MS211 – Cálculo Numérico (2º Semestre de 2021)
Marcos Eduardo Valle

Atividade 10 – Identificação de Lixo Eletrônico (SPAM).

Condições e Datas

O projeto deve ser realizado **individualmente** utilizando `Python`. Lembramos que o `Python` é livre e pode ser instalado, por exemplo, usando o ambiente `Conda` disponível em `https://conda.io`. Ele também pode ser acessado online usando o Google Colab através do link `https://research.google.com/colaboratory/`.

O projeto deve ser entregue no prazo especificado no Google Classroom. O arquivo deve descrever de forma clara os procedimentos adotados e as conclusões. Em particular, responda a(s) pergunta(s) abaixo de forma clara, objetiva e com fundamentos matemáticos. Recomenda-se que os códigos sejam anexados, mas **não serão aceitos trabalhos contendo apenas os códigos**! Pode-se submeter o arquivo .ipynb do Google Colab com os comandos e comentários.

O termo *lixo eletrônico* ou *spam* refere-se, de um modo geral, a mensagens (e-mails) indesejadas que são enviadas constantemente nos meios eletrônicos sem o consentimento do destinatário. Nesse projeto aplicaremos os conceitos de quadrados mínimos para auxiliar na identificação automática de *spams*. Especificamente, usaremos uma técnica de aprendizado de máquina que tem recebido bastante destaque nos últimos anos chamada *extreme learning machine* (EML). Em termos gerais, uma ELM é sintetizada e avaliada com base num conjunto de mensagens que já foram identificadas como spam ou não-spam pelo(s) usuário(s). O conjunto usado para sintetizar a EML é chamado *conjunto de treinamento* enquanto que o conjunto usado para avaliar o desempenho do modelo é chamado *conjunto de teste*. É importante destacar que o conjunto de teste não pode ser usado em nenhum momento para sintetizar a ELM. O aluno interessado em aprendizado de máquinas e nos detalhes da ELM pode consultar [1, 2].

Obtenção e Pre-Processamento dos Dados

Primeiramente, devemos baixar a base de dados `spambase` do repositório `OpenML` usando o scikit-learn (sklearn). Além de baixar os dados, iremos dividi-los em conjuntos de treinamento e teste e também aplicaremos uma sequência de pre-processamento. Especificamente, os seguintes passos incluem as bibliotecas e funções necessárias:

```
import numpy as np
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
```

Obtemos os dados, dividimos em conjuntos de treinamento e teste e efetuamos o pre-processamento usando os comandos:

```
X, y = fetch_openml(data_id = 44, return_X_y=True)

Xtr, Xte, ytr, yte = train_test_split(X,y,random_state=1234)

scaler = StandardScaler()
Xtr = scaler.fit_transform(Xtr)
Xte = scaler.transform(Xte)

encoder = LabelEncoder()
ytr = 2*encoder.fit_transform(ytr)-1
yte = 2*encoder.transform(yte)-1
```

Dessa forma, obtemos uma matriz $\texttt{Xtr} \in \mathbb{R}^{K\times N}$ e um vetor $\texttt{ytr} \in \{-1,1\}^K$ em que $K = 3450$ e $N = 57$. A linha $\texttt{Xtr}[\texttt{k},:]$, ou equivalentemente $\texttt{Xtr}[\texttt{k}]$, contém informações coletadas para identificação da k-ésima mensagem (e-mail). A componente $\texttt{ytr}[\texttt{k}]$ contém o valor $+1$ se a k-ésima mensagem foi identificada como *spam* e -1 se foi identificada como não-*spam*. O objetivo do projeto é construir um modelo capaz de classificar uma mensagem como *spam* ou não-*spam* utilizando somente o conjunto de treinamento, ou seja, `Xtr` e `ytr`.

Funcionamento e Treinamento de uma Extreme Learning Machine

Uma *extreme learning machine* (ELM) é uma rede neural artificial de múltiplas camadas [1]. Nesse projeto, vamos considerar uma rede neural muito utilizada na literatura conhecida por *perceptron de múltiplas camadas*. Resumidamente, vamos assumir que a rede neural define uma função $\varphi : \mathbb{R}^N \to \mathbb{R}$ através da equação

$$\varphi(\mathbf{x}) = \alpha_1 g_1(\mathbf{x}) + \alpha_2 g_2(\mathbf{x}) + \ldots + \alpha_M g_M(\mathbf{x}) = \sum_{i=1}^{M} \alpha_i g_i(\mathbf{x}), \tag{1}$$

em que $\alpha_1, \alpha_2, \ldots, \alpha_M$ são parâmetros e as funções $g_1, g_2, \ldots, g_M$ são dadas por

$$g_i(\mathbf{x}) = \tanh\left(\sum_{j=1}^{N} w_{ij}x_j + \theta_i\right), \tag{2}$$

em que $\mathbf{w}_i = [w_{i1}, \ldots, w_{iN}] \in \mathbb{R}^N$ e $\theta_i \in \mathbb{R}$ para todo $i = 1, \ldots, M$[1]. Em termos matriciais, podemos descrever a função $\varphi : \mathbb{R}^N \to \mathbb{R}$ como segue:

$$\varphi(\mathbf{x}) = \tanh(\mathbf{x}\mathbf{W} + \boldsymbol{\theta}) \cdot \boldsymbol{\alpha}, \tag{3}$$

em que $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_M] \in \mathbb{R}^M$, $\mathbf{W} \in \mathbb{R}^{N\times M}$ é a matriz cujas linhas correspondem aos vetores $\mathbf{w}_i$ e $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_M] \in \mathbb{R}^M$. O código que implementa a função φ descrita pela rede neural artificial em `python` é:

```
def EvalELM(alpha,W,theta,X):
    return np.dot(np.tanh(X@W+theta),alpha)
```

Observação. *Note que o comando*

```
G = tanh(Xtr@W+theta)
```

fornece uma matriz $G \in \mathbb{R}^{K\times M}$ cujo elemento $G[k,i]$ corresponde à avaliação da i-ésima função base g_i calculada nos dados no vetor de características da k-ésima mensagem, isto é, `G[k,i]` $= g_i(\texttt{Xtr}[\texttt{k}])$. Além disso, o produto $\mathbf{s} = \texttt{G}\boldsymbol{\alpha}$ fornece um vetor $\mathbf{s} \in \mathbb{R}^K$ contendo o valor de φ calculado em cada mensagem, ou seja, $\mathbf{s} = [s_1, \ldots, s_K]$ em que $s_k = \varphi(\texttt{Xtr}[k])$ para todo $k = 1, \ldots, K$.

Numa ELM, os vetores $\mathbf{w}_i = [w_{i1}, \ldots, w_{iN}] \in \mathbb{R}^N$ e o escalar $\theta_i \in \mathbb{R}$ que definem a função g_i são gerados aleatoriamente utilizando uma distribuição normal padrão, para todo $i = 1, \ldots, M$. Usando a forma matricial explícita em (3), no `python`, utilizamos os seguintes comandos para construir $\mathbf{W}$ e $\boldsymbol{\theta}$:

```
W = np.random.randn(N,M)
theta = np.random.randn(M)
```

Os parâmetros $\alpha_1, \ldots, \alpha_M$ são determinados resolvendo o problema de quadrados mínimos

$$\varphi\big(\texttt{Xtr}[k]\big) \approx \texttt{ytr}[k], \quad \forall k = 1, \ldots, K, \tag{4}$$

[1]Observe que K refere-se ao número de dados de treinamento enquanto que M corresponde ao número de parâmetros. Nesse projeto, temos $M = 1000$ e $K = 3450$ no conjunto de treinamento.

definido sobre o conjunto de treinamento. Em outras palavras, $\alpha_1, \ldots, \alpha_M$ minimizam a soma dos quadrados dos desvios

$$J(\alpha_1, \ldots, \alpha_n) = \sum_{k=1}^{K} \big(\alpha_1 g_1(\texttt{Xtr}[k]) + \ldots + \alpha_M g_M(\texttt{Xtr}[k]) - \texttt{ytr}[k]\big)^2. \tag{5}$$

Finalmente, se $\mathbf{x} \in \mathbb{R}^N$ é o vetor contendo informações sobre uma mensagem, a identificação é efetuada como segue

$$\begin{cases} \text{A mensagem é um } \textit{spam} \text{ se } 0 \leq \varphi(\mathbf{x}), \\ \text{A mensagem não é } \textit{spam} \text{ caso contrário.} \end{cases} \tag{6}$$

Classificação de Novas Mensagens e Medidas de Desempenho

Conhecidos a função φ e o limiar L, podemos avaliar o desempenho do sistema usando um conjunto de dados que já foram identificados pelo(s) usuário(s). Por exemplo, podemos avaliar o desempenho do sistema no conjunto de teste, em que $\texttt{Xte} \in \mathbb{R}^{1151\times 57}$ e um vetor $\texttt{yte} \in \{-1, 1\}^{1151}$. Tal como nos dados de treinamento, `Xte`[k] e `yte`[k] contém informações sobre o conteúdo da k-ésima mensagem (e-mail) e sua classificação como *spam* ou não-*spam*, respectivamente. O desempenho do sistema pode ser medido quantitativamente, por exemplo, calculando a *acurácia* (AC) ou a *taxa de falsos positivos* (TFP, também chamado "taxa de alerta falso") definidos respectivamente pelas equações:

$$\text{AC} = \frac{\text{Número de mensagens identificadas corretamente pelo sistema}}{\text{Número total de mensagens}}, \tag{7}$$

e

$$\text{TFP} = \frac{\text{Número de mensagens identificadas como } \textit{spam} \text{ pelo sistema mas que não são } \textit{spams}}{\text{Número de mensagens que } \textbf{não} \text{ são } \textit{spams}}. \tag{8}$$

Questão 1

Sintetize a aplicação φ resolvendo o problema de quadrados mínimos em (4) com respeito ao conjunto de treinamento considerando $M = 1000$.

Questão 2

Ainda usando o conjunto de treinamento, isto é, `Xtr` e `ytr`, determine a acurácia e a taxa de falsos positivos.

Questão 3

Usando o conjunto de teste, isto é, `Xte` e `yte`, calcule a acurácia e a taxa de falsos positivos. O desempenho no conjunto de teste é consistente com o esperado, isto é, a acurácia e a taxa de falsos positivos são semelhantes aos valores obtidos considerando o conjunto de treinamento?

Referências

[1] HAYKIN, S. *Neural Networks and Learning Machines*, 3rd edition ed. Prentice-Hall, Upper Saddle River, NJ, 2009.

[2] HUANG, G.-B., WANG, D., AND LAN, Y. Extreme learning machines: a survey. *Int. J. Machine Learning & Cybernetics 2*, 2 (2011), 107–122.