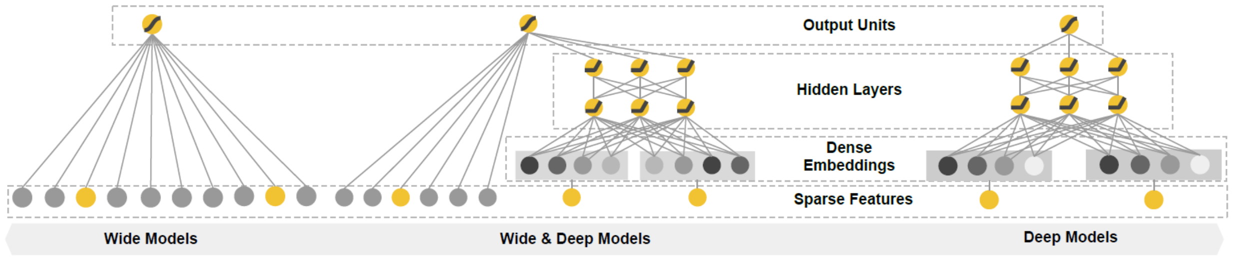
Wide & Deep Learning for Recommender Systems

Project Report Paper - CS795/895 Big Data and Recommender Systems – Raja Harsha Chinta

Old Dominion University



A Recommender system is a search ranking system where input query is a set of “User” & “Contextual Information”, and the output is a ranked list of items. Highly efficient general search ranking problems, is to achieve both memorization and generalization.

1 MODEL DISCUSSION

1.1 MEMORIZATION

Learning frequent co-occurrences of items or features and exploiting the correlation available in the historical data. Recommendations based on memorization are usually more topical and directly relevant to the items on which users have already performed actions. The wide component here is generalized linear model of form:

$$y = \mathbf{w}^T \mathbf{x} + b \qquad (1)$$

y is the prediction, x = [x1, x2, x3, ……., xd] is a vector of d features.

w is the model parameters and b is the bias.

Memorization can be achieved effectively using cross-product transformations over sparse features. Below equation denotes cross-product transformation:

$$\phi_k(\mathbf{x}) = \prod_{i=1}^{d} x_i^{c_{ki}} \quad c_{ki} \in \{0, 1\} \qquad (2)$$

where c_{ki} is Boolean variable and equal to 1 if the i-th feature is part of k-th transformation ϕ_k, 0 otherwise. This captures the interactions between binary features, adds nonlinearity.

One limitation of cross-product transformations is that they do not generalize to query-item feature pairs that have not appeared in the training data.

1.2 GENERALIZATION

Explores new feature combinations that are never or rarely occurred in the past based on transitivity of correlation. Generalization tends to improve the diversity of the recommended items. Generalization can be added by using features that are less granular. Using Deep Neural Networks we can generalize to previously unseen query-item feature pairs by learning a low-dimensional dense embedding vector. These low dimensional dense embedding vectors are then fed into the hidden layers of a neural network in the forward pass following assumption:

$$a^{(l+1)} = f(W^{(l)} a^{(l)} + b^{(l)}) \tag{3}$$

Here, l is the layer number and f is the ReLU activation function. a(l), b(l), and W(l) are the activations, bias, and model weights at l-th layer.

1.3 JOINING – MEMORIZATION & GENERALIZATION

Both the components discussed above are combined using a weighted sum of their output log odds as the prediction, which is then fed to one common logistic loss function for joint training. For a logistic regression problem, the model's prediction is given as follows:

$$P(Y = 1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^T[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^T a^{(l_f)} + b) \tag{4}$$

Where Y is the binary outcome label, $\sigma(\cdot)$ is the sigmoid function, $\phi(\mathbf{x})$ are the cross-product transformations of the original features x, and b is the bias term. W*wide* is the vector of all wide model weights, and W*deep* are the weights applied on the final activations $a^{(l_f)}$.

2 METHODS

Now that, we discussed the mathematical models used in Wide and Deep learning it would be intuitive if we take an example and explain how these methods contribute to individually and jointly. Let us take an example of a food ordering mobile app "FOODIO" which takes user inputs: "query" over speech and order relevant: "item**"** (not the same).

2.1 THE WIDE COMPONENT The wide component is a generalized linear model of form in equation (1). Y is predicted for various features along with X. The feature set includes raw input features and transformed input which are generated using equation (2).

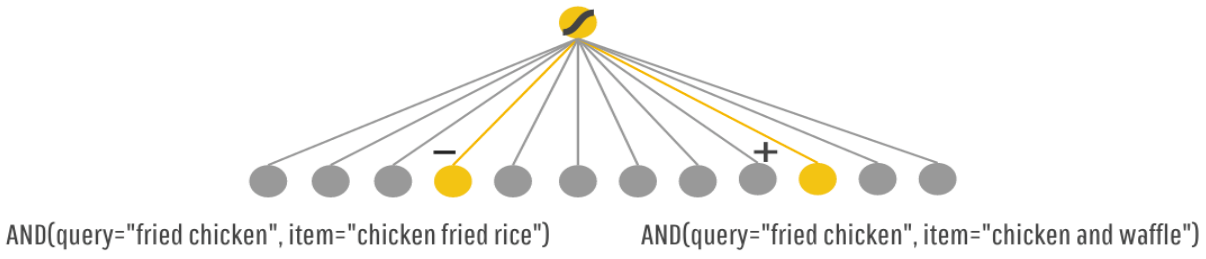

−
+
AND(query="fried chicken", item="chicken fried rice")
AND(query="fried chicken", item="chicken and waffle")

Figure 1: Example showing negative and positive "item" predictions for user input "query"

Here the user requested for a food suggestion related to query: "fried chicken" and the network is fed with raw feature and cross-transformed feature inputs to obtain below predictions.

- AND (query="fried chicken", item="chicken and waffles") is Liked by User,
- AND (query="fried chicken", item="chicken fried rice") is Not Liked by User even though the character match is higher.

2.2 THE DEEP COMPONENT

Is a feed-forward Neural Network. For categorical features, the original inputs are feature strings (e.g., "language=en"). Each of these sparse, high-dimensional categorical features are first converted into a low-dimensional and dense real-valued vector, embedding vector. The embedding vectors are initialized randomly and then the values are trained to minimize the final loss function during model training. These low-dimensional dense embedding vectors are then fed into the hidden layers of a neural network in the forward pass. l is the layer number and Activation Function f used here is ReLU.

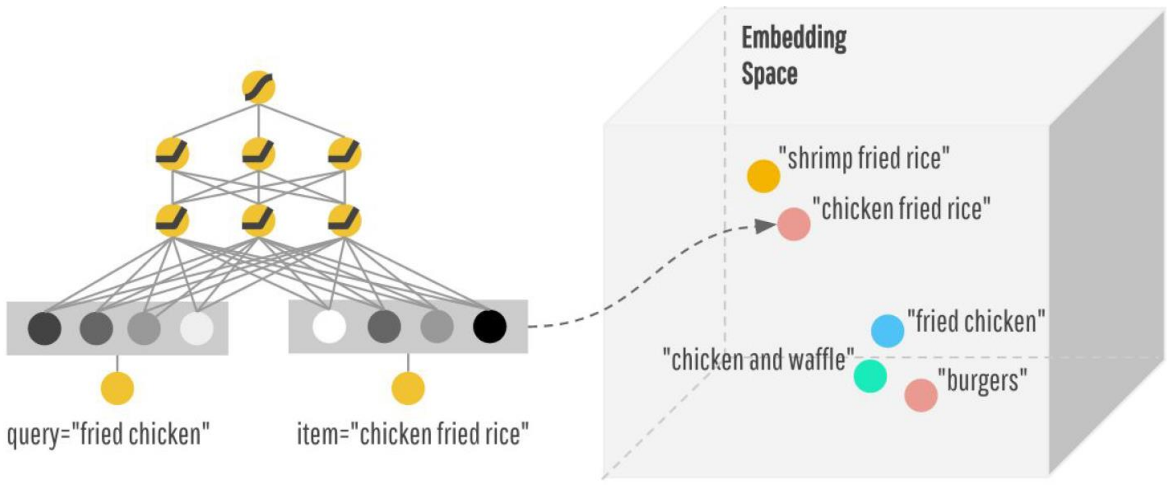


Figure 2: Example showing Deep Neural Network and low dimensional dense embedding vectors in space.

Here, the user asks for query: “fried chicken”. The Deep Neural Network understands the similarity of food from the embedded space and suggests “chicken and waffle” as an output. This is much better choice made understanding the context of food.

2.3 WIDE & DEEP MODEL

Joint training optimizes all parameters simultaneously by taking both the wide and deep part as well as the weights of their sum into account at training time. Joint training of a Wide & Deep Model is done by back propagating the gradients from the output to both the wide and deep part of the model simultaneously using mini-batch stochastic optimization. Follow-the-regularized-leader (FTRL) with L1 regularization as the optimizer for the wide part of the model, and AdaGrad for the deep part.

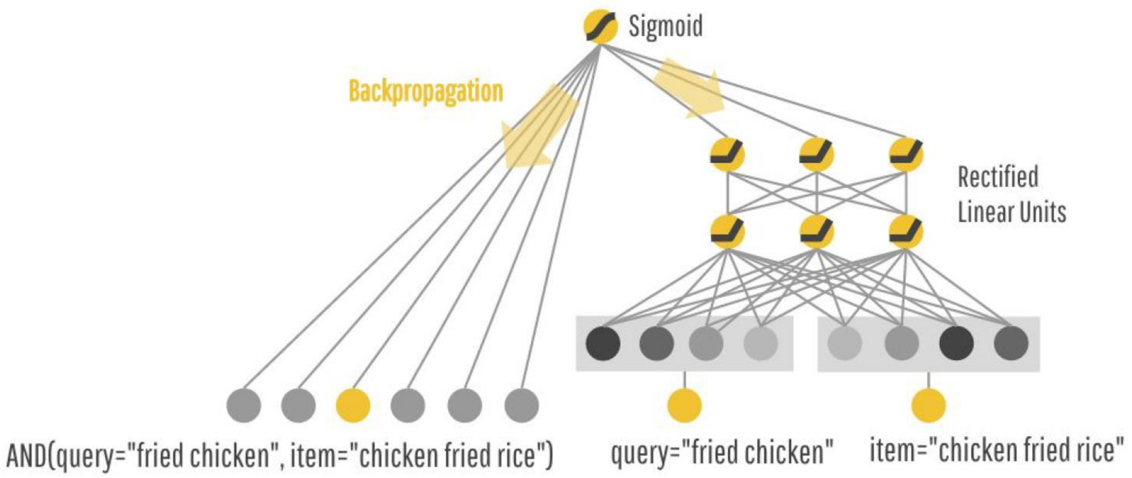

Sigmoid
Backpropagation
Rectified Linear Units
AND(query="fried chicken", item="chicken fried rice")
query="fried chicken"
item="chicken fried rice"

Figure 3: Example showing combination of Wide and Deep Neural Networks

3 PROBLEM DEFINITION

Given census data about a person such as age, gender, education and occupation (the features), we will try to predict whether or not the person earns more than 50,000 dollars a year (the target label). The author has released this scenario solved using Wide & Deep Learning Algorithm, I have re-implemented the same on my machine and in Google cloud for speedup. This notebook uses the tf.learn API in TensorFlow. Unfortunately, an example using the same for recommendation is not open sourced with the paper. Below table shows sample attribute information for our better understanding.

Attribute Information	
age	continuous.
workclass	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov
fnlwgt	continuous.
education	Bachelors, 11th, HS-grad, Prof-school, Masters, 10th, Doctorate, Preschool.
education-num	continuous.
marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, etc.
occupation	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, etc.
relationship	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex	Female, Male.
capital-gain	continuous.
capital-loss	continuous.
hours-per-week	continuous.
native-country	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, etc.

3.2 NEURAL NETWORK FLOW CHART

Figure 4 shows a layout of our neural network architecture we discussed. input layer takes in training data and vocabularies and generate sparse and dense features together with a label. The wide component consists of the cross-product transformation of user installed apps and impression apps. For the deep part of the model, A 32-dimensional embedding vector is learned for each categorical feature. We concatenate all the embeddings together with the dense features, resulting in a dense vector of approximately 1200 dimensions. The concatenated vector is then fed into 3 ReLU layers, and finally the logistic output unit.

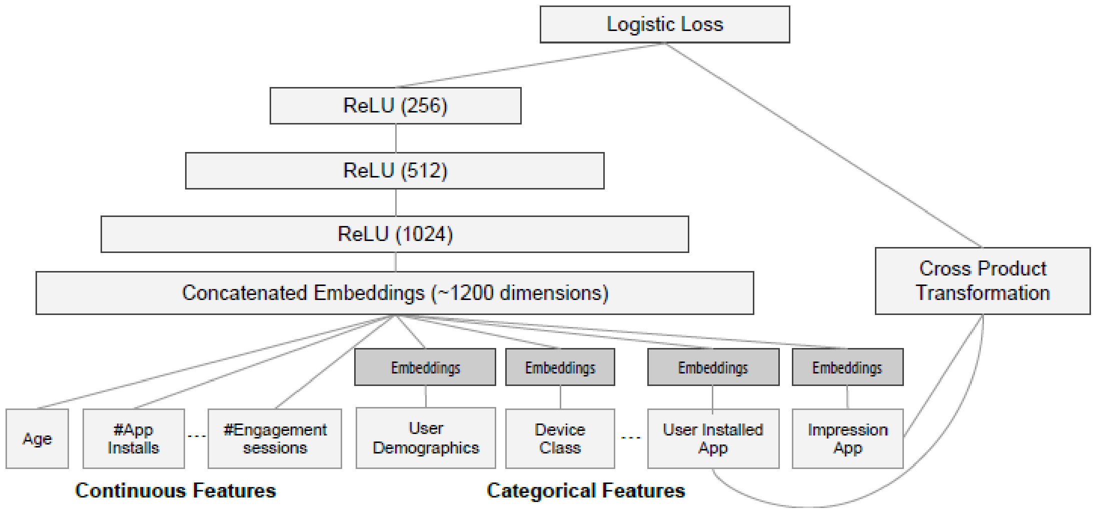


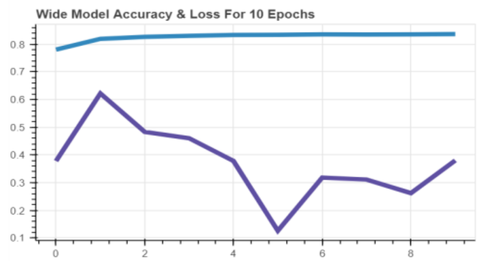
Figure 4: Example showing combination of Wide and Deep Neural Networks

4 RESULTS DISCUSSION

4.1 WIDE PART

The wide part consists simply in the sparse features connected directly to the output neuron (or neurons if the problem is a multiclass classification). In the example here, we will perform a logistic regression, so we need to connect the input features to an output neuron and use a *Sigmoid* activation function. Below is the structure and statistics for 10 epochs.

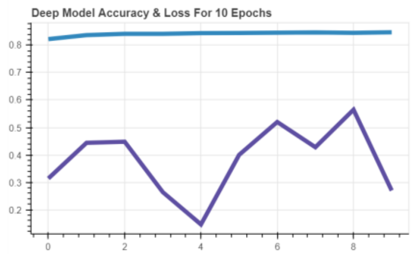
```
Epoch 1 of 10, Loss: 0.378, accuracy: 0.7799
Epoch 2 of 10, Loss: 0.622, accuracy: 0.819
Epoch 3 of 10, Loss: 0.483, accuracy: 0.8264
Epoch 4 of 10, Loss: 0.46, accuracy: 0.83
Epoch 5 of 10, Loss: 0.378, accuracy: 0.8329
Epoch 6 of 10, Loss: 0.126, accuracy: 0.8337
Epoch 7 of 10, Loss: 0.318, accuracy: 0.8354
Epoch 8 of 10, Loss: 0.311, accuracy: 0.8346
Epoch 9 of 10, Loss: 0.262, accuracy: 0.8354
Epoch 10 of 10, Loss: 0.38, accuracy: 0.8361
```



4.2 DEEP PART

The deep part implemented here will be comprised by two layers of 100 and 50 neurons and receives embeddings and can also receive numerical features. The Deep part comprises of below structure. 5 embedding layers of dimensions 10, 10, 10, 8 and 10 respectively. The 5 embedding layers will be concatenated with the two continuous features (age and hours per week). Two hidden layers of 100 and 50 neurons. The output neuron with a Sigmoid activation function.

```
Epoch 1 of 10, Loss: 0.314, accuracy: 0.8225
Epoch 2 of 10, Loss: 0.444, accuracy: 0.8373
Epoch 3 of 10, Loss: 0.448, accuracy: 0.8417
Epoch 4 of 10, Loss: 0.264, accuracy: 0.8418
Epoch 5 of 10, Loss: 0.146, accuracy: 0.8443
Epoch 6 of 10, Loss: 0.4, accuracy: 0.8449
Epoch 7 of 10, Loss: 0.52, accuracy: 0.8459
Epoch 8 of 10, Loss: 0.428, accuracy: 0.8472
Epoch 9 of 10, Loss: 0.565, accuracy: 0.8453
Epoch 10 of 10, Loss: 0.27, accuracy: 0.8474
```



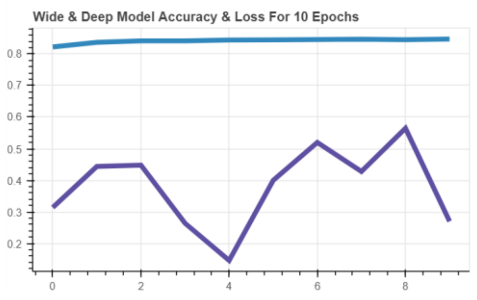
4.3 WIDE & DEEP PART

The output neuron receives now the wide and the deep side, so the input dimensions need to be adapted in both the definition of the model. The output layer, which now receives all the features from the wide side plus the 50 dense features from the deep side. Below is the final model structure.

```
In [45]: wide_deep_model

Out[45]: WideDeep (
           (emb_layer_workclass): Embedding(9, 10)
           (emb_layer_education): Embedding(16, 10)
           (emb_layer_native_country): Embedding(42, 10)
           (emb_layer_relationship): Embedding(6, 8)
           (emb_layer_occupation): Embedding(15, 10)
           (linear_1): Linear (50 -> 100)
           (linear_2): Linear (100 -> 50)
           (output): Linear (848 -> 1)
         )
```

```
Epoch 1 of 10, Loss: 0.502, accuracy: 0.824
Epoch 2 of 10, Loss: 0.472, accuracy: 0.8377
Epoch 3 of 10, Loss: 0.463, accuracy: 0.8405
Epoch 4 of 10, Loss: 0.096, accuracy: 0.8418
Epoch 5 of 10, Loss: 0.454, accuracy: 0.8426
Epoch 6 of 10, Loss: 0.235, accuracy: 0.8441
Epoch 7 of 10, Loss: 0.302, accuracy: 0.8452
Epoch 8 of 10, Loss: 0.342, accuracy: 0.845
Epoch 9 of 10, Loss: 0.405, accuracy: 0.8453
Epoch 10 of 10, Loss: 0.23, accuracy: 0.8477
```



5 CONCLUSION

Overall, Wide Linear models are effective in memorizing the sparse feature interactions and DNN can generalize previously unseen features interactions. The idea of wide linear model part complementing deep learning here is an excellent idea nice. The above results illustrate the implementation of Wide & Deep Neural Networks. I understand there is a scope for improving the accuracy especially with Deep part after careful hyperparameter tuning.

Thank you for reviewing the report.

REFERENCES

https://arxiv.org/abs/1606.07792 - Paper

https://www.kdnuggets.com/2017/08/recommendation-system-algorithms-overview.html

https://archive.ics.uci.edu/ml/datasets/Census+Income

https://research.googleblog.com/2016/06/wide-deep-learning-better-together-with.html

https://medium.com/@libreai/a-glimpse-into-deep-learning-for-recommender-systems-d66ae0681775

https://github.com/yufengg/widendeep/blob/master/wnd_criteo.ipynb