Hyperparameter Selection under Localized Label Noise via Corrupt Validation

David I. Inouye[*], **Pradeep Ravikumar**
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{dinouye, pradeep}@cs.cmu.edu

Pradipto Das, **Ankur Datta**
Rakuten Institute of Technology
Rakuten USA
Boston, MA 02110
{pradipto.das, ankur.datta}@rakuten.com

Abstract

Existing research on label noise often focuses on simple uniform or class-conditional noise. However, in many real-world settings, label noise is often somewhat systematic rather than completely random. Thus, we first propose a novel label noise model called Localized Label Noise (LLN) that corrupts the labels in small local regions and is significantly more general than either uniform or class-conditional label noise. LLN is based on a k-nearest neighbors corruption algorithm that corrupts all neighbors to the same wrong label and reduces to a class-conditional label noise if $k = 1$. Given this more powerful model of label noise, we propose an empirical hyperparameter selection method under LLN that selects better hyperparameters than traditional selection strategies, such as cross validation, by synthetically corrupting the training labels while leaving the test labels unmodified. This method can provide an approximate and more robust validation for hyperparameter selection. We design several label corruption experiments on both synthetic and real-world data to demonstrate that our proposed hyperparameter selection yields better estimates than standard methods.

1 Introduction

Most previous work on label noise has focused on uniform [1–4] or class-conditional label noise [4–10].[1] Uniform label noise assumes that the probability of corruption is independent of both the true label while class-conditional label noise allows the corruption probability to depend on the true label. However, in many real-world settings, the label noise is not uniformly random but rather *locally systematic*. For example, in an e-commerce setting, suppose a merchant sets the default category for their products to be "Computers" but fails (for lack of effort or by accident) to change the category to "Media" for all their DVDs; this error would cause *all* of their DVDs to be labeled incorrectly—a local but systematic label noise. Shen et al. [16] suggest that even when merchants manually select the category for every item, they make approximately 15% error largely because of category ambiguity. Or as another example, hand-crafted rules may be used to label data in large batches so as to reduce labeling effort; however, many simple hand-crafted rules may introduce systematic errors in the labels. As a concrete example, suppose that any product with "ring" in the title is classified as jewelry; while this rule may provide reasonable accuracy, hardware items like "sealant ring" would all be misclassified into jewelry. In e-commerce, this label corruption problem can cause significant revenue loss because users are likely to become confused and navigate away from the website [16].

[*] A majority of this work was done during a summer internship at Rakuten Institute of Technology, Boston.

[1] Some recent work has considered cases of feature-dependent label noise [8, 11–13] or even adversarial label noise [14, 15] but is either primarily theoretical [8, 12, 14] or model-specific [8, 11, 13–15], where our proposed label noise model is empirically-driven and model-agnostic as will be described in future sections.

Presented as workshop paper at the 31st Conference on Neural Information Processing Systems (NIPS 2017).

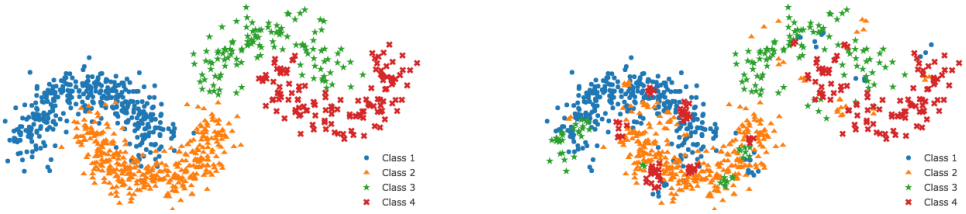

Figure 1: Quadruple moon toy dataset before corruption (left) and after localized label corruption (right), which yields systematic noise in local regions.

In light of this mismatch between uniform label noise and locally systematic label noise, we propose a corruption algorithm based on k-nearest neighbors that generates locally systematic label noise. Fig. 1 shows a toy dataset with true clean labels on the left and a corrupted version of the dataset on the right. Note that small localized regions of the data are corrupted together, e.g. five nearby points in one class are all corrupted to another class. This simple corruption algorithm implies a powerful generative model for localized label noise that we develop in the following sections. The harder question then is that, given such localized label corruption phenomenon, how might we train models to handle this label noise? Frenay and Verleysen [4] present three main approaches to handling label noise: (1) robust methods that are naturally robust to label noise but do not explicitly model label noise, (2) data cleansing methods that attempt to identify and remove corrupt labels before classification, and (3) noise-tolerant methods that jointly model the label noise and classification model. Our proposed method most closely follows approach (1) but can be applied to *any* classification model that has a complexity hyperparameter such as k in KNN, *tree depth* in decision tree, or *regularization parameters* in SVMs. In contrast to our proposed model-agnostic method, methods summarized in [4, Sec. V] are tied to certain loss functions (e.g. 0-1 loss), models (e.g. linear models) or algorithms (e.g. ensemble methods). From another viewpoint, we propose a method for estimating *relative* generalization error for hyperparameter selection that is less biased than traditional methods such as cross validation. In fact, Frenay and Verleysen [4] suggest that choosing hyperparameters under label noise is still an open problem. We make an attempt at resolving this important gap in the literature.

We summarize our two primary contributions as follows: (1) A novel model for systematic localized label noise including an empirical algorithm and probabilistic generative model, and (2) a novel label-noise-aware validation method that can be applied to make *any* classification method more robust to label noise. We believe these two contributions are complementary but can be used independently of each other and open up new research directions in both areas.

2 Localized Label Noise

We first present our localized corruption algorithm based on k-nearest neighbors. We then describe the probabilistic generative label noise model *implied* by the algorithm. Throughout this paper, we assume that we there exist n i.i.d. samples of a d-dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^d$ and corresponding true clean but *hidden* label $y \in \{1, 2, \dots, J\}$; $X \in \mathbb{R}^{n \times d}$ and $\boldsymbol{y} \in \{1, 2, \dots, J\}^n$ denote all n samples. We assume that the training algorithm only observes a *corrupted* version of the label denoted $\boldsymbol{y}_c \equiv c(\boldsymbol{y})$ rather than the true label $\boldsymbol{y}$.

Algorithm We describe a simple base corruption algorithm to build intuition, and then we will extend it to the full algorithm. Let us denote the total corruption rate or amount as π, e.g. $\pi = 0.2$ corresponds to 20% of the labels being corrupted. First, we initialize corrupt labels to the clean labels $\boldsymbol{y}_c = \boldsymbol{y}$ and corruption sets $\mathcal{C}^{(j)} = \emptyset$. Then, for all class values $j = 1, \cdots, J$ and while $|\mathcal{C}^{(j)}|/n < \pi$:

1. Sample data point $\tilde{\boldsymbol{x}}$ uniformly from the set $X^{(j)} \equiv \{\boldsymbol{x}_i \colon y_i = j\}$
2. Compute the k-size neighborhood of $\tilde{\boldsymbol{x}}$ in the set $X^{(j)}$ denoted $\mathcal{N}^{(j)}(\tilde{\boldsymbol{x}})$, which includes $\tilde{\boldsymbol{x}}$
3. Sample corrupted label $\tilde{y}$ uniformly from all labels except the j-th label
4. Corrupt all labels in neighborhood $\mathcal{N}^{(j)}(\tilde{\boldsymbol{x}})$: $[\boldsymbol{y}_c]_{\mathcal{N}^{(j)}(\tilde{\boldsymbol{x}})} \leftarrow \tilde{y}$
5. Update corrupted set: $\mathcal{C}^{(j)} \leftarrow \mathcal{C}^{(j)} \cup \mathcal{N}^{(j)}(\tilde{\boldsymbol{x}})$

Note that the user must specify k, which is not necessarily intuitive to estimate. Thus, we reparameterize this process by normalizing by the number of observations, i.e. $\mu = k/n$, so that the user only needs to specify an approximate percentage of observations that tend to be corrupted together. For example, a data vendor may estimate their overall corruption rate as $\pi = 15\%$ with local regions of size $\mu = 0.5\%$ may be corrupted together. This parameterization avoids having to manually scale

the algorithm based on the number of observations. Additionally, we can allow μ, a proportion parameter, to follow the Beta distribution with standard deviation σ. Thus our algorithm only requires three relatively intuitive parameters: total corruption percentage π (e.g. 10%), mean local corruption percentage μ (e.g. 1%), and standard deviation of local corruption percentage σ (e.g. 0.5%).[2]

Generative Model Building on the intuitions from the simple algorithm above, we now derive the implied population-level probabilistic generative model.[3] We assume the distribution of $\boldsymbol{x}$ given y is fixed but unknown and is denoted by θ. We also define the following epsilon ball given a point and a probability to cover: $\mathcal{B}(\tilde{\boldsymbol{x}}, \delta) = \mathcal{B}_\epsilon(\tilde{\boldsymbol{x}}) = \{\boldsymbol{x} \colon \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| \leq \epsilon\}$ such that $\mathbb{P}(\boldsymbol{x} \in \mathcal{B}_\epsilon(\tilde{\boldsymbol{x}})) = \delta$. Essentially, this is the unit ball in the feature space centered around $\tilde{\boldsymbol{x}}$ whose probability is exactly δ. With this notation, we define the localized label noise model via a generative graphical model:

1. For $j = 1, \ldots, J$ and for $\ell = 1, 2, \ldots$
   (a) Sample $\tilde{\boldsymbol{x}}_{j\ell} \in \mathbb{R}^d$ from the conditional data distribution given $y = j$
   (b) Sample labels $\tilde{y}_{j\ell}$ from a uniform distribution with all labels *except* j
   (c) Sample $\hat{\delta}$ from a Beta distribution parameterized by mean μ and standard deviation σ
   (d) Set $\delta_{j\ell} = \min(\hat{\delta}, \pi - \mathbb{P}(\boldsymbol{x} \in \mathcal{C}^{(j)} \mid y))$ to ensure the corruption does not exceed π
   (e) Update corruption set by new corrupt ball: $\mathcal{C}^{(j)} \to \mathcal{C}^{(j)} \cup \mathcal{B}(\tilde{\boldsymbol{x}}_{j\ell}, \delta_{j\ell})$
   (f) If $\mathbb{P}(\boldsymbol{x} \in \mathcal{C}^{(j)}) = \pi$, break inner loop, set the number of corruption points $m_j(\pi) \leftarrow \ell$
2. For $i = 1, \ldots, n$
   (a) Sample $y \in \{1, \cdots, J\}$ from prior class distribution
   (b) Sample $\boldsymbol{x} \in \mathbb{R}^d$ given the conditional distribution θ_y
   (c) Set y_c as below, where $m \equiv m_y(\pi)$ is the number of corruption points

$$y_c = \begin{cases} \tilde{y}_{y1} & , \boldsymbol{x} \in \mathcal{B}(\tilde{\boldsymbol{x}}_{y1}, \delta_{y1}) \\ \vdots & , \quad \vdots \\ \tilde{y}_{ym} & , \boldsymbol{x} \in \mathcal{B}(\tilde{\boldsymbol{x}}_{ym}, \delta_{ym}) \text{ and } \boldsymbol{x} \notin \cup_{a=1}^{m-1} \mathcal{B}(\tilde{\boldsymbol{x}}_{ya}, \delta_{ya}) \\ y & , \text{otherwise (i.e. uncorrupted)} \end{cases} \tag{1}$$

In Fig. 2, we compare our generative graphical model with the class-conditional and instance-dependent label noise models.[4] Note that while the label corruption for each instance is independent in previous models, our label noise model induces global dependencies between labels.

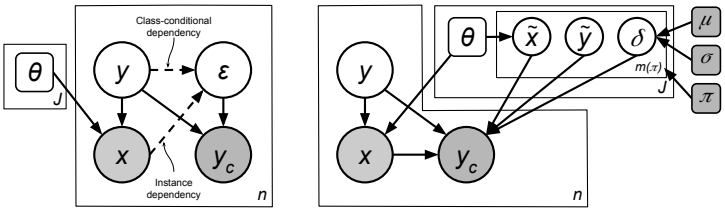

Figure 2: The graphical models of class-conditional label noise (left) and localized label noise (LLN) (right) show that LLN is a significantly more complex label noise model that shares global parameters across samples instead of being independent for each sample. π denotes the total percentage of corruption. μ and σ are the mean and standard deviation of a Beta distribution. Note how δ essentially becomes the percentage to corrupt together.

3 Corrupt Validation for Hyperparameter Selection

We will now consider hyperparameter evaluation methods and propose our method based on synthetically corrupting the training labels while leaving the test labels unmodified. Let $c_1(y \mid \boldsymbol{x}) \colon \{1, \cdots, J\} \times \mathbb{R}^d \to \{1, \cdots, J\}$ and $c_2(\cdot)$ be label corruption functions. In a real world scenario, $c_1(\cdot)$ is the original *unknown* (or hidden) corruption and $c_2(\cdot)$ is some sort of *known*

[2]Note that μ is the *percentage* that is corrupted simultaneously, not the volume in the feature space; thus, the local region will have larger volume in a low density area than a local region in a high density area.

[3]This generative model is primarily to build intuitions about the noise algorithm because we do not attempt to estimate the generative model; however, noise model estimation would be an interesting research direction.

[4]For clarity, we hide the implicit dependency of $\delta_{j\ell}$ on $\tilde{\boldsymbol{x}}_{ja}$ and δ_{ja} for $a < \ell$ via the auxiliary set $\mathcal{C}^{(j)}$.

synthetic corruption—in our case localized label corruption defined previously. We present two oracle validation methods that require oracle access to the true hidden labels, and two corresponding empirical validation methods that can actually be estimated from the observed data. We describe these validation methods below and give a visual summary in Fig. 3.

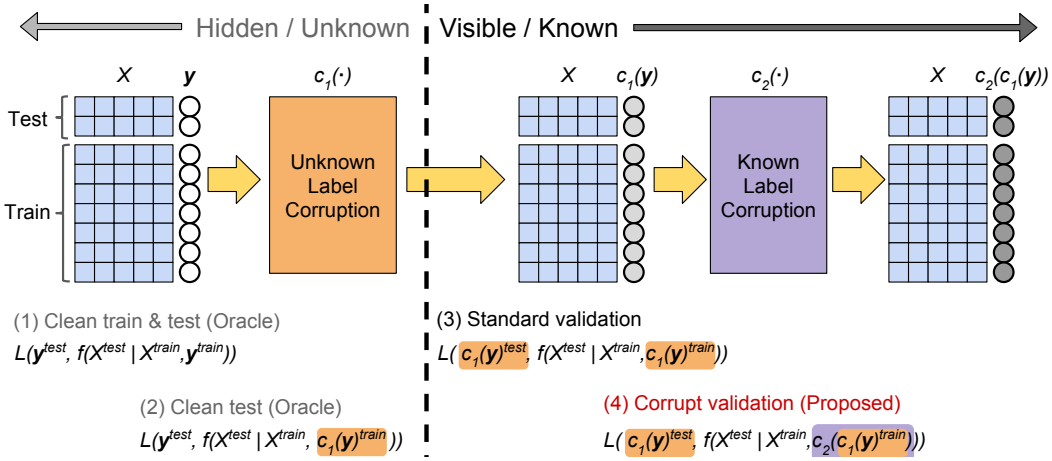

Figure 3: (1) is an oracle (access to hidden clean labels $\boldsymbol{y}$) method that assumes clean labels for both train and test. (2) is an oracle method that assumes clean labels only for testing and is thus the best hyperparameter estimator given only corrupted training labels. (3) is the traditional held-out validation that naively ignores label corruption. (4) is our proposed corrupt validation method that we suggest tracks the trends of method (2) more closely than (3), and thus selects better hyperparameters assuming the algorithm only has access to corrupt labels. $f(a \mid b)$ corresponds to a classifier prediction of a given training data b and L is a loss function.

(1) Clean train & test (Oracle, train: $\boldsymbol{y}$, test: $\boldsymbol{y}$) Validation method if full oracle access to true hidden labels is allowed for both for train and test. This is the classic academic setting where we assume there is no label corruption. **(2) Clean test (Oracle, train: $c_1(\boldsymbol{y})$, test: $\boldsymbol{y}$)** Validation method that only requires oracle access to the true hidden labels for the testing or evaluation phase. This method is the best that any validation model could do if the model can *only train on corrupt labels*. The difference between (1) and (2) is the performance penalty for having label noise. **(3) Standard validation (train: $c_1(\boldsymbol{y})$, test: $c_1(\boldsymbol{y})$)** Traditional validation method that naively ignores the label corruption. The underlying assumption is that this method mimics method (1). However, as we will show, approximating method (1) is actually not the goal because the model must be trained on corrupted labels rather than on the true (but hidden) labels. **(4) Corrupt validation (train: $c_2(c_1(\boldsymbol{y}))$, test: $c_1(\boldsymbol{y})$)** Corrupt validation method where the training labels are synthetically corrupted before training but the labels are left unmodified when evaluating the model. This validation attempts to mimic method (2), and can be seen as a *relative* proxy validation for (2).[5]

4 Experiments

We empirically evaluate our proposed corrupt validation on two datasets: a synthetic toy dataset with four moon-shaped classes from the introduction ($d = 2, n = 1000, J = 4$, see Fig. 1) and a real-world sample of automatically-curated[6] product catalog titles from a leading e-commerce company ($d = 82k, n = 1.2M, J = 29$, see Fig. 1). In these experiments, we set c_1 and c_2 to be the same localized label noise model using the simple empirical algorithm with $\pi = 20\%, \mu = 2\%, \sigma = 1\%$.[7] We implemented the methods and experiments within the scikit-learn Python framework [17]. For the toy dataset, we computed nested cross validation with three replications whereas we used a simple nested train/test validation for the larger dataset of product catalogs. For space reasons, we

[5] While this provides a useful relative model selection criteria, this method does not necessarily provide useful *absolute* performance estimates—estimating (2) correctly without true labels is still a significant open question.

[6] The curation has been done using manually crafted rules that yield high precision.

[7] An important open question is the sensitivity of the method to misspecification of c_2 because c_1 is usually hidden or unknown.

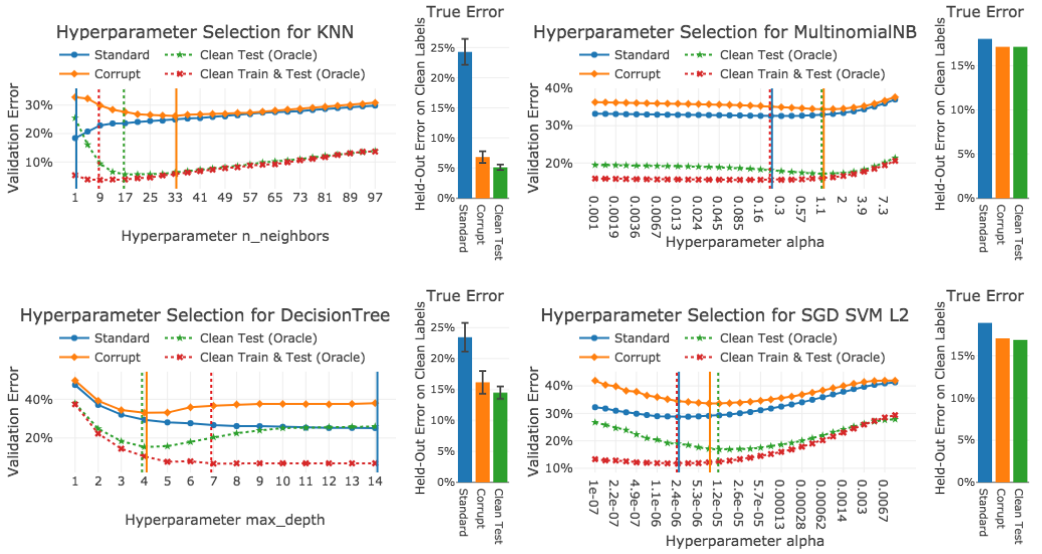

Figure 4: Average hyperparameter selection curves for various datasets and classification algorithms (line graphs, vertical lines are at minimum) and the true held-out test performance on clean labels of the best selected models (bar charts). (Left) Toy quadruple moons dataset. (Right) Product catalog listings from a leading e-commerce company. Our proposed corrupt validation (orange bar) performs better in terms of true held-out error than standard validation (blue bar)—where the gap is larger for non-parametric models like KNN and decision trees. The oracle clean test validation (green bar) is the best hyperparameter selection method if the final model can only be trained on corrupt labels. Note that the validation curves are only used to select the hyperparameter with lowest error; thus, only the *relative* error w.r.t. other hyperparameter values for each method is important. The hyperparameters of the models for the toy dataset are self explanatory. Hyperparameter α for MultinomialNB is the smoothing factor and α for SGD SVM is the L_2 regularization constant.

only give results for KNN and decision trees for the toy dataset, and multinomial naive Bayes and L_2-regularized SVM using SGD for the product catalog dataset as can be seen in Fig. 4.

Across all the experiments on both datasets, the corrupt validation follows the general trends of the oracle clean test validation method (method (2) from previous section) and thus selects hyperparameters that give better *true* generalization performance on the clean data (the bar charts in Fig. 4). While the traditional validation approach does track with the oracle validation on clean test and train data, this is suboptimal because the model can only be trained on corrupt labels—thus it is relative performance estimates are overly optimistic. Intuitively, the traditional method begins to fit the noisy labels when the model has high complexity whereas corrupt validation avoids overfitting the noisy labels. Also, note that the non-parametric models KNN and decision tree are more likely to overfit the label noise than the multinomial naive Bayes or linear SVM, which may be one reason linear models are often used in the real-world situations. However, our corrupt validation method enables the use of even non-parametric models in the presence of significant localized label noise.

5 Conclusion

We proposed a novel localized label noise model that is significantly stronger but often more realistic than the most common uniform or class-conditional models. We then developed a hyperparameter selection methodology that can be applied to any classification model to make it more robust to localized noise. We think this opens up many new and interesting research directions both in terms of label noise models and in terms of model validation under label noise. For example, how sensitive is our hyperparameter selection method to label noise misspecification? What are other label noise models could be used in the corrupt validation scheme that might better reflect real label noise? Can this validation methodology be used to identify the most noisy labels? In general, we believe this work is an important first step in understanding and handling more complex label noise in practice.

Acknowledgments

A majority of this work was completed by D.I. during a summer internship at Rakuten Institute of Technology, Boston. P.R. and D.I. acknowledge the support of NSF via IIS-1149803.

References

[1] Borut Sluban and Nada Lavra. Relating ensemble diversity and performance : A study in class noise detection. *Neurocomputing*, 160:120–131, 2015.

[2] Fabien Rico, Fabrice Muhlenbach, Djamel A Zighed, and Stéphane Lallich. Comparison of two topological approaches for dealing with noisy labeling. *Neurocomputing*, 160:3–17, 2015.

[3] Yiming Qian, Minglun Gong, and Li Cheng. Stocs: An efficient self-tuning multiclass classification approach. pages 291–306. Springer, Cham, 2015.

[4] Benoit Frenay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014.

[5] Jakramate Bootkrajang and Ata Kab. Label-noise robust logistic regression and its applications. In *ECML PKDD*, pages 143–158, 2012.

[6] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *NIPS*, 2013.

[7] S Liu, D Maljovec, B Wang, P Bremer, and V Pascucci. Visualizing high-dimensional data : Advances in the past decade. *Eurographics Conference on Visualization (EuroVis)*, 2015.

[8] Aritra Ghosh, Naresh Manwani, and P S Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93–107, 2015.

[9] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694v2*, 2017.

[10] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: a loss correction approach. In *CVPR*, 2017.

[11] Raj S Chhikara and Jim McKeon. Linear discriminant analysis with misallocation in training samples. *Journal of the American Statistical Association*, 79(388):899–906, 1984.

[12] Aditya Krishna Menon, Brendan Van Rooyen, and Nagarajan Natarajan. Learning from binary labels with instance-dependent corruption. *arXiv preprint arXiv:1605.00751v2*, 2016.

[13] Jakramate Bootkrajang. A generalised label noise model for classification in the presence of annotation errors. *Neurocomputing*, 192:61–71, jun 2016.

[14] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160:53–62, 2015.

[15] Nenad Tomasev and Krisztian Buza. Hubness-aware knn classification of high-dimensional data in presence of label noise. *Neurocomputing*, 160:157–172, 2015.

[16] Dan Shen, Jean-David Ruvini, and Badrul Sarwar. Large-scale item categorization for e-commerce. *Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*, (492):595, 2012.

[17] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.