Lucas Ribeiro Neis

Machine Learning na Área do Petróleo: Implementação de Redes Neurais para Aprendizado de Distribuições de Probabilidade

Florianópolis

2019

Lucas Ribeiro Neis

Machine Learning na Área do Petróleo: Implementação de Redes Neurais para Aprendizado de Distribuições de Probabilidade

Monografia submetida ao Programa de Graduação em Ciência da Computação para a obtenção do Grau de Bacharel.

Universidade Federal de Santa Catarina

Departamento de Informática e Estatística

Ciências da Computação

Orientador: Prof. Dr. Mauro Roisenberg

Florianópolis

2019

Neis, Lucas Ribeiro

Machine Learning na Área do Petróleo: Implementação de Redes Neurais para Aprendizado de Distribuições de Probabilidade / Lucas Ribeiro Neis; Orientador: Prof. Dr. Mauro Roisenberg; 2019.

72 p. : il. (algumas color.) ; 30 cm.

Trabalho de Conclusão de Curso de Graduação – Universidade Federal de Santa Catarina, Departamento de Informática e Estatística, Ciência da Computação, Florianópolis, 2019.

Inclui referências

1. Ciência da Computação. 2. Inteligência Artificial. 3. Deep Learning. I. Roisenberg, Mauro. II. Universidade Federal de Santa Catarina. III. Título

Lucas Ribeiro Neis

Machine Learning na Área do Petróleo: Implementação de Redes Neurais para Aprendizado de Distribuições de Probabilidade

Esta Monografia foi julgada aprovada para a obtenção do Título de "Bacharel em Ciência da Computação", e aprovada em sua forma final pelo Programa de Graduação em Ciência da Computação.

Florianópolis, Data indefinida:

Prof. José Francisco D. de G. C. Fletes
Coordenador do Curso
Banca Examinadora:

Prof. Dr. Mauro Roisenberg
Orientador

Prof. Dra. Silvia Modesto Nassar
Universidade Federal de Santa Catarina

Prof. Dr. Rafael de Santiago
Universidade Federal de Santa Catarina

Florianópolis
2019

Este trabalho é dedicado aos meus pais, Zenaide Ribeiro e João Neis, pessoas incríveis que me possibilitaram e sempre me incentivaram a realizar meus sonhos.

Agradecimentos

Em primeiro lugar, agradeço aos meus familiares, pois são as pessoas mais inestimáveis que tenho em minha vida. São as pessoas que mais ajudaram a moldar meus valores e a eles eu devo muito.

Agradeço também aos meus amigos, tanto aqueles que eu conheço há vinte anos quanto aqueles que eu conheci nessa universidade. Pessoas tão diferentes que me deram nova perspectiva de mundo e me ajudaram a encontrar novas abordagens para cada momento difícil.

Agradeço ao professores que tive ao longo de toda minha jornada. Pessoas que me fizeram amar e apreciar as ciências, história, e tantos outros campos. Em especial, gostaria de agradecer os professores da UFSC que foram sempre tão atenciosos. Além disso, agradeço ao professor Mauro Roisenberg pela orientação neste trabalho.

Aos meus colegas de trabalho, agradeço por serem pessoas que estão sempre dispostas a ajudar. Pela ajuda contínua nas dúvidas que tive durante este trabalho, agradeço Fernando Luis Bordignon, Rafael Lunelli, Tcharlies Dejandir Schmitz e Vinicius Couto Biermann.

Resumo

A exploração e produção de petróleo usa de diversos métodos para adquirir, estudar e preparar dados de um determinado local de interesse. Na procura de adquirir dados geológicos, o método de poço exploratório é usado para capturar diversos dados relevantes ao contexto de exploração. Todavia, a perfuração de poços exploratórios é cara e é comum a necessidade de diversos poços para adquirir uma quantidade útil de dados. Assim, métodos para simular esses dados foram criados para diminuir o número de poços exploratórios a serem perfurados. Usando dados obtidos em poços exploratórios, uma implementação para simulação geradora de dados de *logs* através do uso de Redes Neurais Artificiais se mostra possível, no entanto, os modelos de redes neurais artificiais mais tradicionais implementam modelos funcionais que tendem a representar apenas a média de um grupo de dados gerando resultados que compreendem apenas uma possível resposta para cada variável de entrada. Aplicações podem precisar de uma distribuição de de probabilidade da variável de saída que seja condicionada à uma variável de entrada. Neste trabalho, foi realizado um estudo e uma implementação de uma rede neural artificial que usa de misturas de gaussianas — proposta em meados dos anos noventa por Christopher M. Bishop — para modelar a distribuição de probabilidade para uma variável de saída. Os resultados obtidos demonstraram que o modelo foi capaz de aderir o comportamento e a distribuição de densidade dos dados originais, mesmo com o aumento da complexidade ao adicionar mais de um tipo de *log* na entrada.

Palavras-chaves: Modelo de Densidade de Mistura. Pseudo Poços. Well Logs.

Abstract

The Oil Exploration and Production field has several means to acquire, study and use data gathered from a place of interest. In the search for geologic data gathering, exploration wells are used as means to collect several useful data in the exploration context. However, well drilling is not cheap and it's commom to need for more than a single well to collect all the relevant data. As such, simulation methods have been recieving attention as a possible approach to get around the mutiple well drilling problem by lowering the total wells required. Although using Artificial Neural Networks seems promising for this field, the traditional Neural Networks are usually functional which tends to only represent a cluster of data by its mean. Since in practical terms an input conditioned probability distribution of the output might be useful, in this work a study was followed by an implementation of a Mixture Densisity Model — An Artificial Neural Network that uses Gaussian Mixtures as tool to capture the probability distribution — in the Oil E&P context for log generation simulation once it captures the probability density for an output variable. The results showed that the method is able to mantain grip on the original data behavior while also creating a proper density distribution even after increase in complexity by expansion on the number of input logs.

Key-words: Mixture Density Networks. Pseudo Wells. Well Logs.

Lista de Figuras

Lista de Abreviaturas e Siglas

IA	*Inteligência Artificial*
RNA	*Rede Neural Artificial*
RDM	*Rede de Densidade de Misturas*
MMQ	*Método dos Mínimos Quadrados*
RNMGI	*Redes Neurais de Misturas Gaussianas Incrementais*
MMG	*Modelo de Misturas Gaussianas*
PMC	*Perceptron Multicamadas*
SSG	*Simulação Sequencial Gaussiana*
V_P	*Velocidade da onda P*
V_S	*Velocidade da onda S*
ρ	*Densidade*
L3C	*Laboratório de Conexionismo e Ciências Cognitivas*
NaN	*Not a Number (Não é um Número)*

Sumário

1 Introdução

O conceito de Redes Neurais Artificiais é creditado a McCulloch e Pitts (1943), e com a hipótese dada por HEBB (1949) para aprendizado de máquina, a fundação fora criada. Diversos novos trabalhos ao longo dos anos como o de Rosenblatt (1958) introduzindo o conceitos de *perceptron* e o de Werbos (1975) sugerindo a possibilidade do uso de *back propagation* em redes neurais deram inicio a ferramentas usadas até hoje, além de diversos outros trabalhos importantíssimos que compõe as Redes Neurais Artificiais Modernas.

As Redes Neurais Artificiais (RNA) são modelos matemáticos cujos parâmetros podem ser adquiridos através de dados conhecidos do problema. Uma das grandes deficiências dessa metodologia é que esta captura de forma limitada as incertezas ou variâncias contidas nos dados experimentais. Outra deficiência aparece quando nos dados experimentais ocorrem situações em que a variável independente produz duas ou mais respostas diferentes (*e.g.* encontrar o valor em x que gera $y = 4$ na função $y = x^2$) (BISHOP, 2006).

No entanto, as Redes de Densidade de Misturas (RDM) (BISHOP, 1994) propõe contornar o problema de uma RNA clássica ao não tentar modelar uma função geradora interna, mas sim capturar a verossimilhança dos dados experimentais, representado-os através de um conjunto de modelos de misturas gaussianas. Com isso, o resultado desta rede é uma representação da probabilidade de haver um ponto em uma certa região do contradomínio para cada ponto amostral do domínio.

Uma possível aplicação para RDMs é a de simulação na área de petróleo, pois existe a necessidade de se capturar a correlação estatística dos dados. Devido a natureza geológica do problema, para extrair petróleo, uma área de interesse precisa ser estudada. Métodos de aquisição de dados (*e.g.* Aquisição sísmica, Poços de Exploração (SCHLUMBERGER, 2019)) são usados para, com os dados obtidos, entender a relevância de uma região.

A simulação de poços dá-se através do uso de dados experimentais coletados em poços de exploração. Tais dados são denominados *logs*, que são medidas em função do tempo e/ou profundidade de uma ou mais características físicas de um poço ou ao redor dele (SCHLUMBERGER, 2019). *Logs* são úteis para simulação devido suas intercorrelações que podem ser exploradas para que os dados obtidos sejam mais precisos.

Well Log ou *Log* de poço é uma medição de alguma característica física de um poço ou de uma área ao redor deste em função do tempo, da profundidade ou de ambos. O termo tem origem da palavra em inglês '*log*' sendo usada com sentido de 'registro' (SCHLUMBERGER, 2019).

Logs não são necessariamente bem representados por funções e podem aparecer em um intervalo. Mas, graças a suas intercorrelações, pode-se usar um *log* para calcular um intervalo ótimo onde existe a verossimilhança de um dado aparecer.

A implementação de uma rede neural que capture a distribuição de probabilidade de uma variável de saída que seja condicionada a uma variável de entrada mostra-se viável para a resolução do problema. Neste trabalho, uma implementação que use RDM cuja finalidade é lidar com a distribuição de probabilidade da correlação de dados de perfuração (*e.g.* velocidades sísmicas) foi realizada.

1.1 Objetivos

Pode-se, com isso, dividir os objetivos como segue:

- Objetivo Geral
  - Treinar e analisar a aplicação de uma rede neural artificial que seja capaz de, dado uma variável condicionada a uma variável de entrada, entregar como resultado uma distribuição de probabilidade.
- Objetivos Específicos
  - Propor como treinar uma RDMs para considerar a distribuição de probabilidade e a correlação entre logs;
  - Amostragem de modelos de misturas gaussianas geradas por RDMs.
  - Realizar uma análise comparativa dos resultados obtidos com o modelo proposto com o a Simulação Sequencial Gaussiana por Redes Neurais de Misturas Gaussianas Incrementais.

1.2 Metodologia de Pesquisa

Em uma primeira etapa, foi estudado como redes neurais funcionam, seu estado da arte e possíveis maneiras de implementar com tecnologias amplamente disponíveis. As principais fontes encontradas foram Bishop (2006) e Ng (2018).

Ao estudar sobre RDMs, estas mostraram-se com alto potencial de resolver o problema proposto, visto a capacidade de descrever múltiplas concentrações de dados em um único ponto do domínio além de capturar um certo grau das variâncias. Com isso, Ha (2015) proveu um bom exemplo de uma simples implementação de uso de RDM e tornou-se importante para melhor entender como funcionam.

Posteriormente, usou-se o conhecimento teórico adquirido na primeira etapa com o propósito de implementar em `MATLAB`® uma RDM capaz de resolver o problema proposto,

estudar seus resultados e comparar os resultados obtidos por Simulação Sequencial Gaussiana por Redes Neurais de Misturas Gaussianas Incrementais. Para treinar a rede, houve a necessidade de os dados receberem tratamentos com o objetivo de não responsabilizar o treinamento de balancear o peso de números em escalas diferentes. Isso foi feito através da normalização destes dados.

Os resultados obtidos foram comparados através de coeficiente de correlação entre o histograma obtido pelos dados experimentais e o histograma dos dados obtidos pela amostragem do conjunto de modelos de misturas gaussianas. Além disso, como o objetivo é obter um resultado que se assemelhe aos *logs* de treino, *crossplots* compararão se os dados se assemelham.

1.3 Organização Textual

No Capítulo 2, são apresentados os conceitos bibliográficos estudados no contexto de Redes Neurais Artificiais, explicando-os de maneira rápida. Uma Rede Neural Artificial geradora de Misturas Gaussianas proposta por Bishop (1994) capaz de capturar um distribuição de probabilidades também receberá a mesma atenção.

O Capítulo 3 introduz os conceitos de Pseudo-Poços e *logs*, seus dados — que foram usados nos experimentos—, alguns tratamentos para eles, além de ferramentas e um breve apresentação da proposta. Os resultados obtidos e os experimentos que levaram até eles são explicados no capítulo 4, onde também se apresenta implementações e comparações com o método de Simulação Sequencial Gaussiana por Redes Neurais de Misturas Gaussianas Incrementais.

Finalmente, o capitulo 5 traz as conclusões obtidas além de quais os possíveis próximos passos uma vez que os resultados obtidos são satisfatórios.

2 Revisão Bibliográfica

Neste Capítulo, são analisados tópicos pertinentes ao domínio do problema e para a solução proposta. RNA são brevemente descritas para que se tenha a base para descrever uma RDM, visto a direta dependência. Além disso, uma rápida exposição do domínio do problema — Poços e seus *Logs* — é descrito.

2.1 Redes Neurais Artificiais

Redes Neurais Artificiais foram inicialmente idealizadas procurando ser uma implementação algorítmica simplificada de como uma Rede Neural Biológica funciona. No entanto, o uso de realismo biológico para se resolver problemas de reconhecimento de padrões traz consigo diversas dificuldades e imposições desnecessárias (BISHOP, 2006). Da base hipotética biológica, RNAs herdam sua estrutura baseada em neurônios com finalidade de simular o funcionamento de um ou de uma rede deles.

2.1.1 Base Teórica

Uma maneira simples de entender o funcionamento de um neurônio sob a ótica de RNAs é que ele é uma unidade computacional que recebe uma entrada, faz uma certa computação e entrega uma saída. Neurônios se comunicam entre si emitindo pulsos de eletricidade e passando através de seus terminais de axônio aos dendrites de um segundo neurônio (NG, 2018). Em suma, RNAs usam de um modelagem simples de como um ou uma rede de neurônios funcionam para criar uma unidade logística.

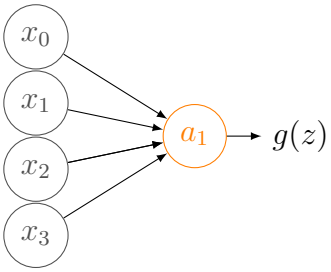

Figura 1 – Unidade logística

Fonte: adaptada de Ng (2018)

Na Figura 1, tem-se um exemplo de um único neurônio onde x_i são as entradas, a

é o 'corpo' do neurônio e $g(z)$ é o resultado da seguinte equação:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{2.1}$$

onde $\mathbf{z}$ é um vetor dado por:

$$\mathbf{z} = \mathbf{xa} + \mathbf{b} \tag{2.2}$$

tal que $\mathbf{x}$ é um vetor de entrada, $\mathbf{a}$ é um vetor de peso da rede e $\mathbf{b}$ é um vetor de ruído dado para a entrada. Mais adiante, $g(\mathbf{z})$ será revisitada para melhor explicar o comportamento desejado de um neurônio. $g(\mathbf{z})$ é chamada de função de ativação e, neste caso, é uma sigmoide. Sigmoides são comumente usadas em RNAs e, por isso, é o exemplo escolhido aqui para representar o funcionamento de neurônios, no entanto, outras funções podem ser usadas para diferentes problemas com diferentes graus de aceitação.

Redes Neurais, portanto, são um grupo de diferentes neurônios arranjados de diversas maneiras — ou arquiteturas — de modo a processar algo (NG, 2018). Esses neurônios costumam ser dispostos em camadas. Comumente, nominadas em camada de entrada, camada(s) oculta(s) e camada de saída. Também é importante notar que neurônios podem ser denominados *ativação da camada m*. Essa nomenclatura será relevante na seção 2.2. Na Figura 2, há uma versão simplificada de uma rede:

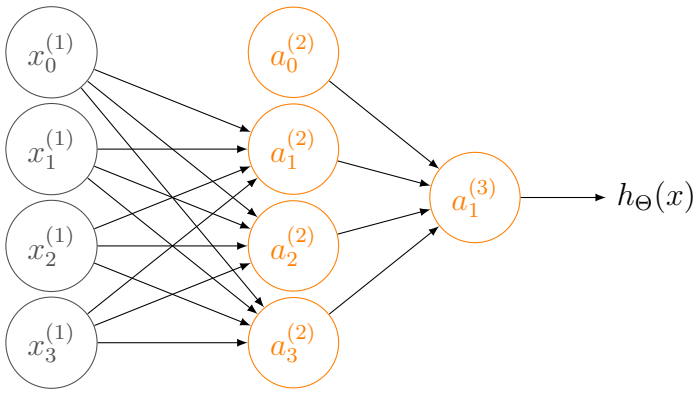

Figura 2 – Rede Neural

Fonte: adaptada de Ng (2018)

Assim, tem-se que $h_\Theta(\mathbf{x})$ pode ser definida como:

$$a_0^{(2)} = 1 \tag{2.3}$$

$$a_1^{(2)} = g(\Theta_{10}^{(2)} x_0 + \Theta_{11}^{(2)} x_1 + \Theta_{12}^{(2)} x_2 + \Theta_{13}^{(2)} x_3) \tag{2.4}$$

$$a_2^{(2)} = g(\Theta_{20}^{(2)} x_0 + \Theta_{21}^{(2)} x_1 + \Theta_{22}^{(2)} x_2 + \Theta_{23}^{(2)} x_3) \tag{2.5}$$

$$a_3^{(2)} = g(\Theta_{30}^{(2)} x_0 + \Theta_{31}^{(2)} x_1 + \Theta_{32}^{(2)} x_2 + \Theta_{33}^{(2)} x_3) \tag{2.6}$$

$$h_\Theta(\mathbf{x}) = a_1^3 = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}) \tag{2.7}$$

dado que $a_i^{(j)}$ é a ativação da unidade i na camada j e $\Theta^{(j)}$ é uma matriz contendo os pesos de controle da função de mapeamento de uma camada j a uma camada $j+1$. Na camada de entrada, x_0 é considerado o ruído de entrada. Já na camada oculta, $a_0^{(2)}$ é conhecido como o viés da rede na primeira camada oculta.

Com isso, pode-se redefinir:

$$z_i^{(j)} = \Theta_{i0}^{(j)} x_0 + \Theta_{i1}^{(j)} x_1 + \Theta_{i2}^{(j)} x_2 + \Theta_{i3}^{(j)} x_3 \tag{2.8}$$

$$a_i^{(j)} = g(z_i^{(j)}) \tag{2.9}$$

$$h_\Theta(\mathbf{x}) = a_1^{(3)} = g(z^{(3)}) \tag{2.10}$$

O processo para calcular a saída final — $h_\Theta(\mathbf{x})$ é chamado de *Forward Propagation*, pois pega-se a entrada, *passa-se adiante* para calcular a ativação da primeira camada, seguindo até que se calcule a ativação de saída (NG, 2018).

A implementação computacional de redes neurais, por questões de desempenho acaba sendo feita de maneira vetorial. Portanto, a implementação básica de um modelo de redes neurais pode ser caracterizadas como uma série de transformações funcionais (BISHOP, 2006).

2.1.2 Treinamento

Em 2.1.1, RNs foram descritas sumariamente como uma classe de funções paramétricas não-lineares onde um conjunto — ou vetor — x é dado como entrada e um vetor y é esperado como saída. No entanto, é necessário encontrar os parâmetros da rede para que esta descreva corretamente o problema. Para isso, a maneira mais simples é usando o método dos mínimos quadrados (MMQ). Dado o conjunto de treinamento contendo valores em um vetor de entrada x_n e um vetor de valores t_n onde um t_i esperado representa a saída para um dado x_i e Θ é um vetor contendo todos os pesos a serem utilizados durante o processo de treinamento da rede, é necessário minimizar a função 2.11 (BISHOP, 2006):

$$E = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \Theta) - t_n\}^2 \tag{2.11}$$

2.1.3 *Error Back Propagation*

Como mostrando em 2.1.1, uma RN propaga o processamento de dados até se calcular a saída. No entanto, de modo a encontrar os melhores pesos e parâmetros, usa-se o que se chama de *Back Propagation*. De modo geral, o treinamento é implementado de maneira iterativa tentando minimizar uma função de erro — tal qual a apresentada em 2.1.2 — ajustando a cada iteração os valores de pesos. Cada uma dessas iterações avalia a derivada da função de erro em relação aos pesos e a derivada é então usada para calcular os ajustes dos pesos. Quando originalmente proposto, *error back propagation* foi

usado junto ao método de gradiente descendente, que procura o mínimo global através de pequenos passos de tamanhos variáveis (BISHOP, 2006).

Usando *error back propagation* é possível encontrar os parâmetros que descrevem com um erro aceitável[1] um dado problema. Idealmente, a cada iteração do procedimento, os parâmetros passam a descrever cada vez melhor a função desejada e é possível ter resultados muito próximos dos esperados.

2.2 Redes de Densidades de Misturas

Apesar da capacidade excepcional de RNAs de aproximarem adequadamente problemas de classificação, elas mostram-se ineficazes de entregar o mesmo tipo de resultado para problemas que necessitem da predição de variáveis continuas. Um exemplo notável é o caso de solução de problemas de inversão (*i.e.* encontrar o valor no domínio quando já se tem o valor do contradomínio). Isso acontece devido à minimização do métodos dos mínimos quadrados levar a saídas aproximadas de médias condicionais aos dados desejados. Haja visto que problemas de inversão podem ter soluções multi-valoradas, a média de várias possíveis soluções não implica necessariamente em uma saída válida (BISHOP, 1994).

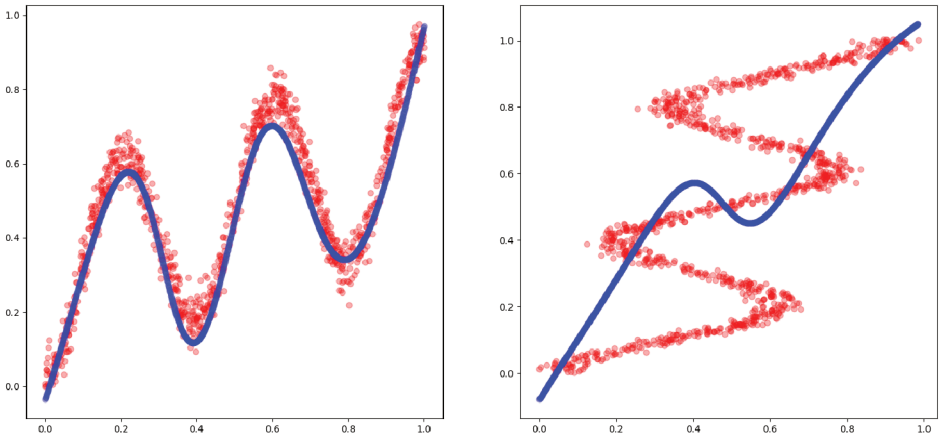

Figura 3 – Exemplos de predições de Redes Neurais

Fonte: Adaptada de Ha (2015)

Na Figura 3, na esquerda pode-se ver um exemplo onde uma RNA (em azul) é capaz de predizer com um grau de erro aceitável as saídas esperadas. No entanto, na

[1] um valor aceitável de erro varia dado o problema e pode ser melhor definido pelo seu respectivo domínio

direita vê-se o problema inverso ao anterior e a RNA (também em azul) falhando em predizer as saídas.

Com isso, em Bishop (1994), foi apresentada uma solução para tal problema. Para tanto, foi usada uma RNA cujo objetivo seria gerar parâmetros para um Modelo de Misturas para substituir a distribuição probabilística usadas intrinsecamente pelas RNAs, criando, com isso, as Redes de Densidades de Misturas (RDMs). Sendo assim, uma RDM é composta por uma RNA cujo objetivo é encontrar os parâmetros de um Modelo Misturas Gaussianas.

2.2.1 Base Teórica

Devido à necessidade de se contornar a possibilidade multi-valorada ao longo do domínio, encontrar a probabilidade de cada um dos possíveis valores no contra-domínio para seu respectivo valor do domínio mostra-se importante para tornar possível a implementação do modelo do gerador intrínseco dos dados apresentados.

Em outras palavras, uma RDM utiliza uma RNA para encontrar os coeficientes de mistura e a densidade de componentes que geram um Modelo de Misturas capaz de descrever $p(t|x)$ adequadamente usando x como entrada (BISHOP, 2006).

Bishop (2006) expande a ideia:

> *[...] For any given value of x, the mixture model provides a general formalism for modelling an arbitrary conditional density function $p(t|x)$. Provided we consider a sufficiently flexible network, we then have a framework for approximating arbitrary conditional distributions.*[2]

A Gaussiana usada será da forma[3]:

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \tag{2.12}$$

tal que x é o ponto no domínio, μ é a média e σ^2 é a variância.

O objetivo é, portanto, que a equação 2.13 corretamente corresponda a $p(\mathbf{t}|\mathbf{x})$:

$$p(\mathbf{t}|\mathbf{x}) = \sum_{k=1}^{K} \pi_k(\mathbf{x})\mathcal{N}(\mathbf{t}|\mu_k(\mathbf{x}),\sigma_k(\mathbf{x})^2) \tag{2.13}$$

[2] Tradução do Autor: [...] Para qualquer dado valor de x, o modelo de misturas provê um formalismo genérico para modelar uma função de densidade condicional $p(t|x)$. Dado que consideremos uma rede suficientemente flexível, nós podemos então ter um *framework* para aproximar distribuições condicionais arbitrárias.

[3] Apesar de apresentada como usando um modelo Gaussiano, RDMs podem ser implementadas com outras distribuições caso necessário (BISHOP, 1994).

Tem-se, com isso, os parâmetros do Modelo de Mistura. São eles os já conhecidos μ e σ^2 adicionados de π que representa o coeficiente de mistura (*i.e.* o 'peso'). Além disso, K representa o número total de componentes na mistura. Tais parâmetros precisam ser funções genéricas condicionadas a x, o que é facilmente realizável usando a RNA onde x é a entrada (BISHOP, 1994).

Ao escolher um Modelo de Misturas com funções componentes — ou funções *kernel* — suficientes, e uma RNA com um número de camadas ocultas apropriado, a RDM pode aproximar qualquer densidade condicional $p(\mathbf{t}|\mathbf{x})$ (BISHOP, 1994).

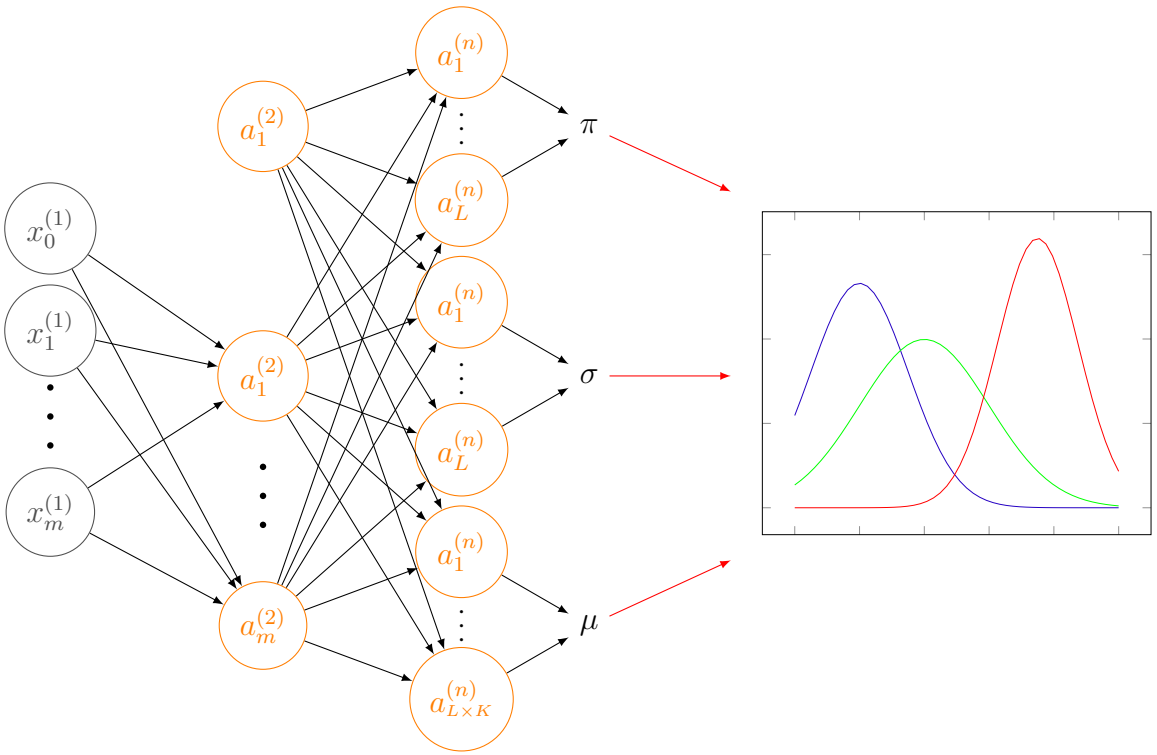

Figura 4 – Rede de Mistura de Densidades

Fonte: adaptada de Bishop (1994) e Bishop (2006)

A Figura 4 mostra resumidamente a estrutura de uma RDM. A RNA presente na RDM pode ter, por exemplo, duas camadas ocultas cujo comportamento é sigmoidal (BISHOP, 2006). Caso o Modelo de Misturas 2.13 tenha um total de componentes igual a L e se o resultado alvo $\mathbf{t}$ tem K dimensões, a RNA então precisará de um número total de $M = (K + 2) \times L$ ativações na camada de saída. Essas M ativações serão alocadas da seguinte maneira (BISHOP, 2006):

- L serão destinadas a a_k^{π} que serão usadas para calcular os coeficientes da mistura $\pi_k(\mathbf{x})$;
- L serão destinadas a a_k^{σ} que serão usadas para definir as larguras de *kernel* $\sigma_k(\mathbf{x})$;

- $L \times K$ serão destinadas a a_{kj}^{μ} que serão usadas para definir as componentes $\mu_{ij}(\mathbf{x})$ dos centros de *kernel* $\mu_{\mathbf{k}}(\mathbf{x})$.

Diferente das K saídas esperadas por RNA que predizem as médias condicionais das variáveis desejadas, RDMs precisam entregar três tipos distintos de parâmetros K vezes, além de, em casos onde $L > 1$, ter um desses parâmetros sendo tratado como uma matriz.

Algumas restrições que precisam ser impostas sobre os parâmetros, no entanto. São elas:

$$\sum_{k=1}^{K} \pi_k(\mathbf{x}) = 1, \qquad 0 \leq \pi_k(\mathbf{x}) \leq 1 \tag{2.14}$$

Para garantir 2.14, é usado um conjunto de saídas de *softmax*:

$$\pi_k(\mathbf{x}) = \frac{\exp(a_k^{\pi})}{\sum_{l=1}^{K} \exp(a_l^{\pi})} \tag{2.15}$$

Já as variâncias precisam respeitar $\sigma_k^2 \geq 0$. Para tanto, usa-se exponenciais nas saídas da RNA correspondentes:

$$\sigma_k(\mathbf{x}) = \exp(a_k^{\sigma}) \tag{2.16}$$

Por fim, $\mu_{kj}(\mathbf{x})$ tem componentes reais, nenhuma imposição precisa ser feita e, por seguinte, pode usar a própria saída correspondente da RNA:

$$\mu_{kj}(\mathbf{x}) = a_{kj}^{\mu}(\mathbf{x}) \tag{2.17}$$

2.2.2 Treinamento de uma Rede de Densidade de Misturas

Como visto em 2.1.2, RNAs comumente usam de uma minimização de uma função de erro. Apesar de RNAs normalmente usarem o famoso método dos mínimos quadrados, RDMs precisam que os parâmetros do seu Modelo de Misturas estejam envolvidos, removendo assim a possibilidade de uso dela. O objetivo da RNA dentro da RDM é encontrar três tipos de parâmetro para um Modelo de Mistura quando dado um $\mathbf{x}$. E, como coloca Bishop (2006):

> *The adaptive parameters of the mixture density network comprise the vector* $\mathbf{w}$ *of weights and biases in the neural network, that can be set by maximum*

> *likelihood, or equivalently by minimizing an error function defined to be the negative logarithm of the likelihood.*[4]

precisa levar em consideração os pesos e vieses da rede. Para isso, define-se a função de erro como (BISHOP, 2006):

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k(\mathbf{x_n}, \mathbf{w}) \mathcal{N}(\mathbf{t_n} | \mu_{\mathbf{k}}(\mathbf{x_n}, \mathbf{w}), \sigma_k^2(\mathbf{x_n}, \mathbf{w})) \right\} \tag{2.18}$$

onde $\mathbf{w}$ é deixado explicito. Para se minimizar a função de erro, calcular as derivadas do erro E condicionado por $\mathbf{w}$ na rede neural é necessário. A avaliação dele pode ser feita usando o procedimento de *back propagation* (2.1.3), dado que as derivadas sejam apropriadas para a função de erro levando as ativações de saída em consideração. (BISHOP, 1994)

Portanto, as ativações de camada de saída para os coeficientes de mistura são dadas pelas seguintes derivadas:

$$\frac{\partial E_n}{\partial a_k^{\pi}} = \pi_k - \gamma_k \tag{2.19}$$

onde, por conveniência, π_k passa a ser tratado como as probabilidades a priori dependentes de $\mathbf{x}$ e γ_k é dado por:

$$\gamma_k(\mathbf{t}|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{t_n} | \mu_{\mathbf{k}}(\mathbf{x_n}), \sigma_{\mathbf{k}}^{\mathbf{2}}(\mathbf{x_n}))}{\sum_{l=1}^{K} \pi_l \mathcal{N}(\mathbf{t_n} | \mu_{\mathbf{l}}(\mathbf{x_n}), \sigma_{\mathbf{l}}^{\mathbf{2}}(\mathbf{x_n}))} \tag{2.20}$$

e são as probabilidades a posteriori correspondentes. (BISHOP, 2006)

Já as ativações da camada de saída para as médias de componentes são dadas pelas seguintes derivadas (BISHOP, 2006):

$$\frac{\partial E_n}{\partial a_{kl}^{\mu}} = \gamma_k \left\{ \frac{\mu_{kl} - t_l}{\sigma_k^2} \right\} \tag{2.21}$$

Por fim, as ativações da camada de saída para as variâncias das componentes são dadas pelas seguintes derivadas (BISHOP, 2006):

$$\frac{\partial E_n}{\partial a_k^{\sigma}} = -\gamma_k \left\{ \frac{||\mathbf{t} - \mu_{\mathbf{k}}||^2}{\sigma_k^3} - \frac{1}{\sigma_k} \right\} \tag{2.22}$$

Na Figura 5, uma RDM foi implementada para resolver o mesmo problema apresentado no começo desta seção. É possível notar que a RDM foi capaz de aproximar adequadamente os dados, e mostra-se, de fato, promissora para resolver problemas desse tipo.

[4] Tradução do autor: Os parâmetros adaptativos da rede de densidade de misturas contém o vetor $\mathbf{w}$ de pesos e vieses na rede neural que podem ser definidos pela máxima verossimilhança, ou equivalentemente ao minimizar uma função de erro dada pelo logaritmo negativo da verossimilhança.

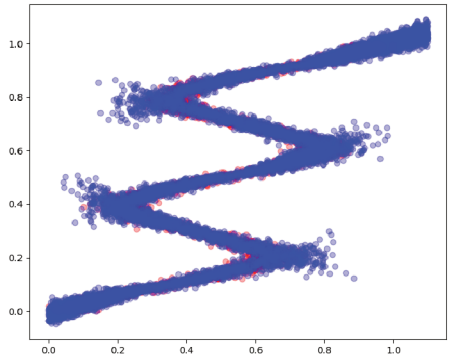

Figura 5 – Exemplos de predições de Redes Neurais

Fonte: Adaptada de Ha (2015)

2.3 Trabalhos Correlatos

Em Soares, Neto e Roisenberg (2016), os autores propuseram uma melhoria para Redes Neurais de Misturas Gaussianas Incrementais (RNMGI), conseguindo, assim, melhor desempenho que os obtidos por *Perceptron* Multicamadas (PMC) e, também, em relação à versão original da própria RNMGI. Além disso, propuseram um algoritmo para Simulação Sequencial Gaussiana (SSG), de forma a substituir a etapa de *krigagem* com o uso da RNMGI proposta.

Krigagem é uma método de interpolação que enfatiza a reprodução de médias e correlações espaciais dos dados, em vez de simplesmente reproduzir a variabilidade existente (Soares; Neto; Roisenberg, 2016). O método é usado para calcular SSG que é muito usado na indústria petrolífera justamente por conta da tentativa de se resgatar a correlação espacial. O problema encontrado na *Krigagem* e, assim, no SSG, é a necessidade de profissionais capacitados para análise e modelagem de variogramas. Nesse sentido, a implementação proposta não carece de parametrizações de modelos de variograma para a simulação, sendo essa etapa substituída pela maquinaria interna da RNMGI.

Os autores concluem com um resultado positivo, em que o modelo proposto é capaz de gerar resultados próximos daqueles gerados por *krigagem*, mas com a vantagem de não precisar de um especialista. O algoritmo de SSG por RNMGI proposto é capaz de honrar as médias mantendo a correlação espacial dos dados usados.

3 Dados e Procedimentos

Uma vez que o modelo de uma RDM já foi apresentado, é necessário abordar o problema o qual este trabalho se propõe a resolver. Na indústria de exploração e produção — comumente referenciada por E&P —, diversos são os passos necessários para estudar uma região, normalmente na procura de reservas. Devido ao incorrimento do custo necessário para se explorar detalhadamente uma área através de perfurações. Métodos foram desenvolvidos para simular os dados contidos nesses poços. Um desses métodos é a simulação de pseudo-poços. Este método busca usar dados de poços já perfurados para simular os dados que um poço nessa região poderia apresentar, tornando o número de poços necessários menor. A seção 3.1 descreve brevemente o conceito de pseudo-poços e de como esses dados são armazenados além de trazer em detalhes os três tipos mais importantes de *logs* para este trabalho.

A seção 3.2 apresenta os dados que serão usados para experimentos. Devido como são armazenados, estes dados tem uma estrutura que traz a necessidade de gerenciar a leitura e as estruturas usadas para manter os dados em memória. Além de oferecer uma amostra de como esses dados realmente se apresentam uma vez que fora da teoria, a seção também permite e aponta certos detalhes que os *logs* usados apresentam, como por exemplo a correlação entre eles.

Todavia, os dados precisam receber alguns tratamentos para serem devidamente usados nas ferramentas apresentadas. Certos aspectos como a diferença de escala entre diferentes *logs* podem ser problemas quando treinando uma RNA. No entanto, eles podem ser evitados com uma simples normalização. Além do mais, estes dados serão divididos em dois grupos para testar a eficácia do método. A seção 3.3 traz mais detalhes nestes aspectos. Por fim, as seções 3.4 e 3.5 trazem mais detalhes sobre a implementação e uso da RDM e quais foram as devidas notas tomadas em testes preliminares.

3.1 Pseudo-Poços

Quando se trata de análise de dados sísmicos para exploração de reservas tais como petróleo, o processo de aquisição desses dados pode assumir diversas formas. Visto a variedade de métodos que existem, diferentes maneiras de interpretar esses dados foram conceituados.

Um desses métodos é a perfuração de poços exploratórios. Esses poços são perfurados para se estudar uma região a fim de procurar reservas a serem exploradas, quais o tamanho dessas reservas e, portanto, se vale o custo de se extrair os recursos ali encon-

trados.

Dados sísmicos tornaram-se muito importantes para se desenhar características de dadas reservas em espaço inter-poços (JOSEPH; FOURNIER; VERNASSA, 1999). No entanto, diversos desafios surgem quando se tenta adquirir as propriedades de reservas como coloca Joseph, Fournier e Vernassa (1999):

> *[...] Firstly due to an indirect relationship between the reservoir parameters (lithology, petrophysics, fluid) and its seismic response, and secondly due to the difference of scale between well and seismic data. Thus lithoseismic interpretation requires a calibration of seismic parameters with actual reservoir properties in the vicinity of wells, to establish the potential link between both informations.*[1]

Eventualmente, surgiu a ideia de se gerar pseudo-poços baseados em cenários geológicos esperados de uma reserva, calculando os rastros sísmicos sintéticos para usá-los como pontos de calibração sintéticos (JOSEPH; FOURNIER; VERNASSA, 1999). Diversas abordagens foram tomadas, desde o uso de cadeias de Markov até dados gerados através de métodos de Monte-Carlo.

3.1.1 *Logs* de Poços

A fim de fazer um detalhado registro das formações geológicas encontradas em uma perfuração, *logs* de poços são usados. Os *logs* podem ser adquiridos de algumas diferentes formas tal como amostras visuais trazidas a superfície ou medidas feitas com instrumentos introduzidos no poço (SCHLUMBERGER, 2019).

Logs são amplamente usados e podem ser dados de densidade ou das velocidades sísmicas. Além disso, os dados não são necessariamente da perfuração inteira e podem descrever apenas um trecho de interesse. Alguns *logs* têm relação entre si o que torna interessante para pseudo-poços no caso da simulação deles.

Antes de entender os *logs* que serão usados neste trabalho, é necessário ter uma breve ideia do que são ondas sísmicas. Como descrito em Milsom (2003):

> *A seismic wave is acoustic energy transmitted by vibration of rock particles. Low-energy waves are approximately elastic, leaving the rock mass unchanged*

[1] Tradução do autor: [...] Primeiramente, devido a uma relação indireta entre os parâmetros da reserva (litologia, petrofísica, fluido) e a sua resposta sísmica, e, em segundo lugar, devido à diferença de escala entre poço e dados sísmicos. Então, interpretação litosísmica requer uma calibração de parâmetros sísmicos com propriedades existentes de reservas na vizinhança dos poços, estabelecendo a ligação potencial entre ambas informações.

by their passage, but close to a seismic source the rock may be shattered and permanently distorted.[2]

Duas dessas ondas são interessantes no contexto dos *logs* mencionados:

- Onda-P: Quando uma onda viaja em um meio (*e.g.* ar), as moléculas oscilam para trás e então para frente na direção do transporte de energia. Ou seja, conforme ela viaja causando compressões e rarefações ao empurrar o meio e é de onde vem seu nome — *pressure wave*. É importante destacar que, quando em meio sólido, a onda-P tem a maior velocidade entre os movimentos ondulatórios e é comumente chamada de onda primária (MILSOM, 2003).
- Onda-S: Quando uma onda viaja em meios sólidos, partículas vibrando em ângulo reto a direção do fluxo de energia causam o que é chamado de onda-S (vindo de *Shear wave*). Por ter velocidade baixa, pode ser chamada de onda secundária, comumente, a Onda-S tem velocidade aproximada da metade da Onda-P em várias rochas consolidadas (MILSOM, 2003).

No entanto, apesar de V_P e V_S serem um resultado direto da existência das ondas P e S, elas também dependem diretamente da densidade da rocha (ρ).

3.1.1.1 *Densidade* (ρ)

O *log* de densidade representa a medição das mudanças no campo magnético da Terra gerada pela diferença da densidade de rochas. Sua unidade é dada no Sistema Internacional de Unidades como $kg \times m^{-3}$ mas variações são comumente usadas (MILSOM, 2003).

3.1.1.2 *P-Velocity* (V_P)

As velocidades sísmicas de rochas são as velocidades em que as os movimentos ondulatórios viajam através delas. As velocidades (V) de ondas elásticas podem ser representadas como

$$V_y = \sqrt{(x/\rho)}$$

onde y é a onda a qual se procura saber a velocidade e x é o módulo elástico relevante à onda. (MILSOM, 2003)

[2] Tradução do autor: Uma onda sísmica é energia acústica sendo transmitida através da vibração de partículas de rocha. Ondas de baixa energia são próximas de elásticas, deixando a massa da rocha inalterada pela sua passagem, mas quando próximo a uma fonte sísmica a rocha pode ser partida e permanentemente distorcida.

No caso de V_P, o módulo elástico usado é a elasticidade de alongamento dada por

$$j = K + (4/3)\mu$$

onde K é dado por $compressibilidade^{-1}$ e representa o Módulo Volumétrico e $\mu = \rho V_S^2$ e é denominado Módulo de Cisalhamento. (MAVKO, 2005; MILSOM, 2003)

Sendo assim, tem-se que

$$V_P = \sqrt{j/\rho} \tag{3.1}$$

3.1.1.3 *S-Velocity* (V_S)

Como visto em 3.1.1.2, a fórmula das velocidades de onda é geral e só precisa ser definido qual é o módulo elástico relevante. Para V_S é o já citado Módulo de Cisalhamento (μ). Portanto, tem-se que

$$V_S = \sqrt{\mu/\rho} \tag{3.2}$$

é a fórmula para V_S (MAVKO, 2005; MILSOM, 2003).

3.2 Os Dados Experimentais

Todos os dados utilizados são de poços experimentais e foram cedidos ao Laboratório de Conexionismo e Ciências Cognitivas (L3C) pela Petrobras para projetos de pesquisa. Devido a questões de sigilo, certas informações não foram compartilhadas, tais como localização, nome e etc.. No entanto, para os objetivos propostos, os dados fornecidos são suficientes. Com acesso a diversos *logs* cujo alguns possuem correlações que podem ser exploradas no enfoque deste trabalho.

Os dados de *logs* são salvos num arquivo que engloba todos os dados de um poço em formato **LAS**. O arquivo tem treze *logs* e suas devidas unidades de medida. Os *logs* são arranjados em colunas onde os valores medidos são dispostos. Algumas informações extras como qual o Δx de profundidade para medir os pontos. Além disso, pontos os quais um *log* não foi devidamente medido recebem o valor -999.

Para ler o conteúdo deste arquivo, uma 'micro'-biblioteca fora cedida pelo L3C chamada de LAS_READER cujo principal interface para seu uso é através do arquivo READ_LAS2_FILE.M. A LAS_READER foi implementada em MATLAB® e quando chamada através de READ_LAS2_FILE.M é necessário passar por parâmetro o caminho relativo até o arquivo que se procura ler.

Após lido, recebe-se da função uma STRUCT contendo diversas informações tais como nome do arquivo, pseudônimo do poço e os devidos dados armazenados. Os *logs* são armazenados em uma matriz onde cada coluna representa um *log* diferente. É importante notar também que dados que foram armazenados com valor -999 foram convertidos para **NaN**. Na Seção 3.3 explica-se como os dados precisam ser tratados para que sejam devidamente usados dentro da RDM.

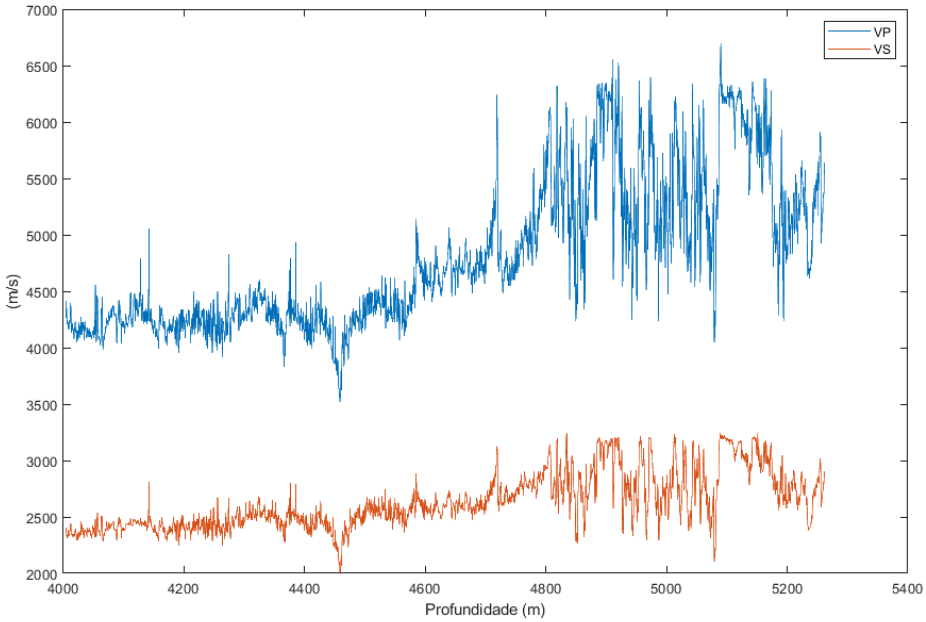

Figura 6 – V_P e V_S ao longo da profundidade

Fonte: Elaborada pelo Autor

A Figura 6 apresenta o comportamento da velocidade das ondas P e S conforme essas são refletidas pelas rochas em um dos poços — com o pseudônimo caricato de *Well 1*. É interessante notar como as curvas são parecidas em termos de amplitudes, apesar da diferença na velocidade propriamente dita, mostrando a correlação presente.

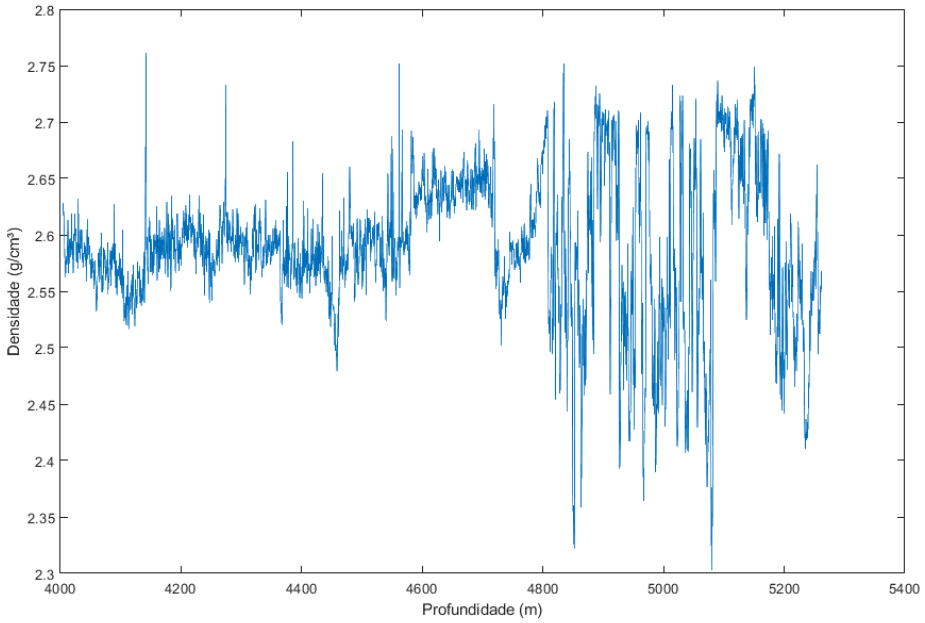

Figura 7 – Densidade ao longo da profundidade

Fonte: Elaborada pelo Autor

Além de V_P e V_S, a densidade também é importante de ser mencionada. A Figura 7 traz ρ em mais detalhe. A diferença de escala entre a densidade e as velocidades previamente apresentadas vai se mostrar importante na próxima seção. Por fim, a Figura 8 apresenta um *crossplot* par-a-par entre os três *logs* de maior enfoque para este trabalho.

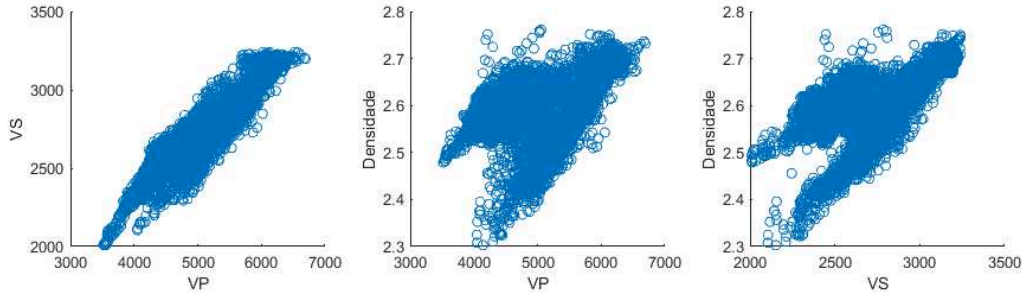

Figura 8 – *Crossplot* de V_PxV_S, V_Pxρ e V_Sxρ

Fonte: Elaborada pelo Autor

3.3 Tratamento dos Dados

Os dados apresentados até aqui, por serem dados experimentais, precisam receber um pouco de atenção antes de se iniciar devidamente a usá-los em MDNs. Um primeiro problema encontrado foi que os *logs* de V_P, V_S e ρ apresentavam pontos onde dados não

haviam sido coletados. Com isso, todos valores que em uma das três variáveis recebiam um valor **NaN** foram removidos de todas elas.

Por fim, como visto na seção 3.2, a densidade não tem a mesma escala que as velocidades e um pequeno prenúncio foi dado. A relevância disso é bem clara quando se leva em consideração RNAs. Devido como o treinamento de RNAs funciona levando em consideração os dados de entrada para formular pesos para variáveis internamente, torna-se problemático colocar dados com tamanha diferença. Conforme o treinamento ocorre, vai ser necessário que a rede procure calcular pesos para compensar essa diferença (NG, 2018).

Portanto, o último tratamento sobre os dados que foi feito é justamente a normalização por `min-max`. A Fórmula 3.3 mostra como ocorre este processo:

$$NormData[i] = \frac{Data[i] - min(Data)}{max(Data) - min(Data)} \tag{3.3}$$

O resultado esperado é que após a normalização todos os *logs* estejam em uma mesma escala e, portanto, haja uma simplificação no processo de treinamento para a rede encontrar seus pesos. Para isso, todos os *logs* foram normalizados. A Figura 9 mostra o mesmo *crossplot* da Figura 8, mas agora com todos os dados no intervalo $[0, 1]$.

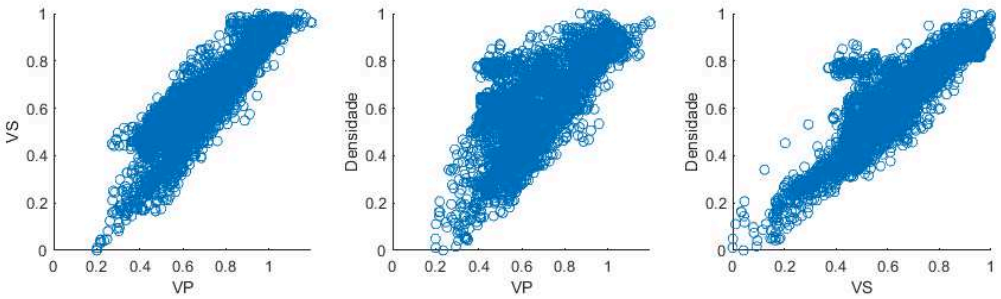

Figura 9 – *Crossplot* de V_PxV_S, V_Pxρ e V_Sxρ normalizados

Fonte: Elaborada pelo Autor

3.4 NETLAB

O NETLAB é um *toolbox* em MATLAB® de funções e *scripts* desenvolvidas por Ian Nabney e Christopher M. Bishop para acompanhar as técnicas e abordagens apresentadas no livro *Pattern Recognition and Machine Learning* (BISHOP, 2006). A biblioteca inclui implementação de análise de dados das quais, na época de sua implementação, muitas ainda não estavam disponíveis em pacotes padrões de simulação de RNAs (NETLAB. . . , 2019).

A escolha de usar o NETLAB foi dada em grande parte por sua implementação de MDNs o qual ainda não está disponível nativamente no MATLAB®. Tentativas foram

feitas para modificar o funcionamento do treinamento das RNAs nativas do MATLAB® para usar a minimização da verossimilhança como descrito por Bishop (1994), no entanto, a biblioteca além de já disponibilizar essa função, para usá-la basta incluir os arquivos no projeto. Sendo assim, ambas as alternativas eram igualmente viáveis e, portanto, a mais simples foi escolhida.

3.5 Distribuição de Probabilidade com RDMs

A situação de interesse para este trabalho se define quando, por exemplo, dados de um poço foram capturados, no entanto, algumas variáveis não estão completamente medidas, seja por problemas na sonda ou devido ao pseudo-poço não ter esses dados simulados. O que se procura é uma maneira de obter esses dados para um poço/pseudo-poço se aproveitado das correlações entre esses *log* das variáveis.

Para tanto, é proposto treinar uma RDM que gere um conjunto de modelos de misturas gaussianas que capturem a distribuição de probabilidade dado uma variável (*e.g.* V_P) ao longo de uma outra (*e.g.* ρ). Uma vez com os modelos gerados, amostragens podem ser feitas em cada um das misturas a fim de gerar uma 'simulação' de *log* que respeite a relação dentre os dados experimentais.

Portanto, espera-se dois modelos de uso dessa implementação, um quando se tem um único *log* de entrada com alvo em um único *log* de saída (*e.g.* a partir de V_P gerar ρ) e a outro partindo de dois *logs* para também um único alvo (*e.g.* a partir de [V_P, V_S] gerar ρ). Devido como o NETLAB funciona, essas implementações podem ser feitas de maneira única. A partir daqui, o primeiro uso será referenciado como 'modelo 1×1' e o segundo como 'modelo 2×1'.

4 Experimentos e Resultados

Agora que todas as ferramentas e dados já foram apresentados, resta explicar a abordagem e os diferentes experimentos e resultados obtidos. Dado que este trabalho precisa fazer amostragem dentro de misturas gaussianas e trabalhar com ferramentas para RNAs e outras utilidades disponíveis, existe a necessidade de implementar funções para facilitar a leitura e o uso do código fonte desenvolvido. A seção 4.1 detalha mais esse assunto e as explica brevemente.

As seções 4.2 e 4.3 mostram os experimentos, seus resultados e uma breve comparações com o método de simulação de SSG por RNMGI. Para mostrar os resultados obtidos, serão mostrados um *crossplot* dos dados experimentais, um de amostragem e um último juntará os dois anteriores de modo a compará-los. Ademais, um histograma comparativo também estará disponível para cada um dos experimentos.

O método de SSG por RNMGI que será usado para comparações neste trabalho foi implementado no L3C para fazer a simulação de *logs* de poços quando estes são bem comportados e separados em grupos denominados *facies*. O método é capaz de manter a correlação espacial, o que é muito importante dentro do domínio que se encontra.

4.1 Implementações

Com o propósito de facilitar a apresentação, algumas funções foram implementadas em arquivos separados. Essas funções são apenas facilitadoras na interação com as bibliotecas e são puramente organizacionais. Na prática, todas elas poderiam estar em apenas um arquivo, mas ao separá-las, facilita-se uso e legibilidade de código.

Para amostrar, a função SAM foi implementada. A função recebe um conjunto de MMGs, o número de amostras por cada MMG que se deseja obter e retorna um *Array* de valores gerados dentro do espaço normalizado, ou seja, se houverem três mil MMGs e pede-se três amostragens por MMG, ter-se-á um *Array* com três mil linhas e três colunas. Quando amostrar em MMGs, precisa-se primeiro levar em consideração qual a componente que será escolhida. Faz-se isto usando a função do MATLAB® denominada RANDSRC que, dado um conjunto de número e suas devidas probabilidades de serem sorteadas, escolhe uma delas levando as probabilidade em conta. Assim sorteia-se um índice que representa o índice da componente sorteada. Com isso, basta usar a função MVNRND para gerar um número aleatório dentro da gaussiana que a componente representa. Em situações que mais de um valor precisa ser sorteado por MMG, a componente precisa ser re-sorteada.

Uma interface entre para a RDM foi criada — denominada EXE_MDN — de modo a facilitar e separar a passagem de dados e a modelagem da rede. A interface recebe o *log* de entrada, o *log* alvo, o número de dimensões na entrada, quantos neurônios na camada escondida, o número de componentes das MMGs, a taxa de aprendizado, a dimensão alvo da saída e as iterações de inicialização (por K-means) e da época. Uma vez chamada, ela aplica os parâmetros que recebeu a função implementada no NETLAB de RDM e, após treinar, extrai as misturas obtidas e as retorna.

4.2 Configuração 1×1

No modelo 1×1, foram feitos um total de cinco experimentos. Quatro para o método proposto e um para o método de simulação de SSG por RNMGI ao qual se comparará aos resultados obtidos das MMGs geradas pela RDM. Os experimentos são feitos da seguinte forma: Se usou do método para gerar ρ dado V_P. Além disso, ainda existe o teste com SSG por RNMGI.

O objetivo deste modelo é mostrar como a implementação se comporta, visto a facilidade de se conseguir *plots* dos resultados. Curvas de contorno podem ser usadas neste caso, pois teria-se um gráfico tridimensional para mostrar as distribuições de probabilidade.

Para gerar as MMGs, foi passado a EXE_MDN os parâmetros para uma RDM com 5 neurônios na camadas ocultas, 5 componentes gaussianas por MMG, 50 de taxa de aprendizado, 50 iterações de inicialização e uma época de 10000 iterações. Em experimentos preliminares, notou-se que essa configuração é capaz de capturar a probabilidade de todos os pares dos *logs* testados de maneira parecida com a que será apresentada a seguir.

A Figura 10 mostra como as distribuições de probabilidade foram capturadas pela RDM em uma curva de níveis. É interessante de se notar a concentração de probabilidade no canto superior direito da imagem em contraste às demais regiões. Usando este conjunto de MMGs, foi feita uma amostragem por MMG. A Figura 11 traz mais detalhes.

A Figura 11a mostra um *scatter* dos *logs* originais. Nota-se que existem agrupamentos em diferentes regiões da densidade. Já na Figura 11b compara-se diretamente os dados originais (em vermelho, ao fundo) e aqueles gerados na amostragem (em azul, à frente). Os pontos são gerados em mesmo número do que haviam no conjunto de dados originais. A Figura mostra com clareza a capacidade da RDM para descrever o comportamento dos dados de aprendizagem mas sem haver uma relação geradora desses pontos. Por fim, a Figura 11c mostra somente os pontos gerados pela amostragem.

No entanto, apesar das figuras demonstrarem que a implementação consegue cap-

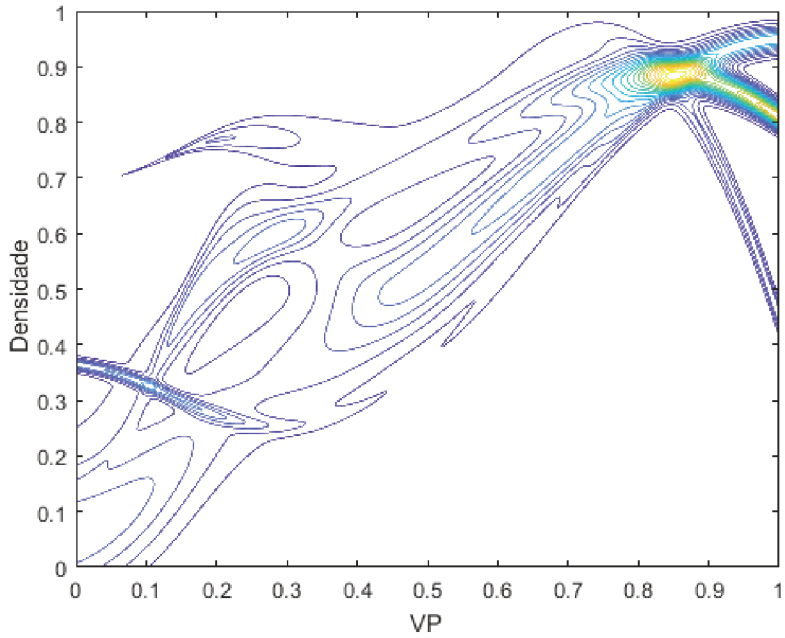

Figura 10 – *Plot* de contornos do conjunto de MMGs

Fonte: Elaborada pelo Autor

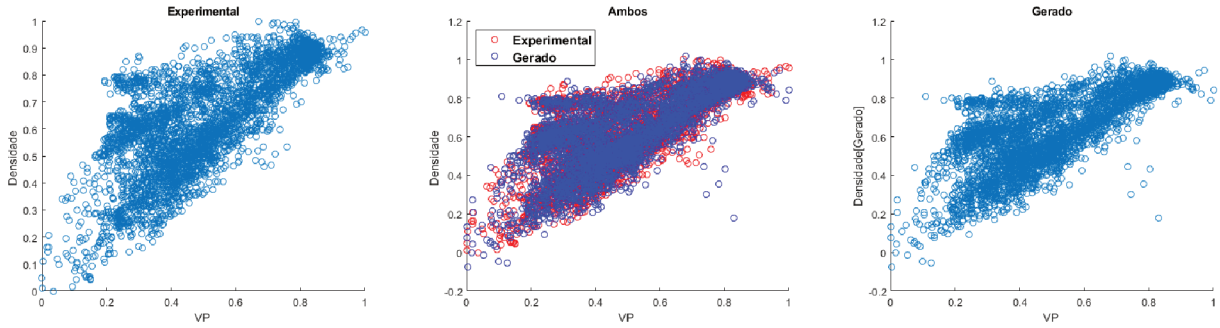

Figura 11 – *Scatter* de $V_P \times \rho$ comparativo

Fonte: Elaborada pelo Autor

turar o comportamento dos dados, ainda é necessário investigar se houve captura da distribuição desses pontos. Para tanto, um histograma comparativo entre os dados empíricos e os amostrados das MMGs pode ser revelador. A Figura 12 traz esse histograma.

Apesar do histograma ser uma maneira interessante de verificar o quanto a amostragem se aproxima da distribuição dos pontos, é mais vantajoso colocar um número nessa aproximação para poder comparar matematicamente com outro método. Para isso, calculou-se o coeficiente de correlação entre os dois histogramas. O valor alcançado é de 0.9943 para estes histogramas.

Os resultados obtidos na SSG por RNMGI com parâmetros *trend range* em 100,

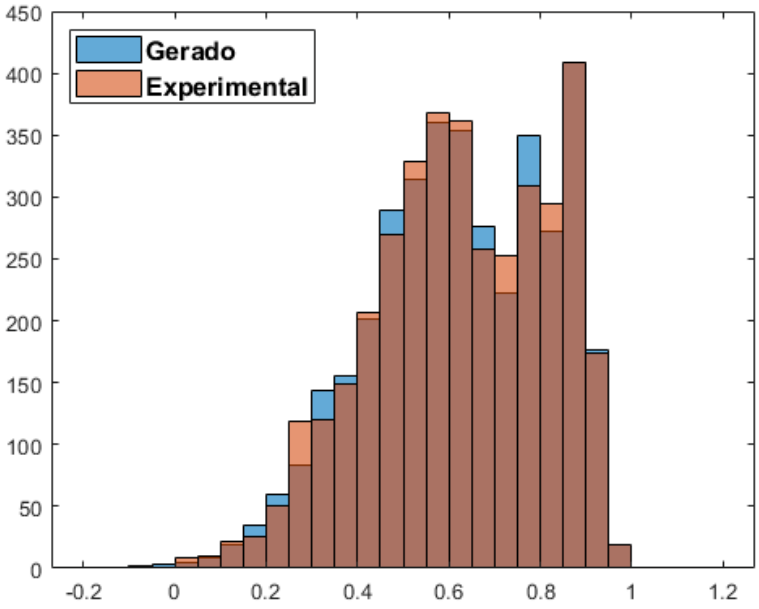

Figura 12 – Histogramas de ρ comparativo

Fonte: Elaborada pelo Autor

stability criteria também em 100, *novelty*(τ) igual a 0.1, *delta* em 0.07 e *minimum samples* é 3. A Figura 13 mostra um resultado de SSG por RNMGI. Diferente dos resultados mostrados anteriormente nesta seção, o comportamento dos valores amostrados não contém a mesma distribuição que os valores dos dados originais. Um importante lembrete é que essa implementação ainda se encontra em fase de testes e seus resultados podem melhorar.

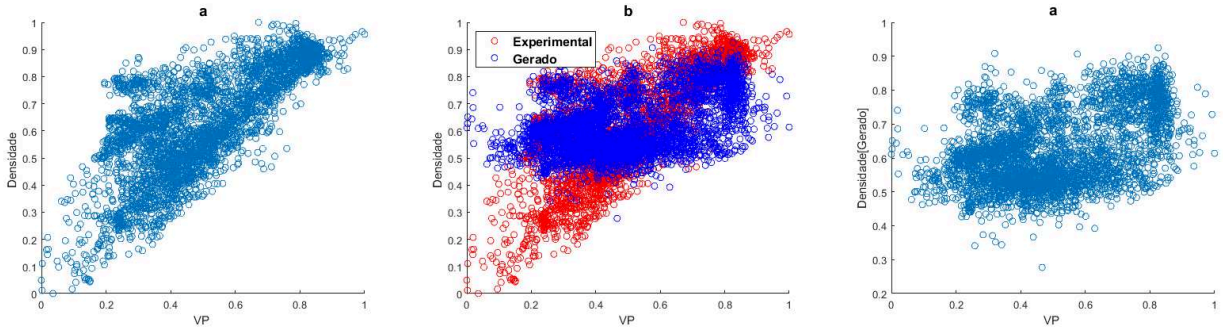

Figura 13 – ρ gerado com SSG por RNMGI

Fonte: Elaborada pelo Autor

Por fim, o histograma dos valores gerados mostra que, além de não capturar o comportamento dos *logs* experimentais, eles também não foi capturado as distribuições de probabilidade. Apesar de parecerem tão diferentes, o coeficiente de correlação dos histogramas é de 0.7681.

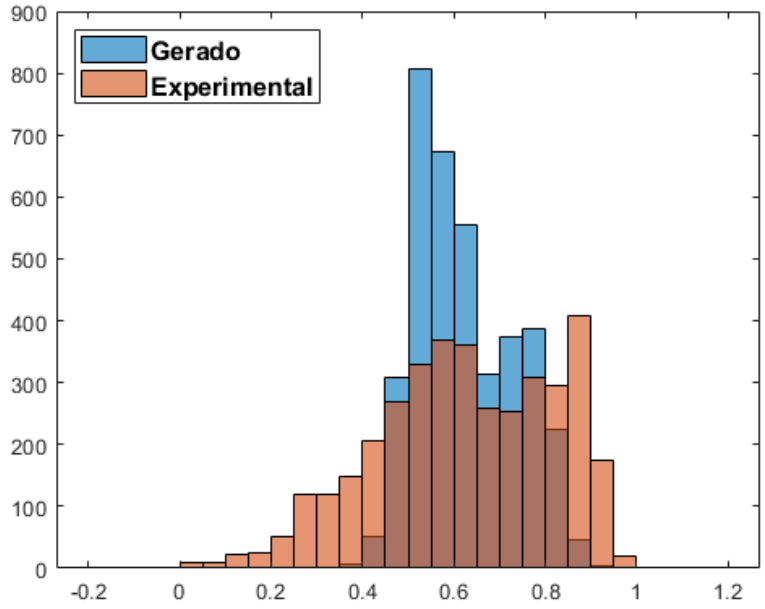

Figura 14 – Histogramas de ρ comparativo gerado com SSG por RNMGI

Fonte: Elaborada pelo Autor

4.3 Configuração 2×1

Com a implementação de 1×1 mostrando uma boa aproximação, pode-se verificar se uma variação do método para 2×1 é prática. Para isso, nenhuma mudança significativa precisou ser feita nas implementação, além de precisar lidar com uma nova dimensão na hora de adquirir os gráficos. Todavia, é notável que a RDM vai precisar de uma nova configuração (*i.e.* quantas componentes na mistura? Quantos neurônios nas camadas ocultas? Etc.), e não é surpreendente que alguns parâmetros recebem alterações.

A maior mudança em ralação ao método 1×1 no âmbito de testes é a forma de visualizar os resultados obtidos. Agora com uma nova dimensão nos gráficos, diversos ângulos são necessários para entender o que os resultados expressam. Portanto, nas figuras dessa seção, todas apresentação quatro diferentes visões dos *plots*. O primeiro mostrando todos as dimensões e os outros mostrando apenas as visões de cada plano. Infelizmente, o gráfico de contornos ficará de fora nesta seção.

Nesta seção, serão apresentados os resultados de geração de MMGs e de amostragem nelas para capturar as distribuições de probabilidade quando tem-se duas variáveis de *log* para entrada da RDM e uma como alvo. O objetivo é, portanto, dados V_P e V_S descrever ρ. A RDM foi treinada com 15 neurônios na camada oculta e também 15 componentes. O outro valor que foi alterado é o de iterações na inicialização, agora com 15.

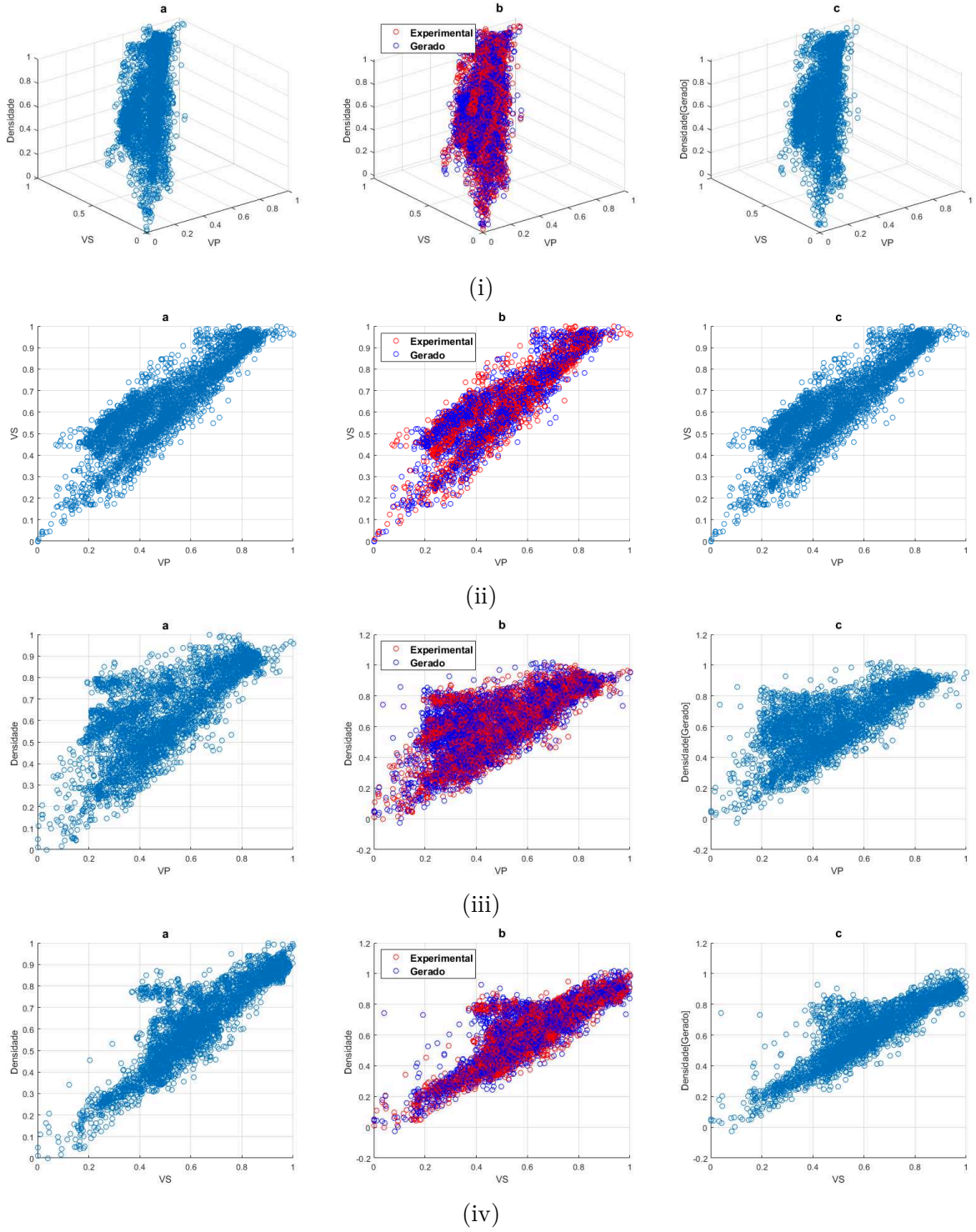

Figura 15 – *Scatter* de $V_P \times V_S \times \rho$

Fonte: Elaborada pelo Autor

Assim como em 4.2, os gráficos de *scatter* desta seção apresentam três imagens mostrando os dados originais (a), tanto originais quando gerados (b) e uma com somente os gerados (c). A Figura 15 apresenta e compara os resultados do treinamento para 2×1. É possível verificar o comportamento dos valores experimentais e gerados nas seguintes

visões:

(i). visão panorâmica;

(ii). o plano $V_P \times V_S$;

(iii). o plano $V_P \times \rho$;

(iv). o plano $V_S \times \rho$.

É possível observar que existe uma captura de comportamento nos dados amostrados tal qual havia em 4.2. No entanto, como apresenta a Figura 15, existe uma correlação menor da distribuição dos dados gerados com os empíricos, apontando para um menor captura da distribuição de probabilidade em 2×1. Essa suposição é apoiada com o coeficiente de correlação calculado, com o valor de 0.9079.

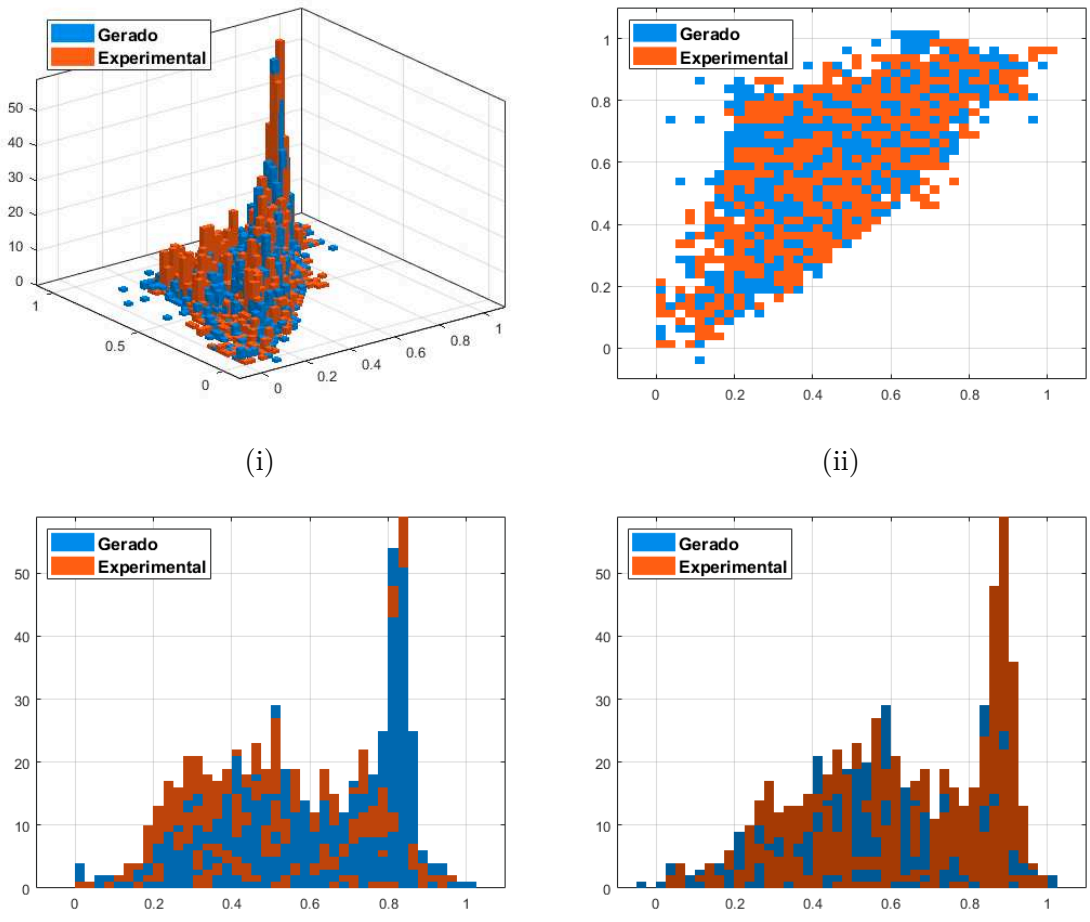

Figura 16 – Histogramas de ρ baseados em $V_P \times V_S$

Fonte: Elaborada pelo Autor

Diferente de RDM, o SSG por RNMGI não teve mudanças de parâmetros. Os resultados são os que seguem na Figura 17. A figura apresenta a mesma organização presente na Figura 15.

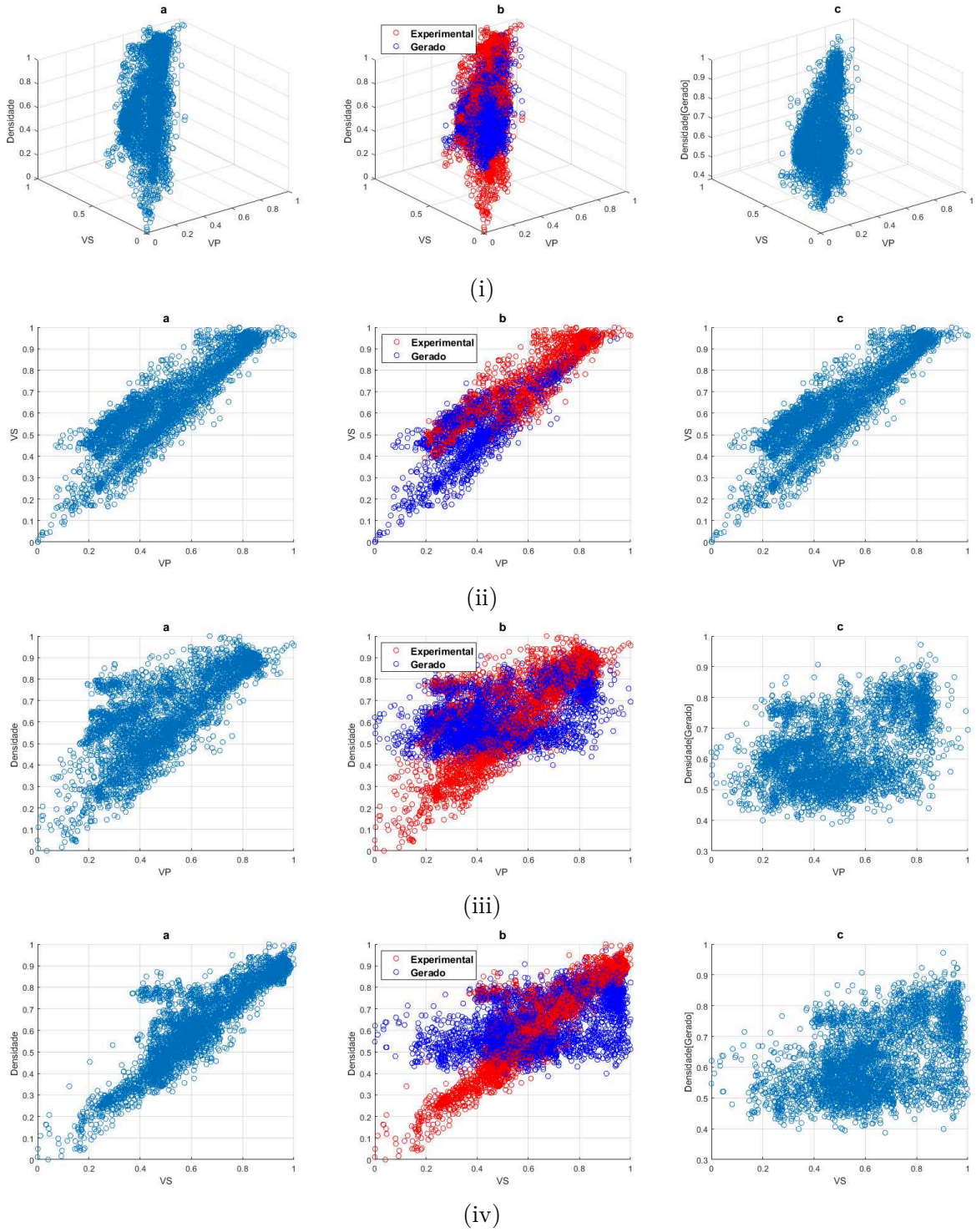

Figura 17 – *Scatter* de $V_P \times V_S \times \rho$

Fonte: Elaborada pelo Autor

Os histogramas a seguir mostram para o modelo 2×1 algo semelhante ao que se viu

na Figura 14. Novamente, o método de captura de distribuição de probabilidade por RDM parece ter sido melhor em dispor os dados de maneira semelhante ao observado nos dados experimentais. Com um coeficiente de 0.5432, os dados parecem estar longe de capturar a distribuição. O que realmente acontece é que SSG por RNMGI tem a proposta de manter as características correlação espacial e isso ele faz muito bem, contrário à geração por RDM que, no modelo apresentado neste trabalho, não mantém a correlação espacial, mas captura essa distribuição de probabilidade.

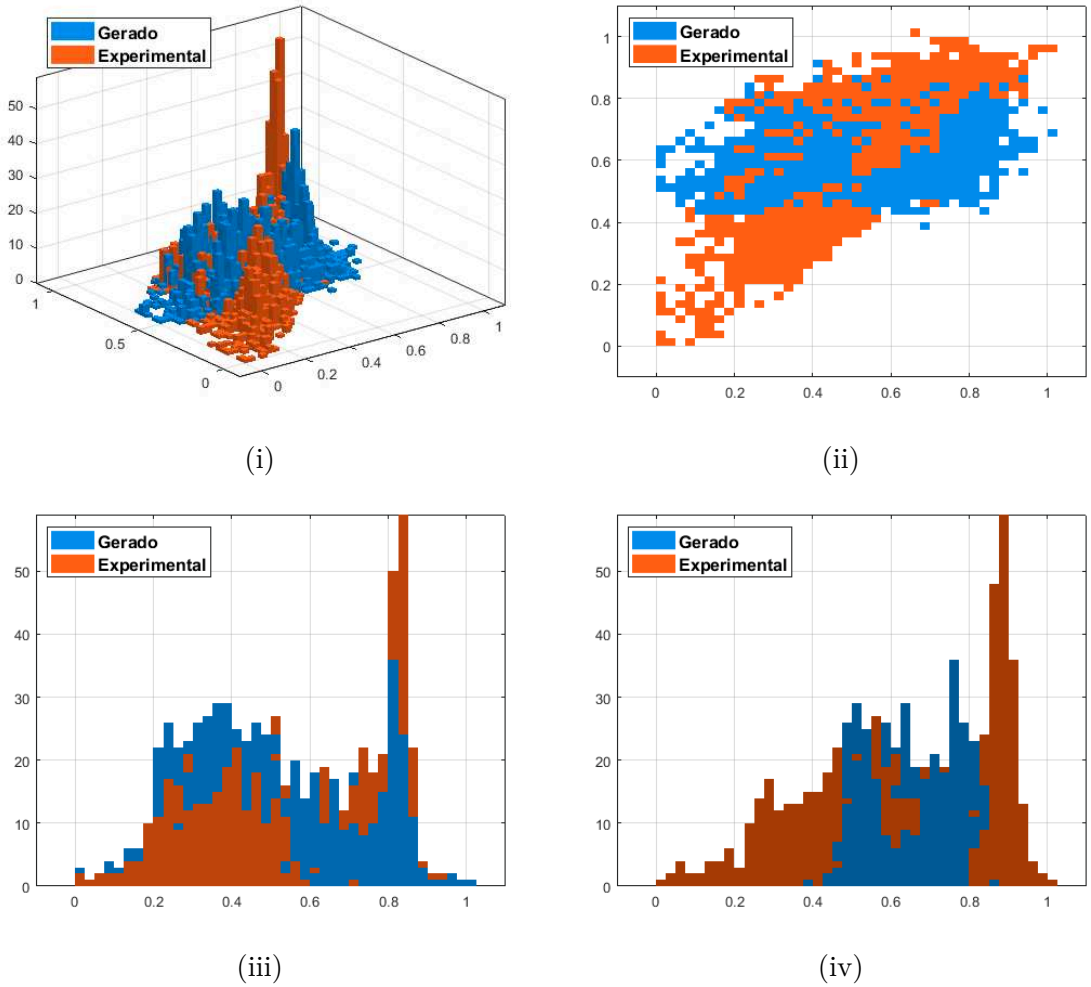

Figura 18 – Histogramas de ρ baseados em $V_P \times V_S$ por SSG com RNMGI

Fonte: Elaborada pelo Autor

5 Conclusões

Neste trabalho estudou-se redes neurais artificiais, procurando uma implementação que fosse capaz de capturar a distribuição de probabilidade de uma variável de saída quando esta é condicionada a uma variável de entrada. A RNA proposta por Bishop mostrou ser capaz de fazer isto. Para tanto, a teoria de sua proposta e funcionamento foi apresentada brevemente.

No campo da exploração e produção de petróleo, existe a necessidade de se explorar esse tipo de correlação, não apenas devido a sua existência nos dados, mas também para encontrar novas metodologias que podem simplificar diversos métodos muito complicados. Uma vez que os resultados apresentados sejam de acordo com os requirimentos, o trabalho aqui realizado pode vir a ser útil no futuro para a área.

Com os dados estudados e tratados, usou-se a biblioteca NETLAB para treinar uma RDM nela implementada. Pequenas funções implementadas em MATLAB® facilitaram os estudos e experimentos realizados. Através delas gerou-se de maneira eficiente as amostragens e os gráficos analisados. Com isso, foi possível analisar os resultados, compará-los com dados experimentais de poços reais.

De modo geral, os resultados estão dentro do esperado e mostram que as RDMs são capazes de gerar saídas que descrevam o comportamento probabilístico dos dados alvo. Os resultados mostram aderência ao comportamento apresentado nos dados obtidos e, além disso, devido ao resultado das Redes de Densidade de Misturas serem Misturas de Gaussianas, existe também uma captura de densidade e variância que pode ser explorada ao gerar amostras.

Em experimentos com uma variável de entrada e uma de saída condicionada a esta entrada, foram obtidos resultados que conseguiram descrever o comportamento probabilístico dos dados alvo. Os resultados capturaram o comportamento dos dados de forma clara, além de capturar a densidade deles. Esta captura foi analisada através da comparação entre histogramas usando coeficiente de correlação encontrando um valor de 0.9943. Neste experimento, diversas configurações foram testadas, mas a melhor captura foi encontrada quando usou-se de cinco neurônios na camada oculta, cinco componentes gaussianas para cada Modelo de Misturas de Gaussianas, cinquenta para a taxa de aprendizado, cinquenta iterações de inicialização e dez mil iterações na época.

Para averiguar a expansibilidade, foi experimentado também sobre duas variáveis de entradas e uma de saída condicionada a estas variáveis de entrada. Devido ao aumento na complexidade inerente na adição de mais dados correlacionados, existiu também uma redução na captura das características desejadas. O coeficiente de correlação sobre os

histogramas gerou um resultado de 0.9079 e o comportamento um pouco difuso nos *plots*. No entanto, os resultados são promissores devido ao já mencionado salto de complexidade.

Para esse segundo experimento, as configurações da RDM sofreram pequenas alterações. Além da alteração na entrada, foram necessários quinze neurônios na camada oculta, quinze componentes gaussianas por Modelo de Misturas Gaussianas além de quinze iterações na inicialização da componente de rede neural. Todavia, as iterações na época e taxa de aprendizado puderam ser mantidas do primeiro experimento.

Os experimentos realizados sobre o método proposto foram comparados a outro método que também usa de Redes Neurais Artificiais e Misturas de Gaussianas — Redes Neurais de Misturas Gaussianas Incrementais — mostraram resultados com menos aderência aos dados experimentais em ambas instâncias. RNMGI é um método que procura manter as características de correlação espacial entre dados simulados e experimentais, no entanto, diferente do método proposto neste trabalho que não é capaz de fazer esta mesma captura.

5.1 Trabalhos futuros

Como os resultados obtidos demonstram uma captura das características buscadas, o método ainda obtém pontos fracos que podem ser trabalhados. O principal deles, no momento, é a não captura da correlação espacial que é necessária quando se procura simular *logs* de uma região no contexto da indústria de E&P. A inclusão do *log* de profundidade nos experimentos é uma possibilidade de se capturar essa correlação, o que eleva o custo do processo, tanto em tempo de treinamento da rede quanto nos resultados que podem sofrer um grande decréscimo na captura da densidade de probabilidades devido ao aumento da complexidade.

Referências

BISHOP, C. M. Mixture density networks. *NCRG/94/004*, 1994. Disponível em: <https://publications.aston.ac.uk/id/eprint/373/1/NCRG_94_004.pdf>.

BISHOP, C. M. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, 2006. 12-32, 225-277 p. Disponível em: <http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf>. Acesso em: 5.6.2019.

HA, D. Mixture density networks with tensorflow. *blog.otoro.net*, 2015. Disponível em: <http://blog.otoro.net/2015/11/24/mixture-density-networks-with-tensorflow/>.

HEBB, D. O. *The organization of behavior: a neuropsychological theory*. [S.l.]: John Wiley & Sons, 1949.

JOSEPH, C.; FOURNIER, F.; VERNASSA, S. Pseudo-well methodology: A guiding tool for lithoseismic interpretation. In: *SEG Technical Program Expanded Abstracts 1999*. Society of Exploration Geophysicists, 1999. Disponível em: <https://doi.org/10.1190/1.1821264>.

MAVKO, G. *Basic Geophysical Concepts*. Stanford Rock Physics Laboratory, 2005. Disponível em: <https://pangea.stanford.edu/courses/gp262/Notes/2.Basic%20Concepts.pdf>. Acesso em: 26.10.2019.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 1943.

MILSOM, J. *Field geophysics*. 3. ed. [S.l.]: Wiley, 2003. 179, 180 p.

NETLAB: Algorithms for Pattern Recognition. 2019. Disponível em: <https://www2.aston.ac.uk/eas/research/groups/ncrg/resources/netlab>.

NG, A. *Deep Learning Specialization*. Coursera, 2018. Disponível em: <https://www.deeplearning.ai/>. Acesso em: 16.04.2019.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, p. 65–386, 1958.

SCHLUMBERGER. *well log*. 2019. Disponível em: <https://www.glossary.oilfield.slb.com/en/Terms/w/well_log.aspx>.

Soares, S. A. F.; Neto, G. S.; Roisenberg, M. Improving the incremental gaussian mixture neural network model for spatial interpolation and geostatistical simulation. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2016. p. 2507–2514.

WERBOS, P. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University, 1975. Disponível em: <https://books.google.com.br/books?id=z81XmgEACAAJ>.

Apêndices

APÊNDICE A – Código Fonte

Código A.1 – main.m

```
% Read .las file
Well = read_las2_file('Well_1.las');   % Can't provide :(

% Identificar coluna do log
VP_index = find(ismember(Well.curve_info(:, 1), 'VP'));
VS_index = find(ismember(Well.curve_info(:, 1), 'VS'));
Rho_index = find(ismember(Well.curve_info(:, 1), 'RHOB'));

% Find point to be removed for the non-existent in at least one of the logs
cut = (~isnan(Well.curves(:,VP_index))&(Well.curves(:,VP_index)~=-999)) ...
    &(~isnan(Well.curves(:,VS_index))&(Well.curves(:,VS_index)~=-999)) ...
    &(~isnan(Well.curves(:,Rho_index))&(Well.curves(:,Rho_index)~=-999));

% Make it so only indexes to be removed have value 1
cut = ~logical(sum(isnan(Well.curves(:,[VP_index VS_index Rho_index])),2));

% Deploy cut
VPo = Well.curves(cut, VP_index);
VSo = Well.curves(cut, VS_index);
Rhoo = Well.curves(cut, Rho_index);

% Nomalize data
VP = (VP - min(VP))/(max(VP)-min(VP));
VS = (VS - min(VS))/(max(VS)-min(VS));
Rho = (Rho - min(Rho))/(max(Rho)-min(Rho));

% Simplify
X = VP;
Y = VS;
Z = Rho;

% Configure and run MDN training for 1x1 model
mix = exe_mdn(X, Z, 1, 5, 5, 50, 1, 55, 10000);
% Plot 1x1 model
[a b] = show_scattering(mix, X, Z, 'VP ', 'Densidade ');

% Configure and run MDN training for 2x1 model
%mix = exe_mdn([VP VS], Rho, 2, 10, 10, 50, 1, 15, 10000);
% Plot 2x1 model
%[a b] = show_scattering3(mix, X, Y, Z, 'VP ', 'VS', 'Densidade ');
```

Código A.2 – exe_mdn.m

```
function mixes = exe_mdn(X, T, nin, nhidden, ncentres, alpha, dim_target, init_it
    , it)
% Adapted from NETLAB: Sets up MDN for training and then do it.

    % Create and initialize network weight vector.
    net = mdn(nin, nhidden, ncentres, dim_target, '0');

    % Set up options
    init_options = zeros(1, 18);
    init_options(1) = 0;         % Suppress all messages
    init_options(14) = init_it; % iterations of K means in gmminit

    % Initializate MDN
    net = mdninit(net, alpha, T, init_options);

    % Set up vector of options for the optimiser.
    options = foptions;
    options(1) = 0;              % This provides display of error values
    options(14) = it;            % Number of training cycles

    % Train using scaled conjugate gradients.
    [net, options] = netopt(net, options, X, T, 'scg');

    % Extract GMM from MDN
    mixes = mdn2gmm(mdnfwd(net, X));
```

Código A.3 – show_scattering.m

```
function [a b] = show_scattering(mix, X, Y, XN, YN)
% All kinds of scattering. Also histograms.

    % Samples from received mixes
    spls = sam(mix, 1);

    % Creates a Figure and sets up for first scatter
    figure; subplot(1, 3, 1);

    % Scatter for original data
    scatter(X, Y);
    xlabel(XN)
    ylabel(YN)
    title('Experimental')

    % Sets up for second scatter
    subplot(1, 3, 2);

    % Scatter for original overlayed with sampled
    scatter(X, Y, 'r');
    hold on;
    scatter(X, spls, 'b');
    xlabel(XN)
    ylabel(YN)
    l = legend('Experimental', 'Gerado', 'Location', 'northwest');
    l.FontSize = 12;
    l.FontWeight = 'bold';
    title('Ambos')
    hold off;

    % Sets up for third scatter
    subplot(1, 3, 3);

    % Scatter for sampled
    scatter(X, spls);
    xlabel(XN)
    ylabel(strcat(YN, '[Gerado]'))
    title('Gerado')

    % Creates a figure for histogram
    figure; a = histogram(spls, [-0.2:0.05:1.2]); hold on;
    b = histogram(Y, [-0.2:0.05:1.2]);
    l = legend('Gerado', 'Experimental', 'Location', 'northwest');
    l.FontSize = 12;
    l.FontWeight = 'bold';
```

Código A.4 – show_scattering3.m

```
function [a b] = show_scattering3(mix, X, Y, Z, XN, YN, ZN)
% All kinds of scattering. Also histograms. Now in 3D.

    %samples from received mixes. Now in 3D.
    spls = sam(mix, 1);

    % Creates a Figure and sets up for first scatter
    figure; subplot(1, 3, 1);

    % Scatter for original data. Now in 3D.
    scatter3(X, Y, Z);
    xlabel(XN)
    ylabel(YN)
    zlabel(ZN)
    title('a', 'FontSize', 14, 'FontWeight', 'bold')

    % Sets up for second scatter
    subplot(1, 3, 2);

    % Scatter for original overlayed with sampled. Now in 3D.
    scatter3(X, Y, Z, 'r');
    hold on;
    scatter3(X, Y, spls, 'b');
    xlabel(XN)
    ylabel(YN)
    zlabel(ZN)
    l = legend('Experimental', 'Gerado', 'Location', 'northwest');
    l.FontSize = 12;
    l.FontWeight = 'bold';
    title('b', 'FontSize', 14, 'FontWeight', 'bold')
    hold off;

    % Sets up for third scatter
    subplot(1, 3, 3);

    % Scatter for sampled. Now in 3D.
    scatter3(X, Y, spls);
    xlabel(XN)
    zlabel(strcat(ZN, '[Gerado]'))
    ylabel(YN)
    title('c', 'FontSize', 14, 'FontWeight', 'bold')

    % Creates a figure for histogram. Now in 3D.
    figure; a = histogram2(X, spls, 'BinWidth', [0.025 0.025], 'EdgeAlpha',...
        0.75, 'XBinLimits', [-0.1, 1.1], 'YBinLimits', [-0.1, 1.1]); hold on;
    b = histogram2(X, Z, 'BinWidth', [0.025 0.025], 'EdgeAlpha', 0.75,...
        'XBinLimits', [-0.1, 1.1], 'YBinLimits', [-0.1, 1.1]);
    l = legend('Gerado', 'Experimental', 'Location', 'northwest');
    l.FontSize = 12;
    l.FontWeight = 'bold';
    hold off;
```

Código A.5 – sam.m

```
function am = sam(mix, ppx)
% Some good sampling from mixes

mixlen = size(mix, 2);
am = zeros(mixlen, ppx);
for j = 1:mixlen
    for k = 1:1:ppx
        i = randsrc(1, 1, [1:1:mix(j).ncentres; mix(j).priors]);
        variance = reshape([mix.covars], mix(j).ncentres, mixlen)';
        am(j, k) = mvnrnd(mix(j).centres(i), variance(j, i), 1);
    end
end
```

APÊNDICE B – Artigo

Machine Learning na Área do Petróleo: Implementação de Redes Neurais para Aprendizado de Distribuições de Probabilidade

Lucas Ribeiro Neis[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

Abstract. *The exploration well method is used to gather several data for the Exploration and Production industry. However, well drilling is not cheap and it's commom to need for more than a single well. As such, simulation methods have been recieving attention as a possible approach to get around the mutiple well drilling problem. Although using Artificial Neural Networks seems promising for this field, the traditional Neural Networks are usually functional, making them unsuitable for this purpose. In this work a study was followed by an implementation of a Mixture Densisity Model in the E&P industry context for log generation simulation once it captures the probability density for an output variable. The results showed that the method is able to mantain grip on the original data behavior while also creating a proper density distribution even after increase in complexity by expansion on the number of input logs.*

Resumo. *O método de poço exploratório é usado para capturar diversos dados para a indústria de exploração e produção. Todavia, a perfuração de poços é cara e é comum a necessidade de diversos poços. Assim, métodos para simular esses dados foram criados para diminuir este número. Os modelos de redes neurais artificiais mais tradicionais implementam modelos funcionais, tornando-as inadequadas para este domínio. Neste trabalho, foi realizado um estudo e uma implementação de uma rede neural artificial que usa de misturas de gaussianas para modelar a distribuição de probabilidade. Os resultados obtidos demonstraram que o modelo aderiu ao comportamento e a distribuição de densidade dos dados originais, mesmo com o aumento da complexidade ao adicionar mais de um tipo de log na entrada.*

1. Introdução

Em problemas de comportamento não funcional, Redes Neurais Artificiais convencionais capturam de forma limitada o comportamento dos dados devido a esta metodologia ser uma implementação funcional. Com isso, problemas tais como os de inversão (*e.g. e.g.* encontrar o valor em x que gera $y = 4$ na função $y = x^2$) não podem ser bem descritos por essas Redes Neurais convencionais, uma vez que elas tendem a representar a média de um grupo de dados ao invés de capturar todas as possibilidades [Bishop 2006].

Com isso, o modelo de *Mixture Density Network* foi criado. Ele procura contornar essa deficiência de Redes Neurais convencionais usando Misturas de Gaussianas para capturar a distribuição de probabilidade. Sendo assim, *Mixture Density Networks* podem

ser usadas para capturar a variância, a incerteza e a possibilidades de múltiplos pontos para um valor do domínio [Bishop 1994].

A simulação de poços dá-se através do uso de dados experimentais coletados em poços de exploração. Tais dados são denominados *logs*, que são medidas em função do tempo e/ou profundidade de uma ou mais características físicas de um poço ou ao redor dele [Schlumberger 2019]. *Logs* são úteis para simulação devido suas intercorrelações que podem ser exploradas para que os dados obtidos sejam mais precisos.

Uma vez que *Mixture Density Networks* se dispõe a contornar o problema de multiplas respostas para uma entrada, o seu uso na área de petróleo pode obter resultados interessantes visto a necessidade de se capturar a correlação estatística dos dados. Logo, procura-se usar em uma implementação uma *Mixture Density Network* e verificar se esta seria capaz de gerar Misturas Gaussianas que capturem o comportamento dos dados de modo a usá-las para extrair valores que se assemelham aos experimentais usando da intercorrelação dos *logs* para treinar a rede.

2. *Logs* e Tratamento de Dados

Quando se trata de análise de dados sísmicos para exploração de reservas tais como petróleo, o processo de aquisição desses dados pode assumir diversas formas. Visto a variedade de métodos que existem, diferentes maneiras de interpretar esses dados foram conceituados. Um desses métodos é a perfuração de poços exploratórios. Esses poços são perfurados para se estudar uma região a fim de procurar reservas a serem exploradas, quais o tamanho dessas reservas e, portanto, se vale o custo de se extrair os recursos ali encontrados. Dados sísmicos tornaram-se muito importantes para se desenhar características de dadas reservas em espaço inter-poços [Joseph et al. 1999].

A fim de fazer um detalhado registro das formações geológicas encontradas em uma perfuração, *logs* de poços são usados. Os *logs* podem ser adquiridos de algumas diferentes formas tal como amostras visuais trazidas a superfície ou medidas feitas com instrumentos introduzidos no poço [Schlumberger 2019].

Logs são amplamente usados e podem ser dados de densidade ou das velocidades sísmicas. Além disso, os dados não são necessariamente da perfuração inteira e podem descrever apenas um trecho de interesse. Alguns *logs* têm relação entre si o que torna interessante para pseudo-poços no caso da simulação deles.

Os dados de *logs* usados neste trabalho são reais. Os *logs* são salvos num arquivo que engloba todos os dados de um poço em formato **LAS**. O arquivo tem treze *logs* e suas devidas unidades de medida. Os *logs* são arranjados em colunas onde os valores medidos são dispostos. Algumas informações extras como qual o Δx de profundidade para medir os pontos. Além disso, pontos os quais um *log* não foi devidamente medido recebem o valor -999. Os seguintes *logs* serão usados nos experimentos:

- Densidade (ρ): O *log* de densidade representa a medição das mudanças no campo magnético da Terra gerada pela diferença da densidade de rochas [**?**];
- Velocidade da Onda P (V_P): A velocidade cujo a onda de pressão[1] viajou através de rochas

[1]Onda-P: Quando uma onda viaja em um meio (*e.g.* ar), as moléculas oscilam para trás e então para frente na direção do transporte de energia. Ou seja, conforme ela viaja causando compressões e rarefações

- Velocidade da Onda S (V_S): A velocidade cujo a onda de pressão[2] viajou através de rochas

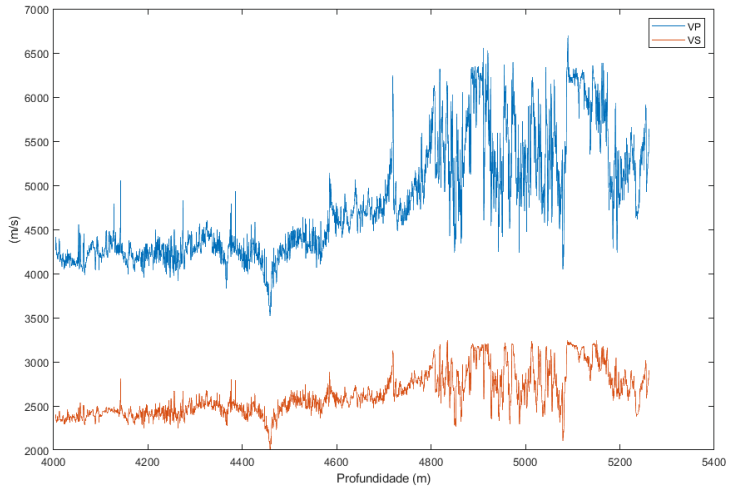

Figura 1. V_P e V_S ao longo da profundidade

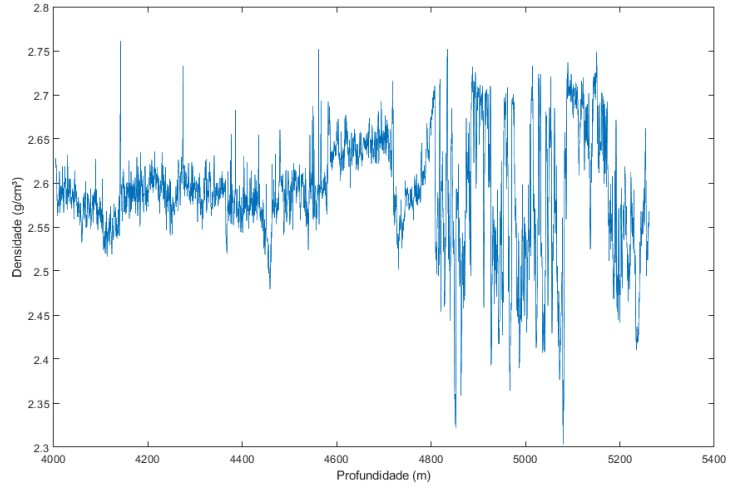

Figura 2. Densidade ao longo da profundidade

A Figura 1 apresenta o comportamento da velocidade das ondas P e S conforme essas são refletidas pelas rochas. A Figura mostra como as curvas são parecidas em termos de amplitudes, apesar da diferença na velocidade propriamente dita, mostrando a

ao empurrar o meio e é de onde vem seu nome — *pressure wave*. É importante destacar que, quando em meio sólido, a onda-P tem a maior velocidade entre os movimentos ondulatórios e é comumente chamada de onda primária [Milsom 2003].

[2]Onda-S: Quando uma onda viaja em meios sólidos, partículas vibrando em ângulo reto a direção do fluxo de energia causam o que é chamado de onda-S — *shear wave*. Por ter velocidade baixa, pode ser chamada de onda secundária, comumente, a Onda-S tem velocidade aproximada da metade da Onda-P em várias rochas consolidadas [Milsom 2003].

correlação presente. Além de V_P e V_S, a densidade também é importante de ser mencionada. A Figura 2 traz ρ em mais detalhes. A diferença de escala entre a densidade e as velocidades das ondas é importante de se considerar. Por fim, a Figura 3 apresenta um *crossplot* par-a-par entre os três.

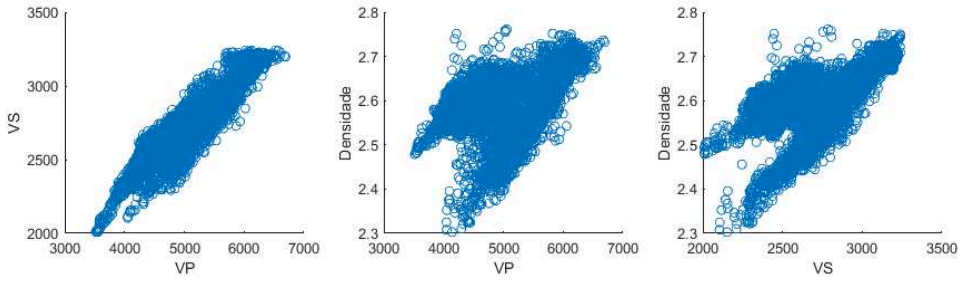

Figura 3. *Crossplot* de V_PxV_S, V_Pxρ e V_Sxρ

Por serem dados experimentais, os *logs* precisam receber um pouco de atenção antes de serem usados no treinamento da rede. Um primeiro problema encontrado foi que os *logs* de V_P, V_S e ρ apresentavam pontos onde dados não haviam sido coletados. Com isso, todos valores que em uma das três variáveis recebiam um valor **NaN** foram removidos de todas elas. Além disso, a densidade não tem a mesma escala que as velocidades e como o treinamento de Redes Neurais Artificiais funciona levando em consideração os dados de entrada para formular pesos para variáveis internamente, torna-se problemático colocar dados com tamanha diferença. Conforme o treinamento ocorre, vai ser necessário que a rede procure calcular pesos para compensar essa diferença [Ng 2018].

Com isso, aplicou-se um *min-max nomalization* para remover a discrepância, como traz a equação 1. A Figura 4 mostra o mesmo *crossplot* da Figura 3, mas agora com todos os dados no intervalo $[0, 1]$.

$$NormData[i] = \frac{Data[i] - min(Data)}{max(Data) - min(Data)} \tag{1}$$

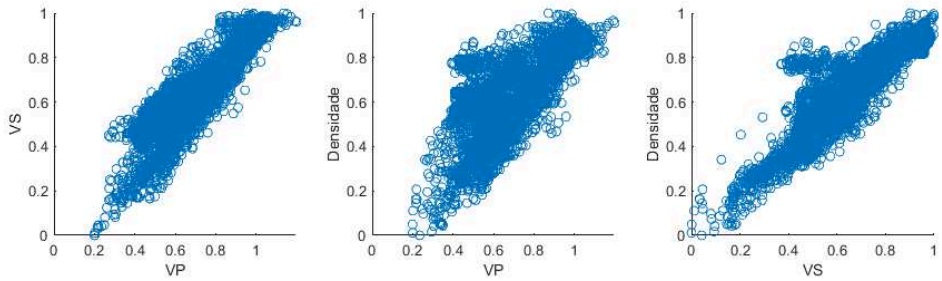

Figura 4. *Crossplot* de V_PxV_S, V_Pxρ e V_Sxρ normalizados

3. Metodologia

Para testar a escalabilidade do método, dois modelos foram criados. O primeiro modelo procura testar a viabilidade dos resultados enquanto o segundo busca verificar se a qualidade dos resultados tem perda muito significativa quando se aumenta a complexidade do

escopo. Os modelos são dados, respectivamente, pela descrição da distribuição probabilidade de um *log* quando dado apenas um outro e a captura quando são dados dois *logs* para descrever um terceiro.

Neste estudo não foi levado em consideração a profundidade, fazendo com que os resultados não mantenham a correlação espacial. No entanto, adicionar o *log* de profundidade em experimentos é viável e, em trabalhos futuros, verificar qual a qualidade dos resultados obtidos para este cenário é uma possibilidade. Contudo, como demonstrado a seguir, a adição de novos *logs* durante o treinamento tem custo na qualidade geral dos resultados.

Sendo assim, os resultados são submetidos a duas análises para atestar a sua qualidade. A primeira é verifica se existe uma captura da distribuição dos dados — não da densidade, mas do comportamento. Para isso, os dados são normalizados e em um *crossplots* são sobrepostos aos dados experimentais — também normalizados — para comparação.

Já a segunda procura verificar se a densidade dos dados foi capturada adequadamente. Para isso são usados histogramas tanto dos resultados quanto dos dados experimentais. Estes histogramas são sobrepostos para uma analise visual e um coeficiente de correlação é calculado para melhor estimar o quão próximos são os dois histogramas.

De modo a facilitar a leitura, o primeiro modelo implementado será nominado de 'modelo 1×1' e o segundo como 'modelo 2×1'.

4. Resultados

Os resultados estão separados por modelo. Cada modelo mostrará primeiro a captura de comportamento e depois a captura de densidade de probabilidade.

4.1. Modelo 1×1

Este modelo procura capturar a probabilidade condicional $P(\mathbf{y}|\mathbf{x})$ onde $\mathbf{y}$ é o conjunto de dados do *log* alvo e $\mathbf{x}$ é o conjunto de dados do *log* de entrada. Sendo assim, nos resultados a seguir, apresenta-se os resultados obtidos para capturar $P(\rho|\mathbf{V_P})$.

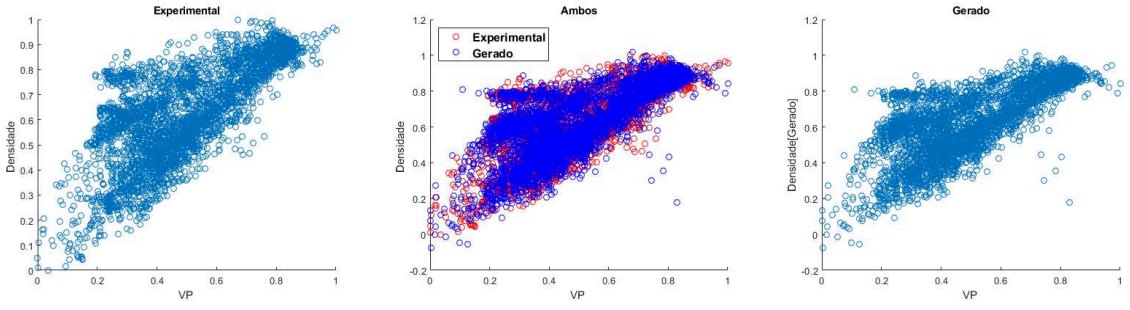

Figura 5. As três imagens comparam os dados experimentais com os gerados. À direita mostra-se os dados experimentais, à esquerda os dados gerados e no centro é sobreposto os dados gerados aos dados experimentais.

A Figura 5 mostra a captura de comportamento obtida. Na imagem central, é possível notar que o comportamento obtido é próximo do presente nos dados experimentais.

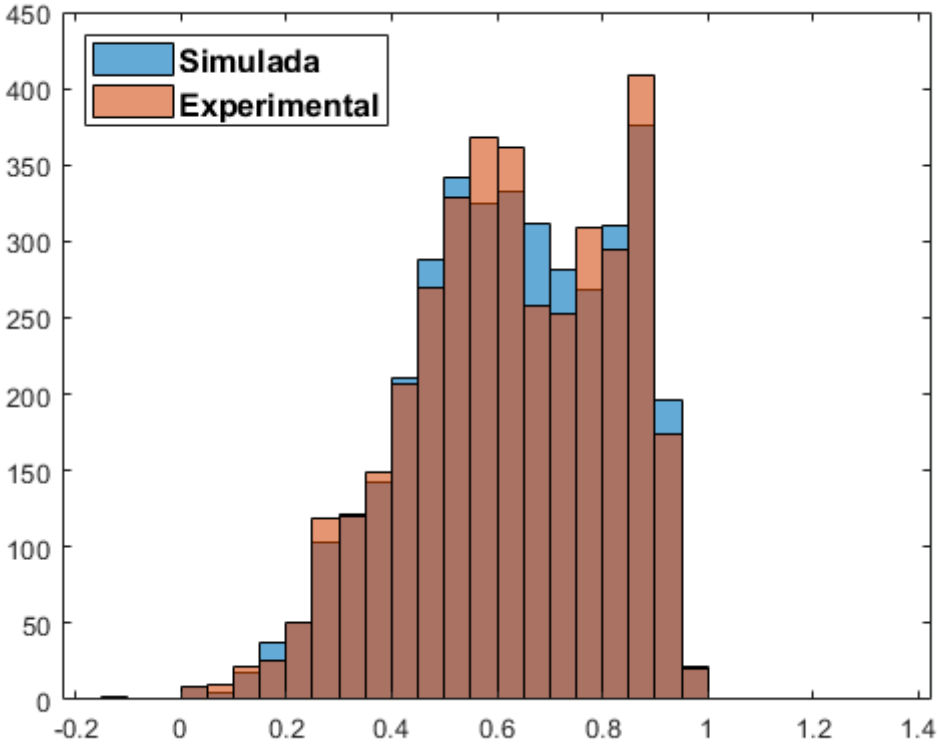

Figura 6. Histograma comparativo de ρ dos dados simulados com os dados experimentais.

A Figura 6 demonstra a captura da densidade de probabilidades. Pode-se observar que a captura de densidade foi devidamente capturada obtendo-se o coeficiente de correlação de 0.9943 entre ambos histogramas.

4.2. Modelo 2×1

Neste modelo, procura-se capturar a probabilidade condicional $P(\mathbf{z}|\mathbf{x},\mathbf{y})$, onde $\mathbf{z}$ é o conjunto de dados do *log* alvo e $\mathbf{x}$ e $\mathbf{y}$ são os conjuntos de dados de dois outros *logs*. Assim, nos resultados a seguir serão apresentados a captura de $P(\rho|\mathbf{V_P},\mathbf{V_S})$.

A Figura 7 traz quatro conjuntos de imagens que mostram em *a* os dados experimentais, em *c* os dados obtidos pelo método e em *b* a sobreposição dos dois anteriores. O conjunto *i* mostra os dados a partir de uma projeção ortogonal, enquanto as demais mostram cada um dos planos:

ii O plano $V_P \times V_S$;
iii O plano $V_P \times \rho$;
iv O plano $V_S \times \rho$.

Devido ao aumento da complexidade do problema ao adicionar um novo *log* para o treinamento da rede, existe também menos aderência nos resultados. Sendo assim, observa-se uma pior captura em comparação ao modelo 1×1. No entanto, a captura ainda é visivelmente próximo do observados nos dados experimentais, mostrando que

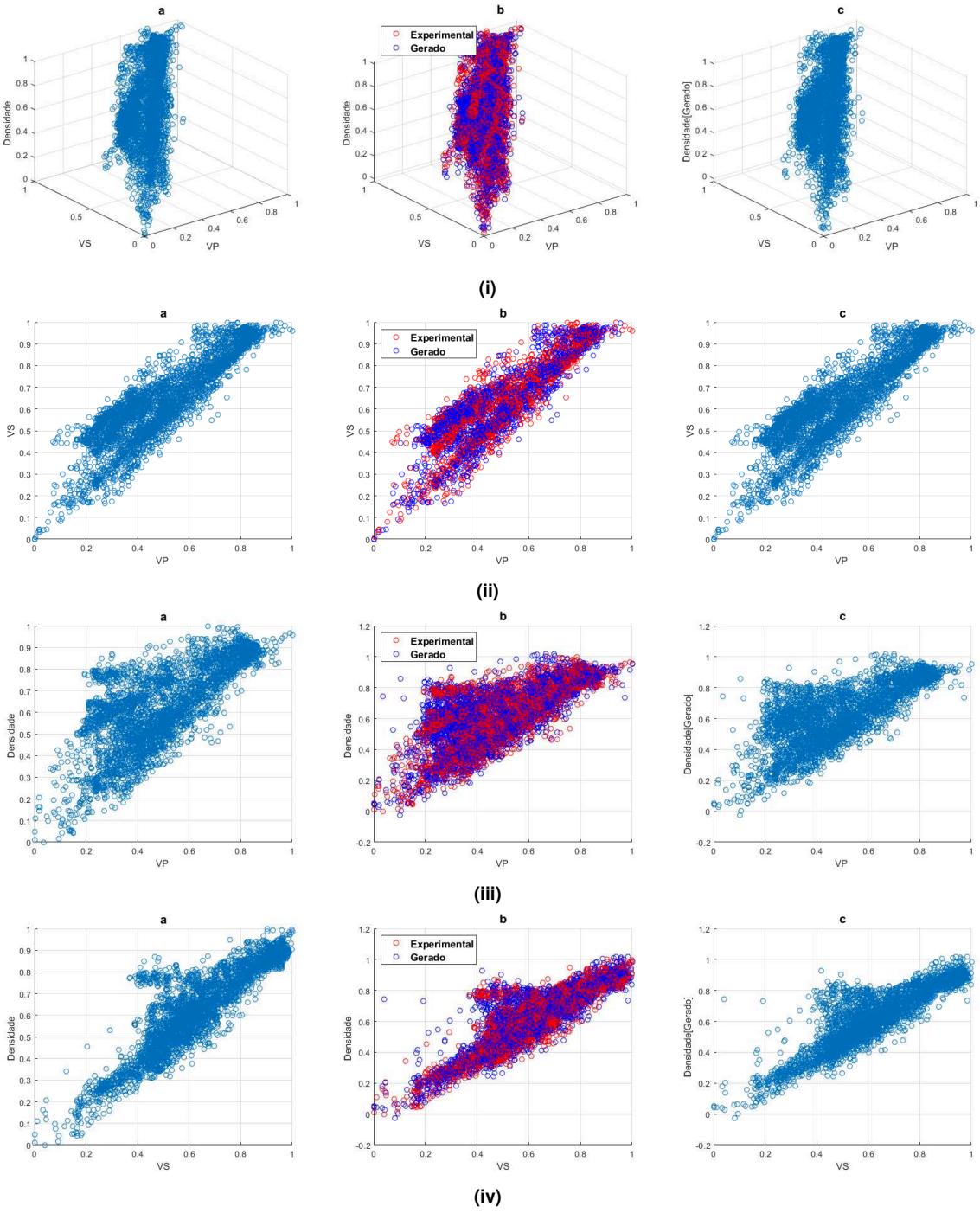

Figura 7. *Scatter* de $V_P \times V_S \times \rho$

método é passível de expansão em troca de uma certa qualidade dos resultados já que existe um aumento considerável no tamanho do problema.

Uma vez que também é necessário observar a captura das densidades de probabilidades, é importante notar que neste caso o histograma será tridimensional devido às duas entradas que geral uma probabilidade condicional.

Pode-se, por fim, observar que assim como no comportamento dos dados, a densi-

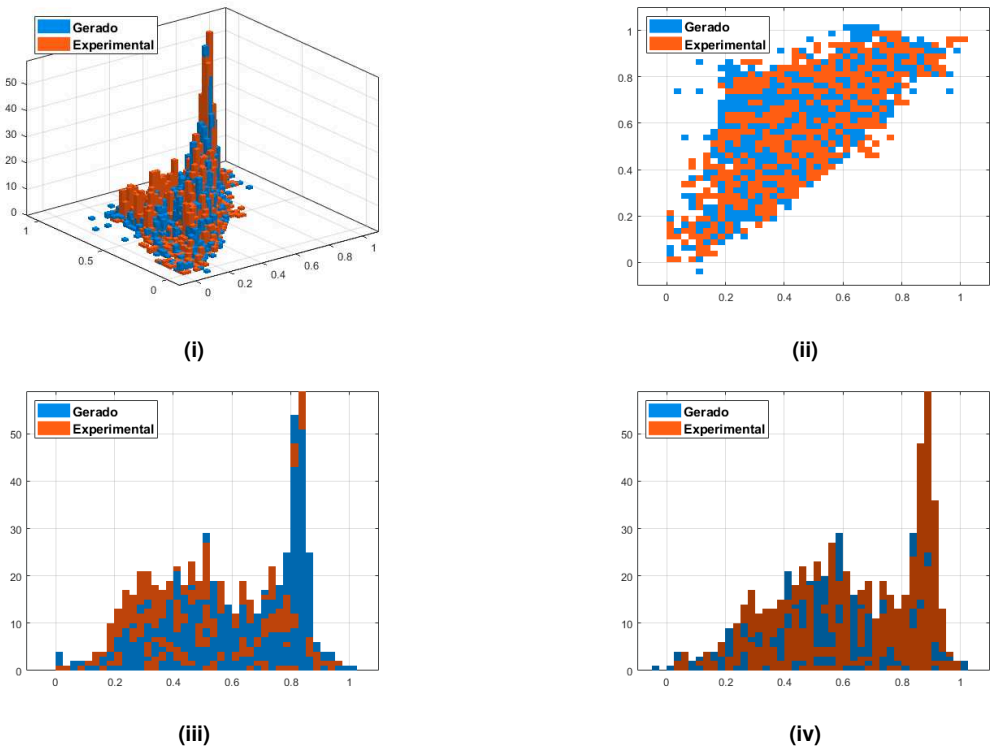

Figura 8. Histogramas de ρ baseados em $V_P \times V_S$

dade também perde um pouco sua qualidade devido ao aumento da complexidade. Aqui, o coeficiente de correlação é de 0.9079. Uma perda de 0.09 em relação ao anterior em troca de mais relação com os dados experimentais.

5. Conclusão

No campo da exploração e produção de petróleo, existe a necessidade de explorar a correlação entre *logs*, não apenas devido a sua existência nos dados, mas também para encontrar novas metodologias que possam simplificar diversos métodos muito complicados. Uma vez que os resultados apresentados sejam de acordo com os requirimentos, o trabalho aqui realizado pode vir a ser útil no futuro para a área.

De modo geral, os resultados estão dentro do esperado e mostram que as Redes de Densidades de Misturas são capazes de gerar saídas que descrevam o comportamento probabilístico dos dados alvo. Os resultados mostram aderência ao comportamento apresentado nos dados obtidos e, além disso, devido ao resultado das Redes de Densidade de Misturas serem Misturas de Gaussianas, existe também uma captura de densidade e variância que pode ser explorada ao gerar amostras.

Em experimentos com uma variável de entrada e uma de saída condicionada a esta entrada, foram obtidos resultados que conseguiram descrever o comportamento probabilístico dos dados alvo. Os resultados capturaram o comportamento dos dados de forma clara, além de capturar a densidade deles. Esta captura foi analisada através da comparação entre histogramas usando coeficiente de correlação encontrando um valor de 0.9943. Neste experimento, diversas configurações foram testadas, mas a melhor captura foi encontrada quando usou-se de cinco neurônios na camada oculta, cinco componen-

tes gaussianas para cada Modelo de Misturas de Gaussianas, cinquenta para a taxa de aprendizado, cinquenta iterações de inicialização e dez mil iterações na época.

Para averiguar a expansibilidade, foi experimentado também sobre duas variáveis de entradas e uma de saída condicionada a estas variáveis de entrada. Devido ao aumento na complexidade inerente na adição de mais dados correlacionados, existiu também uma redução na captura das características desejadas. O coeficiente de correlação sobre os histogramas gerou um resultado de 0.9079 e o comportamento um pouco difuso nos *plots*. No entanto, os resultados são promissores devido ao já mencionado salto de complexidade.

Para esse segundo experimento, as configurações da Redes de Densidades de Misturas sofreram pequenas alterações. Além da alteração na entrada, foram necessários quinze neurônios na camada oculta, quinze componentes gaussianas por Modelo de Misturas Gaussianas além de quinze iterações na inicialização da componente de rede neural. Todavia, as iterações na época e taxa de aprendizado puderam ser mantidas do primeiro experimento.

6. Trabalhos Futuros

Como os resultados obtidos demonstram uma captura das características buscadas, o método ainda obtém pontos fracos que podem ser trabalhados. O principal deles, no momento, é a não captura da correlação espacial que é necessária quando se procura simular *logs* de uma região no contexto da indústria de E&P. A inclusão do *log* de profundidade nos experimentos é uma possibilidade de se capturar essa correlação, o que eleva o custo do processo, tanto em tempo de treinamento da rede quanto nos resultados que podem sofrer um grande decréscimo na captura da densidade de probabilidades devido ao aumento da complexidade.

Referências

Bishop, C. M. (1994). Mixture density networks. *NCRG/94/004*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer Science+Business Media.

Joseph, C., Fournier, F., and Vernassa, S. (1999). Pseudo-well methodology: A guiding tool for lithoseismic interpretation. In *SEG Technical Program Expanded Abstracts 1999*. Society of Exploration Geophysicists.

Milsom, J. (2003). *Field geophysics*. Wiley, 3 edition.

Ng, A. (2018). Deep learning specialization. Available at: `https://www.deeplearning.ai/`.

Schlumberger (2019). well log. Available at: `https://www.glossary.oilfield.slb.com/en/Terms/w/well_log.aspx`.