TOWARDS UNDERSTANDING THE DATA DEPENDENCY OF MIXUP-STYLE TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

In the Mixup training paradigm, a model is trained using convex combinations of data points and their associated labels. Despite seeing very few true data points during training, models trained using Mixup seem to still minimize the original empirical risk and exhibit better generalization and robustness on various tasks when compared to standard training. In this paper, we investigate how these benefits of Mixup training rely on properties of the data in the context of classification. For minimizing the original empirical risk, we compute a closed form for the Mixup-optimal classification, which allows us to construct a simple dataset on which minimizing the Mixup loss can provably lead to learning a classifier that does not minimize the empirical loss on the data. On the other hand, we also give sufficient conditions for Mixup training to also minimize the original empirical risk. For generalization, we characterize the margin of a Mixup classifier, and use this to understand why the decision boundary of a Mixup classifier can adapt better to the full structure of the training data when compared to standard training. In contrast, we also show that, for a large class of linear models and linearly separable datasets, Mixup training leads to learning the same classifier as standard training.

1 INTRODUCTION

Mixup (Zhang et al., 2018) is a modification to the standard supervised learning setup which involves training on convex combinations of pairs of data points and their labels instead of the original data itself. In the original paper, Zhang et al. (2018) demonstrated that training deep neural networks using Mixup leads to better generalization performance, as well as greater robustness to adversarial attacks and label noise on image classification tasks. The empirical advantages of Mixup training have been affirmed by several follow-up works (He et al., 2019; Thulasidasan et al., 2019; Lamb et al., 2019; Arazo et al., 2019). The idea of Mixup has also been extended beyond the supervised learning setting, and been applied to semi-supervised learning (Berthelot et al., 2019; Sohn et al., 2020), privacy-preserving learning (Huang et al., 2021), and learning with fairness constraints (Chuang & Mroueh, 2021).

However, from a theoretical perspective, Mixup training is still mysterious even in the basic multi-class classfication setting – why should the output of a linear mixture of two training samples be the same linear mixture of their labels, especially when considering highly nonlinear models? Despite several recent theoretical results (Guo et al., 2019; Carratino et al., 2020; Zhang et al., 2020; 2021), there is still not a complete understanding of why Mixup training actually works in practice. In this paper, we try to understand *why* Mixup works by first understanding *when* Mixup works: in particular, how the properties of Mixup training rely on the structure of the training data.

We consider two properties for classifiers trained with Mixup. First, even though Mixup training does not observe many original data points during training, it usually can still correctly classify all of the original data points (empirical risk minimization (ERM)). Second, the aforementioned empirical works have shown how classifiers trained with Mixup often have better adversarial robustness and generalization than standard training. In this work, we show that both of these properties can rely heavily on the data used for training, and that they need not hold in general.

Main Contributions and Related Work. The idea that Mixup can potentially fail to minimize the original risk is not new; Guo et al. (2019) provide examples of how Mixup labels can conflict with actual data point labels. However, their theoretical results do not characterize the data and

model conditions under which this failure can provably happen when minimizing the Mixup loss. In Section 2 of this work, we provide a concrete classification dataset on which continuous approximate-minimizers of the Mixup loss can fail to minimize the empirical risk. We also provide sufficient conditions for Mixup to minimize the original risk, and show that these conditions hold approximately on standard image classification benchmarks.

With regards to generalization and robustness, the parallel works of Carratino et al. (2020) and Zhang et al. (2020) showed that Mixup training can be viewed as minimizing the empirical loss along with a data-dependent regularization term. Zhang et al. (2020) further relate this term to the adversarial robustness and Rademacher complexity of certain function classes learned with Mixup. In Section 3, we take an alternative approach to understanding generalization and robustness by analyzing the margin of Mixup classifiers. Our perspective can be viewed as complementary to that of the aforementioned works, as we directly consider the properties exhibited by a Mixup-optimal classifier instead of considering what properties are encouraged by the regularization effects of the Mixup loss. In addition to our margin analysis, we also show that for the common setting of linear models trained on high-dimensional Gaussian features both Mixup (for a large class of mixing distributions) and ERM with gradient descent learn the same classifier with high probability.

Finally, we note the related works that are beyond the scope of our paper; namely the many Mixup-like training procedures such as Manifold Mixup (Verma et al., 2019), Cut Mix (Yun et al., 2019), Puzzle Mix (Kim et al., 2020), and Co-Mixup (Kim et al., 2021).

2 Mixup and Empirical Risk Minimization

The goal of this section is to understand when Mixup training can also minimize the empirical risk. Our main technique for doing so is to derive a closed-form for the Mixup-optimal classifier over a sufficiently powerful function class, which we do in Section 2.2 after introducing the basic setup in Section 2.1. We use this closed form to motivate a concrete example on which Mixup training does not minimize the empirical risk in Section 2.3, and show under mild nondegeneracy conditions that Mixup will minimize the emprical risk in Section 2.4.

2.1 Setup

We consider the problem of k-class classification where the classes $1, ..., k$ correspond to compact disjoint sets $X_1, ..., X_k \subset \mathbb{R}^n$ with an associated probability measure $\mathbb{P}_X$ supported on $X = \bigcup_{i=1}^k X_i$. We use $\mathcal{C}$ to denote the set of all functions $g : \mathbb{R}^n \to [0,1]^k$ satisfying the property that $\sum_{i=1}^k g^i(x) = 1$ for all x (where g^i represents the i-th coordinate function of g). We refer to a function $g \in \mathcal{C}$ as a *classifier*, and say that g classifies x as class j if $j = \operatorname{argmax}_i g^i(x)$. The cross-entropy loss associated with such a classifier g is then:

$$J(g, \mathbb{P}_X) = -\sum_{i=1}^k \int_{X_i} \log g^i(x) d\mathbb{P}_X(x)$$

The goal of standard training is to learn a classifier $h \in \operatorname{argmin}_{g \in \mathcal{C}} J(g, \mathbb{P}_X)$. Any such classifier h will necessarily satisfy $h^i(x) = 1$ on X_i since the X_i are disjoint.

Mixup. In the Mixup version of our setup, we are interested in minimizing the cross-entropy of convex combinations of the original data and their classes. These convex combinations are determined according to a probability measure $\mathbb{P}_f$ whose support is $[0,1]$, and we assume this measure has a density f. For two points $s, t \in \bigcup_{i=1}^k X_i$, we let $z_{st}(\lambda) = \lambda s + (1-\lambda)t$ (and use z_{st} when λ is understood) and define the Mixup cross-entropy on s, t with respect to a classifier g as:

$$\ell_{mix}(g, s, t, \lambda) = \begin{cases} -\log g^i(z_{st}) & s, t \in X_i \\ -\left(\lambda \log g^i(z_{st}) + (1-\lambda) \log g^j(z_{st})\right) & s \in X_i, t \in X_j \end{cases}$$

Having defined ℓ_{mix} as above, we may write the component of the full Mixup cross-entropy loss corresponding to mixing points from classes i and j as:

$$J_{mix}^{i,j}(g, \mathbb{P}_X, \mathbb{P}_f) = \int_{X_i \times X_j \times [0,1]} \ell_{mix}(g, s, t, \lambda) \, d(\mathbb{P}_X \times \mathbb{P}_X \times \mathbb{P}_f)(s, t, \lambda)$$

We omit some, or all of, the arguments of $J^{i,j}_{mix}$ when they are clear from context. The final Mixup cross-entropy loss is then the sum of $J^{i,j}_{mix}$ over all $i, j \in \{1, ..., k\}$ (corresponding to all possible mixings between classes, including themselves):

$$J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f) = \sum_{i=1}^{k} J^{i,i}_{mix} + 2 \sum_{i=1}^{k} \sum_{j=i+1}^{k} J^{i,j}_{mix}$$

Where the coefficient of 2 in front of the second term comes from the fact that $J^{i,j}_{mix} = J^{j,i}_{mix}$ from Fubini's Theorem (we consider only classifiers g for which the ℓ_{mix} terms are defined and integrable).

Relation to Prior Work. We have opted for a more general definition of the Mixup loss (at least when constrained to multi-class classification) than prior works. This is not generality for generality's sake, but rather because many of our results apply to any mixing distribution supported on $[0, 1]$. One obtains the original Mixup formulation of Zhang et al. (2018) for multi-class classification on a finite dataset by taking the X_i to be finite sets, and choosing $\mathbb{P}_X$ to be the normalized counting measure (corresponding to a discrete uniform distribution). Additionally, $\mathbb{P}_f$ is chosen to have density $\mathrm{Beta}(\alpha, \alpha)$, where α is a hyperparameter.

2.2 Mixup-optimal Classifier

Given our setup, we now wish to characterize the behavior of a Mixup-optimal classifier at a point $x \in \mathbb{R}^n$. However, if the optimization of J_{mix} is considered over the class of functions $\mathcal{C}$, this is intractable (to the best of our knowledge) due to the lack of regularity conditions imposed on functions in $\mathcal{C}$. We thus wish to constrain the optimization of J_{mix} to a class of functions that is sufficiently powerful (so as to include almost all practical settings) while still allowing for local analysis. To do so, we will need the following definitions, which will also be referenced throughout the results in this section and the next:

$$\begin{aligned}
A^{i,j}_{x,\epsilon} &= \{(s, t, \lambda) \in X_i \times X_j \times [0, 1] : \ \lambda s + (1 - \lambda)t \in B_\epsilon(x)\} \\
A^{i,j}_{x,\epsilon,\delta} &= \{(s, t, \lambda) \in X_i \times X_j \times [0, 1 - \delta] : \ \lambda s + (1 - \lambda)t \in B_\epsilon(x)\} \\
X_{mix} &= \left\{ x \in \mathbb{R}^n : \ \bigcup_{i,j} A^{i,j}_{x,\epsilon} \text{ has positive measure for every } \epsilon > 0 \right\} \\
\xi^{i,j}_{x,\epsilon} &= \int_{A^{i,j}_{x,\epsilon}} d(\mathbb{P}_X \times \mathbb{P}_X \times \mathbb{P}_f)(s, t, \lambda) \\
\xi^{i,j}_{x,\epsilon,\lambda} &= \int_{A^{i,j}_{x}} \lambda d(\mathbb{P}_X \times \mathbb{P}_X \times \mathbb{P}_f)(s, t, \lambda)
\end{aligned}$$

The set $A^{i,j}_{x,\epsilon}$ represents all points in $X_i \times X_j$ that have lines between them intersecting an ϵ-neighborhood of x, while the set $A^{i,j}_{x,\epsilon,\delta}$ represents the restriction of $A^{i,j}_{x,\epsilon}$ to only those points whose connecting line segments intersect an ϵ-neighborhood of x with λ values bounded by $1 - \delta$. The set X_{mix} corresponds to all points for which every neighborhood factors into J_{mix}. The $\xi^{i,j}_{x,\epsilon}$ term represents the measure of the set $A^{i,j}_{x,\epsilon}$ while $\xi^{i,j}_{x,\epsilon,\lambda}$ represents the expectation of λ over the same set. To provide better intuition for these definitions, we provide visualizations in Section B of the appendix. We can now define the subset of $\mathcal{C}$ to which we will constrain our optimization of J_{mix}.

Definition 2.1. Let $\mathcal{C}^*$ to be the subset of $\mathcal{C}$ for which every $h \in \mathcal{C}^*$ satisfies $h(x) = \lim_{\epsilon \to 0} \operatorname{argmin}_{\theta \in [0,1]^k} J_{mix}(\theta)|_{B_\epsilon(x)}$ for all $x \in X_{mix}$ when the limit exists. Here $J_{mix}(\theta)|_{B_\epsilon(x)}$ represents the Mixup loss for a constant function with value θ with the restriction of each term in J_{mix} to the set $A^{i,j}_{x,\epsilon}$.

We immediately justify this definition with the following proposition.

Proposition 2.2. Any function $h \in \operatorname{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$ satisfies $J_{mix}(h) \leq J_{mix}(g)$ for any continuous $g \in \mathcal{C}$.

Proof Sketch. We can argue directly from definitions by considering points in X_{mix} for which h and g differ.

Proposition 2.2 demonstrates that optimizing over $\mathcal{C}^*$ is at least as good as optimizing over the subset of $\mathcal{C}$ consisting of continuous functions, so we cover most cases of practical interest (i.e. optimizing deep neural networks). As such, the term "Mixup-optimal" is intended to mean optimal with respect to $\mathcal{C}^*$ throughout the rest of the paper. We may now characterize the classification of a Mixup-optimal classifier on X_{mix}.

Lemma 2.3. For any point $x \in X_{mix}$ and $\epsilon > 0$, there exists a continuous function h_ϵ satisfying:

$$h_\epsilon^i(x) = \frac{\xi_{x,\epsilon}^{i,i} + 2\left(\sum_{j=i+1}^{k} \xi_{x,\epsilon,\lambda}^{i,j} + \sum_{j=1}^{i-1}\left(\xi_{x,\epsilon}^{j,i} - \xi_{x,\epsilon,\lambda}^{j,i}\right)\right)}{\sum_{q=1}^{k} \xi_{x,\epsilon}^{q,q} + 2\left(\sum_{j=q+1}^{k} \xi_{x,\epsilon,\lambda}^{q,j} + \sum_{j=1}^{q-1}\left(\xi_{x,\epsilon}^{j,q} - \xi_{x,\epsilon,\lambda}^{j,q}\right)\right)} \tag{1}$$

With the property that $h_\epsilon(x) \to h(x)$ for every $h \in \operatorname{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$ when the limit exists.

Proof Sketch. We define h_ϵ to be $\operatorname{argmin}_{\theta \in [0,1]^k} J_{mix}(\theta)|_{B_\epsilon(x)}$ and show that this is well-defined and continuous using the strict convexity of the minimization problem.

Remark 2.4. For the important case of finite datasets, it will be shown that the limit above always exists as part of the proof of Theorem 3.2.

Although the expression for h_ϵ^i looks complicated, it just represents the expected location of the point x on all lines between class i and other classes, normalized by the sum of the expected locations for all classes. Importantly, we note that while h_ϵ as defined in Lemma 2.3 is continuous for every $\epsilon > 0$, its pointwise limit h need not be, which we demonstrate below.

Proposition 2.5. Let $X_1 = \{(0,1),(0,-1)\}$ and let $X_2 = \{(1,0),(-1,0)\}$, with $\mathbb{P}_X$ being discrete uniform over $X_1 \cup X_2$ and $\mathbb{P}_f$ being continuous uniform over $[0,1]$. Then the Mixup-optimal classifier h is discontinuous at $(0,0)$.

Proof Sketch. One may explicitly compute for $x = (0,0)$ that $h^1(x) = h^2(x) = \frac{1}{2}$.

Proposition 2.5 illustrates our first significant difference between Mixup training and standard training: there *always* exists a minimizer of the empirical cross-entropy J that can be extended to a continuous function (since a minimizer is constant on the class supports and not constrained elsewhere), whereas depending on the data the minimizer of J_{mix} can be discontinuous.

2.3 A Mixup Failure Case

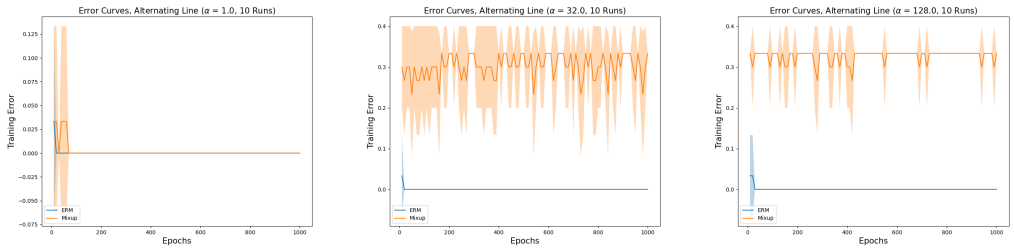
With that in mind, several model classes popular in practical applications consist of continuous functions. For example, neural networks with ReLU activations are continuous, and several works have noted that they are Lipschitz continuous with shallow networks having approximately small Lipschitz constant (Scaman & Virmaux, 2019; Fazlyab et al., 2019; Latorre et al., 2020). Given the regularity of such models, we are motivated to consider the continuous approximations h_ϵ in Lemma 2.3 and see if it is possible to construct a dataset on which h_ϵ (for a fixed ϵ) can fail to classify the original points correctly. We thus consider the following dataset:

Definition 2.6. [3-Point Alternating Line] We define $\mathcal{X}_3^2$ to be the binary classification dataset consisting of the points $\{0,1,2\}$ classified as $\{1,2,1\}$. In our setup, this corresponds to $X_1 = \{0,2\}$ and $X_2 = \{1\}$ with $\mathbb{P}_X = \frac{1}{3}1_{\{0,1,2\}}$.

Intuitively, the reason why Mixup can fail on $\mathcal{X}_3^2$ is that, for choices of $\mathbb{P}_f$ that concentrate about $\frac{1}{2}$, we will have by Lemma 2.3 that the Mixup-optimal classification in a neighborhood of point 1 should skew towards class 1 instead of class 2 due to the sandwiching of point 1 between points 0 and 2. The canonical choice of $\mathbb{P}_f$ corresponding to a mixing density of $\mathrm{Beta}(\alpha,\alpha)$ is one such choice:

Theorem 2.7. Let $\mathbb{P}_f$ have associated density $\mathrm{Beta}(\alpha,\alpha)$. Then for any classifier h_ϵ on $\mathcal{X}_3^2$ (as defined in Lemma 2.3), we may choose α such that h_ϵ does not achieve 0 classification error on $\mathcal{X}_3^2$.

Proof Sketch. For any $\epsilon > 0$, we can bound the ξ terms in Equation 1 using the fact that $\mathrm{Beta}(\alpha,\alpha)$ is strictly subgaussian (Marchal & Arbel, 2017), and then choose α appropriately.



(a) $\alpha = 1$ (b) $\alpha = 32$ (c) $\alpha = 128$

Figure 1: Training error for Mixup and regular training on $\mathcal{X}_3^2$. Each curve corresponds to the mean of 10 training runs, and the area around each curve represents a region of one standard deviation.

Experiments. The result of Theorem 2.7 leads us to believe that the Mixup training of a continuous model should fail on $\mathcal{X}_3^2$ for appropriately chosen α. To verify that the theory predicts the experiments, we train a two-layer feedforward neural network with 512 hidden units and ReLU activations on $\mathcal{X}_3^2$ with and without Mixup. The implementation of Mixup training does not differ from the theoretical setup; we uniformly sample pairs of data points and train on their mixtures. Our implementation uses PyTorch (Paszke et al., 2019) and is based heavily on the open source implementation of Manifold Mixup (Verma et al., 2019) by Shivam Saboo. Results for training using (full-batch) Adam (Kingma & Ba, 2015) with the suggested (and common) hyperparameters of $\beta_1 = 0.9, \beta_2 = 0.999$ and a learning rate of 0.001 are shown in Figure 1. The class 1 probabilities for each point in the dataset outputted by the learned Mixup classifiers from Figure 1 are shown in Table 1 below:

h	0	1	2
$\alpha = 1$	0.996	0.174	0.977
$\alpha = 32$	0.999	0.604	0.994
$\alpha = 128$	1.000	0.654	0.995

Table 1: Mixup model evaluations on $\mathcal{X}_3^2$ for different choices of α.

We see from Figure 1 and Table 1 that Mixup training fails to correctly classify the points in $\mathcal{X}_3^2$ for $\alpha = 32$, and this misclassification becomes more exacerbated as we increase α. The choice of α for which misclassifications begin to happen is largely superficial; we show in Section D of the Appendix that it is straightforward to construct datasets in the style of $\mathcal{X}_3^2$ for which Mixup training will fail even for the very mild choice of $\alpha = 1$. We focus on the case of $\mathcal{X}_3^2$ here to simplify the theory. The key takeaway is that, for datasets that exhibit (approximately) collinear structure amongst points, it is possible for inappropriately chosen mixing distributions to cause Mixup training to fail to minimize the original empirical risk.

2.4 Sufficient Conditions for Minimizing the Original Risk

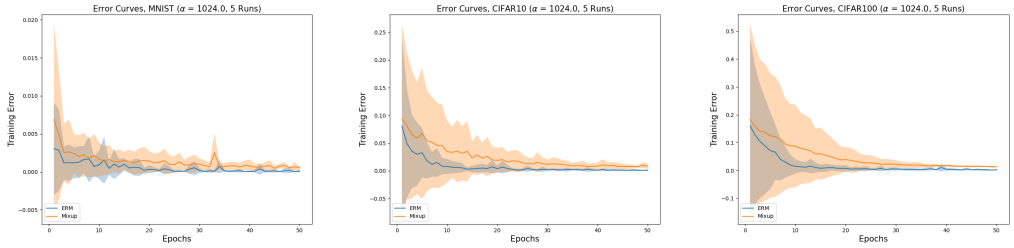
The natural follow-up question to the results of the previous subsection is: under what conditions on the data can this failure case be avoided? In other words, when can the Mixup-optimal classifier classify the original data points correctly while being continuous at those points?

Prior to answering that question, we first point out that if discontinuous functions are allowed, then Mixup training always minimizes the original risk on finite datasets:

Proposition 2.8. Consider k-class classification where the supports $X_1, ..., X_k$ are finite and $\mathbb{P}_X$ corresponds to the discrete uniform distribution. Then for every $h \in \text{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$, we have that $h^i(x) = 1$ on X_i.

Proof Sketch. This is just because the $(P_X \times P_X \times P_f)$-measure of mixing a point with itself is constant as $\epsilon \to 0$.

Note that Proposition 2.8 holds for **any** continuous mixing distribution $\mathbb{P}_f$ supported on $[0, 1]$ - we just need a rich enough model class. In order to obtain the result of Proposition 2.8 with the added restriction of continuity of h on each of the X_i, we need to make further assumptions. Namely, we



(a) MNIST (b) CIFAR-10 (c) CIFAR-100

Figure 2: Training error plots for Mixup using $\alpha = 1024$ and regular training on MNIST, CIFAR-10, and CIFAR-100. Each curve above corresponds to the mean of 5 training runs of 50 epochs, and the area around each curve represents a region of one standard deviation. Each of the Mixup models have a final training error that is within 1% of the non-Mixup models.

need to avoid the collinearity of different class points that occurred in the previous subsection, and we do so with the following assumption which is a function of a class i and a point x:

Assumption 2.9. For a class i and a point x, there exists an $\epsilon > 0$ such that $A^{j,q}_{x,\epsilon'}$ has measure zero for all $\epsilon' \leq \epsilon$ when both $j \neq i$ and $q \neq i$.

A visualization of Assumption 2.9 is provided in Section B of the appendix. With this assumption in hand, we obtain the following result as a corollary of Theorem 3.2 which is proved in the next section:

Theorem 2.10. We consider the same setting as Proposition 2.8 and further suppose that Assumption 2.9 is satisfied by all points in $\bigcup_{i=1}^{k} X_i$. Then for every $h \in \operatorname{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$, we have that $h^i(x) = 1$ on X_i and that h is continuous on $\bigcup_{i=1}^{k} X_i$.

Application of Sufficient Conditions. Theorem 2.10 suggests a way to test whether a dataset will be amenable to Mixup; we simply attempt to verify if Assumption 2.9 holds for some large enough ϵ value (depending on the Lipschitz constant of the model). This is, however, computationally intensive for large, high-dimensional datasets. We thus consider the following approximate verification scheme: we sample an epoch's worth of Mixup points (to simulate training) from a downsampled version of the train dataset, and then compute the minimum distances between each Mixup point and points (from both train and test data) of classes other than the mixed classes. The minimum over these distances corresponds to an estimate of ϵ in Assumption 2.9. For our experiments, we consider MNIST, CIFAR-10, and CIFAR-100 (Krizhevsky, 2009) downsampled to 20% of their sizes (replicating the setting of Guo et al. (2019)) and use angular distance instead of Euclidean distance since ReLU activations are positive homogeneous. We use $\mathrm{Beta}(\alpha, \alpha)$ with $\alpha = 1024$ as the mixing distribution both because the results/experiments of Subsection 2.3 demonstrate that the underfitting issue manifests in practice more readily for concentrated distributions and because larger values of α move further away from the ERM regime.

We find that the ϵ value computed according to our scheme is approximately 0.1 or greater (in angular distance) for each of MNIST, CIFAR-10, and CIFAR-100. Given the large Lipschitz constant estimates (Scaman & Virmaux, 2019) for the deep models typically used for these datasets, this much separation between the original points and the mixed points seems to imply that Mixup training should minimize the original risk. We verify this by training ResNet-18 (He et al., 2015) (using the popular implementation of Kuang Liu) on MNIST, CIFAR-10, and CIFAR-100 with and without Mixup for 50 epochs using $\alpha = 1024$ and a batch size of 128 (with otherwise identical settings to the previous subsection). Results are shown in Figure 2. We checked that the graphs shown are not sensitive to small changes in the aforementioned hyperparameters, although we did not perform an exhaustive hyperparameter search due to resource constraints.

As predicted from our approximate ϵ calculation and Theorem 2.10, Mixup training minimizes the empirical risk on MNIST, CIFAR-10, and CIFAR-100. However, we find that the test performance of Mixup at our choice of $\alpha = 1024$ is significantly worse than ERM for CIFAR-10 and CIFAR-100, affirming what was observed previously by Guo et al. (2019) for choices of $\alpha >> 1$. This is in contrast to the fact that the test data points exhibit greater angular distance to the mixed training

points than the original training points do themselves. As such, we challenge the implication in Guo et al. (2019) that collisions between mixed points and test points are the cause of the degradation in test performance - understanding when Mixup generalizes poorly seems to require more than just this perspective.

2.5 The Rate of Empirical Risk Minimization Using Mixup

Another striking aspect of the experiments in Figure 2 is that Mixup training minimizes the original empirical risk at a very similar rate to that of direct empirical risk minimization. A priori, there is no reason to expect that Mixup should be able to do this - a simple calculation shows that Mixup training only sees one true data point per epoch in expectation (each pair of points is sampled with probability $\frac{1}{m^2}$ and there are m true point pairs and m pairs seen per epoch, where m is the dataset size). The experimental results are even more surprising given that we are training using $\alpha = 1024$, which essentially corresponds to training using the midpoints of the original data points. This seems to imply that it is possible to recover the classifications of the original data points from the midpoints alone (not including the midpoint of a point and itself). We make this rigorous with the following result:

Theorem 2.11. Suppose $\{x_1, ..., x_m\}$ with $m \geq 6$ are sampled from $\bigcup_{i=1}^{k} X_i$ according to $\mathbb{P}_X$, and that $\mathbb{P}_X$ has a density (in other words, a continuous distribution). Then with probability 1, we can uniquely recover the points $\{x_1, ..., x_m\}$ given only the $\binom{m}{2}$ midpoints $\{x_{i,j}\}_{1 \leq i < j \leq m}$.

Proof Sketch. The idea is to represent the recovery problem as a linear system, and show using rank arguments that the non-recoverable points are a measure zero set.

Theorem 2.11 shows, in an information-theoretic sense, that it is possible to obtain the original data points (and therefore also their labels) from only their midpoints. While this gives more theoretical backing as to why it is possible for Mixup training using $\text{Beta}(1024, 1024)$ to recover the original data point classifications with very low error, it does not explain why this actually happens in practice at the rate that it does. A full theoretical analysis of this phenomenon would necessarily require analyzing the training dynamics of neural networks (or another model of choice) when trained only on midpoints of the original data, which is outside the intended scope of this work. That being said, we hope that such analysis will be a fruitful line of investigation for future work.

3 Generalization Properties of Mixup Classifiers

Having discussed how Mixup training differs from standard empirical risk minimization with regards to the original training data, we now consider how a learned Mixup classifier can differ from one learned through empirical risk minimization on unseen test data. To do so, we analyze the *per-class margin* of Mixup classifiers, i.e. the distance one can move from a class support X_i while still being classified as class i.

3.1 The Margin of Mixup Classifiers

Intuitively, if a point x falls only on line segments between X_i and some other classes $X_j, ...$, and if x always falls closer to X_i than the other classes, we can expect x to be classified according to class i by the Mixup-optimal classifier due to Lemma 2.3. To make this rigorous, we introduce another assumption in the same vein as Assumption 2.9:

Assumption 3.1. For a class i and a point x, suppose there exists an $\epsilon > 0$ and a $0 < \delta < \frac{\min_j d(X_i, X_j)}{2}$ such that $A^{i,j}_{x,\epsilon',\delta}$ has measure zero for all $\epsilon' \leq \epsilon$.

Here the measure zero condition on the sets $A^{i,j}_{x,\epsilon',\delta}$ is codifying the aforementioned idea that the point x falls closer to X_i than any other class on every line segment that intersects it. A visualization of Assumption 3.1 is provided in Section B of the Appendix. Now for points for which Assumptions 2.9 and 3.1 hold, we can prove:

Theorem 3.2. Consider k-class classification where the supports $X_1, ..., X_k$ are finite and $\mathbb{P}_X$ corresponds to the discrete uniform distribution. If a point x satisfies Assumptions 2.9 and 3.1 with

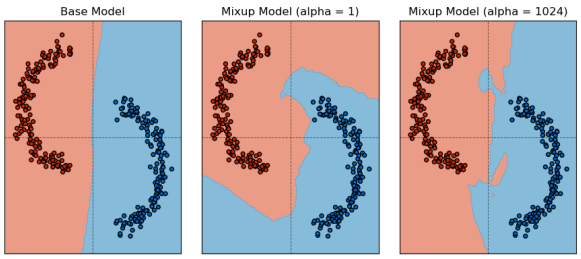


Figure 3: Decision boundary plots for standard and Mixup training on the two moons dataset of Pezeshki et al. (2020) with a class separation of 0.5. Each boundary represents the average of 10 training runs of 1500 epochs.

respect to a class i, then for every $h \in \operatorname{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$, we have that h classifies x as class i and that h is continuous at x.

Proof Sketch. The limit can be shown to exist using the Lebesgue differentiation theorem, and we can bound the limit below since the $A^{i,j}_{x,\epsilon',\delta}$ have measure zero.

Any point $x \in X_i$ is easily seen to satisfy Assumption 3.1 with respect to class i, and hence we get Theorem 2.10 as a corollary of Theorem 3.2 as mentioned in Section 2. To use Theorem 3.2 to understand generalization, we make the observation that a point x can satisfy Assumptions 2.9 and 3.1 while being a distance of up to $\frac{\min_j d(X_i, X_j)}{2}$ from some class i. This distance can be significantly farther than, for example, the optimal linear separator in a linearly separable dataset.

Experiments. To illustrate this, we consider the two moons dataset (Buitinck et al., 2013), which consists of two classes of points supported on semicircles with added Gaussian noise. Our motivation for doing so comes from the work of Pezeshki et al. (2020), in which it was noted that neural network models trained on a separated version of the two moons dataset essentially learned a linear separator while ignoring the curvature of the class supports. While Pezeshki et al. (2020) introduced an explicit regularizer to encourage a nonlinear decision boundary, we expect due to Theorem 3.2 that Mixup training will achieve a similar result without any additional modifications.

To verify this empirically, we train a two-layer neural network with 500 hidden units with and without Mixup, to have a 1-to-1 comparison with the setting of Pezeshki et al. (2020). We use $\alpha = 1$ and $\alpha = 1024$ for Mixup to capture a wide band of mixing densities. The version of the two moons dataset we use is also identical to that of the one used in the experiments of Pezeshki et al. (2020), and we are grateful to the authors for releasing their code under the MIT license. We do full-batch training with all other training, implementation, and compute details remaining the same as the previous section. Results are shown in Figure 3.

Our results affirm the observations of Pezeshki et al. (2020) and previous work (des Combes et al., 2018) that neural network training dynamics may ignore salient features of the dataset; in this case the "Base Model" learns to differentiate the two classes essentially based on the x-coordinate alone. On the other hand, the models trained using Mixup have highly nonlinear decision boundaries. Further experiments for different class separations and values of α are included in Section F of the Appendix.

3.2 When Mixup Training Learns the Same Classifier

The experiments and theory of the previous sections have shown how a Mixup classifier can differ significantly from one learned through standard training. In this subsection, we now consider the opposing question - when is the Mixup classifier the same as the one learned through standard training? The motivation for doing so is the increasing computational cost of model training; knowing when Mixup produces the same result as ERM allows a practitioner to avoid having to try Mixup training.

Towards that end, we consider the case of binary classification using a linear model $\theta^\top x$ on high-dimensional Gaussian data, which is a setting that arises naturally when training using Gaussian kernels. Specifically, we consider the dataset X to consist of n points in $\mathbb{R}^d$ distributed according to $\mathcal{N}(0, I_d)$ with $d > n$ (to be made more precise shortly). We also consider the mixing distribution to

be any symmetric distribution supported on $[0, 1]$ (thereby including as a special case $\mathrm{Beta}(\alpha, \alpha)$). We let the labels of points in X be ± 1 (so that the sign of $\theta^\top x$ is the classification), and use X_1 and X_{-1} to denote the individual class points. We will show that in this setting, the optimal Mixup classifier is the same (up to rescaling of θ) as the ERM classifier learned using gradient descent with high probability. To do so we need some additional definitions.

Definition 3.3. We say $\hat{\theta}$ is an interpolating solution, if there exists $k > 0$ such that

$$\hat{\theta}^\top x_i = -\hat{\theta}^\top z_j = k \quad \forall x_i \in X_1,\ \forall z_j \in X_{-1}.$$

Definition 3.4. The maximum margin solution $\tilde{\theta}$ is defined through:

$$\tilde{\theta} := \underset{\|\theta\|_2 = 1}{\operatorname{argmax}} \left\{ \min_{x_i \in X_1, z_j \in X_{-1}} \left\{ \theta^\top x_i, -\theta^\top z_j \right\} \right\}$$

When the maximum margin solution coincides with an interpolating solution for the dataset X (i.e. all the points are support vectors), we have that Mixup training leads to learning the max margin solution (up to rescaling).

Theorem 3.5. If the maximum margin solution for X is also an interpolating solution for X, then any θ that lies in the span of X and minimizes the Mixup loss J_{mix} for a symmetric mixing distribution $\mathbb{P}_f$ is a rescaling of the maximum margin solution.

Proof Sketch. It can be shown that θ is an interpolating solution using a combination of the strict convexity of J_{mix} as a function of θ and the symmetry of the mixing distribution.

Remark 3.6. For every θ, we can decompose it as $\theta = \theta_X + \theta_{X^\perp}$ where θ_X is the projection of θ onto the subspace spanned by X. By definition we have that $\theta_{X^\perp}$ is orthogonal to all possible mixings of points in X. Hence, $\theta_{X^\perp}$ does not affect the Mixup loss or the interpolating property, so for simplicity we may just assume θ lies in the span of X.

To characterize the conditions on X under which the maximum margin solution interpolates the data, we use a key result of Muthukumar et al. (2020), restated below. Note that Muthukumar et al. (2020) actually provide more settings in their paper, but we constrain ourselves to the one stated below for simplicity.

Lemma 3.7. [Theorem 1 in Muthukumar et al. (2020), Rephrased] Assuming $d > 10n \ln n + n - 1$, then with probability at least $1 - 2/n$, the maximum margin solution for X is also an interpolating solution.

To tie the optimal Mixup classifier back to the classifier learned through standard training, we appeal to the fact that minimizing the empirical cross-entropy of a linear model using gradient descent leads to learning the maximum margin solution on linearly separable data (Soudry et al., 2018; Ji & Telgarsky, 2018). From this we obtain the desired result of this subsection:

Corollary 3.8. Under the same conditions as Lemma 3.7, the optimal Mixup classifier has the same direction as the classifier learned through minimizing the empirical cross-entropy using gradient descent with high probability.

4 Conclusion

The main contribution of our work has been to provide a theoretical framework for analyzing how Mixup training can differ from empirical risk minimization. Our results characterize a practical failure case of Mixup, and also identify conditions under which Mixup can provably minimize the original risk. They also show in the sense of margin why the generalization of Mixup classifiers can be superior to those learned through empirical risk minimization, while again identifying model classes and datasets for which the generalization of a Mixup classifier is no different (with high probability). We also emphasize that the generality of our theoretical framework allows most of our results to hold for any continuous mixing distribution. Our hope is that the tools developed in this work will see applications in future works concerned with analyzing the relationship between benefits obtained from Mixup training and properties of the training data.

5 ETHICS STATEMENT

We do not anticipate any direct misuses of this work due to its theoretical nature. That being said, the failure case of Mixup discussed in Section 2 could serve as a way for an adversary to potentially exploit a model trained using Mixup to classify data incorrectly. However, as this requires knowledge of the mixing distribution and other hyperparameters of the model, we do not flag this as a significant concern - we would just like to point it out for completeness.

6 REPRODUCIBILITY STATEMENT

Full proofs for all results in the main body of the paper can be found in Sections C and E of the Appendix. We have also included all of the code used to generate the plots and experimental results in this paper in the supplementary material. We have tried our best to organize the code to be easy to use and extend. Detailed instructions for how to run each type of experiment are provided in the `README` file included with the code.

REFERENCES

Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. *arXiv preprint arXiv:1904.11238*, 2019.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/1cd138d0499a68f4bb72bee04bbec2d7-Paper.pdf.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. *CoRR*, abs/1309.0238, 2013. URL http://arxiv.org/abs/1309.0238.

Luigi Carratino, Moustapha Cissé, Rodolphe Jenatton, and Jean-Philippe Vert. On mixup regularization, 2020.

Ching-Yao Chuang and Youssef Mroueh. Fair mixup: Fairness via interpolation. *CoRR*, abs/2103.06503, 2021. URL https://arxiv.org/abs/2103.06503.

Remi Tachet des Combes, Mohammad Pezeshki, Samira Shabanian, Aaron C. Courville, and Yoshua Bengio. On the learning dynamics of deep neural networks. *CoRR*, abs/1809.06848, 2018. URL http://arxiv.org/abs/1809.06848.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks, 2019.

Hongyu Guo, Yongyi Mao, and Richong Zhang. Mixup as locally linear out-of-manifold regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3714–3722, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning, 2021.

Ziwei Ji and Matus Telgarsky. Risk and parameter convergence of logistic regression. *CoRR*, abs/1803.07300, 2018. URL http://arxiv.org/abs/1803.07300.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning*, pp. 5275–5285. PMLR, 2020.

JangHyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. Co-mixup: Saliency guided joint mixup with supermodular diversity. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=gvxJzw8kW4b.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Alex Lamb, Vikas Verma, Juho Kannala, and Yoshua Bengio. Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pp. 95–103, 2019.

Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization, 2020.

Olivier Marchal and Julyan Arbel. On the sub-gaussianity of the beta and dirichlet distributions. *Electronic Communications in Probability*, 22(none), Jan 2017. ISSN 1083-589X. doi: 10.1214/17-ecp92. URL http://dx.doi.org/10.1214/17-ECP92.

Vidya Muthukumar, Adhyyan Narang, Vignesh Subramanian, Mikhail Belkin, Daniel Hsu, and Anant Sahai. Classification vs regression in overparameterized regimes: Does the loss function matter? *arXiv preprint arXiv:2005.08054*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pp. 8026–8037. Curran Associates, Inc., 2019.

Mohammad Pezeshki, Sékou-Oumar Kaba, Yoshua Bengio, Aaron C. Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. *CoRR*, abs/2011.09468, 2020. URL https://arxiv.org/abs/2011.09468.

Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation, 2019.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 596–608. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/06964dce9addb1c5cb5d6e3d9838f733-Paper.pdf.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1): 2822–2878, 2018.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32:13888–13899, 2019.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pp. 6438–6447. PMLR, 2019.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization?, 2020.

Linjun Zhang, Zhun Deng, Kenji Kawaguchi, and James Zou. When and how mixup improves calibration, 2021.

A Review of Definitions and Assumptions

For convenience, we first recall the definitions and assumptions stated throughout the paper below.

$$\ell_{mix}(g,s,t,\lambda) = \begin{cases} -\log g^i(z_{st}) & s,t \in X_i \\ -\left(\lambda \log g^i(z_{st}) + (1-\lambda)\log g^j(z_{st})\right) & s \in X_i, t \in X_j \end{cases}$$

$$J^{i,j}_{mix}(g,\mathbb{P}_X,\mathbb{P}_f) = \int_{X_i \times X_j \times [0,1]} \ell_{mix}(g,s,t,\lambda)\, d(\mathbb{P}_X \times \mathbb{P}_X \times \mathbb{P}_f)(s,t,\lambda)$$

$$J_{mix}(g,\mathbb{P}_X,\mathbb{P}_f) = \sum_{i=1}^{k} J^{i,i}_{mix} + 2\sum_{i=1}^{k}\sum_{j=i+1}^{k} J^{i,j}_{mix}$$

$$A^{i,j}_{x,\epsilon} = \{(s,t,\lambda) \in X_i \times X_j \times [0,1] :\ \lambda s + (1-\lambda)t \in B_\epsilon(x)\}$$

$$A^{i,j}_{x,\epsilon,\delta} = \{(s,t,\lambda) \in X_i \times X_j \times [0,1-\delta] :\ \lambda s + (1-\lambda)t \in B_\epsilon(x)\}$$

$$X_{mix} = \left\{ x \in \mathbb{R}^n :\ \bigcup_{i,j} A^{i,j}_{x,\epsilon} \text{ has positive measure for every } \epsilon > 0 \right\}$$

$$\xi^{i,j}_{x,\epsilon} = \int_{A^{i,j}_{x,\epsilon}} d(\mathbb{P}_X \times \mathbb{P}_X \times \mathbb{P}_f)(s,t,\lambda)$$

$$\xi^{i,j}_{x,\epsilon,\lambda} = \int_{A^{i,j}_{x}} \lambda d(\mathbb{P}_X \times \mathbb{P}_X \times \mathbb{P}_f)(s,t,\lambda)$$

Definition 2.1. Let $\mathcal{C}^*$ to be the subset of $\mathcal{C}$ for which every $h \in \mathcal{C}^*$ satisfies $h(x) = \lim_{\epsilon\to 0} \operatorname{argmin}_{\theta\in[0,1]^k} J_{mix}(\theta)|_{B_\epsilon(x)}$ for all $x \in X_{mix}$ when the limit exists. Here $J_{mix}(\theta)|_{B_\epsilon(x)}$ represents the Mixup loss for a constant function with value θ with the restriction of each term in J_{mix} to the set $A^{i,j}_{x,\epsilon}$.

Definition 2.6. [3-Point Alternating Line] We define $\mathcal{X}^2_3$ to be the binary classification dataset consisting of the points $\{0,1,2\}$ classified as $\{1,2,1\}$. In our setup, this corresponds to $X_1 = \{0,2\}$ and $X_2 = \{1\}$ with $\mathbb{P}_X = \frac{1}{3}1_{\{0,1,2\}}$.

Assumption 2.9. For a class i and a point x, there exists an $\epsilon > 0$ such that $A^{j,q}_{x,\epsilon'}$ has measure zero for all $\epsilon' \le \epsilon$ when both $j \ne i$ and $q \ne i$.

Assumption 3.1. For a class i and a point x, suppose there exists an $\epsilon > 0$ and a $0 < \delta < \frac{\min_j d(X_i,X_j)}{2}$ such that $A^{i,j}_{x,\epsilon',\delta}$ has measure zero for all $\epsilon' \le \epsilon$.

Definition 3.3. We say $\hat{\theta}$ is an interpolating solution, if there exists $k > 0$ such that

$$\hat{\theta}^\top x_i = -\hat{\theta}^\top z_j = k \quad \forall x_i \in X_1,\ \forall z_j \in X_{-1}.$$

Definition 3.4. The maximum margin solution $\tilde{\theta}$ is defined through:

$$\tilde{\theta} := \underset{\|\theta\|_2=1}{\operatorname{argmax}} \left\{ \min_{x_i \in X_1, z_j \in X_{-1}} \left\{ \theta^\top x_i, -\theta^\top z_j \right\} \right\}$$

B Visualizations of Definitions and Assumptions

Due to the technical nature of the definitions and assumptions above, we provide several visualizations in Figures 4 to 7 to help aid the reader's intuition for our main results.

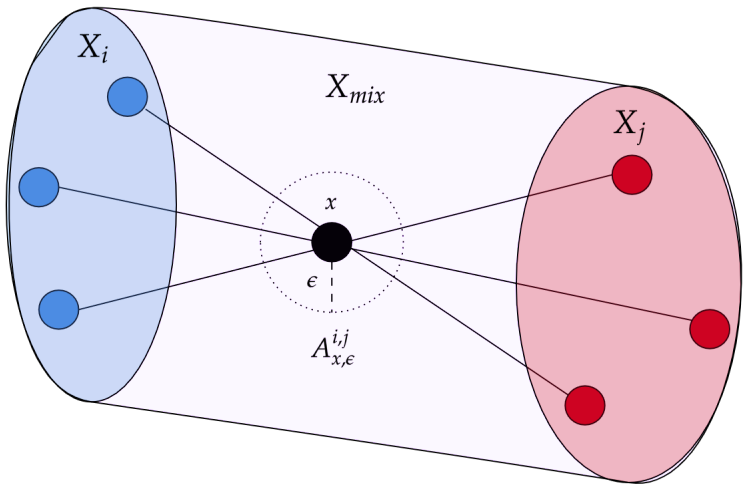


Figure 4: A visualization of X_{mix} and $A^{i,j}_{x,\epsilon}$.

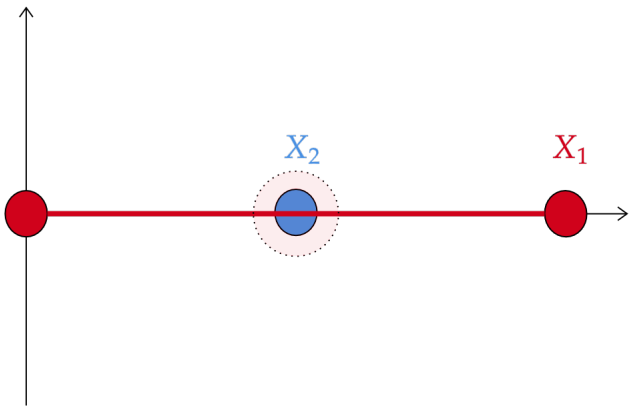


Figure 5: A visualization of the $\mathcal{X}_3^2$ dataset and how the Mixup sandwiching works.

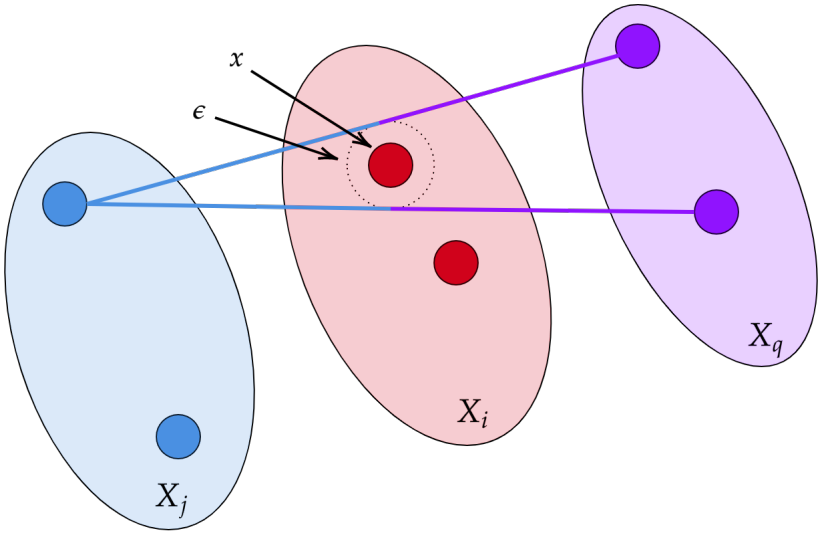


Figure 6: A visualization of Assumption 2.9, i.e. the "no collinearity" assumption.

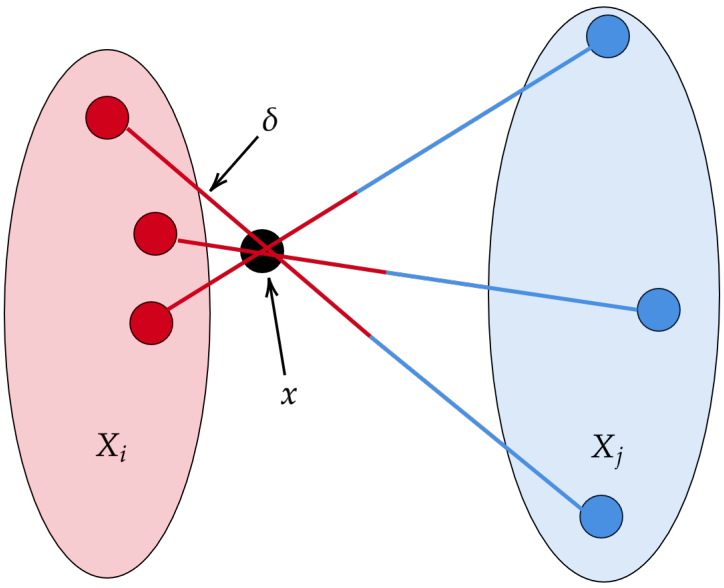


Figure 7: A visualization of Assumption 3.1. Once again, the key idea is that x falls at most δ away from X_i on every line between X_i and X_j that intersects it.

C FULL PROOFS FOR SECTION 2

We now prove all results found in Section 2 of the main body of the paper in the order that they appear.

C.1 PROOFS FOR PROPOSITIONS, LEMMAS, AND THEOREMS 2.2 - 2.10

Proposition 2.2. Any function $h \in \operatorname{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$ satisfies $J_{mix}(h) \leq J_{mix}(g)$ for any continuous $g \in \mathcal{C}$.

Proof. Intuitively, the idea is that when h and g differ at a point $x \in X_{mix}$, there must be a neighborhood of x for which the constant function that takes value $h(x)$ has lower loss than g due to the continuity constraint on g. We formalize this below.

Let h be an arbitrary function in $\operatorname{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$ and let g be a continuous function in $\mathcal{C}$. Consider a point $x \in X_{mix}$ such that the limit in Definition 2.1 exists and that $h(x) \neq g(x)$ (if such an x did not exist, we would be done). Now let θ_h and θ_g be the constant functions whose values are $h(x)$ and $g(x)$ (respectively) on all of $\mathbb{R}^n$, and further let $\theta_{h_\delta} = \operatorname{argmin}_{\theta \in [0,1]^k} J_{mix}(\theta)|_{B_\delta(x)}$ (this is shown to be a single value in the proof of Lemma 2.3 below).

Since $\theta_h \neq \theta_g$, we have that there exists a $\delta' > 0$ such that for $\delta \leq \delta'$ we have $|\theta_{h_\delta} - \theta_g| = \epsilon > 0$. From this we get that there exists $\epsilon' > 0$ depending only on ϵ and θ_g such that:

$$J_{mix}(\theta_g)|_{B_\delta(x)} - J_{mix}(\theta_h)|_{B_\delta(x)} \geq J_{mix}(\epsilon')|_{B_\delta(x)}$$

Where $J_{mix}(\epsilon')|_{B_\delta(x)}$ is an abuse of notation indicating the result of replacing all $\log g^i$ terms in the integrands of $J_{mix}(g)|_{B_\delta(x)}$ with ϵ' (this corresponds to rescaling all the ξ terms defined in Section 2.2 by ϵ'). Now by the continuity of g (and thus the continuity of $\log g^i$), we may choose $\delta \leq \delta'$ such that $J_{mix}(g)|_{B_\delta(x)} \in J_{mix}(\theta_g)|_{B_\delta(x)} \pm J_{mix}(\epsilon')|_{B_\delta(x)}$. This implies $J_{mix}(g)|_{B_\delta(x)} \geq J_{mix}(\theta_h)|_{B_\delta(x)}$, and since x was arbitrary (within the initially mentioned constraints, as outside of them h is unconstrained) we have the desired result. □

Lemma 2.3. For any point $x \in X_{mix}$ and $\epsilon > 0$, there exists a continuous function h_ϵ satisfying:

$$h_\epsilon^i(x) = \frac{\xi_{x,\epsilon}^{i,i} + 2\left(\sum_{j=i+1}^{k} \xi_{x,\epsilon,\lambda}^{i,j} + \sum_{j=1}^{i-1} \left(\xi_{x,\epsilon}^{j,i} - \xi_{x,\epsilon,\lambda}^{j,i}\right)\right)}{\sum_{q=1}^{k} \xi_{x,\epsilon}^{q,q} + 2\left(\sum_{j=q+1}^{k} \xi_{x,\epsilon,\lambda}^{q,j} + \sum_{j=1}^{q-1} \left(\xi_{x,\epsilon}^{j,q} - \xi_{x,\epsilon,\lambda}^{j,q}\right)\right)} \tag{1}$$

With the property that $h_\epsilon(x) \to h(x)$ for every $h \in \operatorname{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$ when the limit exists.

Proof. Firstly, the condition that $x \in X_{mix}$ is necessary, since if $\bigcup_{i,j} A_{x,\epsilon}^{i,j}$ has measure zero the LHS of Equation 1 is not even defined.

Now we simply take $h_\epsilon(x) = \operatorname{argmin}_{\theta \in [0,1]^k} J_{mix}(\theta)|_{B_\epsilon(x)}$ as in Definition 2.1 and show that h_ϵ is well-defined.

Since the argmin is over constant functions, we may unpack the definition of $J_{mix}|_{B_\epsilon(x)}$ and pull each of the $\log \theta^i(z_{st}(\lambda))$ terms out of the integrands and rewrite them simply as $\log \theta^i$. From this we obtain that:

$$J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)|_{B_\epsilon(x)} = -\sum_{i=1}^{k} \left(\xi_{x,\epsilon}^{i,i} + 2\left(\sum_{j=i+1}^{k} \xi_{x,\epsilon,\lambda}^{i,j} - \sum_{j=1}^{i-1} \left(\xi_{x,\epsilon}^{j,i} - \xi_{x,\epsilon,\lambda}^{j,i}\right)\right)\right) \log \theta^i$$

Where the first part of the summation above is from mixing X_i with itself, and the second part of the summation corresponds to the λ and $(1-\lambda)$ components of mixing X_i with X_j. Discarding the terms for which the coefficients above are 0 (the associated θ^i terms are taken to be 0, as anything else is suboptimal due to the summation constraint), we are left with a linear combination

of $\log \theta^{i_1}, \log \theta^{i_2}, ..., \log \theta^{i_m}$, where the set $\{i_1, i_2, ..., i_m\}$ is a subset of $\{1, ..., k\}$. We obviously cannot minimize $J_{mix}|_{B_\epsilon(x)}$ by choosing any of the θ^{i_p} to be 0, and as a consequence we also cannot choose any of the θ^{i_p} to be 1 since they are constrained to add to 1.

As such, we may consider each of the $\theta^{i_p} \in [\delta, 1-\delta]$ for some $\delta > 0$. With this consideration in mind, $J_{mix}|_{B_\epsilon(x)}$ is strictly convex in terms of the θ^{i_p}, since the Hessian of a linear combination of $\log \theta^{i_p}$ will be diagonal with positive entries when the arguments of the log terms are strictly greater than 0. Thus, as $[\delta, 1-\delta]^m$ is compact, there exists a unique solution to $h_\epsilon(x) = \operatorname{argmin}_{\theta \in [0,1]^k} J_{mix}(\theta)|_{B_\epsilon(x)}$, justifying the use of equality. This unique solution is easily computed via Lagrange multipliers, and the solution is given in Equation 1.

We have thus far defined h_ϵ on X_{mix}, and it remains to check that this construction of h_ϵ corresponds to the restriction of a continuous function from $\mathbb{R}^n \to [0,1]^k$. To do so, we first note that X_{mix} is closed, since any limit point x' of X_{mix} must necessarily either be contained in one of the supports X_i (due to compactness) or on a line segment between/within two supports (else it has positive distance from X_{mix}), and every ϵ'-neighborhood of x' contains points x for which $\bigcup_{i,j} A^{i,j}_{x,\epsilon}$ has positive measure for every ϵ (immediately implying that $\bigcup_{i,j} A^{i,j}_{x',\epsilon'}$ has positive measure).

Now we can check that h_ϵ is continuous on X_{mix} as follows. Consider a sequence $x_m \to x^*$; we wish to show that $h_\epsilon(x_m) \to h_\epsilon(x^*)$. Since the codomain of h_ϵ is compact, the sequence $h_\epsilon(x_m)$ must have a limit point h^*. Furthermore, since each $h_\epsilon(x_m)$ is the unique minimizer of $J_{mix}|_{B_\epsilon(x_m)}$, we must have that h^* is the unique minimizer of $J_{mix}|_{B_\epsilon(x^*)}$, implying that $h^* = h_\epsilon(x^*)$. We have thus established the continuity of h_ϵ on X_{mix}, and since X_{mix} is closed (in fact, compact), we have by the Tietze Extension Theorem that h_ϵ can be extended to a continuous function on all of $\mathbb{R}^n$.

Finally, if $\lim_{\epsilon \to 0} h_\epsilon(x)$ exists, then it is by definition $h(x)$ for any $h \in \mathcal{C}^*$ and therefore for any $h \in \operatorname{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$. □

Proposition 2.5. Let $X_1 = \{(0,1), (0,-1)\}$ and let $X_2 = \{(1,0), (-1,0)\}$, with $\mathbb{P}_X$ being discrete uniform over $X_1 \cup X_2$ and $\mathbb{P}_f$ being continuous uniform over $[0,1]$. Then the Mixup-optimal classifier h is discontinuous at $(0,0)$.

Proof. Considering the point $x = (0,0)$, we note that only $\xi^{1,1}_{x,\epsilon}$ and $\xi^{2,2}_{x,\epsilon}$ are non-zero for all $\epsilon > 0$. Furthermore, we have that $\xi^{1,1}_{x,\epsilon} = \xi^{2,2}_{x,\epsilon} = \frac{\epsilon}{4}$ since the $\mathbb{P}_f$-measure of $B_\epsilon(\frac{1}{2})$ is 2ϵ and the sets $A^{1,1}_{x,\epsilon}, A^{2,2}_{x,\epsilon}$ have $\mathbb{P}_X$-measure $\frac{1}{8}$. From this we have that the limit $\lim_{\epsilon \to 0} h_\epsilon(x) = \left[\frac{1}{2}\ \frac{1}{2}\right]^\top$ is the Mixup-optimal value at x.

On the other hand, every other point on the line segment connecting the points in X_1 will have an ϵ-neighborhood disjoint from the line segment between the points in X_2 (and vice versa), so we will have $h^1_{\epsilon'}(x) = 1$ for all $\epsilon' \leq \epsilon$. This implies that $h_{\epsilon'}(x) \to [1\ 0]$ (and an identical result for points on the X_2 line segment), so we have that the Mixup-optimal classifier is discontinuous at $(0,0)$ as desired. □

Theorem 2.7. Let $\mathbb{P}_f$ have associated density $\text{Beta}(\alpha, \alpha)$. Then for any classifier h_ϵ on $\mathcal{X}^2_3$ (as defined in Lemma 2.3), we may choose α such that h_ϵ does not achieve 0 classification error on $\mathcal{X}^2_3$.

Proof. Fix a classifier h_ϵ as defined in Lemma 2.3 on $\mathcal{X}^2_3$. Now for $Y \sim \text{Beta}(\alpha, \alpha)$ we have by the fact that $\text{Beta}(\alpha, \alpha)$ is strictly subgaussian that $P\left(\left|Y - \frac{1}{2}\right| \leq \epsilon\right) \geq 1 - 2\exp\left(-\epsilon^2/(2\sigma^2)\right)$ where $\sigma^2 = \frac{1}{4\alpha+2}$ is the variance of $\text{Beta}(\alpha, \alpha)$. As a result, we can choose $\alpha > \frac{1}{2}\left(\frac{\log 4}{\epsilon^2} - 1\right)$ to guarantee that $P\left(\left|Y - \frac{1}{2}\right| \leq \epsilon\right) > \frac{1}{2}$ and therefore that $\xi^{1,1}_{1,\epsilon} > \frac{1}{9} = \xi^{2,2}_{1,\epsilon}$.

Now we have by Lemma 2.3 that:

$$h^1_\epsilon(1) = \frac{\xi^{1,1}_{1,\epsilon} + 2\xi^{1,2}_{1,\epsilon,\lambda}}{\xi^{1,1}_{1,\epsilon} + 2\xi^{1,2}_{1,\epsilon} + \xi^{2,2}_{1,\epsilon}} = \frac{\xi^{1,1}_{1,\epsilon} + \xi^{1,2}_{1,\epsilon}}{\xi^{1,1}_{1,\epsilon} + 2\xi^{1,2}_{1,\epsilon} + \xi^{2,2}_{1,\epsilon}} > \frac{1}{2}$$

Where the second line followed from the fact that $\xi^{1,2}_{1,\epsilon,\lambda} = \xi^{1,2}_{1,\epsilon} - \xi^{1,2}_{1,\epsilon,\lambda}$ by the symmetry of $\text{Beta}(\alpha, \alpha)$. Thus, we have shown that h_ϵ will classify the point 1 as class 1 despite it belonging to class 2. □

Proposition 2.8. Consider k-class classification where the supports $X_1, ..., X_k$ are finite and $\mathbb{P}_X$ corresponds to the discrete uniform distribution. Then for every $h \in \text{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$, we have that $h^i(x) = 1$ on X_i.

Proof. The full proof is not much more than the proof sketch in the main body. For $x \in X_i$, we have that $\xi^{j,q}_{x,\epsilon} \to 0$ and $\xi^{j,q}_{x,\epsilon,\lambda} \to 0$ as $\epsilon \to 0$ for every $(j,q) \neq (i,i)$, while $\xi^{i,i}_{x,\epsilon} \to \frac{1}{|\bigcup_i X_i|^2}$ and $\xi^{i,i}_{x,\epsilon,\lambda} \to \frac{\mathbb{E}_{\mathbb{P}_f}[\lambda]}{|\bigcup_i X_i|^2}$. As a result, we have $\lim_{\epsilon \to 0} h^i_\epsilon(x) = 1$ as desired. □

Theorem 2.10. We consider the same setting as Proposition 2.8 and further suppose that Assumption 2.9 is satisfied by all points in $\bigcup_{i=1}^k X_i$. Then for every $h \in \text{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$, we have that $h^i(x) = 1$ on X_i and that h is continuous on $\bigcup_{i=1}^k X_i$.

Proof. Obtained as a corollary of Theorem 3.2. □

C.2 Proof for Theorem 2.11

Prior to proving Theorem 2.11, we first introduce some additional notation as well as a lemma that will be necessary for the proof.

Notation: Throughout the following m corresponds to the number of data points being considered (as in Theorem 2.11), and as a shorthand we use $[m]$ to indicate $\{1, ..., m\}$. We use e_i to denote the i-th basis vector in R^m, and use e'_i to denote the i-th basis vector in $R^{\binom{m}{2}}$. Let $A \in \mathbb{R}^{\binom{m}{2} \times m}$ be the "Mixup matrix" where each row has two 1s and the other entries are 0, representing a mixture of the two data points whose associated indices have a 1. The $\binom{m}{2}$ rows enumerate all the possible mixings of the m data points. In this way, A is uniquely defined up to a permutation of rows. We can pick any representative as our A matrix, and prove the following lemma.

Lemma C.1. Assume $m > 6$, and $P \in \mathbb{R}^{\binom{m}{2} \times \binom{m}{2}}$ is a permutation matrix. If PA is not a permutation of the columns of A, then the rank of $[A, PA]$ is larger than m.

Using this lemma we can prove Theorem 2.11.

Theorem 2.11. Suppose $\{x_1, ..., x_m\}$ with $m \geq 6$ are sampled from $\bigcup_{i=1}^k X_i$ according to $\mathbb{P}_X$, and that $\mathbb{P}_X$ has a density (in other words, a continuous distribution). Then with probability 1, we can uniquely recover the points $\{x_1, ..., x_m\}$ given only the $\binom{m}{2}$ midpoints $\{x_{i,j}\}_{1 \leq i < j \leq m}$.

Proof. We only need to show that the set of original points that do not allow unique recovery has measure zero. It suffices to show this for the first entry (dimension) of the x_i's, as the result then follows for all dimensions. For convenience, we group the first entries of the data points $\{x_i\}_{i=1}^m$ into a vector $w^* \in \mathbb{R}^m$, and similarly obtain $w \in \mathbb{R}^m$ from $\{x'_i\}_{i=1}^m$. Suppose $w^* \in \mathbb{R}^m$ is not a permutation of $w \in \mathbb{R}^m$ but that they have the same set of Mixup points. We only need to show that the set of such w^* has measure zero.

Suppose A is a Mixup matrix and P is a permutation matrix. Suppose PA is not a permutation of the columns in A. We would need $Aw^* = PAw$, which is equivalent to $[A, PA][(w^*)^\top, w^\top]^\top$. According to Lemma C.1, we know the rank of $[A, PA]$ is at least $m + 1$, which implies that the solution set of w^* is at most $m - 1$.

So fixing A and P, the set of non-recoverable w^* has measure zero. There are only a finite number of combinations of A and P. Thus, considering all of these A and PA, the full set of non-recoverable w^* still has measure zero. □

C.2.1 Proof of Supporting Lemma

Lemma C.1. Assume $m > 6$, and $P \in \mathbb{R}^{\binom{m}{2} \times \binom{m}{2}}$ is a permutation matrix. If PA is not a permutation of the columns of A, then the rank of $[A, PA]$ is larger than m.

Proof. First, we show that both the ranks of A and PA are m. For all $i \in [m-1]$, define $u_i = e_1 + e_{i+1}$, and define $u_m = e_2 + e_3$. Note that these m vectors are all rows of A. The first $(m-1)$ vectors $\{u_i\}_{i=1}^{m-1}$ are linearly independent because each u_i has a unique direction e_{i+1} that is not a linear combination of any other vectors in $\{u_i\}_{i=1}^{m-1}$. Besides, we know that the span of $\{u_i\}_{i=1}^{m-1}$ is a subspace of $\{v \in \mathbb{R}^m : \sum_{i=2}^{m} v(i) = v(1)\}$ where $v(i)$ is the i-th entry of v. Therefore, u_m doesn't lie in the span of $\{u_i\}_{i=1}^{m-1}$, which implies that these n vectors $\{u_i\}_{i=1}^{n}$ are linearly independent. This shows that A is at least rank m. Since A only has m columns, we know that the rank of A is m. The matrix PA is also rank m because P is full rank.

Therefore, to show that the rank of $[A, PA]$ is larger than m, we only need to find a vector v that lies in the column span of A but is not in the column span of PA. To do this, we need the following claim:

Claim 1. There exists a row index subset $I \subseteq [\binom{m}{2}]$ with size $(m-1)$ such that the sub-matrix composed by the rows of A with indices in I has a column that is all-one vector, but every column of the sub-matrix composed by the rows of PA with indices in I is not all-one vector.

Proof of Claim 1. By the definition of A, each column of A has only $(m-1)$ ones and the other entries are zero. Therefore, the position of $(m-1)$ ones in a column of A can uniquely determine that column vector. Besides, for an index set I with size $(m-1)$, the sub-matrix composed by the rows of A with indices in I cannot have two columns that are both all-one vector. This is because otherwise A will have duplicate rows, which contradicts the definition of A. Therefore, there are m possible choices of I, each of which corresponds to the positions of all ones in a column of A.

Assume by contradiction that for these m choices of I, there exist a column of the sub-matrix composed by the rows of PA with indices in I that is all-one vector. Then these m choices of I also correspond to the positions of all ones in a column of PA. This means that the columns of A and PA are the same up to permutations, which contradicts the assumption of our theorem. This finishes the proof of this claim.

Now define B_1 as the sub-matrix composed by the rows of A with indices in I, and C_1 as the sub-matrix composed by the rows of PA with indices in I. Without loss of generality, suppose $I = [m-1]$, and suppose the first column of B_1 is all-one vector. Let $u = -e_1 + \sum_{i=2}^{m} e_i \in \mathbb{R}^m$, we know that $Au = 2\sum_{i=m}^{\binom{m}{2}} e'_i$, i.e., the first $(m-1)$ entries of Au are 0, and the other entries are 2. Define $v \triangleq Au$, we are going to show that v is not in the column span of PA. Let C_2 be the sub-matrix in PA consisting of the rows that are not in C_1, then the following claim shows that C_2 has full column rank.

Claim 2. $rank(C_2) = m$.

Proof of Claim 2. In this proof, we will consider each column of C_2 as a vertex. Since each row of C_2 has only two 1s, we view each row as an edge connecting the two columns which correspond to the two 1s in that row. From the definition of A we know that the graph we constructed is a simple undirected graph. Then we are going to show that we can select m "edges" from C_2 which are linearly independent.

There are $\binom{m}{2} - (m-1)$ edges in C_2. From $m > 6$ we know that $\binom{m}{2} - (m-1) > \frac{m^2}{4}$, so from Turán's theorem, C_2 contains at least one triangle. Assume this triangle is (i, j, k), then we select edges (i, j), (j, k) and (i, k) and define $E_3 = \{i, j, k\}$. $\forall r \in [m], r \geq 3$, we select an edge connecting E_r with $[m] \setminus E_r$. Assume that edge is (s, t) where $s \in E_r$ and $t \notin E_r$, we then add t to E_r, i.e., $E_{r+1} = E_r \cup \{t\}$. In the next two paragraphs, we are going to show that there are always edges between E_r and $[m] \setminus E_r$ (so we can successfully select m edges in total), and the m edges we selected are linearly independent.

In matrix PA, there are $r(m-r)$ edges between E_r and $[m] \setminus E_r$. Since C_2 is constructed by deleting $(m-1)$ edges from PA, the number of edges left between E_r and $[m] \setminus E_r$ is at least $r(m-r) - (m-1)$. When $3 \leq r \leq m-2$, we have $r(m-r) - (m-1) > 0$. When $r = m-1$, the only case where there is no edge between E_r and $[m] \setminus E_r$ is when all edges from vertex $[m] \setminus E_r$ is in C_1, which means that C_1 has a column that is all-one vector and is a contradiction. Therefore, there are always edges between E_r and $[m] \setminus E_r$ and we can successfully select m edges.

Then we only need to show that these m selected edges are linearly independent. We use $\{u_i\}_{i=1}^m$ to denote the vectors that correspond to these edges, i.e., $u_1 = e_i + e_j$, $u_2 = e_j + e_k$, $u_3 = e_i + e_k$, $\cdots$ Assume by contradiction that they are linearly independent, then there exists $x \in \mathbb{R}^m, x \neq 0$ such that $\sum_{i=1}^m x(i)u_i = 0$. By the selection process of the edges, we know that $\forall r \geq 4$, u_r has a unique direction, so $x(r) = 0$. Therefore, $x(1)u_1 + x(2)u_2 + x(3)u_3 = 0$. Since $\{u_1, u_2, u_3\}$ are linearly independent, we have $x = 0$, which is a contradiction. Thus, these m selected edges are linearly independent, proving the claim.

Now assume by contradiction that v is in the column span of PA, then $\exists w \in \mathbb{R}^m$ such that $v = PAw$. Let $v_2 \in \mathbb{R}^{\binom{m}{2}-(m-1)}$ be the bottom $\binom{m}{2} - (m-1)$ entries of v, then $v_2 = C_2 w$. Define w_0 to be the all-one vector in $\mathbb{R}^m$, we know that w_0 is a valid solution to $v_2 = C_2 w$. Since C_2 has full column rank, w_0 must be the unique solution to $v_2 = C_2 w$. This implies that $v = PAw_0$. However, we know that $PAw_0 = 2\sum_{i=1}^{\binom{m}{2}} e'_i \neq v$, which is a contradiction. Thus, v is not in the column span of A, which finishes the proof of the lemma. □

D Additional Experiments for Section 2

As noted in the main paper, it is not difficult to use Definition 8 to construct datasets on which Mixup training empirically fails for small values of α. The observation to make is that a major part of the proof of Theorem 2.7 is the fact that the same-point mixing probability of point 1 is small relative to the same-class mixing probability of class 0. As the former probability decreases quadratically with the dataset size m, we are motivated to consider extensions of $\mathcal{X}_3^2$ consisting of more points and more classes. Towards that end, we consider the following generalization of $\mathcal{X}_3^2$:

Definition D.1 (m-Point k-Class Alternating Line)**.** We define $\mathcal{X}_m^k$ to be the k-class classification dataset consisting of the points $\{0, 1, ..., m-1\}$ classified according to their value mod k incremented by 1. As before, $\mathbb{P}_X$ is the normalized counting measure on $\mathcal{X}_m^k$.

We now consider training a two-layer feedforward network on $\mathcal{X}_{10}^2$ and $\mathcal{X}_{10}^{10}$ using the same procedure and hyperparameters as in the main paper. The results are shown in Figure 8 below.

Here we see that even at $\alpha = 1$, Mixup training fails to minimize the original empirical risk on $\mathcal{X}_{10}^2$ and $\mathcal{X}_{10}^{10}$, whereas in the main paper we noted that at $\alpha = 1$ Mixup training had no issues minimizing the original risk on $\mathcal{X}_3^2$. Interestingly, we find that even standard training does not completely minimize the original empirical risk on $\mathcal{X}_{10}^2$, once again perhaps a result of the regularity of the two-layer network model (although this is merely a hypothesis, and we do not analyze this phenomenon further here).

E Full Proofs for Section 3

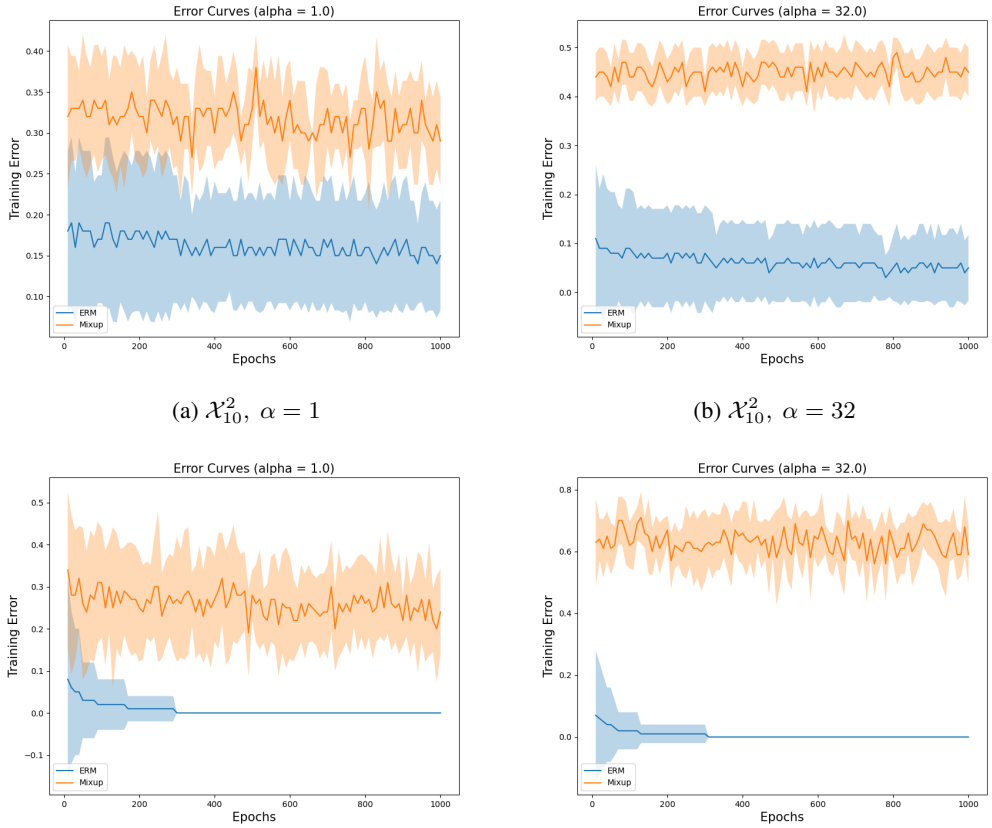
E.1 Proof of Theorem 3.2

Theorem 3.2. Consider k-class classification where the supports $X_1, ..., X_k$ are finite and $\mathbb{P}_X$ corresponds to the discrete uniform distribution. If a point x satisfies Assumptions 2.9 and 3.1 with respect to a class i, then for every $h \in \operatorname{argmin}_{g \in \mathcal{C}^*} J_{mix}(g, \mathbb{P}_X, \mathbb{P}_f)$, we have that h classifies x as class i and that h is continuous at x.

Proof. For two points p, q in the supports $X_1, ..., X_k$ with a line segment between them intersecting x, let $\lambda(p, q, x)$ denote the value of λ for which $\lambda p + (1-\lambda)q = x$. Since the supports are finite, we have that there exists $\epsilon_1 > 0$ such that for all $\epsilon \leq \epsilon_1$ we have that:

$$\xi_{x,\epsilon}^{i,j} = \sum_p \sum_q \int_{B_\epsilon(\lambda(p,q,x)} d\mathbb{P}_f$$

Where the summations are over all points p, q with line segments containing x. Now we have by the Lebesgue differentiation theorem applied to the integral term in the summations above that the



(a) $\mathcal{X}_{10}^{2},\ \alpha = 1$ (b) $\mathcal{X}_{10}^{2},\ \alpha = 32$

(c) $\mathcal{X}_{10}^{10},\ \alpha = 1$ (d) $\mathcal{X}_{10}^{10},\ \alpha = 32$

Figure 8: Training error plots for Mixup and regular training on $\mathcal{X}_{10}^{2}$ and $\mathcal{X}_{10}^{10}$. Each curve corresponds to the mean of 10 training runs, and the area around each curve represents a region of one standard deviation.

following limit (as well as each limit for the other coordinate functions) exists:

$$\lim_{\epsilon\to 0} h_\epsilon^i(x) = \lim_{\epsilon\to 0} \frac{\xi_{x,\epsilon}^{i,i} + 2\left(\sum_{j=i+1}^{k} \xi_{x,\epsilon,\lambda}^{i,j} + \sum_{j=1}^{i-1}\left(\xi_{x,\epsilon}^{j,i} - \xi_{x,\epsilon,\lambda}^{j,i}\right)\right)}{\sum_{q=1}^{k} \xi_{x,\epsilon}^{q,q} + 2\left(\sum_{j=q+1}^{k} \xi_{x,\epsilon,\lambda}^{q,j} + \sum_{j=1}^{q-1}\left(\xi_{x,\epsilon}^{j,q} - \xi_{x,\epsilon,\lambda}^{j,q}\right)\right)}$$

$$= \lim_{\epsilon\to 0} \frac{\frac{1}{\epsilon}\xi_{x,\epsilon}^{i,i} + \frac{2}{\epsilon}\left(\sum_{j=i+1}^{k} \xi_{x,\epsilon,\lambda}^{i,j} + \sum_{j=1}^{i-1}\left(\xi_{x,\epsilon}^{j,i} - \xi_{x,\epsilon,\lambda}^{j,i}\right)\right)}{\frac{1}{\epsilon}\sum_{q=1}^{k} \xi_{x,\epsilon}^{q,q} + 2\left(\sum_{j=q+1}^{k} \xi_{x,\epsilon,\lambda}^{q,j} + \sum_{j=1}^{q-1}\left(\xi_{x,\epsilon}^{j,q} - \xi_{x,\epsilon,\lambda}^{j,q}\right)\right)}$$

Now by Assumption 2.9, we need only consider the $\xi_{x,\epsilon}^{i,i}, \xi_{x,\epsilon}^{i,j}, \xi_{x,\epsilon,\lambda}^{i,j}$ terms (as there are no other line segments that contain x). Furthermore, by Assumption 3.1 we have that there exists $\epsilon_2 > 0$ such that for all $\epsilon \leq \epsilon_2$ we have $\lambda(p, q, x) - \epsilon \geq \frac{1}{2}$ for $p \in X_i$ and $q \in X_j$. Now by similar logic to the proof of Theorem 2.7, we have for all $\epsilon \leq \min(\epsilon_1, \epsilon_2)$ that:

$$h_\epsilon = \frac{\xi_{x,\epsilon}^{i,i} + \sum_{j=i+1}^{k} \xi_{x,\epsilon}^{i,j} + \sum_{j=1}^{i-1} \xi_{x,\epsilon}^{j,i}}{\xi_{x,\epsilon}^{i,i} + 2\left(\sum_{j=i+1}^{k} \xi_{x,\epsilon}^{i,j} + \sum_{j=1}^{i-1} \xi_{x,\epsilon}^{j,i}\right)} \geq \frac{1}{2}$$

Since the above also holds for a sufficiently small neighborhood about x, we have the desired result.

□

E.2 PROOF OF THEOREM 3.5

Let us first recall the setting of Theorem 3.5, since it is more specialized than that of the previous results.

Setting. We consider the case of binary classification using a linear model $\theta^\top x$ on high-dimensional Gaussian data, which is a setting that arises naturally when training using Gaussian kernels. Specifically, we consider the dataset $\mathcal{X}$ to consist of n points in $\mathbb{R}^d$ distributed according to $\mathcal{N}(0, I_d)$ with $d > n$ (to be made more precise shortly). We let the labels of points in X be ± 1 (so that the sign of $\theta^\top x$ is the classification), and use X_1 and X_{-1} to denote the individual class points. Additionally, we let $n_1 = |X_1|$ and $n_{-1} = |X_{-1}|$.

Before introducing the proof of Theorem 3.5, we first present a lemma that will be necessary in the proof.

Lemma E.1. [Strict Convexity of J_{mix} on Data Span] Suppose $n_1 = n_{-1} = 1$, i.e. there are two data points x, z with opposite labels. If x and z are linearly independent, then J_{mix} is strictly convex with respect to θ on the span of x and z.

The reason this lemma focuses on the two data point case is that in the proof of Theorem 3.5 we will break up the Mixup loss into the sum over these cases. With this lemma, we may now prove Theorem 3.5. Before doing so, we point out that the definition of J_{mix} considered in Theorem 3.5 and Lemma E.1 appears different from the definition in the main paper and in Section 2.2 in two respects. Firstly, because we are now considering binary classification with linear models, we write the cross-entropy as the logistic loss applied to the parameter vector θ, which is why J_{mix} is written in the expanded form seen in the proofs below. Secondly, we use J_{mix} to refer to only the non-ERM terms (i.e. no mixing of identical points) since we need only prove the results for this part of the loss (as an interpolating solution is automatically ERM-optimal).

Theorem 3.5. If the maximum margin solution for X is also an interpolating solution for X, then any θ that lies in the span of X and minimizes the Mixup loss J_{mix} for a symmetric mixing distribution $\mathbb{P}_f$ is a rescaling of the maximum margin solution.

Proof. Since $d > n$, we have the data sets $\{x_i\}_{i=1}^{n_1}$, $\{z_j\}_{j=1}^{n_{-1}}$ are linearly independent with probability one. We will now break the proof into two parts.

Part I: $n_1 = n_{-1} = 1$**.** Denote $\theta^\top x_1 = u$ and $\theta^\top z_1 = -v$, then

$$J_{mix} = \mathbb{E}_\lambda \left[\lambda \log(1 + \exp(-\lambda u + (1-\lambda)v)) + (1-\lambda)\log(1 + \exp(\lambda u - (1-\lambda)v))\right]. \quad (2)$$

Note λ is sampled from a distribution which is symmetric and has full support on $[0, 1]$. Therefore, we can do the change of variables $\lambda := 1 - \lambda$, and

$$J_{mix} = \mathbb{E}_\lambda \left[\lambda \log(1 + \exp(-\lambda v + (1-\lambda)u)) + (1-\lambda)\log(1 + \exp(\lambda v - (1-\lambda)u))\right]. \quad (3)$$

Combining Eq.(2) and Eq.(3), we know if $(u, -v)$ is a global minimum of J_{mix}, then so is $(v, -u)$. But the strict convexity in Lemma E.1 implies such a global minimum is unique, so we must have $u = v = k(\mathbb{P}_f)$, where $k(\mathbb{P}_f)$ is a constant that only depends on $\mathbb{P}_f$. Furthermore, $\forall \lambda \in [0, 1]$, define

$$h_\lambda(k) = \lambda \log(1 + \exp((1-2\lambda)k)) + (1-\lambda)\log(1 + \exp((2\lambda - 1)k)), \quad (4)$$

then $\forall k > 0$,

$$\begin{aligned} h_\lambda(-k) - h_\lambda(k) &= (1-2\lambda)\log(1 + \exp((1-2\lambda)k)) + (2\lambda - 1)\log(1 + \exp((2\lambda-1)k)) \\ &= (1-2\lambda)\log\left(\frac{1 + \exp((1-2\lambda)k)}{1 + \exp((2\lambda-1)k)}\right) \geq 0, \end{aligned} \quad (5)$$

and

$$h'_\lambda(k)|_{k=0} = -\frac{1}{2}(1-2\lambda)^2 \leq 0. \quad (6)$$

Hence, we must have $k(\mathbb{P}_f) > 0$.

Part II: General n_1 and n_{-1}. With the ERM terms ablated and $J_{mix}^{i,j}$ representing the loss incurred from mixing of points x_i and z_j, the decomposition of the empirical Mixup loss can be written as:

$$J_{mix}(\theta) = \frac{1}{n_1 n_{-1}} \sum_{i=1}^{n_1} \sum_{j=1}^{n_{-1}} J_{mix}^{i,j}(\theta). \tag{7}$$

By the proof in the previous part we know if $J_{mix}^{i,j}$ is minimized, then we must have $\theta^\top x_i = -\theta^\top z_j = k(\mathbb{P}_f) > 0$. On the other hand, if $J_{mix}^{i,j}(\theta)$ are minimized simultaneously for all pairs (i, j), then clearly $J_{mix}(\theta)$ is also minimized. This is possible since the datasets are linearly independent, so there exists $\theta \in \mathbb{R}^d$, such that

$$\theta^\top x_i = -\theta^\top z_j = k(\mathbb{P}_f) > 0 \quad \forall i \in [n_1],\ \forall j \in [n_{-1}]. \tag{8}$$

Now we can conclude that any θ that minimizes the Mixup loss J_{mix} is an interpolating solution. Restricting θ to the span of $\mathcal{X}$ finishes the proof. □

E.2.1 Proof of Supporting Lemma

Lemma E.1. [Strict Convexity of J_{mix} on Data Span] Suppose $n_1 = n_{-1} = 1$, i.e. there are two data points x, z with opposite labels. If x and z are linearly independent, then J_{mix} is strictly convex with respect to θ on the span of x and z.

Proof. Denote

$$f(\lambda) = \lambda x + (1 - \lambda) z, \tag{9}$$

then J_{mix} can be expressed as

$$\begin{aligned} J_{mix}(\theta) &= \mathbb{E}_{\lambda \sim p} \left[\lambda \log\big(1 + \exp\big(-\theta^\top f(\lambda)\big)\big) + (1 - \lambda) \log\big(1 + \exp\big\{\theta^\top f(\lambda)\big\}\big) \right] \\ &= \mathbb{E}_{\lambda \sim p} \left[\log\big(1 + \exp\big(-\theta^\top f(\lambda)\big)\big) + (1 - \lambda) \theta^\top f(\lambda) \right]. \end{aligned} \tag{10}$$

Note that the second term in Eq.(10) is linear in θ, hence the Hessian of J_{mix} can be written as

$$\begin{aligned} \nabla^2 J_{mix}(\theta) &= \mathbb{E}_{\lambda \sim p} \left[\frac{\exp\big(-\theta^\top f(\lambda)\big)}{(1 + \exp(-\theta^\top f(\lambda)))^2} f(\lambda) f(\lambda)^\top \right] \\ &:= \mathbb{E}_{\lambda \sim p} g(\lambda). \end{aligned} \tag{11}$$

Define $\mathcal{B} := \text{Span}\{x, z\}$. To show J_{mix} is strictly convex on $\mathcal{B}$, it suffices to show for every non-zero vector $a \in \mathcal{B}$, we always have

$$a^\top \nabla^2 J_{mix}(\theta) a > 0. \tag{12}$$

Note that $g(\lambda)$ is continuous *w.r.t.* λ and that $\mathbb{P}_f$ has full support on $[0, 1]$, it suffices to show either

$$a^\top f(0) f(0)^\top a > 0 \ \text{ or } \ a^\top f(1) f(1)^\top a > 0, \tag{13}$$

which is equivalent to either

$$a^\top x \neq 0 \ \text{ or } \ a^\top z \neq 0. \tag{14}$$

This is obvious since a is a non-zero vector in $\mathcal{B}$, and that x and z are linearly independent. □

F Additional Experiments for Section 3

In this section, we consider different class separations and choices of the mixing parameter α when training on the two moons dataset, with all other experimental settings being the same as in Section 3.1.

Upon decreasing the class separation to 0.1, we note that even standard training captures more of the nonlinear aspects of the data, as was observed in the prior work of Pezeshki et al. (2020).

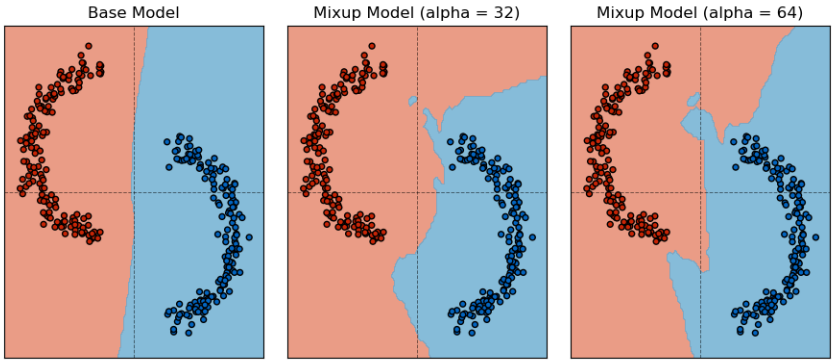


Figure 9: Decision boundary plots for $\alpha = 32,\ 64$ and a class separation of 0.5.

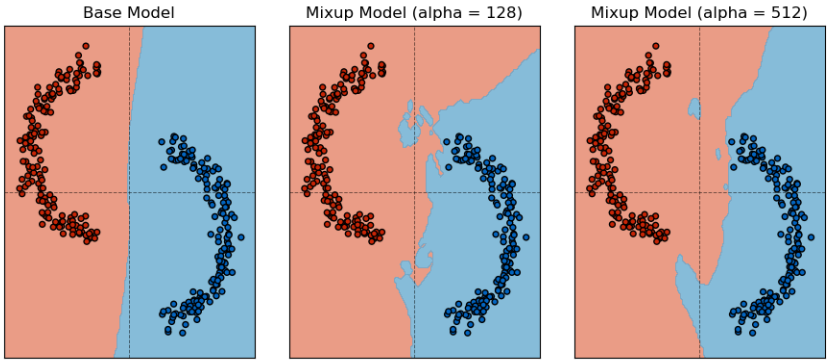


Figure 10: Decision boundary plots for $\alpha = 128,\ 512$ and a class separation of 0.5.

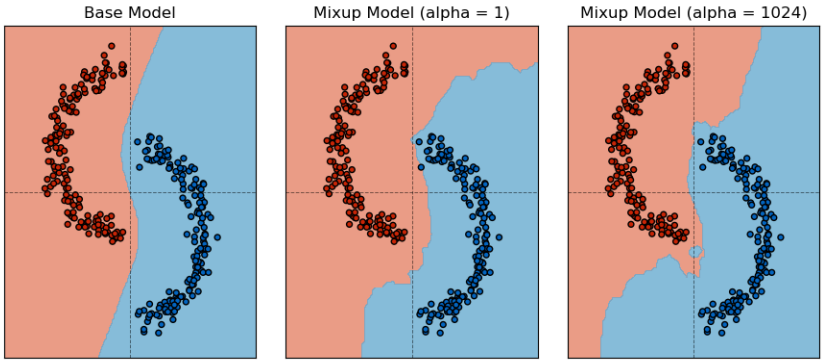


Figure 11: Decision boundary plots for $\alpha = 1,\ 1024$ and a class separation of 0.1.

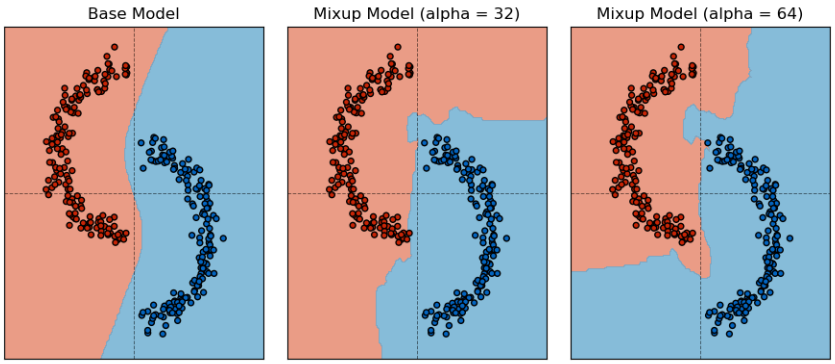


Figure 12: Decision boundary plots for $\alpha = 32,\ 64$ and a class separation of 0.1.

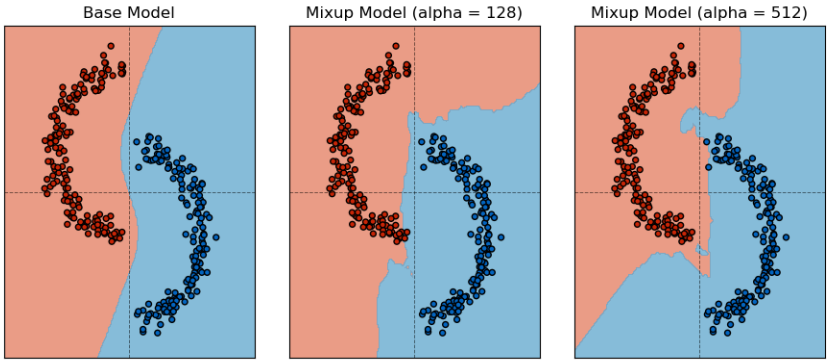


Figure 13: Decision boundary plots for $\alpha = 128,\ 512$ and a class separation of 0.1.