UFC

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS JARDINS DE ANITA - ITAPAJÉ

Reconhecimento de Padrões: Uma Comparação Entre Máquinas de Vetores de Suporte e Máquinas de Vetores de Relevância

Autores: Giovanna Dias Castro de Oliveira
Jônatas Fernandes Silva
Leandro Nascimento Adegas

Disciplina: Aprendizado de Máquina e Reconhecimento de Padrões

Itapajé, Junho, 2025

Conteúdo

Resumo

Este trabalho trata da comparação entre os classificadores máquinas de vetores de suporte (support vector machines, SVM) e máquinas de vetores de relevância (relevance vector machines, RVM). Dois conjuntos de dados gerados a partir da biblioteca *numpy*, um definido previamente e outro gerado de forma aleatória. A partir dos conjuntos de dados é realizado o treinamento dos classificadores, no SVM usando classificação de vetores de suporte (support vector classification, SVC) e RVM usando classificador de vetores de relevância de maximização de expectativa (expectation-maximization relevance vector classifier, EMRVC), ambos testados com os seguintes kernels: RFB, polinomial, sigmoid e linear. A contribuição principal deste trabalho encontra-se na observação das vantagens e desvantagens do uso de cada classificador. Dado que, o modelo RVM é bayesiano, ele pode fornecer propriedades a posteriores, porém, cada classificador tem seu uso específico.

1 Introdução

Os classificadores são modelos de aprendizado de máquina, eles dividem os dados em grupos, isto é, agrupa os dados em classes [1]. A classificação é uma etapa essencial para que reconhecimento de padrões aconteça de forma eficaz. Em aprendizado de máquina escolher um bom classificador para um determinado problema é de fato imprescindível para um bom desempenho.

O classificador SVM é uma técnica de classificação que utiliza máquinas de kernel esparso, normalmente é utilizado para classificações binárias. Porém, por ser uma máquina de decisão, não fornece probabilidades a porteriores [2]. Já o RVM é baseado em uma formulação bayesiana [2].

Para realizar a comparação dos dois classificadores e extrair as informações necessárias para que se chegasse a uma conclusão quanto as vantagens e desvantagens de cada modelo foi utilizado dois conjuntos de dados gerados através da biblioteca *numpy*, nativa do da linguagem Python, um definido previamente e outro gerado de forma aleatória. Para melhor entendimento, vamos chamar o conjunto de dados que foi definido previamente de conjunto de dados 1, e o conjunto de dados aleatório XOR, de conjunto de dados 2. Dado que, o modelo RVM é bayesiano, ele pode fornecer propriedades a posteriores, porém, cada classificador tem seu uso específico.

Listing 1: Conjunto de Dados 1

```
X = np.array(
    [
        [0.4, -0.7],
        [-1.5, -1.0],
        [-1.4, -0.9],
        [-1.3, -1.2],
        [-1.1, -0.2],
        [-1.2, -0.4],
        [-0.5, 1.2],
        [-1.5, 2.1],
        [1.0, 1.0],
        [1.3, 0.8],
        [1.2, 0.5],
        [0.2, -2.0],
        [0.5, -2.4],
        [0.2, -2.3],
        [0.0, -2.7],
        [1.3, 2.1],
    ]
)

y = np.array([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1])
```

Listing 2: Conjunto de Dados 2

```
np.random.seed(0)
X = np.random.randn(300, 2)
y = np.logical_xor(X[:, 0] > 0, X[:, 1] > 0).astype(int)
```

O código inicial foi escrito e disponibilizado pelo professor Dr. Júlio César, assim como os materiais necessários para estudo dos classificadores objeto de estudo deste trabalho.

A organização deste trabalho segue a seguinte ordem. A seção 2 contém informações a respeito dos conjuntos de dados utilizado nos testes, os kernels utilizados e alguns trabalhos relacionados. A seção 3 explica de maneira breve o funcionamento dos classificadores SVM e RVM, além da utilização do SVM e EMRVC. A seção 4 explica o ambiente e os métodos utilizados para as demonstrações empíricas deste trabalho. A seção 5 traz os resultados obtidos após os testes e análises dos modelos de classificação. Por fim, a seção 6 traz a conclusão deste trabalho.

2 Background e Trabalhos Relacionados

2.1 Background

2.1.1 Conjunto de Dados 1

Como mostrados no código acima, o conjunto de dados 1 foi gerado de forma previamente definida. Onde os dados são formados por um vetor de 16 (dezesseis) posições. Na Figura 1 podemos ver a distribuição dos valores de X em relação a y. Nota-se que, os dados apesar de simples, estão com uma boa distribuição. Outrossim, se pode perceber que os valores de X_0 (0., -0.7), classe 0, estão em uma região de divisão da classe 1, ou seja, se X_0 se encontra em uma fronteira de decisão, como veremos mais a frente.

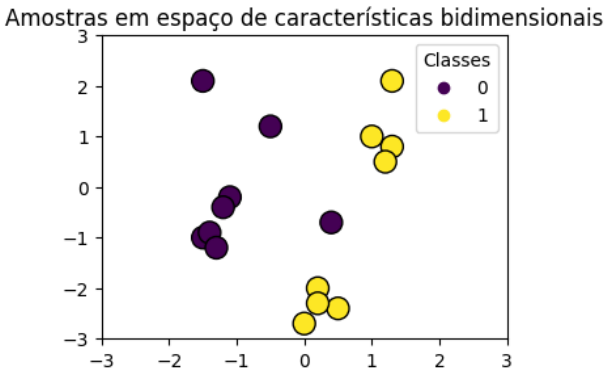


Figura 1: Amostra dos Dados do Conjunto 1.

Quanto aos dados aleatório, eles foram gerados como XOR, isto é, um conjunto de dados não linearmente separáveis, ao contrário do conjunto de dados 1, demonstrados na Figura 1.

2.1.2 Kernels

Para os testes nos modelos SVM e RVM foram utilzados 04 (quatro) kernels, sendo eles, kernel RFB, polinomial, sigmoid e linear. A seguir veremos como funciona cada um deles.

Kernel RFB O kernel da função de base radial (RBF), também conhecido como kernel Gaussiano, é o kernel padrão para Máquinas de Vetores de Suporte no scikit-learn. Ele mede a similaridade

entre dois pontos de dados em dimensões infinitas e, em seguida, se aproxima da classificação por maioria de votos. A função kernel é definida como:

$$K(x_1, x_2) = exp(-\gamma||x_1 - x_2||^2) \tag{1}$$

onde γ controla a influência de cada amostra de treinamento individual na fronteira de decisão.

Quanto maior a distância euclidiana entre dois pontos $||x_1 - x_2||^2$ mais próxima de zero a função kernel. Isso significa que dois pontos distantes têm maior probabilidade de serem diferentes.

Kernel Polinomial O kernel polinomial altera a noção de similaridade. A função kernel é definida como:

$$K(x_1, x_2) = (\gamma \cdot x_1^T x_2 + r)^2 \tag{2}$$

onde d é o grau (degree) do polinômio, γ controla a influência de cada amostra de treinamento individual no limite de decisão e r é o termo de viés ($coef0$) que desloca os dados para cima ou para baixo. Aqui, usamos o valor padrão para o grau do polinômio na função kernel ($degree$=3). Quando $coef0$=0 (o padrão), os dados são apenas transformados, mas nenhuma dimensão adicional é adicionada. Usar um kernel polinomial é equivalente a criar :class: sklearn.preprocessing.PolynomialFeatures e então ajustar um :class: sklearn.svm.SVC com um kernel linear nos dados transformados, embora essa abordagem alternativa seja computacionalmente dispendiosa para a maioria dos conjuntos de dados.

Kernel Sigmoid A função do núcleo sigmoide é definida como:

$$K(x_1, x_2) = tanh(\gamma \cdot x_1^T x_2 + r) \tag{3}$$

onde o coeficiente do kernel γ controla a influência de cada amostra de treinamento individual na fronteira de decisão e r é o termo de viés ($coef0$) que desloca os dados para cima ou para baixo.

No kernel sigmoide, a similaridade entre dois pontos de dados é calculada usando a função tangente hiperbólica ($tanh$). A função do kernel dimensiona e possivelmente desloca o produto escalar dos dois pontos (x_1 e x_2).

Kernel Linear O kernel linear é o produto escalar das amostras de entrada:

$$K(x_1, x_2) = x_1^T x_2 \tag{4}$$

Em seguida, ele é aplicado a qualquer combinação de dois pontos de dados (amostras) no conjunto de dados. O produto escalar dos dois pontos determina a :func: sklearn.metrics.pairwise.cosine_similarity entre ambos os pontos. Quanto maior o valor, mais semelhantes os pontos são.

2.2 Trabalhos Relacionados

2.2.1 Aprendizagem Bayesiana Esparsa e a Máquina de Vetores de Relevância

Tem o objetivo de apresentar o modelo RVM [3]. Mostra que o funcionamento do RVM é semelhante ao SVM, porém, fundamentado em um modelo bayesiano, assim como em [2]. Faz uma comparação entre o SVM e o RVM, tendo como aspectos de comparação o tipo de saída, esparsidade, kernel necessário, se necessita da validação cruzada (*cross validation*) e o custo computacional.

2.2.2 Um Tutorial Sobre Máquinas de Vetores de Relevância para Regressão e Classificação com Aplicações

O objetivo deste trabalho é apresentar a teoria, o funcionamento e aplicações do RVM [4]. Ele mostra que, o RVM é um classificador bayesiano [2], que induz esparsidade no peso dos modelos [4], gera modelos mais esparsos com um número menor de vetores relevantes em relação ao SVM. Também demonstra que o SVM e RVM tem estruturas semelhantes, mas com formulação propabilística e treinamento mais complexo.

2.2.3 Uma Comparação Entre SVM e RVM para Classificação de Documentos

O objetivo deste trabalho é comparar o desempenhos das técnicas SVM e RVM na tarefa de classificação de documentos. [5]. Apesar de a comparação ser realizada em um cenário diferente dos testes realizados no presente artigo, os resultados obtidos pelos autores também mostram vantagens e desvantagens entre SVM e RVM. Onde o RVM foi superior ao SVM em quase todos os conjuntos de dados, e com um desempenho melhor nas métricas Micro e Macro F1, porém, ele necessitou de um maior tempo de treinamento. Concluindo que, mesmo o custo computacional do RVM ser maior que o SVM, a abordagem bayesiana permite maior generalização, como podemos ver em [2, 4].

2.2.4 Máquinas de Vetores de Relevância: Uma Introdução

O objetivo deste artigo é apresentar uma introdução ao RVM [6]. Ele também nos mostra vantagens e desvantagens do RVM em relação ao SVM, utilizando um conjunto de dados preparado com amostras de sondagens geológicas. Os resultados mostram que, o RVM foi melhor na questão probabilística, esparsidade e para além disso, foi mais eficaz com várias classes. Em razão de ser bayesiano, como citado em [2,5]. Contudo, algumas desvantagens também foram notadas aqui, como por exemplo, o alto custo computacional, estando assim de acordo com [5].

3 Modelo de Solução

3.1 SVM

A técnica de classificação SVM é uma técnica mais moderna. Semelhante aos *perceptrons*. O SVM surge como uma melhoria do *perceptron*. O SVM seleciona um hiperplano, não somente separa os pontos em duas classes, ele maximiza as margens. A equação (1), exemplifica o a fórmula que o SVM usa para isso [7].

$$SVM = w \cdot x + b = 0 \tag{5}$$

Ao selecionar o hiperplano ele traça 03 (três) retas ou vetores, onde um vetor representa uma margem de dados de uma determinada classe X e outro representa outra margem de um dados de uma classe Z. Mas ao se fazer isso abre-se uma brecha para erro, e daí surge a pergunta: e se houver um ponto no no espaço entre as classes X e Z? Para resolver este problema o SVM se utiliza de um terceiro vetor central, que como dito acima, maximiza as margens das classes, aproximando-se para encontrar a classificação correta, como demonstrado na Figura 2.

3.1.1 SVC

Conforme a documentação do *scikit-learn* [8], a implementação é baseada em libsvm. É ideal para um conjunto de dados pequenos, como o do nosso caso de uso. O suporte para multiclasse é tratado em um esquema um contra um, também chamada de estratégia `"O-O"`. Essa estratégia consiste em uma comparação binária, onde ele compara as classes uma com outra, por exemplo, X_0 com Z_0, Z_0 com C_0, dessa forma não deixando de ser um classificador de decisão binário.

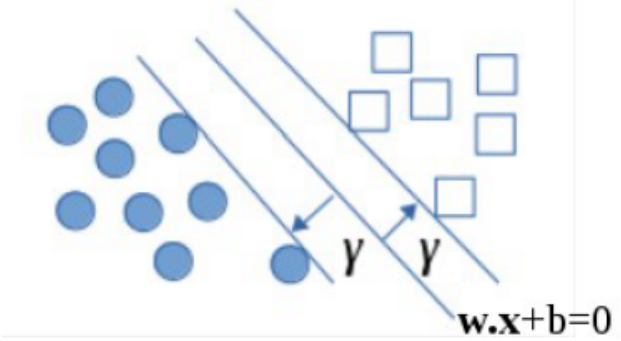


Figura 2: SVM Seleciona Maximizando a Distância γ entre os hiperplanos
Source: Algoritmos e Técnicas de Classificação, Prof. Dr. Júlio César, [7]

3.2 RVM

Conforme a documentação do *scikit-learn* [9], o RVM é um conjunto de métodos de aprendizagem supervisionados para problemas de regressão e classificação que exigem apenas uma representação esparsa do kernel. O funcionamento do RVM é semelhante ao SVM, porém, fundamentado em um modelo bayesiano, assim como em [2].

O modelo define uma distribuição condicional para uma variável de valor real t, dado um vetor de entrada x, que assume a forma

$$p(t|x, w, \beta) = N(t|y(x), \beta^{-1} \tag{6}$$

onde $\beta = \sigma^2$ é a precisão do ruído, e a média é dada por um modelo linear [2].

3.2.1 EMRVC

O classificador de vetores de relevância de maximização de expectativa, é segundo a documentação [10], "Implementação da máquina vetorial de relevância de Mike Tipping para classificação usando a API scikit-learn.". Para o suporte de multiclasse é utilizada a estratégia de um contra-descanso [10].

4 Metodologia

4.1 Pesquisa Bibliográfica

Nesta primeira etapa buscamos trabalhos já publicados que tivessem relação com o tema abordado neste artigo, além de obter informações nos materiais de aula, disponibilizados pelo já citado anteriormente, professor Dr. Júlio César.

As pesquisas pelos trabalhos foram realizadas no buscador *Google*, onde foi encontrado os trabalhos referenciados anteriormente na seção 2.2. Os materiais de aula se encontravam disponíveis no programa SIGAA.

4.2 Pesquisa Experimental

Nesta fase, executamos testes com os dados gerados.

4.2.1 O Ambiente

Os testes foram realizados em um ambiente *cloud*, por meio do google colab. A linguagem de programação utilizada foi Python em sua versão mais atualizada (*latest*). Para os testes foram

criados dois conjuntos de dados, um linear (conjunto de dados 1) e outro não linear, isto é, um conjunto de dados XOR (conjunto de dados 2). No colab o código que estava escrito usando SVM foi reescrito utilizando o classificador RVM.

Os códigos utilizados estão disponíveis em:
https://github.com/jonatasfernandessilva7/Trabalho-04—Reconhecimento-de-Padroes.git

4.2.2 Passo a Passo para a Prova Empírica

Para criar os dados foram utilizados os códigos descritos na seção 1 do presente artigo. Após a criação dos dois conjuntos de dados o código que estava utilizando o SVM foi reescrito utilizando o RVM, no repositório disponibilizado acima está todo o notebook que foi utilizado, nele ainda está o código com SVM, em caráter apenas informativo, para que dessa forma possa haver uma comparação das diferenças entre os dois modelos.

Para realizar a reescrita foi seguido o tutorial descrito na documentação do RVM, disposta em [11]. A base do código é

```
print(__doc__)

model = EMRVC(kernel="kernel").fit(X, y)

x0, x1 = np.meshgrid(np.linspace(-4, 4, 100), np.linspace(-4, 4, 100))
x = np.array([x0, x1]).reshape(2, -1).T
plt.scatter(X[:, 0], X[:, 1], s=40, c=y, marker="x")
plt.scatter(model.relevance_vectors_[:, 0], model.relevance_vectors_[:, 1],
s=100, facecolor="none", edgecolor="g")
plt.contourf(x0, x1, model.predict_proba(x)[:, 1].
reshape(100, 100), np.linspace(0, 1, 5), alpha=0.2)
plt.colorbar()
plt.xlim(-4, 4)
plt.ylim(-4, 4)
plt.gca().set_aspect("equal", adjustable="box")
```

com esta base foi feito a realização dos testes utilizando os kernels supracitados na seção 2.1.

Para o conjunto de dados 2, a ideia foi a mesma, porém, a base de código um pouco diferente, o código usado para a classificação do conjunto de dados 2 foi.:

```
def plot_rvm_decision_boundary(kernel, ax):
    model = EMRVC(kernel=kernel).fit(X, y)
    Z_prob = model.predict_proba(grid)[:, 1].reshape(x0.shape)
    cont = ax.contourf(x0, x1, Z_prob, levels=np.linspace(0, 1, 20), cmap="coolwarm", alpha=0.3)
    plt.colorbar(cont, ax=ax)
    ax.scatter(X[:, 0], X[:, 1], c=y, edgecolors="k", s=30)
    ax.scatter(model.relevance_vectors_[:, 0], model.relevance_vectors_[:, 1],
               s=100, facecolors="none", edgecolors="g", label="Vetores relevantes")
    ax.set_title(f"Kernel: {kernel}")
    ax.set_xlim(-4, 4)
    ax.set_ylim(-4, 4)
    ax.set_aspect("equal", adjustable="box")
    ax.legend()

np.random.seed(0)
X = np.random.randn(300, 2)
y = np.logical_xor(X[:, 0] > 0, X[:, 1] > 0).astype(int)
x0, x1 = np.meshgrid(np.linspace(-4, 4, 100), np.linspace(-4, 4, 100))
grid = np.array([x0.ravel(), x1.ravel()]).T
fig, axs = plt.subplots(2, 2, figsize=(12, 12))
kernels = ["linear", "poly", "rbf", "sigmoid"]
for ax, kernel in zip(axs.ravel(), kernels):
```

```
    plot_rvm_decision_boundary(kernel, ax)
plt.show()
```

5 Resultados

5.1 Comparação entre o SVM com Uso do SVC e RVM com Uso do EMRVC para o Conjunto de Dados 1

As imagens abaixo representam os resultados obtidos com o SVM e SVC. Cada imagem é o resultado de um teste com um kernel escolhido, chamamos elas de fronteira de decisão. Note que, os vetores aqui são os "pontos circulados", para melhor compreensão dos resultados apresentados.

A Figura 3 nos mostra a comparação do resultado de classificação do SVC e do EMRVC utilizando o kernel linear. Podemos notar que, a fronteira de decisão é uma linha reta, por esta razão é linear.

Tanto na subfigura 3(a) quanto na subfigura 3(b), podemos notar que a classificação ocorre relativamente ok, embora haja um ponto na classe 0 que não foi classificado da forma correta. Outro pontos que é notável e que estrá presente em todos os resultados a seguir é o número de vetores relevantes utilizados por cada classificador, enquanto no SVM com SVC foram destacados 05 (conco) vetores, sendo 03 (três) na classe 1 e 02 (dois) na classe 0, no RVM com EMRVC foram destacados apenas 02 (dois), e ambos na classe 0. Também é visto que, a distância γ entre o vetor central e os vetores de margem é mais estreita no RVM em relação ao SVM. Essa maior aproximação denota que o RVM tem uma melhor classificação devido sua fundamentação bayesiana. Nesse caso o RVM foi melhor que o SVM.

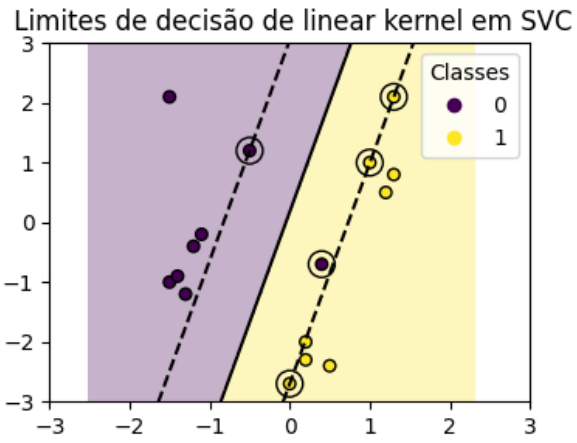
Nas subfiguras 3(c) e 3(d), demonstra-se a fronteira de decisão do SVC e EMRVC com kernel polinomial. A fronteira de decisão na subfigura 3(c) é bem mais complexa do que a fronteira utilizando o kernel linear. Nota-se que a classificação utilizando o SVC foi bem melhor neste caso que a classificação do EMRVC, enquanto no SVC todos os pontos tanto da classe 0 quanto da classe 1 foram separados de forma correta, no EMRVC, a fronteira de decisão não classificou corretamente pelo menos 01 (um) ponto da classe 0, contudo, também classificou corretamente todos os pontos da classe 1. Neste resultado o SVM foi melhor que o RVM.

Na Figura 4 é representado os resultados obtidos no SVM e RVM utilizando os kernels sigmoid e RFB. Podemos notar que na subfigura 4(a), o SVC foi piorando a sua classificação, pois cometeu mais erros, uma vez que 03 (três) pontos da classe 0 foram classificados como classe 1 e 04 (quatro) pontos da classe 1 foram classificados como classe 0. Ainda nota-se que, a fronteira de decisão não cobre uma área do hiperplano, justamente onde estão aproximadamente 85% dos pontos que foram classificados de errôneamente, isto é, 6 dos 7 pontos. Também houve muitos vetores escolhidos como relevantes. Em contraposição a isto, o EMRVC, subfigura 4(b), utilizou apenas 02 (dois) vetores de relevância, dessa forma ele conseguiu classificar corretamente 16/16 pontos entre as classes 0 e 1. Sendo assim, no kernel sigmoid o RVM foi melhor no quesito classificação.

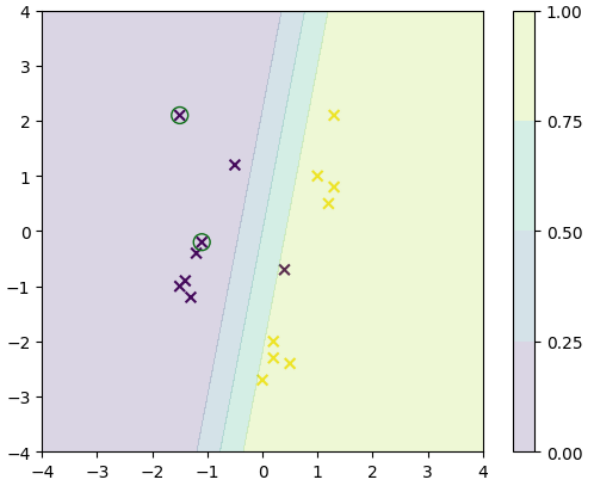
Analisando a subfigura 4(c), o SVC com kernel RBF gerou fronteiras de decisão complexas, típicas deste kernel, que buscam separar as classes de forma não linear. A distribuição dos vetores de suporte indica que múltiplos (12 no total) pontos foram considerados cruciais para definir as margens de separação entre as classes 0 e 1. Neste caso não houve nenhum erro de classificação por parte do SVM com uso do SVC. Por sua vez, a subfigura 4(d) mostra o desempenho do EMRVC com kernel RBF. Observa-se que o EMRVC utilizou um número reduzido de vetores de relevância (03 no total), o que é característico de sua natureza esparsa e da abordagem bayesiana. As regiões de probabilidade demonstram a confiança do modelo na classificação. A capacidade do RVM de gerar modelos mais esparsos com um número menor de vetores relevantes em relação ao SVM foi observada em trabalhos anteriores. Neste caso também não houve erro de classificação, sendo

Figura 3: Comparação Utilizando o Kernel Linear.

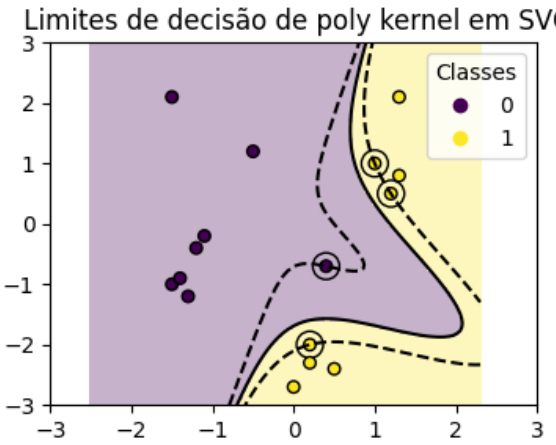
((a)) Linear SVC



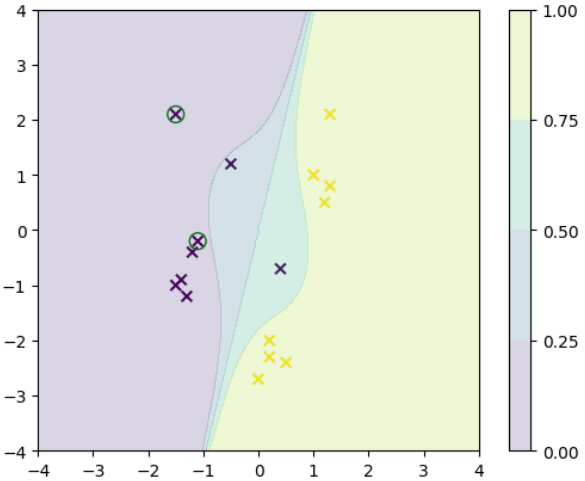
((b)) Linear EMRVC



((c)) Polinomial SVC



((d)) Polinomial EMRVC



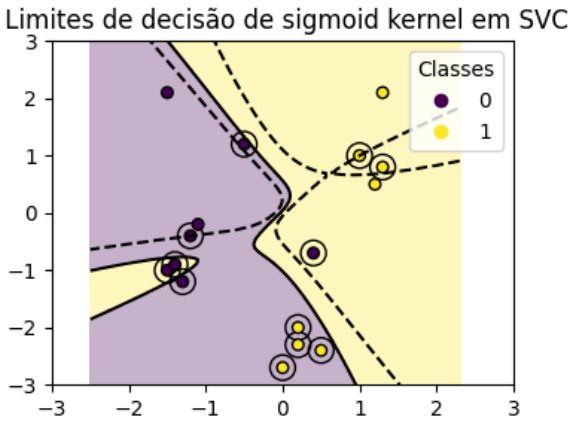
considerado um fator de desempate o número de vetores necessários para a classificação, assim, considera-se que o RVM foi superior em relação ao SVM, embora ambos modelos tenham acertado todas as classificações.

5.2 Comparação entre o SVM com Uso do SVC e RVM com Uso do EMRVC para o Conjunto de Dados 2

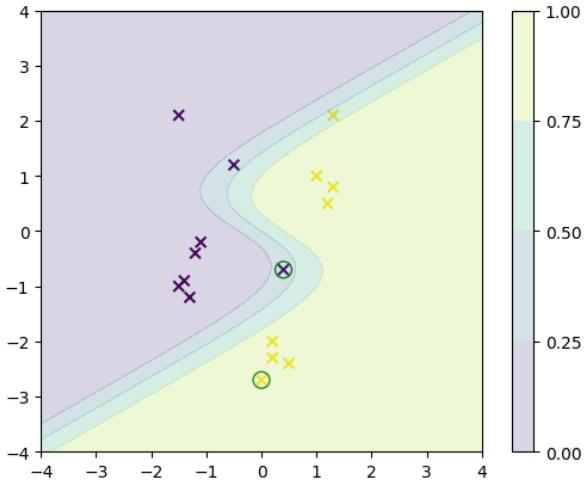
As Figuras 5 e 6 apresentadas ilustram os resultados de classificação dos modelos SVM (com SVC) e RVM (com EMRVC) em um conjunto de dados XOR, que é inerentemente não linearmente separável. Neste caso observa-se que, o kernel RBF é o mais eficaz para ambos os modelos em problemas não lineares. O EMRVC se destaca pela esparsidade do modelo e pela capacidade de fornecer probabilidades, o que é uma vantagem em relação ao SVC. O trabalho destaca as vantagens e desvantagens de cada classificador.

Figura 4: Comparação Utilizando o Kernel Linear.

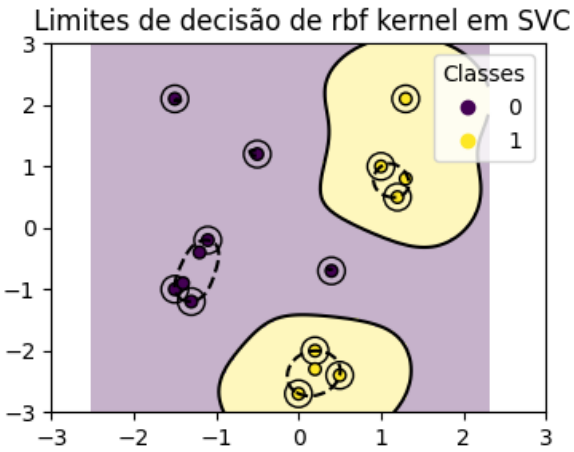
((a)) Sigmoid SVC



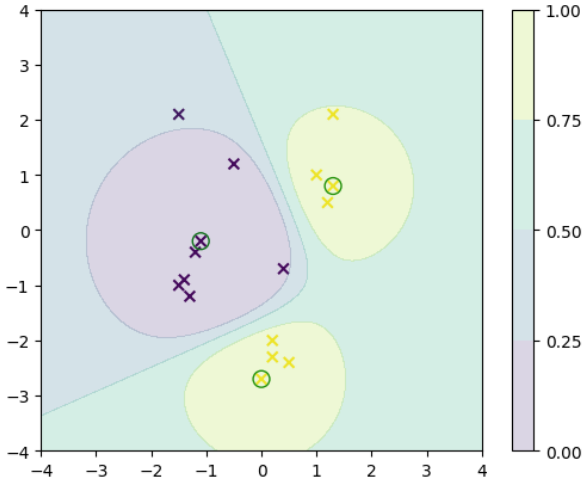
((b)) Sigmoid EMRVC



((c)) RFB SVC



((d)) RFB EMRVC



6 Conclusão

6.1 Contribuição

Ao findar das pesquisas bibliográfica e experimental, chegamos a seguinte conclusão quanto as vantagens e desvantagens de cada modelo. Baseando-se nos resultados observados por outros autores e por nossa experiência prática.

6.1.1 Vantagens e Desvantagens SVM

As vantagens do SVM são: eficácia em espaços de alta dimensão; eficaz em casos onde o número de dimensões é maior que o número de amostras; utiliza máquinas de kernel esparso; pode de lidar com dados não linearmente separáveis (com o uso de kernels não lineares como RBF, polinomial e sigmoid); implementação baseada em libsvm, ideal para conjuntos de dados pequenos; e suporte para multiclasse através da estratégia "O-O".

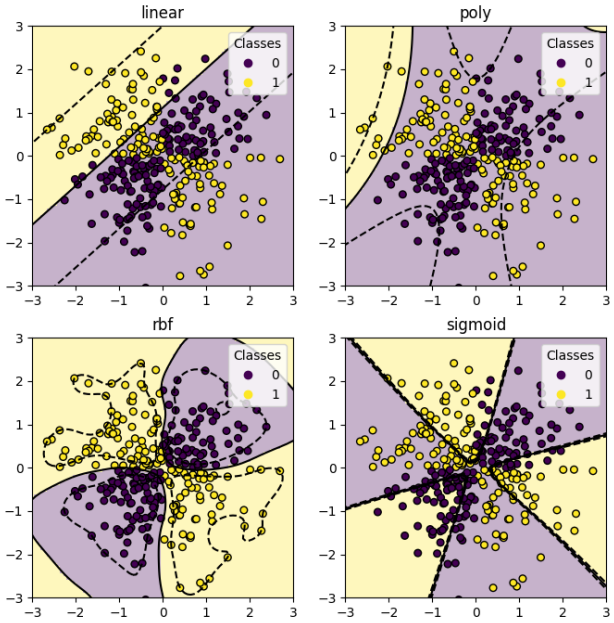


Figura 5: Resultado SVM com Dados XOR.

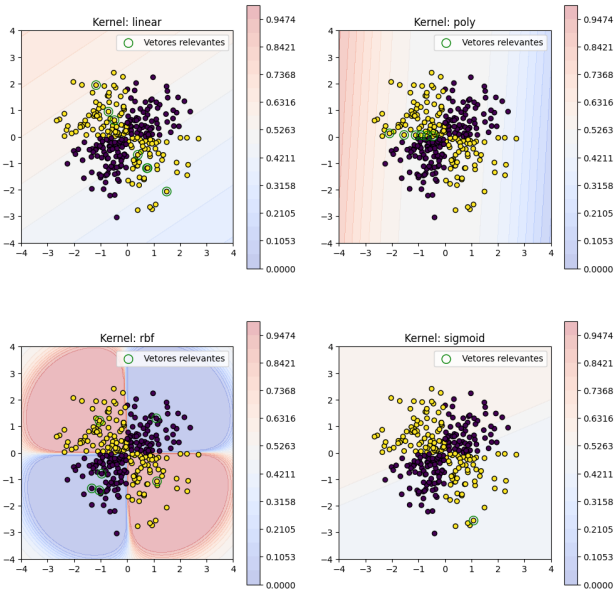


Figura 6: Resultado RVM com Dados XOR.

Suas desvantagens são: não fornece probabilidades a posteriori, por ser uma máquina de decisão; pode ser computacionalmente dispendioso para a maioria dos conjuntos de dados ao usar certas abordagens com kernel polinomial; pode ter um maior número de vetores de suporte em comparação com o RVM; em alguns casos, como no kernel sigmoid para o conjunto de dados 1, pode apresentar classificação com mais erros e fronteiras de decisão que não cobrem adequadamente o hiperplano.

6.1.2 Vantagens e Desvantagens RVM

As vantagens do SVM são: baseado em uma formulação Bayesiana, o que permite fornecer propriedades a posteriori (probabilidades); gera modelos mais esparsos, utilizando um número menor de vetores de relevância em comparação com o SVM; maior generalização, devido à sua

abordagem bayesiana; possui capacidade de classificar corretamente todos os pontos em cenários específicos, mesmo com menos vetores de relevância (e.g., kernel sigmoid para o conjunto de dados 1); o funcionamento é semelhante ao SVM, facilitando o entendimento para quem já conhece SVM; ideal para problemas de regressão e classificação que exigem apenas uma representação esparsa do kernel; e tem muita eficácia com várias classes.

Suas desvantagens são: maior custo computacional em comparação com o SVM; treinamento mais complexo devido à sua formulação probabilística; e pode não classificar corretamente todos os pontos em certos cenários, como o kernel polinomial para o conjunto de dados 1, onde o SVM teve um desempenho melhor.

6.2 Limitatações e Sugestões para Estudos Futuros

Algumas limitações que tivemos na produção deste artigo embora tenhamos testado 04 (quatro) kernels, outros kernels poderia ter sido explorados, mas nos detivemos ao proposto. Basear-se apenas nas análises visuais das fronteiras de decisão. A falta de métricas de desempenho quantitativas (acurácia, precisão, recall, F1-score, tempo de treinamento) para todos os testes dificulta uma comparação mais robusta.

Para estudos futuros podemos aplicar os classificadores em bases de dados reais e com um maior volume, para que possamos realizar testes de maior complexidade e comparar os resultados. Para além disso, podemos incluir métricas de avaliação para que a análise possa ter uma maior qualidade, robustez e confiança. Também podemos explorar novos kernels e ver como os algoritmos SVM e RVM se comportariam.

7 Agradecimentos

Agradecemos primeiramente a Deus pela inspiração e guia, ao professor Júlio César Santos dos Anjos pela orientação e dedicação, e aos valorosos colegas pela parceria e contribuições.

8 Referências

[1] "O que são modelos de classificação?." https://www.ibm.com/br-pt/think/topics/classification-models. Online; Acesso em 2025-06-20.

[2] J. C. S. dos Anjos, "Machine learning." Aula ou apresentação acadêmica, 2025.

[3] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, 2001.

[4] D. G. Tzikas, L. Wei, A. Likas, Y. Yang, and N. P. Galatsanos, "A tutorial on relevance vector machines for regression and classification with applications," *ResearchGate*, 2006.

[5] M. Rafi and M. S. Shaikh, "A comparison of svm and rvm for document classification," *Procedia Computer Science*, 2012.

[6] K. Koruk, "Relevance vector machines: An introduction," *Geomet Queens University*, 2021.

[7] J. C. S. dos Anjos, "Algoritmos e técnicas de classificação." Aula ou apresentação acadêmica, 2025.

[8] "Svc - documentação scikit-learn." https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html Online; Acesso em 2025-6-20.

[9] "Rvm - documentação scikit-learn." https://sklearn-rvm.readthedocs.io/en/latest/introduction.html. Online; Acesso em 2025-6-19.

[10] "Emrvc - documentação scikit-learn." https://sklearn-rvm.readthedocs.io/en/latest/generated/sklearn_rvm.e Online; Acesso em 2025-6-19.

[11] "Rvm for classification - documentação scikit-learn." https://sklearn-rvm.readthedocs.io/en/latest/auto_exa Online; Acesso em 2025-6-19.