UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
DISCIPLINA: SISTEMAS INTELIGENTES
PROFESSORA: DEBORAH MAGALHÃES
MONITORA: ORRANA

2a PRÁTICA COMPUTACIONAL

SVM com função kernel Gaussiana

1. Descrição do Trabalho

O objetivo dessa prática reside em adquirir noções básicas sobre como o SVM funciona e como utilizá-lo em conjunto com a função kernel da Gaussiana. A prática é dividida em **2 etapas**, descritas a seguir.

A **primeira** etapa conta com duas bases de dados (**ex6data1.mat**) e (**ex6data2.mat**) e os códigos necessários sua execução são:

- main_part1.m -> código principal que realiza a chamada para as demais funções
- svmTrain.m -> função de treinamento do SVM
- linearKernel.m -> implementação do kernel linear para o SVM
- visualizeBoundaryLinear.m -> plota a fronteira de decisão linear
- visualizeBoundary.m -> plota a fronteira de decisão não linear
* gaussianKernel.m -> implementação do kernel da Gaussiana
- svmPredict.m -> uma vez que o modelo foi treinado eu posso estimar uma fronteira com SVM

A primeira base de dados (**ex6data1.mat**) apresenta 2 classes com exemplos positivos (+) e os negativos (o), conforme ilustrado na Figura 1.

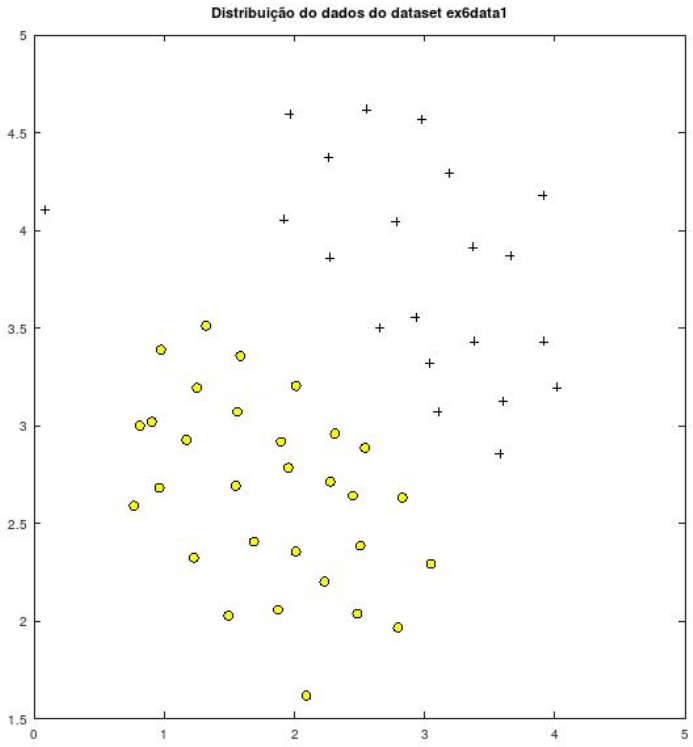


Figura 1: Gráfico de dispersão das amostras do primeiro dataset.

A segunda base de dados (**ex6data2.mat**) apresenta 2 classes com exemplos positivos (+) e os negativos (o). No entanto, conforme ilustrado na Figura 2, trata-se de uma fronteira de decisão não linear.

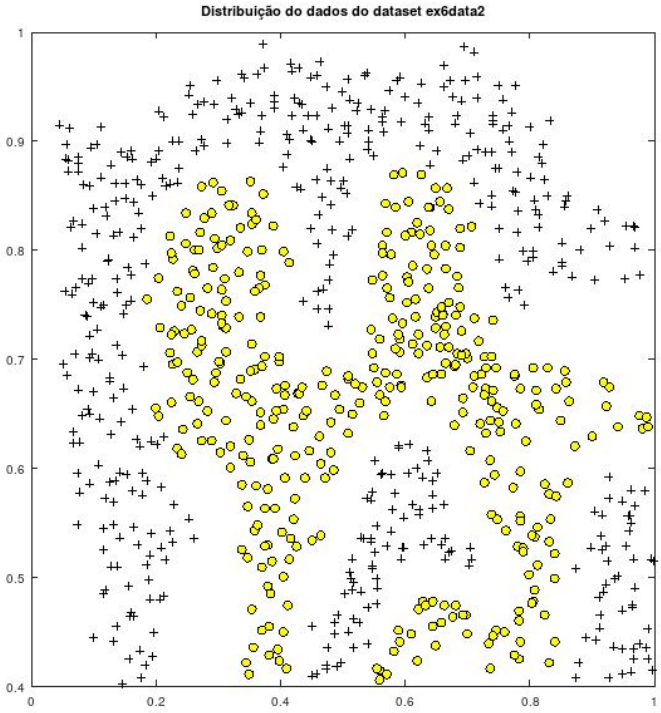


Figura 2: Gráfico de dispersão das amostras do dataset 2.

Na segunda parte do exercício, você usará SVMs para criar seu próprio filtro de spam. Essa etapa conta com um subconjunto de dados retirado do SpamAssassin Public Corpus[1]. O SVM será treinando para classificar se um determinado email é spam (y = 1) ou não spam (y = 0). Para tanto, apenas o corpo do email será utilizado, excluindo os cabeçalhos.

Os códigos necessários para realização da **segunda** parte do trabalho são:

- main_part2.m -> código principal que realiza a chamada para as demais funções

* processEmail.m -> função que realiza a filtragem na amostra de email

- vocab.txt -> contém o vocabulário das palavras mais frequentes em email de spam

* emailFeatures.m -> extrai as características da amostra de email

- spamTrain.mat -> conjunto de treino

- spamTest.mat -> conjunto de test;

Ainda da segunda etapa, você precisa converter cada email em um vetor de características binárias. Com esse fim, você desenvolverá a função de pré-processamento chamada **processEmail.m** de modo a produzir um vetor de índices de palavras para um determinado email. São disponibilizados 2 exemplos para serem processados **emailSample1.txt** e **emailSample2.txt**. A Figura 3 mostra

um exemplo de email que não passou pelo pré-processamento. Tal exemplo contém URL, endereço de email, números e valores monetários, fortes indicadores de spam.

```
> Anyone knows how much it costs to host a web portal ?
>
Well, it depends on how many visitors youre expecting.  This can be
anywhere from less than 10 bucks a month to a couple of $100.  You
should checkout http://www.rackspace.com/ or perhaps Amazon EC2 if
youre running something big..

To unsubscribe yourself from this mailing list, send an email to:
groupname-unsubscribe@egroups.com
```

Figura 3: exemplo de email (emailSample1.txt).

A função **processEmail.m** já realiza as seguintes filtragens:

- Converte todo e email para letra minúscula;
- Todas as tags HTML são removidas;
- Todas as URLs são substituídas pela expressão “httpaddr”
- Todas os emails são substituídos pela expressão "emailaddr"
- Todas os números são substituídos pela expressão "number"
- Todo cifrão é substituído pela expressão "dollar"
- Todas as palavras são reduzidas para o radical, exemplo, “include”, “includes”, “included”, e “including” são substituídas por “includ”
- Remover todas as pontuações, espaços, tabs, símbolo de nova linha

O resultado deste pré-processamento reside em uma lista de palavras para cada amostra de email, conforme Figura 4.

```
anyon know how much it cost to host a web portal well it depend on how
mani visitor your expect thi can be anywher from less than number buck
a month to a coupl of dollarnumb you should checkout httpaddr or perhap
amazon ecnumb if your run someth big to unsubscrib yourself from thi
mail list send an email to emailaddr
```

Figura 4: exemplo da amostra emailSample1.txt após pré-processamento.

Depois do pré-processamento, o objetivo é selecionar quais palavras são relevantes para discriminar se um email é spam ou não spam. Para isso, foi utilizado um vocabulário de palavras (**vocab.txt**) contendo todas as palavras, e seus respectivos identificadores, que ocorrem pelo menos 100 vezes no corpo de um spam, resultando em uma lista de 1899 palavras. De posse do vocabulário, é possível mapear cada palavra da amostra pré-processada com uma palavra do vocabulário, resultando em uma lista de identificadores, conforme apresentado na Figura 5.

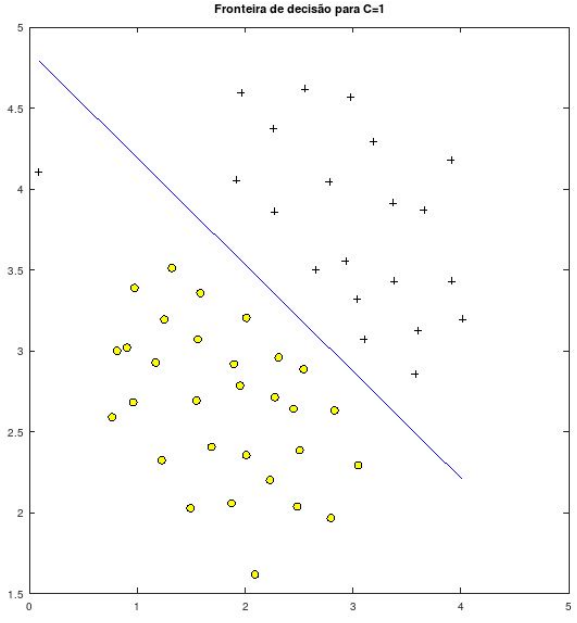
```
86 916 794 1077 883
370 1699 790 1822
1831 883 431 1171
794 1002 1893 1364
592 1676 238 162 89
688 945 1663 1120
1062 1699 375 1162
479 1893 1510 799
1182 1237 810 1895
1440 1547 181 1699
1758 1896 688 1676
992 961 1477 71 530
1699 531
```

Figura 5: índices de palavras presentes no email que ocorrem no vocabulário.

* São as funções que você deve modificar. Todas elas estão disponíveis no sigaa.
1: http://spamassassin.apache.org/old/publiccorpus/

1.1. O que deve ser feito?

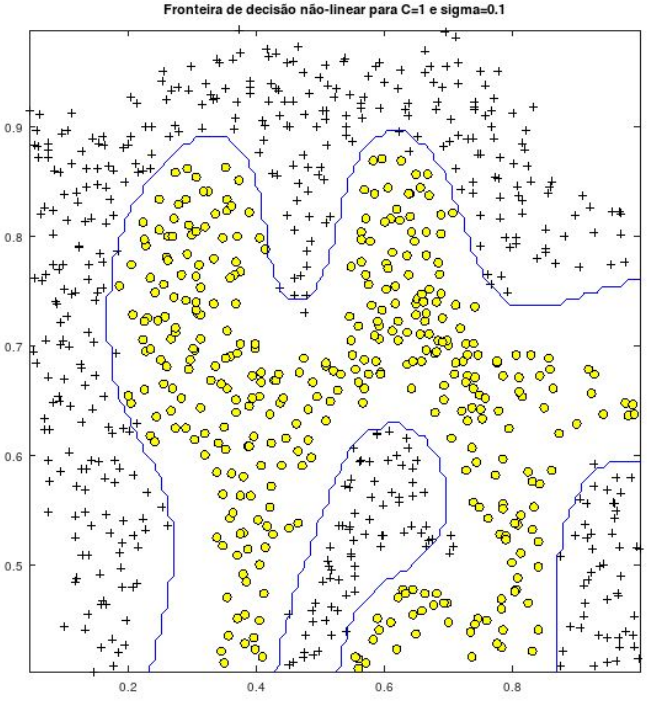
I. Primeira parte:
   - Variar o valor do parâmetro de entrada C = [1;10;100;1000] e verificar como se comporta a fronteira de decisão do dataset1. Em seguida, plote a fronteira de decisão para cada valor de C para fins de comparação, conforme ilustrado abaixo:



- O kernel gaussiano é uma função de similaridade que mede a "distância" entre um par de amostras, (x(i), x(j)). Tal função é parametrizada por, σ, que determina a rapidez com que a métrica de similaridade diminui à medida que os exemplos se afastam, conforme formalizado na Equação 1 .Portanto, você deve implementar a função kernel Gaussiana no arquivo gaussianKernel.m. A fim de ser validado, o valor retornado pela função para os parâmetros definidos no main_part1.m deve ser 0.324652;

$$K_{gaussian}(x^{(i)}, x^{(j)}) = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{k=1}^{n}(x_k^{(i)} - x_k^{(j)})^2}{2\sigma^2}\right)$$

- Como, o kernel da Gaussiana pode aprender uma fronteira de decisão não-linear, aplique-o para a segunda base de dados e plote os dados com a respectiva fronteira de decisão aprendida para diferentes valores de σ = [0.01, 0.03, 0.1, 0.3, 1] e C=1 , conforme exemplificado abaixo:



II. Segunda parte:

- Sua tarefa agora é inserir na função **processEmail.m** o mapeamento entre o resultado do pré-processamento e um vetor com os índices das palavras que ocorrem no vocabulário. No código, você recebe uma string str que é uma única palavra do email processado. Você deve procurar a palavra na lista de vocabulário **vocabList.txt** e descobrir se a palavra existe na lista de vocabulário. Se a palavra existir, você deve adicionar o índice da palavra no vetor **word_indices**. Se a palavra não existir e, portanto, não estiver no vocabulário, você pode pular para a próxima palavra. Você deve encontrar um resultado semelhante ao da Figura 5 para a amostra **emailSample1.txt**.

  Dica: a função strcmp (str1, str2) compara duas strings e retorna 1, se as duas seqüências forem equivalentes.

- Agora você deve desenvolver dentro da função **emailFeatures.m** a extração de características a partir do vetor **word_indices.** Cada email será convertido em um vetor de 1899 posições composto de 0s e 1s, onde 1 representa a presença da palavra daquele índice no email. Após a extração de características, a função principal **main_part2.m** carregará um conjunto de dados de treinamento pré-processado (spamTrain.mat) que será usado para treinar um classificador SVM. O spamTrain.mat contém 4000 exemplos de e-mails de spam e não spam, enquanto o conjunto spamTest.mat contém 1000 exemplos de teste. Uma vez concluído o treinamento, a acurácia do SVM deve ficar em torno de 99,8% para o treinamento e 98,5% para o teste.

2. Avaliação

Este trabalho corresponde a uma parte da segunda avaliação parcial da disciplina e deverá ser entregue no dia **02/05**. O trabalho poderá ser apresentado de dupla e assumirá o valor de **0-2.5**.

Os seguintes critérios serão considerados na avaliação:

1. Atender ao que foi pedido na descrição deste documento;
2. Compreender os conceitos discutidos em sala;
3. Código está executando sem erros.

Atenção: se identificada a cópia de código, a nota **zero** será atribuída aos envolvidos.