Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Árvores de Decomposição Binária: Um novo esquema para a classificação de múltiplas classes

Caio Rocha Pereira

Trabalho de Graduação

Recife
28 de Novembro de 2018

Universidade Federal de Pernambuco
Centro de Informática

Caio Rocha Pereira

Árvores de Decomposição Binária: Um novo esquema para a classificação de múltiplas classes

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Renata Maria Cardoso Rodrigues de Souza*

Recife
28 de Novembro de 2018

Dedico este trabalho a meus pais, que desde criança me ensinaram a valorizar o conhecimento e o saber.

Agradecimentos

Quero agradecer a meus pais, que me apoiaram e me incentivaram para todas as decisões que tomei ao longo da vida. Pelo incentivo à ciência e a alimentação da minha curiosidade, que vieram tanto por tubos de ensaio, livros e microscópios de brinquedo quanto por revistas em quadrinhos, idas ao cinema ou bonecos. Agradeço imensamente por tudo que me proporcionaram desde minha infância até os momentos mais difíceis da graduação. Gostaria também de agradecer às minhas irmãs, pelas palavras motivadoras e por serem exemplos a serem seguidos que sempre admirarei.

Agradeço também à minha namorada, que sempre esteve disponível para me ouvir, tanto nos bons quanto nos maus momentos. Além disso, gostaria de agradecer a todos os meus amigos que me acompanharam por toda a graduação, sempre me motivando a seguir aquilo que eu buscava e facilitaram minha jornada.

Agradeço a todos da Oncase, por serem flexíveis com a demanda e por me darem espaço e tempo para a execução deste projeto.

Por fim, não poderia deixar de agradecer à minha orientadora, por todas as vezes que a pedi para sanar minhas dúvidas, por todas as vezes que pedi orientações e também por me acolher como seu monitor, aluno de Iniciação Científica e orientando.

Nem todos os que vagueiam estão perdidos
—J.R.R. TOLKIEN (O Senhor dos Anéis)

Resumo

A utilização de métodos de aprendizagem não supervisionada para particionamento dos dados pode trazer benefícios e contribuições consideráveis para a classificação de múltiplas classes. Portanto, este trabalho toma como objetivo a criação de um novo método utilizando conhecimentos de técnicas do estado da arte, como o agrupamento hierárquico, juntamente com as novas abordagens desenvolvidas, como a árvore binária de SVM. Após uma diversa e extensa pesquisa sobre o estado da arte em técnicas de classificação, binarização e agrupamento, foi possível elaborar o novo modelo, fazendo uso de tais técnicas. Testes foram realizados nas seguintes bases de dados reais: *absenteeism, forest types, glass, iris, image segmentation, image segmentation(test), wine, wine quality, zoo* e *wholesale customers*. O estudo foi feito visando comparar os resultados de métodos já existentes com o método proposto, podendo ser constatada a eficiência do novo modelo, assim como uma análise de configurações e cenários que o favorecem.

Palavras-chave: binarização, múltiplas classes, aprendizado supervisionado, agrupamento hierárquico, classificação

Abstract

The utilization of unsupervised learning methods for data partitioning can bring considerable benefits and contributions for multiclass classification. Therefore, the main goal of this project is the creation of a new method, by using knowledge of state-of-the-art techniques, such as hierarchical clustering, as well as new developed approaches, such as binary tree of SVM. After diverse and extense research about state-of-the-art techniques for classification, binarization and clustering, it was possible to elaborate the new model, making use of those techniques. Tests were made in the following real datasets: *absenteeism, forest types, glass, iris, image segmentation, image segmentation(test), wine, wine quality, zoo* and *wholesale customers*. The study was elaborated aiming to compare the results of existing methods with the proposed method, as well as an analysis of settings and scenarios that favor it.

Keywords: binarization, multiclass, supervised learning, hierarchical clustering, classification

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Com o surgimento de tecnologias para armazenamento e manipulação de dados, é possível verificar que muitos desses dados não são utilizados e por fim, acumulados, gerando bases de dados de grande volume. Com as tecnologias atuais, tornam-se cada vez menos interessantes tomadas de decisão que não envolvam análise dos dados. O crescimento deste tipo de decisão é explorado em [24]. As mesmas influenciam diretamente na decisão em diversos campos, tais quais *e-commerce*, ciências sociais, ciências biológicas, marketing digital etc.

A Aprendizagem de Máquina[20][3] é um campo cada vez mais considerado em aplicações tanto em empresas do setor privado, seja na geração de perfis de clientes usando *clustering*, como em [7], ou na classificação de clientes em bons e maus pagadores, por exemplo. Tal campo, diante das tecnologias de armazenamento de dados atuais, se torna ainda mais útil pela grande quantidade de informação e *insights* possíveis ao analisar tais dados. Com o interesse em tal informação, o desenvolvimento desse campo é um dos mais explorados pelos pesquisadores da ciência da computação no mundo todo.

A binarização de problemas de múltiplas classes [13] é constantemente empregada em problemas de aprendizagem de máquina, visto que muitos dos problemas de classificação são de múltiplas classes. A técnica é comumente utilizada pois reduz um problema complexo à um número x de problemas mais simples. Tal número x pode variar conforme a técnica empregada para a binarização, sendo variado sempre em termos do número n de classes. As técnicas mais aplicadas a esse tipo de problema são *One-vs-Rest* e *One-vs-One*[1][3]. Caso a técnica seja *One-vs-Rest*, são usados n classificadores, e cada classificador é treinado especificamente para uma classe contra todas as outras, transformando o problema de n classes em n problemas binários. Contudo, tal técnica apresenta o problema advindo do desbalanceamento da base de treinamento[7], uma vez que ao tomar uma base balanceada, a mesma se torna desbalanceada ao mesclar diferentes classes em uma única. Além disso, para problemas simples, a abordagem pode apresentar bons resultados, o que normalmente não é verdade para problemas mais complexos, vistas as sensibilidades da técnica.

Já se a técnica for *One-vs-One*, o problema se reduz a $\frac{n \times (n-1)}{2}$ problemas binários, uma vez que treina $\frac{n \times (n-1)}{2}$ classificadores com todas as combinações de classes dois a dois[3]. Essa abordagem não gera problemas de desbalanceamento como a anterior. Porém, utiliza um número muito alto de classificadores para esse tipo de problema. Apesar de apresentar alto desempenho, essa técnica é uma das mais, se não a mais complexa técnica de binarização, e para um número alto de classes, o problema se torna irredutivelmente caro.

Por isso, há grande interesse por parte de pesquisadores do campo na busca de novas técnicas, abordagens e estratégias de binarização para estes problemas. Este trabalho visa explorar

a possibilidade do uso de técnicas de agrupamento para manter a eficácia das técnicas atuais, com menor número de classificadores e portanto, com menor complexidade.

1.1 Objetivos

Este trabalho visa apresentar a Árvore de Decomposição Binária: um novo método de esquematização, com o propósito de otimizar a classificação de múltiplas classes por meio do aprendizado não supervisionado.

Será apresentado um esquema de decomposição binária no qual há n-1 classificadores, o que, portanto, é um sistema menos complexo. O método se baseia em trabalhos já realizados, visando abordagens *multiclass* para o SVM, como [11] e [5]. Também são usados conceitos fundamentais da aprendizagem não supervisionada, pelo fato do método ser construído com base em um. O modelo também se baseia na teoria Bayesiana[8], por utilizar conceitos fundamentais da mesma em suas predições.

Com o objetivo de avaliar o desempenho do método proposto e dos demais métodos presentes na literatura, um estudo comparativo foi feito utilizando a validação cruzada repetida. Foram planejados experimentos com conjuntos de dados reais e sintéticos, os classificadores usados foram a Árvore de Decisão, Regressão Logística, Máquinas de Vetores de Suporte, *Perceptrons* multicamadas e k Vizinhos mais próximos e as métricas usadas para a avaliação foram a métrica Kappa e a acurácia.

1.2 Estrutura do trabalho

Este trabalho é dividido conforme a seguinte descrição: O capítulo 2 apresenta o estado da arte em técnicas de binarização, aprendizagem supervisionada e não supervisionada; O capítulo 3 explica a funcionalidade do método proposto, bem como a metodologia usada neste trabalho; O capítulo 4 apresenta os experimentos realizados e os resultados obtidos, bem como as análises obtidas através destes. Por fim, o capítulo 5 apresenta as conclusões extraídas a partir dos resultados encontrados e possíveis melhorias e desafios futuros.

CAPÍTULO 2

Fundamentos

Muitos dos esquemas de classificação binária para múltiplas classes envolvem diferentes abordagens [13] para treinar seus classificadores. Neste capítulo, será visto como os dois principais métodos de esquematização funcionam, bem como serão explorados cinco dos mais usados classificadores, além de métricas de classificação e métodos de aprendizado não-supervisionado.

2.1 Esquemas de Binarização

Para muitos dos classificadores, a classificação de múltiplas classes é impossível. Classificadores de estimação direta do hiperplano, como Regressão Logística ou Máquinas de Vetores de Suporte [3], estimam apenas uma única fronteira de decisão, o que os torna obsoletos se usados unicamente nesse tipo de problema. A figura 2.1 apresenta um exemplo de problema do tipo *multiclass*. As técnicas de binarização surgiram como uma tentativa de tornar esse tipo de classificador útil em problemas de múltiplas classes, transformando o problema de N classes em K problemas binários, onde K pode variar conforme o método usado para a binarização. Nesta seção, veremos como funcionam os métodos mais utilizados para essa técnica: *One-vs-One* e *One-vs-Rest*.

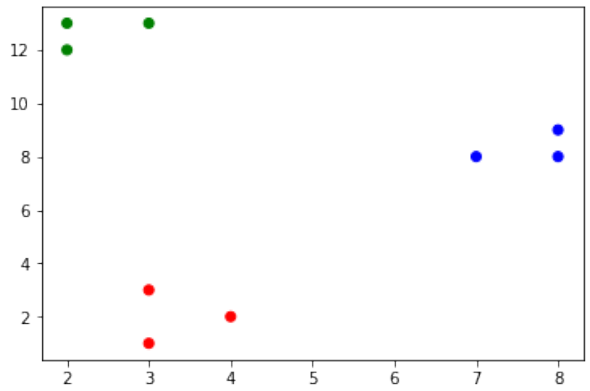
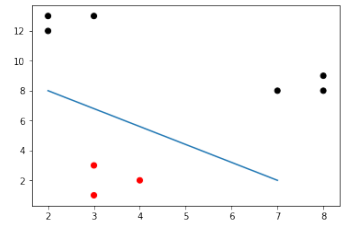


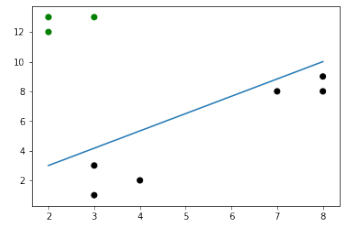
Figura 2.1: Exemplo de problema de múltiplas classes.

2.1.1 One-vs-Rest

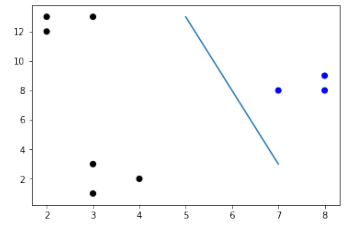
Essa abordagem consiste em treinar N classificadores. Ou seja, para cada classe, treinamos um classificador binário. Isso se torna possível se treinarmos cada classificador se especiali-



(a) Hipótese do classificador da classe vermelha



(b) Hipótese do classificador da classe verde



(c) Hipótese do classificador da classe azul

Figura 2.2: Esquema de binarização de classificadores *One-vs-Rest*

zando em uma classe, treinando a mesma como 1 e todo o resto como 0[1]. Ao fazer isso, criamos um sistema no qual cada classificador representa uma classe, e portanto, quando um classificador prediz a resposta como 1 a um novo padrão, significa que a classe associada àquele classificador é a certa para aquele padrão[3]. A figura 2.2 mostra a resolução do problema da figura 2.1 por um esquema One-vs-Rest. O problema advindo desse tipo de solução é o desbalanceamento gerado ao particionar os dados dessa forma. Sendo o problema de três classes, e assumindo o balanceamento da base, cada classificador será treinado com $\frac{n}{3}$ exemplos contra $\frac{n \times 2}{3}$ exemplos, sendo n o tamanho da base. Para muitos classificadores sensíveis a desbalanceamento, esse problema torna totalmente impraticável o uso da técnica, mesmo ela sendo a menos custosa.

2.1.2 One-vs-One

A alternativa mais utilizada, mais eficiente e também mais custosa em termos de decomposição binária é a estratégia *One-vs-One*. A estratégia, diferentemente da anterior, gera $\frac{n \times (n-1)}{2}$ classificadores, pois combina as classes dois a dois e com isso, transforma um

problema de n classes em $\frac{n \times (n-1)}{2}$ problemas binários. A predição é dada pelo comitê de todos os classificadores formados[3]. Além disso, algumas abordagens sugerem a utilização de probabilidades ao invés do voto como um número discreto[13]. Dessa forma, a classe de maior probabilidade a posteriori será a escolhida.

Em bases balanceadas, essa estratégia funciona de forma coerente com o classificador escolhido, além de não gerar o problema do desbalanceamento. Apesar de mostrar grande eficácia sobre quase todos os métodos propostos para decomposição binária até então[13], a mesma tem o defeito de ser custosa. Dependendo do número de classes e do classificador, o tempo de treinamento pode ser demasiado e tornar a técnica menos interessante do que o próprio *One-vs-Rest*, dependendo do ganho em métricas de avaliação de classificação[1]. Uma representação de solução de problema de múltiplas classes por meio de *One-vs-One* pode ser visto na figura 2.3.

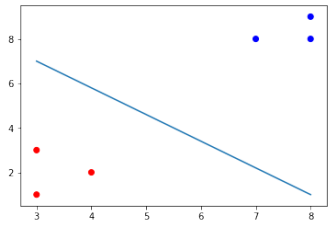
Apesar de pouco falada nos últimos anos, com o advento do *Deep Learning* e consequente retorno das redes neurais, a binarização é uma técnica fundamental para classificadores bastante eficientes como SVM e Regressão Logística. Além disso, alguns estudos recentes sugerem a binarização como prática preferível à abordagens *multiclass* de classificadores[13]. Portanto, é uma área que tem muito a ser explorada e expandida.

2.2 Aprendizado não Supervisionado

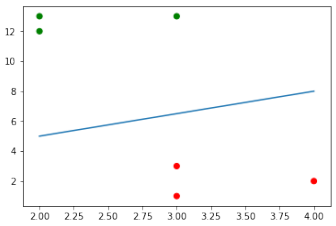
O aprendizado não supervisionado consiste em achar rótulos para diferenciar instâncias não rotuladas. Muitos associam o aprendizado não supervisionado com o agrupamento, pelo fato do mesmo ser usado na maioria das vezes para agrupar dados em diferentes *clusters*. O aprendizado não supervisionado tem muitos paradigmas, como a divisão entre *hard* e *fuzzy*[8]. Para o agrupamento *hard*, aprende-se apenas a qual grupo cada instância pertence, enquanto no agrupamento *fuzzy*, aprendemos o grau de pertinência de cada instância para cada grupo, numa abordagem mais probabilística[16]. Um dos algoritmos mais usados, em ambas as abordagens é o algoritmo K Médias.

2.2.1 K Médias

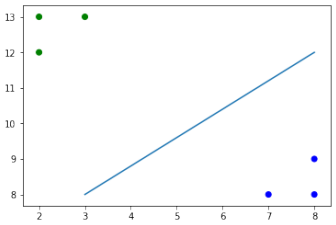
O algoritmo K Médias consiste em particionar os dados por meio de protótipos[8], representados por um vetor de médias de cada atributo dos dados. O algoritmo consiste em três passos. O primeiro passo consiste em inicializar os protótipos. Os mesmos são inicializados como vetores de mesma dimensão dos dados, com números aleatórios. O segundo passo é associar os dados mais próximos de cada protótipo à classe do mesmo. Para cada dado, verificamos a qual protótipo o mesmo está mais próximo, e o associamos à ele. Em seguida, atualizamos os protótipos para a média dos exemplos associados a ele. Os últimos dois passos se repetem até a convergência, ou seja, até os protótipos não mudarem. A figura 2.4 exemplifica um caso de uso típico de partição *hard* com a visualização sob duas dimensões dos dados da famosa base *iris*.



(a) Hipótese do classificador treinado para classe vermelha versus classe azul



(b) Hipótese do classificador treinado para classe verde versus classe vermelha



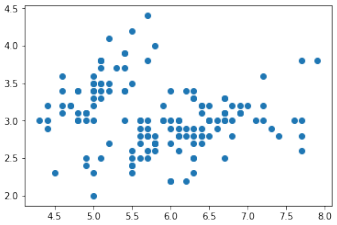
(c) Hipótese do classificador treinado para classe azul versus classe verde

Figura 2.3: Esquema de binarização de classificadores *One-vs-One*

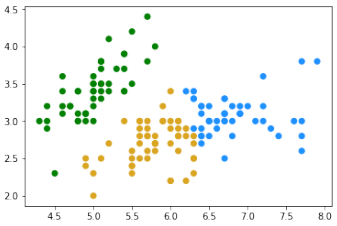
2.2.2 Agrupamento Hierárquico

Outra abordagem do aprendizado não supervisionado é agrupar os dados em múltiplos níveis[8]. Essa abordagem é interessante por permitir que analisem-se os dados de forma hierárquica, observando como os mesmos se agrupam de forma "multiparamétrica", como se simulássemos múltiplos K Médias, com K variando de 2 até o número de exemplos no conjunto de dados.

A principal diferença é que o agrupamento hierárquico tem dependência entre os níveis de partição[31]. Esse tipo de coisa pode não ocorrer em algoritmos como o possível K Médias hierárquico, pois o mesmo pode dividir agrupamentos em níveis posteriores, quebrando a consistência de níveis anteriores. A principal característica do agrupamento hierárquico é a visualização multinível do mesmo. Essa visualização se chama dendrograma. Um exemplo de dendrograma para a visualização do agrupamento a hierárquico do problema da figura 2.1 pode ser visto na figura 2.5. Os dois principais algoritmos associados a essa abordagem são o Agrupamento Divisivo e o Agrupamento Aglomerativo[8].



(a) Dados não agrupados



(b) Dados agrupados

Figura 2.4: Visualização do agrupamento por K Médias da base de dados *iris*, com K igual a 3

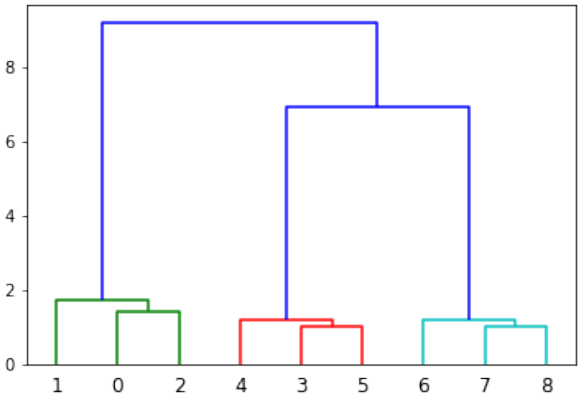


Figura 2.5: Exemplo de Dendrograma.

2.2.2.1 Agrupamento Aglomerativo

Esse método é o mais comum dentre todos os tipos de agrupamento hierárquico. O agrupamento aglomerativo se trata de uma abordagem *bottom-up*[8] onde começa-se com os próprios exemplos como grupos no nível mais baixo, e a partir de dois grupos por iteração, cria-se um novo grupo, um nível acima. Essas iterações se repetem até se formar um único grupo englobando todos os exemplos. As operações feitas em cada iteração consistem em agir sobre uma matriz de distâncias entre os exemplos. Para criar um novo grupo, selecionamos os dois exemplos com menor distância na matriz. Após isso, cria-se um grupo com os dois exemplos, e substitui-se os dois exemplos na matriz pelo grupo. A distância nova a ser entrada na matriz é feita utilizando a fórmula de recorrência, que pode ser a distância máxima entre os exemplos e o *cluster*, a distância mínima ou mesmo a média.

2.2.2.2 Agrupamento Divisivo

Essa técnica é menos usada para o agrupamento hierárquico pelo fato de ser mais custosa. A mesma consiste em uma abordagem *top-down*, oposta ao agrupamento aglomerativo, onde inicia-se com um único grupo e gradualmente divide os agrupamentos até que cada grupo seja um exemplo[31]. Como o primeiro passo do algoritmo envolve todas as combinações de divisões possíveis em dois grupos, visando maximizar a matriz de dissimilaridade, seu uso se torna impraticável para um número alto de elementos[10].

O agrupamento hierárquico tem fundamental importância para a execução desse trabalho, pois apresenta a característica de apresentar múltiplos níveis de agrupamento de forma simultânea. Isso se torna muito interessante para a decomposição binária, pois para cada nível, podemos construir um classificador de forma esquemática, seguindo uma arquitetura de árvore. A descrição do algoritmo proposto se encontra no capítulo 3. Na próxima seção, veremos alguns fundamentos e algoritmos de aprendizado supervisionado.

2.3 Aprendizado Supervisionado

Diferentemente do aprendizado não supervisionado, essa parte do aprendizado de máquina se volta à resolução de problemas que possuam um rótulo para os cada um dos exemplos[3]. São problemas recorrentes na aprendizagem de máquina, pois podem servir, por exemplo, para saber se determinado cliente é bom ou mau pagador ou para saber se certo acidente é fraude ou não[12]. Essas práticas fazem parte da rotina de desenvolvedores e cientistas de dados que trabalham em bancos, empresas especializadas em crédito ou mesmo corretoras de seguro.

Os algoritmos de aprendizado supervisionado se dividem em como os rótulos estão associados aos dados. Se os mesmos forem discretos, indicando um número de possibilidades finitas para a os rótulos, o problema é o de classificação, enquanto se os mesmos forem contínuos, o problema se torna o de regressão[3]. Classificadores e regressores se dividem em diversas abordagens ao longo da literatura do aprendizado supervisionado[20]. Nesta seção, são enfatizados os classificadores usados para este trabalho bem como métricas e fundamentos para tais métricas.

2.3.1 Algoritmos de Aprendizado

Para este trabalho, foi visada uma certa gama diversificada de algoritmos para que fosse averiguada, sem viés, a eficácia do modelo proposto. Para isso, foram usadas árvores de decisão, *perceptrons* multicamadas, k-Vizinhos mais próximos[13], máquinas de vetores de suporte e regressão logística[1]

2.3.1.1 Regressão Logística

A regressão logística consiste em uma abordagem direta para a estimação de pesos de um hiperplano que separe os rótulos do problema em duas classes[13], além do uso de uma função sigmóide[11] para a ativação ou classificação. Essa função consiste em uma curva em S, de forma que valores muito baixos da soma ponderada das entradas pelos pesos resultem em zero,

e que valores muito altos resultem em um. Após essa modelagem, o problema se torna um problema de otimização para achar o hiperplano que minimize o erro do classificador. Abordagens mais comuns para otimizar esse problema são o gradiente conjugado de Newton[21], algoritmos genéticos[17] ou até mesmo o gradiente descendente[19], muito usado nas redes neurais.

2.3.1.2 Árvore de Decisão

Esse grupo de algoritmos são conhecidos por árvores por tomarem decisões aninhadas[13]. As árvores são criadas com sua raiz e seus primeiros nós em atributos categóricos considerados mais importantes para a separação dos dados, e conforme são construídas, os atributos menos importantes são utilizados para a classificação. As funções mais utilizadas para essa medição de importância dos atributos são funções baseadas em entropia, como o ganho de informação[25], por exemplo.

Essa classe de algoritmos possuem diversas variações, como algoritmos que podam certos nós, de forma a evitar a especialização, ou algoritmos de comitês de árvores, onde cada árvore é treinada com parte dos atributos, como o *Random Forest* ou uso das árvores para *bagging* e *boosting*[26].

2.3.1.3 Máquinas de Vetores de Suporte

Esse algoritmo se tornou bastante popular na década de 1990, fase de certa decadência temporária das redes neurais, quando Vapnik propôs métodos de *Kernel* para a separação não linear dos dados.

As máquinas de vetores de suporte(em inglês, *Support Vector Machines*, assim como a regressão logística, são classificadores que estimam diretamente o hiperplano de separação das classes[3]. A principal diferença entre os dois é o fato dos SVM maximizarem a margem da classificação, tornando-os conhecidos como classificadores de larga margem.

O nome desse tipo de classificador é devido ao fato dos mesmos buscarem justamente os vetores de suporte, que estão nos limites intrínsecos de suas classes para estimar a mais larga margem. Esse tipo de otimização envolve certas restrições, pois a margem tem que estar restrita aos vetores de suporte[15]. Por isso, o método mais comum de otimização para os SVM é através dos multiplicadores de Lagrange[23].

O grande motivo desse classificador ser popular até os dias atuais é o fato do mesmo não ter limites em termos de fronteiras não lineares. Esse grande diferencial se deu por meio dos *Kernels* e pelo popular *Kernel Trick*, onde através de uma função simétrica, normalmente uma Gaussiana, os dados são elevados a um espaço de maior dimensionalidade, onde os mesmos são linearmente separáveis e portanto, torna as máquinas de vetores de suporte um forte candidato a qualquer problema de classificação.

2.3.1.4 Perceptrons Multicamadas

Os perceptrons são classificadores lineares propostos por Rosenblatt[27]. Por serem apenas classificadores lineares, foram aos poucos se tornando menos populares no meio acadêmico, abrindo muito o espaço para outros classificadores.

Na década de 1980, pesquisadores da área propuseram um método que pudesse quebrar a linearidade do perceptron adicionando aos mesmos duas características fundamentais: uma função de ativação, e uma topologia mais complexa. Com isso, as populares redes neurais ganharam grande espaço no meio acadêmico, vista a diversa gama de combinações de topologia, função de ativação ou mesmo unidade básica.

Outra coisa que as tornou bastante populares foi o *backpropagation*[29], algoritmo que se trata de uma versão estendida do gradiente descendente, mas que minimiza o erro para todos os neurônios de todas as camadas de uma rede neural. A rede estima, similarmente ao SVM, um hiperplano em um espaço de dimensão elevada, a depender do número de camadas da mesma.

2.3.1.5 k-Vizinhos mais Próximos

Esse classificador é o mais simples de todos aqui listados. No mesmo, não há uma fase de treinamento, e ele se trata apenas de um cálculo de distâncias de um novo exemplo para todos os exemplos da base. Para predizer a classe de um novo exemplo, filtramos toda a base e a reduzimos apenas aos k vizinhos mais próximos[9]. Com isso, a classe majoritária dentro desse subconjunto é a escolhida para o exemplo.

Outra forma de usar os k vizinhos mais próximos comumente empregada em tarefas de classificação é ponderar o voto de cada vizinho pelo inverso da distância[9]. Assim, a classe de "soma"maior é a escolhida. Possibilidades assim levaram à sugestões de classificar exemplos usando a base toda ponderando cada exemplo pelo inverso da distância.

Como se trata de um classificador baseado em distâncias, muitos problemas podem ter a escolha de determinado tipo de distância como um parâmetro circunstancial. As métricas de distância mais comuns são a euclidiana, a *City-Block*, a *Chebyshev* e variações da *Minkowski*[6].

Esse tipo de algoritmo estima o hiperplano de maneira indireta, pois não é possível traçar um hiperplano apenas com os parâmetros do mesmo.

2.3.2 Métricas para Classificadores

As métricas mais utilizadas para comparação entre classificadores são métricas baseadas em acertos tanto da classe positiva quanto da classe negativa. Quase todas podem ser representadas a partir da explicação da matriz de confusão. Portanto, vamos abordá-la na próxima seção, seguida de uma abordagem das duas métricas que usaremos neste trabalho.

2.3.2.1 Matriz de Confusão

A matriz de confusão consiste em uma tabela, ou uma matriz, na qual cada célula é uma quantificação do quão errado ou o quão certo um classificador está sob duas diferentes perspectivas: Uma relacionada à classe positiva, e outra relacionada à classe negativa[3].

As linhas da matriz representam as predições de um determinado classificador, enquanto as colunas representam a verdadeira classe dos exemplos para os quais foram geradas as predições (também conhecido como *ground truth*)[8].

Com essa descrição, a primeira célula da matriz é a contagem de verdadeiros positivos do

	Classe 1 verdadeira	Classe 0 verdadeira
Classe 1 predita	TP	FP
Classe 0 predita	FN	TN

Tabela 2.1: Tabela representativa de uma matriz de confusão

classificador, ou uma quantificação de acerto do classificador para a classe positiva. Na segunda célula consta a quantidade de exemplos da classe positiva erroneamente classificados, ou os falsos positivos, uma quantificação de erro do classificador para a classe positiva. A segunda linha, com as duas seguintes células, se tratam das mesma descrição que a linha anterior, com a diferença de se tratar da classe negativa.

A tabela 2.1 é uma representação do que a matriz e confusão é. Esse objeto é muito importante para o estudo da classificação pois boa parte das métricas usadas para avaliação de classificadores são feitas com base nas células da mesma.

2.3.2.2 Acurácia

Como uma das mais usadas métricas para todos os classificadores[13], a acurácia se trata de uma simples taxa de acerto que consiste no que o classificador acertou para as duas classes dividido por todas as predições feitas. Uma melhor representação é dada pela fórmula a seguir.

$$\text{Acurácia} = \frac{TP+TN}{TP+FP+FN+TN}$$

Apesar de direta quanto à qualidade de um classificador baseada em predições acertadas, essa métrica não é completamente ausente de problemas. Um muito claro é a sensibilidade da mesma ao desbalanceamento [4]. Um caso de uso pode ilustrar esse problema.

Em uma base de dados onde 90% é da classe positiva e 10% é da classe negativa, um classificador que sempre diga que exemplos são da classe positiva terá uma acurácia de 90%.

Portanto, a acurácia não é a métrica ideal para todas as bases, nem é a única métrica que deve ser mensurada na avaliação de classificadores.

2.3.2.3 Métrica Kappa de Cohen

O coeficiente *kappa* de Cohen é uma alternativa à acurácia muito usada por pesquisadores por compensar predições aleatórias[13]. O coeficiente avalia predições respectivas ao classificador em si, em contraste com predições que não podem ser atribuídas ao classificador por si só.

Uma maneira bastante usada de se calcular essa métrica é utilizando uma expansão da matriz de confusão vista anteriormente para um problema de m classes, onde cada linha é a classe predita e cada coluna é a classe verdadeira. Com isso, cada célula tem uma quantidade de predições errôneas, exceto na diagonal da matriz, onde tem-se os acertos de cada classe. A fórmula a seguir apresenta um breve esboço de como é calculado o coeficiente. As notações a serem observadas são: h_{ii} são os acertos de cada classe, ou células da diagonal da matriz, T_{ci} é o total de contagens de cada coluna da matriz, T_{ri} é o total de contagens de cada linha da matriz,

n é o número de exemplos e m é o número de rótulos de classe.

$$kappa = \frac{n\sum_{i=1}^{m} h_{ii} - \sum_{i=1}^{m} T_{ri}T_{ci}}{n^2 - \sum_{i=1}^{m} T_{ri}T_{ci}}$$

A métrica Kappa tem intervalo de -1 a 1, sendo -1 discordância total, 0 aleatoriedade e 1 total correspondência.

Com tudo que foi visto, foi possível teorizar sobre o possível comportamento do modelo que será visto à seguir, além de ser possível criar um esboço dos experimentos a serem feitos, quais classificadores testar, quais métricas usar e qual o melhor *pipeline* a ser feito para averiguar a eficiência do modelo a ser proposto.

2.4 Trabalhos relacionados

Diversos trabalhos foram feitos no âmbito de diminuir a complexidade de estratégias existentes, ou mesmo como tentativas de alavancar a capacidade de certos algoritmos a nível de múltiplas classes. Porém, poucos trabalhos apresentam a arquitetura de árvore como essência do modelo. Esses casos são Fei & Liu[11] e Cheong, Oh & Lee [5].

Por fim, é importante observar que existem muitos trabalhos comparativos das técnicas do estado da arte da esquematização de classificadores binários, especialmente [13]. Neste, destaca-se a importância do uso da binarização, defendendo-a como técnica padrão para problemas de múltiplas classes.

CAPÍTULO 3

Árvore de Decomposição Binária

A elaboração de um modelo representativo de múltiplas classes baseado em árvores já não é tão nova. Abordagens como [11] e [5] sugerem o uso da arquitetura de uma árvore binária para separar as classes em múltiplos níveis e portanto, obter melhor acurácia ou menor tempo de treinamento. Ambas as abordagens citadas consistem em uma árvore, na qual o classificador nos primeiros nós consiste em separar os dados em dois grandes grupos. A divisão em dois subgrupos ocorre recursivamente para cada grupo separado, formando uma árvore.

As principais diferenças da abordagem aqui proposta às previamente citadas são o agrupamento aglomerativo e a predição. O agrupamento hierárquico é empregado pelo fato de haver dependência entre separações entre níveis mais altos e mais baixos[8]. Em cada nível da árvore, as classes são separadas com base na separação do nível do agrupamento referente à altura do nó correspondente. Já a predição é dada pelo caminho na árvore. Ou seja, dado um padrão, o caminho que o mesmo faz da raiz até a folha, computando a classe majoritária da folha, ou classe de maior probabilidade a posteriori [8]. A figura 3.1 mostra um diagrama exemplo de como funciona essa arquitetura.

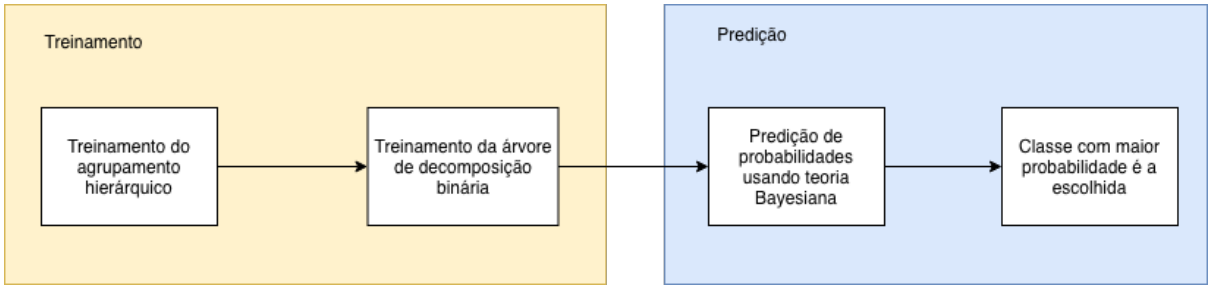


Figura 3.1: Diagrama representativo do funcionamento da Árvore de Decomposição Binária.

A figura 3.2 mostra graficamente a árvore de Cheong, Oh & Lee [5] treinada.

É interessante notar que a abordagem proposta nessa figura consiste em uma solução para o problema do SVM, devido ao fato do mesmo ser um dos poucos classificadores que não possui uma abordagem para múltiplas classes aceita sob consenso de toda a comunidade acadêmica.

O uso desse tipo arquitetura se beneficia do fato da mesma conseguir níveis de especificação de cada classificador. Por isso, é possível saber a qual *cluster* o novo padrão pertence em múltiplos níveis, de forma semelhante ao agrupamento hierárquico. Vista a semelhança ao agrupamento hierárquico, foi elaborada uma nova abordagem para essa arquitetura. As seções seguintes apresentam como a mesma foi elaborada.

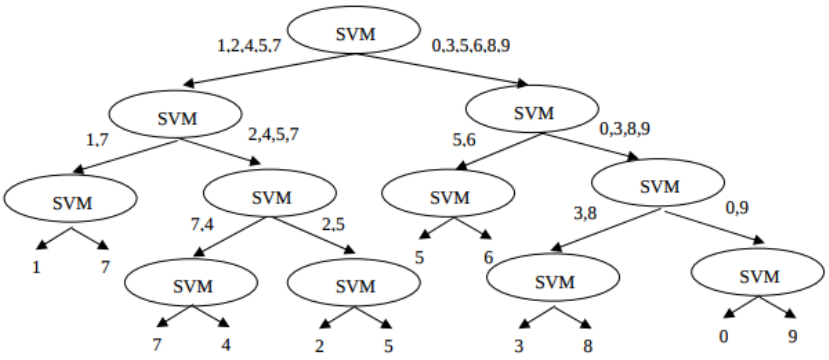


Figura 3.2: Árvore de SVM proposta por Cheong, Oh & Lee.

3.1 Treinamento

O treinamento da árvore começa adquirindo os rótulos dos grupos de um agrupamento hierárquico. Para isso, os dados são entrada para um algoritmo de agrupamento aglomerativo, e após isso são obtidos os rótulos. Visto que o desejável é ter menos classificadores que *One-vs-Rest*, são adicionados aos dados apenas os $n-1$ níveis do agrupamento, pois os classificadores são construidos sobre esses níveis.

Após isso, cria-se o nó raíz com um classificador, esse treinado com os dados e o primeiro nível do agrupamento aglomerativo. Em seguida, adiciona-se um nó à esquerda e à direita da raiz. Os dados de treinamento são separados entre as duas possíveis classes e vão cada parte para um lado resultante. Por exemplo: Se os dados forem da classe 0, vão para a esquerda, se são da classe 1, vão para a direita. É importante notar que classes, nesse âmbito, são referentes ao agrupamento aglomerativo no nível observado e não a variável independente y.

Ambos os nós verificam se no nível seguinte do dendrograma os dados são separados em mais de um rótulo. Se o forem, constrói-se outro classificador, que separa os dados entre outros dois nós, um à esquerda e outro à direita. Se não, o mesmo nó verifica se algum dos níveis seguintes possui mais de um rótulo, e assim constrói outro classificador, separando os dados entre um nó à esquerda e outro à direita.

Essa estrutura se repete recursivamente até que todos os níveis da hierarquia obtida sejam usadas para o treinamento. Ou seja, até que tenhamos uma estrutura de classificação hierárquica com $n-1$ classificadores. Dessa forma, ao final do treinamento, teremos uma estrutura hierárquica de classificadores, cada um em um nível do dendrograma.

Diferentemente da árvore de [5], o modelo proposto pode usar como base, qualquer classificador, sob a garantia que o mesmo tem a capacidade de extrair probabilidades para novos padrões. O porquê da necessidade dessa condição será melhor explicado na próxima seção.

Os algoritmos 1 e 2 são uma ilustração mais direta do algoritmo de treinamento da árvore. É importante entender que o parâmetro nível é usado também como um índice para que seja possível saber a coluna correta para a construção dos classificadores. Assim, gera-se a hierarquia de classificadores sob as quais podemos fazer predições para novos padrões.

Algorithm 1 Algoritmo de Treinamento da Árvore de Decomposição Binária

1: treine o algoritmo hierárquico sobre X
2: adicione as *labels* dos $n-1$ níveis do dendrograma como colunas em X
3: crie a raiz da árvore chamando o algoritmo de adequação do nó com os parâmetros: nível=0, limite=$n-1$, X e y.

Algorithm 2 Algoritmo de adequação de cada nó

1: **if** nível > limite **then** retorne o nó com X e y.
2: **else**
3: **if** tamanho dos rótulos = 1 **then** chame o algoritmo de adequação do nó com nível+1
4: **else if** tamanho dos rótulos = 2 **then**
5: crie um nó com X e y passados pela função
6: construa um classificador com X e a coluna do agrupamento referente ao nível
7: chame o algoritmo de adequação de nó para o nó filho da esquerda passando X e y particionados pelo primeiro rótulo do agrupamento, com o mesmo limite e nível+1
8: chame o algoritmo de adequação de nó para o nó filho da direita passando X e y particionados pelo segundo rótulo do agrupamento, com o mesmo limite e nível+1
9: retorne o nó gerado

3.2 Predição

Algo que é importante ser observado é o fato da árvore não gerar separações perfeitas. Isso fica evidente ao se notar que um dendrograma pode não representar, em determinado nível, as classes como elas exatamente são. Para contornar esse problema, é necessário o uso de probabilidades.

Uma probabilidade que se faz necessária para tal solução é a extração de probabilidades de classificadores. Em casos como as *Support Vector Machines*, que não se baseiam naturalmente em probabilidades, a solução é usar um estimador artificial. No caso de SVM, esse estimador pode ser elaborado por meio de casamentos par-a-par[30].

Portanto, a estimação de probabilidades é estimada de forma similar a uma rede *bayesiana*[8]: Como ter-se-á probabilidades binárias em cada nível da árvore com exceção das folhas, multiplica-se a probabilidade de um nó pertencer ao lado esquerdo pela probabilidade do nó do lado esquerdo, e o mesmo para o lado direito. Isso se repete até que o próximo nó no cálculo seja uma folha. Se assim for, é calculada na folha as probabilidades a priori, como porcentagens das classes da variável independente y e as mesmas entram no cálculo, resultando em uma soma ponderada dessas probabilidades a priori em todas as folhas existentes na árvore.

A figura 3.3 demonstra como poderíamos extrair probabilidades da árvore e por tanto realizar predições. No problema, há três classes e portanto dois classificadores, representados pelos ícones redondos. Nos ícones retangulares, estão representadas as folhas, que retornam em um nível abaixo, a probabilidade à priori, representada pelos ícones coloridos, sendo uma classe azul, uma classe verde e a outra vermelha.

Vamos agora observar como é feito o cálculo da probabilidade de um novo padrão X:

Começando pela direita, usa-se diretamente a probabilidade estimada pelo classificador para o padrão X ser da direita. Em seguida, multiplica-se a probabilidade pelo vetor da folha, ou as probabilidades a priori. Tem-se:

$$proba(x) = (p1|root) * [0.15, 0.65, 0.20]$$

Agora, adiciona-se ao cálculo a probabilidade do lado do esquerdo.

$$proba(x) = (p1|root) * [0.15, 0.65, 0.20] + (p0|root) * p(c0)$$

Em seguida, decompondo o p(c0) em outro cálculo, multiplicamos, igualmente à forma anterior, a probabilidade extraída do classificador do exemplo ser da esquerda, multiplicamos pelo vetor de probabilidades a priori da esquerda e repetimos o mesmo procedimento com a direita. Com isso, tem-se:

$$p(c0) = (p0|c0) * [0.10, 0.30, 0.60] + (p1|c0) * [0.50, 0.20.0.30]$$

Por fim, temos $proba(X) = (p0|root) * ((p0|c0) * [0.10, 0.30, 0.60] + (p1|c0) * [0.50, 0.20.0.30]) + (p1|root) * [0.15, 0.65, 0.20])$

Como se tratam de problemas de n classes, as probabilidades resultantes precisam ser um vetor de dimensão n. Por isso, o cálculo anterior resulta em um vetor de n elementos. Para extrair-se a classe resultante, basta extrairmos a classe de maior probabilidade a posteriori do vetor resultante do cálculo feito[8].

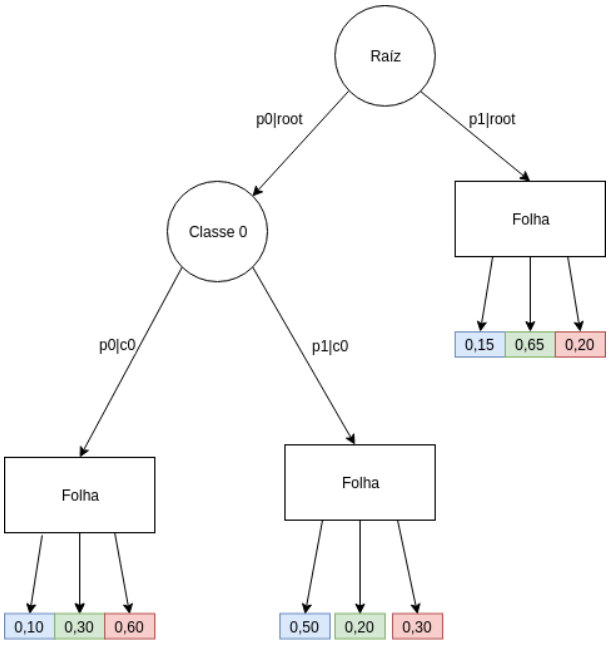


Figura 3.3: Ilustração da predição de probabilidades da árvore

CAPÍTULO 4

Experimentos

Para averiguarmos a eficácia do modelo proposto, é necessário testá-lo com as técnicas do estado da arte, *One-vs-Rest* e *One-vs-One*. Para isso, usaremos um *framework* similar ao usado por [13], testando as estratégias com múltiplos classificadores-base, múltiplas métricas e múltiplas bases. Não estão no escopo dessa experimentação testes comparando o modelo a estratégias de múltiplas classes do próprio classificador, pelo fato da utilização de esquemas de binarização ter sido validada como técnica ideal em boa parte dos problemas [13].

Nesse caso, serão testados os classificadores: Regressão Logística, Árvore de Decisão(C4.5), k Vizinhos mais próximos, Máquinas de vetores de Suporte e Perceptrons multicamadas. Algo que ficará fora do escopo desse estudo é a comparação do método com abordagens *multiclass* do próprio classificador, devido ao fato de ser comprovado o favorecimento de abordagens de binarização a estratégias do próprio classificador [13].

As bases utilizadas serão: *absenteeism, forest types, glass, iris, image segmentation, image segmentation test, wine, wine quality, zoo* e *wholesale customers*. Além disso, foram feitos experimentos em bases simuladas, por meio da biblioteca Scipy[18], da linguagem Python[28]. Essa biblioteca possui uma função que gera uma população simulada a partir de uma distribuição normal.

As métricas escolhidas foram as mesmas usadas em [13], a métrica Kappa de Cohen e acurácia, vistas no capítulo 2. Para extração das métricas, a prática utilizada foi a validação cruzada estratificada repetida, tanto nas bases simuladas quanto nas reais. São usados 10 *folds* e 10 repetições, gerando 100 resultados para cada métrica em cada base para cada classificador. Desses resultados, foram extraídos a média e o desvio padrão, para se ter uma noção da variabilidade da eficiência dos modelos.

4.1 Dados Simulados

A simulação dos dados é sempre sensível pois na maioria das vezes a mesma não representa o funcionamento do modelo em casos reais. Por isso, deve ser feita com bastante cautela.

Para gerar os dados, foi utilizada a biblioteca scipy[18], que possui a função de gerar populações simuladas com base em parâmetros para uma distribuição normal multivariada. Para isso, são passados os parâmetros: vetor de médias, matriz de covariância e número de indivíduos a serem simulados. As figuras 4.2, 4.4, 4.6, 4.8 mostram os parâmetros de cada distribuição usada para gerar os dados sintéticos. É importante ressaltar que os parâmetros foram ajustados de forma arbitrária, visando testar a sobreposição entre classes e a distância *intercluster*, por exemplo.

	Número de classes	Número de indivíduos	Balanceamento
Base 1	3	300	Sim
Base 2	3	400	Não
Base 3	6	600	Sim
Base 4	6	730	Não

Tabela 4.1: Características das bases de dados simulados

Foram geradas quatro bases, variando o número de indivíduos por classe, número de classes, sobreposição entre as classes e grau de desbalanceamento, ou diferença de número de indivíduos de uma classe a outra. Esses testes tornaram possível a visão de sucesso do modelo em diferentes características de dados, como é o caso do *One-vs-Rest*, famoso por ser sensível a desbalanceamento.

A tabela 4.1 mostra as características de cada base gerada, explanando também se há ou não balanceamento em cada base. É dado o seguimento, a partir da próxima seção, à análise de desempenho em cada base.

4.1.1 Primeira base

A primeira base consiste de 300 dados balanceados. Ou seja, 100 dados de cada classe. Cada classe é distribuída de forma normal, tal qual uma Gaussiana, como visto anteriormente. Uma visualização da primeira base pode ser vista na figura 4.1. A partir da visualização, é possível ter a noção de que se trata de um problema fácil. Apesar dos poucos *outliers*, vê-se que é os dados são bem separados.

A seguir, são mostrados os resultados de cada classificador e cada esquema sobre a base.

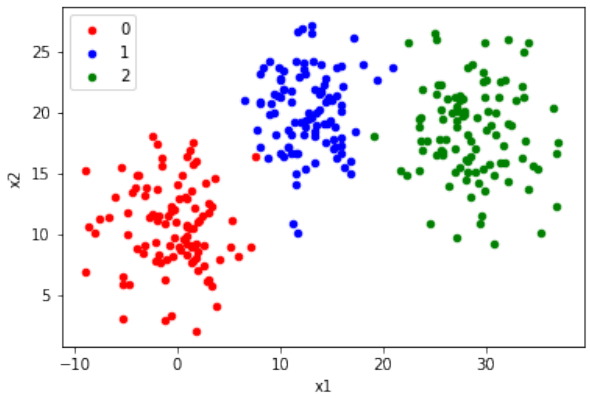


Figura 4.1: Gráfico de dispersão da primeira base.

Como vemos nas tabelas 4.2 e 4.3, os resultados da esquematização *One-vs-One* se mostram superiores às demais em todos os classificadores testados. Tanto pelas altas médias em relação quanto aos demais, quanto pelos baixos desvios, o que indicam maior consistência do modelo.

Apesar disso, pode-se também notar uma boa performance da árvore em relação ao *One-*

$$\mu = [0.0, 10.0], \sum = \begin{bmatrix} 12.0 & 0.0 \\ 0.0 & 12.0 \end{bmatrix}$$

(a) Parâmetros da classe 0 na primeira base

$$\mu = [13.0, 20.0], \sum = \begin{bmatrix} 10.0 & 0.0 \\ 0.0 & 10.0 \end{bmatrix}$$

(b) Parâmetros da classe 1 na primeira base

$$\mu = [29.0, 18.0], \sum = \begin{bmatrix} 15.0 & 0.0 \\ 0.0 & 15.5 \end{bmatrix}$$

(c) Parâmetros da classe 2 na primeira base

Figura 4.2: Parâmetros das distribuições Gaussianas na primeira base

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.84	0.06	**0.99**	0.02	0.98	0.02
LR	0.85	0.05	**0.97**	0.03	0.96	0.03
AD	0.98	0.02	**1.0**	0.0	0.98	0.03
MLP	0.90	0.11	**0.97**	0.03	0.92	0.09
KNN	0.99	0.02	**1.0**	0.01	0.98	0.03

Tabela 4.2: Acurácia dos classificadores na primeira base por esquema, com média e desvio padrão

vs-Rest, no qual era esperado ter-se uma melhor performance pelo fato do mesmo ser mais complexo, ou possuir mais classificadores.

4.1.2 Segunda base

A primeira base consiste de 400 dados desbalanceados, possuindo a classe majoritária 250 exemplos, a classe minoritária 50 e a terceira classe 100 exemplos. Além disso, aumenta-se a sobreposição entre as classes visando observar a mudança na performance com esse tipo de alteração.

Como vemos nas tabelas 4.4 e 4.5, o modelo *One-vs-One* se mantém superior aos demais, apesar de possuir Desvio Padrão mais elevado em alguns casos, como SVM e Regressão Logística. Algo interessante a se notar é o fato da árvore se mostrar superior a *One-vs-Rest* justamente nos mesmos classificadores. Ambos classificadores são estimadores diretos do hiperplano, di-

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.76	0.09	**0.99**	0.02	0.97	0.03
LR	0.77	0.07	**0.95**	0.04	0.94	0.05
AD	0.97	0.03	**1.0**	0.0	0.97	0.04
MLP	0.85	0.17	**0.95**	0.04	0.88	0.13
KNN	0.98	0.03	**0.99**	0.01	0.97	0.04

Tabela 4.3: Kappa dos classificadores na primeira base por esquema, com média e desvio padrão

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.71	0.03	**0.91**	0.04	0.74	0.01
LR	0.73	0.04	**0.87**	0.05	0.75	0.02
AD	0.82	0.05	**0.99**	0.02	0.74	0.02
MLP	0.78	0.08	**0.84**	0.05	0.72	0.04
KNN	0.84	0.06	**0.99**	0.02	0.74	0.02

Tabela 4.4: Acurácia dos classificadores na segunda base por esquema, com média e desvio padrão

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.48	0.05	**0.83**	0.08	0.40	0.04
LR	0.50	0.06	**0.74**	0.09	0.39	0.05
AD	0.67	0.09	**0.98**	0.04	0.40	0.05
MLP	0.57	0.16	**0.70**	0.09	0.33	0.14
KNN	0.71	0.11	**0.98**	0.04	0.41	0.05

Tabela 4.5: Kappa dos classificadores na segunda base por esquema, com média e desvio padrão

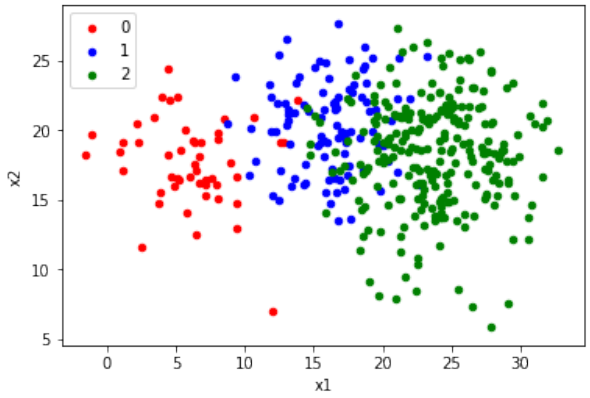


Figura 4.3: Gráfico de dispersão da segunda base.

$$\mu = [6.0, 17.0], \sum = \begin{bmatrix} 12.0 & 0.0 \\ 0.0 & 12.0 \end{bmatrix}$$

(a) Parâmetros da classe 0 na segunda base

$$\mu = [16.0, 20.0], \sum = \begin{bmatrix} 10.0 & 0.0 \\ 0.0 & 10.0 \end{bmatrix}$$

(b) Parâmetros da classe 1 na segunda base

$$\mu = [24.0, 18.0], \sum = \begin{bmatrix} 15.0 & 0.0 \\ 0.0 & 15.5 \end{bmatrix}$$

(c) Parâmetros da classe 2 na segunda base

Figura 4.4: Parâmetros das distribuições Gaussianas na primeira base

ferentemente dos demais listados nesse estudo.

4.1.3 Terceira base

A terceira base consiste de 6 classes balanceadas, com 100 exemplos cada uma. A base apresenta pouca sobreposição entre as classes e assim como a primeira base, se trata de um problema "fácil".

As tabelas 4.6 e 4.7 apresentam os resultados para os classificadores e esquemas previamente citados. Os resultados comprovam e reiteram o que foi observado nos resultados para a segunda base, nos quais a árvore tem melhores resultados que o *One-vs-Rest* nos classificadores SVM e Regressão Logística.

Algo ímpar destes resultados é a acurácia e Kappa baixos observados no *One-vs-Rest*. Isso

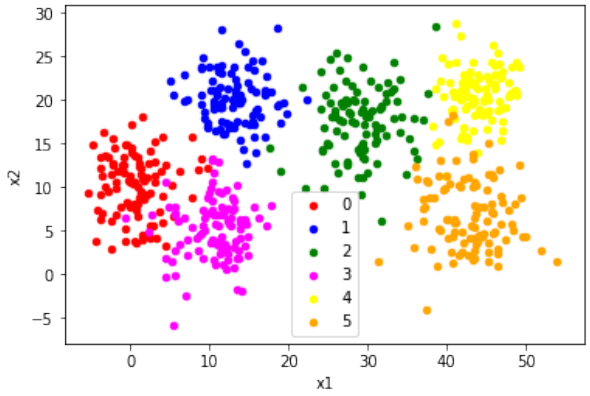


Figura 4.5: Gráfico de dispersão da terceira base.

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.66	0.04	**0.96**	0.02	0.82	0.06
LR	0.65	0.04	**0.92**	0.03	0.80	0.06
AD	0.92	0.04	**0.99**	0.01	0.82	0.06
MLP	**0.69**	0.11	0.63	0.02	0.63	0.11
KNN	0.94	0.03	**0.99**	0.01	0.81	0.06

Tabela 4.6: Acurácia dos classificadores na terceira base por esquema, com média e desvio padrão

se deve ao fato discutido previamente do mesmo criar o problema do desbalanceamento para seus classificadores. Em bases com alto número de classes, esse problema se torna ainda mais agravante na performance do esquema.

4.1.4 Quarta base

A quarta base consiste de 730 exemplos, com 6 classes desbalanceadas, sendo a classe minoritária com 30 e a majoritária com 250 exemplos. A sobreposição entre as classes é aumentada, tal qual a segunda base.

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.60	0.05	**0.96**	0.03	0.79	0.07
LR	0.58	0.05	**0.90**	0.04	0.76	0.07
AD	0.90	0.05	**0.99**	0.02	0.79	0.08
MLP	**0.62**	0.14	0.56	0.25	0.55	0.13
KNN	0.92	0.03	**0.99**	0.02	0.77	0.07

Tabela 4.7: Kappa dos classificadores na terceira base por esquema, com média e desvio padrão

$$\mu = [0.0, 10.0], \sum = \begin{bmatrix} 12.0 & 0.0 \\ 0.0 & 12.0 \end{bmatrix}$$

(a) Parâmetros da classe 0 na terceira base

$$\mu = [13.0, 20.0], \sum = \begin{bmatrix} 10.0 & 0.0 \\ 0.0 & 10.0 \end{bmatrix}$$

(b) Parâmetros da classe 1 na terceira base

$$\mu = [29.0, 18.0], \sum = \begin{bmatrix} 15.0 & 0.0 \\ 0.0 & 15.5 \end{bmatrix}$$

(c) Parâmetros da classe 2 na terceira base

$$\mu = [10.0, 5.0], \sum = \begin{bmatrix} 12.0 & 0.0 \\ 0.0 & 12.0 \end{bmatrix}$$

(d) Parâmetros da classe 3 na terceira base

$$\mu = [43.0, 20.0], \sum = \begin{bmatrix} 10.0 & 0.0 \\ 0.0 & 10.0 \end{bmatrix}$$

(e) Parâmetros da classe 4 na terceira base

$$\mu = [42.0, 7.0], \sum = \begin{bmatrix} 15.0 & 0.0 \\ 0.0 & 15.5 \end{bmatrix}$$

(f) Parâmetros da classe 5 na terceira base

Figura 4.6: Parâmetros das distribuições Gaussianas na terceira base

As tabelas 4.8 e 4.9 apresentam os resultados dos classificadores e esquemas.

Algo que pode ser observado é, mais uma vez, a baixa performance do OvR com o aumento do número de classes. Outra observação a ser feita é a manutenção da superioridade da árvore sobre o OvR tanto em Kappa quanto em acurácia nos classificadores SVM e Regressão Logística e MLP em acurácia.

Algo que pode ser observado em todos os casos é a superioridade do *One-vs-One* em todos os casos, com uma ou outra exceção de variabilidade(desvio padrão). A árvore se mostra pouco efetiva em relação ao *One-vs-One*, com exceção de alguns casos.

É importante observar a troca feita entre performance e complexidade do modelo. Enquanto

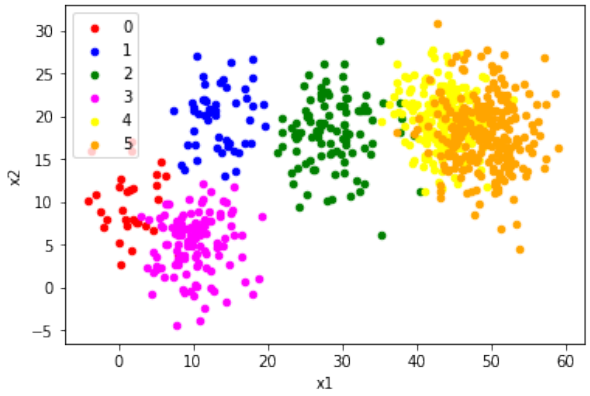


Figura 4.7: Gráfico de dispersão da quarta base.

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.50	0.04	**0.86**	0.04	0.70	0.03
LR	0.50	0.04	**0.83**	0.04	0.69	0.02
AD	0.79	0.04	**0.97**	0.03	0.71	0.02
MLP	0.52	0.12	0.48	0.02	**0.56**	0.10
KNN	0.84	0.04	**0.99**	0.01	0.70	0.02

Tabela 4.8: Acurácia dos classificadores na quarta base por esquema, com média e desvio padrão

se tem baixa performance e baixa complexidade no caso do *One-vs-Rest* e alta performance e alta complexidade no *One-vs-One*, árvore apresenta menor complexidade que o *One-vs-Rest* e melhor performance que o mesmo.

A seguir, é dado o prosseguimento de análise das performances obtidas em bases de dados reais.

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.41	0.04	**0.82**	0.05	0.60	0.04
LR	0.40	0.04	**0.77**	0.06	0.59	0.03
AD	0.73	0.05	**0.96**	0.04	0.61	0.03
MLP	**0.41**	0.13	0.27	0.03	0.39	0.16
KNN	0.79	0.05	**0.99**	0.02	0.59	0.03

Tabela 4.9: Kappa dos classificadores na quarta base por esquema, com média e desvio padrão

$$\mu = [0.0, 10.0], \sum = \begin{bmatrix} 12.0 & 0.0 \\ 0.0 & 12.0 \end{bmatrix}$$

(a) Parâmetros da classe 0 na quarta base

$$\mu = [13.0, 20.0], \sum = \begin{bmatrix} 10.0 & 0.0 \\ 0.0 & 10.0 \end{bmatrix}$$

(b) Parâmetros da classe 1 na quarta base

$$\mu = [29.0, 18.0], \sum = \begin{bmatrix} 15.0 & 0.0 \\ 0.0 & 15.5 \end{bmatrix}$$

(c) Parâmetros da classe 2 na quarta base

$$\mu = [10.0, 5.0], \sum = \begin{bmatrix} 12.0 & 0.0 \\ 0.0 & 12.0 \end{bmatrix}$$

(d) Parâmetros da classe 3 na quarta base

$$\mu = [43.0, 20.0], \sum = \begin{bmatrix} 10.0 & 0.0 \\ 0.0 & 10.0 \end{bmatrix}$$

(e) Parâmetros da classe 4 na quarta base

$$\mu = [49.0, 18.0], \sum = \begin{bmatrix} 15.0 & 0.0 \\ 0.0 & 15.5 \end{bmatrix}$$

(f) Parâmetros da classe 5 na quarta base

Figura 4.8: Parâmetros das distribuições Gaussianas na quarta base

4.2 Dados Reais

Como dito anteriormente, foram exploradas análises acerca de dez bases reais. Todas as bases foram coletadas do site da UCI(*University of California, Irvine*), site famoso por conter diversas bases de dados. A tabela 4.10 apresenta as características de cada base explorada neste trabalho. Foram escolhidas bases com número variado de classes, de balanceamento, de indivíduos e de atributos. A seguir, serão mostradas todas as tabelas, referentes às métricas Kappa e Acurácia de cada base.

Algo que pode-se perceber logo de início, ao analisarmos as tabelas, é a clara queda de

	Número de classes	Número de indivíduos	Número de atributos	Balanceamento
Absenteeism	4	740	20	Não
Forest Types	4	198	27	Não
Glass	7	214	10	Não
Image Segmentation	7	210	19	Sim
Image Segmentation(test)	7	2100	19	Sim
Iris	3	150	4	Sim
Wholesale Customers	3	440	7	Não
Wine	3	178	14	Sim
Wine Quality	6	1599	12	Não
Zoo	7	101	17	Não

Tabela 4.10: Características das bases de dados reais

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.85	0.02	**0.98**	0.02	0.82	0.01
LR	0.95	0.02	**0.99**	0.01	0.82	0.01
AD	**1.0**	0.0	**1.0**	0.0	0.82	0.0
MLP	**0.86**	0.04	0.83	0.01	0.82	0.01
KNN	0.98	0.01	**1.0**	0.0	0.82	0.01

Tabela 4.11: Acurácia dos classificadores na base Absenteeism por esquema, com média e desvio padrão

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.24	0.12	**0.94**	0.1	0.0	0.0
LR	0.84	0.07	**0.98**	0.03	0.0	0.0
AD	**1.0**	0.0	**1.0**	0.0	0.0	0.0
MLP	**0.22**	0.3	0.0	0.0	0.0	0.0
KNN	0.94	0.05	**0.99**	0.01	0.0	0.0

Tabela 4.12: Kappa dos classificadores na base Absenteeism por esquema, com média e desvio padrão

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.27	0.02	**0.92**	0.08	0.29	0.03
LR	0.94	0.05	**1.0**	0.0	0.48	0.08
AD	0.93	0.06	**0.99**	0.01	0.47	0.08
MLP	0.45	0.17	**0.47**	0.06	0.37	0.11
KNN	0.96	0.04	**0.99**	0.01	0.49	0.09

Tabela 4.13: Acurácia dos classificadores na base Forest Types por esquema, com média e desvio padrão

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.0	0.0	**0.90**	0.11	-0.01	0.03
LR	0.92	0.07	**1.0**	0.0	0.28	0.12
AD	0.91	0.07	**0.99**	0.02	0.27	0.12
MLP	0.25	0.23	**0.31**	0.08	0.13	0.14
KNN	0.94	0.06	**0.99**	0.02	0.30	0.14

Tabela 4.14: Kappa dos classificadores na base Forest Types por esquema, com média e desvio padrão

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.61	0.08	**0.75**	0.08	0.50	0.06
LR	0.47	0.07	**0.65**	0.09	0.51	0.07
AD	0.65	0.10	**0.96**	0.05	0.48	0.07
MLP	**0.34**	0.04	0.13	0.02	0.32	0.09
KNN	0.74	0.08	**0.97**	0.04	0.48	0.06

Tabela 4.15: Acurácia dos classificadores na base Glass por esquema, com média e desvio padrão

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.44	0.12	**0.65**	0.11	0.28	0.09
LR	0.23	0.10	**0.51**	0.13	0.30	0.10
AD	0.51	0.14	**0.95**	0.06	0.27	0.10
MLP	0.0	0.05	-0.03	**0.03**	0.02	0.08
KNN	0.64	0.11	**0.96**	0.06	0.26	0.09

Tabela 4.16: Kappa dos classificadores na base Glass por esquema, com média e desvio padrão

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.23	0.02	**0.96**	0.04	0.22	0.04
LR	0.82	0.02	**0.97**	0.01	0.26	0.06
AD	0.94	0.01	**1.0**	0.0	0.26	0.06
MLP	**0.88**	0.03	0.86	0.02	0.26	0.06
KNN	0.96	0.01	**1.0**	0.0	0.27	0.05

Tabela 4.17: Acurácia dos classificadores na base Image Segmentation por esquema, com média e desvio padrão

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.1	0.02	**0.95**	0.05	0.09	0.05
LR	0.79	0.02	**0.96**	0.01	0.13	0.07
AD	0.93	0.02	**1.0**	0.0	0.14	0.07
MLP	**0.86**	0.03	0.84	0.03	0.14	0.06
KNN	0.95	0.01	**0.99**	0.0	0.15	0.06

Tabela 4.18: Kappa dos classificadores na base Image Segmentation por esquema, com média e desvio padrão

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.23	0.02	**0.96**	0.04	0.22	0.04
LR	0.82	0.02	**0.97**	0.01	0.26	0.06
AD	0.94	0.01	**1.0**	0.0	0.26	0.06
MLP	**0.88**	0.03	0.86	0.02	0.26	0.06
KNN	0.96	0.01	**1.0**	0.0	0.27	0.05

Tabela 4.19: Acurácia dos classificadores na base Image Segmentation(test) por esquema, com média e desvio padrão

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.10	0.02	**0.95**	0.04	0.09	0.05
LR	0.79	0.02	**0.96**	0.01	0.13	0.07
AD	0.93	0.02	**1.0**	0.0	0.14	0.07
MLP	**0.86**	0.03	0.84	0.03	0.14	0.06
KNN	0.95	0.01	**1.0**	0.0	0.15	0.07

Tabela 4.20: Kappa dos classificadores na base Image Segmentation(test) por esquema, com média e desvio padrão

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.97	0.08	**0.98**	0.03	0.33	0.0
LR	0.69	0.08	**0.98**	0.03	0.33	0.0
AD	0.94	0.05	**0.99**	0.02	0.33	0.0
MLP	0.80	0.12	**0.98**	0.03	0.33	0.0
KNN	0.96	0.04	**1.0**	0.0	0.33	0.0

Tabela 4.21: Acurácia dos classificadores na base Iris por esquema, com média e desvio padrão

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.96	0.06	**0.97**	0.05	0.0	0.0
LR	0.53	0.11	**0.97**	0.04	0.0	0.0
AD	0.92	0.08	**0.99**	0.05	0.0	0.0
MLP	0.71	0.17	**0.97**	0.0	0.0	0.0
KNN	0.94	0.07	**1.0**	0.03	0.0	0.0

Tabela 4.22: Kappa dos classificadores na base Iris por esquema, com média e desvio padrão

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.72	0.01	**0.97**	0.03	0.72	0.01
LR	0.71	0.02	**0.72**	0.02	**0.72**	0.01
AD	0.45	0.07	**0.95**	0.06	0.72	0.01
MLP	0.47	0.12	**0.62**	0.05	0.72	0.01
KNN	0.56	0.06	**0.96**	0.04	0.72	0.01

Tabela 4.23: Acurácia dos classificadores na base Wholesale Customers por esquema, com média e desvio padrão

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.0	0.0	**0.92**	0.11	0.0	0.0
LR	0.01	0.05	**0.03**	0.06	0.0	0.0
AD	-0.01	0.10	**0.88**	0.14	0.0	0.0
MLP	0.02	0.10	**0.01**	0.11	0.0	0.01
KNN	0.0	0.11	**0.92**	0.09	0.0	0.0

Tabela 4.24: Kappa dos classificadores na base Wholesale Customers por esquema, com média e desvio padrão

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.36	0.04	**0.94**	0.07	0.48	0.06
LR	0.93	0.06	**0.98**	0.03	0.67	0.07
AD	0.90	0.07	**1.0**	0.0	0.66	0.08
MLP	0.37	0.09	**0.39**	0.09	**0.39**	0.11
KNN	0.77	0.09	**0.98**	0.03	0.66	0.08

Tabela 4.25: Acurácia dos classificadores na base Wine por esquema, com média e desvio padrão

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.05	0.06	**0.90**	0.12	0.15	0.10
LR	0.90	0.08	**0.96**	0.05	0.48	0.11
AD	0.85	0.10	**1.0**	0.0	0.46	0.12
MLP	0.04	0.13	**0.10**	0.13	0.05	0.16
KNN	0.64	0.14	**0.97**	0.05	0.47	0.13

Tabela 4.26: Kappa dos classificadores na base Wine por esquema, com média e desvio padrão

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.42	0.03	**0.72**	0.04	0.48	0.02
LR	0.38	0.03	**0.61**	0.03	0.49	0.03
AD	0.57	0.04	**0.97**	0.03	0.48	0.03
MLP	0.28	0.07	**0.56**	0.03	0.49	0.03
KNN	0.60	0.04	**0.96**	0.04	0.48	0.03

Tabela 4.27: Acurácia dos classificadores na base Wine Quality por esquema, com média e desvio padrão

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.21	0.04	**0.55**	0.06	0.13	0.04
LR	0.17	0.03	**0.35**	0.05	0.13	0.05
AD	0.38	0.05	**0.95**	0.05	0.13	0.05
MLP	0.11	0.04	**0.25**	0.05	0.14	0.05
KNN	0.37	0.06	**0.93**	0.06	0.12	0.05

Tabela 4.28: Kappa dos classificadores na base Wine Quality por esquema, com média e desvio padrão

Acurácia	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.86	0.08	**0.96**	0.05	0.83	0.09
LR	0.87	0.09	**0.98**	0.04	0.82	0.10
AD	0.91	0.07	**1.0**	0.0	0.83	0.10
MLP	0.90	0.08	**1.0**	0.0	0.82	0.10
KNN	0.98	0.04	**1.0**	0.0	0.83	0.10

Tabela 4.29: Acurácia dos classificadores na base Zoo por esquema, com média e desvio padrão

Kappa	OvR		OvO		ADB	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
SVM	0.80	0.12	**0.95**	0.06	0.77	0.12
LR	0.82	0.12	**0.97**	0.05	0.77	0.13
AD	0.89	0.10	**1.0**	0.0	0.77	0.13
MLP	0.85	0.12	**1.0**	0.0	0.76	0.13
KNN	0.97	0.05	**1.0**	0.0	0.78	0.13

Tabela 4.30: Kappa dos classificadores na base Zoo por esquema, com média e desvio padrão

desempenho do método proposto em relação aos resultados com bases simuladas.

Casos como a base *iris* ou a base *image segmentation*, tabelas 4.17, 4.18, 4.21 e 4.22, evidenciam a queda, sendo essas duas as mais visíveis. Ao observarmos a *Iris*, é visível que trata-se de predições aleatórias, uma vez que, em todos os casos, a acurácia foi de 33% e o Kappa foi 0.

Algo muito importante que pôde ser observado a partir da métrica Kappa é a aleatoriedade de algumas predições que muitas vezes parecem não ser completamente aleatórias. Casos como o da base *iris* deixam isso bastante claro.

O *One-vs-Rest* apresenta grande variabilidade em sua performance, algo já previsto nos experimentos prévios. Já a Árvore de Decomposição Binária demonstra muito maior variação que as previamente calculadas em dados simulados.

O *One-vs-One* demonstra imensa robustez. Isso se deve a fácil adaptação do esquema para diversos casos. O fato do mesmo não gerar desbalanceamento em bases balanceadas o difere bastante do *One-vs-Rest*, o que fica evidente em bases como *Image Segmentation, Image Segmentation(test) e Iris*, por exemplo. A robustez do método se reflete na transposição de um problema de múltiplas classes para múltiplos problemas binários.

O baixo desempenho e alta variação em todas as bases provaram a ineficácia do modelo com relação à dados reais, mesmo com o ganho em custo computacional advindo do mesmo. A provável razão por trás da queda de desempenho pode ter relação com o fato das classes em cada base não se comportarem como uma distribuição normal multivariada, como é o caso das bases simuladas, e por isso as tornam não ideais para o modelo. Outra possível causa para o baixo desempenho é o número alto de variáveis em cada base, e consequente falta de normalização nas variáveis, fazendo com que as mesmas tenham variáveis em escalas não congruentes.

A soberania em desempenho do *One-vs-One* se mantém em todos os casos exceto nas bases *wine*, *wholesale customers*, *image segmentation*, *image segmentation(test)*, *glass* e *absenteeism*, todas em relação aos perceptrons multicamadas. Isso aconteceu pelo fato deste estudo não tratar hiperparâmetros da rede neural. Por isso, o classificador ficou sujeito a *underfitting* nesses casos.

CAPÍTULO 5

Conclusão

Este trabalho apresentou um novo esquema para a binarização de problemas de múltiplas classes, a Árvore de Decomposição Binária. A árvore de decomposição binária difere-se de todos os outros esquemas vistos, pelo fato de utilizar probabilidades aninhadas, similarmente a uma rede Bayesiana, para predizer a classe à qual um exemplo pertence. A árvore de decomposição binária possui referências à outros esquemas de binarização usando árvores binárias, como a Árvore de SVM proposta por Cheong, Oh & Lee. Neste trabalho, assim como no referente à árvore de SVM, foi demonstrado que o uso de aprendizado não supervisionado pode ser útil para problemas de classificação, especialmente os de múltiplas classes.

Foram realizados experimentos com dados sintéticos e reais em comparação com os métodos mais utilizados: *One-vs-One* e *One-vs-Rest*. As bases simuladas foram geradas por meio de distribuições normais bivariadas, enquanto as bases reais foram extraídas do repositório da UCI. Tanto nas bases sintéticas quanto nas reais, foi utilizado o método de validação cruzada estratificada repetida, com 10 partes e 10 repetições, gerando 100 resultados. Destes, foram extraídos a média e o desvio padrão das métricas acurácia e Kappa.

Os resultados obtidos a partir desses experimentos levam à conclusão de que o método tem desempenho frágil quando aplicado à bases reais. Apesar de algumas vezes apresentar desempenhos que poderiam ser considerados relativamente bons, o uso da métrica Kappa mostrou que parte desses resultados são na verdade advindo de predições que se aproximam de predições aleatórias por parte do classificador. A ineficácia do método proposto nas bases reais pode ser explicada pelo fato de ser mais fácil treinar o agrupamento aglomerativo para modelar os dados se os mesmos forem gerados a partir de uma distribuição Gaussiana, que é o caso das bases sintéticas. Outro possível fator a ser considerado é o fato de todas as bases sintéticas terem apenas duas dimensões, enquanto todas as bases reais tem mais de quatro. Vista a maior variabilidade de escala das dimensões em bases reais e a pouca normalização pode afetar no cálculo das distâncias, que são base para a computação do agrupamento hierárquico aglomerativo.

Como trabalhos futuros, pretende-se avaliar outras formas de usar o agrupamento hierárquico na esquematização de problemas de múltiplas classes, tal qual testar outras fórmulas de recorrência. Pretende-se, também, investigar o uso de probabilidades na predição de arquiteturas já existentes que usem árvores binárias ou outras estruturas hierárquicas, como o grafo direcionado acíclico, realizando testes em comparação aos métodos do estado da arte previamente mencionados.

Referências Bibliográficas

[1] ALY, M. Survey on multiclass classification methods. *Neural Netw 19* (2005), 1–9.

[2] BERKHIN, P. *A Survey of Clustering Data Mining Techniques*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 25–71.

[3] BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[4] CHAWLA, N. V. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*. Springer, 2009, pp. 875–886.

[5] CHEONG, S., OH, S. H., AND LEE, S.-Y. Support vector machines with binary tree architecture for multi-class classification. *Neural Information Processing-Letters and Reviews 2*, 3 (2004), 47–51.

[6] CUNNINGHAM, P., AND DELANY, S. J. k-nearest neighbour classifiers. *Multiple Classifier Systems 34*, 8 (2007), 1–17.

[7] DIBB, S., AND WENSLEY, R. Segmentation analysis for industrial markets: Problems of integrating customer requirements into operations strategy. *European journal of marketing 36*, 1/2 (2002), 231–251.

[8] DUDA, R. O., HART, P. E., AND STORK, D. G. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, New York, NY, USA, 2000.

[9] DUDANI, S. A. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 4 (1976), 325–327.

[10] EVERITT, B., LANDAU, S., AND LEESE, M. *Cluster Analysis*, 4th ed. Arnold, 2001.

[11] FEI, B., AND LIU, J. Binary tree of svm: a new fast multiclass training and classification algorithm. *IEEE Transactions on Neural Networks 17*, 3 (May 2006), 696–704.

[12] FITZGERALD, W. Machine learning for fraud detection, Sept. 21 2017. US Patent App. 15/070,138.

[13] GALAR, M., FERNÁNDEZ, A., BARRENECHEA, E., BUSTINCE, H., AND HERRERA, F. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition 44*, 8 (2011), 1761–1776.

[14] HAN, J., AND MORAGA, C. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks* (1995), Springer, pp. 195–201.

[15] HEARST, M. A., DUMAIS, S. T., OSUNA, E., PLATT, J., AND SCHOLKOPF, B. Support vector machines. *IEEE Intelligent Systems and their applications 13*, 4 (1998), 18–28.

[16] KRISHNAPURAM, R., AND KELLER, J. M. A possibilistic approach to clustering. *IEEE transactions on fuzzy systems 1*, 2 (1993), 98–110.

[17] MARLER, R. T., AND ARORA, J. S. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization 26*, 6 (2004), 369–395.

[18] MCKINNEY, W. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (2010), S. van der Walt and J. Millman, Eds., pp. 51 – 56.

[19] MEIER, L., VAN DE GEER, S., AND BÜHLMANN, P. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 70*, 1 (2008), 53–71.

[20] MITCHELL, T. M. *Machine Learning*, 1 ed. McGraw-Hill, Inc., New York, NY, USA, 1997.

[21] MØLLER, M. F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks 6*, 4 (1993), 525–533.

[22] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[23] PLATT, J. Sequential minimal optimization: A fast algorithm for training support vector machines.

[24] PROVOST, F., AND FAWCETT, T. Data science and its relationship to big data and data-driven decision making. *Big data 1*, 1 (2013), 51–59.

[25] QUINLAN, J. R. Induction of decision trees. *Machine learning 1*, 1 (1986), 81–106.

[26] QUINLAN, J. R., ET AL. Bagging, boosting, and c4. 5. In *AAAI/IAAI, Vol. 1* (1996), pp. 725–730.

[27] ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review 65*, 6 (1958), 386.

[28] ROSSUM, G. Python reference manual. Tech. rep., Amsterdam, The Netherlands, The Netherlands, 1995.

[29] WERBOS, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE 78*, 10 (1990), 1550–1560.

[30] WU, T.-F., LIN, C.-J., AND WENG, R. C. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research 5*, Aug (2004), 975–1005.

[31] XU, R., AND WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks 16*, 3 (May 2005), 645–678.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).