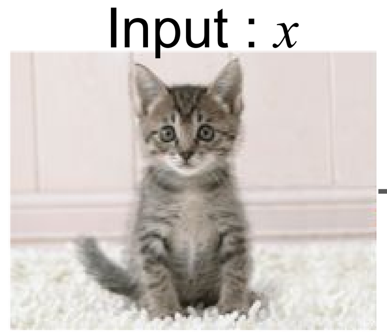
ID155: Distributional Smoothing with Virtual Adversarial Training

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, Shin Ishii (Kyoto University, Japan)

Introduction

Construct a "Good" classfier with new regularization method.

Input : x

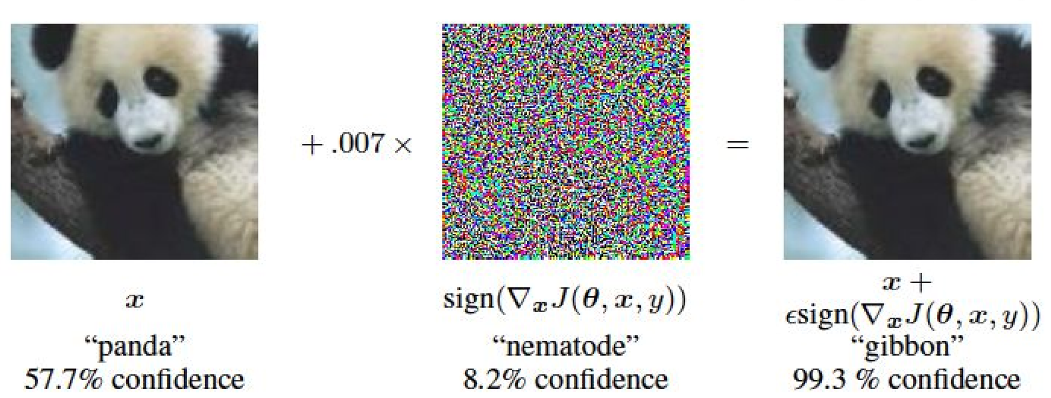


Model $p(y|x,\theta)$ → Output : y "Cat"

What are the properties of good classifier?

- Recognition should be robust againt minor perturbatation!

Adversarial example



[Goodfellow, et al 2015]

- Adversarial training ahchieved good generalization performance on MNIST.

$$\frac{1}{N}\sum_{n=1}^{N}\log p(y^{(n)}|x^{(n)},\theta) + \lambda\frac{1}{N}\sum_{n=1}^{N}\log p(y^{(n)}|x^{(n)} + r_{\mathrm{adv}}^{(n)},\theta)$$

$$r_{\mathrm{adv}} = -\epsilon\,\mathrm{sign}(\nabla_x \log p(y|x,\theta))$$

Adversarial examples for unlabeled inputs

- Define the difference between $p(y|x,\theta)$ and $p(y|x+r,\theta)$ as:

$$\Delta_{\mathrm{KL}}(r, x^{(n)}, \theta) \equiv \mathrm{KL}[p(y|x^{(n)},\theta)\|p(y|x^{(n)}+r,\theta)]$$

- Define *virtual* adversarial perturbation as:

$$r_{\text{v-adv}}^{(n)} \equiv \arg\max_r\{\Delta_{\mathrm{KL}}(r, x^{(n)}, \theta);\ \|r\|_2 \le \epsilon\}$$

(ϵ : *norm constraint*)

Virtual Adversarial Training (VAT)

- Local Distributional Smoothness(LDS)

$$\mathrm{LDS}(x^{(n)},\theta) \equiv -\Delta_{\mathrm{KL}}(r_{\text{v-adv}}^{(n)}, x^{(n)}, \theta)$$

- LDS regularized objective function

$$\frac{1}{N}\sum_{n=1}^{N}\log p(y^{(n)}|x^{(n)},\theta) + \lambda\frac{1}{N'}\sum_{n'=1}^{N'}\mathrm{LDS}(x^{(n')},\theta)$$

(λ : balance factor, N : num of labeled examples, N' : num of labeled and unlabeled examples)

Advantages of our method

- Applicability to both supervised and semi-supervised training.
- At most two hyperparameters (ϵ and λ).
  - In our experiments, only optimized ϵ.
- Parametrization invariant formulation.
- Low computational cost.

LDS for linear regression and logistic regression

- Linear regression,

$$\mathrm{LDS}(x,\theta) \propto \epsilon\|\theta\|^2$$

- Logistic regression $p(y=1|x,\theta)$

$$\mathrm{LDS}(x,\theta) \simeq \frac{1}{2}\sigma(\theta^{\mathrm{T}}x)\left(1-\sigma(\theta^{\mathrm{T}}x)\right)\|\theta\|^2$$

Computation of LDS

- Seemingly insurmountable bottleneck of computing LDS.

$$r_{\text{v-adv}} = \arg\max_r\{\Delta_{\mathrm{KL}}(r,x,\theta);\|r\|_2 \le \epsilon\}$$

- This can be computed fast! (See ↓)

Approximation of $r_{\text{v-adv}}$

- Approximate Δ_{KL} with 2nd Taylor expansion:

$$\Delta_{\mathrm{KL}}(r,x,\theta) \cong \frac{1}{2}r^T H(x,\theta) r$$

$$H(x,\theta) \equiv \nabla\nabla_r \Delta_{\mathrm{KL}}(r,x,\theta)|_{r=0}$$

- $r_{\text{v-adv}}$ is also the dominant eigenvector of $H(x,\theta)$ with magnitude ϵ :

$$r_{\text{v-adv}}(x,\theta) \cong \arg\max_r\{r^T H(x,\theta) r;\ \|r\|_2 \le \epsilon\} = \epsilon u(x,\theta),$$

$u(x,\theta)$ is 1st eigen vector of $H(x,\theta)$

→Eigen vector of $H(x,\theta)$ can be approximated easily with...

- Power iteration method :

$$d \leftarrow Hd$$

  - d is approaching 1st eigen vector of $H(x,\theta)$

- Finite difference method :

$$Hd \cong \frac{\nabla_r\Delta_{\mathrm{KL}}(r+\xi d,x,\theta)|_{r=0} - \nabla_r\Delta_{KL}(r,x,\theta)|_{r=0}}{\xi} = \frac{\nabla_r\Delta_{KL}(r+\xi d,x,\theta)|_{r=0}}{\xi},$$

→For nerual network, we can get $r_{\text{v-adv}}$ with just two additional back-propagations (one @ power method, one @ the grad of LDS)

Summary of approximation of $r_{\text{v-adv}}$

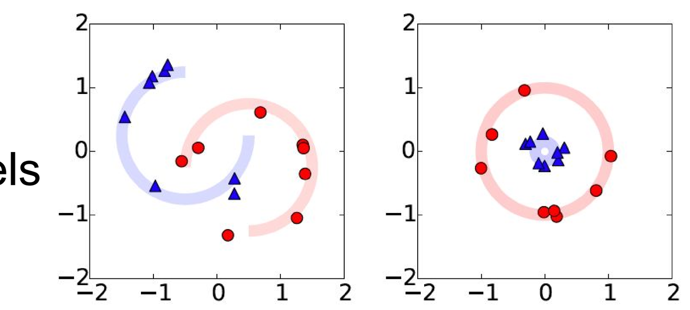
Algorithm 1 Generation of $\tilde{r}_{\text{v-adv}}^{(n)}$

Function GenVAP(θ,$x^{(n)}$,ϵ,I_p,ξ)
1. Initialize $d \in R^I$ by a random unit vector.
2. **Repeat For** i in $1 \dots I_p$ (Perform I_p-times power method)
   $d \leftarrow \nabla_r\Delta_{KL}(r+\xi d, x^{(n)},\theta)|_{r=0}$
3. **Return** ϵd

∗ In our experiment, we set the number of power iteration I_p to 1.

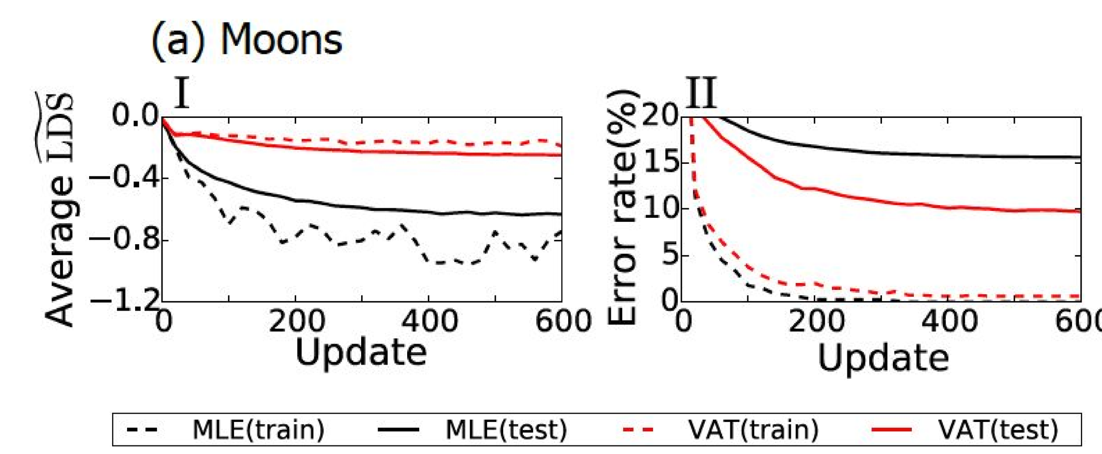
Demo : 2D synthetic dataset

- 1 hidden layer NN classifier for two labels

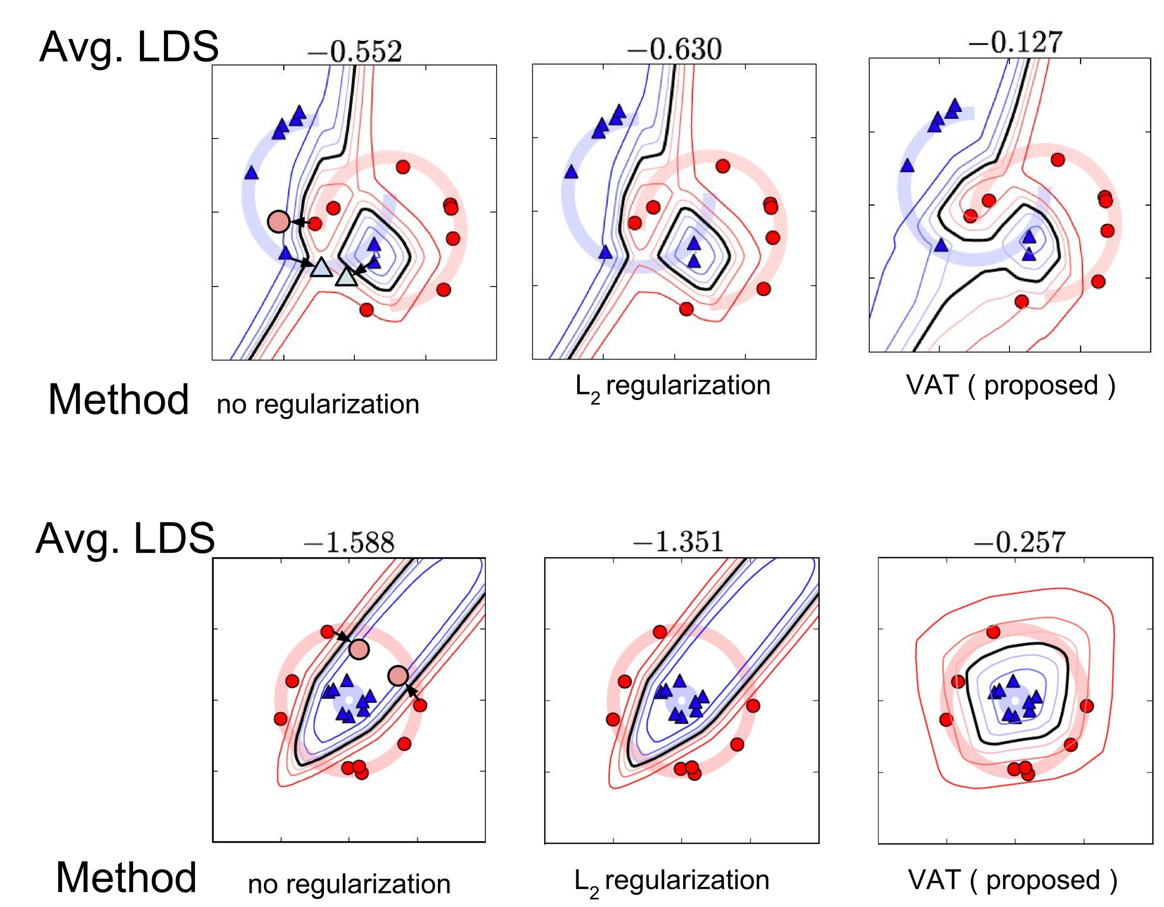


(a) Moons dataset (b) Circles dataset

- Learing curves on MLE and VAT



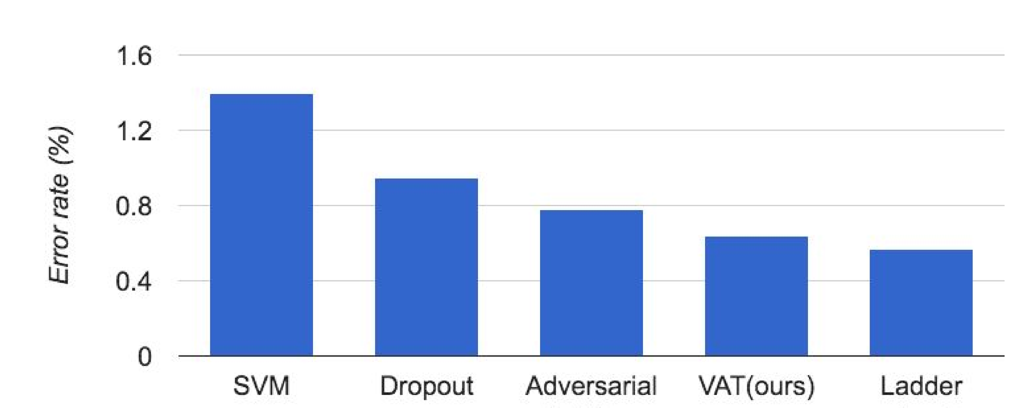
- Contour plot of $p(y=\bullet|x,\theta)$ on MLE, L2 and VAT



Supervised learning task for MNIST

- MNIST (permutation invariant task)
  - 60000 training samples
  - 784-1200-600-300-150-10NN
  - ADAM[Kingma, 2015] optimizer

- Performance results



Semi-supervised learning tasks (permutation invariant tasks)

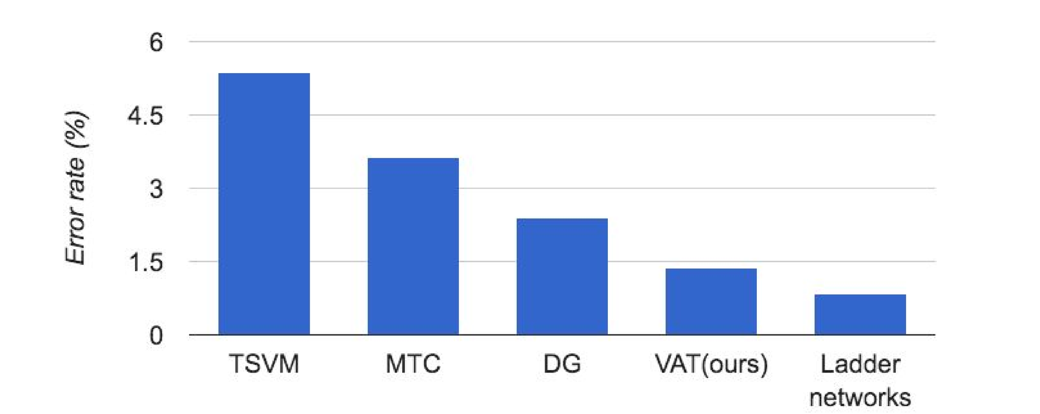
- MNIST

- SVHN (street view housing numbers)
  - 73,257 training samples (did not use "extra"set)

- NORB (NYU Object Recognition Benchmark)
  - 24,300 training samples

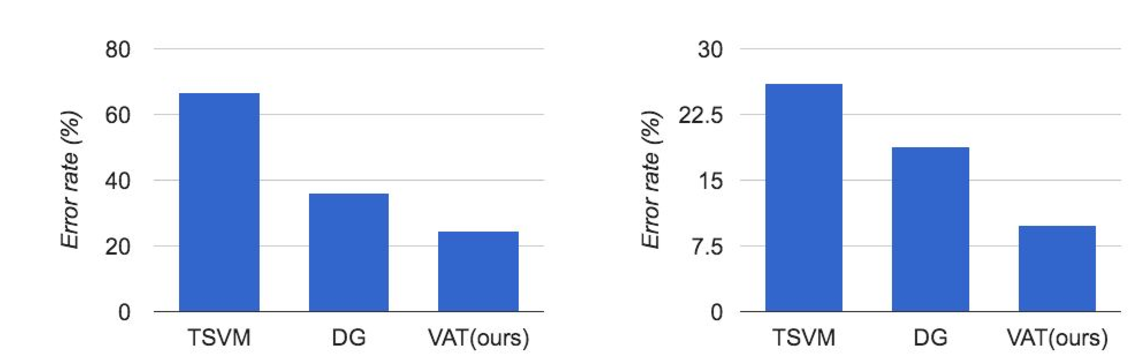
※To demonstrate the performance of semi-supervised learning, We picked up 1000 samples as labeled samples and treated the rest as unlabeled samples.

- Results on MNIST (1000 labeled samples)



TSVM: Transductive SVM, MTC: Manifold Tangent Classifier, DG:Deep Generative Model

- SVHN (1000 labeled)
- NORB (1000 labeled)



Conclusion

- Our approach was effective for supervised and semi-supervised learning for benchmark datasets.
- With **only 1 hyperparameter** ϵ , our method achieved good performance.

References

[1] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In International Conference on Learning Representation , 2015.

- Uploaded on github for reproduction of results on synthetic dataset and MNIST results. See https://github.com/takerum/vat .