ISSN 1677-9266
Dezembro/2019

BOLETIM DE PESQUISA E DESENVOLVIMENTO

45

Uso de redes neurais multicamadas para classificação de perfis de solos

Embrapa

ISSN 1677-9266
Dezembro/2019

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura, Pecuária e Abastecimento

BOLETIM DE PESQUISA E DESENVOLVIMENTO 45

Uso de redes neurais multicamadas para classificação de perfis de solos

Kleber Xavier Sampaio de Souza
João Camargo Neto

Embrapa Informática Agropecuária
Campinas, SP
2019

Exemplares desta publicação podem ser adquiridos na:

Embrapa Informática Agropecuária
Av. André Tosello, nº 209 Campus da Unicamp,
Barão Geraldo - Campinas - SP
CEP: 13083-886
Fone: (19) 3211-5700

www.embrapa.br
www.embrapa.br/fale-conosco/sac

Supervisão editorial
Kleber X. Sampaio de Souza

Revisão de texto
Adriana Farah Gonzalez

Normalização bibliográfica
Carla Cristiane Osawa

Projeto gráfico da coleção
Carlos Eduardo Felice Barbeiro

Editoração eletrônica
Felipe Prado Jaconi sob supervisão de Magda Cruciol

Foto da capa
Pexels

1ª edição
Versão digital (2019)

Dados Internacionais de Catalogação na Publicação (CIP)
Embrapa Informática Agropecuária

Uso de redes neurais multicamadas para classificação de perfis de solos / Kleber Xavier Sampaio de Souza, João Camargo Neto. -- Campinas : Embrapa Informática Agropecuária, 2019.

PDF (23 p.) : il. color. - (Boletim de pesquisa e desenvolvimento / Embrapa Informática Agropecuária, ISSN 1677-9266 ; 45).

1. Classificação de solos. 2. Redes neurais. 3. Aprendizado profundo. 4. Deep learning. I. Souza, Kleber Xavier Sampaio de Souza. II. Camargo Neto, João. III. Título. IV. Embrapa Informática Agropecuária. V. Série.

CDD (21. ed.) 006.32

Carla Cristiane Osawa (CRB-8/10421)

Sumário

Uso de redes neurais multicamadas para classificação de perfis de solos

Kleber Xavier Sampaio de Souza[1]
João Camargo Neto[2]

Resumo – O processo de classificação de solos executado por especialistas é uma tarefa laboriosa, que envolve várias etapas de coleta e a aplicação de regras de classificação de acordo com o Manual do Sistema Brasileiro de Classificação de Solos (SiBCS). Este trabalho relata a aplicação de redes neurais do tipo perceptron multicamadas na classificação de solos, nos níveis categóricos 1 a 4. Os dados de perfis de solo utilizados vieram de uma base de dados do Instituto Brasileiro de Geografia e Estatística (IBGE). A otimização da função de custo de entropia cruzada, usada no treinamento da rede neural, foi realizada por meio do algoritmo de *stochastic gradient descent.* As redes apresentaram resultados de acurácia que variam de 63,38 a 27,79, tendo o valor maior sido obtido para o primeiro nível e o menor para o quarto nível de classificação. Os resultados mostraram o alto potencial de uso do perceptron multicamadas para a classificação de perfis de solo, resultado que pode ser ainda melhorado caso se disponha de um conjunto maior e mais balanceado de perfis de solos previamente classificados.

Termos para indexação: aprendizado profundo, perceptron multicamadas, amostras de solo, classificação.

1 Engenheiro eletricista, doutor em Telemática, pesquisador da Embrapa Informática Agropecuária, Campinas, SP.

2 Engenheiro eletricista, Ph.D. em Processamento de imagens, analista da Embrapa Informática Agropecuária, Campinas, SP.

Using multilayer neural networks for soil profile classification

Abstract – The soil classification process executed by experts is a laborious task, which encompasses several steps of collecting and applying classification rules in accordance to the Brazilian Soil Classification System (SiBCS) Handbook. This work reports the application of neural networks of the type multilayer perceptron in soil classification, from categorical levels 1 to 4. The soil profiles data set used came from an IBGE database. The optimization of the cross entropy cost function used in the neural network training was performed using stochastic gradient descent algorithm. The networks presented accuracy results ranging from 63,38 to 27,79, with the highest value obtained for the first level and the lowest value for the fourth categorical classification level. The results show the high potential of using multilayer perceptron for soil profiles classification, a result that can be further improved by having a larger and more balanced set of previously classified soil profiles.

Index terms: deep learning, multilayer perceptron, soil sampling, classification.

Introdução

O processo de classificação de solos analisa as informações coletadas em suas unidades básicas, chamadas perfis (Santos et al., 2006). Trata-se de um processo laborioso, pois envolve a coleta de material no campo, sua análise química em laboratórios e a avaliação de dados morfológicos, mineralógicos e físicos, tais como textura, cor úmida e seca, presença de cascalho e consistência, entre outros. A análise é realizada para cada uma das camadas de solo observadas, às quais se dá o nome de horizontes.

O conjunto das informações coletadas nos perfis é então confrontado com regras de classificação, de acordo com a sequência descrita no Sistema Brasileiro de Classificação de Solos (SiBCS) (Santos et al., 2006). Este sistema prevê a classificação em até seis níveis categóricos, mas os dois últimos níveis ainda são objeto de discussão.

As regras de classificação descritas no SiBCS estão contidas apenas no manual em papel que o descreve. Não há auxílio computacional para quem o deseja utilizar e esta foi a principal motivação para a proposição do projeto "Uso de dispositivos móveis inteligentes na classificação de solos brasileiros – SmartSolos". Este projeto é liderado pela Embrapa Solos (Rio de Janeiro, RJ) em parceria com a Embrapa Informática Agropecuária (Campinas, SP). Um de seus alvos é levar a classificação de solos para smartfones e tablets para que os interessados tenham a classificação de um determinado solo de forma automática, a partir de seus parâmetros.

O trabalho descrito nesta pesquisa objetivou o desenvolvimento de um classificador baseado em algoritmos de aprendizado de máquina para integrar o sistema do SmartSolos. Especificamente, são utilizadas redes neurais multicamadas de aprendizado profundo (deep learning), na forma de perceptrons multicamadas.

Não foram encontrados na literatura publicações relativas ao uso de algoritmos de redes neurais artificiais aplicadas à classificação de solos com a abrangência dos dados que se está tratando neste trabalho. Em uma escala menor, Biondi Neto et al. (2007) relataram o uso de redes neurais para classificar solos usando dados obtidos de ensaios com cone de penetração montado com sistema hidráulico. Nesses ensaios, as forças medidas durante a penetração do cone e seu atrito lateral geram 14 variáveis que são usadas

como entrada em um perceptron multicamadas. Na saída, obtém-se a indicação de um dos 12 tipos de solo explorados no trabalho. Este método de classificação é diferente do descrito no SiBCS, assim como são suas variáveis de entrada e de saída, cujas classes possuem nomenclatura diferente.

Mendes (2014) usou redes neurais do tipo perceptron multicamadas para estimar a retenção e disponibilidade de água em solos do estado de Santa Catarina. Foram usados 940 horizontes pertencentes a 57 perfis de solo. Neste caso, a rede foi usada como um regressor, pois o objetivo eram os valores numéricos das funções de pedotransferência, ao invés da classificação do solo, que é uma variável categórica. Para estimar os valores numéricos das funções de pedotransferência foram usados entre 7 e 11 variáveis de entrada, tendo-se obtido, para a melhor configuração de rede neural testada, valores de R^2 entre 0,58 e 0,99, dependendo da função de pedotransferência.

Bisi et al. (2015) relataram o uso de redes neurais artificiais para classificação de níveis de degradação de solos, usando como entrada 5 atributos de sua análise química. Usou-se 55 amostras, sendo 11 de cada uma das 5 classes possíveis de saída, correspondentes aos níveis de degradação. As amostras foram divididas em 33 para treinamento, 11 para validação e 11 para teste da rede. Após o treinamento, a rede foi testada em operação com 108 amostras de solos de área degradada, tendo-se relatado que os resultados foram muito bons, apresentando erro de $1{,}69 \times 10^{-4}$.

Schmitt (2009) usou redes neurais do tipo perceptron multicamadas para auxiliar no processo de interpretação de dados geofísicos obtidos em sondagens, tais como: profundidade, perfis raios gama, potencial espontâneo, resistência e resistividade. Os perfis foram classificados em três classes: 1) arenito; 2) siltito; e 3) carvão. Foram usadas 1227 amostras para treinamento, sendo 409 de cada classe, e 3171 amostras para teste, obtendo-se uma acurácia de 81%.

Enquanto em Schmitt (2009) os perfis são classificados em três classes, no trabalho de pesquisa aqui relatado existem 13 classes só no primeiro nível: 1) ARGISSOLO; 2) CAMBISSOLO; 3) CHERNOSSOLO; 4) ESPODOSSOLO; 5) GLEISSOLO; 6) LATOSSOLO; 7) LUVISSOLO; 8) NEOSSOLO; 9) NITOSSOLO; 10) ORGANOSSOLO; 11) PLANOSSOLO; 12) PLINTOSSOLO; e 13) VERTISSOLO. Para o segundo nível, tem-se 59 classes possíveis na base, tais como: LATOSSOLO VERMELHO, LATOSSOLO

AMARELO, etc. Para o terceiro nível, são 177 classes e para o quarto, 496 classes.

Material e Métodos

1. Aquisição e tratamento dos dados

A base de dados utilizada neste trabalho para treino e teste do sistema procede do Instituto Brasileiro de Geografia e Estatística (IBGE, 2018) e possui classificação até o quarto nível categórico. A base de dados possui tanto atributos categóricos, quanto numéricos. Os dados categóricos contêm, por exemplo, classe textural (média, argilosa, muito argilosa, etc.), presença de cascalho (cascalhenta, pouco cascalhenta, sem cascalho etc.), consistência úmida (friável, muito friável, solta etc), consistência pegajosa, matiz (4YR; 2,5YR; 7,5YR etc). Os atributos numéricos contêm, por exemplo, limites inferior e superior do horizonte (valores em cm), teor de argila (valor em g/kg), teor de carbono orgânico (valor em g/kg), pH em solução KCL, pH em água, capacidade de troca de cátions (valor em cmol/kg).

Usou-se um total de 4.221 perfis de solo, sendo 2.956 usados na fase de treino da rede neural e 1.265 perfis usados para teste. Os perfis, por sua vez, são subdivididos em horizontes, mas esta divisão não é uniforme. Existem perfis que possuem apenas um horizonte, enquanto outros possuem 8 horizontes. Então, nesta subdivisão dos perfis para treino e teste também foram considerados os horizontes, de modo que não ficasse uma parte dos horizontes de um perfil na base de treino e outra parte deste mesmo perfil na base de teste. Assim, todos os horizontes de um perfil estão em uma base ou em outra.

Durante a fase de treino, o programa separa os 2.956 perfis em outros 2 subconjuntos, pois a cada iteração a rede verifica a sua acurácia extraindo dados de um conjunto de validação. Esta separação é feita na proporção de 70% para treino e 30% para validação, resultando, portanto, em 2.069 usados na fase de aprendizado durante o treino e 887 perfis usados na validação do que a rede aprendeu e, consequentemente no cálculo da correção que o sistema deve sofrer para o ajuste correto do aprendizado. Vale ressaltar que

a base de teste não é vista em nenhum momento pela rede durante o treinamento, caso contrário, o teste não seria válido.

Esta base foi analisada quanto aos valores faltantes em seus atributos, de forma que atributos que estavam preenchidos em no máximo 20% do conjunto dos dados foram eliminados. Outras transformações ocorreram aos atributos que permaneceram, para que pudessem ser devidamente processados pelos algoritmos de treinamento, conforme a seguir:

1.1. Atributos numéricos

Sabe-se que algoritmos de aprendizado de máquina não executam adequadamente quando seus atributos têm valores muito discrepantes (Geron, 2017). Neste caso, é como se alguns valores “gritassem” e outros “sussurrassem”. Logo, o sistema de aprendizado da rede teria que atribuir pesos menores para o primeiro e maiores para o segundo, o que causaria problema com o cálculo dos gradientes de correção dos pesos. Uma forma de resolver o problema é por meio da padronização (*standardization*). Nesta operação, a média e o desvio padrão dos valores de cada atributo são calculados e os valores originais são substituídos em função desses valores. Primeiro, se subtrai o valor da média do valor do atributo e depois se divide pelo desvio padrão. Então, tem-se sempre um valor relativo à média, e não seus valores absolutos. Quem está próximo à média, tem valor próximo de 0, portanto. Isto tem lógica do ponto de vista de classificação, pois quem está na média não deveria excitar o sistema de classificação nem para baixo nem para cima em termos de classificação. Um cuidado que se teve neste processo é de guardar as médias originais de cada atributo para que quando novos perfis não vistos pelo sistema sejam submetidos, estes sejam também padronizados antes de serem submetidos à classificação. Os dados faltantes foram preenchidos com a média. Assim, quando o processo de padronização é aplicado, eles recebem valor zero, não influenciando no processo de otimização.

1.2. Atributos categóricos

Para esta classe de atributos, surge um novo problema, pois eles precisam ser transformados também em números para serem tratados matematicamente. Um atributo que possua três valores distintos na base, por exemplo AMARELO, AZUL, VERMELHO, poderiam ter valores 0, 1 e 2, respectivamente para as três situações. Ocorre que a distância numérica entre os valores atribuídos para AMARELO e AZUL (1 – 0 = 1) e a distância entre o AMARELO e o VERMELHO (2 – 0 = 2) gera uma distorção que não tem significado real, pois uma cor não é significativamente diferente da outra para gerar esta diferença numérica. A solução para este problema é implementada em uma técnica chamada *one-hot bit encoding*. Nesta solução, cada valor de um atributo gera uma coluna diferente, à qual é atribuído valor 1 ou 0. Se o objeto é azul, a coluna correspondente ao valor AZUL é marcada com 1 e as demais com 0. Esta expansão causa um aumento significativo do número de atributos, mas é a recomendada para retirar este viés de diferença numérica que não existe na realidade. Devido a esta transformação, as 52 variáveis usadas na classificação são transformadas em 3.028 variáveis de entrada na rede neural. Os dados categóricos com valores faltantes receberam um marcador “?” na coluna correspondente para marcar esta ausência e indicar que nenhuma das colunas do *one-hot bit encoding* deve ser preenchida.

2. Sistema de classificação usando redes neurais multicamadas

O sistema usado neste trabalho para a classificação dos perfis é uma rede neural do tipo perceptron multicamadas. Isoladamente, o perceptron é um modelo de rede neural muito simples, pois implementa uma função que realiza a soma ponderada das suas entradas e aplica uma função degrau para ativar ou não a saída.

Entretanto, quando os perceptrons são usados com funções de ativação que comprimam os valores das entradas (squashing functions), em redes dispostas com pelo menos uma camada oculta, constrói-se um aproximador universal que pode, em teoria, representar qualquer função contínua em um subconjunto limitado de R^n (Goodfellow et al., 2016). Esta capacidade de representação de qualquer função foi estendida por Leshno et al. (1993) para

qualquer mapeamento de um espaço discreto de dimensões finitas para outro, incluindo funções não diferenciáveis em qualquer ponto, como a função de ativação rectified linear unit (ReLU) usada neste trabalho.

A Figura 1 ilustra a diferença entre algumas das funções de ativação usadas em redes neurais. As duas primeiras imagens representam a função sigmóide e a função degrau, que podem ser consideradas compressoras (squashing functions), e a terceira é a função de ativação ReLU.

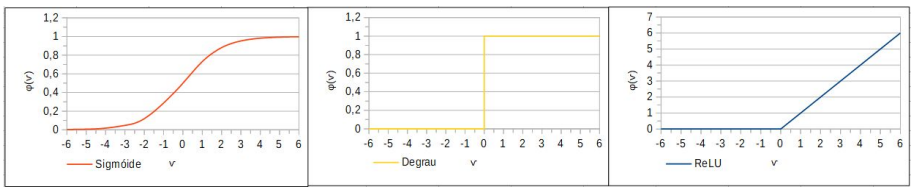

Figura 1. Funções de ativação comumente usadas em redes neurais.

O sistema foi implementado na linguagem de programação Python 2.7. Sobre o ambiente que implementa a linguagem Python, foram ainda instaladas duas plataformas: TensorFlow e Keras. TensorFlow é uma plataforma de código aberto, desenvolvida pela Google, com o propósito de desenvolver pesquisa em aprendizado de máquina e em redes neurais profundas (Geron, 2017). Esta plataforma possui suporte para processamento paralelo usando unidades de processamento gráfico (Graphics Processing Units (GPUs)).

Keras é uma API de alto nível para redes neurais escrita em Python que executa sobre a plataforma TensorFlow (Chollet, 2018). O principal objetivo do uso do Keras é a redução na sobrecarga cognitiva do usuário necessária para se implementar modelos de redes neurais, pois coloca o usuário como centro da atividade de construção dos modelos, permitindo uma prototipação rápida que executa tanto em CPUs, quanto em GPUs.

O núcleo da implementação para a geração de um modelo de rede neural multicamadas, do tipo perceptron, é feito de forma relativamente simples, pois o Keras se encarrega de traduzir para o TensorFlow as diretivas de criação das camadas. No exemplo abaixo estão os comandos do Keras para a criação da rede e o sumário do modelo criado (Figura 2).

Comandos para geração do modelo:

```
model = Sequential()
model.add(Dense(hl1Neurons, input_dim=mInput))
model.add(Activation('relu'))
model.add(Dense(hl2Neurons, input_dim=hl1Neurons,
init='normal', kernel_regularizer= regularizers.l2(0.003)))
model.add(Activation('relu'))
model.add(Dense(hl3Neurons, input_dim=hl2Neurons,
init='normal', kernel_regularizer= regularizers.l2(0.003)))
model.add(Activation('relu'))
model.add(Dense(hl4Neurons, input_dim=hl3Neurons,
init='normal',kernel_regularizer= regularizers.l2(0.003)))
model.add(Activation('relu'))
model.add(Dense(mOutput, init='normal'))
model.add(Activation('softmax'))
# Compile model
model.compile(loss='categorical_crossentropy',
optimizer='SGD',  metrics=['accuracy'])
```

Sumário do modelo:

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 12112)             36687248
_________________________________________________________________
activation_1 (Activation)    (None, 12112)             0
_________________________________________________________________
dense_2 (Dense)              (None, 6056)              73356328
_________________________________________________________________
activation_2 (Activation)    (None, 6056)              0
_________________________________________________________________
dense_3 (Dense)              (None, 3028)              18340596
_________________________________________________________________
activation_3 (Activation)    (None, 3028)              0
_________________________________________________________________
dense_4 (Dense)              (None, 13)                39377
_________________________________________________________________
activation_4 (Activation)    (None, 13)                0
=================================================================
Total params: 128,423,549
Trainable params: 128,423,549
Non-trainable params: 0
```

Figura 2. Geração do modelo neural e seu sumário durante a execução.

O primeiro desses comandos, “model = Sequential()”, estabelece que será criado um modelo sequencial, ou seja uma pilha de camadas na qual uma camada está conectada na camada subsequente, formando uma sequência linear. Isto é necessário porque o Keras dispõe de meios para a construção de grafos arbitrários de interligação entre as camadas.

A partir daí, adicionam-se as camadas a “model” com “model.add”. O parâmetro “Dense”, indica que deverá ser criada uma camada densamente conectada, em que cada elemento se conecta com todos os da camada de entrada. Este comando recebe o número de neurônios que será criado na camada de saída, a dimensão da entrada que provém da camada anterior ou da entrada da rede, e também especifica se utilizará algum tipo de regularizador na matriz de pesos das conexões.

Um regularizador é definido como qualquer modificação que se faz no algoritmo de aprendizado com o objetivo de reduzir seu erro de generalização, mas não seu erro de treinamento (Goodfellow et al, 2016). Uma forma de se fazer isso é adicionando uma penalidade na função que se deseja otimizar durante o treinamento da rede. Explicando melhor, o treinamento de uma rede neural envolve a otimização de uma função objetivo que procura aproximar o que a rede está recebendo como entrada com o que se espera como resposta. Quando esta resposta difere do esperado, o algoritmo de otimização, neste caso o *back propagation*, realiza uma operação nos pesos dos neurônios da última camada e de todas as camadas para trás para corrigir o problema. Ocorre que esta propagação para trás tem uma tendência a fazer com que os pesos de algumas conexões aumentem muito. Assim, para manter os pesos sob controle, adiciona-se um regularizador para forçar que o aumento de alguns pesos ocorram sempre em detrimento de outros, para forçar o algoritmo a fazer uma escolha controlada.

Neste trabalho usa-se um regularizador que considera a norma L^2. Este regularizador adiciona, à função objetivo, a soma dos quadrados dos pesos, mantendo, desta forma, o processo de ajuste dos pesos controlado. O parâmetro 0.003 indica ao algoritmo o quanto a minimização dos quadrados dos pesos influenciará na função objetivo.

O parâmetro init=“normal” indica que a matriz de pesos seja iniciada com valores extraídos da distribuição normal com média 0 e desvio padrão 0.05. Esta é uma das possibilidades de estabelecimentos dos pesos de partida

do treinamento. Existem outras, como a distribuição uniforme, lecun_normal, etc. Selecionou-se a distribuição normal porque ela é uma das distribuições mais usadas para modelar fenômenos naturais.

Para o projeto da arquitetura da rede, usou-se múltiplos do número de variáveis de entrada, onde 6-4-2 significa uma rede que tem na primeira camada escondida 6 vezes o número de variáveis de entrada, na segunda 4 vezes e na terceira 2 vezes. Foram testadas várias arquiteturas de redes perceptron multicamadas: 8-6-4-2, 8-4-2, 6-4-2, 4-2, 4-6-4-2, 4-2-1 e 1-0.5-0.25, entretanto, a que apresentou os melhores resultados foi a 4-2-1. Para a classificação no primeiro nível categórico, como são 3.028 dados de entrada e 13 possíveis classes na saída, a rede tem a configuração mostrada na Figura 3.

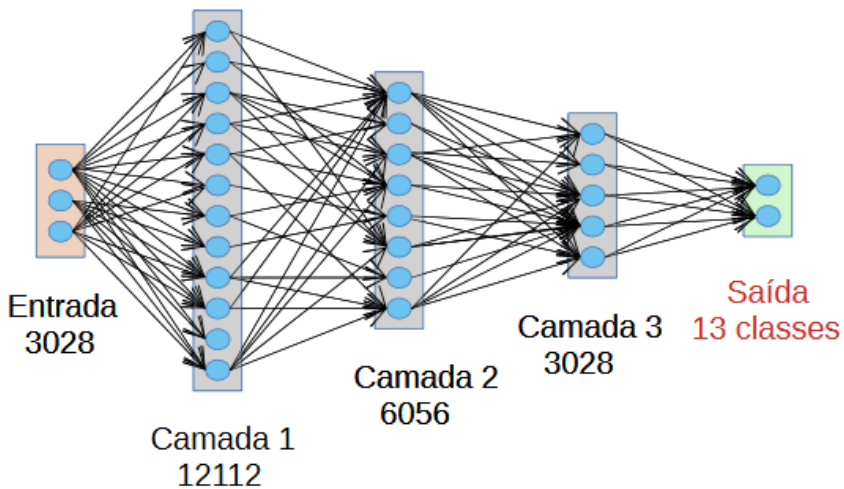

Figura 3. Arquitetura da rede neural para classificação no primeiro nível categórico.

O número de variáveis de entrada resulta de uma expansão no número de variáveis originais de acordo com as técnicas de tratamento dos dados descrita na seção 1.

3. Treinamento da rede neural

A função de entropia cruzada (*cross entropy*), usada como função de custo para guiar o algoritmo de treinamento da rede neural, é usada frequen-

temente quando se tem um problema de classificação. Matematicamente, esta função encontra-se representada na Equação 1 (Geron, 2017). Sem entrar em detalhes teóricos, dada uma configuração do sistema em termos dos seus parâmetros (θ^k), esta função procura minimizar a diferença entre a distribuição de probabilidade que o sistema calculou da instância i pertencer à classe k ($\hat{p}_k^i$), com a distribuição real (y_k^i) dos valores categóricos, ou seja, ela compara duas distribuições de probabilidades, uma que está resultando do treinamento da rede e a distribuição real que os dados de treinamento e validação possuem.

A cada iteração, o algoritmo avalia esta diferença entre as duas distribuições e calcula a correção que os pesos da rede devem sofrer para minimizar o erro. Para o cálculo desta correção, usou-se o otimizador *stochastic gradient descent* (SGD) (Geron, 2017), conforme Equação 2. Este gradiente indica a direção para a correção do modelo. De posse da direção, os novos valores de θ^k são calculados usando-se a Equação 3, onde ϵ é a taxa de aprendizado (*learning rate*).

- Equação 1: $J(\Theta) = -\frac{1}{m_i}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^i \log(\hat{p}_k^i)$
- Equação 2: $\nabla_{\theta^k} J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}(\hat{p}_k^i - y_k^i)x^i$
- Equação 3: $\theta^k = \theta^k - \epsilon\nabla_{\theta^k} J(\Theta)$

onde θ^k é o modelo para a classe k, $\hat{p}_k^i$ o valor do modelo para a classe k e y_k^i é 1 se a classe alvo da i-ésima instância é a classe k; caso contrário ele é igual a 0. Os valores de K e m são respectivamente o número de classes que o sistema precisa classificar e o tamanho do *batch* em que o sistema está operando, conforme explicado a seguir.

O cálculo do gradiente é um método usado em otimização de sistemas que considera a derivada em primeira ordem da função a ser otimizada e calcula o passo a ser dado na direção do ponto ótimo. O método é chamado estocástico porque trabalha com pequenos *batches*, extraídos da base de dados de treino, uma vez que seria computacionalmente intratável a utilização do conjunto completo de dados da base a cada iteração do algoritmo.

Resultados e Discussão

A Figura 4 mostra os resultados obtidos durante o treinamento do modelo para o primeiro nível de classificação. A linha azul representa os resultados obtidos com os dados amostrados no conjunto de treino e o amarelo os resultados da validação. Embora os dados de treino apresentem desempenho próximo a 100% de acurácia, o conjunto de validação já demonstra que os resultados são bem aquém, na faixa dos 80%, atingindo um pico por volta das 90 iterações (epochs).

Embora se perceba uma tendência de melhoria da acurácia ao longo do treinamento, a medida da perda na função de custo (model loss) otimizada já cai drasticamente logo nas primeiras iterações. Isto ocorre porque a função de custo usada neste caso a função de entropia cruzada converge rapidamente para o seu ponto ótimo, pois o sistema rapidamente atinge mais de 90% de acerto no treino. Embora haja esta convergência rápida, observa-se que ainda há vantagem em se prosseguir o treinamento até 100 iterações, dada a tendência de melhoria observada.

Quando apresentado à base de teste, o sistema teve um desempenho ainda menor do que os 80% do conjunto de validação, de cerca de 63,38%, como já era esperado, uma vez que estes dados nunca haviam sido apresentados ao sistema, enquanto que os de validação guiaram o processo de otimização para a escolha dos melhores hiperparâmetros e da melhor arquitetura.

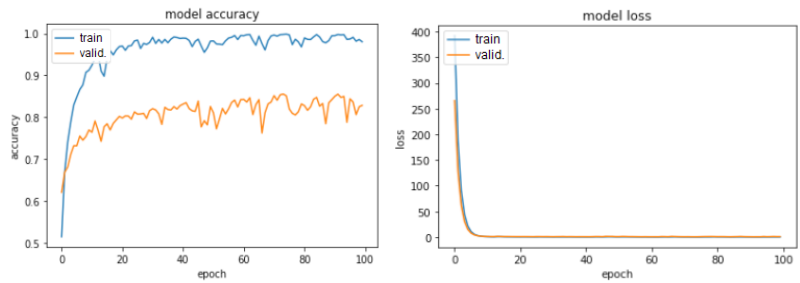

Figura 4. Gráfico de convergência da rede neural para o primeiro nível categórico

Embora o escore global de classificação do sistema seja 63,23, os valores variaram muito de classe para classe, conforme se observa pela matriz de confusão (Figura 5) obtida para o primeiro nível de treinamento. Nesta matriz, a diagonal representa o percentual das instâncias que foram corretamente classificadas para uma dada classe e os outros elementos da linha fora da diagonal, as demais instâncias daquela classe que foram erroneamente classificadas em outras classes. Por exemplo, para a classe ARGISSOLO 78% das instâncias apresentadas foram classificadas como ARGISSOLO, enquanto que 7% erradamente classificadas como CAMBISSOLO, 1% como CHERNOSSOLO, 9% como LATOSSOLO, 1% como NEOSSOLO, 2% como PLANOSSOLO e 2% como PLINTOSSOLO.

As classes que tiveram as maiores acurácias foram: a) ARGISSOLO (78%); b) GLEISSOLO (74%); e c) CHERNOSSOLO (72%). As de menor acurácia foram: a) ORGANOSSOLO (0%); b) VERTISSOLO (0%); e c) ESPODOSSOLO (22%). O sistema não aprendeu corretamente estas últimas classes e isto se deve ao alto desbalanceamento do número de instâncias disponíveis para cada classe, tanto na base de treino quanto na de teste. Por exemplo, ORGANOSSOLO possui apenas 8 perfis na base de treino e 2 na base de teste. VERTISSOLO possui 25 perfis na base de treino e 7 na base de teste; e ESPODOSSOLO possui 46 perfis na base de treino e 18 na base de teste. Nos perfis mais frequentes, tem-se 739 de LATOSSOLO e 941 de ARGISSOLO.

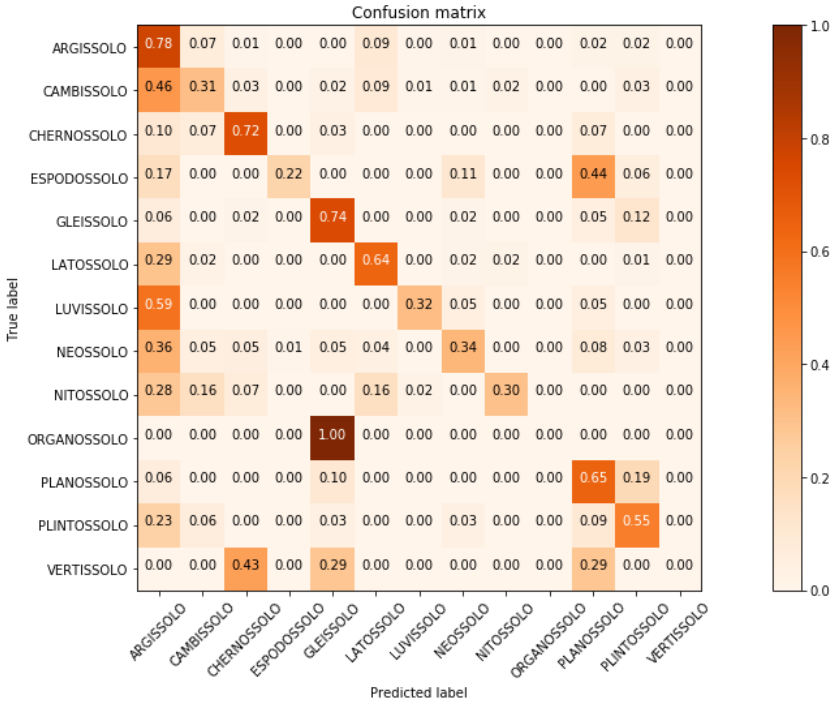

Figura 5. Matriz de confusão para o primeiro nível de classificação.

O desbalanceamento entre classes, somado ao baixíssimo número de instâncias de algumas classes, impede que o sistema aprenda a diferenciá-las corretamente. A situação se torna ainda mais crítica nos demais níveis de classificação. Quando processamos a classificação para o segundo nível, o valor tem uma queda significativa, para cerca de 13% de acurácia. Isto ocorre porque o sistema salta de 13 classes no primeiro nível para 59 classes no segundo nível. Uma forma de melhorar este desempenho foi a construção de uma arquitetura em cascata, conforme ilustrado na Figura 6.

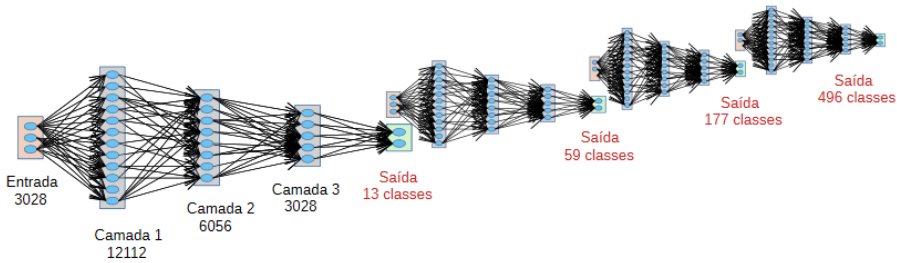

Figura 6. Redes neurais em cascata para melhorar a classificação.

Com esta arquitetura, ao invés do sistema tentar classificar um perfil como LATOSSOLO AMARELO diretamente, ele usa como entrada, além das informações do perfil, o resultado da classificação anterior, que avalia que se trata de um LATOSSOLO. Então, esta informação é usada como antecedente na segunda rede, fazendo com ela passe a representar, por exemplo, qual a probabilidade de um perfil ser LATOSSOLO AMARELO, dado que existe uma probabilidade de que ele seja LATOSSOLO.

Usando-se esta arquitetura em cascata, na qual os níveis são classificados em sequência, tendo a informação do nível anterior, os resultados foram um pouco melhores, alcançando 48.14% de acurácia para o segundo nível, que possui 59 classes de saída; 37,76% de acurácia para o terceiro nível, que possui 177 classes; e 27,79% de acurácia para o quarto nível, que possui 496 classes, conforme ilustrado na Tabela 1.

Tabela 1. Valores de acurácia e índice Kappa para as redes multicamadas.

Nível de Classificação	Acurácia	Índice Kappa
Nível 1	63,38	0,529
Nível 2	48,14	0,418
Nível 3	37,76	0,339
Nível 4	27,79	0,249

Esta tabela contém também outra importante medida para avaliar o grau de concordância quando se observa o número de instâncias que gerou a matriz de confusão, o índice Kappa. Este índice ou coeficiente é calculado de acordo com a Equação 4 (Bishop et al., 1975).

- Equação 4: $K = \frac{N \sum_1^r x_{ii} - \sum_1^r x_{i+} x_{+i}}{N^2 - \sum_1^r x_{i+} x_{+i}}$

Transpondo para o problema de classificação de solos, x_{ii} são todas as instâncias corretamente classificadas, x_{+i} são todas as que deveriam ter sido classificadas corretamente como solo i, e x_{i+} são todas as que foram classificadas como solo i. Este coeficiente mede, portanto, o grau de dispersão que ocorreu no sistema classificador, avaliando a matriz de confusão como um todo. De acordo com a Tabela 2, a classificação obtida para os níveis 1 e 2 podem ser consideradas boas, enquanto que as dos níveis 3 e 4 são consideradas razoáveis.

Tabela 2. Desempenho do índice Kappa.

Índice Kappa	Desempenho
<0	Péssimo
0 < k ≤ 0,2	Ruim
0,2 < k ≤ 0,4	Razoável
0,4 < k ≤ 0,6	Bom
0,6 < k ≤ 0,8	Muito bom
0,8 < k ≤ 1,0	Excelente

Fonte: Fonseca (2000).

Conclusões

Os resultados descritos mostram o alto potencial de uso da rede neural de aprendizado profundo aqui descrita para a classificação de perfis de solo. Entretanto, os resultados não foram tão bons quanto imaginados para todas as classes, principalmente devido ao forte desbalanceamento do número de instâncias de cada classe no banco de dados utilizado e reduzido número de instâncias em várias classes, já no nível 1. Este problema se torna ainda mais evidente nos demais níveis devido ao aumento considerável das classes possíveis. Ainda assim, o índice Kappa dos resultados para os níveis de classificação 1 e 2 são considerados bons na literatura, enquanto que para os níveis 3 e 4 é considerado razoável.

Agradecimentos

Agradecemos ao IBGE pela disponibilização dos dados para download online e à NVIDIA Corporation pela doação da placa GeForce usada no processamento e teste das redes neurais.

Referências

BIONDI NETO, L.; SIEIRA, A. C. C. F.; DANZIGER, B. R.; SILVA J. G. S. da. Classificação de solos usando-se redes neurais artificais. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 39., 2007, Fortaleza. **A pesquisa operacional e o desenvolvimento sustentável:** anais. Fortaleza: [s.n.], 2007. 12 p. Disponível em: <http://www.din.uem.br/sbpo/sbpo2007/pdf/arq0130.pdf>. Acesso em: 20 out. 2019.

BISHOP, Y.; FIENBERG, S.; HOLLAND, P. **Discrete multivariate analysis**: theory and practive. Cambridge: MIT, 1975.

BISI, B. S.; BONINI NETO, A.; BONINI, C. dos S. B. Redes neurais artificiais: utilização do algoritmo de retropropagação para classificação de grupos de biossistemas, parte 2: aplicação. **Fórum Ambiental da Alta Paulista**, v. 11., n. 2, 2015. DOI: http://dx.doi.org/10.17271/1980082711220151098.

CHOLLET, F. **Deep learning with Python**. Shelter Island: Manning Publications, 2018.

FONSECA, L. M. G. **Processamento digital de imagens**. São José dos Campos: INPE, 2000.

GERON, A. **Hands-on machine learning with Scikit-Learn & TensorFlow:** concept, tools, and techniques to build intelligent systems. Sebastopol: O'Reilly Media, 2017.

GOODFELLOW, I.; BENGIO, Y; COURVILLE, A. **Deep learning.** Cambridge: The MIT Press, 2016. Disponível em: <http://www.deeplearningbook.org>. Acesso em: 18 jun. 2018.

IBGE. **Mapeamento de recurso naturais do Brasil**: escala 1:250.000: documentação técnica geral. Rio de Janeiro, 2018. 8 p. Disponível em: <http://geoftp.ibge.gov.br/informacoes_ambientais/vegetacao/vetores/escala_250_mil/DOCUMENTACAO_TECNICA_MRN.pdf>. Acesso em: 10 out. 2019.

LESHNO, M.; LIN, V. Y.; PINKUS, A.; SCHOCKEN, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. **Neural Networks**, v. 6, n. 6, p. 861-867, 1993. DOI: https://doi.org/10.1016/S0893-6080(05)80131-5.

MENDES, R. B. **Predição da retenção de água em solos de Santa Catarina através de Redes Neurais Artificiais**. 2014. 147 p. Dissertação (Mestrado em Ciência do Solo) – Universidade do Estado de Santa Catarina, Lages.

SANTOS, H. G. dos; JACOMINE, P. K. T.; ANJOS, L. H. C. dos; OLIVEIRA, V. A. de; OLIVEIRA, J. B. de; COELHO, M. R.; LUMBRERAS, J. F.; CUNHA, T. J. F. (Ed.). **Sistema Brasileiro de Classificação de Solos**. 2. ed. Rio de Janeiro: Embrapa Solos, 2006. 306 p.

SCHMITT, P. **Redes neurais artificiais aplicadas na classificação litológica das formações Palermo e Rio Bonito na Jazida do Leão – RS, com base em perfis geofísicos**. 2009. 92 p. Dissertação (Mestrado) - Universidade do Rio dos Sinos, São Leopoldo.

Embrapa

Informática Agropecuária

MINISTÉRIO DA
AGRICULTURA, PECUÁRIA
E ABASTECIMENTO

PÁTRIA AMADA
BRASIL
GOVERNO FEDERAL

CGPE 15680