Estimação, Deteção e Aprendizagem II

Redução da dimensionalidade

Catarina Oliveira

DCT DEPARTAMENTO CIÊNCIA E TECNOLOGIA

UPT UNIVERSIDADE PORTUCALENSE

CONTEÚDO

1. Principal Component Analysis (PCA)
2. Linear Discriminant Analysis (LDA)
3. Self-Organizing Map (SOM)

Principal Component Analysis (PCA)

PCA

Transforma um *dataset* com muitas variáveis num *dataset* com menos variáveis com (quase) a mesma informação

Passos:

1. Standardização
2. Computação da matriz de covariância
3. Computação dos *eigenvectors* e *eigenvalues* da matriz de covariância para identificar as componentes principais
4. *Feature vector*
5. Remodelar os dados ao longo dos eixos dos componentes principais

PCA: 1 – Standardização

Objetivo: standardizar o intervalo das variáveis iniciais contínuas para que cada uma delas contribua igualmente para a análise

Razão: O PCA é muito sensível às variâncias das variáveis iniciais

Processo: para cada observação v_i de cada variável v com média μ e desvio padrão σ, o seu valor passa a ser:

$$z = \frac{v_i - \mu}{\sigma}$$

PCA: 2 – Computação da matriz de covariância

Objetivo: entender como as variáveis do conjunto de dados de entrada variam da média entre si, ou seja: para ver se há alguma relação entre elas.

Razão: às vezes as variáveis são altamente correlacionadas de tal forma que contêm informações redundantes.

Processo: para cada variável v_i do *dataset*, calcular a matriz de covariância:

$$\begin{bmatrix} Cov(v_1, v_1) & Cov(v_1, v_2) & \cdots & Cov(v_1, v_n) \\ Cov(v_2, v_1) & Cov(v_2, v_2) & \cdots & Cov(v_2, v_n) \\ \vdots & \vdots & & \vdots \\ Cov(v_n, v_1) & Cov(v_n, v_2) & \ldots & Cov(v_n, v_n) \end{bmatrix}$$

Sinal da covariância:

- Positivo: as variáveis são correlacionadas
- Negativo: as variáveis são inversamente correlacionadas

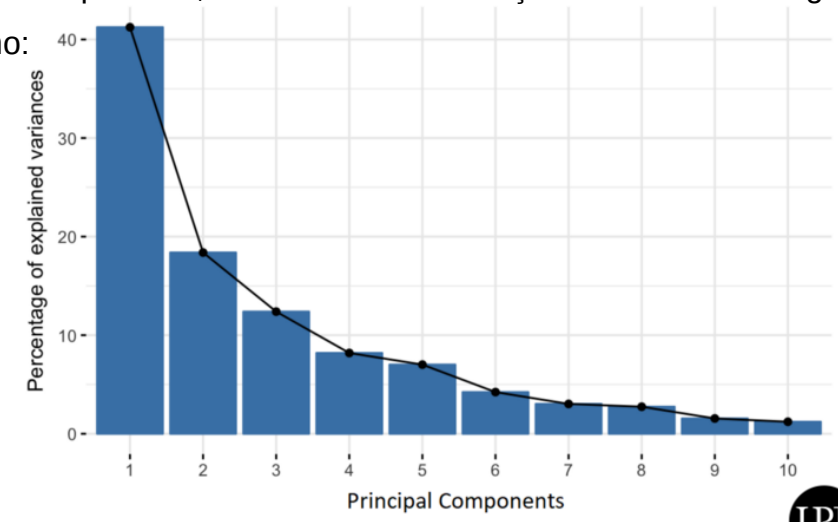
PCA: 3 – Computação dos *eigenvectors* e *eigenvalues*

Eigenvectors e *eigenvalues*: conceitos de álgebra linear calculados a partir da matriz de covariância para determinar os componentes principais dos dados

Componentes principais: novas variáveis construídas como combinações lineares ou misturas das variáveis iniciais, feitas de forma que:

- As novas variáveis (componentes principais) não estão correlacionadas
- A maioria das informações dentro das variáveis iniciais é comprimida nos primeiros componentes.

Exemplo: um *dataset* com 10 dimensões fornecem 10 componentes principais, mas o PCA tenta colocar o máximo de informações possíveis no primeiro componente, o máximo de informações restantes no segundo e assim sucessivamente, até ter algo como:



PCA: Componentes principais

A organização dos principais componentes desta forma permite reduzir a dimensionalidade sem perder muita informação, descartando os componentes com pouca informação e considerando os restantes como novas variáveis

Nota: os componentes principais são menos interpretáveis e não têm significado real, já que são construídos como combinações lineares das variáveis iniciais

Geometricamente, os componentes principais representam a direção dos dados que explicam a maior parte da variância, ou seja: as linhas que capturam a maioria da informação dos dados.

- Quanto maior a variância de uma linha, maior a dispersão dos dados ao longo dessa linha
    - Quanto maior a dispersão, mais informação tem

Os componentes principais podem ser vistos como novos eixos que mostram o melhor angulo para ver e avaliar os dados para que as diferenças entre as observações sejam mais visíveis

PCA: construção dos componentes principais

Os *eigenvectors* da matriz de covariância são as direções dos eixos onde está a maior variância (mais informação), chamados componentes principais.

Os *eigenvalues* são coeficientes dos *eigenvectors*, e dão a quantidade de informação de cada componente principal

Ordenando os *eigenvectors* pela ordem dos seus *eigenvalues*, do maior ao menor, obtém-se os componentes principais por ordem de significância.

PCA: 4 – Feature vector

Depois de computar os *eigenvectors* e ordená-los pelos *eigenvalues* decrescentemente, encontrando os componentes principais por ordem de significância

Determinar se vamos manter todos os componentes (n) ou descartar os que têm significância mais baixa (menores *eigenvalues*), formando com os restantes (p - não descartados) uma matriz de vetores a que chamamos *feature vector*

Feature vector: matriz que tem como colunas os *eigenvectors* que decidimos manter

Redução da dimensionalidade: escolhemos manter apenas p componentes dos n iniciais

PCA: 5 – Remodelar os dados ao longo dos eixos dos componentes principais

Utilizar o *feature vector* formado usando os *eigenvectors* da matriz de covariância para reorientar os dados dos eixos originais para os representados pelos componentes principais

Processo: multiplicar a transposta do *dataset* original pela transposta do *feature vector*

$$DatasetFinal = FeatureVector^T \times DatasetOriginalStandardizado^T$$

Linear Discriminant Analysis (LDA)

LDA

Linear Discriminant Analysis (LDA): algoritmo de previsão para classificação *multi-class*

Usado para redução de dimensionalidade, providenciando uma projeção do *dataset* que melhor separa (discrimina) os exemplos pelas classes

Tenta encontrar uma combinação linear de variáveis de input que permite a máxima separação de amostras entre as classes (calculando centroides ou médias) e separação mínima de amostras dentro de cada classe

Podemos usar LDA para calcular a projeção do *dataset* e selecionar um número de dimensões ou componentes da projeção para usar como input para um modelo

Algoritmo LDA

Assume:

- Os dados seguem uma distribuição de Gauss
- Cada atributo tem a mesma variância: valores de cada variável variam à volta da média, em média, o mesmo valor

Processo:

- Estima a média μ_k e a variância σ^2 dos dados para cada classe *k*

Previsão:

- Estima a probabilidade (usando teorema de Bayes) de um novo input pertencer a cada classe. A classe com maior probabilidade é prevista.

LDA para redução da dimensionalidade em python

```
# prepare dataset
data = ...

# define transform
lda = LinearDiscriminantAnalysis()

# prepare transform on dataset
lda.fit(data)

# apply transform to dataset
transformed = lda.transform(data)
```

Self-Organizing Map (SOM)

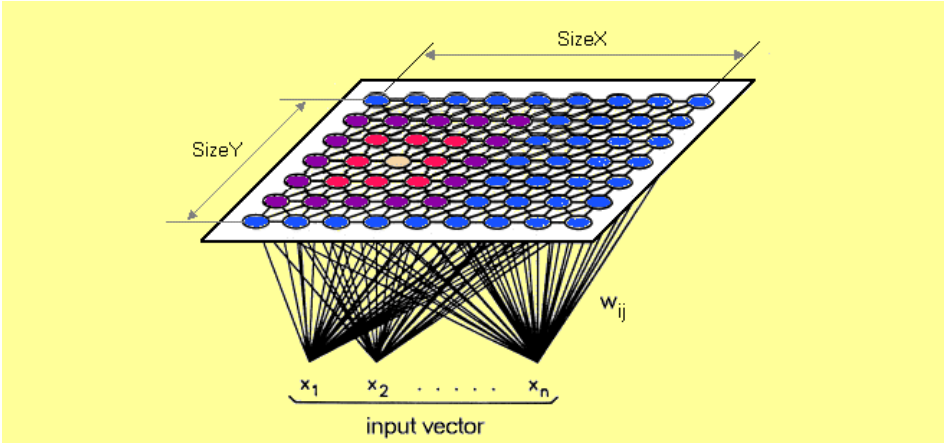
Self-Organizing Map (SOM)

Tipo de rede neuronal artificial treinada usando *unsupervised learning* para produzir uma representação com menos dimensões (geralmente duas) com base num *input space*

Diferente de redes neuronais porque aplicam *competitive learning* em vez de *error-correction lerarnig* (ex: *backpropagation* com *gradient descent*) e usam uma função de vizinhança para preservar as propriedades topológicas do *input space*

Self-organizing map = Kohonen map

Kohonen, T. Self-organized formation of topologically correct feature maps. Biol. Cybern. 43, 59–69 (1982). https://doi.org/10.1007/BF00337288



Algoritmo SOM

1. Os pesos de cada nó são inicializados.
2. Um vetor é escolhido aleatoriamente do conjunto de dados de treino.
3. Cada nó é examinado para calcular que pesos são mais parecidos com o vetor de entrada. O nó vencedor é chamado *Best Matching Unit* (BMU).
4. A vizinhança do BMU é calculada. A quantidade de vizinhos diminui com o tempo.
5. O peso vencedor é recompensado tornando-se mais parecido com o vetor escolhido em 2. Os vizinhos também se tornam mais parecidos com esse vetor. Quanto mais próximo um nó está do BMU, mais seus pesos são alterados e quanto mais longe o vizinho está do BMU, menos ele aprende.
6. Repetir o passo 2 durante N iterações

Best Matching Unit: técnica que calcula a distancia (ex: euclidiana) de cada peso ao vetor escolhido. O peso com a menor distância é o vencedor.

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.