CbDGen: A Complexity-based Synthetic Dataset Generation Tool

Thiago R. França[1], **Péricles B.C. Miranda**[1], **Ricardo B. C. Prudêncio**[2], **André C. A. Nascimento**[1]

[1]Departamento de Computação – Universidade Federal Rural de Pernambuco (UFRPE)
Recife – PE – Brazil

{pericles.miranda}@ufrpe.br

Abstract. *Synthetic database generation tools have gained attention as an alternative for the effective evaluation of classifiers. As they are highly configurable, the tools allow the creation of bases with specific characteristics, allowing the classifier's evaluation in different scenarios. This work proposes CbDGen, a tool for generating synthetic bases based on data complexity. With a simple and configurable interface, CbDGen allows the insertion of basic information (i.e., number of classes, attributes) and selecting different distributions and noise levels. Its great differential is to allow the choice of the type of complexity desired (i.e., balance, separability), as well as the level of this complexity. The results showed that CbDGen is capable of producing synthetic bases adhering to the user's choices, proving to be a propitious platform for the evaluation of classifiers.*

Resumo. *Ferramentas de geração de bases de dados sintéticos ganharam atenção como uma alternativa para a avaliação eficaz de classificadores. Como são altamente configuráveis, as ferramentas possibilitam a criação de bases com características específicas, permitindo a avaliação do classificador em diferentes cenários. Este trabalho propõe a CbDGen, uma ferramenta para geração de bases sintéticas baseada em complexidade de dados. Com uma interface simples e parametrizável, a CbDGen possibilita a inserção de informações básicas (i.e., número de classes, atributos), a seleção de diferentes distribuições e nível de ruído. O seu grande diferencial é permitir que a escolha do tipo de complexidade desejado (i.e., balanceamento, separabilidade), bem como o nível desta complexidade. Os resultados mostraram que a CbDGen é capaz de produzir bases sintéticas aderentes às escolhas realizadas pelo usuário, se mostrando uma plataforma propícia para a avaliação de classificadores.*

1. Introdução

Diversos algoritmos de classificação têm sido desenvolvidos e aplicados em uma grande variedade de domínios como medicina, segurança, negócios, e educação. Como saber qual algoritmo é mais adequado para um dado problema? Uma prática comum para avaliar o desempenho de um ou mais classificadores é treiná-los e testá-los em diferentes bases de dados, em sua maioria provenientes de repositórios públicos (Lorena et al. 2019). Essa prática pode trazer uma má interpretação dos resultados por

duas razões (Amancio et al. 2014). A primeira é que a estimativa de desempenho do classificador pode estar associada à sua própria limitação ou à complexidade da própria base de dados (por exemplo, dados pouco representativos, dados faltosos, desbalanceamento, e outros). Testar diferentes algoritmos nas mesmas bases de dados pode parecer superar esse problema, pois a complexidade do conjunto de dados é comum a todas as abordagens. No entanto, isso também pode levar a conclusões enganosas, pois existem características dos conjuntos de dados que afetam algoritmos de maneiras diferentes. A segunda limitação é que mesmo que seja feita uma comparação consistente com uma variedade de bases de dados do mundo real, os resultados ainda permanecem específicos aos dados usados (Lorena et al. 2019). Tentar estender as informações obtidas nessas análises para um conjunto de dados diferente pode ser ineficaz. Além disso, a obtenção de mais bases de dados reais, com diferentes características, para avaliar classificadores em diferentes aspectos, é uma tarefa difícil e muitas vezes custosa (Lorena et al. 2019).

Diante disto, geradores de bases de dados sintéticas ganharam a atenção dos pesquisadores como alternativa para a avaliação de classificadores. Os geradores de dados controlam os aspectos dos dados (i.e. padrões, tendências, tipo de dados, *outliers*, dimensões) para que suas características se ajustem a um problema específico. Isto fornece uma diversidade de conjuntos de dados diferentes para testar exaustivamente algoritmos de aprendizado de máquina de maneira controlada (Lorena et al. 2019). Embora bases de dados sintéticos não estejam associadas diretamente a conjuntos de dados reais específicos, elas podem ser usadas como representações de grandes classes de dados (Amancio et al. 2014). Existem vários algoritmos e ferramentas que podem ser usados para criar dados sintéticos (Mendonça et al. 2020). No entanto, a maior parte do trabalho realizado até o momento foi desenvolvida para aplicações específicas (Mendonça et al. 2020). Além disso, os dados gerados por estas ferramentas costumam seguir distribuições conhecidas, ou combinações de funções (Mendonça et al. 2020). Diferente de abordagens prévias, o escopo deste trabalho envolve o uso de medidas de complexidade para a geração de dados sintéticos. As medidas de complexidade foram introduzidas por (Ho and Basu 2002), e extraem características estatísticas e geométricas da base de dados visando estimar a sua complexidade.

O presente trabalho propõe uma ferramenta de geração de dados sintéticos, chamada de CbDGen, que usa algoritmos de otimização para gerar bases de dados com características de complexidade desejadas pelo usuário. Nesse caso, tenta-se reduzir a distância entre a complexidade das bases de dados geradas e o valor de complexidade definido pelo usuário como alvo. A CbDGen possui uma interface amigável e parametrizável, permitindo que o usuário selecione parâmetros como, o número de dimensões, número e tipo dos atributos, número de instâncias e classes, e tipo de distribuição. Além disso, o especialista pode definir quais características a base deve ter em termos de diferentes medidas de complexidade (i.e., balanceamento, separabilidade, sobreposição). A ferramenta também conta com um módulo, que possibilita a visualização da base de dados gerada, e dos resultados de obtidos por diferentes classificadores em determinadas medidas de desempenho. Para avaliação da ferramenta, foram propostos diferentes cenários de uso. Os resultados mostraram que o processo de geração adotado pela ferramenta é capaz de produzir bases de dados aderentes às escolhas realizadas pelo usuário. Isto foi mostrado através de visualizações gráficas, e da satisfação das funções objetivo.

Este artigo está estruturado da seguinte forma: A seção 2 apresenta as principais medidas de complexidade existentes. A seção 3 apresenta trabalhos relacionados. A seção 4 detalha a ferramenta proposta. A seção 5 apresenta os experimentos realizados para avaliar o método proposto, bem como os resultados obtidos. A seção 6 destaca as conclusões e trabalhos futuros.

2. Complexidade dos Dados

É um fato conhecido na comunidade de aprendizado de máquina que o desempenho de um classificador depende fortemente da natureza dos dados usados na análise de treinamento e teste (Lorena et al. 2019). Diante disso, a análise da complexidade dos dados é uma tarefa que tem alcançado grande relevância no entendimento da relação desempenho-complexidade, de como superar essa dependência, e como melhorar o desempenho do classificador.

O trabalho desenvolvido por (Ho and Basu 2002) é um dos trabalhos a propor medidas de complexidade para caracterizar um determinado problema de classificação, a fim de estimar sua dificuldade. Ultimamente, essas medidas têm sido utilizadas para diferentes fins, como caracterizar o domínio de competência de diferentes algoritmos de aprendizado de máquina (Luengo and Herrera 2015); descrição de problemas de classificação em estudos de meta-aprendizagem (Garcia et al. 2016); e geração de novas bases de dados de classificação (Macia and Bernado-Mansilla 2014). De acordo com (Lorena et al. 2019), as principais medidas de complexidade podem ser agrupadas nas seguintes categorias:

- *Medidas baseadas em atributos*: quantificam o quão informativo são os atributos disponíveis para a separação das classes;
- *Medidas de linearidade (separabilidade)*: quantificam se as classes do problema são linearmente separáveis;
- *Medidas de vizinhança*: quantificam a presença e densidade de classes idênticas e diferentes em vizinhanças locais, podendo ser útil na captura da forma do limite de decisão e caracterizar a sobreposição de classes;
- *Medidas de rede*: extraem a informação estrutural da base de dados através da sua modelagem em formato de grafo;
- *Medidas de dimensionalidade*: quantifica a esparsidade dos dados com base no número de amostras relacionadas à dimensionalidade dos dados;
- *Medidas de desbalanceamento*: quantifica a razão do número de exemplos entre as classes.

3. Trabalhos Relacionados

A geração de conjuntos de dados sintéticos para teste tem importância em muitas áreas da computação, como visualização de dados, mineração de dados, engenharia de software e inteligência artificial (Popić et al. 2019).

(Albuquerque et al. 2011) descreveram uma ferramenta para gerar dados multidimensionais. O usuário pode construir uma representação dos dados desejados manipulando distribuições estatísticas através da interface gráfica. No entanto, é limitada a números inteiros e flutuantes, não abordando a geração de dados categóricos. (Wang et al. 2013) apresentaram uma aplicação em que é possível que o usuário desenhe

a distribuição dos dados manualmente. Assim, o sistema cria o modelo de dados dos geradores com base no que o usuário desenhou. (Liu et al. 2016) criou um gerador de dados sintéticos para avaliar a classificação das regras de aprendizado. O trabalho gera regras de aprendizado com base nos atributos inseridos pelo usuário para construir relacionamentos entre esses atributos, e a técnica utilizada foi a árvore de decisão. Existem também trabalhos que produzem dados para problemas específicos, como (Dahmen and Cook 2019) que escreveram um artigo sobre o sistema de geração de dados para aplicativos de assistência médica, limitando a geração de novos dados apenas para esses casos.

Um trabalho recente desenvolvido por (Mendonça et al. 2020) propõe uma ferramenta de geração de dados sintéticos que permite que os usuários definam e componham distribuições estatísticas conhecidas para produzir o resultado desejado. Além disso, possibilita a visualização do comportamento dos dados em tempo real para analisar se possui as características necessárias para testes eficientes. Existe uma variedade de geradores de distribuições disponíveis, tais como, aleatório (i.e. uniforme, gaussiano e outros); geométrico (i.e. Bézier); funções (i.e. linear, quadrática, logarítmica e outras). Além disso, disponibiliza geradores acessórios para dados faltantes, escala linear, ruído, e outros.

O presente trabalho propõe uma ferramenta cujo foco, diferente dos demais trabalhos, não é gerar dados sintéticos apenas a partir de distribuições escolhidas e geradores acessórios, mas também da escolha de complexidades desejadas pelo especialista. O usuário deve fornecer como entrada: a descrição da base (i.e. número de atributos, classes, instâncias, tipo dos atributos); o tipo de distribuição desejada (i.e. lua, círculos concêntricos); e os níveis de complexidade que deseja que a base sintética possua, em termos de balanceamento, separabilidade e sobreposição.

4. CbDGen: Ferramenta para Geração de Dados Sintéticos baseada em Complexidade de Dados

Este trabalho propõe uma nova ferramenta para geração de dados sintéticos baseada em complexidade. Esta ferramenta é chamada de CbDGen, e sua arquitetura é apresentada na Figura 1. Cada módulo da ferramenta será detalhado nas seções a seguir.

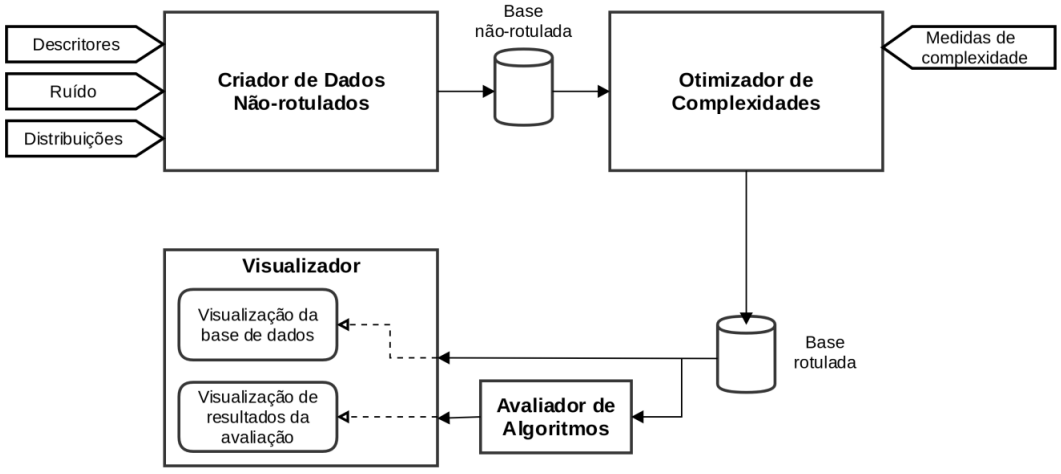


Figura 1. Arquitetura da CbDGen.

4.1. Módulo Criador de Dados Não-rotulados

A primeira etapa a ser realizada na ferramenta é a criação de uma base de dados composta por instâncias não-rotuladas. Para a criação desta base é necessário que o usuário forneça os seguintes dados de entrada: dados descritores da base, nível de ruído e tipo de distribuição desejada.

Descritores: O usuário deve definir informações que descrevem a base de dados, tais como: número de atributos, e o tipo de cada atributo; número de instâncias (ou registros) e classes. Esses dados são fornecidos através da tela apresentada na Figura 2 (esq.).

Distribuições: O usuário deve definir o tipo de distribuição a ser adotada na base de dados sintéticos. As distribuições disponíveis na ferramenta são: bolhas com centroide(s), através de funções Gaussianas; luas; círculos concêntricos; ou nenhuma distribuição específica. A escolha da distribuição é feita na tela apresentada na Figura 2 (centro).

Ruído: O usuário pode incluir ruído na base de dados sintéticos. O nível de ruído pode ser configurado com valores entre 0 (nenhum ruído) e 0.5 (nível alto de ruído). O valor de ruído é escolhido na tela da Figura 2 (centro).

Figura 2. Telas para fornecimento de informações para a geração de bases sintéticas. A tela (esq.) recebe dados descritivos. Na tela (centro), o usuário escolhe a distribuição e o valor de ruído. Na tela (dir.), o usuário especifica quais complexidades deseja considerar e seus respectivos níveis.

A saída deste módulo é uma base de dados de instâncias não-rotuladas inicializadas aleatoriamente, seguindo a parametrização definida pelo usuário. Esta base é fornecida como entrada para o módulo Otimizador de Complexidades.

4.2. Módulo Otimizador de Complexidades

Este módulo é o núcleo do processo de geração de dados sintéticos baseado em medidas de complexidade. Este módulo rotula as instâncias da base não-rotulada tentando satisfazer uma ou mais medidas de complexidade. Inicialmente, é fornecida uma base de dados sem rótulos (da etapa anterior) para um algoritmo inteligente de otimização, uma vez que esta tarefa se trata de um problema tradicional de otimização combinatória de natureza exponencial (França et al. 2020). O algoritmo de otimização tem o objetivo de encontrar a melhor combinação de instâncias-rótulos (solução) dentre todas as possíveis combinações (espaço de soluções), que satisfaça as medidas de complexidade definidas pelo usuário.

A ferramenta disponibiliza seis medidas de complexidade: uma medida de linearidade (L2), duas medidas baseadas em vizinhança entre instâncias de diferentes classes (N1 e N2), uma medida de balanceamento de classes (C2), uma de sobreposição de instâncias de diferentes classes (F1), e uma medida de agrupamento (ClsCoef). Mais detalhes a respeito destas medidas podem ser encontradas em (Lorena et al. 2019). Cada medida é descrita a seguir:

Error rate of linear classifier (L2): A medida L2 calcula a taxa de erro do linear da *Support Vector Machine* (SVM). Esta medida é calculada usando a seguinte equação:

$$L2 = \frac{\sum_{n}^{i=1} I(h(x_i) \neq y_i)}{n}. \tag{1}$$

Fraction of borderline points (N1): A medida N1 estima o tamanho e a complexidade do limite de decisão necessário por meio da identificação dos pontos críticos na base de dados - aqueles muito próximos um do outro, mas pertencentes a diferentes classes. Esta medida é calculada através da seguinte equação:

$$N1 = \frac{1}{n}\sum_{i=1}^{n} I((x_i, x_j)\epsilon MST \wedge y_i \neq y_j). \tag{2}$$

onde MST é o algoritmo de Prim.

Ratio of average intra/inter nearest neighbor (NN) distance (N2): A medida N2 calcula a razão entre duas somas, i) a soma das distâncias entre cada exemplo e seu vizinho mais próximo da mesma classe (intra-classe), ii) a soma das distâncias entre cada exemplo e seu vizinho mais próximo de outra classe (extra-classe). Esta medida é calculada através da seguinte equação:

$$intra\_extra = \frac{\sum_{i=1}^{n} d(x_i, NN(x_i)\epsilon y_i)}{\sum_{i=1}^{n} d(x_i, NN(x_i)\epsilon y_j \neq y_i)}. \tag{3}$$

Imbalance ratio (C2): A medida C2 é um índice que mede o balanceamento das classes. Este índice é calculado através da seguinte equação:

$$C2 = 1 - \frac{1}{IR},\ onde,\ IR = \frac{n_c - 1}{n_c}\sum_{i=1}^{n_c}\frac{n_{c_i}}{n - n_{c_i}}. \tag{4}$$

Maximum Fisher's Discriminant Ratio (F1): A medida de F1 mede a sobreposição entre os valores dos atributos em diferentes classes. Esta medida é calculada através da seguinte equação:

$$F1 = \frac{1}{1 + \max_{i=1}^{m} r_{f_i}}, \tag{5}$$

Clustering coefficient (ClsCoef): A medida de ClsCoef calcula o coeficiente de agrupamento de um vértice v_i (cada vértice é uma instância da base) que é dado pela razão

entre o número de arestas entre seus vizinhos e o número máximo de arestas possíveis entre eles. Bases de dados mais simples tenderão a ter conexões densas entre exemplos da mesma classe. Esta medida é calculada através da seguinte equação:

$$ClsCoef = 1 - \frac{1}{n}\sum_{i=1}^{n}\frac{2|e_{jk} : v_j, v_k \ \epsilon \ N_i|}{k_i\left(k_i - 1\right)}, \tag{6}$$

Cada uma das medidas de complexidade produz um valor no intervalo entre 0 e 1. Quanto mais próximo de 0, significa que o problema é mais simples; e quanto mais próximo de 1, mais complexo. Por exemplo, se a medida $L2 = 0$, significa que os dados são linearmente separáveis. Por outro lado, se $L2 = 1$, significa que os dados são não linearmente separáveis ao extremo, tornando o problema mais complexo. A escolha das complexidades desejadas é realizada na tela da Figura 2 (dir.). Nela, o usuário seleciona a(s) medida(s) que deseja, e a(s) preenche com valores entre 0 e 1. Esses valores definidos pelo usuário são chamados de *target* ou valores alvo, pois o algoritmo de otimização tentará encontrar soluções que mais se aproximem das complexidades desejadas.

Para calcular o valor de aptidão de uma solução candidata (combinação de rótulos), cada uma das medidas de complexidade são calculadas sobre a base de dados rotulada (com os rótulos da solução). As funções objetivo/aptidão aqui definidas se tratam da distância entre o valor obtido, subtraído pelo valor de complexidade desejado pelo especialista. Deste modo, existem seis funções objetivo (f_{L2}, f_{N1}, f_{N2}, f_{C2}, f_{F1}, e $f_{ClsCoef}$) para minimização da distância entre o valor alcançado e o valor esperado. As seis funções objetivo são:

$$f_{L2} = |target\_L2 - L2| \tag{7}$$
$$f_{N1} = |target\_N1 - N1| \tag{8}$$
$$f_{C2} = |target\_C2 - C2| \tag{9}$$
$$f_{F1} = |target\_F1 - F1| \tag{10}$$
$$f_{ClsCoef} = |target\_ClsCoef - ClsCoef|, \tag{11}$$

onde, $target\_L2$, $target\_N1$, $target\_N2$, $target\_C2$, $target\_F1$, e $target\_ClsCoef$ são os valores alvo definidos pelo especialista. Cada um assumindo valores entre 0 (mais fácil) e 1 (mais difícil).

Dependendo do número de objetivos selecionado, o algoritmo de otimização usado muda. Diante disso, quando o número de objetivos é igual a um, é adotado um Algoritmo Genético mono-objetivo. Caso o número de objetivos seja maior que 1 e menor que quatro objetivos, é adotado o NSGA-II (Deb et al. 2002). Se o número de objetivos for superior a três, é adotado o NSGA-III (Deb and Jain 2014), assim como no trabalho de (França et al. 2020). A escolha desses algoritmos é justificada pela popularidade e bons resultados em problemas combinatórios. Todos os três algoritmos foram configurados com os valores de parâmetros apresentados na Tabela 1. Esses valores estão determinados no arquivo *optimizers.config*, e podem ser alterados manualmente. Pretende-se automatizar a definição destes atributos de acordo com as escolhas feitas pelo usuário em termos de número de instâncias e atributos.

Tabela 1. Parâmetros usados pelos algoritmos de otimização para geração de dados sintéticos.

Parâmetro	Valor
Método de inicialização	Uniforme aleatório
Tamanho da população	100
#Gerações	1000
Método de seleção	Torneio
Taxa de reprodução	0.05
Taxa de mutação	0.2
Taxa de cruzamento	0.7
Método de cruzamento	Two-point
Método de mutação	Shuffle Indexes

4.3. Avaliador de Algoritmos

O módulo avaliador de algoritmos mede o desempenho de diferentes classificadores na base de dados sintéticos em termos de diferentes medidas de avaliação. No momento existem quatro algoritmos pré-configurados: *Decision Tree*, *Naive Bayes*, SVM e *Random Forest*. Estes algoritmos são implementações provenientes da biblioteca Scikit-Learn[1], e seus parâmetros foram definidos com valores default. Para inclusão de novos algoritmos para avaliação, será habilitada a função de upload de arquivo *pickle*[2] com o objeto do algoritmo, e sua parametrização desejada. Para avaliação, o usuário pode selecionar diferentes medidas, como: acurácia, precisão, revocação, kappa e medida-F. Atualmente a ferramenta realiza apenas o experimento de validação cruzada 10-fold, executado 10 vezes para geração de média e desvio padrão. No entanto, pretende-se disponibilizar outras opções de experimentos (i.e. *Leave-one-out*) parametrizáveis.

4.4. Visualizador

O módulo visualizador dá opções de: visualização gráfica da base de dados sintéticos criada, bem como dos resultados de desempenho dos classificadores na respectiva base. Exemplos de visualizações gráficas e de resultados podem ser vistas na seção 5.

4.5. Implementação do CdBGen

A CbDGen é implementada em Python e R, e as seguintes bibliotecas foram usadas: Scikit learn para o uso de algoritmos de aprendizado de máquina, DEAP[3] para o uso dos algoritmos de otimização, e ECoL[4] que contém medidas de complexidade implementadas para caracterizar problemas de classificação. Essas bibliotecas são confiáveis e amplamente utilizadas em trabalhos científicos.

5. Experimentos e Resultados

Nesta seção são apresentados alguns cenários de uso da ferramenta, em termos de: visualização gráfica das bases de dados geradas; e visualização dos resultados de avaliação dos classificadores.

[1]https://scikit-learn.org/

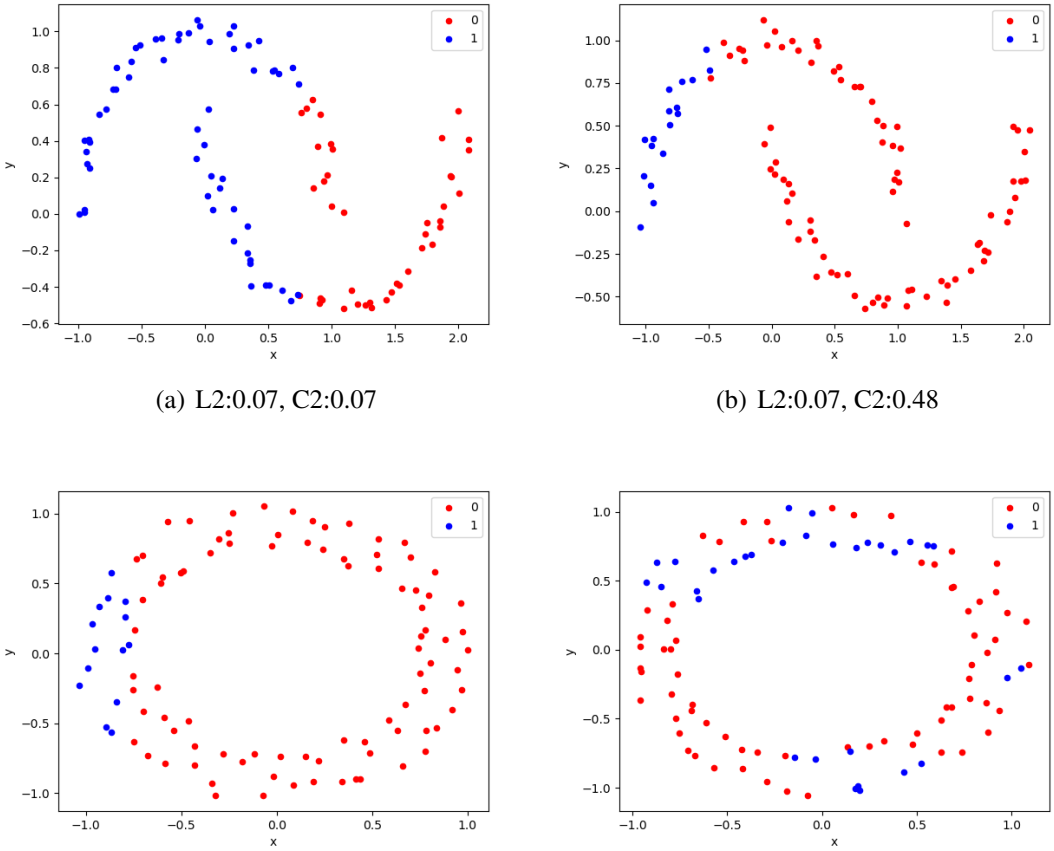
[2]https://docs.python.org/3/library/pickle.html

[3]https://deap.readthedocs.io/en/master/

[4]https://cran.r-project.org/web/packages/ECoL/index.html

5.1. Visualização Gráfica

Para a visualização gráfica foram definidos diferentes cenários. A Figura 3 apresenta bases de dados sintéticos que foram geradas a partir de quatro cenários de uso, e que foram apresentadas no módulo visualizador.



(a) L2:0.07, C2:0.07

(b) L2:0.07, C2:0.48

(c) L2:0.07, N1:0.07, C2:0.48

(d) L2:0.22, N1:0.22, C2:0.22

Figura 3. Cenários de uso e os respectivos valores *target* usados para a otimização.

Nos dois primeiros cenários (Figuras 3-(a) e -(b)), a parametrização escolhida pelo usuário foi: atributos reais no intervalo entre 0 e 1, 100 instâncias, e duas classes; distribuição em formato de lua; ruído igual a 0.05; foram escolhidos apenas dois objetivos, L2 e C2, a serem otimizados (valores nas Figura 3). O algoritmo NSGA-II é escolhido automaticamente uma vez que apenas dois objetivos foram selecionados. Nos outros dois cenários (Figuras 3-(c) e -(d)), a parametrização escolhida pelo usuário foi: atributos reais no intervalo entre 0 e 1, 100 instâncias, e duas classes; distribuição em formato de círculos concêntricos; ruído igual a 0.05; foram escolhidos quatro objetivos, L2, N1, N2 e C2, a serem otimizados (valores nas Figura 3). O algoritmo NSGA-III é escolhido automaticamente uma vez que mais de três objetivos foram selecionados.

A base de dados com os valores *target* $L2 = 0.07$ e $C2 = 0.07$ deve ter um comportamento linearmente separável e balanceado. Já uma base de dados com os valores *target* $L2 = 0.07$ e $C2 = 0.48$ deve ter um comportamento linearmente separável e

desbalanceado ao extremo. Como se pode ver nas Figuras 3-(a) e -(b), respectivamente, os comportamentos esperados em ambos os cenários foram alcançados através da otimização do NSGA-II.

A base de dados com os valores *target* $L2 = 0.07$, $N1 = 0.07$, $N2 = 0.07$ $C2 = 0.07$ deve ter um comportamento linearmente separável, com baixa sobreposição de instâncias de classes diferentes, e balanceado. Já uma base de dados com os valores *target* $L2 = 0.07$, $N1 = 0.07$, $N2 = 0.07$ $C2 = 0.07$ deve ter um comportamento não-linearmente separável, de alta sobreposição, e desbalanceado ao extremo. Como se pode ver nas Figuras 3-(c) e -(d), respectivamente, os comportamentos esperados em ambos os cenários foram alcançados através da otimização do NSGA-III.

Além da visualização gráfica, o módulo visualizador também disponibiliza os resultados da otimização. O intuito é que o usuário veja o quão aderente a base de dados gerada está em relação aos valores *target* definidos. A Tabela 2 mostra o resultado dos valores alcançados pela otimização em relação aos valores alvo dos quatro cenários anteriores. Como se pode ver, os algoritmos de otimização foram capazes de alcançar os valores *target* em quase todos os objetivos dos diferentes cenários. Naqueles objetivos em que os valores não foram iguais ao alvo, se chegou próximo do ótimo. Isto mostra que embora os objetivos sejam conflitantes, a otimização empregada foi eficaz na geração a base de dados, seguindo a complexidade estipulada.

Tabela 2. Valores das medidas de complexidades alcançados em relação aos respectivos alvos.

Cenários	***Target***	**Alcançado**
Figura 3-(a)	(0.07, 0.07)	(0.07, 0.07)
Figura 3-(b)	(0.07, 0.48)	(0.08, 0.48)
Figura 3-(c)	(0.07, 0.07 , 0.48)	(0.09, 0.07, 0.48)
Figura 3-(d)	(0.22, 0.22, 0.22)	(0.22,0.22, 0.22)

5.2. Visualização da Avaliação dos Classificadores

Nesta seção, são apresentados resultados da avaliação de diferentes classificadores em diferentes bases de dados sintéticos geradas pela ferramenta. Foram definidos dois diferentes cenários de complexidade. Ambos os cenários consideraram 100 instâncias, duas classes, e 2 atributos. Além disso, não foi selecionada nenhuma distribuição específica, e o nível de ruído é igual a 0.1. A diferença entre os cenários está na escolha do nível de complexidade. O primeiro cenário adotou $L2 = 0.07$, $C2 = 0.07$ e $N1 = 0.07$. Já o segundo cenário adotou $L2 = 0.20$, $C2 = 0.35$ e $N1 = 0.35$.

A Figura 4-(a), mostra a acurácia média alcançada pelos classificadores no primeiro cenário. Como se trata de um cenário mediano, os algoritmos alcançaram bons resultados atingindo uma acurácia média superior a 90%. Em termos de medida-F (Figura 4-(b)), o desempenho médio dos classificadores também é elevado, mostrando que são capazes de distinguir falsos positivos e falsos negativos.

As Figuras 4-(c) e -(d), mostram a acurácia e a medida-F média alcançadas pelos classificadores no segundo cenário. Neste cenário, a base considerada apresenta um nível maior de complexidade, apresentando um desbalanceamento intermediário, sobreposição

de instâncias de classes diferentes, e ainda é não linearmente separável. Como se pode ver, houve uma queda geral do desempenho dos algoritmos. A SVM foi a mais impactada neste novo cenário. Isto se explica, pois este algoritmo, em particular, tem dificuldade de lidar com problemas desbalanceados. Além de apresentar uma média abaixo em termos de acurácia e medida-F, a variância de seus resultados também é expressiva. De uma forma geral, a abordagem que obteve os melhores resultados foi a Random Forest, tanto em termos de média, como em variância. Atualmente. a CbDGen disponibiliza a visualização do desempenho em boxplots. No entanto, pretende-se incluir novos formatos de visualização de desempenho, tal como, matriz de confusão.

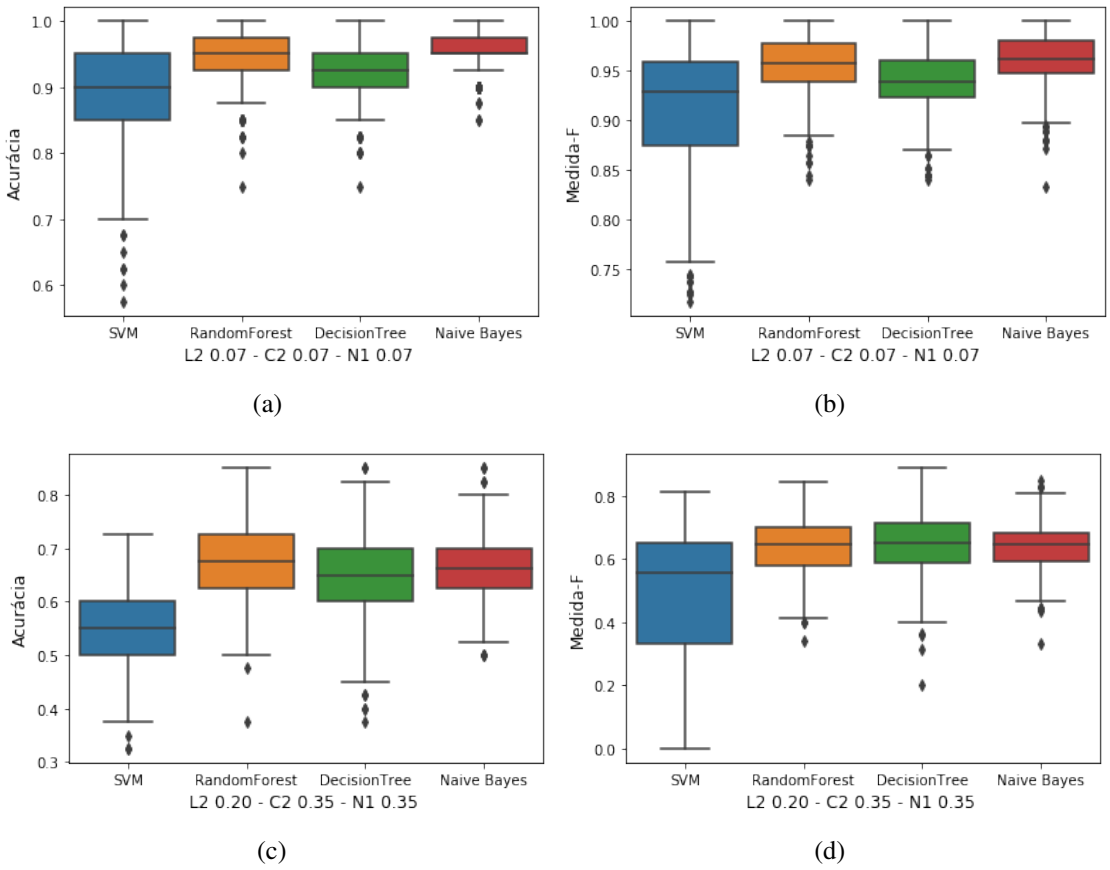


Figura 4. Avaliação de diferentes classificadores em termos de acurácia e medida-F.

6. Conclusão

Este artigo propõe a CbDGen, uma ferramenta para geração de bases de dados sintéticos baseada em complexidades. A CbDGen apresenta uma interface simples e parametrizável, possibilitando ao usuário informar as características da base que deseja criar. Além disso, a ferramenta disponibiliza uma funcionalidade de visualização gráfica da base de dados, bem como dos resultados das avaliações de diferentes classificadores. Neste artigo, foram apresentados diferentes cenários de uso da ferramenta. Os resultados mostraram que o processo de geração adotado pela ferramenta é capaz de produzir bases de dados aderentes às escolhas realizadas pelo usuário. Isto foi mostrado através de visualizações gráficas,

e da satisfação das funções objetivo. Como trabalhos futuros, pretende-se tornar a ferramenta mais completa, com novas medidas de avaliação, novas distribuições, e inclusão de outras características como percentual de *outliers*. Além disso, deseja-se criar a funcionalidade em que usuários fazem o upload dos seus próprios classificadores via API, e incluir novos formatos de visualizações de resultados de avaliação.

Referências

[Albuquerque et al. 2011] Albuquerque, G., Lowe, T., and Magnor, M. (2011). Synthetic generation of high-dimensional datasets. *IEEE transactions on visualization and computer graphics*, 17(12):2317–2324.

[Amancio et al. 2014] Amancio, D. R., Comin, C. H., Casanova, D., Travieso, G., Bruno, O. M., Rodrigues, F. A., and da Fontoura Costa, L. (2014). A systematic comparison of supervised classifiers. *PloS one*, 9(4).

[Dahmen and Cook 2019] Dahmen, J. and Cook, D. (2019). Synsys: A synthetic data generation system for healthcare applications. *Sensors*, 19(5):1181.

[Deb and Jain 2014] Deb, K. and Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Trans. Evolutionary Computation*, 18(4):577–601.

[Deb et al. 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

[França et al. 2020] França, T. R., Miranda, P. B., Lorena, A. C., and Nascimento, A. C. ((Accepted) 2020). A many-objective optimization approach for complexity-based data set generation. In *To Appear in IEEE Congress on Evolutionary Computation, 2020.*

[Garcia et al. 2016] Garcia, L. P., de Carvalho, A. C., and Lorena, A. C. (2016). Noise detection in the meta-learning level. *Neurocomputing*, 176:14–25.

[Ho and Basu 2002] Ho, T. K. and Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE transac. on pattern analysis and machine intelligence*, 24(3):289–300.

[Liu et al. 2016] Liu, R., Fang, B., Tang, Y. Y., and Chan, P. P. (2016). Synthetic data generator for classification rules learning. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pages 357–361. IEEE.

[Lorena et al. 2019] Lorena, A. C., Garcia, L. P., Lehmann, J., Souto, M. C., and Ho, T. K. (2019). How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys (CSUR)*, 52(5):1–34.

[Luengo and Herrera 2015] Luengo, J. and Herrera, F. (2015). An automatic extraction method of the domains of competence for learning classifiers using data complexity measures. *Knowledge and Information Systems*, 42(1):147–180.

[Macia and Bernado-Mansilla 2014] Macia, N. and Bernado-Mansilla, E. (2014). Towards uci+: A mindful repository design. *Information Sciences*, 261:237–262.

[Mendonça et al. 2020] Mendonça, S. D. P., Brito, Y. P. D. S., Dos Santos, C. G. R., Lima, R. D. A. D., De Araújo, T. D. O., and Meiguins, B. S. (2020). Synthetic datasets generator for testing information visualization and machine learning techniques and tools. *IEEE Access*, 8:82917–82928.

[Popić et al. 2019] Popić, S., Pavković, B., Velikić, I., and Teslić, N. (2019). Data generators: a short survey of techniques and use cases with focus on testing. In *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*, pages 189–194. IEEE.

[Wang et al. 2013] Wang, B., Ruchikachorn, P., and Mueller, K. (2013). Sketchpadn-d: Wydiwyg sculpting and editing in high-dimensional space. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2060–2069.