Slide 1

Per parlare di visualizzazione scientifica bisogna parlare della visualizzazione dei dati.

La visualizzazione scientifica è un sottoinsieme della visualizzazione dei dati.

**IA**: si occupa di rendere fruibile l’informazione a chi ne ha bisogno.

Tutte le ricerche di Wurman hanno portato all’*Information Design*.

La visualizzazione dei dati non è semplicemente visualizzazioni di numeri, ma anche di dati espressi n modo diverso, come ad esempio i grafi.

I grafi sono una rappresentazione comoda dei dati.

Per riuscire a rappresentare dati multimodali, quindi ad alta dimensionalità viene in aiuto la visualizzazione scientifica; se non è di aiuto ci sono delle tecniche alla base dell’informatica come la visulizzazione tramite grafi.

I nodi generalmente sono indicati da un indice.

Slide 2

Gli archi consentono di visualizzare le relazioni tra i nodi.

Gli omini sono i pazienti.

A DX ci sono le caratteristiche del paziente, tipo: ferritina, emoglobina…

Si rappresenta la distanza tra pazienti con una **matrice di similarità** o ***adjacency matrix***.

Gli elementi sulla diagonali sono la similarità, ossia l’inverso della distanza.

Sulla diagonale degli elementi di una matrice di similarità (se è normalizzata ossia può assumere valori solo tra 0 e 1) si hanno tutti 1 in quanto ogni elemento è massimamente simile a sé stesso.

I grafi sono utili per la classificazione dei nodi, quindi per nodi di cui non si ha una chiara conoscenza.

Come si possono classificare i nodi, si possono classificare anche gli edge.

I metodi basati sui grafi in cui si fa una predizione dei nodi di un grafo vengono detti *node prediction problems*, cioè:

sulla base delle conoscenza sugli altri nodi del grafo/degli edge, si fa una predizione su un nodo di cui non si ha conoscenza.

Nel caso si hanno dei nodi e degli edge di cui si è a conoscenza e si voglia predire l’esistenza o meno di un arco, questa predizione viene detta *link prediction problem*.

Node label prediction → si cerca di predire la classe di un nodo

Edge label prediction → si cerca di predire la classe di un edge

Edge prediction/Link prediction → si predige l’esistenza o meno di edge che collegano due nodi

Quindi i grafi sono spesso utili per effettuare tutti questi tipi di predizione.

Slide 3

Un esempio di *link prediction* è quella che viene fatta nei casi di drug repurposing.

( Vedere come i farmaci progettati per curare un certo male, possono curare anche altro.)

È stato utilizzato un grafo di conoscenza che lega il medicinale a tutte le varie malattie.

Gli edge rappresentano diversi tipi di classi. Ogni edge porta una classificazione.

I diversi colori rappresentano concetti provenienti da classi diversi, ossia malattie o tutto ciò che contribuisce a quella malattia.

Un *knowledge graph* riesce a trovare delle relazioni tra diversi nodi.

Un altro esempio di grafo di conoscenza è il GOOGLE GRAPH: l’interserzione dei cammini produce le pagine che vengono poi visualizzate.

[ Questo era il primo algoritmo, una versione molto semplificata. Ora è tutto diverso. ]

Slide 4

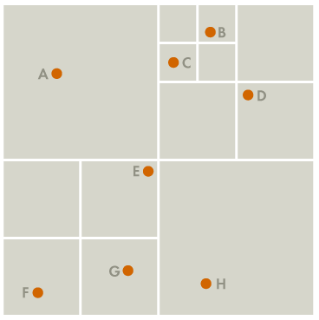
Degree → numero di nodi a cui un nodo di un grafo è connesso

Slide 6

Esempio di algoritmo che utilizza le forze (quella che si vuole, attrattiva o repulsiva) per stabilire la posizione dei nodi sul grafo.

È un algoritmo molto semplice per visualizzare i grafici (ovviamente sono già presenti delle implementazioni, non dobbiamo implementare noi).

È un ***force-directed layout algorithm***.



Questo è uno degli algoritmi di inizializzazione della visualizzazione dei grafi randomica: si parte da un’inizializzazione totalmente casuale dei nodi del grafo.

Si divide un quadrante in 4 quadranti e si analizzano.

Se un quadrante contiene un solo punto, quel punto viene considerato da solo, se contiene più punti quel quadrante viene suddiviso utleoriomente finché non rimane ad avere un solo punto.

La radice della suddivisione in quadranti è ignota.

Si parte da un quadrante che ha 4 figli.

Calcolo delle forze

**INTERNAL nodes** → quadranti che contengono più nodi

**EXTERNAL nodes** → quadranti che contengono un solo corpo

Quindi:

si divide la figura in tanti quadranti finché ogni punto è in un miniquadrante.

In generale ogni quadrante è rappresentato da un unico corpo, che costituisce il centro di massa.

Un quadrante prima di essere ulteriormente suddiviso ha n centro di massa pari al centro di massa dei corpi che contiene:

il centro di massa di questo quadrante è la mediana tra B e C.

Quadranti lontani velocizzano l’algoritmo perché i corpi al loro interno non vengono tutti considerati, ma viene considerato un solo corpo ch è rappresentato dalla media delle loro masse.

Se capisco che i quadranti sono lontani si utilizzano forze repulsive, se sono vicini si utilizzano forze attrattive.

Slide 7

Altri algoritmi che fanno la visualizzazione dei grafi, sono quelli che fanno il loro embedding.

Un nodo è identificato unicamente dalla sua posizione.

Ci si è chiesti: c’è la possibilità di considerare la topologia dei grafi, intesa come relazioni tra nodi legati tra di loro, e fare il loro embedding in uno spazio in cui lo spazio tra ogni nodo sia rappresentato da un insieme di numeri/feature?

Questi metodi che riescono a rappresentare i punti del grafo come vettori all’interno di uno spazio bidimensionale o tridimensionale, si chiamano metodi di embedding.

Embedding/proiezione dei punti su uno spazio vettoriale

GRAPH EMBEDDING: proiezione di un grafo su un autovettore.

Cosa sono gli autovettori di una matrice?

La matrice rappresenta una trasformazione in uno spazio.

Si possono vedere tutte le colonne della matrice come dei vettori su cui i punti di un’altra matrice possono essere proiettati.

• Gli autovettori sono l’insieme di assi ortogonali nello spazio vettoriale che spiegano sostanzialmente la variabilità dello spazio vettoriale.

• Gli autovalori dicono quanto ogni singolo autovettore è importante.

Un autovettore che è 0 dice che il suo corrispondente autovettore è 0.

Slide 8

Se si calcolano gli autovettori della matrice di adiacenza/Laplaciana, si scarta l’autovettore che ha come autovalore 0 (in quanto non porta alcuna informazione all’interno dello spazio vettoriale), si allineano i rimanenti autovettori verticalmente ed il j-esmo nodo è rappresentato dal j-esimo elemento di tutti gi autovettori.

Quindi:

il primo nodo è rappresentato da tutti i primi elementi degli autovettori.

Se si ha una rappresentazione vettoriale, si possono prendere i 3 autovettori che hanno l’autovalore maggiore.

Ciò vuol dire che si ha una miglior descrizione dello spazio descritto dalla Laplaciana ed i nodi verrano rappresentati tra numeri e questi numeri possono essere plottati magari con uno scatter plot.

Questo è un modo per visualizzare i grafi.

Si è trasformato il grafo dall’essere un insieme di punti all’essere un insieme di punti intesi come vettori in uno spazio che decidiamo noi di che dimensione deve essere.

Ovviamente più aumentano i vettori, più aumenta la “risoluzione”.

Per dire se si hanno 2 dimensioni, i nodi, quando verranno plottati su uno spazio bidimensionale, si sovrappongono, in uno spazio tridimensionale invece magari no.

Slide 9

DeepWalk

È un metodo di embedding che risulta comodo non solo quando si vogliono considerare i nodi per i loro attributi, ma anche quando si vuole considerare la topologia del grafo, quindi la sua struttura/le connessioni tra nodi vicini.

Si basa sul concetto che nodi vicini devono anche star vicini nello spazio di embedding, nello spazio vettorial in cui vengono proiettati.

**k** è scelto da noi in modo empirico.

I nodi vicini sono dei nodi raggiungibili camminando randomicamente nel grafo.

Random walk: parto dal nodo di cui voglio calcolare i vicini e l’edge con il lancio di moneta più alto si percorre, finché non si vede a che vicino si arriva.

Questi cammini permettono di conoscere il vicinato; più l è alto più saranno i nodi che si considerano come vicini.

Parametro k

Se si fa un numero maggiore di *random walk* si conoscono meglio i vicini: quelli che sono a pochi passi hanno meno possibilità di essere miei vicini.

Si viaggia in modo randomico e si può tornare indietro → quindi c’è la possibilità di entrare in un ciclo.

Slide 10

Cosa farsene di k random walk?

Per ogni nodo di partenza si hanno tutti i nodi che sono attarversati da quella random walk e si ha il nodo di arrivo.

DeepWalk utilizza l’algoritmo word2vec (nato dal *Natural Language Processing* – **NLP**) che trasforma le parole in vettori.

È stato realizzato inizialmente per le traduzioni, poi è stato presto utilizzato per mappare ogni parola in un vettore.

Utilizza lo stesso concetto di DeepWalk, parole che sono vicine in un testo e quindi hanno lo stesso significato, appaiono nello stesso contesto, dovrebbero mappare vicine nello spazio vettoriale in cui si trasformano.

Si analizzano i documenti e si analizza il contesto di ogni parola.

Mentre nel testo le *random walk* sono obbligate poiché si passa da una parola all’altra, nei grafi non si ha una linea di testo quundi non si può prendere tutto l’insieme di parole che rappresentano il vicinato, quindi si fa una random walk che parte da un nodo ed arriva ad altri nodi.

Dato un corpus, ossia un documento di testo, e data la larghezza di una finestra, l’algoritmo SkipGram trova la codifica delle parole che consente di massimizzare la probabilità di parola che appaiono nello stesso contesto.

Slide 11

SkipGram, allo stesso modo, quando applicato ad un grafo, crea dei vettori per ogni nodo che sono tanto simili quanto i nodi sono vicini tra di loro.

Slide 12

Sia nel caso delle parole che nel caso dei nodi, SkipGram è una rete neurale, la più semplice.

Una rete neurale ha:

- un input layer → ogni elemento del vettore di input è visto come un neurone di input.

C’è un 1 in corrispondenza della parola;

- tanti hidden layer che ha diversi neuroni, detti **lineari**;

- sugli archi viaggiano dei pesi, che trasformano il valore dell’input load, lo molitplicano al valore dell’input che corre sul ramo e lo portano all’hidden layer.

All’interno del nodo c’è la somma pesata. Prende i valori, li fa passare all’hidden layer.

L’output layer ha la stessa dimensione del dizionario in ingresso.

Sostanzialmente il training di una rete neurale (il suo apprendimento) si basa su una serie di esempi.

**Tecnica di apprendimento supervisionata**

Passo al primo esempio e si vede se il primo esempio dice che il primo nodo è stato raggiunto da una random walk.

Ogni random walk parte da un nodo ed arriva ad un altro, considerato il vicino.

Dopo che la rete continua a vedere per n volte k random walk i pesi sui rami diminuiscono.

Il training della rete porta ad avere una matrice che è della dimensione del numero di neuroni nell’hidden layer.

Basta che per ogni nodo prendo tutti i pesi che entrano nel nodo a lui riferito.

SkipGram è un esempio di rete che viene utilizzata per l’embedding.

È la più semplice poiché ha un solo hidden layer.

Se si utilizza un hidden layer con 25 nodi si avrà una rappresentazione 25-dimensionale.

Slide 19

In Node2vec si è cambiato solamente l’ordine delle random walk, ossia sono di 2° ordine.

**GraphX** per provare random walks.

Si vuole che punti vicini siano massimamente vicini e punti lontani siano massimamente lontani.