Universidade Federal do Rio Grande do Sul
Instituto de Informática

Futebol de Robôs
Redes Neurais 2015/2

Gabriel Osório Alves
Cartão: 217871

1 - Introdução:

O relatório contém os dados dos robôs, um usando lógica fuzzy (SIF) e outro utilizando redes neurais, o seu desenvolvimento, escolhas e codificação, resultados, conclusão e possíveis melhorias. Este trabalho foi desenvolvido na cadeira de Redes Neurais e Lógica Fuzzy ministrada pelo professor Paulo Martins Engel no Instituto de Informática da UFRGS.

2 - Robô Fuzzy:

2.1 - Bases da Implementação

Implementação feita em C++ utilizando as bibliotecas disponibilizadas pelo professor tendo a divisão de classes mostrada à direita.

As principais classes desenvolvidas no trabalho foram:

- **FuzzyTrapezium, FuzzyTriangle, FuzzyDelta e FuzzyGamma:**
  Modelo matemático das funções fuzzy representadas pelo nome. Todas tem sua função membership e cálculos para encontrar o centro diferenciados.
- **FuzzyLogic**:
  Cálculo das Regras de Ativação;
  Método de Mamdani;
  Cálculo da matriz fuzzy;
  Defuzificação (Centróide).
- **Main:**
  Inicia as regras;
  Controla todas as jogadas e os cálculos;
  Salva os logs para o robô neural.

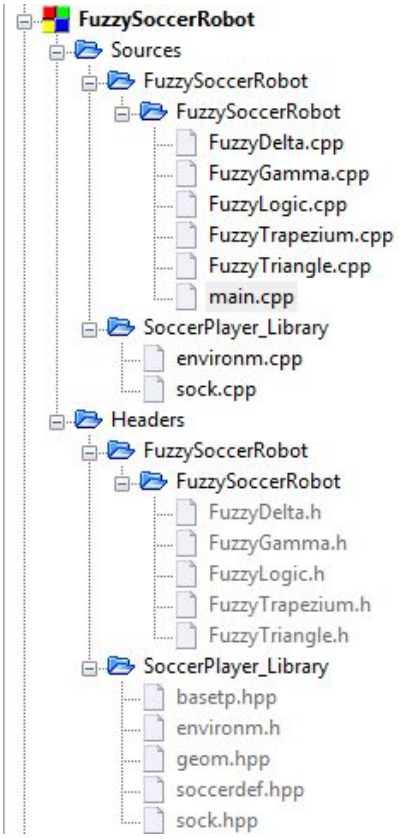

Figura 1: Divisão de classes do Robô Fuzzy

Em resumo, todas as classes que representam funções de ativação (FuzzyTrapezium, etc) herdam da classe FuzzySet. Esta classe tem uma função membership que é a função de pertinência e center o valor utilizado pelo método de inferência das alturas.

```
1  class FuzzySet
2  {
3      public:
4      virtual float membershipFunction(float u) = 0;
5      float center;
6  };
7
```

Figura 2: Classe FuzzySet, base das outras classes de funções de ativação do FuzzyRobot;

Como ilustração temos a classe FuzzyDelta que, assim como as outras classes do mesmo gênero, tem sua função de cálculo de pertinência e inicializações:

```
#include "FuzzyDelta.h"

FuzzyDelta::FuzzyDelta(float alpha, float beta)
{
    this->alpha = alpha;
    this->beta = beta;
    this->center = alpha;
}

float FuzzyDelta::membershipFunction(float u)
{
    float mi = 0.0;
    if (u < alpha) mi = 1.0;
    if ((u >= alpha) && (u <= beta)) mi = (beta - u)/(beta - alpha);
    if (u > beta) mi = 0.0;
    return mi;

}
```

Figura 3: Classe FuzzyDelta, com sua função membership e suas inicializações.

Além disso, temos a classe FuzzyLogic que calcula a força de disparo das regras, esse cálculo é feito da seguinte maneira:

```
float FuzzyLogic::calculateActivationValue(FuzzySet* first, FuzzySet* second, FuzzySet* third)
{
    float mi[3];

    mi[0] = first->membershipFunction(ballAngle);
    mi[1] = second->membershipFunction(targetAngle);
    mi[2] = third->membershipFunction(ballDistance);
    float result = 1.0;
    if (mi[0] < result) result = mi[0];
    if (mi[1] < result) result = mi[1];
    if (mi[2] < result) result = mi[2];

    return result;
}
```

Figura 4: Método que calcula a força de disparo das regras

Também foi desenvolvida uma função para calcular o final calculado, para facilitar a implementação. Essa função foi chamada de mamdaniValue, já que é utilizada por ele no método de defuzificação:

```
float FuzzyLogic::mamdaniValue(float alpha, FuzzySet* group, float z)
{
    float mi = group->membershipFunction(z);
    float result = 1.0;

    if (mi < result)
        result = mi;
    if (alpha < result)
        result = alpha;
    return result;
}
```

Figura 5: Resultado final do que é calculado na função de pertinência após ser dado um alpha e um z.

2.2 - Regras do Sistema de Inferência Fuzzy:

Foram criadas regras parecidas com as vistas na atividade prática em laboratório 1 da disciplina onde utilizamos os conjuntos fuzzy *NB, NS, ZE, PS e PB* para os ângulos da bola e do alvo e os 3 regras adicionais para distância da bola que foram definidas como *Near*, *Medium* e *Long*.

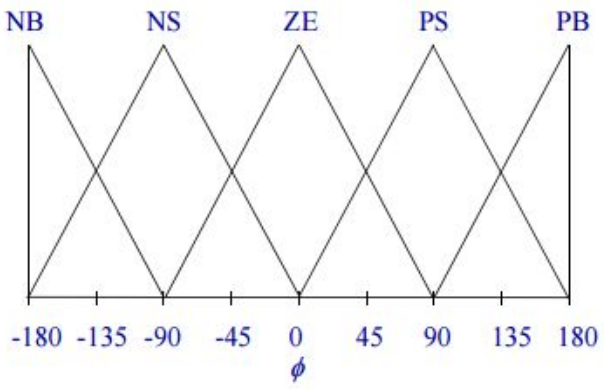

Figura 5: Exemplo das Regras utilizadas para “fuzzyficar” os ângulos do alvo e da bola

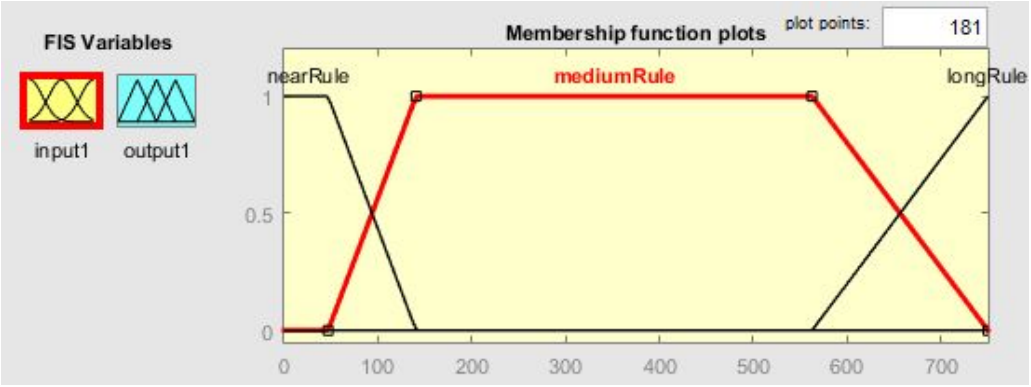

Figura 6: Exemplo das Regras utilizadas para “fuzzyficar” a distância da bola (gerado com o MatLab)

No código foi utilizado o método do centróide (mamdani) com 101 iterações para obter bons resultados para defuzificar o sistema:

```
void FuzzyLogic::centroidDeffuzification(float longMatrix[], float mediumMatrix[], float nearMatrix[],float* sum, float* weightedSum)
{
    float outValue = 0.0;
    // Feito com 101 iterações
    for (int i = 0; i <= 100; i++)
    {
        float z = i*(M_PI/50) - M_PI;
        outValue = 0.0;
        float C = 0.0;

        if ((C = FuzzyLogic::mamdaniValue(longMatrix[0], NB, z)) > outValue) outValue = C;
        if ((C = FuzzyLogic::mamdaniValue(longMatrix[1], NB, z)) > outValue) outValue = C;
        if ((C = FuzzyLogic::mamdaniValue(longMatrix[2], NB, z)) > outValue) outValue = C;
        if ((C = FuzzyLogic::mamdaniValue(longMatrix[3], NB, z)) > outValue) outValue = C;
        if ((C = FuzzyLogic::mamdaniValue(longMatrix[4], NB, z)) > outValue) outValue = C;
```

Figura 7: Método de Mamdani com 101 iterações, são somadas todos os valores calculados e após isso os valores de sum e weightedSum são atualizados;

2.3 - Matrizes de Regras Fuzzy:

Como tinhamos 5x5x3 conjuntos, geramos um total de 75 regras possíveis para o robô fuzzy.

A ideia geral era perseguir a bola da forma mais direta possível caso a bola estivesse a uma distância considerável para diminuir o tempo de trajetória até a mesma (e consequêntemente ser mais rápida que o robô fuzzy disponibilizado pelo professor).

Caso estivesse a uma distância média, tentasse sempre alcançar ela pelos lados sempre que a bola estivesse na frente do robô e o gol atrás ou dos lados, isso fazia com que ele de certo modo protegesse a sua goleira (não ia marcar gol contra, por exemplo) e ao mesmo tempo iria aumentar as chances de marcar um gol no adversário

Caso a bola estivesse praticamente colada no robô, independentemente da distância em relação ao gol ele deveria sempre tentar jogar a bola na direção da goleira adversária, assim de certa forma “chutar” a bola.

Matriz Near					
	NB	**NS**	**ZE**	**PS**	**PB**
NB	NS	NS	PS	PS	NS
NS	NB	NS	NS	PS	NS
ZE	NB	NS	ZE	PS	PB
PS	PS	NS	PS	PS	PB
PB	PS	NS	NS	PS	PB

Matriz Medium					
	NB	**NS**	**ZE**	**PS**	**PB**
NB	NB	NS	PS	PS	NS
NS	NB	NS	PS	PS	NS
ZE	NB	NS	ZE	PS	PB
PS	PS	NS	NS	PS	PB
PB	PS	NS	NS	PS	PB

Matriz Long					
	NB	**NS**	**ZE**	**PS**	**PB**
NB	NB	NS	ZE	PS	PB
NS	NB	NS	ZE	PS	PB
ZE	NB	NS	ZE	PS	PB
PS	NB	NS	ZE	PS	PB
PB	NB	NS	ZE	PS	PB

Figura 6: Matrizes de regras fuzzy geradas

2.4 - Defuzificação e Saída:

A defuzificação foi dada pelo método das alturas pois apesar de ter um cálculo mais complexo ele é mais preciso em comparação com o método do centróide.
//Explicação sobre como foi feito

A saída do SIF era um ângulo que foi convertido para a força utilizada nos dois motores pela função existente no robô de testes que também foi disponibilizado.

Notamos que a saída estava fazendo o robô girar no mesmo ponto ou nunca alcançar a bola de forma certa então diminuímos a intensidade da força aplicada nos motores em 40%, o que já foi o suficiente para gerar resultados bem melhores.

2.5 - Resultados:

O robô fuzzy gerado mostrou-se melhor que o robô disponibilizado pelo professor como robô de testes já que ele vencia em média 66% dos jogos (a cada 2 gols do nosso robô, um gol do robô adversário era feito).

3 - Robô Neural

O robô neural foi desenvolvido utilizando para treino os dados gerados pelo nosso próprio robô fuzzy (chamados de logs em algumas partes futuras do relatório) pois ele tinha obtido resultados melhores.

Em todos os testes, fizemos com que o robô fuzzy jogasse contra ele mesmo e os logs fossem sempre salvos em situações que levassem a um gol. Assim tivemos dados o suficiente para os arquivo .tst e .lrn (treinamento e aprendizado).

3.1 - Neurônios na camada de Entrada e Saída:

Inicialmente o trabalho tinha sido desenvolvido utilizando as recomendações do professor com 8 neurônios na camada de entrada e 2 na camada de saída porém após alguns testes com o resultado do aprendizado da rede neural o robô parecia ter tentado aprender regras que não existiam (apesar da convergência e do MSE não terem sido baixos).

Como solução para o possível problema o arquivo do NeuralRobot.exe foi modificado tendo como entrada somente 3 valores (ângulo da bola, ângulo do alvo, distância até a bola) e continuou com a mesma saída de 2 valores que eram as forças dos motores do robô.

Com essas mudanças o robô teve um desempenho superior, este que será relatado nas seções de treinamentos e resultados do robô neural.

3.2 - Treinamento da Rede Neural:

A seleção de bons parâmetros se mostrou difícil pois boa parte do desenvolvimento foi feito levando em conta os dados que não eram utilizados pelo robô fuzzy (o que fazia com que o método nunca chegasse a bons resultados).

Após várias tentativas com tamanhos de entrada diferentes e com a topologia da camada oculta tendo entre 4 e 16 neurônios, variando o fator de aprendizado entre valores de 0,001 até 0,000001 e com número de épocas indo de 1000 até 50000 foi constatado alguns principais fatores, que são:

a. O fator de aprendizado variar dependendo do tamanho da entrada gerava uma boa taxa de convergência (menor número de épocas);
b. Por causa do primeiro, caso o número de épocas fosse maior do que 10.000 não havia diferença significativa;
c. O número de neurônios na camada oculta demonstrava a maior mudança significativa no resultado final do robô jogando.

3.2.1 - Primeiro Robô Neural Funcional:

Dados de Entrada:

- Tempo de Jogo dos registros gerados pelo robô fuzzy: aproximadamente 30 minutos;
- Número de linhas aproximado do registro: 300.000 linhas de registros;
- Fator de Aprendizado utilizado: 0.0001;
- Número de Épocas utilizado: 5000;
- Melhor Época obtida: 1990;
- MSE obtido: 0.0028;
- Tempo de aprendizado total: 1343 segundos;

Resultado: Esse robô perseguia a bola, porém não tinha parecia ter aprendido as regras mais específicas. (esse robô ainda utilizava dados irrelevantes)

3.2.2 - Segundo Robô Neural Funcional (Correção das Entradas):

Dados de Entrada:

- Tempo de Jogo dos registros gerados pelo robô fuzzy: aproximadamente 10 minutos;
- Número de linhas aproximado do registro: 140.000 linhas de registros;
- Fator de Aprendizado utilizado: 1/140.000;
- Número de Épocas utilizado: 5000;
- Melhor Época obtida: 880;
- MSE obtido: 0.0008;
- Tempo de aprendizado total: 656 segundos;

Resultado: apesar dos dados de entrada serem muito menores (3x menores) e do tempo de aprendizado ter sido um pouco menos da metade do primeiro robô este teve uma melhora gigantesca em relação ao primeiro por algumas razões:

1) Neste robô não estavam sendo utilizados os dados que não faziam parte do conjunto de dados pertinentes as regras do robô fuzzy. Em outras palavras ele só tinha 3 neurônios na camada de entrada.
2) O fator de aprendizado passou a ser utilizado como um valor variante dependendo da entrada (aumentou a convergência do método).

3.2.3 - Terceiro Robô Neural Funcional (Maior tamanho de Entrada):

Como a melhora em relação ao robô anterior foi muito notável

Dados de Entrada:

- Tempo de Jogo dos registros gerados pelo robô fuzzy: aproximadamente 60 minutos;
- Número de linhas aproximado do registro: 550.000 linhas de registros
- Fator de Aprendizado utilizado: 1/550.000
- Número de Épocas utilizado: 1000
- Melhor Época obtida: 9980
- MSE obtido: 0.006
- Tempo de aprendizado total: 7139 segundos

Resultado: O resultado desse robô se assemelhou muito ao do robô fuzzy, com a única excessão de que ele parecia não conseguir representar todas as regras corretamente (eventualmente caso a bola estivesse a uma distância muito grande ele ia em zigue-zague até ela, e não em linha reta como o robô fuzzy) e também tinha certa dificuldade em “chutar” no meio do gol, apesar de acertar na maioria das vezes, boa parte delas ele “chutava” nos cantos da goleira. Isto poderia ser causado por ele não ter um número suficiente de neurônios na camada oculta.

3.2.4 - Quarto Robô Neural Funcional (melhorias nos parâmetros do terceiro robô):

Dados de Entrada:

- Tempo de Jogo dos registros gerados pelo robô fuzzy: aproximadamente 60 minutos;
- Número de linhas aproximado do registro: 550.000 linhas de registros
- Fator de Aprendizado utilizado: 1/550.000
- Número de Épocas utilizado: 1000
- Melhor Época obtida: 9980
- MSE obtido: 0.006
- Tempo de aprendizado total: 3975 segundos;

Resultado: Esse robô obteve resultados quase iguais aos do robô fuzzy mas mesmo assim em jogos contra o robô fuzzy desenvolvido ele perdia (o que até é aceitável).

4 - Resultados:

Em sistemas de inferência fuzzy notamos que é muito difícil representar o ambiente utilizando regras sem um grande conhecimento da área (necessidade de um especialista ou de métodos por tentativa e erro) o que pode ser descrito como a modelagem dos conjuntos e regras. Todavia tivemos um ótimo resultado no robô fuzzy desenvolvido, apesar do grande tempo gasto para aperfeicoá-lo.

Foi comprovado que é possível e muito fácil simular um SIF utilizando aprendizado por redes neurais, por mais que os resultados não tenham ficado ótimos (i.e. uma situação onde o robô neural sempre ou quase sempre ganhe do robô fuzzy), a aproximação é muito útil já que o tempo gasto foi minimizado e focado somente em encontrar os parâmetros certos que representassem corretamente a topologia da rede.

A atividade prática do trabalho consolidou a matéria teórica vista em aula, em diversos pontos do trabalho (com ajuda de pesquisas da internet) cheguei a conclusão de que o que eu precisava já tinha sido visto em aula de forma teórica e cabia a mim desenvolver e implementar essas teorias de forma prática e simples.