Revista Eletrônica de Iniciação Científica em Computação, 2026, 24:1
ISSN: 3085-8461 • doi: 10.5753/reic.2026.6697
This work is licensed under a Creative Commons Attribution 4.0 International License.

Artigo de Pesquisa/Research Paper

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

Pattern Recognition: A Comparison Between Support Vector Machines and Relevance Vector Machines

Jônatas Fernandes [Universidade Federal do Ceará | *jonatasfernandes@alu.ufc.br*]
Leandro Adegas [Universidade Federal do Ceará |*leandroadegas@alu.ufc.br*]
Giovanna Dias [Universidade Federal do Ceará |*giovannadias@alu.ufc.br*]
Júlio César Santos dos Anjos [Universidade Federal do Ceará |*jcsanjos@ufc.br*]

Campus Jardins de Anita, Universidade Federal do Ceará, Rua Francisco José de Oliveira, s/n, Centro. Itapajé-CE. CEP: 62600-000.

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: RBF, 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 posteriori, porém, cada classificador tem seu uso específico.

Abstract. This work deals with the comparison between support vector machines (SVM) and relevance vector machines (RVM) classifiers. Two data sets were generated from the *numpy* library, one predefined and the other randomly generated. The classifiers are trained using the datasets, with SVM using support vector classification (SVC) and RVM using expectation-maximization relevance vector classifier (EMRVC), both tested with the following kernels: RBF, polynomial, sigmoid, and linear. The main contribution of this work lies in observing the advantages and disadvantages of using each classifier. Given that the RVM model is Bayesian, it can provide a posteriori properties, but each classifier has its specific use.

Palavras-chave: SVM, RVM, Reconhecimento de Padrões

Keywords: SVM, RVM, Pattern Recognition

Recebido/Received: 15 October 2025 • **Aceito/Accepted:** 06 March 2026 • **Publicado/Published:** 13 March 2026

1 Introdução

Os classificadores são modelos de aprendizado de máquina, eles dividem os dados em grupos, isto é, agrupa os dados em classes (Murel and Kavlakoglu, 2025). 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 (Tipping, 2001). Já o RVM é baseado em uma formulação bayesiana (Cervantes *et al*., 2020), que fornece probabilidade a posteriori, porém cada classificador tem seu uso específico. Para este estudo é realizada a comparação entre esses algoritmos de classificação. A escolha do SVM como modelo de comparação se justifica pelo fato de o RVM ter sido proposto originalmente como uma extensão bayesiana do SVM, mantendo uma estrutura de modelo semelhante, baseada em funções kernel e na representação esparsa dos dados.

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 foram utilizados dois conjuntos de dados gerados através da biblioteca *numpy*[1], nativa 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.

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

[1] `https://numpy.org/`

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.

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)
```

2 Background e Trabalhos Relacionados

2.1 Background

2.1.1 Conjunto de Dados 1

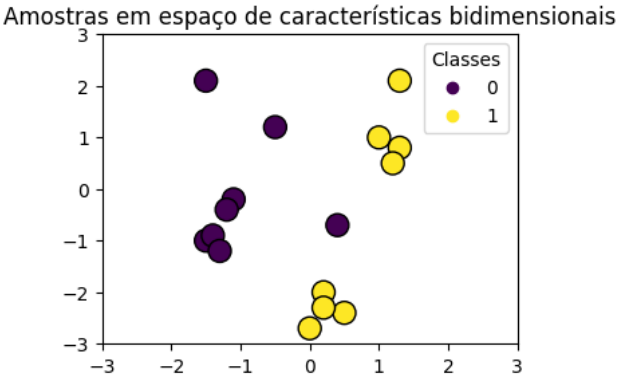
Como mostrados na listagem 1, o conjunto de dados 1 foi gerado de forma previamente definida. Onde os dados são formados por um vetor de 16 (dezesseis) posições. Na Figura 1 podemos ver a distribuição dos valores de X em relação a y. Nota-se que, os dados apesar de simples, estão com uma boa distribuição. Outrossim, se pode perceber que os valores de X_0 (0.4, -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.

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. A listagem 2 gera o conjunto de dados 2.

2.1.2 Kernels

Para os testes nos modelos SVM e RVM foram utilzados 04 kernels, sendo eles, kernel RBF, polinomial, sigmoid e linear. Esses quatro kernels foram escolhidos porque cobrem os principais comportamentos de separação linear e não linear usados em SVM/RVM, e permitem uma comparação abrangente. A seguir veremos como funciona cada um deles.

Figura 1. Amostra dos Dados do Conjunto 1.



Kernel RBF 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*[2]. 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:

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

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:

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

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 inclusa. 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:

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

onde o coeficiente do kernel γ controla a influência de cada amostra de treinamento individual na fronteira de decisão

[2] `https://scikit-learn.org/`

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.

O uso de kernels em SVM e RVM baseia-se no chamado *truque do kernel* (*kernel trick*), que consiste em mapear implicitamente os dados de entrada $x \in \mathbb{R}^n$ para um espaço de características de maior (ou infinita) dimensão, por meio de uma função $\phi(x)$, no qual os dados tornam-se linearmente separáveis. Em vez de calcular explicitamente esse mapeamento, o kernel permite computar diretamente o produto interno nesse espaço transformado, isto é,

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle. \tag{5}$$

Dessa forma, algoritmos originalmente lineares podem ser aplicados a problemas não lineares sem o custo computacional associado à transformação explícita dos dados. Tanto o SVM quanto o RVM exploram esse princípio, sendo que no SVM o kernel é utilizado na formulação do problema de otimização convexa, enquanto no RVM ele é incorporado à função de base do modelo probabilístico bayesiano (Tipping, 2001). Assim, a escolha do kernel influencia diretamente a forma da fronteira de decisão e a capacidade de generalização dos modelos.

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 (Tipping, 2001). Mostra que o funcionamento do RVM é semelhante ao SVM, porém, fundamentado em um modelo bayesiano, assim como em Cervantes *et al*., 2020. 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 (Tzikas *et al*., 2006). Ele mostra que, o RVM é um classificador bayesiano (Cervantes *et al*., 2020), que induz esparsidade no peso dos modelos (Tzikas *et al*., 2006), 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 quando comparado ao SVM.

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 (Rafi and Shaikh, 2013). Apesar da 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 sendo maior que o SVM, a abordagem bayesiana permite maior generalização, como podemos ver em Tzikas *et al*., 2006.

2.2.4 Máquinas de Vetores de Relevância: Uma Introdução

O objetivo deste artigo é apresentar uma introdução ao RVM (Koruk, 2021). 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 Rafi and Shaikh, 2013. Contudo, algumas desvantagens também foram notadas aqui, como por exemplo, o alto custo computacional, estando assim de acordo com Rafi and Shaikh, 2013.

3 Modelo de Solução

3.1 SVM

A técnica de classificação SVM é uma técnica mais moderna, podendo ser entendida como uma evolução do perceptron. Diferentemente deste, o SVM não busca apenas um hiperplano que separe os dados em duas classes, mas sim aquele que maximiza a margem de separação entre elas (Cortes and Vapnik, 1995; Cervantes *et al*., 2020). Formalmente, o hiperplano de decisão é definido por

$$w \cdot x + b = 0 \tag{6}$$

onde w é o vetor normal ao hiperplano e b é o termo de viés (Cortes and Vapnik, 1995).

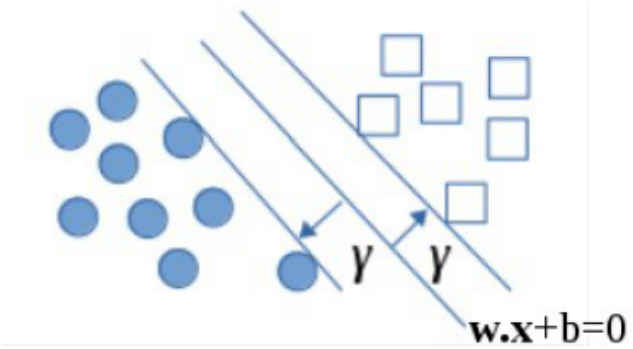
A maximização da margem é obtida pela definição de dois hiperplanos paralelos, dados por

$$w \cdot x + b = +1 \quad \text{e} \quad w \cdot x + b = -1, \tag{7}$$

que delimitam a região de separação entre as classes. Os pontos que pertencem a esses hiperplanos são denominados *vetores de suporte*, pois são as amostras que efetivamente determinam a posição do hiperplano ótimo. O hiperplano central $w \cdot x + b = 0$ é aquele que maximiza a distância entre os hiperplanos de margem, distância esta denotada por γ, conforme ilustrado na Figura 2.

Caso existam amostras localizadas dentro da margem ou mal classificadas, o SVM permite a introdução de variáveis de folga (*slack variables*), possibilitando uma separação mais robusta em cenários onde os dados não são perfeitamente separáveis.

Figura 2. SVM Seleciona Maximizando a Distância γ entre os hiperplanos



Source: Autores

3.1.1 SVC

Conforme a documentação do *scikit-learn* (sickit learn, d(, a implementação é baseada em *libsvm*[3]. 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 SVM 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.

3.2 RVM

Conforme a documentação do *scikit-learn* (sickit learn, b), 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 (Tipping, 2001; Dostál, 2025).

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

$$p(t|x, w, \beta) = N(t|y(x), \beta^{-1}) \tag{8}$$

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

3.2.1 EMRVC

O classificador de vetores de relevância de maximização de expectativa, é segundo a documentação (sickit learn, a), "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 (sickit learn, a).

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 todos os trabalhos referenciados anteriormente na seção 2.2.

[3] `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

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 está o notebook 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.

A reescrita seguiu o tutorial descrito na documentação do RVM, disposta em sickit learn, c. A base do código é apresentado na listagem 3.

Listing 3: Base do Código RVM para o Conjunto de Dados 1

```
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 realizou-se os 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 que está na listagem 4.

Listing 4: Base do Código RVM para o Conjunto de Dados 2

```
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 Figuras 3, 4, 5 e 6 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.

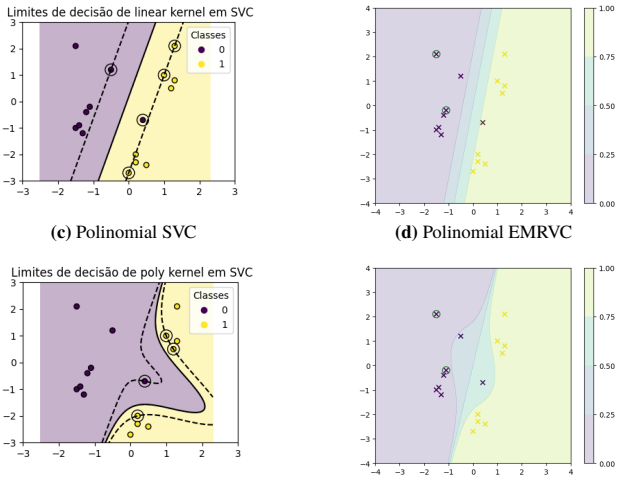
As Figuras 3a e 3b mostram a comparação do resultado de classificação do SVC e do EMRVC utilizando o kernel linear. Podemos notar que, a fronteira de decisão é uma linha reta, por esta razão é linear.

Tanto na Figura 3a quanto na Figura 3b, podemos notar que a classificação ocorre relativamente bem, embora haja um ponto na classe 0 que não foi classificado da forma correta. Outro ponto que é notável e que está presente nos resultados representados pelas Figuras 3, 4, 5 e 6 é o número de vetores relevantes utilizados por cada classificador, enquanto no SVM com SVC foram destacados 05 vetores, sendo 03 na classe 1 e 02 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 Figuras 3c e 3d, demonstra-se a fronteira de decisão do SVC e EMRVC com kernel polinomial. A fronteira de decisão na subfigura 3c 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.

Figura 3. Comparação Utilizando os Kernels Linear e Polinomial.

(a) Linear SVC **(b)** Linear EMRVC

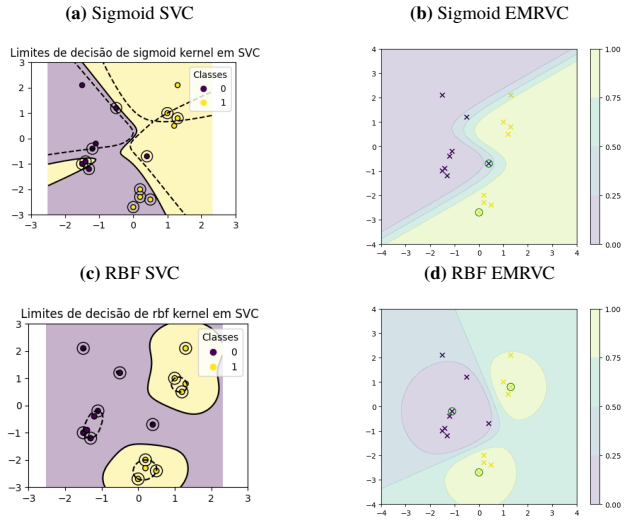


(c) Polinomial SVC **(d)** Polinomial EMRVC

Na Figura 4 é representado os resultados obtidos no SVM e RVM utilizando os kernels sigmoid e RBF. Podemos notar que na Figura 4a, o SVC foi piorando a sua classificação, pois cometeu mais erros, uma vez que 03 pontos da classe 0 foram classificados como classe 1 e 04 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 errôneamente, isto é, 6 dos 7 pontos. Também houveram muitos vetores escolhidos como relevantes. Em contraposição a isto, o EMRVC, Figura 4b, utilizou apenas 02 vetores de relevância, dessa forma ele conseguiu classificar corretamente 16/16 pontos entre as classes 0 e 1. Sendo assim, no kernel sigmoid com RVM foi melhor no quesito classificação.

Analisando a Figura 4c, 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 Figura 4d 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 (Rafi and Shaikh, 2013). Neste caso também não houve erro de classificação, sendo 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.

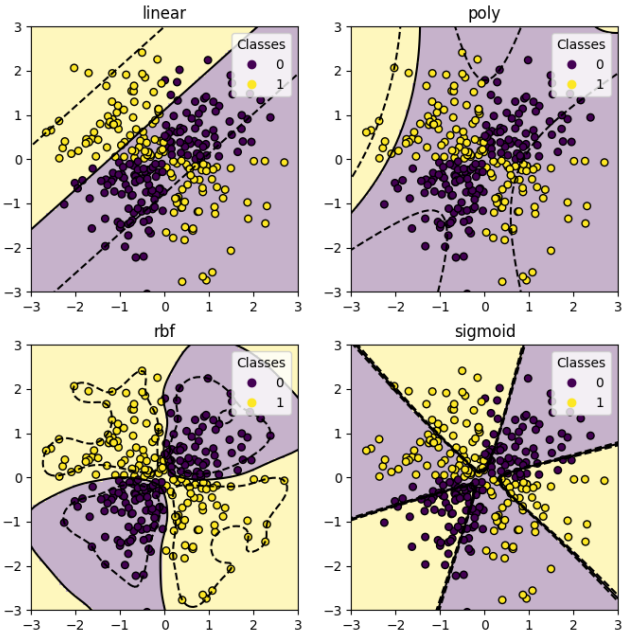
Figura 4. Comparação Utilizando os Kernels Sigmoid e RBF.



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 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 5. Resultado SVM com Dados XOR.



5.3 Comparando Métricas

Com o objetivo de complementar a análise visual das fronteiras de decisão, foram utilizadas métricas quantitativas de desempenho para avaliar os classificadores SVM (com SVC) e RVM (com EMRVC) no conjunto de dados XOR, conforme a

Figura 6. Resultado RVM com Dados XOR.

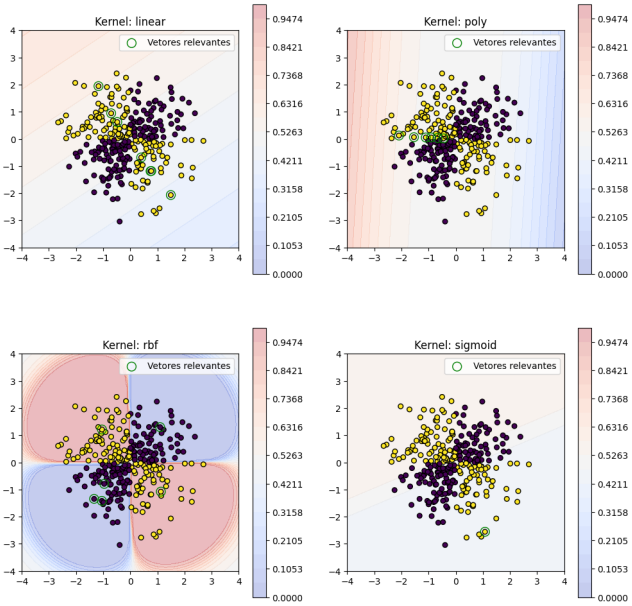


Figura 7. As métricas consideradas foram acurácia, precisão, recall e F1-score, amplamente empregadas na avaliação de modelos de classificação binária.

A acurácia mede a proporção total de predições corretas em relação ao número total de amostras. A precisão indica a proporção de amostras corretamente classificadas como positivas dentre todas aquelas preditas como positivas. O recall avalia a capacidade do modelo em identificar corretamente as amostras da classe positiva, enquanto o F1-score representa a média harmônica entre precisão e recall, sendo particularmente útil em cenários onde há interesse em equilibrar ambas as métricas.

Para garantir consistência experimental, os modelos foram treinados e avaliados sobre o mesmo conjunto de dados, uma vez que o objetivo desta análise é comparativo, e não a estimativa de desempenho generalizado. As métricas foram calculadas utilizando ponderação por classe (*weighted*), de modo a considerar possíveis desequilíbrios na distribuição das classes.

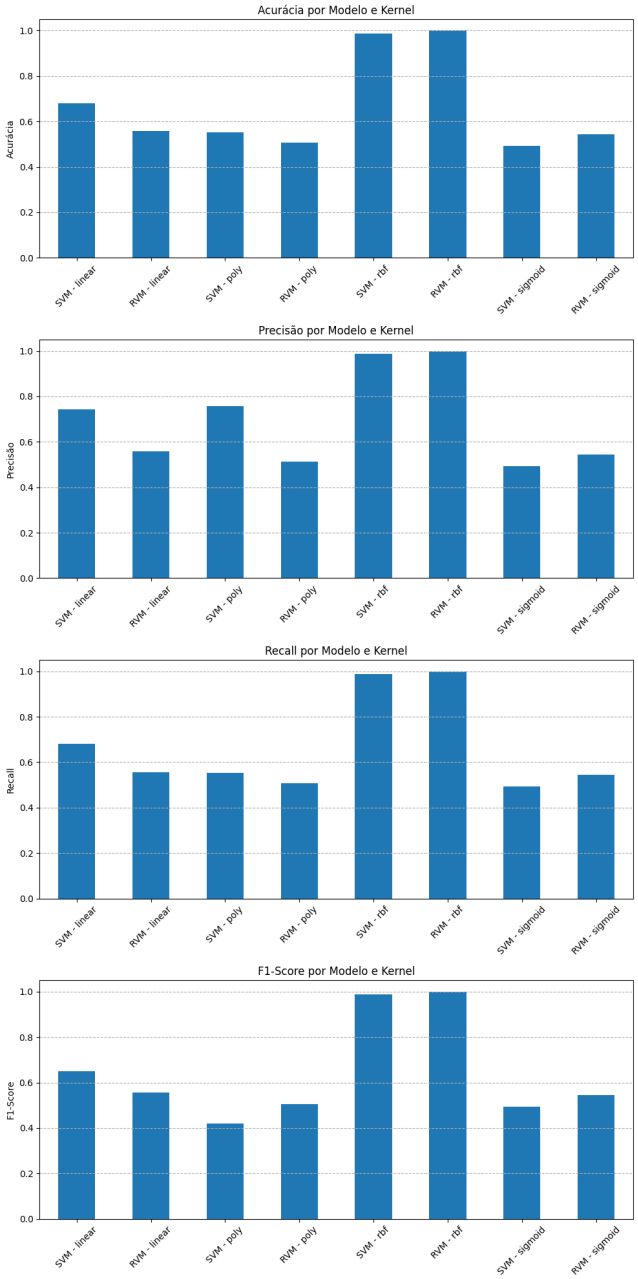
Os resultados obtidos indicam que os kernels lineares e polinomiais apresentaram desempenho intermediário para ambos os modelos, com o SVM obtendo métricas ligeiramente superiores em alguns casos, especialmente no kernel polinomial. Em contraste, o kernel sigmoide apresentou desempenho inferior de forma consistente, tanto para SVM quanto para RVM, refletindo sua menor adequação ao padrão não linear do problema XOR.

O melhor desempenho foi observado com o uso do kernel RBF, no qual ambos os modelos alcançaram valores próximos ou iguais a 1.0 para todas as métricas analisadas. Destaca-se que o RVM, mesmo apresentando desempenho equivalente ao SVM em termos de acurácia, precisão, recall e F1-score, manteve sua característica de esparsidade, utilizando um número reduzido de vetores de relevância, conforme discutido nas análises visuais das fronteiras de decisão.

Esses resultados corroboram a literatura, que aponta o kernel RBF como particularmente eficaz para problemas não linearmente separáveis, como o conjunto de dados XOR, e

reforçam que o RVM pode alcançar desempenho comparável ao SVM, com a vantagem adicional de uma formulação probabilística e modelos mais esparsos.

Figura 7. Métricas.



6 Conclusão

6.1 Contribuição

Ao findar da pesquisa 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

Vantagens do SVM: 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".

Desvantagens do SVM: 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

Vantagens do RVM: 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.

Desvantagens do RVM: 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 Limitações e Sugestões para Estudos Futuros

Algumas limitações na produção deste artigo embora se tenha testado 04 kernels, outros kernels poderia ter sido explorados, mas nos detivemos ao proposto.

Para estudos futuros pode-se 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, pode-se explorar novos kernels e ver como os algoritmos SVM e RVM se comportariam.

Declarações complementares

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.

Contribuições dos autores

JF contribuiu para a concepção deste estudo (Conceptualização). LA contribuiu para o desenvolvimento de software e visualização (ajudou com códigos e imagens de resultados). GD validou o trabalho. JC contribuiu com a supervisão. JF é o principal contribuidor e escritor deste manuscrito. Todos os autores leram e aprovaram o manuscrito final.

Conflitos de interesse

Os autores declaram que não têm nenhum conflito de interesses.

Disponibilidade de dados e materiais

Os conjuntos de dados (e/ou softwares) gerados e/ou analisados durante o estudo atual estão disponíveis em …'https://github.com/jonatasfernandessilva7/Trabalho-04—Reconhecimento-de-Padroes.git'.

Referências

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. DOI: https://doi.org/10.1016/j.neucom.2019.10.118.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297. DOI: 10.1007/BF00994018.

Dostál, Z. (2025). Support vector machines. In *Optimal Quadratic Programming and QCQP Algorithms with Applications*, pages 339–347. Springer. DOI: 10.1007/978-3-031-95167-1_16.

Koruk, K. (2021). Relevance vector machines: An introduction. *Geomet Queens University*.

Murel, J. and Kavlakoglu, E. (2025). O que são modelos de classificação? Online; Acesso em 2025-06-20.

Rafi, M. and Shaikh, M. S. (2013). A comparison of svm and rvm for document classification. DOI: `https://doi.org/10.48550/arXiv.1301.2785`.

sickit learn. Emrvc - documentação scikit-learn. `https://sklearn-rvm.readthedocs.io/en/latest/generated/sklearn_rvm.em_rvm.EMRVC.html`. Online; Acesso em 2025-6-19.

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

sickit learn. Rvm for classification - documentação scikit-learn. `https://sklearn-rvm.readthedocs.io/en/latest/auto_examples/plot_rvm_for_classification.html`. Online; Acesso em 2025-6-19.

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

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244.

Tzikas, D., Wei, L., Likas, A., and Yang, Y. (2006). A tutorial on relevance vector machines for regression and classification with applications. *Eurosip*, 17.