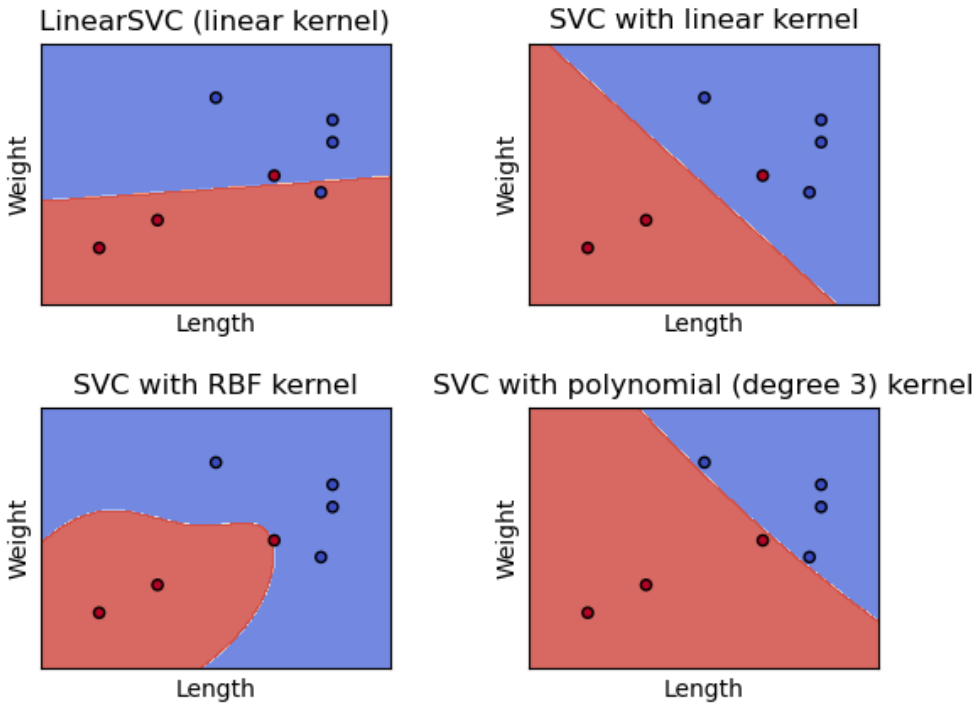
IC –UFPA – Homework sobre SVMs

Diretrizes gerais: *Crie um notebook Jupyter para colocar a resolução e código usado nas questões abaixo no notebook. Depois crie um projeto na sua conta do github.com (abra uma se não tiver ainda) e informe no Google Classroom o URL do projeto público de onde se possa baixar o notebook criado. No próprio notebook, responda às questões abaixo. Descreva em detalhes suficientes os passos de matemática usados para explicar suas respostas. Você pode contar com software para lhe ajudar a calcular valores, mas transponha os resultados para texto e explique os passos importantes. Você pode usar apenas duas casas decimais para simplificar. A menos de indicado contrário, assuma como conjuntos de treino, teste e validação os arquivos dataset_train.txt, dataset_test.txt e dataset_validation.txt que se encontram no nextcloud da disciplina, na pasta: code\ml-ufpa\homeworks\Homework_5_SVMs.*

1) A figura abaixo mostra os exemplos de treino e as regiões de decisão de uma SVM (ou SVC, de "support vector classifier") para um problema binário (duas classes) onde o vetor **x** de "features" de entrada tem dois parâmetros (weight e length). Os pontos em azul e vermelho, representam os exemplos de cada classe. As regiões em azul e vermelho representam as regiões nas quais o classificador irá prever uma classe azul e vermelha, respectivamente. Olhando para as figuras, informe:

a) Quantos erros cada SVM possui no conjunto de treino ?
b) Qual a que lhe parece ser a melhor SVM? Por que?



2) A ***regularização*** é uma importante ferramenta para controlar a "complexidade do modelo" durante a fase de seu projeto. No projeto de uma SVM, o hiperparâmetro **C** é usado para efetuar a regularização do modelo na etapa de seleção do modelo, ou seja, quando os hiperparâmetros são variados e observamos o impacto através do erro de classificação no conjunto de validação. Suponha que você se encontra em uma situação onde está descontente com o atual resultado de sua SVM e observa que o número de vetores de suporte está relativamente pequeno. Você quer potencialmente aumentar o número de vetores de suporte, fazendo com que sua nova SVM seja um modelo mais "complexo". Para isso, você deve aumentar ou diminuir o parâmetro **"C"** da classe SVC do scikit-learn?

3) Um classificador usa vetores de entrada com dimensão **K=5 "features"**. Após treinar uma **SVM** linear, o número de vetores de suporte foi de **450** exemplos. Neste caso, calcular o kernel linear corresponde a um produto interno entre dois vetores de dimensão **K=5**, cada, o que requer K multiplicações e K-1 adições. **Estime o fator $F = C_{original} / C_{perceptron}$ de redução do custo computacional da etapa de teste ao converter esta SVM linear para um perceptron.** Assuma que os custos $C_{original}$ e $C_{perceptron}$ correspondem ao número de multiplicações e adições usando-se, respectivamente, a SVM original com os 450 vetores de suporte e após sua conversão para perceptron.

4) Interpretação do resultado de projeto de SVMs.
O treinamento de uma SVM linear usando a classe SVC do sklearn retornou o resultado:

```
svm.n_support_= [1 2]  # número de vetores de suporte por classe
svm.support_vectors_=      [[ 1. 4.]  # vetores de suporte
                            [-2.  3.]
                            [-2. -5.]]
svm.dual_coef_= [[-0.5 -0.3 0.8]] # valores de \lambda
svc.intercept_= [-2] # “bias”
```

Observe que esta questão não usa o conjunto de treino indicado, mas outro. Pede-se:

a) A função de decisão para essa SVM de acordo com a fórmula geral:

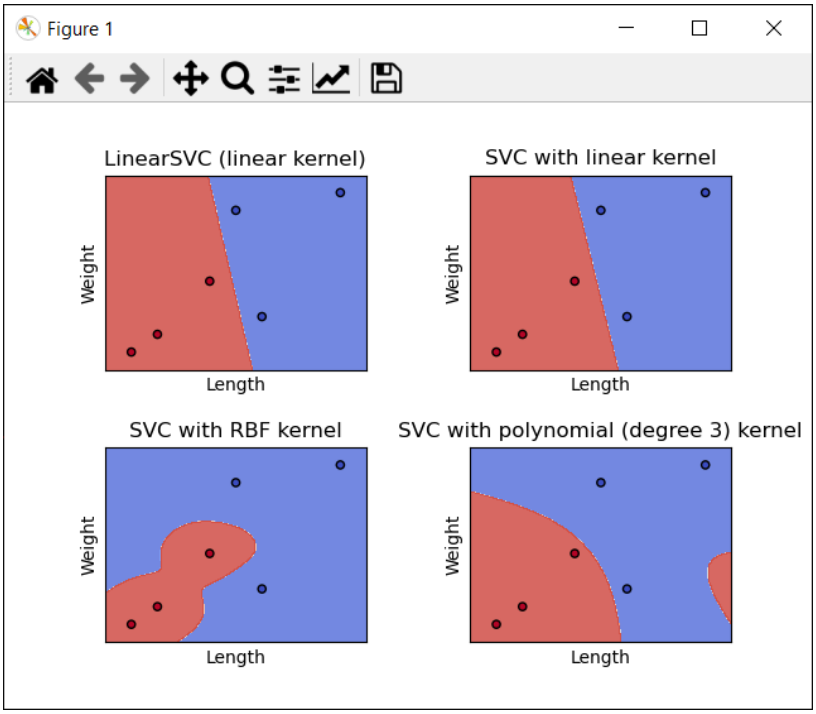
$$f(\mathbf{z}) = \left(\sum_{n=0}^{N-1} \lambda_n K(\mathbf{z}, \mathbf{x}_n) \right) + b$$

b) A função de decisão desta SVM quando escrita como um perceptron $f(\mathbf{z})=<\mathbf{z},\mathbf{w}>+b$.

c) A saída da função de decisão quando o vetor de entrada é $\mathbf{z}=[0, 0]$ e o respectivo rótulo predito y assumindo-se que $y = I(f(\mathbf{z}) > 0)$, onde I(.) é a função “indicador”, que é 1 se o argumento é verdadeiro, ou 0 em caso contrário.

5) Interpretação do resultado de projeto de SVMs.

O código ak_svm_prova1.py e o conjunto de treino em dataset_train.txt foram usados para gerar a figura e o log (saída de texto) abaixo. Em suma, foram treinadas 4 SVMs, duas com kernel linear (SVMs 1 e 2) e outras duas com kernel não-linear (SVMs 3 e 4).



..*

optimization finished, #iter = 16
obj = -0.740000, rho = -1.799545
nSV = 3, nBSV = 0
Total nSV = 3

Adapted from https://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html#sphx-glr-auto-examples-svm-plot-iris-svc-py

Normalized training set:
[[0. -4.]
 [-1. 2.]
 [3. 3.]
 [-5. -6.]
 [-4. -5.]
 [-2. -2.]]
1) Linear SVM with LinearSVC
linear_svc.coef_= [[-0.68611287 -0.10653139]]
linear_svc.intercept_= [-0.99556119]
[LibSVM]

Generic SVMs with SVC
2) SVM with SVC
{'C': 1, 'break_ties': False, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 3, 'gamma': 'scale', 'kernel': 'linear', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': False, 'tol': 0.001, 'verbose': 1}
svm.n_support_= [2 1]
svm.support_= [0 1 5]
svm.support_vectors_= [[0. -4.]
 [-1. 2.]
 [-2. -2.]]
svm.dual_coef_= [[-0.45994152 -0.27992202 0.73986354]]
svc.intercept_= [-1.79954513]
At most 10 decisions: svm.decision_function(X)= [-1.00032491 -0.99935019 -5.99837547 5.39831049 3.99870038 0.99967509]
Accuracy via svm.score(X,y)= 1.0
svc_with_linear_kernel.coef_= [[-1.19980506 -0.19980506]]
Estimated perceptron_weights= [-1.19980506 -0.19980506]
Estimated bias= -1.7997075844389612
3) SVM with SVC
{'C': 1, 'break_ties': False, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 3, 'gamma': 0.7, 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001, 'verbose': False}
svm.n_support_= [3 3]
svm.support_= [0 1 2 3 4 5]
svm.support_vectors_= [[0. -4.]
 [-1. 2.]
 [3. 3.]
 [-5. -6.]
 [-4. -5.]

[-2. -2.]]
svm.dual_coef_= [[-0.91722233 -0.91351914 -0.91300432 0.87185969 0.8718861 1.]]
svc.intercept_= [-0.08676121]
At most 10 decisions: svm.decision_function(X)= [-1.00027976 -1.00027976 -0.99977173 1.00010297 1.00022828 0.90993821]
Accuracy via svm.score(X,y)= 1.0
4) SVM with SVC
{'C': 1, 'break_ties': False, 'cache_size': 200, 'class_weight': None, 'coef0': 0, 'decision_function_shape': 'ovr', 'degree': 3, 'gamma': 'auto', 'kernel': 'poly', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001, 'verbose': False}
svm.n_support_= [2 1]
svm.support_= [0 1 5]
svm.support_vectors_= [[0. -4.]
[-1. 2.]
[-2. -2.]]
svm.dual_coef_= [[-0.00887134 -0.03133903 0.04021037]]
svc.intercept_= [-1.03731897]
At most 10 decisions: svm.decision_function(X)= [-1.00028193 -0.99943614 -7.91231897 38.49667194 20.25085641 0.99971807]
Accuracy via svm.score(X,y)= 1.0
('LinearSVC (linear kernel)', 'SVC with linear kernel', 'SVC with RBF kernel', 'SVC with polynomial (degree 3) kernel') have accuracies:
svm_scores= [1. 1. 1. 1.]

Pergunta-se:

a) Quais as expressões para as SVMs 3 e 4 (não-lineares) e para a SVM 2 (linear) de acordo com a nomenclatura abaixo (escreva a "fórmula" da função de decisão de cada SVM indicando o valor do bias "b", etc., considerando que $\mathbf{z}$ e $\mathbf{x}_n$ são vetores já normalizados (com o primeiro elemento dividido por 1000).

$$f(\mathbf{z}) = \left(\sum_{n=0}^{N-1} \lambda_n K(\mathbf{z}, \mathbf{x}_n) \right) + b$$

b) Quais as expressões para as SVMs 1 e 2 (lineares) quando escritas como perceptrons?

c) Suponha que para a SVM 2 (linear), apenas as informações abaixo fossem fornecidas. Com elas, você já pôde escrever essa SVM no formato geral de uma SVM, como no item a). Explique agora com clareza quais os passos que você adotaria para converter essa SVM linear em um perceptron como descrito no item b) e indique em termos do número de multiplicações e adições, qual economia no custo computacional que a implementação como perceptron alcança.

svm.n_support_= [2 1]
svm.support_= [0 1 5]
svm.support_vectors_= [[0. -4.]
[-1. 2.]

[-2. -2.]]
svm.dual_coef_= [[-0.45994152 -0.27992202 0.73986354]]
svc.intercept_= [-1.79954513]

d) Para as SVMs 3 e 4 (não-lineares), indique:
d.1) qual o número total de vetores de suporte (SVs), considerando todas as classes.
d.2) Quais os índices desses SVs no conjunto de treino e os respectivos valores dos "lambdas" (coeficientes duais)
d.3) Qual o valor do termo independente b chamado de "bias" ou "intercept_".

e) Considerando as saídas da SVM abaixo para o conjunto de treino, em qual dos exemplos de treino esta SVM está menos "confiante" (assumindo que esses números são "confidence scores") em sua decisão e qual é a classe que esta SVM prediz para este exemplo?
svm.decision_function(X)= [-1.00027976 -1.00027976 -0.99977173 1.00010297 1.00022828 0.90993821]

6) Use o scikit-learn para treinar uma **SVM com kernel linear**, usando hiperparâmetro: **C=1**.

Neste caso use a versão normalizada dos dados, com os fatores de normalização projetados com base no conjunto de treino, para que este conjunto tenha média 0 e variância 1.

1) Treine a SVM com o conjunto de treino indicado no arquivo dataset_train.txt.
2) Indique o desempenho do modelo SVM no conjunto de teste dataset_test.txt.
3) Converta esta SVM para um perceptron, estime o custo computacional durante a fase de teste da SVM na forma original com sua versão implementada como perceptron. Indique se há algum ganho computacional em termos de memória e número de operações ao se converter essa SVM para um perceptron.

7) Sobre o projeto de SVMs:
a) Use o scikit-learn (exemplificado no código ak_svm.py) para treinar uma **SVM** com kernel Gaussiano (também chamado de **"RBF"**), usando o **conjunto de validação** para fazer o "tuning" (otimização) de dois hiperparâmetros: **"C"** e **"gamma",** avaliando os valores:

C = 0.01
C = 1
C = 100

gamma = 0.5
gamma = 1

Em suma:
1) Treine as SVMs com o conjunto de treino variando C e gamma;
2) Observe o resultado usando o conjunto de validação;
3) Indique qual SVM apresenta melhor performance no conjunto de validação.

Obs: O produto Cartesiano das opções de hiperparâmetros irá exigir 3 x 2 = 6 treinos de SVMs.

b) Descreva totalmente (por extenso) os parâmetros que compõem a fórmula final do classificador SVM escolhido, indicando:

b1) Os parâmetros escolhidos do kernel (gamma neste caso, visto que C é apenas um hiperparâmetro, usado no treino mas não no teste)

b2) O número de vetores de suporte (SVs)

b3) Seus índices no conjunto de treino

b4) Os respectivos valores dos "lambdas" (ou "dual_coef_" de coeficientes duais) e

b5) O termo independente b chamado de "bias" ou "intercept_". Em outras palavras, indique os números que serão usados na função de decisão *f*, que pode ser escrita abaixo, onde n é o índice do SV no conjunto de treino:

$$f(\mathbf{z}) = \left(\sum_{n=0}^{N-1} \lambda_n K(\mathbf{z}, \mathbf{x}_n) \right) + b$$

ou como na documentação do scikit-learn em https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation, onde $l_i = y_i a_i$, onde y_i é o "label" do SV no conjunto de treino. Diferente do n no somatório acima, o índice i no somatório abaixo já seleciona apenas os vetores de suporte. O somatório acima, percorre todo o conjunto de treino, e os exemplos que não são vetores de suporte possuem $\lambda_n = 0$ e daí não são contabilizados.

Once the optimization problem is solved, the output of decision_function for a given sample x becomes:

$$\sum_{i \in SV} y_i \alpha_i K(x_i, x) + b,$$

and the predicted class correspond to its sign. We only need to sum over the support vectors (i.e. the samples that lie within the margin) because the dual coefficients α_i are zero for the other samples.

These parameters can be accessed through the attributes `dual_coef_` which holds the product $y_i \alpha_i$, `support_vectors_` which holds the support vectors, and `intercept_` which holds the independent term b

Note que a documentação é confusa, pois chama de “dual coefficients” para a_i e depois indica que dual_coef_ armazena o produto $l_i = y_i\, a_i$.

c) Usando a SVM escolhida e os dois primeiros exemplos $\mathbf{z}_1$ e $\mathbf{z}_2$ do conjunto de teste descrito em dataset_test.txt, **compare** o resultado numérico do “score” que você obtém para a função de decisão $f(\mathbf{z})$ calculada pelos parâmetros do item **b)** anterior com os gerados pela função ***decision_function()*** da classe SVM do scikit-learn. Busque explicar eventuais discrepâncias.