PUL-SSC: Aprendizado baseado em uma única classe com agrupamento semissupervisionado

Shih Ting Ju

PUL-SSC: Aprendizado baseado em uma única classe com agrupamento semissupervisionado

Shih Ting Ju

Orientador: *Prof. Dr. Bruno Magalhães Nogueira*
Coorientador: *Prof. Dr. Rafael Geraldeli Rossi*

Dissertação apresentada à Faculdade de Computação – FACOM-UFMS como parte dos requisitos necessários à obtenção do título de Mestra em Ciência da Computação.

FACOM - UFMS
Agosto/2021

Abstract

The large amount of data currently available is a source for extracting information for commercial and academic purposes. One approach for extracting knowledge on such bases that has gained prominence is one-class classification (OCC). The use of OCC in classifying whether an example is of a specific class is appropriate in datasets where the classes are unbalanced or where only the data of the class of interest are present during the training. Several OCC algorithms found in the literature use unsupervised clustering to delimit the border of the class of interest. These algorithms present competitive results with those presented by other OCC algorithms. Although semi-supervised learning has shown the possibility to achieve better results in several areas than with unsupervised, semi-supervised clustering is still little explored for OCC. One approach for OCC is Positive and Unlabeled Learning (PUL), in which learning occurs only with positive (interest) and unlabeled data. PUL algorithms seek to find a delimitation of the positive class. This master's degree project proposes a new algorithm PUL-SSC (Positive and Unlabeled Learning with Semi-Supervised Clustering) that learns the delimitation of the class of interest by creating and using must-link and cannot-link restrictions, clustering data with semi-supervised algorithm and a transductive learning process for label propagation. Two widely used semi-supervised clustering algorithms were employed: PCK-Means and MPCK-Means. In our experimental evaluation, semi-supervised algorithms outperformed the k-Means based algorithm and one-class SVM (OC-SVM) in most of the scenarios. In particular, the distance-based algorithm MPCK-Means was dominant in most of the comparisons using numerical and textual databases.

Resumo

A grande quantidade de dados disponíveis atualmente é uma fonte de extração de informações para fins comerciais e acadêmicos. Uma abordagem para extrair conhecimento em bases de dados que ganhou destaque é a classificação de uma única classe (em inglês, *One-class Classification* - OCC). O uso de OCC para classificar se um exemplo é de uma classe específica é apropriado em conjuntos de dados em que as classes são desbalanceadas ou apenas os dados da classe de interesse estão presentes durante o treinamento. Vários algoritmos de OCC encontrados na literatura utilizam agrupamento não supervisionado para delimitar a fronteira da classe de interesse. Esses algoritmos conseguem ter resultados competitivos com aqueles apresentados por outros algoritmos de OCC. Embora o aprendizado semissupervisionado tenha mostrado a possibilidade de alcançar melhores resultados em várias áreas do que com o agrupamento semissupervisionado, o agrupamento semissupervisionado ainda é pouco explorado para OCC. Uma abordagem para OCC é o *Positive and Unlabeled Learning* (PUL), em que o aprendizado ocorre apenas com dados positivos (interesse) e não rotulados. Os algoritmos de PUL procuram encontrar uma delimitação da classe positiva. Este trabalho de mestrado propõe um novo algoritmo PUL-SSC (*Positive and Unlabeled Learning with Semi-Supervised Clustering*) que aprende a delimitar a classe de interesse através da criação e utilização de restrições *must-link* e *cannot-link*, agrupamento de dados com algoritmo semisupervisionado e um processo de aprendizado transdutivo para propagação de rótulos. Foram explorados dois algoritmos de agrupamento semissupervisionados amplamente usados: PCK-Means e MPCK-Means. Na avaliação experimental, os algoritmos semissupervisionados superaram o algoritmo baseado em k-Means e o SVM de uma classe (OC-SVM) na maioria dos cenários. Em particular, o algoritmo baseado em distância MPCK-Means foi dominante na maioria das comparações usando conjuntos de dados numéricos e textuais.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

BOW *Bag of Words*

FN False Negative

FP False Positive

KNN *K-Nearest Neighbor*

K-NND *K-Nearest Neighbor Density-based*

K-NNRD *K-Nearest Neighbors Relative Density-based*

MC *Mapping Convergence*

MPCK-Means *Metric Pairwise Constrained K-means*

OC-SVM *One-class Support Vector Machine*

OCC *One-Class Classification*

PCK-Means *Pairwise Constrained K-Means*

PUL *Positive and Unlabeled Learning*

PUL-SSC *Positive and Unlabeled Learning with Semi-Supervised Clustering*

SVM *Support Vector Machine*

SVMC *Support Vector Mapping Convergence*

TN True Negative

TP True Positive

Lista de Algoritmos

CAPÍTULO

1

Introdução

A abordagem de classificação de única classe (em inglês, *One-class classification* - OCC) [Moya et al., 1993] é utilizada em contextos em que uma única classe está presente ou em que a classe negativa está subamostrada. OCC é um caso especial de classificação multiclasse, uma das classes, referenciada como classe positiva ou classe de interesse, é bem caracterizada pelos exemplos no conjunto de dados de treinamento. Para as demais classes, possui-se nenhum ou poucos exemplos, que não formam uma amostra representativa dos dados negativos (não fazem parte da classe de interesse). Neste sentido, Tax [2002] afirma que a tarefa em OCC é definir um limiar ou superfície de decisão para os exemplos positivos, que possa incluir o máximo de exemplos positivos, enquanto minimiza as chances de incluir um exemplo negativo.

Uma das abordagens para solucionar a classificação de uma única classe, chamada de *Positive and Unlabeled Learning* (PUL), em que o aprendizado ocorre apenas com dados positivos e não rotulados, e busca-se encontrar essa delimitação da classe positiva [Khan and Madden, 2009]. Essa subárea vem ganhando atenção nos últimos anos conforme os seguintes *surveys*: Jaskie and Spanias [2019] e Bekker and Davis [2020], em que diversos pesquisadores têm formulado diferentes abordagens algorítmicas a fim de obter uma melhor delimitação da classe positiva. Em vários domínios de aplicação, é difícil e custoso obter amostras balanceadas de cada classe, sendo importante usar OCC na obtenção de resultados de classificação melhores. Um exemplo disso é classificar o tipo de um texto em um site de discussão de tópicos variados, é difícil saber todos os tipos de tópicos existentes e rotular todos os textos. Além disso, os tipos de tópicos variam com o tempo e novos tópicos emergem. Fazer rotulação dos dados é uma tarefa, muitas vezes, manual e demorada.

E, saber de antemão quais são as classes existentes num conjunto de dados é difícil e às vezes impossível.

A abordagem de OCC foi explorada em diversos artigos, sendo a grande maioria dos algoritmos baseados em Máquina de Vetores de Suporte (em inglês, *Support Vector Machines* - SVM) [Tax and Duin, 1999a,b, Schölkopf et al., 2000, Manevitz and Yousef, 2001, Yu et al., 2003, 2004, Yu, 2005, Muñoz-Marí et al., 2010]. Outras abordagens de OCC usaram algoritmos de classificação, regressão, redes neurais, métodos estatísticos, métodos probabilísticos ou algoritmos genéticos [Moya et al., 1993, Ridder, 1998, Letouzey et al., 2000, Liu et al., 2002, Leng et al., 2015, Erfani et al., 2016, Ruff et al., 2018, Perera and Patel, 2019].

O uso de algoritmos de agrupamento aplicados a OCC ainda é pouco explorado [Perera et al., 2021]. Krawczyk et al. [2014] conseguiram obter resultados usualmente melhores do que os métodos tradicionais de OCC, através do uso de algoritmos de agrupamento não supervisionado em conjunto com um classificador OSVM ou OSVM ponderado. Em Gôlo et al. [2019], o algoritmo baseado no k-Means obteve as melhores avaliações experimentais em base de dados textuais, ao comparar diversos algoritmos de OCC de diferentes categorias (baseados em similaridade, probabilísticos e estatísticos).

Na literatura, diversos algoritmos de agrupamento semissupervisionado conseguiram obter em geral, resultados melhores que os algoritmos de agrupamento não supervisionado, através do aprendizado de métricas e a inclusão de conhecimento dos especialistas em forma de restrições, conforme Basu et al. [2002] e Bilenko et al. [2004]. Segundo Nogueira [2013], algoritmos semissupervisionados podem atingir uma melhora significativa na qualidade de agrupamento sem aumentar o custo computacional dos métodos na maioria dos casos, quando comparada a algoritmos de agrupamento não supervisionado.

Desta forma, no problema de agrupamento é possível obter resultados melhores com o uso de algoritmos semissupervisionados ao invés de não supervisionados [Nogueira, 2013]. Além disso, o uso do agrupamento semissupervisionado permite encontrar grupos de formatos variados, através da distorção do espaço de representações. Porém, de acordo com a revisão bibliográfica realizada, abordagens baseadas em agrupamento semissupervisionado aplicado ao problema de PUL ainda são pouco explorados [Bekker and Davis, 2020, Jaskie and Spanias, 2019] e há trabalhos [Gôlo et al., 2019, Krawczyk et al., 2014] que já apresentaram o uso de agrupamento não supervisionado com resultados melhores que demais abordagens. O uso de agrupamento semissupervisionado para PUL é uma lacuna a ser explorada.

1.1 Objetivo

O objetivo é propor um novo algoritmo de agrupamento semissupervisionado de dados para o PUL, que utiliza restrições *must-link* e *cannot-link*, juntamente com um processo de propagação de rótulos para aprender a delimitar a classe de interesse com poucos dados rotulados.

1.2 Hipóteses

A hipóteses deste trabalho são:

- O uso de algoritmos de agrupamento semissupervisionado em classificação de única classe tem resultados melhores do que abordagens não supervisionadas já exploradas em OCC [Gôlo et al., 2019].
- O aprendizado de métricas e a inclusão de conhecimento dos especialistas em forma de restrições melhora o processo de agrupamento e, consequentemente, os resultados de PUL.

1.3 Organização

Esta dissertação está organizada em mais cinco capítulos:

- Capítulo 2: é apresentado o aprendizado semissupervisionado e o agrupamento semissupervisionado, com ênfase em alguns algoritmos baseados em agrupamento particional que foram a base do algoritmo proposto. Além disso, algumas medidas de avaliação de agrupamentos são abordadas.
- Capítulo 3: é introduzido o problema de OCC, cujo objetivo é extrair padrões dos dados rotulados e inferir alguma regra para classificar os dados não rotulados como sendo da classe de interesse ou não. Neste capítulo, são apresentados alguns tipos de abordagens, algoritmos e domínios de aplicações encontrados na literatura.
- Capítulo 4: a abordagem proposta é descrita, sua formulação e funcionamento são ilustrados.
- Capítulo 5: são apresentados os experimentos realizados e os resultados obtidos com 13 conjuntos de dados, sendo 8 desses bases numéricas e 3 bases textuais, avaliando os algoritmos k-Means, PCK-Means e MPCK-Means, e comparando com a *baseline* OC-SVM.

- Capítulo 6: são apresentadas as principais contribuições deste trabalho, as limitações e direcionamentos para trabalhos futuros.

CAPÍTULO

2

Agrupamento Semissupervisionado

O agrupamento não supervisionado particiona um conjunto de dados de forma que os dados dentro de um agrupamento sejam similares entre si, e dados em agrupamentos diferentes sejam dissimilares. Para isso, dados não rotulados são utilizados para fazer esse particionamento, sem intervenção ou conhecimento adicional. Alguns trabalhos de agrupamento não supervisionado estão focados em encontrar um critério de similaridade que auxilie no bom particionamento desses dados [Basu et al., 2008]. O agrupamento semissupervisionado tem como objetivo melhorar os resultados do agrupamento não supervisionado através de informações adicionais, como restrições.

Segundo Nogueira [2013], algoritmos semissupervisionados podem atingir uma melhora significativa na qualidade de agrupamento sem aumentar o custo computacional dos métodos na maioria dos casos, quando comparada a algoritmos de agrupamento não supervisionado. Essa abordagem permite adicionar informações ao modelo sem ter que saber o rótulo dos dados. Apesar de ser mais fácil fornecer restrições do que rótulos, restrições fornecem menos informação do que rótulos. Mesmo assim, com poucas restrições o algoritmo de agrupamento semissupervisionado consegue ter uma performance boa comparado a algoritmos de agrupamento não supervisionado e supervisionado [Bilenko et al., 2004]. Como pode ser visto em alguns trabalhos da literatura, em que os autores usaram esses dados adicionais para geração de restrições [Basu et al., 2002, Wagstaff et al., 2001] ou para melhorar o aprendizado de métricas de particionamento [Xing et al., 2003, Bilenko et al., 2004]. A evolução desses algoritmos e suas variações é descrita em Bair [2013]. Além disso, existem outros métodos de agrupamento semissupervisionados na literatura [Van Engelen and Hoos, 2020, Ahmad and Khan, 2019], entretanto,

este capítulo está focado nos algoritmos de particionamento citados anteriormente, que são utilizados nos experimentos.

Este capítulo está organizado da seguinte forma: na Seção 2.1, o agrupamento semissupervisionado é abordado de maneira a apresentar os principais conceitos, dando ênfase a algoritmos que são as bases deste trabalho; e depois, na Seção 2.2, são abordadas as medidas de avaliação de agrupamentos; por fim, Seção 2.3, as considerações finais.

2.1 Agrupamento Semissupervisionado

Aprendizado semissupervisionado é um paradigma de aprendizagem relacionado ao estudo de como os computadores e sistemas naturais, como os humanos, aprendem na presença de dados rotulados e não rotulados [Zhu and Goldberg, 2009].

Tradicionalmente, o aprendizado tem sido estudado no paradigma não supervisionado (por exemplo, *clustering*), onde todos os dados não possuem rótulo, ou no paradigma supervisionado (por exemplo, classificação, regressão), onde todos os dados são rotulados. O objetivo do aprendizado semissupervisionado é entender como a combinação de dados rotulados e não rotulados pode alterar o comportamento do aprendizado e projetar algoritmos que tiram proveito dessa combinação.

Algoritmos de agrupamento semisupervisionados têm como objetivo melhorar os resultados de agrupamento com supervisão limitada. O conjunto de treino consiste de instâncias não rotuladas e algumas informações supervisionadas sobre os agrupamentos. Por exemplo, essas informações podem ser chamadas de restrições *must-link*, em que duas instâncias devem estar num mesmo agrupamento; e restrições *cannot-link*, em que não podem estar no mesmo agrupamento Wagstaff et al. [2001].

Os métodos de agrupamento semissupervisionados são utilizados para incorporar o conhecimento de domínio de um especialista ao processo de agrupamento, possibilitando, assim, que os grupos formados atendam melhor às expectativas. Existem diversos tipos de algoritmos de agrupamento semissupervisionado disponíveis na literatura conforme os *surveys*: Van Engelen and Hoos [2020], Ahmad and Khan [2019].

Os algoritmos bases deste trabalho são variações de k-Means para agrupamento semissupervisionado como: *COP-KMeans* [Wagstaff et al., 2001], *Seeded-KMeans* [Basu et al., 2002], *Constrained-KMeans* [Basu et al., 2002], *Pairwise Constrained K-Means* (*PCK-Means*) [Basu et al., 2004] e *Metric Pairwise Constrained K-Means* (*MPCK-Means*) [Bilenko et al., 2004].

O k-Means é um algoritmo de agrupamento não supervisionado baseado

na iterativa relocação que particiona um conjunto de dados em k agrupamentos, minimizando localmente o erro quadrático entre os dados e os centroides dos agrupamentos. Para um conjunto de dados $X = \{x_1, ..., x_n\}$, k-Means cria k-partições $\{X_j\}_{j=1}^{k}$ de X, em que $\{\mu_1, ..., \mu_k\}$ representam os centroides das k partições, com a seguinte função objetivo conforme Equação 2.1:

$$EQ = \sum_{j=1}^{k} \sum_{x_i \in X_j} \| x_i - \mu_j \|^2 \tag{2.1}$$

Iterativamente, cada instância de dados $x_i \in X$ é alocado ao *cluster* cuja distância entre o centroide do *cluster* e x_i é a menor comparada aos demais *clusters*. E, cada *cluster* tem seu centroide μ_j atualizado com o valor médio das instâncias que o constitui.

2.1.1 COP-KMeans

O algoritmo de agrupamento particional *COP-KMeans* [Wagstaff et al., 2001], é uma variação do *K-Means* que utiliza restrições pareadas, da forma *must-link* e *cannot-link*, para auxiliar no processo de agrupamento de dados conforme Algoritmo 1. O fecho transitivo das restrições é utilizado para aumentar o número de restrições existentes. Isso é feito das seguintes maneiras: se existe uma restrição *must-link* entre x_a e x_b, e outra *must-link* entre x_b e x_c, então a restrição *must-link* entre x_a e x_c é incorporada; e se tem uma restrição *must-link* entre x_a e x_b e uma restrição *cannot-link* entre x_b e x_c, então a restrição *cannot-link* entre x_a e x_c é incorporada. O conjunto de restrições resultante é utilizada no *COP-KMeans*.

Esse algoritmo segue os mesmos passos do *K-Means*, exceto pela etapa de verificação das restrições, em que antes de atribuir uma instância de dados a um determinado *cluster*, os conjuntos de restrições *must-link* e *cannot-link* são verificados. Neste algoritmo, as restrições não podem ser quebradas, caso alguma restrição não possa ser respeitada, o *COP-KMeans* retorna um agrupamento vazio e ocorre falha no processo de agrupamento.

2.1.2 Seeded-KMeans

O *Seeded-KMeans* é um algoritmo de agrupamento proposto por Basu et al. [2002] que utiliza conjunto de *seeds* iniciais para guiar o k-Means conforme Algoritmo 2. Cada conjunto de *seeds* iniciais é formado por exemplos que pertencem a um mesmo *cluster*. Para cada *cluster*, esse *seed* inicial é informado pelo especialista, e esses *seeds* são utilizados para calcular os centroides iniciais do k-Means. Assume-se que, para cada partição de X, há normalmente pelo menos um *seedpoint*, isto é, existe pelo menos uma instância de dados

Algoritmo 1: *COP-KMeans*

Entrada: $X = \{x_1, x_2, \cdots, x_n\}$: conjunto de dados;
k: número de *clusters*;
$M = \{(x_a, x_b), \cdots, (x_y, x_z)\}$: conjunto de restrições *must-link* pareadas de X;
$C = \{(x_i, x_j), \cdots, (x_t, x_u)\}$: conjunto de restrições *cannot-link* pareadas de X.
Saída: k partições $\{X_j\}_{j=1}^{k}$ disjuntas de elementos pertencentes a X tal que a função objetivo de *K-Means* é otimizada.

início
  Inicialize os k centroides $\mu = \{\mu_1, \mu_2, ..., \mu_k\}$ aleatoriamente
  enquanto *não atingir o critério de convergência* **faça**
    para cada $x_i \in X$ **faça**
      associe x_i ao *cluster* h_j, tal que $h_j = argmin_h \|x_i - \mu_h\|^2$
      e $\nexists x_r \in h_j$, tal que $(x_i, x_r) \in C$
      e $\nexists x_r \notin h_j$, tal que $(x_i, x_r) \in M$
    fim
    para cada *cluster* $h_j \in h$ **faça**
      calcule o novo centroide do *cluster*, tal que $\mu_{h_j} = \frac{1}{|h_j|} \sum_{x \in h_j} x$
    fim
  fim
fim

inicial para cada *cluster*.

2.1.3 Constrained-KMeans

Constrained-KMeans, proposto por Basu et al. [2002], é uma variação do *Seeded-KMeans*. Similar ao *Seeded-KMeans*, os conjuntos de *seeds* iniciais são utilizados para criar os centroides iniciais. Entretanto, a atribuição de *cluster* dos dados dos *seeds* iniciais são mantidos inalterados e apenas os rótulos dos dados que não fazem parte dos *seeds* iniciais são reestimados conforme Algoritmo 3.

Constrained-KMeans inicializa o algoritmo *k*-Means com os rótulos dos dados especificados pelo usuário e mantém esses rótulos ao longo do algoritmo. Esse algoritmo é apropriado quando os *seeds* iniciais são livres de ruídos, ou se o usuário não quer que o rótulo dos dados dos *seeds* mudem.

2.1.4 Metric K-Means (MK-Means)

Metric K-Means (MK-Means) [Xing et al., 2003] propõe aprendizado de métricas como um problema de otimização convexa. O foco do algoritmo é aprender métricas de distância que consigam capturar a noção de similaridade entre os dados através de uma matriz que faz ponderação dos dados de entrada. Seja S um conjunto de pares de X em que $(x_i, x_j) \in S$ se forem similares, a fórmula da métrica de distância é definida na Equação 2.2. Esse critério de

Algoritmo 2: *Seeded-KMeans*

Entrada: $X = \{x_1, x_2, \cdots, x_n\}$: conjunto de dados;
k: número de *clusters*;
$S = \cup_{j=1}^{k} S_j$ de *seeds* iniciais.
Saída: k partições disjuntas de elementos pertencentes a X tal que a função objetivo de *K-Means* é otimizada.

```
início
    para cada seed S_j ∈ S faça
        calcular os centroides iniciais, tal que μ_j = (1/|S_j|) Σ_{x∈S_j} x
    fim
    enquanto não atingir o critério de convergência faça
        para cada x_i ∈ X faça
            associe x_i ao cluster h_j, tal que h_j = argmin_h ||x_i − μ_h||^2
        fim
        para cada cluster h_j ∈ h faça
            calcule o novo centroide do cluster, tal que
            μ_{h_j} = (1/|h_j|) Σ_{x∈h_j} x
        fim
    fim
fim
```

similaridade que define os pares de S pode ser associada às restrições *must-link* definidas em *COP-KMeans*.

$$d(x_i, x_j) = \|x_i - x_j\|_{\mathbf{A}} = \sqrt{(x_i - x_j)^T \mathbf{A}(x_i - x_j)} \tag{2.2}$$

Configurar a matriz A igual a matriz identidade é equivalente a usar uma distância euclidiana. Restringir A como uma matriz diagonal corresponde a aprender métricas em que para cada eixo (atributo) é dado pesos diferentes. De forma geral, A parametriza uma família de distâncias de Mahalanobis. Aprender essa métrica é equivalente a encontrar um redimensionamento dos dados que substitui cada ponto x por $A^{1/2}x$ e aplicar distância euclidiana sobre os dados redimensionados [Xing et al., 2003]. Conforme a Figura 2.1, (b) e (c) mostram o resultado de $A^{1/2}x$, em que a métrica consegue distorcer o espaço dimensional, a fim de deixar os pontos similares mais próximos, enquanto mantém pontos dissimilares distantes.

A matriz A é aprendida através da minimização das distâncias entre os pontos de S, e atualizada através de iterações de descida gradiente. Os autores propõem o uso dessa métrica substituindo o uso da distância euclidiana no *K-Means* ou no *COP-KMeans*.

Algoritmo 3: *Constrained-KMeans*

Entrada: $X = \{x_1, x_2, \cdots, x_n\}$: conjunto de dados;
k: número de *clusters*;
$S = \cup_{j=1}^{k} S_j$ de *seeds* iniciais.
Saída: k partições disjuntas de elementos pertencentes a X tal que a função objetivo de K-Means é otimizada.

```
início
    para cada seed S_j ∈ S faça
        calcular os centroides iniciais, tal que μ_{h_j} = (1/|S_j|) Σ_{x∈S_j} x
    fim
    para cada x_i ∈ S faça
        associe x_i ao cluster h_j, tal que x_i ∈ S_j
    fim
    enquanto não atingir o critério de convergência faça
        para cada x_i ∉ S faça
            associe x_i ao cluster h_j, tal que h_j = argmin_h ||x_i − μ_h||^2
        fim
        para cada cluster h_j ∈ h faça
            calcule o novo centroide do cluster, tal que
            μ_{h_j} = (1/|h_j|) Σ_{x∈h_j} x
        fim
    fim
fim
```

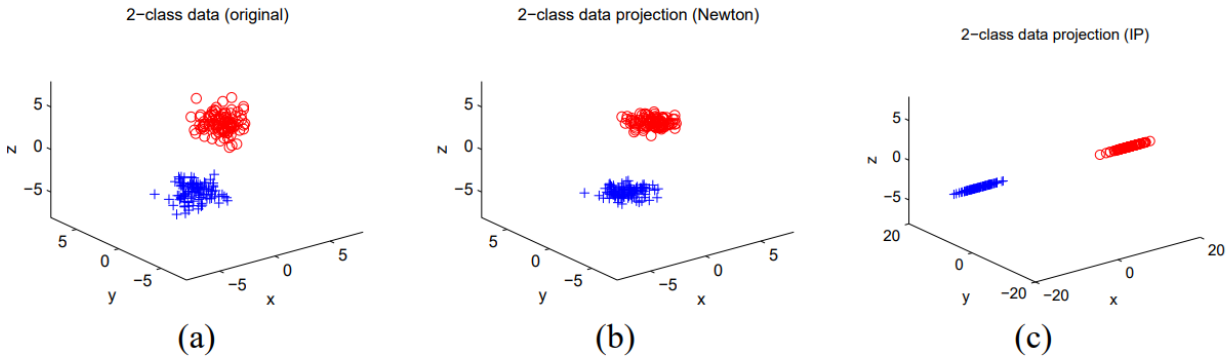
2.1.5 Pairwise Constrained K-Means (PCK-Means)

Pairwise Constrained K-Means (PCK-Means) [Basu et al., 2004] é um algoritmo baseado no *K-Means* que utiliza restrições de emparelhamento *must-link* e *cannot-link* com um custo associado à violação de cada restrição.

O peso associado à violação das restrições é representado por $W = \{w_{ij}\}$ e $\overline{W} = \{\overline{w}_{ij}\}$, que são respectivamente de, violação de restrições *must-link* e *cannot-link*. Seja l_i o *cluster* atribuído ao ponto x_i, em que $l_i \in \{h\}_{h=1}^{k}$, o custo de violar uma restrição *must-link* $(x_i, x_j) \in M$ é dado pela função $w_{ij}\mathbb{1}[l_i \neq l_j]$, isto é, se duas instâncias de dados com restrição *must-link* são atribuídas a *clusters* diferentes, o custo w_{ij} é aplicado. Similarmente, o custo de violar uma restrição *cannot-link* $(x_i, x_j) \in C$ é dado por $\overline{w}_{ij}\mathbb{1}[l_i = l_j]$, isto é, se duas instâncias de dados possuem restrição *cannot-link* entre elas, e são atribuídas ao mesmo *cluster*, o custo $\overline{w}_{ij}$ é aplicado. Note que $\mathbb{1}$ é a função indicadora, com $\mathbb{1}[verdadeiro] = 1$ e $\mathbb{1}[falso] = 0$.

Através desse modelo, o problema é formulado como a minimização da função objetivo, onde o ponto x_i é atribuído à partição X_{l_i}, com o centróide μ_{l_i} conforme Equação 2.3:

Figura 2.1: (a) Dados originais, cada classe é representada por símbolos e cores diferentes. (b) Redimensionamento dos dados com a matriz diagonal A aprendida. (c) Redimensionamento com matriz A completa.



Fonte: Extraído de Xing et al. [2003, p. 5].

$$EQ = \frac{1}{2}\sum_{x_i \in X} \|x_i - \mu_{l_i}\|^2 \quad + \sum_{(x_i,x_j)\in M} w_{ij} \quad \mathbb{1}[l_i \neq l_j] \quad + \sum_{(x_i,x_j)\in C} \overline{w}_{ij} \quad \mathbb{1}[l_i = l_j] \tag{2.3}$$

No passo de inicialização, é utilizado um fecho transitivo das restrições *must-link* e ampliado esse conjunto M com as restrições transitivas. Esse conjunto aumentado de restrições *must-link* é utilizado para gerar λ componentes conectados de M, que são utilizados para criar as $\{N_p\}_{p=1}^{\lambda}$ vizinhanças iniciais.

As λ vizinhanças são utilizadas para inicializar os centroides dos *clusters*. Se $\lambda \geq k$, em que k é o número de *clusters* requeridos, é selecionado k vizinhanças de maior tamanho e inicializado os centroides dos k *clusters* com os centroides dessas vizinhanças. Se, $\lambda < k$, é inicializado λ centros com os centroides dessas λ vizinhanças. É buscado uma instância x que está conectado por restrições *cannot-link* a cada vizinhança, e é inicializado o $(\lambda+1)$-ésimo *cluster* com ela. Se tiver mais *clusters* não inicializados, eles são inicializados com perturbações randômicas do centroide global de X.

O algoritmo *PCK-Means* alterna entre atribuição de *cluster* e estimação de centroide conforme Algoritmo 4. No passo de atribuição de *cluster*, cada ponto $x \in X$ é atribuído ao *cluster* que minimiza a distância euclidiana entre x e o centroide do *cluster* e o custo aplicado às violações das restrições. A ordem em que os exemplos são atribuídos aos *clusters* influencia quais restrições de M e C são violadas, e consequentemente, na atribuição de *clusters* dos próximos pontos, desta forma, no algoritmo, é considerado uma ordem aleatória. A reestimação dos centroides é feita de forma igual ao *K-Means*.

Algoritmo 4: *Pairwise Constrained K-Means (PCK-Means)*

Entrada: $X = \{x_1, x_2, \cdots, x_n\}$: conjunto de dados;
k: número de *clusters*;
$M = \{(x_a, x_b), \cdots, (x_y, x_z)\}$: conjunto de restrições *must-link* pareadas de X;
$C = \{(x_i, x_j), \cdots, (x_t, x_u)\}$: conjunto de restrições *cannot-link* pareadas de X;
W: conjunto de custos associados às violações de restrições *must-link*;
$\overline{W}$: conjunto de custos associados às violações de restrições *cannot-link*.
Saída: k partições disjuntas de elementos pertencentes a X tal que a função objetivo de *K-Means* é otimizada.

início
  Crie um conjunto de λ vizinhanças $N = \{N_p\}_{p=1}^{\lambda}$ a partir dos conjuntos M e C e ordene os índices p por ordem decrescente do tamanho de N_p
  se $\lambda \geq k$ **então**
    inicialize os k centroides $\mu = \{\mu_1, \mu_2, \cdots, \mu_k\}$ com os centroides das vizinhanças $\{N_p\}_{p=1}^{k}$
  fim
  senão se $\lambda < k$ **então**
    inicialize os k centroides $\mu = \{\mu_1, \mu_2, \cdots, \mu_k\}$ com os centroides das λ vizinhanças $N = \{N_1, N_2, \cdots, N_\lambda\}$
    se $\exists$ *x que possui restrição cannot-link em relação a todas as vizinhanças de N* **então**
      inicialize $\mu_{\lambda+1}$ com x
    fim
    inicialize os centroides restantes aleatoriamente
  fim
  enquanto *não atingir o critério de convergência* **faça**
    para cada $x_i \in X$ **faça**
      associe x_i ao *cluster* h_z, tal que

$$h_z = argmin_h \left(\frac{1}{2} \|x_i - \mu_h\|^2 + \sum_{(x_i, x_j) \in M} \omega_{ij} \mathbb{1}[l_i \neq l_j] + \sum_{(x_i, x_j) \in C} \overline{\omega}_{ij} \mathbb{1}[l_i = l_j] \right)$$

    fim
    para cada *cluster* $h_j \in h$ **faça**
      calcule o novo centroide do *cluster*, tal que $\mu_{hj} = \frac{1}{|h_j|} \sum_{x \in h_j} x$
    fim
  fim
fim

2.1.6 Metric Pairwise Constrained K-Means (MPCK-Means)

O algoritmo *MPCK-Means*, proposto por Bilenko et al. [2004], é baseado no aprendizado de métricas de *MK-Means* e no uso de penalização da violação de restrições de *PCK-Means*. A inicialização das vizinhanças é feita de forma similar ao descrito em *PCK-Means*, exceto que, no caso do número de vizinhanças encontradas ser maior que o número de *clusters*, $\lambda > k$, uma variação do algoritmo *farthest-first traversal* [Gonzalez, 1985] é utilizada para que k vizinhanças sejam utilizadas para gerar os k centroides iniciais. E, se $\lambda \leq k$, λ vizinhanças são utilizadas e os $k - \lambda$ centroides restantes são inicializados de forma randômica.

No *MPCK-Means*, a métrica de distância entre os pontos e os centroides é definida na Equação 2.4, seja A a matriz de simetria. Com isso, há dois casos possíveis: (i) a matriz A é diagonal; ou (ii) a matriz A é cheia. Quando A é diagonal, ela faz uma ponderação das características dos pontos de entrada, e quando ela é cheia, características novas são criadas a partir de combinações lineares das originais [Bilenko et al., 2004].

$$||x_i - \mu_{l_i}||_{\mathbf{A}} = \sqrt{(x_i - \mu_{l_i})^T \mathbf{A}(x_i - \mu_{l_i})} \tag{2.4}$$

Para cada *cluster* l_i, uma matriz de pesos A_{l_i} é aprendida, desta forma a função objetivo de *MPCK-Means* envolve minimizar a Equação 2.5:

$$EQ = \sum_{x_i \in X} \left(||x_i - \mu_{l_i}||^2_{\mathbf{A}_{l_i}} - log(det(\mathbf{A}_{l_i})) \right) \tag{2.5}$$

A informação agregada às restrições é incorporada ao aprendizado, de maneira a penalizar mais as restrições *must-link* violadas entre pontos distantes comparada a pontos próximos. Como a violação envolve dois *clusters*, a penalização da violação de restrições *must-link* deve afetar as métricas de ambos *clusters* de acordo com a Equação 2.6:

$$f_M(x_i, x_j) = \frac{1}{2}||x_i - x_j||^2_{\mathbf{A}_{l_i}} + \frac{1}{2}||x_i - x_j||^2_{\mathbf{A}_{l_j}} \tag{2.6}$$

De forma análoga, a penalidade de violar uma restrição *cannot-link* entre dois pontos próximos deve ser maior que entre dois pontos distantes. Logo, a função de penalização *cannot-link* é definida na Equação 2.7. Em que, (x'_{l_i}, x''_{l_i}) é o par de pontos maximamente separados no conjunto de dados de acordo com a l_i-ésima métrica. Isso garante que a penalidade seja não negativa, já que o segundo termo nunca é maior que o primeiro.

$$f_C(x_i, x_j) = ||x'_{l_i} - x''_{l_i}||^2_{\mathbf{A}_{l_i}} - ||x_i - x_j||^2_{\mathbf{A}_{l_i}} \tag{2.7}$$

O problema resume-se a minimizar a função objetivo resultante da combinação das Equações 2.5, 2.6 e 2.7. O algoritmo *MPCK-Means* alterna entre atribuição de *cluster*, estimação de centroide e atualização de métricas das matrizes $A = \{A_1, A_2, \cdots, A_k\}$, conforme Algoritmo 5. No passo de atribuição de *cluster*, cada ponto $x \in X$ é atribuído ao *cluster* h_z, ao qual possui menor valor combinado de distância com o centroide e penalização de violação de restrições do *cluster*, conforme Equação 2.8. A ordem de atribuição de *cluster* para cada ponto é feita de ordem aleatória. A reestimação dos centroides é feita de forma igual ao *K-Means* e a atualização das matrizes A é definida na Equação 2.9, seja A_{h_z} a matriz do cluster h_z.

$$h_z = argmin_h \Bigg(\Big(\|x_i - \mu_h\|^2_{\mathbf{A}_h} - log\,(det\,(\mathbf{A}_h)) \Big) + \sum_{(x_i,x_j)\in M} w_{ij} f_M(x_i,x_j) \mathbb{1}[l_i \neq l_j] + \sum_{(x_i,x_j)\in C} \overline{w}_{ij} f_C(x_i,x_j) \mathbb{1}[l_i = l_j] \Bigg) \tag{2.8}$$

$$\mathbf{A}_{h_z} = |h_z| \cdot \Bigg(\sum_{x_i \in h_z} (x_i - \mu_{h_z})(x_i - \mu_{h_z})^T + \sum_{(x_i,x_j)\in M_{h_z}} \frac{1}{2} \omega_{ij} (x_i - x_j)(x_i - x_j)^T \mathbb{1}[l_i \neq l_j] + \sum_{(x_i,x_j)\in C_{h_z}} \overline{\omega}_{ij} \Big((x'_i - x''_j)(x'_i - x''_j)^T - (x_i - x_j)(x_i - x_j)^T \Big) \mathbb{1}[l_i = l_j] \Bigg) \tag{2.9}$$

Desta forma, *MPCK-Means* é uma combinação entre aprendizado baseado em métricas e restrições que permite os *clusters* ficarem em diferentes subespaços e terem formas diferentes.

2.2 Medidas de Avaliação de Agrupamentos

O objetivo principal da função objetivo dos algoritmos de agrupamento é encontrar alta similaridade entre as instâncias de dados de um mesmo agrupamento e baixa similaridade entre os agrupamentos [Manning et al., 2008]. Por exemplo, documentos dentro de um grupo serem similares e documentos de diferentes grupos serem dissimilares. Esse nível de similaridade entre agrupamentos é um índice interno de qualidade de agrupamento. Entretanto, através desse índice não é possível comparar a efetividade de diferentes agrupamentos.

Outra forma de avaliar um agrupamento é através de índices externos que calculam o quanto os agrupamentos gerados se assemelham dos grupos que os dados realmente pertencem. Alguns índices externos são: pureza, informação mútua normalizada, índice *Rand* e *f-score*.

Algoritmo 5: *Metric Pairwise Constrained K-Means (MPCK-Means)*

Entrada: $X = \{x_1, x_2, \cdots, x_n\}$: conjunto de dados;
K: número de *clusters*;
$M = \{(x_a, x_b), \cdots, (x_y, x_z)\}$: conjunto de restrições *must-link* pareadas de X;
$C = \{(x_i, x_j), \cdots, (x_t, x_u)\}$: conjunto de restrições *cannot-link* pareadas de X;
W: conjunto de custos associados às violações de restrições *must-link*;
$\overline{W}$: conjunto de custos associados às violações de restrições *cannot-link*
Saída: K partições disjuntas de elementos pertencentes a X
início
crie um conjunto de λ vizinhanças $N = \{N_1, N_2, \cdots, N_\lambda\}$ a partir dos conjuntos M e C
se $\lambda \geq K$ **então**
inicialize os k centroides $\mu = \{\mu_1, \mu_2, ..., \mu_k\}$ utilizando o algoritmo *farthest-first traversal* começando do maior conjunto N
fim
senão se $\lambda < K$ **então**
inicialize os k centroides $\mu = \{\mu_1, \mu_2, ..., \mu_k\}$ com os centroides das λ vizinhanças $N = \{N_1, N_2, \cdots, N_\lambda\}$
inicialize os centroides restantes aleatoriamente
fim
enquanto *não atingir o critério de convergência* **faça**
para cada $x_i \in X$ **faça**
associe x_i ao *cluster* h_z, tal que

$$h_z = argmin_h \Bigg(\|x_i - \mu_h\|^2_{A_h} - \log(\det(A_h)) + \sum_{(x_i, x_j) \in M} \omega_{ij} f_M(x_i, x_j) \mathbb{1}[l_i \neq l_j] + \sum_{(x_i, x_j) \in C} \overline{\omega}_{ij} f_C(x_i, x_j) \mathbb{1}[l_i = l_j] \Bigg)$$

fim
para cada *cluster* $h_j \in h$ **faça**
calcule o novo centroide do *cluster*, tal que $\mu_{hj} = \frac{1}{|h_j|} \sum_{x \in h_j} x$
fim
para cada *cluster* $h_z \in h$ **faça**

$$A_{h_z} = |h_z| \cdot \Bigg(\sum_{x_i \in h_z} (x_i - \mu_{h_z})(x_i - \mu_{h_z})^T + \sum_{(x_i, x_j) \in M_{h_z}} \frac{1}{2} \omega_{ij} (x_i - x_j)(x_i - x_j)^T \mathbb{1}[l_i \neq l_j] + \sum_{(x_i, x_j) \in C_{h_z}} \overline{\omega}_{ij} \Big((x'_i - x''_j)(x'_i - x''_j)^T - (x_i - x_j)(x_i - x_j)^T \Big) \mathbb{1}[l_i = l_j] \Bigg)$$

fim
fim
fim

A pureza é uma medida simples, para calculá-la, cada grupo é atribuído à classe que é mais frequente dentro dele e, em seguida, a precisão dessa tarefa é medida contando o número de documentos atribuídos corretamente e dividindo por N, conforme Equação 2.10:

$$pureza(h, \mathbb{C}) = \frac{1}{N} \sum_{k} max_j |h_k \cap c_j| \tag{2.10}$$

Em que, $h = \{h_1, h_2, \cdots, h_k\}$ é o conjunto de agrupamentos e $\mathbb{C} = \{c_1, c_2, \cdots, c_j\}$ é o conjunto de classes.

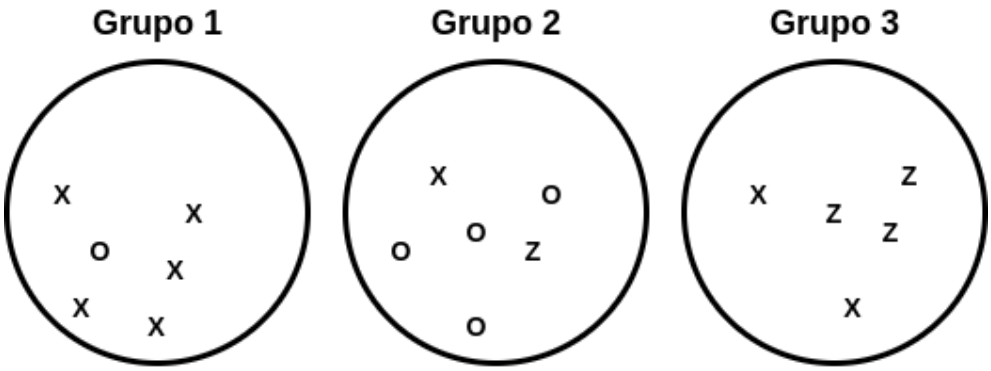


Figura 2.2: Exemplo de agrupamento.

A Figura 2.2 ilustra um exemplo de agrupamento, a classe majoritária e número de membros da classe majoritária para os grupos livres são: X, 5 (grupo 1); O, 4 (grupo 2); e Z, 3 (grupo 3). A pureza, neste caso, é $(1/17) \times (5 + 4 + 3) \approx 0.71$. Agrupamentos ruins tem índice de pureza próximos de zero, um agrupamento perfeito tem pureza igual a 1. É fácil obter alta pureza enquanto que o número de grupos é grande, em particular, a pureza seria 1 se cada grupo tivesse apenas uma instância, o que acabaria trocando a qualidade de agrupamento por número de grupos.

Um índice que consegue lidar com esse problema é informação mútua normalizada (em inglês, *Normalized Mutual Information* - NMI), conforme Equação 2.11:

$$NMI(h, \mathbb{C}) = \frac{MI(h, \mathbb{C})}{[H(h) + H(\mathbb{C})]/2} \tag{2.11}$$

De acordo com Manning et al. [2008], a informação mútua (em inglês, *Mutual Information* - MI), mede quanta informação a presença ou ausência de um termo contribui para tomar a decisão correta de classificação. Representada pela Equação 2.12 e 2.13, $P(h_k)$, $P(c_j)$ e $P(h_k \cap c_j)$ são as probabilidades de um documento estar no *cluster* h_k, classe c_j e a intersecção de estar no *cluster* h_k e ser da classe c_j. A Equação 2.13 é equivalente a Equação 2.12 pela estimativa por máxima verossimilhança.

$$MI(h,\mathbb{C}) = \sum_k \sum_j P(h_k \cap c_j) \cdot \log\left(\frac{P(h_k \cap c_j)}{P(h_k) \cdot P(c_j)}\right) \tag{2.12}$$

$$MI(h,\mathbb{C}) = \sum_k \sum_j \frac{|h_k \cap c_j|}{N} \cdot \log\left(\frac{N \cdot |h_k \cap c_j|}{|h_k| \cdot |c_j|}\right) \tag{2.13}$$

A função H é a entropia, conforme Equação 2.14 e 2.15, em que 2.15 é a estimativa por máxima verossimilhança da primeira.

$$H(h) = -\sum_k P(h_k) \cdot \log P(h_k) \tag{2.14}$$

$$H(h) = -\sum_k \frac{|h_k|}{N} \cdot \log\left(\frac{h_k}{N}\right) \tag{2.15}$$

MI tem o mesmo problema da pureza, quando o número de *clusters* é igual ao número de exemplos, seu valor chega no valor máximo. A normalização feita pelo denominador $[H(h)+H(\mathbb{C})]/2$ resolve esse problema já que a entropia cresce com número maior de *clusters*. *NMI* varia entre 0 e 1.

Outra forma de avaliar um agrupamento, é interpretar o agrupamento como uma série de decisões, uma para cada um dos $N \cdot (N-1)/2$ pares de documentos de uma coleção, em que deseja-se colocar dois documentos em um mesmo *cluster* apenas se forem similares. Uma decisão verdadeira positiva (em inglês, *True Positive* - TP) coloca dois documentos similares em um mesmo *cluster*. Uma decisão verdadeira negativa (em inglês, *True Negative* - TN) coloca dois documentos dissimilares em *clusters* diferentes. A decisão falsa positiva (em inglês *False Positive* - FP) coloca dois documentos dissimilares em um mesmo *cluster*. E a decisão falsa negativa (em inglês, *False Negative* - FN) coloca dois documentos similares em *clusters* diferentes. Essas decisões estão ilustradas na matriz de confusão da Figura 2.3.

O índice *Rand* (em inglês, *Rand index* - RI) mede a porcentagem de decisões corretas, conforme Equação 2.16. Pelo exemplo de agrupamento da Figura 2.2, $RI = (20/72)/(20+20+24+72) \approx 0.68$.

$$RI = \frac{TP+TN}{TP+FP+FN+TN} \tag{2.16}$$

Além dessas medidas, também são utilizadas a precisão (P), revocação (R) e *f-score*. A precisão, intuitivamente, mede a habilidade do algoritmo de agrupamento de não agrupar exemplos dissimilares em um *cluster* (Equação 2.17).

$$P = \frac{TP}{TP+FP} \tag{2.17}$$

A revocação mede, com que frequência o classificador está encontrando exemplos de uma classe (Equação 2.18).

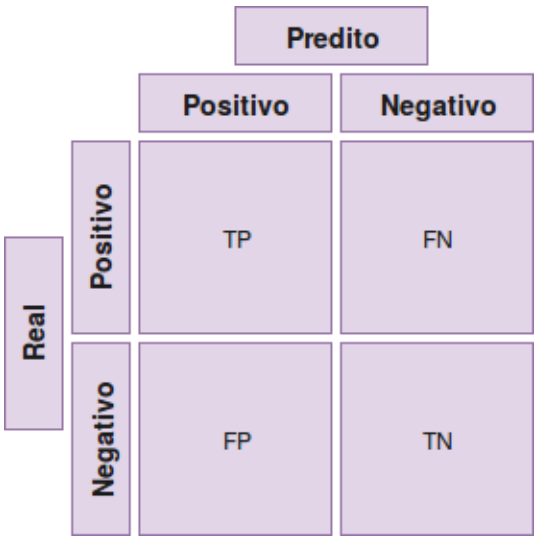


Figura 2.3: Matriz de confusão.

$$R = \frac{TP}{TP + FN} \tag{2.18}$$

E, o *f-score* é uma média ponderada entre precisão e revocação, como mostra a Equação 2.19. E será utilizado neste trabalho para avaliar o resultado dos agrupamentos.

$$R = \frac{2 \cdot P \cdot R}{P + R} \tag{2.19}$$

2.3 Considerações Finais

Neste capítulo foram apresentados fundamentos e conceitos de agrupamento semissupervisionado com ênfase em alguns algoritmos que foram a base de desenvolvimento deste trabalho. Após uma breve introdução sobre aprendizado semissupervisionado, foi apresentada uma visão geral sobre agrupamento semissupervisionado, cujo objetivo é melhorar resultados de agrupamento com supervisão limitada. Posteriormente, medidas de avaliação de agrupamentos foram apresentadas, entre elas, o *f-score* será utilizado neste trabalho.

A inclusão de conhecimento na forma de restrições como penalizações à função objetivo melhorou os resultados dos agrupamentos particionais [Wagstaff et al., 2001, Basu et al., 2002] comparados ao particionamento não supervisionado. O aprendizado de métricas [Xing et al., 2003] junto com o uso de restrições resultaram em um algoritmo mais robusto [Bilenko et al., 2004]. O uso do agrupamento semissupervisionado permite encontrar grupos de formatos variados, através da distorção do espaço de representações. Em específico, o PCK-Means e o MPCK-Means serão utilizados nos experimentos em

comparação ao agrupamento não supervisionado com K-Means.

Entretanto, as abordagens de agrupamento por particionamento possuem algumas desvantagens, uma vez que a inicialização dos centroides pode influenciar no resultado do agrupamento e a necessidade de saber de antemão a quantidade de particionamentos existentes. Existem na literatura outros algoritmos de agrupamento semissupervisionados, sem essas limitações, que podem ser explorados futuramente. A fundamentação apresentada neste capítulo serve de base para solucionar o problema que este trabalho busca abordar, a classificação de uma única classe, que será explicada no próximo capítulo.

CAPÍTULO

3

Aprendizado baseado em Uma Única

A abordagem de classificação de única classe (em inglês, *One-class classification* - OCC) [Moya et al., 1993] é utilizada em contextos em que uma única classe está presente ou em que a classe negativa está subamostrada.

Em problemas convencionais de classificação, dado um conjunto de classes predefinidas, busca-se classificar os novos exemplos em alguma das categorias já preestabelecidas. Por exemplo, se o modelo de classificação está treinado a categorizar uma fruta como maçã ou pera, mas se depara com um novo exemplo que é uma laranja, o modelo irá tentar categorizá-lo como maçã ou pera. Às vezes o problema de classificação não envolve alocar em algumas das categorias já predefinidas, mas em decidir se um objeto é de uma classe específica ou não [Tax, 2002].

Para resolver esse problema de classificar um exemplo como sendo de uma classe ou não, é possível usar classificação binária. Entretanto, de acordo com Bellinger et al. [2012], quando o nível de desbalanceamento dos exemplos de cada classe aumenta, a performance dos classificadores binários diminui, enquanto a performance dos classificadores de uma única classe permanece relativamente estável.

No mundo real, o desbalanceamento dos dados e a dificuldade em predefinir as classes que podem existir num conjunto de dados é muito comum, e a classificação de uma única classe é uma abordagem que vem sendo explorada para contemplar esse cenário.

Este capítulo está organizado da seguinte forma: na Seção 3.1 é feita uma breve introdução sobre classificação de uma única classe; na Seção 3.2, al-

guns tipos de abordagens são apresentadas; na Seção 3.3 alguns dos principais algoritmos da literatura são detalhados; e depois, os domínios de aplicações de OCC (Seção 3.4); por fim, as considerações finais (Seção 3.5).

3.1 Introdução a Classificação de Uma Única Classe

O problema de classificação de uma única classe [Moya et al., 1993] é fazer uma descrição de um conjunto de objetos de interesse e detectar quais objetos (novos) se assemelham a esse conjunto de treinamento. A descrição dos dados pode ser usada para um problema de classificação em que uma das classes é amostrada muito bem, enquanto a outra classe é bastante subamostrada. A subamostragem pode ser devido ao alto custo e obtenção desses exemplos. Por exemplo, em um sistema de monitoramento de máquinas em que a condição atual de uma máquina é examinada, um alarme é acionado quando a máquina mostra algum problema. As medições nas condições normais de trabalho de uma máquina são muito baratas e fáceis de obter. Por outro lado, medições de *outliers* exigiriam a destruição da máquina de todas as maneiras possíveis, isso é muito caro, ou até impossível [Japkowicz et al., 1995].

Em OCC, uma das classes referenciada como classe positiva ou classe de interesse, é bem caracterizada pelos exemplos no conjunto de dados de treinamento. Para as demais classes, possui nenhum ou poucos exemplos rotulados, que não formam uma amostra representativa dos dados negativos (não fazem parte da classe de interesse). Nesse sentido, Tax [2002] afirma que a tarefa em OCC é definir um limite ao redor dos exemplos positivos, que possa incluir o máximo de exemplos positivos, enquanto minimiza as chances de incluir um exemplo negativo.

Na literatura, um grande número de termos diferentes foram usados para se referir a esse problema. O termo *one-class classification* origina-se de Moya et al. [1993], mas também são usadas detecção de *outliers* (anomalias) [Ritter and Gallegos, 1997], detecção de novidades [Bishop, 1994] ou *concept learning* [Japkowicz, 1999]. Os diferentes termos se originam das diferentes aplicações às quais a classificação de classe única pode ser utilizada [Khan and Madden, 2009].

Em todos os métodos de OCC, dois elementos distintos podem ser identificados. O primeiro elemento é uma medida para a distância $d(z)$ ou semelhança (ou probabilidade) $p(z)$ de um objeto z com a classe de interesse. O segundo elemento é um limite θ em relação à distância ou semelhança. Novos objetos são aceitos pela descrição quando a distância em relação a classe de interesse é menor que o limiar θ_d, conforme Equação 3.1 [Tax, 2002]:

$$f(z) = I(d(z) < \theta_d) \tag{3.1}$$

ou quando uma semelhança é maior que o limiar θ_p, conforme Equação 3.2:

$$f(z) = I(p(z) > \theta_p) \tag{3.2}$$

na qual a função I é a função indicadora definida na Equação 3.3:

$$I(A) = \begin{cases} 1, & \text{se A é verdadeiro;} \\ 0, & \text{caso contrário.} \end{cases} \tag{3.3}$$

Os métodos de classificação de uma única classe se diferem nas definições de $p(z)$ (ou $d(z)$), na sua otimização de $p(z)$ (ou $d(z)$) e limiares em relação ao conjunto de treino X^{tr}.

Na maioria dos métodos de classificação de uma única classe, o foco está na otimização do modelo de semelhança p ou distância d. A otimização do limiar é feito posteriormente. Apenas alguns métodos de OCC otimizam o modelo $p(z)$ ou $d(z)$ para um limiar predefinido [Tax, 2002].

3.2 Tipos de Abordagens em OCC

De acordo com Khan and Madden [2009], diversas abordagens em OCC foram propostas usando algoritmos, metodologias e domínios de aplicações diferentes podendo ser divididas da seguinte forma:

- Disponibilidade dos dados de treino: Aprendizado com dados positivos apenas, aprendizado com dados positivos e dados negativos subamostrados e aprendizado com dados positivos e não rotulados.

- Metodologia usada: algoritmos baseados em *One-class Support Vector Machines* (OSVMs) ou metodologias baseadas em algoritmos diferentes de OSVMs.

- Domínio de aplicação: OCC aplicada na área de classificação textual ou outros domínios de aplicação.

Existem outras taxonomias mais elaboradas propostas na literatura, em Jaskie and Spanias [2019] Bekker and Davis [2020], mas por simplicidade, a de Khan and Madden [2009] foi utilizada.

O aprendizado com dados positivos e não rotulados (podem ser positivos ou não) é também chamada de *PU learning* (em inglês, *Learning from Positive and Unlabeled data*) [Liu et al., 2003, Denis et al., 2005, Li and Liu, 2005, Mordelet

and Vert, 2014, Du Plessis et al., 2015]. Essa categoria vem recebendo atenção na comunidade de classificação textual devido a relevância das aplicações dessa natureza. *PU learning* combina com algoritmos de aprendizado que não exigem abordagens totalmente supervisionadas como aprendizado com apenas dados positivos ou de classe única e aprendizado semissupervisionado [Bekker and Davis, 2020].

PU learning pode ser ilustrado com os seguintes exemplos extraídos de Bekker and Davis [2020]. Por exemplo, propaganda personalizada que usa páginas visitadas e cliques como exemplos positivos de páginas e propagandas de interesse. Todas as demais páginas ou propagandas que não foram visitadas ou clicadas não podem ser consideradas como não interessantes e não devem ser tratados como exemplos negativos, mas sim como não rotulados. Outro exemplo, registros médicos normalmente registram apenas a lista de doenças que um paciente foi diagnosticado e normalmente não incluem as doenças que um paciente não tem. Entretanto, a ausência do diagnóstico não significa que o paciente não tenha a doença. Um paciente pode simplesmente optar por não ir a um médico e, muitas doenças, como diabetes, muitas vezes não são diagnosticadas.

3.3 Alguns Algoritmos de OCC

A maioria dos algoritmos de OCC podem ser classificados em duas grandes áreas: algoritmos baseados em OSVM ou métodos não OSVMs [Khan and Madden, 2009]. Algumas das principais abordagens com OSVM são de:

- Tax and Duin [1999a,b]: propuseram o método *Suport Vector Data Description* (SVDD) que constrói uma hiperesfera ao redor da classe positiva que engloba maior parte dos dados positivos com um raio mínimo. Os exemplos que ficam fora dessa hiperesfera são classificados como negativos. Entretanto, a diminuição desse raio faz com que exemplos positivos fiquem fora dessa hiperesfera, sendo considerados como negativos. É possível usar diferentes *kernels* como *kernel* polinomial ou gaussiano. Esse método se torna ineficiente quando os dados tem alta dimensionalidade. Além disso, os autores propõe um método sofisticado para gerar de forma artificial *outliers* para evitar *overfitting* e *underfitting* [Tax and Duin, 2001]. Os autores aplicam essa abordagem no diagnóstico de falhas em máquinas e classificação de dígitos manuscritos.

- Schölkopf et al. [2000]: propuseram o método *One-Class Support Vector Machine* (OC-SVM), uma abordagem alternativa ao SVDD, em que é construído um hiperplano ao invés de uma hiperesfera, que fica longe

da origem e busca separar a região que possui dados positivos da região que não possui dados. Também usa *kernels* diferentes para melhorar a definição desse hiperplano. Os autores introduzem uma variável que controla o efeito de *outliers* como a dureza ou suavidade do limite em torno dos dados positivos. Esse método foi aplicado em dados de serviços postais dos EUA e dígitos manuscritos.

- Yu et al. [2003, 2004], Yu [2005]: propuseram o algoritmo *Support Vector Mapping Convergence* (SVMC), com o uso de SVM modificado, esse algoritmo é composto de duas etapas: uma etapa de mapeamento dos dados que separa os dados em positivos ou negativos e outra de convergência que amplia os limites das classes de acordo com os exemplos classificados.

Diversas váriações de SVDD ou OC-SVM são encontradas na literatura [Manevitz and Yousef, 2001, Zhao et al., 2005, Li and Liu, 2003, Campbell and Bennett, 2001, Liu and Madden, 2007, Sanchez-Hernandez et al., 2007, Muñoz-Marí et al., 2010]. OC-SVM será abordado brevemente na Seção 3.3.1, e SVMC será abordado com mais detalhes na Seção 3.3.2 para exemplificar e ilustrar um pouco a abordagem OCC com algoritmos baseados em SVM.

Outros algoritmos utilizados envolvem redes neurais ou redes neurais profundas [Moya et al., 1993, Moya and Hush, 1996, Leng et al., 2015, Perera and Patel, 2019, Sabokrou et al., 2018, Erfani et al., 2016, Ruff et al., 2018, Schlachter et al., 2019], árvores de decisão [De Comité et al., 1999, Letouzey et al., 2000], *random forests* [Désir et al., 2013], *Naive Bayes* (NB) e *Expectation Maximization* [Liu et al., 2002], KNN [Ridder, 1998], regressão logística [Lee and Liu, 2003] e demais abordagens [Ridder, 1998, Janssens et al., 2009, Ganesan et al., 2013].

Além disso há estudos relacionados ao uso de algoritmos de agrupamento não supervisionado usando *K-Means* ou suas variações. Krawczyk et al. [2014] propuseram uma arquitetura baseada no uso de algoritmos de agrupamento não supervisionado (como *k-means*, *fuzzy c-means* e *kernel fuzzy c-means*) em conjunto com um classificador OSVM ou OSVM ponderado. Em Gôlo et al. [2019], avaliações experimentais com algoritmos de OCC de diferentes categorias e técnicas de pré-processamento de textos foram feitas. O algoritmo baseado no *k-means* obteve as melhores performances de classificação para a maioria dos experimentos realizados.

Uma visão geral sobre diversas abordagens algorítmicas para PUL e OCC podem ser encontradas nas seguintes *surveys*: Jaskie and Spanias [2019], Bekker and Davis [2020], Perera et al. [2021]. A variedade de algoritmos vão desde modelos probabilísticos, estatísticos, aprendizado profundo entre outros. Alguns algoritmos de OCC são abordados com mais detalhes nas pró-

ximas seções: OC-SVM, que será utilizado como *baseline* de comparações; alguns algoritmos de similaridade, K-NND, K-NNRD e K-Means, que foram utilizadas em Gôlo et al. [2019], que estão mais próximos das abordagens de agrupamento semissupervisionado que são avaliadas nesse trabalho.

3.3.1 One-class Support Vector Machine (OC-SVM)

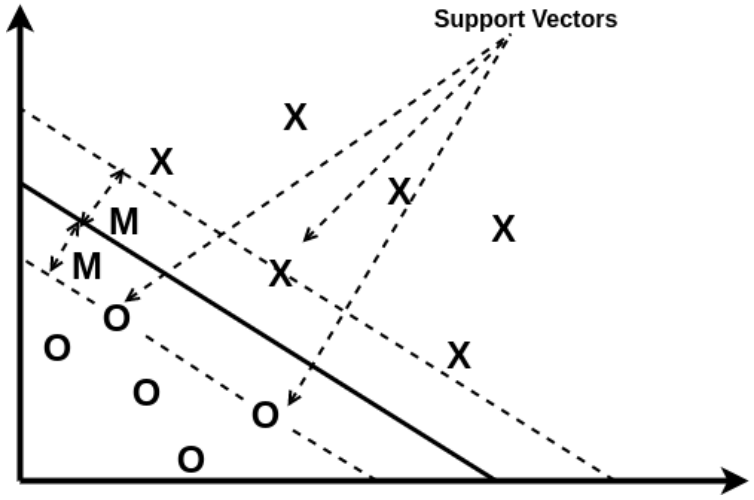
SVM fornece várias propriedades, como maximização de margem e transformação não linear do espaço de entrada no espaço de recurso usando métodos de *kernel*. Para ilustrar, considere sua forma mais simples, um SVM linear. Um SVM linear é um hiperplano que separa um conjunto de dados positivos de um conjunto de dados negativos com margem máxima no espaço de recurso. A margem (M) indica a distância do hiperplano (limite da classe) até os dados positivos e negativos mais próximos no espaço de características. A Figura 3.1 mostra um exemplo de um problema bidimensional simples que é separável linearmente. Cada recurso corresponde a uma dimensão no espaço de recursos. A distância do hiperplano até um ponto de dados é determinada pela força de cada recurso dos dados. Por exemplo, considere um classificador de página de currículo. Se uma página tem várias características fortes relacionadas ao conceito "currículo", a página seria classificada como positiva, e a localização seria longe dos limites da classe. Da mesma forma que, uma página que não possui características de currículo, estaria localizada longe dos limites do lado negativo.

Nos casos em que os pontos não são linearmente separáveis, SVM possui um parâmetro C (v em v-SVM) que controla o ruído nos dados de treinamento. SVM calcula o hiperplano que maximiza as distâncias para suportar vetores para uma determinada configuração de parâmetro. Para problemas que não são linearmente separáveis, métodos de *kernel* podem ser usados para transformar o espaço de características inicial para outro de alta dimensão.

Schölkopf et al. [2000] sugeriram um método para adaptar o SVM padrão ao chamado problema de OCC, onde apenas exemplos positivos estão disponíveis para o aprendizado. Com uma forte base matemática de SVM, o SVM de classe única (OC-SVM) calcula um limite de dados positivos sem dados negativos no espaço de recursos. OC-SVM possui dois parâmetros - v (para controlar o ruído nos dados de treinamento) e γ (para controlar a "suavidade" dos limites).

OC-SVM fornece as mesmas vantagens de SVM, no entanto, requer uma quantidade muito maior de dados de treinamento positivos para induzir um limite de classe preciso, porque seus vetores de suporte (SVs) do limite vêm apenas do conjunto de dados positivos e o pequeno número de SVs positivos dificilmente pode cobrir as principais direções do limite, especialmente em

Figura 3.1: Uma representação de SVM linear bidimensional. M é a distância do separador aos *support vectors* no espaço de características.



Fonte: Extraído de Yu et al. [2004, p. 73].

espaços de alta dimensão. Como resultado, OC-SVM tende a ter *overfitting* ou *underfitting*.

3.3.2 Support Vector Mapping Convergence (SVMC)

Yu et al. [2003, 2004] propuseram um classificador de páginas *Web* sem exemplos negativos, em que é utilizado a técnica SVMC que consegue atingir resultados melhores do que OC-SVM. Sendo:

- $\mathbb{U}$: espaço de características para o conjunto universal tal que $\mathbb{U} \in \mathbb{R}^m$, em que m é o número de dimensões;
- U: conjunto de dados não etiquetados, é uma amostra uniforme do conjunto universal;
- x: uma instância de dados tal que $x \in \mathbb{U}$;
- $\mathbb{P}$: o subespaço da classe positiva dentro de $\mathbb{U}$, do qual conjuntos de dados positivos P é amostrado.

Por exemplo, na classificação de páginas *Web*, o conjunto universal é a *Web* inteira, U é uma amostra da *Web*, P é uma coleção de páginas da *Web* de interesse, e $x \in \mathbb{R}^m$ é uma instância de página *Web*.

SVMC tem como base o algoritmo *Mapping Convergence* (MC). O algoritmo MC é composto de dois estágios: o estágio de mapeamento e o estágio de convergência. No estágio de mapeamento, o algoritmo usa um algoritmo de classificação C_1 (ex: OC-SVM), que desenha uma aproximação inicial de "fortes negativos" - os dados negativos localizados longe da fronteira da classe positiva

Algoritmo 6: *Mapping Convergence - MC*

Entrada: P: conjunto de dados positivos;
U: conjunto de dados não etiquetados.
Saída: Hipóteses em cada iteração $h_1, h_2, \cdots, h_k$.
C_1: um algoritmo para identificar "fortes negativos" de U (ex: OC-SVM).
C_2: um algoritmo de aprendizado supervisionado que maximiza a margem (ex: SVM).
início

1. Use C_1 para construir um classificador h_1 a partir de P e U que classifica apenas os "fortes negativos" como negativos e os outros como positivos
2. Classificar U por h_1
   - N_1 = exemplos de U classificados como negativos por h_1
   - P_1 = exemplos de U classificados como positivos por h_1
3. Seja $N = \emptyset$ e $i = 1$
4. **Faça**
   (a) $N = N \cup N_i$
   (b) Use C_2 para construir h_{i+1} a partir de P e N
   (c) Classifique P_i por h_{i+1}
      - N_{i+1} = exemplos de P_i classificados como negativos por h_{i+1}
      - P_{i+1} = exemplos de P_i classificados como positivos por h_{i+1}
   (d) $i = i + 1$

   Enquanto $N_i = \emptyset$
5. Retorne $\{h_1, h_2, \cdots, h_k\}$

fim

em $\mathbb{U}$ (passo 1 e 2 no Algoritmo 6). Com base na aproximação inicial, o estágio de convergência é executado na iteração usando um segundo classificador C_2 (por exemplo, SVM), que maximiza a margem para fazer uma aproximação progressivamente melhor dos dados negativos (passo 3 a 5 no Algoritmo 6). Logo, o limite da classe eventualmente converge para o limite ao redor do conjunto de dados positivos no espaço de características e os fortes negativos são usados como exemplos rotulados da classe negativa para o SVM.

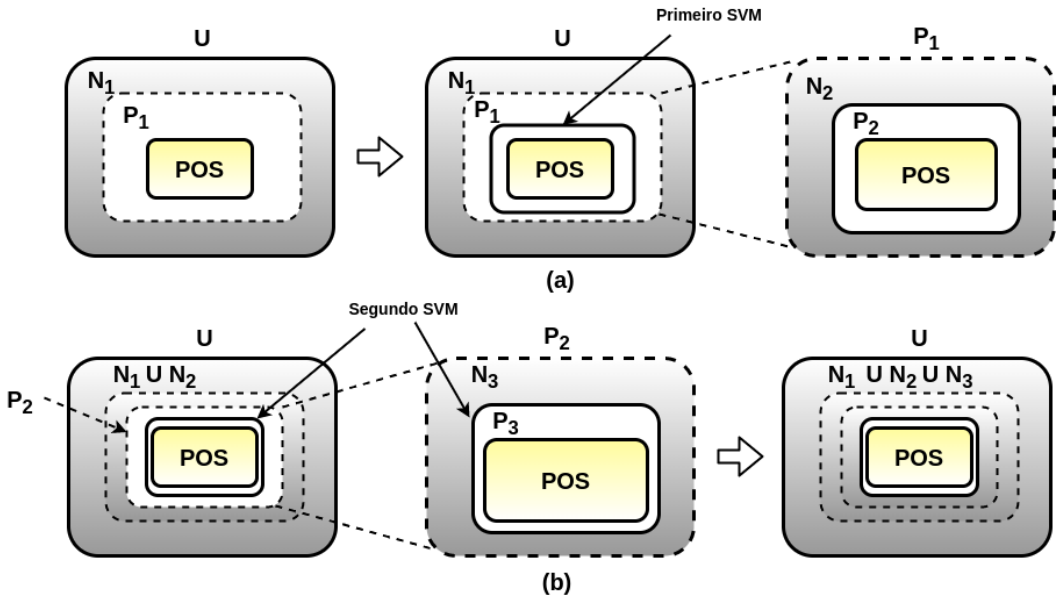
As características "fortes positivas" dos dados positivos e não rotulados podem ser identificadas através da checagem dessas características entre os dados de treinamento positivo e não rotulados. Por exemplo, se uma característica ocorre em 90% dos dados positivos e apenas em 10% dos dados não rotulados, essa característica seria uma "forte positiva".

Ao criar a lista de todas as características positivas que ocorrem mais frequentemente nos dados positivos do conjunto de treino do que nos dados não rotulados, com o uso dessa lista, é possível filtrar qualquer dado provavelmente positivo do conjunto de dados não rotulados, o que resta apenas os dados fortemente negativos. Um dado que não possui nenhuma característica da lista de características positivas é considerado um forte negativo. Com o uso dessa lista, pode-se apenas identificar os fortes negativos que estão localizados longe do limite da classe.

Se um SVM for construído apenas com positivos e negativos fortes, o limite da classe estará longe de ser exato devido à insuficiência de dados negativos no conjunto de dados de treinamento. O limite será localizado demais para o lado negativo porque o limite do SVM terá margens iguais aos dados de treinamento positivo e negativo. No entanto, esse limite tendencioso pode ser útil porque ainda maximiza a margem. Ao aplicar a classificação sobre os dados não rotulados que estão nesse limite tendencioso, é possível construir uma aproximação melhor dos dados negativos. Ao iterar esse processo, é possível extrair continuamente dados negativos dos dados não rotulados até que não exista dados negativos nos dados não rotulados. O limite também convergirá para o limite verdadeiro. Isso pertence ao estágio de convergência do algoritmo MC. Esse processo é ilustrado na Figura 3.2.

O tempo de treinamento de MC pode ser longo se $|U|$ for muito grande, porque o tempo de treinamento de SVM depende do tamanho do conjunto de dados, e MC roda iterativamente. Para evitar isso, o algoritmo SVMC foi projetado para usar minimamente o conjunto de dados em cada iteração tal que não há degradação da acurácia dos limites. Para ilustrar como o SVMC consegue isso, considere o ponto de iniciar a terceira iteração (quando i = 3) no MC (passo 4 (a) no Algoritmo 6). Depois de mesclar N_2 em N, para construir h_3, talvez não sejam necessários todos os dados de N, pois os dados distantes

Figura 3.2: Margem de convergência de SVM: (a) treinando primeiro SVM (a partir de N_1 e P) que divide P_1 em N_2 e P_2. (b) treinando segundo SVM a partir de ($N_1 \cup N_2$ e P) que divide P_2 em N_3 e P_3.



Fonte: Extraído de Yu et al. [2004, p. 76].

de h_3 não contribuirão para os SVs. O conjunto de SVs negativos de h_2 é o conjunto de dados representativo para N_1 e N_2, portanto, mantemos apenas os SVs negativos de h_2 e o conjunto de dados N_2 recém-induzido para suportar o lado negativo de h_3.

3.3.3 K-Nearest Neighbor (KNN) aplicado à OCC

KNN é um classificador baseado em proximidade que usa medidas baseadas em distância para realizar a classificação. A ideia principal é que os documentos que pertencem à mesma classe são mais provavelmente "semelhantes" ou próximos uns dos outros com base nas medidas de similaridade. A classificação do documento de teste é inferida dos rótulos de classe dos documentos semelhantes no conjunto de treinamento. Em que, a classe mais comum desses k vizinhos é relatada como o rótulo de classe [Allahyari et al., 2017]. Foi um dos primeiros algoritmos utilizados para determinar uma solução para o problema do problema do caixeiro viajante e atualmente tem sido explorada para outros problemas.

Em Gôlo et al. [2019], o KNN foi aplicado à OCC. Duas variantes que foram usadas e avaliadas em Gôlo et al. [2019]: o *k-Nearest Neighbor Density-based* (k-NND) e o *k-Nearest Neighbor Relative Density-based* (k-NNRD). Em que na primeira, seja x_i um exemplo do conjunto de dados, a semelhança entre o exemplo e a classe de interesse $f(x_i)$ é definida na Equação 3.4, em que $N_{(x_i,k)}$ representa o conjunto dos k vizinhos mais próximos de x_i, e $cos(x_i, x_j)$ é a similaridade de cosseno entre as representações vetoriais dos exemplos.

$$f(x_i) = densidade(x_i,k) = \sum_{x_j \in N_{(x_i,k)}} cos(x_i,x_j)/|N_{(x_i,k)}| \tag{3.4}$$

Enquanto que k-NNRD, verifica se a densidade do exemplo a ser classificado é similar à densidade dos exemplos vizinhos, tendo sua a semelhança $f(x_i)$ definida na equação 3.5:

$$f(x_i) = densidade\_relativa(x_i,k) = \frac{densidade(x_i,k)}{\sum_{x_j \in N_{(x_i,k)}} densidade(x_i,k)/|N_{(x_i,k)}|} \tag{3.5}$$

3.3.4 K-Means aplicado à OCC

O *k*-Means é um algoritmo de agrupamento não supervisionado baseado na iterativa relocação que particiona um conjunto de dados em k agrupamentos, minimizando localmente o erro quadrático entre os dados e os centroides dos agrupamentos. Para um conjunto de dados $X = \{x_1,...,x_n\}$, *k*-Means cria k-partições $\{X_j\}_{j=1}^{k}$ de X, em que $\{\mu_1,...,\mu_k\}$ representam os centroides das k partições, com a seguinte função objetivo conforme Equação 3.6:

$$EQ = \sum_{j=1}^{k} \sum_{x_i \in X_j} \| x_i - \mu_j \|^2 \tag{3.6}$$

Iterativamente, cada instância de dados $x_i \in X$ é alocado ao *cluster* cuja distância entre o centroide do *cluster* e x_i é a menor comparada aos demais *clusters*. E, cada *cluster* tem seu centroide μ_j atualizado com o valor médio das instâncias que o constitui.

É um algoritmo de agrupamento não supervisionado, seu uso para OCC pode ser formulado da seguinte maneira [Gôlo et al., 2019] conforme Equação 3.7: seja um conjunto de dados X formado por k grupos diferentes $G_1 \cup G_2 \cup ... G_k$, e g_i é a média dos vetores dos pontos em G_i, em que o valor de similaridade $f(x_i)$ é dado considerando a similaridade com o centroide do grupo mais próximo.

$$f(x_i) = max_{Gj \in X} cos(x_i, g_j) \tag{3.7}$$

3.3.5 Deep Autoencoders aplicado à OCC

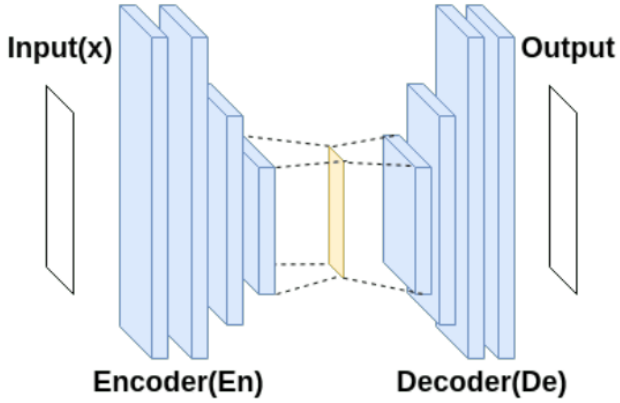
Um *autoencoder* é uma rede neural (convolucional) que consiste em um codificador e redes de decodificador, conforme mostrado na Figura 3.3. O codificador compreende um conjunto de camadas de convolução seguidas por funções de ativação. O decodificador consiste em camadas convolucionais transpostas e geralmente tem uma estrutura semelhante à de um codificador invertido. Um *autoencoder* é treinado com o objetivo de minimizar a distância entre a entrada e a saída da rede. Em teoria, qualquer medida de distância,

como Equação 3.8, pode ser considerada para aprender os parâmetros do codificador automático, onde x é a imagem de entrada [Perera et al., 2021].

$$L_{mse} = ||x - De(En(x))||_2^2 \tag{3.8}$$

Tem sido explorado em OCC para aplicações como detecção de novidades e detecção de anomalias conforme Chen et al. [2017], Oza and Patel [2019], Pidhorskyi et al. [2018], Salehi et al. [2020], Tran and Hogg [2017].

Figura 3.3: Exemplo de um *autoencoder*.



Fonte: Extraído de Perera et al. [2021, p. 4].

3.4 Alguns Domínios de Aplicações de OCC

OCC pode ser aplicado a diversos domínios de aplicação, principalmente em mineração de textos. Em mineração de textos é abordada em:

- Classificação de páginas da *internet* ou textos: identificar os documentos de uma classe específica entre vários outros tipos de documentos, por exemplo obter documentos relacionados a ciência num grupo de discussão de temas variados [Liu et al., 2002, Li and Liu, 2003, Peng et al., 2016, Denis et al., 2005], ou classificar páginas Web pelo tipo, por exemplo, um classificador que saiba identificar a página inicial de um *website* [Yu et al., 2003, 2004].

- Verificação de autoria: apenas os exemplos da escrita de um autor é dado e a tarefa é determinar se o determinado texto é ou não escrito pelo autor [Koppel and Schler, 2004].

- *Text Streams with Concept Drift*: em que um grande volume de dados de documentos chega em velocidade muito rápida e com a mudança de interesse e distribuição de dados. Por exemplo, um gerente de uma seção

de uma empresa grande quer em poucos minutos encontrar os *e-mails* de *feedbacks* de um produto recentemente lançado, o sistema de classificação deveria conseguir todos os *feedbacks* relacionados ao tópico nos *e-mails* que estão chegando [Zhang et al., 2008].

- Qualidade do texto: verificar se um dado texto sofre de uma falha de qualidade específica ou não. Por exemplo, detectar se um texto da Wikipedia possui seções vazias, falta de referências bibliográficas, entre outros [Anderka et al., 2011].

Alguns dos outros domínios de aplicação envolvem [Khan and Madden, 2009]: reconhecimento de dígitos manuscritos [Schölkopf et al., 2000, Tax and Duin, 2001], recuperação de informação [Manevitz and Yousef, 2001], reconhecimento de faces ou objetos [Wang et al., 2004, Zeng et al., 2006, Munroe and Madden, 2005], análises médicos [Gardner et al., 2006], bioinformática [Spinosa and de Leon Ferreira, 2004, Alashwal et al., 2006], detecção de *spam* [Sun et al., 2005], detecção de anomalias [Li et al., 2003, Perdisci et al., 2006] e detecção de falhas em máquinas [Shin et al., 2005]. Além de serem extensamente utilizados para detecção de imagens anormais, detecção de eventos anormais, aplicações de biometria, *antispoofing*, reconhecimento de fala, aplicações industriais, aplicativos de dados de sensoriamento remoto e hiperespectral [Perera et al., 2021, Alam et al., 2020].

3.5 Considerações Finais

Neste capítulo foram apresentados conceitos de classificação de uma única classe e as abordagens relacionados à disponibilidade dos dados de treino, algoritmos ou metodologias utilizadas e domínios de aplicação na literatura. As abordagens relacionadas à disponibilidade dos dados são aquelas que usam apenas dados positivos, dados positivos e amostra não representativa dos dados negativos, ou dados positivos e dados não rotulados. Sendo essa última abordagem amplamente explorada em aplicações de mineração de textos devido à abundância de dados textuais e o custo de conseguir dados rotulados.

A maior parte dos algoritmos e metodologias existentes na literatura são baseados em SVM aplicado à OCC. A natureza de OCC combina com abordagens que não são totalmente supervisionadas, por ser aplicada em dados desbalanceados ou em que apenas a classe de interesse está presente. Entretanto, não foi encontrado nessa revisão bibliográfica, abordagens com agrupamento semissupervisionado, além de que a performance de agrupamento supervisionado consegue obter resultados melhores que abordagens convencionais em OCC. Como visto no capítulo anterior abordagens semissupervisionados, conseguem obter resultados melhores do que abordagens não supervisionadas.

Desta forma, agrupamento semissupervisionado aplicado a OCC, em específico PUL, é o foco desse projeto.

CAPÍTULO

4

PUL-SSC: Algoritmo de PUL com Agrupamento Semissupervisionado Proposto

O uso de algoritmos de similaridade, em específico, de agrupamento parecem ser promissores para resolver o problema de OCC, conforme apresentado no capítulo anterior, trabalhos com abordagens de agrupamento não supervisionados (ex: *k-Means*) conseguiram ter resultados de OCC melhores do que métodos estatísticos, probabilísticos e baseados em similaridade. A inclusão de conhecimento supervisionado na forma de restrições, o uso de agrupamento semissupervisionado, será explorado neste trabalho, que busca através de pouco conhecimento supervisionado adicionado ao processo de agrupamento, definir uma boa descrição da classe positiva ou de interesse.

Neste capítulo é detalhado a proposta deste trabalho, o algoritmo proposto PUL-SSC: na Seção 4.1 uma visão geral e sua formulação são apresentadas, detalhes e ilustração do funcionamento na Seção 4.2, e na Seção 4.3, as considerações finais.

4.1 Positive and Unlabeled Learning with Semi-Supervised Clustering (PUL-SSC)

Este trabalho propõe o algoritmo PUL-SSC, que busca fazer aprendizado de uma única classe através de dados positivos e não rotulados para o problema de classificação de uma única classe. O algoritmo proposto é composto por

um algoritmo de agrupamento semissupervisionado (ex: MPCK-Means) e um processo transdutivo de propagação de rótulos.

O algoritmo de agrupamento semissupervisionado empregado, MPCK-Means, tem o papel de agrupar os dados, e através do aprendizado de métricas conseguir distorcer o espaço de representações para que os dados similares possam estar próximos e, dados dissimilares possam estar distantes. De forma análoga ao algoritmo SVMC [Yu et al., 2004], que através do uso de OSVM, iterativamente busca criar uma delimitação da classe de interesse, identificando fortes candidatos da classe de interesse ou não, o algoritmo PUL-SSC proposto, com apenas poucos dados rotulados da classe de interesse, aprende a rotular os dados através da identificação de fortes candidatos da classe de interesse (fortes positivos) e de fortes candidatos que não fazem parte da classe de interesse (fortes negativos). A cada iteração, esses fortes positivos são rotulados como positivos, restrições *must-link* são geradas entre esses novos positivos e os positivos já existentes. Enquanto que, os fortes negativos são rotulados como negativos e restrições *cannot-link* são geradas entre esses novos negativos e os dados positivos.

Para definir quem são esses fortes positivos ou fortes negativos é utilizada a distância euclidiana. É calculada a distância euclidiana entre todos os grupos ainda não rotulados e os grupos positivos, o grupo não rotulado mais distante dos grupos positivos, é considerado como forte negativo, e propagado como sendo da classe negativa. Enquanto que, o grupo não rotulado mais próximo é considerado como forte positivo, e propagado como sendo da classe positiva. Esse processo de propagação de rótulos é repetido até que não existam mais grupos não rotulados.

Dessa forma, seja $X = \{x_1, x_2, \cdots, x_m\}$ um conjunto de dados e P um subconjunto de X composto por dados positivos, P é inicialmente composto por p exemplos rotulados de X, enquanto que para os demais exemplos de X, durante o treinamento são iniciados como não rotulados. Através do uso de um algoritmo de agrupamento, k grupos de X são criados, $\{G_1, G_2, \cdots, G_k\}$, e cada grupo tem como seu respectivo centroide $\{g_1, g_2, ..., g_k\}$. Para cada grupo gerado, é verificado se possui algum exemplo da classe de interesse, seja G_i um grupo com algum exemplo da classe de interesse todos os exemplos de G_i são incluídas em P e considerado como um grupo positivo. Para os demais grupos que não possuem nenhum dado positivo, é realizado o cálculo de distância entre o centroide de cada grupo não rotulado e os centroides dos grupos positivos. O cálculo da distância é expressa na Equação 4.1, seja G_u um grupo não rotulado, a distância é definida como a soma das distâncias euclidianas entre o centroide g_u do grupo G_u e cada centroide positivo, g_p tal que $g_p = \{\forall g_i \mid p_i \in G_i\}$.

$$f(G_u) = \sum_{g_p \in P} distancia\_euclidiana(g_u, g_p) \tag{4.1}$$

O grupo com a maior distância calculada, Equação 4.2, ou seja, o grupo não rotulado mais distante dos grupos positivos, G_u tal que:

$$f(G_u) = max(\sum_{g_p \in P} distancia\_euclidiana(g_u, g_p)) \tag{4.2}$$

tem todos os seus exemplos rotulados como sendo da classe negativa, e restrições *cannot-link* são criadas entre esses exemplos e os dados positivos. De forma análoga, o grupo G_u da Equação 4.3:

$$f(G_u) = min(\sum_{g_p \in P} distancia\_euclidiana(g_u, g_p)) \tag{4.3}$$

é o grupo não rotulado mais próximo dos grupos positivos, todos os seus exemplos são rotulados como sendo da classe positiva, e restrições *must-link* são criadas entre esses exemplos e os dados previamente rotulados como positivos. Esse processo se repete até não ter dados não rotulados. Esse processo iterativo é ilustrado com mais detalhes na próxima seção.

4.2 Funcionamento do PUL-SSC

O PUL-SSC funciona da seguinte maneira:

1. Apenas p exemplos de dados positivos possuem a rotulação conhecida $x \in P$, o restante dos exemplos do conjunto são considerados dados não rotulados $x \in U$.

2. Conjunto de restrições *must-link* M e *cannot-link* C são inicializadas como vazias.

3. Algoritmos de agrupamento semissupervisionado são rodados com conjunto de restrições M e C.

4. Uma vez que um agrupamento inicial é feito, o processo de propagação de rótulos transdutivo é realizado. Primeiro, todos os *clusters* gerados que possuem algum dado positivo, $x \in P$, têm todas as instâncias incluídas em P, rotuladas como positivas.

5. Para os demais *clusters*, é calculada a soma da distância euclidiana entre cada *cluster* não rotulado e cada *cluster* positivo. O *cluster* não rotulado mais distante será considerado como *cluster* negativo, e todas as instâncias deste *cluster* são movidas de U para N. Restrições *cannot-link* são

geradas entre os exemplos positivos e negativos e adicionados ao conjunto C. O *cluster* não rotulado mais próximo é então, considerado como *cluster* positivo e todas as instâncias são movidas de U para P. Restrições *must-link* são geradas entre os novos exemplos positivos e os positivos atuais, e adicionados ao conjunto M.

6. Etapa 3 a 5 são repetidas até que todas as instâncias tenham sido rotuladas, isto é, U esteja vazia.

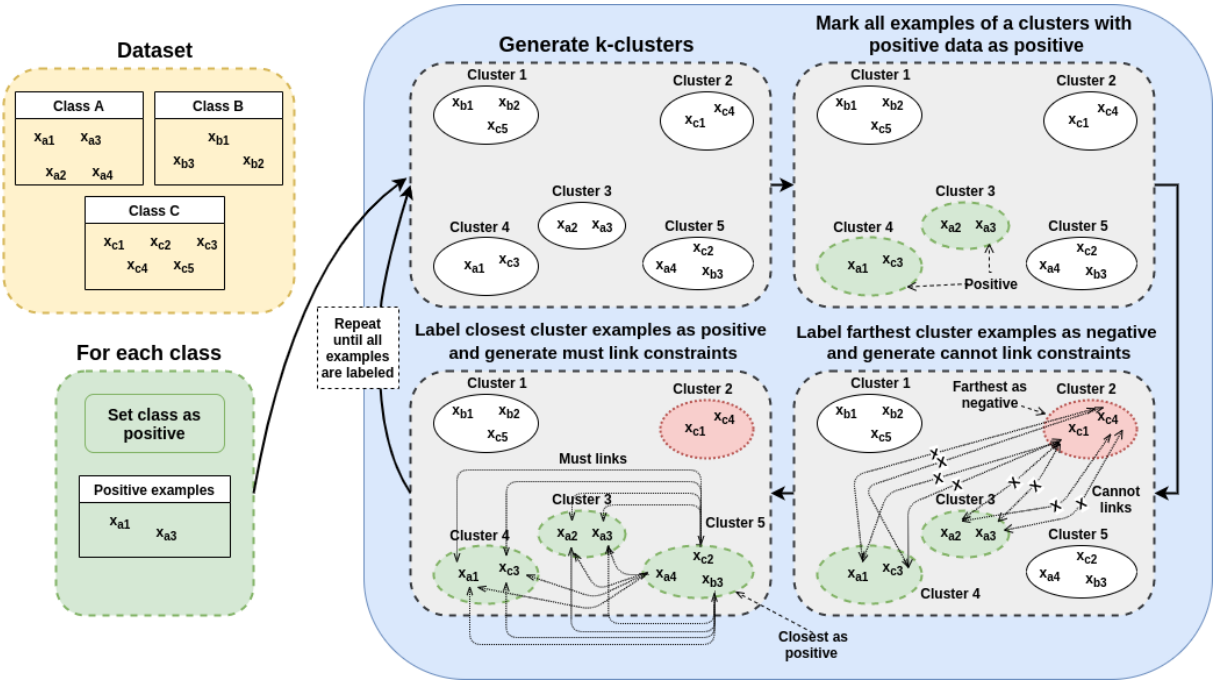


Figura 4.1: Fluxo PUL com agrupamento semissupervisionado implementado.

Para exemplificar como funciona o PUL-SSC, conforme o exemplo da Figura 4.1, suponha que o conjunto de dados utilizado possui 3 classes (A, B e C). Seja A considerada como a classe de interesse, apenas p exemplos da classe A possuem seus rótulos conhecidos durante o treinamento (x_{a_1}, x_{a_3}), os demais exemplos da classe A e os exemplos das demais classes (B e C) são consideradas como não rotuladas. Através de um algoritmo de agrupamento, k grupos são gerados, os grupos que possuem algum dado positivo (grupo 3 e 4), possuem todos os seus exemplos propagados com rótulo positivo. Para os demais grupos não rotulados (1, 2 e 5) é feito a soma da distância euclidiana entre seu centroide e cada centroide positivo, o grupo mais distante (grupo 2) considerado como forte negativo, possui todos os seus exemplos considerados como negativos e restrições *cannot-link* são geradas entre os exemplos do grupo 2 com grupo 3 e 4. Enquanto que o grupo mais próximo dos grupos positivos, grupo 5, possui todos os seus exemplos considerados como sendo positivos, e restrições *must-link* são geradas entre os exemplos do grupo 5 com grupo 3 e 4. Como ainda há algum grupo não rotulado, o grupo 1, esse processo é

feito novamente, é rodado o algoritmo de agrupamento juntamente com essas restrições geradas, e repetido o processo de propagação de rótulos.

Para avaliar o resultado, o *F-score* para a classe de interesse [Gôlo et al., 2019] é calculada comparando os rótulos preditos e os rótulos reais.

4.3 Considerações Finais

A forma como foi estruturada esse algoritmo, está flexível para testar diferentes tipos de algoritmos de agrupamento. O que viabilizou os experimentos realizados no próximo capítulo. Outras abordagens como a definição de um valor de limiar fixo para classificar a instância como sendo da classe de interesse ou não também foi testada, entretanto não abordada nesse trabalho, já que os resultados não foram significativos, uma vez que a definição desse valor de limiar foi arbitrário e a calibração desse valor de limiar seria custoso. Dessa forma, ao optar por escolher o *cluster* mais distante como sendo da classe negativa, esse valor de limiar relativo em relação aos centroides positivos, acabou gerando melhores resultados, e sendo de fato apresentado nesse trabalho.

O algoritmo PUL-SSC proposto, busca aprender os limites da classe de interesse através de poucos dados, conforme apresentado, ele faz essa identificação dos fortes candidatos da classe positiva ou da classe negativa através de uma função de distância, que a princípio usou-se a distância euclidiana, outras medidas de distância podem ser facilmente usadas e avaliadas. Nesse processo de propagação de rótulos, restrições *must-link* e *cannot-link* são geradas e utilizadas para a próxima iteração de agrupamento. Os algoritmos de agrupamento semissupervisionado utilizados penalizam a violação das restrições, então esses grupos podem ser reestruturados e não necessariamente vão se manter conforme o agrupamento inicial. A cada iteração mais restrições são geradas, e tornando-se um problema mais complexo, sendo cada vez mais difícil não violar as restrições. O impacto do crescimento dessas restrições, a verificação de quais e quantas restrições de fato o algoritmo consegue seguir, é interessante para passos futuros. A avaliação desse algoritmo proposto e os resultados obtidos são apresentados no próximo capítulo.

CAPÍTULO

5

Experimentos e Resultados

Neste capítulo, a metodologia de avaliação é apresentada. Os experimentos comparam o desempenho de abordagens baseadas em agrupamento, tanto não supervisionadas quanto semissupervisionadas, em cenários típicos de OCC e PUL, lidando com dados numéricos e textuais e usando poucos dados rotulados. Foi usado OC-SVM, o algoritmo de OCC mais conhecido, como *baseline* para a avaliação.

Este capítulo está organizado da seguinte forma: na Seção 5.1, as bases de dados utilizadas nos experimentos são especificadas, na Seção 5.2 os detalhes dos experimentos - as configurações utilizadas nos algoritmos, na Seção 5.3, os resultados são apresentados de acordo com tipo da base (numérica ou textual), e por fim, as considerações finais na Seção 5.4.

5.1 Bases de Dados

Foram utilizados oito conjunto de dados numéricos de *UCI Machine Learning Repository*[1], listados na Tabela 5.1, e três conjuntos de dados textuais [Rossi et al., 2013], marcados com um asterisco na Tabela 5.1. Esses dados textuais já passaram por etapas de pré-processamento textual: remoção de *stopwords*, *stemming* com algotitmo de Porter [Porter et al., 1980] e transformação para representação BOW usando TF-IDF como critério de ponderação dos termos. Nos conjuntos de dados numéricos, exemplos com atributos faltantes foram desconsiderados. A seguir, uma breve contextualização de cada conjunto de dados é apresentada:

[1] https://archive.ics.uci.edu/ml/index.php

Tabela 5.1: Sumário dos conjuntos. Conjunto de dados textuais estão marcados com um asterisco.

Nome	#Examplos	#Classes	#Atributos/Termos
Iris	150	3	4
Haberman	306	2	3
Glass	214	7	10
Wine	178	3	13
Balance	625	3	4
Ecoli	336	8	8
Transfusion	748	2	5
Mammographic-Masses	830	2	6
Cstr*	299	4	1726
Syskillwebert*	334	4	4340
Tr23*	204	6	5833

- *Iris*[2]: possui 3 classes de 50 instâncias cada, em que cada classe se refere a um tipo de planta íris.
- *Haberman's Survival*[3]: possui casos de estudo sobre sobrevivência de pacientes que passaram por cirurgia de câncer de mama, se sobreviveram 5 anos ou mais, ou morreram dentro de 5 anos.
- *Glass Identification*[4]: é resultado de estudo de classificação de tipos de vidro, motivado por uma investigação criminal.
- *Wine*[5]: resultado de uma análise química de vinhos de uma mesma região da Itália mas que derivaram em três diferentes cultivares.
- *Balance*[6]: gerados para modelar resultados de experimentos psicológicos.
- *Ecoli*[7]: é sobre localização celular de proteínas.
- *Blood Transfusion Service Center*[8]: dados do Centro de Serviços de Transfusão de Sangue em Hsin-Chu City, Taiwan.
- *Mammographic-Masses*[9]: discriminação de massas mamográficas benignas e malignas com base nos atributos do BI-RADS e na idade do paciente.
- *CSTR*[10]: possui resumos e relatórios técnicos publicados no Departamento de Ciência da Computação da Universidade de Rochester, de 1991 a 2007.

[2] https://archive.ics.uci.edu/ml/datasets/iris
[3] https://archive.ics.uci.edu/ml/datasets/haberman's+survival
[4] https://archive.ics.uci.edu/ml/datasets/glass+identification
[5] https://archive.ics.uci.edu/ml/datasets/wine
[6] https://archive.ics.uci.edu/ml/datasets/balance+scale
[7] https://archive.ics.uci.edu/ml/datasets/ecoli
[8] https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center
[9] http://archive.ics.uci.edu/ml/datasets/mammographic+mass
[10] http://sites.labic.icmc.usp.br/text_collections/CSTR.arff.zip

- *SyskillWebert*[11]: consiste em páginas da web sobre bandas, ovelhas, cabras e biomedicina.

- *Tr23*[12]: do conjunto 19MclassTextWc, derivado das coleções Trec-5, Trec-6 e Trec-7 de dados de Conferência de Recuperação de Texto.

5.2 Configuração Experimental

Foram avaliados três algoritmos de agrupamento: não supervisionado *k*-Means; e dois semissupervisionados, PCK-Means e MPCK-Means. Foi realizado o procedimento descrito no Capítulo 4. Os algoritmos foram rodados com as seguintes configurações: número máximo de iterações igual a 50, tolerância de inércia relativa para declarar convergência com valor $1e-4$, e o peso de cada restrição em PCK-Means e MPCK-Means com valor 1. Os parâmetros do algoritmo foram configurados, considerando os seguintes valores: número de dados rotulados foram semelhantes aos usados para conjuntos de dados numéricos e textuais, número de iterações máximas definida em 100, *kernels* lineares e RBF, escala e gama automática, e valores de nu variando de 0.05 a 0.95 com 0.5 de passo. Para conjunto de dados numéricos, número de *clusters* testados foram $\{2,4,6,10,20,40,60,80,100,120,140\}$ e número de dados rotulados da classe de interesse, $\{1,3,5,10,20,30\}$. Para conjunto de dados textuais, o número de *clusters* testados foram $\{2,6,10,40\}$ e número de dados da classe de interesse, $\{1,5,10,30\}$. Foi usado o algoritmo OC-SVM como baseline para comparações.

5.3 Resultados e Discussões

Nesta seção, apresentam-se os resultados de *k*-Means, PCK-Means, MPCK-Means e OC-SVM com diferentes configurações, conforme descrito na seção anterior. Como os conjuntos de dados numéricos e textuais são bastante diferentes, a análise desses dois cenários foram divididos.

5.3.1 Conjunto de dados numéricos

O *f-score* de diferentes algoritmos de agrupamento nos conjuntos de dados numéricos pode ser observada na Figura 5.1. É possível concluir que todos os algoritmos de agrupamento superaram o OC-SVM em todos os cenários. Algoritmos de agrupamento mostraram alguma similaridade comportamental, visto que aumentar o número de *clusters* melhorou o *f-score* até cerca de 40

[11] http://sites.labic.icmc.usp.br/text_collections/SyskillWebert.arff.zip
[12] http://sites.labic.icmc.usp.br/text_collections/tr23.arff.zip

clusters. Isso nos mostra que dividir a superfície de decisão para a classe positiva em subgrupos tende a aumentar a detecção de exemplos positivos. Quando o número de *clusters* é extremamente grande, grupos consistentes podem ser separados, prejudicando o procedimento de propagação do rótulo.

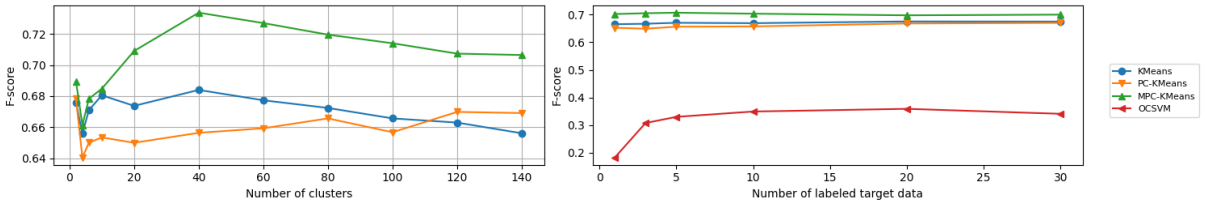


Figura 5.1: *F-score* médio sobre todos os conjuntos de dados numéricos.

Ao observar o número de dados inicialmente rotulados, continuar aumentando esse número traz uma pequena melhoria no desempenho dos algoritmos. Mas, novamente, OC-SVM foi superado em todas as comparações. Isso mostra que o processo de propagação do rótulo é eficiente e obtém bons resultados, mesmo com pouquíssimos exemplos rotulados.

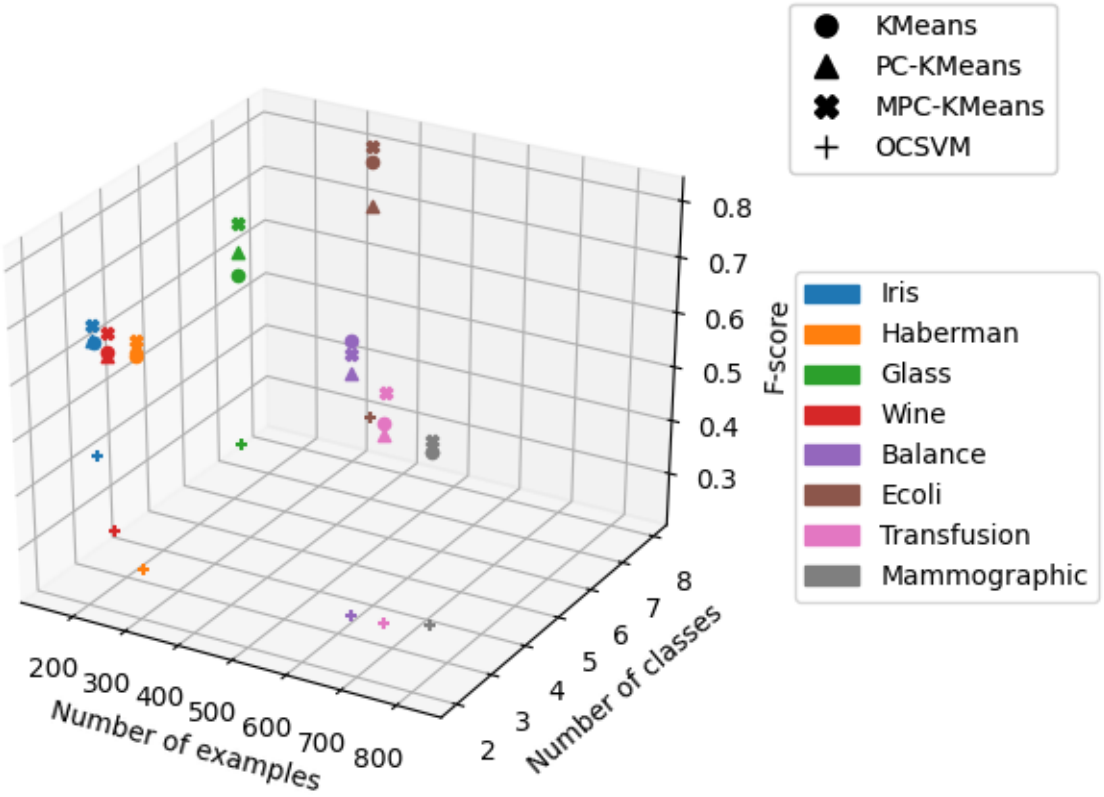


Figura 5.2: *F-score* médio por conjunto de dados, número de exemplos e classes.

Na Figura 5.2, são apresentados o *f-score* médio por conjunto de dados, ex-

plorando as características de cada conjunto de dados. Pode-se observar que MPCK-Means foi dominante na maioria dos conjuntos de dados. Conjuntos de dados semelhantes em tamanho, como Iris (azul), Haberman (laranja) e Wine (vermelho), têm resultados de *f-score* médios semelhantes. Além disso, todos os algoritmos comparados tendem a obter um melhor *f-score* em conjuntos de dados com mais classes como o Ecoli (marrom) e Glass (verde).

Os resultados detalhados para os conjuntos de dados numéricos são relatados nas Figuras 5.3-5.10. No geral, o aumento do número de *clusters* impactou positivamente conjuntos de dados maiores, como Mammographic, Transfusion e Balance. E com poucos dados de interesse rotulados 5 ou 10, ele já pode alcançar resultados melhores do que com mais dados rotulados. Os conjuntos de dados que tiveram melhor o *f-score* médio do MPCK-Means comparado a outros algoritmos são: Glass na Figura 5.5 e Transfusion na Figura 5.9, ao variar o número de *clusters*, o *f-score* se mantém estável e expressivamente melhor que demais algoritmos.

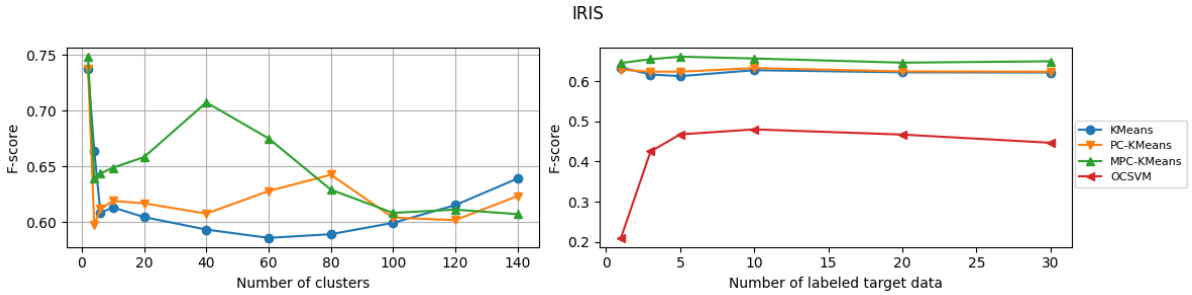


Figura 5.3: Iris: *f-score* médio por número de *clusters* e número de exemplos rotulados.

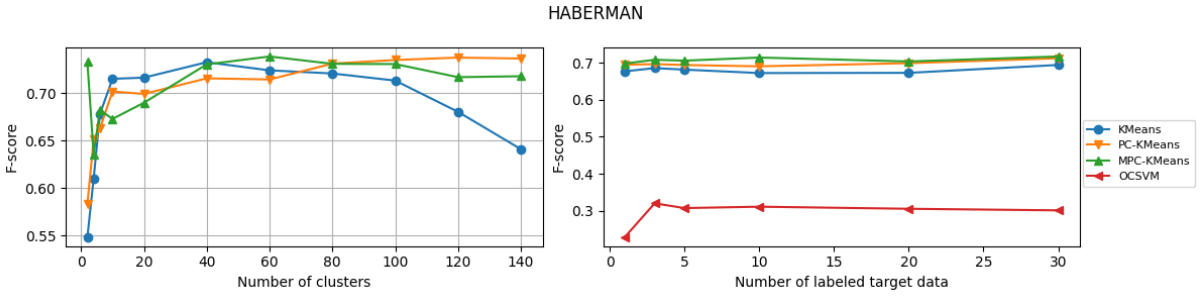


Figura 5.4: Haberman: *f-score* médio por número de *clusters* e número de exemplos rotulados.

5.3.2 Conjunto de dados textuais

Na Figura 5.11, pode-se analisar o impacto da variação do número de *clusters* e exemplos rotulados para conjuntos de dados textuais. Como nos conjuntos de dados numéricos, OC-SVM foi superado em todos os cenários, e

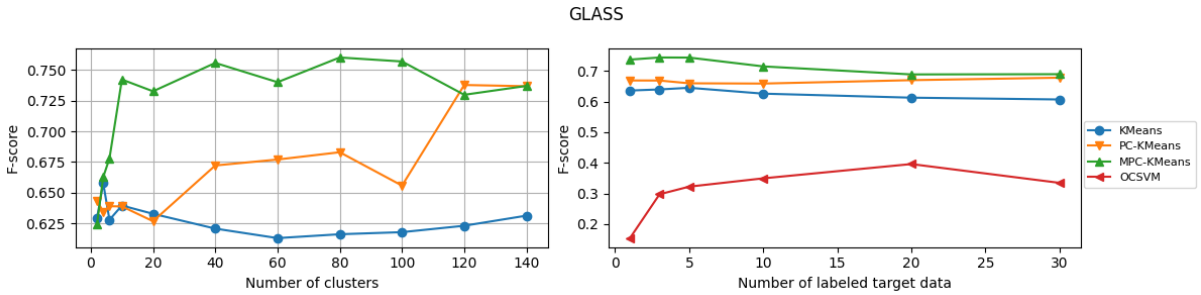


Figura 5.5: Glass: *f-score* médio por número de *clusters* e número de exemplos rotulados.

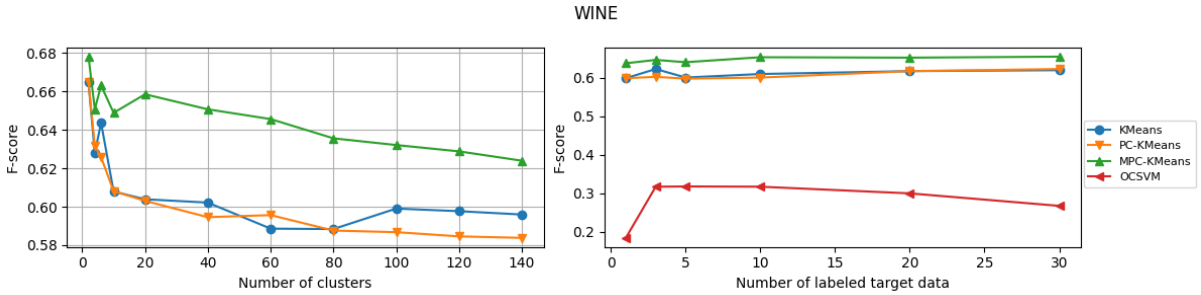


Figura 5.6: Wine: *f-score* médio por número de *clusters* e número de exemplos rotulados.

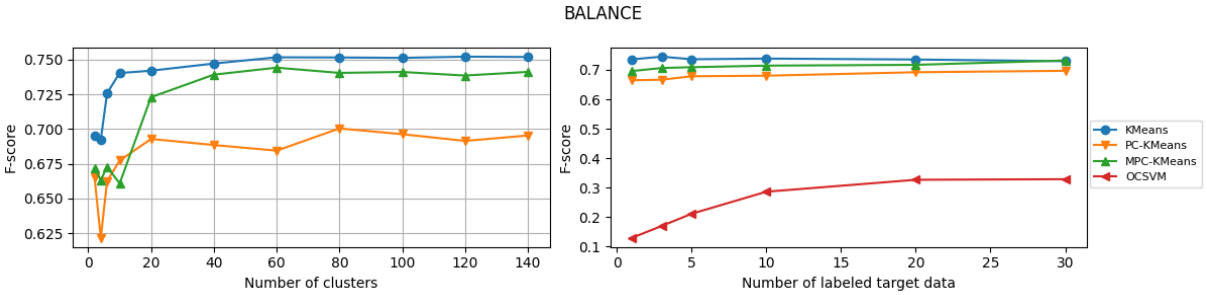


Figura 5.7: Balance: *f-score* médio por número de *clusters* e número de exemplos rotulados.

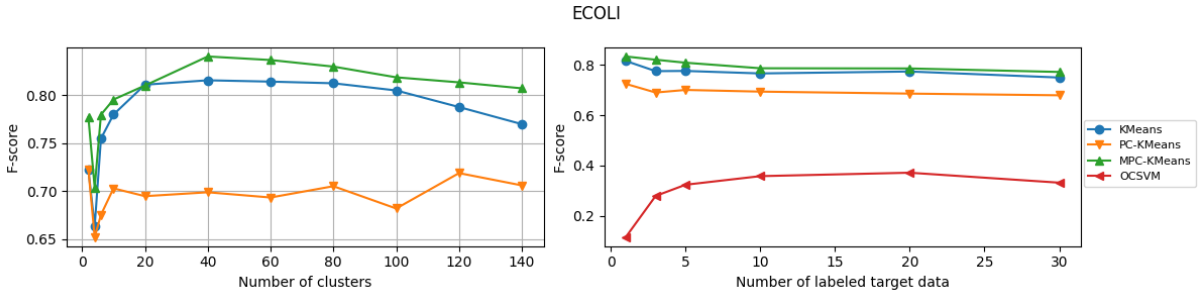


Figura 5.8: Ecoli: *f-score* médio por número de *clusters* e número de exemplos rotulados.

MPCK-Means dominou todas as avaliações. No entanto, o aumento do número de *clusters* não apresentou melhora significativa no desempenho dos algoritmos nos conjuntos de dados textuais. E, usando um número diferente

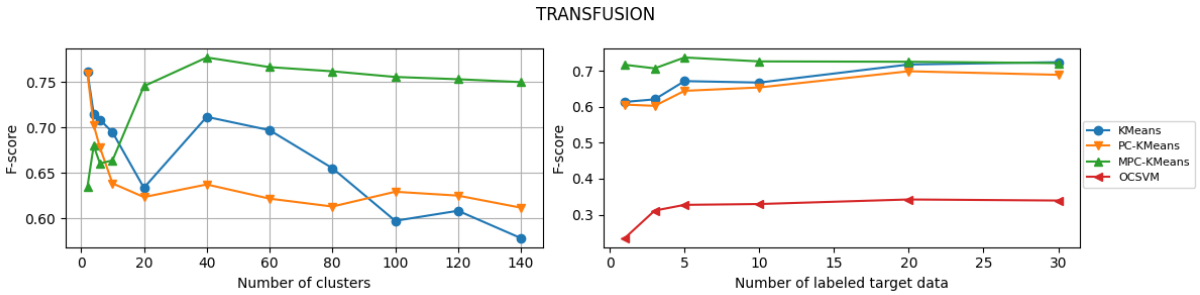


Figura 5.9: Transfusion: *f-score* médio por número de *clusters* e número de exemplos rotulados.

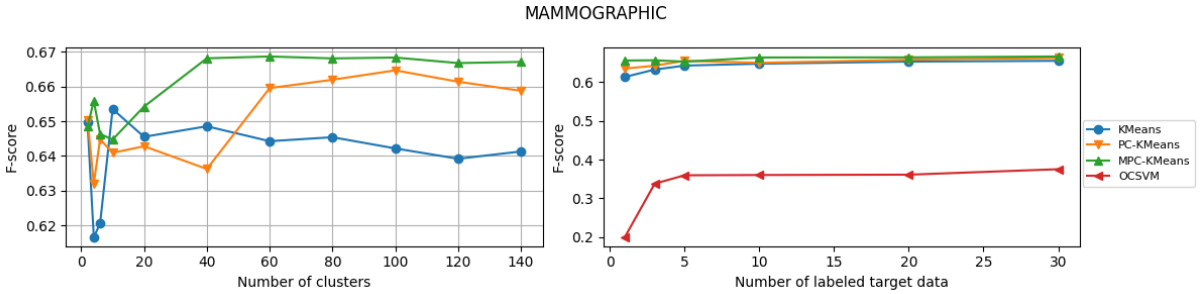


Figura 5.10: Mammographic: *f-score* médio por número de *clusters* e número de exemplos rotulados.

de dados do alvo rotulados, o resultado foi semelhante aos conjuntos de dados numéricos, nos quais usando cinco dados da classe de interesse tiveram principalmente os melhores resultados.

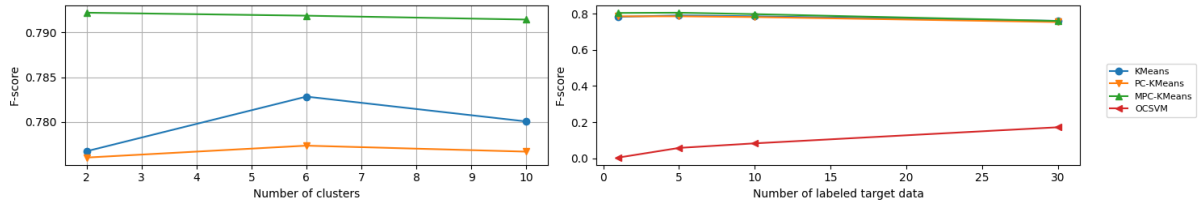


Figura 5.11: *F-score* médio sobre todos os conjuntos de dados textuais.

A análise separada para cada conjunto de dados pode ser observada nas Figuras 5.12 - 5.14. É possível observar que o comportamento de todos os algoritmos comparados foi semelhante junto com os conjuntos de dados, com um desempenho dominante do MPCK-Means. Este comportamento pode ser explicado porque os conjuntos de dados são muito semelhantes no número de exemplos e classes. Além disso, a abordagem transdutiva apresentada teve um desempenho satisfatório, uma vez que todos os métodos alcançaram os melhores resultados com muito poucos exemplos rotulados.

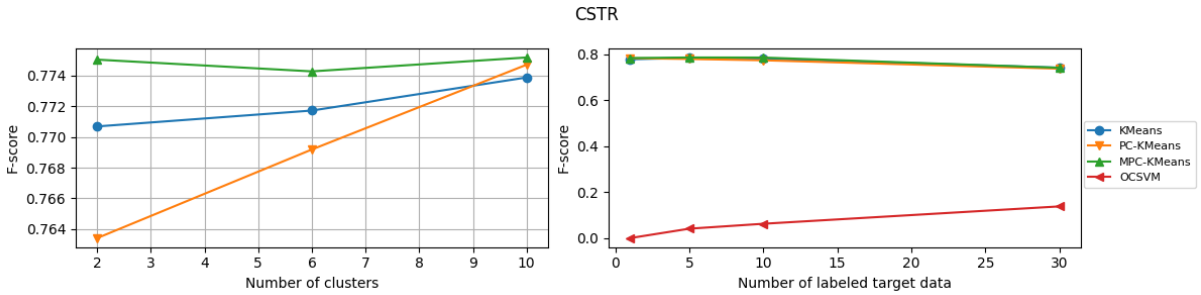


Figura 5.12: CSTR: *f-score* médio por número de *clusters* e número de exemplos rotulados.

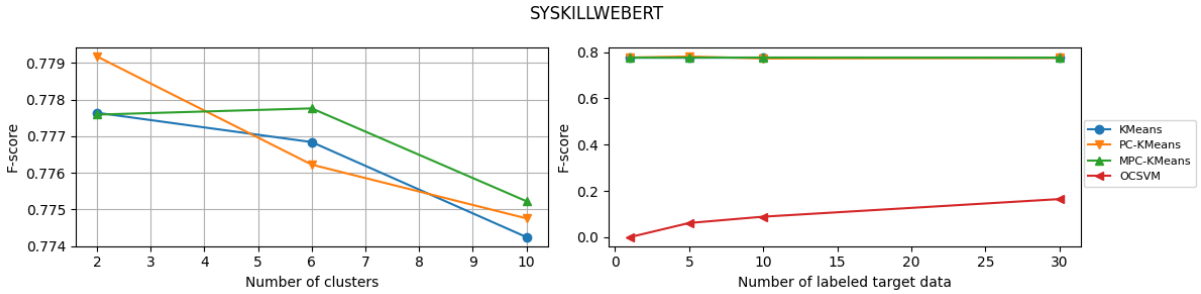


Figura 5.13: Syskillwebert: *f-score* médio por número de *clusters* e número de exemplos rotulados.

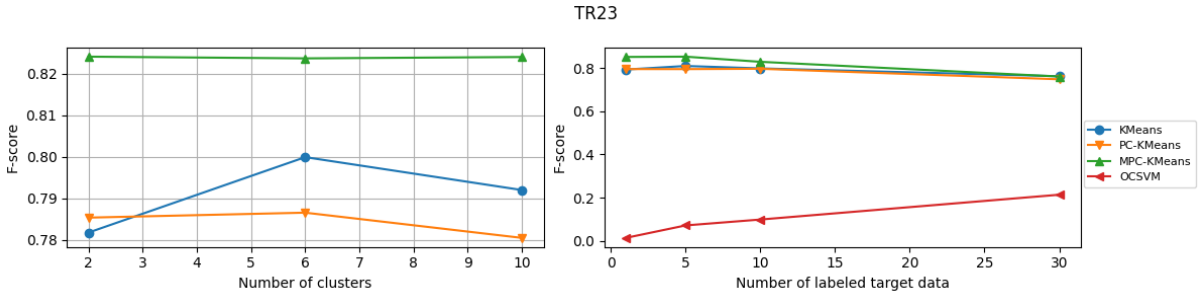


Figura 5.14: Tr23: *f-score* médio por número de *clusters* e número de exemplos rotulados.

5.4 Considerações Finais

Este trabalho apresenta uma abordagem de agrupamento semissupervisionado para PUL e uma abordagem de propagação transdutiva de rótulos usando resultados de agrupamento. Foram comparados os resultados do uso de um algoritmo de agrupamento não supervisionado K-Means e dois algoritmos de agrupamento semissupervisionado, PCK-Means e MPCK-Means. OC-SVM, o algoritmo de OCC mais conhecido, foi usado como *baseline*. Para isso, foram avaliados oito conjuntos de dados numéricos do repositório UCI e três bancos de dados textuais. Nesses experimentos, todos os algoritmos de agrupamento superaram o OC-SVM. Também observou-se que MPCK-Means pareceu promissor e obteve melhores resultados do que outros algoritmos.

Aumentar o número de *clusters* tende a melhorar a detecção de exemplos positivos até cerca de 40 *clusters* e, quando esse número é enorme, prejudica o procedimento de propagação de rótulos. Além disso, MPCK-Means apresentou bom desempenho, mesmo quando usando muito poucos exemplos rotulados. Isso indica que aprender métricas para distorcer o espaço de exemplos e usar procedimento de propagação de rótulo proposto é útil para aplicações OCC e PUL.

No próximo capítulo, as conclusões finais desse projeto são apresentadas, juntamente com as principais contribuições e limitações da abordagem proposta, e uma breve direção dos trabalhos futuros.

CAPÍTULO

6

Conclusões

Com o avanço da tecnologia, cada vez mais dados são gerados e compartilhados na internet, geralmente de forma não estruturada e sem rótulo. Rotular esses dados é um processo caro em tempo e esforço humano, além de que, os conjuntos de dados rotulados existentes são amostras pequenas do universo do qual fazem parte e novas classes emergem com o tempo, sendo difícil prever todas as classes que podem existir em um domínio. O PUL se aplica à tarefa de classificação onde apenas dados positivos estão presentes e procura aprender um limite que descreve os dados positivos ou classe de interesse. Existem propostas para agrupamento não supervisionado na literatura que se destacam nesta tarefa de classificação de uma única classe. Este trabalho apresentou uma abordagem usando agrupamento semissupervisionado, com inclusão de conhecimento supervisionado na forma de restrições a fim de melhorar os resultados de agrupamento, além de propor um processo de propagação de rótulos, conforme os resultados e discussões apresentados no capítulo anterior.

Neste capítulo, as principais contribuições desse projeto são elencadas na Seção 6.1, algumas limitações dessa abordagem são explicitadas na Seção 6.2, e por fim, possíveis trabalhos futuros são listados na Seção 6.3.

6.1 Contribuições

- PUL-SSC: novo algoritmo proposto que é a primeira abordagem de agrupamento semissupervisionado para a tarefa de PUL. Com poucos exemplos rotulados da classe de interesse, PUL-SSC tem um processo iterativo de delimitação da classe positiva: faz agrupamento dos dados de treino,

cria restrições *must-link* entre os dados da classe de interesse e os dados não rotulados que estão no mesmo grupo desses dados positivos e entre os dados não rotulados do grupo mais próximo aos grupos positivos, e restrições *cannot-link* entre os dados da classe de interesse e os dados rotulados do grupo mais distante dos grupos positivos;

- Processo de aprendizado transdutivo de propagação de rótulos proposto: em que os exemplos não rotulados do grupo mais distante são fortes negativos, e são rotulados como negativos, e os exemplos não rotulados do grupo mais próximo dos grupos positivos são fortes positivos, e rotulados como positivos. Esse processo de verificação de grupo mais próximo e distante, e posterior, rotulação como positivo e negativo, ocorre até que não existam mais grupos sem rótulo;

- Os resultados apontam que com o algoritmo proposto com MPCK-Means é possível ter resultados de agrupamento melhores do que abordagens não supervisionadas (como *k*-Means), e metódos estátisticos (como OC-SVM), sendo promissor explorar mais a fundo esse assunto.

6.2 Limitações

Embora MPCK-Means pareça promissor para conjunto de dados textuais, para a representação BOW usada, calcular as matrizes para aprendizado de métricas foi um processo caro computacionalmente, uma vez que os dados textuais são muito esparsos, e para conjunto de dados muito grandes torna-se pouco viável sem uso de supercomputadores com grande capacidade de memória e processamento.

O uso de algoritmos de agrupamento particionais usados, exigem a definição de número k de *clusters*, o que exige *fine-tuning*. E o processo de agrupamento pode ser impactado negativamente quando os centroides iniciais são ruins.

6.3 Trabalhos Futuros

Futuramente, pretende-se:

- Usar outras representações para dados textuais como *embeddings* para aprendizado de métricas;

- Explorar outros algoritmos de agrupamento semissupervisionado em trabalhos futuros, como algoritmos de agrupamento por densidade ou hierárquicos;

- Estudar novas formas de propagação de rótulos e melhores maneiras de definir os limites positivos da classe;
- Avaliar outras funções de distância entre os centroides, no ranking de *clusters* mais próximos ou mais distantes, como a distância de cossenos;
- Analisar o impacto das restrições geradas e a quantidade de restrições que o algoritmo consegue seguir.

Referências Bibliográficas

A. Ahmad and S. S. Khan. Survey of state-of-the-art mixed data clustering algorithms. 2019. Citado nas páginas 5 e 6.

S. Alam, S. K. Sonbhadra, S. Agarwal, and P. Nagabhushan. One-class support vector classifiers: A survey. 2020. Citado na página 33.

H. Alashwal, S. Deris, and R. M. Othman. One-class support vector machines for protein-protein interactions prediction. 2006. Citado na página 33.

M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut. A brief survey of text mining: Classification, clustering and extraction techniques. 2017. Citado na página 30.

M. Anderka, B. Stein, and N. Lipka. Detection of text quality flaws as a one-class classification problem. 2011. Citado na página 33.

E. Bair. Semi-supervised clustering methods. 2013. Citado na página 5.

S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. 2002. Citado nas páginas 2, 5, 6, 7, 8, e 18.

S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. 2004. Citado nas páginas 6 e 10.

S. Basu, I. Davidson, and K. Wagstaff. *Constrained clustering: Advances in algorithms, theory, and applications*. 2008. Citado na página 5.

J. Bekker and J. Davis. Learning from positive and unlabeled data: A survey. 2020. Citado nas páginas 1, 2, 23, 24, e 25.

C. Bellinger, S. Sharma, and N. Japkowicz. One-class versus binary classification: Which and when? 2012. Citado na página 21.

M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. 2004. Citado nas páginas 2, 5, 6, 13, e 18.

C. M. Bishop. Novelty detection and neural network validation. 1994. Citado na página 22.

C. Campbell and K. P. Bennett. A linear programming approach to novelty detection. 2001. Citado na página 25.

J. Chen, S. Sathe, C. Aggarwal, and D. Turaga. Outlier detection with autoencoder ensembles. 2017. Citado na página 32.

F. De Comité, F. Denis, R. Gilleron, and F. Letouzey. Positive and unlabeled examples help learning. 1999. Citado na página 25.

F. Denis, R. Gilleron, and F. Letouzey. Learning from positive and unlabeled examples. 2005. Citado nas páginas 23 e 32.

C. Désir, S. Bernard, C. Petitjean, and L. Heutte. One class random forests. 2013. Citado na página 25.

M. Du Plessis, G. Niu, and M. Sugiyama. Convex formulation for learning from positive and unlabeled data. 2015. Citado na página 24.

S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. 2016. Citado nas páginas 2 e 25.

K. Ganesan, U. R. Acharya, C. K. Chua, C. M. Lim, and K. T. Abraham. One-class classification of mammograms using trace transform functionals. 2013. Citado na página 25.

A. B. Gardner, A. M. Krieger, G. Vachtsevanos, and B. Litt. One-class novelty detection for seizure analysis from intracranial eeg. 2006. Citado na página 33.

M. Gôlo, R. Marcacini, and R. Rossi. Uma extensa avaliação empírica de técnicas de pré-processamento e algoritmos de aprendizado supervisionado de uma classe para classificação de texto. 2019. Citado nas páginas 2, 3, 25, 26, 30, 31, e 39.

T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. 1985. Citado na página 13.

J. H. Janssens, I. Flesch, and E. O. Postma. Outlier detection with one-class classifiers from ml and kdd. 2009. Citado na página 25.

N. Japkowicz. Concept-learning in the absence of counter-examples: an autoassociation-based approach to classification. 1999. Citado na página 22.

N. Japkowicz, C. Myers, M. Gluck, et al. A novelty detection approach to classification. 1995. Citado na página 22.

K. Jaskie and A. Spanias. Positive and unlabeled learning algorithms and applications: A survey. 2019. Citado nas páginas 1, 2, 23, e 25.

S. S. Khan and M. G. Madden. A survey of recent trends in one class classification. 2009. Citado nas páginas 1, 22, 23, 24, e 33.

M. Koppel and J. Schler. Authorship verification as a one-class classification problem. 2004. Citado na página 32.

B. Krawczyk, M. Woźniak, and B. Cyganek. Clustering-based ensembles for one-class classification. 2014. Citado nas páginas 2 e 25.

W. S. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. 2003. Citado na página 25.

Q. Leng, H. Qi, J. Miao, W. Zhu, and G. Su. One-class classification with extreme learning machine. 2015. Citado nas páginas 2 e 25.

F. Letouzey, F. Denis, and R. Gilleron. Learning from positive and unlabeled examples. 2000. Citado nas páginas 2 e 25.

K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu. Improving one-class svm for anomaly detection. 2003. Citado na página 33.

X. Li and B. Liu. Learning to classify texts using positive and unlabeled data. 2003. Citado nas páginas 25 e 32.

X.-L. Li and B. Liu. Learning from positive and unlabeled examples with different data distributions. 2005. Citado na página 23.

B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. 2002. Citado nas páginas 2, 25, e 32.

B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. Building text classifiers using positive and unlabeled examples. 2003. Citado na página 23.

Y. Liu and M. G. Madden. One-class support vector machine calibration using particle swarm optimisation. 2007. Citado na página 25.

L. M. Manevitz and M. Yousef. One-class svms for document classification. 2001. Citado nas páginas 2, 25, e 33.

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. 2008. Citado nas páginas 14 e 16.

F. Mordelet and J.-P. Vert. A bagging svm to learn from positive and unlabeled examples. 2014. Citado na página 23.

M. M. Moya and D. R. Hush. Network constraints and multi-objective optimization for one-class classification. 1996. Citado na página 25.

M. M. Moya, M. W. Koch, and L. D. Hostetler. One-class classifier networks for target recognition applications. 1993. Citado nas páginas 1, 2, 21, 22, e 25.

J. Muñoz-Marí, F. Bovolo, L. Gómez-Chova, L. Bruzzone, and G. Camp-Valls. Semisupervised one-class support vector machines for classification of remote sensing data. 2010. Citado nas páginas 2 e 25.

D. T. Munroe and M. G. Madden. Multi-class and single-class classification approaches to vehicle model recognition from images. 2005. Citado na página 33.

B. M. Nogueira. *Hierarchical semi-supervised confidence-based active clustering and its application to the extraction of topic hierarchies from document collections*. PhD thesis, 2013. Citado nas páginas 2 e 5.

P. Oza and V. M. Patel. Active authentication using an autoencoder regularized cnn-based one-class classifier. 2019. Citado na página 32.

X. Peng, S. Xiao, J. Feng, W.-Y. Yau, and Z. Yi. Deep subspace clustering with sparsity prior. 2016. Citado na página 32.

R. Perdisci, G. Gu, and W. Lee. Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. 2006. Citado na página 33.

P. Perera and V. M. Patel. Learning deep features for one-class classification. 2019. Citado nas páginas 2 e 25.

P. Perera, P. Oza, and V. M. Patel. One-class classification: A survey. 2021. Citado nas páginas 2, 25, 32, e 33.

S. Pidhorskyi, R. Almohsen, D. A. Adjeroh, and G. Doretto. Generative probabilistic novelty detection with adversarial autoencoders. 2018. Citado na página 32.

M. F. Porter et al. An algorithm for suffix stripping. 1980. Citado na página 41.

D. Ridder. An experimental comparison of one-class classification methods. 1998. Citado nas páginas 2 e 25.

G. Ritter and M. T. Gallegos. Outliers in statistical pattern recognition and an application to automatic chromosome classification. 1997. Citado na página 22.

R. G. Rossi, R. M. Marcacini, and S. O. Rezende. Benchmarking text collections for classification and clustering tasks. 2013. Citado na página 41.

L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft. Deep one-class classification. 2018. Citado nas páginas 2 e 25.

M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli. Adversarially learned one-class classifier for novelty detection. 2018. Citado na página 25.

M. Salehi, A. Arya, B. Pajoum, M. Otoofi, A. Shaeiri, M. H. Rohban, and H. R. Rabiee. Arae: Adversarially robust training of autoencoders improves novelty detection. 2020. Citado na página 32.

C. Sanchez-Hernandez, D. S. Boyd, and G. M. Foody. One-class classification for mapping a specific land-cover class: Svdd classification of fenland. 2007. Citado na página 25.

P. Schlachter, Y. Liao, and B. Yang. Deep one-class classification using data splitting. 2019. Citado na página 25.

B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. 2000. Citado nas páginas 2, 24, 26, e 33.

H. J. Shin, D.-H. Eom, and S.-S. Kim. One-class support vector machines—an application in machine fault detection and classification. 2005. Citado na página 33.

E. J. Spinosa and A. C. P. de Leon Ferreira. Support vector machines for novel class detection in bioinformatics. 2004. Citado na página 33.

D. Sun, Q.-A. Tran, H. Duan, and G. Zhang. A novel method for chinese spam detection based on one-class support vector machine. 2005. Citado na página 33.

D. M. Tax and R. P. Duin. Data domain description using support vectors. 1999a. Citado nas páginas 2 e 24.

D. M. Tax and R. P. Duin. Support vector domain description. 1999b. Citado nas páginas 2 e 24.

D. M. Tax and R. P. Duin. Uniform object generation for optimizing one-class classifiers. 2001. Citado nas páginas 24 e 33.

D. M. J. Tax. One-class classification: Concept learning in the absence of counter-examples. 2002. Citado nas páginas 1, 21, 22, e 23.

H. T. Tran and D. Hogg. Anomaly detection using a convolutional winner-take-all autoencoder. 2017. Citado na página 32.

J. E. Van Engelen and H. H. Hoos. A survey on semi-supervised learning. 2020. Citado nas páginas 5 e 6.

K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. Constrained k-means clustering with background knowledge. 2001. Citado nas páginas 5, 6, 7, e 18.

Q. Wang, L. S. Lopes, and D. M. Tax. Visual object recognition through one-class learning. 2004. Citado na página 33.

E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng. Distance metric learning with application to clustering with side-information. 2003. Citado nas páginas 5, 8, 9, 11, e 18.

H. Yu. Single-class classification with mapping convergence. 2005. Citado nas páginas 2 e 25.

H. Yu, C. Zhai, and J. Han. Text classification from positive and unlabeled documents. 2003. Citado nas páginas 2, 25, 27, e 32.

H. Yu, J. Han, and K.-C. Chang. Pebl: Web page classification without negative examples. 2004. Citado nas páginas 2, 25, 27, 30, 32, e 36.

Z. Zeng, Y. Fu, G. I. Roisman, Z. Wen, Y. Hu, and T. S. Huang. One-class classification for spontaneous facial expression analysis. 2006. Citado na página 33.

Y. Zhang, X. Li, and M. Orlowska. One-class classification of text streams with concept drift. 2008. Citado na página 33.

Y. Zhao, B. Li, X. Li, W. Liu, and S. Ren. Customer churn prediction using improved one-class support vector machine. 2005. Citado na página 25.

X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. 2009. Citado na página 6.