
tensorflow-DeepFM

This project includes a Tensorflow implementation of DeepFM [1].

NEWS

- A modified version of DeepFM is used to win the 4th Place for Mercari Price Suggestion Challenge on Kaggle. See the slide here how we deal with fields containing sequences, how we incorporate various FM components into deep model.

Usage

Input Format

This implementation requires the input data in the following format: - [] **\mathbf{Xi}** : $[[ind1_1, ind1_2, \dots], [ind2_1, ind2_2, \dots], \dots, [indi_1, indi_2, \dots, indi_j, \dots], \dots]$ - $indi_j$ is the feature index of feature field j of sample i in the dataset - [] **\mathbf{Xv}** : $[[val1_1, val1_2, \dots], [val2_1, val2_2, \dots], \dots, [vali_1, vali_2, \dots, vali_j, \dots], \dots]$ - $vali_j$ is the feature value of feature field j of sample i in the dataset - $vali_j$ can be either binary (1/0, for binary/categorical features) or float (e.g., 10.24, for numerical features) - [] **\mathbf{y}** : target of each sample in the dataset (1/0 for classification, numeric number for regression)

Please see [example/DataReader.py](#) an example how to prepare the data in required format for DeepFM.

Init and train a model

```
1 import tensorflow as tf
2 from sklearn.metrics import roc_auc_score
3
4 # params
5 dfm_params = {
6     "use_fm": True,
7     "use_deep": True,
8     "embedding_size": 8,
9     "dropout_fm": [1.0, 1.0],
10    "deep_layers": [32, 32],
11    "dropout_deep": [0.5, 0.5, 0.5],
12    "deep_layers_activation": tf.nn.relu,
13    "epoch": 30,
14    "batch_size": 1024,
```

```
15     "learning_rate": 0.001,
16     "optimizer_type": "adam",
17     "batch_norm": 1,
18     "batch_norm_decay": 0.995,
19     "l2_reg": 0.01,
20     "verbose": True,
21     "eval_metric": roc_auc_score,
22     "random_seed": 2017
23 }
24
25 # prepare training and validation data in the required format
26 Xi_train, Xv_train, y_train = prepare(...)
27 Xi_valid, Xv_valid, y_valid = prepare(...)
28
29 # init a DeepFM model
30 dfm = DeepFM(**dfm_params)
31
32 # fit a DeepFM model
33 dfm.fit(Xi_train, Xv_train, y_train)
34
35 # make prediction
36 dfm.predict(Xi_valid, Xv_valid)
37
38 # evaluate a trained model
39 dfm.evaluate(Xi_valid, Xv_valid, y_valid)
```

You can use `early_stopping` in the training as follow

```
1 dfm.fit(Xi_train, Xv_train, y_train, Xi_valid, Xv_valid, y_valid,
        early_stopping=True)
```

You can refit the model on the whole training and validation set as follow

```
1 dfm.fit(Xi_train, Xv_train, y_train, Xi_valid, Xv_valid, y_valid,
        early_stopping=True, refit=True)
```

You can use the FM or DNN part only by setting the parameter `use_fm` or `use_dnn` to `False`.

Regression

This implementation also supports regression task. To use DeepFM for regression, you can set `loss_type` as `mse`. Accordingly, you should use `eval_metric` for regression, e.g., `mse` or `mae`.

Example

Folder [example](#) includes an example usage of DeepFM/FM/DNN models for Porto Seguro's Safe Driver Prediction competition on Kaggle.

Please download the data from the competition website and put them into the [example/data](#) folder.

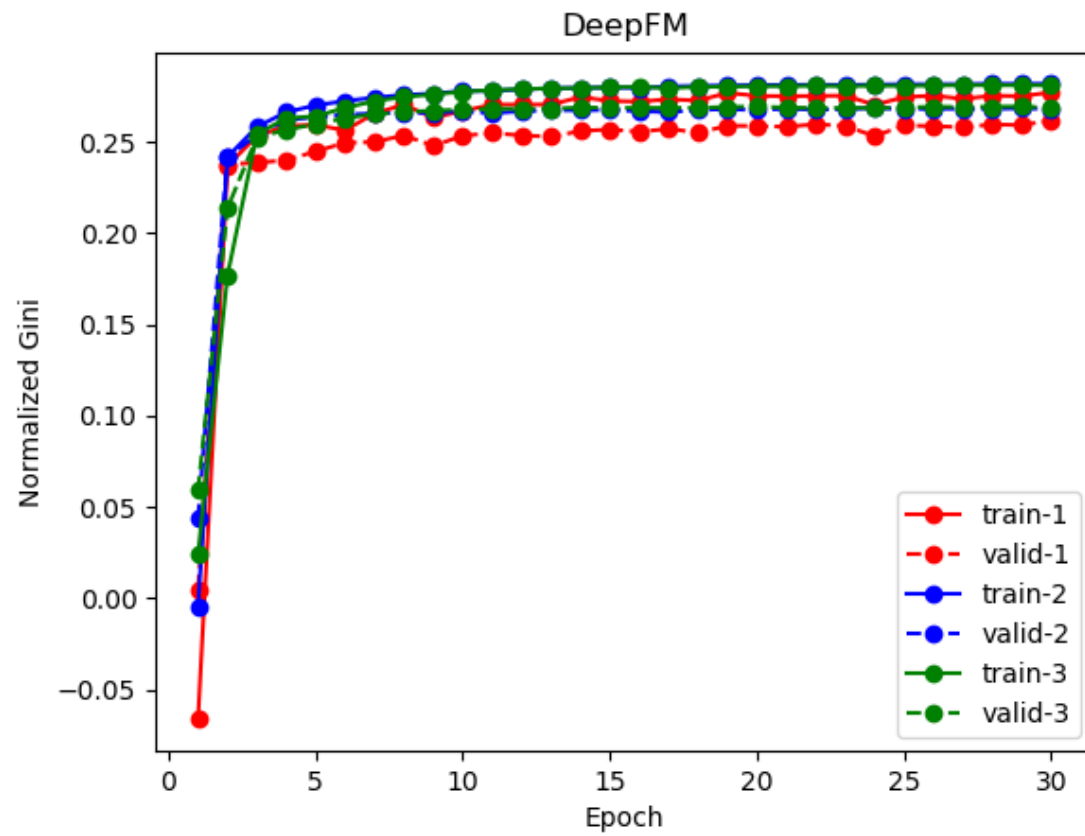
To train DeepFM model for this dataset, run

```
1 $ cd example
2 $ python main.py
```

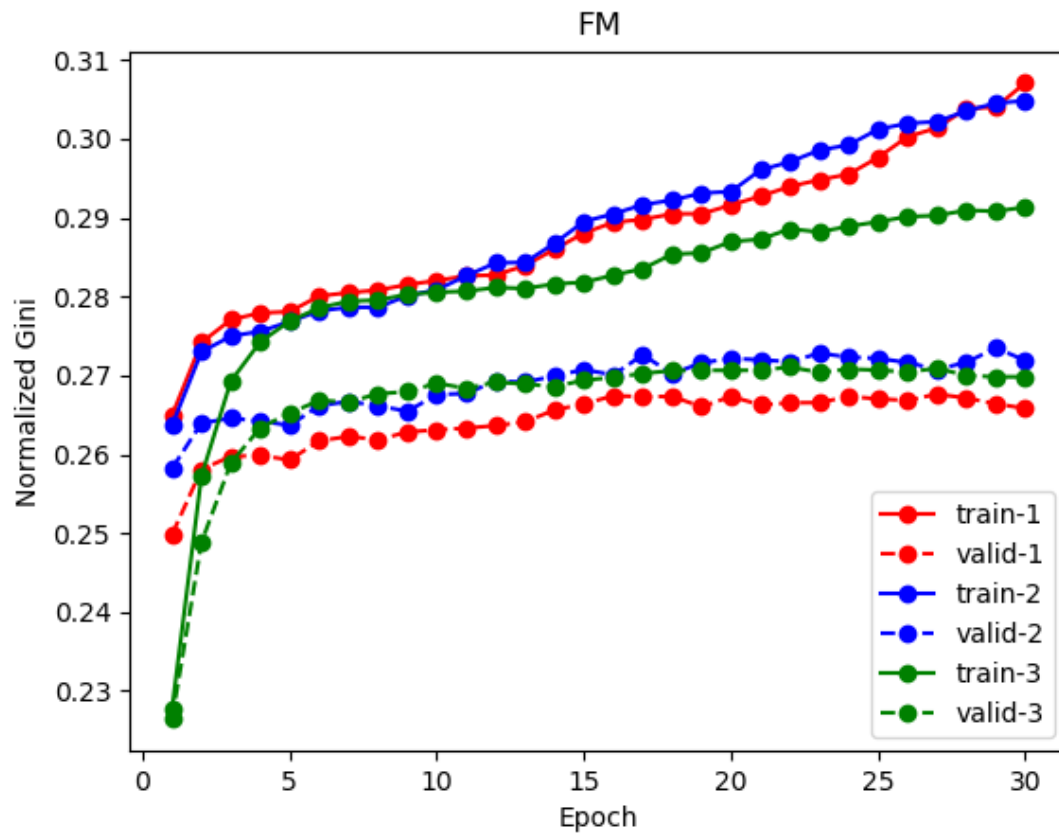
Please see [example/DataReader.py](#) how to parse the raw dataset into the required format for DeepFM.

Performance

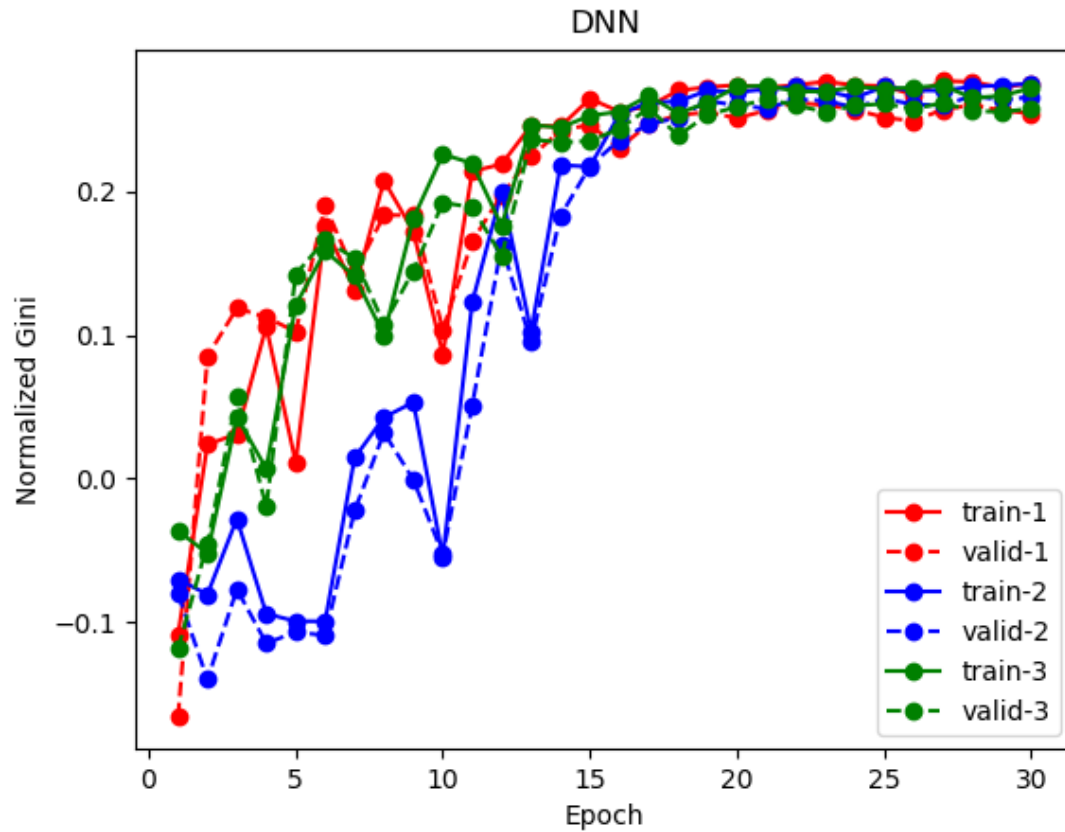
DeepFM



FM



DNN



Some tips

- ☐ You should tune the parameters for each model in order to get reasonable performance.
- ☐ You can also try to ensemble these models or ensemble them with other models (e.g., XGBoost or LightGBM).

Reference

[1] *DeepFM: A Factorization-Machine based Neural Network for CTR Prediction*, Huifeng Guo, Ruiming Tang, Yunming Yey, Zhenguo Li, Xiuqiang He.

Acknowledgments

This project gets inspirations from the following projects: - [] He Xiangnan's neural_factorization_machine
- [] Jian Zhang's YellowFin (yellowfin optimizer is taken from here)

License

MIT