We are going to talk about Generative Adversarial Networks.

Reading:
- Chapter 20. of the Deep Learning book
- Techniques for training GANs https://arxiv.org/abs/1606.03498

For different models you will find URLs for relevant papers in the slides.
Generative models are aimed at capturing a data distribution

\[ p(x) = \int_z p(x|z)p(z)dz \]

where \( z \) are latent variables, for example, codes. However computation of the marginal probability is challenging. Recall the RBM training in your homework.

We are going to look at a different framework.
What is a good generative model

In our ML intro we have shown that log-likelihood maximization can be seen as minimizing KL divergence between empirical distribution

\[
\hat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} [x = x^i]
\]

and the model \( p(x) \)

\[
\min_{\theta} KL(\hat{p}(x) || p(x|\theta)) = \int \hat{p}(x) \log p(x|\theta) dx + \int \hat{p}(x) \log \hat{p}(x) dx
\]

\[
\begin{align*}
\text{does not depend on } \theta
\end{align*}
\]

In effect, the model distribution and data distribution have to be close.
Suppose we have a generating network that can take input code $z$ and generate a data sample $G(z)$.

What if we had an algorithm $D$ that could tell us if $G(z)$ is like the true data?

Q: How would you use $D$ in order to train $G$?

Q: How would you obtain $D$?
Adversarial networks approach pits two networks, generator and another discriminator, against each other.

**Generator** generates new data from codes $z \rightarrow G(z)$

**Discriminator** tries to discriminate between the true data (training set) and the generated data (output of generator).

The training is successful when discriminator is no longer able to tell generated samples from true data.
Setting up notation

\( p_{\text{Data}} \) is data generating distribution.

\( p_{\text{Data}} \) is the distribution we are trying to learn.

\( p_{\text{Data}} \) is not the empirical distribution of the training data

\[
p_{\text{Data}}(x) \neq \hat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} [x = x^i],
\]

although they are close.

\( p_z(z) \) denote distribution over codes, for example \( \mathcal{N}(0, 1_p) \).

\( p_g(x) \) denote distribution over generated data. Learning succeeds when \( p_g = p_{\text{Data}} \)

We will be very explicit in expressing expectations

\[
E_{y \sim p(y)} [f(y)] = \int p(y)f(y)dy
\]
Coming up with an objective

Generator network $G$ generates a new synthetic sample from code $z$.

Discriminator networks $D$ outputs probability that sample is from the true data.

Discriminator tries to classify real and generated samples

- $x$, real data: $D(x)$ close to 1
- $G(z)$, generated data: $D(G(z))$ close to 0.

Generator tries to generate data so that $D(G(z))$ is close to 1.
Coming up with an objective

Objective for discriminator – maximize log prob of correct classification

$$\max_{D} \mathbb{E}_{x \sim p_{\text{Data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_{z}(z)}[\log 1 - D(G(z))]$$

Objective for generator – minimize log probability of classifying generated samples as fake

$$\min_{G} \mathbb{E}_{z \sim p_{z}(z)}[\log 1 - D(G(z))]$$

Combined

$$\min_{G} \max_{D} \mathbb{E}_{x \sim p_{\text{Data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_{z}(z)}[\log 1 - D(G(z))]$$
Training GANs

The objective

$$\min_G \max_D E_{x \sim p_{\text{Data}}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log 1 - D(G(z))]$$

calls for evaluating expectations over distributions.

In practice, these expectations will be approximated using finite samples

- $p_{\text{Data}}(x)$ will be approximated by empirical distribution of a minibatch
  $$p_{\text{Data}}(x) \approx \frac{1}{M} \sum_{i \in \text{MiniBatch}} [x = x^i]$$

- $p_z(z)$ will be approximated by a finite sample
  $$p_z(z) \approx \frac{1}{M} \sum_{i=1}^{M} [z = z^i]$$

where $z^i$s are randomly drawn from a Gaussian distribution
Hence we can approximate the objective by

$$\min_G \max_D \frac{1}{M} \sum_{i \in \text{MiniBatch}} \left[ \log D(x^i) \right] + \sum_{i=1}^{M} \left[ \log 1 - D(G(z^i)) \right].$$

Every iteration we can resample the mini-batch and $z^i$.  

Training GANs

**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 $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
        • Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
        • Update the discriminator by ascending its stochastic gradient:
          \[
          \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D(x^{(i)}) + \log \left(1 - D\left(G(z^{(i)})\right)\right) \right].
          \n          \]
    end for
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Update the generator by descending its stochastic gradient:
      \[
      \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(G(z^{(i)})\right)\right).
      \]
end for
```

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

Figure 1: Generative adversarial nets are trained by simultaneously updating the discriminative distribution \( D \), blue, dashed line so that it discriminates between samples from the data generating distribution (black, dotted line) \( p_x \) from those of the generative distribution \( p_g \) (G) (green, solid line). The lower horizontal line is the domain from which \( z \) is sampled, in this case uniformly. The horizontal line above is part of the domain of \( x \). The upward arrows show how the mapping \( x = G(z) \) imposes the non-uniform distribution \( p_g \) on transformed samples. \( G \) contracts in regions of high density and expands in regions of low density of \( p_g \). (a) Consider an adversarial pair near convergence: \( p_g \) is similar to \( p_{\text{data}} \) and \( D \) is a partially accurate classifier. (b) In the inner loop of the algorithm \( D \) is trained to discriminate samples from data, converging to \( D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \). (c) 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. (d) After several steps of training, if \( G \) and \( D \) have enough capacity, they will reach a point at which both cannot improve because \( p_g = p_{\text{data}} \). The discriminator is unable to differentiate between the two distributions, i.e. \( D(x) = \frac{1}{2} \).
Proposition 1. For $G$ fixed, the optimal discriminator $D$ is

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

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

$$V(G, D) = \int_x p_{data}(x) \log(D(x))dx + \int_z p_z(z) \log(1 - D(g(z)))dz$$

$$= \int_x p_{data}(x) \log(D(x)) + p_g(x) \log(1 - D(x))dx$$

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 $Supp(p_{data}) \cup Supp(p_g)$, concluding the proof. □
GANs theoretical results

\[ C(G) = \max_D V(G, D) \]
\[ = \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log D^*_G(x) \right] + \mathbb{E}_{z \sim p_z} \left[ \log (1 - D^*_G(G(z))) \right] \]
\[ = \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log D^*_G(x) \right] + \mathbb{E}_{x \sim p_g} \left[ \log (1 - D^*_G(x)) \right] \]
\[ = \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[ \log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right] \]

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

In your implementation, you know that objective cannot go lower than \(-\log 4\)
GANs theoretical results

**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

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

then $p_g$ converges to $p_{data}$.
GAN code interpolation

Figure 3: Digits obtained by linearly interpolating between coordinates in $z$ space of the full model.
Inverting GANs\textsuperscript{1}

How can we obtain $z$ such that $G(z) \approx x$?
There are two or more possible answers.

\textsuperscript{1}https://arxiv.org/pdf/1611.05644.pdf
<table>
<thead>
<tr>
<th></th>
<th>Deep directed graphical models</th>
<th>Deep undirected graphical models</th>
<th>Generative autoencoders</th>
<th>Adversarial models</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training</strong></td>
<td>Inference needed during training.</td>
<td>Inference needed during training. MCMC needed to approximate partition function gradient.</td>
<td>Enforced tradeoff between mixing and power of reconstruction generation</td>
<td>Synchronizing the discriminator with the generator.</td>
</tr>
<tr>
<td><strong>Inference</strong></td>
<td>Learned approximate inference</td>
<td>Variational inference</td>
<td>MCMC-based inference</td>
<td>Learned approximate inference</td>
</tr>
<tr>
<td><strong>Sampling</strong></td>
<td>No difficulties</td>
<td>Requires Markov chain</td>
<td>Requires Markov chain</td>
<td>No difficulties</td>
</tr>
<tr>
<td><strong>Evaluating</strong> $p(x)$</td>
<td>Intractable, may be approximated with AIS</td>
<td>Intractable, may be approximated with AIS</td>
<td>Not explicitly represented, may be approximated with Parzen density estimation</td>
<td>Not explicitly represented, may be approximated with Parzen density estimation</td>
</tr>
<tr>
<td><strong>Model design</strong></td>
<td>Nearly all models incur extreme difficulty</td>
<td>Careful design needed to ensure multiple properties</td>
<td>Any differentiable function is theoretically permitted</td>
<td>Any differentiable function is theoretically permitted</td>
</tr>
</tbody>
</table>

Table 2: Challenges in generative modeling: a summary of the difficulties encountered by different approaches to deep generative modeling for each of the major operations involving a model.
Conditional GANs

If we have a label $y$ we can include it in the model

Deep Convolutional GANs\textsuperscript{3}

If the data we are modeling is high-dimensional such as images, we would not use a fully-connected layers.

Convolution layers here use transposed filters.
Discriminator is a convolutional neural net.

\textsuperscript{3}https://arxiv.org/pdf/1511.06434.pdf
InfoGAN\(^4\)

One issue with plain GANs is that the codes may not be interpretable.

A information theoretic change to the GAN objective

\[
\min_G \max_D V(D, G) = \mathbb{E}_{x \sim P_{data}} [\log D(x)] + \mathbb{E}_{z \sim \text{noise}} [\log (1 - D(G(z)))]
\]

\[
I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)
\]

\[
\min_G \max_D V_I(D, G) = V(D, G) - \lambda I(c; G(z, c))
\]

where \(c\) denotes a subset of the code that needs to have high mutual information with the generated data.

\(^4\)https://arxiv.org/abs/1606.03657
Approximation of the objective using variational lower bound:

$$L_I(G, Q) = E_{c\sim P(c), x\sim G(z, c)}[\log Q(c|x)] + H(c)$$

$$\min_G \max_Q V_{\text{InfoGAN}}(D, G, Q) = V(D, G) - \lambda L_I(G, Q)$$

Expectation and $Q$ are approximated using the reparameterization trick.

As a reminder, reparameterization trick allows us to construct differentiable approximations of expectations

$$E_{y\sim \mathcal{N}(f_{\mu}(t), f_{\sigma}(t)^2)}[r(y)] \approx \frac{1}{N} \sum_{i=1}^{N} r(y^i)$$

$$y^i = f_{\mu}(t) + f_{\sigma}(t) * \epsilon^i, \epsilon^i \sim \mathcal{N}(0, I)$$

\footnote{https://arxiv.org/abs/1606.03657}
Meaningful codes in infoGANs – generated digit data

(a) Varying $c_1$ on InfoGAN (Digit type)  
(b) Varying $c_1$ on regular GAN (No clear meaning)

(c) Varying $c_2$ from $-2$ to $2$ on InfoGAN (Rotation)  
(d) Varying $c_3$ from $-2$ to $2$ on InfoGAN (Width)
Meaningful codes in infoGANs – generated faces data
Meaningful codes in infoGANs – generated chair data

(a) Rotation

(b) Width
We talked about Generative Adversarial Networks

- plain GANs
- conditional GANs
- deep convolutional GANs
- infoGAN

Very rich framework and plenty of cool work left to be done here.