Feature Engineering for Click Prediction

Fernando Lourenço
fernando.lourenco@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2020

Abstract

Keywords: Deep-learning, click-through rate, conversion rate, Machine learning

1. Introduction

In the digital advertising world, publishers and advertisers have many pricing models to negotiate from. These models (Cost per Click - CPC, Cost per Mille - CPM , Cost per action - CPA, etc.) describe what an advertiser is actually paying for, so that the right model can be chosen according to their needs. Click prediction plays a very important role in this type of business because it gives us an approximate expectation about which ads can lead to a higher income. In this work we focus on two paying models, CPC and CPA.

"Deep learning is a specific sub field of machine learning: a new take on learning representations from data that puts an emphasis on learning successive layers of increasingly meaningful representations" [2]. When referring to the deep part in this field, we mean that instead of only one layer of representation (linear models), these models have successive layers stacked one after the other learning more complex projections as it deepens, each layer creates a representation based on the previous layer. Machine learning problems can be seen as a function $f(x)$ which transforms x into a certain y. The difference between machine learning (ML) - shallow and deep learning (DL) architectures is the way we compute $f(x)$, in ML can be defined as $f(x) = y$, however in DL we define as $f(x) = fL(...(f2(f1(x1))))$, where L is the number of layers.

2. Related Work

The next sections will briefly present each of the selected works, while trying to answer three important questions, namely: what is the problem being addressed, what approach did the authors propose to address that problem, and what results they were able to achieve. Finally, there are some other related works which represent other good ideas that could be applied. The proposed CCPM model is based on a convolutional neural network, and has the ability to extract local and global key features from the input instances, through wide convolutional layers alternated with dynamic pooling layers. The Deep and Cross network (DCN) has a very interesting structure because it is the combination of two separate approaches: a Cross Network, where feature crossing is done explicitly at each layer, and a Deep Network, which is a fully-connected feed-forward neural network. The solution architecture is a factorization machine (FM) component introduced by [5] to learn feature interactions up to order 2, and a deep component which is a feed-forward neural network whose objective is to learn high-order feature interaction. The first is called Factorization Machine supported Neural Network (FNN) inspired by FMs [5, 6, 8]. The second is a Sampling-based Neural Network (SNN) powered by either a sampling-based restricted Boltzmann machine (SNN-RBM) [4] or a sampling-based denoising auto-encoder (SNN-DAE) [1] using a proposed negative sampling method. The AutoInt model proposed in [7], uses a multi-head attention mechanism [9] combined with residual connections.

3. Proposed Approach

As aforementioned our goal is to compare different models that execute feature engineering automatically. Our approach began by creating a processing pipeline. First we have to select a data set, it can be either for click-through rate or conversion rate. We have selected two data sets by coincidence both CTR. The second step is to select a model to test, in the 4 we show both the theory behind the selected models and their possible implementation. Last step is to chose a proper evaluation metric, there are many functions that can be selected which will be explain further. What to expect in CTR/CVR data sets? Data comes with different types, however in these specific cases the most common are numerical and categorical, we can also find time series. The numerical features, are the ones having a meaningful ranking among them. Taking as example grades, we know that if student A had a better grade than student B, then $A > B$. Hence, these values domain is ordered, and this order has a meaning. Categorical data on the other hand, does not have an order. Lets consider another example, colors ['Red', 'Blue', 'Green'], a naive approach to encode this data would be assigning a number to each value, e.g red - 1, blue - 2 and green - 3. However, we cannot state that red ¡ blue ¡ green, so this approach would induce our model to error.

Our research lead to believe that some mathematical operations, such as: inner product, outer product, multiplication, factorization, etc; have properties that explore the relation between features. These are applied in many model layers, see chapter 2. We will test and study some of these models, in order to conclude which ones perform better and how these layers influence the prediction. To do this we selected the following models:

- Autoint
- deep & cross network
- deep factorization machine
- product-based neural network - inner and outer product version

4. Implementation

We will briefly explain the logic behind our selected models and make the bridge between the math behind each model logic and their corresponding implementation using python data science libraries. These models can be implemented using keras custom layer. By defining the build method we can create the weights and bias needed, which are learnable parameters that allow our model to keep track of how much importance the result components of our layer should have. The logic to how the features can be combined with the corresponding weights is defined in the call method, where the math that make our models unique. Exploring different algorithms to combine the features and create low and high-order combinatorial features to improve the model performance. Deep factorization machine [3], combines two architectures the factorization and deep component. The first, measure feature interactions through the inner product of their latent vectors having the following output: $yFM = \langle w, x \rangle + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \langle V_i, V_j x_i \cdot x_j$. Deep and cross network [10], as also uses a deep part which implementation would be similar to the one in the previous section. So we will only focus on the cross layer, that as the following formula: $x_{l+1} = x_0 x_l^T w_l + b_l + x_l$. This last model uses an interesting approach which can be enhanced by changing the operation related to feature interaction. In this work it is used the inner product but can be defined as a neural network. The correlation between features is define as: $\alpha_{m,k}^{(h)} = \frac{\exp^{\psi^{(h)}}(e_m,e_k)}{\sum_{i=01}^{n} \exp^{\psi^{(h)}}(e_m,e_k)}$, where ψ is our similarity function between feature m and k under the attention head h define as $\psi^h(e_m, e_k) = \langle W_{Query}^h e_m, W_{Key}^h e_k \rangle$, W_{Query} and W_{Key} are transforming matrices from the original embedding space to a new space.

5. Conclusions

In conclusion to this work, the techniques which performed better for our case studies were the outer product in PNN and the inner product in the Attention network for AutoInt. Although in training these models perform better, we were not able to extract the full power from the other models. Because, training is a process that requires a lot of resources related to time and memory; We were able to train our models through 1000 epochs achieving a stable loss and validation loss values. But with some hyper-parameter tuning, and other data strategy would have given us best results for these two models. The results we achieved were not as strong as the papers introducing these algorithms. Our main hypothesis resides in the fact that we need to apply a good embedding to the data before we apply it to the model. The best results in this area reside in weather a good representation can be found for our features. These must be in the same representation dominion so semantics is very important for this task in order to represent data in a way that their combination is meaningful.

References

[1] Y. Bengio, L. Yao, G. Alain, and P. Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems 26*, pages 899–907. Curran Associates, Inc., 2013.

[2] F. Chollet. *Deep Learning with Python.* Manning Publications, 1st edition, 2017.

[3] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He. DeepFM: A factorization-machine based neural network for CTR prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1725–1731, 2017.

[4] G. E. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*, pages 599–619. Springer, 2012.

[5] S. Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Confer-*

ence on Data Mining, ICDM '10, pages 995–1000. IEEE Computer Society, 2010.

[6] S. Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, 2012.

[7] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. *CoRR*, 2018.

[8] A. P. Ta. Factorization machines with follow-the-regularized-leader for ctr prediction in display advertising. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2889–2891, 2015.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[10] R. Wang, B. Fu, G. Fu, and M. Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, ADKDD'17, pages 12:1–12:7. ACM, 2017.