Avaliação Support Vector Machines

Paulo Roberto Farah
Universidade Federal do Paraná (UFPR)
Programa de Pós-Graduação em Informática (PPGInf)
INFO 7004 – Aprendizagem de Máquina

I. Introdução

Support Vector Machine (**SVM**) é um conjunto de métodos de aprendizado supervisionado que têm como objetivo solucionar problemas de reconhecimento de padrões. Ele identifica uma linha, denominada hiperplano, para separar classes distintas. Sua função-objetivo utiliza vetores de suporte para maximizar a distância entre os pontos mais próximos das diferentes classes encontradas.

O SVM usa o recurso de *kernel trick* para reconhecer padrões em problemas que não são linearmente separáveis. Por meio de funções, é capaz de mapear um vetor de características de entrada de baixa dimensionalidade para um espaço de alta dimensionalidade e construir um hiperplano de separação ótimo neste espaço. Assim, a função projeta um problema não separável para outra dimensão na qual possa encontrar uma fronteira de decisão entre as classes. As funções podem ser lineares, polinomiais, radiais (RBF), sigmoidais, entre outras.

As várias funções de kernel levam ao trabalho de escolher adequadamente a função que melhor identifica os padrões tanto para aprendizagem do modelo quanto para identificar dados da base de teste. Assim, surge a questão de qual função de kernel escolher para cada caso.

Nesse contexto, o objetivo desse trabalho é avaliar o método SVM sob duas perspectivas. Primeiro, analisar o comportamento das funções de kernel linear e RBF ao classificarem dois conjuntos de dados de treinamento, sendo um linearmente separável e outro não. Segundo, comparar o desempenho do classificador SVM com kernel linear e RBF com os classificadores kNN, Naïves Bayes, LDA e Logistic Regression.

II. Metodologia

A metodologia para a execução dessa atividade incluiu o planejamento dos experimentos, uma descrição da implementação e as instruções de instalação e execução do programa.

A. Planejamento

Na primeira etapa, foi utilizado o programa svmtoy.py para gerar dados de treinamento e analisar o comportamento do uso dos kernels linear e RBF. A função *make_blobs* gerou dados linearmente separáveis em alguns casos e a função *make_gaussian_quantiles* gerou classes não separáveis linearmente. Foram realizadas 30 execuções distintas dos classificadores para cada função de kernel e coletados os resultados de acurácia, tempo de execução, matriz de confusão e quantidade de vetores de suporte. Além disso, as respectivas médias desses resultados foram calculadas.

Na segunda etapa, foi utilizado o programa svm.py para classificar a base *digTrain20k.txt*, composta por 20000 exemplos. Foi utilizado 50% da base para aprendizagem e validação e os 50% restantes para teste. Foi utilizada a função *GridSearch* para buscar os parâmetros C e gamma do método SVM com RBF. Para o SVM linear, o parâmetro C recebeu valor 1. O algoritmo kNN foi executado com os parâmetros k igual a 9, métrica *euclidean* e *weight distance*. Foi calculada a acurácia, matriz de confusão, tempo de treinamento e de execução dos seis classificadores. Além disso, foi registrado o tempo de busca de parâmetros C e gamma do SVM com RBF e a quantidade de vetores de suporte para os classificadores SVM.

Os experimentos foram executados em um computador com processador Intel i7 8^{th} geração, com 16GB de memória e sistema operacional Windows 10. Foi utilizado python 2.7 e IDE PyCharm para o desenvolvimento.

B. Implementação

O programa svmtoy.py foi modificado para executar 30 vezes, calcular análise estatística contendo maior e menor valores, média, mediana e desvio padrão. Além disso, o programa grava os resultados nos arquivos svm_linear.csv e svm_rbf.csv.

O programa svm.py foi alterado para ler apenas uma entrada e dividir a base em 50% para treinamento e 50% para teste. Foram adicionados os métodos kNN, Naïve Bayes, LDA e Logistic Regression. Além disso, estão sendo coletados os tempos de treinamento, de busca de parâmetros e total de execução para os métodos, acurácias e vetores de suporte e salvos no arquivo svm_results.csv.

C. Instalação e Execução

A instalação dos pacotes necessários a execução do classificador pode ser feita com os seguintes comandos, descritos a seguir.

```
unzip svm.zip
cd svm
virtualenv venv
source venv/bin/activate
pip install -r requirements.txt
```

Para executar o programa:

```
python svm.py <treino> <teste>
```

III. Resultados

Os resultados são descritos e discutidos nesta seção. São mostrados os resultados da análise da base de treinamento gerada pelas funções *make_blobs* e *make_gaussian_quantiles*. Em seguida, estão descritos os resultados da comparação do desempenho desses métodos com os classificadores kNN, Naïve Bayes, LDA e Logistic Regression.

A. Análise da base de treino das funções make_blobs *e* make_gaussian_quantiles

Nesta seção estão descritos os resultados da quantidade de vetores de suporte, acurácia e suas respectivas análises estatísticas.

1) Vetores de Suporte: As Figuras 1 e 2 mostram as quantidades de vetores de suporte para as bases de aprendizagem *make_blobs* e *make_gaussian_quantiles*. No primeiro caso, aonde a base pode ter classes linearmente separáveis, a função de kernel linear apresentou uma quantidade significativamente menor do que a função RBF. Já na base gaussiana, a situação inverte e a função linear fica próxima de 300 vetores, enquanto a radial abaixo de 50.

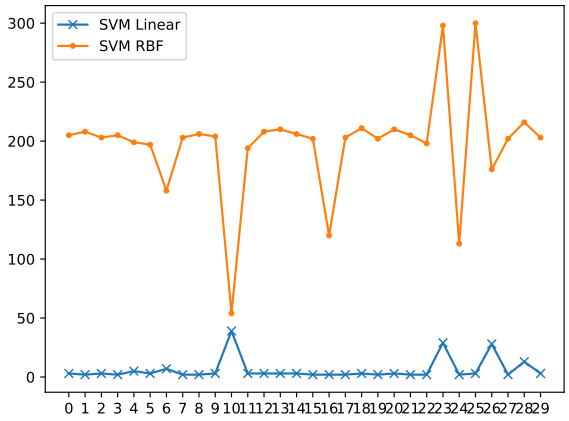


Figura 1: Quantidade de vetores de suporte do classificador SVM com kernel linear e rbf para base de dados gerada pela função *make_blobs* em 30 execuções.

A análise estatística das quantidades dos vetores de suporte é apresentada na Tabela I. Também pode-se observar que para a base gerada pela função *make_blobs*, a função linear apresenta em média uma quantidade menor de vetores de suporte e muito maior para a base *make_gaussian_quantiles* em relação à rbf.

2) Acurácia: As Figuras 3 e 4 apresentam os valores da métrica *acurácia* das funções de kernel linear e BRF do SVM para as bases de aprendizagem *make_blobs*, possivelmente separável linearmente, e *make_gaussian_quantiles*, não separável linearmente. Além disso, a Tabela II descreve a análise estatística dos métodos em cada base. Pode-se observar que para a primeira base, a acurácia foi muito próxima para ambas as funções de kernel. Já para a segunda, o kernel RBF apresentou um desempenho muito substancial, em relação à

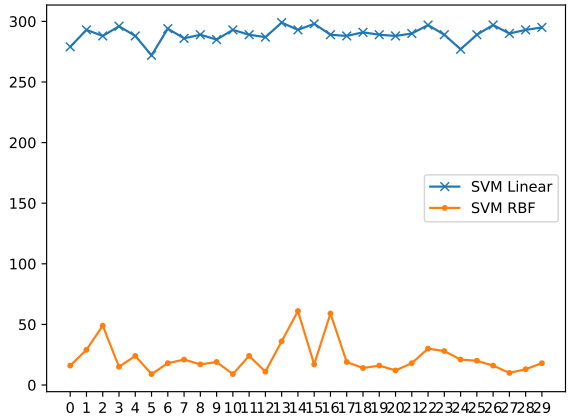


Figura 2: Quantidade de vetores de suporte do classificador SVM com kernel linear e rbf para base de dados gerada pela função *make_gaussian_quantiles* em 30 execuções.

Tabela I: Análise estatística dos vetores de suporte da base *digTrain20k*.

make_blobs	*SVM Linear*	*SVM RBF*
Maior	39	300
Menor	2	54
Média	6,03	197,3
Mediana	3,0	203
Desvio Padrão	9,04	43,96
make_gaussian_quantiles	*SVM Linear*	*SVM RBF*
Maior	299	61
Menor	272	9
Média	289,7	22,3
Mediana	289	18
Desvio Padrão	5,8886	13,0336

função linear. A média da acurácia foi de $0,9961$ nesse caso.

B. Avaliação de Desempenho utilizando a base de teste digTrain20k

Nesta seção são apresentados os resultados obtidos para a avaliação de desempenho do SVMs linear e rbf para a base de teste *digTrain20k*. A acurácia foi comparada com outros

Tabela II: Análise estatística da acurácia da base de treino.

make_blobs	*SVM Linear*	*SVM RBF*
Maior	1	1
Menor	0,9466	0,95
Média	0,9955	0,9957
Mediana	1	1
Desvio Padrão	0,0123	0,0116
make_gaussian_quantiles	*SVM Linear*	*SVM RBF*
Maior	0.6733	1
Menor	0.59	0.98
Média	0.6288	0.9961
Mediana	0.6283	0.9966
Desvio Padrão	0.0237	0.0046

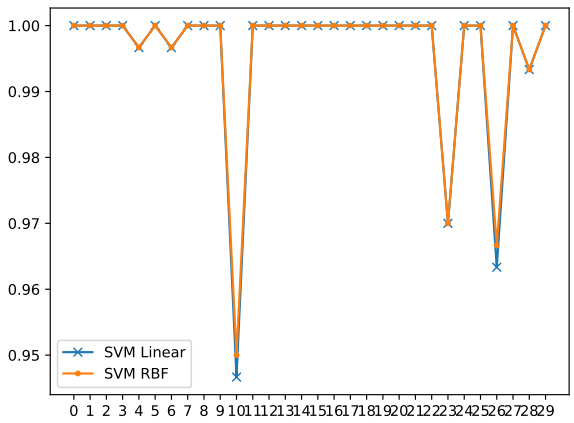


Figura 3: Acurácia do classificador SVM com kernel linear e rbf para base de dados gerada pela função *make_blobs* em 30 execuções.

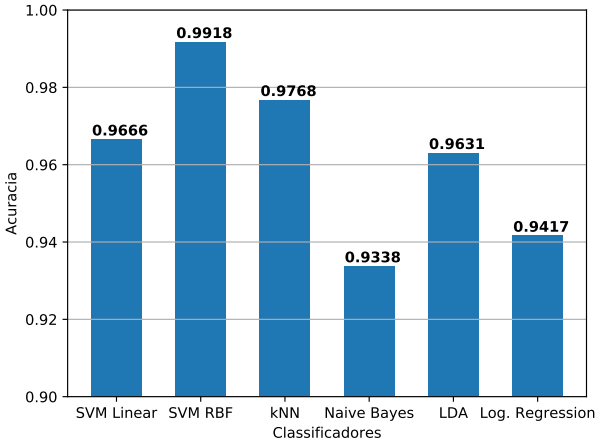


Figura 5: Histograma de acurácia dos classificadores SVM linear e RBF, kNN, Naive Bayes, LDA e Logistic Regression para a base *digTrain20k*.

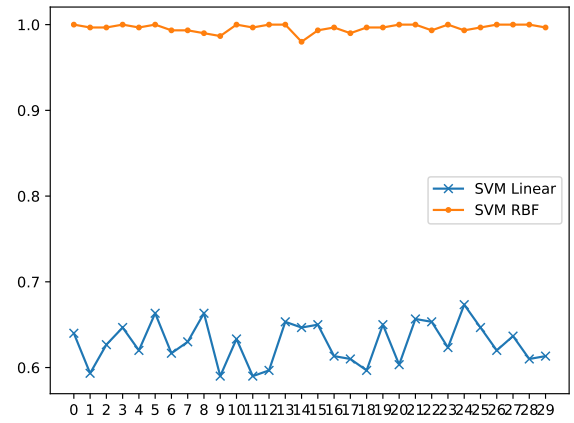


Figura 4: Acurácia do classificador SVM com kernel linear e rbf para base de dados gerada pela função *make_gaussian_quantiles* em 30 execuções.

quatro classificadores: kNN, Naïve Bayes, LDA e Logistic Regression. Em seguida, são mostrados os resultados do tempo de busca de parâmetros do método SVM RBF, os tempos de aprendizagem dos seis classificadores e o tempo total de execução. Além disso, são mostrados a quantidade de vetores de suporte das duas funções de kernel do SVM e, por fim, as matrizes de confusão dos seis classificadores.

1) Acurácia: A Figura 5 mostra as acurácias obtidas para os classificadores SVM linear, SVM RBF, kNN, Naïve Bayes, LDA e Logistic Regression. Pode-se observar que o melhor resultado apresentado foi para o método SVM com a função de kernel RBF, atingindo 0,9918. Em seguida, ficaram o kNN, SVM linear, LDA, Logistic Regression e Naïve Bayes.

2) Tempos: A Tabela III apresenta o tempo de busca de parâmetros, treino e execução dos classificadores. O algoritmo naïve bayes apresentou o menor tempo de treinamento, em contrapartida, o SVM RBF apresentou um tempo muito maior para realizar a classificação. Em geral, os classificadores SVM levaram mais tempo para treinar. Observa-se que o tempo de busca de parâmetros para SVM RBF é muito alto. Esse tempo foi coletado durante a chamada da função *GridSearch*. Os parâmetros encontrados foram $C = 512$ e $\gamma = 0,5$.

Tabela III: Comparação do tempo de execução dos classificadores para conjunto de dados de teste *digTrain20k*.

Classificador	**Busca**(h)	**Treino** (s)	**Total**
SVM linear		3.877	0:00:14.237
SVM RBF	1:12:16.019	8.554	1:12:29.620
kNN		2.881	0:00:03.193
Naïve Bayes		0.153	0:00:40.119
LDA		0.291	0:00:01.062
Logistic Regression		0.928	0:00:01.251

3) Vetores de Suporte: A quantidadae de vetores de suporte para as funções de kernel linear e RBF está na Figura 6. Observa-se que a função linear necessita de 614 vetores, uma quantidade bem maior de vetores de suporte para definir a fronteira de decisão entre as classes do que a função RBF que utilizou 194.

C. Matrizes de Confusão

Observa-se, a partir da Figura 7, a quantidade de verdadeiros e falsos positivos e negativos para cada um dos seis métodos de classificação utilizados. Deste modo, caracteriza-se a diagonal principal como os acertos dos respectivos métodos. Além disso, nota-se que a quantidade de acertos na diagonal principal é maior para o método SVM RBF. Falsos-positivos e falsos-negativos foram encontrados em todos os classificadores avaliados. Destaca-se que para os classificadores SVM linear

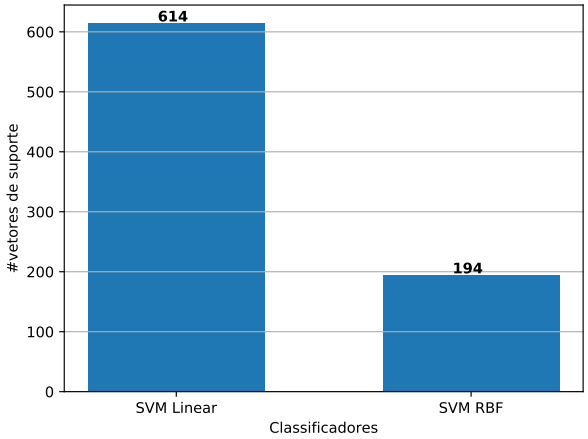
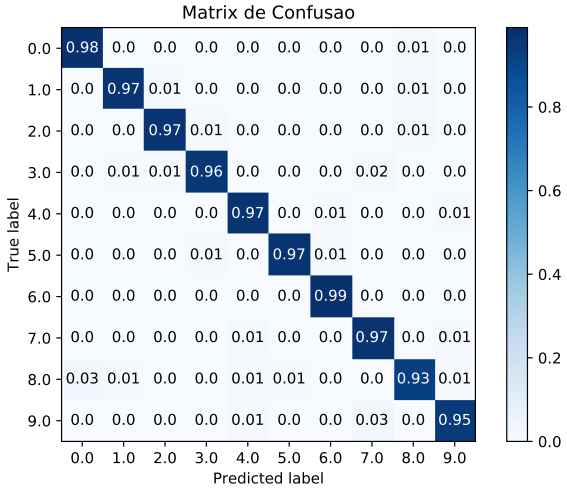


Figura 6: Quantidade de vetores de suporte para conjunto de dados de teste *digTrain20k*.

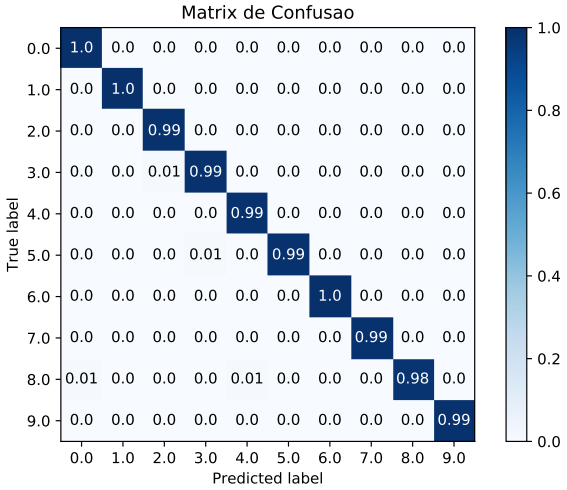
e RBF, LDA e Logistic Regression, a classe 8 apresentou mais falsos positivos e negativos. No kNN, varias classes empataram com maior taxa de erro e para o Naïve Bayes a classe 2 apresentou mais erros de classificação.

IV. Conclusão

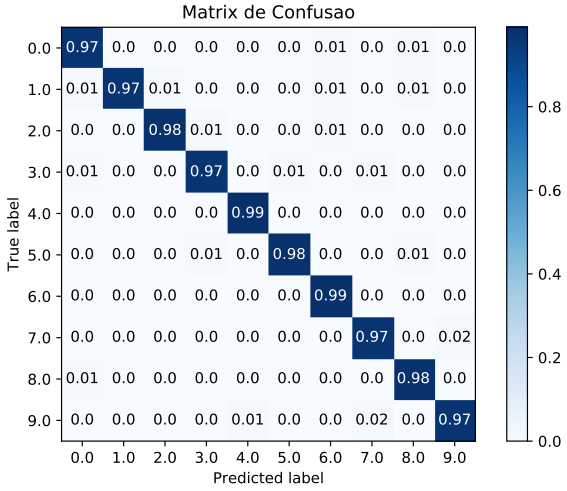
O presente trabalho teve como objetivo principal a avaliação do desempenho das de kernel linear e RBF do método SVM. A função RBF apresentou melhor acurácia em todos os resultados. Contudo, para a base de aprendizagem gerada pela função *make_blobs*, a função RBF demonstrou quantidade de vetores de suporte muito maior em relação a função linear e o tempo de treinamento também foi maior. Assim, conclui-se que o SVM RBF apresentou desempenho superior em relação ao SVM linear para esses conjuntos de dados.



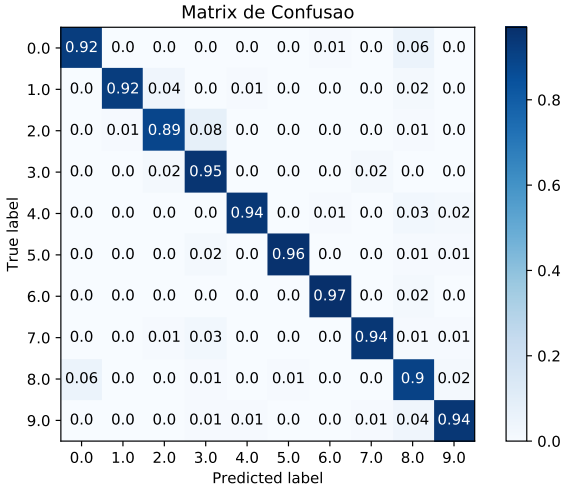
(a) SVM linear.



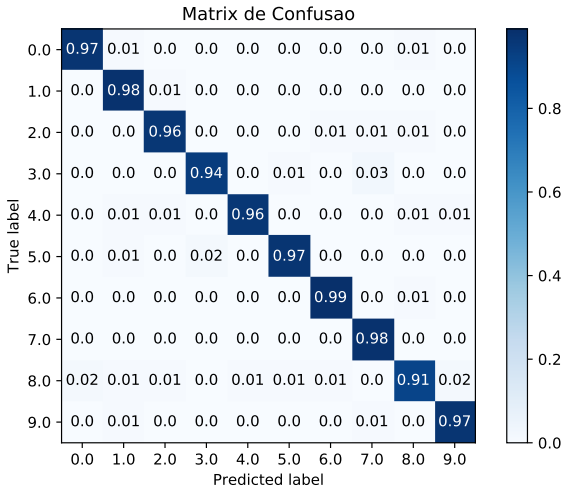
(b) SVM RBF.



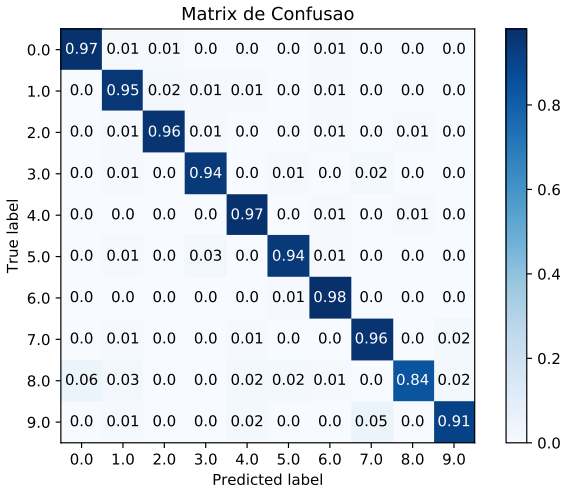
(c) kNN.



(d) Naïve Bayes.



(e) LDA.



(f) Logistic Regression.

Figura 7: Matrizes de confusão para conjunto de dados de teste *digTrain20k*.