Robust Naive Bayes

Aditya Mishra

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-66
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-66.html

May 13, 2021

Acknowledgement

First, I would like to thank my advisor Laurent El Ghaoui for supervising this project and Professor Somayeh Sojoudi for serving as a second reader. Next, I would like to thank Anand Siththaranjan and Armin Askari for being wonderful collaborators on this project. Finally, I would like to thank my family for their support throughout my years at Berkeley.

Robust Naïve Bayes

by Aditya Mishra

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Laurent El Ghaoui
Research Advisor

5/03/2021

(Date)

* * * * * * *

Professor Somayeh Sojoudi
Second Reader

4/20/2021

(Date)

Robust Naive Bayes

Aditya Mishra *

Abstract

Robustness of deep learning methods remains an open issue in a variety of NLP tasks due to the inherent complexity of neural networks. In this paper, we focus on a simple, yet effective model for large-scale text classification: Multinomial Naive Bayes (MNB). In this work, we derive the robust counterpart to MNB, *Robust Naive Bayes* (RNB), in different adversarial settings that are relevant to text. We compare the robustness of our model against SVM, logistic regression and neural networks in a variety of settings. Our results show that RNB is comparable to other models under random perturbations but vastly outperforms them against targeted attacks. We describe an algorithm for training our model which is orders of magnitude faster than the training time of more complex models.

1 Introduction

Deep neural networks and transformer models are the current state-of-the-art for text classification problems. However, a key concern surrounding these models is the potential lack of robustness in the presence of noise and ill-intentioned adversaries affecting the data. For instance, Iyyer et al. (2018) show that syntactic transformations can drastically reduce the accuracy of bidirectional LSTMs, while Jin et al. (2020) are able to generate semantically and syntactically similar examples to training data that drastically reduce the test performance of BERT.

Since robustness is difficult to analyze for highly complex models, we turn our attention to a much simpler model: the Multinomial Naive Bayes (MNB) classifier, a model that has persisted in text classification for its ease of implementation. We pose the following question: *Can a robustly trained MNB model outperform complex models when subject to random or adversarial attacks?* For one instance of a structured adversary, we give an affirmative answer. In this paper:

- We formulate a Robust Naive Bayes problem that accounts for the worst-case noise under a typo-based uncertainty model.
- We present a fast algorithm that scales linearly (up to logarithmic factors) with the dimensionality of the data points.
- We show that our Robust Naive Bayes classifier not only trains faster, but significantly outperforms other models when the data is subject to adversarial perturbations.

1.1 Related Works

Robust classification is a well-studied topic for traditional machine learning models such as support vector machines, logistic regression and decision trees (Bertsimas et al., 2019), in which a $\min\max$ formulation is used (Ben-Tal et al., 2009). These techniques offer the opportunity to consider uncertainty under worst-case and/or probabilistic assumptions, and improve the robustness of algorithms against real-world data perturbations. In our work we employ the robust optimization methodology to MNB with an uncertainty set that is actually meaningful in a text classification settings (as opposed to ℓ_p-norm ball uncertainties).

A comprehensive study of robust deep learning for problems in natural language processing is explored in (Wang et al., 2019). This work provides a characterization of different adversarial examples and the constraints they obey. They note a variety of techniques including those based on fast-gradient sign-method (FGSM), optimization-based and importance-score based approaches. Attack types are also varied, including character-, word-

*Please see Section D of the appendix for acknowledgement of collaborators and their contributions.

and sentence-level attacks that undergo a mixture of inserting, flipping, swapping (for character-level attacks), replacing (for word-level attacks) and removing features. In our work, we explore word-level attacks for frequency-based encodings of sentences (that is, count vectors) by considering typo-based uncertainties (i.e. perturbations that insert, replace or remove words from the original sentence).

A complete survey on Naive Bayes and its extensions can be found in (Jiang et al., 2007). Of particular interest are works that try to make Naive Bayes models more robust. There have been various works making naive Bayes classifiers robust to different types of noise. (Ramoni and Sabastini, 2001) formulated a naive Bayes model that takes into account missing entries, both in the form of data points not having a label, and labels not having a corresponding data point. In our work, we specifically consider a noise model where words are added or removed from sentences.

2 Background on Naive Bayes

2.1 Notation

We are given a non-negative data-matrix of count vectors $X = [x^{(1)}, \ldots, x^{(n)}]^\top \in \mathbb{N}^{n \times d}$, and a vector $y \in \{-1, 1\}^n$ that encodes the class information for the n data points of dimension d. Furthermore, let $C_\pm$ denote the positive and negative class respectively. Unless otherwise specified, functional operations (such as $\max(x, y)$) on vectors are performed element-wise. We also define

$$\mathcal{I}_\pm := \{i | y^{(i)} = \pm 1\}, \quad f_\pm := \sum_{i \in \mathcal{I}_\pm} x^{(i)}$$

Finally, $\mathcal{P} = \{\theta \mid \theta \in [0, 1]^d, \ \mathbf{1}^\top \theta = 1\}$ denotes the d-dimensional simplex. Throughout the paper, we will use terms count vectors and sentences interchageably.

2.2 Naive Bayes

Here, we wish to predict the class label of a test point $x \in \mathbb{N}^d$ via $\hat{y}(x) = \arg\max_{\epsilon \in \{+,-\}} P(C_\epsilon | x)$. To calculate the posterior probability, we use Bayes' Rule and the "naive" assumption that features are conditionally independent, leading to the prediction rule

$$\hat{y}(x) = \arg \max_{\epsilon \in \{+,-\}} \log P(C_\epsilon) + \sum_{j=1}^{m} \log P(x_j | C_\epsilon)$$

The above equation requires an explicit model for $P(x_j | C_\epsilon)$. Throughout this paper, we use the multinomial distribution and determine the optimal parameters of the distribution via maximum likelihood estimation.

2.3 Multinomial Naive Bayes

With integer-valued features, we parameterize the conditional distribution $P(x_j | C_\epsilon)$ with two non-negative d-dimensional vectors $\theta^\pm \in \mathcal{P}$. The probability distribution $P(x^{(i)} | C_\pm)$ is

$$P(x | C_\pm) = \frac{(\sum_{j=1}^d x_j)!}{\Pi_{j=1}^d x_j!} \prod_{j=1}^d (\theta_j^\pm)^{x_j}$$

and the maximum likelihood estimator $\theta_*^\pm$ is

$$\theta_*^\pm = \arg \max_{\theta^\pm \in \mathcal{P}} (f^\pm)^\top \log \theta^\pm \tag{1}$$

The solution to (1) is $\theta_*^\pm = \frac{f^\pm}{\mathbf{1}^\top f^\pm}$, and we get the following classification rule:

$$\hat{y}(x) = \textbf{sign} \left(\log \frac{P(C_+)}{P(C_-)} + \left(\log \frac{\theta_*^+}{\theta_*^-} \right)^\top x \right)$$

We refer the reader to Askari et al. (2020) for further details on the derivation of the model.

3 Robust Classification

Here we outline our main contribution; the robust multinomial naive bayes (RNB) model under different types of noise. We also show how to train other classifiers under the same uncertainty sets.

3.1 Constructing Robust classifiers

In the problem of robust classification, we seek to find a classifier that operates well under a certain distribution shift. In order to learn a robust classifier, we solve the following optimization problem:

$$\theta^* = \arg \max_\theta \min_{U \in \mathcal{U}(X)} \mathcal{L}(U, y)$$

where $\mathcal{L} : \mathbb{R}^{n \times d} \times \mathbb{R}^n \to \mathbb{R}$ is a loss function that takes in a data matrix U and labels y; $\mathcal{U} : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ returns a set of noisy perturbation around our nominal data matrix X, according to our uncertainty set; and θ are the parameters of our classifier. This traditional approach to robust classification utilises the inner problem to consider the worst-case uncertainty in our data to learn a

classifier that is resistant to these adversarial settings. Traditionally, the robust training problem is a $\min\max$ problem but since our original problem is a maximization problem it is instead a $\max\min$.

We assume that the uncertainty affects each data point *independently*. This leads to uncertainty sets of the form

$$\mathcal{U}(X) = \left\{ [u_1, \ldots, u_n] \in \mathbb{R}^{n \times d} : u_i \in \mathcal{V}(x_i) \right\}$$

where a set $\mathcal{V}(x) \subseteq \mathbb{R}^d$ models the uncertainty on a generic data point $x \in \mathbb{R}^d$. We study on two different types of uncertainty sets; a *shift uncertainty* for a real-valued vector $x \in \mathbb{R}^d$

$$\mathcal{V}_{\text{shift}}(x) = \{\delta \ : \ \delta + x \in \mathbb{N}^d, \delta \in [-\gamma, \gamma]^d \cap \mathbb{Z}^d\}, \tag{γ-shift}$$

and a *typo uncertainty* for an integer valued count vector $x \in \mathbb{N}^d$

$$\mathcal{V}_{\text{typo}}(x) = \{\delta \ : \ \delta + x \in \mathbb{N}^d, \|\delta\|_1 \leq \kappa\} \tag{κ-typo}$$

$\mathcal{V}_{\text{typo}}(x)$ models the fact that count vectors can be altered by adding or subtracting at most κ counts to x. In Section 4, we parameterize $\kappa = t\|x\|_1$ where $t \geq 0$ denotes the proportion of typos as a percentage and $\|x\|_1$ denotes the number of words in the sentence x.

In the remainder of this section, we consider the following robust optimization problem under various uncertainty sets $\mathcal{D}_{\pm}$:

$$\max_{\theta_{\pm} \in \mathcal{P}} \min_{\delta_{\pm} \in \mathcal{D}_{\pm}} \sum_{\epsilon \in \{+,-\}} (f_\epsilon + \delta_\epsilon)^\top (\log \theta_\epsilon)$$

where $\mathcal{D}_{\pm}$ denotes either the shift or typo uncertainty for the positive and negative class accordingly. Also note that the optimization problem is separable in terms of the pairs of variables (θ_+, δ_+) and (θ_-, δ_-), so we focus on efficiently solving the optimization problem:

$$\max_{\theta \in \mathcal{P}} \min_{\delta \in \mathcal{D}} (f + \delta)^\top (\log \theta) \tag{2}$$

We first show an interesting connection between the (γ-shift) uncertainty set and laplace smoothing for MNB. The rest of the section is then focused on the (κ-typo) uncertainty set since this is the uncertainty set we consider for our experiments in Section 4.

RNB with shift uncertainty. Under the shift uncertainty set (γ-shift), we can derive the representative RNB model.

Lemma 3.1. *The robust counterpart of MNB under the uncertainty set (γ-shift) can be cast as the following optimization problem:*

$$\max_{\theta_{\pm} \in \mathcal{P}} \sum_{\epsilon \in \{+,-\}} (f_\epsilon + |C_\epsilon|\gamma \mathbf{1})^\top (\log \theta_\epsilon)$$

which has closed form solution

$$\theta_*^{\pm} = \frac{f_{\pm} + |C_{\pm}|\gamma \mathbf{1}}{\sum_{i=1}^m f_{\pm,i} + |C_{\pm}|\gamma}$$

where $|C_+|$ and $|C_-|$ are the number of data points for the positive and negative classes respectively.

Proof. See section C.6 of the Appendix. □

If we consider $|C_+| = |C_-|$ and $\gamma = \frac{\gamma'}{|C_+|}$, then the solution is of the same form as that when Laplace smoothing with parameter γ' is used in the Multinomial Naive Bayes solution. Alternatively, we could also considered two different parameter $\gamma_{\pm} = \frac{\gamma'}{|C_{\pm}|}$ for each class to find this solution. This showcases the observed relationship between regularization and robustness explored in regression methods when a prior is enforced (Bertsimas and Copenhaver, 2017). In fact, Laplace smoothing is equivalent to enforcing a Dirichlet prior on θ (see Section C.7 of the Appendix).

RNB with typo uncertainty set Under the typo uncertainty (κ-typo), we can derive the representative RNB model.

Lemma 3.2. *The robust counterpart of MNB (2) under the uncertainty set (κ-typo) can be solved via the scalar convex optimization problem*

$$\max_{v > 0} \sum_{i=1}^{d} (f_i \log \max(f_i, v) - \max(f_i, v)) - \kappa \log v \tag{3}$$

The optimal estimate θ_* can be recovered via

$$\theta_{*,i} = \frac{\max\{f_i, v_*\}}{\kappa + \mathbf{1}^\top f}$$

and the optimal adversary is given by

$$\delta_{*,i} = \begin{cases} \kappa & \text{if } i = \arg\min_j \theta_j \\ 0 & \text{o.w} \end{cases}$$

Proof. See Section C.1 of the Appendix. □

Algorithm 1 solves Eq 3 with complexity $\mathcal{O}(nd + d \log d)$. The key idea is to split the domain of the scalar variable v into $n+1$ intervals based off of the f_i's and then find a closed form maximizer v_k for each interval (see Section C.2 of the Appendix).

Algorithm 1: Solution to (3)

Input: A sorted vector $f \in \mathbb{R}^d$; $\kappa \in \mathbb{R}_{\geq 0}$
Output: A scalar $v^* \in \mathbb{R}_{>0}$ minimizing Equation 3
Define $\rho_k = \kappa + \sum_{i=k+1}^{n} f_i$, and $h_k = \sum_{i=1}^{k} f_i \log f_i - f_i$;
Initialize $\rho_0 = \kappa + \mathbf{1}^\top f$, $v_0 = \max(f_1, \frac{\rho_0}{n})$, and $h_0 = 0$;
Set $v^* = v_0$;
for *i = 1 . . . n* **do**
  Set $\rho_k = \rho_{k-1} - f_k$, $h_k = h_{k-1} + f_k \log f_k - f_k$;
  Set $v_k = \min(f_k, \max(f_{k+1}, \frac{\rho_k}{n-k}))$ and $F_k = h_k + \rho_k \log v_k - (n-k)v_k$;
  If $F(v_k) > F(v^*)$, set $v^* = v_k$;
end
return v^*;

3.2 Robust Logistic Regression and Support Vector Machine

In the cases of using ℓ_2-regularized logistic regression (LR) and soft-margin ℓ_2-regularized support vector machine (SVM) classifiers, we can consider the following problems:

$$\min_{\beta,\beta_0} \max_{\delta_i \in \Delta_i} \sum_{i=1}^{n} \log(1 + e^{-y_i(\beta^\top(x_i+\delta_i)+\beta_0)}) + \lambda(\|\beta\|_2^2 + \beta_0^2) \quad (4)$$

$$\min_{w} \max_{\delta_i \in \Delta_i} \frac{1}{n} \sum_{i=1}^{n} \max\{0, z_i\} + \lambda \|w\|_2^2 \quad \text{s.t.} \quad z_i = 1 - y_i((x_i + \delta_i)^\top w - b) \quad (5)$$

where the uncertainty set is defined by

$$\delta_i \in \Delta_i = \{\delta_i : x_i + \delta_i \geq 0, \ , \|\delta_i\|_1 \leq \kappa_i\} \quad (6)$$

We consider both problems jointly due to the similar nature by which the inner maximization problem is solved. Additionally, we only consider the the uncertainty set (κ-typo) since it is more interpretable for the experiments in Section 4.

Lemma 3.3. *The robust counterpart under the uncertainty set* (κ-typo) *for* ℓ_2*-regularized logistic regression can be cast as*

$$\min_{\beta,\beta_0,\lambda_i \geq 0} \sum_{i=1}^{n} \log(1 + \exp(-y_i(\beta^\top x_i + \beta_0) + \kappa_i \|y_i\beta - \lambda_i\|_\infty + \lambda_i^\top x_i) + \lambda(\|\beta\|_2^2 + \beta_0^2).$$

For soft-margin ℓ_2*-regularized SVM the robust problem reads*

$$\min_{w, z_i \geq 0, \lambda_i \geq 0} \frac{1}{n} \sum_{i=1}^{n} z_i + \lambda \|w\|_2^2 \quad \text{s.t.} \quad z_i \geq 1 - y_i(x_i^\top w - b) + \kappa_i \|y_i w - \lambda_i\|_\infty + \lambda_i^\top x_i$$

both of which are convex. The optimal adversary δ_i^* *for both models is found using Algorithm 2.*

Proof. See Sections C.3, C.4 of the Appendix. □

We present a greedy method of solving for the optimal adversary in in the inner optimization of (4) and (5) in Algorithm 2 (see Section C.5 of the Appendix), which has a runtime of $\mathcal{O}(d \log d)$. The same algorithm is able to solve for both adversaries due to the linear structure of the inner maximization. Though the robust counterpart presented in Lemma 3.3 are both convex programs, they are difficult to solve efficiently at scale. We consider this beyond the scope of this work.

4 Experiments

We now study the robustness and training times of different classifiers with the (κ-typo) uncertainty set. We show that subject to targeted attacks, our RNB classifier outperforms other common NLP classfication techniques, while having comparable performance against random perturbations.

4.1 Experimental Set Up

The models we consider are RNB, RNB where the noise parameter κ is misspecified by 20% (RNB±20), the classical MNB, ℓ_2-regularized support vector machine (SVM), ℓ_2-regularized logistic regression (LR), a multi-layer perceptron (MLP) with 3 layers, and a pretrained $\text{BERT}_{\text{CASE}}$ model (Devlin et al., 2019). We consider five different datasets: Amazon (Ni et al., 2019), SST2 (Socher et al., 2013), MPQA (Deng and Wiebe, 2015), MR

Dataset	Vocabulary Size	n_{train}	n_{test}	Avg Document Length
Amazon (Ni et al., 2019)	34816	7997	2000	35.8
SST2 (Socher et al., 2013)	16893	63723	15931	4.8
MPQA (Deng and Wiebe, 2015)	5952	5299	1325	1.8
MR (Pang and Lee, 2005)	19587	8529	2133	17.9
Spam (Almeida et al., 2011)	6958	4456	1115	7.9

Table 1: Details of dataset used, containing the size of the vocabulary (ie. number of unique words), the size of the training set (n_{train}), the size of test set (n_{test}) and the average document length after pre-processing across both train and test set (to 1 decimal places).

(Pang and Lee, 2005) and Spam (Almeida et al., 2011). We preprocess each dataset and transform the data into count vectors (see Appendix A). Only basic preprocessing was done and as a result the post processed data contains typos and mispellings. The size of the datasets can be found in Table 1.

4.2 Comparing Robust Classifiers

We evaluate the robustness of the different models subject to (κ-typo) uncertainty. We do this by perturbing the datasets randomly, training our models on this perturbed data and then evaluating the performance at test time either on randomly or adversarially perturbed data. A random perturbation means for each data point, we choose an index at random, randomly decide if we are going to add or subtract a count from it, and repeat this process until the total number of changes made is κ. For each data point, the total number of words we add/remove is proportional to the length of the sentence (e.g. if the length of a sentence is 100 words, a proportion of 0% means we do not alter the sentence while 50% indicates we add/remove up to 50 words). An optimal perturbation means given a model and proportion, we find the best perturbation of the data point by solving an optimization problem (see Section 3). Since computing the optimal adversary is not possible in general for neural networks, we compute a pseudo-adversary by sampling ten random perturbations and taking the one that maximizes the loss at test time.

In order to make the comparison between RNB and the other models fair, we use ℓ_2-regularized SVM and LR models (since regularized models reduce over fitting and are known to exhibit robustness properties) and we use data augmentation for training the neural network models. Specifically, for MLP and BERT, we take our training set, perturb it randomly two times as explained above, and then use that augmented dataset in order to train our neural networks.

In Figures 1 and 2, we plot the performance of the models as we increase the strength of our perturbations at training and test time. For all models and proportions, we randomly perturb the training set 10 times and we perform 3-fold cross validation each time for tuning hyperparameters. We only randomly perturb the training set three and two times for MLP and pretrained $\text{BERT}_{\text{CASE}}$ respectively, and use fixed hyperparameters (see Section B for more details).

Figure 1 shows that under an optimal adversary, RNB significantly outperforms other models for high noise levels. In particular, the accuracy of all the other models steeply drops down for sufficiently high proportions of typos while RNB declines gradually. Figure 2 shows that RNB gives comparable performance to other models (in particular the 3-layer MLP and pretrained BERT models) when subject to a random adversary. We emphasize that the important takeaway from Figures 1 and 2 is not the accuracy values themselves but rather *the trends* of the different models. In particular, the MLP and BERT models could have been trained to attain higher overall accuracy, but our main focus is the degradation in accuracy as the proportion of typos increases. Even when the noise parameter is mispecified by 20%, RNB still significantly outperforms the other models.

The high variance of the MNB models in Figure 1 can be attributed to the interplay of two factors; the MNB model itself and the sparsity level/dimensionality of the datasets. As per Lemma 3.2, the optimal adversary for the MNB model puts all its weight on the minimum entry of a discrete probability distribution. From (1), we see that $\theta_*^{\pm}$ is directly proportional to f. For an extremely sparse dataset, where some words may only appear once, it is possible that the index corresponding to the minimum entry of $\theta_*^{\pm}$ is not unique. As a result, when constructing our adversary we simply choose

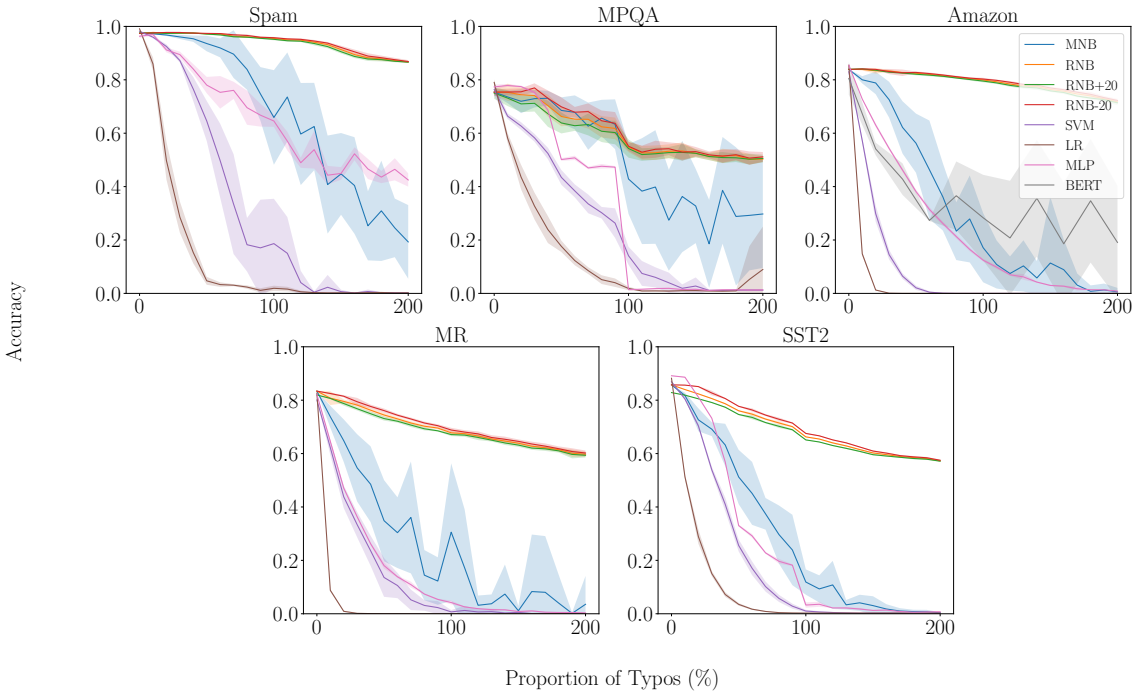

Figure 1: Test accuracy on the various datasets. An **optimal** adversary was used to perturb the dataset at test time for different proportions of typos.

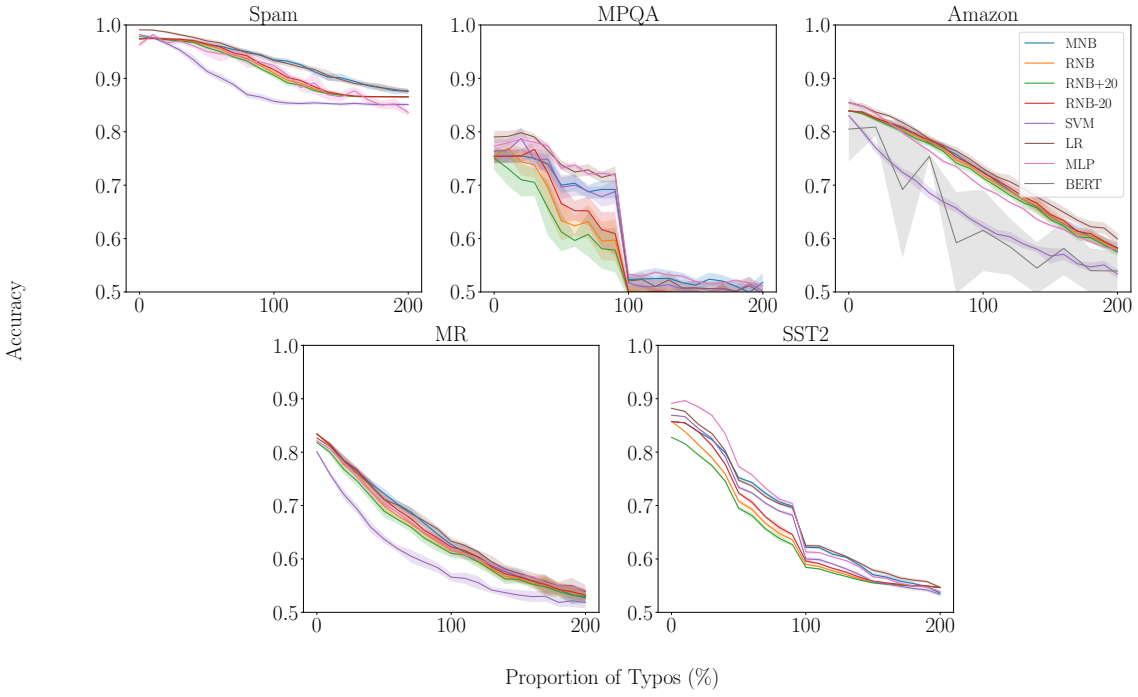

Figure 2: Test accuracy on the various datasets. A **random** adversary was used to perturb the dataset at test time for different proportions of typos. Note that the y-axis has a range of [0.5,1.0].

	Amazon	MPQA	SST2	MR	Spam
MNB	0.67(0.11)	0.08(0.02)	5.22(5.38)	0.3409(0.01)	0.07(0.0007)
RNB	1.69(0.0831)	0.20(0.01)	6.59(3.42)	0.9531(0.02)	0.21(0.0022)
LR	22.06(1.68)	1.37(0.08)	58.47(2.91)	22.66 (0.33)	0.78(0.0032)
SVM	16.29(5.93)	4.46(0.51)	351.03(49.06)	8.479(0.18)	2.06(0.29)
3 Layer MLP	155.15(43.48)	82.30(40.82)	992.25(138.15)	116.10(27.97)	75.02(31.22)
Pretrained BERT	802.79(2.10)	-	-	-	-

Table 2: Mean training time (one standard deviation in parentheses) in seconds for each model.

Classifier	Sentence	*Removed*/**Added**
RNB	‘3d rocks this video is incredible in 3d it was a gift for a relative and he absolutely loves it viewed on a 3d tv and 3d bluray player it was just like going to a movie on the big screen’	**000** x2
	‘noise pollution every time i hear that song one wish it makes me want to throw up i hope he never makes another cd’	**000**
MNB	‘wonderful this is a wonderful story it left me thinking instead of my usual forgetting about it 5 minutes later the characters are intriging and the story is interesting making me not want to put it down i being only in high school would definitly suggest that you read it’	**001** x2
	‘not safe and no fun once you get this toy in your hands you can see the cheap constructionthe buttons dont work smoothly and the eggtops are easy to pull off just the right size for my toddler to pop in his mouth i chucked this one right in the garbage’	**000** x2
LR	‘clean excellent copy and shipped fast this is one of the books that i have been waiting to read clean book’	**boring**
	‘i couldnt get into it i just coulafter getting more than 50 through the book i had to stop i just couldnt get into it and didnt like the writting style’	**excellent**
SVM	‘great movie i thought it was a very entertaining movie should have gotten more press when it came out very good acting also’	*great*
	‘this is high density module this is high density module it works only with a fewmainboards it does not with the two i haveamazoncom please correct me if i am wrong’	**great**
MLP	‘clean excellent copy and shipped fast this is one of the books that i have been waiting to read clean book’	*excellent* **thanked**
	‘fort apache the movie is old fashioned and not well acted it was melodrmatic and artificialthe supplier of the dvd was prompt and professional’	*supplier* **collecting**
BERT	‘its nice i was replacing my old cook book i thought this was the same one but it isnt but it is a nice cook book’	*but* **bucked**
	‘not kindle ready its hard to put together any nice words about this transcription it is not kindle ready although it is textually complete should be a freebie’	*this freebie*

Table 3: Sentences and their adversarial perturbation at 10% error strength for each classifier and class on the Amazon dataset. **Bold** indicates an addition, *italics* indicate a removal, and unless otherwise stated these refer to a single addition/removal respectively. *(Top)* Examples from positive class. *(Bottom)* Examples from negative class.

one at random which in turn results in the accuracy greatly varying across different trials at test time. Note that regularization (via Laplace smoothing) does not solve this problem while other forms of regularization may help.

4.2.1 Adversarial Examples

We present a handful of adversarial examples for various classifiers in Table 3. We considered optimal perturbations acting at 10% strength where the predicted class label changed.

Note that for RNB and MNB, the optimal adversary is *the same* for every test point in the same class (positive or negative) since the adversary is constructed solely based on the minimum index of $\theta_*^{\pm}$ (see Lemma 3.2). This is not the case for logistic regression, SVM and neural networks (see Sections C.3 and C.4 in the Appendix). This is because for the RNB and MNB models, the optimal adversary at test time is *only* a function of the learned parameters and label and *not* the data point itself, whereas in the other models the adversary is additionally a function of the test point itself. This is a result of the fact that RNB/MNB are trained on the aggregate statistics of the data ($f^{\pm}$) instead of individual data points (x_i) which results in this observed robust behaviour. This is also seen in Table 3 where the RNB model coincidentally chooses the same adversarial perturbation (adding the word `000`) for both classes. This also means that if any other examples from the positive or negative class were shown, the optimal perturbation for those examples would also be `000`. On the other hand, the other models have more interesting adversarial examples that are semantically more meaningful and reflective of the drastic performance decrease that the classifiers seen in Figure 1. In particular, Table 3 shows how brittle the more complex models are. For example, for the SVM classifier, simply by removing the word `great` from the sentence, the predicted label changes even if the sentiment of the modified sentence is still clearly positive. Similarly for the MLP, by replacing the word `excellent` with `thanked`, the MLP also changes the label, while it is clear the sentiment has not changed.

We also look at the case when the proportion of typos is set to 200% and consequently our training data is extremely noisy. Consider the following example:

> *"boring my friend likes this movie so i checked it out i didnt like it kind of boring couldnt get through the whole thing fastforward solved the problem"*

At proportion 200%, LR adds the word `excellent` 24 times and removes `boring` twice, which results in a class change. BERT also changes the label but changes the sentence altogether:

> *"mundi dubbed ethnically likes webcam mackellan pointed cure seller drumspleaseit tommy lowry totally withtwelve otheremail protector 70 mckellan coral csection through the whole addict nora solved she funnywe"*

For RNB, the perturbation does not change the label; it adds `000` 26 times, which unlike LR and BERT, preserves the original semantics. As expected, even for extremely high proportions of typos, the RNB model only adds *one unique* word a fixed number of times to a sentence. In this case, `000` is not semantically meaningful and RNB is able to resist against extremely strong adversaries.

4.3 Training Speed

We compare the run time of the different models in Table 2 [1]. We see that the training time of the RNB model is comparable to that of MNB while providing significantly improved accuracy as in Figure 1. The other classifiers take orders of magnitude longer to train than RNB; in the extreme case we see that on the SST2 dataset, the MLP takes 100x longer to train.

5 Conclusion

In this work, we make the case that under a typo uncertainty set, robustly training a Multinomial Naive Bayes model significantly outperforms other models when subject to an adversarial attack. We show that our new classifier is extremely robust both qualitatively and quantitatively against adversaries as compared to much more complex models. We also show that the adversary is the same across all data points in the class (which is not the case for other more complex models), which results in much better performance against adversarial perturbations, while performing as well as other models when subject to a random perturbation. We also show that

[1] All our models were trained on a 72 CPU (2.30 GHz Intel Xeon E5-2686 Processor) machine with 64 GB of RAM. The deep learning models additionally were trained using a single NVIDIA Titan V GPU.

the main complexity in training our model comes from sorting a vector, and as a result trains orders of magnitude faster than other models.

References

Tiago A. Almeida, José María G. Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 11th ACM Symposium on Document Engineering*, DocEng '11, page 259–262, New York, NY, USA. Association for Computing Machinery.

Armin Askari, Alexandre d'Aspremont, and Laurent El Ghaoui. 2020. Sparse weighted naive bayes classifier for efficient classification of categorical data. *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*.

A. Ben-Tal, L. El Ghaoui, and A.S. Nemirovski. 2009. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press.

Dimitris Bertsimas and Martin S. Copenhaver. 2017. Characterization of the equivalence of robustification and regularization in linear and matrix regression.

Dimitris Bertsimas, Jack Dunn, Colin Pawlowski, and Ying Daisy Zhuo. 2019. Robust classification. *INFORMS Journal on Optimization*, 1(1):2–34.

Lingjia Deng and Janyce Wiebe. 2015. MPQA 3.0: An entity/event-level sentiment corpus. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1323–1328, Denver, Colorado. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Liangxiao Jiang, Dianhong Wang, Zhihua Cai, and Xuesong Yan. 2007. Survey of improving naive bayes for classification. *International Conference on Advanced Data Mining and Applications*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *International Conference on Learning Representations*.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, and V Dubourg. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*.

Marco Ramoni and Paola Sabastini. 2001. Robust bayes classifiers. *Artificial Intelligence*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Wenqi Wang, Benxiao Tang, Run Wang, Lina Wang, and Aoshuang Ye. 2019. A survey on adversarial attacks and defenses in text. *CoRR*, abs/1902.07285.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. pages 38–45, Online. Association for Computational Linguistics.

A Dataset Preprocessing

We enumerate the preprocessing steps for each of the datasets that have used.

- **Amazon:** The complete Amazon reviews data set was collected from here; only a subset of this data was used which can be found here. This data set was randomly split into 80/20 train/test.
- **SST2:** The Stanford Sentiment Treebank dataset can be found here, and was pre-processed using the code here.
- **MPQA:** The MPQA opinion corpus can be found here and was pre-processed using the code found here. We balance the dataset by generating a random 80-20 ratio between the size of training and test data.
- **MR:** The Movie review dataset can be found here and was pre-processed using the same code as the Amazon dataset.
- **Spam:** SMS Spam Collection dataset can be found here and was pre-processed using the code found here.

B Model Details

B.1 Classical Models

For each of our classical models, we used Sklearn's implementation (Pedregosa et al., 2011). We list out the hyperparameter tuning procedure for each of the models below:

- **RNB/MNB:** We performed a grid search over the Laplace smoothing parameter using values ranging from $\alpha \in [0.5, 9.5]$ in increments of 0.5. The RNB model used the same α as the tuned MNB model.
- **SVM:** We performed a grid search over the regularization parameter using values ranging from $\alpha \in [0.5, 9.5]$ in increments of 0.5.
- **LR:** We evaluated each model with the following values of regularization parameters: $\{0.001, 0.01, 0.1, 1, 10, 100\}$.

We use 3-fold cross validation to determine the best set of hyperparameters for each model.

B.2 Deep Learning Models

B.2.1 3 Layer MLP

We trained a 3 layer fully connected network, with 512 and 128 as the dimensions of the first and second hidden layers respectively. For training, we use an Adam Optimizer with 10^{-4} as the learning rate. We employ batchnormalization and apply dropout ($p = 0.1$) after the second hidden layer. We trained the model for 10 epochs, keeping the model with the best performing validation accuracy.

B.2.2 Pretrained BERT$_{\text{CASE}}$

We used a pretrained BERT model that had a hidden layer with softmax activation from the HuggingFace transformers repository (Wolf et al., 2020). Per the authors' recommendation, we trained the model for 3 epochs, keeping the model with the highest validation accuracy. Due to memory constraints, we trained the model with a batch size of 16 and a maximum padding length of 256. During training, we used a AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of $5 \cdot 10^{-5}$ and $\epsilon = 10^{-8}$.

C Proofs of Section 3

C.1 Proof of Lemma 3.2

Recall that $\mathcal{P}$ is the d-dimensional probability simplex and that $\mathcal{D}$ is the typo uncertainty set (κ-typo). Note since the objective function is linear, we can perform the minimization over the convex hull of the feasible region. Thus, we can set $\mathcal{D} = \{\delta \in \mathbb{R}^d : f + \delta \geq 0, \|\delta\|_1 \leq \kappa\}$.

We have that

$$\begin{aligned} p^*(f, \kappa) &= \max_{\theta \in \mathcal{P}} f^\top \log \theta + \min_{\delta \in \mathcal{D}} \delta^\top \log \theta \\ &= \max_{\theta \in \mathcal{P}} f^\top \log \theta + \kappa \min_{1 \leq i \leq n} \log \theta_i \\ &= \max_{\theta \in \mathcal{P}} f^\top \log \theta + \kappa t \ : \ t\mathbf{1} \leq \log \theta \end{aligned}$$

We form the dual problem:

$$= \min_{\tau} \max_{\theta, t} f^\top \log \theta + \kappa t + \tau(1 - \mathbf{1}^\top \theta) \ : \ \exp(t)\mathbf{1} \leq \theta$$

$$= \min_{\tau} \max_{\theta, u > 0} f^\top \log \theta + \kappa \log u + \tau(1 - \mathbf{1}^\top \theta) \ : \ u\mathbf{1} \leq \theta$$

where we make the substitution $u = \exp(t)$. Now, consider the following problem, given fixed u, τ.

$$q^* = \max_{\theta \in [0,1]^d, u > 0} f^\top \log \theta - \tau \mathbf{1}^\top \theta \ : \ u\mathbf{1} \leq \theta$$

If $\tau \leq 0$, the problem is unbounded. Otherwise, we see that the optimal point is $\theta^* =$

$\max(f/\tau, u\mathbf{1})$. Plugging it back in to our expression for p^*, we have that:

$$q^* = \sum_{i=1}^{n}(f_i \max(\log f_i, \log(\tau u)) - \max(f_i, \tau u)) - (\mathbf{1}^\top f)\log\tau$$

Setting $\rho = \kappa + \mathbf{1}^\top f$ and plugging the expression for q^* back, we simplify $p^*(f, \kappa)$:

$$\max_{\tau>0}\max_{u>0}\sum_{i=1}^{n}[f_i \log\max(f_i, \log(\tau u)) - \max(f_i, \tau u)] - (\mathbf{1}^\top f)\log\tau + \kappa\log u + \tau$$

$$= \max_{\tau>0}\max_{v>0}\sum_{i=1}^{n}[f_i \log\max(f_i, \log v) - \max(f_i, v)] + \kappa\log v + \tau - \rho\log\tau$$

$$= \max_{v>0}\sum_{i=1}^{n}[f_i \log\max(f_i, v) - \max(f_i, v)] - \kappa\log v$$

Note that $\tau^* = \rho$. Going back to our original problem, we know that $p^*(f, \kappa)$ is

$$\max_{\theta,u>0} f^\top \log\theta + \kappa\log u + \tau^*(1 - \mathbf{1}^\top\theta)$$

We conclude that the optimal $\theta_i(f, \kappa) = \max(f_i/\tau^*, v^*/\tau^*)$. Note that finding the optimal adversary is equivalent to solving $\arg\min_{\delta\in\mathcal{D}} \delta^\top \log\theta^*$.

C.2 Proof of Algorithm 1

We consider the following optimization problem.

$$\max_{v>0} F(v) = \sum_{i=1}^{n}(f_i \log\max(f_i, v) - \max(f_i, v)) - \kappa\log v$$

The key idea of the algorithm is to split the domain of v (which is all of $\mathbb{R}_{>0}$) into $n+1$ intervals based off of the f_i's and find a closed form maximizer v_k for each interval. We then output the v_k that maximizes Equation 3.

Suppose without loss of generality that $f_1 \geq f_2 \geq \ldots \geq f_n > 0$. Let $f_0 = \infty$ and $f_{n+1} = 0$. Let $\mathcal{I}_k = [f_{k+1}, f_k]$, and denote $v_k = \text{argmax}_{v\in\mathcal{I}_k} F(v)$.

To prove that the algorithm is correct, it suffices to show the following:

- $v_0 = \max\left(f_1, \frac{\rho_0}{n}\right)$
- $v_n = f_n$
- $v_k = \min\left(f_k, \max\left(f_{k+1}, \frac{\rho_k}{n-k}\right)\right)$

Verifying the stated expressions for F_0, F_n, and F_k is a matter of plugging in the relevant v into the main expression.

We first prove the first statement. Since $v_0 \in \mathcal{I}_0$, we can simplify $F(v)$ as follows:

$$F(v) = -nv + (\kappa + \sum_{i=1}^{n} f_i)\log v$$

Taking derivatives and setting it to 0 yields that the optimal v^* is $\frac{\kappa+\sum_{i=1}^n f_i}{n}$. However, depending on κ, v^* might not be in $\mathcal{I}_0$. In the case that it isn't, it follows that:

$$\kappa + \sum_{i=1}^{n} f_i < nf_1$$

We analyze the derivative $F'(v)$ in the interval $\mathcal{I}_0$. We have that

$$F'(v) = -nv + \frac{\kappa + \sum_{i=1}^n f_i}{n} < \frac{nf_1}{v} - n \leq 0$$

We conclude that F is decreasing on $\mathcal{I}_0$, and so in the case where $v^* \notin \mathcal{I}_0$, we choose our new optimal point as f_1. To account for both these cases, we conclude that $v_0 = \max\left(f_1, \frac{\rho_0}{n}\right)$.

We proceed to prove the second statement. We note that now $v_n \in \mathcal{I}_n = (0, f_n]$. Like before, we can use this information to simply $F(v)$ to:

$$F(v) = \sum_{i=1}^{n}[f_i \log f_i - f_i] + \kappa\log v$$

We see that $F'(v)$ is strictly positive on the interval $\mathcal{I}_n$, so the optimal v_n is therefore f_n.

Finally, we show the last statement. We note that now $v_n \in \mathcal{I}_k = [f_{k+1}, f_k]$. We rewrite $F(v)$ as

$$F(v) = h_k + \rho_k \log v - (n-k)v$$

The optimal v^* is $\frac{\rho_k}{n-k}$. Like before, depending on κ, v^* may not be in $\mathcal{I}_k$. There are two cases we need to consider: $v^* < f_{k+1}$ and $v^* > f_k$. We analyze the derivative $F'(v)$ in each of the two regimes. In the first case, we see that

$$F'(v) = \frac{\rho_k}{v} - (n-k) < \frac{f_{k+1}(n-k)}{v} - (n-k) \leq \frac{f_{k+1}(n-k)}{f_{k+1}} - (n-k) = 0$$

We conclude that F is strictly decreasing, in which case the optimal v^* would be f_{k+1}.

Moving onto the second case, we see that $v^* > f_k$ implies that $\kappa > \sum_{i=k+1}^{n}(f_k - f_i)$. We thus have:

$$\begin{aligned} F'(v) &= \frac{\rho_k}{v} - (n-k) \\ &> \frac{\sum_{i=k+1}^{n}(f_k - f_i) + f_i}{v} - (n-k) \\ &= \frac{(n-k)f_k}{v} - (n-k) \geq 0 \end{aligned}$$

We conclude that F is strictly increasing, in which case the optimal v^* would be f_k since v^* needs to be in $\mathcal{I}_k$. To account for both these cases, we express the optimal v_k as $\min\left(f_k, \max\left(f_{k+1}, \frac{\rho_k}{n-k}\right)\right)$.

C.3 Solving the Robust SVM Problem

Recall the robust counterpart to be the following:

$$\min_w \max_{\delta_i \in \Delta_i} \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i((x_i+\delta_i)^\top w - b)\} + \lambda\|w\|_2^2$$

where the uncertainty set is defined by:

$$\delta_i \in \Delta_i = \{\delta_i : x_i + \delta_i \geq 0\ ,\ \|\delta_i\|_1 \leq \kappa_i\}$$

We proceed first by solving the outermost maximization by noting the order of maximization does not matter:

$$\min_w \frac{1}{n}\sum_{i=1}^{n} \max\{0, \max_{\delta_i \in \Delta_i} 1 - y_i((x_i+\delta_i)^\top w - b)\} + \lambda\|w\|_2^2$$

We can first solve the following problem:

$$\begin{aligned} &\max_{\delta_i \in \Delta_i} 1 - y_i((x_i+\delta_i)^\top w - b) \\ &= 1 - y_i(x_i^\top w - b) - \min_{\delta_i \in \Delta_i} \delta_i^\top (y_i w) \end{aligned}$$

Instead of solving the minimization problem directly we can proceed via duality as we know Slater's condition holds and the problem is convex:

$$\begin{aligned} &\max_{\lambda_i \geq 0} \min_{\|\delta_i\|_1 \leq \kappa_i} \delta_i^\top (y_i w) - \lambda_i^\top (x_i + \delta_i) \\ =&\max_{\lambda_i \geq 0} \min_{\|\delta_i\|_1 \leq \kappa_i} \delta_i^\top (y_i w - \lambda_i) - \lambda_i^\top x_i \\ =&\max_{\lambda_i \geq 0} -\kappa_i\|y_i w - \lambda_i\|_\infty - \lambda_i^\top x_i \end{aligned}$$

We can now write the original robust problem as the following:

$$\min_w \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i(x_i^\top w - b) + \min_{\lambda_i \geq 0} \kappa_i\|y_i w - \lambda_i\|_\infty + \lambda_i^\top x_i\} + \lambda\|w\|_2^2$$

Via an epigraph reformulation of the problem we can equivalently write this as

$$\min_{w, z_i \geq 0, \lambda_i \geq 0} \frac{1}{n}\sum_{i=1}^{n} z_i + \lambda\|w\|_2^2$$

subject to

$$z_i \geq 1 - y_i(x_i^\top w - b) + \kappa_i\|y_i w - \lambda_i\|_\infty + \lambda_i^\top x_i \ \forall i$$

The above robust counterpart is a convex quadratic program, which can be seen by rewriting the ℓ_∞ norm in the constraint with $2d$ linear constraints. Note that solving for the adversary δ_i can be accomplished using Algorithm 2, which is detailed in Section C.5 of the Appendix.

C.4 Solving the Robust Logistic Regression Problem

$$\min_{\beta, \beta_0} \max_{\delta_i \in \Delta_i} \sum_{i=1}^{n} \log(1 + \exp(-y_i(\beta^\top (x_i + \delta_i) + \beta_0))) + \lambda(\|\beta\|_2^2 + \beta_0^2)$$

where the uncertainty set is defined by

$$\delta_i \in \Delta_i = \{\delta_i : x_i + \delta_i \geq 0,\ ,\|\delta_i\|_1 \leq \kappa_i\}$$

Solving the inner maximization is the same as the solving the following:

$$\begin{aligned} &\min_{\delta_i \in \Delta_i} y_i(\beta^\top (x_i + \delta_i) + \beta_0) \\ &= y_i(\beta^\top x_i + \beta_0) + \min_{\delta_i \in \Delta_i} y_i \beta^\top \delta_i \end{aligned}$$

Note that we can solve the minimization problem via the same duality-based approach detailed for

robust SVM (see Section C.3 in the Appendix) as well as the same algorithm to generate the optimal adversary (see Section C.5):

$$\min_{\beta,\beta_0,\lambda_i\geq 0} \sum_{i=1}^{n} \log(1+\exp(-y_i(\beta^\top x_i+\beta_0) + \kappa_i\|y_i\beta-\lambda_i\|_\infty + \lambda_i^\top x_i) + \lambda(\|\beta\|_2^2+\beta_0^2)$$

C.5 Deriving Adversaries for SVM and Logistic Regression

In this subsection, we detail Algorithm 2, which solves the inner optimization for the robust variants of SVM and logisitic regression under typo uncertainty.

Algorithm 2: Generate optimal adversary δ for SVM and logistic regression

Input: Data point $x \in \mathbb{R}^d$; Label y; Weight vector $w \in \mathbb{R}^d$; Proportion Error t
Output: Return optimal adversary δ_i^*
Initialize $\delta = \mathbf{0}_{d\times 1}$, $k = \frac{t}{100}\cdot\|x\|_1$ (error budget), ind, z = sortDescending(wy);
Set $i = 0$;
while $k > 0$ **do**
  instructions;
  if $|z_i| = 0$ **then**
    break;
  else
    if $|z_i| < 0$ **then**
      $\delta_{\text{ind}_i} = k$;
      break;
    else
      $\alpha = \max\{-1\cdot x_i, -k\}$;
      $\delta_{\text{ind}_i} = \alpha$;
      $k = k - \alpha$;
    end
  end
  $i \leftarrow i + 1$;
end

C.6 Proof of Lemma 3.1

Note that the lemma at hand can be proved by solving a problem of the following form:

$$\max_{p\in\mathcal{P}} \min_{\delta\in\mathcal{V}_{\text{shift}}(f)} (f+\delta)^\top \log\theta$$

which can be seen due to separability across $\theta^{\pm}$, $f_{\pm}$ and $\delta_{\pm}$. Noting that the inner minimization problem is a linear program over a non-convex set, we can equivalently undergo the minimization over the convex hull of the set:

$$\mathbf{Co}(\mathcal{V}_{\text{shift}}(f)) = \{\delta : \delta + f \geq 0,\ \|\delta\|_\infty \leq \gamma\}$$

This allows the inner minimization to be easily solved, where we take advantage of the fact that $\log(\theta) \leq 0$ and $f \geq 0$:

$$\min_{\delta\in\mathbf{Co}(\mathcal{V}_{\text{shift}}(f))} \delta^\top \log\theta = -\gamma\|\log(\theta)\|_1 = \gamma\mathbf{1}^\top\log(\theta)$$

This results in the following maximization problem:

$$\max_{p\in\mathcal{P}} (f+\gamma\mathbf{1})^\top \log\theta$$

Choosing different γ for each uncertainty set with respect to each class, ie $\gamma_1 = \gamma|C_1|$ and $\gamma_2 = \gamma|C_2|$, gives the final form of the optimization problem presented in Lemma 3.1.

C.7 Dirichlet Prior and Laplace Smoothing

Lemma C.1. *MAP estimation of θ for a multinomial distribution under a Dirichlet prior parameter $\alpha = (\lambda+1)\mathbf{1}$ is equivalent to Laplace smoothing with parameter λ.*

Proof. This proof relies on the Dirichlet distribution being a conjugate prior to the multinomial distribution. Hence given a multinomial and Dirichlet distribution parameterized by $\theta \in \mathbb{R}^d$ and $\alpha \in \mathbb{R}^d$ respectively, we can say the following about the posterior distribution of θ given an observation of the multinomial distribution x:

$$\begin{aligned} P(\theta|x) &\propto P(x|\theta)P(\theta) \\ &\propto \prod_{j=1}^{d}\theta_j^{x_j}\prod_{j=1}^{d}\theta_j^{\alpha_j-1} \\ &= \prod_{j=1}^{d}\theta_j^{\alpha_j-1+x_j} \\ &\propto P(x+\alpha-\mathbf{1}|\theta) \end{aligned}$$

Hence it can be seen that choosing $\alpha = (\lambda+1)\mathbf{1}$ will lead to Laplace smoothing with smoothing parameter λ. □

D Acknowledgements

I would like to thank my team of collaborators for this project: Anand Siththaranjan, Armin Askari, and Laurent El Ghaoui. Each member provided invaluable contributions to this project: (1) AS built

scripts for evaluating each of the classical models and helped write up the paper, (2) AA provided high-level ideas throughout the project and contributed to writing. Finally, LEG was the primary adviser for this project.