Aprendizagem de Máquina e Mineração de Dados

Trabalho 2

Data de Entrega: 26/09/2018
Email de entrega: jonatas.aquino@uece.br

Obs: Qualquer dúvida sobre o trabalho, entrar em contato por e-mail ou ao final da aula.

Revisão Regressão Polinomial no espaço de atributos

No laboratório feito em sala, utilizamos uma base de dados artificial "seno.txt". Então realizamos a regressão linear. Em seguida realize a regressão polinomial no espaço dos atributos, para isso adicionamos colunas artificiais na base de dados de modo a incluir colunas para X^2 X^3 até X^7.

Inicialmente a estrutura da base estava no seguinte formato:

```
          [ X11 y1 ]
          [ X12 y2 ]
BASE:     [ X13 y3 ]
           …
          [ X1m ym ]
```

Separamos X e y, e adicionamos uma coluna de 1's.

```
          [ 1 X11]                  [y1]
          [ 1 X12]                  [y2]
X:        [ 1 X13]          y:      [y3]
           …
          [ 1 X1m]                  [ym]
```

Aplicamos os mínimos quadrados para achar os coeficientes da reta de regressão (Polinômio de grau 1); Em seguida adicionamos artificialmente novas colunas na variável X de modo a permitir a regressão polinomial no espaço dos atributos. Note que cada coluna adicionada aumenta o grau do polinômio.

```
           C1 C2     C3         C4
          [ 1 X11   (X11)^2   (X11)^3 ]
          [ 1 X12   (X12)^2   (X12)^3 ]
X:        [ 1 X13   (X13)^2   (X13)^3 ]
           …
          [ 1 X1m   (X1m)^2   (X1m)^3 ]
```

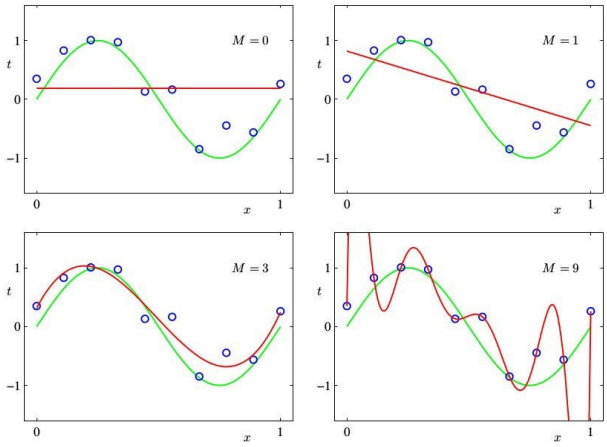


Figura 1: Plotagem regressão de polinômios de ordem 0, ordem 1, ordem 3 e ordem 9.
(Bishop, 2006)

1) Comente o que ocorreu ao se aumentar o grau do polinômio de regressão. Sobre o resultado da plotagem do polinômio de grau 9 (M=9) é um efeito desejado quando se pretende generalizar o aprendizado? (2 Pontos)

Overfitting, quanto maior o grau da curva,em determinado tipo de base, há uma maior chance de overfit, não é desejado,pois o erro sobe substancialmente, passando há prever somente quando o dado está no banco ou é extremamente parcido com o banco….

2) Como podemos detectar e solucionar o problema do overfitting? (2 Pontos)

1→ Maior numero de amostras pode diminuir o overfit
2→ Menor Treinamento pode diminuir o overfiting
3→ Mudar o algoritimo de aprendisagem(no caso mudar o grau)
4→ Ter uma amostra de teste para detecção...

Métrica de Classificação e K-Vizinhos Mais Próximos (K-NN)

Nesta seção será feita a divisão de uma base de dados entre treino e teste para o cálculo de acurácia e a classificação será feita usando o algoritmo K-NN.

Instrução:
Carregue a base de dados "base_artificial.txt".
A base é composta de um 80 exemplos, sendo 40 da classe 0 (negativos), e 40 da classe 1 (positivos).
Divida a base entre treino e teste:

- 25 positivos para treino e 15 negativos para teste
- 25 negativos para treino e 15 positivos para teste
- *Totalizando 50 exemplos para treino e 30 para teste*

Utilize a base de treino como referência para que o K-NN classifique a base de teste.

Ao final teremos dois vetores que serão utilizados para o cálculo da acurácia:

- um vetor contendo a nova classificação "y_hipótese", e
- o vetor y_teste

Esses dois vetores serão usados como entrada para função "confmat", e como retorno teremos a matriz de confusão e a acurácia no seguinte formato:

```
[C, RATE]=confmat(Y,T)
Entrada:
     Y é o vetor predito (Y predito)
     T é o vetor real (Target)
Saída:
    C é a matriz de confusão onde as linhas representam o valor da classe real e as colunas o valor pretido.
     RATE é um vetor de duas posições, [PORCENT QUANT]
    PORCENT indicando a porcentagem de valores corretamente classificados.
    QUANT indicando a quantidade de valores corretamente classificados.
```

1) Implemente o KNN e mostre o resultado da classificação plotando a matriz de confusão e a taxa de acurácia com o valor de k=3. (3 pontos).

(Sugestão de implementação ao final do trabalho)

```
conf =

    14     1
     1    14

rate =

    93.333   28.000
```

conf → Matrix de confusão
Rate[1] → Acurácia
Rate[2] → Acertos

Regressão Logística

2) Altere a 6ª Questão do trabalho 1 de regressão linear com gradiente descendente para realizar classificação com regressão logística. Utilize a base de dados "alunos.txt". Plote apenas o gráfico J, e a acurácia. [Obs: Para essa questão específica, pode-se utilizar a mesma base para treino e para teste.] (2 Pontos)

Sugestão: Normalize os dados: `X = (X.-mean(X))./std(X);`

Utilize alpha = 0.1, epocas_max = 200

conf → Matrix de confusão
Rate[1] → Acurácia
Rate[2] → Acertos

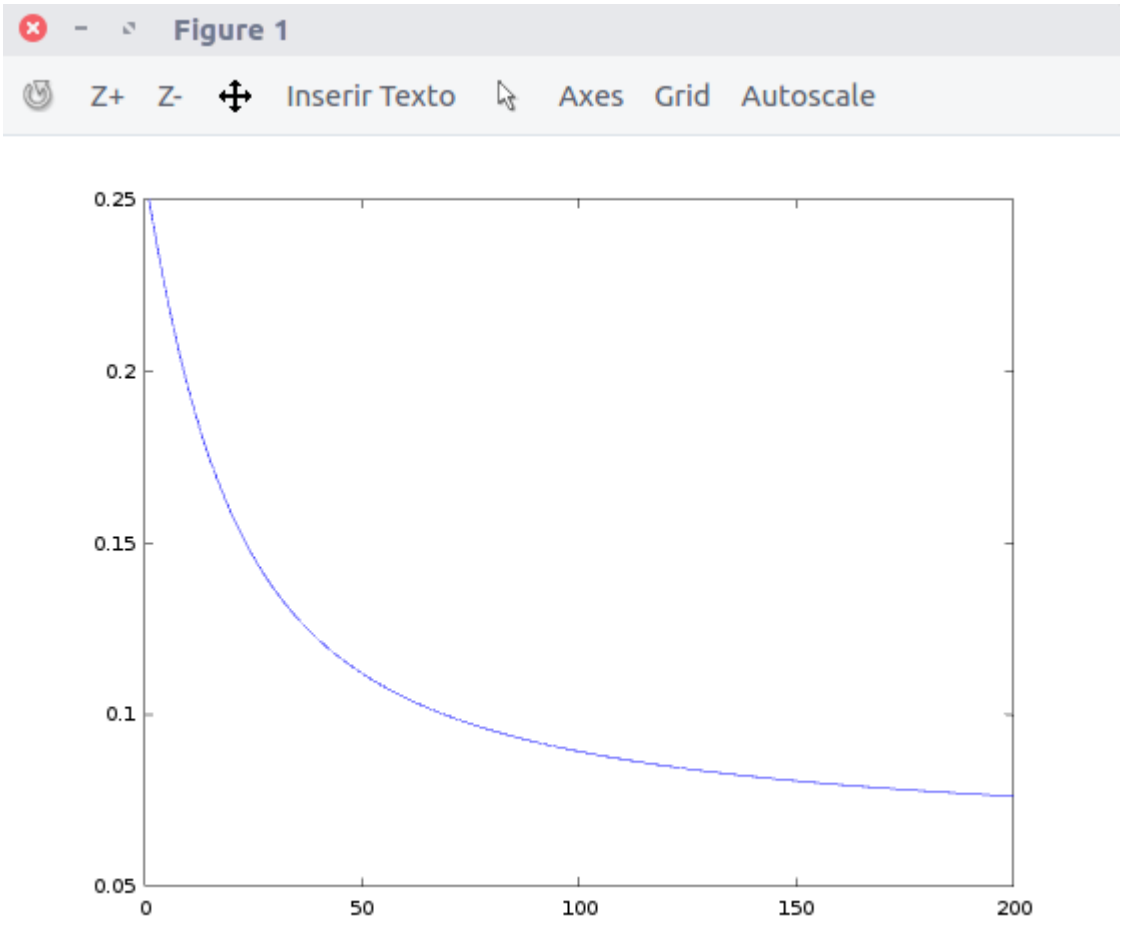
3D:

```
rate =

   90   90

conf =

   35    5
    5   55
```



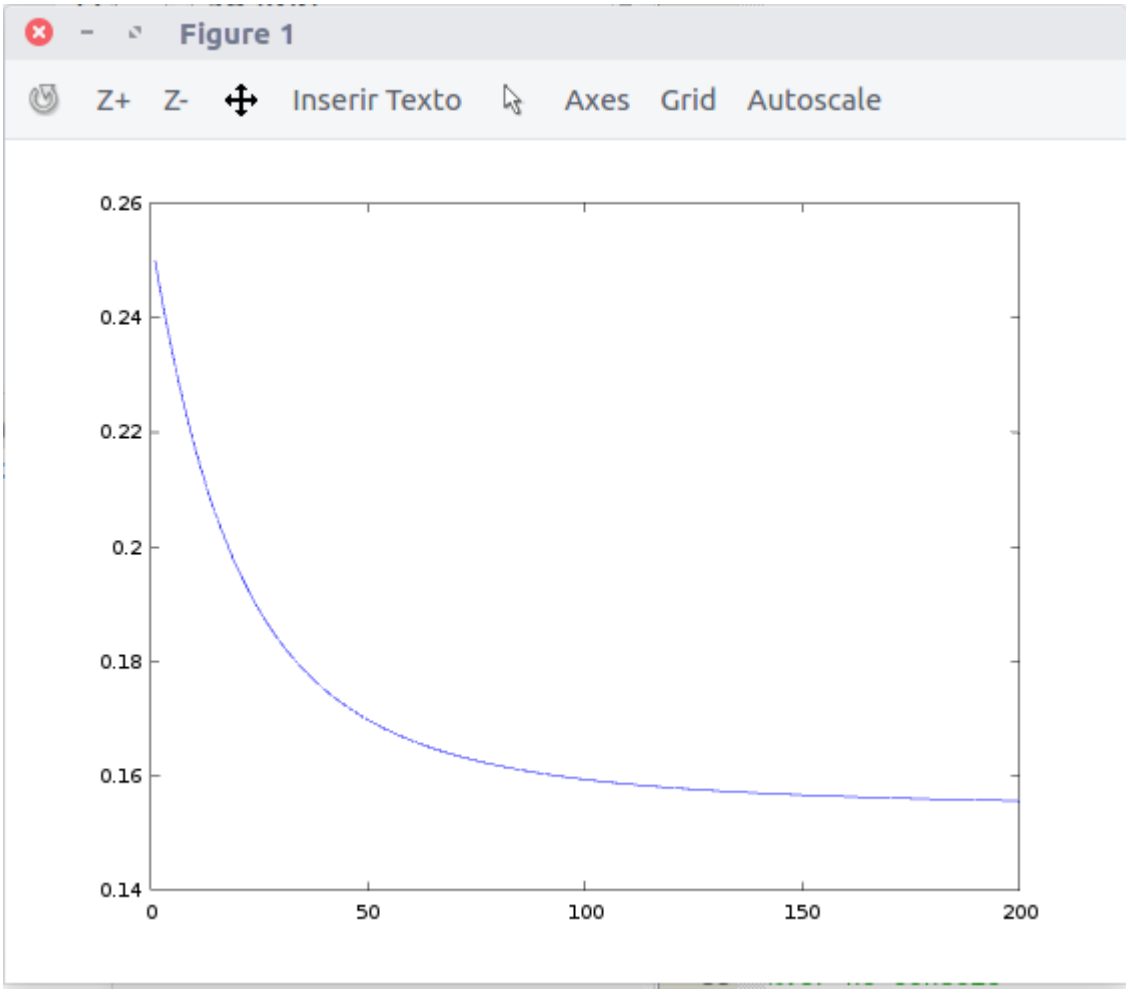
2D-X1:

```
rate =

   80   80

conf =

   28   12
    8   52
```



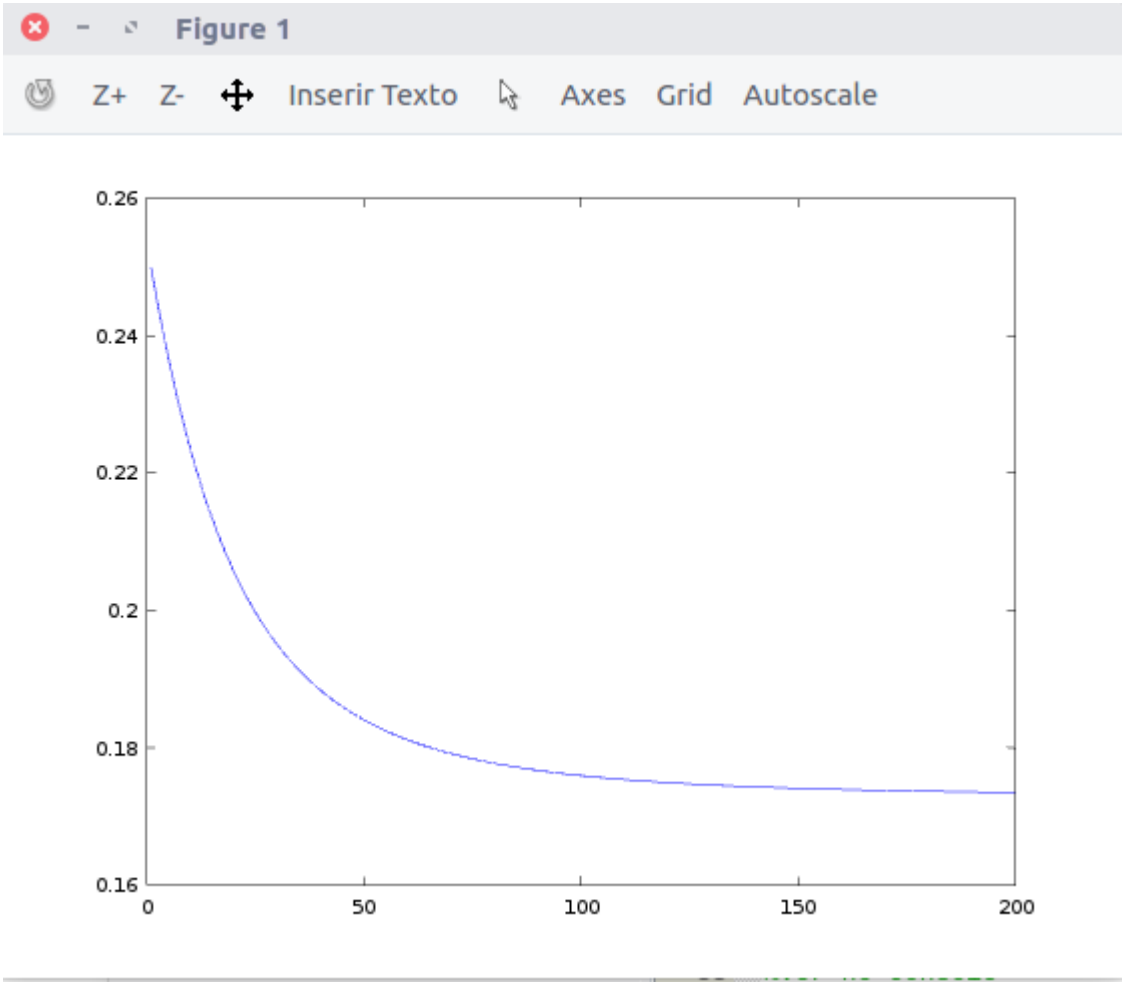
2D-X2:

```
rate =

   75   75

conf =

   26   14
   11   49
```



3) Indique as principais diferenças entre a regressão linear com gradiente descendente e a regressão logística. Em outras palavras, onde ocorreram as principais alterações conceituais na resolução da questão 4. (1 Ponto)

Basicamente na hipotese , transformar as saida em classes e a normalisação dos dados(o que gera um fator de correção maluco e uma tendencia a 1 sempre).
Porem, há algumas diferenças que percebi,mas não sei se é da base ou do algoritimo:
o erro tende a minimisar muito mais rapido apartir de 250 ,já não fazia muita diferença.
Parece nessecitar de mais processamento em cada epoca.
Erros minimos na hipotese tende a ter Resutados completamente malucos(no meu caso gerendo numeros imaginarios para todo lugar)
Plotar é especialmente dificil, apesar de fazendo na marra dá...

Sugestão para implementação em Octave do KNN

Pseudo-código

Lê uma base de dados X com um rótulos y e classifica um novo exemplo x, usando k vizinhos mais próximos de x em X.

```
Classifica_KNN(X,y,x,k){
        para i de 1 até m faça
                Compute a distancia d(X(i),x)
        fim para
        Compute um conjunto I contendo os índices das k menores distâncias
        d(X(i),x).  retorne o rótulo majoritário em {yi onde i pertence a I}
}
```

Octave

Cálculo da distância

```
distancia = @(x,M) sqrt(sum((M.-x).^2,2));
```

Calcula a distância de um único vetor x para cada vetor da matriz M (no contexto do knn, cada linha da matriz M representa um exemplo da base de treino).

A função retorna um vetor de distâncias.

Ordenação

```
[S, I] = sortrows (A)
S: vetor ordenado
I: Índice correspondente no vetor original
```

(No contexto do knn, quando sortrows é utilizada para ordenar o vetor de distâncias, os índices I estarão associados ao vetor de y_treino, ou seja, os k primeiros valores de I representarão os k mais próximos do novo exemplo “x_novo”).

Contagem de elementos em vetor

```
sum(y ==0 )
histc(y_I, 0)
```

Votação

```
[w iw] = max(vote_for_1);
W é o elemento mais votado, iw é o índice do elemento mais votado
Iw é o índice do elemento mais votado
```

(Obs: pode-se associar os índices às classes, porém a indexação do octave inicia em 1, dificultado a representação da classe 0)