Multiclass Classification, Structured Prediction, & Decision Trees

Mengye Ren

NYU

Nov 7, 2023

Margin for Multiclass

Binary

- Margin for $(x^{(n)}, y^{(n)})$:

$$y^{(n)} w^T x^{(n)} \tag{1}$$

- Want margin to be large and positive ($w^T x^{(n)}$ has same sign as $y^{(n)}$)

Margin for Multiclass

Binary

- Margin for $(x^{(n)}, y^{(n)})$:

$$y^{(n)} w^T x^{(n)} \tag{1}$$

- Want margin to be large and positive ($w^T x^{(n)}$ has same sign as $y^{(n)}$)

Multiclass

- Class-specific margin for $(x^{(n)}, y^{(n)})$:

$$h(x^{(n)}, y^{(n)}) - h(x^{(n)}, y). \tag{2}$$

- Difference between scores of the correct class and each other class
- Want margin to be large and positive for all $y \neq y^{(n)}$.

Multiclass SVM: separable case

Binary

$$\min_{w} \quad \frac{1}{2}\|w\|^2 \tag{3}$$

$$\text{s.t.} \quad \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}} \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D} \tag{4}$$

Multiclass SVM: separable case

Binary

$$\begin{aligned} \min_w \quad & \frac{1}{2}\|w\|^2 && (3) \\ \text{s.t.} \quad & \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}} \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D} && (4) \end{aligned}$$

Multiclass As in the binary case, take 1 as our target margin.

$$m_{n,y}(w) \stackrel{\text{def}}{=} \underbrace{\left\langle w, \Psi(x^{(n)}, y^{(n)}) \right\rangle}_{\text{score of correct class}} - \underbrace{\left\langle w, \Psi(x^{(n)}, y) \right\rangle}_{\text{score of other class}} \tag{5}$$

$$\begin{aligned} \min_w \quad & \frac{1}{2}\|w\|^2 && (6) \\ \text{s.t.} \quad & m_{n,y}(w) \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D}, y \neq y^{(n)} && (7) \end{aligned}$$

Multiclass SVM: separable case

Binary

$$\begin{aligned}
\min_{w} \quad & \frac{1}{2}\|w\|^2 && (3)\\
\text{s.t.} \quad & \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}} \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D} && (4)
\end{aligned}$$

Multiclass As in the binary case, take 1 as our target margin.

$$m_{n,y}(w) \overset{\text{def}}{=} \underbrace{\left\langle w, \Psi(x^{(n)}, y^{(n)}) \right\rangle}_{\text{score of correct class}} - \underbrace{\left\langle w, \Psi(x^{(n)}, y) \right\rangle}_{\text{score of other class}} \tag{5}$$

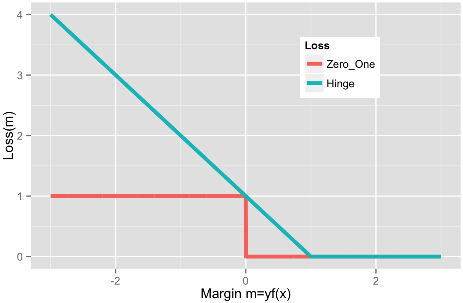
$$\begin{aligned}
\min_{w} \quad & \frac{1}{2}\|w\|^2 && (6)\\
\text{s.t.} \quad & m_{n,y}(w) \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D}, y \neq y^{(n)} && (7)
\end{aligned}$$

Exercise: write the objective for the non-separable case

Recap: hingle loss for binary classification

- Hinge loss: a convex upperbound on the 0-1 loss

$$\ell_{\text{hinge}}(y,\hat{y}) = \max(0, 1 - yh(x)) \tag{8}$$



Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\tag{9}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y \neq y'\} \tag{9}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y \neq y'\} \tag{9}$$

- In general, can also have different cost for each class.

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y \neq y'\} \tag{9}$$

- In general, can also have different cost for each class.

- Upper bound on $\Delta(y, y')$.

$$\hat{y} \overset{\text{def}}{=} \underset{y' \in \mathcal{Y}}{\arg\max} \left\langle w, \Psi(x, y') \right\rangle \tag{10}$$

(12)

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y,y') = \mathbb{I}\{y \neq y'\} \tag{9}$$

- In general, can also have different cost for each class.

- Upper bound on $\Delta(y,y')$.

$$\begin{align} \hat{y} &\overset{\text{def}}{=} \operatorname*{arg\,max}_{y' \in \mathcal{Y}} \langle w, \Psi(x,y') \rangle \tag{10} \\ \implies \langle w, \Psi(x,y) \rangle &\leqslant \langle w, \Psi(x,\hat{y}) \rangle \tag{11} \\ & \tag{12} \end{align}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y \neq y'\} \tag{9}$$

- In general, can also have different cost for each class.

- Upper bound on $\Delta(y, y')$.

$$\hat{y} \overset{\text{def}}{=} \underset{y' \in \mathcal{Y}}{\arg\max} \left\langle w, \Psi(x, y') \right\rangle \tag{10}$$

$$\Longrightarrow \langle w, \Psi(x, y) \rangle \leqslant \langle w, \Psi(x, \hat{y}) \rangle \tag{11}$$

$$\Longrightarrow \Delta(y, \hat{y}) \leqslant \Delta(y, \hat{y}) - \langle w, (\Psi(x, y) - \Psi(x, \hat{y})) \rangle \qquad \text{When are they equal?} \tag{12}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y,y') = \mathbb{I}\{y \neq y'\} \tag{9}$$

- In general, can also have different cost for each class.

- Upper bound on $\Delta(y,y')$.

$$\hat{y} \overset{\text{def}}{=} \underset{y' \in \mathcal{Y}}{\arg\max} \left\langle w, \Psi(x,y') \right\rangle \tag{10}$$

$$\Longrightarrow \langle w, \Psi(x,y)\rangle \leqslant \langle w, \Psi(x,\hat{y})\rangle \tag{11}$$

$$\Longrightarrow \Delta(y,\hat{y}) \leqslant \Delta(y,\hat{y}) - \langle w, (\Psi(x,y) - \Psi(x,\hat{y}))\rangle \qquad \text{When are they equal?} \tag{12}$$

- Generalized hinge loss:

$$\ell_{\text{hinge}}(y,x,w) \overset{\text{def}}{=} \max_{y' \in \mathcal{Y}} \left(\Delta(y,y') - \left\langle w, \left(\Psi(x,y) - \Psi(x,y')\right)\right\rangle\right) \tag{13}$$

Multiclass SVM with Hinge Loss

- Recall the hinge loss formulation for binary SVM (without the bias term):

$$\min_{w \in \mathrm{R}^d} \frac{1}{2}||w||^2 + C \sum_{n=1}^{N} \max\left(0, 1 - \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}}\right).$$

Multiclass SVM with Hinge Loss

- Recall the hinge loss formulation for binary SVM (without the bias term):

$$\min_{w\in\mathbb{R}^d} \frac{1}{2}||w||^2 + C\sum_{n=1}^{N} \max\left(0, 1 - \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}}\right).$$

- The multiclass objective:

$$\min_{w\in\mathbb{R}^d} \frac{1}{2}||w||^2 + C\sum_{n=1}^{N} \max_{y'\in\mathcal{Y}}\left(\Delta(y, y') - \underbrace{\left\langle w, \left(\Psi(x, y) - \Psi(x, y')\right)\right\rangle}_{\text{margin}}\right)$$

  - $\Delta(y, y')$ as target margin for each class.
  - If margin $m_{n,y'}(w)$ meets or exceeds its target $\Delta(y^{(n)}, y')$ $\forall y \in \mathcal{Y}$, then no loss on example n.

Recap: What Have We Got?

- Problem: Multiclass classification $\mathcal{Y} = \{1, \ldots, k\}$

Recap: What Have We Got?

- Problem: Multiclass classification $\mathcal{Y} = \{1, \ldots, k\}$
- Solution 1: One-vs-All
  - Train k models: $h_1(x), \ldots, h_k(x) : \mathcal{X} \to \mathrm{R}$.
  - Predict with $\arg\max_{y \in \mathcal{Y}} h_y(x)$.
  - Gave simple example where this fails for linear classifiers

Recap: What Have We Got?

- Problem: Multiclass classification $\mathcal{Y} = \{1, \ldots, k\}$
- Solution 1: One-vs-All
  - Train k models: $h_1(x), \ldots, h_k(x) : \mathcal{X} \to \mathrm{R}$.
  - Predict with $\arg\max_{y \in \mathcal{Y}} h_y(x)$.
  - Gave simple example where this fails for linear classifiers
- Solution 2: Multiclass loss
  - Train one model: $h(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathrm{R}$.
  - Prediction involves solving $\arg\max_{y \in \mathcal{Y}} h(x, y)$.

Does it work better in practice?

- Paper by Rifkin & Klautau: "In Defense of One-Vs-All Classification" (2004)
  - Extensive experiments, carefully done
    - albeit on relatively small UCI datasets
  - Suggests one-vs-all works just as well in practice
    - (or at least, the advantages claimed by earlier papers for multiclass methods were not compelling)

Does it work better in practice?

- Paper by Rifkin & Klautau: "In Defense of One-Vs-All Classification" (2004)
  - Extensive experiments, carefully done
    - albeit on relatively small UCI datasets
  - Suggests one-vs-all works just as well in practice
    - (or at least, the advantages claimed by earlier papers for multiclass methods were not compelling)
- Compared
  - many multiclass frameworks (including the one we discuss)
  - one-vs-all for SVMs with RBF kernel
  - one-vs-all for square loss with RBF kernel (for classification!)
- All performed roughly the same

Why Are We Bothering with Multiclass?

- The framework we have developed for multiclass
  - compatibility features / scoring functions
  - multiclass margin
  - target margin / multiclass loss

Why Are We Bothering with Multiclass?

- The framework we have developed for multiclass
  - compatibility features / scoring functions
  - multiclass margin
  - target margin / multiclass loss
- Generalizes to situations where k is very large and one-vs-all is intractable.

Why Are We Bothering with Multiclass?

- The framework we have developed for multiclass
  - compatibility features / scoring functions
  - multiclass margin
  - target margin / multiclass loss
- Generalizes to situations where k is very large and one-vs-all is intractable.
- Key idea is that we can generalize across outputs y by using features of y.

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	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

- $\mathcal{V} = \{\text{all English words}\} \cup \{[\text{START}], ''.''\}$
- $\mathcal{X} = \mathcal{V}^n$, $n = 1, 2, 3, \ldots$ [Word sequences of any length]

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	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

- $\mathcal{V} = \{\text{all English words}\} \cup \{\text{[START]}, ''.''\}$
- $\mathcal{X} = \mathcal{V}^n, \ n = 1, 2, 3, \ldots$ [Word sequences of any length]
- $\mathcal{P} = \{\text{START}, \text{Pronoun}, \text{Verb}, \text{Noun}, \text{Adjective}\}$
- $\mathcal{Y} = \mathcal{P}^n, \ n = 1, 2, 3, \ldots$[Part of speech sequence of any length]

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: $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \to \mathrm{R}\}$
  - $h(x, y)$ 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: $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \to \mathrm{R}\}$
  - $h(x, y)$ gives compatibility score between input x and output y
- Multiclass hypothesis space

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

- 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:

$$h(x,y) = w^T \Psi(x,y) \tag{14}$$

$$\mathcal{F} = \left\{ x \mapsto \operatorname*{arg\,max}_{y \in \mathcal{Y}} h(x,y) \mid h \in \mathcal{H} \right\} \tag{15}$$

- 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):

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

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):

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

- 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:

$$\sum_i \langle w, \Psi_i(x, y_{i-1}, y_i)\rangle = \left\langle w, \sum_i \Psi_i(x, y_{i-1}, y_i)\right\rangle = \langle w, \Psi(x, y)\rangle, \tag{16}$$

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

$$\ell_{\text{hinge}}(y,\hat{y}) \overset{\text{def}}{=} \max_{y' \in \mathcal{Y}(x)} \left(\Delta(y,y') + \left\langle w, \left(\Psi(x,y') - \Psi(x,y)\right)\right\rangle\right) \tag{17}$$

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

Structured hinge loss

- Recall the generalized hinge loss

$$\ell_{\text{hinge}}(y, \hat{y}) \overset{\text{def}}{=} \max_{y' \in \mathcal{Y}(x)} \left(\Delta(y, y') + \left\langle w, \left(\Psi(x, y') - \Psi(x, y)\right)\right\rangle\right) \tag{17}$$

- 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
- 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 (similar to the Viterbi algorithm)

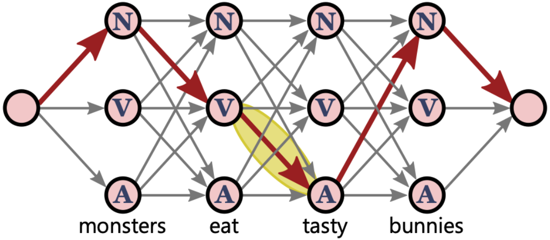
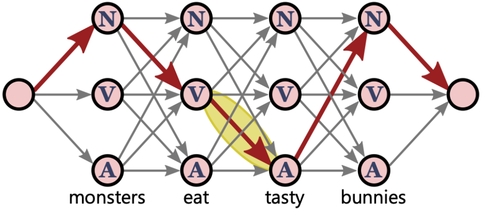


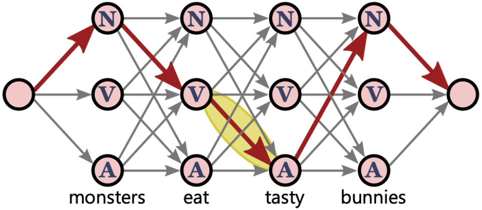
Figure by Daumé III. A course in machine learning. Figure 17.1.

Structured SVM inference (linear chain)



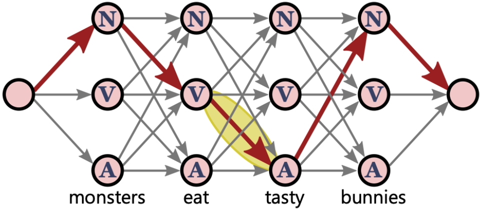
- Initiate $\alpha_j(1) = w^\top \psi(y_1 = j, x_1)$

Structured SVM inference (linear chain)



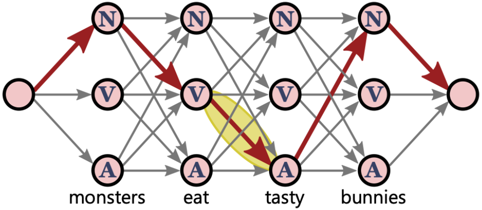
- Initiate $\alpha_j(1) = w^\top \psi(y_1 = j, x_1)$
- Recursion $\alpha_j(t) = \max_i \alpha_i(t-1) + w^\top \psi(y_t = j, y_{t-1} = i, x_t)$

Structured SVM inference (linear chain)



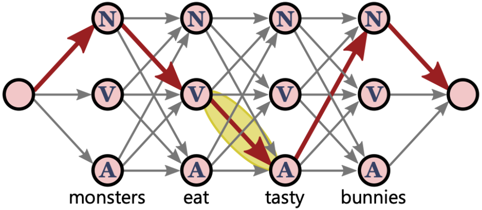
- Initiate $\alpha_j(1) = w^\top \psi(y_1 = j, x_1)$
- Recursion $\alpha_j(t) = \max_i \alpha_i(t-1) + w^\top \psi(y_t = j, y_{t-1} = i, x_t)$
- Pointer $\gamma(t,j) = \arg\max_i \alpha_i(t-1) + w^\top \psi(y_t = j, y_{t-1} = i, x_t)$

Structured SVM inference (linear chain)



- Initiate $\alpha_j(1) = w^\top \psi(y_1 = j, x_1)$
- Recursion $\alpha_j(t) = \max_i \alpha_i(t-1) + w^\top \psi(y_t = j, y_{t-1} = i, x_t)$
- Pointer $\gamma(t,j) = \arg\max_i \alpha_i(t-1) + w^\top \psi(y_t = j, y_{t-1} = i, x_t)$
- Backtrack: $r(T) = \arg\max_i \alpha_i(T), r(t) = \gamma(t, r(t+1))$

Structured SVM inference (linear chain)



- Initiate $\alpha_j(1) = w^\top \psi(y_1 = j, x_1)$
- Recursion $\alpha_j(t) = \max_i \alpha_i(t-1) + w^\top \psi(y_t = j, y_{t-1} = i, x_t)$
- Pointer $\gamma(t,j) = \arg\max_i \alpha_i(t-1) + w^\top \psi(y_t = j, y_{t-1} = i, x_t)$
- Backtrack: $r(T) = \arg\max_i \alpha_i(T), r(t) = \gamma(t, r(t+1))$

What's the running time?

The argmax problem in general

Efficient problem-specific algorithms:

problem	structure	algorithm
constituent parsing	binary trees with context-free features	CYK
dependency parsing	spanning trees with edge features	Chu-Liu-Edmonds
image segmentation	2d with adjacent-pixel features	graph cuts

The argmax problem in general

Efficient problem-specific algorithms:

problem	structure	algorithm
constituent parsing	binary trees with context-free features	CYK
dependency parsing	spanning trees with edge features	Chu-Liu-Edmonds
image segmentation	2d with adjacent-pixel features	graph cuts

General algorithm:

- Integer linear programming (ILP)

$$\max_z a^T z \quad \text{s.t. linear constraints on } z \tag{18}$$

- z: indicator of substructures, e.g., $\mathbb{I}\{y_i = \text{article and } y_{i+1} = \text{noun}\}$
- constraints: z must correspond to a valid structure

Conditional random field (CRF)

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

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

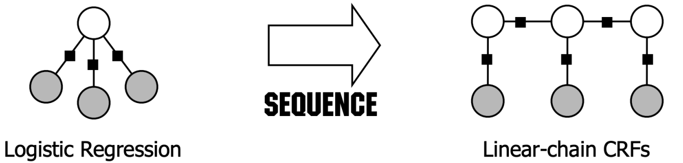
- Z is normalization constant: $Z(x) = \sum_{y \in Y} \exp(w^\top \psi(x,y))$.

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: $p(y|x) = \frac{1}{Z(x)} \exp(\sum_t w^\top \psi(x, y_t, y_{t-1}))$



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: $\lambda\|w\|^2$.
- $p(y|x) = \frac{1}{Z(x)}\exp(w^\top \psi(x,y))$.
- 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:

$$\frac{\partial l(w)}{\partial w_k} = -\frac{1}{N}\sum_i \sum_t \sum_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)}) \tag{19}$$

$$+\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1})) + \sum_k \lambda w_k \tag{20}$$

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

$$\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1})) \tag{21}$$

$$= \frac{1}{N}\sum_i \left[\sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1}))\right]^{-1} \tag{22}$$

$$\left[\sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)})) \sum_t \psi_k(x^{(i)}, y'_t, y'_{t-1})\right] \tag{23}$$

$$= \frac{1}{N}\sum_i \sum_t \sum_{y' \in Y} p(y'_t, y'_{t-1}|x) \psi_k(x^{(i)}, y'_t, y'_{t-1}) \tag{24}$$

Conditional random field (CRF)

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

$$\frac{1}{N}\sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1})) \tag{21}$$

$$= \frac{1}{N}\sum_i \left[\sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1}))\right]^{-1} \tag{22}$$

$$\left[\sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y^{(i)}_t, y^{(i)}_{t-1})) \sum_t \psi_k(x^{(i)}, y'_t, y'_{t-1})\right] \tag{23}$$

$$= \frac{1}{N}\sum_i \sum_t \sum_{y' \in Y} p(y'_t, y'_{t-1}|x) \psi_k(x^{(i)}, y'_t, y'_{t-1}) \tag{24}$$

- 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 $p(y|x)$.

Conditional random field (CRF)

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

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.

- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$
- Result: $Z(x) = \sum_j \alpha_j(T)$

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.

- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$

- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$

- Result: $Z(x) = \sum_j \alpha_j(T)$

- Similar for the backward direction.

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$
- Result: $Z(x) = \sum_j \alpha_j(T)$
- Similar for the backward direction.
- Test time, again use Viterbi algorithm to infer argmax.

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$
- Result: $Z(x) = \sum_j \alpha_j(T)$
- Similar for the backward direction.
- Test time, again use Viterbi algorithm to infer argmax.
- The inference algorithm can be generalized to belief propagation (BP) in a tree structure (exact inference).

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$
- Result: $Z(x) = \sum_j \alpha_j(T)$
- Similar for the backward direction.
- Test time, again use Viterbi algorithm to infer argmax.
- 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.

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, and a sky pixel is likely to be above a grass pixel.

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, and a sky pixel is likely to be above a grass pixel.
- Multi-label learning
  An image may contain multiple class labels, e.g. a bus is likely to co-occur with a car.

Conclusion

Multiclass algorithms

- Reduce to binary classification, e.g., OvA, AvA
    - Good enough for simple multiclass problems
    - They don't scale and have simplified assumptions

Conclusion

Multiclass algorithms

- Reduce to binary classification, e.g., OvA, AvA
    - Good enough for simple multiclass problems
    - They don't scale and have simplified assumptions
- Generalize binary classification algorithms using multiclass loss
    - Multi-class perceptron, multi-class logistics regression, multi-class SVM

Conclusion

Multiclass algorithms

- Reduce to binary classification, e.g., OvA, AvA
    - Good enough for simple multiclass problems
    - They don't scale and have simplified assumptions
- Generalize binary classification algorithms using multiclass loss
    - Multi-class perceptron, multi-class logistics regression, multi-class SVM
- Structured prediction: Structured SVM, CRF. Data containing structure. Extremely large output space. Text and image applications. More in-depth content in a probabilistic graphical model (PGM) course.

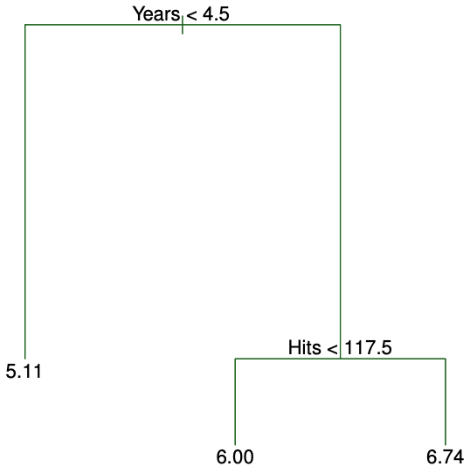
Decision Trees

Overview: Decision Trees

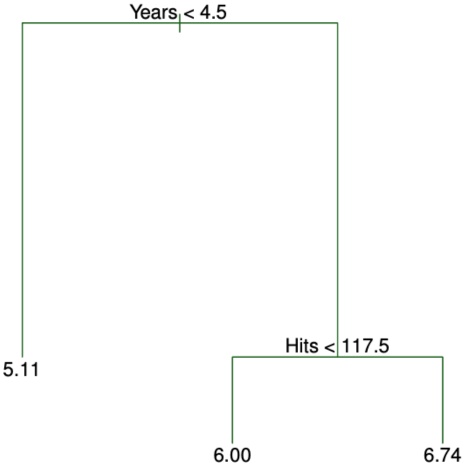
- Our first inherently non-linear classifier: decision trees.
- Ensemble methods: bagging and boosting.

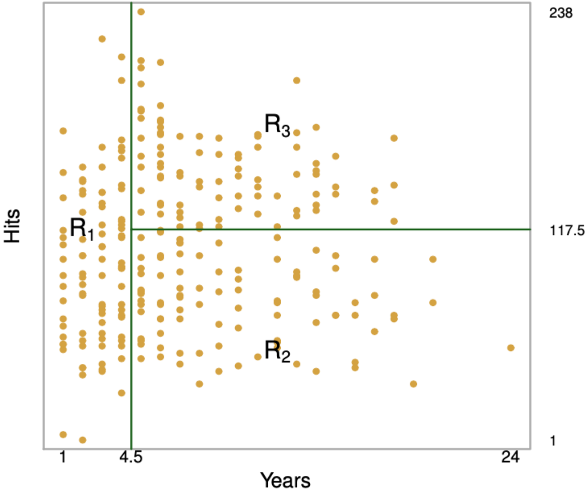
Decision Trees

Regression trees: Predicting basketball players' salaries

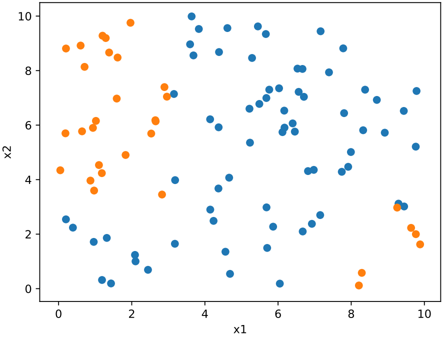


Regression trees: Predicting basketball players' salaries



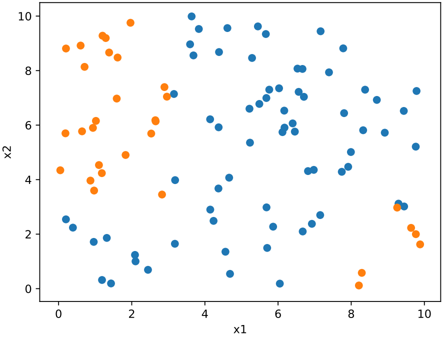


Classification trees



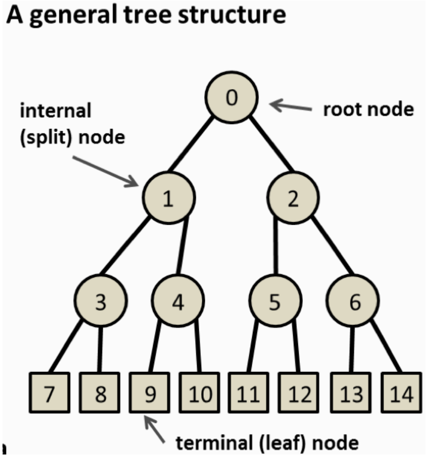
- Can we classify these points using a linear classifier?

Classification trees



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

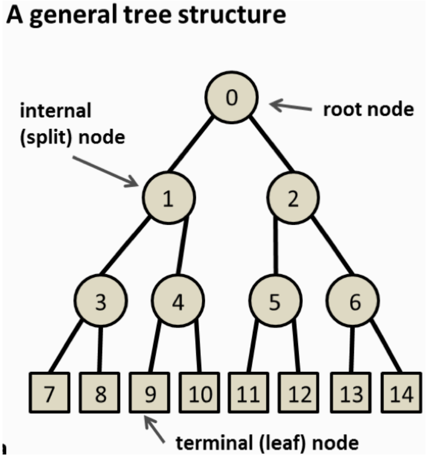
Decision trees setup



- 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.

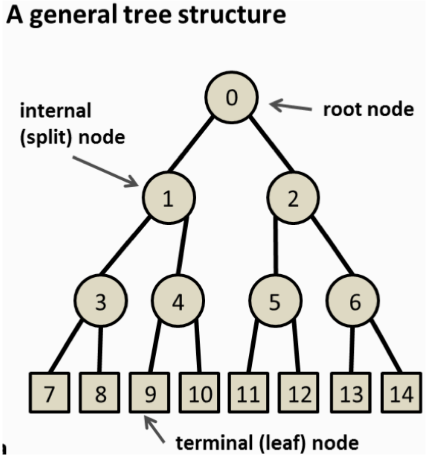
Decision trees setup



- 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.

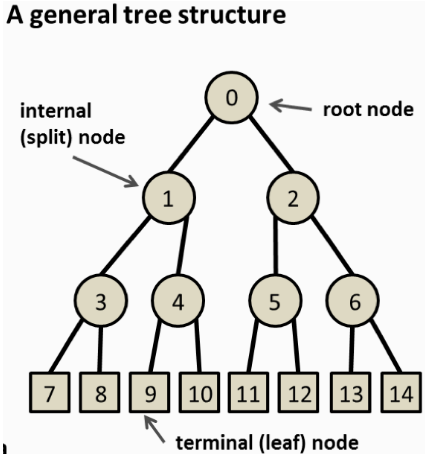
Decision trees setup



- 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.

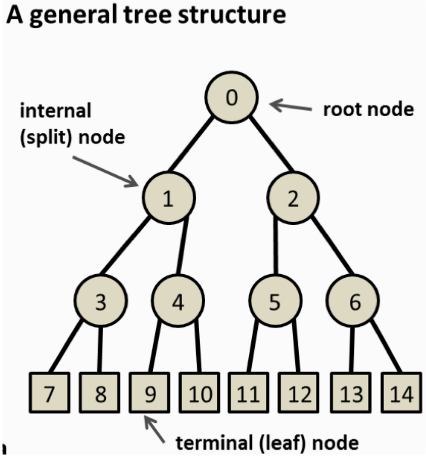
Decision trees setup



- 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.

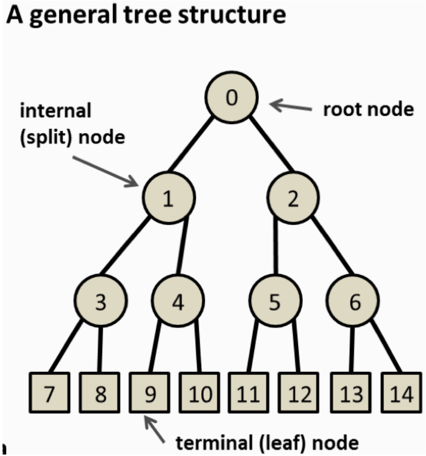
Decision trees setup



- 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



- 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

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

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.

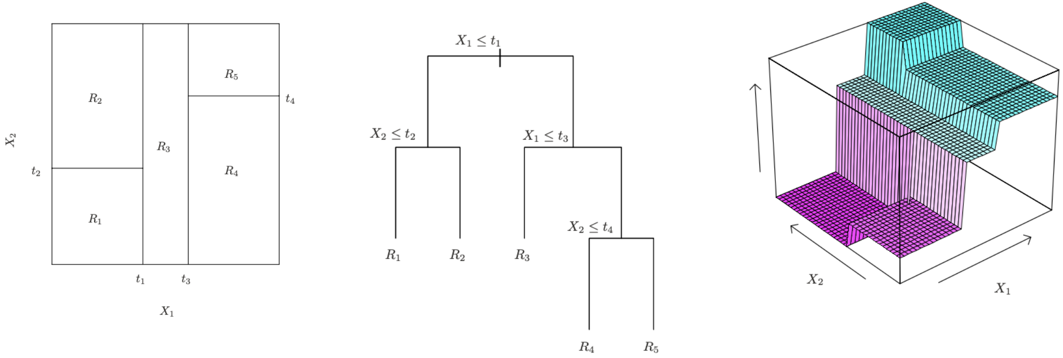
Solution Greedy algorithm: starting from the root, and repeating until a stopping criterion is reached (e.g., max depth), find the non-terminal node that results in the "best" split

- 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:

$$s_j \in \left\{ \frac{1}{2}\left(x_{j(r)} + x_{j(r+1)}\right) \mid r = 1, \ldots, n-1 \right\}. \qquad n-1 \text{ splits} \qquad (25)$$

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):

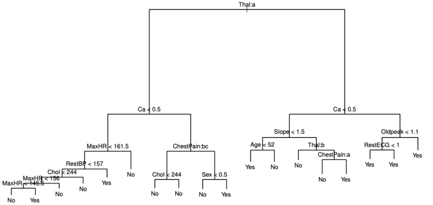
Decision Trees and Overfitting

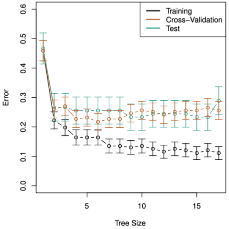
- What will happen if we keep splitting the data into more and more regions?
  - Every data point will be in its own region—overfitting.
- When should we stop splitting? (Controlling the complexity of the hypothesis space)
  - Limit total number of nodes.
  - Limit number of terminal nodes.
  - Limit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

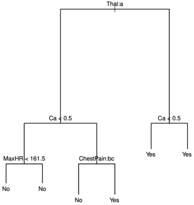
Decision Trees and Overfitting

- 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

$$1 - \hat{p}_{mk(m)}.$$

Node Impurity Measures

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

$$1 - \hat{p}_{mk(m)}.$$

  - The Gini index encourages $\hat{p}_{mk}$ to be close to 0 or 1

$$\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

Node Impurity Measures

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

$$1 - \hat{p}_{mk(m)}.$$

  - The Gini index encourages $\hat{p}_{mk}$ to be close to 0 or 1

$$\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

  - Entropy / Information gain

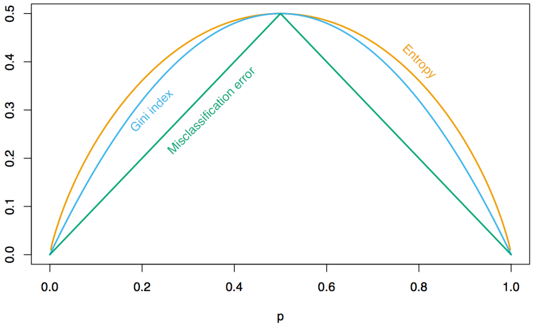
$$-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

Node Impurity Measures

- Three measures of **node impurity** for leaf node m:
  - Misclassification error
    $$1 - \hat{p}_{mk(m)}.$$
  - The Gini index encourages $\hat{p}_{mk}$ to be close to 0 or 1
    $$\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$
  - Entropy / Information gain
    $$-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$
- 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.

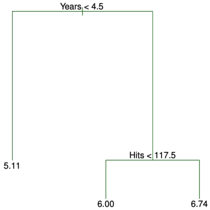
Quantifying the Impurity of a Split

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*:

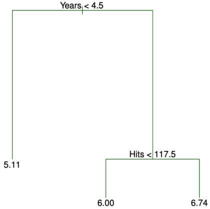
$$\frac{N_L Q(R_L) + N_R Q(R_R)}{N_L + N_R}$$

Discussion: Interpretability of Decision Trees



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

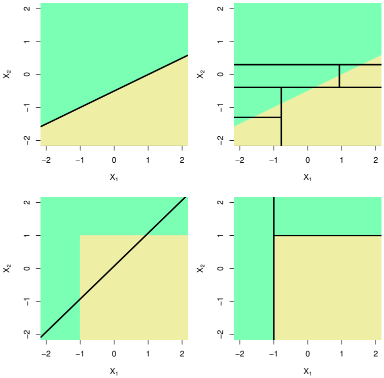
Discussion: Interpretability of Decision Trees



- 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)