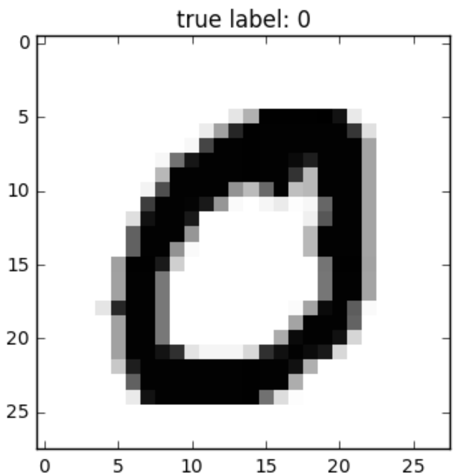
Deep Mahalanobis Detector

2020. 02. 06
Soonwoo Kwon
soonwoo.kwon@kaist.ac.kr

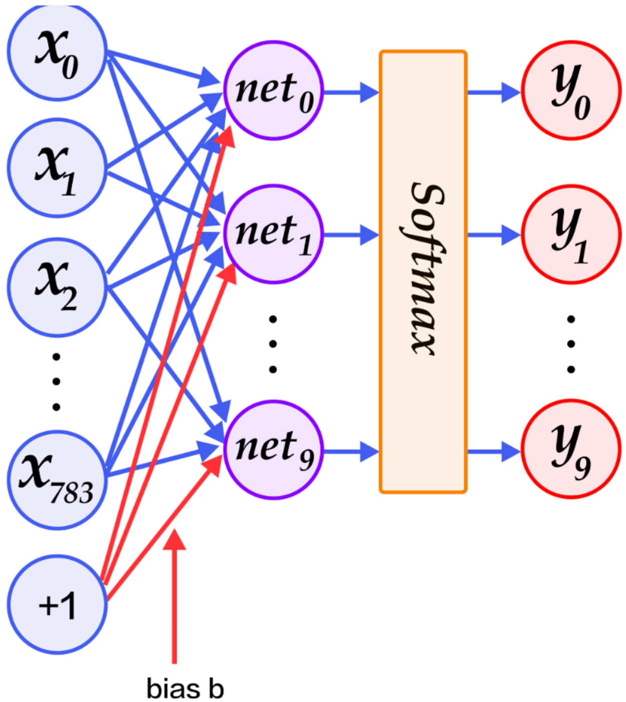
Table of Contents

Table of Contents

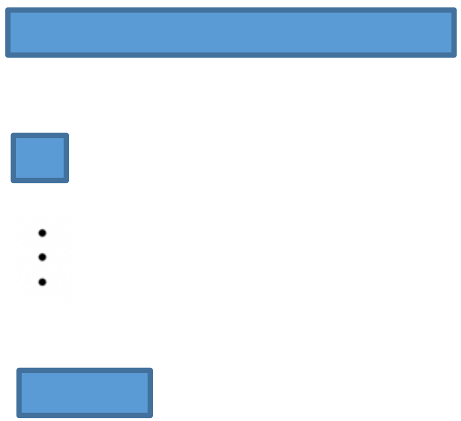
Motivation



Test Image
(MNIST)

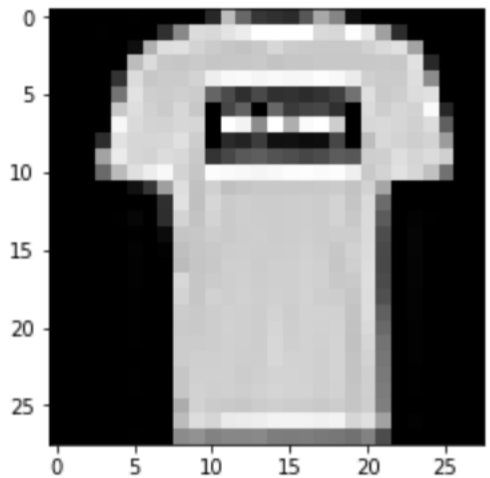


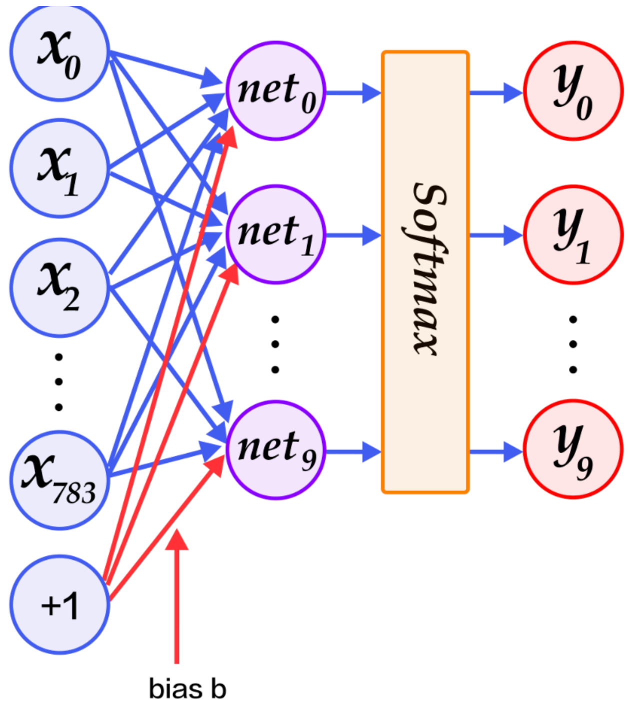
Model
(Trained on MNIST)

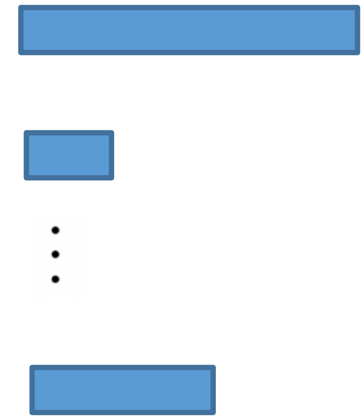


Output
(Probability)

Motivation







Test Image
(Fashion MNIST)

Model
(Trained in MNIST)

Output
(Probability)

Still our model makes confident prediction..!

MLILAB Machine Learning & Intelligence

OOD(Out-Of-Distribution)?

- **In-distribution**: distribution of training samples
- **(OOD) Out-of-distribution**

Table of Contents

Deep Mahalanobis Detector

- can use pre-trained model without any modification in test time
- is not only **simple** but also **shows good performance on OOD detection**
- can **detect** both **OOD samples** and **adversarial attack samples**
- can be extended to be utilized in class-incremental learning

Method

- Pretrained Softmax classifier:

$$P(y=c|\mathbf{x}) = \frac{\exp\left(\mathbf{w}_c^\top f(\mathbf{x}) + b_c\right)}{\sum_{c'} \exp\left(\mathbf{w}_{c'}^\top f(\mathbf{x}) + b_{c'}\right)}$$

(where, X: input, c: class, f(.): penultimate layer of DNN)

- To get generative classifier,
  - **Suppose that class conditional distribution follows the multivariate Gaussian distribution**.

$$P\left(f(\mathbf{x})|y=c\right) = \mathcal{N}\left(f(\mathbf{x})|\mu_c, \mathbf{\Sigma}\right)$$

empiricial mean

$$\widehat{\mu}_c = \frac{1}{N_c} \sum_{i:y_i=c} f(\mathbf{x}_i),$$

empiricial (**tied**) covariance

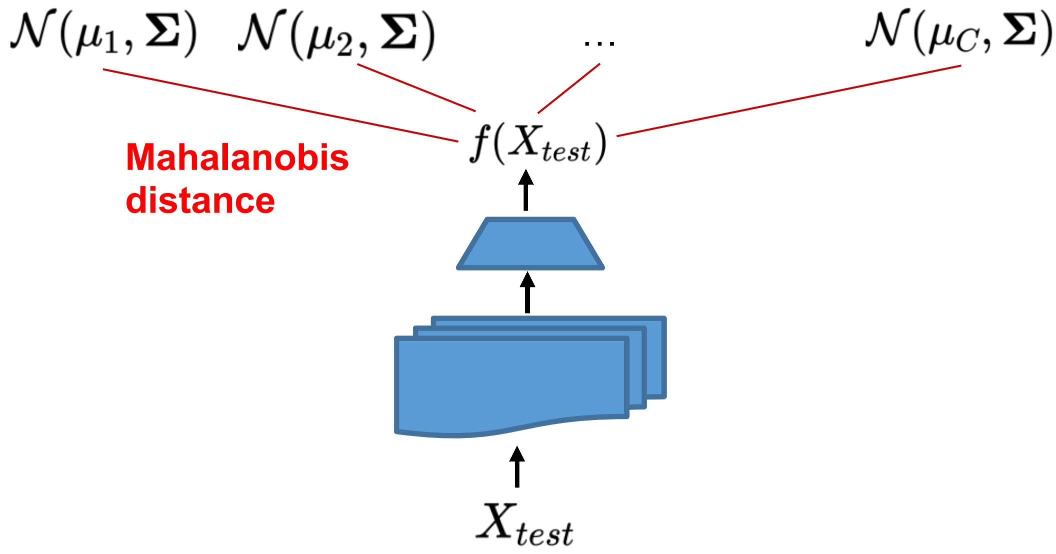
$$\widehat{\mathbf{\Sigma}} = \frac{1}{N} \sum_c \sum_{i:y_i=c} \left(f(\mathbf{x}_i) - \widehat{\mu}_c\right)\left(f(\mathbf{x}_i) - \widehat{\mu}_c\right)^\top$$

Method

- **Class conditional distribution** follows the **multivariate Gaussian distribution**.

$$P\left(f(\mathbf{x})|y=c\right) = \mathcal{N}\left(f(\mathbf{x})|\mu_c, \mathbf{\Sigma}\right)$$

- Then, calculate **Mahalanobis distance** from $f(X_{test})$ to $\mathcal{N}(\mu_c, \mathbf{\Sigma})$, $c \in \{1, \ldots, C\}$



- Confidence score: $M(\mathbf{x}) = \max_c \; -\left(f(\mathbf{x}) - \widehat{\mu}_c\right)^\top \widehat{\mathbf{\Sigma}}^{-1}\left(f(\mathbf{x}) - \widehat{\mu}_c\right)$
- If confidence score of X_{test} **<** threshold : X_{test} is predicted as **OOD**
- Classification: $\widehat{y}(\mathbf{x}) = \arg\min_c \left(f(\mathbf{x}) - \widehat{\mu}_c\right)^\top \widehat{\mathbf{\Sigma}}^{-1}\left(f(\mathbf{x}) - \widehat{\mu}_c\right)$

Method

- Input pre-processing

$$\widehat{\mathbf{x}} = \mathbf{x} + \varepsilon\,\mathrm{sign}(\underline{\nabla_{\mathbf{x}} M(\mathrm{x})}) = \mathbf{x} - \varepsilon\,\mathrm{sign}\Big(\nabla_{\mathbf{x}} (f(\mathbf{x}) - \widehat{\mu}_{\hat{c}})^{\top} \widehat{\mathbf{\Sigma}}^{-1} (f(\mathbf{x}) - \widehat{\mu}_{\hat{c}})\Big)$$

Noise is generated to **increase** the proposed **confidence score**

- Feature ensemble
  - Measuring and combining the confidence scores from not only the final features, but also the other low-level features in DNNs

Method

Algorithm 1 Computing the Mahalanobis distance-based confidence score.

Input: Test sample $\mathbf{x}$, weights of logistic regression detector α_ℓ, noise ε and parameters of Gaussian distributions $\{\widehat{\mu}_{\ell,c}, \widehat{\mathbf{\Sigma}}_\ell : \forall \ell, c\}$

Initialize score vectors: $\mathbf{M}(\mathbf{x}) = [M_\ell : \forall \ell]$
for each layer $\ell \in 1, \ldots, L$ **do**
    Find the closest class: $\widehat{c} = \arg\min_c \ (f_\ell(\mathbf{x}) - \widehat{\mu}_{\ell,c})^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} (f_\ell(\mathbf{x}) - \widehat{\mu}_{\ell,c})$
    Add small noise to test sample: $\widehat{\mathbf{x}} = \mathbf{x} - \varepsilon \text{sign}\left(\bigtriangledown_{\mathbf{x}} \left(f_\ell(\mathbf{x}) - \widehat{\mu}_{\ell,\widehat{c}}\right)^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} \left(f_\ell(\mathbf{x}) - \widehat{\mu}_{\ell,\widehat{c}}\right)\right)$
    Computing confidence score: $M_\ell = \max\limits_c - \left(f_\ell(\widehat{\mathbf{x}}) - \widehat{\mu}_{\ell,c}\right)^\top \widehat{\mathbf{\Sigma}}_\ell^{-1} \left(f_\ell(\widehat{\mathbf{x}}) - \widehat{\mu}_{\ell,c}\right)$
end for
return Confidence score for test sample $\sum_\ell \alpha_\ell M_\ell$

Experimental Result

Method	Feature ensemble	Input pre-processing	TNR at TPR 95%	AUROC	Detection accuracy	AUPR in	AUPR out
Baseline [13]	-	-	32.47	89.88	85.06	85.40	93.96
ODIN [21]	-	-	86.55	96.65	91.08	92.54	98.52
Mahalanobis (ours)	-	-	54.51	93.92	89.13	91.56	95.95
	-	✓	92.26	98.30	93.72	96.01	99.28
	✓	-	91.45	98.37	93.55	96.43	99.35
	✓	✓	**96.42**	**99.14**	**95.75**	**98.26**	**99.60**

Table 1: Contribution of each proposed method on distinguishing in- and out-of-distribution test set data. We measure the detection performance using ResNet trained on CIFAR-10, when SVHN dataset is used as OOD. All values are percentages and the best results are indicated in bold.

MLILAB Machine Learning & Intelligence

TODO

- Setting:
  - In-distribution: CIFAR-10 / Out-of-distribution: SVHN
  - Model: ResNet34 (already trained on CIFAR-10)
  - https://github.com/sooonwoo/Deep_Mahalanobis_Detector
- TODO1:
  - Calculate **TNR at TPR 95% (*as Table1)***
    - Implement **ood_test_mahalanobis()** function in 'test.py'
    - (optional) Input pre-processing & feature ensemble
    - (optional) Instead of using a tied covariance, give different covariance to each class conditional distribution
    - (optional) In-distribution: CIFAR-10 (5 classes) / out-distribution: CIFAR-10 (5classes)
- TODO2:
  - Calculate **test accuracy of CIFAR-10 test set** by using **Mahalanobis classification method (p.10)**
    - Implement **id_classification_test()** function in 'test.py'

Appendix: Method

- Discriminative classifier
  - Directly define **posterior** probability distribution P(y|x)
  - Ex) Softmax classifier

$$P(y=c|\mathbf{x}) = \frac{\exp\left(\mathbf{w}_c^\top \mathbf{x} + b_c\right)}{\sum_{c'} \exp\left(\mathbf{w}_{c'}^\top \mathbf{x} + b_{c'}\right)}$$

- Generative classifier
  - Define **class conditional** prob distribution P (x|y) and **class prior** P (y)
  - GDA (Gaussian Discriminative Analysis)
    - Class conditional: M. Gaussian $P(\mathbf{x}|y=c) = \mathcal{N}(\mathbf{x}|\mu_c, \mathbf{\Sigma}_c)$
    - Prior: Bernoulli $P(y=c) = \frac{\beta_c}{\sum_{c'} \beta_{c'}}$

$$\left.\begin{array}{l} P(\mathbf{x}|y=c) = \mathcal{N}(\mathbf{x}|\mu_c, \mathbf{\Sigma}_c) \\ P(y=c) = \frac{\beta_c}{\sum_{c'} \beta_{c'}} \end{array}\right\} P(\mathbf{x}, y) = P(y)P(\mathbf{x}|y)$$

  - LDA(Linear Discriminant Analysis): $\mathbf{\Sigma}_c = \mathbf{\Sigma}$

$$P(y=c|\mathbf{x}) = \frac{P(y=c)P(\mathbf{x}|y=c)}{\sum_{c'} P(y=c')P(\mathbf{x}|y=c')} = \boxed{\frac{\exp\left(\mu_c^\top \mathbf{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\mu_c^\top \mathbf{\Sigma}^{-1}\mu_c + \log \beta_c\right)}{\sum_{c'} \exp\left(\mu_{c'}^\top \mathbf{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\mu_{c'}^\top \mathbf{\Sigma}^{-1}\mu_{c'} + \log \beta_{c'}\right)}}$$

Softmax!

- **x might be fitted in Gaussian distribution during training a softmax classifier.**

Appendix

- Cf. TPR, TPN

Pred \ Real	P	N
P	TP	FP
N	FN	TN

- P: in-distribution (CIFAR-10)
- N: Out-of-distribution (SVHN)

- TPR = TP / (TP +FN)
- TPN = TN / (FP + TN)