Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

模型预测可靠性的直接保障
—深度学习模型校准技术

硕士研究生 吴肖龙

2024年6月23日

问题回溯

- 总结反思
  - 演讲时整体语速过快，缺乏停顿和关键性总结
  - 内容过多和较为抽象，缺乏生动形象化的示例
  - 未注意自己演讲时的仪态和肢体动作

- 相关内容
  - 2024.01.07 段学明《DNN中的理论可解释性》
  - 2023.11.19 夏志豪《智能模型解释技术及其攻击方法》
  - 2023.11.05 吴肖龙《智能模型的不确定性估计》
  - 2022.08.23 王若辉《AI测试：历史与发展》

内容提要

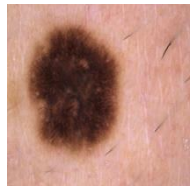
- 预期收获
- 题目内涵解析
- 研究背景与意义
- 研究历史与现状
- 知识基础
- 算法原理
  - CALS-ALM
  - GC-ETS
- 特点总结与工作展望
- 参考文献

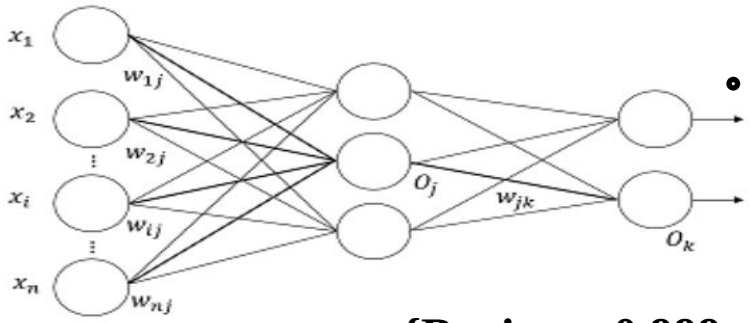
背景简介

- 预期收获
  - 掌握模型校准的基本概念和目标
  - 理解模型校准与不确定性量化的异同
  - 理解模型校准性能不佳的主要原因
  - 了解现有的模型校准框架和方法

研究背景　实例引入

- 使用模型预测数据





测试准确率90%

Benign

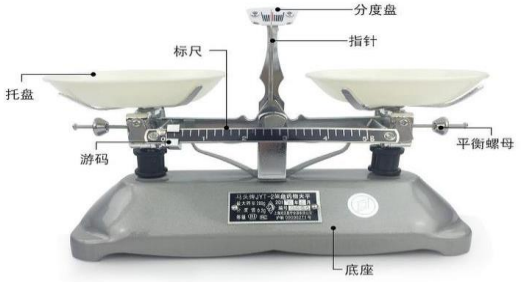
- 提供置信度的预测
- 统计测试
- 提供更可靠的评估
  - 不确定性估计
  - 调整预测输出（校准）

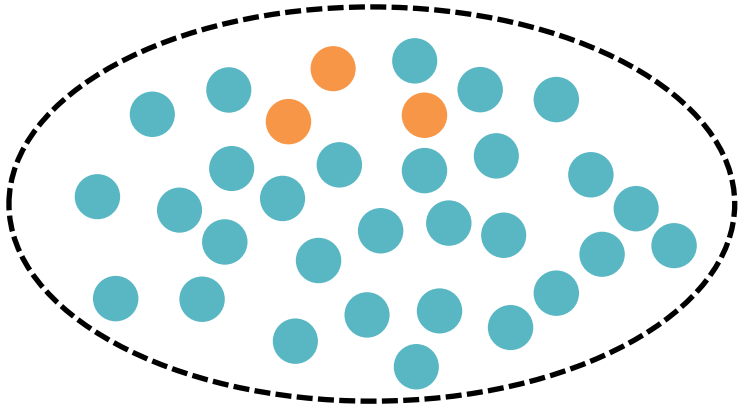
{Benign：0.999；malignant：0.001}

准确率 = 97%

概率学派
- 内部度量
- 贝叶斯方法

频率学派
- 深度集成
- 测试时间增强

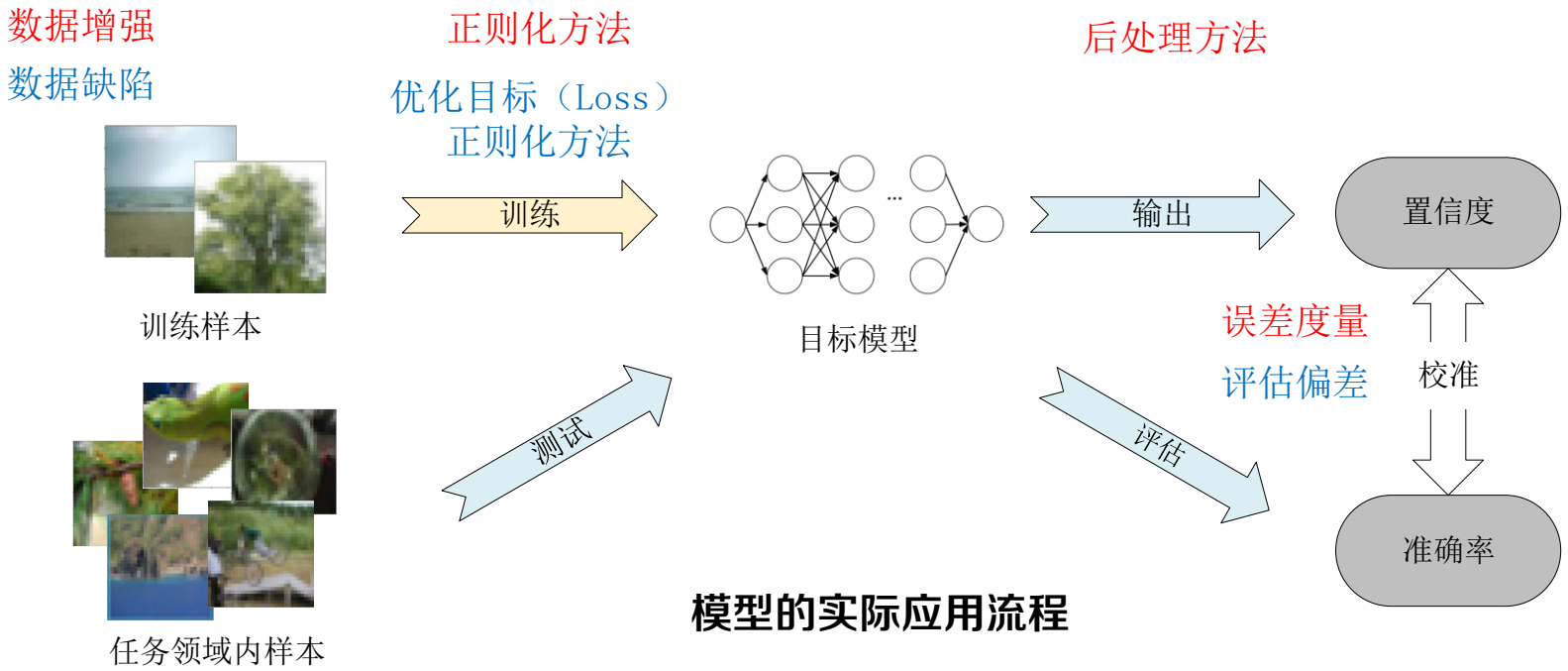




- 模型预测错误的样本
- 模型预测正确的样本

内涵解析与研究目标　模型校准技术

- 模型校准
  - 预测置信度与事件发生的真实概率的匹配程度
- 校准不佳的原因
  - 过度参数化和过拟合
  - 数据不平衡
  - 评估指标的偏差
- 研究目标
  - 提供校准良好的模型



模型的实际应用流程

高预测性模型的校准可能很差，表现为高校准误差;
深度模型往往主要是过度自信，表现为尖峰后验分布

研究背景

- 研究背景
  - 高安全性需求领域的关键决策需求
    - 自动驾驶、医疗、法律、国防和关基设施
  - 对单个预测的性能关注而非模型性能
  - 当前模型普遍表现出校准问题
    - 过度自信（主要）、信息不足

2021年，NHTSA报告807起自动驾驶相关案件

- 研究意义
  - 防止模型过度自信或信息不足
  - 帮助人们更好的放弃或接受预测
  - 暴露和发掘智能模型的潜在缺陷
  - 促进可信（可靠）人工智能的构建

2000~2013年间，机器人手术致死患者达144人

研究历史

2017 Guo等人提出温度缩放的后处理校准方法，通过添加温度系数调整预测输出，这是最流行且实施最容易的校准方法之一

2018 Kumar等人首次从校准误差度量角度改进校准性能，提出MMCE指标并将其引入训练过程

2019 Kull等人面向多分类问题，提出了用狄利克雷分布拟合和校准数据集的预测后验

2019 Thulasidasan等人提出Mixup方法，通过在两个样本进行线性插值进行数据增强，降低模型校准误差

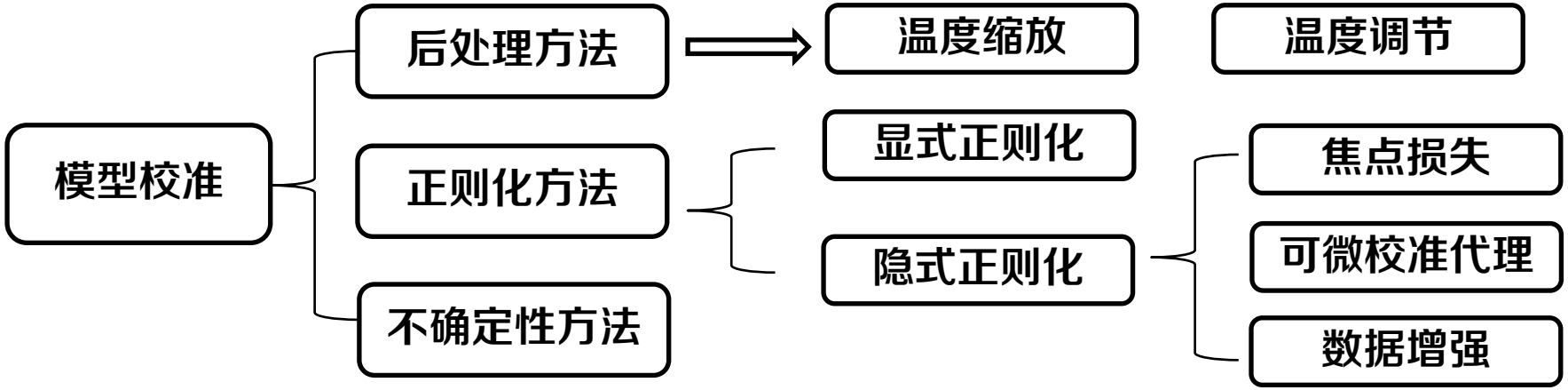
2019 Muller等人提出了通过标签平滑技术，降低one-hot编码在训练时过拟合的影响，缓解模型过度自信问题

2020 Zhang等人结合集成方法和温度缩放，集成多组温度缩放的调整结果，提升校准性能

2021 Niethammer等人改进了温度缩放方法，在图片的不同位置使用局部温度缩放，使得校准更加准确

2022 Hebbalaguppe等人提出了MDCA方法，考虑各个类别下的静态校准，并将其引入正则化损失函数进行训练优化

2023 Bosselut等人提出大模型上的校准方法，结合随机文本生成，在普通标签、上下文标签、领域标签等进行校准

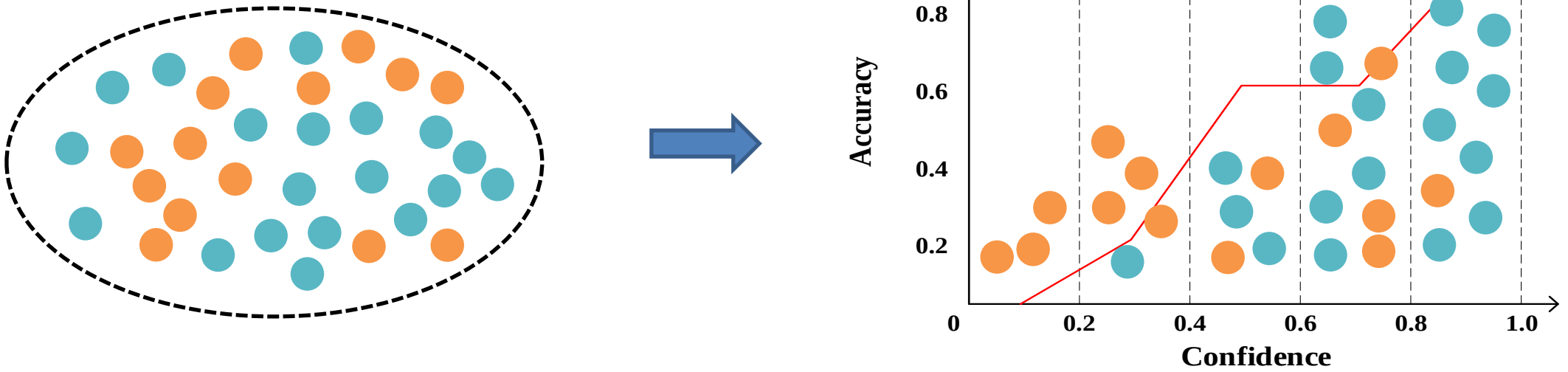


知识基础　预期校准误差ECE

- 衡量置信度和测试准确率的相对差异

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{N} |A_m - C_m|$$

- M为分组数，$|B_m|$为组中的样本数，N表示样本总数



算法原理 CALS-ALM

【 IEEE CVPR 】

Class Adaptive Network Calibration

CALS-ALM TIPO

T	目标	实现具有类别自适应的校准方法
I	输入	目标模型*1、训练数据集*1、验证数据集*1
P	处理	1. 将带边界约束的标签平衡引入损失项进行训练 2. 使用拉格朗日乘数法在验证集上更新各类别的惩罚系数 3. 循环前两步至模型收敛，使模型在各类别实现校准
O	输出	良好校准的模型*1

P	问题	1. 现有校准方法在类别上提供了相等的权重 2. 校准结果固定限制了模型进一步优化性能
C	条件	掌握完整模型设计与训练信息
D	难点	1. 惩罚函数和约束函数的设定 2. 校准系数的动态调整
L	水平	CVPR 2023（CCF A）

CALS-ALM 基于边界的标签平滑

- 标签平滑
  - 在one-hot编码的目标分布中加入噪声

$$y_i = y_{one-hot}(1-\alpha) + \frac{\alpha}{K}$$

  - K表示类别个数， α是较小的超参数（一般为0.1）

- 基于边际的约束
  - 控制输出logits范围

$$\max_k \left\{ l_k^{(i)} \right\} - l_j^{(i)} \leq m \cdot 1_{j \neq y(i)}$$

  - K表示第K个类别， $l^{(i)}$表示第i个样本的logits输出， j是真实类别， m是边界参数，$1_{j \neq y(i)}$是一个指示函数，当$j \neq y(i)$时值为1，否则为0
  - 将约束条件转化为损失项

$$Loss = \min_{\theta} \sum_{i=1}^{N} L_{CE}\left(x^{(i)}, y^{(i)}\right) + \lambda \sum_{i=1}^{N} \sum_{j=1}^{K} \max\left\{0, \max_k \left\{ l_k^{(i)} \right\} - l_j^{(i)} - m\right\}$$

- 样本级约束
  - 问题：惩罚系数λ对所有目标都是相同的
  - 为每个样本和类别引入了不同的惩罚权重$\lambda_{i,j}$
    - 变成需要选择 $N \times K$个惩罚权重的问题， N为样本数量， K为类别数量

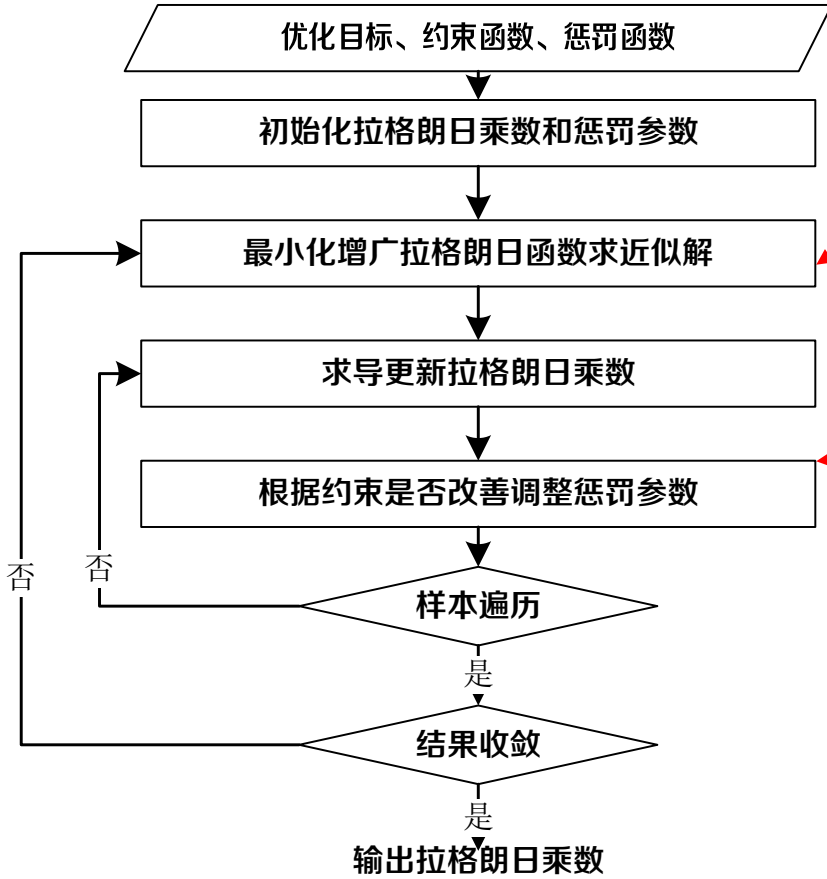
    以Cityscapes数据集为例，其包含19个类别的3000张像素为2048x1024的图像，以float32存储惩罚权重需445GB（开销过大）
- 惩罚粒度的放松
  - 由样本级约束转化为类别约束（K个权重的优化问题）
    - 增广拉格朗日乘数（ $\mathcal{L}^{(j)}(x)$ ）方法（附录A）

$$\min_x \mathcal{L}^{(j)}(x) = f(x) + \sum_{i=1}^{n} P(h_i(x), p_i^{(j)}, \lambda_i^{(j)})$$

    - P是惩罚-拉格朗日函数， $p^{(j)}$是惩罚参数，$\lambda_i^{(j)}$是与第i个约束相关联的拉格朗日乘数

CALS-ALM 增广拉格朗日乘数一般框架

- 类粒度的双层优化
  - 内循环更新惩罚参数
  - 外循环求解乘数算子



Algorithm 1 Augmented Lagrangian Multiplier algorithm

Require: Objective function f
Require: Constraint functions $h_i, i = 1, \ldots, n$
Require: Penalty function P, initial $\boldsymbol{\lambda}^{(0)} \in \mathbb{R}^n_{++}$, $\boldsymbol{\rho}^{(0)} \in \mathbb{R}^n_{++}$
Require: Initial variable $x^{(0)}$, iterations $j = 1$

while not converged **do**
  Initialize with $x^{(j-1)}$ and minimize (approximately):
  $\mathcal{L}^{(j)}(x) = f(x) + \sum_{i=1}^{n} P(h_i(x), \rho_i^{(j)}, \lambda_i^{(j)})$
  $x^{(j)} \leftarrow$ (approximate) minimizer of $\mathcal{L}^{(j)}$
  for $i = 1, \ldots, n$ **do**
    $\lambda_i^{(j+1)} \leftarrow P'(h_i(x^{(j)}), \rho_i^{(j)}, \lambda_i^{(j)})$
    if the i-th constraint does not improve **then**
      $\rho_i^{(j+1)} \leftarrow \gamma \rho_i^{(j)}$
    else
      $\rho_i^{(j+1)} \leftarrow \rho_i^{(j)}$
    end if
  end for
  $j \leftarrow j + 1$
end while

CALS-ALM 训练过程的实际优化

- 优化对象

$$\min_{\theta} \sum_{i=1}^{N} L_{CE}\left(x^{(i)}, y^{(i)}\right) + \sum_{k=1}^{K} P\left(d_k^{(i)} - m, p_k, \lambda_k\right)$$

- 归一化（提高稳定性）

$$\sum_{i=1}^{N} L_{CE}\left(x^{(i)}, y^{(i)}\right) + \frac{1}{K}\sum_{k=1}^{K} P\left(\frac{d_k^{(i)}}{m} - 1, p_k, \lambda_k\right)$$

- 更新拉格朗日乘数

$$\lambda_k^{(j+1)} = \frac{1}{|D_{val}|} \sum_{(x,y)\epsilon D_{val}} P'\left(\frac{d_k^{(i)}}{m} - 1, p_k, \lambda_k\right)$$

- 使用的惩罚函数（经验）

$$PHR(z,p,\lambda) = \begin{cases} \lambda z + \frac{1}{2} p z^2, \lambda + pz \geq 0 \\ -\frac{\lambda^2}{2p}, \quad else \end{cases}$$

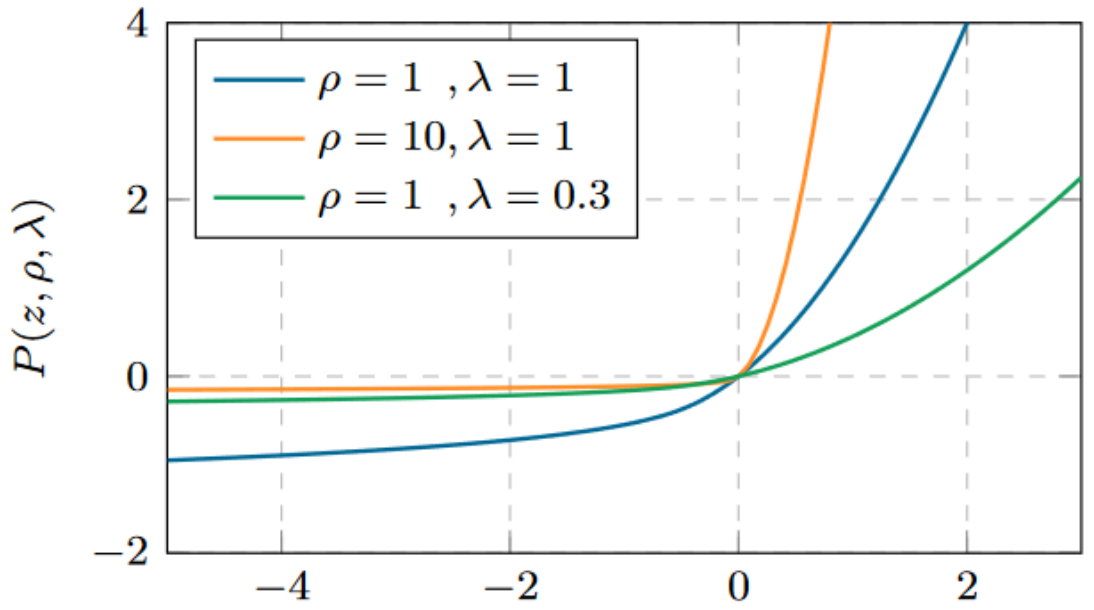
Algorithm 2 CALS-ALM training

Require: DNN initial θ^0, margin m
Require: Dataset: $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, batch size B
Require: Penalty function P, $\gamma > 1$, $\tau \in (0, 1)$
Require: Initial $\boldsymbol{\lambda}^{(0)} \in \mathbb{R}^n_{++}$, $\boldsymbol{\rho}^{(0)} \in \mathbb{R}^n_{++}$

for $j = 0, \ldots, T$ **do** // Epochs of training
  for each mini-batch $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{B}$ in $\mathcal{D}_{\text{train}}$ **do**
    $\mathcal{L}_{\text{c}} = \sum_{i=1}^{B} \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)})$ // Cross-entropy
    $\mathcal{L}_{\text{p}} = \sum_{i=1}^{B} \frac{1}{K} \sum_{k=1}^{K} P\left(\frac{d_k^{(i)}}{m} - 1, \rho_k^{(j)}, \lambda_k^{(j)}\right)$ // Penalties
    $\mathcal{L} = \frac{1}{B}(\mathcal{L}_{\text{c}} + \mathcal{L}_{\text{p}})$
    $\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha \cdot \nabla_\theta \mathcal{L}$ // Gradient descent
  end for 训练集做梯度下降
  for $k = 1, \ldots, K$ **do** // Adjust $\boldsymbol{\lambda}$ and $\boldsymbol{\rho}$ on validation
    $\lambda_k^{(j+1)} = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\text{val}}} P'\left(\frac{d_k}{m} - 1, \rho_k^{(j)}, \lambda_k^{(j)}\right)$
    $\overline{d_k}^{(j)} = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\text{val}}} \frac{d_k}{m} - 1$ // Average constraint
    if $j \geq 1$ **and** $\overline{d_k}^{(j)} > \tau \max\{0, \overline{d_k}^{(j-1)}\}$ **then**
      $\rho_k^{(j+1)} \leftarrow \gamma \rho_k^{(j)}$ // Constraint has not improved
    else
      $\rho_k^{(j+1)} \leftarrow \rho_k^{(j)}$
    end if
  end for 验证集更新惩罚参数
end for

CALS-ALM 创新分析

- 核心思想
  - 在更细粒度的条件下进行模型校准
  - 将预测输出约束至有限的范围内
  - 在训练过程中实时调整惩罚系数



拉格朗日函数关于惩罚参数和拉格朗日乘数的变化曲线

- 创新点
  - 将优化问题转化为损失函数的惩罚项，并推广至类级别进行优化
  - 应用增广拉格朗日乘数方法交替进行目标函数最小化和拉格朗日乘数更新来求解

实验设计　数据集、评价指标、对比方法

- 数据集

数据集	说明
Tiny-ImageNet	ImageNet 的小型子集，包含 200 个类别的 100,000 张图像
ImageNet	1,000 个类别的超 1M 张图像大型测试基准
ImageNet-LT	ImageNet 数据集的扩展，专注于类别的长尾分布，类别极不平衡

- 评价指标
  - 预期校准误差（ECE）
  - 自适应校准误差（AECE）：ECE的变体，样本均匀分配到不同箱体
  - 分类校准误差（CECE）：ECE的变体，按类别进行校准
- 对比方法
  - CE（基线）、MMCE（2018 ICML）、ECP（2017 ICLR）、LS（2016 CVPR）
  - FL/FLSD（2020 NIPS）、CPC（2022 CVPR）、MbLS（2022 CVPR）

实验结果　对比实验

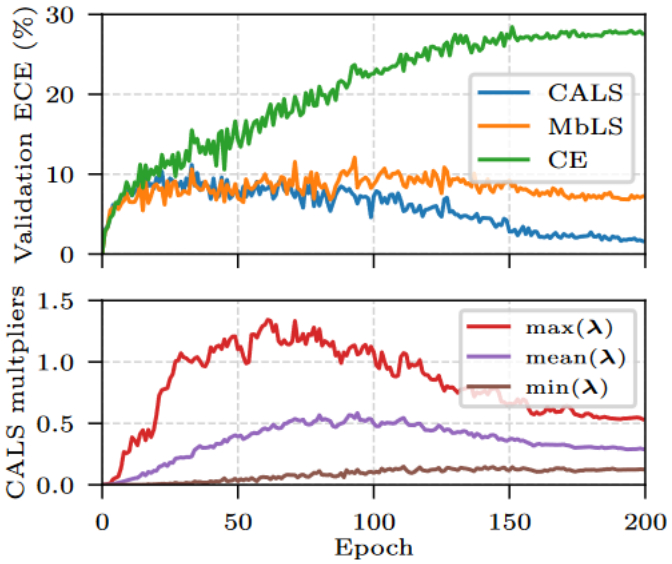
- 实验分析
  - 校准错误的严重程度与数据集和模型架构相关
  - 方法在全部场景下取得了较小的校准误差，对长尾分布数据集的提升效果显著

	TinyImageNet			ImageNet						ImageNet-LT					
	ResNet-50			ResNet-50			SwinV2-T			ResNet-50			SwinV2-T		
Method	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE
CE	65.02	3.73	3.69	75.16	9.19	9.18	75.60	9.95	9.94	37.90	28.12	28.12	31.82	31.82	36.68
MMCE [18]	65.34	2.81	2.61	74.85	8.57	8.56	76.68	9.07	9.08	37.79	28.41	28.40	33.14	26.41	26.41
ECP [38]	64.90	4.00	3.92	75.22	8.27	8.26	75.82	9.88	9.86	37.69	28.14	28.13	31.22	33.70	33.70
LS [42]	**65.78**	3.17	3.16	76.04	2.57	2.88	75.42	7.32	7.33	37.88	10.46	10.38	31.70	11.42	11.40
FL [31]	63.09	2.96	3.12	73.87	1.60	1.65	75.60	3.19	3.18	36.04	18.37	18.36	30.73	25.50	25.50
FLSD [31]	64.09	2.91	2.95	73.97	2.08	2.06	74.70	2.44	2.37	36.18	17.77	17.78	32.56	25.16	25.17
CPC [5]	64.49	4.88	4.91	76.33	3.66	3.59	76.34	5.50	5.33	**38.90**	16.00	15.99	32.54	13.21	13.19
MbLS [22]	64.74	1.64	1.73	75.82	4.44	4.26	77.18	1.95	1.73	38.32	6.16	6.16	32.05	7.65	7.64
CALS-HR	65.09	2.50	2.42	76.34	5.63	5.69	**77.58**	3.06	2.95	38.50	2.83	2.78	**34.31**	2.37	**2.45**
CALS-ALM	65.03	**1.54**	**1.38**	**76.44**	**1.46**	**1.32**	77.10	**1.61**	**1.69**	38.56	**2.15**	**2.30**	33.94	**2.32**	**2.45**

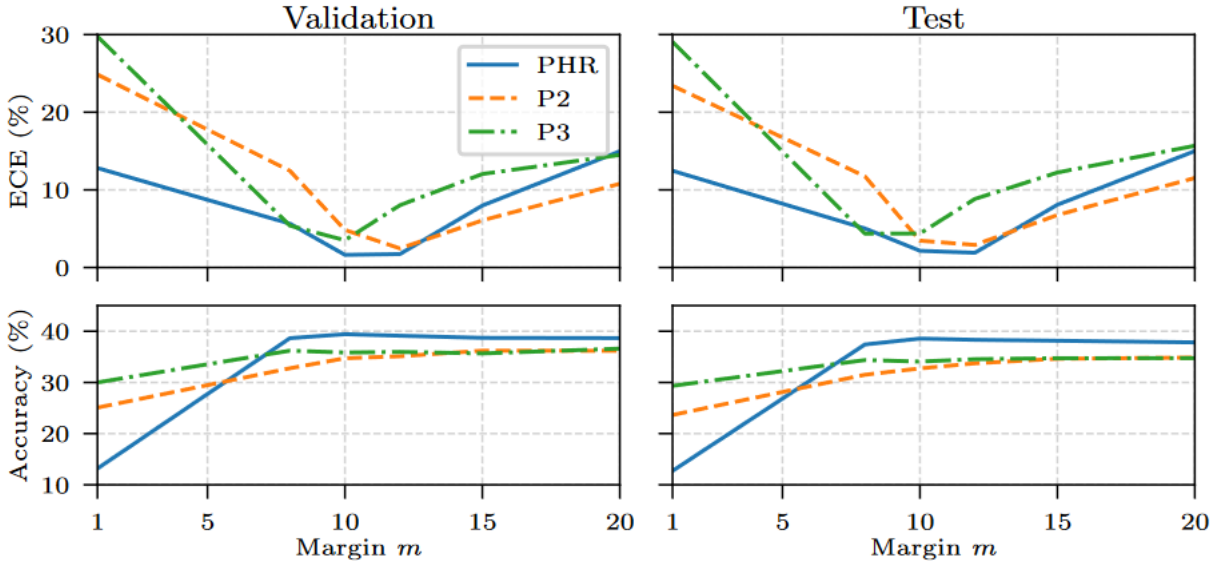
实现了在不同类别间的自适应

实验结果　消融实验

- 实验分析
  - 拉格朗日乘数方法和PHR惩罚函数都实现了最佳的效果
  - 随着训练周期增加，校准误差和惩罚系数都先快速增加再逐渐收敛，呈正相关



(a) Validation ECE and multipliers during training.



(b) Effect of penalty function and margin on accuracy and ECE for CALS.

实现了在模型训练过程中的动态调整

【 MIT Press 】

Beyond Probability Partitions: Calibrating Neural Networks with Semantic Aware Grouping

GC-ETS TIPO

T	目标	实现更广义的校准误差评估和更准确的校准方法
I	输入	目标模型*1、训练数据集*1、验证数据集*1、测试数据集*1
P	处理	1. 使用L-BFGS优化算法（附录B）对样本在语义信息上进行分组 2. 使用温度缩放在分组上对模型进行校准 3. 将分组函数和校准函数同时进行端到端优化
O	输出	样本分组*N、校准后的模型*1

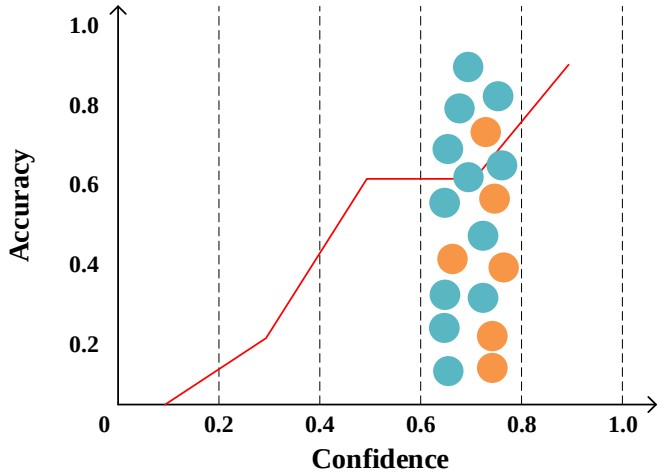
P	问题	1. 固定化的分箱方式在评估时可能存在校准偏差 2. 从预测结果出发进行分箱缺失样本在输入空间的语义信息
C	条件	训练数据和验证数据间具有良好的分布适应性
D	难点	1. 如何选择合适的样本分组方式 2. 如何兼顾样本分组和模型校准
L	水平	NeurIPS 2024（CCF A）

GC-ETS 分组函数与输入空间划分

- 分组函数基本原则
  - 每个输入都能对应一个分组
  - 分组彼此是互斥的
  - 所有可能的分组都被覆盖
- 常见的分组方式
  - ECE（等间距划分置信度）、CECE（在各个类别上等间距划分置信度）、TCE（按预测最高概率标签分组）、AECE（等个数对样本进行分组）
- 抽象表示

$$PCE(S, g, \mathcal{L}, f, \mathcal{D}) = \sum_{P \in \mathcal{P}} p(P) \sum_{G \in P} p(G) \mathcal{L}(S(G), S(f(G)))$$

  - $\mathcal{P}$表示所有可能的分组方式集合,$p(P)$是选择特定分组方式的概率,$p(G)$是观察到分组G中样本的概率,$\mathcal{L}(\cdot)$衡量绝对误差,$S(G)$是分组上的统计量,$f(\cdot)$是模型预测函数



GC-ETS 分组函数和分类函数

- 假定分组函数为$g(x)$,共分为K组，PCE表示为

$$PCE = \sum_{i=1}^{K} \frac{|G_i|}{|D|} \mathcal{L}(S(G_i), S(f_{G_i}(G_i)))$$

  - 其中$|D|$为样本数量，$|G_i|$为组中的样本数量，$\mathcal{L}(\cdot)$表示相对差异，$S(\cdot)$表示统计量

- 使用温度缩放作为校准函数

$$\mathcal{L}_{group} = \frac{1}{|G_i|} \sum_{x,y \in G_i} log \frac{e^{\frac{o(x)y}{\tau_i}}}{\sum_j e^{\frac{o(x)j}{\tau_i}}}$$

- 将分组函数合并到公式中

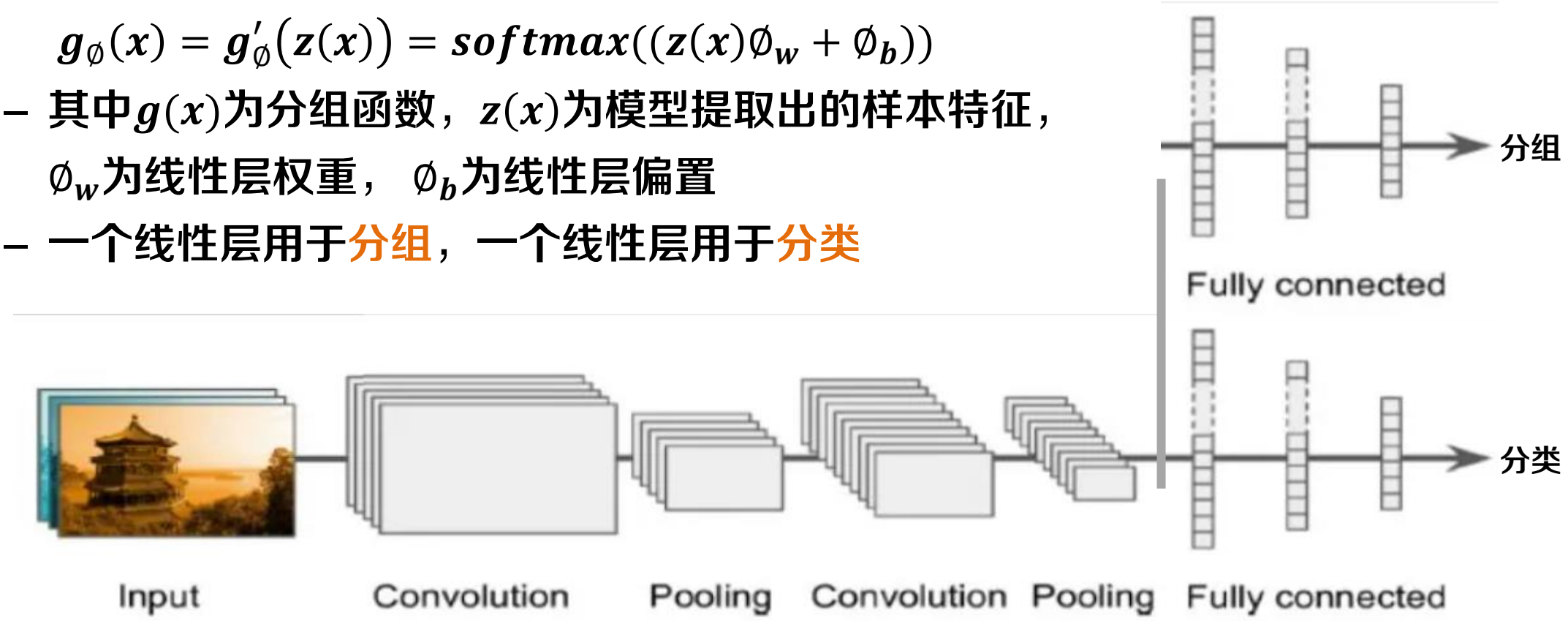
$$\mathcal{L}_{GC+TS} = \frac{1}{|D|} \sum_{x,y \in D} log \sum_i g(x)_i \frac{e^{\frac{o(x)y}{\tau_i}}}{\sum_j e^{\frac{o(x)j}{\tau_i}}}$$

同时考虑分组和校准

- 分组以one-hot编码方式存在，引入平滑技术输出概率值
- 引入新的线性层，通过训练权重和偏置计算分组函数

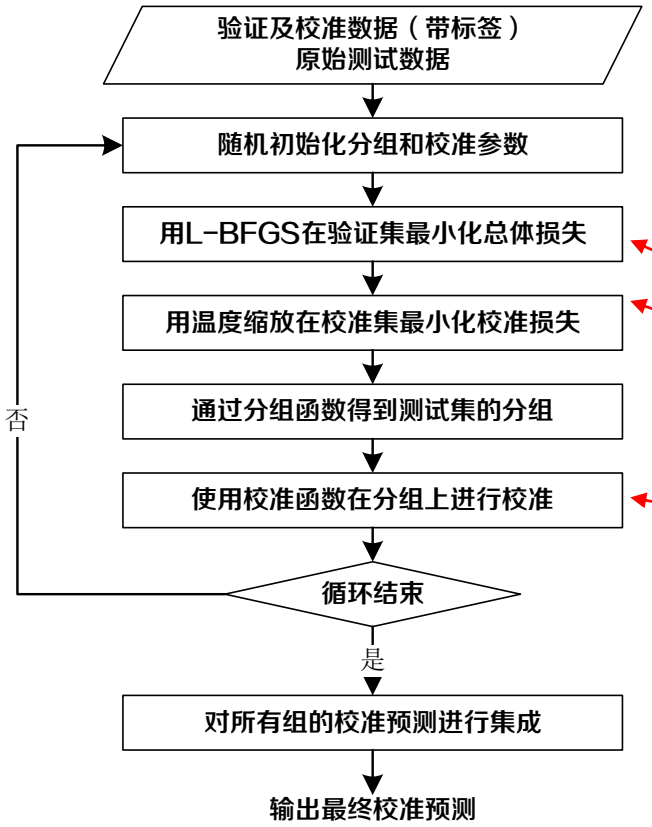
$$g_{\emptyset}(x) = g'_{\emptyset}\big(z(x)\big) = softmax((z(x)\emptyset_w + \emptyset_b))$$

  - 其中$g(x)$为分组函数，$z(x)$为模型提取出的样本特征，$\emptyset_w$为线性层权重，$\emptyset_b$为线性层偏置
  - 一个线性层用于分组，一个线性层用于分类



GC-ETS 概率分区校准算法框架

- 类粒度的双层优化
  - 内循环在各自分组条件下校准模型
  - 外循环尝试找到更好的分组方式



Algorithm 1 Train group calibration with temperature scaling (GC+TS)

Input: $D_{val} = \{Z_{val}, O_{val}, Y_{val}\}, D_{ho} = \{Z_{ho}, O_{ho}, Y_{ho}\}, D_{test} = \{Z_{test}, O_{test}\}, U, K, \lambda$

Output: Calibrated $\hat{Y}_{test}$

for $u \leftarrow 0$ to $U-1$ **do**
  Randomly initialize ϕ_u and θ_u
  Optimize ϕ_u and θ_u on D_{val} to minimize Eq. (13) with L-BFGS[30]
  Optimize θ_{ui} on D_{ho} to minimize Eq. (11) with base calibration method(e.g., TS)
  Calculate partition $P_u = \{G_{ui}\}$ with grouping function g_{ϕ_u} on D_{test}
  for $i \leftarrow 0$ to $K-1$ **do**
    Calibrate G_{ui} with $f_{\theta_{ui}}(\cdot)$ to obtain $\hat{Y}_{ui}$
  end for
  Merge predicts in different groups $\hat{Y}_u = \{\hat{Y}_{ui}\}$
end for
Ensemble predicts in different partitions $\hat{Y}_{test} = \frac{1}{U}\sum_u \hat{Y}_u$

GC-ETS 创新分析

- 核心思想
  - 将校准的评估方法统一至概率分组框架
  - 原始语义信息应作为分组依据而非预测概率
  - 同时优化分组函数和校准函数的端到端优化

- 创新点
  - 基于语义特征训练分组函数得到更好的校准评估方法
  - 使用集成方法组合多组校准结果提升校准性能

实验设计　数据集、评价指标、对比方法

- 数据集
  - CIFAR10、CIFAR100、ImageNet
- 模型架构
  - ResNet（2016）、VGG（2015）、DenseNet（2017）、SWIN（2021）、ShuffleNet（2018）
- 评价指标
  - 预期校准误差（ECE）
- 对比方法
  - 精度保持方法：TS（2017 ICML）、ETS（2020 ICML）、 IRM(AP)（ 2020 ICML ）
  - 非精度保持方法： Hist.B （ 2017 ICML ）、 Beta（2017 AISTATS ）、BBQ（2015 AAAI）、DirODIR（2019 NeurIPS）、 GPC（2020 AISTATS ）、IRM(NAP)

实验结果 对比实验

- 实验分析
  - 方法整体上取得了最好结果
  - 集成起到了重要作用
  - 在数据集和模型架构上差异性较大

Dataset	Model	Uncal	TS	ETS	IRM(AP)	GC+TS(ours)	GC+ETS(ours)
CIFAR10	Resnet152	0.0249	0.0086	0.0101	0.0115	**0.0079**	0.0089
CIFAR10	Shufflenet	0.0464	0.0107	0.0103	0.0146	0.0099	**0.0093**
CIFAR10	VGG11	0.0473	0.0125	0.0135	0.0157	**0.0120**	**0.0122**
CIFAR100	Densenet121	0.0558	0.0418	0.0289	0.0415	0.0411	**0.0280**
CIFAR100	Resnet50	0.0580	0.0435	0.0292	0.0424	0.0427	**0.0269**
CIFAR100	VGG19	0.1668	0.0485	0.0472	0.0476	0.0414	**0.0360**
Imagenet	Resnet18	0.0279	0.0178	0.0104	0.0188	0.0173	**0.0100**
Imagenet	Resnet50	0.0382	0.0182	**0.0102**	0.0218	0.0174	**0.0103**
Imagenet	Swin	0.0266	0.0367	0.0218	**0.0140**	0.0366	0.0193
Average improvements						**-5.03%**	**-8.92%**

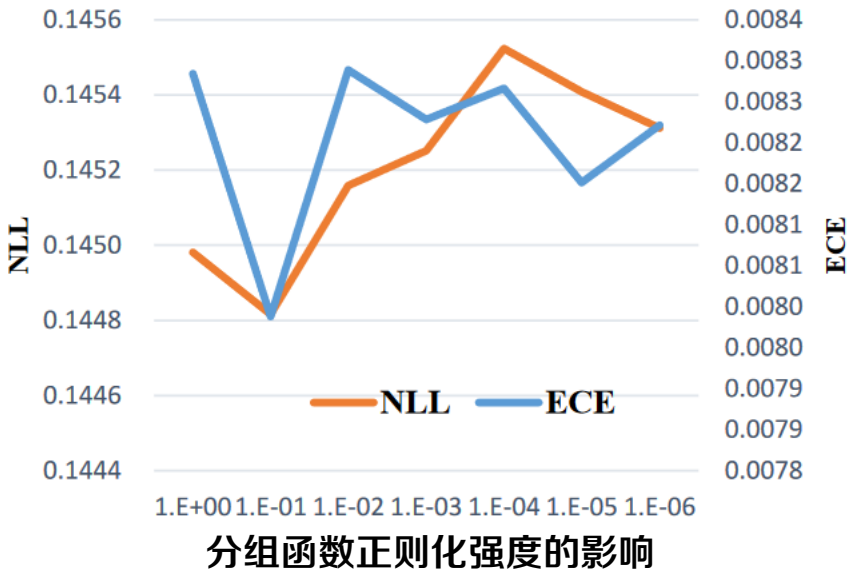
精度保持方法实验结果

Dataset	Model	Hist.B	Beta	BBQ	DirODIR	GPC	IRM(NAP)	Acc. Dec.	GC+ETS (ours)
CIFAR10	Resnet152	0.0172	0.0095	0.0097	0.0101	0.0189	***0.0087***	-0.079%	**0.0089**
CIFAR10	Shufflenet	0.0322	0.0137	0.0139	0.0158	0.0341	*0.0119*	-0.050%	**0.0093**
CIFAR10	VGG11	0.0279	0.0150	0.0137	0.0156	0.0376	***0.0119***	-0.129%	**0.0122**
CIFAR100	Densenet121	0.0632	0.0520	0.0307	0.0533	0.0306	***0.0203***	-0.149%	0.0280
CIFAR100	Resnet50	0.0618	0.0550	*0.0334*	0.0553	0.0346	0.0422	-3.686%	**0.0269**
CIFAR100	VGG19	0.0453	0.0642	*0.0446*	0.0932	0.1406	0.0470	-5.046%	**0.0360**
Imagenet	Resnet18	0.0714	0.0690	0.0483	0.0386	-*	*0.0119*	-0.027%	**0.0100**
Imagenet	Resnet50	0.0502	0.0707	0.0482	0.0326	-	***0.0107***	-0.006%	**0.0103**
Imagenet	Swin	0.0335	0.0629	0.0478	0.0148	-	***0.0110***	-0.060%	0.0193

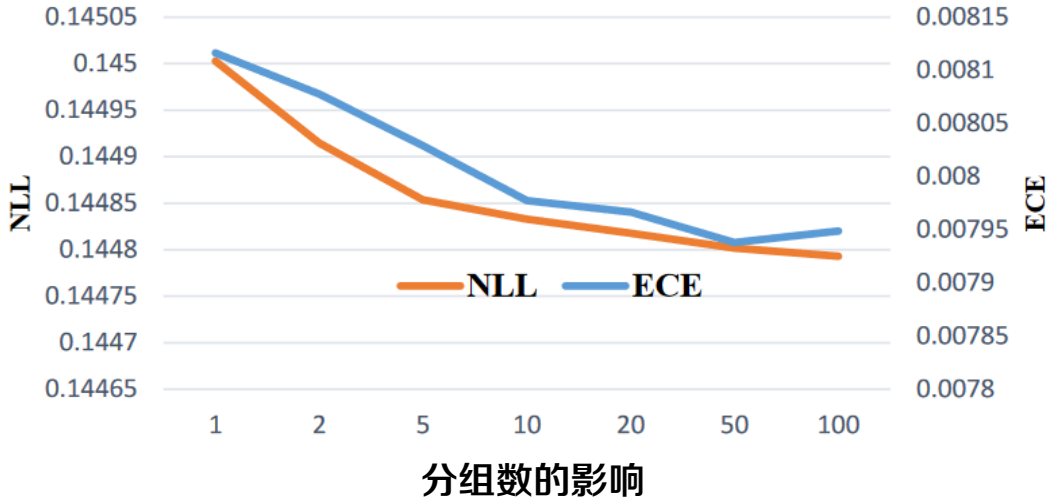
非精度保持方法实验结果

实验结果　消融实验（参数实验）

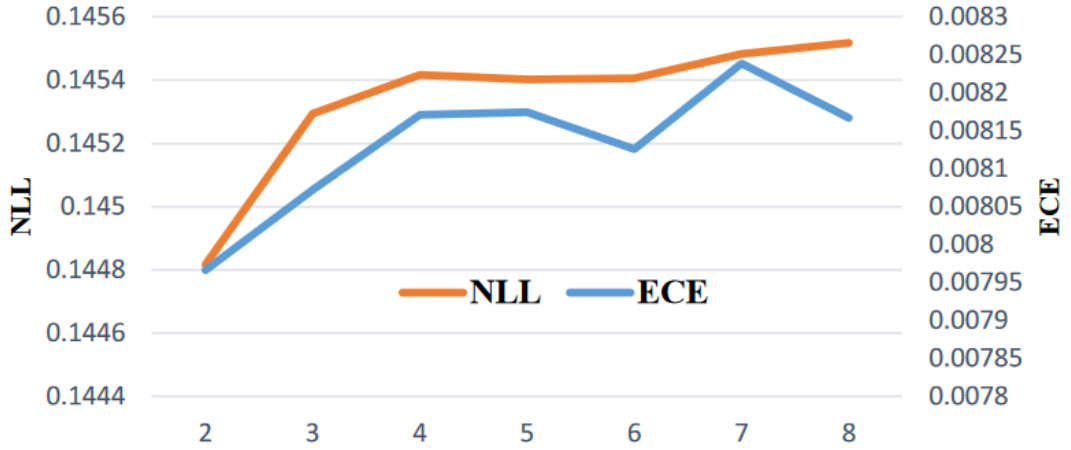
- 实验分析
  - 分组数量有助于提升校准性能
  - 组内类别复杂度影响校准结果



分组函数正则化强度的影响



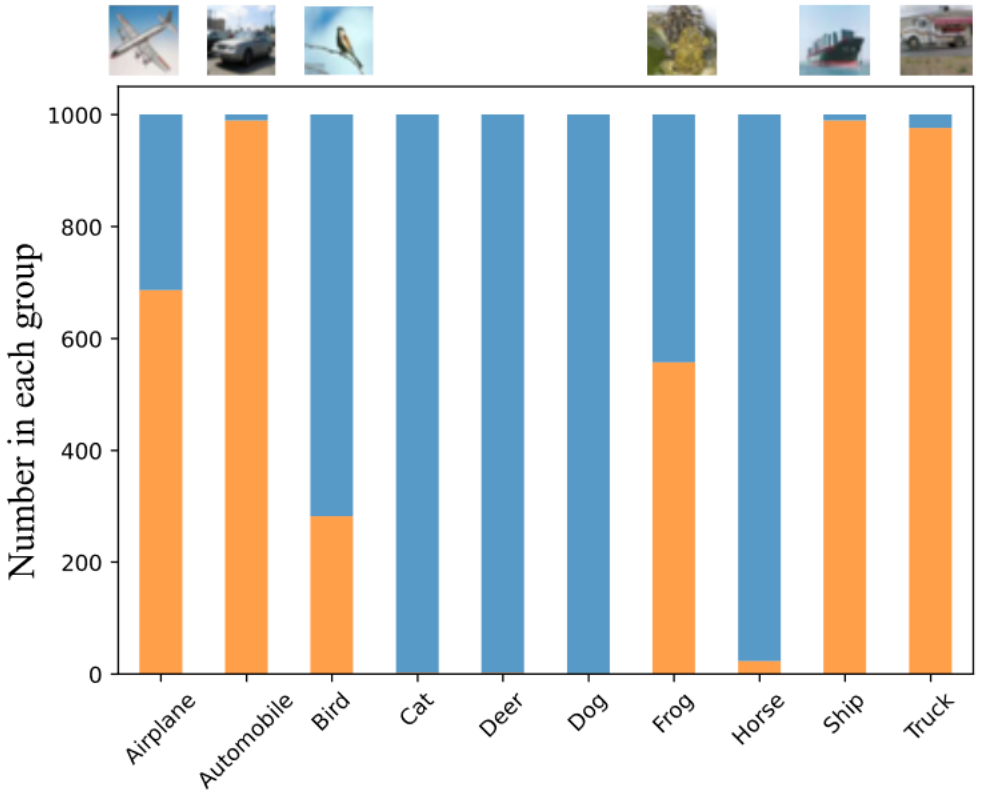
分组数的影响

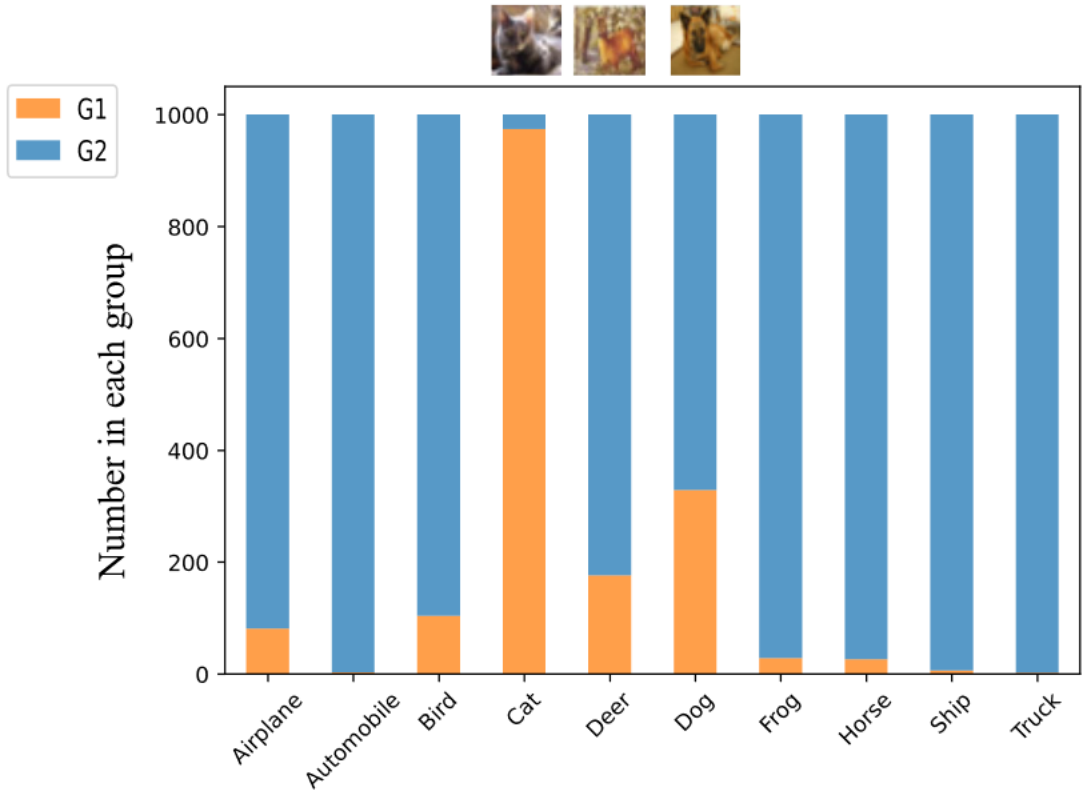


组内类别数的影响

实验结果　学习分区可视化

- 实验分析
  - 在同一分组中展现相似的视觉特征（不局限于同一类别）





特点总结与未来展望

特点总结与未来展望

- **CALS-ALM**
  - 优点：有效应对了类别长尾分布下的校准，自适应调整校准系数同步模型优化
  - 缺点：对同分布验证集的强依赖，复杂的超参数和约束项设置
- **GC-ETS**
  - 优点：考虑了校准误差评估的灵活性，具有良好的校准性能
  - 缺点：计算开销和成本较高，缺少对分组数量和子集选择的控制
- 前沿发展与关键挑战
  - 无偏校准估计器（克服分箱带来的影响）
  - 校准生成模型和大模型（如何理解一致性）
  - 计算效率和可解释性（集成等大量重复开销）
  - 实际应用和伦理问题（医疗、法律）

参考文献

[1] Wang C. Calibration in deep learning: A survey of the state-of-the-art[EB/OL]. (2024.05.10) [2024.06.20], https://arxiv.org/abs/2308.01222.

[2] Zhu F, Zhang X Y, Cheng Z, et al. Revisiting Confidence Estimation: Towards Reliable Failure Prediction[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.

[3] Liu B, Rony J, Galdran A, et al. Class adaptive network calibration. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition[C]. Vancouver, BC, Canada: IEEE, 2023: 16070-16079.

[4] Yang J Q, Zhan D C, Gan L. Beyond probability partitions: Calibrating neural networks with semantic aware grouping[J]. Advances in Neural Information Processing Systems, 2024, 36.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！

附录A 拉格朗日乘子法

假设有一个方程为$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}-1=0$的椭球，我们要求这个椭球内接长方体的最大体积。

那么也就是说我们要在$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}-1=0$的条件下，求$f(x,y,z)=8xyz$的最大值。

现在我们定义一个拉格朗日函数$F(x,\lambda)$:

$$F(x,\lambda)=f(x)+\sum_{k=1}^{l}\lambda_k h_k(x)$$

其中λ_k是各个约束条件的待定系数。

接下来求偏导为0的解:

$$\frac{\partial F}{\partial x}=0,\frac{\partial F}{\partial \lambda}=0$$

回到我们的题目，就可以将问题转化为:

$$F(x,y,z,\lambda)=8xyz+\lambda(\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}-1)$$

附录A 拉格朗日乘子法

接下来对$F(x,y,z,\lambda)$求偏导可得：

$$\frac{\partial F(x,y,z,\lambda)}{\partial x} = 8yz + \frac{2\lambda x}{a^2} = 0$$

$$\frac{\partial F(x,y,z,\lambda)}{\partial y} = 8xz + \frac{2\lambda y}{b^2} = 0$$

$$\frac{\partial F(x,y,z,\lambda)}{\partial z} = 8xy + \frac{2\lambda z}{c^2} = 0$$

$$\frac{\partial F(x,y,z,\lambda)}{\partial \lambda} = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0$$

联立前三个方程并代入第四个方程后可以求出：

$$x = \frac{\sqrt{3}}{3}a, y = \frac{\sqrt{3}}{3}b, z = \frac{\sqrt{3}}{3}c$$

最后可得

$$f(x,y,z) = 8xyz = \frac{8\sqrt{3}}{9}abc$$

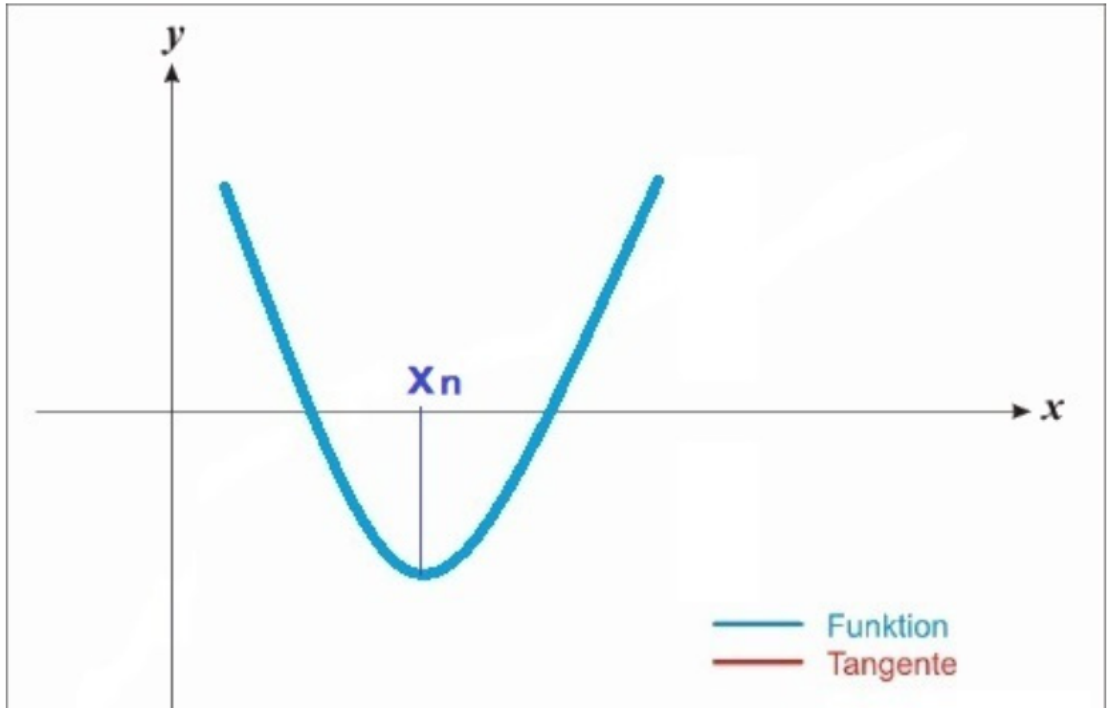
附录B　L-BFGS大规模优化算法

一、牛顿法

函数最优化算法方法不唯一，其中耳熟能详的包括梯度下降法，梯度下降法是一种基于迭代的一阶优化方法，优点是计算简单；牛顿法也是一种很重要的优化方法，是基于迭代的二阶优化方法，优点是迭代次数少，收敛速度很快。下面我们简要介绍一下牛顿法。

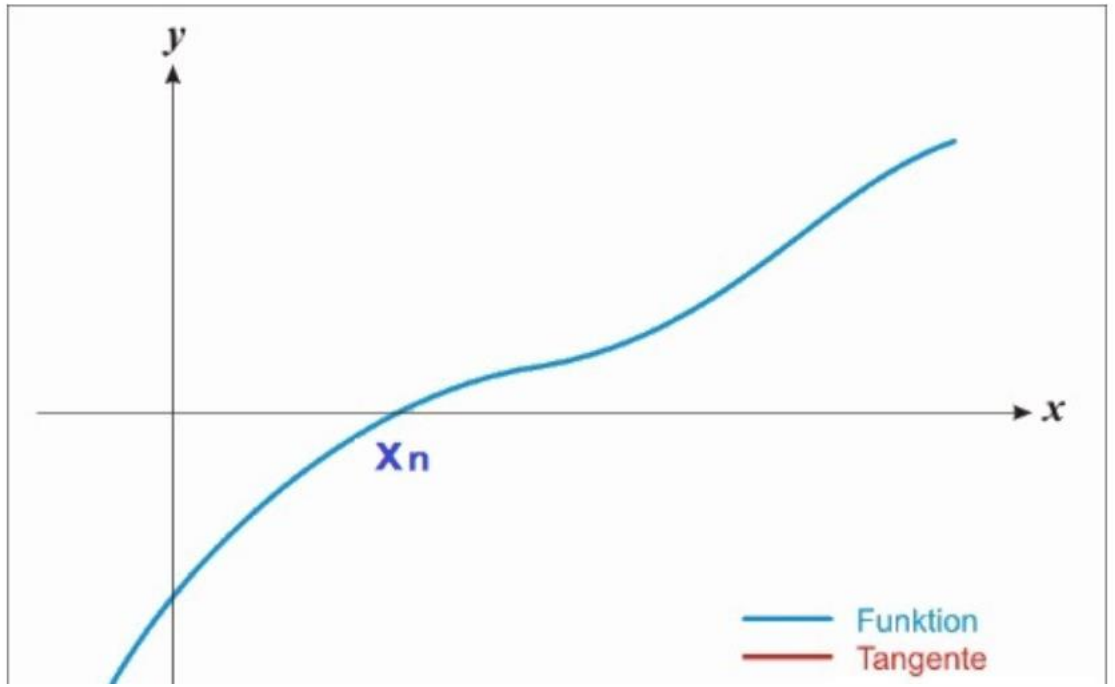
1.看图理解牛顿法

最优化问题就是寻找能使函数最小化的x，所以目标函数应当是一个凸函数（起码是局部凸函数），假如一个函数如下图：



附录B　L-BFGS大规模优化算法

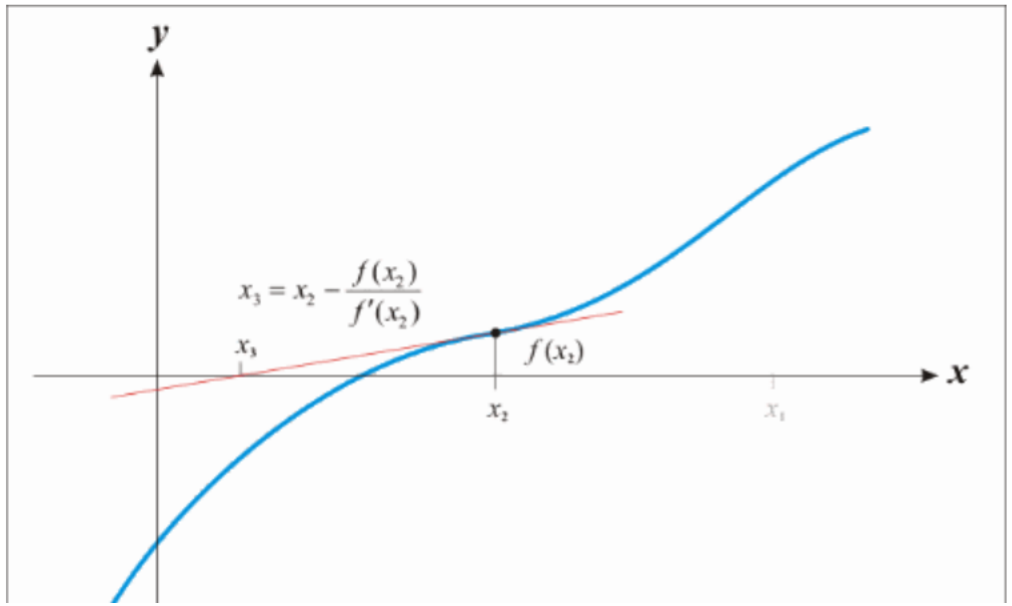
他的一阶导数可能长下面这个样子：



很显然函数在x_n处取得最小值，同时这个点的导数等于0，如果使用梯度下降，经过多次迭代，x的取值会慢慢接近x_n，我们都能想象这个过程。

附录B　L-BFGS大规模优化算法

如果使用牛顿法，x也会逼近x_n，不过速度会快很多，示例图如下：



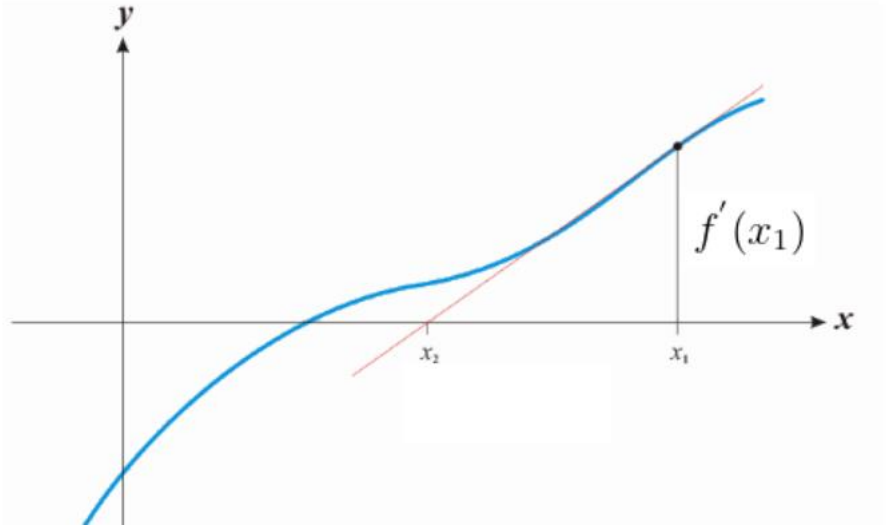
这个过程可以这样描述：

a.在X轴上随机一点x_1,经过x_1做X轴的垂线,得到垂线与函数图像的交点$f(x_1)$.

b.通过$f(x_1)$做函数的切线,得到切线与X轴的交点x_2.

c.迭代a/b两步，当前后两次求的x相同或者两个值的差小于一个阈值的时候，我们就认为找到了x_n。

附录B　L-BFGS大规模优化算法



如图4，蓝色的线是函数的$f(x)$的导数$f'(x)$，则曲线在x_1处的导数为$f''(x_1)$，我们要求x_2，根据三角函数有：

$$f''(x_1) = \frac{f'(x_1)}{x_1 - x_2} \tag{1}$$

得出：

$$x_2 = x_1 - \frac{f'(x_1)}{f''(x_1)} \tag{2}$$

利用x_2开始进行下一轮的迭代。迭代公式可以简化如下：

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)} \tag{3}$$

附录B　L-BFGS大规模优化算法

任意一点在x_k附近的二阶泰勒展开公式为：

$$f(x) = f(x_n) + f^{'}(x_n)(x - x_n) + \frac{1}{2}f^{''}(x_n)(x - x_n)^2 \qquad (4)$$

对$f(x)$求导：

$$f^{'}(x) = f^{'}(x_n) + f^{''}(x_n)(x - x_n) \qquad (5)$$

令$f^{'}(x) = 0$：

$$x = x_n - \frac{f^{'}(x_n)}{f^{''}(x_n)} \qquad (6)$$

写成迭代形式：

$$x_{n+1} = x_n - \frac{f^{'}(x_n)}{f^{''}(x_n)} \qquad (7)$$

可以看到使用三角函数和二阶泰勒展开最终得到的结果是一样的。虽然牛顿法收敛速度很快，但是当x的维度特别多的时候，我们想求得$f^{''}(x)$是非常困难的，而牛顿法又是一个迭代算法,所以这个困难我们还要面临无限多次，导致了直接使用牛顿法最为优化算法很难实际落地。为了解决这个问题出现了拟牛顿法，下面介绍一种拟牛顿法BFGS，主要就是想办法一种方法代替二阶导数。

附录B L-BFGS大规模优化算法

二、BFGS公式推导

函数$f(x)$在$x=x_{k+1}$处的二阶泰勒展开式为:

$$f(x)=f(x_{n+1})+f^{'}(x_{n+1})(x-x_{n+1})+\frac{1}{2}f^{''}(x_{n+1})(x-x_{n+1})^2 \tag{8}$$

当x为向量的时候，上式写成:

$$f(x)=f(x_{n+1})+\bigtriangledown f(x_{n+1})(x-x_{n+1})+\frac{1}{2}\bigtriangledown^2 f(x_{n+1})(x-x_{n+1})^2 \tag{9}$$

令$G_{n+1}=\bigtriangledown^2 f(x_{n+1})$，同时对$f(x)$求导:

$$\bigtriangledown f(x)=\bigtriangledown f(x_{n+1})+G_{n+1}(x-x_{n+1}) \tag{10}$$

接下来我们要想办法去掉G_{n+1}，我们使用B_{n+1}代替G_{n+1}，B_{n+1}是在迭代中一点点计算出来的而不使用二阶导数。

上式变为:

$$\bigtriangledown f(x)=\bigtriangledown f(x_{n+1})+B_{n+1}(x-x_{n+1}) \tag{11}$$

$$B_{n+1}(x_{n+1}-x)=\bigtriangledown f(x_{n+1})-\bigtriangledown f(x) \tag{12}$$

附录B　L-BFGS大规模优化算法

我们认为每次迭代B_k与上次变化E_k，形式如下：

$$B_{n+1} = B_n + E_n \tag{13}$$

令：

$$y_n = \bigtriangledown f(x_{n+1}) - \bigtriangledown f(x_n), \quad s_n = x_{n+1} - x_n \tag{14}$$

将式（13）（14）带入式子（12）：

$$(B_n + E_n)s_n = y_n \tag{15}$$

令：

$$E_n = \alpha u_n u_n^T + \beta v_n v_n^T \tag{16}$$

其中 u_n, v_n均为 n维的向量，带入（15）

$$(B_n + \alpha u_n u_n^T + \beta v_n v_n^T)s_n = y_n \tag{17}$$

$$\alpha(u_n^T s_n)u_n + \beta(v_n^T s_n)v_n = y_n - B_n s_n \tag{18}$$

附录B　L-BFGS大规模优化算法

已知：$u_n^T s_n, v_n^T s_n$ 为实数，$y_n - B_n s_n$ 为向量。式（18）中，参数 u_n 和 v_n 解的可能性有很多，我们取特殊的情况，假设 $u_n = rB_n s_n, v_n = \theta y_n$。带入（16）得：

$$E_n = \alpha r^2 B_n s_n s_n^T B_n + \beta \theta^2 y_n y_n^T \tag{19}$$

将 $u_n = rB_n s_n, v_n = \theta y_n$ 带入（18）得：

$$\alpha[(rB_n s_n)^T s_n](rB_n s_n) + \beta[(\theta y_n)^T](\theta y_n) = y_n - B_n s_n \tag{20}$$

$$[\alpha r^2 (s_n^T B_n s_n) + 1] + [\beta \theta^2 (y_n^T s_n) - 1](y_n) = 0 \tag{21}$$

令 $\alpha r^2 (s_n^T B_n s_n) + 1 = 0$，则：

$$\alpha r^2 = -\frac{1}{s_n^T B_n s_n} \tag{22}$$

令 $\beta \theta^2 (y_n^T s_n) - 1 = 0$，则

$$\beta \theta^2 = \frac{1}{y_n^T s_n} \tag{23}$$

附录B L-BFGS大规模优化算法

将式（22）和（23）带入（19）：

$$E_n = -\frac{B_n s_n s_n^T B_n}{s_n^T B_n s_n} + \frac{y_n y_n^T}{y_n^T s_k} \tag{24}$$

将（24）带入（13）得到B_n的迭代公式：

$$B_{n+1} = B_n - \frac{B_n s_n s_n^T B_n}{s_n^T B_n s_n} + \frac{y_n y_n^T}{y_n^T s_k} \tag{24}$$

当x为向量的时候，式（7）写成：

$$x_{n+1} = x_n - B_n^{-1} \bigtriangledown f(x_n) \tag{25}$$

加上学习率得到BFGS的迭代公式：

$$x_{n+1} = x_n - \eta(B_n^{-1} \bigtriangledown f(x_n)) \tag{26}$$

我们发现，还需要求B_n的逆，这里可以引入sherman-morrism公式，求解B_n的逆：

$$B_{n+1}^{-1} = (I - \frac{s_n y_n}{y_n^T s_n})^T B_n^{-1} (I - \frac{y_n s_n^T}{y_n^T s_n}) + \frac{s_n s_n^T}{y_n^T s_n} \tag{27}$$

我们用H代替B^{-1}，得到最终的BFGS迭代公式和H的迭代公式：

$$x_{n+1} = x_n - \eta(H_n \bigtriangledown f(x_n) \tag{28}$$

$$H_{n+1} = (I - \frac{s_n y_n}{y_n^T s_n})^T H_n (I - \frac{y_n s_n^T}{y_n^T s_n}) + \frac{s_n s_n^T}{y_n^T s_n} \tag{29}$$

其中s_n是本轮x与上一轮x的差，y_n是本轮梯度与上一轮梯度的差。

附录B　L-BFGS大规模优化算法

三、L-BFGS

在BFGS算法中，仍然有缺陷，每次迭代计算需要前次迭代得到的H_n，H_n的存储空间至少为N(N+1)/2（N为特征维数），对于高维的应用场景，需要的存储空间将是非常巨大的。为了解决这个问题，就有了L-BFGS算法。L-BFGS即Limited-memory BFGS。 L-BFGS的基本思想就是通过存储前m次迭代的少量数据来替代前一次的矩阵，从而大大减少数据的存储空间。

令$\rho_n = \frac{1}{y_n^T s_n}, V_k = I - \frac{y_n s_n^T}{y_n^T s_k}$，则式（29）可以表示为：

$$H_{n+1} = V_n^T H_n V_n + \rho_n s_n s_n^T \tag{30}$$

若在初始时，假定初始的矩阵$H_0 = I$，则我们可以得到：

$$H_1 = V_0^T H_0 V_0 + \rho_0 s_0 s_0^T \tag{31}$$

$$\begin{aligned} H_2 &= V_1^T H_1 V_1 + \rho_1 s_1 s_1^T \\ &= V_1^T (V_0^T H_0 V_0 + \rho_0 s_0 s_0^T) + \rho_1 s_1 s_1^T \\ &= V_1^T V_0^T H_0 V_1 + V_1^T \rho_0 s_0 s_0^T V_1 + \rho_1 s_1 s_1^T \end{aligned} \tag{32}$$

$$\begin{aligned} H_{n+1} = &(V_n^T V_{n-1}^T \cdots V_1^T V_0^T) H_0 (V_0 V_1 \cdots V_{n-1} V_n) \\ &+ (V_n^T V_{n-1}^T \cdots V_1^T) \rho_1 s_1 s_1^T (V_1 \cdots V_{n-1} V_n) \\ &+ \cdots \\ &+ V_n^T \rho_{n-1} s_{n-1} s_{n-1}^T V_n \\ &+ \rho_n s_n s_n^T \end{aligned}$$

附录B　L-BFGS大规模优化算法

假设当前迭代为n，只保存最近的m次迭代信息，按照上面的方式迭代m次，可以得到如下的公式：

$$
\begin{aligned}
H_{n+1} = &(V_n^T V_{n-1}^T \cdots V_{n-m}^T) H_0 (V_{n-m} \cdots V_{n-1} V_n) \\
&+ (V_n^T V_{n-1}^T \cdots V_{n-m}^T) \rho_1 s_1 s_1^T (V_{n-m} \cdots V_{n-1} V_n) \\
&+ \cdots \\
&+ V_n^T \rho_{n-1} s_{n-1} s_{n-1}^T V_n \\
&+ \rho_n s_n s_n^T
\end{aligned}
$$

由于ρ, V这些变量都最终可以由s、y两个向量计算得到，因此，我们只需存储最后m次的s、y向量即可算出H_{n+1}，加上对角阵H_0，总共需要存储2*m+1个N维向量（实际应用中m一般取4到7之间的值，因此需要存储的数据远小于Hesse矩阵）。

附录B　L-BFGS大规模优化算法

四、算法迭代过程

1. 选初始点x_0，最小梯度阈值$\varepsilon > 0$，存储最近 m 次的选代数据；

2.初始化$n = 0, H_0 = I, r = \bigtriangledown f(x_0)$；

3.如果$|| \bigtriangledown f(x_{n+1})|| \leqslant \varepsilon$，则返回最优解 x，否则转入步骤4；

4.计算本次选代的可行方向$p + n = -r_k$；

5.计算步长α_k，用下面的式子进行线搜索；

$$f(x_n + \alpha_n p_n) = min f(x_n - \alpha p_n)$$

6.用下面的更新公式更新x；

$$x_{n+1} = x_n + \alpha_n p_n$$

附录B　L-BFGS大规模优化算法

7.如果 n大于 m，保留最近 m 次的向量对，删除s_{n-m}, y_{n-m};

8.计算并保存向量对

$$s_n = x_{n+1} - x_n$$

$$y_n = \bigtriangledown f(x_{n+1}) - \bigtriangledown f(x_n)$$

9.用 two-loop recursion算法求：

$$r_n = B_n \bigtriangledown f(x_n)$$

10.设置$n = n + 1$，转到步骤3