Symbolic Regression

Matheus Cândido Teixeira

1 Introdução

A regressão simbólica (RS) é utilizada para resolver o problema de *curve fitting*. Para isso, um conjunto de amostras é fornecido, e o resultado é uma função que possui o menor erro entre os pontos amostrados e o valor dela nesses pontos.

A regressão simbólica pode ser resolvida de diversas maneiras. Uma delas é utilizando programação genética (GP, do inglês *Genetic Programming*). A GP é semelhante ao Algoritmo Genético (GA, do inglês Genetic Algorithm) no que tange os operadores genéticos, pois ambos definem operadores de inicialização, seleção, cruzamento, mutação e *fitness*.

Na literatura, há diversas possiveis implementações dos operadores de GP. Por exemplo, na fase de geração de indivíduos, que podem ser gerados utilizando o método *full* ou *grow*. No primeiro método, o indivíduo é complemente gerado, isto é, todas os seus locus são preenchidos, enquanto que no último, não há essa necessidade. Na prática, é comum haver a combinação dos dois métodos, denominado *ramped half-and-half*, onde parte da população é gerada um dos método e o restante utilizando o outro. A seguir é apresentado as alternativas comuns para o desenvolvimento de cada operador.

Os operadores de seleção são os mesmos dos utilizados em GA: *roulette whell* e *k-tournament*. O primeiro seleciona o indivíduo com probabilidade proporcional a *fitness* do indivíduo, ou seja, se a fitness de um indivíduo for f_k em uma população com N indivíduos, a probabilidade dele ser selecionado é igual a $p(k) = f_k / \sum_{i=0}^{N} f_i$. O outro método é o *k-tournament*, que amostra k indivíduos aleatóriamente e seleciona o indivíduo com maior *fitness* nesse grupo. A diferença entre esses algoritmos está na pressão seletiva imposta aos indivíduos. Falar sobre menor pressão seletiva no começo e aumentar no final.

O operador de cruzamento (ou *crossover*) mais comum é denominado troca de sub-árvore. Esse operador funciona da seguinte maneira: dois indivíduos (I_1 e I_2, respectivamente) são selecionados da população utilizando o operador de seleção, após isso, para cada indivíduo, é escolhido um ponto aleatório (p_1 e p_2) e um novo indivíduo é gerado pela junção da árvore I_1 sem a sub-árvore com raíz no ponto p_1 com a sub-árvore extraida do I_2 com raíz em p_2.

No caso do operador de mutação há diversas alternativas, entre elas estam a mutação de um ponto e mutação de sub-árvore. O primeiro método, percorre todo o indivíduo muda o gene com uma probabilidade p_{op}. O segundo método, seleciona um ponto aleatório na árvore e, a partir desse ponto, uma sub-árvore é gerada aleatóriamente. Note que no primeiro método há duas probabilidades envolvidas: (1) a probabilidade de ocorrer mutação (p_m) e (2) a probabilidade de haver mutação em cada nó (p_{op}), caso o indivíduo tenha sido selecionado para mutação.

O último operador é o cálculo da *fitness*. Como a regressão simbólica busca minimizar o erro entre função gerada é os pontos amostrais, é comum utilizar o erro (a diferença entre o ponto e o resultado da função nesse ponto) como a forma de mensurar a adequação do indivíduo. Portanto, a *fitness* pode ser calculada como a somatória do erro absoluto (MAE), somatória do quadrado do erro (MSE) ou raíz quadrada da somatória do quadrado do erro (RMSE) entre a função gerado e

os pontos amostrais fornecidos, onde as equações são fornecidas a seguir:

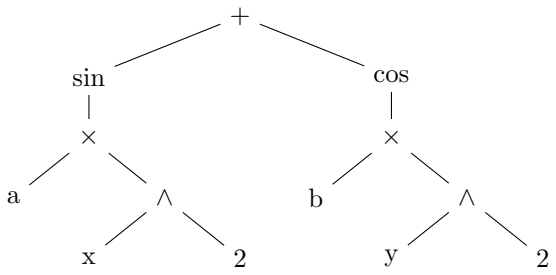
$$\text{MAE} = \sum_{i}^{N} |y - \hat{y}|$$

$$\text{MSE} = \sum_{i}^{N} (y - \hat{y})^2$$

$$\text{RMSE} = \sqrt{\sum_{i}^{N} (y - \hat{y})^2} = \sqrt{\text{MSE}}$$

Outro aspecto importanto em GP é a representação dos indivíduos, que podem ser representados linearmente ou em árvore. Ambas as representações possuem vantagens, porém é mais comum a implementação em árvore. Juntamente com a representação é importante definir os conjuntos de valores que eles podem assumir. A escolha do conjunto de funções e operadores devem atender a três restrições: Suficiência[1], Fechamento[2] e Parcimônia[3].

Por exemplo, para uma árvore com um conjunto de funções F: $\{\sin(\cdot), \cos(\cdot)\}$, com um conjunto de operadores $S : \{\times, +, -, \div\}$, uma possível árvore, cuja expressão é $\sin(ax^2) + \cos(by^2)$, onde a e b são constantes numéricas e x e y são variáveis independentes, é:



Neste trabalho, o GP é utilizado para resolver o problema da RS. Os detalhes e parâmetros da implementação são apresentados nas próximas seções. Para mensurar a eficiência, o algoritmo é aplicado a 3 dataset, dois do quais possuem apresentam versões com e sem ruídos aleatórios. O último dataset é um real e contém oito váriaveis aleatórias.

O restante deste relatório é dividido em 3 seções: (1) A seção de metodologia apresenta os detalhes e escolhas de implementação e design de projeto. (2) A seção de experimentos apresenta os resultados obtidos do treinamento do algoritmo nos datasets. (3) Por fim, a seção de conclusão analisa os resultados obtidos da seção de experimentos.

2 Metodologia

Nesta seção é descrito os datelhes de implementação e as decisões de design aplicadas neste projeto. A descrição e detalhes são apresentados na ordem: representação dos indivíduos, métodos de geração de indivíduos, métodos de seleção, operador de crossover, operador de mutação, eletismo e mecânismos de controle (ou de parada).

2.1 Reprentação dos indivíduos

A representação dos indivíduos impacta diretamente na performance do sistema e na expressividade dos invidívuos. Portanto, a princípio é apresentado como o genótipo do indivíduo é definido e como ele é codificado.

Os indivíduos em GP são representados por árvores sintáticas, que possuem nós e terminais. Os nós são operadores aritméticos ou funções e os terminais são constantes ou variáveis. O conjunto de nós e terminais juntos formam o *primitive set* do GP. Neste sistema o conjunto de funções são

[1] O conjunto de operadores deve ser capaz de representar uma solução apropriada.
[2] Os operadores devem suportar todos os resultados dos demais.
[3] O conjunto de operadores não deve conter elementos desnecessários.

as funções trigonométricas (sin, cos e tan) e o logaritmo com bases diferentes (log e $\log 10$). As constantes são geradas aleatóriamente utilizando uma distribuição multimodal. A escolha desse método se deve ao fato que muitas função matemáticas possuem contantes no intervalo $(0, 1]$ e com sinal variado. As variáveis são geradas aleatóriamente, onde um número inteiro no intervalo $(0, N]$, onde N deve ser menor ou igual ao número de colunas no dado de entrada, ou seja, se há 10 colunas de *features* e uma que representa a variável de resposta, então N deve ser igual ao número de *features* ou menor.

Outro aspecto importante na representação dos invíduos são, geralmente, representados utilizam a estrutura de árvore. A literatura define diversas estruturas de árvores, como árvores binárias, B-*tree*, entre outras. Um problema na representação utilizando esses tipos de árvores é a indireção, que pode afetar a performance do sistema, portanto, para evitar os efeitos da indireção de ponteiros, os indivíduos são representados utilizando *Heaps*, que são árvores especializadas. Nesse tipo de estrutura, os ponteiro são implicitamente substituidos por indices. O nó a esquerda está na posição $2n + 1$ e $2n + 2$. Outra vantagem no use de *Heaps* está no fato dos dados serem armazenados linearmente, o que reduz os efeitos do *cache miss*, causado por dados espalhados na memória.

O sistema é flexível na máxima profundidade da indivíduo (representado por uma árvore) e permite ao usuário especificar qualquer profundidade aos indivíduos. Um fator que pode ser vantajoso ou não é que o espaço ocupado por um indivíduo é determinístico, ou seja, cada nível possui 2^l nós, onde l é o nível. Cada indivíduo possui $\sum_{l=0}^{L} 2^l = 2^L - 1$, onde L é a profundidade máxima da árvore. Portanto, para uma população com N indivíduos e profundidade máxima L, há $N \times (2^L - 1) \times$ `sizeof(nó)`.

Em conclusão, os indivíduos são representados como árvore no ponto de vista semântico e como vetor contíguo na memória. Espera-se que com essa estrutura haja um ganho na performance devido a redução dos efeitos da indireção e por conseguência de *cache miss*.

2.2 Operadores de Geração

A literatura de define vários operadores de geração de indivíduos, como o método *Full*, *Grow* e a combinação de ambos, denominado *Ramped Half and Half*. O método *Full*, gerada um indivíduo preenchendo todos os seus nós. O método *Grow*, em contraste com o método anterior, preenche um usuário não necessariamente preenchendo todos os nós. Por fim, o último método gera individuos utilizando os dois métodos anteriores, com 50% de probabilidade para ambos.

Nessa implementação, há suporte para os três tipos de geração. O método *Full* é gerado iterativamente, preenchendo os $L - 1$ niveis da árvore do indívido com funções ou operadores e a última camada com terminais (constantes ou variáveis). O método *Grow* é gerado com uma função recursiva, que escolhe um operador ou função como raíz e utiliza uma função de distribuição uniforme para decidir se avança ou não mais um nível em cada ramo da árvore, isto é, o filho à esquerda ou à direita. Já o método *Ramped Half-and-Half* utiliza também uma função de distruibuição uniforme para decidir qual dos dois métodos utilizar.

2.3 Operadores de Seleção

Dois operadores de seleção que são frequentemente utilizados são o *Roulette Wheel* e o *k-Tournament*. O primeiro método seleciona algum indivíduo da população com probabilidade proporcional a sua *fitness*, isto é, $p(i) = f_i \sum_{k=0}^{N} f_k$. Já o último método amostra k indivíduos aleatoriamente e seleciona o indivíduo que possui a maior *fitness* dessa amostragem.

2.4 Operadores de Crossover

2.5 Operadores de Mutação

2.6 Eletismo

2.7 Fitness

3 Experimentos

4 Conclusão