Assignment 2

Tutors: **Nicholas James, Zhuozhuo Tu, Liu Liu**

Group members: **Chen Chen (cche5002) 480458339,**
Yutong Cao (ycao5602) 470347494,
Yixiong Fang (yfan5798) 480133344

Abstract

This report documents our modifications to the soft-margin support vector machine to improve its performance against class-dependent classification noise where the flip rates are $\rho_{-1} = 0.2$ and $\rho_{+1} = 0.4$.

Contents

1 Introduction

This report documents our modifications to the soft-margin support vector machine (SVM) to improve its performance against class-dependent classification noise (CCN).

The capability of learning with label noise is crucial for training machine learning models. According to the Law of Large Numbers (**?**), the empirical risk converges to the expected risk asymptotically as the sample size approaches infinity. However, classification labels in large data set can be easily corrupted. For example, images are sometimes labelled manually by employing casual staff with minimal training. In this case, training a classification models using these images requires special treatment that accounts for noisy labels. Section 3 proposes three such treatments based on SVM to increase the accuracy with presence of CCN.

Section 3.4 proposed a learning method by deriving a new loss function using Expectation Maximisation (**?**, p.423). This method models the label noise using a Bernoulli distribution, and includes the label noise information into the loss function of SVM. This method is first studied by **?** for random classification noise (RCN), and we extended it to CCN.

Section 3.5 implements importance reweighting proposed by **?**. This method uses a reweighting coefficient to reweight the loss function for each sample point. We fit SVMs with the given noisy data. We then implement cross-validation of SVM models to predict the probabilities of classifying each sample, and then use these probabilities to calculate the weightings.

Section 3.6 introduces a filtering approach based on the work of **?**. This method follows the heuristic observation that the predicted probability of a sample point with wrong label is usually close to 0.5, where our model is not able to classify confidently. In the meanwhile, the predicted classification result of contaminated label is likely to be different from the observed label. We then exclude the potential contaminated data points and train the model again with the remaining data which seems to be cleaner.

To fairly compare these three approaches of attacking noisy data, all of the three methods implement the same base classification algorithm—SVM. We use SVM because it is proved to has a strong generalisation ability (**???**), and usually gives high classification accuracy when turned properly (**?**). In addition, SVM is more susceptible to label noise than many other algorithms (**?**), so we can better assess our modification to the original classification algorithm.

Section **??** applies the three methods proposed in Section 3 to two sets of noisy data with binary labels, both containing $10,000$ images. The first data set is a subset of the fashion-MNIST database

and the second data set is from the CIFER database. For each data set, Section **??** also compares the simulation results, including accuracy and running times, from the three methods. In addition, we estimate the flip rate ρ_{-1}, ρ_{+1} using the method proposed by **?**. Section **??** also compares our estimation with the true values.

2 Related work

Many recent literature discusses the topic of learning with label noise. One popular approach is to use classification algorithms that are proved to be robust to label noise. **?** reported their findings that $0-1$ loss and least-squares loss are robust loss functions to uniform label noise. They also found the method of bagging is robust against label noise. Bagging detect the contaminated samples by estimating the variability of its base classifier when including and excluding them. **?** also discussed filtering methods for learning with label noise. These methods remove mislabelled data before training a final model. Our second model follows this idea by excluding data points with vague predictions. Instead of removing the contaminated candidates directly, **?** proposed a filter-like method that estimating the probability of mislabelling for each sample.

These approaches mentioned above do not require noise rates. Although this makes the approaches general, the methods may not be able to handle highly contaminated data set, especially when the noise rates are given. **?** applies Expectation Maximisation method to fit data with RCN. This method reformulates the loss function to include noise rate information. The label noise is not usually observed. They use the expected value as an estimation of unobserved labels contaminated by noise. Section 3.4 extends his method to solve problems with CCN. The idea behind this method is straightforward and intuitive, but simulation results in Section **??** finds it is less accurate than the method of importance reweighting proposed by **?**. **?** assigned a weight to each sample according to its probability of being contaminated. These probabilities are estimated from a pre-train model. They also provided an efficient method for estimating the noise rate. However, the need of fitting the model twice makes the method of reweighting slow.

3 Methods

We use SVM with Gaussian kernel as the base classification method for this task because of its generalisation ability.

Let vector x_i to represent the ith image in the dataset. The corresponding true classification labels y_i of images are corrupted by CCN and the we only observe the contaminated labels s_i.The clean and observed label y_i and s_i are both binary variables drawn from set $\{-1, 1\}$. The nature of CCN implies that class dependent noises $\epsilon(s_i)$ follows Bernoulli distributions with the mean parameters as either ρ_{-1} or ρ_{+1}, depending on the value of label s_i. The observed label s_i is correct if and only if the noise $\epsilon(s_i) = 0$. For the ith image, it is straightforward to verify that

$$y_i = s_i(1 - 2\epsilon(s_i)). \tag{1}$$

SVM with Gaussian kernel was first published by **?**. **?** proposed an improvement on the original SVM with soft-margin to avoid over-fitting problem. The SVM is to minimise the Hinge loss function

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(w \cdot x_i - b)\right)\right] + \lambda\|w\|^2.$$

Here, w is the weighting factor, b is a constant. We classify the ith image into category 1 or -1 when $w \cdot x_i - b \geq 1$ or $w \cdot x_i - b \leq -1$, respectively. This loss function not only penalises points that misclassified, but also points that are closed to the dividing hyperplane, with a regularisation term $\lambda\|w\|^2$. The decision function of SVM is

$$g(x_i) = \text{sign}(w \cdot x_i - b). \tag{2}$$

? asserted that SVM was very likely to be one of the most powerful classifiers, by comparing 179 classification algorithms over 121 large data sets. The distances between data points and the dividing hyperplane give an intuitive estimate of the generalisation ability of the trained SVM model (**?**).

we use SVM as our classification method as of its strong generalisation ability. In this image classification assignment, we do not observe the true labels y_i. This section proposes three different approaches to modify ordinary SVM to attack label noise problem. However, we do not have access to a test data set with true labels (x_i, y_i) to verify the generalisation ability of our three different models, which are all trained by noisy data (x_i, S_i). In SVM, the gap between hyperplane and the training data set provides a natural and 'free' metric of its generalisation ability (**?**), and hence use of test data set is not mandatory. Using this geometry factor as well as probably approximately correct learning framework, **?** proved that the SVM models have strong generalisation ability. Further, (**??**) also shows the strong generalisation ability from different perspectives.

you might make a mistake on the reference. I changed it from **?** to (**?**). please cross check for me

The other reason of implementing SVM as the base algorithm is that the hinge loss function makes SVM not robust against label noise (**?**). Nevertheless, our proposed modifications in Sections 3.4, 3.5 and 3.6 give accurate classification results as shown in Section **??**. If we implement a robust algorithm, e.g. deep neural networks, then one could argue that the flip rates had minimal impact on the classification result anyway. Hence whether our methods perform well would become a mystery.

3.1 Preprocess

3.1.1 Photometric normalisation improves classification performance

We applied Photometric normalisation suggested by **?** to re-scaled the image data sets to have mean of zero and standard deviation of one. This scaling visually removes the brightness difference among different images, and dramatically improves the performance of Gaussian kernel SVM. This preprocess procedure is mandatory because Gaussian kernel is a radius based kernel, which only performs well when data are on a similar scale (**?**).

3.1.2 Principle component analysis reduces dimensionality

For the large CIFER data set, we applied principle component analysis (PCA) to decrease the dimensionality of the data set from 3072 to 100. Although the dimensionality is reduced by 97%, the remaining 3% features explains more than 85% of the variance in the data set.

When fitting our models in Sections 3.4, 3.5 and 3.6, no PCA is applied to the MNIST data set because the dimensionality (784) is already low comparing with the number of samples $(10,000)$ in the data set. Nevertheless, when estimating the flip rates ρ_{-1} and ρ_{+1} in Section 3.3, we applied PCA to decrease the number of features of MNIST data set to 50.

3.2 The original data set is balanced

Define random variables Y and S as the true and contaminated binary classification label of a random image vector X, respectively. The assignment instruction states the probabilities $\rho_{-1} = P(S = 1|Y = -1) = 0.2$ and $\rho_{+1} = P(S = -1|Y = 1) = 0.4$. The contaminated data has 40.0% labels S as one (i.e. $P(S = 1)$). Using this factor and the law of total probability

$$P(S = 1) = P(S = 1|Y = 1)P(Y = 1) + P(S = 1|Y = -1)P(Y = -1),$$

An observe label can either be 1 or -1, $P(S = 1|Y = 1) = 1 - P(S = -1|Y = 1)$,

$$P(S = 1) = [1 - P(S = -1|Y = 1)]\, P(Y = 1) + P(S = 1|Y = -1)P(Y = -1)$$

Knowing the flip rates $P(S = -1|Y = 1) = 0.4$ and $P(S = 1|Y = -1) = 0.2$

$$P(S = 1) = 0.6P(Y = 1) + 0.2\,[1 - P(Y = 1)] = 0.4, \tag{3}$$

which implies $P(Y = 1) = P(Y = -1) = 0.5$. Thus, the original classification problem is balanced. In addition, define the Bernoulli random variable $\epsilon(S)$ with the means $E(\mu(\epsilon)(S = -1)) = P(Y = 1|S = -1) = 0.5 \times 0.4/0.6 = 1/3$ and $E(\epsilon(S = 1)) = P(Y = -1|S = 1) = 0.5 \times 0.2/0.4 = 0.25$. This random variable ϵ describe the unobserved random label noise. Hence the expectation

$$E\epsilon(S) = P(Y = -1|S = 1)P(S = 1) + P(Y = 1|S = -1)P(S = -1) = 0.3. \tag{4}$$

3.3 Flip rates estimation

We apply techniques proposed by **?** to estimate the flip rates ρ_{-1} and ρ_{+1}. The technique consists of two steps—firstly, we estimate probability $P_{D_\rho}(S|X)$; secondly, we estimate the flip rates ρ_{-1} and ρ_{+1} using the probability $P_{D_\rho}(S|X)$.

3.3.1 Density ratio method estimates conditional probability

Bayesian formula expands the conditional probability $P_{D_\rho}(S = s|X)$ to

$$P_{D_\rho}(S = s|X) = \frac{P_{D_\rho}(X|S = s)P_{D_\rho}(S = s)}{P_{D_\rho}(X)}. \tag{5}$$

Here the probability $P_{D_\rho}(S = s)$ is given by equation (3). We use the density ratio method proposed by **?** to estimate the ratio $P_{D_\rho}(X|S = s)/P_{D_\rho}(X)$.

Define a Gaussian Kernel

$$k(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|). \tag{6}$$

Then the set $\{k(X, x_j)\}_j$ forms the basis function for the probability space of $P_{D_\rho}(X|S = s)/P_{D_\rho}(X)$. Hence we can model the probability by a linear combination of the basis functions

$$\frac{P_{D_\rho}(X|S = s)}{P_{D_\rho}(X)} = \sum_j \alpha_{sj} k(X, x_j) \text{ where } \alpha_{sj} > 0.$$

To compute coefficients α_{sj}, we treat the centres x_j and the number of basis functions as hyperparameters and minimises the objective function

$$\min_\alpha \int \left(\frac{P_{D_\rho}(X|S = s)}{P_{D_\rho}(X)} - \sum_j \alpha_{sj} k(X, x_j) \right)^2 P_{D_\rho}(X)dx \text{ where } \alpha_{sj} > 0.$$

Treating terms without coefficients α_{sj} as constants and applying Monte Carlo to transform the integration into a summation, this objective function further simplifies to a constrained quadratic programming problem about coefficients α_{sj}. Solving this quadratic programming problem gives coefficients α_{sj}, and hence an approximation of the ratio $P_{D_\rho}(X|S = s)/P_{D_\rho}(X)$. Substituting this ratio into Bayesian formula (5) gives required conditional probability $P_{D_\rho}(S = s|X)$.

The advantage of the density ratio method is to avoid estimating probabilities $P_{D_\rho}(X|S = s)$ and $P_{D_\rho}(X)$. Estimating these are usually difficult for high dimensional data X, due to curse of dimensionality.

3.3.2 Estimate flip rates

Theorem 4 proposed by **?** estimates the flips rates using the global minimum of conditional probability $P_{D_\rho}(S = s|X)$ computed in Section 3.3.1

$$\rho_s = \min_X P_{D_\rho}(S = s|X).$$

As our sample size is large $(10,000)$, the minimal probability of the samples approximates the global minimum well.

The method proposed here requires tuning a few hyperparameters by grid search. Thus, later sections do not use this density ratio method for the speed of our algorithms. In addition, when numerically computing these flip rates, we only use the data set MNIST. This data set is much smaller comparing with CIFAR, and hence allows us to grid search the best hyperparameter faster. To further avoid over fitting, we applied PCA to decreases the number of features to 50 when estimating flip rates.

As the true flips rates ρ_{-1} and ρ_{+1} are given, here and after, we use the true flip rates $\rho_{-1} = 0.2$ and $\rho_{+1} = 0.4$ in Sections 3.4, 3.5 and 3.6.

3.4 Method 1: Expectation Maximisation

This section extends the Expectation Maximisation algorithm proposed by **?** to improve ordinary SVM against labels noises, with mathematical justifications.

3.4.1 Expectation Maximisation derives loss function

The original algorithm proposed by **?** was proposed to study classification problems with label noise where the flip rate $\rho_{-1} = \rho_{+1}$ and we extend it to manage the case dependent label noise where the flip rates can be different.

Recall the dual problem of an SVM is to maximise

$$f(c_1 \dots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i k(x_i, x_j) y_j c_j, \tag{7}$$

subject to $\sum_{i=1}^{n} c_i y_i = 0$, and $0 \leq c_i \leq \frac{1}{2n\lambda}$ for all i. Here y_i and x_i are the labels and features of the ith image, c_i is the ith Lagrangian multiplier, λ is a regularisation parameter,and $k(x_i, x_j)$ is the Gaussian Kernel product (6) of vectors x_i and x_j.

Substituting label noise (1) into loss function (7) gives

$$f(c_1 \dots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} S_i c_i k(x_i, x_j) S_j c_j (1 - 2\epsilon(S_i))(1 - 2\epsilon(S_j)). \tag{8}$$

This loss function involves random variables. For a similar but simpler problem with RCN, **?** applied the technique of Expectation Maximisation, which uses the expected value of the loss function as the objective function for optimisation algorithms.

Here, when $i = j$, the expectation $E(1-2\epsilon(S_i))(1-2\epsilon(S_j)) = 1-4E\epsilon(S_j)+4E\epsilon(S_j^2) = 1$. When $i \neq j$, by substituting these expectations (4) from Section 3.2, the expectation $E(1 - 2\epsilon(S_i))(1 - 2\epsilon(S_j)) = (1 - 2E\epsilon(S_i))(1 - 2E\epsilon(S_j)) = 0.16$.

Define kernel correction matrix M with the (i, j)-th entry being m_{ij}.The diagonal entries $m_{ii} = 1$, and the off diagonal entries $m_{ij} = 0.16$ when indexes $i \neq j$. Taking the expected values of the loss function (8)

$$Ef(c_1 \dots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} S_i c_i k(x_i, x_j) S_j c_j m_{ij}. \tag{9}$$

3.4.2 Modifying kernel improves robustness against label noise

Comparing this loss function and the ordinary loss function of SVM(7), the only difference is to replace the kernel matrix K with our new proposed matrix $Q := K \circ M$, where $\circ$ denote the Hadamard product (**?**).

This method gives us an accuracy of 94.6% on the testing data.

3.5 Method 2: Importance Reweighting

This section applies theorems proved by **?** to construct a weighted loss that includes noise information ρ_{-1} and ρ_{+1}.

3.5.1 Sigmoid function estimates conditional probability

Define D to be the probability density function of the clean data (X, Y) and D_ρ to be that of contaminated data (X, S).

This section estimates the conditional probability $P_{D_\rho}(S = y|X)$.

? provided three methods to estimate the conditional probability $P_{D_\rho}(S = y|X)$. The first method implements a probabilistic regresion model. The second and third method both implement Bayesian formula (5).

$$P_{D_\rho}(S = y|X) = \frac{P_{D_\rho}(x|S = y) P_{D_\rho}(S = y)}{P(X)}.$$

However, the two methods are computationally different. The formal method requires to estimate two density functions $P_{D_\rho}(X|S = y)$ and $P(X)$ by kernel smoothing. Our data sets are high dimensional (784 and 3072), so kernel smoothing suffers from the curse of dimensionality. The later

method avoids kernel smoothing by estimating the ratio $P_{D_\rho}(x|S=y)/P(X)$ directly through density ratio method (**?**). For example, Section 3.3 applies this method to estimate the flip rates ρ_{-1} and ρ_{+1}. Despite of the need of additional hyperparameters, **?** claims that the density ratio method is "equivalent to the regression problem". To keep our model simple and more interpretable, we implement a probabilistic regression model proposed in by **?**, Section 5.1.

We apply probability estimation method proposed by **?**. We first fit a pre-training model using an ordinary SVM. We then estimate the parameters of an additional logistic regression model to map the pre-training SVM outputs into probabilities. We apply this approach rather than traditional logistic regression because Remark 1 given by **?** indicates the logistic regression "does not perform well".

3.5.2 Reweighting coefficient improves robustness against label noise.

To improve the robustness against label noise, **?** constructed a reweighting coefficient vector

$$\beta(X,Y,S) = \frac{P_D(Y|X)}{P_{D_\rho}(S|X)}. \tag{10}$$

With some reasonable assumptions (what are the assumptions), **?** also derived the distribution of the unobserved true label using the observed label S, the flip rates ρ_0 and ρ_1

$$P_D(Y=y|X) = \frac{P_{D_\rho}(S=y|X) - \rho_{-y}}{1-\rho_y-\rho_{-y}}. \tag{11}$$

Substitute this estimation into the weighting coefficient (10):

$$\beta(X,Y,S) = \frac{P_{D_\rho}(S=y|X) - \rho_{-y}}{(1-\rho_y-\rho_{-y})P_{D_\rho}(S=y|X)}. \tag{12}$$

Further substituting the conditional probability $P_{D_\rho}(S=y|X)$ estimated in Section 3.5.1 gives reweighting coefficient $\beta(X,Y,S)$. Recall from equation (2) that g is the decision function. This reweighting coefficient $\beta(X,Y,S)$ is then used to estimate the expected risk $R_{\ell,D}(g)$ in the with the empirical risk $R_{\ell,D_\rho}(g)$

$$R_{\ell,D}(g) = R_{\beta\ell,D_\rho}(g). \tag{13}$$

This loss function allows us to estimate the true loss function $R_{\ell,D}(g)$ without knowing the true labels Y.

3.6 Method 3: heuristic approach by relabelling

Method 3 implements SVM to select samples that we can more confidently make predictions.

3.6.1 Conditional probability filters samples

Ordinary SVM only gives a classification without revealing a probability that indicates the confidence of classification. **?** proposed a five-fold cross-validation method to calculate the classification probabilities $P(Y=1|X)$ for SVM. Using this method, we calculated the probability $P(Y=1|X)$ with label noise by a pre-training SVM with Gaussian kernel. Recall from equation (2) that we use $g(X)$ to denote the decision made by this SVM.

Each data set has $10,000$ samples. For each data set, we find the $1/3$ and $2/3$ percentiles of the sample conditional probability $P(Y=1|X)$, and denote the 3333rd smallest and the 3333rd greatest probability sample as $P(Y=1|X_{(3333)})$ and $P(Y=1|X_{(6667)})$, respectively. We only use a subset of samples where the predicted probability is very high or very low

$$\left\{X|P(Y=1|X) \geq P(Y=1|X_{(3333)}) \cup P(Y=1|X) \leq P(Y=1|X_{(6667)})\right\}. \tag{14}$$

This method is inspired by the fact that the contaminated samples are more likely to have a probability $P(Y=1|X)$ close to 0.5. One third of sample are truncated because of the error rate $P(\epsilon=1) = 0.3 \approx 1/3$. Note that this subset of sample is imbalanced.

3.6.2 Label correction

We fit our model using the predictions $g(X)$ rather than contaminated labels S, because the way we filtered samples assure us exactly half of predictions $g(X)$ in the filtered samples (14) are ones. Using a balanced data is important because section 3.2 shows that the original data set is also balanced. In addition, filtered samples (14) have predicted probabilities far from 0.5. Hence, the pre-training SVM are confident with its predictions $g(X)$. Training our SVM with Gaussian Kernel again with this balanced subset of relabeled data gives the result.

Denote the indicator function $I_{g(X)=S} \in \{0, 1\}$ as the binary result indicating whether decision made $g(X)$ is the same as the observed labels S.With this definition, the expression $2I_{g(X)=S} - 1$ is -1 when $g(X) \neq S$ and is one when $g(X) = S$. This model is equivalent to minimise the loss function (13) with the weighting coefficient

$$\beta \begin{cases} 2I_{g(X)=S} - 1 & \text{when } P(Y|X) \leq P(Y = 1|X_{(3333)}) \cup P(Y|X) \geq P(Y = 1|X_{(6667)}), \\ 0 & \text{when } P(Y = 1|X_{(3333)}) < P(Y|X) < P(Y = 1|X_{(6667)}). \end{cases}$$

This method gives us an accuracy of 95.0% on the testing data.

3.7 Tuning hyperparameters

The hyperparameters are tuned by Newton's iteration and coordinate descent. Knowing the total error rate (4), the objective is to achieve an accuracy of 70% training the given data set with CCN.

The Kernel parameter for Gaussian Kernel γ is initially chosen to maximise the variance of Kernel matrix $K(x_i, x_j)$ over all i, j. The initial value of regularisation parameter λ is chosen as 1. The gradient is estimated by finite difference method. The model seems to be insensitive to the regularisation parameter λ.

3.8 Bootstrap constructs confidence intervals and hypothesis tests

The assignment instruction asks us to compare the accuracy, denoted as A, of the proposed label noise robust algorithms.

3.8.1 Bootstrapping percentile confidence intervals

To systematically compare the accuracy of the three methods introduced in Sections 3.4, 3.5 and 3.6, we construct the 95% bootstrapping percentile confidence intervals for accuracy A. The idea of bootstrapping is straightforward—we resample a subset of 8 samples among the sample space of the 16 accuracy results and calculate their mean. We repeat this process 1000 times. The 2.5% and 97.5% percentiles of the 1000 re-sampled means are then the bootstrapping percentile confidence interval

$$(A^*_{2.5}, A^*_{97.5}), \tag{15}$$

where A^*_α is the α percentile of the bootstrapped distribution from our sample space with 16 accuracy result from Monte-Carlo simulations.

Bootstrapping does not require the sample space follows specific distributions. We apply this non-parametric method to construct confidence interval here because we do not know about the exact distributions of accuracy A.

3.8.2 Kolmogorov-Smirnov test compares the accuracy of algorithms

We implement the Kolmogorov-Smirnov test to test the hypothesis that the algorithms proposed in Section 3 have different accuracy. Again, Kolmogorov-Smirnov test is distribution free, so we do not need to know the distributions of the evaluation metrics.

Let A_{ij} be the accuracy generated from our ith algorithm from the jth Monte-Carlo simulation. We define the empirical distribution of accuracy results generated by algorithm i as

$$\hat{F}_i(x) = \frac{1}{n} \sum_{j=1}^{16} I_{A_{ij} \leq x}. \tag{16}$$

The test statistic is the supremum among the differences of the empirical distribution generated using empirical distribution (16) of i_1th and i_2th algorithm (**?**)

$$D = \sup_x |F_{i_1}(x) - F_{i_2}(x)| . \tag{17}$$

We compare the test statistics D with the critical value of 0.433, which corresponds to 16 samples and a 95% of confidence level. We reject the null hypotheses that the two algorithms produce the same accuracy A with 95% confidence level if the test statistics $D > 0.433$.

4 Experiments

4.1 Experiment Setting

To provide a more rigorous evaluation on performance, we train the model 16 times using each method on each dataset and calculate the mean and standard deviation of the results. We randomly sample 80% of the training data for training each model.

To reduce the running time, we use multi-threading programming and train the 16 models simultaneously.

As machine learning algorithm often involve a lot of hyper-parameter tuning, thus time and computational consuming. To better carry out the experiments, we empl Linux server with high standard configuration to run our code. Here's the configuration detail.

talk about bootstrap resamples of 8000 sample from 10000

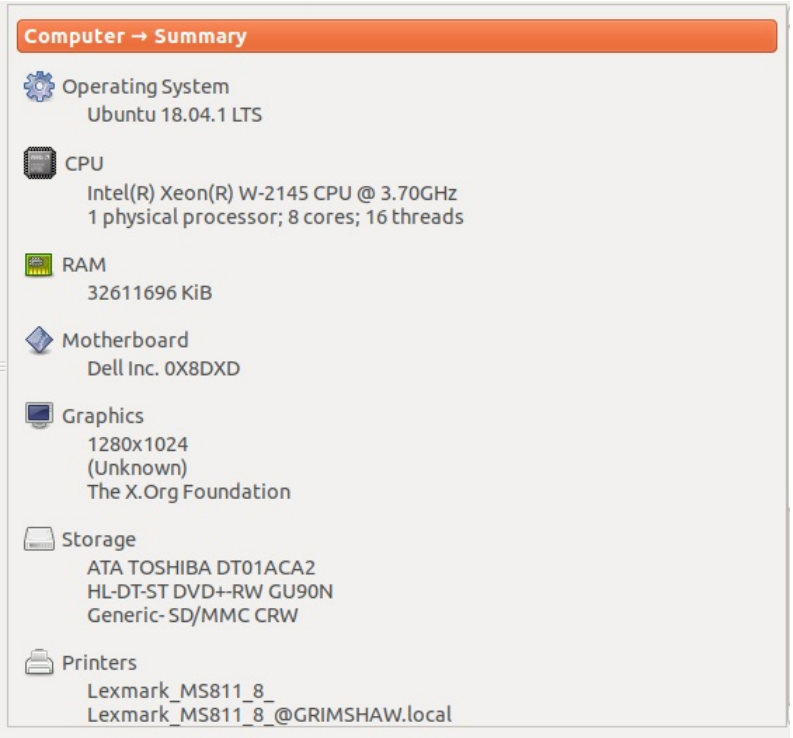

Figure 1: HPC

4.2 Compare Images

As mentioned previously, we preprocess the image dataset by using scaler and PCA. Here are some examples from the CIFAR dataset to compare the effects intuitively.

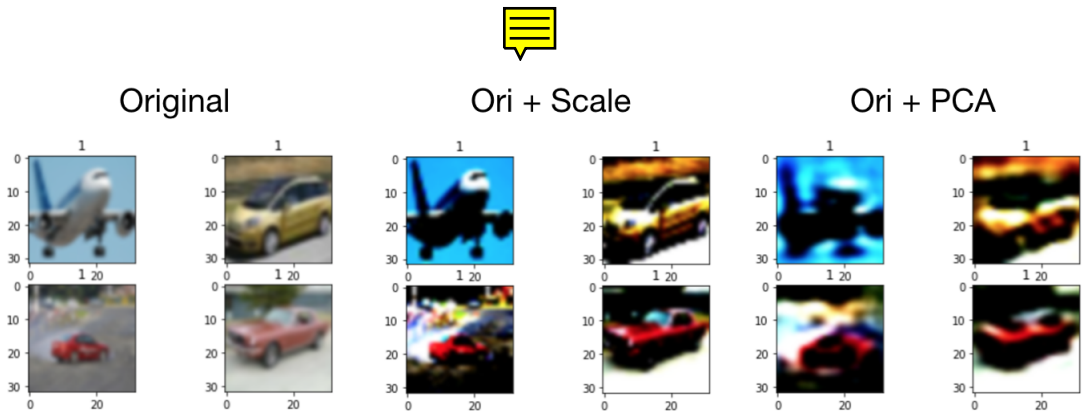

Figure 2: Comparison between original images and processed ones

Table 1: Mean sd

Data set	Algorithm	Mean	Standard deviation	Confidence interval	Average running time
		0.938	0.003	(0.936,0.939)	
		0.942	0.003	(0.941,0.944)	
		0.94	0.003	(0.939,0.942)	
		0.835	0.004	(0.833.0.837)	
		0.832	0.008	(0.829,0.836)	
		0.844	0.005	(0.841,0.846)	

We can tell from the figure above that the scale process somehow "enlighten" the original picture to some extent so that the differences in "lightness" are diminished. However, the reconstructed images from PCA are less meaningful? of course, it discarded 15% of the information

4.3 Discuss flip rate

Both datasets have flip rates $\rho_{-1} = 0.2$ and $\rho_{+1} = 0.4$. TODO

4.4 Hayperparameter Tuning

Please copy the hyperparameters from script to here.

4.5 Compare Results

Table **??** records the descriptive statistics of 16 Monte Carlo simulations for the three algorithms detailed in Sections 3.4, 3.5 and 3.6. We test each of the algorithms with data sets MNIST and CIFAR.

4.5.1 Expectation Maximisation is the fastest

The last column of Table **??** illustrates the running time of the three algorithms on the two data sets. The method of Expectation Maximisation is more than four times faster than the other two methods. There are two causes for the difference in speed. Firstly, the method of Expectation Maximisation only requires to train the base SVM model once. In contrast to Expectation Maximisation, the other two methods all require a pre-training step that predicts the conditional probabilities $P(S|X)$. Secondly and most importantly, the loss function (9) knows the structure of the label noises. Hence, it converges much faster than the other two methods. In this pre-training step, the method of relabelling and the method of Importance Reweighting have no adequate information about the CCN. The lack of CCN information slows down the convergence of the algorithms.

Table 2: Hypothesis test

Data	H_0	D	P-value	Test result
MNIST	$A_1 = A_2$	0.3125	0.4154	Fail to reject
	$A_1 = A_3$	0.625	0.0038	Reject
	$A_2 = A_3$	0.3125	0.4154	Fail to reject
CIFAR	$A_1 = A_2$	0.6875	0.0010	Reject
	$A_1 = A_3$	0.5	0.0366	Reject
	$A_2 = A_3$	0.6875	0.0010	Reject

4.5.2 CIFAR is more difficult to classify

In terms of accuracy, all algorithms perform significantly better classifying the images in MNIST in comparison with classifying images in CIFAR. A possible explanation for this might be that CIFAR is much more complex in nature—it requires 100 principal components to reconstruct 85% of the image information, where as MNIST data set only needs 40 such principal components. This observation are also visualised box plot in Figure **??**. This figure plots the accuracy of the three algorithms against running time using box plot. For the MNIST data set, it seems that the relabelling method are marginally more accurate than the other two algorithms. However, for the more complex CIFAR data set, the Importance reweighting seems to be significantly more accurate.

4.5.3 Heuristic approach is inconsistent

For the CIFAR data set, both the high inter quarter range (IQR) in Figure **??** and high standard deviation in Table **??** suggest that the method of relabelling gives inconsistent classification results. This inconsistency may be due to the heuristic nature of the method as described in Section 3.6. No asymptotic properties were prove for the process of relabelling. In contrast, Expectation Maximisation method and Importance Reweighting method are theoretically justified. For example, Section 3.4 mathematically derives the choice of the loss function (9) by Expectation Maximisation. Similarly, **?** give rigorous proofs justifying the properties of the Importance Reweighting formula (12). As a result of solid theoretical support, these two methods produce more consistent results.

The inconsistency is not observed in dataset MNIST. This may partly be explained by the complexity of the data set. For the MNIST data (784 dimensional), a sample of $8,000$ images is large. However, for the CIFAR data, the sample size of $8,000$ is on a similar scale with the dimensionality (3072).

4.5.4 Hypothesis test justifies observations

Kolmogorov-Smirnov test statistic (**??**) confirms the visualisations observed from the box plot **??**. As illustrated in Table **??**, the test statistic ($D = 0.625 > 0.433$) suggests that the relabelling method (A_3) more accurately classifies the MNIST data set with label noise, comparing with the Expectation Maximisation method (A_1). However, the hypothesis test are not able to conclude whether Importance Reweighting (A_2) outperforms Expectation Maximisation (A_3) with these 16 simulations of accuracy results, for MNIST data set.

In terms of CIFAR data set, there are strong statistical evidences suggesting Importance Reweighting method classifies the images in the noisy CIFAR data set more accurately than the expectation maximisation method. Similarly, and we are also 95% confident to accept the hypothesis that the Expectation Maximisation method is more accurate than the importance relabelling method.

Density plot (**??**) of the accuracy results visualise these results concluded by Kolmogorov-Smirnov hypothesis tests.

4.6 Discussion

Please write these in full good English sentences.

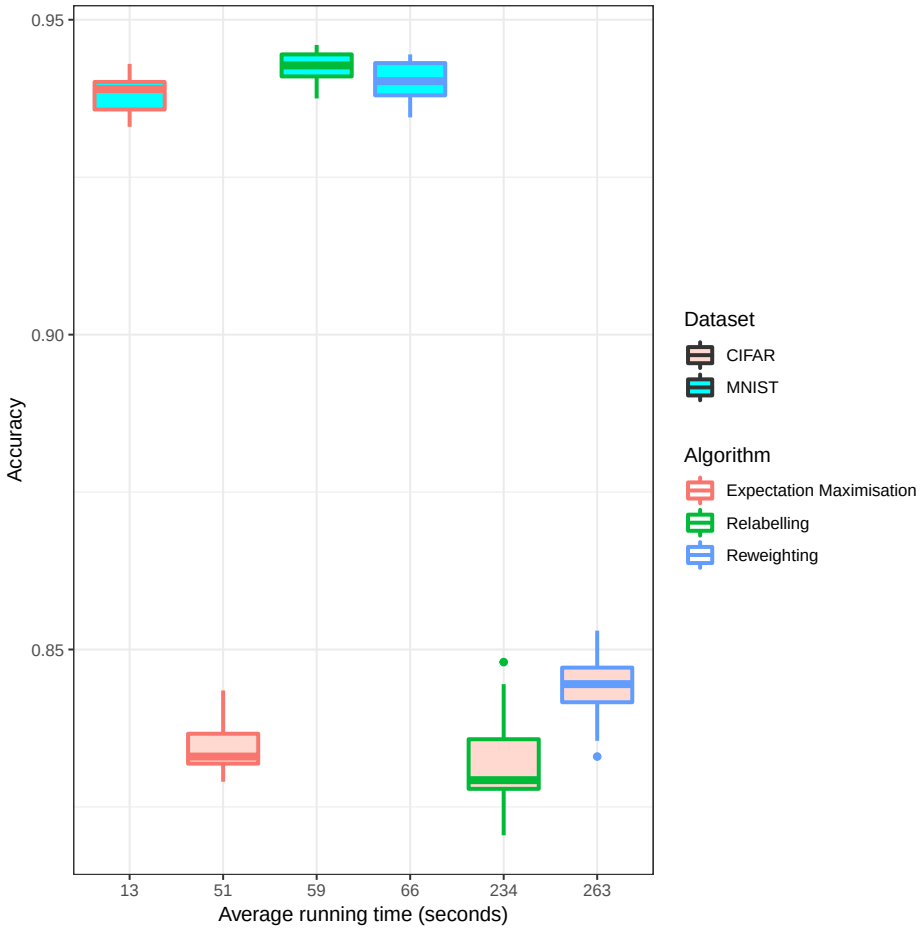

Figure 3: Box plots of the accuracy results against running time. Describe various lines and the dots as well as colours here.

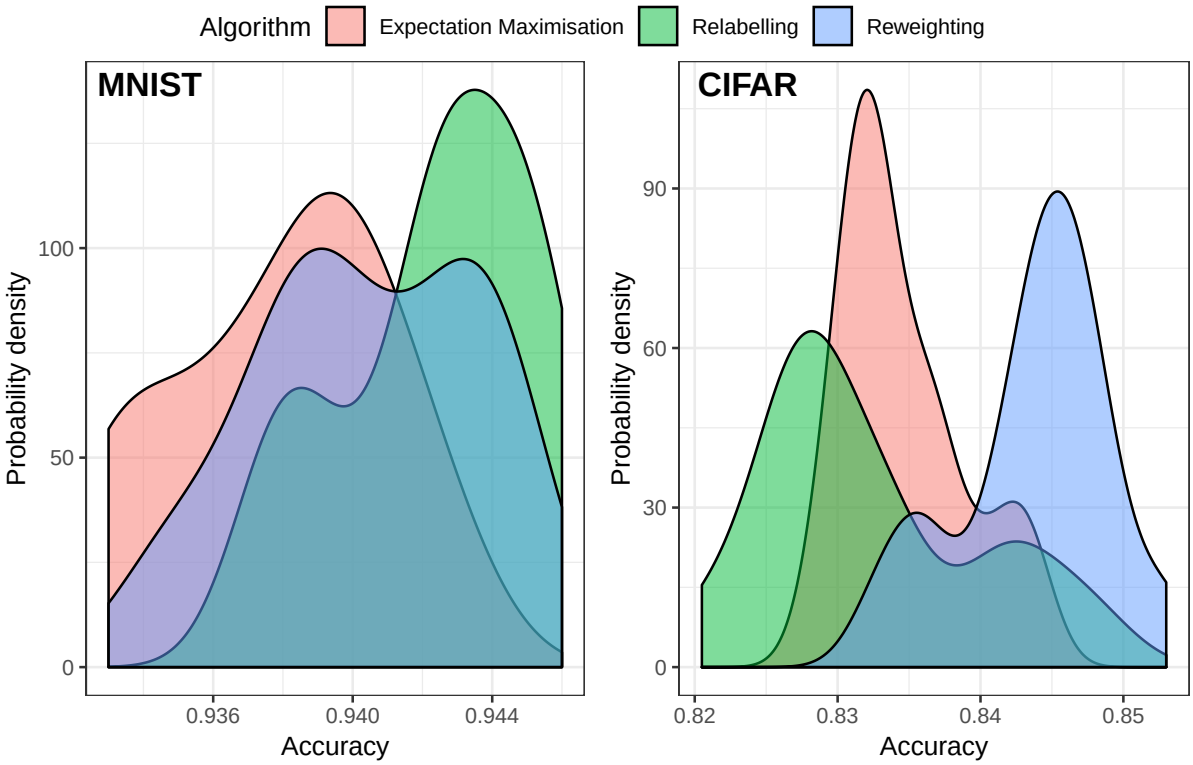

Figure 4: Density function from kernel smoothing

EM is fast, require good estimate of ρ. Need mathematical derivation of objective function. Derivation depends on base method and error distribution, needs mathematical skills. Low accuracy.

Relabelling no need to know ρ. Select a region where $P(S = g(x))$ is high where $g(x)$ is the decision function (2). However, in consistent result. Only useful with large data.

Reweighting is the best overall, more accurate and consistent.

4.7 Backup Statements

compare photos discuss pca heruristic no need to know rho.

please plot the cifar with and without PCA, put them into one graph and compare. codes are already available. just run them and save and insert.

We use the methods,3.4, 3.6 and 3.5, to conduct experiments on two datasets, fashion-mnist and CIFAR. Both datasets are re-organized to have two classes 0 and 1. For clearer notation, we refer to class 0 as -1 and class 1 as +1. Both datasets have flip rates $\rho_{-1} = 0.2$ and $\rho_{+1} = 0.4$. The fashion-mnist has dimension $d = 784$, while CIFAR has $d = 3072$. original image As mentioned in section 3.1, we need to run PCA on CIFAR to reduce the dimensionality. pca image. decribe the difference Then we scale the image standardised image. decribe the difference

5 Conclusion

TODO.

References

Wolfgang Härdle and Léopold Simar. *Applied multivariate statistical analysis*, volume 22007. Springer, 2007.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Proceedings of the Asian Conference on Machine Learning*, volume 20, pages 97–112. PMLR, 2011.

Carla E Brodley, Mark A Friedl, et al. Identifying and eliminating mislabeled training instances. In *Proceedings of the National Conference on Artificial Intelligence*, pages 799–805, 1996.

Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2016.

Chi Jin and Liwei Wang. Dimensionality dependent pac-bayes margin bound. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1034–1042. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4500-dimensionality-dependent-pac-bayes-margin-bound.pdf.

Matthias Seeger. Pac-bayesian generalisation error bounds for gaussian process classification. *J. Mach. Learn. Res.*, 3:233–269, March 2003. ISSN 1532-4435. doi: 10.1162/153244303765208386. URL https://doi.org/10.1162/153244303765208386.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995. ISSN 1573-0565. doi: 10.1007/BF00994018.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, 15(1):3133–3181, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2697065.

Benoît Frénay 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.

Pengyi Yang, John T Ormerod, Wei Liu, Chendong Ma, Albert Y Zomaya, and Jean YH Yang. Adasampling for positive-unlabeled and label noise learning with bioinformatics applications. *IEEE Transactions on Cybernetics*, (99):1–12, 2018.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM. ISBN 0-89791-497-X. doi: 10.1145/130385.130401. URL http://doi.acm.org/10.1145/130385.130401.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Kenneth Jonsson, Josef Kittler, YP Li, and Jiri Matas. Support vector machines for face authentication. *Image and Vision Computing*, 20(5-6):369–375, 2002.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

Tingfan Wu, ChihJen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2003.

Christian Walck. *Hand-book on statistical distributions for experimentalists*. 1996. URL http://www.fysik.su.se/~walck/suf9601.pdf.