A GAN based solver of black-box inverse problems

1 Abstract

We propose a GAN based approach to solve inverse problems which have non-differential or non-continuous forward relations. In the standard sense, an inverse problem is interpreted as the process of calculating factors that produce observations. We reformulate the inverse problem such that the discriminator is a binary classifier and the generator is used to produce samples in a local region of the input domain of the forward relation. Our GAN based approach solves inverse problems by using adversarial training but without relying on the gradients of the original problem formulation. We prove the efficacy of our approach by applying it to an artificially generated topology optimization problem. We demonstrate that despite not having access to derivatives of \( f \) our method leads to similar results than more traditional topology optimization methods.

2 Introduction

Inverse problems is a long established field of problems with applications in science and engineering [11]. Inverse problems consist of calculating the causal factors \( x \) that produced a set of observations via a forward relation \( f \). Their solutions are widespread including the fields of physics [3] and medicine [4]. In the linear case, even large-scale inverse problems can be solved by established algorithms [11]. However, in the non-linear case, they often result in ill-posed problems which for example need strong regularization [8]. Problems get even more severe when a non-differential or non-continuous forward relation is involved.

We propose a novel method to solve these kind of inverse problems by applying core principles of Generative Adversarial Networks (GANs) [5]. In this work, the discriminator \( D \) is used as a continuous proxy of \( f \). However, instead of trying to approximate \( f \) directly, we reformulate the task such that \( D \) is a binary classifier, and therefore simplify the problem. As a second simplification, the generator \( G \) is used to produce samples in a local region of the input domain of the forward relation \( \mathcal{X} \). The generator can be seen as spotlight which illuminates parts of \( \mathcal{X} \) and thereby enables the discriminator to locally learn a continuous approximate of \( f \). The usage of a generator allows to encode prior knowledge of the problem and thus helps to restrict the search space in \( \mathcal{X} \) and avoids non-plausible or infeasible solutions.

One major advantage of our proposed method is, that we find meaningful update directions in the sample space \( \mathcal{X} \) without having access to the gradients of \( f \). Our method is therefore capable of performing derivative free optimization [2] as well as black box optimization [9]. Furthermore, it does not directly depend on the response of \( f \) and is therefore less prone to rapid changes of \( f \).

3 Problem characterization

In the following, we define the class of inverse problems our method intends to solve. We consider an arbitrarily complicated (black-box) forward relation \( f : \mathcal{X} \rightarrow \mathbb{R} \). The only two restrictions we need to impose on \( f \) are the following: i) given some \( x \in \mathcal{X} \) it is computationally feasible to retrieve the

value \( f(x) \). ii) a ranking on the input domain \( \mathcal{X} \) is imposed by \( f \), i.e. \( x_1 \) is ranked higher than \( x_2 \) iff \( f(x_1) > f(x_2) \). More precisely, we demand a weak ordering of the input domain \( \text{w.r.t.} f \). The inverse problem at hand is the task to find a subset of samples \( S \subset \mathcal{X} \) that lead to a specific response \( c: s \in S \implies f(s) = c \). In practice, it is often sufficient to find a solution which meets a certain quality threshold. We can exploit this line of thought and relax the problem by not specifying an exact response value \( c \), but rather an upper bound \( u \). This way, we end up with the following problem formulation: Given an upper bound \( u \in \mathbb{R} \) and a forward relation \( f: \mathcal{X} \to \mathbb{R} \) that suffices restrictions i) and ii), we want to find a non-empty set \( S \) of samples that lead to a response equal to or lower than \( u \):

\[
S \subseteq \{x \in \mathcal{X} \mid f(x) \leq u\}. \tag{1}
\]

We call any set of the above form a solution to our inverse problem.

### 4 How GANs solve complex non-linear inverse problems

To solve the inverse problem, the GAN approach needs to be modified. In the original GAN setting the target distribution \( \mathbb{P}_w \) is fixed and can be accessed through the training data set. However, this is not the case for the inverse problem. We do not know any sample from the set \( \{x \in \mathcal{X} \mid f(x) \leq u\} \) and we do not require to learn the whole distribution of samples with a score lower than \( u \). In principle, one solution is sufficient, and therefore our generator is not required to produce a diverse set of designs.

The initially generated samples are likely to not overlap with the set of possible solutions. Therefore, it has to be ensured that during training the generator is pushed into regions of \( \mathcal{X} \) with lower response. We observe the following: Firstly, for an update only local information can be used, i.e. we can only use the current subspace that is populated by the generated samples. Secondly, we make the assumption that even if only local information is used, generalization beyond the current co-domain of the generator might be achieved through an inductive bias on \( G_0 \) and \( D_w \).

Our strategy is to train \( D_w \) to classify the generator’s co-domain into regions with higher and lower response. By choosing a percentile \( p \in [0, 1] \) we split the co-domain into two sets, where one set contains the best ((\( 1 - p \)) \cdot 100)\% of samples with lowest response and the other one contains the rest.

Once the discriminator is trained on this split, the task of the generator is to produce more samples with lower response. We can track the progress of the generator by following the responses of the \(( (1 - p) \cdot 100) \%)\%-percentile. More specific, we track the value of the highest response \( \tau \) of our best samples. A set of solutions is found if this value has arrived at our upper bound \( \tau = u \).

More formally, we choose the latent space of \( G_0 \) to be \( d \)-dimensional uniformly distributed \( z \sim U(0, 1)^d \). For a fixed update step, we denote the generator’s co-domain as \( \mathcal{G} \). We impose a weak ordering on \( \mathcal{G} \) via our forward relation \( f \). This allows to split \( \mathcal{G} \) into two sets \( \mathcal{G}^0 \) and \( \mathcal{G}^1 \), such that (a) \( \tilde{x} \in \mathcal{G}^0, \ x \in \mathcal{G}^1 \implies f(\tilde{x}) \geq f(x) \) and (b) for a percentile of our choice \( p \in [0, 1] \) we have \( \mathbb{P}_\theta(\mathcal{G}^0) = p \) and \( \mathcal{G}^1 = \mathcal{G} \setminus \mathcal{G}^0 \). This split defines a threshold \( \tau \in \mathbb{R} \) such that

\[
\tau = \sup_{x \in \mathcal{G}^1} f(x), \tag{2}
\]

\[
f(\tilde{x}) \geq \tau \geq f(x), \ \forall \tilde{x} \in \mathcal{G}^0, \ \forall x \in \mathcal{G}^1. \tag{3}
\]

where \(( (1 - p) \cdot 100) \%)\% of the highest scoring samples lie in \( \mathcal{G}^1 \). The task for \( D_w \) now is to discriminate between samples of the two sets. This is a classification task where the samples \( x \in \mathcal{G} \) are equipped with labels \( y(x) = \mathbb{I}[x \in \mathcal{G}^1] \), where \( \mathbb{I}[\cdot] \) is the indicator function. We minimize the discriminator loss

\[
L_D = -\mathbb{E}_{x \sim \mathbb{P}_\theta}[ y(x) \log(D_w(x)) + (1 - y(x)) \log(1 - D_w(x))^\]. \tag{4}
\]

Now let’s assume that for a fixed update step the discriminator is able to perfectly classify the co-domain of the generator. As discussed before, we assume that even though the discriminator was trained only locally, it has learned features that reflect the structure of the problem. Its decision boundary therefore is meaningful even in regions outside of \( \mathcal{G} \), i.e. we assume that samples from one side of the decision boundary are prone to have higher responses than sample from the other side.

The task of the generator is to move its mass into the direction of samples with lower response. For that we can leverage the discriminator, which has learned the decision boundary between the samples
with higher and lower response than $\tau$. Only for generated samples with a response higher than the
threshold $\theta$ needs to be adjusted. We minimize the generator loss:

$$L_G = -\mathbb{E}_{x \sim U(0,1)^{\dim(x)}}[ (1 - y(G_\theta(z))) \log(1 - D_w(G_\theta(z))) ] ,$$

where we ignore the labeling function when taking the derivative:

$$\nabla L_G = -\frac{\partial}{\partial \theta} \bigg|_{\theta = \hat{\theta}} \mathbb{E}_{x \sim U(0,1)^{\dim(x)}}[ (1 - y(G_\theta(z))) \log D_w(G_\theta(z)) ]$$

$$= -\mathbb{E}_{x \sim U(0,1)^{\dim(x)}}[ (1 - y(G_\theta(z))) \frac{\partial}{\partial \theta} \bigg|_{\theta = \hat{\theta}} \log D_w(G_\theta(z)) ] .$$

If we update the generator accordingly, this will amount to a change of mass in $\mathbb{P}_\theta$ into the region
of samples with lower score and thus will also lead to a decrease of $\tau$. In practice, initially we
draw a sample $\hat{G} = \{G_\theta(z_i)\}_{i=1}^n$ from the generator’s co-domain. We split $\hat{G}$ according to the
percentile $\rho$ with respect to the responses $f(G_\theta(z_i))$ into the sets $\hat{G}^0$ and $\hat{G}^1$, set our threshold to
$\hat{\tau} = \max_{x \in \hat{G}} f(x)$ and label the samples according to $y_i = \mathbb{I}[\hat{\tau} \geq f(G_\theta(z_i))]$. However, instead
of drawing a completely new set from the generator after every update, we use $\hat{G}$ as a buffer, where
we only replace samples if they have lower score than one of the samples in the buffer. While this
decouples the set $\hat{G}$ to some extend from $\mathbb{P}_\theta$, the hope is to gain a stronger pull towards smaller
responses.

5 Application: Topology optimization

In the field of Computer Aided Design (CAD), the goal often is to find the stiffness or its inverse, the
compliance $c$, of a given design $x$. This design is exposed to external forces $F$ and is fixed at one
or more anchor points where the load-dependent displacement is fixed to zero. Here, our forward
relation is the mapping

$$f_{\text{FEM}} : (0,1)^{\dim(x)} \rightarrow \mathbb{R},$$

$$x \mapsto f_{\text{FEM}}(x),$$

where $c$ is the compliance of sample $x$. Such class of problems is often solved by the Finite Elements
Method (FEM) \[12\]. In topology optimization one usually wants to find a design $x$ whose quality in
terms of compliance is minimized \[10\].

We use a CNN-based \[7\] generator consisting of 5 convolutional layers to propose designs $x$ to
the topology optimization problem. The forward relation $c = f_{\text{FEM}}(x) = U^T K U$ assigns a value
c $\in \mathcal{R}$ to generated designs $x$ and thus, introducing a weak ordering to the set of designs. The global
displacement vector $U$ is found by solving the finite element equation system $KU = F$, where the
sparse global stiffness matrix $K$ is a function of $x$, and $F$ describes the external forces acting on the
system. Anchor points are treated as additional constraints on $U$. To enable local exploration we
added random noise on the boundaries of the generated designs. This is achieved by masking the
noise with edge-detector convolutions applied to the design $x$.

6 Results

We evaluate our topology optimization method on artificially designed problems on a $128 \times 128$
FEM grid and compare it to standard topology optimization software \[10\] on the same problem with
identical model hyper-parameters. The problem setup consisted of (i) ranges where the designed
part can be anchored and (ii) areas with distributed loads. The task is to design a structure which
minimizes the compliance while using only 25% of the available design-space. While standard
software performs smoothing of solutions we omitted this step. Training our GAN based solver on
the topology optimization task took about 5 hours on a single GPU. Two results of the design process
are shown in Fig.\[1\].

Although our GAN based optimization approach does not have access to gradients or any other
information of the model $f_{\text{FEM}}$, it delivers similar structured parts than a standard optimization
software \[6\]. The structured parts have on par to slightly worse quality scores depending on the
problem. Our system is capable of coming up with a variety of solutions which are local minima of the artificially designed problems by changing the initial parameters of generator and discriminator. One could change the initial solution for traditional methods too. However, in practice they often converge to a single minimum solution regardless of initialization.

7 Discussion

We propose a GAN based method to solve non-differential or non-continuous inverse problems. We reformulate the inverse problem such that the discriminator is a binary classifier and the generator is used to produce samples in a local region of the input domain of the forward relation. Our GAN based approach solves inverse problems without relying on the gradients of the original problem formulation. We prove the efficacy of our approach by applying it to a topology optimization problem. We observe a generator which designs gradually become better.

As a future direction, one could extend this approach to a much broader set of general optimization problems. In these problems the generator comes up with parameters $\theta$ which optimize a function $f(\theta)$. The discriminator learns a surrogate of $f$. The proposed method is expected to also perform well on optimization problems in high dimensions since GANs are particularly good at approximating high-dimensional distributions.

References


