Adversarial Variational Optimization of Non-Differentiable Simulators

Sebastian Pina Otey

Grupo AIA/IFAE

21 December 2017

Outline

Contents

Problem statement

In many fields, complex data generation processes are used to relate observations $\vec{x}$ to parameters $\vec{\theta}$ of an underlying theory or mechanistic problem. This is,

$$\vec{x} \sim p(\vec{x}|\vec{\theta}).$$

The problem at hand is to find the parameters $\vec{\theta}^*$ which fits the observed data the best.
But the likelihood functions are usually intractable!!

Problem statement

Consider a family of parametrized densities $p(\vec{x}|\vec{\theta})$, where $\vec{x} \in \mathbb{R}^d$ is the data and $\vec{\theta}$ the parameters. Consider some complicated latent process where $\vec{z} \in \mathcal{Z}$ when running the simulation that adds randomness to the generation. Then, the stochastic generative process of $p(\vec{x}|\vec{\theta})$ is specified through $g(\cdot;\vec{\theta}) : \mathcal{Z} \to \mathbb{R}^d$, which means,

$$x \sim p(\vec{x}|\vec{\theta}) \text{ is the same as } \vec{z} \sim p(\vec{z}|\vec{\theta}),\ \vec{x} = g(\vec{z};\vec{\theta}),$$

such that

$$p(\vec{x}|\vec{\theta}) = \int_{\{\vec{z}:g(\vec{z};\vec{\theta}=\vec{x}\}} p(\vec{z}|\vec{\theta})\mu(d\vec{z}).$$

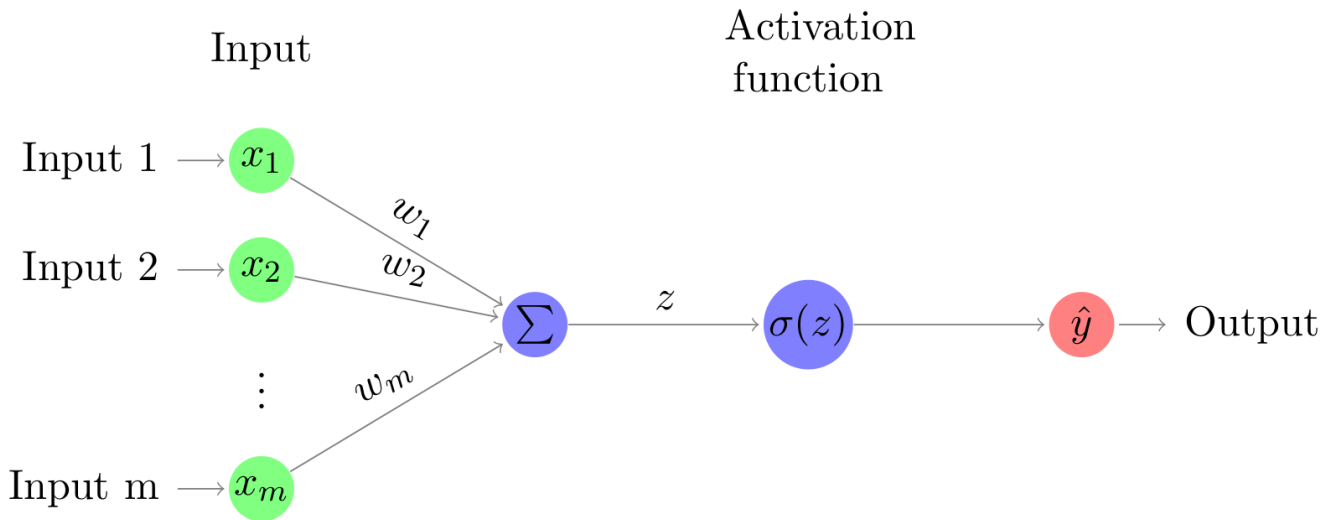
Problem statement

Given some observed data $\{\vec{x}_i\}_{i=1,\ldots,N}$ from the true distribution $p_r(\vec{x})$, we want to find the parameters $\vec{\theta}^*$ that minimize the divergence/distance ρ between $p_r(\vec{x})$ and the implicit model $p(\vec{x}|\vec{\theta})$:

$$\vec{\theta}^* = \underset{\vec{\theta}}{\text{argmin}}\ \rho(p_r(\vec{x}), p(\vec{x}|\vec{\theta})).$$

Contents

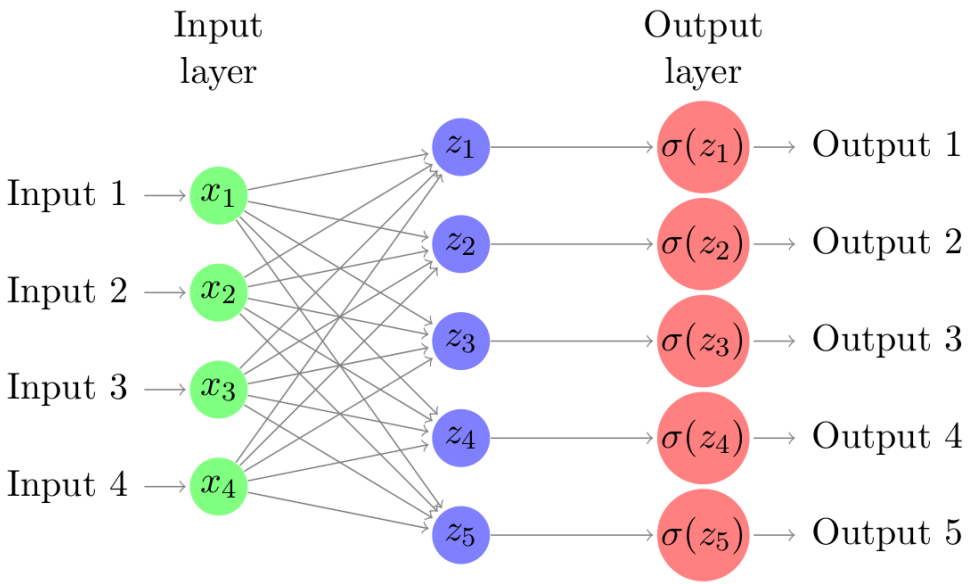
McCulloch and Pitts neuronal model



$$\hat{y} = \sigma(z), \quad z = \sum_{i=1}^{m} x_i w_i,$$

where σ is the **activation function** and w_i are the weights of the network.

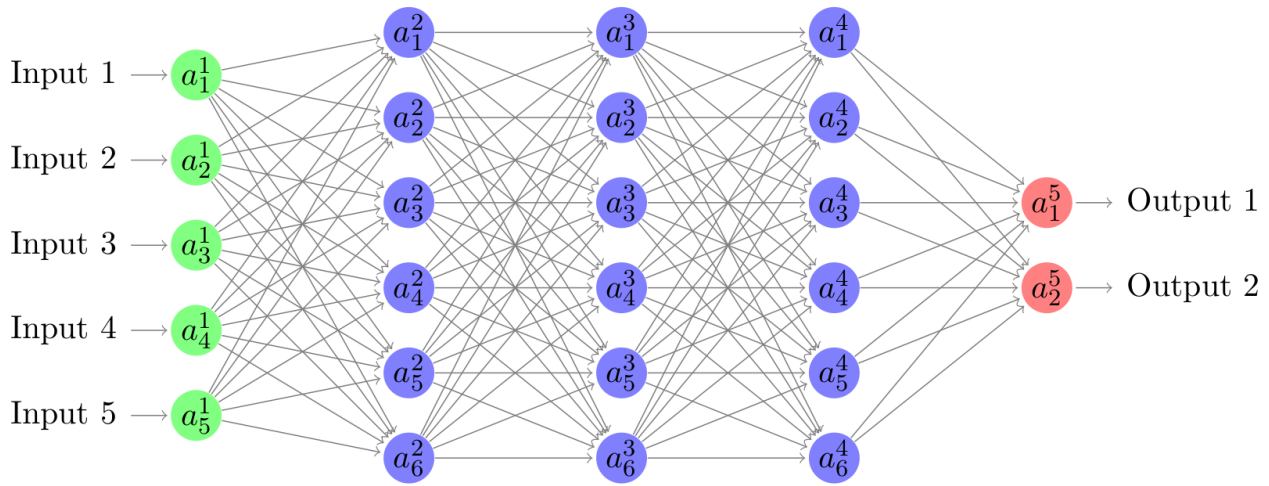
Perceptron



$$z_j = \sum_{i=0}^{m} w_{ji} x_i = w_{j0} + \sum_{i=1}^{m} w_{ji} x_i = b_j + \sum_{i=1}^{m} w_{ji} x_i,$$

where b_j is the **bias term**.

Deep Neural Network



$$a_j^l = \sigma(z_j^l) = \sigma\left(\sum_i w_{ji}^l a_i^{l-1} + b_j^l\right),$$

or in matrix notation,

$$a^l = \sigma(z^l) = \sigma(w^l a^{l-1} + b^l).$$

Backpropagation

Consider the cost (loss) function C. Instead of

$$\frac{\partial C}{\partial \theta_i} \simeq \frac{C(\theta_i + \varepsilon) - C(\theta_i)}{\varepsilon},$$

we use the **backpropagation** method,

$$\delta^L = \nabla_{a^L} C \odot \sigma'(z^L), \tag{1}$$

$$\delta^l = \left(\left(w^{l+1} \right)^T \delta^{l+1} \right) \odot \sigma'(z^l), \tag{2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j, \tag{3}$$

$$\frac{\partial C}{\partial w^l_{jk}} = \delta^l_j a^{l-1}_k. \tag{4}$$

Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an example of how to update the weights using backpropagation. M random samples of the training data are selected and used:

$$w^l \rightarrow w^l - \frac{\eta}{M} \sum_{m=1}^{M} \delta^{m,l} \left(a^{m,l-1}\right)^T,$$

$$b^l \rightarrow b^l - \frac{\eta}{M} \sum_{m=1}^{M} \delta^{m,l},$$

where η is the learning parameter. There are other optimizers, being Adam the most popular one.

Generative Adversarial Networks

GANs are a kind of **generative models**.

Generative Adversarial Networks

GANs are a kind of **generative models**.
We will consider them to be any model that, from a training set sampled from a distribution $p_r(\vec{x})$, learns to produce an estimation of such distribution in any form of it. The resulting estimation will be denoted as $p(\vec{x})$. This estimation can be an explicit form of the distribution. It might also be a mechanism only able to sample new data from it. It might be both.

GAN framework

The idea behind GANs is a game (in a mathematical sense) between two players: a *generator* (a function G, representing the first NN) and a *discriminator* (a function D, representing the second NN). The first one learns to produce samples imitating the distribution of the training data set. The later one learns to distinguish real data, from the training set, from fake one, produced by the generator. Hence the game consists in the discriminator classifying data into real or fake one, while the generator has to try to trick the discriminator by generating more realistic data. It is an adversarial situation.

$$Z \xrightarrow{G} X \xrightarrow{D} \{0, 1\}.$$

GAN framework

The networks have each a cost function C that depend on the parameters of the networks. The discriminator wishes to minimize $C^{(D)}(\phi, \theta)$ while only controlling ϕ. On the other hand, the generator wishes to minimize its own cost function $C^{(G)}(\phi, \theta)$ while only controlling θ.

GAN framework

The networks have each a cost function C that depend on the parameters of the networks. The discriminator wishes to minimize $C^{(D)}(\phi, \theta)$ while only controlling ϕ. On the other hand, the generator wishes to minimize its own cost function $C^{(G)}(\phi, \theta)$ while only controlling θ.
The solution to a game is a Nash equilibrium, which in our case is a tuple (ϕ, θ) which is a local minimum of $C^{(D)}$ with respect to ϕ and a local minimum of $C^{(G)}$ with respect to θ.

GAN framework

The networks have each a cost function C that depend on the parameters of the networks. The discriminator wishes to minimize $C^{(D)}(\phi, \theta)$ while only controlling ϕ. On the other hand, the generator wishes to minimize its own cost function $C^{(G)}(\phi, \theta)$ while only controlling θ.
The solution to a game is a Nash equilibrium, which in our case is a tuple (ϕ, θ) which is a local minimum of $C^{(D)}$ with respect to ϕ and a local minimum of $C^{(G)}$ with respect to θ.
If we rewrite it as a zero-sum game, the optimal parameters for the generator are

$$\theta^* = \arg\min_{\theta} \max_{\phi} V(\phi, \theta).$$

Wasserstein GANs

WGANs are a reformulation where the adversarial setup is to minimize the Wasserstein-1 distance $W(p_r(x), p(x|\theta)$ by replacing the adversarial classifier with a 1-Lipschitz adversarial critic $d(\cdot; \phi) : \mathbb{R}^d \to \mathbb{R}$, with

$$
\begin{aligned}
& W(p_r(x), p(x|\theta) \\
& = \sup_{\phi} \left[\mathbb{E}_{\tilde{x} \sim p(x|\theta)}[d(\tilde{x}; \phi)] - \mathbb{E}_{x \sim p_r(x)}[d(x; \phi)] \right] \\
& \equiv \sup_{\phi} \mathcal{L}_W .
\end{aligned}
$$

Wasserstein GANs

$$
\begin{aligned}
& W(p_r(x), p(x|\theta) \\
& = \sup_{\phi} \left[\mathbb{E}_{\tilde{x} \sim p(x|\theta)}[d(\tilde{x}; \phi)] - \mathbb{E}_{x \sim p_r(x)}[d(x; \phi)] \right] \\
& \equiv \sup_{\phi} \mathcal{L}_W .
\end{aligned}
$$

WGAN-GP introduce the concept of gradient penalty to stabilize the process. Here, ϕ and θ are updated to respectively minimize

$$
\begin{aligned}
\mathcal{L}_d &= \mathcal{L}_W + \lambda \mathbb{E}_{\hat{x} \sim p(\hat{x})}[(\|\nabla_{\hat{x}} d(\hat{x}; \phi)\|_2 - 1)^2], \\
\mathcal{L}_g &= -\mathcal{L}_W,
\end{aligned}
$$

where $\hat{x} := \epsilon x + (1 - \epsilon)\tilde{x}$, for $\epsilon \sim U[0, 1]$, $x \sim p_r(x)$ and $\tilde{x} \sim p(x|\theta)$.

Contents

Variational optimization

Assume we want to minimize a non-differentiable function f. We know that

$$\min_{\theta} f(\theta) \leq \mathbb{E}_{\theta \sim q(\theta|\psi)}[f(\theta)] = U(\psi),$$

where $q(\theta, \psi)$ is a proposal distribution with parameters ψ over the input values θ. The problem then translate to update the parameters ψ so that the proposal distribution is arbitrarily close to the optimum θ^*.

Variational optimization

Under restrictions, the bound $U(\psi)$ is differentiable with respect to ψ, and its gradient can be written as

$$\nabla_\psi U(\psi) = \mathbb{E}_{\theta \sim q(\theta|\psi)}[f(\theta) \nabla_\psi \log q(\theta|\psi)],$$

which can be used to use any kind of gradient optimization method to find the best ψ.

Contents

Adversarial Variational Optimization

The idea of the AVO is to use the simulator that has a model with meaningful parameters θ as the generator instead of training a neural network. If we would apply this directly to the WGAN-GP model, we would encounter the problem that the generator in most of the cases is non-differentiable. But we have just learned how to deal with this by using variational optimization. This means that we can define a proposal distribution $q(\theta|\psi)$ for the parameters and can define the bounds

$$U_d = \mathbb{E}_{\theta\sim q(\theta|\psi)}[\mathcal{L}_d], \tag{5}$$

$$U_g = \mathbb{E}_{\theta\sim q(\theta|\psi)}[\mathcal{L}_g]. \tag{6}$$

and use the method described before to compute the gradient to optimize the parameters.