CEFET/RJ
Programa de Pós-graduação
em Ciência da Computação
Aprendizado de Máquina - Trabalho 02 - parte II

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

Outubro/2017

Conteúdo

1 Introdução

Neste exercício, você implementará o algoritmo de detecção de anomalia e irá aplicá-lo para detectar servidores com falha em uma rede. Na segunda parte, você usará filtragem colaborativa para criar um sistema de recomendação para filmes.

2 Detecção de Anomalias

Nesta parte, você implementará um algoritmo de detecção de anomalia para detectar comportamentos anômalos nos servidores de um centro de dados. As características medem a vazão (throughput) (mb/s) e a latência (ms) de resposta de cada servidor. Enquanto seus servidores estavam funcionando, foram coletados $m = 307$ exemplos de como eles estavam se comportando. Há a suspeita de que a grande maioria desses exemplos são exemplos "normais"(não anômalos) dos servidores que operam normalmente, mas também pode haver alguns exemplos de servidores que atuam de forma anômala nesse conjunto de dados.

Você usará um modelo gaussiano para detectar exemplos anômalos em seu conjunto de dados. Você usará um conjunto de dados 2D que permitirá que você visualize o que o algoritmo está fazendo. Nesse conjunto de dados, você ajustará uma distribuição gaussiana e então encontrará valores que têm probabilidade muito baixa e, portanto, podem ser considerados anomalias.

2.1 Distribuição Gaussiana

Para realizar a detecção de anomalia, você precisará primeiro ajustar um modelo à distribuição dos dados.

Dado um conjunto de treinamento $\{x^{(1)}, \ldots, x^{(m)}\}$ (onde $x^{(i)} \in R^n$), você deve estimar a distribuição gaussiana para cada uma das características x_j. Para cada característica $j = 1, \ldots, n$, você precisa encontrar os parâmetros μ_j e σ_j^2 que se encaixam nos dados na j-ésima dimensão $x(1), ..., x(m)$ (a j-ésima dimensão de cada exemplo).

A distribuição gaussiana é dada por

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

2.2 Estimativa de parâmetros para uma gaussiana

Você pode estimar os parâmetros, (μ_j, σ_j^2), da j-ésima característica usando as equações apresentadas em aula.

Sua tarefa é implementar código em um arquivo de nome estimativaGaussian.py, que deve conter uma função de mesmo nome. Esta função toma como entrada a matriz de dados X e deve produzir um vetor de dimensão n mu que contém a média de todas as características n e outro vetor de dimensão n sigma2 que contém as variâncias de todas as características.

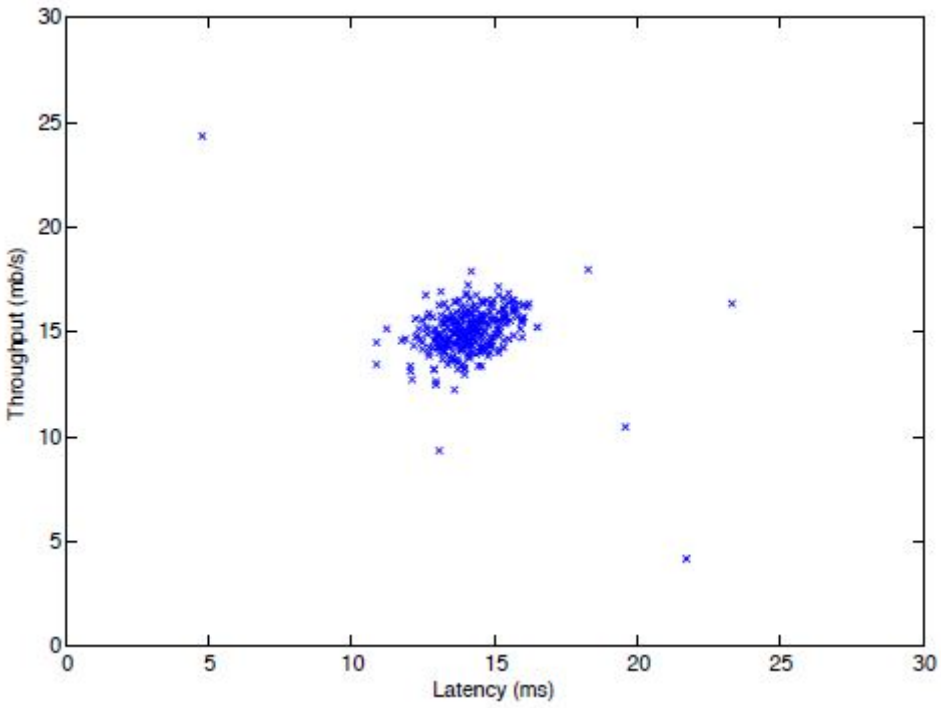

Figura 1: Conjunto de dados - servidores em um *data center*.

Uma vez que você tenha completado o código em estimativaGaussian.py, sua próxima tarefa será visualizar os contornos da distribuição gaussiana ajustada. Você deve obter um gráfico semelhante à Figura 1. Dessa figura, você pode ver que a maioria dos exemplos está na região com maior probabilidade, enquanto os exemplos anômalos estão nas regiões com probabilidades menores.

2.3 Selecionando ϵ

Agora que você avaliou os parâmetros gaussianos, você pode investigar quais exemplos têm uma probabilidade muito alta dada essa distribuição e quais exemplos têm uma probabilidade muito baixa. Os exemplos de baixa probabilidade são mais prováveis de ser as anomalias em nosso conjunto de dados. Uma maneira de determinar quais exemplos são anomalias é selecionar um limite ϵ com base em um conjunto de validação cruzada. Nesta parte do exercício, você implementará um algoritmo para selecionar o limite ϵ usando a medida F_1 em um conjunto de validação cruzada. Você deve implementar código em um arquivo de nome selectThreshold.py. Para isso, use um conjunto de validação cruzada $\{(x_{cv}^{(1)}, y_{cv}^{(1)}), \ldots, x_{cv}^{(m)}, y_{cv}^{(m)})\}$, onde o rótulo $y = 1$ corresponde a uma exemplo anômalo. Para cada exemplo no conjunto de validação, você deve computar $\Pr(x_{cv}^{(i)})$. O vetor de todas essas probabilidades $\Pr(x_{cv}^{(1)}), \ldots, \Pr(x_{cv}^{(m_{cv})})$, assim como os rótulos correspondentes, devem ser passados a função implementada em selectThreshold.py.

A função selectThreshold.py deve retornar dois valores; o primeiro é o limite selecionado ϵ. Se um exemplo x tiver uma baixa probabilidade $\Pr(x) < \epsilon$, então

é considerado uma anomalia. A função também deve retornar a pontuação F_1, o que indica o quão bem você está realizando a tarefa de encontrar as anomalias verdadeiras, dado um determinado limite. Para muitos valores diferentes de ϵ, você irá calcular o resultado F_1 resultante ao calcular quantos exemplos o limite atual classifica corretamente e incorretamente.

O escore F_1 é calculado com precisão ($prec$) e revocação (rec):

$$F_1 = \frac{2 \times prec \times rec}{prec + rec}$$

Você calcula a precisão e o revocação por:

$$prec = \frac{tp}{tp + fp}$$

$$rec = \frac{tp}{tp + fn}$$

onde

- tp é o número de verdadeiros positivos: o rótulo de verdade do solo diz que é uma anomalia e nosso algoritmo classificou-o corretamente como uma anomalia.
- fp é o número de falsos positivos: o rótulo de verdade do solo diz que não é uma anomalia, mas nosso algoritmo classificou-o incorretamente como uma anomalia.
- fn é o número de falsos negativos: o rótulo de verdade do solo diz que é uma anomalia, mas nosso algoritmo incorretamente classificou-o como não sendo anômalo.

Em selectThreshold.py, adicione um loop para testar muitos valores diferentes de ϵ e selecionar o melhor ϵ com base no resultado F_1.

Você pode implementar o cálculo do escore F_1 usando um loop for para todos os exemplos de validação cruzada (para calcular os valores tp, fp, fn). Você deve ver um valor para epsilon de cerca de 8.99e-05.

3 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. O arquivo que contém o conjunto de dados tem nome ex8_movies.mat, e contém as variáveis Y e R.

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

3.1 Conjunto de dados de classificações de filme

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 estará trabalhando 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 $n_u \times 100$.

3.2 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 cofiCostFunc.py para calcular a função de custo e o gradiente para a filtragem colaborativa. Os parâmetros para a função (ou seja, os valores que você está tentando aprender) devem ser X e $Theta$.

3.2.1 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 cofiCostFunc.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.

3.2.2 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.

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_T2.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`