Probabilistic Robustness Estimates for Deep Neural Networks

Nicolas Couellan

Probabilistic Robustness Estimates for Deep Neural Networks

Nicolas Couellan [1 2]

Abstract

In the case of deep dense neural networks, under random noise attacks, we propose to study the probability that the output of the network deviates from its nominal value by a given threshold. We derive a simple concentration inequality for the propagation of the input uncertainty through the network using the Cramer-Chernoff method and estimates of the local variation of the neural network mapping computed at the training points. We further discuss and exploit the resulting condition on the network to regularize the loss function during training. Finally, we assess the proposed tail probability estimate empirically on three public regression datasets and show that the observed robustness is very well estimated by the proposed method.

1. Introduction

Deep neural networks have been found to be very sensitive to data uncertainties (Fawzi et al., 2017; Szegedy et al., 2014) to the point that a whole research community is now addressing the so-called network attacks. Attacks may be random, when data are corrupted by some random noise or may be adversarial, when the noise is specifically designed to alter the network output (Szegedy et al., 2014). In this article, we will focus on the random case. Most data are usually uncertain, either because the data are related to naturally noisy phenomenon and we only have access to some of its statistics or because assessing devices to do not have sufficient accuracy to record precisely the data.

Most people have addressed the problem of robustness to bounded input perturbations through the use of regularization techniques (Gouk et al., 2018; Oberman & Calder, 2018; Virmaux & Scaman, 2018; Finlay et al., 2018). The main idea is to consider the neural network as a Lipschitz map between the input and output data. The Lipschitz constant of the network is then estimated or upper bounded by the norm of the layer-by-layer weights product. This estimates the expansion or contraction capability of the network and is then used to regularize the loss during training. Often, there is a price to pay: the expressivity of the network may be reduced, especially if the weights are too constrained or constrained layer by layer instead of constrained accross layers (Couellan, 2019). Such strategies are enforcing robustness but do not provide guarantees or estimates on the level of robustness that has been achieved. In the case of adversarial perturbation, some authors have proposed methods for certifying robustness (Kolter & Wong, 2017; Boopathy et al., 2018). Recently, a probabilistic approach has also been proposed in the case of random noise for convolutional neural networks (Weng et al., 2019). The authors derive probabilistic bounds based on the idea that the output of the network can be lower and upper bounded by two linear functions. The work proposed here is along the same line but distinct in several aspects. It combines upper bounds on tail probabilities calculated by deriving a specific Cramer-Chernoff concentration inequality for the propagation of uncertainty through the network with a network sensitivity estimate based on a network gradient calculation with respect to the inputs. The network gradient is computed by automatic differentiation and estimates the local variation of the output with respect to the input of the network. The estimation is carried out and averaged over the complete training set. A maximum component-wise gradient variation is also calculated in order to give probabilistic certificates rather than estimates. The certificates can be used in place of estimates whenever guaranteed upper bounds are needed, however they are often not as accurate since they are based on variation bounds rather than averages. We then discuss the use of the derived bounds and estimates in the design of deep robust neural networks and conduct experiments in order to assess the robustness probabilitic estimates for various regularization strategies.

[1]ENAC, Université de Toulouse, Toulouse, France [2]Institut de Mathématiques de Toulouse UMR 5219, Université de Toulouse UPS IMT, Toulouse, France. Correspondence to: Nicolas Couellan <nicolas.couellan@recherche.enac.fr>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning.

2. Probabilistic certificates of robustness

Consider feed-forward fully connected neural networks that we represent as a successive composition of linear weighted combination of functions such that $x^l = f^l((W^l)^\top x^{l-1} + b^l)$ for $l = 1, \dots, L$, where $x^{l-1} \in \mathbb{R}^{n_{l-1}}$ is the input of the l-th layer, the function f^l is the L_f-Lipschitz continuous activation function at layer l, and $W^l \in \mathbb{R}^{n_{l-1} \times n_l}$ and $b^l \in \mathbb{R}^{n_l}$ are the weight matrix and bias vector between layer $l-1$ and l that define our model parameter $\theta = \{W^l, b^l\}_{l=1}^L$ that we want to estimate during training. The network can be seen as the mapping $g_\theta : x^0 \to g_\theta(x^0) = x^L$. The training phase of the network can be written as the minimization of the empirical loss $\mathcal{L}(x, y, \theta) = \frac{1}{n} \sum_{i=1}^n l_\theta(g_\theta(x_i), y_i)$ where l_θ is a measure of discrepancy between the network output and the desired output.

Assume now that we only have access to noisy observations x_i of the input sample. However, we know that these observations are drawn from a distribution D with finite support. We first consider the special case where the functions f_l are linear or piece-wise linear (this includes the case of ReLu activation functions) and then extend the analysis to more general functions.

We first consider the one layer simple case where the outputs of the network $y = x^L$ (with $L = 1$ in this case) depends linearly of the inputs $x = x^0 \in \mathbb{R}^n$ as follows: $w^\top x + b$ where $w = W^1$ is the single layer weights vector and $b = b^1$ is the layer bias.

We assume that our input observations are corrupted by some additive noise ϵ such that $\epsilon \sim D$ and $\forall i = 1, \dots, n$, we have $\epsilon_i \in [-\gamma, \gamma]$ with $\gamma < +\infty$. Our objective is to ensure the following property:

$$\mathbb{P}_{\epsilon \sim D} \left(\|y - y_\epsilon\| \leq \Gamma \right) \geq 1 - \alpha \tag{1}$$

where $y_\epsilon = w^\top (x + \epsilon) + b$, Γ is the allowed output uncertainty and $1 - \alpha$ is some predefined level of confidence.

In the following proposition, we give a condition on w that will ensure, with probability greater than $1 - \alpha$, that the output uncertainty remains below Γ.

Proposition 2.1 *If the network inputs are subject to an additive uncertainty ϵ where $\forall i = 1, \dots, n$, $\epsilon_i \sim D$ and $\mathrm{supp}(D) \in [-\gamma, \gamma]$ ($\gamma < +\infty$), then for a given $\alpha \in [0, 1]$ and a given output uncertainty level Γ, the following condition holds:*
If the layer weights vector w satisfies $\|ww^\top\|_F \leq \frac{\Gamma^2}{\gamma^2 \sqrt{2 \log(1/\alpha)}}$, then $\mathbb{P}_{\epsilon \sim D} \left(\|y - y_\epsilon\| \leq \Gamma \right) \geq 1 - \alpha$.

(the proof of Proposition 2.1 is provided in Appendix A).

We now address the case where the network is composed of several layers $l = 1, \dots, L$ with linear or piece-wise linear activation functions. Let $\mathcal{W} = \prod_{l=1}^L W^l$. Property (1) should now relate to the output of the last layer as follows:

$$\mathbb{P}_{\epsilon \sim D} \left(\|x^L - x_\epsilon^L\| \leq \Gamma \right) \geq 1 - \alpha \tag{2}$$

where x_ϵ^L is the output of the layer L when a noisy input $x_\epsilon^0 = x^0 + \epsilon$ is propagated through the network and ϵ is the additive input noise such that $\epsilon \sim D$ and $\forall i = 1, \dots, n$, we have $\epsilon_i \in [-\gamma, \gamma]$ ($\gamma < +\infty$). With this setting, we can now state and prove the following:

Proposition 2.2 *If the network inputs are subject to an additive uncertainty ϵ where $\forall i = 1, \dots, n$, $\epsilon_i \sim D$ and $\mathrm{supp}(D) \in [-\gamma, \gamma]$ ($\gamma < +\infty$), then for a given $\alpha \in [0, 1]$ and a given output uncertainty level Γ, the following condition holds:*
If the layer weights vector w satisfies $\left\|\mathcal{W}\mathcal{W}^\top\right\|_F \leq \frac{\Gamma^2}{\gamma^2 \sqrt{2 \log(1/\alpha)}}$ then $\mathbb{P}_{\epsilon \sim D} \left(\|x^L - x_\epsilon^L\| \leq \Gamma \right) \geq 1 - \alpha$.

(the proof of Proposition 2.2 is provided in Appendix A).

The bound derived above relies on the following inequality (see (8) in Appendix A for multilayer architectures):

$$\mathbb{P}_{\epsilon \sim D} \left(\|x^L - x_\epsilon^L\| \leq \Gamma \right) \geq \mathbb{P}_{\epsilon \sim D} \left[tr\left(\epsilon\epsilon^\top \mathcal{W}\mathcal{W}^\top \right) \leq \Gamma^2 \right] \tag{3}$$

The Chernoff bound proposed above may be tight with respect to the right hand side of the above inequality, however, with respect to the left hand side, (3) is not tight in general as the deep complex layer structure of the neural network generates a highly non linear behavior. The linear upper bound is often of poor quality. To address this issue, we propose an alternative to estimate the variation of the neural network response. Without loss of generality, in the following, we will consider that the network output is 1-dimensional. Since $\|\epsilon\|$ may be considered small with respect to the magnitude of the network inputs, we are interested in the local behavior of the network output that we will approximate as follows:

$$x_\epsilon^L - x^L = F(x + \epsilon) - F(x) \simeq \nabla_x F(x)^\top \epsilon \tag{4}$$

where $F(x) = f^L((W^L)^\top f^{L-1}(\dots f^0((W^0)^\top x + b^0)) + b^{L-2}) + b^{L-1})$ and $\nabla_x F$ is the gradient of F with respect to the network input x.

The linear approximation (4) is only local but its advantage is that it can easily be evaluated at many x values. Indeed, while visiting all input vectors x during training, this information is usually available at a very low extra computational cost through automatic differentiation in most training computer algorithms and packages for neural networks (such as `TensorFlow` (Abadi et al., 2015) and `Pytorch` (Paszke et al., 2017)). Therefore, from local estimates at various training points x_i^0 for $i = 1, \dots, n$, we calculate two n_0-dimensional vectors of network variation estimates $\widehat{\nabla_x F}$ such that $\left(\widehat{\nabla_x F}\right)_k =$

$\text{sign}\left(\left(\nabla_x F(x^0_{i_k})\right)_k\right) \times v_k$ and $\overline{\nabla_x F} = \frac{1}{n}\sum_{i=1}^{n} \nabla_x F(x^0_i)$ where $v_k = \max_{i\in\{1\dots,n\}} \left|\left(\nabla_x F(x^0_i)\right)_k\right|$ and $i_k = \text{argmax}_{i\in\{1\dots,n\}} \left|\left(\nabla_x F(x^0_i)\right)_k\right|$.

The quantity $\widehat{\nabla_x F}$ accounts for the maximum variation of the network response in every component direction of the network input encountered during the training and gives therefore a tighter linear upper bound than $\mathcal{W}$ when input data are part of the training set. The quantity $\overline{\nabla_x F}$ does not provide a linear upper bound but estimates an average linear behavior of the network response that can be used in practice to estimate the required Γ value to reduce $\mathbb{P}_{\epsilon\sim D}\left(\|x^L - x^L_\epsilon\| \geq \Gamma\right)$ below α. The larger is the training dataset, the higher is the quality of these estimates. Replacing $\mathcal{W}$ by these quantities in (12), we derive the following robustness bound for the network:

$$\left\|\widehat{\nabla_x F}\,\widehat{\nabla_x F}^\top\right\|_F \lesssim \frac{\Gamma^2}{\gamma^2\sqrt{2\log(1/\alpha)}}. \tag{5}$$

and the following estimate of Γ to achieve a robustness confidence level of $1-\alpha$:

$$\Gamma \gtrsim \gamma\left(2\log(1/\alpha)\|\overline{\nabla_x F}\,\overline{\nabla_x F}^\top\|_F^2\right)^{\frac{1}{4}}. \tag{6}$$

3. Controlling the bound during training

In this section, we are interesting in exploiting the bounds derived above during the process of training the neural network. The main idea would be to ensure that optimal weights after training are satisfying the bound constraint (5). Naturally, this could be formulated as a constrained optimization training problem. Stochastic projected gradient techniques (Nedic & Lee, 2014; Lacoste-Julien et al., 2012) could be used to solve such a problem. However, in the general case, the projection operator for such constraint is not simple and would require important computational effort. Therefore, instead of ensuring the constraint, we propose to regularize the loss function during training by adding a penalization term as follows:

$$\min_{\theta=(W,b)} \frac{1}{\lambda}\mathcal{L}(x,y,\theta) + \|Q_W\|_F^2$$

where λ is a positive parameter, $\mathcal{L}$ is a loss function (mean squared error for example) and $\|Q_W\|_F$ is the Frobenius norm of a matrix Q_W that could be chosen as $\widehat{\nabla_x F}\,\widehat{\nabla_x F}^\top$, $\overline{\nabla_x F}\,\overline{\nabla_x F}^\top$ or $\mathcal{W}\mathcal{W}^\top$, depending on which bound from above we want to exploit. Regularization is a common practice in machine learning (Bishop, 2006; Goodfellow et al., 2016) and is usually proposed to avoid overfitting and increase model generalization. The connection between generalization and robustness to input uncertainty in machine learning models has been established in several studies (Xu et al., 2009; Staib & Jegelka, 2019). Intuitively, the $\|Q_W\|_F^2$ regularization term acts as a special weight contraction and it is natural to consider alternative possibilities to reduce the magnitude of the network weights. One alternative is the squared spectral norm of $\mathcal{W}$ (largest eigenvalue of Q_W) that would also account for the maximum absolute contraction of a vector when multiplied by Q_W. Finally, in (Couellan, 2019), the product $\prod_{l=1}^{L}\|W^l\|$ which is an upper bound of the Lipschitz constant of the network has also been proposed as regularization that promotes robustness. It accounts for the overall Lipschitz regularity of the network and acts also as an overall control on the contraction power of the network by coupling layers and allowing some weights to grow for some layers as long as in other layers others weights are getting smaller to compensate. When $Q_W = \mathcal{W}\mathcal{W}^\top$, its Frobenius norm and the Lipschitz constant gradient can be explicitely derived and integrated into the backpropagation scheme and chain rule of gradients in order to optimize the augmented loss during the training phase. However, for the spectral norm, approximation methods are necessary and gradient will have to be computed using numerical differentiation techniques. In Appendix B, we discuss several available approximation methods and in the next section, we propose to carry out experiments with these various regularization strategies and evaluate their respective impact on the robustness properties of the network.

4. Experiments

In order to assess the quality of the calculated bound, experiments are conducted on public datasets. We focus on deep neural network regression tasks (linear output activation) and the `BOSTON` (Harrison & Rubinfeld, 1978), `DIABETES` (Tibshirani et al., 2004) and `CALIFORNIA` (Pace & Barry, 1997) datasets. The neural network and its training and testing are implemented in the `python` (Team, 2015) environment using the `keras` (Chollet et al., 2015) library and `Tensorflow` (Abadi et al., 2015) backend. The neural network architecture is composed of 4 dense hidden layers with 50 ReLu activations neuronal units and one dense linear output layer. All results that are presented below are average results from 10 independent runs that are carried out after random shuffling and random splitting of datasets. All dataset samples are scaled so that they lie in $[-1,1]$. All neural network training procedures are executed with the `ADAM` stochastic optimization algorithm with default parameters as given in (Kingma & Ba, 2015). Additionnal details about the datasets dimensions and training parameters are given in Table 1 of Appendix C. All comparison results provided below are referring to training with mean squared loss and the following regularization schemes as described in section 3:

- `no reg`: without regularization

- `Lipschitz reg`: $\prod_{l=1}^{L} \|W^l\|$ regularizer as described in (Couellan, 2019).
- `Gradient reg`: $\|Q_W\|^2$ regularizer as described above and where $Q_W = \widehat{\nabla_x F}\,\widehat{\nabla_x F}^\top$.
- `MaxEig reg`: $\lambda_{max}(Q_W)^2$ regularizer.

In order to estimate the probability $\mathbb{P}_{\epsilon\sim D}\left(\|y - y_\epsilon\| \leq \Gamma\right)$, the following procedure is applied. For each test sample from the validation set, we generate random vectors ϵ_j with $j \in \{1, \ldots, 10\}$ and calculate the following probability estimate:

$$\frac{1}{10 \times T} \sum_{i=1}^{T} \sum_{j=1}^{10} \mathbf{1}_{\{\|y^{(i)} - y^{(i)}_{\epsilon_j}\| \leq \Gamma\}}(\epsilon_j) \tag{7}$$

where T is the number of samples in the testing set, $y^{(i)}$ is the desired output for the i-th testing sample and $y^{(i)}_{\epsilon_j}$ is the output of the network calculated via a forward pass through the network for the input vector $x^{(i)} + \epsilon_j$. In all experiments, we have $supp(D) = [-\gamma, \gamma]$ with various levels of γ. Results are reported in the figures of Appendix D where γ is referred as `gamma` and Γ as `Gamma`. Figure 1(right) reports these observed probabilities together with the estimated tail probabilities given by $\exp\{-\Gamma^4/(2\gamma^4\|\overline{\nabla_x F}\,\overline{\nabla_x F}^\top\|_F^2)\}$ for various values of Γ, while Figure 1(left) reports the corresponding mean validation error achieved during the training process. The probability level $1 - \alpha$ (with $\alpha = 0.05$) is also marked with a blue dashed line on each plot on the right. Figure 2 provides further details about the magnitude of the norm of the network gradient $\|\widehat{\nabla_x F}\|$ and $\lambda_{max}(Q_W)^2$ (re-scaled by a factor 10 to ease the reading of the plot) for each dataset and the four regularization strategies. Finally, Figure 3 provides, for each dataset and each regularization strategies a comparison of the Γ values achieved to reach a $1 - \alpha$ probability level. Three values are reported each time, $\Gamma(max) = (2\log(1/\alpha))^{1/4}\sqrt{\|\widehat{\nabla_x F}\,\widehat{\nabla_x F}^\top\|_F}$, $\Gamma(mean) = (2\log(1/\alpha))^{1/4}\sqrt{\|\overline{\nabla_x F}\,\overline{\nabla_x F}^\top\|_F}$ and the Γ value observed so that the probability given in (7) reaches a level $1 - \alpha$.

We see in Figure 1 that, for the three datasets and for a probability of 0.95, the calculated Γ value (x axis) obtained by the expression of the exponential tail probability, provides a very good estimate of the Γ value given by the observed probability (probability that the output deviates from its nominal value by more than Γ). This validates experimentally, at least for these datasets, the relevance of the estimate given in (6). For the `BOSTON` dataset, all regularizing strategies give similar Γ values whereas for the `DIABETES` and `CALIFORNIA`, the Γ values are more sensitive to the type of regularization employed. However, surprisingly, no general rule can be given from these results. It is difficult to say which regularizer performs best. It is dataset dependent. The left hand side plots of Figure 1, representing the validation mean absolute error, show that similar training performance were achieved by the four regularizing methods and do not provide further explanation of this phenomenon. We believe, without providing any evidence of it, that the high nonlinearity of the neural network error surface may explain it. Indeed, after training, the optimization algorithm has reached a local minimum where the loss value may not have decreased sufficiently to really express the regularization power of the regularizer. This depends on the geometry of the error surface that is greatly dependent on the input data.

On Figure 2, the norm of the network gradient tends to be slightly smaller for the `Gradient reg` strategy. This would confirm that regularizing by the network gradient would help in achieving better robustness. Additionally, the figure also shows that the maximum eigenvalue regularization is not correlated to the network gradient norm and may not be a suitable alternative for robustness purposes. The Γ value comparison in Figure 3 confirms that the Γ estimates calculated by the proposed method are very closed to the observed values. This is true for all datasets and regularizing strategies. Furthermore, the Γ upper bound values $\Gamma(max)$, are loose as expected in Section 2 but provide certificates for robustness. These certificates follow the same pattern as the norm of the network gradient in Figure 2, which was also expected since their expression in (5) are directly dependent. Therefore, as for the network gradient, we observed that these certificates are better (tighter but still quite loose) for the network gradient regularizing strategy.

5. Conclusions

In this study, we have proposed analytical probabilistic estimates (and certificates) for deep dense neural networks. The idea combines tail probability bound calculation using the Cramer-Chernoff scheme and the estimation of the network local variation. The network gradient computation is using the automatic differentiation procedure available in many neural network training packages and carried out only at the training samples which does not require much extra computational cost. Experiments with this method has been conducted on public datasets and has shown that the robustness estimates are very good compared to the observed network robustness. Further analysis on these datasets show that the quality of the estimates is not really impacted by the regularization strategy, however, the network gradient regularization tends to generate slightly more robust network architectures.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Allaire, G. *Numerical Analysis and Optimization*. Oxford Science Publications, 2007.

Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Boopathy, A., Weng, T.-W., Chen, P.-Y., Liu, S., and Daniel, L. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks, 2018.

Boucheron, S., Lugosi, G., and Massart, P. *Concentration Inequalities, a nonasymptotic theory of independence*. Oxford, 2013.

Chollet, F. et al. Keras. `https://keras.io`, 2015.

Couellan, N. The coupling effect of Lipschitz regularization in deep neural networks. working paper or preprint, 2019. URL `https://hal-enac.archives-ouvertes.fr/hal-02090498`.

Dembo, A. Bounds on the extreme eigenvalues of positive-definite toeplitz matrices. *IEEE Transactions on Information Theory*, 34(2):352–355, 1988.

Fawzi, A., Moosavi-Dezfooli, S., and Frossard, P. The robustness of deep networks: A geometrical perspective. *IEEE Signal Processing Magazine*, 34(6):50–62, 2017.

Finlay, C., Oberman, A., and Abbasi, B. Improved robustness to adversarial examples using lipschitz regularization of the loss. *arXiv:1810.00953v3*, 2018.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv:1804.04368v2*, 2018.

Harrison, D. and Rubinfeld, D. Hedonic prices and the demand for clean air. *J. Environ. Economics and Management*, 5:81–102, 1978.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Kolter, J. Z. and Wong, E. Provable defenses against adversarial examples via the convex outer adversarial polytope. *CoRR*, abs/1711.00851, 2017.

Lacoste-Julien, S., Schmidt, M., and Bach, F. A simpler approach to obtaining an o(1/t) convergence rate for the projected stochastic subgradient method, 2012.

Lanczos, C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4):255 – 282, 1950.

Nedic, A. and Lee, S. On stochastic subgradient mirror-descent algorithm with weighted averaging. *SIAM Journal on Optimization*, 24(1):84–107, 2014.

Oberman, A. and Calder, J. Lipschitz regularized deep neural networks converge and generalize. *arXiv:1808.09540v3*, 2018.

Pace, R. K. and Barry, R. Sparse spatial autoregressions. *Statistics and Probability Letters*, 33(3):291 – 297, 1997.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Staib, M. and Jegelka, S. Distributionally robust optimization and generalization in kernel methods, 2019.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., and et al., I. G. Intriguing properties of neural networks. *International Conference on learning representations (ICLR)*, 2014.

Team, P. C. Python: A dynamic, open source programming language, python software foundation. URL https://www.python.org/, 2015.

Tibshirani, R., Johnstone, I., Hastie, T., and Efron, B. Least angle regression. *The Annals of Statistics*, 32(2):407499, 2004.

Virmaux, A. and Scaman, K. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 2018.

Weng, L., Chen, P.-Y., Nguyen, L., Squillante, M., Boopathy, A., Oseledets, I., and Daniel, L. PROVEN: Verifying robustness of neural networks with a probabilistic approach. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6727–6736, 2019.

Xu, H., Caramanis, C., and Mannor, S. Robustness and regularization of support vector machines. *J. Mach. Learn. Res.*, 10:1485–1510, 2009.

A. Proofs

Proposition A.1 *If the network inputs are subject to an additive uncertainty ϵ where $\forall i = 1, \dots, n$, $\epsilon_i \sim D$ and $\mathrm{supp}(D) \in [-\gamma, \gamma]$ ($\gamma < +\infty$), then for a given $\alpha \in [0,1]$ and a given output uncertainty level Γ, the following condition holds:*
If the layer weights vector w satisfies $\|ww^\top\|_F \leq \frac{\Gamma^2}{\gamma^2\sqrt{2\log(1/\alpha)}}$, then $\mathbb{P}_{\epsilon\sim D}\left(\|y - y_\epsilon\| \leq \Gamma\right) \geq 1 - \alpha$.

Proof: We start by observing that in general we have

$$\mathbb{P}_{\epsilon\sim D}\left(\|y - y^\epsilon\| \leq \Gamma\right) \geq \mathbb{P}_{\epsilon\sim D}\left(tr\left(w^\top\epsilon(w^\top\epsilon)^\top\right) \leq \Gamma^2\right), \tag{8}$$

where $tr(A)$ defines the trace of a matrix A. Note that in the particular case where all activation functions are linear, equality is achieved.
Therefore inequality (1) may be satisfied by ensuring that

$$\mathbb{P}_{\epsilon\sim D}\left(tr(w^\top\epsilon(w^\top\epsilon)^\top) \geq \Gamma^2\right) \leq \alpha,$$

or

$$\mathbb{P}_{\epsilon\sim D}\left(tr(\epsilon\epsilon^\top ww^\top) \geq \Gamma^2\right) \leq \alpha.$$

We now state and prove the following Lemma which provides a simple concentration inequality for the above probability.

Lemma A.2 *For any random matrix $M \in \mathbb{R}^{n\times n}$ of the form $M = vv^\top$ where v is a random vector such that for all i in $\{1, \dots, n\}$, v_i are all independent, have finite support included in $[-\delta, \delta]$ and $\mathbb{E}(v_i) = 0$, we have*

$$\forall Q \in \mathbb{R}^{n\times n}, \forall t > 0, \quad \mathbb{P}\left(tr(MQ) \geq t\right) \leq e^{-\frac{t^2}{2\beta^2\delta^4\|Q\|_F^2}}. \tag{9}$$

where $\beta = \min\{\rho > 0 \, s.t. \, \frac{(\rho z)^2}{2} - \log(cosh(z)) \geq 0\}$.

Proof: The proof is based on the Cramer-Chernoff method (Boucheron et al., 2013) to bound the tail probability of the random variable $tr(MQ)$. Applying Markov inequality to the left hand side of (9), we have:

$$\mathbb{P}\left(\mathrm{tr}(MQ) \geq t\right) \leq \frac{\mathbb{E}(\mathrm{tr}(MQ))}{t}$$

and for any $p \in \mathbb{R}^+$, since $m_{ij} = v_iv_j = m_{ji}$,

$$\begin{aligned}\mathbb{P}\left(tr(MQ) \geq t\right) &\leq e^{-pt}\mathbb{E}\left(e^{p\,tr(MQ)}\right)\\ &\leq e^{-pt}\mathbb{E}\left(e^{p\sum_{i,j=1}^n m_{ij}q_{ji}}\right)\end{aligned}$$

leading to

$$\begin{aligned}&\mathbb{P}\left(tr(MQ) \geq t\right)\\ &\leq e^{-pt}\mathbb{E}\left(\left(\prod_{i=1}^n e^{pm_{ii}q_{ii}}\right)\left(\prod_{i,j=1:i<j}^n e^{2pm_{ij}q_{ji}}\right)\right).\end{aligned} \tag{10}$$

Observe now that since for all $i \neq j$, we have that $\mathrm{cov}(m_{ii}, m_{jj})$=0 and since $\mathbb{E}(v_i) = \mathbb{E}(v_j) = 0$ that

$$\mathrm{cov}(m_{ii}, m_{ij}) = \mathrm{cov}(v_iv_i, v_iv_j) = 0.$$

Therefore, from (10), we have

$$\mathbb{P}\left(tr(MQ) \geq t\right) \leq e^{-pt}\prod_{i,j=1}^n \mathbb{E}\left(e^{pm_{ij}q_{ji}}\right).$$

Let now $\psi_{\mathrm{tr}(MQ)}$ be the moment generating function of $\mathrm{tr}(MQ)$, its Cramer transform obtained by Fenchel-Legendre duality is

$$\psi^*_{\mathrm{tr}(MQ)}(t) = \sup_{p\geq 0}(pt - \psi_{\mathrm{tr}(MQ)}(z)).$$

Note that $m_{ij} \in [-\delta^2, \delta^2]$ and that by convexity of the exponential function, we can bound the moment generating function of m_{ij} as follows:

$$\forall z \in \mathbb{R}, \quad \mathbb{E}(e^{zm_{ij}}) \leq \frac{1}{2}e^{-z\delta^2} + \frac{1}{2}e^{z\delta^2} = cosh(z\delta^2).$$

Therefore, around zero, we can write

$$\log(\mathbb{E}(zm_{ij})) \leq \log(cosh(z\delta^2)) \leq \frac{(\beta z\delta^2)^2}{2}$$

where β is a coefficient in $(0, 1]$ that tightens the bound as much as possible and can be defined as $\beta = \min\{\rho > 0 \,\text{s.t.}\, \frac{(\rho z)^2}{2} - \log(cosh(z)) \geq 0\}$. Replacing z by pq_{ji} with $p > 0$, we can derive the following upper bound for the Cramer transform of $\psi^*_{\mathrm{tr}(MQ)}$:

$$\forall t \in \mathbb{R}, \quad \psi^*_{\mathrm{tr}(MQ)}(t) \leq \min_{p>0}\left\{-pt + \sum_{i,j=1}^n \frac{(pq_{ji}\beta\delta^2)^2}{2}\right\}.$$

The minimum in the right hand side of the expression above is reached at $p^* = \frac{t}{\beta^2\delta^4\|Q\|_F^2}$ and therefore, applying Cramer inequality, we finally get

$$\log(\mathbb{P}\left(\mathrm{tr}(MQ) \geq t\right)) \leq -\frac{t^2}{2\beta^2\delta^4\|Q\|_F^2},$$

which completes the proof of the lemma. □

Note that $\forall (i,j) \in \mathbb{R}^{n\times n}$, we have $(\epsilon\epsilon^\top)_{ij} \in [-\gamma^2, \gamma^2]$. Hence, applying Lemma A.2 to bound $\mathbb{P}_{\epsilon\sim D}\left(\mathrm{tr}(\epsilon\epsilon^\top ww^\top) \geq \Gamma^2\right)$ will lead to

$$\begin{aligned}\mathbb{P}_{\epsilon\sim D}\left(\mathrm{tr}(\epsilon\epsilon^\top ww^\top) \geq \Gamma^2\right) &\leq e^{-\frac{\Gamma^4}{2\beta^2\gamma^4\|ww^\top\|_F^2}} \\ &\leq e^{-\frac{\Gamma^4}{2\gamma^4\|ww^\top\|_F^2}},\end{aligned}$$

since $\beta \in (0,1]$. In order to remain below the level α, we then need that

$$-\frac{\Gamma^4}{2\gamma^4\|ww^\top\|_F^2} \leq \log(\alpha). \tag{11}$$

This can also be written as $\|ww^\top\| \leq \frac{\Gamma^2}{\gamma^2\sqrt{2\log(1/\alpha)}}$, proving Proposition 2.1.

Note that in (11), one can keep the tightening coefficient β to get a sharper bound whenever needed and write $\|ww^\top\| \leq \frac{\Gamma^2}{\beta\gamma^2\sqrt{2\log(1/\alpha)}}$. The value of the coefficient of the matrix β can be estimated numerically. □

Proposition A.3 *If the network inputs are subject to an additive uncertainty ϵ where $\forall i = 1,\dots,n$, $\epsilon_i \sim D$ and $\mathrm{supp}(D) \in [-\gamma,\gamma]$ ($\gamma < +\infty$), then for a given $\alpha \in [0,1]$ and a given output uncertainty level Γ, the following condition holds:*
If the layer weights vector w satisfies $\left\|\left(\prod_{l=1}^L W^l\right)\left(\prod_{l=1}^L W^l\right)^\top\right\|_F \leq \frac{\Gamma^2}{\gamma^2\sqrt{2\log(1/\alpha)}}$ then $\mathbb{P}_{\epsilon\sim D}\left(\|x^L - x_\epsilon^L\| \leq \Gamma\right) \geq 1-\alpha$.

Proof: Propagating forward the input uncertainties through the network, we can write:

$$\begin{aligned}x^L - x_\epsilon^L &= (W^L)^\top \epsilon^{L-1} \\ &= (W^L)^\top (W^{L-1})^\top \epsilon^{L-2} \dots (W^1)^\top \epsilon \\ &= \left(\prod_{l=1}^L W^l\right)^\top \epsilon\end{aligned}$$

where ϵ^l is the propagated noise from layer 1 to layer l. Furthermore, we have

$$\begin{aligned}\left\|\left(\prod_{l=1}^L W^l\right)^\top \epsilon\right\|^2 &= \mathrm{tr}\left(\left(\prod_{l=1}^L W^l\epsilon\right)\left(\prod_{l=1}^L W^l\epsilon\right)^\top\right) \\ &= \mathrm{tr}\left(\left(\prod_{l=1}^L W^l\right)\epsilon\epsilon^\top\left(\prod_{l=1}^L W^l\right)^\top\right) \\ &= \mathrm{tr}\left(\epsilon\epsilon^\top\left(\prod_{l=1}^L W^l\right)\left(\prod_{l=1}^L W^l\right)^\top\right).\end{aligned}$$

Therefore, applying again Lemma A.2 to upper bound the probability in (2), the following condition on the layer weights matrices to ensure property (2) is directly obtained:

$$\left\|\left(\prod_{l=1}^L W^l\right)\left(\prod_{l=1}^L W^l\right)^\top\right\|_F \leq \frac{\Gamma^2}{\gamma^2\sqrt{2\log(1/\alpha)}}. \tag{12}$$

□

B. Spectral norm approximation

Among available approximation methods, the power iteration algorithm (Allaire, 2007), or preferably the Lanczos algorithm (Lanczos, 1950) since Q_W is symmetric, is well suited for the purpose. Note that there is no real need to approximate $\lambda_{max}(Q_W)$ (the largest eigenvalue of Q_W) with great accuracy as it is only used to as a regularization function to guide the optimization process towards optimal regions where the spectral norm is reduced. Therefore, an alternative approximation technique is to use an upper bound of $\lambda_{max}(Q_W)$. As Q_W is positive definite, we propose to use the Dembo's upper bound (Dembo, 1988) defined as follows:
Let $A_n \in \mathbb{R}^{n\times n}$ be an Hermitian positive definite matrix and let $\lambda^(n)_1 \leq \dots \lambda^(n)_n$ be the eigenvalues of A_n. The matrix A_n can be written as

$$A_n = \begin{pmatrix} A_{n-1} & b \\ b^H & c\end{pmatrix}$$

where b^H denotes the Hermitian transpose and the largest eigenvalue of A_n satisfies

$$\lambda_n^{(n)} \leq \frac{c+\lambda_n^{(n-1)}}{2} + \sqrt{\frac{(c-\lambda_n^{(n-1)})^2}{4} + b^H b}.$$

C. Tables

Table 1. Dataset information and experimental setup

Dataset	# inputs	# samples	test/train ratio	batch size	# epochs	learning rate
BOSTON	13	606	0.2	50	100	0.001
DIABETES	10	442	0.2	200	30	0.001
CALIFORNIA	8	20640	0.4	600	30	0.001

D. Figures

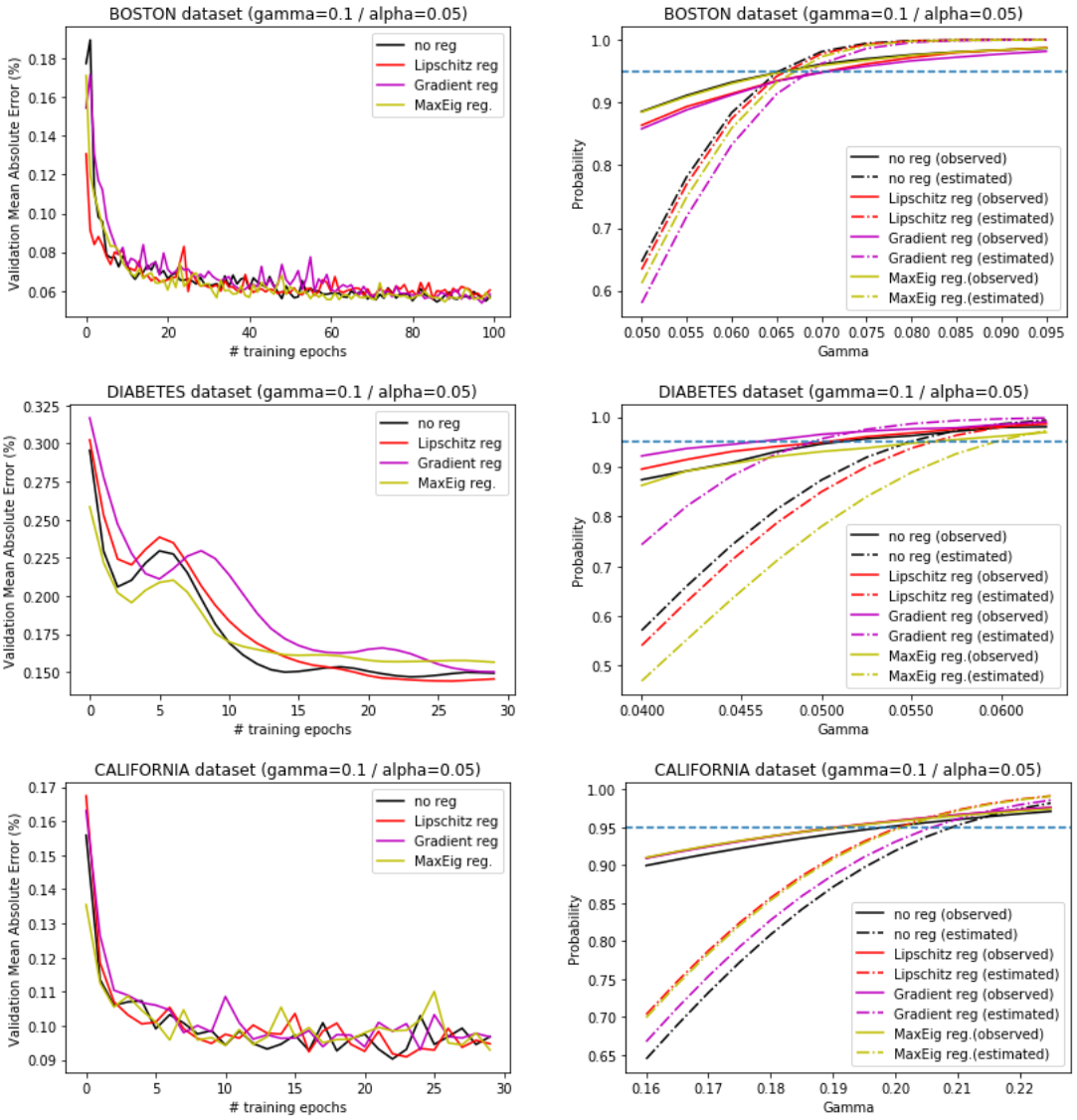

Figure 1. Mean absolute validation error profiles during training (left) & Comparison of $\mathbb{P}_{\epsilon\sim D}\left(\|y - y_\epsilon\| \leq \Gamma\right)$ and exponential tail bound for various levels of Γ (right)

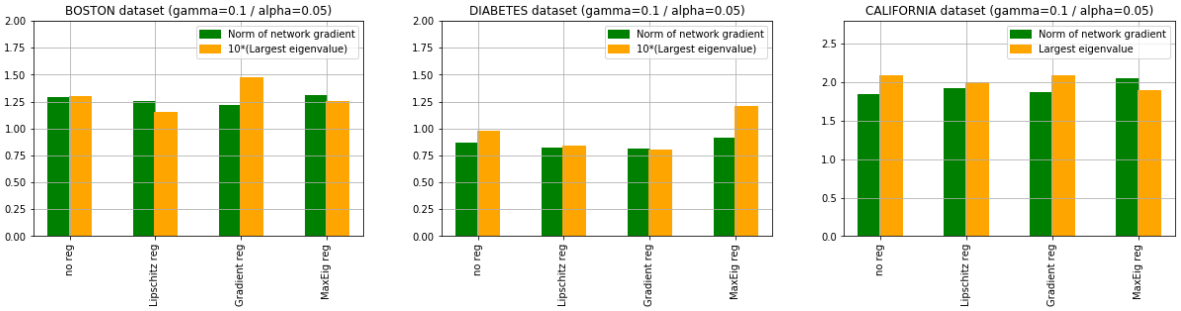

Figure 2. Norm of network gradient $\|\widehat{\nabla_x F}\|$ and Estimate of $\lambda_{max}(Q_W)$ after training

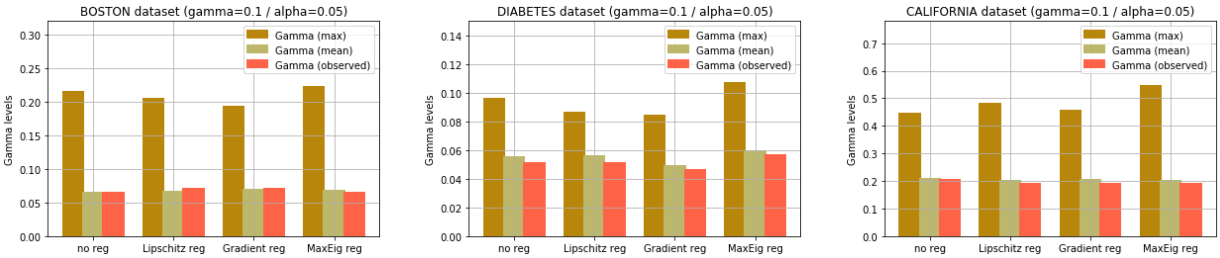

Figure 3. Comparison of calculated $\Gamma(max)$ and $\Gamma(mean)$ values for 0.95 confidence level using $\widehat{\nabla_x F}$ and$\overline{\nabla_x F}$ respectively with the $\Gamma(observed)$ value observed so that the probability given in (7) reaches a level 0.95