UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CENTRO ESTADUAL DE PESQUISAS EM SENSORIAMENTO REMOTO E METEOROLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM SENSORIAMENTO REMOTO

SUPPORT VECTOR MACHINES NA CLASSIFICAÇÃO DE IMAGENS HIPERESPECTRAIS

por
RAFAELA ANDREOLA

Orientador: Prof. Vitor Haertel, PhD

Porto Alegre, agosto de 2009.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CENTRO ESTADUAL DE PESQUISAS EM SENSORIAMENTO REMOTO E METEOROLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM SENSORIAMENTO REMOTO

SUPPORT VECTOR MACHINES NA CLASSIFICAÇÃO DE IMAGENS HIPERESPECTRAIS

por

Rafaela Andreola
Engenheira de Computação (2006) – FURG

Orientador: Prof Vitor Haertel, PhD

Área de concentração: Desenvolvimento de Novas Metodologias.

Banca Examinadora: Prof. Dr. João Luiz Dihl Comba - UFRGS
Prof. Dr. Glauber Acunha Gonçalves - FURG
Prof. Dr. Luciano Vieira Dutra - INPE

Dissertação submetida ao Programa de Pós-Graduação em Sensoriamento Remoto do Centro Estadual de Pesquisas em Sensoriamento Remoto e Metereologia – UFRGS, como requisito parcial para a obtenção do grau de Mestre em Senroriamento Remoto.

Porto Alegre, agosto de 2009

The mind, once expanded to the dimensions of larger ideas, never returns to its original size.

Oliver Holmes

AGRADECIMENTOS

Aos meus amigos e família pelo apoio, paciência e compreensão em todos os momentos. Em especial aos meus pais, meus primeiros grandes mestres, pelo encorajamento;

Aos amigos, colegas, funcionários e professores do Centro Estadual de Pesquisas em Sensoriamento Remoto e Meteorologia (CEPSRM) pelo apoio e auxílio. Especialmente ao Prof. Vitor Haertel, exemplo de competência e dedicação à ciência, pela orientação.

Ao pesquisador Elad Yom Tov, membro do IBM Haifa Research Lab, pelo auxílio nos primeiros passos da implementação do algoritmo SVM;

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro recebido.

À Universidade Federal do Rio Grande do Sul (UFRGS) e a todos que, de alguma maneira, contribuíram com o desenvolvimento desta dissertação.

SUPPORT VECTOR MACHINES NA CLASSIFICAÇÃO DE IMAGENS HIPERESPECTRAIS

RAFAELA ANDREOLA
Orientador: Prof. Vitor Haertel, PhD

RESUMO

É de conhecimento geral que, em alguns casos, as classes são espectralmente muito similares e que não é possível separá-las usando dados convencionais em baixa dimensionalidade. Entretanto, estas classes podem ser separáveis com um alto grau de acurácia em espaço de alta dimensão. Por outro lado, classificação de dados em alta dimensionalidade pode se tornar um problema para classificadores paramétricos, como o Máxima Verossimilhança Gaussiana (MVG). Um grande número de variáveis que caracteriza as imagens hiperespectrais resulta em um grande número de parâmetros a serem estimados e, geralmente, tem-se um número limitado de amostras de treinamento disponíveis. Essa condição causa o fenômeno de Hughes que consiste na gradual degradação da acurácia com o aumento da dimensionalidade dos dados. Neste contexto, desperta o interesse a utilização de classificadores não-paramétricos, como é o caso de *Support Vector Machines* (SVM). Nesta dissertação é analisado o desempenho do classificador SVM quando aplicado a imagens hiperespectrais de sensoriamento remoto. Inicialmente os conceitos teóricos referentes à SVM são revisados e discutidos. Em seguida, uma série de experimentos usando dados AVIRIS são realizados usando diferentes configurações para o classificador. Os dados cobrem uma área de teste da Purdue University e apresenta classes de culturas agrícolas espectralmente muito similares. A acurácia produzida na classificação por diferentes kernels são investigadas em função da dimensionalidade dos dados e comparadas com as obtidas com o classificador MVG. Como SVM é aplicado a um par de classes por vez, desenvolveu-se um classificador multi-estágio estruturado em forma de árvore binária para lidar como problema multi-classe. Em cada nó, a seleção do par de classes mais separáveis é feita pelo critério distância de Bhattacharyya. Tais classes darão origem aos nós descendentes e serão responsáveis por definir a função de decisão SVM. Repete-se este procedimento em todos os nós da árvore, até que reste apenas uma classe por nó, nos chamados nós terminais. Os softwares necessários foram desenvolvidos em ambiente MATLAB e são apresentados na dissertação. Os resultados obtidos nos experimentos permitem concluir que SVM é uma abordagem alternativa válida e eficaz para classificação de imagens hiperespectrais de sensoriamento remoto.

Palavras chaves: Support Vector Machines, classificador em árvore binária, sensoriamento remoto, Máxima Verossimilhança Gaussiana, imagens hiperespectrais.

HYPERSPECTRAL IMAGE CLASSIFICATION WITH SUPPORT VECTOR MACHINES

RAFAELA ANDREOLA
Orientador: Prof. Vitor Haertel, PhD

ABSTRACT

This dissertation deals with the application of Support Vector Machines (SVM) to the classification of remote sensing high-dimensional image data. It is well known that in many cases classes that are spectrally very similar and thus not separable when using the more conventional low-dimensional data, can nevertheless be separated with an high degree of accuracy in high dimensional spaces. Classification of high-dimensional image data can, however, become a challenging problem for parametric classifiers such as the well-known Gaussian Maximum Likelihood. A large number of variables produce an also large number of parameters to be estimated from a generally limited number of training samples. This condition causes the Hughes phenomenon which consists in a gradual degradation of the accuracy as the data dimensionality increases beyond a certain value. Non-parametric classifiers present the advantage of being less sensitive to this dimensionality problem. SVM has been receiving a great deal of attention from the international community as an efficient classifier. In this dissertation it is analyzed the performance of SVM when applied to remote sensing hyper-spectral image data. Initially the more theoretical concepts related to SVM are reviewed and discussed. Next, a series of experiments using AVIRIS image data are performed, using different configurations for the classifier. The data covers a test area established by Purdue University and presents a number of classes (agricultural fields) which are spectrally very similar to each other. The classification accuracy produced by different kernels is investigated as a function of the data dimensionality and compared with the one yielded by the well-known Gaussian Maximum Likelihood classifier. As SVM apply to a pair of classes at a time, a multi-stage classifier structured as a binary tree was developed to deal with the multi-class problem. The tree classifier is initially defined by selecting at each node the most separable pair of classes by using the Bhattacharyya distance as a criterion. These two classes will then be used to define the two descending nodes and the corresponding SVM decision function. This operation is performed at every node across the tree, until the terminal nodes are reached. The required software was developed in MATLAB environment and is also presented in this dissertation.

Key words: Gaussian Maximum Likelihood, binary tree classify, high-dimensional image data, remote sensing, SVM.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AVIRIS	*Airbone Visible Infrared Imaging Spectrometer*
CAB	Classificador em Árvore Binária
CDA	Classificador de Decisão em Árvore
CEM	Classificador em estágio-múltiplo
CEU	Classificador em estágio-único
EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
LANDSAT	*Land Remote Sensing Satellite*
LV	Limiar de Veressimilhança
MVG	Máxima Verossimilhança Gaussiana
RBF	*Radial Basis Function*
SFS	*Sequential Forward Selection*
SFFS	*Sequential Forward Floating Selection*
SPOT	*Systeme pour l'Observation de la Terre*
SRM	*Structural Risk Minimization*
SVM	*Support Vector Machines*
TM	*Thematic Mapper*

CAPÍTULO I

INTRODUÇÃO

1.1 JUSTIFICATIVA

Dados em alta dimensionalidade podem oferecer um poder discriminante bem mais elevado do que dados tradicionais em baixa dimensionalidade. FUKUNAGA (1990) demonstra que classes espectralmente muito semelhantes entre si (classes que compartilham vetores de médias muito próximas) podem ser separadas satisfatoriamente em espaços de dimensão mais altas. Esta é uma das motivações para o desenvolvimento de sistemas sensores com um número grande de bandas espectrais, conhecidos como sensores hiperespectrais.

Entretanto, uma das principais dificuldades que surgem no processo de classificação de dados em alta dimensionalidade por meio de classificadores paramétricos diz respeito ao número de amostras de treinamento (em geral limitado) em comparação com o número de parâmetros a serem estimados. Um número limitado de amostras de treinamento resulta em uma estimativa pouco confiável dos parâmetros em um classificador paramétrico como, por exemplo, o classificador de *Bayes* e, conseqüentemente, em um valor reduzido na acurácia da imagem temática produzida. Iniciando o processo de classificação com dados em dimensionalidade reduzida, a acurácia da imagem temática tende, inicialmente, a aumentar na medida em que informações (na forma de bandas espectrais) adicionais são incluídas. Em um determinado momento, a acurácia atinge um máximo para em seguida passar a diminuir, na medida em que a dimensionalidade dos dados continua a aumentar. Este problema, conhecido pela comunidade internacional como *fenômeno de Hughes*, vem sendo objeto de estudo por pesquisadores, como HOFFBECK & LANDGREBE (1996) e JIMENEZ & LANDGREBE (1999), por exemplo. Redução na dimensionalidade dos dados por meio de técnicas de extração ou seleção de variáveis (feature extraction/selection), introdução de mostras de treinamento semi-rotuladas, técnicas de análise discriminante regularizada, são abordagens que vem sendo investigadas com o objetivo de minimizar as conseqüências de tal fenômeno. Neste contexto, desperta o interesse a utilização de classificadores não-paramétricos, como é o caso

de SVM, que apresenta a vantagem de não ser afetado por este tipo de problema (HUANG et al., 2002).

No contexto da classificação de imagens hiperespectrais, algumas investigações experimentais apontam a eficácia do *Support Vector Machines* (SVM) para a análise desses dados. A utilização do classificador SVM apresenta, entretanto, algumas dificuldades. Possivelmente a mais óbvia resida no fato de este classificador ser aplicável diretamente a apenas um par de classes a cada vez (ABE, 2005).

Nesta dissertação investiga-se a utilização de SVM na classificação de imagens digitais hiperespectrais de sensoriamento remoto, nas quais várias classes estão presentes. Na metodologia proposta, investiga-se a implementação de SVM em um classificador em estágio múltiplo estruturado na forma de árvore binária. A estrutura binária em múltiplos estágios permite tratar pares de classes a cada estágio (nó) contornando, desta forma, a principal limitação apresentada pelo classificador. Uma vantagem adicional da estrutura em estágio múltiplo reside na possibilidade de otimização na escolha das variáveis (*features*) que conferem um maior poder discriminante entre o par de classes a cada nó individual da árvore binária. Esta possibilidade é também investigada nesta dissertação, implementando-se técnicas de seleção de variáveis a cada nó da árvore binária.

1.2 OBJETIVOS

Este estudo tem por objetivo investigar o uso do método de classificação *SVM* em imagens digitais hiperespectrais como aquelas obtidas por sistemas sensores a bordo de satélites para estudo da superfície da Terra.

Como objetivos específicos tem-se a implementação da função de decisão SVM em um classificador de decisão em árvore binária, implementando métodos para seleção de variáveis a cada nó da árvore. Dois kernels e seus respectivos parâmetros são testados com a finalidade de investigar a influência na acurácia da imagem temática produzida. Para validação da metodologia proposta, os resultados obtidos pelo classificador em questão serão confrontados com os obtidos pelo Classificador Máxima Verossimilhança Gaussiana (MVG) tradicionalmente usado para este fim.

1.3 RECONHECIMENTO DE PADRÕES

Reconhecimento de padrões tem como objetivo classificar objetos de interesse em uma dada classe ou categoria. Os objetos de interesse são genericamente denominados de padrões e podem ser caracteres, gráficos, células em biologia, sinais

eletrônicos ou qualquer outro objeto que se deseje classificar em uma classe, dentre as várias disponíveis.

Em imagens digitais, normalmente “padrão” vem a ser pixels individuais a serem atribuídos a uma das classes definidas pelo analista. Neste contexto, um pixel é representado por um vetor cuja dimensão é igual à dimensionalidade dos dados. No caso de imagens multiespectrais, a dimensionalidade é igual ao número de bandas espectrais adotadas.

Um modelo simples do sistema de reconhecimento de padrões consiste, basicamente, no receptor (a bordo de um avião ou satélite de observação) e no classificador. A saída do receptor é um conjunto de *n* medidas, cada uma correspondendo a um canal do sensor. O classificador rotula esse vetor de medidas em uma das pré-especificadas classes de acordo com uma regra de classificação apropriada (Figura 1) (LANDGREBE, 2003).

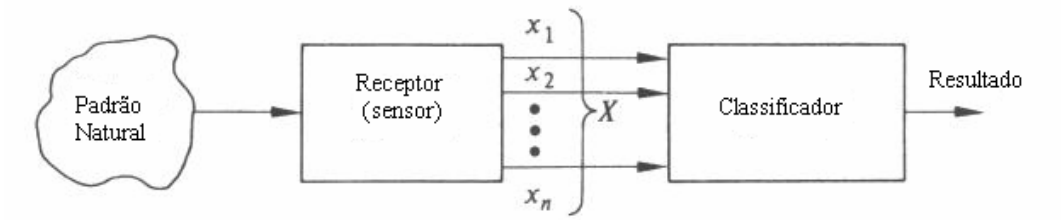


Figura 1: Modelo de Sistema de Reconhecimento de Padrões. Fonte: LANDGREBE (2003).

1.4 MÉTODOS DE CLASSIFICAÇÃO

Os métodos de classificação podem ser agrupados em duas grandes categorias (DUDA, 2000): métodos supervisionados e métodos não-supervisionados.

Nos métodos da classificação supervisionada, as classes são previamente definidas pelo analista, isto é, definidas ou caracterizadas através das amostras de treinamento. Cada classe pode, então, ser caracterizada por uma função decisão que pode ser de natureza probabilística (paramétricos) ou de natureza determinística (não-paramétricos), características estas que serão descritas em seguida.

Os métodos não-supervisionados oferecem um outro tipo de abordagem. Em alguns casos, existem problemas na área de reconhecimento de padrões, nas quais a natureza (ou definição) das classes, e mesmo o número de classes presentes são desconhecidos. Neste caso, o problema a ser tratado consiste não somente na classificação propriamente dita, mas também na própria definição das classes. Ao contrário do método supervisionado, onde se tem um conhecimento prévio das classes, os métodos não-supervisionados atribuem à técnica ou ao algoritmo

escolhido a tarefa de identificar as classes existentes num conjunto de dados. São exemplos de classificadores não-supervisionados o k-médias e o isodata.

Os processos de classificação podem também ser agrupados em outras duas grandes categorias (LANDGREBE, 2003): métodos paramétricos e métodos não-paramétricos.

Nos métodos paramétricos, se supõe conhecida a forma geral da função densidade de probabilidade que descreve o comportamento dos dados. Os parâmetros existentes são estimados a partir das amostras de treinamento disponíveis para cada classe. Quando a forma geral da função densidade probabilidade associada aos dados não é conhecida, a alternativa consiste na utilização dos chamados métodos não-paramétricos. Dentro de uma abordagem paramétrica, o método mais utilizado em sensoriamento remoto é o MVG, que é um caso particular do classificador de Bayes. Já redes neurais, sistemas Fuzzy, e SVM são exemplos de classificadores não-paramétricos. Para o treinamento desses classificadores usam-se pares entrada-saída, determinando a função de decisão que classificará os dados de entrada em uma das dadas classes.

Para SAVAVIAN & LANDGREBE (1991), existe ainda a possibilidade de dividir os classificadores em estágio-único (CEU's) e estágio-múltiplo (CEM's): CEU's são os classificadores onde cada amostra é testada contra todas as classes de uma só vez, e, a partir deste confronto entre as classes, a amostra já é rotulada. Por outro lado, CEM's é uma abordagem que busca quebrar decisões complexas em uma união de decisões mais simples. No caso de classificadores de decisão em árvore binária (exemplo de CEM's) cada amostra é direcionada a um dos nós descendentes, até chegar à um nó final onde será rotulada. MVG é usualmente dito como CEU's, mas pode ser implementado como CEM's.

1.5 DIMENSIONALIDADE DOS DADOS

O produto de um sistema multiespectral é um conjunto de imagens do mesmo objeto ou cena onde cada imagem é obtida em uma região distinta no espectro eletromagnético, isto é, em bandas espectrais próprias, todas elas registradas (pixels correspondentes nas diversas bandas espectrais cobrem exatamente a mesma região no terreno). Os dados espectrais obtidos pelos sensores multiespectrais possuem baixa dimensionalidade, como por exemplo, os sistemas Landsat e Spot, deixando descobertas extensas regiões do espectro (Figura 2).

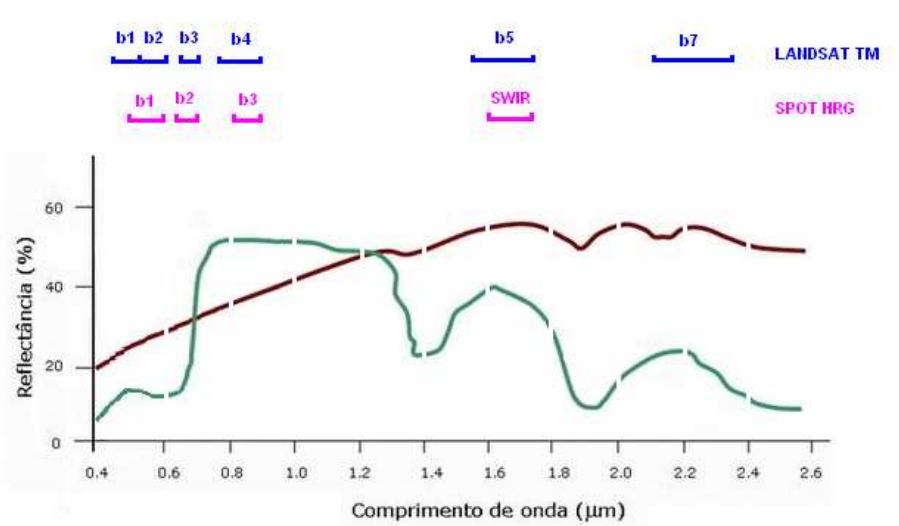


Figura 2: Áreas do espectro eletromagnético cobertas pelos satélites SPOT HRG e LANDSAT TM. Fonte: EMBRAPA MONITORAMENTO POR SATÉLITE.

LEWOTSKY (1994) relata que o espectro da reflectância da maioria dos materiais da superfície terrestre apresenta feições ou picos de absorção. O fato de novas tecnologias de sensores remotos serem capazes de adquirir o espectro detalhado de reflectância de uma determinada área (dados em alta dimensionalidade, designados como dados hiperespectrais) torna-a uma ferramenta poderosa para o estudo da Terra e do seu meio-ambiente. Esses sensores coletam dados em intervalos contíguos e estreitos no espectro eletromagnético, nas regiões do visível, infravermelho próximo e infravermelho médio com intervalos de comprimento de onda extremamente estreitos. Um exemplo consiste no sistema sensor aerotransportado AVIRIS (*Airbone Visible Infrared Imaging Spectrometer*), que fornece dados em 224 bandas espectrais na região 0.4 *µm* – 2.4 *µm* do espectro eletromagnético (LANDGREBE, 2003). Este sistema emprega a varredura do tipo *whiskbroom* (feita no sentido transversal à linha de vôo), com 224 detectores, cada um cobrindo um intervalo de comprimento de onda (banda espectral), de aproximadamente 10*nm* (Figura 3). O tamanho do pixel e a largura da faixa explorada dependem da altitude a partir do qual os dados são coletados. Quando coletados pelo ER-2 (20 km acima do solo) cada pixel produzido pelo instrumento abrange uma área de aproximadamente $20m^2$ no chão (com alguma sobreposição entre os pixels) produzindo, assim, uma faixa explorada terreno cerca de 11 quilômetros de largura. Quando recolhidos pelo Twin Otter (4 km acima do solo), cada pixel é terreno $4m^2$, e a faixa explorada passa a ser de aproximadamente 2 km de largura. A Tabela 1 mostra as principais características técnicas do sensor AVIRIS (AVIRIS) e na Tabela 2 são apresentados alguns dados comparativos entre os sensores AVIRIS e Landsat-TM.

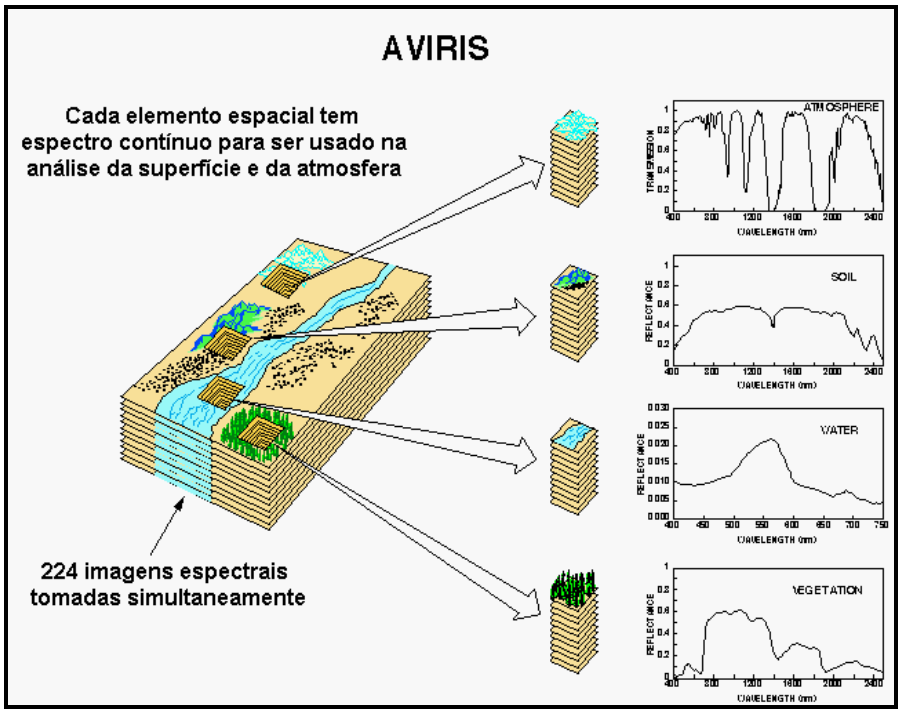


Figura 3: Concepção do sensor hiperespectral AVIRIS: a alta resolução espectral torna a informação de cada pixel próxima à obtida por meio de medições realizadas em laboratório e/ou campo. Fonte: AVIRIS.

TABELA 1- Características técnicas do sensor AVIRIS. Fonte: AVIRIS.

Taxa de dados em 1994, 17 Mbps, a partir de 1995 passou para 20.4 Mbps
Resolução Espectral em 1994, de 10 bits. A partir de 1995, passou para 12 bits.
Detectores para os λ's da faixa do visível da *Silicon* (Si), e detectores de *Indium-Antimonide* (InSb) para o infravermelho-próximo
Varredura do tipo *Whisk Broom*
Taxa de varredura de 12Hz
Canais de bandas nominais de 10 *nm*, calibradas para 1 *nm*
Campo de visada total de 34º (completo 677 amostras)
Campo de visada instantâneo de 1 mrad (IFOV, 1 amostra), calibrados para 0.1 mrad

TABELA 2– Comparação entre o sensor hiperespectral AVIRIS e o TM do LandSat. Fonte: AVIRIS

SENSOR	**FAIXA ESPECTRAL (NM)**	**Nº DE BANDAS ESPECTRAIS**	**RESOLUÇÃO ESPECTRAL (NM)**	**LARGURA DE FAIXA IMAGEADA (KM)**	**RESOLUÇÃO ESPACIAL (m^2)**
AVIRIS	400 a 2500	224	10	11	20
TM	450 a 12500	7	Variável	185	900 e 14400

1.6 ESTRUTURA DA DISSERTAÇÃO

No primeiro capítulo é feita uma introdução aos objetivos da dissertação, bem como uma rápida exposição sobre reconhecimento de padrões em imagens digitais e características dos dados multidimensionais.

No segundo capítulo é apresentada uma revisão bibliográfica sobre problemas associados ao processo de classificação de dados em alta dimensionalidade empregando classificadores paramétricos e algumas propostas que visam minimizar tais efeitos. São ainda discutidos o classificador MVG (que é, tradicionalmente, um classificador em estágio-único, mas pode ser implementado de forma a se tornar estágio-múltiplo) e o Classificador de Decisão em Árvore (um dos possíveis classificadores em estágio-múltiplo).

O terceiro capítulo, Metodologia, aborda o Classificador SVM e sua formulação matemática, assim como a proposta de algoritmo, desenvolvida em forma de árvore binária, empregado nesta dissertação.

No quarto capítulo, Testes e Experimentos, é apresentado os materiais e discutidos os resultados dos testes e experimentos realizados de acordo com o algoritmo proposto no terceiro capítulo.

O quinto capítulo são apresentadas as conclusões, análise final e sugestões.

O Apêndice A contém a listagem dos programas elaborados com o propósito de viabilizar a pesquisa. No apêndice B são apresentadas as tabelas de contingência dos resultados obtidos e discutidos no capítulo de experimentos.

CAPÍTULO II

REVISÃO BIBLIOGRÁFICA

2.1 FENÔMENO DE HUGHES

Em geral, um número pequeno de bandas que caracteriza os sensores multiespectrais tradicionalmente utilizados em sensoriamento remoto é suficiente para discriminar a maioria das classes que ocorrem em cenas naturais (florestas, culturas agrícolas, corpos de água, rochas e solos, áreas urbanas, etc.). Entretanto, essa capacidade de discriminar é limitada quando estão presentes na cena que está sendo analisada classes espectralmente muito semelhantes, isto é, classes cujos vetores de médias são muito próximos entre si. Sensores hiperespectrais podem ser usados para auxiliar nesse problema. Pode-se mostrar que classes espectralmente muito semelhantes entre si, ou mesmo idênticas, isto é, classes que compartilham do mesmo vetor de médias podem ser separadas com alta acurácia em espaços de alta dimensionalidade, desde que suas matrizes de covariância sejam suficientemente distintas (FUKUNAGA, 1990).

Do ponto de vista metodológico, a análise automática de dados hiperespectrais não é uma tarefa trivial. Possivelmente, no âmbito da classificação supervisionada, o maior desafio consista na estimação de parâmetros associados a classificadores paramétricos. O número de parâmetros a serem estimados, na matriz covariância particularmente, cresce rapidamente na medida em que a dimensionalidade dos dados aumenta. Nestas circunstâncias, se o número das amostras de treinamento não aumenta proporcionalmente, a confiabilidade na estimação destes parâmetros decresce. Para calcular quantos parâmetros é necessário estimar em cada caso específico, aplica-se:

$$N_P = \left(n + \frac{n^2 - n}{2} \right) \qquad \textbf{(1)}$$

onde N_P é o número total de parâmetros a serem estimados e n a dimensionalidade dos dados. Os experimentos mostram que inicialmente o acréscimo de novas bandas espectrais faz com que a acurácia tenda a aumentar. Em um determinado ponto, um máximo para a acurácia é atingido. A partir daí, um acréscimo adicional do número de

bandas resulta em um decréscimo na acurácia fornecida pelo processo de classificação. Este fato é conhecido como *Fenômeno de Hughes* ou a Maldição da Dimensionalidade (Figura 4).

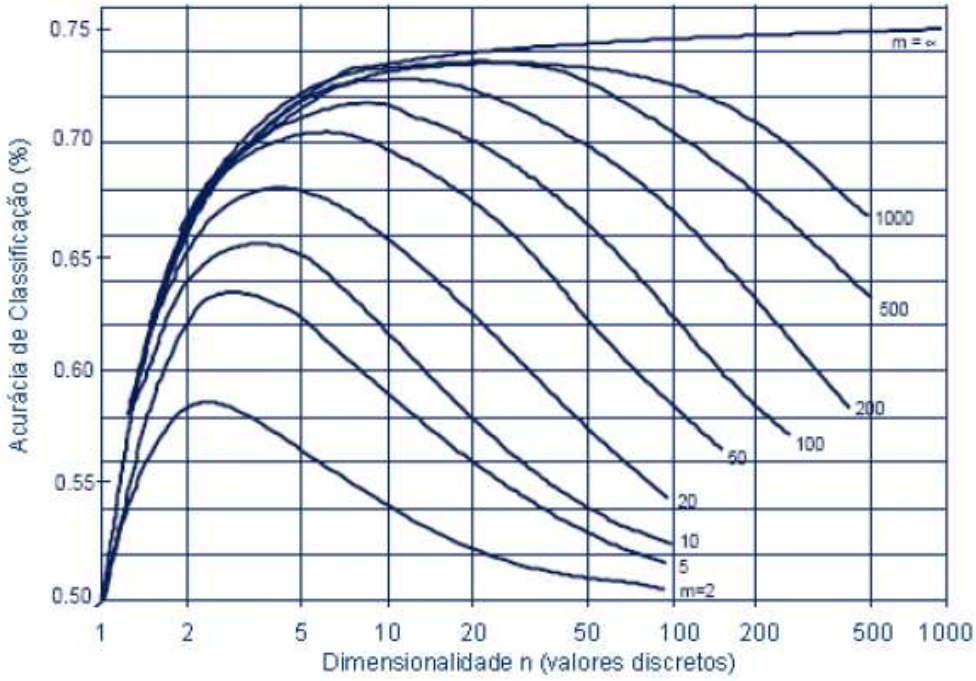


Figura 4: Fenômeno de Hughes. Fonte: LANDGREBE, (2003).

2.1.2 Métodos para Mitigar o Fenômeno de Hughes

Esforços vêm sendo desenvolvidos pela comunidade internacional na busca de metodologias que permitam reduzir os efeitos do Fenômeno de Hughes, viabilizando desta forma a utilização de dados em alta dimensionalidade em situações práticas. Uma revisão na literatura especializada permite identificar três linhas principais na abordagem deste problema:

- ✓ técnicas em análise discriminante regularizada (regularização das matrizes de covariância). Em termos gerais, busca-se nesta abordagem obter uma estimativa mais confiável das matrizes de covariância de cada uma das classes envolvidas, combinando-se estimativas amostrais destas matrizes com estimativa da matriz de covariância comum entre as classes. Dentre as pesquisas recentes, destacam-se as feitas por HASTIE et al. (1995), KUO & CHANG (2007), BERGE et al. (2007), HOFFBECK & LANDGREBE (1996), AEBERHARD et al. (1994) e CORTIJO & DE LA BLANCA (1999);
- ✓ incremento no número de amostras de treinamento, com a introdução das chamadas amostras semi-rotuladas, juntamente com as amostras de treinamento

disponíveis (amostras rotuladas). LICZBINSKI & HAERTEL (2008), JACKSON & LANDGREBE (2001), e MINGMIN & BRUZZONE (2005) demonstram, através de seus experimentos, que a abordagem contribui para o aumento da acurácia no processo de classificação,

- ✓ redução na dimensionalidade dos dados, com perda mínima de informação (técnicas de seleção ou extração de variáveis). Estudos relacionados com técnicas que envolvem redução da dimensionalidade (JIA & RICHARDS (1999), JIMENEZ & LANDGREBE (1999), SERPICO & BRUZZONE (2001), WANG (2008), ZHONG & WANG (2008), e ZORTEA et al. (2007), por exemplo) obtiveram resultados notáveis;

Uma alternativa à esses métodos é a utilização de classificadores não-paramétricos. Recentemente, particular atenção tem sido dedicada, pela comunidade internacional, às técnicas SVM para fins de reconhecimento de padrões em imagens de alta dimensionalidade em sensoriamento remoto. Estudos desenvolvidos por alguns autores, como MELGANI & BRUZZONE (2004), FOODY & MATHUR (2008) e HUANG et al. (2002), afirmam que SVM binários, aplicados em problemas multi-classe, tem freqüentemente proporcionado uma acurácia na classificação melhor que outras técnicas de reconhecimento de padrões largamente utilizadas. Outros autores, como GUO et al. (2008), BAZI & MELGANI (2006) e ZHAN & SHEN (2006) propõe métodos para aumentar a acurácia no classificador SVM, fazendo uso de conhecimento à priori e atribuindo pesos espectrais às bandas, aplicando algoritmos genéticos para estimação de parâmetros, e incluindo um termo de penalidade na função de decisão para suprimir *outliers*, respectivamente.

2.2 REDUÇÃO DA DIMENSIONALIDADE DOS DADOS

No presente trabalho, faz-se uso do algoritmo SFS para a seleção de variáveis em cada nó da árvore binária, e, por este motivo, um resumo das técnicas para redução da dimensionalidade dos dados fez-se necessário ao entendimento do mesmo. Optou-se em usar tal abordagem pois, segundo LANDGREBE (2003), o emprego de todas as variáveis originais disponíveis (número total de bandas) ou de um número grande destas pode ser prejudicial à eficiência do processo de classificação. Deste modo, uma etapa importante que deve preceder o processo de classificação refere-se redução da dimensionalidade dos dados, que pode ser feita a partir de duas abordagens distintas: seleção ou extração de variáveis. Trata da escolha de um subconjunto ótimo ou pelo menos sub-ótimo de variáveis para cada situação.

2.2.1 Seleção de Variáveis

A maneira mais direta de reduzir a dimensionalidade dos dados, provavelmente seja simplesmente selecionando um subconjunto de bandas espectrais, contendo aquelas que oferecem melhores condições de separabilidade entre as classes em consideração. A seleção de um subconjunto contendo *m* variáveis entre um total de *n* bandas pode se tornar muito alto e, portanto, computacionalmente proibitivo, caso a procura seja exaustiva. O número de possíveis combinações (N_C) pode ser calculado pelo coeficiente binomial:

$$N_C = \binom{n}{m} = \frac{n!}{(n-m)!m!} \qquad \textbf{(2)}$$

Desta forma, as técnicas de seleção de variáveis geralmente envolvem um algoritmo de procura associado a uma função critério. O algoritmo de procura gera e compara possíveis soluções, e aplicando uma função critério seleciona o melhor subconjunto.

Entre as estratégias de procura sub-ótimas esta o algoritmo "*Sequential Forward Selection*" (SFS) (SERPICO et al., 2003). O algoritmo SFS identifica iterativamente o melhor subconjunto de variáveis que pode ser obtido pela adição de uma variável por vez ao subconjunto selecionado. Parte-se do conjunto das variáveis originais *X*, com dimensionalidade *n* e de um conjunto *S*, inicialmente vazio ($S=\varnothing$), que conterá as variáveis selecionadas. Uma variável em *X*-*S*, aquela que maximiza o critério de separabilidade escolhido é acrescentada a *S*. A cada etapa do processo iterativo, o critério de separabilidade leva em conta o conjunto já obtido, acrescentado de uma banda de *X*-*S* para o cálculo e determinação do próximo elemento a ser inserido naquele subconjunto. O processo continua até que o número de variáveis em *S* atinja o valor desejado *m* ($m<n$).

Esse método de procura tem uma complexidade computacional relativamente baixa, mas com o inconveniente de que, uma vez que a variável foi selecionada, ela não pode ser descartada. Esta peculiaridade caracteriza o processo como sub-ótimos (SERPICO et al., 2003). Entre outros, algoritmos como o "*Sequential Forward Floating Selection*" (SFFS) trouxe melhoras ao algoritmo SFS permitindo reconsiderar as variáveis incluídas numa iteração prévia.

2.2.2 Extração de Variáveis

Nesta abordagem, são utilizadas transformações (lineares ou não-lineares) das bandas espectrais originais. Estas transformações são selecionadas de forma que o poder de separação fique concentrado em um número menor de bandas, permitindo assim uma diminuição na dimensionalidade dos dados, minimizando a perda de informações. Na literatura encontram-se citados vários métodos para extração de variáveis, sendo as mais conhecidas: análise de componentes principais, também conhecida como transformação de *Karhunen-Loéve* e análise canônica. Para informações mais detalhadas ver LANDGREBE (2003).

2.3 MEDIDAS ESTATÍSTICAS DE SEPARAÇÃO ENTRE CLASSES

A metodologia que implementa as técnicas de seleção de variáveis faz uso de medidas estatísticas de distância para selecionar um subconjunto ótimo ou sub-ótimo das bandas espectrais originais para fins de separação entre as classes em consideração. Entre as medidas de separação entre classes conhecidas, está a Distância de *Bhattacharyya* e sua derivada, Distância *Jeffries-Matusita*.

2.3.1 Distância de Bhattacharyya

A distância de *Bhattacharyya* é uma distância estatística que pode ser usada na estimação da separabilidade entre um par de classes (THERRIEN, 1989) e (DUDA et al., 2000). A forma geral da distância de Bhattacharyya é definida por:

$$B = -\ln\left[\int_{-\infty}^{\infty}\sqrt{p(X/\omega_1)\,p(X/\omega_2)}\,dX\right] \qquad \textbf{(3)}$$

onde $p(x|\,\omega_1)$ e $p(x|\,\omega_2)$ são funções densidade de probabilidade das classes 1 e 2.

Assumindo a distribuição Normal multivariada para os dados, esta distância estatística assume a seguinte forma:

$$B = \frac{1}{8}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\left(\frac{\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2}{2}\right)^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)+\frac{1}{2}\ln\left(\frac{\left|\frac{(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)}{2}\right|}{|\boldsymbol{\Sigma}_1|^{1/2}|\boldsymbol{\Sigma}_2|^{1/2}}\right) \qquad \textbf{(4)}$$

onde $\boldsymbol{\mu}_1$ e $\boldsymbol{\mu}_2$ são os vetores média e $\boldsymbol{\Sigma}_1$ e $\boldsymbol{\Sigma}_2$ são as matrizes de covariância das classes. Na expressão acima, pode-se observar que a primeira parcela no membro da

direita estima a contribuição dos vetores de médias (*Bhatt Mean*) no valor da distância de *Bhattacharyya* (B), enquanto que a segunda parcela (*Bhatt Cov*) estima a contribuição das matrizes de covariância. Note que B é um número real, variando no intervalo [0, ∞].

A distância de *Bhattacharyya* é uma medida teórica da distância entre duas distribuições gaussianas que é equivalente a um limiar superior (*upper bound*) do erro mínimo que pode ser obtido utilizando um classificador bayesiano. Ela apresenta a vantagem adicional de ser computacionalmente simples:

$$\xi_{Bayes} \leq \frac{1}{2} e^{-B} \qquad \textbf{(5)}$$

Note que quando as matrizes de covariância para as duas classes são iguais, a distância de *Bhattacharyya* (B), e a distância da Divergência (D), são medidas equivalentes, tal que:

$$D = 8B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \qquad \textbf{(6)}$$

Essa equação é também conhecida como a distância de *Mahalanobis* entre duas classes.

2.3.2 Distância de Jeffries-Matusita

Para o caso particular de dados normalmente distribuídos, pode-se provar, segundo SERPICO et al. (2003), que a distância *Jeffries-Matusita* é derivada da Distância de *Bhattacharyya* e igual a:

$$J_{ij} = 2\left[1 - \exp(-B_{ij})\right] \qquad \textbf{(7)}$$

onde B_{ij} é dada pela Distância de *Bhattacharyya*. Ao contrário desta, que tem limite superior infinito, o limite superior da Distância *Jeffries-Matusita* é dado por 2. A ausência de um limite superior para B_{ij} produz um comportamento indesejável pois um aumento na distância entre as classes pode significar uma redução insignificante na probabilidade de erro. Neste sentido, a saturação de J_{ij} ($J_{ij} \rightarrow 2$ para $B_{ij} \rightarrow +\infty$) evita esse efeito e faz com que esta seja uma medida mais realística entre as duas classes.

2.4 Classificador de Bayes e Máxima Verossimilhança

A efetiva utilização do classificador Máxima Verossimilhança requer que seja conhecida a forma genérica da função densidade de probabilidade $p(x|\omega_i)$. A

experiência prática utilizando dados multiespectrais em sensoriamento remoto tem mostrado que a função densidade de probabilidade multivariada Gaussiana descreve bem o comportamento destes dados. A função de probabilidade a posteriori $P(\omega_i|x)$ pode ser expressa utilizando-se o teorema de Bayes (RICHARDS & JIA, 1999):

$$P(\omega_i \mid x) = \frac{p(x \mid \omega_i).P(\omega_i)}{p(X)} \qquad \textbf{(8)}$$

onde $P(\omega_i)$ é a probabilidade a priori para a classe ω_i, $p(\omega_i|x)$ é a probabilidade a posteriori da classe *i* e $p(x|\ \omega_i)$ é o valor da função densidade de probabilidade de x condicional à classe ω_i. Sendo $p(X)$ apenas um fator de escala, que pode ser considerada comum a todas as classes, pode-se retira-la da equação sem que o resultado da classificação seja alterado. Assim

$$\boldsymbol{x} \in \omega_i \quad \text{se} \quad p(x \mid \omega_i).P(\omega_i) > p(x \mid \omega_j).P(\omega_j) \qquad \forall \quad i \neq j \qquad \textbf{(9)}$$

é conhecida como a regra de decisão de Bayes. Como geralmente não se possui informação suficiente sobre as probabilidades a priori das classes, é freqüente assumir-se valores iguais para as probabilidades a priori. A regra de classificação resultante é conhecida como Máxima Verossimilhança:

$$g_i(x) = p(x/\omega_i) \qquad \textbf{(10)}$$

Finalmente, para a implementação efetiva de (10) deve-se optar por uma função densidade de probabilidade. Pelas razões anteriormente mencionadas, opta-se pela função de Gauss multivariada:

$$p(\mathbf{x} \mid \omega_i) = (2\pi)^{-N/2} \mid \boldsymbol{\Sigma}_i \mid^{-1/2} \exp\left\{-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_i\right)^T \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_i\right)\right\} \qquad \textbf{(11)}$$

onde $\boldsymbol{\mu}_i$ representa o vetor média, $\boldsymbol{\Sigma}_i$ representa a matriz de covariância da classe ω_i, e *N* é a dimensionalidade dos dados.

Neste caso, a regra de decisão Máxima Verossimilhança fica:

$$g_i(\mathbf{x}) = -\ln \mid \boldsymbol{\Sigma}_i \mid -\left(\mathbf{x}-\boldsymbol{\mu}_i\right)^T \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_i\right) \quad \text{para} \quad i=1, 2, ..., k. \qquad \textbf{(12)}$$

Decidida a forma geral da função de decisão a ser utilizada, os parâmetros que constam na função decisão devem ser estimados para cada uma das classes presentes na cena sendo analisada. Fica assim definida a função de decisão $g_i(x)$ associada a cada uma das classes ω_i , com $i = 1, 2, \ldots m$, sendo *m* o número de classes em consideração. A regra de classificação de padrões individuais é dada por:

$$\mathbf{x} \in \omega_i \quad se \quad g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \quad \forall \quad i \neq j \qquad \textbf{(13)}$$

É importante ressaltar que os classificadores de Bayes e Máxima Verossimilhança são paramétricos (usam, por exemplo, como função densidade de probabilidade a função de Gauss multivariada) e, portanto, estão sujeitos aos efeitos do fenômeno de Hughes.

Outro fato importante a ser considerado é que a função de decisão dada em (12) pode ser utilizada levando-se em conta todas as classes juntas (CEU), ou um subconjunto de classes a cada vez (CEM). Neste trabalho tal função de decisão foi implementada na forma de árvore binária (portanto, um CEM) onde apenas um par de classes é considerado em cada nó.

2.5 CLASSIFICADORES DE DECISÃO EM ÁRVORE

Classificadores de Decisão em Árvore (CDA) são usados com sucesso em diversas áreas do conhecimento, como classificação de sinais de radar, reconhecimento de caracteres, sensoriamento remoto, diagnósticos médicos, entre outros. Talvez, a característica mais importante dos CDA's é a sua capacidade de quebrar processos complexos em uma coleção de decisões mais simples. Segundo SAVAVIAN & LANDGREBE (1991), dados estruturados em árvores são grafos direcionados acíclicos e devem satisfazer algumas propriedades:

- ✓ possui apenas um nó chamado raiz, no nível 0 e sem arestas de chegada;
- ✓ o nó raiz contém todos os padrões de todas as *n* classes a serem classificados pelo CDA;
- ✓ cada um dos demais nós contém uma, e apenas uma aresta de chegada; Existe um único caminho do nó raiz aos demais nós;
- ✓ considera-se como nó filho os nós que são originados por determinado nó que será chamado de pai. Na Figura 5, os nós situados no nível 1 são filhos (ou descendentes) do nó raiz situado no nível 0, que por sua vez é pai dos nós situados nível 1;
- ✓ numa árvore binária, os nós filhos serão diferenciados entre si, sendo chamados de nó filho direito e nó filho esquerdo;
- ✓ os nós que não possuírem nós filhos são chamados de folhas ou terminais. Em tais nós, o padrão agora discriminado, recebe a identificação (rótulo) da classe do nó (ω_1 e ω_2); os demais nós (com exceção do nó raiz) são chamados de nós internos;
- ✓ com exceção dos nós folhas, os demais nós (representado por t), são definidos por três componentes: uma regra de decisão (*D(t)*), as classes presentes no nó (*C(t)*), e as feições usadas por esse nó (*F(t)*);

✓ o processo de discriminação dos padrões no CDA utiliza em cada nó uma regra de decisão sobre um conjunto de feições para tentar discriminar um determinado conjunto de classes (Figura 5).

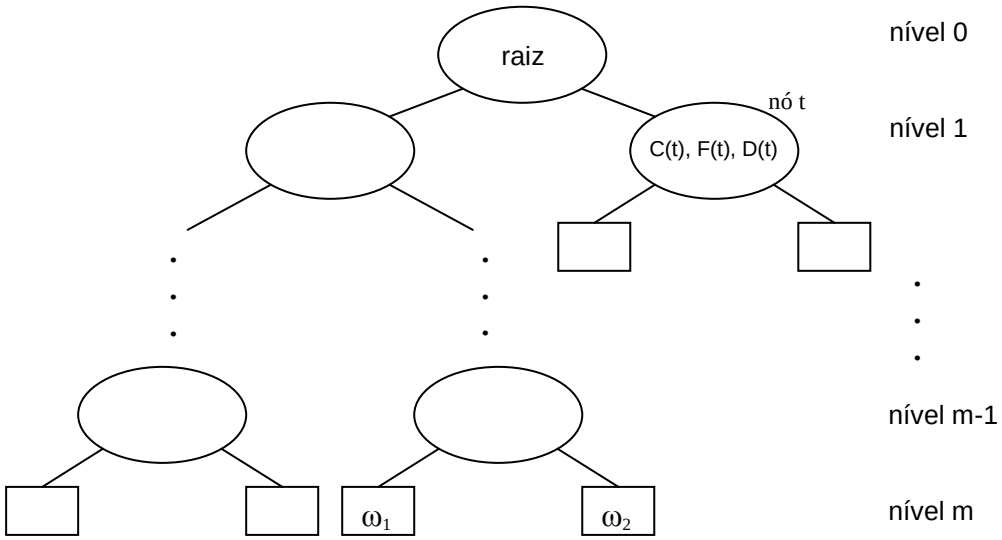


Figura 5: Estrutura de um classificador de decisão em árvore binária. Fonte: Adaptado de Savavian & Landgrebe (1991)

SAVAVIAN & LANDGREBE (1991) apresentam ainda uma série de vantagens e desvantagens no uso de CDA's (CEM), em comparação com CEU's:

✓ decisões complexas globais podem ser aproximadas pela união de simples decisões locais, nos vários níveis da árvore;

✓ em contraste com os convencionais CEU's, onde cada amostra é testada contra todas as outras classes, perdendo eficiência, em um classificador em árvore a amostra é testada contra apenas um subconjunto de classes, eliminando, assim, processamento desnecessário;

✓ em CEU's apenas um subconjunto de variáveis (bandas) são usados para a discriminação entre as classes. Esse subconjunto é selecionado, normalmente, por um critério global ótimo. Nos CDA's, por outro lado, tem-se a flexibilidade de escolher diferentes subconjuntos de variáveis nos diferentes nós da árvore para que, este subconjunto possa discriminar otimamente as classes neste nó;

✓ quando se tem um número grande de variáveis e classes, precisa-se estimar os parâmetros das distribuições das classes como, por exemplo, as probabilidades à priori, com um conjunto pequeno de amostras de treinamento. Com isso, tem-se o problema provocado pela alta dimensionalidade dos dados. Esse problema pode ser resolvido nos CDA's porque este poderá fazer uso um número menor de variáveis em cada nó, sem degradação excessiva da performance.

Por outro lado, um dos problemas que os CDA's podem apresentar é em relação à sobreposição de classes. Quando o número de classes é relativamente grande, pode haver uma quantidade de nós terminais muito maiores que o número classes, aumentando o custo de processamento. Sabe-se das dificuldades de modelar uma estrutura de CDA ótima, pois o desempenho de um CDA está fortemente associado à estrutura de árvore utilizada. Um classificador não é construído sem que haja um objetivo, eles estão baseados em experiências passadas. A construção de um classificador impõe este aprendizado como pré-requisito, podendo inferir, assim, que parte do processo é empírico ou baseado em relatos da literatura. Sabe-se ainda que os CDA's podem apresentar erros cumulativos, propagando-os de nível para nível, o que requer cuidados com relação à estrutura de árvore utilizada e ao método de abordagem sobre o CDA.

2.5.1 Modelos de Classificadores de Decisão em Árvore

De acordo com SAVAVIAN & LANDGREBE (1991), os principais objetivos dos CDA's são classificar tão corretamente quanto possível as amostras de treinamento, generalizar as amostras de treinamento para que as amostras desconhecidas possam ser classificadas com alta acurácia e decidir por uma estrutura de CDA simples. Desta forma, a determinação do *design* dos CDA's podem ser decompostas nas seguintes tarefas:

- ✓ escolha apropriada da estrutura da árvore;
- ✓ escolha do subconjunto de variáveis a serem usados em cada nó;
- ✓ escolha da regra de decisão ou a estratégia a ser usada em cada nó.

Existem vários métodos heurísticos para a construção dos CDA's, entre eles a abordagem *bottom-up* e a *top-down*. Na abordagem *bottom-up*, a árvore binária é construída a partir de um conjunto contendo todas as classes, dispostas no nó raiz. Através de um conjunto de feições das amostras de treinamento das classes é estimada uma medida de separabilidade com o objetivo de identificar o par de classes que apresentar a maior distância entre as suas componentes. Estas duas classes são então utilizadas na definição das regras de decisão que irão caracterizar os dois nós descendentes. Este procedimento repete-se a cada nó, até que os nós terminais (constituídos por apenas uma classe) sejam atingidos. A discriminação, nesta abordagem, é mais significante quanto menor for o nível da árvore, ou seja, quanto mais próximo estiver da raiz. A discriminação é mais suave quanto mais próximo estiver do nível *m*, próximo ao nó terminal (SAVAVIAN & LANDGREBE, 1991). A principal vantagem que a abordagem *bottom-up* do tipo binário apresenta é que

somente duas classes são consideradas em cada nó, fornecendo um ganho representativo ao permitir a seleção mais propícia das variáveis em cada nó, ao invés de selecionar do conjunto total de variáveis, a com maior separabilidade entre a totalidade de classes, proporcionando uma maior acurácia no processo de classificação.

Na abordagem *top-down*, o *design* dos CDA's são reduzidas a três tarefas: seleção da regra de decisão, determinação dos nós terminais e atribuição aos nós terminais do rótulo da classe. Usando alguma medida de distância, como a Distância de Bhattacharyya, por exemplo, as distâncias entre as classes definidas a priori são computadas em cada etapa e as duas classes que obtiverem a menor distância serão fundidas em um novo grupo. O vetor de médias e a matriz de covariância para cada grupo são computados a partir das amostras de treinamento das classes, e o processo é repetido até que apenas um grupo reste na raiz (SAVAVIAN & LANDGREBE, 1991). A maior parte das pesquisas em *design* dos CDA's esta concentrada na área que estuda as regras de decisão (SAVAVIAN & LANDGREBE, 1991).

CAPÍTULO III

METODOLOGIA

3.1 INTRODUÇÃO

Conforme descrito nos capítulos anteriores, em situações reais o número de amostras de treinamento disponíveis é geralmente limitado. Esta deficiência resulta em estimativas amostrais pouco confiáveis para os parâmetros dos classificadores paramétricos, em especial para a matriz de covariância.

No capítulo anterior foram rapidamente revistas as três metodologias que vem sendo investigadas pela comunidade internacional com o objetivo de reduzir os efeitos nocivos causados por este problema (análise discriminante regularizada, uso de amostras semi-rotuladas e redução na dimensionalidade das variáveis por métodos de seleção ou extração de variáveis). Uma outra possível alternativa, citada anteriormente, consiste no emprego de classificadores não paramétricos, que apresentam menor sensibilidade à questão do número de amostras de treinamento. Nesta dissertação é investigada a performance de um classificador não-paramétrico que vem despertando considerável interesse na comunidade internacional, quando aplicado à imagens hiperespectrais de sensoriamento remoto. Este classificador, SVM, apresenta um grande potencial para aplicações em imagens de sensoriamento remoto sendo, portanto, objeto de investigação nesta dissertação.

3.2 SUPPORT VECTOR MACHINE (SVM)

SVM é uma técnica de aprendizado de máquina, fundamentada nos princípios da Minimização do Risco Estrutural (*Structural Risk Minimization* – SRM) (VAPNIK, 1999). Esta técnica busca minimizar o erro com relação ao conjunto de treinamento (risco empírico), assim como o erro com relação ao conjunto de teste, isto é, conjunto de amostras não empregadas no treinamento do classificador (risco na generalização). O objetivo de SVM consiste em obter um equilíbrio entre esses erros, minimizando o excesso de ajustes com respeito às amostras de treinamento (*overfitting*) e aumentando conseqüentemente a capacidade de generalização (VAPNIK, 1999). O problema denominado de *overfitting* consiste em o classificador memorizar os padrões

de treinamento, gravando suas peculiaridades e ruídos, ao invés de extrair as características gerais que permitirão a generalização ou reconhecimento de padrões não utilizados no treinamento do classificador (SMOLA et al., 2000).

A questão da generalização pode ser mais bem avaliada para o caso de duas classes. Assumindo que as amostras de treinamento das duas classes são linearmente separáveis, a função de decisão mais adequada é aquela para a qual a distância entre os conjuntos das amostras de treinamento é maximizada. Neste contexto, a função de decisão que maximiza esta separação é denominada de ótima (Figura 6).

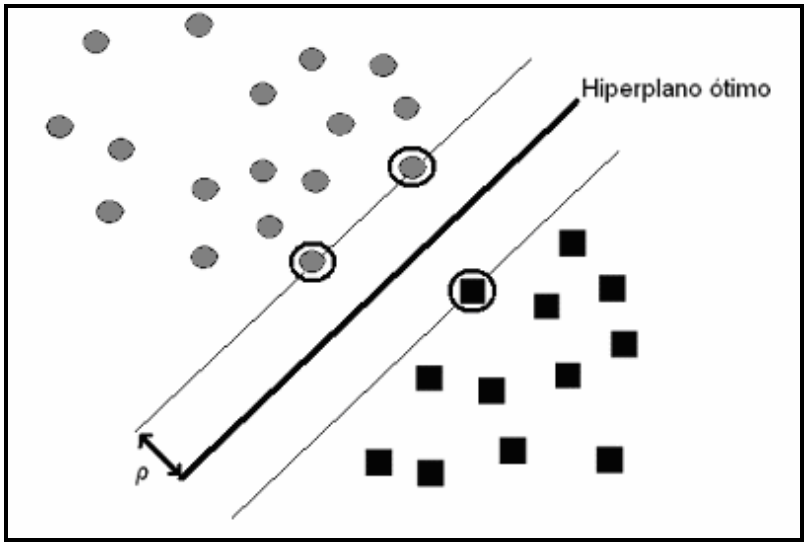


Figura 6. O hiperplano ótimo separando os dados com a máxima margem ρ. Os support vectors (amostras circuladas) em uma distribuição dos dados no R^2. Fonte: Adaptado de ABE (2005).

Este princípio é implementado em *SVM* e a correspondente formulação matemática dada a seguir está baseada em ABE (2005).

Seja x_i *(i=1, 2, ..., m)* um conjunto de treinamento em um problema que consiste de duas classes linearmente separáveis (ω_1 *e* ω_2). Cada amostra fica associada a um rótulo: $y_i = 1$ se $x_i \in \omega_1$, $y_i = -1$ se $x_i \in \omega_2$. A forma geral da função de decisão linear é:

$$D(\mathbf{x}) = \sum_{i=1}^{m} w_i x_i + b \qquad \textbf{(14)}$$

ou equivalentemente, em termos do produto interno dos dois vetores:

$$D(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \qquad \textbf{(15)}$$

onde **w** é um vetor *m*-dimensional (pesos) e *b* é o termo independente. Ao longo da dissertação, *m* representa a dimensionalidade dos dados, e *M* o número de amostras de treinamento, ou seja, os vetores **w** e **x** são representados por w_i e x_i para i= 1,..,*m* bandas e $\mathbf{x}_j$ para j=1,..,*M* amostras.

O processo de classificação fica, portanto:

$$\begin{aligned} &\text{caso } \mathbf{w}\mathbf{x}_i + b > 0, \quad \mathbf{x}_i \in \omega_1 \ (y_i = 1) \\ &\text{caso } \mathbf{w}\mathbf{x}_i + b < 0, \quad \mathbf{x}_i \in \omega_2 \ (y_i = -1) \end{aligned} \qquad \textbf{(16)}$$

Como as amostras são linearmente separáveis, não ocorrerá a situação em que $\mathbf{w}\mathbf{x}_i + b = 0$. Portanto, as condições acima podem ser reescritas como:

$$\begin{aligned} &\mathbf{w}\mathbf{x}_i + b > a \quad \text{para } \mathbf{x}_i \in \omega_1 \ (y_i = 1) \\ &\mathbf{w}\mathbf{x}_i + b < -a \quad \text{para } \mathbf{x}_i \in \omega_2 \ (y_i = -1) \end{aligned} \qquad \textbf{(17)}$$

sendo *a* uma constante (*a*>0). As inequações (17) podem ser rearranjadas, dividindo ambos os membros por *a* e ajustando **w**, b

$$\mathbf{w}^T\mathbf{x}_i + b \begin{cases} \geq 1 & \text{para } y_i = 1 \\ \leq -1 & \text{para } y_i = -1 \end{cases} \qquad \textbf{(18)}$$

Deste modo, ambas as condições podem ser combinadas em uma única:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \text{para } i=1, 2, ..., M \qquad \textbf{(19)}$$

O hiperplano:

$$D(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = c \ \text{ para } -1 < c < 1 \qquad \textbf{(20)}$$

forma, então, a superfície de separação entre as duas classes. Para *c=0*, a equação (20) define um hiperplano situado à meia distância entre os dois hiperplanos extremos (*c=1 e c= -1*). A distância entre estes dois hiperplanos extremos é denominada de "margem", representada por ρ na Figura 6. Supondo a existência de pelo menos uma amostra **x** para a qual *D*(***x***) = 1, e pelo menos uma outra amostra para a qual *D*(***x***) = -1, então o hiperplano *D*(***x***) = 0 representa a melhor superfície de separação entre estas amostras, no sentido de que maximiza o poder de generalização do classificador. A região entre os dois hiperplanos extremos (-1 ≤ *D*(***x***) ≤ 1) é a região de generalização. O hiperplano *D*(***x***)=0, ao maximizar o valor da margem, maximiza a região de generalização sendo, portanto, neste sentido ótimo (Figura 6).

Se considerarmos que a Distância Euclidiana de uma amostra ***x*** a um plano *D(**x**)* é dada por:

$$|D(\mathbf{x})| / ||\mathbf{w}|| \qquad \textbf{(21)}$$

sendo *D(**x**)* dado por (20), o hiperplano ótimo será aquele para o qual esta distância é máxima. Esta condição pode ser obtida minimizando-se ||***w***||, ou equivalentemente, minimizando

$$Q(\mathbf{w}) = \frac{1}{2} \| \mathbf{w} \|^2 = \frac{1}{2} \mathbf{w}^T \mathbf{w} \qquad \textbf{(22)}$$

com respeito aos parâmetros da função D(**x**) (***w*** e *b*). Para satisfazer a convenção adotada com relação ao rótulo de cada amostra (y_i), a restrição (19) deve ser imposta. Tal restrição é imposta de maneira a assegurar que não ocorram amostras de treinamento na região de separação entre as duas classes (entre as margens).

A inclusão das restrições (19) no problema de minimização de (22) pode ser resolvido por meio da técnica dos multiplicadores de Lagrange (α). Esta abordagem pode ser expressa por:

Minimizar

$$Q(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^{M} \alpha_i \left\{ y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) - 1 \right\} \qquad \textbf{(23)}$$

sendo $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)$ os multiplicadores de Lagrange, um vetor de dimensão M, com $\alpha_i \geq 0$. A solução para este problema de extremos pode então ser encontrada minimizando-se Q(**w**,b,$\boldsymbol{\alpha}$) com relação a **w**, b e maximizando-se com relação a α_i (≥0)

$$\frac{\partial Q(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{M} \alpha_i \, y_i \, \mathbf{x_i} = 0 \qquad \textbf{(24)}$$

ou seja, $\mathbf{w} = \sum_{i=1}^{M} \alpha_i \, y_i \, \mathbf{x_i}$

e

$$\frac{\partial Q(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^{M} \alpha_i \, y_i = 0 \qquad \textbf{(25)}$$

acrescidas das condições:

$$\alpha_i \{ y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \} = 0 \quad \text{para} \quad i=1, \ldots, M \qquad \textbf{(26)}$$

As equações 24, 25 e 26 são conhecidas como as condições de Karush-Kuhn-Tucker (KKT) (ABE, 2005). Substituindo-se (24) e (25) em (23), obtém-se uma equação expressa em termos de $\boldsymbol{\alpha}$ somente:

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \qquad \textbf{(27)}$$

E o problema transforma-se, portanto, em maximizar (27) com respeito à $\boldsymbol{\alpha}$ e sujeito às restrições:

$$\sum_{i=1}^{M} \alpha_i y_i = 0 \quad \text{e} \quad \alpha_i \geq 0 \quad \text{para } i=1, \ldots, M \qquad \textbf{(28)}$$

Essa formulação é denominada na literatura de forma dual, enquanto o problema original é referenciado como forma primal. A forma dual possui os atrativos de apresentar restrições mais simples e permitir a representação do problema de otimização em termos de produtos internos entre dados, o que será útil quando se tratar de SVM para dados não-lineares. É interessante observar também que o problema dual é formulado utilizando apenas os dados de treinamento e seus rótulos (LORENA & CARVALHO, 2007).

Substituindo a equação (24) em (15), tem-se a função de decisão:

$$D(\mathbf{x}) = \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b \qquad \textbf{(29)}$$

onde *S* é o conjunto de índices dos *support vectors*, isto é, as amostras de treinamento para as quais $\alpha_i > 0$.

Da equação (26) pode-se observar que para $\alpha_i > 0$, deve-se ter $y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0$. As amostras $\mathbf{x}_i$ que satisfazem esta condição são aquelas para as quais D(**x**)=1, isto é, as que são *support vectors*, conforme mencionado anteriormente. O termo independente *b* é dado, portanto, por $b = y_i - \mathbf{w}^T \mathbf{x}_i$, na condição de que $\mathbf{x}_i$ seja um *support vector* ($\alpha_i > 0$). Uma estimativa mais confiável pode ser obtida tomando-se um valor médio com respeito a todos os *support vectors*:

$$b = \frac{1}{|S|} \sum_{i \in S} (y_i - \mathbf{w}^T \mathbf{x}_i) \qquad \textbf{(30)}$$

sendo *S* o conjunto de todos os *support vectors* e |*S*| o número de *support vectors* que ocorrem no problema.

A formulação acima apresenta solução somente no caso de as amostras $\boldsymbol{x}_i$ pertencentes às duas classes serem linearmente separáveis. Em situações reais, é difícil encontrar aplicações cujos dados sejam linearmente separáveis. Isso se deve a diversos fatores, entre eles a presença de ruídos e *outliers* nos dados ou ainda como resultado da própria natureza do problema, que pode ser não linear. Para estender SVMs lineares de margens rígidas para lidar com conjuntos de treinamento mais gerais, permite-se que alguns dados possam violar a restrição da Equação (19). Neste

caso, introduz-se o conceito de variável de folga (*slack variable*) representada por ξ_i ($\xi_i \geq 0$). Tais variáveis relaxam as restrições impostas ao problema de otimização primal, que se torna, então:

$$\alpha_i\{y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i\} = 0 \quad \text{para} \quad i=1, ..., M \qquad \textbf{(31)}$$

A aplicação desse procedimento suaviza as margens do classificador linear, permitindo que alguns dados permaneçam entre os hiperplanos formado pelos SVs e também a ocorrência de alguns erros de classificação. Por esse motivo, as SVMs obtidas neste caso também podem ser referenciadas como SVMs com margens suaves (*soft-margin*) (LORENA & CARVALHO, 2007; ABE, 2005).

Para o caso de $0 < \xi_i < 1$ a amostra correspondente não terá margem máxima, mas será rotulada corretamente. No caso de $\xi_i \geq 1$, a amostra $\boldsymbol{x}_i$ será rotulada erroneamente (Figura 7).

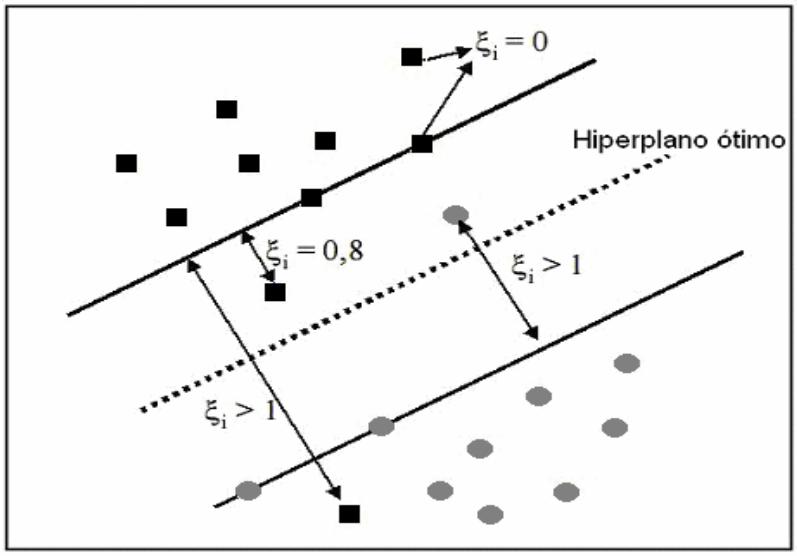


Figura 7. Exemplos de valores e situações da variável de folga ξ. Distribuição dos dados no R^2. Fonte: Adaptado de ABE (2005).

Para levar em consideração o termo ξ_i, minimizando assim o erro sobre os dados de treinamento, a equação 22 é reformulada como:

$$Q(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{M} \xi_i \qquad \textbf{(32)}$$

A constante C é conhecida como “parâmetro de margem”, e estabelece a importância relativa das duas parcelas do lado direito da igualdade da equação (32) neste processo de minimização. Como visto no caso anterior, a minimização de $\|\mathbf{w}\|^2$

resulta na maximização da margem enquanto que a minimização da segunda parcela ($\sum_{i=1}^{M}\xi_i$) resulta na minimização do erro de classificação. Como já visto, um valor de ξ_i (0; 1] indica uma amostra entre as margens (LORENA & CARVALHO, 2007; ABE, 2005).

A solução de (32) envolve passos matemáticos semelhantes aos apresentados anteriormente, com a introdução de uma abordagem Lagrangiana similar à aquela desenvolvida acima. Tem-se como resultado o problema dual igual à encontrada na equação (27), mas agora sujeita às seguintes restrições:

$$\sum_{i=1}^{M}\alpha_i y_i = 0 \quad \text{e} \quad C \geq \alpha_i \geq 0 \quad \text{para } i=1, \ldots, M \qquad \textbf{(33)}$$

Como nas SVMs de margens rígidas, os pontos $\mathbf{x}_i$ para os quais $\alpha_i>0$ são denominados *support vectors* (SVs), que são as amostras que formam o hiperplano separador. Da equação (31) e da condição complementar $(C-\alpha_i)\xi_i = 0$ (as condições KKT são dadas no teorema C1, em ABE, 2005), tem-se três diferentes casos para α_i (PONTIL & VERRI, 1998):

- ✓ Se $\alpha_i = 0$ e $\xi_i = 0$ tem-se $y_i(\mathbf{w}^T\mathbf{x}_i + b) = z + 1$, sendo z um número real positivo e, portanto, a amostra é corretamente classificada.
- ✓ Se $0 < \alpha_i < C$: têm-se pela equação (31) que $y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i = 0$; e pela condição $(C-\alpha_i)\xi_i = 0$ tem-se que $\xi_i = 0$; portanto, $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$. Neste caso $\mathbf{x}_i$ é SV e encontra-se sobre as margens, sendo denominado *unbounded support vector*.
- ✓ Se $\alpha_i = C$, tem-se pela condição $(C-\alpha_i)\xi_i = 0$ que $(C-\alpha_i) = 0$. Neste caso podem acontecer três casos:
  - o para $\xi_i>1$, $y_i(\mathbf{w}^T\mathbf{x}_i + b) = z$, sendo z um número real negativo. Deste modo a amostra $\mathbf{x}_i$ é mapeada do outro lado do hiperplano e, portanto, erroneamente classificado;
  - o para $0 < \xi_i \leq 1$, $y_i(\mathbf{w}^T\mathbf{x}_i + b) = z$, sendo z um número real positivo entre zero e um. Neste caso, a amostra é corretamente classificada, porém entre as margens;
  - o para $\xi_i=0$, $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$, e a amostra é mapeada sobre as margens;

    Os SVs para os quais $\alpha_i=C$ com $\xi_i=0$ são denominados *bounded support vectors*.

Tem-se como resultado final a mesma função de decisão representada pela Equação 29 e b (Equação 30) calculado sobre os *unbounded support vectors*. Neste caso, as variáveis α_i da função de decisão são determinadas pela solução de (27) com as restrições da Equação 33.

As SVMs lineares são eficazes na classificação de conjuntos de dados linearmente separáveis com a presença de alguns ruídos e *outliers*. Entretanto, em situações reais ocorre com bastante freqüência classes que não são linearmente separáveis, isto é, a separação entre as amostras de treinamento das duas classes requer uma função não-linear. A solução mais simples nestes casos consistiria na adoção de polinômios de grau mais elevado. Entretanto, esta abordagem apresenta o risco de excesso de ajuste (*overfitting*), e a conseqüente redução no poder de generalização do classificador (DUDA et al., 2000).

A notação utilizada é relacionada a seguir:

- ✓ Espaço original **X** (*input space*);
- ✓ Espaço característico: é o espaço no qual os dados são mapeados (*feature space*): {$\mathbf{g}(\mathbf{x})$: $\mathbf{x} \in \mathbf{X}$}, sendo $\mathbf{g}=(g_1, g_2, \cdots g_n)$ uma função não-linear mapeando do espaço original, que apresenta uma dimensão *m*, para um novo espaço (espaço característico), que apresenta uma dimensão *n*, com *n>m;*
- ✓ Variáveis originais: amostra **x** com dimensão *m*. O símbolo *M* representa o número de amostras (**x**) disponíveis no espaço original ($\mathbf{x}_1$, $\mathbf{x}_2$, . . . , $\mathbf{x}_M$);
- ✓ As *M* variáveis mapeadas no novo espaço (*espaço característico*) são, portanto, $\mathbf{g}(\mathbf{x}_1), \mathbf{g}(\mathbf{x}_2), \cdots, \mathbf{g}(\mathbf{x}_M)$, sendo $\mathbf{g}=(g_1, g_2, \cdots g_n)$ uma função não-linear mapeando cada uma das *M* amostras do espaço original para o espaço característico (com dimensão *n*):

$$\begin{bmatrix} g_1(\mathbf{x}_1) \\ g_2(\mathbf{x}_1) \\ \vdots \\ g_n(\mathbf{x}_1) \end{bmatrix}, \begin{bmatrix} g_1(\mathbf{x}_2) \\ g_2(\mathbf{x}_2) \\ \vdots \\ g_n(\mathbf{x}_2) \end{bmatrix}, \ldots, \begin{bmatrix} g_1(\mathbf{x}_M) \\ g_2(\mathbf{x}_M) \\ \vdots \\ g_n(\mathbf{x}_M) \end{bmatrix}$$

Observe que este mapeamento não deve ser confundido com um aumento no número de bandas. Como se pode notar, as novas variáveis resultam da aplicação de uma função sobre as variáveis originais (bandas espectrais) resultando no aumento no número de variáveis em cada banda.

A forma geral da função de decisão no espaço original é dada pela Equação 14. No espaço característico, selecionada a função $\mathbf{g}(\mathbf{x})$, a forma geral da função de decisão torna-se, portanto:

$$D(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{w}_i \, \mathbf{g}_i(\mathbf{x}) + b \qquad \textbf{(34)}$$

ou em notação vetorial:

$$D(\mathbf{x}) = \mathbf{w} \,.\, \mathbf{g}(\mathbf{x}) + b \qquad \textbf{(35)}$$

Considerando que **w**, similarmente à (24), é dado por:

$$\mathbf{w} = \sum_{i=1}^{M} \alpha_i \, y_i \, \mathbf{g}(\mathbf{x_i}) \qquad \textbf{(36)}$$

e substituindo-o em (34), a expressão para a função de decisão neste novo espaço pode ser escrita:

$$D(\mathbf{x}) = \sum_{i \in S} \alpha_i y_i \mathbf{g}(\mathbf{x}_i) \mathbf{g}(\mathbf{x}) + b \qquad \textbf{(37)}$$

Os elementos do produto interno $\mathbf{g}(\mathbf{x}_i).\mathbf{g}(\mathbf{x})$ podem ser representado pela matriz **H**, que é conhecida por matriz kernel, de dimensionalidade (*M,M*):

$$\mathbf{H}_{i,j} = \mathbf{g}(\mathbf{x}_i) \,.\, \mathbf{g}(\mathbf{x}_j) \quad \text{para} \quad i, j = 1, \ldots, M \qquad \textbf{(38)}$$

Observe que as amostras nunca aparecem isoladamente, mas sempre em pares, em um produto interno. Kernel pode, portanto, ser definido como o produto interno das variáveis (no espaço característico). A condição necessária para que uma função H seja um kernel é conhecida como condição de Mercer (ABE, 2005):

$$\sum_{i,j=1}^{M} h_i h_j H(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \qquad \textbf{(39)}$$

para todo M, $\mathbf{x}_i$ e h_i, onde M é um número natural (número de amostras disponíveis) e h_i é um número real, para os quais existe uma função de mapeamento $\mathbf{g}(\mathbf{x})$, que mapeia **x** no espaço característico.

Existem, portanto, duas possíveis abordagens ao problema do mapeamento de dados em espaços de dimensão mais elevada para fins de classificação empregando funções de decisão lineares $\mathbf{g}(\mathbf{x})$ (HERBRICH, 2002):

1- Selecione explicitamente uma função **g** para mapeamento dos dados em um espaço de dimensão mais alta.

2- Selecione diretamente um kernel H que satisfaça as condições de Mercer. Este kernel vai definir de uma forma implícita a função de mapeamento **g.**

Do ponto de vista matemático, as duas possíveis abordagens citadas acima são equivalentes. A segunda abordagem (escolha direta de um kernel) apresenta,

entretanto, a vantagem de ser mais fácil de implementar e de ser interpretada. Outra vantagem oferecida por esta abordagem consiste em não se necessitar operar diretamente no espaço em dimensão mais alta, no qual os dados estão sendo mapeados. Tanto a fase de treinamento do classificador quanto a fase de classificação dos dados utiliza-se diretamente H($\mathbf{x}_i$, $\mathbf{x}$) em lugar da função de mapeamento $\mathbf{g}(\mathbf{x})$.

Seguindo a metodologia anteriormente utilizada (Equações 24 e 25), o problema de maximização adquire a forma:

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j H(\mathbf{x}_i \mathbf{x}_j) \qquad \textbf{(40)}$$

sujeito às restrições:

$$\sum_{i=1}^{M} y_i \alpha_i = 0 \quad \text{e} \quad 0 \le \alpha_i \le C \quad \text{para} \quad i=1, \ldots, M \qquad \textbf{(41)}$$

Como se pode notar, as restrições dadas em (41) são as mesmas restrições dadas em (33), ou seja, aqui se utiliza a versão de SVM linear com margens suaves, que permite lidar com ruídos e *outliers* presentes nos dados. Pode-se mostrar que neste caso, a função de decisão assume a seguinte forma:

$$D(\mathbf{x}) = \sum_{i \in S} \alpha_i y_i H(\mathbf{x}_i, \mathbf{x}) + b \qquad \textbf{(42)}$$

sendo o coeficiente linear *b* dado por:

$$b = \frac{1}{|U|} \sum_{j \in U} (y_j - \sum_{i \in S} (\alpha_i y_i H(\mathbf{x}_i, \mathbf{x}_j)) \qquad \textbf{(43)}$$

e *U* representa o subconjunto composto pelos *support vectors* denominados de *unbounded*, isto é, aqueles para os quais ($0 < \alpha_i < C$).

Como já foi dito anteriormente, a forma da função discriminante depende do kernel adotado. Exemplos comuns de kernel são a Função Base Radial (RBF) (Equação 44) e o kernel Polinomial (Equação 45):

$$H(\mathbf{x}, \mathbf{x}_i) = \exp(-\gamma \| \mathbf{x} - \mathbf{x}_i \|^2) \qquad \textbf{(44)}$$

onde γ é um parâmetro positivo para controle.

$$H(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}^T \mathbf{x}_i + 1)^d \qquad \textbf{(45)}$$

onde *d* é um número natural e determina o grau do polinômio.

A regra de classificação é dada por:

$$D(\mathbf{x}_i)>0 \qquad \mathbf{x}_i \in \omega_1 \qquad\qquad (46)$$
$$D(\mathbf{x}_i)<0 \qquad \mathbf{x}_i \in \omega_2$$

Se $D(\mathbf{x}_i)=0$, então $\boldsymbol{x}_i$ está sobre o hiperplano separador e não é classificado. Quando as amostras de treinamento são linearmente separáveis, a região $\{\mathbf{x} \mid 1 > D(\boldsymbol{x}) > -1\}$ é a região de generalização.

Pode-se mostrar que SVM apresenta vantagens com respeito a classificadores convencionais, especialmente quando o número de amostras de treinamento é pequeno e a dimensionalidade dos dados é grande, devido ao fato de que os classificadores convencionais não têm mecanismos para maximizar a margem (distância entre os dois hiperplanos extremos). A maximização da margem permite aumentar a capacidade de generalização do classificador (ABE, 2005).

Finalmente, deve-se mencionar que SVM só pode ser utilizado na separação de um par de classes a cada vez. Dados de sensoriamento remoto de cenas naturais envolvem a presença de um número maior de classes. Desta forma, aplicações de técnicas SVM na classificação de imagens de sensoriamento remoto requerem abordagens adequadas. A maioria faz uma combinação de classificadores binários produzidos independentemente (MELGANI & BRUZZONE, 2004). Como exemplos de abordagens para o problema de uso de classificadores binários (caso do SVM) em problemas multi-classes pode-se citar, além das árvores binárias, as abordagens "um contra um" e "um contra todos".

3.3 ALGORITMO PROPOSTO

A metodologia adotada apresenta uma solução multi-classe usando SVM desenvolvido em forma de árvore binária, de acordo com a abordagem *bottom-up*. Espera-se, desta forma, incrementar a acurácia nas imagens temáticas e, então, comparar com a produzida pelo classificador MVG - largamente usado na comunidade científica em reconhecimento de padrões e que apresenta alta acurácia em dados multiespectrais para classes espectralmente diferentes. Para esse classificador, com dados hiperespectrais e com poucas amostras de treinamento disponíveis, percebe-se que os parâmetros não são corretamente estimados, e a acurácia deste classificador decresce conforme vão sendo adicionadas variáveis (bandas).

A metodologia proposta está descrita nos fluxogramas da Figura 10 e Figura 11, respectivamente, para os algoritmos de treinamento e teste, e sucintamente explicitada nos itens abaixo. Para o algoritmo de treinamento do classificador proposto, consideram-se os seguintes passos:

- ✓ todas as amostras de treinamento de todas as classes são atribuídas ao nó raiz;
- ✓ supondo-se os dados normalmente distribuídos, pelo critério distância de Bhattacharyya (Equação 4), escolhe-se o par de classes que, apresentando a distância estatística maior, darão origem aos nós descendentes. Para o cálculo da distância de Bhattacharyya o usuário determinará o número de bandas que serão utilizadas, dentre as disponíveis. Essas bandas serão coletadas a intervalos regulares por todo espectro eletromagnético;
- ✓ determinadas as classes que darão origem aos nós filhos, faz-se uso do algoritmo SFS que tem por objetivo, em cada nó, selecionar o subconjunto de *n* variáveis com maior poder discriminante para este par de classes (SERPICO *et al.,* 2003);
- ✓ neste ponto, usa-se a regra de decisão dos dois classificadores que serão comparados: com o subconjunto de variáveis selecionadas por SFS, calcula-se os coeficientes para a regra de decisão do classificador SVM (Equações 42 e 43) e os parâmetros para o classificador MVG (Equação 12);
- ✓ utilizando-se as respectivas funções de decisão, classifica-se as amostras de treinamento das demais classes em um dos dois nós filhos;
- ✓ neste momento usa-se o conceito de Limiar de Verossimilhança (LV) apresentado por MORAES, (2005). Caso a porcentagem das amostras de treinamento de uma dada classe classificada em um dos nós filhos seja maior que o LV determinado pelo usuário - entre 0 e 100% - todas as amostras serão atribuídas a este nó filho. Caso contrário, as amostras de treinamento desta classe são copiadas em ambos os nós filhos;
- ✓ esse processo será repetido até que cada nó contenha apenas uma classe.

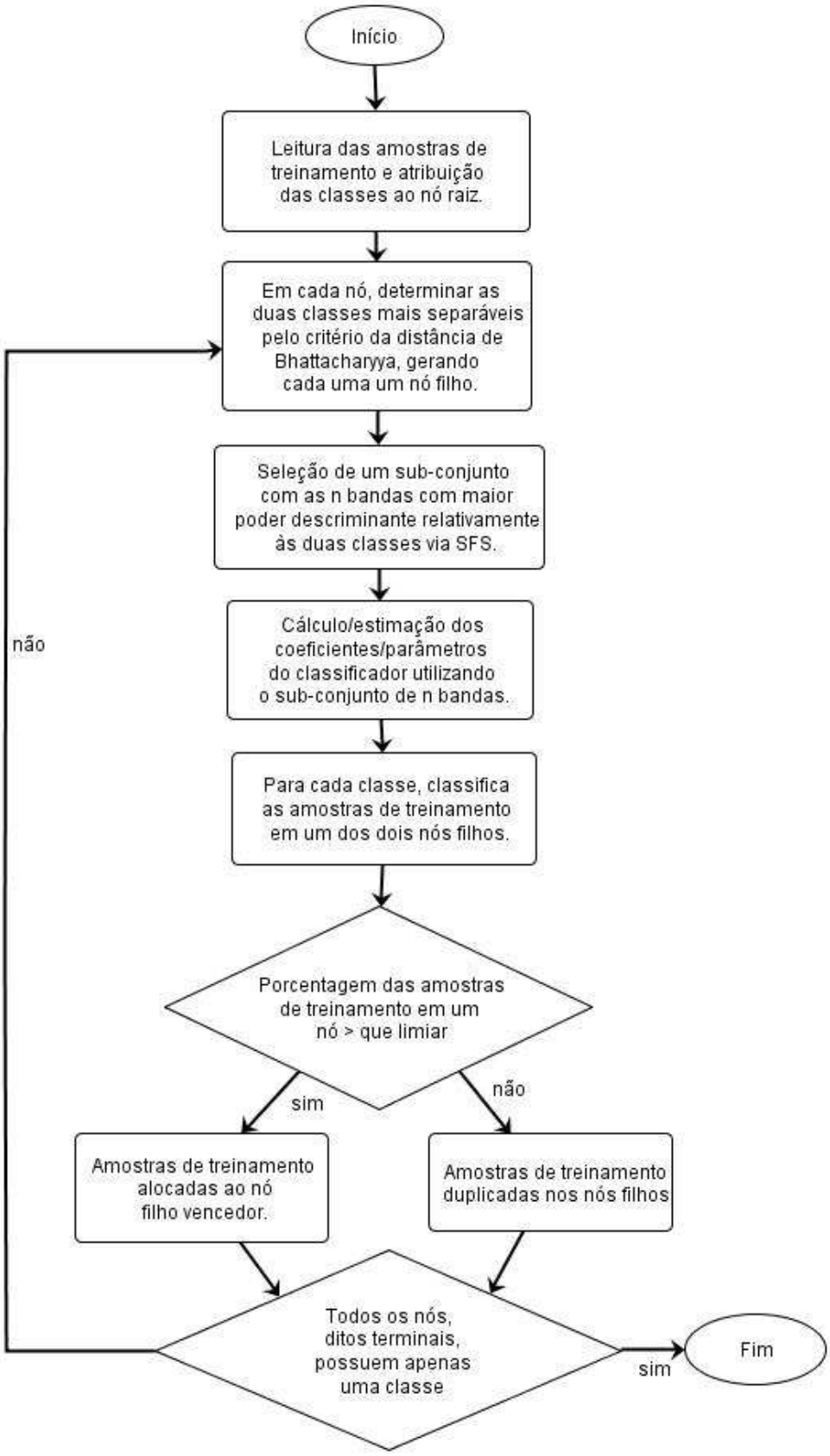


Figura 8: Fluxograma do algoritmo de treinamento do classificador.

Para o algoritmo de teste do classificador, pode-se observar o fluxograma da Figura 9, que obedece aos seguintes passos:

- ✓ todas as amostras de teste são atribuídas ao nó raiz;
- ✓ com base nos parâmetros estimados (caso do classificador MVG) ou nos coeficientes calculados (caso do classificador SVM), e levando-se em conta as variáveis selecionadas (dados obtidos na fase de treinamento), em cada nó decide-se em qual nó filho a amostra de teste será classificada;
- ✓ este processo é repetido para cada amostra, ao longo dos vários níveis na árvore binária, até que um nó terminal seja atingido, atribuindo desta forma um rótulo a cada uma das amostras.

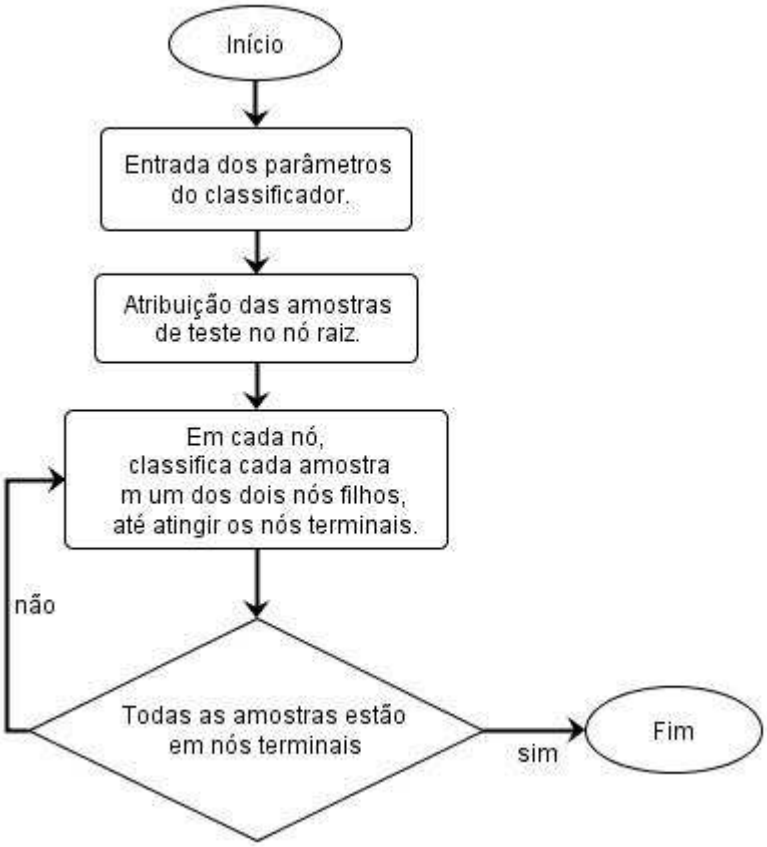


Figura 9: Fluxograma do algoritmo de teste do classificador.

CAPÍTULO IV

TESTES E EXPERIMENTOS

4.1 INTRODUÇÃO

Nesta seção são descritos e discutidos os materiais e os experimentos que tem por finalidade validar a metodologia descrita na seção anterior. Para esta finalidade foi desenvolvido nesta dissertação o aplicativo CAB-SVM, em ambiente MATLAB. Os experimentos foram desenvolvidos empregando dados em alta dimensionalidade (imagens hiperespectrais).

4.2 CENA DE ESTUDO

Nestes experimentos são empregados dados em alta dimensionalidade coletados pelo sistema sensor AVIRIS sobre uma área agrícola de testes, desenvolvida pela Purdue University, e denominada de *Indian Pines*, localizada no noroeste do Estado de Indiana, EUA (Figura 10), sob a denominação de 92AV220.

Figura 10: Localização do Estado de Indiana, nos EUA. Fonte: Mapsofworld.

Da cena 92av220, foi selecionado de um segmento de imagem de (435x435) um recorte de (145x118), num total de 17110 pixels (Figura 11). Esta área dispõe de dados de verdade terrestre conforme ilustrados na imagem temática 92avGT (Figura 13).

O que torna a área atraente para os estudos que empregam dados em alta dimensionalidade é esta possuir classes com características espectrais muito semelhantes entre si e, portanto, difíceis de serem separados com dados tradicionais em baixa dimensionalidade como, por exemplo, dados Landsat-TM.

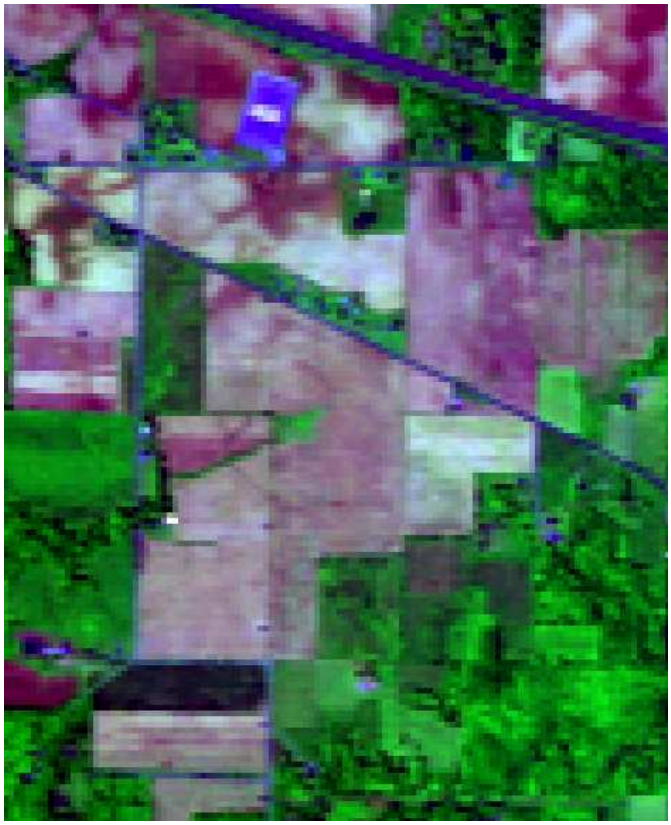


Figura 11: Composição colorida falsa cor RGB 170,80,10 da área de estudo.

4.2.1 Seleção de Bandas Espectrais

Do conjunto de 220 bandas que dispõe a cena AVIRIS, foram removidas as bandas ruidosas causados por problemas atmosféricos (vapor de água, CO_2, O_3). As bandas excluídas podem ser vistas em destaque na Figura 12.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29	30	31	32	33	34	35	36	37	38	39	40	41	42
43	44	45	46	47	48	49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96	97	98
99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126
127	128	129	130	131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150	151	152	153	154
155	156	157	158	159	160	161	162	163	164	165	166	167	168
169	170	171	172	173	174	175	176	177	178	179	180	181	182
183	184	185	186	187	188	189	190	191	192	193	194	195	196
197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224

Figura 12: bandas do sensor AVIRIS, em destaque as bandas ruidosas que fora excluídas para a realização dos experimentos.

O número final de bandas espectrais a ser usada, isto é, a dimensionalidade dos dados é de 190, resultando em vetores com 190 valores de contador digital para cada pixel da imagem. Sobre este subconjunto com 190 bandas espectrais foi aplicado a metodologia SFS para fins de seleção de variáveis, conforme descrito nos experimentos.

4.2.2 Seleção de Classes

A área de estudo apresenta 10 classes de cobertura do solo conforme ilustrado na imagem 92avGT (Figura 13). Para fins dos experimentos foram selecionadas seis classes (Ver Tabela 3).

Tabela 3. Relação das classes usadas nos experimentos.

Classes	Descrição	Amostras Disponíveis
ω_1 – corn min	milho cultivo mínimo	834
ω_2 – corn notill	milho plantio direto	1434
ω_3 – grass trees	pastagens e árvores	747
ω_4 – soybean clean	soja cultivo convencional	614
ω_5 – soybean min	soja cultivo mínimo	2468
ω_6 – soybean notill	soja plantio direto	968

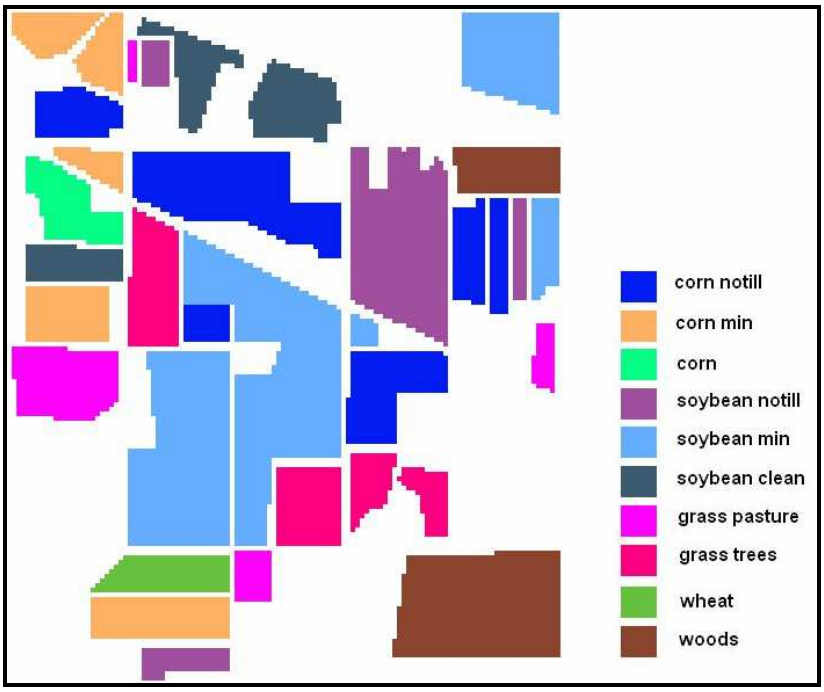
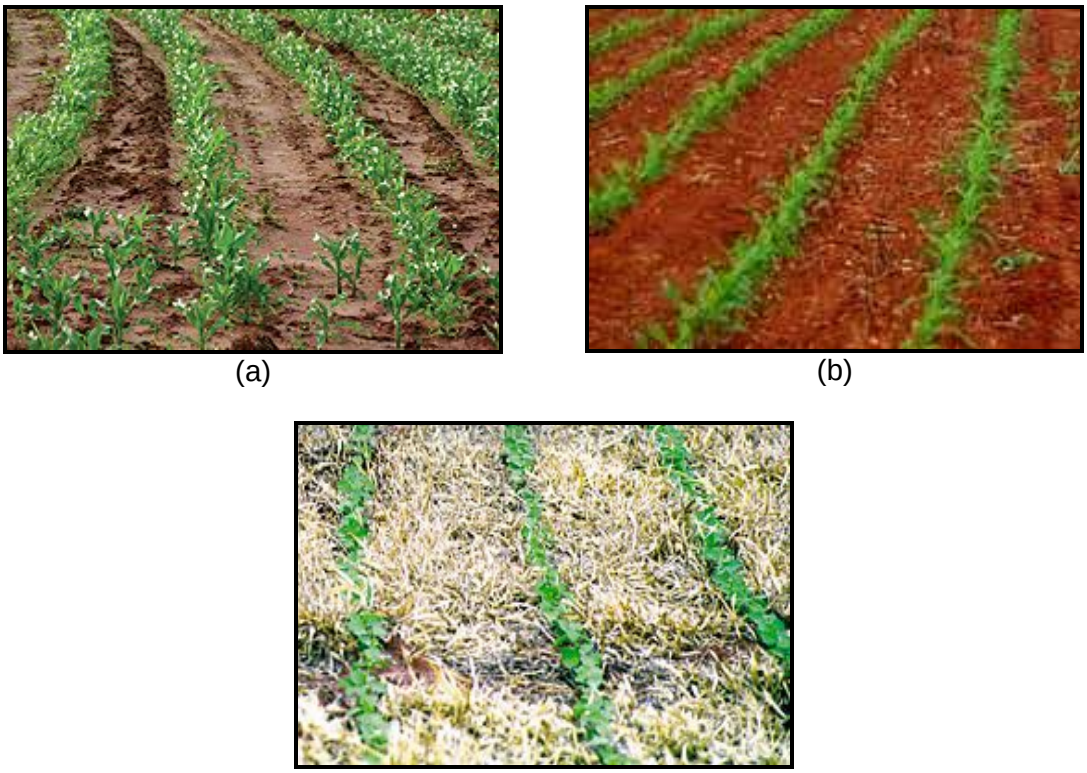


Figura 13: Verdade terrestre referente à cena 92AV3GT.

Dentre as seis classes selecionadas cinco apresentam alta semelhança espectral (vetores de média muito semelhantes entre si) sendo, portanto, de difícil discriminação. Para um melhor entendimento das diferenças entre as classes espectralmente difíceis de serem separadas (diferentes técnicas de cultivo de soja e milho), faz-se, a seguir, uma breve descrição dos diferentes tipos de manejo do solo (EMBRAPA):

- ✓ preparo convencional**:** provoca inversão da camada arável do solo, mediante o uso de arado; a esta operação seguem outras, secundárias, com grade ou cultivador, para triturar os torrões; 100% da superfície são removidos por implementos. Este tipo de preparo é utilizado quando necessária a correção de algumas características na sub-superfície do solo, com a incorporação de corretivos ou rompimento de camadas compactadas (Figura 14a);
- ✓ preparo mínimo: intermediário, que consiste no uso de implementos sobre os resíduos da cultura anterior, com o revolvimento mínimo necessário para o cultivo seguinte (Figura 14b);
- ✓ plantio direto: aqui, as sementes são semeadas através de semeadora especial sobre a palhada (resteva) de culturais do cultivo anterior ou de culturas de cobertura palha produzidas no local para este fim (Figura 14c).



(a) (b)

(c)

Figura 14: Tipos de manejos de Solo. (a) milho cultivo convencional; (b) milho cultivo mínimo; (c) soja cultivo direto. Fonte: EMBRAPA.

A imagem foi obtida no início da época de crescimento das culturas de soja e milho. Nesta etapa apenas aproximadamente 5% da área está efetivamente coberta pela vegetação, sendo os restantes 95% composto por solo exposto e resíduo de colheitas anteriores. Estas condições resultam em classes espectralmente muito semelhantes entre si, constituindo-se por esta razão em um desafio o processo de classificação. A experiência tem mostrado que dados em baixa dimensionalidade, freqüentemente utilizados na classificação de imagens de cenas naturais como, por exemplo, dados Landsat-TM e SPOT, neste caso produzem resultados insatisfatórios.

A classe ‘grass trees’ foi incluída por possuir características espectrais bem diferentes das demais sendo, portanto, facilmente separável das demais classes, servindo de referência no processo de classificação.

As Figuras 15 e 16 ilustram o comportamento espectral médio das classes da Tabela 3, onde se verificam dois aspectos principais: a diferença espectral da classe *pastagens/árvores* (grass trees) com relação às demais classes, e a alta semelhança entre as outras cinco classes (variações das culturas de *milho* e *soja*).

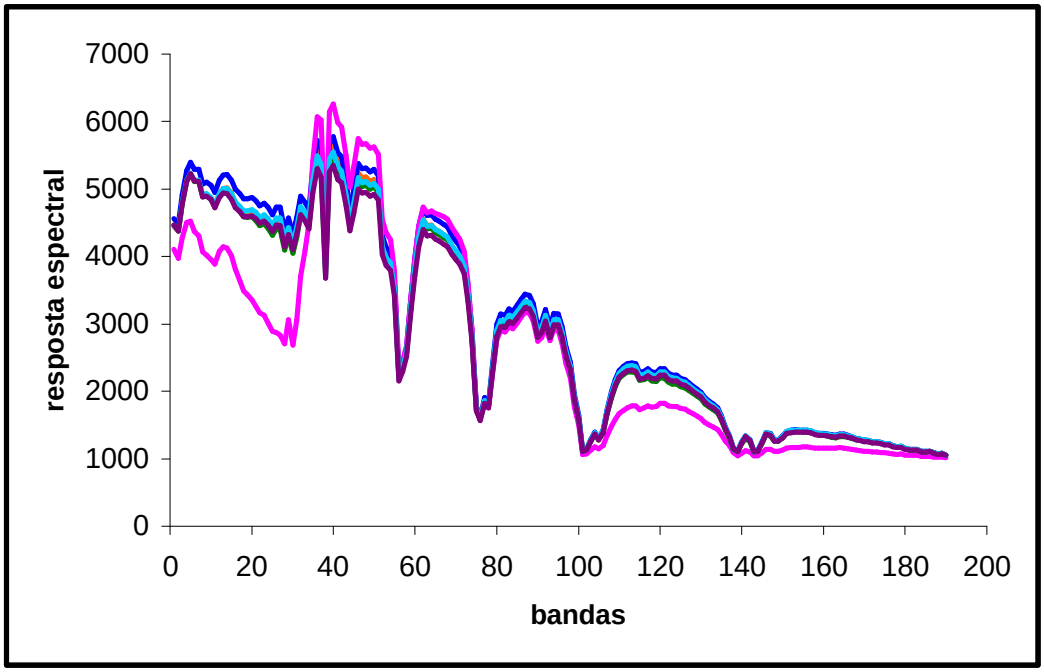


Figura 15: Curvas de resposta espectral média para cada uma das classes: milho cultivo mínimo (laranja), milho plantio direto (azul), pastagens/árvores (rosa), soja cultivo mínimo (ciano), soja plantio direto (bordo), soja cultivo convencional (verde).

Como nos dados utilizados o intervalo numérico de variação dos contadores digitais ao longo do conjunto das bandas espectrais é muito grande, decidiu-se padronizar estes dados (equações 46 e 47) deixando-os com média igual a zero e desvio padrão igual a um (JOHNSON E WICHERN ,1982):

$$\mathbf{Z} = (\mathbf{V}^{1/2})^{-1}(\mathbf{X} - \boldsymbol{\mu}) \qquad \textbf{(46)}$$

onde $\boldsymbol{\mu}$ é o vetor de médias, $\boldsymbol{X}$ é o espaço original, $\boldsymbol{Z}$ é o espaço normalizado e $\boldsymbol{V}^{1/2}$ é dado por:

$$\mathbf{V}^{1/2} = \begin{bmatrix} \sqrt{\sigma_{11}} & 0 & \cdots & 0 \\ 0 & \sqrt{\sigma_{22}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{\sigma_{pp}} \end{bmatrix} \qquad \textbf{(47)}$$

Os resultados deste processo de padronização estão ilustrados na Figura 16. Pode-se perceber que, com a padronização, os padrões das classes ficam significativamente mais separáveis. O código fonte desenvolvido para o procedimento de padronização pode ser encontrado no Apêndice A1.

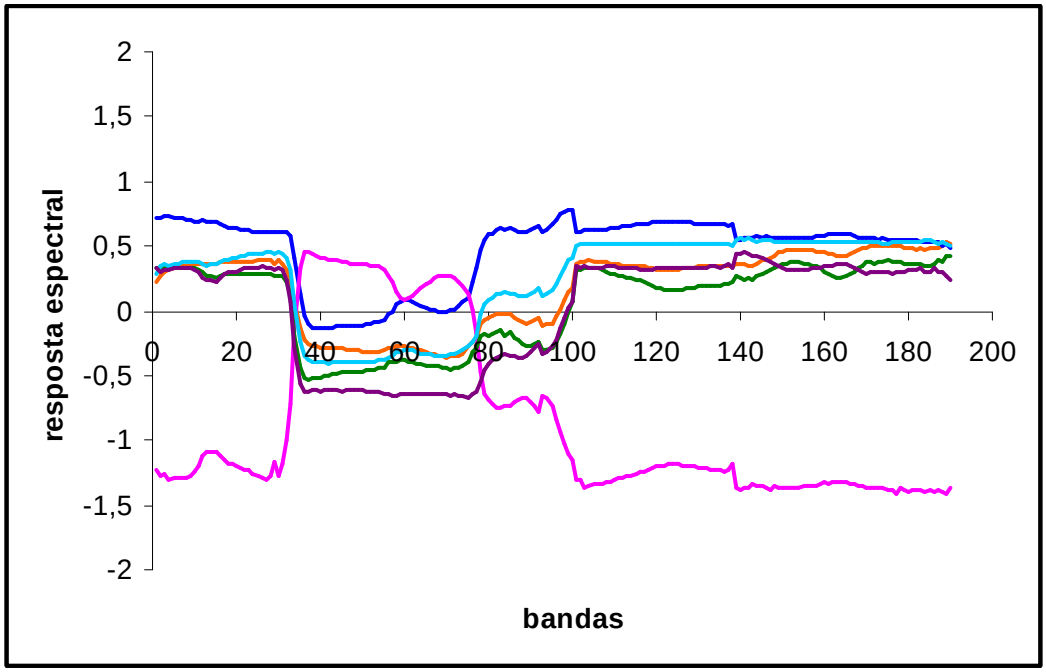


Figura 16: Curvas de resposta espectral média para as classes após a padronização: milho cultivo mínimo (laranja), milho plantio direto (azul), pastagens/árvores (rosa), soja cultivo mínimo (ciano), soja plantio direto (bordo), soja cultivo convencional (verde).

4.2.3 Amostras de Treinamento

Do conjunto das amostras disponíveis para cada classe foram extraídos dois subconjuntos: um com amostras de treinamento e um segundo com amostras de teste. Com a finalidade de capturar as variações naturais que ocorrem ao longo da área coberta pela imagem, as amostras em ambos os subconjuntos foram extraídas alternadamente do conjunto das amostras disponíveis nos dados de verdade terrestre.

Para tornar os resultados obtidos para as várias classes comparáveis entre si, foram utilizados subconjuntos de treinamento e de teste de mesmo tamanho para todas as classes em estudo: inicialmente foram tomadas 50 amostras por classe para treinamento e 300 amostras por classe para teste; em um segundo momento, 99 amostras para treinamento e 300 amostras para teste; em seguida coletou-se 200 amostras para treinamento e outras 300 para fins de teste; e finalmente um quarto conjunto, com 300 amostras de treinamento e 300 amostras de teste.

As amostras de treinamento e teste foram tomadas a intervalos regulares do conjunto total de amostras para cada classe, ou seja, não necessariamente as 50 primeiras amostras estão contidas no conjunto de 99 amostras, e estas não necessariamente estão contidas nas 200 amostras seguintes e assim sucessivamente; as 300 amostras de teste são coletadas da mesma forma e são diferentes para cada

caso. O motivo porque se esta utilizando 99 amostras de treinamento ao invés de 100 na realização dos experimentos será explicado na sub-seção 4.3.2.1.

4.3 FERRAMENTA CAB – CLASSIFICADOR EM ÁRVORE BINÁRIA

Para fins de implementação da metodologia proposta neste estudo, foi desenvolvida uma ferramenta denominada de Classificador em Árvore Binária (CAB), cujo código fonte encontra-se no Apêndice A. O CAB, implementado em forma de árvore binária, possui duas versões, uma para o classificador MVG e outra para o classificador SVM. Desenvolvidos em ambiente MATLAB 6.1, o CAB-MVG e o CAB-SVM apresentam como resultado a Matriz de Confusão.

Os valores de acurácia em cada experimento foram estimados a partir de matrizes de contingência, ou matrizes de confusão. Neste processo, a terminologia empregada é definida a seguir (CONGALTON, 1991):

a) Acurácia do Produtor: é estimada pela fração do número total das amostras de teste fornecidas ao classificador que foram rotuladas corretamente pelo classificador. Esta acurácia estima, portanto, a capacidade de o classificador reconhecer corretamente uma amostra.

b) Acurácia do Usuário: é estimada pela fração das amostras de teste rotuladas pelo classificador em cada uma das classes e que efetivamente pertencem a esta classe. Esta acurácia estima, portanto, o grau de confiança que se pode ter na imagem temática produzida pelo classificador.

c) Acurácia Média: é estimada pela razão do total de amostras classificadas corretamente em cada classe, ou seja, a soma dos valores na diagonal principal na matriz de contingência, pelo número total das amostras de teste.

4.3.1 Experimentos

Os experimentos foram desenvolvidos com o objetivo de quantificar numericamente os resultados de desempenho da metodologia proposta, especialmente no que diz respeito ao comportamento da acurácia no processo de classificação de imagens digitais de sensoriamento remoto, para os diferentes *kernels* e parâmetros implementados na ferramenta CAB-SVM.

Foi realizada uma série de experimentos, tomando-se a dimensionalidade dos dados como variável independente e a resultante acurácia na classificação, em porcentagem, como variável dependente. O valor da dimensionalidade dos dados, isto é, o número de bandas espectrais empregadas, variou entre 20 e 180. Em cada

experimento as bandas espectrais foram selecionadas por meio do algoritmo SFS, a um intervalo de 20 bandas. Desta maneira objetiva-se analisar o comportamento da acurácia produzida pelo classificador SVM em função da dimensionalidade dos dados e dos parâmetros escolhidos. Os resultados assim obtidos são comparados com aqueles obtidos nas mesmas condições, empregando-se um classificador mais tradicional (MVG), implementado pela ferramenta CAB-MVG. Nota-se que o valor mínimo admissível para as amostras de treinamento no caso do CAB-MVG é igual à dimensionalidade dos dados mais um. Um valor inferior resultará em uma matriz de covariância singular e, portanto, não utilizável (LANDGREBE, 2003).

O tamanho das amostras de treinamento foi escolhido deliberadamente pequeno com relação à dimensionalidade dos dados para desta forma melhor evidenciar os problemas que ocorrem em situações reais, ou seja, o pequeno número de amostras de treinamento normalmente disponíveis. Para os experimentos realizados empregando a ferramenta CAB-SVM (com 50, 99, 200 e 300 amostras de treinamento e 300 amostras de teste para cada caso), foram usadas 80 bandas para o cálculo da distância de Bhattacharyya e LV de 99%. Decidiu-se fixar o LV em 99% para que fosse obtida sempre a maior estrutura possível, ou seja, o número máximo de nós terminais (MORAES, 2005). Ainda segundo o autor, valores mais altos para o LV produzem, uma menor variabilidade no valor estimado da acurácia de cada classe individual, em função da dimensão dos dados.

Para a estimação do multiplicadores de Lagrange (Equação 40) foi utilizado a função quadprog.m disponível em MATLAB® e em todos os casos C foi tomado igual a 1. As Tabelas (4 - 12) e Figuras (17 - 24) abaixo mostram o comportamento da acurácia média variando-se o grau do polinômio, no caso do uso do Kernel Polinomial (Equação 45), e variando-se o parâmetro gamma (γ) no caso do Kernel RBF (Equação 44).

4.3.1.1 Experimento 1

Neste experimento, fez-se uso do Kernel Polinomial, variando-se o grau do polinômio de 1 à 4. Na Figura 17 e Tabela 4 encontram-se os resultados para a acurácia média com o kernel linear, ou seja, kernel polinomial grau 1. Nas tabelas 5, 6 e 7 e Figuras 18, 19 e 20, estão as acurácias médias para o kernel polinomial grau 2, 3 e 4 respectivamente.

Tabela 4: Acurácia Média para kernel polinomial grau 1.

Dimensionalidade dos Dados (bandas)	50 amostras de treinamento	99 amostras de treinamento	200 amostras de treinamento	300 amostras de treinamento
20	57,6	68,3	76	71,6
40	69,2	69,5	70,2	73,6
60	78,5	61,8	63,7	74,7
80	78,9	63,1	64,2	70,1
100	79,1	72,3	64,6	70,5
120	79,6	72,5	65,2	71,5
140	80,1	72,1	64,3	71,8
160	79,2	72,7	65,4	71,6
180	79,3	72,6	65,1	71,9

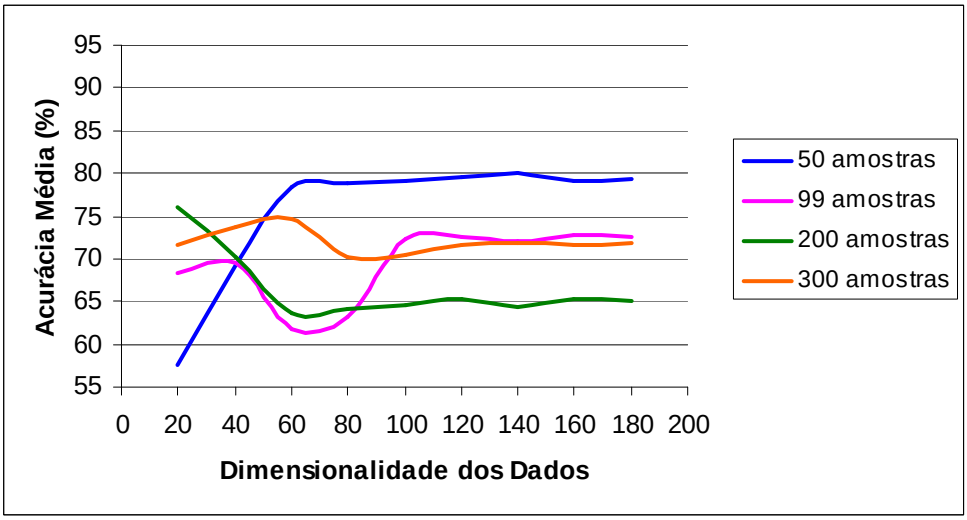


Figura 17: Acurácia Média para kernel polinomial grau 1.

Comparando os resultados ilustrados na Figura 17 e na Tabela 4 com os resultados correspondentes produzidos por kernels polinomiais de grau mais elevado (experimentos seguintes), observa-se que o kernel linear é o que apresenta os piores resultados com relação a acurácia média. Tais resultados podem ser explicados pelo fato de que o kernel linear não mapeia os dados em uma dimensão maior, apenas determina um plano de separação entre as classes. E como se pode notar, os dados não são linearmente separáveis. Aumentando o grau do polinômio e, conseqüentemente, ocorrendo o mapeamento dos dados em uma dimensionalidade mais alta, os resultados melhoram significativamente como demonstram as figuras e tabelas abaixo.

Tabela 5: Acurácia Média para kernel polinomial grau 2.

Dimensionalidade dos Dados (bandas)	50 amostras de treinamento	99 amostras de treinamento	200 amostras de treinamento	300 amostras de treinamento
20	84,9	66,2	90,2	76,1
40	84,4	80,8	90	82,6
60	85	81,1	89,5	83,2
80	83,5	81	90,3	83
100	84,9	80,7	89,6	83,8
120	85,6	81,1	89,8	83,4
140	85,8	80,7	90,2	83,8
160	86,2	80,1	89,9	83,9
180	86,2	80,2	89,7	90,3

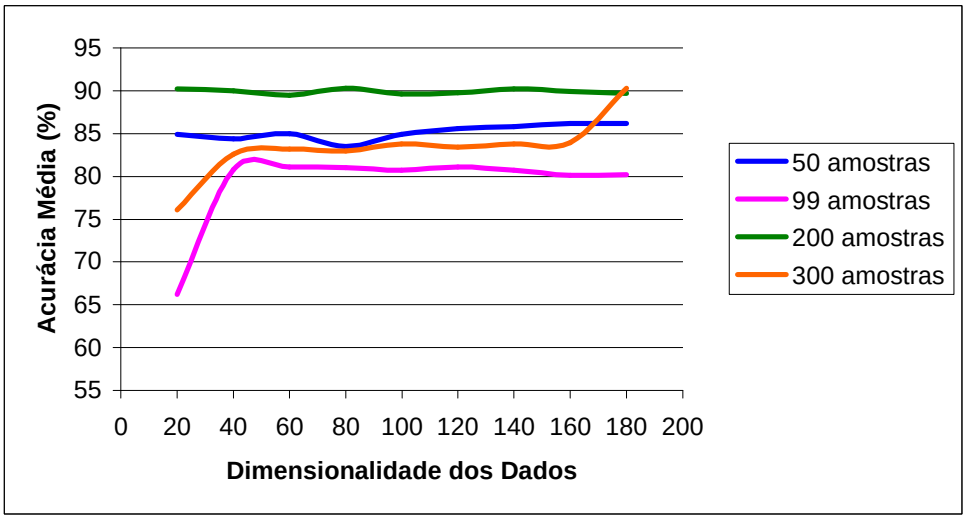


Figura 18: Acurácia Média para kernel polinomial grau 2.

Tabela 6: Acurácia Média para kernel polinomial grau 3.

Dimensionalidade dos Dados (bandas)	50 amostras de treinamento	99 amostras de treinamento	200 amostras de treinamento	300 amostras de treinamento
20	76	82,6	90,4	89,9
40	84,6	81,1	89,9	90,2
60	84	81,3	89,4	89,9
80	84,5	81,7	89,5	89,8
100	83,9	89,1	89,4	89,7
120	84,3	81	90,6	90,4
140	84,2	81	89,7	90,3
160	85,6	81,2	90,2	89,9
180	85,2	81,7	90,4	90,1

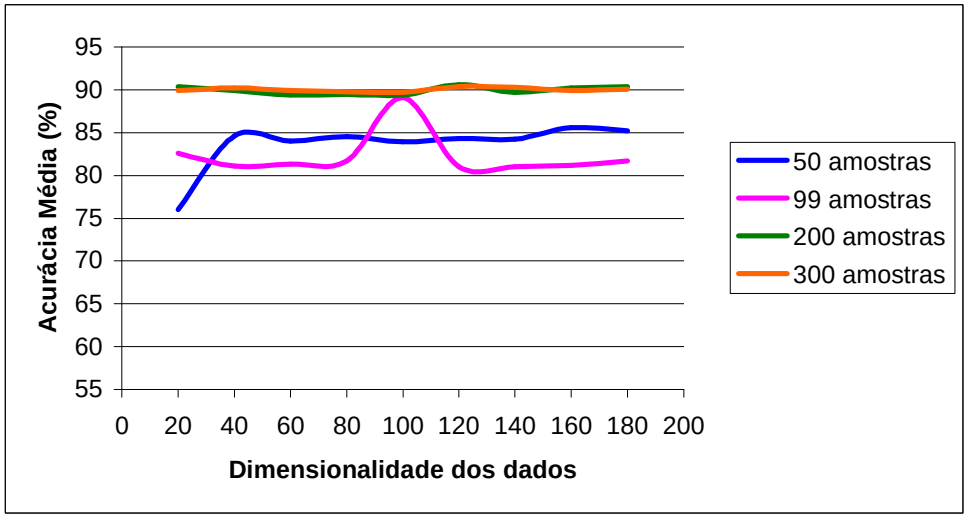


Figura 19: Acurácia Média para kernel polinomial grau 3.

Analisando as Figuras 18 e 19 pode-se perceber que os melhores resultados são os obtidos com 200 amostras de treinamento. Para o conjunto de 300 amostras de treinamento obteve-se melhores resultados com o kernel polinomial grau 3, cujas acurácias médias são semelhantes às obtidas com 200 amostras.

Tabela 7: Acurácia Média para kernel polinomial grau 4.

Dimensionalidade dos Dados (bandas)	50 amostras de treinamento	99 amostras de treinamento	200 amostras de treinamento	300 amostras de treinamento
20	75,9	81,4	88,8	89
40	82,9	79,1	88,6	89
60	83,1	79,1	87,4	87,7
80	81,6	77,6	87,5	86,9
100	82	77,2	87,2	86,8
120	81,8	76,3	87,2	86,1
140	81,7	84,5	85,8	85,4
160	82,6	76,9	85,6	85,3
180	82,9	83,9	85,5	85,6

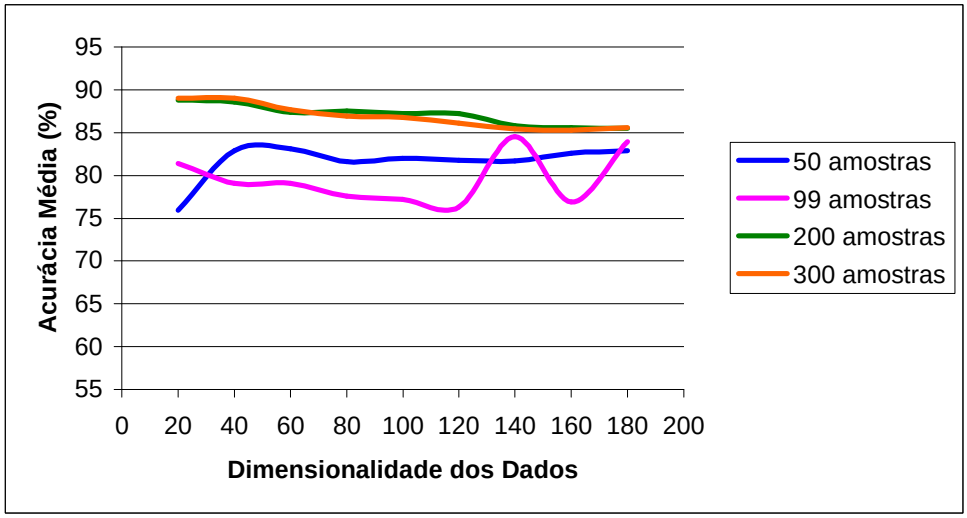


Figura 20: Acurácia Média para kernel polinomial grau 4.

Fato interessante a ser observado é que, no geral, para o kernel polinomial independente do grau adotado, as acurácias obtidas com 50 amostras de treinamento são superiores às obtidas com 99 amostras de treinamento.

Como foi dito anteriormente, para o desenvolvimento dos experimentos, as amostras de treinamento utilizadas são selecionadas uniformemente do conjunto contendo a totalidade das amostras disponíveis. Com esta abordagem busca-se selecionar para cada experimento amostras distribuídas o mais uniformemente possível ao longo de toda a área teste. Espera-se assim obter uma melhor representatividade na distribuição de cada uma das classes, evitando tendenciosidades que poderiam surgir ao concentrar-se as amostras em segmentos da imagem. Desta abordagem resulta que em cada um dos experimentos as amostras individuais diferem substancialmente entre si, isto é, o conjunto das amostras no experimento com 50 amostras de treinamento não será um subconjunto daquele utilizado no experimento com 99 amostras, com uma situação semelhante acontecendo em todos os demais experimentos. Sabe-se, por outro lado, que em imagens de cenas naturais algumas amostras de treinamento apresentam variações maiores em seus atributos (*outliers*), comportando-se de certa forma quase que como amostras ruidosas. Esta situação, na fase de treinamento, pode comprometer o poder de generalização do classificador resultando nas variações visíveis no comportamento da acurácia nos vários experimentos conforme ilustrado nas figuras acima. Esta suposição é analisada em maior detalhe na seção 4.3.2.1. Portanto, a causa provável para este fenômeno (acurácias superiores para 50 amostras de treinamento) seja a presença de ruído no conjunto de 99 amostras de treinamento.

Com o kernel polinomial grau 4 (Tabela 7 e Figura 20) a acurácia média passa a cair para os 4 conjuntos de treinamento, o que pode ser interpretado como um possível excesso de ajustes, ou seja, o fenômeno conhecido como *overffiting*.

4.3.1.2 Experimento 2

Para a execução deste experimento, fez-se uso do Kernel RBF, variando-se o parâmetro γ de 0.5 à 2. Na Figura 21 e Tabela 8 encontram-se os resultados para a acurácia média com o kernel RBF, com γ 0.5. Nas tabelas 9, 10 e 11 e Figuras 22, 23 e 24, estão as acurácias médias para o kernel RBF com γ igual a 1, 1.5 e 2 respectivamente.

Tabela 8: Acurácia Média para kernel RBF γ 0.5.

Dimensionalidade dos Dados (bandas)	50 amostras de treinamento	99 amostras de treinamento	200 amostras de treinamento	300 amostras de treinamento
20	75,8	64,3	89,6	85,2
40	85,3	71,3	90,4	84,9
60	85,4	79,1	90,1	80,3
80	85,8	79,7	90,4	83,7
100	85,7	86,7	90,2	89,6
120	85,8	87,4	89,9	89,6
140	86	87,4	89,5	89,4
160	85,8	87,4	89,7	88,9
180	86,1	87,8	89,7	88,7

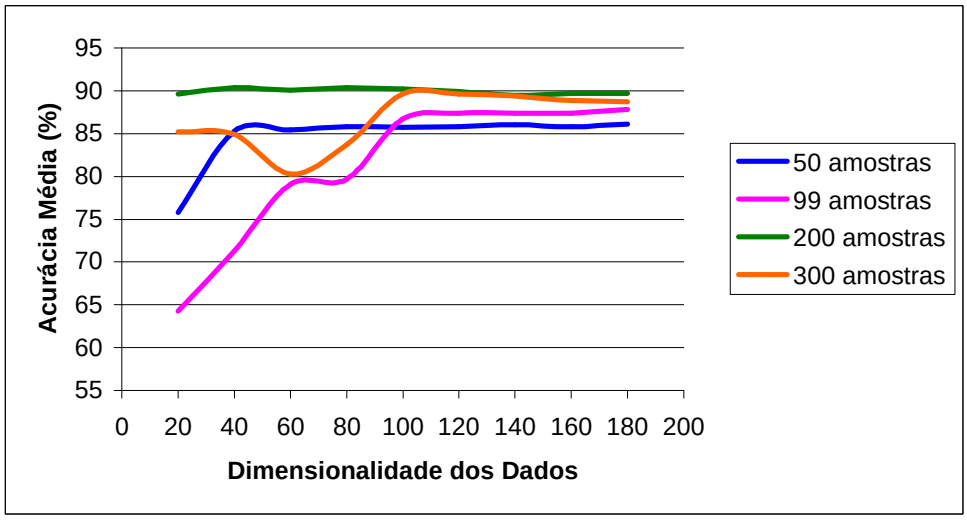


Figura 21: Acurácia Média para kernel RBF γ 0.5.

Percebe-se que, para o kernel RBF γ igual a 0.5, quando a dimensionalidade dos dados é baixa acontece alguma flutuação na acurácia principalmente para os

subconjuntos de 99 e 300 amostras de treinamento, como pode ser observado nas Figura 21. Essa flutuação acontece provavelmente, como explicado anteriormente, devido à presença de amostras ruidosas em tais conjuntos de treinamento.

Tabela 9: Acurácia Média para kernel RBF γ 1.

Dimensionalidade dos Dados (bandas)	50 amostras de treinamento	99 amostras de treinamento	200 amostras de treinamento	300 amostras de treinamento
20	84,9	65,7	90,2	80,2
40	85,1	79,7	90,4	85,1
60	85,2	86,9	89,9	89,7
80	85,6	87,3	89,8	88,9
100	85,6	87,6	89,6	89,1
120	85,3	87,8	89,5	88,7
140	85,8	87,8	88,9	88,2
160	85,9	88,1	88,8	88,2
180	85,9	88,3	88,9	88,7

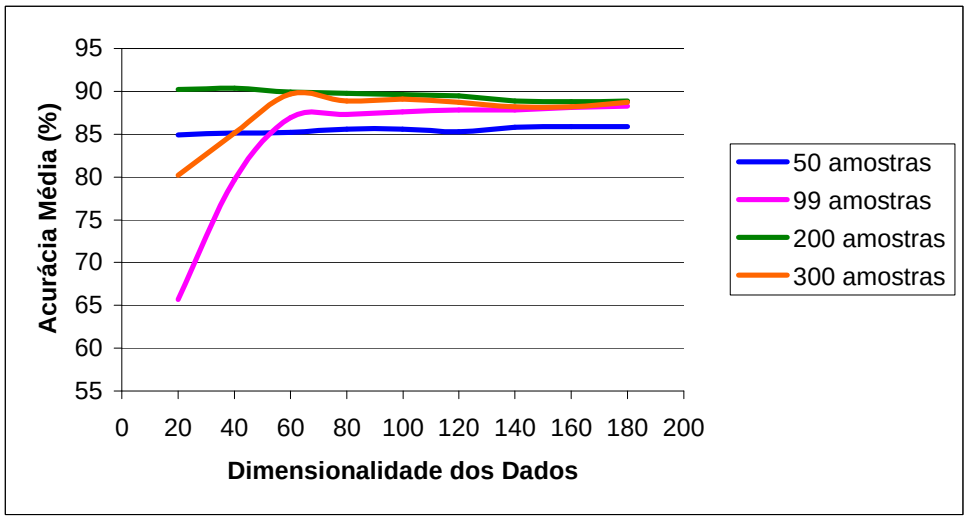


Figura 22: Acurácia Média para kernel RBF γ 1.

Tabela 10: Acurácia Média para kernel RBF γ 1.5.

Dimensionalidade dos Dados (bandas)	50 amostras de treinamento	99 amostras de treinamento	200 amostras de treinamento	300 amostras de treinamento
20	85,1	72,7	90,1	86,7
40	84,7	87,5	89,6	90,7
60	85,3	86,9	89,1	89,3
80	85,9	87,3	89,5	88,6
100	85,3	87,8	89,3	88,4
120	84,6	88,2	88,9	88,4
140	84,7	88	88,7	88,2
160	84,4	88	88,4	88,4
180	84,7	88,2	88,4	88,9

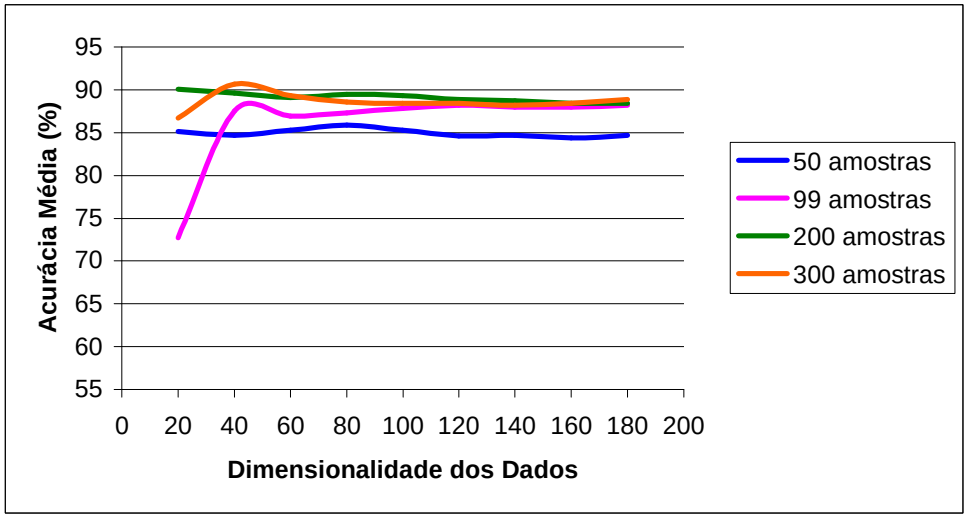


Figura 23: Acurácia Média para kernel RBF γ 1.5.

Tabela 11: Acurácia Média para kernel RBF γ 2.

Dimensionalidade dos Dados (bandas)	50 amostras de treinamento	99 amostras de treinamento	200 amostras de treinamento	300 amostras de treinamento
20	85,2	80,4	89,9	86,8
40	84,3	87,6	89,1	90,4
60	85,2	86,9	88,8	88,8
80	84,8	87,8	88,9	88,1
100	84,3	87,7	88,7	88,2
120	83,8	88,3	88,6	88,4
140	83,6	87,8	88	87,8
160	83,3	87,7	87,7	88,2
180	83,1	87,5	87,8	88,7

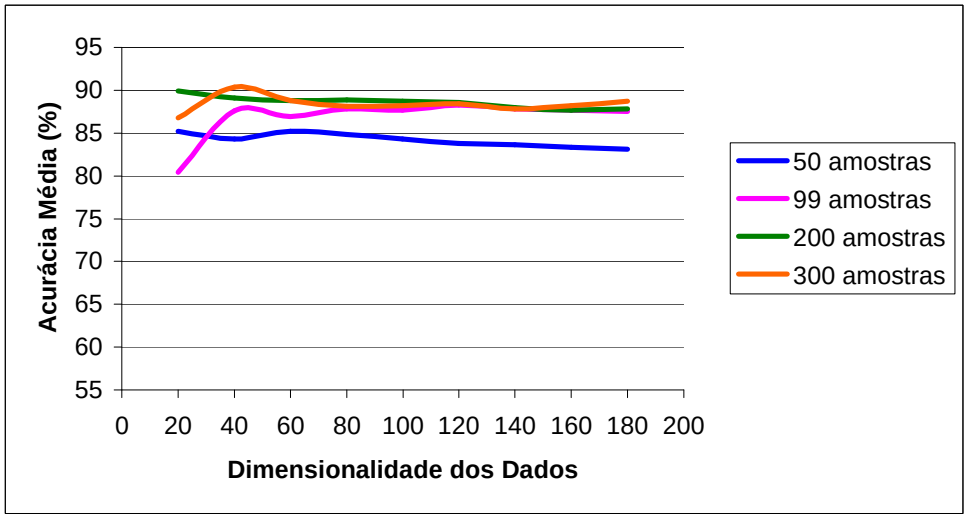


Figura 24: Acurácia Média para kernel RBF γ 2.

Em todos os experimentos realizados para o kernel RBF percebe-se que com a inclusão de informação (aumento do número de bandas), a acurácia tende a aumentar para todos os conjuntos de treinamento até atingir um patamar máximo. A queda

suave na acurácia para um número grande de bandas acontece devido à inclusão de informação ruidosa. Assim como acontece com o kernel polinomial, o kernel RBF obtêm as melhores acurácias com 200 amostras de treinamento.

Pode-se perceber que, mesmo com o aumento do número de amostras de treinamento de 200 para 300, a acurácia média não se eleva substancialmente (com o uso do CAB-SVM); pelo contrário, na maioria dos casos a acurácia média para 300 amostras de treinamento é levemente menor ou igual que aquela para 200. Isso acontece porque o poder máximo de generalização do classificador é atingido com 200 amostras de treinamento, ou seja, o conjunto de 200 amostras representa bem as características de cada classe, e o incremento para 300 apresenta o risco de aumento no número de amostras ruidosas conforme mencionado acima, e com um reduzido acréscimo de informação. A partir da análise dos resultados obtidos supõe-se que o plano de separação determinado para ambos os conjuntos de treinamento situam-se muito próximos.

4.3.1.3 Experimento 3 – Máxima Verossimilhança Gaussiana

Nos três segmentos desenvolvidos nestes experimentos (50, 99, 200 e 300 amostras de treinamento e 300 amostras de teste), foram empregadas 80 bandas na estimação da distância de Bhattacharyya, e um valor de 99% para o limiar LV. Os resultados obtidos com esta ferramenta (CAB-MVG) serão usados como comparação com os resultados obtidos com o CAB-SVM. A Tabela 12 e a Figura 25 apresentam a acurácia média.

Tabela 12: Acurácia Média para classificador Máxima Verossimilhança Gaussiana.

Dimensionalidade dos Dados (bandas)	50 amostras de treinamento	99 amostras de treinamento	200 amostras de treinamento	300 amostras de treinamento
10	84,1	79,9	78,9	76
20	82,9	89	89,2	82,7
40	77,1	83,2	88,4	90,4
60		79,8	86,2	89,6
80		72,1	84,9	89,3
100			82,6	87,2
120			79,3	86,6
140			75,4	84,5
160			71,2	83,3
180			64,7	83,1

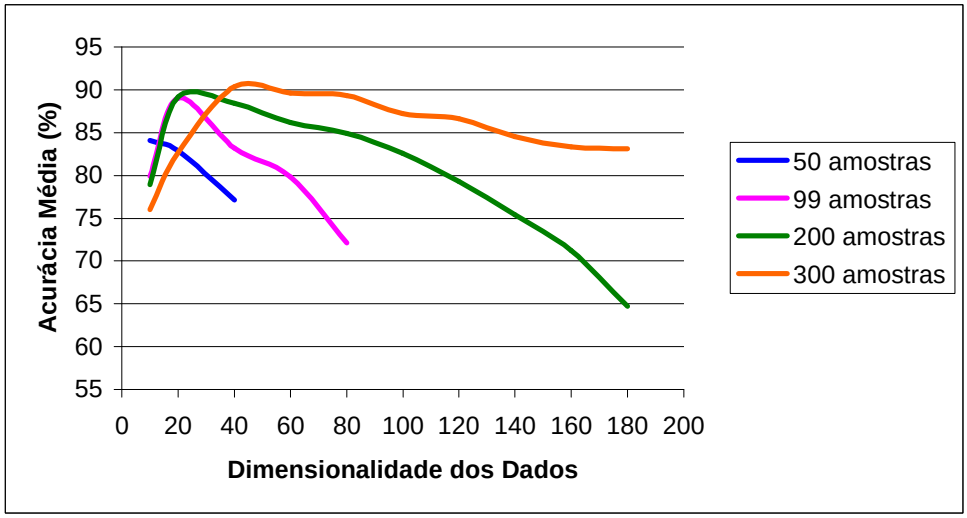


Figura 25: Acurácia Média para classificador Máxima Verossimilhança Gaussiana.

Analisando a Figura 25 pode-se perceber claramente os efeitos do fenômeno de Hughes. Na medida em que a dimensionalidade dos dados aumenta, aumenta também o número de parâmetros a serem estimados pelas mesmas amostras de treinamento. Esta situação resulta em estimativas cada vez menos confiáveis e que, por sua vez, causam uma degradação na performance do classificador paramétrico. Nota-se, como era de se esperar, que para 300 amostras de treinamento o efeito do fenômeno de Hughes é mais suave.

Como mencionado anteriormente, o número mínimo de amostras de treinamento necessárias para a realização dos experimentos com o classificador MVG é igual à dimensionalidade dos dados (número de bandas) mais um. Tendo isto em vista, para 50 amostras de treinamento só foi possível realizar os experimentos com até 40 bandas e para 99 amostras de treinamento até 80 bandas. Com a seleção de 60 e 100 bandas respectivamente, a matriz de covariância se torna singular e, portanto, impossibilita a execução dos experimentos.

O uso do algoritmo SFS (realiza seleção de variáveis, isto é, a escolha das bandas espectrais que mais contribuem para a separação entre as classes) é o responsável pelos altos picos de acurácia média obtidas pelo CAB-MVG. De outra forma os picos de acurácia estariam mais baixos. Pode-se notar ainda que os picos de acurácia para 50, 99, 200 e 300 amostras de treinamento estão em dimensionalidades diferentes, ou seja, 84,1% em 10 bandas para 50 amostras de treinamento, em 20 bandas 89% para 99 amostras e 89,2% para 200 amostras e 90,4% em 40 bandas para 300 amostras de treinamento, refletindo a confiabilidade nos valores estimados para os parâmetros do classificador.

4.3.2 Comparação entre Resultados

Para fins de comparação, resolveu-se reunir os melhores resultados obtidos com o CAB-SVM (com os kernels polinomial e RBF) e com o CAB-MVG para 50, 99, 200 e 300 amostras de treinamento. Estes resultados estão ilustrados nas Figuras 26, 27, 28 e 29 e nas Tabelas 13, 14, 15 e 16, respectivamente. As tabelas de contingência para todos estes experimentos são apresentadas no Apêndice B.

Tabela 13: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 2 e RBF γ 1.5 para 50 amostras de treinamento

Dimensionalidade dos Dados (bandas)	MVG	Poly grau 2	RBF γ 1.5
10	84,1	76,1	83,2
20	82,9	84,9	85,1
40	77,1	84,4	84,7
60		85	85,3
80		83,5	85,9
100		84,9	85,3
120		85,6	84,6
140		85,8	84,7
160		86,2	84,4
180		86,2	84,7

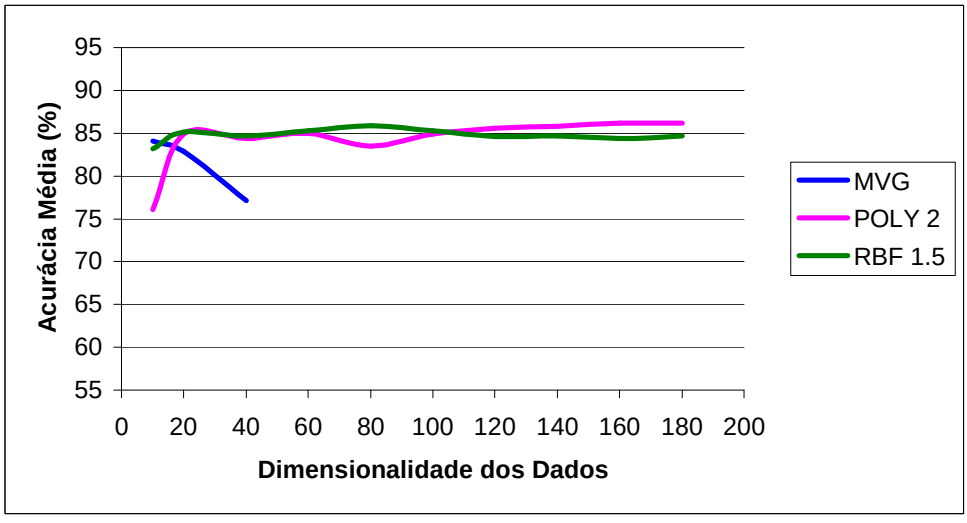


Figura 26: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 2 e RBF γ 1.5 para 50 amostras de treinamento

O baixo número de amostras de treinamento e a impossibilidade da realização dos experimentos com mais de 40 bandas para o CAB-MVG confere à esta comparação especial atenção. Quando o número de amostras de treinamento disponíveis é limitado, fato que ocorre com freqüência em situações reais, o classificador SVM (não paramétrico) se mostra mais eficaz do que o classificador

paramétrico MVG, fortemente afetado pelos efeitos limitantes do fenômeno de Hughes. Com o aplicativo CAB-SVM, utilizando o kernel Polinomial grau 2, o pico de acurácia acontece em 160 e 180 bandas (ou seja, com praticamente toda informação disponível) com 86,2%; utilizando o kernel RBF γ 1.5 o pico acontece em 80 bandas com 85,9%, enquanto que com a ferramenta CAB-MVG o pico acontece em 10 bandas com 84,1% de acurácia média. A diferença a favor do CAB-SVM de 1.8% a 2.1%, para os melhores casos de ambos os classificadores, é considerada significativa, ou seja, pode-se considerar o CAB-SVM melhor que o CAB-MVG para 50 amostras de treinamento.

Tabela 14: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 3 e RBF γ 2 para 99 amostras de treinamento

Dimensionalidade dos Dados (bandas)	MVG	Poly grau 3	RBF γ 2
10	79,9	71,8	79,1
20	89	82,6	80,4
40	83,2	81,1	87,6
60	79,8	81,3	86,9
80	72,1	81,7	87,8
100		89,1	87,7
120		81	88,3
140		81	87,8
160		81,2	87,7
180		81,7	87,5

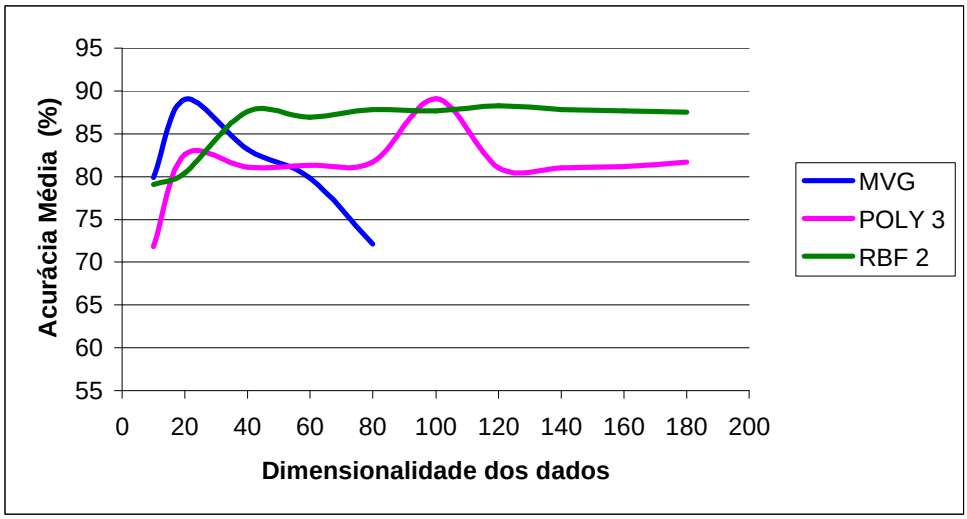


Figura 27: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 4 e RBF γ 2 para 99 amostras de treinamento

Excetuando-se o caso de 100 bandas que atinge um pico de 89,1% de acurácia (Tabela B31), a ferramenta CAB-SVM com kernel polinomial grau 3 apresenta acurácia média baixa, provavelmente causado, como já foi dito anteriormente, pela presença de amostras de treinamento ruidosas neste conjunto. O uso do kernel RBF γ 2 no aplicativo CAB-SVM apresenta, por outro lado, os melhores resultados médios, chegando a 88,3% em 120 bandas.

A acurácia média obtida pelo aplicativo CAB-MVG, para 99 amostras de treinamento, chega a 89% na dimensionalidade 10 (10 bandas selecionadas por SFS). A partir daí a acurácia média passa a cair vertiginosamente pois, com poucas amostras de treinamento, as estimativas para os parâmetros do classificador MVG ficam menos confiáveis (fenômeno de Hughes).

Tabela 15: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 3 e RBF γ 0.5 para 200 amostras de treinamento

Dimensionalidade dos Dados (bandas)	MVG	Poly grau 3	RBF γ 0.5
10	78,9	89,2	83,2
20	89,2	90,4	89,6
40	88,4	89,9	90,4
60	86,2	89,4	90,1
80	84,9	89,5	90,4
100	82,6	89,4	90,2
120	79,3	90,6	89,9
140	75,4	89,7	89,5
160	71,2	90,2	89,7
180	64,7	90,4	89,7

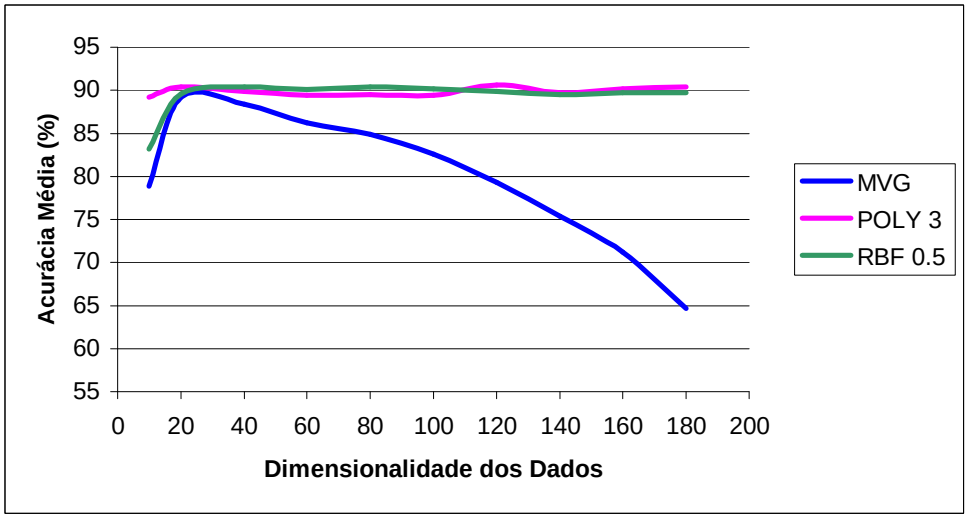


Figura 28: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 3 e RBF γ 0.5 para 200 amostras de treinamento

Com 200 amostras de treinamento, a ferramenta CAB-MVG (que implementa o classificador MVG) apresenta pico de acurácia média em 89,2% em 20 bandas. Com respeito à ferramenta CAB-SVM, o uso de ambos os kernels apresentam bons resultados, com 90,6% em 120 bandas e 90,4% em 20 e 180 bandas para o kernel polinomial grau 3 e 90,4% em 40 e 80 bandas para o kernel RBF γ 0.5. Note-se novamente no caso do classificador paramétrico MVG os efeitos limitantes causados pelo fenômeno de Hughes.

A diferença a favor do CAB-SVM de 1.2% a 1.4%, para os melhores casos de ambos os classificadores, é considerada significativa, ou seja, pode-se considerar o CAB-SVM melhor que o CAB-MVG para 200 amostras de treinamento.

Tabela 16: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 3 e RBF γ 1.5 para 300 amostras de treinamento

Dimensionalidade dos Dados (bandas)	MVG	Poly grau 3	RBF γ 1.5
10	76	80,7	80
20	82,7	89,9	86,7
40	90,4	90,2	90,7
60	89,6	89,9	89,3
80	89,3	89,8	88,6
100	87,2	89,7	88,4
120	86,6	90,4	88,4
140	84,5	90,3	88,2
160	83,3	89,9	88,4
180	83,1	90,1	88,9

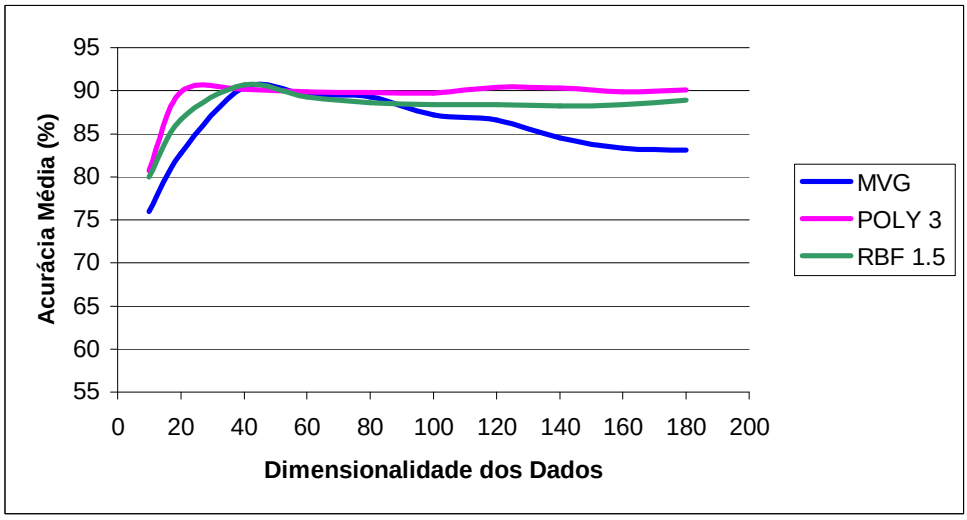


Figura 29: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 3 e RBF γ 1.5 para 300 amostras de treinamento

Para 300 amostras de treinamento os resultados são semelhantes aos obtidos com 200 amostras. No geral os resultados da acurácia média são inferiores aos obtidos com 200 amostras, entretanto, os picos de acurácia são superiores: 90,4% pra o CAB-MVG em 40 bandas, e para o CAB-SVM com o kernel polinomial grau 3 90,4% em 120 bandas e com o kernel RBF γ 1.5 90,7% em 40 bandas.

Analisando as Figuras 26 à 29, que se referem à comparação entre os classificadores para os quatro conjuntos de amostras de treinamento, percebe-se que classificadores paramétricos como o implementado no aplicativo CAB-MVG (MVG), sofre os efeitos do fenômeno de Hughes, ou seja, para um número limitado de amostras de treinamento, seus parâmetros se tornam pouco confiáveis com o acréscimo do número de bandas, causando degradação em sua performance; por outro lado, as acurácias obtidas pelo classificador proposto (CAB-SVM – não-paramétrico), de um modo geral, após atingirem um patamar sofrem pouca variação na acurácia média.

4.3.2.1 Comparação de resultados para conjuntos de amostras de treinamento aproximadamente iguais

Como se pode perceber ao longo da seção Experimentos, utilizou-se um subconjunto de 99 amostras de treinamento ao invés de 100, que seria o naturalmente escolhido diante da escolha dos demais subconjuntos. Para este experimento, selecionou-se para comparação três subconjuntos como amostras de treinamento: 99 amostras de treinamento (resultados apresentados nas seções anteriores), em um segundo momento 100 amostras de treinamento e finalmente 101 amostras de treinamento; o conjunto de teste composto de 300 amostras é idênticos para os três casos. Note que as primeiras 99 amostras de treinamento não estão necessariamente contidas nas 100 amostras seguintes, e estas não estão contidas no subconjunto de 101 amostras, ou seja, são subconjuntos de tamanho aproximadamente igual, mas as amostras contidas em cada caso são essencialmente distintas. Deste modo, as diferenças nas acurácias médias apresentadas nas tabelas e figuras abaixo se deve exclusivamente à diferença na determinação do plano de separação de acordo com a seleção de bandas realizadas e do conjunto de amostras de treinamento em questão e não do número de amostras. Realizou-se os experimentos utilizando a ferramenta CAB-SVM com o kernel polinomial grau 2 (Figura 30 e Tabela 17) e com o kernel RBF γ 1 (Figura 31 e Tabela 18).

Tabela 17: Acurácia média para CAB-SVM utilizando o kernel polinomial grau 2, com o mesmo conjunto de amostras de teste

Dimensionalidade dos Dados (bandas)	99 amostras de treinamento	100 amostras de treinamento	101 amostras de treinamento
20	66,2	76,3	74,2
40	80,8	78,5	72,5
60	81,1	77,7	72,6
80	81	79,4	71,7
100	80,7	79,4	62,8
120	81,1	70,9	71,3
140	80,7	72,9	72,1
160	80,1	72,6	71,6
180	80,2	71,9	71,1

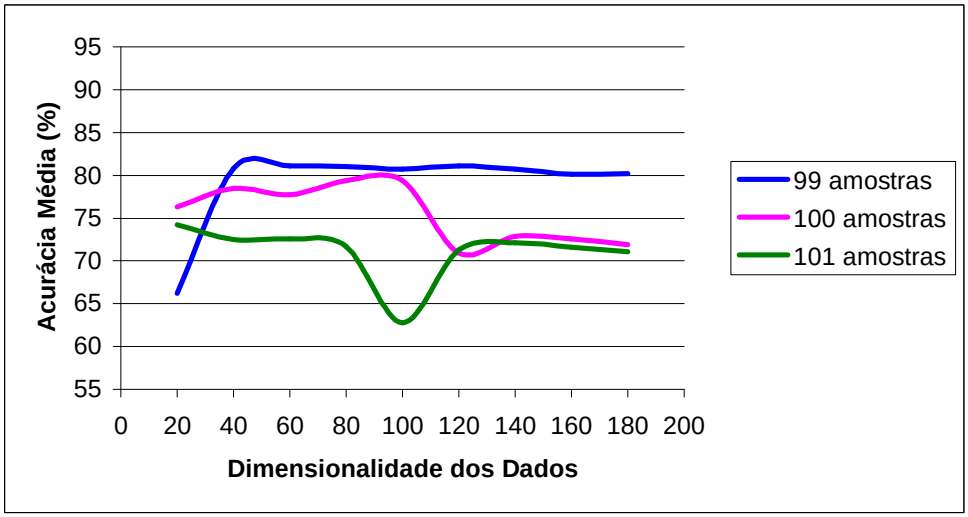


Figura 30: Acurácia média para CAB-SVM utilizando o kernel polinomial grau 2, com o mesmo conjunto de amostras de teste

Tabela 18: Acurácia média para CAB-SVM utilizando o kernel RBF γ 1, com o mesmo conjunto de amostras de teste

Dimensionalidade dos Dados (bandas)	99 amostras de treinamento	100 amostras de treinamento	101 amostras de treinamento
20	65,7	73	80
40	79,7	74,4	80,2
60	86,9	80,9	79,5
80	87,3	81,1	71,9
100	87,6	73,1	71,7
120	87,8	73,2	72,1
140	87,8	80,2	79,1
160	88,1	80,2	79,5
180	88,3	80,2	79,6

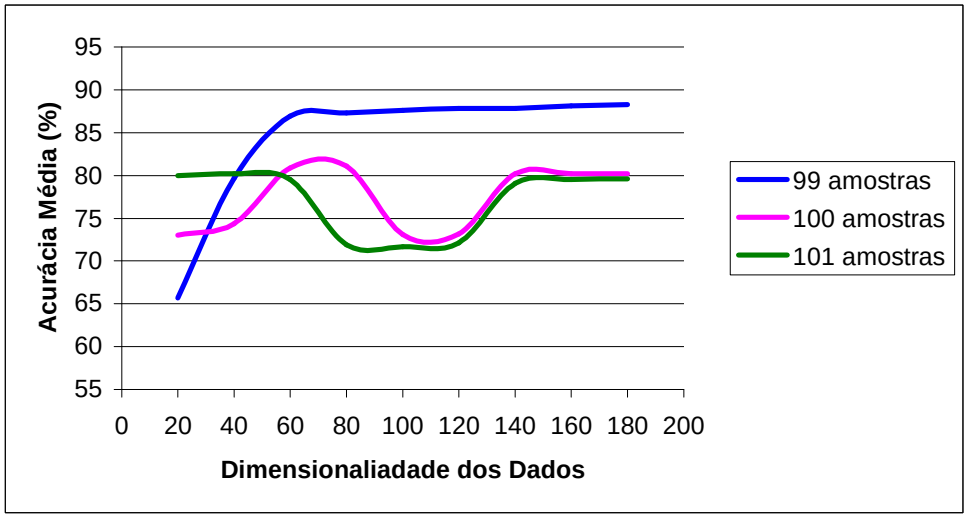


Figura 31: Acurácia média para CAB-SVM utilizando o kernel RBF γ 1, com o mesmo conjunto de amostras de teste

Nota-se que a diferença na acurácia média entre os três subconjuntos, em ambos os experimentos, não seria esperada dado que os conjuntos de amostras de treinamento são aproximadamente iguais. A partir destes experimentos pode-se observar que o classificador SVM (implementado em forma de árvore binária) é criticamente afetado pela presença de ruído, apresentando baixa acurácia média ou oscilações ao longo da dimensionalidade dos dados. De acordo com o padrão obtido nos experimentos anteriores, na ausência de amostras de treinamento ruidosas, o classificador SVM apresenta acurácia média praticamente constante a partir do momento que atinge determinado patamar. Tal padrão, no geral, foi encontrado para 99 amostras de treinamento, e apesar de não atingir a acurácia média esperada, substituiu o subconjunto de 100 amostras na realização dos experimentos da seção 4.3.1.

4.3.2.2 Resultados sem seleção de variáveis (sem SFS)

Com o intuito de verificar a eficácia do uso da seleção de variáveis via SFS, realizou-se os mesmos experimentos que os da seção 4.3.2, nas mesmas condições que estes foram realizados, bem como com os mesmos parâmetros para os kernels polinomial e RBF. Utilizou-se, para a realização destes experimentos, o mesmo subconjunto de bandas em todos os nós da árvore, selecionadas a intervalos regulares do espectro eletromagnético. Estes resultados, expressos em termos de acurácia média, estão ilustrados nas Figuras 32, 33, 34 e 35 e nas Tabelas 19, 20, 21 e 22 respectivamente para 50, 99, 200, e 300 amostras de treinamento.

Tabela 19: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 2 e RBF γ 1.5 para 50 amostras de treinamento

Dimensionalidade dos Dados (bandas)	MVG	Poly grau 2	RBF γ 1.5
20	82,2	84,2	82,3
40	73,9	73,5	84
60		76,4	85,2
80		85,2	85,4
100		82,5	86
120		85,6	85,8
140		85,6	85,2
160		85,8	84,8
180		86,5	84,7

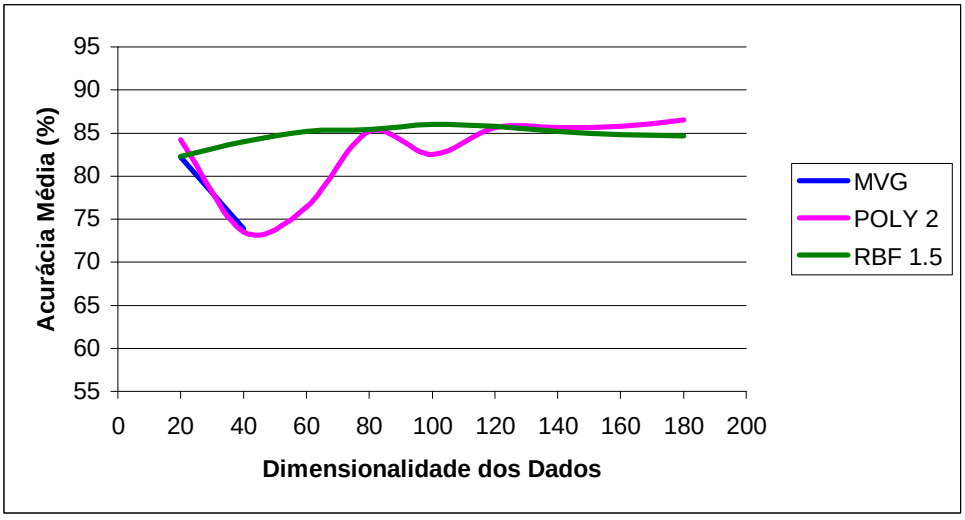


Figura 32: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 2 e RBF γ 1.5 para 50 amostras de treinamento

Tabela 20: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 3 e RBF γ 2 para 99 amostras de treinamento

Dimensionalidade dos Dados (bandas)	MVG	Poly grau 3	RBF γ 2
20	74,4	78,6	76,7
40	79,7	80,8	86,1
60	77,2	88,4	86,6
80	70,6	89,2	87,1
100		88,8	87,3
120		88,5	87,5
140		88,3	87,7
160		88,8	87,6
180		81,2	87,6

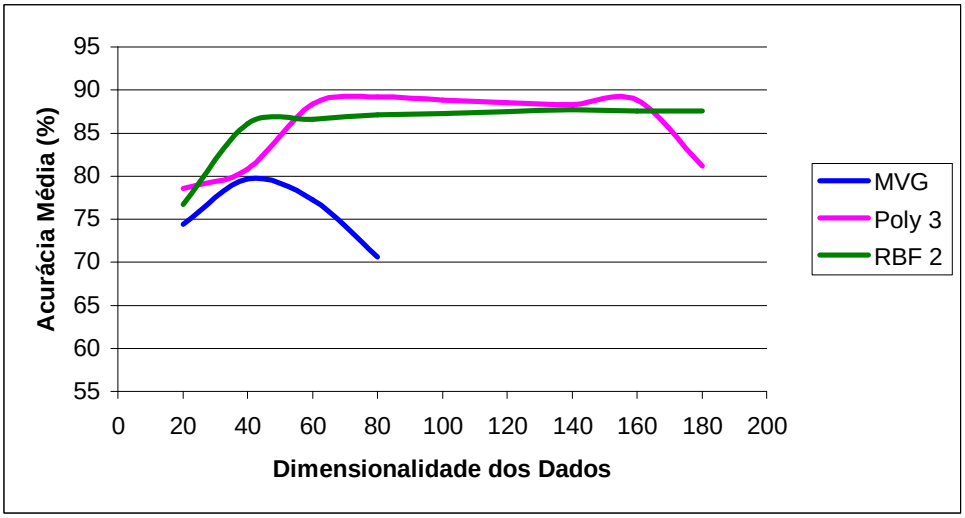


Figura 33: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 3 e RBF γ 2 para 99 amostras de treinamento

Tabela 21: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 3 e RBF γ 0.5 para 200 amostras de treinamento

Dimensionalidade dos Dados (bandas)	MVG	Poly grau 3	RBF γ 0.5
20	73,3	78,3	77,9
40	85,7	87,9	87,2
60	84,3	89,6	88,9
80	83,6	89,4	89,4
100	81,4	88,9	89,2
120	78,7	89,8	89,6
140	77,4	88,8	89,3
160	73,2	88,9	89,7
180	65,9	89,3	89,6

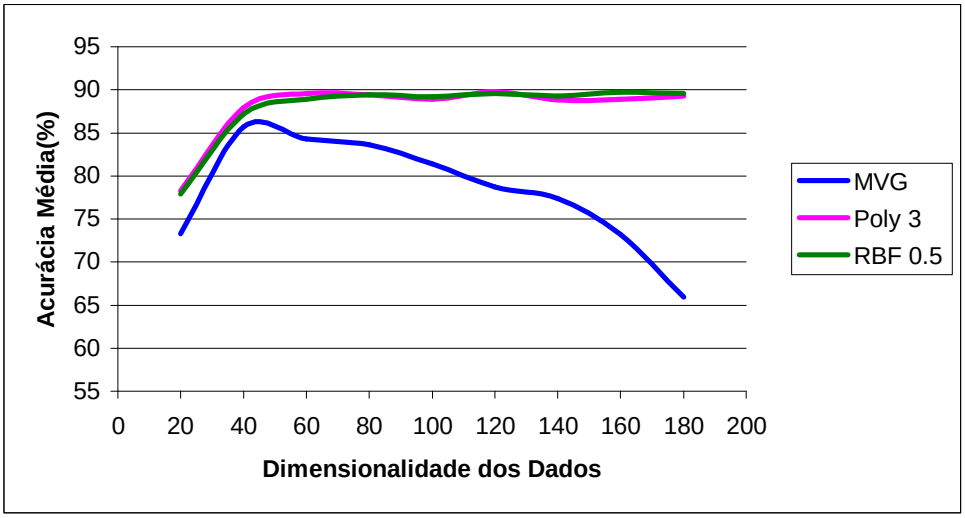


Figura 34: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 3 e RBF γ 0.5 para 99 amostras de treinamento

Tabela 22: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 3 e RBF γ 1.5 para 200 amostras de treinamento

Dimensionalidade dos Dados (bandas)	MVG	Poly grau 3	RBF γ 1.5
20	80,3	78,1	80,1
40	86,1	87,1	87,3
60	87,1	88,6	87,6
80	85,7	87,6	87,6
100	84,6	88,7	88,2
120	84,9	88,3	88,6
140	84,7	88,2	88,1
160	84,2	89,3	88,4
180	83,7	89	88,7

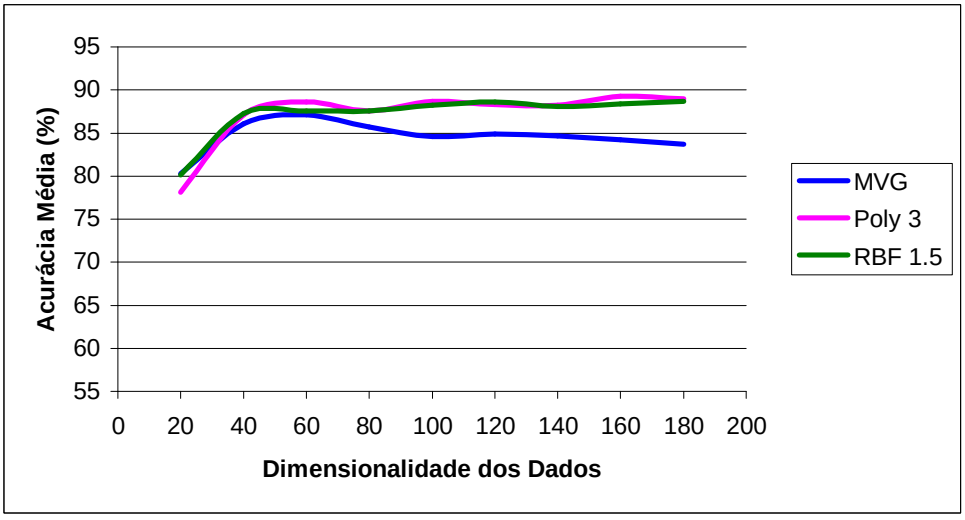


Figura 35: Acurácia Média para os classificadores Máxima Verossimilhança Gaussiana e SVM com kernel Polinomial grau 3 e RBF γ 1.5 para 99 amostras de treinamento

Comparando-se estes resultados com os obtidos na seção 4.3.2 pode-se perceber que o maior ganho quando se faz a seleção de variáveis via SFS é para o classificador MVG. O ganho em termos de acurácia média para este classificador com o uso do SFS é notável, com o pico passando de 79,7% para 89% com 99 amostras de treinamento e de 85,7% para 89,2% com 200 amostras de treinamento. O mesmo não acontece para o classificador SVM. Utilizando-se SFS com o critério Distância de Bhattacharyya para seleção de variáveis o ganho para o classificador SVM é mínimo e não significativo.

De acordo com os experimentos realizados nesta subseção e na subseção anterior (4.3.2.1) conclui-se que o problema de oscilação na acurácia média encontrado nos experimentos realizados com a ferramenta CAB-SVM deve-se à seleção de amostras de treinamento ruidosas e, principalmente, à seleção de bandas. Quando a seleção de variáveis é feita através do algoritmo SFS, este seleciona as

bandas mais separáveis, mas provavelmente também as que possuem o maior número de *outliers*. Para o classificador paramétrico esta situação não é tão problemática pois são tomadas suas médias e matrizes de covariâncias, não influenciando diretamente na classificação final. Como o classificador SVM é geométrico, o mapeamento destes *outliers* afetam diretamente na definição da fronteira de decisão influenciando, deste modo, nos resultados obtidos em termos de acurácia média.

Percebe-se, assim, que o uso do SFS com distâncias estatísticas como critério não é apropriado para seleção de variáveis em classificadores não-paramétricos, como SVM.

4.3.3 Desempenho Computacional

Na Figura 36 é apresentado o tempos de processamento para treinamento utilizando as ferramentas CAB-SVM (kernel polinomial grau 3 e RBF γ 0.5) e CAB-MVG para 200 amostras de treinamento, empregando-se um Core 2 Duo, com 1GB memória.

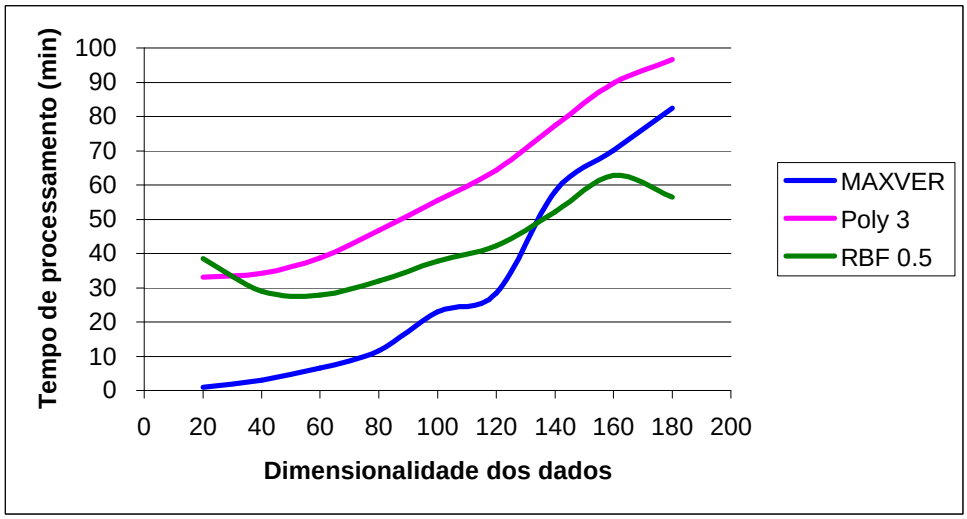


Figura 36: Tempos de processamento para treinamento dos aplicativos CAB-MVG e CAB-SVM (com os kernels polinomial grau 3 e RBF γ 0.5).

Em termos computacionais, a ferramenta implementada a partir do classificador MVG (CAB-MVG) é mais rápido que o implementado a partir do classificador SVM (CAB-SVM). No caso do aplicativo CAB-MVG, o único gargalo é a função SFS para seleção de bandas espectrais. Deve-se notar que, na Figura 36, o tempo de processamento é dado pela soma dos tempos tomados em cada nó da árvore e, deste modo, sua estrutura influencia no tempo final obtido. Esperava-se que o tempo de

processamento para treinamento do aplicativo CAB-MVG seguisse menor ou igual aos tempos encontrados para o CAB-SVM, entretanto, a partir de 140 bandas, a estrutura da árvore desenhada pelo CAB-MVG conta com 63 nós entre terminais e não-terminais, enquanto que CAB-SVM, com o kernel RBF a árvore é constituída de 51 nós entre terminais e não-terminais e em 180 bandas conta com 45 nós entre terminais e não-terminais. A diferença no número de nós não-terminais é que determina o tempo maior para treinamento o aplicativo CAB-MVG a partir de 140 bandas.

Já para o CAB-SVM, duas funções são responsáveis pelo longo tempo tomado pelo treinamento do classificador: SFS (seleção de bandas espectrais) e quadprog (estimação dos multiplicadores de Lagrange. Na Figura 37, apresenta-se o percentual do tempo utilizado pelas funções quadprog e SFS no treinamento do classificador com o kernel polinomial grau 3 e na Figura 38, o mesmo é feito para o kernel RBF γ 0.5.

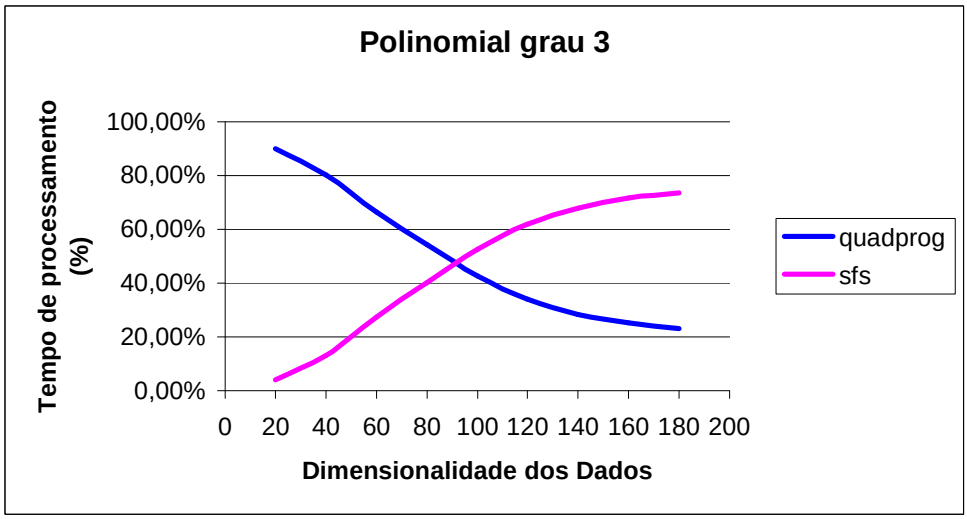


Figura 37: tempo de processamento das funções quadprog e SFS em porcentagem do tempo total despendido para treinamento do classificador utilizando o kernel polinomial grau 3.

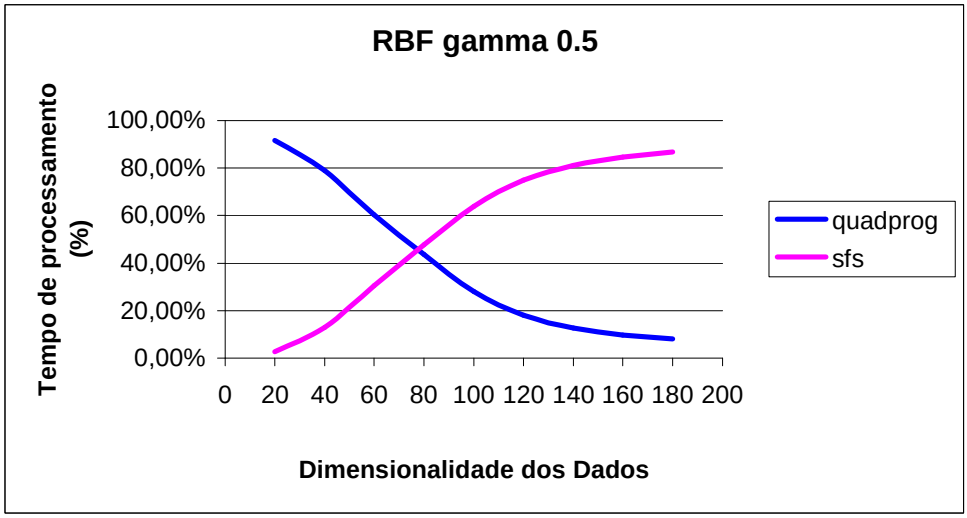


Figura 38: tempo de processamento das funções quadprog e SFS em porcentagem do tempo total despendido para treinamento do classificador utilizando o kernel RBF γ 0.5.

Observe que, com o aumento da dimensionalidade dos dados, o tempo requerido para a função quadprog diminui, enquanto que para a função SFS, aumenta. As duas funções juntas tomam de 90% à quase 97% do tempo total em todas as dimensionalidades.

CAPÍTULO V

CONCLUSÕES E SUGESTÕES

5.1 CONCLUSÕES

Nesta dissertação investiga-se a utilização do classificador não-paramétrico SVM na classificação de imagens em alta dimensionalidade como as imagens hiperespectrais que vem sendo mais recentemente utilizadas em sensoriamento remoto. SVM é um classificador binário, isto é, trata da separação de duas classes a cada vez. Como em imagens de cenas naturais um numero maior de classes está presente, propõe-se nesta dissertação inserir o classificador SVM em um outro classificador, este em estágio múltiplo e estruturado em forma de uma árvore binária. Nesta estrutura, somente duas classes são tratadas em cada nó possibilitando, portanto, o emprego do classificador SVM. Na abordagem proposta nesta dissertação, a estrutura da árvore binária, isto é, a caracterização dos dois nós descendentes é definida a cada estágio com o auxilio da distancia de Bhattacharyya. Esta distância estatística serve para estimar, em cada nó, o par de classes que apresenta a maior separabilidade e que darão origem aos dois nós descendentes no nível seguinte da arvore binária. As variáveis empregadas em cada estágio da árvore binária para fins de alocação das amostras das demais classes nos nós descendentes são selecionadas pelo algoritmo SFS.

Como já foi dito, a metodologia proposta é dirigida ao processo de classificação de dados em alta dimensionalidade (imagens hiperespectrais). Como é bem conhecido, o atrativo da utilização destes dados reside em possibilitar a separação de classes com características espectrais muito semelhantes (FUKUNAGA, 1990), não separáveis quando se emprega dados convencionais em baixa ou média dimensionalidade. Esta vantagem fica, entretanto, prejudicada em classificadores paramétricos (caso do classificador MVG) sempre que o número de amostras de treinamento disponíveis não é limitado, fato este que ocorre com freqüência em situações reais, resultando no conhecido problema conhecido por fenômeno de Hughes. É no fato de que tal fenômeno não atinge classificadores não-paramétricos que reside a vantagem do SVM.

Com o intuito de comparar os resultados, o processo de classificação empregando a estrutura em árvore binária foi repetido, empregando-se também o classificador tradicional MVG, implementado no aplicativo CAB-MVG.

Para fins de teste da metodologia proposta, foi empregada uma imagem coletada pelo sistema sensor AVIRIS, o qual disponibiliza 190 bandas espectrais cobrindo as porções do visível, infra-vermelho próximo e infra-vermelho médio (0.4μm – 2.4 μm) no espectro eletromagnético. A cena utilizada consiste de uma área teste coberta por classes espectralmente muito semelhantes, constituindo-se assim em um desafio ao classificador. Com o intuito de investigar o comportamento da metodologia proposta, os experimentos foram realizados utilizando quatro conjuntos de dados, empregando respectivamente 50, 99, 200 e 300 amostras para treinamento. Para estimar a acurácia dos resultados produzidos pela metodologia proposta, foram utilizados 300 amostras para teste por classe em cada caso.

Os experimentos desenvolvidos tendem a confirmar a eficácia da metodologia proposta. Os experimentos mostraram que a metodologia de classificação aqui proposta, empregando o classificador SVM em uma árvore binária apresentou resultados superiores em termos de acurácia quando comparado com um classificador paramétrico (MVG). A maior diferença de acurácia média obtida pelos dois métodos de classificação ocorre para o menor conjunto de amostras de treinamento testado, 50 amostras, com o kernel polinomial grau 2, com um ganho relativo de 2%. O melhor resultado absoluto obtido pela metodologia proposta, em termos de acurácia média, acontece com 300 amostras de treinamento, utilizando-se o kernel RBF γ 1.5, com 90,7%. Além de proporcionar resultados melhores que os obtidos empregando um classificador paramétrico (MVG), a metodologia empregando SVM apresenta ainda a vantagem de não sofrer os efeitos dos fenômeno de Hughes: a partir do momento que atinge determinado patamar, a acurácia média se mantém relativamente constante, quando na ausência de amostras ruidosas em seus conjuntos de treinamento.

Apesar dos resultados promissores o classificador proposto tem limitações, principalmente no que diz respeito ao tempo de processamento e seleção de variáveis via SFS. A função “quadprog.m” (disponível no toolbox de Otimização em MATLAB e aqui utilizada para fins de estimação dos multiplicadores de Lagrange) e também a função SFS (responsável pela seleção das bandas mais separáveis) demandam um longo tempo de processamento. Além de tomar um longo tempo de processamento, a função SFS (com o critério de separabilidade Distância de Bhattacharyya) não é apropriada para seleção de variáveis em classificadores não-paramétricos. Os resultados obtidos com a ferramenta CAB-SVM (feita a seleção de variáveis via SFS em cada nó da árvore) são praticamente iguais aos resultados obtidos sem o uso de

SFS. Tal limitação do classificador proposto ficou bem demonstrado nos experimentos, e acontece devido à sua alta sensibilidade à ruídos ou desvios maiores presentes em amostras de treinamento, causando degradação de sua performance em termos de acurácia média.

5.2 SUGESTÕES

Como sugestão para futuros desenvolvimentos neste tópico, sugere-se que sejam investigadas outras abordagens no que se refere à estratégias multi-classe envolvendo SVM, bem como que sejam realizadas pesquisas mais criteriosas para detectar os efeitos que amostras de treinamento ruidosas têm sobre este classificador, investigando abordagens alternativas para seleção de variáveis em cada nó da árvore binária, visando maximizar os resultados em termos de acurácia. Outra área significativa a ser estudada são métodos mais eficientes para a estimação dos multiplicadores de Lagrange e o desenvolvimento de ferramentas computacionais mais eficientes capazes de processar o grande volume de dados que caracterizam imagens em alta dimensionalidade, possibilitando desta forma seu emprego em mais larga escala.

REFERÊNCIAS BIBLIOGRÁFICAS

ABE, S.; **Support Vector Machines for Pattern Classifications.** Kobe, Japão: Ed. Springer, 2005.

AEBERHARD, S.; COOMANS, D.; DE VEL, O.; Comparative Analysis of Statistical Pattern Recongnition Methods in High Dimensional Settings. **Pattern Recognition**, vol.27, no 8, pp 1065-1077, 1994.

AVIRIS. **Aviris Concept.** Disponível em: <http://aviris.jpl.nasa.gov>. Acesso em 3 de dezembro de 2008.

BAZI, Y.; MELGANI, F.; Towardan Optimal SVM Classification System for Hyperspectral Remote Sensing Images. **IEEE Transactions on Geoscience and Remote Sensing,** vol.44, no11, pp 3374-3385, novembro de 2006.

BERGE, A.; JENSEN, A.C.; SOLBERG, A.S.; Sparse inverse covariance estimates for hyperspectral image classification. **IEEE Transactions on Geoscience and Remote Sensing**, vol.45, no5, pp 1399–1407, maio de 2007.

CONGALTON, R.; A review of assessing the accuracy of classifications of remotely sensed data. **Remote Sensing of Environment**, vol.37, no1, pp 35-46, 1991.

CORTIJO F.J.; DE LA BLANCA N.P.; The Performance of Regularized Discriminant Analysis Versus Non-Parametric Classifiers Applied to High-Dimensional Image Classification. **International Journal of Remote Sensing**, vol.20, no17, 1999.

DUDA, O. R.; HART, P. E.; STORK, D. G.; **Pattern Classification.** Second Edition. A Wiley-Interscience Publication: 2000.

EMBRAPA. **Manejo do Solo**. Disponível em: <http://sistemasdeproducao.cnptia.embrapa.br>. Acesso em 3 de dezembro de 2008.

EMBRAPA MONITORAMENTO POR SATÉLITE. **Sistemas Orbitais de Monitoramento e Gestão Territorial.** Campinas: Embrapa Monitoramento por Satélite, 2009. Disponível em: <http://www.sat.cnpm.embrapa.br>. Acesso em 2 de março de 2009.

FOODY, G.M.; MATHUR, A.; Multiclass and Binary SVM Classification: Implications for Training and Classification Users. **IEEE Geoscience and Remote Sensing Letters,** vol.5, no2, pp 241–245, Abril de 2008.

FUKUNAGA, K.; **Introduction to Statistical Pattern Recognition.** Second Edition. Academic Press: 1990.

GUO, B.; GUNN, S.R.; DAMPER, R.I.; NELSON, J.D.B.; Customizing Kernel Functions for SVM-Based Hyperspectral Image Classification. **IEEE Transactions on Image Processing**, vol.17, no 4 , pp 622-629, abril de 2008.

HASTIE, T.; BUJA, A.; TIBSHIRANI, R.; Penalized discriminant analysis. **Annals of Statistics**, vol.23, pp 73–102, 1995.

HERBRICH, R.; **Learning Kernel Classifiers:** *Theory and Algorithms*. The MIT Press, 2002.

HOFFBECK J.P.; LANDGREBE D.A.; Covariance Matrix Estimation and Classification with Limited Training Data. **IEEE Transactions on Pattern Analysis and Machine Intelligence** vol.18, nº7, julio 1996.

HUANG, C.; DAVIS, L.S.; TOWNSHEND, J.R.G.; An Assessment of Support Vector Machines for Land Cover Classification. **International Journal of Remote Sensing,** vol.23, no4, pp 725–749, 2002.

JACKSON, Q.; LANDGREBE, D.A.; An adaptive classifier design for high-dimensional data analysis with a limited training data set. **IEEE Transactions on Geoscience and Remote Sensing**, vol.39, no12, pp. 2664 – 2679, dezembro de 2001.

JIA, X.; RICHARDS, J.; Segmented Principal Components Transformation for Efficient Hyperspectral Remote-Sensing Image Display and Classification. **IEEE Transactions on Geoscience and Remote Sensing,** vol.37, no1, pp 538-542 janeiro de 1999.

JIMENEZ, L.O.; LANDGREBE, D.A.; Hyperspectral Data Analysis and Supervised Feature Reduction Via Projection Pursuit. **IEEE Transactions on Geoscience and Remote Sensing**, vol.37, no6, pp 2653 – 2667, novembro 1999.

JOHNSON, R. A.; WICHERN, D. W. **Applied Multivariate Statistical Analysis**. New Jersey, USA: Prentice-Hall,1982.

KUO, B.C.; CHANG, K.Y.; Feature extractions for small sample size classification problem; **IEEE Trans. Geosci. Remote Sensing**, vol.45, no3, pp 756–764, março de 2007;

LANDGREBE, D. A.; **Signal Theory Methods In Multispectral Remote Sensing**. Wiley Interscience, 2003.

LEWOTSKY, K.; Hyperspectral Imaging: Evolution of Imaging Spectrometry. **OE Reports**. november 1994 issue, Disponível em <http://www.spie.org/web/oer/november/image_spectro. html>.

LICZBINSKI, C.; HAERTEL, V.; A new Approach to Estimate a Priori Probabilities in Remote Sensing Digital Image Classification. **Canadian Journal of Remote Sensing Journal Canadien De Télédétection,** vol.34, no2, pp 135-142, abril de 2008.

LORENA, A. C.; CARVALHO, A. C. P. L. F.; Uma Introdução às Support Vector Machines. Revista de Informática Teórica e Aplicada. **Revista de Informática Teórica e Aplicada**, vol.14, no2, p 43-67, 2007.

MAPSOFWORLD. **Location Map of Indiana**. Disponível em: <www.mapsofworld.com/usa/states/indiana/indiana-location-map.html>. Acesso em 2 de dezembro de 2008.

MELGANI, F.; BRUZZONE, L.; Classification of Hyperspectral Remote Sensing Images with Support Vector Machines. **IEEE Transactions on Geoscience and Remote Sensing,** vol.42, no8, pp 1778- 1790, agosto de 2004.

MINGMIN CHI; BRUZZONE, L.; A semilabeled-sample-driven bagging technique for ill-posed classification problems. **IEEE Geoscience and Remote Sensing Letters,** vol.2, no1, pp. 69-73, janeiro de 2005.

MORAES, D. A. O.; **Extração de Feições em Dados Imagem com Alta Dimensão por Otimização da Distância de Bhattacharyya em um Classificador de Decisão em Árvore.** Porto Alegre, RS. 2005. Dissertação de Mestrado – Departamento de Pós-Graduação em Sensoriamento Remoto – UFRGS.

PONTIL, M.; VERRI, A.; Support vector machines for 3-D object recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence,** vol.20, no6, p 637–646, 1998.

RICHARDS, J. A.; JIA, X.; **Remote Sensing Digital Image Analysis**. New York: Springer, 1999. Third Edition.

SAVAVIAN, S. R.; LANDGREBE, D.; A Survey of Decision Trees Classifier Methodology. **IEEE Trans Systems, Man and Cybernetics**, vol 21, no3, pp 660-674,1991.

SERPICO, S. B.; BRUZZONE, L.; A New Search Algoritthm for Feature Selection in Hiperespectral Remote Sensing Images". **IEEE Transaction on Geoscience and Remote Sensing;** Especial Issue on Analysis of Hiperspectral Image Data, vol.39, no7, pp 1360-1367, julho de 2001.

SERPICO, S. B.; D'INCA, M.; MELGANI, F.; MOSER, G.; A Comparison of Feature Reduction Techniques for Classification of Hyperspectral Remote-Sensing Data. **Procedings of Spie, Image and Signal Processing of Remote Sensing VIII**, vol.4885, pp 347-358, 2003.

SMOLA, A.J.; BARTLETT P.L.; SCHOLKOPF B.; SCHUURMANS D.; **Advances in large margin classifiers**. Massachusetts Institute of Technology. London, England: Ed. MIT Press, 2000.

THERRIEN, C.W.; **Decision Estimation and Classification**: *an Introduction to Pattern Recognition and Related Topics*.; USA: John Wiley & Sons, 1989.

VAPNIK, V.N.; **The Nature of Statistical Learning Theory**. USA: Springer, 2nd ed., 1999.

WANG, L; Feature Selection with Kernel Class Separability. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol.30, no9, pp1534-1546, Setembro de 2008.

ZHAN, Y.; SHEN, D.; An adaptive error penalization method for training an efficient and generalized SVM. **Pattern Recognition,** vol.39, no3, pp 342-350, março de 2006.

ZHONG, P.; WANG, R.; Learning Sparse CRFs for Feature Selection and Classification of Hyperspectral Imagery. **IEEE Transactions on Geoscience and Remote Sensing**, vol.46, no12, pp 4186–4197, dezembro de 2008.

ZORTEA, M. HAERTEL, V. CLARKE, R. Feature Extraction in Remote Sensing High-Dimensional Image Data. **IEEE Geoscience and Remote Sensing Letters**, vol.4, no1, pp 107-111, janeiro de 2007.

APÊNDICE A

CÓDIGO FONTE

1 FUNÇÃO PARA PADRONIZAÇÃO DAS AMOSTRAS AVIRIS

```
function x = standardize_data(samples)

%-------------------------------------------------------------------
%
% padronizacao das amostras de treinamento de acordo com as Equacoes
% (46) e (47), dadas por (JOHNSON E WICHERN ,1982). As amostras devem
% estar no formato ERDAS 7.4
%
%-------------------------------------------------------------------

M = mean(samples);
S = sqrt(var(samples));
V = inv(diag(S));

for pixel=1:size(samples,1)
   std_samples(pixel,:) = (V * (samples(pixel,:) - M)')';
end

save std_samples std_samples
```

2 FUNÇÃO PARA SELEÇÃO DE AMOSTRAS DE TREINAMENTO E TESTE

```
function a=aviris()

%-------------------------------------------------------------------
%
% selecao de amostras de treinamento e teste
%
%-------------------------------------------------------------------

%amostras disponiveis
load corn_min_std;
load corn_notill_std;
load grass_trees_std;
load soy_clean_std;
load soy_min_std;
load soy_notill_std;

%amostras na variavel s
s(1).amostra= corn_min_std;
s(2).amostra= corn_notill_std;
s(3).amostra= grass_trees_std;
s(4).amostra= soy_clean_std;
s(5).amostra= soy_min_std;
```

```
s(6).amostra= soy_notill_std;

%divisao das amostras em amostras de teste e treinamento
for d=1:size(s,2)
   train=[];
   test=[];
   for pixel=1:2:size(s(d).amostra,1)-1
      train=[train;s(d).amostra(pixel,:)];
      test =[test; s(d).amostra(pixel+1,:)];
   end
   x(d).amostra=train;
   teste(d).amostra=test;
   tamanho(d,1)=size(x(d).amostra, 1);
end
w=min(tamanho);
disp(['Numero maximo de amostras para treinamento: ' num2str(w)])
n_train=input('Numero de amostras de treinamento: ')

%selecao das amostras de treinamento e atribuicao das demais amostras
%para teste
for j=1:size(x, 2)
   aux=[];
   fl=[];
   step=(size(x(j).amostra, 1)/n_train);
   for k=1:step:size(x(j).amostra, 1)
      fl=[fl; round(k)]; %amostras para treino
      aux=[aux; x(j).amostra(k,:)];
   end
   outras=[];
% atribuicao das demais amostras para teste
   for d=1:size(x(j).amostra, 1)
      f2=any(d==fl);
      if (f2==0)
         outras=[outras; x(j).amostra(k, :)];
      end
   end
   x(j).amostra=aux;
   teste(j).amostra=[teste(j).amostra; outras];
   tamanho(j,1)=size(teste(j).amostra, 1);
end
save x x;

% selecao das amostras de teste
w=min(tamanho);
disp(['Numero maximo de amostras disponiveis para teste: 'num2str(w)])
n_teste=input('Numero de amostras de teste: ')
for j=1:size(teste, 2)
   step=(size(teste(j).amostra, 1)/n_teste);
   aux=teste(j).amostra(1:step:size(teste(j).amostra, 1),:);
   teste(j).amostra=aux;
end
save teste teste;
```

3 FUNÇÕES PARA TREINAMENTO DA ÁRVORE (CAB-SVM)

```
function c=cab_svm(kernel, param, slack, dim, n_bandas, limiar)

%-------------------------------------------------------------------
%
%   Funcao para treinamento para o Classificador em arvore binária
% utilizando a função de decisão SVM
%
%   kernel: pode ser RBF ou Poly
%   param: parametro do kernel: d ou gamma
%   slack: valor do parametro C
%   dim: dimensao de bandas selecionadas para determinar o plano de
% separaçao entre as classes usado em sfs
%   n_bandas: numero de bandas utilizadas para calculo da distancia de
% Bhattacharyya para decidir as classes que darão origem aos nos
% descendentes
%   limiar: Limiar em escala percentual (0 a 100) para atribuicao de
% classes nos nos da arvore no processo de classificacao
%
%   Exemplo de sintaxe: cab_svm('RBF',0.5,10,6,100,99);
%
%-------------------------------------------------------------------

inicio=clock; %inicio
load x; %amostras de treinamento

step=round(size(x(1).amostra, 2)/n_bandas);
auxil=0; %variavel auxiliar para contagem de nos terminais
save auxil auxil;
nos=1;
temp=1;
nnn=[1]; %nos nao nulos
terminal=[]; %nos terminais
nos_svm=[]; %nos que contem dados para aplicaçao da funcao SVM

t_sfs=0; %tempo total da funcao sfs em todos os nos
t_quadprog=0; %tempo da funcao quadprog
save t_quadprog t_quadprog;


if (limiar<0) | (limiar>100)
   error('Valor de limiar deve ficar entre 0 e 100')
end


if (dim>size(x(1).amostra, 2))
    disp(['Numero maximo de bandas disponiveis:' ...
         ... num2str(size(x(1).amostra, 2))])
    error('dimensao deve ser menor que numero total de bandas')
end


if (n_bandas>size(x(1).amostra, 1))
    disp(['Numero de amostras de treinamento:' ...
         ... num2str(size(x(1).amostra, 1))])
    warning('n_bandas deveria ser menor que numero de amostras
         ... disponiveis para treinamento')
end
```

```
while nos~=1000
   var_nos(nos).kernel=kernel;
   var_nos(nos).param=param;

   if nos==1 %atribuicao de todas as classes no no um
      arvore(1).no=[];
      for i=1:size(x,2)
         arvore(1).no=[arvore(1).no,i];
      end
   end

   if size(arvore(nos).no,2)>=3  %classes no no >=3
      disp(['calculo do no: ' num2str(nos)])
      nos_svm=[nos_svm; nos];
      in_distancia=clock;
% calcula da distância de Bhattacharyya
      [bhatt]=distance(arvore(nos).no, step, nos);
      var_nos(nos).bhatt=bhatt(nos);
      var_nos(nos).classe=arvore(nos).no;
      fim_distancia=clock;
      tempo_distancia=etime(fim_distancia, in_distancia)
      in_sfs=clock;
      [feat]= sfs(nos, bhatt, dim);
      var_nos(nos).feat=feat;
      fim_sfs=clock;
      tempo_sfs=etime(fim_sfs, in_sfs)
      t_sfs=t_sfs+tempo_sfs;
      in_svm=clock;
% padroniza os parametros da funcao SVM_train
      [train, target]=stand_svm(nos, feat, bhatt);
      var_nos(nos).train=train;
      var_nos(nos).target=target;
% chamada da funcao que calcula os coeficientes da funcao SVM_train
      [svm_par]=svm_train(train, target, kernel, param, slack, nos);
      var_nos(nos).svm_par_train=svm_par(nos);
      save var_nos var_nos;
      fim_svm=clock;
      tempo_svm=etime(fim_svm, in_svm)
      in_class=clock;
% chamada da funcao que determina se as demais classes vao para no
% direito ou esquerdo
      [no]=buildCAB(kernel, param, limiar, auxil, nos);
      var_nos(nos).no=no;
      fim_class=clock;
      tempo_build=etime(fim_class, in_class)
% atualiza valor das classes nos nos filhos (no direito e esquerdo)
      arvore(nos*2).no=no(1).esq;
      nnn=[nnn, nos*2];
      arvore(nos*2+1).no=no(2).dir;
      nnn=[nnn, nos*2+1];
      temp=nos*2+1; %valor temporario de noh jah acessado

   elseif size(arvore(nos).no,2)==2  %classes no no ==2
      disp(['calculo do no: ' num2str(nos)])
      nos_svm=[nos_svm; nos];
      in_distancia=clock;
% calcula da distancia de Bhattacharyya
      [bhatt]=distance(arvore(nos).no, step, nos);
      var_nos(nos).bhatt=bhatt(nos);
      var_nos(nos).classe=arvore(nos).no;
      fim_distancia=clock;
```

```
      tempo_distancia=etime(fim_distancia, in_distancia)
      in_sfs=clock;
      [feat]= sfs(nos, bhatt, dim);
      var_nos(nos).feat=feat;
      fim_sfs=clock;
      tempo_sfs=etime(fim_sfs, in_sfs)
      t_sfs=t_sfs+tempo_sfs;
      in_svm=clock;
% padroniza os parametros da funcao SVM_train
      [train, target]=stand_svm(nos, feat, bhatt);
      var_nos(nos).train=train;
      var_nos(nos).target=target;
% chamada da funcao que calcula os coeficientes da funcao SVM_train
      [svm_par]=svm_train(train, target, kernel, param, slack, nos);
      var_nos(nos).svm_par_train=svm_par(nos);
      save var_nos var_nos;
      fim_svm=clock;
      tempo_svm=etime(fim_svm, in_svm)
      in_class=clock;
% chamada da função que determina se as demais classes vao para no
% direito ou esquerdo
      [no]=buildCAB(kernel, param, limiar, auxil, nos);
      var_nos(nos).no=no;
      fim_class=clock;
      tempo_build=etime(fim_class, in_class)
% atualiza valor das classes nos nos filhos (no direito e esquerdo)
      arvore(nos*2).no=no(1).esq;
      nnn=[nnn, nos*2];
      arvore(nos*2+1).no=no(2).dir;
      nnn=[nnn, nos*2+1];
      temp=nos*2+1;

   elseif size(arvore(nos).no,2)==1  %classes no no terminal
      disp(['no terminal: ' num2str(nos)])
      terminal=[terminal; nos];
      var_nos(nos).classe=arvore(nos).no;
      cont=0;
      for aux2=1:temp
         if size(arvore(aux2).no,2)==1
            cont=cont+1;   %soma nos que contem apenas uma classe
         end
      end

% se nº de nos que contem apenas uma classe sao iguais ao nº de
% classes total + nº de vezes q as classes q foram para ambos os nos
      load auxil;
      if (cont==(size(arvore(1).no,2)+auxil))
         for i=(nos+1):temp
            if size(arvore(i).no,2)==1
               disp(['no terminal: ' num2str(i)])
               terminal=[terminal; i];
            end
% atualiza valor do no temp->nos*2+1
            var_nos(i).classe=arvore(i).no;
         end
         nos=999;
      end
   end
   nos=nos+1;
end
```

```
save nos_svm nos_svm; %nos com fronteiras de decisao svm
save terminal terminal; % nos terminais
save nnn nnn; %nos nao nulos -> nos_svm+terminal;
save var_nos var_nos;
save arvore arvore;

fim=clock;
tempo_de_processamento = etime(fim, inicio)
save t_sfs t_sfs;
save tempo_de_processamento tempo_de_processamento;
```

```
function [bhatt]=distance(classes, step, nos)

%-----------------------------------------------------------------------
%
% retorna as classes i e j mais separaveis entre as classes dada por
% 'classes', no noh 'nos'.
% a variavel classe retorna com valor diferente de '0' para as classes
% restantes
%
%-----------------------------------------------------------------------

load x;
distancia=[];

for i=1:(size(classes,2))
   for j=1:(size(classes, 2))
      if (i ~= j)
         amostra1=[];
         amostra2=[];
         for k=1:step:size(x(classes(i)).amostra, 2)
            amostra1=[amostra1 x(classes(i)).amostra(:,k)];
            amostra2=[amostra2 x(classes(j)).amostra(:,k)] ;
         end
         valor= bhattacharyya(amostra1, amostra2);
         distancia= [distancia; [classes(i) classes(j) valor]];
      end
   end
end

[Y, I]=max(distancia);
bhatt(nos).valor=Y(3); %recupera a maior distancia entre as classes
linha=I(3); %recupera a linha q contem a maior distancia
%i e j sao as classes mais separaveis entre si.
bhatt(nos).i=distancia(linha, 1);
bhatt(nos).j=distancia(linha, 2);

for k= 1:size(classes, 2)
% '0' para classes i e j, as demais classes ficam com seus valores
% originais
   if ((classes(k)== (bhatt(nos).i)) | (classes(k)== (bhatt(nos).j)))
      classes(k)=0;
   end
end

save classes classes;
```

```
function B=bhattacharyya(i, j)

%-------------------------------------------------------------------
%
% calcula distancia entre duas amostras por Bhattacharyya (Equação 4)
%
%-------------------------------------------------------------------

i_media=mean(i);
i_covar=cov(i);
j_media=mean(j);
j_covar=cov(j);

% distancia de Bhattacharyya
B=(1/8)*(i_media-j_media)*...
   (inv((i_covar+j_covar)/2))*...
   (i_media-j_media)'+...
   (0.5)*log((det((i_covar+j_covar)/2))/...
   (det(i_covar)^(1/2)*(det(j_covar)^(1/2))));
```

```
function [feat]=sfs(nos, bhatt, dim)

%-------------------------------------------------------------------
%
% funcao que seleciona as n feicoes mais significativas para
% determinar a separabilidade por bhattacharyya entre duas classes
%
%-------------------------------------------------------------------

load x;
i=bhatt(nos).i;
j=bhatt(nos).j;
feat=0;
cont=0;
coluna=[]; % colunas (bandas) selecionadas por Bhattacharyya
bandas=[]; % conjunto das bandas disponiveis a serem selecionadas
sel_am1=[]; %banda selecionada classe i
sel_am2=[]; %banda selecionada classe j

%atribuiçao da dimensionalidade total da imagem a variavel bandas
for k=1:size(x(i).amostra,2)
   bandas=[bandas; k];
end

%ate que o numero de bandas seja igual a dimensionalidade desejada
while (cont ~= dim)
   cont=cont+1;
   distancia=[];
% de 1 ateh numero de bandas ainda disponiveis para seleçao
   for m=1:size(bandas,1)
      sample1=[];
      sample2=[];
%banda m da amostra da classe i
      sample1=[sel_am1 x(i).amostra(:,bandas(m))];
%banda m da amostra da classe j
      sample2=[sel_am2 x(j).amostra(:,bandas(m))];
      valor=bhattacharyya(sample1, sample2);
      distancia= [distancia; [bandas(m) valor]];
   end
   [Y, I]=max(distancia);
%linha da lista que contem a coluna com maior distancia
```

```
   coluna=[coluna; distancia(I(2))];
   aux=[];
   for b=1:size(bandas, 1)
      if (bandas(b)~= coluna(cont))
         aux=[aux; bandas(b)];
      end
   end
   bandas=aux;
   sel_am1=[sel_am1 x(i).amostra(:,coluna(cont))];
   sel_am2=[sel_am2 x(j).amostra(:,coluna(cont))];
%numero de bandas selecionadas igual ao numero de bandas disponiveis
   if ((size(x(i).amostra,2))==dim) & (size(bandas,1)==1)
      coluna=[coluna; bandas(1)];
      cont=cont+1;
   end
end
feat=coluna;
```

```
function [train, target]=stand_svm(nos, feat, bhatt)

%-------------------------------------------------------------------
%
% padronizacao dos parametros train e target para a chamada da
% funcao SVM_train
%
%-------------------------------------------------------------------

load x; % amostras de treinamento
train=[];
target=[];
feat=sort(feat); %ordena as bandas selecionadas

for g=1:(size(x(bhatt(nos).i).amostra,1))
   y=[];
   for t=1:(size(feat,1))
      y=[y; x(bhatt(nos).i).amostra(g,feat(t))];
   end
   train=[train; y'];
   target=[target; -1];
end
for k=1:(size(x(bhatt(nos).j).amostra,1))
   y=[];
   for t=1:(size(feat,1))
      y=[y; x(bhatt(nos).j).amostra(k,feat(t))];
   end
   train=[train; y'];
   target=[target; 1];
end
```

```
function [svm_par]=svm_train(train, target, kernel, param, slack, nos)

%-------------------------------------------------------------------
%
% funcao que implementa o classificador SVM como descrito em (ABE,
% 2005). Para o problema da estimacao dos alphas nas equacoes (40) e
%(41) usa-se a funcao 'quadprog.m'.
%
%-------------------------------------------------------------------
N1= size(train,1);
kernel_matrix=[];
```

```
switch kernel
   case 'RBF' % Equação (44)
      for row=1:N1
         for col=1:N1
            dif=train(row,:)-train(col,:);
            kernel_matrix(row,col)=double(exp(-param*(dif*dif')));
         end
      end
   case 'Poly' % Equação (45)
      for i=1:N1
         for j=1:N1
            kernel_matrix(i,j)=(train(i,:)*train(j,:)'+1)^param;
         end
      end
   otherwise
      error('kernel deve ser RBF ou Poly')
end

% calculo dos multiplicadores de Lagrange
   load t_quadprog;
   in=clock;
   alpha=quadprog(diag(target)*kernel_matrix*diag(target),...
      -ones(1,N1),zeros(1,N1), 1, target', ...
      0, zeros(N1,1),slack*ones(N1,1));
   fim=clock;
   tempo=etime(fim, in);
   t_quadprog=t_quadprog+tempo;
   save t_quadprog t_quadprog;

% identificacao dos support vectors (sv) e dos unbounded support
% vectors (usv)
sv  = find (alpha>0);
usv = find ((alpha>0) & (alpha<slack));
% calculo de (b), Equação (43)
b=0;
for j=1:size(usv,1)
   sum_i=0;
   for i=1:size(sv,1)
      sum_i=sum_i+alpha(sv(i))*target(sv(i)) ...
         *kernel_matrix(sv(i),usv(j));
   end
   b=target(usv(j))-sum_i;
end
b=b/size(usv,1);

svm_par(nos).alpha=alpha;
svm_par(nos).usv=usv;
svm_par(nos).sv=sv;
svm_par(nos).b=b;
```

```
function [no]= buildCAB(kernel, param, limiar, auxil, nos)

%-------------------------------------------------------------------
%
%determina se a classe vai para o noh da direita ou esquerda, ou ambos
%
%-------------------------------------------------------------------
load var_nos;
load classes;
load x;
```

```
load auxil;

feat=var_nos(nos).feat;
feat=sort(feat); %ordena as bandas selecionadas
clas_teste=find(classes); %classes que não são zero
m=1;
n=1;
no(1).esq=[]; %no da esquerda
no(2).dir=[]; %no da direita
no(1).esq(m)=var_nos(nos).bhatt.i; %no da esquerda
no(2).dir(n)=var_nos(nos).bhatt.j; %no da direita

for k=1:size(clas_teste,2)
   teste=[];
%seleciona amostras para teste segundo feat
   for g=1:(size(x(classes(clas_teste(k))).amostra,1))
      y=[];
      for t=1:(size(feat,1))
         y=[y; x(classes(clas_teste(k))).amostra(g,feat(t))];
      end
      teste=[teste; y'];
   end

%se saida -1 ->classe i, se saida 1 -> classe j
   [class]=svm_classify(var_nos(nos).train,var_nos(nos).target,  teste,
kernel, param, nos);
   saida=class>0; %amostras com valor -1, ficam com 0
% tam_clas determina o numero de amostras por classe das classes em
% teste
   tam_clas(k)=size(x(classes(clas_teste(k))).amostra, 1);
% var_nod devolve o nº de padroes que pertencem `a classe j para cada
% classe k
   var_nod(k)=size((find(saida)), 1);
% var_noe devolve o nº de padroes que pertencem `a classe i para cada
% classe k
   var_noe(k)=tam_clas(k)-var_nod(k);
   if (var_nod(k)~=0)
      perc_d(k)=(var_nod(k)/tam_clas(k))*100;
   else
      perc_d(k)=0;
   end

   if (var_noe(k)~=0)
      perc_e(k)=(var_noe(k)/tam_clas(k))*100;
   else
      perc_e(k)=0;
   end

% se percentual esquerda maior q direita e que limiar, classe vai para
% esquerda
   if ((perc_e(k)>perc_d(k)) & (perc_e(k)>=limiar))
      m=m+1;
      no(1).esq(m)= classes(clas_teste(k));

% se percentual direita maior q esquerda e que limiar, classe vai para
% direita
   elseif ((perc_d(k)>perc_e(k)) & (perc_d(k)>=limiar))
      n=n+1;
      no(2).dir(n) = classes(clas_teste(k));

% caso perc nao seja maior q limiar, a classe vai para ambos os nos
```

```
   elseif ((perc_d(k)<=limiar)|(perc_e(k)<=limiar))
      m=m+1;
      n=n+1;
      no(1).esq(m)=classes(clas_teste(k));
      no(2).dir(n)=classes(clas_teste(k));
      auxil=auxil+1;
      save auxil auxil;
   end
end
save auxil auxil;
```

```
function [class]=svm_classify(train, target, test, kernel, param, nos)

%-------------------------------------------------------------------
%
% fase de treinamento: classifica as amostras das demais classes (as
% que nao participaram para formacao do plano de separacao) para
% posterior atribuicao de tal classe a um dos nos descendentes
% fase de teste: classifica as amostras de teste de acordo com a
% função de decisao, Equacao(42)
%
%-------------------------------------------------------------------

load var_nos;

class=[];
for sample = 1:size(test,1)
   dx=0;
   for i =1:size(var_nos(nos).svm_par_train.sv,1)
      switch kernel
         case 'RBF'
            dif = test(sample,:) – ...
               train(var_nos(nos).svm_par_train.sv(i),:);
            dx = dx + ...
               var_nos(nos).svm_par_train.alpha ...
               (var_nos(nos).svm_par_train.sv(i))* ...
               target(var_nos(nos).svm_par_train.sv(i)) * ...
               exp(-param*dif*dif');
         case 'Poly'
            dx = dx + ...
               var_nos(nos).svm_par_train.alpha ...
               (var_nos(nos).svm_par_train.sv(i))* ...
               target(var_nos(nos).svm_par_train.sv(i))*...
               ((train(var_nos(nos).svm_par_train.sv(i),:)*...
               test(sample,:)'+1).^param);
      end
   end
   dx = dx + (var_nos(nos).svm_par_train.b);
   if  dx>0
      class(sample,1)=1;
   else
      class(sample,1)=-1;
   end
end

save class class;
```

4 FUNÇÕES PARA TESTE DA ÁRVORE (CAB-SVM)

```
function c=classify(kernel, param)

%-------------------------------------------------------------------
%
% classifica as amostras de teste em suas respectivas classes (ou nos)
%
% Exemplo de sintaxe: classify('RBF', 2)
%
%-------------------------------------------------------------------

inicio=clock; %inicio
load teste; % carrega amostras de teste
% configuracoes da arvore - nos q contem dados
% 'var_nos(nos).svm_par_train' de treinamento por svm_train
load var_nos;
load nos_svm;
load nnn; % no validos 'nnn = nos nao nulos'
tabela=zeros(size(teste, 2), size(teste, 2));
save tabela tabela;
p=size(terminal,1);

for i=1:(size(teste, 2)) %numero de classes de teste

   for m=1:(terminal(p))
      tree(m).amostras=[];
   end

%todas amostras no no inicial
   amostras(1).no=[];
   for j=1:size(teste(i).amostra,1)
      amostras(1).no=[amostras(1).no,j];
   end
   aux=0; %numero de amostras nos nos terminais
   nos=1;

   while nos<=size(var_nos, 2) %inicio da arvore para cada amostra
      flag=0;
      for k=1:(size(nos_svm,1)) %numero de nos validos para svm
         if nos==nos_svm(k)
            [saida_svm, nof]=classvm(amostras(nos).no, i, k, kernel,
param);
            tree(nos).resultado=saida_svm;
            tree(nos*2+1).dir=nof(2).dir;
            tree(nos*2).esq=nof(1).esq;
            amostras(nos*2).no=nof(1).esq;
            amostras(nos*2+1).no=nof(2).dir;
            flag=1;
         end
      end

      if flag==0  %no terminal
         for k=1:size(nnn,2)
            if nos==nnn(k) %se no eh valido
               tree(nos).terminal=nos; %no terminal
%amostras no noh terminal
               tree(nos).amostras=amostras(nos).no;
               aux=aux+size(amostras(nos).no,2);
%se todas as amostras estao nos nos terminais, entao para.
```

```
               if aux==size(amostras(1).no, 2)
                  nos=999;
               end
            end
         end
      end
      nos=nos+1;
   end

%forma e salva tabela de contingencia
   switch i
      case 1
         tree1=tree;
         save tree1 tree1; %arvore das amostras classificadas
         acuracia(tree1, i); %calculo da confusion matrix
      case 2
         tree2=tree;
         save tree2 tree2;
         acuracia(tree2, i);
      case 3
         tree3=tree;
         save tree3 tree3;
         acuracia(tree3, i);
      case 4
         tree4=tree;
         save tree4 tree4;
         acuracia(tree4, i);
      case 5
         tree5=tree;
         save tree5 tree5;
         acuracia(tree5, i);
      case 6
         tree6=tree;
         save tree6 tree6;
         acuracia(tree6, i);
   end
end

load tabela;
acuracia_media=sum(diag(tabela))/sum(sum(tabela))

fim=clock;
tempo_de_proces_classificacao = etime(fim, inicio)
save tempo_de_proces_classificacao tempo_de_proces_classificacao;
```

```
function [out_svm, nof]= classvm(amostras, i, k, kernel, param)

%-------------------------------------------------------------------------
%
% responsavel pela classificacao das amostras em classe "a" ou "b"
%
%-------------------------------------------------------------------------

load var_nos;
load nos_svm;
load teste;

feat=var_nos(nos_svm(k)).feat;
feat=sort(feat); %ordena as bandas selecionadas

m=1;
```

```
n=1;
nof(1).esq=[]; %no da esquerda
nof(2).dir=[]; %no da direita
test=[];

for j=1:size(amostras, 2)
   y=[];
   for t=1:(size(feat,1))
      y=[y; teste(i).amostra(amostras(j),feat(t))];
   end
      test=[test; y'];
end

[out_svm] = svm_classify(var_nos(nos_svm(k)).train,
var_nos(nos_svm(k)).target, test, kernel, param, nos_svm(k));

for i=1:size(out_svm, 1)
   if  out_svm(i)==1 %amostra para direita
      nof(2).dir(n)=amostras(i);
      n=n+1;
   else %amostra para esquerda
      nof(1).esq(m)=amostras(i);
      m=m+1;
   end
end
```

```
function [class]=svm_classify(train, target, test, kernel, param, nos)
```

```
function f=acuracia(arvore, amostra)

%-------------------------------------------------------------------------
%
% calculo tabela de contingencia
%
%-------------------------------------------------------------------------

load var_nos;
load terminal;
load teste;
load tabela;
flag=0;
t=0;
i=0;

while t==0
i=i+1;
   if (amostra==var_nos(terminal(i)).classe)
      if tabela(amostra, amostra)>0
         tabela(amostra,amostra)=size(arvore(terminal(i)).amostras,
2)+tabela(amostra, amostra);
      else
         tabela(amostra,amostra)=size(arvore(terminal(i)).amostras,2);
      end
      if (size(teste(amostra).amostra, 1)==(tabela(amostra, amostra)))
         flag=1;
      end

   elseif (flag==0)
      c=var_nos(terminal(i)).classe;
      if tabela(c, amostra)>0
```

```
         tabela(c, amostra)=size(arvore(terminal(i)).amostras,
2)+tabela(c, amostra);
      else
         tabela(c, amostra)=size(arvore(terminal(i)).amostras, 2);
      end
   end
   if (size(teste(amostra).amostra, 1)==(sum(tabela(:, amostra))))
       t=1;
   end
end

save tabela tabela;
```

5 FUNÇÕES PARA TREINAMENTO DA ÁRVORE (CAB-MVG)

```
function w=cab_mvg(dim, n_bandas, limiar)

%-------------------------------------------------------------------
%
% Classificador em árvore binária utilizando a função de decisao MVG
%   dim: numero de bandas a serem selecionadas por sfs
%   n_bandas: numero de bandas utilizadas para calculo da distancia de
% Bhattacharyya para decidir as classes que darão origem aos nos
% descendentes
%   limiar: Limiar em escala percentual (0 a 100) para atribuição de
% classes nos nós da árvore no processo de classificação
%
% exemplo de sintaxe: cab_mvg(20, 80, 99);
%
%-------------------------------------------------------------------

inicio=clock; %inicio
load x; %amostras

auxil=0;
save auxil auxil;
nos=1;
temp=1;
step=round(size(x(1).amostra, 2)/n_bandas);
nnn=[1]; %nos nao nulos
terminal=[]; %nos validos e terminais
n_mvg=[]; %nos validos e nao terminais

if (limiar<0) | (limiar>100)
   error('Valor de limiar deve ficar entre 0 e 100')
end
if (dim>size(x(1).amostra, 2))
    disp(['Numero maximo de bandas disponiveis:' ...
         ... num2str(size(x(1).amostra, 2))])
    error('dimensao deve ser menor que numero total de bandas')
end
if (n_bandas>size(x(1).amostra, 1))
    disp(['Numero de amostras de treinamento:' ...
         ... num2str(size(x(1).amostra, 1))])
    warning('n_bandas deveria ser menor que numero de amostras
         ... disponiveis para treinamento')
end
```

```
while nos~=1000
   if nos==1 % Atribuicao de todas as classes no no um
      arvore_mvg(1).no=[];
      for i=1:size(x,2)
         arvore_mvg(1).no=[arvore_mvg(1).no,i];
      end
   end

   if size(arvore_mvg(nos).no,2)>=3  %classes no no >=3
      disp(['calculo do no: ' num2str(nos)])
      n_mvg=[n_mvg; nos];
      in_distancia=clock;
% calculo da distância de Bhattacharyya
      [bhatt]=distance(arvore_mvg(nos).no, step, nos);
      mvg_nos(nos).bhatt=bhatt(nos);
      mvg_nos(nos).classe=arvore_mvg(nos).no;
      fim_distancia=clock;
      tempo_distancia=etime(fim_distancia, in_distancia);
      in_sfs=clock;
      [feat]=sfs(nos, bhatt, dim);
      mvg_nos(nos).feat=feat;
      fim_sfs=clock;
      tempo_sfs=etime(fim_sfs, in_sfs)
      in_mvg=clock;
      [amostras1, amostras2]=stand_mvg(nos, bhatt, feat);
      save mvg_nos mvg_nos;
      [media1,   media2,   covar1,   covar2,   no]=mvg(nos,   amostras1,
amostras2, feat, limiar);
      mvg_nos(nos).media1=media1;
      mvg_nos(nos).media2=media2;
      mvg_nos(nos).covar1=covar1;
      mvg_nos(nos).covar2=covar2;
      fim_mvg=clock;
      tempo_mvg=etime(fim_mvg, in_mvg)
% atualiza valor das classes nos nos filhos (no direito e esquerdo)
      arvore_mvg(nos*2).no=no(1).esq;
      nnn=[nnn, nos*2];
      arvore_mvg(nos*2+1).no=no(2).dir;
      nnn=[nnn, nos*2+1];
      temp=nos*2+1; %valor temporario de noh jah acessado

   elseif size(arvore_mvg(nos).no,2)==2  %classes no no ==2
      disp(['calculo do no: ' num2str(nos)])
      n_mvg=[n_mvg; nos];
      in_distancia=clock;
% calcula da distância de Bhattacharyya
      [bhatt]=distance(arvore_mvg(nos).no, step, nos);
      mvg_nos(nos).bhatt=bhatt(nos);
      mvg_nos(nos).classe=arvore_mvg(nos).no;
      fim_distancia=clock;
      tempo_distancia=etime(fim_distancia, in_distancia);
      in_sfs=clock;
      [feat]=sfs(nos, bhatt, dim);
      mvg_nos(nos).feat=feat;
      fim_sfs=clock;
      tempo_sfs=etime(fim_sfs, in_sfs)
      in_mvg=clock;
      [amostras1, amostras2]=stand_mvg(nos, bhatt, feat);
      save mvg_nos mvg_nos;
      [media1,   media2,   covar1,   covar2,   no]=mvg(nos,   amostras1,
amostras2, feat, limiar);
```

```
      mvg_nos(nos).media1=media1;
      mvg_nos(nos).media2=media2;
      mvg_nos(nos).covar1=covar1;
      mvg_nos(nos).covar2=covar2;
      fim_mvg=clock;
      tempo_mvg=etime(fim_mvg, in_mvg)
% Atualiza valor das classes nos nos filhos (no direito e esquerdo)
      arvore_mvg(nos*2).no=no(1).esq;
      nnn=[nnn, nos*2];
      arvore_mvg(nos*2+1).no=no(2).dir;
      nnn=[nnn, nos*2+1];
      temp=nos*2+1;

   elseif size(arvore_mvg(nos).no,2)==1  %classes no no terminal
      terminal=[terminal; nos];
      disp(['no terminal: ' num2str(nos)])
      mvg_nos(nos).classe=arvore_mvg(nos).no;
      cont=0;
      for aux2=1:temp
         if size(arvore_mvg(aux2).no,2)==1
            cont=cont+1;   %soma nos que contem apenas uma classe
         end
      end
% se nº de nos que contem apenas uma classe sao iguais ao nº de
% classes total + nº de vezes q as classes q foram para ambos os nos
      load auxil;
      if (cont==(size(arvore_mvg(1).no,2)+auxil))
         for i=(nos+1):temp
            if size(arvore_mvg(i).no,2)==1
               disp(['no terminal: ' num2str(i)])
               terminal=[terminal; i];
            end
%atualiza valor do no temp->nos*2+1
            mvg_nos(i).classe=arvore_mvg(i).no;
         end
         nos=999;
      end
   end
   nos=nos+1;
end

save mvg_nos mvg_nos;
save arvore_mvg arvore_mvg;
save nnn nnn; %nos nao nulos;
save terminal terminal;
save n_mvg n_mvg;

fim=clock;
tempo_de_processamento_mvg = etime(fim, inicio)
save tempo_de_processamento_mvg tempo_de_processamento_mvg;
```

```
function [bhatt]=distance(classes, step, nos)
```

```
function B=bhattacharyya(i, j)
```

```
function [feat]=sfs(nos, bhatt, dim)
```

```
function [amostras1, amostras2]=stand_mvg(nos, bhatt, feat)

%-------------------------------------------------------------------
%
% selecao das amostras para aplicação da função de decisão MVG
%
%-------------------------------------------------------------------

load x;

amostras1=[];
amostras2=[];
feat=sort(feat); %ordena as bandas selecionadas

for g=1:(size(x(bhatt(nos).i).amostra,1))
    y=[];
    for k=1:size(feat,1)
        y=[y; x(bhatt(nos).i).amostra(g,feat(k))];
    end
    amostras1=[amostras1; y'];
end
for g=1:(size(x(bhatt(nos).j).amostra,1))
    y=[];
    for k=1:size(feat,1)
        y=[y; x(bhatt(nos).j).amostra(g,feat(k))];
    end
    amostras2=[amostras2; y'];
end
```

```
function [media1, media2, covar1, covar2, no]=mvg(nos, amostra1,
amostra2, feat, limiar)

%-------------------------------------------------------------------
%
% implementa a função de decisão Maxima Verossimilhança Gaussiana
% (MVG) e determina a qual noh a classe deve ser atribuida
%
%-------------------------------------------------------------------

load mvg_nos;
load classes;
load x;
load auxil;

clas_teste=find(classes);
m=1;
n=1;
no(1).esq=[]; %no da esquerda
no(2).dir=[]; %no da direita
feat=sort(feat); %ordena as bandas selecionadas
no(1).esq(m)=mvg_nos(nos).bhatt.i; %no da esquerda
no(2).dir(n)=mvg_nos(nos).bhatt.j; %no da direita
covar1=cov(amostra1);
media1=mean(amostra1);
covar2=cov(amostra2);
media2=mean(amostra2);

for k=1:size(clas_teste,2)
   saida=[];
%demais classes a serem testadas jah com as bandas selecionadas
   amostra_teste=[];
```

```
%selecionas as bandas para demais classes a serem testadas
   for h=1:(size(x(classes(clas_teste(k))).amostra,1))
      y=[];
      for v=1:size(feat,1)
         y=[y; x(classes(clas_teste(k))).amostra(h,feat(v))];
      end
      amostra_teste=[amostra_teste; y'];
   end

   for pixel=1:size(amostra_teste, 1)
      g1=-(logdet(covar1))-((amostra_teste(pixel,:)-
media1)*inv(covar1)*(amostra_teste(pixel,:)-media1)');
      g2=-(logdet(covar2))-((amostra_teste(pixel,:)-
media2)*inv(covar2)*(amostra_teste(pixel,:)-media2)');

      if (g1>g2)
         saida(1,pixel)=0;
      else
         saida(1,pixel)=1;
      end
   end

% tam_clas determina o numero de amostras por classe das classes em
% teste
   tam_clas(k)=size(x(classes(clas_teste(k))).amostra, 1);
% var_nod devolve o nº de padroes que pertencem `a classe j para cada
% classe k
   var_nod(k)=size((find(saida)), 2);
% var_noe devolve o nº de padroes que pertencem `a classe i para cada
% classe k
   var_noe(k)=tam_clas(k)-var_nod(k);

   if (var_nod(k)~=0)
      perc_d(k)=(var_nod(k)/tam_clas(k))*100;
   else
      perc_d(k)=0;
   end

   if (var_noe(k)~=0)
      perc_e(k)=(var_noe(k)/tam_clas(k))*100;
   else
      perc_e(k)=0;
   end

% se percentual esquerda maior q direita e que limiar, classe vai para
% esquerda
   if ((perc_e(k)>perc_d(k)) & (perc_e(k)>=limiar))
      m=m+1;
      no(1).esq(m)= classes(clas_teste(k));
% se percentual direita maior q esquerda e que limiar, classe vai para
% direita
   elseif ((perc_d(k)>perc_e(k)) & (perc_d(k)>=limiar))
      n=n+1;
      no(2).dir(n) = classes(clas_teste(k));
%caso perc nao seja maior q limiar, a classe vai para ambos os nos
   elseif ((perc_d(k)<=limiar)|(perc_e(k)<=limiar))
      m=m+1;
      n=n+1;
      no(1).esq(m)=classes(clas_teste(k));
      no(2).dir(n)=classes(clas_teste(k));
      auxil=auxil+1;
```

```
      save auxil auxil;
   end
end

save auxil auxil;
```

6 FUNÇÕES PARA TESTE DAS AMOSTRAS (CAB-MVG)

```
function f=classify_mvg()

%-------------------------------------------------------------------------
%
% classifica as amostras de teste em suas respectivas classes (ou nos)
% Exemplo de sintaxe: classify_mvg()
%
%-------------------------------------------------------------------------

inicio=clock; %inicio

load teste;
load mvg_nos;
load nnn; %nos validos
load terminal;
load n_mvg;
tabela=zeros(size(teste, 2), size(teste, 2));
save tabela tabela;

for i=1:(size(teste, 2)) %numero de amostras de teste
   disp(['amostra numero: ' num2str(i)])
   t=[];
   amost(1).no=[];
   for j=1:size(teste(i).amostra,1)
      amost(1).no=[amost(1).no, j]; %todas amostras no no inicial
   end

   aux=0; %numero de amostras nos nos terminais
   nos=1;
   while nos<=size(mvg_nos, 2) % inicio da arvore para cada amostra
      flag=0;
      for j=1:(size(n_mvg, 1))
         if nos==n_mvg(j)
            [noh]=decisao(mvg_nos(nos).covar1, mvg_nos(nos).covar2,...
               ...mvg_nos(nos).media1, mvg_nos(nos).media2, ...
               ...mvg_nos(nos).feat, amost(nos).no, i);
            tree_mvg(nos*2).esq=noh(1).esq;
            tree_mvg(nos*2+1).dir=noh(2).dir;
            amost(nos*2).no=noh(1).esq;
            amost(nos*2+1).no=noh(2).dir;
            flag=1;
         end
      end
      if flag==0  %no terminal
         for k=1:size(terminal,1)
            if nos==terminal(k) %nos terminais
               if (size(amost(nos).no,2))~=0
                  t=[t; nos];
                  disp(['no final: ' num2str(nos)])
                  tree_mvg(nos).terminal=nos; %no terminal
```

```
%amostras no noh terminal
                  tree_mvg(nos).amostras=amost(nos).no;
                  aux=aux+size(amost(nos).no,2);
%se todas as amostras estao nos nos terminais, entao para.
                  if aux==size(amost(1).no, 2)
                     nos=999;
                  end
               end
            end
         end
      end
   nos=nos+1;
   end

%calcula tabela de contingencia p/ 6 classes
   switch i
      case 1
         tree1_mvg=tree_mvg;
         save tree1_mvg tree1_mvg;
         [acur]=acuracia_mvg(tree1_mvg, i, t);
      case 2
         tree2_mvg=tree_mvg;
         save tree2_mvg tree2_mvg;
         [acur]=acuracia_mvg(tree2_mvg, i, t);
      case 3
         tree3_mvg=tree_mvg;
         save tree3_mvg tree3_mvg;
         [acur]=acuracia_mvg(tree3_mvg, i, t);
      case 4
         tree4_mvg=tree_mvg;
         save tree4_mvg tree4_mvg;
         [acur]=acuracia_mvg(tree4_mvg, i, t);
      case 5
         tree5_mvg=tree_mvg;
         save tree5_mvg tree5_mvg;
         [acur]=acuracia_mvg(tree5_mvg, i, t);
      case 6
         tree6_mvg=tree_mvg;
         save tree6_mvg tree6_mvg;
         [acur]=acuracia_mvg(tree6_mvg, i, t);
   end
end

load tabela;
acuracia_media=sum(diag(tabela))/sum(sum(tabela))
fim=clock;
tempo_de_proces_classificacao_mvg = etime(fim, inicio)
save tempo_de_proces_classificacao_mvg
tempo_de_proces_classificacao_mvg
```

```
function [noh]=decisao(covar1, covar2, media1, media2, feicoes,
amostras, classe)

%-------------------------------------------------------------------
%
% decide se as amostras devem ir para noh esquerdo ou direito
%
%-------------------------------------------------------------------

load teste;
m=1;
```

```
n=1;
noh(1).esq=[]; %no da esquerda
noh(2).dir=[]; %no da direita
%demais classes a serem testadas jah com as bandas selecionadas
amostra_teste=[];
feicoes=sort(feicoes); %ordena as bandas selecionadas

%seleciona as bandas para demais classes a serem testadas
for h=1:(size(amostras,2))
   y=[];
   for v=1:size(feicoes,1)
      y=[y; teste(classe).amostra(amostras(h),feicoes(v))];
   end
   amostra_teste=[amostra_teste; y'];
end

saida=[];
for pixel=1:size(amostra_teste, 1)
   g1=-(logdet(covar1))-((amostra_teste(pixel,:)-
media1)*inv(covar1)*(amostra_teste(pixel,:)-media1)');
   g2=-(logdet(covar2))-((amostra_teste(pixel,:)-
media2)*inv(covar2)*(amostra_teste(pixel,:)-media2)');

   if (g1>g2)
      saida(1,pixel)=0;
   else
      saida(1,pixel)=1;
   end
end

for i=1:size(saida, 2)
   if  saida(i)==1
      noh(2).dir(n)=amostras(i);
      n=n+1;
   else %amostra para esquerda
      noh(1).esq(m)=amostras(i);
      m=m+1;
   end
end
```

```
function [total]=acuracia_mvg(arvore, amostra, no_term)

%-------------------------------------------------------------------------
%
% calculo tabela de contingencia
%
%-------------------------------------------------------------------------

load mvg_nos;
load teste;
load tabela;

%calculo da acuracia
flag=0;
total=0;
for i=1:size(no_term,1)
   if (amostra==mvg_nos(no_term(i)).classe)
      if total>0
         aux=size(arvore(no_term(i)).amostras, 2)+aux;
      else
         aux=size(arvore(no_term(i)).amostras, 2);
```

```
      end
      total=(aux/size(teste(amostra).amostra, 1))*100;
      flag=1;
   end
end
if flag==0
   warning(['acuracia impossivel!!!'])
   total=0;
end

%calculo tabela de contingencia
bdr=0;
t=0;
i=0;
while t==0
   i=i+1;
   if (amostra==mvg_nos(no_term(i)).classe)
      if tabela(amostra, amostra)>0
         tabela(amostra, amostra)=size(arvore(no_term(i)).amostras,
2)+tabela(amostra, amostra);
      else
         tabela(amostra, amostra)=size(arvore(no_term(i)).amostras,
2);
      end
      if (size(teste(amostra).amostra, 1)==(tabela(amostra, amostra)))
         bdr=1;
   end
   elseif (bdr==0)
      c=mvg_nos(no_term(i)).classe;
      if tabela(c, amostra)>0
         tabela(c, amostra)=size(arvore(no_term(i)).amostras,
2)+tabela(c, amostra);
      else
         tabela(c, amostra)=size(arvore(no_term(i)).amostras, 2);
      end
   end
   if (size(teste(amostra).amostra, 1)==(sum(tabela(:, amostra))))
      t=1;
   end
end

save tabela tabela;
```

APÊNDICE B

TABELAS DE CONTINGÊNCIA

1 TABELAS – 50 AMOSTRAS DE TREINAMENTO

Tabela B1 – CAB-MVG com 10 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	262	24	0	17	43	3	75,07%
ω_2 – corn notill	12	241	0	5	34	30	74,84%
ω_3 – grass trees	0	0	300	1	2	4	97,72%
ω_4 – soybean clean	7	0	0	262	9	1	93,91%
ω_5 – soybean min	14	30	0	12	200	13	74,35%
ω_6 – soybean notill	5	5	0	3	12	249	90,88%
Acurácia do Produtor	87,33%	80,33%	100,00%	87,33%	66,67%	83,00%	
Acurácia Média							84,11%

Tabela B2 – CAB-MVG com 20 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	261	30	0	14	47	4	73,31%
ω_2 – corn notill	14	233	0	4	33	30	74,20%
ω_3 – grass trees	0	2	300	0	2	4	97,40%
ω_4 – soybean clean	6	0	0	259	7	0	95,22%
ω_5 – soybean min	13	27	0	20	197	20	71,12%
ω_6 – soybean notill	6	8	0	3	14	242	88,64%
Acurácia do Produtor	87,00%	77,67%	100,00%	86,33%	65,67%	80,67%	
Acurácia Média							82,89%

Tabela B3 – CAB-MVG com 40 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	269	48	0	55	67	21	58,48%
ω_2 – corn notill	13	214	0	2	25	53	69,71%
ω_3 – grass trees	0	1	294	2	2	3	97,35%
ω_4 – soybean clean	8	0	0	218	3	1	94,78%
ω_5 – soybean min	7	26	2	21	189	18	71,86%
ω_6 – soybean notill	3	11	4	2	14	204	85,71%
Acurácia do Produtor	89,67%	71,33%	98,00%	72,67%	63,00%	68,00%	
Acurácia Média							77,11%

Tabela B4 – CAB-SVM com 20 bandas espectrais e kernel polinomial grau 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	283	34	0	37	38	9	70,57%
ω_2 – corn notill	2	231	0	5	18	6	88,17%
ω_3 – grass trees	5	5	297	8	7	12	88,92%
ω_4 – soybean clean	4	1	0	234	3	0	96,69%
ω_5 – soybean min	5	20	0	14	221	10	81,85%
ω_6 – soybean notill	1	9	3	2	13	263	90,38%
Acurácia do Produtor	94,33%	77,00%	99,00%	78,00%	73,67%	87,67%	
Acurácia Média							84,94%

Tabela B5 – CAB-SVM com 40 bandas espectrais e kernel polinomial grau 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	285	36	0	46	51	12	66,28%
ω_2 – corn notill	4	236	0	4	23	9	85,51%
ω_3 – grass trees	4	4	298	6	4	12	90,85%
ω_4 – soybean clean	5	0	0	237	5	1	95,56%
ω_5 – soybean min	2	13	0	4	204	7	88,70%
ω_6 – soybean notill	0	11	2	3	13	259	89,93%
Acurácia do Produtor	95,00%	78,67%	99,33%	79,00%	68,00%	86,33%	
Acurácia Média							84,39%

Tabela B6 – CAB-SVM com 60 bandas espectrais e kernel polinomial grau 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	281	28	0	45	52	8	67,87%
ω_2 – corn notill	4	247	0	6	23	9	85,47%
ω_3 – grass trees	6	4	298	10	5	12	88,96%
ω_4 – soybean clean	5	0	0	236	3	1	96,33%
ω_5 – soybean min	1	10	0	2	206	8	90,75%
ω_6 – soybean notill	3	11	2	1	11	262	90,34%
Acurácia do Produtor	93,67%	82,33%	99,33%	78,67%	68,67%	87,33%	
Acurácia Média							85,00%

Tabela B7 – CAB-SVM com 80 bandas espectrais e kernel polinomial grau 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	280	31	0	54	51	13	65,27%
ω_2 – corn notill	9	239	0	7	20	7	84,75%
ω_3 – grass trees	4	4	299	11	5	13	88,99%
ω_4 – soybean clean	3	2	0	218	4	0	96,04%
ω_5 – soybean min	1	13	0	6	208	8	88,14%
ω_6 – soybean notill	3	11	1	4	12	259	89,31%
Acurácia do Produtor	93,33%	79,67%	99,67%	72,67%	69,33%	86,33%	
Acurácia Média							83,50%

Tabela B8 – CAB-SVM com 100 bandas espectrais e kernel polinomial grau 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	283	35	0	31	53	10	68,69%
ω_2 – corn notill	8	237	0	5	13	6	88,10%
ω_3 – grass trees	3	4	299	17	5	13	87,68%
ω_4 – soybean clean	2	2	0	239	5	2	95,60%
ω_5 – soybean min	1	13	0	4	213	11	88,02%
ω_6 – soybean notill	3	9	1	4	11	258	90,21%
Acurácia do Produtor	94,33%	79,00%	99,67%	79,67%	71,00%	86,00%	
Acurácia Média							84,94%

Tabela B9 – CAB-SVM com 120 bandas espectrais e kernel polinomial grau 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	278	28	0	21	32	3	76,80%
ω_2 – corn notill	7	229	0	3	12	5	89,45%
ω_3 – grass trees	4	4	298	17	8	14	86,38%
ω_4 – soybean clean	4	3	0	244	7	3	93,49%
ω_5 – soybean min	4	20	0	6	226	10	84,96%
ω_6 – soybean notill	3	16	2	9	15	265	85,48%
Acurácia do Produtor	92,67%	76,33%	99,33%	81,33%	75,33%	88,33%	
Acurácia Média							85,56%

Tabela B10 – CAB-SVM com 140 bandas espectrais e kernel polinomial grau 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	278	26	0	22	31	4	77,01%
ω_2 – corn notill	8	230	0	2	14	4	89,15%
ω_3 – grass trees	3	4	299	13	6	17	87,43%
ω_4 – soybean clean	5	7	0	248	6	3	92,19%
ω_5 – soybean min	3	16	0	6	221	4	88,40%
ω_6 – soybean notill	3	17	1	9	22	268	83,75%
Acurácia do Produtor	92,67%	76,67%	99,67%	82,67%	73,67%	89,33%	
Acurácia Média							85,78%

Tabela B11 – CAB-SVM com 160 bandas espectrais e kernel polinomial grau 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	278	31	0	23	31	6	75,34%
ω_2 – corn notill	6	223	0	2	10	4	91,02%
ω_3 – grass trees	3	4	299	11	6	12	89,25%
ω_4 – soybean clean	7	5	0	249	2	2	93,96%
ω_5 – soybean min	3	23	0	8	231	5	85,56%
ω_6 – soybean notill	3	14	1	7	20	271	85,76%
Acurácia do Produtor	92,67%	74,33%	99,67%	83,00%	77,00%	90,33%	
Acurácia Média							86,17%

Tabela B12 – CAB-SVM com 180 bandas espectrais e kernel polinomial grau 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	284	38	0	29	34	6	72,63%
ω_2 – corn notill	4	223	0	1	11	5	91,39%
ω_3 – grass trees	4	4	300	11	6	14	88,50%
ω_4 – soybean clean	4	3	0	248	3	1	95,75%
ω_5 – soybean min	1	19	0	8	230	7	86,79%
ω_6 – soybean notill	3	13	0	3	16	267	88,41%
Acurácia do Produtor	94,67%	74,33%	100,00%	82,67%	76,67%	89,00%	
Acurácia Média							86,22%

Tabela B13 – CAB-SVM com 20 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	245	13	0	12	16	3	84,78%
ω_2 – corn notill	6	242	0	17	23	16	79,61%
ω_3 – grass trees	1	3	300	1	2	5	96,15%
ω_4 – soybean clean	13	7	0	235	2	3	90,38%
ω_5 – soybean min	27	19	0	18	239	3	78,10%
ω_6 – soybean notill	8	16	0	17	18	270	82,07%
Acurácia do Produtor	81,67%	80,67%	100,00%	78,33%	79,67%	90,00%	
Acurácia Média							85,06%

Tabela B14 – CAB-SVM com 40 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	244	8	0	15	11	5	86,22%
ω_2 – corn notill	10	238	0	14	20	8	82,07%
ω_3 – grass trees	1	1	299	2	2	7	95,83%
ω_4 – soybean clean	16	10	1	238	4	2	87,82%
ω_5 – soybean min	23	27	0	16	243	16	74,77%
ω_6 – soybean notill	6	16	0	15	20	262	82,13%
Acurácia do Produtor	81,33%	79,33%	99,67%	79,33%	81,00%	87,33%	
Acurácia Média							84,67%

Tabela B15 – CAB-SVM com 60 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	246	5	0	10	15	6	87,23%
ω_2 – corn notill	11	251	1	22	23	8	79,43%
ω_3 – grass trees	1	0	298	2	2	4	97,07%
ω_4 – soybean clean	15	7	1	239	2	5	88,85%
ω_5 – soybean min	21	20	0	12	239	14	78,10%
ω_6 – soybean notill	6	17	0	15	19	263	82,19%
Acurácia do Produtor	82,00%	83,67%	99,33%	79,67%	79,67%	87,67%	
Acurácia Média							85,33%

Tabela B16 – CAB-SVM com 80 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	250	2	0	11	13	6	88,65%
ω_2 – corn notill	12	250	3	18	24	11	78,62%
ω_3 – grass trees	0	0	296	2	2	2	98,01%
ω_4 – soybean clean	13	10	1	247	6	8	86,67%
ω_5 – soybean min	19	25	0	8	240	9	79,73%
ω_6 – soybean notill	6	13	0	14	15	264	84,62%
Acurácia do Produtor	83,33%	83,33%	98,67%	82,33%	80,00%	88,00%	
Acurácia Média							85,94%

Tabela B17 – CAB-SVM com 100 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	252	1	0	10	18	7	87,50%
ω_2 – corn notill	17	251	5	24	28	11	74,70%
ω_3 – grass trees	0	0	293	1	3	2	97,99%
ω_4 – soybean clean	10	9	2	244	4	9	87,77%
ω_5 – soybean min	16	23	0	4	231	7	82,21%
ω_6 – soybean notill	5	16	0	17	16	264	83,02%
Acurácia do Produtor	84,00%	83,67%	97,67%	81,33%	77,00%	88,00%	
Acurácia Média							85,28%

Tabela B18 – CAB-SVM com 120 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	245	3	0	9	13	7	88,45%
ω_2 – corn notill	21	250	7	28	33	15	70,62%
ω_3 – grass trees	0	0	291	1	3	2	97,98%
ω_4 – soybean clean	10	9	2	241	4	7	88,28%
ω_5 – soybean min	19	22	0	5	232	6	81,69%
ω_6 – soybean notill	5	16	0	16	15	263	83,49%
Acurácia do Produtor	81,67%	83,33%	97,00%	80,33%	77,33%	87,67%	
Acurácia Média							84,56%

Tabela B19 – CAB-SVM com 140 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	245	3	0	8	14	6	88,77%
ω_2 – corn notill	21	258	15	29	35	20	68,25%
ω_3 – grass trees	0	0	285	1	2	2	98,28%
ω_4 – soybean clean	11	8	0	242	2	4	90,64%
ω_5 – soybean min	17	17	0	5	234	7	83,57%
ω_6 – soybean notill	6	14	0	15	13	261	84,47%
Acurácia do Produtor	81,67%	86,00%	95,00%	80,67%	78,00%	87,00%	
Acurácia Média							84,72%

Tabela B20 – CAB-SVM com 160 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	244	3	0	9	14	6	88,41%
ω_2 – corn notill	27	258	15	29	37	22	66,49%
ω_3 – grass trees	0	0	285	1	2	1	98,62%
ω_4 – soybean clean	11	7	0	242	2	4	90,98%
ω_5 – soybean min	15	18	0	7	231	7	83,09%
ω_6 – soybean notill	3	14	0	12	14	260	85,81%
Acurácia do Produtor	81,33%	86,00%	95,00%	80,67%	77,00%	86,67%	
Acurácia Média							84,44%

Tabela B21 – CAB-SVM com 180 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	245	3	0	8	14	5	89,09%
ω_2 – corn notill	27	259	16	32	37	22	65,90%
ω_3 – grass trees	0	0	284	1	1	0	99,30%
ω_4 – soybean clean	10	6	0	241	2	4	91,63%
ω_5 – soybean min	14	17	0	6	232	5	84,67%
ω_6 – soybean notill	4	15	0	12	14	264	85,44%
Acurácia do Produtor	81,67%	86,33%	94,67%	80,33%	77,33%	88,00%	
Acurácia Média							84,72%

2 TABELAS – 99 AMOSTRAS DE TREINAMENTO

Tabela B22 – CAB-MVG com 10 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	239	10	0	5	160	0	57,73%
ω_2 – corn notill	16	265	0	0	8	9	88,93%
ω_3 – grass trees	0	1	297	1	1	2	98,34%
ω_4 – soybean clean	10	0	0	278	13	1	92,05%
ω_5 – soybean min	34	15	3	15	107	35	51,20%
ω_6 – soybean notill	1	9	0	1	11	253	92,00%
Acurácia do Produtor	79,67%	88,33%	99,00%	92,67%	35,67%	84,33%	
Acurácia Média							79,94%

Tabela B23 – CAB-MVG com 20 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	237	7	0	7	14	1	89,10%
ω_2 – corn notill	18	263	0	0	4	8	89,76%
ω_3 – grass trees	0	1	299	1	0	1	99,01%
ω_4 – soybean clean	8	0	0	276	4	0	95,83%
ω_5 – soybean min	36	26	1	14	267	30	71,39%
ω_6 – soybean notill	1	3	0	2	11	260	93,86%
Acurácia do Produtor	79,00%	87,67%	99,67%	92,00%	89,00%	86,67%	
Acurácia Média							89,00%

Tabela B24 – CAB-MVG com 40 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	213	12	0	11	10	2	85,89%
ω_2 – corn notill	15	237	0	1	8	11	87,13%
ω_3 – grass trees	0	1	297	0	1	1	99,00%
ω_4 – soybean clean	6	0	0	255	4	0	96,23%
ω_5 – soybean min	62	43	1	32	261	51	58,00%
ω_6 – soybean notill	4	7	2	1	16	235	88,68%
Acurácia do Produtor	71,00%	79,00%	99,00%	85,00%	87,00%	78,33%	
Acurácia Média							83,22%

Tabela B25 – CAB-MVG com 60 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	204	16	0	13	11	5	81,93%
ω_2 – corn notill	24	229	0	4	19	19	77,63%
ω_3 – grass trees	0	0	294	0	0	1	99,66%
ω_4 – soybean clean	5	2	0	240	9	0	93,75%
ω_5 – soybean min	65	44	0	40	246	52	55,03%
ω_6 – soybean notill	2	9	6	3	15	223	86,43%
Acurácia do Produtor	68,00%	76,33%	98,00%	80,00%	82,00%	74,33%	
Acurácia Média							79,78%

Tabela B 26 – CAB-MVG com 80 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	199	28	0	19	26	5	71,84%
ω_2 – corn notill	34	208	2	15	28	50	61,72%
ω_3 – grass trees	0	0	289	0	0	1	99,66%
ω_4 – soybean clean	4	1	0	192	4	2	94,58%
ω_5 – soybean min	56	50	3	61	213	46	49,65%
ω_6 – soybean notill	7	13	6	13	29	196	74,24%
Acurácia do Produtor	66,33%	69,33%	96,33%	64,00%	71,00%	65,33%	
Acurácia Média							72,06%

Tabela B27 – CAB-SVM com 20 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	252	5	0	1	25	4	87,80%
ω_2 – corn notill	14	269	0	0	7	11	89,37%
ω_3 – grass trees	3	2	300	7	6	138	65,79%
ω_4 – soybean clean	15	4	0	287	7	2	91,11%
ω_5 – soybean min	14	12	0	4	238	5	87,18%
ω_6 – soybean notill	2	8	0	1	17	140	83,33%
Acurácia do Produtor	84,00%	89,67%	100,00%	95,67%	79,33%	46,67%	
Acurácia Média							82,56%

Tabela B28 – CAB-SVM com 40 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	257	12	0	4	19	3	87,12%
ω_2 – corn notill	9	251	0	2	13	3	90,29%
ω_3 – grass trees	2	4	300	10	4	138	65,50%
ω_4 – soybean clean	21	3	0	278	11	6	87,15%
ω_5 – soybean min	8	17	0	3	234	10	86,03%
ω_6 – soybean notill	3	13	0	3	19	140	78,65%
Acurácia do Produtor	85,67%	83,67%	100,00%	92,67%	78,00%	46,67%	
Acurácia Média							81,11%

Tabela B29 – CAB-SVM com 60 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	261	10	0	1	25	1	87,58%
ω_2 – corn notill	8	242	0	1	7	4	92,37%
ω_3 – grass trees	2	5	300	9	3	137	65,79%
ω_4 – soybean clean	18	6	0	288	18	7	85,46%
ω_5 – soybean min	11	22	0	1	230	9	84,25%
ω_6 – soybean notill	0	15	0	0	17	142	81,61%
Acurácia do Produtor	87,00%	80,67%	100,00%	96,00%	76,67%	47,33%	
Acurácia Média							81,28%

Tabela B30 – CAB-SVM com 80 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	262	6	0	1	22	4	88,81%
ω_2 – corn notill	6	254	0	1	12	2	92,36%
ω_3 – grass trees	2	4	299	7	3	137	66,15%
ω_4 – soybean clean	18	6	0	287	20	10	84,16%
ω_5 – soybean min	11	17	0	3	230	9	85,19%
ω_6 – soybean notill	1	13	1	1	13	138	82,63%
Acurácia do Produtor	87,33%	84,67%	99,67%	95,67%	76,67%	46,00%	
Acurácia Média							81,67%

Tabela B31 – CAB-SVM com 100 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	270	9	0	3	30	6	84,91%
ω_2 – corn notill	6	255	0	0	8	4	93,41%
ω_3 – grass trees	1	2	299	7	4	5	94,03%
ω_4 – soybean clean	14	9	0	284	17	9	85,29%
ω_5 – soybean min	8	11	1	3	226	6	88,63%
ω_6 – soybean notill	1	14	0	3	15	270	89,11%
Acurácia do Produtor	90,00%	85,00%	99,67%	94,67%	75,33%	90,00%	
Acurácia Média							89,11%

Tabela B32 – CAB-SVM com 120 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	266	8	0	4	29	7	84,71%
ω_2 – corn notill	5	251	0	0	7	3	94,36%
ω_3 – grass trees	2	3	299	7	4	137	66,15%
ω_4 – soybean clean	18	12	0	283	23	10	81,79%
ω_5 – soybean min	8	11	1	3	222	6	88,45%
ω_6 – soybean notill	1	15	0	3	15	137	80,12%
Acurácia do Produtor	88,67%	83,67%	99,67%	94,33%	74,00%	45,67%	
Acurácia Média							81,00%

Tabela B33 – CAB-SVM com 140 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	261	5	0	1	30	5	86,42%
ω_2 – corn notill	4	256	0	0	13	3	92,75%
ω_3 – grass trees	2	3	300	7	4	139	65,93%
ω_4 – soybean clean	22	14	0	282	21	8	81,27%
ω_5 – soybean min	4	8	0	6	215	1	91,88%
ω_6 – soybean notill	7	14	0	4	17	144	77,42%
Acurácia do Produtor	87,00%	85,33%	100,00%	94,00%	71,67%	48,00%	
Acurácia Média							81,00%

Tabela B34 – CAB-SVM com 160 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	265	5	0	5	28	5	86,04%
ω_2 – corn notill	3	260	0	0	19	4	90,91%
ω_3 – grass trees	2	2	300	6	3	137	66,67%
ω_4 – soybean clean	21	12	0	282	21	7	82,22%
ω_5 – soybean min	7	8	0	5	213	6	89,12%
ω_6 – soybean notill	2	13	0	2	16	141	81,03%
Acurácia do Produtor	88,33%	86,67%	100,00%	94,00%	71,00%	47,00%	
Acurácia Média							81,17%

Tabela B35 – CAB-SVM com 180 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	268	5	0	4	30	5	85,90%
ω_2 – corn notill	2	266	0	1	17	6	91,10%
ω_3 – grass trees	2	2	300	6	3	137	66,67%
ω_4 – soybean clean	20	8	0	284	16	10	84,02%
ω_5 – soybean min	5	9	0	5	215	4	90,34%
ω_6 – soybean notill	3	10	0	0	19	138	81,18%
Acurácia do Produtor	89,33%	88,67%	100,00%	94,67%	71,67%	46,00%	
Acurácia Média							81,72%

Tabela B36 – CAB-SVM com 20 bandas espectrais e kernel RBF γ 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	250	9	0	10	11	6	87,41%
ω_2 – corn notill	8	110	0	4	8	1	83,97%
ω_3 – grass trees	1	2	300	2	1	2	97,40%
ω_4 – soybean clean	13	5	0	258	8	4	89,58%
ω_5 – soybean min	21	156	0	19	257	14	55,03%
ω_6 – soybean notill	7	18	0	7	15	273	85,31%
Acurácia do Produtor	83,33%	36,67%	100,00%	86,00%	85,67%	91,00%	
Acurácia Média							80,44%

Tabela B37 – CAB-SVM com 40 bandas espectrais e kernel RBF γ 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	241	8	0	11	9	7	87,32%
ω_2 – corn notill	14	250	1	5	7	3	89,29%
ω_3 – grass trees	0	1	299	2	1	2	98,03%
ω_4 – soybean clean	18	13	0	254	8	3	85,81%
ω_5 – soybean min	21	15	0	19	258	11	79,63%
ω_6 – soybean notill	6	13	0	9	17	274	85,89%
Acurácia do Produtor	80,33%	83,33%	99,67%	84,67%	86,00%	91,33%	
Acurácia Média							87,56%

Tabela B38 – CAB-SVM com 60 bandas espectrais e kernel RBF γ 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	240	6	0	15	12	7	85,71%
ω_2 – corn notill	17	251	2	7	7	5	86,85%
ω_3 – grass trees	0	0	298	2	1	3	98,03%
ω_4 – soybean clean	22	16	0	251	12	0	83,39%
ω_5 – soybean min	18	13	0	17	250	10	81,17%
ω_6 – soybean notill	3	14	0	8	18	275	86,48%
Acurácia do Produtor	80,00%	83,67%	99,33%	83,67%	83,33%	91,67%	
Acurácia Média							86,94%

Tabela B39 – CAB-SVM com 80 bandas espectrais e kernel RBF γ 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	245	7	0	14	11	6	86,57%
ω_2 – corn notill	12	255	2	5	10	5	88,24%
ω_3 – grass trees	0	0	298	2	0	3	98,35%
ω_4 – soybean clean	23	14	0	258	15	2	82,69%
ω_5 – soybean min	17	11	0	14	249	9	83,00%
ω_6 – soybean notill	3	13	0	7	15	275	87,86%
Acurácia do Produtor	81,67%	85,00%	99,33%	86,00%	83,00%	91,67%	
Acurácia Média							87,78%

Tabela B40 – CAB-SVM com 100 bandas espectrais e kernel RBF γ 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	246	9	0	15	14	7	84,54%
ω_2 – corn notill	13	257	1	3	9	4	89,55%
ω_3 – grass trees	0	0	297	2	0	2	98,67%
ω_4 – soybean clean	25	13	2	258	14	3	81,90%
ω_5 – soybean min	13	9	0	13	244	7	85,31%
ω_6 – soybean notill	3	12	0	9	19	277	86,56%
Acurácia do Produtor	82,00%	85,67%	99,00%	86,00%	81,33%	92,33%	
Acurácia Média							87,72%

Tabela B41 – CAB-SVM com 120 bandas espectrais e kernel RBF γ 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	248	6	0	15	13	6	86,11%
ω_2 – corn notill	14	260	1	6	11	4	87,84%
ω_3 – grass trees	0	0	297	2	0	2	98,67%
ω_4 – soybean clean	22	12	2	262	15	4	82,65%
ω_5 – soybean min	13	8	0	9	246	7	86,93%
ω_6 – soybean notill	3	14	0	6	15	277	87,94%
Acurácia do Produtor	82,67%	86,67%	99,00%	87,33%	82,00%	92,33%	
Acurácia Média							88,33%

Tabela B42 – CAB-SVM com 140 bandas espectrais e kernel RBF γ 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	242	7	0	12	14	6	86,12%
ω_2 – corn notill	15	259	1	5	8	5	88,40%
ω_3 – grass trees	0	0	295	2	0	2	98,66%
ω_4 – soybean clean	29	12	4	261	19	4	79,33%
ω_5 – soybean min	12	9	0	10	245	4	87,50%
ω_6 – soybean notill	2	13	0	10	14	279	87,74%
Acurácia do Produtor	80,67%	86,33%	98,33%	87,00%	81,67%	93,00%	
Acurácia Média							87,83%

Tabela B43 – CAB-SVM com 160 bandas espectrais e kernel RBF γ 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	243	6	0	10	14	6	87,10%
ω_2 – corn notill	13	260	1	6	12	5	87,54%
ω_3 – grass trees	0	0	290	2	0	1	98,98%
ω_4 – soybean clean	31	14	9	264	15	7	77,65%
ω_5 – soybean min	11	8	0	10	245	4	88,13%
ω_6 – soybean notill	2	12	0	8	14	277	88,50%
Acurácia do Produtor	81,00%	86,67%	96,67%	88,00%	81,67%	92,33%	
Acurácia Média							87,72%

Tabela B44 – CAB-SVM com 180 bandas espectrais e kernel RBF γ 2.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	241	6	0	10	12	6	87,64%
ω_2 – corn notill	13	258	1	4	10	5	88,66%
ω_3 – grass trees	0	0	288	2	0	0	99,31%
ω_4 – soybean clean	32	15	11	268	18	13	75,07%
ω_5 – soybean min	12	8	0	9	247	3	88,53%
ω_6 – soybean notill	2	13	0	7	13	273	88,64%
Acurácia do Produtor	80,33%	86,00%	96,00%	89,33%	82,33%	91,00%	
Acurácia Média							87,50%

3 TABELAS – 200 AMOSTRAS DE TREINAMENTO

Tabela B45 – CAB-MVG com 20 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	237	3	0	0	10	2	94,05%
ω_2 – corn notill	19	269	0	0	13	6	87,62%
ω_3 – grass trees	0	0	300	1	1	3	98,36%
ω_4 – soybean clean	16	4	0	287	10	0	90,54%
ω_5 – soybean min	25	16	0	11	254	31	75,37%
ω_6 – soybean notill	3	8	0	1	12	258	91,49%
Acurácia do Produtor	79,00%	89,67%	100,00%	95,67%	84,67%	86,00%	
Acurácia Média							89,17%

Tabela B46 – CAB-MVG com 40 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	240	10	0	2	14	3	89,22%
ω_2 – corn notill	15	262	0	0	12	6	88,81%
ω_3 – grass trees	0	0	299	0	2	3	98,36%
ω_4 – soybean clean	13	3	0	283	5	1	92,79%
ω_5 – soybean min	27	16	0	12	253	32	74,41%
ω_6 – soybean notill	5	9	1	3	14	255	88,85%
Acurácia do Produtor	80,00%	87,33%	99,67%	94,33%	84,33%	85,00%	
Acurácia Média							88,44%

Tabela B47 – CAB-MVG com 60 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	238	14	0	3	16	3	86,86%
ω_2 – corn notill	16	251	0	1	16	8	85,96%
ω_3 – grass trees	0	1	298	0	2	4	97,70%
ω_4 – soybean clean	7	2	0	275	6	1	94,50%
ω_5 – soybean min	36	19	0	17	243	38	68,84%
ω_6 – soybean notill	3	13	2	4	17	246	86,32%
Acurácia do Produtor	79,33%	83,67%	99,33%	91,67%	81,00%	82,00%	
Acurácia Média							86,17%

Tabela B48 – CAB-MVG com 80 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	235	11	0	3	18	2	87,36%
ω_2 – corn notill	17	243	0	2	12	14	84,38%
ω_3 – grass trees	0	1	298	0	2	2	98,35%
ω_4 – soybean clean	9	2	0	271	6	3	93,13%
ω_5 – soybean min	37	34	0	21	244	41	64,72%
ω_6 – soybean notill	2	9	2	3	18	238	87,50%
Acurácia do Produtor	78,33%	81,00%	99,33%	90,33%	81,33%	79,33%	
Acurácia Média							84,94%

Tabela B49 – CAB-MVG com 100 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	221	12	0	4	15	3	86,67%
ω_2 – corn notill	26	237	0	1	18	22	77,96%
ω_3 – grass trees	0	0	297	0	2	3	98,34%
ω_4 – soybean clean	7	1	0	264	7	0	94,62%
ω_5 – soybean min	43	39	0	27	243	48	60,75%
ω_6 – soybean notill	3	11	3	4	15	224	86,15%
Acurácia do Produtor	73,67%	79,00%	99,00%	88,00%	81,00%	74,67%	
Acurácia Média							82,56%

Tabela B50– CAB-MVG com 120 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	216	15	0	11	23	12	77,98%
ω_2 – corn notill	34	242	0	6	27	34	70,55%
ω_3 – grass trees	0	2	297	0	2	3	97,70%
ω_4 – soybean clean	9	1	0	243	6	1	93,46%
ω_5 – soybean min	39	31	0	36	228	49	59,53%
ω_6 – soybean notill	2	9	3	4	14	201	86,27%
Acurácia do Produtor	72,00%	80,67%	99,00%	81,00%	76,00%	67,00%	
Acurácia Média							79,28%

Tabela B51 – CAB-MVG com 140 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	222	21	0	17	28	16	73,03%
ω_2 – corn notill	30	217	0	11	29	42	65,96%
ω_3 – grass trees	0	0	293	0	2	2	98,65%
ω_4 – soybean clean	5	0	2	217	5	3	93,53%
ω_5 – soybean min	41	50	0	49	225	54	53,70%
ω_6 – soybean notill	2	12	5	6	11	183	83,56%
Acurácia do Produtor	74,00%	72,33%	97,67%	72,33%	75,00%	61,00%	
Acurácia Média							75,39%

Tabela B52 – CAB-MVG com 160 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	198	36	0	17	33	30	63,06%
ω_2 – corn notill	36	197	1	7	24	26	67,70%
ω_3 – grass trees	0	1	279	0	2	2	98,24%
ω_4 – soybean clean	17	4	8	218	11	7	82,26%
ω_5 – soybean min	47	51	3	53	216	62	50,00%
ω_6 – soybean notill	2	11	9	5	14	173	80,84%
Acurácia do Produtor	66,00%	65,67%	93,00%	72,67%	72,00%	57,67%	
Acurácia Média							71,17%

Tabela B53 – CAB-MVG com 180 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	200	41	0	27	35	51	56,50%
ω_2 – corn notill	30	178	5	24	30	39	58,17%
ω_3 – grass trees	1	1	232	0	3	0	97,89%
ω_4 – soybean clean	22	15	22	203	17	14	69,28%
ω_5 – soybean min	42	47	10	39	199	43	52,37%
ω_6 – soybean notill	5	18	31	7	16	153	66,52%
Acurácia do Produtor	66,67%	59,33%	77,33%	67,67%	66,33%	51,00%	
Acurácia Média							64,72%

Tabela B54 – CAB-SVM com 20 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	268	5	0	3	22	3	89,04%
ω_2 – corn notill	7	269	0	1	10	10	90,57%
ω_3 – grass trees	1	1	300	1	2	6	96,46%
ω_4 – soybean clean	7	0	0	287	8	3	94,10%
ω_5 – soybean min	15	17	0	8	250	25	79,37%
ω_6 – soybean notill	2	8	0	0	8	253	93,36%
Acurácia do Produtor	89,33%	89,67%	100,00%	95,67%	83,33%	84,33%	
Acurácia Média							90,39%

Tabela B55 – CAB-SVM com 40 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	274	6	0	4	19	2	89,84%
ω_2 – corn notill	4	268	0	2	10	19	88,45%
ω_3 – grass trees	1	3	299	2	1	5	96,14%
ω_4 – soybean clean	9	3	0	276	9	1	92,62%
ω_5 – soybean min	10	16	0	13	254	26	79,62%
ω_6 – soybean notill	2	4	1	3	7	247	93,56%
Acurácia do Produtor	91,33%	89,33%	99,67%	92,00%	84,67%	82,33%	
Acurácia Média							89,89%

Tabela B56 – CAB-SVM com 60 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	272	7	0	4	17	3	89,77%
ω_2 – corn notill	3	264	0	2	14	15	88,59%
ω_3 – grass trees	1	1	300	5	1	4	96,15%
ω_4 – soybean clean	12	0	0	279	7	2	93,00%
ω_5 – soybean min	9	22	0	8	250	32	77,88%
ω_6 – soybean notill	3	6	0	2	11	244	91,73%
Acurácia do Produtor	90,67%	88,00%	100,00%	93,00%	83,33%	81,33%	
Acurácia Média							89,39%

Tabela B57 – CAB-SVM com 80 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	275	7	0	14	15	1	88,14%
ω_2 – corn notill	5	268	0	3	15	12	88,45%
ω_3 – grass trees	1	1	300	4	1	4	96,46%
ω_4 – soybean clean	7	0	0	271	9	3	93,45%
ω_5 – soybean min	11	19	0	8	245	28	78,78%
ω_6 – soybean notill	1	5	0	0	15	252	92,31%
Acurácia do Produtor	91,67%	89,33%	100,00%	90,33%	81,67%	84,00%	
Acurácia Média							89,50%

Tabela B58 – CAB-SVM com 100 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	275	7	0	12	19	4	86,75%
ω_2 – corn notill	3	268	0	1	17	9	89,93%
ω_3 – grass trees	1	1	299	6	1	4	95,83%
ω_4 – soybean clean	6	2	0	266	8	2	93,66%
ω_5 – soybean min	12	14	1	13	243	22	79,67%
ω_6 – soybean notill	3	8	0	2	12	259	91,20%
Acurácia do Produtor	91,67%	89,33%	99,67%	88,67%	81,00%	86,33%	
Acurácia Média							89,44%

Tabela B59 – CAB-SVM com 120 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	280	10	0	15	15	3	86,69%
ω_2 – corn notill	2	266	0	2	14	6	91,72%
ω_3 – grass trees	1	3	300	5	1	4	95,54%
ω_4 – soybean clean	7	1	0	268	9	1	93,71%
ω_5 – soybean min	9	13	0	9	250	19	83,33%
ω_6 – soybean notill	1	7	0	1	11	267	93,03%
Acurácia do Produtor	93,33%	88,67%	100,00%	89,33%	83,33%	89,00%	
Acurácia Média							90,61%

Tabela B60 – CAB-SVM com 140 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	281	9	0	18	17	6	84,89%
ω_2 – corn notill	3	272	1	3	17	10	88,89%
ω_3 – grass trees	2	3	299	6	1	2	95,53%
ω_4 – soybean clean	5	1	0	261	9	1	94,22%
ω_5 – soybean min	8	12	0	10	246	26	81,46%
ω_6 – soybean notill	1	3	0	2	10	255	94,10%
Acurácia do Produtor	93,67%	90,67%	99,67%	87,00%	82,00%	85,00%	
Acurácia Média							89,67%

Tabela B61 – CAB-SVM com 160 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	277	8	0	11	19	3	87,11%
ω_2 – corn notill	5	274	0	2	14	9	90,13%
ω_3 – grass trees	3	2	300	7	1	4	94,64%
ω_4 – soybean clean	5	0	0	266	10	1	94,33%
ω_5 – soybean min	8	13	0	13	249	26	80,58%
ω_6 – soybean notill	2	3	0	1	7	257	95,19%
Acurácia do Produtor	92,33%	91,33%	100,00%	88,67%	83,00%	85,67%	
Acurácia Média							90,17%

Tabela B62 – CAB-SVM com 180 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	279	7	0	14	24	4	85,06%
ω_2 – corn notill	3	274	1	2	13	10	90,43%
ω_3 – grass trees	3	2	299	6	0	3	95,53%
ω_4 – soybean clean	5	0	0	264	7	0	95,65%
ω_5 – soybean min	9	15	0	12	249	21	81,37%
ω_6 – soybean notill	1	2	0	2	7	262	95,62%
Acurácia do Produtor	93,00%	91,33%	99,67%	88,00%	83,00%	87,33%	
Acurácia Média							90,39%

Tabela B63 – CAB-SVM com 20 bandas espectrais e kernel RBF γ 0.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	248	2	0	13	8	3	90,51%
ω_2 – corn notill	4	259	0	5	13	7	89,93%
ω_3 – grass trees	2	0	300	3	2	3	96,77%
ω_4 – soybean clean	10	2	0	257	1	1	94,83%
ω_5 – soybean min	28	14	0	15	266	3	81,60%
ω_6 – soybean notill	8	23	0	7	10	283	85,50%
Acurácia do Produtor	82,67%	86,33%	100,00%	85,67%	88,67%	94,33%	
Acurácia Média							89,61%

Tabela B64 – CAB-SVM com 40 bandas espectrais e kernel RBF γ 0.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	255	5	0	17	6	7	87,93%
ω_2 – corn notill	2	262	0	5	10	1	93,57%
ω_3 – grass trees	1	1	300	3	2	4	96,46%
ω_4 – soybean clean	11	1	0	258	3	1	94,16%
ω_5 – soybean min	24	11	0	14	266	1	84,18%
ω_6 – soybean notill	7	20	0	3	13	286	86,93%
Acurácia do Produtor	85,00%	87,33%	100,00%	86,00%	88,67%	95,33%	
Acurácia Média							90,39%

Tabela B65 – CAB-SVM com 60 bandas espectrais e kernel RBF γ 0.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	247	3	0	16	5	6	89,17%
ω_2 – corn notill	6	263	0	5	9	1	92,61%
ω_3 – grass trees	2	1	300	2	2	4	96,46%
ω_4 – soybean clean	15	2	0	258	3	1	92,47%
ω_5 – soybean min	24	10	0	13	268	2	84,54%
ω_6 – soybean notill	6	21	0	6	13	286	86,14%
Acurácia do Produtor	82,33%	87,67%	100,00%	86,00%	89,33%	95,33%	
Acurácia Média							90,11%

Tabela B66 – CAB-SVM com 80 bandas espectrais e kernel RBF γ 0.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	249	4	0	11	5	6	90,55%
ω_2 – corn notill	4	262	0	9	8	2	91,93%
ω_3 – grass trees	1	1	300	2	2	3	97,09%
ω_4 – soybean clean	17	3	0	261	4	1	91,26%
ω_5 – soybean min	21	11	0	12	269	1	85,67%
ω_6 – soybean notill	8	19	0	5	12	287	86,71%
Acurácia do Produtor	83,00%	87,33%	100,00%	87,00%	89,67%	95,67%	
Acurácia Média							90,44%

Tabela B67 – CAB-SVM com 100 bandas espectrais e kernel RBF γ 0.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	251	4	0	11	5	6	90,61%
ω_2 – corn notill	1	254	0	7	6	2	94,07%
ω_3 – grass trees	1	1	299	2	2	3	97,08%
ω_4 – soybean clean	18	4	0	262	4	1	90,66%
ω_5 – soybean min	19	16	0	10	271	2	85,22%
ω_6 – soybean notill	10	21	1	8	12	286	84,62%
Acurácia do Produtor	83,67%	84,67%	99,67%	87,33%	90,33%	95,33%	
Acurácia Média							90,17%

Tabela B68 – CAB-SVM com 120 bandas espectrais e kernel RBF γ 0.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	247	5	0	9	7	5	90,48%
ω_2 – corn notill	5	254	0	6	6	3	92,70%
ω_3 – grass trees	2	1	299	2	2	2	97,08%
ω_4 – soybean clean	17	4	0	266	6	1	90,48%
ω_5 – soybean min	20	15	0	11	267	3	84,49%
ω_6 – soybean notill	9	21	1	6	12	286	85,37%
Acurácia do Produtor	82,33%	84,67%	99,67%	88,67%	89,00%	95,33%	
Acurácia Média							89,94%

Tabela B69 – CAB-SVM com 140 bandas espectrais e kernel RBF γ 0.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	248	4	0	10	9	5	89,86%
ω_2 – corn notill	7	254	0	9	8	3	90,39%
ω_3 – grass trees	2	1	299	2	2	2	97,08%
ω_4 – soybean clean	15	2	0	262	7	3	90,66%
ω_5 – soybean min	19	18	0	10	264	3	84,08%
ω_6 – soybean notill	9	21	1	7	10	284	85,54%
Acurácia do Produtor	82,67%	84,67%	99,67%	87,33%	88,00%	94,67%	
Acurácia Média							89,50%

Tabela B70 – CAB-SVM com 160 bandas espectrais e kernel RBF γ 0.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	246	5	0	9	9	6	89,45%
ω_2 – corn notill	8	256	0	7	7	3	91,10%
ω_3 – grass trees	2	1	299	2	3	3	96,45%
ω_4 – soybean clean	16	2	0	264	5	2	91,35%
ω_5 – soybean min	19	15	0	10	266	3	84,98%
ω_6 – soybean notill	9	21	1	8	10	283	85,24%
Acurácia do Produtor	82,00%	85,33%	99,67%	88,00%	88,67%	94,33%	
Acurácia Média							89,67%

Tabela B71 – CAB-SVM com 180 bandas espectrais e kernel RBF γ 0.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	246	4	0	8	9	5	90,44%
ω_2 – corn notill	7	256	0	8	9	3	90,46%
ω_3 – grass trees	2	1	299	2	2	2	97,08%
ω_4 – soybean clean	17	4	0	266	6	2	90,17%
ω_5 – soybean min	19	17	0	9	264	4	84,35%
ω_6 – soybean notill	9	18	1	7	10	284	86,32%
Acurácia do Produtor	82,00%	85,33%	99,67%	88,67%	88,00%	94,67%	
Acurácia Média							89,72%

4 TABELAS – 300 AMOSTRAS DE TREINAMENTO

Tabela B72 – CAB-MVG com 20 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	263	12	0	4	17	2	88,26%
ω_2 – corn notill	11	270	0	0	16	11	87,66%
ω_3 – grass trees	0	0	300	1	0	4	98,36%
ω_4 – soybean clean	6	4	0	277	12	1	92,33%
ω_5 – soybean min	18	9	0	16	121	25	64,02%
ω_6 – soybean notill	2	5	0	2	134	257	64,25%
Acurácia do Produtor	87,67%	90,00%	100,00%	92,33%	40,33%	85,67%	
Acurácia Média							82,67%

Tabela B73 – CAB-MVG com 40 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	261	17	0	3	16	2	87,29%
ω_2 – corn notill	13	266	0	0	11	17	86,64%
ω_3 – grass trees	0	0	300	0	0	4	98,68%
ω_4 – soybean clean	6	4	0	288	9	0	93,81%
ω_5 – soybean min	17	11	0	8	259	24	81,19%
ω_6 – soybean notill	3	2	0	1	5	253	95,83%
Acurácia do Produtor	87,00%	88,67%	100,00%	96,00%	86,33%	84,33%	
Acurácia Média							90,39%

Tabela B74 – CAB-MVG com 60 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	262	23	0	3	14	2	86,18%
ω_2 – corn notill	15	257	0	1	17	12	85,10%
ω_3 – grass trees	0	0	300	0	0	6	98,04%
ω_4 – soybean clean	5	4	0	285	8	1	94,06%
ω_5 – soybean min	14	10	0	8	257	27	81,33%
ω_6 – soybean notill	4	6	0	3	4	252	93,68%
Acurácia do Produtor	87,33%	85,67%	100,00%	95,00%	85,67%	84,00%	
Acurácia Média							89,61%

Tabela B75 – CAB-MVG com 80 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	264	25	0	6	19	3	83,28%
ω_2 – corn notill	10	256	0	1	16	15	85,91%
ω_3 – grass trees	0	0	300	0	0	3	99,01%
ω_4 – soybean clean	7	1	0	284	3	1	95,95%
ω_5 – soybean min	15	16	0	6	254	29	79,38%
ω_6 – soybean notill	4	2	0	3	8	249	93,61%
Acurácia do Produtor	88,00%	85,33%	100,00%	94,67%	84,67%	83,00%	
Acurácia Média							89,28%

Tabela B76 – CAB-MVG com 100 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	253	23	0	6	25	3	81,61%
ω_2 – corn notill	15	254	0	2	9	22	84,11%
ω_3 – grass trees	0	1	300	0	0	2	99,01%
ω_4 – soybean clean	10	2	0	276	9	1	92,62%
ω_5 – soybean min	20	18	0	14	248	33	74,47%
ω_6 – soybean notill	2	2	0	2	9	239	94,09%
Acurácia do Produtor	84,33%	84,67%	100,00%	92,00%	82,67%	79,67%	
Acurácia Média							87,22%

Tabela B77 – CAB-MVG com 120 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	255	18	0	6	30	5	81,21%
ω_2 – corn notill	14	252	0	3	11	16	85,14%
ω_3 – grass trees	0	2	298	0	0	2	98,68%
ω_4 – soybean clean	10	1	1	276	12	2	91,39%
ω_5 – soybean min	16	21	0	12	240	37	73,62%
ω_6 – soybean notill	5	6	1	3	7	238	91,54%
Acurácia do Produtor	85,00%	84,00%	99,33%	92,00%	80,00%	79,33%	
Acurácia Média							86,61%

Tabela B78 – CAB-MVG com 140 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	252	22	0	6	34	4	79,25%
ω_2 – corn notill	18	245	0	3	12	22	81,67%
ω_3 – grass trees	0	1	295	0	0	4	98,33%
ω_4 – soybean clean	7	1	1	271	9	1	93,45%
ω_5 – soybean min	18	21	1	18	227	38	70,28%
ω_6 – soybean notill	5	10	3	2	18	231	85,87%
Acurácia do Produtor	84,00%	81,67%	98,33%	90,33%	75,67%	77,00%	
Acurácia Média							84,50%

Tabela B79 – CAB-MVG com 160 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	237	26	0	4	30	9	77,45%
ω_2 – corn notill	26	234	0	3	16	25	76,97%
ω_3 – grass trees	0	0	294	1	0	1	99,32%
ω_4 – soybean clean	10	2	0	276	11	2	91,69%
ω_5 – soybean min	22	29	2	13	226	30	70,19%
ω_6 – soybean notill	5	9	4	3	17	233	85,98%
Acurácia do Produtor	79,00%	78,00%	98,00%	92,00%	75,33%	77,67%	
Acurácia Média							83,33%

Tabela B80 – CAB-MVG com 180 bandas espectrais.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	249	22	0	8	28	16	77,09%
ω_2 – corn notill	16	234	0	6	21	27	76,97%
ω_3 – grass trees	0	0	291	1	0	1	99,32%
ω_4 – soybean clean	8	1	1	272	9	1	93,15%
ω_5 – soybean min	21	26	2	12	221	27	71,52%
ω_6 – soybean notill	6	17	6	1	21	228	81,72%
Acurácia do Produtor	83,00%	78,00%	97,00%	90,67%	73,67%	76,00%	
Acurácia Média							83,06%

Tabela B81 – CAB-SVM com 20 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	277	10	0	1	28	10	84,97%
ω_2 – corn notill	12	280	0	0	20	34	80,92%
ω_3 – grass trees	1	2	299	3	0	5	96,45%
ω_4 – soybean clean	6	2	0	290	8	0	94,77%
ω_5 – soybean min	4	4	0	5	237	16	89,10%
ω_6 – soybean notill	0	2	1	1	7	235	95,53%
Acurácia do Produtor	92,33%	93,33%	99,67%	96,67%	79,00%	78,33%	
Acurácia Média							89,89%

Tabela B82 – CAB-SVM com 40 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	278	8	0	6	23	5	86,88%
ω_2 – corn notill	12	279	0	1	25	23	82,06%
ω_3 – grass trees	0	1	298	6	0	3	96,75%
ω_4 – soybean clean	3	2	1	282	6	0	95,92%
ω_5 – soybean min	7	7	0	4	237	19	86,50%
ω_6 – soybean notill	0	3	1	1	9	250	94,70%
Acurácia do Produtor	92,67%	93,00%	99,33%	94,00%	79,00%	83,33%	
Acurácia Média							90,22%

Tabela B83 – CAB-SVM com 60 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	270	5	0	8	24	3	87,10%
ω_2 – corn notill	18	278	0	3	21	28	79,89%
ω_3 – grass trees	0	2	298	5	0	2	97,07%
ω_4 – soybean clean	3	2	1	273	3	3	95,79%
ω_5 – soybean min	8	10	1	10	246	10	86,32%
ω_6 – soybean notill	1	3	0	1	6	254	95,85%
Acurácia do Produtor	90,00%	92,67%	99,33%	91,00%	82,00%	84,67%	
Acurácia Média							89,94%

Tabela B84 – CAB-SVM com 80 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	278	5	0	5	15	3	90,85%
ω_2 – corn notill	11	276	0	6	21	26	81,18%
ω_3 – grass trees	0	3	299	5	4	4	94,92%
ω_4 – soybean clean	3	2	0	269	3	4	95,73%
ω_5 – soybean min	5	12	1	12	252	21	83,17%
ω_6 – soybean notill	3	2	0	3	5	242	94,90%
Acurácia do Produtor	92,67%	92,00%	99,67%	89,67%	84,00%	80,67%	
Acurácia Média							89,78%

Tabela B85 – CAB-SVM com 100 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	279	6	0	10	16	2	89,14%
ω_2 – corn notill	12	278	0	3	25	26	80,81%
ω_3 – grass trees	0	3	299	6	4	5	94,32%
ω_4 – soybean clean	3	2	0	267	5	3	95,36%
ω_5 – soybean min	5	9	1	10	245	17	85,37%
ω_6 – soybean notill	1	2	0	4	5	247	95,37%
Acurácia do Produtor	93,00%	92,67%	99,67%	89,00%	81,67%	82,33%	
Acurácia Média							89,72%

Tabela B86 – CAB-SVM com 120 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	278	4	0	10	15	1	90,26%
ω_2 – corn notill	9	282	0	1	29	23	81,98%
ω_3 – grass trees	0	4	300	4	3	4	95,24%
ω_4 – soybean clean	4	1	0	272	5	3	95,44%
ω_5 – soybean min	6	7	0	8	241	15	87,00%
ω_6 – soybean notill	3	2	0	5	7	254	93,73%
Acurácia do Produtor	92,67%	94,00%	100,00%	90,67%	80,33%	84,67%	
Acurácia Média							90,39%

Tabela B87 – CAB-SVM com 140 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	279	6	0	5	17	3	90,00%
ω_2 – corn notill	8	282	0	2	26	28	81,50%
ω_3 – grass trees	0	3	300	5	4	3	95,24%
ω_4 – soybean clean	6	2	0	273	5	3	94,46%
ω_5 – soybean min	4	6	0	12	244	15	86,83%
ω_6 – soybean notill	3	1	0	3	4	248	95,75%
Acurácia do Produtor	93,00%	94,00%	100,00%	91,00%	81,33%	82,67%	
Acurácia Média							90,33%

Tabela B88 – CAB-SVM com 160 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	275	5	0	7	18	2	89,58%
ω_2 – corn notill	8	283	0	3	29	29	80,40%
ω_3 – grass trees	1	3	300	9	2	5	93,75%
ω_4 – soybean clean	4	2	0	272	5	3	95,10%
ω_5 – soybean min	9	3	0	6	237	9	89,77%
ω_6 – soybean notill	3	4	0	3	9	252	92,99%
Acurácia do Produtor	91,67%	94,33%	100,00%	90,67%	79,00%	84,00%	
Acurácia Média							89,94%

Tabela B89 – CAB-SVM com 180 bandas espectrais e kernel polinomial grau 3.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	276	5	0	3	19	2	90,49%
ω_2 – corn notill	7	283	0	3	26	26	82,03%
ω_3 – grass trees	2	4	300	6	4	7	92,88%
ω_4 – soybean clean	3	1	0	270	5	3	95,74%
ω_5 – soybean min	9	3	0	14	239	9	87,23%
ω_6 – soybean notill	3	4	0	4	7	253	93,36%
Acurácia do Produtor	92,00%	94,33%	100,00%	90,00%	79,67%	84,33%	
Acurácia Média							90,06%

Tabela B90 – CAB-SVM com 20 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	196	9	0	9	15	6	83,40%
ω_2 – corn notill	7	263	0	2	10	1	92,93%
ω_3 – grass trees	0	0	300	3	1	3	97,72%
ω_4 – soybean clean	7	3	0	268	4	0	95,04%
ω_5 – soybean min	18	6	0	8	253	9	86,05%
ω_6 – soybean notill	72	19	0	10	17	281	70,43%
Acurácia do Produtor	65,33%	87,67%	100,00%	89,33%	84,33%	93,67%	
Acurácia Média							86,72%

Tabela B91 – CAB-SVM com 40 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	263	9	0	10	11	5	88,26%
ω_2 – corn notill	7	261	0	8	7	1	91,90%
ω_3 – grass trees	0	0	300	4	1	3	97,40%
ω_4 – soybean clean	11	3	0	264	3	1	93,62%
ω_5 – soybean min	13	10	0	7	265	10	86,89%
ω_6 – soybean notill	6	17	0	7	13	280	86,69%
Acurácia do Produtor	87,67%	87,00%	100,00%	88,00%	88,33%	93,33%	
Acurácia Média							90,72%

Tabela B92 – CAB-SVM com 60 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	254	6	0	11	11	5	88,50%
ω_2 – corn notill	8	258	0	6	10	1	91,17%
ω_3 – grass trees	0	0	300	3	1	2	98,04%
ω_4 – soybean clean	12	8	0	258	4	2	90,85%
ω_5 – soybean min	20	10	0	10	260	12	83,33%
ω_6 – soybean notill	6	18	0	12	14	278	84,76%
Acurácia do Produtor	84,67%	86,00%	100,00%	86,00%	86,67%	92,67%	
Acurácia Média							89,33%

Tabela B93 – CAB-SVM com 80 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	245	3	0	13	14	5	87,50%
ω_2 – corn notill	10	255	0	5	8	0	91,73%
ω_3 – grass trees	0	0	300	4	1	1	98,04%
ω_4 – soybean clean	15	10	0	253	3	2	89,40%
ω_5 – soybean min	24	14	0	12	262	13	80,62%
ω_6 – soybean notill	6	18	0	13	12	279	85,06%
Acurácia do Produtor	81,67%	85,00%	100,00%	84,33%	87,33%	93,00%	
Acurácia Média							88,56%

Tabela B94 – CAB-SVM com 100 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	244	3	0	12	10	5	89,05%
ω_2 – corn notill	8	256	0	7	9	0	91,43%
ω_3 – grass trees	1	0	299	3	1	1	98,03%
ω_4 – soybean clean	19	10	0	255	7	3	86,73%
ω_5 – soybean min	22	15	1	12	261	15	80,06%
ω_6 – soybean notill	6	16	0	11	12	276	85,98%
Acurácia do Produtor	81,33%	85,33%	99,67%	85,00%	87,00%	92,00%	
Acurácia Média							88,39%

Tabela B95 – CAB-SVM com 120 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	242	3	0	12	10	6	88,64%
ω_2 – corn notill	8	256	0	5	8	2	91,76%
ω_3 – grass trees	1	0	299	3	1	1	98,03%
ω_4 – soybean clean	22	7	0	259	9	3	86,33%
ω_5 – soybean min	21	17	1	9	260	13	81,00%
ω_6 – soybean notill	6	17	0	12	12	275	85,40%
Acurácia do Produtor	80,67%	85,33%	99,67%	86,33%	86,67%	91,67%	
Acurácia Média							88,39%

Tabela B96 – CAB-SVM com 140 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	242	3	0	10	9	7	89,30%
ω_2 – corn notill	7	251	0	9	7	0	91,61%
ω_3 – grass trees	1	0	299	3	1	1	98,03%
ω_4 – soybean clean	21	12	0	258	8	5	84,87%
ω_5 – soybean min	22	18	1	8	263	12	81,17%
ω_6 – soybean notill	7	16	0	12	12	275	85,40%
Acurácia do Produtor	80,67%	83,67%	99,67%	86,00%	87,67%	91,67%	
Acurácia Média							88,22%

Tabela B97 – CAB-SVM com 160 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	244	3	0	10	10	6	89,38%
ω_2 – corn notill	10	256	0	8	10	0	90,14%
ω_3 – grass trees	1	0	298	2	1	1	98,35%
ω_4 – soybean clean	16	11	1	261	7	6	86,42%
ω_5 – soybean min	22	16	1	11	261	15	80,06%
ω_6 – soybean notill	7	14	0	8	11	272	87,18%
Acurácia do Produtor	81,33%	85,33%	99,33%	87,00%	87,00%	90,67%	
Acurácia Média							88,44%

Tabela B98 – CAB-SVM com 180 bandas espectrais e kernel RBF γ 1.5.

Classes	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	Acurácia do Usuário
ω_1 – corn min	245	2	0	12	9	5	89,74%
ω_2 – corn notill	8	258	0	6	8	1	91,81%
ω_3 – grass trees	1	0	298	2	1	1	98,35%
ω_4 – soybean clean	21	11	1	264	6	7	85,16%
ω_5 – soybean min	20	17	1	11	264	14	80,73%
ω_6 – soybean notill	5	12	0	5	12	272	88,89%
Acurácia do Produtor	81,67%	86,00%	99,33%	88,00%	88,00%	90,67%	
Acurácia Média							88,94%