Prediction of malignant/benign of breast mass with DNN classifier

Sophie Tan
Maplesoft

Introduction

According to Canadian Cancer Society, in 2017, about 72 Canadian women were diagnosed with breast cancer everyday. Despite of its high incidence, the mortality rate of breast cancer has in fact been declining since the mid-1980s. This improvement in statistics is attributed to the impact of advancing screening techniques and the resulting increasing cases of early diagnosis.

The application of deep learning in oncology has always been a study of focus due to its potential impact on the human society. In the field of assisted cancer diagnosis, the involvement of machine learning in disease diagnosis is expected to give physicians a second opinion and help them make quicker/better diagnosis.

Google's DNN system has in fact achieved a dermatologist level of accuracy at identifying deadline skin cancers, suggesting that accessibility of cancer diagnosis can potentially be extended beyond medical clinics.

This application employs the DeepLearning package to train a DNN classifier with one of the Breast Cancer Wisconsin (Diagnostic) Data Sets and uses the classifier to predict the diagnosis of a breast mass with 30 real, numeric input values that characterize cell nucleus properties of the breast mass. The major focus in this field is to apply natural image classification techniques and perform the classification/prediction directly from the digital image. However, for the purposes of demonstrating the basic techniques, this application performs breast cancer classification with numeric data computed from digitalized image of a fine needle aspirate (FNA) of a breast mass.

Overview of the data set:

- The first column (of the dataset) is the ID code of patients, which is not considered in the training process.

- The second column is the Diagnosis, where 1 indicates Malignant and 0 indicates Benign.
- The rest of the columns are 30 real numeric values that indicates physical measurements of the cell nucleus. For each cell nucleus, ten properties including radius, texture, perimeter, area, smoothness, compactness, concavity, symmetry and fractal dimension are measured. The mean, standard error and largest values of those ten properties are calculated, thus resulting in 10 × 3, 30 input data columns.

References*:* Original Dataset accessible at wdbc.data (UCI Machine Learning Repository)

Source information and complete documentation about the dataset can be found at wdbc.names

Import Data

```
> restart;
> train_data := Import("this:///wdbc_train.csv"):
> test_data:= Import("this:///wdbc_test.csv"):
> cols := ["ID number", "Diagonosis", "radius", "texture", "perimeter", "area", "smoothness",
  "compactness", "concavity", "concave_points", "symmetry", "fractal_dimension",
  "radius_SE", "texture_SE", "perimeter_SE", "area_SE", "smoothness_SE", "compactness_SE" ,
  "concavity_SE", "concave_points_SE", "symmetry_SE", "fractal_dimension_SE",
  "worst_radius", "worst_texture", "worst_perimeter", "worst_area", "worst_smoothness",
  "worst_compactness", "worst_concavity", "worst_concave_points", "worst_symmetry",
  "worst_fractal_dimension"]:
> train_data:=DataFrame(train_data, columns=cols);
```

```
train_data := [[, "ID number", "Diagonosis", "radius", "texture", "perimeter", "area",          (2.1)
    "smoothness", "compactness", ...],
    [1, 842302., 1., 17.9900000000000, 10.3800000000000, 122.800000000000, 1001.,
    0.118400000000000, 0.277600000000000, ...],
    [2, 842517., 1., 20.5700000000000, 17.7700000000000, 132.900000000000, 1326.,
    0.0847000000000000, 0.0786000000000000, ...],
    [3, 8.4300903 10^7, 1., 19.6900000000000, 21.2500000000000, 130., 1203.,
    0.109600000000000, 0.159900000000000, ...],
    [4, 8.4348301 10^7, 1., 11.4200000000000, 20.3800000000000, 77.5800000000000,
    386.100000000000, 0.142500000000000, 0.283900000000000, ...],
    [5, 8.4358402 10^7, 1., 20.2900000000000, 14.3400000000000, 135.100000000000,
    1297., 0.100300000000000, 0.132800000000000, ... ],
    [6, 843786., 1., 12.4500000000000, 15.7000000000000, 82.5700000000000,
    477.100000000000, 0.127800000000000, 0.170000000000000, ... ],
    [7, 844359., 1., 18.2500000000000, 19.9800000000000, 119.600000000000, 1040.,
    0.0946000000000000, 0.109000000000000, ... ],
    [8, 8.4458202 10^7, 1., 13.7100000000000, 20.8300000000000, 90.2000000000000,
    577.900000000000, 0.118900000000000, 0.164500000000000, ... ],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ...]]
```

```
> test_data:=DataFrame(test_data, columns=cols);
```

```
test_data := [[, "ID number", "Diagonosis", "radius", "texture", "perimeter", "area",           (2.2)
    "smoothness", "compactness", ... ],
    [1, 9.112085 10^6, 0., 13.3800000000000, 30.7200000000000, 86.3400000000000,
    557.200000000000, 0.0925000000000000, 0.0743000000000000, ... ],
    [2, 9.112366 10^6, 0., 11.6300000000000, 29.2900000000000, 74.8700000000000,
    415.100000000000, 0.0936000000000000, 0.0857000000000000, ... ],
    [3, 9.112367 10^6, 0., 13.2100000000000, 25.2500000000000, 84.1000000000000,
    537.900000000000, 0.0879000000000000, 0.0521000000000000, ... ],
    [4, 9.112594 10^6, 0., 13., 25.1300000000000, 82.6100000000000, 520.200000000000,
    0.0837000000000000, 0.0507000000000000, ... ],
    [5, 9.112712 10^6, 0., 9.75500000000000, 28.2000000000000, 61.6800000000000,
    290.900000000000, 0.0798000000000000, 0.0463000000000000, ... ],
    [6, 9.11296201 10^8, 1., 17.0800000000000, 27.1500000000000, 111.200000000000,
    930.900000000000, 0.0990000000000000, 0.111000000000000, ... ],
    [7, 9.11296202 10^8, 1., 27.4200000000000, 26.2700000000000, 186.900000000000,
    2501., 0.108400000000000, 0.198800000000000, ... ],
    [8, 9.113156 10^6, 0., 14.4000000000000, 26.9900000000000, 92.2500000000000,
    646.100000000000, 0.0700000000000000, 0.0522000000000000, ... ],
    [..., ..., ..., ..., ..., ..., ..., ..., ..., ...]]
```

The dataset has 569 instances, 455 are used for training and 114 are used for validation/testing (ratio 8:2).

Training and evaluating the Deep Neural Network Classifier

```
> with(DeepLearning):
> fc := [seq(NumericColumn(u, shape=[1]), u in cols[3..])]:
```

We set three hidden layers in our model with 20, 40, 20 nodes on each layer respectively, num_classes is the number of resulting classes we want in the final classification step, which is two in this case (i.e. benign or malignant).

```
> classifier := DNNClassifier(fc, hidden_units = [20, 40, 20], num_classes = 2):
> classifier:-Train(train_data[3..32], train_data[2], steps =256, num_epochs = 3, shuffle = true):
```

We specify that the entire dataset would be passed to the model for three times and each iteration is consist of 256 steps. In addition, data batches for training will be created by randomly shuffling the tensors. (please refer to hyperparamter optimization for details, remember this application is for demonstration purpose)
NOTE that, the initial weights of the neural network are selected randomly, which means that each time we run the train command (with a fixed input data), we get a slightly different model, thus producing different accuracy and prediction results below.

Now the training process is complete, we can use the validation set to test the effectiveness of our model, as we can see the accuracy is ~ 92.14% in this case. Evaluation indices like accuracy baseline, average loss can also help us decide if we want to further adjust the architecture of the network model. Note that in general, we want to avoid tweaking the model architecture too much to prevent overfitting.

```
> classifier:-Evaluate(test_data[3..32],test_data[2], steps = 32);
```

$$\left[\begin{array}{cc} \text{"accuracy"} & 0.921386718750000 \\ \text{"accuracy\_baseline"} & 0.772949218750000 \\ \text{"auc"} & 0.883204281330109 \\ \text{"auc\_precision\_recall"} & 0.860257983207703 \\ \text{"average\_loss"} & 0.290851414203644 \\ \text{"label/mean"} & 0.227050781250000 \\ \text{"loss"} & 37.2289810180664 \\ \dots & \dots \end{array}\right] \quad (3.1)$$

Build Predictor Function

```
> predictor := proc (ds) classifier:-Predict(Transpose(DataFrame(ds)), num_epochs = 1, shuffle =
  false)[1] end proc;
predictor := proc(ds)
    classifier:-Predict(Transpose(DataFrame(ds)), num_epochs = 1, shuffle = false)[1]
end proc                                                                                  (4.1)
```

Above we have built a predictor function that takes an arbitrary set of measurements as a DataSeries and returns a prediction generated by the trained DNN classifier. Now we can pass a the predictor a DataSeries containing physical traits that characterize a single cell nucleus from a breast mass sample, and use it to predict whether the breast mass is benign or malignant.

```
> ds := DataSeries([11.49, 14.59, 73.99, 404.9, 0.1046, 8.23E-02, 5.31E-02, 1.97E-02, 0.1779,
  6.57E-02, 0.2034, 1.166, 1.567, 14.34, 4.96E-03, 2.11E-02, 4.16E-02, 8.04E-03, 1.84E-02,
  3.61E-03, 12.4, 21.9, 82.04, 467.6, 0.1352, 0.201, 0.2596, 7.43E-02, 0.2941, 9.18E-02], labels =
  cols[3..]);
```

$$ds := \begin{bmatrix} \text{"radius"} & 11.49 \\ \text{"texture"} & 14.59 \\ \text{"perimeter"} & 73.99 \\ \text{"area"} & 404.9 \\ \text{"smoothness"} & 0.1046 \\ \text{"compactness"} & 0.0823 \\ \text{"concavity"} & 0.0531 \\ \dots & \dots \end{bmatrix} \qquad \textbf{(4.2)}$$

```
> predictor(ds);
```

$$\begin{bmatrix} \text{"logits"} & \begin{bmatrix} -2.28828620910645 \end{bmatrix} \\ \text{"logistic"} & \begin{bmatrix} 0.0920977517962456 \end{bmatrix} \\ \text{"probabilities"} & \begin{bmatrix} 0.907902240753174 \\ 0.0920977517962456 \end{bmatrix} \\ \text{"class\_ids"} & \begin{bmatrix} 0 \end{bmatrix} \\ \text{"classes"} & \begin{bmatrix} \text{"<Python object: b'0'>"} \end{bmatrix} \end{bmatrix} \qquad \textbf{(4.3)}$$

The output indicates that the probability of this breast mass cell being benign (class_id[0]) is ~90.79% according to our predictive model.