Redes Neurais Profundas para Análise e Síntese de Imagens e Vídeos
Lista 1: Perceptron and MLP

PROF. GILSON A. GIRALDI[1]

[1] Pós-Graduação em Modelagem Computacional
LNCC, RJ, Brazil
{gilson}@lncc.br

Abstract. Ano 2025.

DEADLINE: 12 April, 2025
Cuidado: Perde 2,0 por cada dia de atraso
Não serão aceitos listas/trabalhos após a entrega da correção

1 IMPORTANTE

- **Não serão aceitos** códigos como explicação de solução de exercícios.
- **Não serão aceitos** trabalhos escritos manualmente. Somente serão aceitos trabalhos em pdf gerados usando Word, Latex, ou equivalentes
- Forneça detalhes de implementação: linguagem, operadores, otimizador, inicialização dos pesos e bias, valores dos hyperparametros da função de perda, bibliotecas usadas, TODOS os parâmetros necessários para executar o programa, etc. O leitor tem que ser capaz de reproduzir sua implementação a partir da descrição da implementação.
- Os códigos e o pdf deve ser enviados em um arquico ZIP ou RAR.
- Usar corretamente os editores de fórmulas matemáticas de cada sistema de edição
- Justificar sua afirmações e conclusões
- Discutir os resultados apresentados nos gráficos e figuras. **Nunca** colocar um gráfico ou figura sem alguma explicação acompanhando.

2 Exercises

1. Implementation and test of perceptron model [1, 2]:
   (a) Generate and visualize a database S such that:
   $$S \subset \mathbb{R}^2 \times \{+1, -1\}, \tag{1}$$
   with $|S| = 100$ samples, composed by two classes C_1 and C_2.
   (b) Shuffle the dataset S and randomly subdivide it into disjoint sets $\mathbb{D}_{tr}$ and $\mathbb{D}_{te}$ for training and test, respectively. OBS: Verify if the classes C_1 and C_2 are **balanced** in $\mathbb{D}_{tr}$ and $\mathbb{D}_{te}$.
   (c) Implement the perceptron model (Figure 1) for classification in $\mathbb{R}^2 \times \{+1, -1\}$ and perform training using the training set $\mathbb{D}_{tr}$. Show some graphical configurations of the line that partitions the pattern space together with the final solution.
   (d) Evaluate the model using the accuracy measure (section 8.6) computed over $\mathbb{D}_{te}$.

2. Consider the FEI face image database and the gender classification problem (see link in [3]). Choose a feature space and apply leave-one-out multi-fold cross-validation explained in section 8.5, with $K = 4$, for perceptron model. In this case, use the perceptron available in libraries for neural network implementation, like Keras, Tensor flow or scikit-learn [4].

   (a) Show the graphical representation of the evolution of training and validation stages (see Figure 8.8 of [5]).

   (b) Perform a statistical analysis of the performance (section 8.6 of [5]) of the four models applied over the $\mathbb{D}_{te}$.

3. Repeat exercise 2 but now using a MLP network.

4. Compare Sigmoid and ReLU activation functions[6].

References

[1] Simon Haykin. *Neural Networks - A Comprehensive Foundation, Second Edition*. Prentice Hall, 2 edition, 1998.

[2] R. Beale and T. Jackson. *Neural Computing*. MIT Press, 1994.

[3] Gilson A. Giraldi. *Course GB500: Redes Neurais Profundas para Análise e Síntese de Imagens e Vídeos*. `https://www.lncc.br/~gilson/NN-GB500/`.

[4] Wikipedia. *Comparison of deep-learning software*. `https://en.wikipedia.org/wiki/Comparison_of_deep-learning_software`.

[5] Gilson A. Giraldi. *Course Monography: Machine Learning and Pattern Recognition*. `https://www.lncc.br/~gilson/NN-GB500/book.pdf`.

[6] Wikipedia. *ReLU (rectified linear unit) activation function*. `https://en.wikipedia.org/wiki/Rectifier_(neural_networks)`.

TABLE 3.2 Summary of the Perceptron Convergence Algorithm

Variables and Parameters:

$\mathbf{x}(n) = (m+1)$-by-1 input vector

$= [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$

$\mathbf{w}(n) = (m+1)$-by-1 weight vector

$= [b(n), w_1(n), w_2(n), \ldots, w_m(n)]^T$

$b(n) =$ bias

$y(n) =$ actual response (quantized)

$d(n) =$ desired response

$\eta =$ learning-rate parameter, a positive constant less than unity

1. *Initialization.* Set $\mathbf{w}(0) = \mathbf{0}$. Then perform the following computations for time step $n = 1, 2, \ldots$.

2. *Activation.* At time step n, activate the perceptron by applying continuous-valued input vector $\mathbf{x}(n)$ and desired response $d(n)$.

3. *Computation of Actual Response.* Compute the actual response of the perceptron:

$$y(n) = \text{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)]$$

where $\text{sgn}(\cdot)$ is the signum function.

4. *Adaptation of Weight Vector.* Update the weight vector of the perceptron:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)$$

where

$$d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1 \\ -1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2 \end{cases}$$

5. *Continuation.* Increment time step n by one and go back to step 2.

Figure 1: Percetron learning algorithm (Source [1]. See also [2])