关于如何基于 PPT 做文献调研和科学研究

各位同学好：

以前的组会上我多次说过，我们实验室开展工作应以 PPT 为载体，本文试图进一步说明此想法。

1. 总的来说，做一个好的研究工作至少包括两部分：(a)工作本身的思路清晰、逻辑严密；(b)对工作在相关方向中的贡献定位准确。

2. 在实际操作中，往往是先有(b)再有(a)。换言之，(b)是通过对本方向的文献进行梳理，找到一个有潜力的切入点，而(a)是根据该切入点，提出我们原创性的想法。

3. 要把(b)做好，又进一步有两点，(b1)文献覆盖面大（分时空两个维度）；(b2)形成自己的“知识树”，形象地说，本方向每篇文献将构成知识树的一个叶子节点，而树的根节点是我们要突破的大问题（例如会出现在大家毕业论文标题中的问题），形成知识树，就是把中间节点补全的过程。如果不具备一棵好的知识树，我们很难准确地发现可做的题目。本文下方“深度模型约减调研”是皈颍和春晖总结的一个(b)的 ppt 例子，供大家参考。

4. 当(b)做好后，每一个具体的研究工作都会形成一个对应(a)的 ppt。这个 ppt 应在我们选好点的那一刻起构建，然后随着研究的深入不断完善。这个 PPT 一般需要（但未必限于）明确回答几个问题（每个问题对应一个 Section）：
(a1) 我们要解决的问题是什么？
(a2) 我们为什么要解决这个问题
(a3) 问题的宏观解决思路，该思路在知识树中的定位
(a4) 我们提出的具体步骤
(a5) 我们的验证思路（理论推导、还是实验？理论推导试图回答什么问题？实验试图验证什么问题？）
(a6) 实验结果以及结论
本文下方“A Cost-sensitive Method for Learning from Crowds”是锦红最近一个工作的 PPT（(a)的例子），供大家参考。

5. 需要说明的是，不论(b)还是(a)，都不太可能一蹴而就。尤其是(a)。但对于(a)的 ppt 来说，每次更新并非简单的多加几页，而是有可能需要对之前已经写好的内容作调整，使得整个 ppt 更加完善。

6. 从附件中给出的例子可以看出，当(a)的 ppt 完成时，大家一篇小论文的提纲（甚至不少具体内容）就已经出来了。也就是可以正式开始撰写小论文。而(b)的 ppt，将构成小论文（甚至毕业论文）的 literature review 部分。今后，我们实验室的所有工作都以 PPT 上述的两类 ppt 为基准。每位同学的研究工作流程

大体如下：产生(b)的 ppt（没有(b)，不要进(a)）--》产生(a)的 ppt，(a)除实验结果之外的部分大体完成--》进入正式写程序、做实验的阶段（换言之没有 a1-a5，不要开始写任何程序）--》最后进入写论文。我们每一次的讨论（组会或单独讨论），其“输入”为一个 ppt 的版本，“输出”为对该 PPT 下一步完善的具体建议（除非已进入写小 / 大论文的阶段）。PPT 迭代的周期可由大家自行决定。

7. 附件的两个 PPT 耗费了三位同学的很多心血，例如锦红的 PPT 前后应更新了超过 10 版（但总比论文大改 10 遍好），跨度 1 年有余。尽管两个 PPT 还有完善的空间，但已基本符合我对于 PPT 的要求，我能感觉到在这过程中几位同学的工作能力也有显著提升（这个最重要）。所以建议其他同学仔细体会这些 PPT，若有不明之处，也可直接问我。

希望通过逐步形成上述工作方式，我能帮助大家在读研期间获得更扎实的提升（而不仅仅是学了几行代码，写了几篇论文）。

祝好。

唐珂

深度模型约减调研

李皈颖，姜春晖

2016年7月23日

UBRI

问题描述

对象： 一个已经训练的深度神经网络模型

目的： 缩小网络模型的大小

约束： 约减后模型的泛化性能损失尽量小

期望： 1. 模型规模减小带来存储空间的减少
2. 模型规模减小带来运行速度的提升
3. 模型规模减小使得网络可以进一步训练而不overfiting（可选）

衡量算法好坏的度量

数据集： 与未约减的网络使用的测试集一致

参数减少： 对比约减后算法与原算法的参数数量

运行速度： 确定平台下（cpu，手机等），约减后算法与原算法的速度对比

深度模型约减：优化问题

- 对深度神经网络的约减，可以统一形式化为一个优化问题

- 优化目标1： 最小化参数数量
- 优化目标2： 最小化泛化性能的改变

- 这即可以看作一个带约束的单目标， 又可以看作是一个两目标的多目标问题

深度模型约减方法综述

有哪些压缩模型的思维切入点

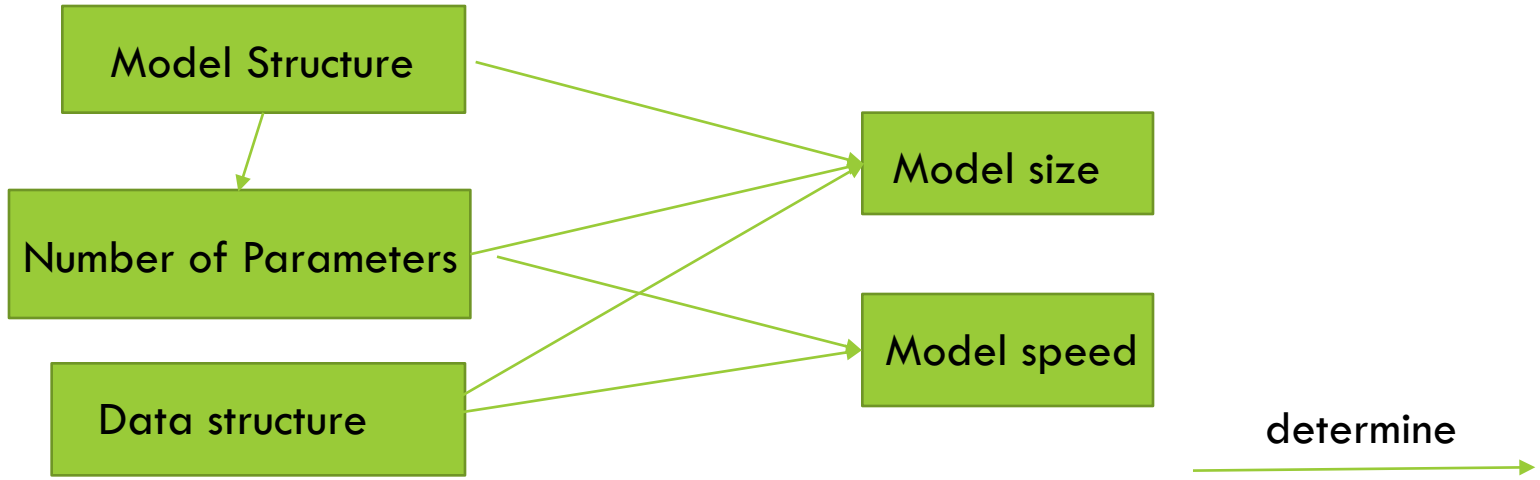
UBRI

概要

思路、流派

问题的思路

- 深度模型参数数量与 模型大小／运行速度之间的对照关系：



- 有两种角度看待问题
  - 直接优化上图的左侧（自变量）来看右侧的变化（因变量）
  - 根据对上图右侧的侧重（因变量的限制）来针对性优化左侧（自变量）
- 最终的实现方式都是通过优化自变量来达成，即优化上图左侧；而要达成我们的目标有两种路线可以走
  - 一是算法层面上减少参数数量/优化结构。
  - 二是工程层面上使用更简化的数据存储运算方式。

对问题的不同看法

- 数据结构、计算法方法优化的思路

1. 这是个工程性加速问题

- 减少参数数量的思路

1. 删去无用的信息
   - 去除的参数对约减后的网络将毫无影响
2. 压缩无用的信息
   - 信息量小的参数比重变小

- 从压缩、加速的效果侧重来看

1. 对于不同场景对模型压缩／加速的侧重是不同的
2. 使用不同的约减方法（左侧思路），对压缩／加速两个目标的显著程度影响

如何更改自变量 (主要介绍)

自变量和因变量的关系

如何约减网络

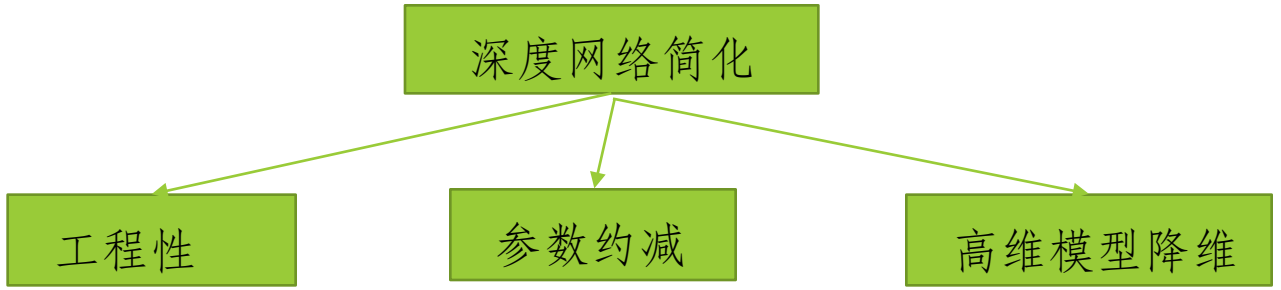
如何修改网络约减中的自变量

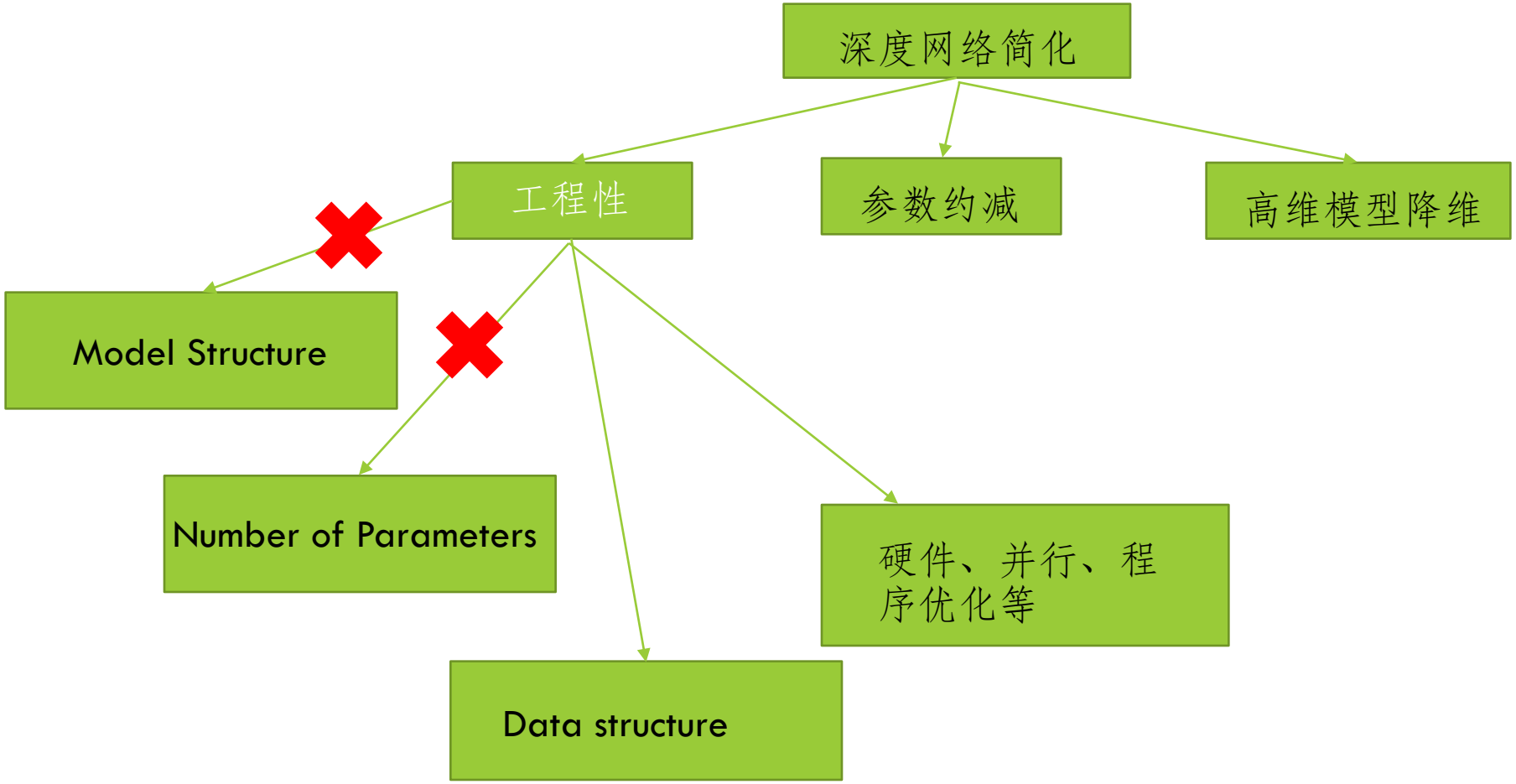
具体方法分类

1. 工程加速问题
   - 略

2. 参数约减问题
   - 剪枝(pruning)
   - 离散化(quantization)
     - Fixed-points 方法
     - 共享权值(feature sharing)

3. 高维模型降维问题
   - 连接矩阵降维
   - 编码压缩
   - 模型替换





工程性问题

硬件、编程等，简单介绍

工程性问题

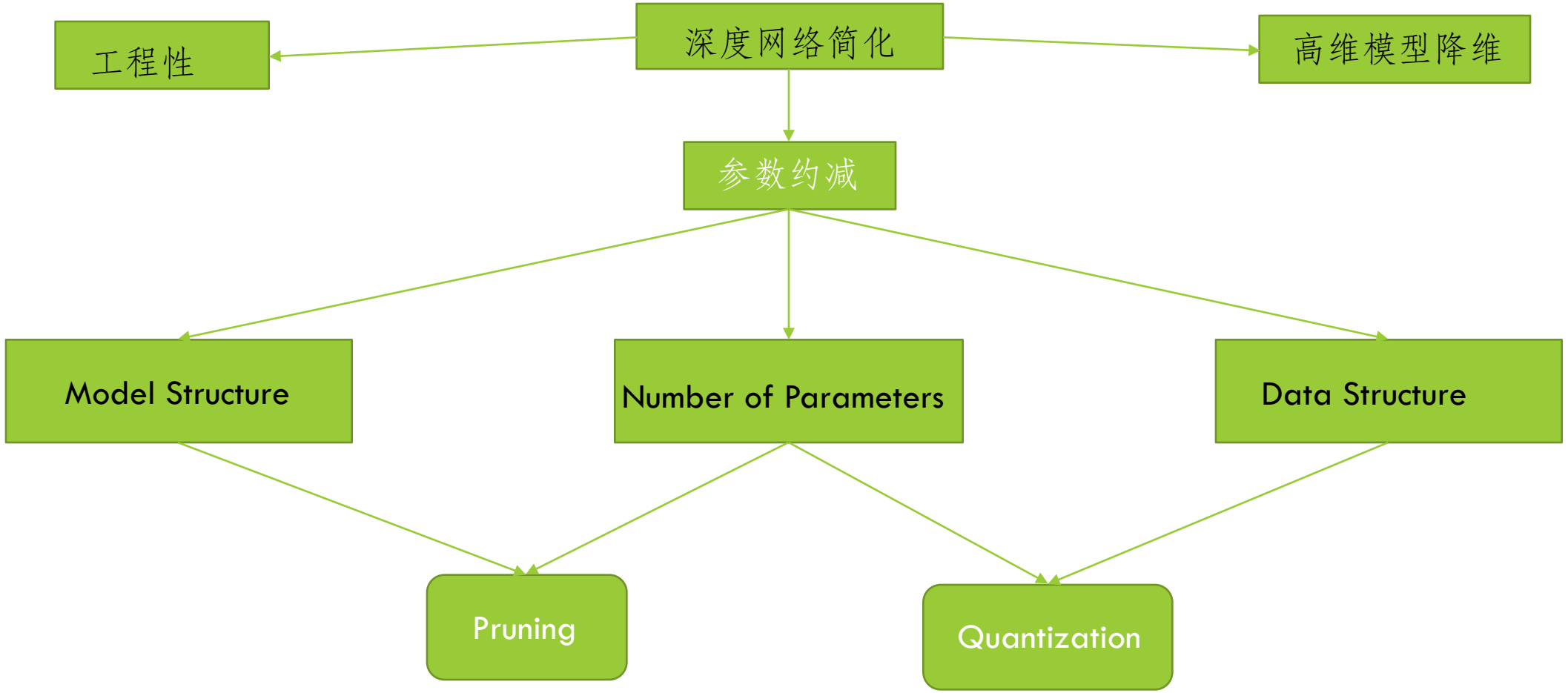
- 不对算法逻辑本身造成影响

- 不对网络结构本身造成影响
  - 结构：层层直接连接关系、隐层数、节点数等

- 在编译时进行代码优化*[13]*
  - 创建类似于GCC一样的专门针对神经网络的编译器

- 更适应硬件的网络向量化方法[18]
  - 更适应于并行加速

- 专用硬件[44][58]
  - 低bit位宽数据结构→适应这种位宽的神经网络→相应位宽的硬件



参数约减问题

降低网络中参数的多样性

参数约减问题

- Pruning（剪枝）[1][20][21][22][23][28][31][32][33]
  - 减去不重要的连接，重训练以保持准确率

- Quantization(离散化)
  - 目的是 **最小化** 取值的值域大小
  - Feature sharing(权值共享)[1][3]
    - 将权值离散化，只有数个可选值，将多个权值连接进行权值共享
    - 通过codebook方法来在权值和连接间构成映射[1][4][8][14]
  - Fixed-points 方法*[4][11][14]*
    - *数据精度的变化，也即意味着可选值的范围变化。*
    - 数据类型的变化：double → float → int → binary

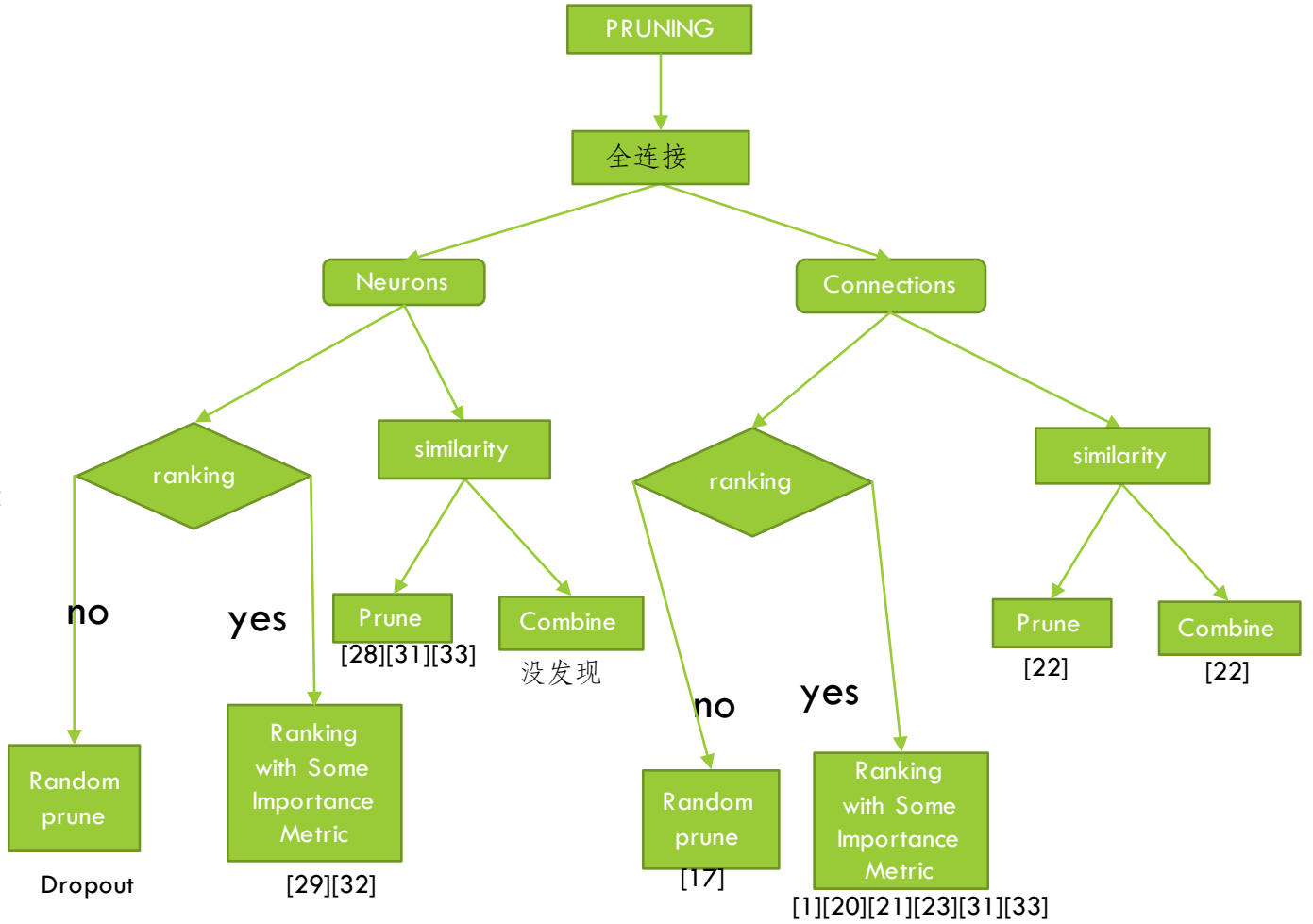
PRUNING

1989～2016 review

IDEAS

- 神经网络是一种图结构，节点是神经元，边是神经元见的连接

- 从prune的角度看，简化一张图，无非两种想法（都是number of parameters和model structure思路）：
  - 截去图里的边（神经网络的连接）
    - 考虑边的重要程度
      - 排序(利用某种预定的度量标准)
        - 贪心算法，利用阈值删去排序低的连接 [1][20][21][23][31][33]
          - Magnitude-based 算法
      - 边的相似性
        - 去掉相似的边[22]
        - 合并相似的边[31]
    - 不考虑边的重要程度
      - 随机删除，重训练[17]
  - 截去图里的节点（神经网络的神经元）
    - 考虑节点重要性
      - 排序(利用某种预定的度量标准)[29][32]
        - 贪心算法
    - 不考虑节点重要性
      - 随机删，重训练，dropout
    - 节点相似性
      - 去掉相似点[28][31][33]
      - 合并相似点



PRUNE：神经元

- 好处：在目前的工程环境下，除非有高效的稀疏矩阵函数库，不然实际中只能将prune掉的权值设为0，并没有真正起到简化存储和计算的效果。
- 减神经元则不同，减去神经元后的网络依然是稠密矩阵，可以用已有的矩阵计算库高效求解
- 通过某种预定义的度量对神经元进行评估，对于不要的神经元进行删减。[28][29][32]
- (感觉)减神经元和减权值在数学上应该是一样的，可以相互转化否?
- （方向）减神经元的文章比减权值的少，但思路上很相近，可不可以将减权值的方法迁移到减神经元里来发几篇文章?

PRUNE：连接

- 对连接排序
  - 将排序的依据称为“magnitude”
  - 有不同的评价指标来计算“magnitude”
  - 大部分是贪心的，利用阈值直接除去排序低的连接

- 找寻相似的连接（有点像聚类）
  - 将相似的连接合并在一起
  - 将相似的连接删去

对连接排序：MAGNITUDE-BASED

- 按“magnitude”对权值进行排序，将“magnitude”小于某个阈值的所有连接删去。
- 最简单的“magnitude”就是权值的绝对值[1][20][30][31][33]
- 对全连接层会有很好的效果
- 不同的magnitude计算方法：OBD,OBS, Fisher Information

不同MAGNITUDE计算方法

- OBD: 使用loss函数对于权值的二阶导数作为“magnitude”[21]
- OBS:为解决OBD对于矩阵假设过强而提出[23]
- 使用信息检索使用的fisher information来作为权值的剪枝先验[30]
  - Fisher information的好处是可以评估变量信息量的大小，因此可以用来衡量权值重要性

找寻相似的连接

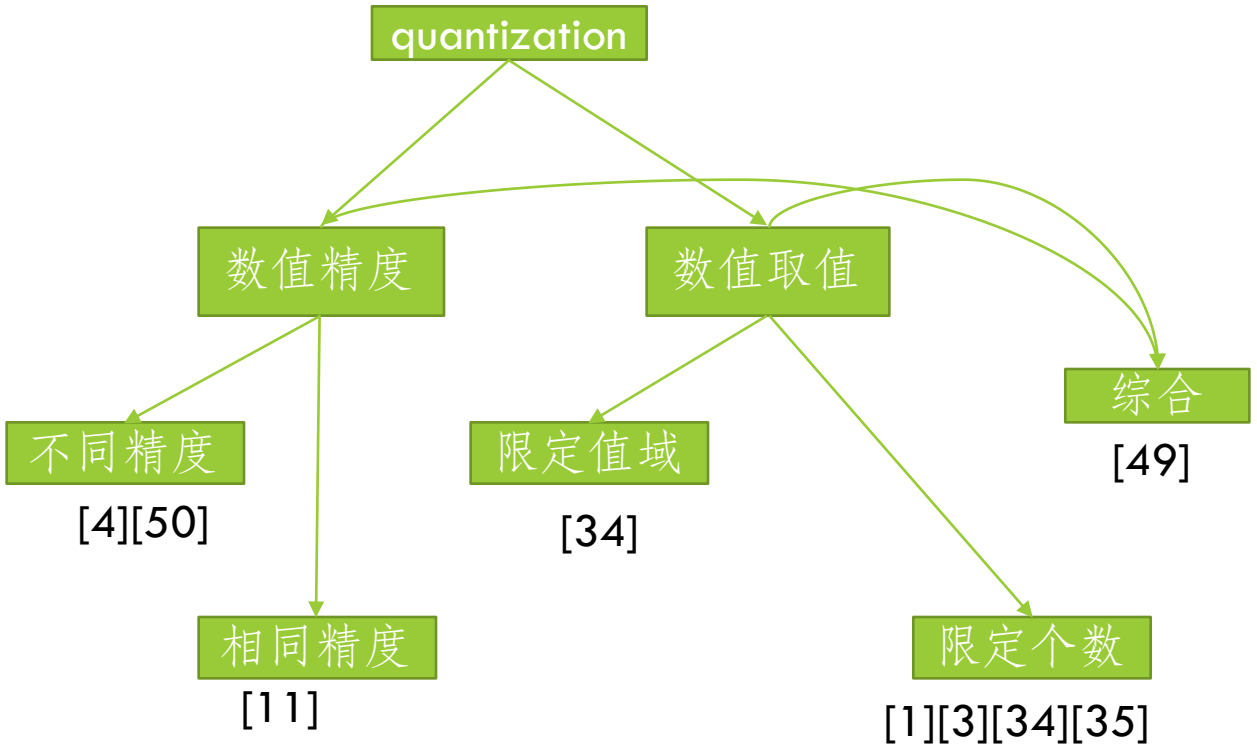
- 将相互之间强相关的权值删除[31]
- 但有些相似的信息，虽然冗余但也提供了些许独特的信息，聚少成多可能会对结果有大影响
- 将相似度比较高的权值进行合并而不是简单删除
  - 整个训练中变化不大的权值合并入bias[22][31]

QUANTIZATION

~2016 review

IDEAS

- 通过近似的手段，缩小模型中取值的多样性，从而降低网络存储开销。
- 这一系列方法都需要retrain来恢复准确率
- 这种想法有两种思路：
  - 缩小数值的取值数量(number of parameters思路)
    - 限定可选值的值域，一般是离散、确定的[34]
    - 限定可选值的数量[1][3][34][35]
  - 缩小数值的数值精度（data structure 思路）
    - 整个模型统一精度[11]
    - 模型不同部分不同精度[4][50]
  - 综合上面两种[49]



QUANTIZING: 数值精度(FIXED-POINT方法)

- 低精度值比高精度值的存储开销、计算开销低
- 在保证网络泛化性能的前提下，缩小数值精度可以极大的减小存储开销。
- 从模型的来看，缩小精度的范围无非两种：
  - 整个网络精度统一缩小[11]
  - 网络个部分分别缩小精度[4][50]
- 控制数值精度方法又称为“Fixed-point”方法
- 控制精度的方法天然契合与低精度的硬件，所以经常用于硬件加速。[58]

QUANTIZING:精度控制

- 统一控制精度
  - [11][39]方法通过设置不同的精度，并重训练网络，从中选出保持泛化性能且数值最小的数值精度。

- 模型局部控制精度
  - [50]对于一组确定的bit位宽，通过L2 误差优化的方法，优化每一层的网络的重构误；这个优化是逐层的，当优化某一层时，高层网络保持高精度结构。
  - [4]和[50]非常类似，但是更改了使用的优化方法，它指出[50]使用的确定bit位宽的方法费时费力（相当于穷举），其实完全可以把确定bit位宽看成一个优化问题。本文使用一个专门的优化算法，在最小化每层的重构误差的同时，选择最优的bit位宽。

QUANTIZING:精度控制

基于fixed-points方法，将每个网络层使用的最佳数值位宽作为优化目标。[4]

$$
\begin{aligned}
&for\ \boldsymbol{layer}\ in\ netowrk\\
&do\\
&\quad \alpha \leftarrow bit\ width\ of\ every\ elements\ in\ \boldsymbol{layer}\\
&\quad minimize(\alpha)\ ;subject\ to\ \boldsymbol{accuracy}\ of\ network\\
&end\ for
\end{aligned}
$$

QUANTIZING:数值取值

- 除了对每个值用近似值来降低存储，还可以对所有值的取值可能性进行限制，用一个值代替多个近似值的方法，降低模型中不同的取值数，从而降低存储。

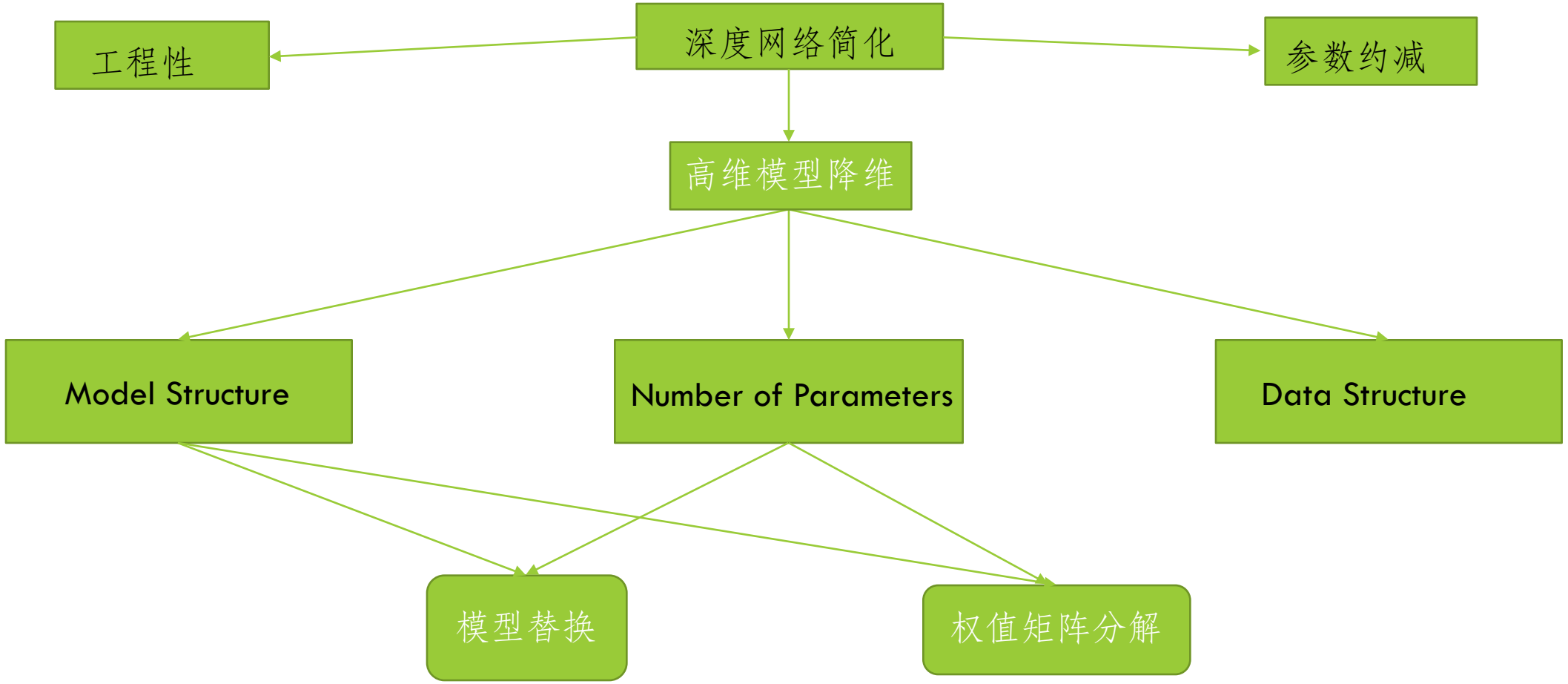
- 这种对数值取值的quantization有两种思路：
  - 限定值域：也就是规定只能使用某些值[34][49]
  - 限定个数：也就是不同值的个数是确定的[1][3][34][35]

QUANTIZING:数值取值

- 限定值域：也就是规定只能使用某些值
  - 二值网络，>0,<0各为一个值[34]
  - 3值网络：网络连接被限制为+1,0,-1[49]

- 限定个数：也就是不同值的个数是确定的
  - 对权值进行hash分组，同组的权值进行共享,利用codebook对权值进行存储[1][3]
  - 对权值进行聚类,利用codebook进行存储[34][35]
  - 将对每层的quantization看作是一个优化过程，在最小化每层重构误差时，学得最优的quantization，利用codebook进行存储[8]

Codebook：可以看成是一种映射表结构



高维模型降维问题

维度分解、压缩

高维模型降维问题

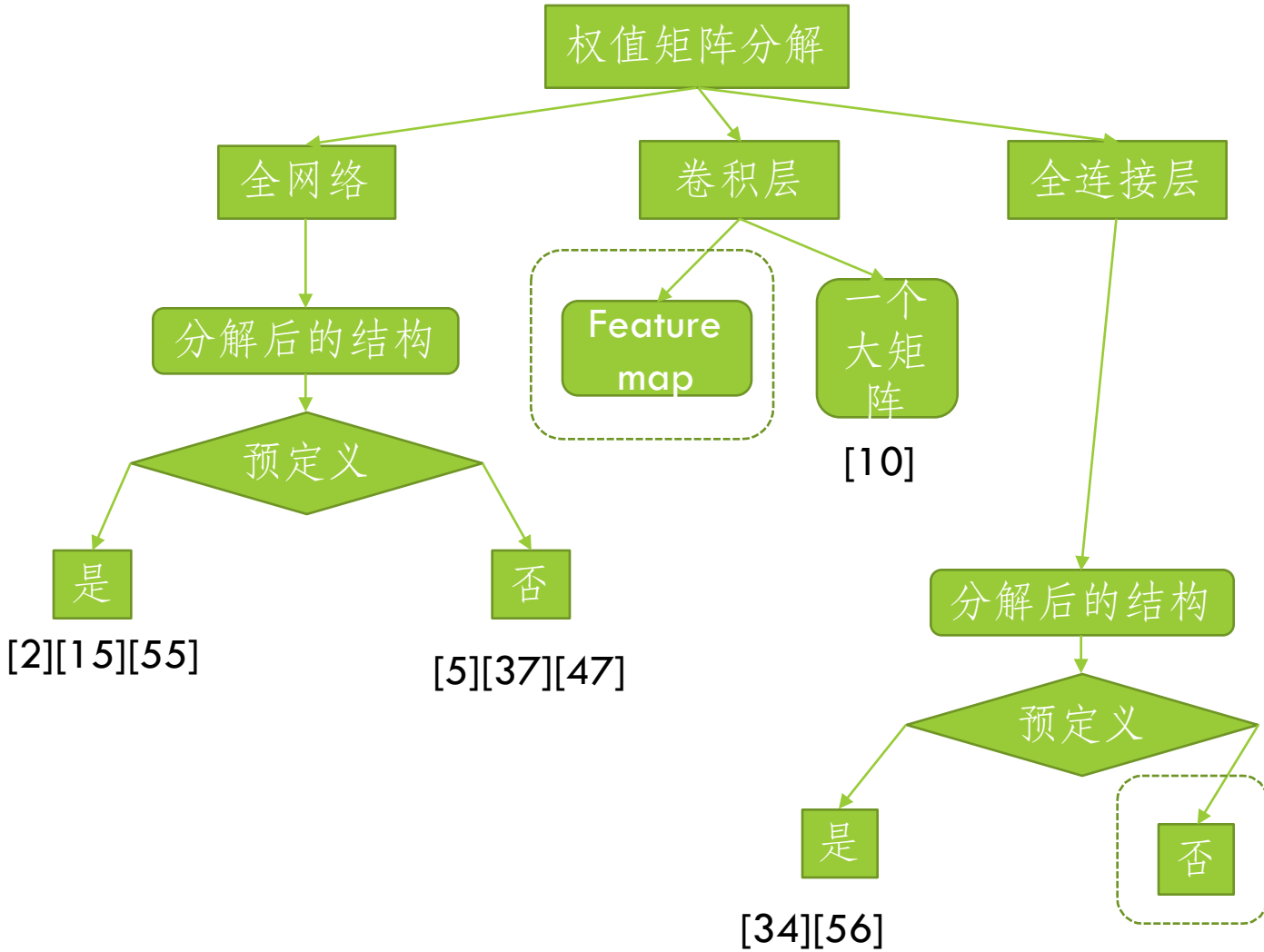
- 目的是用低复杂度的模型去近似原始的高复杂度模型
- 网络中的权值矩阵降维[2][5][6][10][15][33][37][55][56]
- 模型替换[7][41]

MATRICES FACTORIZATION

权值矩阵降维，～2016 review

IDEA

- 观察事实：深度网络中的权值有大量冗余。
  - [47]最优情况下，通过5%的参数可以预测出剩余参数。
- 基本假设：神经网络中的矩阵是低秩的。
  - 秩是矩阵线性不相关的列数,既这个假设神经网络中的连接有极大的冗余,而且冗余之间线性相关. [43]
- 所以可以通过将高维矩阵降维的方法，来缩减神经网络的权值矩阵大小。
- 神经网络是多层结构，每层连接都可以视作是一个矩阵，层与层之间相互关联，所以分解有两种思路，对全网络做，或是对部分层做：
  - 全网络：
    - 分解方式固定：[2][15][55]
    - 分解方式不固定：[5][37][47]
  - 针对卷积层结构：[10]
  - 针对全连接层：
    - 分解方式固定：[34][56]
    - 分解方式不固定：没有针对已训练好模型的方法



只有从头开始训练模型的方法

MATRICES FACTORIZATION:全连接层

- 因为对矩阵的分解是有损的，为了确保准确率，前期工作大部分都在单层或是某几层做。

- 对于一个全连接矩阵，它是一个二维矩阵（卷积层的矩阵是大于3维的）。

- 通过一些预定义好的分解规则对网络进行分解：
  - SVD[33]
  - Tensor-Train decomposition[56]

MATRICES FACTORIZATION:全网络

- 只要对于每层网络的分解可以将误差控制在一定范围内，那就可以将相关方法推广至全网络。

- 通过预定义的分解规则分解矩阵：
  - CP分解(CANDECOMP/PARAFAC decomposes)[15][55]
  - Tucker分解[2][55]
- 非预定义的分解方式：
  - 将对每层的分解转为**最小化非线性重构误差**和**最小化分解矩阵维度**两个子问题[5]
  - [37]同样是考虑每层非线性response的矩阵分解，和分解矩阵维度的选择,但它是通过PCA energy指标来选择rank

MATRICES FACTORIZATION:全网络

矩阵分解并不是无损的，所以**保持泛化性能**的同时**最小化分解矩阵的大小**是个优化问题。[5]

$for\ \boldsymbol{layer}\ in\ netowrk$

do

$A\ is\ the\ connecting\ matrix\ of\ layer$

$\mathrm{A} \overset{dimension}{\Longleftarrow\!=\!=} [m \times n]$

$\mathrm{A} \overset{dimension}{\Longleftarrow\!=\!=} [m \times l] \times [l \times n]$

$minimize(l)\ ;subject\ to\ \boldsymbol{accuracy}\ of\ network$

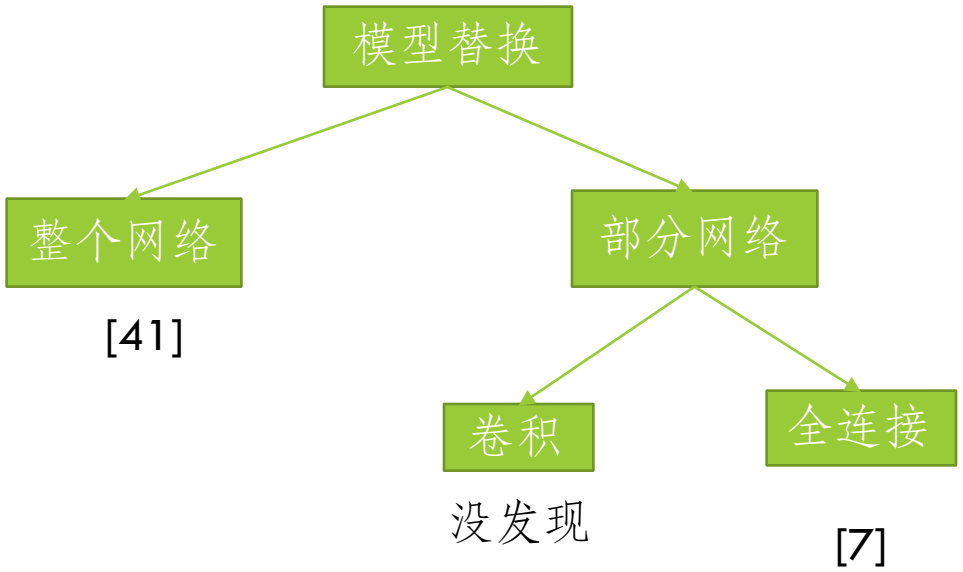
$end\ for$

模型替换

Simple review

IDEA

- 深度神经网络的结构有大量冗余
- 可否用更紧致结构去组织深度神经网络中得到的有用信息?
- 既然要用更适合的结构去替换已有的神经网络结构，那自然有两种方式
  - 替换全部[41]
  - 替换部分[7]



模型替换:替换全连接网络

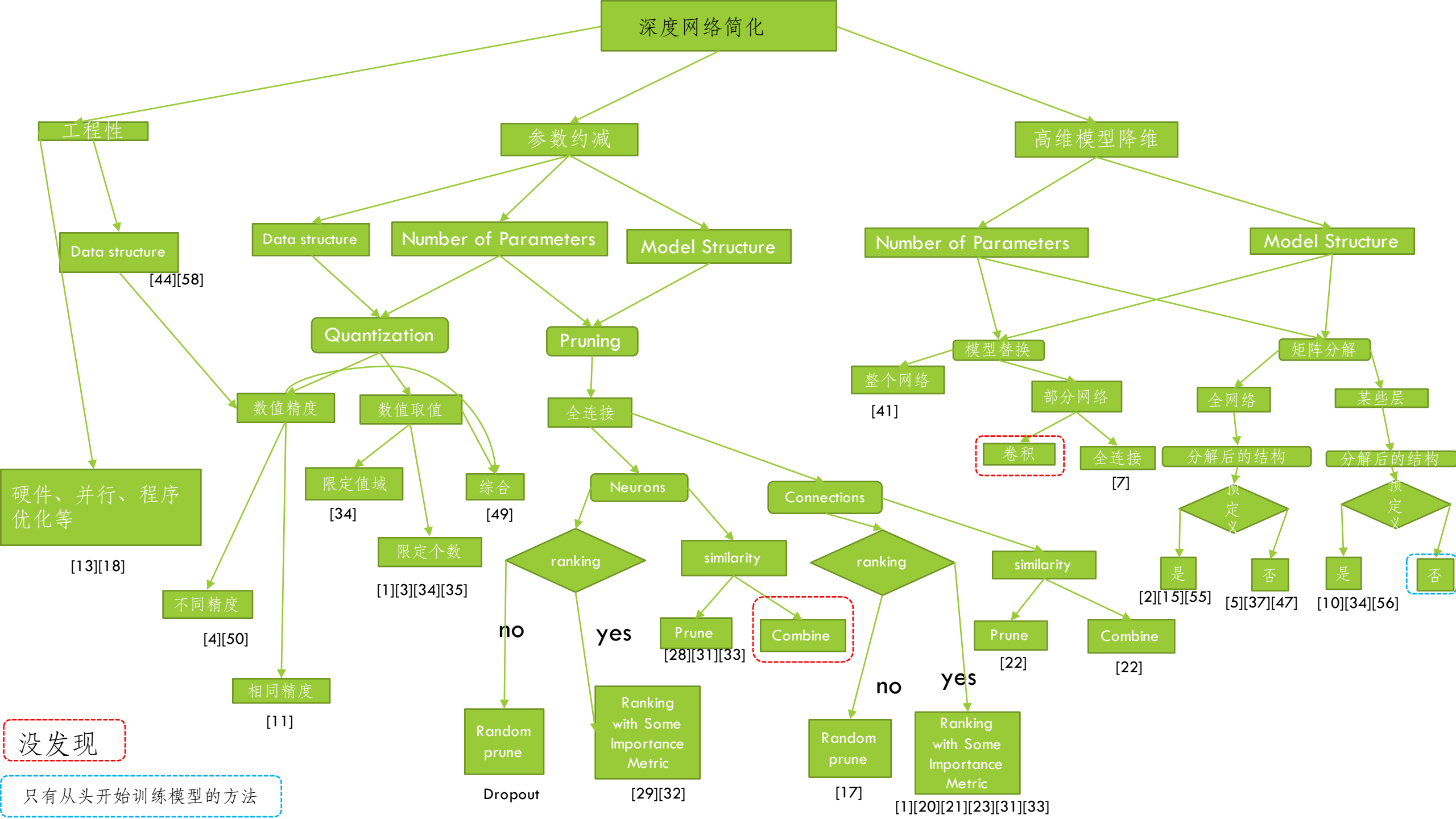
- 使用其他模型替换掉神经网络中计算量繁重的部分。
- 深度卷积网络的卷积层可以看作是特征抽取器
- 使用其他轻量级的模型替换全连接层，以提高模型的测试速度[7]
  - 论文[7]使用一个boosting forest model替换全连接层，但是目的侧重于提升速度，没有提及这样可以带来多大的压缩效率

模型替换：替换整个网络

- 训练好的深度神经网络中有大量的冗余

- 利用一个小的网络去拟合这个深度网络，将其中的有效信息提取出来[41]
  - 这种方法称为**Distilling**，蒸馏的意思。

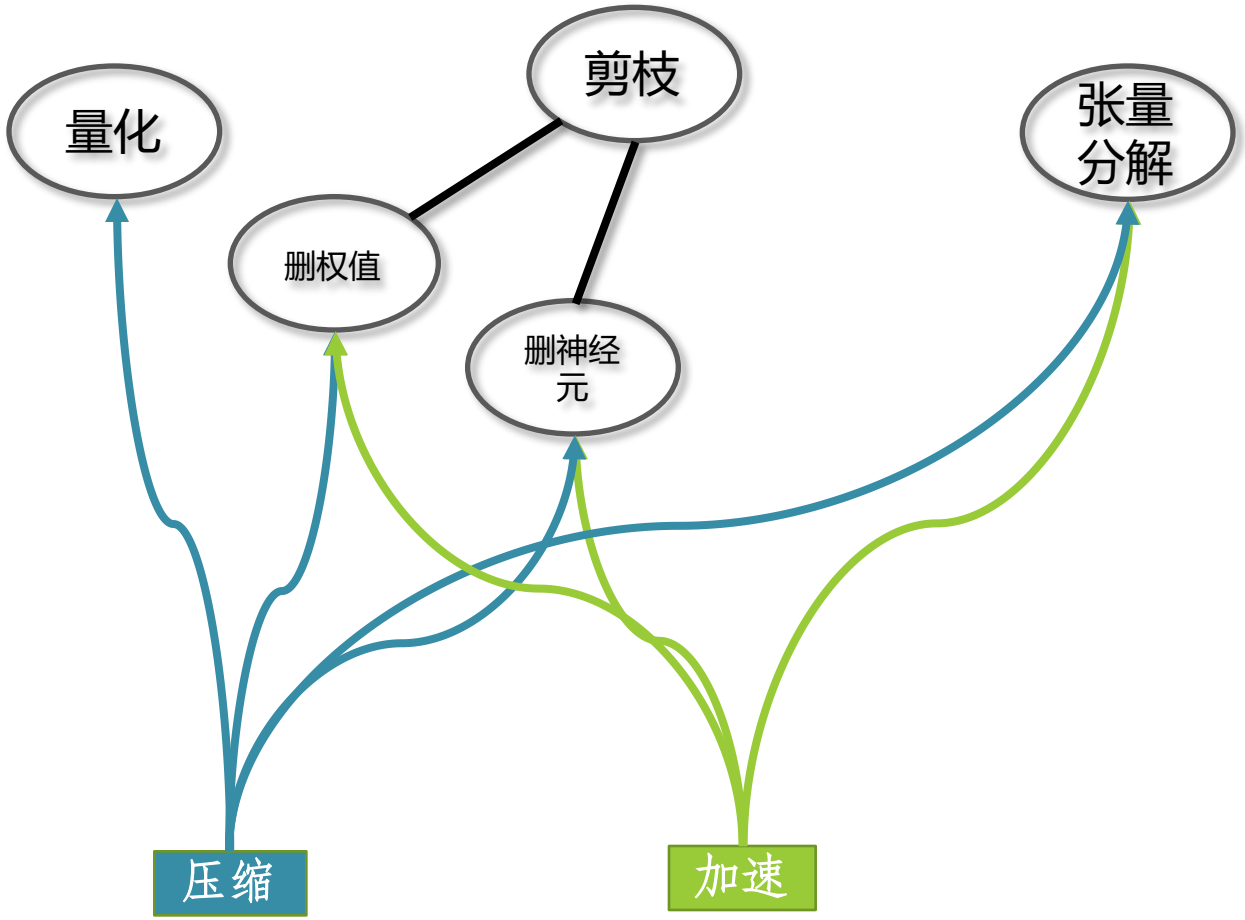
- 步骤：
  1. 使用大网络来产生大量的soft targets（其实就是最后输出的概率值）
  2. 用一个控制参数来对soft targets做归一化，控制它们的重要性（相对于target为1的ground-truth），这个参数称为T
  3. 用小模型去拟合混合了soft targets的数据
  4. 调整T，重复1～4，直至得到个最好的小网络。


深度网络简化
工程性
参数约减
高维模型降维
Data structure
[44][58]
硬件、并行、程序优化等
[13][18]
Data structure
Number of Parameters
Model Structure
Quantization
Pruning
数值精度
数值取值
不同精度
[4][50]
相同精度
[11]
限定值域
[34]
综合
[49]
限定个数
[1][3][34][35]
全连接
Neurons
Connections
ranking
similarity
no
yes
Random prune
Dropout
Ranking with Some Importance Metric
[29][32]
Prune
Combine
[28][31][33]
ranking
similarity
no
yes
Random prune
[17]
Ranking with Some Importance Metric
[1][20][21][23][31][33]
Prune
[22]
Combine
[22]
Number of Parameters
Model Structure
模型替换
矩阵分解
整个网络
[41]
部分网络
卷积
全连接
[7]
全网络
某些层
分解后的结构
分解后的结构
预定义
预定义
是
[2][15][55]
否
[5][37][47]
是
[10][34][56]
否
没发现
只有从头开始训练模型的方法

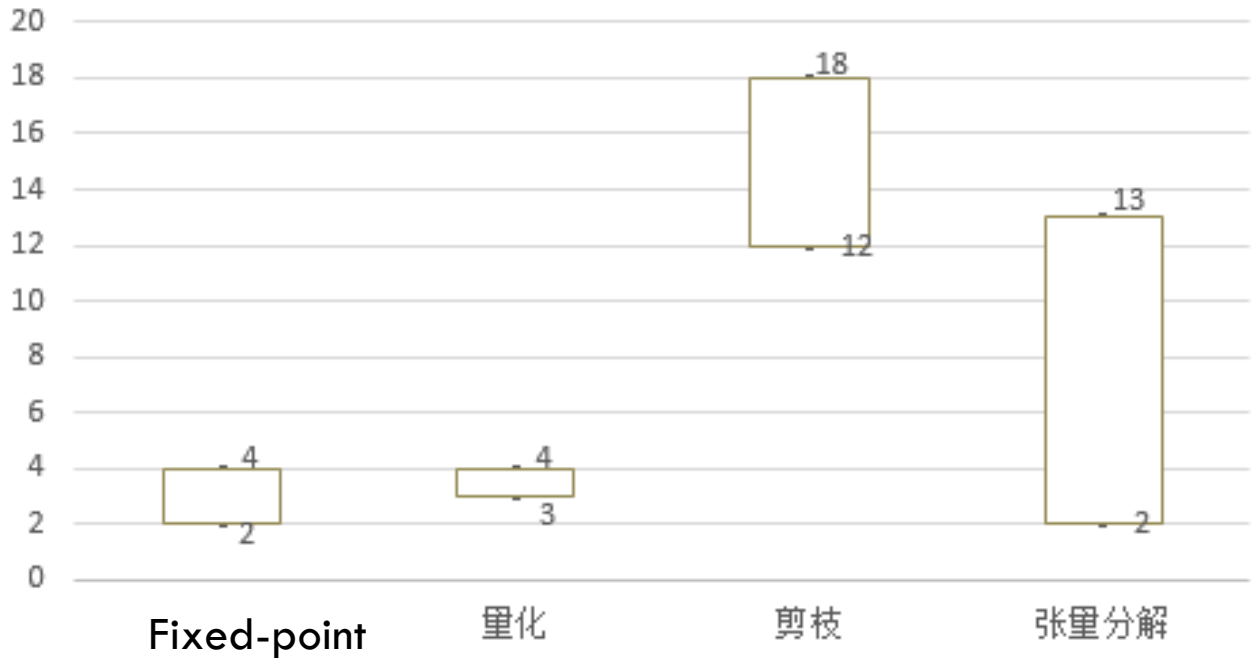
约减方法与约减效果

深度网络约减中自变量与因变量的关系

方法分类



方法效果（压缩性能）



- 张量分解（tensor decomposition）即是权值矩阵分解
- 这里将fixed-points方法与其它quantization方法区别开，因为fixed-points和其它quantization相比效果差异更大

方法效果（加速性能）

- pruning算法普遍在3～4倍之间的加速

- Quantization系列算法，没有关于速度的分析，只有关于压缩率的分析
  - 对于精度fixed-point方法，在通用计算机上运行时并不实际（通用的运算环境不能随意指定数据的bit位宽）
  - 对于限制网络取值的方法，网络的接连结构没有变化，实际计算时并不加速

- 矩阵分解系算法，根据使用方法的不同，差异很大：
  - [3],[[5],[37],[56]对于不同深度的网络，加速效果在1~4倍之间浮动
  - [15]在实验的网络中有8.5倍加速
  - [10]有5～10倍加速

总结与思考

我们应该怎么做

UBRI

总结

- 除pruning之外的方法，基本不存在不retrain而保持准确率降幅（<2%）不大的。
- Pruning的方法主要应用于全连接层，所以很多实验是在语音网络上做
- Fixed points提出的初衷和硬件加速很相关，因为嵌入式硬件一般精度偏低。
- quantization的方法偏向于对网络进行压缩存储，最极端的形式可以是编码压缩；这一类网络存在一个问题就是结果模型无法直接运行，或是需要专门定制化的方法来运行。
- Matrices factorization的方法面临两个主要问题，第一是如何确定分解后矩阵的最优大小，第二是如何保证每一层网络的重构误差最小。这两方面都已经有人在尝试解决。
- 在我们总结出的深度神经网络约减工作的树形结构中（PPT 41页）中，对模型替换的研究是最少的。
- 本综述限定于对已训练好的网络进行约减，如扩展到包括训练阶段的话，quantization和matrices factorization会有更多的工作，因为它们对信息的丢失比较多，更依赖于训练。

疑虑

现在最好的算法效果着实惊人[1][44], 不知道继续做的话提升空间是否足够。

Song Han Xingyu Liu Huizi Mao Jing Pu Ardavan Pedram Mark A. Horowitz William J. Dally *Stanford University, NVIDIA*

[1]→压缩率35x～49x, speed up, 3x~4x

- 240M→6.9M, 552M→11.2M,
- 缺点，无法直接运行，需要解压，这样实际运行的网络其实比压缩的结果大：
  - 解压的实际运行模型比压缩模型大1.19~1.56倍
- 这个工作室对已有Pruning, quantization的一个结合工作，和单纯提升pruning, quantization的工作性能是互补的。

[44]→在[1]的基础上，做硬件加速；相对于CPU，189x提速，相对于GPU，13x提速

想法（皈颖）

1. Pruning →基于阈值的pruning →用优化的方法选择最优 阈值（尝试中）
2. 可不可以用学习的方法，学出对每一层学出一个简化结构？类似于模型替换，我们将用于网络的的模型替换方法来逐层应用。
3. 模型替换中的全网络替换，怎样的小网络才是合适的？这可不可能是个优化问题？
4. 对于矩阵分解部分，能不能保证分解后的矩阵久没有冗余了？能不能对分解后的矩阵再分解？这样会不会进一步提升？
5. 可不可以通过统计的方法，跳出卷积层中线性不相干的feature map，然后用他们去组合生成其他的feature map？

想法(春晖)

1. 删特征图
    1. VGG-16网络结构的调试过程证明CNN中的特征图有冗余
    2. CNN中与全连接网络中删神经元对应的就应该是删特征图
    3. 可以考虑某种feature map重要性的度量，然后删去“不重要”的feature map

2. 演化算法与超参数

DNN约减中对各层指定不同的超参数是算法性能提高的重要方法。

1. [4]将浮点数网络转变为定点数网络时，对不同层指定不同的位宽（有人做但没人用演化算法做）
2. 剪枝中对不同层设定不同的阈值（皈皈在做）
3. 张量分解中也可以对不同层分解后矩阵的秩优化不同的值（还没人做）

如果这些问题都可以用演化算法做，那么合起来应该是篇很不错的文章。

想法（春晖）

1. 从多目标优化的角度思考深度网络约减问题

2. 用多目标（或者是演化算法）应该加一个限定条件：不用fine-tuning，因为fine-tuning破坏解的好坏的评估

3. 如果设3个目标：加速，压缩，准确率
   - 对剪枝和量化算法，压缩与准确率是冲突的，加速与另外两个目标的关系尚不确定
   - 对张量分解类算法，加速与压缩是一致的，加速与准确率是冲突的，压缩与准确率是冲突的

文献引用

所参考的资料汇总

引用

[1] Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding." *ICLR*(2016).

[2] Kim, Yong-Deok, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. "Compression of deep convolutional neural networks for fast and low power mobile applications." ICLR (2016).

[3] Chen, Wenlin, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. "Compressing neural networks with the hashing trick." *ICML* (2015).

[4] Lin, Darryl D., Sachin S. Talathi, and V. Sreekanth Annapureddy. "Fixed Point Quantization of Deep Convolutional Networks." ICML (2015).

[5] Zhang, Xiangyu, Jianhua Zou, Xiang Ming, Kaiming He, and Jian Sun. "Efficient and accurate approximations of nonlinear convolutional networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1984-1992. 2015.

[6] Yang, Zichao, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. "Deep fried convnets." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1476-1483. 2015.

UBRI

引用

[7] Yang, Bin, Junjie Yan, Zhen Lei, and Stan Z. Li. "Convolutional channel features." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 82-90. 2015.

[8] Wu, Jiaxiang, Qinghao Hu, Cong Leng, and Jian Cheng. "Shoot to Know What: An Application of Deep Networks on Mobile Devices." In *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[9] Coates, Adam, and Andrew Y. Ng. "Selecting receptive fields in deep networks." In *Advances in Neural Information Processing Systems*, pp. 2528-2536. 2011.

[10] Denton, Emily L., Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. "Exploiting linear structure within convolutional networks for efficient evaluation." In *Advances in Neural Information Processing Systems*, pp. 1269-1277. 2014.

[11] Gupta, Suyog, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. “Deep learning with limited numerical precision.” ICML (2015).

[12] Jin, Jonghoon, Aysegul Dundar, and Eugenio Culurciello. "Flattened convolutional neural networks for feedforward acceleration.” ICLR (2015).

UBRI

引用

[13] Mcafee L, Olukotun K. Utilizing Static Analysis and Code Generation to Accelerate Neural Networks[C]//Proceedings of the 29th International Conference on Machine Learning (ICML-12). 2012: 1847-1854.

[14] Courbariaux M, Bengio Y, David J P. Binaryconnect: Training deep neural networks with binary weights during propagations[C]//Advances in Neural Information Processing Systems. 2015: 3123-3131.

[15] Lebedev V, Ganin Y, Rakhuba M, et al. Speeding-up convolutional neural networks using fine-tuned cp-decomposition[J]. ICLR, 2015.

[16] Zhang Y, Wu J, Cai J. Compact representation for image classification: To choose or to compress?[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014: 907-914.

[17] Wan L, Zeiler M, Zhang S, et al. Regularization of neural networks using dropconnect[C]//Proceedings of the 30th International Conference on Machine Learning (ICML-13). 2013: 1058-1066.

[18] Ren, Jimmy, AND Xu, Li. "On Vectorization of Deep Convolutional Neural Networks for Vision Tasks" *AAAI Conference on Artificial Intelligence* (2015): n. pag. Web. 27 Jun. 2016

UBRI

引用

[19] Stollenga, Marijn F., Jonathan Masci, Faustino Gomez, and Jürgen Schmidhuber. "Deep networks with internal selective attention through feedback connections." In *Advances in Neural Information Processing Systems*, pp. 3545-3553. 2014.

[20] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network[C]//Advances in Neural Information Processing Systems. 2015: 1135-1143.

[21] LeCun, Yann, John S. Denker, Sara A. Solla, Richard E. Howard, and Lawrence D. Jackel. "Optimal brain damage." In NIPs, vol. 2, pp. 598-605. 1989.

[22] Thimm G. & Fiesler E. (1995): “Evaluationg pruning methods,” In 1995 International Symposium on Artificial Neural Networks, Proc. ISANN ‘95, pp. A2 20-25, National ChiaoTung University, Hsinchu, Taiwan.

[23] Hassibi, Babak, and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. Morgan Kaufmann, NIPS 1993.

[24] Ström, Nikko. "Phoneme probability estimation with dynamic sparsely connected artificial neural networks." The Free Speech Journal 5 (1997): 1-41.

UBRI

引用

[25] Hanson, Stephen Jos&#, Pratt L. Comparing biases for minimal network construction with back-propagation[M]// Advances in neural information processing systems 1. Morgan Kaufmann Publishers Inc. 1989:177-185.

[26] Culurciello E, Jin J, Dundar A, et al. An Analysis of the Connections Between Layers of Deep Neural Networks[J]. Computer Science, 2013, 53(3):20-5. arxiv.org

[27] Pan, Wei, Hao Dong, and Yike Guo. "DropNeuron: Simplifying the Structure of Deep Neural Networks." arXiv preprint arXiv:1606.07326 (2016).

[28] Srinivas S, Babu R V. Data-free parameter pruning for deep neural networks[J]. arXiv preprint arXiv:1507.06149, 2015.

[29] Chung, Jaeyong, Shin, Taehwan. Simplifying deep neural networks for neuromorphic architectures[C]// Design Automation Conference. ACM, 2016.

[30] Tu M, Berisha V, Cao Y, et al. Reducing the Model Order of Deep Neural Networks Using Information Theory[J]. arXiv preprint arXiv:1605.04859, 2016.

[31] Reed R. Pruning algorithms-a survey[J]. IEEE transactions on Neural Networks, 1993, 4(5): 740-747.

UBRI

引用

[32] He T, Fan Y, Qian Y, et al. Reshaping deep neural network for fast decoding by node-pruning[C]//2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014: 245-249.

[33] Abigail See, Minh-Thang Luong, Christopher D. Manning. Compression of Neural Machine Translation Models via Pruning. CoNLL 2016

[34] Gong, Yunchao, Liu Liu, Ming Yang, and Lubomir Bourdev. "Compressing deep convolutional networks using vector quantization." arXiv preprint arXiv:1412.6115 (2014).

[35] Jegou H, Douze M, Schmid C. Product quantization for nearest neighbor search[J]. IEEE transactions on pattern analysis and machine intelligence, 2011, 33(1): 117-128.

[36] Ge T, He K, Ke Q, et al. Optimized product quantization for approximate nearest neighbor search[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013: 2946-2953.

[37] Zhang X, Zou J, He K, et al. Accelerating Very Deep Convolutional Networks for Classification and Detection[J]. IEEE transactions on pattern analysis and machine intelligence, 2015.

[38] Chen D, Cao X, Wen F, et al. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013: 3025-3032.

UBRI

引用

[39] Lane N D, Georgiev P. Can deep learning revolutionize mobile sensing?[C]//Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications. ACM, 2015: 117-122.

[40] Lane N D, Bhattacharya S, Georgiev P, et al. DeepX: A software accelerator for low-power deep learning inference on mobile devices[C]//2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). IEEE, 2016: 1-12.

[41] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015.

[42] Gong Y, Lazebnik S. Iterative quantization: A procrustean approach to learning binary codes[C]//Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011: 817-824.

[43] Structured Transforms for Small-footprint Deep LearningV. Sindhwani, T. Sainath, S. KumarNeural Information Processing Systems (NIPS), to appear, 2015

[44] Han, Song, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. "EIE: efficient inference engine on compressed deep neural network." ISCA (2016).

UBRI

引用

[45]这是个ppt, CoSDEO 16上presentation: https://www.ibr.cs.tu-bs.de/Cosdeo2016/talks/invitedTalk.pdf

[46] V VanhouckeDeep et al. “Improving the speed of neural networks on CPUs”.Learning and Unsupervised Feature Learning Workshop, NIPS 2011

[47] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. Predicting Parameters in Deep Learning. In Neural Information Processing Systems (NIPS). 2013

[48] Jaderberg, Max, Andrea Vedaldi, and Andrew Zisserman. “Speeding up convolutional neural networks with low rank expansions.” arXiv preprint arXiv:1405.3866 (2014).

[49] Hwang, Kyuyeon, and Wonyong Sung. “Fixed-point feedforward deep neural network design using weights+ 1, 0, and− 1.” In 2014 IEEE Workshop on Signal Processing Systems (SiPS), pp. 1-6. IEEE, 2014.

[50] Anwar, Sajid, Kyuyeon Hwang, and Wonyong Sung. “Fixed point optimization of deep convolutional neural networks for object recognition.” In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1131-1135. IEEE, 2015.

UBRI

引用

[51] Courbariaux, Matthieu, Jean-Pierre David, and Yoshua Bengio. “Low precision storage for deep learning.” ICLR 2015.

[52] Lin, Zhouhan, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. “Neural networks with few multiplications.” ICLR 2016.

[53] Jaderberg, Max, Andrea Vedaldi, and Andrew Zisserman. “Deep features for text spotting.” In European conference on computer vision, pp. 512-528. Springer International Publishing, 2014.

[54] Rigamonti, Roberto, Amos Sironi, Vincent Lepetit, and Pascal Fua. “Learning separable filters.” In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2754-2761. 2013.

[55] Kolda T G, Bader B W. Tensor decompositions and applications[J]. SIAM review, 2009, 51(3): 455-500.

[56] Novikov A, Podoprikhin D, Osokin A, et al. Tensorizing neural networks[C]//Advances in Neural Information Processing Systems. 2015: 442-450.

[57] Nowlan S J, Hinton G E. Simplifying neural networks by soft weight-sharing[J]. Neural computation, 1992, 4(4): 473-493.

[58] Kim J, Hwang K, Sung W. X1000 real-time phoneme recognition VLSI using feed-forward deep neural networks[C]//2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014: 7510-7514.

UBRI

A Cost-sensitive Method for Learning from Crowds

Jinhong Zhong

UBRI

Outline

- Introduction
- Related work
- Cost-sensitive method for LFC
  - Cost-sensitive model
  - Determine the cost for each label
- Further Discussion
- Experiments
  - Configuration and Datasets
  - The Performance of Target Classifier
  - The Robustness to Noisy Annotator
  - Sensitiveness to Weighted Function
  - Computing Time

Introduction

- 过去几十年见证了机器学习的长足发展。不过传统的机器学习在实际应用中可能会遇到一个问题：真实类标无法获得或是获得真实的类标代价太昂贵。
- 为了克服这个问题，一种叫群体学习的新技术出现了，群体学习可以利用多个不是完全可靠的标记者的标记结果训练得到分类模型。
- 从思路上来分，解决群体学习问题的方法可以分为两大类。一类是首先估计每个样本的真实类标。然后应用传统的机器学习方法训练得到分类模型。这一类方法我们称之为二阶段方法
- 另一类方法是针对群体学习问题提出专门的学习算法，这些学习算法可以直接在群体标记结果数据上进行训练得到最终的分类模型。这一类方法我们统称为直接方法

Introduction

- 现有的这两类方法都有他们各自的缺点：对于现有的二阶段方法来说，首先在估计真实类标的过程中不会关注每个样本的特征值是什么，因此无法利用样本特征值与样本类标之间的内在关系。其次，在估计完真实类标之后，二阶段方法在训练目标分类器过程只关注每个样本估计得到的类标，而无法利用其他标记结果的信息。

- 而对于现有的直接方法，都需要事先假定标记者标记结果的分布，从而建立起一个条件概率模型。比如[6]中假设每个标记者的标记结果都可以通过两个贝努利分布来描述，[9]中把个人分类器的Logistics回归当做标记者标记结果的分布。这类方法的缺点比较明显，首先标记结果的分布需要事先假设，而且在实际应用中无法得知这些假设是否符合真实情况。其次这类方法得到的最终优化问题都很复杂，为使问题可解，通常仅限于线性分类器，很难推广到决策树、神经网络或是其他复杂的非线性分类器。最后，即使在以上重重限定下，最终得到的优化问题经常还是非凸的。

Introduction

- 群体学习最大的难点是如何鉴别每个标记结果的可靠性以及在训练最终分类模型的时候如何合理利用这些可靠性信息。针对这两个问题，本文从一个全新的角度（Cost-sensitive）出发，提出了一种费用敏感方法用于解决群体学习问题。该方法首先估计每个标记的可靠性，并根据标记的可靠性确定每个标记与最终分类器的预测值相悖的惩罚力度，从而得到一个费用敏感模型。最后通过求解这个模型得到最终的分类器。

- 代价敏感的群体学习算法有许多优点：1. 在训练最终分类器时可以保留所有标记结果以及他们的可靠性信息；2. 对标记者标记结果的分布不做具体先验假设；3.最终分类器不受任何限定，因此极大地扩宽了群体学习的应用领域；4. 不用求解非凸问题

Related Work

- 群体学习刚出现的时候研究者的焦点主要放在如何估计真实的类标。这类工作按照建模的角度看又可以分为两类，一类是判别式方法。这类方法关注于获得一个判别式，这个判别式的输入是每个标记者对一个样本的标记结果，输出是估计得到的类标。这类方法的例子有MV方法以及基于MV的方法 [1][2][3]。另一类是生成方法，这类方法关注于建立当前观测到的标记结果的条件概率与每个样本的真实类标和标记者可靠性之间的关系。这类方法的例子有Dawid-Skene (DS) estimator [16], the minimax entropy (Entropy) estimator [24, 25]以及他们的变种。

- 这类方法解决的问题是估计已知样本的真实类标，无法获得一个能够对未来新数据的类标进行预测的模型。对于这类方法来说要获得预测模型需要额外再加入学习模块。比如在估计完真实类标后再加入一个训练的过程，我们称这类方法为二阶段方法。但这类方法有个明显的劣势： 由于在估计真实类标的过程中不会引入分类器，因此无法利用样本特征值与样本类标之间的内在关系。其次在学习的过程中由于只保留了估计后的类标，所以会有信息损失。

Related Work

- 随着群体学习的发展，越来越多人把关注点放在了直接基于标记结果建立分类模型。目前这类方法的做法都需要事先为每个标记者假定其标记结果的分布，通过这个分布建立起样本特征值与每个标记者标记结果的关系。进而获得当前观测到标记结果的条件概率，通过最大化这个条件概率得到目标分类器以及每个标记者的标记结果分布的参数。

- Raykar et al.[6] 首先假定每个标记者的标记结果可以用两个贝努利分布刻画，Yan yan et al. [7]放松了[6]中标记者标记结果分布与样本特征值无关的假设，但又引入了标记者标记准确率与样本特征值呈线性关系的假设。 Wei bi et al.[8]进一步引入了样本困难度以及标记者的奉献程度来刻画可靠性模型 。Kajino et al.[9] 第一个把个人分类器的Logistic回归当做标记者标记结果的分布。这种方法将分类器限定为线性分类器后可以使得最终的优化问题是个凸优化。由于[9]无法很好的区别标记者之间不同的可靠性，Valizadegan et al.[10] 引入了两个参数来进一步刻画标记者之前的区别，一个是用于刻画一个标记者标记结果与其个人分类器预测结果的一致程度，另一个参数用于刻画标记者标记结果与最终分类器的预测结果的一致性。 Kajino et al.[11] 利用聚类将所有的标记者聚成几种不同类型的标记者。

Related Work

- 这类方法的缺点比较明显，首先标记结果的分布需要事先假设，而且在实际应用中无法得知这些假设是否符合真实情况。其次这类方法得到的最终优化问题都很复杂，为使问题可解，通常仅限于线性分类器，很难推广到决策树、神经网络或是其他复杂的非线性分类器。最后，即使在以上重重限定下，最终得到的优化问题经常还是非凸的。

- 由于众包的发展，群体学习的应用越来越广泛群。体学习技术在不断发展的同时，群体学习与其他学习技术也结合的越来越多。比如与主动学习相结合形成了主动群体学习，与在线学习相结合形成了在线群体学习，与排序相结合形成了群体排序学习。

Cost-sensitive Model

- 在传统的机器学习中，分类模型都是通过最小化由正则项和损失函数构成的优化目标获得：

$$f = \underset{f \in H}{\operatorname{argmin}} \frac{1}{2} \|f(\mathbf{x})\|_H^2 + C \sum_i l(f(\mathbf{x}_i), z_i) \qquad (1)$$

其中$\frac{1}{2} \|f(\mathbf{x})\|_H^2$是正则项，$\sum_i l(f(\mathbf{x}_i), z_i)$是损失函数。

Cost-sensitive Model

- 在传统的机器学习中，分类模型对样本的预测应该尽量与样本的类标相同，如果有误差，则根据误差给予相应的惩罚。在群体学习中也一样，分类模型对样本的预测应该尽量与样本的类标相同。不同的是在群体学习中我们无法获知真实的类标，而标记者标记结果中每个类标的可信度是不一样的。有些可信度比较高，如果分类器在这个样本上的预测结果与它存在误差，则给予的惩罚应该大一些，反之，对于可信度比较低的类标，惩罚力度应该小些。根据上面的分析，我们可以将传统的机器学习改造成代价敏感的模型，使之可以处理群体学习问题：

$$f = \underset{f \in H}{\operatorname{argmin}} \frac{1}{2} \|f(\mathbf{x})\|_H^2 + C \sum_t \sum_{i \in \mathbb{I}^t} w_i^t l(f(\mathbf{x}_i), y_i^t) \qquad (2)$$

其中$\mathbb{I}^t$是第t个标记者标记的所有样本。w_i^t是类标y_i^t的代价，用于控制模型在该类标上出现误差将给予的惩罚力度。

Cost-sensitive method for LFC

- 从⑵中可知,二阶段方法是代价敏感方法的一个特例: $w_i^t = \frac{[y_i^t = \hat{z}_i]}{\sum_t [y_i^t = \hat{z}_i]}$。其中$\hat{z}_i$是对第$i$个样本类标的估计。在二阶段方法中，每个样本估计的类标的权重之和为1。而在代价敏感模型中，w_i^t的取值可以是非负的任意值。

Determine the Cost for Each Label

- 根据(2)可知，如果w_i^t一旦确立，那么最终的分类器就可以通过求解(2)获得。

- 如何定w_i^t方法并不是唯一的，它的作用就是体现不同类标之间重要性不同。不过确定w_i^t的时候也不是没有任何限制的，它应该跟该类标的可靠性$r_i^t = \mathrm{P}(z_i = y_i^t)$有关，记$w_i^t = \mathrm{q}(r_i^t)$。事实上，一个理想的$\mathrm{q}(x)$应该满足如下两个条件：

  - 权重函数$\mathrm{q}(x)$ 在[0,1]上是单调递增的。显然，如果分类器在一个比较可靠的类标上犯错了惩罚力度要大一些才合理；

  - 权重函数$\mathrm{q}(x)$是关于$(0.5,\ 0)$中心对称的。这是因为$r_i^t = 0.5$的时候意味着y_i^t与真实类标相等和与真实类标相反的概率都是0.5。也就是说y_i^t等价于一个纯随机产生的类标。对于这种类标，分类器的预测值与该类标不同也不应该给予惩罚

Determine the Cost for Each Label

- 需要注意的是因为函数$y=\mathrm{q}(x)$过(0.5, 0)且$\mathrm{q}(\mathrm{x})$是单调递增的。所以$r_i^t < 0.5$的时候$w_i^t < 0$,而根据cost的定义，不应该出现负的cost。在这种情况下y_i^t与真实类标相等的概率小于0.5，与真实类标相反的概率大于0.5.相对于y_i^t来说，真实类标是$-y_i^t$的概率更大。所以我们可以对y_i^t取反，新的可靠性就成了1-r_i^t。新的w_i^t就大于0了。

- 满足上述条件的函数有很多，可以选择自身就带有这些性质的函数，比如$a(x-0.5)$,$log_2\left(\frac{x+a}{1-x+a}\right)$，其中$a$为任意正数。也可以先构造在[0.5, 1]区间上满足上述条件的函数，然后根据中心对称原则将函数在[0,0.5]区间上进行延伸。比如本文选择具有指数增长的函数作为$\mathrm{q}(x)$，因为要过(0.5, 0)，所以$\mathrm{q}(x)$在[0.5, 1]上可以选择$e^x - e^{0.5}$,根据中心对称原则，$\mathrm{q}(x)$在[0,0.5]上则为$-(e^{1-x} - e^{0.5})$

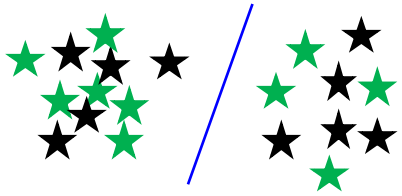
Determine the Cost for Each Label

- 现在最后一个问题就是如何确定类标的可靠性r_i^t。由于每个样本的类标和真实分类器我们都是不清楚的。所以很难直接估计r_i^t。

- 当我们在一份数据集上学习一个分类模型的时候，我们就已经默认样本特征与真实类标之前是有一定规律的，比如在我们选择线性模型当做分类模型的时候我们就默认样本特征与真实类标之前是线性关系的。那么对于比较可靠的标记者来说他的标记结果与真实数据差别很小，所以数据之间的规律性比较好地被保留下来。而对于很不可靠的标记者来说，他的标记结果和真实数据分布相差比较大，随机性比较强，数据之间的规律性被破坏地比较严重。下图是说明这两种不同情况的一个例子。图中绿色和黑色的点代表不同类别的样本，蓝色的线为真实的分类面

(a).真实样本类标

(b).比较可靠标记者标记结果



(c).比较不可靠标记者标记结果

Determine the Cost for Each Label

- 受以上分析的启发，我们可以将某个标记者t标记结果分成训练集和测试集，在训练集上训练得到标记者t的分类器$f^t(\mathbf{x})$, 并将测试集测试得到的准确率r^t当做标记者整体可靠性的一个估计。比如我们有理由相信图(b)得到的r^t会比图(c)得到的r^t要高。

- r^t是对标记者t整体可靠性的一种估计，而对于一个具体的标记结果的可靠性，除了要考虑这个标记者是否可靠，还要考虑这个标记结果是不是这个标记者意外标记错误。以图(b)为例，假设最终得到这个标记者的可靠性$r^t = 0.9$，绿色样本中间的黑色样本显然不能用0.9当做其可靠性的估计。这背后揭示了一个道理：在我们得到了一个标记者整体可靠性估计后，要估计某一个样本的可靠性的话需要考虑这个样本的标记结果是否离群（与$f^t(\mathbf{x})$不同）。在本文中，我们用如下公式进行可靠性估计：

$$r_i^t = (r^t)^{[f^t(\mathbf{x}_i)=y_i^t]}(1-r^t)^{[f^t(\mathbf{x}_i)=-y_i^t]}$$

Determine the Cost for Each Label

- 为了估计更加准确，我们采用bagging的方式对每个标记者都训练得到多个分类器$\{f_k^t(\mathbf{x}_i)\}_{k=1}^K$以及他们的测试准确率$r^{t,\mathrm{k}}$。估计$r_i^t$的时候综合这K个分类器的结果：

$$r_i^t = \frac{1}{\Phi}\prod_{k=1}^{K}(r^{t,\mathrm{k}})^{[f_k^t(\mathbf{x}_i)=y_i^t]}(1-r^{t,\mathrm{k}})^{[f_k^t(\mathbf{x}_i)=-y_i^t]}$$

- 其中Φ是归一化常数项：

$$\Phi = \sum_{y}(\prod_{k=1}^{K}(r^{t,\mathrm{k}})^{[f_k^t(\mathbf{x}_i)=y]}(1-r^{t,\mathrm{k}})^{[f_k^t(\mathbf{x}_i)=-y]})$$

- 至此，问题⑵就已经完全确定了。最终分类器可通过求解⑵获得。

Further Discussion

- **简化优化问题：**在w_i^t计算之后，最终分类器需要通过求解⑵获得。如果每个标记者都标记了所有的样本，那么一共有nT个类标。相当于在具有nT个样本的训练集上训练分类模型，每个样本都赋予了不同的权重。在n和T都比较大的时候，训练样本数量会变得非常多。这将会使求解⑵变得非常费时。事实上， 问题⑵可以通过将相同样本相同类标的项进行合并使优化问题变得更加简化。

$$\begin{aligned}
&\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}^t} w_i^t l(f(\mathbf{x}_i), y_i^t) \\
=&\sum_{t\in\mathcal{T}}(\sum_i w_i^t l(f(\mathbf{x}_i), y_i^t)[y_i^t = 1] + \sum_i w_i^t l(f(\mathbf{x}_i), y_i^t)[y_i^t = -1]) \\
=&\sum_i(\sum_{t\in\mathcal{T}} w_i^t [y_i^t = 1] l(f(\mathbf{x}_i), 1) + \sum_{t\in\mathcal{T}} w_i^t [y_i^t = -1] l(f(\mathbf{x}_i), -1)) \\
=&\sum_i u_i l(f(\mathbf{x}_i), 1) + \sum_i v_i l(f(\mathbf{x}_i), -1)
\end{aligned}$$

其中 $u_i = \sum_{t\in\mathcal{T}} w_i^t [y_i^t = 1]$ $v_i = \sum_{t\in\mathcal{T}} w_i^t [y_i^t = -1]$.

Further Discussion

- 可以看出化简后的loss function只剩下2n项了。将其代入公式⑵就相当于在一个含有2n个样本的训练集上训练一个分类模型。其中n个正样本n个负样本。第i个正样本和第i个负样本的特征向量都是$\mathbf{x}_i$，第i个正样本的权重是u_i，第i个负样本的权重是v_i.

$$f = \underset{f \in H}{\operatorname{argmin}} \frac{1}{2} \|f(\mathbf{x})\|_H + C\left(\sum_i u_i l(f(\mathbf{x}_i), 1) + \sum_i v_i l(f(\mathbf{x}_i), -1)\right) \quad (3)$$

Further Discussion

- **问题求解：**将简化后的loss function代入(2)之后得到(3)，最后需要做的就是求解(3)。求解(3)的方式有两种。一种方式是直接求解，以SVM为例，简化后的(3)的对偶问题如下：

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2}\sum_{i=1}^{2n}\sum_{j=1}^{2n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) + \sum_{j=1}^{2n}\alpha_i$$

$$\begin{aligned}
\mathbf{x}_{n+i} &= \mathbf{x}_i & i \le i \le n \\
y_i &= 1 & 1 \le i \le n \\
y_i &= -1 & n+1 \le i \le 2n \\
0 \le \alpha_i &\le u_i & i \le i \le n \\
0 \le \alpha_i &\le v_i & n+1 \le i \le 2n
\end{aligned}$$

其中$K(\cdot,\cdot)$是SVM的核函数。直接求解这个对偶问题就得到了最终的分类器。

Further Discussion

- 对于有些分类方法，无法直接求解这种每个样本带有不同权重的问题。在这种情况下，可以通过按权重等比例复制样本数量的方式实现加权的目的。比如按照权重的10倍复制，如果u_i=0.3 $v_i = 0.5$，那么就复制3份$(\mathbf{x}_i, 1)$，5份$(\mathbf{x}_i, -1)$加入到训练样本中。复制完所有样本后便得到了最终的训练集。在这上面进行训练也可以得到最终的目标分类器。

Experiments

- 在做实验的时候，本文选择3个UCI数据集和2个真实数据作为实验数据。通过实验部分，我们打算研究如下问题：

  1. 本文提出的算法在这些数据集上得到的目标分类器性能如何；
  2. 本文提出的算法对于低质量的标记者是否具有鲁棒性；
  3. 选择估计r_i^t的不同方法进行比较，看本文估计r_i^t的方法是否有优势；
  4. 本文提出的算法花费的计算时间如何。

Experiments

研究这些问题的原因：

- 在选择一个群体学习算法时，我们除了会关心它是否对分类器有限制以及是否涉及非凸优化，我们还会关心这个算法是否能获得性能良好的分类器（问题1）。
- 由于群体学习中让标记者标记的系统通常是很开放的，这就有可能导致很多很低质的标记者进入标记系统，甚至有些人会用机器去标注以获得标记报酬。这些都会给标记结果带来很多的噪声。所以算法是否能够在大量的低质量标记者的情况下保持鲁棒性也是群体学习算法非常重要的一项指标(问题2)；
- 由于本文提出的算法涉及到r_i^t的估计，而r_i^t的估计也可以选择一些其他现有方法，那么在选择本文提出的估计r_i^t的方法之前需要回答这种估计方法是否比现有的方法更具优势
- 最后，对于所有算法，在实际应用中，算法的计算代价都是需要考虑的（问题4）

Configuration

- 在本文实验中，我们选择了4种方法作为对比算法，用于研究本文提出的方法在不同指标上的表现。这四种方法分别是：
  - Majority Voting, 先通过Majority Voting估计真实类标，然后再训练得到目标分类器；
  - M3V, 先通过Tian et, al. [12] 提出的M3V方法估计真实类标，然后再训练得到目标分类器；
  - Raykar's Model。Raykar et, al. [6] 提出的群体学习算法，把真实类标当做中间变量，求解目标分类器；
  - Kajino's Model。Kajino et, al. [9] 提出的群体学习算法，把个人分类器当做中间变量，求解目标分类器；

- 在比较不同算法的各种指标的时候，所有的实验都会重复跑20遍并取其平均值。

Dataset

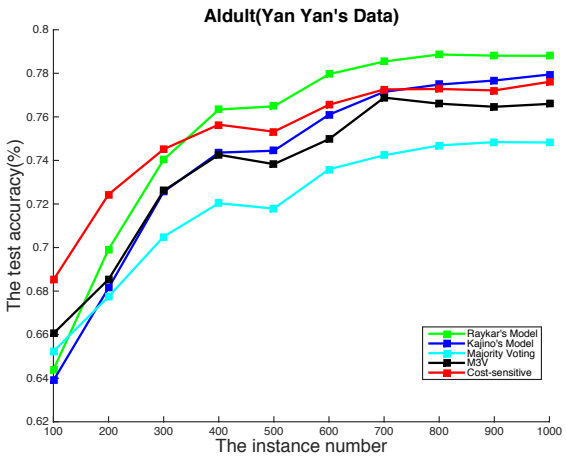
- 在本实验中一共用到了6个数据集，其中三个UCI的数据集：Conect-4, Adult 和 mushroom。还有三个是真实Twitter 数据集, 这两个数据集都有多个真实标记者标记过。一个在jinhong et, al. [13]中使用过 ，是关于Twitter话题的。一个是Kajino et, al.[9]使用过，是关于Twitter 实体名字识别的。一个在tiantian [12] 中对花进行分类的数据，数据基本情况介绍如下：

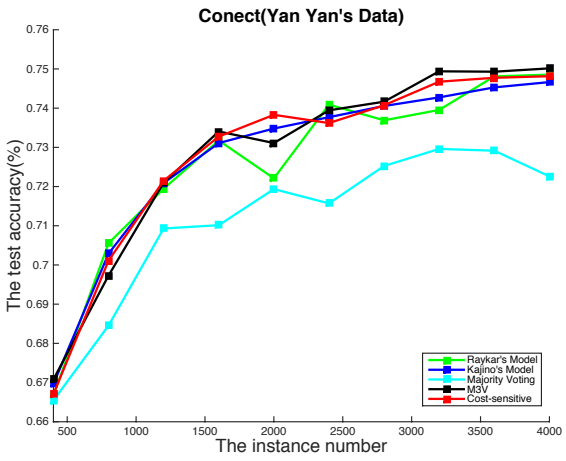
数据集名称	数据维数	随机选择的样本数	随机选择的测试样本数
Conect-4	126	4000	2000
Adult	123	1000	2000
Mushroom	112	1000	1000
Twitter Topics	295	1000	1000
Twitter Name Entity Recognition	161904	1400	1400
Flower	4096	200	200

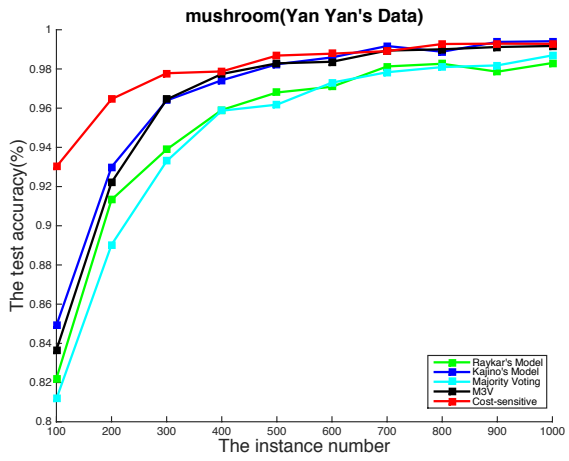
Dataset

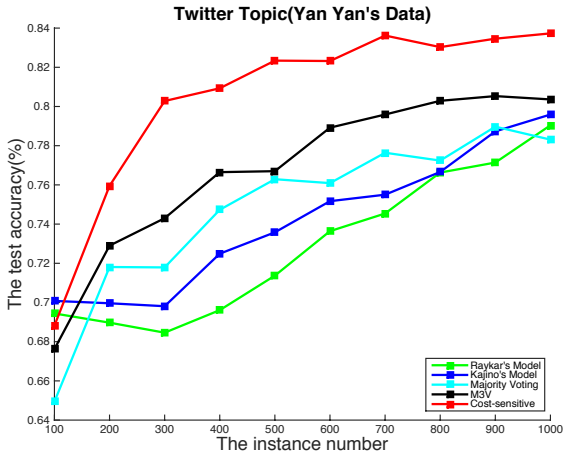
- 为了产生更加多样化的数据，以更全面地测试算法的性能，除了真实数据之外，我们还采用了三种标记者标记数据的方式，他们分别是：
  - 聚类模型：通过聚类决定每个标记者的擅长领域，其他样本则为其不擅长领域，对于其擅长的样本，标记者t能够正确标记能够以P^t的准确率标记正确，对于她/他不擅长的样本，每个标记者能够正确标记的概率是p^t；
  - Raykar模型：和Raykar et, al.[6]类似，不过更加一般化，我们在产生数据的时候每个标记者在每个样本上的准确率P_i^t都是独立随机产生的。
  - Whiltehill 模型：和Whilehill et, al.[14] 与 Zhao et al [15]等一样，每个标记者有一个参数表示其专业程度，每个样本有一个参数表示其困难程度，在实验的时候所有标记者的专业程度和所有样本的困难程度都是由N(1,1)随机产生

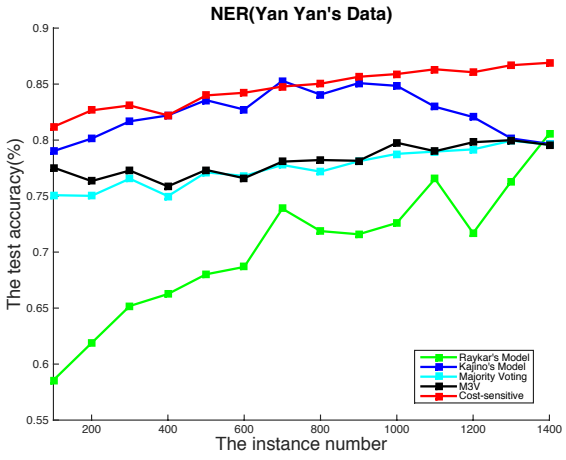
The Performance of Target Classifier(Yan Yan's Data)

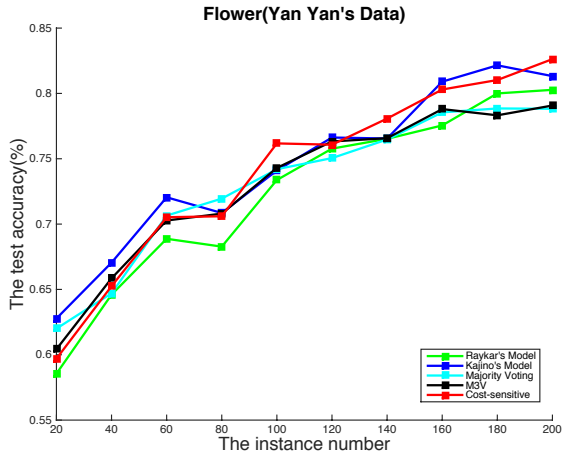






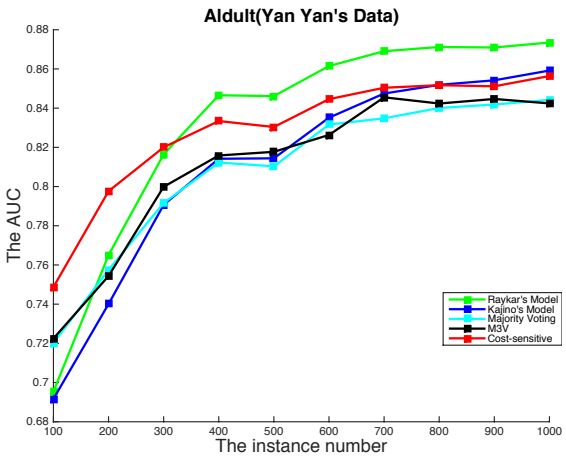


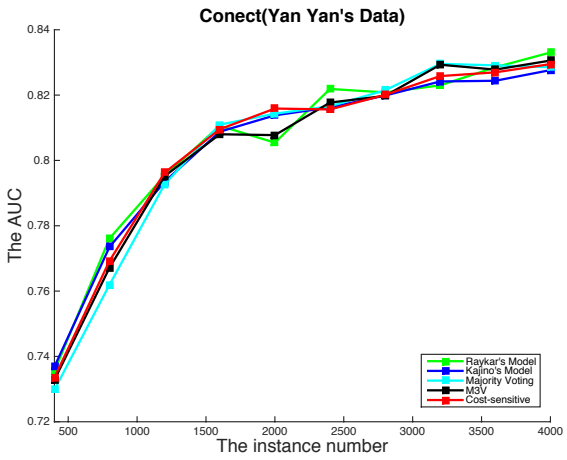


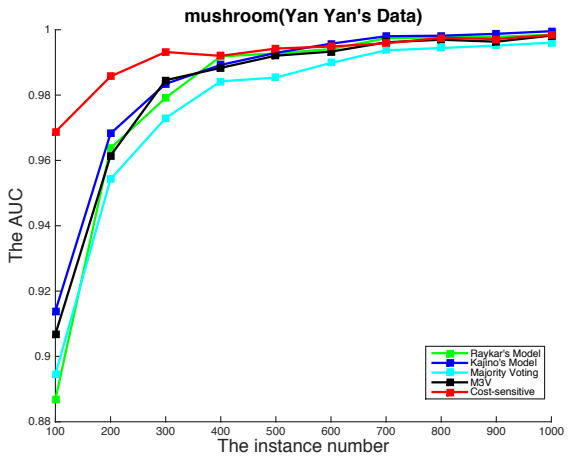


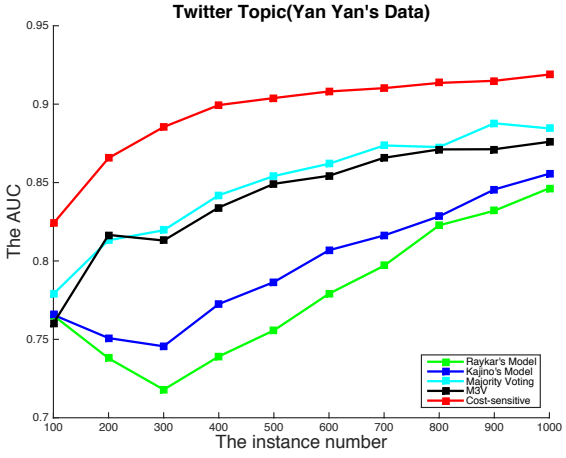
UBRI

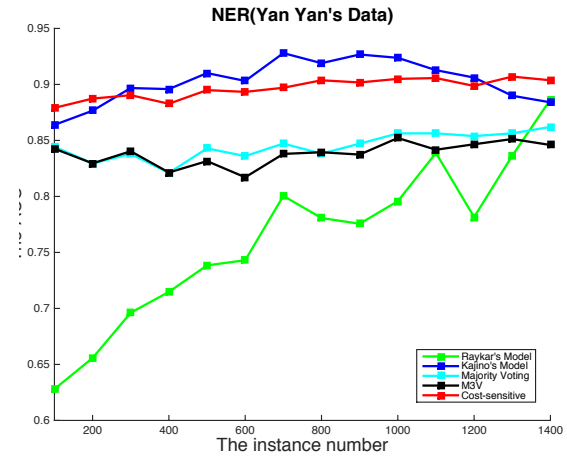
The Performance of Target Classifier(Yan Yan's Data)

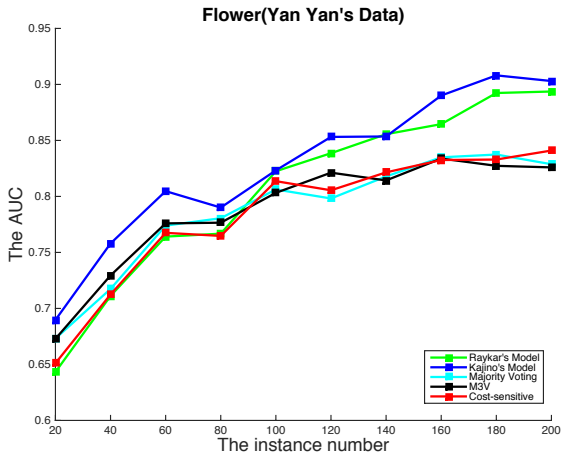




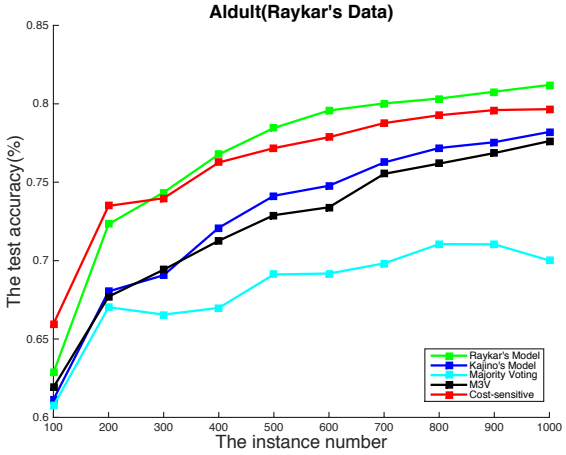


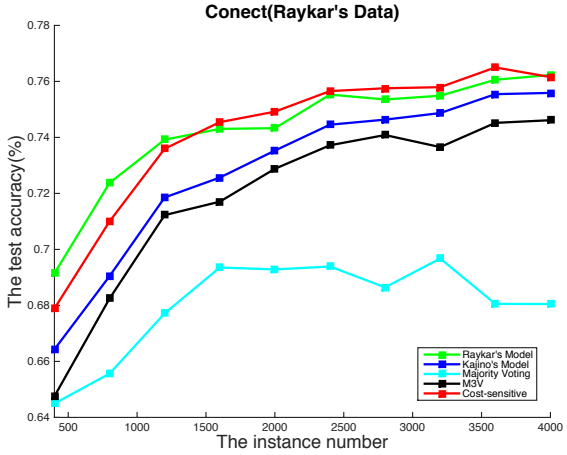


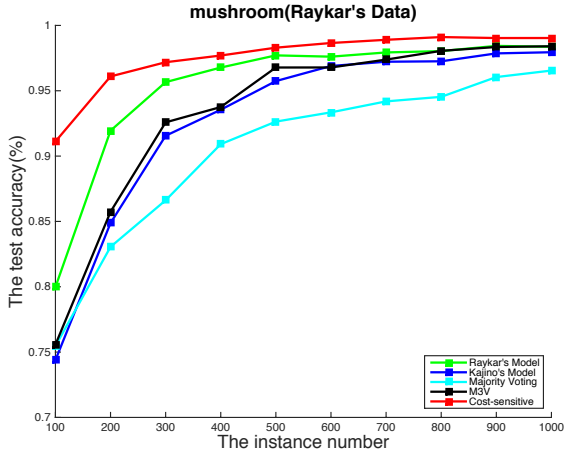


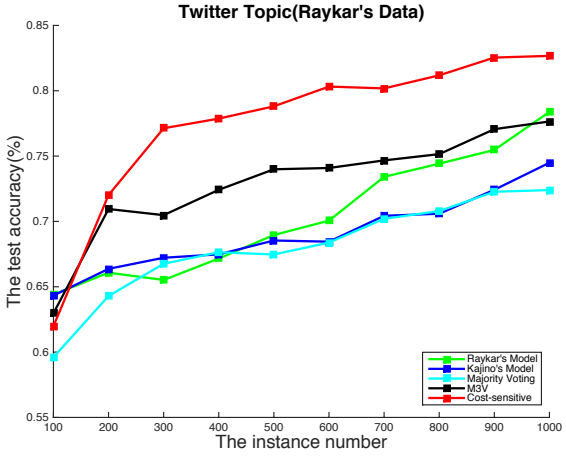


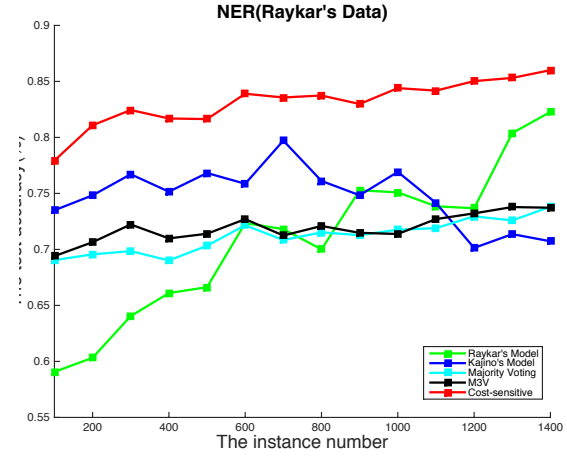
The Performance of Target Classifier(Raykar's Data)

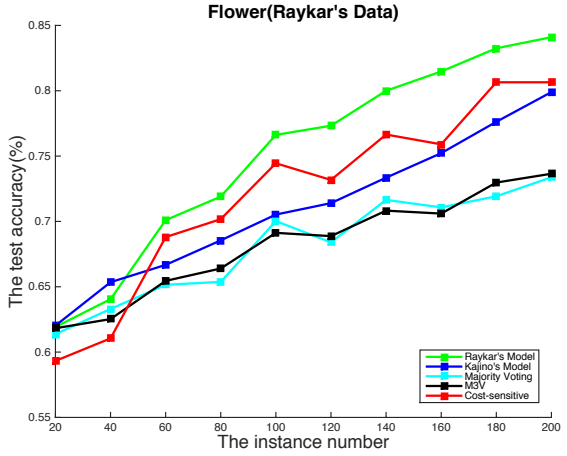




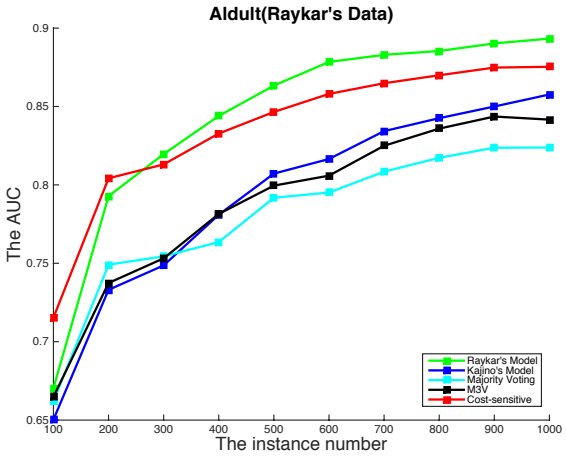


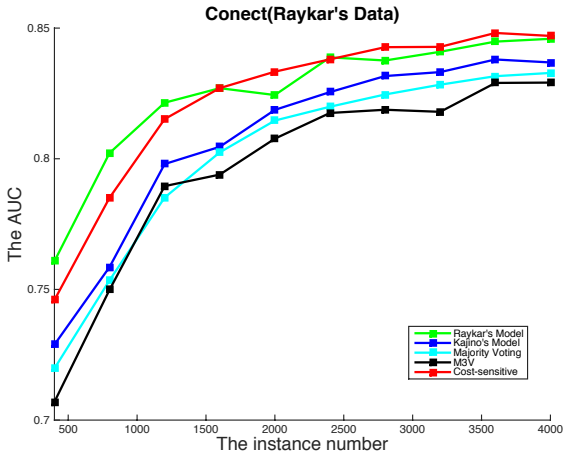


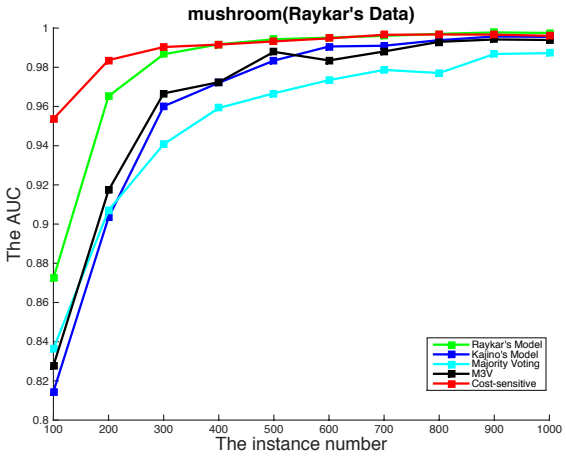


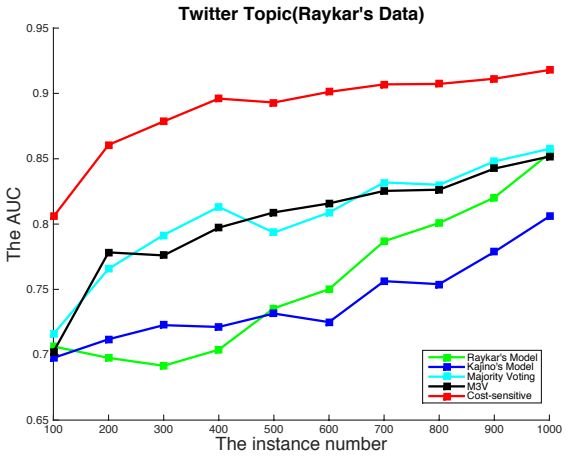


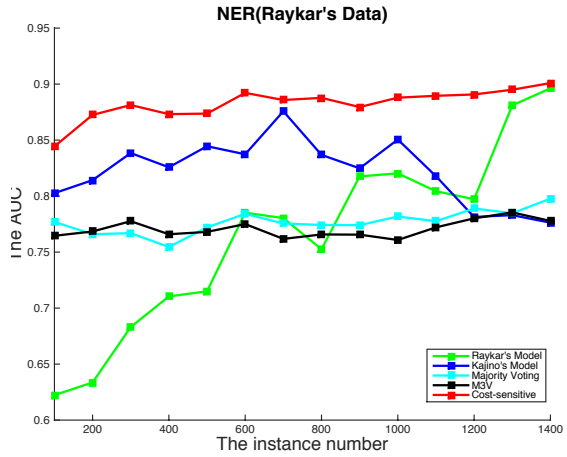
The Performance of Target Classifier(Raykar's Data)

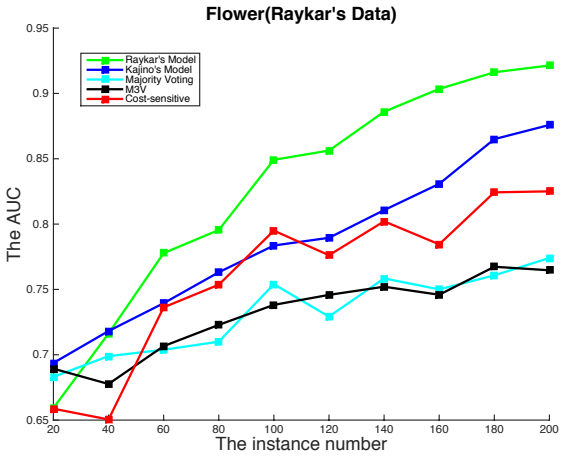




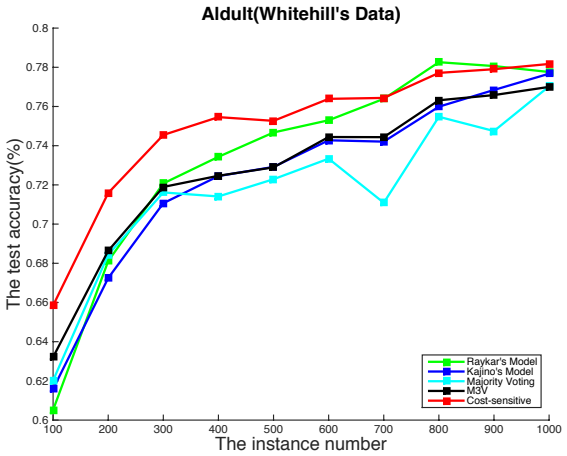


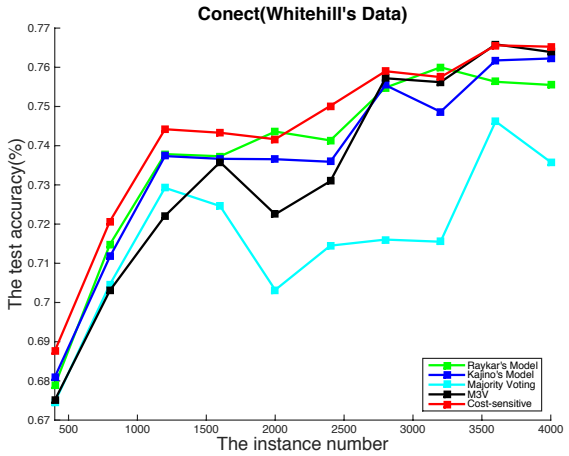


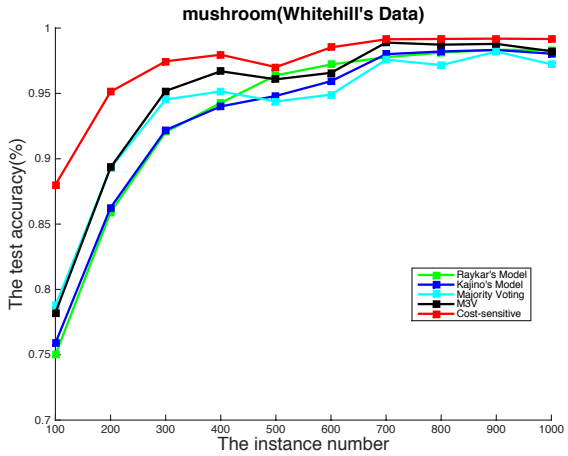


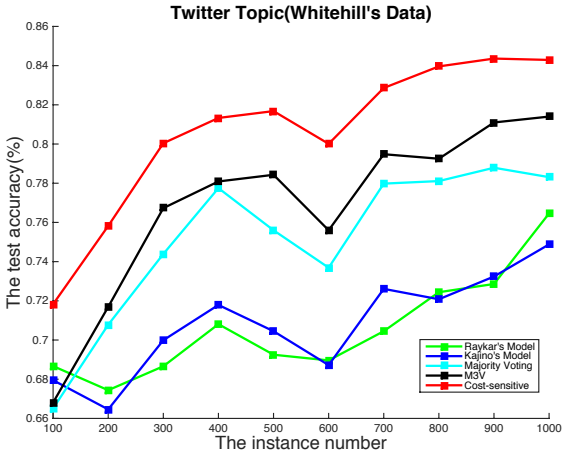


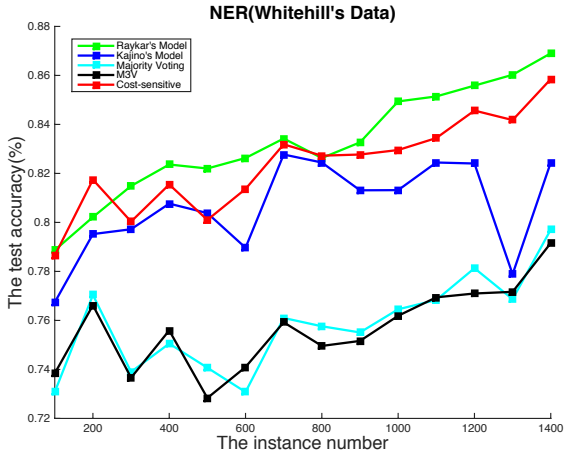
The Performance of Target Classifier(Whitehill's Data)

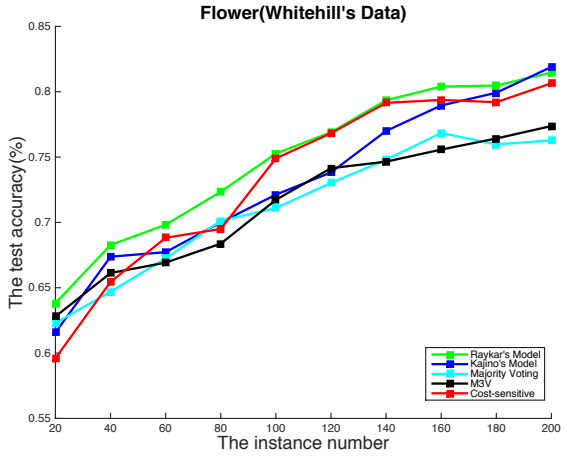




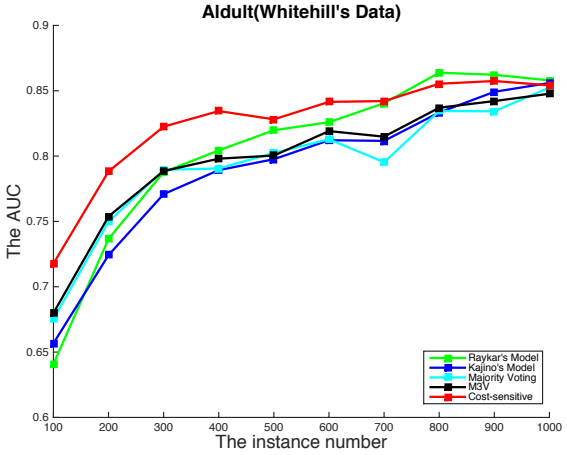


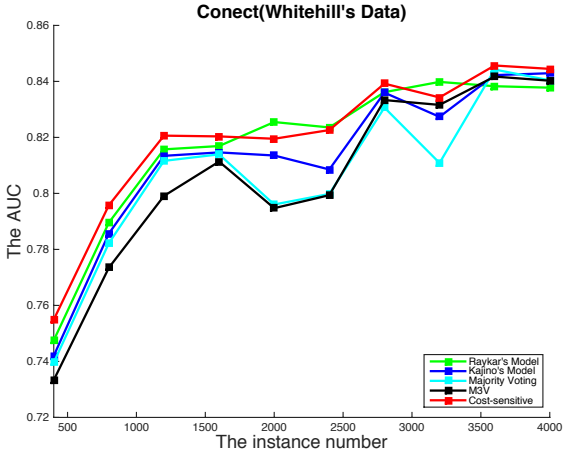


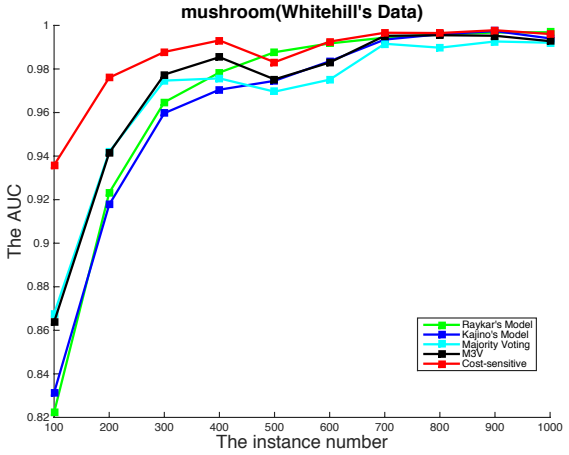


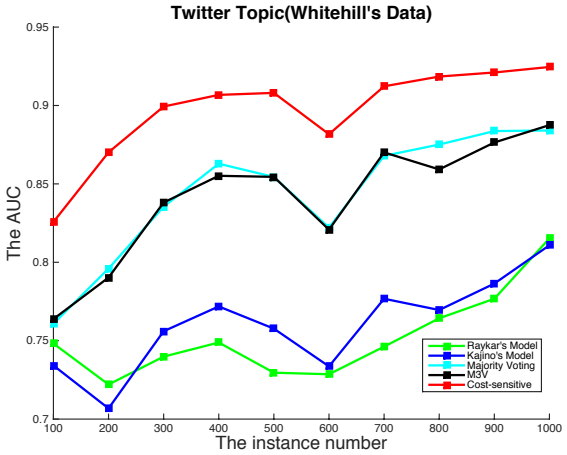


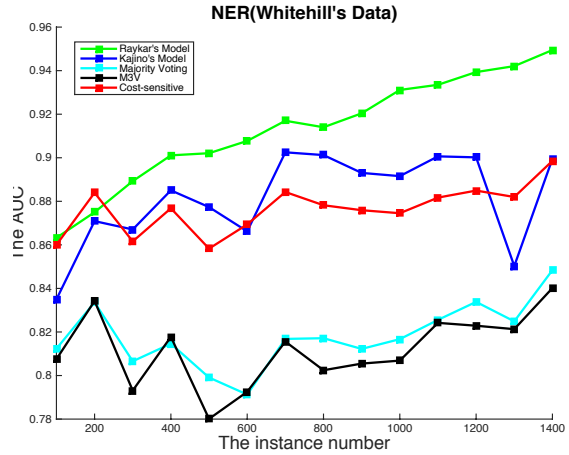
The Performance of Target Classifier(Whitehill's Data)

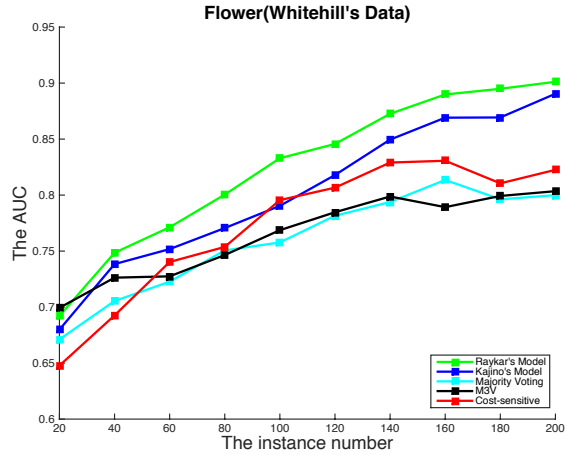




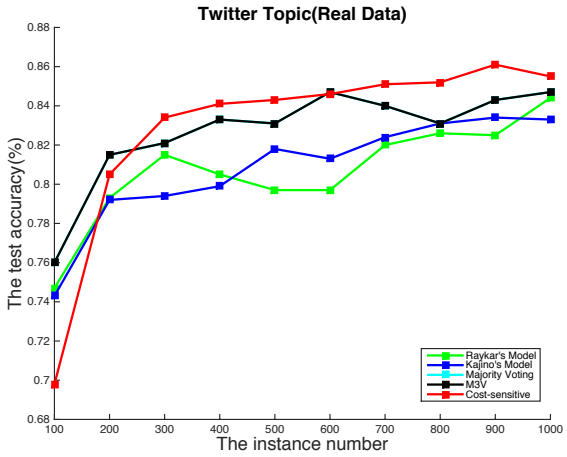


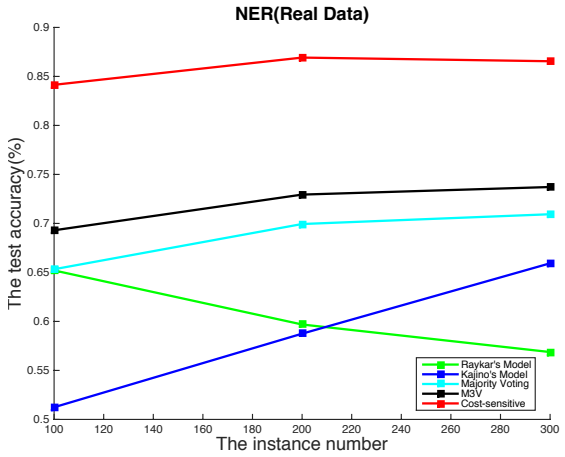


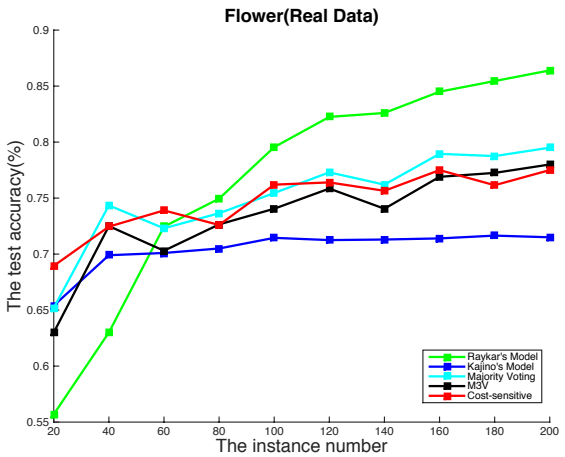


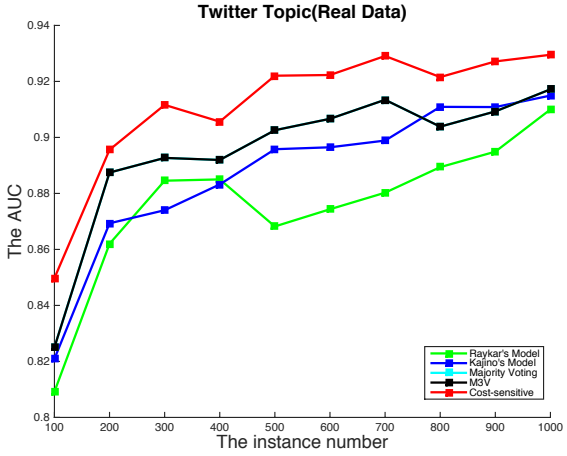


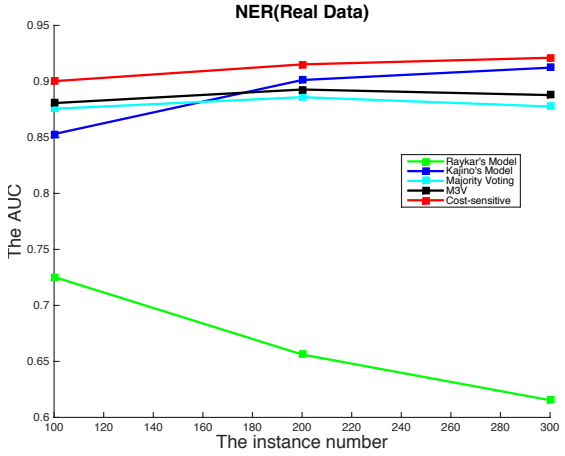
The Performance of Target Classifier(Real Data)

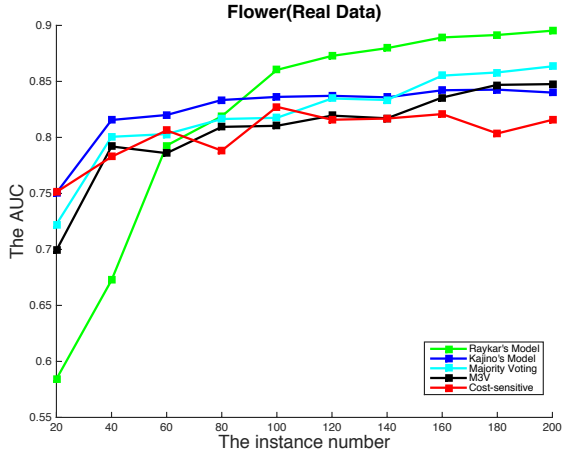




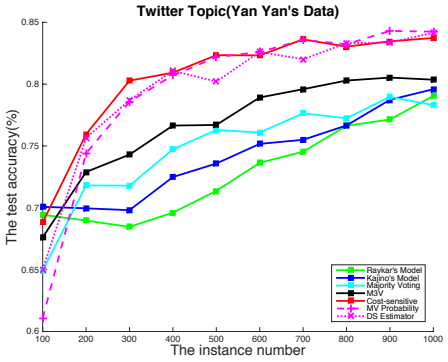


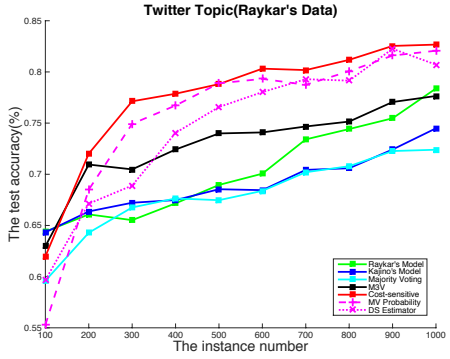


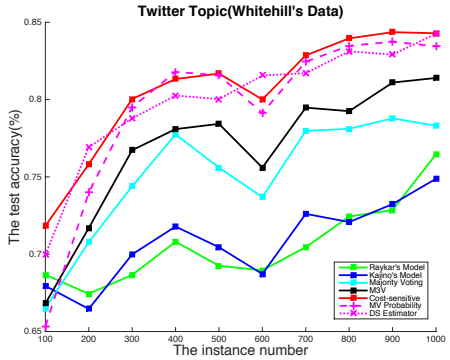


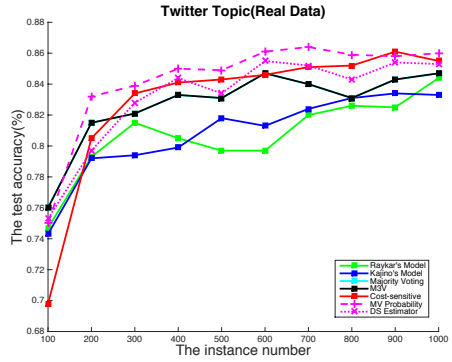


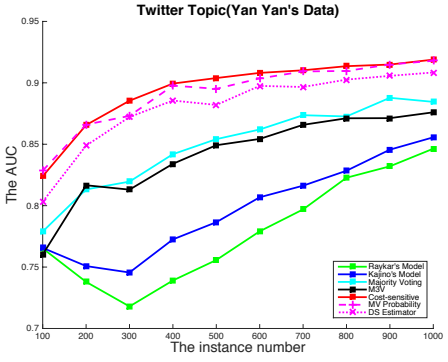
Sensitiveness to Weighted Function

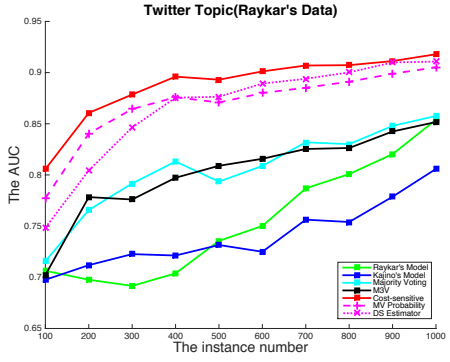


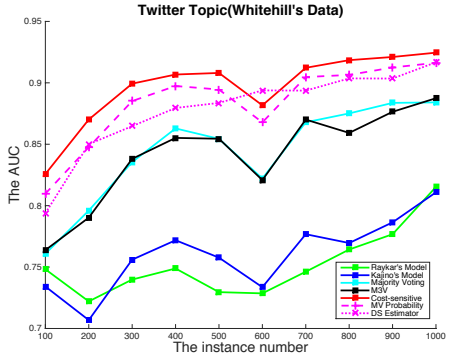


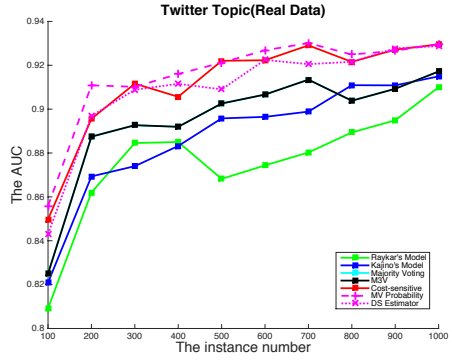




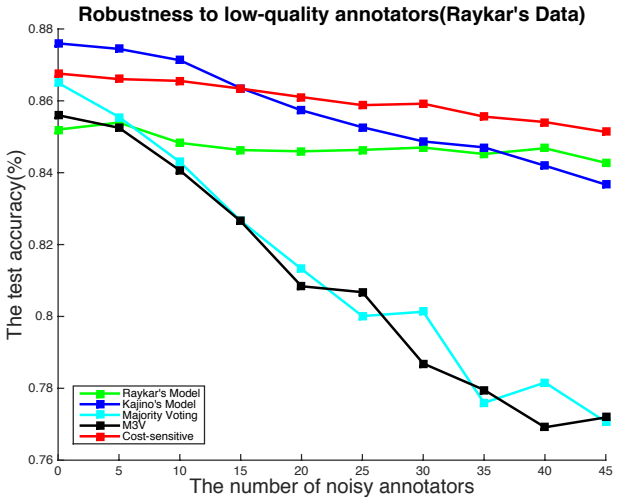


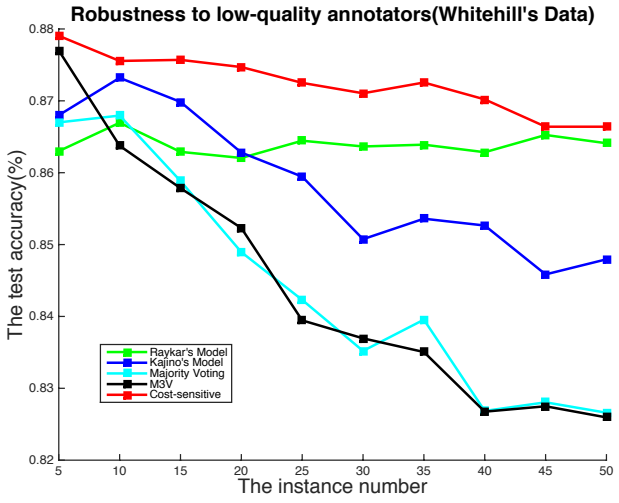


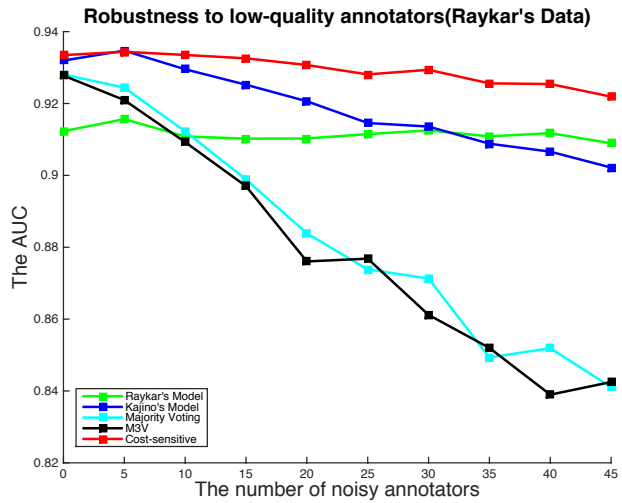


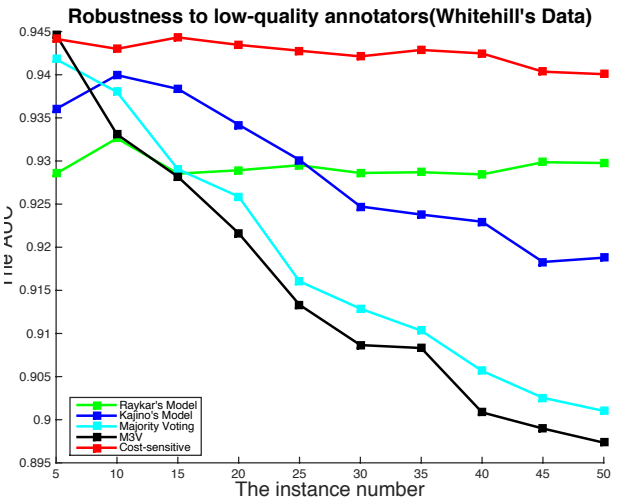


The Robustness to Noisy Annotators



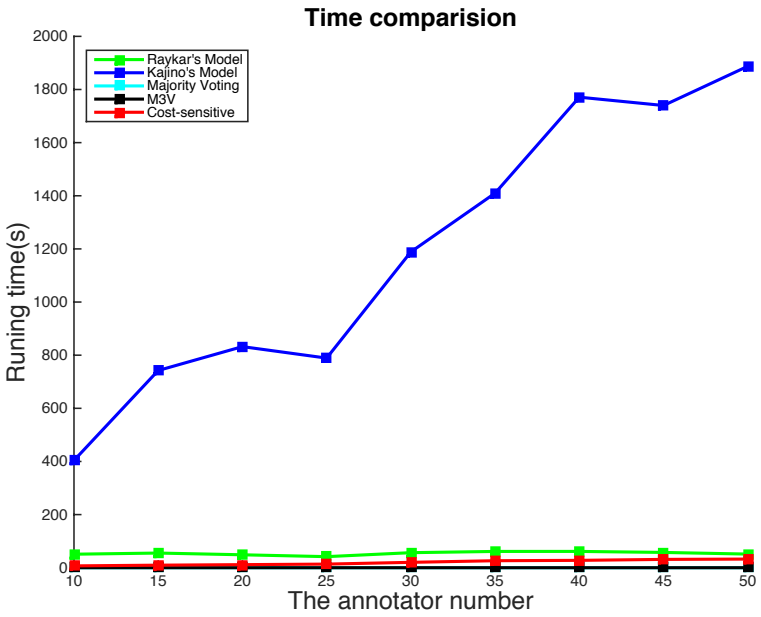


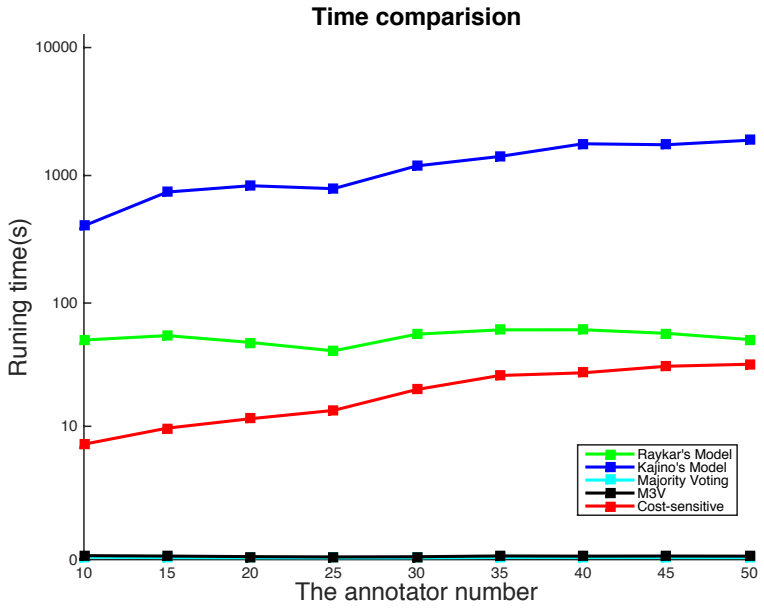




随着随机标记者数量的增加，Kajino's Model, Majority Voting,与 M3V的结果都急剧下降，Raykar's Model 和 Cost-sensitive 对于随机标记者的鲁棒性比较好，其中又以 Raykar's Model表现最好，随着随机标记者数的增加，性能基本保持不变。Cost-sensitive的结果随着随机标记者数的增加性能出现了一定的下降，但是显著优于其他三种算法。

Computing Time





上图面两张图描述的是不同算法计算时间的对比，右图是将左图中纵坐标变换成指数级增长后的结果

Computing Time

- 从上图可以看出，所有对比算法中Kajino's Model 不管从计算花费总时间还是随着标记者数量增加计算时间的增长速度来看，它的表现都是最差的。这在Kajino's Model最终需要求解的优化问题中可以找到答案：它最后的求解问题中待求解参数数量是(T+1)*d。随着标记者数量的增加，求解问题的规模也在不断变大。

- Raykar's Model的计算时间在这个实验中一直是第二花费时间的算法，但它的优点是计算时间比较稳定，随着标记者数量的增加，总计算时间不怎么增加。这是因为Raykar's Model的最终优化问题中的参数个数是2n+d，与标记者数量无关。但因为优化问题是个非凸问题，求解非常费时，所以即使在标记者数量很少的情况下总计算时间也非常长。

Computing Time

- Cost-sensitive的计算时间排在第三，且可以看出，它的计算时间与标记者数量呈线性增长关系，这是因为该算法大部分的时间花在bagging多个分类器上面，而需要bagging的分类器个数是KT个。可以预见的是随着标记者数量的增长，Cost-sensitive的计算时间将会超过Raykar's Model，但在标记者数量不是特别大的情况下，它的计算时间不算非常长。

- 在所有对比算法中Majority和M3V算法的计算总时间和随着标记者数量增长的变化都是表现最好的。不仅总计算时间最少而且增长也不明显。这是因为他们没有挖掘样本特征值与样本真实类标之间的关系的机制。而在其他三个对比群体学习算法中，挖掘这个联系是最重要也是最费时的。

Reference

[1] Victor S Sheng. Simple multiple noisy label utilization strategies. In Data Mining (ICDM), 2011 IEEE 11th International Conference on , pages 635–644. IEEE, 2011.
[2] Faiza Khan Khattak and Ansaf Salleb-Aouissi. Quality control of crowd labeling through expert evaluation. In Proceedings of the NIPS 2nd Workshop on Computational Social Science and the Wisdom of Crowds , 2011.
[3] Tian Tian and Jun Zhu. Max-margin majority voting for learning from crowds. In Advances in Neural Information Processing Systems , pages 1612–1620, 2015
[4] P. Welinder, S. Branson, P. Perona, and S. J. Belongie. The multidimensional wisdom of crowds. In NIPS , pages 2424–2432, 2010.
[5] Qiang Liu, Jian Peng, and Alexander T Ihler. Variational inference for crowdsourcing. In Advances in Neural Information Processing Systems , pages 692–700, 2012.
[6] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. The Journal of Machine Learning Research , 11:1297–1322, 2010.
[7] Yan Yan, R´omer Rosales, Glenn Fung, MarkWSchmidt, Gerardo H Valadez, Luca Bogoni, Linda Moy, and Jennifer G Dy. Modeling annotator expertise: Learning when everybody knows a bit of something. In International conference on artificial intelligence and statistics , pages 932–939, 2010.

Reference

[8] Bi, Wei, et al. "Learning to predict from crowdsourced data." Proceedings of the 30th International Conference on Uncertainty in Artificial Intelligence (UAI). 2014.
[9] Hiroshi Kajino, Yuta Tsuboi, and Hisashi Kashima. A convex formulation for learning from crowds, 2012.
[10] Hamed Valizadegan, Quang Nguyen, and Milos Hauskrecht. Learning classification models from multiple experts. Journal of biomedical informatics , 46(6):1125–1135, 2013.
[11] Hiroshi Kajino, Yuta Tsuboi, and Hisashi Kashima. Clustering crowds. In Twenty-Seventh AAAI Conference on Artificial Intelligence , 2013.
[12] Tian Tian and Jun Zhu. Max-margin majority voting for learning from crowds. In Advances in Neural Information Processing Systems, pages 1612–1620, 2015
[13] Zhong, Jinhong, Ke Tang, and Zhi-Hua Zhou. "Active learning from crowds with unsure option." Proceedings of the 24th International Conference on Artificial Intelligence. AAAI Press, 2015.
[14] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In NIPS , pages 2035–2043, 2009.
[15] L. Zhao, Y. Zhang, and G. Sukthankar. An active learning approach for jointly estimating worker performance and annotation reliability with crowdsourced data. arXiv preprint , 2014.
[16] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. Applied Statistics , pages 20–28, 1979.

Reference

[17] D. Zhou, S. Basu, Y. Mao, and J. C. Platt. Learning from the wisdom of crowds by minimax entropy. In NIPS , 2012.
[18] D. Zhou, Q. Liu, J. Platt, and C. Meek. Aggregating ordinal labels from crowds by minimax conditional entropy. In ICML , 2014.

Thank You