SERVIÇO PÚBLICO FEDERAL
UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA

PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA – PIBIC

Métodos Surrogates Aplicados à Otimização Multiobjetivo de Hiperparâmetros de SVMs

Análise de Desempenho de Algoritmo da Otimização com Muitos Objetivos Baseados em Surrogates

Área do conhecimento: Metodologia e Técnicas da Computação
Subárea do conhecimento: Inteligência Artificial
Especialidade do conhecimento: Otimização Multiobjetivo

Relatório Final
Período da bolsa: de (Setembro 2023) a (Agosto 2024)

Este projeto é desenvolvido com bolsa de iniciação científica

Autor: Carlos Eduardo Santana Nascimento
Orientador: André Britto de Carvalho

PIBIC/CNPq

Sumário

1 Introdução

A aprendizagem de máquina é uma subárea da inteligência artificial, ela lida com algoritmos computacionais que podem ser melhorados pelo treinamento de modelos. A base do treinamento dos modelos são dados, geralmente um modelo é treinado usando uma base de dados, para que então, possa ser possível utilizá-lo em outras bases (WITTEN et al., 2005). Existem diversos modelos que podem ser usados para alcançar seus objetivos, como: Redes Neurais, Regressão Linear, Support Vector Machine, etc (BISHOP, 2006).

Os modelos de aprendizagem dependem fortemente de uma boa escolha dos seus parâmetros, caso os parâmetros escolhidos não sejam bons o suficiente, a eficácia do modelo será prejudicada. Esses parâmetros são chamados de hiperparâmetros e devem ser definidos antes do processo de treinamento do modelo (BERGSTRA et al., 2011). Para encontrar bons parâmetros pode ser utilizado um processo de otimização, servindo como uma busca pelos melhores candidatos.

Na computação encontram-se diversos problemas que podem ser resolvidos utilizando mecanismos de busca, para isso, existem diversas técnicas diferentes que se adequam ao tipo de problema a ser resolvido. Dentro dessa área de busca, surge um conceito chamado metaheurística. Metaheurística é um conceito relacionado a otimização estocástica, a otimização estocástica é uma classe de algoritmos e técnicas que usam algum grau de aleatoriedade para encontrar soluções ótimas (LONES, 2011). Muitos desses algoritmos usam técnicas de inteligência artificial para obter os resultados.

No processo de busca normalmente o resultado almejado é chamado de resultado ótimo. A otimização é uma técnica que procura encontrar um valor mínimo para resolver um determinado problema (TAKAHASHI, 2007). Normalmente os problemas possuem uma função que calcula a qualidade de um resultado, essa função é chamada de função objetivo. Existem problemas que buscam otimizar mais de uma função ao mesmo tempo, esses problemas são chamados de problemas multiobjetivos (COELLO, 2007) e possuem uma maior complexidade para serem resolvidos. Dado o aumento de complexidade para calcular os valores de funções objetivos faz-se necessário, na maioria dos casos, utilizar técnicas de aprendizagem de máquina para melhorar a busca.

No processo de otimização, normalmente as funções associadas ao problema possuem um alto custo computacional, levando muito tempo para serem executadas. Para driblar este problema, pode ser feito o uso de surrogates. Surrogates são modelos de aprendizagem que são usados para substituir funções custosas, tendo uma velocidade mais rápida de execução (COELLO, 2007). O problema de hiperparâmetros se enquadra nessa classe de problemas de otimização, pois deve ser feito uma busca para encontrar a melhor configuração. Além disso, esse problema de hiperparâmetros pode ser modelado como um problema multiobjetivo, para isso será feito uma modelagem usando como base a sensitividade e a especificidade.

Sejam A e B as duas classes de um problema desse tipo. Cada métrica será usada como uma função objetivo, dessa forma, será buscado hiperparâmetros que maximizem a sensitivade e a especificidade. Ou seja, os hiperparâmetros onde o modelo classifica corretamente ambas as classes desejáveis (NETO,

2023). As variáveis escolhidas foram o parâmetro de regularização, chamado C, e o coeficiente de kernel, chamado gamma. Com isso, temos duas funções objetivos que serão otimizadas ao mesmo tempo, configurando este problema como multiobjetivo.

Existem na literatura diversos trabalhos que tratam do uso de surrogates para otimização. Em Métodos Surrogate Aplicados à Otimização Multiobjetivo de Hiperparâmetros de SVMs (NETO, 2023) é aplicado o uso de surrogates para analisar o problema de configuração de hiperparâmetros, além disso é feito uma comparação entre os métodos que usam surrogate e alguns que não utilizam. Em On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice (YANG; SHAMI, 2020) foi feita a otimização de hiperparâmetros de diversos modelos, incluindo SVM. Em A hyper-parameter tuning approach for cost-sensitive support vector machine classifiers (GUIDO; GROCCIA; CONFORTI, 2023) foi feita uma otimização de dois hiperparâmetros de uma SVM com kernel RBF. Além disso, existem um conjunto de surrogates que ainda não foram explorados no contexto do problema multiobjetivo de configuração de hiperparâmetros.

2 Objetivos

Este trabalho tem como objetivo analisar a performance de algoritmos multiobjetivos no problema de otimização de hiperparâmetros de SVMs. Estes algoritmos utilizam surrogates para prever resultados de funções, tornando o tempo de execução mais rápido e menos custoso. A análise é feita comparando resultados dos diferentes algoritmos que foram executados, usando conceitos de estatística para conferir esses dados. Para isso, serão comparados os algoritmos M1, DVL, TDEADP e NSGA-II, que utilizam modelos distintos, usando as métricas de hipervolume, precisão, AUC, F-measure e tempo para analisar a qualidade de cada resultado.

3 Metodologia

3.1 Conceitos básicos

A otimização busca encontrar um valor mínimo (ou máximo) para resolver um determinado problema. O problema pode ser definido como uma função $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$, onde a solução deve ser otimizada. Seja x^* a solução ótima para este problema, temos que (TAKAHASHI, 2007):

$$\begin{aligned} x^* &= argmin_{x \in X} f(x) \\ \text{sujeito a: } &\begin{cases} g(x) \leq 0 \\ h(x) = 0 \end{cases} \end{aligned} \tag{1}$$

f é chamada de função objetivo, enquanto g e h representam as restrições que devem ser satisfeitas dentro da otimização (COELLO, 2007).

A otimização multiobjetivo é definida como a minimização (ou maximização) de $F(x) = (f_1(x), f_2(x), ..., f_k(x))$, sujeito a $g_i(x) \leq 0$ e $h_j(x) = 0$. Dessa maneira,

a solução de um problema multiobjetivo é a minimização de $F(x)$, onde **x** será um vetor de variáveis de decisão com dimensão n. Podemos representar o vetor de variáveis de tamanho n da seguinte maneira: (COELLO, 2007)

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2}$$

Em um problema multiobjetivo normalmente temos soluções onde para um objetivo i um elemento é o melhor, porém para o objetivo j ele é pior. Para isso temos o conceito de Fronteira de Pareto, ela é definida como o conjunto de soluções em que cada elemento não é melhor que o outro, em outras palavras os elementos não se dominam.

Sejam dois elementos x e y, dizemos que x domina y, denotado por $x \preceq y$, se e somente se, x é parcialmente menor que y (COELLO, 2007), ou seja:

$$\begin{aligned} &\forall i \in 1, ..., k, x_i \leq y_i \\ &\exists i \in 1, ..., k, x_i < y_i \end{aligned} \tag{3}$$

Dessa forma o conjunto ótimo de Pareto para um dado problema multiobjetivo é definido:

$$P^* = \{x \in \Omega | \nexists x' \in \Omega, F(x') \preceq F(x)\} \tag{4}$$

Surrogate é uma técnica que utiliza modelos matemáticos para prever valores de uma função que teria um custo alto para ser calculada. Em outras palavras, o surrogate é um modelo de aprendizagem de máquina que foi treinado utilizando um conjunto de dados, para prever os valores de uma dada função. Seja a função f e o conjunto de dados x, calculando todos os valores de $f(x)$ e entregando esse resultado para o treinamento do surrogate, é possível obter uma nova função $g(x)$ que se aproxima dos valores de $f(x)$ (NETO, 2023). Existem diversos modelos de aprendizagem que podem ser treinados e esses modelos podem ter vantagens e desvantagens de acordo com o domínio do problema.

3.2 Configuração de Hiperparâmetros

A configuração de hiperparâmetros refere-se ao processo de selecionar os melhores valores para os hiperparâmetros do modelo de SVM. Os hiperparâmetros são parâmetros que não são aprendidos durante o treinamento do modelo, eles devem ser definidos antes do treinamento (BERGSTRA et al., 2011).

As variáveis de decisão escolhidas foram o parâmetro de regularização, chamado C, e o coeficiente do kernel, chamado gamma. As funções objetivos são o cálculo da sensitividade e especificidade do classificador, ou seja, será feito a busca de hiperparâmetros onde o modelo SVM classifica corretamente as soluções testadas. Dessa forma o problema pode ser modelado como multiobjetivo, dado as duas funções objetivas é feita a otimização de ambas, gerando novas soluções a cada iteração.

O processo de otimização multiobjetivo normalmente é muito custoso, dado que as funções objetivos nem sempre são facilmente calculáveis. Para melhorar este processo será utilizado surrogates, para substituir cálculos custosos dentro do processo de otimização. Cada modelo pode utilizar o surrogate em uma etapa específica, e também pode ser utilizado mais de um para tentar melhorar ainda mais o tempo de execução.

O processo de otimização de hiperparâmetros consiste em 4 componentes principais, o estimador (regressor ou classificador) com sua função objetivo, o espaço de busca, o método de otimização usado para encontrar as combinações de hiperparâmetros, e uma função de avaliação para comparar a performance das diferentes configurações (YANG; SHAMI, 2020).

3.3 Trabalhos relacionados

Em Decision Variable Learning (SANTOS; OLIVEIRA; BRITTO, 2019) é proposto um algoritmo que explora uma abordagem de surrogate em problemas de otimização multiobjetiva. Esse algoritmo usa como base a aprendizagem inversa, ou seja, são dados valores no espaço objetivo para treinamento do modelo, para que ele possa ser usado para prever valores no espaço de decisão. Como resultado o DVL performou melhor que o NSGA-III em alguns problemas.

Em Expensive Multiobjective Evolutionary Optimization Assisted by Dominance Prediction (YUAN; BANZHAF, 2021), foi proposto um algoritmo evolucionário assistido por surrogate para otimização de problemas multiobjetivos. Esse algoritmo usa uma estratégia de seleção em duas etapas, usando conceitos de deep learning para montar dois modelos de surrogates diferentes, um para prever a pareto dominância e outro para prever a θ-dominância. O objetivo deste trabalho é tentar melhorar certos defeitos da pareto dominância, pois ela não consegue regular a diversidade das soluções, para isso foi considerado outra dominância chamada θ-dominância, que consegue preservar a diversidade compensando as limitações da pareto dominância mantendo um balanço entre a convergência e a diversidade no espaço objetivo.

Em A Taxonomy for Metamodeling Frameworks for Evolutionary Multiobjective Optimization (DEB et al., 2018) foi proposto alguns frameworks usados em algoritmos multiobjetivos, dentre eles foi proposto o M1. No M1 todos os objetivos e restrições são modeladas independentemente. Dessa forma, o modelo é treinado inicialmente com as soluções exatas, e então é otimizado por algum algoritmo de forma progressiva. Portanto, no algoritmo é necessário que exista um modelo diferente para cada função objetivo do problema.

3.4 Algoritmos

3.4.1 M1

O algoritmo M1 usa como ponto principal um modelo diferente para cada função objetivo. O pseudocódigo é exibido em 1, inicialmente é feito o LHS para gerar a população inicial. Com isso, é realizado um laço, onde a população P (população inicial que armazena todos os indivíduos) é unida com a população Q (população que filtra apenas as melhores soluções de P), é calculado os objetivos das soluções na linha 9. Na linha 10 é gerado um novo modelo de surrogate para

cada função objetivo, o número de avaliações é incrementado, na linha 15 é feita a filtragem das melhores soluções. E por fim na linha 19 é feita a otimização utilizando um algoritmo evolucionário multiobjetivo (neste trabalho foi utilizado o NSGA-II (DEB et al., 2002)), esse ciclo é repetido até esgotarem o número de avaliações.

O algoritmo M1 utiliza um controle de evolução fixa (FEC), em que apenas alguns indivíduos são avaliados utilizando a previsão do surrogate, enquanto a população restante é avaliada utilizando a função de objetivo real (DEB et al., 2018). Neste trabalho o M1 fez a previsão de 25% dos indivíduos, ou seja, no teste com 100 avaliações, 25 delas foram previstas pelo surrogate, enquanto as 75 restantes foram calculadas utilizando a função objetivo real.

Algorithm 1 M1 pseudocódigo

1: $t \leftarrow 0$
2: $k \leftarrow 0$
3: $P \leftarrow LHS(\rho, n)$
4: $Q \leftarrow \varnothing$
5: $eval \leftarrow 0$
6: **while** $eval < SE_{max}$ **do**
7: **if** t mod $\tau = 0$ **then**
8: $P \leftarrow P \cup Q$
9: $F \leftarrow f(P)^T$
10: $S_i \leftarrow$ Create-Surrogate-Model(F_i)
11: $eval \leftarrow eval + \rho$
12: **if** $k = 0$ **then**
13: $Q \leftarrow LHS(\rho, n)$
14: **else**
15: $Q \leftarrow$ filtrar melhores μ soluções de P
16: **end if**
17: $k \leftarrow k + 1$
18: **end if**
19: $Q \leftarrow EMO(S, Q, \Gamma)$
20: $t \leftarrow t + 1$
21: **end while**
22: **return** soluções não-dominadas de P

3.4.2 DVL

O algoritmo DVL usa uma ideia um pouco diferente dos algoritmos convencionais, nele é utilizada um modelo inverso, ou seja, o modelo de aprendizagem é treinado usando os valores objetivos, para que seja possível prever valores no espaço de decisão.

O DVL é baseado em três etapas básicas. Primeiro a geração da população inicial. Segundo o treinamento do modelo de aprendizagem de acordo com os resultados das funções objetivos. E por último, a previsão de novas soluções baseadas nos pontos de referência ideais.

Nesse trabalho foi utilizado um modelo de regressão linear, visando a simplicidade e eficiência do algoritmo. O pseudocódigo do DVL pode ser visto em 2.

Algorithm 2 DVL algoritmo pseudocódigo

1: $P \leftarrow LHS(k)$
2: Avaliar(P)
3: $RP \leftarrow$ GerarPontosReferencia()
4: $P_{best} \leftarrow \varnothing$
5: $HV_{best} \leftarrow 0.0$
6: **repeat**
7: $\quad P_{rp} \leftarrow \varnothing$
8: $\quad$ **for** $r \in RP$ **do**
9: $\quad\quad P_{near} \leftarrow$ EncontrarSolucoesMaisProximas(r,P,c)
10: $\quad\quad M_n \leftarrow$ TreinarModelos(P_{near})
11: $\quad\quad s \leftarrow M_n.Predict(r)$
12: $\quad\quad P_{rp} \leftarrow P_{rp} \cup \{s\}$
13: $\quad$ **end for**
14: $\quad HV \leftarrow$ HipervolumeNormalizado(P_{rp})
15: $\quad$ **if** $HV > HV_{best}$ **then**
16: $\quad\quad HV_{best} \leftarrow HV$
17: $\quad\quad P_{best} \leftarrow P_{rp}$
18: $\quad$ **end if**
19: $\quad P \leftarrow P \cup P_{rp}$
20: **until** chegar ao máximo de iterações ou a diferença de hipervolumes for menor que ε

3.4.3 TDEADP

No algoritmo θ-DEA-DP é introduzido um novo conceito chamado θ-dominância. Nela temos um vetor de pesos que dividem o espaço objetivo em N clusters, em cada clusters é computado qual é a solução não dominada por meio de uma função chamada PBI (penalty boundary intersection), que faz o cálculo levando em consideração a distância entre o ponto e a origem e a distância entre o ponto e a projeção do vetor de referência (YUAN; BANZHAF, 2021). Basicamente uma solução só é comparada com soluções que pertencem ao mesmo cluster, trazendo uma maior diversidade para este algoritmo.

O framework do θ-DEA-DP (TDEADP) é mostrado no algoritmo 3. Inicialmente é gerado um vetor de pesos, em seguida a população é inicializada usando o método de Hipercubo Latino o tamanho da população inicial é de $11n - 1$. Nos passos das linhas 3-9 a população inicial é avaliada e armazenada em um arquivo $\mathbb{A}$. No passo 10 é feito o treinamento de uma rede neural para prever a Pareto dominância, chamada p-net. Com esta rede é possível prever, se dados dois indivíduos **x** e **y**, qual é a relação de dominância entre eles (se **x** domina **y**, se **y** domina **x** ou se são não-dominantes). No passo 11 é treinado outra rede neural para prever a θ-dominância.

Como foi dito anteriormente, os N vetores de pesos dividem o espaço objetivo em N clusters, nos passos 12 e 13, é determinado a θ-solução x_j^* e a Pareto-

solução y_j^* para cada cluster C_j. No algoritmo o tamanho da população é N, mas a população inicial tem tamanho $11n-1$, que pode ser maior que N, no passo 14 é feita a ordenação de não-dominância baseada na θ-dominância para selecionar N indivíduos caso $11n-1 > N$.

Nos passos seguintes é realizada a iteração para encontrar a melhor solução. Na linha 16 é escolhido o cluster alvo, linha 17 é gerado uma população de filhos (de tamanho N^*) utilizando o SBX crossover e a mutação polinomial.

No passo 18 é realizada uma pré-seleção de dois estágios auxiliada pela Pareto-net e θ-net, para selecionar uma solução $\mathbf{z}^*$ para a avaliação. É esperado que a solução $\mathbf{z}^*$ tenha melhorias consideráveis comparada com x_j^* e y_j^*. Em seguida é feito a atualização das redes e por fim na linha 26 a população é truncada para o tamanho N usando ordenação pela θ-não-dominância.

Algorithm 3 Framework $\theta - DEA - DP$

1: $\{w_1, w_2, ..., w_N\} \leftarrow$ InicializarVetorPesos(m)
2: $\mathbb{P} \leftarrow$ HipercuboLatino(n)
3: $evals \leftarrow 0$
4: $\mathbb{A} \leftarrow \varnothing$
5: **for** $x \in \mathbb{P}$ **do**
6: Avaliar(x)
7: $evals \leftarrow evals + 1$
8: $\mathbb{A} \leftarrow \mathbb{A} \cup x$
9: **end for**
10: p-net $\leftarrow InicializarParetoNet(\mathbb{A})$
11: $\theta - net \leftarrow Inicializar - \theta - net(\mathbb{A})$
12: $x_1^*, x_2^*, ..., x_n^* \leftarrow Get - \theta - Reps(\mathbb{A})$
13: $y_1^*, y_2^*, ..., y_n^* \leftarrow Get - Pareto - Reps(\mathbb{A})$
14: $\mathbb{P} \leftarrow$ TruncarPopulacao$(\mathbb{P}, N)$
15: **while** $evals < MaxEval$ **do**
16: $j \leftarrow$ EscolherIndexClusterAlvo(N)
17: $\mathbb{Q} \leftarrow$ GerarOffsprings$(\mathbb{P}, N^*)$
18: $\mathbf{z}^* \leftarrow$ PreSelecaoDoisEstagios$(\mathbb{Q}, x_j^*, y_j^*, p-net, \theta-net)$
19: Avaliar$(\mathbf{z}^*)$
20: $evals \leftarrow evals + 1$
21: $\mathbb{A} \leftarrow \mathbb{A} \cup \mathbf{z}^*$
22: AtualizarParetoNet(p-net,$\mathbb{A}$)
23: Atualizar-θ-Net$(\theta - net, \mathbb{A})$
24: Atualizar-θ-Reps$(x_1^*, x_2^*, ..., x_n^*, \mathbb{A})$
25: AtualizarParetoReps$(y_1^*, y_2^*, ..., y_n^*, \mathbb{A})$
26: $\mathbb{P} \leftarrow$ TruncarPopulacao$(\mathbb{P} \cup \mathbf{z}^*, N)$
27: **end while**

4 Resultados e discussões

Nesta seção serão discutidos os experimentos que foram realizados neste trabalho, a base de dados, os algoritmos e as métricas utilizadas para comparar a eficácia de cada algoritmo. Esses experimentos tem como objetivo comparar

os resultados de algoritmos diferentes para um mesmo problema de otimização multiobjetivo.

Os experimentos foram feitos usando 4 algoritmos diferentes, NSGA-II, DVL, TDEADP e M1, onde foram testados com 100 e 500 avaliações da função objetivo cada e o teste foi repetido 25 vezes para cada algoritmo, os resultados foram postos em tabelas e gerados gráficos para melhor comparação entre os algoritmos.

4.1 Métricas

Após serem realizadas as 25 execuções de cada algoritmo é calculado a média do tempo de execução, da AUC, F-measure, precisão e do hipervolume (GUERREIRO; FONSECA; PAQUETE, 2020).

Em seguida utilizando os valores calculados é feito um teste estatístico de Wilcoxon (DERRAC et al., 2011) com o hipervolume, dessa forma é montado um gráfico de boxplot e um heatmap para facilitar a visualização dos resultados. Com isso é feito uma comparação entre as métricas calculadas, usando como base o resultado do teste estatístico.

4.2 Algoritmos e parâmetros

A base de dados utilizada nos testes foi a Breast Cancer, ela foi coletada do repositório de aprendizagem de máquina da UCI. Possui 201 dados de uma classe e 85 da outra.

Todos os algoritmos utilizaram o SBX Crossover como algoritmo de cruzamento e Polynomial Mutation como algoritmo de mutação. A distribuição de cruzamento foi 20 e a probabilidade 0.9, enquanto a distribuição de mutação foi 20 e a probabilidade 0.5. O algoritmo de seleção foi o Binary Tournament, o torneio foi do tamanho da metade da população.

O M1 utilizou como algoritmo de otimização o NSGA-II com os mesmos parâmetros, a única mudança foi no tamanho das populações. O NSGA-II também foi testados sem surrogates. A tabela 1 com os parâmetros detalhados é mostrada abaixo:

Tabela 1: Parâmetros do M1 e NSGA-II

Algoritmo	Num. avaliações	ρ	τ	SE_max	NSGA-II pop.	NSGA-II max.avaliações
M1 25%	100	5	2	25	10	20
NSGA-II	100				10	100
M1 25%	500	25	1	125	25	75
NSGA-II	500				50	500

Os parâmetros de cruzamento e mutação do TDEADP são iguais aos que foram falados anteriormente. Os parâmetros de configuração da Rede Neural do TDEADP são definidos na tabela 2 abaixo:

Tabela 2: Parâmetros TDEADP

Parâmetros	Valor
Depth of FNN(D)	2
No. of Units in Each Hidden Layer (U)	200
Weight Decay Coefficient (λ)	0.00001
Epochs for Initiating FNN (E_init)	20
Batch Size(B)	32
Learning Rate (ϵ)	0.001
Maximum Size for Updating (T_max)	$11 * n + 24$
Accuracy Threshold (γ)	0.9
Maximun Category Size (Q_max)	300

No TDEADP e no DVL foram utilizados o mesmo número de população inicial, com 100 avaliações a população inicial foi de 50 indivíduos enquanto com 500 avaliações foram 100 indivíduos.

4.3 Resultados

4.3.1 Hipervolume

A tabela 3 mostra os dados dos hipervolumes das execuções dos algoritmos na base de dados Breast Cancer. A tabela possui 8 linhas, que mostram os dados de 25 execuções de determinado algoritmo. São 4 colunas que representam o algoritmo, o número de avaliações, a média do hipervolume das 25 execuções e o desvio padrão do hipervolume. Álem disso, tem o boxplot que mostra graficamente a variação dos hipervolumes das execuções do algoritmo, os valores são mostrados no eixo y. E por fim, foi montado um heatmap que apresenta dados do teste estatístico de Wilcoxon, que mostra se existem diferenças entre os algoritmos testados.

Tabela 3: Hipervolume - Breast Cancer

Algoritmo	Num. avaliações	Média Hipervolume	Std. Hipervolume
DVL	100	0.4135	0.011
M1 25%	100	0.3233	0.0729
NSGA-II	100	0.3919	0.0113
TDEADP	100	0.4152	0.0042
DVL	500	0.4255	0.0103
M1 25%	500	0.4257	0.0174
NSGA-II	500	0.4263	0.0087
TDEADP	500	0.4381	0.0094

Observando a tabela é possível notar que todos os algoritmos desempenham melhor com um número de avaliações maior, dessa forma o resultado de todos os algoritmos com 500 avaliações é superior ao mesmo algoritmo com 100 avaliações. Fazendo uma comparação entre todos é possível observar que com 100 iterações o TDEADP e o DVL tiveram resultados bem próximos, porém o TDEADP se saiu melhor contra seus todos os concorrentes. Apesar do resultado de todos os algoritmos serem semelhantes com 500 avaliações, o TDEADP se saiu

um pouco melhor, mesmo com o desvio padrão não sendo o menor dentre os algoritmos.

Na figura 1 temos um gráfico boxplot do hipervolume dos algoritmos, o gráfico mostra no eixo y os valores do hipervolume e no eixo x os algoritmos que foram testados. Por meio deste gráfico é possível perceber que o M1 25% com 100 avaliações teve uma distribuição muito alta, isso pode ter ocorrido devido o baixo número de avaliações que impediu que o algoritmo conseguisse obter um melhor resultado na otimização. O restante dos algoritmos possuem resultados bem parecidos, o DVL e o NSGA-II com 100 avaliações mostraram alguns outliers. O TDEADP 100 foi o algoritmo que obteve o resultado com menor distribuição, e o M1 25% 500 obteve o maior pico entre os resultados, porém ele teve uma distribuição maior que o TDEADP 500.

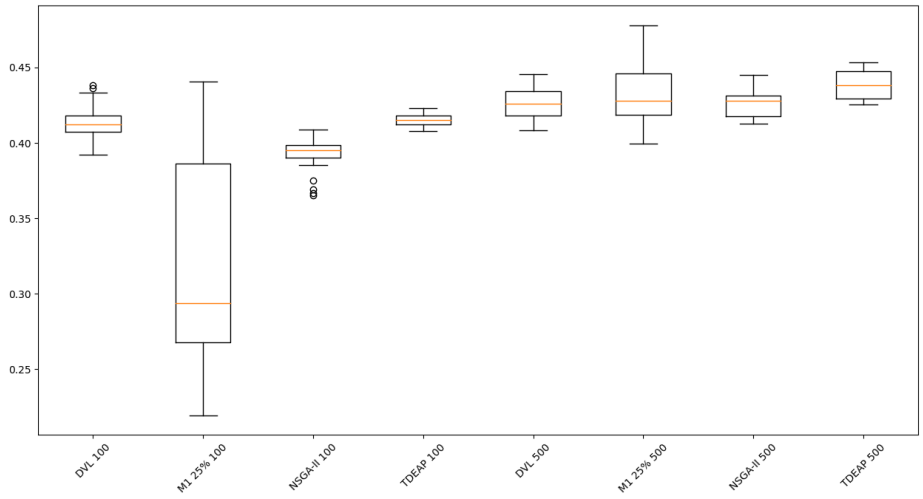

Figura 1: Gráfico boxplot do hipervolume

Na figura 2 temos um heatmap que mostra o resultado do teste estátistico, é possível notar que existe diferença na maior parte dos algoritmos. Os casos onde os algoritmos não possuem diferenças de hipervolume são TDEADP e DVL com 100 avaliações, DVL, NSGA-II e M1 25% com 500 avaliações e também o M1 25%, TDEADP e NSGA-II com 500 avaliações.

4.3.2 Análise de classificação

A tabela 4 contém informações da qualidade de classificação dos algoritmos, nela é mostrada a média da precisão, desvio padrão da precisão, média da AUC e média da F-measure.

Dentre os algoritmos é possível observar que com 100 avaliações o M1 25% obteve o maior resultado, enquanto que com 500 avaliações o DVL se sobressaiu entre os algoritmos. Na AUC os maiores resultados foram para o TDEADP com 100 e 500 avaliações. Já para a F-measure o NSGA-II sem surrogates obteve o maior resultado dentre todos os algoritmos.

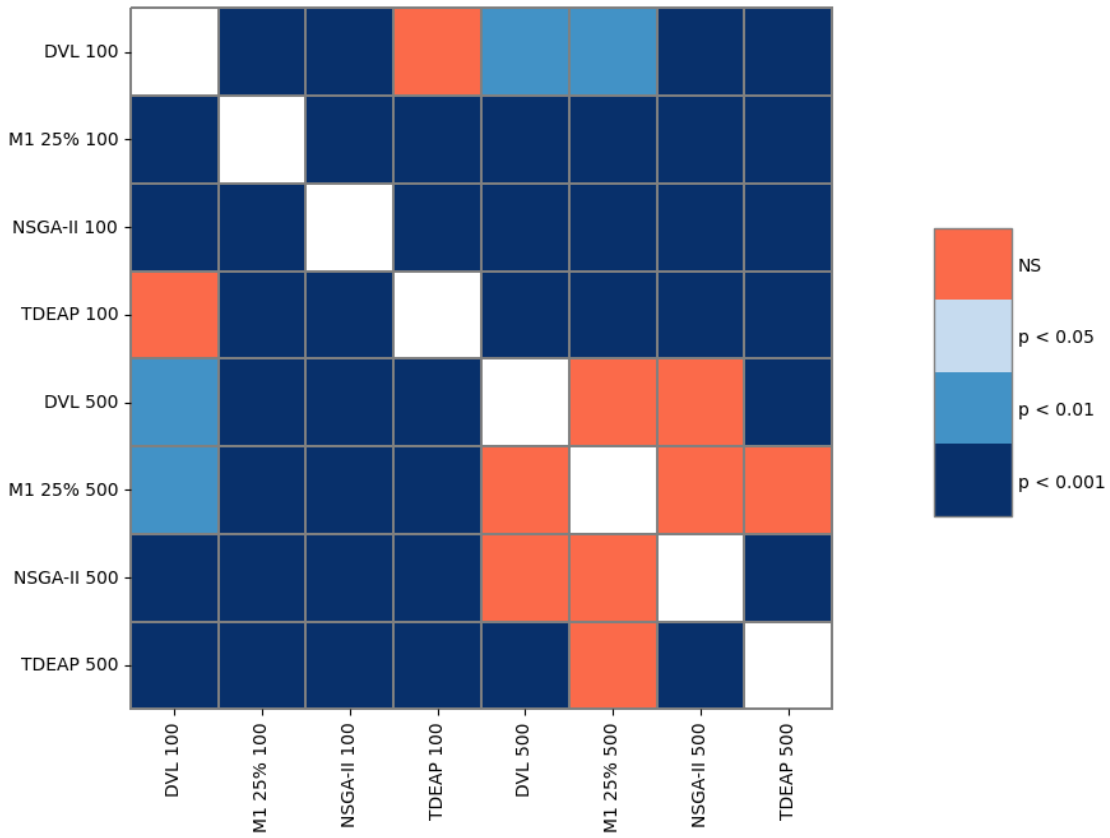

Figura 2: Heatmap do hipervolume

Tabela 4: Classificação - Breast Cancer

Algoritmo	Num. avaliações	Média Precisão	Std. Precisão	Média AUC	Média F-measure
DVL	100	0.734	0.0456	0.6826	0.1438
M1 25%	100	0.7367	0.0355	0.8778	0.1147
NSGA-II	100	0.6654	0.0338	0.8972	0.4527
TDEADP	100	0.6994	0.0473	0.9266	0.3361
DVL	500	0.7433	0.038	0.6182	0.0945
M1 25%	500	0.7379	0.0431	0.752	0.0965
NSGA-II	500	0.6785	0.0259	0.8816	0.461
TDEADP	500	0.7387	0.042	0.9397	0.1112

4.3.3 Tempo

A tabela 5 mostra o tempo médio e o desvio padrão das 25 execuções dos algoritmos. É possível observar que os algoritmos obtiveram um tempo de execução bastante rápido, com exceção do TDEADP. O TDEADP obteve péssimos resultados no tempo de execução, mostrando que a diferença entre sua execução com 100 e 500 avaliações variou cerca de 10 vezes mais. Dessa forma os outros algoritmos possuem técnicas mais rápidas que tornam seu uso mais viável.

Tabela 5: Tempo - Breast Cancer

Algoritmo	Num. avaliações	Média Tempo	Std. Tempo
DVL	100	2.7081	0.2046
M1 25%	100	1.2775	0.1325
NSGA-II	100	2.9537	0.4391
TDEADP	100	1306.7875	181.4085
DVL	500	7.9594	0.7935
M1 25%	500	2.4219	0.1713
NSGA-II	500	11.3625	2.0022
TDEADP	500	10934.6044	1662.9541

5 Conclusões

Neste trabalho foi feita uma análise de algoritmos multiobjetivos para a otimização de hiperparâmetros de SVMs. Para isso foram realizados experimentos com alguns algoritmos específicos, usando técnicas de estatística para comparar os resultados entre si.

É possível concluir que o uso de surrogates, normalmente diminui o tempo de execução e melhora os valores de precisão e hipervolume, apesar de alguns modelos demorarem mais tempo para executar. Essa vantagem do uso de surrogates fica mais evidente com maiores números de avaliações da função objetivo.

6 Perspectivas de futuros trabalhos

Para trabalhos futuros seria interessante adicionar novos algoritmos para a comparação, de preferência algoritmos que utilizem modelos de aprendizagem de natureza diferentes, além de aumentar o número de iterações nos testes para identificar melhor a diferença de eficiência entre os modelos de aprendizagem.

7 Outras atividades

Durante o período de atividades foram feitas leituras de textos e artigos, além de implementações de algoritmos que foram sendo estudados ao longo do período. Foi realizado o curso "Discutindo o Plágio no ambiente acadêmico"feito na SEMAC, e também um curso online sobre aprendizagem de máquina.

Referências

BERGSTRA, J. et al. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, v. 24, 2011. page.33, page.55

BISHOP, C. M. Pattern recognition and machine learning. *Springer google schola*, v. 2, p. 1122–1128, 2006. page.33

COELLO, C. A. C. *Evolutionary algorithms for solving multi-objective problems*. [S.l.]: Springer, 2007. page.33, page.44, page.55

DEB, K. et al. A taxonomy for metamodeling frameworks for evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 23, n. 1, p. 104–116, 2018. page.66, page.77

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, IEEE, v. 6, n. 2, p. 182–197, 2002. page.77

DERRAC, J. et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, Elsevier, v. 1, n. 1, p. 3–18, 2011. page.1010

GUERREIRO, A. P.; FONSECA, C. M.; PAQUETE, L. The hypervolume indicator: Problems and algorithms. *arXiv preprint arXiv:2005.00515*, 2020. page.1010

GUIDO, R.; GROCCIA, M. C.; CONFORTI, D. A hyper-parameter tuning approach for cost-sensitive support vector machine classifiers. *Soft Computing*, Springer, v. 27, n. 18, p. 12863–12881, 2023. page.44

LONES, M. *Sean Luke: essentials of metaheuristics*. [S.l.]: Springer, 2011. page.33

NETO, J. I. C. F. Métodos surrogate aplicados à otimização multiobjetivo de hiperparâmetros de svms. 2023. page.44, page.55

SANTOS, M.; OLIVEIRA, J. A. de; BRITTO, A. Decision variable learning. In: IEEE. *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*. [S.l.], 2019. p. 497–502. page.66

TAKAHASHI, R. H. Otimização escalar e vetorial. *Notas de aula*, 2007. page.33, page.44

WITTEN, I. H. et al. Practical machine learning tools and techniques. In: ELSEVIER AMSTERDAM, THE NETHERLANDS. *Data mining*. [S.l.], 2005. v. 2, n. 4, p. 403–413. page.33

YANG, L.; SHAMI, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, Elsevier, v. 415, p. 295–316, 2020. page.44, page.66

YUAN, Y.; BANZHAF, W. Expensive multiobjective evolutionary optimization assisted by dominance prediction. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 26, n. 1, p. 159–173, 2021. page.66, page.88