Inteligência Computacional I - Lista I

Luan Vieira

Questão 1

What types of Machine Learning, if any, best describe the following three scenarios:

(i) A coin classification system is created for a vending machine. The developers obtain exact coin specifications from the U.S. Mint and derive a statistical model of the size, weight, and denomination, which the vending machine then uses to classify coins.

A classificação é dada por uma fórmula já conhecida através do modelo estatístico, não há aprendizado pelos dados. Classificação: not learning.

(ii) Instead of calling the U.S. Mint to obtain coin information, an algorithm is presented with a large set of labeled coins. The algorithm uses this data to infer decision boundaries which the vending machine then uses to classify its coins.

Utiliza dados rotulados para o aprendizado, trata-se, portanto, de supervised learning. Classificação: supervised learning.

(iii) A computer develops a strategy for playing Tic-Tac-Toe by playing repeatedly and adjusting its strategy by penalizing moves that eventually lead to losing.

O aprendizado é baseado em um sistema de recompensa/ penalidade, e portanto é um exemplo de reinforcement learning. Ao penalizar movimentos que levem a derrota o algoritmo aprende com as jogadas anteriores e busca não cometer os mesmos erros a fim de maximizar a recompensa. Classificação: reinforcement learning.

Resposta: letra d

Which of the following problems are best suited for Machine Learning?

(i) Classifying numbers into primes and non-primes.

Classificação de números em primos e não primos é um problema determinístico com solução conhecida. Não é um caso de utilizar aprendizado de máquina.

(ii) Detecting potential fraud in credit card charges.

Um conjunto de dados pode ser analisado de forma a . É um problema no qual a aplicação de aprendizado de máquina pode trazer bons resultados.

(iii) Determining the time it would take a falling object to hit the ground.

Existem equações determinísticas oriundas do estudo da física para modelar este problema, não é um evento estocástico. Não é um caso de utilizar aprendizado de máquina.

(iv) Determining the optimal cycle for traffic lights in a busy intersection.

Pode ser auxiliado por aprendizado de máquina, possivelmente em uma simulação com aprendizado por reforço que pune eventos indesejados como trânsito bloqueado e congestionamento e recompensa a diminuição no tempo médio levado pelos carros para saírem da interseção. É um problema no qual a aplicação de aprendizado de máquina pode trazer bons resultados.

Resposta: letra a

Questão 3

We have 2 opaque bags, each containing 2 balls. One bag has 2 black balls and the other has a black ball and a white ball. You pick a bag at random and then pick one of the balls in that bag at random. When you look at the ball, it is black. You now pick the second ball from that same bag. What is the probability that this ball is also black?

Sejam $B_1, B_2 \in \{w, b\}$ as variáveis aleatórias representando as bolas 1 e 2. A probabilidade desejada é de que a bola 2 seja preta dado que a bola 1 é preta, i.e., $P(B_2 = preta \mid B_1 = preta)$:

$$P(B_2 = preta \mid B_1 = preta) = \frac{P(B_2 = preta \cap B_1 = preta)}{P(B_1 = preta)}$$

Sejam Bag_1 e Bag_2 os recipientes contendo as bolas. Sem perda de generalidade, suponhamos que Bag_1 tenha duas bolas pretas e Bag_2 tenha uma preta e uma branca.

A probabilidade do numerador da equação acima representa a probabilidade conjunta das bolas 1 e 2 serem pretas.

Isso equivale a

$$\begin{aligned}
&P(Bag = Bag_1 \cap B_1 = preta \cap B_2 = preta) + P(Bag = Bag_2 \cap B_1 = preta \cap B_2 = preta) \\
&= \frac{1}{2} \cdot 1 \cdot 1 + \frac{1}{2} \cdot \frac{1}{2} \cdot 0 \\
&= \frac{1}{2}
\end{aligned}$$

Quanto ao denominador temos:

$$P(B_1 = preta) = P(Bag = Bag_1 \cap B_1 = preta) + P(Bag = Bag_2 \cap B_1 = preta)$$
$$= \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$$

Então,

$$P(B_2 = preta \mid B_1 = preta) = \frac{\frac{1}{2}}{\frac{3}{4}} = \frac{4}{6} = \frac{2}{3}$$

Resposta: letra d

Consider a sample of 10 marbles drawn from a bin containing red and green marbles. The probability that any marble we draw is red is μ = 0.55 (independently, with replacement). We address the probability of getting no red marbles ($\nu = 0$) in the following cases:

Questão 4)

We draw only one such sample. Compute the probability that $\nu = 0$. The closest answer is ('closest answer' means: |your answer-given option| is closest to 0):

Seja $p = 0,55$ a probabilidade de uma bola retirada ser vermelha. Então $q = 1 - 0,55 = 0,45$ é a probabilidade de não ser vermelha. Como há reposição e cada evento de sorteio de bola de gude é independente, a probabilidade de não haver bola vermelha em 10 retiradas é dada por $q^{10} = 0,45^{10}$

```
p_amostra = 0.45**10
print(p_amostra)
```

```
## [1] 0.0003405063
```

Resposta: letra b

Questão 5)

We draw 1,000 independent samples. Compute the probability that (at least) one of the samples has $\nu = 0$. The closest answer is:

A probabilidade de pelo menos uma das 1000 amostras ter $\nu = 0$ equivale a $1 - P$(todas as amostras terem $\nu \geq 1$).

A probabilidade de uma amostra ter $\nu \geq 1$ é dada por 1− p_amostra, ou seja, $1 - P$(amostra ter 0 bolas vermelhas), calculada anteriormente. Portanto, a probabilidade de 1000 amostras independentes terem $\nu \geq 1$ é dada por $(1 - \text{p\_amostra})^{1000}$.

Assim, a probabilidade de pelo menos uma das 1000 amostras ter $\nu = 0$ é dada por: $1 - [(1 - \text{p\_amostra})^{1000}]$

```r
result = 1 - (1 - p_amostra)**1000
print(result)
```

```
## [1] 0.2886312
```

Resposta: letra c

Questões práticas

Resolução utilizando linguagem de programação R

Importando pacotes

```r
require(ggplot2)
```

Como estamos trabalhando em duas dimensões, dados x_1 e x_2 dois pontos quaisquer em $\mathbb{R}^2 \cap [-1, 1]$, uma reta do tipo $y = ax + b$ é definida univocamente por eles.

```r
# gerar função f
gerar_f <- function() {
  x1 <- runif(2, -1, 1)
  x2 <- runif(2, -1, 1)
# calcular a e b, coeficiente angular e intercepto, respectivamente, da reta (f) passando por x1 e x2
  a <- (x2[2] - x1[2]) / (x2[1] - x1[1])
  b <- x1[2] - a * x1[1]
  return(list(x1 = x1, x2 = x2, a=a, b=b))
}
f <- gerar_f()
print(f)
```

```
## $x1
## [1] -0.2291256  0.1109874
##
## $x2
## [1] 0.7725196 0.9384577
##
## $a
## [1] 0.8261111
##
## $b
## [1] 0.3002706
```

Função para gerar N pontos em $\mathbb{R}^2 \cap [-1, 1]$

```r
gerar_dados <- function(N) {
    X <- matrix(runif(2*N, -1, 1), ncol = 2)
    colnames(X) <- c("x1", "x2")
    rownames(X) <- paste0("x", 1:N)
    return(X)
}
```

Avaliar os dados utilizando a função sinal. Atribuiremos 1 se o ponto estiver acima da reta da função alvo e -1 caso contrário.

```
avaliar_dados <- function(X,a = f$a, b = f$b){
  #o ponto receberá sinal 1 se estiver acima da reta, e -1 caso contrário
  sinal <- ifelse(X[,2] - a*X[,1] - b > 0, 1, -1)
}
```

Algoritmo perceptron

```
perceptron <- function(X, sinal, taxa_aprendizagem = 1) {
  #adicionar coordanada artificial "x0 = 1"
  X <- cbind(1, X)
  # inicializar vetor de pesos com zeros
  w <- rep(0, ncol(X))
  # inicializar número de iterações
  iters <- 0

  #iniciar vetor de sinais preditos com 0 conforme enunciado
  sinal_pred <- rep(0, length(sinal))

  # iterar enquanto algum componente do vetor de sinais do modelo for diferente do esperado
  while (any(sinal_pred != sinal)) {

    #atualizar sinais
    sinal_pred <- X %*% w
    sinal_pred <- ifelse(sinal_pred > 0, 1, -1)

    #encontrar dados classificados erroneamente
    erro_classificacao <- which(sinal != sinal_pred)
    if (length(erro_classificacao) > 0) { # garantir que não haja iteração extra.
      i <- erro_classificacao[1] # selecionar o primeiro dado classificado erradamente
      w <- w + taxa_aprendizagem * sinal[i] * X[i,] # atualizar vetor de peso
      iters <- iters + 1 # atualizar número de iterações
    }

  }

  # retornar vetor de pesos final e número de iterações
  return(list(w = w, iters = iters))
}
```

Exemplo ilustrativo

Neste exemplo iremos gerar 100 observações para treinar o modelo perceptron, e representar graficamente os pontos e as funções alvo e função g modelada pelo algoritmo.

Avaliando dados

```
f <- gerar_f()
X <- gerar_dados(100)
sinal <- avaliar_dados(X)
print(cbind(X,sinal)[1:10, ]) # Visualizar apenas os 10 primeiros
```

```
##              x1          x2 sinal
## x1   0.03428450  0.94748578     1
## x2   0.18171000 -0.07037695     1
## x3   0.88250620 -0.25338279    -1
## x4  -0.02070891 -0.92454240     1
## x5   0.01820447  0.63349807     1
## x6   0.78859316  0.70954408     1
## x7  -0.86271515  0.61579227     1
## x8  -0.36701681 -0.96258180     1
## x9  -0.39024642  0.39075043     1
## x10  0.90912834 -0.08164908     1
```

Dado o vetor de pesos w, podemos chegar na fórmula analítica da reta g encontrada pelo modelo. $w_0x_0 + w_1x_1 + w_2x_2 = 0 \iff x_2 = \frac{-w_0x_0 - w_1x_1}{w2}$ Neste exemplo, temos o seguinte vetor de pesos e intercepto e coeficiente angular da função g:

```
w <- perceptron(X,sinal)$w # no R o primeiro índice é 1, não 0
coef_angular <- -w[2] / w[3]
intercepto <- -w[1]/ w[3]

w
```

```
##                  x1        x2
##  4.000000 -4.037289  3.963617
```

```
coef_angular
```

```
##       x1
## 1.018587
```

```
intercepto
```

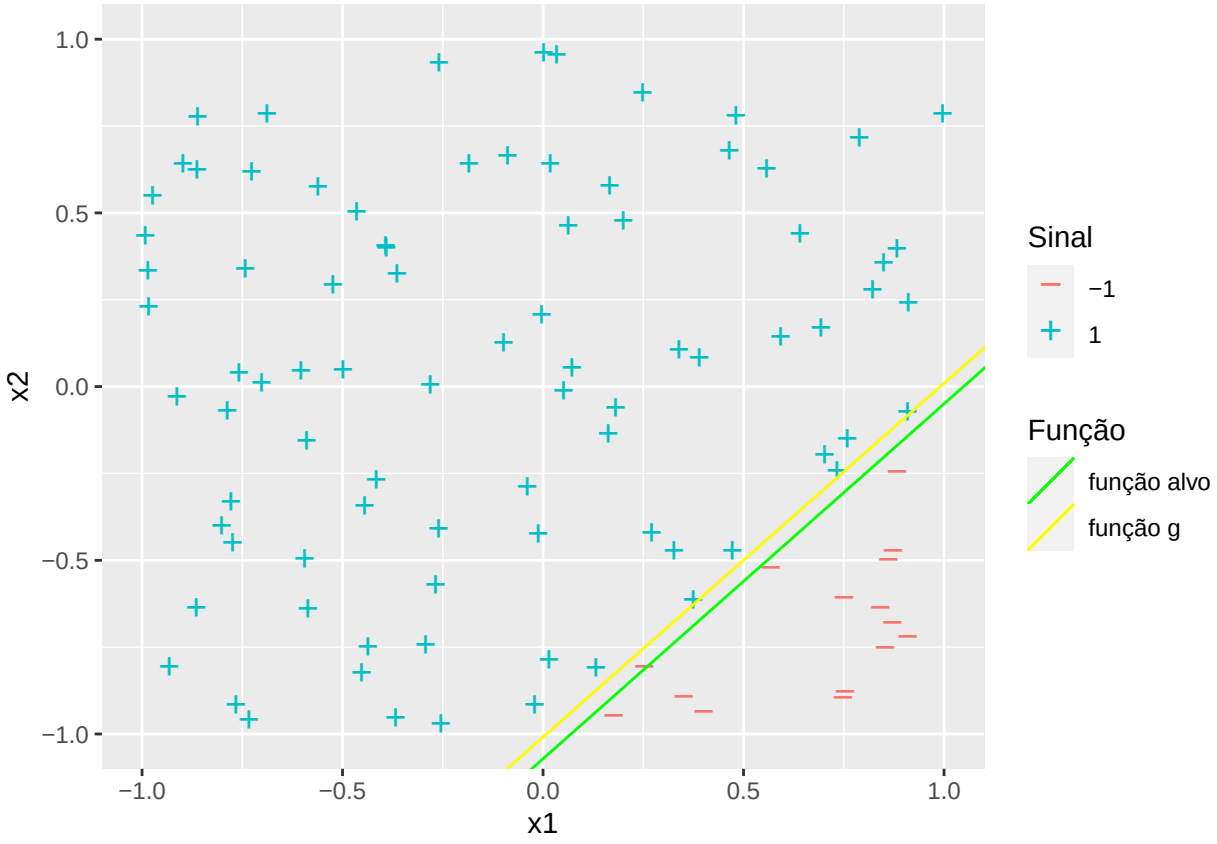
```
##
## -1.009179
```

Ilustrando o problema de exemplo

```
# Criando dataframe para facilitar a representação gráfica
df <- data.frame(X, sinal)

# Gerando gráfico
ggplot(df, aes(x=X[,1], y=X[,2], size = 3,color=factor(sinal), shape = factor(sinal))) +
  geom_point(show.legend = TRUE) +
  xlim(-1, 1) + ylim(-1, 1) +
  geom_abline(aes(slope = f$a, intercept = f$b, linetype = "função alvo"), color = "green") +
```

```
  geom_abline(aes(slope = coef_angular, intercept = intercepto, linetype = "função g"),
              color = "yellow") +
  scale_shape_manual(values = c("-", "+"), name = "Sinal") +
  scale_linetype_manual(values = c("solid", "solid"), name = "Função") +
  guides(color = guide_legend(override.aes = list(shape = c("-", "+"), size = 5))) +
  guides(linetype = guide_legend(override.aes = list(color = c("green", "yellow"),
                                                     shape = c(NA, NA)))) +
  labs(color = "Sinal", shape = "Sinal", linetype = "Função", x = "x1", y = "x2") +
  scale_size(guide = FALSE)
```



Questão 7

Take N = 10. How many iterations does it take on average for the PLA to converge for N = 10 training points? Pick the value closest to your results (again, 'closest' means: |your answer - given option| is closest to 0).

```
iters <- c()
for (i in 1:1000){
  f <- gerar_f()
  X_treino <- gerar_dados(10)
  sinal <- avaliar_dados(X_treino)
  iters[i] <- perceptron(X_treino ,sinal)$iters
```

```
}
summary(iters)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    3.00    6.00   11.86   11.00 1025.00
```

O número de iterações médio está mais próximo de 15.

Resposta: letra b

Questão 8

Para testar se f e g são diferentes vamos utilizar 1000 iterações. Em cada uma delas geraremos uma função alvo e amostras de 10 observações, aplicaremos o perceptron utilizando a função alvo e a amostra, obtendo a função g definida pelo modelo. Após isso, geraremos novas 100 mil observações, e verificaremos quantas delas seriam classificadas diferentemente por f e por g.

```
prob_erro <- c()
for (i in 1:1000){
  f <- gerar_f()
  X <- gerar_dados(10)
  sinal <- avaliar_dados(X)
  w <- perceptron(X,sinal)$w
  coef_angular <- -w[2] / w[3]
  intercepto <- -w[1]/ w[3]

  n_observacoes = 100000
  X_teste <- gerar_dados(n_observacoes)
  sinal_teste <- avaliar_dados(X_teste)
  sinal_pred_teste <- avaliar_dados(X_teste, a = coef_angular, b = intercepto)
  prob_erro[i] = length(which(sinal_teste != sinal_pred_teste))/n_observacoes
}
summary(prob_erro)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.04607 0.09144 0.16369 0.16248 0.98474
```

A média das 1000 iterações está mais próxima de 0,1.

Resposta: letra c

Questão 9

Análoga à questão 7, porém desta vez o treinamento é feito com amostras de treino de tamanho 100.

```r
iters <- c()
for (i in 1:1000){
  f <- gerar_f()
  X_treino <- gerar_dados(100)
  sinal <- avaliar_dados(X_treino)
  iters[i] <- perceptron(X_treino,sinal)$iters
}
summary(iters)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    36.0    72.0   202.9   158.2 25731.0
```

O número de iterações médio está mais próximo de 100.

Resposta: letra b

Questão 10

Análoga à questão 8, porém desta vez o treinamento é feito com amostras de tamanho 100.

```r
prob_erro <- c()
for (i in 1:1000){
  f <- gerar_f()
  X_treino <- gerar_dados(100)
  sinal <- avaliar_dados(X_treino)
  w <- perceptron(X_treino,sinal)$w
  coef_angular <- -w[2] / w[3]
  intercepto <- -w[1]/ w[3]

  n_observacoes = 100000
  X_teste <- gerar_dados(n_observacoes)
  sinal_teste <- avaliar_dados(X_teste)
  sinal_pred_teste <- avaliar_dados(X_teste, a = coef_angular, b = intercepto)
  prob_erro[i] = length(which(sinal_teste != sinal_pred_teste))/n_observacoes

}
summary(prob_erro)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00006 0.00647 0.01160 0.02181 0.01944 0.99185
```

A média das 1000 iterações está mais próxima de 0,01.

Resposta: letra b