Lecture 21 — Generative Adversarial Networks (GANs).

Alex Schwing and Matus Telgarsky

April 10, 2018

Schedule for today.

1. Sampling with neural networks.
2. Original GAN formulation; Jensen-Shannon divergence.
3. Wasserstein GAN.
4. Some applications.

Readings.

- *(Original GAN paper.)* Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron C., and Bengio, Yoshua. "Generative adversarial nets". *NIPS*, 2014.
- (Wasserstein GAN papers.)
  - Arjovsky, Martin, Chintala, Soumith, and Bottou, Leon. "Wasserstein generative adversarial networks". *ICML*, 2017.
  - Gulrajani, Ishaan, Ahmed, Faruk, Arjovsky, Martin, Dumoulin, Vincent, and Courville, Aaron C. "Improved training of wasserstein gans". *NIPS*, 2017.

Sampling with neural networks.

Sampling with neural networks.

Generative adversarial networks (GANs) are a way to approximately sample from a distribution with neural networks.

1. Obtain a sample $(x_i)_{i=1}^n$ from the distribution.
2. Use **adversarial training** to fit a neural network g to this sample. Specifically, perform the following alternating minimization.
   2.1 Generate fake sample $(\tilde{x}_i)_{i=1}^m$ from g, and train another network f, the *adversary/discriminator*, to distinguish $(x_i)_{i=1}^n$ and $(\tilde{x}_i)_{i=1}^m$.
   2.2 Now leave f fixed, and train g so that f no longer distinguishes fake and true samples.

Sampling with neural networks.

Generative adversarial networks (GANs) are a way to approximately sample from a distribution with neural networks.

1. Obtain a sample $(x_i)_{i=1}^n$ from the distribution.
2. Use **adversarial training** to fit a neural network g to this sample. Specifically, perform the following alternating minimization.
    2.1 Generate fake sample $(\tilde{x}_i)_{i=1}^m$ from g, and train another network f, the *adversary/discriminator*, to distinguish $(x_i)_{i=1}^n$ and $(\tilde{x}_i)_{i=1}^m$.
    2.2 Now leave f fixed, and train g so that f no longer distinguishes fake and true samples.

Immediate questions.

1. What is the objective function?
2. How can networks generate data? (We'll discuss this now.)

Generating data with neural networks.

Here is the approach used in GANs.

1. First sample $z \sim \mu$ from some efficiently-sampled distribution μ; e.g., Gaussian or uniform.
2. Now output $g(z)$, where g is a neural network.

Generating data with neural networks.

Here is the approach used in GANs.

1. First sample $z \sim \mu$ from some efficiently-sampled distribution μ; e.g., Gaussian or uniform.
2. Now output $g(z)$, where g is a neural network.

Notation: new distribution is $g\#\mu$.

Generating data with neural networks.

Here is the approach used in GANs.

1. First sample $z \sim \mu$ from some efficiently-sampled distribution μ; e.g., Gaussian or uniform.
2. Now output $g(z)$, where g is a neural network.

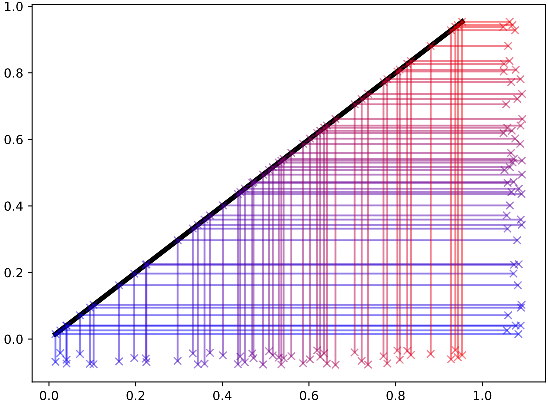
Notation: new distribution is $g\#\mu$.

Remark.

- This is a way to sample, but how to estimate *probabilities*?
  - If we could invert g (in the set-valued sense), then $\Pr[g^{-1}(S)]$ would make sense, where $\Pr[\cdot]$ measures probability according to z.
  - Another approach is to use a kernel density estimate; this was used in the original GAN paper, but has a bad curse of dimension.
  - Basically, though: no one knows.

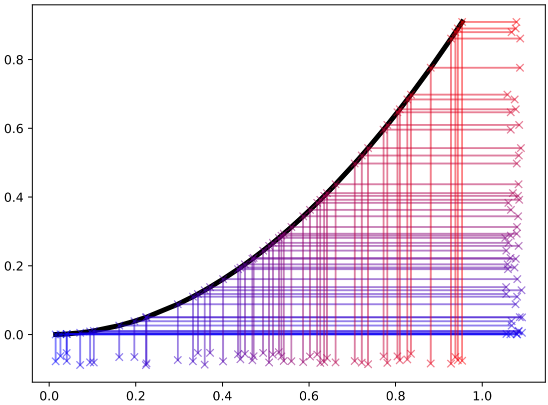
Examples — mappings.

$g(x) = x$, the identity function; target distribution Uniform$([0, 1])$.



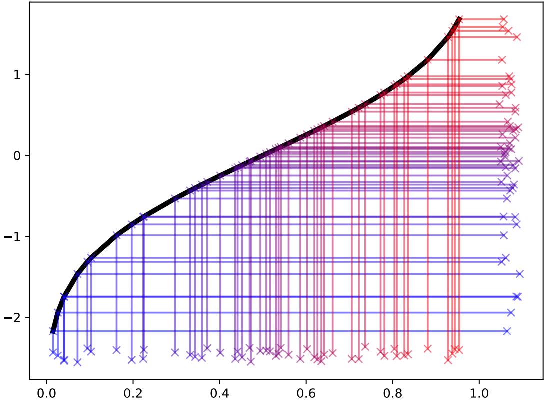
Examples — mappings.

$g(x) = x^2$; target density proportional to $\frac{2}{\sqrt{x}}$ along $(0, 1]$.



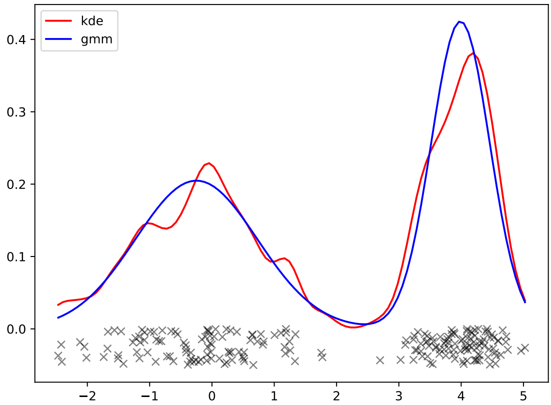
Examples — mappings.

$g(x)$ is inverse CDF of Gaussian; target distribution is Gaussian.



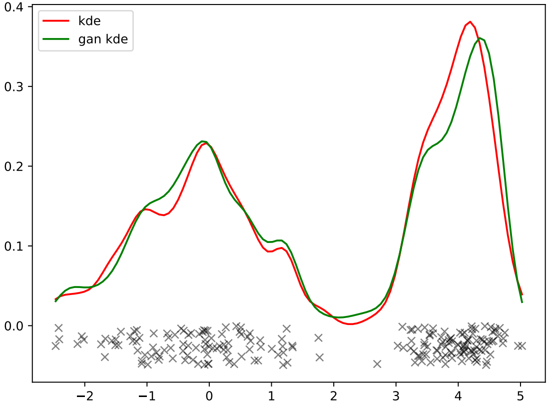
Examples — univariate sampling.

Univariate sample, kernel density estimate (kde), GMM E-M.



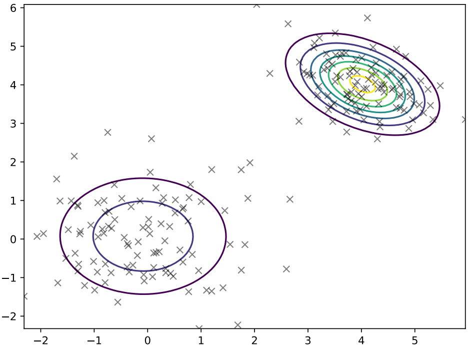
Examples — univariate sampling.

Univariate sample, kernel density estimate (kde), GAN kde.



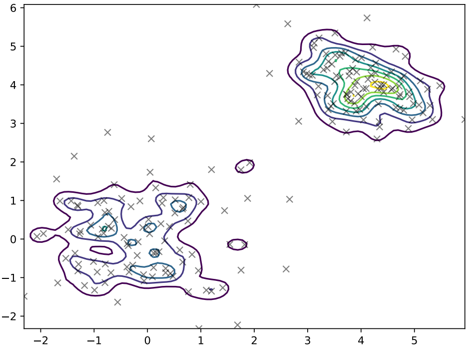
Examples — bivariate sampling.

Bivariate sample, GMM E-M.



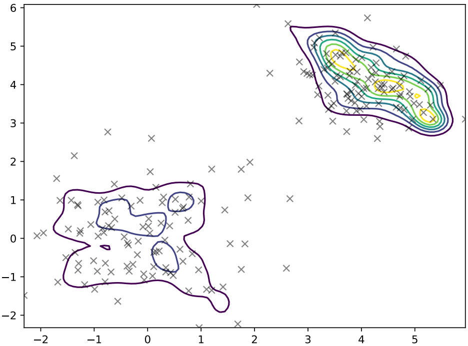
Examples — bivariate sampling.

Bivariate sample, kernel density estimate (kde).



Examples — bivariate sampling.

Bivariate sample, GAN kde.



Question: how will this plot change with network capacity?

Original GAN formulation.

Let's now discuss the choice of objective function.

Original GAN formulation.

Let's now discuss the choice of objective function.
Recall the GAN setup:

- True sample $(x_i)_{i=1}^n$, fake sample $(\tilde{x}_j)_{j=1}^m = (g(z_j))_{j=1}^m$.
- We want these to look similar.

How to enforce similarity? One idea is to look at

$$\inf_{g \in \mathcal{G}} \sup_{f \in \mathcal{F}} \mathbb{E}(f(X)) - \mathbb{E}(f(g(Z)))$$

where $\mathcal{F}$ is a big set of *adversaries/discriminators/critics*.

Original GAN formulation.

Let's now discuss the choice of objective function.
Recall the GAN setup:

- True sample $(x_i)_{i=1}^n$, fake sample $(\tilde{x}_j)_{j=1}^m = (g(z_j))_{j=1}^m$.
- We want these to look similar.

How to enforce similarity? One idea is to look at

$$\inf_{g \in \mathcal{G}} \sup_{f \in \mathcal{F}} \mathbb{E}(f(X)) - \mathbb{E}(f(g(Z))$$

where $\mathcal{F}$ is a big set of *adversaries/discriminators/critics*.

Remarks.

- Suppose $\mathcal{F}$ is all polynomials. Then we get value 0 if true and fake data agree on all moments.
- If g is fixed, can optimize f over some neural nets $\mathcal{F}$. Similarly, can optimize g while holding f fixed.

A transformation.

Now consider an adjustment, used in the original gan paper:

$$\inf_{g \in \mathcal{G}} \sup_{\substack{f \in \mathcal{F} \\ f : X \to (0,1)}} \frac{1}{n} \sum_{i=1}^{n} \ln \left(f(x_i) \right) + \frac{1}{m} \sum_{j=1}^{m} \ln \left(1 - f(g(z_j)) \right)$$

Remarks.

- Interpret f as a probability;
  e.g., $\Pr[x \text{ is fake}]$.
  Then g is doing well if $f = 1/2$ forced everywhere.
  (We'll do this systematically in a moment.)

Original GAN formulation and algorithm.

Original GAN objective:

$$\inf_{g \in \mathcal{G}} \sup_{\substack{f \in \mathcal{F} \\ f: X \to (0,1)}} \frac{1}{n} \sum_{i=1}^{n} \ln\left(f(x_i)\right) + \frac{1}{m} \sum_{j=1}^{m} \ln\left(1 - f(g(z_j))\right)$$

Algorithm alternates these two steps:

1. Hold g fixed and optimize f. Specifically, generate a sample $(\tilde{x}_j)_{j=1}^m = (g(z_j))_{j=1}^m$, and approximately optimize

$$\sup_{\substack{f \in \mathcal{F} \\ f: X \to (0,1)}} \frac{1}{n} \sum_{i=1}^{n} \ln\left(f(x_i)\right) + \frac{1}{m} \sum_{j=1}^{m} \ln\left(1 - f(\tilde{x}_j)\right).$$

2. Hold f fixed and optimize g. Specifically, generate $(z_j)_{j=1}^m$ and approximately optimize

$$\inf_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \ln\left(f(x_i)\right) + \frac{1}{m} \sum_{j=1}^{m} \ln\left(1 - f(g(z_j))\right).$$

Some implementation issues.

Algorithm alternates these two steps:

1. Hold g fixed and optimize f. Specifically, generate a sample $(\tilde{x}_j)_{j=1}^m = (g(z_j))_{j=1}^m$, and approximately optimize

$$\sup_{\substack{f \in \mathcal{F} \\ f: X \to (0,1)}} \frac{1}{n} \sum_{i=1}^{n} \ln\left(f(x_i)\right) + \frac{1}{m} \sum_{j=1}^{m} \ln\left(1 - f(\tilde{x}_j)\right).$$

2. Hold f fixed and optimize g. Specifically, generate $(z_j)_{j=1}^m$ and approximately optimize

$$\inf_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \ln\left(f(x_i)\right) + \frac{1}{m} \sum_{j=1}^{m} \ln\left(1 - f(g(z_j))\right).$$

Remarks.

- Common practice: do many f ascents for each g descent.
- Training has all sorts of instabilities and heuristics fixes; e.g., **mode collapse** (g output a small set of training elements).

Original GAN objective function.

Original GAN objective function.

We can interpret the original GAN as minimizing a divergence/distance between probability distributions.
(In particular, with no adversary.)

Original GAN objective function.

We can interpret the original GAN as minimizing a divergence/distance between probability distributions.
(In particular, with no adversary.)

For this part, assume the following densities exist:

- p_X, a density generating the data $(x_i)_{i=1}^n$.
- p_Z, the density generating the input z to the generator network.
- p_g, the density of the fake distribution of generator network g.

Some other technical caveats will be mentioned.

Jensen-Shannon Divergence.

Proposition.

- Given generator g, the optimal discriminator is $f_g(x) := {}^{p_X(x)}/_{(p_g(x)+p_X(x))}$.
- If discriminator set $\mathcal{F}$ is all functions, the GAN objective is

$$\sup_{f \in \mathcal{F}} \mathbb{E}(\ln f(X)) + \mathbb{E}(\ln(1 - f(g(Z)))) = 2 \cdot \mathrm{JS}(p_X, p_g) - \ln 4,$$

  where JS is the *Jensen-Shannon divergence*: defining $p := {}^{(p_X+p_g)}/_2$,

$$\begin{aligned} 2 \cdot \mathrm{JS}(p_X, p_g) &= \mathrm{KL}(p_X, p) + \mathrm{KL}(p_g, p) \\ &= \int p_X(x) \ln \frac{p_X(x)}{p(x)} \, \mathrm{d}x + \int p_g(x) \ln \frac{p_g(x)}{p(x)} \, \mathrm{d}x. \end{aligned}$$

- The optimal p_g satisfies $p_g = p_X$.

Remarks.

- We can make this the starting point for the original GAN derivation.
  1. We want to find g so that $\rho(p_g, p_X)$ is small for some reasonable notion of distance ρ.
  2. We choose Jensen-Shannon divergence for ρ. We could have chosen something else.
- From this perspective, the 2-player game view is a consequence which is used to derive a *training algorithm*.
- As with k-means/E-M, we massaged the objective to add another variable, and trained with alternating minimization.

Proof of optimal discriminator.

Using the assumed densities p_X, p_Z, p_g,

$$
\begin{aligned}
&\mathbb{E}\ln f(X) + \mathbb{E}\ln(1 - f(g(Z))) \\
&= \int \ln f(x) p_X(x)\,\mathrm{d}x + \int \ln(1 - f(g(z))) p_Z(z)\,\mathrm{d}z \\
&= \int \ln f(x) p_X(x)\,\mathrm{d}x + \int \ln(1 - f(x)) p_g(x)\,\mathrm{d}x. \\
&= \int \Big(\ln f(x) p_X(x) + \ln(1 - f(x)) p_g(x)\Big)\,\mathrm{d}x.
\end{aligned}
$$

Since f can be any function, we can maximize it pointwise. Note $r \mapsto a\ln(r) + b\ln(1-r)$ is concave with maximum $a/(a+b)$.

Therefore, optimal discriminator satisfies $f(x) = \frac{p_X(x)}{p_X(x)+p_g(x)}$.

Proof of alternate form.

Plugging this back in,

$$
\begin{aligned}
&\sup_{f\in\mathcal{F}} \mathbb{E}\ln f(X) + \mathbb{E}\ln(1-f(g(Z))) \\
&= \sup_{f\in\mathcal{F}} \int \Big(\ln f(x) p_X(x) + \ln(1-f(x)) p_g(x)\Big)\,\mathrm{d}x. \\
&= \int \left(p_X(x)\ln\frac{p_X(x)}{p_X(x)+p_g(x)} + p_g(x)\ln\frac{p_g(x)}{p_X(x)+p_g(x)}\right)\mathrm{d}x. \\
&= \int \left(p_X(x)\ln\frac{2p_X(x)}{p_X(x)+p_g(x)} + p_g(x)\ln\frac{2p_g(x)}{p_X(x)+p_g(x)}\right)\mathrm{d}x - \ln 4 \\
&= \mathrm{KL}\left(p_X, \frac{p_X+p_g}{2}\right) + \mathrm{KL}\left(p_g, \frac{p_X+p_g}{2}\right) - \ln 4.
\end{aligned}
$$

Technical remarks.

- This derivation is over the **true** distribution, not the sample! The sample induces a **discrete** distribution!
  - How to regularize/generalize?
  - The optimum of memorizing training set is trivial and doesn't need a GAN to train (just randomly sample the training set).
- The discriminator need only satisfy the stated equality with probability 1.
- By this derivation, mode collapse is not baked into the objective function; it is a side effect of training (e.g., non-convexity).
- The analysis needs $\mathcal{F}$ to be all possible functions. But in general we use some restricted/regularized set of neural networks. What is the corresponding optimal discriminator?

Technical remarks.

- The optimality condition on the generator, $p_g = p_X$, is a consequence of strict concavity of ln inside Jensen-Shannon divergence.
- Similarly to the optimal discriminator equation, this only holds with probability 1, and requires the generator set $\mathcal{G}$ to be everything (in general).
- There are many standard choices for the (restricted/regularized) set of generators. One is the "DCGAN".

Technical remarks.

- Rather than arguing pointwise, the optimal discriminator can be found by variants of "take gradient of objective, set to zero".
  - A variant is needed because the optimization variable f, is an arbitrary function, not just a vector.
  - One such variant is mentioned in the homework. Namely, the "Euler-Lagrange equation". This equation is for objectives that possess not only f, but its derivative. Namely, let $\int L(x, f, f')\,\mathrm{d}x$ denote the GAN objective. A corresponding optimality condition is

    $$\frac{\partial L(x, f, f')}{\partial f} - \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial L(x, f, f')}{\partial f'} = 0.$$

    Since f' does not appear, the second term is zero, and this equation becomes

    $$\frac{p_X}{f} - \frac{p_g}{1-f} = 0,$$

    which gives $f = {}^{p_X}\!/\!_{(p_g+p_X)}$ as before.
  - There are other versions of "take derivative and set to 0" in function spaces which can be used (e.g., ones without f').

Wasserstein GAN (WGAN).

Wasserstein GAN (WGAN).

Let's build another GAN around another objective function.

Let's start from the 2-player perspective again.

WGAN: 2-player formulation.

Recall the original abstract adversary optimization:

$$\inf_{g \in \mathcal{G}} \sup_{f \in \mathcal{F}} \mathbb{E}(f(X)) - \mathbb{E}(f(g(Z)).$$

- The Wasserstein GAN imposes a specific constraint: $\|f\|_{\text{Lip}} \leq 1$: the set of functions with Lipschitz constant less than 1 (this means $\sup_{x \neq y} \frac{f(x)-f(y)}{\|x-y\|} \leq 1$, and for differentiable functions means $|f'| \leq 1$).
- In practice, $\mathcal{F}$ is a set of neural nets plus some regularization/constraints aiming to enforce $\|f\|_{\text{Lip}} \leq 1$. It's not clear how best to do this; the two WGAN papers mentioned in the readings do it differently.

WGAN: non-adversarial objective.

Due to the Lipschitz constraint, the objective is the Wasserstein-1 distance; here is a special case for densities p_g, p_X as before:

$$
\begin{aligned}
W(p_g, p_X) &= \sup_{\|f\|_{\mathrm{Lip}} \leq 1} \left(\int f(x) p_X(x)\, \mathrm{d}x - \int f(x) p_g(x)\, \mathrm{d}x \right) \\
&= \inf \left\{ \int \|x - h(x)\| p_g(x)\, \mathrm{d}x \ : \ h \# p_g = p_X \right\}
\end{aligned}
$$

- This is called the **Kantorovich-Rubinstein** duality.
- The meaning of the last expression is: if we sample $x \sim p_g$ and output $h(x)$, this is the same as sampling p_X.
- This distance is also called "Earth mover's distance" because it can be interpreted as moving the mass from one distribution into the shape of another, so that particles are moved a minimal distance. (Pictures in class.)

Summary.

- Generating random samples with neural networks.
- Original GAN.
  - Objective function.
  - Alternating optimization (min/max).
  - The optimal choices (supposing all possible discriminators/generators).
- Wasserstein GAN
  - Objective function.