INTELIGÊNCIA COMPUTACIONAL II
Prof. Carlos Eduardo Pedreira
PESC/COPPE/UFRJ

Trabalho Prático III

Data de Entrega: 31 de Agosto 2018

O Trabalho Prático III visa implementar duas tarefas e gerar dois textos científicos distintos:
1) Parte dos exercícios computacionais descritos no **Homework #5**
Disponível em: https://work.caltech.edu/homework/hw5.pdf

2) Implementar um experimento computacional com uso do algoritmo *Support Vector Machine* (SVM) em uma conhecida base de testes disponibilizada pela *Knowledge Extraction Evolutionary Learning* (KEEL) e de um segundo modelo de classificação a sua escolha. A implementação de ambas as tarefas deverá ser com a linguagem de sua maior familiaridade.

Para Tarefa 1 um relatório técnico sucinto deve ser escrito e entregue com o seguinte conteúdo:

- Você deve informar o seu nome no topo do relatório (primeira informação);
- Uma breve introdução sobre o assunto de cada tarefa, separando o relatório técnico por seções 1 e 2;
- O código fonte da implementação devidamente comentado como anexo;
- Resultados alcançados e sua interpretação dos mesmos.

Para Tarefa 2 escreva um artigo científico (coluna dupla padrão *IEEE Conference - 6 pages*). Incluindo seu código como anexo ao artigo.

* O código deverá ser apresentado e executado em tempo oportuno, para fins de avaliação, com possível arguição do mesmo. A ser agendado.

Descrição das tarefas

Tarefa 01 - Implementação exercícios computacionais Homework #5
*Uma tradução livre do enunciado presente no **Homework #5** foi gerada. Caso ocorram dúvidas na interpretação recorram ao texto original em inglês.

Gradiente Descendente
Considere o erro não linear de superfície $E(u, v) = (ue^v - 2ve^{-u})^2$. Vamos iniciar o ponto $(u, v) = (1, 1)$ e minimizar este erro usando gradiente descendente no espaço (uv). Use $\eta = 0.1$ (taxa de aprendizado, não tamanho de parada).
1) Quantas iterações (das opções dadas) que encontra o erro $E(u, v)$ abaixo de 10^{-14} pela primeira vez? Na sua implementação, use precisão dupla para obter a exatidão necessária.
a) 1 b) 3 c) 5 d) 10 e) 17

2) Após executar iterações suficientes para que o erro seja menor que 10^{-14}, quais são os valores próximos (na distância euclidiana) entre as seguintes opções de valores (u, v) que você obteve no Problema 1?

a) (1.000, 1.000) b) (0.713, 0.045) c) (0.016, 0.112)
d) (-0.083, 0.029) e) (0.045, 0.024)

3) Agora, nós vamos comparar a performance de "coordenada descendente". Em cada iteração, nós vamos ter dois passos ao longo de duas coordenadas. O passo 1 é mover apenas a coordenada u para reduzir o erro (assuma a aproximação de primeira ordem assim como no gradiente descendente), e o passo 2 é reavaliar e mover apenas a coordenada v para reduzir o erro (novamente assua a aproximação de primeira ordem). Continue usando a taxa de aprendizagem $\eta = 0.1$ como feito no gradiente descendente. Qual vai ser o valor de erro $E(u, v)$ mais próximo após 15 iterações completas (30 passos)?

a) 10^{-1} b) 10^{-7} c) 10^{-14} d) 10^{-17} e) 10^{-20}

Regressão Logística

Neste problema nós vamos criar uma função *target* própria, f (probabilidade neste caso) e o conjunto de dados D para verificar como a Regressão Logística funciona. Para simplicidade, nós vamos usar f para ser uma probabilidade 0/1, então apenas y é uma função determinística de $\mathbf{x}$. Use $d = 2$ apenas para você visualizar o problema, e seja $\mathcal{X} = [-1, 1] \times [-1, 1]$ com probabilidade uniforme de seleção para cada x pertencente a X. Escolha uma reta no plano com limite entre $f(x) = 1$ (onde y deve ser +1) e $f(x) = 0$ (onde y deve ser -1), a partir de dois pontos aleatórios de $\mathcal{X}$ e tomando a reta passando por eles como limite entre $y = \pm 1$. Use N = 100 pontos aleatórios para treinamento a partir de $\mathcal{X}$, e avalie as saídas y_n para cada um desses pontos x_n.

Execute a Regressão Logística com o Gradiente descendente para encontrar g, e estimar E_{out} (o erro de entropia cruzada) gerando um conjunto de pontos suficientemente grande e separado para avaliar o erro. Repita o experimento para 100 execuções com diferentes alvos e use a média. Inicialize o vetor de pesos para a Regressão Logística zerado (todos os valores iguais a zero) em cada execução. Pare o algoritmo quando $\left\|\mathbf{w}^{(t-1)} - \mathbf{w}^{(t)}\right\| < 0.01$, no qual $\mathbf{w}^{(t)}$ representa o vetor de pesos final da época t. Uma época é uma passagem completa pelos N pontos de dados (use uma permutação aleatória de $1, 2, ..., N$ para representar os pontos de dados do algoritmo dentro de cada época, e uso diferentes permutações para diferentes épocas). Use uma taxa de aprendizagem de 0.01.

4) Qual das seguintes opções é a mais próxima de E_{out} para N = 100?
a) 0.025 b) 0.050 c) 0.075 d) 0.100 e) 0.125

5) Quantas épocas, em média, são necessárias para a Regressão Logística convergir para N = 100 usando as regras de inicialização e finalização especificadas na taxa de aprendizado? Escolha o valor que mais se aproxima dos seus resultados.
a) 350 b) 550 c) 750 d) 950 e) 1750

Tarefa 02 - Experimento computacional: *Support Vector Machine(SVM)*

Este experimento computacional tem o objetivo de averiguar a capacidade do SVM na classificação de uma base de dados sintética da KEEL. Conhecida como Base Banana, em que as instâncias pertencem a vários clusters que se apresentam com formato de banana, tal base apresenta duas classes (1) e (-1) para um conjunto de 5300 observações contendo 2 atributos. A base de dados está disponível em:

http://sci2s.ugr.es/keel/dataset.php?cod=182#sub2

A Figura 1 apresenta o diagrama de dispersão de dados simulados da base de dados Banana.

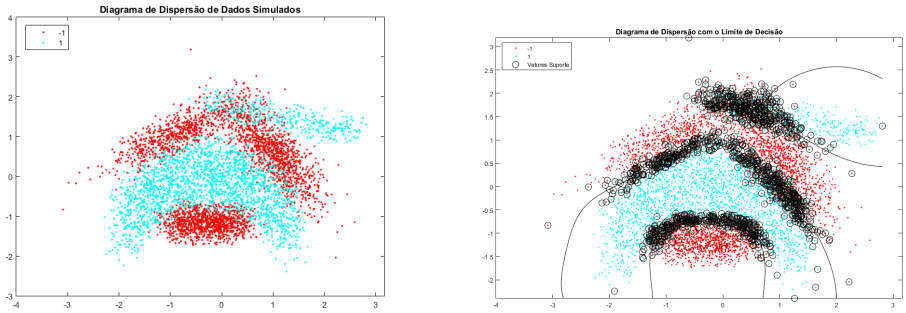

Figura 1: (1) Gráfico característico da base de dados banana (2) Gráfico de dispersão com o limite de decisão - SVM.

Abordagem

1) Gere o gráfico de dispersão da base de dados Banana e discuta;

2) Você deverá testar o SVM com os diferentes *kernels*: sigmoide (usando γ: 1, 0.5 e 0.01), linear, polinomial e RBF. Para cada um dos *kernels* execute *k-fold cross-validations* com 3 valores distintos de k: 2, 5 e 10. Monte uma tabela contendo o valor do erro fora da amostra E_{out} e, por conseguinte, a taxa de acerto do SVM a cada caso.

3) Após sua análise pessoal e discussão sobre os resultados obtidos, apresente gráficos que indiquem os vetores de suporte para o modelo *kernel/k-fold* de melhor e pior desempenho (se baseie no exemplo apresentado pela Figura 1(2) - Você pode gerar um gráfico mais elaborado inclusive). Discuta sobre os resultados gráficos obtidos.

4) Escolha um outro modelo de classificação justificando sua escolha e o apresente brevemente. Faça uma implementação computacional deste modelo e compare o resultado com a melhor solução obtida dentre os modelos SVM da questão 2. Discuta os resultados obtidos.

5) Apresente seu código fonte como anexo.

Neste experimento você deve fazer uso de bibliotecas prontas do Matlab, Python ou de outras linguagens caso existam. Exclusivamente para este experimento a ideia é não reinventar a roda.

Para seu auxílio veja: http://scikit-learn.org || https://www.mathworks.com