UNIVESIDADE ESTADUAL PAULISTA

"Júlio de Mesquita Filho"

Pós-Graduação em Ciência da Computação

Luís Augusto Martins Pereira

Explorando Abordagens de Múltiplos Rótulos por Floresta de Caminhos Ótimos

UNESP

2014

Pereira, Luís Augusto Martins.
Explorando abordagens de múltiplos rótulos por floresta de caminhos ótimos / Luís Augusto Martins Pereira. -- São José do Rio Preto, 2014
64 f. : il., gráfs., tabs.

Orientador: João Paulo Papa
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação - Matemática. 2. Processamento de imagens - Técnicas digitais. 3. Reconhecimento de padrões. 4. Floresta de caminhos ótimos. 5. Árvores (Teoria dos grafos) I. Papa, João Paulo. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. III. Título.
CDU – 518.72:76

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Explorando Abordades de Múltiplos Rótulos por Floresta de Caminhos Ótimos

Luís Augusto Martins Pereira[1]

Abril de 2014

Explorando Abordagens de Múltiplos Rótulos por Floresta de Caminhos Ótimos

Banca Examinadora:

- Prof. Dr. João Paulo Papa (Orientador)
- Prof. Dr. José Remo Ferreira Brega (DCo/FC/UNESP)
- Prof. Dr. Estevam Rafael Hruschka Júnior (DC/UFSCar)

[1]Este Trabalho recebeu auxílio financeiro da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) através do processo de número 2011/14094-1. As opiniões, hipóteses, conclusões e recomendações expressa neste material são de responsabilidade do(s) autor(es) e não necessariamente refletem a visão da FAPESP.

Agradecimentos

A Deus pela graça, força e misericordia.

Ao meu orientador João Paulo Papa pela mão estendida, pela amizade e por ter acreditado no meu trabalho.

A UNESP pela infraestruta para o desenvolvimento do projeto, e a Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo auxílio financeiro.

A toda minha família, especialmente minha mãe Maria por tudo que fez e faz por mim, minhas irmãs Luzia, Dês, Nana, Laura e Lene por terem me incentivado e me ajudado em tudo, meus cunhados Jô, Bico, Walter e Celso, meus sobrinhos Léo, Otávio, João e Vicente pela alegria.

A minha namorada Simone pelo amor, amizade e incentivo.

A todos os meus amigos do grupo Recogna Clayton, Douglas, Mizobe, Luís Sugi, Rafael, Silas, Adriana e Peixinho.

A todos aqueles que diretamente ou indiretamente me ajudaram.

Resumo

Em problemas convencionais de reconhecimento de padrões, dado um conjunto de classes, cada instância do problema é associada a uma e somente uma classe. No entanto, alguns problemas reais de classificação apresentam instâncias que podem ser associadas a mais de uma classe simultaneamente, esses problemas são denotados como classificação com múltiplos rótulos. Entre problemas dessa natureza, podemos destacar categorização de filmes e músicas, classificação de documentos, análise funcional de genes etc. Contudo, os problemas de classificação com múltiplos rótulos não são diretamente tratáveis por técnicas convencionais, o que justifica o interesse da comunidade de reconhecimento de padrões nesses tipos de problemas. Embora muitos métodos tenham sido propostos na literatura, há ainda muito a ser explorado, principalmente no uso de novos algoritmos convencionais de aprendizado de máquinas adaptados ou não aos problemas com múltiplos rótulos. O classificador supervisionado Floresta de Caminhos Ótimos (*Optimum-Path Forest* - OPF) é um algoritmo determinístico aplicado à problemas convencionais de classificação, no entanto, ainda não foi investigado em problemas com múltiplos rótulos. Nesse contexto, investigamos neste trabalho a aplicação de classificadores baseados em OPF em problemas de múltiplos rótulos. Analisamos duas versões do classificador OPF: (i) a tradicional baseada em grafo completo e (ii) a versão baseada no grafo k-vizinhos mais próximos (OPFkNN). Para manipulação das bases com múltiplos rótulos, utilizamos dois métodos de transformação de problemas, o *Binary Relevance* e *Label Powerset*. Propusemos também algumas modificações nas fases de treinamento e classificação do OPFkNN com o objetivo de melhor os resultados desse classificador combinado a métodos de transformação de problemas. Os experimentos realizados em sete bases de dados públicas mostraram que as modificações no OPFkNN melhoram os resultados. Comparação com os classificadores J48, Naïve Bayes e SVM foram realizados, e demostraram que os classificadores OPF, principalmente o OPFkNN, foram similares ao SVM e superiores aos outros dois classificadores.

***Palavras-chave**:* Reconhecimento de Padrãoe, Floresta de Caminhos Ótimos, Aprendizado com múltiplos rótulos, Métodos de transfomação de problemas.

Abstract

In conventional problems of pattern recognition, given a set of classes, each instance of the problem is associated with one and only one class. However, some real classification problems have instances that can be associated with more than one class at the same time, these problems are denoted as classification with multilabel. Among such problems, we highlight movies and music categorization, document classification, functional gene analysis etc. Nevertheless, the classification problems with multilabel are not directly treatable by conventional techniques, which explains the interest of pattern recognition community in these types of problems. Although many methods have been proposed in the literature, there is still much to be explored, especially in the use of novel conventional machine learning algorithms adapted or not to problems with multlabels. The Optimum-Path Forest (OPF) classifier is a supervised and deterministic algorithm applied to conventional classification problems, however, it has been not investigated in problems with multilabel. In this context, we investigated in this work the application of OPF-based classifiers on multilabel problems. We analyzed two versions of OPF-based classifiers: (i) the traditional one based on complete graph and (ii) the one based on k-nearest neighbors graph (OPFkNN). For manipulation of multilabel datasets, we used two transformation methods, the Binary Relevance and Label Powerset. We also proposed some changes in the training and classification phases of OPFkNN aiming to achieve better results when combined it with transformation methods. Experiments performed in seven public datasets showed that changes in OPFkNN improve outcomes. Comparison with the J48 classifier, Naïve Bayes and SVM were performed, and they showed that the OPF classifiers, especially OPFkNN, performed similarly to SVM and better than the other two classifiers.

Keywords: Pattern Recognition, Optimum-Path Forest, Learning with multilabel, Problem transformation methods.

Sumário

Lista de Tabelas

Lista de Figuras

1 Introdução

Em problemas convencionais de reconhecimento de padrões, dado um conjunto de c classes, cada instância do problema é associada a uma e somente uma classe. Se $|c| = 2$, então o problema é dito binário, enquanto $|c| > 2$ denota um problema com múltiplas classes. Esses tipos de problemas podem ser tratados por diferentes algoritmos de aprendizado de máquinas, tais como: *Artificial Neural Networks (ANN)* [1], *k-Nearest Neighbors (k-NN)* [2], *Support Vector Machines (SVM)* [3], dentre outros. Contudo, alguns problemas reais de classificação apresentam instâncias que podem ser associadas a mais de uma classe (ou rótulo) simultaneamente. Um filme, por exemplo, pode ser categorizado como comédia e romance, ou seja, pode pertencer aos dois gêneros ao mesmo tempo. Problemas como esse são denotados como classificação com múltiplos rótulos. Muito embora, problemas convencionais de classificação possam também ser vistos como um caso especial de múltiplos rótulos, onde somente um rótulo é o correto [4].

O interesse da comunidade de aprendizado de máquinas por problemas com múltiplos rótulos deu-se, principalmente, no estudo de categorização de textos [5], onde um documento pode ser associado a vários tópicos, tais como história, religião e artes. No entanto, há outros domínios onde a classificação com múltiplos rótulos pode ser aplicada. Em medicina, por exemplo, um sistema de auxílio ao diagnóstico médico pode ser modelado como uma tarefa de múltiplos rótulos, dado que um paciente pode estar acometido por mais de uma doença. Em análise funcional de genes, cada um deles pode pertencer a múltiplas funções genômicas [6, 7].

Uma outra área em ascensão é a de categorização de imagens em um grande número de classes. Sistemas de recuperação de imagens baseados em conteúdo tem enfrentado

tais situações, onde uma imagem pode pertencer a mais de uma classe, pois depende da rotulação do usuário, a qual é bastante subjetiva. Bases de dados com cenas naturais geralmente são as mais utilizadas para a validação de tais sistemas, dado que paisagens são comumente classificadas como pertencentes a mais de uma classe. Tal tarefa requer uma classificação semântica de cenas, dado que uma fotografia pode pertencer à categoria de praia e pôr-do-sol ao mesmo tempo, por exemplo [8].

Atualmente, novas aplicações que requerem métodos de classificação com suporte a múltiplos rótulos tem surgido. Aplicações que são dependentes de rotulações por uma vasta gama de usuários como, por exemplo, categorização musical, estão entre as aplicações com maior potencial. Sistemas que podem recomendar uma determinada música a uma pessoa de acordo com o seu estado emocional e/ou perfil são muito úteis no sentido de prover uma melhor qualidade de vida e, possivelmente, atrair possíveis consumidores para determinados produtos [9].

Como pode ser observado, os problemas de classificação com múltiplos rótulos demonstram que técnicas convencionais não são diretamente aplicáveis a eles. Dessa forma, várias abordagens têm sido propostas nos últimos anos. Segundo Tsoumakas e Katakis [10], essas abordagens podem ser sumarizadas em dois grupos principais: (a) algoritmos de adaptação de métodos e (b) métodos de transformação de problemas.

Os algoritmos de adaptação de métodos propõem modificações em técnicas convencionais de classificação para que essas possam manipular os problemas com múltiplos rótulos diretamente. *Neural networks* [11], *lazy learning algorithms* [12], *decision trees* [13] e *boosting algorithms* [5] são exemplos de técnicas que foram adaptadas ao contexto de múltiplos rótulos. Por outro lado, os métodos de transformação de problemas transformam o aprendizado com múltiplos rótulos em um ou mais problemas convencionais. Assim, um classificador convencional pode ser aplicado para "aprender" o comportamento dos dados e, durante a fase de classificação, o rótulo predito a uma instância é transformado em um conjunto de múltiplos rótulos novamente. Um método de transformação de problemas muito utilizado é *Binary Relevance* [10]. Basicamente, esse método decompõe o problema em um número de subproblemas binários relativo ao número de rótulos que o problema inicial apresenta. Então, aplicam-se classificadores binários convencionais a

cada subproblema.

Embora muitos métodos tenham sido propostos na literatura, por ser uma área de estudo relativamente recente, há ainda muito a ser explorado, principalmente no uso de novos algoritmos convencionais de aprendizado de máquinas adaptados ou não aos problemas com múltiplos rótulos.

O classificador supervisionado Floresta de Caminhos Ótimos (*Optimum-Path Forest* - OPF) [14] é um algoritmo determinístico aplicado à problemas convencionais de classificação. Seu processo de classificação é desenvolvido em duas fases: aprendizado dos dados (treinamento) e classificação (teste). Durante a fase de treinamento o algoritmo utiliza um grafo induzido (grafo completo ou k-vizinhos mais próximos) para a escolha de nós que melhor representem as classes do problema. A partir desses nós, inicia-se um processo de conquista baseado em uma função de conectividade. Ao final desse fase, o espaço de atributos está particionado em árvores de caminhos ótimos. Na fase de teste, uma instância a ser classificada é conectada aos nós do modelo de treinamento e recebe a classe da árvore que lhe oferecer o melhor custo.

O OPF tem demonstrado similaridade com relação a taxa de acerto do classificador SVM (estado-da-arte em classificação), porém muito mais rápido para o aprendizado do comportamento dos dados de treinamento. Esse classificador tem sido aplicado com sucesso à diferentes problemas, tais como detecção de intrusão em redes de computadores [15], classificação de plantas daninhas aquáticas [16], detecção de falhas em sistemas elétricos [17], problemas de sensoriamento remoto [18, 19, 20], dentre outros. No entanto, o classificador OPF ainda não foi aplicado à problemas com múltiplos rótulos.

Nesse contexto, investigamos neste trabalho a aplicação do classificador OPF em problemas de múltiplos rótulos. O objetivo foi verificar como o classificador se comporta frente a esses problemas, verificando pontos fortes e fracos do classificador aliado à métodos de transformação de problemas. Entre as contribuições, pode-se destacar que este trabalho foi o primeiro a abordar técnicas de aprendizado com suporte a múltiplos rótulos utilizando o classificador OPF.

O presente trabalho está organizado da seguinte forma. O Capítulo 2 apresenta uma

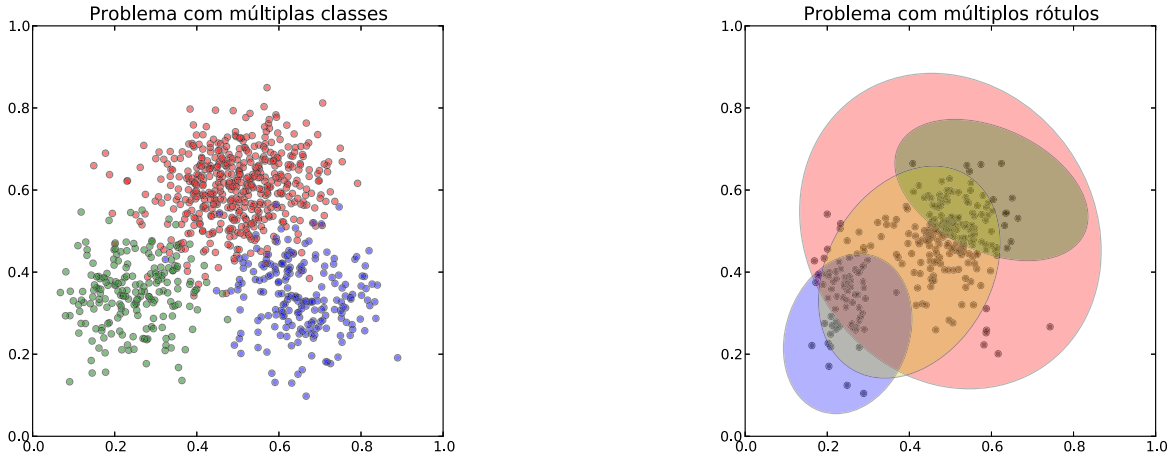
definição formal sobre o aprendizado com múltiplos rótulos, bem como um conjunto de métricas de avaliação para o problema. Algumas técnicas de classificação com suporte a múltiplos rótulos estudadas neste trabalho são apresentadas no Capítulo 3. O Capítulo 4 aborda a teoria sobre o classificador OPF supervisionado utilizando indução por grafo completo e k-nn. Experimentos utilizando o classificador OPF associado a métodos de tranformação de problema são discutidos no Capítulo 5, enquanto as conclusões são apresentadas no Capítulo 6. Finalmente, o Capítulo 7 apresenta os artigos desenvolvidos durante o projeto.

2 Aprendizado com Múltiplos Rótulos

Neste capítulo, apresentamos o aprendizado com múltiplos rótulos. Inicialmente, mostramos a diferença entre o aprendizado com múltiplos rótulos e múltiplas classes. Em seguida, apresentamos uma fundamentação teórica do aprendizado com múltiplos rótulos. Finalmente, introduzimos algumas métricas utilizadas para medir o desempenho de métodos com suporte a múltiplos, e métricas utilizadas para medir a distribuição de rótulos nas bases de dados multirrotuladas.

2.1 Múltiplos rótulos vs. múltiplas classes

Problemas que apresentam um conjunto de instâncias onde cada uma dessas é associada a uma única classe é denominado classificação com múltiplas classes. Tais problemas apresentam, por definição, classes mutuamente exclusivas [8]. Essa definição não deve ser confundida com problemas de múltiplos rótulos, onde cada instância pode estar associada a um conjunto de rótulos. Os rótulos pertencentes a tal conjunto são chamados de relevantes, enquanto os que não pertencem são chamados de irrelevantes [21]. A Figura 2.1(a) apresenta um gráfico de dispersão com instâncias pertencentes exclusivamente a três classes distintas, o que caracteriza um problema com múltiplas classes. A Figura 2.1(b) também apresenta um gráfico dispersão e quatro elipses, cada uma delas representando um rótulo. Pode-se observar que, nesse caso, instâncias que pertencem a regiões onde há intersecção de elipses são instâncias que compartilham mais do que um rótulo simultaneamente.



(a) (b)

Figura 2.1: (a) representação de um problema com classes mutuamente exclusivas. (b) problema onde instâncias pertencentes à regiões de intersecção das elipses são múltirrotuladas

2.2 Fundamentação teórica

Nesta seção, apresentamos uma definição formal para o problema de aprendizado com múltiplos rótulos. A notação utilizada nessa seção será empregada em definições de capítulos subsequentes.

Seja $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ um conjunto com m instâncias, e $m = |\mathcal{X}|$. Seja $\mathcal{L} = \{r_1, r_2, \ldots r_n\}$ um conjunto com n rótulos, para $n = |\mathcal{L}|$, e $\mathcal{Y}_i \subseteq \mathcal{L}$, com $i = 1, 2, \ldots, m$, um subconjunto de rótulos $r \in \mathcal{L}$. Dado um conjunto de instâncias com múltiplos rótulos $\mathcal{T} = \{(x_1, \mathcal{Y}_1), (x_2, \mathcal{Y}_2), \ldots, (x_m, \mathcal{Y}_m)\}$, um classificador $h : x_i \rightarrow \mathcal{Y}'_i$, com suporte a múltiplos rótulos, tem como objetivo predizer um conjunto de rótulos a cada instância do conjunto de avaliação que maximize um critério q. Entretanto, em muitos casos, o aprendizado produz uma função de ranqueamento $f' : \mathcal{X} \times \mathcal{L} \rightarrow \mathbb{R}$ tal que, para uma dada instância $x \in \mathcal{X}$, o conjunto de rótulos em $\mathcal{L}$ deve ser ordenado de acordo com $f'(x, \cdot)$, onde $\mathbb{R}$ denota a função de ranqueamento utilizada na tarefa. Assim, um rótulo r_1 é considerado melhor ranqueado que um outro rótulo r_2 se $f'(x, r_1) > f'(x, r_2)$. Desta forma, um classificador com suporte a múltiplos rótulos h' pode ser derivado da função

de ranqueamento $f'(\cdot,\cdot)$:

$$h'(x) = \{r | f'(x,r) > t(x), r \in \mathcal{L}\}, \tag{2.1}$$

onde $t(x)$ é uma função de limiarização.

2.3 Métricas de Avaliação

As métricas de avaliação aplicadas a problemas com múltiplos rótulos são exclusivas ou adaptadas das utilizadas em tarefas tradicionais de classificação. Geralmente, métricas diferentes e contrastantes são aplicadas ao mesmo problema, buscando contornar os graus de liberdade introduzidos pelos conjuntos de rótulos [21]. Abaixo, seguem algumas métricas utilizadas em tarefas com múltiplos rótulos (classificação e ranqueamento) apresentadas por Tsoumakas et al. [10] e Madjarov et al. [21].

2.3.1 Métricas para tarefas de classificação

As métricas apresentadas nesta seção são definidas para um classificador $h(\cdot)$ com suporte a múltiplos rótulos.

2.3.1.1 Accuracy

A métrica *Accuracy* para uma instância x_i é definida pelos coeficientes de similaridade de Jaccard entre o conjunto de rótulos $h(x_i)$ e $\mathcal{Y}_i$:

$$accuracy(h) = \frac{1}{m}\sum_{i=1}^{m}\frac{|h(x_i) \cap \mathcal{Y}_i|}{|h(x_i) \cup \mathcal{Y}_i|}, \tag{2.2}$$

2.3.1.2 Hamming Loss

A função de avaliação *Hamming Loss* é bastante utilizada e é definida como:

$$\Phi(h) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|\mathcal{L}|}\left|h(x_i)\Delta\mathcal{Y}_i\right|, \tag{2.3}$$

onde Δ denota a diferença simétrica entre dois conjuntos. Assim, quanto menor o valor de $\Phi(h)$, melhor a eficácia do classificador.

2.3.1.3 Precision

A métrica *Precision* é definida como segue:

$$precision(h) = \frac{1}{m}\sum_{i=1}^{m}\frac{|h(x_i)\cap\mathcal{Y}_i|}{|\mathcal{Y}_i|}, \tag{2.4}$$

2.3.1.4 Recall

A métrica *Recall* é definida como segue:

$$recall(h) = \frac{1}{m}\sum_{i=1}^{m}\frac{|h(x_i)\cap\mathcal{Y}_i|}{|h(x_i)|}, \tag{2.5}$$

2.3.1.5 Subset Accuracy

A métrica *Subset Accuracy* é definida como segue:

$$subset\_accuracy(h) = \frac{1}{m}\sum_{i=1}^{m}I(h(x_i) = \mathcal{Y}_i), \tag{2.6}$$

onde $I(true) = 1$ e $I(false) = 0$. Está métrica é bem rigorosa, dado que o conjunto de rótulos predito pelo classificador de ser exatamente igual ao verdadeiro conjunto de rótulos.

2.3.1.6 F1 score

F_1 *score* representa a media harmônica entre *precision* e *recall*, sendo definida como:

$$F_1(h) = \frac{1}{m}\sum_{i=1}^{m} 2 \times \frac{|h(x_i) \cap \mathcal{Y}'_i|}{|h(x_i)| + |\mathcal{Y}'_i|}, \tag{2.7}$$

2.3.2 Métricas para tarefas de ranqueamento

As métricas utilizadas para avaliação de ranqueamento são definidas para $f(\cdot, \cdot)$.

2.3.2.1 One_Error

Quanto menor o valor apresentado pela função *one_error* melhor a atuação do algoritmo. Sua definição é apresentada como segue:

$$\begin{aligned} H(x_i) &= \begin{cases} 0 & \text{se } \arg\max_{r \in \mathcal{L}} f(x_i, r) \in \mathcal{Y}_i, \\ 1 & \text{caso contrário,} \end{cases} \\ one\_error(f) &= \frac{1}{m}\sum_{i=1}^{m} H(x_i) \end{aligned} \tag{2.8}$$

2.3.2.2 Raking Loss

Raking Loss representa a fração média de pares que não são corretamente ordenados:

$$ranking\_loss(f) = \frac{1}{m}\sum_{i=1}^{m} \frac{|R(x_i)|}{|\mathcal{L}||\bar{\mathcal{Y}}|}, \tag{2.9}$$

onde $\bar{\mathcal{Y}}$ denota o conjunto complementar $\mathcal{Y} \subset \mathcal{L}$ e $R(x_i) = \{r_1, r_0 | f(x_i, r_0), (r_1, r_0) \in \mathcal{Y} \times \bar{\mathcal{Y}}\}$. Quanto o menor o valor de $ranking\_loss(f)$ melhor a atuação do algoritmo.

2.3.2.3 Average Precision

A função *Average Precision* avalia a fração média dos rótulos classificados acima de um determinado rótulo $r \in \mathcal{Y}_i$, que realmente estão em $\mathcal{Y}_i$.

$$avg\_precision(f) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|\mathcal{Y}|_i}\sum_{r\in\mathcal{Y}_i}\frac{|\mathcal{Y}|}{rank_f(x_i, r)} \tag{2.10}$$

Se $avg\_precision(f) = 1$ o algoritmo atingiu uma atuação máxima. Quanto o maior o valor de $avg\_precision(f)$ melhor a atuação do algoritmo.

2.4 Cardinalidade e densidade

Tsoumakas [10] propós duas métricas para avaliar o número de rótulos por instância em relação ao número total de rótulos. A Cardinalidade ($\mathcal{C}$), é a média do número de rótulos por instância, enquanto a Densidade ($\mathcal{D}$) é a média do número de rótulos por instância dividida pelo número de total de rótulos. As Equações 2.11 e 2.12 apresentam as duas métricas, respectivamente:

$$\mathcal{C} = \frac{1}{m}\sum_{i=1}^{m}|\mathcal{Y}_i| \tag{2.11}$$

$$\mathcal{D} = \frac{1}{m}\sum_{i=1}^{m}\frac{|\mathcal{Y}_i|}{|\mathcal{L}|} \tag{2.12}$$

A análise dessas duas métricas é importante, pois a distribuição dos rótulos pode influenciar o desempenho de diferentes abordagens multirrótulos.

3 Métodos de Classificação com Suporte a Múltiplos Rótulos

Métodos de classificação com suporte a múltiplos rótulos podem ser divididos em dois grupos principais: (i) métodos de transformação de problemas e (ii) algoritmos de adaptação de métodos [10]. Ao passo que as abordagens pertencentes ao primeiro grupo transformam os problemas com múltiplos rótulos em problemas tradicionais, isto é, com um único rótulo para cada instância, as técnicas de adaptação tentam modificar os algoritmos originais com o intuito de torná-los abordagens com suporte à múltiplos rótulos. As próximas seções tratam de explicar ambas abordagens.

3.1 Métodos de transformação de problemas

Os métodos pertencentes ao grupo de transformação de problemas são, geralmente, mais simples e fáceis de serem implementados, pois as transformações são, simplesmente, aplicadas à base de dados a ser classificada. Introduziremos esses métodos utilizando um exemplo apresentado por Tsoumakas e Katakis [10], onde tem-se um conjunto de dados composto por quatro instâncias que podem estar associadas 4 rótulos diferentes. A Tabela 3.1 ilustra esse exemplo.

Tabela 3.1: Base de dados utilizada para exemplificar os algoritmos de transformação.

Instâncias	Esportes	Religião	Ciência	Política
1	✓			✓
2			✓	✓
3	✓			
4		✓	✓	

3.1.1 Escolha de rótulos aleatórias e exclusão de múltiplos rótulos

Existem, basicamente, dois algoritmos simplistas, aqui chamados de A_1 e A_2, para resolver o problema apresentado na Tabela 3.1. O primeiro deles, A_1, seleciona aleatoriamente uma classe para as instâncias que possuem múltiplos rótulos, conforme ilustrado na Tabela 3.2. A segunda abordagem, A_2, exclui da base de dados todas as instâncias que contem múltiplos rótulos, restando apenas instâncias pertencentes a uma única classe, conforme mostra a Tabela 3.3. Como pode ser notado, ambas abordagens não são muito adequadas, dado que o algoritmo A_1 atribui uma classe a uma dada instância sem um critério objetivo, enquanto A_2 pode reduzir o número de instâncias da base consideravelmente.

Tabela 3.2: Base de dados da Tabela 3.1 transformada utilizando A_1.

Instâncias	Esportes	Religião	Ciência	Política
1				✓
2			✓	
3	✓			
4		✓		

Tabela 3.3: Base de dados da Tabela 3.1 transformada utilizando A_2.

Instâncias	Esportes	Religião	Ciência	Política
3	✓			

3.1.2 Label Powerset

Já um terceiro método, denominado pela comunidade como *Label Powerset* (LP), considera cada diferente conjunto de rótulos $\mathcal{Y} \in \mathcal{L}$ como sendo um único rótulo $P(\mathcal{Y})$, possibilitando a utilização de um único classificador $h : \mathcal{X} \rightarrow P(\mathcal{Y})$ [8]. A Tabela 3.4 mostra o resultado da transformação dos dados da Tabela 3.1 utilizando essa abordagem. Embora esse método leve em consideração a correlação entre os rótulos de cada instância, ele apresenta a desvantagem de pode gerar bases de dados com muitas classes e poucas instâncias por classe.

Tabela 3.4: Base de dados da Tabela 3.1 transformada utilizando LP.

Instâncias	Esportes	(Esportes∪Política)	(Ciência∪Política)	(Ciência∪Religião)
1		✓		
2			✓	
3	✓			
4				✓

3.1.3 Binary Relevance

Um dos métodos de transformação mais utilizado, o *Binary Relevance* (BR), constrói $|\mathcal{L}|$ classificadores binários $h_r : \mathcal{X} \rightarrow \{1, -1\}$ [8]. A ideia consiste em transformar o conjunto de dados original em $|\mathcal{L}|$ conjuntos de dados $\mathcal{D}_i$, os quais contém todas as instâncias da base de dados original classificadas como 1 quando essa instância tiver sido rotulada pela classe i, ou 0 caso contrário. Para a etapa de classificação, o método utiliza como saída um conjunto de rótulos obtido pela união dos rótulos de saída de cada um dos $|\mathcal{L}|$ classificadores. As Tabelas 3.5, 3.6, 3.7 e 3.8 mostram os quatro conjuntos de dados gerados por essa abordagem.

Tabela 3.5: Base de dados da Tabela 3.1 transformada utilizando BR: classe "Esportes".

Instâncias	Esportes	¬Esportes
1	✓	
2		✓
3	✓	
4		✓

Tabela 3.6: Base de dados da Tabela 3.1 transformada utilizando BR: classe "Política".

Instâncias	Política	¬Política
1	✓	
2	✓	
3		✓
4		✓

Tabela 3.7: Base de dados da Tabela 3.1 transformada utilizando BR: classe “Religião”.

Instâncias	Religião	¬Religião
1	✓	
2	✓	
3		✓
4		✓

Tabela 3.8: Base de dados da Tabela 3.1 transformada utilizando BR: classe “Ciência”.

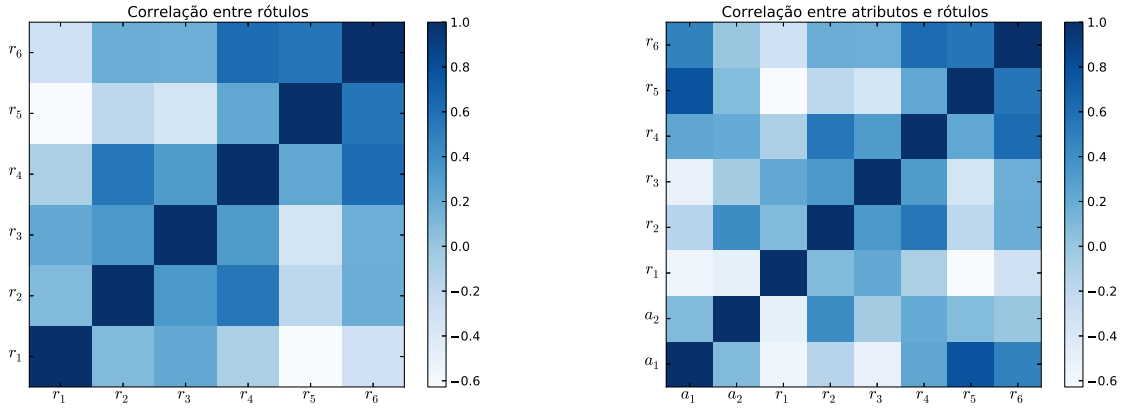
Instâncias	Ciência	¬Ciência
1		✓
2	✓	
3		✓
4	✓	

3.2 Visualizando os métodos Label Powerset e Binary Relevance

Nesta seção, utilizamos uma base artificial[1] com dois atributos para melhor ilustrarmos os problemas com múltiplos rótulos utilizando os métodos de transformação *Label Powerset* e *Binary Relevance*. A base tem quinhentas instâncias e seis rótulos.

As Figuras 3.1a e 3.1b apresentam a vizualização da matriz de correlação entre rótulos da base artificial e, a matriz de correlação entre os atributos e os rótulos, respectivamente. Na Figura 3.1a, pode-se observar como os seis rótulos $(r_1 - r_6)$ se correlacionam. Para essa base, os rótulos r_4 e r_6 apresentaram maior valor de correlação, enquanto os rótulos r_1 e r_5 a menor correlação. Isso significa que, a base apresenta um número representativo de instâncias onde os rótulos r_4 e r_6 aparecem simultaneamente. O contrário pode ser verificado para os rótulos r_1 e r_5. Na Figura 3.1b, pode-se verificar que o atributo a_1 e o rótulo r_5 apresentaram o maior valor de correlação, enquanto o atributo a_1 associada aos rótulos r_1 e r_3 os menores valores de correlação. Em problemas reais, a análise da correlação entre os rótulos e os atributos é uma prática interessante, principalmente quando utiliza-se os métodos LP, pois a correlação é considerada nesse método.

[1] `http://sites.labic.icmc.usp.br/mldatagen/`

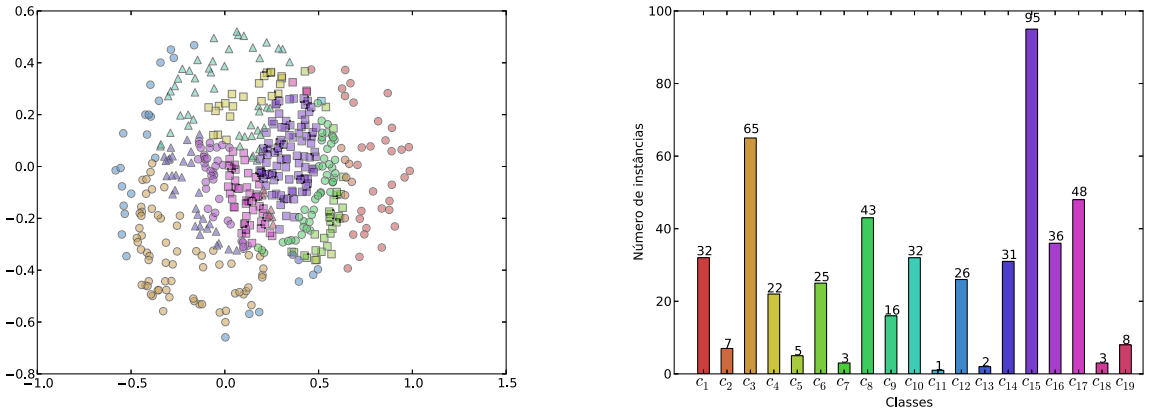


(a) (b)

Figura 3.1: (a) visualização da matriz de correlação dos rótulos da base artificial. Os rótulos r_4 e r_6 apresentaram maior valor de correlação, enquanto os rótulos r_1 e r_5 a menor correlação. (b) matriz de correlação entre os atributos a_1 e a_2 e os rótulos $r_1 - r_6$. O atributo a_1 e o rótulo r_5 apresentaram o maior valor de correlação, enquanto o atributo a_1 associada aos rótulos r_1 e r_3 os menores valores de correlação.

O resultado da aplicação do método LP na base artificial é apresentado na Figura 3.2a. Cada figura geométrica em uma dada cor representa uma instância de uma dada classe gerada por esse método. A Figura 3.2b, apresenta a distribuição do número de instâncias por cada classe. Nessa figura é possível verificar o grau de desequilíbrio entre o número de instâncias e as classes. Por exemplo, a classe c_{15} apresenta o maior número de instâncias (95), enquanto a classe c_{11} é a classe com o menor número de instâncias (1). Classificadores que utilizam o método de transformação LP devem ser capazes de gerenciar o elevado número de classes além do desequilíbrio entre elas.

A aplicação do método BR na base artificial pode ser verificada ao longo dos seis gráficos de dispersão dispostos na Figura 3.3. Em cada um dos gráficos, as instâncias são rotuladas segundo o seu conjunto de rótulos. Por exemplo, na Figura 3.3a as instâncias que têm o rótulo 1 são consideradas como uma classe (círculos azuis), ao passo que as instâncias que não têm o rótulo 1 são consideradas de outra classe (círculos verdes). Seguindo esse procedimento para todos os rótulos, gera-se seis espaços binários de classificação. Um ponto importante, é que classificadores utilizando o método BR devem ter uma boa capacidade de traça diferentes superfícies de decisão implícita ou explicitamente, dado



(a) (b)

Figura 3.2: (a) apresenta o gráfico de dispersão da base artificial após a aplicação do método de transformação LP. (b) apresenta a distribuição do número de instâncias ao longo das 19 classes geradas pelo método LP. A classe c_{15} contém o maior número de instâncias (95), ao passo que a classe c_{11} o menor número (1).

que à aplicação do método BR gera diferentes formas para as classes a medida que cada rótulo é considerado.

3.3 Algoritmos de Adaptação de Métodos

Os algoritmos de Adaboost.MH e Adaboost.MR [5] são duas extensões do algoritmo Adaboost original [22] para classificação com suporte a múltiplos rótulos. Ambos aplicam Adaboost em classificadores minimalistas da forma $f' : \mathcal{X} \times \mathcal{L} \to \mathbb{R}$. No caso da abordagem Adaboost.MH, se o sinal de saída do classificador minimalista é positivo para uma instância $x \in \mathcal{X}$ e um rótulo $r \in \mathcal{L}$, então consideramos que x pode ser rotulado com 1, ou 0 caso contrário. No caso de Adaboost.MR, a saída dos classificadores minimalistas é considerada para o ranqueamento dos rótulos em $\mathcal{L}$.

Embora essas duas abordagens são adaptações de um algoritmo de aprendizado já consagrado, eles na verdade utilizam uma abordagem de transformação de problema. Cada exemplo (x, y), onde $x \in \mathcal{X}$ e $y \in \mathcal{L}$, é decomposto em $|\mathcal{L}|$ exemplos da forma (x, r, y), $\forall r \in \mathcal{L}$, onde $y = 1$ se $r \in \mathcal{L}$ e $y = 0$, caso contrário. A Tabela 3.9 mostra o resultado da

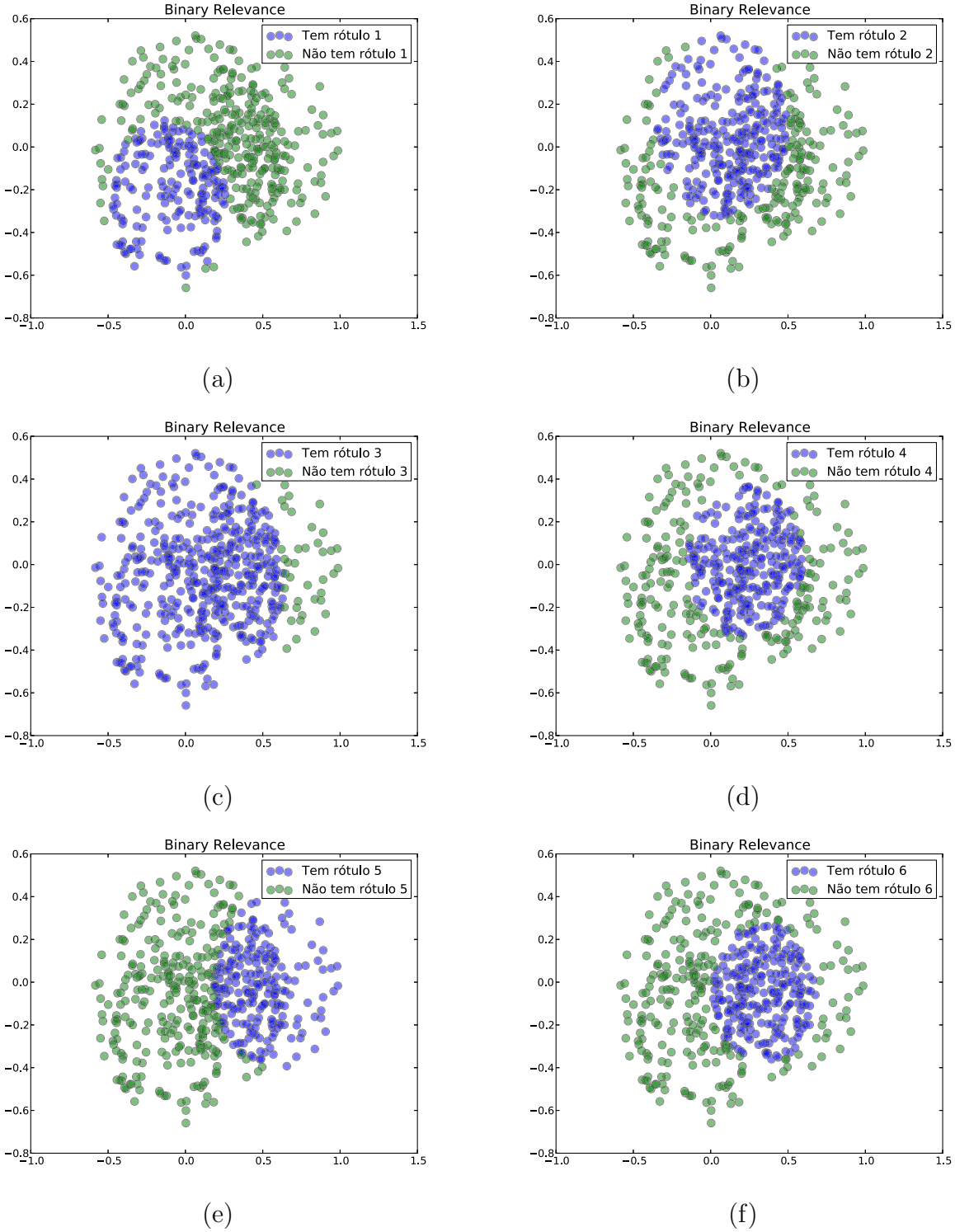


Figura 3.3: (a)-(f) apresentam a variação do dispersão do espaço binário considerando os rótulos de 1 a 6.

transformação da base de dados original (Tabela 3.1) utilizando essa abordagem.

Tabela 3.9: Dados originais da Tabela 3.1 transformados utilizando Adaboost.MR e Adaboost.MH.

Instância	r	y
1	Esportes	1
1	Religião	-1
1	Ciência	-1
1	Política	1
2	Esportes	-1
2	Religião	-1
2	Ciência	1
2	Política	1
3	Esportes	1
3	Religião	-1
3	Ciência	-1
3	Política	-1
4	Esportes	-1
4	Religião	1
4	Ciência	1
4	Política	-1

O algoritmo ML-kNN [12] é uma adaptação do k-NN original para o problema de classificação com suporte a múltiplos rótulos. A ideia consiste em, para cada rótulo $r \in \mathcal{L}$, encontrar os k vizinhos mais próximos da instância de teste e considerar aquelas rotuladas com a classe r como positivas, e as restantes como negativas. Luo e Zincir-Heywood [23] apresentaram dois sistemas para classificação de documentos com múltiplos rótulos baseados no classificador k-NN. A principal contribuição desse trabalho é o estágio de pré-processamento das instâncias da base de dados para a representação efetiva dos documentos: para a classificação de uma nova instância, o sistema inicialmente encontra as k instâncias mais próximas. Em seguida, para cada ocorrência de um determinado rótulo em cada uma dessas instâncias, um contador é incrementado para este rótulo. Finalmente, os Z rótulos com maior ocorrência são considerados. Note que Z é escolhido com base no número de rótulos $|\mathcal{L}|$.

Elisseeff e Weston [24] apresentaram um algoritmo de ranqueamento para SVM, o

qual tenta minimizar uma função de custo dada por uma função de perda qualquer. Já Godbole e Sarawagi [25] introduziram uma melhoria para o classificador SVM em conjunto com o método de transformação BR citado na seção anterior para suporte a múltiplos rótulos: a abordagem proposta estende o conjunto de dados original com $|\mathcal{L}|$ características extras contendo a predição de cada classificador binário. Em seguida, um retreinamento dos dados com esse novo vetor de características é executado. Para a classificação de uma nova instância, os classificadores binários da primeira rodada de treinamento são inicializados para a geração inicial de rótulos e sua saída é adicionada às características dessa instância para uma nova classificação pelos classificadores binários da segunda etapa. Cerri e de Carvalho [26, 27] propuseram algoritmos hierárquicos para aprendizado com suporte a múltiplos rótulos para Redes Neurais e Máquinas de Vetores de Suporte, respectivamente.

4 Aprendizado por Floresta de Caminhos Ótimos

4.1 Fundamentação teórica

Seja Z uma base de dados e Z_1 e Z_2 os conjuntos de treinamento e teste, respectivamente, com $|Z_1|$ e $|Z_2|$ instâncias, as quais podem ser pontos ou elementos de imagem (*pixels*, *voxels*, formas e objetos), tais que $Z = Z_1 \cup Z_2$. Seja $\lambda(s)$ uma função que associa o rótulo correto i, $i = 1, 2, \ldots, c$ da classe i a qualquer instância $s \in Z_1 \cup Z_2$. Seja, também, $S \in Z_1$ um conjunto de protótipos de todas as classes (isto é, instâncias importantes que melhor representam as classes). Seja $\vec{v}$ um algoritmo que extrai n atributos (cor, forma e propriedades de textura) de qualquer instância $s \in Z_1 \cup Z_2$, e retorna um vetor de atributos $\vec{v}(s) \in \Re^n$. A distância $d(s,t)$ entre duas instâncias, s e t, é dada pela distância entre seus vetores de características $\vec{v}(s)$ e $\vec{v}(t)$. Podemos utilizar qualquer métrica válida (Euclideana, por exemplo) ou algum algoritmo de distância mais elaborado [28]. Nosso problema consiste em usar S, $(\vec{v}, d)$, Z_1 e Z_2 para projetar um classificador otimal, o qual pode-se predizer o rótulo correto $\lambda(s)$ de qualquer instância $s \in Z_2$.

4.2 OPF com grafo completo

Seja (Z_1, A) um grafo completo cujos nós são as instâncias em Z_1, onde qualquer par de instâncias define um arco em A (isto é, $A = Z_1 \times Z_1$) (Figura 4.1a). Note que os arcos não precisam ser armazenados e o grafo não precisa ser explicitamente representado.

Um caminho é uma seqüência de instâncias $\pi = \langle s_1, s_2, \ldots, s_k \rangle$, onde $(s_i, s_{i+1}) \in A$ para $1 \leq i \leq k-1$. Um caminho é dito ser trivial se $\pi = \langle s_1 \rangle$. Nós associamos a cada caminho π o custo dado por uma função de custos suave f [29], denotada $f(\pi)$.

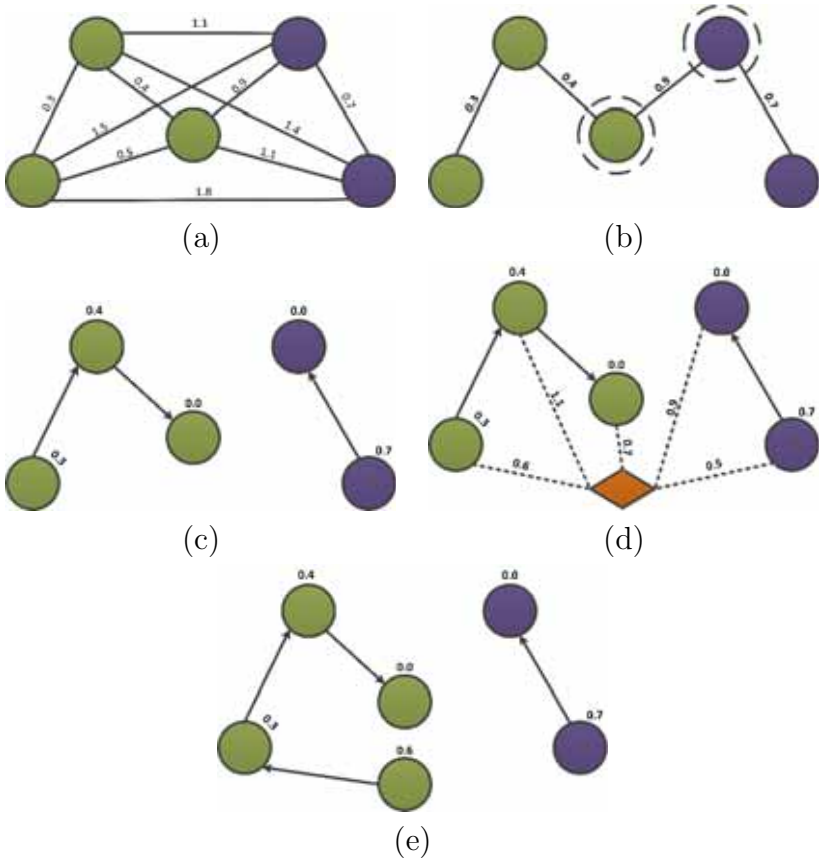


Figura 4.1: (a) Grafo completo ponderado nas arestas para um determinado conjunto de treinamento. (b) Floresta de caminhos ótimos resultante para f_{max} e dois protótipos (nós contornados). As entradas (x, y) acima dos nós denotam, respectivamente, o custo e o rótulo das instâncias. Os arcos direcionados indicam os nós predecessores no caminho ótimo. (c) Amostra de teste (losango) e suas conexões (linhas tracejadas) com os nós de treinamento. (d) O caminho ótimo do protótipo mais fortemente conexo, seu rótulo 2, e o custo de classificação 0.3 é associado à instância de teste.

Dizemos que um caminho π é ótimo se $f(\pi) \leq f(\tau)$ para qualquer caminho τ, onde π e τ terminam no mesma instância s, independente de sua origem. Também denotamos $\pi \cdot \langle s, t \rangle$ a concatenação do caminho π com término em s e o arco (s, t).

O algoritmo OPF pode ser utilizado com qualquer função de custo suave que pode agrupar instâncias com propriedades similares [29]. Entretanto, o OPF foi projetado usando a função de custo f_{max}, por causa de suas propriedades teóricas para estimar

protótipos ótimos [30]:

$$\begin{aligned} f_{max}(\langle s \rangle) &= \begin{cases} 0 & \text{se } s \in S, \\ +\infty & \text{caso contrário} \end{cases} \\ f_{max}(\pi \cdot \langle s,t \rangle) &= \max\{f_{max}(\pi), d(s,t)\}, \end{aligned} \tag{4.1}$$

sendo que $f_{max}(\pi)$ computa a distância máxima entre instâncias adjacentes em π, quando π não é um caminho trivial.

O algoritmo OPF associa um caminho ótimo $P^*(s)$ de S a toda instância $s \in Z_1$, formando uma floresta de caminhos ótimos P (uma função sem ciclos, a qual associa a todo $s \in Z_1$ seu predecessor $P(s)$ em $P^*(s)$, ou uma marca *nil* quando $s \in S$). Seja $R(s) \in S$ a raiz de $P^*(s)$, a qual pode ser alcançada usando $P(s)$. O algoritmo OPF computa, para cada $s \in Z_1$, o custo $V(s)$ de $P^*(s)$, o rótulo $L(s) = \lambda(R(s))$ e o seu predecessor $P(s)$, como segue abaixo.

Algoritmo 1 – OPF

ENTRADA:	Um conjunto de treinamento Z_1 λ-rotulado, protótipos $S \subset Z_1$ e o par (v, d) para vetor de características e cálculo das distâncias.
SAÍDA:	Floresta de caminhos ótimos P, mapa de valores de custo de caminhos V e mapa de rótulos L
AUXILIARES:	Fila de prioridades Q, e variável *cst*.

1. **Para todo** $s \in Z_1$, **Faça** $P(s) \leftarrow nil$ *e* $V(s) \leftarrow +\infty$.
2. **Para todo** $s \in S$, **Faça** $V(s) \leftarrow 0$, $P(s) \leftarrow nil$, $L(s) = \lambda(s)$ *e insira* s *em* Q.
3. **Enquanto** Q *é não vazia,* **Faça**
4. *Remova de* Q *uma instância* s *tal que* $V(s)$ *é mínimo.*
5. **Para cada** $t \in Z_1$ *tal que* $s \neq t$ *e* $V(t) > V(s)$, **Faça**
6. *Calcule* $cst \leftarrow \max\{V(s), d(s,t)\}$.
7. **Se** $cst < V(t)$, **Então**
8. **Se** $V(t) \neq +\infty$, **Então** *remova* t *de* Q.
9. $P(t) \leftarrow s$, $L(t) \leftarrow L(s)$ *e* $V(t) \leftarrow cst$.
10. *Insira* t *em* Q.

As linhas 1–2 inicializam os mapas e inserem protótipos em Q. O laço principal calcula um caminho ótimo de S para cada instância $s \in Z_1$ em uma ordem não decrescente de custos (linhas 3–10). A cada iteração um caminho de custo de ótimo $V(s)$ é obtido em P (linha 4). Empates são resolvidos em Q utilizando a política FIFO (first-in-first-out), ou seja, quando dois caminhos atingem uma determinada instância s com o mesmo custo mínimo, s é associado ao primeiro caminho que o atingiu. O restante das linhas avalia se o caminho que atinge uma instância adjacente t através de s é mais barato que o caminho que termina em t. Caso positivo, atualiza Q, $P(t)$, $L(t)$ e $V(t)$. No final do algoritmo, V armazena o valor do custo do caminho ótimo de S a cada instância $s \in Z_1$ de acordo com f_{max}.

O algoritmo OPF com grafo completo utilizando f_{max} pode ser entendido como uma transformada de Watershed usando a IFT [31] computada no espaço de características n-dimensional. Além dessa extensão, outras contribuições significativas foram os processos de treinamento e aprendizado, o qual encontra protótipos ótimos (marcadores) na fronteira entre as classes afastando *outliers* (instâncias de uma determinada classe que estão presentes em uma região de outra classe no espaço de atributos) do conjunto de treinamento, aumentando a acurácia do classificador.

4.2.1 Treinamento

Dizemos que S^* é um conjunto ótimo de protótipos quando o Algoritmo 1 propaga os rótulos $L(s) = \lambda(s)$ para todo $s \in Z_1$. Desta forma, S^* pode ser encontrado explorando a relação teórica entre a Árvore de Espalhamento Mínima (*Minimum Spanning Tree* - MST) [30] e a árvore de caminhos mínimos para f_{max}. O treinamento consiste essencialmente em encontrar S^* e um classificador OPF enraizado em S^*.

Computando uma MST no grafo completo (Z_1, A), obtemos um grafo conexo acíclico cujos nós são todas as instâncias em Z_1, e os arcos são não direcionados e ponderados (Figura 4.1b). Seus pesos são dados pela distância d entre os vetores de atributos de instâncias adjacentes. Esta árvore de espalhamento é ótima no sentido em que a soma dos pesos de seus arcos é mínima se comparada a outras árvores de espalhamento no grafo completo. Na MST, cada par de instâncias é conectado por um caminho, o qual é ótimo

de acordo com f_{max}. Os protótipos ótimos são os elementos conectados na MST com diferentes rótulos em Z_1, isto é, elementos mais próximos de classes diferentes. Removendo-se os arcos entre classes diferentes, tais instâncias adjacentes tornam-se protótipos em S^* e o Algoritmo 1 pode computar uma floresta de caminhos ótimos sem erros de classificação em Z_1.

4.2.2 Classificação

Para qualquer instância $t \in Z_2$, consideramos todos os arcos conectando t com instâncias $s \in Z_1$, tornando t como se fosse parte do grafo (ver Figura 4.1c, onde a instância t é representada pelo losango no grafo). Considerando todos os possíveis caminhos entre S^* e t, desejamos encontrar o caminho ótimo $P^*(t)$ de S^* até t com a classe $\lambda(R(t))$ de seu protótipo mais fortemente conexo $R(t) \in S^*$. Este caminho pode ser identificado incrementalmente, avaliando o valor do custo ótimo $V(t)$ como

$$V(t) = \min\{\max\{V(s), d(s,t)\}\},\ \forall s \in Z_1. \tag{4.2}$$

Seja $s^* \in Z_1$ o nó que satisfaz a equação acima (isto é, o predecessor $P(t)$ no caminho ótimo $P^*(t)$). Dado que $L(s^*) = \lambda(R(t))$, a classificação simplesmente associa $L(s^*)$ como a classe de t (Figura 4.1d). Um erro ocorre quando $L(s^*) \neq \lambda(t)$.

4.3 OPF com grafo *k*-nn

A técnica de classificação supervisionada baseada em floresta de caminhos ótimos apresentada nesta seção modela as instâncias novamente como sendo os nós de um grafo [32]. Entretanto, a relação de adjacência utilizada é a dos k-vizinhos mais próximos e tanto os arcos quanto os nós são ponderados. A diferença básica entre esta abordagem e a previamente apresentada (Seção 4.2), é o fato de a abordagem que faz uso do grafo completo estimar protótipos na fronteira das classes, enquanto a metodologia a ser apresentada aqui estima os protótipos nos pontos de alta concentração de instâncias.

A principal motivação desta representação seria o estudo de outras metodologias para classificadores supervisionados baseados em florestas de caminhos ótimos, utilizando outras funções de custo, relações de adjacência e maneiras diferentes de se encontrar os protótipos. As definições, bem como os algoritmos desta abordagem, serão apresentados nas próximas seções.

4.3.1 Fundamentação teórica

Dado um grafo (Z_1, A_k), o algoritmo OPF com grafo k-nn (OPFkNN) particiona o conjunto Z_1 em uma floresta de caminhos ótimos de acordo com uma função de densidade de probabilidade (fdp). Nesta abordagem temos um grafo ponderado nos vértices e nos arcos. A distância $d(s,t)$ entre duas instâncias $s,t \in Z_1$ novamente define o peso do arco (s,t). Os nós são ponderados pelos seus valores de densidade de probabilidade, os quais são computados usando o peso dos arcos, como segue:

$$\rho(s) = \frac{1}{\sqrt{2\pi\sigma^2}|\mathcal{A}(s)|} \sum_{\forall t \in \mathcal{A}(s)} \exp\left(\frac{-d^2(s,t)}{2\sigma^2}\right), \tag{4.3}$$

onde $\sigma = \frac{d_f}{3}$ e d_f é o comprimento do maior arco em (Z, A_k). A escolha deste parâmetro considera todos os nós para o calculo da densidade, assumindo que uma função gaussiana cobre a grande maioria das instâncias com $d(s,t) \in [0, 3\sigma]$.

A função de valor de caminho dada pela Equação 4.4 associa a um nó terminal s de cada caminho o mínimo entre os valores de densidade ao longo do mesmo e um valor de densidade inicial.

$$\begin{aligned} f_2(\langle t \rangle) &= \begin{cases} \rho(t) & \text{se } t \in S \\ h(t) & \text{caso contrário} \end{cases} \\ f_2(\pi_s \cdot \langle s,t \rangle) &= \min\{f(\pi_s), \rho(t)\}. \end{aligned} \tag{4.4}$$

O caminho de valor máximo para cada instância s, a partir de um conjunto de elementos protótipos (raízes), particiona o conjunto de treinamento Z_1 em uma floresta de caminhos ótimos. As raízes da floresta formam um subconjunto dos máximos da fdp onde, cada

raiz, define uma árvore de caminhos ótimos (zona de influência do respectivo máximo) composta pelas suas instâncias mais fortemente conexas, as quais recebem o mesmo rótulo de sua raiz, como pode ser demonstrado pelo Algoritmo 2.

Algoritmo 2 – Classificador Supervisionado baseado em Floresta de Caminhos Ótimos usando grafo k-nn

Entrada:	Grafo k-nn (Z_1, A_k), $\lambda(s)$ para todo $s \in Z_1$ e função de valor de caminho f_1.
Saída:	Mapa de rótulos L, mapa de valores de caminho V e a floresta de caminhos ótimos P.
Auxiliares:	Fila de prioridade Q e a variável *tmp*.

1. **Para todo** $s \in Z_1$, **Faça** $P(s) \leftarrow nil$, $V(s) \leftarrow \rho(s) - \delta$, $L(s) \leftarrow \lambda(s)$ *e insira* s *em* Q.
2. **Enquanto** Q *é não vazia,* **Faça**
3. *Remova de* Q *uma instância* s *tal que* $V(s)$ *é máximo.*
4. **Se** $P(s) = nil$, **Então**
5. $V(s) \leftarrow \rho(s)$.
6. **Para cada** $t \in A_k(s)$ *e* $V(t) < V(s)$, **Faça**
7. $tmp \leftarrow \min\{V(s), \rho(t)\}$.
8. **Se** $tmp > V(t)$, **Então**
9. $L(t) \leftarrow L(s)$, $P(t) \leftarrow s$, $V(t) \leftarrow tmp$.
10. *Atualize posição de* t *em* Q.

Inicialmente (Linha 1), todos os caminhos são triviais com valores $f_1(\langle t \rangle) = \rho(t) - \delta$. Os máximos globais da fdp são os primeiros elementos a serem removidos de Q, os quais são identificados como sendo as raízes da floresta pelo teste $P(s) = nil$ na linha 4, onde atualizamos o seu valor de caminho correto $f_1(\langle t \rangle) = V(s) = \rho(t)$. Cada no s removido de Q oferece um caminho $\pi_s \cdot \langle s, t \rangle$ para cada no t adjacente no laço da Linha 6 até a Linha 10. Se o valor de caminho $f_1(\pi_s \cdot \langle s, t \rangle) = \min\{V(s), \rho(t)\}$ (Linha 7) é melhor que o valor de caminho atual $f_1(\pi_t) = V(t)$ (Linha 8), então π_t é atualizado para $\pi_s \cdot \langle s, t \rangle$ (isto é, $P(s) \leftarrow s$, e o valor do caminho e o rótulo de t são atualizados (Linha 9). Máximo locais da fdp são também identificados como sendo raízes durante a execução do algoritmo, o

qual tem como saída o mapa de rótulos L, mapa de valores de caminho V e a floresta de caminhos ótimos P.

4.3.2 Treinamento

Um erro de classificação no conjunto de treinamento ocorre quando $L(t) \neq \lambda(t)$. Assim, definimos o melhor valor de $k^* \in [1, k_{max}]$ como sendo aquele que maximiza a acurácia da classificação no conjunto de treinamento.

É esperado que cada classe seja representada por pelo menos um máximo da fdp e $L(t) = \lambda(t)$ para todo $t \in Z_1$ (erro zero de classificação no conjunto de treinamento). Entretanto, essas propriedades não podem ser garantidas com a função de valor de caminho f_2 e o melhor valor k^*. Com o intuito de assegurar tais propriedades, primeiro encontramos k^* utilizando f_2 e então executamos o Algoritmo 2 mais uma vez usando a função de valor de caminho f_3 ao contrário de f_2, dada por

$$\begin{aligned} f_3(\langle t \rangle) &= \begin{cases} \rho(t) & \text{se } t \in \mathcal{S} \\ \rho(t) - \delta & \text{caso contrário} \end{cases} \\ f_3(\pi_s \cdot \langle s, t \rangle) &= \begin{cases} -\infty & \text{se } \lambda(t) \neq \lambda(s) \\ \min\{f_2(\pi_s), \rho(t)\} & \text{caso contrário.} \end{cases} \end{aligned} \tag{4.5}$$

A Equação 4.5 pondera todos os arcos $(s, t) \in A_k$ tais que $\lambda(t) \neq \lambda(s)$ com $d(s, t) = -\infty$, impedindo que tais arcos pertençam a algum caminho ótimo. O processo de treinamento acima é descrito pelo Algoritmo 3.

Algoritmo 3 – TREINAMENTO

ENTRADA:	Conjunto de treinamento Z_1, $\lambda(s)$ para todo $s \in Z_1$, k_{max} e funções de valor de caminho f_2 e f_3.
SAÍDA:	Mapa de rótulos L, mapa de valores de caminho V e a floresta de caminhos ótimos P.
AUXILIARES:	Variáveis i, k, k^*, $MaxAcc \leftarrow -\infty$, Acc e vetores FP e FN de tamnho c.

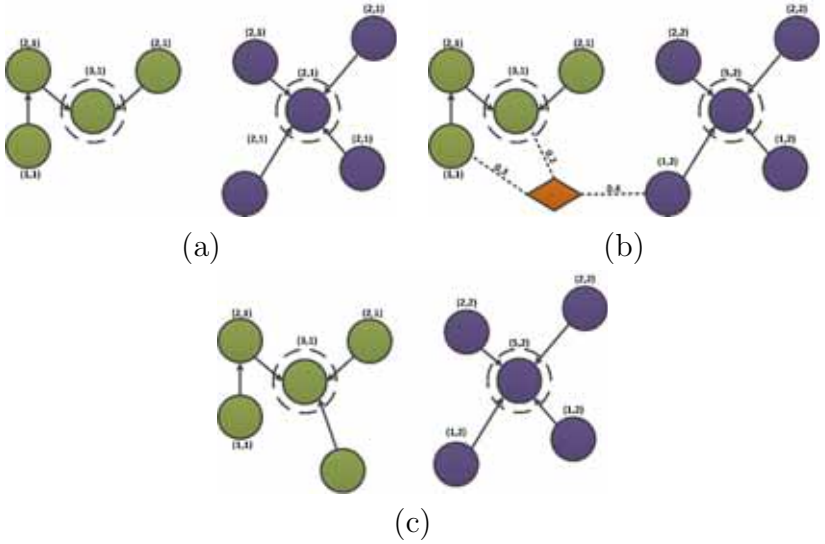
1. *Para* $k = 1$ *até* k_{max} *faça*
2. *Crie um grafo* (Z_1, A_k) *ponderado nos nós através da Equação 4.3.*
3. *Calcule* (L, V, P) *utilizando o Algoritmo 2 com* f_2.
4. *Para cada classe* $i = 1, 2, \ldots, c$, *faça*
5. $FP(i) \leftarrow 0$ *e* $FN(i) \leftarrow 0$.
6. *Para cada instância* $t \in Z_1$, *faça*
7. *Se* $L(t) \neq \lambda(t)$, *então*
8. $FP(L(t)) \leftarrow FP(L(t)) + 1$.
9. $FN(\lambda(t)) \leftarrow FN(\lambda(t)) + 1$.
10. *Calcule Acurácia.*
11. *Se* $Acc > MaxAcc$, *então*
12. $k^* \leftarrow k$ *e* $MaxAcc \leftarrow Acc$.
13. *Destrua o grafo* (Z_1, A_k).
14. *Crie o grafo* (Z_1, A_{k^*}) *ponderados nos nós através da Equação 4.3.*
15. *Calcule* (L, V, P) *utilizando o Algoritmo 2 com* f_3.

4.3.3 Classificação

Uma instância $t \in Z_3$ pode ser associada a uma dada classe $i, i = 1, 2, \ldots, c$ simplesmente identificando qual raiz (protótipo) oferece o caminho ótimo como se esta instância pertencesse à floresta. Considerando os k-vizinhos mais próximos de t em Z_3, podemos utilizar a Equação 4.3 para computar $\rho(t)$, avaliar os caminhos ótimos $\pi_s \cdot \langle s, t \rangle$ e selecionar o que satisfaz a seguinte equação :

$$V(t) = \max_{\forall (s,t) \in A_{k*}} \{\min\{V(s), \rho(t)\}\} \tag{4.6}$$

O rótulo de t será o mesmo rótulo do seu predecessor $P(t)$, ou seja, $L(t) = L(P(s))$. Um erro ocorre quando $L(t) \neq \lambda(t)$. A Figura 4.2 ilustra esse processo.



(a) (b)

(c)

Figura 4.2: (a) Floresta de caminhos ótimos obtida através do Algoritmo 2, cujos elementos são ponderados nos nós com seus respectivos valores de densidade. Os indicadores (x, y) acima dos nós são, respectivamente, o seu valor de densidade e o rótulo da classe a qual ele pertence. Os elementos circulados representam os máximos de cada classe. (b) Processo de classificação, onde um nó $t \in Z_3$ (losango) a ser classificado é inserido no grafo e conectado aos seus k-vizinhos mais próximos ($k = 3$ no exemplo), e o seu valor de densidade é calculado. (c) Processo final da classificação, no qual a instância t é classificada com o rótulo da instância $s \in Z_1$ que satisfaz a Equação 4.6, ou seja, $L(t) = L(s)$. No exemplo acima, o elemento a ser classificado foi conquistado pelo máximo da classe 1.

A Figura 4.2a ilustra uma floresta de caminhos ótimos obtida através do Algoritmo 2, cujos elementos são ponderados nos nós com seus respectivos valores de densidade. A Figura 4.2b apresenta o processo de classificação, onde um nó $t \in Z_3$ (losango) a ser classificado é inserido no grafo e conectado aos seus k-vizinhos mais próximos ($k = 3$ no exemplo), e o seu valor de densidade é calculado (seu rótulo é ainda desconhecido). A Figura 4.2c ilustra o processo final da classificação, no qual a instância t é classificada com o rótulo da instância $s \in Z_1$ que satisfaz a Equação 4.6, ou seja, $L(t) = L(s)$. No exemplo acima, o elemento a ser classificado foi conquistado pelo máximo da classe 1.

4.4 OPFkNN para problemas com múltiplos rótulos

Nesta seção, propomos algumas modificações no algoritmo OPFkNN com o objetivo de melhorar as taxas de classificação do mesmo em problemas com múltiplos rótulos. É importante ressaltar que essas modificações não conferem ao OPFkNN condições de ser aplicado diretamente em problemas com múltiplos rótulos.

4.4.1 Treinamento

Dado o conjunto de treinamento Z_1, definimos um grafo (Z_1, A'_k) onde A'_k representa uma relação de adjacência entre os k-vizinhos mais próximos de uma instância $s \in Z_1$ e que pertençam a mesma classe de s, ou seja, $A'_k(s) = \{\forall t \in Z_1 | L(t) = L(s)\}$. O objetivo do uso dessa relação de adjacência é aumentar a probabilidade de que protótipos sejam escolhidos em regiões densas e homogêneas, ou seja, regiões onde a concentração de instâncias de mesma classe é alta. Essa modificação faz com que o grafo seja menos conexo, o que pode contribuir para a escolha do parâmetro k^* mais adequado ao conjunto de treinamento, dado que rótulos incorretos não serão propagados em caminhos longos, assim diminuindo a taxa de erro em Z_1.

Essa política de escolha de protótipos pode ser útil quando o classificador OPFkNN é combidado a métodos de tranformação de problemas tais como o BR e o LP. Em problemas transformados pelo método BR, por exemplo, pode ocorrer regiões onde a vizinhança de uma instância candidata a protótipo não apresente em sua maioria instâncias de mesma classe, o que pode ocasionar escolha de protótipos inconsistentes que não representam bem uma classe. Nos problemas transformados pelo método LP, devido ao número elevado de classes que o método pode gerar, o OPFkNN pode ser induzido a um número elevado de nós trivias durante o treinamento, o que pode degenerar o OPFkNN para um classificador 1-nn, um comportamento nem sempre desejável. Assim, torna-se importante determinar protótipos que realmente represente uma dada classe.

4.4.2 Classificação

Propomos nesta seção uma modificação na forma em que uma classe é associada a uma amostra $t \in Z_3$. Nesta abordagem, o rótulo de t não é mais definido pela instância $s \in Z_1$ que ofereça o melhor custo a t, como definido na Equação 4.6. A ideia é verificar entre os k^*-vizinhos mais próximos de t, qual rótulo tem o maior peso. O peso de um rótulo é definido pela soma dos custos oferecido pelos elementos pertencentes àquela classe. Matematicamente, o rótulo de t é definido pela seguinte equação:

$$L(t) = \max_i \{ \sum_{\forall (s,t) \in A'_{k^*}(t) | L(s)=i} \min\{V(s), \rho(t)\}\}, i = 1, 2, ..., c \tag{4.7}$$

O objetivo dessa abordagem de classificação é fazer com que o OPFkNN seja "influenciado" pelo contexto, ou seja, pela observação dos rótulos dos k^*-vizinhos mais próximos. Esse tipo de abordagem contextual tem monstrado bons resultados no em problemas com múltiplos rótulos. Um bom exemplo é o classificador ML-kNN [12].

5 Experimentos

Neste capítulo, apresentamos os experimentos realizados para verificar o desempenho dos classificadores baseados em OPF associados ao métodos BR e LP no contexto de múltiplos rótulos. Inicialmente, apresentamos a metodologia e as bases de dados utilizados nos experimentos. Em seguida, apresentamos os resultados obtidos da comparação entre os classificadores OPFkNN1 e OPFkNN2. Finalmente, comparamos os classificadores OPF com outros classificadores utilizados na literatura para classificação com múltiplos rótulos.

5.1 Metodologia e métodos

Os classificadores OPF com grafo completo e grafo k-nn (OPFkNN1 e OPFkNN2) foram implementados em linguagem Java utilizando o framework Weka [33]. Para comparar o desempenho desses classificadores, optamos por três diferentes classificadores: a árvore de decisão J48 (*decision tree*), o classificador bayesiano Naïve Bayes (NB) e o SVM, todos disponíveis no framework Weka. Para os métodos BR e LP, utilizamos a implementação da biblioteca Mulan [34].

Para os classificadores OPF, utilizamos a distância euclidiana como função de similaridade entre as instâncias. Durante comparação entre os classificadores OPFkNN1 e OPFkNN2, variamos o valor do parâmetro k de 1 a 25 com passos de 2. Para o treinamento do SVM, utilizamos a implementação da LibSVM [35] e o kernel RBF (*Radial Base Function*). Os parâmetros foram otimizados por validação cruzada. Os valores $2^{-15}, 2^{-13}, \ldots, 2^{1}, 2^{3}$ foram utilizados para o parâmetro *gamma*, e os valores $2^{1}, 2^{2}, \ldots, 2^{14}, 2^{15}$ para otimização do parâmetro C.

Com objetivo de avaliar a capacidade de generalização dos classificadores, utilizamos a metodologia de validação cruzada *k-fold* com $k = 5$. Como medida de desempenho dos classificadores, utilizamos seis métricas diferentes: *Accuracy*, F_1 *score*, *Hamming Loss*, *Precision*, *Recall* e *Subset Accuracy*.

Para a avaliação dos experimentos, foram realizadas duas rodadas de testes estatísticos. Primeiramente, aplicamos o teste não-paramétrico de Friedman, o qual foi utilizado para verificar se havia diferença significativa entre os classificadores. Nos casos onde o teste de Friedman apontou diferença significativa, aplicamos um teste estatísticos posterior. Para essa finalidade, utilizamos o teste de Nemenyi [36], o qual permitiu verificar entre quais classificadores houve diferença crítica (*critical difference*-CD). Os resultados dos testes são apresentados em um diagrama, no qual os postos médios dos classificadores são dispostos em um eixo horizontal. Grupos de classificadores sem diferença significativa entre eles (ao nível de significância de $p = 0,10$) são conectados por uma linha horizontal. Mais detalhes sobre estes procedimentos podem ser encontrados em Demšar et al. [37].

5.1.1 Bases de Dados

Os experimentos foram realizados utilizando sete bases de dados públicas[1] muito utilizadas pela comunidade interessada em problemas com múltiplos rótulos [10, 21]. A Tabela 5.1 apresenta as principais características das bases:

Tabela 5.1: Base de dados utilizadas nos experimentos. A coluna "#Distintos" denota o número de combinações de rótulos em cada base.

Base	Domínio	#Instâncias.	#Atributos.	#Rótulos	Cardinalidade	Densidade	#Distintos
Birds	audio	465	258	19	1.014	0.053	133
CAL500	música	502	68	174	26.044	0.150	502
Flags	imagens	194	19	7	3.392	0.485	54
Genbase	biologia	662	1186	27	1.252	0.046	32
Emotions	música	593	72	6	1.869	0.311	27
Scene	imagens	2407	294	6	1.074	0.179	15
Yeast	biologia	2417	103	14	4.237	0.303	198

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

5.2 Resultados

5.2.1 OPFkNN1 e OPFkNN2

Nesta seção, apresentamos os resultados dos experimentos realizados para a comparação entre os classificadores OPFkNN1 e OPFkNN2. Os resultados foram dispostos nos gráficos das Figuras 5.1 a 5.14. As Tabelas 5.2 e 5.3 apresentam (em porcentagem) a relação entre as melhores taxas de acerto do classificador OPFkNN1 e OPFkNN2 utilizando os métodos LP e BR, respectivamente. Os valores positivos denotam que o OPFkNN2 foi superior ao OPFkNN1, enquanto valores negativos denotam que o OPFkNN2 foi inferior ao OPFkNN1.

De modo geral, os classificadores OPFkNN2+BR e OPFkNN2+LP apresentaram respectivamente melhores resultados que o classificador OPFkNN1+BR e OPFkNN1+LP na maioria das bases e métricas. Esse comportamento pode ser atribuído à análise contextual realizada para a propagação da classe na fase de classificação, ou seja, na utilização das k-instâncias com melhor custo para o cômputo da classe com mair peso entre elas. No entanto, o classificador OPFkNN2+BR não melhorou as taxas de *Recall* nas bases *Birds*, *Emotions*, *Genbase*, *Scene* e *Yeast*, e obteve *Recall* inferior na *CAL500* (base com maior número de rótulos) comparado ao OPFkNN1. Pode-se verificar que o valor de *Recall* foi inversamente proporcional ao aumento no número dos k-vizinhos mais próximos. Isso significa que, nessas bases de dados, o OPFkNN2+BR elevou o número de falsos negativos no conjunto de rótulos predito a medida que variamos o valor do parâmetro k. Comportamento oposto nas taxas de *Recall* pode ser observado na base *Flags*. É importante notar que essa base de dados apresenta maior valor de densidade entre todas as bases, característica que pode ter influenciado no número de falsos positivos e, consequentemente nos valores de *Recall*. Contudo, essa característica da base *Flags* pode ter influenciado nas baixas taxas do OPFkNN2+BR segundo a métrica *Subset Accuracy*, ou seja, o método não foi eficaz em classificar conjuntos de rótulos com exatidão.

Já o OPFkNN2+LP demonstrou bons resultados principalmente no que diz respeito à métrica *Hamming Loss* e, consequentemente, boas taxas de *Subset Accuracy*, métricas relacionadas à erros de classificação e exatidão de conjuntos de rótulos, respectivamente.

Além disso, o OPFkNN2+LP manteve o compromisso entre as métricas *Precision* e *Recall*, o que significa que o classificador não elevou o número de falsos positivos e falsos negativos nos conjuntos de rótulos classificados. Cabe ressaltar que na base *Birds* o OPFkNN2+LP elevou apenas as taxas da métrica *Subset Accuracy*. Pode se verificar que essa base apresenta um elevado número de conjunto de rótulos distintos, o que exige do classificador alta capacidade de generalização em um número elevado de classes (133) com poucas instâncias por classe.

O comportamento dos classificadores OPFkNN2+BR e OPFkNN2+LP foi idêntico na base *Genbase*. Os resultados apresentaram pouca melhora apenas nas taxas de *Precision*, e não melhoraram as taxas segundo as outras métricas comparado aos resultados obtidos pelos classificadores OPFkNN1+BR e OPFkNN1+LP.

Tabela 5.2: Relação (%) entre os classificadores OPFkNN1 e OPFkNN2 utilizando o método LP. Os valores positivos denotam que o OPFkNN2 foi superior ao OPFkNN1, enquanto os valores negativos que o OPFkNN2 foi inferior ao classificador OPFkNN1.

Base	Accuracy	F_1 score	Hamming Loss	Precision	Recall	Subset Accuracy
Birds	0.00	0.00	0.00	0.00	0.00	3.98
CAL500	3.55	3.23	2.65	4.91	2.34	0.00
Emotions	8.24	5.93	9.73	2.52	7.81	22.51
Flags	6.01	4.79	9.40	5.47	3.57	12.08
Genbase	0.00	0.00	0.00	0.05	0.00	0.00
Scene	5.82	4.93	13.19	5.98	3.12	8.60
Yeast	7.85	6.89	12.40	8.20	3.66	13.47
Média	4.49	3.68	6.76	3.87	2.92	8.66

Tabela 5.3: Relação (%) entre os classificadores OPFkNN1 e OPFkNN2 utilizando o método BR. Os valores positivos denotam que o OPFkNN2 foi superior ao OPFkNN1, enquanto os valores negativos que o OPFkNN2 foi inferior ao classificador OPFkNN1.

Base	Accuracy	F_1 score	Hamming Loss	Precision	Recall	Subset Accuracy
Birds	3.48	1.97	15.04	4.86	0.00	10.05
CAL500	4.37	4.43	29.48	41.89	-2.76	0.00
Emotions	4.73	2.95	17.36	6.84	0.00	14.80
Flags	8.91	8.13	15.77	8.71	9.35	-0.32
Genbase	0.00	0.00	0.00	0.05	0.00	0.00
Scene	1.02	0.54	17.02	0.96	0.00	3.05
Yeast	6.51	6.89	19.35	15.87	0.00	0.00
Média	4.14	3.55	16.28	11.31	0.94	3.94

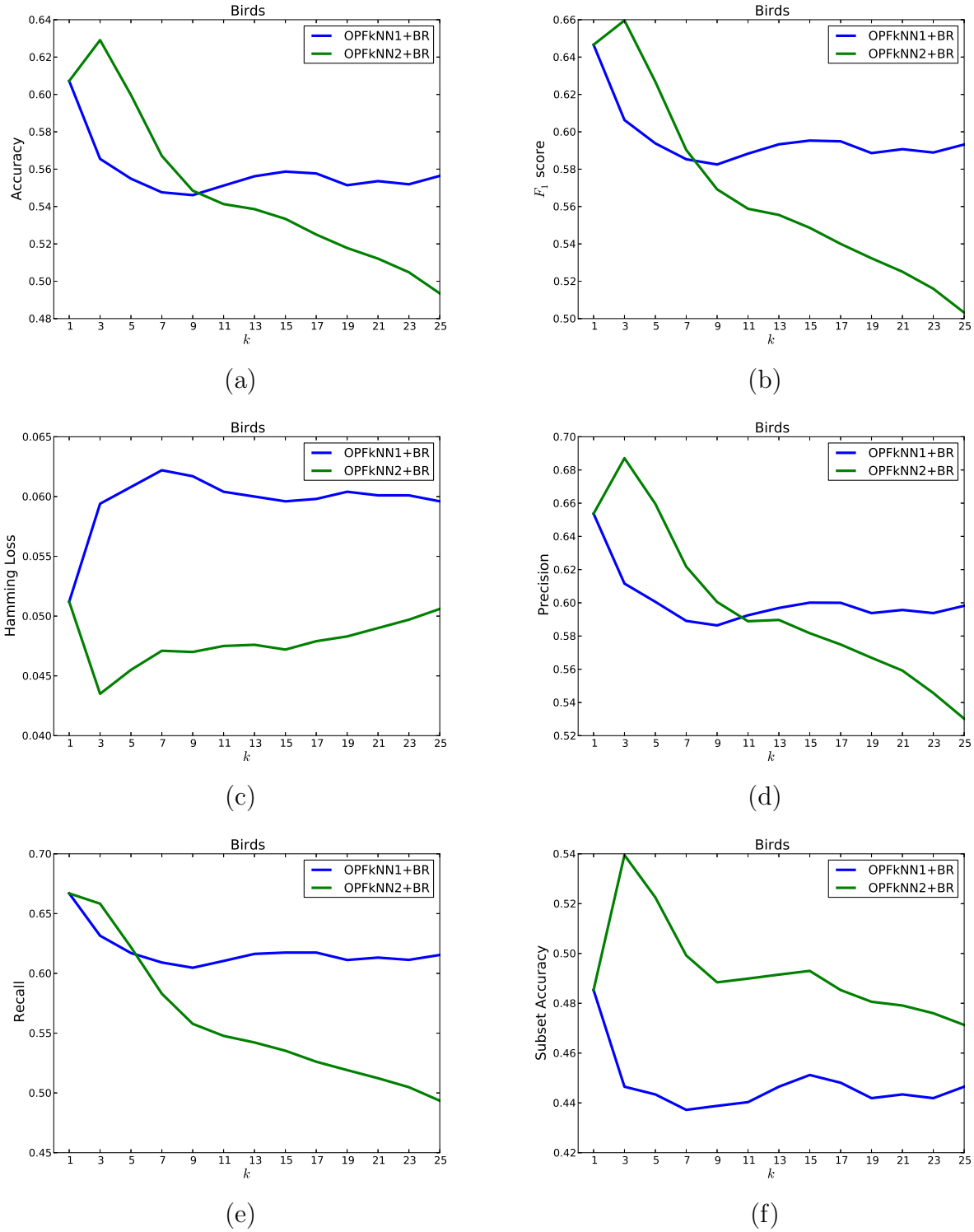


Figura 5.1: Curva de desempenho dos classificadores OPFkNN1+BR e OPFkNN2+BR na base *Birds* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

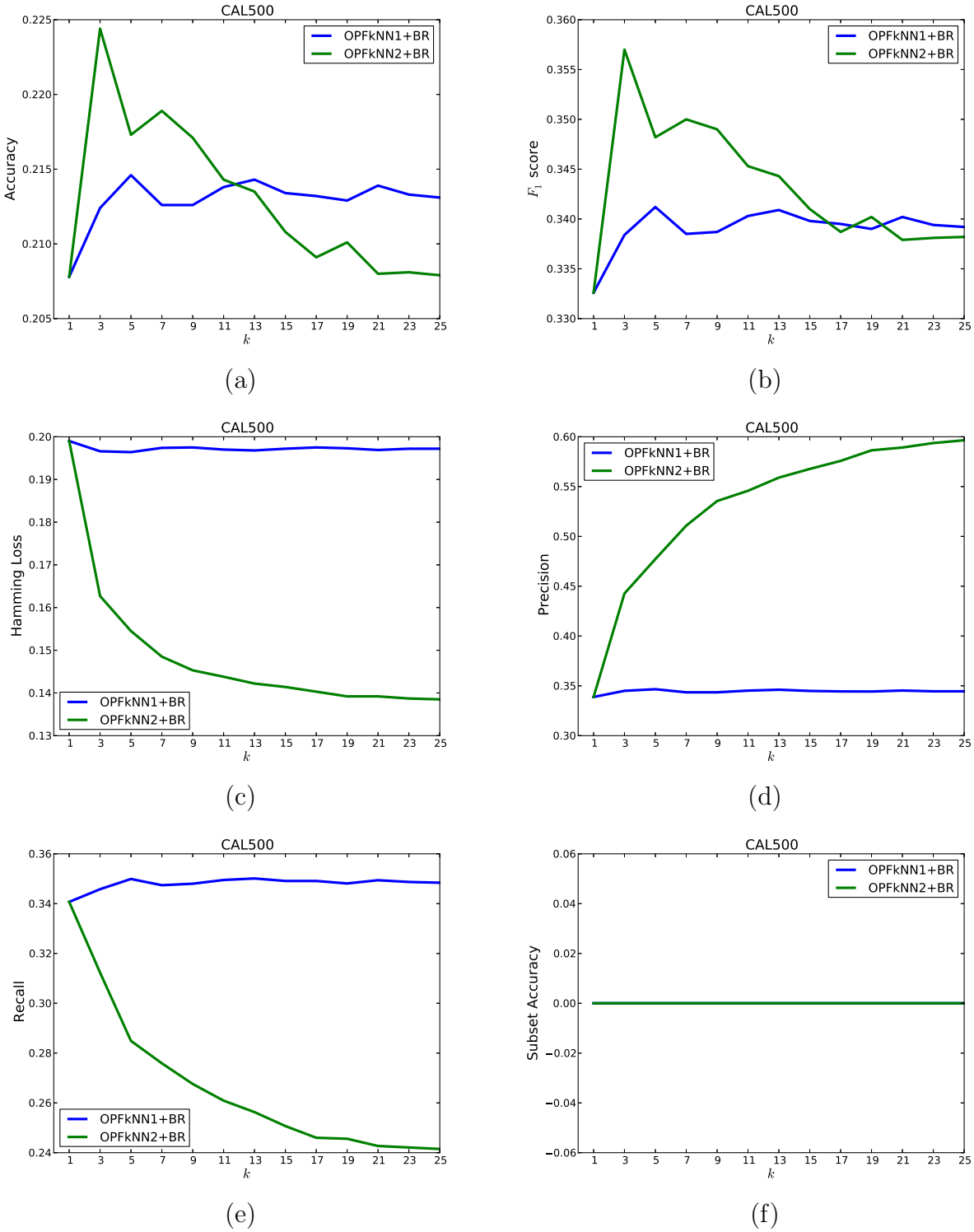


Figura 5.2: Curva de desempenho dos classificadores OPFkNN1+BR e OPFkNN2+BR na base *CAL500* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

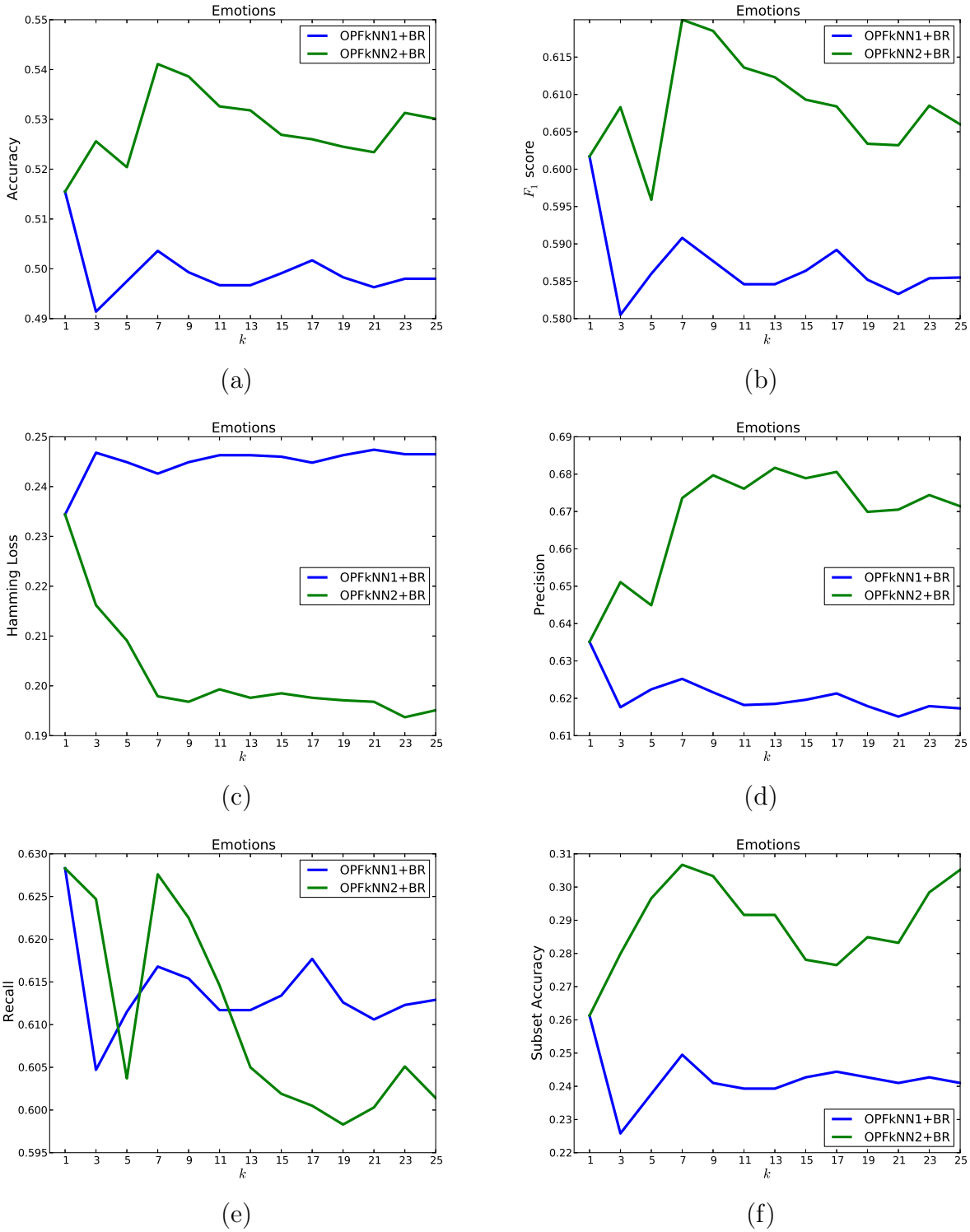


Figura 5.3: Curva de desempenho dos classificadores OPFkNN1+BR e OPFkNN2+BR na base *Emotions* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

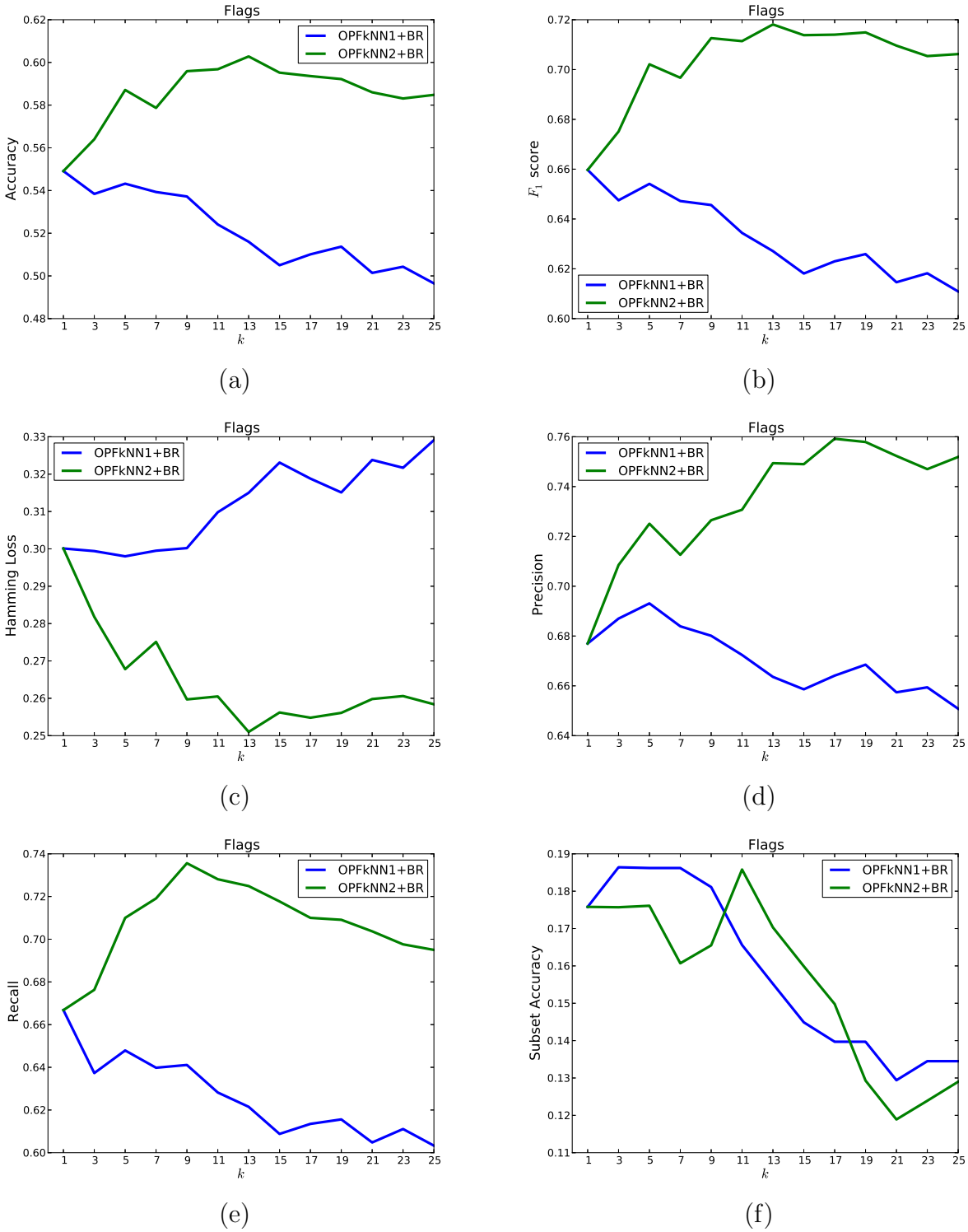


Figura 5.4: Curva de desempenho dos classificadores OPFkNN1+BR e OPFkNN2+BR na base *Flags* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

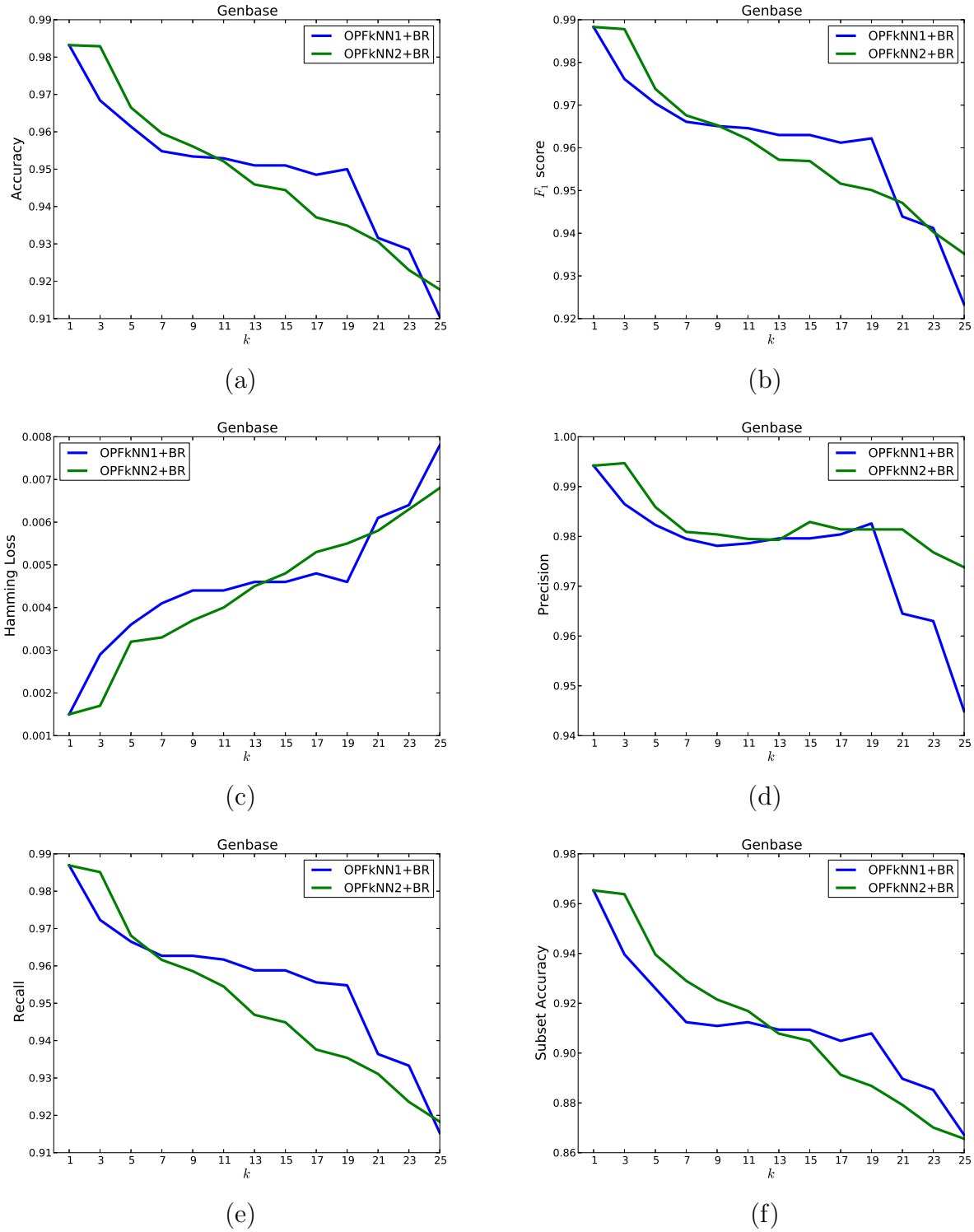


Figura 5.5: Curva de desempenho dos classificadores OPFkNN1+BR e OPFkNN2+BR na base *Genbase* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

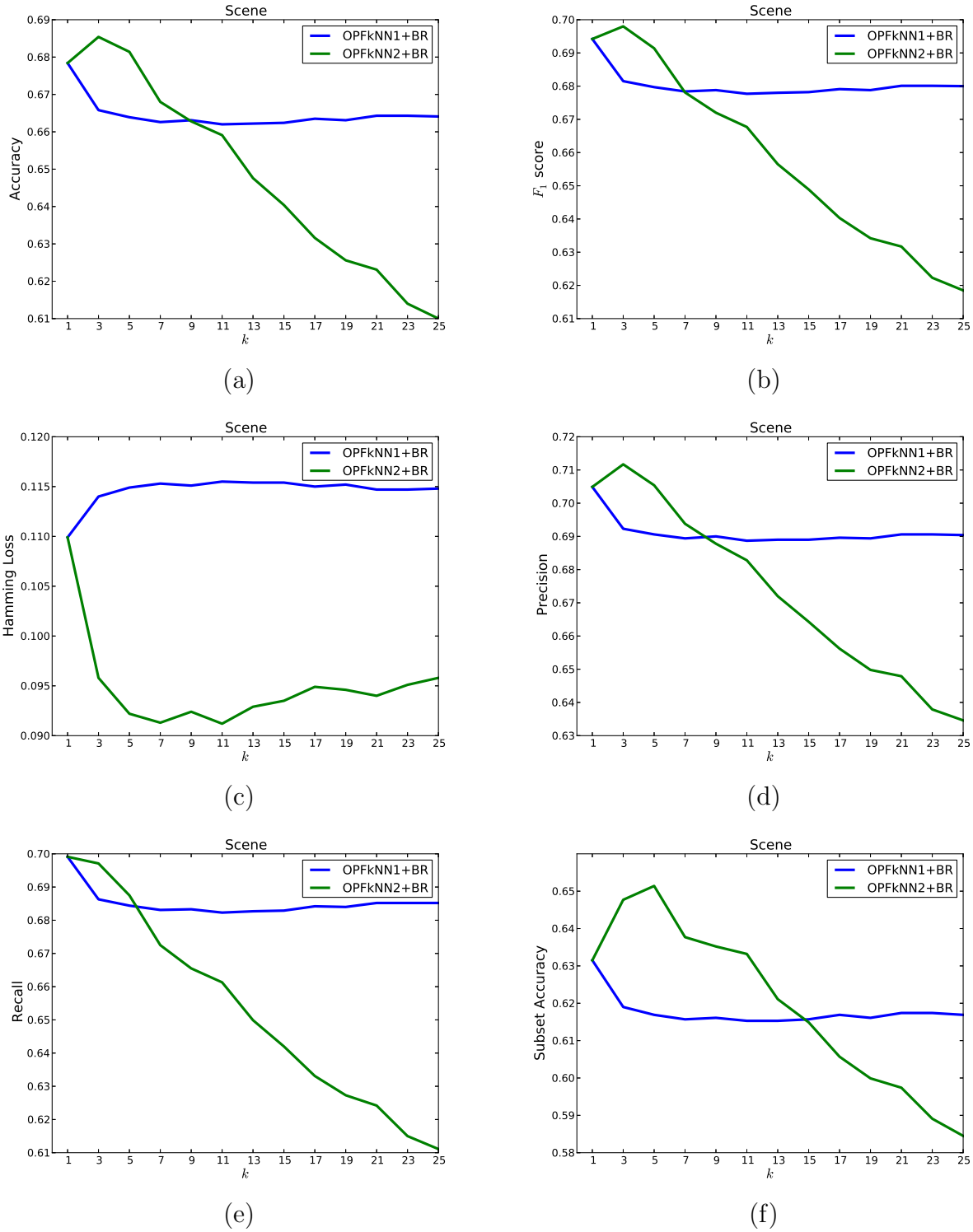


Figura 5.6: Curva de desempenho dos classificadores OPFkNN1+BR e OPFkNN2+BR na base *Scene* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

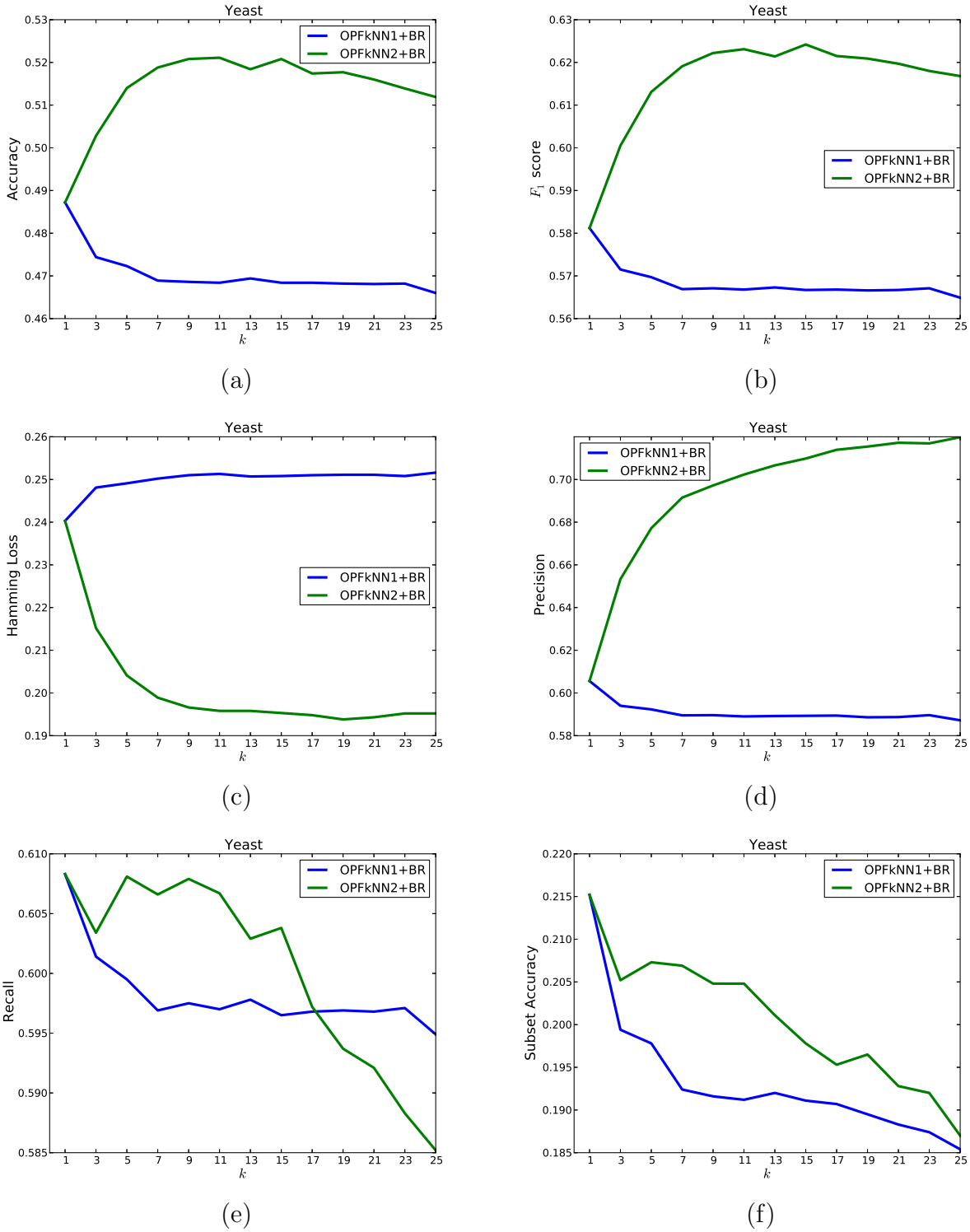


Figura 5.7: Curva de desempenho dos classificadores OPFkNN1+BR e OPFkNN2+BR na base *Yeast* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

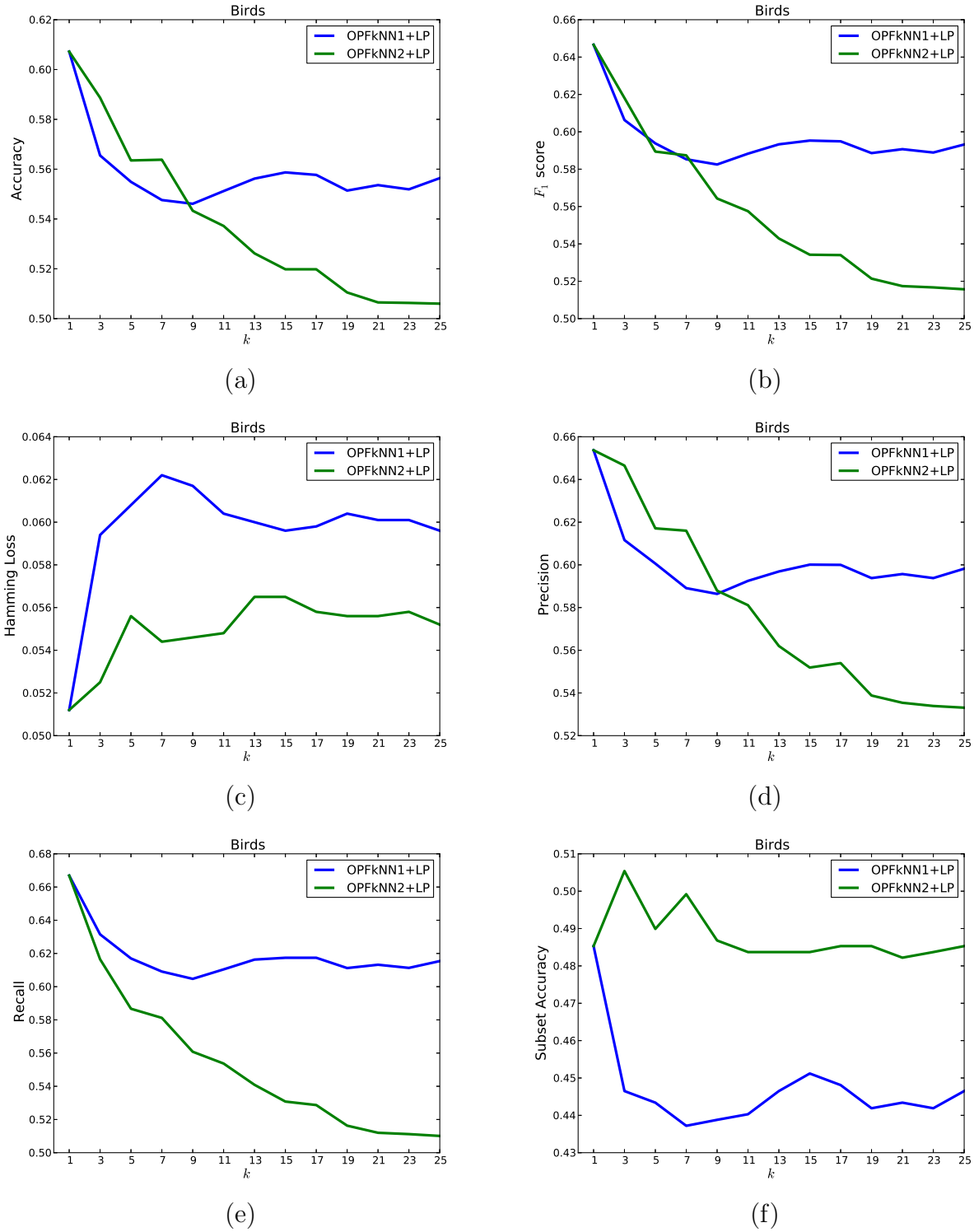


Figura 5.8: Curva de desempenho dos classificadores OPFkNN1+LP e OPFkNN2+LP na base *Birds* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

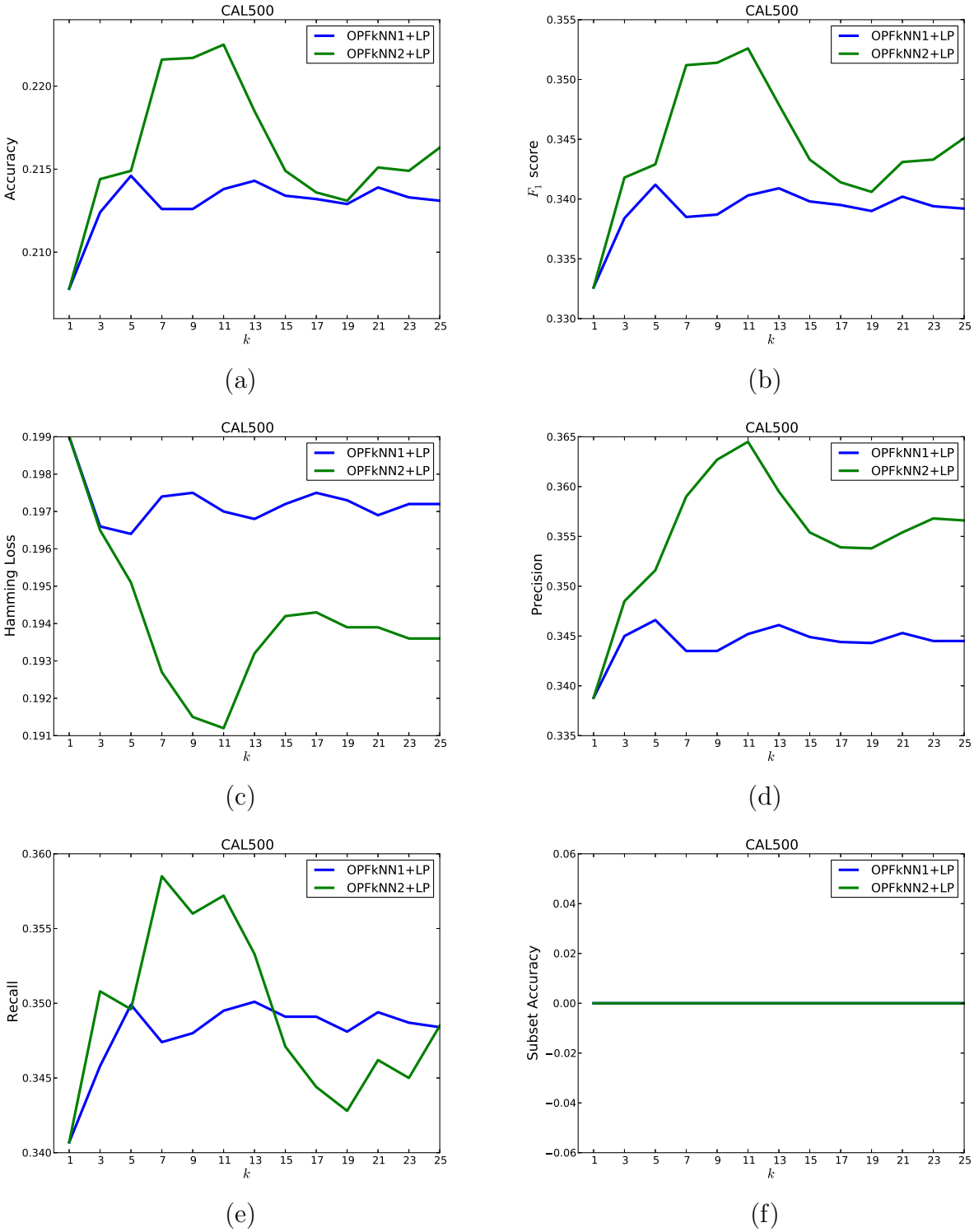


Figura 5.9: Curva de desempenho dos classificadores OPFkNN1+LP e OPFkNN2+LP na base *CAL500* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

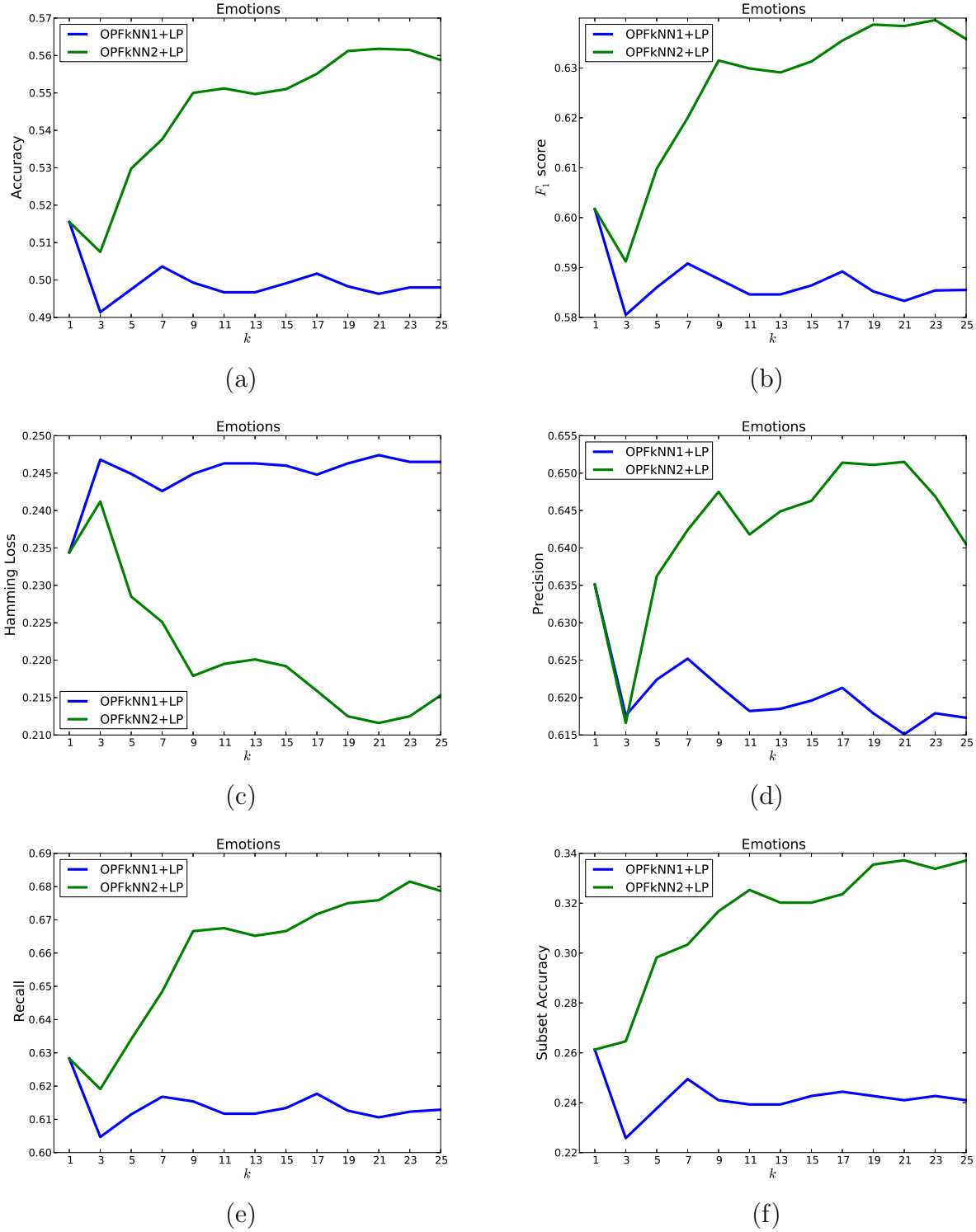


Figura 5.10: Curva de desempenho dos classificadores OPFkNN1+LP e OPFkNN2+LP na base *Emotions* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

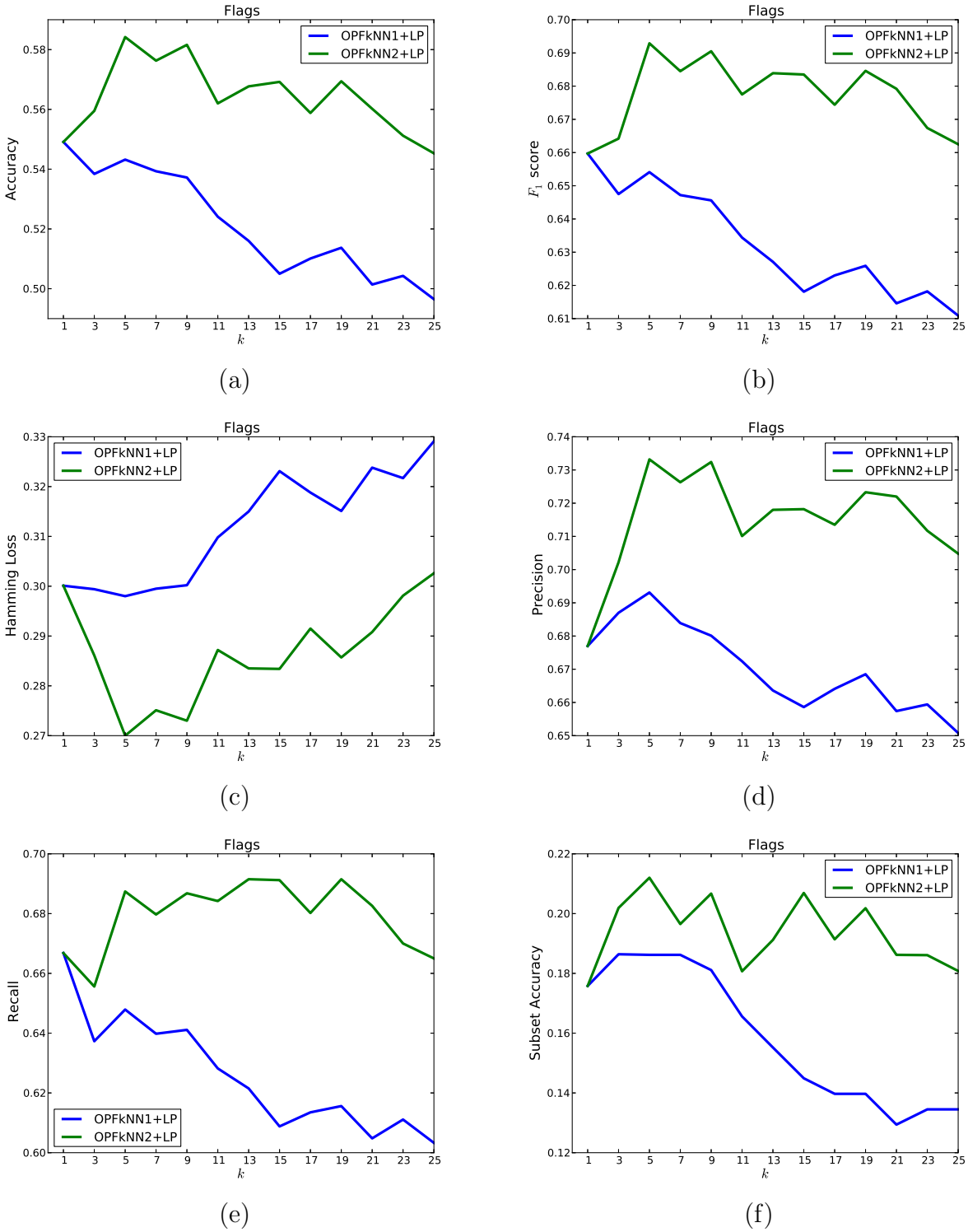


Figura 5.11: Curva de desempenho dos classificadores OPFkNN1+LP e OPFkNN2+LP na base *Flags* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

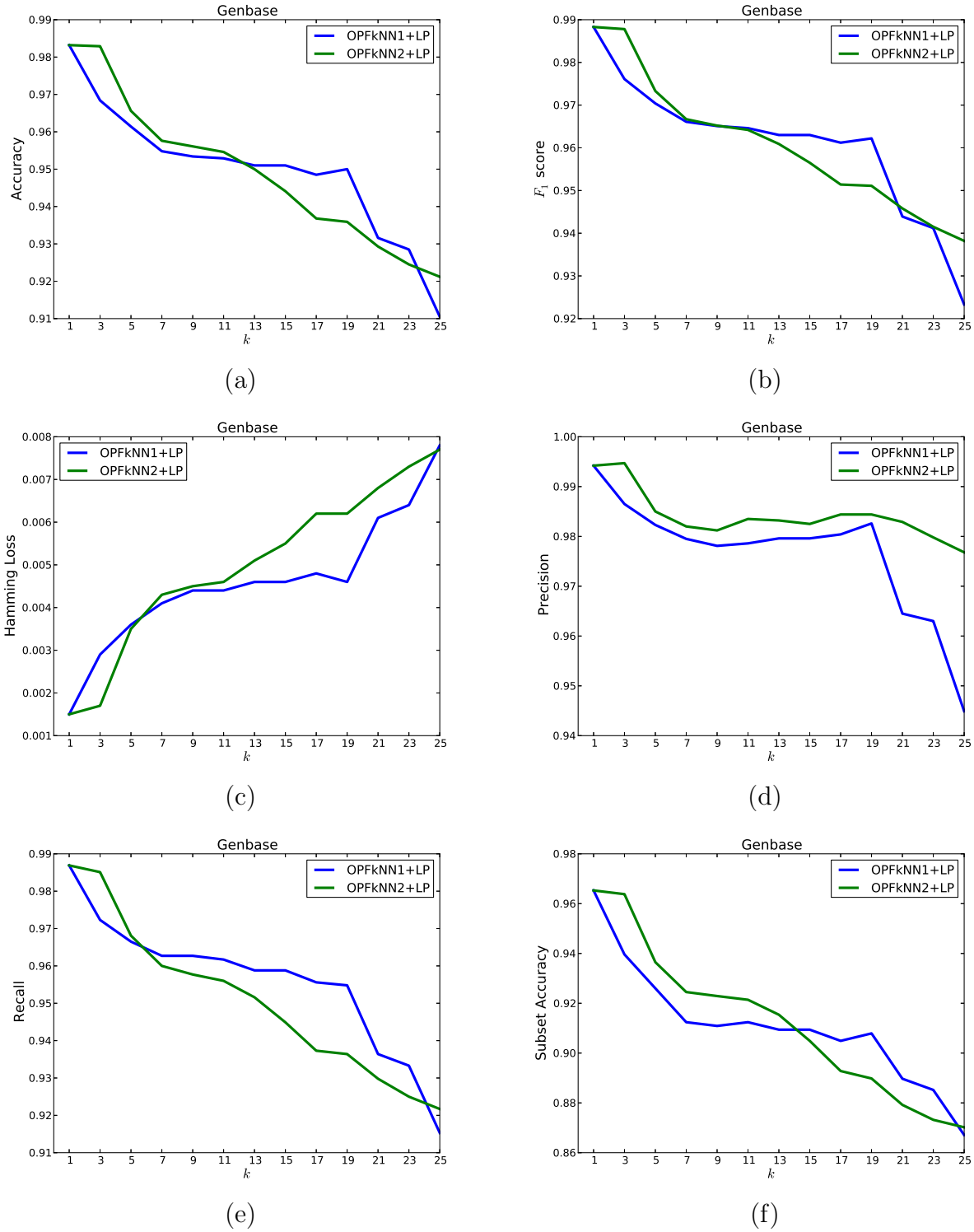


Figura 5.12: Curva de desempenho dos classificadores OPFkNN1+LP e OPFkNN2+LP na base *Genbase* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

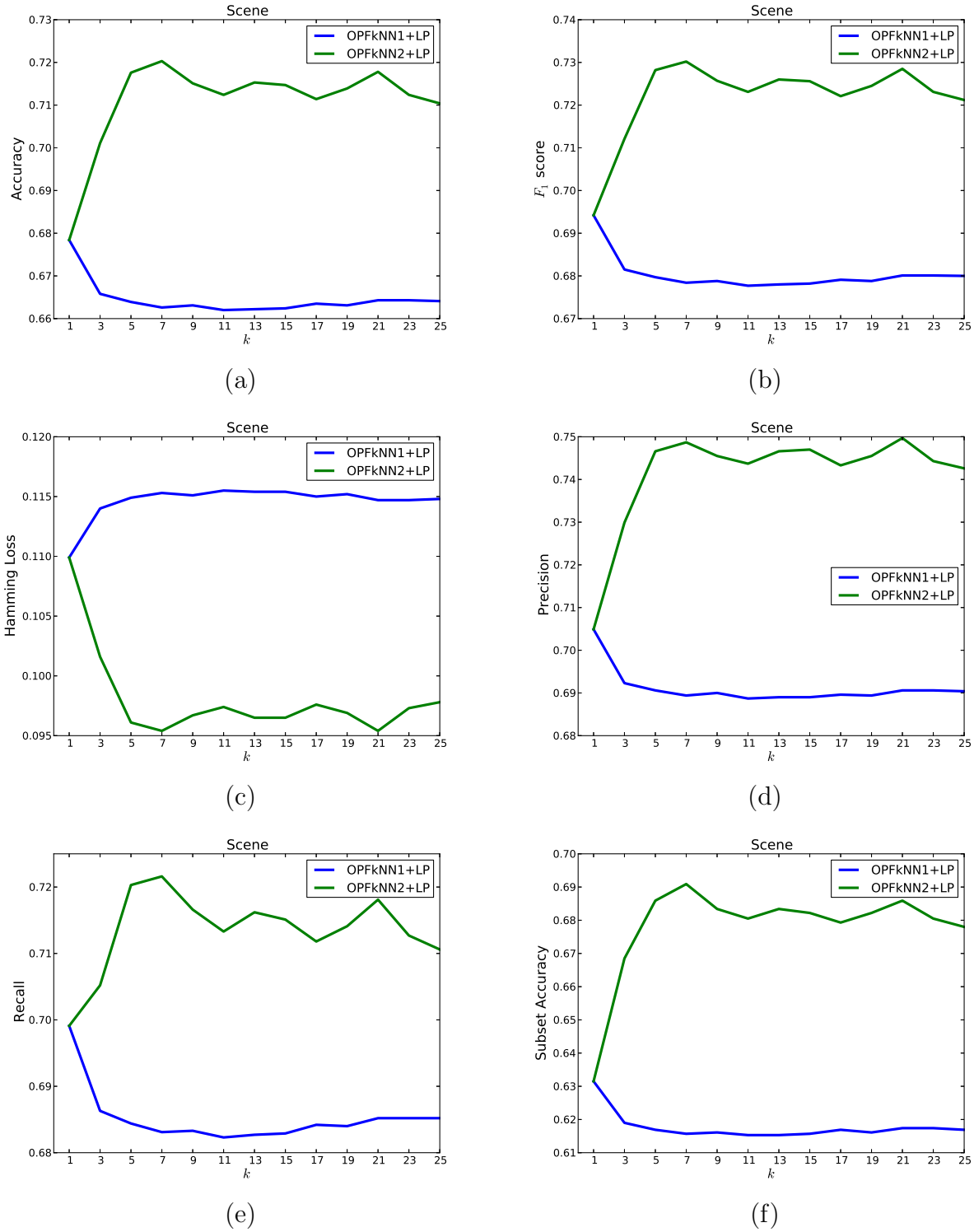


Figura 5.13: Curva de desempenho dos classificadores OPFkNN1+LP e OPFkNN2+LP na base *Scene* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

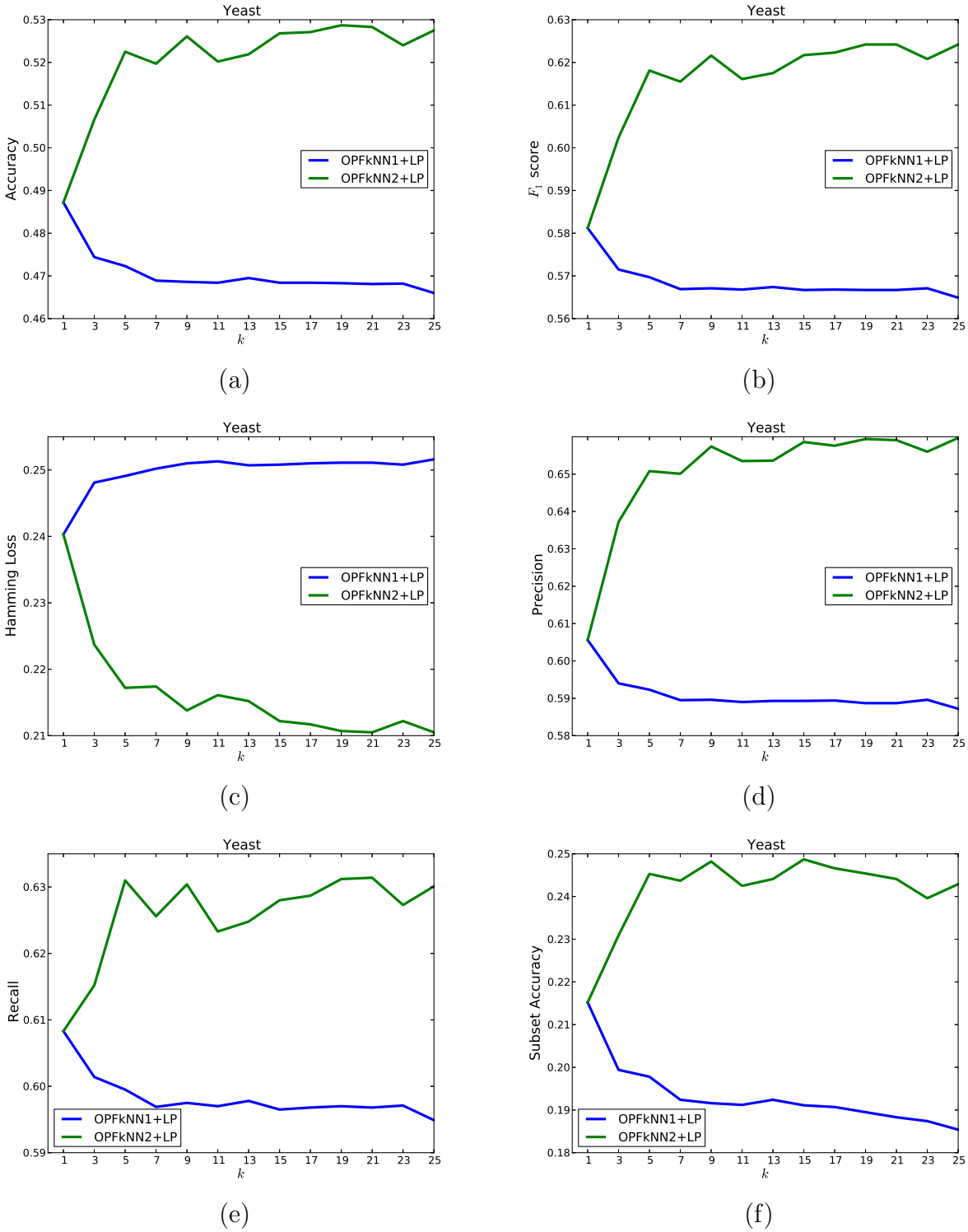


Figura 5.14: Curva de desempenho dos classificadores OPFkNN1+LP e OPFkNN2+LP na base *Yeast* variando o parâmetro k de 1 a 25 com passos de 2. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

5.2.2 Comparação entre classificadores

Nesta seção apresentamos os resultados obtidos na comparação entre os classificadores baseados em OPF e os classificadores J48, NB e SVM. Como o OPFkNN2 apresentou melhores resultados que o OPFkNN1, optamos por utilizá-lo como versão k-nn do OPF. Os resultados do OPFkNN2 apresentados nessa seção consideram o valor do parâmetro k que forneceu o melhor resultado em cada base de dados. As Tabelas 5.4 a 5.9 apresentam os resultados completos dos experimentos para cada uma das seis métricas, respectivamente, enquanto as Figuras 5.15 e 5.16 mostram os resultados dos testes estatísticos do desempenho dos classificadores utilizando os métodos BR e LP, respectivamente.

Como pode ser observado, o classificador OPFkNN2 foi ranqueado como o melhor técnica em todas as métricas e nos dois métodos, BR e LP. No entanto, em algumas métricas o teste de Friedman não apontou diferença estatística entre o OPFkNN2 e os outros classificadores. Isso pode ser justificado pelo desequilíbrio que os outros classificadores apresentaram entre as métricas *Precision* e *Recall*.

O classificador NB+BR, por exemplo, apresentou altas taxas de *Recall* (Tabela 5.8) nas bases *CAL500*, *Emotions*, *Scene* e *Yeast*. Contudo, baixos valores de *Precision* (Tabela 5.7) denotam alto número de falsos positivos, o que significa que o NB+BR gera muitos falsos positivos, o que diminui o número de falsos negativos e leva a altas taxas de *Recall*. Uma hipótese para esse comportamento é que, como o método BR utiliza a classificação binária e não considera a correlação entre os rótulos, o NB tenha que lidar com instâncias do conjunto de treinamento pertencentes a uma determinada distribuição de probalidade, devido ao rótulo dado pelo BR, mas que tenha maior probabilidade de pertencer a outra classe, dada a correlação com os outros rótulos não considerados pelo BR. Pode-se verificar ainda que esse desequilíbrio entre a *Precision* e a *Recall* não são tão evidentes no classificador NB+LP, dado que o método LP considera a correlação entre os rótulos. Os classificadores J48 e SVM, embora em escala menor, também têm dificuldades em manter o compromisso entre as métricas *Precision* e *Recall*.

Com exceção da base de dados *CAL500*, o classificador OPFkNN2 obteve desempenho superior ao OPF tradicional. No entanto, podemos verificar que na base de dados *CAL500*

o OPF obteve altas taxas de precision *Precision* associado a baixas taxas de *Recall*. Com isso, podemos concluir que o OPF inseriu muitos falsos negativos na classificação, o que interfiriu diretamente nas taxas de *Recall* e F_1 *score*. Essa análise demonstra que o OPFkNN2 tem maior capacidade de generalização nos problemas de múltiplos rótulos que o OPF. Uma justificativa para esse comportamento é que o OPFkNN2 considera o contexto das instâncias tanto no treinamento para a escolha dos protótipos, como na classificação para a propagação da classe com maior peso entre as k-instâncias vizinhas de uma instância de teste. Já OPF apresenta uma estrutura mais estrita na escolha dos protótipos: considera instâncias de classes diferentes conectadas ao longo da MST, sem considerar o contexto (vizinhança) que essas instâncias estão inseridas. Ainda, um protótipo no OPF pode propogar seu rótulo por um caminho mais longo, o que pode aumentar o erro no conjunto de treinamento, dado que a função de custo não restringe a propagação de rótulos entre amostras de conjuntos diferentes.

Tabela 5.4: Taxas de classificação segundo a métrica *Accuracy*.

Base	J48		NB		OPF		OPFkNN2		SVM	
	BR	LP	BR	LP	BR	LP	BR	LP	BR	LP
Birds	0.5687	0.5615	0.1389	0.5615	0.5943	0.6064	**0.6291**	**0.6072**	0.4872	0.458
CAL500	0.199	0.2168	0.2105	0.2055	**0.2244**	**0.2225**	0.2077	0.2074	0.2115	0.209
Emotions	0.4219	0.4235	0.5262	0.5157	0.5065	0.5147	**0.5411**	**0.5618**	0.4427	0.4938
Flags	**0.6068**	0.5519	0.5405	0.4839	0.5337	0.5491	0.6028	**0.5842**	0.529	0.5251
Genbase	0.9771	0.9803	0.2799	0.3636	0.9771	0.9835	0.9832	0.9832	**0.9874**	**0.985**
Scene	0.5341	0.5825	0.4519	0.6122	0.6724	0.6751	0.6854	0.7203	**0.7279**	**0.7876**
Yeast	0.4335	0.4098	0.4212	0.4716	0.4781	0.4858	0.5211	0.5283	**0.5333**	**0.5477**

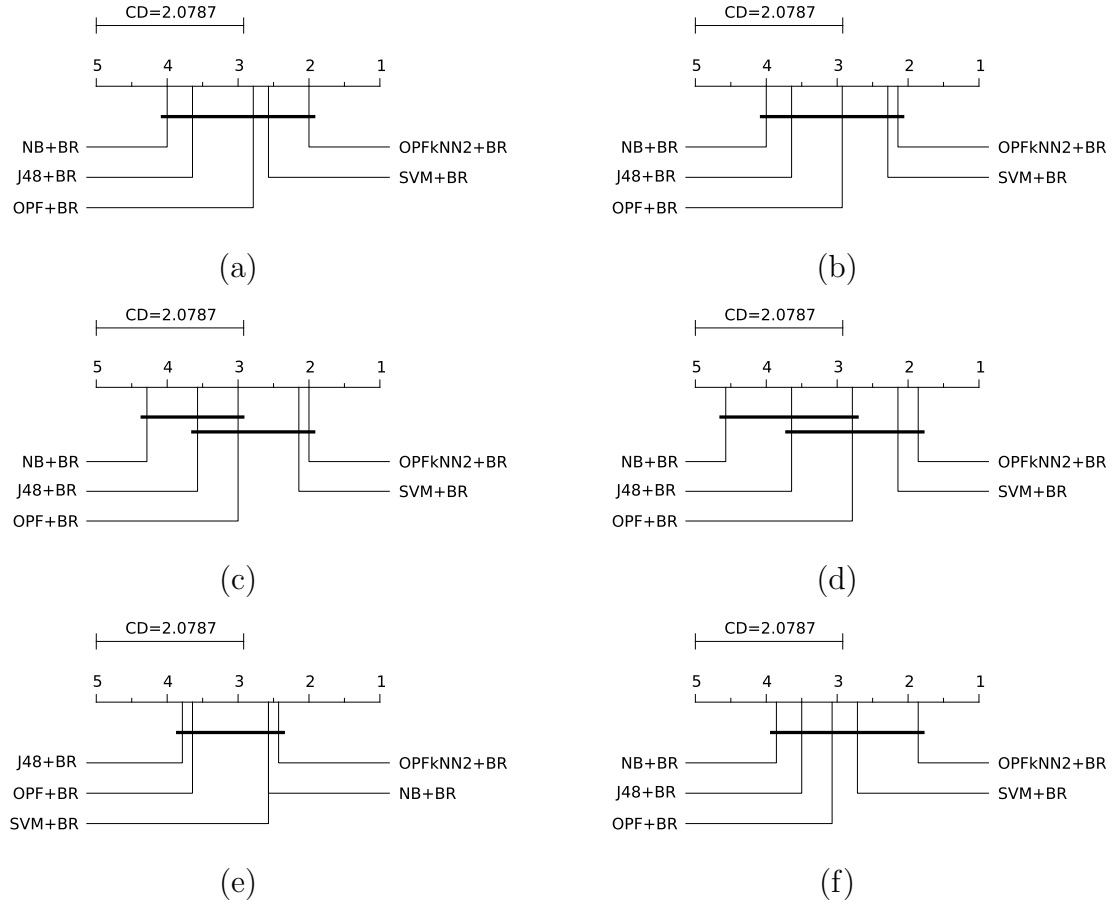


Figura 5.15: Diagramas de distância crítica para o teste de Nemenyi com $p = 0.10$. Classificadores utilizando o método BR para transformação das bases de dados. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

Tabela 5.5: Taxas de classificação segundo a métrica F_1 *score*.

Base	J48		NB		OPF		OPFkNN2		SVM	
	BR	LP	BR	LP	BR	LP	BR	LP	BR	LP
Birds	0.6017	0.593	0.1914	0.593	0.6351	0.6456	**0.6596**	**0.6466**	0.5812	0.461
CAL500	0.3224	0.3506	0.3367	0.3312	**0.357**	**0.3526**	0.3325	0.332	0.3382	0.34
Emotions	0.5085	0.5044	**0.6301**	0.6022	0.5936	0.6006	0.62	**0.6384**	0.5201	0.5695
Flags	**0.7208**	0.6626	0.6555	0.6049	0.6433	0.6597	0.7181	**0.6929**	0.6607	0.6459
Genbase	0.9818	0.9843	0.2863	0.3654	0.9818	0.9883	0.9883	0.9883	**0.9907**	**0.9896**
Scene	0.5725	0.5973	0.5656	0.6377	0.6887	0.6909	0.698	0.7302	**0.7443**	**0.7984**
Yeast	0.556	0.5114	0.5396	0.5687	0.5746	0.5802	0.6231	0.6242	**0.6351**	**0.6365**

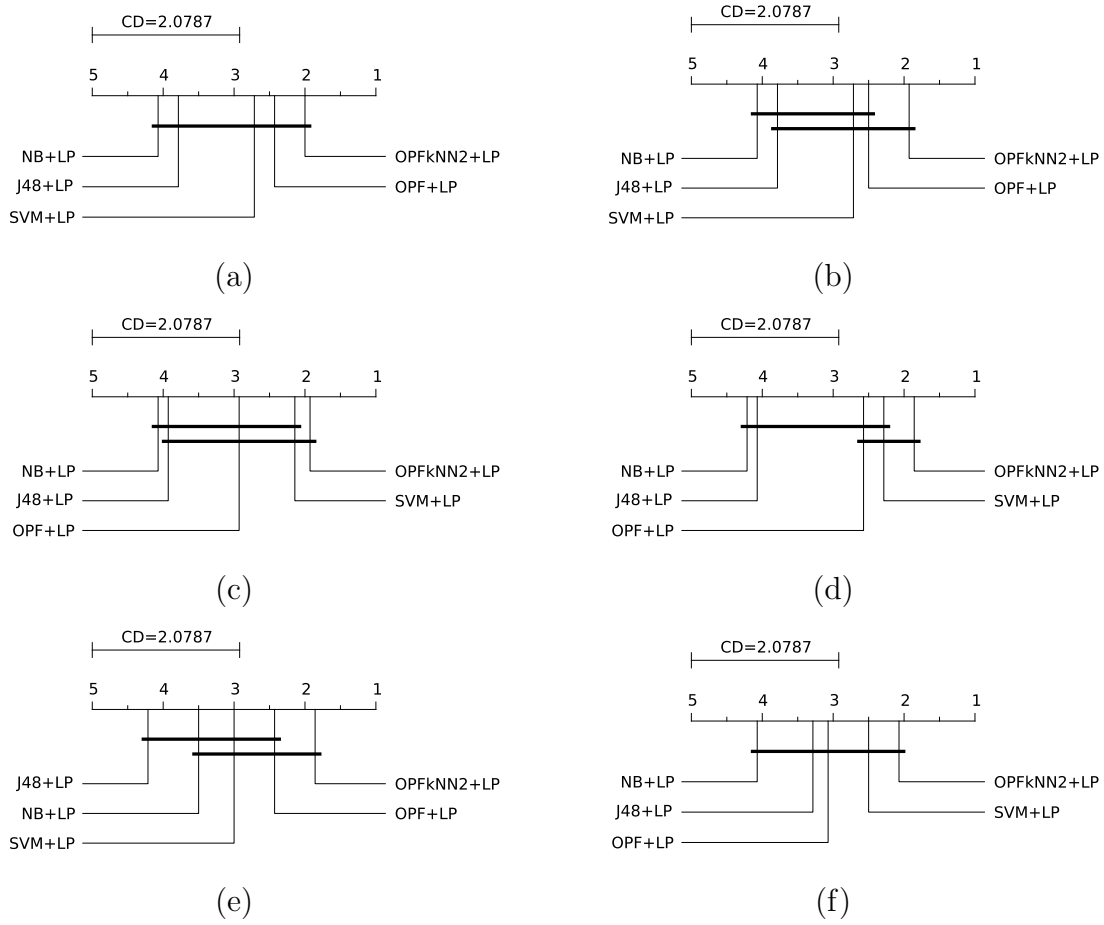


Figura 5.16: Diagramas de distância crítica para o teste de Nemenyi com $p = 0.10$. Classificadores utilizando o método LP para transformação das bases de dados. (a) *Accuracy*; (b) F_1 *score*; (c) *Hamming Loss*; (d) *Precision*; (e) *Recall*; e (f) *Subset Accuracy*.

Tabela 5.6: Taxas de classificação segundo a métrica *Hamming Loss*.

Base	J48		NB		OPF		OPFkNN2		SVM	
	BR	LP	BR	LP	BR	LP	BR	LP	BR	LP
Birds	0.0509	0.0595	0.3469	0.0595	0.0538	0.0513	**0.0435**	**0.0512**	0.2403	0.0547
CAL500	0.2014	0.1618	0.1971	0.3122	**0.1627**	0.1912	0.1988	0.2028	0.1957	**0.1386**
Emotions	0.2644	0.2824	0.2544	0.2308	0.2409	0.235	**0.1979**	**0.2116**	0.2291	0.2651
Flags	0.2532	0.2959	0.3031	0.3401	0.3089	0.3001	**0.251**	**0.27**	0.3071	0.3233
Genbase	0.0015	0.0021	0.0351	0.056	0.0015	0.0015	0.0015	0.0015	**0.0009**	**0.0013**
Scene	0.1347	0.1466	0.2421	0.1375	0.112	0.111	0.0958	0.0954	**0.0698**	**0.072**
Yeast	0.249	0.2784	0.3015	0.2399	0.2471	0.2409	0.1958	0.2105	**0.1854**	**0.1985**

Tabela 5.7: Taxas de classificação segundo a métrica *Precision*.

Base	J48		NB		OPF		OPFkNN2		SVM	
	BR	LP	BR	LP	BR	LP	BR	LP	BR	LP
Birds	0.6278	0.6036	0.1678	0.6036	0.6384	0.6522	**0.6871**	**0.6537**	0.6056	0.4698
CAL500	0.3304	0.4462	0.3437	0.2571	**0.4427**	0.3645	0.3389	0.3333	0.3471	**0.5983**
Emotions	0.5357	0.5369	0.5743	0.6277	0.6245	0.6334	**0.6736**	**0.6515**	0.5997	0.5629
Flags	0.7109	0.662	0.6546	0.6056	0.6649	0.677	**0.7494**	**0.7332**	0.6897	0.6736
Genbase	0.9864	0.9909	0.3052	0.3702	0.9864	0.9931	0.9942	0.9942	**0.9947**	**0.9958**
Scene	0.5518	0.6048	0.459	0.6271	0.6985	0.7016	0.7117	0.7487	**0.7511**	**0.8164**
Yeast	0.603	0.5382	0.5291	0.599	0.5958	0.6044	0.7023	0.6591	**0.7285**	**0.6802**

Tabela 5.8: Taxas de classificação segundo a métrica *Recall*.

Base	J48		NB		OPF		OPFkNN2		SVM	
	BR	LP	BR	LP	BR	LP	BR	LP	BR	LP
Birds	0.6034	0.6036	0.3377	0.6036	**0.6614**	0.6661	0.6583	**0.6669**	0.6083	0.458
CAL500	0.3293	0.2996	**0.3447**	**0.5149**	0.3123	0.3572	0.3404	0.3455	0.3435	0.2428
Emotions	0.5457	0.5242	**0.7698**	0.636	0.6227	0.6275	0.6276	**0.6759**	0.507	0.6219
Flags	**0.7572**	0.6651	0.662	0.6449	0.6447	0.6668	0.7249	**0.6874**	0.6733	0.6499
Genbase	0.9816	0.9815	0.2799	0.3636	0.9816	0.9874	0.9869	0.9869	**0.9897**	**0.9877**
Scene	0.6332	0.6036	**0.8561**	0.673	0.695	0.6958	0.6971	0.7216	0.7538	**0.7912**
Yeast	0.5711	0.5383	**0.6119**	0.5948	0.6053	0.6079	0.6067	0.6314	0.6085	**0.6367**

Tabela 5.9: Taxas de classificação segundo a métrica *Subset Accuracy*.

Base	J48		NB		OPF		OPFkNN2		SVM	
	BR	LP	BR	LP	BR	LP	BR	LP	BR	LP
Birds	0.4775	0.4698	0.0279	0.4698	0.4713	**0.4853**	**0.5395**	**0.4853**	0.2152	0.4512
CAL500	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Emotions	0.172	0.1905	0.2057	0.2663	0.2495	0.2613	**0.3067**	**0.3372**	0.2125	0.2799
Flags	0.1756	**0.2324**	**0.1914**	0.0617	0.1656	0.1758	0.1703	0.212	0.0667	0.1445
Genbase	0.9607	**0.9683**	0.266	0.3596	0.9607	0.9638	0.9653	0.9653	**0.9758**	0.9683
Scene	0.4225	0.5393	0.1695	0.5368	0.624	0.6282	0.6477	0.6909	**0.6789**	**0.7553**
Yeast	0.0662	0.1357	0.0985	0.1804	0.199	0.2127	**0.2048**	0.2441	0.1994	**0.2834**

6 Conclusões e trabalhos futuros

Problemas que apresentam múltiplos rótulos vem se tornando frequentes no contexto de reconhecimento de padrões. Dessa forma, muitos estudos tem concentrado esforços na busca por técnicas eficazes, dado que tais problemas apresentam uma complexidade maior que os tradicionais, já que uma instância pode estar associada a um conjuto de rótulos ao invés de apenas um rótulo, como acontece em problemas tradicionais. Assim, torna-se necessário insvestigar o desempenho de classificadores tradicionais aplicados a esses problemas, principalmente, aqueles que ainda não foram abordades nesse contexto, como é o caso do classificador OPF.

Nesse contexto, o presente trabalho apresentou um estudo sobre a aplicação do classificador OPF em problemas de mútliplos rótulos. Para isso, utilizamos dois métodos de tranformação de problemas: *Binary Relevance* e *Label Powerset*. Além disso, propusemos uma customização do classificador OPFkNN com o objetivo de melhorar os resultados do classificador baseando-nos nas característica dos métodos de tranformação de problemas. Realizamos experimentos para comparar a versão tradicional do OPFkNN (OPFkNN1) com o a nova versão proposta (OPFkNN2). Além disso, comparamos os classificadores baseados em OPF com outros três classificadores.

Os resultados dos experimentos demonstraram que o OPFkNN2 obteve desempenho superior ao OPFkNN1. Ainda, na comparação com os outros classificadores o OPFkNN2 também foi superior, mesmo quando comparado à versão do OPF que utiliza grafo completo. Pode-se destacar que, segundo os testes estatísticos, OPFkNN2 obteve desempenho similar ao SVM, o qual é o classificador estado-da-arte quando se utiliza métodos de tranformação de problemas. No entando, o OPFkNN2 mostrou-se mais completo, pois

apresentou maior compromisso na classificação de rótulos relevantes segundo as métricas *Precision* e *Recall*. Com isso, podemos concluir que o OPFkNN2 pode ser considerado como uma opção para classificação de múltiplos utilizando os métodos de transformação de problemas analisados nesta dissertação.

Embora os classificadores OPF tenham demonstrado um bom desempenho no contexto de múltiplos, torna-se importante avaliar esses classificadores associados a novas abordagens para múltiplos rótulos. Em futuras investigações, pretendemos observar os classificadores OPF combinados a outros classificadores na tentativa de melhorar a resposta do método *Binary Relevance*, dado que esse método de tranformação pode gerar configurações favoráveis a um classificador e desfavorável a outro. Também, pretendemos estudar modificações nos classificadores baseados em OPF que permitam a sua aplicação direta em problemas de múltiplos rótulos sem a utilização de métodos de transfomação de problemas.

7 Artigos Científicos

Neste capítulo, apresentamos os artigos científicos desenvolvidos:

1. Nakamura, R. Y. M; PEREIRA, L. A. M.; Silva, D.; CARDOZO, P.; PEREIRA, C. R.; FERASOLI FILHO, H.; ALVES, S.; PIRES, R. G.; SPADOTTO, A. A.; Papa, J. P. . Fast Robot Voice Interface through Optimum-Path Forest. In: 16th IEEE International Conference on Intelligent Engineering Systems, 2012, Lisboa. (sem Qualis)

2. Nakamura, R. Y. M ; PEREIRA, L. A. M. ; COSTA, K. A. P. ; RODRIGUES, D. ; Papa, J. P. ; YANG, X.-S.. BBA: A Binary Bat Algorithm for Feature Selection. In: SIBGRAPI 2012 - Conference on Graphics, Patterns and Images, 2012, Ouro Preto. (Qualis B1)

3. Nakamura, R. Y. M ; PEREIRA, L. A. M. ; COSTA, K. A. P. ; RODRIGUES, D. ; Papa, J. P. ; YANG, X.-S.. BBA: A Binary Bat Algorithm for Feature Selection. Swarm Intelligence (Elsevier), 2013. (Capítulo de livro).

4. COSTA, K. A. P.; PEREIRA, C. R.; Nakamura, R. Y. M; PEREIRA, L. A. M.; Papa, J. P.. Boosting Optimum-Path Forest Clustering Through Harmony Search and Its Applications for Intrusion Detection in Computer Networks. In: 8th International Conference On Information Assurance and Security, 2012, São Carlos. 8th International Conference On Information Assurance and Security, 2012. (Qualis B4)

5. PIRES, R. G.; PEREIRA, L. A. M.; Papa, J. P.. A Hybrid Image Restoration Algorithm Based on Projections Onto Convex Sets and Harmony Search. IEEE

International Symposium on Circuits and Systems. Beijing, China, 2013. (Qualis A1)

6. RODRIGUES, D.; PEREIRA, L. A. M. ;Almeida, T. N. S.; Papa, J. P.;Souza, A. N.; Ramos, C. C. O.; YANG, X.-S. BCS: A Binary Cuckoo Search Algorithm for Feature Selection. Beijing, China, 2013. (Qualis A1)

7. PEREIRA, L. A. M. ; Papa, J. P. ; ALMEIDA, J. ; TORRES, R. S. ; AMORIM, W. P. . A Multiple Labeling-based Optimum-Path Forest for Video Content Classification. In: SIBGRAPI 2013 The Conference on Graphics, Patterns, and Images, 2013, Arequipa. (aceito para publicação), 2013, Arequipa. SIBGRAPI 2013 The Conference on Graphics, Patterns, and Images, 2013, Arequipa. (Qualis B1)

8. PEREIRA, L. A. M. ; SOUZA, A. N. ; Papa, J. P. . Harmony Search applied for Support Vector Machines Training Optimization. In: IEEE Eurocon 2013, 2013, Zagreb. IEEE Eurocon 2013. (sem Qualis)

9. PEREIRA, L. A. M.; AFONSO, L. C. S.; Papa, J. P.; VALE, Zita; Ramos, C. C. O.; SILVA, T. F.; Souza, A. N.. Multilayer Perceptron Neural Networks Training Through Charged System Search and its Application for Non-Technical Losses Detection. IEEE PES Conference on Innovative Smart Grid Technologies. 2013. (sem Qualis)

10. PEREIRA, L. A. M. ; RODRIGUES, D.; Almeida, T. N. S.; Papa, J. P.;Souza, A. N.; Ramos, C. C. O.; YANG, X.-SA. Binary Cuckoo Search and its Application for Feature Selection. Swarm Intelligence and Bio-Inspired Computation. 2013. Elsevier. (Capítulo de Livro)

11. RODRIGUES, D. ; PEREIRA, L. A. M. ; Papa, J. P. ; RAMOS, C. O. ; SOUZA, A. N. ; PAPA, L. . Optimizing Feature Selection through Binary Charged System Search. In: CAIP 2013 : International Conference on Computer Analysis of Images and Patterns, 2013, York. CAIP 2013 : International Conference on Computer Analysis of Images and Patterns, (Qualis B2) 2013.

12. RODRIGUES, D. ; PEREIRA, L. A. M. ; Nakamura, R. Y. M ; COSTA, K. A. P. ; Papa, J. P. ; YANG, X.-S.. A Wrapper Approach for Feature Selection based on Bat Algorithm and Optimum-Path Forest. Expert Systems and Applications, 2013. (Qualis A1)

7.1 Artigos submetidos

1. PIRES, R. G.; PEREIRA, L. A. M.; Papa, J. P.. Projections onto Convex Sets Parameter Estimation through Evolutionary Optimization and its Application for Image Restoration. Natural computing. 2013. (Qualis B3)

2. PEREIRA, L. A. M. ; Souza, A. N.; Papa, J. P.; Tavares, J. M. R. S.. A Harmony Search-based to Estimate Parameters for Support Vector Machines Training. Expert Systems and Applications. 2013. (Primeira rodada de revisões).(Qualis A1)

3. PEREIRA, L. A. M. ; Afonso, L. C. S.; RODRIGUES, D.; Kuroda, M. C.; Vidal A. C.; Bueno, J. F.; Yang, X.S.; Papa, J. P. On the Training of Artificial Neural Networks through Metaheuristic Algorithms. Neurocomputing. 2014. (Qualis A1)

7.2 Artigos em revisão

1. COSTA, K. A. P.; PEREIRA, L. A. M.; PEREIRA, C. R.; Nakamura, R. Y. M; Papa, J. P.. A Nature-Inspired Framework to Speed Up Optimum-Path Forest Clustering and its Aplications for Intrusion Detection in Computer Networks. Information Science. 2012. (Qualis A1)

Referências

[1] S. Haykin. *Neural Networks: a comprehensive fundation*. Prentice Hall, 1994.

[2] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, 2 edition, 2000.

[3] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[4] T.-H. Chiang, H.-Y. Lo, and S.-D. Lin. A ranking-based knn approach for multi-label classification. In Steven C. H. Hoi and Wray L. Buntine, editors, *ACML*, volume 25 of *JMLR Proceedings*, pages 81–96. JMLR.org, 2012.

[5] R. E. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[6] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, April 2006.

[7] H. Blockeel, L. Schietgat, Jan Struyf, A. Clare, and S. Dzeroski. Hierarchical multilabel classification trees for gene function prediction (Extended abstract). In *Probabilistic Modeling and Machine Learning in Structural and Systems Biology*, pages 9–14. Helsinki University Printing House, 2006.

[8] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[9] T. Li and M. Ogihara. Detecting emotion in music. In *In Proceedings of the Fifth International Symposium on Music Information Retrieval*, pages 239–240, 2003.

[10] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[11] M.-L. Zhang and Z.-H. Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18:1338–1351, 2006.

[12] M.-L. Zhang and Z.-H. Zhou. A k-nearest neighbor based algorithm for multi-label classification. In Xiaohua Hu, Qing Liu, Andrzej Skowron, Tsau Young Lin, Ronald R. Yager, and Bo Zhang, editors, *Proceedings of the IEEE International Conference on Granular Computing*, pages 718–721. IEEE, 2005.

[13] A. Clare, A. Clare, and R.D. King. Knowledge discovery in multi-label phenotype data. In *In: Lecture Notes in Computer Science*, pages 42–53. Springer, 2001.

[14] J.P. Papa, A.X. Falcão, and C.T.N. Suzuki. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19:120–131, 2009.

[15] C.R. Pereira, R.Y.M. Nakamura, K.A.P. Costa, and J.P. Papa. An optimum-path forest framework for intrusion detection in computer networks. *Engineering Applications of Artificial Intelligence*, 25(6):1226 – 1234, 2012.

[16] L.A.M. Pereira, R.Y.M. Nakamura, G.F.S. De Souza, D. Martins, and J.P. Papa. Aquatic weed automatic classification using machine learning techniques. *Comput. Electron. Agric.*, 87:56–63, September 2012.

[17] C.C.O. Ramos, A.N. Souza, J.P. Papa, and A.X. Falcão. Fast non-technical losses identification through optimum-path forest. In *Proocedings of the 15th International Conference on Intelligent System Applications to Power Systems*, pages 1–5, 2009.

[18] J.P. Papa, A.X. Falcao, G.M. de Freitas, and A. de Avila. Robust pruning of training patterns for optimum-path forest classification applied to satellite-based rainfall occurrence estimation. *Geoscience and Remote Sensing Letters, IEEE*, 7(2):396–400, April 2010.

[19] J.A. Santos, A.T. Silva, R.S. Torres, A.X. Falcão, L.P. Magalhães, and R.A.C. Lamparelli. Interactive classification of remote sensing images by using optimum-path forest and genetic programming. In *Computer Analysis of Images and Patterns*, volume 6855 of *Lecture Notes in Computer Science*, pages 300–307. Springer Berlin Heidelberg, 2011.

[20] R.Y.M. Nakamura, L.M.G. Fonseca, J.A. Dos Santos, R.S. Torres, X.-S. Yang, and J.P. Papa. Nature-inspired framework for hyperspectral band selection. *Geoscience and Remote Sensing, IEEE Transactions on*, 52(4):2126–2137, April 2014.

[21] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. D¸eroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084 – 3104, 2012. Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011).

[22] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995. Springer-Verlag.

[23] X. Luo and A. Nur Zincir-Heywood. Evaluation of two systems on multi-class multi-label document classification. In Mohand-Said Hacid, Neil V. Murray, Zbigniew W.

Ras, and Shusaku Tsumoto, editors, *Proceedings of the 15th International Symposium on Foundations of Intelligent Systems*, volume 3488 of *Lecture Notes in Computer Science*, pages 161–169. Springer, 2005.

[24] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *Annual ACM Conference on Research and Development in Information Retrieval*, pages 274–281, 2005.

[25] S. Godbole, , and S. Sarawagi. Discriminative Methods for Multi-labeled Classification. *Advances in Knowledge Discovery and Data Mining*, pages 22–30, 2004.

[26] R. Cerri and A. C. P. L. F. Carvalho. Hierarchical multilabel classification using top-down label combination and artificial neural networks. In *Proceedings of the 2010 Eleventh Brazilian Symposium on Neural Networks*, pages 253–258, Washington, DC, USA, 2010. IEEE Computer Society.

[27] R. Cerri and A. C. P. L. F. Carvalho. New top-down methods using SVMs for hierarchical multilabel classification problems. In *Proceedings of the International Joint Conference on Neural Networks*, pages 3064–3071, Washington, DC, USA, 2010. IEEE Computer Society, Los Alamitos.

[28] Y. P. Wang and T. Pavlidis. Optimal correspondence of string subsequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1080–1087, 1990.

[29] A.X. Falcão, J. Stolfi, and R.A. Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):19–29, 2004.

[30] C. Allène, J.Y. Audibert, M. Couprie, J. Cousty, and R. Keriven. Some links between min-cuts, optimal spanning forests and watersheds. In *Proc. of the 8th International Symposium on Mathematical Morphology*, pages 253–264, 2007.

[31] R.A. Lotufo and A.X. Falcão. The ordered queue and the optimality of the watershed approaches. In *Procedings of the 8th International Symposium on Mathematical Morphology*, volume 18, pages 341–350. Kluwer, 2000.

[32] J.P. Papa and A.X. Falcão. A new variant of the optimum-path forest classifier. *International Symposium on Visual Computing*, 47(1):935–944, 2008.

[33] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[34] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

[35] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[36] P. Nemenyi. *Distribution-free Multiple Comparisons*. Princeton University, 1963.

[37] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, December 2006.

Autorizo a reprodução xerográfica para fins de pesquisa.

São José do Rio Preto, 25/03/2014

Assinatura