Lower Bounds on $0-1$ Loss for Multi-class Classification with a Test-time Attacker

Sihui Dai[*]
Princeton University
sihuid@princeton.edu

Wenxin Ding[*]
University of Chicago
wenxind@uchicago.edu

Arjun Nitin Bhagoji
University of Chicago
abhagoji@uchicago.edu

Daniel Cullina
Pennsylvania State University
cullina@psu.edu

Prateek Mittal
Princeton University
pmittal@princeton.edu

Ben Y. Zhao
University of Chicago
ravenben@cs.uchicago.edu

Abstract

Finding classifiers robust to adversarial examples is critical for their safe deployment. Determining the robustness of the *best possible classifier* under a given threat model and comparing it to that achieved by state-of-the-art training methods is thus an important diagnostic tool. In this paper, we find achievable information-theoretic lower bounds on loss in the presence of a test-time attacker for *multi-class classifiers on any discrete dataset*. We provide a general framework for computing lower bounds on $0-1$ loss based on solving a linear program (LP). This LP is constructed based on what we introduce as a conflict hypergraph, and we explore different settings in the construction of this hypergraph and their impact on the computed lower bound. Our work enables, for the first time, an analysis of the gap to optimal robustness for classifiers in the multi-class setting.

1 Introduction

Machine learning models are susceptible to small imperceptible perturbations known as adversarial examples [Szegedy et al., 2013]. Several empirical approaches have been proposed for training classifiers robust to adversarial examples [Madry et al., 2018, Zhang et al., 2019, Croce et al., 2020]. In this paper, we ask: *what is the minimum achievable* $0-1$ *loss on the training data for any multi-class classifier in the presence of an adversary?* Addressing this question allows us to assess the effectiveness of existing techniques for robust training and goes significantly beyond previous work [Bhagoji et al., 2019, Pydi and Jog, 2020, Bhagoji et al., 2021] restricted to binary classification.

We formulate and solve an optimization problem to determine classifier-agnostic lower bounds for the robust $0-1$ loss for multi-class classification. We provide techniques to efficiently compute these bounds for commonly used image datasets including MNIST, CIFAR-10, and CIFAR-100, and compare these bounds to current robust training methods. Our **contributions** are as follows:

1. We introduce a framework for computing lower bounds for $0-1$ loss in the presence of an adversary for multi-class classification. In this framework, given a dataset and adversary, we construct a conflict hypergraph which contains vertices representing training examples in the dataset and hyperedges representing overlaps between adversarial neighborhoods around each training example. This hypergraph is then used to construct a linear program whose solution is a lower bound on $0-1$ loss. This conflict hypergraph is analogous to the conflict graph introduced by Bhagoji et al. [2021] for binary classification.
2. We compute lower bounds on $0-1$ loss for commonly used image datasets including MNIST, CIFAR-10, and CIFAR-100. Using these lower bounds, we analyze the effectiveness of robust training methods like adversarial training and TRADES.

ML Safety Workshop, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

3. We provide a method to optimally aggregate lower bounds from binary classification to determine a lower bound on the true multi-class bound. We find the full multi-class bound is much more informative in most practical settings. In addition, we analyze the impact of hyperedges of different sizes on the multi-class lower bound.

2 Formulating Lower Bounds as a Linear Program

In this section, we provide a framework for obtaining lower bounds on $0-1$ loss in the presence of a test-time adversary. We will show that the lower bound can be computed as the solution of a linear program (LP), which is defined based on a hypergraph constructed from the classification problem. We note that these bounds are classifier agnostic and can be computed for any discrete distribution (*e.g.* training dataset).

2.1 Problem Formulation

Notation. We consider a supervised classification problem where inputs are sampled from input space $\mathcal{X}$, and labels belong to K classes: $y \in \mathcal{Y} = \{1, ..., K\}$. Let P be the joint probability over $\mathcal{X} \times \mathcal{Y}$. Let $\mathcal{H}_{\text{soft}}$ denote the space of all soft classifiers; specifically, for all $h \in \mathcal{H}_{\text{soft}}$ we have that $h : \mathcal{X} \to [0,1]^K$ and $\sum_{i=1}^{K} h(x)_i = 1$ for all $x \in \mathcal{X}$. Here, $h(x)_i$ represents the probability that the input x belongs to the i^{th} class. For classification, we consider a randomized hard-decision classifier $f(x) \sim \text{Cat}(h(x))$ where Cat denotes the categorical distribution. We use classification error probability of this randomized classifier as our loss function: $\ell(h, (x, y)) = 1 - h(x)_y = \Pr[f(x) \neq y]$.

Test-time adversaries. We are interested in the setting where there exists a test-time adversary that can modify any data point x to generate an adversarial example $\tilde{x}$ that lies within the neighborhood $N(x)$ of x. We require that for all $x \in \mathcal{X}$, $N(x)$ always contains x. The objective of the learner is to find h that outputs classification probabilities such that in expectation the classifier f achieves the minimum $0-1$ loss over P in the presence of an adversary restricted to N. The optimal loss, which depends on a distribution P, hypothesis class $\mathcal{H}$, and neighborhood N, is

$$L^*(P, \mathcal{H}, N) = \min_{h \in \mathcal{H}} \mathop{\mathbb{E}}_{(x,y)\sim P} \max_{\tilde{x} \in N(x)} \ell(h, (\tilde{x}, y)) = 1 - \max_{h \in \mathcal{H}} \mathop{\mathbb{E}}_{(x,y)\sim P} \min_{\tilde{x} \in N(x)} h(\tilde{x})_y. \tag{1}$$

In general, when $\mathcal{H}' \subseteq \mathcal{H}$, $L^*(P, \mathcal{H}, N) \leq L^*(P, \mathcal{H}', N)$. In particular, the class of hard-decision classifiers is a subset of the class of soft classifiers ($\mathcal{H}_{\text{soft}}$). For any fixed architecture, the class of functions represented by some parameterization of that architecture is another subset. Thus optimal loss over $\mathcal{H}_{\text{soft}}$ provides a lower bound on loss in these settings.

Optimal loss for distributions with finite support. Since we would like to compute the optimal loss for distributions P *with finite support*, we can rewrite the expectation in Equation 1 as an inner product. Let V be the support of P, i.e. the set of points $(x, y) \in \mathcal{X} \times \mathcal{Y}$ that have positive probability in P. Let $p \in [0,1]^V$ be the probability mass vector for P: $p_v = P(\{v\})$. For a soft classifier h, let $q(h) \in \mathbb{R}^V$ be the vector of correct classification probabilities for vertices $v = (x, y) \in V$: $q(h)_v := \min_{\tilde{x} \in N(x)} h(\tilde{x})_y$. Rewriting (1) with our new notation, we have $1 - L^*(P, \mathcal{H}_{\text{soft}}, N) = \max_{h \in \mathcal{H}_{\text{soft}}} p^T q(h)$. This is the maximization of a linear function over all possible vectors $q(h)$. In fact, the convex hull of all correct classification probability vectors is a polytope and this optimization problem is a linear program, as described next.

2.2 Conflict Hypergraph Construction

In order to characterize the possible vectors of correct classification probabilities, we represent the structure of the classification problem with a hypergraph. We introduce a conflict hypergraph $\mathcal{G}^{(\leq m)} = (V, \mathcal{E}^{(\leq m)})$, which records intersections between neighborhoods of points in $\mathcal{X}$. The conflict hypergraph is parameterized by $m \in \{2, 3, ..., K\}$, which controls the maximum size of hyperedges in $\mathcal{E}^{(\leq m)}$. For all m, the set of vertices V of $\mathcal{G}^{(\leq m)}$ is the support of P . For a set $S \subseteq V$, $S \in \mathcal{E}^{(\leq m)}$ (i.e. S is a hyperedge in $\mathcal{G}^{(\leq m)}$) if all vertices in S belong to different classes, $|S| \leq m$, and the neighborhoods of all vertices in S overlap: $\bigcap_{(x,y)\in S} N(x) \neq \emptyset$. Thus, the size of each hyperedge is at most m, $\mathcal{E}^{(\leq m)}$ is downward-closed, and every $v \in V$ is a degree 1 hyperedge.

2.3 Linear Program

Using the conflict hypergraph $\mathcal{G}^{(\leq m)}$, we can now describe the linear program for $L^*(P, \mathcal{H}_{\text{soft}}, N)$.

Lemma 1 (Optimal loss as an LP (Adapted from [Bhagoji et al., 2021])). *Let $\mathcal{H}_{soft}$ be the class of all soft classifiers and N define the neighborhood of the adversary. For any distribution P with finite support, let $p \in \mathbb{R}^V$ be the probability mass vector for P ($p_v = P(\{v\})$). Then,*

$$1 - L^*(P, \mathcal{H}_{soft}, N) = \max_q p^T q \quad s.t \quad q \geq 0, \quad B^{(\leq K)} q \leq 1. \tag{2}$$

where $B^{(\leq K)} \in \mathbb{R}^{\mathcal{E}^{(\leq K)} \times V}$ is the incidence matrix of the conflict hypergraph.

See Appendix A for proof. Because $\mathcal{E}^{(\leq m)} \subseteq \mathcal{E}^{(\leq K)}$, the inequality $B^{(\leq m)} q \leq 1$ is a relaxation of the constraint in (2). By decreasing m, we get a sequence of linear programs with fewer constraints. The values of these programs are a sequence of looser lower bounds on $L^*(P, \mathcal{H}_{\text{soft}}, N)$. These values can also be interpreted as optimal risks in variations on the adversarial classification game. We describe this in Appendix B. In the remainder of the text, we will denote the solution to the variation on the LP in Equation 2 with constraint $B^{(\leq m)} q \leq 1$ as $L^*(m)$.

2.4 Approximating multiclass lower bounds with lower bounds for binary classification

Prior works [Bhagoji et al., 2019, 2021] proposed methods of computing lower bounds for binary classification problems. We now ask the question: *how informative are lower bounds for binary classification in computing a lower bound for multi-class classification?*

Consider the setting where we obtain lower bounds on $0-1$ loss for all one-versus-one binary classification tasks. We can consider a weaker adversary which chooses pairings of classes such that for each pair of classes (y_1, y_2), the adversary's strategy is to perform a targeted attack on all inputs with label y_1 to class y_2 and vice versa. These pairings can be optimized through maximum weight coupling on the $K \times K$ matrix of one-versus-one binary classification lower bounds. Thus, by using maximum weight coupling to find optimal pairs and then averaging the one-versus-one losses across these pairs, we can obtain a lower bound on multi-class $0-1$ loss using one-versus-one $0-1$ loss lower bounds. We call this lower bound the *class-only* lower bound and denote this lower bound as $L^*_{\text{co}}(2)$. Since the adversary is using a weaker strategy in comparison to the adversary considered for computing multi-class lower bounds, we note that $L^*_{\text{co}}(2) \leq L^*(m)$ for all $m \in \{2...K\}$. In Appendix B, we demonstrate that L^* and L^*_{co} are the optimal losses for different classification games and provide additional details about the corresponding games.

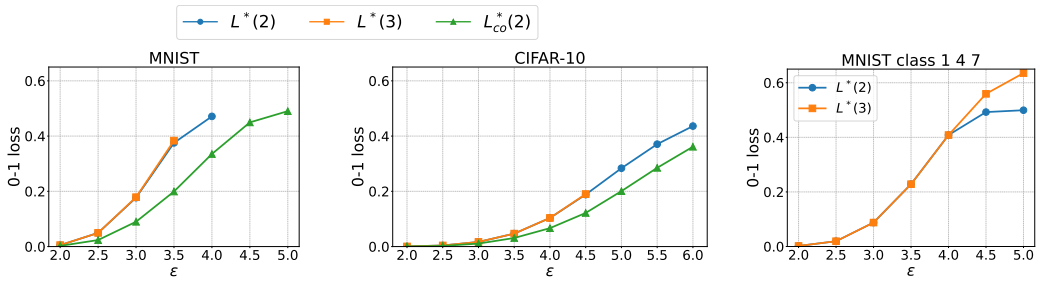
3 Empirical Results

We now compute lower bounds on $0-1$ loss for the training sets of MNIST and CIFAR-10 under the presence of an ℓ_2-bounded adversary [1]. We use the 3 variations of $0-1$ loss lower bounds described in Section 2.3 and 2.4 to capture the differences between the various classification games. Further experimental details are in Appendix D. Additionally, we include results for 3-class Gaussian data in Appendix E.4.

Binary vs Multi-class Lower Bounds: We first investigate how much more information we gain from computing multi-class lower bounds in comparison to aggregating lower bounds from all pairs of binary classification problems. In other words, we compare $L^*(2)$ with $L^*_{\text{co}}(2)$. We obtain $L^*_{\text{co}}(2)$ by performing maximum weight coupling over a $K \times K$ matrix of optimal binary classification losses. In Figures 1a and 1b, we see a noticeable gap between the two losses, and the gap increases as ϵ increases, showing that L^* is a tighter bound that L^*_{co} for use as a diagnostic tool. This demonstrates that an adversary that is restricted to class specific targeted attacks can be much weaker than an adversary that is allowed to use untargeted example dependent attacks. The difference tapers off as $L^*(2)$ approaches its maximum value of 0.5, due to the high connectivity of all pairwise graphs.

Impact of Hyperedges: We further investigate the impact of changing m when computing $L^*(m)$. In Figures 1a and 1b, there is minimal change between $L^*(2)$ and $L^*(3)$ up to the ϵ we are able

[1] Due to the computational expense at larger budgets and sample sizes, some results could not be obtained in a reasonable time (See Appendix E.2 for runtime details). In these cases, we reduce the sample size to obtain representative results.

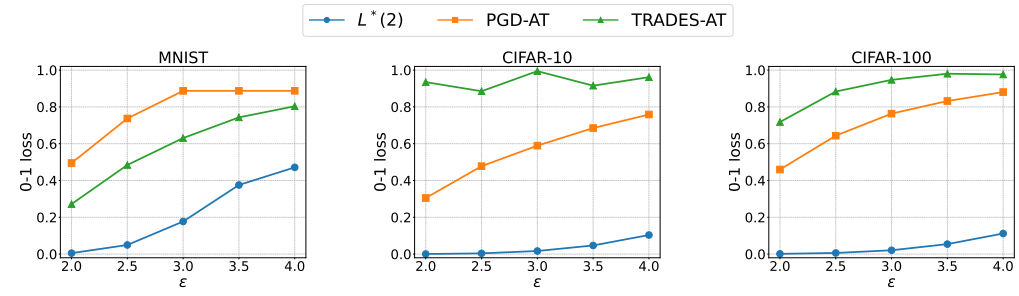


(a) MNIST (b) CIFAR-10

Figure 1: Computed multiclass $0-1$ loss lower bound considering up to degree 2 ($L^*(2)$) and degree 3 hyperedges ($L^*(3)$), and maximum weight coupling of pairs of binary $0-1$ loss lower bounds ($L^*_{\text{co}}(2)$). MNIST with 1000 samples to compute $L^*(3)$ for.

Figure 2: Multiclass $0-1$ lower bounds for 3-class MNIST with 1000 samples per class (classes 1, 4, and 7) to observe the difference between $L^*(2)$ and $L^*(3)$ as ϵ increases (Fig. 2). After $\epsilon = 4$, there is a clear separation arising from the use of hyperedges. $L^*(3)$ approaches its maximum of $\frac{2}{3}$ while $L^*(2)$ tapers off at $\frac{1}{2}$. We provide additional results on visualizing the impact of hyperedges in Appendix E.1.

State of Current Defenses To assess how well existing robust training methods are able to fit the training data, we compare the training error of models trained with PGD adversarial training (PGD-AT) [Madry et al., 2018, Gowal et al., 2020] and TRADES (TRADES-AT) [Zhang et al., 2019] to $L^*(2)$. For TRADES-AT, we use $\beta = 1$ for MNIST and $\beta = 6$ for CIFAR-10 and CIFAR-100. We evaluate models using APGD [Croce and Hein, 2020] and report our results in Figure 3. Across all datasets, we find that a large gap exists between the classifiers found using robust training and the computed lower bound. This further validates previous work that existing training methods are unable to robustly fit the training data well, with an even larger gap than in the 2-class setting. We note that on CIFAR-10 and CIFAR-100 for the ϵ tested, TRADES-AT at $\beta = 6$ leads to models with low clean loss but high robust loss, and performance may improve with better tuning of β.



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

Figure 3: Comparison of loss obtained via current adversarial training (AT) techniques (PGD-AT [Madry et al., 2018] and TRADES-AT [Zhang et al., 2019]) to computed lower bounds ($L^*(2)$).

4 Discussion

Our work in this paper firmly establishes for the multi-class case what was known only in the binary setting before: *there exists a large gap in the performance of current robust classifiers and the optimal classifier*. This raises the question: *why does this gap arise and how can we improve training to decrease this gap?* In the future, we would like to see if increasing architecture size or using additional data [Sehwag et al., 2022, Gowal et al., 2021] can close this gap. A surprising finding from our experiments was that the addition of hyperedges to the multi-way conflict graph did not change the lower bounds much, indicating we are in a regime where multi-way intersections minimally impact optimal probabilities. One major limitation of our work is the computational expense at larger budgets and sample sizes. We suspect this is due to the general-purpose nature of the solvers we use and are developing custom algorithms to speed up the determination of lower bounds.

Acknowledgments and Disclosure of Funding

ANB, WD and BYZ were supported in part by NSF grants CNS-1949650, CNS-1923778, CNS-1705042, the C3.ai DTI, and the DARPA GARD program. SD was supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-2039656. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. SD and PM were supported in part by the National Science Foundation under grants CNS-1553437 and CNS-1704105, the ARL's Army Artificial Intelligence Innovation Institute (A2I2), the Office of Naval Research Young Investigator Award, the Army Research Office Young Investigator Prize, Schmidt DataX award, and Princeton E-ffiliates Award.

References

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Lower bounds on adversarial robustness from optimal transport. In *Advances in Neural Information Processing Systems*, pages 7496–7508, 2019.

Muni Sreenivas Pydi and Varun Jog. Adversarial risk via optimal transport and optimal couplings. In *Proceedings of the 37th International Conference on Machine Learning*, pages 7814–7823, 2020.

Arjun Nitin Bhagoji, Daniel Cullina, Vikash Sehwag, and Prateek Mittal. Lower bounds on cross-entropy loss in the presence of test-time adversaries. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pages 2206–2216. PMLR, 2020.

Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL `https://openreview.net/forum?id=WVX0NNVBBkV`.

Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A. Mann. Improving robustness using generated data. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 4218–4233, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/21ca6d0cf2f25c4dbb35d8dc0b679c3f-Abstract.html`.

Yann LeCun and Corrina Cortes. The MNIST database of handwritten digits. 1998.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

MOSEK ApS. *MOSEK Optimizer API for Python 10.0.22*, 2019. URL `https://docs.mosek.com/10.0/pythonapi/index.html`.

Martin S Andersen, Joachim Dahl, and Lieven Vandenberghe. Cvxopt: Python software for convex optimization, 2013.

Appendix

A Proof of Lemma 1

This proof consists of 2 parts. First, we will show that the classification probabilities in q must satisfy the constraints of the LP. We will then show that for any q^* that is a solution to the LP that there exists an $h \in \mathcal{H}_{\text{soft}}$ which attains q^*.

Constraints of the LP must hold. We note the first constraint hold because classification probabilities must lie in the range $[0, 1]$. We will now demonstrate that the second constraint must also hold. Let $e = ((x_1, y_1), ..., (x_n, y_n))$ be an nth degree hyperedge in $\mathcal{E}^{(\leq m)}$. By construction of $\mathcal{E}$, $\exists \tilde{x} \in \bigcap_{i=1}^{n} N(x_i)$. By definition of $q(h)$, for each vertex $(x_i, y_i), i \in \{1...n\}$, we have that $q(h)_{(x_i,y_i)} \leq h(\tilde{x})_{y_i}$. Thus, $\sum_{i=1}^{n} q(h)_{(x_i,y_i)} \leq \sum_{i=1}^{n} h(\tilde{x})_{y_i} \leq \sum_{i=1}^{K} h(\tilde{x})_i = 1$. This gives us the second constraint.

q^* is attainable by some $h \in \mathcal{H}_{\text{soft}}$. We will prove this by constructing an oracle classifier $h^* \in \mathcal{H}_{\text{soft}}$ that outputs the classification probabilities specified by q^* (thus achieving the lower bound on $0-1$ loss) by doing the following:

1. Initialize $h^*(x)_y = \frac{1}{K}$ for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$
2. For each (x_1, y_1) in the support of P, find all hyperedges
3. Sort hyperedges by degree. Starting from the hyperedge of lowest degree do the following: for each hyperedge $((x_1, y_1), ..., (x_n, y_n))$,
   (a) Set $h^*(x)_{y_i} = q^*_{(x_i,y_i)}$ for all $x \in \bigcap_{i=1}^{n} N(x_i)$.
   (b) For all classification labels $j \in \mathcal{Y} \setminus \{y_1...y_n\}$ not involved in the hyperedge, set $h^*(x)_j = \frac{1-\sum_{i=1}^{n} q^*_{(x_i,y_i)}}{|\mathcal{Y} \setminus \{y_1...y_n\}|}$.

By construction, we can see that for any $x \in \mathcal{X}$, $h^*(x)$ outputs values in $[0, 1]^K$ and $\sum_{i=1}^{K} h^*(x) = 1$. This makes h^* a valid soft classifier that achieves the lower bound.

B Hierarchy of Classification Games

Consider the following variations on the multiclass classification game with adversarial perturbations. Both variations have a parameter m. In both the adversary selects $C \subseteq \mathcal{Y}$ such that $y \in C$ and $|C| = m$ and gives C to the classifier as side information. Thus the classifier is a function $h : \mathcal{X} \times \binom{\mathcal{Y}}{m} \to \mathcal{Y}$.

- *Example-dependent class list*: The adversary samples $(x, y) \sim P$, then selects $\tilde{x}$ and C.
- *Class-only class list*: The adversary samples $y \sim P_y$, then selects C, then samples $x \sim P_{x|y}$ and selects $\tilde{x} \in N(x)$.

In other words, in the former variation the adversary's choice of C can depend on x and in the latter variation it cannot. Let $L^*(m, P, \mathcal{H}, N)$ be the minimum loss in the example-dependent class list game with list length m and let $L^*_{\text{co}}(m, P, \mathcal{H}, N)$ be the minimum loss in the class-only class list game. When $m = K$, $C = \mathcal{Y}$ and both variations are equivalent to the original game, so $L^*(P, \mathcal{H}, N) = L^*(K, P, \mathcal{H}, N) = L^*_{\text{co}}(K, P, \mathcal{H}, N)$. For each variation, the game becomes more favorable for the adversary as m increases: $L^*(m, P, \mathcal{H}, N) \leq L^*(m+1, P, \mathcal{H}, N)$ and $L^*_{\text{co}}(m, P, \mathcal{H}, N) \leq L^*_{\text{co}}(m+1, P, \mathcal{H}, N)$. For each m, it is more favorable for the adversary to see x before selecting C, *i.e.* $L^*_{\text{co}}(m, P, \mathcal{H}, N) \leq L^*(m, P, \mathcal{H}, N)$.

Furthermore, the values of these games are each characterized by a natural variation on the linear program for the value of the main game. For classifiers using class list side-information, the correct classification probability is defined as follows: $q(h)_{(x,y)} = \min_{\tilde{x} \in N(x)} \min_{C: y \in C} h(\tilde{x}, C)_y$.

Lemma 2. *In the example-dependent side information game, correct classification probabilities satisfy $B^{(\leq m)} q(h) \leq 1$. Furthermore the optimal loss satisfies*

$$1 - L^*(m, P, \mathcal{H}, N) = \max_{h \in \mathcal{H}} p^T q(h) = \max_q p^T q \quad s.t \quad q \geq 0, \quad B^{(\leq m)} q \leq 1 \tag{3}$$

In the class-only variation, the adversarial strategy for selecting C is a specified by a conditional p.m.f. $p_{C|y}(C|y)$. Thus $p_{y|L}(y|C) = p_{C|y}(C|y) p_{\mathbf{y}}(y) / \sum_{y'} p_{C|y}(C|y') p_y(y')$.

The optimal loss of the classifier against a particular adversarial strategy is just a mixture of the optimal losses for each class list: $\sum_L p_{C|y}(C|y) p_y(y) L^*(P_{x|y} P_{y|C}, \mathcal{H}, N)$.

If $p_{C|y}(C|y) = p_{C|y}(C|y')$ for all $y, y' \in C$, then $p_{y|C}(y|C) = p_{\mathbf{y}}(y) / \sum_{y' \in C} p_{\mathbf{y}}(y')$ and the adversary has not provided the classifier with extra information beyond the fact that $y \in C$. Thus $P_{x|y} P_{y|C} = P|(y \in C)$. Call the optimal loss under this restriction $L^*_{\text{co,sym}}(m, P, \mathcal{H}, N)$.

Lemma 3. $L^*_{co,sym}(2, P, \mathcal{H}, N) = \min_z \sum_{i,j} p_y(i) z_{i,j} L^*(P|(y \in \{i, j\}, \mathcal{H}, N)$ *where z is a symmetric doubly stochastic matrix.*

C Hyperedge Finding

One challenge in computing lower bounds for $0-1$ loss in the multi-class setting is that we need to find hyperedges in the conflict hypergraph. In this section, we will consider an ℓ_2 adversary: $N(x) = \{x' \in \mathcal{X} | \ ||x' - x||_2 \leq \epsilon\}$ and describe an algorithm for finding hyperedges within the conflict graph.

We first note that for an n-way hyperedge to exist between n inputs $\{x_i\}_{i=1}^n$, $\{x_i\}_{i=1}^n$ must all lie on the interior of an $n-1$-dimensional hypersphere of radius ϵ.

Given input x_1, ..., x_n where $x_i \in \mathbb{R}^d$, we first show that distance between any two points in the affine subspace spanned by the inputs can be represented by a distance matrix whose entries are the squared distance between inputs. This allows us to compute the circumradius using the distance information only, not requiring a coordinate system in high dimension. Then we find the circumradius using the properties that the center of the circumsphere is in the affine subspace spanned by the inputs and has equal distance to all inputs.

We construct matrix $X \in \mathbb{R}^{d \times n}$ whose i^{th} column is input x_i. Let $D \in \mathbb{R}^{n \times n}$ be the matrix of squared distances between the inputs, i.e., $D_{i,j} = \|x_i - x_j\|^2$.

We first notice that D can be represented by X and a vector in $\mathbb{R}^n$ whose i^{th} entry is the squared norm of x_i. Let $\Delta \in \mathbb{R}^n$ be such vector such that $\Delta_i = \|x_i\|^2 = (X^T X)_{i,i}$. Then given that $D_{i,j}$ is the squared distance between x_i and x_j, we have

$$D_{i,j} = \|x_i\|^2 + \|x_j\|^2 - 2\langle x_i, x_j \rangle,$$

which implies that

$$D = \Delta \mathbf{1}^T + \mathbf{1} \Delta^T - 2X^T X.$$

Let $\alpha, \beta \in \mathbb{R}^n$ be vectors of affine weights: $\mathbf{1}^T \alpha = \mathbf{1}^T \beta = 1$. Then $X\alpha$ and $X\beta$ are two points in the affine subspace spanned by the columns of X. The distance between $X\alpha$ and $X\beta$ is $\frac{-(\alpha-\beta)^T D(\alpha-\beta)}{2}$, shown as below:

$$\begin{aligned}
\frac{-(\alpha-\beta)^T D(\alpha-\beta)}{2} &= \frac{-(\alpha-\beta)^T(\Delta \mathbf{1}^T + \mathbf{1}\Delta^T - 2X^T X)(\alpha-\beta)}{2} \\
&= \frac{-(0+0-2(\alpha-\beta)^T X^T X(\alpha-\beta)}{2} \\
&= \|X\alpha - X\beta\|^2.
\end{aligned}$$

Now we compute the circumradius using the squared distance matrix D. The circumcenter is in the affine subspace spanned by the inputs so we let $X\alpha$ to be the circumcenter where $\mathbf{1}^T\alpha = 1$. Let $e^{(i)} \in \mathbb{R}^n$ be the i^{th} standard basis vector. The distance between the circumcenter and x_i is $\|X\alpha - Xe^{(i)}\|^2$. From previous computation, we know that $\|X\alpha - Xe^{(i)}\|^2 = \frac{-(\alpha-e^{(i)})^T D(\alpha-e^{(i)})}{2}$. Since the circumcenter has equal distance to all inputs, we have

$$(\alpha - e^{(1)})^T D(\alpha - e^{(1)}) = \ldots = (\alpha - e^{(n)})^T D(\alpha - e^{(n)}). \tag{4}$$

Note that the quadratic term in α is identical in each of these expressions. In addition, $e^{(i)T} De^{(i)} = 0$ for all i. So equation 4 simplifies to the linear system

$$\begin{aligned} e^{(i)T} D\alpha = c &\implies D\alpha = c\mathbf{1} \\ &\implies \alpha = cD^{-1}\mathbf{1} \end{aligned}$$

for some constant c. Since $\mathbf{1}^T\alpha = 1$, we have

$$\begin{aligned} 1 = \mathbf{1}^T\alpha &= c\mathbf{1}^T D^{-1}\mathbf{1} \\ \implies \frac{1}{c} &= \mathbf{1}^T D^{-1}\mathbf{1} \end{aligned}$$

assuming that D is invertible. The square of the circumradius, r^2, which is the squared distance between the circumcenter and x_1, is

$$\begin{aligned} &\|X\alpha - Xe^{(1)}\|^2 \\ =&\frac{-(\alpha - e^{(1)})^T D(\alpha - e^{(1)})}{2} \\ =&e^{(1)T} D\alpha - \frac{\alpha^T D\alpha}{2} \\ =&c - \frac{c^2\mathbf{1}^T D^{-1}\mathbf{1}}{2} \\ =&\frac{c}{2} \\ =&\frac{1}{2\mathbf{1}^T D^{-1}\mathbf{1}}. \end{aligned}$$

Therefore, assuming matrix D is invertible, the circumradius is $\frac{1}{\sqrt{2\mathbf{1}^T D^{-1}\mathbf{1}}}$.

The inverse of D can be computed as $\frac{\det D}{\operatorname{adj} D}$. Since $\alpha = cD^{-1}\mathbf{1}$, we have $\alpha = c\frac{\det D}{\operatorname{adj} D}\mathbf{1}$. As $r^2 = \frac{c}{2}$, constant c is non-negative. Therefore, $\alpha \propto \frac{\det D}{\operatorname{adj} D}\mathbf{1}$.

When all entries of α are non-negative, the circumcenter is a convex combination of the all inputs and the circumsphere is the minimum sphere in $\mathbb{R}^{n-1}$ that contains all inputs. Otherwise, the circumsphere of $\{x_i | \alpha_i > 0\}$ is the minimum sphere contains all inputs.

After finding the radius of the minimum sphere that contains all inputs, we compare the radius with the budget ϵ. If the radius is no larger than ϵ, then there is a hyperedge of degree n among the inputs.

D Experimental Setup

Datasets We compute lower bounds for MNIST [LeCun and Cortes, 1998], CIFAR-10, and CIFAR-100 [Krizhevsky and Hinton, 2009]. Since we do not know the true distribution of these datasets, we compute lower bounds based on the empirical distribution of the training set for each dataset.

LP solver For solving the LP in Equation 2, we primarily use Mosek LP solver [ApS, 2019]. When Mosek solver did not converge, we use CVXOpt's LP solver [Andersen et al., 2013].

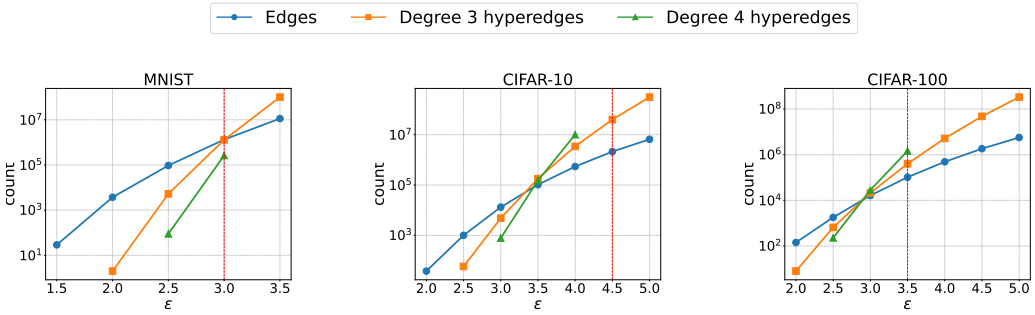
Training Details For MNIST, we use 40 step optimization to find adversarial examples during training and use step size $\frac{\epsilon}{30}$ and train all models for 50 epochs. For CIFAR-10 and CIFAR-100, we use 10 step optimization to find adversarial examples and step size $\frac{\epsilon}{\sqrt{10}}$ and train models for 200 epochs. For MNIST TRADES training, we use $\beta = 1$ and for CIFAR-10 and CIFAR-100, we use $\beta = 6$. Additionally, for CIFAR-10 and CIFAR-100, we optimize the model using SGD with learning rate and learning rate scheduling from Gowal et al. [2020]. For MNIST, we use learning rate 0.01.

Architectures used For CIFAR-10 and CIFAR-100, we report results from training a WRN-28-10 architecture. For MNIST, we train a small CNN architecture consisting of 2 convolutional layers, each followed by batch normalization, ReLU, and 2 by 2 max pooling. The first convolutional layer uses a 5 by 5 convolutional kernel and has 20 output channels. The second convolutional layer also uses a 5 by 5 kernel and has 50 output channels. After the set of 2 convolutional layers with batch normalization, ReLU, and pooling, the network has a fully connected layer with 500 output channels followed by a fully connected classifier (10 output channels).

E Additional Experimental Results

E.1 Impact of hyperedges

In Figure 4, we show the count of edges, degree 3 hyperedges, and degree 4 hyperedges found in the conflict hypergraphs of the MNIST, CIFAR-10, and CIFAR-100 train sets. We note that we did not observe any increase in loss when considering degree 4 hyperedges at the ϵ with a data point for number of degree 4 hyperedges in Figure 4. We find that the relative number of edges and hyperedges is not reflective of whether we expect to see an increase in loss after considering hyperedges. For example in CIFAR-10, at $\epsilon = 4.0$, we there are about 100 times more hyperedges than edges, but we see no noticeable increase in the $0-1$ loss lower bound when incorporating these hyperedge constraints.



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

Figure 4: Number of edges, degree 3 hyperedges, and degree 4 hyperedges found in the conflict hypergraphs of MNIST, CIFAR-10, and CIFAR-100 train sets. The red vertical line indicates the ϵ at which we noticed an increase in the $0-1$ loss lower bound when considering degree 3 hyperedges.

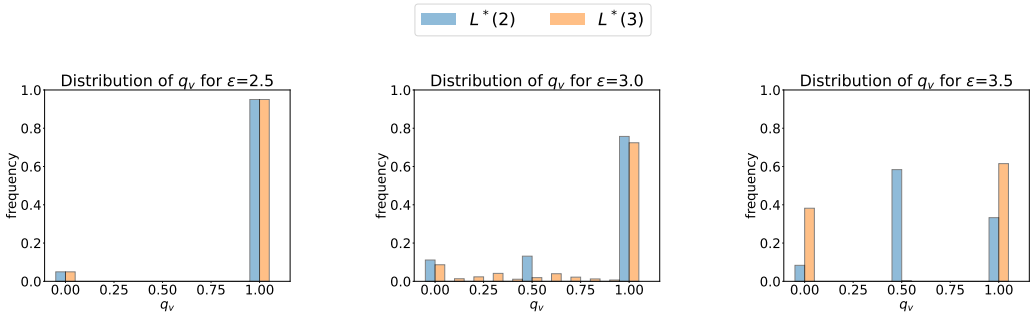
We further examine the optimal classification probability q_v of each vertex obtained as a solution to the LP with ($L^*(3)$) and without considering constraints from degree 3 hyperedges ($L^*(2)$). We show histograms of the distributions of q_v for MNIST in Figure 5 and for CIFAR-10 in Figure 6. For small ϵ, there is no change in the distribution of q_v, which results in no change in overall loss. For example, for MNIST at $\epsilon = 2.5$ and CIFAR-10 at $\epsilon = 3.5$, we observe that the distribution of q_v stays the same between $L^*(2)$ and $L^*(3)$. However at larger values of ϵ, we find that for $L^*(2)$, more values are assigned q_v near 0.5, and these probabilities are highly impacted after incorporating hyperedge constraints.

E.2 Computational complexity of computing lower bounds

Our experiments of $L^*(3)$ for higher ϵ are limited due to computation constraints. From Figure 7 we see that the time taken to compute the losses grows exponentially with ϵ. We are seeking algorithmic optimization to achieve more results at high ϵ.

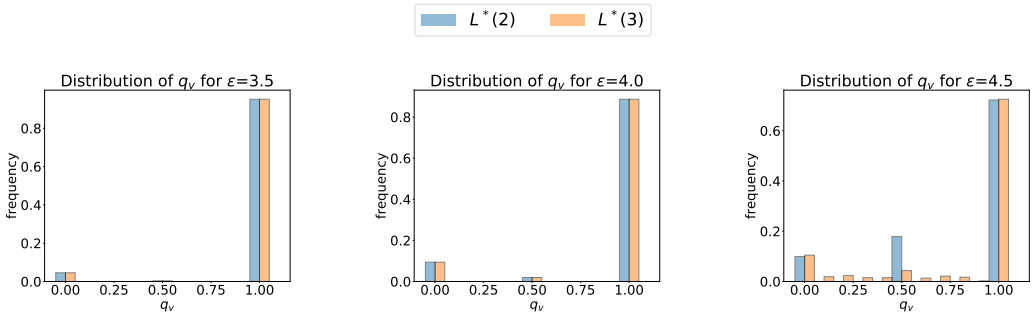
E.3 Clean errors of adversarially trained models

We report the training error of adversarially trained models shown in Figure 3 on unperturbed inputs in Figure 8 (denoted as clean err) in comparison to the robust error. We find that in general, TRADES-AT finds models with much lower clean error compared to PGD-AT, but for CIFAR-10 and CIFAR-100



(a) $\epsilon = 2.5$ (b) $\epsilon = 3.0$ (c) $\epsilon = 3.5$

Figure 5: Distribution of optimal classification probabilities q obtained by solving the LP with up to degree 2 hyperedges ($m = 2$) and up to degree 3 hyperedges ($m = 3$) on the MNIST training set.



(a) $\epsilon = 3.5$ (b) $\epsilon = 4.0$ (c) $\epsilon = 4.5$

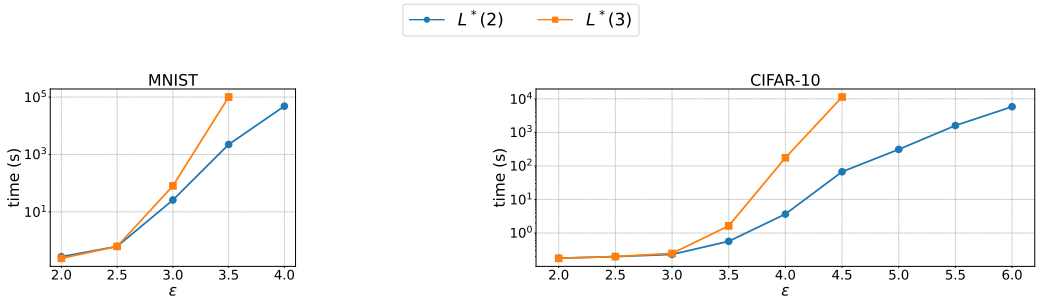
Figure 6: Distribution of optimal classification probabilities q obtained by solving the LP with up to degree 2 hyperedges ($L^*(2)$) and up to degree 3 hyperedges ($L^*(3)$) on the CIFAR-10 training set.

this does not equate to finding more robust models. Tuning of the β hyperparameter of TRADES may improve the robust loss on CIFAR-10 and CIFAR-100.

E.4 Results for Gaussian Data

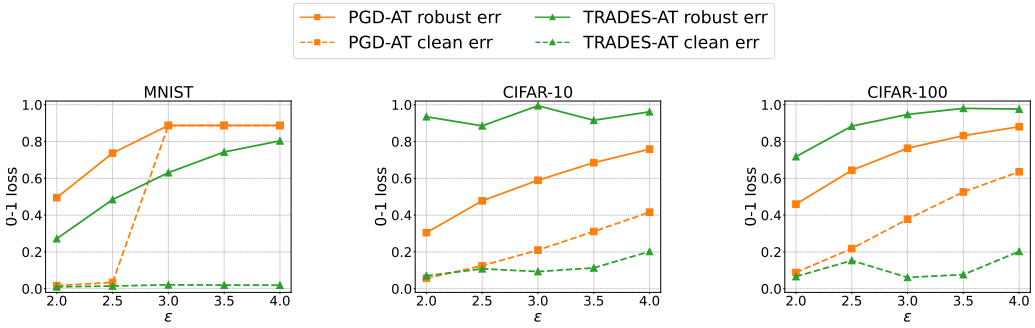
In this section, we consider a 3-class problem on 2 dimensional input. We sample data from 3 spherical Gaussians with means at distance 3 away from from the origin (a sample is shown in Figure 9). We compute multiclass lower bounds on robust accuracy at various ℓ_2 budget ϵ. We compute 2 sets of lower bounds: lower bounds considering only pairwise edges ($L^*(2)$) and lower bounds considering up to degree 3 hyperedges ($L^*(3)$). Additionally, we compare to a deterministic 3 way classifier shown in black in Figure 9. This classifier classifies incorrectly, when a sample lies over the edge of the nearest ϵ margin of the classifier.

We report results for the 3-class Gaussian classification problem in Figure 10. We find that generally until the multiclass pairwise lower bound on $0-1$ error reaches 0.5, the difference between the lower bound considering only pairwise edges and lower bound considering up to degree 3 hyperedges is negligible. Similarly, we find that up until the $0-1$ lower bound reaches 0.5, we find that the error of the deterministic classifier also lies close to the computed lower bound. This suggests that our lower bound is not vacuous for deterministic classifiers.



(a) MNIST (b) CIFAR-10

Figure 7: Time taken to compute $L^*(2)$ and $L^*(3)$ for MNIST and CIFAR-10.



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

Figure 8: Robust and clean error on the training set for adversarially trained models in Figure 3

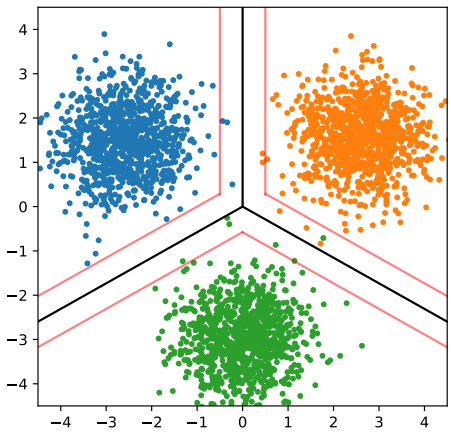
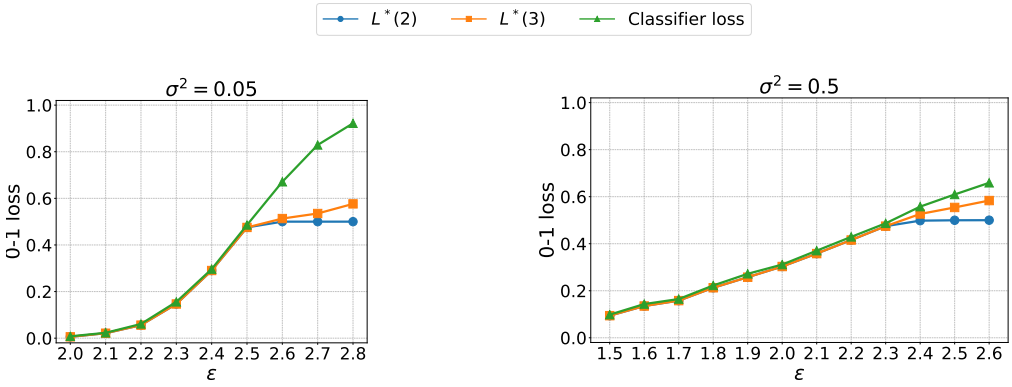


Figure 9: A sample 3-class gaussian problem (each color pertains to a class) and a corresponding classifier for this problem shown in black. The classifier classifies a sample incorrectly when it lies over the edge of the ϵ margin (shown by the red lines) nearest the corresponding gaussian center.



(a) $\sigma^2 = 0.05$ (b) $\sigma^2 = 0.5$

Figure 10: Lower bounds on error for the Gaussian 3-class problem (for variances $\sigma^2 = 0.05$ and $\sigma^2 = 0.5$) computed using only constraints from pairwise edges ($L^*(2)$) and up to degree 3 hyperedges ($L^*(3)$) in comparison to the performance of the deterministic 3-way classifier depicted in Figure 9.