Machine Learning: Chenhao Tan

University of Colorado Boulder

LECTURE 3

Logistics

- Zoom recording: https://canvas.colorado.edu/courses/63981/pages/zoom-recordings
- HW1 available on Github, due on Sep 11
- Start early! It can take a while and do not wait until the night before

Learning objectives

- Training decision tree and introduce the concept of interpretability
- Understand information gain and different splitting criteria
- Understand the formal definition of supervised learning

Outline

Decision tree review & Interpretability

Information gain as splitting criteria

Formal definition of supervised learning

Outline

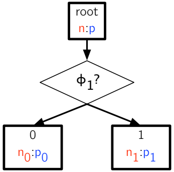
Decision tree review & Interpretability

Information gain as splitting criteria

Formal definition of supervised learning

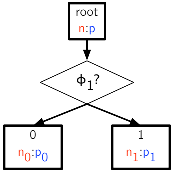
Growing a Decision Tree

Growing a Decision Tree



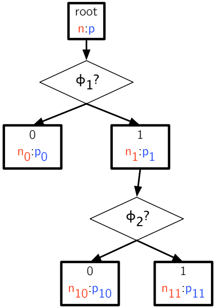
We chose feature ϕ_1. Note that $n = n_0 + n_1$ and $p = p_0 + p_1$.

Growing a Decision Tree

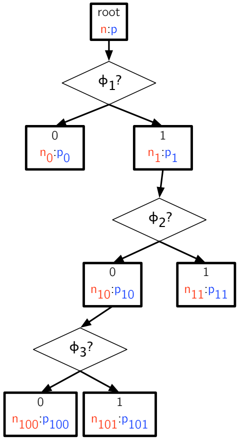


We chose not to split the left partition. Why not?

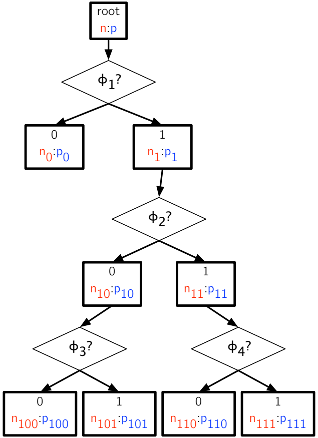
Growing a Decision Tree



Growing a Decision Tree



Growing a Decision Tree



Greedily Building a Decision Tree (Binary Features)

Algorithm: DTreeTrain

Data: data D, feature set Φ

Result: decision tree

if *all examples in D have the same label y, or Φ is empty and y is the best guess* **then**

    return Leaf(y);

else

    for *each feature ϕ in Φ* **do**

        partition D into D_0 and D_1 based on ϕ-values;

        let mistakes(ϕ) = (non-majority answers in D_0) + (non-majority answers in D_1);

    end

    let ϕ^* be the feature with the smallest number of mistakes;

    return Node(ϕ^*, {0 $\rightarrow$ DTreeTrain($D_0, \Phi \setminus \{\phi^*\}$), 1 $\rightarrow$ DTreeTrain($D_1, \Phi \setminus \{\phi^*\}$)});

end

Greedily Building a Decision Tree (Binary Features)

Algorithm: DTREETRAIN

Data: data D, feature set Φ
Result: decision tree
if *all examples in D have the same label y, or Φ is empty and y is the best guess*
 then
  return LEAF(y);
else
  for *each feature ϕ in Φ* **do**
    partition D into D_0 and D_1 based on ϕ-values;
    let mistakes(ϕ) = (non-majority answers in D_0) + (non-majority answers in D_1);
  end
  let ϕ^* be the feature with the smallest number of mistakes;
  return NODE(ϕ^*, {0 $\rightarrow$ DTREETRAIN($D_0, \Phi \setminus \{\phi^*\}$), 1 $\rightarrow$ DTREETRAIN($D_1, \Phi \setminus \{\phi^*\}$)});
end

Does this algorithm always terminate? Why?

Greedily Building a Decision Tree (Binary Features)

Algorithm: DTreeTrain

Data: data D, feature set Φ
Result: decision tree
if *all examples in D have the same label y, or Φ is empty and y is the best guess*
 then
  return Leaf(y);
else
  for *each feature ϕ in Φ* **do**
    partition D into D_0 and D_1 based on ϕ-values;
    let mistakes(ϕ) = (non-majority answers in D_0) + (non-majority answers in D_1);
  end
  let ϕ^* be the feature with the smallest number of mistakes;
  return Node(ϕ^*, {0 $\rightarrow$ DTreeTrain($D_0, \Phi \setminus \{\phi^*\}$), 1 $\rightarrow$ DTreeTrain($D_1, \Phi \setminus \{\phi^*\}$)});
end

Φ is finite and every call will either reach a leaf node or reduce the size of feature set by 1.

Greedily Building a Decision Tree (Binary Features)

Algorithm: DTreeTrain

Data: data D, feature set Φ
Result: decision tree
if *all examples in D have the same label y, or Φ is empty and y is the best guess* **then**
    return Leaf(y);
else
    for *each feature ϕ in Φ* **do**
        partition D into D_0 and D_1 based on ϕ-values;
        let mistakes(ϕ) = (non-majority answers in D_0) + (non-majority answers in D_1);
    end
    let ϕ^* be the feature with the smallest number of mistakes;
    return Node(ϕ^*, {$0 \rightarrow$ DTreeTrain($D_0, \Phi \setminus \{\phi^*\}$), $1 \rightarrow$ DTreeTrain($D_1, \Phi \setminus \{\phi^*\}$)});
end

Is the algorithm guaranteed to find optimal decision tree?

Greedily Building a Decision Tree (Binary Features)

What is the optimal training error?

Greedily Building a Decision Tree (Binary Features)

What is the optimal training error?

One can always get the optimal training error by splitting based on all features.

Greedily Building a Decision Tree (Binary Features)

What is the optimal training error?
One can always get the optimal training error by splitting based on all features.

Optimal decision tree refers to the smallest tree that minimizes the training error.

Greedily Building a Decision Tree (Binary Features)

What is the optimal training error?
One can always get the optimal training error by splitting based on all features.

Optimal decision tree refers to the smallest tree that minimizes the training error.

A	B	C	$-$	$+$
0	0	0	10	5
0	0	1	5	20
0	1	0	5	20
0	1	1	10	5
1	0	0	30	5
1	0	1	5	10
1	1	0	5	10
1	1	1	30	5

Greedily Building a Decision Tree (Binary Features)

What is the optimal training error?
One can always get the optimal training error by splitting based on all features.

Optimal decision tree refers to the smallest tree that minimizes the training error.

A	B	C	$-$	$+$
0	0	0	10	5
0	0	1	5	20
0	1	0	5	20
0	1	1	10	5
1	0	0	30	5
1	0	1	5	10
1	1	0	5	10
1	1	1	30	5

The greedy algorithm splits by A first, but the optimal tree only needs B and C.

Interpretability of machine learning

Example definitions:

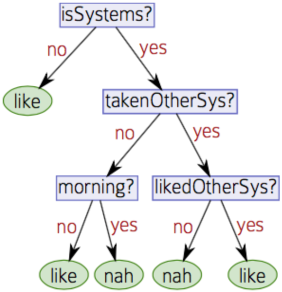
- Can you understand how a prediction is made?
- Can you explain how the model works as a whole?
- Can you make better decisions with assistance of the model?

Interpretability of decision trees

Decision trees are generally considered interpretable.

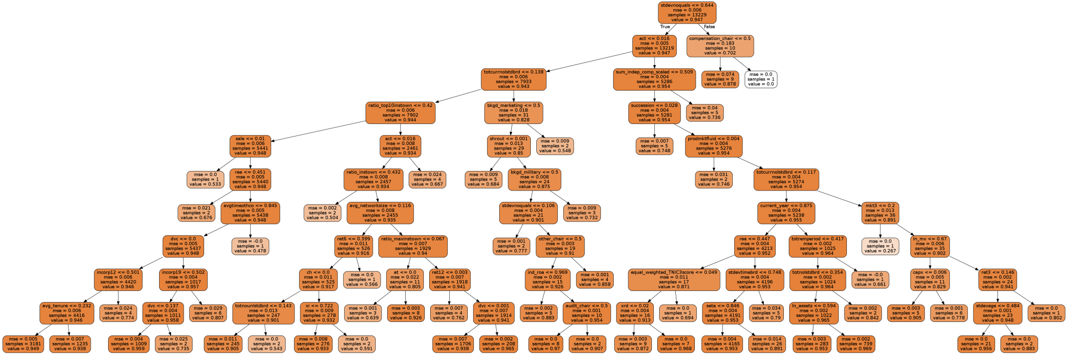
Interpretability of decision trees

Decision trees are generally considered interpretable.



Interpretability of decision trees

Decision trees are generally considered interpretable.
But what about this?



Outline

Decision tree review & Interpretability

Information gain as splitting criteria

Formal definition of supervised learning

Information gain as splitting criteria

- Inspired by information theory
- Entropy: measure of impurity of set of examples

Entropy

$$H(X) = -\sum_c p_c \log_2(p_c),$$

where p_c is the fraction of examples in class c.

Entropy

$$H(X) = -\sum_c p_c \log_2(p_c),$$

where p_c is the fraction of examples in class c. Note that for binary classification, let p be the fraction in the positive class, then

$$H(X) = -p \log_2 p - (1-p) \log_2(1-p)$$

Entropy

$$H(X) = -\sum_{c} p_c \log_2(p_c),$$

where p_c is the fraction of examples in class c. Note that for binary classification, let p be the fraction in the positive class, then

$$H(X) = -p \log_2 p - (1-p) \log_2(1-p)$$

What is the largest/smallest entropy?

Entropy

$$H(X) = -p \log_2 p - (1-p) \log_2 (1-p)$$

What is the largest/smallest entropy?

$$0 \leq p \leq 1$$

Entropy

$$H(X) = -p\log_2 p - (1-p)\log_2(1-p)$$

What is the largest/smallest entropy?

$$0 \leq p \leq 1$$

- When all examples are in the same class, entropy is 0
- When samples are equally balanced, entropy is 1

Splitting

Consider the tennis problem now with binary features

sun	wind	humidity	tennis
sunny	windy	not humid	tennis
sunny	not windy	not humid	tennis
not sunny	not windy	humid	no tennis
sunny	windy	humid	no tennis

Splitting

Converting to features and labels

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

Splitting

Converting to features and labels

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is the entropy of the root node?

Splitting

Converting to features and labels

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is the entropy of the root node?

Easy: the root node is balanced, so the entropy is 1

Splitting

Converting to features and labels

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is the entropy of the root node?

Easy: the root node is balanced, so the entropy is 1

$$p = \frac{1}{2} \Rightarrow H(X) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

Information gain

The higher entropy is, the lower the information is.
Information gain is defined as the difference between impurity at the parent and (weighted average) of impurity at the children

Information gain

The higher entropy is, the lower the information is.
Information gain is defined as the difference between impurity at the parent and (weighted average) of impurity at the children
Splitting based on feature i

- X_{parent}: training subset of the parent node
- $X_{i,\text{left}}$: training subset of the left node
- $X_{i,\text{right}}$: training subset of the right node

Information gain

The higher entropy is, the lower the information is.
Information gain is defined as the difference between impurity at the parent and (weighted average) of impurity at the children
Splitting based on feature i

- X_{parent}: training subset of the parent node
- $X_{i,\text{left}}$: training subset of the left node
- $X_{i,\text{right}}$: training subset of the right node

$$IG(X_{\text{parent}}, i) = H(X_{\text{parent}}) - \frac{|X_{i,\text{left}}|}{|X_{\text{parent}}|} H(X_{\text{left}}) - \frac{|X_{i,\text{right}}|}{|X_{\text{parent}}|} H(X_{\text{right}})$$

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{sun})$?

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{sun})$?

- $X_{\text{parent}} : \{1, 1, 0, 0\}$
- $X_{\text{left},\textsf{sun}} : \{0\}$
- $X_{\text{right},\textsf{sun}} : \{1, 1, 0\}$

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{sun})$?

- $X_{\text{parent}} : \{1, 1, 0, 0\}$
- $X_{\text{left,sun}} : \{0\}$
- $X_{\text{right,sun}} : \{1, 1, 0\}$

- $H(X_{\text{parent}}) = 1$
- $H(X_{\text{left,sun}}) = 0$
- $H(X_{\text{right,sun}}) = 0.918$

$$IG(X, \text{sun}) = 1 - \frac{1}{4} * 0 - \frac{3}{4} * 0.918 = 0.3112$$

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{wind})$?

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{wind})$?

- $X_{\text{parent}} : \{1, 1, 0, 0\}$
- $X_{\text{left},\text{wind}} : \{1, 0\}$
- $X_{\text{right},\text{wind}} : \{1, 0\}$

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{wind})$?

- $X_{\text{parent}} : \{1, 1, 0, 0\}$
- $X_{\text{left,wind}} : \{1, 0\}$
- $X_{\text{right,wind}} : \{1, 0\}$

- $H(X_{\text{parent}}) = 1$
- $H(X_{\text{left,wind}}) = 1$
- $H(X_{\text{right,wind}}) = 1$

$$IG(X, \text{wind}) = 1 - \frac{1}{2} * 1 - \frac{1}{2} * 1 = 0$$

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{humid})$?

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{humid})$?

- $X_{\text{parent}} : \{1, 1, 0, 0\}$
- $X_{\text{left},\textsf{humid}} : \{1, 1\}$
- $X_{\text{right},\textsf{humid}} : \{0, 0\}$

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{humid})$?

- $X_{\text{parent}} : \{1, 1, 0, 0\}$
- $X_{\text{left,humid}} : \{1, 1\}$
- $X_{\text{right,humid}} : \{0, 0\}$

- $H(X_{\text{parent}}) = 1$
- $H(X_{\text{left,humid}}) = 0$
- $H(X_{\text{right,humid}}) = 0$

$$IG(X, \text{humid}) = 1 - \frac{1}{2} * 0 - \frac{1}{2} * 0 = 1$$

Splitting

- $IG(X, \text{sun}) = 0.3112$
- $IG(X, \text{wind}) = 0$
- $IG(X, \text{humid}) = 1$

Which feature should we split on?

Splitting

- $IG(X, \text{sun}) = 0.3112$
- $IG(X, \text{wind}) = 0$
- $IG(X, \text{humid}) = 1$

Which feature should we split on?
humid, since it brings the greatest information gain.

Different splitting criteria

$$IG(X_{\text{parent}}, i) = I(X_{\text{parent}}) - \frac{|X_{i,\text{left}}|}{|X_{\text{parent}}|} I(X_{\text{left}}) - \frac{|X_{i,\text{right}}|}{|X_{\text{parent}}|} I(X_{\text{right}})$$

- Entropy
- Misclassification error ($I = \min_c p_c$)
- Gini index ($I = 1 - \sum_c p_c^2$)

Different splitting criteria

$$IG(X_{\text{parent}}, i) = I(X_{\text{parent}}) - \frac{|X_{i,\text{left}}|}{|X_{\text{parent}}|} I(X_{\text{left}}) - \frac{|X_{i,\text{right}}|}{|X_{\text{parent}}|} I(X_{\text{right}})$$

- Entropy
- Misclassification error ($I = \min_c p_c$)
- Gini index ($I = 1 - \sum_c p_c^2$)

Entropy and Gini index are more often used. For more information, read pointers on Piazza.

Outline

Supervised Learning

Hutzler #571 Banana Slicer

The only banana slicer you will ever need.

Gourmac's easy-to-use Banana Slicer provides a quick solution to slice a banana uniformly each and every time. Simply press the slicer on a peeled banana and the work is done. Safe, fun and easy for children to use. Kids just love eating bananas with this as their favorite kitchen tool. The Banana Slicer may also be used as a quick way to add healthy bananas to breakfast cereal or to make uniform slices for a fruit salad or ice cream dessert.

Data

X

Labels

Y

- **Supervised methods** find patterns in **fully observed** data and then try to predict something from **partially observed** data.

Formal Definitions

- Labels Y, e.g., binary labels $y \in \{+1, -1\}$
- Instance space X, all the possible instances (based on data representation)
- Target function f: $X \rightarrow Y$ (f is unknown)

Formal Definitions

- Labels Y, e.g., binary labels $y \in \{+1, -1\}$
- Instance space X, all the possible instances (based on data representation)
- Target function f: $X \rightarrow Y$ (f is unknown)
- Example/instance $(\boldsymbol{x}, y)$
- Training data S_{train}: collection of examples observed by the algorithm

Formal Definitions

- Goal of a learning algorithm:

  Find a function $h : X \to Y$ from training data S_{train} so that h approximates f

Supervised learning in a nutshell

$$S_{\text{train}} = \{(\boldsymbol{x}, y)\} \rightarrow h$$

Supervised learning in a nutshell

$$S_{\text{train}} = \{(\boldsymbol{x}, y)\} \rightarrow h$$

Given loss function l and $(\boldsymbol{x}, y) \sim D$, expected loss is defined as:

$$\epsilon \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim D}\left[l(h(\boldsymbol{x}), y)\right] = \sum_{\boldsymbol{x}\in X, y\in Y} D(X = \boldsymbol{x}, Y = y) l(h(\boldsymbol{x}), y),$$

Supervised learning in a nutshell

$$S_{\text{train}} = \{(\boldsymbol{x}, y)\} \rightarrow h$$

Given loss function l and $(\boldsymbol{x}, y) \sim D$, expected loss is defined as:

$$\epsilon \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim D}\left[l(h(\boldsymbol{x}), y)\right] = \sum_{\boldsymbol{x}\in X, y\in Y} D(X = \boldsymbol{x}, Y = y) l(h(\boldsymbol{x}), y),$$

but we only have access to training error for a training sample $S_{\text{train}} = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, N\}$:

$$\hat{\epsilon} \triangleq \frac{1}{N} \sum_{1}^{N} l(h(\boldsymbol{x}_i), y_i).$$

Supervised learning in a nutshell

$$S_{\text{train}} = \{(\boldsymbol{x}, y)\} \rightarrow h$$

Given loss function l and $(\boldsymbol{x}, y) \sim D$, expected loss is defined as:

$$\epsilon \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim D}\left[l(h(\boldsymbol{x}), y)\right] = \sum_{\boldsymbol{x}\in X, y\in Y} D(X = \boldsymbol{x}, Y = y) l(h(\boldsymbol{x}), y),$$

but we only have access to training error for a training sample $S_{\text{train}} = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, N\}$:

$$\hat{\epsilon} \triangleq \frac{1}{N} \sum_{1}^{N} l(h(\boldsymbol{x}_i), y_i).$$

Minimizing training error is not ideal.

No Free Lunch Theorems

- No free lunch for supervised machine learning [Wolpert, 1996]:
  in a noise-free scenario where the loss function is the misclassification rate, if one is interested in off-training-set error, then there are no a priori distinctions between learning algorithms.

No Free Lunch Theorems

- No free lunch for supervised machine learning [Wolpert, 1996]:
  in a noise-free scenario where the loss function is the misclassification rate, if one is interested in off-training-set error, then there are no a priori distinctions between learning algorithms.
  Corollary I: there is no single ML algorithm that works for everything.
  Corollary II: every successful ML algorithm makes assumptions.

No Free Lunch Theorems

- No free lunch for supervised machine learning [Wolpert, 1996]:
  in a noise-free scenario where the loss function is the misclassification rate, if one is interested in off-training-set error, then there are no a priori distinctions between learning algorithms.
  Corollary I: there is no single ML algorithm that works for everything.
  Corollary II: every successful ML algorithm makes assumptions.
- No free lunch for search/optimization [Wolpert and Macready, 1997]: All algorithms that search for an extremum of a cost function perform exactly the same when averaged over all possible cost functions.

References

David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7): 1341–1390, 1996.

David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.