CEFET/RJ
Programa de Pós-graduação
em Ciência da Computação
Aprendizado de Máquina - Trabalho 01

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

Setembro/2017

Conteúdo

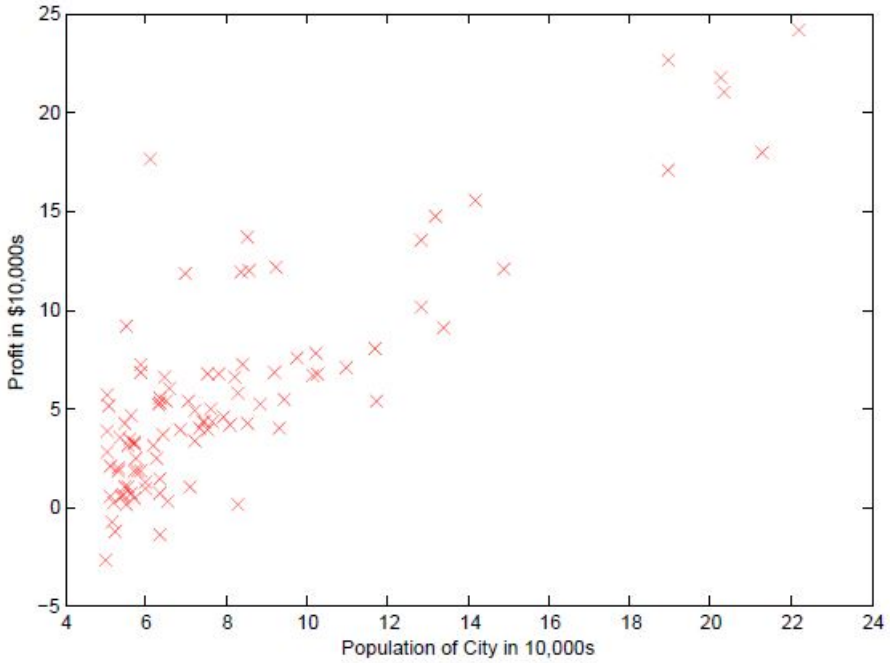

Figura 1: Pontos de dados do conjunto `ex1data1.txt`.

1 Regressão Linear com uma Variável

Nessa parte do trabalho, você irá implementar a regressão linear para predizer o lucro para uma cadeia de *food truck*. Essa cadeia já possui diversos filiais em diferentes cidades. Você possui dados dos lucro e população para cada uma dessas cidades.

O arquivo `ex1data1.txt` contém os dados a serem usados nessa parte do trabalho. A primeira coluna corresponde à população de cada cidade, enquanto que a segunda coluna corresponde ao lucro da filial daquela cidade.Um valor negativo para o lucro indica que a filial correspondente está dando prejuízo.

1.1 Visualização dos Dados

Para a maioria dos conjunto de dados do mundo real, não é possível criar um gráfico para visualizar seus pontos. Mas, para o conjunto de dados fornecido nesse trabalho, isso é possível. Implemente um script em Python que produza um *gráfico de dispersão* (*scatter plot*) dos dados fornecidos. Após finalizado, seu script deve produzir um resultado similar ao apresentado na Figura 1.1.

1.2 Gradiente Descendente

Nessa parte, sua tarefa é determinar os parâmetros do modelo de regressão linear por meio do algoritmo Gradiente Descendente. Use a versão *batch gradient descendente* desse algoritmo. Inicie os parâmetros todos com o valor 0 (zero). Além disso, use o valor 0,01 para a taxa de aprendizado. Crie uma função em

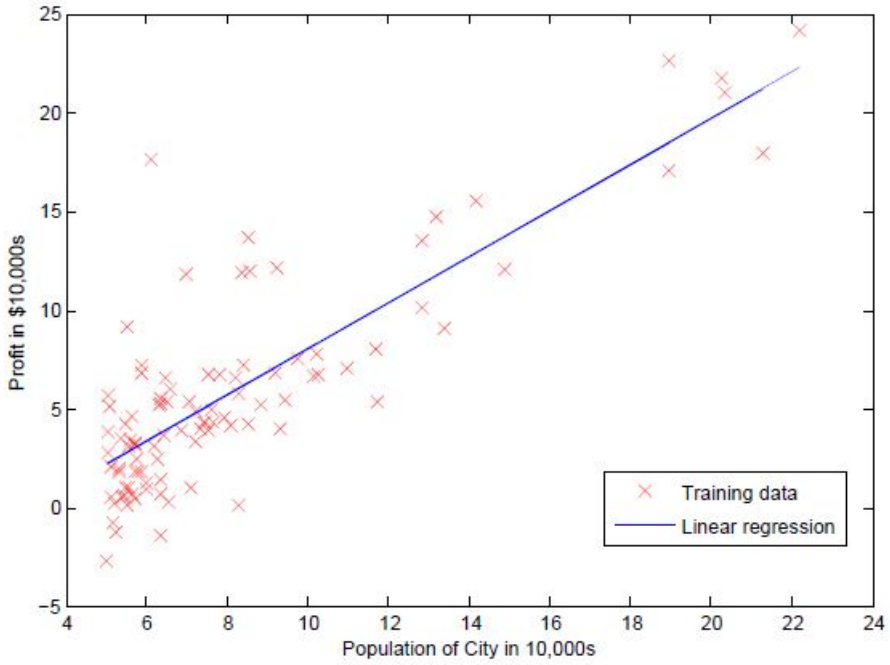

Figura 2: Pontos de dados do conjunto `ex1data1.txt`.

Python denominada `computarCusto`. Essa função deve ser definida no arquivo `computarCusto.py`. Após implementar essa função, você pode verificar a corretude executando com todos os parâmetros iguais a zero. Nessa situação, sua função deve gera um valor igual a 32,07.

Após implementar o cálculo da função de custo $J(\theta)$, você deve implementar o GD propriamente dito. Sua implementação do algoritmo gradiente descendente deve ser feita em um arquivo de nome `gduni.py`. Essa implementação deve chamar a função `computarCusto` de forma apropriada.

Presumindo que você implementou o GD e o cálculo da função de custo corretamente, os valores sucessivos do gradiente nunca devem crescer. Além disso, o valor de $J(\theta)$ deve convergir no fim da execução do algoritmo.

Após implementar o GD, você devem implementar um script em Python para visualizar a reta correspondente aos parâmetros determinados pela execução de sua implementação. O resultado desse script deve ser similar ao apresentado na Figura 1.2.

A última tarefa nessa parte do trabalho, você deve usar o modelo de regressão linear produzido pelo seu código para predizer o lucro em regiões com populações de 35.000 e 70.000 habitantes. Forneça no seu relatório o código (em Python) para isso, assim como os valores correspondentes do lucro para cada uma daquelas duas cidades.

1.3 Visualização de $J(\theta)$

Para melhor entender a função de custo, você irá nessa parte do trabalho plotar o custo sobre uma grade bidimensional de valores de θ_0 e de θ_1. Para isso, você

deve usar sua implementação da função `computarCusto`.

O código que você deve implementar deve gerar um array bidimensional de valores de $J(\theta)$. Os valores gerados pelo seu código devem estar na faixa a seguir: $-10 \leq \theta_0 \leq +10$ e $-1 \leq \theta_0 \leq +4$. Utilize incremento de 0,01 para gerar os valores de θ_0 e de θ_1.

A seguir, usando a função `matplotlib.pyplot.contour` da biblioteca matplotlib[1], produza um gráfico de curvas de contorno (*contour plot*). Também utilizando a biblioteca matplotlib, crie um gráfico da superfície correspondente a $J(\theta)$.

2 Regressão Linear com Múltiplas Variáveis

Nessa parte do trabalho, você irá implementar a regressão linear com múltiplas variáveis para predizer o preço de venda de imóveis. O arquivo `ex1data2.txt` contém informações acerca de preços de imóveis. A primeira coluna corersponde ao tamanho do imóvel (em pés quadrados[2]). A segunda coluna corresponde à quantidade de dormitórios no imóvel em questão. A terceira coluna corresponde ao preço do imóvel.

2.1 Normalização das características

Se você inspecionar os valores do conjunto de dados fornecido, irá notar que os tamanhos dos imóveis são aproximadamente 1000 vezes maiores que as quantidades encontradas na coluna de quantidade de dormitórios. Sua tarefa nessa parte é implementar uma função denominada `normalizarCaracteristica` em um arquivo denominado `normalizarCaracteristica.py`. Essa função deve:

- subtrair o valor médio de todas as características do conjunto de dados.
- após subtrair a média, dividir cada característica pelos seus respectivos desvios padrões.

A sua função `normalizarCaracteristica` deve a matriz de dados X de dados como parâmetro (na forma de um *numpy array*). Além disso, essa função deve funcionar com conjuntos de dados de variados tamanhos (qualquer quantidade de características / exemplos). Repare que cada coluna da matriz de dados X passada para a função `normalizarCaracteristica` deve corresponder a um característica.

Nota de Implementação: Ao normalizar as características, é importante armazenar os valores utilizados para a normalização - o valor médio e o desvio padrão utilizados para a normalização. Depois de aprender os parâmetros do modelo, muitas vezes queremos prever os preços das casas que não temos visto antes. Dado um novo valor x (área da sala de estar e número de quartos), devemos normalizar x usando a média e o desvio padrão que nós previamente calculamos a partir do conjunto de treinamento.

[1] (`https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.contour.html`)

[2] 1 pé corresponde a 0,3048 metros.

2.2 Gradiente descendente

Anteriormente, você implementou o GD em uma regressão linear univariada. A única diferença agora é que há mais uma característica na matriz de dados X. A função de hipótese $h_\theta(x)$ e a atualização dos gradientes em lote permanecem inalteradas. Você deve implementar código nos arquivos denominados `computarCustoMulti.py` e `gdmulti.py` para implementar a função de custo e o algoritmo GD para regressão linear com múltiplas variáveis, respectivamente. Se o seu código na parte anterior (variável única) já provê suporte a múltiplas variáveis, você também pode reusá-lo aqui.

Se assegure de que o seu código dá suporte a qualquer número de características e está bem vetorizado.

3 Regressão Logística

Nessa parte do trabalho, você irá implementar a regressão logística. Em particular, você irá criar uma classificador para predizer se um estudante será admitido em uma universidade, com base nos resultados de duas avaliações. Suponha que estão disponíveis dados históricos acerca de realizações passadas dessas avaliações, e que esses dados históricos podem ser usados como conjunto de treinamento. Para cada exemplo desse conjunto de treinamento, temos as notas das duas avaliações e a decisão acerca do candidato (aprovado ou reprovado).

Sua tarefa é construir um modelo de classificação que provê uma estimativa da probabilidade de admissão de um candidato, com base na notas que ele obteve nas duas avaliações.

O arquivo `ex2data1.txt` contém os dados a serem usados nessa parte do trabalho.

3.1 Visualização dos dados

Antes de começar a implementar qualquer algoritmo de aprendizado, é adequado visualizar os dados, quando possível. Nessa parte do trabalho, você deve carregar o arquivo com o conjunto de treinamento e plotar (i.e., produzir um gráfico) os pontos de dados. O resultado dessa tarefa deve ser um gráfico similar ao apresentado na Figura 3.1.

3.2 Implementação

3.2.1 Função sigmoide

Como primeiro passo nessa parte, implemente a função em Python que calcula o valor da função sigmoide. Defina essa função em um arquivo denominado `sigmoide.py`, de tal forma que ela possa ser chamada de outras parte do seu código. Após finalizar sua implementação, você pode verificar sua corretude. Se você chamar `sigmoide(0)`, o valor retornado deve ser 0,5. Se você chamar essa função com muito grandes e positivos (muito grandes negativos), ela deve

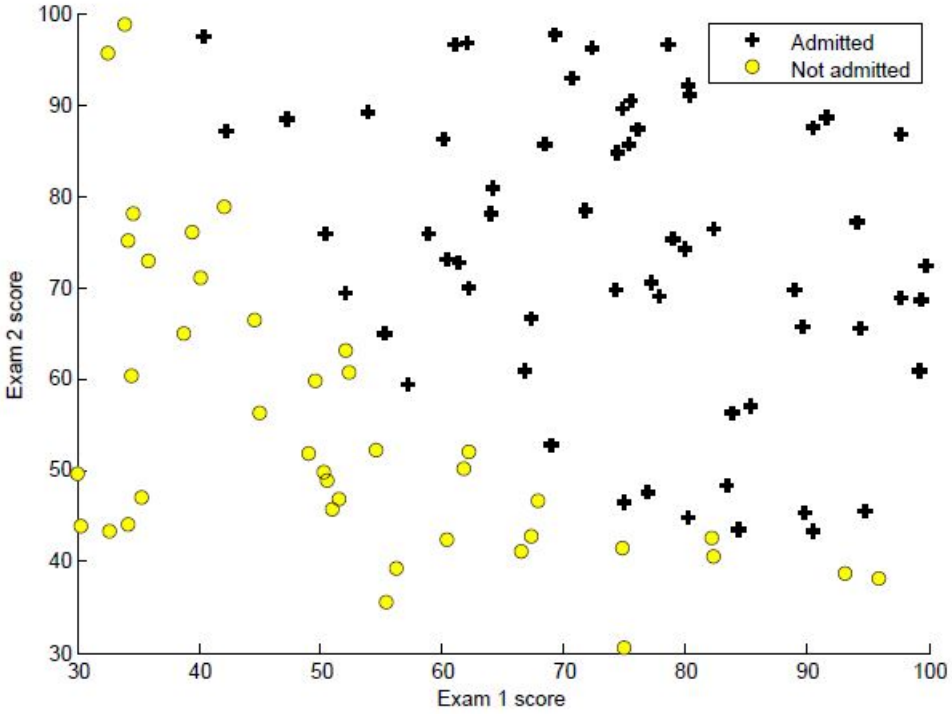

Figura 3: Pontos de dados do conjunto `ex2data1.txt`.

retornar um valor muito próximo de 1 (0). O seu código também deve funcionar com vetores (i.e., o seu código deve estar vetorizado). Em particular, se uma matriz for passada, o seu código deve aplicar a função sigmoide a cada componente.

3.2.2 Função de custo e gradiente

Agora, você deverá implementar a função de custo para a regressão logística. Essa função deve retornar o valor de função de custo e o gradiente. Implemente esse código em um arquivo denominado `funcaoCustoRegressaoLogistica.py`. Lembre-se de que o gradiente é um vetor com o mesmo número de elementos que θ.

Uma vez que tenha implementado essa função, realize uma chamada usando o valor inicial de θ. Você deve confirmar que o valor produzido é aproximadamente 0,693.

3.2.3 Aprendizado dos parâmetros

Para a regressão logística, o objetivo é minimizar $J(\theta)$) com relação ao vetor de parâmetros θ. Sendo assim, nessa parte você deve implementar uma função em Python para encontrar o vetor θ que minimiza a função de custo. Utilize a função `funcaoCustoRegressaoLogistica` que você implementou previamente.

3.2.4 Avaliação do modelo

Após o aprendizado dos parâmetros, você pode usar o modelo correspondente para predizer se um candidato qualquer será aprovado. Para um candidato com notas 45 e 85 na primeira e segunda avaliações, respectivamente, você deve esperar que ele seja aprovado com probabilidade 0,776.

Outro modo de avaliar a qualidade dos parâmetros é verificar o quão bem o modelo aprendido prediz os pontos de dados do conjunto de treinamento. Nessa parte, você deve implementar uma função denominada `predizer`. Essa função deve produzir os valores 0 ou 1, dados um exemplo do conjunto de treinamento o vetor de parâmetros θ. Use essa função para produzir a porcentagem de acertos do seu classificador sobre o conjunto de treinamento.

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[3].

Independente de escolher entregar um relatório em PDF ou na forma de um notebook Jupyer, entregue também todos os arquivos em Python que você criou para cada parte deste trabalho. Todos os arquivos em Python deve estar na mesma 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_T1.zip`. Esse arquivo compactado deve ser entregue pelo Moodle, até a data acordada.

5 Créditos

Esse trabalho é uma tradução/adaptação do *programming assignment* 1 do curso *Machine Learning*[4] encontrado no Coursera. O material original é de autoria do prof. Andrew Ng.

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

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