Lezione 3

Corso	Riconoscimento e recupero dell'informazione per Bioinformatica
Data	@October 18, 2021 1:40 PM
Rifacimento	☑
Status	Completed
Tipo	Lezione

Data trasformation

> 💡 Cercare di creare una rappresentazione più adeguata.

Differenza con la standardizzazione: nella data standardization tutte le operazioni sono implementate dimensione per dimensione, nella data transformation le operazioni agiscono su tutte le dimensioni contemporaneamente (tipicamente in modo lineare)

Esempio più semplice: supponendo di avere un punto $o \begin{bmatrix} x & y \end{bmatrix}$ suppongo di avere un nuovo spazio in cui la dimensione è $\begin{bmatrix} x+y & x-y \end{bmatrix}$, la prima nuova feature non dipende solo dalle x del dataset ma anche dalla y.

Costruisco nuove feature sfruttando le precedenti

Esempio

> Schizofrenia: esiste un cambio strutturale nel cervello? In questo caso la domanda è posta in questi termini: abbiamo un po' di pazienti e le loro risonanze magnetiche e l'obiettivo era discriminare tra sani e malati.

Approccio classico:

> Trasformazione lineare dello spazio delle features.
>
> Ogni nuova feature è una combinazione lineare di tutte le feature precedenti
>
> Una combinazione lineare si può scrivere come prodotto tra vettori.

Provo a vedere se riesco, visualizzando i dati, a fare una discriminazione. Non posso fare visualizzazione solo dei volumi.

La trasformazione lineare dello spazio delle features è quindi una moltiplicazione tra matrici

$$Y = A^T * X$$

Questa matrice mi dice come fare a trasformare i dati.

X contiene gli oggetti del problema (per colonna), A contiene i vettori utilizzati per la trasformazione (una colonna per ogni dimensione), Y è lo spazio dopo la trasformazione

Per fare la riduzione della dimensionalità ho due approcci:

- **non supervisionati** → uso solo i dati (cerco di comprimere)
- **supervisionati** → uso anche l'informazione di cosa voglio fare
  - Es. Problema di classificazione, non voglio estrarre in generale features, ma estrarne di funzionali al mio task e uso questa informazione all'interno del modo in cui riduco la dimensione.

Ci sono moltissime tecniche che fanno questa cosa:

vediamo la **PCA**.

Principal Component Analysis

Approccio lineare non supervisionato.

Obiettivo: trovare la matrice A ($Y = A^T * X$)

Supponiamo di avere una nuvola di punti

La componente principale ha più valore: ha una varianza più ampia. Perché? Due punti che hanno una componente dello stesso valore, una volta effettuata la riduzione delle dimensioni, collassano sulla stessa rappresentazione. Ciò succede di meno se ci sono dati più variabili → con una varianza maggiore.

Idea principale: l'informazione più importante da mantenere è la varianza dei dati!

- Estrae le direzioni di massima varianza dei dati.

- Mantiene anche la maggior aderenza ai dati originali
- Minimizza lo scarto quadratico medio tra i dati originali e quelli ricostruiti

La direzione migliore per la **pca** è quindi quella a massima varianza (ragionando anche sul range si arriva allo stesso risultato)

> esempio
>
> Compressione a pardita: perdo informazioni nella compressione (per ridurre al massimo la descrizione dell'oggetto) → es. jpeg
>
> La differenza tra l'immagine originale e quella compressa, mi misura quanta informazione ho perso. Se misuro l'errore quadratico medio, la PCA è la scelta migliore per minimizzare lo scarto quadratico medio (non posso fare meglio della PCA)

Come funziona la PCA?

La **prima** componente che estrae è quella a massima varianza.

La **seconda** componente principale è quella ortogonale alla prima, di massima varianza.

Come si realizza PCA?

Esiste un **algoritmo** specifico che ci permette di trovare direttamente la matrice di trasformazione A che si basa sulla creazione di autovalori e autovettori della matrice di trasformazione.

- **Dataset** X

$$X = \begin{bmatrix} x_{1,1} & x_{\dots} & x_{1,n} \\ x_{\dots,1} & x_{\dots} & x_{\dots} \\ x_{d,1} & x_{\dots} & x_{d,n} \end{bmatrix}$$

n punti in uno spazio d-dimensionale

$$A = \begin{bmatrix} 3 & 4 & 0 \end{bmatrix}$$
$$B = \begin{bmatrix} 7 & 8 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

$$X = \begin{bmatrix} 3 & 7 & 1 & 3 \\ 4 & 8 & 2 & 4 \\ 0 & 3 & 4 & 2 \end{bmatrix}$$

Obiettivo: trovare uno spazio ottimale di dimensione L, con $L < d$

Lo spazio è definito come:

"

quindi **trovare A**.

Nel particolare quindi voglio ridurre la dimensionalità a L dimensioni.

Nel nostro caso ho $d = 3$, se voglio arrivare a $L = 2$, voglio:

$$Y = \begin{bmatrix} a & b & c & d \\ e & f & g & h \end{bmatrix}$$

Riduco a due le dimensioni, mantenendo n invariato ovviamente.

A dovrà quindi essere:

$$A = \begin{bmatrix} a^1 & b^1 \\ c^1 & d^1 \\ e^1 & f^1 \end{bmatrix}$$

Algoritmo PCA

1. **Primo passo:**
   calcolare la media lungo ogni direzione

   $$U_m = \frac{1}{N} \sum_{n=1}^{N} x_{m,n}$$

   Questo vettore U contiene tutti gli elementi:

   $$U = \begin{bmatrix} u_1 \\ ... \\ u_d \end{bmatrix}$$

2. **Secondo passo:**

   centrare i dati

   Voglio far si che la media diventi zero

   $B = X - Uh$

   dove $h = \begin{bmatrix} 1 & .... & 1 \end{bmatrix}$ e sono n valori

3. **Terzo passo:**
   trovare la matrice di covarianza

$$C = \frac{1}{N-1} B * B^T$$

   $\frac{1}{N-1} \sum (x - \mu_x)(y - \mu_y)$ ← per ricordare la covarianza

4. **Quarto passo:**
   trovare autovalori e autovettori di C

   Definizioni per ricordare: $Cx = \lambda x$, con $x \neq 0$, allora x = autovalore di C e λ = autovettore di C.

   $C = dxd$

   ho d autovalori $\lambda_1, ..., \lambda_d$

   e d autovettori $v_1, ..., v_d$

5. **Quinto passo:**
   ordinare gli autovalori

6. **Sesto passo:**
   costruire la matrice A

   La matrice si ottiene con gli L autovalori corrispondenti agli L autovalori più grandi

Le colonne di A in numero sono pari alle dimensioni che voglio ottenere (L), invece le righe sono sempre d.

Vantaggi e Svantaggi di PCA

Vantaggi

- Migliore tecnica lineare di riduzione della dimensionalità di un insieme di dati

    - migliore in senso di “errore quadratico medio”
- I parametri del modello possono essere ricavati direttamente dai dati
    - Io ho un algoritmo e se lo seguo ottengo la matrice, tanti altri metodi non hanno questa possibilità (hanno bisogno di ottimizzazione o addirittura operazioni più complesse)
- La proiezione nello spazio è un'operazione molto veloce (moltiplicazione di matrici)
- Tecnica molto conosciuta e utilizzata (permette di avere una prima idea sui dati)

Svantaggi

- Alto costo computazionale per il calcolo dei parametri del modello (soprattutto in caso di dimensionalità elevata)
- Non è chiaro come questa tecnica possa gestire il caso di dati incompleti
    - Grosso problema soprattutto in medicina
- PCA non tiene conto della densità di probabilità dello spazio considerato (viene considerata solo la vicinanza del vettore trasformato al vettore originale)
- Non è detto in tutti i casi che le direzioni a varianza maggiore siano le direzioni ottimali

Feature Selection

Approccio alternativo alla riduzione della dimensionalità

Scelgo semplicemente le feature.

- **Vantaggio**:
  spesso alcune features sono ridondanti, non informative o addirittura dannose
- **Svantaggio**:
  spesso è computazionalmente oneroso, occorre trovare il miglior sottoinsieme

Problemi da risolvere

- scegliere il criterio di ottimalità: informazione (come si misura?), varianza, capacità di classificazione (solo per problemi supervisionati)

- come trovare il sottoinsieme ottimale senza provarli tutti (troppo oneroso computazionalmente)

Ho un criterio di ottimalità (es. l'accuratezza del nearest mean classifier), faccio un ranking e trovo la migliore, poi formo le coppie tenendo fissa la prima. Ad ogni passo faccio la scelta migliore localmente → **greedy**

Forward Sequential Feature Selection: algoritmo greedy per trovare l'insieme ottimale di features

Schema:

- si valuta il criterio per tutte le features singolarmente
- si sceglie la feature che massimizza il criterio (chiamata f_1)
- si valuta il criterio per tutte le coppie (f_1, f_x) – cioè tenendo fissata f_1
- si sceglie la coppia che massimizza il criterio (la coppia (f_1, f_2))
- si valuta il criterio per tutte le terne (f_1, f_2, f_x) – cioè tenendo fissate f_1 e f_2
- ...

Algoritmo greedy – ad ogni istante la scelta migliore: computazionalmente efficiente ma sub ottimale
Esempio neuropatologie: Da un certo punto in poi aggiungere feature semplicemente crea rumore (disturba il classificatore).