Perturbation, Optimization and Statistics

Editors:

Tamir Hazan tamir.hazan@technion.ac.il
Technion - Israel Institute of Technology
Technion City, Haifa 32000, Israel

George Papandreou gpapan@google.com
Google Inc.
340 Main St., Los Angeles, CA 90291 USA

Daniel Tarlow dtarlow@microsoft.com
Microsoft Research
Cambridge, CB1 2FB, United Kingdom

This is a draft version of the author chapter.

The MIT Press
Cambridge, Massachusetts
London, England

uniform distribution over output classes when presented with such an input. Similarly, a model that reports an independent probability estimate for the detection of each class should preferably indicate that no classes are present. However, both formulations are easily fooled into reporting that a specific class is present with high probability simply by using Gaussian noise as input to the model (Goodfellow et al., 2014b). Nguyen et al. (2014) demonstrated that large, state of the art convolutional networks can also be fooled using rich, structured images generated by genetic algorithms. Because rubbish class examples do not correspond to small perturbations of a realistic input example, they are beyond the scope of this chapter.

1.2.7 Defenses

To date, the most effective strategy for defending against adversarial examples is to explicitly train the model on them, as described in Section 1.3.

Many traditional regularization strategies such as weight decay, ensemble methods, and so on, are not viable defenses. Regularization strategies can fail in two different ways. Some of them reduce the error rate of the model on the test set, but do not reduce the error rate of the model on adversarial examples. Others reduce the sensitivity of the model to adversarial perturbation, but only have a significant effect if they are applied so powerfully (e.g., with such a large weight decay coefficient) that they cause the performance of the model to seriously degrade on the validation set. The failure of some traditional regularization strategies to provide a defense against adversarial examples is discussed by Szegedy et al. (2014b) and the failure of many more traditional regularization strategies is discussed by Goodfellow et al. (2014b). In summary, most traditional neural network regularization techniques have been tested and do not provide a viable defense.

In addition to these traditional methods, some new methods have been devised to defend against adversarial examples. However, none of these methods are yet very effective. For even the best methods, the error rate on adversarial examples remains noticeably higher than on unperturbed examples.

Gu and Rigazio (2014) trained a denoising autoencoder, where the noise corruption process was the adversarial example generation process. In other words, the autoencoder is trained with adversarial example to predict the corresponding unperturbed example as output. The goal was to use the autoencoder as a preprocessing step before applying a classifier, in order to make the classifier resistant to adversarial examples. Unfortunately, the combination of the autoencoder and the classifier then becomes vulnerable to a different class of adversarial examples that the autoencoder has not been trained to resist. Gu and Rigazio (2014) reported that the combined system was vulnerable to adversarial examples with smaller perturbation size than the original classifier. The authors of this chapter speculate that this can be explained by the linearity hypothesis; the autoencoder is still built mostly out of linear components. If one views the classifier as being roughly a product of matrices, then the autoencoder simply introduces two more matrix factors into this product. If these matrices have any singular values that are larger than one, then they amplify adversarial perturbations in the corresponding directions.

Papernot et al. (2015) introduced an approach called *defensive distillation.* First, a teacher model is trained to maximize the likelihood of the training set labels:

$$\boldsymbol{\theta}^{(t)*} = \text{argmax} \sum_{i=1}^{m} \log p^{(t)} \left(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}^{(t)} \right).$$

The teacher model is then used to provide soft targets for a second network, called the student network. The student network is trained not just to predict the same class as the teacher network, but to predict the same probability distribution over classes:

$$\boldsymbol{\theta}^{(s)*} = \underset{\boldsymbol{\theta}^{(s)}}{\text{argmin}} \sum_{i=1}^{m} D_{\text{KL}} \left(p^{(t)} \left(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}^{(t)} \right) \| p^{(s)} \left(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}^{(s)} \right) \right).$$

This technique noticeably reduces the vulnerability of a model to adversarial examples but does not completely resolve the problem.

As an original contribution of this chapter, an experimental observation shows that a simpler method than defensive distillation also has a beneficial effect. Rather than training a teacher network to provide soft targets, it is possible to simply modify the targets from the training set to be soft, e.g., for a k class problem, replace a target value of 1 for the correct class with a target value of .9, and for the incorrect classes replace the target of 0 with a target of $\frac{1}{10k}$. This technique is called *label smoothing* and is a component of some state of the art object recognition systems (Szegedy et al., 2015). The label smoothing experiment was based on a near-replication of the MNIST classifier of Goodfellow et al. (2013a). This classifier is a feedforward network with two hidden layers consisting of maxout units, trained with dropout (Srivastava et al., 2014). The model was trained on only on the first 50,000 examples and was not re-trained on the validation set, so the test error rate was higher than in the original investigation of Goodfellow et al. (2013a). The model obtained an error rate of 1.28% on the MNIST test set. The error rate of the model on adversarial examples on the MNIST test set using the fast gradient sign method (Equation 1.3) with $\epsilon = .25$ was 99.97%. A second instantiation of exactly the same model was trained using label smoothing. The error on the test set dropped to 1.17%, and the error rate on the adversarially perturbed test set dropped to 33.0%. This error rate indicates a significant remaining vulnerability but it is a vast improvement over the pre-smoothing adversarial example error rate.

The linearity hypothesis can explain the effectiveness of label smoothing. Without label smoothing, a softmax classifier is trained to make infinitely confident predictions on the training set. This encourages the model to learn large weights and strong responses. When values are pushed outside the areas where training data concentrates, the model makes even more extreme predictions when extrapolating linearly. Label smoothing penalizes the model for making overly confident predictions on the training set, forcing it to learn either a more non-linear function or a linear function with smaller slope. Extrapolations by the label-smoothed model are consequently less extreme.

1.3 Adversarial Training

Adversarial training corresponds to the process of explicitly training a model to correctly label adversarial examples. In other words, given a training example $\boldsymbol{x}$ with label y, the training set may be augmented with an adversarial example $\tilde{\boldsymbol{x}}$ that is still associated with training label y. Szegedy et al. (2014b) proposed this method, but were unable to generate large amounts of adversarial examples due to reliance on the expensive L-BFGS method of crafting adversarial examples. Goodfellow et al. (2014b) introduced the fast gradient sign method and showed that it enabled practical adversarial training. In their approach, the model is trained on a minibatch consisting of both unmodified examples from the training set and adversarially perturbed versions of the same examples. Crucially, the adversarial perturbation is recomputed using the latest version of the model parameters every time a minibatch is presented. Adversarial training can be interpreted as a minimax game,

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x},y} \max_{\boldsymbol{\eta}} \left[J(\boldsymbol{x}, y, \boldsymbol{\theta}) + J(\boldsymbol{x} + \boldsymbol{\eta}, y) \right],$$

with the learning algorithm as the minimizing player and a fixed procedure (such as L-BFGS or the fast gradient sign method) as the maximizing player.

Goodfellow et al. (2014b) found that adversarial training on MNIST reduced both the test set error rate and the adversarially perturbed test