Projeto 3

Redes neurais

Vitor Boschi da Silva e Felipe Tassario Gomes

19 de novembro de 2015

1 Kohonen - Mapa Auto-Organizável

A primeira parte do projeto consistia em utilizar um Mapa Auto-Organizável de Kohonen para realizar o agrupamento de um banco de dados sobre informações de vinhos de uma certa região da Itália.

Os dados consistiam da análise de 178 amostras de vinhos obtidas de 3 vinhedos diferentes. Cada amostra de vinho teve 13 paramêtros medidos, sendo estes paramêtros diferentes caraterísticas químicas da amostra (podendo ser representada por números inteiros ou reais). Além disso, na entrada também é dado um identificador (0, 1 ou 2) representando o vinhedo de origem desta amostra.

A tarefa consistia em rodar um rede de Kohonen para visualizar a saída do mapa. Idealmente, espera-se que o agrupamento das amostras na saída mostre similaridades entre amostras de vinhos obtidas de um mesmo vinhedo.

Uma implementação em C++ da rede de Kohonen foi feita. Como paramêtros, utilizou-se um mapa 15x15, e o treinamento foi rodado por 100 épocas. A saída obtida é mostrada na Figura 1:

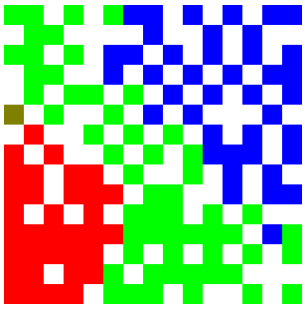

Figura 1: Saída do mapa de Kohonen para dados de vinhos.txt.

Como podemos inspecionar visualmente, as amostras de cada vinhedo (cada uma com uma cor atribuída), se agruparam espacialmente pelo mapa.

2 PCA - Análise de Componente Principal

A PCA é uma ferramenta matemática utilizada para analisar dados multi-dimensionais e determinar a colaboração de cada dimensão na variância dos dados. A PCA é comumente utilizada para transformar linearmente os dados originais numa nova base N-Dimensional (com N menor ou igual à dimensão original), sendo que a contribuição à variância de cada dimensão seja a maior possível.

Além disso, a PCA informa à contribuição de cada uma dessas dimensões. Isso possibilita que descartemos dimensões que contribuem pouco para à variância, reduzindo assim a dimensão dos dados, de uma forma informada (sabendo o erro gerado ao descartar cada dimensão).

A realização da PCA consiste no seguinte algoritmo:

- Normalização dos dados para uma distruibuição Gaussiana, com média = 0 e variância = 1
  - Isto pode ser feito subtraindo a média de todos os valores e os dividindo pelo desvio padrão
- Obtenção da matriz de covariância dos dados normalizados
- Obtenção dos auto-valores e auto-vetores da matriz de covariância
  - Os auto-valores ordenados representam a contribuição à variância daquela dimensão
- Escolhe-se a quantidade de dimensões a serem mantidas e utiliza-se os auto-vetores como bases ortogonais para a transformação da matriz original.

2.1 PCA clássica no conjunto iris.dat

Na primeira parte do segundo exercício, realizou-se a PCA sobre o conjunto iris.dat. As contribuições obtidas das dimensões, calculadas a partir dos auto-valores, foram:

Tabela 1: Contribuição de cada componente de acordo com a PCA

	1a dim.	2a dim.	3a dim.	4a dim.
Colaboração	72.77%	23.03%	3.68%	0.52%
Cumulativa	-	95.80%	99.48%	100%

Utilizando as duas primeiras componentes (o que nos dará uma variância cumulativa de 95.80%, ou seja, um erro de 4.20%), convertemos os dados para uma nova base bi-dimensional e plotamos os resultados, como pode ser visto na Figura 2.

2.2 Rede PCA adaptativa no conjunto iris.dat

Para a realização dessa tarefa, implementou-se o algoritmo da PCA adaptativa na linguagem C++. A PCA adaptativa é implementada na forma de uma rede neural,

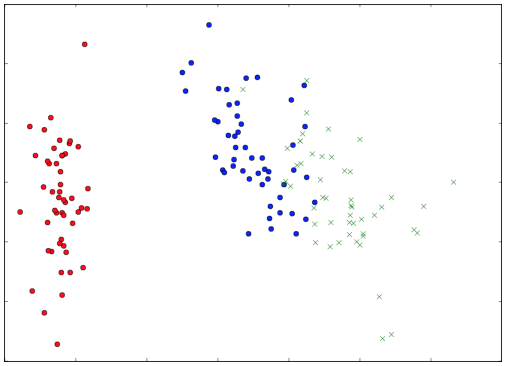

Figura 2: Saída de duas dimensões com a PCA clássica.

que contem N neurônios de entrada e M de saída. Os neurônios na entrada estão completamente ligados à todos os neurônios na saída, e os pesos de cada ligação é representado por uma matriz NxM. Além disso, cada neurônio na saída está ligado aos neurônios de saída posteriores à ele (por exemplo, o neurônio 1 está ligado aos neurônios 2, 3 e 4; o 2, aos neurônios 3 e 4; etc).

A atualização dos pesos é dado através da Regra de Hebb.

Com a implementação pronta, o algoritmo foi executado utilizando 4 entradas (de acordo com a dimensão do conjunto de dados Iris), e 2 saídas (para compararmos com o resultado anterior, e também plotarmos a saída num gráfico bi-dimensional). A saída obtida pode ser visualizada na Figura 3.

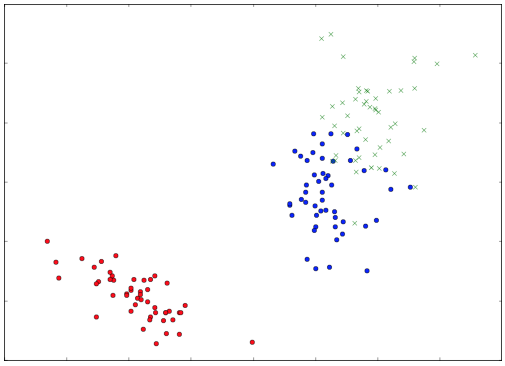

Figura 3: Saída de duas dimensões com a PCA clássica.

3 MLP nos resultados

Por último, após finalizar a PCA, realizou-se a classificação dos dados com a utilização de uma MLP. Essa classificação foi feita utilizando tanto os dados obtidos pela PCA, quanto os dados originais, com o propósito de comparar e verificar a qualidade da saída do PCA.

Para a criação da MLP, utilizou-se a biblioteca pybrain do Python. O conjunto de dados foi dividido em 75% para treino e 25% para teste, e foi utilizada uma rede com uma camada invisível de 5 neurônios. A saída possui 3 neurônios, sendo cada um responsável pela ativação de um grupo (vinhedo) dos dados. O learning rate foi fixado em 0.5 e os critérios de parada foram:

- 1000 épocas; ou
- erro <0.01

Com isso, obtivemos os seguintes resultados:

Tabela 2: MLP sobre resultados

	Dimensões	Taxa de erro	Acerto (1 - erro)
Dados originais	4	5.26%	94.74%
PCA clássica	2	10.52%	89.48%
PCA adaptativa	2	31.57%	68.43 %