UNIVERSITÀ DEGLI STUDI DI BARI
Facoltà di Scienze Matematiche, Fisiche e Naturali
Dipartimento di Informatica

Text Categorization

Corso di
Metodi per il Ritrovamento dell'Informazione

Credits

- Prof. Mooney, Professor of Computer Science, University of Texas, Austin
- Baeza-Yates and Riberiro-Neto
- Marco de Gemmis
- Giovanni Semeraro
- Pasquale Lops

Outline

- Problem Definition: Categorization
- Machine Learning (ML) e Text Categorization (TC)
- Indexing
- Building a Classifier
- Evaluation
- Applications of ML & TC (& IR)
- References

Il task della classificazione

- Dati:
  - La descrizione di una istanza, $x \in X$, dove X è *lo spazio delle istanze*
  - Un insieme finito di categorie: $C=\{c_1, c_2,\ldots c_n\}$
- Determina:
  - La categoria di x: $c(x) \in C$, dove $c(x)$ è una funzione di classificazione il cui dominio è X e il cui range è C.

Machine Learning for Categorization

- Un esempio di training è un'istanza $x \in X$, associata alla categoria di appartenenza $c(x)$: $<x, c(x)>$ dove *c è una funzione di classificazione non nota*
- Dato un insieme di esempi di training, *D*
- Trova una ipotesi di funzione di classificazione, $h(x)$, tale che:

$$\forall < x, c(x) > \in D : h(x) = c(x)$$

Consistenza

Sample Category Learning Problem

- Spazio delle istanze: <size, color, shape>
  - ✓ size ∈ {small, medium, large}
  - ✓ color ∈ {red, blue, green}
  - ✓ shape ∈ {square, circle, triangle}
- *C* = {positive, negative}
- *D*:

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

Generalizzazione

- L’obiettivo è quello di generalizzare le osservazioni (esempi) al fine di classificare correttamente istanze non appartenenti al training set.
- Memorizzare semplicemente gli esempi garantisce la consistenza ma non generalizza!
- Il processo di appendimento mira a definire un modello generale di classificazione

Text Categorization

- Il termine Text Categorization designa l'assegnazione automatica di una o più categorie a testi scritti in linguaggio naturale
- Applicazioni:
  - ✓ Web pages
    - Recommending
    - Yahoo-like classification
  - ✓ Newsgroup Messages
    - Recommending
    - spam filtering
  - ✓ News articles
    - Personalized newspaper
  - ✓ Email messages
    - Folderizing
    - spam filtering

Definizione del problema (1)

- Formalmente il problema di TC consiste nell'approssimare la funzione target:
  - ✓ ϕ'**: D×C→{T,F}** attraverso la funzione
  - ✓ ϕ**: D×C→{T,F}**
- ϕ si dice classificatore e deve risultare il più fedele possibile a ϕ'
- D è l'insieme dei documenti
- C è l'insieme delle categorie

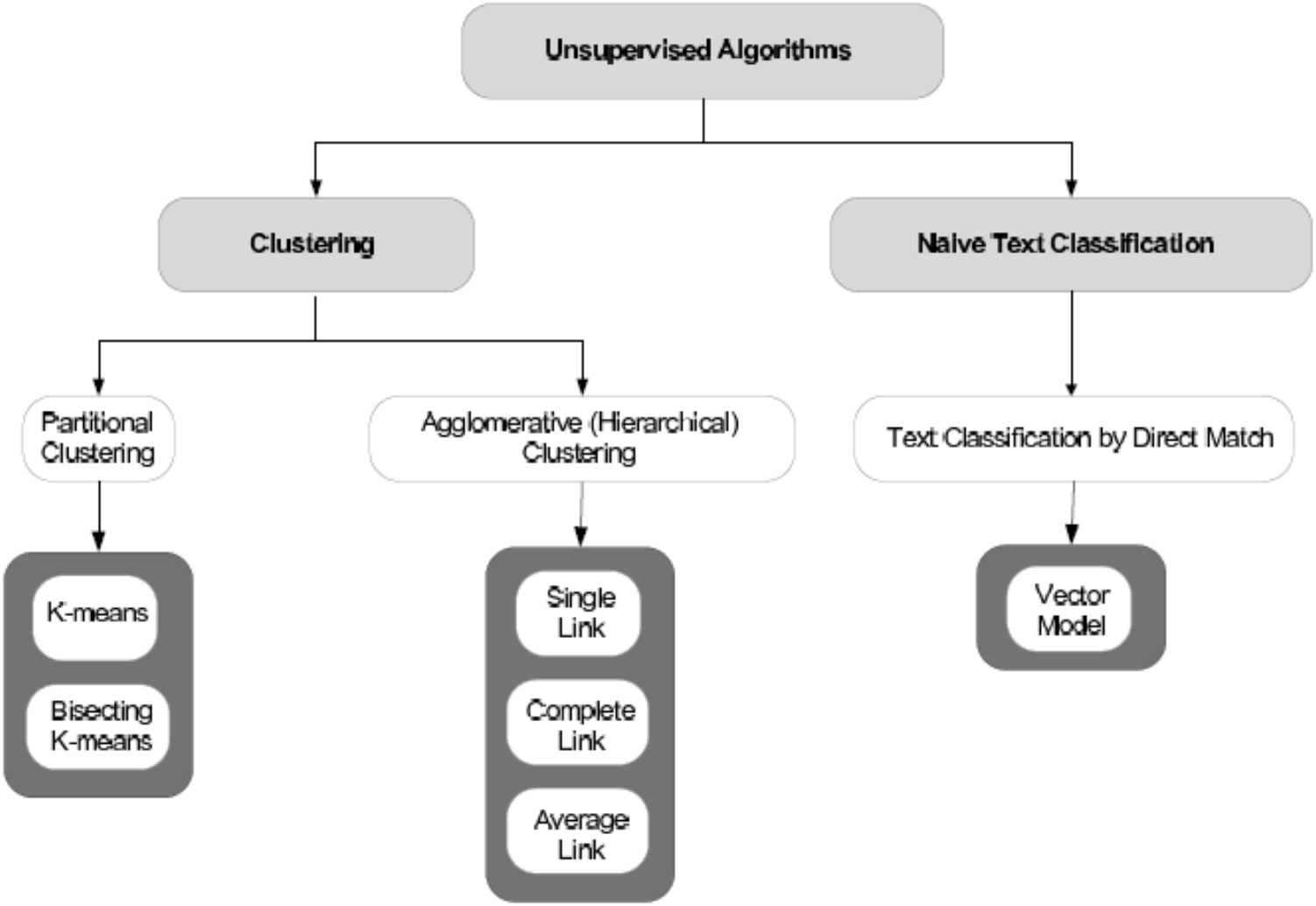
Definizione del problema (2)

- **Single-label**: una singola classe assegnata ad ogni documento
  - ✓ Binario
- **Multi-label**: è possibile assegnare una o più classi ad ogni documento
- Il single label è possibile solo nel caso vi sia indipendenza tra le categorie (solitamente vero)
- Utilizzando il single label binario, classificare sotto C vuol dire risolvere esattamente $|C|$ volte il problema di categorizzazione single-label binario
- Il caso SL binario è
  - ✓ Più generale
  - ✓ Usato molto frequentemente
  - ✓ Molto dettagliato in letteratura

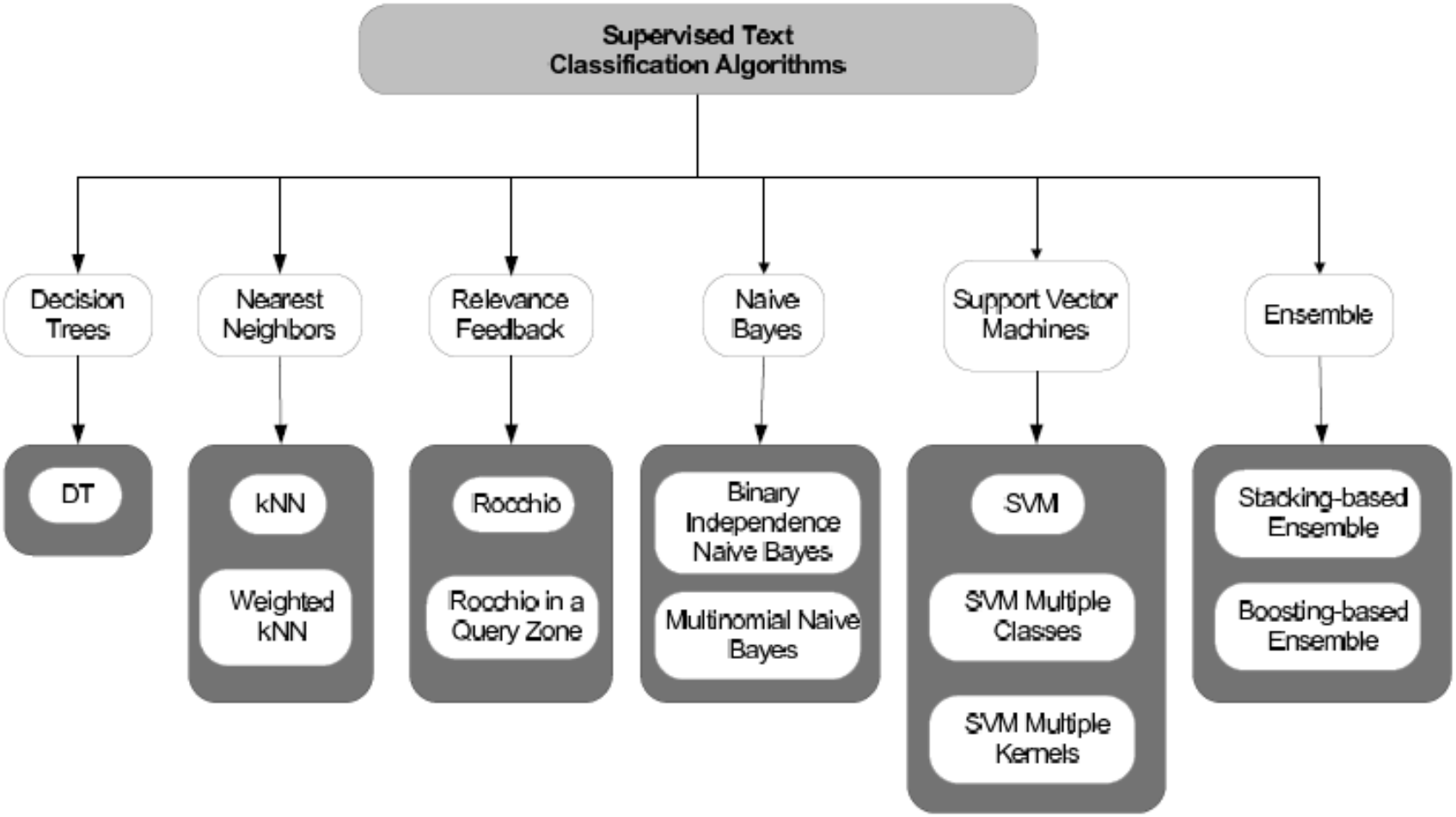
Text Classification Algorithms

- **Supervised learning**
  - ✓ Training data provided as input
  - ✓ Training data: classes for input documents
- **Unsupervised learning**
  - ✓ No training data is provided
- **Semi-supervised learning**
  - ✓ Small training data
  - ✓ Combined with larger amount of unlabeled data

Text Classification Algorithms



Text Classification Algorithms



Train and Test

- Sia dato $D=\{d_1,\ldots,d_{|D|}\}$ e $C=\{c_1,\ldots,c_{|C|}\}$
- La funzione ϕ' è nota per ogni valore di $<d_j,c_i> \in D \times C$
  - ✓ Se il suo valore è True allora si dice che $<d_j,c_i>$ è un esempio positivo per ϕ', in caso contrario si dice negativo
- Definiamo due sotto-insiemi di D:
  - ✓ Tr (Training Set)
  - ✓ Te (Test Set)
  - ✓ $D = Tr \cup Te$
- Il classificatore è costruito a partire dagli esempi di Tr
- Il testing (validazione) viene effettuato usando gli esempi di Te

Train and Test

- Supervised algorithms depend on a **training set**
  - ✓ set of classes with examples of documents for each class
  - ✓ examples determined by human specialists
  - ✓ training set used to **learn** a classification function
- The larger the number of training examples, the better is the fine tuning of the classifier
  - ✓ **Overfitting**: classifier becomes specific to the training examples
- To evaluate the classifier
  - ✓ Use a set of **unseen** objects
  - ✓ Commonly referred to as **test set**

Machine Learning e TC

- Learning Algorithms:
  - ✓ **Bayesian (naïve)**
  - ✓ Neural network
  - ✓ **Relevance Feedback (Rocchio)**
  - ✓ Rule based (Ripper)
  - ✓ **Nearest Neighbor (case based)**
  - ✓ Support Vector Machines (SVM)
- Costruzione induttiva di un classificatore che svolge le funzioni di un sistema esperto
- Vantaggi: buona manutenibilità (non è necessario ingegnerizzare di continuo le regole di classificazione)
- Svantaggi: necessita base di documenti pre-classificati che permettano la costruzione del classificatore

Indicizzazione

- Il classificatore deve ricevere una codifica compatta del testo da classificare, perciò usiamo tecniche di indicizzazione
- Solitamente un documento d_j viene rappresentato come un insieme di pesi, che indicano la presenza (o la frequenza) di termini tratti da un insieme di parole (o frasi)
- La scelta tipica per tale insieme è il *bag of words*
- In alternativa si possono usare frasi denotate
  - ✓ Sintatticamente (frasi del linguaggio)
  - ✓ Statisticamente (insiemi di parole)
- Solitamente i pesi hanno valori in [0,1] oppure assumono i soli valori 0 e 1 (caso binario)

Indicizzazione (2)

- La formula standard per il calcolo dei pesi è:

$$tfidf(t_k, d_j) = \#(t_k, d_j) \cdot \log \frac{|Tr|}{\#_{Tr}(t_k)}$$

- Per far ricadere i pesi in [0,1] bisogna usare la *normalizzazione del coseno*:

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|\mathcal{T}|}(tfidf(t_s, d_j))^2}}$$

  dove *T* è l'insieme dei termini (*features*) che compaiono almeno una volta in nei documenti di *Tr*

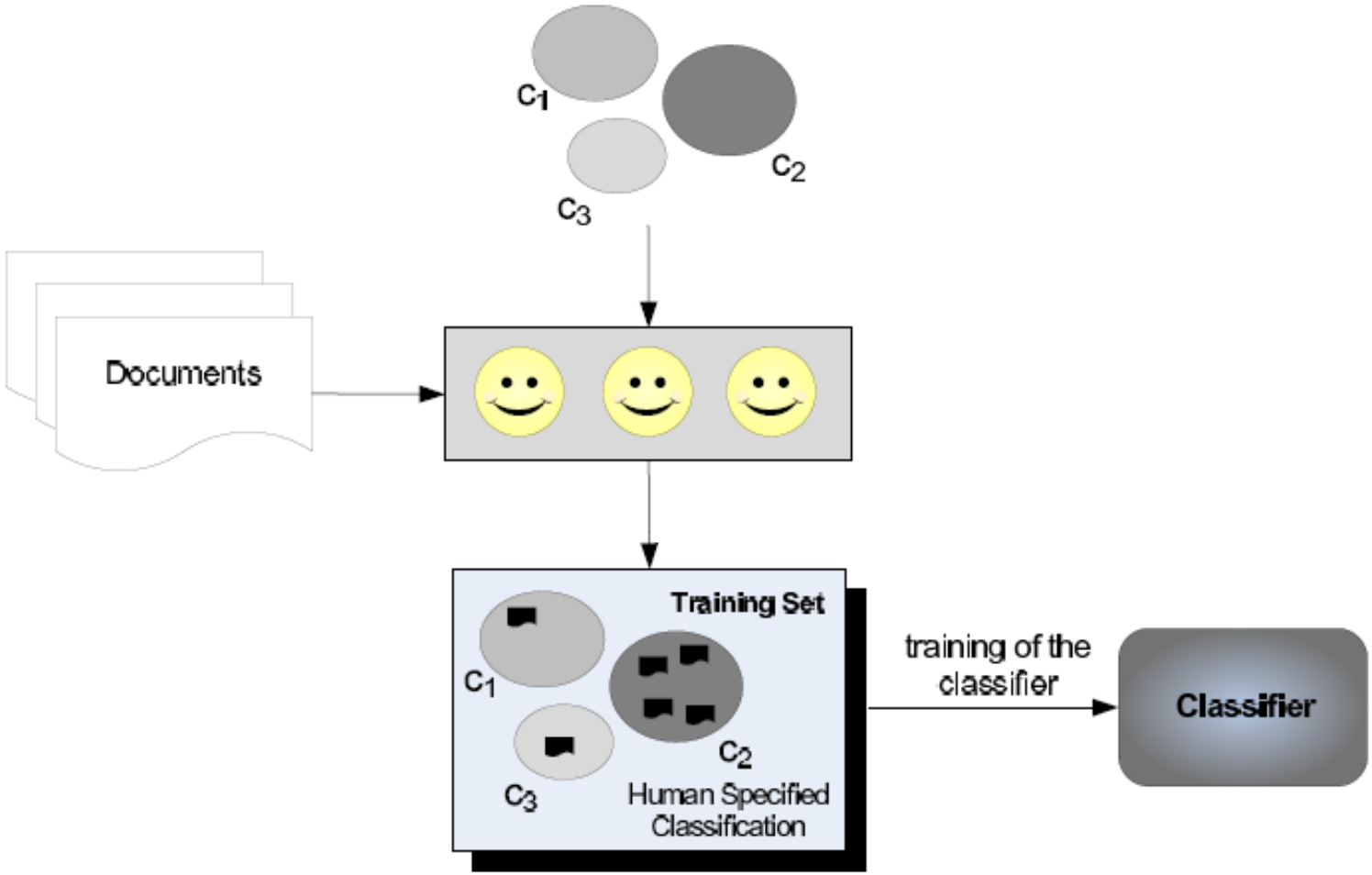
- Prima dell'indicizzazione vengono rimosse le parole come articoli e proposizioni non relazionate coi topic in esame
- L'uso dello *stemming* è controverso (raggruppamento di parole con la stessa radice morfologica)

Dimensionality reduction

- La dimensione dello spazio dei termini può costituire un problema perché:
  - Gli algoritmi di learning non "scalano" facilmente su grandi valori della dimensione
  - Se la dimensione è alta spesso si verificano fenomeni di *overfitting*
- Abbiamo due scelte:
  - Riduzione locale (un insieme di termini diverso per ciascuna categoria)
  - Riduzione globale (il set di termini è valido per qualunque categoria)
- La riduzione può essere:
  - Per selezione
  - Per estrazione

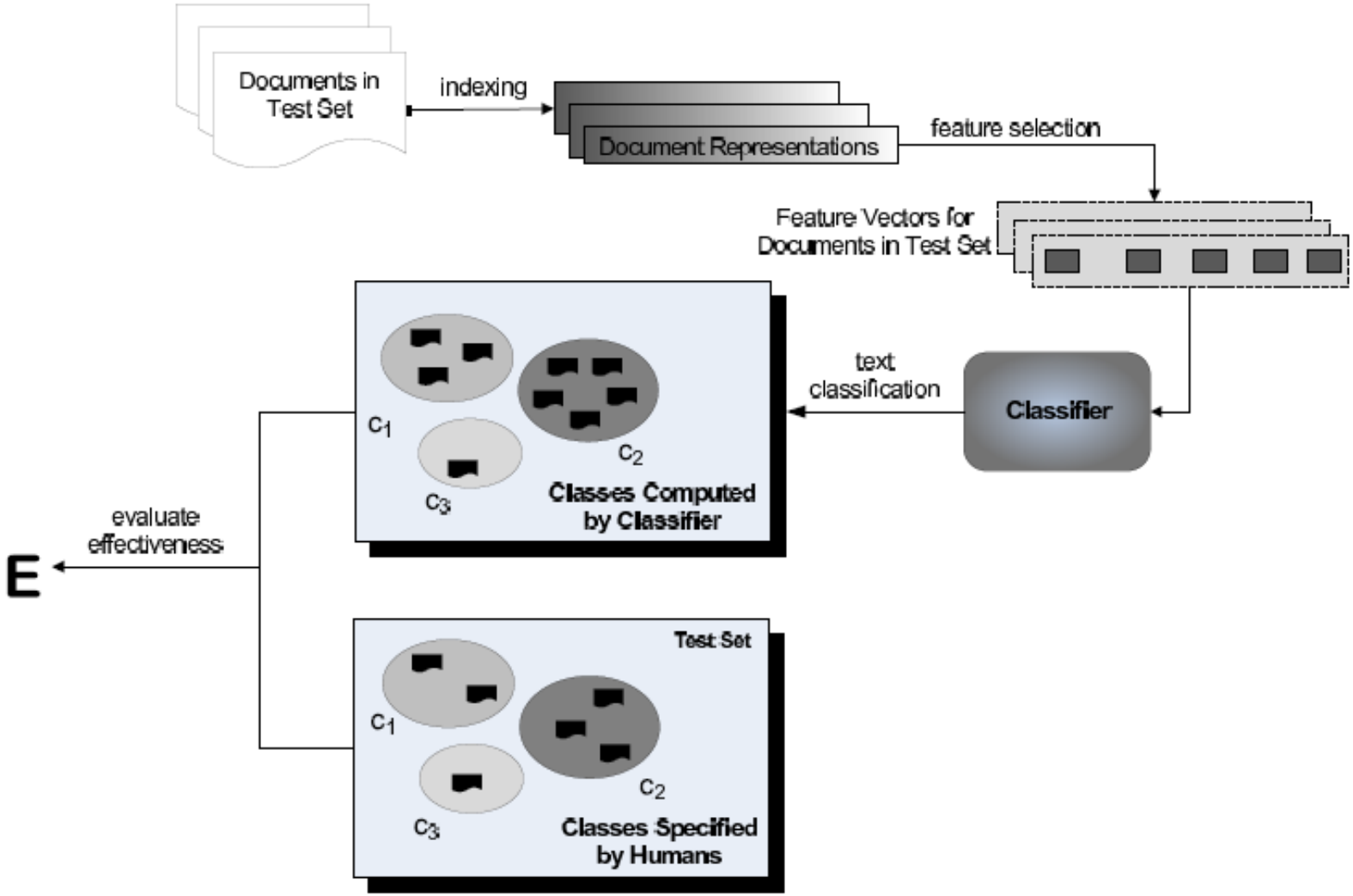
Supervised Algorithms

- The training phase of a classifier



Supervised Algorithms

- Classification and evaluation process



Dimensionality reduction (selection)

- *TSR* (Term Space Reduction) seleziona un sotto-insieme dell'insieme dei termini che massimizzi la effectiveness al momento della indicizzazione
- *Wrapper* partendo da un insieme di termini applica l'algoritmo di learning a sotto-insiemi di termini ottenuti eliminando un termine per volta, fino ad ottenere la effectiveness massima
- *Filtering* seleziona un sotto-insieme dei termini in accordo ai valori forniti da una funzione che misura "l'importanza" dei termini nel processo di categorizzazione (applicato prima della fase di learning)

Funzioni per la selection

Function	Denoted by	Mathematical form
Document frequency	$\#(t_k, c_i)$	$P(t_k \mid c_i)$
DIA association factor	$z(t_k, c_i)$	$P(c_i \mid t_k)$
Information gain	$IG(t_k, c_i)$	$\sum_{c \in \{c_i, \overline{c}_i\}} \sum_{t \in \{t_k, \overline{t}_k\}} P(t, c) \cdot \log \frac{P(t, c)}{P(t) \cdot P(c)}$
Mutual information	$MI(t_k, c_i)$	$\log \frac{P(t_k, c_i)}{P(t_k) \cdot P(c_i)}$
Chi-square	$\chi^2(t_k, c_i)$	$\frac{\lvert Tr\rvert \cdot [P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i) \cdot P(\overline{t}_k, c_i)]^2}{P(t_k) \cdot P(\overline{t}_k) \cdot P(c_i) \cdot P(\overline{c}_i)}$
NGL coefficient	$NGL(t_k, c_i)$	$\frac{\sqrt{\lvert Tr\rvert} \cdot [P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i) \cdot P(\overline{t}_k, c_i)]}{\sqrt{P(t_k) \cdot P(\overline{t}_k) \cdot P(c_i) \cdot P(\overline{c}_i)}}$
Relevancy score	$RS(t_k, c_i)$	$\log \frac{P(t_k \mid c_i) + d}{P(\overline{t}_k \mid \overline{c}_i) + d}$
Odds Ratio	$OR(t_k, c_i)$	$\frac{P(t_k \mid c_i) \cdot (1 - P(t_k \mid \overline{c}_i))}{(1 - P(t_k \mid c_i)) \cdot P(t_k \mid \overline{c}_i)}$
GSS coefficient	$GSS(t_k, c_i)$	$P(t_k, c_i) \cdot P(\overline{t}_k, \overline{c}_i) - P(t_k, \overline{c}_i) \cdot P(\overline{t}_k, c_i)$

Metodi di costruzione

- Classificatori probabilistici
- Classificatori basati su alberi di decisione
- Classificatori basati su regole di decisione
- Metodi di regressione
- Metodi on-line
- Metodo di Rocchio
- Reti neurali
- Classificatori example-based
- SVM

Using Relevance Feedback

- I metodi di *Relevance Feedback* possono essere utilizzati nell'ambito del TC.
- I documenti possono essere rappresentati usando vettori di pesi TF/IDF (normalizzati mediante la frequenza massima dei termini)
- Per ogni categoria, si computa un vettore *prototipo*.
- Si assegnano documenti di testing alla categoria con il vettore prototipo più vicino secondo una misura di similarità

The Rocchio Classifier

- Rocchio relevance feedback
  - ✓ modifies user query based on user feedback
  - ✓ produces new query that better approximates the interest of the user
  - ✓ can be adapted to text classification
- Interpret training set as feedback information
  - ✓ terms that belong to training docs of a given class c_p are said to provide positive feedback
  - ✓ terms that belong to training docs outside class c_p are said to provide negative feedback
- Feedback information summarized by a centroid vector
- New document classified by distance to centroid

The Rocchio Classifier

- Each document d_j represented as a weighted term vector $\vec{d_j}$

$$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$

  - $w_{i,j}$: weight of term k_i in document d_j
  - t: size of the vocabulary

The Rocchio Classifier

- Rocchio classifier for a class c_p is computed as a centroid given by

$$\vec{c}_p = \frac{\beta}{n_p} \sum_{d_j \in c_p} \vec{d}_j - \frac{\gamma}{N_t - n_p} \sum_{d_l \notin c_p} \vec{d}_l$$

where

- n_p: number of documents in class c_p
- N_t: total number of documents in the training set
- terms of training docs in class c_p: positive weights
- terms of docs outside class c_p: negative weights

Classification of Documents

- The classifier assigns test documents to the category with the closest prototype vector based on a similarity measure (for example cosine similarity)

Use of positive feedback only

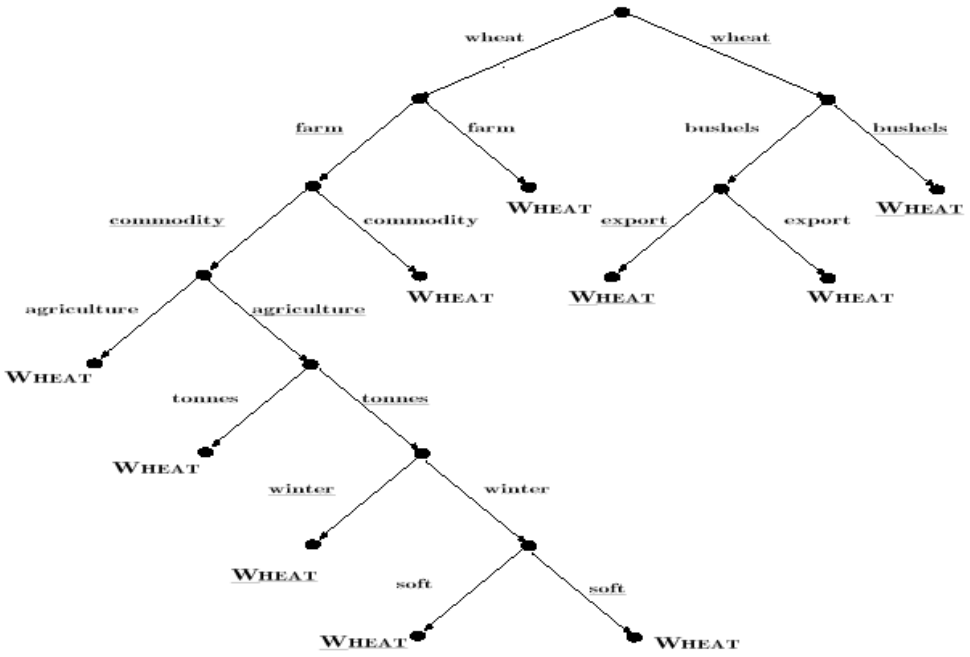
- The prototype vector for each category can be computed using just **training** documents for each category
- The centroid is computed using all the documents that belong to a specific class

Proprietà dell'alg. di Rocchio

- Non garantisce un'ipotesi consistente.
- Forma una semplice generalizzazione degli esempi in ogni classe (*prototipo*)
- La classificazione è basata sulla similarità dei documenti ai vettori prototipo
- I vettori prototipo non necessitano di medie/normalizzazioni poichè la similarità del coseno non è influenzata dalla lunghezza

Alberi di decisione (1)

- Un classificatore basato su alberi di decisione ha:
  - ✓ per nodi interni i *termini* usati per l'indicizzazione
  - ✓ le diramazioni dei nodi sono test sul peso dei termini nel documento di test
  - ✓ per foglie le *categorie* previste



Alberi di decisione (2)

- La classificazione avviene testando ricorsivamente i valori dei pesi relativi ai termini associati ai nodi interni.
- Un metodo per il learning è invece il seguente:
  - ✓ Verificare se tutti gli esempi di training appartengono alla stessa categoria (c_i o la sua negazione).
  - ✓ In caso negativo, selezionare un termine t_k che partizioni il training-set in classi di documenti in cui al termine t_k sia assegnato lo stesso valore, ponendo ciascuna classe in un sotto-albero diverso.
  - ✓ Ripetere ricorsivamente sui sotto-alberi finché tutte le foglie non contengono esempi appartenenti alla stessa categoria (che diviene l'etichetta della foglia)
- La scelta di t_k per partizionare l'albero viene effettuata massimizzando l'*information gain*

Regole di decisione

- Una regola di decisione per categorizzare un testo è espressa in forma normale disgiuntiva (disgiunzione di clausole congiuntive)

if	$((wheat \ \& \ farm)$	**or**
	$(wheat \ \& \ commodity)$	**or**
	$(bushels \ \& \ export)$	**or**
	$(wheat \ \& \ tonnes)$	**or**
	$(wheat \ \& \ winter \ \& \ \neg \ soft))$	**then** WHEAT **else** $\neg$ WHEAT

- I classificatori basati su regole di decisione sono simili a quelli basati su alberi di decisione ma sono in genere più compatti.
- I letterali denotano la presenza (o l'assenza) di termini nel documento sotto esame, mentre la testa della clausola stabilisce di classificare sotto una data categoria.

Regole di decisione (2)

- Inizialmente ogni esempio di training viene trasformato in una clausola:
  - ✓ $\eta_1,\eta_2,\eta_3 \ldots \eta_N \rightarrow \gamma_i$
- I termini η_i rappresentano la presenza di termini nel documento in esame mentre γ_i vale c_i oppure la sua negazione.
- Un set di clausole è già un classificatore che però causa *overfitting*.
- Attraverso una procedura di generalizzazione il set viene semplificato il più possibile giungendo ad una rappresentazione compatta che non sacrifichi però la "copertura"

Nearest-Neighbor Learning Algorithm

- L'apprendimento consiste solo nel memorizzare gli esempi di training presenti in Tr
- Data un'istanza di testing *x*:
  - ✓ Calcola la similarità tra *x* e tutti gli esempi in *Tr*.
  - ✓ Assegna a *x* la categoria dell'esempio più simile in *Tr*.
- Non effettua esplicitamente una generalizzazione o determina prototipi di categoria
- AKA:
  - ✓ Case-based
  - ✓ Memory-based
  - ✓ Lazy learning

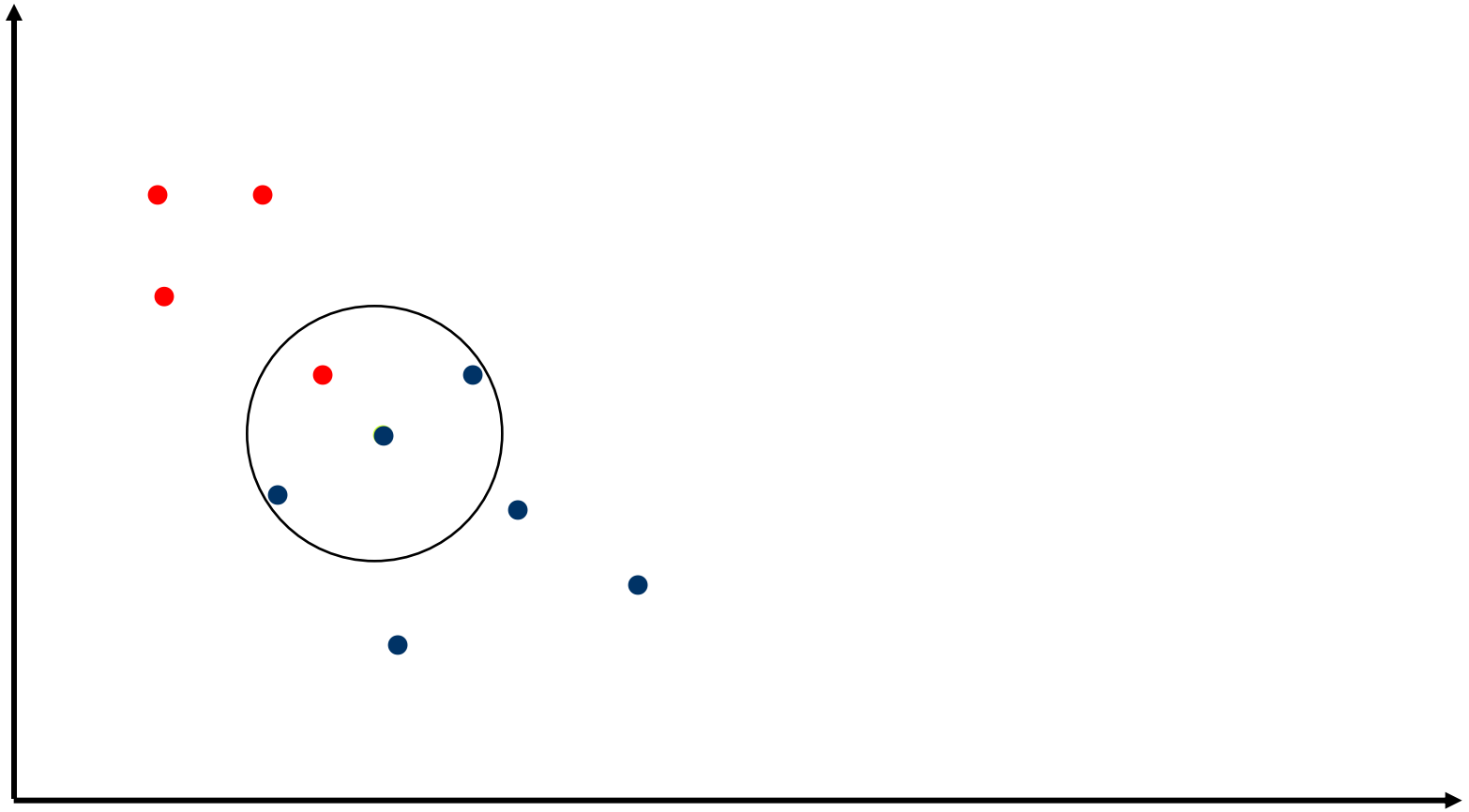
K Nearest-Neighbor

- Effettuare la classificazione basandosi esclusivamente sull'esempio più vicino (simile) può generare errori:
  - ✓ Esempio atipico
  - ✓ Rumore (errori) nell'etichettare gli esempi di training
- Alternativa: trovare i *k* esempi più simili e restituire la classe della maggioranza di questi
- Tipicamente *k* è dispari per evitare parità (3 e 5 sono i valori più comuni)

Similarity Metrics

- Il metodo Nearest neighbor dipende da una misura di similarità (distanza)
- Testi: la similarità del coseno tra vettori TF-IDF è quella più efficace.

3 Nearest Neighbor (Euclidian Distance)

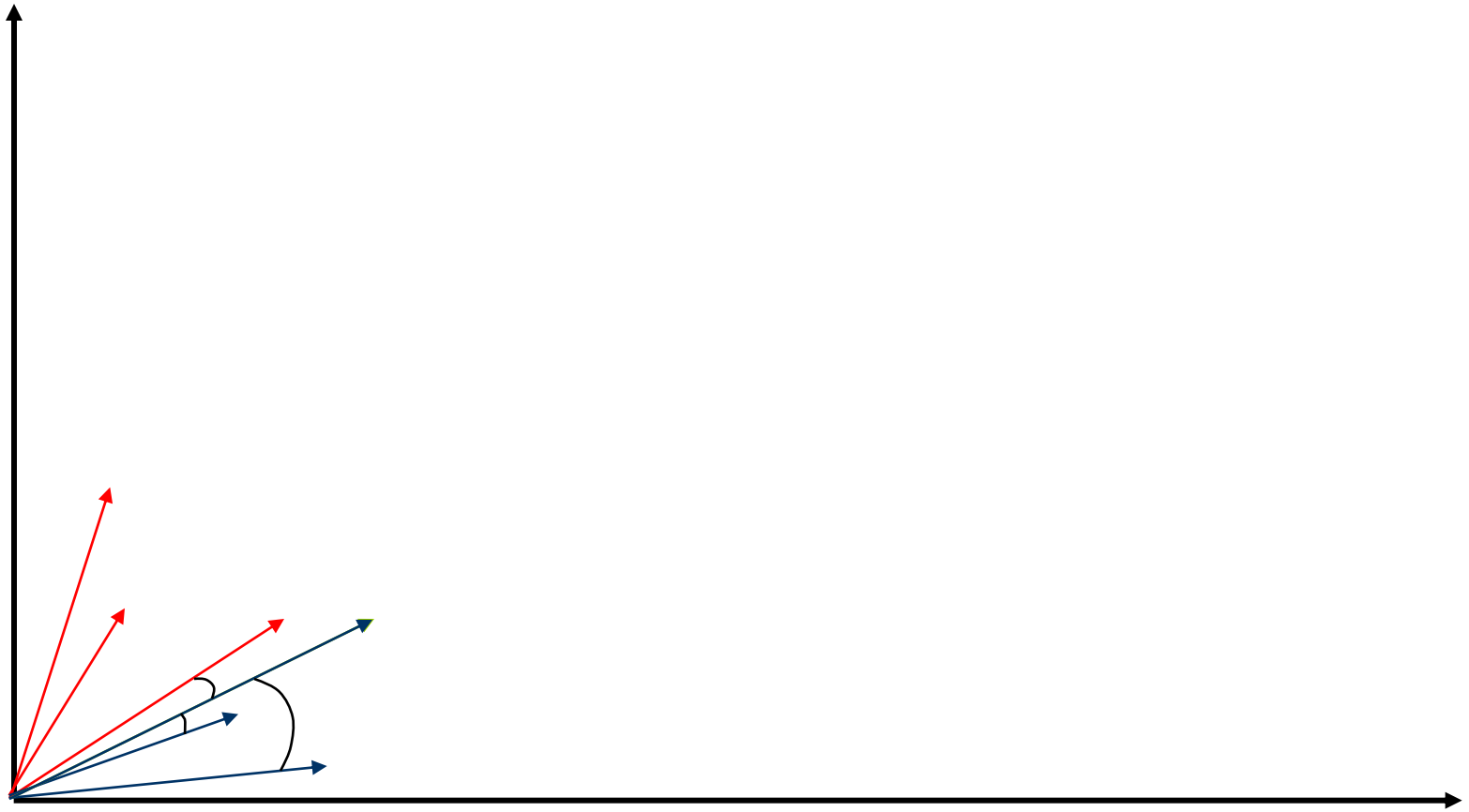


K Nearest Neighbor for Text

Training:
For each training example $\langle x, c(x) \rangle \in Tr$
    Compute the corresponding TF-IDF vector, $\mathbf{d}_x$, for document x

Test instance *y*:
Compute TF-IDF vector **d** for document y
For each $\langle x, c(x) \rangle \in Tr$
    Let $s_x = \text{cosSim}(\mathbf{d}, \mathbf{d}_x)$
Sort examples, x, in Tr by decreasing value of s_x
Let N be the first k examples in Tr
(*get most similar neighbors*)
Return the majority class of examples in N

3 Nearest Neighbor for Text



Bayesian Methods

- Learning and classification methods based on probability theory.
- Bayes theorem plays a critical role in probabilistic learning and classification.
- Uses *prior* probability of each category given no information about an item.
- Categorization produces a *posterior* probability distribution over the possible categories given a description of an item.

Axioms of Probability Theory

- All probabilities between 0 and 1

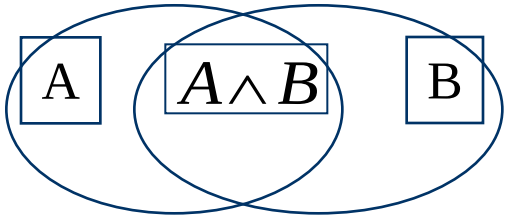
$$0 \leq P(A) \leq 1$$

- True proposition has probability 1, false has probability 0.

  P(true) = 1 P(false) = 0.

- The probability of disjunction is:

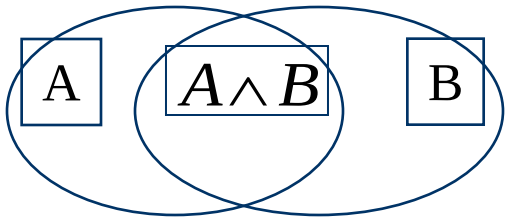
$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$



Conditional Probability

- P(*A* | *B*) is the probability of *A* given *B*
- Assumes that *B* is all and only information known.
- Defined by:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$



Independence

- *A* and *B* are *independent* iff:

$$P(A \mid B) = P(A)$$

$$P(B \mid A) = P(B)$$

- Therefore, if *A* and *B* are independent:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)} = P(A)$$

$$P(A \wedge B) = P(A)P(B)$$

Bayes Theorem

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Simple proof from definition of conditional probability:

$$P(H \mid E) = \frac{P(H \wedge E)}{P(E)}$$

$$P(E \mid H) = \frac{P(H \wedge E)}{P(H)}$$

$$P(H \wedge E) = P(E \mid H)P(H)$$

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Bayesian Categorization

- Let set of categories be $\{c_1, c_2, \ldots c_n\}$
- Let E be description of an instance.
- Determine category of E by determining for each c_i

$$P(c_i \mid E) = \frac{P(c_i)P(E \mid c_i)}{P(E)}$$

- P(E) can be determined since categories are complete and disjoint.

$$\sum_{i=1}^{n} P(c_i \mid E) = \sum_{i=1}^{n} \frac{P(c_i)P(E \mid c_i)}{P(E)} = 1$$

$$P(E) = \sum_{i=1}^{n} P(c_i)P(E \mid c_i)$$

Bayesian Categorization (cont.)

- Need to know:
  - ✓ Priors: $P(c_i)$
  - ✓ Conditionals: $P(E \mid c_i)$
- $P(c_i)$ are easily estimated from data.
  - ✓ If n_i of the examples in D are in c_i, then $P(c_i) = n_i / |D|$
- Assume instance is a conjunction of binary features:

$$E = e_1 \wedge e_2 \wedge \ldots \wedge e_m$$

- Too many possible instances (exponential in m) to estimate all $P(E \mid c_i)$

Naïve Bayesian Categorization

- If we assume features of an instance are independent given the category (c_i) (*conditionally independent*):

$$P(E \mid c_i) = P(e_1 \wedge e_2 \wedge \ldots \wedge e_m \mid c_i) = \prod_{j=1}^{m} P(e_j \mid c_i)$$

- Therefore, we then only need to know $P(e_j \mid c_i)$ for each feature and category.

Naïve Bayes Example

- C = {allergy, cold, well}
- e_1 = sneeze; e_2 = cough; e_3 = fever
- E = {sneeze, cough, ¬fever}

Prob	**Well**	**Cold**	**Allergy**
$P(c_i)$	0.9	0.05	0.05
$P(\text{sneeze}\|c_i)$	0.1	0.9	0.9
$P(\text{cough}\|c_i)$	0.1	0.8	0.7
$P(\text{fever}\|c_i)$	0.01	0.7	0.4

Naïve Bayes Example (cont.)

Probability	Well	Cold	Allergy
$P(c_i)$	0.9	0.05	0.05
P(sneeze | c_i)	0.1	0.9	0.9
P(cough | c_i)	0.1	0.8	0.7
P(fever | c_i)	0.01	0.7	0.4

E={sneeze, cough, ¬fever}

P(well | E) = (0.9)(0.1)(0.1)(0.99)/P(E)=0.0089/P(E)
P(cold | E) = (0.05)(0.9)(0.8)(0.3)/P(E)=0.01/P(E)
P(allergy | E) = (0.05)(0.9)(0.7)(0.6)/P(E)=0.019/P(E)

Most probable category: allergy

Estimating Probabilities

- Normally, probabilities are estimated based on observed frequencies in the training data.
- If D contains n_i examples in category c_i, and n_{ij} of these n_i examples contains feature e_j, then:

$$P(e_j \mid c_i) = \frac{n_{ij}}{n_i}$$

- However, estimating such probabilities from small training sets is error-prone.
- If due only to chance, a rare feature, e_k, is always false in the training data, $\forall c_i : P(e_k \mid c_i) = 0$.
- If e_k then occurs in a test example, E, the result is that $\forall c_i : P(E \mid c_i) = 0$ and $\forall c_i : P(c_i \mid E) = 0$

Naïve Bayes for Text

- Modeled as generating a bag of words for a document in a given category by repeatedly sampling with replacement from a vocabulary $V = \{w_1, w_2, \ldots w_m\}$ based on the probabilities $P(w_j \mid c_i)$.
- Smooth probability estimates with Laplace *m*-estimates assuming a uniform distribution over all words ($p = 1/|V|$) and $m = |V|$

  Laplace correction to deal with missing words → 0 probabilities
  - ✓ Equivalent to a virtual sample of seeing each word in each category exactly once.

Text Naïve Bayes Algorithm (Train)

Let V be the vocabulary of all words in the documents in D
For each category $c_i \in C$
    Let D_i be the subset of documents in D in category c_i
    $P(c_i) = |D_i| / |D|$
    Let T_i be the concatenation of all the documents in D_i
    Let n_i be the total number of word occurrences in T_i
    For each word $w_j \in V$
        Let n_{ij} be the number of occurrences of w_j in T_i
        Let $P(w_j \mid c_i) = (n_{ij} + 1) / (n_i + |V|)$

Text Naïve Bayes Algorithm (Test)

Given a test document x
Let n be the number of word occurrences in x
Return the category:

$$\underset{c_i \in C}{\operatorname{argmax}} \; P(c_i) \prod_{i=1}^{n} P(a_i \mid c_i)$$

where a_i is the word occurring the i-th position in x

Underflow Prevention

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since $\log(xy) = \log(x) + \log(y)$, it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

Naïve Bayes Posterior Probabilities

- Classification results of naïve Bayes (the class with maximum posterior probability) are usually fairly accurate.
- However, due to the inadequacy of the conditional independence assumption, the actual posterior-probability numerical estimates are not.
  - ✓ Output probabilities are generally very close to 0 or 1.

Valutazione di un classificatore (1)

- La valutazione è sperimentale perché il problema non ha una specifica formale che consenta un altro tipo di valutazione
- La valutazione deve essere effettuata su dati di test indipendenti dai dati di training (solitamente insiemi disgiunti di istanze)
- *Classification accuracy*: *c*/*n*, dove *n* è il numero totale di istanze di testing e *c* è il numero di istanze di testing classificate correttamente dal sistema
- I risultati possono variare in base all'uso di diversi training e testing sets
- Solitamente si mediano i risultati ottenuti su diversi training and test sets ("splittando" l'insieme iniziale dei dati) per ottenere risultati più significativi

Valutazione di un classificatore (2)

- Le misure standard di efficacia per i classificatori sono tratte dall' I.R.: *precision* (π) e *recall* (ρ)
- π e ρ sono ottenuti mediando su tutte le categorie
- π può essere pensata come la "correttezza" del classificatore, ρ invece come la sua "completezza"

Category c_i		expert judgments	
		YES	**NO**
classifier judgments	**YES**	TP_i	FP_i
	NO	FN_i	TN_i

Category set $\mathcal{C} = \{c_1, \ldots, c_{\lvert\mathcal{C}\rvert}\}$		expert judgments	
		YES	**NO**
classifier	**YES**	$TP = \sum_{i=1}^{\lvert\mathcal{C}\rvert} TP_i$	$FP = \sum_{i=1}^{\lvert\mathcal{C}\rvert} FP_i$
judgments	**NO**	$FN = \sum_{i=1}^{\lvert\mathcal{C}\rvert} FN_i$	$TN = \sum_{i=1}^{\lvert\mathcal{C}\rvert} TN_i$

$$\hat{\pi}_i = \frac{TP_i}{TP_i + FP_i} \qquad \hat{\rho}_i = \frac{TP_i}{TP_i + FN_i}$$

Valutazione di un classificatore (3)

- Microaveraging

$$\hat{\pi}^{\mu} = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{|\mathcal{C}|} TP_i}{\sum_{i=1}^{|\mathcal{C}|} (TP_i + FP_i)}$$

$$\hat{\rho}^{\mu} = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{|\mathcal{C}|} TP_i}{\sum_{i=1}^{|\mathcal{C}|} (TP_i + FN_i)}$$

- Macroaveraging

$$\hat{\pi}^{M} = \frac{\sum_{i=1}^{|\mathcal{C}|} \hat{\pi}_i}{|\mathcal{C}|} \qquad \hat{\rho}^{M} = \frac{\sum_{i=1}^{|\mathcal{C}|} \hat{\rho}_i}{|\mathcal{C}|}$$

Valutazione di un classificatore (1)

- La valutazione è sperimentale perché il problema non ha una specifica formale che consenta un altro tipo di valutazione
- La valutazione deve essere effettuata su dati di test indipendenti dai dati di training (solitamente insiemi disgiunti di istanze)
- *Classification accuracy*: *c*/*n*, dove *n* è il numero totale di istanze di testing e *c* è il numero di istanze di testing classificate correttamente dal sistema
- I risultati possono variare in base all'uso di diversi training e testing sets
- Solitamente si mediano i risultati ottenuti su diversi training and test sets ("splittando" l'insieme iniziale dei dati) per ottenere risultati più significativi

K-Fold Cross-Validation (1)

- Teoricamente, test and training sets devono essere indipendenti ad ogni prova (trial).
  - ✓ Richiede una notevole quantità di dati etichettati
- Partiziona i dati in *K* insiemi disgiunti.
- Esegue K prove: ogni prova usa un diverso fold (insieme) per il testing, addestrando il sistema sugli altri *K–1* insiemi.
- Le metriche di valutazione sono mediate sulle *K* prove
- La procedura garantisce l'indipendenza
- Tipicamente, $K = 10$

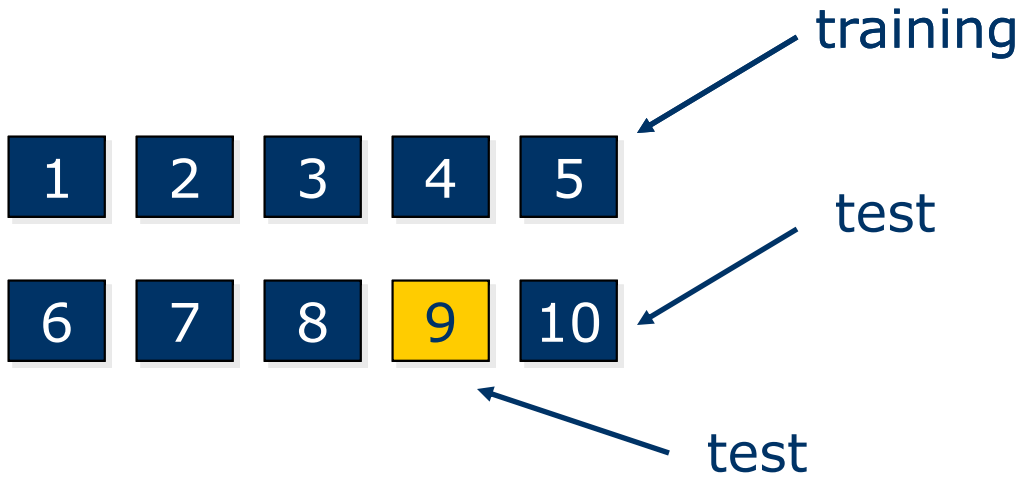
K-fold cross validation (2)

- Partiziono D ottenendo *K* insiemi Te_i
- Applico iterativamente il train-and-test a

$$<Tr_i=D-Te_i , Te_i>$$

- La effectiveness del sistema è ottenuta mediando quella dei *K* classificatori

10-fold cross-validation



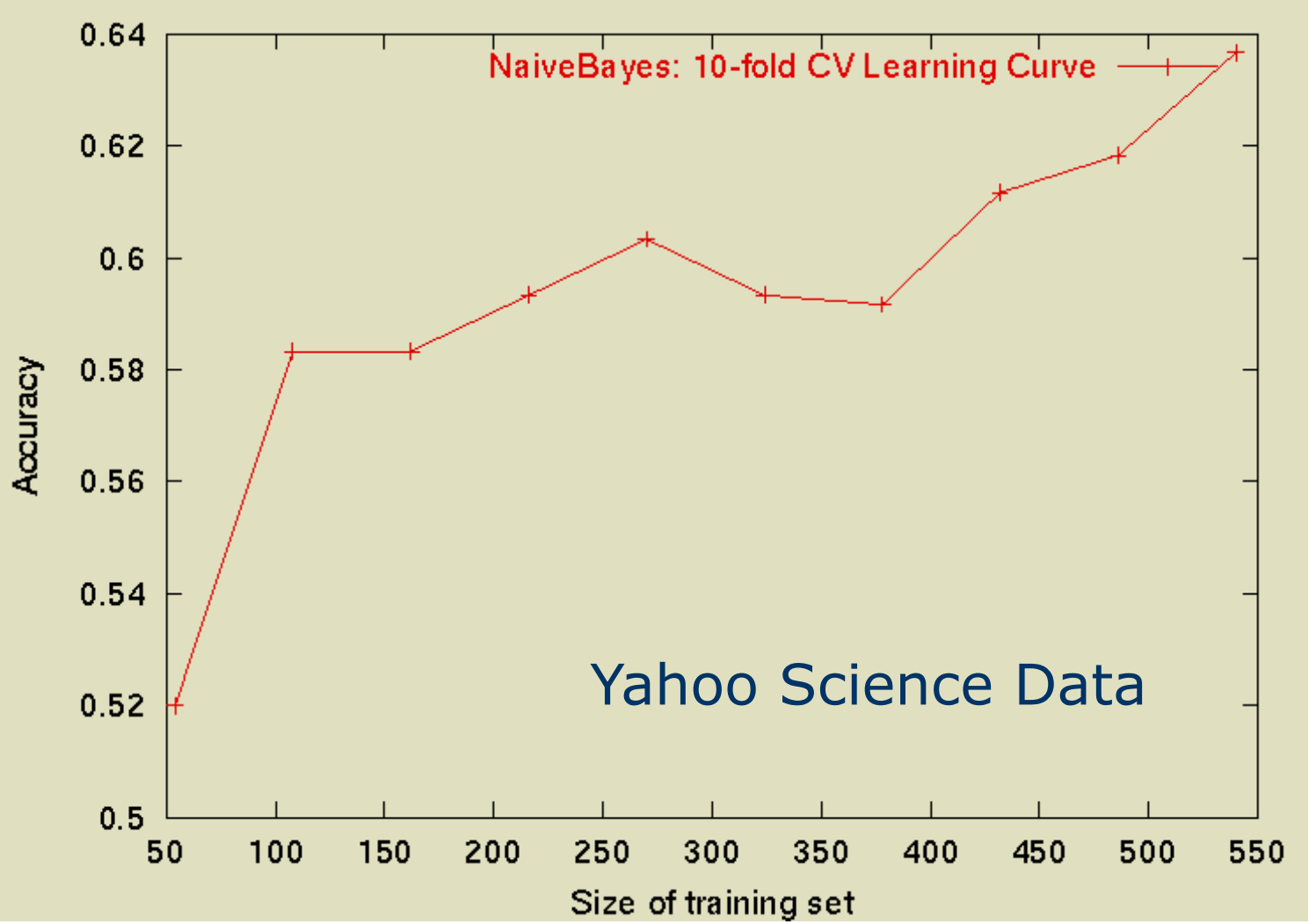
Curve di Learning

- Nella pratica, ottenere dataset etichettati non è facile
- È interessante osservare come varia l'accuratezza predittiva di un classificatore in base al numero di istanze usate nella fase di learning
- *Le curve di appendimento (learning curves)* rappresentano la classification accuracy su dati di test indipendenti (asse *Y*) rispetto al numero di esempi di training (asse *X*)

K-Fold Learning Curves

- Curve di learning mediate sulle varie prove indipendenti
- *K*-fold cross validation: *K prove*
- Per ogni prova, si addestra il classificatore su frazioni del training set con un numero crescente di instanze, misurando l'accuratezza sui dati di test per ogni punto della curva di learning desiderata

Sample Learning Curve



References

- M. Ackerman, D. Billsus, S. Gaffney, S. Hettich, G. Khoo, D. J. Kim, R. Klefstad, C. Lowe, A. Ludeman, J. Muramatsu, K. Omori, M. Pazzani, D. Semler, B. Starr & P. Yap, Learning Probabilistic User Profiles: Applications for Finding Interesting Web Sites, Notifying Users of Relevant Changes to Web Pages, and Locating Grant Opportunities, *AI Magazine* 18(2):47-56, 1997.
- M. Balabanovic & Y. Shoham, Learning information retrieval agents: Experiments with automated Web browsing, *Proc. of the 1995 AAAI Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, Stanford, 1995.
- Firefly Network, Inc. 1997. *Collaborative Filtering Technology: An Overview*. http://www.firefly.net/company/CollaborativeFiltering.fly
- T. Joachims, D. Freitag & T. M. Mitchell, Web Watcher: A Tour Guide for the World Wide Web, *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI 97)*, 770-777, Nagoya, Japan, August 23-29, Morgan Kaufmann, 1997.
- B. Krulwich & C. Burkey, The InfoFinder Agent: Learning User Interests through Heuristic Phrase Extraction, *IEEE Expert* 12(5):22-27, 1997.
- Y. Lashkari, M. Metral & P. Maes: Collaborative Interface Agents. *Proc. of the 12th National Conf. on Artificial Intelligence (AAAI 1994)*, 444-449, Seattle, WA, USA, July 31 - August 4, AAAI Press.
- H. Lieberman, Letizia: An Agent That Assists Web Browsing, *Proc. of the 14th Int. Joint Conference on Artificial Intelligence (IJCAI 95)*, 924-929, Montréal, Québec, Canada, August 20-25, Morgan Kaufmann, 1995.
- T.M. Mitchell, "Machine Learning", McGraw-Hill, 1997.
- P. Maes, Intelligent Software: Easing the Burdens that Computers Put on People. *IEEE Expert*, 11(6) 62-63, 1996.
- P. Maes, *Information filtering system. Overview, slides*. Available at: *http://infoweb.vub.ac.be/~lasse.software_agents/infofiltering/sld002.htm.*
- M. Pazzani, "Machine Learning and Information Filtering on the Internet", IJCAI-97 Tutorial, Nagoya, Japan, Aug 1997.
- M. Pazzani, J. Muramatsu & D. Billsus, Syskill & Webert: Identifying Interesting Web Sites, *Proc. of the 13th National Conf. on Artificial Intelligence and 8th Innovative Applications of Artificial Intelligence Conf. (AAAI 96/IAAI 96)*, 54-61, August 4-8, Portland, Oregon, AAAI Press/The MIT Press, 1996.
- B.J. Rhodes & T. Starner, Remembrance Agent: A continuously running automated information retrieval system, *Proc. of the 1st Int. Conf. on The Practical Application Of Intelligent Agents and Multi Agent Technology (PAAM '96)*, 487-495, 1996.
- Salton, G., & McGill, M. J. "Introduction to Modern Information Retrieval", McGraw-Hill, 1983.
- F. Sebastiani, "Machine Learning in Automated Text Categorization", ACM Computing Surveys, 34(1):1-47, 2002.