Uma Representação Restritiva para Regressão Simbólica

Projeto de pesquisa para o programa de doutorado em Ciência da Computação na UFABC.

Candidato: Fabrício Olivetti de França
Orientador: Prof. Dr. Fabrício Olivetti de França
E-mails: folivetti@ufabc.edu.br

Universidade Federal do ABC
Centro de Matemática, Computação e Cognição

2 de abril de 2018

O resumo deve conter uma breve descrição da motivação do projeto, o que você fará e resultados esperados.

Resumo

Uma ferramenta de análise muito utilizada em diversas aplicações é a análise de regressão. Ela consiste em determinar uma função que mapeia valores mensurados em um valor alvo que tem seu significado dependente do objeto de estudo. Dentre as ferramentas mais conhecidas estão a regressão linear e a rede neural artificial. A primeira tem como característica principal a simplicidade e facilidade de interpretação, que pode ser interessante para entender o funcionamento do sistema, porém ela é restritiva quanto as relações que pode aproximar. Já a segunda é conhecida como um aproximador universal, e permite aproximar qualquer relação dadas certas condições. Porém, a rede neural não permite uma fácil interpretação e é adequada apenas quando o objetivo do estudo demanda a predição da variável-alvo. Um meio termo entre essas duas é a Regressão Simbólica que busca por uma expressão que minimize o erro de aproximação enquanto mantém sua simplicidade. Essa técnica geralmente utiliza uma representação por meio de Árvore de Expressão, que permite representar qualquer expressão matemática, porém isso leva a um espaço de busca que contém tanto funções simples como complicadas, levando a múltiplos ótimos locais indesejados. Esse projeto tem o objetivo de avançar na área de Regressão Simbólica através da proposta de uma representação de expressões matemáticas que reduz o espaço de busca de soluções em torno apenas das expressões desejadas como solução.

Palavras-chave: análise de regressão, regressão simbólica, engenharia de atributos

1 Introdução

A **análise de regressão** compreende um conjunto de análises estatísticas com o objetivo de explicar as relações entre variáveis mensuráveis [6]. Esse tipo de análise é utilizado para fazer predições, estimativas ou entender um sistema de interesse.

Os primeiros parágrafos introduzem o leitor aos conceitos básicos necessários para o entendimento da proposta.

Exemplos de uso desse tipo de análise podem ser encontrados em diversas áreas de conhecimento. Predição de indicadores financeiros [11], análise de mercado [1], comportamento social [14], análise de performance de atletas [13], dentre outros.

Formalmente, podemos definir essa análise como, dado um conjunto de amostras d-dimensionais $X \in \mathbb{R}^{n \times d}$ [1], denominadas variáveis dependentes, e um conjunto de escalares $\mathbf{y} \in \mathbb{R}^{n \times 1}$, denominadas variáveis independentes. Busca-se por uma função $f : \mathbf{x} \rightarrow \hat{y}$ que aproxima a relação das medições efetuadas. Ou seja, desejamos obter $f(X) \approx \mathbf{y}$, ou equivalente, $f(X) = E(\mathbf{y} \mid X)$. A função de aproximação pode ser definida por um modelo **paramétrico** ou **não-paramétrico**.

Nos modelos paramétricos a função de aproximação é pré-definida através de parâmetros a serem ajustados, tornando-se efetivamente em $f(X, \beta)$. Para esse caso, partimos de uma forma fechada da função e o algoritmo de regressão ajusta apenas o parâmetro β (que pode ser um escalar ou um vetor d-dimensional). Esse tipo de modelo costuma ser menos custoso computacionalmente e, em alguns casos, permitir um entendimento do sistema estudado.

Já os modelos não-paramétricos, a forma da função é estimada e adaptada em torno das amostras mensuradas. Isso faz com que esse modelo se torne, em geral, mais custoso computacionalmente, necessite de muitas amostras e, muitas vezes, não permita uma interpretação do modelo gerado.

Um exemplo simples de modelo paramétrico é a **regressão linear**, que assume

[1] Nesse projeto adotaremos letras maiúsculas para representar matrizes, minúsculas em negrito ou letras gregas para vetores e minúsculas para escalares.

Apresente as notações adotadas no texto no rodapé ou no próprio texto, se for extensa

uma função paramétrica na forma:

$$f(\mathbf{x}, \beta) = \beta \cdot \mathbf{x},$$

com $(\cdot)$ representando o operador de produto interno. Esse modelo assume que a relação entre as variáveis independentes e a dependente é linear, o que muitas vezes não é o caso. Apesar dessa suposição simplista, esse modelo é muito utilizado por sua fácil interpretação. O modelo torna bem claro o quanto uma variável independente afeta o resultado do sistema.

Um exemplo simples de modelo não-paramétrico é a **média local** que simplesmente estima o valor de y como:

$$\hat{y} = \frac{1}{|\mathcal{N}(\mathbf{x})|} \sum_{i \in \mathcal{N}(\mathbf{x})} y_i,$$

em que $\mathcal{N}(x)$ retorna o índice dos vizinhos de $\mathbf{x}$. Basicamente, esse modelo calcula a média dos valores mensuradas da variável dependente em torno da vizinhança de $\mathbf{x}$.

Nesse modelo não temos como prever o quanto uma determinada variável afeta o sistema. Além disso, para obter um erro baixo de predição, é preciso obter um número suficiente de amostras em torno de cada $\mathbf{x}$ a ser avaliado.

Os modelos não interpretáveis são denominados de modelos **caixa-preta**, pois eles não explicitam a relação entre as variáveis independentes com a variável dependente.

apresentamos uma desvantagem dos métodos utilizados atualmente.

Um exemplo de modelo paramétrico que é considerado caixa-preta é o Perceptron de Múltiplas Camadas (do inglês *Multi-Layer Perceptron* ou MLP). O MLP define um modelo computacional baseado em grafos que permite a criação de modelos paramétricos, como por exemplo:

$$\hat{y} = f(\mathbf{x}, \beta, \gamma) = \gamma \cdot \mathbf{g}(B \cdot \mathbf{x}),$$

com $\mathbf{g}(.)$ sendo um vetor de funções denominadas de função de ativação. A função de ativação geralmente é uma função não-linear com formato sigmoidal.

Esse modelo apresenta a característica de um aproximador universal [4]. Dados os valores corretos para as dimensões de B, γ e de seus valores, é possível aproximar qualquer função salvo um erro pequeno ϵ.

Uma outra família de algoritmos capazes de encontrar as relações entre variáveis independentes e dependente é a Regressão Simbólica [2], geralmente representados pelos algoritmos de Programação Genética [5].

Na Regressão Simbólica, tanto a forma da função como os coeficientes são ajustados buscando atender dois objetivos principais: a minimização do erro de aproximação e a maximização da simplicidades da função. O significado de *simplicidade* no contexto de Regressão Simbólica se refere a facilidade de interpretação. Como exemplo, vamos ver as seguintes funções:

$$f(x) = \frac{x^3}{6} + \frac{x^5}{120} + \frac{x^7}{5040} \tag{1}$$

$$f(x) = \frac{16x(\pi - x)}{5\pi^2 - 4x(\pi - x)} \tag{2}$$

$$f(x) = \sin x. \tag{3}$$

Assumindo que a terceira função é a função que gerou os dados mensurados, as duas primeiras funções podem ser consideradas aproximações razoáveis dentro de um determinado domínio. Se considerarmos, porém, o objetivo de interpretação do comportamento dos dados mensurados, a terceira função é a única capaz de ser lida claramente por um especialista. A simplicidade da função geralmente é

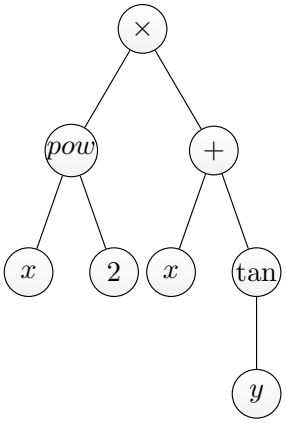

Figura 1.1: Expression tree for the expression $x^2 \cdot (x + \tan y)$.

medida através de uma combinação linear entre o tamanho da árvore de expressão e o número de funções não-lineares utilizadas.

Uma árvore de expressão é uma estrutura de dados de árvore n-ária representando a função de aproximação. Cada nós da árvore pode representar uma função, uma variável ou uma constante. Os nós representando funções devem necessariamente possuir n filhos de acordo com a aridade da função. As variáveis e constantes são nós folhas dessa árvore e, portanto, nós terminais. Por exemplo, a expressão $x^2 \cdot (x + \tan y)$ pode ser representada como a árvore da Fig. 1.1. Nesse exemplo, o tamanho da expressão pode ser medida pelo número de nós (8) ou a altura da árvore (3).

A árvore de expressão permite a representação de qualquer expressão matemática. Seu espaço de busca contém, portanto, infinitos elementos. Embora isso permita uma maior flexibilidade na busca por uma solução com erro mínimo, por outro lado aumenta a complexidade da busca e permite diversas soluções alternativas não-interpretáveis, conforme exposto nas Eqs. 3. Isso faz com que o processo de busca esteja sujeito a convergência para ótimos locais indesejados e, potencialmente

apresentamos aqui algumas desvantagens do estado-da-arte atual, que queremos resolver.

gerando uma solução tão *caixa-preta* quanto uma Rede Neural.

Para aliviar esse problema da complexidade do espaço de busca, esse projeto irá propôr uma nova representação de expressões matemáticas restritiva, capaz de manter certa flexibilidade quanto as expressões que podem ser representadas, mas eliminando expressões consideradas complexas ou não-interpretáveis.

O que eu farei para resolver o problema exposto acima.

Embora um dos pontos fortes da Regressão Simbólica seja justamente a possibilidade em representar qualquer expressão matemática, o que permite encontrar exatamente a função geradora do sistema estudado, isso também faz com que tais técnicas estejam sujeitas ao *sobreajuste* e aumenta a probabilidade em encontrar soluções muito mais complicadas do que realmente deveriam ser.

por que esse estudo é interessante?

Uma representação restritiva, se bem formulado, pode continuar permitindo um poder de aproximação alto o suficiente para representar as funções necessárias para o entendimento de boa parte dos sistemas. Partindo do pressuposto de que grande parte dos sistemas estudados na física e nas engenharias são descritos como expressões simples [8], essa restrição se torna não só justificável como desejável.

1.1 Objetivos Gerais e Específicos

O objetivo principal desse projeto é a proposta de uma nova representação de expressões matemáticas que:

- Permita a representação de expressões matemáticas mais flexíveis do que uma simples relação linear.
- Não permita a representação de expressões excessivamente complicadas como a de uma Rede Neural.

Especificamente, podemos listas os seguintes objetivos a serem cumpridos ao longo do projeto:

- Estudo sistemático de sistemas reais e as expressões matemáticas que os representam.
- Definição de uma forma de função genérica capaz de representar a maioria desses sistemas.
- Proposta de uma estrutura de dados computacional para essa representação, levando em conta as operações que serão realizadas nela.
- Proposta de um algoritmo de busca para encontrar a melhor expressão para uma base de dados representando um sistema alvo de estudo.

De acordo com esses objetivos, formulamos as seguintes questões de pesquisa a serem respondidas ao longo da pesquisa:

Dependendo do projeto, troque por hipótese.

RQ1 Uma representação restritiva é capaz de encontrar relações entre variáveis dependentes e independentes em uma análise de regressão com erro mínimo?

RQ2 A forma da função encontrada é interpretável?

RQ3 O algoritmo proposto apresenta um desempenho similar aos algoritmos estado-da-arte de análise de regressão?

2 Proposta

Essa seção começa com um breve resumo da literatura de Regressão Simbólica e formas de aliviar o problema do espaço de busca induzido pela árvore de expressão. Em seguida, detalharemos a proposta desse projeto de pesquisa.

2.1 Programação Genética

O algoritmo de Programação Genética tenta encontrar a forma de função que minimiza o erro de aproximação e, ao mesmo tempo, maximiza a simplicidade da expressão.

Conte uma história resumida da área de pesquisa que irá estudar, justificando e contextualizando sua proposta.

Esse algoritmo faz parte dos Algoritmos Evolutivos que se baseiam na teoria da evolução para otimizar uma população de soluções. Esses algoritmos compartilham a mesma sequência de passos diferenciando entre si apenas na definição de suas funções internas. O algoritmo genérico de um Algoritmo Evolutivo está descrito em Alg. 1.

Algoritmos devem ser preferencialmente declarativos ao invés de imperativos.

Algorithm 1: Algoritmo Evolutivo

input : população P de soluções iniciais, quantidade de gerações g, função-objetivo de maximização f.
output: a melhor solução $p \in$ P.

for g *geracoes* **do**
    P' ← `cruzamento(P);`
    P" ← `mutacao(P');`
    P ← `selecao(P");`
return $\text{argmax}_{p \in \mathsf{P}} f(p)$

Iniciando com uma população de soluções, geralmente criadas aleatoriamente, repete-se por g gerações os passos de *cruzamento*, em que duas ou mais soluções são combinadas para gerar novas soluções, *mutação*, em que cada solução pode sofrer pequenas alterações na solução, e *seleção*, em que as soluções transformadas são selecionadas para a próxima geração.

A ideia geral é que ao combinar duas ou mais soluções boas, a informação contida nelas é passada para as soluções *filhas* com a esperança de encontrar uma solução melhor.

Na Programação Genética as soluções são representadas na forma de árvores de

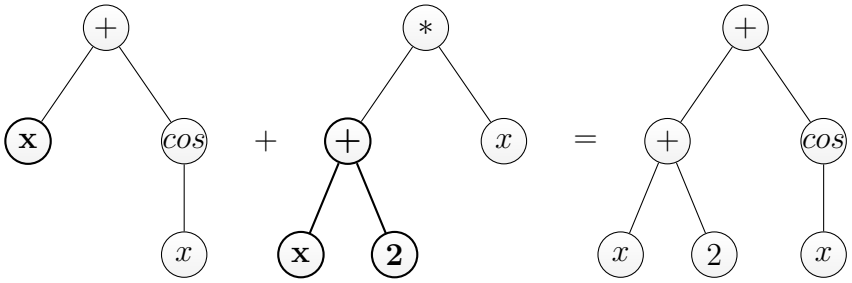

Figura 2.1: Cruzamento entre duas soluções.

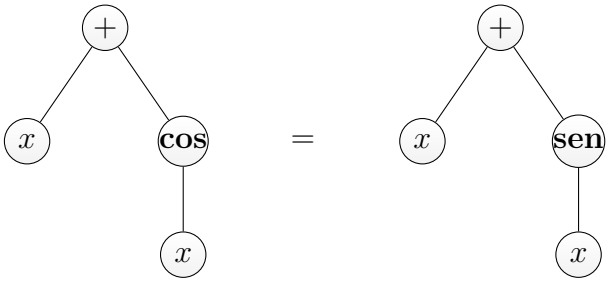

Figura 2.2: Mutação de uma solução.

expressão. A árvore de expressão tem a vantagem de permitir apenas expressões válidas em todos os procedimentos de transformação.

A função de cruzamento combina partes de duas árvores gerando uma nova árvore de solução. Essa operação tem o objetivo de capturar partes de duas soluções boas que se complementem, gerando uma solução ainda melhor. O procedimento é ilustrado na Fig. 2.1.

Já a função de mutação, altera um nó ou uma sub-árvore da árvore de expressão. O intuito dessa função é gerar uma pequena modificação da expressão de tal forma a melhorar incrementalmente a solução final. Esse procedimento é ilustrado na Fig. 2.2

O procedimento de seleção escolhe quais soluções dentre a população gerada pelo processo de cruzamento e mutação substituirá a população atual. Isso geralmente é feito através do *torneio*, em que pares de soluções são amostradas aleatoriamente

para disputar quem será escolhido para a próxima geração.

Apesar do sucesso obtido pelo algoritmo de Programação Genética [7], a representação por árvore e as funções de cruzamento e mutação não possuem características desejáveis em um algoritmo evolutivo. Primeiramente, a representação de uma solução deve permitir que parte de uma solução boa possa ser repassada para as futuras gerações, porém a extração de uma parte da árvore de expressão não garante que ela irá contribuir positivamente com outra solução por conta da dependência dos nós pais.

Por exemplo, na expressão $cos(x^2)$, a expressão x^2 pode contribuir negativamente em outra expressão caso a função de transformação ($cos()$) seja alterada. Da mesma forma, a mutação pode alterar totalmente a resposta da expressão ao alterar um único nó.

Para resolver esses problemas foram propostos operadores de cruzamento e mutação sensíveis ao contexto ou à semântica [12]. Para tanto é definido o espaço semântico como um vetor $\mathbf{s} = \{f'(\mathbf{x_1}, ..., \mathbf{x_n}\}$ como sendo a avaliação da expressão definida por uma sub-árvore extraida da árvore de expressão nos pontos mensurados utilizados como referência. Dessa forma, o cruzamento faz a troca de sub-árvores com um vetor semântico similar. A mutação permite apenas alterações que influenciam pouco no vetor semântico.

Conforme notado em [10] as expressões geradas pela variante Programação Genética com Semântica Geométrica (*Geometric Semantic Genetic Programming*), uma das variantes mais utilizadas atualmente, são combinações lineares de expressões aleatórias, o que pode ser obtida de forma mais rápida e precisa por outros métodos.

Outra variante é a Programação de Expressão Gênica (*Gene Expression Programming*) que utiliza uma representação de tamanho fixo e linear denominada

expressão-K. A expressão é formada por uma cabeça de tamanho h, definida pelo usuário, contendo terminais e não-terminais, e uma cauda de tamanho $h(n-1)+1$, com n sendo o número de terminais, contendo apenas terminais. A expressão-K é facilmente transformada em uma Árvore de Expressão ao ser lida da esquerda para a direita e expandindo os nós como em uma busca em profundidade.

Uma vantagem dessa técnica é o controle do tamanho da expressão, definido pelo parâmetro h. Porém, diversos experimentos da literatura não conseguiram notar melhoras em relação ao algoritmo de Programação Genética Padrão, em muitos casos obtendo soluções piores [9].

2.2 Representação Restritiva

A ideia geral desse projeto parte da forma da função de uma expressão linear $\hat{f}(\mathbf{x}) = \beta \cdot \mathbf{x}$. Tal forma, embora originalmente restritiva, pode capturar relações não-lineares ao aplicar funções de transformação das variáveis originais para um novo espaço de variáveis $z = g(\mathbf{x})$. Uma função de transformação frequentemente utilizada é a interação entre as variáveis:

A sua proposta deve ser uma ideia concreta, não pode ser algo vago. É necessário especificar o "como"e não só o "o que".

$$\hat{f}(\mathbf{x}) = \beta \cdot \mathbf{p}(\mathbf{x}),$$

com p_i sendo definida como:

$$p_i(\mathbf{x}) = \prod_{j=d}^{d} x_j^{a_j}.$$

Isso é conhecido como **regressão polinomial** [3]. Além disso, em muitos casos um determinado atributo de $\mathbf{x}$ é *transformado* através de uma função não-linear univariada ($t(.)$) para tentar linearizar sua relação com a variável dependente. Essa transformação costuma ser feita através das funções *logaritmo, tangente hiperbólica,*

gaussiana, etc.

Partindo dessas duas técnicas de engenharia de atributos, proporemos uma representação que crie novos atributos a partir da composição de uma função de transformação com uma função de interação polinomial.

A composição é possível pois $t : \mathbb{R} \to \mathbb{R}$ e $p_i : \mathbb{R}^d \to \mathbb{R}$, gerando então $g = p_i \circ t$. Com isso, a função de regressão será uma combinação linear de diversas dessas funções g.

Dessa forma, tal representação não permitirá encadeamento de funções não-lineares, sendo possível apenas uma função de transformação e uma de interação por termo linear.

Os atributos gerados por essas composições de funções serão combinados pela mesma forma de uma regressão linear, gerando efetivamente a seguinte forma fechada:

$$\hat{f}(\mathbf{x}) = \beta \cdot \mathbf{g}(\mathbf{x}),$$

com a ressalva de que cada elemento de $\mathbf{g}$ é uma função distinta.

3 Infraestrutura Necessária

A infraestrutura necessária para a execução desse projeto é de apenas um computador pessoal de médio porte a ser utilizado para a pesquisa de artigos da literatura, criação dos códigos-fontes associados ao projeto e execução de experimentos.

4 Cronograma

O cronograma de execução do projeto está dividido em tarefas com período de conclusão mínimo de quatro meses (quadrimestre), totalizando 12 quadrimestres até a conclusão do doutorado. As tarefas a serem realizadas são listadas em seguida e seu tempo de conclusão ilustrados na Fig. 4.1:

1. Revisão da literatura de Análise de Regressão, Regressão Simbólica e Programação Genética.
2. Implementação ou análise de códigos-fontes pertinentes da revisão anterior.
3. Estudo sistemático de expressões matemáticas utilizadas na área de física e engenharias de tal forma a traçar um padrão de forma de função.
4. Verificação dos padrões de forma de função do estudo anterior.
5. Escrita e apresentação da qualificação.
6. Formulação da representação restritiva e estrutura de dados computacionais relacionadas.
7. Proposta de heurísticas construtivas, de busca local e meta-heurísticas utilizando a representação.
8. Experimentos e avaliações de resultados.
9. Publicações.
10. Escrita e defesa da tese.

5 Revistas e Conferências Relevantes

Dentre os veículos de comunicação relevantes para essa pesquisa destacam-se as revistas:

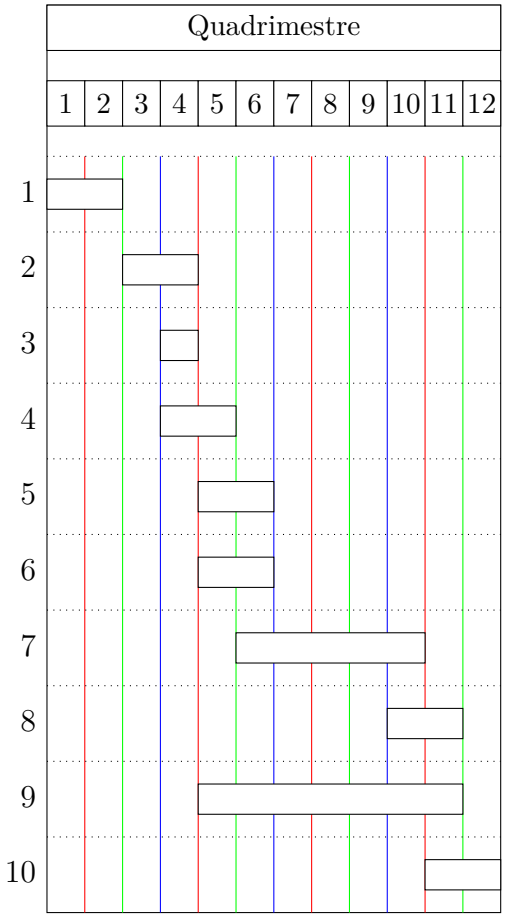

Figura 4.1: Cronograma estimado das tarefas do projeto proposto.

- Information Sciences, qualis $A1$ na Ciência da Computaçao.
- IEEE Transactions on Evolutionary Computation, qualis $A1$ na Ciência da Computação.
- Evolutionary Intelligence, qualis $B1$ na Ciência da Computação.

E os congressos:

- Genetic and Evolutionary Computation Conference (GECCO), qualis $A2$ na Ciência da Computação.
- IEEE Congress on Evolutionary Computation (CEC), qualis $A2$ na Ciência da

Computação.

6 Considerações Finais

Muitos algoritmos de análise de regressão são caracterizados por um dentre dois possíveis extremos: i) simples de interpretar, mas com baixo poder de aproximação ou, ii) complicados de interpretar, com alto poder de aproximação. Uma técnica muito utilizada para encontrar um meio termo é a Regressão Simbólica que, embora tenha potencial de encontrar expressões matemáticas simples e com baixo de erro de aproximação, falha em muitos casos de estudo por conta do enorme espaço de busca induzido por sua representação padrão.

Nesse projeto esperamos encontrar uma forma de representação que reduza o espaço de busca em torno apenas das expressões matemáticas desejáveis para uma função de regressão interpretável.

7 Nota aos candidatos

Esse projeto ultrapassou as 15 páginas por utilizar uma margem mais larga para as anotações. Com a margem correta (vide modelo no site da Poscomp), esse texto possui exatamente 15 páginas. É importante não ultrapassar esse limite.

Bibliografia

[1] Lee G Cooper and Masako Nakanishi. *Market-share analysis: Evaluating competitive marketing effectiveness*, volume 1. Springer Science & Business Media, 1989.

[2] JW Davidson, Dragan A Savic, and Godfrey A Walters. Symbolic and numerical regression: experiments and applications. *Information Sciences*, 150(1):95–117, 2003.

[3] A Ronald Gallant and Wayne A Fuller. Fitting segmented polynomial regression models whose join points have to be estimated. *Journal of the American Statistical Association*, 68(341):144–147, 1973.

[4] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[5] Ilknur Icke and Joshua C Bongard. Improving genetic programming based symbolic regression using deterministic machine learning. In *2013 IEEE Congress on Evolutionary Computation*, pages 1763–1770. IEEE, 2013.

[6] Robert E Kass. Nonlinear regression analysis and its applications. *Journal of the American Statistical Association*, 85(410):594–596, 1990.

[7] John R Koza. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):251–284, 2010.

[8] Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, 2017.

[9] Mihai Oltean and Crina Grosan. A comparison of several linear genetic programming techniques. *Complex Systems*, 14(4):285–314, 2003.

[10] Tomasz P Pawlak. Geometric semantic genetic programming is overkill. In *European Conference on Genetic Programming*, pages 246–260. Springer, 2016.

[11] Theodore B Trafalis and Huseyin Ince. Support vector machine for regression and applications to financial forecasting. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 6, pages 348–353. IEEE, 2000.

[12] Leonardo Vanneschi, Mauro Castelli, and Sara Silva. A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines*, 15(2):195–214, 2014.

[13] BRENDA L Webster and SUSAN I Barr. Body composition analysis of female adolescent athletes: comparing six regression equations. *Medicine and science in sports and exercise*, 25(5):648–653, 1993.

[14] Kathryn R Wentzel. Does being good make the grade? social behavior and academic competence in middle school. *Journal of Educational Psychology*, 85(2):357, 1993.