数据预处理：

13 个连续特征，26 个类别特征
对 13 个连续特征进行值的截断操作，按照给定的最大值对每个连续特征进行剪裁。
对 26 个类别特征进行词典生成，将每个类别特征转化为唯一的整数编码，同时对于低频次类别进行过滤。

对原始数据进行 one_hot 编码并存储每一个特征非空的索引，连续型变量保留数值，分类变量保留索引。得到如下数据：（train.txt）

```
1,1,5,0,1382,4,15,2,181,1,2,0,2,2,10,0,5430,1,1,2217,3,1,368,1251,0,1129,3,218,0,1,21,1,1,0,0,2,2134,2,1841,0
2,0,44,1,102,8,2,2,4,1,1,0,4,2,16,10,12,1,3,2134,1,1,786,681,11,661,1,41,12,2,16,1,2,12,0,2,17,2,9,0
2,0,1,14,767,89,4,2,245,1,3,3,45,24,48,2,1,1,1,383,1,1,1,470,3,417,2,729,3,7,312,0,0,3,1,2,3,0,0,0
0,893,0,0,4392,0,0,0,0,0,0,0,0,2,13,0,226,1,3,72,1,1,2,2,0,2,2,18,0,8,30,0,0,0,0,2,7,0,0,0
3,-1,0,0,2,0,3,0,0,1,1,0,0,4,30,0,6,1,4,1440,1,1,963,720,40,695,3,354,16,8,296,0,0,16,0,1,2,0,0,0
0,-1,0,0,12824,0,0,0,6,0,0,0,0,1,122,68,95,3,5,1052,1,1,1,936,73,869,2,490,83,5,161,0,0,78,4,5,66,0,0,0
0,1,2,0,3168,0,0,1,2,0,0,0,0,17,66,50,64,3,3,0,1,1,1,637,56,621,2,181,63,5,85,0,0,58,0,9,27,0,0,0
1,4,2,0,0,0,1,0,0,1,1,0,0,2,13,0,59,10,4,8,3,1,22,232,0,3,2,18,0,1,30,0,0,0,0,1,7,0,0,1
0,44,4,8,19010,249,28,31,141,0,1,0,8,1,22,1,1,1,1,1493,1,1,7681,928,1,679,1,226,1,1,149,0,0,1,0,1,3,0,0,0
0,35,0,1,33737,21,1,2,3,0,1,0,1,1,127,65,89,1,0,71,1,1,12,2,70,2,2,426,79,3,335,0,0,74,0,1,86,0,0,0
0,2,632,0,56770,0,0,5,65,0,0,0,2,1,18,485,253,1,1,1024,1,2,1,166,227,186,1,1443,232,5,674,1,2,220,0,2,97,3,71,0
0,6,6,6,421,109,1,7,107,0,1,0,6,1,427,0,0,2,0,2742,1,1,1,1103,0,1005,2,2010,0,3,1182,0,0,0,0,1,0,0,0,0
0,-1,0,0,1465,0,17,0,4,0,4,0,0,13,1,0,8,1,3,1,3,1,21,1,0,1,2,1110,7,1,31,0,0,7,0,1,2,0,0,1
0,2,11,5,10262,34,2,4,5,0,1,0,5,5,5,0,5055,1,0,272,16,1,292,9,0,7,3,12,0,2,2,126,3,0,0,1,133,5,1839,1
0,51,84,4,3633,26,1,4,8,0,1,0,4,3,10,0,747,6,4,2990,1,1,3341,779,0,746,3,1138,2524,4,288,1,1,0,0,8,441,2,537,0
0,2,1,18,20255,0,0,1,1306,0,0,0,20,1,261,186,96,2,1,818,8,1,1,273,194,272,22,2186,214,5,489,0,0,203,0,2,93,0,0,0
1,987,0,2,105,2,1,2,2,1,1,0,2,2,3,0,0,3,1,3679,1,1,1,958,0,887,2,25,0,3,9,1,1,0,0,1,4998,1,5,1
0,1,0,0,16597,557,3,5,123,0,1,0,1,4,100,5,3,2,2,0,1,1,0,3733,6,2882,1,853,6,1,598,0,0,6,1,1,6,0,0,0
0,24,4,2,2056,12,6,10,83,0,1,0,2,1,16,0,13656,4,2,7,4,1,36,6,0,9,3,30,0,1,16,1,1,0,0,1,194,3,1443,0
7,102,0,3,780,15,7,15,15,1,1,0,3,27,152,2736,3739,2,4,1388,1,1,1299,787,2808,582,11,0,3193,1,2466,0,0,2914,0,10,2145,0,0,
0,47,0,0,6399,38,19,10,143,0,10,0,6,20,1,39,51,1,1,1464,2,1,1,1050,44,965,3,315,51,1,122,0,0,45,0,1,55,0,0,1
0,1,80,0,1848,287,1,4,46,0,1,0,4,1,7,0,6497,1,1,64,1,1,1,1055,0,110,2,6,0,3,5,12,3,0,0,1,2695,2,844,0
0,0,14,6,7132,171,2,2,6,0,1,0,6,1,1,2300,1106,1,2,670,2,1,9,114,2360,42,1,2020,1907,2,676,0,0,1735,0,2,4,0,0,0
0,9,9,17,11774,0,0,23,128,0,0,0,17,1,12,2361,107,1,1,1260,1,1,750,617,203,603,2,22,97,8,27,0,0,91,0,6,23,0,0,0
0,1,2,0,6190,84,1,27,71,0,1,0,0,3,107,1,1,1,1,3120,1,1,7312,1816,1,1042,2,1399,1,3,346,0,0,1,2,2,3,0,0,0
0,4,16,0,5925,2,2,0,0,0,1,0,0,3,7,1498,2021,1,0,41,1,1,70,26,1538,28,4,211,1749,2,5,823,3,1600,0,1,1,2,1,0
0,1,20,16,1548,93,42,32,912,0,15,1,16,4,43,0,5492,1,1,24,3,1,68,14,0,17,2,107,0,6,37,1,3,0,0,1,815,13,641,1
0,20,2,2,7188,170,2,3,24,0,2,0,2,2,1,300,384,2,1,621,1,1,10363,236,312,242,2,905,337,4,531,0,0,320,0,2,370,0,0,1
0,78,2,15,4311,85,4,18,230,0,3,0,15,2,195,0,0,2,5,387,1,1,236,71,0,83,1,1082,0,1,479,0,0,0,0,1,1733,0,0,1
3,0,4,13,224,28,3,35,27,1,1,0,13,1,27,2323,3199,1,4,2,1,1,5,2,2381,2,4,184,2718,1,137,0,0,2476,0,2,260,0,0,1
0,277,0,3,7318,24,6,3,98,0,1,0,3,4,144,2577,2595,2,3,721,1,1,1,597,2646,585,2,1342,3009,1,893,0,0,2748,0,2,84,0,0,1
0,-1,0,0,4956,0,0,37,97,0,0,0,0,5,34,1,1,1,3,866,1,1,1,1061,1,88,7,47,1,5,57,0,0,1,0,7,3,0,0,0
1,0,1,0,1427,3,16,11,50,0,2,1,0,1,170,1528,425,2,1,24,1,1,25,14,1569,17,2,991,1783,4,324,0,0,1625,0,1,152,0,0,0
4,0,55,8,859,13,4,12,13,1,1,0,8,1,28,0,747,1,0,64,1,1,1,260,0,110,5,773,2524,5,23,1,3,0,0,1,441,2,1894,0
```

模型原理部分

$$y = \omega_0 + \sum_{i=1}^{n} \omega_i x_i + \sum_{i \neq j} \omega_{ij} x_i x_j$$

为了构造 deepfm(FM+DNN)数据，以及对应一阶参数项+二阶参数矩阵

ω_{ij}写成一个大矩阵W，W是实对称

因此尝试对 W 进行矩阵分解，$W = V^T V$
W 为 n*n,V 为 k*n,k 是自己设定的参数，
所以$\omega_{ij} = V_i * V_j$

$$\sum_{i\neq j}\omega_{ij}x_i x_j = \frac{1}{2}\sum_{f=1}^{k}[\left(\sum_{i=1}^{n}v_{i,f}x_i\right)^2 - \sum_{i=1}^{n}{v_{i,f}}^2{x_i}^2]$$

Step1：我们需要构建一个 dataset_self.py 文件，输出 Xi, Xv, target_i

Xi 表示每个特征对应的非空索引，后续用于 embedding 层
Xv 表示数据的真实值，后续用于数值计算

Step2：**DeepFM.py**

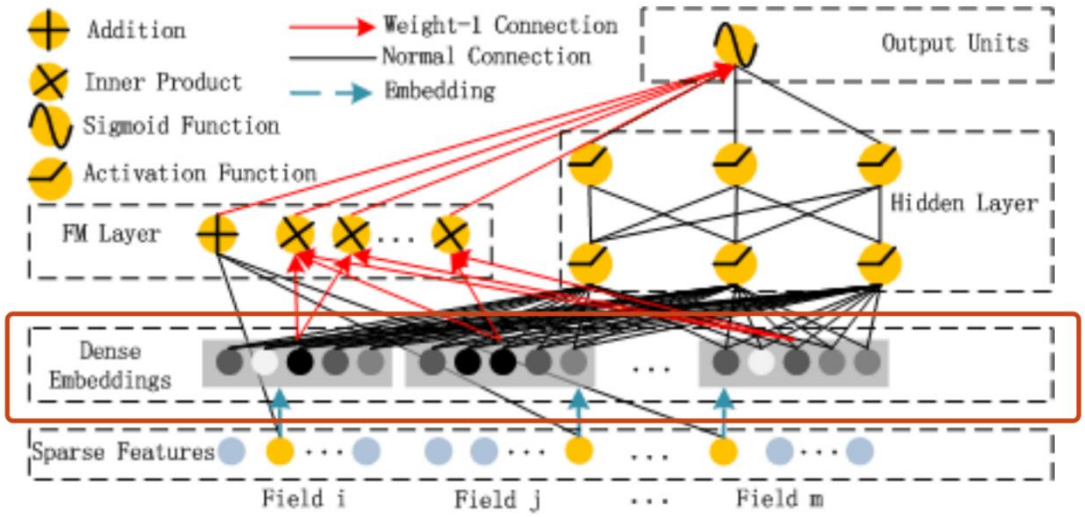
FM 部分构建：

```
nn.Embedding(feature_size, 1)#构建一阶参数项
nn.Embedding(feature_size, self.embedding_size)#构建二阶参数项矩阵
self.embedding_size 默认为 4
```

结合上述公式：

```
fm_first_order_emb_arr = [(torch.sum(emb(Xi[:, i, :]), 1).t() *
Xv[:, i]).t() for i, emb in
enumerate(self.fm_first_order_embeddings)]
        fm_first_order = torch.cat(fm_first_order_emb_arr, 1)
        fm_second_order_emb_arr = [(torch.sum(emb(Xi[:, i, :]),
1).t() * Xv[:, i]).t() for i, emb in
enumerate(self.fm_second_order_embeddings)]
        fm_sum_second_order_emb = sum(fm_second_order_emb_arr)
        fm_sum_second_order_emb_square = fm_sum_second_order_emb *
\
            fm_sum_second_order_emb  # (x+y)^2
        fm_second_order_emb_square = [
            item*item for item in fm_second_order_emb_arr]
        fm_second_order_emb_square_sum = sum(
            fm_second_order_emb_square)  # x^2+y^2
        fm_second_order = (fm_sum_second_order_emb_square -
                           fm_second_order_emb_square_sum) * 0.5
```

DNN 部分构建:
嵌入拼接：首先，将二阶交互特征嵌入拼接成一个大向量，作为深度网络的输入。
逐层处理:
1、通过全连接层（线性层）将输入进行线性变换。
2、通过批量归一化层对线性变换后的输出进行标准化。
3、通过 Dropout 层对输出进行正则化，随机丢弃部分神经元。
(项目中做了一个两层的)



Step3:实际运行结果 **main.py**:

验证集上检查准确率
正确 410 / 1000 (41.00%)

迭代 500，损失 = 1298982.8750
验证集上检查准确率
正确 412 / 1000 (41.20%)

迭代 0，损失 = 8253220.5000
验证集上检查准确率
正确 411 / 1000 (41.10%)

迭代 100，损失 = 4338895.0000
验证集上检查准确率
正确 413 / 1000 (41.30%)

迭代 200，损失 = 479501.5312
验证集上检查准确率

正确 412 / 1000 (41.20%)

迭代 300，损失 = 12744058.0000
验证集上检查准确率
正确 410 / 1000 (41.00%)

迭代 400，损失 = 4321941.5000
验证集上检查准确率
正确 410 / 1000 (41.00%)

迭代 500，损失 = 679137.1250
验证集上检查准确率
正确 414 / 1000 (41.40%)

迭代 0，损失 = 1631117.7500
验证集上检查准确率
正确 414 / 1000 (41.40%)

迭代 100，损失 = 1417371.0000
验证集上检查准确率
正确 417 / 1000 (41.70%)

迭代 200，损失 = 10609286.0000
验证集上检查准确率
正确 420 / 1000 (42.00%)

迭代 300，损失 = 2725411.7500
验证集上检查准确率
正确 420 / 1000 (42.00%)

迭代 400，损失 = 270113.3438
验证集上检查准确率
正确 419 / 1000 (41.90%)

迭代 500，损失 = 790204.8125
验证集上检查准确率
正确 420 / 1000 (42.00%)