Projeto de Programação Não Linear

Support Vector Machine (SVM)

Alunos:
Vinicius Barcellos
Samuel Kutz
Matheus Morishita
Marcelo Baptista

2º Semestre de 2023

1 Introdução

As **Máquinas de Vetores de Suporte** (SVMs) são uma classe poderosa de algoritmos de Aprendizado de Máquina, amplamente utilizadas em tarefas de *classificação* e *regressão*. Originalmente desenvolvidas para problemas de classificação binária, as SVMs foram posteriormente estendidas para abordar cenários multiclasse e regressão.

A essência dos SVMs reside na capacidade de encontrar o *hiperplano* de decisão ideal que separa distintamente as diferentes classes no espaço de características.

2 O problema

O propósito das **Máquinas de Vetores de Suporte** (SVMs) é identificar um *hiperplano* que consiga separar eficientemente dois grupos distintos de dados. No entanto, esta tarefa enfrenta um desafio intrínseco: a potencial existência de infinitos hiperplanos capazes de dividir os dados.

Um *hiperplano* ideal não apenas separa as classes, mas também maximiza a margem entre elas. A margem, neste contexto, representa a distância entre o *hiperplano* e os pontos mais próximos de cada classe. Quanto maior a margem, mais eficaz é a separação dos dados, conferindo maior robustez e capacidade de generalização para dados não observados.

A otimização do *hiperplano* não é uma tarefa trivial. É necessário levar em consideração não apenas a separação das classes, mas também a *minimização* do risco de erro de classificação. Em outras palavras, o hiperplano ótimo é aquele que *equilibra* a busca pela *maior* margem com a *minimização* de erros.

Para lidar com conjuntos de dados *não linearmente separáveis*, as **SVMs** empregam funções *kernel*, que realizam o mapeamento dos dados para espaços de características de dimensões mais altas. Esse procedimento amplia a capacidade das SVMs para encontrar *hiperplanos* ótimos, mesmo em situações onde a separação linear seria impossível.

Entretanto quando os dados são *linearmente separáveis*, significa que existe um *hiperplano* que pode perfeitamente separar as classes sem erros de classificação. Nesses casos, a SVM pode explorar essa propriedade e encontrar o *hiperplano* ótimo sem a necessidade de transformações adicionais.

O desafio central na aplicação das **SVMs** reside na escolha criteriosa do *hiperplano* ótimo, considerando a *maximização* da margem, a *minimização* de erros e a *adaptação* a conjuntos de dados complexos. A busca por essa solução ótima é a essência da eficácia e versatilidade das SVMs ao enfrentar uma variedade de desafios no campo do aprendizado de máquina.

2.1 Fundamentação Teórica

Support Vector Machine (SVM) é um modelo de classificação. O objetivo do modelo é encontrar um *hiperplano* que passa entre os pontos de um dataset, de

maneira a separa-los, para isso o método utiliza-se de modelos de otimização para encontrar o "melhor" hiperplano possível, i.e. o *hiperplano ótimo*.

Os pontos mais importantes são aqueles mais próximos do hiperplano, portanto, todos os pontos que não estão pertos não são levados em conta. Esses pontos que são levados em consideração são chamados *Vetores de Suporte*, pois apenas eles que ajudam na busca pelo hiperplano ótimo.

Os tipos de pontos que estaremos lidando em nossos problemas serão do tipo:

$$\Omega = \{(x_i, y_i) \subset \mathbb{R}^p \times \{-1, 1\}; i = 1, 2, \cdots, n\}$$

Ou seja, como teremos apenas dois grupos, y_i pode apenas pertencer a $\{-1, 1\}$.

Hiperplano

Para o $\mathbb{R}^2$ uma reta pode ser escrito como:

$$y = ax + b \rightarrow ax + by = c$$

Para o $\mathbb{R}^3$ teriamos um plano, que pode ser escrito da forma:

$$ax + by + cz = d$$

Para generalizar para n dimensões poderiamos escrever um hiperplano da forma:

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = 0$$

Chamando $w_0 = b$ teriamos:

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = 0$$

que também pode ser escrito da forma vetorial como:

$$w^Tx + b = 0 \equiv (w \cdot x) + b = 0$$

onde $w, x \in \mathbb{R}^n$ e $b \in \mathbb{R}$

Quando eu sei que o Hiperplano Separa os Dois Conjuntos?

Quando um conjunto de pontos está "acima" do hyperplano, ele pertence a um grupo, quando está "abaixo" do hyperplano, ele pertence a outro. Como o conceito de "cima" e "baixo" não é tão intuitivo em n dimensões, podemos escrever matemáticamente como:

$$\begin{cases} w^Tx_i + b > 0 & \text{p/ } y_i = 1 \\ w^Tx_i + b < 0 & \text{p/ } y_i = -1 \end{cases} \tag{1}$$

2.2 Modelagem Matemática

Considere um hiperplano que separe os ponto no espaço:

$$w^T x + b = 0$$

nós queremos a *Margem Máxima* entre o hiperplano e os pontos perto do hiperplano, ou sejá, queremos *Maximizar* a distância entre os pontos e o plano. Assim teremos que:

$$(w^T x_i) + b \geq 1 \text{ se } y_i = 1 \tag{2}$$
$$(w^T x_i) + b \leq -1 \text{ se } y_i = -1 \tag{3}$$

Queremos que a distância entre os pontos onde $y_i = 1$ e $y_i = -1$ seja igual, portanto pegamos a distância entre(2) e (3) como:

$$\frac{2}{\|w\|} = \frac{2}{\sqrt{w^T w}}$$

ou seja, podemos modelar esse problema como:

$$\max_{w,b} \frac{2}{\|w\|}$$

porém, minimizar um elemento que está no denominador pode ser problemático númericamente, portanto podemos considerar um caso equivalente:

$$\max_{w,b} \frac{2}{\|w\|} \equiv \min_{w,b} \frac{\|w\|^2}{2} \equiv \min_{w,b} \frac{w^T w}{2}$$

portanto, o problema de otimização que teremos de resolver será:

$$\begin{aligned} \min_{w,b} \quad & \frac{w^T w}{2} \\ \text{s.a.} \quad & y_i((w^T x_i) + b) \geq 1, \\ & i = 1, 2, \cdots \end{aligned}$$

Cujo dual é dado por [2]:

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^l} \quad & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j (x_i)^T (x_j) \\ \text{s.a.} \quad & \alpha_i \geq 0; i = 1, \cdots, l \\ & y^T \alpha = 0 \end{aligned}$$

Vale lembrar que esse é modelo para quando assumimos que os pontos são *Linearmente Separáveis*. Para o caso onde os dados não são linearmente

separáveis, nosso modelo sería do tipo:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ \text{s.a.} \quad & y_i((w^T \phi(x_i)) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0; i = 1, \cdots, l \end{aligned}$$

Onde se $\xi_i > 1$ então x_i não está do lado correto do hiperplano e C é o parâmetro de penalidade. Portanto a maioria dos ξ_i são 0.

e o Dual será do tipo:

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^l} \quad & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \\ \text{s.a.} \quad & 0 \leq \alpha_i \leq C; i = 1, \cdots, l \\ & \sum_{i=1}^{l} y_i \alpha_i = 0. \end{aligned} \tag{4}$$

t.q. $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ i.e. o *Kernel*.

At optimum:

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

Onde as condições de **KKT** de (4) são satisfeitas $\forall i$ quando[3]:

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i(w^T x_i + b) \geq 1, \\ 0 < \alpha_i < C &\Rightarrow y_i(w^T x_i + b) = 1, \\ \alpha_i = C &\Rightarrow y_i(w^T x_i + b) \leq 1 \end{aligned}$$

Kernel Trick

No exemplo 4 denotamos $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ como o *Kernel*. Não é nescessário saber explicitamente o que $\phi(x)$ é, já que pode ter diferentes definições dependendo do problema a ser resolvido.

Porém é nescessário saber que o Kernel é uma transformação de nosso dados para caso eles não sejam *Lineramente Separáveis*.

2.3 Escolha de Algorítimo

Apesar de que em vários casos de *Otimização com Restrições*, algorítimos como *Programação Quadrática Sequencial* (*SQR*, do inglês *Sequential Quadratic Programming*)[1] são a melhor escolha, mas para o nosso caso, algorítimos que foram criados especificamente para treinar *SVM* foram criados, como por exemplo, a nossa escolha de algorítimo para esse trabalho *Otimização Sequencial Mínima*(*SMO*, do inglês *Sequntial Minimal Optimization*)[3].

SMO

O algorítimo *SMO* seleciona dois parâmetros α: α_i e α_j e otimiza o valor para esses dois α's, e finalmente, ele ajusta o valor do parâmetro b baseado no valor dos novos α's. Esse precesso é repetido até que todos os α's convirjam.

Selecionando os parâmetros α

Boa parte do algorítimo *SVM* é dedicado a decidir os quais α_i e α_j a se otimizar, para ssim maximizar a função objetivo.

Após escolhermos os *Multiplicadores de Lagrange* α_i e α_j, para otimizar, nós primeiro calculamos o valor desses parâmetros, então, resolvemos o problema de maximização restrita.

Primeiro, queremos encontrar os limites L e H tais que $L \leq \alpha_j \leq H$ precisa ser cumprido para que α_j satisfaça a restrição $0 \leq \alpha_j \leq C$. Pode-se mostrar que esse são dados por:

$$y_i \neq y_j, L = \max(0, \alpha_j - \alpha_i), H = \min(C, C + \alpha_j - \alpha_i) \tag{5}$$

$$y_i = y_j, L = \max(0, \alpha_i + \alpha_j - C), H = \min(C, \alpha_i + \alpha_j) \tag{6}$$

Teremos que o valor de α_j será dado por:

$$\alpha_j := \alpha_j - \frac{y_j(E_i - E_j)}{\eta} \tag{7}$$

onde

$$E_k = f(x_k) - y_k \tag{8}$$

$$\eta = 2\langle x_i, x_j \rangle - \langle x_i, x_i \rangle - \langle x_j, x_j \rangle \tag{9}$$

Pode-se pensar em E_k como o erro entre o output do SVM e o k-ésimo examplo e o valor real y_k. Ao calcular o parâmetro η você pode usar a função Kernel K no lugar do produto interno se preciso. Após, nós limitamos α_j entre o intervalo $[L, H]$

$$\alpha_j := \begin{cases} H & \text{se } \alpha_j > H \\ \alpha_j & \text{se } L \leq \alpha_j \leq H \\ L & \text{se } \alpha_j < L \end{cases} \tag{10}$$

finalmente, tendo resolvido para α_j, queremos encontrar o valor de α_i. Que é dado por

$$\alpha_i := \alpha_i + y_i y_j (\alpha_j^{(old)} - \alpha_j)$$

onde $\alpha_j^{(old)}$ é o valor de α_j antes da otimização por (7) e (10).

Calculando o limite de b

Depois de otimizar α_i e α_j, nós selecionamos o limite b tal que as condições **KKT** sejam satisfeitas for o i-ésimo e j-ésimo exemplo. Se, depois da otimização α_i não está dentro das restrições i.e., $0 < \alpha_i < C$, então o limite seguinte b_1 é válido, já que força o output do SVM a ser y_i quando o input é x_i

$$b_1 = b - E_i - y_i(\alpha_i - \alpha_i^{(old)})\langle x_i, x_i\rangle - y_j(\alpha_j - \alpha_j^{(old)})\langle x_i, x_j\rangle \tag{11}$$

Similarmente, o limite seguinte b_2 é válido se $o < \alpha_j < C$

$$b_2 = b - E_j - y_i(\alpha_i - \alpha_i^{(old)})\langle x_i, x_j\rangle - y_j(\alpha_j - \alpha_j^{(old)})\langle x_j, x_j\rangle \tag{12}$$

Se ambos $0 < \alpha_i < C$ e $0 < \alpha_j < C$, então ambos os limites são válidos, e então eles serão iguais. Se ambos os novos α's estão nos limites (i.e.,$\alpha_i = 0$ e $\alpha_i = C$ ou $\alpha_j = C$) então os limites entre b_1 e b_2 satisfazem as condições de **KKT**, nós fazemos $b := \frac{b_1+b_2}{2}$. Isso nos dá a equação completa para b:

$$b := \begin{cases} b_1 & \text{se } 0 < \alpha_i < C \\ b_2 & \text{se } 0 < \alpha_j < C \\ \frac{b_1+b_2}{2} & \text{caso contrário} \end{cases} \tag{13}$$

Algorithm 1 SMO simplificado

Input:
C: Parâmetro de regularização
tol: Tolerância numérica
maxpasses: número de iterações máximas sem mudar
$(x_1, y_1), \cdots, (x_m, y_m)$: *Dados de Treinamento*

Output:
$\alpha \in \mathbb{R}^m$: *Multiplicadores de Lagrange para solução*
$b \in \mathbb{R}$: *limite para solução*

procedure SMO:
    $\alpha_i \leftarrow 0, \forall i, b \leftarrow 0$
    passes $\leftarrow 0$
    while *(passes* $<$ *maxpasses)* **do**:
        numChangedAlphas $\leftarrow 0$
        for $i = 1, \cdots, m$, **do**
            $E_i \leftarrow f(x_i) - y_i$
            if $((y_i E_i < -tole\ \alpha_i < C) \| (y_i E_i > tol\ e\ \alpha_i > 0))$ **then**
                Selecione $j \neq i$ *Aleatóriamente*
                Calcule $E_j \leftarrow f(x_j) - y_j$
                $\alpha_i^{(old)} \leftarrow \alpha_i$
                $\alpha_j^{(old)} \leftarrow \alpha_j$
                Calcule L e H por (5) e (6)
                if $L == H$ **then**
                    Continue *para o próximo i*
                Compute η *por (9)*
                if $(|\alpha_j - \alpha_j^{(old)}| < 10^{-5})$ **then**
                    Continue *para o próximo i*
                Determine o valor para α_i *usando*
                Compute $b_1 e\ b_2$ *usando (11) e (12)*
                Compute b usando (13)
                numChangedAlfas $\leftarrow$ *numChangedAlfas* $+ 1$
        if numChangedAlfas ==0 **then**
            passes $\leftarrow$ *passes* $+ 1$
        else
        passes $\leftarrow 0$
    EndWhile

3 Experimentos Numéricos

3.1 O algorítimo Implementado

3.2 Resultados Obtidos

4 Conclusões

References

[1] Ribeiro A.A e Karas E.W. *Otimização Contínua, Aspectos teóricos e computacionais*. Cengage Learning, 2014.

[2] Evelin H. M. Krulikovski. "Análise Teórica de Máquinas de Vetores Suporte e Aplicação a Classificação de Caracteres". UFPR, 2017.

[3] John C. Platt. "Sequential Minimal Optimization: A fast Algorithm for Training Support Vector Machines". In: (1998).