Uncertainty Quantification for Robust Classification

Abraar Chaudhry

Advisor: Paul Dupuis
Readers: Rick Kenyon and Eli Upfal
Brown University

May 1, 2019

1 Introduction

We examine the problem of classification. Given some observation we attempt to place it into one of several categories. This decision is based on a model derived from observed data. For example, given a picture we may try to classify whether it shows a cat or a dog. The data is in the form of pairs of observations and categories. Thus in the same example, we may be given thousands of pictures of cats and dogs. The pairs that we are given are known as training data. The components of an observation are called features and the categories are known as labels. We attempt two methods regarding classification models, analyzing their sensitivities and robust optimization.

2 Model Specification

Y is the set of labels; X is the set of possible observations. For example, Y may be 'cat' or 'dog' and X may be the set of pictures of a certain size. We say Q is the true model, representing the true joint distribution over $X \times Y$. We wish to define a decision rule $D : X \to Y$, this represents what our prediction of label is given an observation. We may also construct an approximation of Q, called a design model, denoted P, also a distribution over $X \times Y$. A design model is used since there may not be any assumed structure to the true model and a simpler parametrized model may be more tractable for analysis.

2.1 Example

A common task is the classification of hand-drawn digits. A well-known example is the MNIST dataset which consists of thousands of 28x28 pixel images of digits. In this task, Y is the set of digits $\{1...9\}$ and X is a subset of $\mathbb{R}^{28\times 28}$ that represents the intensity of each of the pixels. Q encodes the relationship between X and Y. We may imagine first picking a digit randomly, then drawing it in a random style and then digitizing it subject to noise; this entire process is modeled with Q.

2.2 Bayes Decision Rule

If we know the conditional distribution of Y given X, then the following decision rule, called the Bayes decision rule is optimal for maximizing accuracy (which we define later):

$$D(x) = \arg\max_{y \in Y} P(y|x)$$

Given this formula we can transform the problem of generating an optimal decision rule to a problem of generating a good model of the distributions, in particular the conditional distribution.

3 Sensitivity Analysis

Given a parametrized distribution P_θ and a function of interest f, and a direction v, we define sensitivity like so:

$$S_{f,v}(P_\theta) = \lim_{\epsilon \to 0} \frac{1}{\epsilon}(E_{P_{\theta+\epsilon v}}[f] - E_{P_\theta}[f])$$

Sensitivity analysis is useful since it can tell us which parameters of our model are very important and which are not as important. Sensitivity measured in the direction of important parameters will be high. In this assessment we can apply the following result [1]:

$$|S_{f,v}(P_\theta)| \leq \sqrt{\mathrm{Var}_{P_\theta}(f)}\sqrt{v^T I(P_\theta) v}$$

The I above represents the Fisher information matrix, which is defined like so:

$$[I(P_\theta)]_{i,j} = -E_{P_\theta}\left[\frac{\partial^2}{\partial\theta_i \partial\theta_j} \log P_\theta\right]$$

From this we can see that the "interesting" part of sensitivity analysis depends on the Fisher matrix, which depends on how complicated the model is. Unfortunately, most generative models used in classification tend to be very simple e.g. Naive Bayes. These sorts of model tend to have Fisher matrices that are diagonal or near diagonal which leads to vacuous conclusions from the analysis.

4 Robust Optimization

In order to optimize we must have some measure of performance or loss to optimize. One common measure of performance of a model is accuracy. This means how often the decision rule is correct, and can be defined like so:

$$\text{Accuracy}(D) = \int 1_{D(x)=y} dQ(x,y) = Q(D(x) = y)$$

To have robustness we seek to have some guaranteed performance even if our training data did not come from Q, but rather a distribution close to Q. Thus we assume that our training data comes from P such that $R(Q||P) \leq r$. Here $R(Q||P)$ is the relative entropy or KL-divergence which is defined as follows:

$$R(Q||P) = \begin{cases} \int \log(\frac{dQ}{dP}) dQ & Q \ll P \\ \infty & \text{else} \end{cases}$$

KL-divergence is certainly not the only measure of similarity between distributions, however it is useful in this context because there are powerful results regarding this type of robustness. We will use the following result, which holds for any bounded, measurable f and unbounded f under conditions [3]:

$$E_Q[f] \leq R(Q||P) + \log E_P[e^f]$$

This way we can bound our performance on the true distribution by optimizing on our design model. One detail to note is that this method requires us to predetermine a value for r. This approach is not guaranteed to produce a different result than a more naive approach since it is possible that optimizing the above is equivalent to simply optimizing over the design model. To put this more formally, if we are considering a set of models and a loss function, and we have information about P for instance, through data, then a straightforward approach would be to choose the model with lowest loss when evaluated on our data, or on P. However since we have the additional assumption that $R(Q||P) \leq r$, it may make more sense to consider performance not only on P but also on distributions Q' such that $R(Q'||P) \leq R$, since any such Q' could be the true distribution Q as far as we know. Since Q is in this set of distributions, we can bound the loss on Q by the maximum loss over all such Q'. It may be the case that the model that minimizes the maximum loss over all distributions in the neighborhood of P is the same model that simply minimizes loss on P. This turns out to be the case when we apply this approach to accuracy.

4.1 Accuracy

Conventionally, loss functions are minimized, therefore we will attempt to minimize inaccuracy which is equivalent to maximizing accuracy. First we fix a design model P, then we consider a range of decision rules, $D_\alpha, \alpha \in A$. We now define F_α as the set on which D_α is inaccurate:

$$F_\alpha = \{(x, y) \in X \times Y | D_\alpha(x) \neq y\}$$

Now we formulate the optimization problem as follows:

$$\min_{\alpha \in A} \max_{R(Q'||P) \leq R} Q'(F_\alpha)$$

Since $Q'(F_\alpha) = E_{Q'}[1_{F_\alpha}]$ we can use a result [2] which is derived as a variation of the previously cited result, to rewrite the above as follows:

$$\min_{\alpha \in A} \min_{c > 0} \frac{1}{c}(R + \log E_P[e^{c 1_{F_\alpha}}])$$

Some manipulation reveals that the α that minimizes the above, is the same α that minimizes the following:

$$\min_{\alpha \in A} P(F_\alpha)$$

This result shows that in this case, it is futile to try to seek a nuanced robust strategy, since the robust optimization is equivalent to a naive non-robust optimization.

4.2 Log Loss

Since optimizing accuracy does not provide very useful results, we can consider optimizing alternate loss functions. The following pointwise loss function is

defined for a distribution P, over a single observation and its corresponding label. Assume that $Y = \{0, 1\}$:

$$\text{Log Loss}_P(x, y) = \begin{cases} -\log(P(y = 1|x)) & y = 0 \\ -\log(P(y = 0|x)) & y = 1 \end{cases}$$

Thus the average loss on the true distribution is as follows:

$$\text{Average Log Loss}_Q(P) = \int \text{Log Loss}_P(x, y) dQ(x, y)$$

An analogous definition can be defined for functions that given an observation x output a number in $[0, 1]$ with numbers near 0 representing confidence that the label of x is 0 and similarly with numbers near 1. These sorts of models include logistic regression and neural nets. This approach can also be expanded to multilabel classification, i.e. if $|L| > 2$.

4.3 Exact Loss Minimization and Robustness

In this section we demonstrate how exact loss minimization can work for a concrete example. We say *exact* to differentiate this from data-driven models we will examine later on. We define a parametrized model and show how the parametrization interacts with relative entropy. We then show what loss minimization looks like, contrasting robust and standard versions.

4.3.1 Model Definition

Consider the following setup and model:

$$X = [-1, 1]^2, Y = \{0, 1\}, \theta \in \mathbb{R}^2$$

$$P_\theta(x, y) = P(x) \cdot P_\theta(y|x)$$

P places uniform weight on X

i.e. $P \propto \lambda$ where λ is the Lebesgue measure

$$\sigma(t) = \frac{1}{(1 + e^{-t})}$$

$$P_\theta(1|x) = \sigma(\theta \cdot x)$$

$$P_\theta(0|x) = 1 - \sigma(\theta \cdot x) = \sigma(-\theta \cdot x)$$

4.3.2 Relative Entropy Neighborhood

With a parametrized model, we can now examine the relationship between distance in the parameter space and the relative entropy. For example, we can write down the KL divergence between $P_{(1,0)}$ and P_θ.

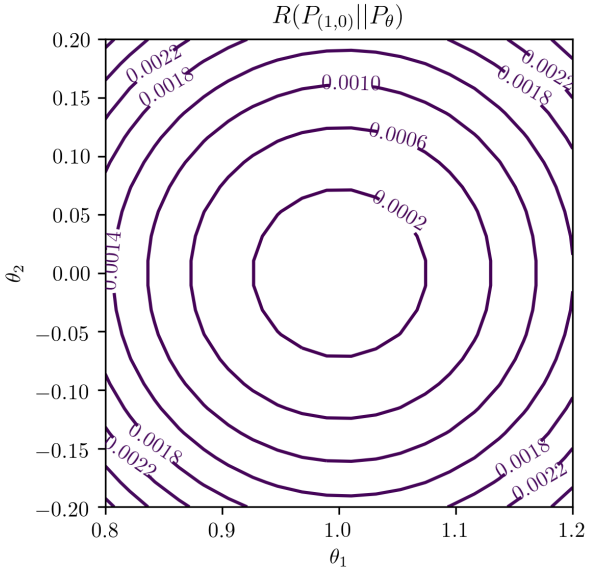
$$R(P_{(1,0)}||P_\theta) = \int_{X\times Y} \log\left(\frac{P_{(1,0)}(x,y)}{P_\theta(x,y)}\right) dP_{(1,0)}(x,y)$$

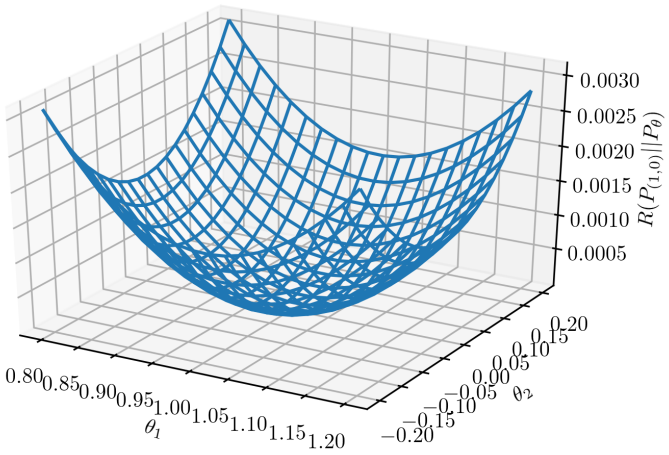
$$(y=1) \sim \frac{1}{4}\int_{-1}^{1}\int_{-1}^{1} \log\left(\frac{\sigma(x_1)}{\sigma(\theta\cdot(x_1,x_2))}\right)\sigma(\theta\cdot(x_1,x_2))dx_1dx_2 +$$

$$(y=0) \sim \frac{1}{4}\int_{-1}^{1}\int_{-1}^{1} \log\left(\frac{\sigma(-x_1)}{\sigma(-\theta\cdot(x_1,x_2))}\right)\sigma(-\theta\cdot(x_1,x_2))dx_1dx_2$$

$$= \frac{1}{2}\int_{-1}^{1}\int_{-1}^{1} \log\left(\frac{\sigma(x_1)}{\sigma(\theta\cdot(x_1,x_2))}\right)\sigma(\theta\cdot(x_1,x_2))dx_1dx_2$$

The expression above is well-defined and can be evaluated numerically using quadrature. The figures below shows the relative entropy $R(P_{(1,0)}||P_\theta)$ for different values of $\theta = (\theta_1, \theta_2)$





The figures show that KL divergence is convex and it achieves its minimum at $\theta = (1, 0)$ as we would expect. From the figures we can also see that $R(P_{(1,0)}||P_\theta) < .001$ roughly when $d((1, 0), \theta) < .15$, where d is Euclidean distance.

4.3.3 Classification Models

We will examine two classification models: a standard model and a robust model. We will use average log loss as the loss function for the models and we will use a design model of the form P_θ as defined above. The standard model will minimize loss relative to the design model, the robust model will attempt to minimize the maximum loss relative to any model within a relative entropy neighborhood of the design model. We abbreviate $l_\theta(x, y) = \text{Log Loss}_{P_\theta}(x, y)$. We denote the parameter of the standard model θ_s and define it as below:

$$\theta_s = \arg\min_{\theta' \in \mathbb{R}^2} \text{Average Log Loss}_{P_\theta}(P_{\theta'})$$

$$= \arg\min_{\theta' \in \mathbb{R}^2} \int_{X \times Y} l_{\theta'}(x, y) dP_\theta(x, y)$$

It turns out that this minimization is equivalent to minimizing the relative entropy between P_{θ_s} and P_θ. It follows that $\theta_s = \theta$. The robust model uses the results of relative entropy. We look at the distributions in the relative entropy neighborhood of P_θ.

$$\text{We would like to minimize this: } \max_{R(Q'||P_\theta) \leq R} E_{Q'}[l_{\theta'}]$$

Using previously mentioned results we modify the optimization problem and write the parameter of the robust model θ_r and define it as below:

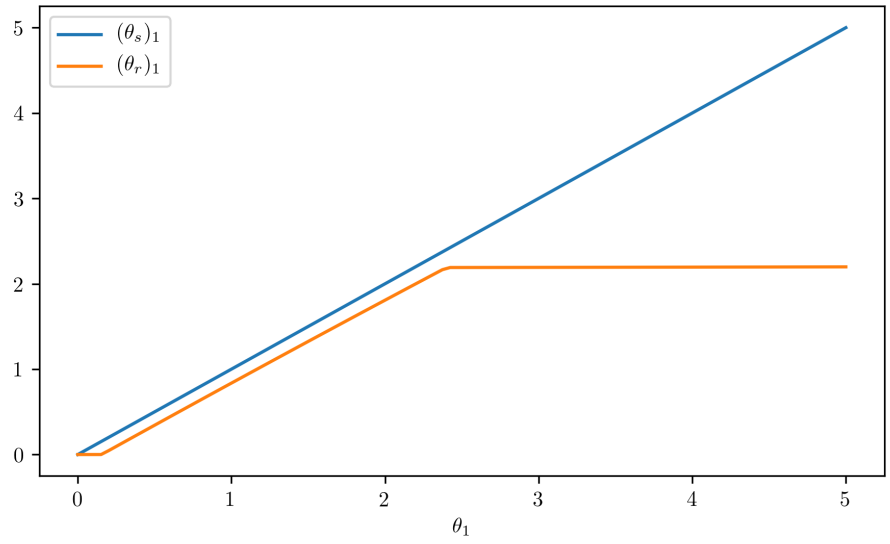
$$\theta_r = \arg\min_{\theta' \in \mathbb{R}^2} \min_{c>0} \frac{1}{c}(R + \log E_{P_\theta}[e^{cl_{\theta'}}])$$

$$= \arg\min_{\theta' \in \mathbb{R}^2} \min_{c>0} \frac{1}{c}\left(R + \log \int_{X \times Y} e^{cl_{\theta'}(x,y)} dP(x,y)\right)$$

4.3.4 Numerical Results

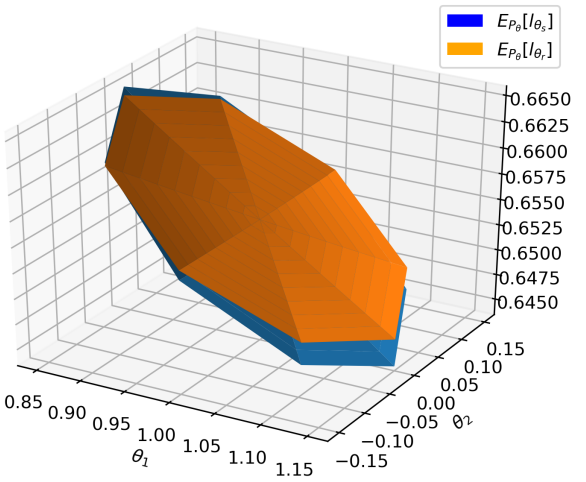
From the previous section, we know that $\theta_s = \theta$, but we also wish to know what θ_r looks like. We will show some value and performance of the model with numerical results. For the robust classifier's optimization we can evaluate the optimizations and integrals with numerical methods from [4]. We evaluate the integrals with Gauss-Legendre quadrature. We evaluate the inner minimization (over a positive c) using Brent's method constrained to positive values. We evaluate the outer minimization using the BFGS algorithm. The following figure shows the first component of θ_s and θ_r when θ is of the form $(\theta_1, 0)$. The second component of both θ_s and θ_r is 0.

Comparison of θ_s and θ_r

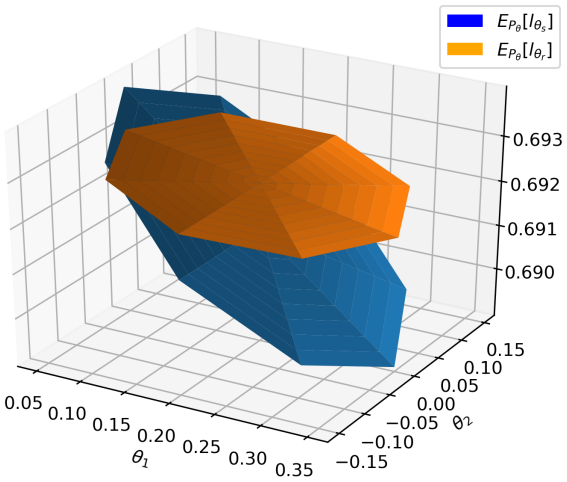


As we can see in the graph, $(\theta_s)_1$ increases linearly with θ_1. While $(\theta_r)_1$ is less than $(\theta_s)_1$, it does seem to increase in a parallel manner before leveling off around 2.4. We now compare losses of the two estimators in a small relative entropy neighborhood of P_θ for $\theta = (1, 0)$. From our earlier contour plot, we know that roughly if $d((1,0),\theta) < .15$, then $R(P_{(1,0)}||P_\theta) < .001$. Therefore, we set our R parameter to be .001. We optimize our models with respect to $\theta = (1, 0)$ and we evaluate them on a range of θ such that $d((1,0),\theta) < .15$.

This is the result:



The robust model has higher amounts of loss on most of the neighborhood, however the standard higher has a higher amount of maximum loss. This is somewhat difficult to see, but at the point $(.85, 0)$ the blue surface is slightly above the orange surface. Since the robust model has a lower maximum loss, we can say the robust minimization has been successful. We now take a different example which shows greater disparity between the robust and non-robust models. This time we optimize our models with respect to $\theta = (.2, 0)$. This is the result:



These results seem promising, since the robust model is succeeding in reducing the maximum loss in the neighborhood. It is worth noting that we are only plotting over distributions P that have the following two properties: $R(P_{(.2,0)}||P) < .001$ and $\exists\theta \in \mathbb{R}^2$ such that $P = P_\theta$. It is entirely possible that there are P in the neighborhood that are not expressible in the parametrization that we have given. Furthermore, it is then possible for the robust model to perform worse than the standard model for every distribution that is expressible in a certain parametrization, but still have a better worst case performance over all distributions in the neighborhood. Although we have only plotted over distributions in our parameter space, we are concerned with all distributions in the neighborhood. If we truly only cared about distributions in a parameter space, then we could formulate a different optimization problem and we would perhaps generate a different model.

4.4 Data-Driven Robustness

In order to create models from data, we must make approximations. Since we do not know the exact details of the distributions we can no longer compute expectations and we therefore employ a Monte Carlo method. We assume we are given n datapoints consisting of pairs of observations and labels $(x_i, y_i), i \in \{1 \dots n\}$ The following two approximations are used for the standard and robust models respectively:

$$E_P\left[l_{\theta'}(x,y)\right] \approx \frac{1}{n}\sum_{i=1}^{n} l_{\theta'}(x_i, y_i)$$

$$E_P\left[e^{cl_{\theta'}(x,y)}\right] \approx \frac{1}{n}\sum_{i=1}^{n} e^{cl_{\theta'}(x_i,y_i)}$$

We can use these approximations to write down formulae for our estimated standard and robust model which we denote $\hat{\theta_s}$ and $\hat{\theta_r}$, respectively:

$$\hat{\theta_s} = \arg\min_{\theta' \in \mathbb{R}^2} \sum_{i=1}^{n} l_{\theta'}(x_i, y_i)$$

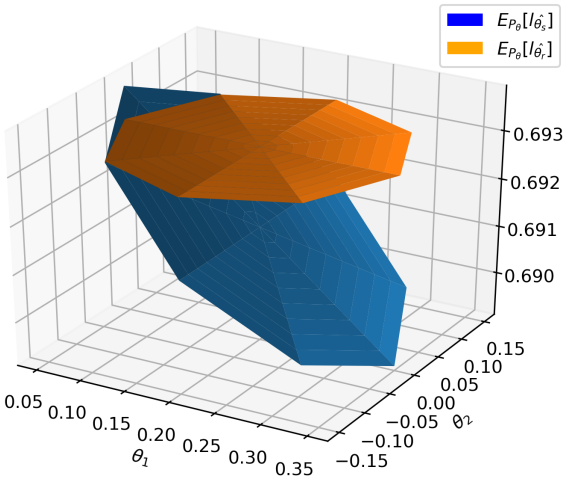
$$\hat{\theta_r} = \arg\min_{\theta' \in \mathbb{R}^2} \min_{c>0} \frac{1}{c}\left(R + \log\left(\sum_{i=1}^{n} e^{cl_{\theta'}(x_i,y_i)}\right)\right)$$

Using the above approximations as appropriately, we use the same optimizers as in the exact case, Brent's method and the BFGS algorithm, to calculate $\hat{\theta_s}$ and $\hat{\theta_r}$. We evaluate this method by again testing $\theta = (.2, 0)$. We generate 10,000 samples from $P_{(.2,0)}$ and train the models to generate $\hat{\theta_s}$ and $\hat{\theta_r}$. These are the values that were generated:

$$\hat{\theta_s} = [0.1870116, -0.0023545]$$

$$\hat{\theta_r} = [1.86870108e-04, -2.29822687e-06]$$

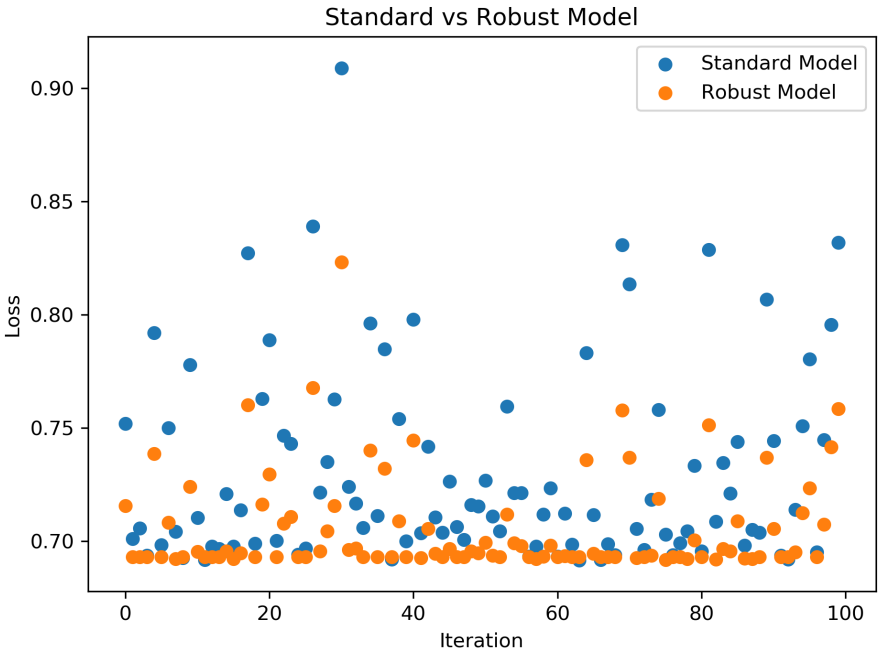
With these values we can now again make a chart to see the losses in the neighborhood of $(.2, 0)$. This is what the losses look like:



Here we again that the robust model has a lower maximum loss in the neighborhood.

4.4.1 Generalization

Since the robust model optimizes itself to perform well in a neighborhood of its input, we would expect it to generalize well. To test this we pick 30 samples at random from the 10,000 total and we train both the standard and robust models on this small data set. Then we evaluate both models on the rest of the samples and compare their losses. If the robustness built in to the robust model causes it to generalize better then we would expect it to perform better when given a small sample and evaluated on the rest. We take this procedure of picking 30 samples and repeat over 100 iterations. The results are shown in this graph:

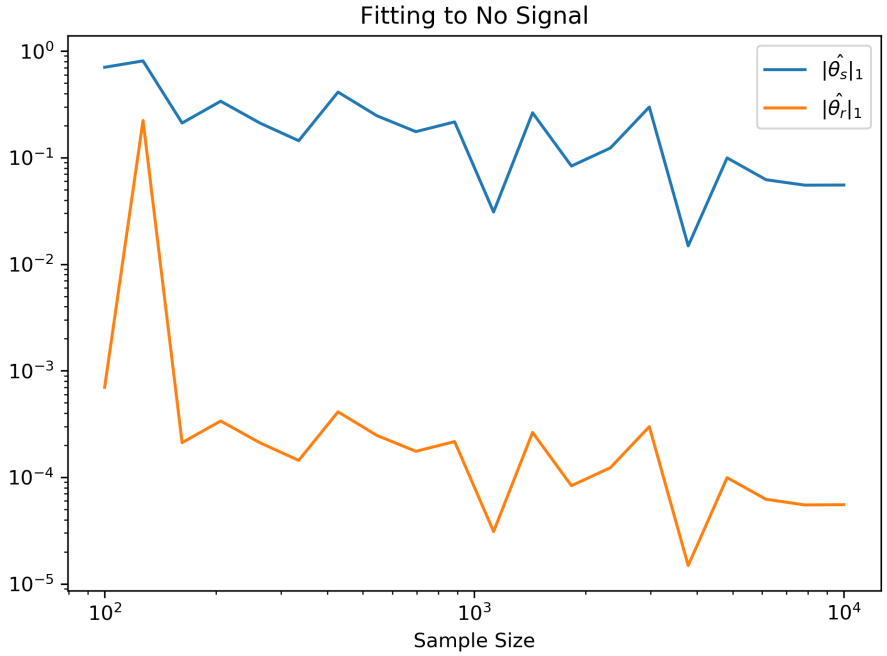


In this graph both models occasionally experience high levels of loss, and occasionally the standard model has less loss than the robust model. Overall however, it seems clear that the robust model is in fact performing much better than the standard model. The robust model performs better than the standard model in almost every iteration and the robust model's performance on average is also significantly better than that of the standard model.

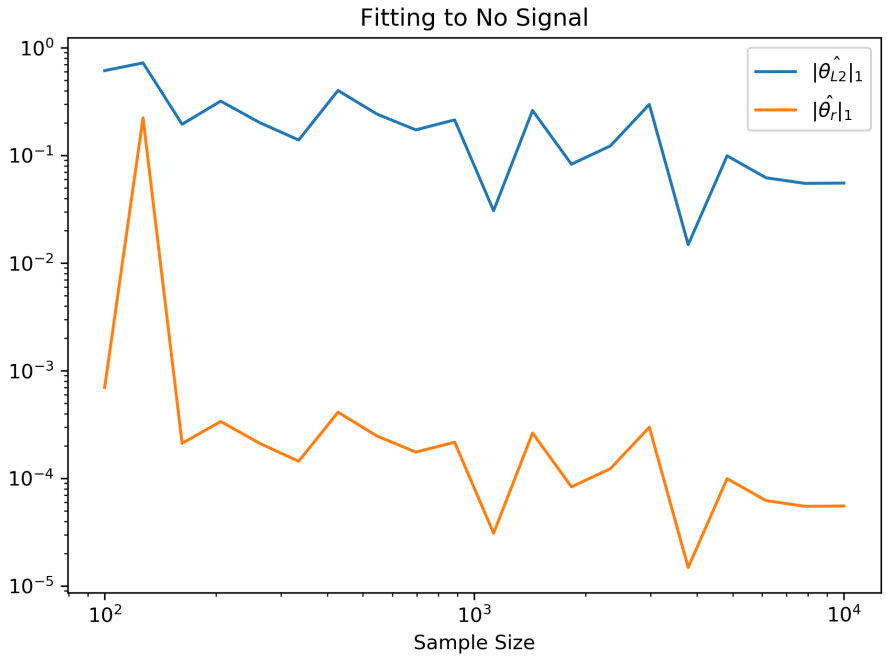
4.4.2 Complete Noise

Another interesting issue is what models do when there is no signal to fit to at all. For this idea we use $P_{(0,0)}$ which corresponds to y being distributed independent of x as a simple Bernoulli random variable with $p = \frac{1}{2}$. In this situation we would want $|\hat{\theta_s}|_1$ and $|\hat{\theta_r}|_1$ to be small. $\hat{\theta_s}$ in particular should go to 0 with

enough data since when computed exactly $\theta_s = (0, 0)$. This is simple since as we have stated before, for a distribution P_θ, $\theta_s = \theta$. We would also expect $\hat{\theta_r}$ to go to 0 since the neighborhood will be centered around $P_{(0,0)}$. It turns out that for large sample sizes the size of $|\hat{\theta_s}|_1$ tends to be significantly larger than $|\hat{\theta_r}|_1$. The following graph shows this:



One possible explanation for this effect is that the difference between the standard model and the robust model manifests simply as a form of regularization. However even when we compare the robust model to a standard implementation of logistic regression with an $L2$ penalty, this effect persists. This is seen in the following graph:



It turns out in this case that the $L2$ penalty makes almost no difference when compared with the standard model. The fact that $|\hat{\theta_r}|_1$ is much smaller is promising. Given that both $|\hat{\theta_s}|_1$ and $|\hat{\theta_r}|_1$ should theoretically go to 0 may suggest that in this instance, the robust model is somehow using the data it is given more efficiently and therefore converges to 0 much faster than a less efficient algorithm.

4.4.3 Computation Concerns

It is worth noting that the robust model takes a larger amount of computation to be trained, and therefore also takes longer to be trained. To train with 10,000 samples, the standard model took 46.9 ms, however the robust model took 812 ms. That is to say the robust model took approximately 17 times as long to train versus the standard model. This is not surprising as the robust model has to solve nested minimization problems where the standard has only a single minimization problem. It is also worth noting that the optimizations and training procedure for the robust model only used 'out-of-the-box' procedures and algorithms that could easily be used in practice. We have not put significant effort into addressing the computational cost of the robust model. It is possible that one could formulate a new algorithm specific to this task that could perform better and therefore make the computational cost of the robust model competitive with that of the standard model.

5 Conclusion

While we did not achieve results using sensitivity analysis, robust optimization seems like a promising approach to problems in classification. It proved effective in the formulation to minimize maximum loss in a relative entropy neighborhood. It also showed promise to help models generalize from small data.

References

[1] P. Dupuis, M. Katsoulakis, Y. Pantazis, and P. Plech. Path-space information bounds for uncertainty quantification and sensitivity analysis of stochastic dynamics. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):80–111, 2016.

[2] Paul Dupuis. Methods for model approximation and optimization in the presence of model uncertainty using information divergences, Oct 2017.

[3] Paul Dupuis, Matthew R. James, and Ian Petersen. Robust properties of risk-sensitive control. *Mathematics of Control, Signals and Systems*, 13(4):318–332, Dec 2000.

[4] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2019-05-01].