KERNEL-BASED LEARNING

WM&R a.a. 2013/14

R. Basili, (A. Moschitti)
Università di Roma “Tor Vergata”
basili@info.uniroma2.it

Outline

- Metodi Kernel
  - Motivazioni
  - Esempio
- Kernel standard
  - Polynomial kernel
  - String Kernel
- Introduzione a metodi Kernel *avanzati*
  - (Tree kernels)
  - Lexical semantic kernels

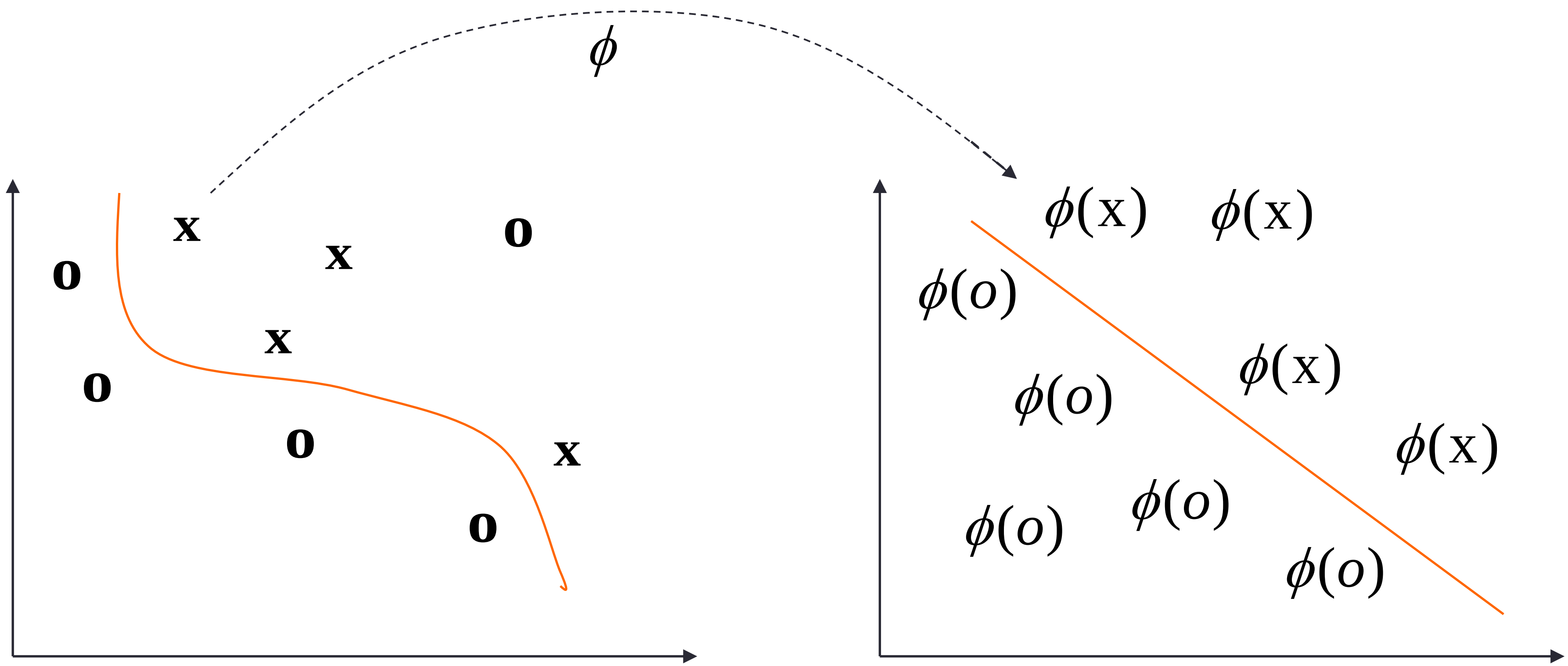
La funzione Kernel

- Il learning dipende solo dal prodotto scalare dei vettori di esempio
- Quindi dipende dalla Gram-matrix. In generale si definisce kernel la funzione:
- $$K(\vec{z}, \vec{x}) = \phi(\vec{z}) \cdot \phi(\vec{x})$$
- La funzione kernel produce il risultato a partire dagli oggetti iniziali
- Quando la mappatura ϕ è l'identità abbiamo l'usuale prodotto scalare.
- Gli esempi compaiono nell'algoritmo di learning (ad es. il percettrone) solo attraverso i loro contributi al kernel

Primo Vantaggio: rendere linearmente separabili gli esempi

- Mappare i dati in uno Spazio di Feature dove sono linearmente separabili $\vec{x} \rightarrow \phi(\vec{x})$

  (i.e. attributi $\longrightarrow$ feature)



Esempio di una funzione di mappatura

- Due masse m_1 e m_2 , una vincolata
- Applico una forza f_a alla massa m_1
- Esperimenti
  - Features m_1 , m_2 e f_a
- Supponiamo di volere apprendere **quando m_1 si allontana da m_2**

- Considerando la legge gravitazionale di Newton

$$f(m_1, m_2, r) = C\frac{m_1 m_2}{r^2}$$

- Dobbiamo determinare se $f(m_1, m_2, r) < f_a$

Esempio di una funzione di mappatura

$$\vec{x} = (x_1, \ldots, x_n) \rightarrow \phi(\vec{x}) = (\phi_1(\vec{x}), \ldots, \phi_n(\vec{x}))$$

- Non esprimibile linearmente, quindi cambio spazio

$$(f_a, m_1, m_2, r) \rightarrow (k, x, y, z) = (\ln f_a, \ln m_1, \ln m_2, \ln r)$$

- Poiché

$$\ln f(m_1, m_2, r) = \ln C + \ln m_1 + \ln m_2 - 2\ln r = c + x + y - 2z$$

- Allora l'iperpiano è la funzione richiesta

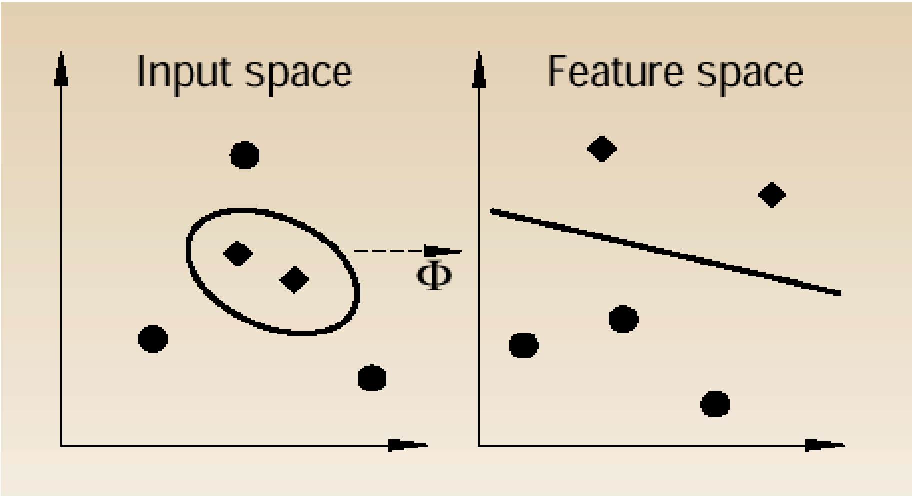
$$\ln f_a - \ln m_1 - \ln m_2 + 2\ln r - \ln C = 0$$

$(1,1,-2,-1) \cdot (\ln m_1, \ln m_2, \ln r, \ln f_a) + \ln C = 0$, posso decidere senza errore se le masse si avvicinano o si allontanano

Feature Spaces and Kernels

- Feature Space
  - Lo spazio di input è mappato in un nuovo spazio dotato di prodotto scalare F (detto *feature space*) attraverso una trasformazione (non lineare)

$$\phi = R^N \rightarrow F$$



- Kernel
  - La valutazione della funzione di decisione richiede il prodotto scalare ma mai i pattern rimappati in forma esplicita $\phi(x)$
  - Il prodotto scalare viene calcolato attraverso la funzione kernel

$$k(x, y) = (\phi(x) \cdot \phi(y))$$

Funzione di classificazione: forma duale

$$\mathrm{sgn}(\vec{w}\cdot\vec{x}+b)=\mathrm{sgn}\left(\sum_{j=1..\ell}\alpha_j y_j \vec{x}_j\cdot\vec{x}+b\right)$$

- Notare che i dati appaiono solo nel prodotto scalare
- La matrice $G=\left(\left\langle \mathbf{x}_i\cdot\mathbf{x}_j\right\rangle\right)_{i,j=1}^{l}$ è chiamata Gram matrix

Algoritmo duale del Percettrone e le funzioni Kernel

- Possiamo riscrivere la funzione di decisione:

$$h(x) = \mathrm{sgn}(\vec{w} \cdot \phi(\vec{x}) + b) = \mathrm{sgn}(\sum_{j=1..\ell} \alpha_j y_j \phi(\vec{x}_j) \cdot \phi(\vec{x}) + b) =$$

$$= \mathrm{sgn}(\sum_{i=1..\ell} \alpha_j y_j k(\vec{x}_j, \vec{x}) + b)$$

- ... e nel processo di aggiornamento

$$y_i(\sum_{j=1..\ell} \alpha_j y_j \phi(\vec{x}_j)) \cdot \phi(\vec{x}_i) + b = \sum_{j=1..\ell} \alpha_j y_i y_j k(\vec{x}_j, \vec{x}_i) + b$$

Kernels in Support Vector Machines

- Nel problema Soft Margin SVMs si deve massimizzare:

$$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j + \frac{1}{2C} \vec{\alpha} \cdot \vec{\alpha} - \frac{1}{C} \vec{\alpha} \cdot \vec{\alpha}$$

Usando le funzioni kernel possiamo riscrivere il problema come:

$$\begin{cases} maximize \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \left(k(o_i, o_j) + \frac{1}{C} \delta_{ij} \right) \\ \alpha_i \geq 0, \quad \forall i = 1, .., m \\ \sum_{i=1}^{m} y_i \alpha_i = 0 \end{cases}$$

Definizione delle Funzioni Kernel

Def. 2.26 *A kernel is a function k, such that* $\forall\ \vec{x}, \vec{z} \in X$

$$k(\vec{x}, \vec{z}) = \phi(\vec{x}) \cdot \phi(\vec{z})$$

where ϕ is a mapping from X to an (inner product) feature space.

- Le funzioni kernel esprimono mappature implicite di questo tipo

$$\vec{x} \in \Re^n, \qquad \vec{\phi}(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), ..., \phi_m(\vec{x})) \in \Re^m$$

Validità delle funzioni kernel (1)

Def. B.11 *Eigen Values*
Given a matrix $\boldsymbol{A} \in \mathbb{R}^m \times \mathbb{R}^n$, *an egeinvalue* λ *and an egeinvector* $\vec{x} \in \mathbb{R}^n - \{\vec{0}\}$ *are such that*

$$\boldsymbol{A}\vec{x} = \lambda\vec{x}$$

Def. B.12 *Symmetric Matrix*
A square matrix $\boldsymbol{A} \in \mathbb{R}^n \times \mathbb{R}^n$ *is symmetric iff* $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji}$ *for* $i \neq j$ $i = 1, .., m$ *and* $j = 1, .., n$, *i.e. iff* $\boldsymbol{A} = \boldsymbol{A}'$.

Def. B.13 *Positive (Semi-) definite Matrix*
A square matrix $\boldsymbol{A} \in \mathbb{R}^n \times \mathbb{R}^n$ *is said to be positive (semi-) definite if its eigenvalues are all positive (non-negative).*

Validità delle funzioni kernel (2)

Proposition 2.27 *(Mercer's conditions)*
Let X be a finite input space with $K(\vec{x}, \vec{z})$ a symmetric function on X. Then $K(\vec{x}, \vec{z})$ is a kernel function if and only if the matrix

$$k(\vec{x}, \vec{z}) = \phi(\vec{x}) \cdot \phi(\vec{z})$$

is positive semi-definite (has non-negative eigenvalues).

- L'idea principale di questa proposizione è che *se la Gram matrix è semidefinita positiva allora esiste il mapping ϕ che realizza la funzione kernel, cioè uno spazio F in cui la separabilità è espressa in modo migliore*

Feature Spaces and Kernels

- Esempio di Kernel
  - Polynomial kernel

    - Se *d=2* e $k(x,y) = (x \cdot y)^d$

      $x, y \in R^2$

$$(x \cdot y)^2 = \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \right)^2 = \left(\begin{bmatrix} x_1^2 \\ \sqrt{2}\, x_1 x_2 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} y_1^2 \\ \sqrt{2}\, y_1 y_2 \\ y_2^2 \end{bmatrix} \right)$$

$$= (\phi(x) \cdot \phi(y)) = k(x,y)$$

Polynomial Kernel (*n* dimensions)

$$
\begin{aligned}
(\vec{x}\cdot\vec{z})^2 &= \Big(\sum_{i=1}^{n} x_i z_i\Big)^2 &&= \Big(\sum_{i=1}^{n} x_i z_i\Big)\Big(\sum_{j=1}^{n} x_i z_i\Big) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j &&= \sum_{i,j\in\{1,..,n\}} (x_i x_j)(z_i z_j) \\
&= \sum_{k=1}^{m} X_k Z_k &&= \vec{X}\cdot\vec{Z}
\end{aligned}
$$

General Polynomial Kernel (*n* dimensions)

$$(\vec{x} \cdot \vec{z} + c)^2 = \Big(\sum_{i=1}^{n} x_i z_i + c\Big)^2 = \Big(\sum_{i=1}^{n} x_i z_i + c\Big)\Big(\sum_{j=1}^{n} x_i z_i + c\Big) =$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j z_i z_j + 2c \sum_{i=1}^{n} x_i z_i + c^2 =$$

$$= \sum_{i,j \in \{1,..,n\}} (x_i x_j)(z_i z_j) + \sum_{i=1}^{n} (\sqrt{2c}\, x_i)(\sqrt{2c}\, z_i) + c^2$$

Polynomial kernel and the conjunction of features

- The initial vectors can be mapped into a higher dimensional space (c=1)

$$\Phi(< x_1, x_2 >) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

- More expressive, as (x_1x_2) encodes

  stock+market vs. *downtown+market* features

- We can smartly compute the scalar product as

$$\begin{aligned}\Phi(\vec{x}) \cdot \Phi(\vec{z}) &= (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2, \sqrt{2}z_1, \sqrt{2}z_2, 1) = \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 + 2x_1z_1 + 2x_2z_2 + 1 = \\ &= (x_1z_1 + x_2z_2 + 1)^2 = \boxed{(\vec{x} \cdot \vec{z} + 1)^2 = K_{p2} \ (\vec{x}, \vec{z})}\end{aligned}$$

Architettura di una SVM

- Classificatore non lineare (basato su kernel)
  - La funzione di decisione è

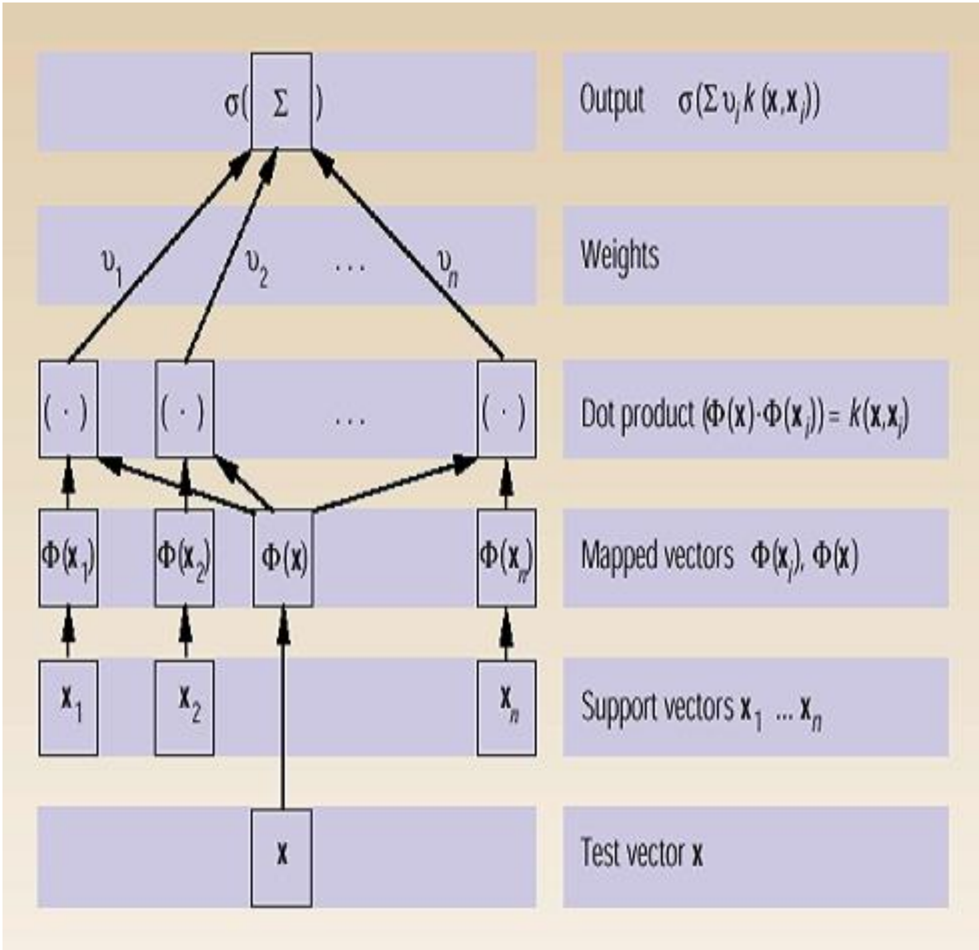
$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{l} v_i (\phi(x)\cdot\phi(x_i)) + b\right)$$

$$= \operatorname{sgn}\left(\sum_{i=1}^{l} v_i k(x, x_i) + b\right)$$

$\phi(x_i)$ *sostituisce ogni esempio di training* x_i

$v_i = \alpha_i y_i$

v_i sono calcolate attraverso la soluzione del problema di ottimizzazione



String Kernel

- Given two strings, the number of matches between their substrings is computed
- E.g. *Bank* and *Rank*
  - *B, a, n, k, Ba, Ban, Bank, an, ank, nk*
  - *R, a , n , k, Ra, Ran, Rank, an, ank, nk*
- String kernel over sentences and texts
- Huge space but there are efficient algorithms

Formal Definition

Sottosequenza di indici <u>ordinati</u> e <u>non contigui</u> di (1, ... $|s|$)

$$s = s_1, \ldots, s_{|s|}$$

$$\vec{I} = \boxed{(i_1, \ldots, i_{|u|})} \qquad u = s[\vec{I}]$$, substring of s defined by $\overrightarrow{I}$

$$\phi_u(s) = \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{l(\vec{I})}, \text{ con } l(\vec{I}) = i_{|u|} - i_1 + 1$$

$$K(s,t) = \sum_{u \in \Sigma^*} \phi_u(s) \cdot \phi_u(t) = \sum_{u \in \Sigma^*} \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{l(\vec{I})} \sum_{\vec{J}:u=t[\vec{J}]} \lambda^{l(\vec{J})} =$$

$$= \sum_{u \in \Sigma^*} \sum_{\vec{I}:u=s[\vec{I}]} \sum_{\vec{J}:u=t[\vec{J}]} \lambda^{l(\vec{I})+l(\vec{J})}, \text{ con } \Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

Kernel tra *Bank* e *Rank*

B, a, n, k, Ba, Ban, Bank, an, ank, nk, Bn, Bnk, Bk and ak are the substrings of *Bank*.

R, a, n, k, Ra, Ran, Rank, an, ank, nk, Rn, Rnk, Rk and ak are the substrings of *Rank*.

- Common substrings:
  - *a*, *n*, *k*, *an*, *ank*, *nk*, *ak*
- Notice how these are the same subsequences as between
  - *Schrianak* and *Rank*

An example of string kernel computation

- $\phi_{\mathrm{a}}(\mathrm{Bank}) = \phi_{\mathrm{a}}(\mathrm{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(2-2+1)} = \lambda,$
- $\phi_{\mathrm{n}}(\mathrm{Bank}) = \phi_{\mathrm{n}}(\mathrm{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(3-3+1)} = \lambda,$
- $\phi_{\mathrm{k}}(\mathrm{Bank}) = \phi_{\mathrm{k}}(\mathrm{Rank}) = \lambda^{(i_1-i_1+1)} = \lambda^{(4-4+1)} = \lambda,$
- $\phi_{\mathrm{an}}(\mathrm{Bank}) = \phi_{\mathrm{an}}(\mathrm{Rank}) = \lambda^{(i_1-i_2+1)} = \lambda^{(3-2+1)} = \lambda^2,$
- $\phi_{\mathrm{ank}}(\mathrm{Bank}) = \phi_{\mathrm{ank}}(\mathrm{Rank}) = \lambda^{(i_1-i_3+1)} = \lambda^{(4-2+1)} = \lambda^3,$

  $\phi_{\mathrm{nk}}(\mathrm{Bank}) = \phi_{\mathrm{nk}}(\mathrm{Rank}) = \lambda^{(i_1-i_2+1)} = \lambda^{(4-3+1)} = \lambda^2,$

  $\phi_{\mathrm{ak}}(\mathrm{Bank}) = \phi_{\mathrm{ak}}(\mathrm{Rank}) = \lambda^{(i_1-i_2+1)} = \lambda^{(4-2+1)} = \lambda^3.$

It follows that $K(\mathrm{Bank}, \mathrm{Rank}) = (\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3)\cdot(\lambda, \lambda, \lambda, \lambda^2, \lambda^3, \lambda^2, \lambda^3)$
$= 3\lambda^2 + 2\lambda^4 + 2\lambda^6.$

Tree Kernels

- Cio’ che è stato mostrato per la gestione di sequenze (stringhe) nell’apprendimento di una SVM non dipende strettamente dalla nozione di sequenza, ma si applica a strutture piu’ complesse
  - Alberi, che sono caratterizzati da un molteplicità sottosequenze, che corrispondono ai cammini radice-foglie
  - Grafi, che sono caratterizzati da piu’ alberi, quindi da una maggiore molteplicità di sottosequenze

Tree kernels

- Applications are related to **text processing** tasks such as
  - Syntactic parsing, when SVM classification is useful to select the best parse tree among multiple legal grammatical interpretations
  - Question Classification, where SVM classification is applied to the recognition of the target of a question (e.g. a person such as in "*Who is the inventor of the light?*" vs. a place as in "*Where is Taji Mahal?*"

or to **pattern recognition** (e.g. in bioinformatics the classification of protein structures)

Tree kernels

- Details are in a separated lesson’s slides

Kernel Lessicale Semantico

- I sistemi di Text Classification agiscono sulle rappresentazioni vettoriali $\vec{d}$ dei documenti d
- Dato un documento d posso tentare di rappresentarlo in uno spazio dei concetti (cioe' sensi secondo Wordnet) invece che nello spazio di parole (VSM tradizionale)
- Questo genera una immagine (complessa) dei sensi delle parole di d che chiameremo $\Phi(d)$
- Per apprendere da esempi d1 e d2 si può utilizzare una funzione della similitudine tra tutti i termini di d1 e quelli di d2, con un kernel (semantico lessicale) che calcola quindi

$$SK(d_1, d_2) = \Phi(d_1) \cdot \Phi(d_2)$$

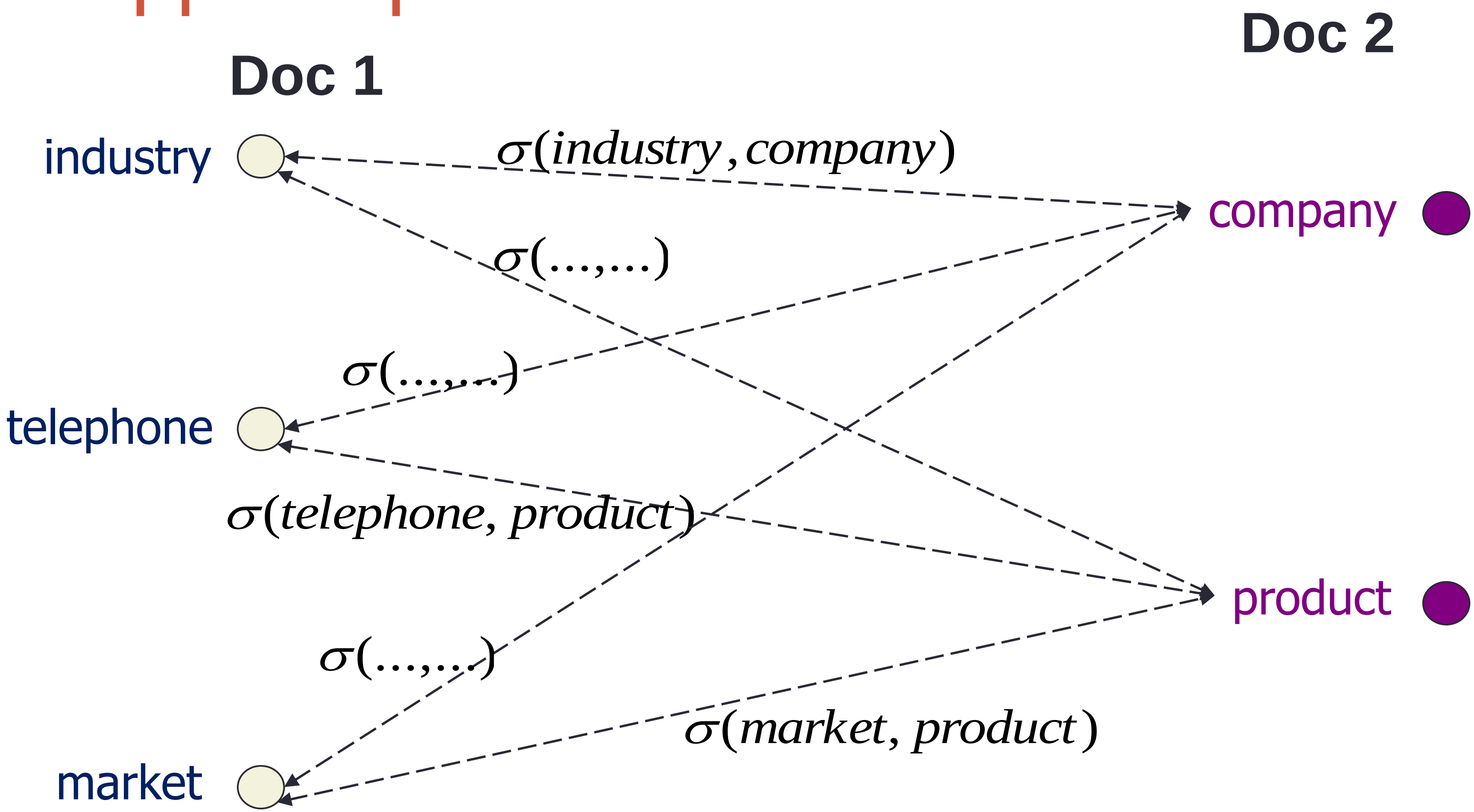
Kernel Lessicale Semantico (2)

- La similarità tra due documenti, d_1 e d_2, è data dal kernel semantico SK seguente:

$$SK(d_1, d_2) = \sum_{w_1 \in d_1, w_2 \in d_2} \sigma(w_1, w_2)$$

- dove σ è una funzione della similarità semantica in Wordnet tra due parole
- Una possibile scelta di σ è data dalla *conceptual density* in Wordnet (vedi lezioni su WSD)

Similarità tra documenti e *conceptual density* su coppie di parole



OSS: in un VSM tradizionale la similarità tra i due documenti sarebbe 0 perche' non ci sono parole comuni tra Doc1 e Doc2!!

Is it a valid kernel?

- It may not be a kernel so we can use ***M´·M***

Proposition B.14 *Let* $\boldsymbol{A}$ *be a symmetric matrix. Then* A *is positive (semi-) definite iff for any vector* $\vec{x} \neq 0$

$$\vec{x}'\boldsymbol{A}\vec{x} > \lambda\vec{x} \quad (\geq 0).$$

From the previous proposition it follows that: If we find a decomposition $\boldsymbol{A}$ in $M'M$, then $\boldsymbol{A}$ is semi-definite positive matrix as

$$\vec{x}'\boldsymbol{A}\vec{x} = \vec{x}'\boldsymbol{M}'\boldsymbol{M}\vec{x} = (\boldsymbol{M}\vec{x})'(\boldsymbol{M}\vec{x}) = \boldsymbol{M}\vec{x} \cdot \boldsymbol{M}\vec{x} = ||\boldsymbol{M}\vec{x}||^2 \geq 0.$$

Summary

- La forma duale del problema di ottimizzazione di una SVM dipende solo dal prodotto scalare degli esempi di adddestramento e NON dalla loro esplicita rappresentazione vettoriale (come nel percettrone)
- Questo suggerisce di sfruttare questa proprietà per
  - Definire funzioni capaci di calcolare in modo il prodotto scalare a partire dalla rappresentazione originale
  - Utilizzando quindi rappresentazioni (cioe’ spazi di feature) piu’ complesse(i) in modo implicito
- Tale ricerca di spazi linearmente separabili
  - Mantiene le proprietà matematiche che garantiscono la minimizzazione dell’errore atteso
  - Contiene la complessità del processo di training e di classificazione

Summary (2)

- Affinchè una funzione K(.,.) sia un kernel la gram matrix corrrispondente deve essere semi-definita positiva
- Possono essere fornite anche semplici prove empiriche di tale proprietà sui data set di addestramento
- Sono state discusse:
  - Funzioni kernel di base (per esempio I kernel polinomiali di grado 2)
  - Funzioni kernel dipendenti dalla struttura del task
    - String (Sequence) kernels
    - Semantic kernel

References

- Kernel Methods for Pattern Analysis, John Shawe-Taylor & Nello Cristianini - Cambridge University Press, 2004
- Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz
- Lodhi, Huma; Saunders, Craig; Shawe-Taylor, John; Cristianini, Nello; Watkins, Chris (2002). "Text classification using string kernels". Journal of Machine Learning Research: 419–444.
- Roberto Basili, Marco Cammisa and Alessandro Moschitti, Effective use of wordnet semantics via kernel-based learning. In Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL 2005), Ann Arbor(MI), USA, 2005
- Building Semantic Kernels for Text Classification using Wikipedia, Pu Wang and Carlotta Domeniconi, Department of Computer Science, George Mason University