CLASSIFICAÇÃO DE DADOS USANDO LÓGICA FUZZY E ALGORITMOS EVOLUTIVOS

Josias Alexandre Oliveira*
Departmento de Informática
Universidade Federal do Espírito Santo
Vitória, Brasil
josiasalexandre@gmail.com

Vinicius Ferraço Arruda†
Departmento de Informática
Universidade Federal do Espírito Santo
Vitória, Brasil
viniciusferracoarruda@gmail.com

14 de Novembro de 2019

1 Método

O método utilizado[1] compreende um classificador *fuzzy* no qual as funções de pertinência são otimizadas por algoritmos evolutivos. Neste trabalho, os algoritmos evolutivos genético (GA) e o de otimização por enxame de partículas (PSO) foram utilizadas nas experimentações.

A tarefa consiste em classificar o conjunto de dados *IRIS* com base em regras definidas por um especialista. As regras fuzzy utilizadas são as mesmas do trabalho [1]. Cada característica do conjunto de dados é convertida para variáveis *fuzzy*. Para tanto, os dados foram normalizados para valores no intervalo $[0, 1]$ e as funções de pertinência foram divididas em três partes. Uma dada característica pode ser considerada como *short*, *middle* ou *long*. O intervalo é então dividido conforme a Figura 1. O objetivo dos algoritmos evolutivos é encontrar o melhor parâmetro w para as funções de pertinência relacionadas a cada uma das características.

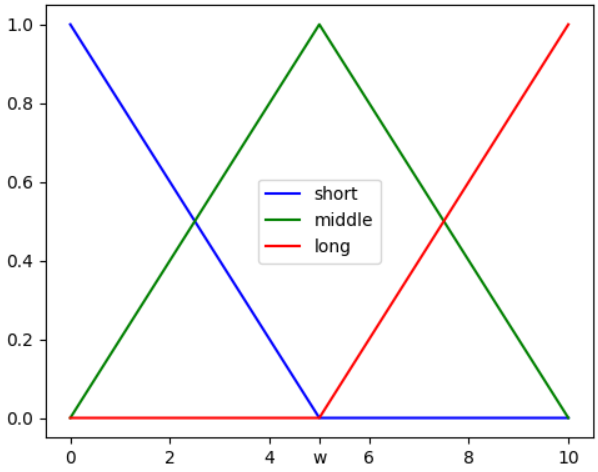

Figura 1: Função de pertinência para cada feature

*github.com/josiasalexandre
†viniciusarruda.github.io
[1]Código disponível em https://github.com/viniciusarruda/ga-pso-fuzzy-classifier

1.1 Classificador

O objetivo do classificador *fuzzy* é determinar a classe de cada amostra no conjunto de dados *IRIS*. O primeiro passo é determinar as variáveis linguísticas a serem utilizadas, uma vez que os dados originais possuem valores numéricos. Para facilitar a conversão, os atributos ou características dos dados foram normalizados conforme Equação 1:

$$x_i = \frac{x_i - min(i)}{max(i) - min(i)} \tag{1}$$

Conforme exibido na figura Figura 1, o intervalo normalizado é dividido em três partes para corresponder às variáveis linguística *short*, *middle* ou *long*. O único parâmetro utilizado para definir essa divisão é um valor w que definirá o término da função de pertinência relacionado a categoria *short*, bem como a posição central ou máxima da função de pertinência da categoria *middle* e o início da função de pertinência da categoria *long*. Precisamente, as categorias são definidas como:

$$\mu_{Short}(x) = \begin{cases} 0; & 0 < x \\ \left(1 - \frac{x}{w}\right); & 0 \leq x \leq w \\ 0; & x > w \end{cases} \tag{2}$$

$$\mu_{Middle}(x) = \begin{cases} 0; & x < 0 \\ \frac{x}{w}; & 0 \leq x \leq w \\ \left(\left(1 + \frac{w}{1-w}\right) - \frac{x}{1-w}\right); & w < x \leq 1 \\ 0; & x > 1 \end{cases} \tag{3}$$

$$\mu_{Long}(x) = \begin{cases} 0; & x < w \\ \left(\frac{x}{1-w}\right) - \left(\left(\frac{1}{1-w}\right) - 1\right); & w \leq x \leq 1 \\ 0; & x > w \end{cases} \tag{4}$$

As funções de pertinência são aplicadas em cada uma das características dos dados. Em posse dos valores de pertinências, o classificador utiliza as seguintes regras de decisão:

R_1: $((x_1 = short \vee long) \wedge (x_2 = middle \vee long) \wedge (x_3 = middle \vee long) \wedge (x_4 = middle))$
$\vee ((x_1 = middle) \wedge (x_2 = short \vee middle) \wedge (x_3 = short) \wedge (x_4 = long)) \rightarrow versicolor$

R_2: $(x_3 = short \vee middle) \wedge (x_4 = short) \rightarrow setosa$

R_3: $(x_2 = short \vee middle) \wedge (x_3 = long) \wedge (x_4 = long) \rightarrow virginica$

Nesse contexto, x_1 é o comprimento da sépala, x_2 é a altura da sépala, x_3 é o comprimento da pétala e x_4 é a altura da pétala.

Após a avaliação da regras R_1, R_2 e R_3, cada amostra possuirá uma grau de pertinência em cada uma das três classes. Para realizar a decisão ou classificação final, consideramos o maior grau de pertinência.

1.2 Algoritmo Genético (GA)

Neste trabalho foi implementado o algoritmo genético na seguinte configuração:

- Cromossomo definido como um vetor com 4 parâmetros contínuos, representando cada w.
- Inicialização do indivíduo aleatoriamente com distribuição uniforme.
- Uso da técnica de seleção por torneio com $s = 2$.
- Taxa de cruzamento de 0.9 com recombinação em dois pontos aleatórios.
- O algoritmo BLX-α foi usado para fazer *crossover* dos indivíduos selecionados para reprodução.
- Taxa de mutação de 0.1, inicializando uma posição aleatória do cromossomo com um valor aleatório de uma distribuição uniforme no intervalo $[0, 1]$.
- 10 indivíduos.
- 30 épocas.

1.3 Otimização por Enxame de Partículas (PSO)

Neste trabalho foi utilizado uma implementação pública[2] do algoritmo de otimização por enxame de partículas que segue a seguinte configuração:

- Cada partícula possui 4 dimensões, todas sendo parâmetros contínuos, representando cada w.
- Inicialização de todas as dimensões com a constante 0.5.
- Limites de busca das partículas no intervalo $[0, 1]$ para todas as dimensões.
- Uso da topologia global.
- Constante de inércia $w = 0.5$.
- Constantes $C1$ e $C2$ iguais a 1.
- Caso a partícula ultrapassar os limites, estas serão truncadas ao limite.
- 10 partículas.
- 30 iterações.

2 Experimentos

O experimento consistiu em comparar o desempenho dos algoritmos evolutivos em otimizar os parâmetros das regras do classificador *fuzzy*. Para o comparativo foi executado cada algoritmo por 30 vezes.

Para ambos algoritmos evolutivos utilizados, a função de custo a ser minimizada foi o erro do classificador, dado pela porcentagem do número de amostras que foram classificadas erroneamente. De modo semelhante ao trabalho [1], todas as 150 amostras do conjunto de dados *IRIS* foram utilizadas.

Os resultados foram reportados em acurácia, isto é, porcentagem do número de amostras que foram classificadas corretamente.

3 Resultados

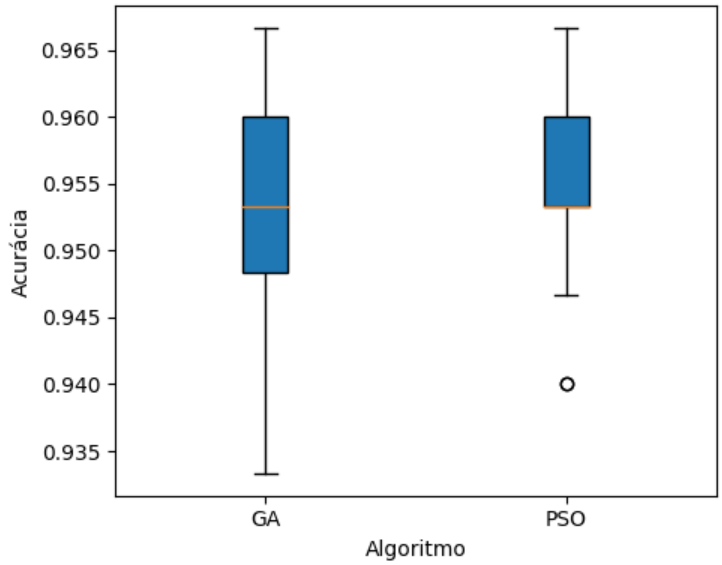

Figura 2: Boxplot da acurácia das 30 execuções dos algoritmos.

A Figura 2 mostra os boxplots da acurácia das 30 execuções de cada algoritmo evolutivo. O algoritmo PSO se mostrou ligeiramente mais robusto, mas ambos mostraram encontrar bons parâmetros.

[2] `https://github.com/nathanrooy/particle-swarm-optimization`

Das 30 execuções, o GA obteve 96.67% e 93.33% como maior e menor acurácia, com média e desvio padrão $95.40\% \pm 0.0776\%$. Já o PSO obteve o mesmo valor de acurácia máxima que o GA, de 96.67%, e um maior valor mínimo, que foi de 94.0%, com média e desvio padrão $95.58\% \pm 0.0655\%$. A média ligeiramente maior do PSO sobre o GA com um menor desvio padrão confirma que o PSO obteve resultados mais robustos, porém, ambos se mostraram capazes em encontrar os mesmos valores máximos de acurácia.

Referências

[1] Stig-Åke Svensson. Implementing a fuzzy classifier and improving its accuracy using genetic algorithms.