Universidade Federal de Santa Catarina – UFSC

Centro Tecnológico – CTC

Curso Sistemas da Informação

Disciplina: Sistemas Inteligentes

Aluno: Artur Silva Muniz Junior (16101095)

Relatório Trabalho 4: NeuroEvolução com Topologias Aumentadas

Florianópolis, 10 de Julho de 2019

1. Pesquisa Teórica

Esse trabalho foi fortemente inspirado no artigo “Evolving Neural Networks through Augmenting Topologies”, que descreve o algoritmo NEAT. NEAT é um algoritmo que permite buscar através de algoritmos genéticos uma solução topologia ótima, representando uma rede neural. A ideia no trabalho foi desenvolver uma versão minimizada do NEAT.

Os genomas foram representados, conforme instrui o artigo, através de dois arrays. O primeiro contém informações sobre os nodos (id, função de ativação, *bias* e seu tipo -- se é uma entrada, saída ou unidade escondida). O segundo representa as ligações entre tais nodos, seu peso, se está habilitada ou não e seu número de inovação.

O número de inovação faz parte do que os autores chamam de “*historical marking*” e permite que uma ligação seja identificada através de genomas herdeiros ao longo do tempo.

O crossover funciona selecionando primeiro o pai mais apto, copiando seus nodos para o filho e extraindo dele a lista de números de inovação. Uma a uma as conexões da lista de inovação são alinhadas. Caso ambos os pais tenham o gene em comum (ou seja, há uma ligação com o mesmo número de inovação nos dois), o filho herdará aleatoriamente de um deles. As demais ligações que o pai mais apto serão herdadas também. Caso ambos os pais tenham a mesma adaptação, os demais genes de ligação de ambos serão herdados.

Além disso, o algoritmo implementado conta com cinco forma de mutações, que ocorrem com igual frequência respeitando a taxa de mutação, abaixo explicarei um pouco de cada mutação

ADD_NODE: Essa mutação adiciona um nó ao genoma. Uma conexão é escolhida aleatoriamente e é desabilitada. O novo nó é iniciado um uma função de ativação e peso aleatório e é conectado ao nó de saída da conexão sorteada. O nó de entrada é conectado ao nó adicionado.

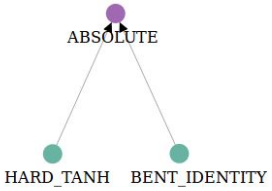
REM_NODE: Remove um nó do genoma. Nós de entrada e saída nunca serão removidos.

MOD_BIAS: Modifica o peso de *bias* de um nó.

MOD_WEIGHT: Modifica o peso de uma conexão.

MOD_ACTV: Modifica a função de ativação de um nó.
O propósito de aplicar esse algoritmo foi, inicialmente, puramente didático, porém os resultados indicam que mesmo que seja simples o algoritmo é capaz de modelar automaticamente redes neurais eficientes.

Foram testada duas formas de seleção, a primeira selecionava apenas os mais aptos e a segunda seleciona no estilo “roleta”, onde quanto maior a aptidão de uma solução maior a chance dela ser escolhida.
Na primeira forma o algoritmo não consegue convergir a uma solução satisfatória e apresenta taxas de erros superiores a 4% para uma rede neural que aproxime a função XOR. Já com a segunda forma, mantendo-se os parâmetros, obtemos taxas de erros em torno de 0,3%.
Usando altas taxas e quantidades de mutação a população se comporta de forma esparsa e explora uma gama maior de “estilos” de solução. Porém usando taxas menores, conseguimos chegar no resultado esperado mais rapidamente



Error: 0.0036567533296186514
Generations: 118
elitism: 4
popSize: 200
maxGenerations: 1500
mutationAmout: 1
mutationRate: 0.01
targetError: 0.005

Com baixas mutações, convergindo em 0,3% em 118 gerações.

Nesse tipo de algoritmo o elitismo protege as “espécies” que estão performando bem, e uma vez que uma mutação de adição/remoção de nó tem grandes impactos na aptidão da solução o algoritmo não converge sem o seu uso.

Referências

1. Evolving Neural Networks throughAugmenting Topologies
http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf