IMPERIAL

Department of Mathematics

Deep Reinforcement Learning for Ad Personalization

Martin Batěk

CID: 00951537

Supervised by Mikko Pakkanen

2 September 2024

Submitted in partial fulfilment of the requirements for the
MSc in Machine Learning and Data Science of
Imperial College London

The work contained in this thesis is my own work unless otherwise stated.

Signed: Martin Batěk Date: 17 July 2024

Abstract

ABSTRACT GOES HERE

Acknowledgements

ANY ACKNOWLEDGEMENTS GO HERE

Contents

1. Introduction

The global digital advertising market is worth approximately $602 billion today. Due to the increasing rate of of online participation since the COVID-19 pandemic, this number has been rapidly increasing and is expected to reach $871 billion by the end of 2027 (eMarketer, 2023). Many of the of the major Ad platforms such as Google, Facebook and Amazon operate on a cost-per-user-engagement pricing model, which usually means that advertisers get charged for every time a user clicks on an advertisment. This means that there is a significant commercial incentive to design Ad-serving platforms that ensure that the content shown to each user is as relevent as possible, so as to maximize user engagement and platform revenues as much as possible.

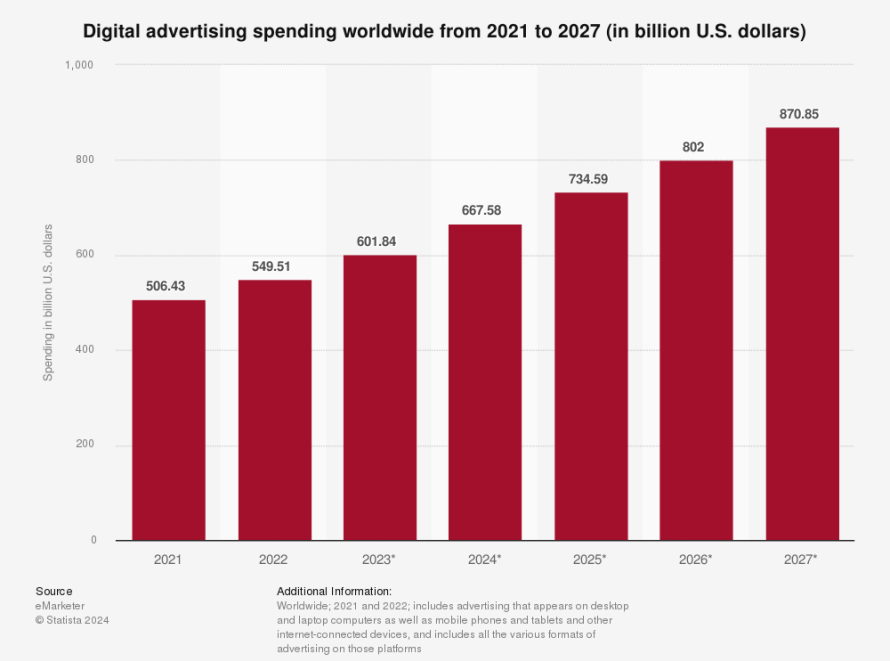

Figure 1.1.: Global Digital Ad Spending 2021-2027. Image taken from eMarketer (2023)

Attaining accurate Click-Through Rate (CTR) prediction is a necessary first step for Ad persionalization, which is why study of CTR prediction methods have been an extremely active part of Machine Learning research over the past through years. Initially, shallow prediction methods such as Logistic Regression, Factorization Machines (Rendle, 2010) and Field-Aware Factorization Machines (Juan et al., 2016) have been used for CTR prediction. However, these methods have often been shown to be unable to capture

the higher order feature interactions in the sparse multi-value categorical Ad Marketplace datasets (Zhang et al., 2021). Since then, Deep Learning methods have been shown to show superior predictive ability on these datasets. A number of Deep Learning models have been proposed, each using a different techniques for feature interaction modelling, ranging from Deep Learning extensions of Factorization Machines such as DeepFM (Guo et al., 2017), to novel methods such as AutoInt (Song et al., 2019). By employing a multi-towered neural network architecture, these models are able to capture both low-order and high-order feature interactions in the data, and therefore tend to achieve supperior predictive performance to their shallow counterparts.

However, irrespective of how well these models perform in a static environments, the reality is that user preferences and advertisment characteristics are constantly changing. Like most online reccomender systems, Ad personalization models must be able to adapt to these changes in order to continue to provide accurate predictions over the longer period (Zheng et al., 2018). This problem necessitates the use of Reinforcement Learning for Ad personalization.

Reinforcement Learning is a subdomain of Machine Learning in which the goal is for an agent to learn an optimal policy that maximizes the expected reward in an environment where the state-action-reward progression can be modelled as a Markov Decision Process (Puterman, 2014). Early Reinforcement Learning methods involved deriving a the transition probabilities for the state-action pairs on the basis of interactions with the environment and then using Dynamic Programming methods such as the Upper Confidence Bound RL (UCB-RL) algorithm (Auer et al., 2008) and the the Thompson Sampling algorithm for Reinforcement Learning (Pike-Burke, 2024). However, in cases where the state-action space is too sparce to be reasonably enumerated, it is often more practical to user a function approximator to directly estimate the expacted cumalative reward for each action in each state. This method of Reinfocement Learning is commonly referred to as Q-learning (Watkins, 1989), and has the advantage of being *model-free*, meaning that it does not require the agent to have a model of the environment thereby making it more scalable to large and sparce datasets. In (Hornik et al., 1989), (Cybenko, 1989) and (Hornik et al., 1990) Deep Neural Networks with activation functions are shown to be universal function approximators which naturally lead to the incorporation of DNN's in Q-Learning. This has lead to the development of the Deep Q-Learning Agent, which has been shown to be able to learn optimal policies in a number of different domains, such as the Atari 2600 game environment Mnih et al. (2015). Beyond this, Deep Reinforcement Learning has shown promising results in a number of different aplications, including robot control and computer vision (Wang et al., 2024). In the context of Ad personalization, DRL has also be applied to online reccomender systems such as News article reccomendation (Zheng et al., 2018) and video reccomendation on Youtube (Chen et al., 2019). In both papers, the authors show that the DRL agent is able to learn an optimal content reccomendation policy on the basis of user engagement data. This reveals that there is potential for applying these methods to the problem of Ad personlization, thereby creating a truely adaptive marketing platform.

Research Question and Contributions

In this report, I aim to construct a Ad serving system that is truely adaptive and personalized to the changing user preferences and advertisment charateristics. In order to achieve this goal, I will first need to find a suitable Deep Learning Model arcitecture for CTR prediction, and then incorporating this model as the Q-function approximator in a Deep Q-Learning algorithm. The key contributions that I make in this report are as follows:

- I evaluate the performance of five popular Deep Learning models for CTR prediction on three well-known benchmark datasets, Criteo (Tien et al., 2014), KDD12 (Aden, 2012) and Avazu (Wang and Cukierski, 2014).
- I construct a novel Deep Reinforcement Learning Frame for Ad personalization, and as a proof-of-concept and evaluate its performance using the KDD12 dataset.

Structure of the Report

In chapter 2, I begin by providing a background introducing the problem of Click-Through Rate prediction in the context of Ad personalization, and explore the unique challenges posed by the typically sparse multi-value categorical datasets that are common in the Ad marketplace. I then proceed to review the literature on Deep Learning models for CTR prediction, highlighting the different techniques that each framework uses to capture the key feature interactions in the data. I also review the literature on Deep Reinforcement Learning, specifically the DRN algorithm introduced by Zheng et al. (2018), which can be analogously applied to the Ad personalization context. In chapter 3, I evaluate the performance of different Deep Learning models for CTR prediction on three well-known benchmark datasets, Criteo (Tien et al., 2014), KDD12 (Aden, 2012) and Avazu (Wang and Cukierski, 2014). In chapter 4, I construct a Deep Reinforcement Learning model for Ad personalization and evaluate its performance on the same benchmark datasets. Finally, in chapter 5, I discuss the results of the experiments and provide some concluding remarks.

2. Background

2.1. Deep CTR Prediction

2.1.1. Problem Formulation and Ad Marketplace Data

In their respective surveys on the use of Deep Learning methods for CTR prediction, Gu (2021) and Zhang et al. (2021) outline the problem of CTR prediction as one that essentially boils down to a binary (click/no-click) classification problem utilizing user/ad-view event level online session records. The goal of CTR prediction is to train a function f that takes in a set of ad marketplace features $\mathbf{x} \in \mathbb{R}^n$, and maps these to a probability that the user will click on the ad in that given context. In other words, $f_\Theta : \mathbb{R}^n \to \mathbb{R}$ such that:

$$\mathbb{P}(\text{click}|\mathbf{x}) = \mathbb{P}(y = 1|\mathbf{x}) = \sigma(f_\Theta(\mathbf{x})) \tag{2.1}$$

where y is the binary click label, Θ represents the parameter vector for f and $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function. To ease the notation for the rest of the report, we will use the shorthand $p(y) = \mathbb{P}(y = 1|\mathbf{x})$ formulations in the following sections.

An instance of the ad marketplace features $\mathbf{x}$ is typically recorded at a user/ad impression event level and typically consists of

- **User Features:** Features that describe the user, such as User ID, demographic information, metrics related to the user's past interactions with the platform, etc.
- **Ad Features:** Features that describe the ad, such as Ad ID, Advertiser ID and Ad Category.
- **Contextual Features:** Features that describe the context in which the ad is being shown, such as the time of day, the position of the ad on the page and the site on which the ad is being shown.

Figure 2.1 shows a snapshot of the KDD12 dataset, which is a typical example of the type of data that is used for CTR prediction.

A defining characteristic for this type of data is that many of the features are multi-value categories with a high degree of of cardinality (He and Chua, 2017). In order to user categorical data in a classifier model, it is common practice to embed the these categorical features as multidimensional vectors. While the dimensionality of these embeddings can vary amongt the different sparse categorical features, for the sake of simplicity of notation, below we assume that all sparse categorical feature embeddings have the same dimensionality, D. Let $\mathbf{x_i}^{OH}$ be the one-hot encoded vector representation of the

Click	Impression	DisplayURL	AdID	AdvertiserID	Depth	Position	QueryID	KeywordID	TitleID	DescriptionID	UserID
0	1	4.29812E+18	7686695	385	3	3	1601	5521	7709	576	490234
0	1	4.86057E+18	21560664	37484	2	2	2255103	317	48989	44771	490234
0	1	9.70432E+18	21748480	36759	3	3	4532751	60721	685038	29681	490234
0	1	1.36776E+19	3517124	23778	3	1	1601	2155	1207	1422	490234
0	1	3.28476E+18	20758093	34535	1	1	4532751	77819	266618	222223	490234
0	1	1.01964E+19	21375650	36832	2	1	4688625	202465	457316	429545	490234
0	1	4.20308E+18	4427028	28647	3	1	4532751	720719	3402221	2663964	490234
0	1	4.20308E+18	4428493	28647	2	2	13171922	1493	11658	5668	490234
0	1	5.85475E+17	20945590	35083	2	1	35143	28111	151695	128782	490234
0	1	9.68455E+18	21406020	36943	2	2	4688625	202465	1172072	973354	490234
	Target		Ad Features			User Features			Contextual Features		

Figure 2.1.: Snapshot of the KDD12 dataset Aden (2012)

categiorical feature x_i. Then the *embedded* feature vector $\mathbf{e}_i$ for categorical feature x_i is given by:

$$\begin{aligned}\mathbf{e}_i &= \mathbf{B}_i \mathbf{x}_i^{OH} \\ &= \left[\mathbf{b}_1^i, \ldots, \mathbf{b}_{m_i}^i\right] \mathbf{x}_i^{OH} \\ &= \begin{bmatrix} b_{1,1}^i & \cdots & b_{1,C_i}^i \\ \vdots & \ddots & \\ b_{D,1}^i & \cdots & b_{D,C_i}^i \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{bmatrix}\end{aligned} \tag{2.2}$$

where $\mathbf{B}_i$ is the embedding matrix for feature x_i, whose dimensions are determined by the chosen embedding dimension D and the cardinality of the feature, C_i. Assuming that value of x_i is equal to the k-th value in the one-hot encoding mapping, and that therefore the k-th value in $\mathbf{x}_i^{OH}$ is equal to one, Equation 2.2 then simplifies to

$$\mathbf{e}_i = [e_{i1}, \ldots, e_{i,D}]^\intercal = \left[b_{1k}^i, b_{2k}^i, \ldots, b_{Dk}^i\right]^\intercal = \mathbf{b}_k^i \tag{2.3}$$

which is the k-th column of the embedding matrix $\mathbf{B}_i$, otherwise referred to as the k-th embedding vector (Hancock and Khoshgoftaar, 2020). The processed data $\tilde{\mathbf{x}}$ that then gets fed into the model is then composed of a concatenation of all sparse feature embeddigs $\mathbf{e}_i$ and standardized dense numerical feature values $z_i = (x_i - \bar{x_i})/\sqrt{Var(x_i)}$:

$$\begin{aligned}\tilde{\mathbf{x}} &= [\mathbf{e}_1^\intercal, \ldots, \mathbf{e}_s^\intercal, z_{s+1}, \ldots, z_{s+d}] \\ &= [e_{1,1}, \ldots, e_{1,D}, e_{2,1}, \ldots, e_{s,D}, z_{s+1}, \ldots, z_{s+d}] \\ &= [\tilde{x}_1, \ldots, \tilde{x}_{\tilde{n}}]\end{aligned} \tag{2.4}$$

where s and d are the number of *sparse* categorical features and *dense* numerical features respectively in $\mathbf{x}$, and $\tilde{n} = D \cdot s + d$ is the resulting dimensionality of $\tilde{\mathbf{x}}$. Again, to ease the notation in the remainder of the report, we will assume that the prepocessing steps described above are applied to the data, and all formulaic expressions of the models

in the following section with be expressed in terms of $\tilde{\mathbf{x}} = \{\tilde{x}_j\}_{j=1}^{\tilde{n}}$.

The problem posed by high cardinality is that when m_i is large, the high sparsity of the one-hot encoded vector x_i^{OH} can make in extremely difficult for a model to learn the key *implicit* features and patterns present in the data (Gu, 2021). This is indeed the key challenge in building an accurate CTR prediction model, and is a key motivating factor as to why Deep Neural networks have out performed the classical shallow counterparts. This transition will be examined in more detail in the following section.

2.1.2. Shallow CTR Models

Logistic Regression

The earliest examples of CTR classification models incorporated classical "shallow" (single layer) statistical regression methods. The most basic example of this was the **Logistic Regression** model, as implemented by Richardson et al. (2007) on advistisment data from the Microsoft Search engine. The LR model is composed by modelling the *log-odds* (also referred to as the *logit*) of a positive binary label as a linear combination of all of the respective feature values:

$$f_\Theta^{LR}(\tilde{\mathbf{x}}) = \theta_0 + \sum_{j=1}^{\tilde{n}} \theta_j \tilde{x}_j \tag{2.5}$$

where $f_\Theta^{LR} : \mathbb{R}^n \to \mathbb{R}$ represents the Logistic regression model parametized by $\Theta = (\theta_0, \ldots, \theta_{\tilde{n}})$. The benefits of the LR model are that due to its simplicity and low number of parameters, it is relatively easy to train in computational terms and also relatively easy to deploy (Zhang et al., 2021). Howevever, the formulation in equation 2.5 reveals that the LR model does not explicitly account for *feature interactions*. As outlined in section 2.1.1, the categorical features tend to have a high cardinality, resulting in highly sparse feature embeddings. Many of the of the important patterns for CTR prediction are therefore likely to be expressed in terms of *combinations of features* rather then the individual feature values themselves. For example, a user's tendancy to click on a given advertisment is likely to by influenced by the *combination* of the category of good or service the given advertisment is trying to sell (e.g. premium fashion retail, travel, electronics et. cetera) and the demographic/socio-economic category that the given user falls into (e.g. university student, young professional, retiree). These features combinations (and the corresponding combination of respective field values in the preprocessed feature vector $\tilde{\mathbf{x}}$) are commonly referred to in the literature as *cross-features* (Zhang and Zhang, 2023) or more commonly *feature interactions* (Cheng et al., 2016; Song et al., 2019; Xiao et al., 2017).

Whilst it is possible to incorporate feature interactions in an LR model through feature engineering, this quickly becomes infeasible for large sparse datasets. A number of techniques have been developed to automate the necessary feature engineering steps for this, either by implicitly assigning a weight to all second order feature interactions (Chang et al., 2010) or by utilizing Gradient Boosted Decision Trees to pick out the key

interactions (Cheng et al., 2014). Unfortunately, the prior still tends to exhibit poor performance with sparse data, whereas the fact that the Gradient Boosting algorithm for the latter is difficult to paralellize makes this solution difficult to scale in many applications in practice (Zhang et al., 2021).

Factorization Machines

Factorization Machines first proposed by Rendle (2010) can be thought of as an extension of the Logistic Regression framework in equation 2.5 with additional terms that explicitly account for the interactions between different features. Its relative simplicity and computational scalability has made it a widely popular framework for CTR modelling (Gu, 2021). A 2-way (maximum feature interaction degree of 2) Factorization Machine model is formulated as:

$$f_{\Theta}^{FM^2}(\tilde{\mathbf{x}}) = \theta_0 + \sum_{j=1}^{\tilde{n}} \theta_j \tilde{x}_j + \sum_{j=1}^{\tilde{n}} \sum_{k=j+1}^{\tilde{n}} \langle \mathbf{v}_j, \mathbf{v}_k \rangle \tilde{x}_j \tilde{x}_k \tag{2.6}$$

where $\langle \cdot, \cdot \rangle$ represents the inner product between two vectors, the final interaction term above is parametized by $\mathbf{V} \in \mathbb{R}^{\tilde{n} \times F}$. Each row $\mathbf{v}_j$ of $\mathbf{V}$ represents the j-th feature in $\tilde{\mathbf{x}}$ in terms of F latent factors. The factorization matrix $\mathbf{V}$ is typically fitted by optimizing the binary cross-entropy loss function by means of Stochastic Gradient Descent. It is intutive to see that $\mathbf{V}$ will be fitted such that if the interaction between feature j and k have a positive impact on $p(y)$, then j-th and k-th rows of $\mathbf{V}$ will have positive inner products (and vice versa) (Zhang et al., 2021).

Rendle (2010) shows that although direct evaluation of equation 2.6 would appear to have a complexity of $O(F\tilde{n}^2)$, the 2-way FM model in fact scales linearly in $\tilde{n}$ and F:

Lemma 2.1.1. The model equation of a 2-way factorization machine (eq. 2.6) can be computed in linear time $O(L\tilde{n})$.

Proof.

Due to the factorization of the pairwise interactions, there is no model parameter that directly depends on two features (j, k). This means that pairwise interactions can be reformulated as such

$$
\begin{aligned}
&\sum_{j=1}^{\tilde{n}} \sum_{k=j+1}^{\tilde{n}} \langle \mathbf{v}_j, \mathbf{v}_k \rangle \tilde{x}_j \tilde{x}_k \\
&= \sum_{j=1}^{\tilde{n}} \sum_{k=j+1}^{\tilde{n}} \sum_{f=1}^{F} v_{j,f} v_{k,f} \tilde{x}_j \tilde{x}_k \\
&= \frac{1}{2} \left(\sum_{j=1}^{\tilde{n}} \sum_{k=1}^{\tilde{n}} \sum_{f=1}^{F} v_{j,f} v_{k,f} \tilde{x}_j \tilde{x}_k - \sum_{j=1}^{\tilde{n}} \sum_{f=1}^{F} v_{j,f} v_{j,f} \tilde{x}_j \tilde{x}_j \right) \\
&= \frac{1}{2} \sum_{f=1}^{F} \left(\sum_{j=1}^{\tilde{n}} v_{j,f} \tilde{x}_j \sum_{k=1}^{\tilde{n}} v_{k,f} \tilde{x}_k - \sum_{j=1}^{\tilde{n}} v_{j,f}^2 \tilde{x}_j^2 \right) \\
&= \frac{1}{2} \sum_{f=1}^{F} \left(\left(\sum_{j=1}^{\tilde{n}} v_{j,f} \tilde{x}_j \right)^2 - \sum_{j=1}^{\tilde{n}} v_{j,f}^2 \tilde{x}_j^2 \right)
\end{aligned}
$$

The complexity of the final line above is $O(L\tilde{n})$, and hence the FM formulation as per equation 2.6 scales linearly in L and $\tilde{n}$. □

This quality greatly simplifies the computational complexity of scaling the FM model to larger datasets with a more sparse categorical features. Morever, the Factorization Machine framework can be generalized to degree R (i.e. up to any limit of feature interation order) as follows:

$$
f_{\Theta}^{FM^R} \quad = \quad \theta_0 \; + \; \sum_{j=1}^{\tilde{n}} \theta_j \tilde{x}_j \; + \; \sum_{r=1}^{R} \sum_{j_1=1}^{\tilde{n}} \cdots \sum_{j_r=j_{r-1}+1}^{\tilde{n}} \left(\prod_{k=1}^{r} \tilde{x}_{j_k} \right) \left(\sum_{f=1}^{F_r} \prod_{k=1}^{r} v_{j_k,f}^{(r)} \right) \tag{2.7}
$$

The FM framework therefore provides an intuitive and computationally scalable method to account for key feature interactions without the need of extensive feature engineering. Extensions and improvements to FM have been proposed, most notably in the form of the Field-Aware Factorization Machine (FFM) framework by Juan et al. (2016), which only acounts for intereactions between features of different fields (in otherwords, it ignores the interaction between $\tilde{x}_j$ and $\tilde{x}_k$ if both are componets of embedding vector $mathbfe_i$ for some categorical feature x_i) as well as Gradient Boosted Factorization Machines (Cheng et al., 2014), which again aims to augment the FM framework by means of the Gradient Boosting algorithm.

2.1.3. Introducing MLP's in CTR prediction

Despite the advantages of the FM framework, a setback of the formulation in equation 2.7 is that the framework grows highly compex and overparametized for higher values of R. As a consequence, only the 2-way FM framework as per equation 2.6 tends to be

implemented in practice, meaning that in FM alone is feasably insufficient for capturing feature interactions of order >> 2 (Guo et al., 2017). Deep Neural Networks present a powerful alternative for addressing this shortcoming. Neural networks benefit from being universal function approximators (Cybenko, 1989) and from the fact that neural network batch training is paralelizable by means of GPU accelerated computation. This has lead to the successfull application of Deep Learning algorithms across multiple fields such as Natural Language Processing and Image classification (He et al., 2016; Krizhevsky et al., 2017; LeCun et al., 1998). These factors and successes showed that DNN's have the potential to extract informative feature representations from highly sparse and abstract data, and as a consequence, the application of DNN's in CTR prediction started recieving attention in the mid-2010's.

The **Multilayer Perceptron** (MLP) is the most elementary type of Deep Neural Network (Webster, 2024). In general, a MLP with L hidden layers is formulated as such:

$$\mathbf{h}^{(0)} := \tilde{\mathbf{x}} \tag{2.8}$$

$$\mathbf{h}^{(l)} = \phi_l \left(\mathbf{W}^{(l-1)} + \mathbf{b}^{(l-1)} \right), l = 1, \ldots, L \tag{2.9}$$

$$\hat{y} = \phi_{out} \left(\mathbf{w}^{(L)} \mathbf{h}^{(L)} + b^{(L)} \right) \tag{2.10}$$

where $\mathbf{W}^{(k)} \in \mathbb{R}^{n_{l+1} \times n_l}$, $\mathbf{b}^{(k)} \in \mathbf{R}^{n_{l+1}}$, $\mathbf{h}^{(l)} \in \ltimes_{\lessdot}$, $n_0 = \tilde{n}$, n_l is the number of hidden units in layer l and ϕ_l is the activation function for layer l, When rearranged in the form of equation 2.1, the above becomes:

$$f_{\Theta}^{MLP}(\tilde{\mathbf{x}}) = \psi_{out} \left(\psi_L \left(\cdots \psi_1 \left(\tilde{\mathbf{x}} \right) \cdots \right) \right) \tag{2.11}$$

where each function ψ_l represents the affine transformation and element-wise activation operation for layer l. MLPs can be thought of as an acyclic graph, as displayed in Figure 2.2. The data $\tilde{\mathbf{x}}$ first gets fed through the *input layer*, then gets processed by multiple *hidden layers* that include a series of affine transfromations followed by activation functions, before the final result is produced by the *output layer*.

Figure 2.2 demonstrates the potential that DNN's have for modelling higher order feature interactions. By training the network by means of Stochastic Gradient Descent, it should be possible to calculate the appropriate weight ($\mathbf{W}_l$) and bias ($\mathbf{b}_l$) parameters in order capture the relevant high-order feature patterns in the data. As such, many CTR modelling frameworks have been developed that use Deep Learning techniques to build upon and imporved the previously discussed classical methods by incorporating Deep Neural Networks such as the ML in the model architecture (Zhang et al., 2021).

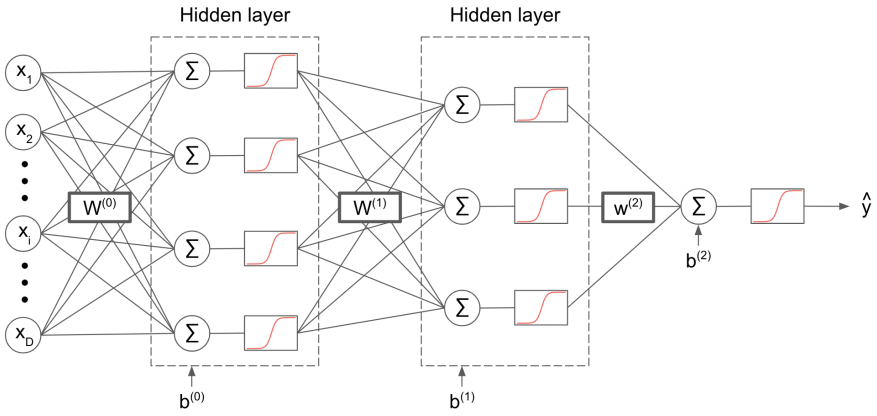

Figure 2.2.: Multilayer Perceptron with two hidden layer. Taken from (Webster, 2024)

2.1.4. Single vs Dual Tower Architectures

2.1.5. MLP Enhanced CTR models

Factorization-machine Supported Neural Networks

Wide and Deep

The so-called **Wide and Deep** (W&D) model was developed by Cheng et al. (2016):

$$f_{\Theta}^{W\&D} = \theta_0 + \sum_{k=1}^{\hat{n}} \theta_k \hat{\mathbf{x}}_k + f_{\Phi}^{MLP}(\tilde{\mathbf{x}}) \tag{2.12}$$

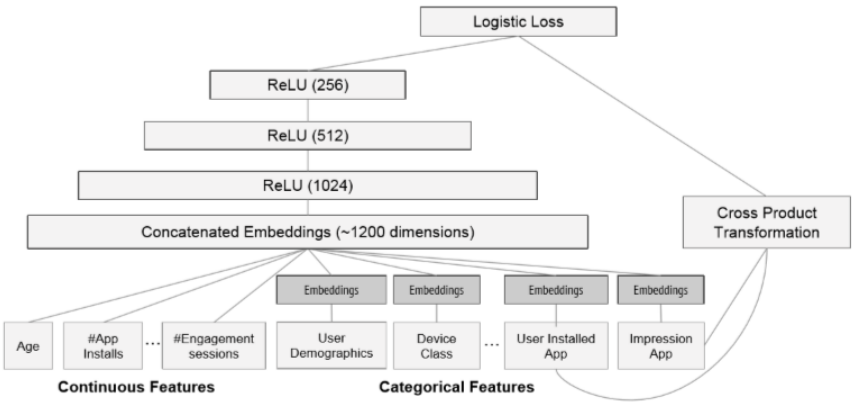

Figure 2.3.: Wide and Deep Model, as illustrated in (Shen, 2017)

Figure 3.3 reveals that the W&D model is composed with a Dual-Tower Architecture, with a Deep Component and a Wide Component (shown on the left and right hand sides

of Figure 3.3 respectively). The Deep Component is composed of a MLP with multiple hidden layers, each with the Rectified Linear Unit (ReLU) activation function. The Wide Component is formulated by the first two terms in equation 2.12, and is composed of a simple linear transformation of the input features. The key aspect of the Wide Component to take note of is that the linear transformation is not simply applied to the preprocessed features $\tilde{x}$, but instead to these concatinated with a set of cross-product transformed features. In other words:

$$\hat{\mathbf{x}} = [\tilde{\mathbf{x}}, \upsilon_1(\tilde{\mathbf{x}}), \ldots, \upsilon_P(\tilde{\mathbf{x}})] \tag{2.13}$$

Where $\upsilon_k(\tilde{\mathbf{x}}) = \prod_{j=1}^{\tilde{n}} \tilde{x}_j^{c_{kj}}$ and $c_{kj} \in \{0, 1\}$.

Consequentially of equation 2.13, the Wide Component *memorizes* the key feature interations that are defined by the specific *cross-product transformations* ($\upsilon_k(\tilde{\mathbf{x}})$) (Cheng et al., 2016). Meanwhile, the Deep Component captures any residual signals that may not have been explicitly included in the manually defined cross-product transformations (Zhang et al., 2021).

DeepFM

2.1.6. Feature Interaction Operator Models

Product Operators - Product Based Neural Network

Convolutional Operators - Feature Generation Convolutional Neural Network

Attention Operators - AutoInt

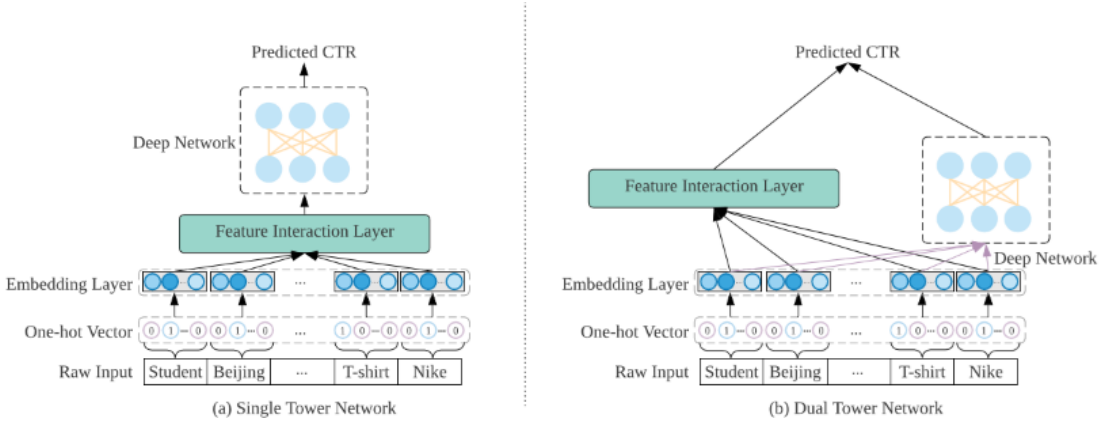

Figure 2.4.: Deep Neural Network Architecture for CTR prediction. Image taken from Zhang et al. (2021)

2.2. Deep Reinforcement Learning

2.2.1. Reinforcement Learning Basics: Markov Decision Processes and Dynamic Programming

2.2.2. Q-Learning and Deep Q-Learning

2.2.3. DRN: Deep Reinforcement Learning for News Recommendation

In their survey, (Wang et al., 2024) describe how deep reinforcement learning combines the aforementioned feature extraction capabilities of DNN's with the decision-making capability of reinforcement learning, which aims to learn an optimal state-action policy which maximizes the expected reward gained in a given environment. In the context of recommendation systems, a significant amount of research has been dedicated to formulating the recommendation problem as a Contextual Multi-Armed Bandit (MAB) problem setting, where the context consists of user, site and item features (Bouneffouf et al., 2012; Li et al., 2010; Zeng et al., 2016). However, a shortcoming for the MAB approach is that it does not explicitly model the future expected reward for the policy, which may be detrimental in the longer term (Zheng et al., 2018). Markov Decision Process (MDP) models solve for this issue by modelling the state-action progression as a Markov Process, allowing for the stochastic valuation of the future potential rewards for a given recommendation policy (Lu and Yang, 2016; Mahmood and Ricci, 2007). DRN (Zheng et al., 2018) is a MDP framework that leverages a Deep Neural Network to approximate the expected total user response for each recommendation at each state. The two major advantages of DRN are firstly that it is composed on the basis of a continuous state and action representation, meaning that it can be scaled to large and sparse datasets, and secondly that the proposed reward function consists of both the immediate reward (user click) as well as the future expected reward (long term user engagement), thereby allowing for better recommendations over a user's lifetime.

3. Deep CTR model Evaluation

3.1. Model Selection Methodology

As explained above, I will explore a number of deep learning models. I selected five popular models on the basis of the following criteria

- Competitive predition accuracy in the KDD12, Criteo and Avazu datasets as published on Papers with Code.
- Ideally, I was looking for a representitive set of models for each model type as discussed in (Zhang et. al. 2021). Therefore I was looking for models that employed Product Interaction Opetators, Attention Operators and Factorization Machines as a basis.
- The code for the model has to be accessible and intuitive to use.

On the basis of the above critea, I have chosen the following models to explore:

- Factorization Supported Neural Networks
- Product Based Neural Networks
- Wide and Deep
- DeepFM
- Automatic Feature Interaction (AutoInt)

In the section below, I briefly introduce each of the models, and evaluate against the benchmark datasets loaded and preprocessed above.

3.2. Model Summaries

3.2.1. Shallow Models

Logistic Regression

Factorization Machines

Factorization Machines were first introduced in (Rendle, 2010) as a model class that "combines the advantages of Support Vector Machines (SVM) with factorization models". The model is able to capture the second order feature interactions in the data, which is a key advantage over Logistic Regression. The model is defined as follows:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \tag{3.1}$$

where w_0 is the bias term, w_i are the weights for the i-th feature, $\mathbf{v}_i$ are the latent vectors for the i-th feature. Rendle (2010) shows that the learned biases and weights of the FM model can be computed in linear time, "and can be learned efficiently by gradient descent methods", such as Stochastic Gradient Descent (SGD).

3.2.2. Deep Models

Factirization Supported Neural Networks

The first Deep Learning model that we will consider is the Factorization Supported Neural Network (FNN) model proposed by Zhang et al. (2016). The model works by first training a Factorization Machine model on the sparse-encoded categorical input features. It then uses the latent vectors learned by the FM model (see $\mathbf{v}_i$ in equation 3.1) as inputs to a Neural Network, as shown in Figure 3.1. In doing so, the FNN model is effectively using the FM latent factors to initialize the embedding layer of the Neural Network. The DNN is then able to learn the higher order feature interactions in the data, which the FM model is unable to capture.

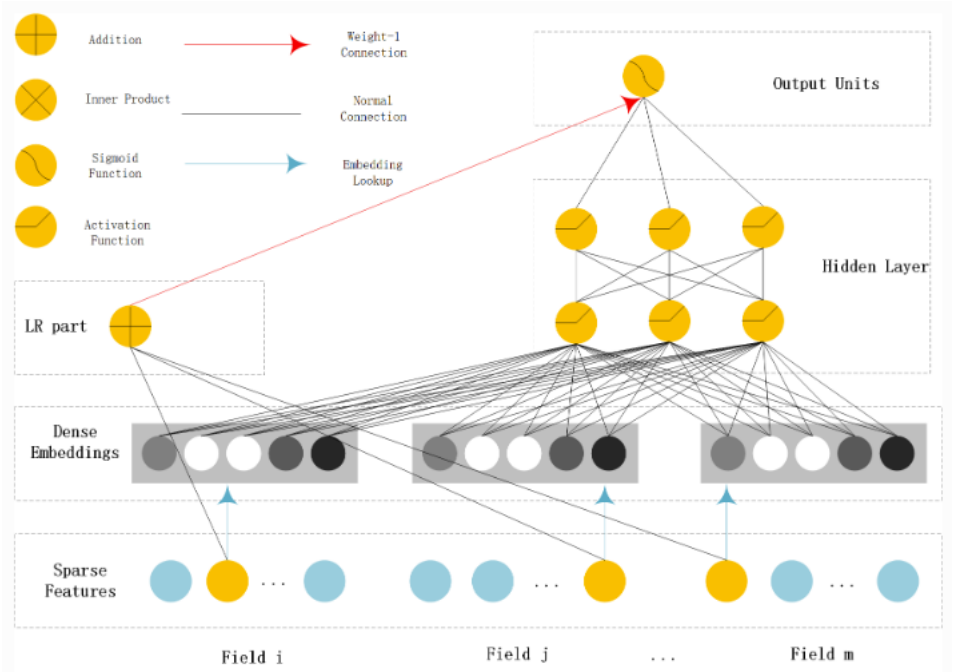

Figure 3.1.: Factorization Supported Neural Network as proposed by Zhang et al. (2016). Image taken from Shen (2017)

Product Based Neural Networks

The Product Based Neural Network (PNN) model proposed by Qu et al. (2016) is another Deep Learning model that was developed around the same time as the FNN model. The key innovation of the PNN moel is the use of a pair-wisely connected Product Layer after a field-wise connected embetting layer for the categorical features, as shown in Figure 3.2. The Product Layer is able to directly model inter-field feature interaction by means of either an inner product or outer production operation, and then further distill higher feature inturactions by passing the output of the Product Layer through fully connected MLP layers.

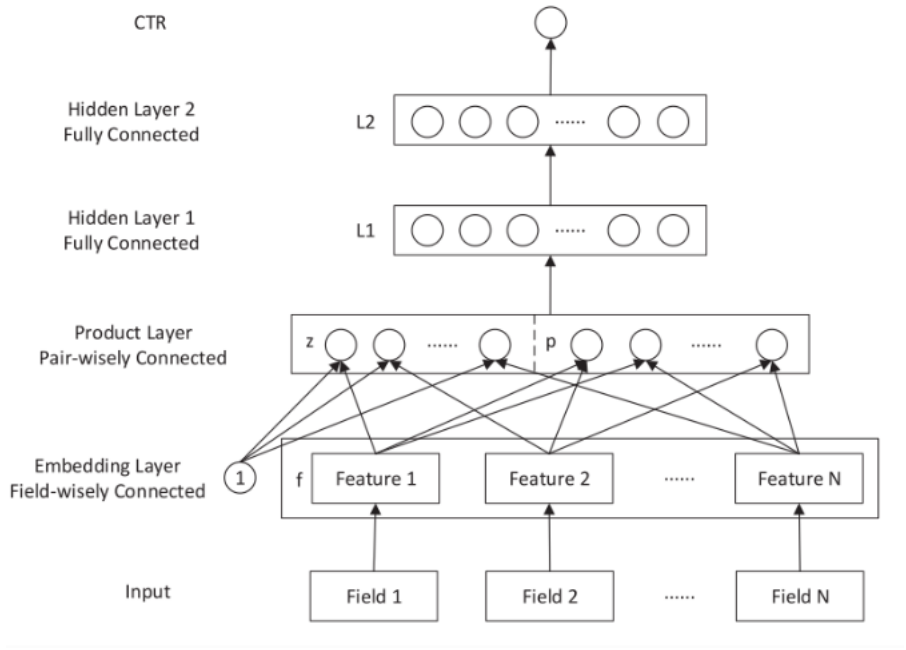

Figure 3.2.: Product Based Neural Network as proposed by Qu et al. (2016). Image taken from Shen (2017)

Wide & Deep Learning

The Wide & Deep Learning (WDL) model proposed by Cheng et al. (2016) introduces the concept of dual-tower model architecture (Zhang et al., 2021). While both the FNN and the PNN models generally tend to be constructed as a single fully connected DNN model, the Wide & Deep model consists of a wide component, consisting of a three layer Deep Neural Network that takes the concatinated embedding vectors of the categorical features as input, and a deep component, consisting of a cross product transformation of selected sparse categorical features. The logits from the wide and deep components are added together to produce the final prediction. The architecture of the WDL model

is shown in Figure 3.3.

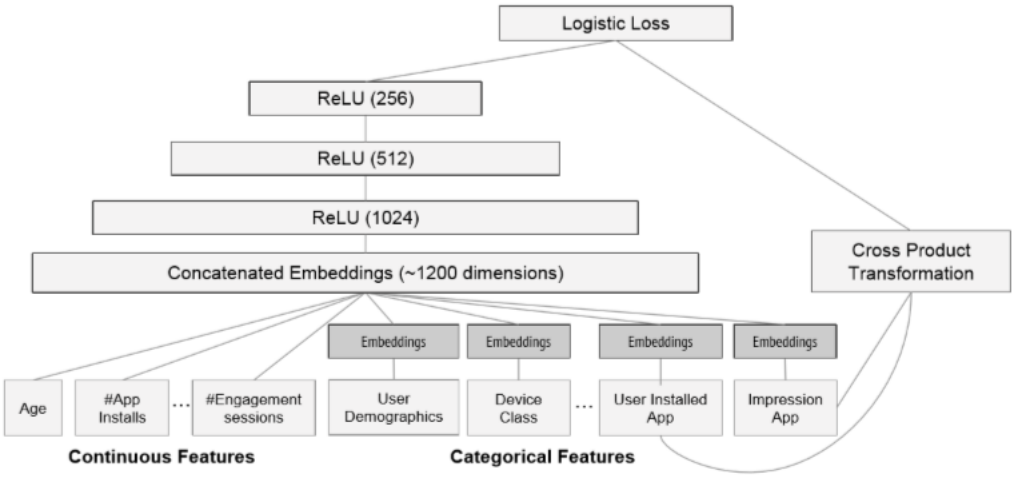

Figure 3.3.: Wide & Deep Learning model as proposed by Cheng et al. (2016)

The purpose behind the Dual-Tower architecture is to counteract the tendancy of the fully connected single tower DNN models to lose the ability to capture low-order feature interactions (Zhang et al., 2021). The Wide component is able to capture the low-order feature interactions, while the Deep component is able to capture the higher order feature interactions.

DeepFM

The DeepFM model proposed by Guo et al. (2017) can be thought of as an imporvement of the aforementioned FNN (Zhang et al., 2016) and WDL (Cheng et al., 2016) models. Like the FNN model, the DeepFM model usilises the Factorization Machine model (Rendle, 2010) to learn lower-order feature interactions. However, it also employs a dual-tower architecture like the WDL model, with the Wide component being the FM model and the Deep component being a fully connected DNN model. The DeepFM model is therefore able to avoid the limitations on capturing low-order interactions that are inherent in the FNN model. In addition, due the the application of the FM to all feature embeddings, the DeepFM model eliminates the need to choose which features to feed through the wide component, as is the case in the WDL model. The architecture of the DeepFM model is shown in Figure 3.4.

Automatic Feature Interaction Learning

The Autotomatic Feature Interaction Learning (AutoInt) model proposed by Song et al. (2019) makes use of a multi-head self attention network to model the important feature interactions in the data. The initial paper separates the model into three parts: an

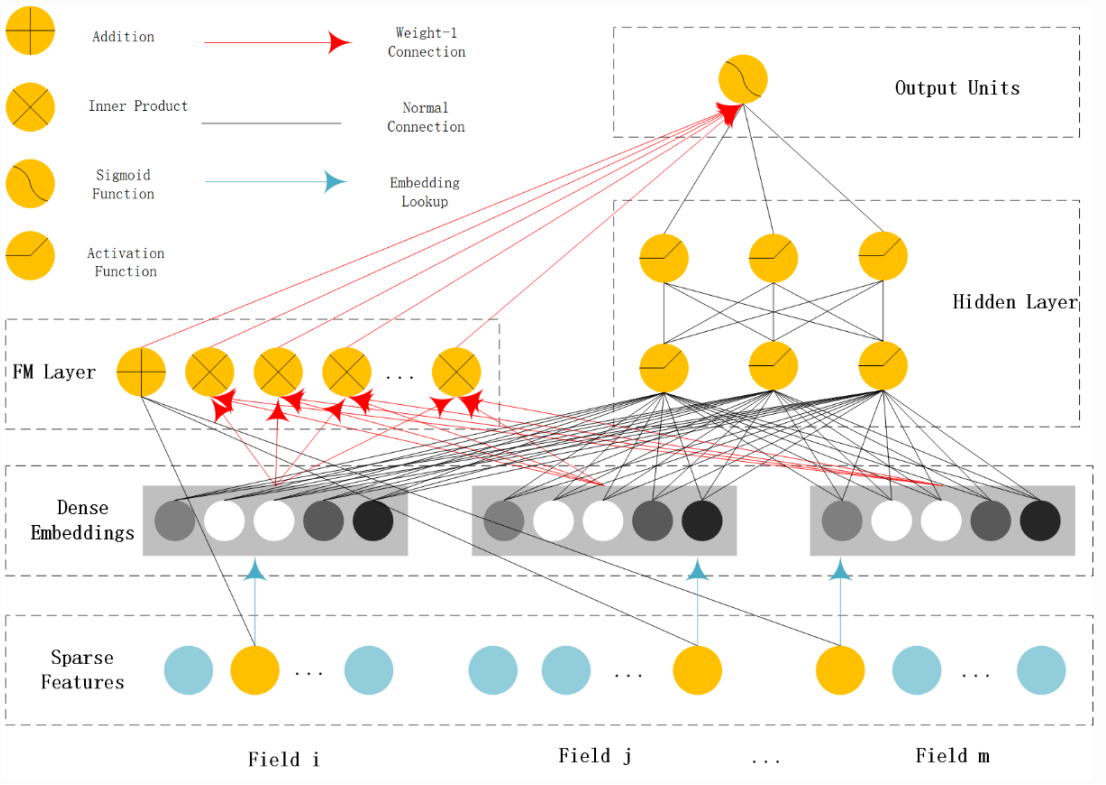

Figure 3.4.: DeepFM model as proposed by Guo et al. (2017). Image taken from Shen (2017)

embedding layer, an interaction layer and an output layer. The embedding layer aims to project each sparse multi-value categorical a and dense numerical feature into a lower dimensional space, as per the equation 3.2:

$$\mathbf{e_i} = \frac{1}{q}\mathbf{V_i x_i} \tag{3.2}$$

where $\mathbf{V_i}$ is the embedding matrix for the i-th field, x_i is a multi-hot vector, and q is the number of non-zero values in x_i. The interaction layer employs the multi-head mechanism to determine which higher order feature interaction are meaningful in the data. This not only improves the efficiency of model traning, but it also improves the model's explainability. Lastly, the output layer is a fully connected layer that takes in the concatinated output of the interaction layer, and applies the sigmoid activation function to produce the final prediction. The architecture of the AutoInt model is shown in Figure 3.5.

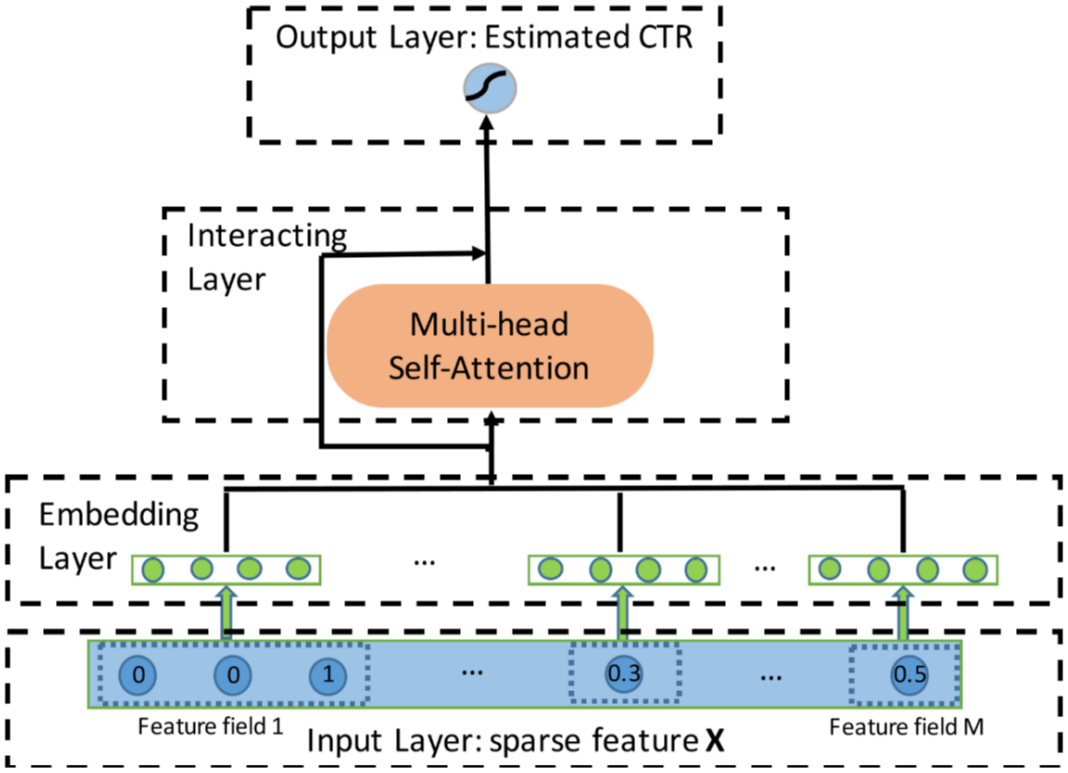

Figure 3.5.: AutoInt model as proposed by Song et al. (2019)

3.3. Experiment Setup

3.3.1. Datasets and Preprocessing

3.3.2. Evaluation Metrics

3.3.3. Hyperparameter Selection

3.4. Deep CTR Model Results

4. Deep Reinforcement Learning for Ad Personalization

4.1. DeepCTR-RL Framework

4.1.1. Model Framework

4.1.2. Feature types

4.1.3. Double Deep Q-Learning Network

4.1.4. Exploration

4.1.5. Experience Replay

4.2. Experiment Setup

4.2.1. Dataset and Preprocessing

4.2.2. Evaluation Metrics

4.2.3. Hyperparameter Selection

4.3. Deep CTR-RL Results

5. Discussion

Discussion goes here.

6. Conclusion

Conclusion goes here.

A. Appendix

A.1. Abbreviations and Acronyms

Term	Definition	Reference
LR	Logistic Regression	
FM	Factorization Machine	
FFM	Field-Aware Factorization Machine	
DNN	Deep Neural Network	
MLP	Multilayer Perceptron	2.1.3

A.2. Notation

Symbol	Definition	Reference
$\mathbf{x}$	Feature vector, before pre-processing	
n	the number of features in $\mathbf{x}$	
x_i	The i-th feature in $\mathbf{x}$	
$\mathbf{x_i}^{OH}$	One-hot encoded vector representation of categorical feature i	
$\mathbf{e}_i$	Embedded vector representation of categorical feature i	
z_i	Mean and variance standardized value for feature i from $\mathbf{x}$	
$\tilde{\mathbf{x}}$	$\mathbf{x}$ after catigorical embedding and numerical standardization.	
f	Pre-sigmoid classification function	
Θ	Parameter vector for f	

Bibliography

Yi Wang Aden. Kdd cup 2012, track 2, 2012. URL https://kaggle.com/competitions/kddcup2012-track2.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2008.

Djallel Bouneffouf, Amel Bouzeghoub, and Alda Lopes Gancarski. A contextual-bandit algorithm for mobile context-aware recommender system. In Tingwen Huang, Zhigang Zeng, Chu Li, ong, and Chi Sing Leung, editors, *Neural Information Processing*, pages 324–331, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-34487-9.

Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. Training and testing low-degree polynomial data mappings via linear svm. *Journal of Machine Learning Research*, 11(4), 2010.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 456–464, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359405. doi: 10.1145/3289600.3290999. URL https://doi.org/10.1145/3289600.3290999.

Chen Cheng, Fen Xia, Tong Zhang, Irwin King, and Michael R. Lyu. Gradient boosting factorization machines. In *Proceedings of the 8th ACM Conference on Recommender systems*, page 265–272, 2014.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, page 7–10, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450347952. doi: 10.1145/2988450.2988454. URL https://doi.org/10.1145/2988450.2988454.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals, and systems*, 2(4):303–314, 1989. doi: 10.1007/BF02551274.

eMarketer. Digital advertising spending worldwide from 2021 to 2027 (in billion u.s. dollars). Technical report, Statista Inc., 2023. URL https://www-statista-com.iclibezp1.cc.ic.ac.uk/statistics/237974/online-advertising-spe

Liqiong Gu. Ad click-through rate prediction: A survey. In Christian S. Jensen, Ee-Peng Lim, De-Nian Yang, Chia-Hui Chang, Jianliang Xu, Wen-Chih Peng, Jen-Wei Huang, and Chih-Ya Shen, editors, *Database Systems for Advanced Applications. DASFAA 2021 International Workshops*, pages 140–153, Cham, 2021. Springer International Publishing. ISBN 978-3-030-73216-5.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction. *CoRR*, abs/1703.04247, 2017. URL http://arxiv.org/abs/1703.04247. 1703.04247.

John T. Hancock and Taghi M. Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7(1):28, 2020. doi: 10.1186/s40537-020-00305-w. URL https://doi.org/10.1186/s40537-020-00305-w. ID: Hancock2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, page 770–778, 2016.

Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics, -08-16 2017.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. doi: 10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208. ID: 271125.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, 1990. doi: 10.1016/0893-6080(90)90005-6.

Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *10th ACM Conference on Recommender Systems*, RecSys '16, page 43–50, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340359. doi: 10.1145/2959100.2959134. URL https://doi.org/10.1145/2959100.2959134.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 661–670, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605587998. doi: 10.1145/1772690.1772758. URL https://doi.org/10.1145/1772690.1772758.

Zhongqi Lu and Qiang Yang. Partially observable markov decision process for recommender systems. *CoRR*, abs/1608.07793, 2016. URL http://arxiv.org/abs/1608.07793. 1608.07793.

Tariq Mahmood and Francesco Ricci. Learning and adaptivity in interactive recommender systems. In *Proceedings of the Ninth International Conference on Electronic Commerce*, ICEC '07, page 75–84, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937001. doi: 10.1145/1282100.1282114. URL https://doi.org/10.1145/1282100.1282114.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 26 2015. doi: 10.1038/nature14236. LR: 20220408; JID: 0410462; CIN: Nature. 2015 Feb 26;518(7540):486-7. doi: 10.1038/518486a. PMID: 25719660; 2014/07/10 00:00 [received]; 2015/01/16 00:00 [accepted]; 2015/02/27 06:00 [entrez]; 2015/02/27 06:00 [pubmed]; 2015/04/16 06:00 [medline]; AID: nature14236 [pii]; ppublish.

Ciara Pike-Burke. Optimism/thompson sampling, 2024.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Yanru Qu, Han Chai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1149–1154. IEEE, 2016. ISBN 2374-8486. doi: 10.1109/ICDM.2016.0151. ID: 1.

Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010. ISBN 1550-4786. doi: 10.1109/ICDM.2010.127. ID: 1.

Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *International Conference on World Wide Web*, WWW '07, page 521–530, New York, NY, USA, 2007. Association for Computing Machinery. doi: 10.1145/1242572.1242643. URL https://doi.org/10.1145/1242572.1242643.

Weichen Shen. Deepctr: Easy-to-use,modular and extendible package of deep-learning based ctr models, 2017. URL https://github.com/shenweichen/deepctr.

Song, Shi, Xiao, Duan, Xu, Zhang, and Tang. Autoint, -11-03 2019.

Jean-Baptiste Tien, joycenv, and Olivier Chapelle. Display advertising challenge, 2014. URL `https://kaggle.com/competitions/criteo-display-ad-challenge`.

Steve Wang and Will Cukierski. Click-through rate prediction, 2014. URL `https://kaggle.com/competitions/avazu-ctr-prediction`.

Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064–5078, 2024. doi: 10.1109/TNNLS.2022.3207346.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, 1989.

Kevin Webster. Week 2: Multilayer perceptron, 2024.

Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks *, 2017. URL `https://arxiv.org/abs/1708.04617`.

Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li. Online context-aware recommendation with time varying multi-armed bandit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 2025–2034, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939878. URL `https://doi.org/10.1145/2939672.2939878`.

Pengtao Zhang and Junlin Zhang. Memonet: Memorizing all cross features' representations efficiently via multi-hash codebook network for ctr prediction, -10-21 2023.

Weinan Zhang, Tianming Du, and Jun Wang. Deep learning over multi-field categorical data: A case study on user response prediction, 2016. URL `https://arxiv.org/abs/1601.02376`. 1601.02376.

Weinan Zhang, Jiarui Qin, Wei Guo, Ruiming Tang, and Xiuqiang He. Deep learning for click-through rate estimation, 21 Apr 2021. URL `https://arxiv.org/abs/2104.10584`.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *2018 World Wide Web Conference*, pages 167–176, Lyon, France, 2018. International World Wide Web Conferences Steering Committee. doi: 10.1145/3178876.3185994. URL `https://doi.org/10.1145/3178876.3185994`.