Review of "Are Generative Classifiers More Robust to Adversarial Attacks?" paper

Mathis Embit
ENS Paris-Saclay
Gif-sur-Yvette, France
mathis.embit@ens-paris-saclay.fr

ABSTRACT

As part of the MVA Probabilistic Graphical Models course, I review the paper "Are Generative Classifiers More Robust to Adversarial Attacks?"[3]. In this work I will use graphical models to understand both discriminative and generative classifier. In this way we will be able to make meaningful statements on the impact the classifier architecture has over their robustness. The generative classifier we will study is deep Bayes, proposed in the paper, which is a improved version of naive Bayes.

1 INTRODUCTION

1.1 Context

Deep neural network can easily be fooled by some imperceptible perturbation. Some attacks can even be effective on black-box networks. A classic example being image classifier. In a 2014 article, Goodfellow et al. [1] showed that perturbed input can results in the model outputting an incorrect answer with high confidence (Figure 1).

Ideas have been proposed to address this problem. For example,

- adversial training: add adversially perturbed data to the training data (Szegedy et al. [6])
- uncertainty estimate in bayesian neural networks (Li and Gal [4])
- removing noise from the inputs with a generative model (autoencoder or GAN) before feeding them to the classifier (Gu and Rigazio [2], Samangouei et al. [5])

However all these attempts concern discriminative classifiers and few research have been lead on generative classifiers. That's why the paper investigates it.

1.2 The paper solution

Concerning generative classifiers, popular ones such as naive Bayes are not well suited for real world data. That's why the paper proposes an improved version of naive Bayes, named Deep Bayes, which is a Deep Latent Variable Model learned with the VAE algorithm. Furthermore, they proposes 3 detection methods and make some experiments based on different kind of attacks.

1.3 My contributions

Reviewing this paper I tried to gain insight on it and put it into perspective thanks to graphical models.

- I add further graphical models explanations
- I provide more details on Deep Bayes construction
- I wrote the pseudo-code for Deep Bayes

MVA 2023, December 2023, Paris, FR
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

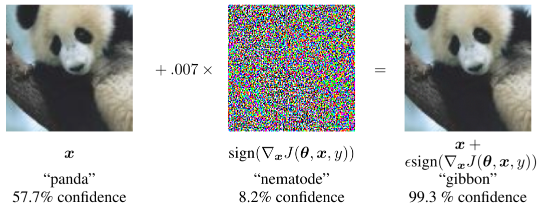

Figure 1: "A demonstration of fast adversarial example generation applied to GoogLeNet on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here the .007 ϵ corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers." by Goodfellow et al. [1]

2 FROM GENERATIVE CLASSIFIER TO DEEP BAYES

In this section I will introduce the deep Bayes model, created in the paper then used for their experiments.

2.1 Discriminative vs Generative classifier

We consider a dataset

$$\mathcal{D} = \left\{(x^{(n)}, y^{(n)}\right\}_{n=1}^{N}$$

of N couples. $x^{(n)} \in \mathbb{R}^D$ is the input and $y^{(n)} \in \{y_c, c = 1, \ldots, C\}$ is the corresponding label. The labels are one-hot encoded:

$$y_c = (0, \ldots, 0, \underset{\underset{\text{c-th}}{\uparrow}}{1}, 0, \ldots, 0) \in \{0,1\}^C$$

Theoritecally, the dataset in generated as follows:

$$y^{(n)} \sim p_{\mathcal{D}}(y) \text{ then } x^{(n)} \sim p_{\mathcal{D}}(x|y^{(n)})$$

where $p_{\mathcal{D}}$ is the ground-truth distribution of the dataset.

A **discriminative classifier** directly assumes functional form for $p_{\mathcal{D}}(y|x) = f(x)$ by learning the parameters θ of $\hat{f}_\theta(x)$ estimating $f(x)$, directly from training data. We will write $p(y|x)$ (here $= \hat{f}_\theta(x)$) the discriminative model approximating the ground-truth distribution.

A **generative classifier** first estimates the prior $p_{\mathcal{D}}(y)$ and the likelihood $p_{\mathcal{D}}(x|y)$ from training data. Then it uses Bayes rule to calculate the posterior $p_{\mathcal{D}}(y|x) = \frac{p_{\mathcal{D}}(y)p_{\mathcal{D}}(x|y)}{p_{\mathcal{D}}(x)} \propto p_{\mathcal{D}}(y)p_{\mathcal{D}}(x|y)$. We will write $p(y|x) = \frac{p(y)p(x|y)}{p(x)}$ the generative model approximating the ground-truth distribution.

Let's focus a little bit more on generative classifiers. We can think of generative classifier as a model aiming to understand the underlying data distribution, i.e. the joint density, as a proxy for estimating the class probabilities. A generative model of $\mathcal{D}$ can also be written as $p(x,y) = p(x|y)p(y)$ where $p(x|y)$ corresponds to one neural network and $p(y)$ to another.

To predict the label y^* of the input x^*, the model predicts the distribution of y^* knowing x^* rather than y^* itself:

$$\begin{aligned} p(y^*|x^*) &= \frac{p(x^*|y^*)p(y^*)}{p(x^*)} = \frac{p(x^*,y^*)}{p(x^*)} \\ &= \left(\frac{p(x^*,y_1)}{\sum_{c=1}^{C} p(x^*,y_c)}, \dots, \frac{p(x^*,y_C)}{\sum_{c=1}^{C} p(x^*,y_c)} \right) \\ &= \text{softmax}_{c=1}^{C} \left(\log p(x^*,y_c) \right) \end{aligned}$$

Intuitively, y^* has more chance to be equal to y_c if the likelihood $p(x^*,y_c)$ is high.

In terms of graphical models:

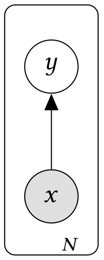

Figure 2: Graphical model of a discriminative model

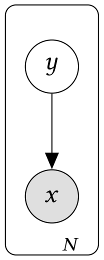

Figure 3: Graphical model of a generative model

In the discriminative model graph (Figure 2), the arrow indicates a direct inference from x to y, signifying the ability to deduce $p(y|x)$ directly from the provided x. Conversely, in the generative model graph (Figure 3), the arrow points in the opposite direction, implying the necessity to initially deduce the values of $p(y)$ and $p(x|y)$ from the data and subsequently employ them to calculate $p(y|x)$.

2.2 Naive Bayes

The most well-known generative classifier is Naive Bayes. This model makes a strong assumption over the dataset: given a class, the features are independent within that class. Formally,

$$\begin{aligned} p(x|y) &= \prod_{d=1}^{D} p(x_d|y) \\ \iff & x_1|y \perp\!\!\!\perp \cdots \perp\!\!\!\perp x_d|y \end{aligned}$$

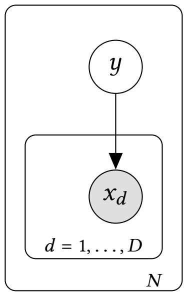

Figure 4: Graphical model of naive bayes

This assumption is indeed pretty naive and is completely inappropriate for data we have in the real world. For example, pixels of an image are not independent within a class of images. To perform test on images, the paper proposes a new generative classifier architecture that does not rely on the naive independence assumption.

2.3 Deep Bayes and graphical models

Their solution is a Deep Latent Variable Model (DLVM), trained by the same algorithm as a Variation AutoEncoder (VAE). As suggested by the name, we introduce a new variable z which is **latent**. We now want to approximate $p_{\mathcal{D}}(x,y)$ by the DLVM $p(x,z,y)$.

Here, $p(x|y) = \frac{p(x,y)}{p(y)} = \frac{\int p(x,z,y)dz}{\int p(x,z,y)dzdx}$ is not factorized as in Naive Bayes. That means the DLVM is able to capture much more underlying structure in the data than Naive Bayes.

As for Naive Bayes, we want to split the computation of the joint distribution $p(x,z,y)$ between multiple (here 3) other distributions, each computed by one neural network. Each definition of the model $p(x,z,y)$ results in a different classifier that can be either discriminative or generative. Let's find the interesting ways of defining $p(x,z,y)$ with our new knowledge in graphical models.

We have 3 random variables: the input x, the label y and the latent variable z. We need to link them together to model the underlying structure explaining the label from the input.
Our graphs can be model from 3 characteristics:

- **D**iscriminative or **G**enerative
  - Discriminative: we use x and z to directly infere $p(y|x)$. In the graphical model the y node only receives arrows, no arrows start from it.
  - Generative: we use y and z to deduce some intermediate distribution before using them to calculate $p(x|y)$. In the graphical model the x node only receives arrows, no arrows start from it.
- **F**ully connected or **B**ottleneck
  - Fully connected: one of the distribution of the $p(x,z,y)$ decomposition is conditioned by 2 variables. In the graphical model, there are 3 edges.
  - Bottleneck: none of the distribution of the $p(x,z,y)$ decomposition is conditioned by 2 variables. In the graphical model, there are 2 edges.
- First node in the topological order: **X**, **Y** or **Z**

- X: $p_{\mathcal{D}}(x)$ appears in the $p(x,z,y)$ decomposition. In the graphical models arrows start from X.
- Y: $p_{\mathcal{D}}(y)$ appears in the $p(x,z,y)$ decomposition. In the graphical models arrows start from Y.
- Z: $p(z)$ appears in the $p(x,z,y)$ decomposition. In the graphical models arrows start from Z.

Therefore, there are $2 \times 2 \times 3 = 12$ possible graphs.
Here are 3 obvious remarks that will help us eliminate useless graphs:

- We do not want a node to be separate from the 2 other, it would have no effect on the classification process so it's pointless. We then need 2 or 3 edges. This one is already satisfied by the Fully connected/Bottleneck characteristic.
- We do not want the latent variable z to be the last node in the topological order. It would mean that z has absolutely no effect on x nor y, and that we added a useless variable. Therefore we eliminate graphs where z is the last node in the topological order. There are 2 such graphs that also satisfy the 3 previous characteristics.
- As graphical models are Directed Acyclic Graphs we eliminate the two possible cycles. There are 2 cycles.

Therefore, there are now $12 - 2 - 2 = 8$ possible graphs.
In the paper they omitted the (DBZ) graph. They did not provide a clear explanation, hence we will keep it. In the following graphical models the node x can represent any configuration between the x_d. Here are the 8 graphical models:

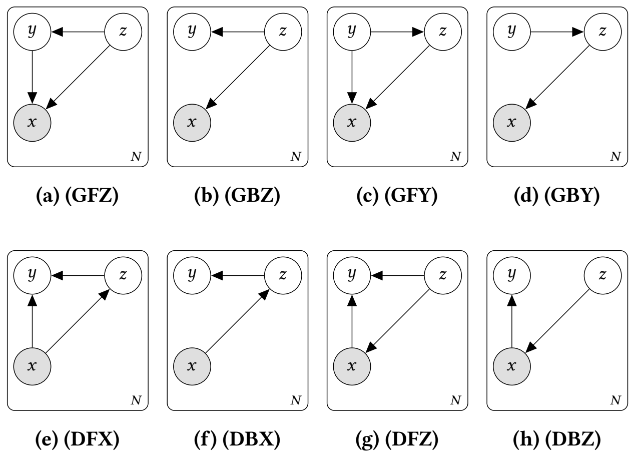

Figure 5: Graphical models of the factorisation structures of $p(x,z,y)$

Here are the formulas corresponding to the different factorisations:

$$p(x,z,y) = p(z)p(y|z)p(x|z,y) \quad \text{(GFZ)}$$
$$p(x,z,y) = p(z)p(y|z)p(x|z) \quad \text{(GBZ)}$$
$$p(x,z,y) = p_{\mathcal{D}}(y)p(z|y)p(x|z,y) \quad \text{(GFY)}$$
$$p(x,z,y) = p_{\mathcal{D}}(y)p(z|y)p(x|z) \quad \text{(GBY)}$$

$$p(x,z,y) = p_{\mathcal{D}}(x)p(z|x)p(y|x,z) \quad \text{(DFX)}$$
$$p(x,z,y) = p_{\mathcal{D}}(x)p(z|x)p(y|z) \quad \text{(DBX)}$$
$$p(x,z,y) = p(z)p(x|z)p(y|x,z) \quad \text{(DFZ)}$$
$$p(x,z,y) = p(z)p(x|z)p(y|x) \quad \text{(DBZ)}$$

For **training** we adopt the VAE algorithm. We learn the approximation $q(z|x,y)$ of $p(z|x,y)$, and p by maximizing the variational lower-bound deriving from:

$$\begin{aligned}
&\log p(x,y) \\
&= \int \log p(x,y) q(z|x,y) dz \\
&= \int \log \left(\frac{p(x,z,y)}{p(z|x,y)} \right) q(z|x,y) dz \\
&= \int \log \left(\frac{p(x,z,y)}{q(z|x,y)} \frac{q(z|x,y)}{p(z|x,y)} \right) q(z|x,y) dz \\
&= \int \log \left(\frac{p(x,z,y)}{q(z|x,y)} \right) q(z|x,y) dz + \underbrace{\int \log \left(\frac{q(z|x,y)}{p(z|x,y)} \right) q(z|x,y) dz}_{=\mathrm{KL}(q(.|x,y)||p(.|x,y)) \underbrace{\geq 0}_{Jensen}} \\
&\geq \int \log \left(\frac{p(x,z,y)}{q(z|x,y)} \right) q(z|x,y) dz = \mathcal{L}_{\mathrm{VI}}(x,y)
\end{aligned}$$

There are 2 equivalent points of view: by maximizing the variational lower-bow we maximize the log-likelihood $\log p(x,y)$ or we minimize the Kullback-Leibler divergence between $p(.|x,y)$ and its approximation $q(.|x,y)$, $\mathrm{KL}(q(.|x,y)||p(.|x,y))$.
When training on our dataset $\mathcal{D}$ the variational lower-bound become:

$$\mathbb{E}_{\mathcal{D}}\left[\mathcal{L}_{\mathrm{VI}}(x,y)\right] = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_q \left[\log \frac{p(x_n, z_n, y_n)}{q(z_n|x_n,y_n)} \right] \quad (1)$$

Once it's trained we can sample as explained in the paragraph on generative classifiers. To predict the label y^* of the input x^* we use:

$$p(y^*|x^*) = \mathrm{softmax}_{c=1}^{C}\left(\log p(x^*, y_c)\right)$$

For Deep Bayes we now have:

$$\begin{aligned}
\log p(x^*, y_c) &= \log \int p(x^*, z, y_c) dz \\
&= \log \int \frac{p(x^*, z, y_c)}{q(z|x^*, y_c)} q(z|x^*, y_c) dz \\
&\approx \log \frac{1}{K} \sum_{k=1}^{K} \frac{p(x^*, z_c^k, y_c)}{q(z_c^k|x^*, y_c)}
\end{aligned}$$

where $z_c^k \sim q(z|x^*, y_c)$.

Then:

$$p(y^*|x^*) \approx \text{softmax}_{c=1}^{C}\left(\log \frac{1}{K}\sum_{k=1}^{K}\frac{p(x^*, z_c^k, y_c)}{q(z_c^k|x^*, y_c)}\right) \tag{2}$$

2.4 Consequences of the architecture

Different graphical models impose different assumptions on the data generation process.

- With GFZ and GBZ: z affects both x and y
- With GFY and GBY: z distribution depends on the class
- The bottleneck models GBZ, GBY, DBX and DBZ, give no other choice than using z compared to the fully-connected models

Here is an important remark because a lot of adversarial attack detection methods rely on the data manifold assumption, i.e. that the attacks are far away from $p_{\mathcal{D}}(x)$:

- For generative classifiers, a certain probability of x affects the predictions
- In DFX and DBX, $p_{\mathcal{D}}(x)$ cancel out in the equation (2). Therefore, it makes it impossible to know if x is close to the data manifold or not. For example, for (DFX):

$$\frac{\frac{1}{K}\sum_{k=1}^{K}\frac{p_{\mathcal{D}}(x^*)p(z_c^k|x^*)p(y_c|z_c^k,x^*)}{q(z_c^k|x^*,y_c)}}{\sum_{c=1}^{C}\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p_{\mathcal{D}}(x^*)p(z_c^k|x^*)p(y_c|z_c^k,x^*)}{q(z_c^k|x^*,y_c)}\right)}$$

$$=\frac{\cancel{p_{\mathcal{D}}(x^*)}\frac{1}{K}\sum_{k=1}^{K}\frac{p(z_c^k|x^*)p(y_c|z_c^k,x^*)}{q(z_c^k|x^*,y_c)}}{\cancel{p_{\mathcal{D}}(x^*)}\sum_{c=1}^{C}\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p(z_c^k|x^*)p(y_c|z_c^k,x^*)}{q(z_c^k|x^*,y_c)}\right)}$$

$$=\frac{\frac{1}{K}\sum_{k=1}^{K}\frac{p(z_c^k|x^*)p(y_c|z_c^k,x^*)}{q(z_c^k|x^*,y_c)}}{\sum_{c=1}^{C}\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p(z_c^k|x^*)p(y_c|z_c^k,x^*)}{q(z_c^k|x^*,y_c)}\right)}$$

3 ALGORITHM

The paper did not provide a clear algorithm and their code was not well documented. Hence, here is the pseudo-code of the Deep Bayes model.

Algorithm 1 proba_for_y

```
Require: $x^* \in \mathbb{R}^D, y_c \in \{0,1\}^C$
Ensure: $\log p(x^*, y_c)$
  # approximate posterior $q$
  $\mu, \sigma \leftarrow$ VAE_encoder$(x^*, y_c)$   ▷ creates latent parameters of $z_c^k$
  $z \leftarrow$ sample_gaussian$(\mu, \sigma)$   ▷ $z_c^k \sim q(z|x^*, y_c)$
  log_q ← log_gaussian_prob$(z, \mu, \sigma)$   ▷ $q(z_c^k|x^*, y_c)$

  # prior $p(z)$
  log_pz ← log_gaussian_prob$(z, 0, 0)$   ▷ $p(z_c^k)$

  # Intermediate $p(y|z)$
  y_logit ← pyz$(z)$   ▷ $p(y|z_c^k)$
  # were pyz $= p(y|z)$ 1 hidden layer MLP
  log_pyz ← softmax_with_logits$(y_c$, y_logit$)$   ▷ $p(y_c|z_c^k)$

  # Likelihood $p(x|z, y)$
  mu_x ← pxzy$(z, y)$   ▷ $x \sim p(x|z_c^k, y_c)$
  # where pxzy $= p(x|z, y)$ 2 hidden layers MLP
  mu, logsig ← mu_x
  log_pxzy ← log_gaussian_prob$(x^*$, mu, logsig$)$   ▷ $p(x^*|z_c^k, y_c)$

  proba ← $\beta \times$ log_pxzy + log_pyz + log_pz − log_q
  # proba = $\log\left(\frac{e^{\beta}p(x^*|z_c^k,y_c)p(y_c|z_c^k)p(z_c^k)}{q(z_c^k|x^*,y_c)}\right)$
  Return proba
```

Algorithm 2 Deep Bayes

```
Require: $x^* \in \mathbb{R}^D$
Ensure: $y* \sim p(y|x^*)$
  logpxy ← []
  for $c = 1, \ldots, C$ do
      proba ← proba_for_y$(x^*, y_c)$
      logpxy.append(proba)
  end for
  Return softmax$_{c=1}^{C}$(logpxy)
```

REFERENCES

[1] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572 [stat.ML]

[2] Shixiang Gu and Luca Rigazio. 2015. Towards Deep Neural Network Architectures Robust to Adversarial Examples. arXiv:1412.5068 [cs.LG]

[3] Yingzhen Li, John Bradshaw, and Yash Sharma. 2019. Are Generative Classifiers More Robust to Adversarial Attacks? arXiv:1802.06552 [cs.LG]

[4] Yingzhen Li and Yarin Gal. 2017. Dropout Inference in Bayesian Neural Networks with Alpha-divergences. arXiv:1703.02914 [cs.LG]

[5] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. arXiv:1805.06605 [cs.CV]

[6] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. arXiv:1312.6199 [cs.CV]

Received 22 December 2023