Using Deep Learning Neural Networks To Find Best Performing Audience Segments

Anup Badhe

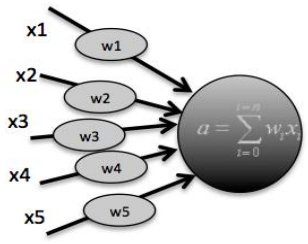
Abstract: Finding the appropriate mobile audience for mobile advertising is always challenging since there are many data points that need to be considered and assimilated before a target segment can be created and used in ad serving by any ad server. Deep learning neural networks have been used in machine learning to use multiple processing layers to interpret large datasets with multiple dimensions to come up with a high-level characterization of the data. During a request for an advertisement and subsequently serving of the advertisement on the mobile device there are many trackers that are fired collecting a lot of data points. If the user likes the advertisement and clicks on it, another set of trackers give additional information resulting from the click. This information is aggregated by the ad server and shown in its reporting console. The same information can form the basis of machine learning by feeding this information to a deep learning neural network to come up with audiences that can be targeted based on the product that is advertised.

Index Terms: deep learning neural networks, mobile advertising yield optimization, increase ROI on ad dollars, smart ad serving

1 Introduction

The success of advertising depends on finding the right audience that can be targeted from the millions of ad requests that come to the ad server. To be equipped to make the right decision to serve the ad, the ad server needs some form of machine intelligence that has been derived from the data collected over a period of time by the ad server. Since the data is huge and attributed with multiple dimensions, it needs to complex processing mechanism to derive meaningful information from it. A deep learning neural network consists of multiple processing layers, each with multiple neurons. Each neuron takes multiple **inputs** that are **weighted** and produces a single output that is fed to the next layer. The challenge is to convert each attribute to a numerical equivalent and then assign weights to it based on its contribution to the overall result. In this case we are considering the result to be a binary 1 or 0 corresponding to whether a user would click the ad or not

2 Proposed Deep Learning Neural Network



A neuron can have any number of inputs from one ton, where n is the total number of inputs. The inputs may be represented therefore as x1,x2,x3… xn. and the corresponding weights for the inputs as w1,w2,w3… wn. Now, the summation of the weights multiplied by the inputs we talked about above can be written as x1w1 + x2w2 + x3w3 …. + xnwn, which when added to a bias value produces the activation value for the neuron using the sigmoid function **σ(w *x+b)** and is defined as

$$\frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

For the purposes of this paper, the deep learning neural network would be a feed forward network. The following inputs would be fed to the network

- City
- Time of day (as Unix timestamp)
- Gender
- Age
- OS
- App Name
- Click / No Click

The sample raw input for this dataset would look like

[0] San Francisco 1457266559 M 20 iOS App1 Click
[1] San Antonio 1457373659 F 18 Android App 2 No Click
[2] Phoenix 1457356379 M 32 Android App 1 Click
[3] Las Vegas 1457446379 F 30 iOS App 2 No Click

This data needs to be standardized to be eligible for ingestion by the neural network since neural networks natively process numeric data. For example, one possibility for Gender is to encode male as 0 and female as 1. In addition to the necessity of encoding categorical data, experience has shown that neural network training is usually more efficient when numeric x-data (Age and time of day) are scaled, or normalized, so that their magnitudes are relatively similar. The last column depicts the data to be predicted which is the whether the user will click the ad after seeing it or not. Since this is a binary result the output can be achieved with 1 equivalent to the user clicking the ad and 0 corresponding to a no click of the ad.

3 Data Normalization Techniques

The numeric data in the raw sample set can be normalized using a technique called Gaussian normalization. The following formula is used **value = (value - mean) / std dev**. For the sample dataset the mean of the age column = 20+18 +32 +30) /4 = 25. The standard deviation would be **sqrt**((20-25)^2 + (18-25)^2 + (32-25)^2 + (30-25)^2)/4) = 6.08. Hence for the first age column the input value would be (20-25)/6.08

- *Anup Badhe is currently the Senior Director of Engineering at OperaMediaworks Inc, USA, PH-510-331-3710.*
- *E-mail: anupbadhe@gmail.com*

= -0.82. The same technique can be applied to the timestamp values in the dataset. As mentioned earlier the gender can be Male=0 and Female=1. For other categorical data, the normalization has to be well understood to give the neural net and balanced set of values to work with. In the sample dataset, the cities can have values based on their population with the assumption that more populous states will offer higher chances of success due to more number of people being shown the ad. Similarly the OS can get their input values based on the their market share and penetration, while the applications can get their values based on the popularity in their app stores and ratings.

4 Conclusion

For an ad server that collects a lot of data, it is often challenging to use the data to its advantage due to the complexities associated with the data and how it can be processed to derive meaningful results. Using Deep learning neural networks can help the ad server make better decisions on serving the ad to the right audience, so that it can optimize the demand it serves out to maximize the ROI for the advertiser.

Acknowledgment

The author wishes to thank Prajakta Badhe for her continuous encouragement and support.

References

[1] http://www.faqs.org/faqs/ai-faq/neural-nets/part2/.

[2] https://visualstudiomagazine.com/articles/2014/01/01/how-to-standardize-data-for-neural-networks.aspx

[3] http://neuralnetworksanddeeplearning.com/chap1.html