Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Programa de Pós-Graduação em Matemática
Aplicada e Estatística
Mestrado em Matemática Aplicada e Estatística

Seleção de variáveis usando o algoritmo genético

Matheus Henrique Tavares Pinto

Natal – RN

Fevereiro de 2022

Matheus Henrique Tavares Pinto

Seleção de variáveis usando o algoritmo genético

Trabalho apresentado ao Programa de Pós-Graduação em Matemática Aplicada e Estatística da Universidade Federal do Rio Grande do Norte, em cumprimento com as exigências legais para obtenção do título de Mestre.
Área de Concentração: Probabilidade e Estatística.
Linha de Pesquisa: Processos Estocásticos.

Universidade Federal do Rio Grande do Norte

Centro de Ciências Exatas e da Terra

Programa de Pós-Graduação em Matemática Aplicada e Estatística

Mestrado em Matemática Aplicada e Estatística

Orientador

Dr. André Gustavo Campos Pereira

Natal – RN

Fevereiro de 2022

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Setorial Prof. Ronaldo Xavier de Arruda - CCET

Pinto, Matheus Henrique Tavares.
Seleção de variáveis usando o algoritmo genético / Matheus Henrique Tavares Pinto. - 2022.
56f.: il.

Dissertação (Mestrado) - Universidade Federal do Rio Grande do Norte, Centro de Ciências Exatas e da Terra, Programa de Pós-Graduação em Matemática Aplicada e Estatística. Natal, 2022.
Orientador: Dr. André Gustavo Campos Pereira.

1. Estatística matemática - Dissertação. 2. AGE - Dissertação. 3. AIC - Dissertação. 4. Seleção de variáveis - Dissertação. 5. Processos estocásticos - Dissertação. 6. Cadeias de Markov - Dissertação. I. Pereira, André Gustavo Campos. II. Título.

RN/UF/CCET CDU 519.2

Elaborado por Joseneide Ferreira Dantas - CRB-15/324

Dissertação de Mestrado sob o título *Seleção de variáveis usando o algoritmo genético* apresentada por Matheus Henrique Tavares Pinto e aceita pelo Programa de Pós-Graduação em Matemática Aplicada e Estatística da Universidade Federal do Rio Grande do Norte, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

Dr. André Gustavo Campos Pereira
Orientador
Departamento de Matemática
Universidade Federal do Rio Grande do Norte

Dr. Fidel Ernesto Castro Morales
Departamento de Estatística
Universidade Federal do Rio Grande do Norte

Dr. Manoel Ferreira dos Santos Neto
Unidade Acadêmica de Estatística
Universidade Federal de Campina Grande

Natal – RN
Fevereiro de 2022

Dedico o presente trabalho a Eliene Tavares, a quem agradeço o apoio incondicional.

Agradecimentos

De forma geral, agradeço a todos que direta ou indiretamente me apoiaram ou contribuiram para a criação do presente trabalho. Em especial, quero agradecer a algumas pessoas e instituições cujo nível de apoio e contribuição foram essenciais para sua realização:

- Primeiramente agradeço ao meu orientador, Dr. Andre Gustavo Campos Pereira. Sua orientação e dedicação foram fundamentais para que eu alcançasse êxito em todas as etapas da execução deste trabalho.
- Aos docentes do departamento de Matemática da UFRN e do PPGMAE que contribuíram para minha formação acadêmica.
- À Universidade Federal do Rio Grande do Norte pela oportunidade de ter acesso a um ambiente acadêmico propício ao desenvolvimento intelectual, social e profissional.

"If nature were not beautiful;
it would not be worth knowing;
and if nature were not worth knowing;
life would not be worth living."
(Henri Poincare)

Resumo

Muitos problemas práticos envolvendo modelos lineares em algum momento necessitam de uma redução do número de varíaveis envolvidas, seja pelo custo envolvido em se trabalhar com muitas variáveis, seja por que uma certa quantidade de variáveis já explica satisfatoriamente o problema abordado. Podemos citar entre outras técnicas a análise de componentes principais, a seleção do melhor subconjunto de variáveis, a seleção progressiva de variáveis, etc. Nesse trabalho apresentaremos como proceder a seleção de variáveis de um modelo linear utilizando o algoritmo genético . Além disso, mostramos que o algoritmo genético elitista (AGE) converge para o conjunto das populações contendo uma solução do problema de otimização considerado, ao mesmo tempo, mostramos como usar a convergência do AGE para obter soluções para o problema de seleção de variáveis.

Palavras-chave: 1. AGE 2. AIC 3. Seleção de variáveis 4. Processos estocásticos 5. Cadeias de Markov

Abstract

Many practical problems involving linear models has a step that consists in reducing the number of variables of the model, either it is very expensive to deal with too many variables or because some of the variables are able to explain the response satisfactorily. We can cite among such methods of reducing the number of variables of a linear model, the principal component analysis, best subset selection, forward stepwise selection, etc. In this work, we present how to use the elitist genetic algorithm in order to select a collection of variables for a linear model. Besides that, we show the convergence of the elitist genetic algorithm to the set of all possible populations containing a solution of the problem under study, at the same time we will obtain solutions to the variable selection problem using the convergence of the elitist genetic algorithm.

Keywords: 1. AGE 2. AIC 3. Variable selection 4. Stochastic processes 5. Markov Chain

Lista de Tabelas

Tabela 1 – Codificação dos pontos da primeira e segunda entradas da discretização 19
Tabela 2 – Valor de imagem dos pontos na população inicial $P^{(0)}$ 20
Tabela 3 – Aplicação da seleção nos pontos da população inicial $P^{(0)}$ 21
Tabela 4 – valor de imagem dos elementos de $P^{(3)}$ pela função f; 24

Sumário

1 INTRODUÇÃO 10

2 ALGORITMOS GENÉTICOS 13
2.1 Introdução 13
2.2 Cadeias de Markov 14
2.3 Algoritmo genético 17
2.3.1 População inicial 19
2.3.2 Seleção 20
2.3.3 Cruzamento 21
2.3.4 Mutação 23
2.4 Classificação dos algoritmos genéticos 26

3 SELEÇÃO DE VARIÁVEIS 30
3.1 Introdução 30
3.2 Seleção de variáveis 30
3.3 O critério da informação de Akaike 34
3.4 Algoritmos genéticos e seleção de variáveis 36

4 APLICAÇÃO DO ALGORITMO GENÉTICO ELITISTA EM UM PROBLEMA DE SELEÇÃO DE VARIÁVEIS 39

REFERÊNCIAS BIBLIOGRÁFICAS 48

APÊNDICE A – IMPLEMENTAÇÃO DO ALGORITMO GENÉTICO ELITISTA 51

1 Introdução

O advento da era da informação provocou uma grande transformação científico-tecnológica no que tange a grande disponibilidade de dados a serem armazenados, processados e transmitidos. Tal transformação, acarretou como consequência o aumento dos desafios inerentes ao processamento computacional de modelos estatísticos relacionando as variáveis dos dados considerados no formato de banco de dados. Os desafios que surgiram para obter modelos relacionando uma variável Y com as demais variáveis $X_1, \ldots, X_n$ no mesmo banco de dados levou ao surgimento do problema de seleção de variáveis (AHMED, 2014). Esse problema, consiste em buscar um subconjunto de $X_1, \ldots, X_n$ que permita descrever ou modelar Y partindo do princípio que em grandes quantidades de informação há redundâncias ou irrelevâncias nas variáveis.

O problema de seleção de variáveis tornou-se especialmente importante no contexto dos modelos lineares (RAO; TOUTENBURG, 1995) porque lidar com modelos com uma grande quantidade de variáveis pode ser teórica e computacionalmente difícil. Devido a esses fatores, surgiram muitas técnicas visando resolver esse problema no contexto linear. Porém, tais técnicas apresentam muitas restrições teóricas (como diferenciação ou convexidade) e práticas (como exigência de muitos recursos computacionais e instabilidade na convergência para uma solução) nos modelos. Assim, apresentamos neste trabalho uma técnica alternativa para seleção de variáveis que permite evitar tais restrições: o algoritmo genético.

Visando contornar os problemas citados acima, muitas técnicas foram desenvolvidas para resolver o problema de seleção de variáveis. Dentre as técnicas existentes podemos citar: RIDGE, análise de componentes principais, LASSO, seleção progressiva e regressiva de variáveis e seleção do melhor subconjunto de variáveis (RISH; GRABARNIK, 2014) (JAMES et al., 2013) . Especificamente, mostraremos que uma classe particular de algoritmo genético pode performar seleção de variáveis: o algoritmo genético elitista (GOLDBERG, 1989).

O algoritmo genético elitista faz parte de uma classe de algoritmos de otimização estocástica. A subclasse desses algoritmos tratada neste trabalho é caracterizada por ser um tipo de processo estocástico conhecido como Cadeia de Markov homogênea multifásica. A convergência desta classe de algoritmos é garantida pela convergência em probabilidade do tipo de Cadeia de Markov (KARR, 1978) que os caracterizam.

Veremos que a existência da distribuição assintótica (ou distribuição limite) associada a Cadeia de Markov será crucial para garantir a convergência do AGE. A convergência das distribuições da cadeia para a distribuição assintótica nem sempre ocorre e o estudo

de condições necessárias e suficientes para obter essa convergência torna-se necessário para que possamos garantir a convergência da Cadeia. Mostramos que quando a Cadeia de Markov é finita, homogênea, irredutível, aperiódica e fechada a convergência ocorre em tempo finito. Sabendo disso, verificamos que o AGE é uma Cadeia de Markov que satisfaz todas as condições citadas anteriormente e portanto converge para uma população que contêm as soluções para o problema de otimização.

Sabendo que o AGE converge para o conjunto das populações contendo a solução do problema de otimização, precisamos especificar como o algoritmo será configurado para seu uso em problemas práticos. A solução do problema de seleção de variáveis usando o AGE depende da representação do problema e da função objetivo a ser otimizada. Neste trabalho, mostramos que uma representação binária do problema através de um vetor binário mostrando a pertinência ou impertinência de uma variável na solução torna maior a interpretabilidade do AGE e ao mesmo tempo simplifica a interpretabilidade do modelo linear ajustado as variáveis selecionadas pelo AGE. Ao mesmo tempo, o uso do estimador da distância de Kullback-Leibler conhecido como critério da informação de Akaike ou AIC (ANDERSON; BURNHAM; ANDERSON, 1998) (SAKAMOTO; ISHIGURO; KITAGAWA, 1986) para ser a função objetivo do AGE nos permite performar a seleção de variáveis de forma consistente e dirigida, evitando as incertezas e intratabilidade associadas a este tipo de problema.

O conjunto de dados *Credit* é um dos conjuntos de dados presentes na biblioteca *ISLR* (ISLR. . . ,) da linguagem de progrmação *R* (TEAM et al., 2013) (COMPUTING, 2018). Este conjunto é composto por 400 exemplos descritos por 12 variáveis. No texto (JAMES et al., 2013) é mostrado que ao usar seleção progressiva e regressiva de variáveis para selecionar dentre as 11 primeiras variáveis descrevendo os exemplos de *Credit* para prever as observações da variável restante, obtemos soluções diferentes para o mesmo problema contrariando o fato de que existe única solução exata para o mesmo. Tal inconsitência nos motivou a usar o AGE para resolver o problema de seleção de variáveis, visto que possuímos garantia de convergência para uma solução desse problema.

Como parte final do trabalho, mostramos a aplicação do AGE para selecionar variáveis no conjunto de dados *Credit* (conjunto este caracterizado por soluções inconsistentes quando aplicamos outras técnicas de solução) para construção de modelos de regressão linear. Observamos em detalhes o funcionamento das etapas do AGE com a representação binária do problema e o uso do AIC como função objetivo. Além disso, concluímos o trabalho mostrando a efetividade do AGE na busca de uma solução do problema de seleção de variáveis. A solução exata do problema foi encontrada e consequentemente encontramos a única configuração de variáveis que minimiza o AIC, provando a versatilidade do algoritmo neste tipo de problema.

Observação. No livro (JAMES et al., 2013) é possível encontrar um exemplo onde o

uso de técnicas diferentes de seleção de variáveis pode levar a soluções inconsistentes. Especificamente, no exemplo mostrado no livro é aplicado a seleção regressiva e progressiva de variáveis sobre o conjunto de dados *Credit* no mesmo contexto explicado acima, ou seja, usar as 11 primeiras variáveis para encontrar um subconjunto que permita descrever as observações da variável restante de *Credit*.

2 Algoritmos genéticos

2.1 Introdução

Dada uma função positiva $f : D \subset \mathbb{R}^n \to \mathbb{R}$, todas as vezes que falarmos em problema de otimização nesse trabalho, estaremos nos referindo a um dos seguintes problemas:

$$x^* = \arg\max_{x \in D} f(x) \quad \text{ou} \quad x^* = \arg\min_{x \in D} f(x).$$

Neste capítulo serão apresentados os fundamentos teóricos necessários para a compreensão de uma classe de algoritmos denominada de algoritmos genéticos (GOLDBERG, 1989). Tal classe é constituída por algoritmos que buscam imitar ou repetir mecanismos de evolução biológica como fontes alternativas para a construção de soluções de problemas de otimização.

O algoritmo genético é uma subclasse dos algoritmos evolutivos (algoritmos baseados em processos evolutivos naturais) desenvolvidos para analisar e resolver problemas de otimização. Tais algoritmos se utilizam de um viés metaheurístico, ou seja, durante o processo de otimização a busca por pontos de ótimo é realizada através da amostragem de um subconjunto de pontos de D (domínio da função considerada no problema) evitando assim o custo e a ineficiência computacional de uma busca generalizada em todos os elementos do domínio.

Os algoritmos genéticos possuem características que os diferenciam dos demais, por exemplo: possuem estratégias implícitas específicas para buscas dirigidas de pontos de ótimo, devido ao viés metaheurístico, esses algoritmos são projetados para buscar pontos de ótimo de forma eficiente, visam buscar pontos próximos dos ótimos através de procedimentos randomizados baseados em amostragem e aplicações de transformações sobre os elementos obtidos na amostragem. Os algoritmos genéticos permitem uma abordagem generalista em problemas de otimização evitando assim exigências comuns em outros procedimentos de otimização como continuidade, diferenciabilidade ou convexidade das funções envolvidas na otimização.

Na próxima seção introduzimos a teoria de Cadeias de Markov necessária para entender a modelagem do algoritmo genético, assim como compreender quais são as suas propriedades que garantem sua convergência.

2.2 Cadeias de Markov

Cadeias de Markov, martingais, movimento browniano, etc, todos são processos estocásticos que possuem alguma propriedade que os caracterizam. Dessa forma, para entendermos o que são Cadeias de Markov precisaremos antes entender o que são processos estocásticos.

Definição 2.2.1. Um processo estocástico $\{X_n\}_{n\in\mathbb{N}}$ é uma cadeia de Markov se satisfaz :

$$P(X_n = i_n \mid X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = P(X_n = i_n \mid X_{n-1} = i_{n-1})$$

para todo vetor de estados $(i_0, \ldots, i_n)$ em que $\{i_0, \ldots, i_n\} \subset S$(em que S é o espaço de estados do processo). Assim, a propriedade que caracteriza uma Cadeia de Markov é que o estado futuro do processo depende apenas do estado presente.

A descrição de um processo estocástico $\{X_n\}_{n\in\mathbb{N}}$ é baseada no conhecimento das probabilidades

$$P(X_0 = i_0, \ldots, X_n = i_n),$$

para todo $n \in \mathbb{N}$ (ou seja, conhecer a distribuição dos estados do sistema). De acordo com o Teorema da multiplicação:

$$P(X_n = i_n, \ldots, X_0 = i_0)$$

$$= P(X_0 = i_0)P(X_1 = i_1 \mid X_0 = i_0) \ldots P(X_n = i_n \mid X_{n-1} = i_{n-1}, \ldots, X_0 = i_0).$$

Por sua vez, se esse processo for uma Cadeia de Markov, a expressão anterior pode ser reescrita da seguinte maneira

$$P(X_n = i_n, \ldots, X_0 = i_0)$$

$$= P(X_0 = i_0)P(X_1 = i_1 \mid X_0 = i_0) \ldots P(X_n = i_n \mid X_{n-1} = i_{n-1}).$$

Logo, quando estamos trabalhando com Cadeias de Markov, a fim de calcularmos todas as distribuições finitas ($P(X_n = i_n, \ldots, X_0 = i_0)$), precisamos tanto conhecer $P(X_0 = i_0)$ quanto $P(X_n = i_n \mid X_{n-1} = i_{n-1})$ para todo $n \in \mathbb{N}$ e $i_0, \ldots, i_n \in S$.

Definição 2.2.2. Seja $S = \{1, \ldots, m\}$ o espaço de estados finito de uma Cadeia de Markov. Uma distribuição de probabilidade inicial para esse espaço é um vetor

$$\pi^0 = (\pi^0(1), \ldots, \pi^0(m))$$

tal que $\sum_{i=1}^{m} \pi^0(i) = 1$.

Assim, quando estamos considerando Cadeias de Markov finitas com espaço de estados $S = \{1, \ldots, m\}$ a distribuição de probabilidade de X_0 é a distribuição inicial da cadeia. Em outras palavras, temos uma distribuição $\pi^0 : S \to [0, 1]$ tal que

$$\pi^0(i) = P(X_0 = i).$$

Agora precisamos olhar para as probabilidades de transição

$$P(X_n = i_n \mid X_{n-1} = i_{n-1}) \quad \forall\, n \in \mathbb{N} \text{ e } i_n, i_{n-1} \in S$$

a fim de sermos capazes de calcular as distribuições finitas da cadeia de Markov.

As Cadeias de Markov são classificadas em duas classes baseado na dependência das probabilidades de transição do índice $n \in \mathbb{N}$.

Definição 2.2.3. Uma Cadeia de Markov $\{X_n\}_{n\in\mathbb{N}}$ é homogênea quando as probabilidades de transição

$$p_{ij}^{(n-1,n)} = P(X_n = i_n \mid X_{n-1} = i_{n-1})$$

não dependem dos índices n, ou seja,

$$p_{ij} = P(X_n = j \mid X_{n-1} = i) = P(X_{n+k} = j \mid X_{n+k-1} = i),$$

para todo $k \in \{-n + 1, -n + 2, \ldots\}$. Quando isso ocorre, não usamos mais o índice n e p_{ij} passa a representar a probabilidade de transição da cadeia do estado i para o estado j, independente do passo em que essa transição ocorra. Por outro lado, quando a probabilidade de transição $p_{ij}^{(n-1,n)}$ depende do tempo n a Cadeia de Markov é denominada de não-homogênea.

Definição 2.2.4. Seja $\{X_n\}_{n\in\mathbb{N}}$ uma Cadeia de Markov homogênea no tempo com espaço de estados $S = \{1, \ldots, m\}$. Vemos que existem m^2 probabilidades de transição $p_{ij} = P(X_n = j \mid X_{n-1} = i)$. A matriz quadrada P de dimensão $m \times m$ cujas entradas são p_{ij} é chamada matriz de transição da cadeia de Markov $\{X_n\}_{n\in\mathbb{N}}$. Iremos representar tal matriz por

$$P = \begin{bmatrix} p_{11} & \cdots & p_{1m} \\ \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mm} \end{bmatrix}.$$

Note que $\sum_{j=1}^{m} p_{ij} = 1$ para todo $1 \leq i \leq m$ (matrizes satisfazendo tais propriedades são chamadas de matrizes estocásticas). Além disso, toda matriz estocástica é a matriz de transição de alguma Cadeia de Markov finita homogênea no tempo.

Exemplo 2.2.1 (Caminhada aleatória com fronteira refletora). Considere o conjunto $S = \{1, \ldots, m\}$ e uma partícula se movendo entre os elementos de S da seguinte forma: Se a partícula estiver nas extremidades, ela sempre vai deslocar para o ponto de S mais próximo dela com probabilidade 1. Se a partícula estiver em qualquer outro ponto de S, ela se moverá com probabilidade p para o ponto mais à direita e com probabilidade $1-p$ para o ponto mais à esquerda. Se definirmos a coleção de eventos por $\{X_k = i, i \in S\}$ como sendo a partícula estar no tempo k no estado i, tal processo estocástico representa o movimento da partícula e as probabilidades de transição (da forma como o processo foi definido) serão dadas pela matriz de transição

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 1-p & 0 & p & 0 & 0 & 0 \\ 0 & 1-p & 0 & p & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1-p & 0 & 1 \\ 0 & \cdots & 0 & 0 & 1-p & 0 \end{bmatrix}.$$

Definição 2.2.5. Definimos a probabilidade de transição de um estado i para um estado j em n passos como

$$p_{ij}^{(n)} = P(X_n = j \mid X_0 = i).$$

Com base na definição acima e no teorema da probabilidade total

$$P(X_n = j) = \sum_{i \in S} P(X_0 = i) p_{ij}^{(n)}.$$

Teorema 2.2.1 (Chapman-Kolmogorov)**.** Seja $\{X_n\}_{n\in\mathbb{N}}$ uma Cadeia de Markov homogênea, então vale para todos os estados $i, j \in S$ que

$$p_{ij}^{(n+m)} = \sum_{k \in S} p_{ik}^{(n)} p_{kj}^{(m)}.$$

Observação. Se P é a matriz de transição da Cadeia de Markov homogênea $\{X_n\}_{n\in\mathbb{N}}$, a probabilidade $p_{ij}^{(n+m)}$ definida acima é o elemento (i, j) da matriz P^{n+m}.

Com base na distribuição de probabilidade inicial $P(X_0 = i) = \pi^0(i)$ e no Teorema de Chapman-Kolmogorov podemos obter a distribuição de probabilidade γ_n dos estados da cadeia em um passo n como o produto matricial

$$\gamma_n = \pi^0 P^n.$$

Definição 2.2.6. Seja P a matriz de transição de uma Cadeia de Markov homogênea $\{X_n\}_{n\in\mathbb{N}}$. Diz-se que a cadeia $\{X_n\}_{n\in\mathbb{N}}$ possui distribuição assintótica (ou é ergódica) quando existe o limite

$$\lim_{n\to\infty} p_{ij}^{(n)} = \pi_j \,, \forall j \in S,$$

independente do vetor de distribuição inicial π^0.

Tal distribuição γ_n nos permite compreender como a cadeia está distribuída nos estados do sistema no tempo n. Ao mesmo tempo, computar tal produto matricial para n grande pode ser inviável e o estudo da distribuição assintótica (também chamada de distribuição de probabilidade invariante ou estacionária) torna-se essencial para a compreensão do sistema no longo prazo, ou seja, se $S = \{1, \ldots, m\}$ a distribuição $\pi = (\pi_1, \ldots, \pi_m)$ definida por

$$\lim_{n\to\infty} p_{ij}^{(n)} = \pi_j, \tag{2.1}$$

quando esses limites existem. Dizemos que uma Cadeia de Markov homogênea $\{X_n\}_{n\in\mathbb{N}}$ converge quando possui distribuição assintótica π. Porém, quando a distribuição π obtida por (2.1) depender da distribuição inicial π^0 concluímos que a Cadeia de Markov $\{X_n\}_{n\in\mathbb{N}}$ não converge para uma distribuição fixa. Em virtude de tal observação, pode-se encontrar nos trabalhos (KARR, 1978) e (LAWLER, 2018) condições suficientes para que uma Cadeia de Markov homogênea seja convergente. Especificamente, tais trabalhos mostram que uma cadeia homogênea, finita, aperiódica e irredutível é convergente.

2.3 Algoritmo genético

Nessa seção vamos explicar a dinâmica realizada pelo algoritmo genético na busca de um ponto de ótimo global supondo a existência de tal ponto. Para ficar mais visível o que acontece em cada etapa interna do algoritmo genético vamos supor que o domínio da função f é da forma $D = [a_1, b_1] \times \cdots \times [a_n, b_n]$. O primeiro passo que o algoritmo genético faz é uma discretização de D, chamado de espaço de busca. A partir dessa discretização, o espaço de busca será o conjunto onde o algoritmo busca a solução do problema. Quando o ponto de ótimo é um dos pontos da discretização de D, o algoritmo genético encotra-o ao final do processo de busca. Para aumentar a chance de que o ponto de ótimo pertença a discretização, toma-se a discretização mais fina possível.

A discretização de D é feita da seguinte maneira: para cada intervalo $[a_i, b_i]$ obtemos uma quantidade de pontos que seja potência de 2 (isso vai ficar claro mais adiante), ou seja, em uma quantidade do tipo 2^l. Dessa forma, obtemos uma discretização de D através do produto cartesiano dos pontos obtidos de cada intervalo $[a_i, b_i]$. Em outras palavras, o nosso espaço de busca será uma malha obtida pelas n-úplas formadas pelos pontos obtidos da discretização de cada intervalo do produto cartesiano que determina D.

Exemplo 2.3.1. Suponha que $f : [0, 1.5] \times [3, 4.5] \to \mathbb{R}$. Caso se decida discretizar uniformemente cada um dos intervalos em 2^2 pontos, teríamos uma discretização de $[0, 1.5]$ dada por $\{0, 0.5, 1, 1.5\}$ e de $[3, 4.5]$ dada por $\{3, 3.5, 4, 4.5\}$. A malha formada por tal partição é

$$\begin{aligned} S = \; & \{(0,3),(0,3.5),(0,4),(0,4.5),(0.5,3),(0.5,3.5),(0.5,4),(0.5,4.5),(1,3), \\ & ,(1,3.5),(1,4),(1,4.5),(1.5,3),(1.5,3.5),(1.5,4),(1.5,4.5)\} \end{aligned}.$$

Uma vez que o espaço de busca esteja definido, o próximo passo é definir a quantidade de elementos que farão parte da população. A população é formada por pontos de nosso espaço de busca e é a partir de modificações que ocorrerão nesses pontos, que o algoritmo busca o ponto de ótimo procurado. Uma vez escolhido o tamanho da população com a qual trabalharemos, o próximo passo é escolher aleatoriamente essa quantidade de pontos do espaço de busca para montarmos nossa população inicial. As modificações que ocorrerão nos pontos da população serão via a aplicação dos operadores de seleção, cruzamento e mutação nessa população. Veremos que após a aplicação desses operadores, um após o outro, obteremos uma nova população e novamente aplicaremos esses operadores na nova população e esse processo se repete até que algum critério de parada pré-estabelecido seja atingido (geralmente considera-se o número de iterações).

Até esse momento, decidimos então o espaço de busca S que é a malha construída a partir da discretização dos intervalos e o número de elementos que a população vai conter, por exemplo K. Assim, podemos montar o conjunto formado por todas as possíveis populações de K elementos, em que cada elemento é um ponto de S,

$$E = \{(x_1, x_2, ..., x_K)/x_i \in S, i = 1, 2, ..., K\}.$$

Com base na definição de E, podemos associar cada ponto $(x_1, x_2, ..., x_K)$ a uma representação binária. Tal representação adequa o formato das populações em E para posterior aplicação dos operadores genéticos de cruzamento e mutação conforme será justificado. Para facilitar a explicação, imaginemos que cada intervalo $[a_i, b_i]$ que compõe D, sejam divididos uniformemente em 2^l pontos. Com isso, podemos fazer a associação de cada ponto obtido com um vetor binário de l coordenadas da seguinte maneira:

$$a_i \leftrightarrow (0, 0, ..., 0), a_i + \frac{(b_i - a_i)}{2^l - 1} \leftrightarrow (0, 0, ..., 0, 1), ..., b_i \leftrightarrow (1, 1, ..., 1).$$

Dessa forma, cada ponto da malha pode ser associado a um vetor binário de nl coordenadas, onde n é o número de intervalos cujo cartesiano forma o D e l é o número de coordenadas do vetor que está associado a cada elemento de um dado intervalo $[a_i, b_i]$. Note que se dividirmos cada intervalo numa quantidade diferente, cada coordenada de um ponto da malha será associado a um vetor binário de dimensão diferente. Contudo, isso não impossibilita o desenvolvimento das etapas seguintes, como veremos.

Com uma população inicial estabelecida, podemos começar a fazer as modificações via os operadores de seleção, cruzamento e mutação. Essas modificações são realizadas da seguinte maneira: o operador de seleção é aplicado na população e isso faz com que a população sofra modificações. Essa população modificada passa pelo processo de cruzamento recebendo novas modificações, sobre as quais o operador de mutação é aplicado. Depois dessa sequência de modificações obtemos outra população que passará por sua vez por esse mesmo processo e isso vai ser repetido até que algum critério de parada seja atingido.

Mais uma vez , para facilitar o entendimento das etapas do algoritmo genético, descreveremos cada etapa baseado num exemplo numérico. Suponhamos que desejamos encontrar o ponto de máximo da função $f : [0,3]\times[1,3] \to \mathbb{R}$ definida por $f(x,y) = x^2+y^2$.

Para isso, particionaremos cada intervalo $[0,3]$ e $[1,3]$ em 2^3 pontos obtendo a Tabela 1

Tabela 1 – Codificação dos pontos da primeira e segunda entradas da discretização

x	identificação x	y	identificação y
0	000	1	000
3/7	001	9/7	001
6/7	010	11/7	010
9/7	100	13/7	100
12/7	011	15/7	011
15/7	101	17/7	101
18/7	110	19/7	110
3	111	3	111

.

A identificação de um par (x,y) da discretização é obtida concatenando a codificação da componente x e da componente y. Por exemplo, se buscamos a identificação de $(\frac{6}{7}, \frac{19}{7})$ concatenamos a identificação de $\frac{6}{7}$ que é 010 com a identificação de $\frac{19}{7}$ que é 110, ou seja, a identificação de $(\frac{6}{7}, \frac{19}{7})$ é 010110. Um vetor binário $a_1a_2a_3b_1b_2b_3$ pode ser decodificado usando sua representação através do par $(\overline{x}, \overline{y})$ em que

$$\overline{x} = \sum_{i=1}^{3} a_i 2^{3-i} \quad \overline{y} = \sum_{i=1}^{3} b_i 2^{3-i}.$$

Com o par $(\overline{x}, \overline{y})$ obtemos o par (x,y) através das relações

$$x = x_{\min} + \overline{x}\left(\frac{x_{\max} - x_{\min}}{2^3 - 1}\right) \quad y = y_{\min} + \overline{y}\left(\frac{y_{\max} - x_{\min}}{2^3 - 1}\right),$$

que é a decodificação desejada.

2.3.1 População inicial

A população inicial consiste em escolher aleatoriamente, com reposição, um subconjunto de K elementos da malha. Dessa forma, podemos ter alguns elementos repetidos na população. Para apresentar as próximas etapas do algoritmo genético vamos supor que obtemos a seguinte população inicial

$$P^{(0)} = \{001100, 101110, 000100, 000000, 110110, 011000\},$$

que corresponde na discretização que fizemos aos pontos em S

$$P^{(0)}_{dual} = \{(3/7, 13/7), (15/7, 19/7), (0, 13/7), (0,1), (18/7, 19/7), (12/7, 1)\}.$$

Calculamos agora o valor da imagem de cada ponto em $P^{(0)}$ por meio da função f conforme mostrado na Tabela 2 (a necessidade de se calcular a imagem pela f dos pontos da população ficará claro mais adiante).

Tabela 2 – Valor de imagem dos pontos na população inicial $P^{(0)}$

Codificação	(x , y)	f (x , y)
001100	$(3/7, 13/7)$	3.632653
101110	$(15/7, 19/7)$	11.95918
000100	$(0, 13/7)$	3.44898
000000	$(0, 1)$	1
110110	$(18/7, 19/7)$	13.97959
011000	$(12/7, 1)$	3.938776

Dessa forma,

$$\sum_{i=1}^{6} f(x_i, y_i) = 37.95918.$$

Observação. Quanto maior for a quantidade de pontos na discretização, maior será a precisão da solução que o algoritmo genético encontrará.

2.3.2 Seleção

A seleção é um operador que tenta imitar a etapa da seleção natural do processo evolutivo. No nosso caso, quando estamos procurando o ponto de máximo de uma função, um ponto ser mais adaptado do que outro significa ter uma imagem maior pela f. Essa etapa, faz então com que aqueles pontos mais aptos continuem a integrar a população enquanto os pontos menos adaptados (de imagem pela f menor) sejam substituídos por cópias desses outros pontos mais adaptados. Para fazer isso, o algoritmo genético nessa etapa, atribui um peso a cada ponto, ou seja, se a população atual é $P = \{x_1, x_2, ..., x_K\}$ cada elemento desse terá o peso

$$p_{x_i} = \frac{f(x_i)}{\sum_{p=1}^{K} f(x_p)}.$$

Existem diversos métodos que podem ser utilizados para selecionar, com maior probabilidade, os elementos da população com maiores valores de imagem. Neste trabalho utilizaremos o método da roleta, que consiste em selecionar os indivíduos da nova população, a qual terá a mesma quantidade de elementos que a população anterior, por meio de sorteio. Como vimos, cada ponto x_i da população tem um peso p_{x_i} e que somados totalizam 1. Dessa forma, podemos definir uma variável aleatória discreta X que assume os valores x_i com probabilidade p_{x_i} e da qual é fácil gerar uma amostra de tamanho K (tamanho da população anterior). Logo, o sorteio nada mais é do que a geração de K valores dessa variável aleatória X. Voltando ao nosso exemplo, mais precisamente à Tabela 2, repetiremos tais valores bem como os pesos de cada um dos indivíduos na Tabela 3.

Tabela 3 – Aplicação da seleção nos pontos da população inicial $P^{(0)}$

Indivíduo	Codificação	(x , y)	f (x , y)	$\frac{f(x_j,y_j)}{\sum f(x_i,y_i)}$
i_1	001100	$(3/7, 13/7)$	3.632653	0.09569893
i_2	101110	$(15/7, 19/7)$	11.95918	0.3150537
i_3	000100	$(0, 13/7)$	3.44898	0.09086023
i_4	000000	$(0, 1)$	1	0.02634409
i_5	110110	$(18/7, 19/7)$	13.97959	0.3682796
i_6	011000	$(12/7, 1)$	3.938776	0.1037635

Espero que agora tenha ficado claro porque precisamos calcular o valor de f em cada ponto da população.

Ao gerarmos uma amostra de tamanho seis de X obtemos uma nova população $P^{(1)}$ dada por

$$P^{(1)} = \{110110, 110110, 110110, 101110, 110110, 101110\},$$

que corresponde na discretização que fizemos a

$$P^{(1)}_{dual} = \{(18/7, 19/7), (18/7, 19/7), (18/7, 19/7), (15/7, 19/7), (18/7, 19/7), (15/7, 19/7)\}.$$

2.3.3 Cruzamento

O operador cruzamento tenta emular a transferência dos genes paternos para a prole que ocorre durante o cruzamento entre indivíduos da mesma espécie. Espera-se que os filhos herdem via tal processo as melhores características de cada progenitor. Assim como na natureza, o operador cruzamento precisa de dois indivíduos para poder atuar. Na literatura dos algoritmos genéticos é natural chamar o vetor binário obtido na codificação de cada elemento de cromossomo e cada bit do vetor de gene. A troca do material genético se dará pela troca dos genes entre os pais, gerando novos elementos que são chamados de filhos. No exemplo que estamos desenvolvendo, cada cromossomo possui 6 genes. Novamente, existem várias formas de procederem os cruzamentos, nesse trabalho usaremos a seguinte forma de gerar os descendentes a partir dos pais. Dado um cromossomo, por exemplo, 011010 vemos que este cromossomo tem 5 possíveis pontos de quebra (pq)

$$0\ pq_1\ 1\ pq_2\ 1\ pq_3\ 0\ pq_4\ 1\ pq_5\ 0.$$

Desta forma, imaginando que dois elementos, por exemplo, 011010 e 110101 são escolhidos para participar do processo de cruzamento. O operador cruzamento agirá da seguinte maneira:

1. Primeiro escolherá aleatoriamente qual ponto de quebra ele irá usar dentre os 5 possíveis.

2. Uma vez escolhido o ponto de quebra, ele manterá inalterado os genes antes do ponto de quebra e trocará os genes depois do ponto de quebra entre os indivíduos, gerando dessa forma dois novos indivíduos que serão chamados de filhos.

No nosso exemplo, se o ponto de quebra sorteado for pq_5 o resultado do operador sobre esses dois pontos será: 011011 e 110100. Note que as cinco primeiras posições (antes de pq_5) foram preservadas nos dois vetores resultantes (vetor inicial - 011010, cinco primeiras posições do vetor resultante - 01101). A última posição do vetor resultante foi a última posição do outro vetor (outro vetor 110101, última posição 1). Ao juntarmos as cinco primeiras posições com esse último bit, teremos o primeiro vetor resultante. A mesma construção se repete no segundo vetor resultante.

Uma vez que sabemos quais vetores sofrerão a aplicação do operador cruzamento, nós sabemos quais possíveis resultados podem ser obtidos dessa operação. O problema é: Como escolhemos os elementos nos quais aplicaremos o operador de cruzamento? O operador cruzamento, ao contrário do de seleção, possui um parâmetro chamado probabilidade de cruzamento, e é esse parâmetro que nos auxiliará na escolha dos elementos que participarão do processo de cruzamento. Novamente, existem várias formas de se fazer isso, nesta dissertação utilizaremos o seguinte método:

1. Primeiro geramos uma amostra de tamanho igual a quantidade de elementos de nossa população (K) de uma Bernoulli de parâmetro p_c.

2. As coordenadas que derem 1, serão os elementos da população que participarão do processo de cruzamento. No exemplo que estamos desenvolvendo (a população é formada por 6 elementos), imagine que ao gerarmos uma amostra de uma Bernoulli de parâmetro p_c obtivéssemos 011101. Isso dirá para nós que o segundo, o terceiro, o quarto e o sexto elemento da população foram escolhidos para participar do processo.

3. Neste trabalho, uma vez determinado os elementos que participarão, os pares de elementos onde o operador seleção será aplicado é montado da seguinte maneira: O primeiro e o segundo escolhidos, o terceiro e o quarto, ... , esse processo continua até o último escolhido. Pode acontecer de ser escolhido um número ímpar de elementos. Nesse caso o último elemento não formará par e portanto voltará para a população sem sofrer mudança.

4. o operador cruzamento é aplicado aos pares e os vetores resultantes substituirão os vetores utilizados para sua obtenção.

Observação. Note que quando existe dois pontos que são iguais (lembre-se que pode haver pontos repetidos na população, principalmente depois da seleção), e estes são escolhidos para o processo de cruzamento, o resultado continua sendo filhos idênticos aos pais. Logo,

se em algum momento da dinâmica do algoritmo genético, chegarmos a uma população formada apenas por repetições de um mesmo ponto, o operador cruzamento e de seleção não conseguem fazer mudança nessa população.

Observação. Note que quanto maior o valor de p_c, mais provável que mais pontos sejam escolhidos para participarem do processo de cruzamento.

Voltando ao nosso exemplo, vemos que depois da seleção nossa população é a $P^{(1)}$. Supondo que ao gerarmos nossa amostra de tamanho 6 (tamanho da população) da Bernoulli(p_c) obtivéssemos 100101. Isso nos diria que o primeiro (110110), o quarto (101110) e o sexto (101110) pontos foram escolhidos para participar do processo de cruzamento. Entretanto, pela forma como se dá a escolha dos pares, o sexto ponto não forma par com ninguém e por isso volta para a sua posição sem ser transformado. Suponha que no passo da escolha do ponto de quebra, o pq_2 tivesse sido o escolhido, ou seja, $11pq_20110$ e $10pq_21110$. Mantendo as informações antes de pq_2 e trocando as informações depois de pq_2 obtemos 111110 e 100110 e portanto a nova população é

$$P^{(2)} = \{111110, 110110, 110110, 100110, 110110, 101110\}.$$

2.3.4 Mutação

A mutação é um operador que tenta emular a mutação que o meio pode provocar no indivíduo. Por exemplo, dizem que no início dos tempos a girafa não tinha um pescoço tão longo e que foi o fato da vegetação disponível ficar cada vez mais alto que provocou ao longo do tempo essa modificação/adaptação (GOLDBERG, 1989). Então, a mutação é vista como um processo que acontece no indivíduo e que a nível dos genes.

O operador mutação assim como o operador cruzamento possui um parâmetro, a saber: a probabilidade de mutação, p_m. A probabilidade de cruzamento ajudava na escolha dos indivíduos que participariam do processo de cruzamento. A probabilidade de mutação vai nos auxiliar na escolha dos genes do indivíduo sofrerão mudanças. Relembrando o processo de cruzamento, era gerado uma amostra de tamanho K (tamanho da população) de uma Bernoulli(p_c). Os indivíduos correspondentes as amostras que resultaram em 1, eram escolhidos para participar. Aqui o processo vai ser muito parecido, só que faremos isso para cada indivíduo. O processo pode ser esquematizado da seguinte maneira:

1. Para cada indivíduo, gere uma amostra de tamanho l (número de coordenadas do vetor binário associado ao ponto escolhido) de uma Bernoulli(p_m).

2. Os genes correspondentes as amostras que resultaram em 1, são escolhidos para sofrer mutação. Por exemplo, se nossos vetores binários correspondentes aos pontos têm tamanho 6, e 100110 é o vetor que estamos trabalhando, se ao gerarmos uma amostra de tamanho 6 de uma Bernoulli (p_m) e obtivermos 001010, concluímos que

o terceiro e quinto genes do vetor vão sofrer mutação. E como é essa mutação? É bem simples, se o valor da terceira posição for 0 ele vai mudar para 1 e se for 1 ele vai mudar para 0. O restante dos genes continuam inalterados. Nesse nosso exemplo, o vetor depois da mutação será 101100.

3. Repita o passo anterior para cada um dos vetores da população.

Voltando ao nosso exemplo que estamos resolvendo, no momento estamos (depois de passar pela seleção e cruzamento) com a população

$$P^{(2)} = \{111110, 110110, 110110, 100110, 110110, 101110\}.$$

Após gerar uma amostra de tamanho 6 de uma Bernoulli(p_m) para cada indivíduo em $P^{(2)}$ analisamos quais entradas dessas amostras são iguais a 1, isso nos dirá que genes serão mutados. Supondo que $p_m = 0.02$ e que o resultado das amostras para cada individuo tenha sido 000000,010000,000000,000000,000000 e 000000. Note que pelas amostras da Bernoulli(p_m), apenas o segundo elemento da população sofrerá mutação e ela se dará no segundo gene que passará de 1 para 0 e a nova população será

$$P^{(3)} = \{111110, 100110, 110110, 100110, 110110, 101110\}.$$

Então , nossa nova população corresponde na nossa discretização a população

$$P^{(3)}_{dual} = \{(3, 19/7), (9/7, 19/7), (18/7, 19/7), (9/7, 19/7), (18/7, 19/7), (15/7, 19/7)\}.$$

Após aplicar o operador de mutação temos de avaliar novamente os valores de imagem dos elementos em $P^{(3)}_{dual}$ pela função f. Na Tabela 4 vemos os valores de imagem de nossa população atual.

Tabela 4 – valor de imagem dos elementos de $P^{(3)}$ pela função f;

Indivíduo	(x, y)	$f(x, y)$
111110	$(3, 19/7)$	16.36735
100110	$(9/7, 19/7)$	9.020408
110110	$(18/7, 19/7)$	13.97959
100110	$(9/7, 19/7)$	9.020408
110110	$(18/7, 19/7)$	13.97959
101110	$(15/7, 19/7)$	11.95918

Então , para encontrar a solução para o problema de otimização que foi exposto repete-se a execução dos passos mostrados de forma iterativa até que o algoritmo genético alcance algum critério de parada (geralmente o número de iterações). Se tivéssemos alcançado os resultados da Tabela 4 na última iteração do algoritmo, então a solução do problema de otimização seria o ponto da última população obtida com maior valor de imagem pela função f, ou seja, o ponto $(3, 19/7)$.

Observação. Note que se antes de começarmos a etapa de mutação a população atual for composta de cópias de um mesmo ponto, o operador mutação (caso $p_m > 0$) pode gerar qualquer população desejada com probabilidade maior que zero. Essa etapa é que faz com que o algoritmo não fique preso em pontos de máximo (mínimo) local.

Após compreendermos cada uma das etapas do algoritmo genético, podemos agora apresentar uma visão geral de tal algoritmo através de um pseudocódigo. Denotando a população no tempo t por $P(t)$ temos a seguinte descrição do algoritmo genético:

Algorithm 1 Algoritmo genético

```
t ← 0
Inicializar(P(t))
Avaliar(P(t))
iter ← número de iterações
while t ≤ iter do
  t ← t + 1
  P(t) ← Seleção(P(t − 1))
  P(t) ← Cruzamento(P(t), p_c)
  P(t) ← Mutação(P(t), p_m)
  Avaliar(P(t))
end while
```

. Os parâmetros do algoritmo genético são determinados de acordo com o tipo de algoritmo que estamos escolhendo. Se estamos lidando com a versão homogênea de tal algoritmo fixamos p_c e p_m antes de iniciar as iterações, mantendo-os inalterados durante toda a execução do algoritmo. No caso não-homogêneo, fixamos p_c e p_m antes de iniciar as iterações e podemos modificar seus valores em cada iteração.

Outra observação a ser feita sobre o pseudocódigo apresentado é quanto ao armazenamento da melhor solução encontrada. No caso elitista, criamos uma posição adicional para armazenar a solução que otimiza a função objetivo usada na etapa de avaliação (guardamos uma cópia do indivíduo que apresenta maior ou menor valor pela função objetivo dependendo se o problema é de maximização ou minimização). Tal armazenamento sempre ocorre após a etapa de avaliação. No caso não-elitista, não ocorre o armazenamento da melhor solução e podemos desconsiderar o que foi dito antes sobre armazenar a melhor solução.

Observação. Uma das características mais importantes do algoritmo genético é que a escolha da codificação do problema é crucial para que o mesmo encontre as soluções buscadas. Escolhas diferentes de codificação podem levar a soluções diferentes para o mesmo problema. Assim, escolher codificações intuitivas, simples e adequadas para o problema é um passo muito importante para conseguir resolver o problema de interesse.

Observação. A escolha da função objetivo é outro ponto importante sobre os algoritmos genéticos. Buscar otimizar tal função, permite lidar com problemas de otimização comple-

xos sem a necessidade de compreender os mecanismos internos que regem esses problemas. Assim, podemos performar uma busca no espaço de estados do problema avaliando os candidatos a solução usando a função objetivo e considerando que as etapas do algoritmo são procedimentos randomizados tal busca não possui vieses implícitos, garantindo que encontraremos soluções ótimas (ao invés de soluções locais). Logo, quanto mais representativa for a função objetivo para as características de interesse de um problema ou de sua dinâmica, maiores serão as probabilidades de encontrar as soluções exatas do problema.

2.4 Classificação dos algoritmos genéticos

No decorrer de seu processo histórico de desenvolvimento os algoritmos genéticos sofreram reformulações e modificações visando torná-los mais eficientes e robustos, de forma a estabelecer garantias para a convergência do algoritmo (ou seja, garantir que o algoritmo encontre pontos de ótimo da função objetivo com base na discretização de seu domínio) para o ponto de ótimo global da função a ser otimizada.

O algoritmo genético que estamos desenvolvendo no exemplo, para encontrar o ponto de máximo da função $f(x, y) = x^2 + y^2$ é o algoritmo que foi desenvolvido por (HOLLAND, 1975) e que foi chamado de algoritmo genético canônico (AGC). Neste tipo de algoritmo os parâmetros p_c e p_m não mudam durante a evolução do algoritmo. Em (RUDOLPH, 1994a) foi mostrado que este algoritmo não converge quase certamente para o conjunto de populações que contenham o ponto de ótimo como um dos seus pontos. Além do mais, neste mesmo artigo (RUDOLPH, 1994a), Rudolph propõe uma modificação neste algoritmo e mostra sua convergência quase certa para o conjunto de populações que contenham o ponto de ótimo como um dos seus pontos. A modificação proposta foi que se criasse uma nova posição no vetor população, a posição $K + 1$ e nela seria guardado o ponto de maior imagem conseguido ao realizarmos o AGC nos K primeiros pontos do vetor população. O valor desse ponto só seria alterado, caso ao final de alguma etapa algum ponto da população atual tivesse imagem maior que a imagem do ponto guardado na posição $K + 1$. Esse algoritmo foi denominado algoritmo genético elitista (AGE).

Em (GREFENSTTETE, 1986) uma série de simulações foram apresentadas no sentido de ilustrar que a variação dos parâmetros p_c e p_m interferem na saída do algoritmo genético. O algoritmo genético não-homogêneo (AGNH) foi apresentado em (CAMPOS V.S.M.; CRUZ, 2012) como uma tentativa de melhorar a eficiência do AGC, uma vez que era permitido aos parâmetros p_c e p_m variarem dentro de certas condições. A versão não-homogênea do AGE, chamada de algorimo genético não-homogêneo elistista (AGNHE), foi apresentado por (CRUZ; PEREIRA, 2012), com objetivo de melhorar o desempenho do AGE. Outras tentativas de melhorar o desempenho do AGC sem que houvesse variação em p_c e p_m, podem ser lidos em (DOREA C.C.Y.; PEREIRA, 2010), comparações numéricas

entre o AGNHE e AGE podem ser encontradas em (CAMPOS V.S.M.; ASSIS, 2012) e maneiras apropriadas de implementar o AGNHE pode ser encontrado em (ANDRADE; PEREIRA, 2015).

Muitas outras modificações foram implementadas a partir dos AGC, mas uma coisa que a maioria dessas versões possuem em comum é que sua análise de convergência se dá via a teoria de Cadeias de Markov. Porque o AGC, AGE, etc, são Cadeias de Markov conforme veremos. Como só estudamos as Cadeias de Markov homogêneas, utilizaremos apenas as versões AGC e AGE que mostraremos serem Cadeias de Markov homogêneas.

Uma vez que AGC é uma Cadeia de Markov, a análise de sua convergência se dará segundo a teoria de Cadeias de Markov abordada nesse capítulo. Contudo, além de fazer a análise de convergência (encontrar a distribuição assintótica) queremos saber qual a probabilidade, quando no limite, da cadeia estar numa população em que um de seus pontos seja o ponto de ótimo desejado. Em (CRUZ, 1998) é mostrado que o AGC é uma Cadeia de Markov e que a saída da etapa de seleção é a entrada da etapa de cruzamento, assim como a saída da etapa de cruzamento é a entrada do processo de mutação e a saída do processo de mutação é o resultado da primeira iteração e a entrada do passo zero é a entrada do processo de seleção. Logo, não apenas o valor do próximo passo é completamente determinado pelo passo presente (caracterização de Cadeia de Markov) como ainda sabemos a forma da matriz de transição (produto das matrizes de cada etapa do processo).

Observação. Como pode ser visto em (CRUZ, 1998) e em (SOBRINHO, 2014), a etapa de mutação pode transformar a população anterior a essa etapa em qualquer outra população com probabilidade maior que zero. Isso significa que a matriz de transição da etapa de mutação é uma matriz com todas as entradas positivas. Isso faz com que o produto das matrizes ABC (A,B e C são as matrizes de transição das etapas mutação, cruzamento e seleção respectivamente) seja uma matriz com todas as entradas positivas. Isso implica que a matriz de transição do AGC possui todas as entradas positivas, portanto o AGC é uma cadeia irredutível, aperiódica e finita, logo, convergente conforme pode foi provado em (SOBRINHO, 2014), implicando que o AGC é convergente.

Como falamos anteriormente, gostaríamos de saber se no equilíbrio (no limite) qual a probabilidade da cadeia estar num conjunto formado por populações em que um dos pontos de qualquer dessas populações seja o ponto de ótimo procurado. Para isso precisamos definir algumas ferramentas que nos ajudarão no cálculo dessa probabilidade.

Definição 2.4.1. Seja $Z_n = \max\{f(b) \mid b \in P^{(n)}\}$ uma sequência de variáveis aleatórias representando o maior valor de f assumido por algum elemento da população $P^{(n)}$. Dizemos que um algoritmo genético converge para um ponto de máximo global quando

$$\lim_{n \to +\infty} P(Z_n = f^*) = 1,$$

em que $f^* = \max\{f(b) \mid b \in S\}$ é o ponto de máximo global do problema.

Observação. No caso de um problema de minimização, trocamos as definições de Z_n e f^*. A definição de Z_n passa a ser

$$Z_n = \min\{f(b) \mid b \in P^{(n)}\}$$

e a definição de f^* passa a ser

$$f^* = \min\{f(b) \mid b \in S\}.$$

O teorema a seguir mostra que o AGC não converge quase certamente para o ponto de ótimo global procurado.

Teorema 2.4.1. O algoritmo genético canônico representado por uma cadeia de Markov homogênea, finita, aperiódica e irredutível não converge para um ótimo global.

A demonstração de tal teorema pode ser encontrada em (SOBRINHO, 2014) com todos os detalhes pertinentes e complementada por resultados em (LAWLER, 2018). Logo, o algoritmo genético não converge para um ponto de ótimo global de acordo com o teorema 2.4.1.

No caso do algoritmo genético elitista, vemos que a única diferença (e essa diferença é fundamental) é que uma nova posição $(K+1)$ é criada no vetor população (nessa posição armazenamos o ponto com maior ou menor valor dependendo se o problema é de maximização ou minimização). A população inicial é formada pelas K primeiras posições sendo formada por pontos da malha que foram sorteados aleatoriamente e na posição $K+1$ uma cópia do ponto de maior imagem (caso haja mais de um, sorteia-se aleatoriamente um dentre eles). Em seguida as etapas do AGC são aplicadas para as K primeiras entradas e ao final da etapa de mutação, verifica-se se algum das K primeiras entradas tem imagem maior que a imagem do ponto que se encontra na posição $K+1$. Caso positivo, o ponto de maior imagem tem uma cópia colocada na posição $K+1$ e em caso negativo, nada se altera. Depois o algoritmo volta a repetir esse procedimento até que um critério de parada seja atingido.

Novamente, percebemos que na dinâmica do AGE a próxima etapa só depende da etapa anterior o que o caracteriza como uma Cadeia de Markov. O espaço de estados agora é

$$\tilde{E} = \{(x_1, x_2, ..., x_K, x_{K+1})/x_i \in S, i = 1, 2, ..., K+1 \text{ e } f(x_{K+1}) \geq f(x_i), i = 1, 2, ..., K\}.$$

Desta vez, essa cadeia não é irredutível mas possui um conjunto fechado tal que a cadeia restrita a esse conjunto é irredutível, aperiódica e finita, logo, convergente. O problema é saber se a cadeia chega nesse conjunto quase certamente em tempo finito. Em (RUDOLPH, 1994b) é mostrado que a cadeia realmente atinge esse conjunto fechado em tempo finito quase certamente.

Teorema 2.4.2. No algoritmo genético elitista temos que

$$P(\lim_{n\to+\infty} \mid Z_n - f^* \mid = 0) = 1,$$

em que $f^* = \max\{f(x) \mid x \in S\}$ e f é a função a ser otimizada.

Observação. No caso de um problema de minimização, trocamos as definições de Z_n e f^*. A definição de Z_n é

$$Z_n = \min\{f(b) \mid b \in P^{(n)}\}$$

e a definição de f^* é

$$f^* = \min\{f(b) \mid b \in S\}.$$

A demonstração do teorema acima pode ser encontrada com todos os detalhes pertinentes em (RUDOLPH, 1994b).

Assim, o algoritmo genético elitista converge quase certamente para o conjunto das populações que contém o ponto ótimo do problema de otimização considerado como um dos seus pontos.

Observação. O ponto de ótimo buscado (no caso do AGE) sempre será o indivíduo na última posição dentro da população (supondo que já estamos no conjunto das populações contendo os pontos de ótimo). A existência desse conjunto é garantida e o AGE entra nele em tempo finito com probabilidade 1, não conseguindo sair dele pois o mesmo é fechado.

3 Seleção de variáveis

3.1 Introdução

Em muitos problemas no contexto da análise de regressão linear buscamos simplificar os modelos lineares em relação a quantidade de variáveis a serem incluídas para que o modelo ajuste-se adequadamente aos dados. Tal necessidade pode justificar-se pela existência de redundâncias e irrelevâncias nos dados que podem acarretar em: redução no nível de interpretabilidade do modelo, maior quantidade de recursos computacionais para determinar os parâmetros do modelo e até mesmo reduzir a percepção de padrões presentes nos dados.

Visando contornar os problemas citados acima, muitas técnicas foram desenvolvidas para resolver o problema de seleção de variáveis. Dentre as técnicas existentes podemos citar: RIDGE, análise de componentes principais, LASSO, seleção progressiva e regressiva de variáveis e seleção do melhor subconjunto de variáveis (RISH; GRABARNIK, 2014) (JAMES et al., 2013) . Neste capítulo, será apresentado uma outra técnica não muito conhecida para performar seleção de variáveis: o algoritmo genético elitista (GOLDBERG, 1989).

3.2 Seleção de variáveis

Seja $S = \{X_1, \ldots, X_d\}$ um conjunto de variáveis reais independentes, uma variável real Y e um modelo de regressão linear múltipla

$$Y = f(\mathbf{X}, \alpha) + \mathbf{e}$$

em que $\mathbf{X} = (X_1, \ldots, X_d)$ é um vetor de variáveis, $\alpha = (\alpha_1, \cdots, \alpha_d)$ é o vetor de parâmetros do modelo (os coeficientes de regressão) e $\mathbf{e}$ é a variável aleatória representando o erro do modelo Y. O problema de seleção de variáveis consiste em determinar um subconjunto $V \subset S$ de variáveis a serem incluídas no modelo. Equivalentemente, o problema de seleção de variáveis consiste em determinar quais são as componentes não-nulas do vetor de parâmetros α.

Tal problema possui diversas abordagens de solução literariamente consolidadas. Dentre elas temos a seleção do melhor subconjunto de variáveis: tal método consiste em construir todos os modelos possíveis para a variável resposta Y com as d variáveis de $\mathbf{X} = (X_1, \ldots, X_d)$ e analisar qual modelo possui o menor nível de erro (quando consideramos

algum critério para mensurar os erros cometidos nos dados)

$$\min_{f} \ erro(f).$$

Observação. $erro(f)$ é o tipo de função erro selecionada pelo usuário para mensurar numericamente o erro cometido pelo modelo f nos dados disponíveis.

Observação. No contexto da análise de regressão linear múltipla, frequentemente consideramos a função f como

$$f(\mathbf{X}, \alpha) = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 + \cdots + \alpha_d X_d.$$

Como as d variáveis em $\mathbf{X}$ permitem construir no máximo 2^d modelos com configurações distintas de variáveis, então temos que realizar os seguintes procedimentos gerais para selecionar qual dos 2^d modelos melhor se ajusta aos dados determinados por $\mathbf{X} = (X_1, \ldots, X_d)$:

1. Começamos com um modelo sem nenhuma variável $f_0(x) = \alpha_0$ definido como um modelo constante e igual ao intercepto α_0. Tal modelo está em um conjunto denotado por M_0.

2. Para cada $k \in \{1, \ldots, d\}$ denotamos por M_k o conjunto formado por todos os modelos que contêm exatamente k variáveis. Pela identidade combinatorial

   $$\sum_{j=0}^{d} \binom{d}{j} = 2^d,$$

   a união dos conjuntos M_k gera todos os modelos distintos possíveis com no máximo d variáveis. Em seguida, escolhemos em cada conjunto M_k *para* $k \in \{1, \ldots, d\}$ um modelo f_k tal que

   $$erro(f_k) = \min_{f \in M_k} erro(f)$$

   gerando ao final um conjunto $M = \{f_1, \ldots, f_d\}$.

3. A partir do conjunto M escolhemos o modelo com o melhor subconjunto de variáveis de $\{X_1, \ldots, X_d\}$ como sendo o modelo f_{k_0} de índice k_0 tal que

   $$erro(f_{k_0}) = \min_{f \in M} erro(f).$$

   Temos ainda outros métodos de seleção de variáveis que não serão discutidos no âmbito deste trabalho.

Quando consideramos o modelo linear

$$Y = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 + \cdots + \alpha_d X_d + \mathbf{e}$$

e um conjunto de observações $(Y, X_1, \ldots, X_d)$ podemos reescrever o modelo no formato de produto matricial da seguinte forma

$$Y = X\alpha + \mathbf{e},$$

em que $X \in M_{m\times d}(\mathbb{R})$ é a matriz reunindo as observações das variáveis do modelo e $\mathbf{e}$ é o vetor aleatório associado ao modelo em questão.

Determinar o modelo linear acima é equivalente a estimar os parâmetros em α e o erro do modelo $\mathbf{e}$. Usando o método de quadrados mínimos, obtemos tais estimativas dos parâmetros minimizando a função objetivo

$$f(\alpha) = (y - X\alpha)^t(y - X\alpha).$$

Em (RAO; TOUTENBURG, 1995) vemos que encontrar uma solução para tal problema é equivalente a encontrar uma solução para o sistema linear

$$X^tX\alpha = X^ty,$$

cuja solução geral é dada por

$$\alpha = (X^tX)^-X^ty + (I - (X^tX)^-X^tX)v,$$

em que $v \in \mathbb{R}^d$ é arbitrário.

Definição 3.2.1. Seja $A \in M_{m\times n}(\mathbb{R})$ uma matriz de posto qualquer. O espaço coluna de A é o espaço vetorial definido pelo conjunto

$$col(A) = \{Ax \in \mathbb{R}^m \mid x \in \mathbb{R}^n\}.$$

Porém, a solução do sistema acima existe independentemente da matriz de observações X ser inversível e é chamada de inversa generalizada cuja definição segue:

Definição 3.2.2. Seja $A \in M_{m\times n}(\mathbb{R})$ uma matriz de posto qualquer. Uma inversa generalizada de A é uma matriz $G \in M_{n\times m}(\mathbb{R})$ tal que $X = Gy$ é uma solução da equação $AX = y$ para todo $y \in col(A)$.

Observação. No caso particular da matriz X ser inversível, temos que

$$\alpha = (X^tX)^{-1}X^ty,$$

é a solução do sistema linear acima.

Os teoremas seguintes nos fornecem métodos para computar os coeficientes de regressão do modelo linear descrito anteriormente. Com base neles podemos fazer o ajuste dos modelos após selecionar as variáveis a serem incluídas no mesmo.

Teorema 3.2.1. A função objetivo $f(\alpha) = (y - X\alpha)^t(y - X\alpha)$ alcança valor mínimo em qualquer solução de $X^tX\alpha = X^ty$. Além disso, a estimativa de y é dada por $\widehat{y} = X\alpha$ em que α é qualquer solução de $X^tX\alpha = X^ty$.

Teorema 3.2.2. O estimador $\widehat{\alpha}$ de $\alpha = (\alpha_1, \ldots, \alpha_d)$ é uma solução do problema

$$\min_{\theta \in \mathbb{R}^n} \|y - X\theta\|_2$$

em que $\widehat{\alpha}$ é ortogonal a $col(X)$, ou seja, $\widehat{\alpha}$ é a projeção ortogonal de y sobre $col(X)$. O estimador $\widehat{\alpha}$ existe e é unico com expressão dada por

$$\widehat{\alpha} = X(X^tX)^-X^ty.$$

O lema seguinte nos fornece uma condição necessária e suficiente para que um estimador seja não-viesado com variância mínima.

Lema 3.2.3. Seja T uma estatística tal que $\mathbb{E}(T) = \theta$, e $Var(T) < \infty$. Então, T é um estimador não-viesado de variância mínima para θ se e somente se

$$cov(T, t) = 0 \ \ \forall t \text{ tal que } \mathbb{E}(t) = 0 \text{ e} \quad Var(t) < \infty.$$

Teorema 3.2.4. O estimador linear não-viesado de dispersão mínima para $\alpha = (\alpha_1, \ldots, \alpha_d)$ é

$$\widehat{\alpha} = (X^tX)^{-1}X^ty.$$

A matriz de dispersão mínima é

$$\sigma^2(X^tX)^{-1}.$$

Visto que temos estimado o vetor de parâmetros α, podemos agora estimar o erro do modelo. Tal erro pode ser estimado como

$$\widehat{\mathbf{e}} = y - X\widehat{\alpha},$$

em que $\widehat{\alpha}$ foi obtido no Teorema 3.2.4.

Teorema 3.2.5. O estimador do erro $\widehat{\mathbf{e}} = y - X\widehat{\alpha}$ do modelo é o estimador linear não-viesado de dispersão mínima de $\mathbf{e}$.

Com base no estimador do erro obtido no Teorema 3.2.5 podemos obter um estimador não-viesado para a variância do modelo linear dado por

$$s^2 = \frac{1}{n - Posto(X)} y^t\widehat{\mathbf{e}}.$$

Até o momento não fizemos qualquer suposição sobre a distribuição do vetor erro $\mathbf{e}$ associado ao modelo linear $y = X\alpha + \mathbf{e}$, implicando que os resultados apresentados até aqui são gerais em sua essência.

A partir deste ponto, vamos supor que o vetor $\mathbf{e}$ possui distribuição normal multivariada com média $\mathbf{0}$ e variância σI e observemos as consequências imediatas. Dada a distribuição do erro $\mathbf{e}$, o modelo para y têm distribuição $N(X\alpha, \sigma^2 I)$, o que nos permite concluir o seguinte teorema.

Teorema 3.2.6. O estimador de máxima verossimilhança dos parâmetros do modelo linear e o estimador obtido por quadrados mínimos são iguais. Além disso, o estimador de máxima verossimilhança $\widehat{\sigma}^2$ de σ^2 é assintoticamente não-viesado.

Observação. Nesse caso, o estimador de máxima verossimilhança coincide com o estimador obtido por quadrados mínimos porque estamos supondo a normalidade da variável aleatória determinando o erro.

Sabendo que o estimador obtido por quadrados mínimos é $\widehat{\alpha} = (X^tX)^{-1}X^ty$ e que $Var(\widehat{\alpha}) = \sigma^2(X^tX)^{-1}$ podemos concluir supondo que existe o limite $\lim_{n\to\infty} \frac{X^tX}{n}$ que $\widehat{\alpha}$ converge em média quadrática para α. Dessa forma, prova-se que $\widehat{\alpha}$ é um estimador consistente para α porque convergência em média quadrática implica convergência em probabilidade.

Observação. A prova dos teoremas acima encontram-se na referência (RAO; TOUTENBURG, 1995).

3.3 O critério da informação de Akaike

O critério da informação de Akaike, popularmente conhecido como AIC(Akaike information criterion) é um estimador de uma medida numérica conhecida como informação de Kullback-Leibler (ANDERSON; BURNHAM; ANDERSON, 1998). Dadas duas funções f e g, a informação de Kullback-Leibler é definida por

$$I(f,g) = \int_{\Omega} f(x) \ln\left(\frac{f(x)}{g(x,\theta)}\right) dx.$$

A importância de tal medida está no fato que dado um conjunto de dados gerado por um modelo f, a informação de Kullback-Leibler entre f e g determina a informação perdida ao usar o modelo g para ajustar os dados gerados por f. Conhecendo a definição da informação de Kullback-Leibler podemos derivar o AIC como um estimador de tal medida, cuja expressão é

$$AIC(g) = -2\ln(L(\widehat{\theta})) + 2K,$$

em que $\widehat{\theta}$ é o estimador de máxima verossimilhança dos parâmetros do modelo g e K é o número de parâmetros do modelo g. O AIC possui ampla aplicação para solucionar o problema de seleção de modelos, ou seja, podemos buscar um modelo em um conjunto finito de modelos $M = \{g_1, \ldots, g_m\}$ que melhor se ajusta aos dados gerados por um modelo

desconhecido f. A seleção de modelos ocorre buscando qual modelo em M possui menor valor de AIC e consequentemente melhor aproxima o modelo desconhecido f.

Especificamente quando tratando de modelos de regressão linear com erro distribuído normalmente, o AIC pode ser escrito como

$$AIC_{LM}(g) = n \ln\left(\frac{SQR(g)}{n}\right) + 2K,$$

em que $SQR(g)$ é a soma dos resíduos quadráticos do modelo g usado para aproximar o modelo desconhecido f, K o número de parâmetros do modelo e n o número de observações disponíveis nos dados. Quando consideramos a estimação dos parâmetros por meio do método dos quadrados mínimos, o número de parâmetros estimados já inclui na contagem o intercepto e a variância do modelo, o que nos leva a observar o AIC como

$$AIC_{LM}(g) = n \ln\left(\widehat{\sigma}^2\right) + 2K,$$

em que

$$\widehat{\sigma}^2 = \frac{SQR(g)}{n}.$$

Devido a estudos posteriores e limitações observadas na expressão inicial do AIC, estimulou-se novas buscas por estimadores da informação de Kullback-Leibler adequados para diferentes contextos dos dados. Nos trabalhos (HURVICH; TSAI, 1991) e (SUGIURA, 1978), perceberam que o AIC pode gerar estimativas imprecisas da informação de Kullback-Leibler se o número de parâmetros é grande em comparação a quantidade de dados disponíveis. Assim, derivaram um AIC ajustado a pequenas amostras chamado AIC_c cuja expressão é

$$AIC_c(g) = AIC(g) + \left(\frac{2K(K+1)}{n-K-1}\right),$$

onde n é o tamanho da amostra disponível e K o número de parâmetros do modelo g.

Apesar de existirem variações do AIC, todas elas possuem termos de penalidade dependendo de K (o número de parâmetros do modelo). A existência de tais penalidades implica que quanto maior o número de parâmetros K do modelo, maior o valor de AIC associado ao modelo. Como o objetivo da seleção de modelos é encontrar um modelo com o menor valor de AIC em M, podemos simultaneamente realizar a seleção de variáveis a serem incluídas no modelo visto que dados dois modelos em M com o mesmo valor de AIC o princípio da parsimônia (variante do princípio da navalha de Occam) declara que o modelo com a menor quantidade de parâmetros é mais adequado para aproximar o modelo desconhecido f. Dessa forma, obter um modelo com menos parâmetros não-nulos é equivalente a obter um modelo com menos variáveis (e suas tranformações).

3.4 Algoritmos genéticos e seleção de variáveis

Os trabalhos (ACOSTA-GONZÁLEZ; FERNÁNDEZ-RODRÍGUEZ, 2007), (PATERLINI; MINERVA, 2010) e (LACERDA; CARVALHO; LUDERMIR, 2002) mostram como os algoritmos genéticos podem ser utilizados para seleção de modelos no contexto da construção de modelos de regressão. Além disso, exibem estudos experimentais que corroboram os benefícios do uso do algoritmo genético para o problema de seleção de variáveis. Porém, tais trabalhos não justificam do ponto de vista teórico porque os algoritmos genéticos são uma classe com propriedades adequadas para a solução deste tipo de problema, nem mesmo especificam em detalhes a estrutura dos algoritmos genéticos que estão utilizando nas análises experimentais. No trabalho (LACERDA; CARVALHO; LUDERMIR, 2002) os autores fazem uso de dois conjuntos de dados encontrados respectivamente em (R.W., 1996) e (PURDIE LUCAS E.A., 1992). O primeiro conjunto de dados é o conjunto *'Body fat measurement'* constituído por 252 observações e 16 variáveis independentes: *idade, peso, altura, densidade, peso corporal líquido* , dez outras medidas de circunferência corporal de um indivíduo e a variável resposta é a porcentagem de gordura corporal. O segundo conjunto de dados é o conjunto *'Cholesterol measurement'* formado por 264 observações e 21 variáveis independentes, em que 20 destas variáveis descrevem medidas de absorção ótica de amostras de sangue em diferentes frequências e a variável resposta é o nível de colesterol no sangue.

Motivado por estas observações fizemos no Capítulo 2 a fundamentação teórica ausente nos trabalhos mencionados e apresentamos a partir de agora como o algoritmo genético elitista (homogêneo) pode ser usado para a seleção de variáveis visto que demonstramos no capítulo 2 que tal algoritmo converge quase certamente para o conjunto das populações que contêm o ponto de ótimo do problema de otimização envolvendo a função objetivo. O uso dos algoritmos genéticos é justificado por sua alta flexibilidade para solução de problemas de otimização, pois permitem uma busca simultânea de soluções através de uma população de candidatos a solução ignorando diversas limitações necessárias em outras técnicas de otimização (como diferenciação ou convexidade das funções envolvidas).

A partir de agora vamos mostrar como faremos uso dos algoritmos genéticos para buscar soluções para o problema de seleção de variáveis. De acordo com a exposição sobre os algoritmos genéticos feita no Capítulo 2, o uso de uma função objetivo a ser otimizada é necessária para utilizar algoritmos genéticos. Para selecionar as variáveis e consequentemente modelos faremos uso do critério de seleção de modelos AIC (que como vimos é uma medida de discrepância entre modelos e pode ser usada para seleção de variáveis) que será usado como função objetivo no algoritmo genético. Para ilustrar como esses procedimentos ocorrem, suponha que temos um conjunto de dados X com seis variáveis $V = \{X_1, X_2, X_3, X_4, X_5, X_6\}$ e desejamos fazer seleção de variáveis em V utilizando o algoritmo genético. Para isso, vamos associar o número 0 a uma variável

quando a mesma não está incluída no modelo e o número 1 quando a variável está incluída no modelo. No processo de inicialização do algoritmo genético, geramos aleatoriamente (de acordo com os procedimentos descritos no Capítulo 2) diversas configurações das variáveis que definem os modelos. Vamos supor que obtivemos os modelos definidos pelas seguintes configurações de variáveis

$$M = \{110010, 010111, 111100, 001010, 000111\}.$$

Com base nessa população, avaliamos os modelos em M usando o AIC para determinar o nível de adequação dos mesmos aos dados em X e mantemos uma cópia do modelo em M que apresenta menor valor de AIC (de forma que nas iterações posteriores possamos comparar as melhores configurações de variáveis encontradas e sempre manter uma cópia da configuração com menor valor de AIC). Após isso, aplicamos os operadores genéticos de seleção, cruzamento e mutação conforme explicado no Capítulo 2. Com base na população inicial usamos o AIC como função objetivo do algoritmo genético da seguinte forma: em cada um dos modelos em M avaliamos o valor de AIC do modelo (da mesma forma que medimos a aptidão de um indivíduo em uma população quando descrevemos o algoritmo genético no Capítulo 2) e aplicaremos os operadores de seleção (com o intuito de manter as configurações de variáveis com menor valor de AIC), cruzamento (visando evitar vieses na população obtida por seleção e diversificar as configurações de variáveis para a próxima etapa) e mutação (para evitar que o algoritmo convirja precipitadamente para uma configuração de variáveis cujo modelo linear associado tenha valor de AIC como um mínimo local quando consideramos a relação de proximidade entre duas configurações como o número de componentes diferentes entre duas sequências binárias) com o objetivo de obter uma configuração de variáveis que apresente menor valor de AIC em relação a todas as outras configurações na população atual, ou seja, vamos usar o algoritmo genético para obter um modelo linear com uma configuração de variáveis alcançando o menor valor possível de AIC. A solução que vamos considerar ao final será aquele modelo obtido após a aplicação do algoritmo genético elitista que possui o conjunto de variáveis que apresente o menor valor de AIC.

Observação. De modo geral, ao usar o algoritmo genético no problema de seleção de variáveis criamos um conjunto aleatório de candidatos a solução. Tal conjunto sofre continuamente transformações randomizadas representadas pelos operadores genéticos de seleção, cruzamento e mutação (visando eliminar vieses na busca pela configuração adequada de variáveis). As sucessivas populações obtidas iterativamente, apresentam soluções cada vez melhores do problema de seleção (no sentido de apresentarem configurações de variáveis cujos modelos lineares ajustados a essas variáveis possuem valores de AIC cada vez menores, implicando em modelos com menos variáveis e melhores qualidades preditivas).

Observação. O uso da codificação binária no algoritmo genético e da escolha do AIC como

função objetivo foi deliberado e motivado pelo fato de que tais escolhas nos permitem performar uma busca dirigida pelos modelos com menos variáveis e melhores propriedades preditivas. Essa codificação específica permite representar o espaço de busca dos modelos lineares ajustados aos dados e determinar a exata configuração de variáveis do melhor modelo para os dados.

4 Aplicação do algoritmo genético elitista em um problema de seleção de variáveis

Conforme foi exposto no capítulo anterior com o exemplo do conjunto de dados *Credit*, existem conjuntos de dados tais que a aplicação de técnicas para o problema de seleção de variáveis baseadas em métodos de seleção gradativa de variáveis pode levar a soluções distintas. Veremos neste capítulo como o algoritmo genético elitista nos permite superar esses problemas através de um método consistente, robusto e estável computacionalmente que nos permite encontrar uma solução para o problema de seleção de variáveis no conjunto de dados *Credit.*

Para contextualizar como faremos uso do algoritmo genético elitista para selecionar variáveis em *Credit* e simplificar o nível de complexidade inerente a compreensão das etapas do mesmo, vamos fixar os parâmetros e hiperparâmetros deste algoritmo. Especificamente, vamos executar uma iteração do AGE para solucionar o problema de seleção de variáveis em *Credit* com uma população inicial de tamanho $n = 8$, probabilidade de cruzamento $p_c = 0,6$ e probabilidade de mutação $p_m = 0,1$.

Detalhamos agora como representaremos o problema para usar o AGE . O espaço de estados do problema é o conjunto

$$S = \{0,1\}^{11},$$

formado por todas as sequências binárias de comprimento 11. Cada sequência em S representa uma configuração das variáveis selecionadas e não-selecionadas para um modelo de regressão linear (no caso do conjunto de dados *Credit* o bit 0 representa que a variável não deve ser escolhida e 1 que a variável deve ser escolhida). A função objetivo a ser utilizada é o AIC (Akaike information criterion) discutido no capítulo anterior e será aplicada sobre os modelos de regressão linear definidos com as configurações de variáveis determinada pelos elementos da população atual que o AGE está lidando iterativamente.

O conjunto de dados *Credit* que será utilizado para a seleção de variáveis pode ser encontrado na bibilioteca *ISLR* e analisado após o carregamento de tal biblioteca como segue:

```
1 head(data)
2  ID  Income Limit Rating Cards Age Education Gender Student Married
3
4  1   1  14.891  3606    283     2  34        11       1        1       2
               3
5 2   2 106.025  6645    483     3  82        15       2        2       2
             2
```

```
6 3   3 104.593   7075    514      4  71         11       1        1        1
              2
7 4   4 148.924   9504    681      3  36         11       2        1        1
              2
8 5   5  55.882   4897    357      2  68         16       1        1        2
              3
9 6   6  80.180   8047    569      4  77         10       1        1        1
10
11     Ethnicity   Balance
12
13 1      3           333
14 2      2           903
15 3      2           580
16 4      2           964
17 5      3           331
18 6      3           1151
```

4.1 – Carregando dados presentes em *Credit*

buscamos encontrar qual a melhor combinação de variáveis em *Credit* de forma a construir modelos de regressão linear para prever a variável *Balance*. As variáveis que consideramos para essa busca são as variáveis de nomes:

```
1 colnames(data[,-12])
2 [1] "ID"        "Income"    "Limit"     "Rating"    "Cards"     "Age"
3 [2] "Education" "Gender"    "Student"   "Married"   "Ethnicity"
```

4.2 – Nomes das variáveis em *Credit*

Definidos os detalhes técnicos podemos agora iniciar a simulação do AGE para solucionar o problema de seleção de variáveis. Podemos gerar uma população inicial com 8 indivíduos contendo 11 bits cada (onde 11 representa a quantidade de variáveis em *Credit* a serem usadas para seleção) com os parâmetros declarados anteriormente:

```
1 [[1]]
2  [1] 0 1 1 1 1 0 0 0 1 0 1     AIC: 4821.188
3
4 [[2]]
5  [1] 1 1 0 1 1 1 1 1 0 0 1     AIC: 5222.036
6
7 [[3]]
8  [1] 1 0 1 0 1 1 1 1 0 1 0     AIC: 5488.230
9
10 [[4]]
11  [1] 1 1 0 0 1 1 0 1 1 0 1     AIC: 5917.383
12
13 [[5]]
14  [1] 0 1 0 0 1 1 1 1 1 0 0     AIC: 5915.468
15
```

```
[[6]]
 [1] 1 0 0 1 1 0 0 1 1 1 1     AIC: 5378.100

[[7]]
 [1] 0 1 0 1 0 0 1 1 1 1 0     AIC: 4854.932

[[8]]
 [1] 1 1 1 0 0 1 1 0 1 0 0     AIC: 4860.889
```

4.3 – Simulação da população inicial e respectivos valores de AIC

Tendo como base essa população inicial armazenamos o indivíduo com menor valor de AIC:

```
[[1]]
 [1] 0 1 1 1 1 0 0 0 1 0 1     AIC: 4821.188
```

4.4 – Indivíduo com menor valor de AIC

.

Aplicamos agora a primeira etapa do AGE : a seleção . Para isso, devemos selecionar os indivíduos da população inicial com maior probabilidade de ter AIC baixo quando consideramos os modelos de regressão linear definidos pelas configurações de variáveis presentes na população inicial. Com base, no algoritmo do Apêndice obtemos a seguinte nova população:

```
fsel(popsel,dadosrecorte)

[[1]]
 [1] 1 1 1 0 0 1 1 0 1 0 0   AIC:  4821.188

[[2]]
 [1] 1 1 0 1 1 1 1 1 0 0 1   AIC:  5222.036

[[3]]
 [1] 0 1 1 1 1 0 0 0 1 0 1   AIC:  5488.230

[[4]]
 [1] 1 0 1 0 1 1 1 1 0 1 0   AIC:  5917.383

[[5]]
 [1] 1 1 1 0 0 1 1 0 1 0 0   AIC:  5915.468

[[6]]
 [1] 0 1 0 1 0 0 1 1 1 1 0   AIC:  5378.100

[[7]]
 [1] 1 1 1 0 0 1 1 0 1 0 0   AIC:  4854.932
```

```
23
24 [[8]]
25  [1]  1  0  0  1  1  0  0  1  1  1  1   AIC:  4860.889
```

4.5 – Simulação da etapa de seleção

Esta população é formada pelos indivíduos da população inicial com maior probabilidade dos modelos de regressão linear ajustados com as variáveis determinadas por eles terem baixo valor de AIC.

A próxima etapa do AGE é realizar o processo de cruzamento. Com a probabilidade de cruzamento estando fixada como $p_c = 0,6$ vamos obter um subconjunto da população obtida no processo de seleção, onde cada elemento da população têm probabilidade $0,6$ de pertencer ao subconjunto. Por meio do AGE podemos gerar o seguinte subconjunto:

```
1 [[6]]
2  [1]  0  1  0  1  0  0  1  1  1  1  0
3
4 [[7]]
5  [1]  1  1  1  0  0  1  1  0  1  0  0
```

4.6 – Simulação obtendo os indivíduos para sofrerem cruzamento

Com base nesse subconjunto o processo de cruzamento ocorrerá aos pares e o elemento que não formar um par retorna inalterado para a população (no nosso caso, não há retorno de nenhum indivíduo). Os pares formados sofrerão cruzamento a partir da troca de bits definida pelo ponto de quebra 3 obtido aleatoriamente (conforme exposto no Capítulo 1). Após isso, os pares modificados de indivíduos voltam a fazer parte da população. Como resultado final do cruzamento temos a nova população :

```
1 fpc(pc,popsel, popaux)
2
3 [[1]]
4  [1]  1  1  0  1  1  1  1  1  0  0  1   AIC: 5222.036
5
6 [[2]]
7  [1]  0  1  1  1  1  0  0  0  1  0  1   AIC: 4821.188
8
9 [[3]]
10  [1]  1  0  1  0  1  1  1  1  0  1  0   AIC: 5488.230
11
12 [[4]]
13  [1]  1  0  0  1  1  0  0  1  1  1  1   AIC: 5378.100
14
15 [[5]]
16  [1]  1  1  1  0  0  1  1  0  1  0  0   AIC: 4860.889
17
18 [[6]]
```

```
 [1] 1 1 1 0 0 1 1 0 1 0 0   AIC: 4860.889

[[7]]
 [1] 0 1 0 0 0 1 1 0 1 0 0   AIC: 5917.726

[[8]]
 [1] 1 1 1 1 0 0 1 1 1 1 0   AIC: 4839.548
```

4.7 – Nova população após o processo de cruzamento

Assim, no processo de cruzamento trocamos aleatoriamente sequências de bits (que representam a pertinência de uma variável ao modelo) entre configurações de variáveis na população na tentativa de obter novas configurações com valores menores de AIC do modelo linear que inclui tais variáveis.

Após a execução do processo de cruzamento na população passamos a última etapa da primeira iteração considerando que a probabilidade de mutação está fixada em $p_m = 0,1$ em cada indivíduo da população obtida por cruzamento, modificaremos os bits de cada indivíduo com probabilidade p_m. Assim, dado um indivíduo da população modificaremos o bit 0 para 1 e vice-versa com probabilidade p_m. Obtemos a seguinte nova população:

```
fpm(pm,popsel)

[[1]]
 [1] 0 0 1 0 0 0 0 0 1 1 0

[[2]]
 [1] 1 0 0 0 0 1 1 1 0 1 0

[[3]]
 [1] 0 1 0 1 0 0 0 0 1 0 1

[[4]]
 [1] 0 1 1 0 0 1 1 0 0 0 0

[[5]]
 [1] 0 0 0 1 1 0 0 1 0 1 1

[[6]]
 [1] 0 0 0 1 1 0 0 1 0 1 1

[[7]]
 [1] 1 0 1 1 1 0 0 1 0 1 1

[[8]]
 [1] 0 0 0 0 1 1 0 0 0 0 1
```

4.8 – Nova população após o processo de mutação

Ao final do processo de mutação, computamos o AIC do modelo de regressão linear ajustado as variáveis em cada indivíduo da população obtida por mutação. O resultado desta computação pode ser visto a seguir:

```
calcmin(popsel,dadosrecorte)

[[1]]
 [1] 0 0 1 0 0 0 0 0 1 1 0   AIC: 5379.169

[[2]]
 [1] 1 0 0 0 0 1 1 1 0 1 0   AIC: 6052.458

[[3]]
 [1] 0 1 0 1 0 0 0 0 1 0 1   AIC: 4853.050

[[4]]
 [1] 0 1 1 0 0 1 1 0 0 0 0   AIC: 5228.321

[[5]]
 [1] 0 0 0 1 1 0 0 1 0 1 1   AIC: 5499.465

[[6]]
 [1] 0 0 0 1 1 0 0 1 0 1 1   AIC: 5499.465

[[7]]
 [1] 1 0 1 1 1 0 0 1 0 1 1   AIC: 5501.367

[[8]]
 [1] 0 0 0 0 1 1 0 0 0 0 1   AIC: 6045.677
```

4.9 – População obtida por mutação e seus respectivos valores de AIC

Após esse procedimento, verificamos se existe algum indivíduo cujo valor de AIC do modelo associado é menor do que o que armazenamos no início da iteração (caso haja mais de um modelo escolhemos aquele com a menor quantidade de variáveis). No nosso caso, não há nessa iteração uma configuração de variáveis cujo valor de AIC seja menor do que o que armazenamos inicialmente. Assim, a melhor solução encontrada até agora permanece inalterada:

```
[[1]]
 [1] 0 1 1 1 1 0 0 0 1 0 1     AIC: 4821.188
```

4.10 – Configuração de variáveis escolhida na primeira iteração

.

Vejamos agora se obtemos uma solução melhor ao realizarmos 10 iterações:

```
[[1]]
```

```
 [1] 1 1 1 1 0 1 1 1 1 0 1   AIC: 4838.009

[[2]]
 [1] 1 1 1 1 0 1 1 1 1 0 1   AIC: 4838.009

[[3]]
 [1] 1 1 1 1 0 1 1 1 1 0 1   AIC: 4838.009

[[4]]
 [1] 1 1 1 1 0 1 1 1 1 0 1   AIC: 4838.009

[[5]]
 [1] 1 1 1 1 0 1 1 1 1 0 1   AIC: 4838.009

[[6]]
 [1] 1 1 1 1 0 1 1 1 1 0 1   AIC: 4838.009

[[7]]
 [1] 1 1 1 1 1 0 1 1 1 1 0   AIC: 4824.936

[[8]]
 [1] 1 1 1 1 1 0 1 1 1 1 0   AIC: 4824.936
```

4.11 – População obtida pelo A.G.E. após 10 iterações

. Vemos que em 10 iterações o algoritmo já está em uma população contendo indivíduos cujos menores valores de AIC são próximos do menor encontrado até agora (embora ainda sejam maiores).

Vejamos o resultado em 20 iterações:

```
[[1]]
 [1] 1 1 1 1 1 1 0 1 1 1 1  AIC: 4822.05

[[2]]
 [1] 1 1 1 1 1 1 0 1 1 1 1  AIC: 4822.05

[[3]]
 [1] 1 1 1 1 1 1 0 1 1 1 1  AIC: 4822.05

[[4]]
 [1] 1 1 1 1 1 1 0 1 1 1 1  AIC: 4822.05

[[5]]
 [1] 1 1 1 1 1 1 0 1 1 1 1  AIC: 4822.05

[[6]]
 [1] 1 1 1 1 1 1 0 1 1 1 1  AIC: 4822.05

```

```
[[7]]
 [1] 1 1 1 1 1 1 0 1 1 1 1  AIC: 4822.05

[[8]]
 [1] 1 1 1 1 1 1 0 1 1 1 1  AIC: 4822.05
```

4.12 – População obtida pelo A.G.E. após 20 iterações

. Na vigésima iteração, o AGE está uma população contendo cópias do mesmo indivíduo. Porém, nenhuma das configurações de variáveis nessa população apresentou valor de AIC menor do que o armazenado até o momento. Logo, nessa iteração a solução do problema de seleção de variáveis em *Credit* permanece a mesma encontrada na primeira iteração:

```
[[1]]
 [1] 0 1 1 1 1 0 0 0 1 0 1     AIC: 4821.188
```

4.13 – Configuração de variáveis escolhida como solução na vigésima iteração

. Assim, as variáveis selecionadas em *Credit* foram:

```
"Income"    "Limit"     "Rating"    "Cards"     "Student"   "Ethnicity
   "
```

4.14 – Variáveis escolhidas ao final da vigésima iteração

.

Para finalizar, quando executamos o AGE com os parâmetros declarados anteriormente 1000 vezes em uma população de 1000 indivíduos obtemos como solução do problema de seleção de variáveis o seguinte resultado:

```
[[1]]
 [1] 0 1 1 1 1 1 0 0 1 0 0    AIC: 4817.039
```

4.15 – Variáveis escolhidas após 1000 iterações

. Esta configuração apresenta valor de AIC menor do que as configurações obtidas no espaço de busca (todos os modelos possíveis). Além disso, tal configuração é a única que apresenta menor valor global de AIC (implicando esta ser a única solução possível para este problema) dentre todas as outras configurações no espaço de busca. Assim, as variáveis selecionadas foram:

```

"Income"  "Limit"   "Rating"  "Cards"   "Age"     "Student"
```

4.16 – Variáveis escolhidas após 1000 iterações

.

Observação. No caso do conjunto de dados *Credit*, existe uma única solução possível para o problema de seleção de variáveis,ou seja, existe única configuração de variáveis que

minimiza o AIC. Em outros conjuntos de dados, é possível existir múltiplas configurações de variáveis minimizando o AIC (múltiplas soluções do problema de seleção de variáveis) e cada uma dessas soluções podem ser adequadas ao contexto do problema específico a ser representado (a escolha de uma solução depende unicamente dos objetivos a serem alcançados com o modelo determinado pela configuração de variáveis).

Observação. Independentemente de escolhermos o AIC como função objetivo para ser otimizada pelo algoritmo genético, a unicidade de solução no problema de seleção de variáveis não pode ser garantida visto que dependendo do tipo de problema em que vamos usar as variáveis selecionadas pelo algoritmo genético, é possível que diversos subconjuntos de variáveis diferentes expliquem bem as observações da variável sob estudo.

Referências Bibliográficas

ACOSTA-GONZÁLEZ, E.; FERNÁNDEZ-RODRÍGUEZ, F. Model selection via genetic algorithms illustrated with cross-country growth data. *Empirical economics*, Springer, v. 33, n. 2, p. 313–337, 2007. Citado na página 36.

AHMED, S. E. *Penalty, shrinkage and pretest strategies: variable selection and estimation*. [S.l.]: Springer, 2014. Citado na página 10.

ANDERSON, D. A.; BURNHAM, K. P.; ANDERSON, D. R. *Model selection and inference: a practical information-theoretic approach*. [S.l.]: Springer, 1998. Citado 2 vezes nas páginas 11 e 34.

ANDRADE, B.; PEREIRA, A. On the genetic algorithm with adaptive mutation rate and selected statistical applications. *Computational Statistics (Zeitschrift) 30*, p. 131–150, 2015. Citado na página 27.

CAMPOS V.S.M., P. A.; CRUZ, J. Modeling the genetic algorithm by a non-homogeneous markov chain: weak and strong ergodicity. *Theory of Probability and its Applications , 1*, p. 185–192, 2012. Citado na página 26.

CAMPOS V.S.M., P. A. C. L.; ASSIS, I. Algoritmo genético por cadena de markov homogénea versus no-homogénea: un estudio comparativo. *Journal of the Chilean Institute of Operations Research 2*, 2012. Citado na página 27.

COMPUTING, R. R Foundation for S. R: a language and environment for statistical computing. *Vienna, Austria*, 2018. Citado na página 11.

CRUZ, J. Non-homogeneous markov chains convergence: strong and weak ergodicities. *Unpublished Ph.D. Thesis, Department of Mathematics, University of Brasilia, Brazil*, 1998. Citado na página 27.

CRUZ, J.; PEREIRA, A. The elitist non-homogeneous genetic algorithm: almost sure convergence. *Statistics and Probability Letters, 83*, p. 2179–2185, 2012. Citado na página 26.

DOREA C.C.Y., J. R.; PEREIRA, A. Multistage markov chain modeling of the genetic algorithm and convergence results. *Numerical Functional Analysis and Optimization , 31*, p. 164–171, 2010. Citado na página 26.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley, 1989. Citado 4 vezes nas páginas 10, 13, 23 e 30.

GREFENSTTETE, J. Optimization of control parameters for genetic algorithm. *IEEE Transactions on Fuzzy Systems, 16*, p. 122–128, 1986. Citado na página 26.

HOLLAND, J. Adaptation in natural and artificial systems. *Ann Arbor: The University of Michigan Press*, 1975. Citado na página 26.

HURVICH, C. M.; TSAI, C.-L. Bias of the corrected aic criterion for underfitted regression and time series models. *Biometrika*, Oxford University Press, v. 78, n. 3, p. 499–509, 1991. Citado na página 35.

ISLR package. Disponível em: <https://cran.r-project.org/package=ISLR>. Citado na página 11.

JAMES, G. et al. *An introduction to statistical learning*. [S.l.]: Springer, 2013. v. 112. Citado 3 vezes nas páginas 10, 11 e 30.

KARR, A. F. Markov chains: Theory and applications (dean l. isaacson and richard w. madsen). *SIAM Review*, Society for Industrial and Applied Mathematics, v. 20, n. 3, p. 606, 1978. Citado 2 vezes nas páginas 10 e 17.

LACERDA, E. G. de; CARVALHO, A. C. de; LUDERMIR, T. B. Model selection via genetic algorithms for rbf networks. *Journal of Intelligent & Fuzzy Systems*, IOS Press, v. 13, n. 2-4, p. 111–122, 2002. Citado na página 36.

LAWLER, G. F. *Introduction to stochastic processes*. [S.l.]: Chapman and Hall/CRC, 2018. Citado 2 vezes nas páginas 17 e 28.

PATERLINI, S.; MINERVA, T. Regression model selection using genetic algorithms. In: WORLD SCIENTIFIC AND ENGINEERING ACADEMY AND SOCIETY (WSEAS). *Proceedings of the 11th WSEAS international conference on nural networks and 11th WSEAS international conference on evolutionary computing and 11th WSEAS international conference on Fuzzy systems*. [S.l.], 2010. p. 19–27. Citado na página 36.

PURDIE LUCAS E.A., T. M. Direct measure of total cholesterol and its distribution among major serum lipoproteins. *Clinical Chemistry*, v. 38, n. 9, p. 1645–1647, 1992. Citado na página 36.

RAO, C. R.; TOUTENBURG, H. Linear models. In: *Linear models*. [S.l.]: Springer, 1995. p. 3–18. Citado 3 vezes nas páginas 10, 32 e 34.

RISH, I.; GRABARNIK, G. *Sparse modeling: theory, algorithms, and applications*. [S.l.]: CRC press, 2014. Citado 2 vezes nas páginas 10 e 30.

RUDOLPH, G. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks, 5*, p. 96–101, 1994. Citado na página 26.

RUDOLPH, G. Convergence analysis of canonical genetic algorithms. *IEEE transactions on neural networks*, IEEE, v. 5, n. 1, p. 96–101, 1994. Citado 2 vezes nas páginas 28 e 29.

R.W., J. Fitting percentage of body fat to simple body measurements. *Journal of statistics education*, n. 4, 1996. Citado na página 36.

SAKAMOTO, Y.; ISHIGURO, M.; KITAGAWA, G. Akaike information criterion statistics. *Dordrecht, The Netherlands: D. Reidel*, Taylor & Francis, v. 81, n. 10.5555, p. 26853, 1986. Citado na página 11.

SOBRINHO, P. d. S. *Algoritmos genéticos canônico e elitista: uma abordagem comparativa*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2014. Citado 2 vezes nas páginas 27 e 28.

SUGIURA, N. Further analysts of the data by akaike's information criterion and the finite corrections: Further analysts of the data by akaike's. *Communications in Statistics-theory and Methods*, Taylor & Francis, v. 7, n. 1, p. 13–26, 1978. Citado na página 35.

TEAM, R. C. et al. R: A language and environment for statistical computing. Vienna, Austria, 2013. Citado na página 11.

APÊNDICE A – Implementação do algoritmo genético elitista

Segue o código fonte do algoritmo genético elitista utilizado para simular o problema de seleção de variáveis no último capítulo deste trabalho.

```
1
2     rm(list=ls())
3
4
5
6  #funcao a ser minimizada
7
8  f<-function(dadosrecorte,popinicial){
9    n<-nrow(popinicial)
10   resultado<-c()
11   for(i in 1:n){
12     aux<-0
13     aux1<-c()
14     if(sum(popinicial[i,]==0)){
15       lm.fit=lm(Y~1, data=dadosrecorte)
16       aux<- AIC(lm.fit) #Aqui e o valor do AIC quando o n.de variaveis
       e zero.
17       }
18     if(sum(popinicial[i,])!=0){
19     aux1<-which(popinicial[i,]!=0)
20     dadosaux<-as.data.frame(dadosrecorte[,c(1,(aux1+1))])
21     lm.fit=lm(Y~., data=dadosaux)
22     #aux<-sum(abs(residuals(lm.fit))) #Se fosse tentar minimizar o EMQ
23     aux<-AIC(lm.fit)
24     }
25     resultado[i]<-aux
26   }
27   return(resultado)
28 }
29
30
31
32 #O espaco de estados sera dado por Uma lista de 11 entradas onde cada
     uma delas
33 #representa uma das variaveis da regressao, se a entrada
     correspondente a ela
```

```
#e 1, ela estara no modelo, se 0 ela nao estara no modelo. Usaremos o AG para
#escolher quais as variaveis que cuja regressao nos retorno o menor valor dos
#residuos.

#Espaco de estados com 2^11 elementos desde 00...0 ate 11...1

#Como o espaco de estados e muito grande teremos que otimizar a forma de gerar
#as populacoes de selecao,cruzamento e mutacao




fsel<-function(popsel,dadosrecorte){
  n<-nrow(popsel)
  aux<-c()
  aux<-f(dadosrecorte,popsel)
  m<-max(aux)
  pesos<-m-aux+1
  pescolha<-pesos/sum(pesos)
  escolha<-sample(1:n,n,prob = pescolha,replace=TRUE)
  paux<-popsel[escolha,]
  return(paux)
}

fpm<-function(pm,popsel){
  n<-nrow(popsel)-1
  aux<-c()
  popaux<-matrix(rep(0,31*(n+1)),ncol=31)
  popaux[n+1,]<-popsel[n+1,]
  for(i in 1:n){
      aux<-sample(c(0,1),31,replace = TRUE,prob=c(1-pm,pm))
      popaux[i,]<-abs(popsel[i,]-aux)
    }
  return(popaux)
}

#Nesse momento popsel e popaux devem ser iguais para aplicar o cruzamento

fpc<-function(pc,popsel, popaux){
  n<-nrow(popsel)-1
  aux<-usados<-c()
  aux<-sample(c(0,1),n,replace = TRUE,prob=c(1-pc,pc))
  usados<-which(aux==1)
```

```
77   q<-length(usados)
78   if ((q%/%2)>=1)
79   {
80     for (s in 1:(q%/%2))
81     {
82       pquebra<-sample(0:31,1)
83       if(pquebra!=0 & pquebra!=31){
84             popsel[usados[(2*s-1)],pquebra:31]<-popsel[usados[(2*s)],
     pquebra:31]
85             popsel[usados[(2*s)],pquebra:31]<-popaux[usados[(2*s-1)],
     pquebra:31]
86           }
87         }
88       }
89   return(popsel)
90   }
91
92
93   #Calculo do minimo
94
95 calcmin<-function(popsel,dadosrecorte){
96   aux<-c()
97   aux<-f(dadosrecorte, popsel)
98   aux1<-c()
99   aux1<-which.min(aux)
100  ifelse(length(aux1)>1,aux1<-sample(aux1,1),aux1<-as.numeric(aux1))
101  return(c(aux1,aux[aux1]))
102 }
103
104
105
106 #Comeca o algoritmo
107
108 bestconf<-function(pm=pm, pc=pc,npop=npop,npassos=10,dadosrecorte=
     dadosrecorte){
109
110 mudpm<-(pm-0.001)/npassos
111
112 resultado<-matrix(rep(0,as.numeric(33*npassos%/%100)),ncol=33)
113
114 #Gerando a populacao inicial
115
116 popaux<-popsel<-matrix(rep(0,31*(npop+1)),ncol=31)
117
118 #Escolhendo a matriz inicial aleatoriamente
119
120 popsel<-matrix(sample(c(0,1),31*(npop+1),replace=TRUE),ncol=31)
```

```
121
122 #Encontrando o minimo para colocar na ultima linha
123
124 linha<-calcmin(popsel[-(npop+1),],dadosrecorte)
125 popsel[(npop+1),]<-popsel[linha[1],]
126
127 #Comeca o loop do AG
128
129 for(o in 1:npassos){
130   if(o%%10){pm<-pm-mudpm}
131
132 popsel[1:npop,]<-fsel(popsel[-(npop+1),],dadosrecorte)
133
134 popaux<-popsel
135
136 popsel<-fpc(pc,popsel, popaux)
137
138 popsel<-fpm(pm,popsel)
139
140 #Vai trocar ou nao o melhor
141
142 #linha<-calcmin(popsel[-(npop+1),],dadosrecorte)[1]
143
144 linha1<-calcmin(popsel,dadosrecorte)
145
146 if(o%%100==0){
147   aux<-0
148   aux1<-c()
149   if(sum(popsel[linha1[1],]==0)){
150     lm.fit=lm(Y~1, data=dadosrecorte)
151     aux<- summary(lm.fit) #Aqui e o valor do R2 do modelo escolhido
     com o menor AIC.
152   }
153   if(sum(popsel[linha1[1],])!=0){
154     aux1<-which(popsel[linha1[1],]!=0)
155     dadosaux<-as.data.frame(dadosrecorte[,c(1,(aux1+1))])
156     lm.fit=lm(Y~., data=dadosaux)
157     #aux<-sum(abs(residuals(lm.fit))) #Se fosse tentar minimizar o EMQ
158     aux<-summary(lm.fit)
159   }
160 resultado[o%/%100,]<-c(popsel[linha1[1],],linha1[2],aux)
161 print(resultado[o%/%100,])
162 }
163 popsel[(npop+1),]<-popsel[linha1[1],]
164 }
165 return(resultado)
166 }
```

```
167
168 pm=0.99
169 pc=0.6
170 npop=1000
171 resultado<-bestconf(pm=pm, pc=pc,npop=npop,npassos=10000,dadosrecorte=
    dadosrecorte)
172
173
174
175 bestconf(pm=pm, pc=pc,npop=npop,npassos=200,dadosrecorte=dadosrecorte)
176
177
178 resultado<-matrix(rep(0,31*10),ncol=31)
179 for(j in 1:10){
180 resultado[j,]<-bestconf(pm=0.9, pc=0.5,npop=100,npassos=10000,
    dadosrecorte=dadosrecorte)
181 }
182
183
184
```

A.1 – Código fonte do algoritmo genético elitista