CS230

Deep Consumer Choice Models

Ayush Kanodia
Department of Computer Science
Stanford University
akanodia@stanford.edu

Sakshi Ganeriwal
Department of Computer Science
Stanford University
sakshig@stanford.edu

1 Problem Description and Motivation

We are using a novel dataset from Susan Athey's Lab in the Graduate School of Business at Stanford, which has approximately at least 10 GB of data on user shopping choice derived from multiple grocery stores in a large geographical region. The primary goal of the project is to set up appropriate Deep Neural Networks which learn how to predict consumer choice given consumer purchase history, as well as some observables of consumers and items as well as information such as price, time of day, day of week and so on.

Downstream Goals: A further goal of the project is to test how well we can optimize for counterfactuals – i.e. how well we predict outcomes when the test point is significantly different from what is observed in the train sample. While traditional Machine Learning theory dismisses counterfactual inference as not its goal, we borrow methods from Economics to test such counterfactual power of our models. A peripheral goal of the project could be to understand and interpret embeddings which our models learn for this setup. Any ability to measure econometric variables such as the price elasticity of demand, or a measure of which item pairs are substitutes or complements will also be interesting and valuable. See also [1]. These secondary goals are mentioned here for exposition, but not addressed in this paper.

2 Data Description

A typical snapshot of data contains complete billing information for a store for a day, with information such as transaction id, date, user id (from loyalty card logs), item id, price and quantity. We have partial user observables such as generic personal attributes, some information about items and information about their hierarchical classification into groups (typical examples are fruit, milk, detergent). There is more data we may use for further work. **Preprocessing**: Examples of preprocessing tasks are weeding out outlying users, items, splitting a huge transactions file into one file per store, setting up user and item ids, selecting categories with enough richness to model, calculating summary statistics and so on. Further details skipped.

2.1 Choice within category for a given trip

The unit of choice in our models is the item that a user purchased from a category on a given trip. All trips on a given date are combined into one trip, thus ensuring that there is at most one trip by a given customer on a given day. Modelling the purchase of every item individually is a very sparse prediction problem, with a lot of choices and very few purchases. [1] models user purchases in the general setup where a user may purchase multiple items from a category with substitutes. However, a natural modeling assumption to consider is that consumers make choices to purchase at most one item within a 'category', which is roughly a set of complete substitutes. Typical examples of categories would include 'apples', or 'fresh orange juice'. This is a testable assumption, and in our dataset, we note that for more than 90% categories, only about 1% of user trips involve purchases of multiple items in the same category, justifying this assumption strongly. This is the choice problem that we model. Further, we do not model the quantity of product purchased, but merely model which item a consumer chooses. See also [2] for a discussion on this. For this project, we model 14 categories selected such that they are available at several stores.

2.1.1 Model no purchase in category

We model no item purchased in category as another item within the category. Therefore, if a category has five items, we add a sixth item which denotes nothing purchased from within the category. This is referred to as the outside option in the decision analysis literature. Most customers do not purchase anything from a category on a visit, so this is the most purchased item for each user, as shown by the baseline models.

2.1.2 Model purchasing unpopular items in category

For items which are not purchased frequently, since we do not have enough data to model them correctly, we group them into an unpopular item for a every category. Continuing with the stylized example above. If there are five relatively popular items in a category and 15 other unpopular items, we introduce an "unpopular item" for every category in our model. Along with the outside option of purchasing nothing from a category, this leads to 7 items that we model over; the five relatively popular items, the outside option of not purchasing anything in this category, and the option to purchase one of the other 15 unpopular items.

3 Related work

[2] and [1] are to our knowledge, state of the art approaches to this problem. These combine Random Utility Models with modern Machine Learning Inference Techniques. We intend to park the limitations and test how effective deep neural networks can be on this task. In a past project [3], a similar attempt was made which concluded that much work remains to be done. [4] presents a method to separately model an intermediate outcome, which is then plugged back into traditional decision theory models. This is similar to the usage of secondary models in primary econometric models. Another example of that is seen in [5], where embeddings from an HPF model [6] are used as fixed effects in a utility choice model.
There have been adhoc attempts at using Neural Network methods for these problems, there is no clear baseline model to beat. However, in the informal setting, there are certain ideas speaking to how to go about doing this kind of modeling: [7] [8] [9].
A strongly related literature that we look at is the neural networks literature on embeddings. The central work in this direction that we draw upon is Neural Collaborative Filtering [10]. We describe how our models related to this work later. We further note that our models find similarity with the models in [11].

4 Challenges

Traditional recommendation algorithms are not well suited for this problem as they do not often address repeated choice of the same item, as well as do not address inventory information and the limited availability of items in any given session. [12] is a recent improvement with respect to exposure. They also do not model price very well. A line of investigation is whether adding structural Economic assumptions will help with modeling. Finally, to the best of our knowledge, there is no to little research on modeling supermarket consumer choice as a fusion of choice as well as embeddings using Neural Networks; we intend to therefore build a first model in this area.

5 Models

We first describe the dataset size and the input features we use. We then define two baseline models which predict user choice based on aggregate summary statistics. These give us an idea of a minimum baseline to beat. We move on to describing per category models, followed by joint modelling across categories. Finally, we describe a state of the art Machine Learning model which borrows methods from the decision theory literature in Economics and Industrial Organization, which we compare our models against.

5.0.1 Data size; Train, test and validation splits

We model 14 categories which are common across four stores, for which we source our data. For some of the models described below, we learn one model per category. There are a total of more than 1.2 million purchase decisions we model in our train set, and 400,000 purchase decisions in our validation and test set each. This comprises data from 4 stores in India from the same supermarket chain. User trips (entire trips) are randomized to be selected for train/test/validation, not a single purchase decision within a trip, since a trip is the unit of interest even though we model purchases over categories independent of each other given the model parameters. Currently, we use a 60 20 20 split between train, validation and test sets. The particularly high validation/test split is important because hyper parameter tuning using cross validation was too computationally expensive. See [2]. We can choose to let go of this restriction and find alternative solutions at a later stage.

5.0.2 Features

The primary goal of our models is to learn user and/or item embeddings which help inform prediction. Our problem is that we do not have rich observable covariates. Even so, the Recommendation Systems literature has shown that learning from history encodes information which cannot be encoded by the best models which learn based on observables. The approach of learning from choice history is known as collaborative filtering, while learning based on attributes (known) of users and items is known as content filtering. See [13] for a basic collaborative filtering model. See [14] for an exposition to models which unify collaborative and content filtering. Our model can be thought of as belonging to this class of models, except that our problem comes with repeated choice and more structure.
We use a demographic user feature which is whether the user is married. Even so, this information is missing for more than 90% of our users. Another feature we derive from the data is the tercile of every user for the number of items they purchase given a visit. This is to maintain parity with state of the art models. We do not use any item features; however modeling choice per category encodes category information much like a feature vector would. We use the following time features; month of year, week of year and day of week; these have been shown to work well (See [2] and [5]).

5.1 Baseline Models

We now describe the baseline models we compare against, based on aggregate statistics.

5.1.1 Baseline model using item frequency per category

This model sets the probability of purchase of an item as the fraction of times that item was purchased from that category across all users. It always predicts the same probabilities for all users, hence. It is the simplest baseline we try to better.

5.1.2 Baseline model item frequency per category per user

This model is identical to the above except the probabilities of purchase of items are calculated from purchase data per user instead of across all users. It is presumable that this model is slightly better than the first because it takes into account a user's identity. One pitfall with this is that certain user-item pairs may not be seen in the train set purchase history – we do add one laplace smoothing to fix this issue by setting the number of times an item was purchased by a user to be at least one.

5.2 Per Category Models

These methods learn a decision model *per category*, independent of other categories. These are all models with neural network architectures. We describe these models in the appendix as they have been described in the milestone evaluation. In summary, the models we consider are classical multinomial logit (MNL), multilayer MNL, unrestricted MNL, unrestricted MNL with one hidden layer and unrestricted MNL with two hidden layers.

5.3 Joint Models across categories

We use multi task learning to model choice across categories with shared parameters. This helps us demonstrate the utility of transferring knowledge across categories. Predicting multiple outcomes aims to improve the generalizability of the model, where our goal is to build the best-suited model for user choice prediction by using choice within categories as the different tasks. As a stylized example, if we understand that a user purchases relatively expensive organic milk, we should infer that when choosing from apples, his choice would be similar, relatively expensive apples. If we have low data for apples for this consumer, then in the category by category modeling setting, it would be harder for the model to infer this behaviour than in this setting. See [15] for an exposition on the benefit of learning embeddings from shared tasks in the Machine Learning setting for word embeddings. Here, the extension is for items and users in the supermarket choice setting, and we use deep neural networks to create these embeddings.

5.3.1 Multi Task Learning

In this model, the input features are as in 1. The prices are for all items across all categories. This forms the input layer. This is followed by two hidden layers. Finally, the output layer has 14 heads, one per category. Each is associated with a softmax loss for all items within the category.

5.3.2 Multi Task Learning with user onehot vectors.

In addition to the input features as in the previous model, in this model we provide a one hot indicator vector to the model informing it about which user is making the purchase.

5.3.3 Multi Task Learning with user embeddings

This is the crucial step in moving to embeddings. In this setup, we start with the input feature vector as in figure 1. Call this vector f_1. Call the user onehot vector f_2. From these are derived two hidden embeddings with the same dimensionality h_1 and h_2 by passing these two separately through a hidden layer each. In the fashion of [10], we also construct $h_3 = h_1.h_2$, where this is the element wise product. In addition, we use a fourth vector $h_4 = concat(f_1, f_2)$. Now consider the concatenated vector $v = concat(h_1, h_2, h_3, h_4)$. This is now our input vector which encodes per user per choice decision embeddings(details in figure 4). We did not find h_3 giving us performance improvements, interestingly. Note that this model does not have per item embeddings; item information is implicitly encoded in the multi task formulation. The downside of this approach is that we do not get interpretable item embeddings this way; however, there seems to be no compromise on performance since this is our **best performing model**. This is to be addressed in future work for which we have models designed but have not implemented in the interest of time.

5.4 State of the art comparison model : Bayesian embedding model

[2] and [5] present state of the art bayesian embedding models for the problem at hand. These infer latent user and item embeddings which are learnt in a shared fashion, along with Bayesian priors. We run these models for our problem in order to compare our models with these. We use two flavours of these models.

5.4.1 Single Level shared Multinomial Logit

In this model, users choose which item to purchase in a category along with the choice to not purchase anything simultaneously.

5.4.2 Two level shared Nested Logit

In this formulation (identical to that in [5], a user chooses, for each category, first whether to buy anything from the category, and given that they do purchase in category, what to purchase in category in a second step. This allows for a more natural structure of consumer choice; some of the

Model	Log Likelihood	Accuracy	Precision	Recall	F1
baseline_item_freq	-0.284942	0.917270	0.855519	0.917270	0.882245
baseline_user_item_freq	-0.282097	0.923744	0.902912	0.923744	0.908003
basic_mnl	-0.379898	0.917270	0.507557	0.534546	0.519109
multilayer_mnl	-0.328202	0.918198	0.519576	0.535474	0.525645
unrestricted_mnl	-0.264767	0.924011	0.689506	0.718780	0.698052
multilayer_mnl_two_layer	-0.245823	0.925648	0.691126	0.720417	0.705042
multilayer_mnl_one_layer	-0.240314	0.926139	0.701658	0.720907	0.705855
multitask_basic	-0.274243	0.918732	0.879776	0.918732	0.889628
multitask_user_onehot	-0.237934	0.926906	0.904628	0.926906	0.909831
multitask_embeddings	-0.236758	0.926886	0.904936	0.926886	0.910115
one level bayesian embeddings	-0.244352	0.925208	-	-	-
two level bayesian embeddings	-0.243596	-	-	-	-

Table 1: Test Set Results

specific consequences of this model specification are more realistic (see [16], chapters 3 and 4 for an exposition).

5.5 Hyperparameter Tuning

The main variable we tuned for most models is the number of hidden layers and the number of hidden units in each hidden layer. Typically, this was required for the joint models across categories, where we chose the number of hidden layers between 1 and 3, and the number of hidden units between 200 and 1000 in each layer. The optimal number of hidden units was different for different models; in general the optimal number of hidden layers is 2 and the number of hidden units is 600 - 1000 in each layer. Apart from this, we used L2 regularization to avoid overfitting, from which we saw huge gains on the validation set. We tried decay values of 10^{-4}, 10^{-5} and 10^{-6}, and achieved best performance across models with the weight 10^{-5}. Batch normalization hurt our modeling for the multi task models; we cannot pin point why; and dropout had the same effect as l2 regularization. We did not use Xavier initialization - that is an improvement we could work on.

6 Results

We report Log Likelihood, Precision, Recall and F1 scores for each of these models. The numbers are calculated per category, and weighted by number of purchases for each category. The results on the test set are shown in table 1. It is important to note that the basic and multilayer mnl models, as we have run them do not have item one hot vectors, whereas the unrestricted mnl with zero, one and two hidden layers, all have user one hot vectors as features. Some numbers are missing for comparison models as they were not calculated in the interest of time. Note that training set metrics are similar; the fits are systematically better on training sets which is expected; we omit those results due to lack of space.

7 Conclusion and Future Work

We note that the best performing model is the multi task model with user embeddings, in the style of [10] and [11]. We find uncanny resemblance between these two methods, our methods and a 15 year old method first demonstrated in [17]. We even see that our model beats state of the art embedding models as in [5], [1] and [2]. However, we note that there is no estimate of Bayes' optimal accuracy for this task. One limitation of our model is that item embeddings are not presently learnt by our models explicitly; structure over items is encoded in the multi task framing of our problem in a sense, and such information is presumably present in the models' hidden layers. In the future, we would like to further setup our models such that item embeddings are learnt explicitly. At this point, we believe that the way to do that would be to use an embedding model which then uses Siamese style loss [18] over utilities. This is left to future work.

References

[1] Francisco JR Ruiz, Susan Athey, and David M Blei. Shopper: A probabilistic model of consumer choice with substitutes and complements. *arXiv preprint arXiv:1711.03560*, 2017.

[2] Susan Athey, David Blei, Robert Donnelly, Francisco Ruiz, and Tobias Schmidt. Estimating heterogeneous consumer preferences for restaurants and travel time using mobile location data. In *AEA Papers and Proceedings*, volume 108, pages 64–67, 2018.

[3] Evan Munro. Deep learning models for restaurant choice. 2018.

[4] Brian Sifringer, Virginie Lurkin, and Alexandre Alahi. Enhancing discrete choice models with neural networks. *hEART*, 2018.

[5] Robert Donnelly, Francisco J. R. Ruiz, David M. Blei, and Susan Athey. Counterfactual inference for consumer choice across many product categories. *CoRR*, abs/1906.02635, 2019.

[6] Prem Gopalan, Jake M Hofman, and David M Blei. Scalable recommendation with hierarchical poisson factorization. In *UAI*, pages 326–335, 2015.

[7] Neural networks for your groceries. `https://towardsdatascience.com/neural-networks-for-your-groceries-f0a643eb411`. Accessed: 2019-02-20.

[8] Recnet: Deep learning based cross-class recommendations at wayfair. `https://tech.wayfair.com/data-science/2019/12/recnet-deep-learning-based-cross-class-recommendations-at-wayfair/`. Accessed: 2019-12-12.

[9] Traditional vs deep learning algorithms in retail industry. `https://towardsdatascience.com/traditional-vs-deep-learning-algorithms-in-retail-industry-i-b7b7f86793d4`. Accessed: 2019-03-27.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[11] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[12] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. Modeling user exposure in recommendation. In *Proceedings of the 25th international conference on World Wide Web*, pages 951–961, 2016.

[13] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[14] Justin Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*, page 9, 2004.

[15] Maja Rudolph, Francisco Ruiz, Susan Athey, and David Blei. Structured embedding models for grouped data. In *Advances in neural information processing systems*, pages 251–261, 2017.

[16] Daniel McFadden et al. Conditional logit analysis of qualitative choice behavior. 1973.

[17] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[18] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

8 Appendix

8.1 Category by Category model details

8.1.1 Multinomial Logit with no cross price effects

We start out with traditional multinomial logit, one per category. As [16] showed, the arguments of the softmax in multinomial logistic regression can be interpreted as representing consumer utility, under certain simple conditions. In the first setup, the multinomial logit formulation that we setup has the following form of utility.

$$U_{ijt} = \alpha_i D_j + \beta_i P_i + \gamma_i T_t + \epsilon_{ijt}$$

Where $Uijt$ is the utility for item i for user j at time instance t. D_j represents the user's covariates (of which we have only gender, age, and marital status, and these are often missing even upto 90 percent of the time). P_i is the price of item i; note that this utility does not include the prices of other items on which the consumer makes their choice. This is different from traditional multinomial logit as understood in the Machine Learning literature where the 'observables' are identically included to calculate utility for each alternative, and only the learnable cofficients (the Greek letters) differ over alternatives. T_t stands for time specific features (we use onehot encodings for day of week and month of year. ϵ is an unobservable shock variable that under multinomial logit is drawn from the Gumbel distribution [16].

8.1.2 Multinomial Logit with cross price effects

This is the most general form of multinomial logit we use, where cross price terms show up in utility. So if there are two items, we have both prices show up in both utilities.

$$U_{1jt} = \alpha_1 D_j + \beta_1 P_1 + \delta_1 P_2 \gamma_1 T_t + \epsilon_{1jt}$$
$$U_{2jt} = \alpha_2 D_j + \beta_2 P_1 + \delta_2 P_2 \gamma_2 T_t + \epsilon_{2jt}$$

.

8.1.3 Restricted Neural Network with one hidden layer

In the spirit of the first model, we use a Neural Network with one hidden layer to perform multi class classification, with 4 hidden units, understood as one for each item. We *keep* in this model the restriction that the price of other items in the category do not enter as inputs to the hidden layer for the corresponding item. While we understand that this is not a perfect restriction since in the output layer these individual price restricted terms do interact, we imagine this is a model in between unrestricted multilayer perceptron classification and our first model.

8.1.4 Unrestricted Neural Network with one hidden layer

This is a Neural Network with one hidden layer where prices of each item are inputs to every neuron in the first hidden layer. There is one hidden layer with 4 hidden units.

8.1.5 Unrestricted Neural Network with two hidden layers

This is a Neural Network like above, but with two hidden layers, each with four units. Note that there is one such model per category.

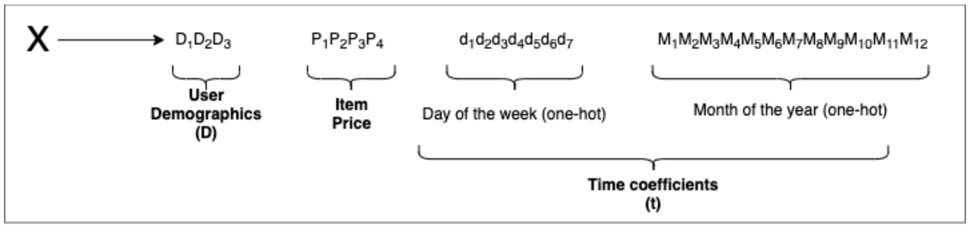
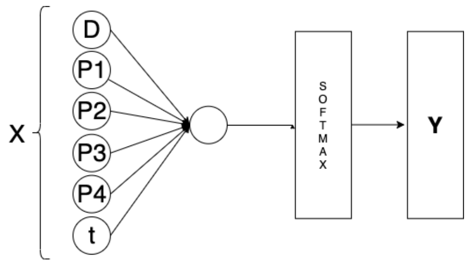


Figure 1: Input Feature Vector



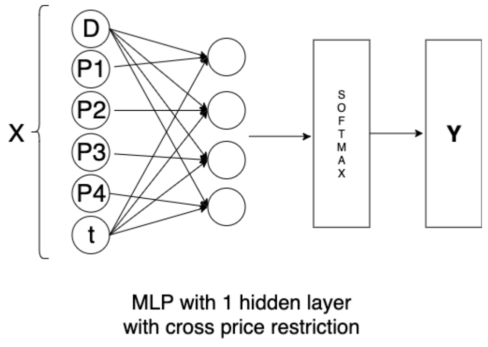
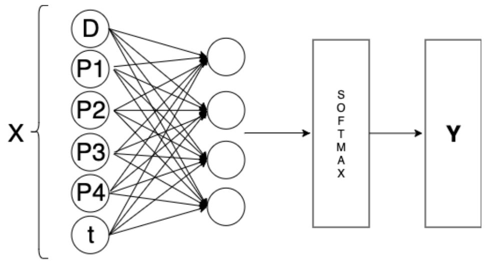


Figure 2: Models 2 and 3



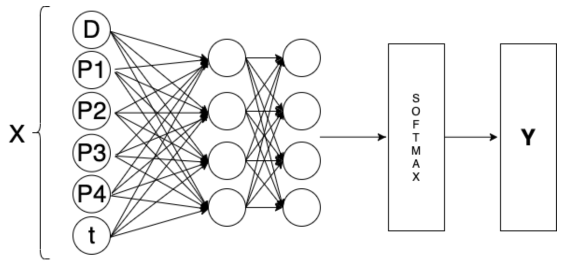


Figure 3: Models 4 and 5

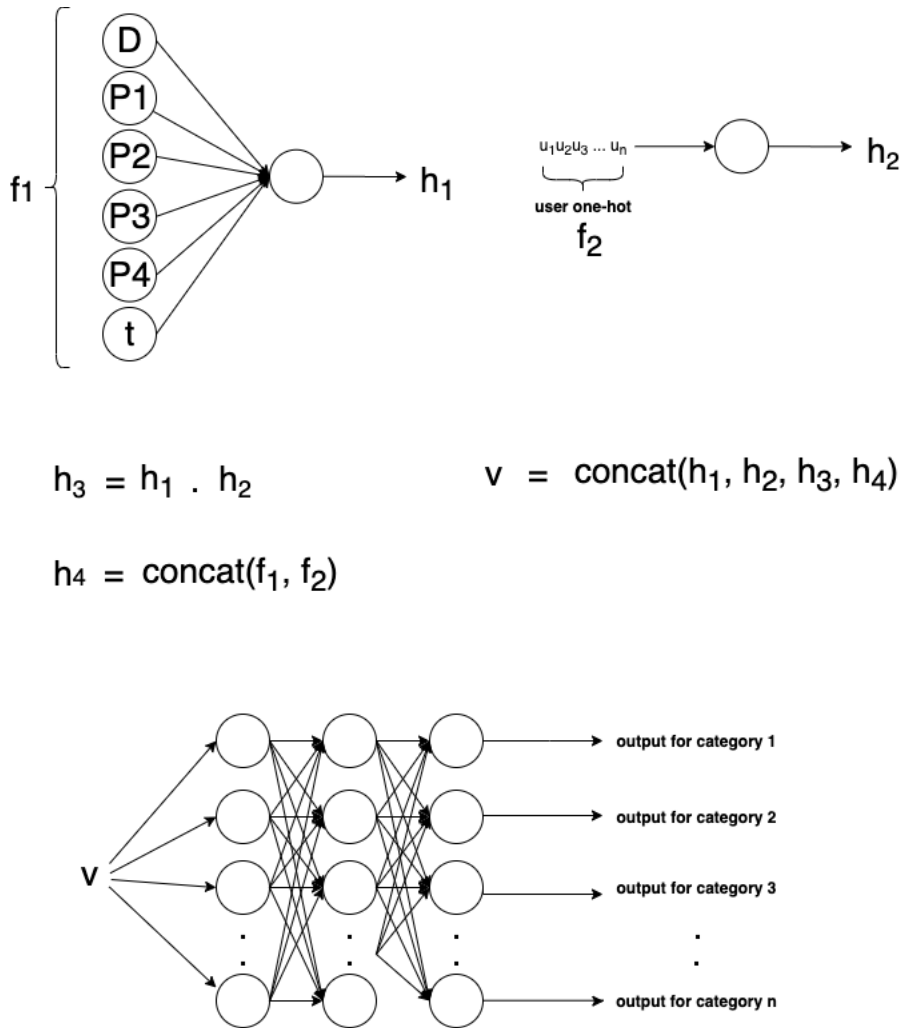


Figure 4: Multi-Task Learning with Embeddings