Text Classification (I)

Text and Web Mining (H6751)

School of Communication and Information

Classification Example

Classification Example

How Machines Learn?

Features Extraction

1) Length of ear
2) Length of face
3) Length of mane

.....

How Machines Learn?

Data variations

Patterns in data

Model training

Recognizing that Documents Fit a Pattern

- To be successful in prediction, we expect to find common characteristics - **patterns of words in documents**.
- The spreadsheet of Fig. 3.4 is **an idealized view of words that form a pattern**.
- We see that **the same two words (Word2 and Word4)** always occur for **class 1** and never occur together for class 0.
- A pattern is formed when a combination of words occurs for the class of interest and not for the negative class.

word1	word2	word3	word4	word5	...	wordN	label
0	1	1	1	0	...	1	1
1	0	1	0	0	...	1	0
1	1	0	1	1	...	0	1
0	1	0	0	1	...	1	0
1	1	1	1	0	...	1	1
0	1	0	1	1	...	0	1
1	0	1	1	0	...	1	0
0	1	1	1	1	...	0	1

Fig. 3.4 Predictive patterns in a spreadsheet

Recognizing that Documents Fit a Pattern

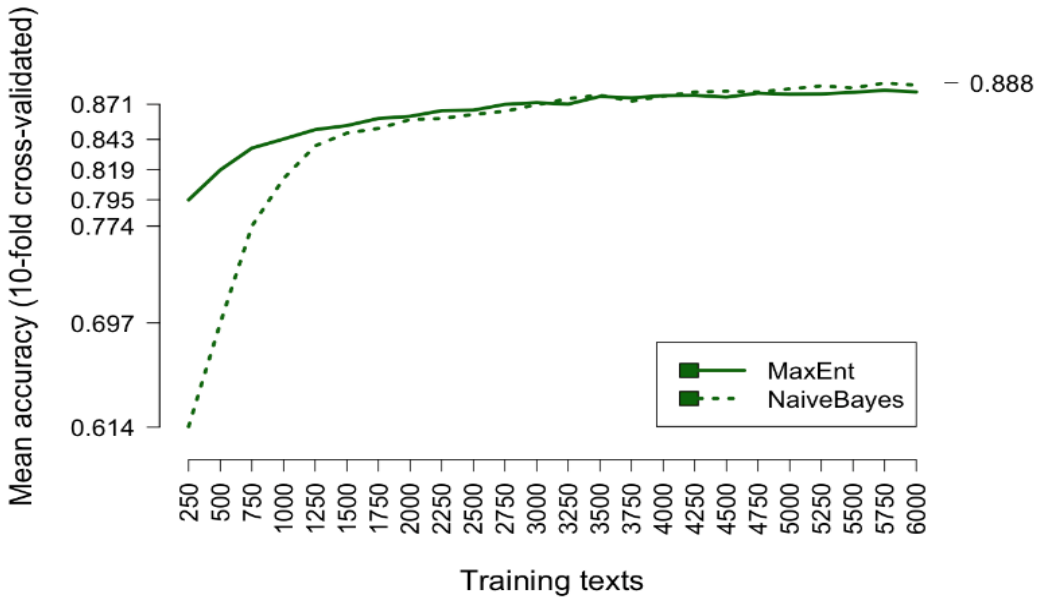
- In contrast to Fig. 3.4, in Fig. 3.5, we see **no obvious predictors** for **class 1**.
  - When both Word2 and word4 are 0, label is either 0 or 1.
  - When Word2 is 0 and word4 is 1, label is either 0 or 1.
- No simple pattern is found that separates the two classes.
  - Some categories are quite challenging: e.g., ***user review*** vs. ***opinion spam*** (a biased review written by a person sponsored by a company).

word1	word2	word3	word4	word5	...	wordN	label
0	1	0	1	0	...	1	1
1	0	1	0	0	...	0	0
1	0	0	1	1	...	0	1
0	1	0	0	1	...	1	0
1	0	0	1	0	...	1	1
0	1	0	0	1	...	0	1
0	0	1	1	0	...	0	0
0	0	1	0	1	...	0	1

Fig. 3.5 Spreadsheet with no obvious patterns

How Many Documents Are Enough?

- How many cases are enough to learn from the sample?
  - Generally, we cannot directly answer that question.
  - Instead, we depend on correct evaluation of proposed solutions, giving us a good idea of their future performance.
  - We are particularly concerned in the early stages of learning from a small sample, anticipating a need for more documents.



How Many Documents Are Enough?

- Although it is **possible to assemble a collection of many thousands of documents**, the sample of documents for building predictors must come with **labels**.
- This initial **assignment of labels** is likely to require **human intervention**, and the effort may be **a time-consuming process**.
  - The same dataset should be tagged by at least two persons, and measure **inter-coder reliability**, such as Agreement Rate and Cohen's Kappa score.
- We may think of text mining as a completely automated process, but, in reality, **the assignment of labels is a bottleneck**.

Cohen's Kappa Scoring

Formula to calculate Cohen's kappa for two coders: $\kappa = \frac{p_o - p_e}{1 - p_e}$

Po = the relative **observed agreement among coders**.
Pe = the hypothetical **probability of chance agreement**.

For example, 2 coders rated **50** documents with Positive or Negative sentiment labels.

20 documents were rated **Positive** by both, **15** documents were rated **Negative** by both

Overall, coder A labelled **25** documents Positive and **25** documents Negative.

Overall, coder B labelled **30** documents Positive and **20** documents Negative

Po = number in **agreement** / total = (**20** + **15**) / 50 = 0.70

Probability of coders both labelling Positive randomly = (25/50) * (30/50) = 0.30

Probability of coders both labelling Negative randomly = (25/50) * (20/50) = 0.20

Probability of coders chance agreement = 0.30 + 0.20

κ = (Po – Pe) / (1 – Pe) = (0.70 – 0.50) / (1 – 0.50)
= 0.40 (fair agreement)

(0.41 to 0.60) = moderate agreement
(0.61 to 0.80) = substantial agreement

Text Categorization

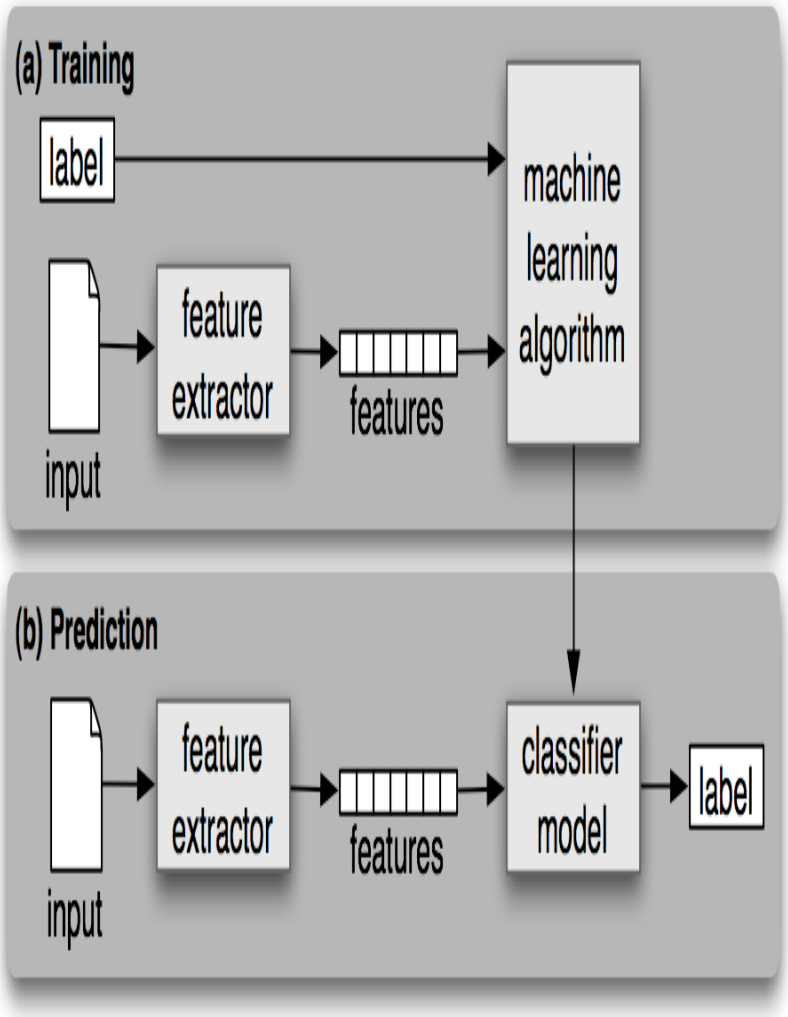
- The classical prediction problem for text is called ***text categorization*** *or* ***text classification.***
- Text or document classification is defined as the process of assigning text documents into one or more classes or categories, assuming that we have a predefined set of classes.
- A text classification system can successfully classify each document to its correct class(es) based on inherent properties of the document.
- For example, we can collect newswire articles and describe a set of fixed topics such as ***financial or sports stories.*** When news arrives, the words are examined, and articles are assigned topics from a fixed list of possible topics.

Text Categorization Examples

- Assigning subject categories, topics, or genres
  - In Library catalogues, library congress of subject headings (LCSH)
  - Genres: Romance novel vs. Detective fiction
- Spam detection
- Authorship identification
- Age/gender identification
- Language Identification
  - Similar languages use the same alphabet and additional characters (or different usage of alphabet).
  - E.g., English, German, French, Spanish, Portuguese, etc.
- Sentiment analysis, etc.

Classification Tasks - Overview

- In classification tasks the application learn to **predict discrete values** for the **response variables** from one or more **explanatory variables** (**features**).
- The model is **trained** according to extracted features using labelled data.
- Prediction occurs through **extracting features** from the inputs and using the classifier to label the class.



Unsupervised / Supervised Machine Learning

- ***Unsupervised Learning*** - the training of machine using information that is neither classified nor labelled. The task of machine is to group information according to similarities, patterns and differences without any prior training of data (e.g., Document categorization using clustering).

- ***Supervised Learning*** - machine learning techniques or algorithms that are trained on prelabelled data samples.
- Features are extracted from this data using feature engineering and each data point has its own feature set and corresponding class/label.
  - *Input:*
    - a document d
    - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
    - a training set of m hand-labeled documents $(d_1, c_1), \ldots, (d_m, c_m)$
  - *Output:*
    - a learned classifier $\gamma: d \rightarrow c$

Classification Methods:

- Hand-coded Rules
- Machine Learning Methods:
  - Naïve Bayes
  - Logistic Regression
  - Support Vector Machines
  - Decision Trees

Hand-coded rules

- Rules based on combinations of **words or other features**
  - **Spam detection**: black-list-address OR ("dollars" AND "have been selected")
- Accuracy can be high.
  - If rules are carefully refined by expert.
- But building and maintaining these rules is expensive.
- Possible overfitting – model only works on the specific dataset.

Naïve Bayes Classifier

- A probabilistic framework for solving classification problems.
- We denote the probability of the occurrence of C *(class)* given the attributes *(A_i) (features)* as follows:
- Conditional Probability: $P(C|A) = \frac{P(A,C)}{P(A)}$ $P(A|C) = \frac{P(A,C)}{P(C)}$
- Bayes Theorem: $P(C|A) = \frac{P(A|C)P(C)}{P(A)}$ $P(A,C) = P(A|C)P(C)$
- Given a record with attributes ($A_1, A_2, \ldots, A_n$)
  - Goal is to predict class C

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C)P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - To find value of C that maximizes P(C| $A_1, A_2, \ldots, A_n$) => equivalent to choosing value of C that maximizes P($A_1, A_2, \ldots, A_n$|C)
- How to estimate P($A_1, A_2, \ldots, A_n$|C)?

Naïve Bayes Classifier

- Assume independence among attributes A_i when class C is given:

$$P(A_1, A_2, \ldots, A_n \mid C) = P(A_1 \mid C_j)\, P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$$

- Estimate $P(A_i \mid C_j)$ for all A_i and C_j

# of class c instances/total instances

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Class: $P(C) = N_c/N$

e.g., $P(\text{No}) = 7/10$, $P(\text{Yes}) = 3/10$

For discrete attributes:

$$P(A_i \mid C_k) = |A_{ik}| / N_{Ck}$$

- Where $|A_{ik}|$ is number of instances having attribute A_i and belongs to C_k
- N_{Ck} is total count of all instances belonging to class C
- E.g.,

$P(\text{Status}=\text{Married} \mid \text{No}) = 4/7$

$P(\text{Refund}=\text{Yes} \mid \text{Yes}) = 0$

Classify as $P(C_j)$ if $P(C_j)\prod P(A_i \mid C_j)$ has highest value amongst the Classes

Applying Naive Bayes to Text Classification

- For a document *d* and a class *c*

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

posterior belief P(c | *d*) is calculated by multiplying the *prior* belief P(c) by the *likelihood* P(d | *c*) that d will occur if c is true.

E.g., "I love this movie" $P(c|d) = argmax\ P(cj) \prod_{i \epsilon positions} P(di|cj)$

Get largest value of function

$P(c_j)$ $P(d_i|c_j)$ $\overline{d_i}$ => term vector of document

E.g., P(pos) * (P(I | pos) * P(love | pos) * P(this | pos) * P(movie | pos)) = 0.7
P(neg) * (P(I | neg) * P(love | neg) * P(this | neg) * P(movie | neg)) = 0.4

Worked example

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

Priors:

$P(c) = 3/4$

$P(j) = 1/4$

Vocabulary Size $|V| = 6$

	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	?

Laplace Smoothing – cater for new words not found in data.

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

Prevent prob of new words =0

Count of total words in Class

c **Conditional Probabilities:**

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

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

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

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

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

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

$$c_{MAP} = \text{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i|c)$$

Choosing a class:

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

≈ 0.0001

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

≈ 0.0003

"Tokyo", "Japan" only appears in Class j, hence higher weightage to j even though more "Chinese" term in document 5

Naive Bayes is Not So Naive

- Very Fast, low storage requirements
  - Requires almost no memory and little computation, so it does have its advocates.
- It usually works best with **a relatively small dictionary** representing the key words needed to make a decision for that class.
- Optimal if the independence assumptions hold:
  - However, **assumption of independent features may not be true** for certain word features (e.g., bigram features, POS tagging of words)
- A **good dependable baseline** for text classification

Logistic Regression

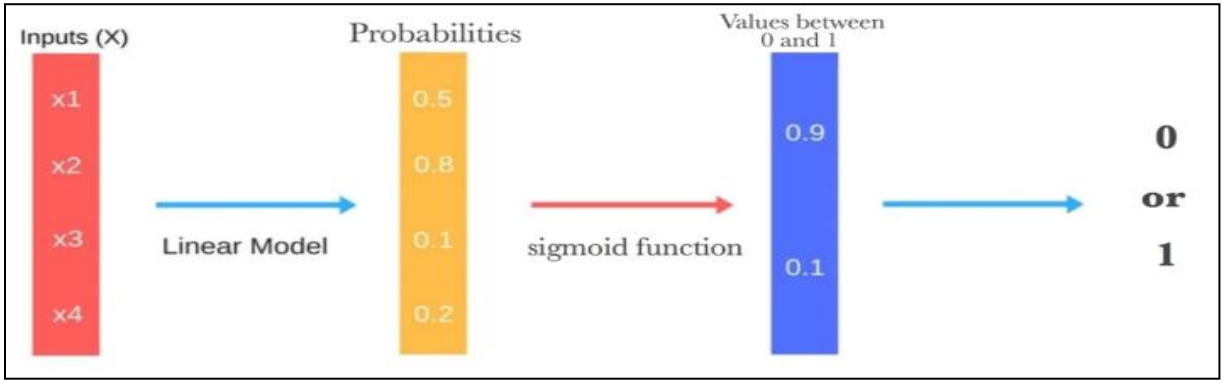
- In the logistic (sigmoidal) model, the log-odds (logarithm of odds) for the class/category label are the equation of the linear regression model (linear combination of one or more independent features).

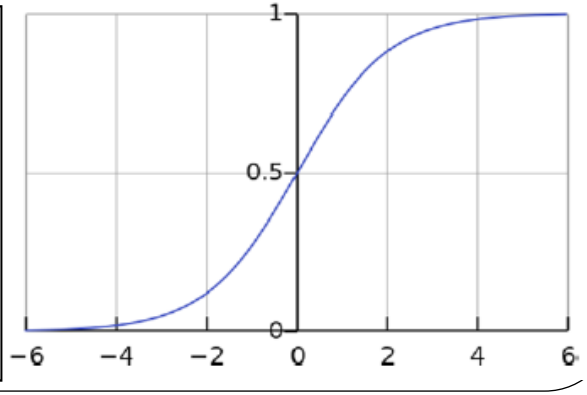
$$p\,(probability\ of\ class) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

- However, we need to predict discrete classes or categories. Corresponding probability of the class labels can vary between 0 and 1, depicting the confidence of the prediction. The function that converts the log-odds to probability is the logistic function:

$$\frac{1}{1 + e^{-x}}$$

where *e* is the exponent and x indicates the equation





Logistic Regression

- Consider a standard multiple linear regression model, depicted as follows: $p\,(probability\ of\ class) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$

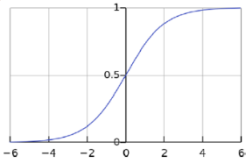
  Such that $\{x1, x2, \ldots, xn\}$ are our features and we are trying to estimate the coefficients, $\{\beta 1, \beta 2, \ldots, \beta n\}$.

- if p is the probability of predicting a specific class, the odds of that is $p/(1-p)$, which is basically the ratio of the favorable outcomes to the unfavorable outcomes.

$$logit(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

- If we want to get to the class probability values that the logistic regression model outputs for us, we can derive the following equation, which is the heart of the logistic regression model:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}}$$

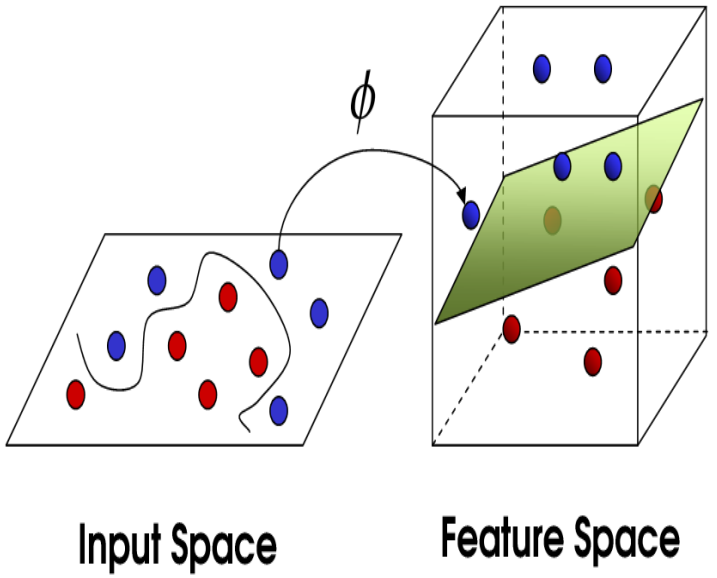


use Maximum Likelihood Estimation to optimize and estimate the coefficients for each feature

Support Vector Machines

- Support vector machines, known popularly as SVMs, are supervised learning algorithms.

- Training data samples are represented as points in space such that points belonging to either class can be separated by a wide gap between them (hyperplane) and the new data points to be predicted are assigned classes based on which side of this hyperplane they fall into.

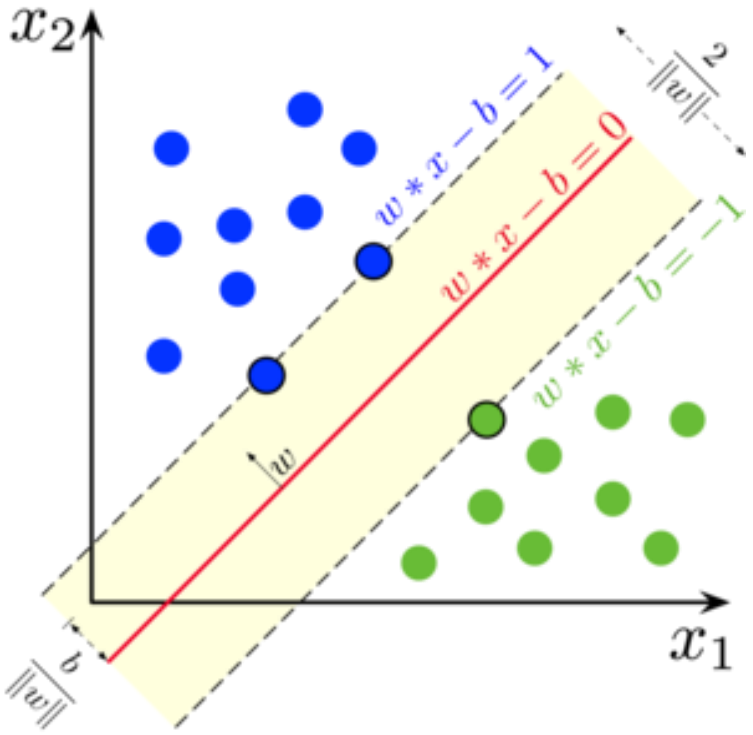


Support Vector Machines

- The SVM algorithm finds **the max-margin** hyperplane which separates the set of data points having **class label** $yi = 1$ from those having **class label** $yi = -1$ so that the distance between the **hyperplane** and **sample data points** from either class nearest to it is **maximized**. These sample data points are known as the **support vectors**.

- The hyperplane is defined as the set of points $\vec{x}$ which satisfy $(\vec{w}.\vec{x} - b = 0)$, where $\vec{w}$ is the normal vector to the hyperplane and $\frac{b}{||\vec{w}||}$ gives the offset of the hyperplane from the origin to the support vectors.

To increase the margin => minimize the magnitude of w = $||\vec{w}||$



Decision Trees

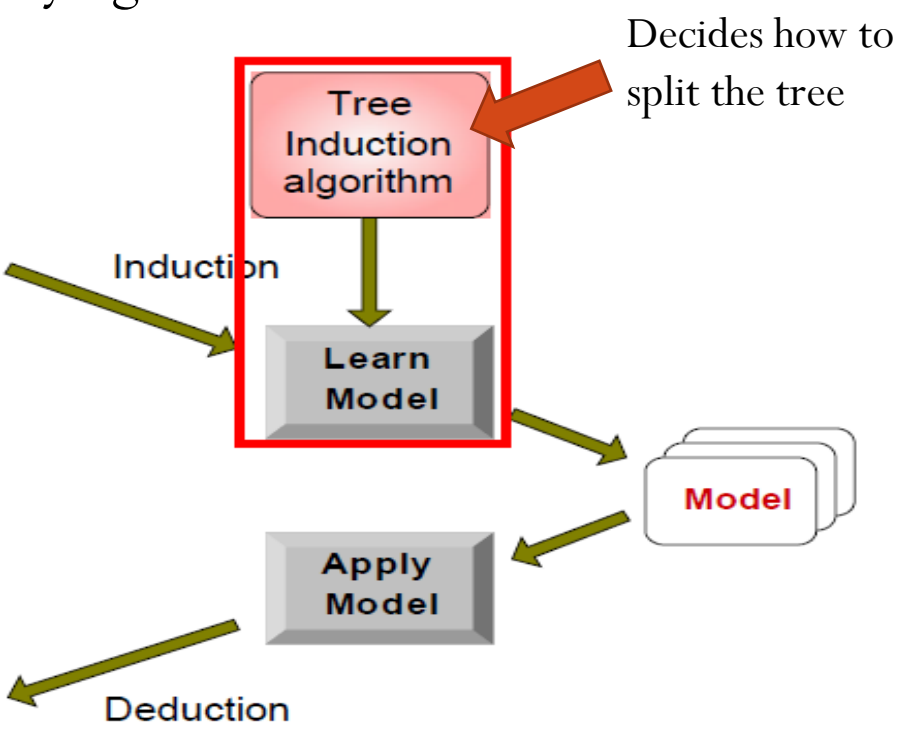
- Decision trees are a family of supervised machine learning algorithms that can represent and interpret sets of rules automatically from the underlying data.

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Decision Trees – Hunt's Algorithm

- Hunt's Algorithm – one of the Tree Induction Algorithm

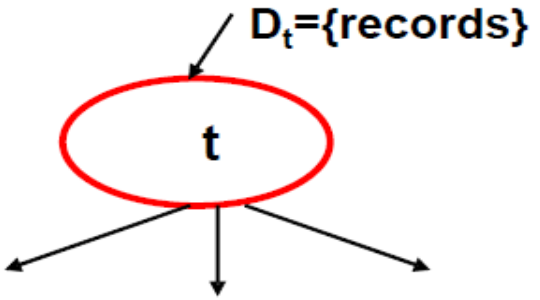
D_t = {training records @ node t}

- **If D_t = {records from different classes}**
  - **Split D_t into smaller subsets via attribute test**
  - **Traverse each subset with same rules**
- **If D_t = {records from single class y_t}**
  - **Set Node t = leaf node with class label y_t**
- **If D_t = {} (empty)**
  - **Set Node t = leaf node with default class label y_d**

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Recursively apply above criterion until ...

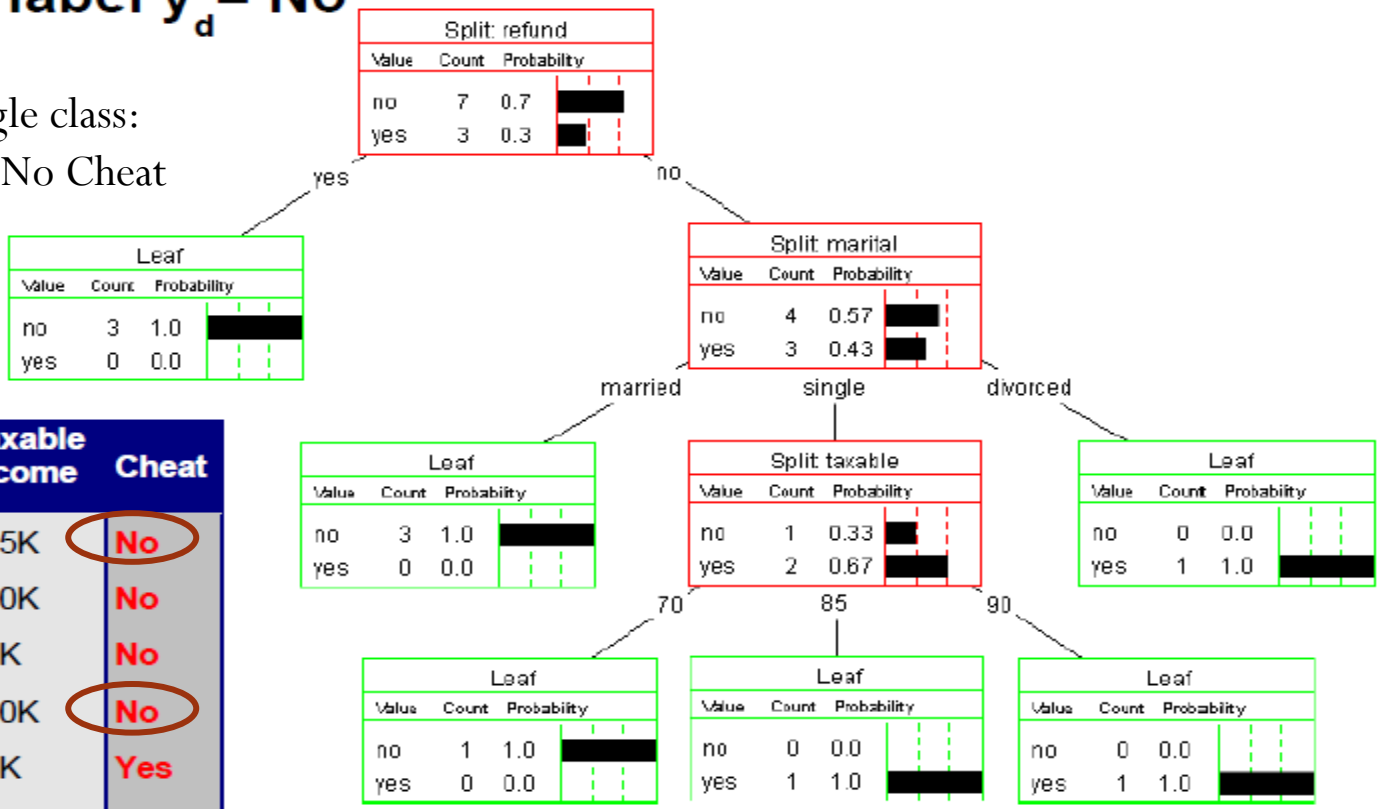
- **No more training records left**



Decision Trees

- **Default class label y_d = No**

All records from single class:
=> All Yes refund = No Cheat



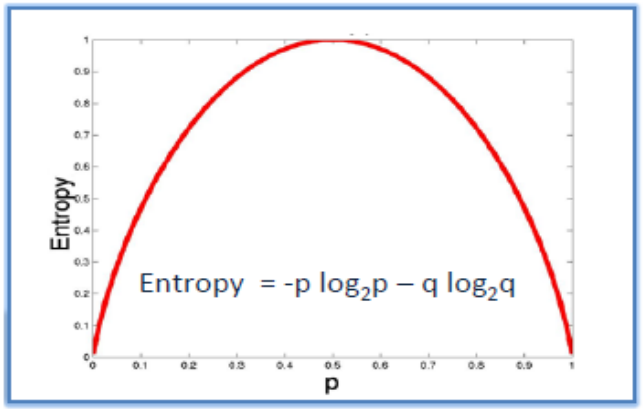
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Decision Tress – Information Gain

- Entropy – the measure of impurity, disorder or **uncertainty** in data.
  - Controls how a Decision Tree decides to split the data.

$$Entropy\ at\ node\ t = -\sum p(i|t) log_2 p(i|t)$$

  - $p(i|t)$ = relative frequency of class i at node t
  - If the data is completely homogeneous entropy is zero and if the data is equally divided the entropy is one.



Entropy = -0.5 $\log_2$0.5 – 0.5 $\log_2$0.5 = 1

- Information Gain (IG) – measures how much "information" a feature gives about the class (higher the better).

$$Information\ Gain = Entropy(parent) - [Weighted\ Average] * Entropy(children)$$

(High) **(Low)**

Decision Trees – How to split the data?

- Computing Entropy

$$Entropy\ at\ node\ t = -\sum p(i|t) log_2 p(i|t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

Homogeneous = Pure => zero uncertainty

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

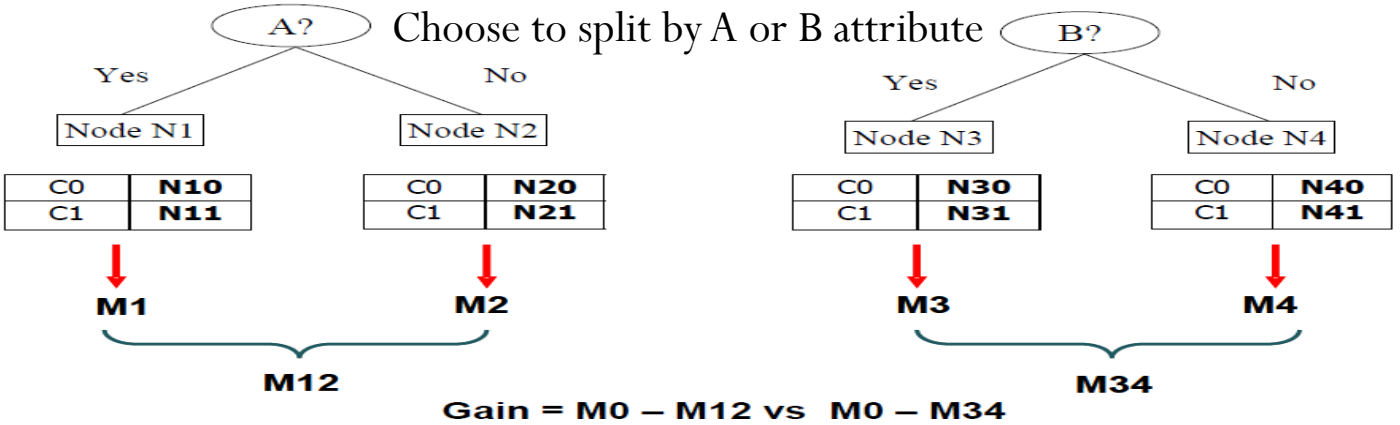
P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

More equally divided => high uncertainty

- Computing Information Gain
  - **Before Split**
    - **Purity @ Node N0 = M0**
  - **After Split (2 choices)**

C0	**N00**
C1	**N01**



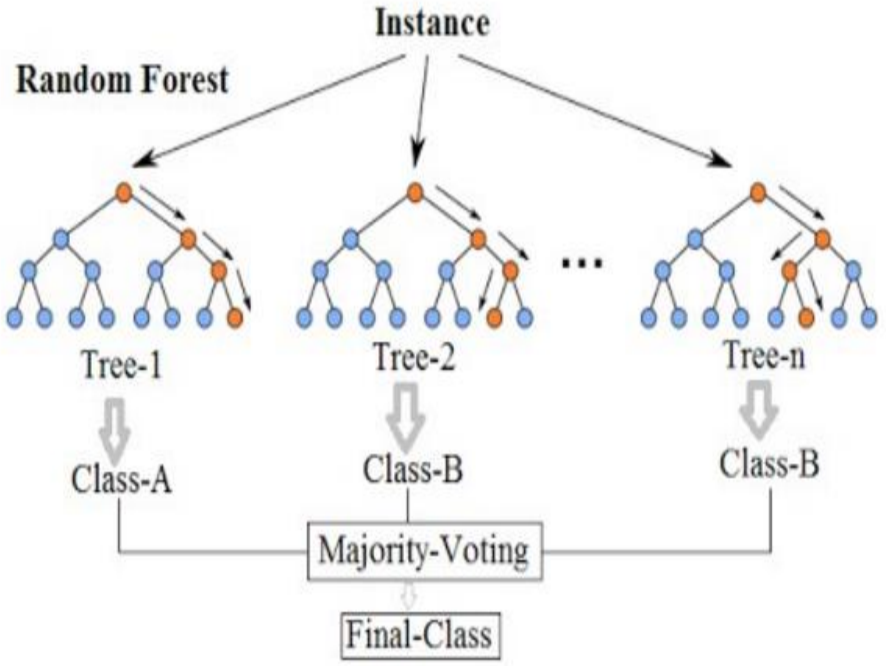
Gain = M0 – M12 vs M0 – M34

Decision Trees algorithm will choose the split that maximize Information Gain => reduces the most uncertainty

Decision Trees – Random Forest

- A major drawback of decision trees is that they are prone to **over-fitting** - the more data there is, greater the depth of the tree. The model might work really well on training data, but instead of learning, it just memorizes all the training samples and builds very specific rules to them.

- Random forest solves the problem by **fitting a number of decision tree classifiers on various sub-samples of the dataset and uses averaging** to improve the predictive accuracy and control over-fitting.



Evaluation of Performance

- The 2-by-2 Contingency Table (Confusion Matrix)

	Truth: Positive Class	Truth: Negative Class
Positive Class Predicted	True Positive (TP) (number of instances of positive class correctly predicted)	False Positive (FP) (number of instances of negative class incorrectly predicted as positive class)
Negative Class Predicted	False Negative (FN) (number of instances of positive class incorrectly predicted as negative class)	True Negative (TN) (number of instances of negative class correctly predicted)

- E.g., spam email classification

	Truth: spam	Truth: not-spam
Predict: spam	10	10
Predict: not-spam	10	970

Performance Metrics

- **Precision**: % of selected items that are correct

  Precision = tp / (tp + fp)

  $$\text{Precision} = \frac{number\ of\ correct\ positive\ predictions}{number\ of\ positive\ predictions}$$

- **Recall**: % of correct items that are selected

  Recall = tp / (tp + fn)

  $$\text{Recall} = \frac{number\ of\ correct\ positive\ predictions}{number\ of\ positive\ class\ documents}$$

	correct	not correct
selected	tp	fp
not selected	fn	tn

	Truth: spam	Truth: not-spam	Class precision
Predict: spam	741	200	**78.75%**
Predict: not-spam (ham)	6	4627	99.87%
Class recall	**99.20%**	95.86%	

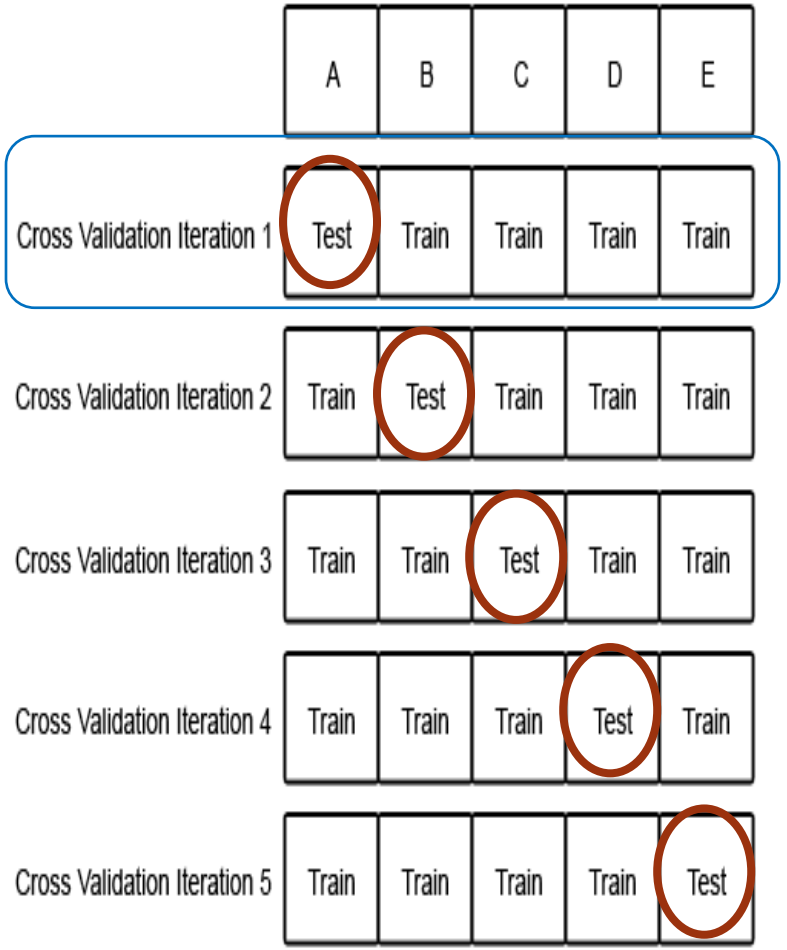
Performance Metrics

- Accuracy – the **overall** proportion of **correct** predictions of the model
  - Accuracy $= \frac{TP+TN}{TP+FP+TN+FN}$

- A combined measure that assesses a trade-off between Precision and Recall is **F measure** (weighted harmonic mean, a.k.a. **F1 Score**)
  - F1-Score $= \frac{2 * precison * recall}{precision+recall}$

Cross validation

- When training data is **scarce**, a practice called **cross-validation** can be used to train and validate an algorithm on the same data.
- In cross-validation, the training data is **partitioned**. Training is performed on all but one of the partitions, and tested on the remaining (one) partition.
- The partitions are then rotated several times so that the algorithm is trained and evaluated on all of the data.
- The following diagram depicts cross-validation with **five partitions** or **folds.**



Overall performance => take average

Referenced Materials

- NLP, Dan Jurafsky and Christopher Manning, http://www.stanford.edu/~jurafsky/NLPCourseraSlides.html
- Fundamentals of Predictive Text Mining, Sholom M. Weiss, Nitin Indurkhya, and Tong Zhang, Springer.
  - Chapter 3