模式识别与神经计算研究组
PAttern Recognition and NEural Computing

Detecting Corrupted Labels Without Training a Model to Predict

Zhaowei Zhu Zihao Dong Yang Liu

ICML 2022

Background

- Label noise in real-world datasets encodes wrong correlation patterns and impairs the generalization of deep neural networks (DNNs).
- It is critical to find efficient ways to detect corrupted patterns.

- Learning-centric
    - prediction & compare

- Data-Centric
    - neighbor information

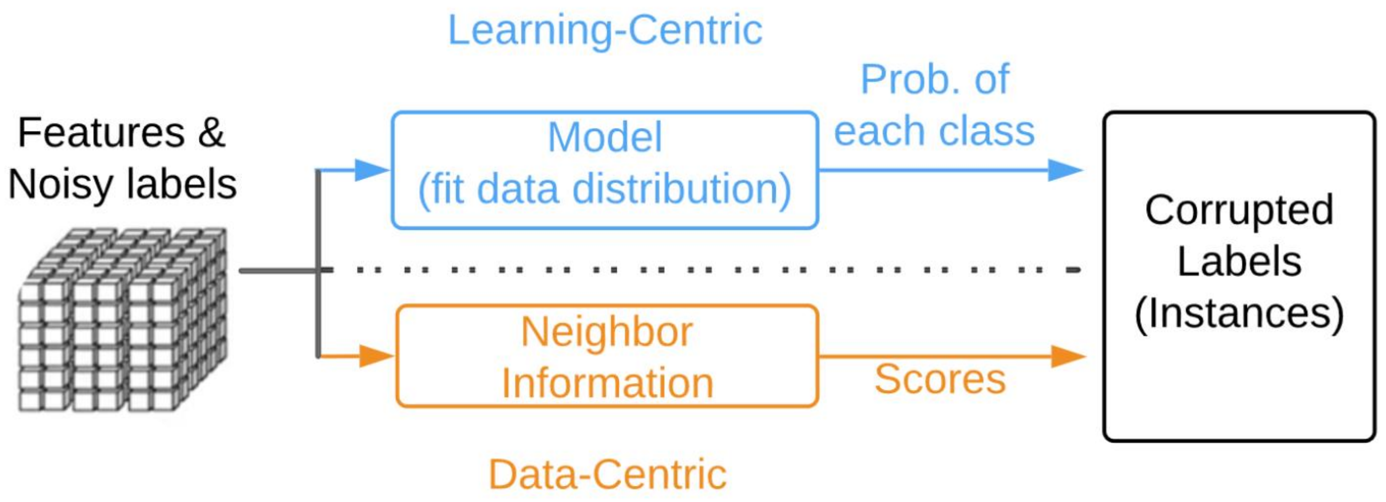

Figure 1. The existing learning-centric pipeline vs. our proposed data-centric pipeline. The inputs are features and the corresponding noisy labels, and the outputs are a set of corrupted labels. Blue: The learning-centric solution. Orange: The data-centric solution.

Motivation

- Previous learning-centric methods
  - train DNNs with noisy supervisions
  - design robust loss functions

- Limitations of the learning-centric methods
  - task-specific and fine-tuning hyperparameters for different datasets/noise
  - as long as the model is trained with noisy supervisions, the memorization of corrupted instances exists.

  The model will “subjectively” and wrongly treat the memorized/overfitted corrupted instances as clean.

- Solution
  - drop the dependency on the noisy supervision
  - design a training-free method to find label errors.

Method

Definition 2.1 (((k, δ_k) label clusterability). A dataset D satisfies (k, δ_k) label clusterability if: $\forall n \in [N]$, the feature x_n and its k-Nearest-Neighbors (k-NN) $x_{n_1}, \cdots, x_{n_k}$ belong to the same true class with probability at least $1 - \delta_k$.

- a probabilistic Y given X
- the quality of features and the value of k

- The (k, 0) label clusterability is also known as k-NN label clusterability

Method

- share the same noise transition matrix

$$\mathbb{P}\left(\widetilde{Y}=\tilde{y}_n \mid X=x_n, Y=y_n\right)=\mathbb{P}\left(\widetilde{Y}=\tilde{y}_n \mid X=x_{n'}, Y=y_n\right), \forall x_{n'} \in \{x_{n_1}, \cdots, x_{n_k}\}$$

- using appropriate features may be better than model logits/predictions when the dataset is noisy.

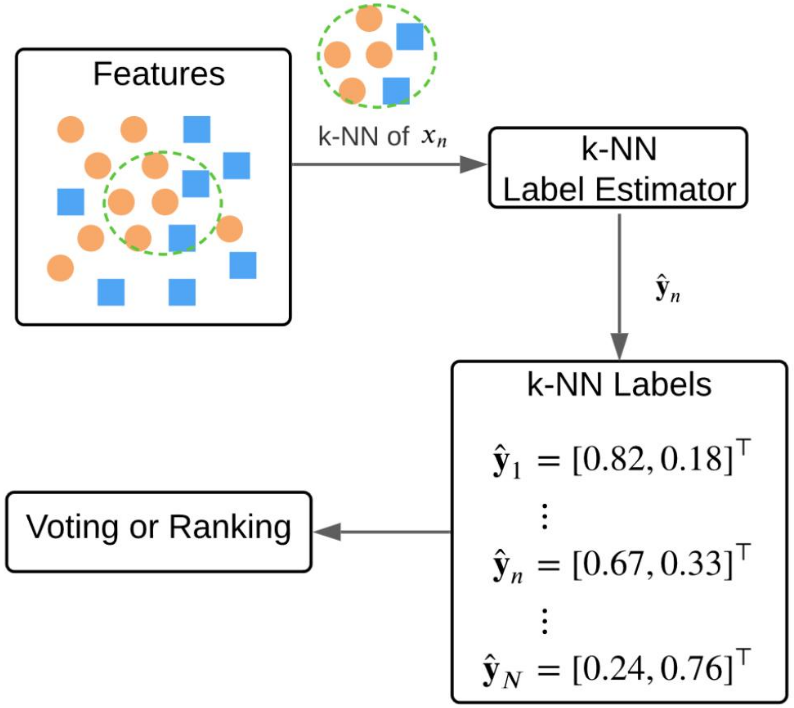

Figure 2. Detect corrupted labels with similar features. Orange circle: instance with noisy label 1. Blue square: instance with noisy label 2. Green dashed circle: A k-NN example.

Method

- The F1 score on corrupted instances is sensitive to the case when the noise rate is mild to low, which is typically the case in practice.

$$F_1 = 2/(\texttt{Precision}^{-1} + \texttt{Recall}^{-1}).$$

$$\texttt{Precision} = \frac{\sum_{n\in[N]} \mathbb{1}(v_n = 1, \tilde{y}_n \neq y_n)}{\sum_{n\in[N]} \mathbb{1}(v_n = 1)},$$

$$\texttt{Recall} = \frac{\sum_{n\in[N]} \mathbb{1}(v_n = 1, \tilde{y}_n \neq y_n)}{\sum_{n\in[N]} \mathbb{1}(\tilde{y}_n \neq y_n)}.$$

Method

- Voting-Based Local Detection

$$y_n^{\text{vote}} = \arg\max_{i\in[K]} \hat{\boldsymbol{y}}_n[i]$$
$$v_n := \mathbf{1}\left(y_n^{\text{vote}} \neq \tilde{y}_n\right)$$

Property 3.1 (Relative score). Within the same instance, the score of the majority class is higher than the others, i.e., $\forall j \neq y_n^{\mathtt{vote}}, j \in [K], \forall n \in [N]$: $\mathsf{Score}(\hat{\boldsymbol{y}}_n, y_n^{\mathtt{vote}}) > \mathsf{Score}(\hat{\boldsymbol{y}}_n, j)$.

Property 3.2 (Absolute score). $\mathsf{Score}(\hat{\boldsymbol{y}}_n, j)$ is jointly determined by both $\hat{\boldsymbol{y}}_n[j]$ and $\hat{\boldsymbol{y}}_n[j'], \forall j' \neq j$.

- Ranking-Based Global Detection
  - scoring function
  - threshold

$$\mathsf{Score}(\hat{\boldsymbol{y}}_n, j) = \frac{\hat{\boldsymbol{y}}_n^\top \boldsymbol{e}_j}{\|\hat{\boldsymbol{y}}_n\|_2 \|\boldsymbol{e}_j\|_2},$$

$$\mathcal{I} = \mathtt{argsort}\{\mathsf{Score}(\hat{\boldsymbol{y}}_n, j)\}_{n\in\mathcal{N}_j},$$

$$v_n = \mathbb{1}(\mathtt{Loc}(n, \mathcal{I}) \leq \widetilde{N}_j),$$

Method

$$\widetilde{N}_j = \mathbb{P}(Y \neq j \mid \widetilde{Y} = j) \cdot N_j$$

$$\mathbb{P}(Y \neq j|\widetilde{Y} = j) = 1 - \mathbb{P}(Y = j|\widetilde{Y} = j),$$

- Borrow the results from the HOC , where the noise transition probability and the marginal distribution of clean label can be estimated with only features and the corresponding noisy labels.

- Bayes' rule:

$$\mathbb{P}(Y = j|\widetilde{Y} = j) = \mathbb{P}(\widetilde{Y} = j|Y = j) \cdot \mathbb{P}(Y = j)/\mathbb{P}(\widetilde{Y} = j),$$

Method

Algorithm 1 Detection with **Simi**lar **Feat**ures (The SimiFeat Detector)

Input: Number of epochs: M. k-NN parameter: k. Noisy dataset: $\widetilde{D} = \{(x_n, \tilde{y}_n)\}_{n\in[N]}$. Feature extractor: $g(\cdot)$. Method: *Vote* or *Rank*. Epoch counter $m = 0$.

repeat

  $x'_n \leftarrow \texttt{RandPreProcess}(x_n), \forall n;$ *# Initialize & Standard data augmentations*

  $x_n \leftarrow g(x'_n), \forall n;$ *# For tasks with rarely clusterable features, extract features with $g(\cdot)$*

  $\hat{\boldsymbol{y}}_n \leftarrow \texttt{kNNLabel}(\{x_n\}_{n\in[N]}, k)$ *# Get soft labels. One can weight instances by the similarity to the center instance.*

  if *Vote* **then**

    $y_n^{\texttt{vote}} \leftarrow \arg\max_{i\in[K]} \hat{y}_n[i];$ *# Apply local majority vote*

    $v_n \leftarrow \mathbb{1}(y_n^{\texttt{vote}} \neq \tilde{y}_n), \forall n \in [N];$ *# Treat as corrupted if majority votes disagree with noisy labels*

  else

    $\mathbb{P}(Y), \mathbb{P}(\widetilde{Y}|Y) \leftarrow \texttt{HOC}(\{(x_n, \tilde{y}_n)\}_{n\in[N]});$ *# Estimate clean priors $\mathbb{P}(Y)$ and noise transitions $\mathbb{P}(\widetilde{Y}|Y)$ by HOC*

    $\mathbb{P}(Y|\widetilde{Y}) = \mathbb{P}(\widetilde{Y}|Y) \cdot \mathbb{P}(Y)/\mathbb{P}(\widetilde{Y});$ *# Estimate thresholds by Bayes' rule*

    for j **in** $[K]$ **do**

      $\mathcal{N}_j := \{n|\tilde{y}_n = j\};$ *# Detect corrupted labels in each set $\mathcal{N}_j$*

      $\mathcal{I} \leftarrow \texttt{argsort}\{\mathsf{Score}(\hat{\boldsymbol{y}}_n, j)\}_{n\in\mathcal{N}_j};$ *# $\mathcal{I}$ records the raw index of each sorted value*

      $v_n \leftarrow \mathbb{1}\big(\texttt{Loc}(n, \mathcal{I}) \leq \lfloor(1 - \mathbb{P}(Y = j|\widetilde{Y} = j)) \cdot N_j\rfloor\big);$ *# Select low-score (head) instances as corrupted ones*

    end for

  end if

  $\mathcal{V}_m = \{v_n\}_{n\in[N]};$ *# Record detection results in the m-th epoch*

until M times

$\mathcal{V} = \texttt{Vote}(\mathcal{V}_m, \forall m \in [M]);$ *# Do majority vote based on results from M epochs*

Output: $[N] \setminus \mathcal{V}$.

Experiments

南京航空航天大學 NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

- Fitting Noisy Distributions May Not Be Necessary

Table 1. Comparisons of F_1-scores (%). CORES, CL, TracIn: Train with noisy supervisions. SimiFeat-V and SimiFeat-R: Get $g(\cdot)$ **without** any supervision. Top 2 are **bold**.

METHOD	CIFAR10				CIFAR100			
	Human	*Symm. 0.6*	*Asym. 0.3*	*Inst. 0.4*	*Human*	*Symm. 0.6*	*Asym. 0.3*	*Inst. 0.4*
CORES	65.00	92.94	7.68	**87.43**	3.52	**92.34**	0.02	9.67
CL	55.85	80.59	76.45	62.89	64.58	78.98	52.96	50.08
TRACIN	55.02	76.94	73.47	58.85	61.75	76.74	48.42	49.89
DEEP k-NN	56.21	82.35	75.24	63.08	57.40	70.69	56.75	63.85
SIMIFEAT-V	**82.30**	**93.21**	**82.52**	81.09	**73.19**	84.48	**65.42**	**74.26**
SIMIFEAT-R	**83.28**	**95.56**	**83.58**	**82.26**	**74.67**	**88.68**	**62.89**	**73.53**

- Features May Be Better Than Model Predictions

Table 2. Comparisons of F_1-scores (%). CORES, CL, TracIn: Use logit layers. SimiFeat-V/R: Use only representations. **All methods use the same fixed extractor from CLIP.** Top 2 are **bold**.

METHOD	CIFAR10				CIFAR100			
	Human	*Symm. 0.6*	*Asym. 0.3*	*Inst. 0.4*	*Human*	*Symm. 0.6*	*Asym. 0.3*	*Inst. 0.4*
CE SIEVE	67.21	94.56	5.24	8.41	16.24	88.55	2.6	1.63
CORES	83.18	**96.94**	12.05	**88.89**	38.52	**92.33**	7.02	**85.52**
CL	69.76	95.03	77.14	62.91	67.64	85.67	62.58	61.53
TRACIN	81.85	95.96	80.75	64.97	**79.32**	**91.03**	63.12	64.31
DEEP k-NN	82.98	87.47	76.96	77.42	72.33	82.95	64.96	74.25
SIMIFEAT-V	**87.43**	96.44	**88.97**	87.11	76.26	86.88	**73.50**	**80.03**
SIMIFEAT-R	**87.45**	**96.74**	**89.04**	**91.14**	**79.21**	90.54	**68.14**	77.37

Experiments

- The Effect of the Quality of Features

Table 3. Comparisons of F_1-scores (%) using $g(\cdot)$ with different δ_k (%). Model names are the same as Figure 3.

PRE-TRAINED MODEL	CIFAR10			CIFAR100		
	$1-\delta_k$	*Human*	*Inst. 0.4*	$1-\delta_k$	*Human*	*Inst. 0.4*
R18-IMG	35.73	75.40	80.22	11.30	74.91	71.99
R34-IMG	48.13	79.52	82.43	16.17	76.88	74.00
R50-IMG	45.77	78.40	82.06	15.81	76.55	73.51
VIT-B/32-CLIP	64.12	87.45	91.14	19.94	79.21	77.37
R34-C10-SSL	69.31	83.28	85.26	2.59	68.03	65.94
R34-C10-CLEAN	99.41	98.39	98.59	0.22	60.90	60.73
R34-C100-SSL	18.59	59.96	74.99	22.46	74.67	73.53
R34-C100-CLEAN	18.58	60.17	76.41	89.07	92.87	95.29

Table 4. Experiments on Clothing1M. None: Standard training with 1M noisy data. R50-Img (or ViT-B/32-CLIP, R50-Img Warmup-1): Apply our method with ResNet50 pre-trained on ImageNet (or ViT-B/32 pre-trained by CLIP, R50-Img with 1-epoch warmup). The clean test accuracy on the best epoch, the last 10 epochs, and the last epoch, are listed. Top-1 is **bold**.

DATA SELECTION	# TRAINING	BEST EPOCH	LAST 10	LAST
NONE	1M (100%)	70.32	69.44 ± 0.13	69.53
R50-IMG	770K (77.0%)	72.37	71.95 ± 0.08	71.89
VIT-B/32-CLIP	700K (70.0%)	72.54	72.23 ± 0.17	72.11
R50-IMG WARMUP-1	767K (76.7%)	**73.64**	$\mathbf{73.28 \pm 0.18}$	**73.41**

Thanks