[Lecture 19]

0.1 Randomized Smoothing

We next discuss a simpler, almost trivial approach to obtaining verified bounds, that nevertheless works very well in practice (it currently has the best certified bounds for ℓ_2 perturbations, and is efficient enough to scale to the ImageNet dataset).

The basic idea is as follows: suppose that we have some classifier $f_\theta : \mathbb{R}^d \to [0,1]^k$, which maps an input $x \in \mathbb{R}^d$ to a k-dimensional vector of class probabilities (so actually the range is $\Delta_k \subset [0,1]^k$). We can define a *smoothed classifier* $\bar{f}_\theta$ as

$$\bar{f}_\theta(x) = \mathbb{E}_{\delta \sim \pi}[f_\theta(x + \delta)]. \tag{1}$$

In other words, $\bar{f}_\theta$ applies f_θ to some randomly perturbed point $x + \delta$ that is close to x. Observe that we can approximate $\bar{f}_\theta$ well by sampling repeatedly from δ.

Let π_x be the distribution of $x + \delta$. The key bound underlying randomized smoothing lets us control the change in $\bar{f}_\theta$ in terms of a certain modulus of continuity:

Proposition 0.1. *Suppose that f_θ maps into $[0,1]^k$. Then for any x, x', we have*

$$\|\bar{f}_\theta(x) - \bar{f}_\theta(x')\|_\infty \leq \mathsf{TV}(\pi_x, \pi_{x'}). \tag{2}$$

In particular, if $d(x, x') \leq \epsilon$, then $\|\bar{f}_\theta(x) - \bar{f}_\theta(x')\|_\infty \leq \max\{\mathsf{TV}(\pi_x, \pi_{x'}) \mid d(x, x') \leq \epsilon\}$.

This says that $\bar{f}$ is stable under perturbations as long as the family of distributions π_x has bounded modulus of continuity of TV with respect to d (note this is the *opposite* direction of the modulus that we considered before).

The way to apply Proposition 0.1 is to somehow obtain a model such that the probability assigned to the correct class under $\bar{f}_\theta$ is at least τ larger than the probability of any incorrect class. Then as long as $\mathsf{TV}(\pi_x, \pi_{x'}) < \tau$ whenever $d(x, x') \leq \epsilon$, we know that no perturbation can change the $\arg\max$ prediction of $\bar{f}_\theta$. In the remainder of this section we will discuss how to choose π, and how to train $\bar{f}_\theta$.

Choosing the smoothing distribution π. We will restrict ourselves to the special case $d(x, x') = \|x - x'\|_2$, i.e. ℓ_2 perturbations. In this case we will take $\pi = \mathcal{N}(0, \sigma^2 I)$ for some σ. Then the modulus becomes

$$\max\{\mathsf{TV}(\mathcal{N}(0, \sigma^2 I), \mathcal{N}(\delta, \sigma^2 I)) \mid \|\delta\|_2 \leq \epsilon\} = \Phi(\epsilon/2\sigma) - \Phi(-\epsilon/2\sigma), \tag{3}$$

where Φ is the normal CDF. When ϵ/σ is small, the right-hand-side is $\Theta(\epsilon/\sigma)$, so we are automatically resistant to perturbations that are small in ℓ_2-norm compared to σ.

Observe that the *per-coordinate* noise we apply is comparable in magnitude to the *overall* norm of the perturbation. Thus in d dimensions, we need to apply noise that is $\sqrt{d}$ times larger than the adversarial perturbation that we seek robustness against. This matches the observation on random vs. adversarial noise for linear models from the previous section. Indeed, the above analysis is essentially tight for linear models (up to constants, and assuming ϵ/σ is small).

Training the model. Recalling that f_θ and $\bar{f}_\theta$ both output probability distributions over y, a natural training objective would be to minimize

$$\mathbb{E}_{(x,y)\sim p}[-\log(\bar{f}_\theta(x)_y)] = \mathbb{E}_{(x,y)\sim p}[-\log(\mathbb{E}_\delta[f_\theta(x + \delta)_y])], \tag{4}$$

i.e. the negative log probability that $\bar{f}_\theta(x)$ assigns to the true label y. However, the derivative of this quantity is inconvenient to work with:

$$\begin{aligned} \nabla_\theta[\log(\mathbb{E}_\delta[f_\theta(x+\delta)_y]] &= \mathbb{E}_\delta[\nabla_\theta[f_\theta(x+\delta)_y]]/\bar{f}_\theta(x)_y & (5) \\ &= \mathbb{E}_\delta[\frac{f_\theta(x+\delta)_y}{\bar{f}_\theta(x)_y} \nabla_\theta \log f_\theta(x+\delta)_y]. & (6) \end{aligned}$$

In particular, the importance weight $\frac{f_\theta(x+\delta)_y}{f_\theta(x)_y}$ could have high variation and so require many samples to obtain a good estimate. An alternative is to instead move the log inside the expectation and minimize

$$\mathbb{E}_{(x,y)\sim p}\mathbb{E}_\delta[-\log(f_\theta(x+\delta)_y)]. \tag{7}$$

Then we can compute stochastic gradients of the objective by sampling (x, y), sampling δ, and taking the gradient of $-\log f_\theta(x+\delta)_y$, which can generally be computed straightforwardly (e.g. via backpropagation in the case of neural networks).