Machine Learning: Yoshinari Fujinuma

University of Colorado Boulder

LECTURE 3

Slides adapted from Chris Ketelsen and Chenhao Tan

Logistics

- HW1 will be available on Github around 5pm today, due on Feb. 5th 11:59pm MT
- Start early (though all materials are not covered yet)

Learning objectives

- Dive deep into decision trees
- Formalize the definition of supervised learning (and prepare for discussing bias-variance trade-off)

Outline

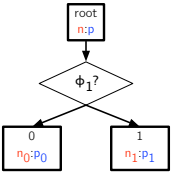
Finishing up Decision Trees

Formal definition of supervised learning

Outline

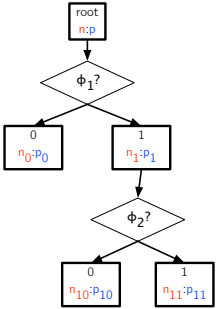
Growing a Decision Tree

Growing a Decision Tree



We chose feature ϕ_1.

Growing a Decision Tree



We chose feature ϕ_2

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* **then**
  return LEAF(y);
else
  for *each feature* ϕ *in* Φ **do**
    partition D into D_0 and D_1 based on ϕ;
    calculate $IG(D, \phi)$
  end
  if $IG(D, \phi) = 0$ *for all* ϕ **then**
    terminate
  else
    Choose ϕ^* be the feature with the largest $IG(D, \phi)$;
    Create NODE(ϕ^*);
    DTREETRAIN($D_0, \Phi \setminus \{\phi^*\}$);
    DTREETRAIN($D_1, \Phi \setminus \{\phi^*\}$);
  end
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* **then**
  return LEAF(y);
else
  for *each feature ϕ in Φ* **do**
    partition D into D_0 and D_1 based on ϕ;
    calculate $IG(D, \phi)$
  end
  if $IG(D, \phi) = 0$ *for all* ϕ **then**
    terminate
  else
    Choose ϕ^* be the feature with the largest $IG(D, \phi)$;
    Create NODE(ϕ^*);
    DTREETRAIN($D_0, \Phi \setminus \{\phi^*\}$);
    DTREETRAIN($D_1, \Phi \setminus \{\phi^*\}$);
  end
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* **then**
  return LEAF(y);
else
  for *each feature ϕ in Φ* **do**
    partition D into D_0 and D_1 based on ϕ;
    calculate $IG(D, \phi)$
  end
  if $IG(D, \phi) = 0$ *for all* ϕ **then**
    terminate
  else
    Choose ϕ^* be the feature with the largest $IG(D, \phi)$;
    Create NODE(ϕ^*);
    DTREETRAIN($D_0, \Phi \setminus \{\phi^*\}$);
    DTREETRAIN($D_1, \Phi \setminus \{\phi^*\}$);
  end
end

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

Different splitting criteria

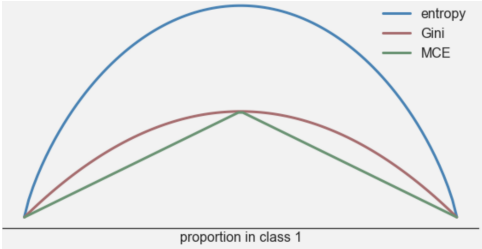
$$IG(X_{\text{parent}}, i) = I(X_{\text{parent}}) - \frac{|X_{i,\text{left}}|}{|X_{\text{parent}}|} I(X_{\text{left}}) - \frac{|X_{i,\text{right}}|}{|X_{\text{parent}}|} I(X_{\text{right}})$$

p_c: number of examples from class/label c in X

- Entropy: $I(X) = -\sum_c p_c \log_2 p_c$
- Misclassification error (MCE): $I(X) = 1 - \max_c p_c$ i.e., minority labels
- Gini index: $I(X) = 1 - \sum_c p_c^2$

Which Different splitting criteria is better?

- Entropy: $I(X) = -\sum_c p_c \log_2 p_c$
- Misclassification error (MCE): $I(X) = 1 - \max_c p_c$
- Gini index: $I(X) = 1 - \sum_c p_c^2$



Which Different splitting criteria is better?

Scikit-learn only uses entropy and Gini index

sklearn.tree.DecisionTreeClassifier

class `sklearn.tree.` **`DecisionTreeClassifier`**(*, *criterion='gini'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *class_weight=None*, *ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entropy"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

Example: Classification Error vs. Entropy

Assume a binary classification task with trianing data with 40 positive and 80 negative examples

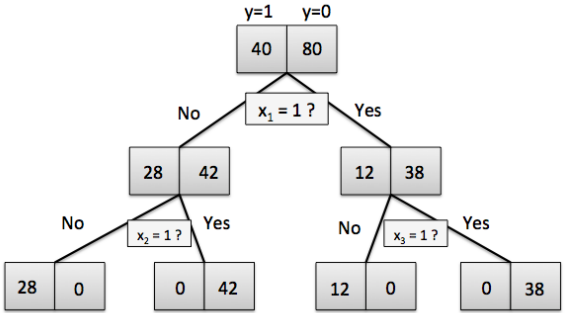
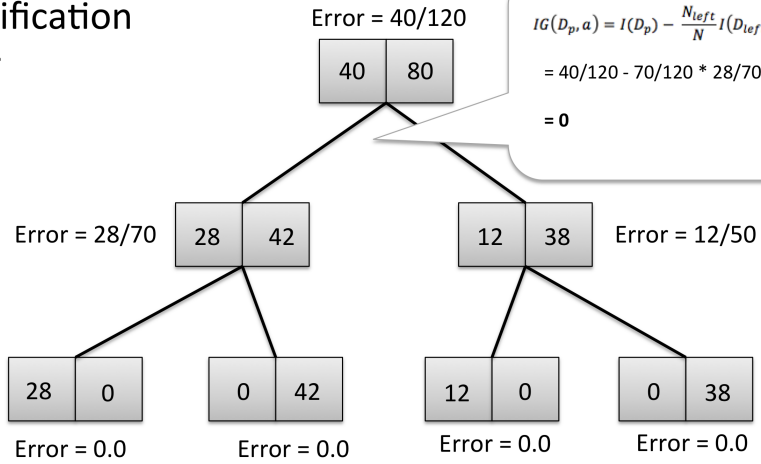


Image from `https://sebastianraschka.com/faq/docs/decisiontree-error-vs-entropy.html`

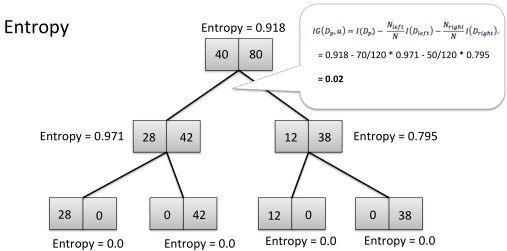
Example: Classification Error vs. Entropy

Classification Error



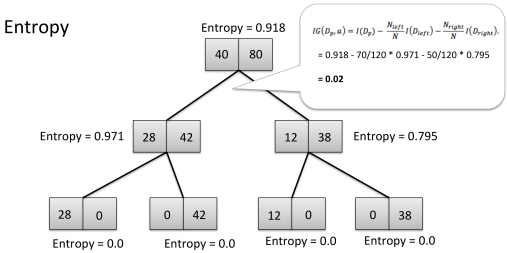
Example: Classification Error vs. Entropy

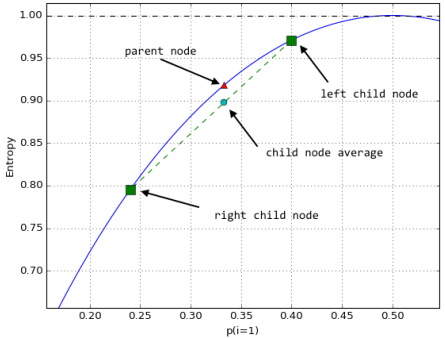
Entropy is strictly concave, avoids IG being 0



Example: Classification Error vs. Entropy

Entropy is strictly concave, avoids IG being 0





Outline

Loss/Cost/Objective Function

We use those terms interchangeably[1] in this class

Gives idea on how "bad" the prediction of the model is

- Information Gain
- Squared loss: $(y_{\text{prediction}} - y_{train})^2$
- Absolute loss: $|y_{\text{prediction}} - y_{train}|$

[1] Following Ian Goodfellow and Yoshua Bengio and Aaron Courville's deep learning book `https://www.deeplearningbook.org/`

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	Labels
X	Y

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

Formal Definitions

- l: Loss function
- Labels Y, e.g., $y \in \{+1, -1\}$, $y \in \mathbb{R}$
- Dataset X
- Target function we wish to learn f: $X \to Y$ (f is unknown)
- Function a machine learning model learns h: $X \to Y$

Formal Definitions

- l: Loss function
- Labels Y, e.g., $y \in \{+1, -1\}$, $y \in \mathbb{R}$
- Dataset X
- Target function we wish to learn f: $X \to Y$ (f is unknown)
- Function a machine learning model learns h: $X \to Y$
- A training example $(\boldsymbol{x}, y) \in (X, Y)$
- Training data S_{train}: collection of examples observed during training

Formal Definitions

- Goal:

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

What is Hard in Finding h?

Assumption: Training and test/unseen sets follow probability distribution D.

- $D(\boldsymbol{x}, y)$: Data generating distribution over $\boldsymbol{x}$ and y
- $(\boldsymbol{x}, y)$: one example in the trainig set sampled from D i.e., $(\boldsymbol{x}, y) \sim D$

What is Hard in Finding h?

Assumption: Training and test/unseen sets follow probability distribution D.

- $D(\boldsymbol{x}, y)$: Data generating distribution over $\boldsymbol{x}$ and y
- $(\boldsymbol{x}, y)$: one example in the trainig set sampled from D i.e., $(\boldsymbol{x}, y) \sim D$

Ideally, we want to minimize the expected loss over D
Hard part of machine learning: D is unknown.

The best we can do is to use the training data sampled from D to approximate expected loss over D

Supervised learning

Given loss function l and $(\boldsymbol{x}, y) \sim D$, expected loss ϵ 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

Given loss function l and $(\boldsymbol{x}, y) \sim D$, expected loss ϵ 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 don't know what D is.
and we only have access to training error for N training samples $S_{\text{train}} = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, N\}$:

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

Supervised learning

Given loss function l and $(\boldsymbol{x}, y) \sim D$, expected loss ϵ 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 don't know what D is.
and we only have access to training error for N training samples $S_{\text{train}} = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, N\}$:

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

Since D is often unknown, we minimize training error in practice.

Empirical Risk Minimization

Minimizing $\hat{\epsilon}$ is also called as **Empirical Risk Minimization**

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

Can derive bounds on the difference between $\hat{\epsilon}$ and ϵ [Vapnik, 1992] (Extended Reading).

So how does machine learning work at all?

Image Classification Task: An Exercise, Daume [2017], chapter 2.

Training sets from class A and class B are given as follows:
What feature would be good to classify A vs. B?

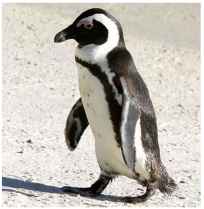
Class A

Class B

An Exercise, Daume [2017], chapter 2.

Test

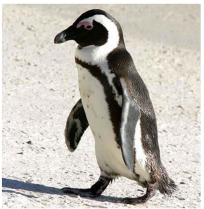


An Exercise, Daume [2017], chapter 2.

Test



Bird or not? Flying animal or not?

So how does machine learning work at all?

Just as you had a tendency to focus on a certain type of function f, machine learning algorithms correspond to classes of functions (F) and preferences within the class.

So how does machine learning work at all?

Just as you had a tendency to focus on a certain type of function f, machine learning algorithms correspond to classes of functions (F) and preferences within the class.

Inductive bias

So how does machine learning work at all?

Just as you had a tendency to focus on a certain type of function f, machine learning algorithms correspond to classes of functions (F) and preferences within the class.

Inductive bias

E.g., Depth of decision trees should be shallow: “Prediction can be made by looking a small subset of features”

References

Hal Daume. *A Course in Machine Learning (v0.9)*. Self-published at `http://ciml.info/`, 2017.

Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992.