Building Robust Ensembles via Margin Boosting

Dinghuai Zhang, Hongyang Zhang, Aaron Courville, Yoshua Bengio, Pradeep Ravikumar, Arun Sai Suggala

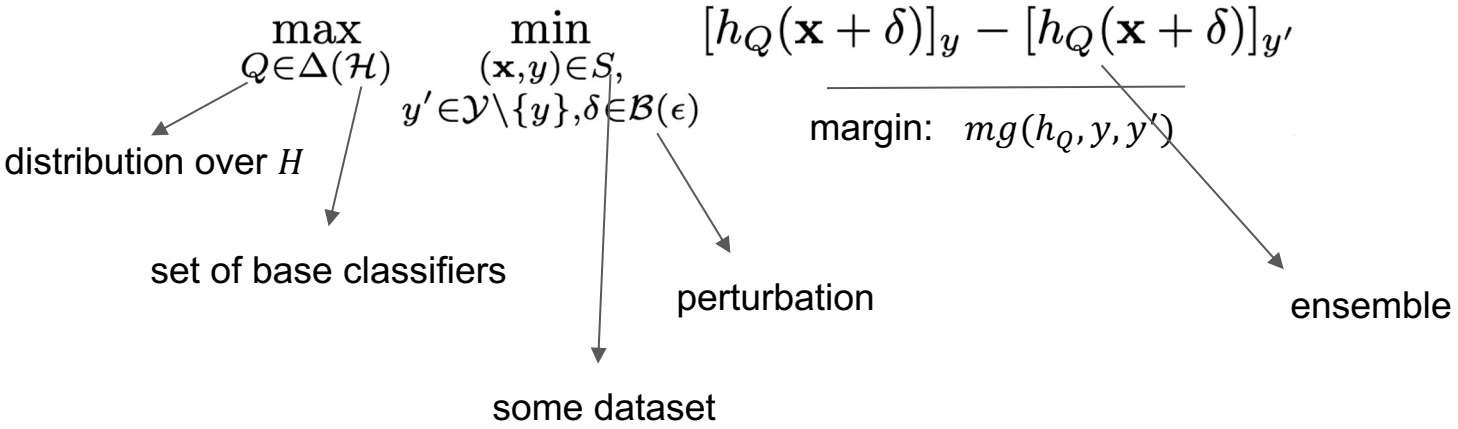
Motivation

Boosting algorithms aim to iteratively learn weak classifiers and combine them as an ensemble to form a strong classifier.

Can we combine multiple base classifiers into a strong classifier that is robust to adversarial attacks?

Margin-boosting framework

We propose a margin-boosting framework (Freund et al., 1996) for robustness



This is a two-player zero-sum game.

Freund, Y., Schapire, R. E., et al. Experiments with a new boosting algorithm. In icml, volume 96, pp. 148–156. Citeseer, 1996.

Optimality of margin boosting (informal)

We show that the following two arguments are equivalent:

- (weak learning condition) for any combination of data points, there exists a base classifier in H that performs slight better than random guessing.

$$\mathbb{E}_{(x,y,y',\delta)\sim P'}[1\{h(x+\delta)=y\}] \geq \mathbb{E}_{(x,y,y',\delta)\sim P'}[1\{h(x+\delta)=y'\}] + \tau$$

- the optimal solution of the aforementioned minimax game achieves perfect adversarial robustness.

A Robust Boosting Algorithm

Algorithm 1 MRBOOST

1: **Input:** training data S, boosting iterations T, learning rate η.
2: Let P_1 be the uniform distribution over S_{aug}.
3: **for** $t = 1 \dots T$ **do**
4: Compute $h_t \in \mathcal{H}$ as the minimizer of:

$$\min_{h \in \mathcal{H}} \mathbb{E}_{(\mathbf{x},y,y',\delta)\sim P_t}[\text{mg}_{\text{L}}\left(h(\mathbf{x}+\delta), y, y'\right)].$$

5: Compute probability distribution P_{t+1}, supported on S_{aug}, as:

$$P_{t+1}(\mathbf{x}, y, y', \delta) \propto \exp\left(\eta \sum_{j=1}^{t} \text{mg}_{\text{L}}\left(h_j(\mathbf{x}+\delta), y, y'\right)\right),$$

6: **end for**
7: **Output:** return the classifier $h^{\text{am}}_{Q(T)}(\mathbf{x})$, where $Q(T)$ is the uniform distribution over $\{h_t\}_{t=1\dots T}$.

online learning framework

$\{(\mathbf{x}, y, y', \delta) : (\mathbf{x}, y) \in S, y' \in \mathcal{Y} \setminus \{y\}, \delta \in \mathcal{B}(\epsilon)\}$

intractable 0-1 margin loss, need differentiable surrogate

need an efficient sampler

resulting “argmax” classifier from the ensemble $Q(T)$

Practical MRBoost.NN algorithm

Algorithm 2 MRBOOST.NN

1: **Input:** training data S, boosting iterations T, learning rate η, SGD iterations E, SGD step size γ, sampling sub-routine: SAMPLER.
2: **for** $t = 1 \dots T$ **do**
3: $\theta_t \leftarrow \begin{cases} \text{random initialization} & (\text{RNDINIT}) \\ \theta_{t-1} & (\text{PERINIT}) \end{cases}$
4: **for** $e = 1 \dots E$ **do**
5: Generate mini-batch

$$\{(\mathbf{x}_b, y_b, y'_b, \delta_b)\}_{b=1}^{B} \leftarrow \text{SAMPLER}\left(S, \{\theta_j\}_{j=1}^{t-1}, \eta\right)$$

6: Update g_{θ_t} using SGD:

$$\theta_t \leftarrow \theta_t - \frac{\gamma}{B} \sum_{b=1}^{B} \nabla_\theta \ell_{\text{MCE}}(g_{\theta_t}(\mathbf{x}_b + \delta_b), y_b, y'_b).$$

7: **end for**
8: **end for**
9: **Output:** Let $Q(T)$ be the uniform distribution over $\{g_{\theta_t}\}_{t=1\dots T}$. Output the classifier $g^{\text{am}}_{Q(T)}(\mathbf{x})$.

different initialization method

compute δ_b as

$$\delta_b \in \underset{\delta \in \mathcal{B}(\epsilon)}{\text{argmax}} \sum_{y' \in \mathcal{Y} \setminus \{y_b\}} \ell_{\text{MCE}}\left(\sum_{j=1}^{t} g_{\theta_j}(\mathbf{x}_b + \delta), y_b, y'\right)$$

we propose the differentiable **margin cross entropy** loss

$$\ell_{\text{MCE}}(g_\theta(\mathbf{x}), y, y') \coloneqq \ell_{\text{CE}}(g_\theta(\mathbf{x}), y) + \ell_{\text{CE}}(-g_\theta(\mathbf{x}), y')$$

$$\ell_{\text{CE}}(g(\mathbf{x}), y) \coloneqq -[g(\mathbf{x})]_y + \log\left(\sum_{j \in \mathcal{Y}} \exp\left[g(\mathbf{x})\right]_j\right)$$

Experiment results

Single classifier case: the proposed MCE loss consistently increase the robustness of many previous algorithms (more details in the paper)

Table 2. Experiments with WideResNet-34-10 on CIFAR10.

METHOD	CLEAN	FGSM	CW	PGD-20	PGD-100	AUTOATTACK
AT	**86.31**	64.01	53.28	54.12	53.75	50.13
AT + MCE	85.56	64.20	53.46	**55.40**	**55.14**	**52.07**
TRADES	83.25	62.48	49.51	54.97	54.80	51.92
TRADES + MCE	84.76	**64.63**	49.49	**56.23**	**55.99**	52.40
MART	83.12	63.68	52.57	55.75	55.49	50.85
MART + MCE	**83.65**	**64.3**	**54.24**	**56.31**	**56.15**	**52.81**
GAIR	83.91	65.79	49.44	58.99	58.97	44.04
GAIR + MCE	**84.55**	**67.96**	**49.94**	**61.79**	**61.93**	44.22
AWP	**85.32**	65.89	55.40	57.37	57.08	53.67
AWP + MCE	84.97	**66.53**	**56.23**	**58.40**	**58.12**	**54.69**

Experiment results

Multiple classifiers case: our proposed MRBoost.NN turns out to be a better robust boosting method than the baselines.

Table 3. Boosting experiments with ResNet-18 being the base classifier.

Method	Iteration 1		Iteration 2		Iteration 3		Iteration 4		Iteration 5	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
Wider model	82.61	51.73	—	—	—	—	—	—	—	—
Deeper model	82.67	52.32	—	—	—	—	—	—	—	—
RobBoost + RndInit	82.00	51.05	84.58	49.95	83.87	51.66	82.56	52.72	81.44	52.92
RobBoost + PerInit	82.18	50.97	85.60	50.13	84.59	51.77	84.21	52.79	82.78	53.28
MRBoost.NN + RndInit	81.04	51.83	84.61	52.68	84.93	53.51	85.01	53.95	85.35	54.13
MRBoost.NN + PerInit	81.34	51.92	84.97	52.97	85.28	53.62	85.99	54.26	86.16	54.42

Thank you very much!