Handling mislabeled training data for classification

Shubham, Siddharth

What is mislabeled data

- Data for supervised learning consists of (x1, x2, x3, … y)
- Some output labels y are incorrect.
- Example: Cat classification

Reasons for mislabeling

- Subjectivity - Information for labeling different from data attributes.
- Data-entry error
- Inadequate information - Hard to perform tests to guarantee 100% diagnosis

Methods for Handling Mislabeling

- Noise Elimination (Filtering data)
- Noise Tolerance (Robust algorithms, handling overfitting)

We will focus on Noise Elimination

- Analyze and include outliers as exceptions.
- Noisy examples do not influence hypothesis construction.

Gamberger D, Lavrac N, Dzeroski S (2000) Noise detection and elimination in data proprocessing: Experiments in medical do- mains. Appl Artif Intell 14(2):205–223

Ideas from the following papers

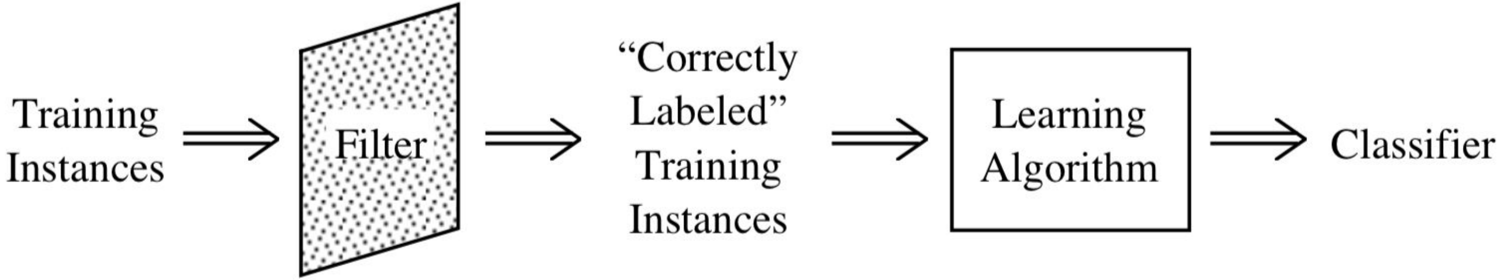
- C. E Brodley and M. A. Friedl (1999) "Identifying Mislabeled Training Data"
- CG Northcutt, T Wu, IL Chuang (2017) “Learning with Confident Examples: Rank Pruning for Robust Classification with Noisy Labels”

Motivation

- Removing outliers in regression analysis.
- An outlier is a case (an instance) that does not follow the same model as the rest of the data and appears as though it comes from a different probability distribution.

Main idea

- Using classifiers as filters.



How to filter

- Mark every instance in the training set as mislabeled (1) or not (0).
- Filter out the mislabeled instances.

Assumption:

- Errors are independent of model being fit.

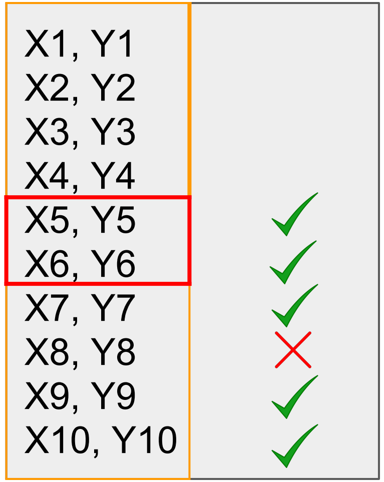
Filtering by Cross-Validation

- Divide training data into n folds
- Train a “filtering model” on (n-1) folds, and add a ‘mislabeled’ class attribute to the examples in the nth fold.
- Repeat for all possible folds.

Filtering Example

Filtering Example

Filtering Example



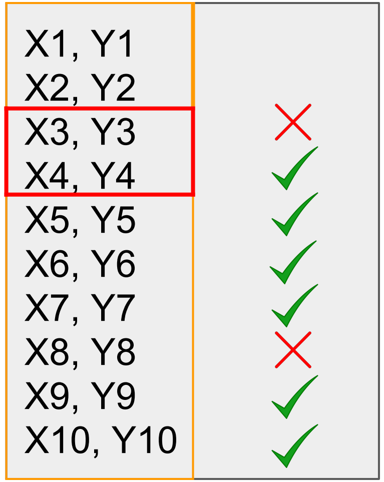
Test Part

Training Part

Correctly Labeled

Mislabeled

Filtering Example



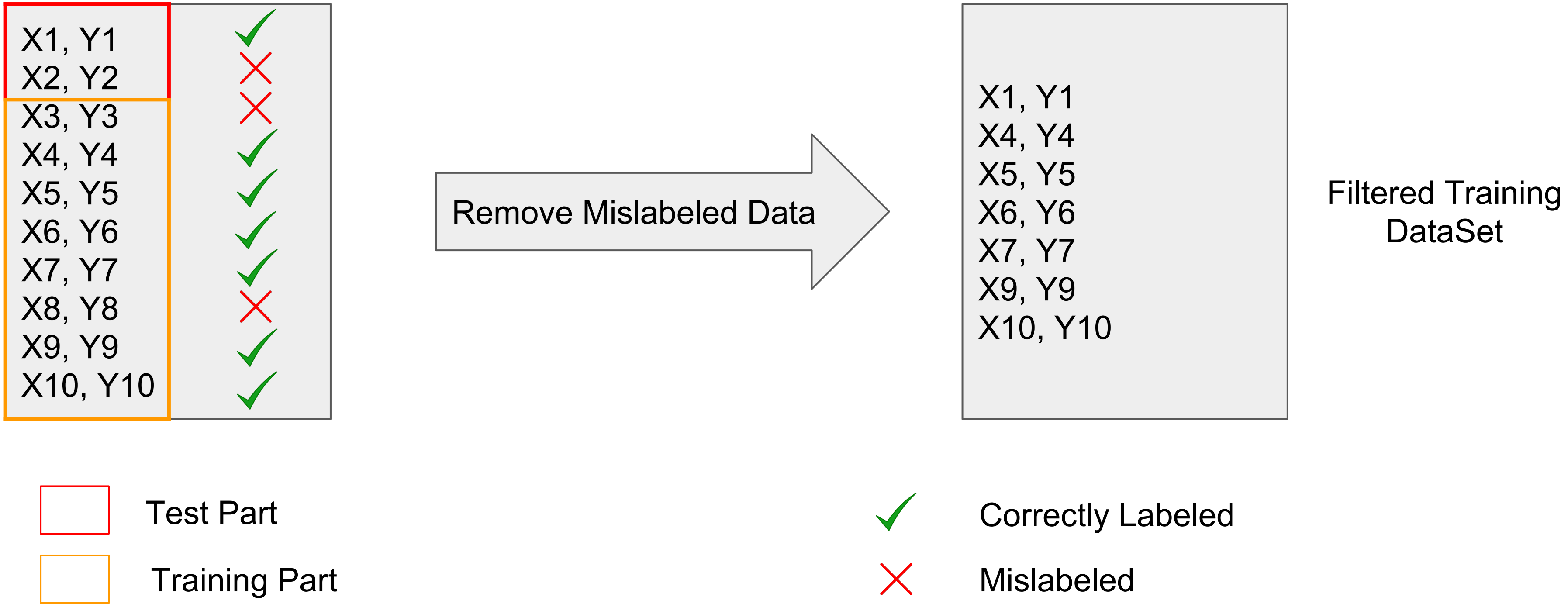
Test Part

Training Part

Correctly Labeled

Mislabeled

Filtering Example



Types of Filtering

- Single Algorithm Filter
  - Filtering is done by one algorithm
  - Instance is marked as mislabeled if this algorithm tagged it as mislabeled
- Majority Vote Filter
  - Filtering is done by multiple algorithms
  - Instance is marked as mislabeled if more than half of the algorithms tagged it as mislabeled
- Consensus Filter
  - Filtering is done by multiple algorithms
  - Instance is marked as mislabeled if all of the algorithms tagged it as mislabeled

Types of Detection Errors

- ❖ E1 - correct instance is tagged as mislabeled and subsequently discarded
- ❖ E2 - mislabeled instance is tagged as correctly labeled

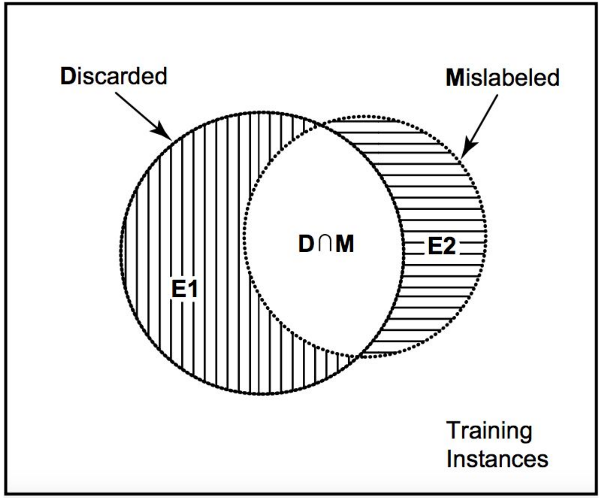


Figure: Types of Detection Errors

Probability of each error

1. Majority Filter

$$P(E1) = \sum_{j>m/2}^{j=m} P(E1_i)^j (1 - P(E1_i))^{m-j} \begin{pmatrix} m \\ j \end{pmatrix}$$

$$P(E2) = \sum_{j>m/2}^{j=m} P(E2_i)^j (1 - P(E2_i))^{m-j} \begin{pmatrix} m \\ j \end{pmatrix}$$

Here,

$P(E1_i)$ = Probability that classifier i makes error E1
$P(E2_i)$ = Probability that classifier i makes error E2
m = number of base level classifiers

Probability of each error

2. Consensus Filter

$$P(E1) = \prod_{i=1}^{m} P(E1_i)$$

$$P(E2) = 1 - \prod_{i=1}^{m} (1 - P(E2_i))$$

Here,

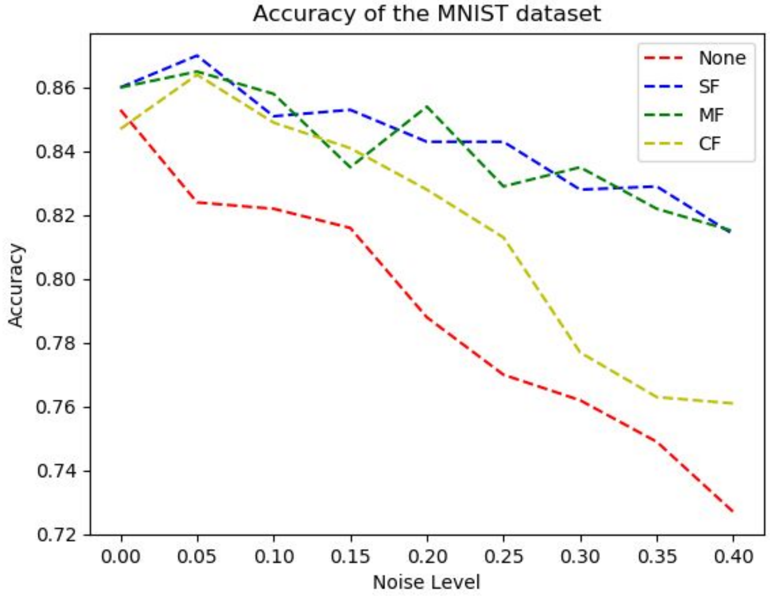
$P(E1_i)$ = Probability that classifier i makes error E1
$P(E2_i)$ = Probability that classifier i makes error E2
m = number of base level classifiers

Empirical analysis

- MNIST Dataset
  - Training dataset = 10000 images
  - Test dataset = 1000 images
- Model used for Filtering
  - Single Algorithm Filter(SF) = Logistic Regression
  - Majority Filter(MF) = Logistic Regression, Random Forest Classifier, MLP Classifier
  - Consensus Filter(CF) = Logistic Regression, Random Forest Classifier, MLP Classifier
- Final Classifier Model = Logistic Regression
- Noise Level Used = [0%, 5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%]

Empirical analysis

❖ Comparison of different types of filters with increasing noise in training data



Empirical analysis

Noise Level	Single Filter $P(E_1)$	$P(E_2)$	Majority Filter $P(E_1)$	$P(E_2)$	Consensus Filter $P(E_1)$	$P(E_2)$
5	0.17	0.10	0.20	0.09	0.06	0.14
10	0.18	0.10	0.20	0.08	0.07	0.14
15	0.20	0.09	0.22	0.08	0.08	0.14
20	0.21	0.10	0.24	0.08	0.08	0.15
25	0.22	0.10	0.27	0.07	0.09	0.16
30	0.24	0.10	0.27	0.07	0.10	0.17
35	0.27	0.10	0.30	0.08	0.10	0.17
40	0.30	0.10	0.36	0.07	0.11	0.17

Rankpruning

- Paper published in (UAI) 2017.
- Approach for solving $\tilde{P}\tilde{N}$ learning problem
- RP can estimate the noise rates.

http://auai.org/uai2017/proceedings/papers/35.pdf

Formulating $\tilde{P}\tilde{N}$ learning

- Given n observed training examples $x \in \mathcal{R}^D$

Observed corrupted labels: $s \in \{0, 1\}$ Unobserved true labels: $y \in \{0, 1\}$

Unfortunately, using (x, s) pairs, we estimate *g*, $x \rightarrow s$

$$g(x) = P(\hat{s} = 1|x)$$

Observed noisy positive and negative sets

$$\tilde{P} = \{x|s = 1\}, \tilde{N} = \{x|s = 0\}$$

We want to estimate $f \; x \rightarrow y$

Main Idea

- Prune the observed (x, s) pairs to obtain confident (x, s) pairs that are close to Unobserved $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$

VAR	CONDITIONAL	DESCRIPTION
ρ_0	$P(s=1 \mid y=0)$	Fraction of N examples mislabeled as positive
ρ_1	$P(s=0 \mid y=1)$	Fraction of P examples mislabeled as negative
π_0	$P(y=1 \mid s=0)$	Fraction of mislabeled examples in $\tilde{N}$
π_1	$P(y=0 \mid s=1)$	Fraction of mislabeled examples in $\tilde{P}$

$$\rho_1 + \rho_0 < 1$$

$$p_{s1} = P(s=1) \qquad \pi_1 = P(y=0 \mid s=1) = \frac{\rho_0(1-p_{y1})}{p_{s1}}$$

$$p_{y1} = P(y=1) \qquad \pi_0 = P(y=1 \mid s=0) = \frac{\rho_1 p_{y1}}{(1-p_{s1})}$$

Estimating thresholds for pruning

$$\hat{\rho}_1^{conf} := \frac{|\tilde{N}_{y=1}|}{|\tilde{N}_{y=1}| + |\tilde{P}_{y=1}|}, \hat{\rho}_0^{conf} := \frac{|\tilde{P}_{y=0}|}{|\tilde{P}_{y=0}| + |\tilde{N}_{y=0}|}$$

$$\begin{cases} \tilde{P}_{y=1} = \{x \in \tilde{P} \mid g(x) \geq LB_{y=1}\} \\ \tilde{N}_{y=1} = \{x \in \tilde{N} \mid g(x) \geq LB_{y=1}\} \\ \tilde{P}_{y=0} = \{x \in \tilde{P} \mid g(x) \leq UB_{y=0}\} \\ \tilde{N}_{y=0} = \{x \in \tilde{N} \mid g(x) \leq UB_{y=0}\} \end{cases}$$

$$\begin{cases} LB_{y=1} := P(\hat{s} = 1 \mid s = 1) = E_{x \in \tilde{P}}[g(x)] \\ UB_{y=0} := P(\hat{s} = 1 \mid s = 0) = E_{x \in \tilde{N}}[g(x)] \end{cases}$$

Pruned training data

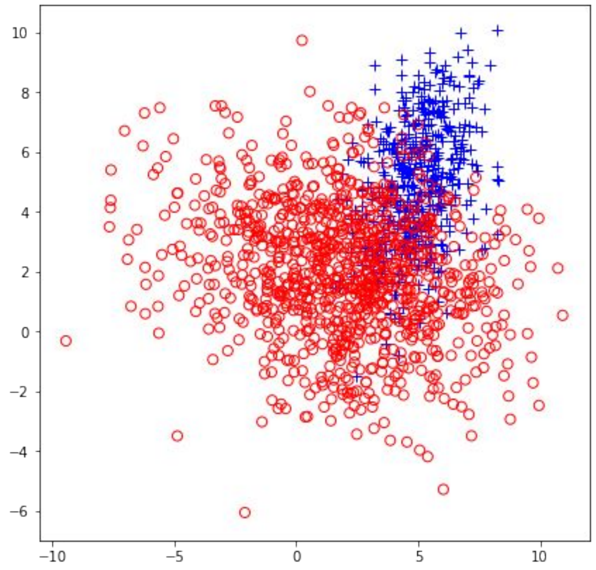
- $\tilde{P}_{conf}$ = {remove $\hat{\pi}_1|\tilde{P}|$ examples from $\tilde{P}$ with least $g(x)$ }
- $\tilde{N}_{conf}$ = {remove $\hat{\pi}_0|\tilde{N}|$ examples from $\tilde{N}$ with highest $g(x)$}
- Fit classifier on $X_{conf} = \tilde{P}_{conf} \cup \tilde{N}_{conf}$

  (Perform class-conditional reweighting of loss function if required)

Results - Accuracy Comparison, N = 1500 (+500, -1000)

multivariate_normal(mean=[5,5], cov=[[1.5,0.3],[1.3,4]], size=500)
multivariate_normal(mean=[2,2], cov=[[10,-1.5],[-1.5,5]], size=1000)

Noise Rates (rho0, rho1)	Baseline (LR)	Rank Pruning	Rank Pruning (Noise rates given)
0, 0	0.845	0.844	0.845
0.2, 0.6	0.666	0.827	0.832
0.4, 0.4	0.828	0.797	0.840
0.6, 0.2	0.338	0.778	0.834



Ongoing work

- Build a simple python wrapper that supports the filtering techniques we’ve analyzed.

Thank you! Questions?