Novelty Detection in NLP

Deeplearning campus in Modulabs

Created by aisolab (@aisolab)

Agenda

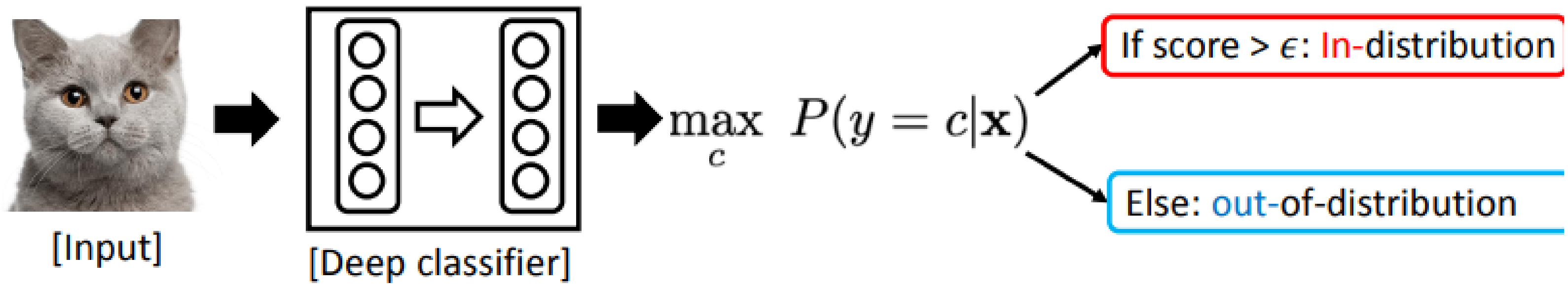
- What is Novelty Detection?
- How to solve?
- Applying to NLP
- To do

What is Novelty Detection?

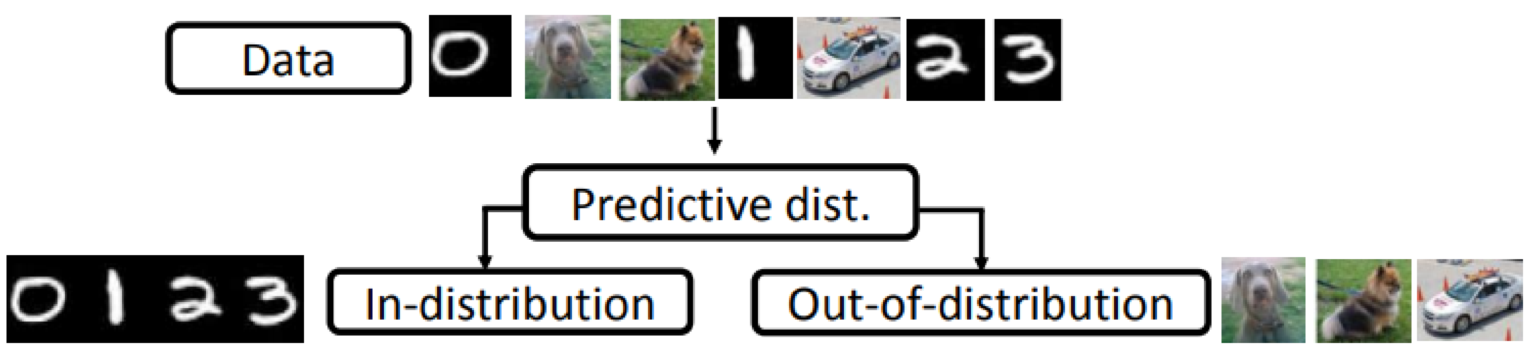
- Deep neural networks (DNNs) can be generalized well when the test samples are from similary distribution (i.e., in-distribution)
- However, in the real word, there are many unknown and unseen samples that classifier can't give a right anwser
- Novelty detection
    - Given pre-trained (deep) classifier,
    - Detect whether a test sample is from in-distribution (i.e, training distribution by classifier) or not (e.g., out-of-distribution / adversarial samples)

How to solve?

- Baseline detector
  - Confidence score = maximum value of predictive distribution



- Evaluation: detecting out-of-distribution
  - Assume that we have classifier trained on MNIST dataset
  - Detecting out-of-distribution for this classifier



How to solve?

- ODIN detector
    - Calibrating the posterior distribution using post-processing

- Two techniques
    - Temperature scaling

$$P(y = \widehat{y}|\mathbf{x};T) = \frac{\exp\left(f_{\widehat{y}}(\mathbf{x})/T\right)}{\sum_y \exp\left(f_y(\mathbf{x})/T\right)},$$

    - Input preprocessing

$$\mathbf{x}' = \mathbf{x} - \varepsilon \mathrm{sign}\left(-\nabla_{\mathbf{x}} \log P_\theta(y = \widehat{y}|\mathbf{x};T)\right),$$

    - Using two methods, the authors define confidence score as follows:

$$\text{Confidence score} = \max_y P(y|\mathbf{x}';T)$$

How to solve?

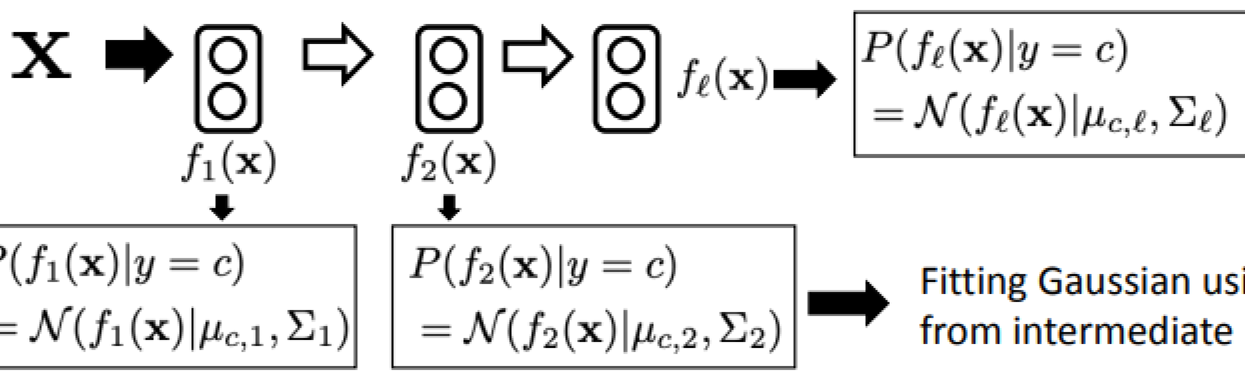
- Mahalanobis distance-based confidence score
  - Using generative classifier, we define new confidence score:

$$M(\mathbf{x}) = \max_c \; -\left(f(\mathbf{x}) - \widehat{\mu}_c\right)^\top \widehat{\mathbf{\Sigma}}^{-1} \left(f(\mathbf{x}) - \widehat{\mu}_c\right)$$

    - Measuring the log of the probability densities of the test sample
  - Boosting the performance
    - Input pre-processing

$$\widehat{\mathbf{x}} = \mathbf{x} + \varepsilon \text{sign}\left(\nabla_{\mathbf{x}} M(\mathbf{x})\right) = \mathbf{x} - \varepsilon \text{sign}\left(\nabla_{\mathbf{x}} \left(f(\mathbf{x}) - \widehat{\mu}_{\widehat{c}}\right)^\top \widehat{\mathbf{\Sigma}}^{-1} \left(f(\mathbf{x}) - \widehat{\mu}_{\widehat{c}}\right)\right)$$

    - Feature ensemble



Fitting Gaussian using features from intermediate layers

How to solve?

- Mahalanobis distance-based confidence score
  - Main algorithm

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 $\sum_\ell \alpha_\ell M_\ell$

- Remark that
  - We combine the confidence sco $\sum_\ell \alpha^\ell M^\ell$ e layers using weighted ensemble
  - Ensemble weights are selected by utilizing the validation set

Applying to NLP

- Applying mcb detection to below classfication models

- Using the Naver sentiment movie corpus v1.0
- Hyper-parameter was arbitrarily selected. (epoch: 5, mini_batch: 128)

	Train ACC (120,000)	Validation ACC (30,000)	Test ACC (50,000)
SenCNN	92.87%	86.87%	86.38%
CharCNN	85.63%	81.58%	81.58%
ConvRec	86.80%	82.66%	82.29%
VDCNN	86.31%	83.87%	83.90%
SAN	93.90%	86.52%	86.35%

- [x] Convolutional Neural Networks for Sentence Classification (as SenCNN)
  - https://arxiv.org/abs/1408.5882
- [x] Character-level Convolutional Networks for Text Classification (as CharCNN)
  - https://arxiv.org/abs/1509.01626
- [x] Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers (as ConvRec)
  - https://arxiv.org/abs/1602.00367
- [x] Very Deep Convolutional Networks for Text Classification (as VDCNN)
  - https://arxiv.org/abs/1606.01781
- [x] A Structured Self-attentive Sentence Embedding (as SAN)
  - https://arxiv.org/abs/1703.03130

Applying to NLP

- Convolutional Neural Networks for Sentence Classification

114 class	tr (133,743)	val (44,581)	tst (9,386)
softmax	96.57%	96.20%	96.06%
generative	99.23%	98.54%	98.42%

ood_tr: 83.86%	precision	recall	f1-score	support
in distribtuion	0.82	0.90	0.86	44,581
out of distribution	0.87	0.76	0.81	38,275

ood_val: 82.98%	precision	recall	f1-score	support
in distribtuion	0.79	0.90	0.84	9,386
out of distribution	0.89	0.76	0.82	9,569

To do

- ensemble MCB features across text classfication model
    - variety tokenization (eg. subword, character, word)
- BERT based