Modelling adversarial training

Michiel Straat

Content

- Adversarial exa

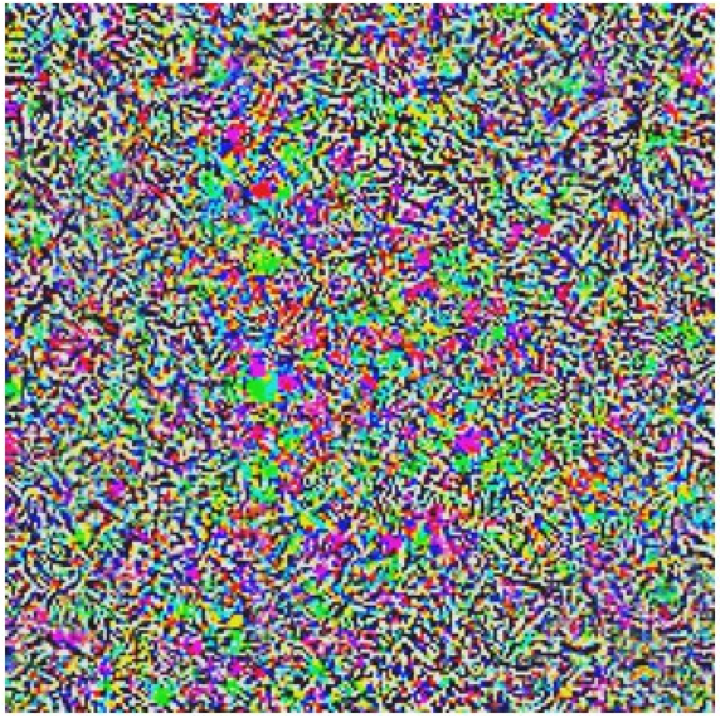
Adversarial examples

Resnet50 trained in imagenet: Kit fox with 76% confidence

Max: 2/255, min: -2/255

Kit fox with 76.0% confidence

+



=

Coyote with 99.8% confidence.
Kit fox with 2.7*10^-6% confidence.

Constructing adversarial examples

Adversarial example: tiny perturbations applied to data aimed at causing incorrect predictions.

For an example $\xi^{(\mu)} \in \mathbb{R}^N$ find small $\delta^{(\mu)} \in \Delta$ such that $\mathrm{argmax}_i \sigma(\xi + \delta)_i \neq \tau^{(\mu)}$

The perturbations should be imperceptible or not change the semantics of the data:

$$\Delta = \{\delta : ||\delta||_\infty \leq \epsilon\}$$

(each component of ξ perturbable at most by $[-\epsilon, \epsilon]$)

Apply a perturbation δ that maximizes the loss:

$$\max_{\delta \in \Delta} \mathcal{L}(\sigma_{\mathbf{w}}(\xi + \delta), \tau)$$

In practice, approximate above by:

$$\tilde{\xi} = \xi + \epsilon \cdot \mathrm{sign}(\nabla_\xi \mathcal{L}(\sigma(\xi), \tau)$$

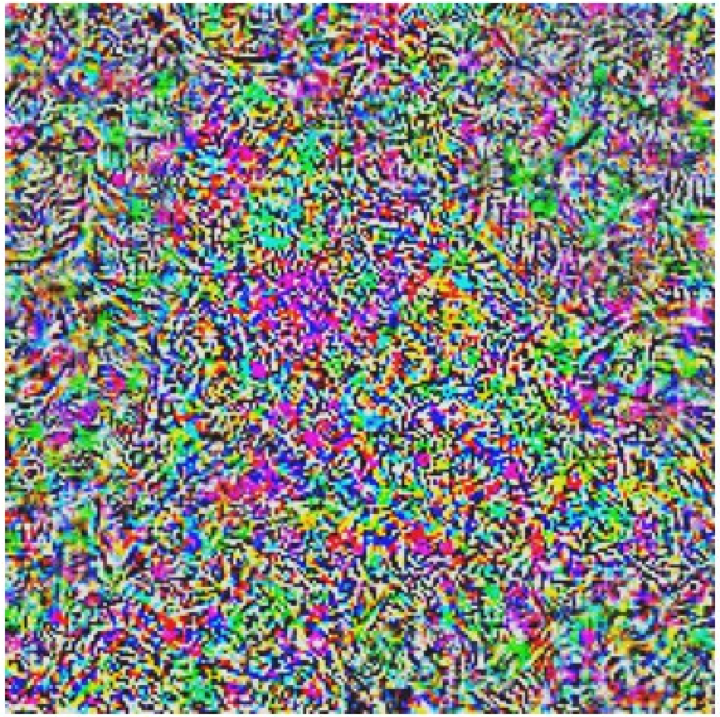
Adversarial Robustness: Theory and Practice”, Z. Kolter, A. Madry, NeurIPS 2018 tutorial

Targeted attack

$$\max_{\delta \in \Delta} \left(\sigma_{\mathbf{w}}(\xi + \delta)_{\tau_{\text{goal}}} - \sigma_{\mathbf{w}}(\xi + \delta)_{\tau} \right)$$

Max: 2/255, min: -2/255

+



=

Kit fox with 76.0% confidence

Whisky jug with 100.0% confidence

Computing adversarial perturbations

$$\max_{\delta\in\Delta} \mathcal{L}(\sigma_{\mathbf{w}}(\xi+\delta),\tau)$$
$$\Delta = \{\delta : ||\delta||_\infty \leq \epsilon\}$$

Fast Gradient Sign Method:

$$\tilde{\xi} = \xi + \epsilon \cdot \mathrm{sign}(\nabla_\xi \mathcal{L}(\sigma(\xi),\tau)$$

Basic Iterative Method:

$$\xi^{(0)} = \xi$$
$$\xi^{(t)} = \mathrm{proj}(\xi^{(t-1)} + \epsilon \cdot \mathrm{sign}(\nabla_\xi \mathcal{L}(\sigma(\xi^{(t)}),\tau)))$$

For t=[1,2...T] iterations

- If only discrete labels available:
  train surrogate model on data labeled by the target model and compute adversarial examples on that model (using above), (same data distr → similar models).

Adversarial training

Assume data, $(\xi, \tau) \in \mathbb{R}^N \times [1, 2, ..K]$ with a distribution *P.*

$\sigma : \mathbb{R}^N \to [0, 1]^K$ is a classifier.

Standard training:

$$\min_{\mathbf{w}} \mathbb{E}_{(\xi,\tau)\sim P}(\mathcal{L}(\sigma_{\mathbf{w}}(\xi), \tau))$$

Adversarial training: allow an adversary to perturb the input first:

$$\min_{\mathbf{w}} \max_{\delta \in \Delta} \mathbb{E}_{(\xi,\tau)\sim P}(\mathcal{L}(\sigma_{\mathbf{w}}(\xi + \delta), \tau)$$

"Towards Deep Learning Models Resistant to Adversarial Attacks", A. Madry et al, ICLR 2018

Assessing robustness of the model

- The *standard classification loss* of the classifier is

$$L(\sigma) = \mathbb{E}_{(\xi,\tau)\sim P}[\mathrm{argmax}_i \sigma(\xi)_i \neq \tau]$$

- The *ϵ-adversarial classification loss* counts considers in addition the non-robust examples:

$$L_\epsilon(\sigma) = \mathbb{E}_{(\xi,\tau)\sim P}[\mathrm{argmax}_i [\sigma(\xi + \delta)]_i \neq \tau]$$

- By varying ϵ in the above definition, one obtains a *monotonically increasing* adversarial loss curve.

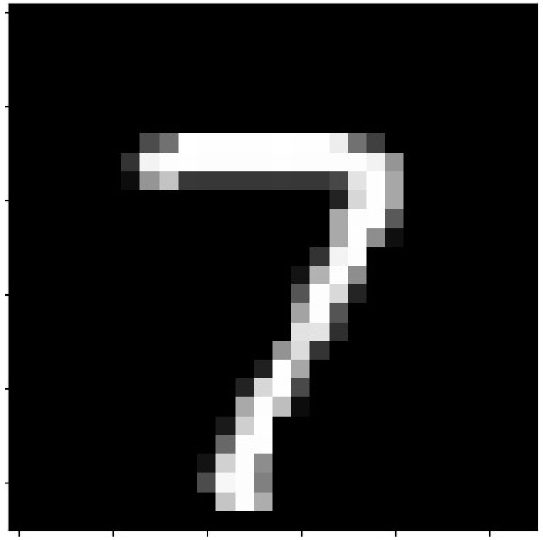
“Adversarial robustness curves”, C. Göpfert, J. Göpfert, B. Hammer, ECML/PKDD 2019

Experiment Logistic Regression MNIST

$$\sigma(\xi) = \zeta(\mathbf{w} \cdot \xi + b)$$

Class labels $\tau = [-1, 1]$

1: label 3.
-1: label is 7.



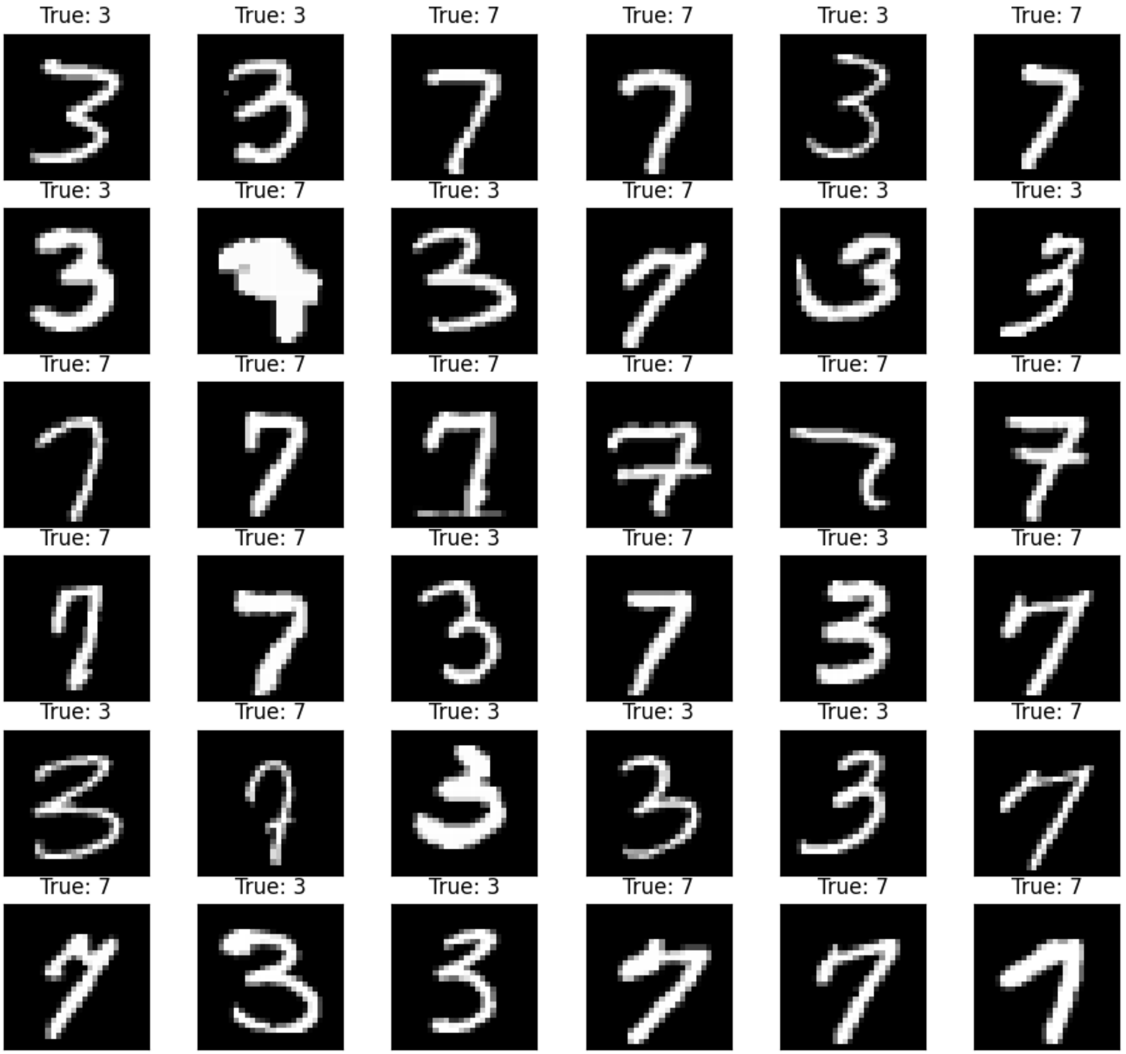
$+0.1 *$

$sign(\square)$

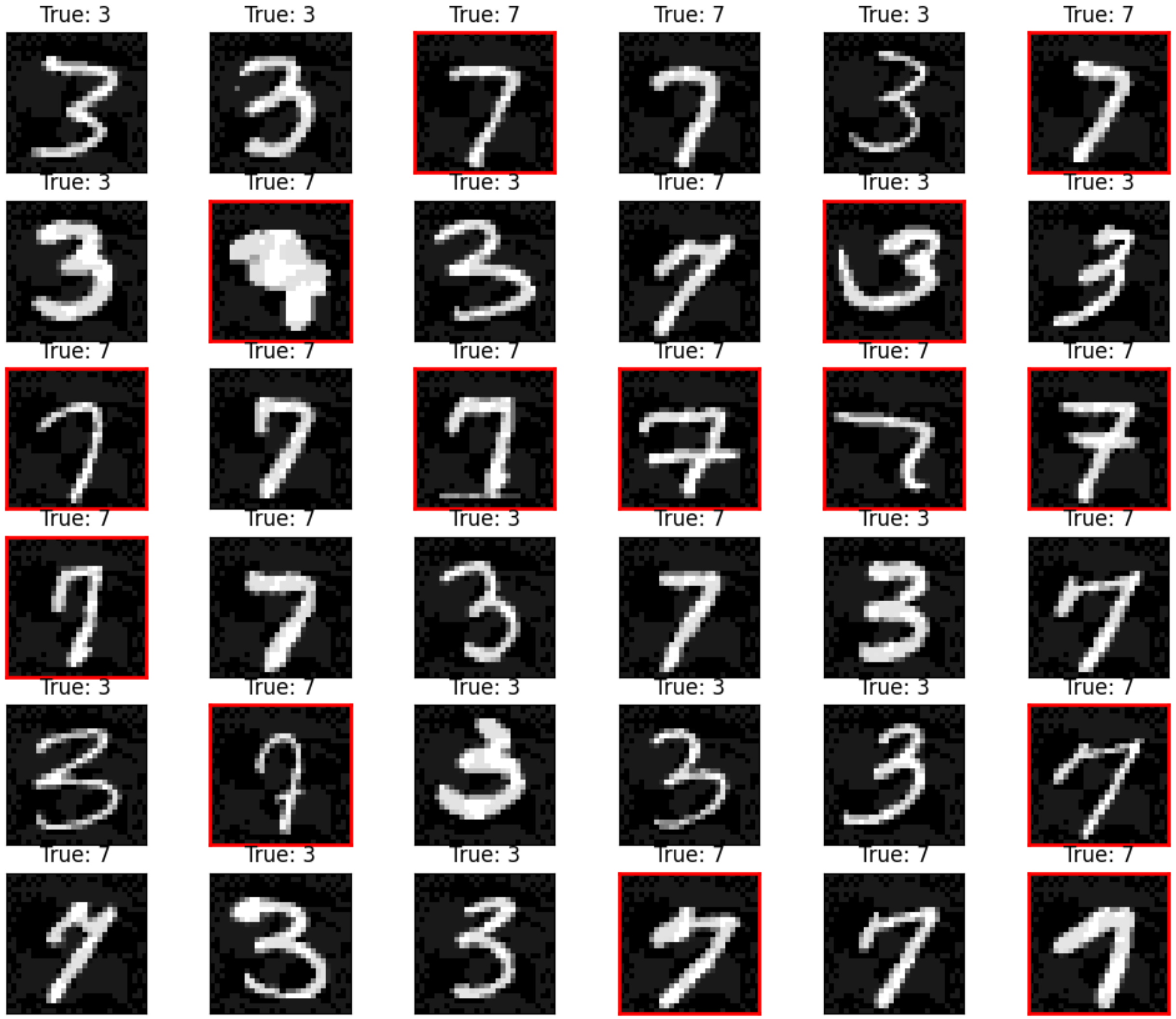
=

“Explaining and Harnessing Adversarial Examples”, I. Goodfellow et al

Classifier output on unperturbed test samples

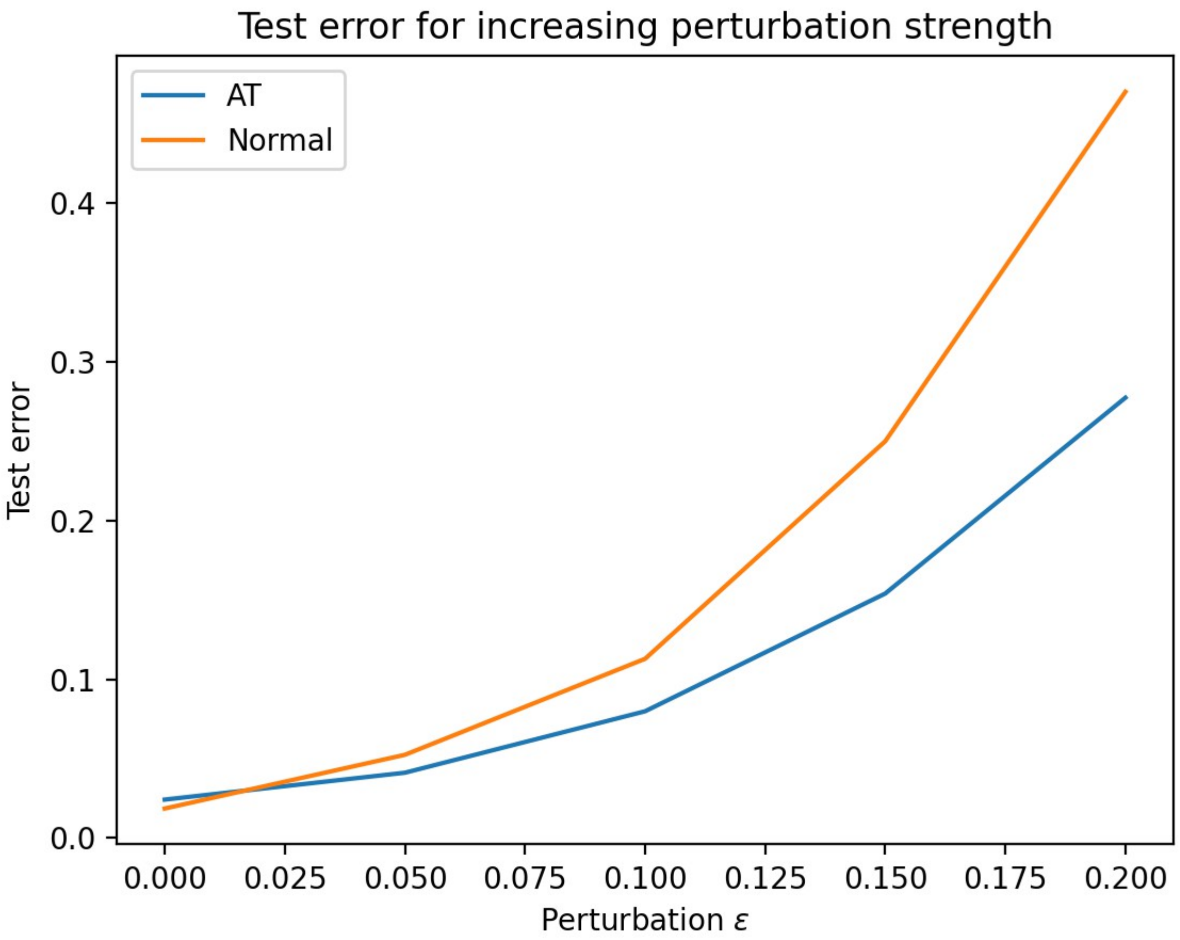


Classifier output on perturbed test samples



→ Applying the adversarial perturbation to all test samples: 40% test error.

Experiment Logistic Regression MNIST



“Explaining and Harnessing Adversarial Examples”, I. Goodfellow et al

Modelling of adversarial training: 1) the data distribution

Stream of examples $\xi \in \mathbb{R}^N$

$$P(\xi) = p\,\mathcal{N}(\xi|\mathbf{B}_1, v_1 I) + (1-p)\,\mathcal{N}(\xi|\mathbf{B}_2, v_2 I)$$

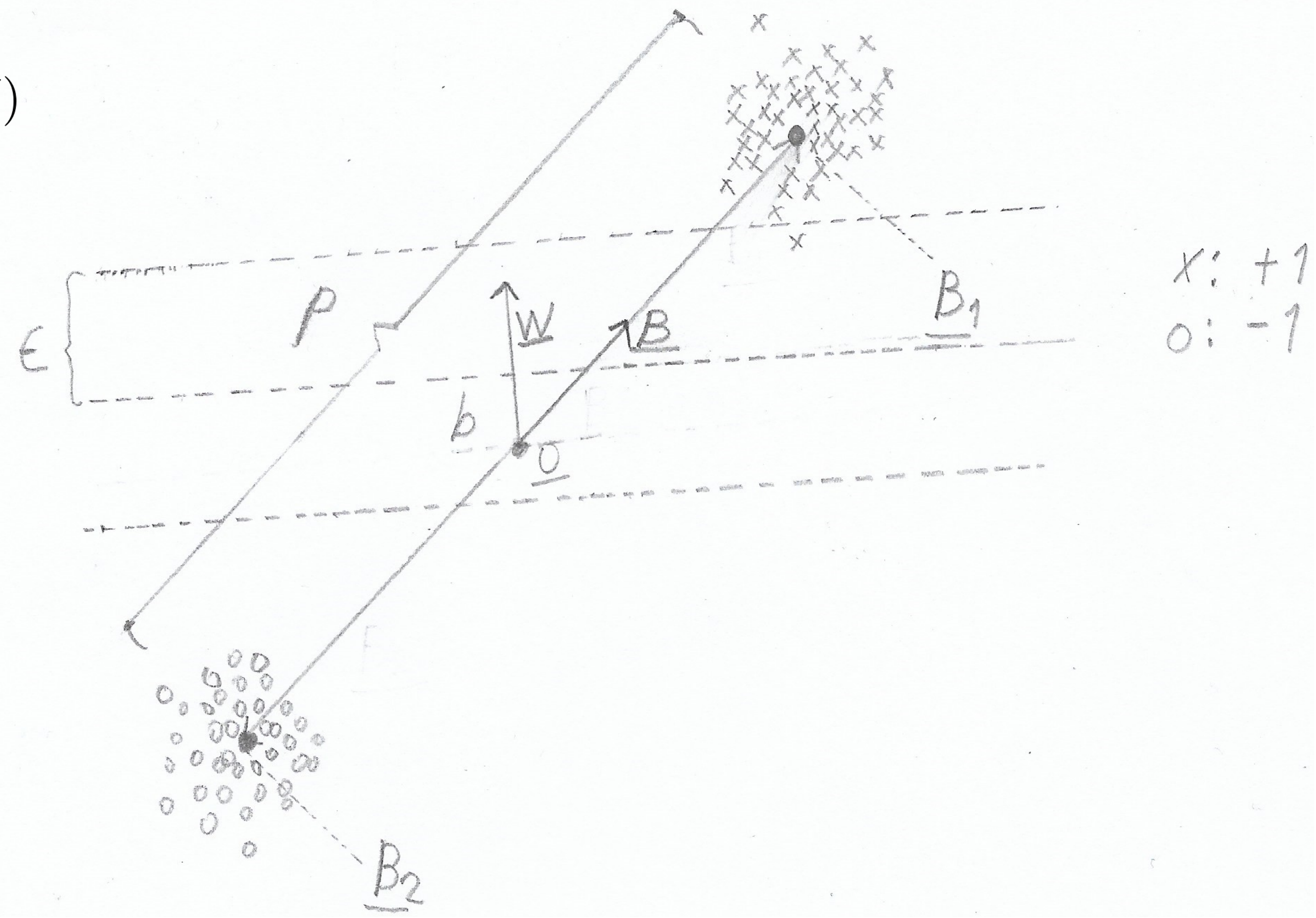
$$\mathbf{B} \in \mathbb{R}^N\,, \quad ||\mathbf{B}||_2 = 1$$

Means of the two modes:

$$\mathbf{B}_1 \;=\; \frac{1}{2}\rho\mathbf{B}$$

$$\mathbf{B}_2 \;=\; -\frac{1}{2}\rho\mathbf{B} = -\mathbf{B}_1$$

(Separation between cluster centers is ρ)

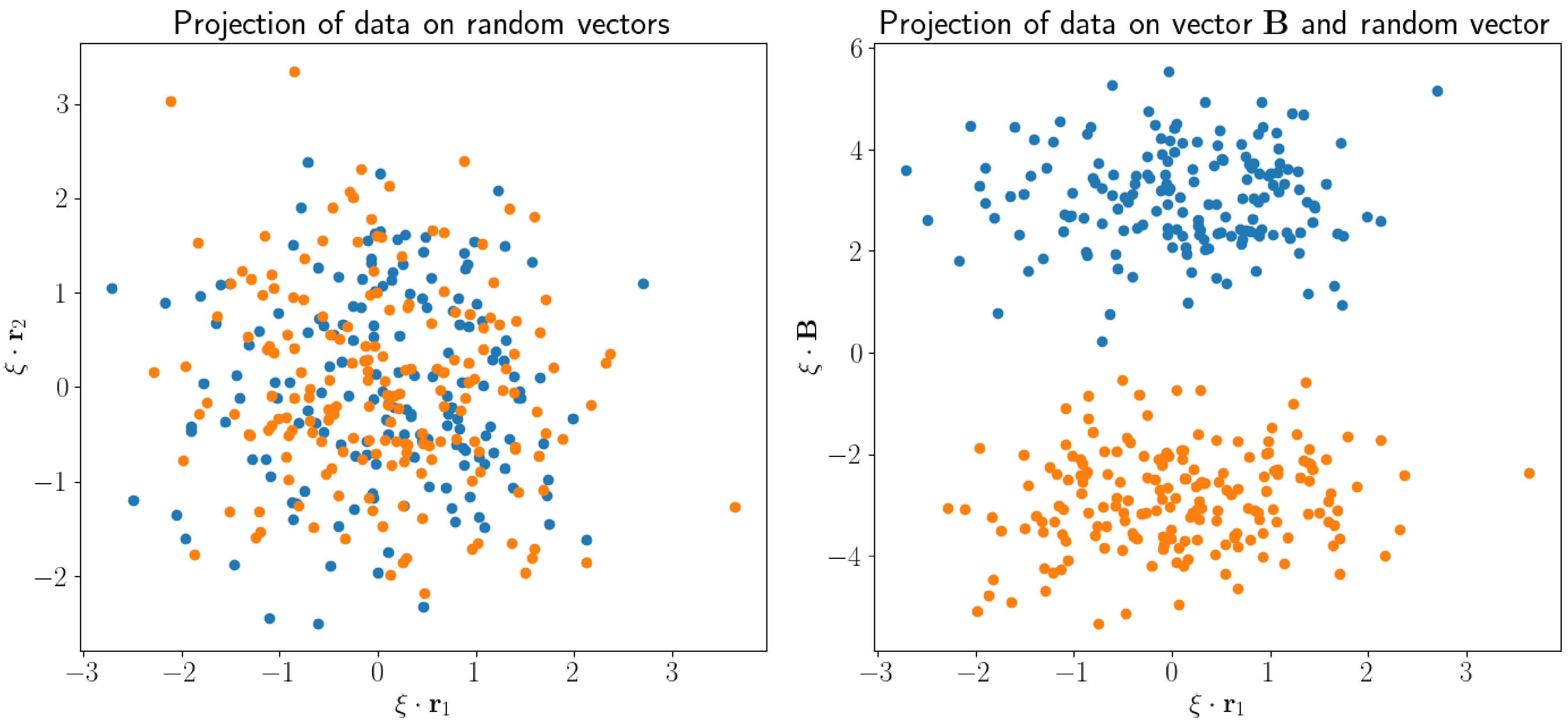


Labels:

$$\tau^{(\mu)} = \begin{cases} 1 & \text{if } \xi^\mu \text{ from } \mathcal{N}(\xi|\mathbf{B}_1, v_1 I) \\ -1 & \text{if } \xi^\mu \text{ from } \mathcal{N}(\xi|\mathbf{B}_2, v_2 I) \end{cases}$$

1) the data distribution, data in high dimension N=1000

$N = 1000,\ \rho = 6.0,\ v_1 = v_2 = 1$

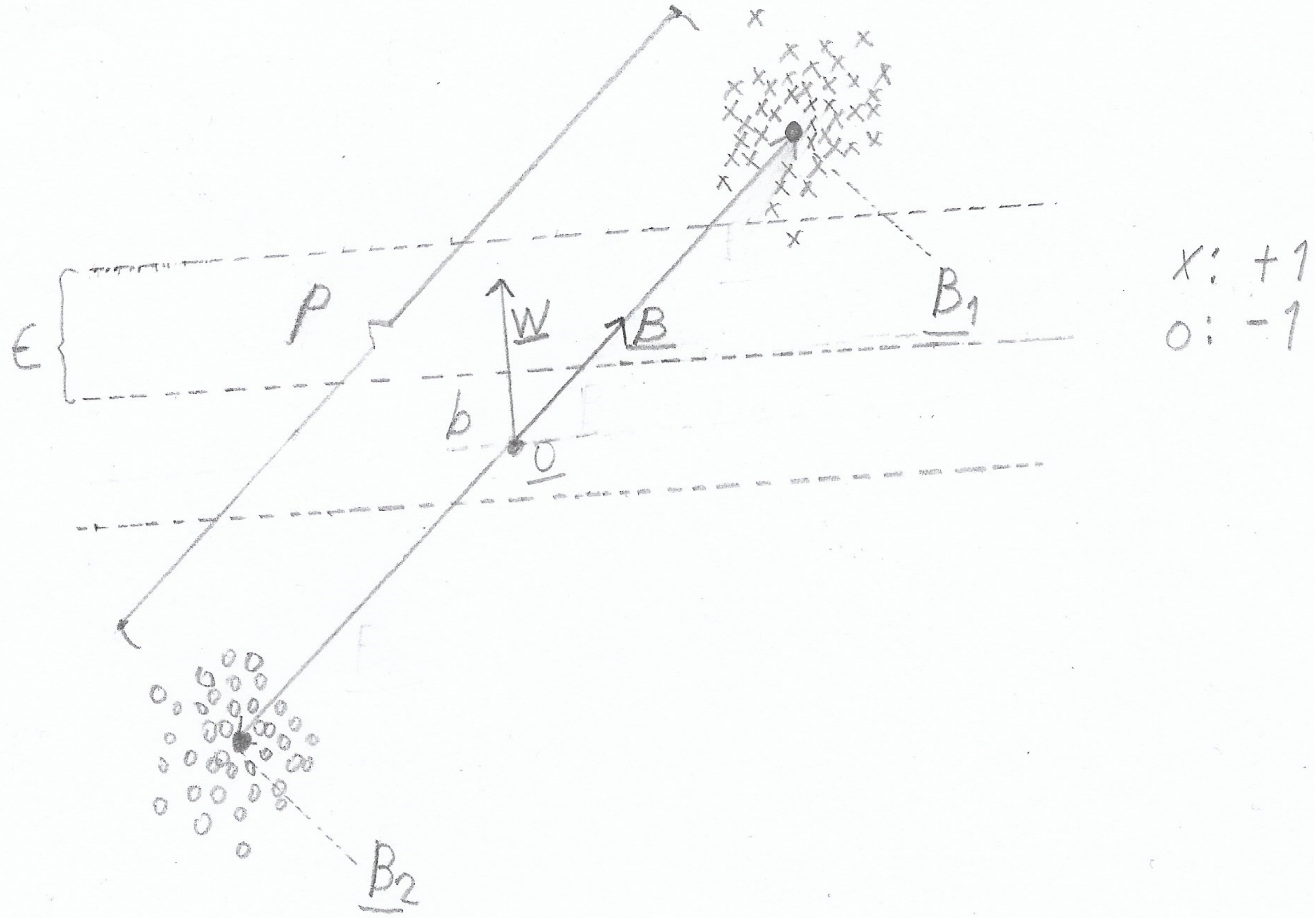


Modelling of adversarial training: 2) the machine learning model

Model with linear decision boundary, parameters are weight vector and bias:

$$\mathbf{w} \in \mathbb{R}^N \,,\, b \in \mathbb{R}$$

$$||\mathbf{w}||_2 = 1$$



Then the output of the model is

$$\sigma(\xi^{(\mu)}) = \mathrm{erf}(\mathbf{w} \cdot \xi^{(\mu)} - b)$$

Define the argument as:

$$x^{(\mu)} = \mathbf{w} \cdot \xi^{(\mu)} - b$$

(Local potential/pre-activation)

Modelling of adversarial training: 3) the learning algorithm

	Normal algorithm	Adversarial Training (AT)
Training data	$\xi^{(\mu)}$	$\tilde{\xi}^{(\mu)} = \xi^{\mu} - \tau^{(\mu)}\epsilon\mathbf{w}$

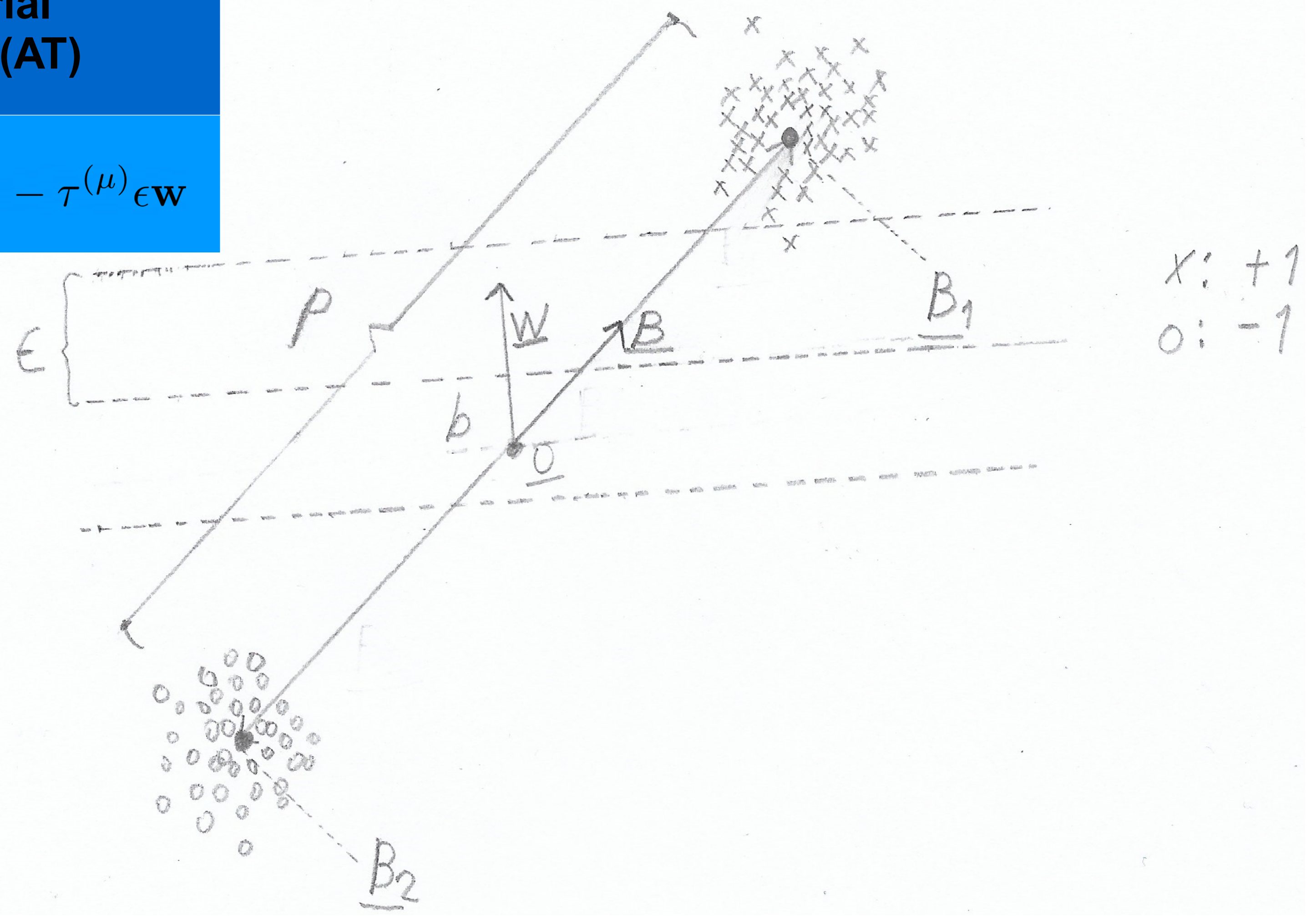
Loss:

$$\mathcal{L}(\{\mathbf{w}, b\}, \tilde{\xi}^{(\mu)}, \tau^{(\mu)}) = \frac{1}{2}(\tilde{\sigma}^{(\mu)} - \tau^{(\mu)})^2$$

Updates:

$$\mathbf{w}^{(\mu+1)} = (\mathbf{w}^{(\mu)} - \frac{\eta_{\mathbf{w}}}{N}\nabla_{\mathbf{w}}\mathcal{L})/||\cdot||$$

$$b^{(\mu+1)} = b^{(\mu)} - \frac{\eta_b}{N}\nabla_b\mathcal{L}$$



Modelling of adversarial training: Order parameters

$$\tilde{\xi}^{(\mu)} = \xi^{\mu} - \tau^{(\mu)}\epsilon\mathbf{w}$$

$$\tilde{\sigma}^{(\mu)} = \mathrm{erf}(\tilde{x}^{(\mu)})$$

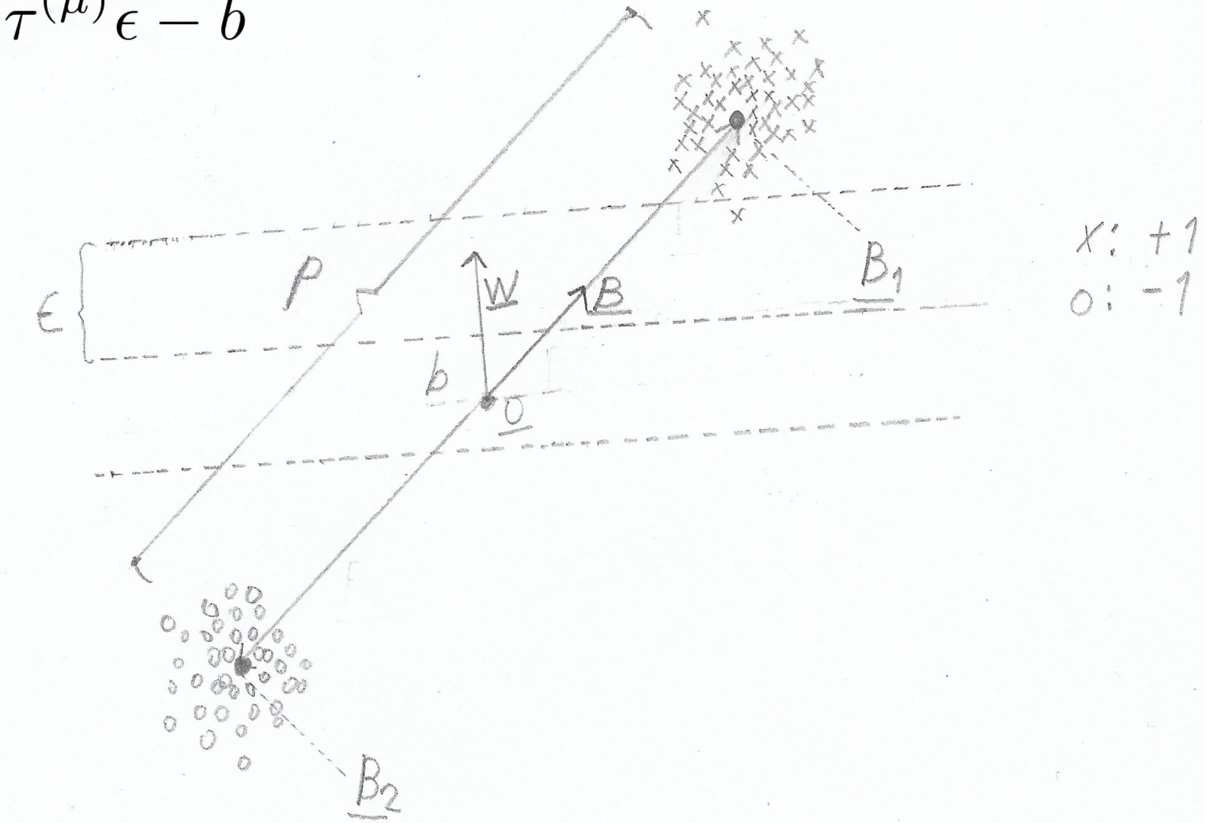
with pre-activation: $\tilde{x}^{(\mu)} = \mathbf{w}\cdot\tilde{\xi}^{(\mu)} - b = \mathbf{w}\cdot\xi^{(\mu)} - \tau^{(\mu)}\epsilon - b$

$\tilde{x}$ is distributed as a Gaussian (in this case also for small N)

It's conditional mean and variance are:

$$\langle\tilde{x}\rangle_1 = \frac{1}{2}\rho\underbrace{\mathbf{w}\cdot\mathbf{B}}_{R} - b - \epsilon\,, \qquad Var[\tilde{x}]_1 = v_1$$

$$\langle\tilde{x}\rangle_2 = -\frac{1}{2}\rho R - b + \epsilon, \qquad Var[\tilde{x}]_2 = v_2$$



R and b are *order parameters* of the system.

Modelling of adversarial training: Dynamics of the order parameters

$$R = \mathbf{w} \cdot \mathbf{B}$$

$$\mathbf{w}^{(\mu+1)} = (\mathbf{w}^{(\mu)} - \frac{\eta_{\mathbf{w}}}{N} \nabla_{\mathbf{w}} \mathcal{L}) / || \cdot ||$$

$$b^{(\mu+1)} = b^{(\mu)} - \frac{\eta_b}{N} \nabla_b \mathcal{L}$$

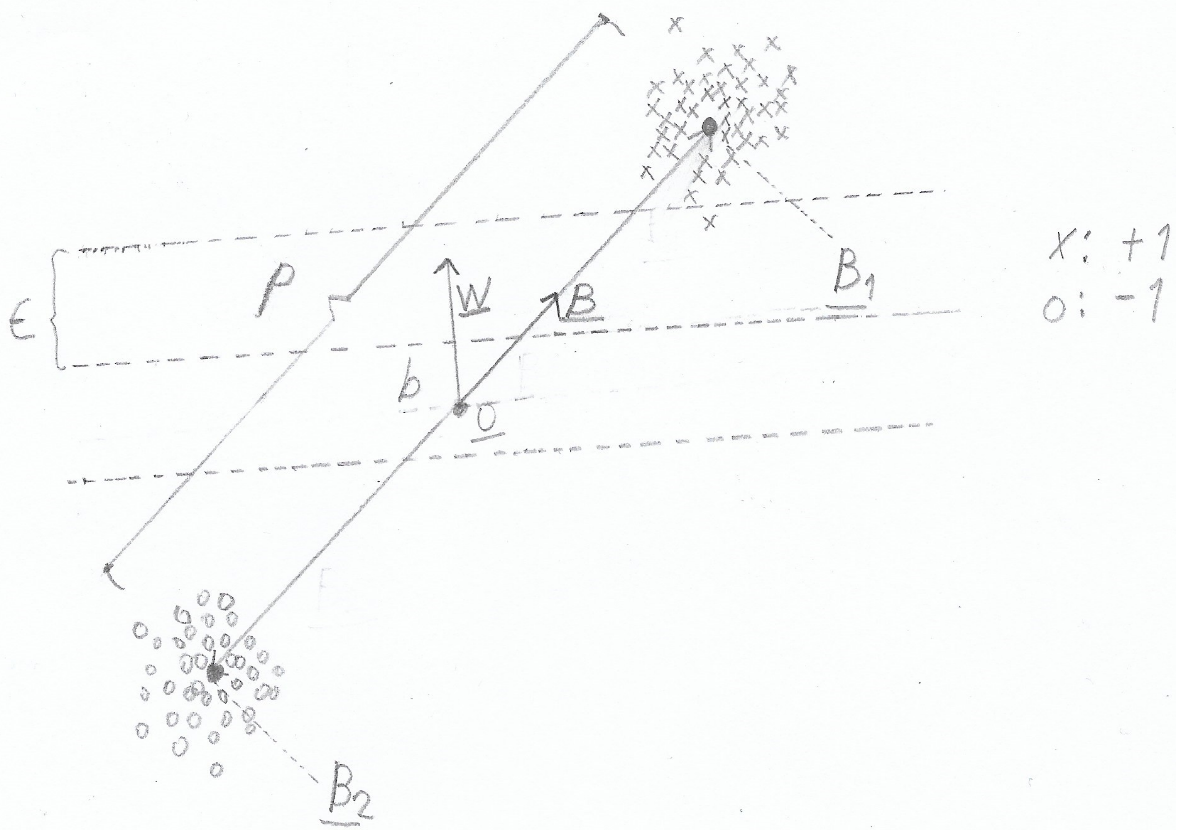
$$R^{(\mu+1)} - R^{(\mu)} = \frac{\eta_{\mathbf{w}}}{N} f(\tilde{x}, \tilde{y})$$

$$b^{(\mu+1)} - b^{(\mu)} = \frac{\eta_b}{N} h(\tilde{x})$$

Self-averages for large N and many example presentations for the normalized learning time $\alpha = \mu/N$, write ODEs:

$$\frac{\partial R}{\partial \alpha} = \eta_{\mathbf{w}} \langle f(\tilde{x}, \tilde{y}) \rangle_{P(\tilde{x}, \tilde{y})}$$

$$\frac{\partial b}{\partial \alpha} = \eta_b \langle h(\tilde{x}) \rangle_{P(\tilde{x})}$$



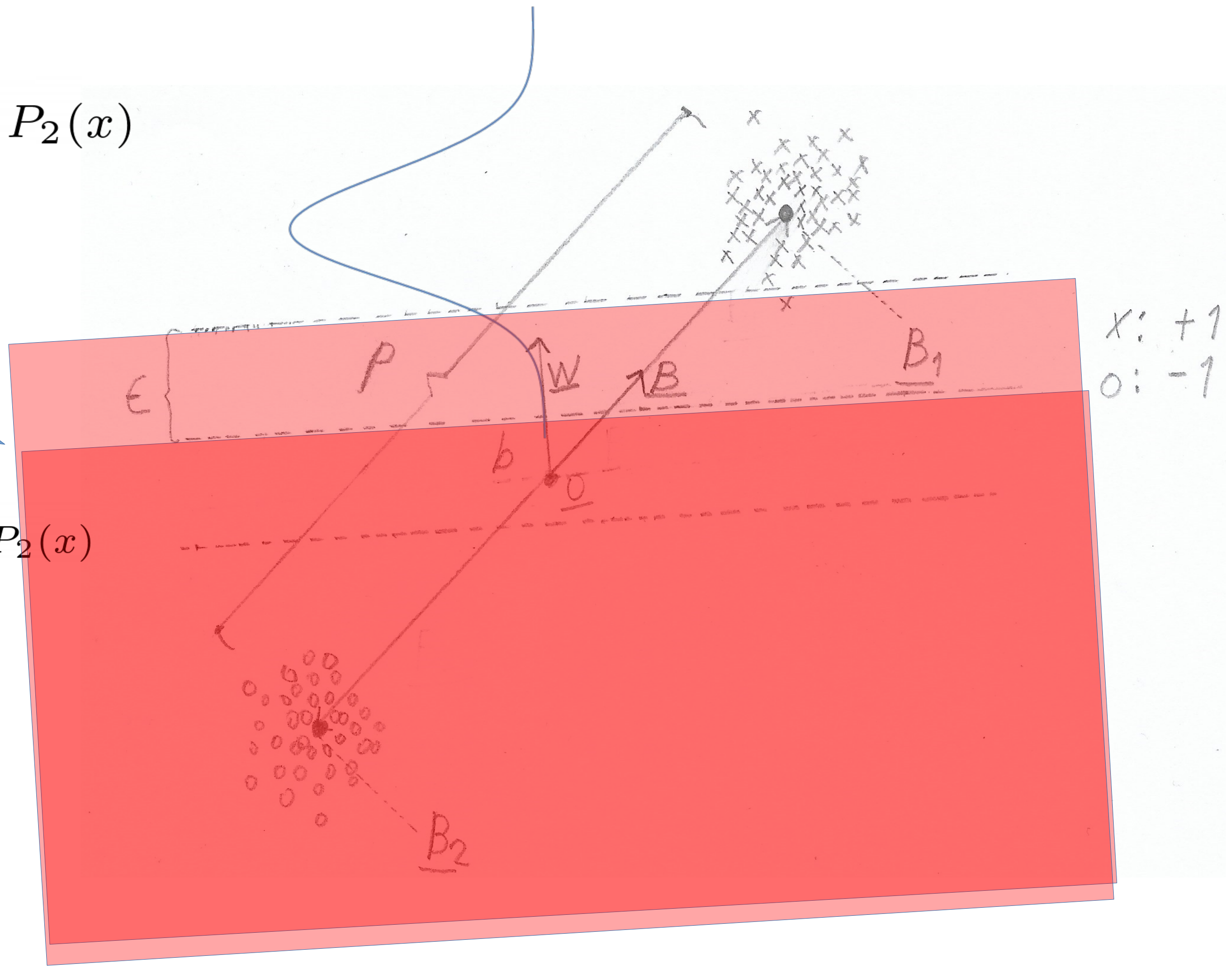
Modelling of adversarial training: Evaluation measures

Test classification error:

$$\mathcal{E}(R,b) = p\langle\underline{\Theta(-x)}\rangle_{P_1(x)} + (1-p)\langle\Theta(x)\rangle_{P_2(x)}$$

Adversarial classification error:

$$S(R,b,\epsilon) = p\langle\underline{\Theta(\epsilon-x)}\rangle_{P_1(x)} + (1-p)\langle\Theta(x+\epsilon)\rangle_{P_2(x)}$$



Obtain both curves by substitution of R(α), b(α):

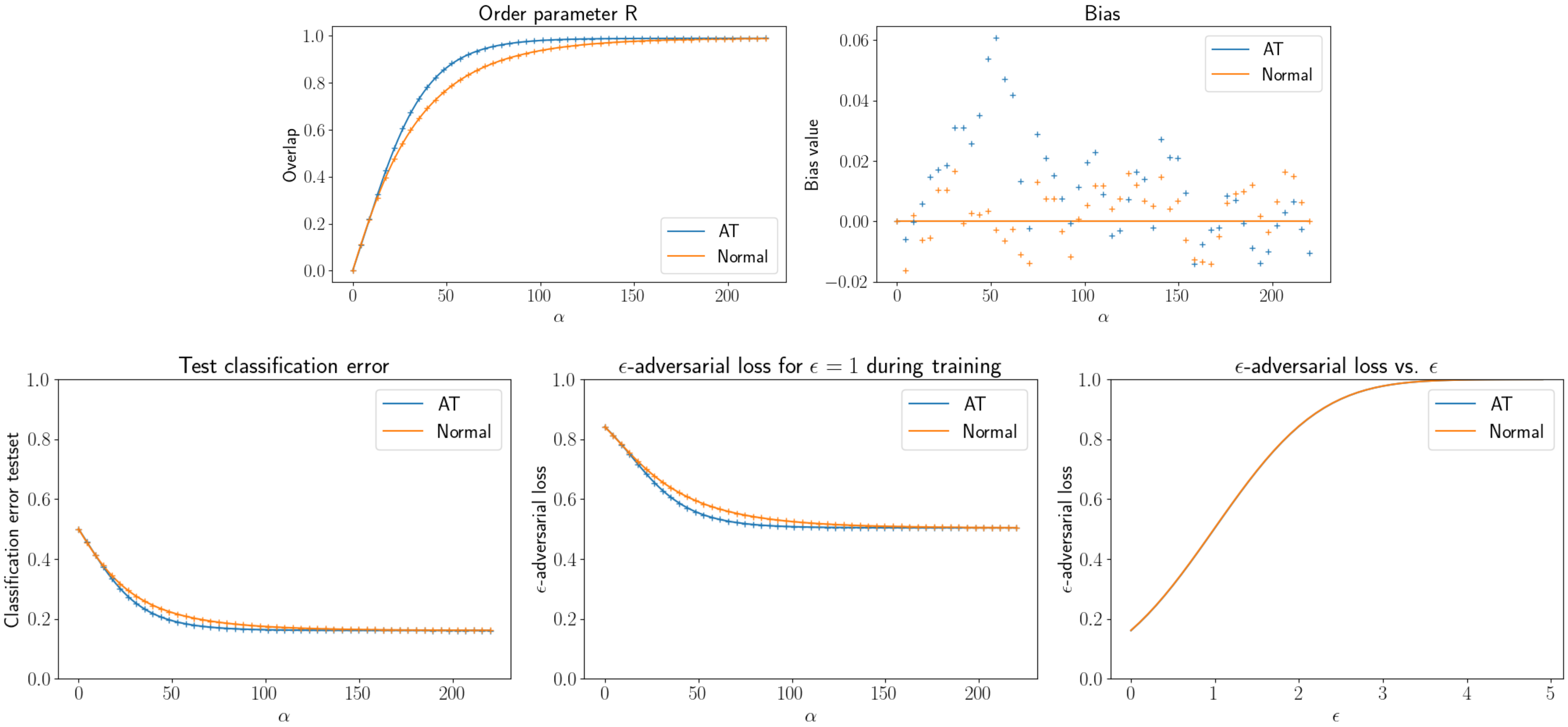
$$\mathcal{E}(R(\alpha), b(\alpha)), \quad S(R(\alpha), b(\alpha), \epsilon)$$

Obtain final robustness vs. a range of epsilon:

$$S(R(\alpha_{\max}), b(\alpha_{\max}), \epsilon = [0 \ldots \epsilon_{\max}])$$

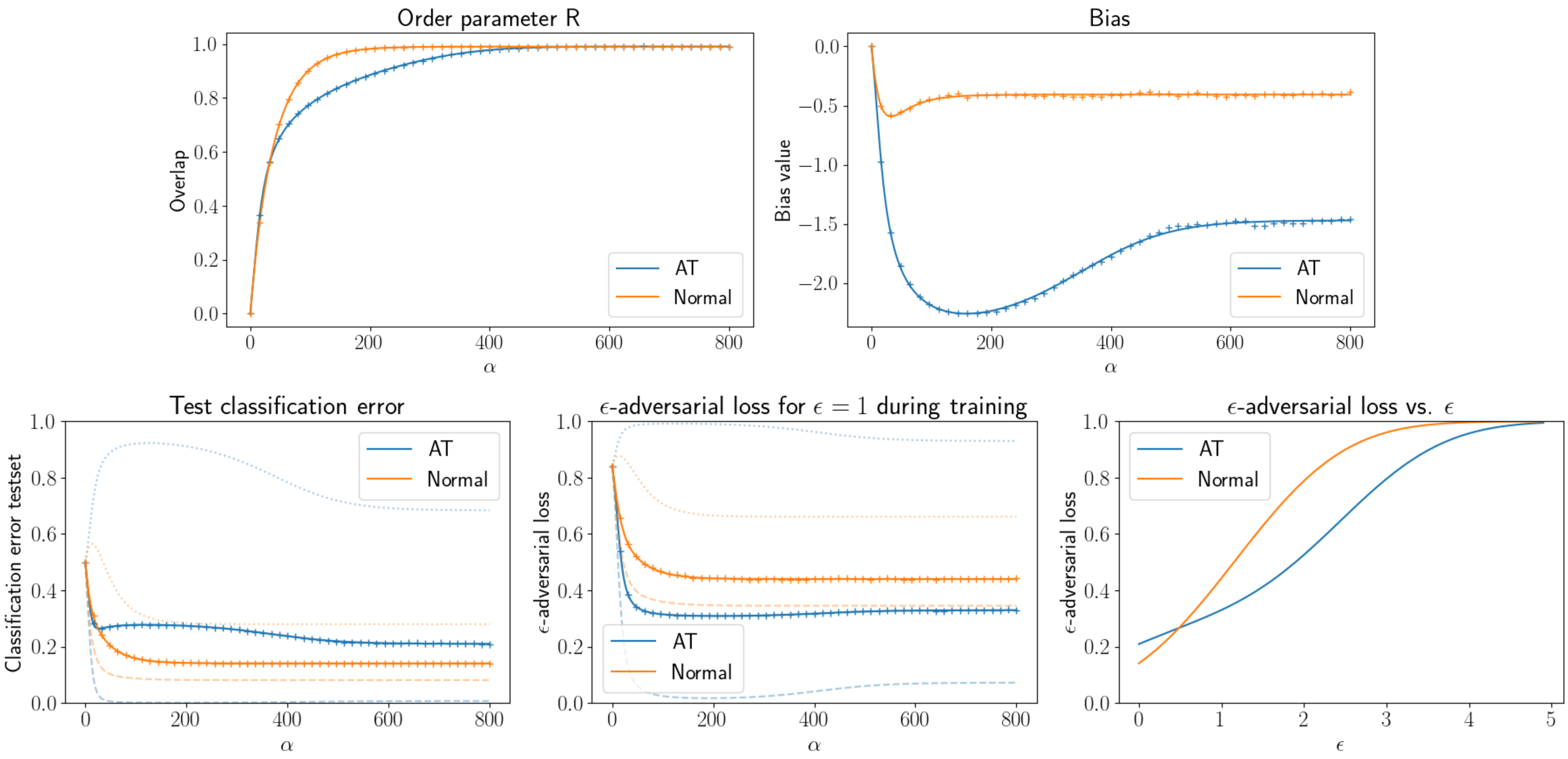
Results – Scenario 1

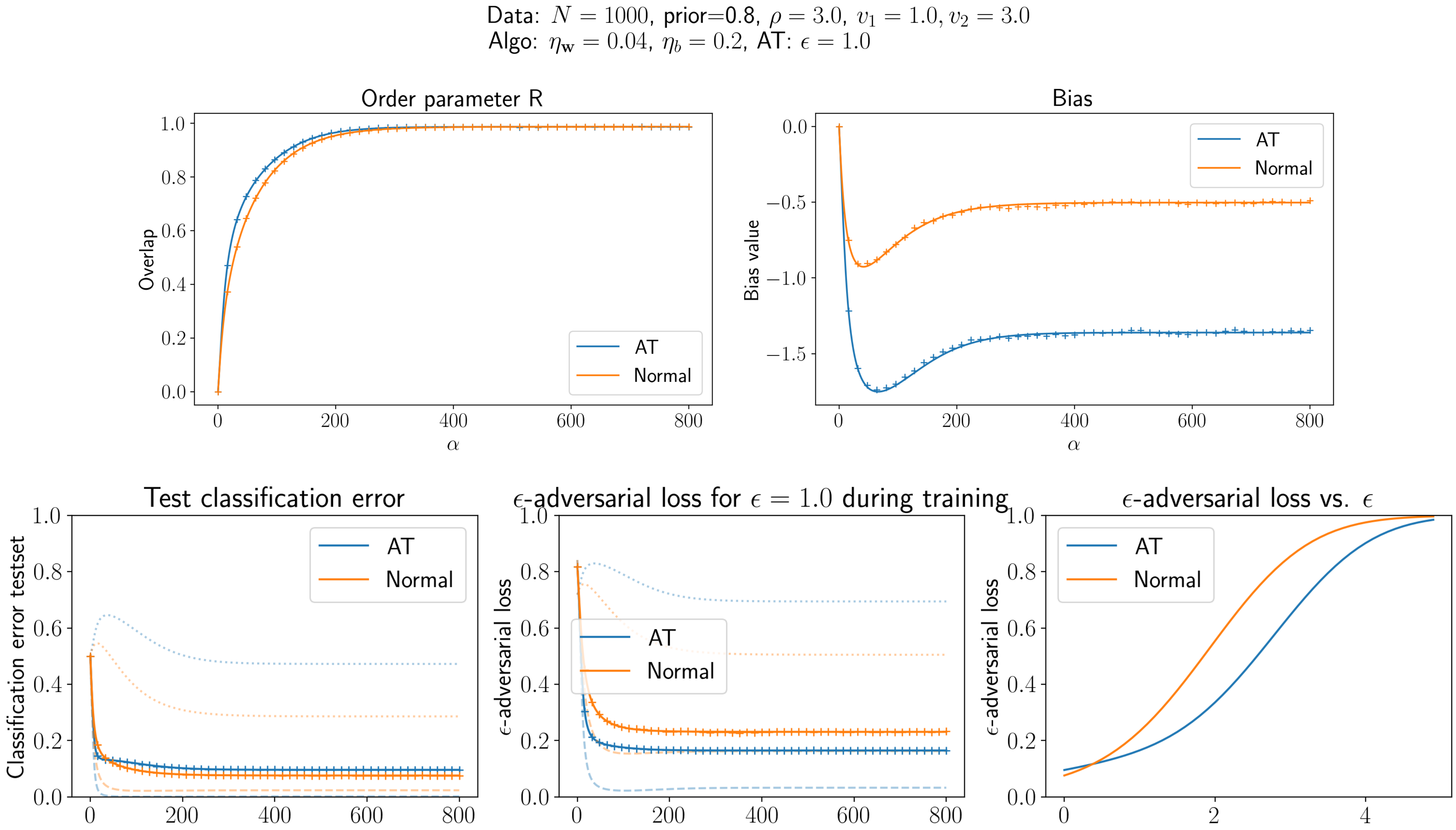
Data: $N = 1000$, prior=0.5, $\rho = 2.0$, $v_1 = v_2 = 1.0$
Algo: $\eta_{\mathbf{w}} = 0.04$, $\eta_b = 0.2$, AT: $\epsilon = 1$



Results – Scenario 2

Data: $N = 1000$, prior=0.7, $\rho = 2.0$, $v_1 = v_2 = 1.0$
Algo: $\eta_{\mathbf{w}} = 0.04$, $\eta_b = 0.2$, AT: $\epsilon = 1$





Data: $N = 1000$, prior=0.8, $\rho = 3.0$, $v_1 = 1.0$, $v_2 = 3.0$
Algo: $\eta_{\mathbf{w}} = 0.04$, $\eta_b = 0.2$, AT: $\epsilon = 1.0$
Order parameter R
Overlap
α
Bias
Bias value
α
AT
Normal
Test classification error
Classification error testset
ϵ-adversarial loss for $\epsilon = 1.0$ during training
ϵ-adversarial loss
ϵ-adversarial loss vs. ϵ
ϵ-adversarial loss

Outlook

- Extension to two layer non-linear neural networks, better model scenarios.
- Analyse variants of adversarial training, randomized epsilon, training only on non-robust examples (hinge loss).