Relatório Exercício 1

Eduardo Santos Carlos de Souza 9293481

Para esse exercício, inicialmente o dataset é criado automaticamente cada vez que o código é executado. Existem 2 modelos, um idêntico ao que estava no PDF e o outro o invertido dele. São feitas 64 cópias desses modelos, a classe escohida aleatoriamente, e inserido um ruído. O ruído consiste da troca do sinal de 0 a 5 posições nos dados. Desse forma ainda será mantida a aparência de um Y, mas com um ruído aleatório que dá robustez ao modelo treinado. Também, a matriz é achatada em um vetor para simplificar o código. Isso é feito com o seguinte código:

```
y = np.asarray([[1, -1, -1, -1, 1], [1, -1, -1, -1, 1], [-1, 1, 1, 1, -1], [-1, -1, 1, -1, -1], [-1, -1, 1, -1, -1]])
y_inv = np.asarray([[-1, -1, 1, -1, -1], [-1, -1, 1, -1, -1], [-1, 1, 1, 1, -1], [1, -1, -1, -1, 1], [1, -1, -1, -1, 1]])

n_samples = 64
dataset_x = np.empty((n_samples, 25))
dataset_y = np.empty(n_samples)
for i in range(n_samples):
    if (np.random.randint(2)):
        cur = np.ndarray.flatten(y)
        dataset_y[i] = 1
    else:
        cur = np.ndarray.flatten(y_inv)
        dataset_y[i] = -1
    rand_flip = np.random.randint(25, size=(np.random.randint(6),))
    cur[rand_flip] *= -1
    dataset_x[i] = cur
```

O Dataset é quebrado em 2 conjuntos, treino e teste, com 80% treino e 20% teste. Isso é feito com o seguinte código:

```
train_per = 0.8
split = int(train_per*dataset_x.shape[0])
dataset_x_train = dataset_x[:split]
dataset_x_test = dataset_x[split:]
dataset_y_train = dataset_y[:split]
dataset_y_test = dataset_y[split:]
```

Em seguida é inicializado um único neurônio aleatóriamente, com pesos seguindo uma distribuição uniforme entre -0.5 e 0.5. Esse neurônio é treinado no conjunto de treino seguindo o algorítimo da “Regra Delta - LMS” mostrado em sala. O treino para depois de um numero máximo de iterações ter sido atingido, ou do erro estar abaixo de um limite. Isso é feito com o seguinte código:

```
neuron = np.random.random_sample(26)-0.5
neuron = fit(dataset_x_train, dataset_y_train, neuron)
```

Depois do treino é feita a validação no conjunto de teste. Isso é feito com o seguinte código:

```
acc = 0
for i in range(dataset_x_test.shape[0]):
    out = forward(dataset_x_test[i], neuron)
    if(out == dataset_y_test[i]):
        acc += 1
```

Em geral são necessárias apenas 2 iterações para erro 0 no conjunto de treino e 100% de acerto no de teste.

As funções de treino e execução codificadas foram:

```
def append_one(x):
    aux = np.ones(x.shape[0]+1)
    aux[0:x.shape[0]] = x

    return aux

def thresh(x):
    if (x > 0):
        return 1
    else:
        return -1
def forward(neuron_input, neuron):
    aux_input = append_one(neuron_input)
    out = thresh(np.sum(aux_input * neuron))

    return out

def fit(dataset_x, dataset_y, neuron, thresh=0, eta=0.1, max_it=10000):
    count = 0
    error = thresh+1
    while(error > thresh and count < max_it):
        error = 0
        for i in np.arange(dataset_x.shape[0]):
            out = forward(dataset_x[i], neuron)
            cur_error = dataset_y[i] - out
            if (cur_error != 0):
                error += np.abs(cur_error)
                neuron = neuron + eta*append_one(dataset_x[i])*cur_error

        error /= dataset_x.shape[0]
        print("Erro médio iteração " + str(count) + ": " + str(error))
        count += 1

    return neuron
```