Recommender Systems & Multitask Learning

林秋强

1 Recommender Systems

1.1 Collaborative Filtering

推荐系统经常依赖于协同过滤(Collaborative Filtering, CF)[1]，协同过滤有两种主要的实现方式：近邻模型(Neighborhood models)和隐因子模型(Latent factors models)。

基本估计(Baseline estimates):

假设用户u对项目i的评分R_{ui}为 $R_{ui} = \mu + b_u + b_i$。其中μ 是所有项目的总平均分，b_u是用户偏差，b_i是项目偏差。为了防止过拟合加入正则项后，模型的目标为:

$\min_{b_u, b_i} \sum_{((u,i)\in K)} (R_{ui} - \mu - b_u - b_i)^2 + \lambda_1 (\sum_u b_u^2 + \sum_u b_i^2)$

近邻模型(Neighborhood models):

近邻模型的思想是在基础估计的结果上加上已知数据集中相邻数据的偏差，并用某种相似系数作为权重进行调整，数据之间越相似，偏差的影响越大。预测结果为：

$\hat{r_{ui}} = b_{ui} + \sum_{(j\in S^k(i;u))} \theta_{ij}^u (r_{uj} - b_{uj})$

其中b_{ui}、b_{uj}是基本模型的预测结果，$S^k(i;u)$是用户u有过评分且与项目i最接近的k个项目构成的集合，r_{uj}是用户u对项目i的评分，θ_{ij}^u是对应的相似系数。

隐因子模型(Latent factor models):

隐因子模型的思想是在基础估计的结果上加上一个与用户u和项目i之间的联系有关的偏差，且认为这个偏差是u对应的隐向量和i对应的隐向量

的内积。预测结果为：

$\hat{r_{ui}} = b_{ui} + p_u^T q_i$

其中p_u和q_i分别是u和i对应的隐向量。

结合模型(Integrated model):

结合模型将上述两个模型的额外项都加入预测结果，即

$\hat{r_{ui}} = b_{ui} + \sum_{(j \in S^k(i;u))} \theta_{ij}^u (r_{uj} - b_{uj}) + p_u^T q_i$

1.2 Factorization Machines

因子分解机(Factorization Machines, FMs)[2]有点类似于上文的隐因子模型。实际上，上一篇文章中提出的SVD++可以认为是FM的一个特例。FM在预测时，除了利用单个特征进行预测(类似于线性回归)，还利用了多个特征的复合特征(类似于使用的核技巧的SVM)，并且FM计算量比SVM更小且能处理稀疏情况。 FM给每个特征都对应了一个因子，复合特征(若干的特征的乘积)的系数是对应特征的因子的积。二维情况下的FM预测结果为：

$\hat{y}(x) = \omega_0 + \sum_{i=1}^n \omega_i x_i + \sum_{i=1}^n \sum_{j=1}^n (v_i, v_j) x_i x_j$

其中ω_0是全局偏置，ω_i是特征x_i对应的系数，v_i是特征x_i对应的隐向量。

1.3 Field-aware Factorization Machines

场感知因子分解机(Field-aware Factorization Machines, FFMs)[3]是FM的一个改进。它和FM的主要区别是它将特征打包为若干个场(field)，每个特征都属于一个场且对应每个场都有一个因子。使用FFM时，将FM中的
$\sum_{i=1}^n \sum_{j=1}^n (v_i, v_j) x_i x_j$ 项替换为
$\sum_{j_1=1}^n \sum_{j_2=j_1+1}^n (v_{j_1,f_2}, v_{j_2,f_1}) x_{j1} x_{j2}$

其中f_1和f_2是j_1和j_2对应的场。 FM的每个特征只对应1个因子，而具有f个场的FFM的每个特征都对应f个特征，即有f个因子。因此FFM的参数比FM更多，计算量也更大,但是结果通常更好。

1.4 Deep Neural NetWworks

[4]用深度神经网络(Deep Neural NetWworks, DNN)拟合并预测分数。为了效率要先从总的项目池中挑选一小部分候选项目，再给每个候选项目

预测分数并挑选其中分数最高的。文章采用的DNN是以ReLU为激活函数的全连接神经网络，比较依赖于输入特征的挑选。

1.5 Wide & Deep Learning

线性模型的执行效率高，能在大规模输入上运行，并且能够在训练集中有与预测数据接近的数据时给出对应的结果，即有记忆性(memorization，个人理解类似于Exploitation)，但是难以提取特征之间的复杂关系，泛化能力不足；相对的，深度神经网络能够提取特征之间的复杂关系，但是容易泛化(gengeralization，个人理解类似于Exploration)过度。将两者结合，能够同时拥有较好的记忆性和泛化性，实现Explore-Exploit trade-off。

文章[5]中结合二者的方法是将一个浅、较宽的广义线性模型的输出和一个深、较窄的深度神经网络的输出加权平均后作为模型的总输出。结果为：

$$P(Y=1|x)=\sigma(w_{wide}^T[x,\phi(x)]+w_{deep}^T a+b)$$

其中σ是sigmoid函数，$\phi(x)$是x的外积变换，a是深度模型的输出，w_{wide}和w_{deep}分别是宽度模型和深度模型的权重向量。

1.6 DeepFM

DeepFM[6]是上文Wide & Deep Learning的一种改进。它也将一个宽度模型和深度模型结合，然而宽度模型不再使用简单的广义线性模型，而是具有较好效果的FM(为什么不用FFM？)。另一个改进是DeepFM不再给深度和宽度模型分别提供输入，而是将同一个输入经过编码后同时提供给模型的两个部分。这样的设计使得整个模型能够自动提取特征而不再需要人工挑选，并且根据文章的实验结果，还能够提升模型的准确性。模型的输出为：

$$P(Y=1|x)=\sigma(y_{FM}+y_{deep})$$

其中σ是sigmoid函数，y_{FM}是宽度部分即FM的输出(具体公式见上文)，y_{deep}是深度部分即DNN的输出。

2 Multitask Learning

多任务学习(Multitask Learning)[7]是指训练模型时，在主任务之外额外增加辅助任务，并对总体输出进行优化。当辅助任务和主任务相关时，

多任务学习的准确性要高于单任务学习，可能的原因有(1)每个任务有独立的噪声，同时学习多个任务是可以抵消噪声；(2)在前一点的基础上，可以更清晰地体现输入特征和任务目标之间的关系实现特征选择；(3)对主任务来说，一个复杂特征可能难以学得，而辅助任务可以更直接地学得这个特征。由于主任务和辅助任务共用部分模型，使得主任务也可以使用辅助任务学得的特征；(4)多任务可以给模型提供一个偏好，即模型倾向于学得使多个任务都有较好效果的表示。

许多单任务学习都可以改写成多任务学习，常用的方法有(列举了文章中的一部分)：(1)把实际预测时无法获得，而训练集中样本可以获得的特征作为辅助任务的目标；(2)把主目标的不同的表示和度量作为辅助任务的预测目标；(3)预测时间序列时不只预测下一时刻，而是后续多个时刻的数据；(4)使用直觉上对主任务目标有帮助的特征作为辅助任务的目标；(5)将一部分已给的特征不作为输入而是辅助任务的目标。

对于DNN，为每个辅助任务增加额外的输出神经元即可实现多任务学习，其他模型例如KNN、决策树等都可以改造为多任务学习的形式。

2.1 MTL in DNN

在DNN上使用多任务学习思想时，有两种常见的实现方法[8]。一种是让所有任务在底层共享同一个网络，而在顶层给每个任务都设置一个独立的网络，这种方式称为硬参数共享(hard parameter sharing)；另一种方式给每个任务都设置独立的网络，但是对网络参数进行限制(例如在损失中增加不同网络的参数之间的l_2距离)，使得不同任务的网络之间具有某种相似性，这种方式称为软参数共享(soft parameter sharing)。

此外还有在硬参数共享方式中的独立网络部分增加额外的先验限制、动态调整网络结构、在每个任务对应独立网络之间加入共享的神经元(cross-stitch)、把网络低层隐藏层的输出作为辅助任务的目标、动态调整多个任务目标之间的权重等在DNN中实现MTL的方法。

2.2 MMoE

MoE(Mixture-of-Experts)是上文硬参数共享网络的一种改进。MoE的共享部分不再是一个全连接网络，而是若干个被称为专家(expert)的独立神经网络，并且额外增加了一个被称为门(gate)的神经网络，它的输出是每个专家的输出在最终输出中的权重。模型的输出为：

$y = \sum_{i=1}^{n} g(x)_i f_i(x)$

其中$g(x)_i$是门网络的第i个神经元的输出，$f_i(x)$是第i个专家网络的输出。

如果将MoE整体作为模型的一层，则称为MoE层(MoE Layer)。

MMoE(Multi-gate Mixture-of-Experts)[9]是MoE的一种改进。它不再对所有任务使用同一个门，而是为每个任务都独立设置一个门，即第k个任务对应的输出为：

$y_k = h^k(f^k(x))$

其中$f^k(x) - \sum_{i=1}^{n} g^k(x)_i f_i(x)$,$h^k$是第k个任务独占的顶端网络对应的函数，$g^k(x)_i$是第k个任务对应门的第i个神经元的输出。

2.3 SNR

SNR(Sub-Network Routing)[10]也是硬参数共享的一种改进方法，它在共享的底部层与层之间加入了一个由矩阵乘法给出的变换(文章中给出了矩阵的两种形式)，矩阵中的参数训练过程得到。当矩阵的某个元素为1时，相当于该元素对应的两个子网络之间有连接;当矩阵的某个元素为0时，相当于该元素对应的两个子网络之间没有连接。因此SNR能起到动态调整网络结构的效果。

参考文献

[1] Koren Y . Factorization meets the neighborhood: A multifaceted collaborative filtering model[C]// Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008. ACM, 2008.

[2] Rendle S. Factorization Machines[C]// IEEE International Conference on Data Mining. 2011.

[3] Juan Y, Zhuang Y, Chin W S, et al. Field-aware factorization machines for CTR prediction[C]//Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 2016: 43-50.

[4] Covington P, Adams J, Sargin E. Deep neural networks for youtube recommendations[C]//Proceedings of the 10th ACM conference on recommender systems. ACM, 2016: 191-198.

[5] Cheng H T, Koc L, Harmsen J, et al. Wide & deep learning for recommender systems[C]//Proceedings of the 1st workshop on deep learning for recommender systems. ACM, 2016: 7-10.

[6] Guo H, Tang R, Ye Y, et al. DeepFM: a factorization-machine based neural network for CTR prediction[J]. arXiv preprint arXiv:1703.04247, 2017.

[7] Caruana R. Multitask learning[J]. Machine learning, 1997, 28(1): 41-75.

[8] Ruder S. An overview of multi-task learning in deep neural networks[J]. arXiv preprint arXiv:1706.05098, 2017.

[9] Ma J, Zhao Z, Yi X, et al. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 2018: 1930-1939.

[10] Ma J, Li Z Z J C A, Hong L. SNR: Sub-Network Routing for Flexible Parameter Sharing in Multi-task Learning[J]. 2019.