Um novo framework para treinamento de Support Vector Machines

Jônatas O. L. da Silva[1], **Paulo S. M. Santos**[2]

[1]DCCMAPI – Universidade Federal do Piauí (UFPI)
Caixa Postal 64.049-550 – Teresina – PI – Brazil

[2]CMRV – Universidade Federal do Piauí
Caixa Postal 64.202-020 – Parnaíba – PI – Brazil

jonatas.iw@gmail.com,paulosms@gmail.com

Abstract. *In this work, a new framework is presented to solve the nonlinear Support Vector Machines (SVM) training problem. The framework is based on the combination of approximations of the objective function by a separable function and the use of one-dimensional methods of low computational cost to solve the resulting sub-problems. Preliminary numerical experiments are reported indicating a promising proposal.*

Resumo. *Neste trabalho, é apresentado um novo framework para resolver o problema de treinamento de Support Vector Machines (SVM) não linear. O framework é baseado na combinação de aproximações da função objetivo por uma função separável e o uso de métodos unidimensionais de baixo custo computacional para resolver os subproblemas resultantes. Experimentos numéricos preliminares são relatados indicando que a proposta é promissora.*

1. Introdução

Entre os diferentes métodos de última geração para resolver problemas de aprendizado de máquina estão as máquinas de vetor de suporte (SVM). Eles combinam excelente desempenho com uma base matemática sólida, como mostrado por [Boser et al. 1992] e depois por [Cortes and Vapnik 1995].

O SVM é um tipo específico de algoritmo de aprendizado de máquina que está entre os mais usados para muitos problemas de aprendizado estatístico, como filtragem de spam, classificação de texto, análise de manuscrito, reconhecimento de rosto e objeto e muitos outros, conforme mostrado em [Cervantes et al. 2020]. [Bhavsar and Panchal 2012] também cita duas características principais do SVM: a teoria da generalização, que leva a uma forma de escolha de uma hipótese baseada em princípios; e funções kernel, que introduzem não linearidade no espaço de hipóteses sem exigir explicitamente um algoritmo não linear.

Algumas implementações atuais usam uma ou mais abordagens para reduzir o tempo de treinamento do SVM: SVMLigth usa a seleção de trabalho e técnicas de redução [Joachims 1999]; O SVMTorch usa *Working selection* e *Shrinking* para melhorar o tempo de treinamento do SVM [Collobert and Bengio 2001]; Pegasos usa métodos de decomposição para reduzir o tempo de treinamento do SVM e é essencialmente um algoritmo de otimização de Subgradiente Estocástico que resolve a formulação primal

[Shalev-Shwartz et al. 2011]; O algoritmo LIBSVM que é baseado no algoritmo *Sequential Minimal Optimization* (SMO), e também usa um algoritmo *work set selection* mais avançado [Chang and Lin 2011]; [Suykens et al. 2002] reformula o problema primal em SVM e propõe a *Least squares support vector machine* (LSSVM); e [Shao et al. 2011] introduziram uma nova forma de classificar os dados, onde ao invés de um único hiperplano, eles propuseram a *Twin support vector machine* (TWSVM) que possui dois hiperplanos não paralelos para um classificador bynário.

Neste trabalho, será definido um framework para resolver o problema de classificação binária para o SVM de forma iterativa, aplicando uma combinação de abordagens. Primeiramente, será definida uma aproximação quadrática separável da função objetivo, permitindo que os subproblemas gerados tenham menor custo computacional. Cada subproblema é abordado em sua forma dual, que pode ser resolvida usando métodos de determinação de zeros de funções de $\mathbb{R}$ em $\mathbb{R}$, veja por exemplo [Cominetti et al. 2014], [Münnich et al. 2012] e suas referências.

Este artigo está organizado da seguinte forma: A Seção 2 apresenta o problema SVM, que é importante para o entendimento do contexto de nossa proposta. Na Seção 3 realizamos um breve estudo sobre os trabalhos relacionados. Na Seção 4 apresentamos a proposta do framework, abordando alguns casos particulares. Os resultados numéricos e as comparações são apresentados na Seção 5, seguidos das conclusões na Seção 6.

2. Support Vector Machines (SVMs)

Segundo [Boser et al. 1992] e [Cortes and Vapnik 1995], SVM são modelos de aprendizado supervisionado usados para classificação e análise de regressão. Ao se referir ao SVM, geralmente não significa SVM linear, mas nos métodos *Kernel* ou SVM não linear. O objetivo de usar um hiperplano ótimo é aumentar a capacidade de generalização da máquina. No entanto, se os dados não forem linearmente separáveis, a máquina não terá uma boa capacidade de generalização, mesmo que o hiperplano ótimo possa ser encontrado. Para resolver esse problema, os dados de entrada são mapeados em um espaço de produto escalar dimensional mais alto, também conhecido como espaço de características (*feature space*) ou espaço de Hilbert proposto por [Mercer 1909]. Tendo essa teoria em mente, os dados ainda são não lineares no espaço de entrada, enquanto um hiperplano pode ser criado no espaço de características para separá-los. A Fig. 2 mostra como o espaço de características pode ser usado para separar dados em uma dimensão mais alta [Christmann and Steinwart 2008].

De acordo com [Mercer 1909], os dados de entrada x são representados por $\phi(x)$ no espaço de características enquanto a forma funcional desse mapeamento é desconhecida. Felizmente, não é essencial mostrar os dados de entrada no espaço de características e apenas o cálculo de seu produto interno é necessário [Christmann and Steinwart 2008]. Este produto interno pode ser calculado selecionando uma função de kernel conforme escrito abaixo:

$$\phi(x_i, x_j) = K(x_i, x_j) \tag{1}$$

Isso torna possível aplicar o SVM para resolver problemas não lineares. A Tabela 1 fornece algumas das funções do kernel que podem ser integradas pelo SVM não linear.

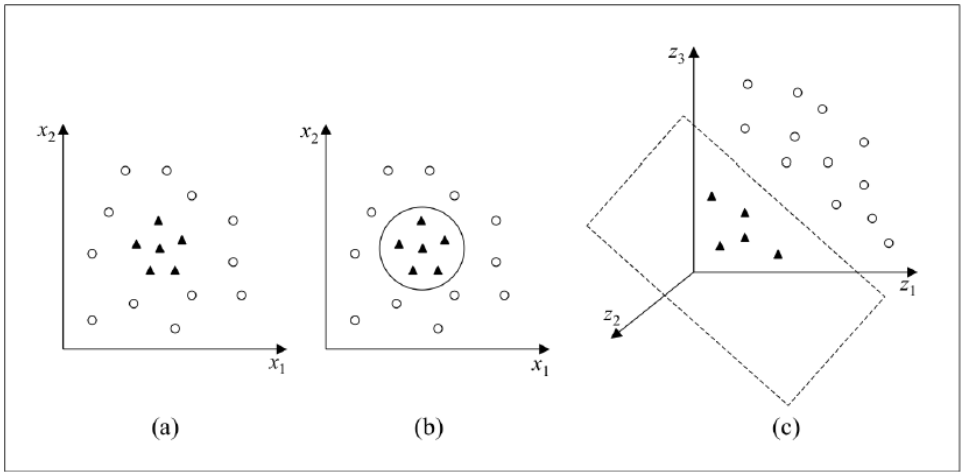


Figura 1. (a) Conjunto de dados não lineares; (b) Fronteira não linear no espaço de entrada; (c) Fronteira linear no espaço de características

Tabela 1. Algumas funções *kernel*

Nome	**Função(x,z)**	**Parâmetros**
Linear	$x^T z$	-
Polinomial	$(x^T z + p)^d$	grau $d \in \mathbb{N},\ p \in \mathbb{R}_+$
Gaussiano	$\exp(\sigma \|\|x - z\|\|^2)$	$\sigma \in \mathbb{R}_+$
Exponencial	$\exp(\sigma x^T z)$	$\sigma \in \mathbb{R}_+$
Sigmoid	$\tanh(k x^T z + \theta)$	k e $\theta < 0$

Tendo considerado a aplicação das funções de Kernel na determinação do produto interno dos dados de entrada trazidos no espaço de características, a formulação dual geral para a classificação de dados não lineares pode ser escrita para um caso de classificação não linear como:

$$\begin{aligned} \min_\alpha \quad & \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \sum_{i=1}^{n} \alpha_i \\ s.a. \quad & \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \end{aligned} \tag{2}$$

onde α são os coeficientes de Langrange, C é uma constante usada para penalizar os erros de treinamento das amostras e $K \in \mathbb{R}^{n \times n}$ é a função Kernel.

Dado um conjunto de exemplos de treinamento $x \in \mathbb{R}^n$ e seus respectivos rótulos $y \in \mathbb{R}^n$, os dados de entrada para o SVM podem ser definido como $\{(x_i, y_i), ..., (x_n, y_n)\}$ com $i = 1, 2, ..., n$ e uma função Kernel K. Cada $y_i = \{-1, +1\}$ é dado como pertencente a uma das duas categorias.

Encontrar o hiperplano ótimo nessas ocasiões, no entanto, não seria uma tarefa fácil e direta devido ao valor desconhecido do vetor de ponderação w que é escrito como

Eq. 3 no espaço de Hilbert.

$$w = \sum_{i=1}^{n} y_i \alpha_i \phi(x_i) \tag{3}$$

Nestas circunstâncias, pode-se usar o truque do kernel de modo que a utilização direta do parâmetro w não seria necessária [Cortes and Vapnik 1995].

A partir da solução α encontrada em (2), podemos construir o classificador de dados binários da seguinte forma:

$$\begin{aligned} D(x) &= sinal\left(\sum_{i=1}^{n} y_i \alpha_i^* K(x, x_i) + b^* \right) \\ &= \begin{cases} \text{classe positiva, se } \sum_{i=1}^{N} y_i \alpha_i^* K(x, x_i) + b^* > 0 \\ \text{classe negativa, se } \sum_{i=1}^{N} y_i \alpha_i^* K(x, x_i) + b^* < 0 \end{cases} \end{aligned} \tag{4}$$

para qualquer $x \in \mathbb{R}^n$. Observe que o classificador também depende do termo b^* e pode ser calculado pela fórmula fechada (5) relativa a cada amostra i quando α_i for diferente de zero ou C. Como é necessário apenas um valor b^*, a formulação apresentada em [Svensén and Bishop 2007] será adotada neste trabalho para calcular b^*,

$$b^* = \frac{1}{|S|} \sum_{j \in S} \left(y_i - \sum_{i,j=1}^{n} \alpha_i^* y_i K(x_i, x_j) \right). \tag{5}$$

onde $S = \{j \in \{1, ..., n\} : 0 < \alpha_j^* < C\}$ é o conjunto de indices, $|S|$ indica quantos elementos o conjunto S possui.

3. Trabalhos relacionados

Muito esforço já foi dedicado à implementação de um método eficiente para resolver o problema SVM. De acordo com [Bottou and Lin 2007] o SVM de [Cortes and Vapnik 1995] é provavelmente o algoritmo de aprendizado de máquina baseado em kernel mais utilizado. Isso se deve ao seu desempenho robusto no reconhecimento de padrões utilizando conceitos já estabelecidos na teoria da otimização. [Bottou and Lin 2007] também diz que o principal motor para a evolução dos *solvers* SVM consiste em encontrar uma maneira de reduzir rapidamente o erro de otimização confortavelmente abaixo dos erros esperados de aproximação e estimativa. Abaixo, algumas abordagens importantes são listadas seguidas de uma breve descrição assim como suas referências.

Os métodos *chunking* são baseados na esparsidade do SVM. O algoritmo começa selecionando um subconjunto arbitrário dos dados chamado *chunk*. O problema de otimização quadrática é resolvido neste pequeno "chunk"e o próximo *chunk* é obtido

com os vetores de suporte resultantes e os pontos que violam as condições de Karush-Kuhn-Tucker (KKT). O processo é interrompido quando todos os dados de treinamento foram considerados e o *chunk* seja composto por todos os vetores de suporte (SV). Este algoritmo reduz a complexidade do SVM reduzindo o problema maior em uma sequência de problemas de otimização menores, determinando iterativamente os vetores de suporte. Alguns exemplos deste uso podem ser encontrados em [Lin et al. 2005], [Chang and Lin 2011], [Chang and Lin 2001] e [Chen et al. 2006].

Resolver o problema SVM consiste em resolver uma otimização de programação quadrática, que pode ser resolvido com alguns solucionadores genéricos de programação quadrática como MINOS, LOQO ou QUADPROG [Bottou and Lin 2007]. No entanto, calcular a matriz de kernel completa é caro e às vezes desnecessário [Bottou and Lin 2007]. Métodos de decomposição foram projetados para mitigar essa dificuldade [Osuna 1998], [Saunders et al. 1998] e [Joachims 1998]. Eles abordam o problema dual completo, resolvendo uma sequência de subproblemas de programação quadrática menores. O método *chunking* é um caso particular do método de decomposição. No entanto, o tamanho dos subproblemas em métodos de decomposição é fixo.

Um importante algoritmo para resolver SVM é o *Sequential Minimal Optimization* (SMO) [Platt 1998], que é obtido a partir da ideia do método de decomposição ao extremo, onde cada subproblema de programação quadrática possui duas variáveis. O poder desta técnica reside no fato de que o novo problema de otimização de dois pontos admite uma solução analítica, eliminando a necessidade de usar um otimizador de programação quadrática iterativa como parte do algoritmo.

[Joachims 1999] também propôs uma técnica chamada *shrinking* que reduz o tamanho do problema eliminando temporariamente as variáveis α_i que provavelmente não serão selecionadas no conjunto de trabalho SMO porque atingiram seu limite inferior ou superior (ou seja, $\alpha_i = 0$ or C).

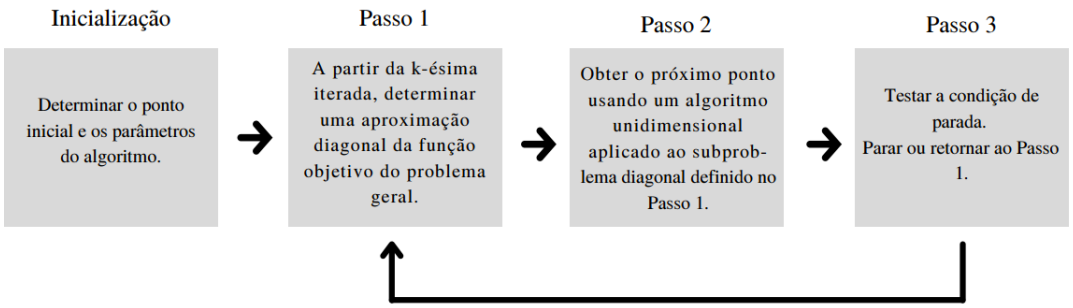
Em abordagens de decomposição, a seleção de um conjunto inicial de variáveis como conjunto de trabalho (*working set*) é importante para que a iteração atual se mova em direção ao mínimo. Por exemplo, há muitas maneiras de selecionar o par de índices (i, j) que representa o conjunto de trabalho para cada iteração do algoritmo SMO. [Lin et al. 2005], [Glasmachers et al. 2006] e [Venkateshan et al. 2014] definiram como os esquemas práticos de seleção de conjuntos de trabalho que podem alcançar um bom compromisso entre o número de iterações e a velocidade de cada iteração.

Mais sobre formulações, técnicas, abordagens, aplicações e tendências de SVM podem ser encontradas em [Cervantes et al. 2020], [Bottou and Lin 2007] e [Tian et al. 2012].

4. Proposta de framework

A proposta esquematizada na Figura 2, é motivada pela necessidade de um procedimento numérico flexível cujos subproblemas sejam de baixo custo computacional. A flexibilidade se caracteriza pelas possibilidades de aproximações separáveis no Passo 1 e pelos muitos algoritmos unidimensionais existentes que podem ser usados no Passo 2. Os algoritmos aqui estudados para a implementação do Passo 2 tem como características a simplicidade e seu baixo custo computacional.

Figura 2. Fluxograma do Framework proposto



No seguinte, descreveremos com mais detalhes as características de nosso esquema.

4.1. Aproximações quadráticas separáveis

Nosso estudo de aproximações da função objetivo do problema de SVM começou com os trabalhos de [Snyman and Hay 2001], para resolver problemas de otimização irrestritos. A abordagem é considerada uma alternativa para os métodos baseados em gradiente conjugado por requerer menos armazenamento de variáveis e por sua eficiência computacional. O método aplica a técnica de maior declive em sucessivas aproximações quadráticas esféricas da função objetivo de tal forma que nenhuma busca linear seja necessária na resolução do problema de minimização. [Snyman and Hay 2001] mostram ainda que o método é convergente quando aplicado a funções quadráticas positivas-definidas gerais e se sai bem com problemas mal condicionados. Mais precisamente, eles consideraram o seguinte problema de otimização irrestrita.

$$\text{minimizar } f(x), x \in \mathbb{R}^n \tag{6}$$

onde $f\colon \mathbb{R}^n \to \mathbb{R}$ é a função objetivo.

Assumindo a diferenciabilidade de f, eles definem uma aproximação diagonal de f no ponto x^k, denominada $\bar{f}_k(x)$, por:

$$\bar{f}_k(x) = \frac{1}{2}(x - x^k)^T Z_k (x - x^k) + \nabla f(x^k)^T (x - x^k) + f(x^k), \tag{7}$$

e $Z_k = diag(z_k, z_k, ..., z_k) = z_k I$. Forçando $\bar{f}_k(x^{k-1}) = f(x^{k-1})$, é possível obter z_k da seguinte forma:

$$z_k = \frac{2[f(x^{k-1}) - f(x^k) - \nabla f(x^k)^T (x^{k-1} - x^k)]}{\|x^{k-1} - x^k\|^2}. \tag{8}$$

A função em (7) será sempre separável e convexa, desde que os termos z_k sejam estritamente positivos, possuindo assim uma matriz hessiana definida positiva e diagonal.

A aproximação da Hessiana da função objetivo em (7) é um múltiplo da matriz identidade. Entretanto, existem outras formas de aproximar a hessiana de f com matrizes

diagonais. Nas aproximações diagonais Quasi-Newton, seguindo [Andrei 2019], uma aproximação diagonal da Hessiana da função objetivo pode ser definida por:

$$B_0 \in \mathbb{R}^{n\times n}, \;\; B_{k+1} = B_k + \frac{s_k^T y_k + s_k^T s_k - s_k^T B_k s_k}{tr(A_k^2)} A_k - I,$$

onde $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ e $A_k = \text{diag}\,((s_k^1)^2, \ldots, (s_k^n)^2)$.

Em [Zhu et al. 1999], é apresentado ainda a aproximação Quasi-Cauchy, no qual é relacionada com a aproximação Quasi-Newton "fraca"de [Dennis and Wolkowicz 1993] onde uma restrição adicional substitui a matriz cheia por uma matriz diagonal. A aproximação Quasi-Cauchy pode ser obtida através da seguinte expressão:

$$D_+ = (s^T y / s^T s) I. \tag{9}$$

onde $s = x_+ - x$, com x_+ e x sendo dois pontos diferentes, $y = \nabla f(x_+) - \nabla f(x)$, I é uma matriz identidade. Por construção, D_+ é uma matriz diagonal.

4.2. Métodos unidimensionais

Os métodos unidimensionais mencionados nesse trabalho são aqueles cujo a finalidade é resolver o Problema da Mochila Quadrático Separável (PMQS), que por sua vez possui a seguinte forma:

$$(PMQS): \begin{cases} \text{Minimizar} \quad f(x) := \displaystyle\sum_{i=1}^{n} \left(\frac{1}{2} p_i x_i^2 - a_i x_i \right), \\ \text{sujeito a} \\ \displaystyle\sum_{i=1}^{n} b_i x_i = c, \\ l_i \leq x_i \leq u_i, \;\; i = 1, 2, \ldots, n, \end{cases} \tag{10}$$

onde $p_i, b_i > 0$ e $l_i < u_i$ para todo $i = 1, 2, \ldots, n$ com $c > 0$ tal que $\langle b, l\rangle \leq c \leq \langle b, u\rangle$.

Dado o problema (10), reformulando x como uma função dependente de λ, temos um problema de encontrar o multiplicador de Lagrange da restrição de igualdade. A prova dos três próximos resultados teóricos pode ser encontrada em [Münnich et al. 2012].

Teorema 4.1 *Um vetor $x^* \in \mathbb{R}^n$ é um mínimo do Problema (10), se e somete se, nele existe um multiplicador de Lagrange $\lambda_* \in \mathbb{R}$, $v^* \in \mathbb{R}^n_+$ e $w^* \in \mathbb{R}^n_+$ tal que*

$$L'_x(x^*, \lambda_*, v^*, w^*) = \nabla f(x^*) + \lambda_* \nabla g(x^*) + (\nabla r(x^*))' v^* + (\nabla s(x^*))' w^* = 0 \tag{11}$$

e além disso,

$$v_i r_i(x_i^*) = 0, \;\; i = 1, \cdots, n, \tag{12}$$

$$w_i s_i(x_i^*) = 0, \;\; i = 1, \cdots, n, \tag{13}$$

onde $g(x) = b^T x - c$, $r(x) = l - x$ e $s(x) = x - u$.

Lema 4.1 *De acordo com as condições do Teorema 4.1, (11) é equivalente a*

$$\left.\begin{array}{rl} 0 \geq \dfrac{p_i u_i - a_i}{b_i} + \lambda_* & , \textit{ se } x_i^* = u_i \\ 0 = \dfrac{p_i x_i^* - a_i}{b_i} + \lambda_* & , \textit{ se } x_i^* \in (l_i, u_i) \\ 0 \leq \dfrac{p_i l_i - a_i}{b_i} + \lambda_* & , \textit{ se } x_i^* = l_i, \end{array}\right\} \tag{14}$$

para todo $i = 1, 2, \ldots, n$.

Motivado por (14), define-se a função x dependendo da variável λ.

$$x : \mathbb{R} \to \mathbb{R}^n,\ \lambda \mapsto x(\lambda)$$

com

$$x_i(\lambda) = \begin{cases} l_i & , \text{ se } \lambda \geq \dfrac{a_i - p_i l_i}{b_i} \\ \dfrac{a_i - \lambda b_i}{p_i} & , \text{ se } \dfrac{a_i - p_i u_i}{b_i} < \lambda < \dfrac{a_i - p_i l_i}{b_i} \\ u_i & , \text{ se } \lambda \leq \dfrac{a_i - p_i u_i}{b_i}. \end{cases} \tag{15}$$

Teorema 4.2 *Um vetor* $x^* \in \mathbb{R}^n$ *é a solução única do problema de otimização (10), se e some se, existe* $\lambda_* \in \mathbb{R}$ *tal que* $x(\lambda_*)$ *definido em (15) satisfaz*

$$g(x(\lambda_*)) = 0. \tag{16}$$

Do desenvolvimento acima, observe que para resolver o PMQS, podemos usar métodos que encontrem zeros de funções $g\colon \mathbb{R} \to \mathbb{R}$.

A seguir uma estratégia de ponto fixo é descrita, para resolver o problema (16).

4.2.1. Formulação Método do Ponto Fixo

Inicialmente, usando (15), são definidos os seguintes conjuntos de índices

$$I_\ell := \left\{ i \in \{1, 2, \ldots, n\} \colon \lambda \geq \frac{a_i - p_i l_i}{b_i} \right\},$$

$$I_u := \left\{ i \in \{1, 2, \ldots, n\} \colon \lambda \leq \frac{a_i - p_i u_i}{b_i} \right\}$$

$$I_{eq} := \{1, 2, \ldots, n\} \setminus (I_\ell \cup I_u).$$

Assim,

$$\begin{aligned} g(\lambda) &= \sum_{i\in I_\ell} b_i l_i + \sum_{i\in I_u} b_i u_i + \sum_{i\in I_{eq}} b_i \bar{x}_i - c \\ &= \sum_{i\in I_\ell} b_i l_i + \sum_{i\in I_u} b_i u_i + \sum_{i\in I_{eq}} b_i \frac{a_i - \lambda b_i}{p_i} - c \\ &= \sum_{i\in I_\ell} b_i l_i + \sum_{i\in I_u} b_i u_i + \sum_{i\in I_{eq}} \left(\frac{b_i a_i}{p_i} - \frac{\lambda b_i^2}{p_i} \right) - c \\ &= \sum_{i\in I_\ell} b_i l_i + \sum_{i\in I_u} b_i u_i + \sum_{i\in I_{eq}} \frac{b_i a_i}{p_i} - \lambda \sum_{i\in I_{eq}} \frac{b_i^2}{p_i} - c. \end{aligned}$$

Então, $g(\lambda) = 0$, se e somente se,

$$\lambda \sum_{i\in I_{eq}} \frac{b_i^2}{p_i} = \sum_{i\in I_\ell} b_i l_i + \sum_{i\in I_u} b_i u_i + \sum_{i\in I_{eq}} \frac{b_i a_i}{p_i} - c$$

Em casos onde $I_{eq} \neq \emptyset$, tem-se

$$\lambda = \frac{\sum_{i\in I_\ell} b_i l_i + \sum_{i\in I_u} b_i u_i + \sum_{i\in I_{eq}} \frac{b_i a_i}{p_i} - c}{\sum_{i\in I_{eq}} \frac{b_i^2}{p_i}} \tag{17}$$

Por outro lado, se $I_{eq} = \emptyset$, obtém-se

$$0 = \sum_{i\in I_\ell} b_i l_i + \sum_{i\in I_u} b_i u_i - c. \tag{18}$$

Baseado em [Cominetti et al. 2014], [Kiwiel 2008] e [Münnich et al. 2012], a resolução do problema pode começar assumindo que a restrição da caixa não existe, então temos

$$p_i x_i^* - a_i + \lambda_* b_i = 0 \quad \Leftrightarrow \quad p_i x_i^* b_i - a_i b_i + \lambda_* b_i^2 = 0, \quad i = 1, \ldots, n$$

e se $p_i \neq 0$

$$x_i^* b_i - \frac{a_i b_i}{p_i} + \lambda_* \frac{b_i^2}{p_i} = 0, \quad i = 1, \ldots, n$$

Somando e usando isso $b^T x^* = c$, obtém-se

$$\sum x_i^* b_i - \sum \frac{a_i b_i}{p_i} + \lambda_* \sum \frac{b_i^2}{p_i} = 0 \quad \Leftrightarrow \quad c - \sum \frac{a_i b_i}{p_i} + \lambda_* \sum \frac{b_i^2}{p_i} = 0, \quad i = 1, \ldots, n.$$

Então, conclui-se que

$$\lambda_* = \frac{\sum \frac{a_i b_i}{p_i} - c}{\sum \frac{b_i^2}{p_i}}, \quad i = 1, \ldots, n.$$

Com isso, pode-se considerar

$$\lambda_0 = \frac{\sum \frac{a_i b_i}{p_i} - c}{\sum \frac{b_i^2}{p_i}}, \quad i = 1, \ldots, n.$$

O Método do Ponto Fixo pode portanto ser formulado na forma do Algoritmo 1:

No seguinte, são relacionados alguns trabalhos com possibilidades de estratégias unidimensionais para a implementação do Passo 2 do Framework proposto.

Algoritmo 1 Método do Ponto Fixo (MPF)

· **Inicialização** Definir $\lambda_0 \in \mathbb{R}$

· **Passo 1** Obter $I_\ell^k, I_u^k, I_{eq}^k$

· **Passo 2** se $I_{eq}^k \neq \emptyset$ então

$$\lambda_{k+1} = \frac{\sum_{i \in I_\ell^k} b_i l_i + \sum_{i \in I_u^k} b_i u_i + \sum_{i \in I_{eq}^k} \frac{b_i a_i}{p_i} - r}{\sum_{i \in I_{eq}^k} \frac{b_i^2}{p_i}}$$

senão

$$\lambda_{k+1} = \lambda_k + \sum_{i \in I_\ell^k} b_i l_i + \sum_{i \in I_u^k} b_i u_i - c$$

· **Passo 3** Se $\lambda_{k+1} = \lambda_k$, PARAR. Caso contrário, $k = k + 1$ e voltar ao Passo 1.

1. Em [Münnich et al. 2012], uma estratégia de ponto fixo para um problema amostragem estratificada (não quadrático mas separável) é comparada com métodos clássicos da literatura: Bissecção, Regula falsi e Secantes.
2. Em [Cominetti et al. 2014], é estudado um método do tipo-Newton para o PMQS e são incluídas comparaçoes com métodos de Secantes, Busca da Mediana e Fixação de variáveis.

4.3. Primeiras aplicações do framework ao problema de SVM

O problema de otimização gerado pelo método de SVM (2) a ser resolvido envolve uma função objetivo quadrática sujeito a uma restrição de igualdade e restrições de caixa, podendo ser reescrito da seguinte forma:

$$\begin{array}{ll} \text{minimizar}_\alpha & \frac{1}{2}\alpha^T P \alpha - \alpha^T e \\ \text{sujeito a} & y^T \alpha = 0 \\ & 0 \leq \alpha \leq Ce, \end{array} \tag{19}$$

onde $\alpha \in \mathbb{R}^n$, $C \in \mathbb{R}$, $y \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$, é o vetor de respostas, $e = (1, 1, \ldots, 1) \in \mathbb{R}^n$ e a matriz $P \in \mathbb{R}^{n \times n}$ é simétrica com componentes definidas como

$$P_{i,j} = y_i y_j K(x_i, x_j), \quad i, k = 1, 2, ..., n$$

para $x \in \mathbb{R}^n$ o vetor de amostras com os p atributos do problema e K a função *Kernel*.

A aproximação da função objetivo é definida como

$$\bar{f}_k(\alpha) = \frac{1}{2}(\alpha - \alpha^k)^T Z_k (\alpha - \alpha^k) + \nabla f(\alpha^k)^T (\alpha - \alpha^k) + f(\alpha^k)$$

onde $Z_k = diag(z_k, z_k, ..., z_k) = z_k I$ pode ser obtido com aproximação diagonal esférica pela fórmula

$$z_k = \frac{2[f(\alpha^{k-1}) - f(\alpha^k) - \nabla f(\alpha^k)^T (\alpha^{k-1} - \alpha^k)]}{||\alpha^{k-1} - \alpha^k||^2},$$

ou com aproximação Quasi-Cauchy pela fórmula

$$z_k = \frac{(\alpha^k - \alpha^{k-1})^T (\nabla f(\alpha^k) - \nabla f(\alpha^{k-1}))}{(\alpha^k - \alpha^{k-1})^T (\alpha^k - \alpha^{k-1})}.$$

Uma vez encontrada a função $\bar{f}_k(\alpha)$, ainda será necessário uma manipulação algébrica para que a mesma corresponda ao formato da função objetivo $f(\alpha)$ que estamos querendo resolver. Tal manipulação pode ser realizada da seguinte maneira:

$$\begin{aligned}
\bar{f}_k(\alpha) &= \tfrac{1}{2}(\alpha-\alpha^k)^T Z_k(\alpha-\alpha^k) + \nabla f(\alpha^k)^T(\alpha-\alpha^k) + f(\alpha^k) \\
&= \nabla f(\alpha^k)^T\alpha - \nabla f(\alpha^k)^T\alpha^k + \tfrac{1}{2}\left[(\alpha-\alpha^k)^T Z_k\alpha - (\alpha-\alpha^k)^T Z_k\alpha^k\right] \\
&= \nabla f(\alpha^k)^T\alpha - \nabla f(\alpha^k)^T\alpha^k + \tfrac{1}{2}\left[\alpha^T Z_k\alpha - \alpha^{k^T} Z_k\alpha - \alpha^T Z_k\alpha^k - \alpha^{k^T} Z_k\alpha^k\right] \\
&= \nabla f(\alpha^k)^T\alpha + \tfrac{1}{2}\alpha^T Z_k\alpha - \tfrac{1}{2}\alpha^{k^T} Z_k\alpha - \tfrac{1}{2}\alpha^T Z_k\alpha^k.
\end{aligned} \tag{20}$$

Dado (20), a nova função $\tilde{f}_k(\alpha)$ que queremos minimizar em cada subproblema será definida como

$$\tilde{f}_k(\alpha) = \frac{1}{2}\alpha^T Z_k\alpha - \left[-\nabla f(\alpha^k) + Z_k\alpha^k\right]^T\alpha \tag{21}$$

Com isso, o framework proposto pode ser definido na forma do Algoritmo 2.

Algoritmo 2 Framework proposto para resolução do Problema (19)

· **Dados** $X \in \mathbb{R}^{n\times p}$ e $y \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$;
· **Parâmetros** $\rho > 0$, $C > 0$, função *Kernel* K;
· **Inicialização** Definir $\alpha^0, \alpha^1, tol_\epsilon, k = 1$
· **Passo 1** A partir da k-ésima iterada, determinar uma aproximação diagonal da função objetivo do Problema (19);
· **Passo 2** Obter o próximo ponto α^k usando um algoritmo unidimensional aplicado ao subproblema diagonal obtido no Passo 1;
· **Passo 1** Se $||\alpha^k - \alpha^{k-1}|| < tol_\epsilon$ então $\alpha^* = \alpha^{k-1}$ e PARAR; caso contrário retornar ao Passo 1;

No Algoritmo 2, foram definidos alguns parâmetros para mitigar a dificuldade que o método de aproximação diagonal esférica encontra quando a restrição superior da caixa possui valores elevados, ou seja, quando $C = 100$ e $C = 1000$, sendo C um parâmetro do problema de SVM. Para tal, o seguinte procedimento foi adotado de acordo com [Groenwold et al. 2010]. Dada a restrição de caixa

$$l_i \leq x_i \leq u_i \quad i = 1, 2, ..., n,$$

é implicitamente entendido que os limites inferiores e superiores podem ser substituídos por alguma sub-região de busca [Groenwold et al. 2010], ficando

$$l_i \leq \check{\delta}_i^k \leq x_i \leq \hat{\delta}_i^k \leq u_i \quad i = 1, 2, ..., n. \tag{22}$$

Nas desigualdades (22), $\hat{\delta}_i^k$ e $\check{\delta}_i^k$ serão atualizados a cada iteração da seguinte maneira:

$$\begin{aligned}
\check{\delta}_i^k &= \max(l_i, x_i^k - \rho(u_i - l_i)), \\
\hat{\delta}_i^k &= \min(u_i, x_i^k + \rho(u_i - l_i)),
\end{aligned} \tag{23}$$

onde $\rho > 0$ e $\rho \in \mathbb{R}$.

5. Experimentos numéricos

Para ilustrar o comportamento numérico do framework proposto foram realizados alguns testes numéricos usando um notebook Intel Core i5 1,6 GHz, 8 GB de RAM, processador dual-core, sistema operacional IOs com código implementado em MATLAB 2021b.

Foram realizadas comparações com o solver LIBSVM versão 3.25 [Chang and Lin 2011], considerado como estado da arte para treinamento de SVM [Chauhan et al. 2019]. O código do solver mencionado foi retirado do site oficial de seus criadores para uso no MATLAB.

Os principais fatores analisados nestes experimentos foram a precisão e o tempo computacional para treinar o classificador. A precisão do classificador é determinada pela matriz de confusão gerada pela classificação. Em uma classificação binária, esta matriz terá elementos classificados corretamente e incorretamente que podem ser definidos de acordo com a Tabela 2.

Tabela 2. Exemplo matriz de confusão

		Valor real	
		Verdadeiro	Falso
Valor previsto	Verdadeiro	Verdadeiro Positivo (VP)	Falso Positivo (FP)
	Falso	Falso Negativo (FN)	Verdadeiro Negativo (VN)

Assim, a precisão pode ser calculada da seguinte forma:

$$\text{Precisão} = \frac{TP + TN}{TP + TN + FP + FN}$$

Os experimentos numéricos estão organizados em seções da seguinte forma: Na Subseção 5.1 descrevemos sobre os conjuntos de dados selecionados e na Subseção 5.2 reportamos os resultados dos testes realizados.

5.1. Conjunto de dados

Os conjuntos de dados foram extraídos do site oficial da LIBSVM. Os autores do LIBSVM já disponibilizam os dados no formato de leitura e o pacote de instalação do LIBSVM oferece uma função para transformar os dados em matrizes e vetores com dimensões relativas as n amostras e p atributos.

Para conjuntos de dados com o termo *NAME.scale*, significa que seus atributos foram dimensionados linearmente para os intervalos [-1,1] ou [0,1]. A principal vantagem do dimensionamento é evitar que os intervalos de valores numéricos de alguns atributos sejam muito maiores do que o intervalo de valores de outros atributos. Outra vantagem é evitar dificuldades numéricas durante o cálculo ao usar a função *Kernel* [Hsu et al. 2003]. Para esses experimentos, apenas dados escalados foram usados. Neste caso, a transformação de um atributo r de qualquer conjunto de dado pode ser descrita dentro do intervalo $[a, b]$ na forma

$$r_{escalado} = (a - b)\frac{r - \min(r)}{\max(r) - \min(r)} + a.$$

A Tabela 3 descreve os dados selecionados, ilustrando o número de amostras que o conjunto de dados possui (n) e o número de atributos que são considerados (p). Nos experimentos realizados, será aplicada a técnica *Cross-Validation* (CV) para obter um conjunto de teste disjunto. A técnica consiste em separar uma porcentagem do conjunto de dados original - neste trabalho foi utilizado 70% - para um conjunto de treinamento do modelo, enquanto o complemento é destinado à fase de teste.

Tabela 3. Conjunto de dados utilizados

Nome	**n**	**p**
heart.scale	270	13
australian.scale	690	14
fourclass.scale	862	2
ionosphere.scale	351	34
sonar.scale	208	60
splice.scale	1000	60

Mais detalhes sobre os conjuntos de testes apresentados podem ser encontrados em [Chang and Lin 2011].

5.2. Resultados

Como parte dos experimentos, as seguintes combinações foram utilizadas no framework: o primeiro o qual denominamos Esf-MPF e Esf-MFV, por ser uma combinação entre a aproximação diagonal quadrática esférica no Passo 1 e o Método do Ponto Fixo apresentado na Seção 4.2 e o Método de Fixação Variável [Kiwiel 2008] no Passo 2, respectivamente; a segunda chamamos de Cauchy-MPF e Cauchy-MFV, por ser uma conbinação entre aproximação Quasi-Cauchy no Passo 1 e o Método do Ponto Fixo e Método de Fixação Variável no Passo 2, repectivamente. Tais combinações foram desenvolvidas com o intúito de exemplicar a capacidade do framework em possibilitar a combinação entre técnicas de aproximação diagonal no Passo 1 e qualquer outro algoritmo da literatura capaz de resolver o Problema (10) no Passo 2.

Sobre os dados, como foi aplicado CV, 50 sementes foram selecionadas aleatoriamente para gerar diferentes subconjuntos de dados de treinamento e teste. Assim, os resultados apresentados serão valores médios dos 50 diferentes problemas gerados.

Nestes experimentos, será variado o valor do parâmetro C do Problema (19), que corresponde ao limite superior da caixa, em 1, 100 e 1000. Para os testes iniciais escolhemos a função de Kernel gaussiana e linear, cuja fórmula está descrita na Tabela 1. O parâmetro σ para função Kernel gaussiana foi definido como $\frac{1}{p}$, onde p é o número de atributos que a amostra apresenta de acordo com a Tabela 3. Essa escolha foi adotada porque o LIBSVM também utiliza o mesmo calculo para obter o valor deste parâmetro em seu modo *default*.

Na Tabela 4 apresentamos os resultados de tempo computacional e acurácia utilizando aproximação esférica com função Kernel Gaussiano. Os resultados apontam que a proposta obteve resultados de acurácia semelhantes a LIBSVM para as bases heart.scale, ionosfere.scale, sonar.scale e splice.scale para todos os valores a variações do parâmetro C. Já para as bases australian.scale e fourclass.scale, a proposta obteve acurácia inferior ao LIBVSM quando o valor do parâmetro $C = 100$ e $C = 1000$. Com relação ao

Tabela 4. Experimentos utilizando Aproximação Esférica com função Kernel Gaussiana

Nome	C	Tempo			Acurácia		
		Libsvm	Esf-MPF	Esf-MFV	Libsvm	Esf-MFV	Esf-MPF
heart scale	1	0.002	0.001	0.001	0.815	0.816	0.816
	100	0.002	0.006	0.009	0.787	0.786	0.786
	1000	0.002	0.013	0.019	0.768	0.747	0.747
australian scale	1	0.007	0.003	0.003	0.858	0.858	0.858
	100	0.010	0.022	0.028	0.849	0.830	0.830
	1000	0.019	0.053	0.067	0.815	0.723	0.720
fourclass scale	1	0.007	0.004	0.005	0.798	0.803	0.800
	100	0.010	0.040	0.045	0.969	0.832	0.803
	1000	0.015	0.088	0.101	0.997	0.753	0.735
ionosphere scale	1	0.003	0.001	0.001	0.916	0.923	0.923
	100	0.002	0.007	0.010	0.930	0.927	0.926
	1000	0.003	0.016	0.022	0.900	0.879	0.878
sonar scale	1	0.003	0.000	0.001	0.760	0.785	0.784
	100	0.003	0.004	0.006	0.851	0.846	0.847
	1000	0.003	0.008	0.008	0.842	0.845	0.845
splice scale	1	0.010	0.001	0.001	0.795	0.795	0.795
	100	0.012	0.007	0.007	0.802	0.804	0.804
	1000	0.011	0.011	0.011	0.792	0.795	0.795

tempo computacional, a Tabela 4 mostra que quando o parâmetro $C = 1$, o tempo de processamento do algoritmo proposto é superior ao LIBSVM em todos as bases de testes, mas a medida que o valor de C aumenta, a proposta consome mais tempo para resolver o problema. O resultado desse comportamento se dá pela quantidade de iterações que o framework precisa para resolver o problema quando o valor do parâmetro C é igual a 100 e 1000. Uma exceção acontece com a base de dados splice.scale, onde a proposta se mantém superior para todos os valores do parâmetro C.

O método adotado para analisar e comparar o desempenho dos algoritmos desenvolvidos neste trabalho foi desenvolvido por [Dolan and Moré 2002]. Os autores o criaram com o objetivo de facilitar a visualização e interpretação dos resultados obtidos em experimentos, comparando um conjunto de algoritmos a fim de identificar aquele com melhor desempenho aplicado a um conjunto de problemas. O método considera um conjunto P de problemas de teste p_j, com $j = 1, 2, .., n_p$, um conjunto de algoritmos a_i, com $i = 1, 2, .., n_a$ e uma métrica de desempenho $t_{p,a}$ (tempo de computação, média dos valores da função objetivo, etc.).

A taxa de desempenho (sempre maior ou igual a 1) é definida como:

$$r_{p,a} = \frac{t_{p,a}}{\min(t_{p,a} : a \in A)} \tag{24}$$

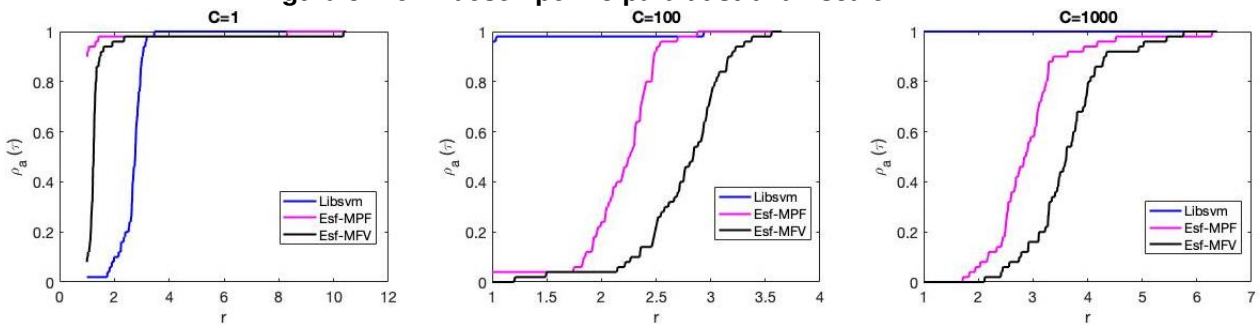
O perfil de desempenho do algoritmo é dado por:

$$\rho_a(\tau) = \frac{1}{n_p}|p \in P : r_{p,a} \leq \tau| \tag{25}$$

onde $\rho_a(\tau)$, é a fração de problemas resolvidos pelo algoritmo com desempenho dentro de um fator do melhor desempenho obtido, considerando todos os algoritmos.

Nas Figuras 3 e 4 é exibido o gráfico resultante do método perfil de desempenho [Dolan and Moré 2002] com relação ao tempo computacional para a base de dados *australian.scale* e $splice.scale$ respectivamente. Os resultados ilustrados nas Figuras 3 e 4 são referentes aos resultados obtidos na Tabela 4.

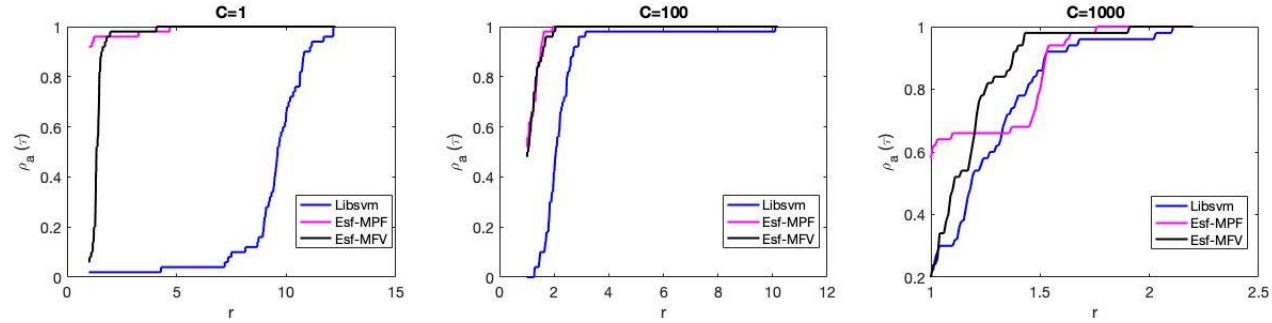
Figura 3. Perfil desempenho para *australian.scale*



Na Figura 3 podemos observar uma mudança no comportamento entre o framework proposto e LIBSVM, onde para $C = 1$ os algoritmos Esf-MPF e Esf-MFV são superiores ao LIBSVM, mas a medida que o valor do parâmetro C aumenta, o algoritmo LIBSVM se torna superior. Como já mensionado, esse comportamento existe por conta de uma alta sensibilidade do método de aproximação diagonal com relação a diferença de escala entre valores na função objetivo e restrições. Tal diferença de escala se torna notória quando o parâmetro C aumenta, por essa razão foi adicionado o parâmetro ρ no Algoritmo 2.

Ainda na Figura 3 observamos que todos os algoritmos resolvem todos os problemas, devido alcançarem o valor 1 no eixo $\rho_a(\tau)$, onde para $C = 1$ o algoritmo Esf-MPF é mais veloz que os outros algoritmos em todos os 50 testes; quando $C = 100$, o LIBSVM se torna superior seguido de Esf-FPM e Esf-MFV; já quando $C = 1000$ LIBSVM continua sendo superior em todos os 50 testes seguido de Esf-MPF e Esf-MFV.

Figura 4. Perfil desempenho para *splice.scale*



Já na Figura 4, a medida que o parâmetro C aumenta, o algoritmo da proposta se

Tabela 5. Experimentos utilizando Aproximação Esférica com função Kernel linear

Nome	C	Tempo			Acurácia		
		Libsvm	Esf-MPF	Esf-MFV	Libsvm	Esf-MFV	Esf-MPF
heart scale	1	0.002	0.001	0.001	0.830	0.819	0.819
	100	0.035	0.007	0.009	0.828	0.822	0.819
	1000	0.376	0.013	0.017	0.838	0.825	0.823
australian scale	1	0.011	0.002	0.003	0.854	0.851	0.851
	100	0.177	0.024	0.027	0.860	0.838	0.838
	1000	4.496	0.052	0.065	0.851	0.687	0.703
fourclass scale	1	0.005	0.005	0.005	0.765	0.743	0.743
	100	0.020	0.040	0.046	0.769	0.739	0.739
	1000	0.097	0.094	0.113	0.769	0.731	0.720
ionosphere scale	1	0.003	0.001	0.001	0.876	0.849	0.843
	100	0.038	0.007	0.011	0.860	0.810	0.817
	1000	0.275	0.017	0.025	0.863	0.838	0.844
sonar scale	1	0.002	0.001	0.001	0.757	0.721	0.719
	100	0.007	0.004	0.005	0.755	0.725	0.726
	1000	0.008	0.008	0.012	0.742	0.700	0.695
splice scale	1	0.016	0.001	0.001	0.773	0.742	0.744
	100	0.967	0.011	0.014	0.748	0.737	0.733
	1000	5.808	0.025	0.031	0.749	0.737	0.733

mantém superior a LIBSVM. Novamente todos os algoritmos resolvem todos os problemas. Para $C = 1$ os algoritmos Esf-MPF e Esf-MFV são superiores ao LIBSVM. Quando $C = 100$ o algoritmo Esf-MPF obteve melhor desempenho em aproximadamente 50% dos casos e Esf-MFV em 50% dos casos. Quando $C = 1000$ o algoritmo Esf-MPF obteve melhor desempenho em aproximadamente 60% dos casos enquanto Esf-FV e LIBSVM em aproximadamente 20% dos casos.

Na Tabela 5 apresentamos o tempo computacional e acurácia utilizando aproximação esférica com função Kernel linear. Os resultamos mostram que o tempo computacional para todas a bases foram superiores para todos os valores do parâmetro C (1, 100 e 1000). Com relação a acurácia, a Tabela 5 mostra que a proposta me mantém competiva com a relação a LIBSVM por apresentar valores semelhantes em todas as bases de dados, com excessão da base splice.scale e australian.scale.

Os parâmetros utilizados no Algoritmo 2 para os resultados utilizando aproximação esférica apresentados nas Tabelas 4 e 5 foram os seguintes: quando $C = 1$ foi definido $\rho = 0,35$; quando $C = 100$ foi definido $\rho = 48$; e finalmente quando $C = 1000$ foi definido $\rho = 0.49$. Nenhum valor foi adotado como o limite máximo de iterações no Algoritmo 2, mas vale ressaltar que para $\rho = 0.35$ a média foram de 35 iterações, quando $\rho = 48$ a média foram de 300 iteracões, e quando $\rho = 0.49$ a média de iterações foram 800, para todos os grupos de testes. O critério de parada foi adotado como $tol_\epsilon =$ 1e-4.

Na Tabela 6 ilustramos os resultados utilizando aproximação diagonal Quasi-

Tabela 6. Tempo computacional utilizando Aproximação Cauchy com função Kernel Gaussiana

Nome	C	Tempo			Acurácia		
		Libsvm	Esf-MPF	Esf-MFV	Libsvm	Esf-MPF	Esf-MPF
heart scale	1	0.002	0.001	0.001	0.813	0.810	0.810
	100	0.002	0.006	0.008	0.789	0.786	0.787
	1000	0.002	0.012	0.018	0.762	0.740	0.738
australian scale	1	0.006	0.002	0.003	0.854	0.856	0.856
	100	0.010	0.022	0.028	0.848	0.832	0.831
	1000	0.019	0.052	0.063	0.812	0.718	0.718
fourclass scale	1	0.007	0.004	0.005	0.803	0.804	0.799
	100	0.010	0.038	0.046	0.968	0.822	0.798
	1000	0.015	0.088	0.107	0.995	0.745	0.723
ionosphere scale	1	0.003	0.001	0.001	0.917	0.924	0.924
	100	0.002	0.007	0.010	0.927	0.923	0.924
	1000	0.003	0.015	0.022	0.904	0.877	0.878
sonar scale	1	0.003	0.000	0.001	0.769	0.788	0.788
	100	0.003	0.004	0.006	0.847	0.844	0.845
	1000	0.003	0.008	0.008	0.842	0.843	0.842
splice scale	1	0.010	0.001	0.001	0.799	0.802	0.800
	100	0.012	0.007	0.006	0.788	0.789	0.790
	1000	0.012	0.011	0.011	0.795	0.793	0.793

Cauchy. Os resultados mostram valores semelhantes entre Esf-MPF e Esf-MFV apresentados na Tabela 4 com Cauchy-MPF e Cauchy-MFV, tanto para tempo computacional quanto para acurácia utilizando função Kernel gaussiana.

Já na Tabela 7 ilustramos os resultados utilizando aproximação diagonal Quasi-Cauchy com função Kernel linear. Os resultados também apresentam valores semelhantes a Tabela 5 onde é apresentado Esf-MPF e Esf-MFV com função Kernel linear com algumas vantagens de tempo computacional em alguns casos para a aproximação Quasi-Cauchy. Para os algoritmos Cauchy-MPF e Cauchy-MFV os valores do parâmetro ρ foram definidos de acordo como já mencionado.

Assim como exibido nas Figuras 3 e 4, para a aproximação diagonal Quasi-Cauchy os resultados para as bases australian.scale e splice.scale se mantém semelhantes.

6. Conclusões e proposta de continuidade

Neste trabalho, foi apresentado um novo framework para resolver o problema de treinamento de Support Vector Machines (SVM) não linear. O framework consiste na combinação de duas abordagens. Na primeira, é utilizada uma técnica de aproximação quadrática separável da função objetivo do problema, que tem como consequência um novo subproblema, caracterizado como problema da mochila quadrático e separável (PMQS). Seundo nossa pesquisa bibliográfica, tal abordagem ainda não foi empregada para resolver um problema de otimização não linear no formato que o treinamento do SVM possui. Resultados numéricos ainda apontam que tempo computacional e acurácia são satisfatórios quando comparado com o solver LIBSVM para os conjuntos de dados

Tabela 7. Tempo computacional utilizando Aproximação Cauchy com função Kernel Linear

Nome	C	Tempo			Acurácia		
		Libsvm	Esf-MPF	Esf-MFV	Libsvm	Esf-MPF	Esf-MPF
heart scale	1	0.002	0.002	0.002	0.825	0.819	0.818
	100	0.047	0.011	0.016	0.837	0.819	0.815
	1000	0.392	0.013	0.017	0.826	0.816	0.815
australian scale	1	0.011	0.003	0.004	0.852	0.854	0.855
	100	0.216	0.048	0.059	0.857	0.847	0.849
	1000	4.145	0.061	0.071	0.854	0.674	0.689
fourclass scale	1	0.006	0.006	0.007	0.769	0.745	0.735
	100	0.021	0.082	0.096	0.764	0.729	0.733
	1000	0.108	0.096	0.111	0.768	0.733	0.723
ionosphere scale	1	0.003	0.001	0.002	0.872	0.848	0.845
	100	0.051	0.014	0.022	0.862	0.834	0.835
	1000	0.258	0.017	0.027	0.864	0.828	0.833
sonar scale	1	0.002	0.001	0.001	0.765	0.740	0.741
	100	0.008	0.008	0.012	0.738	0.715	0.709
	1000	0.009	0.010	0.014	0.746	0.709	0.704
splice scale	1	0.017	0.001	0.002	0.760	0.744	0.742
	100	1.388	0.022	0.031	0.756	0.738	0.734
	1000	6.204	0.023	0.033	0.753	0.736	0.732

testados.

Em nossos estudos preliminares, desenvolvemos ainda uma estratégia de ponto fixo (MPF) baseada em [Münnich et al. 2012] e suas referências no qual proporcionou ao framework um tempo computacional mais competitivo quando comparado a outros métodos unidimensionais.

Nos testes numéricos foram utilizados o MPF, detalhado na Seção 4.2, o Método de Fixação de Variável desenvolvido por [Kiwiel 2008] e o Método Secante apresentado em [Münnich et al. 2012]. Nos resultados, podemos observar a estabilidade que o algoritmo possui devido as propriedades dos subproblemas gerados.

A proposta foi implementada em $MATLAB$ e aplicada para diversos problemas clássicos da literatura. No geral, em relação a acurácia e tempo computacional, a proposta mostrou-se competitiva com os métodos dos estados da arte, como LIBSVM. É importante ressaltar que alguns fatores da proposta ainda precisam ser melhorados, como os parâmetros de entrada. A seguir, serão elencadas algumas direções para continuidade da pesquisa e conclusão da Tese.

- Aprimorar o Método de Fixação de Variável (MPF): o método MPF é resultado de uma pesquisa conjunta que ainda encontra-se em andamento;
- Ampliar o estudo de métodos de aproximações separáveis e aproximações diagonais da matriz Hessiana;
- Estudo de outras técnicas para mitigar o problema de escala: quando o parâmetro C do problema de SVM possui valores altos como $C = 100$ e $C = 1000$, a proposta de algoritmo encontra dificuldades em resolver o problema. Para contornar

tal dificuldade, foi adotada uma técnica de reduzir as fronteiras da restrição da caixa a cada iteração (região de confiança).

Referências

Andrei, N. (2019). A diagonal quasi-newton updating method for unconstrained optimization. *Numerical Algorithms*, 81(2):575–590.

Bhavsar, H. and Panchal, M. H. (2012). A review on support vector machine for data classification. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 1(10):185–189.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classiers. pages 144–152.

Bottou, L. and Lin, C.-J. (2007). Support vector machine solvers. *Large scale kernel machines*, 3(1):301–320.

Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., and Lopez, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215.

Chang, C.-C. and Lin, C.-J. (2001). Training v-support vector classifiers: theory and algorithms. *Neural computation*, 13(9):2119–2147.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27.

Chauhan, V. K., Dahiya, K., and Sharma, A. (2019). Problem formulations and solvers in linear svm: a review. *Artificial Intelligence Review*, 52(2):803–855.

Chen, P.-H., Fan, R.-E., and Lin, C.-J. (2006). A study on smo-type decomposition methods for support vector machines. *IEEE Trans. Neural Networks*, 17(4):893–908.

Christmann, A. and Steinwart, I. (2008). Support vector machines.

Collobert, R. and Bengio, S. (2001). Svmtorch: Support vector machines for large-scale regression problems. *Journal of machine learning research*, 1(Feb):143–160.

Cominetti, R., Mascarenhas, W. F., and Silva, P. J. (2014). A newton's method for the continuous quadratic knapsack problem. *Mathematical Programming Computation*, 6(2):151–169.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Dennis, Jr, J. E. and Wolkowicz, H. (1993). Sizing and least-change secant methods. *SIAM Journal on Numerical Analysis*, 30(5):1291–1314.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213.

Glasmachers, T., Igel, C., Bennett, K. P., and Parrado-Hernández, E. (2006). Maximum-gain working set selection for svms. *Journal of Machine Learning Research*, 7(7).

Groenwold, A. A., Etman, L., and Wood, D. W. (2010). Approximated approximations for sao. *Structural and Multidisciplinary Optimization*, 41(1):39–56.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). A practical guide to support vector classification.

Joachims, T. (1998). Making large-scale svm learning practical. Technical report, Technical report.

Joachims, T. (1999). Svmlight: Support vector machine. *SVM-Light Support Vector Machine http://svmlight. joachims. org/, University of Dortmund*, 19(4).

Kiwiel, K. (2008). Variable fixing algorithms for the continuous quadratic knapsack problem. *Journal of Optimization Theory and Applications*, 136(3):445–458.

Lin, C.-J. et al. (2005). Working set selection using second order information for training svm.

Mercer, J. (1909). Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458):415–446.

Münnich, R. T., Sachs, E. W., and Wagner, M. (2012). Numerical solution of optimal allocation problems in stratified sampling under box constraints. *AStA Advances in Statistical Analysis*, 96(3):435–450.

Osuna, E. E. (1998). *Support vector machines: Training and applications*. PhD thesis, Massachusetts Institute of Technology.

Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines.

Saunders, C., Stitson, M. O., Weston, J., Bottou, L., Smola, A., Lecun, R. Y., et al. (1998). Support vector machine reference manual.

Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30.

Shao, Y.-H., Zhang, C.-H., Wang, X.-B., and Deng, N.-Y. (2011). Improvements on twin support vector machines. *IEEE transactions on neural networks*, 22(6):962–968.

Snyman, J. and Hay, A. (2001). The spherical quadratic steepest descent (sqsd) method for unconstrained minimization with no explicit line searches. *Computers & Mathematics with Applications*, 42(1-2):169–178.

Suykens, J. A., Van Gestel, T., De Brabanter, J., De Moor, B., and Vandewalle, J. P. (2002). *Least squares support vector machines*. World scientific.

Svensén, M. and Bishop, C. M. (2007). Pattern recognition and machine learning.

Tian, Y., Shi, Y., and Liu, X. (2012). Recent advances on support vector machines research. *Technological and economic development of Economy*, 18(1):5–33.

Venkateshan, S., Patel, A., and Varghese, K. (2014). Hybrid working set algorithm for svm learning with a kernel coprocessor on fpga. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(10):2221–2232.

Zhu, M., Nazareth, J. L., and Wolkowicz, H. (1999). The quasi-cauchy relation and diagonal updating. *SIAM Journal on Optimization*, 9(4):1192–1204.