

**UNIVERSIDADE KATYAVALA BWILA**

**INSTITUTO SUPERIOR POLITÉCNICO**

**TRABALHO DE APRENDIZAGEM AUTOMATICA**

**PRÁTICA DE LABORATÓRIO 3:**

**TREINO DE CLASSIFICADORES COM O SUPORTE DA FERRAMENTA WEKA**



Benguela, Novembro de 2015



**UNIVERSIDADE KATYAVALA BWILA**

**INSTITUTO SUPERIOR POLITÉCNICO**

**TRABALHO DE APRENDIZAGEM AUTOMATICA**

**PRÁTICA DE LABORATÓRIO 3:**

**TREINO DE CLASSIFICADORES COM O SUPORTE DA FERRAMENTA WEKA**

4º Ano – Ciências da Computação

Período – Regular

O Docente

Elaborado por:

Venâncio Ezequias Sapalo & Zinga Firmino René

Dr. Lazaro Makili

&

Moisés Ferreira

Benguela, Novembro de 2015

**OBJECTIVOS**

Com a prática se pretende alcançar os seguintes objectivos:

* Utilizar os métodos baseados em classificadores lineares com máquina de vector de suporte para treinar classificadores para conjuntos de dados concretos
* Analisar o modelo aprendido ao treinar um classificador máquina de vector de suporte
* Analisar uma estratégia de selecção do valor ideal para o parâmetro de um classificador com um conjunto de dados concreto
* Explorar o efeito produzido no rendimento de um classificador linear com máquina de vector de suporte pela introdução de parámetro de penalização num conjunto de dados

**INTRODUÇÃO**

Nesta prática examinar-se-à a utilização das máquinas de vectores de suporte (SVM) em problemas de classificação.

Na mesma serão utilizados três conjuntos de dados. Dois dos conjuntos são sintéticos, tendo como finalidade a análise de conceitos básicos acerca das SVM, enquanto que o terceiro é um conjunto de dados real. Este último será utilizado para o estudo do procedimento de treino e teste de um classificador.

Com a prática se pretende alcançar os seguintes objectivos:

* Utilizar as máquinas de vectores de suporte para o treino de classificadores para conjuntos diversos.
* Analisar conceitos básicos relacionados ao método, tais como superfície de decisão, vectores de suporte, função *kernel*, etc.
* Empregar um procedimento para a busca de parâmetros adequados e treino de

**Operação de Pré-processamento:**

* **A normalização** que consiste em escalonar os dados de modo a caírem em uma faixa de pequenos valores. Tem como propósito principal minimizar problemas oriundo do uso de unidades.
* **A discretização** consiste na transformação de atributos numéricos em atributos nominal, essencial se o método inteligente só manuseia atributos nominais. Existem dois tipos de discretização: *não supervisionada*, não tem em conta as classes das instâncias, e *supervisionada*, tem em conta as classes das instâncias ao criar os intervalos (uteis para problemas de agrupamento de dados, onde não se conhecem classes de saída). O principal método de *discretização não supervisionada* existente no Weka é *Weka.filters.unsupervised.attribute.Discretize*. O mesmo implementa duas variantes: *discretização com intervalos igual largura* (seleccionado por defeito) e com *intervalos de igual frequência*.

**Métodos de validação**

* **Medidas de desempenho**: é definido mediante proporções de exemplos incorrectamente classificados de entre a totalidade de exemplos de teste. No caso de problemas de classificação a medida mais comum é a taxa de erro. Existem diversas medidas de desempenho como: Taxa de erro, Taxa de êxito, Especificidade, Sensibilidade, Proporções de falsos positivos;
* **Holdout**: Consiste em dividir os dados em duas partes, um para treino e outra parte para teste. Geralmente usa-se 1/3 dos dados como conjunto de teste e o resto para treinar o algoritmo;
* **Validação cruzada**: Consiste em dividir os dados em k subconjuntos com tamanhos aproximadamente iguais, sendo utilizados k-1 subconjuntos para treinar algoritmo e o resto pra testar. A estimação da taxa de erro é feita através da média das taxas obtidas ao repartir o processo k vezes com as diferentes combinações de k-1 subconjunto de treino.
* **Matriz de Confusão**: é utilizada em classificação, possibilita uma visualização inequívoca dos resultados de um determinado modelo. Os resultados são apresentados sob forma de tabela de duas entradas (considerando problemas de apenas duas classes): uma das entradas é constituída pelas classes desejadas, a outra pelas classes prevista pelo modelo. As células, por sua vez, são preenchidas com o número de instâncias que correspondem ao cruzamento de entrada.

**Tarefa 1: classificação com dados linearmente separáveis**

Nesta actividade utilizaremos o Weka para treinar um classificador SVM para o caso de um problema simples, cujo objectivo é o reforço do entendimento básico dos conceitos relacionados às máquinas de vectores de suporte.

Treinemos primeiro uma SVM para um problema linearmente separável. Para tal utilizaremos o método SMO. SMO, abreviatura de *Sequential Minimal Optimization*, é um algoritmo que permite resolver o problema de optimização quadrática subjacente ao treino das SVM.

1- No painel *Preprocess*, carregamos o conjunto de dados *DataSet\_2D\_LinearlySeparable*. Observamos quais são os atributos e os valores correspondentes e as classes. Inspeccione os dados utilizando a secção *Visualize*.

R: os atributos do conjunto de dados são: x, y.

E os valores correspondentes são:

Para o atributo x são: Minimum 0; Maximum 15; Mean 6,438; StdDev 4,83;

Para o atributo y são: Minimum 0; Maximum 16; Mean 6,563; StdDev 4,98;

As classes são: A, B.

Os dados utilizados na secção visualize são:

Para o atributo x: 10, 6;

Para o atributo y: 11, 5;

Para a classe A: 8;

Para a classe B: 8; como mostra o histograma:

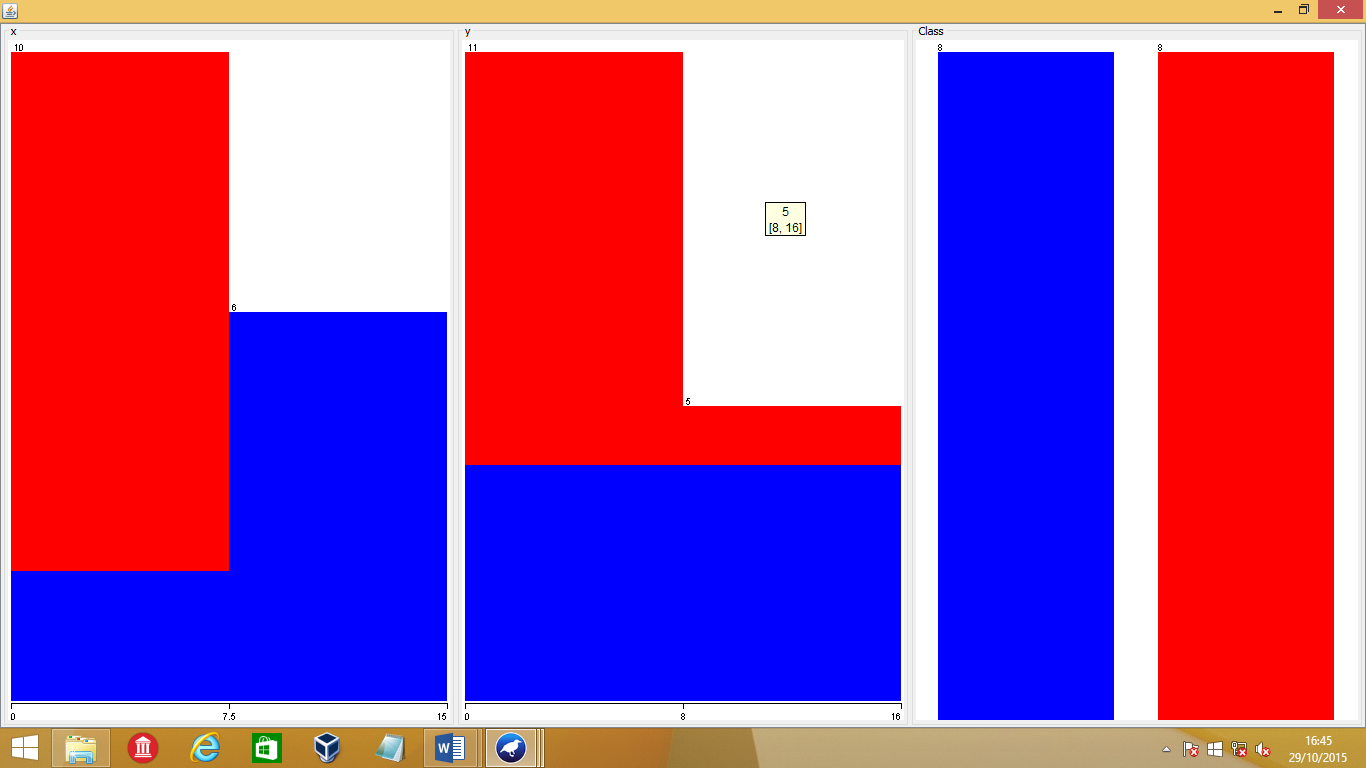
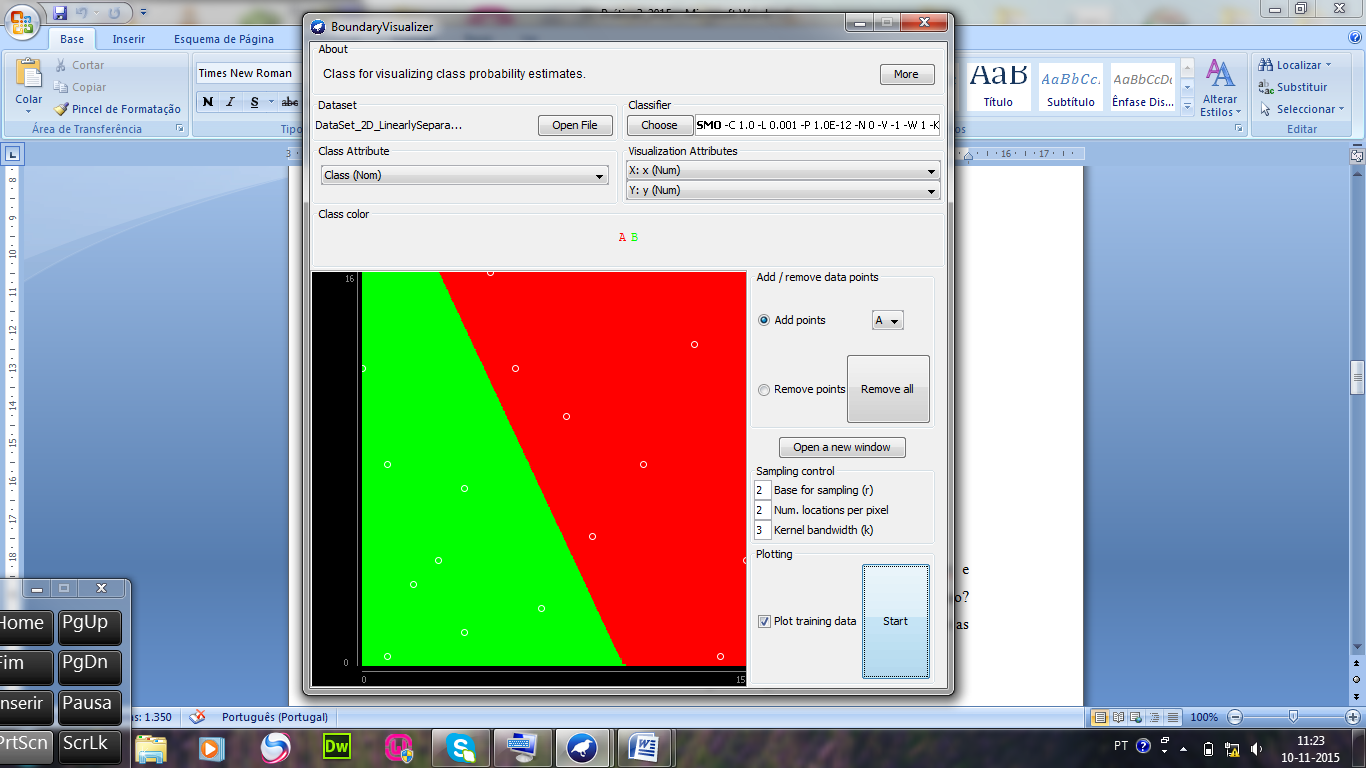


Fig.nº1

B-A outra forma consiste em utilizar a aplicação *ClassificationBoundaryVisualizer* do WEKA. Esta não está incluída no Explorador, porisso, é necessário abrir a aplicação numa nova janela.Construimos a superfície de separação para o mesmo conjunto de dados e classificador utilizado no passo anterior.

* + 1. No WEKA abra a interface de comandos (*SimpleCLI*) e introduzimos o comando *java weka.gui.boundaryvisualizer.BoundaryVisualizer*.
    2. Seleccionamos o conjunto de dados e o classificador.
    3. Activamos a opção *Plot training data* e pressionamos o botão *Start*. Assim como ilustra a nossa figura.



7. Analisamos as superfícies de decisão obtidas e descreva os erros de classificação.

R: Para um conjunto de dados linearmente separável existem várias superfícies de decisão

Possíveis.

1. Acresecentamos e removemos alguns pontos, no *Boundary Visualizer*, e observamos o que aconteceu com a superfície de decisão. Se alterou em algum caso? Descrevemos a quantidade de pontos que acrescentamos/removemos, em que áreas e as alterações que ocorrem na superfície de separação.

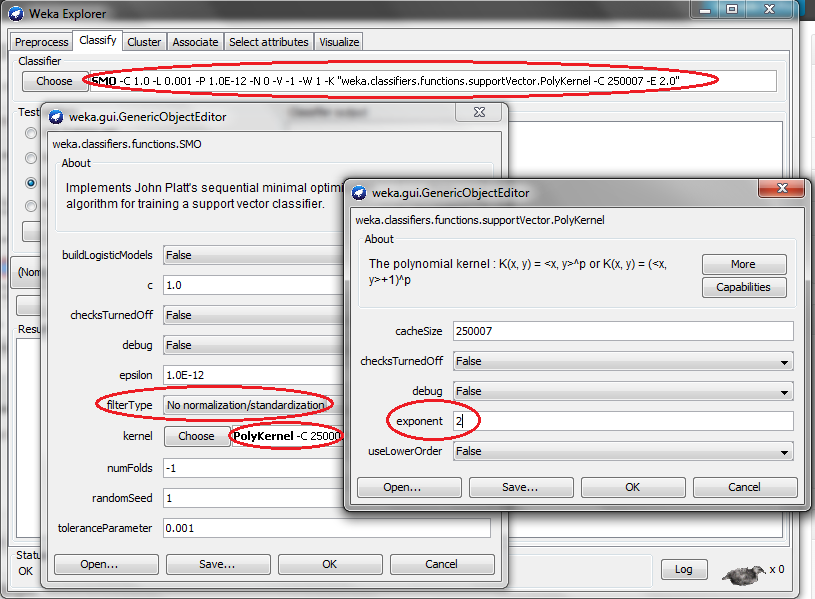
8. R: Descrever os erros cometidos e acrescentar mais pontos para ver se altera alguma coisa

Visto que para a classe A existem 8 erros, e para a classe B também existem 8 erros.

Acrescentar pontos na classe A, B.

Provemos agora treinar o classificador com um *kernel* polinomial primeiro com expoente 2 e depois com expoente 4.

1. Para tal abrimos a caixa de diálogo de opções do classificador (clicamos sobre a barra existente à frente do botão *Choose*) e nas opções do *kernel* alteramos o valor do expoente (clicamos na caixa de diálogo correspondente à propriedade *kernel*).



1. Executamos o classificador.
2. Visualizamos os resultados da classificação utilizando os dois métodos anteriores.
3. Observamos o que aconteceu com a quantidade de vectores de suporte e com o erro de classificação em cada caso.
4. Ilustramos igualmente as suas conclusões.

Analise do conjunto de dados não separáveis linearmente(*DataSet\_2D\_NoLinearlySeparable*).

R: O conjunto de dados tem 3 atributos,9 instacias ,9 sum of weigth ,composto por 2 atributos do tipo numérico e um do tipo nominal.

Após a execução do conjunto de dados *DataSet\_2D\_NoLinearlySeparable com* As opções por defeito correspondem a um *kernel* linear (*kernel* polinómico com expoente 1). É possível observar o tempo que o weka levou para testar o conjunto de dados(0,02 segundos), a percentagem de instacias corectamente classificadas(55,556 %), a percentagem de instacias mal classificadas(44,4444 %) e com o total de 9 instacias...

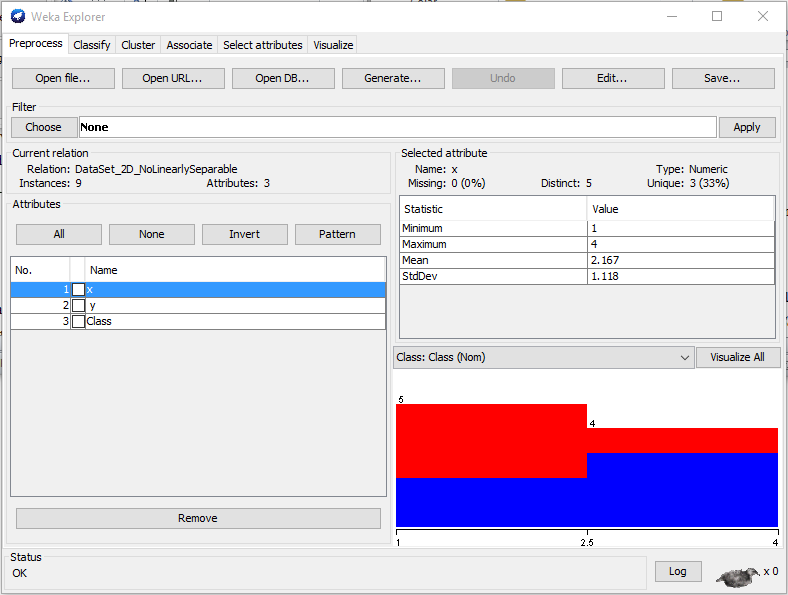
**Tarefa 2: dados não separáveis linearmente**

Nesta actividade vamos experimentar a classificação de um conjunto de dados não separáveis no espaço de entradas. Provaremos o efeito da alteração do parâmetro de penalização C e das diferentes opções de *kernel* nos resultados da classificação.

1. Abra o ficheiro *DataSet\_2D\_NoLinearlySeparable*
2. Visualize e analise os dados correspondentes.
3. Seleccione como classificador SMO e execute-o com as opções por defeito, usando *Use training set* nas opções de teste. As opções por defeito correspondem a um *kernel* linear (*kernel* polinómico com expoente 1).
4. Visualize os resultados da classificação. O que é que se observa?
5. Na caixa de diálogo de opções do classificador experimente alterar o valor do parâmetro de penalização (C) para 10, 100 e depois para 1000. Se observa algum efeito nos resultados da classificação?
6. Independentemente da opção utilizada poderá observar que em nenhum dos casos foi possível classificar correctamente a totalidade dos dados. Isto se deve a que o *kernel* utilizado por defeito é linear e o conjunto de dados utilizado não é separável linearmente no espaço de entradas. Será separável noutro espaço? Provemos para tal outras opções de *kernel*.
7. Treine o mesmo classificador, utilizando desta vez o *kernel* polinomial. Seleccione nas opções do *kernel* o expoente 2.
8. Observe os resultados da classificação. Foram separados correctamente os dados?
9. Prove valores altos de C, 1000, 10000, 100000, etc. e verifique se é possível separar completamente os dados de treino.

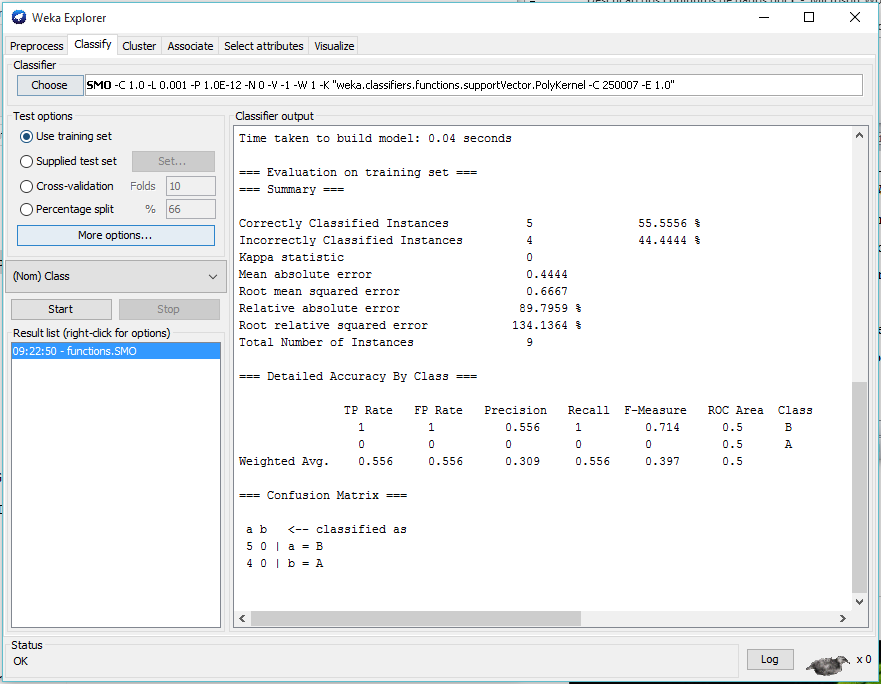
**Tarefa 2: dados não separáveis linearmente**

Nesta actividade vamos experimentar a classificação de um conjunto de dados não separáveis no espaço de entradas. Provaremos o efeito da alteração do parâmetro de penalização C e das diferentes opções de *kernel* nos resultados da classificação.

1. Abrimos o ficheiro *DataSet\_2D\_NoLinearlySeparable*
2. Visualizamos e analisamos os dados correspondentes obtivemos os seguintes resultados:

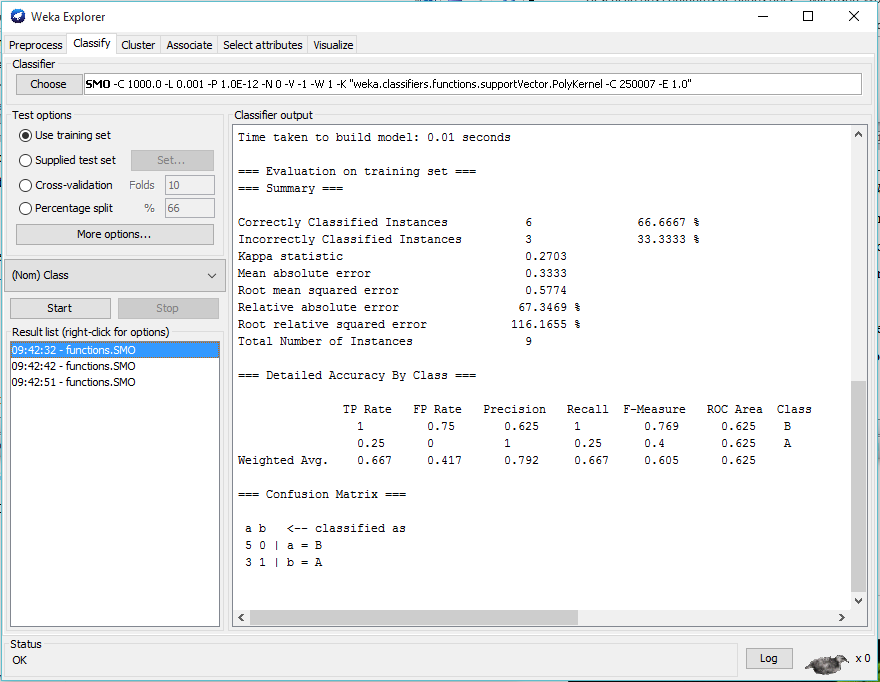
O conjunto de dados contido no ficheiro *DataSet\_2D\_NoLinearlySeparable* contém3 atributos, “X”, ”Y” e a ”Classe, 9 instâncias.

1. Seleccionamos como classificador SMO e executamos com as opções por defeito, usando *Use training set* nas opções de teste. As opções por defeito correspondem a um *kernel* linear (*kernel* polinómico com expoente 1).
2. Visualizamos os resultados da classificação:



O que é que se observa?

1. R: Observou-se que Após a execução do conjunto de dados *DataSet\_2D\_NoLinearlySeparable com* As opções por defeito correspondem a um *kernel* linear (*kernel* polinómico com expoente 1). É possível observar o tempo que o classificador levou para aprender associado ao conjunto de dados foi de “0,04 segundos”, 5 instâncias foram correctamente classificadas a que corresponde a uma percentagem de 55,5556 % e 4 instâncias foram incorrectamente classificadas a que corresponde a uma percentagem de 44,4444%
2. Na caixa de diálogo de opções do classificador alteramos o valor do parâmetro de penalização (C) para 10, 100 e depois para 1000. Observou-se os seguintes efeitos nos resultados da classificação:
3. Para (C) igual a 10:



Para o valor do parámetro (C) igual a 10:

Tempo de execução: 0,04 segundos;

Instâncias correctamente classificadas: 6, correspondente à 66,6667%;

Instâncias incorrectamente classificadas: 3, correspondente à 33,3333%;

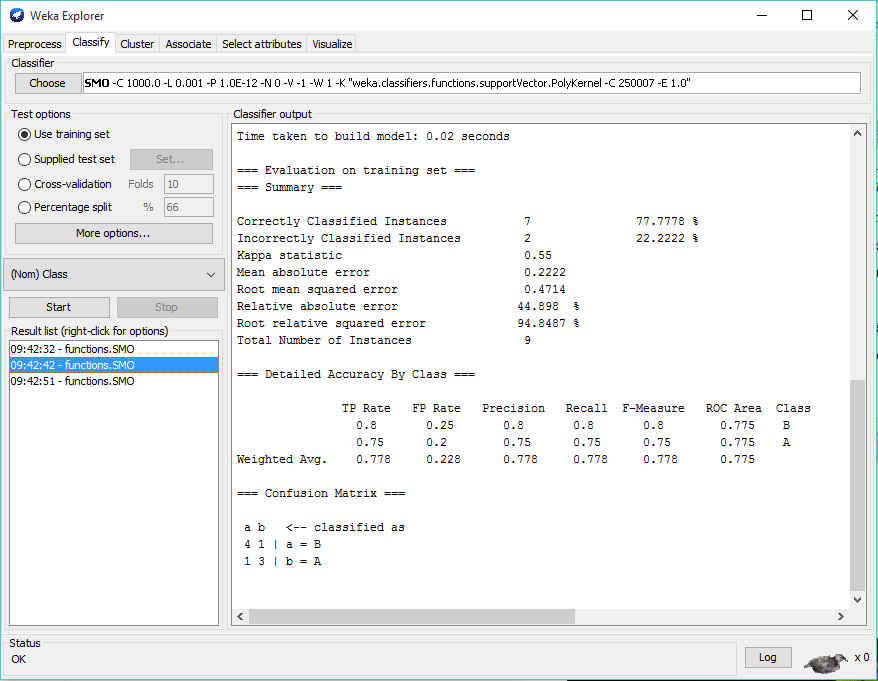
Valor do erro absoluto: 0,3333;

Valor do erro quadrático médio: 0,5774;

Valor do erro relactivo absoluto: 67,3469;

E outros valores de outros atributos....

1. Para (C) igual a 100:



Para o valor do parámetro (C) igual a 100:

Tempo de execução: 0,02 segundos;

Instâncias correctamente classificadas: 7, correspondente à 77,7778%;

Instâncias incorrectamente classificadas: 2, correspondente à 22,2222%;

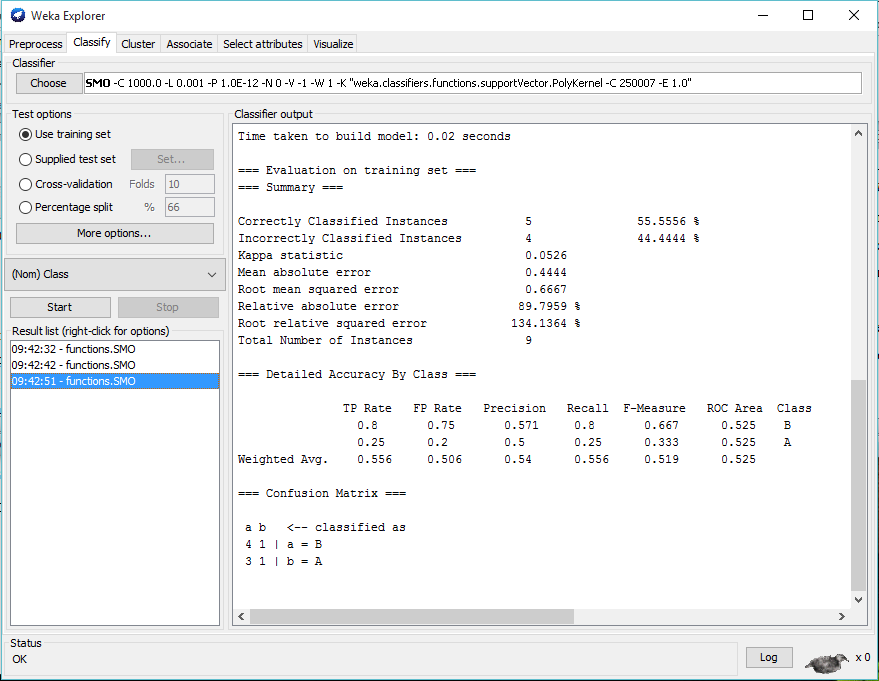
Valor do erro absoluto: 0,2222;

Valor do erro quadrático médio: 0,4714;

Valor do erro relactivo absoluto: 44,898;

E outros valores de outros atributos....

1. Para (C) igual a 1000:



Para o valor do parámetro (C) igual a 1000:

Tempo de execução: 0,02 segundos;

Instâncias correctamente classificadas: 5, correspondente à 55,5556%;

Instâncias incorrectamente classificadas: 4, correspondente à 44,4444%;

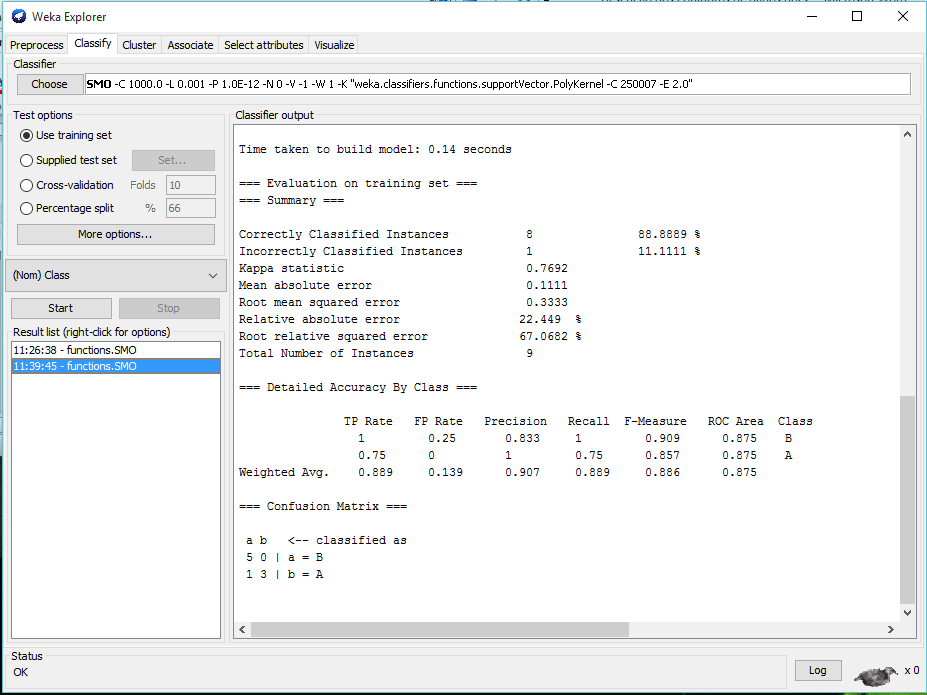
Valor do erro absoluto: 0,4444;

Valor do erro quadrático médio: 0,46667;

Valor do erro relactivo absoluto: 89,7959;

E outros valores de outros atributos....

1. Independentemente da opção utilizada poderá observar que em nenhum dos casos foi possível classificar correctamente a totalidade dos dados. Isto se deve a que o *kernel* utilizado por defeito é linear e o conjunto de dados utilizado não é separável linearmente no espaço de entradas. Será separável noutro espaço? Provemos para tal outras opções de *kernel*.
2. Treinamos o mesmo classificador, utilizando desta vez o *kernel* polinomial. Seleccione nas opções do *kernel* o expoente 2.
3. Observamos os resultados da classificação.



Foram separados correctamente os dados?

R: O classificador apresenta uma margem de erro absoluto de 0,1111, Os dados não foram correctamente separados no conjunto das 9 instâncias, oito (8) foram correctamente classififcadas a que corresponde a 88,8889% e uma (1) não foi correctamente classificada a que corresponde a 11,1111%.

1. R: Provando valores altos de C, 1000, 10000 e 100000, verificamos que é possível separar completamente os dados de treino utilizando a partir de um parámetro de penalização (C) 100000 em diante com o *kernel* polinomial com a opção do *kernel* o expoente 2.

**Tarefa 3: conjunto de dados real**

Investiguemos agora o processo de classificação de um conjunto de dados real, o conjunto p*ima indian diabetes*. Pode ver uma descrição do conjunto de dados no repositório da UCI.

Ao treinar uma SVM é necessário definir os valores adequados para uma série de parâmetros, o parâmetro de penalização C e do *kernel*, caso seja aplicável. Nesta tarefa veremos como fazer a determinação dos referidos parâmetros.

Dividamos primeiro o conjunto de dados em duas partes, conjuntos de treino e de teste.

1. Para tal, no Explorador basta fazer o seguinte:
   1. Conjunto de treino:
      1. Carregue o conjunto de dados *diabetes*.
      2. Seleccione o filtro *RemovePercentage* no painel de pré-processamento (*filters.unsupervised.instance.RemovePercentage*).
      3. Defina a percentagem correcta para a divisão dos dados, 60 % neste caso. Para tal faça click no espaço diante do botão *Choose* para abrir as opções do filtro.
      4. Aplique o filtro.
      5. Guarde os dados gerados como um novo ficheiro (*diabetes\_trn*)
   2. Conjunto de teste:
      1. Recarregue o conjunto de dados completo (ou use apenas o botão *Undo* para reverter as alterações para o conjunto de dados).
      2. Selecione o filtro *RemovePercentage* se ainda não estiver selecionado.
      3. Defina a propriedade como *invertSelection* como verdadeira. Para tal faça click no espaço diante do botão *Choose* para abrir as opções do filtro.
      4. Aplique o filtro.
      5. Guarde os dados gerados como novo ficheiro (*diabetes\_tst*).

Determinemos o melhor conjunto de parâmetros com base nos resultados de uma validação cruzada em 5 folhas realizada sobre o conjunto de treino.

1. Carregue o conjunto de dados de treino (*diabetes\_trn*).
2. Efectue de forma sucessiva a validação cruzada para as seguintes combinações de parâmetros:
   1. Kernel RBF: C = 0,01; 0,1; 1; 10; 100 G = 0,001; 0,01; 0,1; 1,0;10
   2. Kernel polinomial: C = 0,01; 0,1; 1; 10; 100; p = 2, 4, 8, 16
   3. Kernel linear: C = 0,01; 0,1; 1; 10; 100

**KernelRBF**

|  |  |  |
| --- | --- | --- |
| C | Gamma | Taxa de Erro |
| 0,01 | 0,001 | 0.3029 |
| 0,1 | 0,01 | 0.0297 |
| 1 | 0,1 | 0.3029 |
| 10 | 1.0 | 0.2345 |
| 100 | 10 | 0.2943 |

**Kernel Polinomial**

|  |  |  |
| --- | --- | --- |
| C | P | Taxa de Erro |
| 0,01 | 2 | 0.3029 |
| 0.1 | 4 | 0.2248 |
| 1 | 8 | 0.2997 |
| 10 | 16 | 0.3094 |
| 100 |  | 0.2184 |

**Kernel Linear**

|  |  |
| --- | --- |
| **C** | **Taxa de Erro** |
| 0.001 | 0.3029 |
| 0.01 | 0.3029 |
| 0.1 | 0.3029 |
| 1 | 0.3029 |
| 10 | 0.3029 |
| 100 | 0.3029 |

Os menores valores da taxa de erro são registados no caso Kernel RBF quando C=0.1 e G=0.01 e no Kernel Polinomial quando C=100 e G= 1.0 (Valor por defeito) e no caso do Kernel Linear a taxa de erro é constante com o valor de 0.3029

Treinemos agora o classificador utilizando as melhores combinações dos parâmetros correspondentes a cada *kernel* e apliquemos o mesmo aos dados de teste.

Carregue o conjunto de treino.

Seleccione o classificador e defina os valores dos parâmetros a utilizar.

Seleccione *Supplied test set* nas opções de teste e carregue o ficheiro de teste.

Treine o classificador e anote os resultados da classificação para cada alternativa testada.

**KernelRBF**

|  |  |  |
| --- | --- | --- |
| C | Gamma | Taxa de Erro |
| 0,01 | 0,001 | 0.3796 |
| 0,1 | 0,01 | 0.3796 |
| 1 | 0,1 | 0.4228 |
| 10 | 1.0 | 0.2882 |
| 100 | 10 | 0.0087 |

.

**Kernel Polinomial**

|  |  |  |
| --- | --- | --- |
| C | P | Taxa de Erro |
| 0,01 | 2 | 0.3818 |
| 0.1 | 4 | 0.2364 |
| 1 | 8 | 0.1887 |
| 10 | 16 | 0. |
| 100 |  | 0.2184 |

**Kernel Linear**

|  |  |
| --- | --- |
| **C** | **Taxa de Erro** |
| 0.001 | 0.3796 |
| 0.01 | 0.3796 |
| 0.1 | 0.3796 |
| 1 | 0.3796 |
| 10 | 0.3796 |
| 100 | 0.3796 |

Os resultados obtidos da classificação são distintos a quando da realização da mesma com validação cruzada. Em alguns casos os valores são maiores e outros menores comparando com o caso anterior.

**CONCLUSÕES**

Na tarefa 1 utilizamos treinamos um classificador SVM para o caso de um problema simples, cujo objectivo é o reforço do entendimento básico dos conceitos relacionados às máquinas de vectores de suporte. Começamos primeiro por SVM para um problema linearmente separável. Para tal utilizaremos o método SMO.

Na tarefa 2 experimentamos a classificação de um conjunto de dados não separáveis no espaço de entradas e provamos o efeito da alteração do parâmetro de penalização C e das diferentes opções de *kernel* nos resultados da classificação.

Na tarefa 3 definimos uma série de parâmetros, o parâmetro de penalização C e do *kernel*,

**Bibliografia**

Mitchell, pg. 154 – 158, 177 – 184, 230 – 236

Witten, secção 7.1, pg. 307 – 314