MS211 - Projeto 2
Identificação de Lixo Eletrônico (SPAM)

Diogo Teles Sant'Anna - RA: 169966
Luciano Gigantelli Zago - RA: 182835

Introdução

O projeto tem como objetivo utilizar a técnica de aprendizado de máquina *extreme learning machine* (ELM) para identificar spams automaticamente. Em linhas gerais, a ELM é sintetizada com base num conjunto de mensagens já identificadas como spams ou não-spams, e então aplicamos a técnica para identificar spams dentre um outro conjunto de mensagens teste e avaliamos sua eficiência.

Para sintetizar a ELM usamos conhecimentos de cálculo numérico para obter o sistema linear que resolve o problema de quadrados mínimos e também para identificar o melhor método que resolve esse sistema.

Foi utilizado o programa GNU Octave para o desenvolvimento do trabalho, por ser um software livre. Usamos também o Git para versionamento e armazenamento do código.

Procedimentos

Início

Primeiramente, atribuimos as constantes definidas no enunciado:
d = 57; m = 3500; n = 1000;
Depois, geramos os vetores definidos no enunciado:
W = randn(n, d); b = randn(n, 1); G = tanh(W*Xtr+b);
randn é o comando que gera valores aleatórios de acordo com a distruibuição normal
tanh calcula a tangente hiperbólica.

Determinação dos valores de alpha pelo método dos mínimos quadrados

Do enunciado:

$$J(\alpha_1, ..., \alpha_n) = \sum_{k=1}^{m} (\alpha_1 g_1(X_{tr}(:,k)) + \cdots + \alpha_n g_n(Xtr(:,k)) - y_{tr}(k))^2$$

Desenvolvendo o método dos mínimos quadrados, chegamos no sistema linear a seguir:

$$\begin{bmatrix} <\vec{g}_1, \vec{g}_1> & \dots & <\vec{g}_1, \vec{g}_n> \\ \vdots & \ddots & \\ <\vec{g}_n, \vec{g}_1> & & <\vec{g}_n, \vec{g}_n> \end{bmatrix} \times \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} <\vec{y_{tr}}, \vec{g}_1> \\ \vdots \\ <\vec{y_{tr}}, \vec{g}_n> \end{bmatrix}$$

onde $\vec{g_i} = \begin{bmatrix} g_i(X_{tr}(:,1)) \\ \vdots \\ g_i(X_{tr}(:,n)) \end{bmatrix}$ e $\vec{y_{tr}} = \begin{bmatrix} y_{tr}(1) \\ \vdots \\ y_{tr}(n) \end{bmatrix}$

Agora, como também exposto no enunciado deste trabalho:

$G(i,k) = g_i(X_{tr}(:,k))$ e, portanto, $\vec{g_i} = \begin{bmatrix} g_i(X_{tr}(:,1)) \\ \vdots \\ g_i(X_{tr}(:,n)) \end{bmatrix} = \begin{bmatrix} G(i,1) \\ \vdots \\ G(i,n) \end{bmatrix} = G(i,:)$.

Sabendo que podemos reescrever o produto interno como $< \vec{a}, \vec{b} >= \vec{a} \times \vec{b}'$, dados $\vec{a}, \vec{b} \in \mathbb{R}^{1\times n}$ e b' equivalente à b transposta, reescrevemos a matriz do sistema linear como:

$$\begin{bmatrix} < \vec{g}_1, \vec{g}_1 > & \dots & < \vec{g}_1, \vec{g}_n > \\ \vdots & \ddots & \\ < \vec{g}_n, \vec{g}_1 > & & < \vec{g}_n, \vec{g}_n > \end{bmatrix} = \begin{bmatrix} G(1,:) \times G(1,:)' & \dots & G(1,:) \times G(n,:)' \\ \vdots & \ddots & \\ G(n,:) \times G(1,:)' & & G(n,:) \times G(n,:)' \end{bmatrix} =$$

$$\begin{bmatrix} G(1,:) \times G'(:,1) & \dots & G(1,:) \times G'(:,n) \\ \vdots & \ddots & \\ G(n,:) \times G'(1,:) & & G(n,:) \times G'(:,n) \end{bmatrix} = G \times G'$$

Analogamente, podemos refatorar o outro termo do sistema linear da seguinte forma:

$$\begin{bmatrix} < \vec{y_{tr}}, \vec{g}_1 > \\ \vdots \\ < \vec{y_{tr}}, \vec{g}_n > \end{bmatrix} = \begin{bmatrix} G(1,:) \times \vec{y_{tr}}' \\ \vdots \\ G(n,:) \times \vec{y_{tr}}' \end{bmatrix} = G \times \vec{y_{tr}}'$$

Por fim, a execução do método dos mínimos quadrados neste contexto se resume à resolução do sistema linear $\vec{A} \times \vec{x} = \vec{b}$, com $\vec{A} = G \times G'$, $\vec{x} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$ e $\vec{b} = G \times \vec{y_{tr}}$

Solução do sistema linear

Dado o sistema linear, analisamos as propriedades da matriz A para determinar qual o melhor método para resolução do sistema.

Usamos o método da decomposição de Cholesky já que pudemos demonstrar que a matriz A é simétrica e definida positiva. Segue demonstração:

Podemos visualizar que A é simétrica simplesmente pelo fato dela ser resultado de um produto de uma matriz com sua transposta:

$$A = GG' = \begin{bmatrix} G(1,:) \times G'(:,1) & \dots & G(1,:) \times G'(:,n) \\ \vdots & \ddots & \vdots \\ G(n,:) \times G'(:,1) & \dots & G(n,:) \times G'(:,n) \end{bmatrix} =$$

$$\begin{bmatrix} G(1,:) \times G(1,:) & \dots & G(1,:) \times G(n,:) \\ \vdots & \ddots & \vdots \\ G(n,:) \times G(1,:) & \dots & G(n,:) \times G(n,:) \end{bmatrix} = \begin{bmatrix} < G(1,:), G(1,:) > & \dots & < G(1,:), G(n,:) > \\ \vdots & \ddots & \vdots \\ < G(n,:), G(1,:) > & \dots & < G(n,:), G(n,:) > \end{bmatrix}$$

Portanto, dado a comutatividade em produtos escalares, a matriz A é simétrica.

Sabemos, agora, que A é definida positiva se $x'Ax > 0 \ \forall \ x \in R^n \mid x \neq 0$, mas, desenvolvendo a expressão temos: $x'Ax = x'GG'x = (G'x)'(G'x) = ||G'x||^2 \geq 0$

Provamos, portanto, que $x'Ax \geq 0$. Para atestar que $x'Ax = ||G'x||^2 \neq 0$, basta provar que $G'x \neq 0$, ou seja, que G' é uma matriz com o máximo de linhas possíveis LI, ou seja, sendo o **rank** de uma matriz é o número de linhas/colunas linearmente independentes da matriz, $rank(G') = min(dim(G'))$, portanto, $G'x = 0$ somente quando $x = 0$. Porém, $x \neq 0$ como definido anteriormente. Dessa forma, $G'x \neq 0$.

Em nossa matriz G', $rank(G') = min(dim(G'))$= 1000 , então G' é LI. Portanto, A é definida positiva.

O comando C = **chol(A)** atribui a matriz A decomposta pelo método de Cholesky à C.

O comando \ resolve as substituições do sistema linear, sendo utilizado 2 vezes:
alpha = C\(C'\b);

Rede neural

Segundo o enunciado, a rede neural define uma função φ(x), em que x é um vetor contendo os dados de uma mensagem, que já veio implementada no arquivo RNA.m. Essa função RNA gerou um vetor s do tamanho das colunas da matriz Xtr/Xte através do comando s = **RNA(alpha, W, b, X)**;

Uma mensagem é caracterizada como spam se L < φ(x), caso contrário, ela não é caracterizada como spam.

Acurácia

Segundo o enunciado, AC = Número de mensagens identificadas corretamente pelo sistema / Número total de mensagens.

Para esse cálculo, foi relizado um laço que percorre todo o vetor s gerado pela rede artificial e compara com o vetor de controle y (1 é spam e -1 é não-spam) através de sentenças condicionais encadeadas para verificar e contar as mensagens corretas. De forma análoga ao código, se ((L < s(i)) e (y(i) == 1)) ou ((L >= s(i)) e (y(i) == -1)): correto++.

m = **size(s, 2)** atribui o tamanho de colunas de s à m

ac = correto/m.

Taxa de Falsos Positivos

Segundo o enunciado, TFP = Número de mensagens identificadas como spam pelo sistema mas que não são spams / Número de mensagens que não são spams.

Para esse cálculo, foi relizado um laço que percorre todo o vetor de controle y (-1 é não-spam) e compara com o vetor s gerado pela rede artificial através de sentenças condicionais encadeadas para verificar e contar as dectecções incorretas. De forma análoga ao código, se (y(i) == -1): notspam++. se ((y(i) == -1) e (L < s(i))): wrongspam++.

tfp = wrongspam/notspam.

Gráfico

Foi gerado um gráfico para comparar o Limiar, a Acurácia e a Taxa de Falsos Positivos.

Foi criado um laço que varia o limiar desde o limiar mínimo até o limiar máximo, ambos passados como argumento, com uma resolução de 0.1. A função cria 3 vetores, um para cada atributo, e as posições nos vetores são correspondentes.

O gráfico foi gerado a partir desses 3 vetores, com o comando **plot(L, AC, 'o-b', L, TFP, 'o-r')** e foi salvo com o comando **saveas(1, "graph", "png")**.

Questões

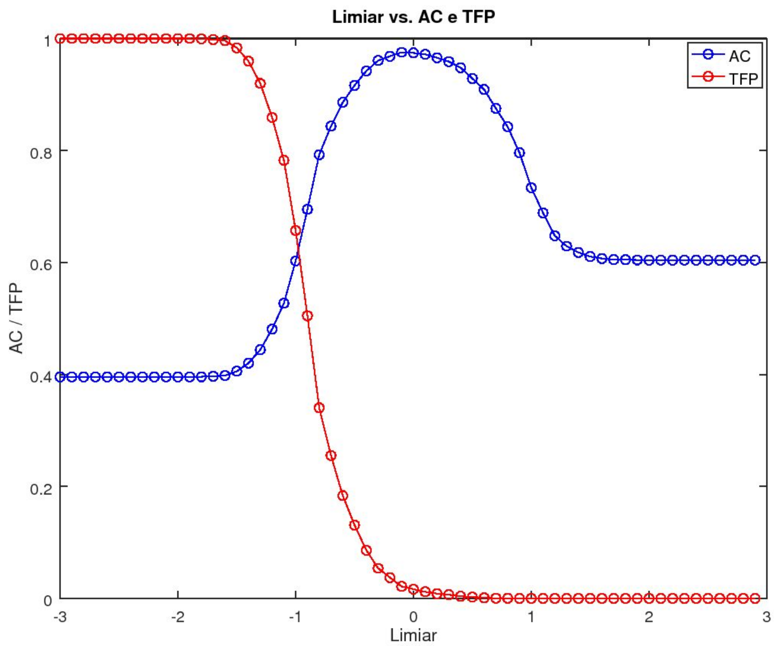
Questão 1:

Realizada segundo explicado no tópico Rede Neural

Questão 2:

L = -2; ac = 0.39571; tfp = 1
L = 0; ac = 0.97229; tfp = 0.018440
L = 2; ac = 0.60457; tfp = 0

Questão 3:



Segundo o gráfico, quanto menor o limiar, maior a taxa de falsos positivos; quanto maior o limiar, menor a taxa de falsos positivos. Quando o limiar é 0, ocorre a maior acurácia.

Quanto menor o limiar, mais mensagens são marcadas como spam. Quanto maior o limiar, menos mensagens são marcadas como spam. Pela definição da marcação como spam, se L< φ(x), a mensagem de x é marcada como spam. Se L <- -∞ , todas as mensagens serão marcadas como spam, se L <- ∞, nenhuma mensagem é marcada como spam. No nosso caso, para o conjunto de treinamento, o menor φ(x) obtido foi -1.8769 e o maior φ(x) obtido foi 2.0235. Isso significa que em L=-2, todas as mensagens são marcadas como spam e em L=2, poucas mensagens são marcadas como spam (2 mensagens). Isso foi possível verificar realizando um sort no vetor φ(x).

Questão 4:

O melhor valor para L, em que o TFP < 0.01 é quando TFP < 0.01 e o L é mínimo, pois pelo gráfico, naquela região, AC é decrescente. Através de um laço que incrementa L em 0.1 e uma condicional para verificar TFP < 0.01, pudemos obter os valores de L e TFP a seguir:

L = 0.20000; ac = 0.96714; tfp = 0.0085106

Questão 5:

L = 0.20000; ac = 0.92100; tfp = 0.049180

Questão 6:

Comparando o desempenho no conjunto de teste com o conjunto de treinamento, pode se dizer que os valores são próximos. Porém a acurácia é um pouco menor (em 0.05) e a taxa de falsos positivos é um pouco maior (em 0.04). Ou seja, o desempenho no conjunto de teste em geral é similar, mas suavemente pior do que no conjunto de treinamento.

Conclusões

O aprendizado através de redes neurais se mostrou muito eficaz quando avaliado sob um bom limiar de decisão. Observou-se que dependendo do limiar todas as mensagens ou nenhuma delas podem ser reportadas como spam, então a escolha desse valor é muito importante para classificar as mensagens.

Escolhendo um limiar L = 0.2 conseguimos nos dados de treinamento uma acurácia de 0.96714 e taxa de falso positivo de 0.0085106. Agora com os dados de testes, alcançamos uma acurácia de 0.92100 e uma taxa de falso positivo de 0.049180. A grande proximidade entre os valores obtidos com os dados de treinamento e de teste demonstra que a qualidade do treino da rede neural: tanto a acurácia quanto a TFP diferem de menos de 0.05 entre os resultados da aplicação da rede neural nos diferentes conjuntos de dados.

Código

main.m:

```
load DadosTreinamento.mat
load DadosTeste.mat

% Constantes definidas no enunciado
d = 57;
m = 3500;
n = 1000;

% Gera os vetores definidos no enunciado
W = randn(n, d);
b = randn(n, 1);
G = tanh(W*Xtr+b);

% Gera o vetor ALPHA
alpha = getAlphaVector(Xtr, ytr, G);

% Define a rede neural para os Dados de Treinamento
s = RNA(alpha, W, b, Xtr);

% Calcula o AC e o TFP para L = -2, 0 e 2
L = -2
ac = getAC(L, s, ytr)
tfp = getTFP(L, s, ytr)

L = 0
ac = getAC(L, s, ytr)
```

```
tfp = getTFP(L, s, ytr)

L = 2
ac = getAC(L, s, ytr)
tfp = getTFP(L, s, ytr)

% Gera o grafico e recebe o melhor limiar para TFPmax 0.01
Lbest = generateGraph(-3, 3, s, ytr, 0.01);

% Calcula o AC e o TFP do Conjunto de Teste com o melhor limiar
% definido no Conjunto de Treinamento
L = Lbest
s = RNA(alpha, W, b, Xte);
ac = getAC(L, s, yte)
tfp = getTFP(L, s, yte)
```

getAlphaVector.m:

```
function alpha = getAlphaVector(Xtr, ytr, G)
  % Inicializa os componentes do sistema linear
  A = (G*G');
  C = chol(A); % obtem A fatorado por Cholesky e salva em C
  b = G*ytr';

  alpha = C\(C'\b); % resolve as substituicoes da matriz de Cholesky
end
```

getAC.m:

```
function ac = getAC(L, s, y)

  m = size(s, 2); % pega o numero de colunas de s
  correto = 0;

  for i = 1:m
       if (L < s(i)) %spam
       if (y(i) == 1)
       correto++;
       end
       else % nao spam
       if (y(i) == -1)
       correto++;
       end
       end
  end

  ac = correto/m;
```

```
end
```

getTFP.m:

```
function tfp = getTFP(L, s, y)

  m = size(s, 2); % pega o numero de colunas de s
  wspam = 0; % wrong spam
  nspam = 0; % not spam

  for i = 1:m
      if (y(i) == -1) % se nao eh spam
      nspam++;
      if (L < s(i)) % mas foi detectado como spam
      wspam++;
      end
      end
  end

  tfp = wspam/nspam;
end
```

generateGraph.m:

```
function Lbest = generateGraph(Lmin, Lmax, s, ytr, TFPmax)

  % Define o numero de pontos no grafico atraves dos limites de L
  n = (Lmax - Lmin)*10;

  % Cria os vetores de L, AC, e TFP para serem usados no grafico
  L(1) = Lmin;
  AC(1) = getAC(L(1), s, ytr);
  TFP(1) = getTFP(L(1), s, ytr);
  for i = 2:n
      L(i) = L(i-1) + 0.1;
      AC(i) = getAC(L(i), s, ytr);
      TFP(i) = getTFP(L(i), s, ytr);
  end

  % Obtem o L que tem o melhor AC com o TFP no max o valor de lim
  for i=1:n
      if TFP(i) < TFPmax
      pos = i;
      break;
      end
  end
```

```
% Imprime os valores para o L encontrado anteriormente
L(pos)
AC(pos)
TFP(pos)

% Gera o grafico atraves dos vetores
plot(L, AC, 'o-b', L, TFP, 'o-r');
xlabel('Limiar');
  ylabel('AC / TFP');
  title('Limiar vs. AC e TFP');
  legend('AC','TFP');

saveas(1, "graph", "png");

% Retorna o L ideal obtido atraves do TFPmax
Lbest=L(pos);

end
```