Automatic FE with Tensor Flow

Omid Safarzadeh

February 2, 2022

Table of contents

Multiple linear regression with interaction

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \varepsilon$$

interaction term

Challenges in learning cross Features

In Web-scale applications, data are mostly categorical.
For Example : City / Neighborhood names $>>$ one-hot encoding $>>$ large and sparse feature space.
feature crosses in this setting often requires manual feature engineering or exhaustive search.

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets a 1x9 vector representation

Automatic FE with TensorFlow

- Deep and Cross Network (Wang et al., 2017, Wang et al., 2021) is a novel cross network that explicitly applies feature crossing at each layer, efficiently learns predictive cross features, and requires no manual feature engineering.
- DCN considers highest polynomial degree. The network consists of all the cross terms of degree up to the highest, with their coefficients all different.

Deep & Cross Network Structure

DCN model Structure:

input layer $>>$ (typically an embedding layer),

a cross network containing multiple cross layers $>>$ explicit feature interactions

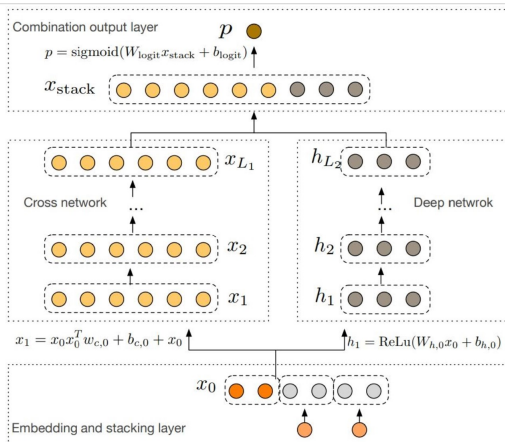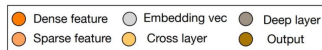deep nn network $>>$ implicit feature interactions.

Deep Network. It is a traditional feed forward multi layer perceptron (MLP).

The deep network and cross network are then combined to form DCN. Commonly, we could stack a deep network on top of the cross network. We could also place them in parallel (parallel structure)
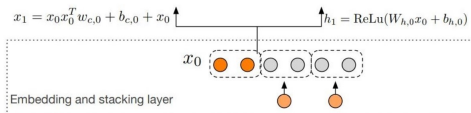
Deep & Cross Network V1

Ruoxi Wang et al.
Stanford University, Google 2017

Combination output layer p

$p = \text{sigmoid}(W_{\text{logit}} x_{\text{stack}} + b_{\text{logit}})$

x_{stack}

Cross network x_{L_1} h_{L_2} Deep netwrok

x_2 h_2

x_1 h_1

$x_1 = x_0 x_0^T w_{c,0} + b_{c,0} + x_0$ $h_1 = \text{ReLu}(W_{h,0} x_0 + b_{h,0})$

x_0

Embedding and stacking layer

- Dense feature
- Embedding vec
- Deep layer
- Sparse feature
- Cross layer
- Output

Input preparation

$$x_1 = x_0 x_0^T w_{c,0} + b_{c,0} + x_0$$

$$h_1 = \text{ReLu}(W_{h,0} x_0 + b_{h,0})$$

x_0

Embedding and stacking layer
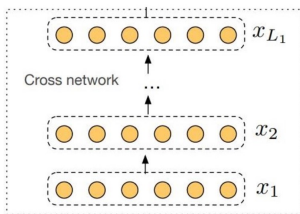
```
# Embedding and stacking layer
dense_inputs, sparse_inputs = inputs
sparse_embed = tf.concat([self.embed_layers['embed_{}'.format(i)](sparse_inputs[:, i])
                          for i in range(sparse_inputs.shape[1])], axis=-1)
x = tf.concat([sparse_embed, dense_inputs], axis=-1)
```

Cross Network

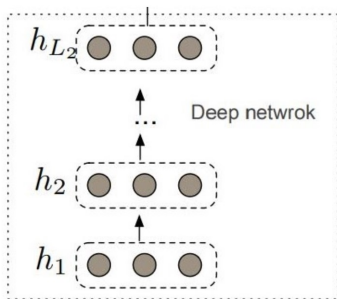
Cross network

x_{L_1}

...

x_2

x_1

```python
# Cross network
x_0 = tf.expand_dims(inputs, axis=2)  # (None, dim, 1)
x_l = x_0  # (None, dim, 1)

for i in range(self.layer_num):
    x_l1 = tf.tensordot(x_l, self.cross_weights[i], axes=[1, 0])  # (None, dim, dim)
    x_l = tf.matmul(x_0, x_l1) + self.cross_bias[i] + x_l  # (None, dim, 1)

x_l = tf.squeeze(x_l, axis=2)  # (None, dim)
```

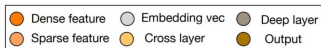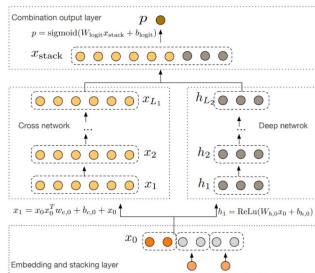$$x_{l+1} = x_0 x_l^T w_l + b_l + x_l = f(x_l, w_l, b_l) + x_l$$

Deep Neural Network


```python
# Deep Network
x = inputs
for dnn in self.dnn_network:
    x = dnn(x)
x = self.dropout(x)
```

$$h_{l+1} = f(W_l h_l + b_l)$$

Deep & Cross Network

```python
# Deep and Cross Network
def call(self, inputs):
    dense_inputs, sparse_inputs = inputs
    sparse_embed = tf.concat([self.embed_layers['embed_{}'.format(i)](sparse_inputs[:,i])
                        for i in range(sparse_inputs.shape[1])], axis=-1)
    x = tf.concat([sparse_embed, dense_inputs], axis=-1)

    # Cross Network
    cross_x = self.cross_network(x)

    # DNN
    dnn_x = self.dnn_network(x)

    # Concatenate
    total_x = tf.concat([cross_x, dnn_x], axis=-1)
    outputs = tf.nn.softmax(self.dense_final(total_x))

    return outputs

def summary(self):
    dense_inputs = Input(shape=(len(self.dense_feature_columns),),dtype=tf.float32)
    sparse_inputs = Input(shape=(len(self.sparse_feature_columns),), dtype=tf.int32)
    keras.Model(inputs=[dense_inputs,sparse_inputs],
                outputs=self.call([dense_inputs, sparse_inputs])).summary()
```

```python
user_model = DCN(hidden_units, dnn_dropout=dnn_dropout)

#========================Compile==========================

user_model.compile(loss=categorical_crossentropy, optimizer=Adam(learning_rate=learning_rate),
                   metrics=[tfa.metrics.F1Score(num_classes = 3)])
#==========================Fit============================
user_model.fit(
    train_X,
    train_y,
    epochs=epochs,
    batch_size=batch_size,
    validation_split=0.1
)
```

```
Model: "sample model"
_____
Layer (type)                    Output Shape        Param #    Connected to
==========================================================================
input_2 (InputLayer)            [(None, 3)]         0
_____
tf.__operators__.getitem (Slici (None,)            0          input_2[0][0]
_____
tf.__operators__.getitem_1 (Sli (None,)            0          input_2[0][0]
_____
tf.__operators__.getitem_2 (Sli (None,)            0          input_2[0][0]
_____
embedding (Embedding)           (None, 32)          192        tf.__operators__.getitem[0][0]
_____
embedding_1 (Embedding)         (None, 32)          64         tf.__operators__.getitem_1[0][0]
_____
embedding_2 (Embedding)         (None, 32)          64         tf.__operators__.getitem_2[0][0]
_____
tf.concat (TFOpLambda)          (None, 96)          0          embedding[0][0]
                                                               embedding_1[0][0]
                                                               embedding_2[0][0]
_____
input_1 (InputLayer)            [(None, 3)]         0
_____
tf.concat_1 (TFOpLambda)        (None, 99)          0          tf.concat[0][0]
                                                               input_1[0][0]
_____
cross_network (CrossNetwork)    (None, 99)          594        tf.concat_1[0][0]
_____
dnn (DNN)                       (None, 4)           1772       tf.concat_1[0][0]
_____
tf.concat_2 (TFOpLambda)        (None, 103)         0          cross_network[0][0]
                                                               dnn[0][0]
_____
dense_3 (Dense)                 (None, 3)           312        tf.concat_2[0][0]
_____
tf.nn.softmax (TFOpLambda)      (None, 3)           0          dense_3[0][0]
==========================================================================
Total params: 2,998
Trainable params: 2,998
Non-trainable params: 0
```

Deep & Cross Network V2
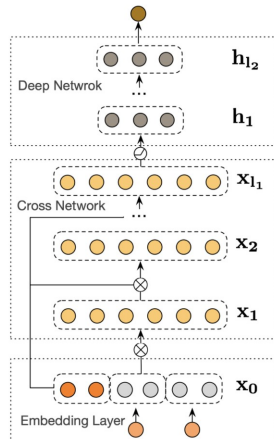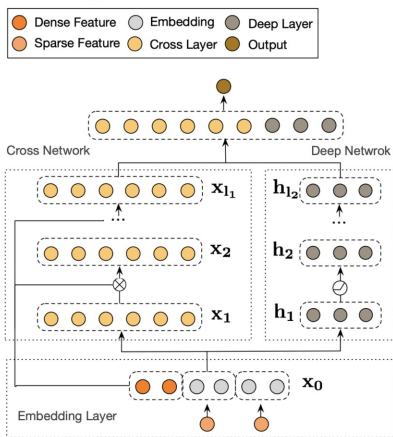
- **Cross Network**. This is the core of DCN. It explicitly applies feature crossing at each layer, and the highest polynomial degree increases with layer depth. The following figure shows the (i+1)-th cross layer.

$$x_{i+1} = x_0 \odot (W \times x_i + b) + x_i$$
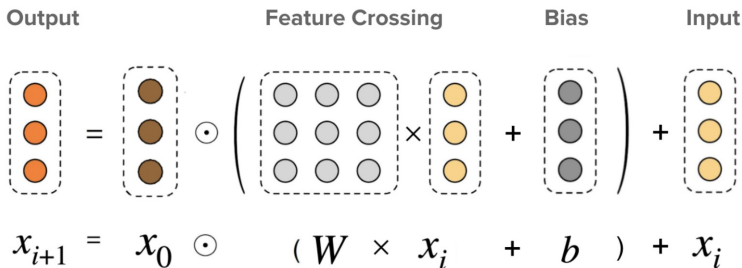
Model construction

data only contains 2nd-order feature interactions,
to model higher-order feature interactions, $>>>$ stack multiple cross layers and
use a multi-layered cross network.
The two models we will be building are:

- Cross Network with only one cross layer;
- Deep Network with wider and deeper ReLU layers.

Model understanding for interpreting cross features

The weight matrix in DCN reveals what feature crosses the model has learned to be important.

The importance of interactions between the i-th and j-th features is captured by the (i,j)-th element of.

The feature embeddings are of size 32 instead of size 1.

Model Performance

The list of the best performance of different models in logloss.
Deep Crossing (DC)
Deep Neural Network (DNN)
Factorization machines (FM)
logistic regression (LR)

Model	DCN	DC	DNN	FM	LR
Logloss	**0.4419**	0.4425	0.4428	0.4464	0.4474

Comparisons Between DCN and DNN

Considering that the cross network only introduces O(d) extra parameters, experimental results while varying memory budget and loss tolerance
number of parameters

Logloss	0.4430	0.4460	0.4470	0.4480
DNN	3.2×10^6	1.5×10^5	1.5×10^5	7.8×10^4
DCN	7.9×10^5	7.3×10^4	3.7×10^4	3.7×10^4

Reference

Wang, R., Fu, B., Fu, G., & Wang, M. (2017). Deep & cross network for ad click predictions. *Proceedings of the adkdd'17* (pp. 1–7).

Wang, R., Shivanna, R., Cheng, D., Jain, S., Lin, D., Hong, L., & Chi, E. (2021). Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. *Proceedings of the Web Conference 2021*, 1785–1797.