CEFET/RJ
Programa de Pós-graduação
em Ciência da Computação
Aprendizado de Máquina - Trabalho 03

Prof. Eduardo Bezerra (ebezerra@cefet-rj.br)

Agosto/2018

Conteúdo

1 Introdução

Neste trabalho, você usará filtragem colaborativa para criar um sistema de recomendação para filmes. Vocêw também irá realizar experimentos no contexto do aprendizado de comitês.

2 Sistemas de Recomendação

Nesta parte, você implementará o algoritmo de aprendizagem de filtragem colaborativa e aplicá-lo-á a um conjunto de dados de avaliações de filmes[1]. Este conjunto de dados consiste em classificações em uma escala de 1 a 5. O conjunto de dados tem $n_u = 943$ usuários e $n_m = 1682$ filmes.

2.1 Conjunto de dados de classificações de filme

O arquivo que contém o conjunto de dados tem nome ex8_movies.mat, e contém as variáveis Y e R. A matriz Y (de ordem *número de filmes* x *número de usuários*) armazena as classificações $y^{(i,j)}$ (de 1 a 5). A matriz R é uma matriz de indicadores de valor binário, onde $R(i, j) = 1$ se o usuário j forneceu uma classificação para o filme i e $R(i, j) = 0$ em caso contrário. O objetivo da filtragem colaborativa é prever as classificações de filmes para os filmes que os usuários ainda não classificaram, ou seja, as entradas com $R(i, j) = 0$. Isso permitirá recomendar os filmes com classificações mais altas previstas para outros usuários.

Ao longo desta parte, você também irá trabalhar com as matrizes X e $Theta$. Essas variáveis se encontram no arquivo ex8_movieParams.mat. A i-ésima linha de X corresponde ao vetor de característica $x^{(i)}$ para o i-ésimo filme. A j-ésima linha de $Theta$ corresponde a um vetor de parâmetros $\theta^{(j)}$, para o j-ésimo usuário. Tanto $x^{(i)}$ quanto $\theta^{(j)}$ são vetores n-dimensionais. Para os fins deste exercício, você usará $n = 100$ e, portanto, $x^{(i)} \in \mathbb{R}^{100}$ e $\theta^{(j)} \in \mathbb{R}^{100}$. Correspondentemente, X é uma matriz $n_m \times 100$ e $Theta$ é uma matriz $n_u \times 100$.

2.2 Algoritmo de aprendizagem de filtragem colaborativa

Agora, você vai começar a implementar o algoritmo de filtragem colaborativa. Você iniciará implementando a função de custo (sem regularização). O algoritmo de filtragem colaborativa no contexto das recomendações de filmes considera um conjunto de vetores de parâmetros n-dimensionais $x^{(i)}, \ldots, x^{(n_m)}$ e $\theta^{(1)}, \ldots, \theta^{(n_u)}$, onde o modelo prediz a avaliação para o filme i pelo usuário j como $y^{(i,j)} = (\theta^{(j)})^T x^{(i)}$. Dado um conjunto de dados que consiste em um conjunto de avaliações produzidas por alguns usuários para alguns filmes, o algoritmo deve aprender os vetores de parâmetros $x^{(i)}, \ldots, x^{(n_m)}$ e $\theta^{(1)}, \ldots, \theta^{(n_u)}$ que produzem o melhor ajuste (i.e., minimizam o erro quadrático).

Você deverá implementar código em um arquivo de nome `cofi_cost_func.py` para calcular a função de custo e o gradiente para a filtragem colaborativa. Os

[1] `https://grouplens.org/datasets/movielens/`

parâmetros para a função (ou seja, os valores que você está tentando aprender) devem ser X e $Theta$.

2.2.1 Função de custo da filtragem colaborativa

A função de custo para a filtragem colaborativa (sem regularização) é dada por

$$J(x^{(i)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2.$$

Você deve implementar código em `cofi_cost_func.py` para retornar esse custo em uma variável de nome J. Note que você deve acumular o custo para o usuário j e o filme i somente se $R(i, j) = 1$.

Depois de implementar a função, você deve testá-la. Você deve ver uma saída de 22,22.

2.2.2 Gradiente de filtragem colaborativa

Agora, você deve implementar o gradiente (sem regularização). Especificamente, você deve implementar código em cofiCostFunc.py para retornar as variáveis X_grad e $Theta\_grad$. Observe que X_grad deve ser uma matriz do mesmo tamanho que X e, de forma semelhante, $Theta\_grad$ é uma matriz do mesmo tamanho que $Theta$. Os gradientes da função de custo são dados por:

$$\frac{\partial J}{\partial x_k^{(i)}} = \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)}$$

$$\frac{\partial J}{\partial \theta_k^{(j)}} = \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(j)}.$$

A função deve retornar o gradiente para ambos os conjuntos de variáveis, armazenando-os em um único vetor.

2.3 Aprendizado de Recomendações para Filmes

Depois de concluir a implementação do gradiente e da função de custo de filtragem colaborativa, agora você pode começar a treinar seu algoritmo para fazer recomendações de filmes. Para testar se o treinamento funcionou, você pode definir suas próprias preferências de filmes, para que mais tarde, quando o algoritmo para executado, você possa obter suas próprias recomendações de filmes. A lista de todos os filmes e seus números no conjunto de dados pode ser encontrada no arquivo `movie_ids.txt`.

Depois que você definir suas para alguns filmtes e adicioná-las ao conjunto de dados, o script continuará a treinar o modelo de filtragem colaborativa. Isso deve fazer com que os parâmetros X e Theta sejam aprendidos. Para prever a classificação do filme i para o usuário j, você precisa calcular $(\theta^{(j)})^T x^{(i)})$. Você deve implementar um script para calcular as classificações principais para

Principais recomendações: Previsão de avaliação 9.0 para Titanic (1997) Previsão de avaliação 8.9 para Star Wars (1977) Previsão de avaliação 8.8 para Shawshank Redemption, The (1994) Previsão de avaliação 8.5 para As Good As It Gets (1997) Previsão de avaliação 8.5 para Good Will Hunting (1997) Previsão de avaliação 8.5 para Usual Suspects, The (1995) Previsão de avaliação 8.5 para Schindler's List (1993) Previsão de avaliação 8.4 para Raiders of the Lost Ark (1981) Previsão de avaliação 8.4 para Empire Strikes Back, The (1980) Previsão de avaliação 8.4 para Braveheart (1995) Avaliações originais fornecidas: Avaliou 4 para Toy Story (1995) Avaliou 3 para Twelve Monkeys (1995) Avaliou 5 para Usual Suspects, The (1995) Avaliou 4 para Outbreak (1995) Avaliou 5 para Shawshank Redemption, The (1994) Avaliou 3 para While You Were Sleeping (1995) Avaliou 5 para Forrest Gump (1994) Avaliou 2 para Silence of the Lambs, The (1991) Avaliou 4 para Alien (1979) Avaliou 5 para Die Hard 2 (1990) Avaliou 5 para Sphere (1998)

Tabela 1: Exemplo de saída produzida pelo script de geração de recomendações.

um dados usuário e exibir os filmes recomendados. A saída desse script deve ser parecida com a apresentada na Tabela 2.3, de acordo com as classificações definidas anteriormente. Observe que você pode obter um conjunto diferente de previsões diferentes devido a inicializações aleatórias.

3 Aprendizado de Comitês

Nesta seção, você irá tomar como ponto de partida o experimento contido no notebook fornecido na aula sobre aprendizado de comitês. Esse notebook treina um comitê de classificadores utilizando o algoritmo AdaBoost, cuja implementação em Python é fornecida. Sua tarefa nessa parte do trabalho é usar a biblioteca scikit-learn para replicar o experimento do notebook, com a diferença que você irá a classe `sklearn.ensemble.AdaBoostClassifier`. Compare os resultados obtidos no notebook e os resultados obtidos com sua implementação. Para isso, apresente em um mesmo gráfico (que plota a taxa de erro contra a quantidade de iterações) as taxas de erro de treinamento e de teste, tanto para a implementação fornecida no notebook, quanto para sua implementação usando a classe `sklearn.ensemble.AdaBoostClassifier`. (de 1 até 400, de 10

em 10).

Agora, usando a classe `sklearn.ensemble.BaggingClassifier`, realize um experimento também similar ao notebook fornecido. Para isso, apresente em um mesmo gráfico (que plota a taxa de erro contra a quantidade de iterações) as taxas de erro de treinamento e de teste obtidas com sua implementação usando a classe `sklearn.ensemble.BaggingClassifier`. Use a mesma faixa de variação de componentes utilizada no notebook (de 1 até 400, de 10 em 10).

4 O que deve ser entregue

Você deve preparar um único relatório para a apresentar sua análise e conclusões sobre as diversas partes desse trabalho. O formato desse relatório deve ser em PDF. Alternativamente à entrega do relatório em PDF, você pode entregar um notebook Jupyter[2].

Independente de escolher entregar um relatório em PDF ou na forma de um notebook Jupyter, entregue também todos os arquivos em Python que você criou para cada parte deste trabalho. Todos os arquivos em Python devem estar em uma única pasta.

Crie um arquivo compactado que contém o relatório (ou notebook Jupyter) e os arquivos (*scripts*) em Python. Esse arquivo compactado deve se chamar `SEU_NOME_COMPLETO_T3.zip`. Esse arquivo compactado deve ser entregue pelo Moodle, até a data acordada.

5 Créditos

Esse trabalho é uma tradução/adaptação dos *programming assignments* encontrados no curso *Machine Learning*[3] encontrado no Coursera. O material original é de autoria do prof. Andrew Ng.

[2] `http://jupyter.org/`

[3] `https://www.coursera.org/learn/machine-learning`