Programação Genética e uma aplicação em Regressão Simbólica

Carlo

18 de dezembro de 2019

Evolução e seus mecanismos: Evolução

Na primeira parte da apresentação vamos utilizar o termo **DNA** para representar uma sequência finita qualquer de 0's e 1's. Ou seja, um elemento do espaço $X = \cup_{n=1}^{\infty} \{0,1\}^n$.

Uma **população** no tempo n, P_n é uma coleção não vazia de indivíduos. Cada indivíduo é representado por um DNA.

Exemplo de população:

$$P_n = \{\ (0,1,1,0,1),\ (1,1),\ (1,0,0),\ (0,1,1),\ (1,0,0),(0,0,1,1,0)\ \}.$$

Evolução

*Chamamos de **evolução** no tempo n uma mudança na proporção da composição genética (DNA) de uma população.*

Evolução e seus mecanismos: Fatores Evolutivos

Fatores Evolutivos

- ***Mutação***: *qualquer mudança no DNA, gerando a variabilidade genética em caracteres que afetam as chances de sobrevivência e reprodução.*
- ***Seleção Natural***: *processo pelo qual os indivíduos mais bem-adaptados ao ambiente aumentam em frequência relativamente aos indivíduos menos bem-adaptados, ao longo de várias gerações.*
- ***Deriva Genética***: *mudanças aleatórias nas frequências gênicas de uma população.*
- ***Fluxo Gênico***: *movimentação de genes para uma população, através do intercruzamento ou por migração.*

Evolução e seus mecanismos: Mutações

Mutações ocorrem no nascimento de novos indivíduos de uma população. São alterações aleatórias do DNA e geraram variabilidade genética.

Exemplo de mutação (pontual) em uma população:

$$P_n = \{ (1,1,0), (1,1,0) \} \rightarrow P_{n+1} = \{ (0,1,0), (1,1,0) \}.$$

Exemplo de mutação (duplicação) em uma população:

$$P_n = \{ (1,1,0), (1,1,0) \} \rightarrow P_{n+1} = \{ (1,1,1,0), (1,1,0) \}.$$

Exemplo de mutação (inserção) em uma população:

$$P_n = \{ (1,1,0), (1,1,0) \} \rightarrow P_{n+1} = \{ (1,0,1,0), (1,1,0) \}.$$

Evolução e seus mecanismos: Seleção Natural

Pressupostos da Seleção Natural

- *Indivíduos da espécie podem ser diferentes entre si e uma parte desta variação pode ser passada para a prole (herdável).*
- *A cada geração nascem mais indivíduos do que podem sobreviver (limitação de recursos). Já que nascem mais indivíduos do que podem suportar os recursos disponíveis, deve haver uma luta pela existência entre eles, resultando na sobrevivência de apenas parte da população de cada geração.*
- *Sobrevivência e reprodução não são ao acaso: alguns indivíduos apresentam características mais favoráveis naquele contexto ambiental. Estes serão selecionados.*

Evolução e seus mecanismos: Seleção Natural

Seleção Natural

- *Processo pelo qual os indivíduos de uma população, mais bem-adaptados ao ambiente, aumentam em frequência relativamente aos indivíduos menos bem-adaptados, ao longo de uma série de gerações.*
- *A seleção natural não ocorre para que a população fique mais adaptada: ela simplesmente ocorre e como consequência a população fica mais adaptada.*

Evolução e seus mecanismos: Fitness

Valor adaptativo ou aptidão (fitness)

- *Contribuição média de um genótipo (composição genética de um indíviduo) para a próxima geração, quando comparada com outros genótipos.*
- *Na natureza, relacionado à capacidade de acasalamento, fertilidade, fecundidade e sobrevivência.*
- *Genótipos com aptidão alta têm maior chance de que seus genes passem à geração seguinte.*
- *Matematicamente, é uma função que atribui um valor para cada indivíduo do espaço de indivíduos, isto é, $f : X \to \mathbb{R}$. Ela é a nossa função para determinar se um indíviduo é melhor que o outro.*

Evolução e seus mecanismos: Reprodução

Dada um população P_n, queremos construir a população P_{n+1}. A criação de novos indíviduos a partir de antigos pode se dar de diferentes formas:

- Na reprodução assexuada, aplicamos apenas a mutação.
- Na reprodução sexuada pegamos dois indivíduos de P_n e antes de mutações fazemos o crossing-over.

Evolução e seus mecanismos: Reprodução

Crossing-over

Selecionados dois DNA's dos pais $x_1, x_2 \in P_n$. Sem perda de generalidade $len(x_1) \leq len(x_2)$. Escolhe-se um valor T entre 0 e $len(x_1)$. O filho é criado então tal que os T primeiros genes são iguais aos de x_1 e os demais $(len(x_2) - T)$ últimos dígitos são iguais aos de x_2.

Exemplo de crossing-over para $T = 3$:

$$\{ x_1 = (0,0,0,0,0,0,0),\ x_2 = (1,1,1,1,1,1,1) \} \rightarrow (0,0,0,1,1,1,1).$$

Exemplo de crossing-over para $T = 1$:

$$\{ x_1 = (0,0),\ x_2 = (1,1,1,1) \} \rightarrow (0,1,1,1).$$

Algoritmo Genético:

Algoritmos genéticos são algoritmos de busca baseados na evolução. Combinamos a ideia de sobrevivência do mais apto com a aleatoriedade e variabilidade para explorar de forma inteligente o espaço de busca.

Em cada nova geração, um novo conjunto de criaturas (e seus respectivos DNAs) é criado usando pedaços dos indíviduos mais aptos da geração anterior.

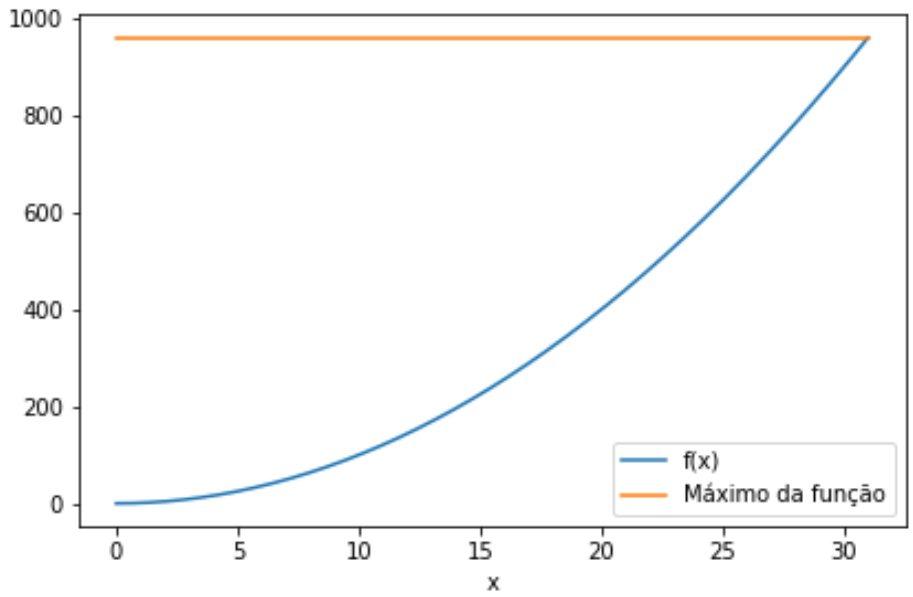
Para entender melhor o funcionamento de um algoritmo genético agora que já relembramos as principais estruturas da evolução natural vamos ver um exemplo prático.

Algoritmo Genético: Otimização

Suponha que queremos otimizar a função $f : [0, 31] \cap \mathbb{Z} \to \mathbb{R}$ dada por

$$f(x) = x^2,$$

utilizando algoritmos genéticos.



Mas como vamos representar um elemento de $[0, 31] \cap \mathbb{Z}$ como um DNA da forma estruturada em uma lista de 0's e 1's?

Algoritmo Genético: Otimização

Utilizando a representação ninária dos números por exemplo. Fixaremos que os elementos da nossa população serão sequências de 5 elementos apenas. E para cada elemento da nossa população $v \in \{0,1\}^5$, associaremos o inteiro:

$$x(v) = 1v[0] + 2v[1] + 4v[2] + 8v[3] + 16v[4].$$

Por exemplo, para $v = (1,1,0,1,0)$ temos $x(v) = 1 + 2 + 8 = 11$.

Precisamos agora usar uma função de fitness, como discutido, elementos com mais fitness serão selecionados e gerarão mais filhos parecidos com eles (idealmente, com fitness alto).

Podemos usar a própria função $f(x)$ neste caso, ela é a função que queremos otimizar.

Algoritmo Genético: Otimização

Aqui vamos fazer uma reprodução assexuada. Mas falta ainda explicar como vamos fazer para selecionar os indíviduos proporcionalmente ao seu fitness. Uma ideia é calcular o vetor de fitness de todos e depois normalizar pela soma de todos os fitness para ter um vetor de probabilidade.

Exemplo do cálculo das probabilidades de ter filho para uma população:

$$
\begin{aligned}
&\{\, v_1 = (0,0,0,1,0),\ v_2 = (1,1,1,0,1) \,\} \rightarrow \\
&\rightarrow \{\, x(v_1) = 8, x(v_2) = 23 \,\} \rightarrow \\
&\rightarrow \{\, f(x(v_1)) = 64, f(x(v_2)) = 529 \,\} \rightarrow \\
&\rightarrow \{\, \mathbb{P}(v_1 \text{ ser escolhido para ter filho}) = 64/(529+64), \\
&\mathbb{P}(v_2 \text{ ser escolhido para ter filho}) = 529/(529+64) \,\}
\end{aligned}
$$

Por fim, para a mutação, vamos fazer apenas as mutações pontuais. Modelamos com variáveis aleatórias do tipo $\mathrm{Ber}(p)$. Se ela der 0, então não fazemos mutação. Caso ela dê 1, então para cada elemento do vetor jogamos outra moeda $\mathrm{Ber}(q)$ e caso dê 1 trocamos o bit daquela coordenada.
Exemplo do cálculo das probabilidades de ter filho para uma população:

$$
\begin{aligned}
&\text{vetor inicial: } (0,0,0,1,0) \rightarrow \\
&\rightarrow \text{ tiro 1 na primeira (entro em mutação)} \rightarrow \\
&\rightarrow \text{tiro 0 na coordenada 1: } (0,0,0,1,0) \rightarrow \\
&\rightarrow \text{tiro 1 na coordenada 2: } (0,1,0,1,0) \rightarrow \\
&\rightarrow \text{tiro 1 na coordenada 3: } (0,1,1,1,0) \rightarrow \\
&\rightarrow \text{tiro 1 na coordenada 4: } (0,1,1,1,0) \rightarrow \\
&\rightarrow \text{tiro 0 na coordenada 5: } (0,1,1,1,0) \rightarrow \\
&\text{vetor final: } (0,1,1,1,0).
\end{aligned}
$$

Acho que agora estamos prontos. Não discutimos critérios de parada, mas vamos fazer até um valor fixado de gerações, por exemplo 15.

Algoritmo Genético: Otimização

```
POPULAÇÃO INICIAL
indivíduo: [1 0 0 0 0], valor: 1, fitness: f(x)=1, probabilidade: 0.029411764705882353
indivíduo: [0 1 0 0 0], valor: 2, fitness: f(x)=4, probabilidade: 0.11764705882352941
indivíduo: [0 1 0 0 0], valor: 2, fitness: f(x)=4, probabilidade: 0.11764705882352941
indivíduo: [0 0 0 0 0], valor: 0, fitness: f(x)=0, probabilidade: 0.0
indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.7352941176470589
indivíduo: [0 0 0 0 0], valor: 0, fitness: f(x)=0, probabilidade: 0.0
_____________________________________________________________________

GERAÇÃO 2
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
_____________________________________________________________________

GERAÇÃO 3
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.16666666666666666
```

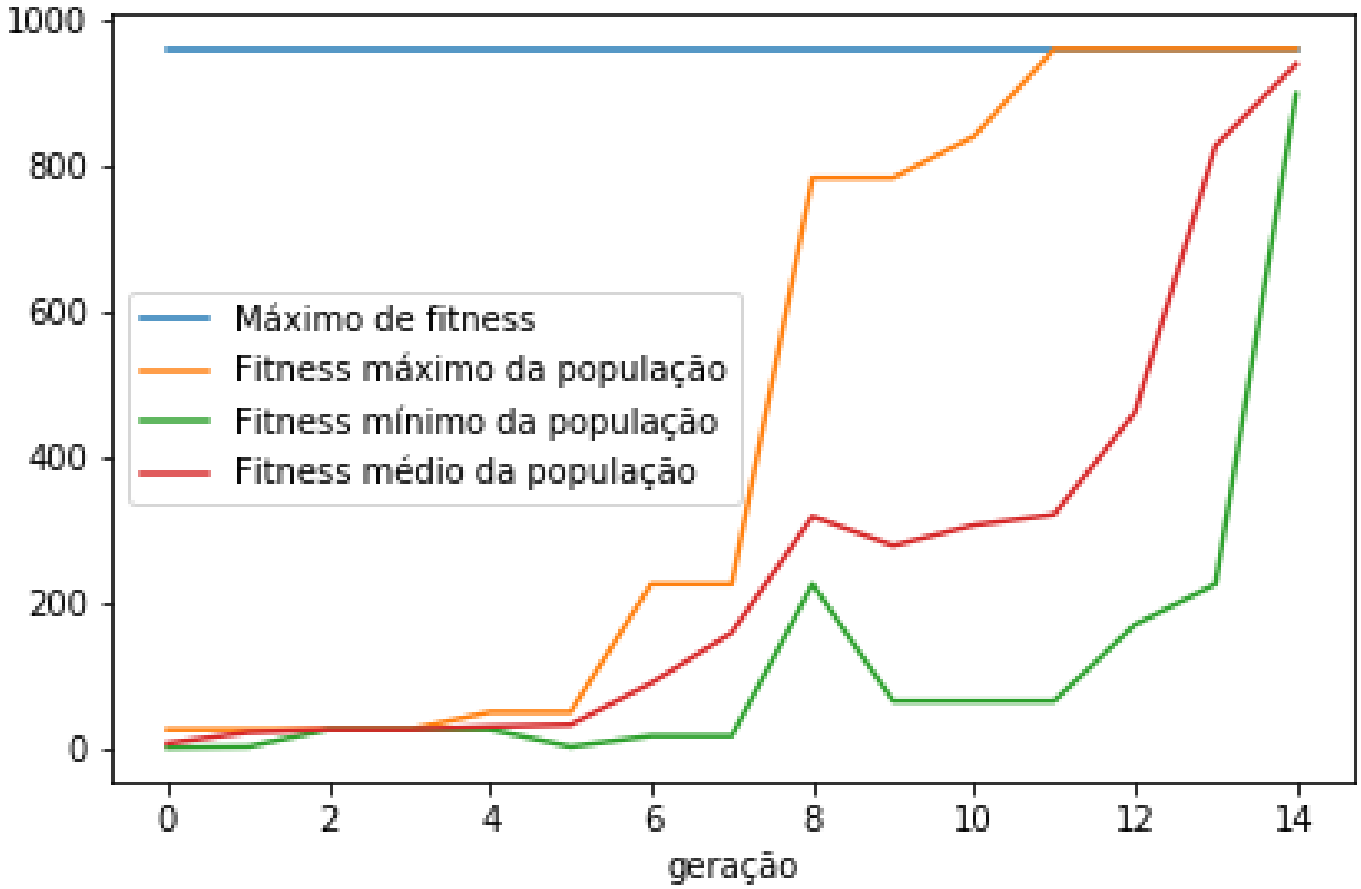
Algoritmo Genético: Otimização

```
GERAÇÃO 4
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.14367816091954022
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.14367816091954022
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.14367816091954022
pai: [1 0 1 0 0], indivíduo: [1 1 1 0 0], valor: 7, fitness: f(x)=49, probabilidade: 0.28160919540229884
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.14367816091954022
pai: [1 0 1 0 0], indivíduo: [1 0 1 0 0], valor: 5, fitness: f(x)=25, probabilidade: 0.14367816091954022
_____________________________________________________________________

GERAÇÃO 8
pai: [0 0 1 1 0], indivíduo: [0 0 1 1 1], valor: 28, fitness: f(x)=784, probabilidade: 0.4106862231534835
pai: [1 1 1 1 0], indivíduo: [1 1 1 1 0], valor: 15, fitness: f(x)=225, probabilidade: 0.1178627553693033
pai: [1 1 1 1 0], indivíduo: [1 1 1 1 0], valor: 15, fitness: f(x)=225, probabilidade: 0.1178627553693033
pai: [1 1 1 1 0], indivíduo: [1 1 1 1 0], valor: 15, fitness: f(x)=225, probabilidade: 0.1178627553693033
pai: [1 1 1 1 0], indivíduo: [1 1 1 1 0], valor: 15, fitness: f(x)=225, probabilidade: 0.1178627553693033
pai: [1 1 1 1 0], indivíduo: [1 1 1 1 0], valor: 15, fitness: f(x)=225, probabilidade: 0.1178627553693033
_____________________________________________________________________

POPULAÇÃO FINAL
pai: [1 1 1 1 1], indivíduo: [1 1 1 1 1], valor: 31, fitness: f(x)=961, probabilidade: 0.17026931254429484
pai: [1 1 1 1 1], indivíduo: [1 1 1 1 1], valor: 31, fitness: f(x)=961, probabilidade: 0.17026931254429484
pai: [0 1 1 1 1], indivíduo: [0 1 1 1 1], valor: 30, fitness: f(x)=900, probabilidade: 0.15946137491141035
pai: [1 1 1 1 1], indivíduo: [1 1 1 1 1], valor: 31, fitness: f(x)=961, probabilidade: 0.17026931254429484
pai: [1 1 1 1 1], indivíduo: [1 1 1 1 1], valor: 31, fitness: f(x)=961, probabilidade: 0.17026931254429484
pai: [0 1 1 1 1], indivíduo: [0 1 1 1 1], valor: 30, fitness: f(x)=900, probabilidade: 0.15946137491141035
```

Algoritmo Genético: Otimização



Hiperparâmetros relevantes: número de indivíduos na população, probabilidade de mutação e probabilidade de mutação em cada coordenada.

Uma aplicação em Aprendizado de Máquina: Regressão

Dado um conjunto de pontos $\mathcal{D}_{\text{train}} = \{(X_i, y_i)\}_{i=1}^{N}$ e $\mathcal{D}_{\text{test}} = \{(X_i, y_i)\}_{i=1}^{M}$ com $X_i \in \mathbb{R}^n$ e $y \in \mathbb{R}$. Seja $\mathcal{C}$ o espaço de funções de $\mathbb{R}^n$ para $\mathbb{R}$.

O problema de regressão pode ser formalizado como encontrar $g \in \mathcal{C}$ tal que minimizamos alguma métrica de interesse nos dados de treinamento, por exemplo MAE. Queremos

$$g = \arg\min_{h \in \mathcal{C}} \frac{1}{N} \sum_{(X_i, y_i) \in \mathcal{D}_{\text{train}}} |h(X_i) - y_i|.$$

Espera-se que quando g faz um bom trabalho em $\mathcal{D}_{\text{train}}$, teremos g tendo um bom score em $\mathcal{D}_{\text{test}}$ também. Isso nem sempre é válido, pois podemos estar tendo apenas um overfit e g é boa no treinamento, mas não generaliza bem.

Uma aplicação em Aprendizado de Máquina: Trade-off viés-variância

Por esse motivo, é razoável restringir o espaço de busca $\mathcal{C}$. Quando escolhemos algum modelo para regressão, por exemplo a regressão linear, estamos impondo que apenas as funções lineares são candidatas. Neste caso, por exemplo, estamos olhando apenas as funções escritas como $g(X) = \beta\dot{X} + \beta_0$ para $\beta \in \mathbb{R}^n$ e $\beta_0 \in \mathbb{R}$.

Entretanto isso tem seu lado negativo. Essa restrição no espaço de busca pode fazer com que nossos modelos não capturem as informações disponíveis nos dados e sofra de underfit.

Por isso é interessante manter esse espaço amplo o suficiente pra conseguir encontrar aproximações interessantes sem deixar ele muito grande e poder apenas decorar os dados.

Num extremo, quando nosso espaço de busca é muito restrito temos coisas como regressões lineares. No outro extremo, quando nosso espaço é amplo e o modelo se adapta bem aos dados temos os vários métodos de ensemble de árvores, redes neurais etc.

Uma aplicação em Aprendizado de Máquina: Trade-off interpretabilidade-complexidade

Em muitas aplicações, pode ser relevante entender explicitamente o funcionamento do nosso modelo. *Modelos black box* são importantes, mas podem estar capturando preconceitos e levando em conta informações que não queremos que sejam usadas para o nosso modelo.

Se o objetivo de um determinado problema é apenas obter a aproximação numérica dessa função, um desses modelos não-lineares pode bastar. Mas, se houver necessidade de entender, inspecionar ou mesmo complementar o modelo obtido então essa aproximação se torna impraticável. Queremos minimizar o erro, mas fazendo uma espécie de regularização tentando manter o modelo interpretável.

Uma aplicação em Aprendizado de Máquina: Regressão Simbólica

No contexto de tentar conteplar modelos mais complexos que ainda podem ser interpretáveis temos a **Regressão Simbólica**.

O problema aqui é o mesmo da regressão. Dado um conjunto de pontos $\mathcal{D} = \{(X_i, y_i)\}_{i=1}^{N}$ com $X_i \in \mathbb{R}^n$ e $y \in \mathbb{R}$. Queremos achar uma função $g : \mathbb{R}^n \to \mathbb{R}$ que minimize alguma métrica de validação.

A questão principal aqui é que queremos a forma explícita de g. Além disso. ela deve ser complexa o suficiente para aprender com os dados e simples o suficiente para ser interpretável.

Uma aplicação em Aprendizado de Máquina: Discutindo um pouco a estrutura de árvore

Vamos aplicar as técnicas vistas nos algoritmos genéticos. Nesse caso, os nossos indivíduos serão as funções. Estruturaremos o DNA não mais como uma lista de 0's e 1's, mas como uma árvore.

Cada nó da árvore pode representar um operador, uma função, uma variável ou uma constante. Um nó que representa um operador ou função precisa ter nós saindo dele para cada um dos parâmetros necessários. Enquanto as variáveis e constantes são nós folhas.

Uma aplicação em Aprendizado de Máquina: Regressão Simbólica

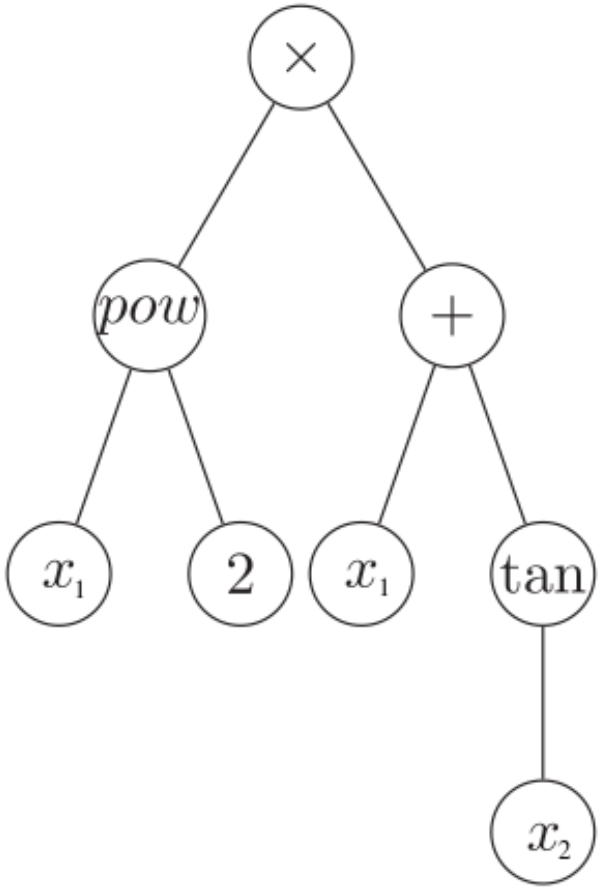
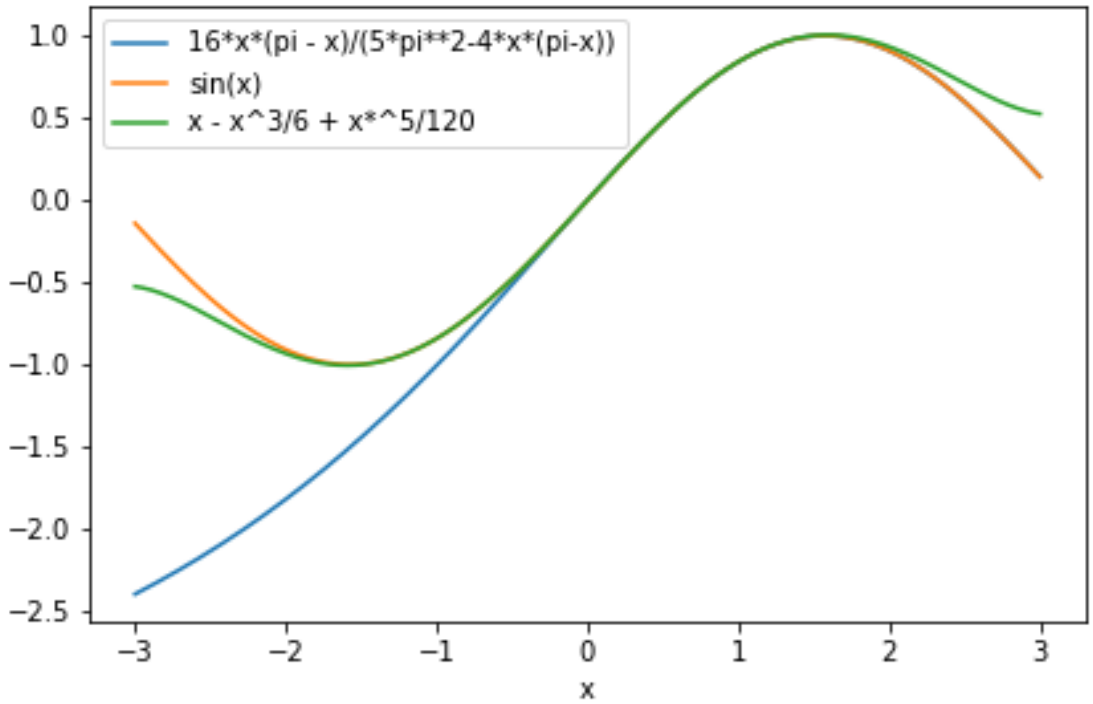


Figura: Exemplo de árvore representando a função $g(x) = x_1^2 (x_1 + \tan(x_2))$

Uma aplicação em Aprendizado de Máquina: O que significa ser mais simples?

O significado de mais simples no contexto da Regressão Simbólica está na facilidade de interpretação. Considere as funções:

$$\frac{16x(\pi - x)}{5\pi^2 - 4x(\pi - x)}, \ \sin(x), \text{ e } x - \frac{x^3}{3!} + \frac{x^5}{5!}.$$



Uma aplicação em Aprendizado de Máquina: O que significa ser mais simples?

Supondo que a função $\sin(x)$ seja o alvo, as duas primeiras funções retornam uma aproximação razoável dentro de um intervalo limitado dos valores x. Se tivermos o objetivo de interpretar o comportamento da função, em vez de apenas obter a aproximação numérica, a função target é a forma mais simples de entender. A simplicidade é frequentemente medida pelo tamanho da árvore de expressão ou o número de funções não-lineares usadas.

Uma aplicação em Aprendizado de Máquina: Fitness

Como falar então que uma função é melhor que a outra? Queremos definir uma função fitness f de uma forma inteligente. Uma abordagem pode ser, dado g função associada à uma árvore, calcularmos

$$f(g) = \frac{1}{1 + \mathrm{MAE}(g, \mathcal{D}) + \lambda\, \mathrm{Regularization}(g)},$$

em que $\mathrm{MAE}(g, \mathcal{D})$ é o erro médio absoluto da função g nos dados $\mathcal{D}$, λ é um fator de regularização e $\mathrm{Regularization}(g)$ é uma penalização associada ao número de nós ou número de operadores não-lineares, por exemplo.

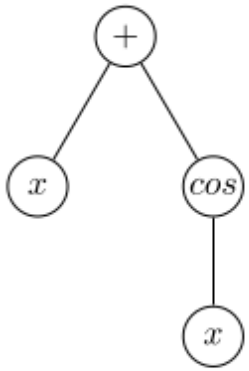
Uma aplicação em Aprendizado de Máquina: Reprodução

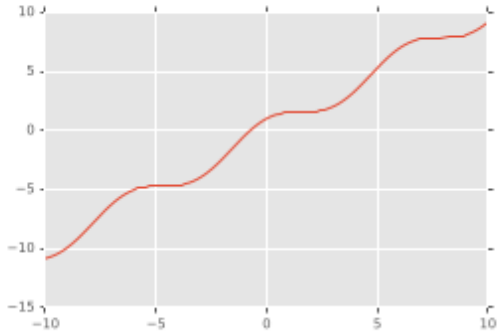
Selecionado o pai, uma forma de fazer a mutação é sorteando um dos nós (podendo colocar mais peso para os nós mais profundos) e o substituindo por um novo nó recriando a árvore a partir dele.

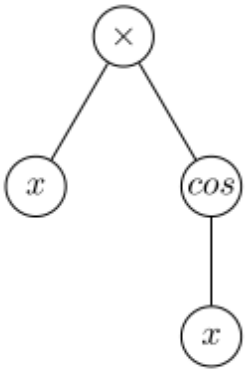
Podemos ainda substituir um dos nós por um nó equivalente, por exemplo trocar um operador $+$ por um operador $\times$ (os dois são operadores que precisam ser ligados a dois nós).

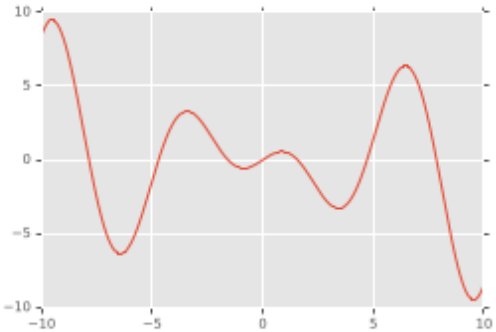
Uma aplicação em Aprendizado de Máquina: Reprodução

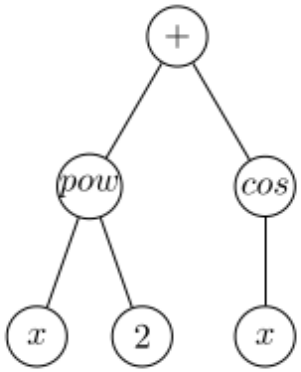
$$x + \cos(x),\ x\cos(x),\ \text{e}\ x^2 + \cos(x).$$

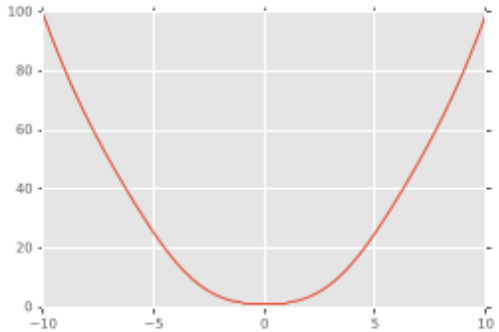












Uma aplicação em Aprendizado de Máquina: Fabrício

Uma forma mais inteligente de fazer essa reprodução pode ser limitando nosso espaço de busca para funções do tipo:

$$f(X) = \sum_j w_j \, g_j(X)$$

em que w_j é o coeficiente encontrado na regressão linear das variáveis transformadas a partir de g_j. A função g é uma composição $g(\cdot) = t(p(\cdot))$, para $t : \mathbb{R} \to \mathbb{R}$ e $p : \mathbb{R}^n \to \mathbb{R}$.

Limitamos que $g \in \mathcal{A}$, em que $\mathcal{A}$ é um conjunto de funções lineares e não-lineares. Por exemplo, podemos fazer

$$\mathcal{A} = \{id(x), \sin(x), \cos(x), \exp(x), \log(x)\}$$

Temos ainda que $p(X) = \prod_{i=1}^{n} x_i^{k_i}$, em que $k_i \in \mathbb{Z}$ é o expoente da variável i.

Uma aplicação em Aprendizado de Máquina: Fabrício

Como representar $f(x) = \sin(x)$ dessa forma? Faça por exemplo $w_1 = 1$, $t(z) = sin(z)$ e $p(x) = x$.

Agora suponha que $X \in \mathbb{R}^3$ e a função target é da forma

$$f(X) = 3.5 \sin(x_1^2 x_2) + 5 \log(x_2^3/x_1).$$

Podemos representá-la como

$$f(X) = 3.5 g_1(X) + 5 g_2(X),$$

para $t_1(z) = \sin(z)$, $p1(X) = x_1^2 x_2$, $t_2(z) = \log(z)$ e $p_2(X) = x_1^{-1} x_2^3$.

Entretanto, não é possível escrever coisas na forma $f(X) = \sin(x_1^2 + x_2)/x_3$.

Uma aplicação em Aprendizado de Máquina: Fabrício

Uma forma de representar as funções agora é sabendo as tuplas $h = (t, p)$ já que os coeficientes são determinados por uma regressão linear. Como para saber p só precisamos saber os valores inteiros dos expoentes podemos representá-lo como o vetor associado $[k_1, k_2, \cdots, k_n]$.

Por exemplo, para $n = 3$, a regressão linear simples

$$f(X) = w_1 x_1 + w_2 x_2 + w_3 x_3,$$

pode ser escrita como

$$\{h_1 = (id, [1, 0, 0]),\ h_2 = (id, [0, 1, 0],\ h_3 = (id, [0, 0, 1])\}.$$

Uma aplicação em Aprendizado de Máquina: Fabrício

Por exemplo, para $n = 3$, a regressão

$$f(X) = w_1 x_1^3 x_2 + w_2 \sin(x_3),$$

pode ser escrita como

$$\{h_1 = (id, [3, 1, 0]),\ h_2 = (\sin, [0, 0, 1])\}.$$

Uma aplicação em Aprendizado de Máquina: Fabrício

Neste caso, a reprodução se dá selecionando os melhores indivíduos da população a aplicando algumas transformações neles.

Dado um indivíduo escrito como $\{(t_1, p_1), (t_2, p_2), \cdots, (t_q, p_q)\}$. Fazemos todas as combinações $i, j \in \{1, 2, \cdots, q\}$construindo $p_i + p_j$ e $p_i - p_j$. Nesses novos candidatos eu aplico ainda todas as funções do conjunto $\mathcal{A}$.

Uma aplicação em Aprendizado de Máquina: Fabrício

Suponha, por exemplo, que começamos com a expressão anterior para regressão linear em $n = 3$ dimensões e que $\mathcal{A} = \{id(x), \log(x)\}$:

$$\{h_1 = (id, [1, 0, 0]),\ h_2 = (id, [0, 1, 0],\ h_3 = (id, [0, 0, 1])\}.$$

Construímos todas as combinações

$$p_4 = p_1 + p_1 = [2, 0, 0], p_5 = p_1 + p_2 = [1, 1, 0], p_6 = p_1 + p_3 = [1, 0, 1],$$
$$p_7 = p_2 + p_2 = [0, 2, 0], p_8 = p_2 + p_3 = [0, 1, 1], p_9 = p_3 + p_3 = [0, 0, 2].$$

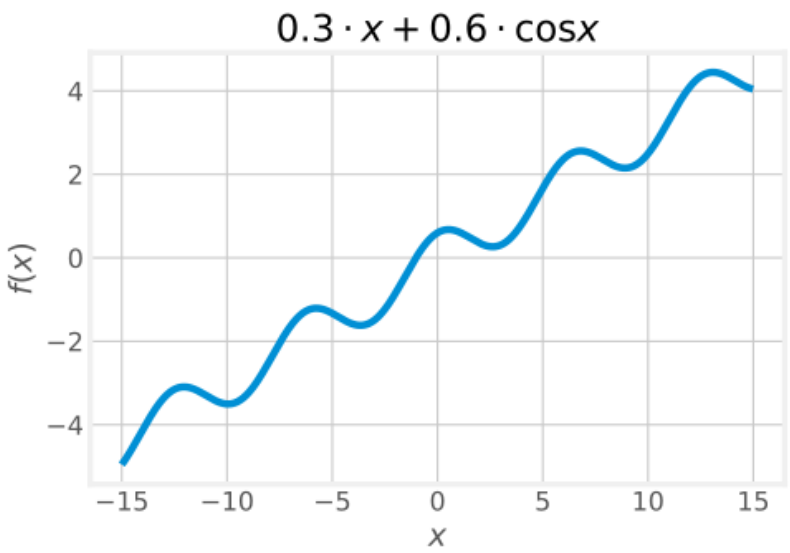
E fazemos as tuplas:

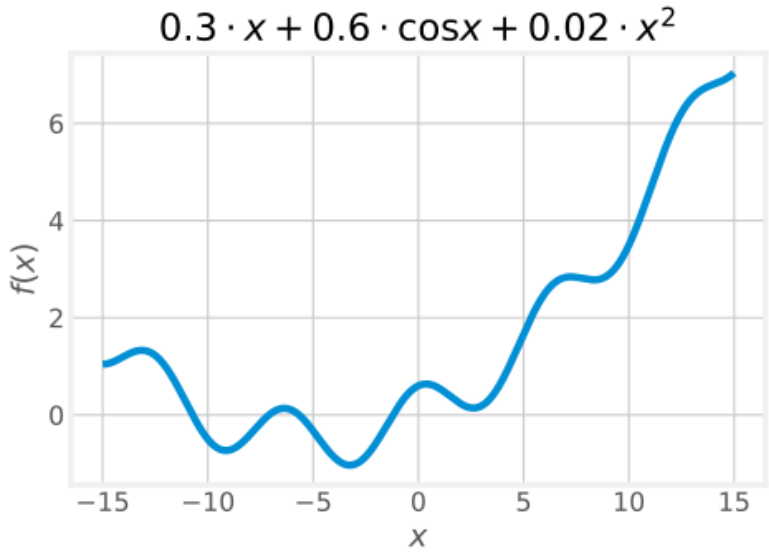
$$\{h_4 = (id, p_4), h_5 = (\log, p_4), h_6 = (id, p_5), h_7 = (\log, p_5), \cdots, h_{14} = (id, p_9),$$

Nesse novo conjunto de funções: $\{h_1, \cdots, h_{15}\}$ (**incluindo as funções do pai**), $g_j(\cdot) = t_j(p_j(\cdot))$ aplicamos uma regressão linear.

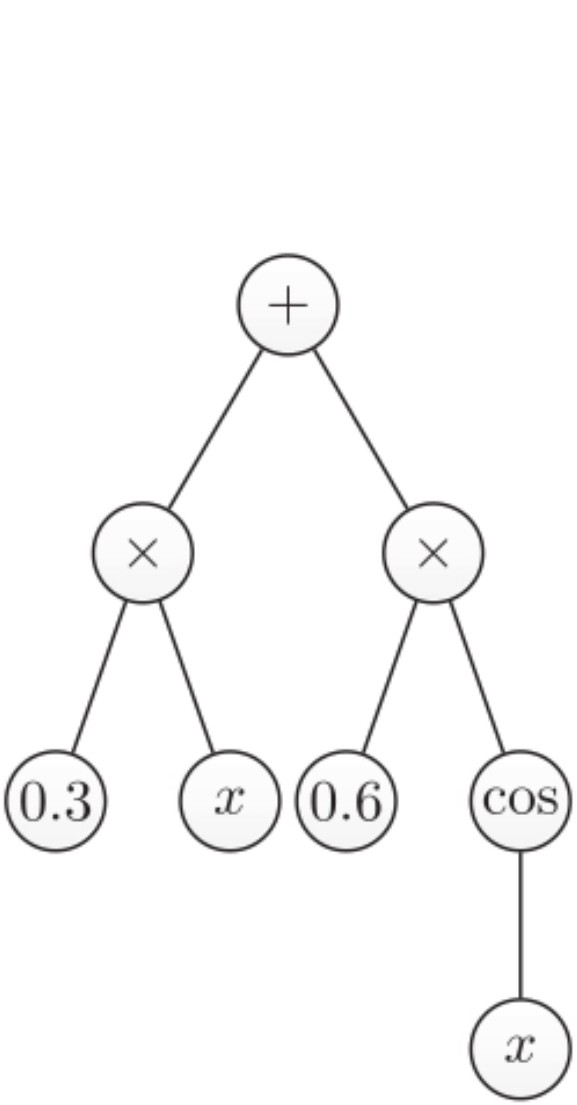
Uma aplicação em Aprendizado de Máquina: Fabrício

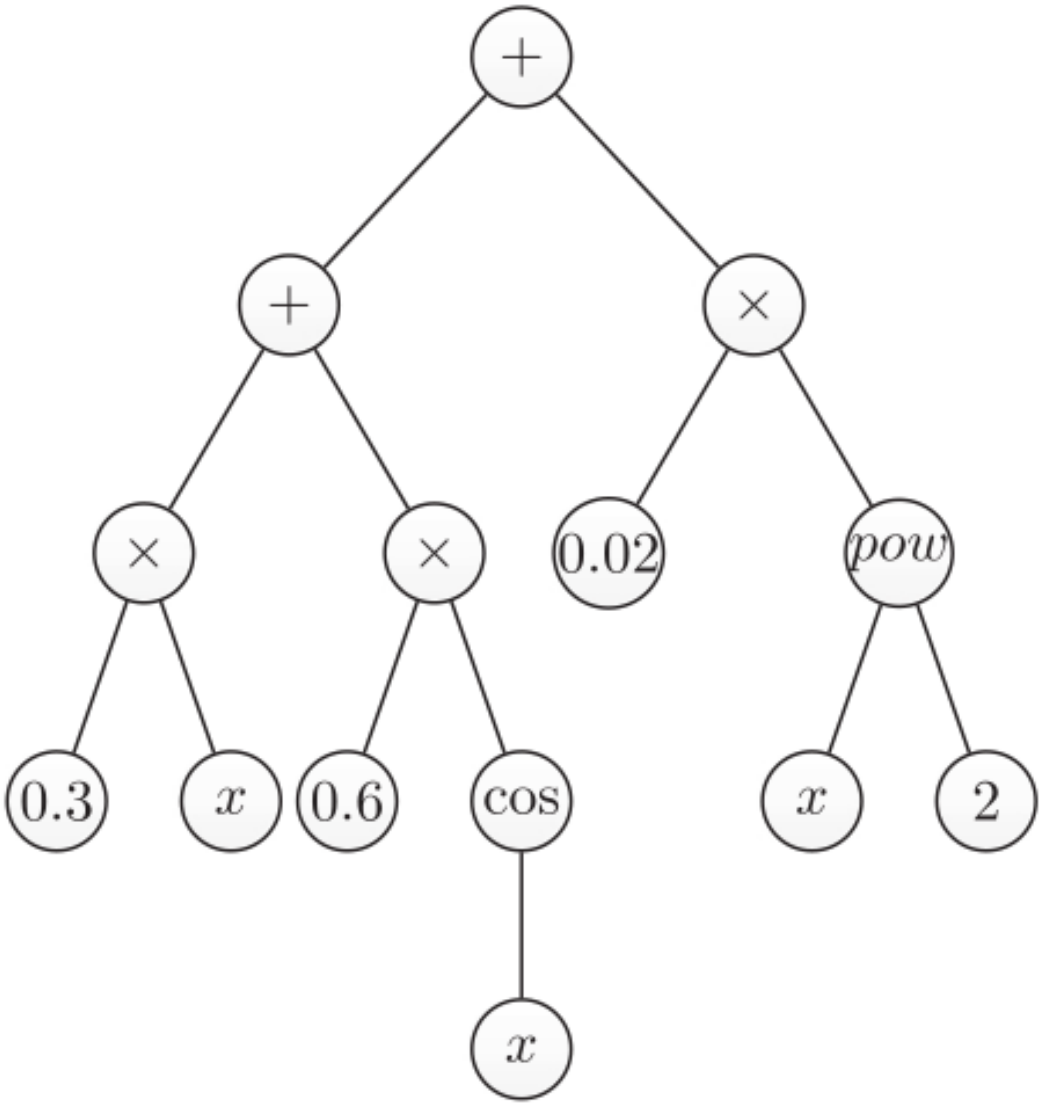
Os termos que produzem indeterminações do tipo log(0) ganham automaticamente peso associado 0. Pesos menores que um limiar fixado são descartados.
Filhos piores que os pais também não sobrevivem. Aplicamos o algoritmos genético nessa estrutura.





Uma aplicação em Aprendizado de Máquina: Fabrício





Referências

- Ricklefs, R. E. (2010). A economia da natureza. 6ª ed. Rio de Janeiro: Editora Guanabara Koogan. 572 p.
- França, F.O. (2018). A greedy search tree heuristic for symbolic regression. Information Sciences, 442-443, 18-32 (https://folivetti.github.io/files/InfoScienceITSR.pdf).
- Biblioteca gplearn para Regressão Simbólica. (https://gplearn.readthedocs.io/en/stable/reference.htmlsymbolic-regressor).