

Análise de redes neurais artificiais

Autor

André Furlan - ensismoebius@gmail.com

O projeto

Este projeto está licenciado sob a licença **GPL versão 3** e está armazenado em um repositório compartilhado no endereço:

<https://github.com/ensismoebius/computacaoInspiradaPelaNatureza>

O documento

Este documento está licenciado sob a licença Creative Commons **Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)**.

Para mais informações acesse: <https://creativecommons.org/licenses/by-sa/4.0/>

A estrutura do projeto

Para melhor acompanhamento deste trabalho é importante entender a estrutura de arquivos do projeto em questão:

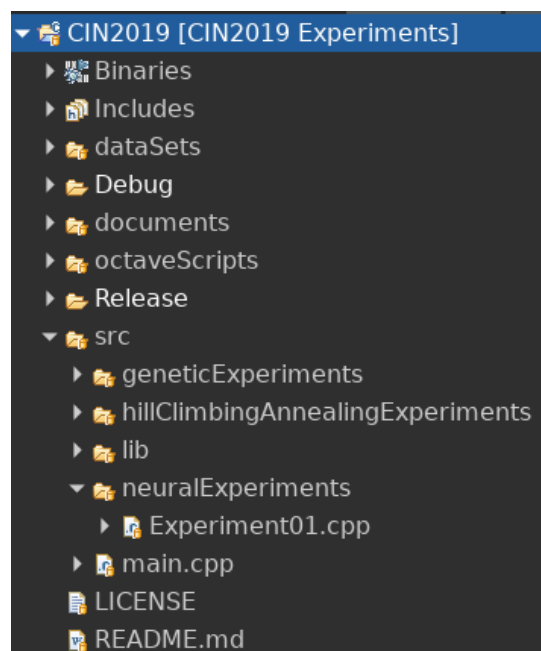


Figura 1: Estrutura do projeto

O **diretório principal** do projeto chama-se “src” é nele que todos os códigos residem.

Para o conteúdo deste documento o diretório de interesse é o “**neuralExperiments**” é nele que serão executados os teste nas redes neurais expostas.

No terceiro diretório estão todas as bibliotecas derivadas do desenvolvimento deste trabalho cada uma

delas procura agrupar e organizar as lógicas e técnicas empregadas de forma que as mesmas possam ser usadas em qualquer outro sistema, ou seja, procurou-se diminuir ao máximo o acoplamento de código. Por fim perto do final da figura se percebe um arquivo chamado "main.cpp" dentro do qual estão as chamadas para todos os experimentos contidos dentro do projeto.

Tecnologia necessárias

- Para que se possa reproduzir os experimentos aqui descritos antes de tudo é necessário ter instalado em seu computador os programas:
- Compilador C++
- GNU/Octave e seu respectivo pacote de estatísticas
- Gnuplot
- Recomenda-se também o computador com sistema operacional GNU/Linux não sendo garantida a execução dos programas em outro sistema operacional embora o mesmo seja totalmente possível.

Função de ativação

Apesar da função de ativação linear cumprir bem o seu papel na maioria dos casos, neste caso escolheu-se usar a função de ativação sigmóide pois apenas assim seria possível plotar os gráficos dos erros quadráticos.

A função sigmóide também permite o uso da rede neural para uma quantidade maior de problemas.

A rede neural

A rede neural em questão foi confeccionada segundo o paradigma de orientação a objetos, ou seja, as bibliotecas confeccionadas foram separadas em duas classes:

- **NeuralNetwork** encontrada no arquivo **NeuralNetwork.cpp**
- **Neuron** encontrada no arquivo **Neuron.cpp**

Assim com todas as outras bibliotecas encontradas neste projeto as classes supracitadas também foram desenvolvidas com a portabilidade em mente, ou seja, evitando-se acoplamento de código.

Essas classes permitem esse crie uma rede neural com qualquer tamanho de entrada (**input layer size**), qualquer tamanho de saída (**output layer size**), com uma quantidade qualquer de camadas escondidas (**inner layers**), que neste trabalho chamaremos de camadas escondidas.

Os experimentos

Os experimentos foram realizados modificando-se alguns parâmetros da rede neural como:

Iterações máximas, quantidade de camadas internas, quantidade de neurônio por camada interna, bias. Os parâmetros tamanho da camada de entrada e tamanho da camada de saída não foram modificados devido ao requerimento dos próprios problemas.

Os experimentos de 1 a 4 manipulam a base de dados das flores, os experimentos de 4 a 8 manipulam a base de dados de vinhos.

O número máximo de iterações varia de acordo com os parâmetros, pois, dependendo dos valores dos parâmetros a rede converge mais devagar ou mais rápido, a quantidade máxima de iterações é dada de forma que não haja overfitting.

Além disso, os gráficos apresentados indicam os valores das somas dos erros das várias iterações.

Experimento 1

Dado	Valor
Bias	1
Taxa de aprendizado	0,1
Máximo de iterações	1300
Quantidade de neurônios por camada interna	0
Quantidade de camadas internas	0

Tabela 1: Experimento 1 - Parâmetros do experimento

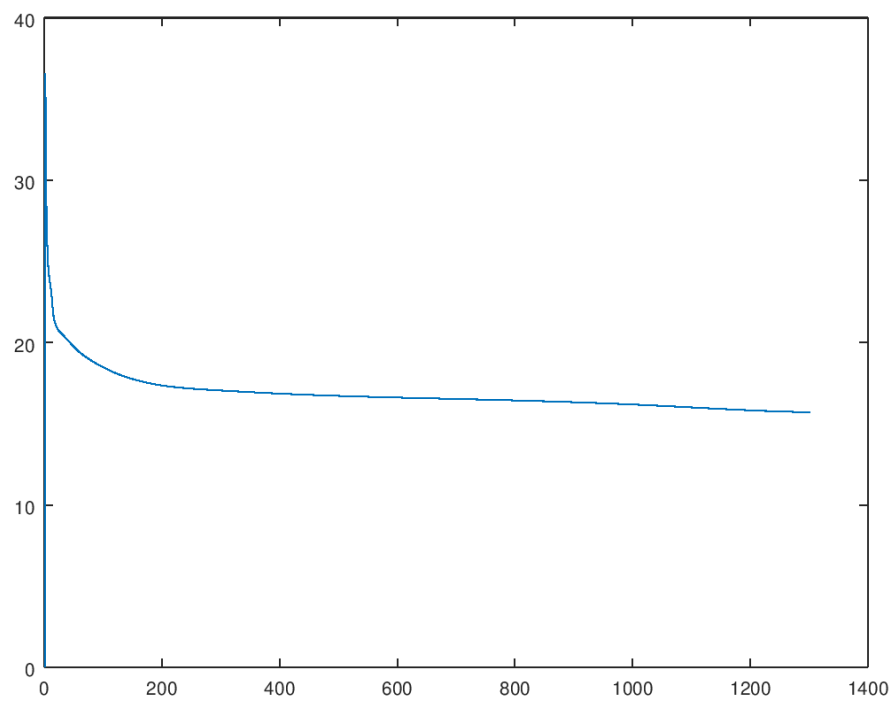


Figura 2: Experimento 1 - Evolução da somatória do erros

		Prediction		
Actual		Iris-setosa	Iris-versicolor	Iris-virginica
	Iris-setosa	8	0	0
	Iris-versicolor	0	8	0
	Iris-virginica	0	0	8

Tabela 2: Experimento 1 - Tabela de confusão para validação

	Prediction			
Actual		Iris-setosa	Iris-versicolor	Iris-virginica
	Iris-setosa	7	0	0
	Iris-versicolor	0	7	0
	Iris-virginica	0	0	7

Tabela 3: Experimento 1 - Tabela de confusão para teste

Apesar da rede neural implementada suportar várias camadas escondidas sua implementação ainda precisa de ajustes, pois, como será possível constatar nos próximos experimentos a adição dessas camadas escondidas faz com que a rede se comporte de forma ineficiente e muitas vezes até mesmo aleatória.

Dada a evolução do somatório dos erros no gráfico acima e as respectivas tabelas de confusão para validação e teste concluímos que esta rede neural cumpriu o seu papel em classificar corretamente todas as flores contidas no banco de dados proposto.

Mesmo com uma pequena taxa de aprendizado percebe-se que o erro da rede diminui rapidamente (aproximadamente após a 200ª iteração), sendo que, após isso a correção dos erros se tornou mais suave.

Experimento 2

Dado	Valor
Bias	0.1
Taxa de aprendizado	0,1
Máximo de iterações	2690
Quantidade de neurônios por camada interna	7
Quantidade de camadas internas	2

Tabela 4: Experimento 2 - Parâmetros do experimento

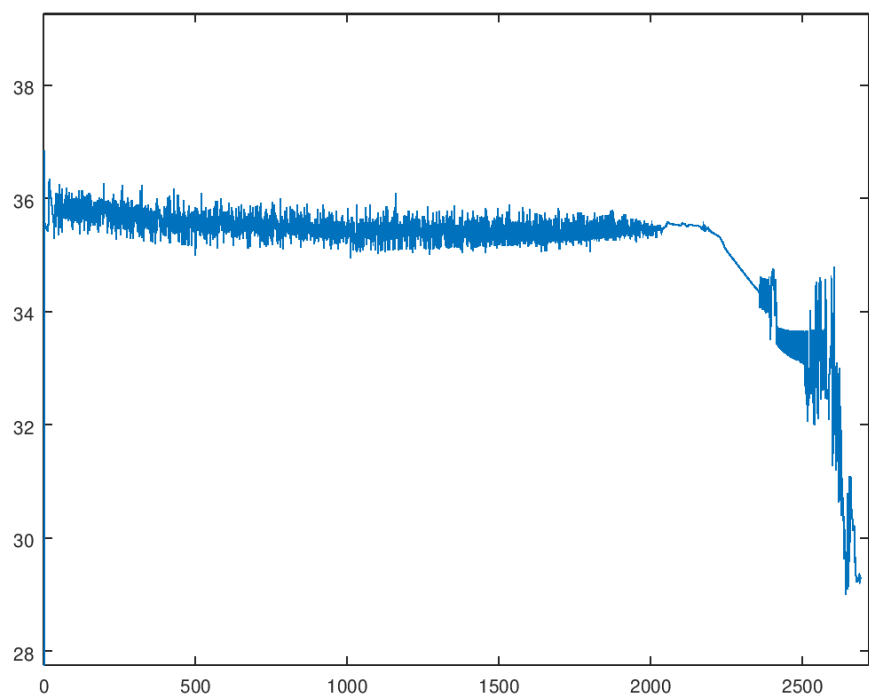


Figura 3: Experimento 2 - Evolução da somatória do erros

		Prediction		
Actual		Iris-setosa	Iris-versicolor	Iris-virginica
	Iris-setosa	6	0	2
	Iris-versicolor	0	0	5
	Iris-virginica	0	0	8

Tabela 5: Experimento 2 - Tabela de confusão para validação

	Prediction			
Actual		Iris-setosa	Iris-versicolor	Iris-virginica
	Iris-setosa	4	0	3
	Iris-versicolor	0	0	7
	Iris-virginica	0	0	7

Tabela 6: Experimento 2 - Tabela de confusão para teste

Como citado anteriormente percebe-se que a adição de camadas escondidas nesta implementação de redes neurais na verdade prejudica o desempenho da mesma. Perceba que mesmo com um número maior de iterações o desempenho da rede na classificação dos dados se torna muito pobre.

É possível perceber isso também no gráfico de evolução dos erros: O gráfico que anteriormente era suave se torna um gráfico com muitos vales e picos e a diminuição dos erros não é mais gradual.

É possível perceber também que uma diminuição significativa nos valores dos erros, ao contrário do experimento anterior, se dá muito tardiamente: Aproximadamente na iteração de número 2000.

Experimento 3

Dado	Valor
Bias	0.1
Taxa de aprendizado	0,1
Máximo de iterações	2400
Quantidade de neurônios por camada interna	4
Quantidade de camadas internas	2

Tabela 7: Experimento 3 - Parâmetros do experimento

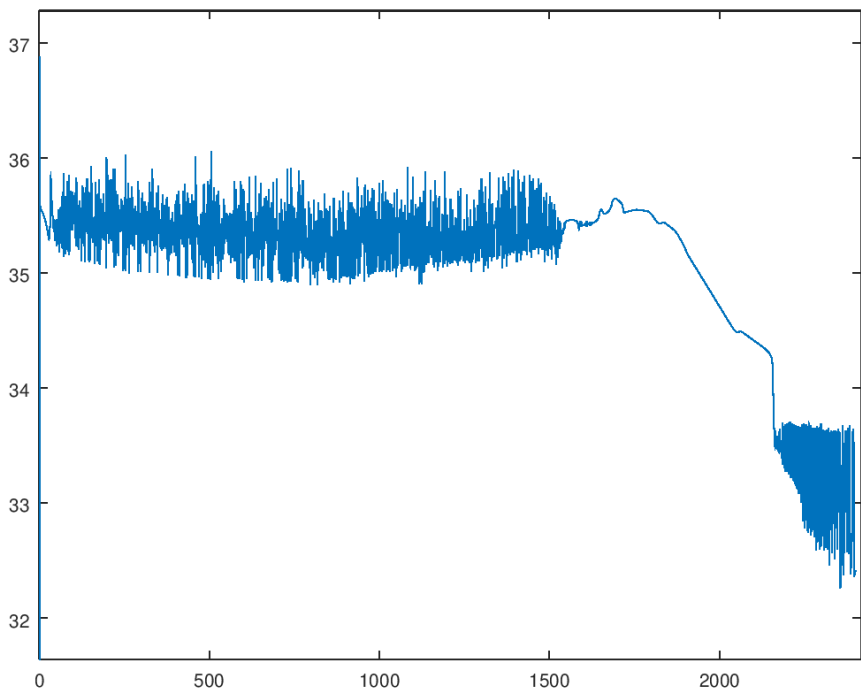


Figura 4: Experimento 3 - Evolução da somatória do erros

		Prediction		
Actual		Iris-setosa	Iris-versicolor	Iris-virginica
	Iris-setosa	0	0	8
	Iris-versicolor	0	0	8
	Iris-virginica	0	0	8

Tabela 8: Experimento 3 - Tabela de confusão para validação

	Prediction			
Actual		Iris-setosa	Iris-versicolor	Iris-virginica
	Iris-setosa	0	0	7
	Iris-versicolor	0	0	7
	Iris-virginica	0	0	7

Tabela 9: Experimento 3 - Tabela de confusão para teste

Este experimento tem basicamente o mesmo comportamento do experimento anterior, com detalhe importante: Diminuiu-se a quantidade de neurônios na camada escondida.

Ao fazer isso se esperava diminuir um pouco o ruído na evolução dos erros da rede, o que de certa forma foi atingido, no entanto apesar dos erros diminuírem de forma um pouco mais suave o desempenho da rede foi ainda pior do que na situação anterior.

Experimento 4

Dado	Valor
Bias	0.1
Taxa de aprendizado	0,1
Máximo de iterações	2690
Quantidade de neurônios por camada interna	16
Quantidade de camadas internas	2

Tabela 10: Experimento 4 - Parâmetros do experimento

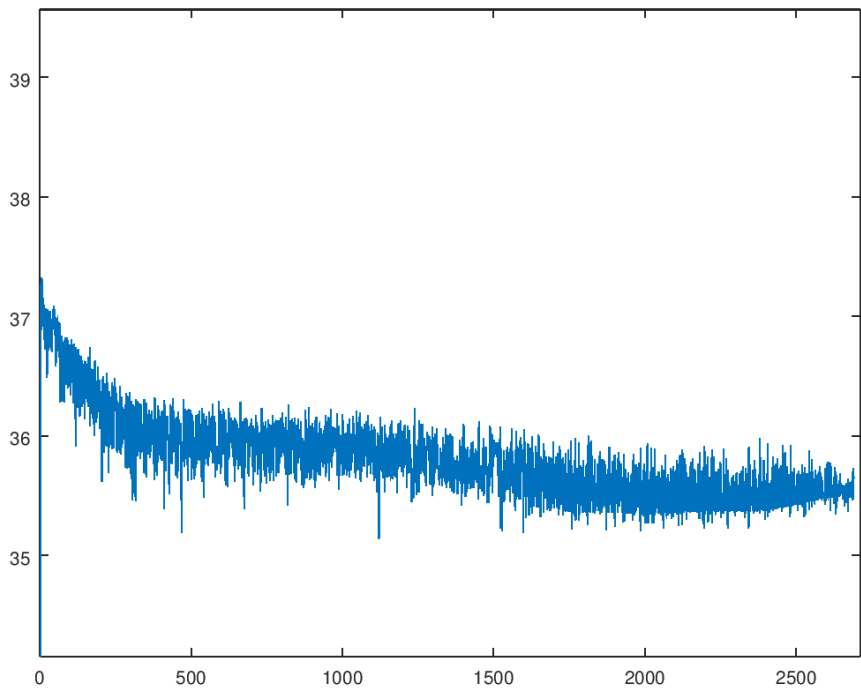


Figura 5: Experimento 4 - Evolução da somatória do erros

	Prediction			
Actual		Iris-setosa	Iris-versicolor	Iris-virginica
	Iris-setosa	0	0	8
	Iris-versicolor	0	0	8
	Iris-virginica	0	0	8

Tabela 11: Experimento 4 - Tabela de confusão para validação

	Prediction			
Actual		Iris-setosa	Iris-versicolor	Iris-virginica
	Iris-setosa	0	0	7
	Iris-versicolor	0	0	7
	Iris-virginica	0	0	7

Tabela 12: Experimento 4 - Tabela de confusão para teste

No experimento anterior concluímos que diminuir a quantidade de neurônios na camada escondida resulta num pior desempenho da rede para classificação das flores, portanto, nesse experimento o número de neurônios em cada camada escondida foi aumentado para 16, no entanto, como já foi comentado no primeiro experimento isso não levou ao melhor desempenho da rede, fato esse contra-intuitivo.

É possível notar também que a somatória do erros não evoluiu muito em cada iteração fazendo com que a rede não consiga classificar corretamente a maioria dos dados.

Experimento 5

Dado	Valor
Bias	1
Taxa de aprendizado	0,1
Máximo de iterações	1000
Quantidade de neurônios por camada interna	13
Quantidade de camadas internas	1

Tabela 13: Experimento 5 - Parâmetros do experimento

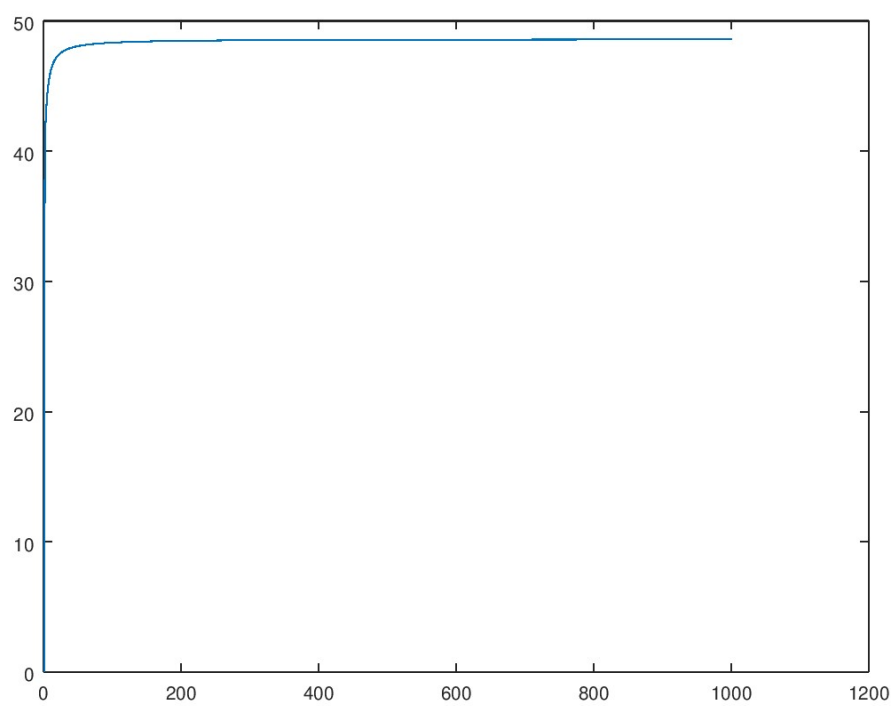


Figura 6: Experimento 5 - Evolução da somatória do erros

	Prediction			
		Classe 1	Classe 2	Classe 3
	Actual	0	9	0
		0	11	0
		0	8	0

Tabela 14: Experimento 5 - Tabela de confusão para validação

	Prediction			
Actual		Classe 1	Classe 2	Classe 3
	Classe 1	0	9	0
	Classe 2	0	11	0
	Classe 3	0	7	0

Tabela 15: Experimento 5 - Tabela de confusão para teste

A partir deste experimento será usada à base de dados de vinhos, considerando os experimentos anteriores nos quais percebemos a tendência de um mal ajuste da rede quando se tem várias camadas, havia que se tentar uma última opção: Diminuir a quantidade de camadas escondidas de 2 para 1, e assim foi feito.

Surpreendentemente a rede se comportou de forma ainda mais estranha, como se pode ver no gráfico os erros acumulados aumentaram exponencialmente fazendo com que o algoritmo divergisse e, como se pode ver na tabela de confusão, tivesse um desempenho muito pobre na classificação dos elementos em questão.

Experimento 6

Dado	Valor
Bias	1
Taxa de aprendizado	0,1
Máximo de iterações	10000
Quantidade de neurônios por camada interna	0
Quantidade de camadas internas	0

Tabela 16: Experimento 6 - Parâmetros do experimento

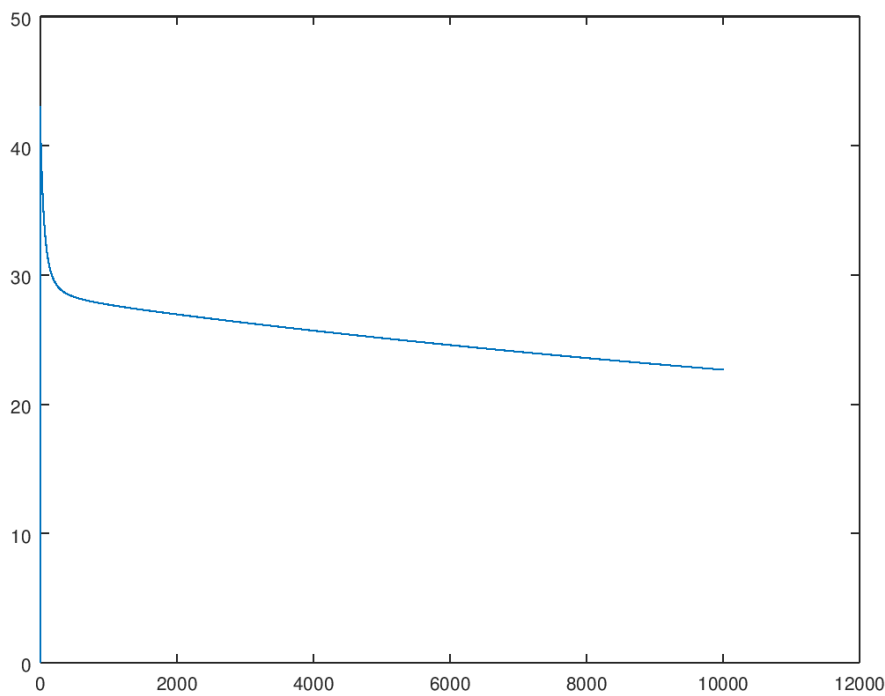


Figura 7: Experimento 6 - Evolução da somatória do erros

	Prediction			
		Classe 1	Classe 2	Classe 3
	Actual	8	1	0
		0	11	0
		0	1	3

Tabela 17: Experimento 6 - Tabela de confusão para validação

	Prediction			
Actual		Classe 1	Classe 2	Classe 3
	Classe 1	9	0	0
	Classe 2	0	11	0
	Classe 3	0	1	6

Tabela 18: Experimento 6 - Tabela de confusão para teste

A partir deste experimento não mais foram usadas redes com camadas escondidas, certamente o problema está **nesta** implementação de tais redes e não num problema intrínseco das mesmas. A partir de agora serão usadas apenas redes de duas camadas: Uma para entrada e outra para saída com seus respectivos pesos intermediários.

Nesta configuração da rede nota-se a evolução esperada: Os erros diminuem rapidamente até aproximadamente a vigésima iteração para então irem diminuindo gradualmente. Pelas tabelas de confusão para validação e testes percebemos que o desempenho dessa rede é bem razoável apresentando muitos poucos erros de classificação.

Experimento 7

Dado	Valor
Bias	1
Taxa de aprendizado	0,9
Máximo de iterações	1000000
Quantidade de neurônios por camada interna	0
Quantidade de camadas internas	0

Tabela 19: Experimento 7 - Parâmetros do experimento

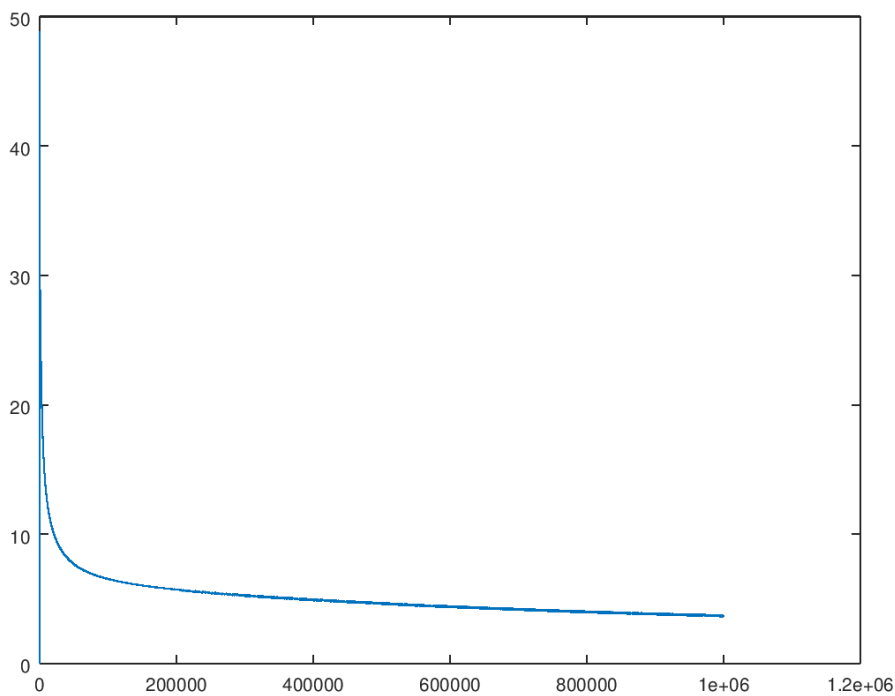


Figura 8: Experimento 7 - Evolução da somatória do erros

	Prediction			
		Classe 1	Classe 2	Classe 3
	Classe 1	9	0	0
	Classe 2	0	11	0
	Classe 3	0	0	7

Tabela 20: Experimento 7 - Tabela de confusão para validação

	Prediction			
Actual		Classe 1	Classe 2	Classe 3
	Classe 1	9	0	0
	Classe 2	0	11	0
	Classe 3	0	1	6

Tabela 21: Experimento 7 - Tabela de confusão para teste

Considerando o experimento anterior, cuja taxa de sucesso foi bem alta, intuitivamente formulou-se uma hipótese: Uma taxa de aprendizado maior e uma quantidade maior de iterações levaria a uma rede cuja classificação seria mais exata.

Após a execução do treinamento da rede que, diga-se de passagem, levou um longo tempo, obteve-se uma rede que cometeu apenas um erro de classificação, confirmando a hipótese anteriormente formulada.

Experimento 8

Dado	Valor
Bias	1
Taxa de aprendizado	0,9
Máximo de iterações	100000
Quantidade de neurônios por camada interna	0
Quantidade de camadas internas	0

Tabela 22: Experimento 8 - Parâmetros do experimento

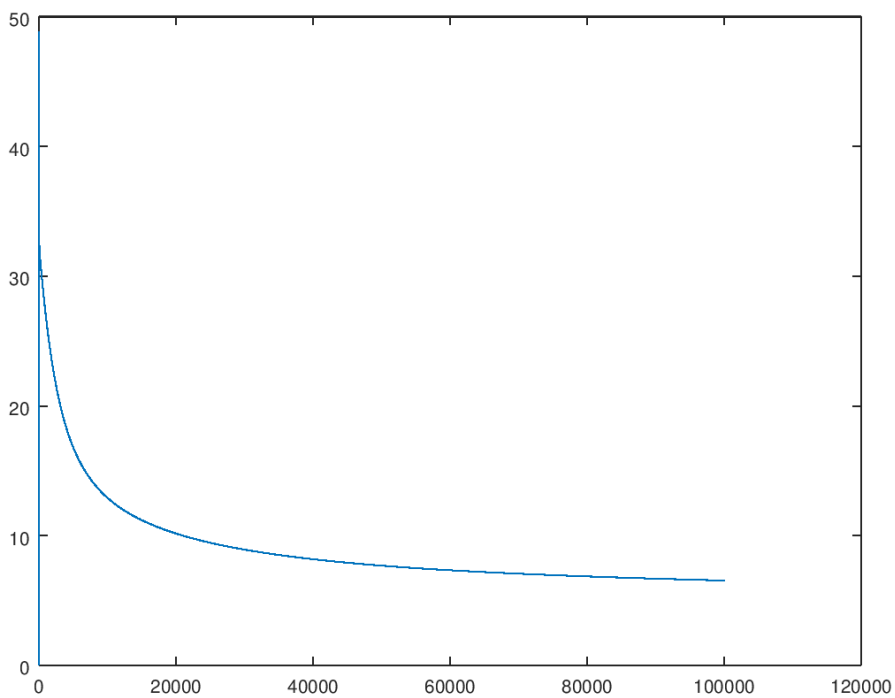


Figura 9: Experimento 8 - Evolução da somatória do erros

	Prediction			
		Classe 1	Classe 2	Classe 3
	Actual	8	1	0
		0	11	0
		0	0	7

Tabela 23: Experimento 8 - Tabela de confusão para validação

	Prediction			
		Classe 1	Classe 2	Classe 3
Actual	Classe 1	9	0	0
	Classe 2	0	11	0
	Classe 3	0	1	6

Tabela 24: Experimento 8 - Tabela de confusão para teste

Dado que a quantidade de iterações do experimento 7 era um tanto quanto extremada, neste experimento procurou-se encontrar um número de menor iterações que resultasse numa rede igualmente eficiente. Perceba que a quantidade de iterações agora é 10 vezes menor do que no experimento anterior, sendo que, os resultados obtidos tem a mesma exatidão.

Referências

Haykin, Simon. **Redes Neurais 2ª edição**: Porto Alegre, Bookman, 2007.

Implementing an Artificial Neural Network in Pure Java (No external dependencies). Disponível em: <<https://medium.com/coinmonks/implementing-an-artificial-neural-network-in-pure-java-no-external-dependencies-975749a38114>>. Acesso em: 14 de Maio de 2019.

Machine Learning Basics and Perceptron Learning Algorithm. Disponível em: <<https://www.codeproject.com/Articles/1211753/WebControls/>>. Acesso em: 14 de Maio de 2019.

Perceptron. Disponível em: <<https://rosettacode.org/wiki/Perceptron#Java>>. Acesso em: 14 de Maio de 2019.

Session 4 - Neural Networks - Intelligence and Learning. Disponível em: <<https://www.youtube.com/watch?v=XJ7HLz9VYz0&list=PLRqwX-V7Uu6Y7MdSCalfsxc561QI0U0Tb>>. Acesso em: 21 de Maio de 2019.

The Nature of Code - Chapter 10. Neural Networks. Disponível em: <<https://natureofcode.com/book/chapter-10-neural-networks/>>. Acesso em: 20 de Maio de 2019.

Vídeo 17 - Formulação da Camada de Saída da Multilayer Perceptron. Disponível em: <<https://www.youtube.com/watch?v=cQ3x15UJsHA>>. Acesso em: 22 de Maio de 2019.

Vídeo 20 - Implementação da Multilayer Perceptron 1/2. Disponível em: <<https://www.youtube.com/watch?v=FSvD2HT0Zfg>>. Acesso em: 8 de Maio de 2019.

Z-Score: Definition, Formula and Calculation. Disponível em: <<https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/z-score/>>. Acesso em: 10 de Maio de 2019.