Relatório Final de Iniciação Científica

Métodos de Otimização aplicados a redes neurais para detecção de anomalias em transações com cartão de crédito

Bolsista: Frederico José Ribeiro Pelogia

Orientador: Prof. Dr. Luís Felipe Bueno

Número do Projeto FAPESP: 2019/13420-4

Período de vigência: 01/09/2019 a 31/08/2020

Período abordado pelo relatório: 10/02/2020 a 31/08/2020

Departamento de Ciência e Tecnologia
Universidade Federal de São Paulo
Julho de 2020

Resumo do projeto

Neste projeto pretende-se estudar métodos de otimização, em especial algoritmos estocásticos, aplicados ao treinamento de redes neurais. Em um trabalho recente proposto por L. F. Bueno e J. M. Martínez, On the complexity of solving feasibility problems, 2018, foi apresentado um algoritmo de primeira ordem com bons resultados de complexidade para problemas de quadrados mínimos. Um dos principais pontos da pesquisa deste projeto será desenvolver uma versão estocástica desse algoritmo. Serão analisados os desempenhos dos algoritmos estudados quando aplicados à detecção de fraudes em operações de cartão de crédito utilizando a base de dados Credit Card Fraud Detection do Kaggle.

Resumo das realizações do período

Este relatório aborda as realizações do período de 10/02/2020 a 31/08/2020. Neste intervalo de tempo, as seguintes atividades foram concluídas:

- Estudo dos artigos [10] e [9]
- Experimentar a aplicação dos algoritmos implementados em Python, sem utilização de bibliotecas prontas de *machine learning*, à base de dados Credit Card Fraud Detection.
- Estudo do método de Levenberg–Marquardt, com base em [2].
- Estudo teórico de [5].
- Testes numéricos associados a [5].
- Aplicações de técnicas de [5] no problema de interesse.

Como planejávamos, foi submetido um resumo e um **vídeo-poster** para apresentação no Congresso Acadêmico UNIFESP 2020, que ocorreu virtualmente, dos dias 13 a 17 de julho.

Além disso, participamos do desenvolvimento da *Ferramenta de apoio ao planejamento de salas de aula pós pandemia de Covid-19*, que é um site com objetivo de otimizar as posições das carteiras nas salas de aulas de acordo com as medidas de distanciamento propostas pelas autoridades sanitárias. Esse projeto foi desenvolvido com pesquisadores da Instituições Unifesp, UEM e IFSP. Assim, houve contato com técnicas de empacotamento, otimização com restrições e a utilização do software ALGENCAN, do projeto TANGO[4].

1 Estudo dos artigos [10] e [9]

BAIXEI OS 2 ARTIGOS DIA 26/06 E ESTOU DANDO UMA OLHADA NELES. EM BREVE ESCREVO AQUI. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed diam vitae quam aliquam facilisis. In hac habitasse platea dictumst. Nam ac ligula cursus odio elementum tempor a non neque. Ut ullamcorper porttitor felis quis dictum. Aenean eu metus eu quam imperdiet vulputate. Phasellus tincidunt sit amet mauris a fringilla. Quisque dapibus risus eu luctus molestie. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed diam vitae quam aliquam facilisis. In hac habitasse platea dictumst. Nam ac ligula cursus odio elementum tempor a non neque.

2 Aplicação dos algoritmos implementados ao problema de deteção de fraudes

A forma como os métodos estavam implementados não era adequada para processar os dados do dataset do Kaggle, além de não possibilitar a alteração da estrutura da rede de forma dinâmica. Procurávamos, então, algum framework que nos permitisse montar uma rede de forma fácil e receber o gradiente em relação aos parâmetros ajustáveis para os refinarmos manualmente de diferentes maneiras. Um exemplo de interface para criação dinâmica de rede é a da bilioteca Keras[7] e está apresentada a seguir.

Listing 1: Interface do Keras

```
model = keras.models.Sequential([
    keras.layers.Dense(units=16, input_dim=30, activation="relu"),
    keras.layers.Dense(units=24, activation="relu"),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(20, activation="relu"),
    keras.layers.Dense(24, activation="relu"),
    keras.layers.Dense(1, activation="sigmoid"),
])
```

A solução foi alterar as implementações para o paradigma de orientação a objeto, construindo uma pequena biblioteca de *Deep Learning* com base na *joelnet*[6], de Joel Grus.

Módulos do repositório joelnet

A biblioteca de Joel era composta, originalmente de 7 módulos.

- O arquivo **data.py** controla a separação dos dados em "mini lotes"(*batches*).
- O arquivo **layers.py** define o comportamento genérico de uma camada e também as diferentes funções de ativação possíveis.
- O arquivo **loss.py** define as possíveis formulações da função Erro da rede. Apenas a *Mean Squared Error* está implementada.
- O arquivo **nn.py** define a Classe *NeuralNetwork*, que é a mais importante da biblioteca. Essa Classe é a portadora dos métodos responsáveis pelo Feed-Forward e pelo Backpropagation.
- O arquivo **optim.py** define os métodos de otimização que poderão ser utilizados par ao treinamento das redes. Originalmente possuía apenas o SGD (Método do Gradiente Estocástico).
- O arquivo **train.py** define a função *train*, que é responsável pelo treinamento da rede neural.
- O arquivo **tensor.py** define o tipo abstrato *Tensor*, que, para nossa implementação simples, nada mais é do que o *ndarray* da biblioteca Numpy.

Alterações e adições no módulo joelnet

Para utilizarmos o repositório apresentado, foram necessárias algumas modificações e a adição de algumas funcionalidades.

- Em **layers.py** foram adicionadas as funções de ativação Sigmoid e reLu, além de suas primeiras derivadas.

$$Sig(x) = \frac{1}{1+e^{-x}}, \qquad Sig'(x) = Sig(x) \cdot (1 - Sig(x))$$

$$reLu(x) = \frac{x + \sqrt{x^2+\epsilon}}{2}, \qquad reLu'(x) = \frac{1}{2}\left(\frac{x}{\sqrt{x^2+\epsilon}} + 1.0\right)$$

- Em **data.py** foi alterada a forma de sorteio de batches. Agora, para cada *epoch*, embaralhamos os dados e retiramos amostras adjuntas do tamanho do *batch_size*.

- Em **nn.py**, o método chamado *params_and_grads* ganhou 3 novas versões, para conseguir retornar a matriz Jacobiana e também valores antigos de parâmetro e de gradiente, pois são necessários para alguns dos métodos de otimização implementados.

- Em **train.py**, a função *train* agora retorna uma lista chamada *loss_list*, que contém os valores do erro de cada *epoch*, definido como a soma dos erros de cada batch avaliado. Essa lista pode ser utilizada para construir um gráfico do erro ao longo das epochs.

- Criado um módulo **jacobian.py** com 2 funções auxiliares para cálculo da matriz jacobiana.

- Em **optim.py**, temos agora os seguintes métodos de otimização:
  - Método do Gradiente Estocástico - SGD
  - RMSProp
  - SGD com momento de Nesterov
  - Adam
  - Método de Barzilai–Borwein
  - Levenberg Marquardt
  - Levenberg Marquardt com condição de busca linear

  Nas implementações atuais, foram adicionados novos tipos de parâmetros. Agora, cada camada, após receber o conteúdo da anterior multiplicado pelos pesos, soma um novo parâmetro, conhecido por *bias* (viés), antes de aplicar a função de ativação. Sendo assim, cada camada L_i tem valor $f_i(L_{i-1} \cdot W_i + b_i)$. Por consequência, o método *backpropagation* teve que ser atualizado, agora fazendo a regra da cadeia também para calcular as derivadas parciais em relação a esses novos termos.

Testando as novas implementações

Para testar as novas implementações dos métodos, cada um deles foi avaliado em 2 cenários.

Problema 1: Regressão de $y = x^2$

O problema é uma regressão simples da função $y = x^2$. O conjunto de treinamento é composto pelos números de 1 a 20 e o resultado esperado é composto pelo quadrado de cada um desses números. Por ser um problema mais rápido de se resolver, foi importante para pequenos testes e ajustes durante a implementação.

Problema 2: Deteção de Fraudes na base de dados do Kaggle

A base de dados *Credit Card Fraud Detection* possui 284,807 transações e 492 fraudes. Foi definida uma divisão de 70% dos dados para treinamento e 30% para teste. Para as comparações entre os métodos, foi fixada uma arquitetura de rede e a mesma inicialização de seus parâmetros.

3 Estudo do Método de Levenberg–Marquardt

Primeiramente foi feito um estudo sobre alguns métodos clássicos de segunda ordem.

Método de Newton

É um dos mais clássicos métodos de segunda ordem. Ele calcula a matriz Hessiana da função erro, resolve o sistema

$$\nabla^2 f(x^k)d = -\nabla f(x^k),$$

e utiliza a direção encontrada para dar o passo $x^{k+1} = x^k + td$.
O método com $t = 1$ apresenta convergência local quadrática[8], porém só consegue convergência global procurando um t adequado com uma busca linear, como a de Armijo[1]

$$f(x^k + td^k) \leq f(x^k) + \alpha t \nabla f(x^k)^T d^k.$$

Como a matriz Hessiana possui formato $n \times n$, onde n é o número de variáveis, seu cálculo demonstra-se custoso. O método é mais caro ainda, pois além de calcular essa matriz, resolve um sistema com ela e com o gradiente, resultando em algo da ordem de $O(n^3)$.

Quasi-Newton: Método da Secante, Barzilai-Borwein e Gauss-Newton

A ideia principal dos métodos Quasi-Newton é herdar a eficácia do Método de Newton utilizando alternativas ou aproximações mais baratas da Hessiana.

Método da Secante

Propõe a escolha de uma matriz M_k tal que

$$M_k s^{k-1} = y^{k-1},$$

onde

$$s^{k-1} = x^k - x^{k-1} \text{ e } y^{k-1} = \nabla f(x^k) - \nabla f(x^{k-1}).$$

Assim, pode-se encontrar d_k tal que

$$M_k d^k = -F(x^k),$$

para que seja possível fazer

$$x^{k+1} = x^k + d^k.$$

Método de Barzilai-Borwein

Sua proposta é ser um intermediário entre o Método do Gradiente, que possui complexidade $O(n)$, e o Método da Secante, que tem complexidade $O(n^2)$. Sendo

$$s^{k-1} = x^k - x^{k-1} \text{ e } y^{k-1} = \nabla f(x^k) - \nabla f(x^{k-1}),$$

A ideia é definir $M_k = \lambda_k I$, implicando que $\lambda_k s^k = y^k$. A melhor aproximação para essa equação é

$$\lambda_k = \frac{(s^k)^T y^k}{\|s^k\|^2}.$$

Então, a direção tomada pelo método será

$$d = -\frac{1}{\lambda_k}\nabla f(x^k),$$

sendo semelhante a uma iteração do Método do Gradiente com passo $\frac{1}{\lambda_k}$. O método também funciona bem em um formato estocástico, se assemelhando, nesse caso, ao SGD com passo $\frac{1}{\lambda_k}$.

Método de Gauss-Newton

Problemas não lineares de quadrados mínimos podem ser formulados da seguinte forma

$$f(x) = \|F(x)\|^2 = \sum_{i=1}^{N} f_i(x)^2.$$

Neste caso,

$$\nabla^2 f(x) = J_F(x)^T J_F(x) + \sum_{i=1}^{N} f_i(x)\nabla^2 f_i(x).$$

Uma possível aproximação para a Hessiana é $J_F(x)^T J_F(x)$, tendo em vista que o segundo termo ficará muito pequeno quando próximo do mínimo da função. Assim, a direção de Gauss-Newton é a direção d_k que satisfaz

$$(J_F(x)^T J_F(x))d_k = -\nabla f(x^k).$$

Segundo [3], o método é bem definido para quando $J_F(x)^T J_F(x)$ é invertível e, portanto, definida positiva.

Método de Levenberg-Marquardt

Por fim, o Método de Levenberg-Marquardt é uma mistura do Método de Gauss-Newton com um passo de Barzilai-Borwein. A dreção é calculada da seguinte forma

$$(J_F(x)^T J_F(x) + \lambda I)d_k = -\nabla f(x^k),$$

onde λ pode ser definido de diversas maneiras. Para a implementação realizada, foi considerado

$$\lambda = ||\nabla f(x^k)||.$$

Note que pequenos valores de λ deixam o método mais parecido com o Método de Gauss-Newton, enquanto valores altos de λ aproximam a um passo de Barzilai-Borwein.

4 Estudo teórico de [5]

O artigo propõe uma versão do método de Levenberg Marquardt em que é imposta uma condição de descida

$$f(x) <= f(x^k) - \alpha||x - x^k||^2$$

e realizada uma busca por um valor de λ que a satisfaça. Segue um pseudo-código do processo:

Listing 2: Levenberg Marquardt com busca linear

```
x, f = LevenbergMarquardt(x[k], lambda);
while(f > f[k] - alpha*(norm(x - x[k]))^2 ){
        lambda = 2*lambda;
}
x[k + 1] = x;
```

Além disso, o artigo demonstra[5] que em alguns casos a solução é atingida com $log(\varepsilon)$ avaliações de função.

5 Resultados e Discussões

Comparação dos métodos no Problema 1

Para as comparações referentes ao Problema 1, foi fixada uma arquitetura de rede **1-2-1**, com ativação **reLu**, **batch_size** = 2 e inicialização fixa dos parâmetros como

$$W_{L1\to L2} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \qquad b_{L1\to L2} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \qquad W_{L2\to L3} = \begin{pmatrix} 3 & 4 \end{pmatrix}, \qquad b_{L2\to L3} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}.$$

A Figura 1 apresenta o resultado das predições de cada método, juntamente com o valor da função erro na ultima *epoch*.

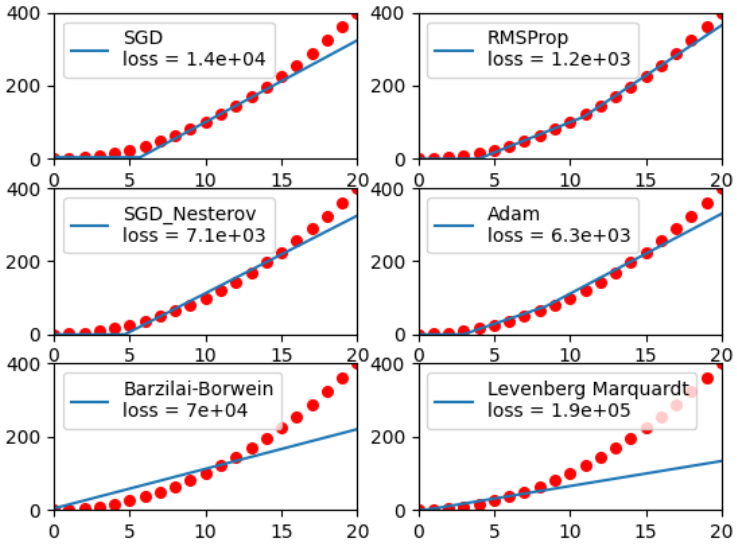

Figura 1: Resultados da aplicação dos métodos ao Problema 1 (Regressão de $y = x^2$). Fonte: Autor

Note que o melhor desempenho foi do **RMSProp**, método com passo adaptativo. Pode-se perceber que os 2 métodos quasi-Newton obtiveram os piores resultados no teste.

Comparação dos métodos no Problema 2

Para o problema 2, foi fixada uma arquitetura de rede **30-24-30-35-1**, com 3 camadas ocultas. A inicialização dos pesos foi aleatória, porém a mesma para todos os métodos. O tamanho do lote (**batch_size**) escolhido foi de 150 dados. A Figura 2 apresenta as curvas de Precision-Recall de cada um dos métodos, além do valor da área debaixo delas, indicado por AUC na legenda.

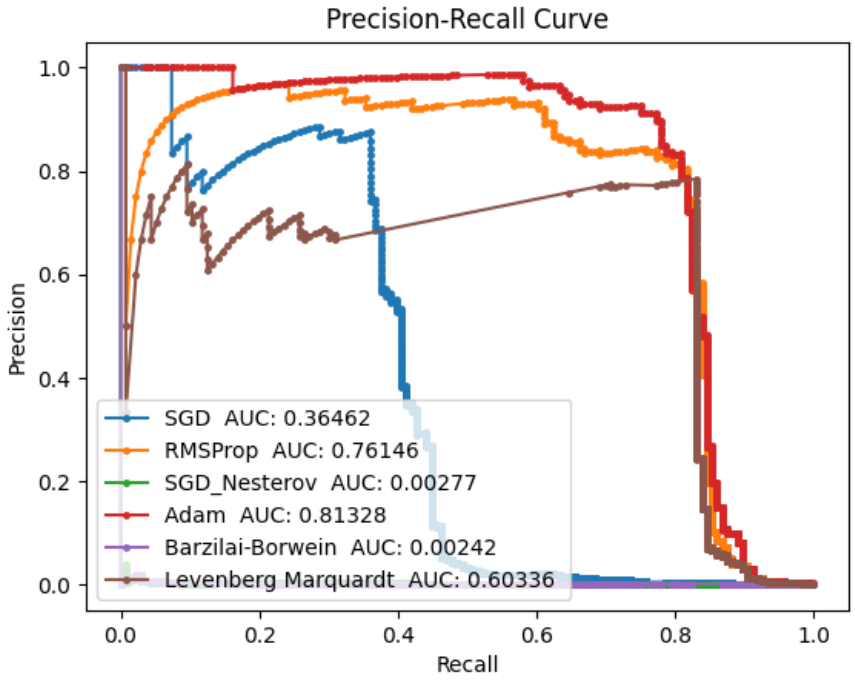

Figura 2: Comparação das curvas de Precision-Recall dos métodos aplicados ao Problema 2. Fonte: Autor

O método **Adam** foi o que mais se destacou, obtendo o maior valor de AUPRC, e os métodos **SGD_Nesterov** e **Barzilai-Borwein** tiveram um desempenho muito abaixo dos outros. Talvez com outras inicializações de parâmetros esses métodos poderiam apresentar melhorias. As curvas de Precision-Recall são montadas variando os *tresholds* de arredondamento e registrando os valores de Precision e de Recall. Foram registradas, também, as **matrizes de confusão** dos métodos para o *treshold* de 0.5. Essas matrizes estão apresentadas na Figura 4.

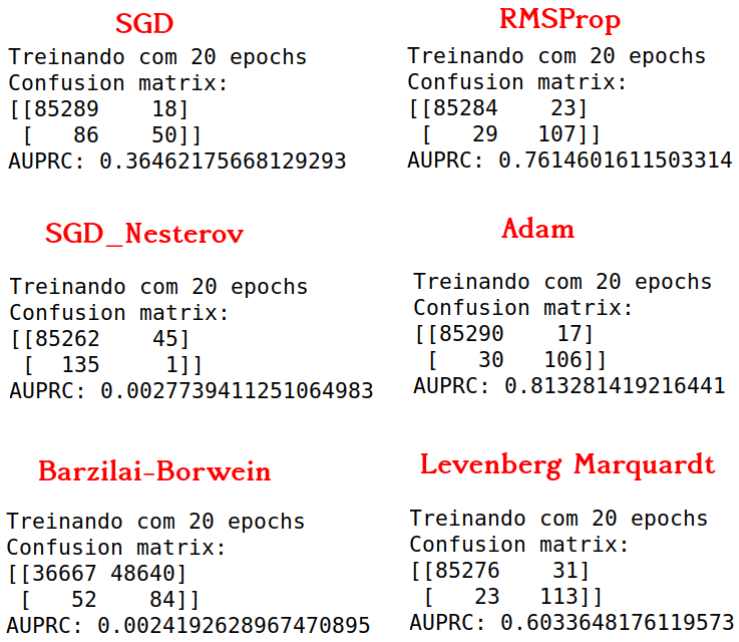

Figura 3: Comparação das matrizes de confusão dos métodos aplicados ao Problema 2. Fonte: Autor

******VERIFICAR POR QUE A SOMA NÃO ESTÁ BATENDO. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed diam vitae quam aliquam facilisis. In hac habitasse platea dictumst. Nam ac ligula cursus odio elementum tempor a non neque. Ut ullamcorper porttitor felis quis dictum. Aenean eu metus eu quam imperdiet vulputate. Phasellus tincidunt sit amet mauris a fringilla. Quisque dapibus risus eu luctus molestie. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed diam vitae quam aliquam facilisis. In hac habitasse platea dictumst. Nam ac ligula cursus odio elementum tempor a non neque.

Comparação com o Keras

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed diam vitae quam aliquam facilisis. In hac habitasse platea dictumst. Nam ac ligula cursus odio elementum tempor a non neque. Ut ullamcorper porttitor felis quis dictum. Aenean eu metus eu quam imperdiet vulputate. Phasellus tincidunt sit amet mauris a fringilla. Quisque dapibus risus eu luctus molestie. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed diam vitae quam aliquam facilisis. In hac habitasse platea dictumst. Nam ac ligula cursus odio elementum tempor a non neque.

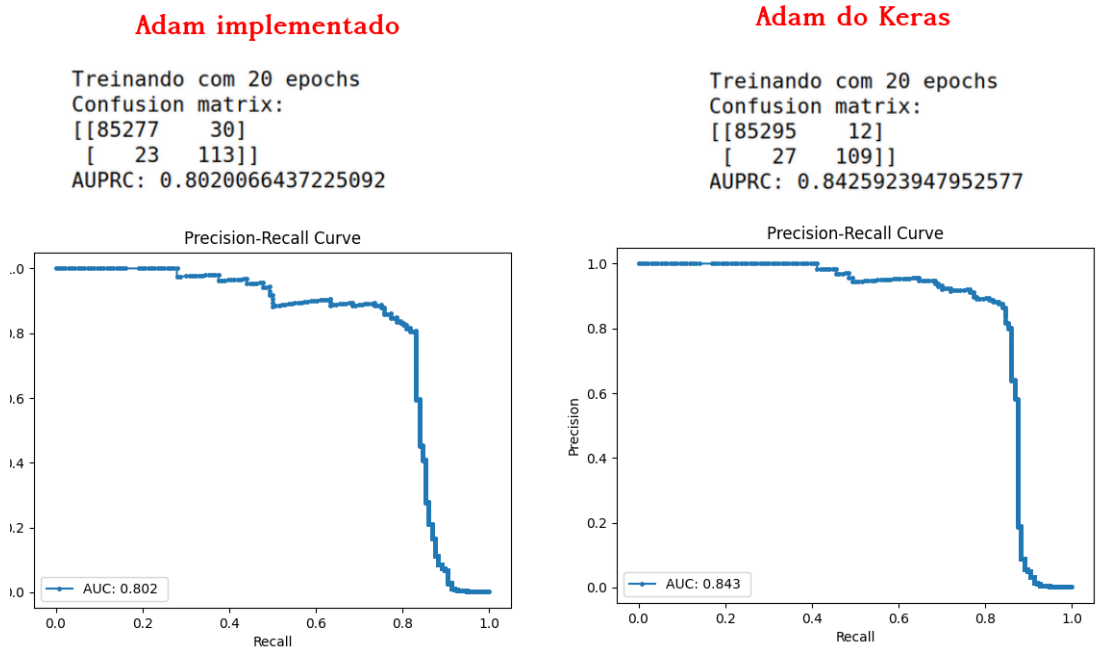

Figura 4: Comparação da implementação do Adam com o Adam da biblioteca Keras. Fonte: Autor

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed diam vitae quam aliquam facilisis. In hac habitasse platea dictumst. Nam ac ligula cursus odio elementum tempor a non neque. Ut ullamcorper porttitor felis quis dictum. Aenean eu metus eu quam imperdiet vulputate. Phasellus tincidunt sit amet mauris a fringilla. Quisque dapibus risus eu luctus molestie. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed diam vitae quam aliquam facilisis. In hac habitasse platea dictumst. Nam ac ligula cursus odio elementum tempor a non neque.

6 Conclusão

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed diam vitae quam aliquam facilisis. In hac habitasse platea dictumst. Nam ac ligula cursus odio elementum tempor a non neque. Ut ullamcorper porttitor felis quis dictum. Aenean eu metus eu quam imperdiet vulputate. Phasellus tincidunt sit amet mauris a fringilla. Quisque dapibus risus eu luctus molestie. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed diam vitae quam aliquam facilisis. In hac habitasse platea dictumst. Nam ac ligula cursus odio elementum tempor a non neque.

Utilização de recursos da reseva técnica da bolsa

Durante o período abordado por este relatório, não houve utilização dos recursos financeiros da reserva técnica da bolsa.

Referências

[1] Larry Armijo. "Minimization of functions having Lipschitz continuous first partial derivatives." Em: *Pacific J. Math.* 16.1 (1966), pp. 1–3. URL: https://projecteuclid.org:443/euclid.pjm/1102995080.

[2] K. A. Benatti, T. S. Nazaré e L. F. Bueno. *O método de Levenberg-Marquardt estocástico aplicado à redes neurais artificiais*. Artigo submetido. 2019.

[3] K. A. Benatti e A. A. Ribeiro. "O Método de Levenberg-Marquardt para o Problema de Quadrados Mínimos não Linear". Em: *II Simpósio de Métodos Numéricos em Engenharia - UFPR* (2017).

[4] E. G. Birgin. *TANGO Project*. URL: https://www.ime.usp.br/~egbirgin/tango/.

[5] Martínez J. M. Bueno L. F. e E. G. Birgin. "On the complexity of solving feasibility problems with regularized models". Em: *Optimization Methods and Software (GOMS)* (2020). DOI: 10.1080/10556788.2020.1786564.

[6] Joel Grus. *joelnet - live coding deep learning library*. URL: https://github.com/joelgrus/joelnet.

[7] *Keras: the Python deep learning API*. URL: https://keras.io/.

[8] J. M. Martínez e S. A. Santos. *Métodos Computacionais de Otimização*. Campinas: Departamento de Matemática Aplicada, IMECC - UNICAMP, 1995.

[9] F. S. Mennini, L. Gittoa, Russoa S., A. Cicchetti, M. Ruggeri, S. Coretti, G. Maurelli e P. M. Buscema. "Artificial neural networks and their potentialities in analyzing budget health data: an application for Italy of what-if theory". Em: *Quality & Quantity: International Journal of Methodology* 51.3 (2016), pp. 1261–1276.

[10] F. S. Mennini, L. Gittoa, Russoa S., A. Cicchetti, M. Ruggeri, S. Coretti, G. Maurelli e P. M. Buscema. "Does regional belonging explain the similarities in the expenditure determinants of Italian healthcare deliveries? An approach based on Artificial Neural Networks". Em: *Economic Analysis and Policy* (2017).