PREDICTING CLASSIFICATION ACCURACY WHEN ADDING NEW UNOBSERVED CLASSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Multiclass classifiers are often designed and evaluated only on a sample from the classes on which they will eventually be applied. Hence, their final accuracy remains unknown. In this work we study how a classifier's performance over the initial class sample can be used to extrapolate its expected accuracy on a larger, unobserved set of classes. For this, we define a measure of separation between correct and incorrect classes that is independent of the number of classes: the *reversed ROC* (rROC), which is obtained by replacing the roles of classes and data-points in the common ROC. We show that the classification accuracy is a function of the rROC in multiclass classifiers, for which the learned representation of data from the initial class sample remains unchanged when new classes are added. Using these results we formulate a robust neural-network-based algorithm, *CleaneX*, which learns to estimate the accuracy of such classifiers on arbitrarily large sets of classes. Our method achieves remarkably better predictions than current state-of-the-art methods on both simulations and real datasets of object detection, face recognition, and brain decoding.

1 INTRODUCTION

Advances in machine learning and representation learning led to automatic systems that can identify an individual class from very large candidate sets. Examples are abundant in visual object recognition (Russakovsky et al., 2015; Simonyan & Zisserman, 2014a), face identification (Liu et al., 2017b), and brain-machine interfaces (Naselaris et al., 2011b; Seeliger et al., 2018). In all of these domains, the possible set of classes is much larger than those observed at training or testing. Increasing the number of classes changes the difficulty of the classification problem, and the magnitude of change varies depending on the classification algorithm and the interactions between the classes. It is therefore hard for researchers to extrapolate results from smaller class samples onto larger sets.

In large multiclass classification tasks, a representation is often learned on a set of k_1 classes, whereas the classifier is eventually used on a new larger class set. On the larger set, classification can be performed by applying simple procedures such as measuring the distances in an embedding space between the new example $x \in \mathcal{X}$ and labeled examples associated with the classes $y_i \in \mathcal{Y}$. Such classifiers, where the score assigned to a data point x to belong to a class y is independent of the other classes, are defined as *marginal classifiers* (Zheng et al., 2018). Their performance on the larger set describes how robust the learned representation is. Examples of classifiers marginal classifier when used on the larger class set include siamese neural networks (Chopra et al., 2005), one-shot learning (Fei-Fei et al., 2006) and approaches that directly optimize the embedding (Schroff et al., 2015). Our goal in this work is to estimate how well marginal classifiers will perform on a large unobserved set of k_2 classes, based on their performance on a smaller set of k_1 classes.

Recent works (Zheng & Benjamini, 2016; Zheng et al., 2018) set a probabilistic model for rigorously studying this problem, assuming that the k_1 available classes are sampled from the same distribution as the larger set of k_2 classes. Following the framework they propose, we assume that the sets of k_1 and k_2 classes on which the classifier is trained and evaluated are sampled independently from a set $\mathcal{Y}$ according to $y_i \sim P_Y(y)$, and for each class, r data points are sampled independently from $\mathcal{X}$ according to the conditional distribution $P_{X|Y}(x \mid y)$. In their work, the authors presented two

methods for predicting the expected accuracy, one of them originally due to Kay et al. (2008). We cover these methods in Section 2.

As a first contribution of this work (Section 3), we provide a theoretical analysis that connects the accuracy of marginal classifiers to a variant of the receiver operating characteristic (ROC) curve, which is achieved by reversing the roles of classes and data points in the common ROC. We show that the *reversed ROC* (rROC) measures how well a classifier's learned representation separates the correct from the incorrect classes of a given data point. We then prove that the accuracy of marginal classifiers is a function of the rROC, allowing the use of well researched ROC estimation methods (Gonçalves et al., 2014; Bhattacharya & Hughes, 2015) to predict the expected accuracy. Furthermore, the reversed area under the curve (rAUC) equals the expected accuracy of a binary classifier, where the expectation is taken over all randomly selected pairs of classes.

We use our results regarding the rROC to provide our second contribution (Section 4): *CleaneX* (CLassification Expected Accuracy Neural EXtrapolation), a new neural-network-based method for predicting the expected accuracy of a classifier when it will be applied on an arbitrary number of classes. In Sections 5 and 6 we compare the results obtained by our method to the ones obtained by previous methods. We show that CleaneX achieves remarkably better predictions of the expected accuracy by achieving both lower prediction errors overall, and lower estimation bias and variance. An immediate implication of this improvement is that it provides a tool that can now be used by practitioners.

1.1 Preliminaries and notation

In this work x are data points, y are classes, and when referred to as random variables they are denoted by X, Y respectively. We denote by $y(x)$ the correct class of x, and use y^* when x is implicitly understood. Similarly, we denote by y' an incorrect class of x.

We assume that for each x and y the classifier h assigns a score $S_y(x)$, such that the predicted class of x is $\arg\max_y S_y(x)$. On a given dataset of k classes, $\{y_1, \ldots, y_k\}$, the balanced accuracy of the trained classifier h is the probability that it assigns the highest score to the correct class

$$\mathcal{A}(y_1, \ldots, y_k) = P_X(S_{y^*}(x) \geq max_{i=1}^{k} S_{y_i}(x)) \tag{1}$$

where P_X is the distribution of the data points x in the sample of classes $y_1, \ldots, y_k$. Since r data points are sampled from each class, P_X assumes a uniform distribution over the classes within the given sample.

An important quantity for a data point x is the probability of the correct class to outscore a randomly chosen incorrect one:

$$C_x = P_{Y'}(S_{y^*}(x) \geq S_{y'}(x)). \tag{2}$$

This is the cumulative distribution function (CDF) of the incorrect scores, evaluated at the value of the correct score.

We denote the expected accuracy over all possible subsets of k classes from $\mathcal{Y}$ by $\mathbb{E}_k[\mathcal{A}]$ and its estimator by $\hat{\mathbb{E}}_k[\mathcal{A}]$. Given a sample of k_1 classes, the average accuracy over all subsets of k classes from the sample, is denoted by $\bar{\mathcal{A}}_k^{k_1}$. We refer to the curve of $\mathbb{E}_k[\mathcal{A}]$ at different values of $k \geq 2$ as the *accuracy curve*.

2 Related work

Learning theory provides bounds for multiclass classification that depend on the number of classes (Shalev-Shwartz & Ben-David, 2014), and the extension to large mutliclass problems is a topic of much interest (Kuznetsov et al., 2014; Lei et al., 2015; Li et al., 2018). However, these bounds cannot be used to estimate the expected accuracy. Generalization to out-of-label set accuracy includes the work of Jain & Learned-Miller (2010). The generalization of classifiers from datasets with few classes to larger class sets include those of Oquab et al. (2014) and Griffin et al. (2007), and are closely related to transfer learning (Pan et al., 2010) and extreme classification (Liu et al., 2017a). More specific works include that of Abramovich & Pensky (2019), which provides lower and upper bounds for the distance between classes that is required in order to achieve a given accuracy.

Kay et al. (2008), as adapted in (Zheng et al., 2018), propose to estimate the accuracy of a marginal classifier on a given set of k classes by averaging over x the probability that its correct class outscores a single random incorrect class, raised to the power of $k-1$ (the number of incorrect classes in the sample), that is

$$\mathbb{E}_k[\mathcal{A}] = \mathbb{E}_X[P_{Y'}(S_{y^*}(x) \geq S_{y'}(x))^{k-1}] = \mathbb{E}_x[C_x^{k-1}]. \tag{3}$$

Therefore, the expected accuracy can be predicted by estimating the values of C_x on the available data. To do so, the authors propose using kernel density estimation (KDE) choosing the bandwidth with pseudo-likelihood cross-validation (Cao et al., 1994).

Zheng et al. (2018) define a discriminability function

$$D(u) = P_X\left(S_{y^*}(x) > P_{Y'}\left(S_{y'}(x)\right) \leq u\right), \tag{4}$$

and show that for marginal classifiers, the expected accuracy at k classes is given by

$$\mathbb{E}_k[\mathcal{A}] = 1 - (k-1)\int_0^1 D(u)u^{k-2}du. \tag{5}$$

The authors assume a non-parametric regression model with pre-chosen basis functions b_j, so that $D(u) = \sum_j \beta_j b_j$. To obtain $\hat{\beta}$ the authors minimize the mean squared error (MSE) between the resulting estimation $\hat{\mathbb{E}}_k[\mathcal{A}]$ and the observed accuracies $\bar{\mathcal{A}}_k^{k_1}$.

3 Reversed ROC

In this section we show that the expected accuracy, $\mathbb{E}_k[\mathcal{A}]$, can be better understood by studying an ROC-like curve. To do so, we first recall the definition of the common ROC: for two classes in a setting in which one class is considered as the positive class and the other as the negative one, the ROC is defined as the graph of the true-positive rate (TPR) against the false-positive rate (FPR) (Fawcett, 2006). The common ROC curve represents the separability that a classifier h achieves between data points of the positive class and those of the negative one. At a working point where the FPR of the classifier is u, we have $\text{ROC}(u) = \text{TPR}\left(\text{FPR}^{-1}(u)\right)$.

In a multiclass setting, we can define ROC_y for each class y by considering y as the positive class, and the union of all other classes as the negative one. An adaptation of the ROC for this setting can be defined as the expectation over the classes of ROC_y, that is $\overline{\text{ROC}}(u) = \int_{\mathcal{Y}} \text{ROC}_y(u)\, dP(y)$. In terms of classification scores, we have $\text{TPR}_y(t) = P_X(S_y(x) > t \mid y(x) = y)$, $\text{FPR}_y(t) = P_X(S_y(x) > t \mid y(x) \neq y)$ and thus $\text{FPR}_y^{-1}(u) = \sup_t \{P_X(S_y(x) > t \mid y(x) \neq y) \geq u\}$.

Here, we single out each time one of the classes y and compare the score of the data points that belong to this class with the score of those that do not. However, when the number of classes is large, we could instead single out a data point x and compare the score that it gets for the correct class with the scores for the incorrect ones. This reverse view is formalized in the following definition, where we exchange the roles of data points x and classes y, to obtain the reversed ROC:

Definition 1. *Given a data point x, its corresponding* reversed true-positive *rate is*

$$\textit{rTPR}_x(t) = \begin{cases} 1 & S_{y^*}(x) > t \\ 0 & S_{y^*}(x) \leq t \end{cases} \tag{6}$$

The reversed false-positive *rate is*

$$\textit{rFPR}_x(t) = P_{Y'}\left(S_{y'}(x) > t\right) \tag{7}$$

and accordingly

$$\textit{rFPR}_x^{-1}(u) = \sup_t \left\{P_{Y'}\left(S_{y'}(x) > t\right) \geq u\right\}. \tag{8}$$

Consequently, the reversed ROC *is*

$$\textit{rROC}_x(u) = \textit{rTPR}_x\left(\textit{rFPR}_y^{-1}(u)\right) = \begin{cases} 1 & S_{y^*}(x) > \sup_t \left\{P_{Y'}\left(S_{y'}(x) > t\right) \geq u\right\} \\ 0 & \textit{otherwise} \end{cases} \tag{9}$$

and the average reversed ROC *is*

$$\overline{\textit{rROC}}(u) = \int_{\mathcal{X}} \textit{rROC}_x(u)\, dP(x). \tag{10}$$

Remark. *Note that $dP(x)$ in Equation 10 assumes a uniform distribution with respect to a given sample of classes $\{y_1, \ldots, y_k\}$ and their corresponding data points.*

Since $P_{Y'}(S_{y'}(x) > t)$ is a decreasing function of t, it can be seen that $\text{rROC}_x(u) = 1$ iff $u > P_{Y'}(S_{y'}(x) > S_{y^*}) = 1 - C_x$ (cf. Proposition 1 in Appendix A). However, even though rROC_x is a step function, the $\overline{\text{rROC}}$ resembles a common ROC curve, as illustrated in Figure 1.

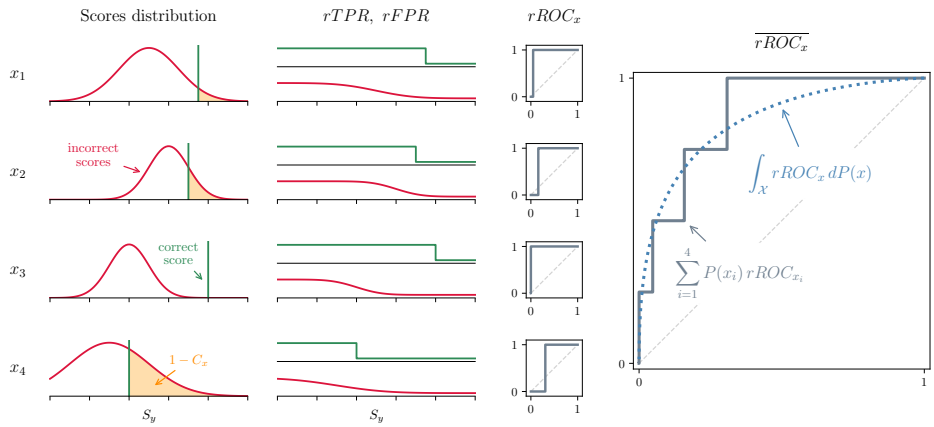

Figure 1: The reversed ROC. The leftmost column shows an example of the score distributions of four data points. The distribution of scores of incorrect classes is shown in red, and the score of the correct class is indicated by a green line. The yellow shaded area is the complement of the CDF of the incorrect scores distribution evaluated at the correct score, that is $1 - C_x$. The second column shows the corresponding rTPR (green, top) and rFPR (red, bottom). The third column depicts the resulting rROC_x curves. Finally, the rightmost plot presents the average rROC over the four data points (solid grey); as the number of averaged data points grows, the $\overline{\text{rROC}}$ curve becomes smoother (dotted blue).

3.1 The reversed ROC and classification accuracy

In what follows, we show that the classification accuracy can be expressed using the average reversed ROC curve. We assume a marginal classifier which assigns scores without ties, that is for all x and all $y_i \neq y_j$ we have $S_{y_i}(x) \neq S_{y_j}(x)$ almost surely. In such cases the following theorem holds:

Theorem 1. *The expected balanced classification accuracy at k classes is*

$$\mathbb{E}_k[\mathcal{A}] = 1 - (k-1) \int_0^1 \left(1 - \overline{rROC}\,(1-u)\right) u^{k-2} du. \tag{11}$$

To prove this theorem we show that

$$1 - \overline{\text{rROC}}(1-u) = P_X\left(P_{Y'}\left(S_{y^*}(x) > S_{y'}(x)\right) \leq u\right) = D(u) \tag{12}$$

and the rest follows immediately from the results of Zheng et al. (2018) (see Equation 5). We provide the detailed proof in Appendix A.

Now, using the properties of the rROC we get

$$\begin{aligned}
\mathbb{E}_k[\mathcal{A}] &= 1 - (k-1) \int_0^1 \left(1 - \overline{\text{rROC}}\,(1-u)\right) u^{k-2} du \\
&= \int_0^1 \left(\int_{\mathcal{X}} \text{rROC}_x(1-u) dP(x) \right) (k-1) u^{k-2} du \\
&= \int_{\mathcal{X}} \int_0^{C_x} (k-1) u^{k-2} du dP(x) = \int_{\mathcal{X}} C_x^{k-1} dP(x) = \mathbb{E}_x[C_x^{k-1}].
\end{aligned} \tag{13}$$

Therefore, in order to predict the expected accuracy it suffices to estimate the values of C_x.

A consequence of the result above is that the expressions that Kay et al. (2008) and Zheng et al. (2018) estimate (Equations 3 and 5 respectively) are in fact the same. Nevertheless, their estimation methods differ significantly.

Finally, we note that the theoretical connection between the reversed ROC and the evaluation of classification models extends beyond this particular work. For example, by plugging $k = 2$ into Theorem 1 it immediately follows that the area under the reversed ROC curve (rAUC) is the expected balanced accuracy of two classes:

$$\text{rAUC} := \int_0^1 \overline{\text{rROC}}\,(u)\,du = \mathbb{E}_2[\mathcal{A}]. \tag{14}$$

4 Expected accuracy prediction

In this section we present a new algorithm, *CleaneX*, for the prediction of the expected accuracy of a classifier. The algorithm is based on a neural network that estimates the values of C_x using the classifier's scores on data from the available k_1 classes. These estimations are then used, based on the results of the previous section, to predict the classification accuracy at $k_2 > k_1$ classes.

The general task of estimating densities using neural networks has been widely addressed (Magdon-Ismail & Atiya, 1999; Papamakarios et al., 2017; Dinh et al., 2016; Uria et al., 2014). However, in our case, we need to estimate the cumulative distribution only at the value of the correct score $S_{y^*}(x)$. This is an easier task to perform and it allows us to design an estimation technique that learns to estimate the CDF in a supervised manner, using the average accuracies $\bar{\mathcal{A}}_k^{k_1}$ for $2 \leq k \leq k_1$ as targets. The proposed method is described in Algorithm 1.

Algorithm 1: CleaneX

Input : The classifier's score function S, a training set of N examples x from the set of k_1 available classes, target number of classes k_2; a feedforward neural network $f(\cdot;\theta)$, initial network weights θ_0, number of training iterations J, learning rate η

Output: Estimated accuracy at k_2 classes

for $k = 2, \ldots, k_1$ **do**
  Compute $\bar{\mathcal{A}}_k^{k_1}$
end
for *each x in training set* **do**
  Set $S'(x) \leftarrow \left(S_{y'_1}(x), \ldots, S_{y'_{k_1-1}}(x)\right)$
  Sort $S'(x)$
  Set $S_x \leftarrow (S_{y^*}(x), S'(x))$
end
for $j = 0, \ldots, J$ **do**
  for *each x* **do**
    Set $\hat{C}_x \leftarrow f(S_x;\theta_j)$
  end
  Update network parameters performing a gradient descent step

$$\theta_{j+1} \leftarrow \theta_j - \eta \nabla_\theta \left(\sum_{k=2}^{k_1} \left(\frac{1}{N} \sum_x \hat{C}_x^{\,k-1} - \bar{\mathcal{A}}_k^{k_1} \right)^2 \right)$$

end

Return $\hat{\mathbb{E}}_{k_2}[\mathcal{A}] = \frac{1}{N}\sum_x \hat{C}_x^{\,k_2-1}$;

Unlike KDE and non-parametric regression, our method does not require a choice of basis functions or kernels. As will be seen in the next sections, in all of our experiments we use the same network architecture and hyperparameters, indicating that our algorithm requires very little tuning when applied to new classification problems.

5 SIMULATION STUDIES

We compare CleaneX, under different parametric settings, to the KDE based method (Kay et al., 2008) and to the non-parametric regression (Zheng et al., 2018). We simulate both classes and data points as d-dimensional vectors, with $d = 5$ (and $d = 3, 10$ shown in Appendix B). Settings vary in the distribution of classes Y and data-points $X|Y$, and in the spread of data-points around the class centroids. We sample the classes $y_1, \ldots, y_{k_2}$ from a multivariate normal distribution $\mathcal{N}(0, I)$ or a multivariate uniform distribution $\mathcal{U}(-\sqrt{3}, \sqrt{3})$[1]. We sample $r = 10$ data points for each class, either from a multivariate normal distribution $\mathcal{N}(y, \sigma^2 I)$ or from a multivariate uniform distribution $\mathcal{U}(y - \sqrt{3\sigma^2}, y + \sqrt{3\sigma^2})$. the difficulty level is determined by $\sigma^2 = 0.1, 0.2$. The classification scores $S_y(x)$ are set to be the euclidean distances between x and y (in this case the correct scores are expected to be lower than the incorrect ones, entailing some straightforward modifications to our analysis). For each classification problem, we subsample 50 times k_1 classes, for $k_1 = 100, 500$, and predict the accuracy at $2 \leq k \leq k_2 = 2000$ classes.

For our method, we use in all the experiments an identical feed-forward neural network with two hidden layers of sizes 512 and 128, rectified linear activation between the layers, and sigmoid activation applied on the output. We train the network according to Algorithm 1 for $J = 10,000$ iterations with learning rate of $\eta = 10^{-4}$ using Adam optimizer (Kingma & Ba, 2014). For the regression based method we choose a radial basis and for the KDE based method a normal kernel, as recommended by the authors. The technical implementation details are provided in Appendix C.

We summarize the results in Figure 2, showing the distribution of the root mean integrated square error (RMSE), $\left(\frac{1}{k_2-1}\sum_{k=2}^{k_2}(\mathbb{E}_k[\mathcal{A}] - \hat{\mathbb{E}}_k[\mathcal{A}])^2\right)^{1/2}$, over 50 repetitions, that is 50 subsamples of k_1 classes. We show the resulting accuracy curves in Figure 4 in Appendix B.

The results show that extrapolation to 20 times the original number of classes can be achieved with reasonable errors (the median RMSE $< 5\%$ in almost all scenarios). Our method does better or similar to the competing methods, often with substantial gain. In contrast, the KDE method has consistently the worst performance. Looking at Figures 2 and 4, we see that the KDE curves have a strong bias. The regression results, on the other hand, are more variable than our method, especially at $k_1 = 100$. The additional results on $d = 3, 10$ (Appendix B) are consistent with these results, though all methods predict better for $d = 10$.

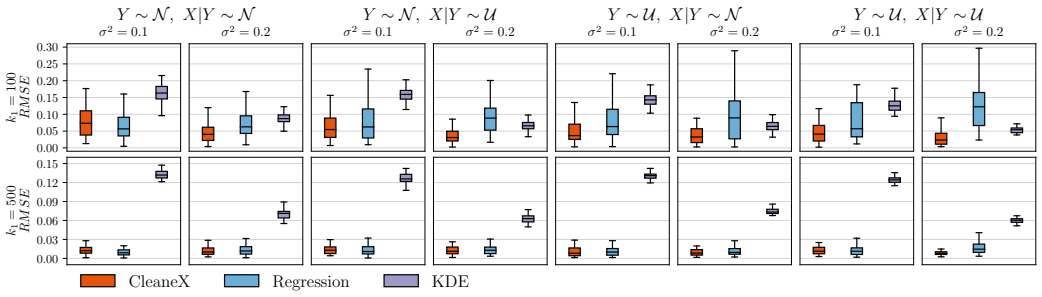

Figure 2: Simulation results. For each scenario we show a boxplot representing the RMSE values obtained over 50 repetitions using CleaneX (left box, orange), regression based method (middle box, blue) and KDE (right box, purple). The boxes extend from the lower to the upper quartile values, with a line at the median; whiskers show values at a distance of at most 1.5 IQR (interquartile range) from the lower and the upper quartiles; outliers are not shown.

6 EXPERIMENTS

In this section we present the results of three experiments performed on different datasets from the fields of computer vision and computational neuroscience. We repeat each experiment 50 times. In

[1]This results in covariances between the classes that equal to those in the multivariate normal case.

each repetition we sub-sample k_1 classes and predict the accuracy at $2 \leq k \leq k_2$ classes. In all the experiments we use the same network architecture as described in Section 5.

Experiment 1 - Object Detection (CIFAR-100) In this experiment we use the CIFAR dataset (Krizhevsky et al., 2009) that consists of 32×32 color images from 100 classes, each class containing 600 images. Each image is embedded into 512-dimensional space by a VGG-16 network (Simonyan & Zisserman, 2014b) which was pre-trained on the ImageNet dataset (Deng et al., 2009). On the training set, the centroid of each class is calculated and the classification scores for each image in the test set are set to be the distance of the image embedding from the centroids. The classification accuracy is extrapolated from $k_1 = 10$ to $k_2 = 100$ classes.

Experiment 2 - Face Recognition (LFW) We use the "Labeled Faces in the Wild" dataset (Huang et al., 2007) and follow the procedure described in Zheng et al. (2018): we restrict the dataset to the 1672 individuals for which it contains at least 2 face photos and include in our data exactly 2 randomly chosen photos for each person. We use one of them as a label y, and the other as a data point x, consistent with a scenario of single-shot learning. Each photo is embedded into 128-dimensional space using OpenFace embedding (Amos et al., 2016). The classification scores are set to be the euclidean distance between the embedding of each photo and the photos that are used as labels. Classification accuracy is extrapolated from $k_1 = 200$ to $k_2 = 1672$ classes.

Experiment 3 - Brain Decoding (fMRI) Here we analyze the "mind-reading" task described in (Kay et al., 2008). A subject watched $n = 1750$ natural images inside a functional MRI scanner, and a vector of $v = 1250$ neural responses was recorded for each image. The goal of decoding is to identify the correct image from the neural response vector. Similar to the decoder in (Naselaris et al., 2011a), we use $n_t = 750$ images and their response vectors to fit an embedding $g(\cdot)$ of images into the brain response space, and to estimate the residual covariance Σ. The remaining $n - n_t = k_2 = 1000$ examples are used as an evaluation set for $g()$. For image y and brain vector x, the score is then the negative Mahalanobis distance $-\|g(y) - x\|^2_\Sigma$. For each response vector, the image with the highest score is selected. The experiment is run on $k_1 = 200$ examples from the evaluation set, and the results are extrapolated to $k_2 = 1000$.

6.1 Experiment results

The results of all the experiments are shown in Figure 3. It can be seen that consistently with our simulations, kernel density estimation provides predictions with the smallest variability, but with higher bias. The regression has the highest variability and the resulting predicted curves do not necessarily follow the known exponential form, with some increasing curves. Our method produces less variance than the regression and smaller bias than kernel density estimation, and remarkably more reliable results overall.

7 Discussion

In this work we presented the reversed ROC and showed its connection to the accuracy of marginal classifiers. We used this result to develop a new method for accuracy prediction.

Analysis of the two previous methods for accuracy extrapolation reveals that each of them uses only part of the relevant information for the task. The KDE method estimates C_x based only on the scores, ignoring the observed accuracies of the classifier. Even if the estimates of C_x were unbiased, the exponentiation in $\mathbb{E}_x[C_x^{k-1}]$ introduces bias, and this is aggregated when k_1 is small and the estimation is noisy. We also found the method to be sensitive to monotone transformations of the scores, such as taking squared-root or logarithm. In contrast, the non-parametric regression based method uses pre-chosen basis functions to predict the accuracy curves, ignoring the distribution density of the scores. As a result, it does not necessarily preserve the correct form of those curves. Our method, on the other hand, combines the information used by both previous methods, with less restriction on the shape of the curve, and therefore consistently outperforms them.

Our simulations show that when the data is embedded in higher dimensions, the accuracy curves behave more regularly and prediction improves across methods. This matches theoretical results

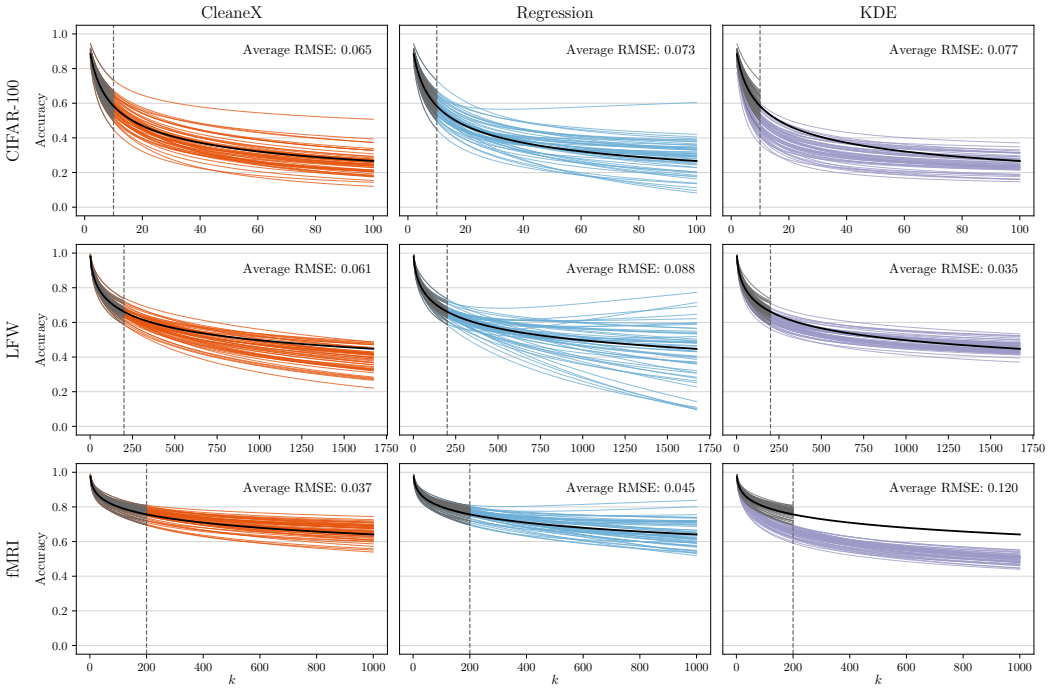

Figure 3: Comparison of predicted accuracy curves produced by CleaneX (left, orange), regression based method (middle, blue) and KDE (right, purple), on the three datasets. The dotted vertical lines denote k_1. The curves of $\bar{\mathcal{A}}_k^{k_1}$ for each repetition are shown in grey. The black curves correspond to $\mathbb{E}_k[\mathcal{A}]$ for $2 \leq k \leq k_2$. The average RMSE is taken over all 50 repetitions.

from (Zheng & Benjamini, 2016), which find that as d increases, the accuracy curves converge to a one-parameter family. In the real datasets, however, the one-parameter family does not effectively describe the observed curves, hence the importance of non-parametric estimation methods.

A considerable source of noise in the estimation is the selection of the k_1 classes. The $\bar{\mathcal{A}}_k^{k_1}$ curves diverge from $\mathbb{E}_k[\mathcal{A}]$ as the number of classes increases, and therefore it is hard to recover if the initial k_1 classes deviated from the true accuracy. (The effect of choosing different initial subsets can be seen comparing the grey curves and orange curves that continue them, for example in Figure 3.) We leave the design of more informative sampling schemes for future work.

Although our work focuses on marginal classifiers, its importance extends beyond this class of algorithms. First, preliminary results show that our method yields good estimates even when applied to (shallow) non-marginal classifiers such as multi-logistic regression. Moreover, if the representation can adapt as classes are introduced, we expect accuracy to exceed that of a fixed representation. We can therefore use our algorithm to measure the degree of adaptation of the representation. Generalization of our method to non marginal classifiers is a prominent direction for future work.

Acknowledgments

TBD

REFERENCES

Felix Abramovich and Marianna Pensky. Classification with many classes: challenges and pluses. *Journal of Multivariate Analysis*, 174:104536, 2019.

Brandon Amos, Bartosz Ludwiczuk, Mahadev Satyanarayanan, et al. Openface: A general-purpose face recognition library with mobile applications. *CMU School of Computer Science*, 6(2), 2016.

Bhaskar Bhattacharya and Gareth Hughes. On shape properties of the receiver operating characteristic curve. *Statistics & Probability Letters*, 103:73–79, 2015.

Ricardo Cao, Antonio Cuevas, and Wensceslao González Manteiga. A comparative study of several smoothing methods in density estimation. *Computational Statistics & Data Analysis*, 17(2):153–176, 1994.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pp. 539–546. IEEE, 2005.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

Luzia Gonçalves, Ana Subtil, M Rosário Oliveira, and P d Bermudez. Roc curve estimation: An overview. *REVSTAT–Statistical Journal*, 12(1):1–20, 2014.

Gregory Griffin, Alex Holub, and Pietro Perona. griffin2007caltech. Technical report, California Institute of Technology, 2007.

Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical report, UMass Amherst Technical Report, 2010.

Kendrick N Kay, Thomas Naselaris, Ryan J Prenger, and Jack L Gallant. Identifying natural images from human brain activity. *Nature*, 452(7185):352–355, 2008.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Vitaly Kuznetsov, Mehryar Mohri, and Umar Syed. Multi-class deep boosting. In *Advances in Neural Information Processing Systems*, pp. 2501–2509, 2014.

Yunwen Lei, Urun Dogan, Alexander Binder, and Marius Kloft. Multi-class svms: From tighter data-dependent generalization bounds to novel algorithms. In *Advances in Neural Information Processing Systems*, pp. 2035–2043, 2015.

Jian Li, Yong Liu, Rong Yin, Hua Zhang, Lizhong Ding, and Weiping Wang. Multi-class learning: From theory to algorithm. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 1586–1595. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7431-multi-class-learning-from-theory-to-algorithm.pdf`.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 115–124. ACM, 2017a.

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 212–220, 2017b.

Malik Magdon-Ismail and Amir F Atiya. Neural networks for density estimation. In *Advances in Neural Information Processing Systems*, pp. 522–528, 1999.

Thomas Naselaris, Kendrick N Kay, Shinji Nishimoto, and Jack L Gallant. Encoding and decoding in fmri. *Neuroimage*, 56(2):400–410, 2011a.

Thomas Naselaris, Kendrick N Kay, Shinji Nishimoto, and Jack L Gallant. Encoding and decoding in fmri. *Neuroimage*, 56(2):400–410, 2011b.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724, 2014.

Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Katja Seeliger, Matthias Fritsche, Umut Güçlü, Sanne Schoenmakers, J-M Schoffelen, SE Bosch, and MAJ Van Gerven. Convolutional neural network-based encoding and decoding of visual object recognition in space and time. *NeuroImage*, 180:253–266, 2018.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014a.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014b.

Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *International Conference on Machine Learning*, pp. 467–475, 2014.

Charles Zheng, Rakesh Achanta, and Yuval Benjamini. Extrapolating expected accuracies for large multi-class problems. *Journal of Machine Learning Research*, 19(65):1–30, 2018. URL `http://jmlr.org/papers/v19/17-701.html`.

Charles Y. Zheng and Yuval Benjamini. Estimating mutual information in high dimensions via classification error. *arXiv e-prints*, art. arXiv:1606.05229, Jun 2016.

A PROOF OF THEOREM 1

We begin by proving the following simple proposition:

Proposition 1. *Given x, let*

$$G_1 = \left\{ y | S_y(x) > \sup_t \{ P_{Y'}(S_{y'}(x) > t) \geq u \} \right\} \tag{15}$$

and

$$G_2 = \{ y \,| P_{Y'}(S_{Y'}(x) > S_y(x)) < u \}, \tag{16}$$

then $G_1 = G_2$.

Proof. Let G_1^c, G_2^c be the complement sets of G_1, G_2, respectively. Then

$$G_1^c = \{ y \mid S_y(x) \leq \sup_t \{ P_{Y'}(S_{y'}(x) > t) \geq u \} \} \tag{17}$$

and

$$G_2^c = \{ y \mid P_{Y'}(S_{y'}(x) > S_y(x)) \geq u \}. \tag{18}$$

Since $P_{Y'}(S_{y'}(x) \geq t)$ is decreasing in t we have

$$\begin{aligned} y \in G_1^c &\Leftrightarrow S_y(x) \leq \sup_t \{ P_{Y'}(S_{y'}(x) > t) \geq u \} \\ &\Leftrightarrow P_{Y'}(S_{y'}(x) > S_y(x)) \geq u \\ &\Leftrightarrow y \in G_2^c \end{aligned} \tag{19}$$

and therefore $G_1 = G_2$. □

We now prove Theorem 1, which was presented in Section 3:

Proof. We show that $1 - \overline{\text{rROC}}(1-u) = D(u)$, and the rest follows immediately according to Theorem 3 of Zheng et al. (2018) (see Equation 5):

$$\begin{aligned} \text{rROC}_x(1-u) &= 1 - \mathbb{1}\left[S_{y^*}(x) > \sup_t \{ P_{Y'}(S_{y'}(x) > t) \geq u \} \right] && \text{(Definition 1)} \\ &= 1 - \mathbb{1}\left[P_{Y'}(S_{y'}(x) > S_{y^*}(x)) < 1-u \right] && \text{(Proposition 1)} \\ &= \mathbb{1}\left[P_{Y'}(S_{y'}(x) \leq S_{y^*}(x)) \leq u \right] \\ &= \mathbb{1}\left[P_{Y'}(S_{y'}(x) < S_{y^*}(x)) \leq u \right], \end{aligned} \tag{20}$$

where $\mathbb{1}[\cdot]$ denotes the indicator function, and the last equality follows from the assumption that $S_{y'}(x) \neq S_y(x)$ almost surely. From here,

$$\begin{aligned} &1 - \overline{\text{rROC}}(1-u) \\ &= \mathbb{E}_X\left[\mathbb{1}\left(P_{Y'}(S_{y'}(x) < S_{y^*}(x)) \leq u \right) \right] \\ &= P_X\left(P_{Y'}(S_{y'}(x) < S_{y^*}(x)) \leq u \right) \\ &= D(u) \end{aligned} \tag{21}$$

as required. □

B SIMULATIONS - ADDITIONAL RESULTS

Figures 4 and 5 show the accuracy curves for which we presented summarized results in Figure 2.

In Figure 6 we provide additional simulation results for data in dimensions $d = 3, 10$. We simulate eight d-dimensional datasets of $k_2 = 2000$ classes and extrapolate the accuracy from $k_1 = 500$ classes. As in Section 5, the datasets are combinations of two distributions of Y and two of $X \mid Y$, in two different levels of classification difficulty. We sample the classes $y_1, \ldots, y_{k_2}$ from a multivariate normal distribution $\mathcal{N}(0, I)$ or a multivariate uniform distribution $\mathcal{U}(-1, 1)$. We sample $r = 10$ data points for each class from a multivariate normal distribution $N(y, \sigma^2 I)$ or from a multivariate

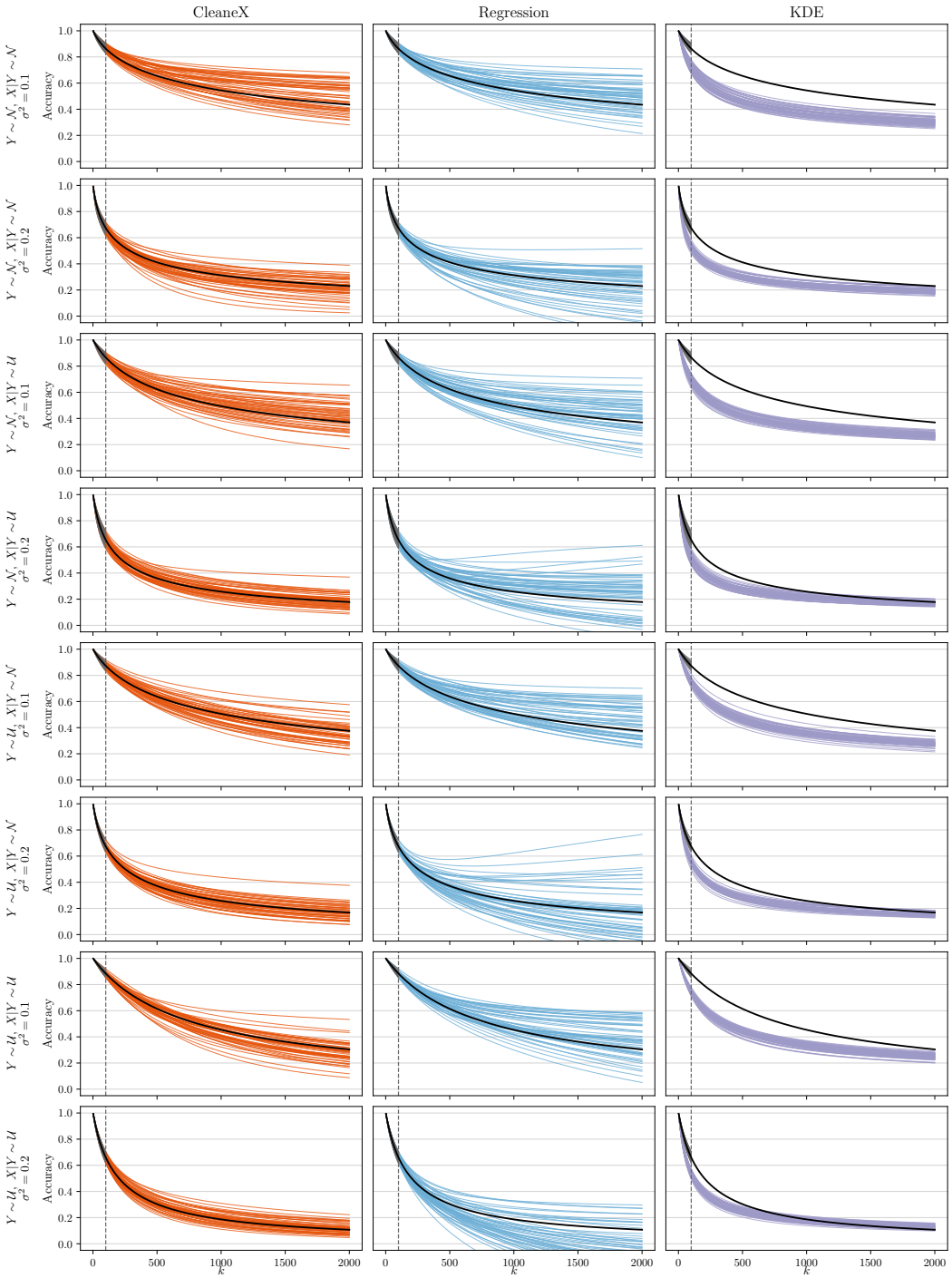

Figure 4: Comparison of predicted accuracy curves produced by CleaneX (left, orange), regression based method (middle, blue) and KDE (right, purple), on the eight simulated datasets with $d = 5$ and $k_1 = 100$ (dotted vertical line). The curves of $\bar{\mathcal{A}}_k^{k_1}$ for each repetition are shown in grey. The black curves correspond to $\mathbb{E}_k[\mathcal{A}]$ for $2 \leq k \leq k_2 = 2000$.

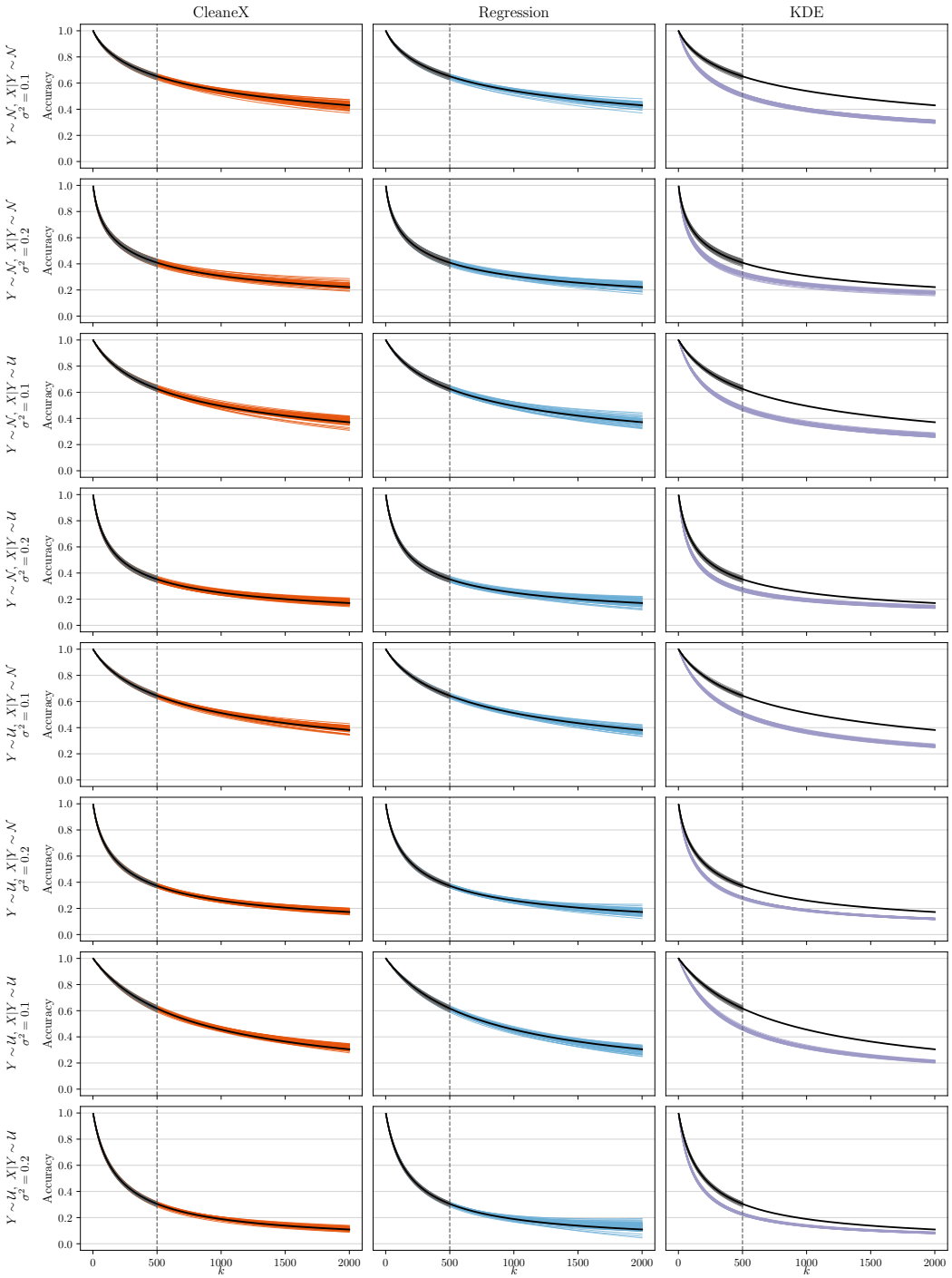

Figure 5: Comparison of predicted accuracy curves produced by CleaneX (left, orange), regression based method (middle, blue) and KDE (right, purple), on the eight simulated datasets with $d = 5$ and $k_1 = 500$ (dotted vertical line). The curves of $\bar{\mathcal{A}}_k^{k_1}$ for each repetition are shown in grey. The black curves correspond to $\mathbb{E}_k[\mathcal{A}]$ for $2 \leq k \leq k_2 = 2000$.

uniform distribution $U(y - \sigma^2, y + \sigma^2)$. For $d = 3$ we set σ^2 to 0.1 for the easier classification task and 0.2 for the difficult one. For $d = 10$, the values of σ^2 are set to 0.6 and 0.9. The classification scores $S_y(x)$ are the euclidean distances between x and y.

It can be seen that the results for $d = 3$ are consistent with those for $d = 5$. That is, our method outperforms the two competing methods, even with a bigger gap, in seven of the eight simulated datasets. For $d = 10$ the regression based method achieves compatible results with ours. As in the other simulations, our method produces less variance than the regression and less bias than KDE.

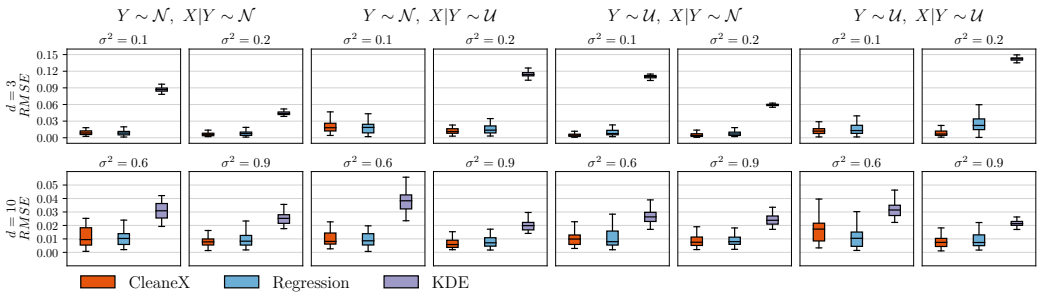

Figure 6: Additional simulation results. For each scenario we show a boxplot representing the RMSE values obtained over 50 repetitions using CleaneX (left box, orange), regression based method (middle box, blue) and KDE (right box, purple). The boxes extend from the lower to the upper quartile values, with a line at the median; whiskers show values at a distance of at most 1.5 IQR (interquartile range) from the lower and the upper quartiles; outliers are not shown.

C Implementation Details

All the code in this work was implemented in Python 3.6. For the CleaneX algorithm we used TensorFlow 1.14; for the regression based method we used the `scipy.optimize` package with the "Newton-CG" method; kernel density estimation was implemented using the `density` function from the `stats` library in R, imported to Python through the `rpy2` package.