REGRESSÃO SIMBÓLICA COM SELEÇÃO POR AMOSTRAGEM

Rodrigo Morgon[1], Silvio do Lago Pereira[2]
[1]Aluno do Curso de Análise e Desenvolvimento de Sistemas – FATEC-SP
[2]Prof. Dr. do Departamento de Tecnologia da Informação – FATEC-SP
rodrigomorgon@gmail.com, slago@pq.cnpq.br

Resumo

Regressão é uma técnica usada para verificar a relação existente entre duas variáveis observadas e obter um modelo que possa expressar essa relação. A abordagem tradicional de regressão usa modelos predefinidos, sendo inviável quando não se tem uma suposição a respeito do melhor modelo a ser adotado. A regressão simbólica, por outro lado, é capaz de encontrar o melhor modelo sem nenhuma hipótese adicional. O objetivo desse artigo é propor um algoritmo evolucionário para regressão simbólica, e com base nele, criar um sistema correspondente chamado SRS (*Symbolic Regression with Sampling*). A novidade no algoritmo é o uso de *amostragem* para selecionar indivíduos da população que terão descendentes na próxima geração. A conjectura é que o uso de amostragem pode acelerar a evolução da população e reduzir o tempo necessário para obtenção do melhor modelo procurado. Os resultados empíricos confirmaram essa conjectura.

1. Introdução

Regressão [1] é uma técnica usada em aplicações práticas em diversas áreas, para verificar se duas variáveis observadas estão relacionadas de alguma forma e, caso estejam, definir um modelo que expresse essa relação. Este tipo de modelagem permite compreender como o comportamento de uma variável x influencia o comportamento de outra variável y. A relação entre essas variáveis pode ser vista como um *processo* com *entrada* x e *saída* y, i.e., $y = f(x)$. Observando-se esse processo, é possível obter vários pares de entrada e saída, $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, a partir dos quais um modelo correspondente pode ser formulado. Como a função $f(x)$ é desconhecida (i.e., representa um processo do tipo *caixa-preta*), o problema de regressão consiste em encontrar uma função $h(x)$ que produza saídas "próximas" daquelas geradas pela função $f(x)$, relativas aos pares de entrada e saída considerados. Ao contrario da regressão tradicional, que adota um modelo de função predefinido e apenas ajusta coeficientes, a *regressão simbólica* [1] encontra o modelo propriamente dito (com coeficientes devidamente ajustados).

Neste artigo, o objetivo é propor um *algoritmo evolucionário* [2] para o problema de regressão simbólica, bem como descrever um sistema correspondente, denominado SRS (*Symbolic Regression with Sampling*).

A motivação para essa proposta foi inspirada no sistema ECL (*Evolutionary Concept Learner*) [3], que trata um problema correlato (porém, com variáveis discretas) e apresenta excelente desempenho quando comparado a sistemas tradicionais da área de *aprendizado de máquina* [4].

Assim como o ECL, o sistema SRS proposto nesse artigo é um algoritmo evolucionário que manipula uma população de indivíduos, cada um deles representando uma possível hipótese (i.e. melhor modelo procurado). Inicialmente, os indivíduos na população são gerados aleatoriamente (de acordo com regras que definem que componentes, ou *genes*, podem ser usadas para criar um indivíduo). Essa população é denominada *geração zero*. A partir daí, a cada nova geração, a aptidão dos indivíduos é calculada (e.g., proximidade da saída esperada em relação àquela obtida) e alguns dos melhores indivíduos são selecionados para produzir descendentes na próxima geração. O processo é repetido até que uma condição terminal seja satisfeita (e.g., até o tempo máximo de execução seja atingido). No final, o melhor indivíduo na última geração é devolvido como resposta do algoritmo.

O restante desse artigo está organizado da seguinte forma: a Seção 2 introduz os fundamentos da regressão simbólica; a Seção 3 descreve o algoritmo proposto, bem como o sistema correspondente implementado; a Seção 4 descreve resultados empíricos obtidos com esse sistema; e, por fim, a Seção 5 apresenta as conclusões do trabalho.

2. Regressão Simbólica

Esta seção introduz os fundamentos da regressão simbólica, necessários para entender o algoritmo proposto.

2.1. Problemas de Regressão

A abordagem tradicional para regressão consiste em supor que $h(x)$ tem uma forma padrão (e.g., $ax^2 + bx + c$) e que apenas seus coeficientes precisam ser encontrados, para que ela se ajuste aos dados observados. O problema com essa abordagem é que diferentes funções precisam ser testadas até que uma apropriada (i.e., que se ajuste bem aos dados observados) seja encontrada. Assim, o resultado desse tipo de análise depende muito da habilidade de quem escolhe as funções a serem testadas. Por isso, até mesmo entre especialistas, é uma prática comum testar apenas funções lineares e quadráticas, ainda que modelos mais complexos possam produzir melhores resultados.

A *regressão simbólica* [1], por outro lado, consiste em encontrar uma função que se ajuste ao conjunto de dados observados, sem que qualquer suposição sobre a forma da função precise ser feita. Diferentemente da abordagem tradicional, a regressão simbólica encontra não apenas os coeficientes de uma função, mas a própria função (seja ela linear ou não). Essa ideia é ilustrada na Figura 1.

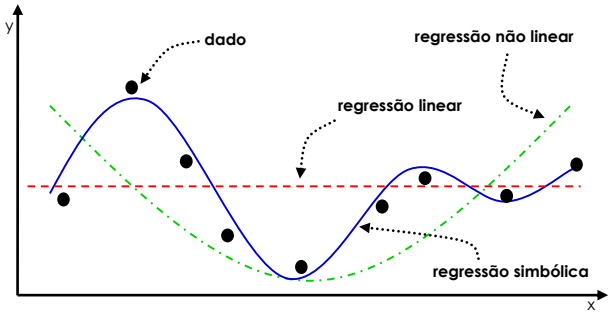

Figura 1 – Problemas de regressão.

Para solucionar o problema de regressão, o sistema correspondente ao algoritmo proposto formula hipóteses aproximadas através do seu aprendizado na observação dos objetos contidos no conjunto de dados.

2.2. Objetos e Conjuntos de Dados

Na regressão simbólica, um *objeto* é um par ordenado (x, y), em que x representa a *entrada* e y representa a *saída* de um processo. Um conjunto de dados observados para um processo caixa-preta é chamado *conjunto de dados*.

À primeira vista, pode parecer que quanto maior o conjunto de dados, mais fácil é encontrar um modelo que se ajuste a eles. Na prática, entretanto, observa-se que uma quantidade excessiva de dados pode levar a um *superajustamento* do modelo obtido, isto é, o modelo se ajusta perfeitamente ao conjunto de dados observados, mas não é capaz de extrapolar esse conjunto (i.e., se ajustar a novos objetos, ainda não observados).

Para evitar o superajustamento do modelo, o *conjunto de dados* usado na regressão é dividido em duas partes: *conjunto de treinamento* e *conjunto de validação.* O conjunto de treinamento é usado pelo sistema para encontrar *hipóteses* que satisfaçam os objetos observados, ou seja, uma função h tal que $y = h(x)$. Por outro lado, o conjunto de validação é usado para verificar se as hipóteses encontradas também valem para objetos que não estavam no conjunto de treinamento (ou seja, o conjunto de validação é usado para avaliar a capacidade de extrapolação dos modelos obtidos).

2.3. Avaliação de Hipóteses

A qualidade de uma hipótese reflete o quanto ela se aproxima da função desconhecida (sobre a qual se conhece apenas os pares de entrada e saída observados). De fato, a qualidade de uma hipótese pode ser definida em termos da diferença entre a saída que ela produz e a saída que é esperada. Por exemplo, seja $f(x) = 2x + 1$ a função que modela um processo caixa-preta observado (evidentemente, na prática, essa função é desconhecida). Então, a entrada 2 produz a saída 5. Esse fato é representado pelo objeto observado $(2,5)$. Suponha agora que a hipótese encontrada seja $h(x) = 2x * 1$. Então, quando a hipótese é avaliada com a entrada 2, obtém-se a saída 4, representada pelo par $(2,4)$. A diferença entre a saída obtida e a saída esperada, nesse caso, é 1. Essa diferença é denominada *erro*. Assumimos que quanto menor for o *somatório dos erros quadrados* obtidos, em relação a todos os objetos no conjunto de dados considerado, maior a qualidade de uma hipótese.

3. Inovações na Abordagem Proposta

O algoritmo usado como base para a implementação do SRS é um algoritmo de busca estocástica, inspirado na teoria da evolução de Darwin, também chamado de *algoritmo evolucionário*. Esse tipo de algoritmo possui as seguintes características principais: uso de uma população de indivíduos representando soluções candidatas; uso de operadores genéticos para geração de novos indivíduos na população; e uso de uma medida de aptidão para selecionar indivíduos que devem sobreviver de uma geração para outra e/ou produzir descendente. O ciclo básico de um algoritmo evolucionário é ilustrado na Figura 2.

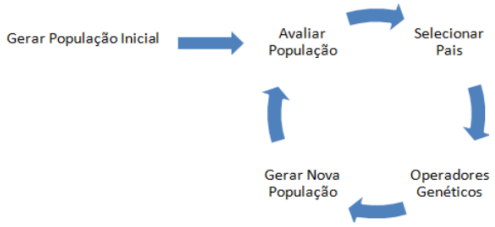

Figura 2 – Ciclo de um algoritmo evolucionário.

Para gerar novos indivíduos, o algoritmo proposto nesse artigo emprega operadores genéticos de mutação e cruzamento. Para selecionar os indivíduos usados nessas operações, o algoritmo usa um novo método (descrito na Subseção 3.5), denominado *seleção por amostragem*. Como observado em experimentos realizados (relatados na Seção 4), esse método é capaz de acelerar a evolução da população e reduzir o tempo necessário para obtenção das melhores hipóteses.

3.1. Criação de Árvores e População Inicial

Uma expressão aritmética pode ser representada por uma árvore cujas folhas são rotuladas com uma variável x e cujos nós internos são rotulados com operadores aritméticos $(+, -, *, /)$. Por exemplo, a Figura 3 mostra uma árvore representando a expressão $x^2 - x + 1$. Note que, mesmo usando apenas variáveis, é possível criar expressões equivalentes a constantes (e.g., $x - x = 0$ e $x/x = 1$, para $x \neq 0$).

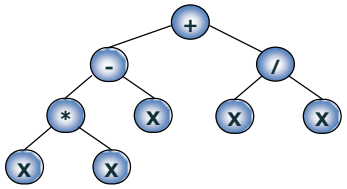

Figura 3 - Expressão aritmética representada por árvore

A representação de expressões em forma de árvore permite que operações genéticas, como mutação e cruzamento, sejam efetuadas mais facilmente.

Em programação genética [1], há dois métodos básicos de geração de árvores aleatórias para compor a população inicial: o método *full* gera apenas árvores completas, nas quais todas as folhas estão no mesmo nível; por outro lado, o método *grow* é capaz de gerar árvores de vários formatos. A implementação desses métodos é relativamente simples. Porém, nenhum deles garante uma distribuição uniforme das árvores na população inicial.

No algoritmo proposto nesse artigo, uma distribuição mais uniforme das árvores aleatórias na população inicial é baseada em um limite superior para o número de árvores que podem ser criadas, com altura de no máximo *h* níveis, compostas por operadores *nulários*, *unários* e *binários*. Esse limite superior, denotado por T_h, pode ser indutivamente definido como segue: Se $h = 1$, a raiz da árvore deve ser rotulada por uma *variável*[1] (tratada como um operador nulário, isto é, um operador que não tem operandos) e, então, $T_1 = k$ (sendo k o número de variáveis consideradas). Caso contrário, se $h > 1$, cada árvore enraizada num nó rotulado por uma variável é uma árvore de altura no máximo h; cada árvore de altura no máximo $h-1$, estendida com uma raiz rotulada com operador unário é uma árvore de altura no máximo *h*; e cada par de árvores de altura no máximo $h-1$, combinadas por uma raiz rotulada com um operador binário é uma árvore de altura no máximo *h*; portanto, $T_h = k + T_{h-1} + 2 \cdot T_{h-1}^2$. Assim, uma distribuição uniforme de árvores aleatórias pode ser garantida se, durante a criação de uma árvore, as escolhas de variáveis e operadores (unário e binário) para rotular sua raiz forem feitas com probabilidades iguais a k/T_n, T_{n-1}/T_n, e $(2 \cdot T_{n-1}^2)/T_n$, respectivamente, e se suas subárvores forem criadas recursivamente.

Na versão do sistema considerado nesse artigo, as folhas das árvores ainda podem ser rotuladas com constantes reais ou potências (e.g., x^2), geradas aleatoriamente. Embora isso possa aumentar a diversidade sintática das árvores geradas, em termos semânticos, não há influência.

3.2. Mutação

Depois de criada, uma árvore pode ser modificada pela função de *mutação*. Essa função seleciona aleatoriamente uma operação de modificação, aplica essa operação na árvore e devolve a árvore mutante resultante. Há dois tipos de operações de modificação que podem gerar uma mutação de uma árvore: *interna* e *externa*.

Quando uma modificação interna é aplicada, um nó interno da árvore é aleatoriamente selecionado para sofrer mutação e seu rótulo é aleatoriamente substituído por outro do mesmo tipo (preservando a altura da árvore, como mostra a Figura 4) ou, então, a subárvore enraizada nesse nó é substituída por uma árvore aleatória de altura no máximo 2 (possivelmente diminuindo a altura da árvore, como mostra a Figura 5).

Por outro lado, quando uma modificação externa é aplicada, um nó externo da árvore é aleatoriamente selecionado para sofrer mutação e seu rótulo é aleatoriamente substituído por outro do mesmo tipo (preservando a altura da árvore, como mostra a Figura 6) ou, então, a subárvore enraizada nesse nó é substituída por uma árvore aleatória de altura no máximo 2 (possivelmente aumentando a altura da árvore, como mostra a Figura 7).

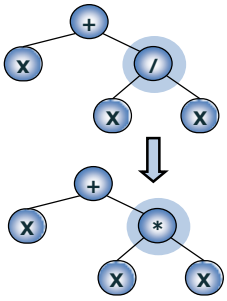

Figura 4 – Mutação por modificação interna

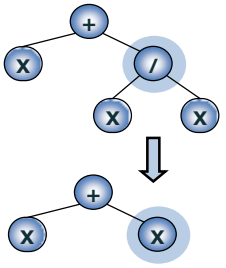

Figura 5 – Mutação por modificação interna

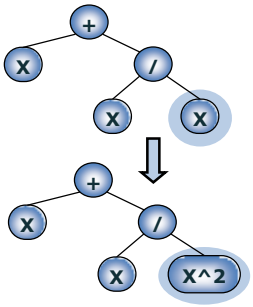

Figura 6 – Mutação por modificação externa

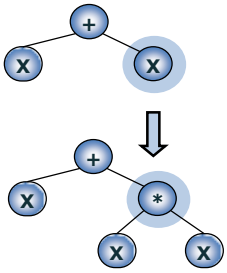

Figura 7 – Mutação por modificação externa

3.3. Cruzamento

Além da operação de mutação, o algoritmo proposto também usa a operação de *cruzamento*, que cria uma nova árvore a partir de duas outras selecionadas da população, cujas características estarão presentes na árvore resultante da operação.

Essa função seleciona aleatoriamente o ponto em que a árvore será segmentada para o cruzamento com outra árvore. O resultado desse cruzamento permite que novas combinações e possibilidades de aproximação da função desejada sejam obtidas. Essa ideia é ilustrada na Figura 8.

[1] Embora o artigo trate apenas de funções *univariadas*, o método também serve para funções *multivariadas* (i.e., com múltiplas entradas).

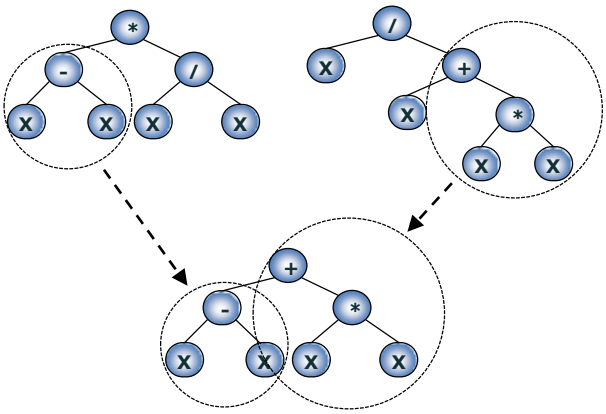

Figura 8 – Resultado de cruzamento entre indivíduos

3.4. Avaliação de Árvores

Cada árvore representa um indivíduo (i.e., uma possível hipótese para o modelo da função a ser encontrada pela regressão). Logo, a qualidade da árvore é avaliada de acordo com o resultado do erro considerado na saída obtida pela mesma (vide Subseção 2.3).

No algoritmo proposto, a avaliação é feita com base no somatório do erro quadrático gerado pela hipótese no conjunto de treinamento. O erro quadrático, definido pela expressão (1), evita que erros positivos e negativos se anulem, o que causaria uma melhora irreal no desempenho do modelo avaliado.

$$\sum_{i=1}^{n}(h(x_i) - y_i)^2 \qquad (1)$$

3.5. Seleção por Amostragem

Para garantir que as árvores na população sejam todas distintas, o algoritmo usa uma tabela de dispersão para manter a população. Nessa tabela, cada árvore t é uma chave à qual são associados um valor $g(t)$, indicando seu erro no conjunto de treinamento, e um valor $s(t)$, indicando seu tamanho (i.e., número de nós). Isso evita o reprocessamento de árvores idênticas, o que seria um desperdício de recursos computacionais, e contribui para aumentar a diversidade da população, o que é um benefício. A cada nova geração, a população corrente é transformada em uma nova população, usando operadores genéticos de cruzamento e mutação.

A técnica tradicionalmente usada para selecionar os indivíduos a serem submetidos às operações genéticas é a seleção proporcional à aptidão (conhecida como *roleta russa*) [1]. Essa técnica, cuja ideia é ilustrada na Figura 9, garante que os indivíduos com menores erros tenham maiores chances de serem escolhidos.

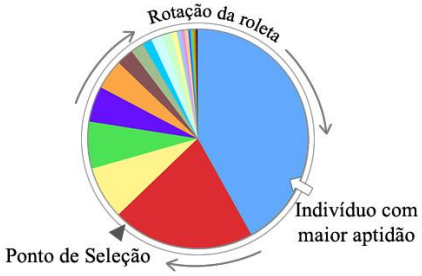

Figura 9 - Método de seleção roleta russa

Nesse trabalho, uma nova estratégia de seleção, baseada em amostragem, foi adotada. Essa estratégia foi inspirada na observação de que a combinação dos indivíduos mais aptos (i.e., com os menores erros) nem sempre gera filhos com os menores tamanhos. Isso acontece porque, à medida que a população evolui, todos os indivíduos têm seus erros gradativamente diminuídos; porém, aqueles com menores erros não necessariamente têm os menores tamanhos. O problema é que, durante a avaliação dos indivíduos, o tamanho das árvores é usado quase que exclusivamente como critério de desempate. Por exemplo, se uma árvore A_1 tem erro 0.08 e tamanho 30 e uma árvore A_2 tem erro 0.08 e tamanho 15, então a árvore A_2 é considerada mais apta que a árvore A_1. Por outro lado, se uma árvore A_3 tem erro 0.08 e tamanho 50 e uma árvore A_4 tem erro 0.09 e tamanho 15, então a árvore A_3 é considerada mais apta que a árvore A_4 (mesmo que a diferença nos erros seja muito pequena e a diferença nos tamanhos seja muito grande).

Assim, quando a população é ordenada em função da aptidão (erro como critério principal e tamanho como critério secundário), os indivíduos no início da sequência ordenada podem ter erros pequenos e tamanhos muito grandes (i.e., podem estar superajustados). Por outro lado, indivíduos no final dessa sequência podem ter tamanhos muito pequenos e erros não muito maiores que aqueles dos indivíduos no início da sequência. Como as características dos indivíduos selecionados são transferidas para seus filhos, o cruzamento entre indivíduos menos aptos pode resultar em indivíduos com excelente desempenho.

O processo de amostragem proposto ocorre da seguinte maneira: a população ordenada é segmentada em quatro partes e cada uma delas é combinada com as demais, resultando nas seguintes combinações (1,2), (1,3), (1,4), (2,3), (2,4), (3,4). Então, durante a fase de amostragem, cada combinação (x, y) é avaliada do seguinte modo: escolhe-se um indivíduo no segmento x e outro no segmento y para gerar um filho; esse procedimento é repetido até que o conjunto de filhos gerados represente 5% do tamanho da população (*amostragem*). Em seguida, calcula-se média de desempenho dos indivíduos gerados para cada combinação. Então, adota-se a combinação que produziu a melhor média como padrão para a geração dos demais indivíduos da população (os indivíduos gerados durante a fase de amostragem também são aproveitados para compor a próxima geração, evitando o desperdício do esforço computacional na fase de amostragem).

3.6. O Sistema Desenvolvido

Com base nas ideias discutidas nas seções anteriores, um sistema denominado SRS (*Symbolic Regression with Sampling*) foi desenvolvido em Python [5]. A escolha da linguagem foi baseada no bom desempenho e facilidade em manipular árvores, expressões *lambda* (necessárias para avaliação das hipóteses que são geradas em tempo de execução) e tabelas de dispersão. O sistema SRS possui duas funções principais: a primeira delas gera uma população inicial, simula sua evolução no decorrer do tempo, aplicando operadores de cruzamento e mutação, e, no final, devolve a melhor hipótese (com relação ao conjunto de treinamento); a segunda delas avalia essa melhor hipótese

(com relação ao conjunto de validação) e reporta algumas estatísticas correspondentes.

4. Resultados Empíricos

Nessa seção são descritos os experimentos feitos com duas versões do sistema SRS: uma que não usa a estratégia de amostragem e outra que usa essa estratégia. O objetivo do experimento foi verificar o impacto do uso de amostragem no tempo necessário para evolução da população e obtenção dos melhores modelos. Todos os resultados foram obtidos com uma população de 4000 árvores e 10 rodadas, cada uma delas com no máximo 200 gerações para cada conjunto de treinamento.

4.1. Conjuntos de Dados

Os conjuntos de dados usados nos experimentos foram obtidos com o uso de uma função geradora do próprio sistema. Vale ressaltar que, embora o programa gerador conheça a função $f(x)$, essa função é desconhecida para o programa evolutivo que gera as aproximações $h(x)$. As funções $f(x)$ usadas nos experimentos são polinômios com grau variando entre 2 e 6.

Os conjuntos de dados têm tamanhos variando entre 20 e 2000 objetos, dos quais 70% são usados para treinamento e 30% são usados para validação.

Durante a geração dos dados, as entradas para função $f(x)$ foram escolhidas no intervalo $[-1,1]$, de duas formas distintas: distribuição uniforme e distribuição aleatória. Na distribuição uniforme, o intervalo é segmentado uniformemente de acordo com quantidade de entradas esperadas e cada ponto separando dois segmentos é usado como uma entrada (a distância entre dois pontos consecutivos quaisquer é constante). Por outro lado, na distribuição aleatória, os pontos usados como entradas são escolhidos aleatoriamente no intervalo (a distância entre dois pontos consecutivos quaisquer é variável).

4.2. Discussão dos Resultados

Como algoritmos evolucionários são estocásticos [6], para cada função foram feitas 10 rodadas e, dos resultados obtidos, o pior e o melhor casos foram desconsiderados no cálculo da média (Tabela I). Isso evita que casos excepcionais influenciem na avaliação.

Os resultados mostraram que o SRS possui uma ótima habilidade em encontrar boas aproximações para vários tipos de funções, sem que seja suposto um tipo ou forma. Essas aproximações podem dar base para o trabalho dos especialistas habituados com a regressão tradicional. Ademais, a inclusão da seleção por amostragem resultou em uma melhora no resultado final na maioria dos casos (i.e., melhor desempenho em menor tempo).

A comparação dos resultados obtidos confirmou a conjectura de que o uso de amostragem pode acelerar a evolução da população, permitindo obter melhores aproximações em menor tempo.

Tabela I – Resultados dos experimentos.

Função do Conjunto de Dados	***Exemplos de Treinamento + Validação***	***Intervalo entre as entradas***	***Seleção com Amostragem***		***Seleção Aleatória***	
			Tempo	***Erro***	***Tempo***	***Erro***
x^3+x^2+x	14+6	Uniforme	2.86	0.00	6.39	0.00
$x^4+x^3+x^2+x$	14+6	Uniforme	4.48	0.00	10.36	0.00
$x^5+x^4+x^3+x^2+x$	14+6	Uniforme	85.83	0.16	32.36	0.00
$x^6+x^5+x^4+x^3+x^2+x$	14+6	Uniforme	14.31	0.00	25.25	0.00
x^3+x^2+x	140+60	Uniforme	10.61	0.00	18.89	0.00
$x^4+x^3+x^2+x$	140+60	Uniforme	15.08	0.00	25.31	0.00
$x^5+x^4+x^3+x^2+x$	140+60	Uniforme	21.99	0.00	40.08	0.00
$x^6+x^5+x^4+x^3+x^2+x$	140+60	Uniforme	26.21	0.00	56.54	0.00
x^3+x^2+x	1400+600	Uniforme	70.76	0.00	102.09	0.00
$x^4+x^3+x^2+x$	1400+600	Uniforme	112.35	0.00	207.98	0.00
$x^5+x^4+x^3+x^2+x$	1400+600	Uniforme	124.16	0.00	244.70	0.00
$x^6+x^5+x^4+x^3+x^2+x$	1400+600	Uniforme	163.89	0.00	400.79	0.00
x^3+x^2+x	14+6	Aleatório	3.33	0.00	6.76	0.00
$x^4+x^3+x^2+x$	14+6	Aleatório	6.04	0.00	10.99	0.00
$x^5+x^4+x^3+x^2+x$	14+6	Aleatório	9.74	0.00	21.54	0.00
$x^6+x^5+x^4+x^3+x^2+x$	14+6	Aleatório	13.16	0.00	34.16	0.00
x^3+x^2+x	140+60	Aleatório	8.76	0.00	12.46	0.00
$x^4+x^3+x^2+x$	140+60	Aleatório	14.84	0.00	26.68	0.00
$x^5+x^4+x^3+x^2+x$	140+60	Aleatório	21.04	0.00	39.31	0.00
$x^6+x^5+x^4+x^3+x^2+x$	140+60	Aleatório	26.84	0.00	57.88	0.00
x^3+x^2+x	1400+600	Aleatório	57.75	0.00	145.33	0.00
$x^4+x^3+x^2+x$	1400+600	Aleatório	95.98	0.00	191.13	0.00
$x^5+x^4+x^3+x^2+x$	1400+600	Aleatório	138.06	0.00	298.11	0.00
$x^6+x^5+x^4+x^3+x^2+x$	1400+600	Aleatório	183.81	0.00	677.70	0.00
$x^4+x^3+x^2+x+1$	14+6	Uniforme	136.55	0.00	37.13	0.00
$2x^5-3x^4+4x^3-5x^2+6x-7$	21+9	Uniforme	1288.97	0.26	1288.23	0.08
$2x^5-3x^4+4x^3-5x^2+6x-7$	70+30	Uniforme	1224.48	0.07	1285.60	0.10

5. Conclusões

A regressão consiste em encontrar a relação entre duas variáveis (uma entrada x e uma saída y) e expressar essa relação por meio de uma função. A abordagem tradicional da regressão adota um modelo específico de função (e.g., $ax^2 + bx + c$) e tenta encontrar coeficientes que ajustem esse modelo aos dados de treinamento. O problema é que, quando esse modelo é desconhecido, o método tradicional não pode ser aplicado. Nesse caso, o uso de regressão simbólica é uma alternativa mais apropriada, pois ela é capaz de encontrar a função que se ajusta aos dados de treinamento, mesmo sem a adoção de um modelo de função específico.

Neste artigo, um novo algoritmo evolucionário para a regressão simbólica foi proposto e um sistema correspondente chamado SRS foi desenvolvido. A principal inovação neste algoritmo está no uso de amostragem para a seleção de indivíduos da população que irão produzir descendentes para compor a próxima geração da população. A conjectura é que o uso de amostragem pode acelerar a evolução da população e reduzir o tempo necessário para obtenção do melhor modelo de regressão. Os resultados dos experimentos feitos com o sistema que foi implementado confirmaram essa conjectura.

Diferentemente de abordagens determinísticas, algoritmos evolucionários podem obter melhores resultados conforme a disponibilidade de tempo (ou seja, quanto mais tempo disponível para a evolução da população, maior a probabilidade de se encontrar um modelo ótimo). Assim, o principal objetivo deste trabalho foi propor um algoritmo evolucionário que pudesse encontrar modelos ótimos em menor tempo.

Evidentemente, modelos mais complexos podem ter múltiplas entradas e envolver outros tipos de operadores (e.g., funções trigonométricas e logarítmicas). Assim, uma extensão natural desse trabalho consiste em alterar a implementação o sistema SRS para lidar com outros tipos de funções (além das polinomiais). Outra extensão interessante é implementar simplificação de expressões (expressões mais simples são representadas por árvores mais simples). Isso poderia reduzir do tempo necessário para avaliação das expressões e aumentar o desempenho do sistema.

Agradecimentos

Ao CNPq pela bolsa de Iniciação Científica[1] (Processo 151579/2014-8) e pela bolsa de Produtividade em Pesquisa[2] (Processo 305484/2012-5).

Referências Bibliográficas

[1] J. R. Koza. **Genetic Programming**, MIT Press, London, 1998.

[2] D. Whitley. **An Overview of Evolutionary Algorithms: Practical Issues and Common Pitfalls**, Information and Software Technology, vol. 43, p. 817–831, 2001.

[3] Morgon, R. and Pereira, S. **Evolutionary Learning of Concepts**, Journal of Computer and Communications, 2, 76-86 (2014).

[4] S. B. Kotsiants; I. D. Zaharakis; P. E. Pintelas. **Machine Learning: A Review of Classification and Combining Techniques**, Artificial Intelligence Review, vol. 26:3, p. 159–190, 2006.

[5] V. L. Ceder. **The Quick Python Book**, 2nd edition, Manning Publications Co., USA, 2010.

[6] T. Weise. **Global Optimization Algorithms: Theory and Application**, 2nd edition, 2008.