Web Mining and Retrival

Lorenzo Cristofori, Claudiu Daniel Hromei

April 2019

Indice

1 Supervised Learning

Ha avuto un grande successo nelle applicazioni del mondo reale. E' usata in praticamente tutti i domini, inclusi testi e web. L'apprendimento supervisionato, in machine learning, è anche detto classificazione o apprendimento induttivo. Questo tipo di apprendimento è analogo all'apprendimento umano per cui si apprende dalle esperienze passate per generare nuova conoscenza con il fine di migliorare le abilità per risolvere i task del mondo reale. Ci sono molti tipi di task per l'apprendimento supervisionato, tra cui trovare una funzione target che possa essere usata per predire i valori di classi di attributi discreti.

1.1 Basic Concepts

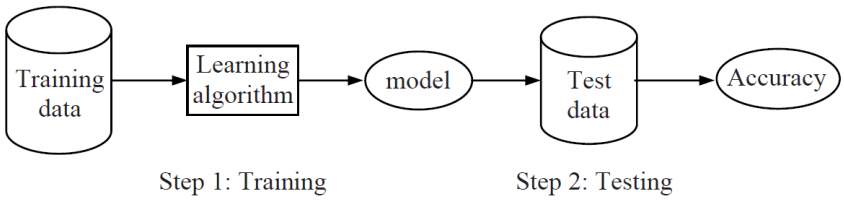
Un insieme di dati usato in un task di apprendimento consiste di un insieme di records, i quali sono descritti da un insieme di attributi $A = A_1, A_2, \ldots, A_n$. L'insieme dei dati ha inoltre un attributo speciale target C, che è chiamato attributo classe. La classe attributo C ha un insieme di valori discreti $C = c_1, c_2, \ldots, c_m$. I dati usati per fare apprendimento sono chiamati training data (o training set). Dopo che un modello è stato appreso o costruito dal training data attraverso un learning algorithm, viene valutato usando un insieme di test data per accurare la precisione del modello.
Accuracy = Numero di classificazioni corrette / numero totale di test
Error Rate = Numero di classificazioni errate / numero totale di test
(oppure $1 - Accuracy$)

Le fasi di apprendimento sono:

1. Training Step: si usa un algoritmo di apprendimento per generare un modello di classificazione a partire dal training data.

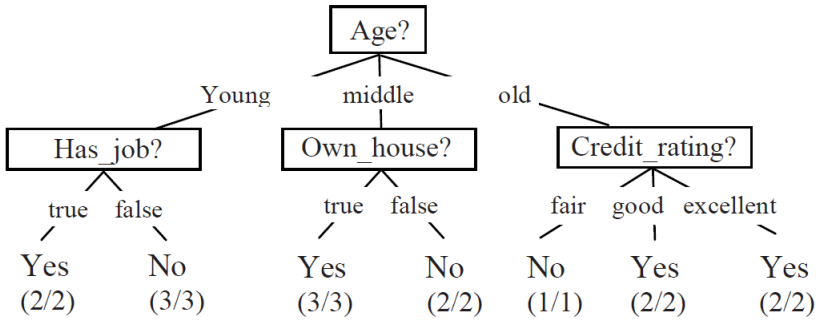
2. Testing Step: Il modello di apprendimento viene testato usando il test set ottenendo la accuracy della classificazione.

3. Se la accuracy del modello è soddisfacente questo può essere usato nel mondo reale altrimenti è necessario tornare indietro e cambiare ad esempio l'algoritmo di apprendimento o qualche processo dei dati (data pre-processing).



2 Decision Tree

La decision tree learning è una delle tecniche più frequentemente usata per la classificazione. La accuracy della classificazione risulta molto competitiva ed è molto efficiente. Una proprietà interessante dei Decision Tree è che i percorsi dell'albero (o le regole) sono mutuamente esclusive e esaustive. Questo significa che ogni istanza di un dato è coperta (soddisfa la condizione) da una singola regola (un percorso dell'albero). Un albero di decisione generalizza bene i dati tanto più è piccolo (compatto). Quindi il problema diventa costruire l'albero migliore che è il più piccolo e accurato (NPC), ogni algoritmo di costruzione degli alberi è un algoritmo euristico. Un albero di decisione può essere facilmente trasformato in un insieme di regole. Un albero di decisione per l'apprendimento è costituito da controlli sulle feature nei nodi interni e nella radice e output della classificazione sulle foglie, un valore per la classe (yes, no).



(x,y) sotto ogni foglia precisa il numero x su y di esempi del training set che arrivati in quella foglia appartengono alla classe (Yes, no). L'obiettivo nella scelta dell'albero è di ridurre l'impurità dei dati o il grado di incertezza di questi. Un sotto insieme di dati è detto puro se finiscono tutti in una stessa classe. Quindi l'euristica è di cercare gli attributi con la massima information gain o gain ratio. Si **preferiscono** alberi più piccoli possibili e senza ripetizioni di features nei nodi interni.

2.1 Information Gain

Più i dati diventano puri piu diminuisce l'entropia. L'**entropia** è una misura di incertezza associata ad una variabile randomica.

Entropia di D: Dato un insieme di esempi D è possibile calcolare l'entropia originale del data set in questo modo:

$$H[D] = -\sum_{j=1}^{\|C|} P(c_j) \log_2 P(c_j)$$

Dove C è l'insieme delle classi target e $P(c_j)$ è la probabilità della classe j, cioè il numero di documenti in quella classe diviso il numero totale del documenti (probabilità frequentista).

Entropia di un attributo A_i: se fissiamo l'attributo A_i, con v valori, come radice questo partizionerà i dati in v sottoinsiemi $D_1, D_2, ..., D_v$. L'entropia attesa quando viene usato A_i come radice è data da:

$$H_{A_i}[D] = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} H[D_j]$$

L'**information Gain** ottenuta dal selezionare A_i per partizionare i dati è data dalla differenza tra l'entropia a priori e l'entropia del ramo scelto:

$$gain(D, A_i) = H[D] - H_{A_i}[D]$$

Per dividere l'albero corrente si sceglie l'attributo con il più alto gain.

2.2 Learning Algorithm

Un albero di decisione T partiziona il training set D in sotto insiemi disgiunti in modo che siano il più puri possibile (della stessa classe). L'apprendimento di un albero è generalmente fatto usando il metodo del dividi et impera che ricorsivamente partiziona i dati per generare l'albero. All'inizio tutti gli esempi si trovano nella radice. Con la crescita dell'albero gli esempi sono divisi ricorsivamente. L'algoritmo si ferma quando tutti gli esempi dei dati correnti del training set fanno parte della stessa classe o quando, all'interno di uno stesso path, sono stati usati tutti gli attributi a disposizione. Il miglior attributo è scelto secondo una funzione detta funzione di impurità, l'attributo sarà quello che in qualche modo minimizza il valore associato a tale funzione. La chiave della costruzione di un albero per l'apprendimento sta nella scelta della funzione di impurità.

3 Classifier Evaluation

Dopo aver costruito un classificatore è necessario valutarne l'accuracy. Ci sono molti modi di valutare un classificatore. Ad esempio utilizzando l'accuracy o l'error rate (1 - accuracy).

Il training set è utilizzato per addestrare il classificatore mentre il test set per valutarlo. Il training set quindi non dovrebbe essere usato per la valutazione poiché questo potrebbe portare il classificatore a dare un valore elevato di accuracy nel training set contrapposto ad un basso valore nel test set. Questo potrebbe essere un sintomo di overfitting, cioè che il modello si adegua fin troppo bene ai dati del training set, senza alcun grado di libertà. Per evitare l'overfitting si puo dividere il data set in due parti: una da usare come training set$(70 - 80\%)$ e uno da usare come test set$(20 - 30\%)$. Per fare questo si possono utilizzare più approcci:

1. si può scegliere randomicamente un sotto insieme dal data set ed utilizzarlo come test set e il resto come training set.

2. Se i dati rimangono significativi nel tempo si possono usare i primi come dati per la fase di training e i dati futuri come test set.

3.1 Multiple Random Sampling

Se il data set è troppo piccolo dividendolo in training set e test set si rischia di non avere abbastanza informazioni, quindi è possibile eseguire la divisione dei dati, l'apprendimento e la valutazione n volte dividendo ogni volta in insiemi differenti, alla fine si avranno vari valori di accuracy e si può prenderne la media.

3.2 n-fold Cross Validation

Quando il data set è piccolo il metodo n-fold cross-validation è molto comune. In questo metodo i dati sono partizionati in n insiemi disgiunti di uguale grandezza. Ogni sottoinsieme è quindi usato come test set a i rimanenti $n-1$ sono combinati a formare il training set per addestrare il classificatore. Questa procedura è avviata n volte e restituirà n accuracy.

3.3 Precision, Recall, F-score and Breakeven Point

In alcune applicazioni si può essere interessati ad una sola classe. Questo è particolarmente vero per i testi e le applicazioni web. Potremmo essere interessati solo in documenti di pagine web di un particolare topic. La classe alla quale siamo interessati è generalmente chiamata positive class e il resto negative class. L'accuracy in alcuni casi non risulta una misura adatta. Precision e Recall risultano più adatte in alcune applicazioni poiché queste misurano quanto la classificazione sia precisa e completa rispetto alla positive class. E' utile introdurre queste misure usando una matrice di confusione. Una matrice di confusione contiene le informazioni riguardo gli attuali e i valori predetti da un classificatore.

	Classificati positivi	classificati negativi
Positivi	TP	FN
Negativi	FP	TN

Dove: **TP**: Il numero di classificazioni corrette di esempi positivi (true positive) **FN**: Il numero di classificazioni errate di esempi positivi (false negative) **FP**: Il numero di classificazioni errate di esempi negativi (false positive) **TN**: Il numero di classificazioni corrette di esempi negativi (true negative)

La Precision e la Recall sono definite come segue:

$$p = \frac{TP}{TP + FP}. \qquad r = \frac{TP}{TP + FN}.$$

La **Precision** è il numero di esempi positivi classificati correttamente diviso il numero totale di esempi che sono stati classificati come positivi. La **Recall** è il numero di esempi positivi classificati correttamente diviso il numero totale di esempi positivi presenti nel test set.

Ad ogni modo risulta difficile comparare classificatori sulla base di 2 misure, le quali non sono funzionalmente correlate. Ad esempio in alcuni test set la precision può essere molto alta mentre la recall bassa e vice versa.

Per comparare più classificatori si può usare una singola misura detta F-score:

$$F = \frac{2pr}{p + r}$$

La F-score è la media armonica delle due misure, precision e recall.

$$F = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

La media armonica di due numeri risulta tendere al più piccolo dei due. Esiste inoltre un'ulteriore misura detta **Break-even Point** la quale viene usata nella comunità IR. La breakeven point si ha quando la precision e la recall sono uguali, denota quindi l'interpolazione della stima dei valori per i quali recall=precision.

4 Naive Bayesian Classification

L'apprendimento supervisionato può essere studiato da un punto di vista probabilistico. Il task della classificazione può essere visto come stimare la probabilità a posteriori delle classe dato un esempio d.

$P(C = c_j|d)$

Allora la classe con la più alta probabilità sarà quella assegnata a d.

Formalmente, sia $A_1, A_2, ..., A_{|A|}$ l'insieme degli attributi con valori discreti nel dataset D. Sia C l'attributo classe con $|C|$ valori, $c_1, c_2, ..., c_{|C|}$. Dato un esempio di test con valori degli attributi osservati da a_1 fino ad $a_{|A|}$, Dove a_i è un possibile valore di A_i (o un membro del dominio di A_i)

$d =< A_1 = a_1, ..., A_{|A|} = a_{|A|} >$

La predizione è la classe c_j tale che $P(C = c_j|A_1 = a_1, ..., A_{|A|} = a_{|A|})$ è massima. c_j è chiamata Maximum a posteriori (MAP) Hypothesis.

Dalla regola di Bayes la quantità precedente può essere calcolata come:

$$\begin{aligned}
&\Pr(C = c_j \mid A_1 = a_1,..., A_{|A|} = a_{|A|}) \\
&= \frac{\Pr(A_1 = a_1,..., A_{|A|} = a_{|A|} \mid C = c_j)\Pr(C = c_j)}{\Pr(A_1 = a_1,..., A_{|A|} = a_{|A|})} \\
&= \frac{\Pr(A_1 = a_1,..., A_{|A|} = a_{|A|} \mid C = c_j)\Pr(C = c_j)}{\sum_{k=1}^{|C|} \Pr(A_1 = a_1,..., A_{|A|} = a_{|A|} \mid C = c_k)\Pr(C = c_k)}.
\end{aligned}$$

$P(C = c_j)$ è la probabilità a priori della classe c_j che può essere estrapolata dal training data. Se siamo interessati a fare classificazione $P(A_1 = a_1, ..., A_{|A|} = a_{|A|})$ risulta irrilevante in quanto non dipende dalla classe. Allora è necessario calcolare solo $P(A_1 = a_1, ..., A_{|A|} = a_{|A|}|C = c_j)$, che può essere scritto come:

$P(A_1 = a_1, ..., A_{|A|} = a_{|A|}|C = c_j) =$

$P(A_1 = a_1|A_2 = a_2, ..., A_{|A|} = a_{|A|}, C = c_j)P(A_2 = a_2, ..., A_{|A|} = a_{|A|}|C = c_j)$

Ricorsivamente questo vale per il secondo, terzo termine e così via.

Assunzione di Indipendenza: Assumiamo che tutti gli attributi siano indipendenti data la classe $C = c_j$. Formalmente, assumiamo che

$$P(A_1 = a_1|A_2 = a_2, ..., A_{|A|} = a_{|A|}, C = c_j) = P(A_1 = a_1|C = c_j)$$

Quindi avremo che:

$$P(C = c_j|A_1 = a_1|A_2 = a_2, ..., A_{|A|} = a_{|A|}) = \frac{P(C=c_j)}{P(A_1=a_1|C=c_j)}$$

Abbiamo poi bisogno di stimare la probabilità a priori $P(C = c_j)$ e la probabilità condizionata $P(A_i = a_i|C = c_j)$ dal training data.

$$\Pr(C = c_j) = \frac{\text{number of examples of class } c_j}{\text{total number of examples in the data set}}$$

$$\Pr(A_i = a_i | C = c_j) = \frac{\text{number of examples with } A_i = a_i \text{ and class } c_j}{\text{number of examples of class } c_j}.$$

Quindi per decidere la classe con più alta probabilità per ogni istanza di test è necessario calcolare:

$$c = \arg\max_{c_j} \Pr(C = c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

Si può osservare che calcolare le probabilità richiede la costruzione di un classificatore Naive Bayesian che può essere trovato analizzando una sola volta il dataset, questo implica che l'algoritmo sia lineare nel numero degli esempi di training. Questa risulta essere la forza del naive bayes, essere estremamente efficente. In termini di classification accuracy, invece, l'algoritmo fa una forte assunzione di indipendenza. Per comprendere praticamente i classificatori naive Bayesian è necessario fare qualche altra osservazione: come gestire gli attributi numerici, gli zeri, e i valori mancanti.

Attributi Numerici: La formulazione di un apprendimento Naive Bayesian assume che tutti gli attributi siano categorici. Quindi ogni attributo numerico deve essere discretizzato in intervalli.

Valori Nulli: è possibile che un particolare valore di un attributo nel test set non abbia altre occorrenze nella classe. Questo risulta problematico in quanto porta ad avere una probabilità nulla che viene poi moltiplicata per tutte le altre probabilità (portando il prodotto a 0). La soluzione principale prevede di incorporare un piccolo valore di correzione su tutte le probabilità. Sia n_{ij} il numero di esempi che hanno $A_i = a_i$ e $C = c_j$, sia n_j il numero totale di esempi che hanno $C = c_j$ nel training data. La stima incorretta di $P(A_i = a_i|C = c_j)$ è uguale a $\frac{n_{ij}}{n_j}$, e la stima corretta è:
$P(A_i = a_i|C = c_j) = \frac{n_{ij}+\lambda}{n_j+m_i\lambda}$ dove m_i è il numero di attributi dell'attributo A_i e λ è un fattore moltiplicativo che è di solito $\lambda = 1/n$, dove n è il numero totale di esempi nel training set D. La forma generale di correzione viene chiamata **smoothing**.

Valori Mancanti: Questi possono essere tranquillamente ignorati.

4.1 Naive Bayesian Text Classification

La text classification o categorization prevede l'apprendimento di un modello di classificazione a partire dai documenti del training set categorizzati con delle predefinite classi. Quindi il modello appreso viene usato per classificare i futuri documenti. Anche se i modelli presentati in precedenza possono essere applicati a problemi di classificazione del testo vedremo come i prossimi modelli rispecchiano meglio le caratteristiche dei documenti, verranno presentati dei modelli di apprendimento Naive Bayesian che usano delle caratteristiche specifiche dei testi.

5 Probabilistic Framework

Il modello di apprendimento Naive Bayesian è generato a partire da un modello generativo probabilistico. Si assume che ogni documento sia generato secondo una qualche distribuzione di probabilità governata da dei parametri nascosti. Il training data viene quindi usato per stimare questi paramentri. I parametri sono usati per classificare ogni documento testuale usando la regola di Bayes calcolando la probabilità a posteriori la quale, insieme alla distribuzione di probabilità associata ad una classe, ha generato il documen-

to in questione. Infine la classificazione si riduce a selezionare la classe con probabilità maggiore per il dato documento. Il modello generativo è basato su 2 assunzioni:

1. I dati (documenti) sono generati da modelli misti. (es. diverse gaussiane sovrapposte)

2. Vi è una corrispondenza uno-a-uno tra le componenti miste e le classi.

Un **modello misto** modella i dati con un certo numero di distribuzioni di probabilità. Intuitivamente ogni distribuzione corrisponde ad un determinato cluster e i paramentri della destribuzione forniscono una descrizione del cluster corrispondente. Ogni distribuzione di un modello misto è detta **componente mista**. Sia K il numero di componenti miste in un modello misto, e θ_j i parametri dell j-esima distribuzione, sia Θ l'insieme dei parametri di tutte le componenti $\Theta = \{\rho_1, \rho_2, ..., \rho_k, \theta_1, \theta_2, ..., \theta_k\}$ dove ρ_j è il peso (probabilità) della componente mista j. Le probabilità delle componenti miste sono tali che $\sum_{j=1} \rho_j = 1$. Mostriamo ora come un modello misto sia in grado di generare una collezione di documenti. Ricordando l'insieme delle classi C del problema di classificazione dove $C = \{c_1, c_2, ..., c_{|C|}\}$ e dato che abbiamo assunto che ci sia una corrispondenza uno-a-uno tra classi e componenti miste allora $|C| = k$ e la j-esima componente mista può essere rappresentata come la corrispondente classe c_j ed essere parametrizzata da θ_j. I pesi delle componenti miste sono le probabilità a priori delle classi corrispondenti $\rho_j = P(c_j|\Theta)$. Un modello misto genera un documento d_j come segue:

1. Seleziona una componente mista (classe) in accordo con le probabilità a priori.

2. dalla componente mista (c_j) genera un documento d_i in accordo con i propri parametri.

La probabilità che un documento d_i sia generato dal modello misto può essere scritta come la somma delle probabilità su tutte le componenti miste

$$\Pr(d_i \mid \Theta) = \sum_{j=1}^{|C|} \Pr(c_j \mid \Theta) \Pr(d_i \mid c_j; \Theta).$$

6 Naive Bayesian Model

Un documento testuale consiste di una serie di frasi e ogni frase consiste di una sequenza di parole, al fine di semplificare le sequenze di parole e le relazioni che intercorrono nei modelli bayesiani vengono fatte alcune assunzioni. In particolare cio che viene fatto nei classificatori Naive Bayesian è considerare i documenti come "bag" of word (insieme di parole).

1. Le parole di un documento sono indipendenti dal resto del documento, ossia sono indipendenti da tutte le altre parole nel documento date le categorie delle classi.

2. La probabilità di una parola è indipendente dalla posizione in cui questa si trova all'interno del documento.

3. La lunghezza del documento è indipendente dalla classe a cui questo appartiene.

Usando queste assunzioni un documento può essere considerato come generato da una distribuzione multinomiale. Ossia, ogni documento è dato da una distribuzione multinomiale sulla parole con tanti processi indipendenti quanti la lunghezza del documento. Un **processo multinomiale** è un processo che può arrivare in ogni k possibile outcome, dove $k >= 2$. Le probabilità dei k outcome sono denotate come $p_1, p_2, ..., p_k$. Un esempio di processo multinomiale è il lancio di un dato a 6 facce, il quale ha 6 possibili outcome e dove, per un dato equo, si ha che $p_1 = p_2 = ... = p_6 = 1/6$.
Ora assumiamo n processi indipendenti ognuno con k possibili outcome collegati alle rispettive k probabilità. Per ogni outcome denotiamo con X_t la variabile aleatoria che conta il numero di processi che risultano in quel outcome, allora la collezione $X_1, X_2, ..., X_k$ è detta avere una distribuzione multinomiale di parametri $n, p_1, p_2, ..., p_k$.
Nel Nostro constesto n corrisponde alla lunghezza del documento e gli outcome corrispondono alle parole nel vocabolario. La probabilità p_k corrisponde alla probabilità di occorrenza delle parole nel vocabolario nel documento che è $P(w_t|c_j, \Theta)$. X_t è la variabile aleatoria che rappresenta il numero di volte che una parola w_t appare in un documento. Possiamo quindi applicare la funzione di probabilità della distribuzione multinomiale per trovare la probabilità di un documento data la sua classe.

7 Distance, similarity and classification

7.1 Rocchio

Rocchio è uno tra i piu semplici metodi di classificazione supervisionata di testi, dove:

- I documenti sono pesati usando una funzione standard, chiamata $tf \cdot idf$
- I vettori delle categorie, $C_1, ..., C_n$ sono ottenuti osservando gli esempi di training in media.

Abbiamo bisogno di definire una funzione peso $\omega(w, d)$ per una singola parola w in un documento d e un metodo per progettare un vettore catogoria come combinazione lineare dei vettori dei documenti. Una volta ottenuti i vettori documenti e i profili di categorie (C_i) è possibile inferire l'appartenenza dei documenti alle categorie usando la cosine similarity, un documento d è assegnato ad una categoria C_i se $(d, C_i) > \pi_i$.

Ogni termine w in un documento d, feature f, riceve un peso nel vettore corrispondente al documento d in accordo al numero di occorrenze di w nel documento stesso e in tutti gli altri documenti della collezione. Il peso di una parola w in un documento d è definito come:

$$\omega(w, d) = \omega_w^d = o_w^d log \frac{N}{N_w}$$

dove:

- N è il numero totale di documenti
- N_w il numero di documenti che contengono la parola w
- o_w^d è il numero di occorrenze della parola w nel documento d

La funzione peso $\omega(w, d)$ come definita sopra viene chiamata $tf \cdot idf$.

tf_w^d = **term frequency** = o_w^d pone in rilievo i termini che risultano molto rilevanti per un documento. Nella versione normalizzata si ha:

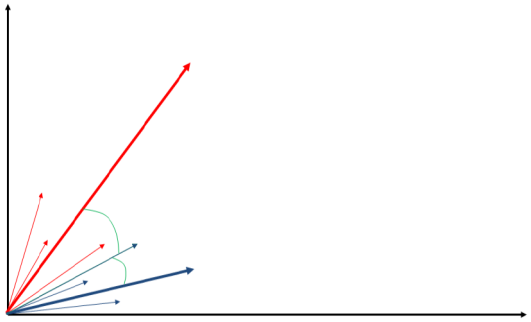
$$tf_w^d = \frac{o_w^d}{max_{x \in d} o_x^d}$$

idf_w = **inverse document frequency** = $log\frac{N}{N_w}$ pone in rilievo solo i termini che non sono relativamente troppo frequenti nel corpus andando a scartare le parole comuni che non caratterizzano nessun sottoinsieme specifico della collezione. Infatti quando una parola w è presente in ogni documento allora $N = N_w$ e quindi $idf_w = log\frac{N}{N_w} = 0$

L'ultimo step prevede di fornire un profilo geometrico per la categorizzazione del testo in modo da avere una rappresentazione della classe C_i. Una parola w ha un peso $\Omega(w, C_i)$ in un vettore catogoria C_i di un documento che è definito come:

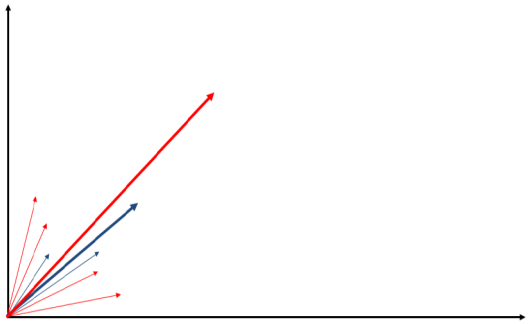
$$\Omega(w, C_i) = \Omega_w^i = max\{0, \frac{\beta}{|T_i|}\sum_{d \in T_i}\omega_w^d - \frac{\gamma}{|T_i'|}\sum_{d \in T_i'}\omega_w^d\}$$

dove T_i è l'insieme dei documenti di training classificati nella classe C_i e T_i' è l'insieme dei documenti non classificati nella classe C_i. I vettori delle categorie e dei documenti sono derivati a partire dai pesi assegnati a tutte le parole del dizionario di una data collezione. Una parola viene aggiunta al volabolario se viene vista in almeno un documento, altrimenti si possono utilizzare varie tecniche di feature selection. Dati i vettori di training di due classi i profili di categorie descrivono un vettore risultante che è la media dei comportamenti dei vettori del training set corrispondenti a quella classe, dato un nuovo vettore questo verrà classificato in una classe rispetto ad un'altra in base a quale di queste massimizza la cosine similarity. In pratica si ha che il nuovo vettore documento d al quale si vuole assegnare la classe C_1 o C_2 verrà asseganto alla classe i che soddisfa questa equazione $(d, C_j) < (d, C_i)$.



Questa procedura soffre però del problema del polimorfismo per il quale si ha il rischio di incorrere in una svrapposizione delle classi che risulta in una errata classificazione, questo

è dovuto ad una perdita di memoria sui vettori che hanno portato alla costruzione del profilo di categoria corrispondente.



7.2 Memory-based Learning: k-NN

L'apprendimento fatto tramite memory-based learning si riduce al memorizzare le rappresentazioni del training set in una collezione T. Ciò che si vuole fare quindi è utilizzare l'intero insieme di dati per generare direttamente predizioni, senza la costruzione di alcun modello. Una possibile applicazione dell'algoritmo k-NN può essere quella di voler fornire dei contenuti a utenti in base alle loro preferenze, al loro passato o al passato di profili utenti che in qualche modo gli somigliano. Ciò che si vuole fare è quindi trovare i k vicini più prossimi al dato (profilo utente) quindi aggregare in qualche modo questi valori.

L'algoritmo:

1. Per ogni esempio nel training set $< x, c(x) > \in D$
   (a) Calcola il vettore $tf - idf$, x', per il documento x.
2. Data l'istanza di test y: calcola il vettore $tf - idf$, y', per y.
3. Per ogni $< x, c(x) > \in D$:
$$s_x = cosSim(y', x') = \frac{(x', y')}{||x'|| \cdot ||y'||}$$
4. Ordina gli esempi $x \in D$ per valori decrescenti di s_x
5. Sia kNN l'insieme dei primi k esempi in D

6. RETURN la classe che ha più esempi in kNN

8 Similarity

In molti casi i dati sono rappresentati come vettori di grandi dimensioni, caratterizzati da una matrice molto sparsa di termini e documenti. In molte situazioni quindi si ha a che fare con il problema della dimensionalità che, anche dopo una adeguada feature reduction, non è risolto. Un modo per diminuire la dimensionalità consiste nel clustering un complesso processo di ricerca di insiemi tra tutti i sottoinsiemi possibili. Per adoperare un processo di clustering è necessario eseguire i seguenti step:

- Per ottenere le features $X \in F$ dagli oggetti "grezzi", bisogna trovare una rappresentazione adeguata degli oggetti stessi.
- Dato un oggetto $O \in D$, ci riferiamo a tale rappresentazione come il vettore delle features x di X.
- E' necessario trovare una misura di distanza $s \in S$ tra gli oggetti, per esempio $s : D2 \to R$. **La scelta della similarità o distanza può avere un impatto non trascurabile sulla qualità del clustering**.

La distanza di Minkowski $L_p(\underline{x}, \underline{y})$ definita come:

$$L_p(\underline{x}, \underline{y}) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$$

è una metrica standard per problemi geometrici. Inoltre per $p = 2$ otteniamo la distanza Euclidea:

$$d(\underline{x}, \underline{y}) = ||\underline{x} - \underline{y}||_2^2$$

Ora dobbiamo mettere in relazione la distanza d con la similarità s. Per lo spazio Euclideo si può fare in questo modo:

$$s = e^{-d^2}$$

Allora, la **similarità normalizzata** $\in [0, 1]$ diventa:

$$s^{(E)}(\underline{x}, \underline{y}) = e^{-||\underline{x}-\underline{y}||_2^2}$$

Introduciamo anche la Similarità Estesa di Jaccard definita come:

$$s^{(J)}(\underline{x}, \underline{y}) = \frac{\underline{x}^T \underline{y}}{||\underline{x}||_2^2 + ||\underline{y}||_2^2 - \underline{x}^T \underline{y}}$$

La similarità Euclidea è invariante rispetto alla traslazione, ma sensibile alla "scalatura", mentre il coseno esattamente il contrario. Jaccard esteso invece ha entrambe le proprietà come illustrato in figura (linee di similarità con $s = 0.25, 0.5$ e 0.75 per i punti $\underline{x} = (3, 1)^T$ e $\underline{y} = (1, 2)^T$ usando Euclide, coseno e Jaccard esteso).

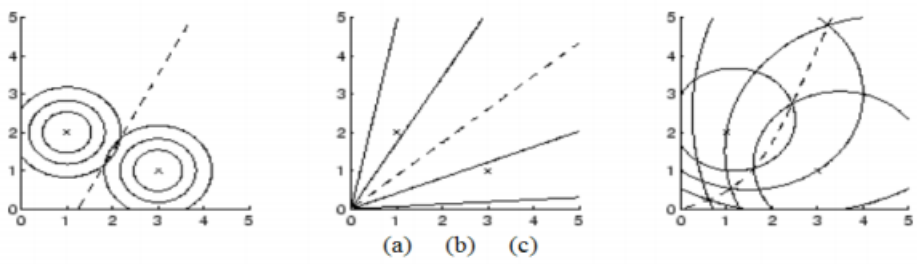


Figure 4.1: Properties of (a) Euclidean-based, (b) cosine, and (c) extended Jaccard similarity measures illustrated in 2 dimensions. Two points $(1, 2)^\dagger$ and $(3, 1)^\dagger$ are marked with $\times$ s. For each point iso-similarity surfaces for $s = 0.25$, 0.5, and 0.75 are shown with solid lines. The surface that is equi-similar to the two points is marked with a dashed line.

Quindi per $s^{(J)} \to 0$ Jaccard esteso si comporta come il coseno e per $s^{(J)} \to 1$ come la distanza Euclidea.

9 Probability and language modeling

Le strutture linguistiche sono esempi di strutture dove l'informazione sintagmatica (strutturata, grammatiche di chomsky) risulta cruciale per il machine learning, La modellizzazione più usata risulta essere la grammatica. Come sappiamo la modellizzazione effettuata tramite grammatiche può portare alla generazione di varie ambiguità, per questo entrano in gioco le grammatiche pesate. Nella grammatiche pesate si calcolano i gradi di ambiguità collegati a differenti derivazione, usando le probabilità associate a queste. Una frase ambigua potrebbe essere: "Imposta pesante". La figura sottostante è una tabella che mostra le probabilità associate ad ogni produzione di una possibile grammatica di interpretazione della frase.

```
1.    S   -> NP  V        .7
2.    S   -> NP           .3
3.    NP  -> PN           .1
4.    NP  -> N            .6
5.    NP  -> Adj N        .3
6.    N   -> imposta      .6
7.    V   -> imposta      .4
8.   Adj  -> Pesante      .8
9.    PN  -> Pesante      .2
...
```

Ci chiediamo qual è la probabilità che "imposta" sia verbo e "pesante" complemento oggetto.

$$prob(((pesante)_{PN}(imposta)_V)_S) = (.7 \cdot .1 \cdot .2 \cdot .4) = 0,0084$$

Ci chiediamo qual è la probabilità che "imposta" sia nome e "pesante" aggettivo.

$$prob(((pesante)_{Adj}(imposta)_N)_S) = (.3 \cdot .3 \cdot .8 \cdot .6) = 0,0432$$

In questo modo quindi da una frase che ha 2 interpretazioni differenti riusciamo ad assegnare un peso che ci aiuta a decidere quali delle due diverse classificazioni assegnare alla frase.

Gli obiettivi della Probability language modeling sono:

- estendere i modelli grammaticali con capacità predittive e di disambiguazione
- offrire metodi induttivi teoricamente ben fondati
- sviluppare modelli quantitativi di fenomeni linguistici

Metodi e risorse:

- teorie matematiche (per esempio modelli di Markov)
- freamworks di verifica/valutazione sistematici
- grandi repository di esempi della lingua in uso
- risorse linguistiche tradizionali (dizionari)

9.1 HMM for modeling linguistic tasks

Ci sono molti vantaggi nel modellare un problema linguistico come un HMM

- è un potente freamwork matematico di modellizzazione
- fornisce chiare impostazioni per differenti applicazioni: **estimation**, **decoding**, **model induction**
- modelli basati su HMM offrono soluzioni efficaci per le applicazioni di cui sopra

Una HMM può essere rappresentata tramite un modello $\lambda = (E, T, \pi)$, dove T è la matrice nxn di transizione tra n stati, E è la matrice nxk di emissione dei k simboli (per convezione assumiamo che la HMM emetta un simbolo all'entrata in uno stato) e π è il vettore 1xn delle probabilità che lo stato iniziale sia uno tra $i \in 1, 2, .., n$.
La complessità del training e decoding può essere limitata tramite l'uso di tecniche di ottimizzazione:

- **Language Modeling:** Data una sequenza di osservazioni $O = O_1, O_2, ..., O_n$ e un modello $\lambda = (E, T, \pi)$, si vuole calcolare efficientemente $P(O|\lambda)$ si utilizzano algoritmi di programmazione dinamica (**forward algorithm**) ($O(n)$)

- **Tagging/Decoding:** Data una sequenza di osservazioni $O = O_1, O_2, ..., O_n$ e un modello $\lambda = (E, T, \pi)$, si vuole calcolare la sequenza di stati ottima $Q = q_1, q_2, ..., q_n$ per la generazione di O, si utilizzano algoritmi di programmazione dinamica (**Viterbi**) ($O(n)$)

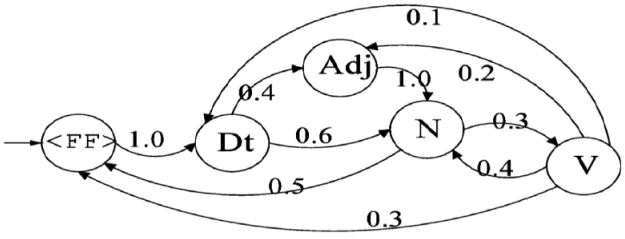
- **Model Induction:** Si vogliono trovare i parametri migliori di $\lambda = (E, T, \pi)$ per massimizzae la probabilità $P(O|\lambda)$, si utilizza la stima dei parametri tramite minimizzazione dell'entropia (**BM**)

9.1.1 POS Tagging task

Data una sequenza di morfemi $w_1, w_2, ..., w_n$ con una descrizione ambigua della sintassi t_j, si calcola la sequenza di n POS tags $t_{j1}, t_{j2}, ..., t_{jn}$ che caratterizzano le corrispondenti parole w_i.

Examples:

- *Secretariat is expected to race tomorrow*
- ⇒ NNP VBZ VBN TO VB NR
- ⇒ NNP VBZ VBN TO NN NR



Emission probabilities

	.	the	this	cat	kid	eats	runs	fish	fresh	little	big
<FF>	1.0										
Dt		0.6	0.4								
N				0.6	0.1			0.3			
V						0.7	0.3				
Adj									0.3	0.3	0.4

9.1.2 HMM and POS tagging

Inserire testo qui.

10 SVM - Support Vector Machines

Le Support Vector Machines sono un altro tipo di sistema di apprendimento, le quali hanno delle buone qualità che le rendono tra gli algoritmi più popolari. Risultano molto utili quando si trattano dati di dimensionalità elevata. Le **SVM** sono un **sistema di apprendimento lineare** che costruisce un classificatore per due classi. Sia l'insieme di esempi di training D

$$(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$$

dove $x_i = (x_{i1}, x_{i2}, ..., x_{ir})$ è un vettore r-dimensionale (valore di input) in uno spazio $X \subseteq R^r$, e dove y_i è il label della classe (valore di output) e $y_i \in \{1, -1\}$. 1 denota la classe positiva mentre -1 la classe negativa. Per costruire un classificatore l'SVM trova una funzione lineare della forma:

$$f(x) = (w \cdot x) + b$$

Un vettore di input x_i è asseganto alla classe positiva se $f(x_i) >= 0$ e alla classe negativa altrimenti. $f(x)$ è una funzione sui valori reali $f : X \subseteq R^r -> R$. $w = (w_1, w_2, ..., w_r) \in R^r$ è chiamato vettore peso. $b \in R$ viene chiamto bias, $(w \cdot x)$ è il prodotto di w e x. La funzione $f(x)$ puo essere scritta come:

$$f(x_1, x_2, ..., x_r) = w_1x_1, +w_2x_2 + ... + w_rx_r + b$$

Dove x_i è la variabile che rappresenta l'i-esima cordinata del vettore x. In sostanza, un SVM trova un **iperpiano**

$$(w \cdot x) + b = 0$$

che separa gli esempi di training positivi dai negativi. Questo iperpiano viene chiamato confine o superficie di decisione. Geometricamente, un iperpiano $(w \cdot x) + b = 0$ divide lo spazio di input in due: una metà per gli esempi positivi e l'altra per i negativi. In uno spazio 2-dimensionale l'iperpiano è comunemente chiamato retta mentre in uno spazio

3-dimensionale viene chiamato piano.

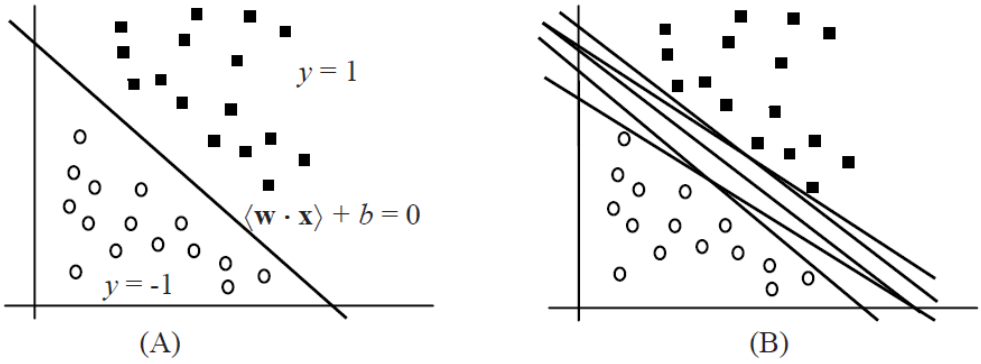


Fig. 3.19. (A) A linearly separable data set and (B) possible decision boundaries

10.1 Linear SVM: Separable Case

in questa sezione si studia il caso semplice di un SVM lineare. Si assuma che i punti positivi e negativi siano separabili linearmente. in $(w \cdot x) + b = 0, w$ definisce la direzione perpendicolare all'iperpiano. w è anche chaimato vettore normale (o semplicemente normale) dell'iperpiano. Senza cambiare il vettore normale w, variando b si muove l'iperpiano pararrelamente a se stesso. Si noti che $(w \cdot x) + b = 0$ ha un grado di libertà intrinseco. Possiamo scalare l'iperpiano con $(\lambda w \cdot x) + \lambda b = 0$ per $\lambda \in R^+$, senza cambiare la funzione/iperpiano.

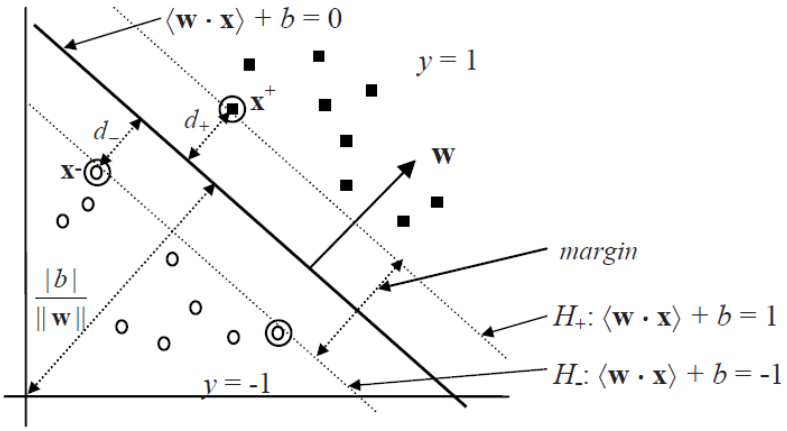


Fig. 3.20. Separating hyperplanes and margin of SVM: Support vectors are circled

Dal momento che le SVM massimizzano la distanza sia dagli esempi esempi positivi che negativi, definiamo il margine. Siano d_+ e d_- le più piccole distanze tra piano di saparazione e, rispettivamente, il piu vicino punto positivo e il più vicino punto negativo. Il **margine** dell'iperpiano di separazione è $d_+ + d_-$. Una SVM cerca l'iperpiano di separazione con il margine maggiore, anche chiamato iperpiano di margine massimo. La ragione per la quale si sceglie questo tipo di iperpiani viene da risultati teorici che mostrano come la massimizzazione del margine riduca l'errore durante la classificazione. Consideriamo un punto positivo $(x^+, 1)$ e un punto negativo $(x^-, -1)$ i più vicini all'iperpiano $(w \cdot x) + b = 0$. Definiamo due iperpiani paralleli H_+, H_- che passano rispettivamente in x^+ e in x^-. Possiamo riscalare w e b ottenendo:

$$H_+ : (w \cdot x^+) + b = 1$$

$$H_- : (w \cdot x^-) + b = -1$$

tale che:

$$(w \cdot x_i) + b \geq 1, y = 1$$

$$(w \cdot x_i) + b \leq -1, y = -1$$

che indica che nessun dato di training giace tra i due iperpiani. La distanza tra i due iperpiani è il margine $(d_+ + d_-)$. La distanza tra un punto x_i e un iperpiano $(w \cdot x) + b = 0$ è:

$$\frac{|(w \cdot x) + b|}{||w||}$$

dove $||w||$ è la norma euclidea di w:

$$||w|| = \sqrt{(w \cdot w)} = \sqrt{w_1^2 + w_2^2 + ... + w_r^2}$$

Per calcolare d_+, invece di calcolare la distanza di x^+ dall'iperpiano di separazione, prendiamo ogni punto x_s in $(w \cdot x) + b = 0$ e calcolare la distanza da x_s a $(w \cdot x^+) + b = 1$,

$$d_+ = \frac{|(w \cdot x) + b - 1|}{||w||} = \frac{1}{||w||}$$

Lo stesso discorso vale per d_-, da cui:

$$margine = d_+ + d_- = \frac{2}{||w||}$$

Dato che SVM cerca un iperpiano di separazione che massimizzi il margine, questo porta ad un problema di ottimizzazione. Massimizzare il margine risulta uguale a minimizzare $\frac{||w||^2}{2} = \frac{(w \cdot w)}{2}$.

Definizione SVM nel caso separabile: dato un insieme di esempi di training separabili linearmente,

$$D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$$

l'apprendimento consiste nel risolvere il seguente problema di minimizzazione:

$$min : \frac{(w \cdot w)}{2}$$

$$vincoli : y_i((w \cdot x) + b) \geq 1, i = 1, 2, ..., n$$

si noti che per il vincolo $y_i((w \cdot x) + b) \geq 1, i = 1, 2, ..., n$ si ha:

$$(w \cdot x_i) + b \geq 1, y_i = 1$$

$$(w \cdot x_i) + b \leq -1, y_i = -1$$

Invece di ottimizzare la sola funzione obiettivo, abbiamo bisogno di ottimizzare anche la lagrangiana del problema, la quale considera allo stesso tempo anche i vincoli.

$$L_p = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{n} \alpha_i[y_i((w \cdot x_i) + b) - 1] \tag{45}$$

dove $\alpha_i \geq 0$ sono i moltiplicatori di lagrange. La teoria dell'ottimizzazione ci dice che una soluzione ottima deve soddisfare delle condizioni, chiamate condizioni di **Kuhn-tucker**, che giocano un ruolo centrale nell'ottimizzazione dei vincoli. Sia un generico problema di ottimizzazione:

$$min : f(x)$$

$$vincoli : g_i(x) \leq b_i, i = 1, 2, ..., n$$

dove $f(x)$ è la funzione obiettivo e $g(x)$ la funzione dei vincoli, la lagrangiana corrispondente è:

$$L_p = f(x) + \sum_{i=1}^{n} \alpha_i[g_i(x) - b_i]$$

Una soluzione ottima al problema deve soddisfare le seguenti condizioni necessarie (ma non sufficienti):

$$\frac{\partial L_P}{\partial x_j} = 0, \;\; j = 1, 2, ..., r$$
$$g_i(\mathbf{x}) - b_i \leq 0, \;\; i = 1, 2, ..., n$$
$$\alpha_i \geq 0, \;\; i = 1, 2, ..., n$$
$$\alpha_i (b_i - g_i(\mathbf{x}_i)) = 0, \;\; i = 1, 2, ..., n$$

Queste condizioni sono chiamate condizioni di **Kuhn-tucker**. Tornando al nostro problema le condizioni di Kuhn-Tucker sono:

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{n} y_i \alpha_i x_{ij} = 0, \;\; j = 1, 2, ..., r \tag{52}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{n} y_i \alpha_i = 0 \tag{53}$$

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 \geq 0, \;\; i = 1, 2, ..., n \tag{54}$$

$$\alpha_i \geq 0, \;\; i = 1, 2, ..., n \tag{55}$$

$$\alpha_i (y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1) = 0, \;\; i = 1, 2, ..., n \tag{56}$$

La disequazione (54) è l'insieme dei vincoli originali. Esiste inoltre un moltiplicatore di lagrange α_i per ogni punto di training, la considizione complementare (56) ci mostra che solo i punti negli iperpiani di margine (p.e. H_+ e H_-) possono avere $\alpha_i \geq 0$ per i quali $y_i((w \cdot x_i) + b) - 1 = 0$. Questi punti vengono chaimati **support vectors**. Tutti gli altri punti hanno $\alpha_i = 0$. Per il nostro problema di minimizzazione le condizoni di Kuhn-Tucker risultano essere necessari e sufficienti per una soluzione ottima.

Risolvere il problema di ottimizzazione risulta essere un task ancora troppo complicato ma è possibile dare la formulazione di un problema duale alternativo che risulta essere di più semplice risoluzione, il problema originale viene chiamato problema primale. Nell'ambito delle SVM il problema duale non è solo di più semplice risoluzione ma risulta cruciale per l'utilizzo delle **funzioni Kernel** al fine di operare con limiti decisionali non lineari poichè non è necessario esplicitare **w**. Trasformare il primale in duale può essere fatto ponendo a zero le derivate parziali della lagrangiana (45) in riferimento alle variabili

primali (p.e. w e b) e sostituendo la relazione risultante nella lagrangiana. Questo viene fatto sostituendo semplicemente la (52) e la (53) nella Lagrangiana originale (45) per eliminare le variabili primali, ottenendo la funzione obiettivo duale:

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{59}$$

L_D contiene solo variabili duali e può essere massimizzato sotto vincoli semplici, (52), (53) e $\alpha_i \geq 0$. La funzione duale della primale (44) risulta:

$$max : L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)$$

$$vincoli : \sum_{i=1}^{n} y_i \alpha_i = 0, \alpha_i \geq 0, i = 1, 2, ..., n$$

Questa formulazione viene chiamata **Wolf dual**. La nosta funzione obiettivo ha quindi la proprietà che i massimi delle α_i in L_D portano ad avere dei minimi di w e b in L_P. Il nostro limite di decisione finale (l'iperpiano di margine massimo) è:

$$(w \cdot x) + b = \sum_{i \in sv} y_i \alpha_i (x_i \cdot x) + b = 0$$

dove sv è l'insieme degli indici dei suppor vectors.

Testing: Dato un'istanza di test z, la classifichiamo usando:

$$sign((w \cdot z) + b) = sign(\sum_{i \in sv} y_i \alpha_i (x_i \cdot z) + b)$$

Se viene restituito 1 allora z viene classificato come positivo in caso contrario come negativo.

10.2 Linear SVM: Non-separable Case

Il caso separabile lineramente risulta essere una situazione ideale. In pratica i dati di training possono presentare del disturbo per varie ragioni. Ad esempio, alcuni dati potrebbero essere etichettati in modo errato. Per rendere le SVM utilizzabili è necessario che permettano la presenza di rumore nei dati. Nel caso di rumore però, le SVM del caso

separabile non riuscirebbero a trovare alcuna soluzione in quanto i vincoli imposti non sarebbero rispettati. Per permettere la presenza di errori nei dati, possiamo rilassare i vincoli del margine introducendo una variabile slack, $\xi_i(\geq 0)$ come segue:

$$(w \cdot x_i) + b \geq 1 - \xi_i \ for \ y_i = 1$$

$$(w \cdot x_i) + b \leq -1 + \xi_i \ for \ y_i = -1$$

Da cui:

$$vincoli : y_i((w \cdot x_i) + b) \geq 1 - \xi_i, \ \xi_i \geq 0, \ i = 1, 2, ..., n$$

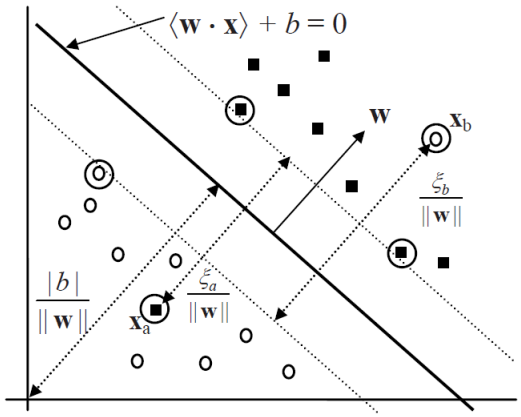


Fig. 3.21. The non-separable case: $\mathbf{x}_a$ and $\mathbf{x}_b$ are error data points

Abbiamo in oltre bidogno di penalizzare gli errori nella funzione obiettivo. Un modo naturale di farlo è quello di assegnare un costo extra per gli errori cambiando la funzione obiettivo in:

$$Min : \frac{(w \cdot w)}{2} + C(\sum_{i=1}^{n} \xi_i)^k \tag{64}$$

dove $C \geq 0$ è un parametro specificato dall'utente, spesso viene usato $k = 1$, in quanto ne le ξ_i ne i moltiplicatori lagrangiani appaiono nella formulazione del duale. Il nuovo problema di ottimizzazione diventa:

$$Min : \frac{(w \cdot w)}{2} + C \sum_{i=1}^{n} \xi_i \tag{64}$$

$$Vincoli : y_i((w \cdot x_i) + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i = 1, 2, ..., n \tag{64}$$

Questa formulazione viene chiamata **soft-margin SVM**. La Lagrangiana primale di questa formulazione è come segue:

$$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] - \sum_{i=1}^{n} \mu_i \xi_i , \tag{66}$$

dove $\alpha_i, \mu_i \geq 0$ sono i moltiplicatori di Lagrange.
Le condizioni di Kuhn-Tucker per l'ottimalità diventano:

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{n} y_i \alpha_i x_{ij} = 0,\ j = 1, 2, ..., r \tag{67}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{n} y_i \alpha_i = 0 \tag{68}$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0,\ \ i = 1, 2, ..., n \tag{69}$$

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i \geq 0,\ \ i = 1, 2, ..., n \tag{70}$$

$$\xi_i \geq 0,\ \ i = 1, 2, ..., n \tag{71}$$

$$\alpha_i \geq 0,\ \ i = 1, 2, ..., n \tag{72}$$

$$\mu_i \geq 0,\ \ i = 1, 2, ..., n \tag{73}$$

$$\alpha_i(y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i) = 0,\ \ i = 1, 2, ..., n \tag{74}$$

$$\mu_i \xi_i = 0,\ \ i = 1, 2, ..., n \tag{75}$$

Come nel caso separabile, trasformiamo il problema primale nel suo duale fissando a 0 le derivate parziali della Lagrangiana (66) rispetto alle variabili primali (p.e. w, b e ξ_i), sostituendo poi le relazioni risultanti nella Lagrangiana. Quindi sostituiamo le equazioni (67), (68) e (69) nella Lagrangiana primale (66). Dall'equazione (69), $C - \alpha_i - \mu_i = 0$, possiamo dedurre che $\alpha_i \leq C$ poichè $\mu_i \geq 0$. Allora, la duale della (65):

$$Max : L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) \tag{76}$$

$$Vincoli : \sum_{i=1}^{n} y_i \alpha_i = 0,\ 0 \leq \alpha_i \leq C,\ i = 1, 2, ..., n \tag{76}$$

E' interestante notare che ξ_i e i suoi moltiplicatori di Lagrange μ_i non sono nel duale e la funzione obiettivo è identica al caso separabile. L'unica differenza sta nel vincolo $\alpha_i \leq C$. Il problema duale (76) può essere risolto numericamente e l'α_i risultante usata per calcolare w e b. w viene calcolato usando l'equazione (67) mentre b viene calcolato usando le condizioni (74) e (75) di Kuhn-Tuker.

Dalle equazioni (69), (74) e (75) osserviamo che se $0 \leq \alpha_i \leq C$ allora si ha $\xi_i = 0$ e $y_i((w \cdot x) + b) - 1 - \xi_i = 0$.

Allora, possiamo usare un qualsiasi punto di training per il quale $0 \leq \alpha_i \leq C$ e l'equazione (74)(con $\xi_i = 0$) per calcolare b:

$$b = \frac{1}{y_i} - \sum_{i=1}^{n} y_i \alpha_i (x_i \cdot x_j) \tag{77}$$

Ancora una volta, a causa degli errori numerici, possiamo calcolare tutti i possibili b e prenderne la media come valore finale. E' importante notare che:

$$\begin{array}{lcl} \alpha_i = 0 & \Rightarrow & y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 \text{ and } \xi_i = 0 \\ 0 < \alpha_i < C & \Rightarrow & y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) = 1 \text{ and } \xi_i = 0 \\ \alpha_i = C & \Rightarrow & y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \leq 1 \text{ and } \xi_i \geq 0 \end{array} \tag{78}$$

Similmente ai support vectors del caso separabile la (78) ci mostra una delle proprietà più importanti delle SVM: La soluzione è sparsa negli α_i. La maggior parte de dati si trovano fuori dall'area di margine e i loro α_i nella soluzione sono 0. Solo i punti che si trovano sul margine (che sono i support vectors nel caso separabile), dentro il margine (p.e. $\alpha_i = C$ e $y_i((w \cdot x) + b) \leq 1$, o gli errori sono diversi da zero. Se non fosse sparsa, la SVM non potrebbe essere usata per dataset di grandi dimensioni.

Il limite di decisione finale è:

$$(w \cdot x) + b = \sum_{i=1}^{n} y_i \alpha_i (x_i \cdot x) + b = 0 \tag{79}$$

La regola di decisione per la classificazione risulta la stessa del caso separabile $sign((w \cdot x) + b)$. Sia per l'equazione (79) che (77) w non deve essere calcolato esplicitamente. Questo risulta cruciale nell'uso delle funzioni di kernel per gestire i limiti di decisione non lineari. Infine, abbiamo ancora il problema di determinare il parametro C. Il valore di

C viene solitamente scelto provando un intervallo di valori sul set di addestramento per costruire più classificatori e poi testarli su un validation set selezionando quello che dà il miglior risultato di classificazione durante la validation. Risulta spesso usata anche la tecnica di cross-validation.

10.3 Nonlinear SVM: Kernel Functions

Le formulazioni discusse precedentemente richiedono che gli esempi positivi e nagativi siano separabili linearmente, il limite di precisione deve essere un iperpiano. Tuttavia, in molti dati reali, il limite di decisione non risulta lineare. Per trattare dati non separabili linearmente vengono usate le stesse formulazioni e tecniche di soluzioni del caso lineare. Si spostano i dati di input dallo spazio originale in un altro spazio (solitamente di dimensionalità maggiore) in modo che si possano dividere linearmente gli esempi positivi dai negativi nel nuovo spazio, che viene chiamato **spazio caratteristico** (feature space). Lo spazio originale dei dati viene chiamato **spazio dell'input** (input space). Quindi, l'idea di base è quella di mappare i dati nello spazio di input X in uno spazio caratteristico F usando una mappatura non lineare ϕ,

$$\phi : X \to F \tag{80}$$

$$x \to \phi(x)$$

Dopo il mapping, il data set originale $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ diventa:

$$\{(\phi(x_1), y_1), (\phi(x_2), y_2), ..., (\phi(x_n), y_n)\} \tag{81}$$

Viene quindi applicato lo stesso metodo della soluzione dell'SVM lineare a F.

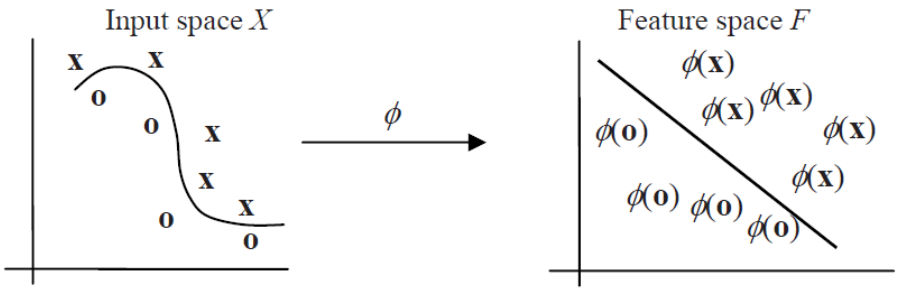


Fig. 3.22. Transformation from the input space to the feature space

Tramite la trasformazione, il problema (65) di ottimizzazione diventa

$$Min : \frac{w \cdot w}{2} + C \sum_{i=1}^{n} \xi_i \tag{82}$$

$$Vincoli : y_i((w \cdot \phi(x)) + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i = 1, 2, .., n,$$

e il corrispondente duale è:

$$Max : L_d = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j (\phi(x_i) \cdot \phi(x_j)) \tag{83}$$

$$Vincoli : \sum_{i=1}^{n} y_i \alpha_i = 0,\ 0 \leq \alpha_i \leq C,\ i = 1, 2, .., n,$$

La regola di decisione finale per la classificazione è:

$$\sum_{i=1}^{n} y_i \alpha_i (\phi(x_i) \cdot \phi(x)) + b \tag{84}$$

Gli esempi di training dello spazio di input devono essere trasformati in esempi di training nello spazio caratteristico. Il problema di questo approccio risiede nel fatto che traformare i dati di input esplicitamente nello spazio caratteristico e quinid applicare SVM lineare può soffrire del problema della dimensionalità. Fortunatamente, le trasformazioni esplicite possono essere evitate se si riesce a mostrare che nella rappresentazione duale sia la costruzione dell'iperpiano ottimo in F che la valutazione della corrispondente funzione di decisione/classificazione richiedono il solo calcolo del prodotto vettoriale $(\phi(x) \cdot \phi(z))$ senza dover mai calcolare esplicitamente il vettore $\phi(x)$. Quindi se abbiamo un modo di calcolare il prodotto vettoriale $\phi(x) \cdot \phi(z)$ nello spazio caratteristico F usando direttamente i vettori x e z, allora non abbiamo bisogno di conoscere il vettore $\phi(x)$ o la stessa funzione di mappatura ϕ. Nelle SVM, questo viene fatto usando le **funzioni kernel**, denotate con K.

$$K(x, z) = (\phi(x), \phi(z)) \tag{86}$$

la quale corrisponde esattamente alle funzioni per calcolare i prodotti vettoriali nello spazio caratteristico usando i vettori di input x e z. Un esempio di funzione kernel è il **kernel polinomiale**,

$$K(x, z) = (x \cdot z)^d. \tag{87}$$

Il numero di dimensioni nello spazio caratteristico per la funzione di kernel polinomiale $(x, z)^d$ è $(\frac{r+d-1}{d})$, che risulta essere un numero enorme anche con un numero ragionevole (r) di attributi nello spazio di input. Fortunatamente, usando la funzione di kernel (87), il numero di dimensioni nello spazio caratteristico non influisce. Quindi non abbiamo bisogno di trovare la funzione di mappatura ma, è necessario solo sostituire tutti i prodotti vettoriali $(\phi(x) \cdot \phi(z))$ in (83) e (84) con la funzione di kernel $K(x, z)$. La strategia di usare direttamente una funzione di kernel per sostituire i prodotti vettoriali nello spazio caratteristico viene chiamata **kernel trick**. Non abbiamo mai bisogno di conoscere ϕ. Il teorema di **Mercer** definisce delle condizioni sotto le quali possiamo essere sicuri che la funzione di kernel rappresenti il prodotto vettoriale nel nuovo spazio, prima di tutto la funzione di kernel deve essere necessariamente simmetrica. Inoltre, in uno spazio finito, la matrice di kernel (gram matrix) deve essere semi definita positiva, questo ci assicura che K è una funzione di kernel. L'idea è che il kernel generalizza il prodotto vettoriale nello spazio di input.

Kernel generalmente utilizzati sono:

$$Polinomiale: \ K(x, z) = ((x \cdot z) + \theta)^d \tag{90}$$

$$Gaussiana \ RBF: \ K(x, z) = e^{-||x-z||^2/2\sigma} \tag{91}$$

dove $\theta \in R, \ d \in N, \ e \ \sigma \geq 0$.

10.4 Summary

SVM è un sistema di apprendimento lineare che trova l'iperpiano con margine di decisione massimo per separare gli esempi positivi dai negativi. L'apprendimento è formulato come un problema di ottimizzazione quadratico. Limiti di decisione non lineari possono essere trovati operando una trasformazione dei dati originali in uno spazio caratteristico di dimensione maggiore. Tuttavia, questa trasformazione non viene mai fatta in modo esplicito. Invece, vengono usate funzioni di kernel per calcolare i prodotti vettoriali necessari all'apprendimento, senza bisogno di conoscere la funzione di mappatura. L'algoritmo di apprendimento e le funzioni di kernel possono essere studiati indipendentemente l'uno dall'altro. In questo modo si può progettare e sperimentare differenti funzioni di kernel

senza intaccare l'algoritmo di apprendimento sottostante.
Le SVM hanno alcune limitazioni:

1. Operano solo su spazi di valori reali. Per un attributo categorico, abbiamo bisogno di convertirlo in un valore numerico. Un modo di farlo è creare attributi aggiuntivi per ogni valore della categoria, settare il valore dell'attributo a 1 se appare il corrispondente valore della categoria e 0 altrimenti.

2. Ammette solo due classi. Per il problema della classificazione di classi multiple, possono essere applicate varie strategie, p.e. One-Against-All.

3. L'iperpiano prodotto dalla SVM è difficile da comprendere dall'uomo. Quindi, le SVM sono comunemente usate in applicazioni che non richiedono la comprensione umana.

11 Latent Semantic Analysis

11.1 Latent Semantic Indexing

Nei modelli di retrival sono spesso basati sulla connessione tra termini, ad esempio trovare documenti che rispondano a query dell'utente. Tuttavia alcuni concetti possono essere espressi in modi differenti ad esempio in base al contesto o per diverse abitudini degli utenti. Se una query di un utente è composta da parole differenti da quelle presenti in un documento questo non verrà restituito all'utente anche se si sarebbe potuto rivelare rilevante, ad esempio perchè conteneva sinonimi delle parole usate nella query. Questo causa una bassa recall. Ad esempio "picture", "photo", "image" sono sinonimi in un contesto di fotografia. Se la query dell'utente dovesse contenere solo la parola "picture" tutti i documenti in cui sono presenti solo "image" e "photo" non verrebbero restituiti. Il Latent semantic indexing (LSI) cerca di trattare questo problema identificando delle associazioni tra termini. Si assume che ci sia una struttura sottostante latente di semantica nei dati che viene nascosta dalla randomicità delle scelte delle parole. Viene usata una tecnica statistica, chiamata **singular value decomposition** (SVD), per stimare questa

struttura latente e per rimuovere il "disturbo". Questa struttura viene anche chiamata spazio concettuale nascosto, che associa documenti e termini differenti a livello di sintassi ma simili per semantica.Questi termini e documenti trasformati nello spazio "concettuale" sono quindi utilizzato nel recupero (retrival) al posto dei termini o dei documenti originali. Inoltre, anche la query viene trasformata nello spazio "concettuale" prima del recupero. Sia D la collezione dei testi, m il numero dei termini distinti in D e n il numero di documenti. La LSI parte da una matrice termini-documenti A mxn. Ogni riga di A rappresenta un termine e ogni colonna un documento. Questa matrice può essere calcolata in vari modi ad esempio usando la frequenza dei termini o con i valori $TF-IDF$. Usando la frequenza di termini ogni cella A_{ij} di A, indica il numero di volte che il termine i compare nel documento j.

11.2 Singular Value Decomposition

La singular value decomposition fattorizza una matrice A nel prodotto di tre matrici:

$$A = U\Sigma V^T$$

dove:

- U è una matrice mxr nella quale le colonne, chiamate left singular vectors, sono autovettori associati a gli autovalori r diversi da zero della matrice AA^T. Inoltre le colonne di U sono vettori ortogonali.

- V è una matrice nxr nella quale le colonne, chiamate right singular vectors, sono autovettori associati a gli autovalori r diversi da zero della matrice A^TA. Anche i vettori colonne di V sono ortogonali.

- Σ è una matrice diagonale rxr $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_r)$ con $\sigma_i \geq 0$. $\sigma_1, \sigma_2, ..., \sigma_r$, chiamati singular values, sono le radici quadrate dei r autovalori diversi da zero di AA^T.

Notiamo che inizialmente U è una matrice mxm e v una matrice nxn e Σ una matrice diagonale mxn. La diagonale di Σ è composta di autovalori non negativi di AA^T. Tuttavia,

a causa di autovalori uguali a zero, la matrice Σ è formata anche da colonne e righe nulle. La moltiplicazione tra matrici ci suggerisce che le righe e le colonne nulle possono essere eliminate da Σ. Allora possono essere eliminate anche le $m - r$ colonne in U e le $n - r$ colonne in V. m è il numero di righe in A, rappresenta il numero di termini. n è il numero di colonne in A, rappresenta il numero di documenti. r è il rango di A, $r \leq min(m, n)$. Un'importante aspetto del SVD è che possiamo eliminare alcune dimensioni insignificanti. Le dimensioni significanti sono indicate dalla grandezza dei singular values in Σ, che sono ordinati. Siano k i singular values più grandi in Σ tutti gli altri vengono messi a 0. La matrice approssimata di A viene chiamata A_k. Allo stesso modo possiamo ridurre la dimensione di Σ, U, V ottenendo:

$$A_k = U_k \Sigma_k V_k^T$$

Quindi usiamo le k più grandi terzine singolari per approssimare la matrice termini-documenti originale A. Il nuovo spazio viene chiamato spazio k-concettuale.

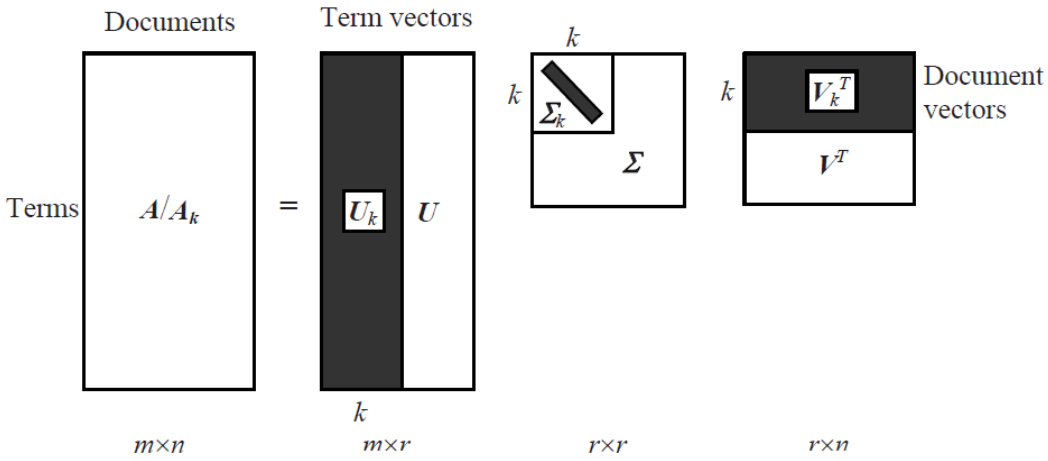


Fig. 6.10. The schematic representation of A and A_k

Il concetto cruciale è che SVD non ricostruisce esattamente la matrice dei termini-documenti A. Questo permette da un lato di catturare le strutture più importante che risiedono nell'associazione di termini e documenti, dall'altra di eliminare il "rumore" o la variabilità nell'uso delle parole che affligge i metodi di retrival per la corrispondenza delle keywords.

12 Unsupervised Learning

In alcune applicazioni i dati non sono classificati per cui si vuole trovare una qualche correlazione instrinseca nei dati. Il **clustering** è una tecnologia che può essere applicata a questo scopo. Questo metodo organizza i dati in gruppo simili, chimati clusters in modo che istanze di dati appartenenti ad uno stesso cluster siano simili tra di loro mentre istanze di dati di cluster differenti differiscano.

12.1 Basic Concepts

Il **clustering** è un processo di organizzione di istanze di dati in gruppi i quali membri siano simili tra di loro (intra-cluster similarity) e differenti con membri di altri (inter-cluster dissimilarity). I metodi di clustering possono essere di due tipi, gerarchici e partizionali.

12.2 Hierarchical Clustering

Il clustering gerarchico è un approccio di clustering che mira a costruire una gerarchia di clusters. Le strategie sono tipicamente di due tipi:

- **Agglomerativo**: si tratta di un approccio "bottom up" (dal basso verso l'alto) in cui si parte dall'inserimento di ciascun elemento in un cluster differente e si procede quindi all'accorpamento graduale dei clusters più simili fra loro(da definire in seguito).
- **Divisivo**: si tratta di un approccio "top down" (dall'alto verso il basso) in cui tutti gli elementi si trovano inizialmente in un singolo cluster che viene via via suddiviso ricorsivamente in sotto-clusters.

12.2.1 Hierarchical Agglomerative Clustering (HAC)

A differenza degli algoritmi di clustering non gerarchici questi offrono una struttura che risulta essere più informativa della sola divisione in clusters. Inoltre non è necessario specificare il numero di clusters che si vogliono creare, questi aspetti portano ad una perdita in termini di efficenza. Gli algoritmi di clustering gerarchico, infatti, hanno una complessità che è quadratica nella dimensione dei dati.

L'algoritmo:

1. Inizializzare ogni istanza in un cluster a sè.
2. Fino a quando si ha più di un cluster:
   (a) Tra i cluster correnti si determinano i 2 cluster, c_i e c_j, che sono più **simili**
   (b) Sostituire c_i e c_j con un unico cluster $c_k = c_i \cup c_j$

Nella costruzione dei clusters è quindi necessario calcolare un qualche tipo di similarità tra di essi. Questo puo essere fatto confrontando diversi i membri dei clusters stessi:

- **Single Link:** si confrontano i due membri più simili. Questo tipo di criterio risulta essere di tipo locale.

- **Complete Link:** si confrontano i due membri meno simili. Questo criterio non risulta essere di tipo locale in quanto considerando la posizione del membro più distante si considera anche la posizione dell'intero cluster. Viene usato principalmente in clusters di piccole dimensioni in quanto c'è il rischio di considerare gli outliers (punti troppo distanti dal centro del cluster) e calcolare così valori di similarità per niente veritieri.

- **Group Average:** si fa una media della similarità tra tutti i membri. Risulta essere un compromesso tra *single link* e *complete link*.

12.3 Non-Hierarchical Clustering

Anche questa categoria di algoritmi divide l'insieme dei dati in sottoinsiemi di clusters. Condividono lo stesso goal degli algoritmi gerarchici, cioè quello di creare clusters che siano internamente coerenti, ma che risultino differenti dagli altri. Questi algoritmi però creano la divisione in clusters senza preservare una struttura nelle relazioni tra questi. Un'altra importante divisione può essere fatta tra gli algoritmi di *hard* e *soft* clustering.

- **Hard Clustering:** ogni documento appartiene ad esattamente un cluster.

- **Soft Clustering:** l'assegnazione di un documento ad un cluster segue una distribuzione di probabilità su tutti i cluster.

Tra gli algoritmi di clustering non gerarchici il *K-means* è sicuramente il più famoso,

12.3.1 K-means Algorithm

Sia D l'insieme dei punti (istanze) con $D = \{x_1, x_2, ..., x_n\}$,
dove $x_i = (x_{i1}, x_{i2}, ..., x_{ir})$ è un vettore di uno spazio vettoriale $X \subseteq R^r$ e r è il numero di attributi nei dati. L'algoritmo *k-means* partiziona i dati in k cluster. Ogni cluster ha un centro, che è chiamato centroide. Il centroide è generalmente usato per rappresentare l'intero cluster ed è ottenuto semplicemente facendo la media tra tutti i punti del cluster. All'inizio l'algoritmo sceglie a caso k punti come centroidi seme (seeds). Quindi viene calcolata la distanza tra ogni seme e ogni altro punto dei dati. Ogni punto viene quindi

assegnato al centroide al quale è piu vicino. Un centroide e i suoi punti rappresentano quindi un cluster. Una volta che ogni punto è stato assegnato ad un centroide vengono calcolati dei nuovi centroidi per ogni cluster, usando tutti i punti appartenenti a quello stesso cluster. Il processo continua fino a quanto non viene raggiunto un criterio di stop. Ad esempio:

- Non è stato riassegnato nessun (o quasi) punto.
- Non c'è stato nessun cambio (o quasi) di centroidi.
- Diminuzione minima della somma dei quadrati degli errori (SSE)

$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2,$$

dove k è il numero di cluster richiesti, C_j è il $j-$esimo cluster, m_j è il centroide del cluster C_j e $dist(x, m_j)$ è la distanza tra il punto x e il centroide m_j.

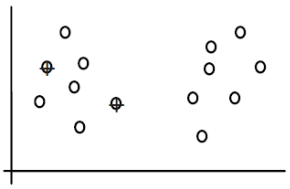
L'algoritmo k-means può essere utilizzato in ogni data set per cui sia possibile definirne e calcolarne la media. Nello spazio Euclideo la media di un cluster può essere calcolata come:

$$m_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

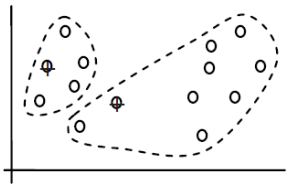
dove $|C_j|$ è il numero di punti nel cluster C_j. La distanza di un punto x_i dal punto medio di un cluster (centroide) m_j puo essere calcolata come la distanza euclidea:

$$dist(x_i, m_j) = ||x_i - m_j|| = \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + ... + (x_{ir} - m_{jr})^2}$$

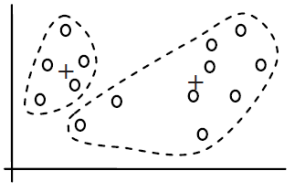
Per esempio qui l'algoritmo si ferma alla terza iterazione perché non ci sono stati cambiamenti dei clusters (fase 2 e 3 sono uguali):



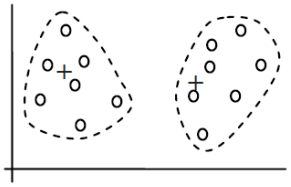
(A). Random selection of k seeds (or centroids)



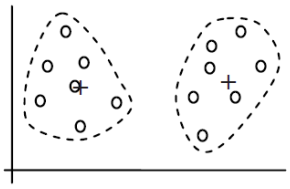
Iteration 1: (B). Cluster assignment



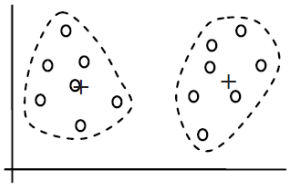
(C). Re-compute centroids



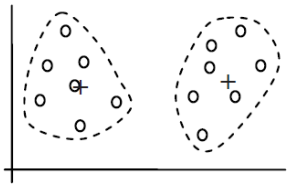
Iteration 2: (D). Cluster assignment



(E). Re-compute centroids



Iteration 3: (F). Cluster assignment



(G). Re-compute centroids

I metodi di misura della distanza sono:

- **Manhattan distance:**

$$dist(x, y) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_r - y_r|$$

- **Euclidean distance:** distanza Euclidea

$$dist(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_r - y_r)^2}$$

- **Weight Euclidean distance:** distanza Euclidea in cui si assegna un peso ad ogni attributo per esprimere l'importanza in relazione agli altri attributi

$$dist(x, y) = \sqrt{w_1(x_1 - y_1)^2 + w_2(x_2 - y_2)^2 + ... + w_r(x_r - y_r)^2}$$

- **Cosine similarity:** (trasformata in una distanza sottrando ad 1)

$$1 - \frac{x \cdot y}{|x| \cdot |y|}$$

- **Chebychev distance:** definisce due vettori come "differenti" se differiscono in ogni attributo

$$dist(x, y) = max(|x_1 - y_1|, |x_2 - y_2|, ..., |x_r - y_r|)$$

Il k-means però presenta alcuni punti di debolezza:

- L'algoritmo è applicabile solo può essere definita una media
- L'algoritmo è sensibile agli outliers
- E' necessario specificare il numero k di cluster (a volte è difficile saperlo se si ha un grande data set)
- Il risultato può essere legato molto alla selezione randomica dei seeds
- Alcuni seeds possono causare uno scarso tasso di convergenza o una convergenza verso raggruppamenti sotto-ottimi

12.3.2 Varianti del K-means

- **QT K-means**: Quality Threshold K-means è un'evoluzione del K-means classico. Cambia in maniera dinamica il numero dei clusters (di fatto non c'è proprio bisogno che l'utente inserisca k) utilizzando due soglie: σ, per la similarità intra-clusters, e τ, per la similarità inter-clusters.
  - Se $sim(w, c_x) < \sigma$ allora l'algoritmo crea un nuovo centroide (quindi un nuovo cluster c_z) e gli assegna w.

- Se $sim(c_x, c_y) > \tau$ allora l'algoritmo unisce i due clusters in uno solo (diminuendo il numero dei clusters) : $c_z = c_x \cup c_y$.

- **LAC** : Locally Adaptive Clustering è una variante del K-means in cui i clusters sono pesati (in base allo spazio in cui si trovano). Questa procedura viene nominata anche Subspace Clustering in quanto si cerca di clusterizzare anche i sottospazi del piano:

  - Ogni centroide è pesato di modo che vengano considerate poche dimensioni quando si assegna un punto ad un cluster. Di conseguenza per il vettore $X = (x_1, x_2, x_3, x_4)$ potrebbero essere considerate solo 1 o 2 dimensioni, piuttosto che tutte e 4.
  - Ad ogni iterazione la pesatura dei centroidi viene aggiornata
  - In ogni cluster i pesi determinano le dimensioni (o sottospazi) informativi.

12.3.3 Data Standardization

Nel clustering uno dei più importanti step del pre-processing dei dati è la standardizzazione. Ad esempio, se si utilizza la distanza Euclidea, la standardizzazione è altamente raccomandata cosicché tutti gli attributi possano avere lo stesso impatto nel calcolo delle distanze. Questo evita di ottenere cluster dominati da attributi con un alto valore di varianza. Per fissare le idee basta pensare ad un'istanza in cui tutti i punti si trovano nel range [0,5] e solo due/tre punti nel range [500,100'000]. Il risultato dell'applicazione del k-means, intuitivamente, sarà poco preciso proprio a caso di questo range.
Risulta evidente che differenti tipi di attriburi richiedono differenti tipi di standardizzazione. Ad esempio gli **interval-scale attributes** sono attributi numerici/reali con valori che seguono una scala lineare. Esempi di tali attributi sono l'età, l'altezza, il peso, i costi, etc. Nel caso degli interval-scale attributes esistono due approcci alla standardizzazione:

- **Range**: ogni valore viene diviso per il range dei valori validi per quell' attributo, si ottiene quindi un valore mappato in (0,1). Dato in valore del f-esimo attributo e

del i-esimo punto x_{if}, il nuovo valore $rg(x_{if})$ è:

$$rg(x_{if}) = \frac{x_{if} - min(f)}{max(f) - min(f)}$$

- **Z-score:** trasforma un valore di un attributo basandosi sulla media e sulla deviazione standard dell'attributo. Quindi lo z-score di un valore indica di quanto e in che direzione il valore devia dalla media dell'attributo. E' spesso quello più utilizzato. La deviazione standard, σ_f dell'attributo f è calcolata come:

$$\sigma_f = \frac{|x_{1f} - \mu_f| + |x_{2f} - \mu_f| + ... + |x_{rf} - \mu_f|}{n}$$

dove μ_f è definita come:

$$\mu_f = \frac{x_{1f} + x_{2f} + ... + x_{rf}}{n}$$

Mentre lo Z-score:

$$z(x_{if}) = \frac{x_{if} - \mu_f}{\sigma_f}$$

12.4 Cluster Evaluation

Dopo aver trovato un insieme di cluster, abbiamo bisogno di valutarne la bontà. A differenza dei metodi supervisionati in cui era semplice misurarne l'accuracy usando i dati di test, qui non si può conoscere il cluster ottimo per un dato data set. Si possono usare vari criteri di valutazione:

- **Internal criteria:** si può misurare la qualità dei cluster usando delle funzioni obiettivo che formalizzino come *goal* l'ottenere un alto valore di similarità interno ai cluster e un basso valore tra i clusters. Un buon valore rispetto ad un criterio interno non si rispecchia, necessariamente, in una effettiva qualità sull'applicazione.

- **Indirect evaluation:** in alcune applicazioni il clustering non è il task primario, le performance possono essere confrontate con quelle del task primario.

- **Exernal criteria:** Si possono testare le performance del clustering su alcuni dati con label (per la classificazione) nei quali ogni classe corrisponde ad un cluster; quindi, dopo aver fatto il clustering, si costruisce una matrice di confusione e se ne misura entropy, purity, precision, recall, F-score.

- Sia $C = (c_1, c_2, ..., c_k)$ l'insieme delle classi del dataset D. Il metodo di clustering produrrà k clusters, che divide D in k sottoinsiemi disgiunti $D_1, D_2, ..., D_k$
- Si può stimare $P(c_j)$ come la proporzione dei punti della classe c_j nel cluster i o D_i.

$$P(c_j) = |c_j \cap D_i|/|D_i|$$

Purity: misura il grado in cui un cluster contiene solo una classe di dati. La purity di un cluster viene calcolata come:

$$purity(D_i) = max_j(P(c_j))$$

La purity totale dell'intero clustering (su tutti i cluster) è:

$$purity_{tot}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \cdot purity(D_i)$$

Se tutti i cluster contengono solo una sola istanza, la purity è massima.

Entropy: per ogni cluster può essere calcolata come:

$$entropy(D_i) = -\sum_{j=1}^{k} P(c_j) log_2 P(c_j)$$

L'Entropy totale dell'intero clustering (su tutti i cluster) è:

$$entropy(D_i) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \cdot entropy(D_i)$$

Precision, Recall e F-score possono essere calcolate su ogni cluster considerando la classe più frequente in ogni cluster.

13 Information Retrival

Un modello IR definisce come rappresentare documenti e query e come viene definita la rilevanza di un documento per una query dell'utente. Mentre rappresentano in modo differente le query e i documenti, i principali modelli usano però lo stesso freamwork, ossia rappresentano tutti le parole come un modello bag of words. Le posizioni e la sequenza dei termini viene ignorata e i documenti vengono rapprensentati come insiemi di termini.

13.1 Boolean Model

Rapprensetazione dei documenti: nel modello booleano i documenti vengono rapprensentati come insiemi di termini. Ossia, ogni termine viene considerato solo come presente o assente. Quindi nella rappresentazione del vettore del documento ogni parola ha un peso (0,1) cioé assente o presente nel documento in questione.

Query Booleane: I termini di una query vengono combinati logicamente usando gli operatori booleani. Ad esempio, la query ((x AND y) AND (NOT z)) significa che i documenti restituiti devono contenere i termini x e y ma non il termine z.

Documenti restituiti: Data una query booleana, il sistema restituisce ogni documento che rende la query vera. Il recupero di un documento segue quindi un criterio booleano, esso può essere solo rilevante o irrilevante. Questo viene chiamato exact match. Non ci sono perciò nozioni di matching parziale o di ranking dei documenti. Questo risulta essere il più grande svantaggio del modello booleano.

13.2 Vector Space Model

Rappresentazione dei documenti: Un documento nel vector space model è rappresentato come un vettore pesato, nel quale ogni componente viene calcolato come una qualche variazione dello schema TF oppure TF-IDF.

Query: Possono essere processate allo stesso modo dei documenti.

Documenti recuperati e ranking: A differenza del modello booleano il modello dello spazio vettoriale non prende un unica decisione, invece, ordina i documenti con un ranking in accordo con il grado di rilevanza con la query. Un modo per calcolare il grado di rilevanza è quello di calcolare la similarità tra la query e ogni documento d_j della collezione D. Una funzione di similarità molto comune è la cosine similarity:

$$\text{cosine}\left(\mathbf{d}_j, \mathbf{q}\right) = \frac{\langle \mathbf{d}_j \bullet \mathbf{q} \rangle}{\|\mathbf{d}_j\| \times \|\mathbf{q}\|} = \frac{\sum_{i=1}^{|V|} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{|V|} w_{ij}^2} \times \sqrt{\sum_{i=1}^{|V|} w_{iq}^2}} \tag{1}$$

13.3 Page Rank

Questa tecnica di analisi dei link associa ad ogni nodo del grafo del web uno score compreso tra 0 e 1, conosciuto per l'appunto, come PageRank. Data una query, il motore di ricerca calcola uno score congiunto per ogni pagina del web, che combina alcune features come la cosine similarity o la term proximity insieme allo score del PageRank. Questo score composto viene usato per calcolare una lista ordinata dei risultati di una query. Consideriamo una navigazione randomica che comincia da una certa pagina ed esegue una passeggiata aleatoria nel web come segue. Ad ogni step, la passeggiata procede dalla pagina A corrente verso una pagina scelta a caso tra quelle verso cui la pagina A punta. Ad ogni step ci troviamo in esattamente una pagina, considerando la matrice di probabilità di transizione NxN (catena di markov con N stati), per $1 \leq i, j \leq N$, la cella della matrice P_{ij} ci dice la probabilità di trovarci, nel prossimo step, nella pagina j se ci troviamo nella pagina i.

In questo tipo di navigazione ci saranno delle pagine visitate più volte di altre, l'idea dietro il Page Rank è che queste pagine siano più importanti. Per eliminare il problema di pagine pozzo viene introdotta l'operazione di teleport, cioè la possibilità di arrivare ad una pagina da una qualsiasi altra pagina. In altre parole, se N è il numero totale di pagine l'operazione di teleport rende tutte le pagine raggiungibili con una probabilità $1/N$. La matrice costruita dal PageRank può quindi essere vista come una matrice dele transizioni di una catena di markov in cui ogni stato rappresenta una pagina.

Più nel dettaglio, possiamo vedere il web come un grafo orientato $G = (V, E)$, dove V è l'insieme dei vertici, l'insieme di tutte le pagine, e E è l'insieme degli archi diretti, hyperlink tra le pagine. Sia n il numero totale di pagine. Il PageRank score della pagina i, denotato con $P(i)$, è definito come:

$$P(i) = \sum_{(j,i) \in E} \frac{P(j)}{O_j} \tag{2}$$

Dove O_j è il numero di out-links (archi uscenti) della pagina j.

13.4 Hub and Authorities

Con questo metodo ad ogni pagina vengono assegnati 2 scores. Il primo viene chiamato hub score mentre il secondo autority score. Per ogni query vengono calcolate due liste di ranking: una per ogni score. Questo approccio porta alla distinzione di due tipi di pagine nel web, le pagine autorevoli (rispetto ad uno specifico topic) e le pagine hub che non sono autorevoli per quel topic ma che puntano alle pagine autorevoli.
Una buona pagina hub è una pagina che punta verso molte buone pagine autorevoli. Una buona pagina autorevole è una pagina che è puntata da molte buone pagine hub. Si arriva cosi ad avere una definizione circolare di pagine autorevoli e hub.
Per una pagina v denotiamo con $h(v)$ il suo score di hub e con $a(v)$ il suo score di authority. Inizialmente poniamo $h(v) = a(v) = 1$, denotiamo inoltre con $v \to y$ l'esistenza di un hyperlink da v a y.

$$\begin{aligned} h(v) &\leftarrow \textstyle\sum_{v \to y} a(y) \\ a(v) &\leftarrow \textstyle\sum_{y \to v} h(y) \end{aligned} \tag{3}$$

Vediamo cosa succede se eseguiamo questi aggiornamenti dei valori di hub e authority iterativamente. Siano $\vec{h}$ e $\vec{a}$ i vettori di tutti i valori di hub e di authority di tutte le pagine di un sottoinsieme delle pagine del web. Sia A la matrice di adiacenza del nostro sotto insieme, A è una matrice quadrata dove $A_{ij} = 1$ sse la pagina i ha un link verso la pagina j, 0 altrimenti. Allora possiamo scrivere l'equazione (2) come:

$$\begin{aligned} \vec{h} &\leftarrow A\vec{a} \\ \vec{a} &\leftarrow A^T\vec{h} \end{aligned} \tag{4}$$

Dove A^T rappresenta la matrice trasposta di A. Possiamo quindi riscrivere la (3) come:

$$\begin{aligned} \vec{h} &\leftarrow AA^T\vec{h} \\ \vec{a} &\leftarrow A^TA\vec{a} \end{aligned} \tag{5}$$

Introducendo gli autovalori (sconosciuti) otteniamo:

$$\begin{aligned} \vec{h} &= (1/\lambda_h)\, AA^T\vec{h} \\ \vec{a} &= (1/\lambda_a)\, A^TA\vec{a} \end{aligned} \tag{6}$$

Questo ci porta ad osservare che in fase di computazione non siamo interessati ad usare il metodo iterativo delle potenze ma invece possiamo usare metodi più veloci per calcolare gli autovettori di una matrice stocastica. La computazione risultante sarà quindi:

1. Costruire il sottoinsieme target delle pagine, calcolare quindi AA^T e A^TA.
2. Calcolare i principali autovettori di AA^T e A^TA per formare il vettore degli hub e quello degli authority .
3. Restituire gli hub con valori maggiori e gli authority con valori maggiori.

Questo metodo è conosciuto come $HITS$ che sta per hyperlink-induced topic search.-

14 Opinion Mining and Sentimental Analisys

L'opinion mining o la sentimental analisys si interessano al significato delle opinioni e a decidere se e quali sentimenti positivi o negativi ne scaturiscono. Si analizzano le opinioni, le valutazioni, le attitudini e le emozioni delle persone nei confronti delle entità, individui, problemi, eventi, argomenti e i loro attributi. Le opinioni sono importanti perché sono i fattori chiave che influenzano i nostri comportamenti. Le nostre convinzioni e le percezioni della realtà, le scelte che facciamo, sono considerevolmente condizionati da come gli altri vedono e valutano il mondo. Per questa ragione, quando abbiamo bisogno di prendere una decisione, spesso cerchiamo le opinioni di altri. Questo è vero non solo per gli individui ma anche per le organizzazioni.
Con il boom dei social media del web, gli individui e le organizzazioni hanno incrementato l'uso di questi materiali per prendere decisioni. Al giorno d'oggi un utente che vuole avere informazioni rigurado un prodotto non ha più bisogno di affidarsi ad amici, familiari, colleghi ma può trovare opinioni nel web. Allo stesso modo le aziende non hanno più bisogno di creare sondaggi per avere feedback rigurado ad un loro prodotto ma possono trovare un grande quantitativo di materiale gratuitamente nel web. Sfortunatamente però, le informazioni presenti nel web non sono facilmente recuperabili, date le dimensioni di quest'ultimo. Inoltre estrarre le informazioni dai testi presenti nel web risulta essere un task notevolmente arduo.

14.1 The Problem of Opinion Mining

Usiamo il seguente frammento di recensione di un IPhone per introdurre il problema
"(1) I bought an iPhone a few days ago. (2) It was such a nice phone. (3) The touch screen was really cool. (4) The voice quality was clear too. (5) However, my mother was mad with me as I did not tell her before I bought it. (6) She also thought the phone was too expensive, and wanted me to return it to the shop. ... "
La prima cosa che vogliamo capire è cosa vogliamo estrarre da questa recensione. Cio che possiamo osservare da subito è che ci sono più opinioni in questa recensione. Le frasi (2), (3) e (4) esprimono opinioni positive mentre le (5) e (6) opinioni negative. Inoltre notiamo che le opinioni hanno diversi target. Ad esempio l' oggetto delle opinioni della frase (2) è l'IPhone nel suo insieme, mentre gli oggetti delle opinioni delle frasi (3) e (4) sono "touch screen" e "voice quality". Infine i titolari delle opinioni cambiano anch'essi, nelle frasi (2),(3) e (4) si tratta dell'autore della recensione mentre nelle frasi (5) e (6) è "my mother".
Opinion Target: si una la parola entità (entity) per denotare l'oggetto target di un'opinione. Un'entità puo avere un'insieme di componenti e di attributi. Formalmente:

Definizione di entità: un'entità e è un prodotto, servizio, persona, evento, organizzazione o argomento. E' associata ad una coppia, $e(T, W)$, dove T è una gerarchia di componenti, sotto-componenti, e W è l'insieme degli attributi di e. Ogni componente e sotto-componente può avere il propro insieme di attributi.

Basandoci su questa definizione possiamo rappresentare un'entità come un albero gerarchico, la cui radice è il nome dell'entità. Ogni arco è una relazione *parte-di*. Ogni nodo è associato ad un insieme di attributi. Può essere espressa un'opinione per ogni nodo e per ogni attributo del nodo. In pratica questo viene semplificato appiattendo l'albero su 2 livelli ed usando il termine aspetti per indicare sia le componenti che gli attributi. Nell'albero semplificato la radice rimane ancora il nome dell'entità mentre gli altri nodi sono i diversi aspetti dell'entità.

Definizione di aspetto (feature): gli aspetti di un'entità e sono le componenti e gli attributi dell'entità e.

Definizione di nome dell'aspetto e espressione dell'aspetto: Un nome della'aspetto è il nome dell'aspetto dato dall'utente, mentre l'espressione di un aspetto è una parola o una frase nel testo che indicano un certo aspetto.

Possiamo inolte differenziare tra espressioni di aspetti esplicite ed implicite. Chiamiamo espressioni di aspetti esplicite le espressioni che sono nomi o frasi nominali. Ad esempio l'espressione "the sound" nella frase "the sound of this phone is clear" viene detta esplicita. Indichiamo invece con espressioni implicite quelle che sottintendono alcuni aspetti. Ad esempio "large" è un'espressione implicita nella frase "this phone is too large" poichè sta sottintendendo l'aspetto "size".

Definizione del titolare: un titolare di un'opinione è la persona o l'organizzazione che esprime l'opinione. Vengono anche chiamati sorgenti dell'opinione (opinion source).

Ci sono due tipi di opinioni: **opinioni regolari** (regular opinion) e **opinioni comparative**(comparative opinion). Le opinioni regolari sono semplicemente delle opinioni. Le opinioni comparative, invece, esprimono una relazione di similarità o differenza tra due o più entità. L'opinione è semplicemente un parere, un'attitudine o un'emozione positiva o negativa riguardo un'entità o un aspetto di un'entità del titolare dell'opinione. Positivo, negativo e neutro sono detti orientamenti dell'opinione, anche chiamati orientamenti del sentimento, orientamenti semantici o polarità.

Definizione dell'opinione: un'opinione è una quintupla,

$$(e_i, a_{ij}, o_{ijkl}, h_k, t_l)$$

dove e_i è il nome di un'entità, a_{ij} è un aspetto dell'entità e_i, o_{ijkl} è l'orientamente dell'opinione riguardo l'aspetto a_{ij} dell'entità e_i, h_k è il titolare dell'opinione e t_l è il tempo nel quale è stata espressa l'opinione da h_k.

Analizziamo quindi l'obiettivo dell'opinion mining, nel quale data una collezione D di documenti contenenti opinioni, si vogliono trovare tutte le quintuple di opinioni $(e_i, a_{ij}, o_{ijkl}, h_k, t_l)$ in D. Per ottenere questo obiettivo abbiamo bisogno di risolvere alcuni task:

- **Task 1** (entity extraction and grouping): Estrarre tutte le espressioni di entità in D, raggruppando entità simili (sinonimi) negli stessi clusters. Ogni cluster di espressione di entità rappresenta un'entità e_i.

- **Task 2** (aspect extraction and grouping): Estrarre tutte le espressioni di aspetti di una entità, raggruppandole in clusters.

- **Task 3** (opinion holder and time extraction): estrarre queste informazioni dal testo o dai dati strutturati.

- **Task 4** (aspect sentiment classification): determinare per ogni opinione di un aspetto se questa è positiva, negativa o neutra.

- **Task 5** (opinion quintuple generation): generare tutte le quintuple delle opinioni contenute in D usando i risultati ottenuti dai precedenti tasks.

La difficoltà dell'obiettivo dell'opinion mining sta nel fatto che nessuno di questi task è un problema facilmente risolvibile, cioè per nessuno di questi esiste una soluzione che tiri sempre fuori una soluzione ottima.