Identificação de traços duros usando uma simples rede neural perceptron e recursos de multiprocessos do *Python*

Anderson S. Santos[1], Nathalia Azevedo[2], Pablo Carvalho[2], Gilmário Maciel[1], Rodrigo Portugal[2];

(1) Petrobras-GDGEOF, (2) Fundação Gorceix

Resumo

A identificação de traços duros em dados sísmicos de reflexão é uma tarefa que exige a análise de um grande volume de dados sísmicos por parte do geofísico. Essa tarefa é extremamente onerosa devido a grande quantidade de traços sísmicos envolvidos em um programa de aquisição, que na maioria dos casos envolve milhões de traços. Os métodos clássicos envolvem a aplicação de métodos determinísticos ou probabilísticos que tem por objetivo identificar aqueles traços que possuem amostras zeradas ou propriedades estatísticas que os diferem dos traços sísmicos esperados. Nós apresentamos nesse trabalho um método para identificação de traços zerados usando uma simples rede neural artificial perceptron construída usando a biblioteca Keras. Este método é robusto e se mostrou bastante eficiente na sua aplicação em um grande volume de dados sísmicos reais de importantes bacias sedimentares brasileiras o que demostra que a sua extensão para outros tipos de traços ruidosos pode ser feita de forma bem eficiente e simples. Uma comparação desse método é feita com o simples cálculo da média RMS do traço sísmico otimizado pela biblioteca *multiprocessing* do *Python*.

Introdução

A atividade de exploração de hidrocarbonetos envolve a aquisição, processamento e interpretação de um grande volume de dados geofísicos (Yilmaz & Doherty, 2001). Dentre esses métodos, o método sísmico de reflexão se destaca devido à alta resolução para o imageamento das estruturas geológicas da subsuperfície (Mora & Tarantola, 1990). Durante o processo de aquisição de dados muitos traços sísmicos duros são adquiridos, são traços que não apresentam nenhuma informação de interesse sobre a geologia. Esses traços serão eliminados na etapa do processamento sísmico conhecido como edição (Onajite, 2014).

Devido ao grande volume de dados a busca manual desses traços duros é impraticável, pois nos levantamentos atuais a quantidade de traços sísmicos obtidos é da ordem de grandeza dos milhões. A correta identificação e eliminação desses traços é de extrema importância para as etapas subsequentes do processamento de dados sísmicos (Hu et al., 2018). Muitas pesquisas atuais têm dado foco na aplicação de redes neurais artificiais para solução de problemas geofísicos, tais como filtragem, imageamento e interpretação de dados geofísicos.

Para identificação de traços duros usando redes neurais artificiais temos o trabalho de McCormack (1990), onde o autor usou uma rede neural artificial perceptron para fazer o *pick* de primeira quebra e a edição automática de traços sísmicos duros.Yang et al. (2018) usou uma rede CNN *(Convolutional Neural Network)* em conjunto com a transformada de *Hough* para fazer a edição automática de traços sísmicos, porém tratava a matriz de dados como uma imagem com o objetivo de treinar a rede para reconhecer o evento indesejado.

Nesse trabalho é apresentado uma forma de identificar traços zerados em um grande volume de dados sísmicos reais utilizando uma rede neural artificial *Perceptron* que usa a matriz de dados diretamente com a suas informações de amplitude e fase. Nessa abordagem que apresentamos, uma rede neural artificial é treinada para identificar traços zerados e uma vez que a rede tenha sido treinada e os seus pesos salvos a aplicação desses pesos em um novo conjunto de dados é feita de forma muito rápida e eficiente usando a biblioteca *Keras* (Keras API). Os resultados se mostraram satisfatórios, pois a rede conseguiu identificar traços duros durante a sua aplicação em um volume considerável de dados sísmicos reais de importantes bacias sedimentares brasileiras com uma taxa de acerto de 100%. O método ainda carece de otimização, pois apesar de rodar em um *Cluster* o *tensorfow-gpu* não foi usado e ainda assim os resultados de tempo alcançados pela rede para identificação dos traços zerados foram mais eficientes quando comparados com o simples cálculo RMS(*Root Mean Square*) dos traços sísmicos.

Arquitetura da Rede Perceptron

A rede neural criada para identificação inicialmente de traços duros em dados sísmicos foi bem simples, pois o problema de classificar traços bons e traços zerados é perfeitamente separável garantindo uma boa acurácia da rede neural artificial. A camada de entrada recebe um traço sísmico com **ns** amostras no tempo e a rede neural artificial possui apenas uma camada oculta com 4 neurônios e 2 neurônios na camada de saída (Figura 1).

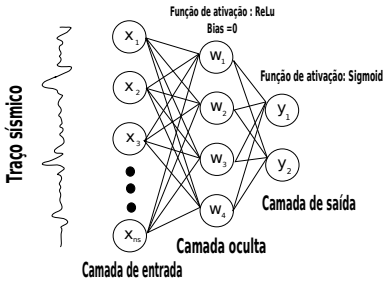

Figura 1 – Arquitetura da rede neural usada para identificação de traços zerados.

Os neurônios da camada oculta possuem função de ativação ReLu *(Rectified Linear Unit)* dada pela equação:

$$f(x) = x^{+} = max(0, x) \quad (1)$$

onde x é a entrada para um neurônio.
Os neurônios da camada de saída possuem como função de ativação a função sigmoide, sendo dada pela equação:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2)$$

onde x é a entrada para os neurônios na última camada.

Todos os vieses foram retirados da etapa de treinamento da rede, pois o problema de identificar traços sísmicos zerados é muito simples e a presença do parâmetro *bias* acabou gerando mais incerteza nos resultados. Um vetor y de dimensões $2 \times nt$, onde nt é quantidade de traços rotulados passados para rede neural artificial. Os traços bons foram rotulados como $[1,0]^T$ e os traços zerados como $[0,1]^T$ (Figura 2).

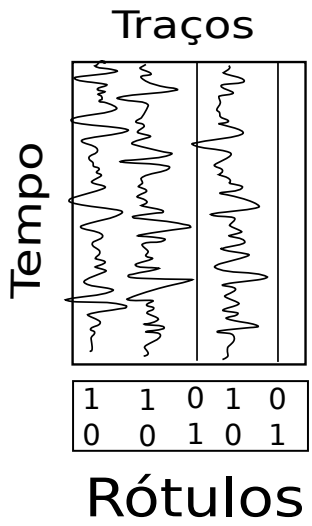

Figura 2 – Rótulo dos traços sísmicos usados no treinamento e validação da rede neural artificial.

Esses traços bons e zerados foram obtidos aleatoriamente de um dado sísmico real usando um método determinístico para verificar se o traço estava zerado e em seguida esses traços foram selecionados aleatoriamente para o treinamento e validação da rede. Após o treinamento e validação, um conjunto de dados não visto pela rede foi entregue para ser feita a predição dos traços bons e zerados no arquivo *segy*. A função de perda, ou função objetivo, foi baseada no erro médio quadrático dada pela equação:

$$EMQ = 1/N \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Onde i é a $i-ésima$ amostra, y é a resposta calculada da rede e $\hat{y}$ é a resposta verdadeira fornecida e N é o número de amostras. O otimizador usado foi Adam *(Adaptative Momentum estimation)*, um método eficiente de otimização estocástica onde somente requer os gradientes de primeira ordem com pouca exigência de memória computacional (Kingma & Ba, 2015).

Construção da Rede Neural Usando o *Keras*

A rede neural artificial pode ser facilmente criada usando a biblioteca keras, que é uma biblioteca de rede neural escrita em *Python* (Keras API). As camadas da rede são criadas de forma muito simples iniciando a classe *Sequential* e definindo os hiperparâmetros da mesma dentro da sintaxe conforme o pedaço do código extraído na Figura 3.

```
    model = Sequential()
    model.add(Dense(NB_CLASSES, input_shape=(nt,), use_bias=False))
    model.add(Dense(N_HIDDEN,activation='relu',use_bias=False, bias_initializer="zeros"))
    model.add(Dense(NB_CLASSES))
    model.add(Activation('sigmoid'))
    model.summary()
    model.compile(loss='mean_squared_error', optimizer=OPTIMIZER, metrics=['accuracy'])
    history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NB_EPOCH,
verbose=VERBOSE,validation_split=VALIDATION_SPLIT)
```

Figura 3 – Fragmento do código da rede neural usando a biblioteca keras para definição das camadas da rede.

Criamos uma função que recebe os hiperparâmetros da rede que estão na tabela 1.

Número de épocas	10
Quantidade de camadas	1
Quantidade de neurônios na camada oculta	4
Função de ativação na camada oculta	ReLu
Quantidade de neurônios na camada de saída	2
Função de ativação da camada de saída	Sigmoid
Tamanho de Batch	600
Função de custo	Erro médio quadrático
Otimizador	Adam
Quantidade de traços para treino	2.927.232
Quantidade de traços para validação	2.927.232
Acurácia	100%

Tabela 1 – Hiperparâmetros da rede e lguns resultados da etapa de treinamento e validação.

Resultados

A rede neural artificial de apenas uma camada com quatro neurônios foi eficiente em identificar traços sísmicos zerados para um grande volume de dados sísmicos 3D conforme notamos nas tabelas 2 e 3. A rede neural, apesar de simples, levou pouquíssimo tempo para predizer traços zerados quando se aumenta o tamanho dos arquivos *segy*. Isso apesar dos processos de otimizações que ainda não foram feitos, tais como a implementação da rede neural em ambiente *cluster* usando o *tensorflow-gpu* e otimização na etapa de aplicação dos pesos salvos da rede. O método que chamamos de determinístico consiste no cálculo da média RMS de cada traço sendo otimizado com a biblioteca de *multiprocessing* do *Python* em um nó com 255 CPUs. Esses resultados são promissores nos encorajando a aumentar complexidade do problema e treinar a rede para identificar outras características que classifiquem o traço sísmico como um traço duro, tais como mono frequências, amplitudes anômalas e etc.

Dado		**Rede Neural**	
Segyfile	**Tamanho**	**Performance**	**Traços Duros**
Arquivo1.sgy	2,2GB	00:00:01	0,06%
Arquivo2.sgy	15GB	00:00:05	2,93%
Arquivo3.sgy	43GB	00:00:13	1,27%

Tabela 2 – Tabela mostrando o desempenho computacional da rede neural artificial usada para identificar traços zerados e a quantidade de traços identificados sendo que a taxa de acerto foi de 100%.

Dado		**Determinísticos**	
Segyfile	**Tamanho**	**Performance**	**Traços Duros**
Arquivo1.sgy	2,2GB	00:00:07	0,06%
Arquivo2.sgy	15GB	00:00:49	2,93%
Arquivo3.sgy	43GB	00:01:42	1,27%

Tabela 3 – Tabela mostrando o desempenho computacional do simples cálculo da RMS do traço sísmico usando a biblioteca de *multiprocessing* no *Python*.

Apesar da identificação de traços sísmicos zerados usando o simples cálculo do RMS (*Root Mean Square*) do traço sísmico otimizado pela biblioteca de *multiprocessing* do *python* ser bastante eficiente rodando em um nó com 255 CPUs, o método que apresentamos para identificação inicial de traços sísmicos zerados foi muito eficiente na identificação desses traços ruidosos, atingindo uma precisão de 100% e foi muito mais rápido do que a abordagem clássica.

Discussões e Conclusões

Foi mostrado como uma simples rede neural artificial pode ser usada no controle de qualidade de dados sísmicos para identificação de traços zerados. Os resultados foram bastantes eficientes devido ao baixo custo computacional associados tanto ao processo de treino e validação da rede como a sua aplicação para um grande volume de dados sísmicos. O tempo para identificação de traços zerados para um grande volume de dados sísmicos também foi bastante satisfatório. Os próximos passos para essa rede neural é a otimização do treinamento e aplicação da rede usando o treinamento da rede para o reconhecimento de outras características que classifiquem os traços como duros que incluem monofrequências, amplitudes anômalas etc, assim como a otimização da rede para rodar em ambiente do *cluster* usando a biblioteca *tensorflow-gpu*.

Agradecimentos

Os autores agradecem a Petrobras e a Fundação Gorceix pela infraestrutura disponibilizada e pela liberação para publicar os resultados iniciais desse trabalho. Os autores também agradecem pelo apoio e suporte de Valério C. L. Dutra, Jéferson S. da Rocha e Gabriel S. Lana da TIC/OI/SSH/SAD Petrobras.

Referências

Hu, Y., Xia, J., Mi, B., Cheng, F. & Chao, 2018. A pitfall of muting and removing bad traces in surface-wave analysis, Journal of Applied Geophysics, vol. 153: 136–142, doi: https://doi.org/10.1016/j.jappgeo.2018.04.013.

Kingma, D. P. & Ba, J., 2015. Adam: A method for stochastic optimization, in: Bengio, Y. & LeCun, Y. (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, URL http://arxiv.org/abs/1412.6980.

McCormack, M., 1990. Seismic trace editing and first break picking using neural networks, in: SEG Technical Program Expanded Abstracts 1990, Society of Exploration Geophysicists, 321–324.

Mora, P. & Tarantola, A., 1990. Seismic imagingseismic imaging: 1024–1033, doi:10.1007/0-387-30752-4_124.

Onajite, E., 2014. Seismic Data Analysis Techniques in Hydrocarbon Exploration, Elsevier, doi:https://doi.org/10.1016/C2013-0-09969-0.

Yang, S. et al., 2018. Seismic trace editing by applying machine learning, 2256–2260.

Yilmaz, O. & Doherty, S., 2001. Seismic data analysis: Processing, Inversion, and Interpretation of Seismic data, vol. 2.