UFPE

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FILIPE MENDES MARIZ

AVALIAÇÃO E COMPARAÇÃO DE VERSÕES MODIFICADAS DO ALGORITMO KNN

RECIFE

DEZEMBRO DE 2017

FILIPE MENDES MARIZ

AVALIAÇÃO E COMPARAÇÃO DE VERSÕES MODIFICADAS DO ALGORITMO KNN

Trabalho de Conclusão de Curso
apresentado como requisito parcial para a
obtenção do grau de Bacharel em Ciência da
Computação no Centro de Informática da
Universidade Federal de Pernambuco

RECIFE

TERMO DE APROVAÇÃO

FILIPE MENDES MARIZ

AVALIAÇÃO E COMPARAÇÃO DE VERSÕES MODIFICADAS DO ALGORITMO KNN

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal de Pernambuco como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação, pela seguinte banca examinadora:

_____

Prof. Dr. Leandro Maciel Almeida

Recife, Dezembro de 2017

DEDICATORIA

Aos meus pais e minha irmã que acreditaram em mim, me apoiaram e me ofereceram paciência e compreensão

AGRADECIMENTOS

Ao professor orientador deste trabalho, Prof. Cleber Zanchettin pelo acompanhamento e orientação.

Ao Centro de Informática.

EPIGRAFO

RESUMO

O algoritmo *k-Nearest Neighbors* é famoso por ser um dos mais simples, porém efetivos métodos de classificação e regressão. Este trabalho tem como objetivo explorar à área de aprendizado de máquina e o grupo de métodos *k-Nearest Neighbors*, analisando suas potencialidades. Isso é feito através da implementação de diferentes versões do algoritmo disponíveis na literatura além da comparação destas sobre um conjunto de bases de dados, de forma a verificar o que estas modificações alteram nos resultados do modelo. Foram escolhidos três algoritmos diferentes para serem implementados e testados, além de uma implementação parcial de um método de clustering para demonstrar seus efeitos.

Palavra-chaves: Classificação, *k-Nearest Neighbor*, Aprendizado de Máquina, Inteligência Artificial, *Big Data*.

ABSTRACT

k-Nearest Neighbors is a popular machine learning algorithm due to its simple implementation yet effective classification and regression results. This work aims to analyze the algorithm and explore the possible modifications that can be applied to it. Using of articles found in the literature, three possible modifications that can be applied to it were implemented and tested over the same datasets. We also used a partial implementation of a clustering algorithm available in one of the papers used aiming to understand its effect on the precision. We then discuss about the results found and the effects that each modification caused on it.

Keywords: Classification, k-Nearest Neighbor, Machine Learning, Artificial Intelligence, Big Data.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS

k-NN / kNN - *k Nearest Neighbors*
LMKNN - *local mean-based k-nearest neighbor*
MLM-KHNN - *multi-local means-based k-harmonic nearest neighbor*

LISTA DE TABELAS

SUMÁRIO

1.INTRODUÇÃO

Inteligência artificial é uma subárea da computação que no decorrer das décadas vem ganhando mais e mais importância. Ela tem como foco dar às máquinas a capacidade de simular algum aspecto de inteligência, seja aprendendo ou realizando tarefas complexas que previamente só humanos eram capazes de fazer.

Com o novo milênio, a extração, catalogação e armazenamento de informações se tornou algo comum, e isso levou a geração desenfreada de informações. O imenso volume de dados disponíveis deu origem ao conceito chamado de Big data (SAGIROGLU e SINANC, 2013), base de dados de tamanho gigantescos. Essas bases de dados são consideradas importantes devido ao fato de serem representações da realidade, mas devido ao seu tamanho se tornam indecifráveis por humanos.

Para isso, foram desenvolvidas máquinas capazes de identificar padrões e aprender sobre o contexto utilizando a base de dados que lhe é alimentada. Esse conceito de extração de informações úteis de bancos de dados foi chamado de Data Mining (CHAKRABATI et al., 2006). Através de conceitos de inteligência artificial e aprendizado de máquina, se tornou viável ter uma máquina que aprendesse com os dados informações relevantes, além de permitir que essas máquinas reconheçam padrões imperceptíveis para seres humanos.

Uma das mais importantes aplicações de Aprendizado de Máquina é predição. Estudos nessa área têm como intuito prever o resultado que um grupo de parâmetros terá baseado num grupo de dados que foram utilizados para treinar uma certa máquina.

Predições podem variar em dois tipos: Classificação e Regressão. Classificação têm como foco definir a que grupo uma amostra de teste pertence, de forma que esta máquina consegue definir áreas específicas que cada grupo possui. Regressão por outro lado, é um método previsão que têm foco em, dado um grupo de valores, conseguir fazer a previsão de um valor utilizando do banco de dados que ele foi alimentado (IMANDOUST e BOLANDRAFTAR, 2013). Um exemplo de classificação seria um algoritmo que consegue discernir entre bons pagadores e mal

pagadores baseados nos dados de um cliente; um exemplo de regressão seria um algoritmo capaz de saber qual o valor que tal cliente estaria disposto a pagar para certo tipo de produto. Este trabalho irá se restringir ao primeiro grupo por motivos de simplicidade.

Devido à sua utilidade, a área de mineração de informações foi sendo aperfeiçoada através dos anos, dando origens a algoritmos mais precisos, rápidos, leves ou específicos para um cenário estudado. Um dos métodos mais direto é conhecido como k-Nearest Neighbors (KOTSIANTIS et al. 2007).

O algoritmo k-Nearest Neighbors é um antigo e popular algoritmo de classificação e regressão. Sua popularidade se dá à sua robustez, flexibilidade e simplicidade de implementação que ainda assim têm uma boa precisão em vários domínios. Mesmo sendo um algoritmo antigo, ainda há uma busca por melhorias nele.

Isso é devido ao fato de que algoritmos do método k-NN ainda hoje recebem muita atenção na área acadêmica e no mercado. Eles são utilizados em áreas como mineração de texto, simulação de precipitação de chuva e clima em fazendas, predição de mercado de ação, predição de se um paciente terá câncer de próstata ou a chance de um paciente que foi internado por um ataque cardíaco irá sofrer outro ataque cardíaco (IMANDOUST e BOLANDRAFTAR, 2013). Assim, ele é um algoritmo que possui importância, sendo utilizado e explorado de diferentes maneiras ainda hoje.

Através de algumas pesquisas sobre o algoritmo, encontramos artigos que especificam diferentes maneiras de abordar certas partes deste, argumentando que a sua abordagem levava a uma taxa de erro menor e resistência ao parâmetro de número de vizinhos (PAN et al., 2017), um novo método de escolha do número de vizinhos (GARCÍA-PEDRAJAS et al. , 2017) ou que tem uma precisão maior em bases de dados que com alta dimensionalidade (TOMAŠEV et al. , 2015). Interessados em aprender mais sobre as metodologias utilizadas e verificar sua eficácia, escolhemos esses algoritmos para analisar no decorrer deste trabalho.

Com o intuito de introduzir o leitor às ideias por trás do algoritmo, esse trabalho irá dar uma visão geral sobre a área e outros fatores relevantes para o entendimento deste trabalho. Em seguida, irá abordar o algoritmo *k-NN* e algumas

de suas modificações. Analisaremos um subgrupo dessas modificações que podem ser aplicadas para melhorar métricas como tempo de classificação, número de vizinhos e precisão.

Escolhemos 3 algoritmos específicos da literatura para comparar, o *Hubness-based Fuzzy* proposto por TOMAŠEV et al. (2014), o *Local-best k* proposto por GARCÍA-PEDRAJAS et al. (2017) e o *MLM-KHNN* proposto por PAN et al. (2017). Esses algoritmos foram escolhidos devido a suas diferentes maneiras de abordar a classificação baseada em seus vizinhos e bases de dados alvos. Além desses, utilizamos de uma implementação parcial de um algoritmo proposto por ZHENG et al. (2017), um método de redução de amostras chamado de *Improved K-Nearest Neighbour Clustering*. Implementamos este algoritmo com a intenção de testar como métodos de clustering podem afetar as métricas dos algoritmos.

Todas as modificações serão testadas em um grupo de bases de dados obtidas no site UCI[1], tanto por estarem sendo utilizadas em um ou mais dos algoritmos que estamos avaliando, quanto por serem de fácil acesso. O subconjunto de bases utilizando busca abordar diferentes de características de bases de dados, propriedades como alta dimensionalidade, grandes bases de dados e bases desbalanceadas.

Serão utilizadas métricas para analisarmos e dissertar sobre os efeitos que as modificações em cada algoritmo causou, além de verificarmos como os algoritmos se comportam com cada base de dados. Concluiremos com o que foi aprendido com a realização deste trabalho, as análises de resultados e sobre possíveis futuras pesquisas.

[1] http://archive.ics.uci.edu/ml

1.1 OBJETIVO

O objetivo deste trabalho é explorar mais a fundo os conceitos dos algoritmos baseados na estratégia de aprendizado baseado em instâncias, especificamente no grupo de algoritmos *k-NN*. Faremos isso implementando diferentes métodos apresentados nos artigos de referência e realizando testes com estes em algumas bases de dados considerando diferentes métricas.

Os testes serão realizados em bases de dados do *UCI Machine Learning Repository* e faremos comparações tanto com o *k-NN* básico como também com as outras versões implementadas, verificando se as métricas variam e analisando como e porque elas variam, considerando as observações dos autores dos artigos escolhidos.

A implementação deste projeto pode ser encontrada no repositório online no Github.[2]

1.2 ESTRUTURA DO DOCUMENTO

No capítulo 2, faremos uma breve revisão dos conceitos da área de inteligência artificial, big data, aprendizado de máquina e aprendizado baseado em instâncias. Definiremos cada um desses conceitos e faremos uma breve descrição deles, mostrando o motivo do interesse em cada um deles.

No capítulo 3, exploraremos o grupo de métodos *k-Nearest Neighbours*. Nele descreveremos como seu algoritmo funciona e examinaremos como características como número de amostras, número de vizinhos, cálculo de distância, peso e dimensionalidade afetam ele. Este capítulo tem como foco dar uma base para conceitos que vão ser falados nos capítulos a seguir.

O capítulo 4 será voltado aos artigos que implementados, examinando a fundo cada um de seus algoritmos. Nesta parte faremos uma análise do que cada uma das modificações que o algoritmo faz têm como objetivo, e explicaremos o motivo por que essas modificações tem esses efeitos.

[2] https://github.com/fmm4/kNN-Comparison

O capítulo 5 será voltado aos testes realizados. Nós falaremos sobre cada base de dados utilizadas, descrevendo elas e justificando seus usos, especificaremos qual foi a metodologia utilizada para os testes e então faremos uma análise de seus resultados. Nessa análise, verificaremos e compararemos cada uma das métricas obtidas e faremos observações sobre os motivos que tais valores foram obtidos.

Concluiremos no capítulo 6, onde faremos um resumo das informações que obtivemos com a análise do funcionamento de cada um e a comparação de seus resultados, além de dar expor algumas ideias de possíveis melhorias.

2. CONCEITOS DA ÁREA

2.1. INTELIGÊNCIA ARTIFICIAL

De acordo com NILSSON (2014), tarefas como entender linguagem, dirigir um veículo ou que requerem um esforço mental são consideradas tarefas que precisam de inteligência para ser realizadas. Nas últimas décadas, máquinas se tornaram mais sofisticadas, sendo capazes de realizar várias tarefas que anteriormente somente um humano pensante era capaz de realizar. NILSSON (2014) indica que essas máquinas contêm o que pode ser chamado de uma Inteligência Artificial.

A definição do que é ser inteligente ainda é discutida em várias áreas, não só da computação, o que torna uma definição concreta de um conceito associado à ela controverso. No domínio de inteligência artificial, NORVIG e RUSSEL (2013) descrevem que autores utilizam de diferentes definições para inteligência dependendo do agir ou pensar, e se a inteligência é perfeitamente racional ou possui emoções, como a inteligência humana.

Segundo SAGE (1999, apud HAYKIN, 2001), a área de inteligência artificial tem foco no desenvolvimento de máquinas capazes de realizar tarefas as quais tem como requerimento a capacidade de pensamento, e que qualquer inteligência artificial deve possuir pelo menos três características: a capacidade de armazenar conhecimento, aplicar o conhecimento armazenado e de aprender com novas experiências.

2.2. *BIG DATA*

Com os avanços tecnológicos das últimas decadas, o nível de informação bruta disponível aumentou de forma desenfreada, quase duplicando a cada dois anos. (LOHR, 2012). Isso vindo tanto da catalogação e digitalização de informações quanto da nova geração de sensores e equipamentos eletrônicos (EATON et al., 2012; SCHNEIDER, 2012; apud SINANC, SAGIROGLU, 2013).

SINANC e SAGIROGLU (2013) definem o termo Big Data como bases de dados gigantescas cuja sua representação, armazenamento e análise é difícil. Tal informação contém um volume de dados importantes para empresas e organizações quando analisadas corretamente.

Várias áreas beneficiam de big data, como saúde, mercados e localização pessoal para rotas ou propagandas baseadas na sua localização. Assim existe um grande mercado interessado em melhorar a precisão das informações obtidas (MICHALSKI et al., 2013).

2.3. APRENDIZADO DE MÁQUINA

Aprendizado de Máquina é definido por GARBONELL et al. (2013) como o estudo e a modelagem de processos de aprendizados em qualquer de suas múltiplas maneiras. Diferentes sistemas de aprendizado foram desenvolvidos, utilizando desde Inteligência Artificial como os baseados em Lógica; como estatísticos como bayesiano (HAYKIN, 2007).

Uma das mais importantes aplicações de aprendizado de máquina se dá à criação de sistemas de mineração de dados (KOTSIANTIS, 2006). Devido a sua necessidade de casos anteriores para aprender sobre um certo cenário, o algoritmo precisa de uma base de dados para ter resultados bons. É aí que a união de big data e Aprendizado de Máquina se torna importante.

2.4. APRENDIZADO DE MÁQUINA BASEADO EM INSTÂNCIAS

Aprendizado de Máquina Indutivo é o desenvolvimento de algoritmos capazes de aprender um comportamento baseados num grupo de instâncias, as quais serão consideradas seu conjunto de treinamento, de forma que ele seja capaz no futuro de gerar seu próprio resultado baseado no que seu treinamento o ensinou. O desenvolvimento de técnicas pode ser baseado em diferentes metodologias: Baseados em lógica como Árvores de Decisão (MURTHY, 1998 apud KOTSIANTIS et al. 2007, p. 163), utilizando de *Perceptrons* como WINNOW (LITTLESTONE e WARMUTH. 1994 apud KOTSIANTIS et al. 2007, p. 168), baseados em estatística

como Naive Bayes e os baseados em instâncias, como o *kNN* (KOTSIANTIS et al. 2007).

Algoritmos do tipo aprendizado baseado em instâncias são algoritmos do tipo *lazy-learning*, onde a computação só é realizada no momento que a tarefa de predição é executada, e assim não ocorre uma fase de treinamento no algoritmo. Em compensação, a fase de classificação é onde toda computação ocorre. (KOTSIANTIS, 2006).

Em aprendizado indutivo, existem dois tipos treinamento com as instâncias: um sistema é considerado de aprendizado supervisionado quando se utiliza de instâncias que já estão classificadas corretamente para fazer o treinamento, ou não-supervisionado quando as instâncias não foram classificadas previamente para fazer o treinamento e o algoritmo é que irá realizar essa separação (JAIN et al., 1999, apud KOTSIANTIS et al., 2006).

As possíveis aplicações da predição podem ser divididas entre classificação, onde se encontra à qual categoria a amostra recebida pertence, ou regressão, onde se utiliza um conjunto de variáveis para prever o valor de uma outra variável utilizando das amostras do conjunto (IMANDOUST e BOLANDRAFTAR, 2013).

3. *K-NEAREST NEIGHBOR*

K-Nearest Neighbor é um método de predição que utiliza da distância entre a amostra atual e seus *k* vizinhos mais próximos no conjunto de treinamento para definir qual será o resultado de sua predição. Ele pode ser utilizado tanto para realização de classificação de instâncias como regressão (KOTSIANTIS, 2006) (KOTSIANTIS, 2007).

O algoritmo de classificação utilizando *kNN* pode ser descrito da seguinte maneira. Dado um parâmetro *k,* e um grupo de amostras de treinamento $M_{x,y}$,nós descobrimos a classe de uma amostra teste *t* comparando a distância de *t* com todos as *x* amostras disponíveis em M.

Diferentes modos de encontrar a distância existem, mas o mais comum é a utilização da distância Euclidiana entre os *y* parâmetros da amostra *t* e os parâmetros de cada outra amostra disponíveis no conjunto de treinamento. Feito isso, são selecionados os *k* vizinhos mais próximos, $0 < k \leq x$, estes farão parte do conjunto que irá definir qual é a vizinhança da amostra. A amostra *t* será associada com a classe mais frequente na vizinhança explorada.

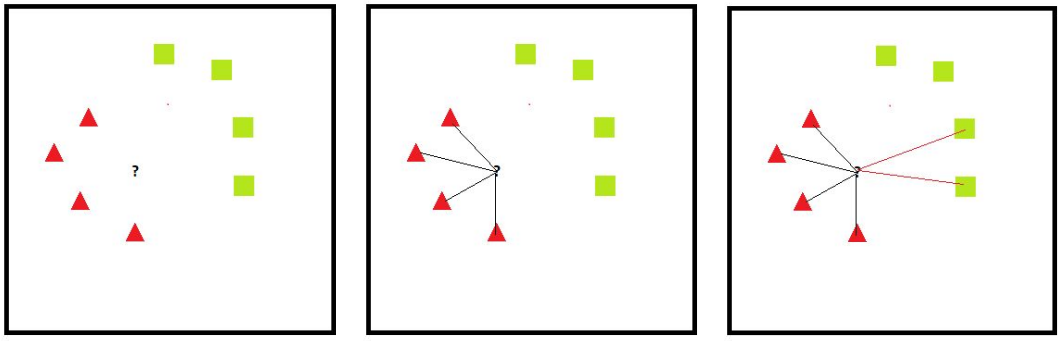

Imagem 3.1 Seleção de vizinhos para uma nova amostra a ser classificada. A segunda imagem mostra a execução se o teste fosse realizado com os 4 vizinhos mais próximos. Para a terceira imagem, utilizamos 6 vizinhos. Note que quando o número aumenta, a classe de quadrados vem a influênciar numa vizinhança que previamente era unicamente de triângulos.

Uma das fraquezas de métodos de aprendizado baseados em instância é devido a seu custo computacional. O resultado da classificação precisa comparar a amostra teste com todas as amostras presentes no conjunto de treinamento, o que em problemas de classificação com bases de dados muito grande leva a classificações muito lentas. A escolha do número de vizinhos também afeta a precisão dos resultados do algoritmo, sendo necessário um maior cuidado na sua escolha (KOTSIANTIS, 2006).

Além disso, o algoritmo é mais viável de ser utilizado em problemas de poucos parâmetros, devido a maldição da dimensionalidade. Quanto mais se aumenta o número de dimensões em um conjunto de dados, mais as métricas de distâncias perdem relevância, levando a distância de uma amostra muito próxima de um ponto não ser tão diferente da distância da amostra mais longe dele (KOUIROUKIDIS et al, 2011).

Para entender melhor essas restrições, iremos fazer uma análise de algumas das propriedades desses algoritmos a seguir. O entendimento delas é essencial para se saber o motivo da existência dos problemas e como as soluções implementadas podem contornar eles.

3.1 NÚMERO DE AMOSTRAS

De acordo com KUSNER et al. (2014), um dos problemas do método *k-NN* é o seu alto custo computacional para realizar testes. O tempo necessário para realizar a classificação de uma amostra requer o cálculo de distâncias de cada amostra e a nova amostra, e dependendo do número de dimensões isso pode se tornar ainda mais demorado. Além disso, o tamanho necessário para armazenar cada amostra para ser testada leva à um problema de escalabilidade de acordo com que uma base de dados cresça.

Isso leva à *k-NN* ser um método pouco indicado para a realização de predições em tempo real em uma base de dados excessivamente grande. Existem múltiplas maneiras de diminuir o tempo e espaço necessário para a computação de predição do algoritmo, alguns desses sendo redução do número de computações

para calcular distâncias, redução de dimensionalidade e a redução do número de amostras (KUSNER et al., 2014)

Vários autores desenvolveram métodos que reduzem o número de instâncias de diferentes maneiras, com o intuito de reduzir o problema de escalabilidade sem perder as informações que o conjunto original oferece. Alguns desses métodos são condensamento de informação (HART, 1986; ANGIULLI, 2005; apud KUSNER et al., 2014), que reduz o número de amostra selecionado subconjuntos e eliminando amostras redundantes, clustering dos centros (CHANG, 1974; KOHONEN, 1990; apud KUSNER et al., 2014) e o aprendizado de protótipos (BERMEJO e CABESTANY, 1999 apud KUSNER et al., 2014).

3.2 NÚMERO DE VIZINHOS

No algoritmo *kNN*, a precisão de um resultado depende da escolha do número de vizinhos. Uma vizinhança pequena pode possuir resultados ruins devido a dados esparsos e ruídos, enquanto uma vizinhança muito grande resulta em resultados muito abrangentes que irão incluir amostras de outras classes (IMANDOUST e BOLANDRAFTAR, 2013).

A escolha do número de vizinhos tem de ser feita com cuidado, e já foi demonstrado que se os pontos não foram uniformemente distribuídos, uma boa escolha do número de vizinhos feita no início do algoritmo se torna difícil (IMANDOUST e BOLANDRAFTAR, 2013). Dessa forma, foram desenvolvidas técnicas as quais irão tentar escolher o melhor número de vizinhos baseado em testes em seus dados de treinamento (GARCÍA-PEDRAJAS et al., 2017).

3.3 CÁLCULO DE DISTÂNCIA

Para se saber quais são os vizinhos mais próximos, precisamos saber primeiro como calcular a distância entre vizinhos. Para isso, são utilizadas as funções de distância que para saber quais são os valores mais próximos. Enquanto a distância Euclidiana é a técnica mais popular, existem outros métodos de se calcular a distância entre duas amostras (IMANDOUST e BOLANDRAFTAR, 2013).

Os resultados usando distância Euclidiana são bons em alguns casos, mas podem se tornar pior do que outros métodos dependendo da situação analisada. Outras técnicas como *Chebyshev,* Euclidiana-Quadrada*, Minkowski* e *Manhattan* também são utilizados para aumentar a precisão do algoritmo em certas bases de dados.

Algumas das diferenças são o que cada cálculo afeta no resultado. Distâncias com a fórmula de *Manhattan* ou *Cityblock* são similares à distâncias Euclidiana, mas o efeito de grandes distâncias é reduzido devido a seu resultado não ser elevado ao quadrado.

$$Dcity = |x2 - x1| + |y2 - y1|$$

Ao contrário da Manhattan, a distância Euclidiana-Quadrada iria aumentar ainda mais o efeito de grandes distâncias sobre o resultado. Minkowski é uma combinação de tanto a distância de Manhattan e Euclidiana.

$$D_{manhattan} = (\sum_{k=1}^{d} \left| x_{ik} - x_{jk} \right|^{\frac{1}{p}})^{P}$$

A adição da variável p permite ajustar a importância das distâncias, de forma que com p = 1, o resultado se torna Manhattan e p=2, torna-se euclidiana.

Enquanto a distância é algo que não vai afetar o grupo de k vizinhos mais próximos obtídos, o efeito dessa mudança de fórmula ocorre na associação de peso baseados na distância para o sistema de votação dos k vizinhos.

O desenvolvimento de novas funções de distâncias se torna importante devido à certas propriedades de dimensionalidade que vamos discutir na parte de dimensionalidade. Autores já desenvolveram diferentes maneiras para se calcular distâncias que sejam resistentes aos efeitos de alta dimensionalidade (AGGARWAL, 2001, apud KOUIROUKIDIS e EVANGELIDIS, 2011) (HSU, CHEN, 2009, apud KOUIROUKIDIS e EVANGELIDIS, 2011).

3.4 PESO

No modelo tradicional, cada vizinho possui a mesma relevância para delimitar qual a classe majoritária na vizinhança, independentemente de sua distância ao ponto de origem. Isso pode levar ao mesmo problema discutido quando falamos da escolha do tamanho da vizinhança, onde resultados esparsos ou ruído afetem negativamente na classificação da amostra (IMANDOUST e BOLANDRAFTAR, 2013).

A escolha de fórmulas que deem maior importância para vizinhos específicos pode melhorar ou piorar a eficiência de um algoritmo dependendo de qual o objetivo da predição. Como já exploramos na Seção 3.3, podemos utilizar grupos diferentes de cálculos de distâncias para associar pesos. Além de distâncias, outros tipos de cálculos para pesos podem ser utilizados, como a utilização de *hubness* (TOMAŠEV e BUZA, 2015).

3.5 DIMENSIONALIDADE

O avanço da tecnologia não só causou um aumento na quantidade de instâncias em bases de dados, mas também um aumento no número de atributos e características nessas bases de dados (VERLEYSEN e FRANÇOIS, 2005). Em problemas de alta dimensionalidade, as diferenças de distâncias de amostras se tornam muito pequena devido ao número excessivo de dimensões (KOUIROUKIDIS e EVANGELIDIS, 2011). Para solucionar esse problema, alguns autores desenvolveram funções específicas de distâncias para estes algoritmo (AGGARWAL, 2001, apud KOUIROUKIDIS e EVANGELIDIS, 2011) (HSU, CHEN, 2009, apud KOUIROUKIDIS e EVANGELIDIS, 2011). Outra possível solução envolve a ideia de hubness. (TOMAŠEV et al., 2014)

Hubness é o conceito que introduz a ideia de *hubs*, amostras as quais possuem um grande número de amostras que escolheriam ela como um vizinho mais próximo. TOMAŠEV et al. (2014) indica que o aumento de dimensões pode levar ao aparecimento de *hubs* em uma base de dados, e que esses podem ser utilizados para se aumentar a precisão de um algoritmo.

4. ALGORITMOS UTILIZADOS

4.1 *MULTI-LOCAL MEANS-BASED K-HARMONIC NEAREST NEIGHBOR*

O algoritmo que PAN et al. (2017) propuseram tem como foco combinar *local mean-based k-nearest neighbor* com a utilização de médias harmônicas para classificar os resultados. Ela tem como intuito reduzir o problema do LMKNN que utiliza um mesmo número de *k* vizinhos para cada classe o que causa a sua efetividade ser influenciada pesadamente pela escolha do *k.*

O algoritmo não utiliza os *k* vizinhos mais próximos da amostra, mas sim um grupo de *k* vizinhos da mesma classe mais próximos da amostra para cada classe disponível no conjunto. Encontrados esses subconjuntos, são encontrados os *r* primeiros vetores médios locais desses vizinhos onde 1 ≤ r ≤ *k*.

$$m_r = \frac{1}{r} \sum_{i=1}^{r} \overline{X}_{i,c}$$

no qual $\overline{X}_{n,c} \in T_c$

T_c = Conjunto de *k* amostras mais próximas da instância pertencentes à classe *c*

Então são calculadas as distâncias da amostra a ser classificada e todos os m_r vetores médios de cada classe. Essas distâncias são utilizadas então numa média harmônica com todos os *d* resultados. A classe que possuir a menor média harmônica vai ser associada à amostra, sendo considerada a classe dominante da vizinhança.

$$H_d = \frac{r}{\sum_{i=1}^{r} \frac{1}{d(mr, x)}}$$

O interesse da utilização da média harmônica se dá devido à sua valorização de valores menores, uma vez que valores pequenos podem reduzir o valor da média

harmônica mais severamente do que valores altos podem aumentar. Assim, esse algoritmo pega uma representação de todas as vizinhanças de cada classe e prioriza aqueles que possuem amostras mais próximas.

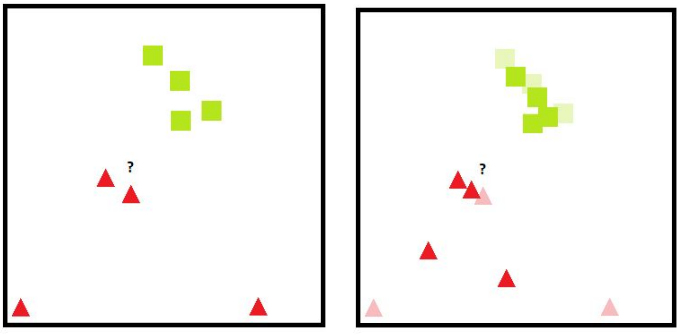

Imagem 4.1.1 Exemplo de uma execução deste algoritmo, assumindo um número de vizinhos 4 e número de vetores médios também 4. Neste exemplo estamos simplificando os valores utilizados para demonstrar como os valores afetam os resultados do algoritmo.

Os vetores médios gerados, como na imagem 2, mostram como a utilização de vetores médios entre as *r* primeiras amostras da classe mais próximas prioriza a classe que tiver os vizinhos mais próximos e condensa suas amostras. Mas ainda assim isso não resolveria o problema de escolha de vizinhos próximos.

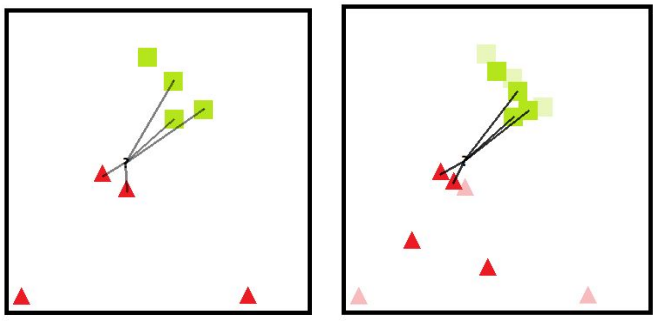

Imagem 4.1.2 Utilizando *kNN* tradicional depois das médias serem encontradas. Mesmo com as novas amostras, o algoritmo ainda vai sempre classificar como a classe quadrados verdes se *k* >= 4. No caso isso pode causar classificações incorretas dependendo da base de dados.

Para solucionar situações como na imagem acima, onde mesmo os vizinhos mais próximos sendo de uma classe, a amostra não será classificada a eles a não ser se os valores forem muito pequenos, o algoritmo utiliza a média harmônica.

$$D_1 = (300,\ 400,\ 500,\ 600),\ D_2 = (50, 50, 850, 850),\ \sum_{i=0}^{4} D_{1,i} = \sum_{i=0}^{4} D_{2,i} = 1800$$

$$\mu_1 = \mu_2 = (1800/4) = 450$$

$$H_1 = 421\ ,\ H_2 = 94$$

O exemplo acima expõe uma situação similar à da imagem 4.1.1 e 4.1.2. Mesmo possuindo dois valores excessivamente altos de distância, a classe que possui amostras mais próximas teve a menor média harmônica das distâncias.

Esse efeito leva ao que foi dito anteriormente onde o algoritmo irá valorizar as amostras mais próximas em favor das mais longe, mas mesmo assim não irá sofrer com amostras aberrantes e ruído, uma vez que mesmo com essas ele ainda irá tomar em conta todos os vizinhos próximos de cada classe e a média deles.

4.2 LOCAL-BEST KNN

Um dos métodos de se descobrir qual a melhor opção para o número de vizinhos é uma validação cruzada com o base de dados de treinamento, buscando o *k* que possui a melhor performance. Mas a escolha do tamanho da vizinhança feita no início da execução pode levar à problemas, como discutido no capítulo anterior. Assim foram desenvolvidos métodos para encontrar valores bons de k que variam, um desses é o *Local-best kNN* (GARCÍA-PEDRAJAS et al., 2017).

Diferente do *kNN* normal, são definidos dois valores inicialmente: o número mínimo de vizinhos e o número máximo de vizinhos. Em seguida são realizados testes do tipo validação cruzada utilizando 10-folds no banco de dados de treinamento com cada valor entre o número máximo e mínimo, onde se encontra o *k*

que possui em média os melhores resultados. Esses são chamado de ótimos globais de cada *k* entre o mínimo e máximo.

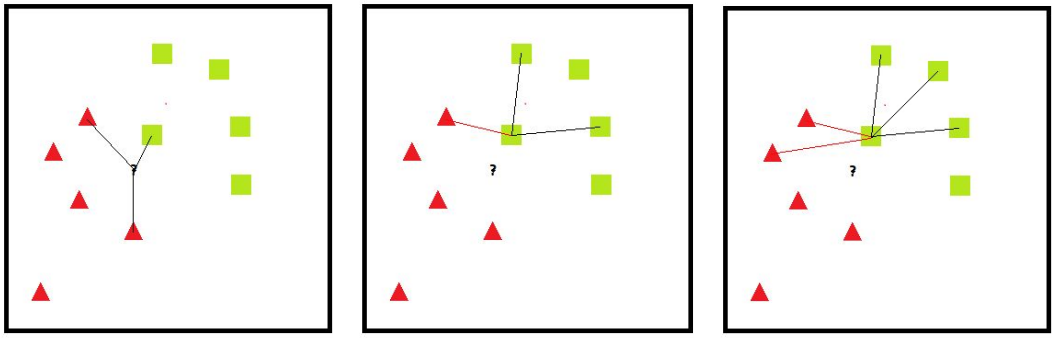

Imagem 4.2.1 Funcionamento do algoritmo de escolha de melhor vizinho do best local. Na imagem 1 aparece a seleção de vizinhos. Imagem 2 mostra o vizinho mais próximo do algoritmo fazendo sua escolha. No caso, comparando ele com *k* minimo = 3 e *k* máximo = 5, ele escolheria *k* = *3* por ter 66% das amostras de mesma classes ao invés do 60% das amostras de mesma classe do *k* = *5;*

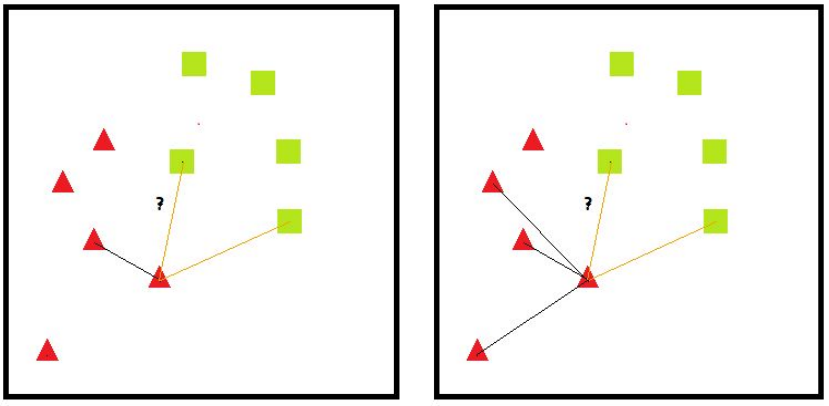

Imagem 4.2.2 No caso do segundo vizinho, a vizinhança de mesma classe do *k* = *3* seria só 33%. Por outro lado, a vizinhança com 5 vizinhos teria 60% de amostras de mesma classe, ele escolheria 5. Assumindo que a terceira amostra seria com 3 também, o nosso resultado seria dois vizinhos escolhendo 3 e um escolhendo 5. O valor associado para a amostra seria (5+3+3)/3 = 3.6, arredondando iria ter como resultado *k* = 4.

Pseudo Algoritmo 1 *Local-best kNN*: Obtenção de *k*

Input:

T = Conjunto formado de N amostras de treinamento x_n.
k_{min} , k_{max} = Número de vizinhos mínimo e máximo a ser explorado.

Output:

melhorK = Melhor k para cada amostra

---

listaDePrecisoesGlobais = calcularPrecisoesGlobais($k_{min,}$ k_{max})
listaDePrecisoesLocais[N][k_{max} - k_{min}]
melhorK[N]
Para cada amostra x_n, $x_n \in T$:
    listaDePrecisoesLocais[n] = calcularMelhorPrecisao($x_{n,}$ k_{min}, k_{max})
    somatorioPrecisao[k_{max} - k_{min}]
    Para cada *k* entre k_{min},k_{max}:
        somatorioPrecisao[k] = listaDePrecisoesGlobais[k] + listaDePrecisoesLocais[n][k]
    kEscolhido = maiorPrecisao(somatorioPrecisao)
    melhorK[n] = kEscolhido

O algoritmo têm como interesse não ficar completamente dependente do resultado do *k-local*, por isso que se utiliza também os ótimos globais para achar o *k* que deve possuir a melhor precisão. Para cada *k* entre o mínimo e máximo, somamos a precisão dos resultados do global e do vizinho para gerarmos a precisão final geral de um *k* (GARCÍA-PEDRAJAS et al., 2017). O *k* com precisão máxima será associado à aquela amostra de treinamento.

A segunda parte deste algoritmo ocorre na classificação. Dada uma amostra para se classificar, se escolhe o vizinho mais próximo à ela. Esse vizinho será o vizinho base para encontrarmos o melhor número para *k* naquela vizinhança. Novamente testamos todos os possíveis *n* valores entre *k* mínimo e máximo, e associamos à esse vizinho base o valor que obtiver o maior número de amostras na vizinhança que são da mesma classe do vizinho base. Enquanto o artigo indicava que usando um único vizinho base tinha uma boa precisão, eles escolheram utilizar 3 vizinhos base para encontrar o *k* da vizinhança, sendo eles os 3 mais próximos da amostra que está sendo classificada. Esse *k* formado pelos vizinhos será o melhor *k-local*, em contraste ao *k-global*.

Pseudo Algoritmo 2 *Local-best kNN*: Escolha de *k* para a classificar a instância.

Input:

listaDeVizinhos[N] = lista de distâncias entre a instância e todo o conjunto de treinamento

melhorK[N] = lista com o melhor *k* para a amostra

Output:

kClassificacao = número de vizinhos para serem utilizados na classificação.

---

valorK = 0
Para os 3 vizinhos mais próximos de *listaDeVizinhos:*
    valorK += melhorK[vizinho]
valorK = int(valorK/3)
kClassificacao = valorK

4.3 FUZZY HUBNESS

O algoritmo se utiliza do conceito de *hubness* de amostras. *Hubness* é uma característica que aparece devido à maldição da dimensionalidade onde, devido ao alto número de dimensões que o domínio está contido, ocorre uma distorção nas ocorrências de amostras numa k-vizinhança (DHILLION et al., 2004, apud TOMAŠEV et al., 2013).

Isso leva à criação de *hubs*, amostras que estão contidas em mais k-vizinhanças que outras (TOMAŠEV et al., 2013). A *hubness* indica a relevância de uma dada amostra. Devido ao problema de distorção, grande parte dos pontos se tornam *anti-hubs* ou órfãos, que indica que eles raramente ou nunca são partes da vizinhança de uma amostra de treinamento (TOMAŠEV et al., 2015).

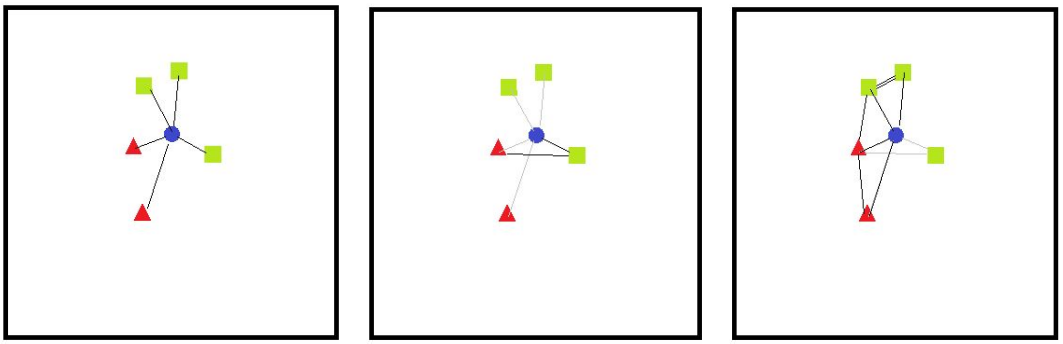

Imagem 4.4.1 Uma representação gráfica do conceito de *hubness*, utilizando *hubness* com k = 2 vizinhos. Na primeira imagem vemos que todas as amostras apontam para a amostra círculo azul. Nesse caso, indicamos que o *hubness* do círculo azul é 5, pois 5 amostras têm ele entre os 2 vizinhos mais próximos. A segunda e terceira imagens mostram o conceito de amostras órfãs: enquanto o quadrado verde isolado vota em dois vizinhos próximos, nenhuma das outras amostras votam no *hubness* dele, logo ele possui *hubness* 0.

A qualidade de um *hub* é dada pelo seu bom *hubness* ou mal *hubness*. Se certa instância é parte de múltiplas vizinhanças de instâncias que sejam da mesma classe que ela é, ela possui uma boa *hubness* alta. Se por outro caso, ela faz parte da vizinhança de muitas amostras que sejam de classes diferentes, ela possui um mal *hubness* elevado (TOMAŠEV et al., 2013). *Hubness* ruim não implica que seja uma instância errônea, mas que sua importância é considerada ruído (TOMAŠEV et al., 2015).

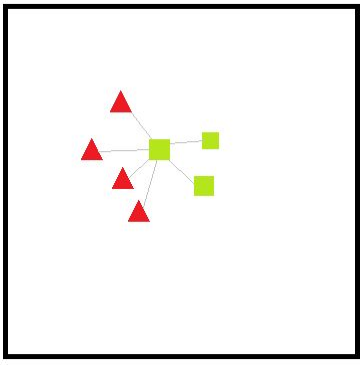

Imagem 4.4.2 Exemplo de uma amostra com mal *hubness* elevado. Neste exemplo, a amostra central possui *hubness* de 6, e mal *hubness* de 4, por

possuir 4 elementos de outra classe que considera ele um vizinho próximo. Seu bom *hubness* é de 2, por possuir um par de vizinhos que é da mesma classe dele.

Algoritmos de *k-NN* que utilizam de hubness envolvem a ideia de votos com peso, onde amostras que possuem mal *hubness* elevados ou *hubness* total baixo possuem votos com menor peso que outras amostras, de forma a reduzir sua influência no resultado.

No algoritmo de TOMAŠEV et al. (2012), ele utiliza de uma versão modificada de *hubness.* Ao invés de usar de bom ou mal *hubness*, se utiliza a ideia de *hubness* de classe. Uma amostra possui uma *hubness* para cada classe do domínio. Isso leva aos valores não indicarem qual é sua participação em cada classe, mas sim a sua aparência em relação à amostras contidas no conjunto de treinamento.

Pseudo Algoritmo 3 Encontrar hubness

Input:
T = lista de amostras
vizinhos[T] = lista de vizinhos que cada amostra possui
C = lista de classes
Output:
hubnessClassList[T][n_c] = lista de hubness das amostras para cada classe
hubnessSum[T] = lista de hubness

```
Para cada amostra x em T:
    Para cada vizinho k em vizinhos[x]:
        hubnessSum[T] += 1
        Para cada classe em C:
            hubnessClassList[T][classe] += 1
```

Nesse algoritmo, se obtém os *k* vizinhos mais próximos para indicar qual é a classe da amostra a ser classificada. Cada vizinho irá utilizar seu *hubness* para votar nas possíveis categoria que a amostra a ser classificada pertence de acordo com a seguinte função.

$$w(n,c) = \frac{(\gamma + h_{n,c})}{(\gamma * N_c + h_n)}$$

Função de peso do voto baseado em *hubness,* onde γ é um estimador de suavização de Laplace, N_c é o número de classes, $h_{n,c}$ é a *hubness* da classe da amostra e h_n é o *hubness* total da amostra.

O estimador γ utilizado foi o valor 0.001 como na implementação original por TOMAŠEV[3].

Devido à propriedade de distorção causada por um grande número de dimensões (TOMAŠEV e BUZA, 2015), é possível que instâncias votem sem peso algum devido a ela ser considerada uma amostra órfã. Para isso TOMAŠEV et al. (2012) utiliza uma função alternativa para essas amostras.

Caso uma amostra possua *hubness* abaixo de certo limiar *θ* especificado pelo usuário, se utiliza uma diferente função. Os quatro tipos que o artigo especifica são *crisp*, *global* e duas diferentes funções locais.

A função local *crisp* envolve simplesmente votar como se o método fosse o *k-NN* tradicional, isso é, votar com peso 1 somente na sua própria classe com certo fator de suavização. Ainda realiza um voto para as outras classes com somente o valor de laplace.

$$w(n,c)_{crisp} = \frac{(\gamma + 1)}{(\gamma * N_c + h_n)} \text{ , se } c = y_i.$$

$$w(n,c)_{crisp} = \frac{(\gamma)}{(\gamma * N_c + h_n)} \text{ , se } c \neq y_i.$$

A função global envolve a utilização de *fuzziness* com o somatório do *hubness* das classes disponíveis no algoritmo dividido pelo *hubness* total delas, somente da classe específica. Isso é um método mais *fuzzy* do que o método *crisp*, uma vez que isso dá uma ideia da probabilidade dada amostra pertencer a tal

[3] Hubness-aware machine learning library. https://github.com/datapoet/hubminer

classe, mas devido ao fato de utilizar todas as amostras, não se consegue dizer nada sobre a vizinhança da amostra.

$$w(n,c)_{global} = \frac{\left(\gamma + \sum h_{n,c}\right)}{(\gamma * N_c + \sum h_n)} \text{ , se c = } y_i.$$

Uma das funções locais, que vamos chamar de *L1,* envolve encontrar a densidade local de classes da amostra. Utilizando os vizinhos ao redor, se utiliza da seguinte função:

$$w(n,c)_{local1} = \frac{\left(\gamma + \sum_{j=0}^{k} \delta_{cy_{ij}}\right)}{(\gamma * N_c + k + 1)}$$

Função de peso *local1,* onde $\delta_{cy_{ij}}$ é a função delta Kronecker, onde o resultado é 1 caso o vizinho pertença à mesma categoria da amostra que está votando, e 0 caso contrário.

Essa função representa a *fuzziness* da amostras, com um valor de suavização de Laplace. A amostra que está votando também é contada nessa função local para dar alguma influência dela no resultado. O *k* utilizado nessa função não necessariamente precisa ser o mesmo do utilizado na função *hubness,* uma vez que isso depende do domínio examinado.

A segunda função local, que vamos chamar de *L2* é uma modificação da função *L1*, ela envolve modificar o valor da fuzziness para garantir um peso maior se a classe da amostra. A nova função funciona da seguinte maneira:

$$w(n,c)_{local2} = 0.51 + 0.49 * \frac{\left(\gamma + \sum_{j=0}^{k} \delta_{cy_{ij}}\right)}{(\gamma * N_c + k + 1)} \text{ , se c = } y_{i,}$$

$$w(n,c)_{local2} = 0.49 * \frac{\left(\gamma + \sum_{j=0}^{k} \delta_{cy_{ij}}\right)}{(\gamma * N_c + k + 1)}$$, se $c \neq y_i$.

Essa função é uma combinação da função *crisp* e a *L1*, dando uma prioridade maior à classe da amostra original enquanto ainda tendo um fator de *fuzziness* local.

As variáveis desse algoritmo são definidas numa fase de treinamento. Utilizando o conjunto de treinamento, se tenta classificar as amostras utilizando um limiar θ diferente cada vez e verificando os resultados de cada possível função alternativa. A combinação de limiar θ e função alternativa que tiver o melhor resultado é utilizado para fazer predições.

Para a nossa implementação, nós escolhemos variar um limiar θ entre 0 e 9, e o número de vizinhos utilizados para a função *L1* e *L2* são o mesmo do *k* da função.

Pseudo Algoritmo 4 Classificação com hubness

Input:

vizinhosAmostra[k] = lista de vizinhos proximos
hubnessTotal[T] = lista de hubness total do sistema
C = lista de classes
θ = limiar de hubness
altFunc = funcao alternativa

Output:

classeEscolhida = classe que o algoritmo encountrou

---

votos[C] = 0
Para cada n em *vizinhosAmostra*:
    hubness = hubnesssTotal[n]
    Se *hubness > θ:*
        Para cada *classe* em *C:*
            votos[classe] += $w(n, c)$
    Caso contrario:
        Para cada classe em C:
            Se altFunc = *"crispy":*
                votos[classe] += $w(n, c)_{crispy}$
            Se altFunc = *"global":*
                votos[classe] += $w(n, c)_{global}$
            Se altFunc = *"local1":*
                votos[classe] += $w(n, c)_{local1}$
            Se altFunc = *"local2":*
                votos[classe] += $w(n, c)_{local2}$
classeEscolhida = classe de votos com maior valor

4.4 *IMPROVED K-NEAREST NEIGHBOUR CLUSTERING*

Em busca de métodos apropriados para se implementar, um artigo que propunha uma implementação diferente do método *kNN* foi parcialmente implementado. No artigo original de ZHENG et al. (2017), se utiliza uma versão diferente de *kNN* que busca utilizar o comportamento específico do domínio para melhorar os resultados gerados.

Enquanto não conseguimos modificar esse algoritmo para ele se tornar um método geral para qualquer domínio, a implementação do método de clustering

descrito pelo algoritmo ainda era possível de ser utilizado. Devido à resultados interessantes quando utilizado com os outros algoritmos, decidimos manter parcialmente a implementação do artigo para testes.

Clustering é a classificação não-supervisionada de amostras em grupos chamados *clusters* (JAIN e MURTY, 1999). O método de clustering descrito por ZHENG et al. (2017) é formado por duas etapas. Inicialmente se adquire um conjunto de centros de clusters C com n_c centros, utilizando do algoritmo de *Fuzzy C-Means*. Nossa implementação obteve esses centros utilizando a função nativa do *scikit-fuzzy*[4] do *Python*.

Com esse conjunto de centros C, nós utilizamos cada centro $c \in C$ para encontrar um subconjunto de amostras descritivas do conjunto original. Se encontram os n_{k1} vizinhos mais próximos e se adiciona eles num novo conjunto. No artigo não especifica se esses vizinhos podem ou não ser repetidos, mas para garantir que não tenha mais ou menos que o número especificado, nós não consideramos as amostras que já estejam no novo subconjunto.

Assim, se reduz o domínio original de amostras para um conjunto formado por n_c x n_{k1} amostras que buscam representar as informações contidas no conjunto original (ZHENG et al., 2017) Na implementação do artigo, se utilizava de 100 centros e que cada centro se pegava as 3 amostras mais próximas para isso. Como alguns dos bancos de dados não possuem amostras o bastante para tornar isso uma redução, nós reduzimos o número de centros para 15 e mantivemos o número de amostras obtidas 3, totalizando 45 amostras. Não realizamos nenhuma técnica para garantir a proporção de amostras das classes similares ao banco de dados original, balancear as amostras utilizadas ou remover amostras que fossem redundantes.

Pseudo Algoritmo 3 Seleção de Amostras

Input:
T = lista de amostras
n_c = número de centros para ser utilizados
n_k = número de vizinhos a serem utilizados
Output:

[4] https://pypi.python.org/pypi/scikit-fuzzy

t = subconjunto das amostras originais

---

listaCentros = *fuzzyCmeans*(*T*, n_c)
novoSubconjunto[n_k,n_c]
Para cada centro em listaCentros:
    novoSubconjunto.add(acharVizinhos(centro,n_k*))*
t = *novoSubconjunto*

---

5. ANÁLISE EXPERIMENTAL

5.1 METODOLOGIA

Para compararmos os métodos implementados, obtemos um grupo de bancos de dados do repositório da UCI[5] (LICHMAN, 2013). Todas as bases de dados utilizadas foram de classificação e que possuíam atributos completos e numéricos. Essas bases foram escolhidas tantos por terem sido utilizadas pelos algoritmos implementados em seus artigos e por cada uma ter uma propriedade importante a ser verificada. *Iris* é uma base de dados básica com menor tamanho, com quantidade de classes e amostras pequenas, *Wine* possui um número de amostras próximo ao *Iris*, mas seu número de dimensões é muito maior, permitindo analisar a resistência a altas dimensionalidades de alguns algoritmos. *Glass* é uma base utilizada por outras análises dos algoritmos e serve como um *Iris* maior, mas uma característica muito importante dele é a grande quantidade de categorias e o fato delas serem desbalanceadas, o que não ocorre nas bases previamente analisadas. A *Ionosphere* é uma versão extrema de multidimensionalidade, com um tamanho relativamente pequeno e uma classificação binária, algoritmos com melhor desempenho com alta dimensionalidade supostamente devem possuir um resultado melhor nesse domínio. *Yeast* é uma base de dados onde o tamanho é muito maior, mas possui dimensionalidade pequena.

Tabela 5.1.1 Sumário dos Bancos de Dados utilizados

	Amostras	Classes	Atributos	Distribuição
Iris	150	3	4	Balanceada
Wine	178	3	13	Balanceada
Glass	214	6	9	Desbalanceada
Ionosphere	351	2	34	Desbalanceada
Yeast	1484	10	8	Desbalanceada

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

Os testes foram realizados utilizando de validação cruzada com 30-*folds*, todos os testes utilizando o mesmo grupo de *folds*. Para cada teste realizado, cada algoritmo otimizou o valor do seu *k* e no caso do *hubness*, o θ e função utilizada. O tempo necessário para realizar essa otimização é o chamado “tempo de treinamento”, que métodos de aprendizado por instâncias normalmente não possuem. Achamos interessante ter essa métrica, mesmo que seja algo que seja executado somente uma vez. A otimização de *k de* todos os algoritmos foi utilizando os valores de *k*-mínimo 1 e *k*-máximo 20, de forma a manter o tempo de treinamento baixo, mas ainda expor os efeitos de tamanhos de vizinhanças diferentes.

As métricas de avaliação que decidimos aplicar são a precisão que cada algoritmo possui, o tempo médio necessário para realizar uma classificação, tempo necessário para “treinar” o algoritmo, e o melhor *k* encontrado para cada método. Mesmo que a precisão seja uma métrica muito importante, escolhemos adicionar o tempo de classificação devido a existência de algoritmos mais computacionalmente exigentes de forma a demonstrar os efeitos em bancos de dados muito grandes e os benefícios de técnicas de redução de amostras para esses modelos.

Para os testes do tipo *clustered*, nós utilizamos um banco de dados reduzido. Depois de dividirmos o banco em 30 possíveis *folds* e criarmos 30 diferentes bancos de teste e treinamento (29/30 para treinamento, 1/30 teste), realizamos o método de redução de amostras especificado na seção anterior em cada conjunto de treinamento. Desta forma, criamos 30 possíveis bancos reduzidos.

Utilizamos cada um desses bancos reduzidos para treinar os algoritmos, e rodamos os algoritmos que foram treinados com o banco reduzidos sobre os mesmos bancos dos que foram classificados pelos que possuem o banco de dados completos. Isso têm interesse em tanto testar a qualidade do banco reduzido gerado pelo algoritmo de redução implementado e a capacidade de cada algoritmo de utilizar um banco de dados menor para realizar suas classificações.

Foram realizados testes One-way ANOVA utilizando o python scipy.stats.f_oneway, a implementação do ANOVA do python. Foi realizado um teste usando as 30 precisões, comparando a de cada um com o *kNN* tradicional para saber se há uma diferença estatística entre os testes.

5.2 RESULTADOS

Os resultados obtidos expõe como as modificações no *kNN* influenciam na performance de cada algoritmo. Nas tabelas abaixo podem ser verificadas a precisão do algoritmo, o tempo de treinamento e teste, além dos valores para *k*.

Tabela 5.2.1 Resultados na base de dados *Iris*

Algoritmo	Precisão	Treinamento	Teste	k	θ	*p-value*
k-NN	96.67 ± 9.07	134.398 ± 13.814 ms	0.368 ± 0.034 ms	15.0 ± 4.0173		
Local-Best k	**98.0 ± 6.0**	307.588 ± 35.167 ms	5.949 ± 0.593 ms	10.0 ± 3.0114		0.5116
MLM-KHNN	95.33 ± 9.91	612.487 ± 59.934 ms	13.345 ± 3.556 ms	6.0 ± 6.1432		0.5950
Fuzzy-Hubness	96.67 ± 7.45	140715.737 ± 13734.696 ms	6.043 ± 0.675 ms	4.0 ± 5.6079	1.0 ± 3.3541	1.0

Tabela 5.2.2 Resultados na base de dados *Wine*

Algoritmo	Precisão	Treinamento	Teste	k	θ	*p-value*
k-NN	75.78 ± 15.47	138.762 ± 3.922 ms	0.349 ± 0.022 ms	1.0 ± 1.6155		
Local-Best k	72.89 ± 15.22	378.088 ± 11.616 ms	8.401 ± 0.534 ms	3.0 ± 3.1574		0.4762
MLM-KHNN	**77.44 ± 17.01**	807.395 ± 13.272 ms	21.668 ± 3.298 ms	12.0 ± 5.2329		0.6976
Fuzzy-Hubness	67.33 ± 18.33	196271.121 ± 2296.805 ms	8.321 ± 0.456 ms	2.0 ± 0.359	0.0 ± 0.0	**0.06290**

Tabela 5.2.3 Resultados na base de dados *Glass*

Algoritmo	Precisão	Treinamento	Teste	k	θ	*p-value*
k-NN	72.08 ± 13.61	150.903 ± 9.97 ms	0.363 ± 0.037 ms	1.0 ± 0.1795		
Local-Best k	66.79 ± 16.13	484.589 ± 18.787 ms	11.53 ± 1.338 ms	4.0 ± 3.0375		0.1817
MLM-KHNN	**72.44 ± 16.49**	1246.561 ± 59.707 ms	24.268 ± 2.9 ms	3.0 ± 2.3907		0.9286
Fuzzy-Hubness	68.63 ± 15.55	274858.329 ± 7154.184 ms	11.746 ± 0.776 ms	1.0 ± 0.4899	0.0 ± 0.0	0.3720

Tabela 5.2.4 Resultados na base de dados *Ionosphere*

Algoritmo	Precisão	Treinamento	Teste	k	θ	*p-value*
k-NN	89.67 ± 8.09	251.944 ± 7.569 ms	0.481 ± 0.046 ms	2.0 ± 0.0		
Local-Best k	87.75 ± 9.56	1302.981 ± 22.05 ms	35.694 ± 2.201 ms	4.0 ± 3.4447		0.4127
MLM-KHNN	89.47 ± 7.1	2356.888 ± 33.813 ms	79.222 ± 5.263 ms	16.0 ± 2.3466		0.9198
Fuzzy-Hubness	**90.28 ± 8.11**	842416.115 ± 3553.608 ms	35.802 ± 1.573 ms	3.0 ± 1.0672	0.0 ± 0.0	0.7767

Tabela 5.2.5 Resultados na base de dados *Yeast*

Algoritmo	Precisão	Treinamento	Teste	k	θ	*p-value*
k-NN	**58.9 ± 8.51**	1228.615 ± 73.472 ms	2.485 ± 0.694 ms	16.0 ± 3.3779		
Local-Best k	57.28 ± 6.44	18106.492 ± 510.902 ms	553.592 ± 53.137 ms	9.0 ± 3.7978		0.417738897 19571133
MLM-KHNN	57.41 ± 7.44	35445.048 ± 962.865 ms	1231.485 ± 49.791 ms	18.0 ± 1.4376		0.480844188 61975853
Fuzzy-Hubness	53.64 ± 6.42	13122858.105 ± 284254.58 ms	568.024 ± 24.584 ms	2.0 ± 0.0	0.0 ± 0.00	**0.010086862 11547272**

Na Tabela 5.2.1, enquanto houve uma diferença das médias dos valores de cada um dos classificadores, não houve diferenças estatísticas. Mesmo assim, o valor de *k* que o Fuzzy-Hubness utilizou foi menor para chegar a um resultado estatisticamente similar, utilizando somente de uma média de 4 vizinhos comparados aos k-NN, o MLM-KHNN utilizou também um valor comparativamente menor que o original.

Na Tabela 5.2.2, o Fuzzy-Hubness possui uma redução de precisão que é estatisticamente relevante, o que demonstra que ele não têm uma boa performance nesse tipo de banco de dados. Não houve uma diferença estatística entre o *kNN* tradicional e os modelos modificados, mesmo quando o *MLM-KHNN* teve uma precisão média maior. Podemos notar que nesse exemplo o *k* necessário pelos outros era baixo, enquanto o *k* utilizado pelo *MLM-KHNN* foi maior.

Na Tabela 5.2.3 têm como seu mais preciso algoritmo o *MLM-KHNN*, enquanto outros têm resultados piores, mas novamente não é estatisticamente relevante. Mesmo não possuindo uma diferença estatística, os outros algoritmos incluindo o MLM-KHNN possuíam um tempo necessário para testes maior que o *kNN*, não sendo um ganho relevante de precisão em relação a perda na velocidade de classificação.

De acordo com que as dimensões aumentam, *Fuzzy-Hubness* se torna mais preciso. Na tabela 5.2.4, a base de dados *ionosphere* possui 34 atributos e o *Fuzzy-Hubness* possui a melhor precisão, mas esse ganho não é estatisticamente relevante. O fato de possuir um θ de 0 mostra que é mais preferível sempre utilizar o cálculo utilizando hubness da amostra ao invés de utilizar os algoritmos alternativos que o algoritmo possui. Como nos exemplos anteriores, o ganho não foi relevante o bastante e o tempo necessário para a sua classificação foi muito alto.

Na base de dados *Yeast*, todas as modificações tiveram precisão média reduzida. O tempo necessário para realizar os algoritmos alternativos também foi muito maior que o algoritmo tradicional, o que apontou problemas na escalabilidade do algoritmo *MLM-KHNN,* e que esse tempo maior não resultou em um ganho na média de precisões, mesmo se não houve diferença estatística. No *Fuzzy-hubness,* não só houve um aumento no tempo necessário para realizar a classificação, mas também houve uma redução estatisticamente significativa na sua precisão. O alto número de classe pode ser a causa desse problema, uma vez que o *yeast* além de grande é formado por um número maior de classe do que as outras bases.

Um dado observado em todos os resultados foi o fato do algoritmo *Best-local k* ser na maioria dos casos igual ou pior que o algoritmo tradicional mesmo quando não há uma diferença estatística, isso pode estar associado ao ao fato que as vizinhanças de duas amostras podem ser muito diferentes. O número mais preciso de vizinhos para uma amostra pode não ser para amostras vizinhas.

Existe a possibilidade de um mal-entendido na implementação no algoritmo, devido a uma ambiguidade de como o autor indica como vai ser utilizado os 3 vizinhos mais próximos para se encontrar o melhor número de vizinhos para a amostra testada. Pode ter sido utilizando o menor, maior dos três ou a média, na nossa implementação utilizou uma média dos 3 *k* para achar o valor do *k*.

5.3 RESULTADOS DA VERSÃO *CLUSTERED*

Neste conjunto de testes, nós estamos avaliando os efeitos do algoritmo de redução de amostras implementado por ZHENG et al. (2017). O objetivo desses testes é verificar como a utilização dessa redução afeta os resultados, principalmente a precisão e velocidades. Como já especificamos no capítulo 4, utilizamos o 15 centros e 3 amostras para cada um, totalizando 45 amostras obtidas. O algoritmo é rodado na base de dados de treinamento dos algoritmos e os testes são realizados na mesma base de dados de teste dos testes originais.

Tabela 5.3.1 Resultados na base de dados *Iris Clustered*

Algoritmo	Precisão	Treinamento	Teste	k	θ	*p-value*
k-NN	96.0 ± 8.0	106.722 ± 15.104 ms	0.355 ± 0.059 ms	7.0 ± 3.0385		
Local-Best k	94.67 ± 11.47	121.297 ± 12.698 ms	2.006 ± 0.495 ms	9.0 ± 2.3293		0.6095
MLM-KHNN	**96.67 ± 9.07**	91.419 ± 10.181 ms	3.758 ± 0.611 ms	1.0 ± 0.0		0.7676
Fuzzy-Hubness	**93.33 ± 11.93**	13979.663 ± 1454.293 ms	2.005 ± 0.405 ms	9.0 ± 2.6418	9.0 ± 0.0	0.3214

Tabela 5.3.2 Resultados na base de dados *Wine Clustered*

Algoritmo	Precisão	Treinamento	Teste	k	θ	*p-value*
k-NN	66.11 ± 19.17	100.564 ± 3.001 ms	0.329 ± 0.027 ms	5.0 ± 6.8748		
Local-Best k	**67.89 ± 16.11**	115.822 ± 3.714 ms	2.241 ± 0.253 ms	3.0 ± 3.1547		0.7036
MLM-KHNN	**67.89 ± 18.77**	92.388 ± 3.271 ms	5.104 ± 1.146 ms	3.0 ± 2.8831		0.7224
Fuzzy-Hubness	61.89 ± 21.32	13763.213 ± 176.443 ms	2.227 ± 0.113 ms	2.0 ± 0.6574	1.0 ± 1.436	0.4309

Tabela 5.3.3 Resultados na base de dados *Glass Clustered*

Algoritmo	Precisão	Treinamento	Teste	k	θ	*p-value*
k-NN	**63.51 ± 17.01**	100.765 ± 3.039 ms	0.325 ± 0.023 ms	1.0 ± 0.5735		
Local-Best k	60.18 ± 17.15	115.688 ± 4.672 ms	2.524 ± 0.304 ms	3.0 ± 2.3211		0.4604
MLM-KHNN	62.14 ± 15.21	82.425 ± 3.304 ms	7.074 ± 1.56 ms	3.0 ± 2.4413		0.7478
Fuzzy-Hubness	57.8 ± 15.1	13593.561 ± 453.194 ms	2.709 ± 0.781 ms	2.0 ± 0.6574	0.0 ± 1.6224	0.1813

Tabela 5.3.4 Resultados na base de dados *Ionosphere Clustered*

Algoritmo	Precisão	Treinamento	Teste	k	θ	*p-value*
k-NN	**64.12 ± 14.68**	100.518 ± 2.733 ms	0.359 ± 0.021 ms	19.0 ± 0.0		
Local-Best k	**64.12 ± 14.68**	117.405 ± 3.327 ms	4.79 ± 0.344 ms	14.0 ± 3.5504		0.9999
MLM-KHNN	**64.12 ± 14.68**	72.027 ± 1.955 ms	9.022 ± 0.492 ms	1.0 ± 0.0		0.9999
Fuzzy-Hubness	**64.12 ± 14.68**	14918.134 ± 151.156 ms	4.895 ± 0.299 ms	20.0 ± 0.0	9.0 ± 0.0	0.9999

Tabela 5.3.5 Resultados na base de dados *Yeast Clustered*

Algoritmo	Precisão	Treinamento	Teste	k	θ	*p-value*
k-NN	36.58 ± 7.1	98.922 ± 3.4 ms	0.48 ± 0.045 ms	4.0 ± 2.183		
Local-Best k	36.18 ± 8.45	113.862 ± 4.382 ms	17.141 ± 0.75 ms	7.0 ± 3.0622		0.8443921524 2294952
MLM-KHNN	**38.87 ± 7.76**	79.508 ± 4.318 ms	55.524 ± 7.892 ms	9.0 ± 3.0734		0.2456452114 5927234
Fuzzy-Hubness	38.48 ± 6.06	13192.142 ± 392.752 ms	17.733 ± 0.771 ms	3.0 ± 1.9947	1.0 ± 1.0858	0.2788853846 0613171

É visível em todas as tabelas que houve uma redução na precisão dos algoritmos, o interesse de utilizar uma redução do tamanho de amostra está na redução do tempo de execução e treinamento, que em todas as amostras ocorreram. Mesmo assim, perder precisão é algo que se deseja evitar na tarefa de classificação, então observar se o ganho de velocidade vale a perda de precisão é algo que têm de ser avaliado quando comparando algoritmos desse tipo.

A influência do algoritmo de *clustering* não é tão grande na base de dados *Iris* devido a seu tamanho ser pequeno o bastante para o algoritmo de clustering afetar tanto os resultados. No caso da nossa implementação, leva de um banco balanceado com 50 amostras para cada classe, totalizando 150, para um de 45 amostras. Mesmo assim, isso prova que pode haver uma redução no tamanho de amostras e ainda assim ter uma precisão considerável. Nesse exemplo, mesmo havendo algoritmos com maior precisão média maior, não há diferença estatística.

Na tabela 5.3.2, a precisão e tempo necessário foi reduzido. A redução do tempo necessário para classificação nos outros testes foi maior, mas comparados ao algoritmo tradicional, não houve nenhum ganho estatisticamente relevante de precisão ou uma redução no número de *k*.

Na tabela 5.3.3, a maior média de precisão de resultados foi do algoritmo tradicional, mas como outros exemplos não houve uma diferença estatisticamente significativa. O resultados foram menos precisos do que a do banco original, mesmo utilizando um *k* muito similar aos que o *glass* original (5.2.3) utilizou.

Os resultados da base *Ionosphere Clustered,* tabela 5.3.4, demonstraram uma perda excessiva de informações para realizar a classificação. Isso pode expor uma fraqueza do algoritmo de *clustering*: Um valor alto de dimensões que já leva à esparsidade de informações se torna ainda pior com esse tipo de *clustering.*

Os resultados obtidos na tabela 5.3.4 utilizando a base de dados *ionosphere* foi um ponto de certa confusão. Todas as variações tiveram os mesmos resultados para precisão mesmo variando tamanho da vizinhança, tempo de treinamento e teste. Revisando o seu algoritmo, tanto o clustering como o uso de resultados, não detectamos nenhum erro nos algoritmos.

Rodamos novamente o teste do *ionosphere*, e toda a execução o valor da precisão mudava, mas os valores dos algoritmos eram sempre iguais. Decidimos

então aumentar o número de centros gerados para verificar se era isso que estava causando o problema de resultados iguais.

Nesta nova base de dados do ionosphere, nós aumentamos o número de centros utilizados para 50 enquanto o valor de vizinhos utilizados permaneceu com 3, resultando em 150 amostras de 351. Enquanto os resultados de 3 algoritmos permaneceram o mesmo com uma precisão de 64.12 ± 16.04, o *MLM-KHNN* teve um resultado diferente com 67 ± 15.83, mas essa diferença não era estatisticamente relevante. Essa diferença de resultados de precisão mostrou que os resultados são diferentes, mas os algoritmos estão tendo as mesmas dificuldades para classificar no grupo de amostras reduzido, resultando neles conseguindo as mesmas precisões.

Para saber o exato motivo que leva à versão reduzida da base de dados apresentar esses valores seria necessário fazer uma análise mais aprofundada da base resultante e como tanto os métodos *kNN* e *Fuzzy C-means* clustering afetam ele. Devido ao foco desse trabalho, abstemos desta análise.

No caso o *Fuzzy-hubness* que deveria ser resistente à esses tipos de bases esparsas com grande dimensionalidade, possui a mesma performance dos outros algoritmos. O valor da média θ expõe que o algoritmo sempre tentava utilizar o algoritmo alternativo, e o número máximo de vizinhos mostra que ele não conseguiu definir um bom tamanho de vizinhança, sempre tentando utilizar a maior vizinhança.

Isso é visível nos outros algoritmos também, o *MLM-KHNN* nunca varia de um único vizinho, o tradicional também utiliza do maior valor possível, sem nunca variar. O Best-local k, mesmo se possui uma variação, não possui uma mudança de resultados na precisão comparados aos outros algoritmos, mostrando que essa variação não acarretou em nenhum ganho. O tradicional também não varia de escolha de vizinhos, permanecendo com o número máximo de vizinhos.

Similar aos resultados do banco de dados original, o *Local-best k* ainda possui resultados com precisões iguais ou menores até do algoritmo tradicional, mesmo se não há uma diferença estatística relevante.

Na versão *clustered* do banco *yeast*, o algoritmo que teve o melhor resultado de precisão média foi o *MLM-KHNN,* mesmo se não houve diferença estatística em relação ao *kNN*. Os resultados gerais foram muito inferiores ao do banco de

treinamento original em todos os algoritmos, mas o tempo de treinamento e teste necessários para executar a classificação também foram muito reduzidos, se tornando mais viável para utilização em problemas de classificação que precisam de um tempo reduzido para classificação. A utilização de um outro algoritmo de *clustering* pode acarretar uma maior precisão para bancos de dados específicos, o que é uma boa observação para pesquisas futuras.

6. CONCLUSÃO

A realização deste trabalho permitiu um entendimento mais aprofundado da área de aprendizado de máquina. A pesquisa inicial do assunto foi essencial para solidificar uma base teórica no que ela envolve, além de dar uma motivação para pesquisa com uma breve descrição de sua importância.

Uma análise mais minuciosa no grupo de métodos *kNN não só* ajudou a entender várias propriedades específicas do método, mas também da área de algoritmo de aprendizado de máquinas em geral. A análise de fatores que pioram a performance de algoritmos de classificação levou à um interesse de como tratar esses problemas e desenvolver melhores soluções.

A implementação de um grupo de soluções disponíveis na literatura foi importante para entender como funciona a execução de cada um destes algoritmos, problemas relacionados a sua escalabilidade e como cada conceito explicado nos artigos de cada um funciona na prática. Essa busca na literatura por algoritmos também permitiu o entendimento de técnicas diferentes e até específicas para certos domínios, o que pode sempre ser explorado se um já têm entendimento de regras específicas do banco de dados.

A aplicação desses algoritmos em múltiplos bancos de dados diferentes, cada um com suas propriedades específicas, demonstrou os verdadeiros efeitos que cada implementação teve. Esses efeitos puderam ser observados nos resultados obtidos e analisando cada um deles pudemos chegar num entendimento do que ocorreu em cada banco.

Enquanto houveram algoritmo com diferentes precisões médias em bases diferentes, a utilização do teste de hipóteses conseguiu verificar quando exatamente havia uma diferença estatística dos resultados. E pelo que foi verificado, o algoritmo *kNN* tradicional ainda possui um resultado bom de perfomance. Os algoritmos alternativos por outro lado, conseguiram ter resultados similares com reduções no número de *k* vizinhos para a classificação.

Desta forma, agora entendemos melhor a área de aprendizado de máquina supervisionado baseado em instâncias e quais são seus pontos fortes e fracos, além de possíveis soluções para seus pontos fracos que podem ser adaptadas ou

melhoradas, além de comprovarmos o que o que foi dito nos artigos dos autores que nós exploramos.

6.1. TRABALHOS FUTUROS

Verificando os resultados, a precisão do algoritmo de *Fuzzy-hubness* comprovou a eficácia do conceito de *hubness* em base de dados de alta dimensionalidade mesmo se não houve um ganho alto o bastante para ser estatisticamente relevante. Mesmo assim, o custo computacional para realizar classificações com ele ainda foi alto, tanto de espaço quanto de tempo necessário para classificar.

No algoritmo proposto por TOMAŠEV (2014), ele utilizava o voto de *hubs* que tinham *hubness* acima de certo limiar para votar e as amostras *órfãs* ou com baixo *hubness* utilizavam de outra fórmula para votar. O argumento era que *hubness* baixo não era bom para utilizar na votação, de forma que seria melhor utilizar uma função alternativa.

Na base de dados *Ionosphere* (Tabela 5.2.4), o *Fuzzy-Hubness* foi o que possuiu a melhor média de precisões, mesmo sem haver uma uma diferença estatística, e nesse ele também teve $\theta = 0$, o que indica que este algoritmo sempre priorizou o voto de *hubness* sobre votos alternativos. Como os que teriam maior pesos seriam amostras que tiveram um *hubness* alto, aqueles que possuíam baixo *hubness* tiveram influência mínima no conjunto.

Assim, a ideia da utilização do conceito *hubness* para *clustering* é um bom ramo para se explorar, como já foi proposto em [25] e [26].

Outra ideia relevante para se explorar seria a da geração de hubness com pesos baseados em distância, uma vez que amostras muito longes da amostra ainda teria a mesma influência de uma próxima. Entretanto, em algoritmos de alta dimensionalidade à ideia de distâncias se torna menos efetiva como nós vimos nesse trabalho, então seus efeitos podem ser piores do que o algoritmo tradicional. Ainda assim é uma viável área a ser explorada.

Os resultados elevados da precisão média do MLM-KHNN, mesmo se não estatisticamente relevante, demonstrou que a utilização do seu método pode vir a

aprimorar efetividade do k-NN, com um bom ramo a ser explorado seria a variação de certos fatores como a escolha de vetor médios e da média para encontrar resultados mais precisos.

Enquanto o valor de vetores médios atual é bom para manter os valores das médias mais condensados, eles podem ainda dar muito peso à uma amostra próxima. Em casos onde um ruído esteja muito próximo, a não ser que os valores de k e r sejam elevados, o fato que o vetor médio da primeira amostra ser ela mesma leva à utilização de média harmônica dar muito peso a ela, o que pode causar erros na classificação.

7. REFERÊNCIAS

[1] IMANDOUST, Sadegh Bafandeh; BOLANDRAFTAR, Mohammad. Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. **International Journal of Engineering Research and Applications**, v. 3, n. 5, p. 605-610, 2013.

[2] RUSSELL, Stuart; NORVIG, Peter; INTELLIGENCE, **Artificial. A modern approach. Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs**, v. 25, p. 27, 1995.

[3] KOTSIANTIS, Sotiris B.; ZAHARAKIS, Ioannis D.; PINTELAS, Panayiotis E. Machine learning: a review of classification and combining techniques. **Artificial Intelligence Review**, v. 26, n. 3, p. 159-190, 2006.

[4] LOHR, Steve. The age of big data. **New York Times**, v. 11, n. 2012, 2012.

[5] HU, Li-Yu et al. The distance function effect on k-nearest neighbor classification for medical datasets. **SpringerPlus**, v. 5, n. 1, p. 1304, 2016.

[6] GARCÍA-PEDRAJAS, Nicolás; DEL CASTILLO, Juan A. Romero; CERRUELA-GARCÍA, Gonzalo. A Proposal for Local $ k $ Values for $ k $-Nearest Neighbor Rule. **IEEE transactions on neural networks and learning systems**, v. 28, n. 2, p. 470-475, 2017.

[7] PAN, Z., WANG, Y., & KU, W. (2017). A new k-harmonic nearest neighbor classifier based on the multi-local means. ***Expert Systems with Applications***, *67*, 115-125.

[8] FAITH, D. P., MINCHIN, P. R., & BELBIN, L.. Compositional dissimilarity as a robust measure of ecological distance. ***Vegetatio***, *69*(1-3), 57-68. 1987.

[9] ZHENG, K., SI, G., DIAO, L., ZHOU, Z., CHEN, J., & YUE, W. (2017, May). Applications of support vector machine and improved k-Nearest neighbor algorithm in fault diagnosis and fault degree evaluation of gas insulated switchgear. **In *Electrical Materials and Power Equipment (ICEMPE), 2017 1st International Conference on*** (pp. 364-368). IEEE.

[10] JAIN, A. K., MURTY, M. N., & Flynn, P. J. (1999). **Data clustering: a review. *ACM computing surveys (CSUR)***, *31*(3), 264-323.

[11] Nilsson, Nils J. ***Principles of artificial intelligence***. Morgan Kaufmann, 2014.

[12] Haykin, Simon. ***Redes neurais: princípios e prática***. Bookman Editora, 2007.

[13] Michalski, Ryszard S., Jaime G. Carbonell, and Tom M. Mitchell, eds. ***Machine learning: An artificial intelligence approach***. Springer Science & Business Media, 2013.

[14] Tomašev, Nenad, et al. "The role of hubness in clustering high-dimensional data." *IEEE **Transactions on Knowledge and Data Engineering*** **26.3** (2014): 739-751.

[15] Tomašev, Nenad, et al. "Hubness-based fuzzy measures for high-dimensional k-nearest neighbor classification." ***International Journal of Machine Learning and Cybernetics*** **5.3** (2014): 445-458.

[16] Tomašev, Nenad, and Krisztian Buza. "Hubness-aware kNN classification of high-dimensional data in presence of label noise." ***Neurocomputing*** **160** (2015): 157-172.

[17] SAGIROGLU, Seref; SINANC, Duygu. Big data: A review. In: **Collaboration Technologies and Systems (CTS), 2013 International Conference on**. IEEE, 2013. p. 42-47.

[18] KOTSIANTIS, Sotiris B.; ZAHARAKIS, I.; PINTELAS, P. Supervised machine learning: A review of classification techniques. 2007.

[19] Kouiroukidis, Nikolaos, and Georgios Evangelidis. "The effects of dimensionality curse in high dimensional knn search." *Informatics (PCI), 2011 15th Panhellenic Conference on*. IEEE, 2011.

[20] Verleysen, Michel, and Damien François. "The Curse of Dimensionality in Data Mining and Time Series Prediction." ***IWANN*. Vol. 5**. 2005.

[21] Keller, James M., Michael R. Gray, and James A. Givens. "A fuzzy k-nearest neighbor algorithm." ***IEEE transactions on systems, man, and cybernetics* 4** (1985): 580-585.

[22] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[23] KUSNER, Matt et al. Stochastic neighbor compression. In: **Proceedings of the 31st international conference on machine learning (ICML-14)**. 2014. p. 622-630.

[24] Chakrabarti, Soumen, et al. "Data mining curriculum: A proposal (Version 1.0)." *Intensive Working Group of ACM SIGKDD Curriculum Committee* (2006): 140.

[25] Shenbakapriya, R., M. Kalimuthu, and P. Sengottuvelan. "Improving Clustering Performance on High Dimensional Data using Kernel Hubness." *IJCA Proceedings on International Conference on Simulations in Computing Nexus*. No. 2. Foundation of Computer Science (FCS), 2014.

[26] Tomašev, Nenad, et al. "Hubness-based clustering of high-dimensional data." *Partitional clustering algorithms*. Springer International Publishing, 2015. 353-386.