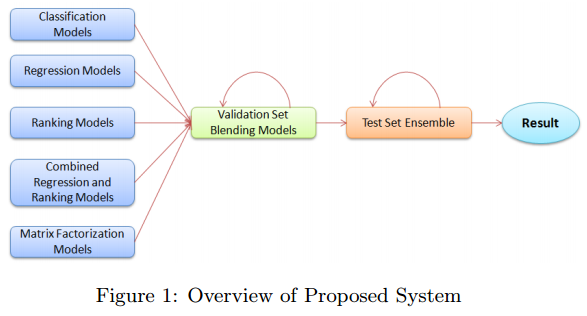
**Paper 总结**

# Wskdd12cup:

## 系统架构:

三个阶段：



第一阶段：单模型训练（individual models，蓝色）。包含分类，回归，排序各种模型。在validation set上求得各模型的参数。

第二阶段：融合模型1（blending，绿色，用validation set调参）。将单模型的输出作为第二阶段中融合模型的feature，进行模型训练，调整参数使得validation set上效果最好。这个阶段再重复一次（re-blending），即将融合模型的输出再作为feature，进行模型训练，调参。

第三阶段：融合模型2（粉色，利用test set进一步调参）。讲了2种方法，第一种方法我没看懂。我讲一下第二种方法（论文里说第二种方法的效果比第一种的好）。将单模型和融合模型的结果分别提交，得到评分后。取评分排名前五的模型，直接取平均作为最后的结果。

值得思考的是：

1. 融合模型也有技巧，即如果某个模型在validation set和线上test set的结果相差很多。有可能该模型已经过拟合了，则该模型不进行模型融合。
2. 得分转化。如果不同模型的输出结果形式不同，则模型融合前需要对不同模型的输出形式进行转化。比如归一化到[0.1]之间。

# Kdd12\_second.pdf

大致思想跟第一篇论文差不多，先单模型训练，然后将单模型的输出作为feature，并人工设置一些feature，输入到神经网络中训练得到最后的输出。

不一样的地方：

单模型使用了两种类型的模型（主要是使用的feature类型不同）：

一、直接拿数据集自带的各种id（raw data）作为feature（出现为1，不出现为0），然后采用以下模型训练：

1. Bias Model（分别分析每个feature出现会提升多少click的概率，直接将相应feature的概率值相加作为输出）
2. Factor Model（考虑这些feature的交互作用）
3. Asymmetric Factor Model（把user点过的item的feature也作为user的feature，考虑feature交互）
4. Naive Bayesian Model（计算这些feature同时出现，click的概率，不考虑feature交互）

二、从raw data中生成复杂feature：

1，计算每个id出现在record中时，click的概率。将概率值作为该id出现时相应feature的值（而不是直接取1）。

2，计算每个单词出现在record中时，click的概率。为每个id（比如query id

）计算2个feature，一个是该id包含的单词（该query中出现的单词）的概率值的平均值，另一个是所包含单词的概率值的最大值。

3, 考虑id的交互，计算每种id组合出现在record中时，click的概率。将概率值作为该id组合出现时相应feature的值。

4, 计算query和item的相似性。相似性是通过训练集中词和词组的统计信息和被click的比例得到的（没有利用其它语料库）。一共有12种相似性，对应于12个feature的值。

在不同类型上分别训练单模型，然后将单模型的输出和人工设置的feature作为输入，通过神经网络得到最后的输出。

值得注意的地方：

1. 数据集的分割很有意思。将数据集分成了training data和2个validation data。单模型在第一个validation data上调参。融合模型（最后的神经网络模型）在第二个validation data上调参。通过单模型在training data训练，比较2个validation data和test data上的测试结果是否一致来判断数据集分割是否合理。
2. 直接优化评价指标（ AUC），对于kddcup 2016的比赛，是否可以考虑直接优化NDCG。

# Kddcup2012\_5th.pdf

先单模型，再模型融合。

但模型融合很简单，没用复杂的机器学习的模型，也没有直接线性融合。

而是根据单模型的rank的结果（Regression，classification，rank模型都可以生成一个rank列表）对rank进行归一化，然后对rank融合（可以理解为求所有模型rank的平均）。

使用的单模型有：

1. Online Bayesian Probit Regression（Regression model）
2. SVM（classification model）
3. Latent Factor Model
4. Maximum Likelihood Estimation

值得注意的是：

本文花了很多篇幅讲如何处理特征

1. 去掉不在test data中出现的特征，及低频特征以加快训练速度。
2. 2种类型的特征，一种是原始数据提供的特征，包括离散的0,1特征（比如userID, AdID等）和连续的实数特征（每个特征的CTR值）。另一种是合成特征（本质上就是特征交互），包括两两组合的特征（比如userID\_AdID）及一种三组合特征（QueryID AdID UserID）。另外，把排序信息也加到特征里（这个我们用不上）。
3. 用LDA对文本进行降维（在其中一个单模型中使用），我们用的是SVD对文本降维（所有模型）。考虑了各种相似度特征，这个其实我们也有考虑，就是query\_in\_title, query\_in\_description，高奔所做的各种相似性工作。
4. （这个方法我们估计用不上）为了加快训练速度，把数据集分成10份，用同样的单模型进行训练，拿这些单模型的测试效果的平均值作为这个单模型的实际测试效果。
5. 发现某些特征发生过拟合时，去掉这些特征，用其他特征训练模型。将预测结果乘以所去掉特征本身的CTR值作为最终的预测结果。如果测试时某个特征没在训练集中出现，则乘以平均的CTR值。
6. 对于一些特殊的特征，给予惩罚。比如明显可以发现，在某些pattern下，CTR很差。则出现这些pattern时直接对结果惩罚。