Docenti: **Giorgio Valentini**
**Matteo Re**

UNIVERSITÀ DEGLI STUDI DI MILANO

C.d.l. Informatica

# Bioinformatica

A.A. 2012-2013 semestre I

## modulo 2 – parte B.2

**2**

## Kernel e funzioni di score kernelizzate

Bio

## MOTIVAZIONI

Rappresentazioni basate su reti sono in grado di modellare molti tipi di oggetti/fenomeni osservabili nel mondo reale :

- reti **tecnologiche** : www, internet, circuiti elettrici,...
- reti **sociali** : amicizie, collaborazioni, diffusione malattie,...
- reti **biologiche**: struttura proteine, regolazione trascrizionale geni, reti metaboliche, protein-protein interaction (PPI), …

CS

## MOTIVAZIONI

I metodi kernel godono di grande popolarità e vengono applicati in diversi ambiti di ricerca tra cui apprendimento automatico e bioinformatica:

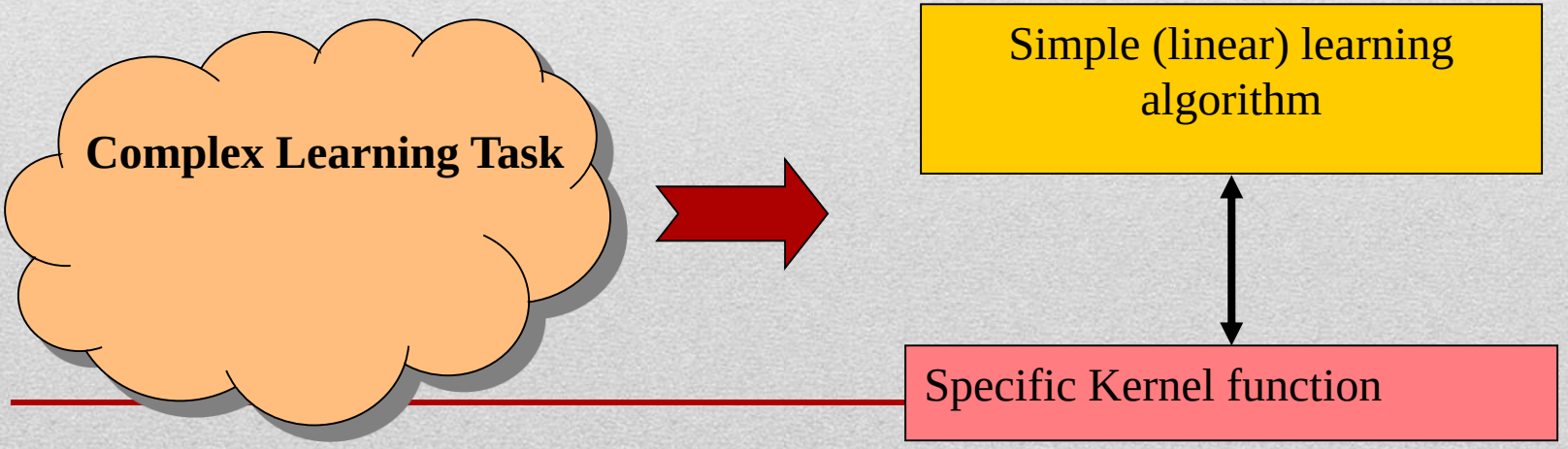
- $k : U \times U \rightarrow R$ : rappresentazione **flessibile** dei dati
- I kernel permettono di operare un'integrazione di dati **eterogenei** (basata sull'integrazione delle matrici di GRAM)
- Disaccoppiamento tra rappresentazione e algoritmi di learning (riusabilità algoritmi di learning e possibilità di costruire kernel per problemi specifici)
- Mediante l'utilizzo di kernel è possibile affrontare problemi di apprendimento non supervisionato, supervisionato e semisupervisionato
- Possibilità di usare kernel in algoritmi di classificazione e ranking.

CS

## METODI KERNEL

Nei metodi kernel la soluzione di un problema è «divisa» in due componenti distinte :

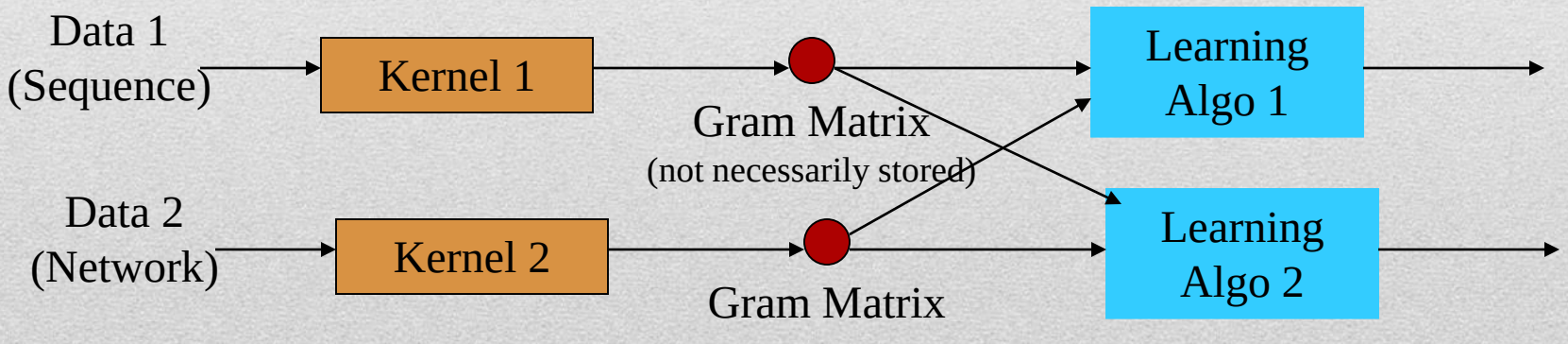
- Un algoritmo di apprendimento/classificazione **generico**, ad es. SVM, PCA, … (spesso lineare)
- Un kernel **specifico** per il problema considerato



## VANTAGGI

**Modularità** e **riusabilità**:

- Stesso kernel, diversi algoritmi di learning
- Kernel diversi, medesimo algoritmo di learning



CS

## Riposizionamento di farmaci e kernel

Per poter realizzare un test di riposizionamento di farmaci (mediante ranking di nodi in un grafo non diretto) basato sull'utilizzo di kernel:

- *Abbiamo bisogno di kernel definiti su grafi.*
- In questo caso il kernel fornirà una misura di similarità **tra i nodi** del grafo.
- Dobbiamo capire cosa intendiamo per «similarità» tra nodi appartenenti ad un grafo…

CS

## Riposizionamento di farmaci e kernel

- *Sia G=(V,E) un grafo non diretto pesato*
- Sia **W** la matrice di adiacenza del grafo G in cui gli elementi $w_{ij}$ rappresentano la «forza» della relazione tra i vertici *i* e *j*
- Sia $V_C$ un set di nodi noti per il fatto che godono di una generica proprietà C (ad es. farmaci di una categoria terapeutica, geni di una data categoria funzionale, ecc..)

**Obiettivo:** costruzione di una scoring function $S : V \rightarrow R^+$ basata su un opportuno **kernel K definito su G** che permetta di ordinare direttamente i nodi *i* : maggiore il valore di **S(*i*)** , maggiore la probabilità che **anche il nodo *i*** appartenente a G goda della proprietà C .

NB: per ora diamo per scontata la disponibilità di K

## Riposizionamento di farmaci e kernel

La funzione di score che vogliamo ottenere sarà definita su una misura di distanza definita in uno spazio di Hilbert **H**.

$\Phi : X \rightarrow H$ ( H è un RKHS)

$K : X \times X \rightarrow R$ (funzione **kernel** associata ad H)

tale che $< \Phi(.), \Phi(.)>_H = K(.,.)$

$< . , . >_H$ è un prodotto interno in H

mapping da X ad H

CS

## Riposizionamento di farmaci e kernel

Definiamo una misura di distanza **D(i, VC, X)** in **H** tra un nodo *i* appartenente a V e l'intero set di nodi VC basata sull'intero set di features X associato ad ogni vertice in G.

**MOTIVAZIONI per la scelta di operare in H :**

La scelta di operare con una distanza definita in H ci permette (sfruttando il «kernel trick») di incorporare **qualsiasi** kernel valido nella misura di distanza stessa

E' quindi possibile sfruttare kernel esistenti o disegnarne di appropriati per il problema che vogliamo affrontare

## Riposizionamento di farmaci e kernel

Misura di distanza **$D(i, V_C, X)$** … X è uguale per tutti i nodi *i* appartenenti a V. Quindi, per semplicità possiamo indicare la distanza in **H** tra un nodo *i* appartenente a V e l'intero set di nodi $V_C$ come:

**$D(i, V_C)$**

Esistono molti tipi di distanza che possiamo definire nello spazio H mediante il kernel associato ad esso.

CS

## Distanza media in H : $D_{AV}$

E' semplicemente la distanza media nello spazio di Hilbert in cui viene mappato X, tra il vertice *i* ed i vertici appartenenti a $V_C$ :

$$D_{AV}(i, V_C) = \left|\left| \phi(\boldsymbol{x}_i) - \frac{1}{|V_C|} \sum_{j \in V_C} \phi(\boldsymbol{x}_j) \right|\right|^2$$

sviluppiamo il quadrato …

**prodotti interni in H**

$$D_{AV}(i, V_C) = < \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_i) > - \frac{2}{|V_C|} \sum_{j \in V_C} < \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) >$$

$$+ \frac{1}{|V_C|^2} \sum_{k \in V_C} \sum_{j \in V_C} < \phi(\boldsymbol{x}_k), \phi(\boldsymbol{x}_j) >$$

CS

## Distanza media in H : $D_{AV}$

$$D_{AV}(i, V_C) = < \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_i) > - \frac{2}{|V_C|} \sum_{j \in V_C} < \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) >$$

$$+ \frac{1}{|V_C|^2} \sum_{k \in V_C} \sum_{j \in V_C} < \phi(\boldsymbol{x}_k), \phi(\boldsymbol{x}_j) >$$

- $D_{AV}$ è una misura di distanza
- $< \Phi(.), \Phi(.) >_H = K( . , . )$
- Vogliamo una misura di similarità (per il ranking)

**cambio segno e uso il kernel !**

CS

## Da $D_{AV}$ a $Sim_{AV}$

$$D_{AV}(i, V_C) = < \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_i) > - \frac{2}{|V_C|} \sum_{j \in V_C} < \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) > + \frac{1}{|V_C|^2} \sum_{k \in V_C} \sum_{j \in V_C} < \phi(\boldsymbol{x}_k), \phi(\boldsymbol{x}_j) >$$

**prodotto interno in H**

Una volta calcolato lo score per tutti i nodi è possibile ordinarli direttamente per score crescente

$$Sim_{AV}(i, V_C) = -K(\boldsymbol{x}_i, \boldsymbol{x}_i) + \frac{2}{|V_C|} \sum_{j \in V_C} K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \frac{1}{|V_C|^2} \sum_{k \in V_C} \sum_{j \in V_C} K(\boldsymbol{x}_k, \boldsymbol{x}_j)$$

**costante**

CS

## graph kernels

Abbiamo dato per scontato di avere a disposizione un kernel definite sul grafo G …

Esistono diversi kernel definiti su grafo. Uno dei più conosciuti (ed utilizzati in applicazioni pratiche) è il **Random Walk Kernel** !

## graph kernels

Sia G=(V,E) un grafo non diretto e non pesato avente n nodi. La matrice di adiacenza di G è **W**. In essa, per costruzione, $W_{i,j}$ = **1** se i vertici (nodi) i e j sono direttamente connessi in G e **0** altrimenti.

Sia **D** una matrice diagonale n x n in cui

$$d_{ii} = \sum_j w_{ij}$$

Il **Laplaciano** del grafo è **L = D - W**

Il **Laplaciano normalizzato** del grafo G è :

$$\tilde{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}} =$$

$$D^{-\frac{1}{2}} D D^{-\frac{1}{2}} - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

dove ***I*** è la matrice identità.

Per ottenere il laplaciano normalizzato del grafo G è sufficiente dividere ogni elemento $w_{ij}$ della matrice di adiacenza ***W*** per la radice quadrata del prodotto delle somme dei pesi in ***W*** degli elementi della riga e della colonna all'intersezione delle quali si trova l'elemento $w_{ij}$ che stiamo normalizzando.

È una sorta di regolarizzazione di **W**

CS

## Random walk kernel

**1-step Random walk kernel** può essere definito utilizzando il laplaciano normalizzato di G :

$$\boldsymbol{K}_{rw} = a\boldsymbol{I} - \tilde{\boldsymbol{L}} = a\boldsymbol{I} - \boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{W}\boldsymbol{D}^{-\frac{1}{2}} = (a-1)\boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{W}\boldsymbol{D}^{-\frac{1}{2}}$$

dove a > 1 .

È una sorta di regolarizzazione di **W**

CS

## Random walk kernel

**p-step Random walk kernel** può essere definito utilizzando $K_{rw}$ :

$$\boldsymbol{K}_{rw}^{q} = (a\boldsymbol{I} - \tilde{\boldsymbol{L}})^{q}$$

dove *q* è un intero che rappresenta il **numero di passi** del random walk sul grafo G . L'implementazione si basa su questa ricorsione:

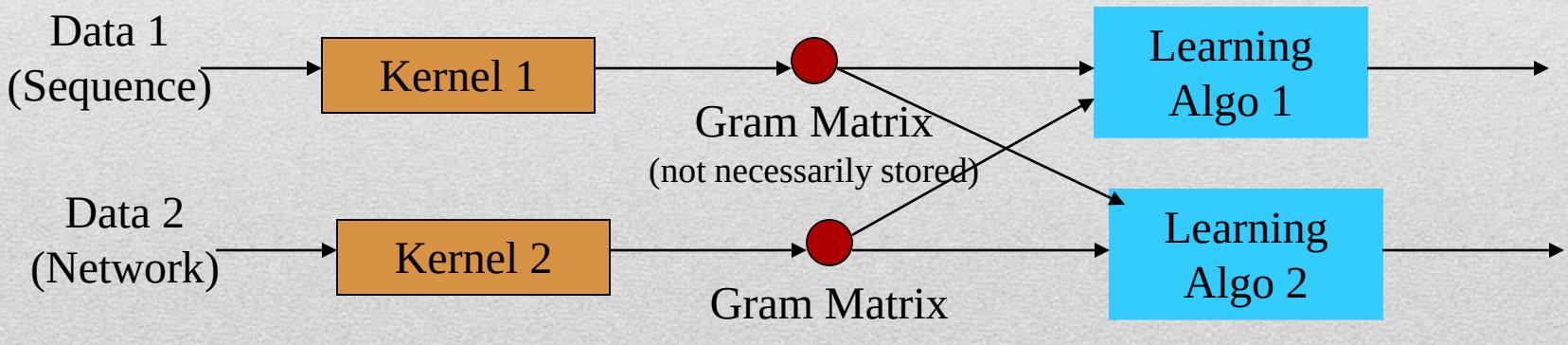
$$\boldsymbol{K}_{rw}^{q} = \boldsymbol{K}_{rw}^{q-1}\boldsymbol{K}_{rw}$$

che applicheremo per ogni *q* >= 2

# VANTAGGI ASSOCIATI ALL'USO DI METODI KERNEL

**Modularità** e **riusabilità**:

- Stesso kernel, diversi algoritmi di learning
- **Kernel diversi**, medesimo algoritmo di learning



CS

## Altri kernel utilizzati in cheminformatica

Fino a questo momento abbiamo visto un kernel definito su grafo (il Random Walk kernel) in grado di esprimere la similarità tra ogni coppia di vertici appartenenti ad un medesimo grafo.

Esistono altri tipi di kernel che permettono di rappresentare nozioni di similarità che coinvolgono grafi. In particolare esistono alcuni kernel in grado di esprimere la **similarità tra grafi**.

Questo può tornarci utile per i test di riposizionamento di farmaci?

CS

## Convolution kernel e kernel spettrali

Dati 2 kernel $k_1$ w $k_2$ definiti sul medesimo set di oggetti X abbiamo diverse opzioni per costruire un nuovo kernel.

La **convoluzione** di $k_1$ e $k_2$ è un nuovo kernel nella forma:

$$k_1 \star k_2 = \sum_{(\mathbf{u}1,\mathbf{u}2)=\mathbf{u};\ (\mathbf{v}1,\mathbf{v}2)=\mathbf{v}} k_1(\mathbf{u}_1,\mathbf{v}_1)\, k_2(\mathbf{u}_2,\mathbf{v}_2)$$

in cui $\mathbf{u} = (\mathbf{u}_1,\mathbf{u}_2)$ denota un partizionamento di $\mathbf{u}$ in 2 sottostrutture $\mathbf{u}_1$ e $\mathbf{u}_2$ .

**Abbiamo già parlato di sottostrutture nelle lezioni precedenti … In che contesto eravamo?**

CS

## Convolution kernel e kernel spettrali

*Abbiamo già parlato di sottostrutture (* ***molecular fingerprints*** *: vettori binari che indicano presenza/assenza di una* ***sottostruttura*** *in una molecola)* **.**

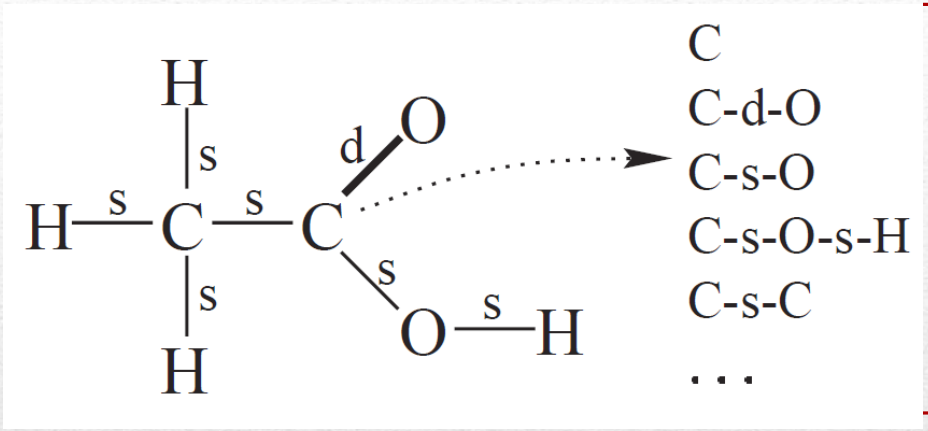
***Le sottostrutture servivano per costruire uno score di similarità da applicare al confronto di molecole.***

*… anche i kernel rappresentano nozioni di similarità tra oggetti (mappati in uno spazio di Hilbert) !*

**E' qui che entra in gioco l'utilità di kernel per il confronto TRA grafi … le molecole POSSONO essere rappresentate come grafi !**

CS

# Tanimoto kernel




**sottostrutture**

**fingerprints**

Molecola rappresentata come **grafo** non diretto etichettato:

- Etichette dei vertici: tipo di atomo
- Etichette degli archi : tipo di legame covalente

**Fissato un intero d consideriamo tutti i possibili percorsi di lunghezza minore o uguale a d . Cosa otteniamo ?**

CS

## Tanimoto kernel

Siano **u**, **v** due molecole e *d* un intero positivo. Consideriamo la funzione di mapping $\Phi_d$ ed il corrispondente kernel $k_d$. Il Tanimoto kernel è definito come:

$$k_d^t(\mathbf{u},\mathbf{v}) = \frac{k_d(\mathbf{u},\mathbf{v})}{k_d(\mathbf{u},\mathbf{u}) + k_d(\mathbf{v},\mathbf{v}) - k_d(\mathbf{u},\mathbf{v})}$$

$\boldsymbol{\Phi_d(u)}$ è interpretabile come **il set di features** (percorsi etichettati di lunghezza <= d , sottostrutture) che possono essere estratti da **u** .

CS

## Tanimoto kernel

Se interpretiamo $\Phi_d(\mathbf{u})$ come **il set di features** che possono essere estratte da **u** ci accorgiamo che Tanimoto kernel calcola il **rapporto** tra le sottostrutture in comune tra **u** e **v** e l'unione delle sottostrutture in **u** e **v** :

$$k_d^t(\mathbf{u},\mathbf{v}) = \frac{|\phi(\mathbf{u}) \cap \phi(\mathbf{v})|}{|\phi(\mathbf{u}) \cup \phi(\mathbf{v})|}$$

Vi ricorda qualcosa?

Dipendentemente dalle scelte di implementazione nel conteggio delle sottostrutture (percorsi etichettati) possiamo ottenere **ESATTAMENTE** lo score di di similarità di Tanimoto !

CS

## Tanimoto kernel

Questo esempio chiarisce il motivo della popolarità dei metodi kernel. In essi il problema della rappresentazione degli oggetti considerati ed il problema dell'apprendimento sono **totalmente separati**.

Avendo a disposizione uno o più kernel validi è possibile applicare il medesimo algoritmo di learning senza cambiare nulla nella sua implementazione.

Ad es. le funzioni kernelizzate possono essere applicate indifferentemente sia ai farmaci rappresentati mediante Random walk kernel che al medesimo set di farmaci rappresentati mediante Tanimoto kernel.

Inoltre è possibile sviluppare **nuovi kernel** specifici per il problema che si vuole affrontare

CS

# Tanimoto kernel

Calcolo **in R** di misure di similarità mediante utilizzo di Tanimoto kernel per un set di molecole:

sd2gramSpectrum *Rchemcpp::sd2gramSpectrum*

**libreria** **funzione**

**Description**

Computes a similarity matrix from molecules by walk-based graph kernels

This function computes several walk-based graph kernel functions based on finite length walks and a fast implementation for input sd file(s).

**scelta del tipo di kernel**

**Usage**

```
sd2gramSpectrum(sdFileName, sdFileName2 = "",
  kernelType = c("spectrum", "tanimoto", "minmaxTanimoto", "marginalized", "lambda"),
  margKernelEndProbability = 0.1, lambdaKernelLambda = 1,
  depthMax = as.integer(0), onlyDepthMax = FALSE,
  flagRemoveH = FALSE, morganOrder = as.integer(0),
  fileType = c("sd", "genericsd", "kcf"),
  silentMode = FALSE, returnNormalized = FALSE)
```

CS

## Tanimoto kernel

Set di molecole:

la funzione **sd2gramSpectrum** richiede in input un file SDF. Questi file sono liberamente scricabili da numerose banche dati pubbliche tra cui DrugBank.

Se volete provare ad usarla potete anche utilizzare direttamente i dati di esempio presenti nella libreria ( guardate l'esempio di utilizzo della funzione nella sua pagina del manuale).

CS

## File utili (paper e manuali)

### Articoli :

kernel su grafi :

- http://enpub.fulton.asu.edu/cseml/07spring/kenel_graphs.pdf

funzioni di score kernelizzate :

- http://homes.di.unimi.it/~valenti/papers/re-mesiti-vale-TCBB.rev2.pdf

kernel in cheminformatica (esempi) :

- http://cbio.ensmp.fr/~jvert/svn/bibli/local/Ralaivola2005Graph.pdf

- http://bioinformatics.oxfordjournals.org/content/21/suppl_1/i359.full.pdf