Discente: Deivison Venicio Souza

Laboratório Aprendizagem de Máquina

Lab4: Avaliação Support Vector Machines

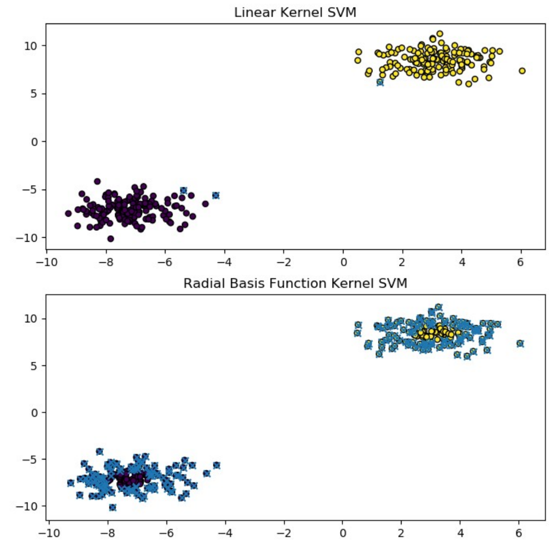
O programa **svmtoy.py** utiliza duas funções diferentes para gerar dados de treinamento. A função **make_blobs** gera dados que, em algumas vezes, são linearmente separáveis. Já a função **make_gaussian_quantiles** gera dados que não são separáveis linearmente.

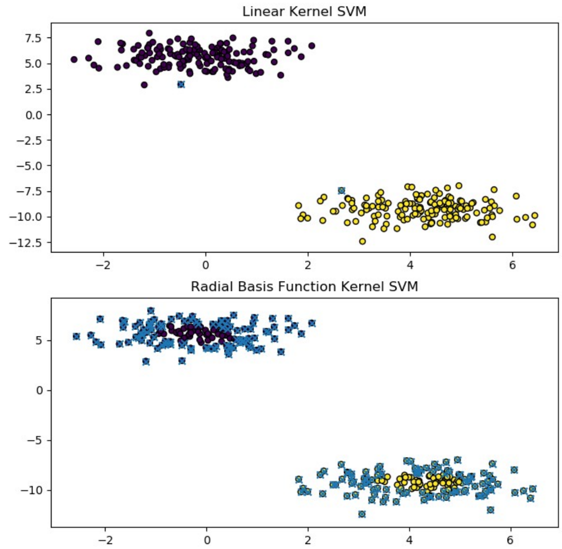
1) Utilizando a função **make_blobs** compare os **SVMs linear** e **RBF**. Qual classificador produz mais vetores de suporte? Em geral qual classificador alcança a melhor acurácia? Qual é a quantidade média de vetores de suporte para em cada caso?
2) Responda as mesmas perguntas utilizando a função make_gaussian_quantiles.

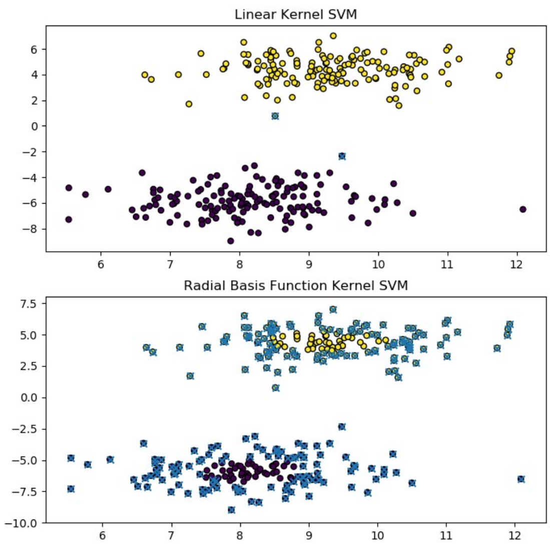
Sample	Função make_blobs					
	Linear Kernel SVM		Radial Basis Function SVM			
	Vetores de Suporte	Acurácia	Vetores de Suporte	Acurácia	C	gamma
Sample 1	3	1,0	203	1,0	0,03125	0,5
Sample 2	2	1,0	198	1,0	0,03125	0,5
Sample 3	2	1,0	210	1,0	0,03125	0,5
Sample 4	4	1,0	207	1,0	0,03125	0,5
Sample 5	5	1,0	246	0,99	0,03125	0,0078125
Sample 6	8	0,990	143	1,0	0,03125	0,125
Média	**4**	**0,9983**	**201**	**0,9983**		
Sample	**Função make_gaussian_quantiles**					
	Linear Kernel SVM		**Radial Basis Function SVM**			
	Vetores de Suporte	**Acurácia**	**Vetores de Suporte**	**Acurácia**	**C**	**gamma**
Sample 1	296	0,633	55	0,983	128	0,03125
Sample 2	275	0,643	26	0,996	32	0,5
Sample 3	292	0,623	270	0,983	0,03125	0,5
Sample 4	277	0,666	37	0,999	8.0	0,5
Sample 5	292	0,600	17	0,993	8192.0	0,03125
Sample 6	295	0,650	34	0,983	8.0	2.0
Média	**288**	**0,6358**	**73**	**0,9895**		

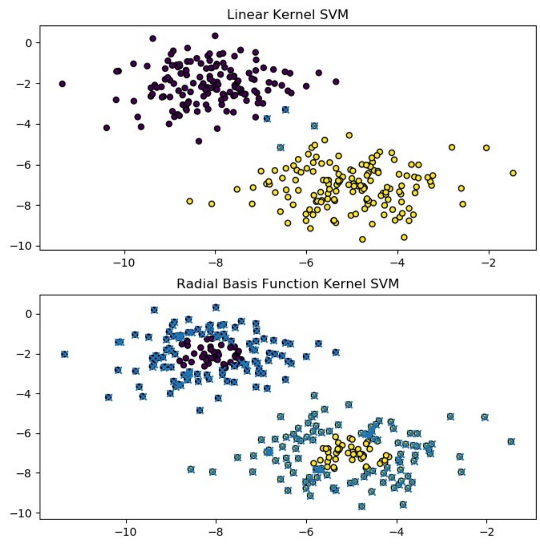
A tabela acima resume a acurácia e o número de vetores de suporte utilizados pelo SVM Linear e SVM Radial Basis Function, variando-se os simuladores de dados make_blobs e make_gaussian_quantiles. Ainda, são apresentados os hiperparâmetros C e gamma do SVM RBF. Foram realizadas 6 simulações para cada função geradora de dados (make_blobs e make_gaussian_quantiles). Quando usada a função **make_blobs** o classificador “Linear Kernel SVM” produziu menor quantidade de vetores de suporte (média = 4), já quando usada a função **make_gaussian_quantiles** o classificador Radial Basis Function SVM foi aquele que se utilizou de menor quantidade de vetores de suporte (média = 73). Em relação à acurácia, quando usada a função **make_blobs** que gera dados linearmente separáveis, em geral, o desempenho médio dos classificadores foi similar. Neste caso, preferência deve ser dada ao uso do SVM Linear pela menor complexidade, e uso de poucos vetores de suporte. Por outro lado, quando usada a função **make_gaussian_quantiles** que gera dados que não são separáveis linearmente, o classificador mostra melhor desempenho na separação de classes, com acurácia média de 98,95%.

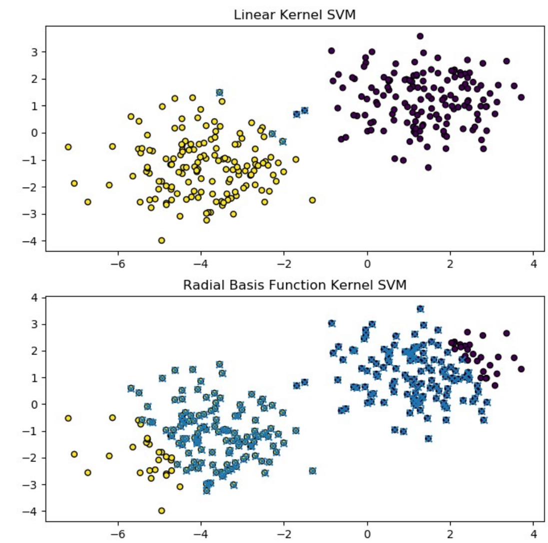
A) Gráficos usando a função make_blobs

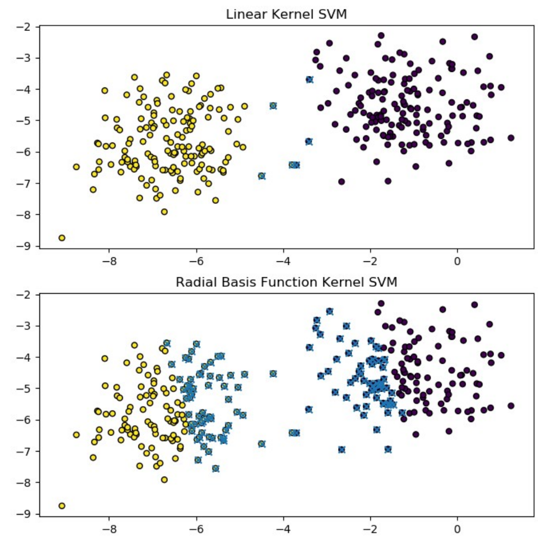




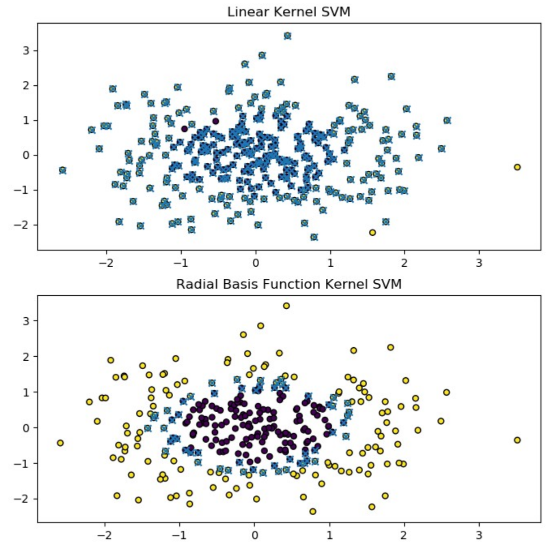


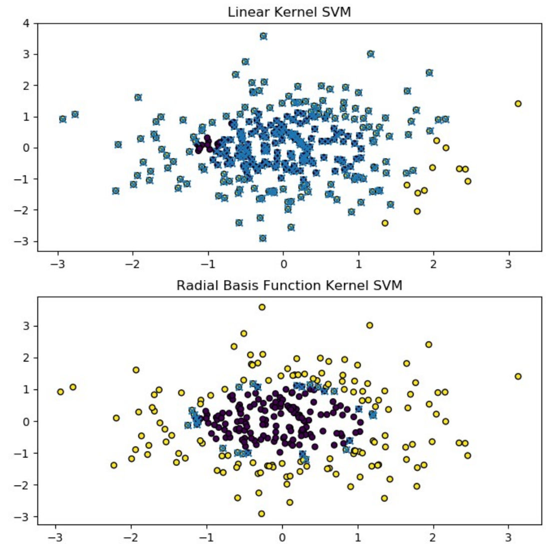


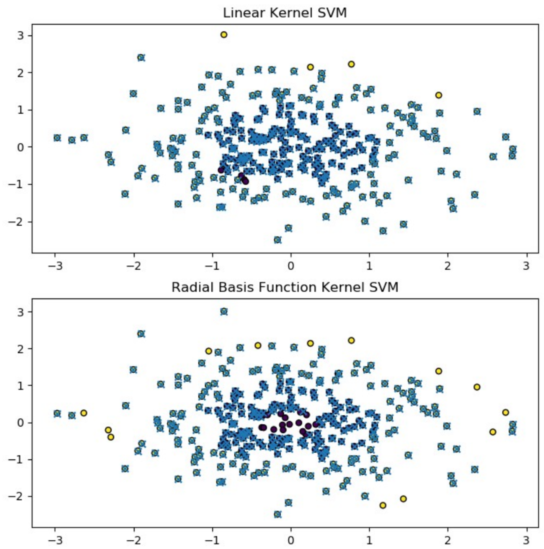


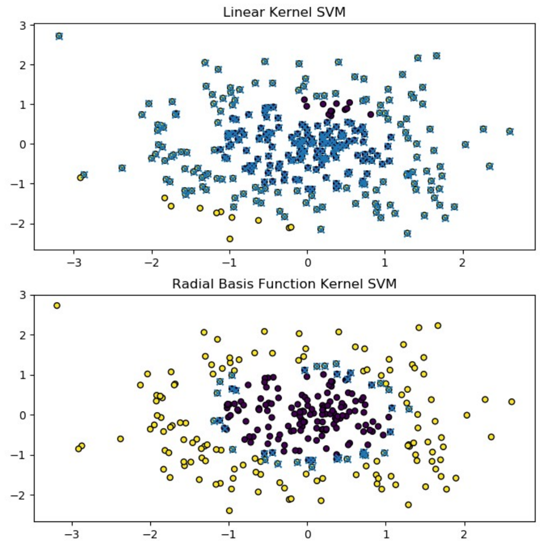


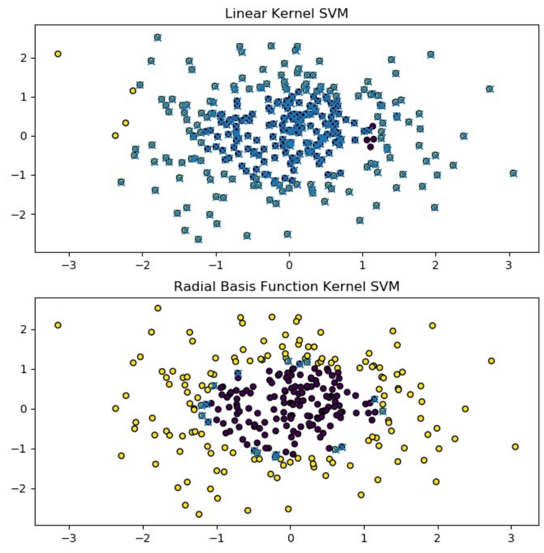
B) Gráficos usando a função make_gaussian_quantiles

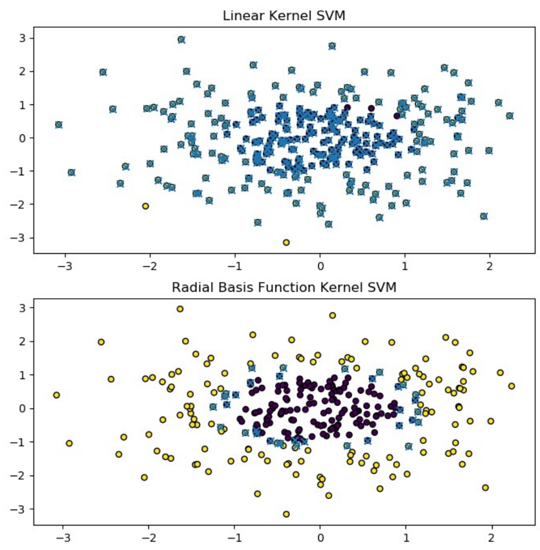












Utilize o programa **svm.py** para classificar a base de digitos utilizada nos exercícios anteriores. Note que esse programa utiliza validação cruzada para buscar os parâmetros **C** e **Gamma** no caso do **kernel RBF**, os quais tem o escopo definido nas variáveis **C_range** e **gamma_range**.

1) Compare o desempenho do SVM linear e RBF com os classificadores utilizados anteriormente. Para esse problemas em particular, qual SVM é melhor em termos de acurácia e tempo de treinamento?

2) Qual foi a quantidade de vetores de suporte para cada SVM?

	Vetores de Suporte	**Acurácia**	**C**	**gamma**	**Real**	**User**	**Sys**
Linear Kernel SVM	935	0,9765	-	-	4m15.564s	4m20.456s	0m0507s
Radial Basis Function SVM	338	0,9644	32.0	8.0	398m13.790s	1617m30.236s	1m9.908s

Tamanho (base treino)	**Acurácia**						
	Logistic Regression	**Gaussian Naïve Bayes**	**Linear Discriminant Analysis**	**KNN**	**Perceptron**	**SVM Linear**	**SVM RBF**
20000	0.9069	0.8891	0.9279	0.9399	0.9475	0.9765	0.9644
	System time						
real	0m31.915s	0m21.761s	0m9.065s	50m39.345s	2m15.372s	4m15.564s	398m13.790s
user	0m41.656s	0m16.080s	0m13.064s	50m24.972s	2m22.608s	4m20.456s	1617m30.236s
sys	0m0.412s	0m3.512s	0m0.440s	0m0.232s	0m0.336s	0m0507s	1m9.908s

O desempenho dos algoritmos **Linear Kernel SVM** e **Radial Basis Function SVM** foi avaliado usando-se a base de treino total (20000 instâncias). No caso do **Radial Basis Function SVM** que possui dois hiperparâmetros (C e gamma) foi necessário “tunar" o algoritmo, isto é, buscar a melhor configuração do modelo que proporcionasse a melhor separação de classes. Para tanto, criou-se uma grade de busca com os hiperparâmetros candidatos a ajuste ótimo. Em seguida, implementou-se o método de validação cruzada para buscar o modelo com hiperparâmetros ótimos. O "sklearn" possui a função "GridSearchCV" que pode ser usada para realizar este procedimento. Por padrão, o "GridSearchCV" usa uma validação cruzada de três dobras. Os modelos SVM tiveram valores de acurácia similares (Linear = 0,9765 e RBF = 0,9644), porém o tempo computacional para treinar o RBF foi demasiadamente grande. Assim, o SVM Linear foi considerado melhor ponderando-se a acurácia e o custo computacional. O SVM Linear usou de 935 vetores de suporte, enquanto que o RBF usou de 338 vetores de suporte. Por fim, comparando-se o desempenho dos SVMs (base de treino = 20000) com os demais algoritmos já treinados, pode-se constatar a superioridade do SVM na separação de classes, haja vista a maior acurácia.