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

Lista de Exercícios computacionais

Data de Entrega: 16 de Agosto 2019

Esta lista de exercícios computacionais visa implementar os modelos propostos pelos homeworks do *Livro Learning from data*, disponíveis em:
https://work.caltech.edu/homework/hw1.pdf
https://work.caltech.edu/homework/hw2.pdf
https://work.caltech.edu/homework/hw5.pdf

Você deverá implementar os exercícios na linguagem de sua escolha.

ATENÇÃO: o intuito é implementar os exercícios computacionais "NA MÃO E SEM O USO" de bibliotecas prontas.

Um relatório prático 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;
- O código fonte da implementação devidamente comentado (anexo);
- Resultados alcançados e sua interpretação dos mesmos.

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

Descrição dos exercícios

*Uma tradução livre dos enunciados presentes nos homeworks foi gerada. Caso ocorram dúvidas na interpretação recorram ao texto original em inglês.

O Algoritmo de Aprendizagem Perceptron

Neste problema, você vai criar a sua função *target* (alvo) f e o conjunto de dados D para verificar como o Algoritmo de Aprendizagem Perceptron trabalha. Utilize $d = 2$ para você visualizar o problema, e assuma $\mathcal{X} =$ [-1,1] $\times$ [-1,1] com probabilidade uniforme de seleção $\mathbf{x} \in \mathcal{X}$.
Em cada execução, escolha uma reta aleatória no plano como sua função *target* f (faça isso utilizando dois valores aleatórios, pontos uniformemente distribuídos em [-1,1] $\times$ [-1,1] gerando a reta que passa entre eles) na qual de um lado a reta mapeia +1 e do outro -1. Escolha as entradas $\mathbf{x_n}$ de uma base de dados de pontos aleatórios (uniformemente em $\mathcal{X}$), e avalie a função *target* em cada $\mathbf{x_n}$ e obtenha a correspondente saída y_n.
Para cada execução use o Algoritmo de Aprendizagem Perceptron (PLA) para encontrar g. Inicie o PLA como o vetor de pesos $\mathbf{w}$ zerado (todos os pesos iguais

a zero) e em cada iteração o algoritmo deverá escolher um ponto aleatório a partir do conjunto de pontos classificados incorretamente. Nós estamos interessados em dois valores: número de iterações que o PLA precisa para convergir a g, e a divergência entre f e g na qual $\mathbb{P}[f(\mathbf{x}) \neq g(\mathbf{x})]$ (a probabilidade que f e g vão divergir na classificação de um ponto aleatório). Você pode calcular exatamente esta probabilidade ou você pode gerar uma aproximação gerando uma grande quantidade de conjuntos separados de pontos para estimá-la.
A fim de obter uma estimativa confiável para estes dois valores, você deve repetir o experimento por 1000 execuções (cada uma como foi especificado acima) e tomar a média sobre estas execuções.

1) Para N = 10. Em média quantas iterações são necessárias para que o PLA convirja para N = 10 pontos treinados? Apresente o valor aproximado de seu resultado (resultado próximo a media | sua resposta - opção | é próxima de 0).
a) 1 b) 15 c) 300 d) 5000 e) 10000

2) Qual a opção mais se aproxima de $\mathbb{P}[f(\mathbf{x}) \neq g(\mathbf{x})]$ para N = 10;
a) 0.001 b) 0.01 c) 0.1 d) 0.5 e)0.8

3) Agora, teste N= 100. Em média quantas iterações são necessárias para que o PLA convirja para N = 100 pontos de treinamento? Informe o valor mais próximo ao seu resultado.
a) 50 b) 100 c) 500 d) 1000 e)5000

4) Qual a opção mais se aproxima de $\mathbb{P}[f(\mathbf{x}) \neq g(\mathbf{x})]$ para N = 100;
a) 0.001 b) 0.01 c) 0.1 d) 0.5 e)0.8

Regressão Linear
Nestes problemas nós vamos explorar como a Regressão Linear para classificação trabalha. Da mesma maneira com uso do Algoritmo de Aprendizagem Perceptron no **Homework #1**, você vai criar a sua própria função *target* (alvo) f e o conjunto de dados D. Utilize $d = 2$ para que você possa visualizar o problema, e assuma $\mathcal{X}$ = [-1,1] × [-1,1] com probabilidade uniforme selecionando cada $\mathbf{x} \in \mathcal{X}$. Em cada execução, escolha uma reta aleatória no plano como sua função *target* f (faça isso selecionando dois pontos aleatórios, uniformemente distribuídos em [-1,1] × [-1,1] e use a reta que passa entre eles), de forma que a reta mapeie +1 por um lado e -1 pelo outro. Escolha as entradas $\mathbf{x_n}$ do conjunto de dados de pontos aleatórios (uniformemente em $\mathcal{X}$), e avalie a função *target* em cada $\mathbf{x_n}$ para encontrar a saída correspondente y_n.

5) Utilize N = 100. Use Regressão Linear para encontrar g e avaliar E_{in}, a fração de pontos dentro da amostra que foram classificados incorretamente. Repita o experimento 1000 vezes e use o valor médio (guarde as g's que serão usadas novamente no Problema 2). Qual é o valor médio aproximado de E_{in}? (aproximado é a opção que faz a expressão | sua resposta - dada opção| próxima a 0. Use esta definição aqui e sempre).

a) 0 b) 0.001 c) 0.01 d) 0.1 e) 0.5

6) Agora, gere 1000 novos pontos e os use para estimar o erro fora da amostra E_{out} de g que você fez no Problema 1 (número de pontos classificados incorretamente/ número total de pontos fora da amostra). Novamente, execute o experimento 1000 vezes e guarde a média. Qual é o valor médio aproximado de E_{out}?

a) 0 b) 0.001 c) 0.01 d) 0.1 e) 0.5

7) Agora, utilize N = 10. Posteriormente, procurando os pesos usando Regressão Linear, os use como um vetor de pesos inicial para o Algoritmo de Aprendizagem Perceptron. Execute PLA até que convirja para o vetor final de pesos que separe completamente todos os pontos dentro da amostra. Entre as opções abaixo, qual é o valor mais próximo do número médio de iterações (mais de 1000 execuções) que o PLA leva para convergir? (Quando estiver implementando o PLA, escolha um ponto aleatório para o conjunto classificado incorretamente para cada iteração).

a) 1 b) 15 c) 300 d) 5000 e) 10000

Regressão Não-Linear
Nestes problemas, nós vamos novamente aplicar Regressão Linear para classificação. Considere a função *target*:

$f(x_1, x_2) = sign(x_1^2 + x_2^2 - 0.6)$.

Gere um conjunto de treinamento de N = 1000 pontos em $\mathcal{X}$ = [-1,1]×[-1,1] com probabilidade uniforme escolhendo cada $\mathbf{x} \in \mathcal{X}$. Gere um ruído simulado lançando o sinal de saída aleatoriamente selecionando 10% do subconjunto de treinamento gerado.

8) Execute a Regressão Linear sem transformação usando o vetor de atributos:

$(1, x_1, x_2)$,

para encontrar o peso $\mathbf{w}$. Qual é o valor aproximado de classificação do erro E_{in} dentro da amostra? (Execute o experimento 100 vezes e use o valor médio de E_{in} para reduzir a variação nos seus resultados.)

a) 0 b) 0.1 c) 0.3 d) 0.5 e) 0.8

9) Agora, transforme os N = 1000 dados de treinamento seguindo o vetor de atributos não-linear:

$(1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$.

Encontre o vetor $\widetilde{\mathbf{w}}$ que corresponde a solução da regressão linear. Quais das hipóteses a seguir é a mais próxima que você encontrou? Neste caso, próximo

significa o valor que mais entra em acordo com sua hipótese (existe uma alta probabilidade de estar acordando com um ponto aleatoriamente selecionado). Em média algumas execuções serão necessárias para assegurar uma resposta estável.

a) $g(x_1, x_2) = sign(-1 - 0.05x_1 + 0.08x_2 + 0.13x_1x_2 + 1.5x_1^2 + 1.5x_2^2)$
b) $g(x_1, x_2) = sign(-1 - 0.05x_1 + 0.08x_2 + 0.13x_1x_2 + 1.5x_1^2 + 15x_2^2)$
c) $g(x_1, x_2) = sign(-1 - 0.05x_1 + 0.08x_2 + 0.13x_1x_2 + 15x_1^2 + 1.5x_2^2)$
d) $g(x_1, x_2) = sign(-1 - 1.5x_1 + 0.08x_2 + 0.13x_1x_2 + 0.05x_1^2 + 0.05x_2^2)$
e) $g(x_1, x_2) = sign(-1 - 0.05x_1 + 0.08x_2 + 1.5x_1x_2 + 0.15x_1^2 + 0.15x_2^2)$

10) Qual o valor mais próximo do erro de classificação fora da amostra E_{out} de sua hipótese no Problema 5? (Estime isso gerando um novo conjunto de 1000 pontos e adicione ruído, como antes. Em média 1000 execuções reduzem a variação em seus resultados).
a) 0 b) 0.1 c) 0.3 d) 0.5 e) 0.8

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 critério de parada).

11) 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

12) 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 11?

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)

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 x. Use $d = 2$ apenas para você visualizar o problema, e seja $\mathcal{X}$ = [-1,1] × [-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 use diferentes permutações para diferentes épocas). Use uma taxa de aprendizagem de 0.01.

13) 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

14) 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