Label Noisy Representation Learning

Wentao Mo[1]

[1]Department of Machine Intelligence
Peking University

2021 年 4 月 24 日

Outline

LNRL in Chart

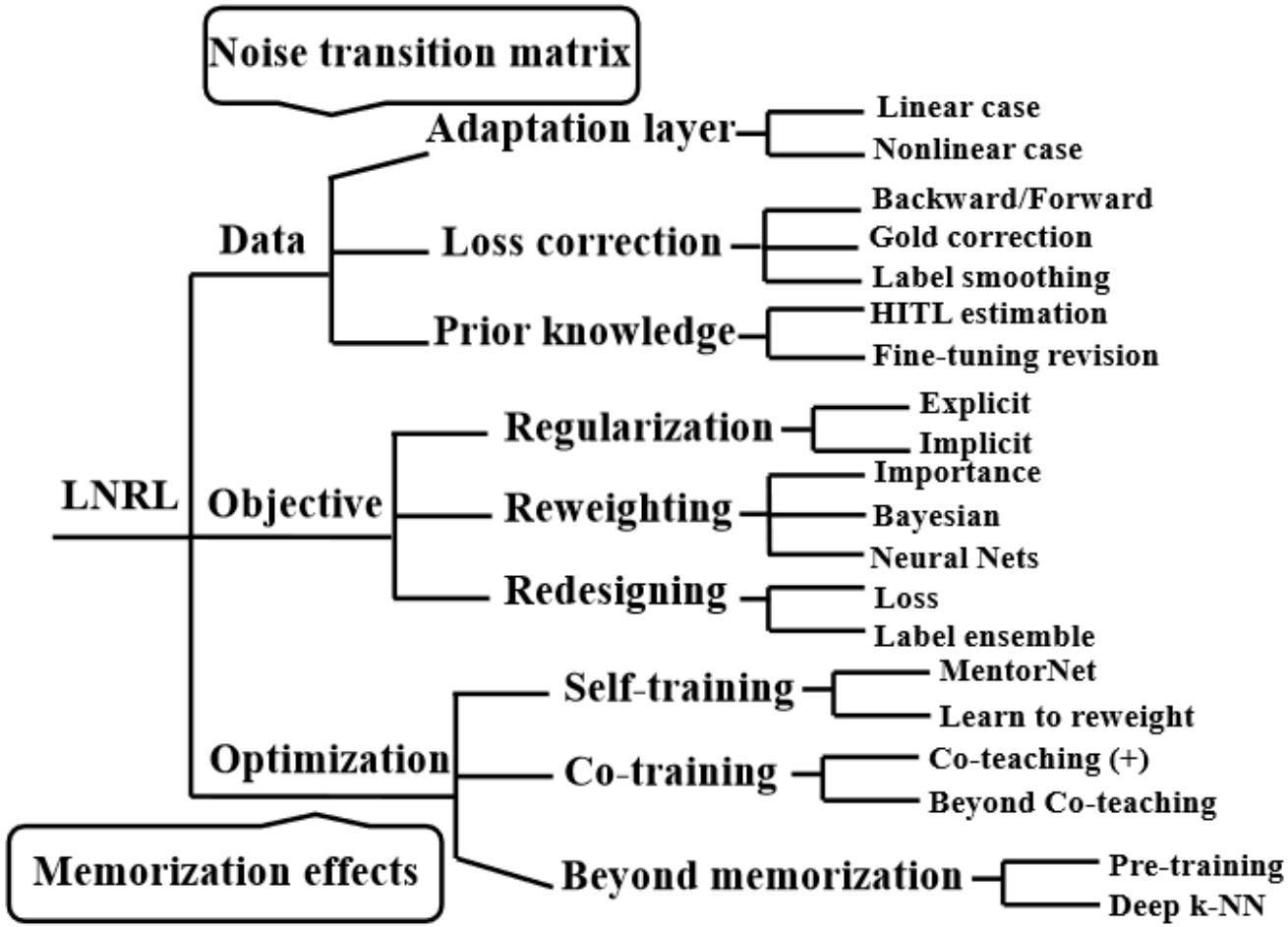

Fig. 2. A taxonomy of LNRL based on the focus of each method. For each technique branch, we list a few representative works here.

Noise Transition Matrix, Forward/Backward Correction

定义

(Noise transition matrix) Suppose that the observed noisy label $\bar{y}$ is drawn independently from a corrupted distribution $p(X, \bar{Y})$, where features are intact. Meanwhile, there exists a corruption process, transition from the latent clean label y to the observed noisy label $\bar{y}$. Such a corruption process can be approximately modeled via a noise transition matrix T, where $T_{ij} = p\left(\bar{y} = e_j \mid y = e_i\right)$

两种经典的 (合成) 噪声转移矩阵, 对称 flipping/配对 flipping

$$\begin{bmatrix} 1-\tau & \frac{\tau}{n-1} & \cdots & \frac{\tau}{n-1} \\ \frac{\tau}{n-1} & 1-\tau & & \frac{\tau}{n-1} \\ \vdots & & \ddots & \vdots \\ \frac{\tau}{n-1} & \frac{\tau}{n-1} & \cdots & 1-\tau \end{bmatrix} \quad \begin{bmatrix} 1-\tau & \tau & 0 & 0 \\ 0 & 1-\tau & \tau & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & & \tau \\ \tau & 0 & \cdots & 1-\tau \end{bmatrix}$$

实际中噪声不一定形式这么好/对称.

Remark 在合成噪声和实际噪声之间存在 domain gap.

Estimate Noise Transition Matrix T

定义

后向矫正

$$\ell^{\leftarrow}(f(x), \bar{y}) = \left[T^{-1}\ell_{y|f(x)}\right]_{\bar{y}} \tag{1}$$

可以证明后向矫正 loss 是 clean label loss 的无偏估计.

定义

前向矫正

$$\ell^{\rightarrow}(f(x), \bar{y}) = \left[\ell_{y|T^{\top}f(x)}\right]_{\bar{y}} \tag{2}$$

可以证明前向矫正 loss 和 clean label loss 上有相同的极小值.

Estimate Noise Transition Matrix T

1. (Patrini et al., 2017) 提出一个两阶段训练. 首先使用 noisy data 训练网络, 再获得一个 T 的估计, 再重新训练网络, 使用 T 校正的 loss.
2. (Hendrycks et al., 2018) 提出了 Gold 校正来处理严重噪声. 关键思路是, 假设一部分数据是可信的且可用的, 比如有一些专家来得出的 trusted set D. 他们使用 D 来估计 T, 再用前向矫正来训练 DNN, 这就是 GLC.
3. 使用 Label Smoothing. 本质上是后向矫正, 且 $T^{-1} = (1-\alpha)I + \frac{\alpha E}{L}$

Estimate Noise by Adaptation Layer

(Sukhbattar, 2015) 提出了在网络输出之后增加一个参数化 T 的 adapt. layer. 单独用 CE 优化两个不同的模块并不能达到 optimal 的 T. 他们又增加了一个 T 的正则化项 trace norm.
(Goldberger et al., 2017) 使用了 base model param. by ω, 以及噪声模型 param. by θ. 既然 base model 的输出是 hidden 的, 那么他们用 EM 算法来估计隐藏输出 (E-step), 以及当前的参数 (M-step). EM 也会导致局部最优和可伸缩性的问题.

Regularization: Explicit

1. (Azadi et al., 2016) 提出了一种正则化项 $\Omega_{\text{aux}}(w) = \|Fw\|_{\text{g}}$ 其中 $\|\cdot\|_g$ 是 group norm, $F^\top = [X_1, \ldots, X_n], X_i = \text{Diag}(\boldsymbol{x}_i)$, 鼓励稀疏性. 这会鼓励一小部分 clean data 来 control model.
2. (Berthelot et al., 2019) 提出了 MixMatch 来进行 SSL. 其中的一个关键部分是 Minimum Ent. Reg.(MER), 也是一种显式正则化. MER 提出于 (Grandvalet & Bengio, 2005), 关键 idea 是把 CE 加入一个正则项, 鼓励在 unlabeled data 上给出 high-confidence 的输出, 具体地, 最小化在 unlabeled 数据上的熵.
3. 类似于 MER, psedo-label 方法 (D.-H. Lee,, 2013)(i.e. lebel guessing) 进行隐式的 ent. 最小化. 具体上讲, 首先计算模型 (通过各种 augmentation) 预测的类型分布, 再通过 temperature sharpening func. 来最小化 label dist. 的熵.

Regularization: Explicit

(Miyato et al., 2018) 提出了一个 virtual adversarial loss, 使用 VA direction, 一种无标签的对抗样本生成法, 类似 FGSM/PGD 但利用了二阶梯度的估计.

定义 $D\left(r, x_*, \theta\right) := D\left[p\left(y \mid x_*, \hat{\theta}\right), p\left(y \mid x_* + r, \theta\right)\right]$ 由于在 $r = 0$ 时, $\nabla_r D(r, x, \hat{\theta})\Big|_{r=0} = 0$, 那么有二阶估计

$$D(r, x, \hat{\theta}) \approx \frac{1}{2} r^T H(x, \hat{\theta}) r \tag{3}$$

那么 r_{vadv} 可以是 Hessian 的第一个 dominant eigenvector 具有长度 ϵ

$$\begin{aligned} r_{\text{vadv}} &\approx \arg\max_r \left\{ r^T H(x, \hat{\theta}) r; \|r\|_2 \leq \epsilon \right\} \\ &= \epsilon \overline{u(x, \hat{\theta})}, \end{aligned} \tag{4}$$

Regularization: Explicit

为了避免直接计算 H, 使用有限差分法来估计这个乘积, 随机采样一个 unit vector$\boldsymbol{d}$, 迭代计算 mat-vec prod. $d \leftarrow \overline{Hd}$.

$$\begin{aligned} Hd &\approx \frac{\nabla_r D(r, x, \hat{\theta})\Big|_{r=\xi d} - \nabla_r D(r, x, \hat{\theta})\Big|_{r=0}}{\xi} \\ &= \frac{\nabla_r D(r, x, \hat{\theta})\Big|_{r=\xi d}}{\xi} \end{aligned} \tag{5}$$

i.e.,

$$d \leftarrow \overline{\nabla_r D(r, x, \hat{\theta})\Big|_{r=\xi d}} \tag{6}$$

他们发现一步迭代就能达到类似 FGSM 里的估计精度.

Regularization: Implicit

(Reed et al., 2015) 的 Bootsrapping. 学习器和自己 bootstrap, 使用 label 和模型目前的 prediction 的凸组合来生成训练目标. 直觉上, 随着 learner 学习, predictions 也变得可信. 进而避免对 noise 的直接建模. 具体地, 有 soft/hard 两种 bootstrapping. 对于 soft bootstrapping, 使用预测的类概率 q 来得到回归目标.

$$\ell_{\text{soft}}(q,t) = \sum_{k=1}^{L} [\beta t_k + (1-\beta) q_k] \log(q_k) \tag{7}$$

这等价于 softmax regression + MER.
对于 hard bootstrapping, 使用 q 的 MAP 估计.

$$\ell_{\text{hard}}(q,t) = \sum_{k=1}^{L} [\beta t_k + (1-\beta) z_k] \log(q_k) \tag{8}$$

其中 $z_k = \mathbf{1}[k = \arg\max_{i=1,\ldots,L} q_i]$ 为了能够优化 hard 版本, 需要使用 EM-like 算法. E-step 中计算凸组合的 targets, M-step 根据 targets 进行优化参数.

Regularization: Implicit

(Zhang et al., 2018) 的 Mixup. 这显然也是一种 label regularization.
(Han et al., 2020) 的 SIGUA(data-agnostic). 注意到随着网络容量的提升, 网络能逐渐地 overfit noisy data. 所以他们提出了 Stochastic Integrated Gradient Underweighted Ascent(SIGUA) 的一种**训练策略**, 在一个 mini-batch 中, 线照常使用 SGD, 再在 bad-data 上使用 (lr 递减的) 梯度递增. 在训练哲学上, SIGUA 让网络忘记不想要的记忆, 来更好的加强想要的记忆.

Objective Reweighting: Importance Reweighting, Bayesian Methods

所谓 Objective Reweighting, 是指 $\ell = \sum_k \lambda_k \ell_k$

(Liu and Tao, 2015) 使用 importance reweighting 来 LNL. 将 noisy data 作为 source domain, clean data 作为目标 domain. Idea 是重写经验风险 w.r.t. clean data, 可以得到

$$\beta(X, \bar{Y}) = p_D(\bar{Y} = i \mid X = x) / p_{\bar{D}}(\bar{Y} = i \mid X = x) \tag{9}$$

就是 IW. 这可以通过转移矩阵 T 或者使用小数据集的 clean data(like GLC) 来学到.

(Wang et al., 2017) 的 reweighted prob. models(RPM) 来应对 label noise. Idea 在于, 降低 bad labels 的权重且增加 clean labels 的权重. 具体地,

- 定义概率模型 $p_\beta(\beta) = \prod_{n=1}^{N} \ell\left(y_n \mid \beta\right)$
- 给出 latent weight 的先验分布 $p_w(w), w = (w_1, \ldots, w_N)$

$$p(y, \beta, w) = 1/z \cdot p_\beta(\beta) p_w(w) \prod_{n=1}^{N} \ell\left(y_n \mid \beta\right)^{w_n} \tag{10}$$

- 推理 β, w, 通过后验分布 $p(\beta, w|y)$. 先验分布 $p_w(w)$ 可以是 Beta 分

Objective Reweighting: Bayesian Methods

(Arazo et al., 2019) 使用了两组分 beta mixture model(BMM), 视为 clean-noisy 的混合, 使用了一个 bootstrapping loss. 具体地, 使用 dynamic weighted boostrapping loss. 数学上, 定义 loss 上的 pdf

$$p(\ell) = \sum_{k=1}^{K} \lambda_k p(\ell \mid k) \tag{11}$$

并且 $p(\ell \mid k)$ 可以使用 Beta 分布建模.

$$p(\ell \mid \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \ell^{\alpha-1}(1-\ell)^{\beta-1} \tag{12}$$

上述问题可以使用 EM 算法来解决.
更具体地, 引入 $\gamma_k(\ell) = p(k \mid \ell)$, E-step 中固定 λ, α, β, 计算 γ. M-step 中固定 γ, 使用带权动量估计 α, β, 动态权重则使用简单的方法来得到 $\lambda_k = \frac{1}{N}\sum_{i=1}^{N} \gamma_k(\ell_i)$. 基于这个 BMM 模型, 他们还提出了动态 hard/soft bootstrapping loss, 其中每个 sample 的 weight 动态的设置为 $p(k=1 \mid \ell_i)$(sample 为 clean 的概率).

Objective Reweighting: NNs

(Shu et al., 2019) 使用 Meta-Weight-Net(MW-Net) 来学习显示的 weighting function. w. func. 是一个单层 MLP, 从 loss 到 weight. 数学上

$$w^*(\theta) = \arg\min_w \ell^{\text{tr}}(w;\theta) = 1/N \sum_{i=1}^{N} \mathcal{V}\left(t_i^{\text{tr}}(w);\theta\right) \ell_i^{\text{tr}}(w) \tag{13}$$

这里, 可以使用**元学习**来优化 MW-Net: 给出一些 clean, balanced 元数据 $\left\{x_i^{(\text{meta})}, y_i^{(\text{meta})}\right\}_{i=1}^{M}$, 最小化 meta-loss

$$\theta^* = \arg\min_\theta \ell^{\text{meta}}\ (w^*(\theta)) = 1/M \sum_{i=1}^{M} \ell_i^{\text{meta}}\ (w^*(\theta))\ . \tag{14}$$

使用 SGD 迭代的分别更新 w 和 θ

Objective Redesigning: Loss Redesign

Remark 重设计目标函数往往是取决于目标场景的.

(Zhang et al., 2018) 提出了推广的 CE, 使用 MAE(mean absolute error/l1-norm)+CCE(categorical CE). 数学上, 他们使用 Box-Cox 变换作为 $\ell_q$loss func.

$$\ell_q\left(f(x), e_j\right) = \left(1 - f_j(x)^q\right)/q \tag{15}$$

注意 $q \to 0$ 为 CCE, $q \to 1$ 为 MAE. 并且引入了截断 ℓ_q 损失的估计

$$\ell_{\text{trunc}}\ \left(f(x), e_j\right) = \begin{cases} \ell_q(k) & \text{if } f_j(x) \le k, \\ \ell_q\left(f(x), e_j\right) & \text{otherwise} \end{cases} \tag{16}$$

并且 $\ell_q(k) = 1 - k^q/q$

(Charoenphakdee et al., 2019) 分析了对称 loss. Idea 是设计 loss 并不一定是对称的, 同时给不对称的区域的惩罚, i.e.,$\ell(z) + \ell(-z) = C$. 所以他们给出了一个 barrier hinge loss

$$\ell(z) = \max(-b(r+z) + r, \max(b(z-r), r-z)) \tag{17}$$

不对称, 并且在对称区外给出惩罚.

Objective Redesigning: Loss Redesign

(Thulasidasan et al., 2019) 提出了放弃一些 confusing data 的方法. 他们的网络 DAC(deep abstaining classi.) 有一个额外输出 p_{k+1}, 是放弃概率. 具体的 loss 是

$$\ell\left(x_j\right)=-\tilde{p}_{k+1}\sum_{i=1}^{k}t_i\log\left(p_i/\tilde{p}_{k+1}\right)-\alpha\log\tilde{p}_{k+1} \tag{18}$$

其中 $\tilde{p}_{k+1}=1-p_{k+1}$ 他们使用了一个动态的调整 α 的算法. DAC 可以作为 data cleaner 使用在不论有无结构的噪声上.
(Aditya et al., 2020) 使用梯度裁剪来设计 loss. 直觉上, 这防止了过于自信的递降. 基于梯度裁剪, 他们设计了部分 Huberized loss

$$\tilde{\ell}_\theta(x,y)=\begin{cases}-\tau p_\theta(x,y)+\log\tau+1 & \text{if } p_\theta(x,y)\leq\frac{1}{\tau}\\ -\log p_\theta(x,y) & \text{otherwise}\end{cases} \tag{19}$$

Objective Redesigning: Loss Redesign

(Lyu et al. 2020) 提出 curriculum loss. 是 0-1 loss 更紧的上界. 此外, CL 可以动态地选择 samples. 对于任何 base loss 是 0-1 loss 的上界 $\ell(u) \geq \mathbf{1}(u < 0), u \in \mathbb{R}$, CL 定义为

$$Q(\mathbf{u}) = \max\left(\min_{\mathbf{v}\in\{0,1\}^n} f_1(\mathbf{v}), \min_{\mathbf{v}\in\{0,1\}^n} f_2(\mathbf{v})\right) \tag{20}$$

其中

$$f_1(\mathbf{v}) = \sum_{i=1}^{n} v_i \ell\left(u_i\right) \text{ and } f_2(\mathbf{v}) = n - \sum_{i=1}^{n} v_i + \sum_{i=1}^{n} \mathbf{1}\left(u_i < 0\right). \tag{21}$$

为了在 DL 中使用 CL, 进一步他们提出了 noise pruned CL.

Label Ensemble

(Laine & Aila, 2017) 引入了 SSL 中的 self-ensembling, 包括 π-model 和时序聚合, 也可用于 noise 净化. self-ensembling 的 idea 是, 使用网络的输出对位置标签形成共识. 具体上, π-model 是让不同的 dropout 模式下对同一个输入得到相容的输出. 除了 π-model, 多个 epochs 的时序聚合也被考虑. π-model 的 loss 是

$$\ell = -1/B \sum_i \log z_i \left[y_i\right] + w(t)/C|B| \sum_i \left\|z_i - \tilde{z}_i\right\|^2 \tag{22}$$

其中第一项是 batch 的 CE, 后一项处理无标签数据. 其中 $z_i, \tilde{z}_i$ 分别是不同 dropout 下的输出. 第二项也被时间相关函数 $w(t)$ 加权.
这个方法的时序集成使用了几次之前的网络求值来进行一个集成预测, 具体上, 在 π-model 中的 $\tilde{z}$ 是前几次的输出的 moving average

$$Z_i \leftarrow \alpha Z_i + (1-\alpha) z_i \tag{23}$$

$$\tilde{z}_i = \frac{Z_i}{1-\alpha^t} \tag{24}$$

Label Ensemble

(Nguyen et al., 2020) 提出了 self-ensemble label filtering(SELF) 的方法来在训练中逐渐地去除错误 labels. 使用网络在不同训练迭代时的不同输出来得到一个预测的共识, 用于去除 bad labels. 假设 $L_i = \{(y, x) \mid \hat{y}_x = y; \forall (y, x) \in L_0\}$ 是 label y 和其最大似然预测, L_0 是最开始的样本集合.
根据他们的策略, 模型首先和 Mean Teacher 进行集成, 后者是一个 exponential running average of model(snapshots). 其次, 使用在多个 epochs 上的预测结果 $\bar{z}_j = \alpha \bar{\bar{z}}_{j-1} + (1 - \alpha)\hat{z}_j$, 代表了 moving average of predicted labels.

Remark 这和之前那个时序集成挺类似的. 区别是, 不是用不同 dropout 来进行集成而是和 mean teacher 集成.
并且 Dropout 和 kWTA 某种意义上挺相似的.

Utilizing Representation Space Metrics: LID

(Ma et al., 2018) 研究了训练样本上的深度表示子空间的维度. 提出了基于维度的训练策略, 在训练中检测子空间维度. Key idea 是使用 local intrinsic dim.(LID), 用于区分 noisy/clean labels. 数学上给出 LID 的一个估计

$$\widehat{\mathrm{LID}} = -\left(1/k \sum_{i=1}^{k} \log r_i(x)/r_{\max}(x)\right)^{-1} \tag{25}$$

其中 $r_i(x)$ 是 x 到 i-th neighbor 的距离. 根据他们的观察, 在 clean label 上训练的时候 LID 一直下降, 但是在 noisy label 上训练的时候先下降后上升. 可以据此观察训练动态.
值得注意的是, 对于 DNN 来说, 应该使用如下 in-batch 的 LID, 使用倒数第二层得到网络输出

$$\widehat{\mathrm{LID}}\left(x, X_B\right) = -\left(\frac{1}{k} \sum_{i=1}^{k} \log \frac{r_i\left(g(x), g\left(X_B\right)\right)}{r_{\max}\left(g(x), g\left(X_B\right)\right)}\right)^{-1} \tag{26}$$

Utilizing Representation Space Metrics: LID

据此, 提出 dimensionality-driven training, 具体地, 使用 LID 动态矫正 labels

$$y^* = \alpha_i y + (1 - \alpha_i)\, \widehat{y} \tag{27}$$

其中

$$\alpha_i = \exp\left(-\frac{i}{T} \frac{\widehat{\mathrm{LID}}_i}{\min_{j=0}^{i-1} \widehat{\mathrm{LID}}_j} \right) \tag{28}$$

校准因子是递增的, 代表随着训练进行, noisy label 变得越来越不可信. 可以直接使用 augmented 的标签的熵作为 loss

$$\mathcal{L} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{y_n^*} y_n^* \log P\left(y_n^* \mid x_n\right) \tag{29}$$

技术上, 设置 w 个 epochs 的初始化窗口. Turning point 是, LID 高于前 w 个 epochs 的 mean 两个标准差时. 此时 rollback.

Memorizaiton Effect

记忆效应指出, NNs 趋向于先记住容易学习的模式 (clean), 再逐渐 overfit 难以学习的模式 (noisy). 这启发了 small-loss trick. 具体上, 这是说把 small-loss 的样本作为 clean samples, 并且只在这些样本上进行模型参数更新.
数学上, 等价于建立一个 masked loss

$$\tilde{\ell} = \mathrm{sort}(\ell, 1 - \tau) \tag{30}$$

其中 τ 时 noise rate.

Self-training

(Jiang et al., 2018) 提出了 MentorNet, 监督称为 StudentNet 的基网络. 数学上, MentorNet g_m 估计一个 predefined curriculum

$$\arg\min_{\theta} \sum_{(x_i,y_i)\in\mathcal{D}} g_m\left(z_i;\theta\right)\ell_i + G\left(g_m\left(z_i;\theta\right);\lambda_1,\lambda_2\right), \tag{31}$$

第一项是课程加权的 loss, 第二项是正则化惩罚项. 可以得到闭形式解

$$g_m\left(z_i;\theta\right) = \begin{cases} \mathbf{1}\left(\ell_i \le \lambda_1\right) & \text{if } \lambda_2 = 0 \\ \min\left(\max\left(0, 1-\ell_i-\lambda_1/\lambda_2\right), 1\right) & \text{if } \lambda_2 \neq 0 \end{cases} \tag{32}$$

直觉上, 当 $\lambda_2 = 0$ 时, M-Net 只会使用 losses 小于 λ_1 的 sample. 否则 MentorNet 不会提供大于 $\lambda_1 + \lambda_2$ 的样本. 同时这也使 MNet 能够自行发现 curriculum.

Self-training

(Ren et al., 2018) 使用 meta-learning 来给不同的 samples 不同的权重, 基于他们的梯度方向. 基本上, 小 loss 数据分配更多权重, 并且他们相信在小的验证集上最小化 loss 的权重是好的权重, 具体上, 在每个 iteration 之前在验证集上决定样本权重. 数学上, 最小化带权损失

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{N} w_i \ell_i(\theta) \tag{33}$$

他们在验证集上进行参数选择

$$w^* = \arg\min_{w} 1/M \sum_{i=1}^{M} \ell_i^v \left(\theta^*(w)\right) \tag{34}$$

具体上有三步

1. 将 noisy data 进行前向/后向传播, 得到 traininig net. 的新的参数 θ 和梯度 $\nabla\theta$
2. 梯度 $\nabla\theta$ 影响 validation net.
3. training net. 使用 meta-leanrning 来更新 w, 通过二次求导/二阶导.

Co-teaching

(Han et al., 2018) 提出了 co-teaching, 让两个 DNN 在每个 batch 上互相 teach. 具体地, 每个网络前向传播所有数据, 然后选出 clean data, 再根据 peer net. 选出的 clean data 来进行参数更新. 在 MentorNet 中误差会不断累积, 但是在 co-teaching 中, 两个模型能够学到不同类型的 noise, 降低误差的累积, 最终达到一个共识. 注意集成学习中关键的一点是让两个学习器有差异.
(Yu et al., 2019) 提出了 co-teaching+, " 通过不同意来更新", 来保持 coteaching 中两个网络的的差异性. 包含 disaggrement-upd. + cross-upd. steps. 在 disaggr.-upd. 中两个网络分别预测, 然后只保留预测不同意的数据 (对称差?). cross-upd. 和 Co-teaching 一样. 两个网络对于 big-loss 有相同的 drop-rate.
(Chen et al., 2019) 使用 noisy dataset 上的随机分割作为 cross-validation. 意味着认为大多数样本是 correct labeled 的. 他们提出了 Iterative Noisy Cross-Validation(INCV), 来选择一个噪声很小的数据子集. 然后他们在这个子集上使用 Co-teaching 来训练, 并且逐步去除 big-loss 样本.

Co-teaching

(Yao et al., 2020) 使用 AutoML 来探索记忆效应. 注意 Co-teaching(+) 两个网络的 drop-rate 相同且都是人工选择的. 他们使用 Domain-specific search, 并且用一种牛顿法来估计最优的 drop-rate, 数学上可建模为 bi-level opt.

$$\begin{aligned} R^* &= \arg\min_{R(\cdot)\in\mathcal{F}} \mathcal{L}_{\text{val}}\left(f\left(w^*; R\right), \mathcal{D}_{\text{val}}\right) \\ \text{s.t. } w^* &= \arg\min_{w} \mathcal{L}_{\text{tr}}\left(f\left(w^*; R\right), \mathcal{D}_{\text{tr}}\right) \end{aligned} \tag{35}$$

(Li et al., 2020) 启发与 MixMatch, 提出了 DivideMix, 使用 SSL 技术. 它使用了 GMM 来动态地分割训练数据为 labeled clean data + unlabeled noisy data. 在 SSL 部分, 他们使用了 Co-training 的一个变体 Co-refinement 在有标签数据上, 并且在无标签数据上使用 Co-guessing. 具体上讲, Co-refinement 把 ground-truth 和网络预测结合, 得到 refined label; Co-guessing 把两个网络的预测输出平均. 使用 MixMatch 中的方法来合并数据.

Beyond Memorization

(Hendrycks et al., 2019) 展示了, pre-training 技术可以提高模型健壮性和不确定性, 包括对抗健壮性和标签噪声.
(Bahri et al., 2020) 提出了 Deep k-NN 方法, 作为 base DNN 的中间层来滤除噪声. 具体上由两步组成, 首先具有结构 $\mathcal{A}$ 的模型 $\mathcal{M}$ 通过 kNN 被训练来得到噪声数据 $\mathcal{D}_{noise}$, 通过去除和其邻居数据 label 不同的样本. 接着在 $\mathcal{D}_{\text{filtered}} \cup \mathcal{D}_{\text{clean}}$ 上重新训练结构 $\mathcal{A}$.

Future

Build Up New Datasets 从合成数据集 (MNIST, CIFAR) 到真实噪声数据集 (Clothing1M). 到更新更大的数据集 ("web-label noise", Jiang et al., 2020). 需要 NLP 的噪声数据集.

Instance-Dependent Noise 之前的所有方法都是基于噪声和 instance 无关的假设. 但是实际场景中这只是一个近似. 我们应该考虑 IDN 模型. 直觉上这是显然的, 因为模糊或者低质量的样本显然有更高概率被 mislabel. 但是和输入 feature 相关的 noise 显然大大提高了体系维度, 增加了复杂度. 并且如果没有额外的假设/信息, IDN 是不可区分的 (Cheng et al., 2020).

(Menon et al., 2018) 提出了一个边界相容的 IDN, 考虑在分类边界附近的样本具有更大的噪声. 但是这个方法仅限于二元分类, 并且无法估计噪声函数.

(Cheng et al., 2020) 研究了一种特殊的 IDN, 其中噪声函数是有上界的. 同样他们的方法也限于二分类, 并且只在小数据集上进行了测试.

(Berthon et al., 2020) 考虑了使用每个 instance 的置信度: confidence-scored IDN. 基于此, 他们提出了 instance-level 的前向矫正.

Future: Adversarial LNRL

(Wang et al., 2020) 提出了新的 defense algorithm MART, 将错误/正确分类的样本分别微分. 具体上讲, 使用了 Misclassification-Aware-Regularization

$$1/n \sum_{i=1}^{n} \mathbf{1}\left(h_\theta\left(x_i\right) \neq h_\theta\left(\hat{x}_i'\right)\right) \cdot \mathbf{1}\left(h_\theta\left(x_i\right) \neq y_i\right) \tag{36}$$

其中 $\hat{x}_i'$ 是某种对抗样本[1]

同时 (Zhang et al., 2020) 也考虑了同样的问题: 对抗训练中的错分类问题. 提出了 friendly adversarial training(FAT), 在训练 DNN 时使用错分类对抗样本最小化 loss, 使用正确分类对抗样本最大化 loss. 使用了 early-stopped PGD, 一旦对抗样本被错分类就停止 PGD.

[1]详见笔记

Beyond Labels: Noisy Data

1. **Feature**: 对抗样本是一种特殊的 feature noise. 可以表示为 $p(\bar{X} \mid Y)$. 对抗训练显然是一种主要办法. 注意随机扰动是另一种噪声. (Zhang et al., 2019) 提出了一种 robust ResNet, 启发于动力系统. 使用显示 Euler 法的 step factor 来研究问题, 他们证明了, 小 step factor 有利于泛化健壮性.
2. **Preference**: ranking 中的 preference 噪声问题, 常见于各种推荐系统/在线排序/评分系统. (Han et al., 2018) 提出了 POPAL 模型, 使用集成了一个 denoising vector 的 Plackett-Luce 模型. 基于 Kendall-tau 距离, 这个向量用确定的概率校正 k-ary noisy preferences. 然而 POPAL 不能处理 k-ary noisy preferences 的动态长度, 于是 (Pan et al., 2018) 提出了 COUPLE, 使用 stagewise training. 使用了 online Bayesian inference 来优化两个模型.

Beyond Labels: Noisy Data

1. **Domain**: Domain Adaptation(DA) 是一个 ML 的基本问题. 传统 DA 认为 source domain 的有标签数据是 clean 的, 但不一定为真. 当 source domain 的 label 是 noisy 的, 称为 wild domain adaptation(WDA). (Liu et al., 2019) 提出了 Butterfly 框架, 同时维护四个 DNNs, 可以迭代地获得 DIR(domain-specific repr.), 和 TSR(target-specific repr.). (Yu et al., 2020) 提出了 Denoising Conditional Invariant Component(DCIC) 框架, 可证明保证能提取不变表示, 并且无偏地估计 target domain 的数据分布.
2. **Similarity**: 基于相似度的学习是 weak-supervised learning 的问题, 可用的是无标签数据和相似数据 (挺像 Graph ULL).(Wu et al., 2020) 使用一个噪声转移模型来建模相似性噪声.

Beyond Labels: Noisy Data

1. **Graph**: 图结构是否对噪声 (node/edge feature/label, noisy links) 健壮? (Wang et al., 2020) 提出了健壮和无监督的框架 Cross-Graph, 可以处理图中的结构损坏. (Hu et al., 2019) 是 pre-training GNNs, 可能能够提高健壮性.
2. **Demonstration**: 模仿学习 (Imitation Learning, IL) 的目标时从高质量演示中学到好的策略. 当演示的质量不齐时, 显然导致了 diverse-quality demos. 可以使用置信度/ranking scores/小部分高质量 demos 来估计 noise, 来轻松的学习. 但是没有已有 expert data 时不一定能 work. (Tangkaratt et al., 2020) 提出了一个模型, 使用概率图模型来 VILD 建模 demo 质量. 具体上, 他们通过奖励函数 (代表了某种专家决策) 来估计质量, 使用了变分法来处理大的 s-a 空间, 使用 importance sampling.

Graph & Label Noise

Few existing literature. (Hoang et al., 2019) 展示了 noisy label 下各种网络的性能下降. 其他一些文献大多关于使用 GNN 来 clean label noise.

Idea

- 图上天然的正则化 $\boldsymbol{x}^T \mathcal{L} \boldsymbol{x}$. 同时图上类似 LID 的类似物 (曲率, Hessian, Dirac Operator)
- 图上相近的 node 会有更高的混淆概率 (变成 neighbor 的 label). 是否可能这么估计一个 noisy transition mat.
- 各种 pre-training 的方法 (e.g., GCC) + 使用类似 LID 的指标来监控 fine-tune 的 early-stopping/rollback.
- 类似 co-teaching, boost 多个模型?