Text Classification

CS 7263 Information Retrieval
Lecture 08

Jiho Noh

Department of Computer Science
Kennesaw State University

Fall 2025

Text classification

- **Classification** (also called categorization) is a ubiquitous enabling technology in data science; studied within pattern recognition, statistics, and machine learning.
- Definition: The activity of assigning a predefined class (or cetegory) to a data item belongs to.
- Formulated as the task of generating a hypothesis (or "classifier" or "model")

$$h : D \to C,$$

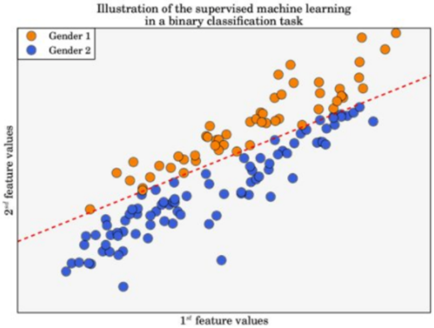
where $D = \{x_1, x_2, \cdots\}$ is a domain of data items and $C = \{c_1, \cdots, c_n\}$ is a finite set of classes.

Text classification — what is and is not

- Different from clustering, where the groups ("clusters") and the number of which are not known in advance
- Classification always involves a subjective judgment; the membership of a data item into a class must not be determinable with certainty.
  - E.g., predicting whether a natural number belongs to Prime or NonPrime is not a classification task.
- In text classification, data items are textual (e.g., news articles, emails, tweets, product reviews, sentences, questions, queries, etc.) or partly textual (e.g., Web pages).

Binary classification

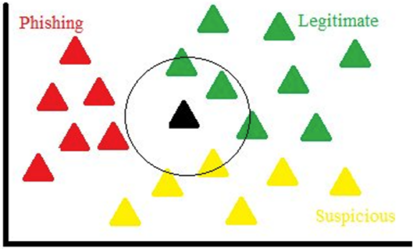
- Tasks that have two classes: {True, False}, {Positive, Negative}, etc.
  - E.g., Assigning emails to one of {Spam, Legitimate}
- Suitable algorithms:
  - Logistic regression
  - Support vector machine (SVM)



Single-label Multi-class classification

- $h : D \to C$ (each item belongs to exactly one class, single-label) and $C = \{c_1, \cdots, c_n\}$ with $n > 2$ (multi-class)
  - E.g., Assigning news articles to one of {Home, News, International, Entertainment, Lifestyles, Sports}
- The number of classes can be very large on some problems. (e.g., biomedical entity classification, $|C| > 30,000$)

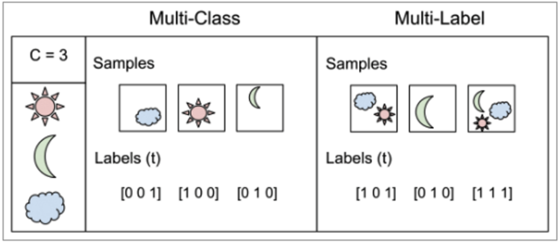
- Suitable algorithms:
  - k-Nearest Neighbors
  - Decision Trees
  - Naïve Bayes
  - Random Forest



Multi-label Multi-class classification

- $h : D \rightarrow 2^C$ (each item may belong to zero, one, or several classes, multi-label) and $C = \{c_1, \cdots, c_n\}$ with $n > 2$ (multi-class)
  - E.g., Assigning computer science articles to the classes in the ACM classification system

- Suitable algorithms:
  - Decision Trees
  - Random Forests
  - Gradient Boosting



Multi-class Classification

How can we perform multi-class classification using a linear classifier, when $C = \{c_1, c_2, \ldots, c_n\}$ and $n > 2$? There are two main solutions:

One-vs-All

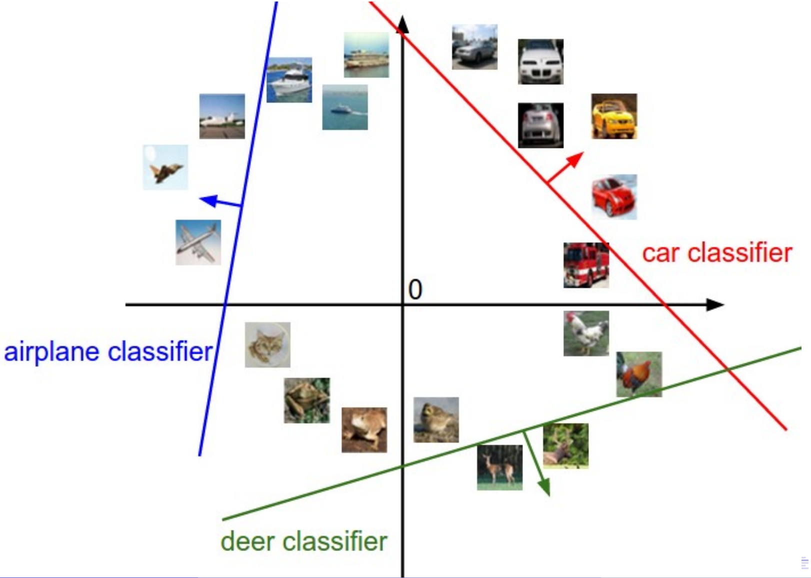
- Train a binary (linear) classifier for each class.
- For example,
  - C_1 vs. $C_2, \ldots, C_n$
  - C_2 vs. $C_1, C_3, \ldots, C_n$
  - C_k vs. $C_1, C_2, \ldots, C_{n-1}$
- If multiple classes are predicted for a single example, choose the one with highest confidence level.

Multi-class Classification (Cont.)

One-vs-One

- Train a classifier for each pair of classes:
- For example,
  - C_1 vs. C_2
  - C_1 vs. C_3,
  - $\cdots$
- A majority vote is then performed to find the correct class.

Multi-class Classification



Application 1: Knowledge Organization

- Long tradition in both science and the humanities; goal was organizing knowledge, i.e., conferring structure to an otherwise unstructured body of knowledge
- The rationale is that using a **structured body of knowledge** is easier / more effective than if this knowledge is unstructured
- **Automated classification** tries to automate the tedious task of assigning data items based on their content, a task otherwise performed by human annotators (a.k.a. “assessors”, or “coders”)

Application 1: Knowledge Organization (cont'd)

- Examples;
  - Classifying news articles for selective dissemination.
  - Classifying scientific papers into specialized taxonomies .
  - Classifying patents.
  - Classifying topic-related tweets by sentiment.
  - ...
- Retrieval (as in search engines) could also be viewed as (binary) classification into *Relevant vs. NonRelevant*.

Application 2: Filtering

- In IR, **Filtering** refers to the activity of blocking a set of NonRelevant items from a dynamic stream, thereby leaving only the Relevant ones.
  - E.g., Spam filtering, attempting to tell legitimate messages from Spam messages.
  - Detecting unsuitable content (e.g., porn, violent content, racist content, cyberbullying, fake news) is also an import application.
- Filtering is an example of binary classification.
- Collaborative filtering in recommendation systems.

Application 3: Empowering other IR tasks

- Functional to improving the effectiveness of other tasks in IR or NLP; e.g.,
  - Classifying queries by intent within search engines
  - Classifying questions by type in question-answering systems
  - Classifying named entities
  - Word sense disambiguation in NLP systems
  - Sentiment analysis
  - ...
- Many of these tasks involve classifying very small texts (e.g., queries, questions, sentences), and stretch the notion of “text” classification quite a bit.

Supervised Learning for Categorization

- A training example is an instance $x \in X$ paired with a class label $c \in C$.
- Given a set of training examples, D,
- we find a hypothesized categorization function, $h()$, such that

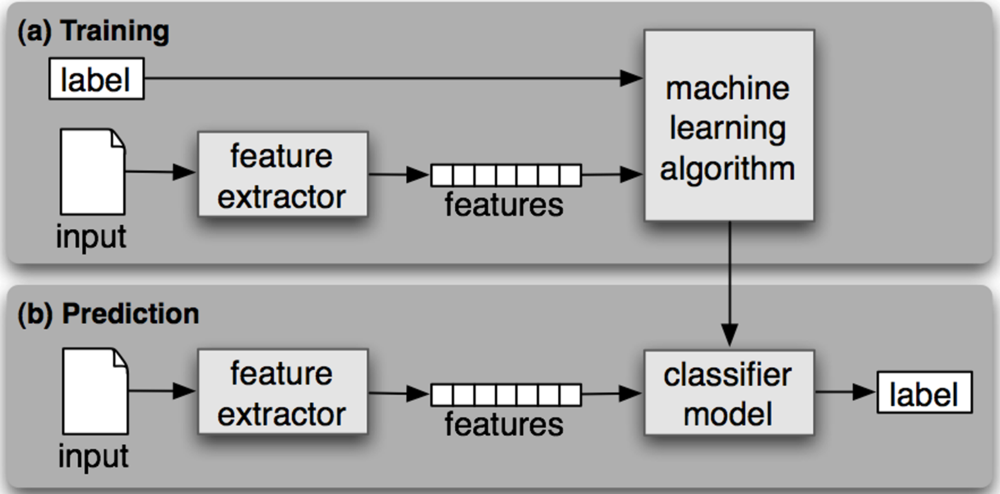
$$\forall (x, c_x) \in D : h(x) = c_x$$

- **Consistency!**

Supervised Learning

- Methods:
  - **Naïve Bayes**: Based on Bayes theorem and the assumption of independence between feature events.
  - **k-Nearest Neighbors**: Decides the class of a sample based on the classes of its k nearest neighbors.
  - **Support-vector machines**: Find the optimal hyperplane that separates classes in a high-dimensional space.
  - **Decision trees and Random forests**: Guilds a tree-like model(s) of decisions based on feature values.
  - **Neural networks**: Deep learning models that can learn complex patterns in data.

Supervised learning for classification



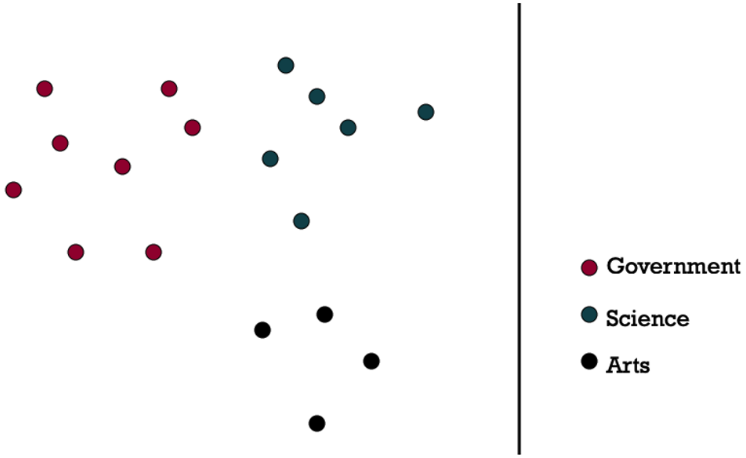
Remember: Vector Space Representation

- Each document is a vector, one component for each term (=word)
- Normally, normalize vectors to unit length.
- High-dimensional vector space:
  - Terms are axes
  - 10,000+ dimensions, or even 100,000+
  - Docs are vectors in this space
- How can we do classification in this space?

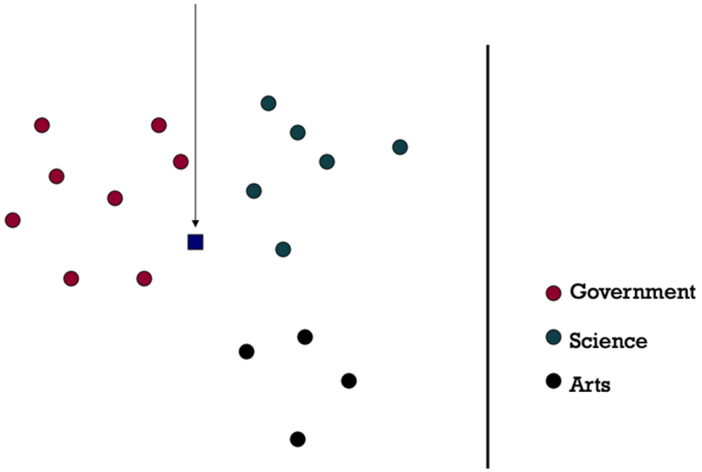
Classification using Vector Spaces

- In vector space classification,
  - training set corresponds to a labeled set of points (equivalently, vectors)
  - **Premise 1**: Documents in the same class form a contiguous region of space (i.e., a cluster)
  - **Premise 2**: Documents from different classes don't overlap (much)
- Learning a classifier is to build surfaces to delineate classes in the space

Documents in a Vector Space

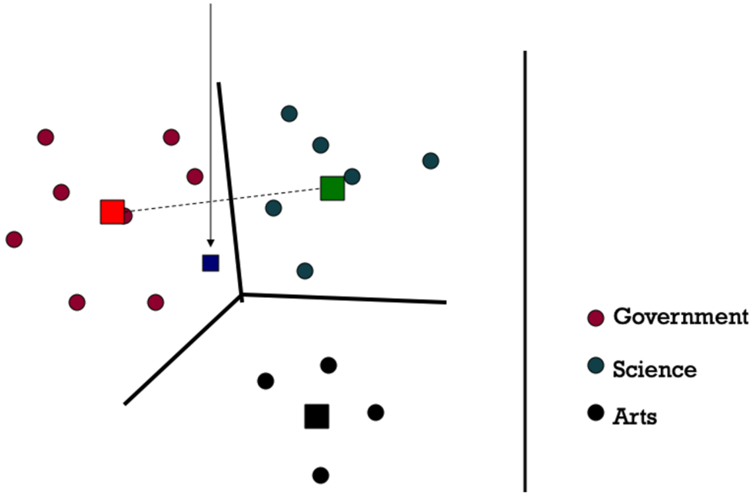


Documents in a Vector Space



A test document of which class?

Documents in a Vector Space



How to find good separators?

Definition of centroid

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

- Where D_c is the set of all documents that belong to class c and $v(d)$ is the vector space representation of d.
- Note that centroid will in general not be a unit vector even when the inputs are unit vectors.

Illustration of Rocchio Text Categorization

- The boundary between two classes in Rocchio classification is the set of points with equal distance from the two centroids.
  - $|a_1| = |a_2|$
  - $|b_1| = |b_2|$
  - $|c_1| = |c_2|$
- The boundary line (or hyperplane) in M-dimensional space is the set of points that satisfy:

$$\vec{w}^{\intercal}\vec{x} = b$$

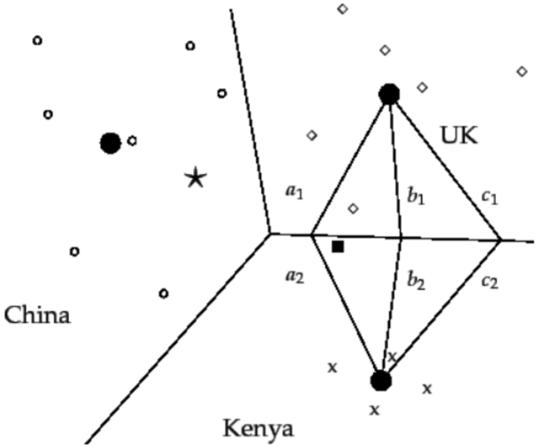


Figure 14.3: Rocchio classification.

Boundary between two classes

- The boundary line (or hyperplane) in M-dimensional space is the set of points that satisfy:

$$\vec{w}^{\intercal}\vec{x} = b$$

- $\vec{w}$ is the M-dimensional normal vector of the hyperplane and b is a constant, such that
  - $\vec{w} = \vec{\mu}(c_1) - \vec{\mu}(c_2)$
  - $b = \frac{1}{2}\left(|\vec{\mu}(c_1)|^2 - |\vec{\mu}(c_2)|^2\right)$
  - **how?**
- Basically, the Rocchio classifier is to determine $\vec{\mu}(c)$ that the point is closest to and then assign it to c.

Rocchio — Training

1. For each class c, compute the centroid $\vec{\mu}(c)$ of the training documents in D_c.
2. Compute the separating hyperplane parameters:

$$\vec{w} = \vec{\mu}(c_1) - \vec{\mu}(c_2)$$

$$b = \frac{1}{2}\left(|\vec{\mu}(c_1)|^2 - |\vec{\mu}(c_2)|^2\right)$$

Rocchio — Test

1. For a test document d, compute the vector $\vec{v}(d)$.
2. Compute the distance to each class centroid:

$$d(c) = |\vec{v}(d) - \vec{\mu}(c)|^2$$

3. Assign d to the class with the smallest distance:

$$c_{pred} = \arg\min_{c \in C} d(c)$$

Rocchio classification — Example

docs	term weights (tf.idf)					
	Chinese	Japan	Tokyo	Macao	Beijing	Shanghai
d1	0	0	0	0	1.0	0
d2	0	0	0	0	0	1.0
d3	0	0	0	1.0	0	0
d4	0	0.71	0.71	0	0	0
d5	0	0.71	0.71	0	0	0
mu_c	0	0	0	0.33	0.33	0.33
mu_j	0	0.71	0.71	0	0	0

$$(1 + \log_{10} tf_{t,d}) \log_{10}(4/df_t)$$

- The separating hyperplane has the following parameters:

$$\vec{w} \approx (0, -0.71, -0.71, 1/3, 1/3, 1/3)^\mathsf{T}$$
$$b = -1/3$$

Simple

a document $d \to x$

Naïve Bayesian Method for Text Classification

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

a class $c \to y$

Naïve Bayes Classifier

$$C_{MAP} = \arg\max_{c \in C} P(c|d)$$

MAP is "Maximum a Posteriori" = most likely class

$$= \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)}$$

Bayes Rule

$$= \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)}$$

Given a document, P(d) is the same over all classes

$$= \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n|c)P(c)$$

Document d represented as features $x_1, x_2, \ldots, x_n$

Naïve Bayes Classifier

$$C_{MAP} = \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n | c) P(c)$$

- Assumptions:
  - Bag of Words: Assume position doesn't matter
  - Conditional Independence: Assume the feature probabilities

$$P(x_1, x_2, \ldots, x_n | c) = P(x_1|c) P(x_2|c) \cdots P(x_n|c)$$

$$C_{NB} = \arg\max_{c \in C} P(c) \prod_{x \in d} P(x|c)$$

MLE in Naïve Bayes Classifier

$$C_{NB} = \arg\max_{c \in C} P(c) \prod_{x \in d} P(x|c)$$

- For estimating the factors, we simply use the frequencies in the data.
- $\hat{P}(c_j) = \dfrac{doc\_count(C = c_j)}{N}$
- $\hat{P}(w_i|c_j) = \dfrac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$
  - $\hat{P}(w_i|c_j)$ is the fraction of times the word w_i appears among all words in documents of topic c_j

Smoothing

- Zero probabilities must be smoothed.
- $\forall i, \hat{P}(w_i|c_j) \neq 0$
- **Laplace (add-1) smoothing**

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$
$$= \frac{count(w_i, c_j) + 1}{\sum_{w \in V}(count(w, c_j) + 1)}$$
$$= \frac{count(w_i, c_j) + 1}{\sum_{w \in V} count(w, c_j) + |V|}$$

Naïve Bayes: Learning

1. From training corpus, build a vocabulary.
2. Calculate $\hat{P}(c_j)$ terms:
   - For each c_j in C Do
     - ★ $D_j \leftarrow$ all docs with class c_j
     - ★ $\hat{P}(c_j) \leftarrow |D_j|/N$
3. Calculate $\hat{P}(w_i|c_j)$ terms:
   - $T_j \leftarrow$ single doc containing all docs in D_j
   - For each word w_i in the vocabulary,
     - ★ $n_i \leftarrow$ # of occurrences of w_i in T_j
     - ★ $\hat{P}(w_i|c_j) \leftarrow (N_i + \alpha)/(n + \alpha|V|)$

NB Classifier — Example

$$\hat{P}(c) = \frac{N_c}{N}$$

$$\hat{P}(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$$

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Priors:

$P(c) = \frac{3}{4}$

$P(j) = \frac{1}{4}$

Choosing a class:

$P(c|d5) \propto 3/4 * (3/7)^3 * 1/14 * 1/14 \approx 0.0003$

Conditional Probabilities:

$P(\text{Chinese}|c) = (5+1) / (8+6) = 6/14 = 3/7$

$P(\text{Tokyo}|c) = (0+1) / (8+6) = 1/14$

$P(\text{Japan}|c) = (0+1) / (8+6) = 1/14$

$P(\text{Chinese}|j) = (1+1) / (3+6) = 2/9$

$P(j|d5) \propto 1/4 * (2/9)^3 * 2/9 * 2/9 \approx 0.0001$

Naïve Bayes Classifier – Analysis

- **Advantages**
  - fast and low storage requirements
  - works well with multi-class prediction problems
  - If the independence assumption holds, works better than other models with less training data
  - With many equally important features (e.g., categorical input variables), performs better in comparison to numerical variables
- **Disadvantages**
  - For "zero frequency" cases, a smoothing technique is required
  - Even though, it's a probability estimate, the outputs can't be directly used for the prediction probability.
  - It assumes that all the features are independent; conditional independence assumption is violated by real-world data.

Bias vs. Capacity

- Consider asking a botanist: Is an object a tree?
  - High capacity, Low bias
    - Botanist who memorizes
    - Will always say "no" to new object (e.g., different # of leaves)
  - Low capacity, high bias
    - Lazy botanist
    - Says "yes" if the object is green
  - We want the middle ground

kNN vs. Naïve Bayes

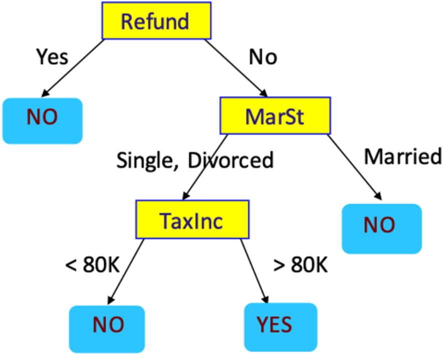
- Bias/Variance tradeoff (Variance $\sim$ Capacity)
- kNN has high variance and low bias
  - Infinite memory
- Rocchio/NB has low variance and high bias
  - Linear decision surface between classes

Summary: Representation of Text Categorization Attributes

- Representations of text are usually very high dimensional
  - "The curse of dimensionality"
- High-bias algorithms should generally work best in high-dimensional space
  - They prevent overfitting
  - They generalize more
- For most text categorization tasks, there are many relevant features & many irrelevant ones

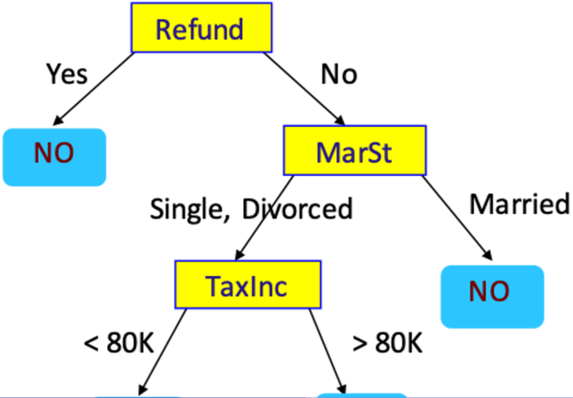
Decision Tree — Representation

- A tree structure
  - Each internal node: test one feature X_i
  - Each branch from a node: selects some value for X_i
  - Each leaf node: prediction for Y
- **Question 1: What function does a decision tree represent?**
  - C.f., In linear regression, we use a linear function of the input to predict the output
- **Question 2: Given a decision tree, how do we assign a label to a test point?**



cheating? under declared income?

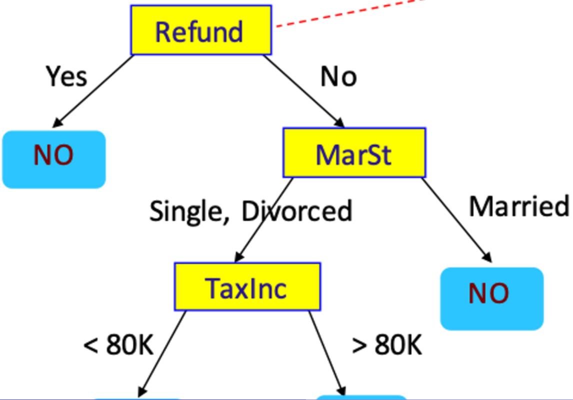
Decision Tree algorithm — Tax Evasion Detection



Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?

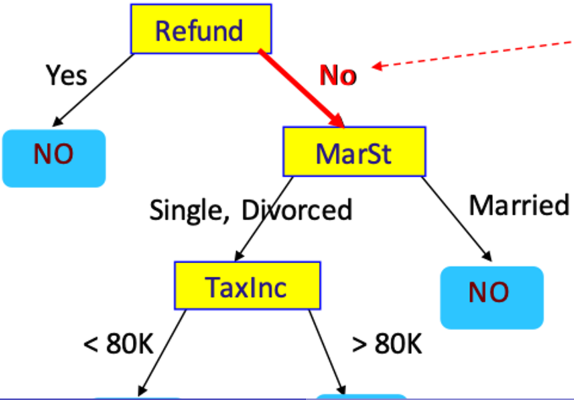
Decision Tree algorithm — Tax Evasion Detection



Query Data

X_1	X_2	X_3	Y
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Decision Tree algorithm — Tax Evasion Detection



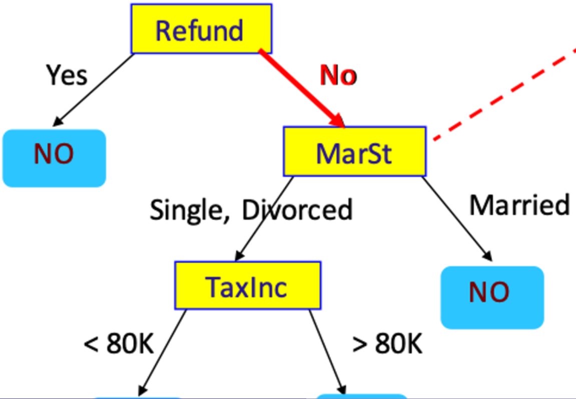
Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?

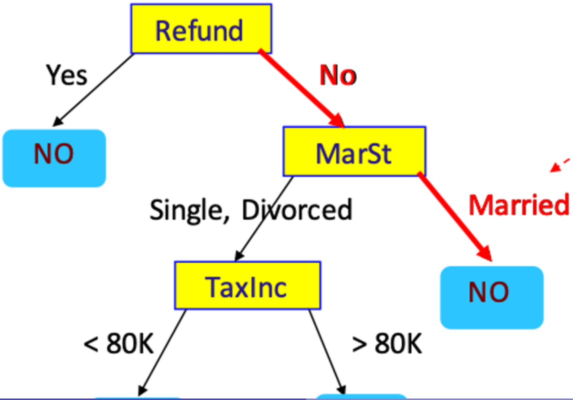
Decision Tree algorithm — Tax Evasion Detection

Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?



Decision Tree algorithm — Tax Evasion Detection



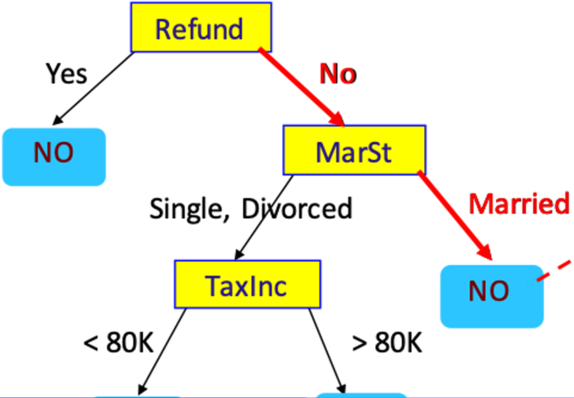
Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?

Decision Tree algorithm — Tax Evasion Detection

Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?



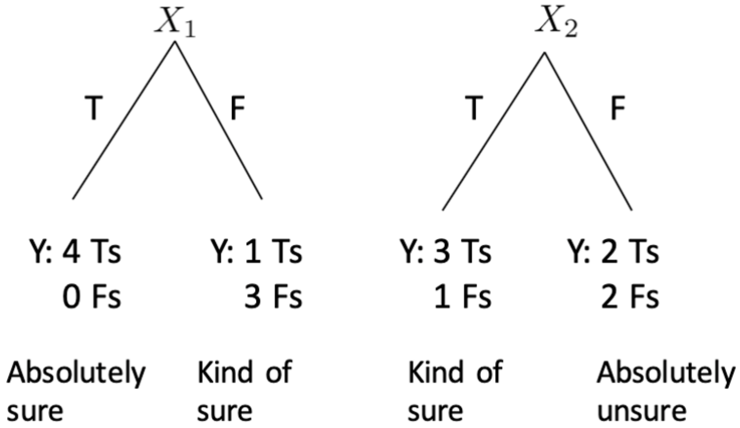
Assign Cheat to "No"

Decision Tree

- **So far...**
  - What function does a decision tree represent
  - Given a decision tree, how do we assign label to a test point
- **Now ...**
  - How do we learn a decision tree from training data?

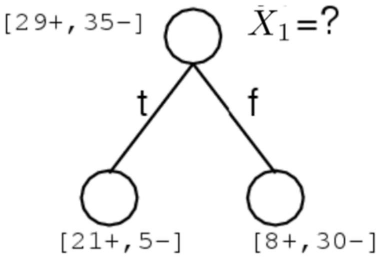
Which feature is better?

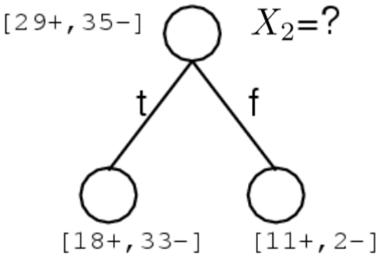
X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F



- Good split if we are more certain about classification after split.

Which feature is better, mathematically?





Pick the attribute/feature which yields **maximum information gain**:

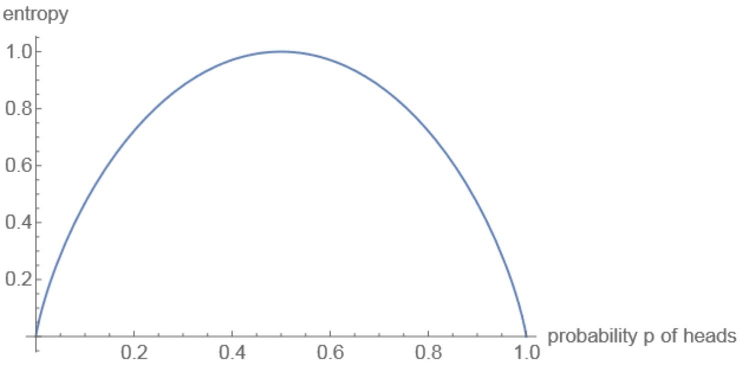
$$\arg\max_i I(Y, X_i) = \arg\max_i \left[H(Y) - H(Y|X_i) \right],$$

where $H(Y)$ is the entrtopy of Y and $H(Y|X_i)$ is the conditional entropy of Y given X.

Entropy

- Entropy of a random variable Y

$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x)$$



Information Theory-based interpretation: $H(Y)$ is the expected number of bits

Information Gain

- Advantage of an attribute means **decrease in uncertainty**.
  - Entroy of Y before split:

$$H(Y) = -\sum_{y} P(Y = y) \log_2 P(Y = y)$$

  - Entropy of Y aftrer splitting based on X_i:

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} P(x) H(Y|X = x) \\ &= -\sum_{x} P(x) P(Y|X = x) \log_2 P(Y|X = x) \end{aligned}$$

Maximum Information Gain

- Information Gain (I) measures the reduction in entropy (or surprise) by observing a feature to a given value of a random variable

$$I(Y, X_i) = H(Y) - H(Y|X_i)$$

- **Maximum Information Gain = Minimum Conditional Entropy**

$$\begin{aligned}
\arg\max_i I(Y, X_i) &= \arg\max_i \left[H(Y) - H(Y|X_i)\right] \\
&= \arg\min_i H(Y|X_i) \\
&= \arg\min_i P(Y = y|X_i = x) \log_2 P(Y = y|X_i = x)
\end{aligned}$$

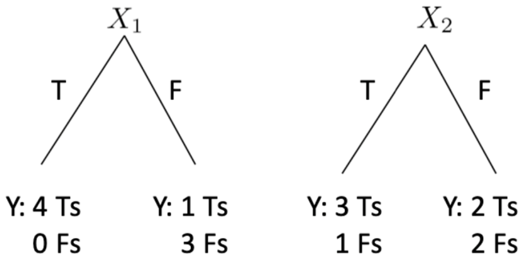
Which feature is best to split?

- Pick the attribute/feature which yields maximum information gain, which provides maximum information about Y.

Maximum Information Gain

$$H(Y \mid X_i) \; = \; -\sum_x P(X_i = x) \sum_y P(Y = y \mid X_i = x) \log_2 P(Y = y \mid X_i = x)$$

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F



$$\widehat{H}(Y|X_1) = \cancel{-\frac{1}{2}[1 \log_2 1 + 0 \log_2 0]} - \frac{1}{2}[\frac{1}{4} \log_2 \frac{1}{4} + \frac{3}{4} \log_2 \frac{3}{4}]$$

$$\widehat{H}(Y|X_2) = -\frac{1}{2}[\frac{3}{4} \log_2 \frac{3}{4} + \frac{1}{4} \log_2 \frac{1}{4}] \underbrace{- \frac{1}{2}[\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}]}_{\geq 0}$$

How to learn a decision tree

Recursive and greedy way to build a decision tree

1. Pick an attribute with the highest IG at an internal node.
2. Categorize data items based on the attribute values.
3. For each group:
    - if no examples – return majority from parent,
    - else if all examples in the same class – return the class,
    - otherwise loop to step 1 after removing the current feature.

ID3

Top-down induction (ID3, iterative Dichotomiser 3) ID3 is one of the DT methods (e.g., C4.5, C5, ...) Information gain is used to select the attributes.

1. $X \leftarrow$ the "best" decision feature for next node
2. Assign X as decision feature for node
3. For each value of X, create new descendant of node (Discrete features)
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, then stop, else iterate over new leaf nodes.
6. Repeat (steps 1-5) after removing current feature
7. When all features exhausted, assign majority label to the leaf node.

Decision Tree — Analysis

Decision Trees

- **Advantages**
  - Easy to understand (interpretable)
  - Easy to generate rules (intuitive)
  - Reduce problem complexity
  - Good with discrete attributes
  - Easily deals with missing values (just treat as another value)
  - Fast at test time
- **Disadvantages**
  - Few hyperparameters. (this can be an advantage too)
  - A document is only connected with one branch (hard clustering)
  - Once a mistake is made at a higher level, any subtree is wrong
  - Does not handle continuous variable well
  - Too big of a tree may suffer from overfitting.

Decision Tree – Summary

- Can be used for classification, regression and density estimation too.
- The overfitting problem:
  - must use tricks to find “simple trees”, e.g.,
    - Pre-pruning: fixed depth/fixed number of leaves
    - post-pruning: Chi-square test of independence
    - Complexity penalized / MDL (minimum description length) model selection
- Decision trees $\rightarrow$ Random Forests $\rightarrow$ Gradient-boosted Decision Trees $\rightarrow \cdots$
- In practice, an ensemble model is used.

Wrap Up

- Naïve Bayes
  - fast and robust to irrelevant features
  - very good in domains with many equally important features
  - A good dependable baseline for text classification
- Decision trees
  - Simple non-linear, discriminative classifier
  - Easy to interpret
- In real-world
  - You should exploit domain specific structure!!

Summary

- and discussion

Reference