Evolução de um Classificador Binário para Deteção de Problemas Cardíacos

Computação Evolucionária

Ana Oliveira, Diogo Pessoa, Nelson Monteiro

Maio de 2018

Resumo

A área da aprendizagem computacional, nomeadamente da classificação, é uma área muito abrangente, com aplicabilidade nos mais diversos campos como, por exemplo, no da saúde. No entanto, torna-se difícil definir o melhor conjunto de parâmetros a aplicar ao classificador pelo facto de existir um imenso número de combinações possíveis. Neste trabalho iremos apresentar a solução que desenhámos para resolver este problema.

Keywords: Algoritmos Evolucionários; Redes Neuronais; *Machine Learning.*

1 Introdução

Os algoritmos evolucionários são inspirados, em grande parte, pela evolução natural das populações na natureza, de acordo com a Teoria evolucionária das espécies de Darwin. De uma forma geral, esta teoria afirma que, ao longo do tempo, o ambiente onde se encontra uma dada população sofre alterações pelo que, indivíduos que possuem caraterísticas que lhes conferem vantagens no novo ambiente, irão reproduzir-se com maior facilidade, perpectuando as suas características por toda a população. Com o passar das gerações será possível observar um predomínio crescente dos indivíduos com as melhores características (*Survival of the fittest*). É de notar que estas caraterísticas, que conferem vantagem evolutiva aos indivíduos, têm origem na maioria dos casos em mutações aleatórias no código genético, pelo que são imprevisíveis e apenas passíveis de observar se a população for monitorizada por um longo período de tempo.

Fazendo a analogia para problemas computacionais, os princípios a aplicar serão os mesmos. Irá existir na mesma uma população que irá evoluir ao longo do tempo e nessa população irão ser impressos vários operadores de variabilidade, como o crossover, que partindo de dois indivíduos, parents, se vão cruzar entre si dando origem a um novo indivíduo diferente, e também operadores de mutação que irão provocar alterações ao nível genótipo. De geração em geração, a população irá ser avaliada e apenas os indivíduos com maior fitness se irão reproduzir, e eventualmente dando origem a *offsprings* com características melhores. Para além do mais, os algoritmos evolucionários são passíveis de sofrer alterações no meio ambiente ao longo do tempo, no entanto estas mudanças não são contempladas no contexto deste trabalho.

Tendo em vista o principal objetivo do trabalho, a evolução de um classificador binário, foi delineada uma estratégia que nos permitisse evoluir a qualidade de um dado classificador. Perante a necessidade de construir classificadores eficientes, torna-se imperativo a aplicação de metodologias evolutivas que sejam capazes de identificar e avaliar o melhor conjunto de parâmetros, visto que existem inúmeras combinações possíveis para os mesmos. Face à relevância, complexidade e atualidade das *Artificial Neural Networks*, foi interessante escolher estas como classificador. Os *weights* das *Artificial Neural Networks* permitem atribuir a influência/importância que cada neurónio tem e, como tal, é necessário encontrar a combinação mais promissora dos mesmos que permita levar à melhor *accuracy*. Assim, foi evoluído um conjunto inicial de pesos de uma dada rede de forma a obter a melhor performance classificativa.

Para além disso, foram também estudados dois operadores de crossover e de mutação diferentes, o que resultou num número de configurações possíveis para o algortimo evolucionário igual a 4, e de que forma condicionavam estes a evolução.

2 Metodologia/Abordagem

De forma a perceber qual o tipo de classificador mais apropriado a utilizar no contexto de um problema evolucionário, foram ponderados diversos classificadores e pensado de que forma estes poderiam ser utilizados num algoritmo deste tipo. Desta forma, chegou-se à conclusão de que uma rede neuronal seria o classificador mais apropriado ao contexto, dado que é uma representação artificial e com capacidades evolutivas de um sistema real (sistema nervoso).

Após ter sido fixado o classificador, foi analisado de que forma este poderia evoluir e melhorar a sua performance. Estando perante uma rede neuronal, foram identificados dois caminhos possíveis:

1. **Evolução de um vetor de pesos inicial** (fixação da topologia).
2. **Variação da topologia da rede**: número de camadas, número de neurónios por camada e função de ativação.

Tendo em conta a complexidade computacional e a dimensionalidade de atuação dos mecanismos de variação sobre as diferentes combinações de topologia da rede, foi escolhido o primeiro caminho. Apesar deste não ter em conta a variação de topologia, houve a necessidade de uma fase de experimentação com o intuito de identificar uma combinação que levasse a um resultado razoável em termos de *accuracy*.

Com a topologia escolhida, o classificador foi enquadrado no algoritmo evolucionário, através da organização dos diferentes pesos de todas as camadas num único vetor a ser evoluído (indivíduo) e aplicação de diferentes mecanismos de variação, nomeadamente, *crossover* e mutação.

Por último, os resultados obtidos, para cada uma das combinações de operadores de variação, foram analisados do ponto de vista estatístico.

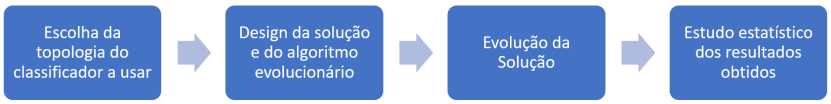


Figura 1: *Overview* da metodologia utilizada.

2.1 Rede Neuronal

A *Neural Network Artificial (ANN)* é um modelo computacional (classificador) que pretende representar o cérebro e, como tal, é composto por vários elementos de processamento designados de neurónios artificiais.

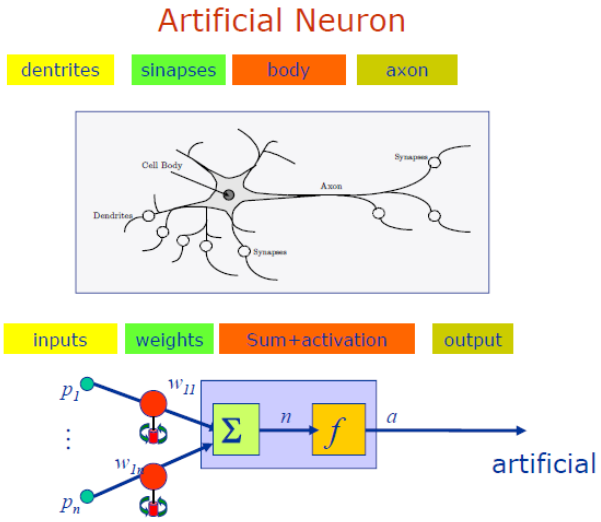
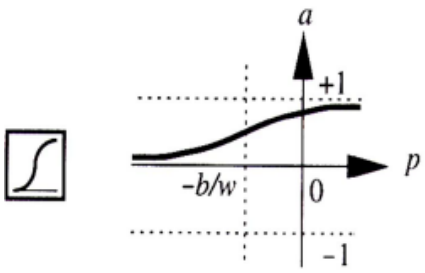


Figura 2: Esquematização de um neurónio artificial.

Cada neurónio artificial é composto por:

- **Inputs** - representam cada variável independente que vai ter influência sobre o output da *Neural Network*;
- **Pesos** : atribuídos a cada *input* e representam a influência/importância de cada um deles;
- **Bias**: serve como um "neurónio" extra que é adicionada em cada camada pré-output, permitindo uma melhor representação dos dados, através da movimentação lateral da função de ativação.
- **Função de ativação**: utilizada para estabelecer limites aos valores obtidos do somatório e decidir se esse "neurónio" deve ser ativado ou não. Tornam-se muito relevantes no processamento de dados. No contexto do problema, a função de ativação escolhida foi a ***sigmoid unipolar***, que apresenta como *output* valores compreendidos entre 0 e 1.



$$a = logsig(wp + b)$$

$$a = \frac{1}{1 + e^{-(wp+b)}}$$

Figura 3: Gráfico de uma função de ativação do tipo *logsig*.

- **Output**: o número de "neurónios" de output deve estar diretamente relacionado com o tipo de trabalho que a *Neural Network* vai desempenhar.

As *Neural Networks*, para além dos seus elementos estruturais, os neurónios artificiais, são ainda caracterizadas pela presença de *hidden layers*, função de treino/aprendizagem e arquitetura. As *hidden layers* funcionam como camadas intermédias entre o *input* e o *output* e são formadas por conjuntos de "neurónios". A cada uma destas camadas está associada uma função de transferência/ativação e a sua importância resume-se à capacidade que têm em transformar os *inputs* em algo que possa ser utilizado pela camada de *output*, através da otimização da informação.

Relativamente ao supramencionado, não existe nenhum método específico para determinar qual a melhor combinação do número de camadas e do número de neurónios que as constituem. No entanto, a escolha deve ser baseada na utilização de, por norma, 1-2 camadas (mais camadas muito raramente resultam em melhores resultados) e o número de neurónios deve ser adequado ao ponto de não ocorrer *over* ou *underfitting*.

A função de treino caracteriza-se como o algoritmo que é usado para treinar a Neural Network, de forma a que esta reconheça um determinado input e o transforme em output.

Por último, ainda existe a necessidade de definir qual a arquitetura a ser utilizada na construção das *Neural Networks*. No caso do trabalho desenvolvido, foi escolhida a *feedforward* com ausência de *backpropagation* para garantir que os pesos iniciais são o fator de maior preponderância na performance da rede, devido ao objetivo primordial se centrar na evolução dos pesos iniciais com base em algoritmos evolutivos.

Na *feedforward neural network*, a informação "move-se" apenas numa direção (*Input-Hidden Layer-Output*), caracterizando-se também pela ausência de ciclos.

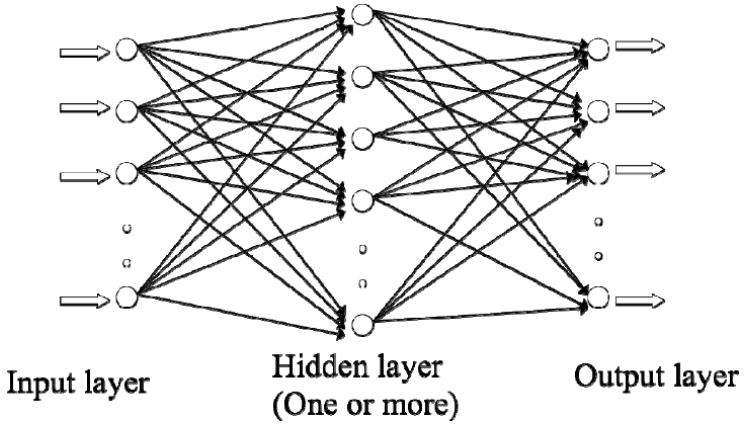


Figura 4: Esquema de rede neuronal *fully-connected feedforward* Retirado de https://www.researchgate.net/figure/A-Multi-Layer-Fully-Connected-Feedforward-Network-20_fig9_224161450?_sg=5B-y2yCXaU33fCPTMTFfreeNk91II6mY7F5AXYO2mCkQlf-gk95GW7-30scWkOUM3TxqLKDPlYR1qFH5FCW-bA.

Na Tabela 1 encontram-se os parâmetros utilizados na rede neuronal escolhida.

Parâmetros	Valores
# neurónios camada interna 1	15
# neurónios camada interna 2	15
# camada de input	13
# camada de output	2
Função de ativação	sigmoidal
Batch Size	3

Tabela 1: Parâmetros da rede neuronal.

2.2 Algoritmo Evolucionário

Um algoritmo evolucionário tem por base a seleção de determinados indivíduos ao longo de gerações, pela seleção natural dos genes que fornecem ao individuo as características (isto é, o fenótipo) que melhor se adequa ao ambiente onde se encontra, permitindo a sua sobrevivência. Desta forma, após a geração da população de soluções que o algoritmo pretende resolver, ocorre seleção e posterior reprodução de alguns pares de soluções, os *parents*, através de algoritmos de variação, isto é, *crossover* e mutação. Estes permitem que as populações possam variar ao longo das gerações, quer através de trocas entre pares de cromossomas (*crossover*), quer através de modificações únicas (mutações), gerando *offsprings*. Após este processo, ocorre a avaliação da *fitness* da nova população, isto é, a avaliação da qualidade das soluções tendo em conta o contexto do problema e, ainda, a substituição das piores soluções da população atual pelos melhores *parents*. Após um determinado número de gerações, é selecionada a melhor solução para o problema.

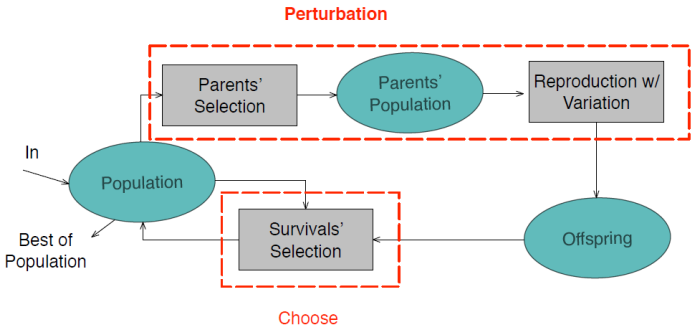


Figura 5: Arquitetura geral de um algoritmo evolucionário.

2.3 Representação da Solução

Como mencionado anteriormente, neste trabalho o objetivo foi evoluir um conjunto de pesos iniciais de uma rede neuronal. Assim, a representação da solução para o problema irá ser um vector de *floats* correspondente ao valor dos pesos das ligações entre todos os neurónios das várias camadas. Visto que a rede escolhida tem 15 neurónios em cada uma das duas camadas internas, o número total de pesos será igual a $13*15+15*15+15*1=435$. É importante ainda explicar que o valor 13 corresponde ao número de *features* que existem e o valor 1 ao *output*.

2.4 *Fitness*

Um aspeto fulcral do ponto de vista de um algoritmo evolucionário, é a avaliação dos indivíduos da população, para que seja possível efetuar comparações entre os mesmos. Dado o problema ser de classificação, a medida de *performance* mais comum a utilizar é a *accuracy*, que avalia o número de classificações corretas em relação ao tamanho total de *samples*. Apesar de ser uma métrica que oferece uma boa visão geral acerca da *performance* do classificador, em determinados casos pode ser ilusória, nomeadamente quando as classes são extremamente desiquilibradas relativamente ao número de amostras. Nestes casos a *accuracy* pode ser elevada, mas os valores de sensibilidade ou de especificidade serem extremamente baixos. No seguimento de não existir qualquer restrição aplicada na escolha de um classificador que seja melhor a detetar uma classe relativamente à outra e o número de amostras de cada classe ser aproximadamente igual, foi apenas utilizada a *accuracy*. Desta forma, a *fitness* de cada indivíduo é dada pela percentagem de observações classificadas corretamente.

2.5 Operadores

2.5.1 Variabilidade

De forma a perceber o grau de variabilidade da população de indíviduos gerados aleatoriamente em termos de *fitness*, foram gerados 3000 indivíduos e avaliados, sendo que a distribuição dos mesmos se encontra na figura 6. Através da análise da distribuição, torna-se percetível que a maioria dos indivíduos se

encontra em torno do valor 50% de *fitness*. Para além do mais, frisando que todos os indivíduos foram gerados aleatoriamente e são todos diferentes entre si, existem apenas 35 valores de *fitness* diferentes. Deste modo, existem inúmeros vetores de pesos iniciais que se traduzem no mesmo valor de *accuracy* da rede e, como tal, para se conseguir obter indivíduos com valores de *fitness* diferentes, tantos os métodos de variação como as suas probabilidades tem uma influência bastante significativa, pelo que, são necessárias probabilidades elevadas.

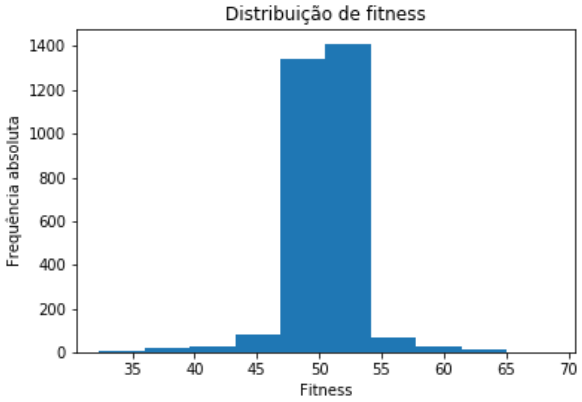


Figura 6: Distribuição dos valores de fitness de 3000 indivíduos gerados aleatoriamente.

Tendo em vista as considerações tecidas anteriormente, foram escolhidos dois operadores de *crossover* e outros dois de mutação considerados como adequados ao problema, nomeadamente: *Uniform* e *Two-Point Crossover*, e *Permutation Scramble Mutation* e *Float Mutation*.

- **Uniform *Crossover***: Após a seleção de dois indivíduos, a cada posição do cromossoma é atribuida uma igual probabilidade de provir do parent 1 ou do parent 2, pelo que no outro cromossoma ficará com o gene proveniente do parent não escolhido inicialmente.
- **Two-Point *Crossover***: Estabelecidas duas posições ao longo do cromossoma, (idêntica para ambos os parents), é posteriormente feita a troca entre as secções, obtendo um dos indivíduos a parte correspondentes entre essas duas posições proveniente do outro indivíduo e vice-versa.
- **Permutation Scramble Mutation**: Tendo definido duas posições ao longo do cromossoma, reordena-se os valores entre as mesmas, obtendo-se um novo indivíduo.
- **Float Mutation**: Para uma determinada proporção do cromossoma (por exemplo, 20%), gera-se valores aleatoriamente dentro de um domínio pre-estabelecido (no nosso caso, [-1;1]) e substitui-se os valores correspondentes a essa proporção (posições aleatórias) pelos gerados.

Listing 1: Função custom float mutation

```
def muta_float_custom(indiv, prob_muta):
    cromo=np.array(indiv,dtype='float64')
    value = random()
    if value < prob_muta:
        index = sample(range(len(indiv)),round(np.size(indiv)*0.2))
        index.sort()
        for i in range(np.size(index)):
            j=index[i]
            cromo[j]=(r.uniform(-1,1))
    return cromo
```

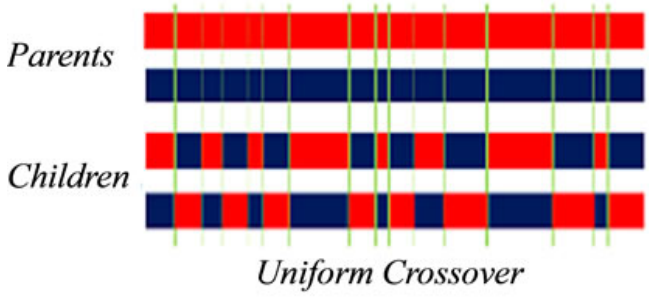


Figura 7: Esquema do método uniform crossover.

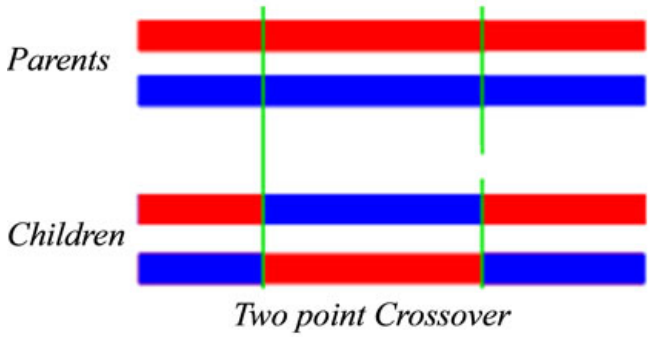


Figura 8: Esquema do método two point crossover.

2.5.2 Seleção

Através de métodos de seleção de indivíduos ao longo das gerações, é possível garantir que as melhores soluções não se perdem e/ou são recombinadas, criando assim uma pressão seletiva nos indivíduos. Assim, da população de *parents* são selecionados aqueles aos quais corresponde uma melhor *fitness* para processos de variação, como o *crossover* e a mutação de modo a criar novos *offsprings*. Posteriormente, os melhores *offsprings*, selecionados após nova avaliação, são mantidos na população de indivíduos correspondentes à geração seguinte. (Tendo

em vista as considerações tecidas anteriormente, foram escolhidos dois operadores de *seleção* considerados como adequados ao problema, nomeadamente: *Tournament Selection* e *Elitism*.)

- **Tournament Selection**: Após escolhido a quantidade de *parents* que podem ser utilizados para processos de variabilidade, é feita a ordenação da população de individuos de acordo com o valor de *accuracy* dos mesmos, e posterior seleção dos melhores tendo em conta o número anteriormente definido.

- *Elitism*: Após seleção da percentagem de **parents** a manter para a geração seguinte, e reordenação da mesma e da população de *offsprings*, ocorre a substituição dos *offsprings* com pior avaliação pelos *parents* com melhor avaliação.

2.6 Estatística

A estatística e os seus métodos derivados permitem inferir conclusões relativas aos resultados obtidos. A aplicação deste conjunto de processos torna-se imperativa dado que é necessário avaliar a qualidade da nossa solução, assim como, a eficiência do nosso algoritmo. É ainda importante referir que, para aplicar testes estatísticos e validar/generalizar as soluções, é necessário ter um conjunto de amostras, ou seja, foi necessário definir um número de *runs* para obter vários valores de *fitness* para cada uma das combinações de operadores de variação (conjunto de soluções). Deste modo, recorreu-se à estatística descritiva e à estatística inferencial.

2.6.1 Estatística Descritiva

A estatística descritiva tem por base atribuír características às nossas soluções e, como tal, procedeu-se ao cálculo da centralidade(média,moda e mediana), dispersão(desvio padrão e variância), forma da distribuição (*skewness* e *kurtosis*), valor máximo e mínimo, boxplot com cálculo dos percentis e ainda histograma.

2.6.2 Estatística Inferencial

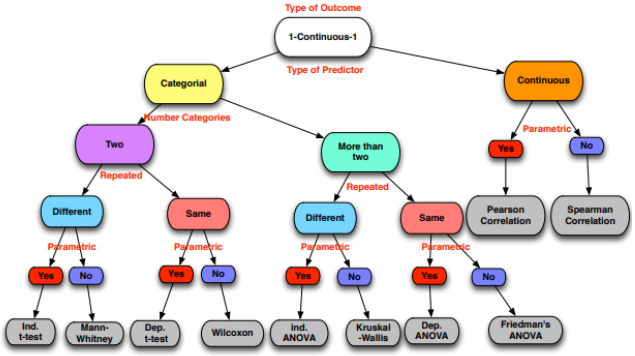


Figura 9: Esquema de testes estatísticos.

A estatística inferencial permite retirar conclusões sobre a distribuição e importância estatística das nossas soluções através da confirmação ou rejeição da hipótese nula. Dada as nossas soluções serem do tipo categórico e em cada *run* ser utilizada a mesma população (*matched*), foram aplicados os seguintes testes:

- ***Kolgomorov-Smirnov*** - tem como hipótese nula duas distribuições serem iguais. É utilizado para verificar se a solução apresenta uma distribuição normal.
- ***Shapiro-Wilk*** - tem como hipótese nula os dados seguirem uma distribuição normal.
- ***Friedman's ANOVA*** - tem como hipótese nula medições repetidas dos mesmos indivíduos terem a mesma distribuição.
- ***Wilcoxon*** - tem como hipótese nula duas medições repetidas dos mesmos indivíduos terem a mesma distribuição.

É importante ainda referir que os dois últimos testes estatísticos são não paramétricos, ou seja, métodos utilizados para o caso dos dados não seguirem uma distribuição normal.

3 Resultados e Discussão

Como mencionado anteriormente, foi definido um número de repetições (30), com o objetivo dos resultados obtidos, para cada configuração, serem estatisticamente significativos e um número de gerações, de modo a avaliar a evolução de cada uma das populações iniciais. Para além do mais, torna-se imperativo a utilização da mesma população inicial, para cada uma das *runs* das diferentes combinações, no sentido da avaliação das diferentes soluções apresentadas ser efeita sob as mesmas condições iniciais.

Parâmetros	Valores
Número de gerações	50
Número de runs	30
Tamanho da população	100
Probabilidade de mutação	0.6
Probabilidade de crossover	0.8
Percentagem da elite	10%
Tamanho do torneio	5

Tabela 2: Parâmetros do algoritmo evolucionário.

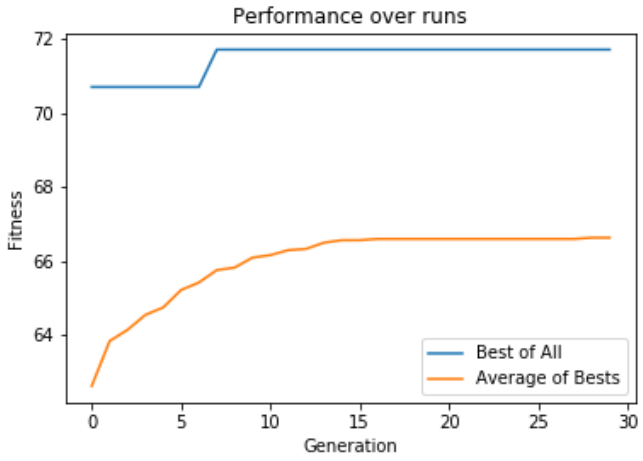


Figura 10: Configuração com Uniform Crossover e *Custom Mutation*.

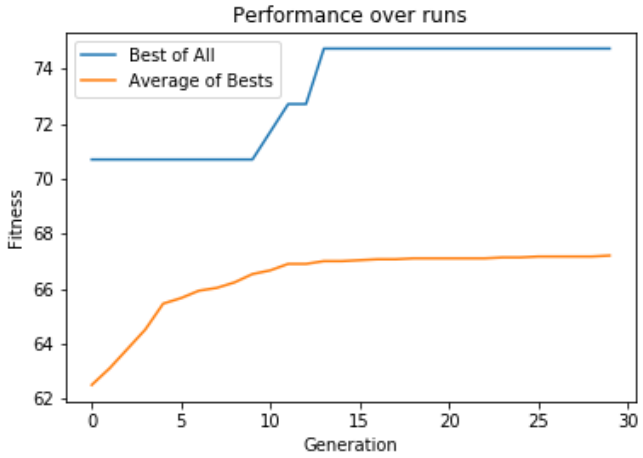


Figura 11: Configuração com *Uniform Crossover* e *Scramble Mutation*.

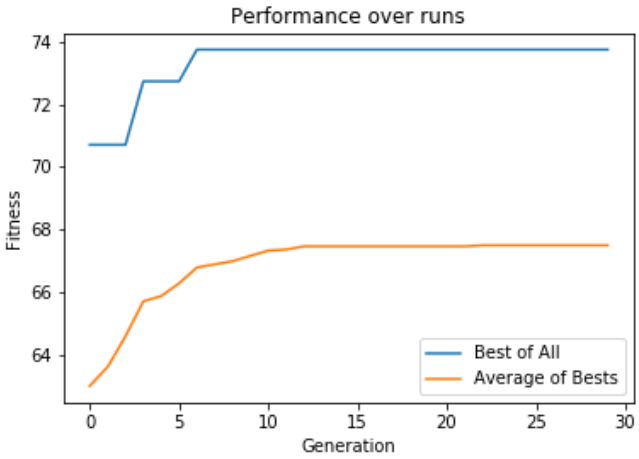


Figura 12: Configuração com Two-Point Crossover e *Custom Mutation*.

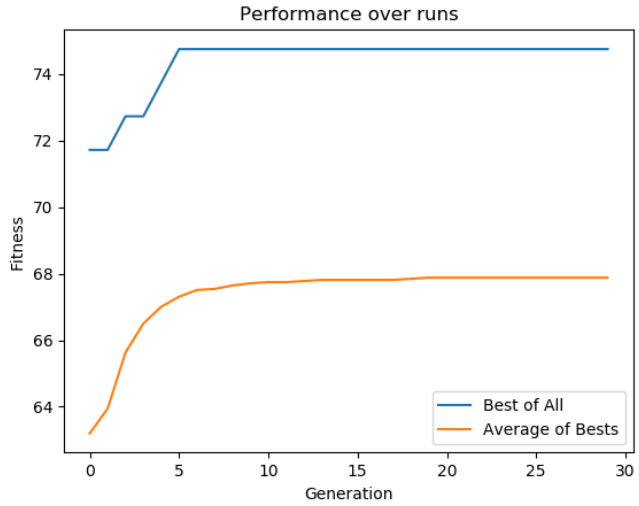


Figura 13: Configuração com Two-Point Crossover e *Scramble Mutation*.

Pela observação dos vários gráficos evolutivos da população, confirma-se a influência dos diferentes operadores de variação, contudo, não se verifica uma diferença significativa entre as diversas combinações, dado que em todas estas foi possível obter um indivíduo com um *fitness* acima de 70%.

Para além do mais, observa-se uma rápida convergência inicial de cada uma das combinações, o que contraria o resultado esperado para os elevados valores atribuídos para os operadores de variação, principalmente no caso da mutação.

Relativamente à média dos melhores indivíduos, é também observado um padrão semelhante, o que indica que a pressão evolutiva tende para uma mesma região do espaço de pesquisa, ou seja, ao longo de um elevado número de

gerações, a população final tende a ser constituída por indivíduos semelhantes (baixa variabilidade).

Por último, é possível concluir graficamente que a combinação de *Two Point Crossover* e *Scramble Mutation* apresenta o melhor resultado.

3.1 Análise Estatística

Com o intuito de retirar conclusões sobre a distribuição e a qualidade de cada uma das soluções, é importante a aplicação da análise estatística. Desta forma, como método inicial da avaliação da qualidade das soluções, recorreu-se à estatística descritiva.

Min	70.70
Max	74.74
Mean	73,09
Median	73.73
Var	2.41
Std	1.55

Tabela 3: Descrição estatística dos dados.

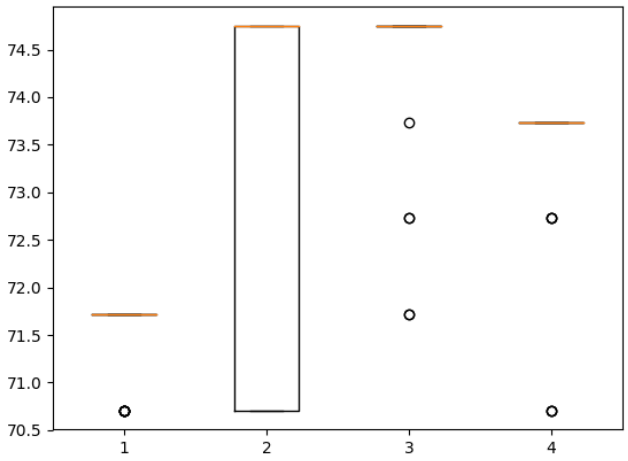


Figura 14: Boxplot para cada uma das combinações de Crossover e Mutation.

1 - Uniform Crossover + Float Custom Mutation 2 - Uniform Crossover + Scramble Mutation 3 - Two-Points Crossover + Scramble Mutation 4 - Two-Points Crossover + Float Custom Mutation

De forma a avaliar a relação das distribuições das soluções, tornou-se necessário a confirmação prévia da normalidade das mesmas. Para este fim, utilizou-se os testes de *Kolgomorov-Smirnov* e *Shapiro-Wilk*, cujo resultados se encontram na tabela 4.

	Kolgomorov-Smirnov		**Shapiro-Wilk**	
	Statistic	P-Value	Statistic	P-Value
Configuração 1	0.765	0.00	0.526	1.01e-8
Configuração 2	0.567	1.25e-9	0.678	7.67e-7
Configuração 3	0.822	0.00	0.467	2.42e-9
Configuração 4	0.792	0.00	0.478	3.19e-9

Tabela 4: Resultados dos testes estatísticos de normalidade

Dado que os *p-values* de cada um dos testes, para cada uma das combinações, se encontra abaixo de 0.05, a hipótese nula pode ser rejeitada e, como tal, as distribuições não seguem uma distribuição normal. Deste modo, a relação entre as distribuições das combinações pode ser avaliada pelo teste de *Friedman's ANOVA*.

Pela aplicação do teste de *Friedman's ANOVA* obteve-se um *p-value* de 4.52E-13, permitindo concluir que as distribuições das configurações são estatísticamente diferentes.

Perante a conclusão anterior, tornou-se ainda importante avaliar a relação entre as configurações que apresentavam as médias mais elevadas no *boxplot*. Para tal, foi utilizado o teste de *Wilcoxon,par wise*, configuração 2 e 3.

Pela aplicação do teste de *Wilcoxon* obteve-se um *p-value* de 0.0013, permitindo concluir que as distribuições são diferentes e, portanto, a configuração 3 foi a escolhida como sendo a melhor, por ter média mais elevada.

4 Conclusão

Os métodos evolutivos traduzem-se em mecanismos inteligentes para obter combinações de parâmetros que sejam mais favoráveis do que aqueles gerados de forma aleatória. Este facto é comprovado quando se compara o histograma (3000 indivíduos gerados aleatoriamente em que o valor a de *fitness* ronda os 50%, nunca excedendo os 65%) com os resultados finais da aplicação do algoritmo evolutivo (*fitness* acima de 70%).

Os operadores de variação e seleção são os meios que constituem toda a parte evolutiva destes métodos. Para tal, foram comparados diferentes métodos, em que se concluiu que a combinação de *Two Point Crossover* e *Scramble Mutation* apresentou o melhor resultado, visto que imprimiu mais variabilidade.

Torna-se ainda importante referir que, para todos os casos, as populações convergiram rapidamente, o que pode ser explicado pela presença de um elevado número de indivíduos diferentes com *fitness* igual. Desde modo, será interessante numa nova análise, aplicar diferentes funções de *fitness* e evoluir mais do que um parâmetro da rede neuronal.

5 Referências

- Saurabh Karsoliya, Student, Department of Computer Science & Engineering, Maulana Azad National Institute of Technology, Bhopal, India. Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture, International Journal of Engineering Trends and Technology- Volume3Issue6- 2012
- Material da cadeira de Computação Neural e Sistemas Difusos, Professor António Dourado, 2017-2018
- Material da cadeira de Computação Evolucionária, Professor Ernesto Costa 2017-2018
- M. Askali, A. Azouaoui, S. Nouh and M. Belkasmi, "On the Computing of the Minimum Distance of Linear Block Codes by Heuristic Methods,"International Journal of Communications, Network and System Sciences, Vol. 5 No. 11, 2012, pp. 774-784. doi: 10.4236/ijcns.2012.511081.