2019 CS420, Machine Learning, Lecture 5

Tree Models

Weinan Zhang

Shanghai Jiao Tong University

http://wnzhang.net

http://wnzhang.net/teaching/cs420/index.html

ML Task: Function Approximation

- Problem setting
  - Instance feature space $\mathcal{X}$
  - Instance label space $\mathcal{Y}$
  - Unknown underlying function (target) $f : \mathcal{X} \mapsto \mathcal{Y}$
  - Set of function hypothesis $H = \{h | h : \mathcal{X} \mapsto \mathcal{Y}\}$

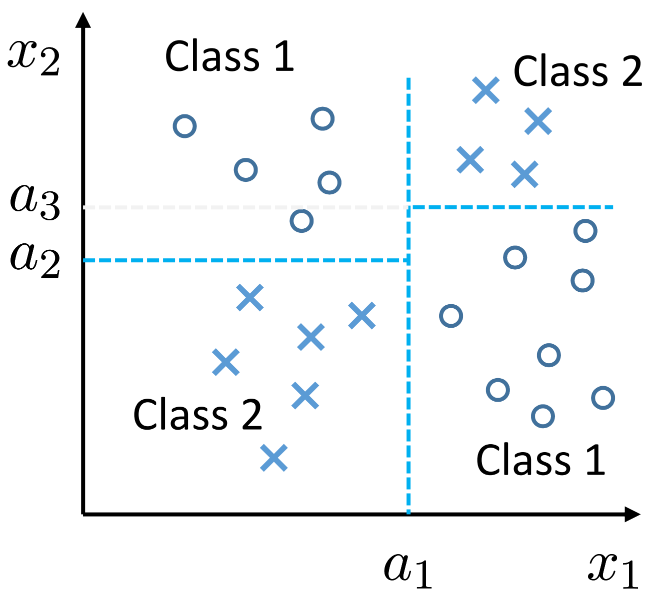
- Input: training data generated from the unknown

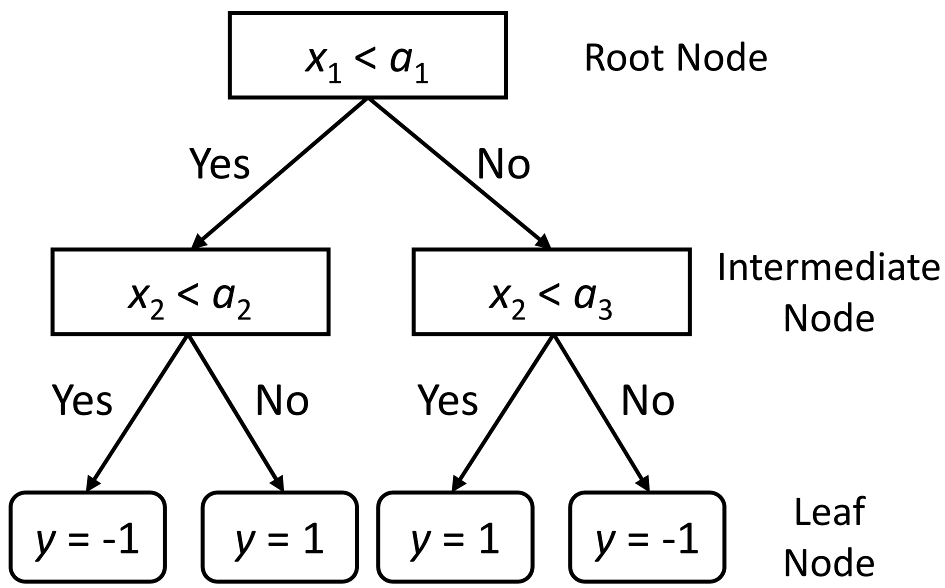
$$\{(x^{(i)}, y^{(i)})\} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$$

- Output: a hypothesis $h \in H$ that best approximates f
- Optimize in functional space, not just parameter space

Optimize in Functional Space

- Tree models
  - Intermediate node for splitting data
  - Leaf node for label prediction
- Continuous data example

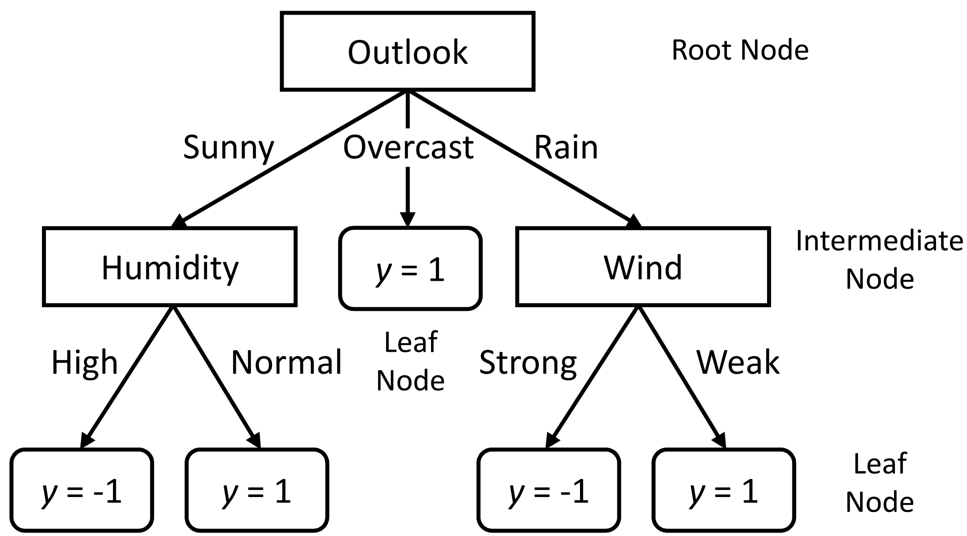




Optimize in Functional Space

- Tree models
  - Intermediate node for splitting data
  - Leaf node for label prediction
- Discrete/categorical data example

Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class Play=Yes Play=No**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No



Decision Tree Learning

- Problem setting
  - Instance feature space $\mathcal{X}$
  - Instance label space $\mathcal{Y}$
  - Unknown underlying function (target) $f : \mathcal{X} \mapsto \mathcal{Y}$
  - Set of function hypothesis $H = \{h | h : \mathcal{X} \mapsto \mathcal{Y}\}$

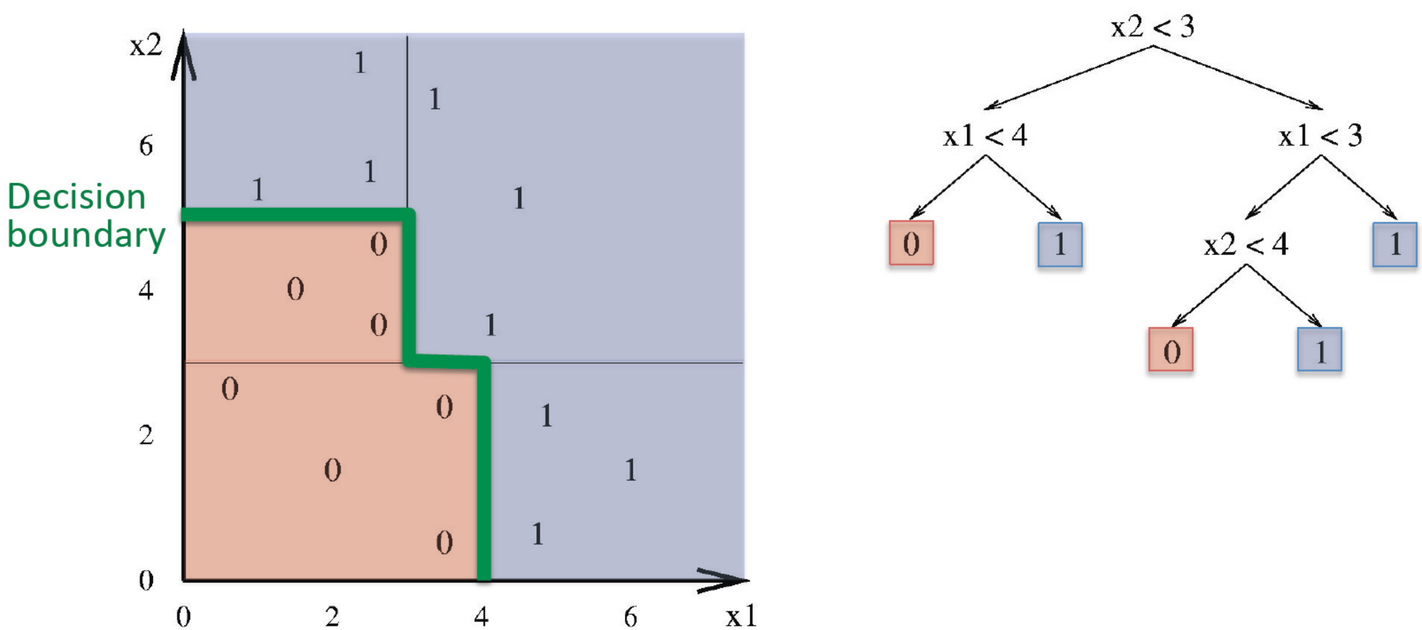
- Input: training data generated from the unknown

$$\{(x^{(i)}, y^{(i)})\} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$$

- Output: a hypothesis $h \in H$ that best approximates f
- Here each hypothesis h is a decision tree

Decision Tree – Decision Boundary

- Decision trees divide the feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label
  - or a probabilistic distribution over labels



Slide credit: Eric Eaton

History of Decision-Tree Research

- Hunt and colleagues used exhaustive search decision-tree methods (CLS) to model human concept learning in the 1960's.
- In the late 70's, Quinlan developed ID3 with the information gain heuristic to learn expert systems from examples.
- Simultaneously, Breiman and Friedman and colleagues developed CART (Classification and Regression Trees), similar to ID3.
- In the 1980's a variety of improvements were introduced to handle noise, continuous features, missing features, and improved splitting criteria. Various expert-system development tools results.
- Quinlan's updated decision-tree package (C4.5) released in 1993.
- Sklearn (python)Weka (Java) now include ID3 and C4.5

Slide credit: Raymond J. Mooney

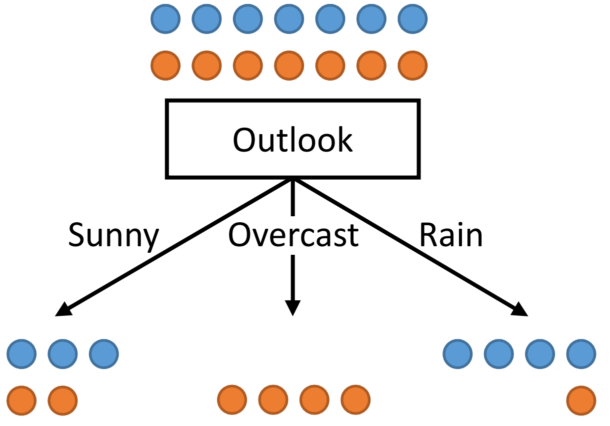
Decision Trees

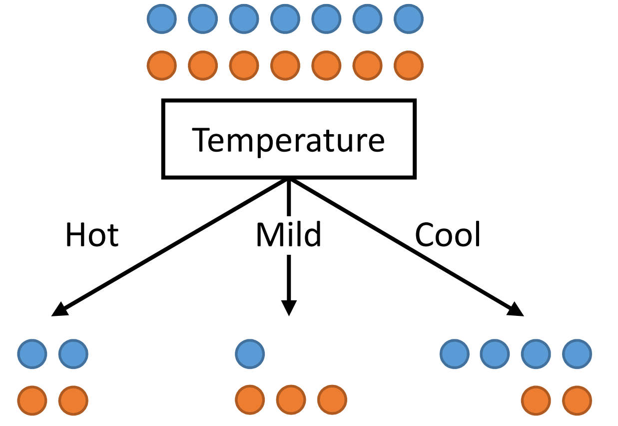
- Tree models
  - Intermediate node for splitting data
  - Leaf node for label prediction

- Key questions for decision trees
  - How to select node splitting conditions?
  - How to make prediction?
  - How to decide the tree structure?

Node Splitting

- Which node splitting condition to choose?





- Choose the features with higher classification capacity
  - Quantitatively, with higher information gain

Fundamentals of Information Theory

- Entropy (more specifically, Shannon entropy) is the expected value (average) of the information contained in each message.
- Suppose *X* is a random variable with *n* discrete values

$$P(X = x_i) = p_i$$

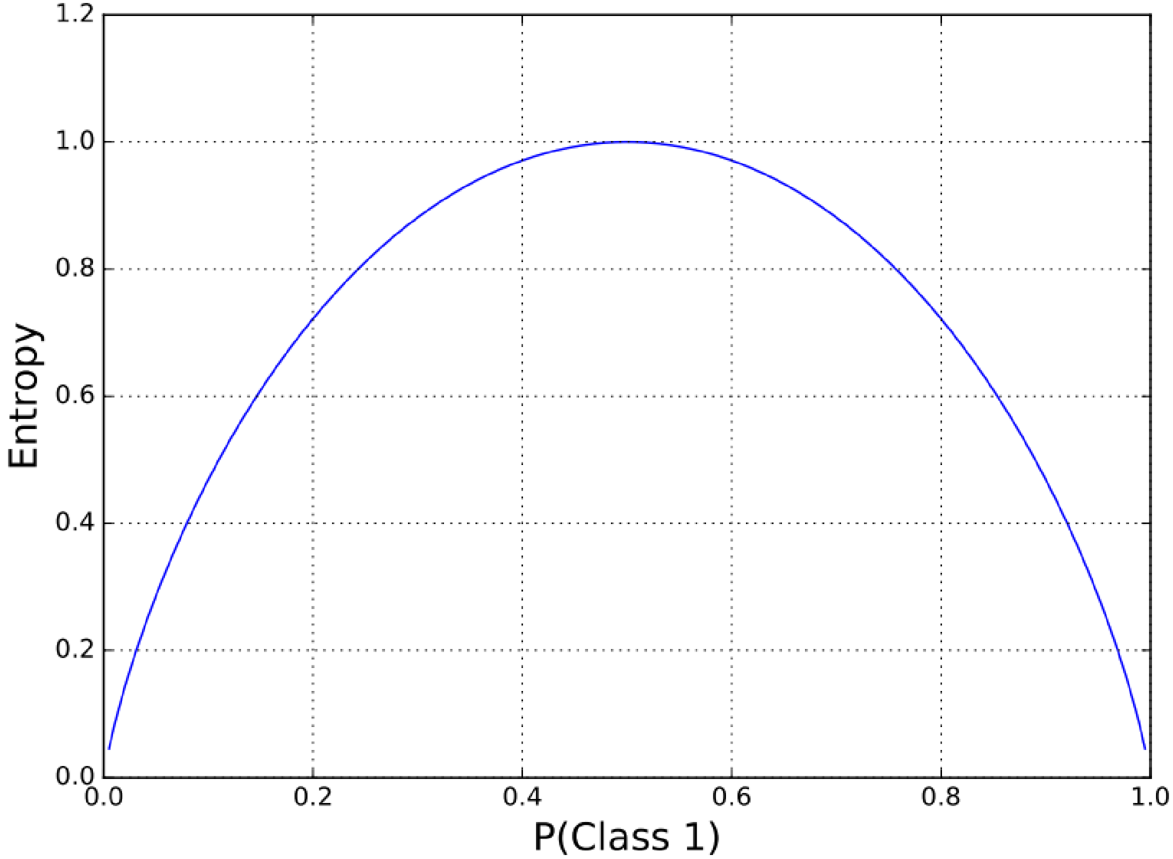
- then its entropy *H*(*X*) is

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i$$

- It is easy to verify

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i \le -\sum_{i=1}^{n} \frac{1}{n} \log \frac{1}{n} = \log n$$

Illustration of Entropy



- Entropy of binary distribution

$$H(X) = -p_1 \log p_1 - (1 - p_1) \log(1 - p_1)$$

Cross Entropy

- Cross entropy is used to measure the difference between two random variable distributions

$$H(X,Y) = -\sum_{i=1}^{n} P(X=i) \log P(Y=i)$$

- Continuous formulation

$$H(p,q) = -\int p(x) \log q(x) dx$$

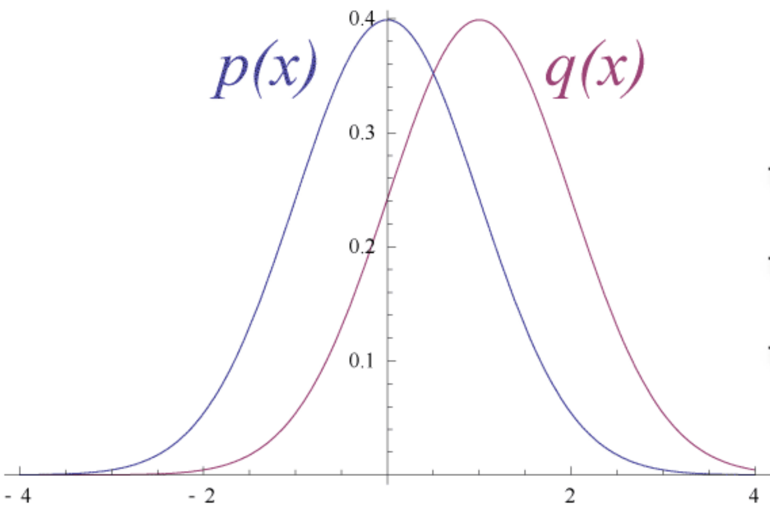
- Compared to KL divergence

$$D_{\mathrm{KL}}(p\|q) = \int p(x) \log \frac{p(x)}{q(x)} dx = H(p,q) - H(p)$$

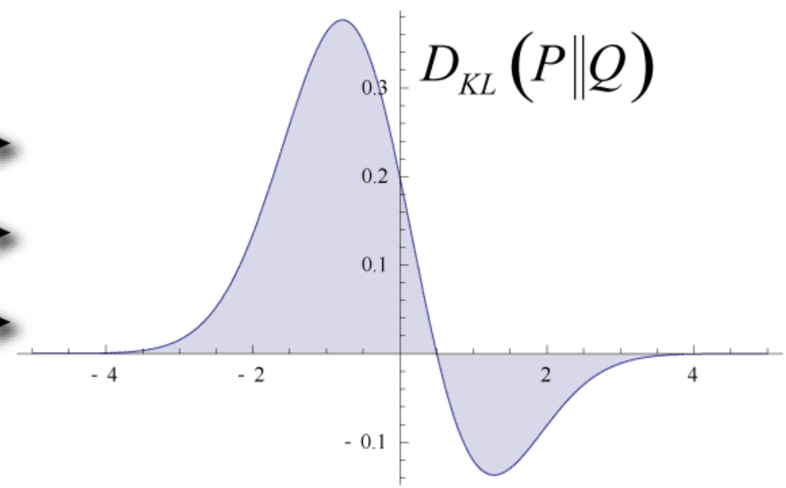
KL-Divergence

Kullback–Leibler divergence (also called relative entropy) is a measure of how one probability distribution diverges from a second, expected probability distribution

$$D_{\mathrm{KL}}(p\|q) = \int p(x)\log\frac{p(x)}{q(x)}dx = H(p,q) - H(p)$$



Original Gaussian PDF's



KL Area to be Integrated

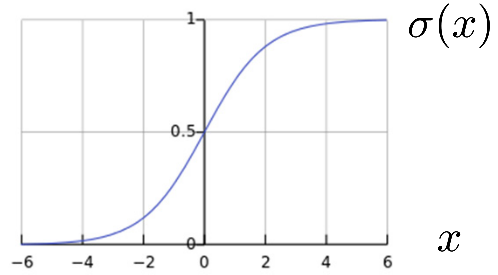
Review

Cross Entropy in Logistic Regression

- Logistic regression is a binary classification model

$$p_\theta(y=1|x) = \sigma(\theta^\top x) = \frac{1}{1+e^{-\theta^\top x}}$$

$$p_\theta(y=0|x) = \frac{e^{-\theta^\top x}}{1+e^{-\theta^\top x}}$$



- Cross entropy loss function

$$\mathcal{L}(y,x,p_\theta) = -y\log\sigma(\theta^\top x) - (1-y)\log(1-\sigma(\theta^\top x))$$

- Gradient

$$\frac{\partial \mathcal{L}(y,x,p_\theta)}{\partial\theta} = -y\frac{1}{\sigma(\theta^\top x)}\sigma(z)(1-\sigma(z))x - (1-y)\frac{-1}{1-\sigma(\theta^\top x)}\sigma(z)(1-\sigma(z))x$$

$$= (\sigma(\theta^\top x) - y)x$$

$$\theta \leftarrow \theta + (y - \sigma(\theta^\top x))x$$

$$\boxed{\frac{\partial\sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))}$$

Conditional Entropy

- Entropy $$H(X) = -\sum_{i=1}^{n} P(X = i) \log P(X = i)$$

- Specific conditional entropy of X given $Y = v$

$$H(X|Y = v) = -\sum_{i=1}^{n} P(X = i|Y = v) \log P(X = i|Y = v)$$

- Specific conditional entropy of X given Y

$$H(X|Y) = \sum_{v \in \text{values}(Y)} P(Y = v) H(X|Y = v)$$

- Information Gain of X given Y

$$\begin{aligned} I(X,Y) =& H(X) - H(X|Y) = H(Y) - H(Y|X) \\ =& H(X) + H(Y) - H(X,Y) \end{aligned}$$

Entropy of (X,Y) instead of cross entropy

Information Gain

- Information Gain of *X* given *Y*

$$
\begin{aligned}
& I(X,Y) = H(X) - H(X|Y) \\
= & -\sum_v P(X=v)\log P(X=v) + \sum_u P(Y=u)\sum_v P(X=v|Y=u)\log P(X=v|Y=u) \\
= & -\sum_v P(X=v)\log P(X=v) + \sum_u \sum_v P(X=v,Y=u)\log P(X=v|Y=u) \\
= & -\sum_v P(X=v)\log P(X=v) + \sum_u \sum_v P(X=v,Y=u)[\log P(X=v,Y=u) - \log P(Y=u)] \\
= & -\sum_v P(X=v)\log P(X=v) - \sum_u P(Y=u)\log P(Y=u) + \sum_{u,v} P(X=v,Y=u)\log P(X=v,Y=u) \\
= & H(X) + H(Y) - H(X,Y)
\end{aligned}
$$

Entropy of (*X*,*Y*) instead of cross entropy

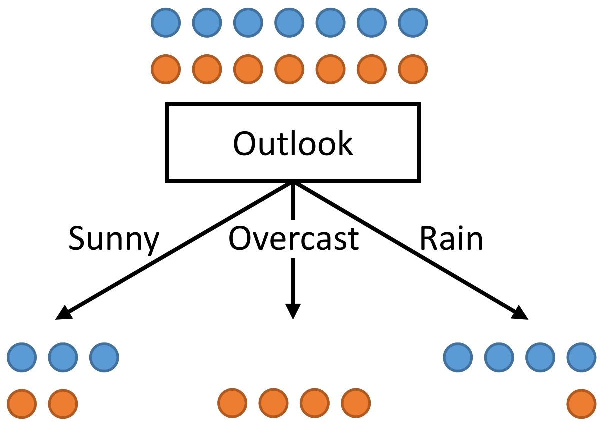
$$H(X,Y) = -\sum_{u,v} P(X=v,Y=u)\log P(X=v,Y=u)$$

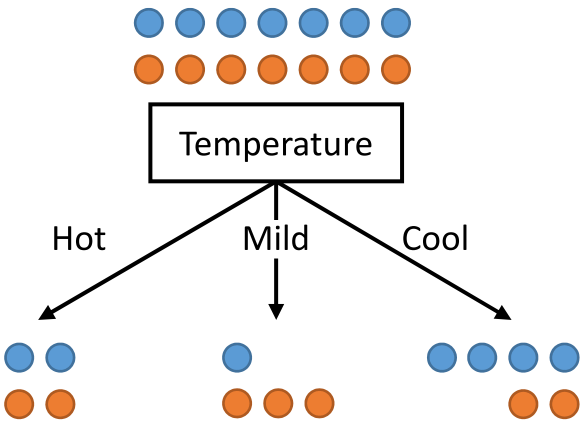
Node Splitting

- Information gain

$$H(X|Y=v) = -\sum_{i=1}^{n} P(X=i|Y=v)\log P(X=i|Y=v)$$

$$H(X|Y) = \sum_{v\in\text{values}(Y)} P(Y=v)H(X|Y=v)$$





$$H(X|Y=S) = -\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} = 0.9710$$

$$H(X|Y=O) = -\frac{4}{4}\log\frac{4}{4} = 0$$

$$H(X|Y=R) = -\frac{4}{5}\log\frac{4}{5} - \frac{1}{5}\log\frac{1}{5} = 0.7219$$

$$H(X|Y) = \frac{5}{14}\times 0.9710 + \frac{4}{14}\times 0 + \frac{5}{14}\times 0.7219 = 0.6046$$

$$I(X,Y) = H(X) - H(X|Y) = 1 - 0.6046 = 0.3954$$

$$H(X|Y=H) = -\frac{2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4} = 1$$

$$H(X|Y=M) = -\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4} = 0.8113$$

$$H(X|Y=C) = -\frac{4}{6}\log\frac{4}{6} - \frac{2}{6}\log\frac{2}{6} = 0.9183$$

$$H(X|Y) = \frac{4}{14}\times 1 + \frac{4}{14}\times 0.8113 + \frac{5}{14}\times 0.9183 = 0.9111$$

$$I(X,Y) = H(X) - H(X|Y) = 1 - 0.9111 = 0.0889$$

Information Gain Ratio

- The ratio between information gain and the entropy

$$I_R(X,Y) = \frac{I(X,Y)}{H_Y(X)} = \frac{H(X) - H(X|Y)}{H_Y(X)}$$

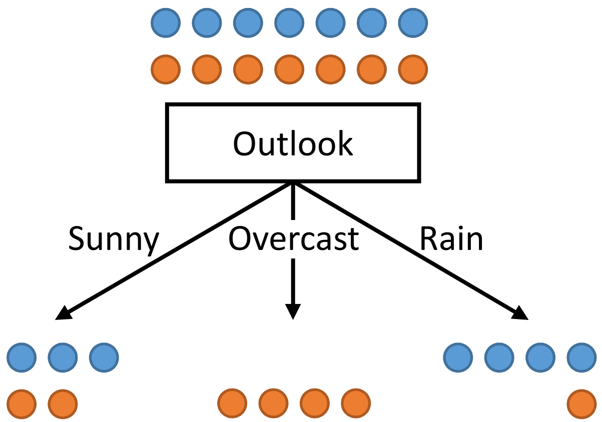
- where the entropy (of *Y*) is

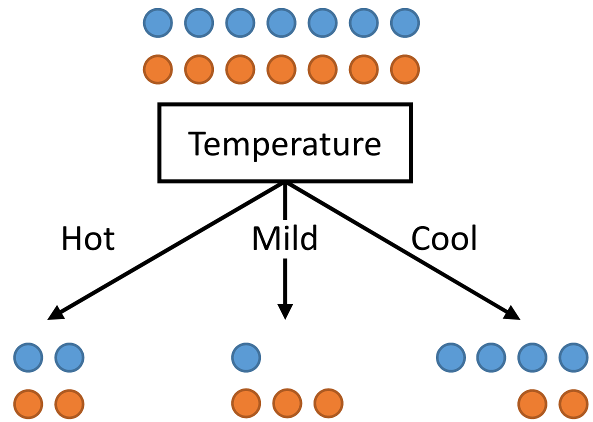
$$H_Y(X) = -\sum_{v \in \mathrm{values}(Y)} \frac{|X_{y=v}|}{|X|} \log \frac{|X_{y=v}|}{|X|}$$

- where $|X_{y=v}|$ is the number of observations with the feature *y*=*v*
- NOTE: $H_Y(X)$ measures how much the variable Y could partition the data itself.
- Normally we don't want a *Y* that yields a good information gain of *X* just because *Y* itself performs a fine-grained partition of the data.

Node Splitting

- Information gain ratio $$I_R(X,Y) = \frac{I(X,Y)}{H_Y(X)} = \frac{H(X) - H(X|Y)}{H_Y(X)}$$





$$I(X,Y) = H(X) - H(X|Y) = 1 - 0.6046 = 0.3954$$
$$H_Y(X) = -\frac{5}{14}\log\frac{5}{14} - \frac{4}{14}\log\frac{4}{14} - \frac{5}{14}\log\frac{5}{14} = 1.5774$$
$$I_R(X,Y) = \frac{0.3954}{1.5774} = 0.2507$$

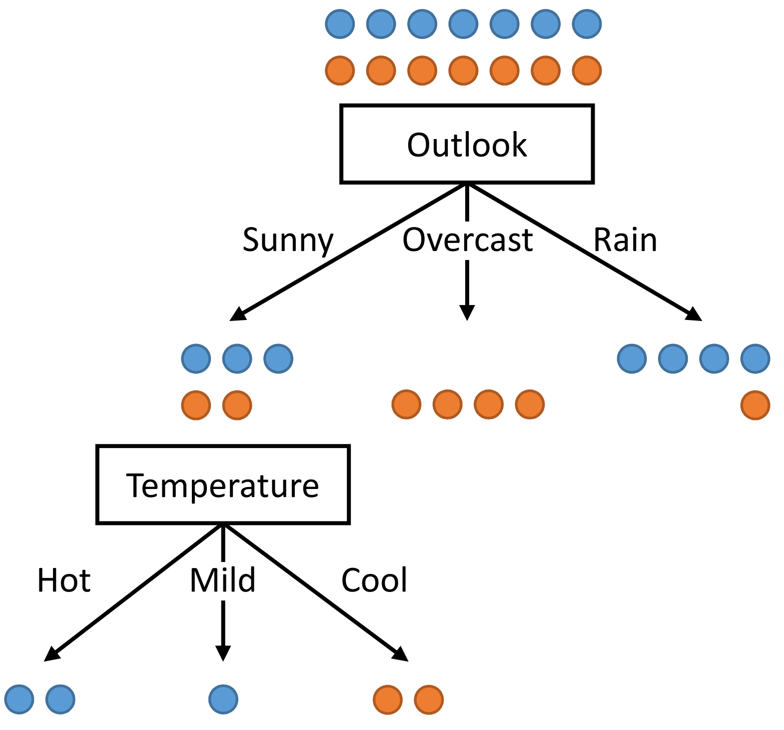
$$I(X,Y) = H(X) - H(X|Y) = 1 - 0.9111 = 0.0889$$
$$H_Y(X) = -\frac{4}{14}\log\frac{4}{14} - \frac{4}{14}\log\frac{4}{14} - \frac{6}{14}\log\frac{6}{14} = 1.5567$$
$$I_R(X,Y) = \frac{0.0889}{1.5567} = 0.0571$$

Decision Tree Building: ID3 Algorithm

- ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan
  - ID3 is the precursor to the C4.5 algorithm
- Algorithm framework
  - Start from the root node with all data
  - For each node, calculate the information gain of all possible features
  - Choose the feature with the highest information gain
  - Split the data of the node according to the feature
  - Do the above recursively for each leaf node, until
    - There is no information gain for the leaf node
    - Or there is no feature to select

Decision Tree Building: ID3 Algorithm

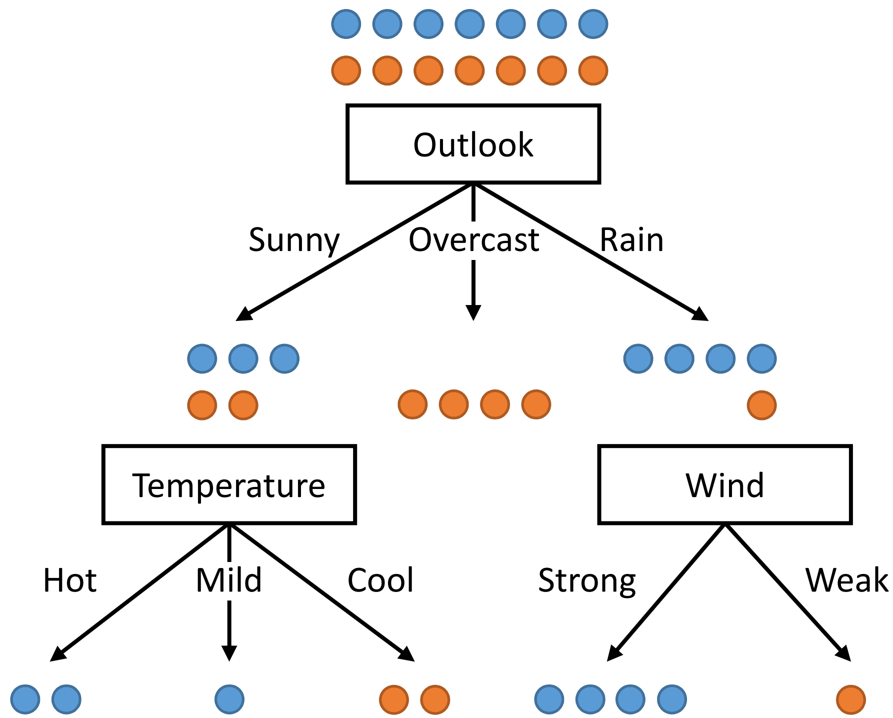
- An example decision tree from ID3



- Each path only involves a feature at most once

Decision Tree Building: ID3 Algorithm

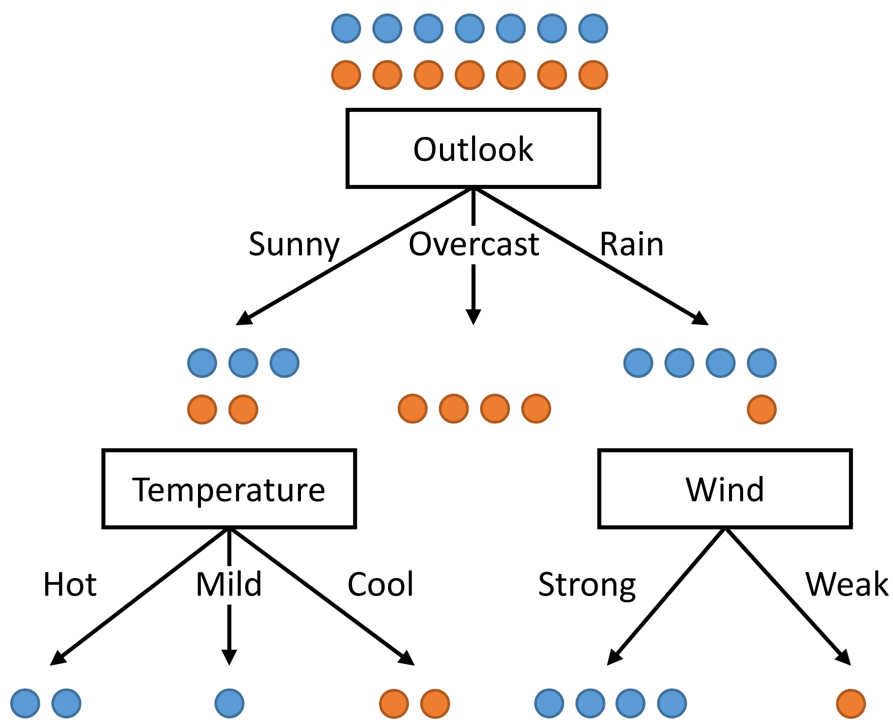
- An example decision tree from ID3



- How about this tree, yielding perfect partition?

Overfitting

- Tree model can approximate any finite data by just growing a leaf node for each instance



Decision Tree Training Objective

- Cost function of a tree *T* over training data

$$C(T) = \sum_{t=1}^{|T|} N_t H_t(T)$$

where for the leaf node *t*

- $H_t(T)$ is the empirical entropy
- N_t is the instance number, N_{tk} is the instance number of class *k*

$$H_t(T) = -\sum_k \frac{N_{tk}}{N_t} \log \frac{N_{tk}}{N_t}$$

- Training objective: find a tree to minimize the cost

$$\min_T C(T) = \sum_{t=1}^{|T|} N_t H_t(T)$$

Decision Tree Regularization

- Cost function over training data

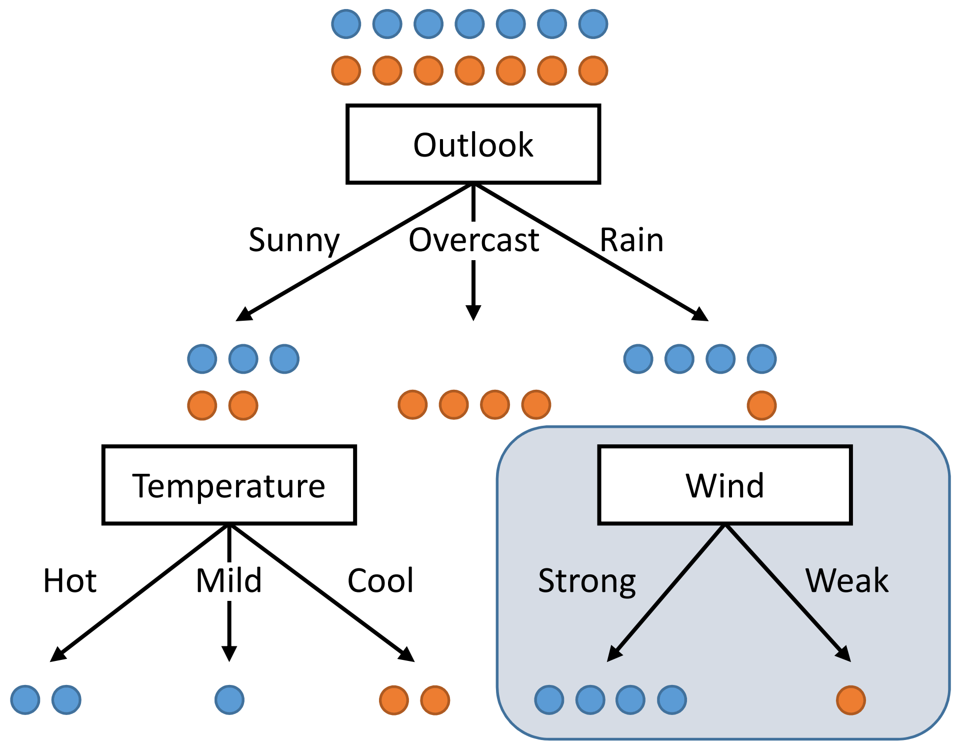
$$C(T) = \sum_{t=1}^{|T|} N_t H_t(T) + \lambda |T|$$

where

- $|T|$ is the number of leaf nodes of the tree T
- λ is the hyperparameter of regularization

Decision Tree Building: ID3 Algorithm

- An example decision tree from ID3



Whether to split this node?

- Calculate the cost function difference. $C(T) = \sum_{t=1}^{|T|} N_t H_t(T) + \lambda |T|$

Summary of ID3

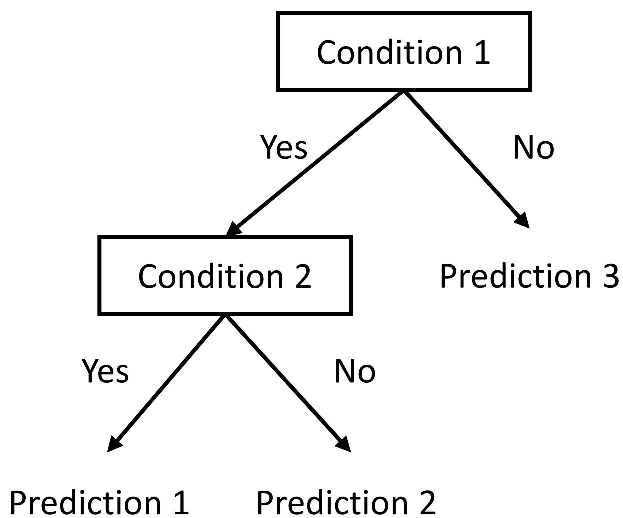
- A classic and straightforward algorithm of training decision trees
  - Work on discrete/categorical data
  - One branch for each value/category of the feature

- Algorithm C4.5 is similar and more advanced to ID3
  - Splitting the node according to information gain ratio

- Splitting branch number depends on the number of different categorical values of the feature
  - Might lead to very broad tree

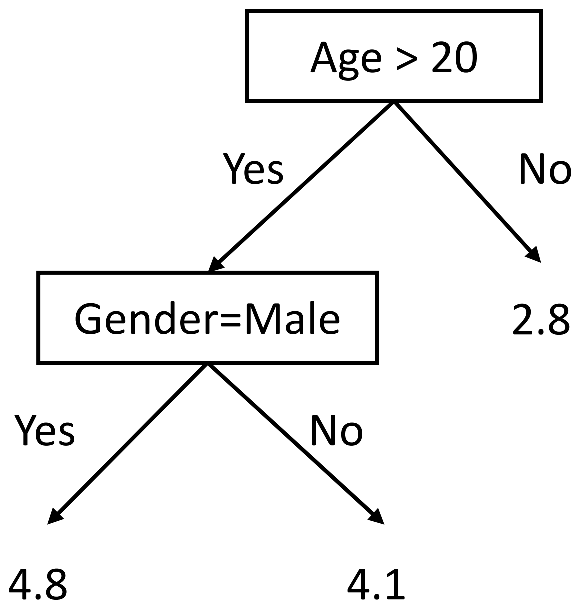
CART Algorithm

- Classification and Regression Tree (CART)
  - Proposed by Leo Breiman et al. in 1984
  - Binary splitting (yes or no for the splitting condition)
  - Can work on continuous/numeric features
  - Can repeatedly use the same feature (with different splitting)



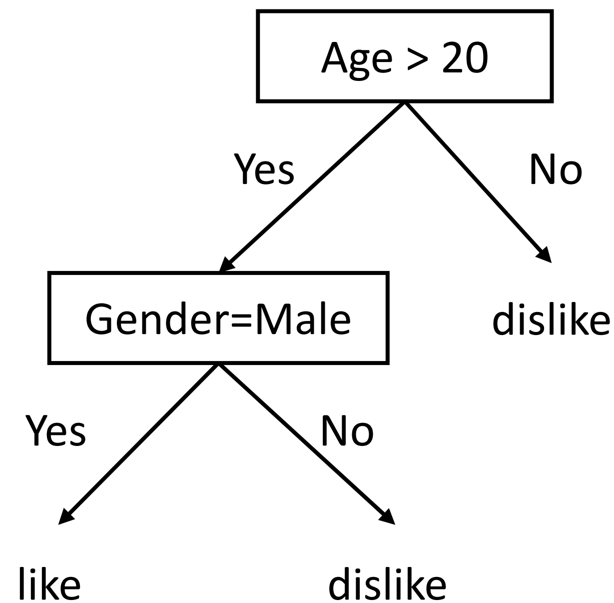
CART Algorithm

- Regression Tree
  - Output the predicted value



For example: predict the user’s rating to a movie

- Classification Tree
  - Output the predicted class



For example: predict whether the user like a move

Regression Tree

- Let the training dataset with continuous targets *y* be

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

- Suppose a regression tree has divided the space into *M* regions R_1, R_2, …, R_M, with c_m as the prediction for region R_m

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

- Loss function for (x_i, y_i)

$$\frac{1}{2}(y_i - f(x_i))^2$$

- It is easy to see the optimal prediction for region *m* is

$$\hat{c}_m = \mathrm{avg}(y_i | x_i \in R_m)$$

Regression Tree

- How to find the optimal splitting regions?
- How to find the optimal splitting conditions?
  - Defined by a threshold value *s* on variable *j*
  - Lead to two regions

$$R_1(j, s) = \{x|x^{(j)} \leq s\} \qquad R_2(j, s) = \{x|x^{(j)} > s\}$$

- Training based on current splitting

$$\min_{j,s} \left[\min_{c_1} \sum_{x \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x \in R_2(j,s)} (y_i - c_2)^2 \right]$$

$$\hat{c}_m = \mathrm{avg}(y_i | x_i \in R_m)$$

Regression Tree Algorithm

- INPUT: training data *D*
- OUTPUT: regression tree *f(x)*
- Repeat until stop condition satisfied:
  - Find the optimal splitting (*j*,*s*)

$$\min_{j,s}\left[\min_{c_1}\sum_{x\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x\in R_2(j,s)}(y_i-c_2)^2\right]$$

  - Calculate the prediction value of the new region R_1, R_2

$$\hat{c}_m=\mathrm{avg}(y_i|x_i\in R_m)$$

- Return the regression tree

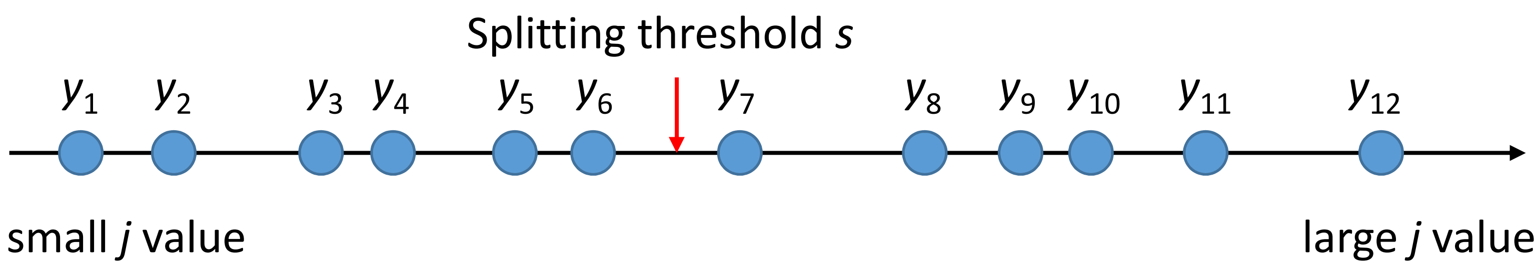
$$f(x)=\sum_{m=1}^{M}\hat{c}_m I(x\in R_m)$$

Regression Tree Algorithm

- How to efficiently find the optimal splitting (*j*,*s*)?

$$\min_{j,s}\left[\min_{c_1}\sum_{x\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x\in R_2(j,s)}(y_i-c_2)^2\right]$$

- Sort the data ascendingly according to feature *j* value



$$\text{loss}=\sum_{i=1}^{6}(y_i-c_1)^2+\sum_{i=7}^{12}(y_i-c_2)^2$$

Online updated

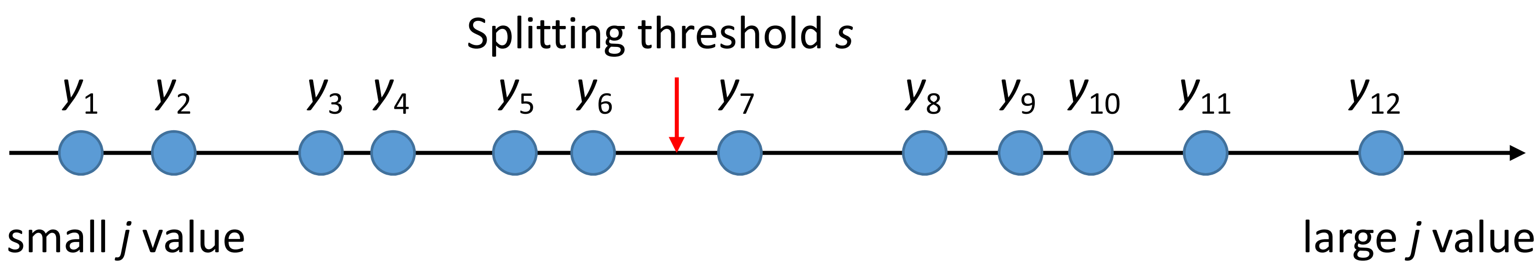
$$=\sum_{i=1}^{6}y_i^2-\frac{1}{6}\Big(\sum_{i=1}^{6}y_i\Big)^2+\sum_{i=7}^{12}y_i^2-\frac{1}{6}\Big(\sum_{i=7}^{12}y_i\Big)^2=-\frac{1}{6}\Big(\sum_{i=1}^{6}y_i\Big)^2-\frac{1}{6}\Big(\sum_{i=7}^{12}y_i\Big)^2+C$$

Regression Tree Algorithm

- How to efficiently find the optimal splitting (j,s)?

$$\min_{j,s}\left[\min_{c_1}\sum_{x\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x\in R_2(j,s)}(y_i-c_2)^2\right]$$

- Sort the data ascendingly according to feature j value



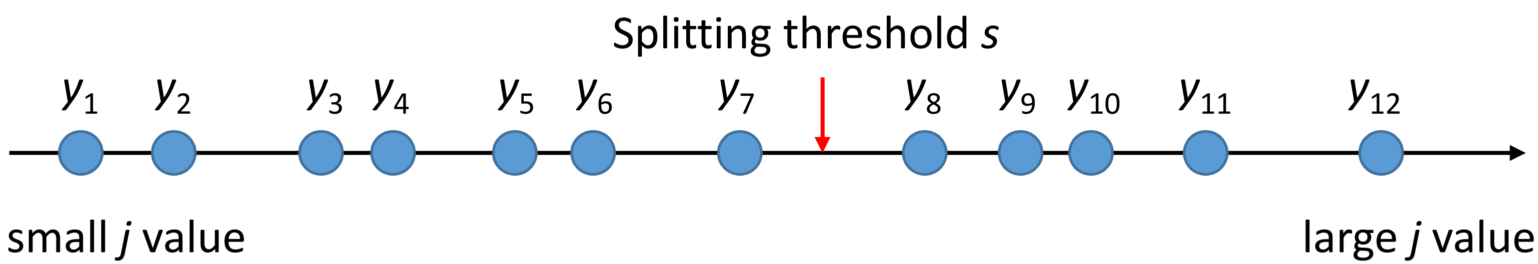
$$\text{loss}_{6,7}=-\frac{1}{6}\Big(\sum_{i=1}^{6}y_i\Big)^2-\frac{1}{6}\Big(\sum_{i=7}^{12}y_i\Big)^2+C$$

Regression Tree Algorithm

- How to efficiently find the optimal splitting (j,s)?

$$\min_{j,s}\left[\min_{c_1}\sum_{x\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x\in R_2(j,s)}(y_i-c_2)^2\right]$$

- Sort the data ascendingly according to feature j value



$$\text{loss}_{6,7}=-\frac{1}{6}\left(\sum_{i=1}^{6}y_i\right)^2-\frac{1}{6}\left(\sum_{i=7}^{12}y_i\right)^2+C$$

$$\downarrow \qquad \downarrow$$

$$\text{loss}_{7,8}=-\frac{1}{7}\left(\sum_{i=1}^{7}y_i\right)^2-\frac{1}{5}\left(\sum_{i=8}^{12}y_i\right)^2+C$$

- Maintain and online update in $O(1)$ Time

$$\text{Sum}(R_1)=\sum_{i=1}^{k}y_i \qquad \text{Sum}(R_2)=\sum_{i=k+1}^{n}y_i$$

- $O(n)$ in total for checking one feature

Classification Tree

- The training dataset with categorical targets *y*

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

- Suppose a regression tree has divided the space into *M* regions R_1, R_2, …, R_M, with c_m as the prediction for region R_m

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

- Here the leaf node prediction c_m is the category distribution

$$\hat{c}_m = \{P(y_k | x_i \in R_m)\}_{k=1 \dots K}$$

- c_m is solved by counting categories

$$P(y_k | x_i \in R_m) = \frac{C_m^k}{C_m}$$

instances in leaf *m* with cat *k*

instances in leaf *m*

Classification Tree

- How to find the optimal splitting regions?
- How to find the optimal splitting conditions?
  - For continuous feature *j*, defined by a threshold value *s*
    - Yield two regions

$$R_1(j,s) = \{x | x^{(j)} \leq s\} \qquad R_2(j,s) = \{x | x^{(j)} > s\}$$

  - For categorical feature *j*, select a category *a*
    - Yield two regions

$$R_1(j,s) = \{x | x^{(j)} = a\} \qquad R_2(j,s) = \{x | x^{(j)} \neq a\}$$

  - How to select? Argmin Gini impurity.

Gini Impurity

- In classification problem
  - suppose there are K classes
  - let p_k be the probability of an instance with the class k
  - the Gini impurity index is

$$\mathrm{Gini}(p) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

- Given the training dataset *D*, the Gini impurity is

$$\mathrm{Gini}(D) = 1 - \sum_{k=1}^{K} \left(\frac{|D^k|}{|D|} \right)^2$$

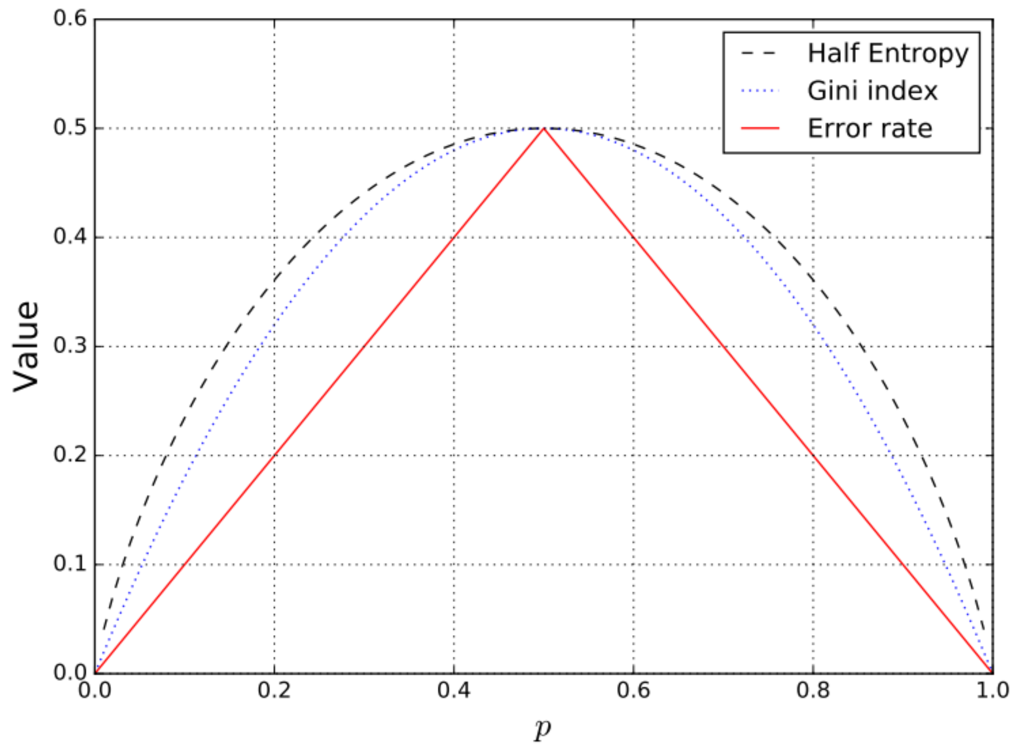
instances in *D* with cat *k*

instances in *D*

Gini Impurity

- For binary classification problem
  - let *p* be the probability of an instance with the class 1
  - Gini impurity is $\mathrm{Gini}(p) = 2p(1-p)$
  - Entropy is $H(p) = -p \log p - (1-p)\log(1-p)$

Gini impurity and entropy are quite similar in representing classification error rate.



Gini Impurity

- With a categorical feature *j* and one of its categories *a*
  - The two split regions R_1, R_2

$$R_1(j,a) = \{x | x^{(j)} = a\} \qquad R_2(j,a) = \{x | x^{(j)} \neq a\}$$

$$D_j^1 = \{(x,y) | x^{(j)} = a\} \qquad D_j^2 = \{(x,y) | x^{(j)} \neq a\}$$

  - The Gini impurity of feature *j* with the selected category *a*

$$\mathrm{Gini}(D_j, j = a) = \frac{|D_j^1|}{|D_j|}\mathrm{Gini}(D_j^1) + \frac{|D_j^2|}{|D_j|}\mathrm{Gini}(D_j^2)$$

Classification Tree Algorithm

- INPUT: training data *D*
- OUTPUT: classification tree *f(x)*
- Repeat until stop condition satisfied:
    - Find the optimal splitting (*j*,a)

$$\min_{j,a} \mathrm{Gini}(D_j, j = a)$$

    - Calculate the prediction distribution of the new region R_1, R_2

$$\hat{c}_m = \{P(y_k | x_i \in R_m)\}_{k=1\ldots K}$$

- Return the classification tree

$$f(x) = \sum_{m=1}^{M} \hat{c}_m I(x \in R_m)$$

1. Node instance number is small
2. Gini impurity is small
3. No more feature

Classification Tree Output

- Class label output
  - Output the class with the highest conditional probability

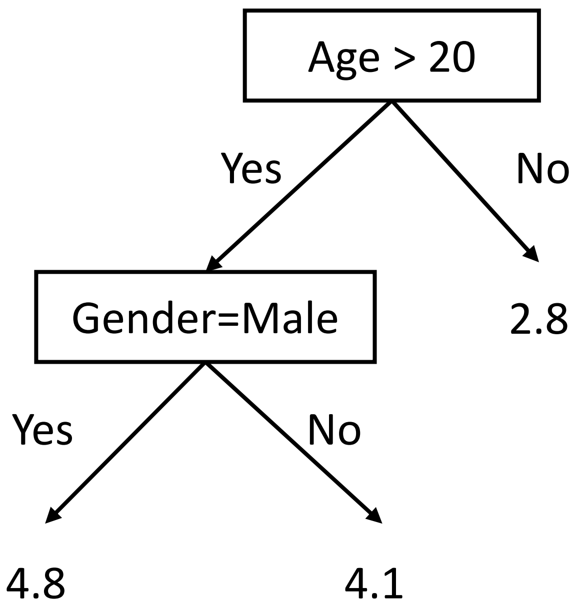
$$f(x) = \arg\max_{y_k} \sum_{m=1}^{M} I(x \in R_m) P(y_k | x_i \in R_m)$$

- Probabilistic distribution output

$$f(x) = \sum_{m=1}^{M} \hat{c}_m I(x \in R_m)$$

$$\hat{c}_m = \{P(y_k | x_i \in R_m)\}_{k=1 \ldots K}$$

Converting a Tree to Rules




```
IF Age > 20:
  IF Gender == Male:
    return 4.8
  ELSE:
    return 4.1
ELSE:
  return 2.8
```

For example: predict the user’s rating to a movie

Decision tree model is easy to be visualized, explained and debugged.

Learning Model Comparison

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of “mixed” type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large N)	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

[Table 10.3 from Hastie et al. Elements of Statistical Learning, 2nd Edition]