Aprendizado de Máquina para Diagnóstico de Câncer de Mama

Marcos Paulo Diniz
Universidade de Brasília
Departamento de Ciência da Computação
Brasília, Brasil
marcosdiniz@aluno.unb.br

***Resumo*—O trabalho busca mensurar a capacidade de diagnóstico do câncer de mama por meio de aprendizagem de máquina. Para isso, foi implementado o algoritmo do SVM usando diferentes kernels em busca de um resuldado melhor, além de, também, usar valores de C e $gamma$ diferentes.**

***Index Terms*—Câncer, diagnóstico, aprendizagem de máquina, algoritmo, SVM.**

I. Introdução

Esse trabalho tem como objetivo usar o aprendizado de máquina supervisionado, por meio do algoritmo *Support Vector Machine* (SVM), para diagnosticar a presença ou não do câncer da mama. No *dataset* utilizado haviam 569 dados de diagnósticos da doença, entre diagnósticos malignos e benignos.

Para a implementção do algoritmo foi-se usado a linguagem de programaçãao Python (versão 3.6.3), com auxílio das bibliotecas *Sklearn, Matplotlib, Pandas e Numpy*.

Por se tratar de um algoritmo de aprendizado de maquina supervisionado, será preciso separar a base em duas partes, sendo uma para treinamento e a outra para testes, essa separação será de 70% e 30%, respectivamente.

Para o bom compreendimento do trabalho realizado, é preciso, primeiramente, entender como o SVM funciona. O algoritmo do SVM mapeia os dados para um espaço dimensional variavel a fim de categorizar os dados, sendo a disposição destes linear ou nao. O algoritmo cria um separador entre as categorias e, depois, transforma os dados de modo a permitir o separador ser desenhado como um hiperplano. Com isso, novos dados podem ser inseridos para serem classificados.

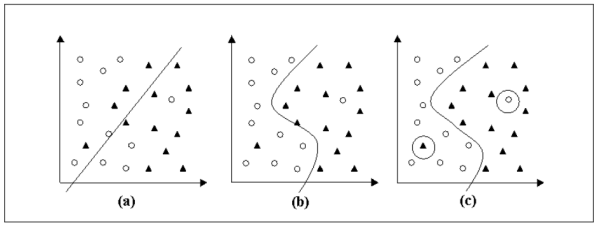

Figura 1. SVM para diferentes kernels

Para separar os dados do conjunto o SVM precisa traçar uma linha para dividir o conjunto, no nosso caso, precisa dividir os dignóstivos malignos e benignos. A forma desse segmento de reta dependerá de qual kernel está sendo utilizado, nesse trabalho usaremos os seguintes kernels: linear, gaussiano e RBF. Na imagem acima pode ser notado o comportamento desse segmento para diferentes *kernels*.

Como pode ser observado na imagem, podemos ter um kernel bem simples como uma reta, como o observado em (a) da figura anterior, mas podemos chegar à um kernel mais próximo do ideal, como observado em (c), que por sua vez separa os dados de uma forma mais precisa.

II. Analise do Experimento

Para a realização do experimento a primeira etapa foi carregar o dataset e entender como os dados estavam distribuidos nele. Após isso, separamos os conjuntos de trainamento e teste, conforme explicado anteriormente.

Após entender os dados disponíveis no dataset, começamos a aplicação do kernel linear. Nesse primeiro momento, aplicamos o kernel linar padrão, isto é, sem nenhuma modificação dos parâmetros da função, e obtivemos a matriz de confusão abaixo.

Tabela I
Matriz de Confusão para o Kernel Linear

	Maligno	Benigno
Maligno	63	0
Benigno	58	50

E, com isso, foi obtido uma acurácia de 66%, mas como pode ser observado na tabela acima, o modelo acertou todos os diagnósticos quando o câncer é maligno, porém quando o dignóstico era benigno a taxa de acerto foi de pouco mais de 45%, ficando bem abaixo do esperado.

Para melhorar o algoritmo em busca de um resultado melhor, aplicamos o *GridSearchCV* combinando diferentes valores de C, em busca de uma melhora na texa de acerto do algoritmo. Na tabela abaixo estão os valores qua foram usados para o parâmetro C.

Tabela II
Valores de C

C:	0.01	0.1	1	10	100	1000	10000	100000

Após aplicar a técnica apresentada acima, o algoritmo retornou que o valor ideal para o parametro seria usar $C = 1000$. Com isso, alteramos o valor padrão de C, usando o indicado anteriormente. Após a execução com o novo parâmetro, obtivemos a matriz de confusão abaixo.

Tabela III
MATRIZ DE CONFUSÃO PARA O KERNEL LINEAR COM $C = 1000$

	Maligno	Benigno
Maligno	58	5
Benigno	8	100

Como pode ser observado na tabela acima, a taxa de acerto melhorou significantemente, após se alterar o valor de C, alcançando uma taxa de 93% de acerto, ainda que tenha uma pequena queda no dignóstico do câncer benigno, que antes não houve nenhuma classificação incorreta.

A melhora condicionada ao valor de C se deve a infuência desse parametro sobre a linha que dividirá o hiperplano do SVM. Para valores de C muito grandes, a otimização escolherá um hiperplano de margem menor se esse hiperplano fizer um trabalho melhor ao classificar todos os pontos de treinamento corretamente. Por outro lado, um valor muito pequeno de C fará com que o otimizador procure por um hiperplano de margem de separação maior, mesmo que esse hiperplano atribua erros a mais pontos.

Sabendo disso, ao analisar as Tabelas I e II em conjunto, percebemos claramente a infuência do valor C grande, que possibilitou a crescente no acerto do dignóstico benigno e, como consequência desse aumento, houve erro de 5 dignósticos do câncer maligno, que não foi observado quando usamos o valor de C padrão.

A seguir, temos um gráfico que relaciona todos as taxas de acerto obtidas com os diferentes valores de C.

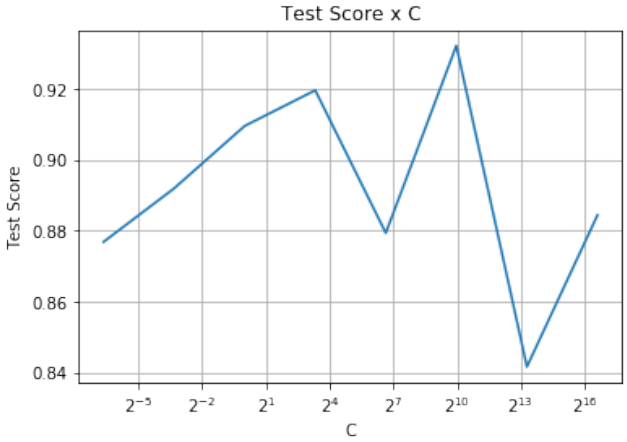

Figura 2. Taxa de acerto de teste x C

Após obter o resultado acima com o kernel linear, passamos a aplicar o kernel Gaussiano. Aplicando o kernel gaussiano sem nenhuma modificação de seus aprametros, obtivemos a matriz de confusão abaixo.

Tabela IV
MATRIZ DE CONFUSÃO PARA O KERNEL GAUSSIANO

	Maligno	Benigno
Maligno	0	63
Benigno	0	108

Novamente o resultado não foi muito satisfatório, alcançando apenas 63% de acerto, considerando que ele acertou todos os dignósticos malignos, mas em contra partida, errou todos os benignos.

Como o resultado foi abaixo do esperado, mnovamente aplicaremos o *GridSearchCV*, porém, dessa vez, iremos testar vários valores de C em conjunto com vários valores de σ. Vale ressaltar que $gamma = 1/\sigma$, sabendo disso, vamos alterar o parâmetro gamma, que por conseguência estaremos alterando o valor de σ.

Na tabela abaixo estão os valores de C e $gamma$ que utilizamos.

Tabela V
VALORES DE C E $gamma$

C:	0.01	0.1	1	10	100	1000	10000	100000
$gamma$:	1000	100	10	1	0.1	0.01	0.001	0.0001

Após aplicar o *GridSearchCV*, foi nos retornado que os melhores valores de C e $gamma$ seriam de 1000 e 0.0001, respectivamente. Sabendo disso, refizemos o treinamento alterando os valores desses parâmetros e obtivemos a matriz de confusão abaixo.

Tabela VI
MATRIZ DE CONFUSÃO PARA O KERNEL GAUSSIANO COM $C = 1000$ E $gamma = 0.0001$

	Maligno	Benigno
Maligno	57	6
Benigno	6	102

Após a alteracão dos parâmetros, foi notával o aumento da taxa de acerto. Isso se deve a influência dos parametros sobre a função, a influência do parametro C, já vimos anteriormente, nesse momento iremos entender a influência do $gamma$ sobre o SVM.

O $gamma$ controla os picos da nossa função gaussiana. Um $gamma$ pequeno nos dá picos mais marcados, sendo esses mais altos e mais pontiagudos, enquanto isso um gama muito grande, conseguimos picos mais amplos (ou seja, sem um pico alto, sim largo e bem mais suave. Ainda outra caracterísca do gamma pequeno é uma alta variância, enquanto um maior é marcado pela baixa variância.

Com isso, ao observar o $gamma$ ideal retornado conforme esperado foi um valor pequeno, em vista da ampla variância dos dados no dataset. E ao analisar os resultados retornados, percebemos que para um mesmo valor de $gamma$ e com o valor de C variando, não havia mudanças na taxa de acerto.

Logo, se conclui que para um valor fixo de C, o valor de $gamma$ não influência na taxa de acerto. A seguir temos um gráfico que mostra a relação da taxa de acerto com o valor de C. Ao lado, podemos ver a tebela completa com as Taxas de Acertos, valores de C e valores de $gamma$.

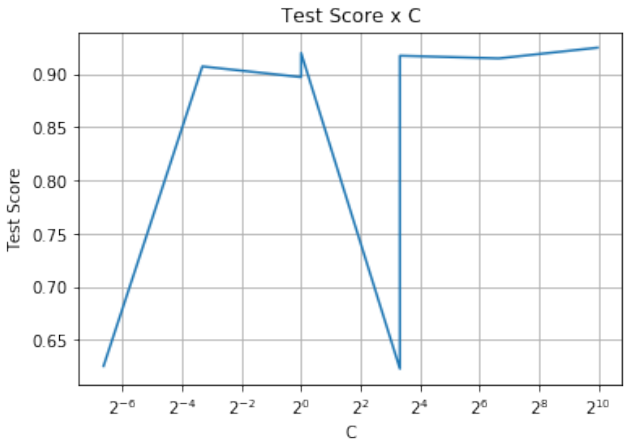

Figura 3. Taxa de acerto de teste x C

Por fim, será treinado o SVM aplicando o kernel RBF, dessa vez, o kernel será aplicado sem nenhuma modificação e faremos, em seguida, a análise dos resultados obtidos. Na tabela abaixo temos a matriz de confusão obtida com a execução.

Tabela VII
MATRIZ DE CONFUSÃO PARA O KERNEL RBF

	Maligno	Benigno
Maligno	47	16
Benigno	0	108

Com o kernel RBF conseguimos uma taxa de acerto por volta dos 90%, valendo ressaltar que o algoritmo teve êxito acima do esperado nos diagnósticos benignos, enquanto no caso dos malignos texe uma taxa de acerto de 75%.

III. CONCLUSÕES

Após a análise de todos os resultados obtidos, é notável que para se conseguir uma boa taxa de acerto com o algoritmo do SVM, não basta apenas aplicá-lo do modo normal dele, isto é, sem alterar seus parâmetros.

Primeiramente, é interessante executar o SVM da sua forma padrão, para em seguida comparar com o SVM aperfeiçoado com mudanças nos valores de seus parâmetros.

Com o SVM linear, a melhora foi bem significativa, após encontrar o valor ideal de C, elevando a taxa de acerto de 66% para 93%, tornando um algoritmo confiável para uso.

O mesmo acontece quando aplicamos o kernel gaussiano sem mudar os parâmetros iniciais. O SVM só consegue mapear os dignósticos para benigno, em todos os casos, o que o torna altamente não confiável, ainda que tenha uma taxa de acerto de 63%.

Tabela VIII
VALORES DE C E $gamma$

Taxa de Acerto de Teste	C	$gamma$
0.6256281407035176	0.01	1000
0.6256281407035176	0.01	100
0.6256281407035176	0.01	10
0.6256281407035176	0.01	1
0.6256281407035176	0.01	0.1
0.6256281407035176	0.01	0.01
0.6256281407035176	0.01	0.001
0.6256281407035176	0.01	0.0001
0.6256281407035176	0.1	1000
0.6256281407035176	0.1	100
0.6256281407035176	0.1	10
0.6256281407035176	0.1	1
0.6256281407035176	0.1	0.1
0.6256281407035176	0.1	0.01
0.6256281407035176	0.1	0.001
0.9070351758793970	0.1	0.0001
0.6256281407035176	1	1000
0.6256281407035176	1	100
0.6256281407035176	1	10
0.6256281407035176	1	1
0.6256281407035176	1	0.1
0.6256281407035176	1	0.01
0.8969849246231156	1	0.001
0.9195979899497487	1	0.0001
0.6256281407035176	10	1000
0.6256281407035176	10	100
0.6256281407035176	10	10
0.6256281407035176	10	1
0.6256281407035176	10	0.1
0.6231155778894473	10	0.01
0.8944723618090452	10	0.001
0.9170854271356784	10	0.0001
0.6256281407035176	100	1000
0.6256281407035176	100	100
0.6256281407035176	100	10
0.6256281407035176	100	1
0.6256281407035176	100	0.1
0.6231155778894473	100	0.01
0.8944723618090452	100	0.001
0.9145728643216080	100	0.0001
0.6256281407035176	1000	1000
0.6256281407035176	1000	100
0.6256281407035176	1000	10
0.6256281407035176	1000	1
0.6256281407035176	1000	0.1
0.6231155778894473	1000	0.01
0.8944723618090452	1000	0.001
0.9246231155778895	1000	0.0001
0.6256281407035176	10000	1000
0.6256281407035176	10000	100
0.6256281407035176	10000	10
0.6256281407035176	10000	1
0.6256281407035176	10000	0.1
0.6231155778894473	10000	0.01
0.8944723618090452	10000	0.001
0.9246231155778895	10000	0.0001
0.6256281407035176	100000	1000
0.6256281407035176	100000	100
0.6256281407035176	100000	10
0.6256281407035176	100000	1
0.6256281407035176	100000	0.1
0.6231155778894473	100000	0.01
0.8944723618090452	100000	0.001
0.9246231155778895	100000	0.0001

Contudo, ao buscar os valores ideais de C e $gamma$, percebemos que o uso do SVM com esse kernel passa a ser notoriamente mais satisfatório. Ainda que ele tenha um resultado pior quando se olha somente os diagnósticos benignos, o resultado foi altamente satisfatório, uma vez que antes o algoritmo não tinha nenhuma credibilidade, uma vez que sempre dava o mesmo diagnóstico, e por no dataset de treste (escolhido de forma aleatória) ter mais dados benignos, tem-se uma ffalsa impressão da confiabilidade nesse caso.

Já no último teste, verificamos um comportamento do SVM com o kernel RBF conforme o esperado, conseguindo classificar de forma satisfatória os diagnósticos malignos e no caso dos benignos de forma que supreendeu.

Diante dos resultados, é notável que para considerar o SVM confiável só será possivél após encontrar os valores ideias de seus parâmetros. Ainda que tenhamos um resultado acima de 60%(como observado no caso do kernel gaussiano sem alteração dos parâmetros), precisamos olhar como ele vem se comportando com a amostra como um todo, fazendo multiplas validações.

REFERÊNCIAS

[1] Bishop, C. Pattern Recognition and Machine Learning. Springer, 2006

[2] Mitchell, T. Machine Learning. McGraw Hill, 1997.

[3] Lichman, M. (2013). UCI Machine Learning Repository. Irvine, CA: the University of California, School of Information and Computer Science.

[4] Bird, S., Klein, E., and Loper, E. (2009). Natural language processing with Python: Analyzing text with the natural language toolkit. Sebastopol, CA: O'Reilly Media.