APPLICATION OF NEURAL NETWORKS

CS 386 - Artificial Intelligence Lab

Problems we tried to solve

1. Prediction of Mortality rate of patients in ICU using neural networks
2. Predicting whether a given 10 letter word is a valid English word

Mortality rate prediction using Neural Networks

Introduction

An intensive care unit (ICU) is for patients with the most serious diseases or injuries. Most of the patients need support from equipment like the medical ventilator to maintain normal body functions and need to be constantly and closely monitored. For decades, the number of ICUs has experienced a worldwide increase. During the ICU stay, different physiological parameters are measured and analysed each day. Those parameters are used in scoring systems to gauge the severity of the patients. These parameters help us predict the mortality of the particular patient. This has become a necessity these days as ICUs need more care and if we can restrict them to the most needed we can maximise our efficiency and help.

Motivation

One of the main reasons why we decided to use neural networks is because of their huge tolerance to noise. In the given data set we have many parameters whose values are not measured at all time stamps. Another fascinating fact that led us to use neural networks is because they learn and generalize from training data and considering that none of us have any medical background the choice seems apt. We tried to achieve the objective of prediction using Stochastic Gradient Descent(SGD) Algorithm.

Problem Statement

1) The aim of this project is to predict risk of death (mortality) in patients admitted to intensive care units (ICU) within hospitals
2) We have the simulated data of around 6000 patients where each patient record has 35 attributes related to him and one being the timestamp of observation
3) A **timestamp** shows when a **vital or lab measurement** is made relative to the first measurement.
4) There can be one or more measurements at each timestamp. The first timestamp for each patient is always 0 (zero)
5) The length-of-stay of each patient is different and so the number of measurements (hence timestamps) are different in each patient record

Attributes

1. ID: a unique identifier for each patient
2. Age
3. 6 Vitals measurements
4. 25 Labs measurements
5. Timestamps: measurement time relative to first measurement for patient
6. ICU flag: indicates whether a patient is in ICU or not at a given time
7. Mortality label: indicates whether a patient survived or died

Neural Networks

1) Artificial neural networks are generally presented as systems of interconnected "neurons" which exchange messages between each other.
2) The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.
3) *Perceptrons* are the mathematical model of a neuron
4) In our project we make use of two types of perceptron apart from input nodes
    a) Sigmoid neurons
    b) Classification or Step neuron's (output 0/1)
5) The hidden layer has sigmoid neurons and the output layer makes use of step neurons.

Stochastic Gradient Descent (SGD)

1) We use learning rate, momentum of SGD and backpropagation in order to do the learning using the stated variables
2) SGD mainly works on the principle of randomly choosing a mini batch from the entire data and use those attributes on the entire data set to do the learning. This repeats till all the elements are chosen into some or the other mini batch.
3) This stochastic approach along with the idea of simulation of a rolling ball helps us do the learning in an intelligent fashion and thus we obtain the aimed result
4) After algorithm is executed completely we shall has the learned neural network which can be tested for accuracy on a test data and thus we can assure our approach using statistical comparison

Approach

1) As discussed above we decided to model a neural network for the given problem and attributes
2) From the given attributes we make wise choice of variables and try to implement neural network on it
3) There might be cases when some attribute is not defined we make that attribute equal to some functional output of the previous readings of the same attribute of the same patient
4) Hence we end up with a set of well defined variables which we intend to use in the modelling of neural network

Approach

Whenever some attribute is not defined we fill it with the average of the previous values of the same attribute of the patient else we make it -1

We make up new variables using the previous data we calculate the online standard deviation and slope of the best fit linear regression function possible for every corresponding attribute and make them as new variables. Our main aim is to picturize the timeline of patients' variables for which we use linear regression to plot the graph and standard deviation to get the instability of the patients.

We have 66 new variables, 33 corresponding to standard deviation and other 33 corresponding to online linear regression

Neural Network and SGD application

We intend to use SGD as stated before on the modelled neural network. We shall use 1 hidden layer in Neural Network, 1 input layer and 1 output layer.

1) Input layer consists of all the valid variables which are to be an input
2) Output layer consists two neurons pertaining to states dead or alive
3) The correct number of neurons in the hidden layer are to be determined using scripting or trial and error, as of now we are using 100 neurons in the hidden layer
4) The learning rate and momentum in SGD also need to be fixed using the same techniques as stated above and right now the stand at 1e-4 and 09 respectively

Code Overview

1) The data set is taken and parsed along with certain processing to get through all the correct attributes for a given patient and timestamp
2) Missing data would be flagged and which will be later filled using the average of online data of that particular patient and his previous observations
3) Later we use the existent variables to get two different sets of data corresponding to the set of attributes
4) First set being the standard deviation pertaining till that extent
5) Second set being the linear regression slope till that extent
6) We add the all the variables' values at that timestamp, standard deviation till that time stamp and slope to a single matrix.

7) Build a neural network with 1 hidden layer(1000 neurons), 99 input nodes(33 variables, 33 standard deviations, and 33 regression slope) and output

8) Try to figure out number of neurons in hidden layer and the learning rates and momentum using trial and error and scripts and finally reached around 90% accuracy

Results and Conclusion

We started of by feeding the the inputs to the neural network in it’s raw form(without and modifications) but this resulted in a very poor accuracy(50-60%)

Later after going through the paper on what parameters and what configuration of the neural network was used, we decided to use linear regression to figure out the missing inputs, next we took the standard deviation of some of the parameters to account for fluctuations in data which could provide insight into how the patient is holding up. Finally we fed these modified parameters to neural network with three hidden layers achieving an accuracy close to (90%).

Deliverables

1) Python scripts with appropriate documentation and guidelines for execution and testing
2) Output is of the accuracy after running on validation data

Requirements

Programming Language : Python.

Packages used : NumPy, Theanets, scikit-learn, climate.

All the data sets (training, validation and test) are obtained from Xerox Mortality Prediction
Challenge(https://www.hackerrank.com/contests/xerox-research-innovation-challenge-2015/challenges/xerox-predict-mortality)

The project was inspired from the above mentioned Challenge.

References :

1) http://www.cinc.org/archives/2012/pdf/0261.pdf
2) http://neuralnetworksanddeeplearning.com/

10 Letter Word Prediction

Code Overview - Word Prediction

1. We used the standard dictionary present in linux as the input data. Dividing it into validation and training data
2. We used the words in the american-english and british-english dictionaries as test data
3. We generate random 10 letter words to serve as examples of non valid english words to the neural network
4. We convert each word into its corresponding ten ASCII key code
5. We feed these 10 inputs to the neural network as the features along with whether or not it’s a word from the dictionary
6. This gives rise to a neural network with 10 input and 2 outputs. On testing we found out that 3 hidden layers with 500 nodes each gives a strong accuracy

Results And Conclusions

We trained the neural network on the standard dictionary words in linux.

We tested the net on the american-english and british english words. The net achieved an accuracy of around 95%.

We further tested the net on obscure english words(from internet). It could classify the words with an accuracy of 80%.

Results And Conclusions-Mortality Prediction

```
I 2015-11-23 12:54:42 downhill.base:232 validation 1 loss=2.078557 err=2.078557 acc=0.906520 *
I 2015-11-23 12:55:41 downhill.base:232 SGD 111 loss=2.162772 err=2.162772 acc=0.902223
I 2015-11-23 12:56:42 downhill.base:232 SGD 112 loss=2.146130 err=2.146130 acc=0.903109
I 2015-11-23 12:57:35 downhill.base:232 SGD 113 loss=2.155427 err=2.155427 acc=0.902603
I 2015-11-23 12:58:27 downhill.base:232 SGD 114 loss=2.161045 err=2.161045 acc=0.902321
I 2015-11-23 12:59:18 downhill.base:232 SGD 115 loss=2.168538 err=2.168538 acc=0.901931
I 2015-11-23 13:00:11 downhill.base:232 SGD 116 loss=2.246571 err=2.246571 acc=0.877672
I 2015-11-23 13:01:04 downhill.base:232 SGD 117 loss=2.222639 err=2.222639 acc=0.878989
I 2015-11-23 13:01:56 downhill.base:232 SGD 118 loss=2.147943 err=2.147943 acc=0.903031
I 2015-11-23 13:02:49 downhill.base:232 SGD 119 loss=2.179957 err=2.179957 acc=0.901293
I 2015-11-23 13:03:40 downhill.base:232 SGD 120 loss=2.235863 err=2.235863 acc=0.878265
I 2015-11-23 13:03:47 downhill.base:232 validation 2 loss=2.078049 err=2.078049 acc=0.906870 *
I 2015-11-23 13:04:41 downhill.base:232 SGD 121 loss=2.254725 err=2.254725 acc=0.877228
I 2015-11-23 13:05:35 downhill.base:232 SGD 122 loss=2.158706 err=2.158706 acc=0.902466
I 2015-11-23 13:06:31 downhill.base:232 SGD 123 loss=2.149720 err=2.149720 acc=0.902924
I 2015-11-23 13:07:23 downhill.base:232 SGD 124 loss=2.223340 err=2.223340 acc=0.878965
I 2015-11-23 13:08:19 downhill.base:232 SGD 125 loss=2.161684 err=2.161684 acc=0.902296
I 2015-11-23 13:09:12 downhill.base:232 SGD 126 loss=2.277110 err=2.277110 acc=0.876023
I 2015-11-23 13:10:08 downhill.base:232 SGD 127 loss=2.163293 err=2.163293 acc=0.902227
I 2015-11-23 13:11:04 downhill.base:232 SGD 128 loss=2.145253 err=2.145253 acc=0.903182
I 2015-11-23 13:11:59 downhill.base:232 SGD 129 loss=2.145577 err=2.145577 acc=0.903163
I 2015-11-23 13:12:52 downhill.base:232 SGD 130 loss=2.145256 err=2.145256 acc=0.903200
I 2015-11-23 13:12:59 downhill.base:232 validation 3 loss=2.077912 err=2.077912 acc=0.906865 *
I 2015-11-23 13:13:55 downhill.base:232 SGD 131 loss=2.147957 err=2.147957 acc=0.903060
I 2015-11-23 13:14:51 downhill.base:232 SGD 132 loss=2.148467 err=2.148467 acc=0.903040
I 2015-11-23 13:15:48 downhill.base:232 SGD 133 loss=2.147395 err=2.147395 acc=0.903074
I 2015-11-23 13:16:44 downhill.base:232 SGD 134 loss=2.147131 err=2.147131 acc=0.903098
I 2015-11-23 13:17:40 downhill.base:232 SGD 135 loss=2.147080 err=2.147080 acc=0.903114
I 2015-11-23 13:18:35 downhill.base:232 SGD 136 loss=2.148203 err=2.148203 acc=0.903064
^CI 2015-11-23 13:18:40 downhill.base:419 interrupted!
(OrderedDict([('loss', 2.148202921067226), ('err', 2.148202921067226), ('acc', 0.9030643108101227
```

```
        0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.], dtype=float32)
>>> x = np.array(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'np' is not defined
>>> import numpy as np
>>> np.array(x)
array([  0.,  86.,  49.,  70.,  -1.,  87.,  -1.,   0.,  -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.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.], dtype=float32)
>>> x = np.array([x])
>>> x
array([[  0.,  86.,  49.,  70.,  -1.,  87.,  -1.,   0.,  -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.,
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.]], dtype=float32)
>>> net.predict(x)
array([0])
>>>
```

Results And Conclusions - Word Prediction

```
I 2015-11-23 12:52:58 downhill.base:232 validation 3 loss=0.255073 err=0.255073 acc=0.917215
I 2015-11-23 12:53:10 downhill.base:232 SGD 31 loss=0.246808 err=0.246808 acc=0.952795
I 2015-11-23 12:53:21 downhill.base:232 SGD 32 loss=0.246825 err=0.246825 acc=0.954158
I 2015-11-23 12:53:32 downhill.base:232 SGD 33 loss=0.246653 err=0.246653 acc=0.954413
I 2015-11-23 12:53:44 downhill.base:232 SGD 34 loss=0.246213 err=0.246213 acc=0.953987
I 2015-11-23 12:53:54 downhill.base:232 SGD 35 loss=0.246397 err=0.246397 acc=0.953732
I 2015-11-23 12:54:05 downhill.base:232 SGD 36 loss=0.246330 err=0.246330 acc=0.953732
I 2015-11-23 12:54:17 downhill.base:232 SGD 37 loss=0.245892 err=0.245892 acc=0.953306
I 2015-11-23 12:54:28 downhill.base:232 SGD 38 loss=0.245675 err=0.245675 acc=0.954158
I 2015-11-23 12:54:39 downhill.base:232 SGD 39 loss=0.246041 err=0.246041 acc=0.953817
I 2015-11-23 12:54:49 downhill.base:232 SGD 40 loss=0.245691 err=0.245691 acc=0.954328
I 2015-11-23 12:54:50 downhill.base:232 validation 4 loss=0.255333 err=0.255333 acc=0.919253
I 2015-11-23 12:55:00 downhill.base:232 SGD 41 loss=0.245604 err=0.245604 acc=0.953732
I 2015-11-23 12:55:10 downhill.base:232 SGD 42 loss=0.245466 err=0.245466 acc=0.954754
I 2015-11-23 12:55:21 downhill.base:232 SGD 43 loss=0.245408 err=0.245408 acc=0.954669
I 2015-11-23 12:55:31 downhill.base:232 SGD 44 loss=0.245448 err=0.245448 acc=0.953732
I 2015-11-23 12:55:41 downhill.base:232 SGD 45 loss=0.245334 err=0.245334 acc=0.953051
I 2015-11-23 12:55:52 downhill.base:232 SGD 46 loss=0.244976 err=0.244976 acc=0.954328
I 2015-11-23 12:56:02 downhill.base:232 SGD 47 loss=0.245046 err=0.245046 acc=0.954583
I 2015-11-23 12:56:12 downhill.base:232 SGD 48 loss=0.244990 err=0.244990 acc=0.954669
I 2015-11-23 12:56:22 downhill.base:232 SGD 49 loss=0.244905 err=0.244905 acc=0.956116
I 2015-11-23 12:56:33 downhill.base:232 SGD 50 loss=0.244662 err=0.244662 acc=0.955605
I 2015-11-23 12:56:33 downhill.base:232 validation 5 loss=0.255039 err=0.255039 acc=0.915516
I 2015-11-23 12:56:43 downhill.base:232 SGD 51 loss=0.244722 err=0.244722 acc=0.953221
^CI 2015-11-23 12:56:45 downhill.base:419 interrupted!
(OrderedDict([('loss', 0.244722477163066), ('err', 0.244722477163066), ('acc', 0.9532210977033864)]),
```

```
>>> import theanets
>>> net = theanets.Classifier(layers=[1,1,1])
I 2015-11-23 12:59:54 theanets.layers.base:371 layer Input "in": 1 inputs
I 2015-11-23 12:59:54 theanets.layers.base:207 layer Feedforward "hid1": (in:out)1 -> 1, relu, 2 parameters
I 2015-11-23 12:59:54 theanets.layers.base:207 layer Feedforward "out": (hid1:out)1 -> 1, softmax, 2 parameters
I 2015-11-23 12:59:54 theanets.graph:79 network has 4 total parameters
>>> net.load('final')
I 2015-11-23 12:59:57 theanets.graph:606 final: loaded model
<theanets.feedforward.Classifier object at 0x7ff1096a49d0>
>>> net = net.load('final')
I 2015-11-23 13:00:01 theanets.graph:606 final: loaded model
>>> x = 'acceptance'
>>> y = [[ord(z) for z in x]]
>>> y
[[97, 99, 99, 101, 112, 116, 97, 110, 99, 101]]
>>> import numpy as np
>>> y = np.array(y)
>>> net.predict(y)
I 2015-11-23 13:01:41 theanets.graph:410 building computation graph
I 2015-11-23 13:01:41 theanets.losses:67 using loss: 1.0 * CrossEntropy (output out:out)
I 2015-11-23 13:01:41 theanets.graph:514 compiling feed_forward function
array([1])
>>> x = 'tkjiafjasi'
>>> len(x)
10
>>> y = [[ord(z) for z in x]]
>>> y
[[116, 107, 106, 105, 97, 102, 106, 97, 115, 105]]
>>> net.predict(y)
array([0])
```

Future improvements

1) Mortality Prediction
    a) Better ways to picturize the timeline of variables
    b) Use other methods instead of linear approximation of curve
2) Word Prediction
    a) Predict from which language it comes
    b) Addition of features to do it irrespective of words' length

Learning from the Project

1) Our aim was to understand and use neural networks for daily life purposes
2) We have understood the use of neural networks and methods of learning in them which lead to the accuracy in the aimed problem solving
3) We also tried to learn how to picturize the graph of the data given at various timestamps as variables to other techniques using Linear Regression and other aggregating variables
4) Use of various techniques of learning in neural network domain helped us understand the trade offs in each of the cases
5) Finally we used SGD in which the learning was basically simulated using laws of 3D mathematics and physics which can be extended to n dimensions
6) This helped us to understand that basic laws of nature help us develop algorithms to solve the toughest problems existent

Contributions

We have mostly worked together as a team. General team meet involved Brainstorming, cross validating ideas and reading corresponding literature and implement the idea.

As and when we found an improvement we discussed pros and cons and took a decision. After cross checking with earlier implementation we followed the intermixing of ideas.

Thus roughly it is equal contribution from all of us, there might be places a person might have coded it up after a person who has brainstormed has explained the idea.

Thank you

1) Karan Vaidya - 130050019
2) Owais Khan - 130050050
3) Aditya Kusupati - 130050054