Um Modelo de Redes Neurais Artificiais tipo Perceptron para Predição de Placares de Jogos de Futebol no Campeonato Brasileiro

Daniel Andrade[1], **Wanderson Silva**[1]

[1]Universidade Estadual de Feira de Santana (UEFS)
Feira de Santana – BA – Brasil

{dca650,bsilva.wanderson}@gmail.com

1. Introdução

Com o crescimento do mercado de apostas esportivas, muitas pessoas buscam métodos para prever o resultado dos jogos. Os sites de apostas, por outro lado, usam métodos para dar pesos aos prêmios das apostas de acordo com a previsibilidade do resultado. Visto este cenário, o presente relatório visa detalhar o projeto e a implementação de uma Rede Neural Artificial (RNA) tipo Perceptron para predição de placares de jogos de futebol do Campeonato Brasileiro.

2. Metodologia

Para implementação da RNA foi utilizado o Keras[Chollet 2015], uma biblioteca de alto-nível para implementação de redes neurais. Escrita em Python, o Keras roda como uma espécie de *front-end* para as bibliotecas TensorFlow ou Theano.

Base de dados: A base de dados utilizada para treinamento, validação e testes foi criada utilizando um crawler implementado para coletar dados do site [Globo.com 2018]. Um desafio encontrado durante a coletas destes dados foi a necessidade de realizar eventos de clique para mudar a tabela e obter dados das rodadas seguintes. Para solucionar este problema, utilizou-se framework Selenium [Salunke 2014], que funciona como uma espécie de *bot*, simulando as ações necessárias dentro do navegador.

Os dados coletados abrangem partidas do campeonato de 2003 até o campeonato 2017, onde para cada partida o crawler capta o nome dos times, placar, escalação, nome do árbitro, estádio, data e o histórico de confronto entre as duas equipes. O sistema gerenciador de banco de dados escolhido foi o MongoDB[Chodorow and Dirolf 2010]. Vale adiantar que nem todos os dados estão sendo levados em consideração para a RNA.

Arquitetura da rede: A RNA implementada é do tipo Perceptron, com 13 neurônios na camada de entrada e 2 neurônios na camada de saída. As tabelas 1 e 2 mostram respectivamente as entradas e saídas do modelo implementado:

Tabela 1. Entradas da rede

	Entradas
1	Gols no histórico de confronto direto feitos pelo time A
2	Gols no histórico de confronto direto feitos pelo time B
3	Vitorias do time A no histórico de confrontos
4	Vitorias do time B no histórico de confrontos
5	Quantidade de empates no histórico de confrontos
6	Saldo de gols na partida anterior (K) do time A
7	Saldo de gols na partida $K-1$ do time A
8	Saldo de gols na partida $K-2$ do time A
9	Saldo de gols na partida $K-3$ do time A
10	Saldo de gols na partida anterior(K) do time B
11	Saldo de gols na partida $K-1$ do time B
12	Saldo de gols na partida $K-2$ do time B
13	Saldo de gols na partida $K-3$ do time B

Tabela 2. Saídas da rede

	Saídas
1	Predição de gols feitos pelo time A
2	Predição de gols feitos pelo time B

Como não existe um método encontrar a quantidade ideal de neurônios na camada intermediária, inicialmente optou-se por utilizar o valor médio entre a quantidade de neurônios na camada de entrada e na camada de saída, podendo, a partir desde ponto, escolher outros valores para quantidade de neurônios na camada intermediária e optar pela configuração que gerou o melhor resultado, visto que, na camada intermediária, o aumento da quantidade de neurônios faz rede ficar mais sensível às variações de dados na camada de entrada, já a redução da quantidade de neurônios na camada intermediária deixa a rede mais rígida em relação às mudanças de dados na camada de entrada. É válido ressaltar que todos os atributos foram individualmente normalizados para possuírem média zero e variação unitária.

Funções de ativação: Para facilitar o projeto da rede, o Keras já possui implementações de algumas funções de ativação. Durante o desenvolvimento da rede, foram escolhidas algumas funções com o objetivo de encontrar a combinação que gera o melhor resultado. As funções de ativação escolhidas para os testes foram: Sigmoid, Linear, Unidade Linear Retificada (ReLu) e Tangente Hiperbólica (TanH). Dado que os placares não são limitados, optou-se por utilizar uma função de ativação ilimitada em ambos os lados na última camada, a função Linear. Para a camada intermediária, dado que o desempenho de um time pode influenciar no número de gols do outro time, optou-se por utilizar uma função de ativação com um intervalo de -1 a 1, ou seja, TanH.

Algoritmos de treinamento e métricas de avaliação: Assim como as funções de ativação, o Keras também já possui implementações de algoritmos de treino e métricas de avaliação dos resultados. Como não há uma forma de afirmar com exatidão qual o algoritmo irá apresentar o melhor resultado, é necessário avaliar alguns algoritmos que se adaptam ao problema para definir qual será utilizado. Dentre os algoritmos de treina-

mento disponíveis, foram escolhidos os seguintes otimizadores de gradiente descendente: Adaptive Moment Estimation(ADAM), RSMprop, Stochastic Gradient Descent (SGD).

Para avaliação dos resultados do treinamento e do teste, utilizou-se a Acurácia Binária para obter a porcentagem de predições completamente corretas e o Erro Absoluto Médio para obter a diferença entre o resultado encontrado e o resultado esperado.

3. Resultados e Discussões

Para efetuar a avaliação do impacto dos parâmetros na rede neural, a configuração da Tabela 3 foi adotada como configuração padrão. Dessa forma, quando um dos parâmetros está sendo estudado (e.g. algoritmo de inicialização de pesos), os outros são mantidos constantes. A proporção de dados para treino e teste foi determinada como 80% e 20%. Outras proporções foram avaliadas porém, por não afetarem substancialmente os resultados, foram omitidas deste relatório.

Inicialização	Neurônios na camada escondida	Algoritmo de Treino
Glorot Uniform	8	ADAM

Tabela 3. Configuração padrão da rede

3.1. Procedimentos sem validação

Variando inicialização de pesos: Dentre as os algoritmos disponibilizados pela ferramenta utilizada, foram utilizados o *Glorot Uniform* (valores gaussianos com média zero), *Ones* (todos os pesos iguais a 1), *Zeros* (todos os pesos iguais a 0) *Random Normal* (valores aleatórios da distribuição normal). As Figuras 1 e 2 mostram os histórico da acurácia binária e do erro absoluto médio, respectivamente, para cada época.

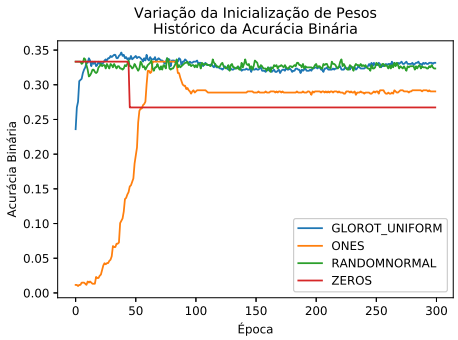

Figura 1. Acurácia binária por época na variação do algoritmo da inicialização de pesos

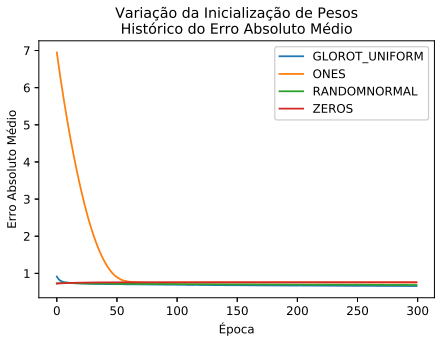

Figura 2. Erro absoluto médio por época na variação do algoritmo da inicialização de pesos

Através da análise das Figuras 1 e 2, é possível perceber que os pesos finais estão próximos a zero. Isso é evidenciado pela crescente curva do algoritmo *Ones* e pela relativa estabilidade dos algoritmos que são iniciados exatamente em zero ou próximo disso. Também é possível perceber que a inicialização com zeros não é flexível, uma vez que seus resultados pouco variam durante as épocas de treinamento. As inicializações aleatórias possuem desempenhos similares e são as alternativas que apresentam melhor desempenho.

Variando quantidade de neurônios na camada escondida: As Figuras 3 e 4 mostram os históricos da acurácia binária e do erro absoluto médio, respectivamente, para cada época enquanto a quantidade de neurônios na camada escondida é variada.

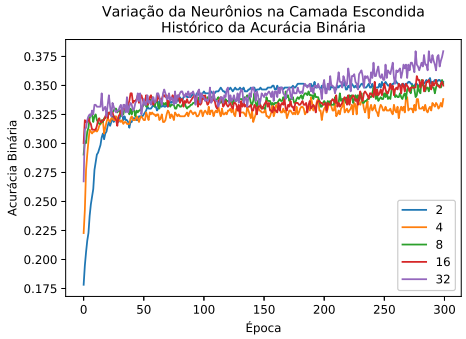

Figura 3. Acurácia binária por época na variação da quantidade de neurônios na camada escondida

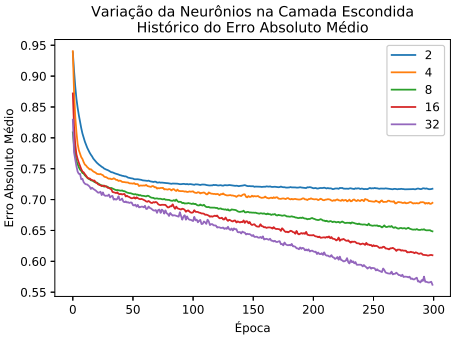

Figura 4. Erro absoluto médio por época na variação da quantidade de neurônios na camada escondida

As medidas durante a variação da quantidade de neurônios na camada escondida apresentem dois comportamentos bem distintos. Enquanto o erro absoluto médio diminui proporcionalmente a tal quantidade, a acurácia apresenta grandes oscilações e não demostra ser proporcional a quantidade de neurônios. Apesar da rede com 32 neurônios intermediários demonstrar maior eficácia final, a rede com apenas 2 é superior a ela em diversas épocas e, ao final, possui desempenho equiparável as redes com 8 e 16 neurônios intermediários.

Variando o algoritmo de treino: As Figuras 11 e 12 mostram os histórico da acurácia binária e do erro absoluto médio, respectivamente, para cada época enquanto o algoritmo de treino é variado.

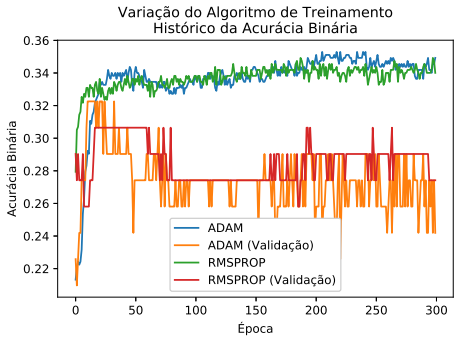

Figura 5. Acurácia binária por época na variação da quantidade do algoritmo de treinamento

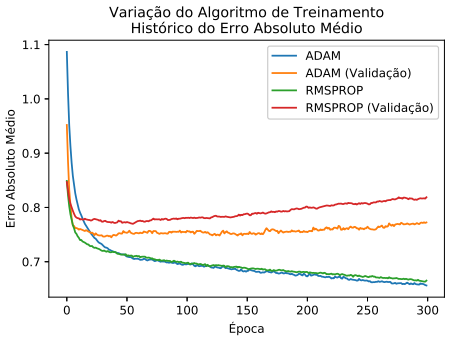

Figura 6. Erro absoluto médio por época na variação da quantidade do algoritmo de treinamento

De forma geral, os algoritmos de treinamento *ADAM* e *RMSPROP*, seguidos do *SGD*, tem melhor performance em relação as duas medidas. Considerando a acurácia binária como o critério mais importante por refletir o placar com mais exatidão, o *ADAM* se mostra superior por convergir mais rapidamente.

3.2. Procedimentos com validação

Para fins de melhor visualização dos resultados, os experimentos de validação foram restritos as duas alternativa com melhor desempenho no estudo sem validação. Em todos os casos, 10% dos dados foi reservado para validação.

Variando inicialização de pesos:

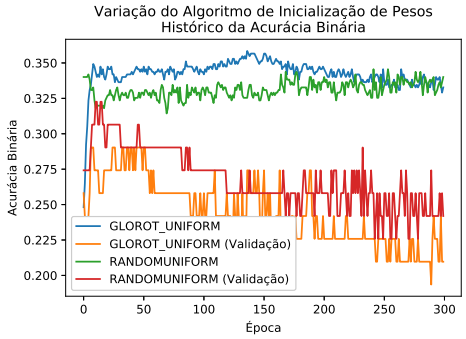

Figura 7. Acurácia binária por época na variação do algoritmo da inicialização de pesos

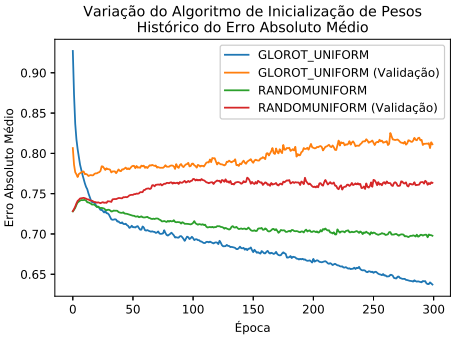

Figura 8. Erro absoluto médio por época na variação do algoritmo da inicialização de pesos

O algoritmo *Glorot Uniform* se mostra mais efetivo tanto no treino como na validação, principalmente quando o treinamento é limitado a 100 épocas.

Variando quantidade de neurônios na camada escondida:

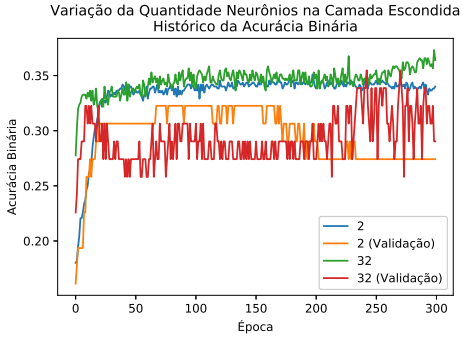

Figura 9. Acurácia binária por época na variação da quantidade de neurônios na camada escondida

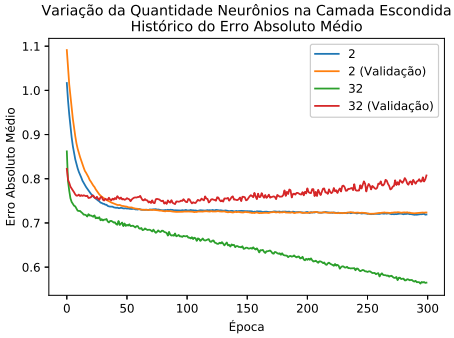

Figura 10. Erro absoluto médio por época na variação da quantidade de neurônios na camada escondida

Analisando em conjunto com os dados de validação, é perceptível que 2 neurônios na camada escondida, na época 100, são suficientes para o caso estudado. Apesar da rede com 32 neurônios intermediários, para a acurácia, apresentar uma boa performance tanto no treino tanto na validação, a curva crescente do erro durante a validação deste caso é um indicador de superadaptação.

Variando algoritmo de treino:

Similar ao caso anterior, verifica-se que o ADAM é o melhor algoritmo no treino e na validação quando parado em torno da época 17.

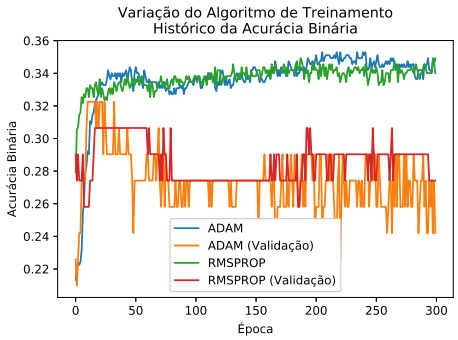

Figura 11. Acurácia binária por época na variação da quantidade do algoritmo de treinamento

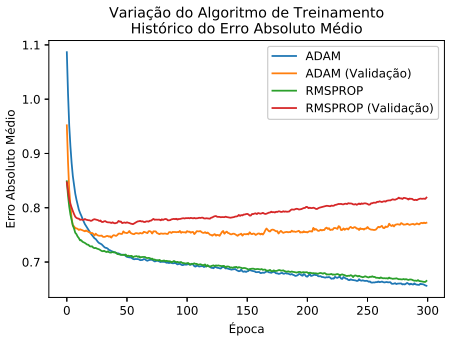

Figura 12. Erro absoluto médio por época na variação da quantidade do algoritmo de treinamento

3.3. Estrutura Final da Rede Neural

Realizando estudos com combinações dos parâmetros anteriores, a configuração final da rede foi definida com 8 neurônios na camada intermediária, ADAM como algoritmo de treinamento e os pesos iniciados pelo algoritmo *Glorot Uniform* e limitado a 20 épocas. Por fim, foram utilizados 260 primeiros jogos do Campeonato de 2017 para efetuar o teste prático da rede. Os resultados estão dispostos na Tabela 4, onde acurácia se refere a acurácia binária e erro se refere ao erro absoluto médio.

Situação	Acurácia (Teste)	Erro (Teste)	Acurácia (Validação)	Erro (Validação)
Teste e Validação	32.17%	73.66%	32.26%	76.06
Teste Prático	27.50%	50.40%	N/A	N/A

Tabela 4. Configuração padrão da rede

4. Conclusão

A RNA construída para predizer os placares de partida de futebol apresentou uma eficácia satisfatória tanto nos testes quanto nas etapas de validação. Apesar de numericamente baixa, as taxas de acurácia e erro obtidas refletem a dificuldade de realizar previsões sobre um evento com inúmeras variáveis, pois até mesmo o clima e a disposição dos jogadores pode afetar o desempenho do time. Diante disso é possível concluir que, dado a quantidade limitada de entradas utilizadas, o projeto e os resultados alcançados pela rede cumprem os requisitos do projeto.

Referências

Chodorow, K. and Dirolf, M. (2010). *MongoDB: The Definitive Guide*. O'Reilly Media, Inc., 1st edition.

Chollet, F. (2015). Keras. `https://github.com/fchollet/keras`.

Globo.com (2018). Futpédia. `http://futpedia.globo.com/`. acessado em 12/05/2018.

Salunke, S. S. (2014). *Selenium Webdriver in Python: Learn with Examples*. CreateSpace Independent Publishing Platform, USA, 1st edition.