Reliable and fair machine learning for risk assessment

Pereira Barata, A.P.

Chapter 4

Noise-Resilient Classifier

Noise in data is a pervasive concern, with causes ranging from human entry errors to flawed automated detection tools. When used to learn a classifier, noisy samples —where class labels and feature values may be corrupted— can seriously deteriorate the resulting classifier performance. In this chapter we propose DENOISE, a unified method to perform classifier learning from noisy samples that leverages noise detection and sample weighting techniques.

The proposed approach consists of learning a noise-resilient classifier through a log-odds sample weighting strategy, in which the weights are derived from the noisy instances in a label noise detection step, as described in Chapter 3. We report on the performance of our method in a controlled scenario where noise was artificially injected into a diverse set of datasets.

Different parameterised configurations of label noise and feature noise proportions were extensively tested against current state-of-the-art methods from the fields of classifier learning under noisy conditions and label noise detection. Results over ten datasets show that overall our method outperforms the state-of-the-art with respect to both learning from noisy data and noise detection, further validating the approach set out in Chapter 3 (which answered RQ2(a)), and most importantly providing an answer to RQ2(b): given data with label noise, how can noisy-samples be used to learn a well-performing model?

The current chapter corresponds to the following publication:
Pereira Barata, A., Takes, F. W., van den Herik, H. J., and Veenman, C. J. (2022). Noise-resilient classifier learning. *Pattern Recognition (under review)*

4.1 Noise and Performance Degradation

Noisy data are a prevalent issue in many data-reliant domains. Random input errors, or even malicious intent, cause problematic hurdles for any data descriptive or predictive task. In this chapter, we consider a supervised classification scenario in which we distinguish between feature noise (see Definition 1.16) and label noise (see Definition 1.17).

Samples within a classification dataset may have noise in either (1) the feature values; (2) the class labels; or (3) a combination of both [Wu, 1995]. A major problem of noisy data towards classifier learning is the severe performance degradation of the learned classifier [Heskes, 1994,Wilson and Martinez, 2000,Zhu and Wu, 2004]. A second issue may come from the complexity of the trained model. The impact of the complexity can be assessed by the extent to which a classifier is explainable [Brodley and Friedl, 1999,Segata et al., 2010]. The two problems may present serious consequences such as leading to inaccurate medical diagnoses [Zhang et al., 2006,Holzinger et al., 2017]. targeting the noisy samples is an endeavour in itself. Yet, an adequate detection of such samples can be used to improve classifier learning with noisy data [Gamberger et al., 1999]. In the case where feature noise is assumed, three different approaches have been proposed. First, if feature noise comes in the form of missing values, imputation and missing-indicator methods can be deployed. They allow for adequate learning [Pereira Barata et al., 2019]. Second, if noise relates to the existing values, a viable solution is to apply standard outlier detection methods. As such, samples with outlying values are targeted and consequently discarded prior to learning a classifier [Li et al., 2015]. Third, the usage of learners which are robust to samples with noisy features has been investigated with positive results [Sáez et al., 2014].

Most literature focuses on label noise [Frénay and Verleysen, 2013], as it is typically more detrimental to classifier learning than feature noise. Two reasons are: (1) there is a greater number of features than the single category label, and (2) not all features are equally important towards learning a classifier, whereas the class label of a sample is always of paramount importance [Sáez et al., 2014]. In [Frénay and Verleysen, 2013], the authors categorise label noise into three types or mechanisms:

1. NCAR, the proportion of mislabelled samples is the same per class, and thus independent of features and class;
2. NAR, the proportion of mislabelling is dependent on class, and independent of features;
3. NNAR, the proportion of mislabelled samples may or may not be the same per class and is dependent upon the features.

Joint class and feature noise (i.e., the NNAR case) is often attributed to class overlap. Samples of different classes which are similarly feature-valued increase class-separability issues [Beigman Klebanov and Beigman, 2009]. From a different perspective, a NNAR mechanism might be a result of malicious data manipulation [Pereira Barata et al., 2018b]. As an example, let us consider the practical scenario of international transportation of waste. Transit of wasteful materials is highly regulated through a system of permits. Waste transportation costs vary considerably depending on the waste category of a permit (i.e., class). Therefore, companies have a financial incentive to allege transporting waste with erroneous categorisation (i.e., mislabelling). To further mask the manipulated label, other *permit values* (i.e., features) might be altered to resemble those of the false class label. In other words, the manipulated feature values may contrast the unobserved true label. This complex NNAR scenario is the focus of our research. To address the issue of learning with label noise, at least three strategies have been proposed in the literature. We mention the first two strategies —*baseline* and *preprocessing*— and discuss to some extent the third strategy: *sample weighting*

The first strategy is a *baseline strategy*. It involves using robust noise-tolerant learning architectures [Abellán and Masegosa, 2010] by deploying regularisation and feature selection protocols [Wang et al., 2019, Ghosh et al., 2017]. Its main advantage is the fact that it does not require any data pre-processing. However, using this strategy alone is not sufficient as the information contained in the noisy samples is not considered.

The second strategy is a *preprocessing strategy* in which noisy samples are targeted for either removal from the dataset or for label swapping —data cleaning or cleansing [Miranda et al., 2009]. The action to remove or relabel a sample depends on chosen decision threshold. In practice, this is a difficult choice. It often leads to the case where too many or too few instances are targeted, hence compromising the performance of the classifier [Koplowitz and Brown, 1981].

The third strategy involves using *sample weighting* (i.e., coefficients) in the loss function during learning [Liu and Tao, 2015]. Optimally, sample weights reflect the (un)certainty of the recorded feature values and class labels. However, current weighting approaches present three main hurdles, which we discuss below.

First, a sample weighting strategy may require a separate non-noisy (i.e., curated) sample to compute the weights which is often not available [Ren et al., 2018]. Second, current literature tends to focus on non-negative weights. This may impact the learned model by not taking advantage of the certainty that a sample is probably a mislabel. Third, to assign adequate weights to samples based on their observed feature values and class label, an adequate measure of sample *belongingness* is required which is not easily tractable.

From the literature, belongingness is defined as follows [Kylberg and Sintorn, 2013].

Definition 4.1 – ***Belongingness***

Belongingness is a broad term representing the extent to which a class-labelled sample belongs to the class indicated by its label.

The computation of belongingness is generally addressed by exploiting the aforementioned baseline approach: a robust learner is trained on a dataset —or on a portion of it in a CV manner— and the prediction scores of the learned classifier are used as a measure of sample belongingness [Gamberger et al., 1999, Jeatrakul et al., 2010].

However, overfitting may occur when the entire dataset is used for training which frequently results in poor detection performance. Moreover, even if scores are computed with CV to avoid overfitting, the scores are not comparable across folds since the training sets are not the same.

Our Novel Approach

Given the lacunae in the literature, to improve classifier learning with noisy samples, we propose (1) a novel sample weighting strategy and (2) a new detection method. Since sample weighting is reliant on an appropriate measure of belongingness, a new method to label noise detection is really required. By using of a robust surrogate learner, we propose DENOISE, a sample weighting strategy for learning which leverages the belongingness of samples. As a result, we arrive at the following two contributions.

1. *Learning a classifier* that is resilient to the type of noise in such a way that performance loss is minimal compared to the non-noisy case;
2. *Detecting samples* of which the label is corrupted, in which feature values may be disharmonious with respect to the true (unknown) label.

Obviously, but quite important, the two contributions are related. Classifier learning with noisy data depends on the initial step of detecting samples with label noise.

The structure of the chapter is as follows. Section 4.2 states the terminology and the problem description precisely. Section 4.3 refers to the literature related to our work and reveals the open issues which lead us to our contributions. Section 4.4 describes our methods for classifier learning and the detection of samples with label noise. Section 4.5 describes our experimental setup. Section 4.6 presents the results of our experiments. Finally, Section 4.7 concludes this chapter and provides direction for future research.

4.2 Problem Description

Given a class-labelled dataset, we consider the scenario in which the feature values and target labels have been compromised. Moreover, we focus on the case where feature and class label noise distributions share dependencies; i.e., the NNAR scenario. Under this scenario, a noisy sample is defined as follows.

Definition 4.2 – ***Noisy sample***

A noisy sample under the NNAR scenario is a sample of which the class label and some of its feature values are untrue.

4.2.1 Noise Interpretation

Given a sample with an incorrect class label, we consider inappropriate feature values to be values which explicitly correlate more to the observed incorrect class than to the true unobserved one. The true (unobserved) label is thus further masked. This translates at least to two real-world phenomena: (1) disease-mapping given genetic admixture in populations [Schrider and Kern, 2018, Chen et al., 2014], and (2) data-tampering activities relevant to the risk assessment and fraud-detection domains [Diekmann and Jann, 2010]. Here we remark that this translation is a generalisation of the label noise case as studied in [Müller and Markert, 2019] with the addition of feature noise.

4.2.2 Formal Problem Description

In formal terminology, let $\mathcal{D}$ represent a distribution of a pair of random variables $(X, Y) \in \mathcal{X} \times \{+, -\}$, where $\mathcal{X} \in \mathbb{R}^m$. Let also $(X_1, Y_1), \dots, (X_n, Y_n)$ be an independent and identically distributed (i.i.d.) sample of $\mathcal{D}$, with $(\tilde{X}_1, \tilde{Y}_1), \dots, (\tilde{X}_n, \tilde{Y}_n)$ as a sample of the corresponding noisy distribution $\tilde{\mathcal{D}}$.

For a given independently distributed label noise rate $\rho_y = P(Y \neq \tilde{Y})$, we denote the feature noise rate ρ_x as the proportion of m dimensions of which the variable (feature) values are sampled from the distribution conditioned with respect to the noisy label $\tilde{Y}$. Under the described NNAR mechanism, the problem description is:

1. can we predict the label Y for an observation X, given noisy training observations $(\tilde{X}_i, \tilde{Y}_i)$? and
2. can we detect the noisy samples $(\tilde{X}_i, \tilde{Y}_i)$ where $\tilde{Y}_i \neq Y_i$?

4.3 Related Work

There is ample literature in both classifier learning in the presence of noisy samples and label noise detection. Here, we report on relevant work related to supervised learning approaches (Section 4.3.1) and detection techniques (Section 4.3.2) when label noise occurs.

4.3.1 Classifier Learning

Throughout the literature, different approaches have been proposed towards the task of learning with noisy data [Pechenizkiy et al., 2006, Manwani and Sastry, 2013, Teng, 2000, Yin and Dong, 2011, Teng, 2001]. At a broad level, these approaches can be partitioned into three not mutually exclusive categories: (1) robust learners; (2) classification filtering; and (3) sample weighting.

Robust Learners

From the theory [Bartlett et al., 2006] we know that commonly used loss functions in machine learning are not robust to label noise. Yet, some learning architectures and regularisation techniques have been empirically shown to present better results than others in mitigating noisy samples. A breakthrough in this area was [Dietterich, 2000]. There, it was shown that ensemble methods based on bagging achieved superior classification performance when compared to boosting. The reasoning behind is two-fold. First, boosting assigns large scores to mislabelled instances and focuses on those samples to produce the following additive decision boundaries. This leads to poor generalisations. Second, bagging uses different sampling subsets during learning improves on the dissimilarity between the base models making the final classifier more robust.

Still, more recent (gradient) boosting techniques such as LogitBoost and XGBoost have been shown also to be robust to label noise [Gómez-Ríos et al., 2017]. When compared to standard boosting approaches, gradient boosting techniques allow for the misclassification of the training samples rather than over-focusing on them during learning, mitigating overfitting. This factor, in connection with regularisation and feature selection protocols, makes up for greater efficacy when dealing with label noise [Abellán and Moral, 2003]. In summary, it is a challenging research area.

Classification Filtering

Filtering approaches are characterised by either (a) removing or (b) relabelling samples based on a threshold set upon the respective belongingness values.

From empirical experimentation published, the literature has shown that removing samples tends to be more efficient towards learning than relabelling [Cuendet et al., 2007]. However, too many samples might be targeted for removal and that negatively impacts the learning process. Conversely, if too many mislabelled instances are kept, the performance of the learned model is also heavily compromised [Koplowitz and Brown, 1981].

Sample Weighting

A more sophisticated approach to classifier learning with label noise involves using sample weights during learning; this approach is termed *sample weighting*.

*Definition 4.3 – **Sample weighting***

Sample weighting is an approach by which, under a classifier learning scenario, training samples are assigned *weights* according to some specific weighting strategy such that the weights reflect the contribution of each sample towards learning the final classification model.

These weights are applied as coefficients in the loss or error function during risk minimisation. As such, samples have either greater or lesser impact towards learning the final model.

Conceptually, instances with a higher belongingness score (see Section 4.3.2) will have higher weights, and vice-versa. The work presented in [Ren et al., 2018] shows how to estimate sample weights as a minimisation objective by having access to a proportion of non-noisy —curated— samples with ground truth labels. Having access to these data is not always possible, therefore limiting the applicability of this particular sample weighting strategy.

In [Liu and Tao, 2015], the authors demonstrate how any surrogate loss function designed towards a standard classification task can take advantage of sample weighting strategies when noisy labels are present. They propose a weighting strategy based on the ratio of distributions, often used in domain adaptation [Gretton et al., 2009], by assigning a sample weight $\mathcal{W}_i$ to the i^{th} instance based on the following ratio of posterior probabilities:

$$\mathcal{W}_i = \frac{P_{\tilde{\mathcal{D}}}(\tilde{Y}_i|\tilde{X}_i) - \rho_{-y}}{(1 - \rho_+ - \rho_-) \cdot P_{\tilde{\mathcal{D}}}(\tilde{Y}_i|\tilde{X}_i)}, \tag{4.1}$$

$$\rho_{-y} = \min_{\tilde{X} \in \tilde{\mathcal{X}}} P_{\tilde{\mathcal{D}}}(\tilde{Y}|\tilde{X}). \tag{4.2}$$

The drawback of this approach relates to the range of values of the weights. Since all weights are non-negative by definition [Scott, 2015], samples with a high probability of being noisy have a low contribution towards learning.

This characteristic is undesirable since the valuable information contained in those instances is lost. To some extent, it may be detrimental to classification performance, analogous to the removal of samples in the filtering methods.

4.3.2 Label Noise Detection

Most label noise detection methods are based on supervised learning approaches [Frénay and Verleysen, 2013]. Their purpose is two-fold: (1) to target samples which may indicate real-world noncompliance within the inspection domains [Pereira Barata et al., 2018b]; and (2) to perform data preprocessing towards classifier learning [Gamberger et al., 1996].

The task of detecting label-noisy samples is most commonly undertaken by analysing measures of belongingness of a sample towards its observed class label. In this sense, belongingness may be represented as either a score function or the class-conditional posterior probability $P(Y|X)$, both usually tractable by classifier learning [Jeatrakul et al., 2010, Thongkam et al., 2008, Brodley and Friedl, 1996]. Instances with a low score or posterior probability with respect to the class label may be flagged as mislabels [Sun et al., 2007]. Throughout the label noise detection literature, a recurring theme is the usage of supervised classification techniques to infer sample belongingness [Frénay and Verleysen, 2013]. Accordingly, belongingness may be computed according to one of the following three strategies.

The first strategy is to learn a classifier on the entire dataset and to deploy the trained model on the same dataset; in recent work [Müller and Markert, 2019], a robust learner was applied with the purpose of detecting mislabelled entries and presented them to human experts for further evaluation. Even though robust learners may be used, however, overfitting may still occur. Thus, mislabelled instances might be evaluated inappropriately, resulting in unreliable belongingness values.

The second strategy involves learning multiple classifiers on training subsets in a CV manner and deploying each trained model on the corresponding validation set [Gamberger et al., 1999]. While this strategy helps mitigate overfitting, classifier outputs are not comparable across folds since difference training sets were used to yield the scores [Bennett, 2000].

The third strategy is ensemble voting (e.g., majority or consensus by different learners). It can be applied to the generate votes by following either strategy previously mentioned [Miranda et al., 2009,John, 1995]. Since this strategy is dependent on the aforementioned strategies, all mentioned issues apply. A further disadvantage of these approaches is their focus on the removal or relabelling of samples: a poor decision described in Section 4.3.1.

Literature in both classifier learning with noisy samples and label noise detection tasks is extensive. With respect to classifier learning, simply using a robust learner is a too generic approach and does not actively leverage the noisy samples. Removing or relabelling samples is also not suitable since the choice of hard thresholds is difficult to justify.

Sample weighting can mitigate the threshold issue, but current strategies either require access to curated samples, which are hard to obtain, or they do not exploit the information of probable mislabels during training. That is, weight values asymmetrically take belongingness and non-belongingness into account. In terms of label noise detection, there is currently no solution to computing belongingness which provides both minimal overfitting and calibrated (i.e., comparable) output. Ultimately, a new sample weighting strategy is required, as well as a novel sample belongingness computation approach. In the following Section 4.4, we describe our method.

4.4 The DENOISE Method

Below now provide the details of our method: DENOISE. It is data-driven method which provides noise-resilient classification by effectively identifying noisy samples, jointly dealing with the problems of label prediction with noisy samples and label noise detection. Succinctly, it learns a surrogate classifier from noisy data which is robust to both label noise and feature noise collectively by means of a log-odds sample weighting strategy. The sample weights are retrieved when addressing the label noise detection problem.

For each instance we retrieve the calibrated belongingness values and use them to compute the respective sample weights. In turn, the belongingness values are yielded by several robust surrogate learners deployed in a CV fashion with the addition of a calibration protocol. The label noise detection is then solved by applying a sensible threshold to the calibrated output. In Section 4.4.1 we detail our classifier learning setup. In Section 4.4.2 we describe the label noise detection process.

4.4.1 Learning with Sample Weights

The main concept behind our method is directly linked to the strategy by which sample weights are computed. Given noisy samples $(\tilde{X}_1, \tilde{Y}_1) \ldots, (\tilde{X}_n, \tilde{Y}_n)$ of $\tilde{\mathcal{D}}$, a loss $\mathcal{L}$, and weights $\mathcal{W}_1, \ldots \mathcal{W}_n$, the task of the learner is to find a function $f \in \mathcal{F}$:

$$\operatorname*{arg\,min}_{f \in \mathcal{F}} \frac{1}{n} \cdot \sum_{i=1}^{n} \mathcal{W}_i \cdot \mathcal{L}(f(\tilde{X}_i), \tilde{Y}_i), \text{ where} \tag{4.3}$$

$$\mathcal{W}_i = \ln\left(\frac{P_{\tilde{\mathcal{D}}}(\tilde{Y}_i|\tilde{X}_i)}{1 - P_{\tilde{\mathcal{D}}}(\tilde{Y}_i|\tilde{X}_i)}\right). \tag{4.4}$$

We propose the sample weight of the i^{th} instance to be the log-odds of the event, i.e., the posterior probability; sample weights can therefore take zero, positive, or negative values.

Zero

Sample weights of zero only occur for samples with a posterior probability of 0.5. It entails no contribution to the learning of the final learner. This is sensible, since a posterior of 0.5 towards one class is the same towards the other.

Positive

Weights greater than zero translate to the learner having the information that the observed label is probably not noisy. The larger the weight associated with a specific instance, the more a learner is impacted by it, while *trusting* its label.

Negative

Weights lesser than zero follow the same logic as positive weights except that the observed label is assumed to be incorrect during learning. A negative weight inverts the output of the loss function: the learner is rewarded for incorrectly learning the observed label. The more negative a sample weight is, the larger the impact of that sample towards learning its opposite label.

Posterior Estimation

Albeit intuitive and conceptually simple, our method requires the estimation of posterior probabilities to compute the sample weights; see Eq. 4.4. To be able to compute these posterior probability estimates, and hence the sample weights, we use sample belongingness as a starting point towards acquiring the posteriors. Since belongingness has a myriad of caveats as detailed in Section 4.3.2, in the following Section 4.4.2 we show how to generate it appropriately, addressing the pitfalls mentioned in the literature.

4.4.2 Posterior Estimation and Detection

Our approach to label noise detection is data-driven in the sense that it uses a set of learning functions to compute the belongingness of samples. The belongingness will translate to the posterior probabilities required to compute the sample weights, required for classifier learning (mentioned in Section 4.4.1).

Learning Functions

We follow a supervised learning approach, where the belongingness of a sample is determined by a set of learned classifiers. The learners should be robust to noise and incorporate (1) regularisation, (2) feature selection, or (3) both protocols [Sharma et al., 2017]. The choice of architecture should be sensibly chosen given the type of data being handled; e.g., tabular data may be handled with gradient boosted approaches [Pafka, 2019], and image datasets by dropout convolutional neural network [Park and Kwak, 2016]. To minimise overfitting, belongingness is computed per class on a left out part, rather than on the training set. Consequently, we need to optimise several classifiers in a CV setup.

Since, each CV fold has a specific training set with which a learned classifier is yielded, the output of each classification model is not necessarily comparable. As a result, all classifiers must be calibrated such that their output is comparable. To note, all hyperparameters can be optimised through standard CV [Claesen and De Moor, 2015, Bergstra and Bengio, 2012].

Calibration

To calibrate the output of the learner functions, Platt scaling [Platt et al., 1999] is used. This is a widely accepted method in supervised learning literature which converts classifier output into well-calibrated posterior probabilities [Böken, 2021, Niculescu-Mizil and Caruana, 2005, Guo et al., 2017]. Here, multiple learner functions are learned and calibrated; calibration sets are used such that the output of a learner function is, itself, used as input towards re-learning the observed class label by sigmoid functions (i.e., LR modelling).

The original output of a learned model then becomes the estimated posterior probability through nested CV. For an outer K-fold CV setup, each K-training fold is further split using L-fold CV. The K-training folds serve to calibrate the learner functions. Probabilities are gathered by deploying the calibrated learners onto the respective K-test folds. The estimated posterior probabilities of all samples become the union of all the folds:

$$P_{\tilde{\mathcal{D}}}(\tilde{Y}|\tilde{X}) = \bigcup_{k=1}^{K} P_{\tilde{\mathcal{D}}}(\tilde{Y}_k|\tilde{X}_k). \tag{4.5}$$

$P_{\tilde{\mathcal{D}}}(\tilde{Y}_k|\tilde{X}_k)$ represents the posterior probabilities of samples from the k^{th} test fold, given by the learners calibrated on the respective k^{th} training fold.

Detection

To solve the noisy sample detection problem, we propose using the aforementioned estimated posterior probabilities as detection scores in which the lower the probability of a sample, the more likely that sample is to be flagged as noisy. A detection threshold may be applied to the sample probabilities, such that posterior values lesser than 0.5 flag samples as having label noise. Conversely, a monotonic transformation could be applied such that higher transformed probabilities equate to higher detection scores; e.g., the $-\log_2(x)$ function transforms probabilities into their information content, in which samples with a value higher than 1 are considered label noise.

4.5 Experiments

In this section we describe the experimental setup by which we evaluate our methods. The setup for classifier learning and label noise detection share three similarities:

1. ten noise-free classification datasets were gathered to allow for a controlled scenario, in which ground truth labels are known (Section 4.5.1);
2. to simulate a NNAR mechanism, a proportion of the class labels was flipped and a proportion of feature values were replaced, replicated for several random seeds (Section 4.5.2);
3. baseline state-of-the-art methods were used to gauge the comparative performance of our approaches (Section 4.5.3).

To measure the performance of classifier learning, classifiers were learned on noise-injected training sets, deployed on non-manipulated test sets, and the AUC [Narkhede, 2018] was computed (Section 4.5.3). For label noise detection, the detection targets were considered as the manipulated samples, and AP [Robertson, 2008, Naseer et al., 2018] was used as the performance measure of the detection task (Section 4.5.3).

For the choice of learner, a gradient boosted framework (XGBoost [Chen and Guestrin, 2016]) was selected for its robustness. The surrogate loss function applied was the logistic loss [Painsky and Wornell, 2018]. For reproducibility purposes, our setup is made available online with the all necessary code to download the datasets, manipulate them, perform the learning and detection tasks, and output the yielded results[1].

[1] `https://github.com/pereirabarataap/denoise`

4.5.1 Data

Since we are interested in measuring the performance of classifier learning and label noise detection, we gathered datasets with known class labels. Ten benchmark [Asuncion and Newman, 2007] classification datasets were retrieved from *openML* [Vanschoren et al., 2014]: an open, organised, and community-driven online ecosystem for machine learning. These datasets were selected for their heterogeneity regarding sample size, dimensionality (i.e., number of features), and feature type (e.g., number of numerical or categorical features).

Table 4.1 summarises each dataset with respect to aforementioned characteristics. Regarding it, *#samples* indicates sample size, and *#features* represents the number of features of which *#numeric* are numerical and *#category* are categorical. The datasets were then manipulated to simulate a NNAR mechanism.

Table 4.1: Datasets retrieved for noise simulations

ID	#samples	#features	#numeric	#category
1495	250	6	0	6
53	270	13	13	0
40710	303	13	5	8
40690	512	9	0	9
335	554	6	0	6
1510	569	30	30	0
40705	959	44	42	2
1462	1372	4	4	0
1504	1941	33	33	0
41143	2984	144	8	136

4.5.2 Synthetic Noise

Noise was synthetically generated by replacing class labels and feature values. Different combinations of label noise ρ_y and feature manipulation ρ_x were considered. Specifically for the learning task, datasets were first split into training (90%) and testing (10%) sets; only the training sets were injected with noise. Noise was generated ten times with different initialisation seeds per pair (ρ_x, ρ_y) to account for randomness.

Label Noise

Several proportions of label noise were considered. For each dataset, $\rho_y \in \{.05, .1, .15, .2, .25, .3, .35, .4\}$ label noise proportions were introduced following a uniformly-distributed sample selection. This range of values was chosen since $\rho_y < 0.05$ would have negligible impact on robust learners and $\rho_y > 0.4$ would prove too corrupt for any meaningful experimental results.

Feature Manipulation

To recreate the scenario in which feature values are manipulated, samples which were label-swapped had a proportion of their feature values replaced. The proportions $\rho_x \in \{0, .05, .1, .15, .2, .25, .3, .35, .4\}$ of randomly-selected features were selected per sample. Manipulated features had their values replaced per label-swapped sample as described previously. Replacement values were drawn from univariate feature distributions with parameters estimated conditionally from the category being mimicked.

The distributions used to sample the replacement values were modelled as either: (a) the normal distribution $\mathcal{N}(\mu, \sigma)$ for numeric features, with μ and σ as the estimated mean and standard deviation; or (b) the multinomial distribution with estimated event probabilities $\{p_1, p_2, \ldots, p_\pi\}$, π being the number of unique feature values.

4.5.3 Evaluation

We compared DENOISE to current methods of classifier learning with sample weighting and label noise detection. For all tasks, a learning framework was required which was robust to label noise.

Since the datasets in our experiments data are tabular and have heterogeneous characteristics, a gradient boosted framework with a surrogate logistic loss function was selected and applied equally to all scenarios being tested. For the classifier learning task, the sample weighting method in [Liu and Tao, 2015] (LT15) —detailed in Section 4.3.1, Eqs. 4.1 and 4.2— was used as benchmark. For the label noise detection task, the solution presented in [Müller and Markert, 2019] (MM19) —described in Section 4.3.2— was selected as our benchmark.

Learning Performance

Our method DENOISE and LT15 were evaluated using 10-fold crossvalidation. For each fold, the training set (90%) was comprised of noisy samples and the test set (10%) was comprised of non-manipulated samples. To note, only the weighting strategies of each method were evaluated during the learning task.

Both methods had access to the same posterior probability values per sample, which were yielded by DENOISE.

Classification performance was measured as AUC on the non-manipulated tests set for each fold. Accordingly, the true class label is our target prediction. AUC values were averaged across all folds per dataset, across different random seed initialisations, for each noise configuration pair (ρ_x, ρ_y). Each learning strategy being evaluated yielded one mean AUC score per dataset, per noise configuration pair.

Detection Performance

To compare DENOISE to MM19, the detection targets were considered to be the synthetically manipulated instances described previously. The measure of belongingness yielded by each specific method was used as the detection score. Accordingly, detection performance was measured in terms of AP, a common measure used in anomaly detection [Frery et al., 2017].

AP values were averaged across the different random initialisations, for each noise configuration pair (ρ_x, ρ_y). Each detection strategy being evaluated yielded one mean AP score per dataset, per noise configuration pair.

4.6 Results

Here, we present the results of our experiments. We present them in two manners for both tasks, classifier learning (Section 4.6.1), and label noise detection (Section 4.6.2). An *aggregated* presentation —Fig. 4.1 and Fig. 4.2— is comprised of averages across all performance measures yielded for all datasets and noise configurations. We present these results as heatmaps in which each cell corresponds to a noise pair (ρ_x, ρ_y): lighter cell tones indicate higher performance values. Numbers indicate the mean performance score yielded for that specific noise configuration, across all datasets and random initialisations, for each specific task and method used to solve that task.

A performance difference heatmap is also provided, showing comparative changes in performance between the methods being gauged. Similarly to the previous heatmaps, each cell represents the average performance gain of our method for a particular noise configuration, comparatively to the literature baseline usedc. Should we present all datasets and all noise configurations discriminably, a total of 720 performance values would need to be provided which would not be practical. A presentation *per dataset* —Table 4.2 and Table 4.3— relays the performance means and standard deviations yielded by all our experiments for a subset of datasets (IDs 1510, 10705, and 41143) and noise configurations $(0, .05)$, $(0, .4)$, $(.2, .2)$, $(.4, .05)$, and $(.4, .4)$.

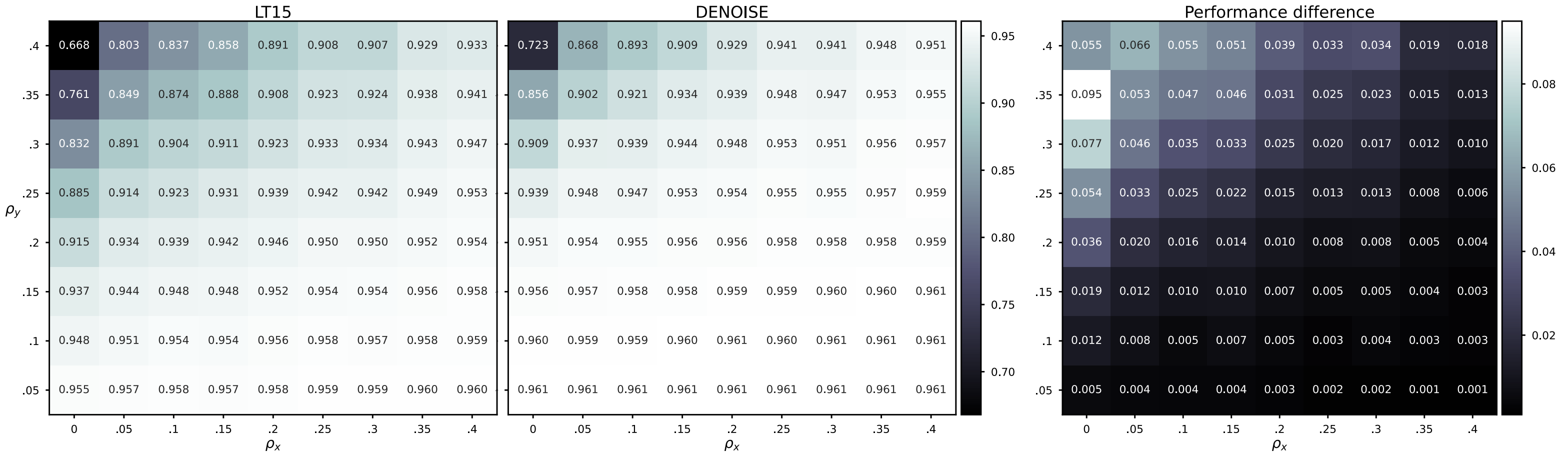

Figure 4.1: **Learning performances with noisy data (aggregated).** The three performance heatmaps represent LT15 (left), our DENOISE method (center), and the performance difference between the two (right). Each heatmap cell depicts the average AUC yielded for the corresponding noise configuration. The vertical axis represents the proportion of samples which have been label-swapped. The horizontal axis denotes the proportion of replaced features values in label-swapped samples.

Table 4.2: Learning performances with noisy data (per dataset)

Dataset ID	(ρ_x, ρ_y)	LT15	DENOISE
1510	$(0, .05)$	0.988 ± 0.002	0.987 ± 0.002
	$(0, .4)$	0.727 ± 0.014	**0.860 ± 0.047**
	$(.2, .2)$	0.984 ± 0.003	0.988 ± 0.002
	$(.4, .05)$	0.991 ± 0.001	0.988 ± 0.001
	$(.4, .4)$	0.987 ± 0.003	0.990 ± 0.002
40705	$(0, .05)$	0.968 ± 0.003	0.976 ± 0.002
	$(0, .4)$	0.710 ± 0.016	**0.827 ± 0.072**
	$(.2, .2)$	0.966 ± 0.003	0.967 ± 0.004
	$(.4, .05)$	0.978 ± 0.001	0.976 ± 0.001
	$(.4, .4)$	0.969 ± 0.004	0.962 ± 0.008
41143	$(0, .05)$	0.858 ± 0.003	0.865 ± 0.002
	$(0, .4)$	0.653 ± 0.012	**0.753 ± 0.014**
	$(.2, .2)$	0.854 ± 0.002	0.857 ± 0.002
	$(.4, .05)$	0.864 ± 0.002	0.867 ± 0.002
	$(.4, .4)$	0.847 ± 0.005	0.849 ± 0.006

These datasets and noise configurations were selected to provide a representative sample of the entire set of experiments. The datasets have mostly disparate characteristics (see Table 4.1), and the noise configurations values are the most spread. Bolded values indicate a mean performance gain of at least 0.01 of the corresponding method versus the other.

4.6.1 Classifier Learning Task

We present the *aggregated* average AUC performance scores yielded for the learning task in Fig. 4.1. The three heatmaps represent LT15 (left), DENOISE (center), and the performance difference between the two methods (right). For every configuration of label swapping (vertical axis) and feature manipulation (horizontal axis), DENOISE shows superior performance. The difference in AUC performance varies significantly across different noise configurations (ρ_x, ρ_y).

The minimum performance change is $\approx$ $.01$, seen in, for example, cell $(.4, .05)$. The maximum performance difference is $\approx .1$ in configuration $(0, .35)$. On average, the difference in performance tends to increase as the noise label proportion increases; i.e., the more label noise is present, the better our method fares. The performance difference also tends to increase as the feature manipulation proportion decreases.

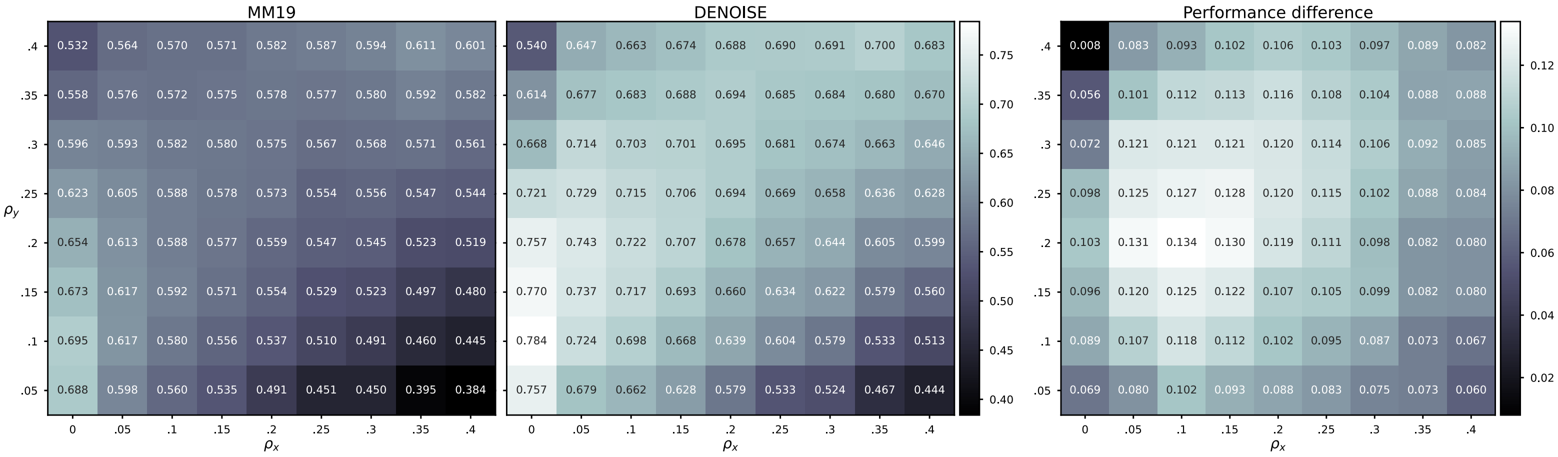

Figure 4.2: **Detection performances of noisy samples (aggregated).** The three performance heatmaps represent MM19 (left), our DENOISE method (center), and the performance difference between the two (right). Each heatmap cell depicts the average AP yielded for the corresponding noise configuration. The vertical axis represents the proportion of samples which have been label-swapped. The horizontal axis denotes the proportion of replaced features values in label-swapped samples.

Table 4.3: Detection performances of noisy samples (per dataset)

Dataset ID	(ρ_x, ρ_y)	MM19	Denoise
1510	$(0, .05)$	0.858 ± 0.037	0.860 ± 0.037
	$(0, .4)$	0.576 ± 0.014	**0.621 ± 0.034**
	$(.2, .2)$	0.719 ± 0.030	**0.837 ± 0.020**
	$(.4, .05)$	0.476 ± 0.072	**0.536 ± 0.052**
	$(.4, .4)$	0.763 ± 0.018	**0.825 ± 0.018**
40705	$(0, .05)$	**0.734 ± 0.033**	0.721 ± 0.032
	$(0, .4)$	0.569 ± 0.018	**0.627 ± 0.042**
	$(.2, .2)$	0.458 ± 0.029	**0.616 ± 0.025**
	$(.4, .05)$	0.196 ± 0.009	**0.322 ± 0.020**
	$(.4, .4)$	0.494 ± 0.016	**0.609 ± 0.022**
41143	$(0, .05)$	0.224 ± 0.033	**0.464 ± 0.019**
	$(0, .4)$	0.491 ± 0.011	**0.533 ± 0.027**
	$(.2, .2)$	0.260 ± 0.006	**0.436 ± 0.011**
	$(.4, .05)$	0.062 ± 0.003	**0.180 ± 0.014**
	$(.4, .4)$	0.364 ± 0.007	**0.503 ± 0.010**

In Table 4.2 we show the *per dataset* average AUC performance scores for the selected datasets and noise configurations, as well as the respective standard deviations along the ten different random initialisations. Bolded entries translate to an increase in mean performance of at least 0.01. The differences in performance are most apparent in noise configuration $(0, .4)$ in all datasets, where our method Denoise vastly outperforms LT15. In those cases, the performance gain is consistently ≥ 0.1.

Regarding the standard deviations, most noise pair configurations present similar values. However, two outliers stand out in the Denoise entries. Noticeably, datasets with IDs 1510 and 40705, since the same noise configuration $(0, .4)$ shows larger standard deviations than the LT15 method. However, it is also for those two datasets and the particular noise configuration that LT15 also has an increased spread of the mean performance comparatively to all other datasets and noise pairs.

4.6.2 Label Noise Detection Task

We display the *aggregated* average AP performance yielded for the detection task in Fig. 4.2. The three heatmaps represent MM19 (left), our Denoise method (center), and the performance difference between the two methods (right).

For every noise configuration, our method shows superior performance overall. The difference in AP performance varies significantly across different noise configurations (ρ_x, ρ_y). The minimum performance change is $\approx .01$, seen in, for example, cell $(0, .4)$. The maximum performance difference is $\approx .13$ in configuration $(.1, .2)$. Comparatively, the performance difference steadily decreases from its maximum, regardless of the direction taken in the horizontal or vertical axes.

Table 4.3 represents the *per dataset* AP performance scores for the selection of datasets and noise configurations and standard deviations across the ten random initialisations. With the exception of two entries, our approach yielded a mean performance gain overall. The differences in performance vary across datasets as well as along the noise pairs. For the majority of entries, our method shows higher standard deviation values. Yet, the differences are small. The largest difference in spread between the two methods is 0.024 for the entry with ID 40705 and noise pair $(0, .4)$.

4.7 Chapter Conclusion

In this chapter we proposed DENOISE, a method for noise-resilient classifier learning which leverages label noise detection via log-odds sample weighting. We compared our method to the state-of-the-art in learning with noise and label noise detection, under a NNAR mechanism in which label noise and feature noise may share dependencies.

In summary, with regard to the problem description, we may conclude that in the NNAR scenario DENOISE achieves overall (1) better class label predictions with noisy training data, and (2) better detections of those noisy samples than current literature methods.

We designed an experimental setup in which ten datasets with heterogeneous characteristics were used, representing different domains. For each dataset, different parameterised combinations of label noise and feature noise were extensively explored. Experimental setups were repeated ten times with different random initialisations.

Within the NNAR setting we considered the two tasks of: (1) learning a classifier that is resilient to this type of noise such that performance loss is minimal compared to the non-noisy case, and (2) detecting samples of which the label is corrupted, while the feature values may be disharmonious with respect to the true (unobserved) label. The results demonstrate that DENOISE overall outperforms the state-of-the-art overall in both learning and detection tasks.

Synthesising the NNAR mechanism is a complex task. On the one hand, corrupting class labels in a binary classification setting is rather straightforward.

On the other hand, the addition of feature manipulation involves applying dependency assumptions. While we chose a random feature selection strategy, a different approach could have been chosen; e.g., features could have been selected per class instead of per sample. A different approach to feature selection would be to follow a non-random selection strategy based on feature importance. Similarly, different robust classifiers could have been used, alongside different types of data, as well we different loss functions instead of the applied logistic loss. As such, our results are bound to our experimental setup and further work may be performed to other experimental setups. Yet, we have laid out a framework upon which experimental design choices can be made to generate specific noise scenarios.

Ultimately, handling noisy data remains a difficult task, even though noise mechanisms can be formally defined. For a NNAR case, as discussed in this chapter, the properties of the underlying distributions of the observed noisy data are often varied and not fully tractable. Future work could improve upon these simulations by using different assumptions over the noise in the data, e.g., by stipulating (1) different types of distributions for feature manipulation, and (2) varying correlations with the well-defined class labels.