Neural Networks: An Introduction

Dante Tam, 5/26/2017

Neural networks are one of the most innovative learning techniques in state of the art machine learning. Classification accuracies continue to rise due to new ML research, and just recently, Google I/O announced new custom processing units that bring large-scale learning to the online cloud. These are also part of my research, which I dissect here in a light, fun fashion — for all, whether the readers are HR, project managers, CEOs, software engineers, or ML experts.

We are given data with labels, and are asked to make predictions. In classification problems, we are given data such as ["data-flavor-icecream" -> cookie dough] and their respective labels ["like-this-flavor" -> yes]. Given ["data-flavor-icecream" -> vanilla], what we do predict for ["like-this-flavor" -> ...]? In regression, we instead need to predict somewhere in a range of numbers, like ["outside-temperature" -> 85F], ["probability of going outside" -> 0.6].

Machine learning methods will return decision functions or boundaries. These are used to determine what the algorithm predicts, given some data. We say that it is a linear decision boundary if it divides a space by a line (or a plane, which can generalized to higher dimensions).

Neural networks are cool for the following reasons:

- They loosely resemble the human brain.
- They are better with subjective, "soft" tasks, like determining emotion in text.
- They can form non-linear decision boundaries.

A neural network is composed of neurons, which are lined up in layers. They are connected in one direction by an axon.

The neurons on the left are the input layer, where we set to be our data. Every other neuron takes in all its inputs, and computes an output value. This is dependent on the weights of the axons, our connections. We define the first layer to be x, x_j represents the jth neuron in the input. Let h be the hidden layer and $\hat{y}$ be the output. $W^{(1)}$ represents the first set of weights, where $W_{ji}^{(1)}$ is the connection from the jth input neuron to the ith hidden neuron. Mathematically, for a neuron z with inputs x,

$$z_i^{(1)} = \sum_{j \in x} W_{ji}^{(1)} x_j$$

Activation functions are the magic behind non-linear decisions. These functions take in the neuron's output and returns the activated output. Commonly used functions:

$$sigmoid(x) = s(x) = \frac{1}{1 + e^{-x}}$$
$$ReLU(x) = max(0, x)$$

So the activated output of a hidden layer neuron z_i is

$$a^{(1)} = s(z^{(1)}) = s(\sum W^{(1)} x)$$

i i ji j
$j \in x$

Similarly, define the next layers:

$$z_k^{(2)} = \sum_{j \in x} W_{jk}^{(2)} a_j^{(1)}$$
$$a_k^{(2)} = s(z_k^{(2)})$$

Finally, we come to loss, which is our objective. We need a measure of the error from the true labels of the input.
The final vector $a^{(2)} = \hat{y}$ represents our prediction, which is the same size as the label y. Two types of loss:

Squared loss:

$$L(a,y) = \frac{1}{2}(a-y)^2$$

Cross-entropy loss:

$$L(a,y) = \sum_i y_i ln(a_i) + (1-y_i) ln(1-a_i)$$

In [1]:
```
import numpy as np

import sklearn

from sklearn.preprocessing import normalize
from sklearn import datasets, linear_model

from random import shuffle
```

In [8]:
```
def quoteCustomSplit(text):
    firstIndex, secondIndex = -1,-1
    for i in range(len(text)):
        c_i = text[i]
        c_l = text[i-1] if i > 0 else None
        c_r = text[i+1] if i < len(text) - 1 else None
        if c_i == '"' and c_l != "\\" and firstIndex == -1:
            firstIndex = i
        elif c_i == '"' and c_r == ',' and firstIndex != -1:
            secondIndex = i
            newText = text[0:firstIndex] + text[firstIndex:secondIndex].re
place(",", "") + text[secondIndex:]
    return newText.split(",")

def readTwitterData(fname):
    parsedX = []
    parsedY = []

    with open(fname) as f:
        content = f.readlines()
    content = [x.strip() for x in content]
    contentParsed = [text for text in content if len(text) > 0]
```

```
    for line in contentParsed:
        data = quoteCustomSplit(line)
        label = float(data[len(data) - 1])
        parsedY.append(label)

        newPoint = [float(x) for x in data[2:(len(data) - 4)]]

        if data[1] == "N": newPoint = [1,0,0,0,0] + newPoint
        if data[1] == "A": newPoint = [0,1,0,0,0] + newPoint
        if data[1] == "S": newPoint = [0,0,1,0,0] + newPoint
        if data[1] == "H": newPoint = [0,0,0,1,0] + newPoint
        if data[1] == "F": newPoint = [0,0,0,0,1] + newPoint

        if data[0] == "S": newPoint = [1,0,0,0] + newPoint
        if data[0] == "C": newPoint = [0,1,0,0] + newPoint
        if data[0] == "P": newPoint = [0,0,1,0] + newPoint
        if data[0] == "T": newPoint = [0,0,0,1] + newPoint

        parsedX.append(newPoint)

    f.close()

    return parsedX, parsedY
```

In [10]:

```
dataX, dataY = readTwitterData("vectorized_tweets.txt")

dataX = np.array(dataX)
dataY = np.array(dataY)

dataX = sklearn.preprocessing.normalize(dataX, axis=1)

bestX, bestY = None, None

regr = linear_model.LinearRegression()
regr.fit(dataX, dataY)

np.set_printoptions(suppress=True)

linRegColumns = ["Topic: Sports", "Topic: Culture", "Topic: Politics", "To
pic: Twitter/Misc.",
          "Emotion: Neutral", "Emotion: Angry", "Emotion: Sad", "Emotion:
Happy/Hopeful", "Emotion: Funny/Satirical",
          "TIME2_6","TIME6_10","TIME10_14","TIME14_18","TIME18_22","TIME22
_2",
          "DATE_SUN","DATE_MON","DATE_TUE","DATE_WED","DATE_THU","DATE_FRI
","DATE_SAT",
          "PHOTO","VIDEO","ANIMATED_GIF",
          "LOG10_USER_FAV","LOG10_USER_STATUS_COUNT"]

# The coefficients
print('Coefficients: \n')

for i in range(len(linRegColumns)):
    print(linRegColumns[i] + " -> %.2f" % regr.coef_[i])

print('\n')
```

```
# The mean squared error
print("Mean squared error: %f"
      % np.mean((regr.predict(dataX) - dataY) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %f' % regr.score(dataX, dataY))
```

```
Coefficients:

Topic: Sports -> 32.90
Topic: Culture -> 20.60
Topic: Politics -> 22.98
Topic: Twitter/Misc. -> 0.00
Emotion: Neutral -> 11.78
Emotion: Angry -> 20.04
Emotion: Sad -> 10.99
Emotion: Happy/Hopeful -> 16.87
Emotion: Funny/Satirical -> 16.80
TIME2_6 -> 18.05
TIME6_10 -> 7.23
TIME10_14 -> 9.28
TIME14_18 -> 15.45
TIME18_22 -> 11.71
TIME22_2 -> 14.75
DATE_SUN -> 9.24
DATE_MON -> -1.45
DATE_TUE -> 28.30
DATE_WED -> 0.00
DATE_THU -> 14.28
DATE_FRI -> 7.49
DATE_SAT -> 18.61
PHOTO -> -2.40
VIDEO -> -1.28
ANIMATED_GIF -> 0.19
LOG10_USER_FAV -> -5.14
LOG10_USER_STATUS_COUNT -> -28.08


Mean squared error: 12.636830
Variance score: 0.297129
```

In [ ]:

In the second trial, we automate this process of categorizing tweets. CoreNLP, a Stanford research project in Natural Language Processing, is used to determine the emotion of every tweet. Topic categories are found from the main topic hashtag. We also add more information to the vector.