Robustness Metrics for AI Predictions using Deep Learning Methods

Davide Posillipo

Alkemy – Deep Learning & Big Data Department

BIAS RISK AND OPACITY IN AI
FIRST RESEARCH MEETING

1. Machine Learning Models can be overconfident

Machine Learning Models trained on a "closed" training set can incorrectly classify test samples from unknown classes as one of the known categories with high confidence. This notorious overconfident behavior of the model is known as "arrogance" or "**agnostophobia**" of the model.

A simple example of this behavior is the following: **what happens if we pass the picture of a chicken to the LeNet model, trained for MNIST digit classification?**

Despite the extremely good performance of this model on the known classes, a chicken picture is classified as a 6 digit picture with over 90% confidence. This result poses the question: **can we trust AI predictions?**

Similar behaviors are observed also when small perturbations are applied to the new samples' features set.

2. OOD Detection vs Model Robustness

Out-Of-Distribution (OOD) are samples that present a **semantic shift**: they belong to a different category than the ones observed for the In-Distribution (ID) samples (the ones we used to train our model). In other terms, the label space *Y* is different between ID and OOD.

The chicken picture is clearly an OOD sample, regardless its covariates distributions. **OOD detection** is a possible path to avoid overconfident predictions, that focuses entirely on the new sample.

Another approach is to focus on the behavior of the model instead: robustness will mean a model capable of stable predictions. This approach is better suited for the **covariate shift** case, situations where perturbation of the features can happen and shouldn't bring to meaningless prediction.

3. Detecting and avoiding overconfident predictions

There are scenarios where it's crucial to **avoid** meaningless predictions (e.g. healthcare, high precision manufacturing, autonomous driving…). We need a safety layer that prevents the model to produce harmful predictions, that we can put on top of the already existing prediction pipeline; it should work well for both semantic and covariate shifts. This system should work as sketched in the following pseudo-code:

```
if robustness_metrics < threshold:
    raise NotReliableClassification("Not reliable classification", y_predicted)
else:
    return y_predicted
```

4. How a (practical) robustness metrics should look like?

A robustness metrics should tell us if the prediction is reliable or if it is likely the result of the model agnostophobia. In order to be **relevant for real-world, in production applications**, this metrics should have the following properties:

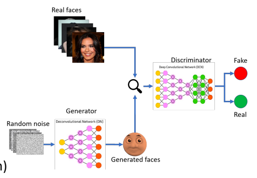
- Computable without labels
- Computable in streaming
- High effectiveness in detecting the anomaly points, either from semantic shift (new labels) or from covariate shift: low false negative rate
- Low rate of discarded "good predictions": low false positive rate
- Deployable into existing machine learning pipelines

5. Using WGAN for creating a robustness metrics

GAN: Generative Adversarial Networks, it's a model composed of two neural networks:

- Generator: generates inputs from random noise
- Discriminator: decides if the inputs from the Generator are authentic or artificial

WGAN: Wasserstein GAN, it uses Wasserstein distance as GAN Loss Function.



Procedure:

- Train a Generator (from random noise to the input space)
- Train a Discriminator (it tries to understand if a point is true or generated by the Generator)
  - The Generator learns how to fool the Discriminator
- Train an **Inverter** (from the input space to the latent representation)
  - The Inverter learns how the Generator maps from the random noise to the input space
- Given the new sample *x*, find the closest point to *x* in the latent space that, once translated back to the original space, is able to confound the classifier *f*. Let's call it *z**:

$$z^* = \operatorname*{argmin}_{\tilde{z}} \|\tilde{z} - \mathcal{I}_\gamma(x)\| \text{ s.t. } f(\mathcal{G}_\theta(\tilde{z})) \neq f(x)$$

Robustness metrics: the distance (delta_z = *z** - *I*(*x*)) between *z** and the inverted sample, in the latent representation.

- If delta_z is "small", the classifier is not confident about its prediction (the classifier "changes its mind" too easily)
- If delta_z is "big", the classifier is confident about the prediction (the classifier "knows what it wants")

Rule: if delta_z < threshold then do not predict

Intuition: decide if a prediction is worth the risk, checking the "stability" of the model *f* for the new input data x. The focus is on the behavior of the model *f* more than on the nature of *x*.

6. Using VAE for creating a robustness metrics

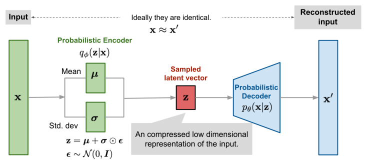
VAE: Variational Autoencoder, it's a model composed of two neural networks:

- Encoder: takes the inputs and "compress" them in a deep latent representation
- Decoder: takes the deep representation from the Encoder and creates back the original input as much as it can

VAE is a *variational* approach because instead of mapping the input into a fixed vector, we map it into a distribution.

Procedure:

- Train an Encoder (from input space to latent vector) on IID data
- Train a Decoder (from latent vector to input space) on IID data
- Compute the VAE loss function of the new sample *x*
- Compute the VAE loss function on known OOD data and use this distribution to define a threshold


Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{x}$
Mean
$\boldsymbol{\mu}$
Std. dev
$\boldsymbol{\sigma}$
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
An compressed low dimensional representation of the input.
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

Robustness metrics: VAE loss function. The VAE loss function expresses both the new sample reconstruction error on the original input space and the distance of the new sample latent representation from distribution learned during the training. An OOD input will have troubles in the encoding-decoding process -> high loss value.

- "Big" loss -> Not robust prediction: the new input data is OOD
- "Small" loss -> Robust prediction: the new input is ID

Rule: if VAE_loss > threshold then do not predict

Intuition: decide if a prediction is worth the risk, checking if the new sample is OOD or ID. The focus in on the nature of *x*, while the model *f* is not considered. This means that with this approach we don't need the classifier *f* for the robustness metrics computation, an inportant practical advantage over the WGAN-based approach.

7. Experiments with WGAN-based metrics

Setting

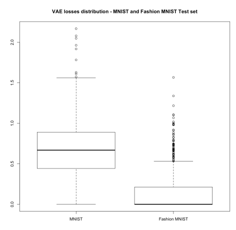
- Generator: 1 Fully Connected + 4 Transposed Conv. Layers (strides = 2), ReLu activation function
- Discriminator ("Critic"): 4 Conv. Layers (strides = 2) + 1 Fully Connected, ReLu activation function
- Inverter: 4 Conv. Layers (strides = 2) + 2 Fully Connected, ReLu activation function
- Adam optimizers for the three nets

- MNIST Dataset: 60K images for training set using for training the WGAN+Inverter, 10K images for test set used for defining the robustness metrics threshold
- Chicken picture: 28x28 pixels image used as OOD sample in Experiment 1
- FashionMNIST Dataset: 70K images used as OOD in Experiment 2

Definition of threshold

Distribution of delta_z computed on 500 test set ID samples.
5° percentile of delta_z distribution used as threshold: **5.3e-02**.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.4421  0.6681  0.6858  0.8901  2.1676 
```



Experiment 1: chicken picture (OOD) vs MNIST (ID)

- Delta_z for the chicken: 1.8e-06 (< 5.3e-02). The chicken picture is correctly identified as OOD.

Experiment 2: Fashion MNIST (OOD) vs MNIST (ID)

- Using the 5° percentile test delta_z as threshold, we get a 5% as false positive rate but a worrying 34.4% of lost good predictions (false negative rate).

8. Experiments with VAE-based metrics

Setting

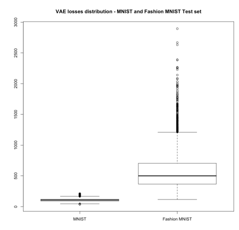
- Encoder: 2 layers fully connected neural network
- Decoder: 2 layers fully connected neural network
- Adam optimizer

- MNIST Dataset: 60K images for training set using for training the VAE, 10K images for test set used for defining the robustness metrics threshold
- Chicken picture: 28x28 pixels image used as OOD sample in Experiment 1
- FashionMNIST Dataset: 70K images used as OOD in Experiment 2

Definition of threshold

Distribution of VAE_loss computed on 10.000 test set ID samples.
Max of VAE_loss distribution used as threshold: **212.85**

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  37.71   90.63  106.45  105.38  121.60  212.85 
```



Experiment 1: chicken picture (OOD) vs MNIST (ID)

- VAE_loss for the chicken: 309.4 (>212). The chicken picture is correctly identified as OOD.

Experiment 2: Fashion MNIST (OOD) vs MNIST (ID)

- Using the maximum test loss as threshold, we get a 0% as false positive rate and a 3.55% of lost good predictions (false negative rate)
- Using the 95° percentile test loss as threshold, we get a 5% as false positive rate and a 0.25% of lost good predictions (false negative rate)

9. Pros and cons

This robustness metrics approach (in its VAE-based winning version) presents the following pros and cons:

Pros:

- No need to modify your predictive models
- The same "monitoring" system can be used for different ML models (for a given dataset)
- Applicable to any kind of data (tabular, images, …)
- "Easy" to explain and easy to plug-in into existing pipelines

Cons:

- Arbitrary thresholds must be set by the data scientists
- It introduces a further model that needs to be maintained and risks-checked

10. Conclusions and next steps

Agnostophobia clearly represents a relevant **risk** for AI applications; it also introduces **opacity**, making it complex to understand and interpret the models behavior.

This simple approach tries to mitigate the risk preventing the AI models to deliver unreliable predictions. It is well suited for deployment into already existing machine learning pipeline, a relevant and common scenario in industry.

Further work will be dedicated to find a more grounded threshold definition procedure; to explore other DL models suitable for the monitoring system; to profile and "explain" the OOD detected samples.