Adversarial Regularization

Jingfeng Wu

Created: May 2018
Last updated: June 27, 2020

1 Adversarial Training

Thanks to FGSM [1], adversarial training could be employed efficiently.

1.1 Fast Gradient Sign Method

Given loss function $J(\theta, x, y)$, its linearization is

$$J(\theta, x+\eta, y) \approx J(\theta, x, y) + \nabla_\theta J(\theta, x, y)^T \eta.$$

The adversarial perturbation could be recognized as the solution of the following linear optimization problem,

$$\begin{aligned} \max_{\eta} \quad & J(\theta, x+\eta, y) = J(\theta, x, y) + \nabla_\theta J(\theta, x, y)^T \eta \\ \text{s.t.} \quad & \|\eta\|_\infty \le \epsilon. \end{aligned}$$

Obviously, the optimal value is

$$\eta^* = \epsilon \operatorname{sign} \nabla_\theta J(\theta, x, y).$$

The so called *fast gradient sign method* (FGSM) is to assume that $x + \eta^*$ is the adversarial example of x. Though FGSM might not find the least perturbed adversarial example as deep fool[4], the main advantage of which is the adversarial example could be identified by merely once back propagation, which makes adversarial training practical.

Note that the approximate solution with respect to L_2 constraint could be obtained similarly as

$$\eta^* \approx \epsilon \frac{\nabla J(\theta, x, y)}{\|\nabla J\|_2}.$$

Ignoring the computational burden, one can use proximal gradient descent (PGD) to find more reliable adversarial examples [2].

1.2 Adversarial Regularization

The FGSM adversarial training regularization works as,

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1-\alpha) J(\theta, x + \epsilon \operatorname{sign} \nabla_\theta J(\theta, x), y).$$

To take a closer look, apply Taylor's expansion,

$$\begin{aligned} \tilde{J}(\theta, x, y) &\approx \alpha J(\theta, x, y) + (1-\alpha)\left(J(\theta, x, y) + \nabla_\theta J(\theta, x, y)^T \epsilon \operatorname{sign} \nabla_\theta J(\theta, x, y)\right) \\ &= J(\theta, x, y) + (1-\alpha)\epsilon \left\|\nabla_\theta J(\theta, x, y)\right\|_1^2. \end{aligned}$$

Thus adversarial training with FGSM, in sense of **small** perturbation, is actually penalizing the L_1 norm of gradient of loss with respect to input, which brings smoothness to the classifier.

2 Virtual Adversarial Training

The spirit of VAT [3] is to find the direction where the output of the classifier changes most. Concisely, let q be the true distribution and p be the model, then the original real adversarial training regularization is,

$$\begin{aligned} \max_r \quad & D\left[q(y|x), p(y|x + r_{\text{qadv}}, \theta)\right] \\ \text{s.t.} \quad & \|r\|_2 \leq \epsilon, \end{aligned}$$

while the virtual adversarial training regularization is,

$$\begin{aligned} \max_r \quad & D\left[p(y|x), p(y|x + r, \theta)\right] \\ \text{s.t.} \quad & \|r\|_2 \leq \epsilon. \end{aligned}$$

Here in some cases we omit θ of $p(y|x)$, to emphasize that the expression is independent with θ, i.e., gradients cannot not back propagate through this part.

Define the following notations,

$$\begin{aligned} D(r, x, \theta) &:= D\left[p(y|x), p(y|x + r, \theta)\right] \\ r_{vadv} &:= \arg\max_{\|r\|_2 \leq \epsilon} D(r, x, \theta) \\ R_{\text{vadv}} &:= \max_{\|r\|_2 \leq \epsilon} D(r, x, \theta) = D(r_{vadv}, x, \theta). \end{aligned}$$

Then the whole VAT loss is,

$$\mathbb{E}_{\text{labeled data}} \ell(x, y, \theta) + \alpha \mathbb{E}_{\text{labeled or unlabled data}} R_{\text{vadv}}.$$

To identify the close form solution of r_{vadv}, consider the second-order Taylor approximation of $D(r, x, \theta)$ at $r = 0$,

$$D(r, x, \theta) \approx \frac{1}{2} r^T H(x, \theta) r.$$

Note that $D(0, x, \theta) = 0, \nabla_r D(0, x, \theta) = 0$, because D is a distance measure and its minimum is achieved when $r = 0$. Therefore

$$r_{\text{vadv}} = \arg\max_{\|r\|_2 \leq \epsilon} D(r, x, \theta) \approx \epsilon u(x, \theta),$$

where $u(x, \theta)$ is the unit eigenvector of $H(x, \theta)$ of the maximum eigenvalue. This eigenvalue problem could be effectively evaluated by power method.

2.1 Virtual Adversarial Regularization

For adversarial perturbation

$$r_{\text{vadv}} = \arg\max_{\|r\|_2 \leq \epsilon} D(r, x, \theta) \approx \epsilon u(x, \theta),$$

the virtual adversarial regularization is,

$$R_{\text{vadv}} = \max_{\|r\|_2 \leq \epsilon} D(r, x, \theta) \approx \max_{\|r\|_2 \leq \epsilon} \frac{1}{2} r^T H(x, \theta) r = \frac{1}{2} \epsilon^2 \left\| H(x, \theta) \right\|_2.$$

Thus the VAT loss is,

$$\begin{aligned} & \mathbb{E}_{\text{labeled data}} \ell(x, y, \theta) + \alpha \mathbb{E}_{\text{all data}} R_{\text{vadv}} \\ = & \mathbb{E}_{\text{labeled data}} \ell(x, y, \theta) + \frac{\alpha \epsilon^2}{2} \mathbb{E}_{\text{all data}} \left\| H(x, \theta) \right\|_2. \end{aligned}$$

Hence with a small perturbation, virtual adversarial regularization is equivalent to penalize the spectrum norm of the Hessian of $D(r, x, \theta)$ with respect to r.

In the perspective of loss $\ell(x, y, \theta)$,

$$\ell(x + r, p(y|x), \theta) = D\left[p(y|x), p(y|x + r, \theta)\right] = D(r, x, \theta).$$

Thus the Hessian of $D(r, x, \theta)$ w.r.t. r could also be viewed as the Hessian of $\ell(x+r, p(y|x), \theta)$ w.r.t. r, which is equal to the Hessian w.r.t. input x,

$$\nabla_r^2\, \ell(x_* + r, p(y_*|x_*), \theta) = \nabla_x^2\, \ell(x, y, \theta)|_{(x,y)=(x_*+r, p(y_*|x_*))}.$$

In summary, VAT penalizes the spectrum norm of the Hessian of ℓ with respect to the input. Different from L_p norm of Jacobian, VAT defines a new type of smoothness on classifier, which is shown to be effective for semi-supervised learning [3].

References

[1] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[2] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[3] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

[4] Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-218057, 2016.