Structured Prediction & Decision Trees

Mengye Ren

(Slides credit to David Rosenberg, He He, et al.)

NYU

Nov 5, 2024

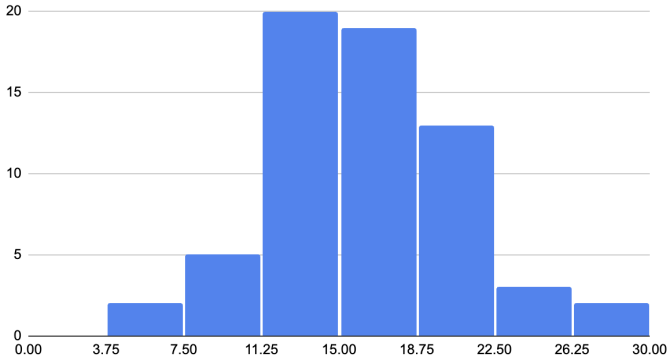
Slides

Final Project

- Project Consultation is mandatory. I am meeting with all 22 teams.
- Every member in the group needs to be present.
- If you have an emergency situation and cannot attend, you need to let me know.
- You may lose participation marks for not showing up in the consultation session without a reason.
- If you plan to show up, be on time. Your team member's time is also valuable.

Midterm

- Out of 30: Average: 16.05, Std: 4.29
- 9 free marks, capped at 30.
- Regrade request — email graders.



Scientific Writing

Introduction

- Introduction is the road map of the full report. Here is a general structure.
- Paragraph 1
  - What are some broad context of the problem? ✓
  - What is the problem that you are studying? ✓
  - Why is the problem interesting? ✓

Introduction

- Introduction is the road map of the full report. Here is a general structure.
- Paragraph 1
  - What are some broad context of the problem? ✓
  - What is the problem that you are studying? ✓
  - Why is the problem interesting? ✓
- Paragraph 2
  - Historically, what have people been doing in the space of similar problems?
  - What is the gap and what extra can this project bring?

Introduction

- Introduction is the road map of the full report. Here is a general structure.
- Paragraph 1
  - What are some broad context of the problem? ✓
  - What is the problem that you are studying? ✓
  - Why is the problem interesting? ✓
- Paragraph 2
  - Historically, what have people been doing in the space of similar problems?
  - What is the gap and what extra can this project bring?
- Paragraph 3
  - What technical approach are you taking?
  - What dataset have you experimented with and what are the core results?
  - What are some broader impact of the work?

Example

Introduction

Cybersecurity threats have become a major concern, posing risks to both personal data and organizational assets. In 2023, cyberattacks led to financial damages of over $10 billion in the United States alone, according to recent reports. Globally, these numbers were even more concerning, with the cost of cybercrime expected to surpass $8 trillion in 2023, as noted by cybersecurity researchers. In the U.S., approximately 66% of organizations experienced at least one form of cyberattack in 2023. These figures highlight an urgent need for robust strategies to detect, prevent, and mitigate cybersecurity breaches across industries.

Example

Introduction

Cybersecurity threats have become a major concern, posing risks to both personal data and organizational assets. In 2023, cyberattacks led to financial damages of over $10 billion in the United States alone, according to recent reports. Globally, these numbers were even more concerning, with the cost of cybercrime expected to surpass $8 trillion in 2023, as noted by cybersecurity researchers. In the U.S., approximately 66% of organizations experienced at least one form of cyberattack in 2023. These figures highlight an urgent need for robust strategies to detect, prevent, and mitigate cybersecurity breaches across industries.

What are the issues?

Revised Example

The rapid escalation in cybersecurity threats has led to significant financial and operational impacts, with damages in the United States alone exceeding $10 billion in 2023, according to the Cybersecurity & Infrastructure Security Agency (CISA) [1]. Despite advancements in cybersecurity solutions, traditional rule-based systems often struggle to detect new or sophisticated attack patterns, especially as cybercriminals evolve their techniques to evade static detection rules. This limitation highlights a critical gap: the need for adaptable, data-driven methods that can dynamically learn and respond to emerging threats.

Continue...

Revised Example

In response to this gap, our project applies machine learning to develop a robust intrusion detection system that leverages a hybrid approach combining supervised and unsupervised learning. Using a labeled dataset of network traffic, our model is first trained to distinguish normal from suspicious activity. To enhance adaptability, we also employ an anomaly detection module using autoencoders, which identifies deviations from typical patterns, even for attack types not present in the training data.

Our results demonstrate the effectiveness of our approach. The model achieved an accuracy of 94% in detecting known attack types and reduced false positives by 30% compared to a traditional rule-based system. Furthermore, the anomaly detection module identified previously unseen attack patterns with an 87% true positive rate. These results suggest that our machine learning-based system can offer a more flexible and accurate defense mechanism against evolving cybersecurity threats.

Abstract

Abstract is the summary of the introduction. Make it into one paragraph.

Cybersecurity threats are increasing in frequency and sophistication, resulting in substantial financial losses and operational disruptions. Traditional rule-based detection systems are limited in their ability to identify novel attack patterns, creating a need for adaptable, data-driven solutions. This project presents a machine learning-based intrusion detection system designed to address this gap by combining supervised learning for known threats with an unsupervised anomaly detection module to identify emerging attack types. Experimental results demonstrate the model's effectiveness, achieving 94% accuracy in detecting known attacks and an 87% true positive rate for novel threats, while reducing false positives by 30% compared to rule-based methods. These findings suggest that our hybrid approach provides a more flexible and accurate defense against evolving cyber threats.

Related Work

- Survey historical attempts of similar problems (not necessarily the same problem!)
- What categories do historical approaches span across? Which category does your method fall into?
- What is the relation of your work in the context of prior literature?

Example

Intrusion detection has been approached from several angles, with methods broadly categorized into rule-based systems, supervised learning-based intrusion detection, unsupervised anomaly detection, and hybrid models.

Rule-Based Detection: Traditional rule-based systems, such as Snort, use known attack signatures to detect intrusions [1]. While effective for known threats, these systems fail to recognize novel or evolving attacks [2]. Our approach seeks to address this limitation by incorporating machine learning for greater adaptability.

Supervised Machine Learning: Supervised models like support vector machines and neural networks are commonly used due to their high accuracy with labeled data, effectively identifying known attacks [3, 4]. However, their reliance on labeled datasets makes them less adaptable to new threats. Our project builds on supervised methods but adds an anomaly detection layer to handle unknown attacks.

Continue...

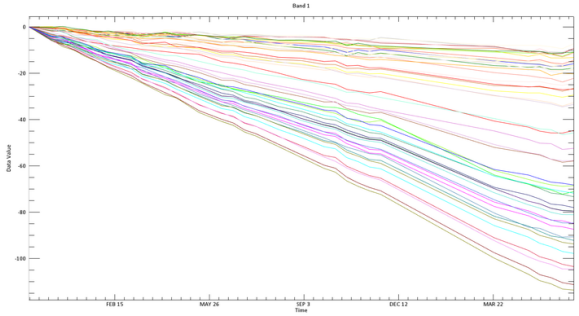
Example

Unsupervised and Anomaly-Based Detection: Unsupervised methods, including autoencoders and clustering, detect unusual patterns in network traffic without labeled data [6, 7]. While flexible, these techniques often produce higher false-positive rates, posing practical challenges [8]. Our model incorporates an autoencoder for anomaly detection, with optimized thresholds to reduce false positives.

Hybrid Approaches: Hybrid models combine supervised and unsupervised techniques to balance accuracy and adaptability. For example, hybrid models like those by Huang et al. (2021) integrate anomaly detection within supervised frameworks, achieving lower false-positive rates [9]. Our work further advances these methods by embedding an autoencoder anomaly module into a supervised model with optimized thresholds, aiming for reliable real-time intrusion detection.

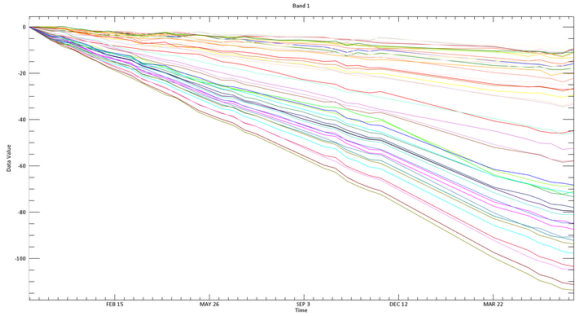
Figures and Tables

- When there is a strong trend, use a figure.



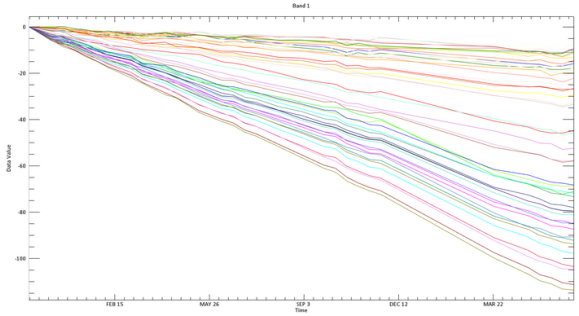
Figures and Tables

- When there is a strong trend, use a figure.
- Try to export as PDF instead of JPG/PNG (rasterized).



Figures and Tables

- When there is a strong trend, use a figure.
- Try to export as PDF instead of JPG/PNG (rasterized).
- Use bigger font size in the figure (same or slightly smaller than the main text).



Figures and Tables

- When you need to emphasize a small difference, use a table.

Table 4: Ablation studies on PooDLe components, reporting mIoU on BDD100K semantic segmentation linear readout. Rows without top-down follow FlowE (Xiong et al., 2021), replacing pooling with dilated convolutions to maintain spatial extent. †Flow model trained without supervised labels.

Variant	Dense	Pool	Top-Down	Lateral	Flow	All	Small	Large	Rare	Common
1 FlowE	✓				RAFT	28.8	8.7	40.5	1.8	29.2
2	✓	✓			RAFT	28.9	7.2	41.6	2.2	28.7
3	✓	✓	✓		RAFT	30.3	6.8	44.0	4.3	30.2
4	✓	✓		✓	RAFT	30.3	10.9	41.7	2.4	31.1
5	✓		✓	✓	RAFT	31.8	12.8	42.8	8.3	31.7
6 PooDLe†	✓	✓	✓	✓	UFlow	33.7	14.1	45.1	8.9	33.8
7 PooDLe	✓	✓	✓	✓	RAFT	**34.2**	**15.0**	**45.5**	**9.0**	**34.5**

Figures and Tables

- When you need to emphasize a small difference, use a table.
- When the table is too wide, use `resizebox` to fit your table with the text width.

Table 4: Ablation studies on PooDLe components, reporting mIoU on BDD100K semantic segmentation linear readout. Rows without top-down follow FlowE (Xiong et al., 2021), replacing pooling with dilated convolutions to maintain spatial extent. †Flow model trained without supervised labels.

Variant	Dense	Pool	Top-Down	Lateral	Flow	All	Small	Large	Rare	Common
1 FlowE	✓				RAFT	28.8	8.7	40.5	1.8	29.2
2	✓	✓			RAFT	28.9	7.2	41.6	2.2	28.7
3	✓	✓	✓		RAFT	30.3	6.8	44.0	4.3	30.2
4	✓	✓		✓	RAFT	30.3	10.9	41.7	2.4	31.1
5	✓		✓	✓	RAFT	31.8	12.8	42.8	8.3	31.7
6 PooDLe†	✓	✓	✓	✓	UFlow	33.7	14.1	45.1	8.9	33.8
7 PooDLe	✓	✓	✓	✓	RAFT	**34.2**	**15.0**	**45.5**	**9.0**	**34.5**

Figures and Tables

- When you need to emphasize a small difference, use a table.
- When the table is too wide, use `resizebox` to fit your table with the text width.
- Keep the same number of significant digits.

Table 4: Ablation studies on PooDLe components, reporting mIoU on BDD100K semantic segmentation linear readout. Rows without top-down follow FlowE (Xiong et al., 2021), replacing pooling with dilated convolutions to maintain spatial extent. †Flow model trained without supervised labels.

Variant	Dense	Pool	Top-Down	Lateral	Flow	All	Small	Large	Rare	Common
1 FlowE	✓				RAFT	28.8	8.7	40.5	1.8	29.2
2	✓	✓			RAFT	28.9	7.2	41.6	2.2	28.7
3	✓	✓	✓		RAFT	30.3	6.8	44.0	4.3	30.2
4	✓	✓		✓	RAFT	30.3	10.9	41.7	2.4	31.1
5	✓		✓	✓	RAFT	31.8	12.8	42.8	8.3	31.7
6 PooDLe†	✓	✓	✓	✓	UFlow	33.7	14.1	45.1	8.9	33.8
7 PooDLe	✓	✓	✓	✓	RAFT	**34.2**	**15.0**	**45.5**	**9.0**	**34.5**

Figures and Tables

- When you need to emphasize a small difference, use a table.
- When the table is too wide, use `resizebox` to fit your table with the text width.
- Keep the same number of significant digits.
- Explain the Figure and Table in the caption.

Table 4: Ablation studies on PooDLe components, reporting mIoU on BDD100K semantic segmentation linear readout. Rows without top-down follow FlowE (Xiong et al., 2021), replacing pooling with dilated convolutions to maintain spatial extent. †Flow model trained without supervised labels.

Variant	Dense	Pool	Top-Down	Lateral	Flow	All	Small	Large	Rare	Common
1 FlowE	✓				RAFT	28.8	8.7	40.5	1.8	29.2
2	✓	✓			RAFT	28.9	7.2	41.6	2.2	28.7
3	✓	✓	✓		RAFT	30.3	6.8	44.0	4.3	30.2
4	✓	✓		✓	RAFT	30.3	10.9	41.7	2.4	31.1
5	✓		✓	✓	RAFT	31.8	12.8	42.8	8.3	31.7
6 PooDLe†	✓	✓	✓	✓	UFlow	33.7	14.1	45.1	8.9	33.8
7 PooDLe	✓	✓	✓	✓	RAFT	**34.2**	**15.0**	**45.5**	**9.0**	**34.5**

Figures and Tables

- When you need to emphasize a small difference, use a table.
- When the table is too wide, use `resizebox` to fit your table with the text width.
- Keep the same number of significant digits.
- Explain the Figure and Table in the caption.
- Nice to have: Bold the column/row and best numbers. Highlight the important rows.

Table 4: Ablation studies on PooDLe components, reporting mIoU on BDD100K semantic segmentation linear readout. Rows without top-down follow FlowE (Xiong et al., 2021), replacing pooling with dilated convolutions to maintain spatial extent. †Flow model trained without supervised labels.

Variant	Dense	Pool	Top-Down	Lateral	Flow	All	Small	Large	Rare	Common
1 FlowE	✓				RAFT	28.8	8.7	40.5	1.8	29.2
2	✓	✓			RAFT	28.9	7.2	41.6	2.2	28.7
3	✓	✓	✓		RAFT	30.3	6.8	44.0	4.3	30.2
4	✓	✓		✓	RAFT	30.3	10.9	41.7	2.4	31.1
5	✓		✓	✓	RAFT	31.8	12.8	42.8	8.3	31.7
6 PooDLe†	✓	✓	✓	✓	UFlow	33.7	14.1	45.1	8.9	33.8
7 PooDLe	✓	✓	✓	✓	RAFT	**34.2**	**15.0**	**45.5**	**9.0**	**34.5**

Introduction to Structured Prediction

Example: Part-of-speech (POS) Tagging

- Given a sentence, give a part of speech tag for each word:

x	$\underbrace{\text{[START]}}_{x_0}$	$\underbrace{\text{He}}_{x_1}$	$\underbrace{\text{eats}}_{x_2}$	$\underbrace{\text{apples}}_{x_3}$
y	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

Example: Part-of-speech (POS) Tagging

- Given a sentence, give a part of speech tag for each word:

x	$\underbrace{\text{[START]}}_{x_0}$	$\underbrace{\text{He}}_{x_1}$	$\underbrace{\text{eats}}_{x_2}$	$\underbrace{\text{apples}}_{x_3}$
y	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

Example: Action grounding from long-form videos

- Given a long video, segment the video into short windows where each window corresponds to an action from a list of actions.
- E.g. slicing, chopping, frying, washing, etc.

Multiclass Hypothesis Space

- Discrete output space: $\mathcal{Y}(x)$
  - Very large but has structure, e.g., linear chain (sequence labeling), tree (parsing)
  - Size depends on input x

Multiclass Hypothesis Space

- Discrete output space: $\mathcal{Y}(x)$
  - Very large but has structure, e.g., linear chain (sequence labeling), tree (parsing)
  - Size depends on input x
- Base Hypothesis Space: $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \rightarrow \mathrm{R}\}$
  - $h(x, y)$ gives compatibility score between input x and output y

Multiclass Hypothesis Space

- Discrete output space: $\mathcal{Y}(x)$
  - Very large but has structure, e.g., linear chain (sequence labeling), tree (parsing)
  - Size depends on input x
- Base Hypothesis Space: $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \to \mathsf{R}\}$
  - $h(x, y)$ gives compatibility score between input x and output y
- Multiclass hypothesis space

$$\mathcal{F} = \left\{ x \mapsto \arg\max_{y \in \mathcal{Y}} h(x, y) \mid h \in \mathcal{H} \right\}$$

- Final prediction function is an $f \in \mathcal{F}$.
- For each $f \in \mathcal{F}$ there is an underlying compatibility score function $h \in \mathcal{H}$.

Structured Prediction

- Part-of-speech tagging

x:	he	eats	apples
y:	pronoun	verb	noun

- Multiclass hypothesis space:

$$h(x, y) = w^T \Psi(x, y) \tag{1}$$

$$\mathcal{F} = \left\{ x \mapsto \arg\max_{y \in \mathcal{Y}} h(x, y) \mid h \in \mathcal{H} \right\} \tag{2}$$

- A special case of multiclass classification
- How to design the feature map Ψ? What are the considerations?

Unary features

- A **unary feature** only depends on
  - the label at a single position, y_i, and x
- Example:

$$\begin{aligned}
\phi_1(x, y_i) &= \mathbb{1}[x_i = \text{runs}]\mathbb{1}[y_i = \text{Verb}] \\
\phi_2(x, y_i) &= \mathbb{1}[x_i = \text{runs}]\mathbb{1}[y_i = \text{Noun}] \\
\phi_3(x, y_i) &= \mathbb{1}[x_{i-1} = \text{He}]\mathbb{1}[x_i = \text{runs}]\mathbb{1}[y_i = \text{Verb}]
\end{aligned}$$

Markov features

- A **markov feature** only depends on
  - two adjacent labels, y_{i-1} and y_i, and x
- Example:

$$\begin{aligned}\theta_1(x, y_{i-1}, y_i) &= \mathbb{1}[y_{i-1} = \text{Pronoun}]\,\mathbb{1}[y_i = \text{Verb}] \\ \theta_2(x, y_{i-1}, y_i) &= \mathbb{1}[y_{i-1} = \text{Pronoun}]\,\mathbb{1}[y_i = \text{Noun}]\end{aligned}$$

- Reminiscent of Markov models in the output space
- Possible to have higher-order features

Local Feature Vector and Compatibility Score

- At each position i in sequence, define the **local feature vector** (unary and markov):

$$\begin{aligned}\Psi_i(x, y_{i-1}, y_i) &= (\phi_1(x, y_i), \phi_2(x, y_i), \ldots, \\ &\quad \theta_1(x, y_{i-1}, y_i), \theta_2(x, y_{i-1}, y_i), \ldots)\end{aligned}$$

Local Feature Vector and Compatibility Score

- At each position i in sequence, define the **local feature vector** (unary and markov):

$$\begin{aligned}\Psi_i(x, y_{i-1}, y_i) &= (\phi_1(x, y_i), \phi_2(x, y_i), \ldots, \\ &\quad \theta_1(x, y_{i-1}, y_i), \theta_2(x, y_{i-1}, y_i), \ldots)\end{aligned}$$

- And **local compatibility score** at position i: $\langle w, \Psi_i(x, y_{i-1}, y_i) \rangle$.

Local Feature Vector and Compatibility Score

- At each position i in sequence, define the **local feature vector** (unary and markov):

$$\begin{aligned}\Psi_i(x, y_{i-1}, y_i) &= (\phi_1(x, y_i), \phi_2(x, y_i), \ldots, \\ &\quad \theta_1(x, y_{i-1}, y_i), \theta_2(x, y_{i-1}, y_i), \ldots)\end{aligned}$$

- And **local compatibility score** at position i: $\langle w, \Psi_i(x, y_{i-1}, y_i) \rangle$.

- The compatibility score for (x, y) is the sum of local compatibility scores:

$$\sum_i \langle w, \Psi_i(x, y_{i-1}, y_i) \rangle = \left\langle w, \sum_i \Psi_i(x, y_{i-1}, y_i) \right\rangle = \langle w, \Psi(x, y) \rangle , \tag{3}$$

where we define the **sequence feature vector** by

$$\Psi(x, y) = \sum_i \Psi_i(x, y_{i-1}, y_i). \qquad \text{decomposable}$$

Structured perceptron

Given a dataset $\mathcal{D} = \{(x, y)\}$;
Initialize $w \leftarrow 0$;
for *iter* $= 1, 2, \ldots, T$ **do**
  for $(x, y) \in \mathcal{D}$ **do**
    $\hat{y} = \arg\max_{y' \in \mathcal{Y}(x)} w^T \psi(x, y')$;
    if $\hat{y} \neq y$ **then** `// We've made a mistake`
      $w \leftarrow w + \Psi(x, y)$; `// Move the scorer towards` $\psi(x, y)$
      $w \leftarrow w - \Psi(x, \hat{y})$; `// Move the scorer away from` $\psi(x, \hat{y})$
    end
  end
end

Structured perceptron

Given a dataset $\mathcal{D} = \{(x, y)\}$;
Initialize $w \leftarrow 0$;
for *iter* $= 1, 2, \ldots, T$ **do**
  for $(x, y) \in \mathcal{D}$ **do**
    $\hat{y} = \arg\max_{y' \in \mathcal{Y}(x)} w^T \psi(x, y')$;
    if $\hat{y} \neq y$ **then** `// We've made a mistake`
      $w \leftarrow w + \Psi(x, y)$; `// Move the scorer towards` $\psi(x, y)$
      $w \leftarrow w - \Psi(x, \hat{y})$; `// Move the scorer away from` $\psi(x, \hat{y})$
    end
  end
end

Identical to the multiclass perceptron algorithm except the arg max is now over the structured output space $\mathcal{Y}(x)$.

Structured hinge loss

- Recall the generalized hinge loss

$$\ell_{\text{hinge}}(y, \hat{y}) \stackrel{\text{def}}{=} \max_{y' \in \mathcal{Y}(x)} \left(\Delta(y, y') + \left\langle w, \left(\Psi(x, y') - \Psi(x, y) \right) \right\rangle \right) \tag{4}$$

- What is $\Delta(y, y')$ for two sequences?

Structured hinge loss

- Recall the generalized hinge loss

$$\ell_{\text{hinge}}(y, \hat{y}) \stackrel{\text{def}}{=} \max_{y' \in \mathcal{Y}(x)} \left(\Delta(y, y') + \left\langle w, \left(\Psi(x, y') - \Psi(x, y) \right) \right\rangle \right) \tag{4}$$

- What is $\Delta(y, y')$ for two sequences?
- **Hamming loss** is common:

$$\Delta(y, y') = \frac{1}{L} \sum_{i=1}^{L} \mathbb{1}[y_i \neq y_i']$$

where L is the sequence length.

Structured SVM

Exercise:

- Write down the objective of structured SVM using the structured hinge loss.
- Stochastic sub-gradient descent for structured SVM (similar to HW3 P3)
- Compare with the structured perceptron algorithm

The argmax problem for sequences

Problem To compute predictions, we need to find $\arg\max_{y \in \mathcal{Y}(x)} \langle w, \Psi(x, y) \rangle$, and $|\mathcal{Y}(x)|$ is exponentially large.

Figure by Daumé III. A course in machine learning. Figure 17.1.

The argmax problem for sequences

Problem To compute predictions, we need to find $\arg\max_{y \in \mathcal{Y}(x)} \langle w, \Psi(x, y) \rangle$, and $|\mathcal{Y}(x)|$ is exponentially large.

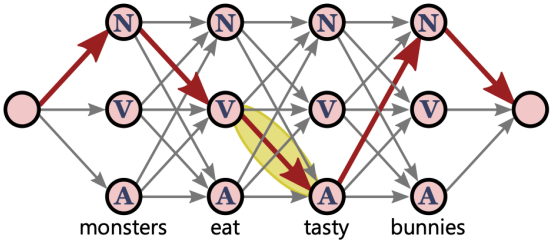
Observation $\Psi(x, y)$ decomposes to $\sum_i \Psi_i(x, y)$.

Figure by Daumé III. A course in machine learning. Figure 17.1.

The argmax problem for sequences

Problem To compute predictions, we need to find $\arg\max_{y \in \mathcal{Y}(x)} \langle w, \Psi(x, y) \rangle$, and $|\mathcal{Y}(x)|$ is exponentially large.

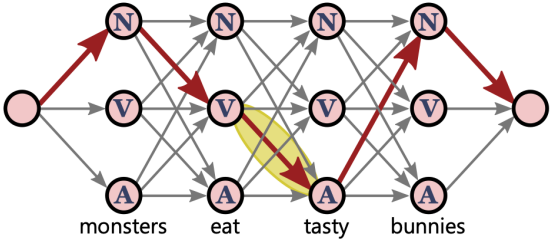
Observation $\Psi(x, y)$ decomposes to $\sum_i \Psi_i(x, y)$.



Solution Dynamic programming

Figure by Daumé III. A course in machine learning. Figure 17.1.

Dynamic Programming (MAP inference)



Conditional random field (CRF)

- Recall that we can write logistic regression in a general form:

$$p(y|x) = \frac{1}{Z(x)} \exp(w^\top \psi(x, y)).$$

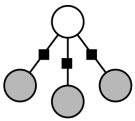
- Z is normalization constant: $Z(x) = \sum_{y \in Y} \exp(w^\top \psi(x, y))$.

Conditional random field (CRF)

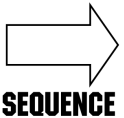
- Recall that we can write logistic regression in a general form:

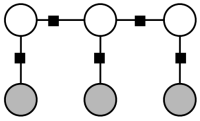
$$p(y|x) = \frac{1}{Z(x)} \exp(w^\top \psi(x, y)).$$

- Z is normalization constant: $Z(x) = \sum_{y \in Y} \exp(w^\top \psi(x, y))$.
- Example: linear chain $\{y_t\}$
- We can incorporate unary and Markov features: $p(y|x) = \frac{1}{Z(x)} \exp(\sum_t w^\top \psi(x, y_t, y_{t-1}))$



Logistic Regression





Linear-chain CRFs

Conditional random field (CRF)

- Compared to Structured SVM, CRF has a probabilistic interpretation.
- We can draw samples in the output space.

Conditional random field (CRF)

- Compared to Structured SVM, CRF has a probabilistic interpretation.
- We can draw samples in the output space.
- How do we learn w? Maximum log likelihood, and regularization term: $\lambda \|w\|^2$
- Loss function:

$$
\begin{aligned}
l(w) &= -\frac{1}{N} \sum_{i=1}^{N} \log p(y^{(i)} | x^{(i)}) + \frac{1}{2} \lambda \|w\|^2 \\
&= -\frac{1}{N} \sum_{i} \sum_{t} \sum_{k} w_k \psi_k(y_t^{(i)}, y_{t-1}^{(i)}) + \frac{1}{N} \sum_{i} \log Z(x^{(i)}) + \frac{1}{2} \sum_{k} \lambda w_k^2
\end{aligned}
$$

Conditional random field (CRF)

- Loss function:

$$l(w) = -\frac{1}{N}\sum_i \sum_t \sum_k w_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)}) + \frac{1}{N}\sum_i \log Z(x^{(i)}) + \frac{1}{2}\sum_k \lambda w_k^2$$

- Gradient:

$$\frac{\partial l(w)}{\partial w_k} = -\frac{1}{N}\sum_i \sum_t \sum_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)}) \tag{5}$$

$$+ \frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp\Big(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y_t', y_{t-1}')\Big) + \sum_k \lambda w_k \tag{6}$$

Conditional random field (CRF)

- What is $\frac{1}{N}\sum_i \sum_t \sum_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)})$?

Conditional random field (CRF)

- What is $\frac{1}{N}\sum_i \sum_t \sum_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)})$?
- It is the expectation $\psi_k(x^{(i)}, y_t, y_{t-1})$ under the empirical distribution $\tilde{p}(x, y) = \frac{1}{N}\sum_i \mathbb{1}[x = x^{(i)}]\mathbb{1}[y = y^{(i)}]$.

Conditional random field (CRF)

- What is $\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1}))$?

Conditional random field (CRF)

- What is $\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1}))$?

$$= \frac{1}{N} \sum_i \sum_t \sum_{y' \in Y} p(y'_t, y'_{t-1}|x) \psi_k(x^{(i)}, y'_t, y'_{t-1}) \tag{7}$$

Conditional random field (CRF)

- What is $\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1}))$?

$$= \frac{1}{N} \sum_i \sum_t \sum_{y' \in Y} p(y'_t, y'_{t-1}|x) \psi_k(x^{(i)}, y'_t, y'_{t-1}) \tag{7}$$

- It is the expectation of $\psi_k(x^{(i)}, y'_t, y'_{t-1})$ under the model distribution $p(y'_t, y'_{t-1}|x)$.

Conditional random field (CRF)

- To compute the gradient, we need to infer expectation under the model distribution $p(y|x)$.
- We contrast between the data expectation and the model expectation.

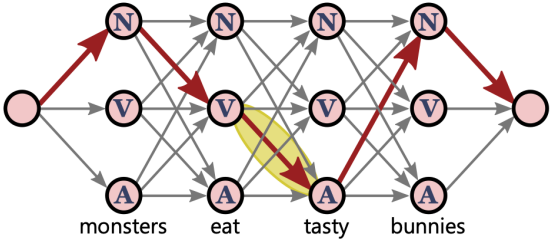
Conditional random field (CRF)

- To compute the gradient, we need to infer expectation under the model distribution $p(y|x)$.
- We contrast between the data expectation and the model expectation.
- Compare the learning algorithms: in structured SVM we need to compute the argmax, whereas in CRF we need to compute the model expectation.

Conditional random field (CRF)

- To compute the gradient, we need to infer expectation under the model distribution $p(y|x)$.
- We contrast between the data expectation and the model expectation.
- Compare the learning algorithms: in structured SVM we need to compute the argmax, whereas in CRF we need to compute the model expectation.
- Both problems are NP-hard for general graphs.

CRF Inference



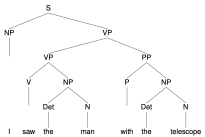
- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- The inference algorithm can be generalized to belief propagation (BP) in a tree structure (exact inference).
- In general graphs, we rely on approximate inference (e.g. loopy belief propagation).

Examples

- POS tag Relationship between constituents, e.g. NP is likely to be followed by a VP.

x → $\mathbf{y} = (y_1, \ldots, y_D)$

I saw the man with the telescope.



x → $\mathbf{y} = (y_1, \ldots, y_D)$

sky/plant life/tree
sky/plant life/tree

water/animals/sea
water/animals/sky

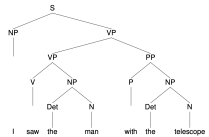
Chen et al. Learning Deep Structured Models. ICML 2015.

Examples

- POS tag Relationship between constituents, e.g. NP is likely to be followed by a VP.
- Semantic segmentation
  Relationship between pixels, e.g. a grass pixel is likely to be next to another grass pixel.

x → $\mathbf{y} = (y_1, \ldots, y_D)$

I saw the man with the telescope.



x → $\mathbf{y} = (y_1, \ldots, y_D)$

sky/plant life/tree
sky/plant life/tree

water/animals/sea
water/animals/sky

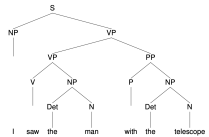
Chen et al. Learning Deep Structured Models. ICML 2015.

Examples

- POS tag Relationship between constituents, e.g. NP is likely to be followed by a VP.
- Semantic segmentation
  Relationship between pixels, e.g. a grass pixel is likely to be next to another grass pixel.
- Multi-label learning
  An image may contain multiple class labels, e.g. water is likely to co-occur with fish.

x → $\mathbf{y} = (y_1, \ldots, y_D)$

I saw the man with the telescope.



x → $\mathbf{y} = (y_1, \ldots, y_D)$

sky/plant life/tree
sky/plant life/tree

water/animals/sea
water/animals/sky

Chen et al. Learning Deep Structured Models. ICML 2015.

Conclusion

Structured Prediction

- Extension to multi-class prediction
- Structured SVM, CRF
- Output space containing structure
- Text and image applications

Decision Trees

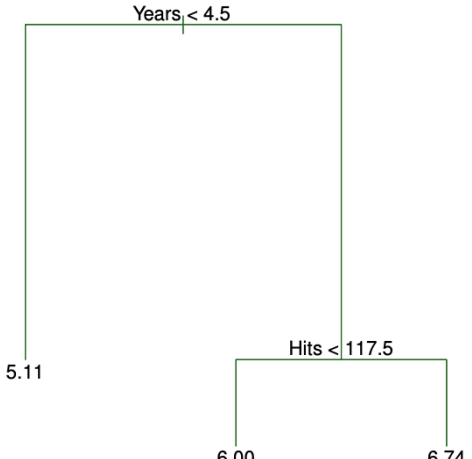
Overview: Decision Trees

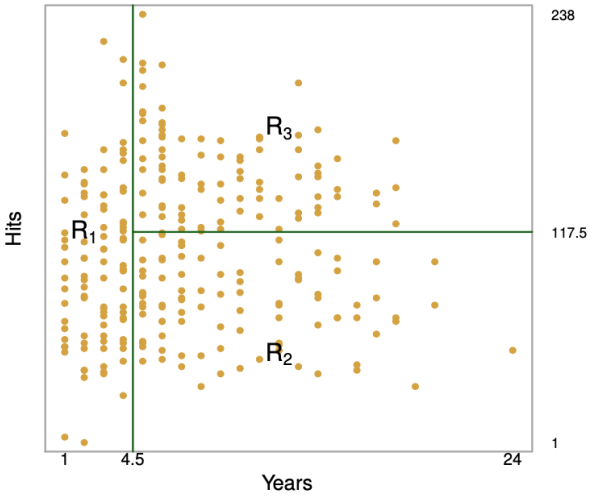
- Our first inherently non-linear classifier: decision trees.
- Ensemble methods: bagging and boosting.

Decision Trees

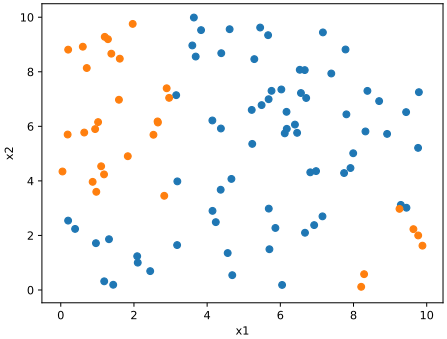
Regression trees: Predicting basketball players' salaries

Regression trees: Predicting basketball players' salaries



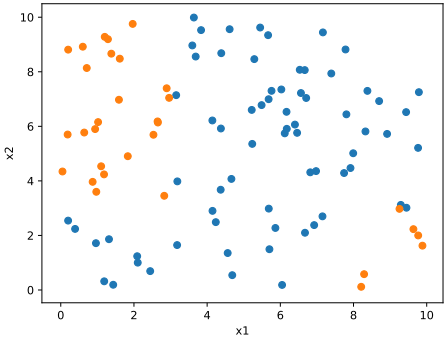


Classification trees



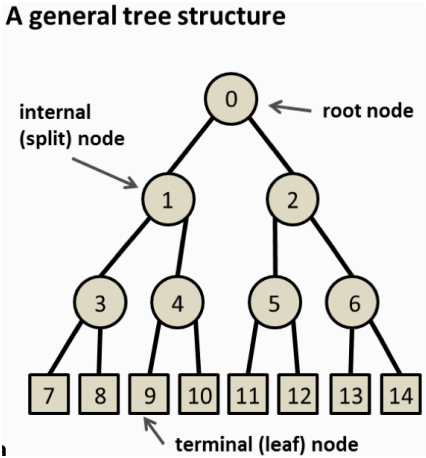
- Can we classify these points using a linear classifier?

Classification trees



- Can we classify these points using a linear classifier?
- Partition the data into axis-aligned regions recursively (on the board)

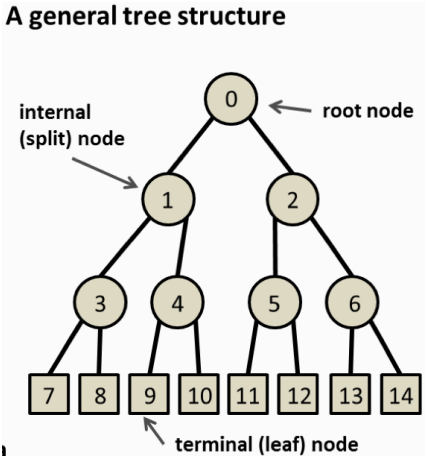
Decision trees setup



- We focus on *binary* trees (as opposed to multiway trees where nodes can have more than two children)

From Criminisi et al. MSR-TR-2011-114, 28 October 2011.

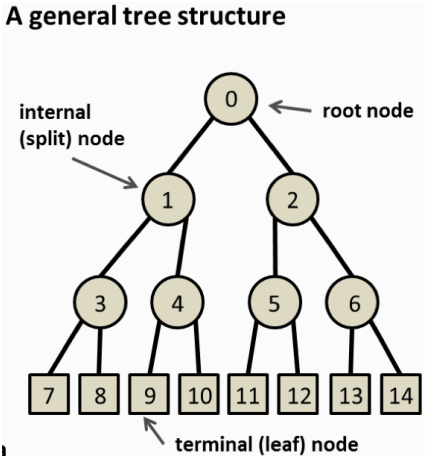
Decision trees setup



- We focus on *binary* trees (as opposed to multiway trees where nodes can have more than two children)
- Each node contains a subset of data points

From Criminisi et al. MSR-TR-2011-114, 28 October 2011.

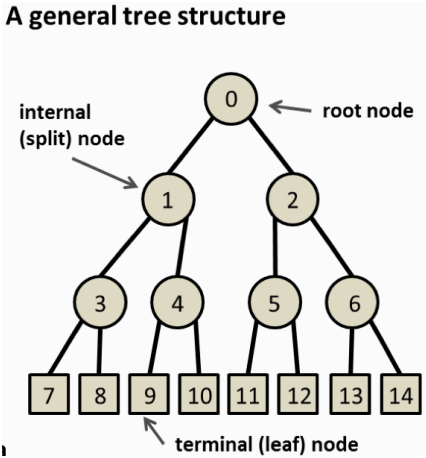
Decision trees setup



- We focus on *binary* trees (as opposed to multiway trees where nodes can have more than two children)
- Each node contains a subset of data points
- The data splits created by each node involve only a *single* feature

From Criminisi et al. MSR-TR-2011-114, 28 October 2011.

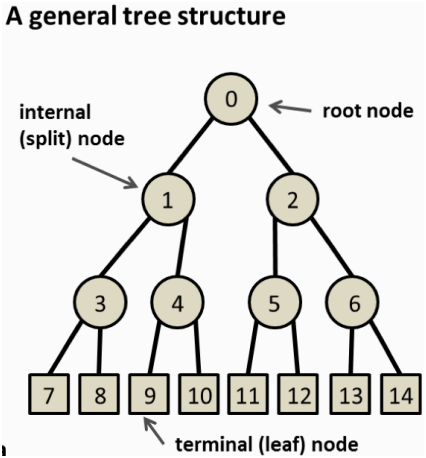
Decision trees setup



- We focus on *binary* trees (as opposed to multiway trees where nodes can have more than two children)
- Each node contains a subset of data points
- The data splits created by each node involve only a *single* feature
- For continuous variables, the splits are always of the form $x_i \leqslant t$

From Criminisi et al. MSR-TR-2011-114, 28 October 2011.

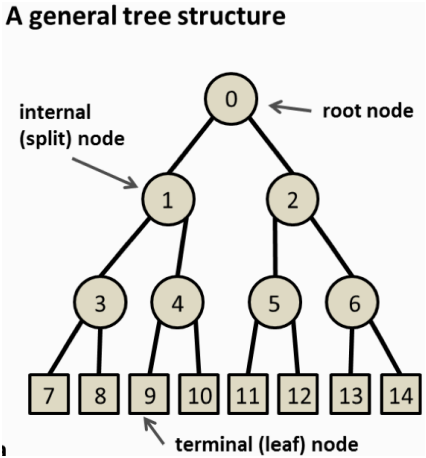
Decision trees setup



- We focus on *binary* trees (as opposed to multiway trees where nodes can have more than two children)
- Each node contains a subset of data points
- The data splits created by each node involve only a *single* feature
- For continuous variables, the splits are always of the form $x_i \leqslant t$
- For discrete variables, we partition values into two sets (not covered today)

From Criminisi et al. MSR-TR-2011-114, 28 October 2011.

Decision trees setup



From Criminisi et al. MSR-TR-2011-114, 28 October 2011.

- We focus on *binary* trees (as opposed to multiway trees where nodes can have more than two children)
- Each node contains a subset of data points
- The data splits created by each node involve only a *single* feature
- For continuous variables, the splits are always of the form $x_i \leqslant t$
- For discrete variables, we partition values into two sets (not covered today)
- Predictions are made in terminal nodes

Constructing the tree

Goal Find boxes $R_1, \ldots, R_J$ that minimize $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$, subject to complexity constraints.

Constructing the tree

Goal Find boxes $R_1, \ldots, R_J$ that minimize $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$, subject to complexity constraints.

Problem Finding the optimal binary tree is computationally intractable.

Constructing the tree

Goal Find boxes $R_1, \ldots, R_J$ that minimize $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$, subject to complexity constraints.

Problem Finding the optimal binary tree is computationally intractable.

Solution Greedy algorithm: starting from the root, and repeating until a stopping criterion is reached (e.g., max depth), find the non-terminal node that results in the "best" split

Constructing the tree

Goal Find boxes $R_1, \ldots, R_J$ that minimize $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$, subject to complexity constraints.

Problem Finding the optimal binary tree is computationally intractable.

Solution Greedy algorithm: starting from the root, and repeating until a stopping criterion is reached (e.g., max depth), find the non-terminal node that results in the "best" split

- We only split regions defined by previous non-terminal nodes

Constructing the tree

Goal Find boxes $R_1, \ldots, R_J$ that minimize $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$, subject to complexity constraints.

Problem Finding the optimal binary tree is computationally intractable.

Solution Greedy algorithm: starting from the root, and repeating until a stopping criterion is reached (e.g., max depth), find the non-terminal node that results in the "best" split

- We only split regions defined by previous non-terminal nodes

Prediction Our prediction is the mean value of a terminal node: $\hat{y}_{R_m} = \text{mean}(y_i \mid x_i \in R_m)$

Constructing the tree

Goal Find boxes $R_1, \ldots, R_J$ that minimize $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$, subject to complexity constraints.

Problem Finding the optimal binary tree is computationally intractable.

Solution Greedy algorithm: starting from the root, and repeating until a stopping criterion is reached (e.g., max depth), find the non-terminal node that results in the "best" split

- We only split regions defined by previous non-terminal nodes

Prediction Our prediction is the mean value of a terminal node: $\hat{y}_{R_m} = \text{mean}(y_i \mid x_i \in R_m)$

- A greedy algorithm is the one that make the best **local** decisions, without lookahead to evaluate their downstream consequences

Constructing the tree

Goal Find boxes $R_1, \ldots, R_J$ that minimize $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$, subject to complexity constraints.

Problem Finding the optimal binary tree is computationally intractable.

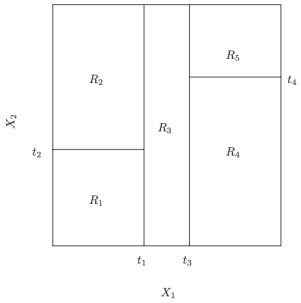
Solution Greedy algorithm: starting from the root, and repeating until a stopping criterion is reached (e.g., max depth), find the non-terminal node that results in the "best" split

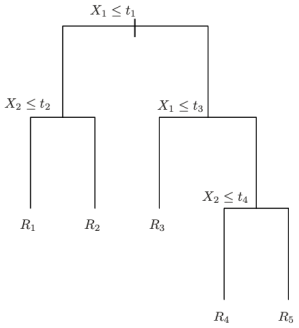
- We only split regions defined by previous non-terminal nodes

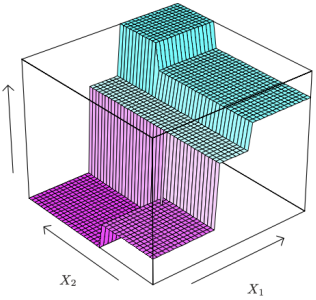
Prediction Our prediction is the mean value of a terminal node: $\hat{y}_{R_m} = \text{mean}(y_i \mid x_i \in R_m)$

- A greedy algorithm is the one that make the best **local** decisions, without lookahead to evaluate their downstream consequences
- This procedure is not very likely to result in the globally optimal tree

Prediction in a Regression Tree







Finding the Best Split Point

- We enumerate all features and all possible split points for each feature. There are infinitely many split points, but...

Finding the Best Split Point

- We enumerate all features and all possible split points for each feature. There are infinitely many split points, but...
- Suppose we are now considering splitting on the j-th feature x_j, and let $x_{j(1)}, \ldots, x_{j(n)}$ be the sorted values of the j-th feature.

Finding the Best Split Point

- We enumerate all features and all possible split points for each feature. There are infinitely many split points, but...
- Suppose we are now considering splitting on the j-th feature x_j, and let $x_{j(1)}, \ldots, x_{j(n)}$ be the sorted values of the j-th feature.
- We only need to consider split points between two adjacent values, and any split point in the interval $(x_{j(r)}, x_{(j(r+1)})$ will result in the same loss

Finding the Best Split Point

- We enumerate all features and all possible split points for each feature. There are infinitely many split points, but...
- Suppose we are now considering splitting on the j-th feature x_j, and let $x_{j(1)}, \ldots, x_{j(n)}$ be the sorted values of the j-th feature.
- We only need to consider split points between two adjacent values, and any split point in the interval $(x_{j(r)}, x_{(j(r+1)})$ will result in the same loss
- It is common to split half way between two adjacent values:

$$s_j \in \left\{ \frac{1}{2} \left(x_{j(r)} + x_{j(r+1)} \right) \mid r = 1, \ldots, n-1 \right\}. \qquad n-1 \text{ splits} \tag{8}$$

Decision Trees and Overfitting

- What will happen if we keep splitting the data into more and more regions?

Decision Trees and Overfitting

- What will happen if we keep splitting the data into more and more regions?
  - Every data point will be in its own region—overfitting.

Decision Trees and Overfitting

- What will happen if we keep splitting the data into more and more regions?
  - Every data point will be in its own region—overfitting.
- When should we stop splitting? (Controlling the complexity of the hypothesis space)

Decision Trees and Overfitting

- What will happen if we keep splitting the data into more and more regions?
  - Every data point will be in its own region—overfitting.
- When should we stop splitting? (Controlling the complexity of the hypothesis space)
  - Limit total number of nodes.

Decision Trees and Overfitting

- What will happen if we keep splitting the data into more and more regions?
  - Every data point will be in its own region—overfitting.
- When should we stop splitting? (Controlling the complexity of the hypothesis space)
  - Limit total number of nodes.
  - Limit number of terminal nodes.

Decision Trees and Overfitting

- What will happen if we keep splitting the data into more and more regions?
  - Every data point will be in its own region—overfitting.
- When should we stop splitting? (Controlling the complexity of the hypothesis space)
  - Limit total number of nodes.
  - Limit number of terminal nodes.
  - Limit tree depth.

Decision Trees and Overfitting

- What will happen if we keep splitting the data into more and more regions?
  - Every data point will be in its own region—overfitting.
- When should we stop splitting? (Controlling the complexity of the hypothesis space)
  - Limit total number of nodes.
  - Limit number of terminal nodes.
  - Limit tree depth.
  - Require minimum number of data points in a terminal node.

Decision Trees and Overfitting

- What will happen if we keep splitting the data into more and more regions?
  - Every data point will be in its own region—overfitting.
- When should we stop splitting? (Controlling the complexity of the hypothesis space)
  - Limit total number of nodes.
  - Limit number of terminal nodes.
  - Limit tree depth.
  - Require minimum number of data points in a terminal node.

Decision Trees and Overfitting

- What will happen if we keep splitting the data into more and more regions?
  - Every data point will be in its own region—overfitting.
- When should we stop splitting? (Controlling the complexity of the hypothesis space)
  - Limit total number of nodes.
  - Limit number of terminal nodes.
  - Limit tree depth.
  - Require minimum number of data points in a terminal node.
  - **Backward pruning** (the approach used in **CART**; Breiman et al 1984):

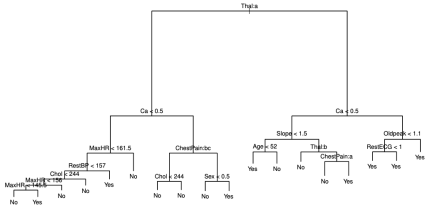
Decision Trees and Overfitting

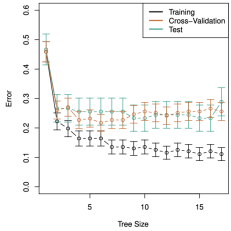
- What will happen if we keep splitting the data into more and more regions?
  - Every data point will be in its own region—overfitting.
- When should we stop splitting? (Controlling the complexity of the hypothesis space)
  - Limit total number of nodes.
  - Limit number of terminal nodes.
  - Limit tree depth.
  - Require minimum number of data points in a terminal node.
  - **Backward pruning** (the approach used in **CART**; Breiman et al 1984):
    1. Build a really big tree (e.g. until all regions have $\leqslant 5$ points).

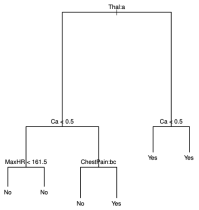
Decision Trees and Overfitting

- What will happen if we keep splitting the data into more and more regions?
  - Every data point will be in its own region—overfitting.
- When should we stop splitting? (Controlling the complexity of the hypothesis space)
  - Limit total number of nodes.
  - Limit number of terminal nodes.
  - Limit tree depth.
  - Require minimum number of data points in a terminal node.
  - **Backward pruning** (the approach used in **CART**; Breiman et al 1984):
    1. Build a really big tree (e.g. until all regions have $\leqslant 5$ points).
    2. *Prune* the tree back greedily, potentially all the way to the root, until validation performance starts decreasing.

Pruning: Example







What Makes a Good Split for Classification?

Our plan is to predict the **majority label** in each region.

Which of the following splits is better?

Split 1 $R_1 : 8+/2-$ $R_2 : 2+/8-$

Split 2 $R_1 : 6+/4-$ $R_2 : 4+/6-$

What Makes a Good Split for Classification?

Our plan is to predict the **majority label** in each region.

Which of the following splits is better?

Split 1 $R_1 : 8+/2-$ $R_2 : 2+/8-$

Split 2 $R_1 : 6+/4-$ $R_2 : 4+/6-$

How about here?

Split 1 $R_1 : 8+/2-$ $R_2 : 2+/8-$

Split 2 $R_1 : 6+/4-$ $R_2 : 0+/10-$

What Makes a Good Split for Classification?

Our plan is to predict the **majority label** in each region.

Which of the following splits is better?

Split 1 $R_1 : 8+/2-$ $R_2 : 2+/8-$

Split 2 $R_1 : 6+/4-$ $R_2 : 4+/6-$

How about here?

Split 1 $R_1 : 8+/2-$ $R_2 : 2+/8-$

Split 2 $R_1 : 6+/4-$ $R_2 : 0+/10-$

Intuition: we want to produce *pure* nodes, i.e. nodes where most instances have the same class.

Misclassification error in a node

- Let's consider the multiclass classification case: $\mathcal{Y} = \{1, 2, \ldots, K\}$.
- Let node m represent region R_m, with N_m observations

Misclassification error in a node

- Let's consider the multiclass classification case: $\mathcal{Y} = \{1, 2, \ldots, K\}$.
- Let node m represent region R_m, with N_m observations
- We denote the proportion of observations in R_m with class k by

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\{i : x_i \in R_m\}} \mathbb{1}[y_i = k].$$

Misclassification error in a node

- Let's consider the multiclass classification case: $\mathcal{Y} = \{1, 2, \ldots, K\}$.
- Let node m represent region R_m, with N_m observations
- We denote the proportion of observations in R_m with class k by

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\{i: x_i \in R_m\}} \mathbb{1}[y_i = k].$$

- We predict the majority class in node m:

$$k(m) = \arg\max_k \hat{p}_{mk}$$

Node Impurity Measures

- Three measures of **node impurity** for leaf node m:

Node Impurity Measures

- Three measures of **node impurity** for leaf node m:
  - Misclassification error

Node Impurity Measures

- Three measures of **node impurity** for leaf node m:
  - Misclassification error
  - The Gini index encourages $\hat{p}_{mk}$ to be close to 0 or 1

Node Impurity Measures

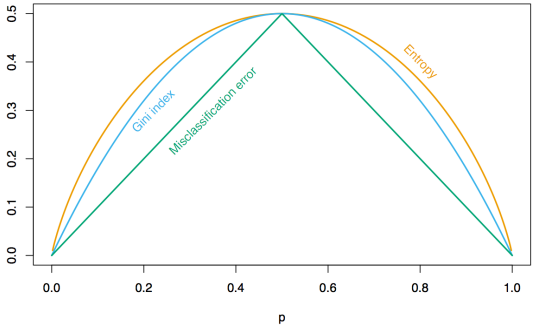
- Three measures of **node impurity** for leaf node m:
  - Misclassification error
  - The Gini index encourages $\hat{p}_{mk}$ to be close to 0 or 1
  - Entropy / Information gain

Node Impurity Measures

- Three measures of **node impurity** for leaf node m:
  - Misclassification error
  - The Gini index encourages $\hat{p}_{mk}$ to be close to 0 or 1
  - Entropy / Information gain
- The Gini index and entropy are numerically similar to each other, and both work better in practice than the misclassification error.

Impurity Measures for Binary Classification

(p is the relative frequency of class 1)



HTF Figure 9.3

Quantifying the Impurity of a Split

Scoring a potential split that produces the nodes R_L and R_R:

- Suppose we have N_L points in R_L and N_R points in R_R.

Quantifying the Impurity of a Split

Scoring a potential split that produces the nodes R_L and R_R:

- Suppose we have N_L points in R_L and N_R points in R_R.
- Let $Q(R_L)$ and $Q(R_R)$ be the node impurity measures for each node.

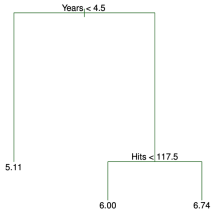
Quantifying the Impurity of a Split

Scoring a potential split that produces the nodes R_L and R_R:

- Suppose we have N_L points in R_L and N_R points in R_R.
- Let $Q(R_L)$ and $Q(R_R)$ be the node impurity measures for each node.
- We aim to find a split that minimizes the *weighted average of node impurities*:

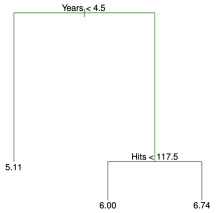
$$\frac{N_L Q(R_L) + N_R Q(R_R)}{N_L + N_R}$$

Discussion: Interpretability of Decision Trees



- Trees are easier to visualize and explain than other classifiers (even linear regression)

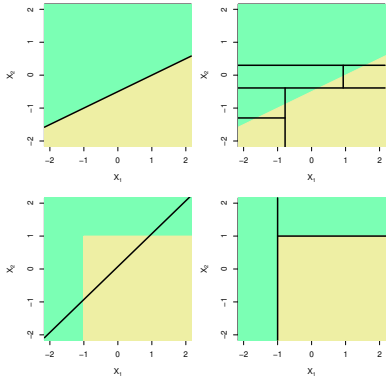
Discussion: Interpretability of Decision Trees



- Trees are easier to visualize and explain than other classifiers (even linear regression)
- Small trees are interpretable – large trees, maybe not so much

Discussion: Trees vs. Linear Models

Trees may have to work hard to capture linear decision boundaries, but can easily capture certain nonlinear ones:



Discussion: Review

Decision trees are:

- Non-linear: the decision boundary that results from splitting may end up being quite complicated
- Non-metric: they do not rely on the geometry of the space (inner products or distances)
- Non-parametric: they make no assumptions about the distribution of the data

Discussion: Review

Decision trees are:

- Non-linear: the decision boundary that results from splitting may end up being quite complicated
- Non-metric: they do not rely on the geometry of the space (inner products or distances)
- Non-parametric: they make no assumptions about the distribution of the data

Additional pros:

- Interpretable and simple to understand

Discussion: Review

Decision trees are:

- Non-linear: the decision boundary that results from splitting may end up being quite complicated
- Non-metric: they do not rely on the geometry of the space (inner products or distances)
- Non-parametric: they make no assumptions about the distribution of the data

Additional pros:

- Interpretable and simple to understand

Cons:

- Struggle to capture linear decision boundaries
- They have high variance and tend to overfit: they are sensitive to small changes in the training data (The ensemble techniques we discuss next can mitigate these issues)