别让数据坑了你！用置信学习找出错误标注（附开源实现）

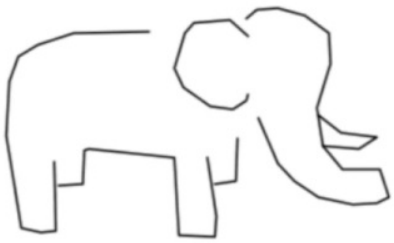
JayLou娄杰 夕小瑶的卖萌屋 昨天

星标/置顶小屋，带你解锁
最萌最前沿的NLP、搜索与推荐技术

文 | JayLou娄杰（NLP算法工程师，信息抽取方向）
编 | 北大小才女小轶
美 | Sonata

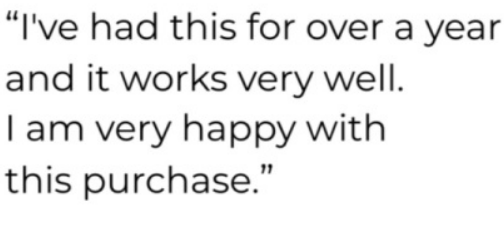
1 前言

在实际工作中，你是否遇到过这样一个问题或痛点：无论是通过哪种方式获取的标注数据，数据标注质量可能不过关，存在一些错误？亦或者是数据标注的标准不统一、存在一些歧义？特别是badcase反馈回来，发现训练集标注的居然和badcase一样？如下图所示，QuickDraw、MNIST和Amazon Reviews数据集中就存在错误标注。



Dataset: **Google Quickdraw!**
Given Label: **Mosquito**
Model: VGG

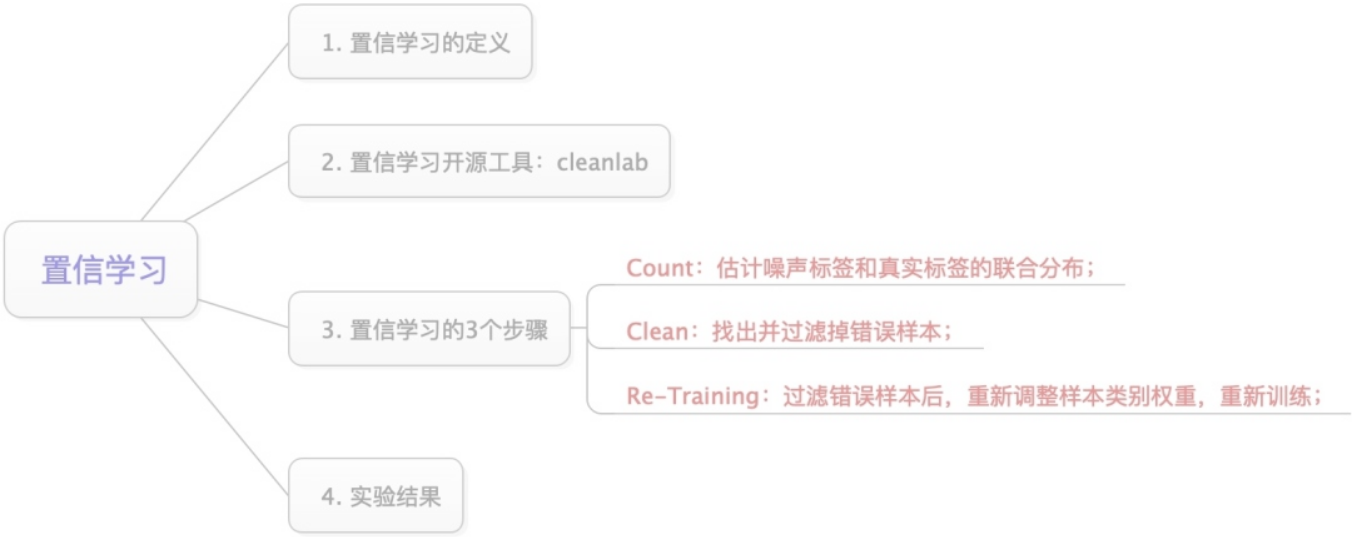
Dataset: **MNIST**
Given Label: **5**
Model: AlexNet



Dataset: **Amazon Reviews**
Given Label: **1 star review**
Model: SGD Classifier + TFIDF

为了快速迭代，大家是不是常常直接人工去清洗这些"脏数据"？（笔者也经常这么干～）。但数据规模上来了咋整？有没有一种方法能够自动找出哪些错误标注的样本呢？基于此，本文尝试提供一种可能的解决方案——**置信学习**。

本文的组织架构是：



2 置信学习

2.1 置信学习的定义

那什么是置信学习呢?这个概念来自于ICML2020的一篇由MIT和Google联合提出的paper:《[Confident Learning: Estimating Uncertainty in Dataset Labels][1] 》。论文提出的 **置信学习**(confident learning， **CL**）**是一种新兴的、具有原则性的框架，以识别标签错误、表征标签噪声并应用于带噪学习**（noisy label learning）。

原文链接：https://arxiv.org/abs/1911.00068 Arxiv访问慢的小伙伴也可以在订阅号后台回复关键词【**0630**】下载论文PDF。

> 笔者注：笔者乍一听「置信学习」挺陌生的，但回过头来想想，好像干过类似的事情，比如：在某些场景下，对训练集通过交叉验证来找出一些可能存在错误标注的样本，然后交给人工去纠正。此外，神经网络的成功通常建立在大量、干净的数据上，标注错误过多必然会影响性能表现，**带噪学习**可是一个大的topic，有兴趣可参考这些文献 https://github.com/subeeshvasu/Awesome-Learning-with-Label-Noise。

废话不说，首先给出这种置信学习框架的**优势**:

- 最大的优势:**可以用于发现标注错误的样本!**
- 无需迭代,**开源了相应的python包,方便地快速使用!**在ImageNet中查找训练集的标签错误仅仅需要3分钟!
- 可直接估计噪声标签与真实标签的联合分布,**具有理论合理性**。
- 不需要超参数,只需使用交叉验证来获得样本外的预测概率。
- 不需要做随机均匀的标签噪声的假设(这种假设在实践中通常不现实)。
- 与模型无关,可以使用任意模型,不像众多带噪学习与模型和训练过程强耦合。

> 笔者注：置信学习找出的「标注错误的样本」，不一定是真实错误的样本，这是一种基于不确定估计的选择方法。

2.2 置信学习开源工具：cleanlab

论文最令人惊喜的一点就是作者这个置信学习框架进行了开源，并命名为cleanlab，我们可以 `pip install cleanlab` 使用。

cleanlab

我们要想找出错误标注的样本，通过使用**cleanlab**操作十分简单，我们仅仅需要提供两个输入，然后只需要1行code就可以找出标注数据中的错误：

```
from cleanlab.pruning import get_noise_indices
# 输入
# s:噪声标签
# psx: n x m 的预测概率概率，通过交叉验证获得
ordered_label_errors = get_noise_indices(
    s=numpy_array_of_noisy_labels,
    psx=numpy_array_of_predicted_probabilities,
    sorted_index_method='normalized_margin', # Orders label error
s
)
```

这个输入是啥？很简单，一个输入是原始的样本标签（由于这些标签可能存在错误，我们称之为「噪声标签」吧～），另一个输入就是通过对训练集交叉验证，来预测的每一个样本在不同标签类别下的概率，这是一个nXm的概率矩阵（n为数据集大小，m为标签类别总数）。

我们来看看**cleanlab**在MINIST数据集中找出的错误样本吧，是不是感觉很 ～

MINIST

如果你不只是想找到错误标注的样本，还想把这些标注噪音clean掉之后重新继续学习，那3行codes也可以搞定，这时候连交叉验证都省了～

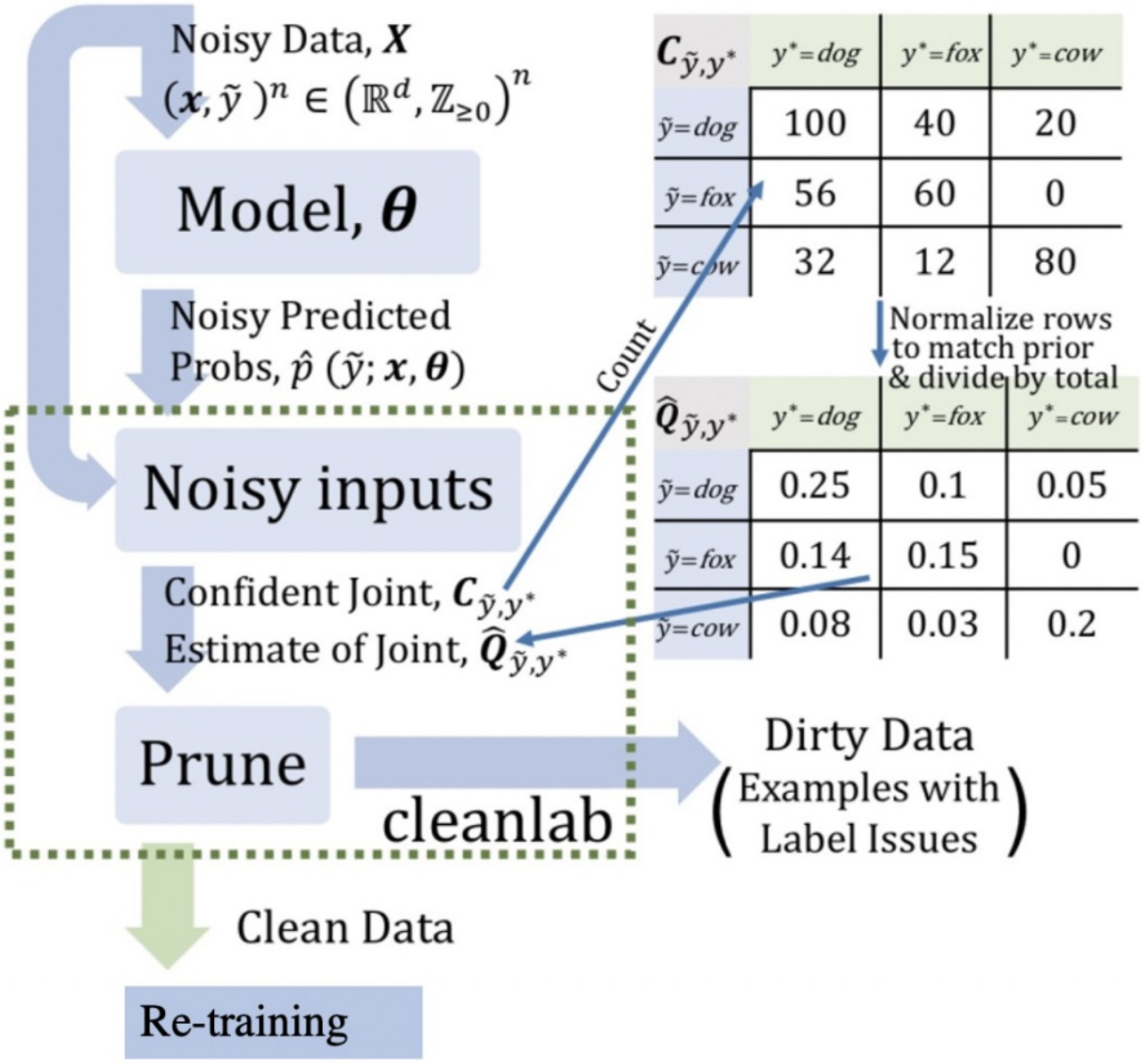
```
from cleanlab.classification import LearningWithNoisyLabel
s
from sklearn.linear_model import LogisticRegression

# 其实可以封装任意一个你自定义的模型.
lnl = LearningWithNoisyLabels(clf=LogisticRegression())
lnl.fit(X=X_train_data, s=train_noisy_labels)
# 对真实世界进行验证.
predicted_test_labels = lnl.predict(X_test)
```

> 笔者注：上面虽然只给出了CV领域的例子，但置信学习也适用于NLP啊～此外，`cleanlab`可以封装任意一个你自定义的模型，以下机器学习框架都适用：scikit-learn, PyTorch, TensorFlow, FastText。

2.3 置信学习的3个步骤

置信学习开源工具`cleanlab`操作起来比较容易，但置信学习背后也是有着充分的理论支持的。事实上，一个完整的置信学习框架，需要完成以下三个步骤（如置信学习框架图所示）：

1. **Count**：估计噪声标签和真实标签的联合分布；
2. **Clean**：找出并过滤掉错误样本；
3. **Re-Training**：过滤错误样本后，重新调整样本类别权重，重新训练；



置信学习框架

接下来，我们对上述3个步骤进行详细阐述。

2.3.1 Count：估计噪声标签和真实标签的联合分布

我们定义噪声标签为 $\tilde{y}$，即经过初始标注（也许是人工标注）、但可能存在错误的样本；定义真实标签为 y^*，但事实上我们并不会获得真实标签，所以通常是采取**交叉验证**对真实标签进行估计。此外，定义样本总数为 n ，类别总数为 m 。

为了估计联合分布，共需要4步：

- step 1 ：交叉验证：
  - 首先需要通过对数据集集进行交叉验证，并计算第i个样本在第 j个类别下的概率$P[i][j]$；
  - 然后计算每个人工标定类别j下的平均概率$t[j]$ 作为置信度阈值；
  - 最后对于样本i，其真实标签y^*为j个类别中的最大概率$argmax_j P[i][j]$，并且$P[i][j] > t[j]$
- step 2 ：计算计数矩阵$C_{\tilde{y},y^*}$（类似于混淆矩阵），如图1中的 $C_{\tilde{y}=dog,y^*=fox} = 40$ 意味着，人工标记为dog但实际为fox的样本为40个。具体的操作流程如下图所示：

PART 1 (COMPUTE THRESHOLDS)
for $j \leftarrow 1, m$ **do**
  for $i \leftarrow 1, n$ **do**
    $l \leftarrow$ new empty list []
    if $\tilde{\boldsymbol{y}}[i] = j$ **then**
      append $\hat{\boldsymbol{P}}[i][j]$ to l
  $\boldsymbol{t}[j] \leftarrow \text{average}(l)$
PART 2 (COMPUTE CONFIDENT JOINT)
$\boldsymbol{C} \leftarrow m \times m$ matrix of zeros
for $i \leftarrow 1, m$ **do**
  $cnt \leftarrow 0$
  for $j \leftarrow 1, m$ **do**
    if $\hat{\boldsymbol{P}}[i][j] \geq \boldsymbol{t}[j]$ **then**
      $cnt \leftarrow cnt + 1$
      $y^* \leftarrow j$
  $\tilde{y} \leftarrow \tilde{\boldsymbol{y}}[i]$
  if $cnt > 1$ **then**
    $y^* \leftarrow \arg\max \hat{\boldsymbol{P}}[i]$
  if $cnt > 0$ **then**
    $\boldsymbol{C}[\tilde{y}][y^*] \leftarrow \boldsymbol{C}[\tilde{y}][y^*] + 1$
output $\boldsymbol{C}$ $_{m \times m}$ unnormalized counts matrix

计数矩阵C计算流程

- step 3：标定计数矩阵：目的就是为了让计数总和与人工标记的样本总数相同。计算公式如下面所示，其中$|X_{\tilde{y}=i}|$为人工标记标签$\tilde{y}=i$的样本总个数：

$$\tilde{C}_{\tilde{y}=i,y^*=j} = \frac{C_{\tilde{y}=i,y^*=j}}{\sum_{j\in 1,2,3...,m} C_{\tilde{y}=i,y^*=j}} \times |X_{\tilde{y}=i}| \tag{1}$$

- step 4：估计噪声标签$\tilde{y}$和真实标签y^*的联合分布$Q_{\tilde{y},y^*}$，可通过下式求得：

$$Q_{\tilde{y}=i,y^*=j} = \frac{\tilde{C}_{\tilde{y}=i,y^*=j}}{\sum_{i\in 1,...,m,j\in 1,...,m} \tilde{C}_{\tilde{y}=i,y^*=j}} \tag{2}$$

看到这里，也许你会问为什么要估计这个联合分布呢?其实这主要是为了下一步方便我们去clean噪声数据。此外，这个联合分布$Q_{\tilde{y},y^*}$其实能充分反映真实世界中噪声（错误）标签和真实标签的分布，随着数据规模的扩大，这种估计方法与真实分布越接近（原论文中有着严谨的证明，由于公式推导繁杂这里不再赘述，有兴趣的同学可以详细阅读原文～后文也有相关实验进行证明）。

看到这里，也许你还感觉公式好麻烦，那下面我们通过一个具体的例子来展示上述计算过程：

- step 1：通过交叉验证获取第i样本在第j个类别下的概率$P[i][j]$；为说明问题，这里假设共10个样本、2个类别，每个类别有5个样本。经过计算每个人工标签类别j下的平均概率$t[j]$分别为：$t[0]=0.58, t[1]=0.66$.

P[i][j]	j=0	j=1	人工标签
i=0	0.9	0.1	0
i=1	0.9	0.1	0
i=2	0.5	0.5	0
i=3	0.3	0.7	0
i=4	0.3	0.7	0
i=5	0.2	0.9	1
i=6	0.2	0.8	1
i=7	0.4	0.7	1
i=8	0.5	0.5	1
i=9	0.6	0.4	1
t[j]	0.58	0.66	

P[i]和t[j]计算

- step2: 根据计算流程和上图结果，我们可以很容易得到计数矩阵 $C_{\tilde{y},y^*}$ 为：

	y*=0	y*=1
y~=0	2	2
y~=1	1	3

计数矩阵C计算

- step3: 标定后的计数矩阵 $\tilde{C}_{\tilde{y},y^*}$为（计数总和与人工标记的样本总数相同，将原来的样本总数进行加权即可，以$\tilde{C}_{\tilde{y}=1,y^*=1}$为例，根据公式(1),其计算为$5\times\frac{3}{1+3}=3.75$）：

	y*=0	y*=1
y~=0	2.5	2.5
y~=1	1.25	3.75

标定计数矩阵

- step4：联合分布$Q_{\tilde{y},y^*}$为：（根据公式(2)直接进行概率归一化即可）

	y*=0	y*=1
y~=0	0.25	0.25
y~=1	0.125	0.375

联合分布Q计算

2.3.2 Clean：找出并过滤掉错误样本

在得到噪声标签和真实标签的联合分布$Q_{\tilde{y},y^*}$，论文共提出了5种方法过滤错误样本。

- Method 1: $C_{confusion}$，选取 的样本进行过滤，即选取$P[i][j]$最大概率对应的下标j与人工标签不一致的样本。
- Method 2:$C_{\tilde{y},y^*}$，选取构造计数矩阵$C_{\tilde{y},y^*}$过程中、进入非对角单元的样本进行过滤。
- Method 3: **Prune by Class (PBC)**，即对于人工标记的每一个类别$i \in 1,2,...,m$选取个样本过滤，并按照最低概率$p(\tilde{y},x,\theta)$排序。
- Method 4: **Prune by Noise Rate (PBNR)**，对于计数矩阵$C_{\tilde{y},y^*}$的非对角单元，选取$n \cdot Q_{\tilde{y},y^*}$个样本进行过滤，并按照最大间隔$p_{\tilde{y}=j} - p_{\tilde{y}=i}$排序。
- Method 5: **C+NR**，同时采用Method 3和Method 4.

我们仍然以前面给出的示例进行说明：

- Method 1：过滤掉i=2,3,4,8,9共5个样本；
- Method 2：进入到计数矩阵非对角单元的样本分别为i=3,4,9,将这3个样本过滤；
- Method 3：对于类别0，选取 $10 * 0.25 = 2.5 \approx 3$ 个样本过滤，按照最低概率排序，选取i=2,3,4；对于类别1，选取$10 * 0.125 = 1.25 \approx 1$个样本过滤,按照最低概率排序选取i=9；综上，共过滤i=2,3,4,9共4个样本；
- Method 4：对于非对角单元$C_{\tilde{y}=0,y^*=1}$选取i=2,3,4过滤，对$C_{\tilde{y}=1,y^*=0}$选取i=9过滤。

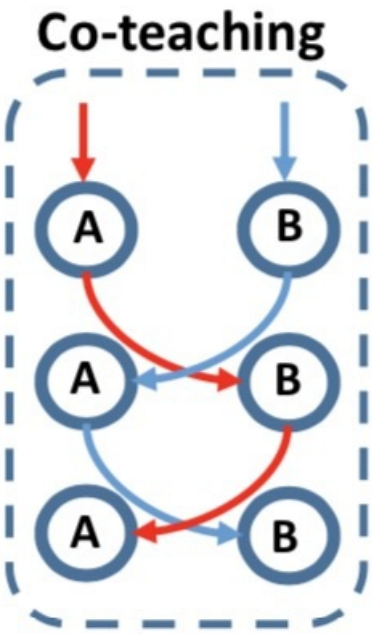
上述这些过滤样本的方法在 `cleanlab` 也有提供，我们只要提供2个输入、1行code即可clean错误样本：

```
import cleanlab
# 输入
# s:噪声标签
# psx: n x m 的预测概率概率，通过交叉验证获得
# Method 3：Prune by Class (PBC)
baseline_cl_pbc = cleanlab.pruning.get_noise_indices(s, psx, prune_method='prune_by_class',n_jobs=1)
# Method 4：Prune by Noise Rate (PBNR)
baseline_cl_pbnr = cleanlab.pruning.get_noise_indices(s, psx, prune_method='prune_by_noise_rate',n_jobs=1
)
# Method 5：C+NR
baseline_cl_both = cleanlab.pruning.get_noise_indices(s, psx, prune_method='both',n_jobs=1)
```

2.3.3 Re-Training：过滤错误样本后，重新训练

在过滤掉错误样本后，根据联合分布$Q_{\tilde{y},y^*}$ 将每个类别i下的损失权重修正为$\frac{Q_{y^*}[i]}{Q_{\tilde{y},y^*}[i][i]}$:

其中.然后采取Co-Teaching[2]框架进行。

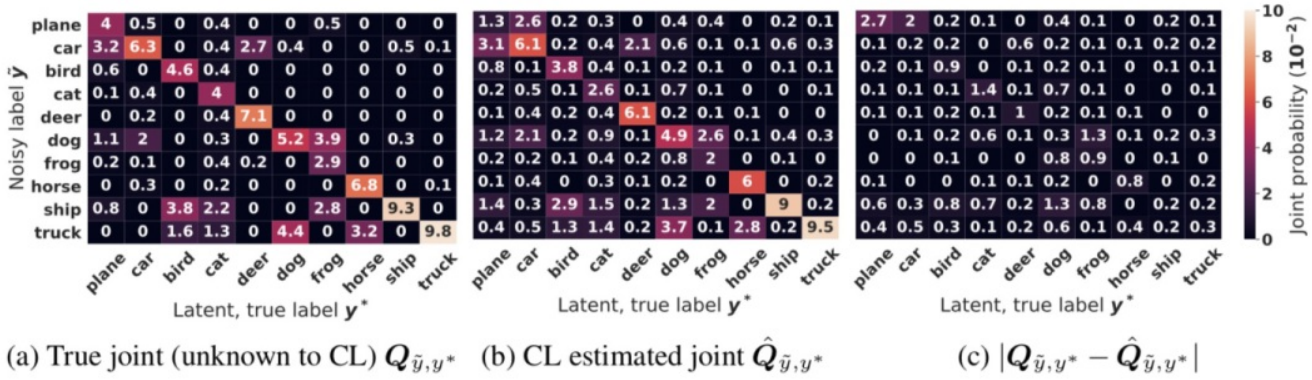


Co-teaching

如上图所示，Co-teaching的基本假设是认为noisy label的loss要比clean label的要大，于是它并行地训练了两个神经网络A和B，在每一个Mini-batch训练的过程中，每一个神经网络把它认为loss比较小的样本，送给它其另外一个网络，这样不断进行迭代训练。

2.4 实验结果

上面我们介绍完成置信学习的3个步骤，本小节我们来看看这种置信学习框架在实践中效果如何？在正式介绍之前，我们首先对**稀疏率**进行定义：稀疏率为联合分布矩阵、非对角单元中0所占的比率，这意味着真实世界中，总有一些样本不会被轻易错标为某些类别，如「老虎」图片不会被轻易错标为「汽车」。



(a) True joint (unknown to CL) $Q_{\tilde{y},y^*}$ (b) CL estimated joint $\hat{Q}_{\tilde{y},y^*}$ (c) $|Q_{\tilde{y},y^*} - \hat{Q}_{\tilde{y},y^*}|$

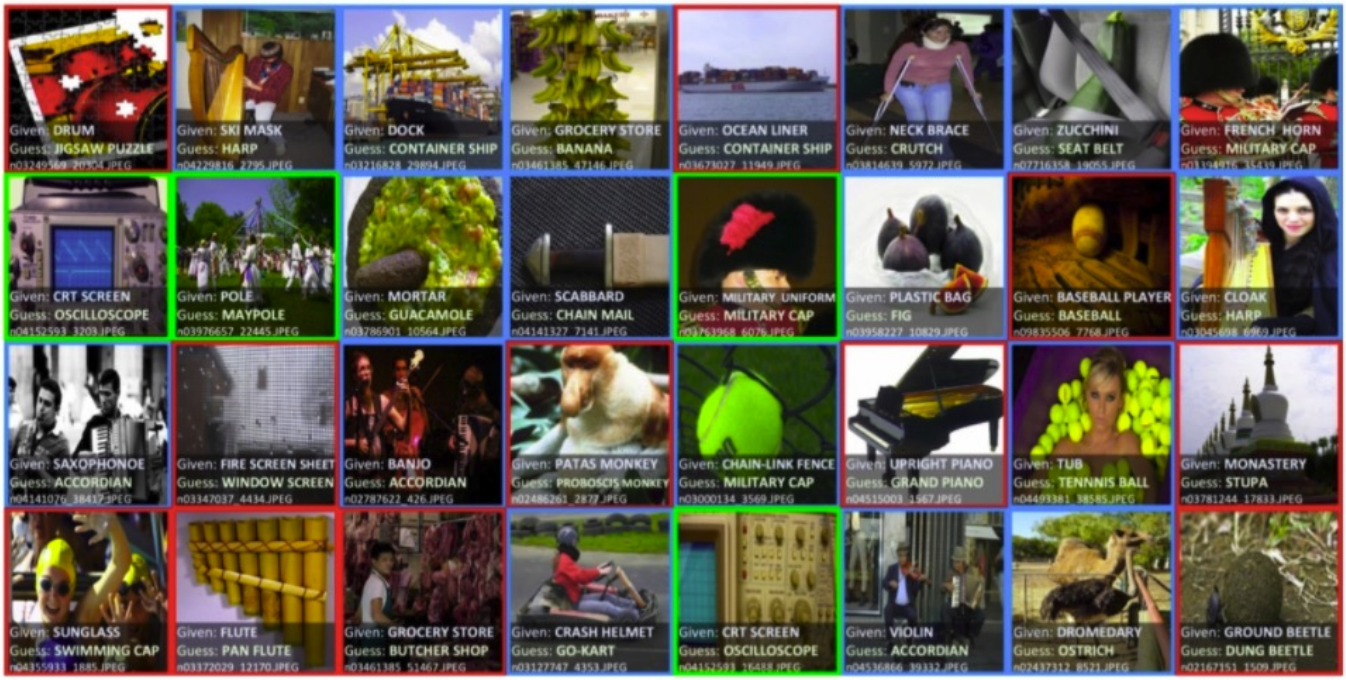
真实联合分布和估计联合分布

上图给出了CIFAR-10中，噪声率为40%和稀疏率为60%情况下，真实联合分布和估计联合分布之间的比较，可以看出二者之间很接近，可见论文提出的置信学习框架用来估计联合分布的有效性；当然，论文也对联合估计有着严谨的推导证明。

NOISE	0.2				0.4				0.7			
SPARSITY	0	0.2	0.4	0.6	0	0.2	0.4	0.6	0	0.2	0.4	0.6
CL: $C_{\text{CONFUSION}}$	0.854	0.854	0.863	0.857	0.806	0.796	0.802	0.798	0.332	0.363	0.328	0.291
CL: $C_{\tilde{y},y^*}$	0.848	0.858	0.862	0.861	**0.815**	**0.810**	**0.816**	0.815	0.340	0.398	0.282	**0.372**
CL: PBC	**0.860**	0.854	0.862	0.855	0.802	0.801	0.810	0.813	0.356	0.373	0.263	0.336
CL: PBNR	0.858	0.854	**0.865**	**0.862**	0.810	0.796	0.814	**0.825**	**0.468**	0.416	**0.399**	0.360
CL: C+NR	0.858	**0.859**	0.862	**0.862**	0.808	0.800	0.807	0.822	0.460	**0.420**	0.382	0.371
MENTORNET	0.849	0.851	0.832	0.834	0.644	0.642	0.624	0.615	0.300	0.316	0.293	0.279
S-MODEL	0.800	0.800	0.797	0.791	0.586	0.612	0.591	0.575	0.284	0.285	0.279	0.273
REED	0.781	0.789	0.808	0.793	0.605	0.604	0.612	0.586	0.290	0.294	0.291	0.268
BASELINE	0.784	0.792	0.790	0.782	0.602	0.608	0.596	0.573	0.270	0.297	0.282	0.268

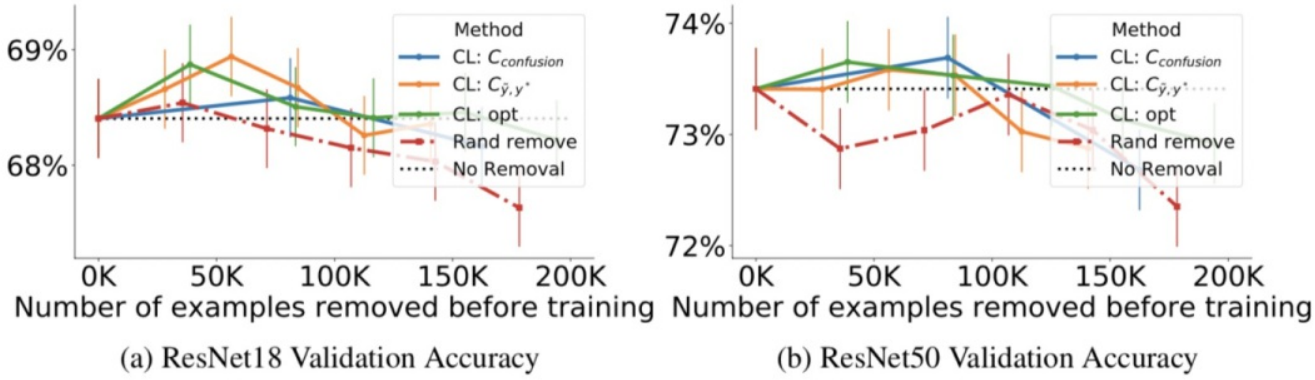
不同置信学习方法的比较

上图给出了CIFAR-10中不同噪声情况和稀疏性情况下，置信学习与噪声学习中的其他SOTA方法的比较。例如在40%的噪声率下，置信学习比之前SOTA方法Mentornet的准确率平均提高34%。



置信学习发现的 ImageNet标签问题

论文还将提出置信学习框架应用于真实世界的ImageNet数据集，利用置信学习的**PBNR**方法找出的TOP32标签问题如上图所示，置信学习除了可以找出标注错误的样本（红色部分），也可以发现多标签问题（蓝色部分，图像可以有多个标签），以及**本体论问题**：绿色部分，包括“是”（比如：将浴缸标记为桶）或 “有”（比如：示波器标记为CRT屏幕）两种关系。



(a) ResNet18 Validation Accuracy　　(b) ResNet50 Validation Accuracy

不同置信学习方法和随机去除的对比

上图给出了分别去除20%，40%...，100%估计错误标注的样本后训练的准确性，最多移除200K个样本。可以看出，当移除小于100K个训练样本时，置信学习框架使得准确率明显提升，并优于随机去除。

3 总结

- 本文介绍了一种用来刻画noisy label、找出错误标注样本的方法——置信学习，是弱监督学习和带噪学习的一个分支。
- 置信学习直接估计噪声标签和真实标签的联合分布，而不是修复噪声标签或者修改损失权重。
- 置信学习开源包cleanlab可以很快速的帮你找出那些错误样本！可在分钟级别之内找出错误标注的样本。

接下来，让我们尝试将置信学习应用于自己的项目，找出那些“可恶”的数据噪声吧～

参考文献

[1] Confident Learning: Estimating Uncertainty in Dataset Labels

[2] Co-teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels

夕小瑶的卖萌屋

关注&星标小夕，带你解锁AI秘籍

订阅号主页下方「撩一下」有惊喜哦