IMPLEMENTAÇÃO DO FREx_SVM: MÁQUINAS DE VETOR SUPORTE PARA CLASSIFICAÇÃO EM MÚLTIPLAS CLASSES

Aluno: Aarão Irving Manhães Marins
Orientador: Marley Maria Bernardes Rebuzzi Vellasco

Introdução

A máquina de vetor suporte (SVM) é uma técnica de aprendizado estatístico, baseada no princípio da Minimização do Risco Estrutural (SRM), e pode ser usada para resolver problemas de classificação [1].

Esta técnica já foi utilizada com sucesso em diversas aplicações de reconhecimento de padrões, como por exemplo: categorização de textos, categorização de SPAM, reconhecimento de caracteres manuscritos, reconhecimento de textura, análise de expressões de genes, etc.

O funcionamento das máquinas de vetor suporte é definido da seguinte maneira: dadas duas classes e um conjunto de pontos que pertencem a estas, as máquinas de vetor suporte determinam o hiperplano que os separa, de forma a colocar a maior quantidade possível de pontos da mesma classe do mesmo lado, ao mesmo tempo que a distância de cada classe a esse hiperplano é maximizada.

Normalmente, as máquinas de vetor suporte operam em um espaço de dimensão maior que a dimensão dos dados originais, no chamado espaço de características. E é neste espaço que se consegue a maximização desejada para obter uma melhor capacidade de generalização de classificação.

Para conseguir essa maximização, é necessária uma formulação para resolver problemas de otimização quadrática com restrições. Mais detalhes sobre esta etapa estão na seção "Classificação Binária".

Existem duas abordagens básicas de estender o problema de classificação binária para o problema de classificação em múltiplas classes. Uma é transformar o problema de múltiplas classes em vários de duas classes, e a outra é uma generalização do algoritmo de classificação binária usando máquinas de vetor suporte para que funcione para mais de duas classes.

No software implementado, foram usados somente dois métodos da abordagem que propõe a divisão em vários problemas binários. Eles são: "Decomposição um por classe" e "Separação de classes duas a duas", e são detalhadas na seção "Classificação em Múltiplas Classes" [2].

O uso de máquinas de vetor suporte é capaz de resolver problemas de classificação de dados, gerando classificadores que apresentam bons resultados. Porém, esses classificadores possuem uma limitação de interpretabilidade, não sendo possível compreender a saída obtida.

Uma forma de solucionar este problema de falta de transparência é desenvolver uma técnica de extração de regras a partir das máquinas de vetor suporte. Então o modelo FREx_SVM [3], para extração de regras fuzzy, foi escolhido para ser implementado.

Uma vez que se têm os resultados das máquinas de vetor suporte e as informações dos dados que definem o classificador gerado, os conjuntos fuzzy são construídos, e então se inicia o processo de geração de regras.

A definição de conjuntos fuzzy permite um conceito de pertinência, onde um elemento pode pertencer parcialmente a um conjunto, com um grau de pertinência com infinitos valores no intervalo [0,1]. Esses graus de pertinência em geral são definidos por funções simples, que por sua vez são chamadas de funções de pertinência.

As funções de pertinência são definidas através dos dados informados, e também dependem do número de conjuntos desejados. Desta forma é possível garantir que os conjuntos fuzzy associados a estas funções representem uma boa divisão dos dados.

Visando melhorar a interpretabilidade das regras geradas, foi proposta a implementação de um método de seleção de atributos que usa os resultados de um treinamento das máquinas de vetor suporte. Esse método consegue organizar os atributos por importância, permitindo então ao usuário refazer o treinamento com menos atributos, acelerando a capacidade de processamento, e melhorando a generalização da classificação.

Etapas

A. Arquivo de Dados

Organização

Para solucionar o problema de classificação binária de um conjunto de dados, é necessário escolher um subconjunto para se fazer o treinamento. Além disso, são criados outros conjuntos para testar a solução encontrada no treinamento.

A organização dos dados nos arquivos é bem simples. Os dados do conjunto são separados por linha no arquivo de dados. Cada um desses dados possui diversos atributos, e estão relacionados a uma classe, e estes são separados por vírgula no arquivo.

Normalmente a ordem das informações de cada dado é representada primeiramente com os atributos, e com a classe no final. Porém, como existem arquivos com uma ordem inversa (classe no início), existe uma opção para ler os arquivos neste formato.

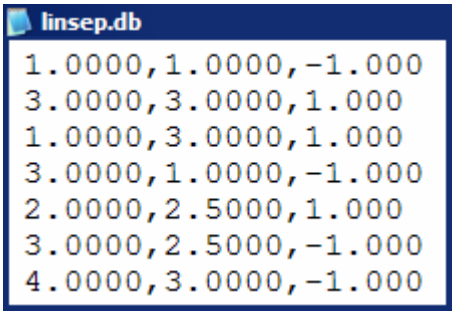

Figura 1. Modelo de organização de um arquivo de dados

Normalização

Os valores dos atributos dos arquivos de dados geralmente são informações empíricas. Logo, os domínios podem variar muito de um atributo para outro, o que torna pior a solução da classificação.

Para obter um resultado melhor, foi necessário normalizar o conjunto de dados de treinamento. Esta normalização foi feita logo após a abertura do arquivo, armazenando os valores máximos e mínimos de cada atributo de todos os dados, usando-os para ter todos os valores dos atributos entre 0 e 1. Assim, pode-se evitar a perda de generalização durante a resolução do problema de minimização gerado.

B. Classificação Binária

Definição

A classificação binária usando máquinas de vetor suporte pode ser dividida em duas etapas: treinamento e teste. Na fase de treinamento, o objetivo é criar um classificador que funcione bem para qualquer amostra de um conjunto de dados. Já a fase de teste é quando são

usadas amostras de dados que não foram usadas na obtenção do classificador, para verificar a generalização do mesmo.

Considerando o exemplo da Figura 2, pode-se ver que existem diversas possibilidades de classificadores lineares que separam as classes azul e laranja, mas somente um deles maximiza a margem (a distância entre o classificador e a amostra mais próxima de cada classe). Este classificador linear é chamado "hiperplano de separação ótimo" (indicado em verde na Figura 2). Intuitivamente, é esperado que este classificador faça uma melhor generalização do que os outros.

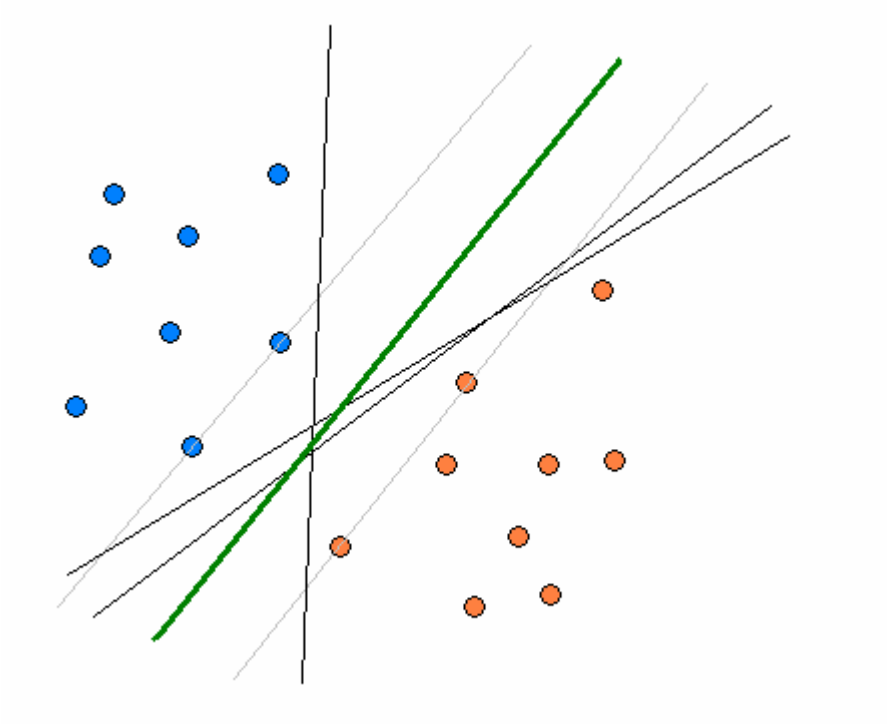

Figura 2. Hiperplano de separação ótimo

Porém esta abordagem é restrita aos casos em que os dados de treinamento são linearmente separáveis. Logo, para casos em que os dados não são linearmente separáveis, existe uma solução, que consiste em mapear os dados para um espaço de dimensão maior (espaço de características).

Utilizando uma função Kernel [1], é possível fazer o mapeamento no espaço de características, então as máquinas de vetor suporte constroem um hiperplano de separação ótimo neste espaço. Na construção do hiperplano ótimo, são obtidos os multiplicadores de Lagrange (α), que definem os vetores suporte.

A idéia da função Kernel é realizar operações sobre os dados de entrada, sem que estes sejam mapeados no espaço de características. Assim, não é necessário realizar operações em um espaço de dimensão maior, o espaço de características. Porém, o esforço computacional continua dependendo do número de dados usados no treinamento.

Os vetores suporte são os dados da amostra de treinamento que definem as margens do hiperplano de separação ótimo. Além dos vetores suporte, é necessário calcular o termo de bias, usado no ajuste do classificador. Mas se a função Kernel escolhida já contém um termo de bias, este não precisa ser calculado.

Implementação

A implementação do método de máquinas de vetor suporte para classificação binária teve como base o algoritmo proposto por Steve Gunn [4].

O aplicativo foi implementado na linguagem de programação C# [5], e também foram utilizados alguns recursos do MATLAB.

Foi criada uma estrutura para armazenar todos os dados, tanto os de entrada quanto os de saída, assim como os métodos a serem utilizados. Também foi necessário adicionar ao aplicativo, uma biblioteca para poder fazer comunicação com o MATLAB.

Considerando que os dados e suas respectivas classes já foram informados pelo usuário, e que esses dados já foram normalizados, o primeiro passo da implementação do método de treinamento foi armazenar a constante de regularização, a função Kernel, e seus possíveis parâmetros.

Uma vez que todas as informações necessárias estão armazenadas no aplicativo, é chamado então um módulo para realizar o treinamento. A primeira etapa deste método é criar uma tolerância para a detecção dos vetores suporte. Para encontrar o valor desta tolerância, é usada a constante de regularização. Também neste momento é feita uma alteração no valor das duas classes, somente durante este método para "1" e "-1", garantindo que o método funcione corretamente.

O passo seguinte do módulo é a contrução da matriz de Kernel, e para isso, é necessário utilizar uma função Kernel. Os elementos desta matriz são definidos utilizando as classes e uma das funções Kernel aplicada nos dados.

Foram definidas três funções Kernel para este aplicativo: Linear, RBF e Polinomial. Sendo que tanto a função RBF quanto a Polinomial precisam de um parâmetro adicional (sigma para RBF e grau para Polinomial). Exemplos dos Kernels para problemas bidimensionais são mostrados na Figura 3.

Linear	RBF	Polinomial
$K(x,y) = (x \cdot y)$	$K(x,y) = \exp(-(x - y)^2 / 2\sigma^2)$	$K(x,y) = (x \cdot y + 1)^d$

Tabela 1. Funções Kernel usadas no aplicativo

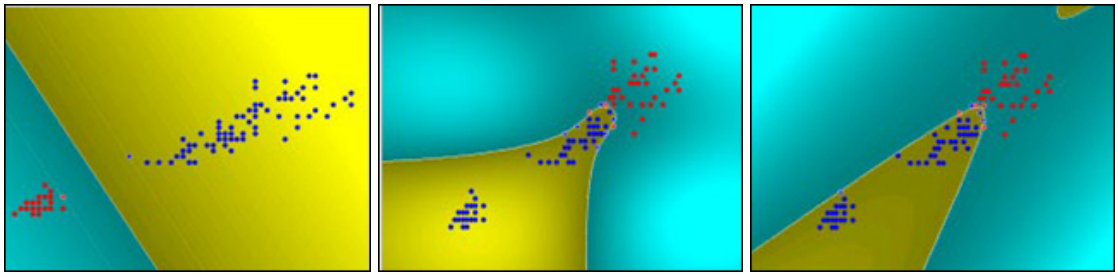

Figura 3. Exemplos de classificadores binários usando diferentes funções Kernel (Na ordem: Linear, RBF, Polinomial)

São adicionados então valores positivos bem pequenos na diagonal da matriz de Kernel, para que esta não fique má condicionada para a resolução do problema de otimização. São ajustados também os outros parâmetros necessários para resolver o problema, como os limites superiores, e inferiores (definidos pela constante de regularização) e o ponto inicial ([0,0,...,0]).

Para resolver o problema de otimização, foi usada uma solução para problemas de programação quadrática do MATLAB (quadprog). Para isto, foi necessário importar uma biblioteca que fizesse a comunicação do aplicativo com a janela de comandos do MATLAB, enviando linhas de comando e recebendo os resultados de volta para o aplicativo.

A matriz de Kernel, assim como todos os outros parâmetros necessários para chamar a função quadprog, são passados para o MATLAB. Então esta função é chamada, para calcular os multiplicadores de Lagrange, que são necessários para identificar os vetores suporte.

Solucionando o problema de otimização, identificam-se quais dados de entrada são os vetores suporte para o classificador. Isso é feito, verificando se os valores dos multiplicadores de Lagrange são maiores do que a tolerância estabelecida.

Utilizando os vetores suporte que estão dentro da margem, é então calculado o termo de bias, quando necessário. Assim o método de treinamento para classificação binária é finalizado.

Uma vez que o treinamento esteja encerrado, inicia-se a parte de teste para a classificação binária. E, neste método, o usuário deve informar um novo arquivo de dados, porém o tipo de informações deve ser do mesmo tipo que as do arquivo aberto inicialmente.

O modelo de testes utiliza os vetores suporte e suas respectivas classes (mais uma vez alteradas para "1" e "-1"), os multiplicadores de Lagrange, a função kernel utilizada no treinamento, o termo de bias, assim como os dados deste novo arquivo de entrada. Com isto é possível usar o classificador de treinamento, encontrando os valores para função de decisão. Usando os sinais desses valores, é que são definidas as classes de cada um dos dados de teste.

Finalmente, é feita uma comparação das classes obtidas pelo classificador, com as classes reais (disponíveis no arquivo de teste). Então o aplicativo retorna a quantidade de acertos obtidos no teste, para cada classe.

Resultados e Discussão

Antes da implementação do aplicativo, foram feitos diversos testes com a caixa de ferramentas para máquinas de vetor suporte disponibilizada por Steve Gunn. Estes testes apresentaram resultados muito parecidos com os encontrados no aplicativo implementado, pois este foi baseado no método de classificação proposto nesta caixa de ferramentas.

Para o treinamento e teste das máquinas de vetor suporte para classificação binária, foram usados dois bancos de dados: bancos de dados: "Bupa Liver Disorders" e "Wisconsin Breast Cancer", disponíveis em http://mlearn.ics.uci.edu/databases/. Os detalhes destes bancos de dados podem ser vistos na Tabela 2.

Banco de dados	Atributos	Dados para treinamento	Dados para teste
Bupa Liver Disorders	6	82	345
Wisconsin Breast Cancer	9	77	683

Tabela 2. Detalhes dos bancos de dados com duas classes

Os métodos de treinamento e teste foram avaliados com diferentes valores para a constante de regularização, e diferentes opções de função Kernel e seu parâmetro (exceto para a função Linear).

Foram usados cinco kernels diferentes para fazer o treinamento e teste dos bancos de dados: Linear, RBF com $\sigma^2=1$ e $\sigma^2=5$, e polinomial com graus 2 e 3. Para cada um dos kernels foram utilizados três valores para a constante de regularização: 0.1, 1 e 10. Os resultados dos treinamentos e testes realizados no bando de dados "Wisconsin Breast Cancer" seguem na Tabela 3.

Banco de dados: breast-cancer-wisconsin				
Função Kernel	Parâmetro	C	Vetores Suporte	Taxa de Acerto
Linear	-	0.1	34	0.9619327
		1	16	0.9677892
		10	12	0.9560761
RBF	1	0.1	51	0.9648609
		1	25	0.9707174
		10	19	0.9619327
	5	0.1	61	0.942899
		1	27	0.9590044
		10	16	0.9692533
Polinomial	2	0.1	19	0.9370425
		1	15	0.9575403
		10	15	0.9502196
	3	0.1	15	0.9326501
		1	15	0.9458272
		10	14	0.9355783

Tabela 3. Resultados do banco de dados: "Wisconsin Breast Cancer"

Os resultados obtidos nos testes neste banco de dados foram muito bons. Sendo que desses resultados, os melhores foram os que utilizaram o kernel linear e o kernel RBF com $\sigma^2=1$. Como já foi dito anteriormente, os resultados foram bem próximos com os obtidos na caixa de ferramentas usada no MATLAB.

C. Classificação em Múltiplas Classes

Definição

As máquinas de vetor suporte foram originalmente propostas para clasificação binária. Porém existem iniciativas para estender a metodologia para o caso de classificação em múltiplas classes [2].

Existem duas abordagens básicas para tratar do problema de classificação com múltiplas classes. Uma é a redução deste problema a vários problemas binários, já a outra abordagem propõe a generalização das máquinas de vetor suporte para que funcione com mais de duas classes.

Comparando essas duas abordagens, verifica-se que em geral, o custo computacional da abordagem de generalização é maior, pois um problema de otimização de dimensões elevadas precisa ser resolvido.

Considerando a abordagem em que o problema é dividido em um conjunto de problemas binários, existem diferentes métodos para se fazer a classificação em múltiplas classes. Um deles é chamado Decomposição um por classe ("One-Against-All"), e outro Separação das classes duas a duas ("One-Against-One").

Decomposição um por classe foi o primeiro método usado nas máquinas de vetor suporte para classificação em múltiplas classes. Seu funcionamento é bem simples e apresenta bons resultados. São construídos k classificadores binários, onde k é o número de classes. Para cada classificador, uma das classes é separada das outras.

Já no método Separação das classes duas a duas, são construídos k(k-1)/2 classificadores. Neste caso, há um número maior de classificadores do que no de

Decomposição um por classe, pois todas as classes se combinam. Porém o número de dados é menor em cada treinamento. Isto acontece porque cada treinamento utiliza duas das classes, logo somente os dados dessas classes são treinados.

Nos dois métodos, é necessário combinar os resultados dos treinamentos. Para isto, existem algumas maneiras diferentes para cada método. Para a Decomposição um por classe, a maneira mais simples de resolver este problema é fazer com que a classe final escolhida dentre os treinamentos seja a que tem maior valor na função de decisão.

Para combinar os resultados dos treinamentos no método de Separação de classes duas a duas, usa-se uma estratégia de votação de classes. Assim, quando um novo dado é testado, ele tem sua classe definida por aquela que recebeu a maior quantidade de votos.

Como mostram alguns experimentos, o método Decomposição um por classe é o mais conveniente para uso prático. Porém esse método tende a produzir poucos vetores suporte, por considerar todas as variáveis de uma só vez.

Metodologia

Para adaptar o aplicativo para funcionar com máquinas de vetores suporte para múltiplas classes, foram adicionados os dois métodos que dividem este problema em vários problemas de classificação binária, já citados anteriomente. Também foi necessário fazer algumas modificações na estrutura do aplicativo, para armazenar resultados de vários treinamentos.

No módulo de treinamento do aplicativo, foi criada uma condição, dependendo do tipo de classificação em múltiplas classes escolhidas pelo usuário. Quando a opção de Decomposição um por classe é escolhida, são realizados os treinamentos para cada classe, onde os dados de uma classe são definidos com uma classe "1" enquanto os outros são definidos como de uma classe "-1". Já quando a opção de Separação das classes duas a duas é escolhida, são construídos treinamentos para cada combinação de duas classes.

O módulo de teste das máquinas de vetor suporte em múltiplas classes também depende do tipo de classificação escolhido. Para isso foram implementadas maneiras de decidir a classe final de cada padrão. No caso da Decomposição um por classes, usa-se a função de decisão obtida no módulo de teste.

Já na Separação de classes duas a duas, foi implementada uma técnica de votação. Que como citada anteriormente, funciona adicionando votos para cada uma das classes que são escolhidas por cada classificador. Porém ainda assim, pode haver alguns pontos que não sejam classificados.

Resultados e Discussão

Assim como na classificação binária, os métodos implementados para classificação em múltiplas classes no aplicativo foram testados utilizando bancos de dados com mais de duas classes. E foram usados dois bancos de dados para estes testes: "Íris" e "Wine". Mais detalhes sobre esse bancos de dados estão na Tabela 4.

Banco de dados	Classes	Atributos	Dados para treinamento	Dados para teste
Íris	3	4	67	150
Wine	3	13	87	155

Tabela 4. Detalhes dos bancos de dados com mais de duas classes

Além de variar as mesmas opções que foram usadas nos testes de classificação binária, foi variado também o método de combinação dos vários treinamentos realizados na classificação em múltiplas classes.

Assim como para os testes dos métodos de classificação binária, foram usados dois kernels diferentes para fazer o treinamento e teste dos bancos de dados. Estes foram Linear e RBF, com quatro parâmetros diferentes: 1, 5, 10 e 50. Para cada um dos kernel foram utilizados dois valores para a constante de regularização: 1 e 10. Os resultados dos treinamentos e testes realizados no banco de dados "Wine", que possui três classes, usando Separação de classes duas a duas, podem ser vistos na Tabela 5 e os resultados usando Decomposição por classe, na Tabela 6.

Banco de dados: wine						
Função Kernel		C	Vetores Suporte			Taxa de Acerto (%)
Linear		1	20	10	14	95,48387
		10	8	8	7	97,41936
RBF	σ^2=1	1	24	12	21	94,19355
		10	15	10	10	94,83871
	σ^2=5	1	40	23	30	93,54839
		10	15	8	11	92,25807
	σ^2=10	1	51	34	44	95,48387
		10	20	11	14	92,90323
	σ^2=50	1	53	51	44	96,12903
		10	38	22	30	96,12903

Tabela 5. Resultados do banco de dados: "Wine" usando Separação das classes duas a duas

Banco de dados: wine						
Função Kernel		C	Vetores Suporte			Taxa de Acerto
Linear		1	21	27	13	95,48387
		10	9	13	7	72,90323
RBF	σ^2=1	1	27	32	21	92,25807
		10	14	17	12	92,90323
	σ^2=5	1	43	51	35	94,19355
		10	16	21	11	90,96774
	σ^2=10	1	52	58	50	94,83871
		10	21	26	13	92,90323
	σ^2=50	1	52	42	60	95,48387
		10	41	40	33	95,48387

Tabela 6. Resultados do banco de dados: "Wine" usando Decomposição um por classe

Os resultados obtidos em ambos os métodos de separação, em geral, foram bons. Sendo que na Separação de classes duas a duas, o melhor resultado foi usando kernel Linear com C=10. Já na Decomposição um por classe, os melhores resultados foram usando kernel Linear com C=1 e kernel RBF com σ^2=50.

D. Extração de Regras Fuzzy

Metodologia

O modelo de extração de regras fuzzy FREx_SVM, propõe a extração de regras de máquinas de vetor suporte, para classificação binária e em múltiplas classes. Em ambos os casos, após o treinamento, o processo de extração pode ser dividido em três etapas:

- Projeção dos vetores suporte.
- Construção dos conjuntos fuzzy.
- Geração de regras.

Primeiramente, para cada vetor suporte, são obtidas suas projeções, de forma que estas são iguais aos valores de cada atributo deste vetor. Sendo o número de projeções sempre igual ao número de atributos.

A construção dos conjuntos fuzzy é definida pela quantidade de conjuntos a serem criados e seus intervalos. Caso estes intervalos não sejam definidos, estes são escolhidos de forma que dividam igualmente o domínio dos atributos. Um exemplo de conjuntos fuzzy pode ser visto na Figura 4.

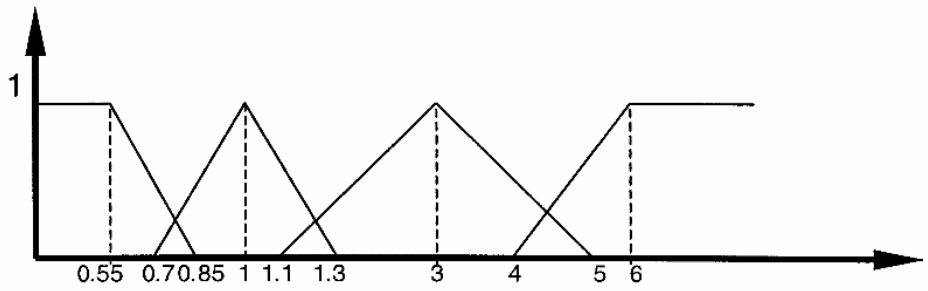

Figura 4. Exemplo de conjuntos fuzzy

A partir dessas informações são definidas as funções de pertinência, que caracterizam cada conjunto. Essas funções cobrem todo o espaço dos atributos e mapeiam os valores para graus de pertinência, que indicam quanto um determinado conceito faz parte de um conjunto. O grau de pertinência sempre varia de zero a um, por isso as funções que definem os conjuntos variam sempre entre zero e um [6,7,8].

Finalmente, para gerar as regras fuzzy, verificam-se, para todos os vetores suporte, quais conjuntos têm os maiores graus de pertinência para cada atributo. Então as regras são definidas exatamente como o grupo de conjuntos que cada vetor suporte gerou. Sendo o número de regras geradas igual ao de vetores suporte.

Cada regra é associada à classe do vetor suporte que a gerou. Porém, é possível que dois ou mais vetores suporte gerem a mesma regra. Neste caso, verifica-se se todos os vetores suporte são da mesma classe, e caso sejam, a regra é associada a esta classe, caso contrário, deve-se medir a acurácia fuzzy da regra para decidir a classe a que a regra estará associada.

A acurácia fuzzy [3] mede a capacidade de uma regra fuzzy de descrever os dados. Ela é medida através dos graus de pertinência associados a cada classe, em relação à soma total dos graus de pertinência.

Além disso, também é feito o cálculo da abrangência [3] fuzzy, que mede o número de padrões afetados pela regra. Para calcular seus valores usou-se a soma de todos os graus de pertinência de cada vetor suporte, em relação número total de padrões.

Uma vez obtidas as regras, estas são gravadas em um arquivo de regras (*.fr), onde cada regra gerada fica separada por linha. Um exemplo de arquivo de regras pode ser visto na Figura 5.

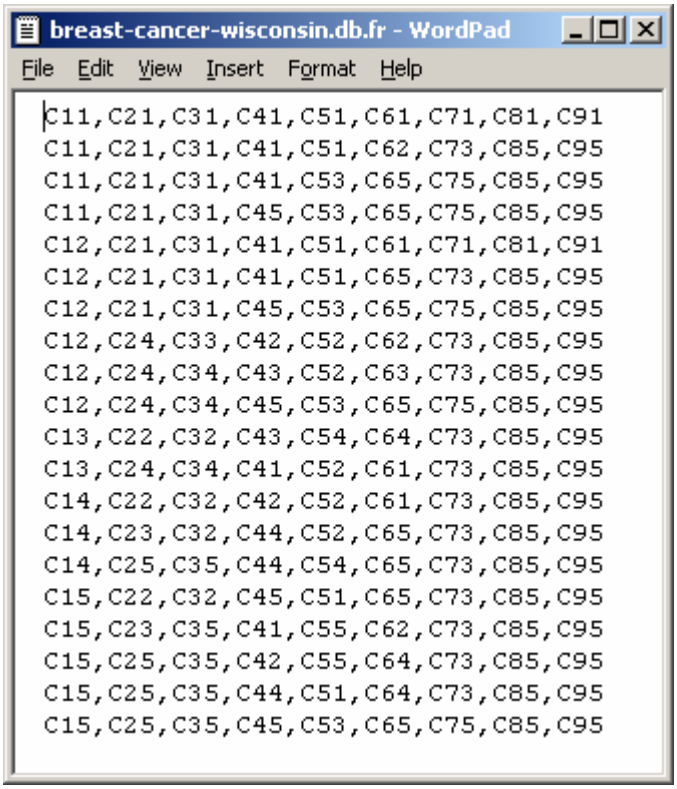

Figura 5. Exemplo de arquivo de regras

Cada uma das regras possui um número de conjuntos relativos ao número de atributos. Cada um desses conjuntos é representado por “Cxy”, sendo x o índice do atributo e y o índice do intervalo que a regra funciona para cada atributo.

Para realizar a extração de regras no método de classificação em múltiplas classes, foram adicionadas ao aplicativo duas funções. A primeira foi para o método de Separação das classes duas a duas (“One-Against-One”), onde os vetores suporte de todos os treinamentos são enviados juntos para extração das regras. A outra foi para o método de Decomposição um por classe (“One-Against-All”), onde somente os vetores suporte da classe principal de cada treinamento são considerados.

Resultados e Discussão

Para avaliar os resultados do módulo de extração de regras fuzzy implementado, foram analisados diversos bancos de dados. Para cada avaliação, foram realizados os treinamentos para obtenção dos vetores suporte necessários.

Foi notado que a escolha de muitos conjuntos fuzzy gera regras demais, além de provocar um enorme esforço computacional. Por isso, os testes, em sua maioria, foram realizados com três conjuntos.

Os resultados de alguns dos testes foram bons, apresentando uma cobertura aceitável das regras geradas no módulo de extração. Porém, em geral, os resultados obtidos não foram satisfatórios, pois na maior parte dos testes, as regras obtidas não foram classificadas.

E. Seleção de Atributos

Metodologia

A técnica de seleção de atributos é baseada na construção das máquinas de vetor suporte e na função de decisão obtida através do treinamento das mesmas. Uma de suas pricipais importâncias é aumentar a interpretabilidade das regras fuzzy geradas pelo aplicativo.

Além de ajudar na interpretação das regras, a técnica de seleção de atributos também colabora muito para a eficiência do aplicativo. Pois, dependendo da quantidade de dados, podem diminir bastante a quantidade de informações a serem processadas pelo computador.

Uma vez que um treinamento já tenha sido feito, o usuário pode escolher a opção "Feature Selection", que retorna o grau de importância (que variam de 0 a 1) de cada atributo, assim como a ordem de importância dos mesmos (do mais importante para o menos importate).

Para continuar o método de seleção de atributos, uma forma que parece ser muito eficiente, seria refazendo o treinamento com uma quantidade menor de atributos. Utilizando desta forma, somente os atributos mais importantes. Porém esta funcionalidade não pôde ser implementada.

Contudo, os resultados da opção "Feature Selection" ainda podem ser úteis. Pois o usuário pode descobrir a ordem de importância dos atributos de um banco de dados usando esta opção. Então pode criar um outro arquivo de dados, retirando os atributos menos importantes, e então, fazendo o treinamento nesse novo banco de dados.

F. Interface

Recursos

Para interagir com o usuário, foi escolhida uma interface gráfica com componentes visuais do Windows, para entrada dos dados para o treinamento das máquinas de vetor suporte, e para exibição de resultados obtidos.

Inicialmente, o usuário deve abrir um arquivo de dados (*.db), então habilitará as opções de treinamento, e será informado da quantidade de dados no arquivo, assim como a quantidade de atributos. Essas opções são: a escolha da função Kernel a ser utilizada, a definição da constante de regularização e, para o caso de múltiplas classes, o método de treinamento.

Existem três opções de funções Kernel, onde a padrão é a função Linear. Outras funções que podem ser escolhidas são a função RBF e a função Polinomial, ambas necessitam de um parâmetro adicional (σ^2 para função RBF e grau para função Polinomial). Esses parâmetros são requisitados ao usuário no momento em que a opção da função Kernel é selecionada.

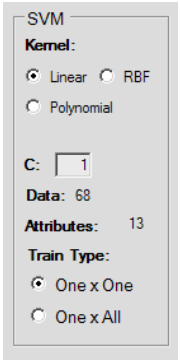

Figura 6. Exemplo de treinamento com opções para função Kernel Linear

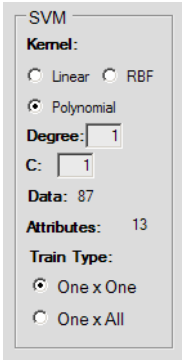

Figura 7. Exemplo de treinamento com opções para função Kernel Polinomial

Uma vez que o usuário define todas as informações necessárias, pode iniciar o treinamento das máquinas de vetor suporte. No fim do treinamento, são exibidos o número de vetores suporte obtidos e o termo de bias. Para o caso de múltiplas classes, são exibidos os resultados de cada um dos treinamentos.

Definidos os vetores suporte, o usuário pode realizar testes com outros arquivos, desde que os dados sejam do mesmo tipo dos dados que foram treinados. A escolha do arquivo de teste é feita no momento em que a opção de teste é acionada. É exibido, para cada classe, o número de acertos em relação ao número total de dados da respectiva classe.

Também usando os resultados do treinamento, a opção para extração de regras fuzzy pode ser ativada. Para execução do módulo de extração de regras, o usuário precisa escolher em quantos conjuntos fuzzy as informações dos atributos serão divididas. As opções disponíveis são três, cinco ou sete conjuntos. Também é possível definir os intervalos desses conjuntos.

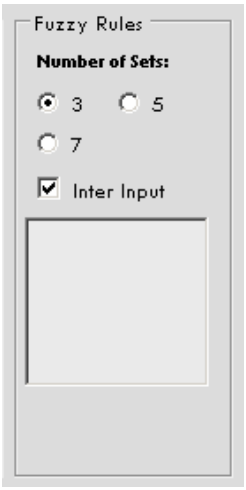

Figura 8. Caixa de configurações para chamada do módulo de extração de regras

O método de seleção de atributos, só pode ser ativado após o treinamento, pois também utiliza os vetores suporte obtidos. Nos resultados da seleção de atributos são exibidas as importâncias de cada atributo e a ordem decrescente dos atributos em relação às importâncias.

O programa possui outras opções, como Reset, que limpa todas as informações da tela e da memória, e também, como já mencionado, existe uma opção para leitura de arquivo de dados que possuem a classe no início de cada linha.

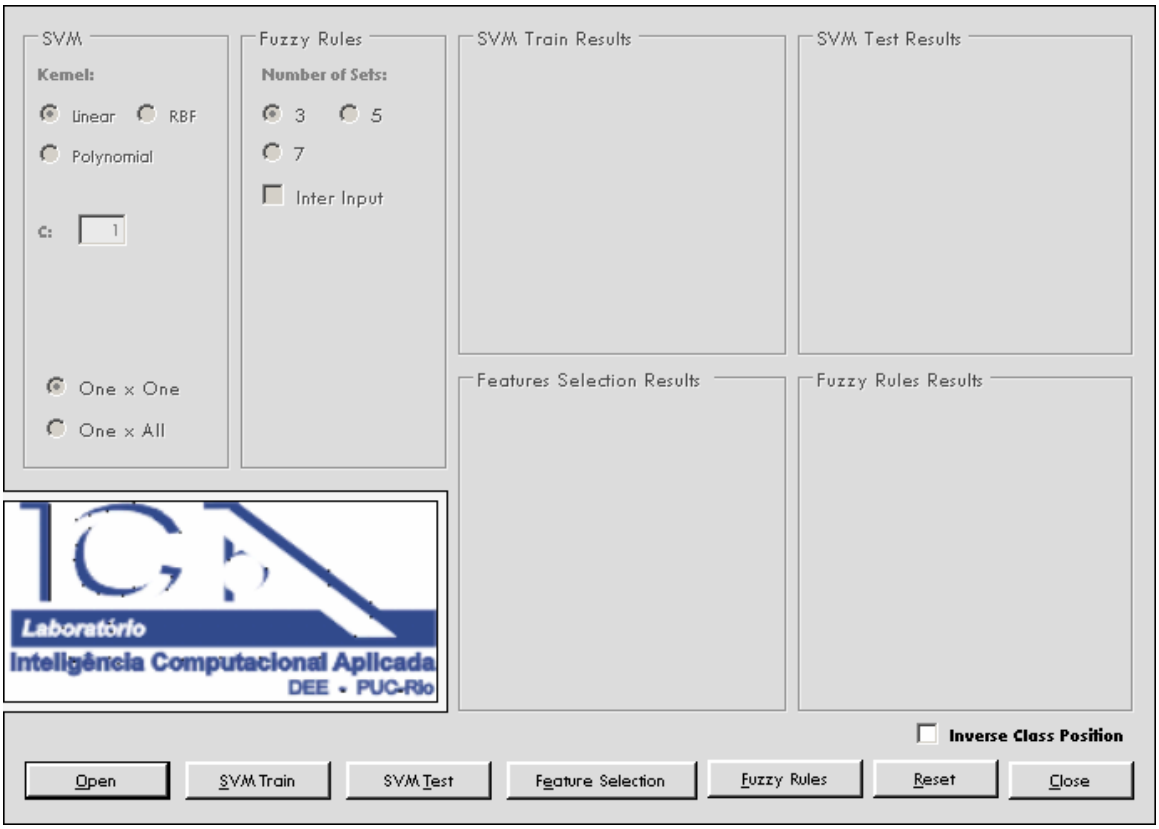

Figura 9. Interface gráfica do aplicativo

Conclusões

As pesquisas realizadas sobre diversas formas de implementação de máquinas de vetor suporte para classificação, assim como o funcionamento das regras fuzzy, foram muito importante para o desenvolvimento do aplicativo. Além disso, foi fundamental estudar o modelo de extração utilizado. Assim, a forma de implementação foi escolhida de modo a obter resultados mais próximos das especificações deste modelo.

O método de treinamento dos dados tem um grande custo computacional, sendo sempre melhor não utilizar conjuntos muito grandes de dados para esta fase do programa. Já no método de teste, o tamanho do conjunto de teste quase não requer esforço, possibilitando assim testar a eficiência dos treinamentos em grandes conjuntos de dados.

Já no módulo de extração de regras, o peso computacional está relacionando ao número de conjuntos escolhidos pelo usuário e ao número de atributos dos dados de entrada. Uma vez que o número total de conjuntos formados é c^p, onde "p" o número de atributos e "c" é o número de conjuntos para cada atributo.

A implementação, com o uso das máquinas de vetor suporte, permite ao aplicativo poder ser usado tanto para problemas de classificação, binária e em múltiplas classes, quanto para a extração de regras fuzzy.

Referências

1 - CRISTIANINI, N., SHAWE-TAYLOR J., **An introduction to support Vector Machines: and other kernel-based learning methods**. New York: Cambridge University Press, 1999. 189p.

2 - HSU, C. W., LIN, C. J., A comparison of methods for multiclass Support Vector Machines. **IEEE Trans on Neural Networks**, v.13, n.2, p. 415-425. 2002.

3 - CHAVES, A. C. F. **Extração de Regras Fuzzy para Máquinas de Vetores Suporte (SVM) para Classificação em Múltiplas Classes**. Rio de Janeiro, 2006. 225p. Tese de Doutorado - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

4 - GUNN, S. Support Vector Machines for Classification and Regression. **ISIS Technical Report**. 1998.

5 - DEITEL, Harvey M., DEITEL Paul J., **Visual C# 2005: How to Program.** 2 ed. Pearson Prentice Hall, 2005. 1591p.

6 - ZADEH, L. A., Fuzzy Sets, **Information and Control,** v. 8, p. 338-353, 1965.

7 - MAMDANI, E. H. Application of fuzzy logic to approximate reasoning, **IEEE Trans. Computing.** v. 26, p. 1182-1191, 1977.

8 - ZIMMERMANN, H. J. **Fuzzy Set Theory and its Applications**, 2nd Ed, Kluwer Academic Publishers, Boston, 1990.