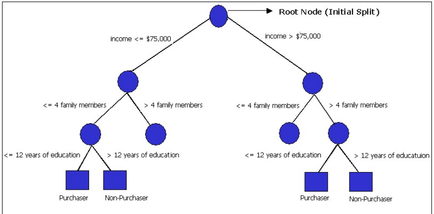
You probably chose **"Is your person male?"** Why is this obviously the best choice?

- You can automatically **eliminate half** of the candidates for Player 1's choice.
- Compared to asking the Hawaii question:
  - If from Hawaii: You have awesomely good luck. You'll likely win soon.
  - If not from Hawaii: You basically **wasted a question.**

Decision Tree Setup

Decision Tree Structure

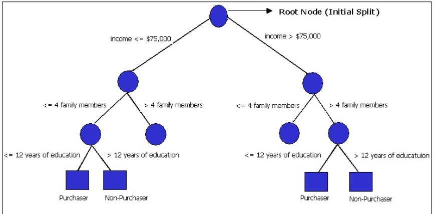
- The **Decision Tree** is a model with a tree structure that encodes a collection of **if, then** statements in the **internal nodes**.



Hypothetical Classification Tree

Decision Tree Structure

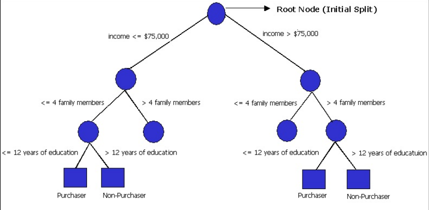
- The **Decision Tree** is a model with a tree structure that encodes a collection of **if, then** statements in the **internal nodes**.
- A query data point's features are used to channel the point to a **leaf node**, where it is assigned a label (classification) or value (regression).



Hypothetical Classification Tree

Decision Tree Structure

- The **Decision Tree** is a model with a tree structure that encodes a collection of **if, then** statements in the **internal nodes**.
- A query data point's features are used to channel the point to a **leaf node**, where it is assigned a label (classification) or value (regression).
- Works naturally for continuous and categorical features!



Hypothetical Classification Tree

It's like playing 20 questions with a data point $x^{(i)}$ like so:

- We wish to "guess" $y^{(i)} \in \{0, 1\}$ for classification and $y^{(i)} \in \mathbb{R}$ for regression.

Making a Game out of Data Science

It's like playing 20 questions with a data point $x^{(i)}$ like so:

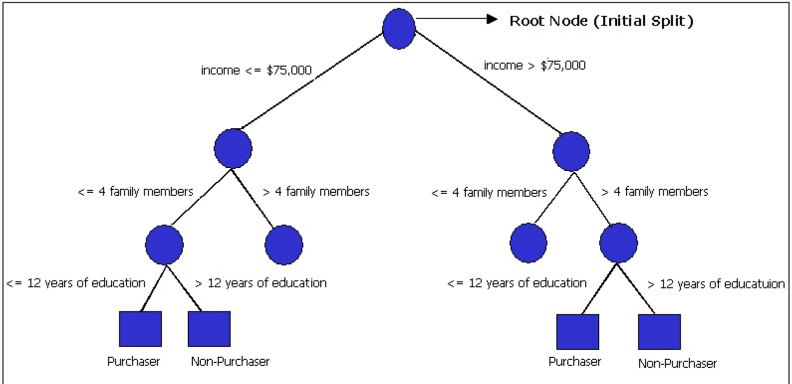
- We wish to "guess" $y^{(i)} \in \{0, 1\}$ for classification and $y^{(i)} \in \mathbb{R}$ for regression.
- We "ask" only binary "questions" of $x^{(i)}$, e.g. $x_3^{(i)} \geqslant 5$?.

ML@B

It's like playing 20 questions with a data point $x^{(i)}$ like so:

- We wish to "guess" $y^{(i)} \in \{0, 1\}$ for classification and $y^{(i)} \in \mathbb{R}$ for regression.
- We "ask" only binary "questions" of $x^{(i)}$, e.g. $x_3^{(i)} \geqslant 5$?.
- These questions, or more formally **data splits** are what parameterize our model.

Decision Tree Example



Hypothetical Classification Tree

Somehow, we must use our training dataset to learn the **features** and **values** to choose all of the splits shown.

Decision Tree Algorithm and Justification

Decision Tree Training Algorithm

We build the decision tree with the following **recursive** algorithm.

β is the value of the feature α on which the point will be split.

GrowTree(Set $S \subseteq \{1 \dots n\}$):

- if StopCriteria(S) is True:

Decision Tree Training Algorithm

We build the decision tree with the following **recursive** algorithm.

β is the value of the feature α on which the point will be split.

GrowTree(Set $S \subseteq \{1 \dots n\}$):

- if StopCriteria(S) is True:
  - return new *Leaf*(Value(S))

Decision Tree Training Algorithm

We build the decision tree with the following **recursive** algorithm.

β is the value of the feature α on which the point will be split.

GrowTree(Set $S \subseteq \{1 \dots n\}$):

- if StopCriteria(S) is True:
    - return new *Leaf*(Value(S))
- $\alpha, \beta \leftarrow$ findSplit(S)

Decision Tree Training Algorithm

We build the decision tree with the following **recursive** algorithm.

β is the value of the feature α on which the point will be split.

GrowTree(Set $S \subseteq \{1 \ldots n\}$):

- if StopCriteria(S) is True:
  - return new *Leaf*(Value(S))
- $\alpha, \beta \leftarrow$ findSplit(S)
- $S_l \leftarrow \{i : x_\alpha^{(i)} \leqslant \beta\}$ // left split

Decision Tree Training Algorithm

We build the decision tree with the following **recursive** algorithm.

β is the value of the feature α on which the point will be split.

GrowTree(Set $S \subseteq \{1 \ldots n\}$):

- if StopCriteria(S) is True:
  - return new *Leaf*(Value(S))
- $\alpha, \beta \leftarrow$ findSplit(S)
- $S_l \leftarrow \{i : x_\alpha^{(i)} \leqslant \beta\}$ // left split
- $S_r \leftarrow \{i : x_\alpha^{(i)} > \beta\}$ // right split

Decision Tree Training Algorithm

We build the decision tree with the following **recursive** algorithm.

β is the value of the feature α on which the point will be split.

GrowTree(Set $S \subseteq \{1 \ldots n\}$):

- if StopCriteria(S) is True:
  - return new *Leaf*(Value(S))
- $\alpha, \beta \leftarrow$ findSplit(S)
- $S_l \leftarrow \{i : x_\alpha^{(i)} \leqslant \beta\}$ // left split
- $S_r \leftarrow \{i : x_\alpha^{(i)} > \beta\}$ // right split
- return new *InternalNode*(α, β, GrowTree(S_l), GrowTree(S_r))

Picking Good Parameters in findSplit(S)

We have shown how to find ideal parameters θ using **gradient descent**. This works because we are **optimizing a loss function**, $J(\theta)$.

Ideal split criteria in decision trees are chosen through minimizing the amount of **entropy** in each split group.

What is Entropy?

- In Data Science, entropy $\rightarrow$ uncertainty $\rightarrow$ surprise in knowing label

What is Entropy?

- In Data Science, entropy $\rightarrow$ uncertainty $\rightarrow$ surprise in knowing label
- $p_c = P(y_i = c) \rightarrow H(p_c) = -log_2 p_c$

What is Entropy?

- In Data Science, entropy $\rightarrow$ uncertainty $\rightarrow$ surprise in knowing label
- $p_c = P(y_i = c) \rightarrow H(p_c) = -log_2 p_c$
- What is H(0)? H(1)? H($\frac{1}{2}$)?

What is Entropy?

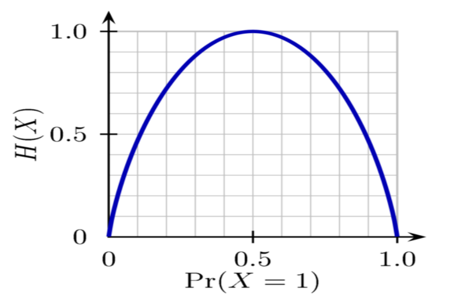
Data points belong to total of c classes, and fraction of data points in each is $[p_0...p_{c-1}] : \sum_i p_i = 1$, then **entropy (H) of the set** is:

$$H = -\sum_{i=0}^{c-1} p_i \log p_i$$

(Using log base 2) If we have two classes, what values of p_0, p_1 will maximize this function? Do you see the connection to high entropy?

Visualizing the Entropy Function

Here is a graph depicting $H(p_0)$ with $p_1 = 1 - p_0$:



To summarize: A Set S will have low entropy when $|p_0 - p_1|$ is large.

Choosing Optimal Splits

Think back to 20 Questions.

Choosing Optimal Splits

Think back to 20 Questions.

- The questions that give you the **most information gain** are the ones that **minimize entropy** in the two branches - S_l, S_r

Choosing Optimal Splits

Think back to 20 Questions.

- The questions that give you the **most information gain** are the ones that **minimize entropy** in the two branches - S_l, S_r
- We weight the size of the sets: $|S_l|, |S_r|$ to **discourage lopsided splits** unless necessary.

Choosing Optimal Splits

Think back to 20 Questions.

- The questions that give you the **most information gain** are the ones that **minimize entropy** in the two branches - S_l, S_r
- We weight the size of the sets: $|S_l|, |S_r|$ to **discourage lopsided splits** unless necessary.

Choosing Optimal Splits

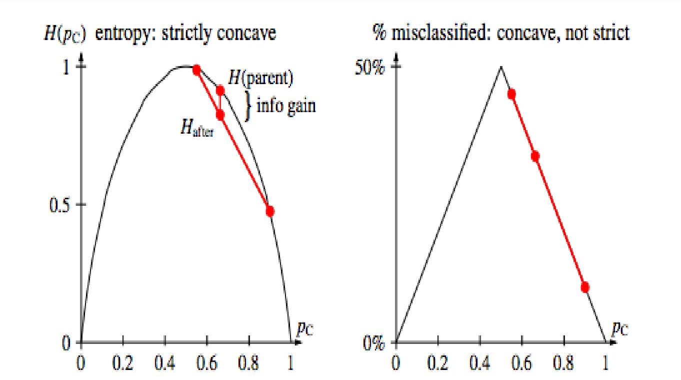
Think back to 20 Questions.

- The questions that give you the **most information gain** are the ones that **minimize entropy** in the two branches - S_l, S_r
- We weight the size of the sets: $|S_l|, |S_r|$ to **discourage lopsided splits** unless necessary.

The **entropy of a split** is given as

$$H_{after} = \frac{|S_l| H_{S_l} + |S_r| H_{S_r}}{|S_l| + |S_r|}$$

Visualizing Optimal Split



Choosing Optimal Splits Contd.

Hence, findSplit(S) gives split from:

$$arg \min H_{after}(S_l, S_r)$$

Choosing Optimal Splits Contd.

Hence, findSplit(S) gives split from:

$$arg \min H_{after}(S_l, S_r)$$

There is no analytical solution for the problem above. Best option is checking each split (α, β) manually.

Choosing Optimal Splits Contd.

Hence, findSplit(S) gives split from:

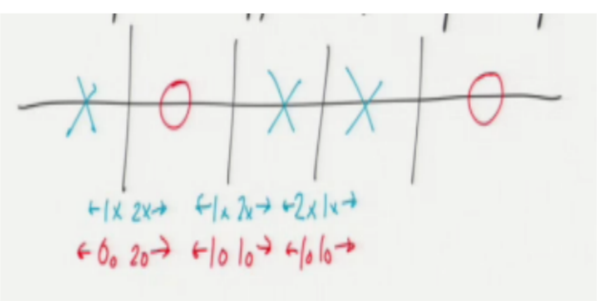
$$arg\min H_{after}(S_l, S_r)$$

There is no analytical solution for the problem above. Best option is checking each split (α, β) manually.

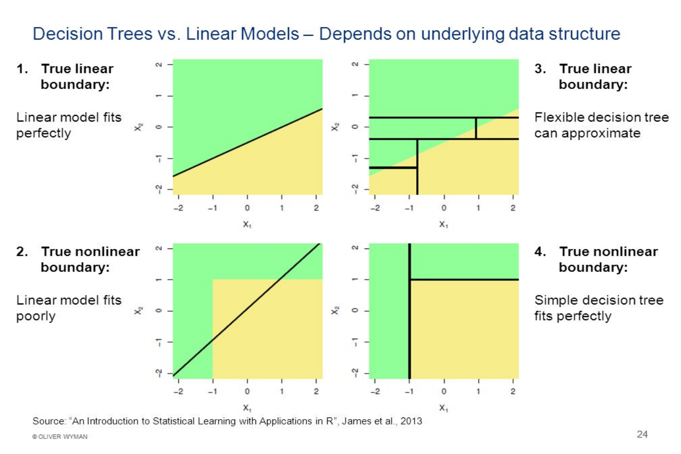
- With n data points, d features, and up to k splits on a feature: runtime is $\mathcal{O}(ndk)$

Optimized Split Selection

Sorting the points once per feature, we can evaluate entropy for that feature on each split in $\mathcal{O}(1)$ time. This gives us an overall runtime of $\mathcal{O}(nd \log n)$

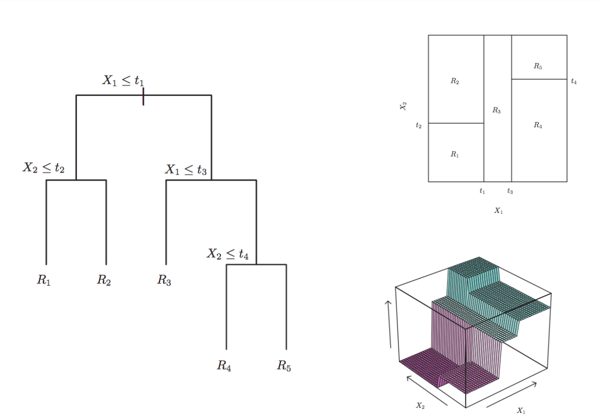


Finished tree can divide feature space into arbitrary number of *axis-aligned* regions.

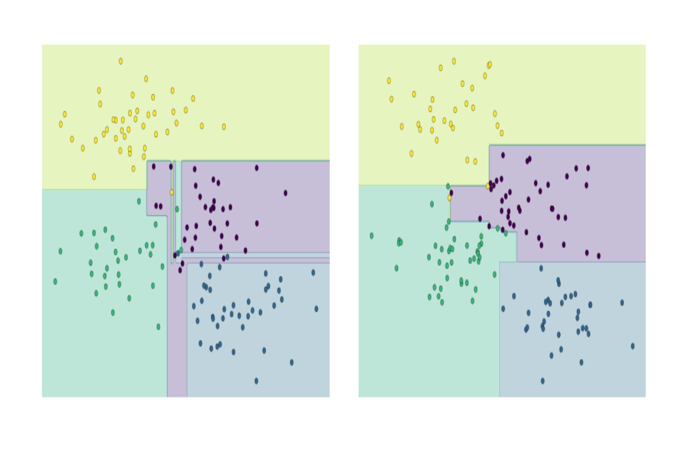


Decision Tree Decision Boundary: Regression

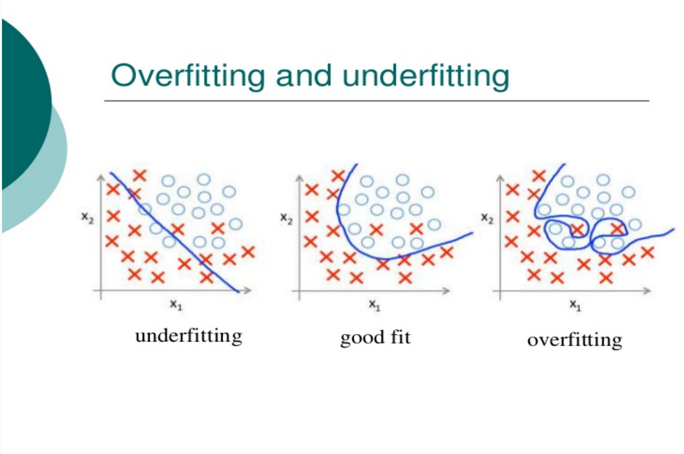
Image Credits to Professor Shewchuk



Which one would you rather have? Points are **training data.**



Overfitting vs Underfitting



Improving Generalization

Two common strategies:

Improving Generalization

Two common strategies:

- **Early Stopping:** Limit depth, threshold purity of leaves (% same class or set range for regression), few data points, tiny volume

Improving Generalization

Two common strategies:

- **Early Stopping:** Limit depth, threshold purity of leaves (% same class or set range for regression), few data points, tiny volume
- **Pruning:** Create whole tree. Greedily remove leaves, test for improved validation accuracy.

Improving Generalization

Two common strategies:

- **Early Stopping:** Limit depth, threshold purity of leaves (% same class or set range for regression), few data points, tiny volume
- **Pruning:** Create whole tree. Greedily remove leaves, test for improved validation accuracy.

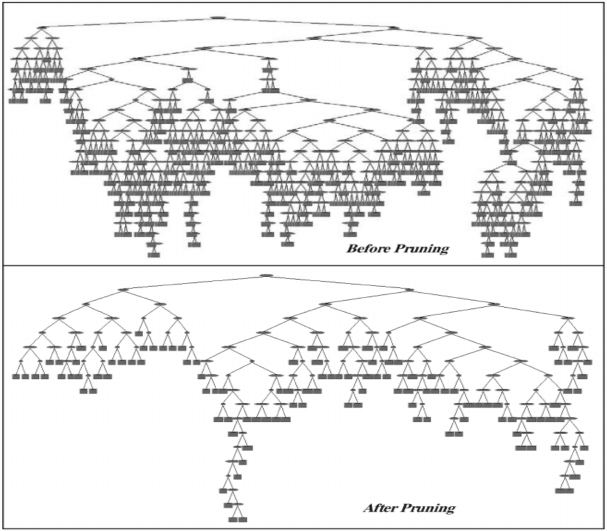
Improving Generalization

Two common strategies:

- **Early Stopping:** Limit depth, threshold purity of leaves (% same class or set range for regression), few data points, tiny volume
- **Pruning:** Create whole tree. Greedily remove leaves, test for improved validation accuracy.

Impure leaves return majority vote (classification), average value (regression).

Neater Tree



Break: Cool Website

Let's see a cool application of Decision Trees to a somewhat-practical application: Click Me!

Random Forests

Making Decisions Trees More Powerful

Back to 20 questions.

Back to 20 questions.

- If many people play against you, it is more likely that **at least one of them** figures it out than the individual chance someone figures it out.

Back to 20 questions.

- If many people play against you, it is more likely that **at least one of them** figures it out than the individual chance someone figures it out.
- Some people are better at asking questions than others.

Making Decisions Trees More Powerful

Back to 20 questions.

- If many people play against you, it is more likely that **at least one of them** figures it out than the individual chance someone figures it out.
- Some people are better at asking questions than others.

Making Decisions Trees More Powerful

Back to 20 questions.

- If many people play against you, it is more likely that **at least one of them** figures it out than the individual chance someone figures it out.
- Some people are better at asking questions than others.

What if we made one model by **combining many decision trees?** Combine their judgments into something nuanced.

Simulating "Different People"

Original decision tree algorithm → **every tree would be identical**. Encourage diversity in finding splits:

Simulating "Different People"

Original decision tree algorithm → **every tree would be identical**. Encourage diversity in finding splits:

- Each tree gets **different data**. Sample points $x^{(i)}$ **with replacement** to form different starting sets S.

Simulating "Different People"

Original decision tree algorithm → **every tree would be identical**. Encourage diversity in finding splits:

- Each tree gets **different data**. Sample points $x^{(i)}$ **with replacement** to form different starting sets S.
- At each call of GrowTree(), constrict the FindSplit() method to examine only a **subset of features** to split on.

Simulating "Different People"

Original decision tree algorithm → **every tree would be identical**. Encourage diversity in finding splits:

- Each tree gets **different data**. Sample points $x^{(i)}$ **with replacement** to form different starting sets S.
- At each call of GrowTree(), constrict the FindSplit() method to examine only a **subset of features** to split on.

Simulating "Different People"

Original decision tree algorithm → **every tree would be identical**. Encourage diversity in finding splits:

- Each tree gets **different data**. Sample points $x^{(i)}$ **with replacement** to form different starting sets S.
- At each call of GrowTree(), constrict the FindSplit() method to examine only a **subset of features** to split on.

These techniques are called **bootstrap aggregating (bagging)** and **random feature selection** respectively.

Diverse Tree Benefits

Why do the aforementioned techniques work?

Diverse Tree Benefits

Why do the aforementioned techniques work?

- **Bagging:** Points (almost surely) repeated in data $\rightarrow$ weighted more $\rightarrow$ minimizes the effect of outliers.

Diverse Tree Benefits

Why do the aforementioned techniques work?

- **Bagging:** Points (almost surely) repeated in data → weighted more → minimizes the effect of outliers.
- **Random feature selection:** Despite bagging, trees may still split on same first feature. This forces different "perspectives" on split paths.

Diverse Tree Benefits

Why do the aforementioned techniques work?

- **Bagging:** Points (almost surely) repeated in data → weighted more → minimizes the effect of outliers.
- **Random feature selection:** Despite bagging, trees may still split on same first feature. This forces different "perspectives" on split paths.
    - *Example:* Asking "Is the person still alive?" might be better than asking if the person is male, *depending on the situation.*

Widely Applicable Strategies

The aforementioned techniques also work for other models.

Widely Applicable Strategies

The aforementioned techniques also work for other models.

- **Bagging:** for SVM's, logistic regression, $\#$ of times point is picked $\rightarrow$ point's weight in loss function. It results in $\approx 62.3\%$ of data represented.

Widely Applicable Strategies

The aforementioned techniques also work for other models.

- **Bagging:** for SVM's, logistic regression, # of times point is picked $\rightarrow$ point's weight in loss function. It results in $\approx 62.3\%$ of data represented.
- **Random Feature Selection** $\rightarrow$ A way to address high dimensionality on optimization based strategies.

Widely Applicable Strategies

The aforementioned techniques also work for other models.

- **Bagging:** for SVM's, logistic regression, # of times point is picked $\rightarrow$ point's weight in loss function. It results in $\approx 62.3\%$ of data represented.
- **Random Feature Selection** $\rightarrow$ A way to address high dimensionality on optimization based strategies.
    - For random forests, $\sqrt{d}$, $\frac{d}{3}$ features are recommended for splits on classification and regression respectively.

Random Forests

A **random forest** is defined to be:

A **random forest** is defined to be:

- One model

A **random forest** is defined to be:

- One model
- Composed of multiple decision trees

Random Forests

A **random forest** is defined to be:

- One model
- Composed of multiple decision trees
- Which all use both, bagging and random feature selection

Random Forests

A **random forest** is defined to be:

- One model
- Composed of multiple decision trees
- Which all use both, bagging and random feature selection
- And work by casting majority votes (classification) or averages (regression), with values made from all of the constituent decision trees.

More About Random Forests

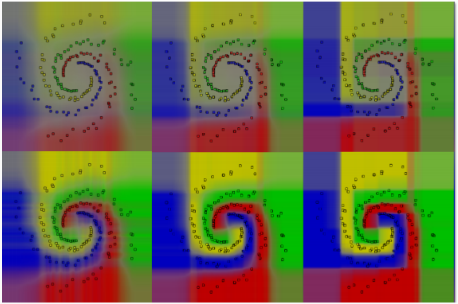
Greatest advantage of random forests is their ability to make **arbitrarily complex decision boundaries** given a large enough number of trees.

More About Random Forests

Greatest advantage of random forests is their ability to make **arbitrarily complex decision boundaries** given a large enough number of trees.

Despite this, random forests are less likely to overfit.

Random Forest Decision Boundaries



Top row: depth = 4, Bottom row: depth = 12.

Left: 1 random feature per split, Middle: 5 features, Right: 50 features

Boosting

Overview of Boosting

Instead of taking the majority vote of **many strong decision trees**, **sequentially generate weak decision trees** where each tree focuses accurately classifying the data points the last tree got wrong.

AdaBoost Algorithm

- An overall predictor takes the majority or average of multiple decision trees

AdaBoost Algorithm

- An overall predictor takes the majority or average of multiple decision trees
- Initially train a weak decision tree on all of the training data

AdaBoost Algorithm

- An overall predictor takes the majority or average of multiple decision trees
- Initially train a weak decision tree on all of the training data
- Re-weight the data points so points that were previously misclassified get a higher weight

AdaBoost Algorithm

- An overall predictor takes the majority or average of multiple decision trees
- Initially train a weak decision tree on all of the training data
- Re-weight the data points so points that were previously misclassified get a higher weight
- Train a new weak decision tree on the re-weighted datasets and repeat

Bagging

- Base model is a low bias (high variance) decision tree

Boosting

Boosting vs Bagging

Bagging

- Base model is a low bias (high variance) decision tree
- Increases validation accuracy

Boosting

Boosting vs Bagging

Bagging

- Base model is a low bias (high variance) decision tree
- Increases validation accuracy
- Can balance out unstable learners

Boosting

Bagging

- Base model is a low bias (high variance) decision tree
- Increases validation accuracy
- Can balance out unstable learners

Boosting

- Base model is a low variance (high bias) decision tree

Boosting vs Bagging

Bagging

- Base model is a low bias (high variance) decision tree
- Increases validation accuracy
- Can balance out unstable learners

Boosting

- Base model is a low variance (high bias) decision tree
- Increases training accuracy

Boosting vs Bagging

Bagging

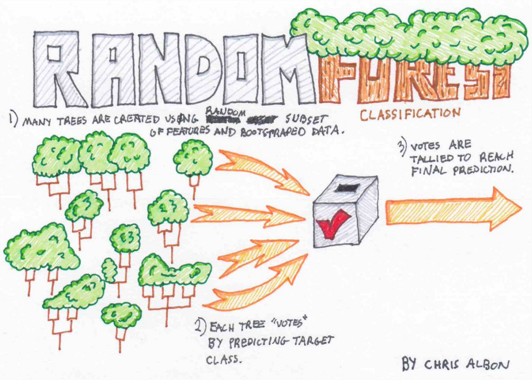
- Base model is a low bias (high variance) decision tree
- Increases validation accuracy
- Can balance out unstable learners

Boosting

- Base model is a low variance (high bias) decision tree
- Increases training accuracy
- Can strengthen weak learners

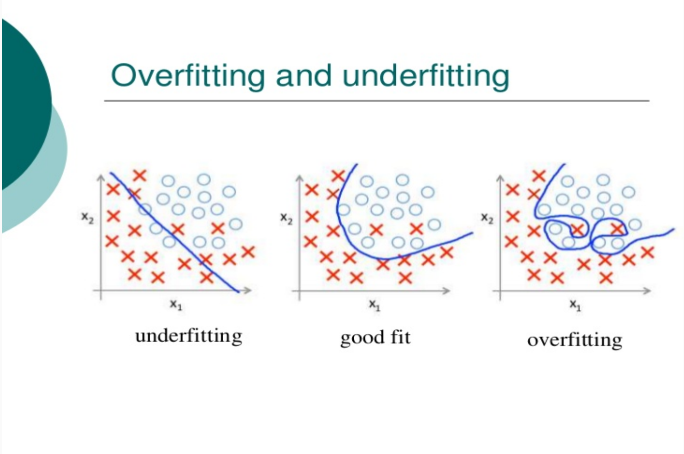
Break: Sklearn's Random Forest Class

Let's check out how our favorite data science model library presents Random Forests: Click Me!



The Bias-Variance Tradeoff

A Common Theme



Setup

We start by assuming our data comes from a true function f but also has random noise ϵ added afterward. We have a hypothesis h which is always imperfect for 2 reasons:

- **Bias:** h is unable to fit f perfectly because it lacks capacity or complexity. (Fitting a line to a parabola).

Setup

We start by assuming our data comes from a true function f but also has random noise ϵ added afterward. We have a hypothesis h which is always imperfect for 2 reasons:

- **Bias:** h is unable to fit f perfectly because it lacks capacity or complexity. (Fitting a line to a parabola).
- **Variance:** h is fitting to the noise in the data, thus missing the true function f.

Setup

We start by assuming our data comes from a true function f but also has random noise ϵ added afterward. We have a hypothesis h which is always imperfect for 2 reasons:

- **Bias:** h is unable to fit f perfectly because it lacks capacity or complexity. (Fitting a line to a parabola).
- **Variance:** h is fitting to the noise in the data, thus missing the true function f.

Setup

We start by assuming our data comes from a true function f but also has random noise ϵ added afterward. We have a hypothesis h which is always imperfect for 2 reasons:

- **Bias:** h is unable to fit f perfectly because it lacks capacity or complexity. (Fitting a line to a parabola).
- **Variance:** h is fitting to the noise in the data, thus missing the true function f.

Now, suppose we have an arbitrary point z, and a generated data point $\gamma = f(z) + \epsilon$.

Analyzing Loss

Since ϵ is a random variable, γ is now one as well. This means we can calculate statistics such as expected value and variance.

Analyzing Loss

Since ϵ is a random variable, γ is now one as well. This means we can calculate statistics such as expected value and variance.

- $\mathbb{E}[\gamma] = f(z)$ and $Var[\gamma] = Var[\epsilon]$.

Since ϵ is a random variable, γ is now one as well. This means we can calculate statistics such as expected value and variance.

- $\mathbb{E}[\gamma] = f(z)$ and $Var[\gamma] = Var[\epsilon]$.

Analyzing Loss

Since ϵ is a random variable, γ is now one as well. This means we can calculate statistics such as expected value and variance.

- $\mathbb{E}[\gamma] = f(z)$ and $Var[\gamma] = Var[\epsilon]$.

Let's analyze the Expectation of Loss when we use Mean-Squared Error as a cost function:

Since ϵ is a random variable, γ is now one as well. This means we can calculate statistics such as expected value and variance.

- $\mathbb{E}[\gamma] = f(z)$ and $Var[\gamma] = Var[\epsilon]$.

Let's analyze the Expectation of Loss when we use Mean-Squared Error as a cost function:

$$
\begin{aligned}
R(h) &= \mathbb{E}[(h(z) - \gamma)^2] \\
&= \mathbb{E}[h(z)^2] + \mathbb{E}[\gamma^2] - 2\mathbb{E}[\gamma h(z)]
\end{aligned}
$$

Analyzing Loss (Contd.)

- We note that γ and $h(z)$ are independent. This allows us to exploit the property: $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.

Analyzing Loss (Contd.)

- We note that γ and $h(z)$ are independent. This allows us to exploit the property: $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.
- Since $Var[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, we can substitute $\mathbb{E}[X^2] = Var[X] + \mathbb{E}[X]^2$ in the derivation

Analyzing Loss (Contd.)

- We note that γ and $h(z)$ are independent. This allows us to exploit the property: $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.
- Since $Var[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, we can substitute $\mathbb{E}[X^2] = Var[X] + \mathbb{E}[X]^2$ in the derivation

Analyzing Loss (Contd.)

- We note that γ and $h(z)$ are independent. This allows us to exploit the property: $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.
- Since $Var[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, we can substitute $\mathbb{E}[X^2] = Var[X] + \mathbb{E}[X]^2$ in the derivation

$$\mathbb{E}[h(z)^2] + \mathbb{E}[\gamma^2] - 2\mathbb{E}[\gamma h(z)]$$

Analyzing Loss (Contd.)

- We note that γ and $h(z)$ are independent. This allows us to exploit the property: $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.
- Since $Var[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, we can substitute $\mathbb{E}[X^2] = Var[X] + \mathbb{E}[X]^2$ in the derivation

$$\mathbb{E}[h(z)^2] + \mathbb{E}[\gamma^2] - 2\mathbb{E}[\gamma h(z)]$$

$$\begin{aligned}
&= Var[h(z)] + \mathbb{E}[h(z)]^2 + Var[\gamma] + \mathbb{E}[\gamma]^2 - 2\mathbb{E}[\gamma]\mathbb{E}[h(z)] \\
&= (\mathbb{E}[h(z)] - \mathbb{E}[\gamma])^2 + Var[h(z)] + Var[\gamma] \\
&= (\mathbb{E}[h(z)] - f(z))^2 + Var[h(z)] + Var[\epsilon]
\end{aligned}$$

From the last slide, we see three important quantities:

Bias, Variance, Irreducible Error

From the last slide, we see three important quantities:

- $(\mathbb{E}[h(z)] - f(z))^2$ is the squared **bias**. It is how much the model expects to differ from the real data distribution function.

Bias, Variance, Irreducible Error

From the last slide, we see three important quantities:

- $(\mathbb{E}[h(z)] - f(z))^2$ is the squared **bias**. It is how much the model expects to differ from the real data distribution function.
- $Var[h(z)]$ is the **variance**. The risk is explicitly composed partly by the variance of our model.

Bias, Variance, Irreducible Error

From the last slide, we see three important quantities:

- $(\mathbb{E}[h(z)] - f(z))^2$ is the squared **bias**. It is how much the model expects to differ from the real data distribution function.
- $Var[h(z)]$ is the **variance**. The risk is explicitly composed partly by the variance of our model.
- $Var[\epsilon]$ is the **irreducible error**. The noise put into the data is entirely out of our control.

Bias, Variance, Irreducible Error

From the last slide, we see three important quantities:

- $(\mathbb{E}[h(z)] - f(z))^2$ is the squared **bias**. It is how much the model expects to differ from the real data distribution function.
- $Var[h(z)]$ is the **variance**. The risk is explicitly composed partly by the variance of our model.
- $Var[\epsilon]$ is the **irreducible error**. The noise put into the data is entirely out of our control.

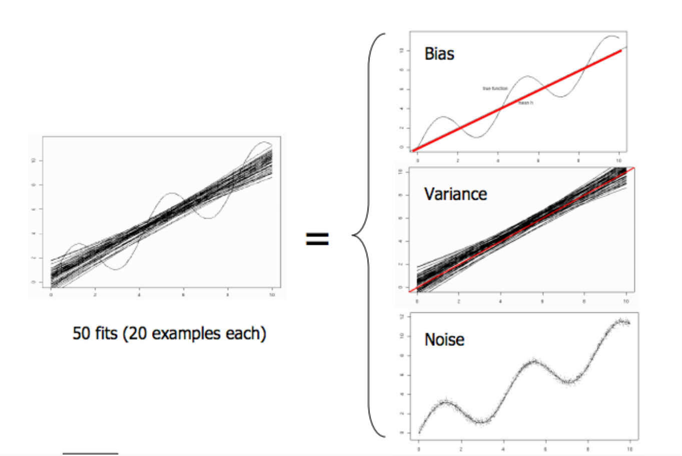
Bias, Variance, Irreducible Error

From the last slide, we see three important quantities:

- $(\mathbb{E}[h(z)] - f(z))^2$ is the squared **bias**. It is how much the model expects to differ from the real data distribution function.
- $Var[h(z)]$ is the **variance**. The risk is explicitly composed partly by the variance of our model.
- $Var[\epsilon]$ is the **irreducible error**. The noise put into the data is entirely out of our control.

Because we are able to break down the risk of any hypothesis like so, this process is called the **Bias-Variance Decomposition**.

Visualizing Each of the Quantities



Knowing the Effects of Each

Lots of issues and techniques center around adjusting or accounting for bias, variance and irreducible error.

Knowing the Effects of Each

Lots of issues and techniques center around adjusting or accounting for bias, variance and irreducible error.

- Underfitting directly relates to having too much bias.

Knowing the Effects of Each

Lots of issues and techniques center around adjusting or accounting for bias, variance and irreducible error.

- Underfitting directly relates to having too much bias.
- Overfitting is most often a result of too much variance.

Knowing the Effects of Each

Lots of issues and techniques center around adjusting or accounting for bias, variance and irreducible error.

- Underfitting directly relates to having too much bias.
- Overfitting is most often a result of too much variance.
- Training error shows bias, not variance. Test error reflects both.

Knowing the Effects of Each

Lots of issues and techniques center around adjusting or accounting for bias, variance and irreducible error.

- Underfitting directly relates to having too much bias.
- Overfitting is most often a result of too much variance.
- Training error shows bias, not variance. Test error reflects both.
- For many distributions, variance $\rightarrow 0$ as $n \rightarrow \infty$.

Knowing the Effects of Each

Lots of issues and techniques center around adjusting or accounting for bias, variance and irreducible error.

- Underfitting directly relates to having too much bias.
- Overfitting is most often a result of too much variance.
- Training error shows bias, not variance. Test error reflects both.
- For many distributions, variance $\rightarrow 0$ as $n \rightarrow \infty$.
- Assuming h has sufficient modeling capacity, bias $\rightarrow 0$ as $n \rightarrow \infty$

- **Adding features:** higher quality feature → higher drop in bias.

Knowing the Effects of Each (Contd.)

- **Adding features:** higher quality feature $\rightarrow$ higher drop in bias.
- *Variance always increases.* $\rightarrow$ only add feature if bias drop outweighs variance gain

Decision Trees vs. Other Models

We decision trees to linear and logistic regression, SVM's.

- **Bias:** Decision trees have highly nonlinear decision boundaries and its parameters can be arbitrarily complex.

Decision Trees vs. Other Models

We decision trees to linear and logistic regression, SVM's.

- **Bias:** Decision trees have highly nonlinear decision boundaries and its parameters can be arbitrarily complex.
  - Decision trees have relatively **low bias.**

Decision Trees vs. Other Models

We decision trees to linear and logistic regression, SVM's.

- **Bias:** Decision trees have highly nonlinear decision boundaries and its parameters can be arbitrarily complex.
    - Decision trees have relatively **low bias.**
- **Variance:** Decision tree leaves will be pure (without early stopping/pruning), so all noise is accounted for. Other regression models have clear regularization.

Decision Trees vs. Other Models

We decision trees to linear and logistic regression, SVM's.

- **Bias:** Decision trees have highly nonlinear decision boundaries and its parameters can be arbitrarily complex.
    - Decision trees have relatively **low bias.**
- **Variance:** Decision tree leaves will be pure (without early stopping/pruning), so all noise is accounted for. Other regression models have clear regularization.
    - Decision trees have relatively **high variance.**

Advantages of Random Forests

- **Bias:** Random Forests average decision tree decision boundaries, so they are no longer axis-aligned.

Advantages of Random Forests

- **Bias:** Random Forests average decision tree decision boundaries, so they are no longer axis-aligned.
- **Variance:** Random Forests average out decision trees.

$$Var[\frac{1}{n}(h_1(z) + ...h_n(z))] = \frac{1}{n^2} nVar[h_i(z)] = \frac{1}{n} Var[h_i(z)]$$

Advantages of Random Forests

- **Bias:** Random Forests average decision tree decision boundaries, so they are no longer axis-aligned.
- **Variance:** Random Forests average out decision trees.

$$Var[\frac{1}{n}(h_1(z) + ...h_n(z))] = \frac{1}{n^2}nVar[h_i(z)] = \frac{1}{n}Var[h_i(z)]$$

  - Because of the properties of Variance, Random Forests have **lower variance** than decision trees.

Key Takeaways

Especially because of variance, an **ensemble learner** comprised of several models that vote or average will generalize better.

- The main consideration simply becomes training costs

Key Takeaways

Especially because of variance, an **ensemble learner** comprised of several models that vote or average will generalize better.

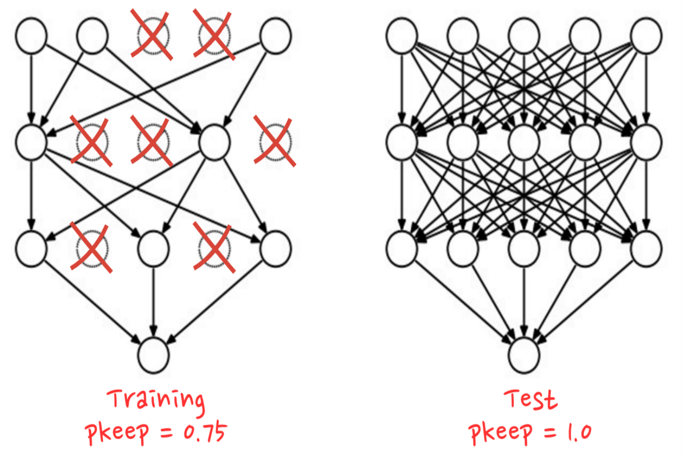
- The main consideration simply becomes training costs
- Prior to deep learning revival in 2010's, random forests and ensembled SVM's were state-of-the-art for data science classification, prediction.

Key Takeaways

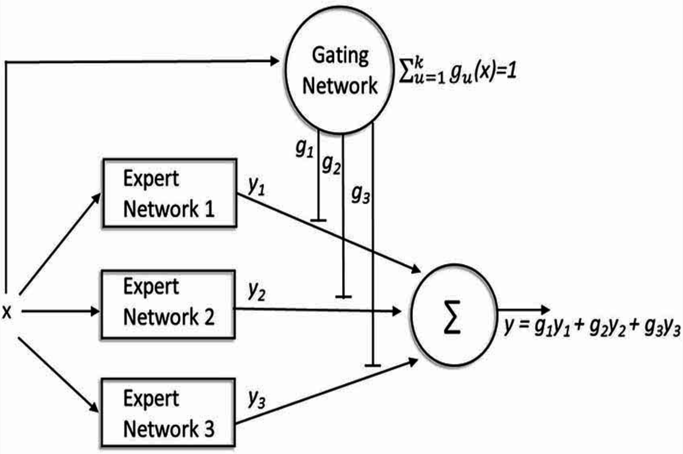
Especially because of variance, an **ensemble learner** comprised of several models that vote or average will generalize better.

- The main consideration simply becomes training costs
- Prior to deep learning revival in 2010's, random forests and ensembled SVM's were state-of-the-art for data science classification, prediction.
- Even now, one can ensemble deep learning models e.g. dropout, mixture of experts

Bonus: Dropout



Bonus: Mixture of Experts



Questions

Questions?