Universidade Federal de Pernambuco - UFPE
Centro de Informática - CIn
Graduação de Ciência da Computação
Aprendizagem de Máquina - IF699

# Relatório do Projeto de Aprendizagem de Máquina

Grupo:
Mariama Celi Serafim de Oliveira (mcso)
Mélody Anne Marie Ballouard (mamb)

Recife, 03 de dezembro de 2015

# Sumário

# 1. Introdução

Esse projeto visa classificar um conjunto de dados relacionado ao jogo “Tic-Tac-Toe”, utilizando diferentes classificadores. O objetivo dessa classificação é o seguinte: para cada padrão de jogo do “Tic-Tac-Toe” presente na base de dados fornecido, qual é a probabilidade que o padrão de jogo X seja ganhador e qual é a probabilidade que ele seja perdedor.
Nesse projeto, mostraremos as diferenças em termos de desempenho dos métodos de classificação, utilizando as abordagens vistas na aula de Aprendizagem da Máquina (Algoritmo Fuzzy, classificador Bayesiano, método dos k-vizinhos, regra da soma, MLP e SVM).
Esse relatório é divido em 6 partes: visão geral, a metodologia usada, os resultados que encontramos e a discussão sobre eles. Acabaremos com uma conclusão e a bibliografia.

# 2. Visão Geral

*2.1. Classificador Fuzzy*

O primeiro classificador utilizado foi o algoritmo Fuzzy dado no artigo do professor F.A.T. de Carvalho et al. [1]. Esse algoritmo é composto de várias etapas. Antes da inicialização, devemos calcular a matriz de dissimilaridade entre todos os elementos da base de dados.
Depois, fazemos a inicialização que permite calcular a primeira partição fuzzy, isto significa que atribuimos a cada elemento um grau de pertinência para cada cluster. Nessa inicialização, os clusters são escolhidos aleatoriamente, assim, permitindo calcular o primeiro J, que é o critério de adequação de um cluster as seus elementos.

$$J^{(t)} = \sum_{k=1}^{K} \sum_{i=1}^{n} (u_{ik}^{(t)})^m D(e_i, G_k^{(t)}) = \sum_{k=1}^{K} \sum_{i=1}^{n} (u_{ik}^{(t)})^m \sum_{e \in G_k^{(t)}} d(e_i, e)$$

Onde:

$u_{ik}$ é o grau de pertinência de um elemento para um cluster.
D é a distancia entre dois elementos (Matriz de dissimilaridade).

A próxima etapa é iterativa: com a partição fuzzy, buscamos os melhores representantes. Depois, com o vetor de representantes definido, podemos calcular a melhor partição fuzzy.
Essa estrutura é repetida até que a diferença entre os J’s de duas iterações seja menor que um numéro determinado anteriormente.

*2.2. Classificador Bayesiano*

O segundo classificador utilizado foi o classificador Bayesiano. Ele consiste na seguinte regra: um elemento **x** pertence à classe $w_1$ se $P(w_1|\mathbf{x}) > P(w_2|\mathbf{x})$. Assim, devemos comparar as probabilidades a

posteriori pelas classes $w_1$ e $w_2$, por todos os **x** da base de dados. O cálculo dessas probabilidades a posteriori respeita a seguinte fórmula:

$$posteriori = \frac{verosimilhanca \times priori}{evidencia}$$

O qué é o equivalente a essa fórmula:

$$P(\omega_j|x) = \frac{p(x|\omega_j)\,P(\omega_j)}{p(x)}$$

A fórmula da probabilidade condicional P(**x**|$w_j$) foi fornecida na descrição do projeto, e ela depende dos parâmetros $p_{ij}$,$q_{ij}$ e $r_{ij}$ que também eram dados.

Formula do cáculo da probabilidade a posteriori:

$$P(\mathbf{x}|\omega_j) = \prod_{i=1}^{d} (p_{ij})^{(\frac{x_i(x_i+1)}{2})} (q_{ij})^{(1-x_i^2)} (r_{ij})^{(\frac{x_i(x_i-1)}{2})}$$

Formula do cáculo das probabilidades de $p_{ij}$,$q_{ij}$ e $r_{ij}$ :

$$p_{ij} = \frac{1}{n_j} \sum_{k=1}^{n_j} \frac{x_{ki}(x_{ki}+1)}{2}; q_{ij} = \frac{1}{n_j} \sum_{k=1}^{n_j} (1 - x_{ki}^2); r_{ij} = \frac{1}{n_j} \sum_{k=1}^{n_j} \frac{x_{ki}(x_{ki}-1)}{2}$$

Para determinar P($w_j$), usamos o método da máxima verossimilhança, que consiste em contar o número de elementos da classe $w_j$ e dividir esse número com o número total de elementos. A probabilidade condicional, também, foi determinada por esse método.

*2.3. Método dos K-vizinhos*

Outra técnica de classificação utilizada foi o método dos k-vizinhos, um algoritmo que é altamente efetivo para muitos problemas práticos, pois consegue resistir a ruido dado grandes conjunto de dados [2]. Esse método consiste em atribuir a classe $w_j$ ao elemento **x** quando essa classe é a mais representada nos k-vizinhos. Podemos representar isso em uma fórmula:

*Seja $k_j$ o número de exemplos da classe $w_j$, atribuir **x** à classe $w_j$ se $k_j > k_l$ para todas as classes l diferente de j.*

Para melhorar o resultado é necessário o conceito de "*k-vizinhos*", que significa que a fórmula precedente é aplicada à apenas os *k-vizinhos mais próximos*. No caso do projeto, eles são definidos a partir da distância de dissimilaridade entre os objetos, ou seja, o número de atributos que são diferentes entre dois elementos **x** (sendo **x** um padrão de jogo). A mesma distância foi utilizada pelo algoritmo Fuzzy neste projeto.

*2.4. Regra da Soma*

A regra da soma consiste em simplesmente fazer a soma seguindo uma fórmula específicada dos classificadores que queremos combinar. Sendo dada por esta fórmula:

$$(1-L)p(\omega_j)+\sum_{i=1}^{L} p(\omega_j \mid \mathbf{x}_i)$$

Onde L é o número de classificador usado, P($w_j$) é a probabilidade a priori da classe $w_j$ e P($w_j|\mathbf{x}_i$) é a probabilidade a posteriori para a classe $w_j$ com o classificador *i*. Em nosso caso, L é dois pois utilizamos os classificadores Bayesiano e k-vizinhos.

*2.5. Multi-Layer Perceptron (MLP)*

O Multi-Layer Perceptron (MLP) é uma rede com pelo menos uma camada intermediaria ou escondida. Nesta rede os pesos da camada escondida podem ser reajustado enquanto dure o treinamento. Na rede MLP são introduzidos os conceitos do uso de funções de ativação não-lineares e diferenciaveis como as sigmoidais e o uso do algoritmo backpropagation, todos usados na rede para possibilitar o ajustes dos pesos da camada intermediária [3]. Nesta rede a quantidade de neurônios na camada de entrada é o número de atributos. E parâmetros como o número de nós na camada escondida e a taxa de aprendizado são geralmente determinados de forma experimental [3].
No estudo realizados utilizaremos a Toolbox do Matlab de redes neurais.

*2.6. Support Vector Machines (SVM)*

O Support Vector Machines (SVM) tem suas bases na teoria de Lagrange e consiste em buscar o hiperplano ótimo para dividir classes. Esse hiperplano deve produzir o menor erro de classificação e a maior margem.

O SVM pode usar uma função não linear sobre os atributo do espaço inicial que permite de tornar o problema bidimensional em um problema linear.

Em nosso caso, utilizamos o LIBSVM no Matlab para simular o classificador SVM.

## 3. Metodologia

*3.1. Breve explicação e tratamento inicial dado a base de dados utilizada no experimento*

Durante o experimento fizemos uso de uma base de dados composta de um conjunto de 958 objetos que são representações de partidas de “Tic-tac-toe”. Nosso objetivo foi de comparar os classificadores que permitiam classificar se o jogador X ganha (classe “Positive”) ou perde (classe “Negative”) a partida.

Os atributos de um objeto do conjunto de dados podem ser dos " tipos": **x**, **o** , **b**. Para que seja mais legível e para melhorar o desempenho, transcrevemos esse conjunto de dados para uma tabela (chamada "data" em nosso código) de tamanho de 958 (elementos) por 10 (9 atributos mais a classe). Nessa tabela, **x** é 1, **o** é 0 e **b** é -1. Assim, a classe "Positive" torna-se em 1 e a classe "Negative" em 2.

*3.2. Método de validação do experimento*

Para validar os experimentos de todos os classificadores, com exceção do classificador Fuzzy, utilizamos o método validação cruzada k-fold. Esse método consiste em dividir o conjunto de dados em k partições. Em cada experiência k-1 dessas partições vão ser usadas para o treinamento e uma partição vai ser usada para o teste. A cada experimento, o conjunto de teste vai mudar para que todos os conjuntos sejam usados uma vez como o conjunto de teste e as outras vezes como um conjunto de treinamento. Então, cada vez, uma estimativa do erro é calculada, e a estimativa final vai ser calculada como a média do erro de cada experiência.

No nosso caso, utilizamos validação cruzada k-fold de tamanho 10 sendo o teste entre os classificadores pareado, método que foi repetido dez vezes como forma de garantir validade estatística do experimento. A fim do experimento, obtivemos 100 erros para posteriormente calcularmos a média de erro e o intervalo de confiança.

*3.3. Descrição do teste utilizado com a finalidade de comparar os classificadores - Teste de Friedman*

O Friedman teste consiste em ranquear os algoritmos segundo o seu desempenho e de calcular uma estatística. Então, o ranqueamento depende do erro médio do classificador. Em nosso caso, vamos comparar k (=5 aqui) classificadores (Fuzzy, Bayesiano, k-vizinhos, MLP e SVM).

A formula dessa estatística depende também do número de conjunto de dados N. Como o tamanho da validação cruzada é de 10, a valor de N vai ser de 10.
A primeira etapa é de fazer o ranqueamento e de atribuir uma valor a cada $r_{ij}$, para que cada classificador com cada conjunto de elementos tém um ranqueamento.
Com isso, podemos depois calcular $R_j$ onde $i$ = 1..N e $j$ = 1..K

$$R_j = \frac{1}{N} \sum_{i=1}^{N} r_{ij}$$

O que permite de calcular a estatística final:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$$

Onde:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\right]$$

*3.4. Classificadores utilizados e seus parâmetros*

O primeiro classificador utilizado, o ***Fuzzy***, foi implementado a partir do algoritmo "Single view fuzzy K-set-medoids clustering algorithm" [1]. Configuramos com os seguintes parâmetros:

- Número de clusters, k=2
- m =2
- Limitação de iteração, T=150
- $e=10^{-10}$
- Multiplicidade do conjunto de prototipo, q=2

Nosso código vai retornar varios resultados:

- A partiçao Fuzzy
- A partição Hard
- A lista de Medoids
- O Índice de Rand corrigido

Este algoritmo foi computado 100 vezes, no qual o resultado final foi referente a iteração que apresentasse o menor valor para a função objetivo do algoritmo.

O segundo classificador utilizado foi o classificador ***Bayesiano*** e o resultado retornado foram 100 erros dos testes do classificador.

Depois, usamos o método dos **k-vizinhos**, com diferentes k. Fizemos a execução para valores de k sendo 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 e 21 com a finalidade encontrar qual valor k gera menos erros ao classificar os padrões de teste.

Utilizamos também a **regra da soma** com para combinar o classificador Bayesiano e o melhor classificador do k-vizinhos.

No experimento do **MLP** utilizamos a seguinte partição: um conjunto de treinamento de 80%, um de validação (10%) e um de teste (10%). Rodamos só uma vez o k-fold para encontrar a melhor

configuração da rede, posto que variamos os valores de nós na camada escondida e a taxa de aprendizagem sendo eles 1, 10 ou 50 e 0.01, 0.1 e 0.9 respectivamente. No entanto a rede também contava com alguns parâmetros fixos como o a funçao de treinamento (gradiente descendente) e função de ativação tangente hiperbólica sigmoidal (configuração padrão do software utilizado).
Será apenas a melhor configuração (menor erro médio) que será comparada com os outros classificadores.

Durante a execução do **SVM** utilizamos diferentes funções de Kernel, podendo ser: linear, polinomial, funçao de base radial e sigmoide. Novamente, apenas a melhor configuração desse classificador vai será comparada com os outros classificadores.

## 4. Resultados

Na seção a seguir serão apresentados os resultados obtidos a partir do uso de diferentes classificadores no conjunto de dados TicTacToe. Para finalizar será apresentado a comparação entre os classificadores feita pelo método do teste de Friedman.

| Classificador | Erro Médio | Intervalo de confiança de 95% |
|---|---|---|
| Fuzzy | 0.464509[1] | - |
| Bayesiano | 0.245895 | (0.239354 <= μ <= 0.252435) = 95% |
| k-vizinhos (k=15) | 0.028421 | (0.016759 <= μ <= 0.040083) = 95% |
| Regra da soma | 0.102421 | (0.092963 <= μ <= 0.111879) = 95% |
| MLP | 0.075684 | (0.066773 <= μ <= 0.084595) = 95% |
| SVM - kernel de base radial | 0.123789 | (0.118528 <= μ <= 0.129050) = 95% |

Tabela 1: Erro médio e intervalo de confiança dos classificadores utilizados durante o experimento a partir de uso de validação cruzada 10x10 (com exceção do Fuzzy).

*4.1. Classificador Fuzzy*

| Resumo dos valores obtidos pelo classificador Fuzzy | |
|---|---|
| Erro global | 0.464509 |
| Erro da classe 1 | 0.436102 |
| Erro da classe 2 | 0.518072 |
| Índice de Rand Corrigido | 0.001540 |

Tabela 2: Resumo dos valores obtidos ao classificar o conjunto de dados TicTacToe pelo classificador fuzzy.

Como pode ser visto na tabela 1 o classificador fuzzy utilizado no experimento apresentou um erro global de 46,45% na classificação dos padrões do conjunto, sendo esta a maior taxa de erro encontrada durante o experimento e muito próximo de valores que seriam obtidos caso a amostra fosse classificada por um classificador que categoriza a amostra de forma aleatória. Vê-se também na tabela 2 que ambas as classes apresentam valores de erro de classificação alto, sendo a classe 2 que mais sofreu ao ser classificada, 51,80% dos seus padrões foram classificados erroneamente.
Além do erro global, ao fim da execução deste algoritmo de classificação obteve-se dois agrupamentos distintos, estes possuem como representantes os seguintes conjunto de medoids:

Classe 1: 124 e 687
Classe 2: 319 e 329

Outro valor importante encontrado ao fim da execução deste classificador foi o índice de Rand Corrigido [4], índice que calcula a dissimilaridade entre uma partição a priori e uma obtida por um algoritmo de classificação. Os valores obtidos a partir do cálculo deste índice estão definidos entre [-1,1], no qual os valores próximos de 1 indicam grande similaridade entre as partições, enquanto valores negativos ou mesmo próximos de zero indicam grande dissimilaridade. No caso do experimento o índice de Rand Corrigido foi 0.001540 o que reforça o valor obtido na taxa de erro de classificação.
Ao fim da computação do algoritmo fuzzy também foram obtidas a partição fuzzy (matriz U) e partição Hard (ver código em .c do projeto).

*4.2. Classificador Bayesiano*

O classificador Bayesiano apresentou um erro médio de 24.59% e sendo seu intervalo de confiança (0.239354 <= μ <= 0.252435) = 95%.

Classificador k-vizinhos

De acordo com a metodologia do experimento para determinar a melhor configuração do k-vizinhos para o conjunto de dados TicTacToe foram computadas diferentes números de vizinhos para ser usado como parâmetro de entrada do classificador como pode ser observado na tabela 3.

| Num. de vizinhos | Erro Médio | Num. de vizinhos | Erro Médio |
|---|---|---|---|
| 1 | 0.192526 | 13 | 0.031684 |
| 3 | 0.113158 | 15 | 0.028421 |
| 5 | 0.082 | 17 | 0.029789 |
| 7 | 0.058632 | 19 | 0.035158 |
| 9 | 0.047579 | 21 | 0.050842 |
| 11 | 0.038105 | | |

Tabela 3: Relação do número de vizinhos e o erro médio encontrado no conjunto TicTacToe com uma validação cruzada 10x10.

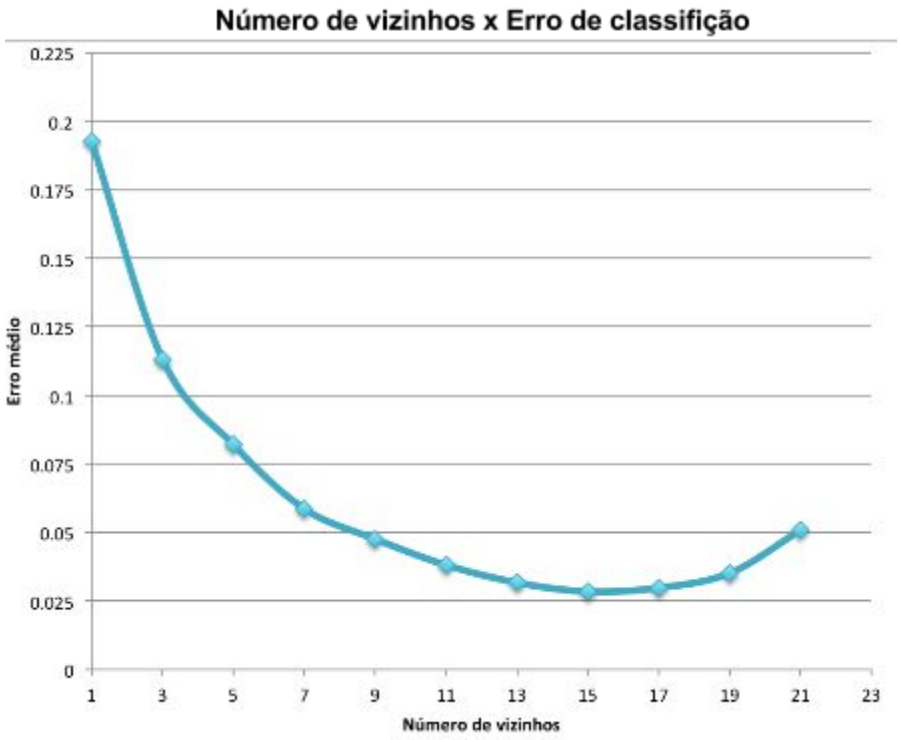


Figura 1: Comparação entre o número de vizinho e o erro médio obtido no conjunto TicTacToe em uma validação cruzada 10x10.

Como pode ser visto na figura 1 e tabela 3 a melhor configuração do classificador k-vizinhos é encontrada quando o número de vizinhos é igula a 15 , visto que antes e depois os valores do erro são maiores.

Determinou-se a configuração de k-vizinhos com 15 vizinhos para ser utilizada como o valor a ser comparado com os outros classificadores como pode ser visto na tabela 1, no qual o erro médio é 0.028421 e intervalo de confiança ($0.016759 <= \mu <= 0.040083$) = 95%.

*4.3. Classificador Regra da Soma (Bayesiano + k-vizinhos)*

Visto que este é um classificador obtido a partir da combinação de outros dois classificadores esperou-se durante uma melhora na classificação. Sendo esta melhora encontrada em relação ao Bayesiano, mas não ao k-vizinhos, visto que os valores do erro médio ficaram entre aos do Bayesiano e k-vizinhos como pode ser observado na tabela 1.

*4.4. Classificador MLP*

Foram testadas 9 configurações de MLP com a intenção de encontrar a melhor configuração da rede. Nos testes foram variadas a taxa de aprendizado e nós na camada escondida. Os erros médio encontrado com cada configuração estão apresentados na tabela 4.

| | | Taxa de aprendizagem | | |
|---|---|---|---|---|
| | | 0.01 | 0.1 | 0.9 |
| Nós | 1 | 0.355789 | 0.341053 | 0.347368 |
| | 10 | 0.185263 | 0.203158 | 0.231579 |
| | 50 | 0.104211 | 0.073684 | 0.310526 |

Tabela 4- Valores de erro de classificação dado a configuração de taxa de aprendizado e número de nós.

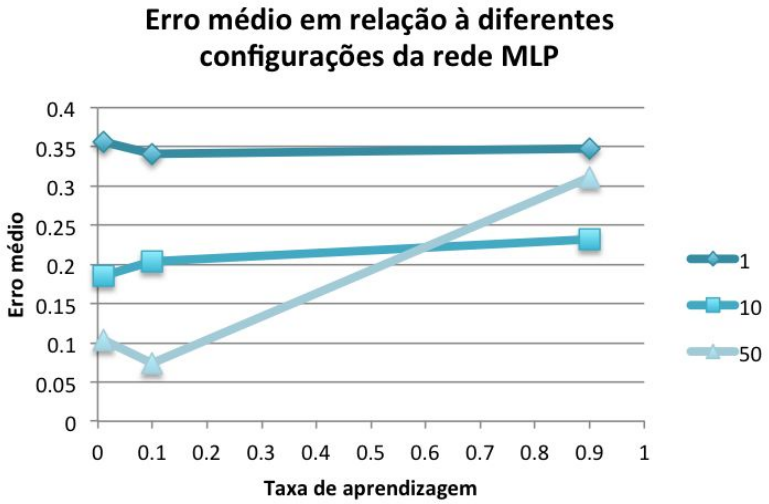


Figura 2 - Gráfico que compara as diferentes configurações do classificador MLP e seu o erro médio.

Como visto na tabela 4 a configuração que possui o menor erro é quando a rede MLP possui 50 nós na camada escondida e taxa de aprendizagem 0,01. Esta configuração foi escolhida para computarmos a taxa de erro do classificador MLP a partir da validação cruzada 10x10, o resultado pode ser visto na tabela 1. Logo a abaixo na tabela 5 também é apresentado a matriz de confusão ao executar o classificador a partir da validação cruzada 10x10.

| | | Classe Predita | |
|---|---|---|---|
| | | Classe 1 | Classe 2 |
| Classe verdadeira | Classe 1 | 6053 (63.71%) | 226 (2.39%) |
| | Classe 2 | 493(5.18%) | 2728 (28.72%) |

Tabela 5: Matriz de confusão dos valores obtidos a partir da validação cruzada 10x10.

Como pode ser visto na tabela 5 o erro de classificação do MLP foi de 8,61%, sendo a maioria deles referente a erro de classificação de padrões da classe 2 (5.65%).

*4.5. Classificador SVM*

Foram testadas quatro diferentes tipos de configuração para o classificador SVM, no qual a diferença entre eles é o tipo de kernel usado. Os resultados do erro médio e intervalo de confiança da classificação gerada a partir da validação cruzada 10x10 estão apresentados na tabela 6.

| Tipo de Kernel | Erro médio | Intervalo de confiança |
|---|---|---|
| Linear | 0.339052 | (0.337473 <= μ <= 0.340632) = 95% |
| Polinomial | 0.228632 | (0.223299 <= μ <= 0.233965) = 95% |
| Base radial | 0.123789 | (0.118528 <= μ <= 0.129050) = 95% |
| Sigmoidal | 0.340947 | (0.339175 <= μ <= 0.342719) = 95% |

Tabela 6: Erro médio gerado a partir do uso de diferentes tipos de kernel.

O resultado mais expressivo foi apresentado pelo SVM que utilizou o kernel de base radial no qual o erro foi de 12.38%.

*4.6. Comparação de classificadores*

A comparação entre os classificadores foi realizada com o teste de Friedman, visto que é um teste apropriado para a comparação de múltiplos classificadores [5]. A seguir são apresentados tabela de rank (tabela 7), sequência dos testes e resultados.

| | Bayesiano | K-vizinhos | Soma | MLP | SVM |
|---|---|---|---|---|---|
| | ranking | | | | |
| test 1 | 3 | 5 | 4 | 1 | 2 |
| test 2 | 5 | 1 | 3 | 2 | 4 |
| test 3 | 5 | 1 | 3 | 2 | 4 |
| test 4 | 5 | 1 | 2 | 4 | 3 |
| test 5 | 5 | 1 | 4 | 3 | 2 |
| test 6 | 5 | 1 | 2.5 | 2.5 | 4 |
| test 7 | 5 | 1 | 2 | 4 | 3 |
| test 8 | 5 | 1 | 2 | 4 | 3 |
| test 9 | 5 | 1 | 3 | 2 | 4 |
| test 10 | 5 | 1 | 3.5 | 2 | 3.5 |
| Média Rank | 4.8 | 1.4 | 2.9 | 2.65 | 3.25 |

Tabela 7: Tabela de rank dos classificadores em relação aos conjuntos de teste.

A partir dos ranks foi possivel calcular $x^2$ e $F_F$ para podermos saber se a hipotése nula, onde todos os classificadores são equivalentes, é falsa.

Os valores calculados são os seguintes:

$x^2 = 23.98$

$F_F = 5.992$

Logo devemos comparar o valor de $F_F$ com o valor de $F(4,36) \approx 2.61$ para $\alpha = 0.05$, como $2.61 < 5.992$ rejeitamos a hipótese nula. Passamos para o pós teste de irá determinar se dois classificadores são significativamente diferentes. Sendo o CD=1.929.

| | Bayesiano | K-vizinhos | Soma | MLP | SVM |
|---|---|---|---|---|---|
| Bayesiano | | | | | |
| K-vizinhos | | | | | |
| Soma | | | | | |
| MLP | | | | | |
| SVM | | | | | |

Figura 3: Figura que compara os classificadores, os pontos marcados em azul significam que os classificadores são significativamente diferentes em relação ao outro.

A partir disso detectamos que o classificador K-vizinhos e MLP tiveram um desempenho superior ao classificador Bayesiano. Quanto aos outros classificadores nada se pode afirmar.

## 5. Discussão

Nesta seção serão discutidos os principais resultados obtidos ao aplicar-se diferentes classificadores no conjunto de padrões TicTacToe. Além disso, discutiremos outros temas como características interessantes de alguns classificadores, comparação de resultados com trabalhos similares e limitações do nosso estudo.

### *5.1. Interpretação geral sobre os classificadores*

Ao analisar os resultados obtidos é possível fazer uma comparação com outros estudos, visto que dados sobre a performance de classificadores sobre a base TicTacToe pode ser encontrado na literatura [6-8]. A partir disso conseguimos uma métrica para verificar se os classificadores utilizados no experimento produzem valores condizentes com a realidade. Foi interessante notar que o classificador que apresentou melhor performance, k-vizinhos, apresenta valores performance de classificação próximos a um classificador similar utilizado por Aha [6] em seu estudo. Outro resultado bastante significativo foi o resultado obtido pelo classificador de agrupamento fuzzy utilizado no experimento. O resultado deste classificador foi bastante pobre, chegando o seu erro ser próximo a de modelos que fazem a previsão de forma aleatória.

No final, viu-se que os classificadores que apresentaram melhor performance são k-vizinhos e a rede MLP. Acredita-se que isso seja devido à distribuição dos atributos dos dados, além do fato que para os dois métodos, diferentes tipo de configurações de cada classificador foram testadas, como visto na metodologia, o que possibilitou um refinamento do classificador utilizado.

*5.2. Características interessantes de certos classificadores*

Entre os classificadores que apresentaram características interessantes a serem discutidas neste estudo estão o Fuzzy, MLP, SVM.

O classificador Fuzzy surpreendentemente apresentou uma performance muito ruim, similar a de um classificador aleatório, visto que o erro global de classificação é extremamente grande, assim como o erro de classificação da classe 1 e classe 2 (ver tabela 2). O índice de Rand (ver tabela 2) calculado após a classificação só reforça a ideia que a classificação apresenta graus de aleatoriedade. Acreditamos que isso ocorre devido o valor da distância utilizada durante a execução do algoritmo que não possibilita a detecção de agrupamentos, não permitindo assim que o algoritmo os classifique.

Já o algoritmo MLP é bastante interessante, pois durante o estudo testamos seis diferentes configurações e podemos observar o impacto da variação da taxa de aprendizagem e o número de camadas escondidas. Como pode ser visto na figura 2 quanto maior o número de nós na camada escondida menor o erro de classificação. No entanto, vale notar que o tempo de computação aumentou tremendamente. O mesmo ocorre quando a taxa de aprendizagem é pequena, visto que durante o treinamento o classificador levou bastante tempo para convergir a um resultado, apesar de apresentar um erro de classificação melhor. Este fato é tão notável que muitas vezes o classificador parou de realizar o treinamento devido a ter atingido o número máximo de épocas permitidas, que no nosso caso foi de 50.000. Outro fato interessante do MLP é a escolha da função de ativação e algoritmo de treinamento, que podem podem afetar profundamente a performance e tempo de execução do classificador. No nosso caso optamos pelo uso do gradiente descendente e o uso da função tangente hiperbólica sigmoide nos nós da camada escondida e uma função linear na camada de saída. Entretanto, observamos em testes não formais que o uso de outras configurações, como o uso da função sigmoid logística, apresentavam valores bem distintos dos encontrado neste experimento.

A configuração da SVM também foi um dos fatores que a tornou intrigante o seu estudo como pode ser visto na tabela 6 onde quatro diferentes tipos de kernel foram testados e entre sí apresentando valores bastantes distintos. A equipe acredita que melhores estimativas de classificação poderiam ter sido alcançadas caso tivéssemos também investigado mais a fundo os parâmetros dos diferentes kernels.

*5.3. Limitações do estudo*

Apesar de durante o estudo encontrarmos classificadores que apresentaram desempenho notável, encontramos algumas limitações no estudo. A principal seria que os classificadores implementados somente foram testados sobre um único conjunto de padrões, o que impede uma avaliação mais profunda sobre cada classificador, visto seria interessante, por exemplo, avaliar os classificadores sobre um contexto no qual os objetos não possuem atributos de valor discreto. Outra limitação ainda refere-se a base de dados, pois esta se encontra desbalanceada, logo não sabemos qual a seria a

perfomance dos classificadores caso a base estivesse balanceada, muito provalvementes classificadores como o MLP que são muito sensíveis a tal falha teriam melhor performance, posto que o erro de classicação atual na classe minoritária foi bem superior que a majoritária (ver tabela 5). Outra limitação seria o fato que classificadores como o MLP ficarem restritos a serem experimentados sobre um número restrito de iterações, visto que o poder computacional disponível era restrito durante o experimento, logo não podemos afirmar com toda certeza qual o real poder do classificador sobre determinada configuração. Todas essas limitações servem como possibilidades para trabalhos futuros que visem comparar classificadores sobre tal base de dados.

## 6. Conclusão

Ao fim do estudo a equipe fez uso de seis diferentes classificadores sobre o conjunto TicTacToe, conseguindo alcançar valores de classificação próximos ao da literatura [6-8].
O time responsável pela execução do estudo acredita que a maior impotância deste experimento esteja em sua característica explorativa que possibilita que o aluno conheça diferentes tipos de classificadores e planeje diferentes tipos de experimento, servindo assim como um bom preparo para futuro estudos na área de aprendizagem de máquina.

## 7. Bibliografia

[1] F. A. T. de Carvalho et al., “Relational Partitioning Fuzzy Clustering Algorithms Based on Multiple Dissimilarity Matrices”. Fuzzy Sets and Systems, v. 215, p. 1-28, 2013.

[2] T. M. Mitchell, *Machine learning*. Burr Ridge, IL: McGraw Hill, 1997.

[3] A. P. Braga et al., *Redes neurais artificiais: teoria e aplicações*. 2 ed. Rio de Janeiro: LTC, 2007.

[4] L. Hubert and P. Arabie, "Comparing partitions." *Journal of classification* 2, no. 1, 1985.

[5] J. Demšar, "Statistical comparisons of classifiers over multiple data sets."*The Journal of Machine Learning Research* 7, 2006, pp. 1-30.

[6] D. W. Aha, "Incremental constructive induction: An instance-based approach." In *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 117-121. 1991.

[7] C. J. Matheus and L. A. Rendell. "Constructive Induction On Decision Trees." In *IJCAI*, vol. 89, pp. 645-650, 1989.

[8] C. J. Matheus, "Adding Domain Knowledge to SBL Through Feature Construction." In *AAAI*, pp. 803-808, 1990.