Machine Learning 6.867 - Final Report
Robust Artificial Neural Networks

Ying (Daisy) Zhuo

December 9, 2015

Abstract

Artificial neural network is a popular and successful machine learning method. However, when data are subject to uncertainty, the model can be unstable and overfit the training data. In this project, I explored the possibility of immunizing ANNs against uncertainties in data via robust optimization. I derived the robust formulation and established its connection with regularization and adversarial training. As a preliminary computational experiment (but not the focus of the paper), I implemented the method and observe that Robust-ANN offers slightly higher out-of-sample accuracy in the MNIST dataset.

1 Introduction

Artificial neural networks (ANN) have been shown to have very strong performance for many challenging machine learning tasks such as image recognition and natural language processing. This is largely due to its ability to approximate non-linear functions arbitrarily well and to extract patterns from complicated or noisy data. However, the generalization performance of neural networks can be severely impacted by overfitting the training set.

Regularization is classical approach to reduce overfitting in many machine learning methods. By adding a penalty on the weight parameters to the objective function, the weight parameters are shrunk toward zero. Smaller weights give smoother network response and can potentially reduce overfit. One simple regularizer is *weight decay*, which adds a $\lambda \mathbf{w}^T \mathbf{w}$ term to the error function. In a probabilistic view, simple weight decay regularization is equivalent to putting a Gaussian prior over the weight parameters.

Unfortunately, simple weight decay does not achieve the desirable property of *consistency*, also known as scaling invariance. The invariance property says if we linearly transform the input or target vector, we should obtain equivalent network with linear transformation of weights as given. However, the simple weight decay regularizer term does not maintain this property, as it penalizes on all weights equally. One scale-invariant regularizer is to use different trade-off parameters for the two weights; i.e., adding $\lambda_1 \|\mathbf{w}^{(1)}\|^2 + \lambda_2 \|\mathbf{w}^{(2)}\|^2$ to the negative log likelihood, biases excluded. In the Bayesian framework, putting different priors on $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$ corresponds to an improper prior, which cannot be normalized. Using improper priors can lead to difficulty in selecting regularization coefficients and Bayesian model comparisons [2].

Given the somewhat ad hoc nature of adding regularizers and difficulty in the Bayesian framework on neural networks, I would like to consider the issue from a different perspective. In a recent paper, Bertsimas et al. proposed a principled approach using robust optimization to model data uncertainty, and have shown its equivalence to regularization in certain circumstances [1]. For example, by modeling uncertainty in input features on hinge-loss minimization, it is equivalent to support vector machines. The computational results in the paper suggest that addressing uncertainties from robust perspective offers better generalization, as demonstrated in logistic regression, support vector machine, and decision trees. The results are encouraging and hint that there is potential improvement via robustification for more complex models such as ANN.

In this project, I first introduce robust optimization at a high level, then derive the formulation of Robust-ANN that can be readily implemented with low incremental computational overhead. I further establish the resemblance between Robust-ANN and regularization, and discuss some of its properties. Since the focus of this paper is more on the theoretical development, I offer limited preliminary computational evidence by testing on the MNIST and compare the performance of between Robust and non-robust ANNs.

2 Overview of Robust Optimization

This section is an overview of robust optimization that introduces the notions of uncertainty sets and robust counterparts that will be used later when applying robust optimization techniques to ANN. Some of the content is adapted from [1].

Robust optimization is arguably one of the fastest growing areas in optimization in the past decade. As a means for modeling uncertainty in optimization problems without the use of probability distributions, it is demonstrated to be both computationally tractable and offer attractive solutions.

Under this modeling framework, we construct deterministic *uncertainty sets* that contain possible values of uncertain parameters. We then seek a solution that is optimal for all such realizations of this uncertainty. Consider the general optimization problem:

$$\begin{aligned} \max_{\mathbf{x}\in\mathcal{X}} \quad & c(\mathbf{u},\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{u},\mathbf{x}) \le 0. \end{aligned}$$

where $\mathbf{x}$ is the vector of decision variables, and $\mathbf{u}$ is a vector of given parameters. Relaxing the assumption that $\mathbf{u}$ is fixed, we assume instead that the realized values of $\mathbf{u}$ are restricted to be within some uncertainty set $\mathcal{U}$. We form the corresponding robust optimization problem by optimizing against the worst-case realization of the uncertain parameters across the entire uncertainty set:

$$\begin{aligned} \max_{\mathbf{x}\in\mathcal{X}} \quad & \min_{\mathbf{u}\in\mathcal{U}} \; c(\mathbf{u},\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{u},\mathbf{x}) \le 0 \quad \forall \mathbf{u} \in \mathcal{U}. \end{aligned}$$

Despite typically having an infinite number of constraints, it is often possible to reformulate the problem as a deterministic optimization problem with finite size, depending on the choice of uncertainty set $\mathcal{U}$. The resulting deterministic problem is deemed the *robust counterpart*, which is generally a problem of the same complexity as the nominal problem.

There is extensive evidence in the literature that robust solutions have significant advantages relative to nominal solutions. In essence, we are willing to accept robust solutions that are suboptimal in the nominal case under the data at hand, but are much more often feasible/optimal when data are perturbed. In the case of machine learning problem, such trade-off is especially important to avoid overfitting the training data.

3 Theoretical Derivation

The derivation in this section is conducted independently of [4]. Let $E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w})$ be the negative log likelihood of single training data point $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ in an ANN. Thus, the optimization problem to be solved is

$$\min_{\mathbf{w}} \sum_{i=1}^{n} E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}) \tag{1}$$

Under a given uncetainty set $\mathcal{U}$ on data $\mathbf{x}$, the robust formulation of ANN is then:

$$\min_{\mathbf{w}} \max_{\Delta\mathbf{x}\in\mathcal{U}} \sum_{i=1}^{n} E(\mathbf{x}^{(i)} + \Delta\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}) \tag{2}$$

The exact robust counterpart for Problem (2) cannot be readily obtained, due to the complex non-linear and non-convex structure of the objective function. However, with some assumptions and a little machinery, we can approximate the original problem while keeping the problem at the same level of computational complexity as the nominal one. To simplify the problem a bit, let's assume $\mathcal{U}$ decouples across the data points. For example, a common uncertainty set can be constructed as $\|\mathbf{x}^{(i)}\|_p \leq \rho$ for each i. This way, the inner minimization is separable among the i's; i.e., we need to solve for

$$\max_{\Delta\mathbf{x}^{(i)}} E(\mathbf{x}^{(i)} + \Delta\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}) \tag{3}$$

for each i. As demonstrated in [1], the particular choice of uncertainty set does not affect the improvement drastically for many classifiers, therefore this choice is warranted.

To solve Problem (3), we consider a Taylor first-order expansion:

$$E(\mathbf{x}^{(i)} + \Delta\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}) = E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}) + \nabla_{\mathbf{x}} E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w})^T \Delta\mathbf{x}^{(i)} \tag{4}$$

When $\mathcal{U} = \{\|\mathbf{x}^{(i)}\|_\infty \leq \rho\}$ (i.e., $\max_r |\mathbf{x}_r^{(i)}| \leq \rho$), the minimizer $\Delta\mathbf{x}^{(i)}$ should take on value of $\rho \cdot sign(\nabla_{\mathbf{x}} E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}))$ component-wise. Substituting to the objective gives

$$\max_{\Delta\mathbf{x}^{(i)}} E(\mathbf{x}^{(i)} + \Delta\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}) = E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}) + \rho\|\nabla_{\mathbf{x}} E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w})\|_1$$

Similarly, when $\mathcal{U} = \{\|\Delta\mathbf{x}^{(i)}\|_1 \leq \rho\}$ (i.e., $\sum_{i=1}^n |x_r^{(i)}| \leq \rho$), at optimality $\Delta\mathbf{x}_r^{(i)}$ takes on ρ at the component with the largest value of $\nabla_{\mathbf{x}} E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w})$, and zero at all other components. Substituting to the objective gives

$$\max_{\Delta\mathbf{x}^{(i)}} E(\mathbf{x}^{(i)} + \Delta\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}) = E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}) + \rho\|\nabla_{\mathbf{x}} E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w})\|_\infty$$

To summarize, Problem (2) can be reformulated as follows:

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \left[E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}) + \rho\|\nabla_{\mathbf{x}} E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w})\|_{p^*} \right], \tag{5}$$

where $\|\cdot\|_p^*$ is the dual norm of $\|\cdot\|_p$ (i.e., $1/p + 1/p^* = 1$).

The approximate robust counterpart problem in (5) can be solved efficiently using numeric optimization methods, if the gradient of the second term can be expressed explicitly. Compared to a regularized ANN (two-layer network for example):

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \left[E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w}) \right] + \lambda_1 \|\mathbf{w}^{(1)}\|^2 + \lambda_2 \|\mathbf{w}^{(2)}\|^2,$$

Robust-ANN adds to the objective a single penalization term that depends on the gradient (which in turn depends on the weights at all levels). Formulation (5) has advantage as it has a stronger theoretical basis rooted in robustness, contains fewer parameters to cross-validate, and avoids the issue of improper prior.

In practice, one can solve the robust formulation in (5) without writing the gradient explicitly. An iterative method is proposed as follows: first fix initial $\mathbf{w}$ and find the most adversarial $\Delta\mathbf{x}_i$ for each i based on $\nabla_{\mathbf{x}} E(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{w})$. Then one can update the $\mathbf{x}_i$ and run a basic ANN forward-backward procedure to obtain $\mathbf{w}$. The algorithm terimnates when $\mathbf{w}$ converges. A more efficient method, proposed in [4], takes a single descent step to update $\mathbf{w}$ under the perturbed data rather than solving each substep to optimality. Under this, the training time is about twice as long as basic ANN.

4 Implementation and Computational Results

I implemented Robust-ANN in the programming language Julia with package MXNet, an efficient and flexible deep learning framework [3]. I chose to use this package rather than writing my own from scratch in order to take advantage of the computational efficiency and to obtain a decent baseline to compare with. To benchmark, the implementation is tested on the MNIST dataset of handwritten digits, with 50,000 data points in training, 10,000 in evaluation, each of dimension 784 (28×28 pixels). The outputs are digits 0 through 9. The network structure has three layers and is presented in Figure 1. Other model specifications include: number of epoch is 3 to ensure baseline method obtains decent accuracy, learning rate is 0.1, momentum is 0.9, weight decay coefficient is 10^{-5}. We use a stochastic gradient descent optimizer, with batch size being 100. Since the problem size is small, we only use CPU for this purpose. To ensure the randomness in initialization does not affect the comparison, we set the same seed in both cases.

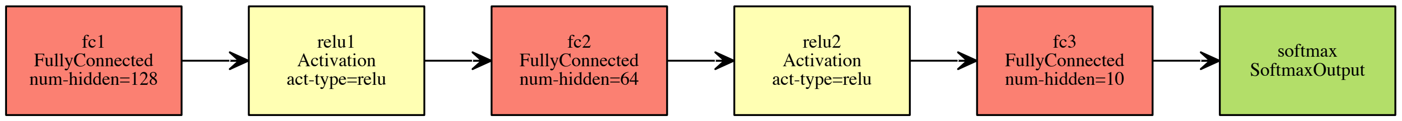

Figure 1: Illustration of ANN architecture in computational experiments.

The procedure of Robust-ANN is as follows: after an iteration of basic ANN, a back-propagation is called on each small batch to obtain the gradients, which in turn is used to calculate the adversarial perturbation $\Delta\mathbf{x}_i$. A small complication is that MXNet does not provide gradient with respect to the data $\mathbf{x}^{(i)}$ directly; to obtain such, we take the current first level weights $\mathbf{w}$, the gradient with respect to the ReLU activation $\frac{\partial J}{\partial \mathbf{a}}$ and do the following calculation:

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{x}} &= \sum_{j=1}^{n_1} \frac{\partial J}{\partial a_j} h'(\mathbf{w}_j^T \mathbf{x}) \mathbf{w}_j \\ &= \sum_{j=1}^{n_1} \frac{\partial J}{\partial a_j} \mathbb{1}_{\{\mathbf{w}_j^T \mathbf{x} > 0\}} \mathbf{w}_j \end{aligned} \qquad (6)$$

The gradient of the activation function (ReLU here) can easily be adapt to that of logistic function or tanh. Depending on the choice of uncertainty set, we can construct different adversarial $\Delta\mathbf{x}^{(i)}$. In this example, we consider the l_∞ norm uncertainty set and therefore obtain $\mathbf{x}_i^{\text{new}} = \mathbf{x}_i^{\text{old}} + \rho * sign(\frac{\partial J}{\partial \mathbf{x}})$. We can now fit the a new basic ANN with the updated $\mathbf{x}_i^{\text{new}}$ until convergence. The convergence is measured as the negative log likelihood being no farther than 1 between iterations.

We obtain the following preliminary results: the base case ANN achieves an out-of-sample accuracy of 96.94%. Robust-ANN converges after 15 iterations and achieves an out-of-sample accuracy of 97.27%. To compare, robustness offers a slight improvement of 0.33%.

5 Conclusions

In this project, I explore the possibility of adding robustness against uncertainties in data. I implemented Robust-ANN and observe that it offers slightly higher out-of-sample accuracy in the MNIST dataset. Whether this improvement is significant needs further testing on other datasets and various other specifications of the model. Regardless, the paper presents a formulation of ANN that offers robustness to uncertainties in principle, and may demonstrate better out-of-sample performance.

References

[1] Dimitris Bertsimas, Jack Dunn, Colin Pawlowski, and Daisy Zhuo. Robust classification. *Submitted for publication*, 2015.

[2] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[3] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. Neural Information Processing Systems, Workshop on Machine Learning Systems, 2015.

[4] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*, 2015.