CEFET/RJ
Programa de Pós-graduação
em Ciência da Computação
Aprendizado de Máquina - Trabalho 03

Prof. Eduardo Bezerra (ebezerra@cefet-rj.br)

Novembro/2017

Conteúdo

1 Redes Completamente Conectadas

Uma instituição financeira possui uma base de dados com o histórico de crediário oferecido aos seus clientes. Baseado neste histórico, a instituição deseja criar um modelo de classificação para inferir se um novo cliente que submeteu uma requisição de empréstimo pagará ou não a dívida, caso o banco resolva realizar esse empréstimo. O conjunto de dados possui 1500 exemplos de créditos concedidos aos seus clientes. Esses registros estão contidos no arquivo credtrain.txt, que é fornecido juntamente com esse documento. Para cada cliente, são definidos 11 atributos (variáveis, características). Além disso, a última coluna de cada exemplo informa se o cliente honrou ou não o pagamento do empréstimo. Na Tabela 1, encontramos a descrição dos atributos.

Tabela 1: Esquema do conjunto de dados com histórico de clientes.

Variável	**Descrição**	**Tipo**	**Domínio**
ESCT	Estado civil	Categórica	0,1,2,3
NDEP	Número de dependentes	Categórica	0,1,2,3,4,5,6,7
RENDA	Renda Familiar	Numérica	300-9675
TIPOR	Tipo de residência	Categórica	0,1
VBEM	Valor do bem a ser adquirido	Numérica	300-6000
NPARC	Número de parcelas	Numérica	1-24
VPARC	Valor da parcela	Numérica	50-719
TEL	Se o cliente possui telefone	Categórica	0,1
IDADE	Idade do cliente	Numérica	18-70
RESMS	Tempo de moradia (em meses)	Numérica	0-420
ENTRADA	Valor da entrada	Numérica	0-1300
CLASSE	=1 se o cliente pagou a dívida	Categórica	0,1

Nesta parte, você deve criar um modelo de classificação, por meio de uma rede neural de múltiplas camadas com propagação do erro (error backpropagation). O objetivo desse modelo de classificação é predizer se um novo cliente pagaria ou não uma dívida contraída, tendo como base as características desse novo cliente. Esse modelo deve ser criado com o uso de funções e classes fornecidas pelo `Keras`.

1.1 Um exemplo introdutório

Para servir de exemplo sobre o procedimento de criação de um modelo de classificação, é relevante que você realize no Matlab os passos descritos nesta seção. O objetivo é que, ao fim desta seção, você tenha uma visão geral do processo de criação de um modelo de classificação em um exemplo pequeno e possa aplicá-lo a um exemplo maior (i.e., o conjunto de dados de crédito bancário). A seqüência de passos descrita nesta seção representa a criação, treinamento (aprendizado) e teste de uma rede neural backpropagation com um conjunto de dados pequeno armazenado nas matrizes P e T. Após realizar esses passos, você deve aplicar o mesmo procedimento sobre o conjunto de dados

de crédito (credtrain) fornecido neste trabalho, para finalmente obter o modelo de classificação.

1. Defina a matriz de padrões. Note que os padrões são organizados em colunas. Por exemplo, a matriz de padrões abaixo contém cinco padrões de entrada (cada um com quatro características), e o primeiro padrão armazenado na matriz P abaixo é (0.4046, 0.9974, 0.3764, 0.6043).

$$P = \begin{bmatrix} 0.4046 & 0.3786 & 0.7010 & 0.8608 & 0.5947 \\ 0.9974 & 0.8479 & 0.6201 & 0.4031 & 0.9653 \\ 0.3764 & 0.9214 & 0.9331 & 0.7514 & 0.6914 \\ 0.6043 & 0.3494 & 0.1438 & 0.6035 & 0.4111 \end{bmatrix}$$

2. Defina a matriz de alvos. Os valores dessa matriz de alvos são os que a rede deve aprender para cada padrão. Note que a quantidade de colunas na matriz de alvos deve ser sempre igual à quantidade de colunas da matriz de padrões. $T = [0\ 1\ 0\ 1\ 1]$

3. Crie uma rede neural artificial completamente conectada de uma única camada oculta, com 3 neurônios nessa camada oculta e 1 neurônio na camada de saída. Em ambas as camadas, a função de ativação deve ser a função sigmoide logística.

4. Realize a configuração da rede recém-criada, conforme instruções a seguir. Quantidade de épocas igual a 100000; Função de perda: soma dos erros quadrados; método de otimização: gradiente descendente estocástico.

5. Treine a rede neural. Esse é o passo em que os pesos e os limiares da rede são ajustados. Esse passo deve ser realizado com a função `fit` da classe `Model`. Para chamar essa função, devemos passar a rede, a matriz de padrões e a matriz de alvos.

6. Gere gráficos para visualizar o histórico de treinamento. Em particular, gere curvas da precisão e da função de custo contRa a quantidade de épocas de treinamento. Em `https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/`, você encontra um exemplo de como gerar esses gráficos usando o `Keras`.

1.2 Criação de modelo de predição de crédito

Após realizar os passos descritos acima, você deve ter condições de repeti-los para o conjunto de dados de créditos (`credtrain`). Uma vez treinada, essa rede poderá inferir se um novo cliente irá ou não honrar um eventual empréstimo.

Inicialmente, você deve carregar o arquivo `credtrain.txt`. Uma vez que os dados estão carregados, você deve criar duas matrizes. A primeira matriz deve ter ordem 1500×11 e deve conter os padrões de entrada (valores das variáveis ESTC, NDEP, RENDA, TIPOR, VBEM, NPARC, VPARC, TEL, IDADE, RESMS e ENTRADA). A outra matriz deve ter ordem 1500×1 e deve conter os valores do atributo CLASSE. Vamos chamar essas duas matrizes de

`P` e `T`, respectivamente. A matriz `P` é chamada a matriz de treinamento. Já a matriz `T` é chamada a matriz de alvo. Diferente do exemplo introdutório, um cuidado adicional que você deve ter é com relação à normalização dos dados de entrada da rede, posto que algumas variáveis possuem ordens de grandeza diferentes.

Uma vez construídas as matrizes P e T, é possível criar, treinar e testar uma RNA de múltiplas camadas feed-forward com o uso do Keras. Essa rede neural corresponde ao modelo de classificação a ser desenvolvido neste trabalho.

Um detalhe importante é que no passo (6), em vez de testar a rede sobre os padrões utilizados durante o treinamento, você deve testá-la sobre os dados contidos no arquivo `credtest.txt`, que também é fornecido. Isso permitirá que você avalie o quão efetivo foi o passo de treinamento da rede neural, ou seja, o quão adequado é o modelo de classificação. Outro detalhe importante é que, no passo (3), você deve experimentar e utilizar diferentes quantidades de neurônios na camada intermediária para gerar uma boa taxa de acerto.

Crie uma função para produzir a *matriz de confusão* (do termo original *confusion matrix*; veja maiores detalhes em `http://en.wikipedia.org/wiki/Confusion_matrix`) relativa aos resultados da fase de testes. Uma das entradas dessa matriz é o valor da taxa de acerto da rede durante a fase de teste, ou seja, a porcentagem de registros do conjunto de padrões de teste (`credtest`) para os quais a rede acertou a resposta correta. Em `http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html`, você encontra um exemplo de como gerar uma matriz de confusão.

2 Redes Convolucionais

Nesta parte, você irá treinar modelos de redes neurais artificiais para classificar imagens. Em particular, dada uma imagem, seus modelos deverão indicar se a imagem contém um gato ou não. São fornecidos os conjuntos de imagens para treinamento e para teste (arquivos `train_catvnoncat.h5` e `test_catvnoncat.h5`). É também fornecido o código para a leitura dos arquivos correspondentes (veja listagem a seguir; veja também a seção de créditos no fim deste documento).

```
1  import numpy as np
2  import h5py
3
4  def load_dataset():
5      train_dataset = h5py.File('./train_catvnoncat.h5', "r")
6      train_set_x_orig = np.array(train_dataset["train_set_x"][:]) #
       your train set features
7      train_set_y_orig = np.array(train_dataset["train_set_y"][:]) #
       your train set labels
8
9      test_dataset = h5py.File('./test_catvnoncat.h5', "r")
10     test_set_x_orig = np.array(test_dataset["test_set_x"][:]) #
       your test set features
11     test_set_y_orig = np.array(test_dataset["test_set_y"][:]) #
       your test set labels
12
```

```
    classes = np.array(test_dataset["list_classes"][:]) # the list
    of classes

    train_set_y_orig = train_set_y_orig.reshape((1,
    train_set_y_orig.shape[0]))
    test_set_y_orig = test_set_y_orig.reshape((1, test_set_y_orig.
    shape[0]))

    return train_set_x_orig, train_set_y_orig, test_set_x_orig,
    test_set_y_orig, classes
```

Após ler os conjuntos de dados, você deve treinar dois modelos utilizando o `Keras`, conforme descrito a seguir:

- Uma rede completamente conectada de uma única camada oculta e com uma camada de saída de duas unidades com softmax.
- Uma rede convolucional.

Em ambos os casos acima, você deverá selecionar os hiperparâmetros e arquitetura de rede. Procure se basear nos exemplos de código e nas arquiteturas de rede apresentadas em aula. Em seu relatório, apresente os detalhes acerca da definição de cada uma dessas redes, assim como o desempenho encontrado em cada um dos casos.

3 Entrega

Você deve preparar um único relatório para a apresentar sua análise e conclusões sobre as diversas partes desse trabalho. O formato desse relatório deve ser em PDF. Alternativamente à entrega do relatório em PDF, você pode entregar um notebook Jupyter[1].

Independente de escolher entregar um relatório em PDF ou na forma de um notebook Jupyter, entregue também todos os arquivos em Python que você criou para cada parte deste trabalho. Todos os arquivos em Python devem estar em uma única pasta.

Crie um arquivo compactado que contém o relatório (ou notebook Jupyter) e os arquivos (*scripts*) em Python. Esse arquivo compactado deve se chamar `SEU_NOME_COMPLETO_T3.zip`. Esse arquivo compactado deve ser entregue pelo Moodle, até a data acordada.

4 Créditos

A segunda parte desse trabalho usa conjuntos de dados e código disponibilizados no curso *Neural Networks and Deep Learning*[2] encontrado no Coursera. O material original é de autoria do prof. Andrew Ng e de seus colaboradores.

[1] `http://jupyter.org/`
[2] `https://www.coursera.org/learn/neural-networks-deep-learning/home/welcome`