Generative Adversarial Nets

2014 NIPS Workshop on Perturbations, Optimization, and Statistics --- Ian Goodfellow

Generative Adversarial Nets

Mehdi Mirza

Research Scientist
OpenAI
2016 年 3 月 – 至今 (1 年) | 美国 旧

Senior Research Scientist
Google
2015 年 11 月 – 2016 年 3 月 (5 个月

As a member of the Google Brain tea
terms of improving products.
I'm writing a textbook on deep learnir

Research Scientist
Google
2014 年 7 月 – 2015 年 11 月 (1 年 5

During my first year and change at G
generating adversarial examples and
visualizations showing that neural ne
supervised intern Tianqi Chen as he
neural networks, and did many other

Software Engineering Intern
Google
2013 年 6 月 – 2013 年 9 月 (4 个月)

Worked with the Street Smart team to create a deep neural network capable of reading address numbers from Street View imagery. This system was used to add or update the location of over 100 million houses within the first few months of its deployment.

Research Scientist
DeepMind
2016 年 11 月 – 至今 (4 个月) | 英国 伦敦

Ph.D Student
Université de Montréal
2011 年 9 月 – 至今 (5 年 6 个月) | Montréal, Canada

Machine Learning

▸ 1 个项目

Research Intern
Google DeepMind
2015 年 8 月 – 2015 年 12 月 (5 个月) | 英国 伦敦

Software Engineering Intern
Flickr
2014 年 8 月 – 2014 年 11 月 (4 个月) | San Francisco

Work in Vision/Search team.

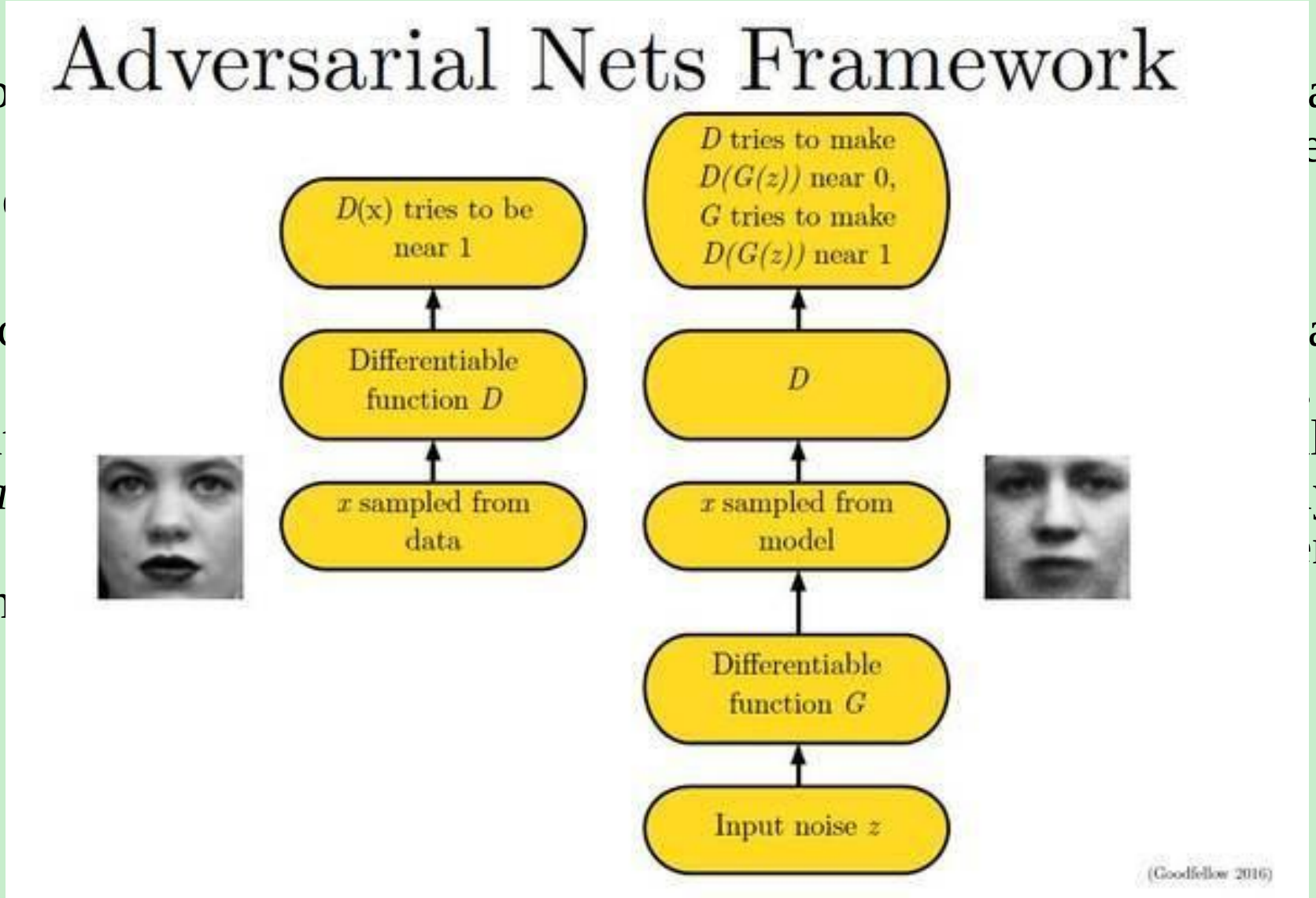
Data Scientist
Atlas Wearables
2013 年 10 月 – 2014 年 3 月 (6 个月)

1 Introduction

1 Introduction

In the prop against an adversary: e is from the model

In this artic ates sample by and the discrimina l case as *adversaria* y successful nerative model usin



2 Related work

Generative stochastic networks [4] are an example of a generative machine that can be trained with exact backpropagation rather than the numerous approximations required for Boltzmann machines .

Such models generally have intractable likelihood functions and therefore require numerous approximations to the likelihood gradient.

These difficulties motivated the development of “generative machines”–models that do not explicitly represent the likelihood, yet are able to generate samples from the desired distribution.

2 Related work

Until recently, most work on deep generative models focused on models that provided a parametric specification of a probability distribution function. The model can then be trained by maximizing the log likelihood. In this family of model, perhaps the most succesful is the deep Boltzmann machine

Such models generally have intractable likelihood functions and therefore require numerous approximations to the likelihood gradient.

These difficulties motivated the development of “generative machines”–models that do not explicitly represent the likelihood, yet are able to generate samples from the desired distribution.

This work extends the idea of a generative machine
by eliminating the Markov chains used in generative stochastic networks

2 Related work

Our work backpropagates derivatives through generative processes by using the observation that

$$\lim_{\sigma \to 0} \nabla_{\boldsymbol{x}} \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})} f(\boldsymbol{x} + \epsilon) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}).$$

We were unaware at the time we developed this work that Kingma and Welling [18] and Rezende et al. [23] had developed more general stochastic backpropagation rules, allowing one to backpropagate through Gaussian distributions with finite variance, and to backpropagate to the covariance parameter as well as the mean.
These backpropagation rules could allow one to learn the conditional variance of the generator, which we treated as a hyperparameter in this work

Kingma and Welling [18] and Rezende *et al.* [23] use stochastic backpropagation to train variational autoencoders (VAEs).

2 Related work

Like generative adversarial networks, variational autoencoders pair a differentiable generator network with a second neural network. Unlike generative adversarial networks, the second network in a VAE is a recognition model that performs approximate inference.

GANs require differentiation through the visible units, and thus cannot model discrete data, while VAEs require differentiation through the hidden units, and thus cannot have discrete latent variables

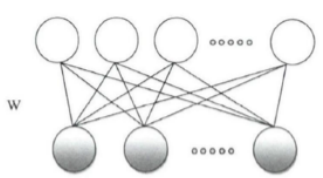
These methods are difficult even to approximate for deep models because they involve ratios of probabilities which cannot be approximated using variational approximations that lower bound the probability .

2 Related work

Some previous work has used the general concept of having two neural networks compete. The most relevant work is **predictability minimization** [26]. In predictability minimization, each hidden unit in a neural network is trained to be different from the output of a second network, which predicts the value of that hidden
hidden units. This work differs from predictabili

1) in this work, the competition between the net
sufficient on its own to train the network. **Predi** **zer**
that **encourages the hidden units of a neural n**
while they accomplish some other task; it is not

Restricted Boltzmann machines



- n_v, n_h
- $\mathbf{v} = (v_1, v_2, \cdots, v_{n_v})^T$:
- $\mathbf{h} = (h_1, h_2, \cdots, h_{n_h})^T$:
- $\mathbf{a} = (a_1, a_2, \cdots, a_{n_v})^T \in \mathbb{R}^{n_v}$:
- $\mathbf{b} = (b_1, b_2, \cdots, b_{n_h})^T \in \mathbb{R}^{n_h}$:
- $W = (w_{i,j}) \in \mathbb{R}^{n_h \times n_v}$

$\forall\, i, j,$ 有 $v_i, h_j \in \{0, 1\}.$

$\theta = (W, \mathbf{a}, \mathbf{b})$

性质 1： 给定可见层神经元的状态时，各隐藏层神经元的激活条件独立，
反之，给定隐藏层神经元的状态时，各可见层神经元的激活也条件独立，

2 Related work

2) The nature of the competition is different.
In **predictability minimization**, two networks' outputs are compared, with one network trying to make the outputs similar and the other trying to make the outputs different. The output in question is a single scalar.
In **GANs**, one network produces a rich, high dimensional vector that is used as the input to another network, and attempts to choose an input that the other network does not know how to process.

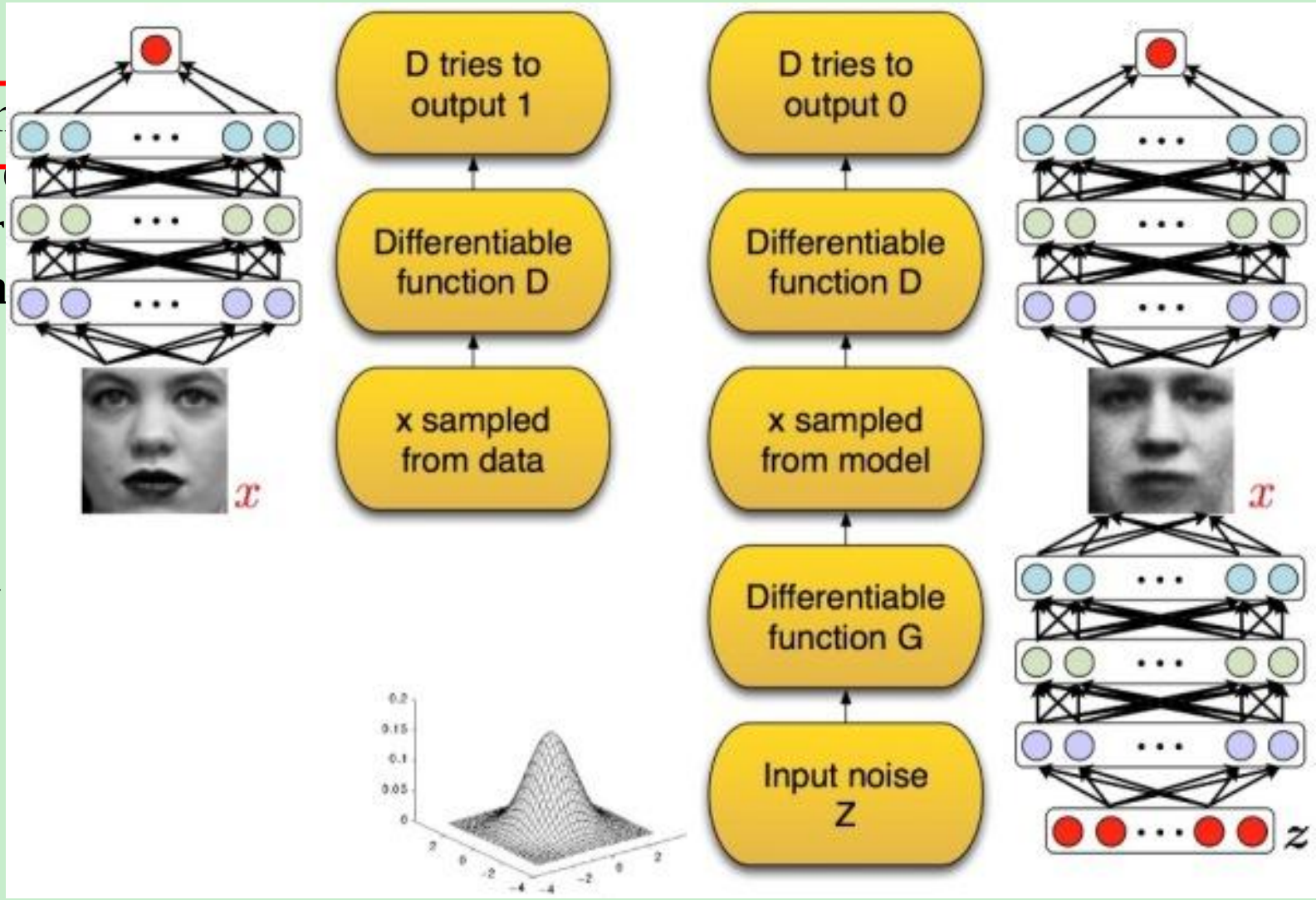
3) The **specification** of the learning process is different. Predictability minimization is described as an optimization problem with an objective function to be minimized, and learning approaches the minimum of the objective function.
GANs are based on a minimax game rather than an optimization problem, and have a **value function** that one agent seeks to maximize and the other seeks to minimize.
The game terminates at a **saddle point**（鞍点） that is a minimum with respect to one player's strategy and a maximum with respect to the other player's strategy.

3 Adversarial Net

To learn the generator's distributio
$p_z(z)$, then represent a mapping to
by a multilayer perceptron with par
$D(\boldsymbol{x}; \theta_d)$ that outputs a **single scala**
data rather than p_g .

1) We train D to maximize the probability of assigning the correct label to both training examples and samples from G.

2) We simultaneously train G to minimize $\log(1-D(G(z)))$.

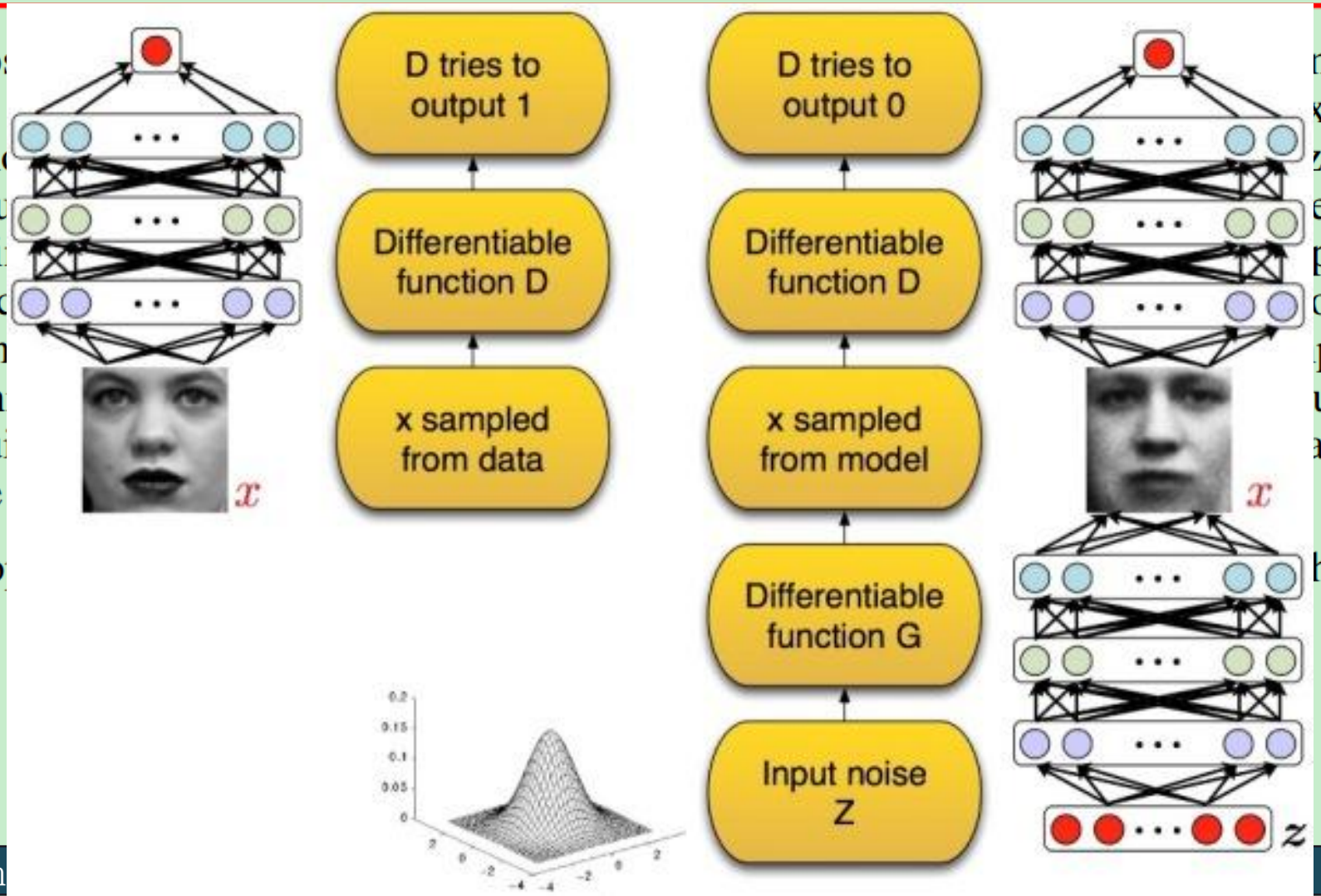


3 Adversarial Net

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \quad (1)$$

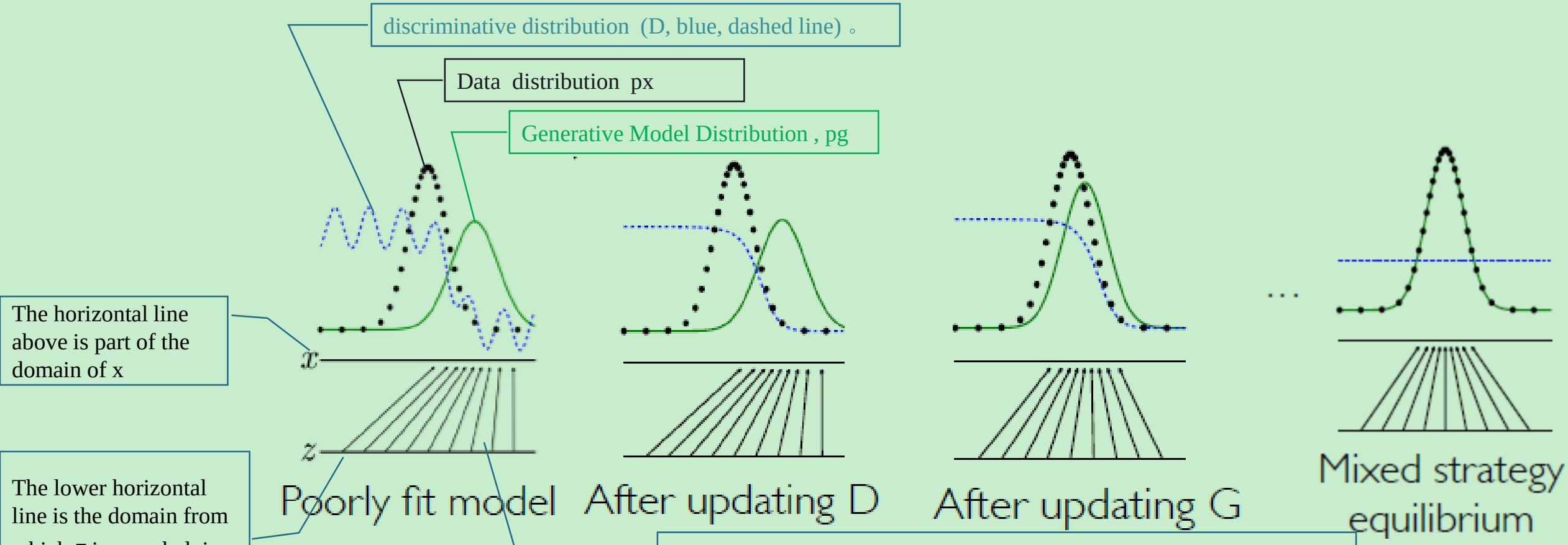
Until recently, mos metric
specification of a ximiz-
ing the log likelih zmann
machine [25]. Su equire
numerous approxi pment
of "generative mac o gen-
erate samples fron ple of
a generative mach us ap-
proximations requ achine
by eliminating the

Our work backpro hat



3 Adversarial Net

See Figure 1 for a less formal, more pedagogical explanation of the approach.



3 Adversarial Net

See Figure 1 for a less formal, more pedagogical explanation of the approach.

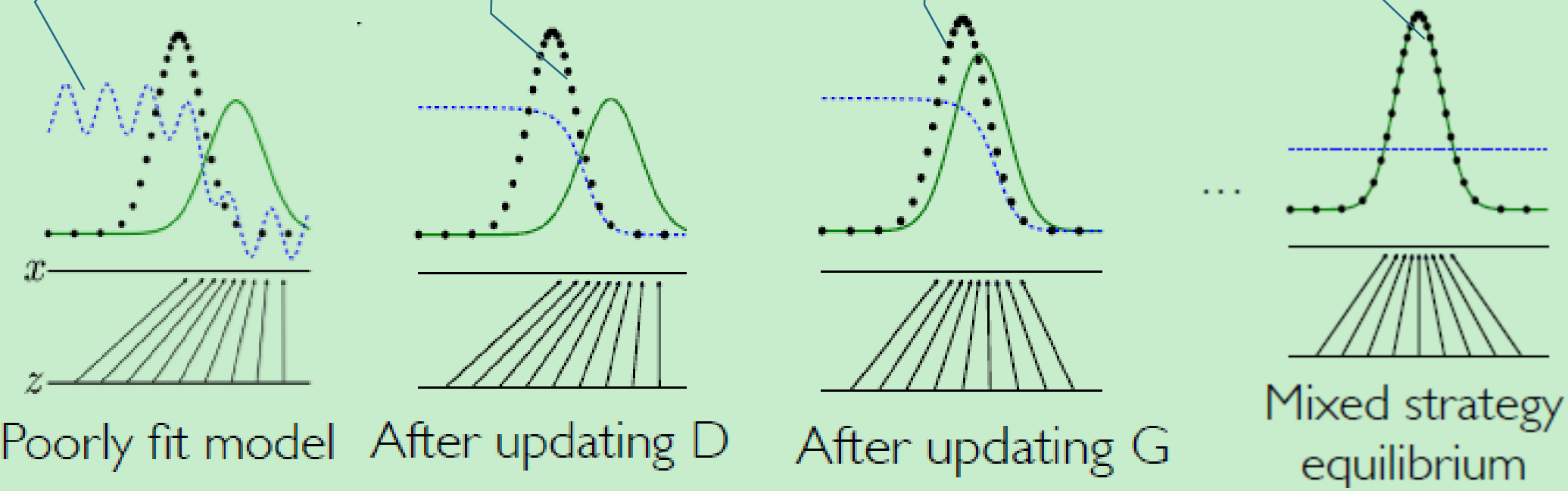
Consider an adversarial pair near convergence : pg is similar to pdata andDis a partially accurate classifier.

In the inner loop of the algorithm D is trained to discriminate samples from data, converging to

$$D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

After an update to G, gradient of D has guided G(z) to flow to regions that are more likely to be classified as data

fter several steps of training, if GandDhave enough capacity, they will reach a point at which both cannot improve because pg = pdata. The discriminator is unable to differentiate between the two distributions, i.e. D(x) = 1/2.



3 Adversarial Net

we alternate be esults in D being maintain he procedure is formally pres

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log D(G(\boldsymbol{z}))$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

4 Theoretical Results

The generator G implicitly defines a probability distribution p_g as the distribution of the samples $G(z)$ obtained when $z \sim p_z$.

theoretical properties
(assuming infinite data ,infinite model capacity direct updating of generator's distribution)

1. Unique global optimum.
2. Optimum corresponds to data distribution.
3. Convergence to optimum guaranteed.

4 Theoretical Results

4.1 **Global Optimality of** $p_g = p_{data}$

We first consider the optimal discriminator D for any given generator G

Proposition 1. For G fixed, the optimal discriminator D is: $D^*_G(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$

$$V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned} \tag{3}$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \to a\log(y) + b\log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $Supp(p_{\text{data}}) \cup Supp(p_g)$, concluding the proof. □

4 Theoretical Results

Theorem 1. *The global minimum of the training criterion C(G) is achieved if and only if* $p_g = p_{data}$ *. At that point, C(G) achieves the value* (- log 4).

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} [\log(1 - D_G^*(G(\boldsymbol{z})))] \qquad (4) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \right] + \mathbb{E}_{\boldsymbol{x} \sim p_g} \left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \right]
\end{aligned}
$$

Proof. For $p_g = p_{\text{data}}$, $D_G^*(\boldsymbol{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D_G^*(\boldsymbol{x}) = \frac{1}{2}$, we find $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

$$\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [-\log 2] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [-\log 2] = -\log 4$$

4 Theoretical Results

4.2 Convergence of Algorithm 1

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data}

Proof. Consider $V(G, D) = U(p_g, D)$ as a function of p_g as done in the above criterion. Note that $U(p_g, D)$ is convex in p_g. The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. In other words, if $f(x) = \sup_{\alpha\in\mathcal{A}} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x) \in \partial f$ if $\beta = \arg\sup_{\alpha\in\mathcal{A}} f_\alpha(x)$. This is equivalent to computing a gradient descent update for p_g at the optimal D given the corresponding G. $\sup_D U(p_g, D)$ is convex in p_g with a unique global optima as proven in Thm 1, therefore with sufficiently small updates of p_g, p_g converges to p_x, concluding the proof. □

4 Theoretical Results

In practice, adversarial nets represent a limited family of p_g distributions via the function $G(\boldsymbol{z}; \theta_g)$, and we optimize θ_g rather than p_g itself, so the proofs do not apply. However, the excellent performance of multilayer perceptrons in practice suggests that they are a reasonable model to use despite their lack of theoretical guarantees.

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	$\mathbf{2110 \pm 50}$
Deep GSN [5]	214 ± 1.1	1890 ± 29
Adversarial nets	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

Table 1: Parzen window-based log-likelihood estimates. The reported numbers on MNIST are the mean log-likelihood of samples on test set, with the standard error of the mean computed across examples. On TFD, we computed the standard error across folds of the dataset, with a different σ chosen using the validation set of each fold. On TFD, σ was cross validated on each fold and mean log-likelihood on each fold were computed. For MNIST we compare against other models of the real-valued (rather than binary) version of dataset.

Thank

You