MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
(Real Academia de Artilharia, Fortificação e Desenho/1792)
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Jan Segre
Victor Bramigk

Heurística Estática para Times Cooperativos de Robôs

Rio de Janeiro
Outubro de 2013

Instituto Militar de Engenharia

Heurística Estática para Times Cooperativos de Robôs

Iniciação à Pesquisa apresentada ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia.
Orientador: Paulo F. F. Rosa - Ph.D

Rio de Janeiro
Outubro de 2013

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

xxxx Segre, J., Bramigk, V.
Heurística Estática para Times Cooperativos de Robôs / Jan Segre, Victor Bramigk : Instituto Militar de Engenharia, 2013.

Iniciação à Pesquisa (IP) - Instituto Militar de Engenharia - Rio de Janeiro, 2013.

1. Engenharia da computação. 2. Redes Neurais. 3. Lógica Nebulosa. 4. Otimização da Colônia de Formigas. Instituto Militar de Engenharia.

Instituto Militar de Engenharia

Jan Segre
Victor Bramigk

Heurística Estática para Times Cooperativos de Robôs

Iniciação à Pesquisa apresentada ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia.

Orientador: Paulo F. F. Rosa - Ph.D

Aprovado em 9 de outubro de 2013 pela seguinte Banca Examinadora:

Paulo F. F. Rosa - Ph.D
Orientador

Julio Cesar Duarte - D.Sc. do IME

Ricardo Choren Noya - Ph.D do IME

Rio de Janeiro
Outubro de 2013

Resumo

O objetivo deste trabalho é prever como um time de futebol de robôs irá se comportar baseado somente nas posições e orientações de um conjunto discreto de amostras. Para isso, foram estudados os métodos da ACO (Ant Colony Optimization), SA (Simulated Annealing), Algorítimo Genético, Lógica Nebulosa e Redes Neurais. A partir do estudo detalhado desse algoritmos definiu-se duas linhas principais de ação para a solução do problema: uma baseada em Logica Nebulosa e a outra baseada em Redes Neurais. Após um estudo mais aprofundado deseja-se implementar um processo de otimização em ambos os algorítimos para que o resultado seja refinado.

Abstract

The main objective of this work is predicting how a robot soccer team whill behave, based on a set of positions and orientations of a discrete sample. For that, some heuristics were studied: ACO (Ant Colony Optimization), SA (Simulated Annealing), GA (Genetic Algorithm), and Neural Networks. Based on the detailed study of these algorithms two branches were defined as candidate solutions: one based on the Neural Network heuristics, and the other based on Fuzzy logic. After a deeper study on of these methods it's desirable to implement an algori- thm which will refinethe result.

Sumário

Lista de ilustrações

1 Introdução

1.1 Contextualização

O Laboratório de robótica da seção de computação do IME possui um time que participa de uma competição de futebol de robôs oficializada pela *RoboCup*. Essa competição consiste em partidas de futebol autônomas entre robôs com formato de cilindro de 18cm de diâmetro e 15cm de altura em um campo de 6m por 4m. Além disso o sistema de visão computacional é padronizado e centralizado. O importante é notar que os times são autônomos e por isso há vários desafios envolvidos. Um dos desafios é modelar o time adversário, de maneira geral modelar um time desconhecido mas com o intuito de conseguir prever com certa confiança suas decisões para tomar decisões melhores durante o jogo e conseguir um desempenho melhor. As partidas são registradas e por isso podem ser feitas análises pré jogo além de em tempo real. A principal vantagem é a liberdade sobre quão eficiente é o método de análise.

O objeto a ser estudado é, portanto, a análise dos registros dessas partidas, denominados *logs*, com foco principal em modelar um time desconhecido. O modelo gerado será utilizado para prever os movimentos do time adversário. Isso deve tornar as jogadas planejadas mais eficazes, uma vez que já levarão em conta os movimentos futuros do time adversário.

1.2 Problema

O problema descrito anteriormente pode ser enunciado da seguinte maneira:

Dado um conjunto E_p de possíveis estados do jogo, A_{t_ad} um conjunto de ações dos robôs do time adversário, um sistema $f : E_p \rightarrow A_{t\_ad}$. Encontrar o sistema $F : E_p \rightarrow A_{t\_ad}$ que minimiza:

$$e_{total}(F) = \sum_i E[f(x_i), F(x_i)] \tag{1.1}$$

Onde $E : A_{t_ad} \rightarrow \Re^+$ é uma função erro que é proporcional a diferença entre as ações e $x_i \in E_p$ é um conjunto limitado de *logs*.

Como o sistema F pode ser escrito de várias maneiras, é necessário assumir uma forma

menos genérica para F de modo a possibilitar a análise e reduzir o espaço de busca. Uma vez que o conjunto das possíveis estruturas de F escolhido afeta o valor mínimo F_{min}, tem-se que o erro $e_{total}(F_{min})$ esta atrelado a estrutura da função F escolhida. Logo, pretende-se aplicar um método de otimização que tem como parâmetros elementos que definem a estrutura escolhida. Por exemplo, o número de camadas arquitetura da rede neural ou as regras utilizadas no sistema difuso.

1.3 Estrutura do Trabalho

Assim, inicialmente são apresentados os métodos candidatos a serem utilizados em um algorítimo de predição automática do comportamento de um time de futebol de robôs, isto é, determinar a função F especificada no problema anterior. juntamente com heurísticas de otimização combinatorial para otimizar a estrutura desta função. Em seguida são propostas abordagens envolvendo o tipo de estrutura da função F e o método utilizado para procurar a estrutura pseudo ótima. Por fim é apresentado as próximas etapas previstas para dar continuidade ao trabalho.

2 Métodos

2.1 Lógica Nebulosa

Sistemas nebulosos aproximam funções. Eles são aproximadores universais se usarem regras suficientes. Neste sentido sistemas difusos podem modelar qualquer função ou sistema contínuos. Aqueles sistemas podem vir tanto da física quanto da sociologia, bem como da teoria do controle ou do processamento de sinais.

A qualidade da aproximação difusa depende da qualidade das regras. Na prática especialistas sugerem regras difusas ou aprendem-nas através de esquemas neurais através de dados e ajustam as regras com novos dados. Os resultados sempre aproximam alguma função não linear desconhecida que pode mudar com o tempo. Melhores cérebros e melhores redes neurais resultam em melhores aproximações (KOSKO, 1997).

2.1.1 Modelo Aditivo Padrão(SAM)

O sistema difuso $F : \Re^n \rightarrow \Re^p$ é em si uma árvore de regras rasa e extensa. É um aproximador por antecipação. Existem m regras da forma "Se X é conjunto difuso A então Y é conjunto difuso B". A partir desse nível o sistema depende cada vez menos em palavras.

Cada entrada x aciona parcialmente todas as regras em paralelo. Então o sistema age como um processador associativo a medida que calcula a saída $F(x)$.

Essas regras relacionam os conjuntos A_j e B_j, gerando o caminho difuso $A_j x B_j$. Na prática, é utilizado o produto para definir $a_j x b_j(x, y) = a_j(x).b_j(y)$. Esta é a parte "padrão"no SAM. A parte "aditiva"se refere ao fato de a entrada x acionar a j-ésima regra em um grau $a_j(x)$ e o sistema soma os acionamentos ou partes escaladas dos conjuntos escalados $a_j(x)B_j$, (KOSKO, 1997):

$$F(x) = \frac{\sum w_i.a_i(x).V_i.c_i}{\sum w_j.a_j(x).V_j} \tag{2.1}$$

Com o volume/área V_j e o centroide c_j são dados por:

$$V_j = \int b_j(y_1, ..., y_p)_{\Re^p}.dy_1...dy_p > 0 \tag{2.2}$$

$$c_j = \frac{\int y.b_j(y_1, ..., y_p)_{\Re^p}.dy_1...dy_p}{V_j} \tag{2.3}$$

2.2 Otimização da Colônia de Formigas

Na busca por alimento, as formigas utilizam de feromônios para encontrar o melhor caminho. Isso acontece da seguinte maneira: cada formiga deposita feromônio ao se deslocar. A partir da avaliação da quantidade de feromônio depositada por formigas que já passaram pelo local, formigas subsequentes tem mais probabilidade de se mover em rotas que tem mais feromônios. Ao decorrer do tempo os feromônios vão evaporando, apagando rastros que não foram reforçados. Com isso, caminhos que são percorridos por mais formigas tem mais chance de serem percorridos por outras formigas do que aqueles que foram percorridos por menos formigas e caminhos que foram percorridos á pouco tempo tem mais chance de serem percorridos que caminhos percorridos a muito tempo. A quantidade de feromônio depositado é mais intensa no trajeto de volta, quando a comida foi encontrada. Outro fator que é levado em consideração é a qualidade da comida encontrada, de maneira que mais feromônio é depositado quanto melhor for a fonte de alimento encontrada. A medida que mais formigas exploram o local e encontram alimento, esse procedimento tende a otimizar o trajeto entre a fonte de alimento e a colônia.

Apesar dessa heurística utilizada pelas formigas ser interessante para se resolver problemas combinatórios do tipo NP(i.e., com complexidade não polinomial), são necessários algumas adaptações na construção de um algoritmo computacional.

A seguir é apresentado a meta-heurística do ACO(*Ant Colony Optimization*) algoritmo, juntamente com observações relacionadas as diferenças entre a heurística do ACO e o comportamento natural das formigas descrito anteriormente.

2.2.1 Pseudo código da meta-heurística do ACO

```
Procedimento
    enquanto n < N_MAX_IT faça
        AgendarAtividade
            ConstruirSolucoesFormigas
            AtualizarFeromonios
            // opcional:
            AcoesGlobais
        fim
    fim
fim
```

Algoritmo 1: Pseudo código da meta-heurística do ACO

A meta-heurística do ACO pode ser subdividida em três partes, conforme proposto por (DORIGO; STÜTZLE, 2004): *ConstruirSolucoesFormigas*, *AtualizarFeromonios* e *AcoesGlobais*.

ConstruirSolucoesFormigas gerencia a movimentação de uma colônia de formigas em torno dos nós vizinhos. A escolha do próximo nó é feita através de uma decisão estocástica que é função da quantidade de feromônio no nós vizinhos e informação heurística. Quando uma formiga encontra uma solução, ou enquanto a solução é construída, esta avalia a qualidade da solução(completa ou parcial) que será utilizada pelo procedimento *AtualizarFeromonios* para decidir a quantidade de feromônio que será depositada. Outro procedimento relevante na construção da solução é a eliminação de possíveis ciclos, utilizado por exemplo, no problema do caixeiro viajante.

AtualizarFeromonios é o processo que atualiza os traços de feromônio depositados pelas formigas no espaço de busca. Os traços de feromônio podem aumentar, caso uma formiga tenha visitado o nó/conexão em questão, ou diminuir, devido ao processo de evaporação do feromônio. Esse procedimento faz com que nós/conexões que foram visitados por muitas formigas ou por uma formiga e que tenha levado em uma solução boa aumentem a probabilidade de serem visitados por futuras formigas. Semelhantemente, reduz a probabilidade de que nós que não foram visitados por novas formigas por muitas iterações sejam visitados novamente. Logo, este procedimento evita a convergência a caminhos sub ótimos, favorecendo também a exploração de novas regiões do espaço de busca.

Por fim, o procedimento *AcoesGlobais* é utilizado para centralizar ações que não podem ser executadas pelas formigas individualmente. Um exemplo de ações desse tipo é a filtragem de soluções ou o favorecimento de regiões por meio de informações globais.

O procedimento *AgendarAtividade* não necessariamente é uma instrução sequencial. Pode-se, portanto, implementá-lo de maneira sequencial ou paralela, síncrona ou assincronamente. O tipo de abordagem que será utilizada depende das características do problema que se deseja resolver.

2.3 Recozimento Simulado

No processo de recozimento de um metal, a quantidade de energia interna livre esta intrinsecamente relacionada ao processo de resfriamento em que o metal é submetido. Quanto mais rápido se resfriam um metal mais energia é armazenada internamente. Isso pode ser explicado considerando que o tempo que a estrutura leva para atingir o estado de menor energia é maior que o disponível devido a redução da mobilidade dos átomos com o decaimento da temperatura. Com efeito, quanto maior a taxa de resfriamento maior o número de defeitos na estrutura do sólido e menor o tamanho médio dos grãos. Quando se reduz a taxa de resfriamento, há uma maior chance de se atingir configurações mais estáveis. Como resultado, a energia interna é reduzida. De acordo com (BERTSIMAS; TSITSIKLIS, 1993), pode-se modelar a probabilidade p_{ij} de uma configuração atômica $\{r_i\}$ com energia $E\{r_i\}$ passar para a configuração $\{r_j\}$ com energia $E\{r_j\}$ na temperatura T como:

$$p_{ij} = \begin{cases} 1 & \text{se } E\{r_j\} \leq E\{r_i\} \\ exp\left\{-\frac{(E\{r_j\}-E\{r_i\})}{k_B.T}\right\} & \text{se } E\{r_j\} > E\{r_i\} \end{cases} \tag{2.4}$$

Onde k_B é a constante de Boltzmann. Para se reduzir a energia livre, é necessário que uma rotina de resfriamento seja escolhida de acordo com o tipo de material a ser resfriado.

Conforme proposto por Kirikpartrick, Gellett e Vechin (1983) e Cerny (1985), pode-se desenvolver uma heurística probabilística para se encontrar o mínimo global de uma função custo que possua vários mínimos locais fazendo-se uma analogia com o fenômeno físico descrito acima. A meta-heurística induzida por este processo é chamada de meta-heurística *Simulated Annealing*(Recozimento Simulado), ou SA, apresentado a seguir.

2.3.1 Meta-heurística do SA

De acordo com (BERTSIMAS; TSITSIKLIS, 1993), os elementos básicos da meta-heurística do SA para a resolução de um problema combinatório são:

1. Um conjunto finito S.

2. Um função custo J de imagem real definida em S. Seja $S^* \subset S$ o conjunto de todos os mínimos globais da função J, suposto subconjunto próprio.

3. Para cada $i \in S$ um conjunto $S(i) \subset S - \{i\}$, chamado de conjunto das vizinhos de i.

4. Para cada i, uma coleção de coeficientes positivos q_{ij}, $j \in S(i)$, tal que $\sum_{j \in S(i)} qij = 1$.

5. Uma função não crescente $T : \mathbf{N} \to (0, \infty)$, chamada de rotina de resfriamento. Aqui $\mathbf{N}$ representa o conjunto de inteiros positivos, e $T(t)$ é chamada de *temperatura* no tempo t.

6. Um estado inicial $x(0) \in S$.

Com base na definições acima, tem-se o seguinte pseudo código para a meta-heurística do SA:

Procedimento
  SetarValoresInicias;
  para $n = 1$ *até* N_{MAX_IT} *ou* $J(x^*) \leq TOL$ **faça**
    para $k = 1$ *até* N_{MAX_IT} *ou a solução convergir* **faça**
      EscolherVizinho
        selecionar algum $j \in S(i)$;
      fim
      CalcTransicao
        $\Delta J \leftarrow J(j) - J(i)$;
        se $DeltaJ \leq 0$ **então**
          $x(t+1) \leftarrow j$;
          $x^* \leftarrow j$;
        fim
        senão
          $q_{ij} \leftarrow exp^{-\frac{\Delta J}{T(t)}}$;
          se $random() < q_{ij}$ **então** $x(t+1) \leftarrow j$;
          senão $x(t+1) \leftarrow i$;
        fim
      fim
    fim
    AtualizarTemperetura;
  fim
fim

Algoritmo 2: Pseudo código da meta-heurística do SA

No algorítimo 2, o procedimento *AtualizarTemperetura* executa a rotina de resfriamento através da função $T(t)$ definida anteriormente. Já o procedimento *EscolherVisinho* escolhe aleatoriamente um dos elementos da vizinhança do vértice atual i.

2.4 Algorítimo Genético

Um *algorítimo genético* é uma heurística de busca que procura imitar a seleção natural que ocorre no processo evolucionário dos organismos vivos.

Nessa heurística, uma população de soluções (também chamadas de indivíduos ou fenótipos) para problemas de otimização é evoluída para conseguir soluções melhores. Cada solução possui um conjunto de propriedades (cromossomos ou genótipos) que podem

ser mutados ou alterados.

Os requerimentos são, tipicamente:

- uma representação genética da solução
- uma função de aptidão para avaliação da solução

2.4.1 O processo

O processo é iniciado com uma população com propriedades geradas aleatoriamente.

A iteração da heurística se da em 3 etapas:

- procriação: indivíduos são pareados e é aplicada a operação de cruzamento (*crossover*)
- mutação: alguns indivíduos são selecionados e é aplicada a operação de mutação (*mutation*)
- seleção: é usada a função de aptidão para descartar os indivíduos menos aptos restando as soluções que de fato trouxeram alguma melhora.

As condições mais comuns para terminação do processo são as seguintes:

- encontrada uma solução que atende os requisitos mínimos
- número fixo de gerações alcançado
- recursos alocados (tempo ou dinheiro) alcançados
- a melhor solução alcançou um patamar estável em que mais iterações não produzem soluções melhores
- inspeção manual

2.4.2 Limitações

As limitações mais comuns no emprego de um algorítimo genético são:

- Funções de avaliação computacionalmente caras tornam essa heurística ineficiente.

- Não escala bem com a complexidade, isto é, quando o número de elementos expostos a mutação é grande o espaço de busca cresce exponencialmente. Por isso, na prática algorítimos genéticos são usados para, por exemplo, projetar uma hélice e não um motor.
- A melhor solução é relativa às outras soluções, por isso o critério de parada não é muito claro em alguns problemas.
- Em muitos problemas os algorítimos genéticos tendem a convergir para um ótimo local ou as vezes pontos arbitrários em vez do ótimo global.
- É difícil aplicar algorítimos genéticos para conjunto de dados dinâmicos. Pois as soluções podem começar a convergir para um conjunto de dados que já não é mais válido.
- Algorítimos genéticos não conseguem resolver eficientemente problemas em que a avaliação é binária (certo/errado), como em problemas de decisão. Nesse caso buscas aleatórias convergem tão rápido quanto essa heurística.
- Para problemas mais específicos existem outras heurísticas que encontram a solução mais rapidamente.

2.4.3 Pseudo código de um Algorítimo Genético

Procedimento
  $k \leftarrow 0$;
  $P_k \leftarrow$ população de n indivíduos escolhidos aleatoriamente;
  enquanto *a avaliacao*(i) *de cada i em* P_k *não for boa o suficiente* **faça**
    Selecionar os $(1 - \chi) \times n$ membros com maior *avaliacao*(i) de P_k e inserir em P_{k+1};
    Selecionar $\chi \times n$ membros de P_k, pareá-los e inserir a cria em P_{k+1};
    Selecionar os $\mu \times n$ membros de P_{k+1} com maior *avaliacao*(i) e inverter um bit aleatório de cada membro;
    $k \leftarrow k + 1$;
  fim
  melhor $\leftarrow$ o membro i em P_k com maior *avaliacao*(i);
  retorna *melhor*
fim

2.5 Rede Neural

O termo mais apropriado é rede neural artificial, já que apenas rede neural pode se referir ao sistema biológico de nervos, no antando dado o contexto desse texto e o uso

consagrado do termo “rede neural”, esse será usado no lugar da versão mais explícita “rede neural artificial”.

Uma rede neural é um sistema inspirado no sistema nervoso central (em especial o cérebro) encontrado em muitos animais. A ideia básica é ter um grafo em que cada nó abstrai um neurônio e é representado como uma função, alguns desses nós são responsáveis pela observação e outros pela saída e os nós de entrada alimentam os próximos nós até chegar nos nós de saída. (HAYKIN, 2001)

A figura 1 exemplifica uma rede neural *feedforward*, que é baseada num grafo direcionado acíclico, em que podem ser vistas 3 camadas a primeira é chamada de camada de entrada, a última, de saída e as intermediárias, de escondidas. (SHIFFMAN; FRY; MARSH, 2012)

Um dos diferencias da rede neural é a capacidade de aprender, essa heurística forma um sistema adaptativo. Existem três tipos de aprendizados:

- Aprendizado supervisionado: alimentar a rede com um problema cuja a solução é conhecida e depois fornecer a resposta certa para que a rede possa se ajustar.
- Aprendizado não supervisionado: consiste em buscar padrões não conhecidos, não se conhece a resposta certa ou se uma resposta é certa ou não.
- Aprendizado por reforço: alimentar a rede com um problema cuja a solução pode ser avaliada em boa ou má. Esse tipo de aprendizado é comum em robótica onde o robô caminha por um ambiente e tem o reforço negativo ou positivo de colidir ou encontrar o objetivo.

2.5.1 O *Perceptron*

O bloco de construção básico de uma rede neural são os neurônios. Um *perceptron* é a rede neural mais simples possível: é formada por apenas um neurônio.

O funcionamento de um *perceptron* pode ser descrito nos seguintes passos:

- Receber e armazenar as entradas;
- Pesar as valores armazenados: consistem em multiplicar cada um pelo peso correspondente àquela entrada;
- Enviar o saída calculara anteriormente.

2.5.1.1 Exemplo de aplicação

Considere uma reta em $\Re^2$, que separa o plano em duas regiões A e B. O problema a ser resolvido pelo *perceptron* é dizer se um ponto (x, y) está em A ou B. As entradas do problema são as coordenadas x e y, e a saída é um escalar cujo o valor é um escalar positivo para sinalizar o conjunto A e negativo para sinalizar o B.

O *perceptron* poderia ser modelado com duas entradas, porém note que desse modo o ponto $(0, 0)$ sempre irá resultar numa saída igual a 0. Para evitar isso são usadas três entradas no *perceptron* sendo que uma delas é sempre igual a 1 e é chamada de *bias*, que serve basicamente para adicionar um peso aditivo e não só multiplicativo. O funcionamento desse *perceptron* ser observado na figura 3.

Esse exemplo usará o método supervisionado de aprendizado, que funciona da seguinte maneira:

- Alimentar o *perceptron* com uma entrada para o qual se conhece a resposta;
- Pedir a saída ao *perceptron*;
- Computar o erro;
- Ajustar os pesos de acordo com o erro;
- Repetir o processo.

O ajuste dos pesos pode ser feito calculando o erro e incrementando o peso um fator de aprendizado vezes o erro vezes a entrada daquele peso.

Assim o *perceptron* é capaz de ser treinado e sua saída ser cada vez mais próxima da desejada.

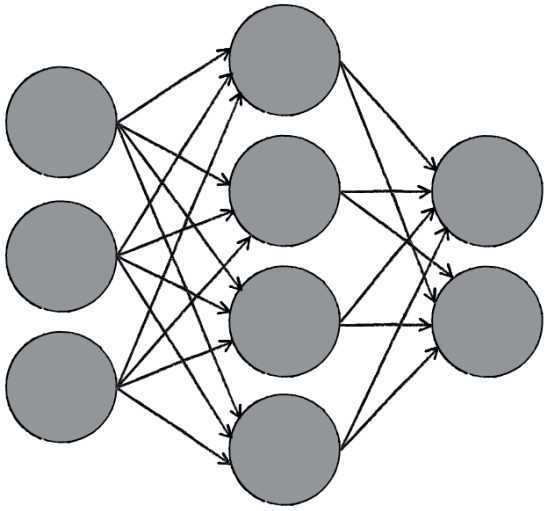

Figura 1 – Rede neural com três camadas.

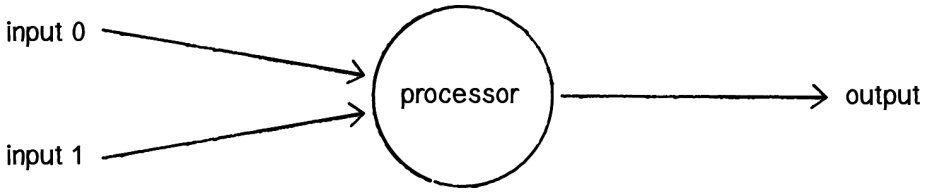

Figura 2 – *Perceptron* de duas entradas e uma saída.

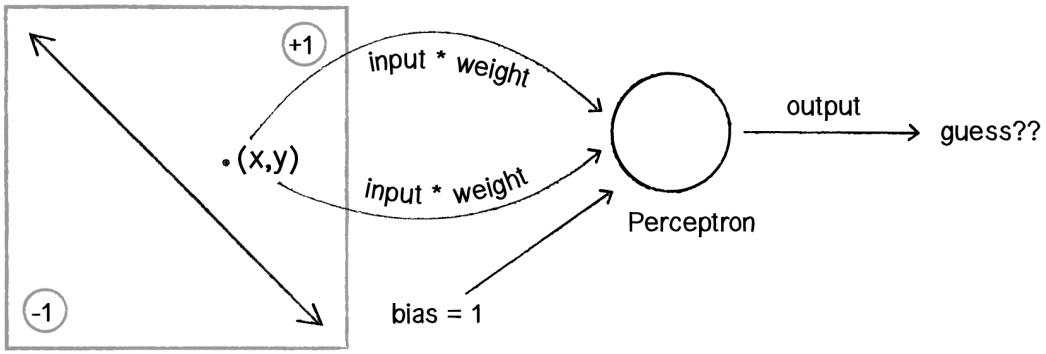

Figura 3 – *Perceptron* para decidir região de um ponto no plano.

3 Análise das Possíveis Abordagens

Como os métodos AG (Algorítimo Genético), SA e ACO são heurísticas de otimização, é necessário que o problema da aproximação de uma função seja reduzido a um problema de otimização. Uma abordagem é se escolher uma base finita de funções ortonormadas e minimizar a soma dos quadrados da norma da diferença entre os valores de treinamento e uma combinação linear das funções da base escolhida. Foi suposto, por simplicidade, que o espaço dos vetores um espaço vetorial com norma. Isso restringe o espaço solução, uma vez que apenas funções que são combinação linear das funções da base serão solução.

O método da Lógica Fuzzy, conforme exposto anteriormente, necessita que um conjunto de regras seja definido. Essas regras poder ser geradas a partir de uma análise mais detalhada do problema, mas também podem ser aprendidas via rede neural.

Já a Rede Neural define implicitamente a estrutura interna que minimiza a diferença entre a saída real e a saída desejada. Entretanto, é necessário definir a topologia mais adequada para o problema em questão. Também, apesar de não ser necessário, pode-se decompor o problema em subproblemas e "atribuir redes neurais um subconjunto de tarefas que coincidem com suas capacidades inerentes"(HAYKIN, 2001, pag. 29) com o objetivo de aumentar a adaptabilidade da rede. Apesar de ser uma modelagem para a arquitetura da inteligência a ser mapeada, não representa uma restrição tão considerável quando a das abordagens citadas anteriormente.

Com base nisso, as seguintes abordagens foram estabelecidas:

1. Utilizar Lógica Fuzzy com regras geradas através de uma rede neural;
2. Utilizar Rede Neural com uma topologia mista.

Na primeira abordagem, pretende-se utilizar os métodos de otimização para se encontrar a melhor topologia de rede neural que melhor otimiza o conjunto de regras buscado.

Pretende-se também, na segunda abordagem, utilizar os métodos de otimização descritos anteriormente para se encontrar a melhor topologia que aproxima da melhor maneira.

4 Próximas Etapas

As próximas etapas consistem em estudar a solução através dos métodos e das abordagens propostas de problemas mais simples para consolidar o estudo dos métodos estudados e analisar as topologias das redes neurais para que futuramente seja possível encontrar a topologia ótima e assim refinar os resultados da rede e das regras do sistema difuso.

Referências

BERTSIMAS, D.; TSITSIKLIS, J. Simulated annealing. *Statistical Science*, JSTOR, p. 10–15, 1993. Citado na página 13.

DORIGO, M.; STÜTZLE, T. *Ant Colony Optimization*. [S.l.]: Bradford Book, 2004. ISBN 0262042193. Citado na página 12.

HAYKIN, S. *Redes neurais*. [S.l.]: Grupo A, 2001. Citado 2 vezes nas páginas 18 e 21.

KOSKO, B. *Fuzzy engineering*. [S.l.]: Prentice-Hall, Inc., 1997. Citado na página 10.

SHIFFMAN, D.; FRY, S.; MARSH, Z. *The Nature of Code*. [S.l.]: D. Shiffman, 2012. Citado na página 18.