Predição de Valores com Reg. LIN

Exercício Aula 1

Laboratório de Álgebra Linear Numérica com Python e Octave

IBTA - Instituto Brasileiro de Tecnologia Avançada
Prof. Moacyr C. Mello F.
Abril de 2016

PROPOSIÇÃO DO MODELO

Uma grande rede de cinemas, especializada em projetos complexos de promoção de filmes, criou um projeto especial que vai combinar rede social com salas de cinema. A Rede pretende expandir suas salas de projeção em pequenas cidades do país com mais de 50 000 habitantes mas, ao mesmo tempo, fará essa expansão através de franquias, usando fidelização e um sistema de recomendação de filmes para seus usuários.

O critério de escolha das cidades para expansão leva em conta as estatísticas populacionais e os hábitos de consumo locais. O critério de escolha dos títulos para exibição leva em consideração filmes exibidos pela Rede durante o ano, em localidades próximas, envolvendo os mais diversos gêneros. Posteriormente esse critério será influenciado pela preferência indicada pela fidelização. O critério de recomendação de filmes leva em conta a preferência do usuário desde que ele já tenha se relacionado com a Rede e assistido a filmes em alguma ocasião. A iniciativa é um projeto que utiliza rede social para aumentar a participação da população nas bilheterias da Rede e, ao mesmo tempo, reaproximar o cinema do grande público.

A primeira providência foi encomendar uma análise custo/benefício para uma consultoria especializada em segmentação de mercado demográfico, que possui estatísticas relacionando o lucro de salas de cinema em função da população. Essa consultoria fornecerá os dados que utilizaremos para nossa análise. Nossa tarefa consiste em elaborar um modelo algoritmico de aprendizado supervisionado, treinado para fazer predições de lucro e custo, a partir dos dados da consultoria.

A primeira aplicação do novo algoritmo é encontrar as cidades mais lucrativas baseando-se nos dados da população. Utilizaremos um conjunto de dados já pré-processados pela consultoria, que deriva a relação entre população e lucro (bruto) baseando-se em estatísticas demográficas de outros serviços. A tabela seguinte é uma amostra desse conjunto de dados que está em **data1.octave**.

Cidade candidata	População [10 000]	Lucro [R$ 1000]
A	6.1101	17.5920
B	5.5277	9.1302
C	8.5186	13.6620
D	7.0032	11.8540
E	5.8598	6.8233

moacyrcmello@gmail.com

Nosso objetivo é construir uma função linear do tipo $y = ax+b$ que seja a melhor aproximação para o conjunto de dados exemplo, e que possa ser utilizada como função preditora para as demais cidades. Trata-se de treinar um modelo através de **aprendizado supervisionado**, que é o processo em que fornecemos os exemplos A, B, C, … etc e construímos a função calculando seus coeficientes.

A segunda aplicação é analisarmos a viabilidade do investimento, para isso calculamos o custo do aluguel local em função de outras variáveis como tamanho e número de salas. Também para este cálculo vamos nos basear em dados pré-processados pela consultoria, que relacionam diversos exemplos das combinações dessas variáveis. Estes dados estão na tabela seguinte e são uma amostra dos dados que estão em **data2.octave**.

Cidade candidata	Tamanho [m2]	Número de salas	Custo [R$]
A	2104	3	399900
B	1600	3	329900
C	2400	3	369000
D	1416	2	232000
E	3000	4	539900

Nosso objetivo é construir uma função do tipo $y = ax_1+bx_2+c$ que aproxime o conjunto de dados exemplo; e que possa ser usada como uma função preditora para as demais cidades. Também aqui, trata-se de treinar um modelo através de aprendizado supervisionado. Fornecemos exemplos A, B, C, … etc e chegamos à função calculando seus coeficientes. Porém, neste caso, a regressão linear não será de uma só variável mas **multivariada**.

Modelo de Hipótese

Hipótese é o nome que se dá à representação do modelo numa forma particular de função. No caso da regressão linear a hipótese é um modelo linear do tipo $y = ax+b$ que, na notação de aprendizagem de máquina costuma ser escrita como $h_\theta(x) = \theta_0 + \theta_1 x$. O nome **hipótese**, e a representação **$h_\theta(x)$**, ressaltam o fato de que o modelo proposto precisa ser comprovado e é diferente de **y**, o resultado real dos exemplos, muitas vezes referido como variável target.

> $h_\theta(x) = \theta_0 + \theta_1 x$ é a hipótese do modelo de regressão linear de uma variável (univariada).
>
> θ_0, θ_1 são parâmetros do modelo; devemos escolhê-los de modo a que $h_\theta(x)$ fique o mais próximo possível de y em todos os exemplos de treinamento (x, y).

Portanto, para criar a hipótese do modelo, o que fazemos é escolher θ_0, θ_1 de modo a minimizar a diferença entre $h_\theta(x)$ e y. A forma de se fazer a escolha é montar uma função e minimizá-la, essa função é denominada função de custo.

Função de Custo

A **função de custo** é uma função que descreve a relação entre $h_\theta(x)$ e y, isto é, a saída, ou resposta do modelo e os valores reais vindos dos exemplos do treinamento. É uma função de erro ou desvio, é denominada custo porque avalia o quanto nosso modelo está longe dos exemplos, há várias

formas possíveis, em geral a mais usada é o erro quadrático médio.

Erro quadrático significa que a **diferença no eixo y**, entre o valor dos exemplos e o valor predito pela hipótese, é elevada ao quadrado para eliminar os valores positivos e negativos que se distribuem em torno do valor projetado da hipótese. Além disso, uma função quadrática permite minimização ou maximização com facilidade, igualando ou fazendo tender a derivada à zero. Para aplicá-lo tomamos a média de todos os exemplos escrevendo-os em função dos parâmetros da hipótese. Nosso objetivo é minimizar esta função para escolhermos os parâmetros θ_0, θ_1 . A constante ½ que aparece na formulação tem função simplificadora para o passo seguinte de derivação, e não influencia o papel de minimização do erro quadrático médio.

$J(\theta_0, \theta_1) = 1/2m \sum_{i=1} (h_\theta(x^{(i)}) - y^{(i)})^2$ é a função de custo para **m** exemplos de treinamento.

$\min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$ é o objetivo da regressão, ajusta a melhor reta possível aos exemplos (x, y).

Gradient Descent

Gradiente Descendente é um algoritmo usado em problemas de otimização, isto é, problemas em que buscamos maximizar ou minimizar uma função. É um algoritmo iterativo e seu objetivo é minimizar a função de custo (ou a função utilidade quando usado para maximização, neste caso denomina-se *gradient ascent*).

Para o nosso exercício vamos usá-lo para encontrar o mínimo local da função de custo, que ajusta a reta aos exemplos e assim encontrar os parâmetros da função hipótese. Partimos de $\theta_0, \theta_1 = (0,0)$, ou de uma outra semente, e a cada passo tomamos a direção negativa do **gradiente**, que corresponde **à direção de declive máximo** para baixo. Podemos imaginar como sendo o caminho seguido pelo curso da água na sua descida sob força da gravidade. Este método também é chamado método do máximo declive, porque é justamente o que procura: caminhar pelo máximo declive.

Repetimos até convergir:

$$\theta_0 = \theta_0 - \alpha \cdot \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

Atualizando simultaneamente !

$$\theta_1 = \theta_1 - \alpha \cdot \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

Onde **α** é o passo ou taxa de aprendizado, também chamado parâmetro de convergência. Escolhemos α empiricamente entre os valores: … 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1.

Por ser iterativo é preciso verificar se está convergindo para uma resposta. Se α é muito pequeno a convergência é demorada e precisa de muitas iterações. Se α é grande, $J(\theta_0, \theta_1)$ pode não decrescer a cada iteração, ficando oscilando ou mesmo divergindo. Portanto a escolha de α é **crítica** para a convergência e a prática mostrou que devemos experimentar valores múltiplos de três.

moacyrcmello@gmail.com

Podemos testar a convergência através das **Curvas de Convergência** que veremos em aula, elas mostram o custo em função do numero de iterações para cada α escolhido. Quando acertamos na escolha de α vamos observar uma curva decrescente que se estabiliza num valor de custo aceitável, a cada passo o processo iterativo se aproxima dos valores ótimos de θ_0 e θ_1 que correspondem ao mínimo custo.

Introduzindo, derivando e rearranjando $J(\theta_0, \theta_1)$ nas equações do gradiente escrevemos:

$$\theta_0 = \theta_0 - \alpha \cdot 1/m \sum (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

Atualizando simultaneamente !

$$\theta_1 = \theta_1 - \alpha \cdot 1/m \sum (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

Observe que, por convenção, $x_0 = 1$.

Normalização

Para muitos problemas de aprendizagem de máquina precisaremos preparar os dados de entrada para melhorar o desempenho do algoritmo. Uma forma de fazer isso é **normalizar** os dados através de uma transformação de variáveis. Em geral normalizamos pela média fazendo a redução de todos os valores referidos à média e desvio padrão dos dados de entrada. No caso do primeiro exercício não usaremos normalização, mas no segundo será necessária. Vamos normalizar aplicando a seguinte operação:

$$x_i = (x_i - \mu_i)/ \sigma_i$$

Onde μ_i é a média dos atributos x_i e σ_i é o desvio padrão.

EXERCÍCIOS DE PROGRAMAÇÃO

Exercício 1

1. Prototipação da solução
Inicie pela prototipação no ambiente Octave(MatLab) e escreva a fórmula da função de custo. Para armazená-la você pode utilizar um arquivo de script onde deixará a função. Em seguida aborde o problema de escrever o algoritmo do gradiente descendente e o armazene, também como uma função num arquivo script. Perceba a relação que ambas as funções tem entre si. Pense no fluxo geral do seu programa, você precisará entrar com dados, processá-los e verificar os resultados. Chamamos a isso **pipeline**, é um "tubo" que executa o procedimento principal que você está elaborando. Usualmente criamos gráficos e testes para verificar a correção do processo, portanto será preciso algum "design" nessa elaboração de sequência, porém quanto mais simples melhor. Experimente as vantagens do ambiente interativo. O procedimento geral é o seguinte:

1. Carga dos dados;
2. Exame dos dados;
3. Design do pipeline;
4. Implementação dos algoritmos em funções;
5. Diagnostico através de curvas e gráficos.

2. Estrutura de dados
Verifique se a sua notação de vetores, matrizes, e dados em geral, está compatível com a descrição

fornecida. Carregue as variáveis, matriz **X** e **y**, a partir da base **data1.octave**.
X é a matriz de população das cidades e **y** é a matriz do correspondente lucro presumido.

```
load data1.octave
[m, n] = size(X)
```

3. Ambiente interativo Octave
Experimente o ambiente com a base de cidades fornecida, veja que é a mesma matriz mostrada na exposição acima neste documento. Verifique as facilidades para criação de matrizes e execução de operações, examine a multiplicação, a multiplicação elemento-por-elemento, transposição e inversão. Consulte a bibliografia de referência do Octave. Comandos úteis:

```
whos
ls -l
pwd
zeros(m,n)
rand(m,n)
```

4. Salve permanentemente suas funções em scripts Octave
Utilize o template de script para Octave e transcreva a sua função de custo e a função de gradient descent no formato de funções Octave, uma única função por arquivo. Salve utilizando o próprio nome da função. (Veja manual Octave sobre o formato de funções e suas descrições). Transcreva o pipeline para um script.

5. Refinando iterativamente
Examine a capacidade de predição do seu algoritmo através de um exemplar externo ao conjunto de dados de treinamento. Utilize para isso a cidade de Andradina (SP) com a quantidade de 55.317 habitantes (na época), você deverá encontrar um valor aproximado de R$28.000,00 por período. (Período é uma "unidade de período", semestral, anual, etc. Mas você não precisa se preocupar com isso pois esse valor é resultado de estatísticas e a finalidade é o cálculo do break-even por período).

A solução deve atender, no mínimo, a seguinte interface de entrada e saída:
Entrada: X e y (*tabelados*), iter (*número de iterações*), α (*passo de iteração*), θ (*semente*)
Saída: θ (*parâmetros ou coeficientes da hipótese*), lucro (*predito pelo algoritmo*)

```
Você deve entregar os scripts Octave, bem como todo arquivo que seja necessário
para orientar a execução, ou explicar a solução. Os scripts serão executados
pelo professor com a matriz fornecida no exemplo.
```

6. Transcrição em PseudoPython
Quando a solução estiver funcionando em protótipo você deve descrevê-la num formato neutro e genérico que denominamos PseudoPython; há um template de PseudoPython para ajudá-lo nisso. Este é o momento de documentar a solução, simplificá-la e torná-la mais geral. Elimine qualquer referência implementacional ao Octave.

A grande vantagem de descrever seu procedimento é obter uma descrição de nível mais alto, mais abstrata, de modo a se distanciar da linguagem Octave. Qualquer pessoa que ler seu pseudocódigo deve entender o procedimento, independentemente de particularidades da linguagem. Essa descrição irá ajudar a implementação em Python e, principalmente permitirá um design mais limpo. Uma outra vantagem é que também servirá como documentação consultiva.

```
Entregar o PseudoPython, que será verificado em referência às orientações do template
```

7. Implementação opcional em Python
Nesta altura já possuimos bastante clareza do problema e temos uma solução prototipada que funciona. Então, opcionalmente neste caso, podemos traduzir a solução para uma implementação limpa e profissional numa linguagem de propósito geral.

Para a implementação em Python a orientação é seguir o pseudocódigo elaborado no passo anterior. Se este estiver corretamente escrito o procedimento deve transparecer claramente; livre da linguagem de programação original. Recomendamos consultar a bibliografia onde há tabelas de correspondência entre operadores Octave e Python. Lembre-se que as aparências, frequêntemente, escondem diferenças sutis.

8. Ambiente integrado Python
Caso tenha optado por implementar em Python você estará num novo ambiente, com uma nova linguagem. Há diferenças em detalhes, como índices que começam de zero, e a utilização da biblioteca numérica Numpy, que precisaremos importar explicitamente. Mas a vantagem da linguagem é permitir implementar um sistema profissional completo. O Python fornece um IDE default muito simples, denominado IDLE, que é um ambiente integrado com shell e editor. Ele nos permite avançar para a implementação sem nos preocupar em aprender um novo ambiente sofisticado e sem abandonar a possibilidade de prototipação rápida. Há inúmeros ambientes Python mais avançados, caso o aluno esteja familiarizado com outro ambiente poderá usá-lo.

Carregue a matriz da base de dados **data1.X.python** na variável **X,** que é do tipo *matrix definido pelo Numpy*, e **data1.y.python** na variável **y**.

```
import pickle
X = pickle.load(open('data1.X.python','rb'))
y = pickle.load(open('data1.y.python','rb'))
m,n = numpy.shape(X)
```

Importante, a estrutura de dados que sugerimos para representar as matrizes em Numpy é *matrix*. Poderia ter sido *array* ou *ndarray* que são estruturas diferentes de *matrix* porém mais comuns na cultura Python. O motivo é a semelhança desse tipo de dados com a álgebra linear, no entanto algumas funções retornarão *array*, basta converter. Verifique, experimentando no shell, as diferenças e também examine a bibliografia.

```
Se você optou por transcrever a solução em Python deve entregar os scripts para execução.
      Os scripts serão executados pelo professor com a matriz fornecida no exemplo.
```

Exercício 2

1. Completando a solução
No exercício 1 obtivemos o lucro presumido. Para completar a análise de custo/benefício vamos agora calcular o custo do aluguel local em função de opções como tamanho da sala de projeção e número de salas. Os dados foram pré-processados pela consultoria que tabelou diversos casos reais das combinações dessas variáveis. Uma amostra destes dados estão na segunda tabela da exposição inicial e percebemos que o custo é função de duas variáveis. A tabela completa se encontra em ***data2.octave***. O procedimento geral continua o mesmo, a única diferença é que, devido à natureza discrepante dos dados, é comum normalizar para facilitar a computação numérica por parte do algoritmo.

1. Carga dos dados;
2. Exame dos dados;
3. Normalização;
4. Design do pipeline;
5. Implementação dos algoritmos em funções;
6. Diagnostico através de curvas e gráficos.

2. Estrutura de dados
Verifique se a sua notação de vetores, matrizes, e dados em geral, está compatível com a descrição fornecida. Carregue as variáveis, matriz **X** e **y**, a partir da base **data2.octave**.
X é a matriz de salas para aluguel nas cidades e **y** é a matriz do correspondente custo presumido.

```
load data2.octave
[m, n] = size(X)
```

3. *Normalização dos dados de entrada*
Neste caso precisamos aplicar a normalização nos dados da matriz **X** e matriz **y**. Para tanto escrevemos uma nova função que recebe uma matriz como argumento e aplica o seguinte procedimento:

1. Encontre a média e o desvio padrão de cada coluna da matriz;
2. Recalcule a coluna subtraindo a média em cada elemento;
3. Recalcule a coluna dividindo cada elemento pelo desvio padrão;

4. Regressão multivariada
Utilize o template de script para Octave e transcreva as funções de custo e gradient descent do exercício anterior, planejadas para regressão univariada, para admitir múltiplos atributos, regressão multivariada. Salve o arquivo utilizando o próprio nome da função. (Veja manual Octave sobre o formato de funções e suas descrições). Crie o script do pipeline.

5. Análise Custo/ Benefício
Examine a predição do seu novo algoritmo através de um exemplar externo ao conjunto de dados de treinamento. Utilize para isso a cidade de Andradina (SP) com 1800 m² e 3 salas para aluguel, você deverá encontrar um custo aproximado de R$314.000,00 por período. (A “unidade de período” não é importante aqui, por exemplo, fazendo Custo/Benefício o período é cortado).

A solução deve atender, no mínimo, a seguinte interface de entrada e saída:
Entrada: X e y (*tabelados*), iter (*número de iterações*), α (*passo de iteração*), θ (*semente*)
Saída: θ (*parâmetros ou coeficientes da hipótese*), custo (*predito pelo algoritmo*)

Você deve entregar os scripts Octave, bem como todo arquivo que seja necessário para orientar a execução, ou explicar a solução. Os scripts serão executados pelo professor com a matriz fornecida no exemplo.

6. Transcrição em PseudoPython
Transcrever a solução para PseudoPython como já visto, lembre-se que há um template para ajudá-lo nisso.

7. Implementação em Python
Com o problema resolvido e o procedimento bem definido, opcionalmente, traduzir a solução para uma implementação limpa e profissional numa linguagem de propósito geral: Python.

/* Regressão linear e predição de valores. */

moacyrcmello@gmail.com