Xiaoxu Qi

Wide & Deep Learning: Practice on Tensorflow

Dec. 29th, 2017

Word2vec: ML with softmax

$$J_{\mathrm{ML}} = \log P(w_t|h)$$
$$= \mathrm{score}(w_t, h) - \log\left(\sum_{\text{Word w' in Vocab}} \exp\{\mathrm{score}(w', h)\}\right).$$

- h: history also as context
- w: target word predicted after history
- Ref: A Neural Probabilistic Language Model, Journal of Machine Learning Research 3 (2003) 1137–1155

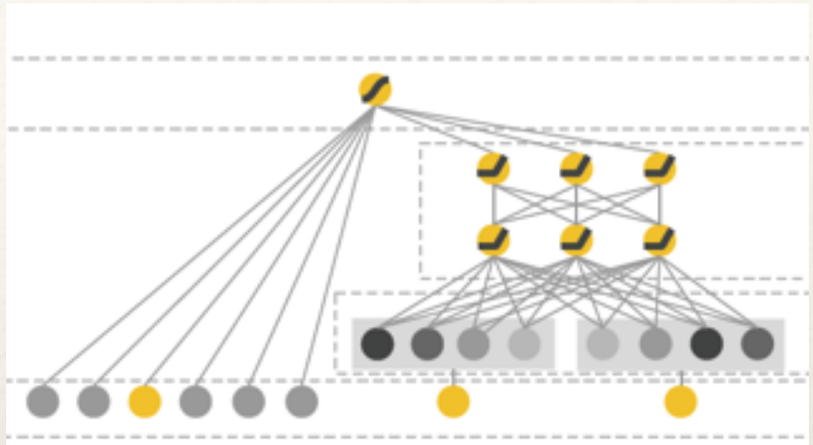
Note on wide & deep learning

- Ref: Wide & Deep Learning for Recommender Systems
- link: https://www.tensorflow.org/tutorials/wide_and_deep

Note on wide & deep learning

- FFM or deep neural network introduces over generalisation with dense embeddings which leads to non-zero predict for all ads(impression).
- Over generalisation makes the recommendation less diversity. Narrow appealing items or users with specific preferences.
- Linear models: memory the “specifics”.

Join training



- Different from model assemble.
- The wide parts with interaction features deal with diversity, or personalised.
- The deep parts deal with generalisation.

Implement on Tensorflow

- Graphs and sessions: https://www.tensorflow.org/versions/r1.3/programmers_guide/graphs
- Graphs: management of ops and tensors
- Sessions: management of cal sources within with Graphs are actually evaled.

Import data from cvs

```
def input_fn(data_file, buffer_size, epochs_per_eval, batch_size):
    def parse_csv(value):
        print('Parsing', data_file)
        columns = tf.decode_csv(value, record_defaults=_CSV_COLUMN_DEFAULTS)
        features = dict(zip(_CSV_COLUMNS, columns))
        labels = features.pop('label')
        return features, tf.equal(labels, '1.0')
    dataset =
tf.data.TextLineDataset(data_file).map(parse_csv).shuffle(buffer_size).repeat(epochs_per_eval).ba
tch(batch_size)
    iterator = dataset.make_one_shot_iterator()
    features, labels = iterator.get_next()
    return features, labels
```

- cvs decoder: https://www.tensorflow.org/api_docs/python/tf/decode_csv

OHE - Categorical Features

```
u_gender = tf.feature_column.categorical_column_with_vocabulary_list(
    'u_gender', ['0', '1', '2'])
```

- Wide & Deep NN only takes _denseColumns.
- tf.feature_column.indicator_column
- Represents multi-hot representation of given categorical column.
- Used to wrap any categorical_column_* (e.g., to feed to DNN). Use embedding_column if the inputs are sparse.
- Ref: https://www.tensorflow.org/api_docs/python/tf/feature_column/indicator_column

Wide and Deep columns

```
base_columns = [u_gender, u_age, u_city, a_category, a_city, c1, c2, c3, c4, c5, c6, c7, c8, c9,
c10, c11, c12, c13, c14, c15, c16, c17, c18, c19, c20, c21, c22]
crossed_columns = [
    tf.feature_column.crossed_column(['u_city', 'a_category'], hash_bucket_size=50660),
    tf.feature_column.crossed_column(['u_age', 'a_category'], hash_bucket_size=100),
    tf.feature_column.crossed_column(['u_gender', 'a_category'], hash_bucket_size=20), ]
wide_columns = base_columns + crossed_columns
deep_columns = [tf.feature_column.indicator_column(u_gender),
tf.feature_column.indicator_column(u_age), tf.feature_column.indicator_column(u_city),
tf.feature_column.indicator_column(a_category), tf.feature_column.indicator_column(a_city)]
```