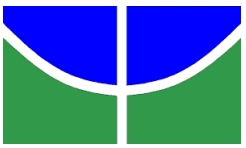


Universidade de Brasília

Departamento de Estatística

Classificação utilizando técnicas de aprendizado estatístico: *Estudo de casos*

Pedro Henrique Toledo de Oliveira Sousa

Monografia apresentada para a obtenção do título de Bacharel em Estatística.

Brasília

2016

Classificação utilizando técnicas de aprendizado estatístico: *Estudo de casos*

Pedro Henrique Toledo de Oliveira Sousa *

Matrícula:13/0016292

Orientador: Bernardo Borba de Andrade†

Monografia apresentada para a obtenção do título de Bacharel em Estatística.

Brasília

2016

*E-mail: pedrok_1@hotmail.com
†E-mail: bbandrade@unb.br

"Our quest for discovery fuels our creativity in all fields, not just science. If we reached the end of the line, the human spirit would shrivel and die."

(Stephen Hawking)

Resumo

Dois dos métodos mais modernos de classificação existentes são SVM (*Support Vector Machines*) e Boosting. O primeiro é um método que procura um hiperplano separador, seja no espaço natural dos dados, seja em um espaço de dimensão maior, a fim de tornar possível a tarefa de classificação. Já o Boosting é um método que aplica, sequencialmente, um determinado classificador em versões reponderadas do conjunto de dados de treinamento, dando maior peso às observações classificadas erroneamente na iteração anterior.

Neste trabalho estuda-se os métodos descritos acima realizando-se uma análise comparativa com os métodos LDA (*Linear Discriminant Analysis*), QDA(*Quadratic Discriminant Analysis*) e KNN(*k-Nearest Neighbors*). A análise comparativa será realizada em quatro bases de dados com diferentes características, verificando quais são as limitações de cada método e qual classificador melhor se ajusta aos diferentes tipos de bases de dados.

O estudo e implementação dos métodos contribuiu para a solidificação da ideia de que não existe um método melhor ou pior e sim o método mais adequado para determinado padrão de dados.

Sumário

Página

1 Introdução

Um procedimento de classificação é um método supervisionado, isto é, um método que fixa uma regra com base em uma variável que determina o grupo a que as observações pertencem. Essa regra deve ser usada para classificar novas observações. Os métodos de classificação mais conhecidos e mais antigos em estatística são regressão logística e análise discriminante. Estes métodos possuem baixo custo computacional e regras bastante interpretáveis, no entanto suas suposições paramétricas pode inviabilizar a aplicação destas técnicas.

Nos últimos anos, a disseminação do processo de tomada de decisões em diversas áreas, despertou um grande interesse pela obtenção de conhecimentos a partir de bases de dados. Com o advento dos computadores e a era da informatização, problemas estatísticos passaram a ser cada vez maiores e mais complexos. Grandes quantidades de dados são geradas em vários campos e o trabalho estatístico foca na extração de padrões e tendências importantes, o que é chamado de aprendizado dos dados (Hastie et al., 2013). Murphy (2012) define aprendizado de máquinas como um conjunto de métodos que podem detectar de forma automática certos padrões em um conjunto de dados de treinamento e, então, os utiliza para prever informações futuras.

Em 1970, Vapnik and Chervonenkis desenvolveram a teoria de aprendizado e, após um longo período marcado com falta de interesse dos pesquisadores pela área, em 1986 as redes neurais renasceram com a criação do algoritmo *backpropagation*. Desde então, a comunidade de aprendizado de máquinas cresceu e, na década de 90, ganhou espaço no meio científico com a criação das máquinas de suporte vetorial (SVM's) e com a introdução da família de algoritmos Boosting.

Desde o seu desenvolvimento, os algoritmos Boosting têm recebido grande atenção na comunidade estatística. Diferentemente dos métodos de classificação mais convencionais, os algoritmos Boosting são livres das suposições presentes nos mesmos e possuem um erro de classificação menor. Porém, exigem que os seus classificadores fracos sejam independentes e possuem uma interpretabilidade pequena. Além disso, são simples de serem implementados, possuem um baixo custo computacional

e podem ser implementados em diferentes cenários.

Posteriormente, Friedman, Hastie e Tibshirani (Friedman et al., 2000) mudaram a visão da comunidade estatística colocando o algoritmo Boosting para duas classes como uma aproximação de um modelo aditivo na escala logística, usando a máxima verossimilhança de Bernoulli como critério.

O presente trabalho visa comparar os seguintes métodos de classificação: classificadores lineares e quadráticos, método KNN, máquinas de suporte vetorial (SVM) e algoritmo *AdaBoost*. Para tanto, serão utilizadas quatro bases de dados, para as quais os algoritmos serão estudados. Uma das bases, que pode ser citada como exemplo, contém dados sobre a doença de Parkinson que são disponibilizados para a comunidade científica pelo grupo de aprendizagem de máquinas da *School of Information and Computer Science* da Universidade da Califórnia (Lichman, 2013).

1.1 Objetivos

O objetivo global deste trabalho é realizar uma análise da acurácia dos classificadores submetidos em bases de dados com diferentes características, verificando quais são os classificadores que melhor se ajustam aos dados e que tenham uma excelente capacidade preditiva para as variadas situações. Já o objetivo específico, envolve aprimorar as técnicas de classificação, encontrando os melhores parâmetros para os respectivos classificadores.

2 Referencial teórico

Os problemas de aprendizado podem ser separados em dois tipos: aprendizado supervisionado (classificação) e não supervisionado (agrupamento). No aprendizado supervisionado tem-se um número conhecido de classes e o objetivo é estabelecer uma regra de classificação, que também pode ser chamada de discriminante, formada com base nos dados de treinamento através da qual novas observações podem ser classificadas em uma determinada classes existente. Já no aprendizado não supervisionado, temos informações sobre os objetos e o intuito é estabelecer a existência de classes.

As bases de dados que serão utilizadas neste trabalho possuem variáveis respostas categóricas e a finalidade é prever os níveis desta variável resposta em novas observações com base em uma regra de classificação criada a partir de um conjunto de características (variáveis explicativas). Assim, o presente trabalho terá como foco a aplicação de técnicas de aprendizado supervisionado.

A seguir é apresentada uma definição formal de um classificador. Segundo Breiman (1984),

Definição 2.1. *Um classificador é uma função*

$$\begin{aligned} f : \Omega &\to \{1, 2, \ldots, K\} \\ x &\mapsto f(x) \in \{1, 2, \ldots, K\}. \end{aligned}$$

Agora, define-se $A_k = \{x \in \Omega; f(x) = k\}$, em que $k = 1, 2, \ldots, K$, ou seja, para cada k, A_k é uma partição de Ω na qual o classificador f prediz a classe k. Assim, um classificador f induz uma partição A_k tal que, para todo $x \in A_k$, a classe predita é k.

A função f pode ser estimada (treinamento do classificador) a partir de uma amostra de treinamento $L = \{\mathbf{x}_i, y_i\}_1^N$ em que $\mathbf{x}_i$ é um vetor com as p variáveis da i-ésima observação e y_i informa a classe da i-ésima observação.

Em uma estrutura de risco, o melhor classificador possível é aquele que minimiza a função de risco em questão. Para uma função de perda zero-um, na qual 0 indica uma classificação correta e 1 indica uma classificação errada, esse melhor classificador

possível é conhecido e é definido como a seguir.

Definição 2.2. *Sob uma função de perda zero-um, o classificador que minimiza o risco é o classificador:*

$$f(\boldsymbol{x}) = \arg\max_{y} \; P(\boldsymbol{x}|y)P(y).$$

Quando aplicado à verdadeira distribuição de (X, Y), esse classificador é chamado de *Classificador Ótimo de Bayes* ou *Regra de Bayes*. Na prática, as distribuições condicionais não são conhecidas, o que torna o uso deste classificador inviável. Apesar da complicação prática, a regra Bayes pode ser utilizada para a comparação de modelos por meio de dados simulados.

O método mais simples e eficaz para se classificar dados de forma linear é por meio da determinação de um hiperplano separador H_o, que pode ser representado por um vetor normal $\mathbf{v}_h$. Assim, um hiperplano pode ser definido por $H_o = \{\mathbf{x} \in \mathbb{R}^d \mid \langle \mathbf{x}, \mathbf{v}_h \rangle = 0\}$.

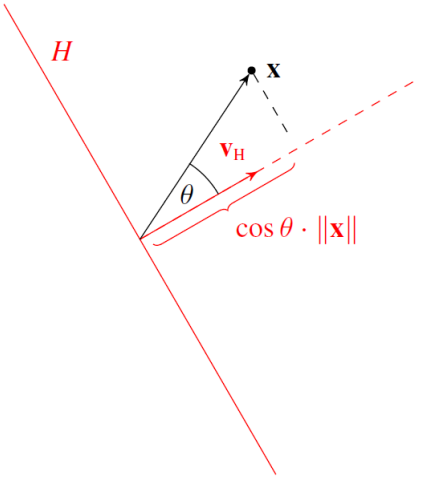


Figura 1: Reprodução de Orbanz (2016)

A Figura 1 mostra que a projeção de $\mathbf{x}$ na direção de $\mathbf{v}_h$ tem comprimento $\langle \mathbf{x}, \mathbf{v}_h \rangle$, medido em unidades de $\mathbf{v}_h$. Assim, $\frac{\langle \mathbf{x}, \mathbf{v}_h \rangle}{\|\mathbf{v}_h\|}$ possui comprimento em unidades das coordenadas.

A distância entre $\mathbf{x}$ e um hiperplano pode ser definida como:

$$d(\mathbf{x}, H_o) = \frac{\langle \mathbf{x}, \mathbf{v}_h \rangle}{\|\mathbf{v}_h\|} = \|\mathbf{x}\| \cos\theta. \tag{1}$$

Tendo em vista que o cosseno é maior que zero se $\theta \in (-\frac{\pi}{2}, \frac{\pi}{2})$, a decisão sobre qual lado do hiperplano a observação $\mathbf{x}$ se encontra pode ser dada por:

$$\text{sgn}(\cos\theta) = \text{sgn}(\langle \mathbf{x}, \mathbf{v}_h \rangle). \tag{2}$$

A definição dada de hiperplano se limita a hiperplanos que passam pela origem dos eixos de coordenadas. Para uma generalização, define-se um hiperplano qualquer H como $H = H_o + c \times \mathbf{v}_h$ em que c é um valor positivo utilizado para transladar um hiperplano H. Com a nova definição de um hiperplano, se $\mathbf{v}_h$ é um vetor unitário, a regra de decisão pode ser dada pela Definição 2.3.

Definição 2.3. *Um classificador linear é uma função da forma:*

$$\text{sgn}(\langle \boldsymbol{x} - c\ \boldsymbol{v}_h, \boldsymbol{v}_h \rangle) = \text{sgn}(\langle \boldsymbol{x}, \boldsymbol{v}_h \rangle - c),$$

em que $\boldsymbol{v}_h \in \mathbb{R}^d$ *e* $c \in \mathbb{R}_+$.

O que equivale a dizer, que para um y_i com dois níveis ($K = 2$), se o termo $(\langle \mathbf{x}, \mathbf{v}_h \rangle - c)$ for positivo, a observação x será designada ao grupo A_1, e se for negativo, será designada ao grupo A_2.

Em 1958 Rosenblatt (1958) propôs o Perceptron, classificador linear que chega a uma solução ótima iterativamente e que pode ser utilizado em qualquer conjunto de dados perfeitamente separáveis. O fato do risco empírico ter um aspecto discretizado dificulta a busca por uma solução que minimize este risco. Devido a isso, o algoritmo é baseado em uma função de custo que é utilizada para obter-se uma aproximação do risco empírico. Com a aproximação da função de custo perceptron e o seu aspecto contínuo, a busca pela melhor solução tornou-se possível com a utilização de métodos de otimização convencionais.

Estudos com conjuntos de dados perfeitamente separáveis tiveram continuidade até que, com a implementação da ideia de variáveis de folga, métodos mais sofisti-

cados como Máquinas de Suporte Vetorial (SVM's) fossem desenvolvidos. Com a criação dos SVM's e demais métodos mais sofisticados, a classificação de dados que não são perfeitamente separáveis tornou-se possível, o que fez dos novos métodos alternativas a serem utilizadas no lugar dos métodos de análise discriminante e do método KNN.

2.1 Análise discriminante

Na análise discriminante é necessário o conhecimento a *priori* das características dos elementos da amostra. Este conhecimento é utilizado para classificar novos elementos nos grupos já existentes. Como é possível que um grupo tenha uma maior probabilidade de ocorrência que outro devido à diferença do número de observações em cada um, a regra de classificação leva em conta estas probabilidades a *priori* de ocorrência, que podem ser estimadas através da proporção das observações em cada classe. A Equação (3) apresenta a regra de classificação, que consiste basicamente em uma comparação da razão entre as verossimilhanças com a razão entre as probabilidades a *priori*, como a seguir:

$$P(1|\mathbf{x}) > P(2|\mathbf{x}) \qquad \Rightarrow \quad \frac{p(\mathbf{x}|1)}{p(\mathbf{x}|2)} > \frac{P(2)}{P(1)}, \tag{3}$$

em que *1* e *2* são duas classes em questão.

A definição dada em análise discriminante é equivalente à Definição 2.2. Sendo assim, a necessidade de que os dados se ajustem a um modelo de distribuição teórico (no caso da análise discriminante o modelo é a distribuição Normal) é de suma importância para o bom funcionamento do método.

Em análise discriminante, destaca-se dois métodos: Análise Discriminante Linear (LDA) e Análise Discriminante Quadrática (QDA). Ambas partem do princípio de que o conjunto de dados segue uma distribuição normal dentro de cada classes especificada, sendo que na LDA pressupõe-se que as matrizes de covariâncias de cada grupo sejam iguais e já na QDA essas matrizes podem ser diferentes.

Se o número de classes é maior que dois, então uma extensão natural do discriminante linear de Fisher existe por meio da analise discriminante múltipla (Johnson

and Wichern, 2007). Para efeitos de esclarecimento do método, serão apresentados os métodos LDA e QDA para duas classes, sendo semelhante a aplicação de tais métodos em casos com mais de duas classes.

2.1.1 LDA

A ideia básica do LDA envolve a obtenção de uma transformação linear que discrimine as classes da melhor forma possível e a classificação é submetida em um espaço transformado baseado em alguma métrica, como por exemplo a distância Euclidiana.

Suponha que as densidades conjuntas de $\mathbf{x}$ para as K classes sejam dadas por:

$$\mathbf{x}|k \ \sim \ N(\mu_k, \Sigma).$$

Seja

$$D(\mathbf{x}) = \log\left(\frac{p(\mathbf{x}|1)\pi_1}{p(\mathbf{x}|2)\pi_2}\right) \tag{4}$$

$$= \left[-\frac{1}{2}\left(\mu_1^{'}\Sigma^{-1}\mu_1 - \mu_2^{'}\Sigma^{-1}\mu_2\right) + \log\left(\frac{\pi_1}{\pi_2}\right)\right] + \left[\Sigma^{-1}(\mu_1 - \mu_2)\right]^{'}\mathbf{x} \tag{5}$$

$$= \mathbf{a} + \mathbf{b}^{'}\mathbf{x}\ , \tag{6}$$

sendo π_k a estimativa da probabilidade a priori da k-ésima classe. Classificamos $\mathbf{x}$ em *1* se $D(\mathbf{x}) > 0$.

2.1.2 QDA

Suponha que as densidades conjuntas de $\mathbf{x}$ para as k classes sejam dadas por:

$$\mathbf{x}|k \ \sim \ N(\mu_k, \Sigma_k).$$

Seja

$$D(\mathbf{x}) = \log\left(\frac{p(\mathbf{x}|1)\pi_1}{p(\mathbf{x}|2)\pi_2}\right) \tag{7}$$

$$= \left\{ -\frac{1}{2} \left[\log \left(\frac{\det \Sigma_1}{\det \Sigma_2} \right) + \mu_1^{'} \Sigma_1^{-1} \mu_1 - \mu_2^{'} \Sigma_2^{-1} \mu_2 \right] + \log \left(\frac{\pi_1}{\pi_2} \right) \right\} + \left(\Sigma_1^{-1} \mu_1 - \Sigma_2^{-1} \mu_2 \right)^{'} \mathbf{x} + \mathbf{x}^{'} \left[-\frac{1}{2} \left(\Sigma_1^{-1} - \Sigma_2^{-1} \right) \right] \mathbf{x} \tag{8}$$

$$= \mathbf{a} + \mathbf{b}^{'} \mathbf{x} + \mathbf{x}^{'} V \mathbf{x} \, . \tag{9}$$

Classifica-se $\mathbf{x}$ em *1* se $D(\mathbf{x}) > 0$.

2.2 Método KNN

O método *k-Nearest Neighbors* (KNN) é extremamente simples e possui um desempenho razoavelmente bom quando se tem muitas observações. Para se classificar um novo ponto do conjunto de teste basta selecionar os K pontos do conjunto de treinamento mais próximos. A classe cujo ponto de teste será designado é definida pela classe predominante entre as K observações mais próximas. Em caso de empate, a escolha pode ser completamente aleatória.

Procedimento:

1. Calcula-se a distância entre todas as observações do conjunto de treinamento. Pode-se usar a distância euclidiana ou outra.

2. Os K vizinhos mais próximos de cada observação são escolhidos para "votar".

3. A observação é atribuída à classe que recebe mais votos.

4. O processo é repetido para diferentes valores de K. O número de vizinhos selecionado no modelo final é escolhido com base no número de erros de classificação.

Assim, é possível submeter o procedimento para diversos $K's$ e escolher aquele que retorna um menor erro de classificação. Vale ressaltar que existe uma preferência por valores ímpares de K a fim se evitar empates.

2.3 Máquinas de suporte vetorial - SVM's

Máquinas de suporte vetorial (SVM's) possuem grande aplicabilidade em dados de duas classes. A ideia básica é procurar pelo hiperplano separador ótimo entre

duas classes através da maximização das margens, criadas pela distância entre o hiperplano e os vetores de suporte das classes. A Figura 2 apresenta um conjunto de dados separado pelo hiperplano separador ótimo baseado nos vetores de suporte, que se encontram sobre as margens.

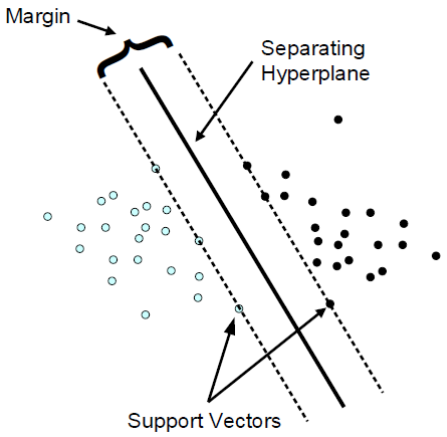


Figura 2: Reprodução de Meyer (2015)

Se A_i é um conjunto de pontos da i-ésima classe, o menor conjunto convexo contendo todos os pontos em A_i é chamado de *convex hull* de A_i, denotado por $\text{conv}(A_i)$. Todo ponto $\mathbf{x}$ em um conjunto convexo pode ser representado por uma combinação convexa dos pontos que delimitam a fronteira desse conjunto, os chamados pontos extremos de um conjunto convexo. Levando em consideração tal definição, os vetores de suporte podem ser interpretados como os pontos extremos, dos *convex hulls*, que se encontram o mais próximo possível do hiperplano separador.

Como este critério de margem máxima foca toda a atenção na região próxima do hiperplano, pequenas mudanças nos vetores de suporte podem resultar em alterações significantes no classificador.

A minimização da norma de $\mathbf{v}_h$ implica na maximização da margem. Dessa forma:

Definição 2.4. *A solução de um problema de otimização SVM é dada por:*

$$\begin{aligned} &\min_{v_h,c} \|\boldsymbol{v}_h\| \\ \text{s. t} \qquad &\tilde{y}_i(\langle \boldsymbol{v}_h, \tilde{\boldsymbol{x}}_i \rangle - c) \geq 1 \quad , \text{ para } \; i = 1, 2, \ldots, n. \end{aligned}$$

Como uma alternativa para a solução com restrição acima, tem-se o problema

dual da Definição 2.5 que envolve mais facilidade de cálculo.

Definição 2.5. *Problema dual:*

$$\min_{\boldsymbol{s} \in conv(A_1)\ \&\ \boldsymbol{r} \in conv(A_2)} \quad \|\boldsymbol{s} - \boldsymbol{r}\|^2$$

$$\boldsymbol{s} = \sum_{\tilde{y}_i \in A1} \alpha_i \tilde{\boldsymbol{x}}_i \qquad \boldsymbol{r} = \sum_{\tilde{y}_i \in A2} \alpha_i \tilde{\boldsymbol{x}}_i$$

$$\text{s. t} \quad \sum_{\tilde{y}_i \in A1} \alpha_i = 1 \qquad \sum_{\tilde{y}_i \in A2} \alpha_i = 1$$

$$\alpha_i \geq 0$$

Se $\tilde{\mathbf{x}}_i$ não é um ponto extremo do *convex hull*, define-se $\alpha_i = 0$. Assim, os vetores de suporte dos dois *convex hulls* ($\text{conv}(A_1)$ e $\text{conv}(A_2)$) podem ser encontrados resolvendo-se o problema quadrático da Definição 2.5. Em outras palavras procura-se alguns dos pontos extremos dos respectivos *convex hulls* cujas distâncias entre eles é a menor existente. Para isso, como $\mathbf{s}$ e $\mathbf{r}$ são representados por suas coordenadas baricêntricas (α_i), basta encontrar $\alpha_1, \alpha_2, \ldots, \alpha_n$ que minimizem o problema quadrático.

Se, para o problema quadrático da Definição 2.5, adicionarmos $\tilde{y}_i$ em $\mathbf{s}$ e $\mathbf{r}$, a junção dos somatórios das classes torna-se possível. Com apenas um somatório, tem-se que $\|\mathbf{s}-\mathbf{r}\|^2 = \sum_{i,j} \tilde{y}_i\tilde{y}_j\alpha_i\alpha_j\langle\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_j\rangle$. Minimizar $\sum_{i,j} \tilde{y}_i\tilde{y}_j\alpha_i\alpha_j\langle\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_j\rangle$ é o mesmo que maximizar $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \tilde{y}_i\tilde{y}_j\alpha_i\alpha_j\langle\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_j\rangle$. Assim, o problema dual da Definição 2.5 pode ser reescrito como na Definição 2.6.

Definição 2.6.

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad \boldsymbol{W}(\boldsymbol{\alpha}) := \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \tilde{y}_i\tilde{y}_j\alpha_i\alpha_j\langle\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j\rangle \ ,$$

$$\text{s. t} \quad \sum_i^n \tilde{y}_i\alpha_i = 0 \ ,$$

$$\alpha_i \geq 0 \ .$$

Com a solução do problema dual, calcula-se o vetor v_h do hiperplano ótimo por $v_h^* := r^* - s^* = \sum_{i=1}^{n} \tilde{y}_i \alpha_i^* \tilde{\mathbf{x}}_i$, em que r^* e s^* são funções de α^*, que é o argumento que minimiza a expressão da Definição 2.5. Tendo v_h, c é calculado da seguinte forma:

$$c^* = \frac{\max_{\tilde{y}_i \in A1} \langle \mathbf{v}_h^*, \tilde{\mathbf{x}}_i \rangle + \min_{\tilde{y}_i \in A2} \langle \mathbf{v}_h^*, \tilde{\mathbf{x}}_i \rangle}{2}. \tag{10}$$

Assim, a função de classificação pode ser expressa em termos de α_i:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} \tilde{y}_i \alpha_i^* \langle \mathbf{x}, \tilde{\mathbf{x}}_i \rangle - c^* \right). \tag{11}$$

As técnicas de SVM's apresentadas até agora não permitem que existam pontos além da margem ou até mesmo pontos do lado errado do hiperplano. Motivados pela resolução deste problema, pesquisadores introduziram o conceito de *soft-margin classifiers*. A ideia geral é utilizar uma variável de folga que permite que os dados de treinamento atravessem a margem ao mesmo tempo que aplica um custo à violação da regra.

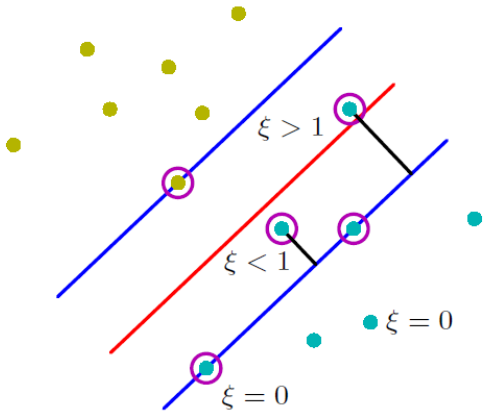


Figura 3: Reprodução de Orbanz (2016)

Com a implementação desta variável de folga, substitui-se a restrição $\widetilde{y}_i(\langle \mathbf{v}_h, \widetilde{\mathbf{x}}_i \rangle - c) \geq 1$ por $\widetilde{y}_i(\langle \mathbf{v}_h, \widetilde{\mathbf{x}}_i \rangle - c) \geq 1 - \xi_i$, com $\xi_i \geq 0$.

Com a introdução da variável de folga, o problema de otimização passa a ser descrito como:

Definição 2.7.

$$\begin{aligned} &\min_{v_h,c,\xi} \quad \|\boldsymbol{v}_h\| + \lambda \sum \xi_i^2 \\ \text{s. t} \qquad &\widetilde{y}_i(\langle \boldsymbol{v}_h, \widetilde{\boldsymbol{x}}_i \rangle - c) \geq 1 - \xi_i \qquad \text{para} \;\; i = 1, 2, \ldots, n \;\; e \;\; \xi_i \geq 0 \; . \end{aligned}$$

Segundo a Definição 2.7, o algoritmo de treinamento tem agora um parâmetro $\lambda > 0$ sobre o qual deve-se realizar uma validação cruzada a fim de se encontrar um valor adequado para λ. Assim como para o problema sem variável de folga, o problema da Definição 2.7 também possui uma solução dual sobre a qual os cálculos podem ser mais facilmente desenvolvidos. Essa solução dual é descrita como:

Definição 2.8.

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad \boldsymbol{W}(\boldsymbol{\alpha}) := \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \tilde{y}_i \tilde{y}_j \alpha_i \alpha_j (\langle \tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j \rangle + \frac{1}{\lambda} I\{i = j\}) \; ,$$

$$\text{s. t} \quad \sum_i^n \tilde{y}_i \alpha_i = 0 \; ,$$

$$\alpha_i \geq 0 \; .$$

Após a otimização do problema dual com as variáveis de folga, a regra de classificação é a mesma que para SVM sem essas variáveis.

2.3.1 Kernels

A teoria de SVM apresentada até então utiliza o produto escalar $\langle \mathbf{x}, \tilde{\mathbf{x}}_i \rangle$ como uma medida de similaridade entre $\mathbf{x}$ e $\tilde{\mathbf{x}}_i$. Como o produto escalar é linear, o SVM também é linear. Se usarmos um termo não linear no lugar do produto escalar, teremos um classificador não linear. Em outras palavras, quando não se pode separar duas classes na dimensão em questão usando um hiperplano, basta mapear as observações em um espaço de dimensão maior no qual é possível separar as classes por um hiperplano. Para isso utiliza-se a teoria de kernels. A seguir tem-se a definição formal de um kernel.

Definição 2.9. *Uma função* $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *é chamada de um kernel em* $\mathbb{R}^d$ *se existe alguma função* $\phi : \mathbb{R}^d \to \vartheta$ *em algum espaço* ϑ *no qual o produto escalar é tal que*

$$k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \langle \phi(\boldsymbol{x}), \phi(\tilde{\boldsymbol{x}}) \rangle_{\vartheta} \qquad \forall \ \ \boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^d$$

A Definição 2.9 mostra que k é um kernel se puder ser interpretado como um produto escalar em um espaço ϑ, ou seja, ϕ transforma os dados de forma que um classificador linear SVM funcione bem. A Figura 4 apresenta uma ideia geométrica da aplicação de um kernel, considerando o kernel presente na Equação (12).

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3 \quad \text{em que} \quad \phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} := \begin{pmatrix} x_1^2 \\ 2x_1x_2 \\ x_2^2 \end{pmatrix} \tag{12}$$

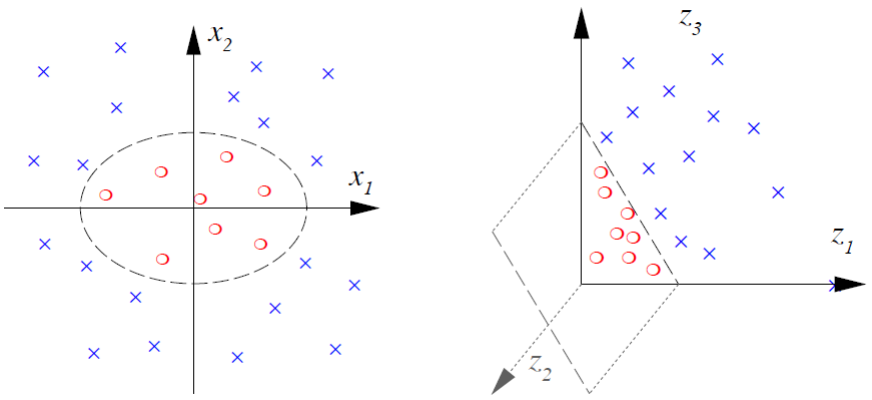


Figura 4: Reprodução de Orbanz (2016)

A seguir, tem-se alguns dos kernels mais usuais no meio científico. Vale ressaltar que para o caso do SVM linear, usa-se o kernel linear, que é simplesmente o produto interno.

- Os algoritmos que utilizam um kernel linear são equivalentes a um procedimento sem a utilização de kernels. O kernel linear é a função kernel mais simples possível. Ele é dado pelo produto interno $\langle x, x' \rangle$ acrescido de uma

constante c, como a seguir:

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \mathbf{x}^{'}\tilde{\mathbf{x}} + c \tag{13}$$

- O kernel polinomial, como o próprio nome sugere, é bastante utilizado em dados cuja regra de decisão possui um padrão polinomial. Esse kernel é descrito por:

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = (\alpha \mathbf{x}^{'}\tilde{\mathbf{x}} + c)^{d} \ , \tag{14}$$

  no qual α e c são parâmetros de ajuste e d representa o grau do polinômio.

- O kernel gaussiano é bastante utilizado em bases cuja disposição dos dados sugere uma separação por hiperesferas. Em muitos casos, mesmo sendo difícil verificar essa disposição devido às altas dimensões, o kernel gaussiano possui um desempenho satisfatório. Esse kernel é descrito por:

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp\left(-\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}{2\sigma^2}\right) \ . \tag{15}$$

Assim, com a utilização desta nova ferramente, torna-se possível a aplicação dos SVM's em dados que não possuem uma fronteira de decisão linear.

Matematicamente a implementação de um kernel em um SVM pode ser representada conforme a Definição 2.10.

Definição 2.10. *O problema de otimização em dados com fronteiras de decisão não lineares é definido por:*

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad \boldsymbol{W}(\boldsymbol{\alpha}) := \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \tilde{y}_i \tilde{y}_j \alpha_i \alpha_j (k(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j) + \frac{1}{\lambda} I\{i = j\}) \ ,$$

$$\text{s. t} \quad \sum_i^n \tilde{y}_i \alpha_i = 0 \ ,$$

$$\alpha_i \geq 0 \ .$$

Consequentemente, a função de classificação passa a ter o kernel em sua estrutura, podendo ser formulada como:

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{n} \tilde{y}_i \alpha_i^* k(\mathbf{x}, \tilde{\mathbf{x}}_i) - c^*\right). \tag{16}$$

2.3.2 SVM aplicado a dados com múltiplas classes

A teoria de SVM foi construída sob a ideia de se classificar conjuntos de dados em problemas com variável resposta binária. No entanto é possível implementar tal técnica em dados com variável resposta com mais de dois níveis, desde que se realize alguns procedimentos.

Um dos procedimentos mais utilizados para que se torne possível a utilização das técnicas de SVM em dados com múltiplas classes é o chamado *one-against-one*, no qual um classificador é treinado para cada par de variáveis, o que corresponde a um total de $\binom{K}{2} = \frac{K(K-1)}{2}$ classificadores. As observações são classificadas por meio de uma regra combinada proveniente dos resultados dos classificadores treinados em cada par. A regra combinada mais conhecida é o método de votos, por meio da qual as observações são designadas às classes segundo o maior número de votos, que representam as decisões encontradas nos classificadores separadamente.

Apesar de generalizar o classificador SVM para múltiplas classes, essa técnica possui algumas inconveniências. Na Figura 5 um exemplo de dados contendo três classes é apresentado.

Quando a estratégia de votos combinados é utilizada, existem áreas com inconsistências. Na Figura 5 a área de inconsistência é indicada pelo sinal de interrogação. No exemplo em questão, é comum que, na região de inconsistência especificada, todos os classificadores discordem, o que ocasiona na rejeição destas observações. A maneira mais simples de se classificar tais observações "problema"é designá-las à classe com maior probabilidade a priori P_{k^*} (Tax and Duin, 2002).

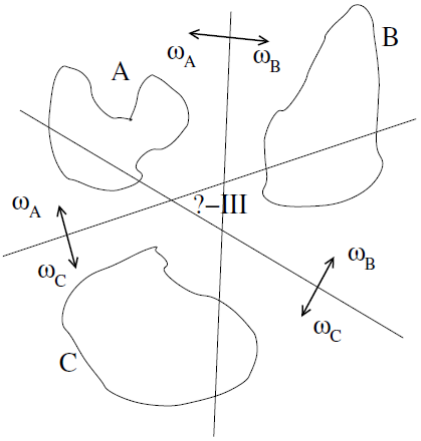


Figura 5: Exemplo de implementação do método *one-against-one*(Tax and Duin, 2002)

2.4 Boosting

Um algoritmo Boosting utiliza diferentes algoritmos de aprendizado como elementos integrantes. O objetivo do algoritmo é gerar de forma eficiente hipóteses de grande acurácia usando algoritmos de aprendizado chamados de classificadores fracos (*weak learners*), que nada mais são que métodos de classificação ligeiramente melhores que uma escolha aleatória. Em outras palavras, isso equivale a perguntar se, dado um método que gere classificadores fracos, é possível se obter um método próximo do ótimo a partir de uma combinação dos resultados provenientes destes classificadores.

Desde que o conceito de um algoritmo Boosting foi introduzido na comunidade de aprendizado de máquinas, surgiram vários algoritmos do tipo Boosting. Neste trabalho foram apresentadas descrições apenas dos algoritmos Boosting que geram hipóteses determinísticas. Para a aplicação do método nas bases de dados a serem utilizadas, optou-se por aplicar o algoritmo *AdaBoost* (Adaptative Boosting), que é um dos algoritmos da família Boosting mais utilizados. Esse algoritmo é iterativo e gera, de forma determinística e adaptativa, uma distribuição sobre as observações da amostra, dando maior peso às observações que foram classificadas erroneamente no passo anterior. Inicialmente, todos os pesos são iguais e, em cada iteração, penaliza-se cada vez mais as observações classificadas erroneamente na iteração passada, fazendo com que o classificador fraco seja forçado a dar mais atenção às observações

"problema"do conjunto de treinamento.

A tarefa dos classificadores fracos é encontrar uma hipótese fraca (*weak hypothesis*), $h_t : X \to \{A_1, A_2\}$ apropriada para o conjunto de dados que está submetido aos pesos. O algoritmo *AdaBoost* discreto pode ser descrito no quadro a seguir:

1. Inicializar os pesos das observações $w_i = \frac{1}{n}$ para $i = 1, 2, \ldots, n$.
2. Fazer m variando de 1 a M.

   2.1 Ajustar um classificador $g_m(x)$ para os dados de treinamento usando os pesos w_i.

   2.2 Computar

$$err_m := \frac{\sum_{i=1}^{n} w_i I\{y_i \neq g_m(x_i)\}}{\sum_i w_i}.$$

   2.3 Computar $\alpha_m = \log\left(\frac{1-err_m}{err_m}\right)$.

   2.4 Definir $w_i \leftarrow w_i \times \exp(\alpha_m I\{y_i \neq g_m(x_i)\})$ para $i = 1, 2, \ldots, n$.

3. Fazer

$$f(x) := \operatorname{sgn}\left(\sum_{m=1}^{M} \alpha_m g_m(x_i)\right).$$

Assim, quando a classificação é correta, w_i não se altera.

Sob certas condições de regularidade, o algoritmo *AdaBoost* é considerado consistente, no sentido de que, durante o treinamento, gera uma sequência de classificadores com erro que converge para o erro do classificador de Bayes. Outro fato interessante é que na maioria dos algoritmos, como por exemplo o perceptron, o processo termina quando o erro de treinamento chega em seu mínimo. No entanto, para o algoritmo *AdaBoost*, os pesos continuam mudando mesmo se o erro de treinamento chegar em seu mínimo. Assim, o algoritmo Boosting possui excelentes propriedades de generalização uma vez que, apesar de não melhorar o erro de treinamento a partir de certo ponto, continua reduzindo o erro de teste.

2.4.1 Boosting como um processo de minimização *stagewise*

O classificador Boosting é da forma $\text{sgn}(F(x))$. Uma combinação linear das funções $g_1, \ldots, g_m$ pode ser interpretada como uma representação de F usando funções base. Pode-se então interpretar a combinação linear $F(x)$ como uma aproximação da região de decisão utilizando uma base de classificadores fracos. Dentro do contexto de minimização *stagewise*, o algoritmo *AdaBoost* é equivalente a um modelo aditivo *stagewise* utilizando uma função de perda exponencial.

A função de risco de um classificador aditivo sob perda exponencial pode ser dada como segue abaixo:

$$\hat{R}_n(F^{(m-1)} + \beta_m g_m) = \frac{1}{n} \sum_{i=1}^{n} \exp(-y_i F^{m-1} - y_i \beta_m g_m(x_i)) \tag{17}$$

$$\hat{R}_n(F^{(m-1)} + \beta_m g_m) = \frac{1}{n} \sum_{i=1}^{n} w_i^m \ \exp(-y_i \beta_m g_m(x_i)) \tag{18}$$

$$\text{com notação} : F^{(m-1)} := \sum_{j \leq m} \beta_m g_m$$

Pode ser mostrado que o classificador obtido, em cada iteração m, resolvendo-se:

$$\underset{\beta_m, g_m}{\arg\min} \ \ \hat{R}_n(F^{(m-1)} + \beta_m g_m), \tag{19}$$

gera o classificador *AdaBoost*. Mais precisamente, se é construído um classificador $F(x) := \sum_{m=1}^{M} \beta_m g_m$ tal que seja a solução da Equação (19), em cada iteração m, tem-se que definir:

- g_m como um classificador que minimize o erro de classificação sob as observações com pesos.
- $\beta_m = \frac{1}{2} \log\left(\frac{1-err_m}{err_m}\right) = \frac{1}{2}\alpha_m$
- $w_i^{m+1} := w_i^m \times \exp(-y_i \beta_m g_m(x_i))$

Isso é precisamente equivalente ao que o *AdaBoost* faz.

2.4.2 Boosting aplicado a dados com múltiplas classes

Até então, a metodologia Boosting aqui apresentada apenas considerou problemas de classificação nos quais o objetivo é distinguir as duas únicas classes existentes. Como muitos dos problemas de aprendizagem encontrados no mundo real possuem variável resposta com mais de dois níveis, problemas de múltiplas classes, pesquisadores de área se dedicaram em desenvolver algoritmos que acabam por ser uma extensão do algoritmo *AdaBoost* já apresentado.

A generalização mais simples e direta é chamada de procedimento *AdaBoost.M1*. A principal exigência do *AdaBoost.M1* é que cada hipótese formulada pelos classificadores fracos tenha um erro de predição menor que 50% com respeito à distribuição na qual a hipótese foi formulada. Com essa condição satisfeita, provou-se que o erro combinado das hipóteses decresce exponencialmente tal como no *AdaBoost* para problemas binários.

É importante ressaltar que essa condição é uma exigência mais forte que a do algoritmo para problemas com duas classes, uma vez que no primeiro bastava que cada classificador fraco obtivesse uma performance sutilmente melhor que uma escolha totalmente aleatória. No problema binário ($k = 2$), uma escolha aleatória seria correta com probabilidade de 50%, mas quando $k > 2$, a probabilidade de uma escolha aleatória ser correta é apenas $\frac{1}{k} < \frac{1}{2}$. Portanto, a exigência de que a hipótese fraca seja melhor do que 50% é significantemente mais forte que uma simples condição de que a hipótese fraca tenha uma performance sutilmente melhor que uma escolha aleatória.

3 Metodologia

Esta seção traz uma breve descrição das bases de dados utilizadas neste trabalho, além de uma prévia sobre os métodos de avaliação de desempenho dos classificadores em questão. Foram selecionadas quatro diferentes bases de dados cada uma com suas características específicas.

- `Hand-Written Digits`: Base que possui 200 observações e 256 variáveis. Cada observação representa uma imagem 16×16 de um número escrito a mão. Existe um arquivo separado com os respectivos rótulos das classes (o arquivo contém duas classes - dígitos 5 e 6 de tal forma que os rótulos são -1 e 1 respectivamente). A Figura 6 possui dimensão 16×16 pixeis e corresponde à primeira linha do conjunto de dados, que a representa por números provenientes de uma escala de cinzas, neste caso cada variável representa um valor na escala de cinzas para o respectivo pixel.

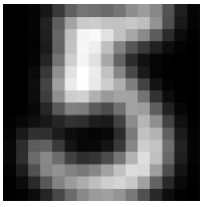


Figura 6: Uma das 200 observações da base `Hand-Written Digits`.

- `Wine`: Dados resultantes de uma análise química de vinhos desenvolvidos na mesma região, mas derivados de três diferentes variedades. A base de dados possui 177 observações e 14 variáveis correspondentes aos componentes presentes nos três tipos de vinho, sendo uma delas correspondente às classes que indicam o tipo de vinho.

- `Sonar`: Conjunto de dados com 208 observações e 61 variáveis, sendo que a última corresponde às classes (0 se o objeto é rocha e 1 se é uma mina).

- `Parkinsons Disease`: Conjunto de dados que foi formado a partir de um estudo em 31 pessoas sendo que 23 delas possuíam a doença de Parkinson. Com exceção da coluna correspondente às classes (0 para pessoas saudáveis e

1 para pessoas com a doença), cada coluna, dentre as 24 existentes, da matriz de dados é uma medida de voz particular e cada linha corresponde a uma das 195 gravações destes indivíduos.

Dentre as quatro bases, a base `hand-written digits` pode ser encontrada em Orbanz (2016), já as demais foram retiradas do repositório `UCI Machine Learning`.

Será utilizada uma divisão em cada base de dados de tal forma que se tenha um conjunto de treinamento e um conjunto de teste.

3.1 Avaliação do desempenho de um classificador

Dentre os vários critérios para se avaliar o desempenho de um classificador, a taxa de erro aparente, ou taxa de erro de treinamento, é uma das medidas de avaliação mais simples existente e é obtida utilizando o próprio conjunto de treinamento. Este procedimento de estimação da taxa de erro de treinamento é consistente, mas viciado (Johnson and Wichern, 2007) e tende a subestimar a verdadeira taxa de erro, que é a probabilidade esperada de se classificar incorretamente um padrão selecionado aleatoriamente. Essa probabilidade é a taxa de erro de um conjunto de teste independente, infinitamente grande, retirado de uma população com a mesma distribuição do conjunto de treinamento. Em problemas de *overfitting* a taxa de erro de treinamento pode chegar a ser nula, mas a capacidade preditiva do método não necessariamente chega a ser a melhor possível. Tendo em vista as diversas complicações dessa métrica de performance, optou-se por utilizar outras duas métricas, a taxa de erro de teste (método *Holdout*), que verifica a capacidade preditiva do método com base em um conjunto de teste, e a taxa de erro proveniente do método de validação cruzada *k-fold*.

O método *Holdout* divide os dados em dois subconjuntos mutuamente excludentes, um que será utilizado para construir a regra de classificação, e o outro para se avaliar o desempenho da regra. A desvantagem deste método se encontra na inviabilidade de sua aplicação em amostras pequenas visto a redução do conjunto de treinamento com a retirada do conjunto de teste. Já métrica de performance *k-fold* é baseada na divisão aleatória do conjunto de dados em k subconjuntos mu-

tuamente exclusivos de tamanho aproximadamente iguais. Em cada passo, um dos k subconjuntos é utilizado como conjunto de teste e os outros *k-1* se unem compondo o conjunto de treinamento. Assim, o erro de validação cruzada *k-fold* nada mais é do que a média dos erros obtidos nos k ensaios.

Exceto para o procedimento *leave-one-out* (*n-fold-cross-validation*), que é considerado um procedimento completo de validação cruzada, este erro é um número aleatório que depende da divisão realizada nos subconjuntos. Devido a essa aleatoriedade, muitos pesquisadores optam por repetir a validação cruzada *k-fold* múltiplas vezes utilizando diferentes subconjuntos, obtendo assim uma estimativa de Monte-Carlo que chega a ser melhor do que a estimativa retornada pelo procedimento completo de validação cruzada (Kohavi, 1995). Uma vez que com uma única execução do procedimento *k-fold* já pode-se obter boas estimativas do erro de classificação, para a avaliação dos métodos aplicados às quatro bases de dados, optou-se por executar o método *k-fold*, com $k = 10$, uma única vez visto também a praticidade computacional.

Apesar do procedimento *k-fold* poder ser aplicado na base de dados completa, para a avaliação do desempenho dos métodos de classificação nas quatro bases de dados, optou-se por executar o *k-fold* apenas no conjunto de treinamento.

4 Resultados

As bases de dados foram importadas para o software R e posteriormente realizou-se a separação aleatória das bases originais em dois conjuntos, seguindo como regra a divisão dessas bases de tal forma que 20% das observações fossem destinadas ao conjunto de teste e os outros 80% ao conjunto de treinamento. Para todas as técnicas implementadas foram utilizados os mesmo conjuntos de teste e treinamento.

Para facilitar a apresentação dos resultados e das peculiaridades de cada algoritmo, as métricas de performance serão exibidas segundo os métodos utilizados. Sendo assim, primeiramente serão apresentadas todas as métricas de performance obtidas na análise discriminante e no KNN para as respectivas bases de dados, posteriormente apresentar-se-á as métricas obtidas com a implementação do SVM e, por fim, as métricas obtidas com o procedimento Boosting.

4.1 Resultados da análise discriminante e KNN

As técnicas de análise discriminante possuem algumas propriedades favoráveis: primeiro, é um método simples e pode ser facilmente implementado; segundo, é eficiente e na maioria dos casos possui um tempo de processamento relativamente pequeno.

Apesar dos aspectos positivos, as técnicas LDA e QDA não permitem a existência de multicolinearidade entre as variáveis, e devido a esta restrição, não pôde-se utilizar todas as variáveis da base de dados `H-W Digits` uma vez que esta possui a maioria de suas variáveis com correlações demasiadamente altas. Dentre as alternativas para contornar este problema, três foram colocadas em questão: redução da dimensão via componentes principais, seleção de variáveis via *stepwise* e retirada de uma variável dentre os pares com correlação acima de 90. A utilização das técnicas clássicas de componentes principais é uma alternativa simples e que torna possível o treinamento do algoritmo de classificação, no entanto torna-se inviável em bases que possuem mais variáveis do que observações. Já o procedimento *stepwise* exige a utilização de outros modelos de classificação (por exemplo o modelo logístico) para que a seleção das variáveis seja realizada, o que retiraria a independência das técnicas LDA e QDA

no procedimento de classificação. Sendo assim, optou-se por submeter os métodos LDA e QDA ao conjunto de dados utilizando a terceira alternativa para contornar o problema.

Como a principal característica da base `H-W Digits` é a existência de variáveis excessivamente correlacionadas, sendo que das 256 variáveis apenas uma tinha todas as suas correlações dois a dois abaixo de 80, a retirada de uma variável dentre os pares com correlação acima de 90 provocou uma redução na dimensão da base, que passou de 256 variáveis para 60 variáveis. Essa redução possibilitou a implementação do método LDA, mas não foi capaz de tornar possível a implementação do QDA uma vez que este método, mesmo com a retirada das variáveis, acusou problema na inversão da matriz de correlação de um dos grupos que representam os níveis da variável resposta.

A Tabela 1 apresenta os erros de classificação para as técnicas LDA e QDA segundo as métricas de performance em cada base de dados. Todas as métricas de performance foram implementadas no software R utilizando os resultados obtidos através da implementação das funções *lda* e *qda* presentes na biblioteca *MASS* do software R.

	`H-W Digits`		`Parkinson`		`Sonar`		`Wine`	
	LDA	QDA	LDA	QDA	LDA	QDA	LDA	QDA
Erro de treinamento	0.012	-	0.089	0.012	0.054	0	0	0.007
Erro de teste (*Holdout*)	0.050	-	0.076	0.051	0.238	0.380	0.027	0
Erro 10-fold	0.075	-	0.134	0.128	0.234	0.409	0.007	0.014

Tabela 1: Métricas de performance para os classificadroes da análise discriminante linear e quadrática.

Analisando a Tabela 1, é fácil verificar que de fato o erro de treinamento é predominantemente otimista possuindo erros menores. As outras duas métricas, que são mais usuais e confiáveis, retornaram resultados próximos, no entanto o *k-fold* tende a retornar erros levemente maiores que os erros provenientes da métrica *Holdout* (erros de teste).

As bases sobre as quais os algoritmos de classificação obtiveram melhor desempenho, independente da métrica de performance, foram a `H-W Digits` e a `Wine`, que possuía variável resposta com três níveis. Apesar das complicações e da enorme

perda de informação, pela retirada de variáveis devido à multicolinearidade existente entre as variáveis da primeira base em questão, o classificador LDA gerado conseguiu discriminar com alta acurácia os dois níveis da variável resposta.

A Tabela 2 apresenta os erros de classificação para a técnica não paramétrica denominada KNN segundo as métricas de performance em cada base de dados. Para a escolha do número de vizinhos para a classificação, calculou-se o erro de validação cruzada 10-fold para k variando de 1 a 20 e selecionou-se o k que retornava o menor erro 10-fold. Assim como na análise discriminante, todas as métricas de performance foram implementadas no software R e o algoritmo KNN foi executado pela função *knn* do pacote *class*.

	`H-W Digits` (k=5)	`Parkinson` (k=1)	`Sonar` (k=2)	`Wine` (k=14)
Erro de treinamento	0.025	0	0.066	0.042
Erro de teste (*Holdout*)	0.025	0	0.1428	0.027
Erro 10-fold	0.025	0.075	0.1503	0.034

Tabela 2: Métricas de performance para os classificadroes KNN.

A implementação do método knn mostrou que, apesar de ser um método extremamente simples, este foi capaz de classificar com acurácia as variáveis respostas de cada base de dados. Vale ressaltar que o método superou os métodos de análise discriminante em todas as bases cujas variáveis respostas são binárias, possuindo uma classificação com maior erro, relativamente às técnicas de análise discriminante, apenas na base `Wine`, que possui variável resposta com três níveis.

4.2 Resultados do SVM

Máquinas de suporte vetorial veem mostrando excelentes performances em tarefas de classificação binária. Os classificadores SVM possuem boa acurácia, são robustos e rápidos. No entanto, a teoria por trás dos uso de margens e hiperplanos separadores não pode ser tão facilmente estendida a problemas de classificação múltipla e apesar de métodos alternativos que utilizam SVM's, como o *one-against-one*, já é esperado que os classificadores SVM não tenham o mesmo desempenho em problemas de múltiplas classes do que em problemas binários.

Os classificadores para as respectivas bases de dados foram obtidos a partir da utilização de funções presentes no pacote *e1071* do software R. O algoritmo para classificação implementado no pacote é o chamado *C-classification* e possui uma formulação equivalente à formulação de classificadores SVM apresentada na Seção 2.3. Essa formulação é descrita da seguinte forma:

Definição 4.1. *O problema dual apresentado por Karatzoglou et al. (2006) e implementado no R é definido por:*

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad \boldsymbol{W}(\boldsymbol{\alpha}) := \sum_{i}^{m} \alpha_i - \frac{1}{2}\sum_{i,j} \tilde{y}_i\tilde{y}_j\alpha_i\alpha_j k(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j) \ ,$$

$$\text{s. t} \qquad \sum_{i}^{m} \tilde{y}_i\alpha_i = 0 \ ,$$

$$0 \leq \alpha_i \leq \frac{C}{m} \ .$$

A Definição 4.1 apresenta a formulação do problema dual com a parametrização utilizada pela função *tune*, que foi utilizada para a classificação SVM de todas as quatro bases de dados.

Karatzoglou et al. (2006) mostra que o problema da Definição 4.1 pode ser interpretado como um problema quadrático da seguinte forma:

$$\begin{aligned} \min \quad & c^T x + \tfrac{1}{2} x^T H x \ , \\ \text{s. t} \quad & b \leq Ax \leq b + r \ , \\ & l \leq x \leq u \ , \end{aligned} \tag{20}$$

em que $H \in \mathbb{R}^{m\times m}$ com entradas $H_{ij} = y_i y_j k(x_i, x_j)$, $c = (1, \ldots, 1) \in \mathbb{R}^m$, $u = (C, \ldots, C) \in \mathbb{R}^m$, $l = (0, \ldots, 0) \in \mathbb{R}^m$, $A = (y_1, \ldots, y_m) \in \mathbb{R}^m$, $b = 0$ e $r = 0$.

O parâmetro de custo C da formulação do SVM na Definição 4.1 controla a penalidade a ser atribuída aos dados de treinamento que atravessam a margem e consequentemente controla a complexidade da função de predição. Esse parâmetro é proporcional ao λ apresentado na Seção 2.3. Altos valores de C irão forçar o algoritmo a criar uma função de predição complexa e com poucos erros de classificação

no conjunto de treinamento, enquanto baixos valores do parâmetro de custo irão possibilitar funções de predição mais simples. Uma vez o parâmetro de custo possuindo um valor alto e consequentemente com poucas classificações erradas no conjunto de treinamento, não necessariamente fornecerá um classificador ótimo visto que existe a possibilidade de ocorrer *overfitting* no conjunto de treinamento. Diante de tais condições, faz-se necessário verificar qual valor do parâmetro C gera o classificador SVM mais conveniente.

Criou-se uma sequência de valores candidatos a parâmetro de custo que assumiam valores de 0.0001 a 0.0951 variando 0.005 de elemento a elemento. Assim, a sequência criada possuía 20 elementos e por meio da função *tune*, foram criados modelos SVM para cada parâmetro candidato sendo que o critério de seleção foi o erro de validação cruzada 10-fold, ou seja, o valor escolhido para C foi o que retornou o menor erro 10-fold.

A sequência foi definida supondo que o erro 10-fold fosse dependente do parâmetro de custo de tal forma que a função gerada a partir desta relação possuísse apenas um mínimo. Como, para todas as bases de dados, o valor selecionado como parâmetro de custo se encontrou no centro da sequência definida, considerou-se esta como um bom *grid* de busca não tendo a necessidade de verificar o ajuste para outros valores de C.

A Tabela 3 apresenta os erros de classificação dos SVM's lineares segundo cada métrica de performance, utilizando o melhor C encontrado em cada base de dados. Para a base de dados `Wine`, a função *tune* executa o método *one-against-one* abordado na Seção 2.3.2.

	`H-W Digits` (C=0.0051)	`Parkinson` (C=0.0601)	`Sonar` (C=0.0701)	`Wine` (C=0.0051)
Erro de treinamento	0.025	0.121	0.084	0.021
Erro de teste (*Holdout*)	0.025	0.051	0.285	0.027
Erro 10-fold	0.025	0.128	0.209	0.028

Tabela 3: Métricas de performance para os classificadores SVM linear.

Com exceção da base de dados `Sonar`, os erros de teste e validação cruzada 10-fold estão bem próximos ou até melhores do que os erros de treinamento, apontando que foi realizada uma boa escolha do parâmetro de custo e que o classificador gerado não

possui *overfitting*. Para a base de dados `Sonar`, o erro de treinamento se encontra bem mais otimista do que as demais métricas de performance, mas ao comparar tais resultados aos encontrados pelos métodos de análise discriminante, métodos que não exigem um parâmetro de entrada fornecido pelo programador, verifica-se uma estrutura semelhante o que aponta para um fenômeno inerente à base de dados e não a um *overfitting* de um classificador específico.

Comparando os resultados dos classificadores SVM com os resultados da análise discriminante e KNN, verificou-se que, em geral, a técnica SVM obteve um melhor desempenho que as técnicas LDA e QDA em bases cuja variável resposta é binária. No entanto, a técnica não paramétrica KNN se sobressaiu em relação ao SVM, com exceção da base de dados `Wine`, que possui múltiplas classes. Apesar do classificador SVM gerado superar o KNN na base com variável resposta com três níveis, o mesmo não conseguiu superar os métodos de análise discriminante nesta base.

Devido a sua grande aplicabilidade, juntamente com a implementação dos SVM's lineares, modelou-se classificadores SVM com a utilização do kernel gaussiano, especificado na Seção 2.3.1. Como o kernel gaussiano possui um parâmetro, para a implementação dos SVM's, utilizou-se a função *tune* visto que esta já seleciona conjuntamente os melhores parâmetros a serem utilizados. Em outras palavras, ao se definir uma sequência de valores para o parâmetro C e uma sequência de valores para o parâmetro σ, a função irá gerar classificadores SVM's com todas as combinações de parâmetros possíveis e posteriormente selecionará o melhor classificador, e consequentemente os melhores parâmetros, utilizando o erro de validação cruzada 10-fold como critério de seleção.

O kernel gaussiano utilizado no pacote *e1071* possui uma formulação equivalente ao kernel apresentado na Seção 2.3.1, sendo que a única diferença se encontra na parametrização, uma vez que, no pacote, o parâmetro utilizado é proporcional ao parâmetro σ, ou seja $\gamma = \frac{1}{2\sigma^2}$. Assim, definiu-se uma sequência para o parâmetro σ cujos cinco primeiros valores variavam de 1 a 5 variando uma unidade de elemento para elemento e os demais termos da sequência assumiam valores de 10 a 100 variando cinco unidades de elemento para elemento. Definida a sequência de candidatos para σ, realizou-se a transformação a fim de encontrar os valores correspondentes

para γ.

Para o parâmetro de custo C, diferentemente do caso linear, verificou-se que uma boa sequência de candidatos se daria pela utilização de elementos que assumissem valores de 0.1 a 2.1 variando 0.2 de elemento a elemento.

Após a implementação e definição dos melhores parâmetros para os respectivos modelos, verificou-se que, para todas as bases de dados, os melhores parâmetros pertenciam a valores contidos no centro das sequências, apontando não haver necessidade de testar outros valores para os parâmetros.

A Tabela 4 apresenta os erros de classificação dos SVM's gaussianos, segundo cada métrica de performance, utilizando os melhores parâmetros encontrados em cada base de dados.

	`H-W Digits` (C=0.9 ; γ=0.005)	`Parkinson` (C=1.1 ; γ=0.125)	`Sonar` (C=1.9 ; γ=0.031)	`Wine` (C=0.9 ; γ=0.125)
Erro de treinamento	0	0.025	0	0
Erro de teste (*Holdout*)	0.025	0.076	0.119	0.027
Erro 10-fold	0.018	0.107	0.127	0.007

Tabela 4: Métricas de performance para os classificadores SVM com kernel gaussiano.

Em todas as bases de dados, ao se considerar o erro 10-fold, que é uma métrica bem elaborada e com excelentes propriedades, como fator para comparação dos desempenhos, verificou-se que o classificador gerado através da técnica do SVM gaussiano obteve um desempenho melhor do que o SVM linear. Ademais, também levando em conta o erro 10-fold, verifica-se que o classificador SVM gaussiano possui o melhor desempenho, dentre todas as técnicas anteriores, para as bases cujas variáveis respostas são binárias e, para a base `Wine`, o classificador SVM gaussiano retorna o mesmo erro 10-fold que o classificador LDA. Assim, apesar das inconveniências em dados com múltiplas classes, ao se utilizar o kernel gaussiano no procedimento *one-agains-one*, a tarefa de classificação foi realizada com sucesso e foi possível obter um erro de classificação tão bom quanto os das técnicas que não possuem inconveniências em dados com respostas múltiplas.

Além do desempenho geral ser satisfatório, a técnica do SVM, com a utilização do kernel, conseguiu reduzir significativamente as taxas de erro de teste e 10-fold

da base `Sonar`, a qual possuía erros razoavelmente altos para os outros métodos até então explorados.

4.3 Resultados do Boosting

A performance relativa do método *AdaBoost* depende do problema em questão, da natureza da região de decisão, da complexidade do classificador base e do número de interações boosting. Assim, um fator importante para o retorno de baixas taxas de erro e o consequente sucesso da aplicação do algoritmo é a definição de um bom classificador base (*weak learner*).

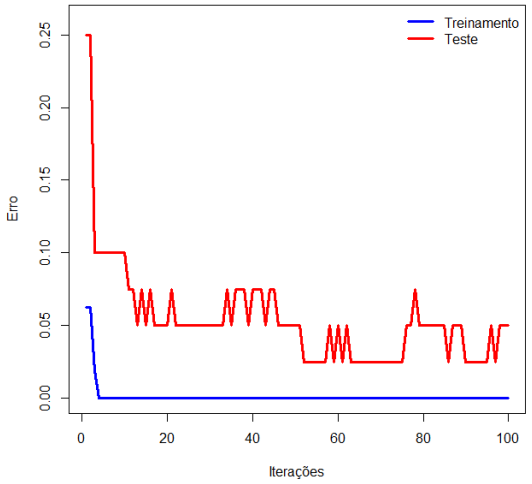
Árvores de classificação são baseadas na partição do espaço, ao qual os dados pertencem, em sub-regiões. Essa técnica vem sendo amplamente utilizada na comunidade estatística devido a sua interpretabilidade e a sua capacidade de selecionar automaticamente características relevantes. Sendo assim, definiu-se tal técnica como o classificador fraco a ser utilizado no algoritmo *AdaBoost*.

Os classificadores resultantes do procedimento *AdaBoost*, para todas as bases de dados, foram obtidos a partir da utilização de funções presentes no pacote *adabag* do software R. No caso da base de dados `Wine`, que possui três classes, a função implementada utilizou o método *AdaBoost.M1*, que foi apresentado na Seção 2.4.2.

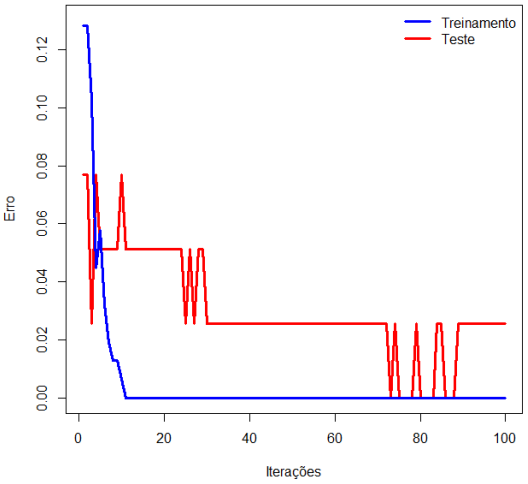
Como a complexidade do classificador fraco é relevante, o algoritmo *AdaBoost* foi submetido a arvores de classificação com diferentes números de nódulos. Variando a profundidade da arvore (número de nódulos) de 2 a 4, para cada base de dados, escolheu-se o melhor parâmetro para o classificador fraco utilizando o erro de validação cruzada 10-fold como critério.

A Tabela 5 apresenta os erros de classificação segundo cada métrica de performance, utilizando o melhor parâmetro encontrado para as arvores de classificação (*Node*) e realizando 100 iterações boosting. Já a Figura 7 apresenta o histórico dos erros de teste e treinamento em cada uma das 100 iterações boosting realizadas em cada base de dados.

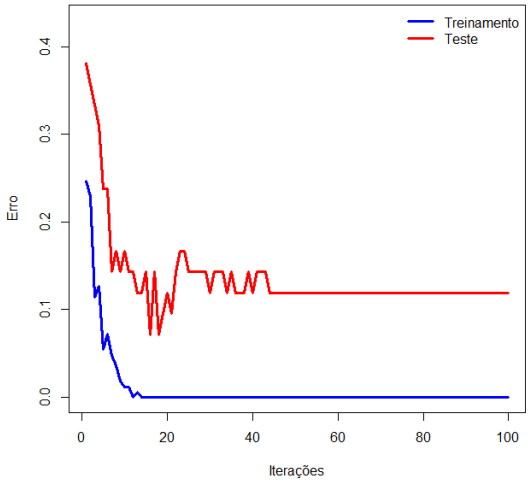
A análise gráfica confirma o pressuposto teórico de que mesmo quando o erro de treinamento chega a 0, se utilizadas mais iterações, o erro de teste pode reduzir. Os erros provenientes da classificação boosting realizada na base `Sonar` foram os que



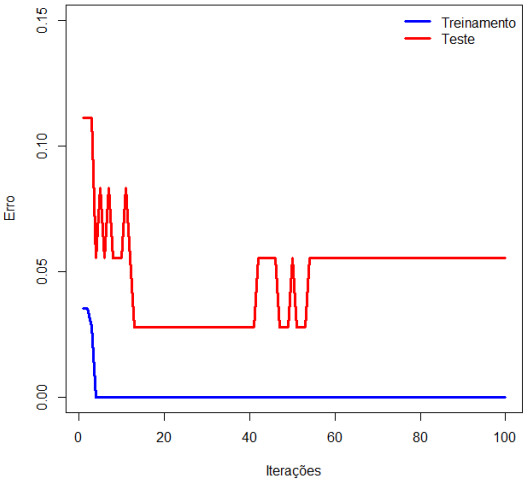
(a) `H-W Digits`



(b) `Parkinson`



(c) `Sonar`



(d) `Wine`

Figura 7: Erros de treinamento e teste segundo o número de iterações boosting em cada base de dados.

possuíram o comportamento mais próximo do esperado, gerando curvas que possuem um aspecto exponencial e um erro de teste que se estabiliza após 44 iterações. Para as demais bases de dados, apesar do erro de teste ter dificuldade em se estabilizar, as curvas sugerem esta tendência de decréscimo exponencial, mantendo o erro de teste com valores bem abaixo dos respectivos valores nas iterações iniciais.

Os saltos do erro de teste nas Figuras 7(b) e 7(d) ao decorrer das iterações

se justifica pelo fato dos classificadores fracos ja começarem classificando muito bem. Assim, como a escala no eixo Y é pequena uma vez que o primeiro erro já é razoavelmente próximo de zero, ao se classificar uma observação erroneamente a mais ou a menos relativamente à classificação da iteração anterior, a curva do erro de teste acaba por ressaltar pequenas variações na classificação.

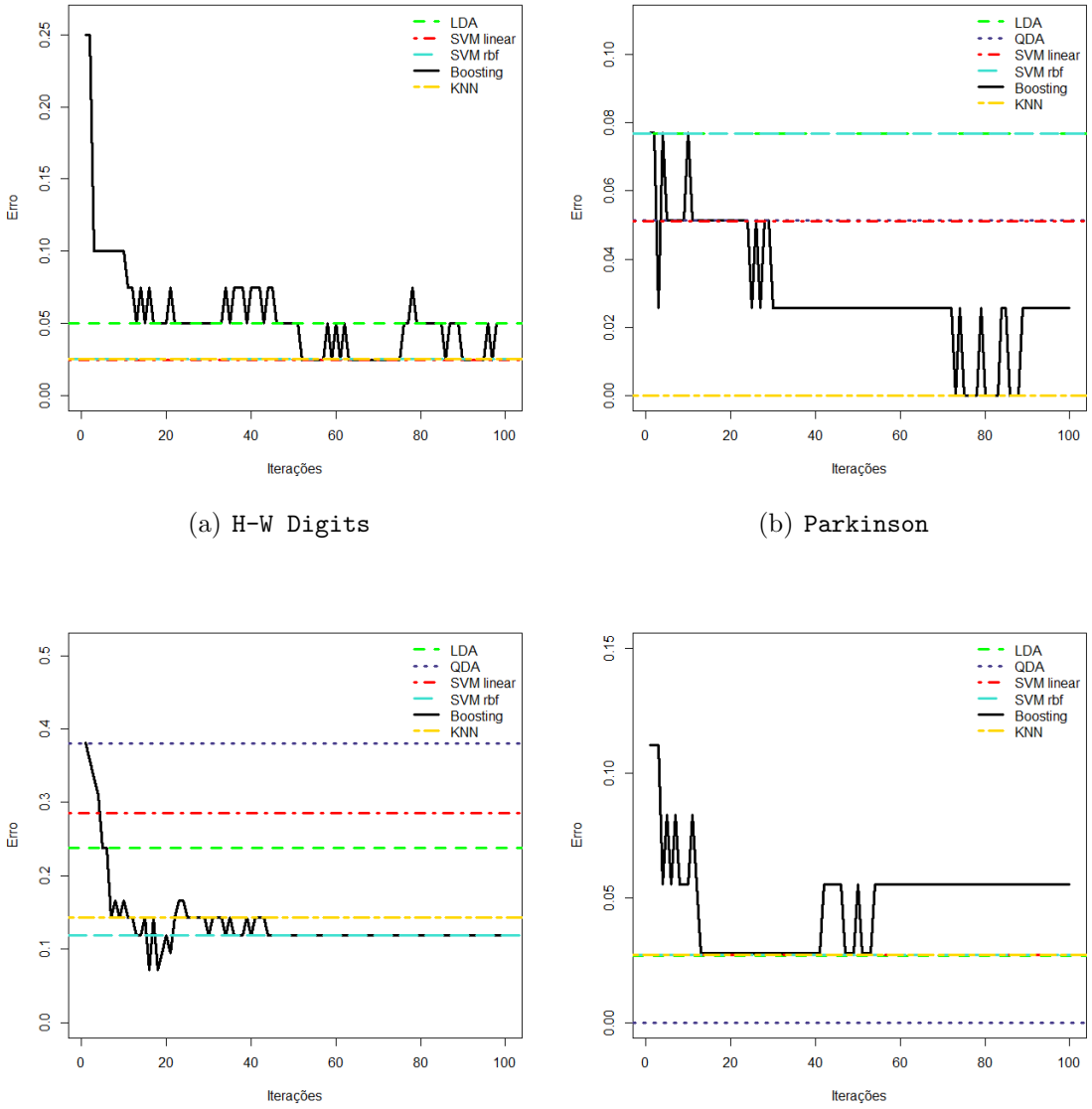
	`H-W Digits` (*Node*=3)	`Parkinson` (*Node*=4)	`Sonar` (*Node*=3)	`Wine` (*Node*=2)
Erro de treinamento	0	0	0	0
Erro de teste (*Holdout*)	0.05	0.025	0.119	0.055
Erro 10-fold	0.031	0.076	0.150	0.028

Tabela 5: Métricas de performance para os classificadores boosting.

Os valores assumidos pelas métricas de performance presentes na Tabela 5 evidenciam que o método de classificação em questão, em geral, performou melhor que os métodos clássicos de análise discriminante e retornou taxas de erro tão boas quanto as taxas retornadas pelas técnicas SVM e KNN.

4.4 Considerações finais

A fim de facilitar a comparação entre as taxas de erro dos diferentes métodos, e selecionando-se o erro de teste como fator de comparação, a Figura 8 apresenta o histórico das taxas de erro de teste do algoritmo boosting juntamente com as taxas retornadas pelos demais algoritmos testados.



(a) `H-W Digits` (b) `Parkinson`

(c) `Sonar` (d) `Wine`

Figura 8: Erros de teste segundo cada método implementado em cada base de dados.

Dentre as quatro bases de dados, apesar do algoritmo *AdaBoost* ser bem versátil com bases caracterizadas por múltiplas classes, este não teve um bom desempenho

na base `Wine` uma vez que o erro de teste, na melhor das hipóteses, chegou a ser igual aos erros de teste retornados pelos outros classificadores. Para as demais bases de dados o algoritmo performou bem estando sempre equiparado aos melhores classificadores de cada bases.

Com exceção da base `Wine`, na qual o classificador QDA obteve excelente desempenho, percebe-se um padrão no qual os erros de teste resultante dos métodos clássicos de análise discriminante (LDA e QDA) tendem a ser maiores que os erros das técnicas mais modernas. Outro fator que pode ser levado em conta a partir da análise dos gráficos é que a utilização do kernel gaussiano no SVM (SVM rbf) pode alterar significativamente a classificação, e a escolha da sua utilização ou não irá depender muito da base em questão.

Diferentemente dos demais classificadores, o algoritmo não paramétrico KNN mostrou-se surpreendentemente eficiente possuindo excelentes taxas de erro em todas as bases estudadas.

5 Conclusão

A aplicação dos métodos aqui apresentados evidenciou a sensibilidade das técnicas de classificação com respeito à natureza dos dados. Essa dependência de cada algoritmo com relação à natureza dos dados ocasionou diferentes resultados para as quatro bases analisadas, ou seja, algoritmos que obtiveram excelentes desempenhos em algumas bases, ocasionalmente, não conseguiam realizar a atividade de classificação com tanta precisão em outras bases. Apesar do observado, as técnicas mais modernas obtiveram um desempenho geral melhor do que as técnicas de análise discriminante, e o método KNN mostrou-se muito eficiente, provando que para se ter qualidade e precisão como as principais características de um classificador não necessariamente é preciso que este seja muito complexo.

A utilização das três métricas de performance deixou bem claro o otimismo presente no erro de treinamento, indicando ser mais conveniente a utilização do erro de teste ou o erro de validação cruzada. A escolha entre a utilização do erro de teste e do erro de validação cruzada irá depender do profissional e da finalidade da sua utilização.

Referências

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation*, 121:256–285.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *Computers and Geosciences,*. 11

Hastie, T., Tibshirani, R., & Friedman, J. (2013). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York. 10

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York.

Johnson, R. & Wichern, D. (2007). *Applied Multivariate Statistical Analysis*. Applied Multivariate Statistical Analysis. Pearson Prentice Hall. 15, 30

Karatzoglou, A., Meyer, D., & Hornik, K. (2006). Support vector machines in r. *Journal of Statistical Software*, 15. 35

Kohavi, R. (1995). A study of cross validation and bootstrap for accuracy estimation and model selection. *International Joint Conference on Articial Intelligence*. 31

Lichman, M. (2013). UCI machine learning repository. 11

Meyer, D. (2015). *Support Vector Machines* The Interface to libsvm in package e1071*. 18

Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective*. Adaptive computation and machine learning series. MIT Press. 10

Nilsson, N. J. (1998). *Introduction to Machine Learning*. Stanford University.

Orbanz, P. (2016). Statistical machine learning (stat w4400). http://www.stat.columbia.edu/p̃orbanz/W4400S14.html. 13, 20, 22, 30

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408. 14

Tax, D. M. & Duin, R. P. (2002). *Using two-class classifiers for multiclass classification*. Faculty of Applied Science, Delft University of Technology. 24, 25