DEEP LEARNING
PART THREE - DEEP GENERATIVE MODELS
GENERATIVE MODELS
**DATA**

<table>
<thead>
<tr>
<th>example 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
example 3

DATA
number of data examples

DATA
<table>
<thead>
<tr>
<th>number of data examples</th>
<th>feature 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DATA
number of data examples

DATA

feature 3
DATA DISTRIBUTION

- Feature 1
- Feature 2
- Feature 3

Examples:
- Example 1
- Example 2
- Example 3
DATA DISTRIBUTION
DENSITY ESTIMATION

estimating the density of the empirically observed data distribution
GENERATIVE MODEL

a model of the density of the data distribution
by modeling the data distribution, generative models are able to **generate** new data examples
discriminative models vs. generative models

can both be trained using supervised learning

generative models are often easier to train with unsupervised methods

generative models typically require more modeling assumptions

straightforward to quantify uncertainty with generative models
one of the main benefits of generative modeling is the ability to automatically extract structure from data reducing the effective dimensionality of the data can make it easier to learn and generalize on new tasks
one of the main benefits of generative modeling is the ability to automatically extract structure from data

reducing the effective dimensionality of the data can make it easier to learn and generalize on new tasks
one of the main benefits of generative modeling is the ability to automatically extract structure from data reducing the effective dimensionality of the data can make it easier to learn and generalize on new tasks
any model that has an output in the data space can be considered a generative model.

nervous systems appear to use this mechanism in part prediction of sensory input using “top-down” pathways.
**deep generative model**

a generative model that uses deep neural networks to model the data distribution
FAMILIES OF (DEEP) GENERATIVE MODELS

- auto-regressive models
- latent variable models
- implicit models
AUTO-REGRESSIVE MODELS
a data example

\[ p(x) = p(x_1, x_2, \ldots, x_M) \]
use \textit{chain rule of probability} to split the joint distribution into a product of conditional distributions

definition of conditional probability
\[ p(a|b) = \frac{p(a, b)}{p(b)} \rightarrow p(a, b) = p(a|b)p(b) \]

recursively apply to \( p(x_1, x_2, \ldots, x_M) \)

\[ p(x_1, x_2, \ldots, x_M) = p(x_1|x_2, \ldots, x_M)p(x_2, \ldots, x_M) \]

\[ \vdots \]

\[ p(x_1, x_2, \ldots, x_M) = p(x_1|x_2, \ldots, x_M)p(x_2|x_3, \ldots, x_M) \ldots p(x_M-1|x_M)p(x_M) \]

\textit{note: conditioning order is arbitrary}

\[ p(x_1, \ldots, x_M) = \prod_{j=1}^{M} p(x_j|x_1, \ldots, x_{j-1}) \]
model the conditional distributions of the data

learn to **auto-regress** to the missing values
model the conditional distributions of the data

learn to **auto-regress** to the missing values

\[ p(x_1) \]
model the conditional distributions of the data

learn to **auto-regress** to the missing values

\[ p(x_2 | x_1) \]
model the conditional distributions of the data

learn to **auto-regress** to the missing values

\[ p(x_3 | x_2, x_1) \]
model the conditional distributions of the data

learn to **auto-regress** to the missing values

\[ p(x_4 | x_3, x_2, x_1) \]
model the conditional distributions of the data

learn to **auto-regress** to the missing values

\[ p(x_5 | x_4, x_3, x_2, x_1) \]
model the conditional distributions of the data

learn to **auto-regress** to the missing values

$$p(x_6|x_5, x_4, x_3, x_2, x_1)$$
model the conditional distributions of the data

learn to \textit{auto-regress} to the missing values

$$p(x_M | x_{M-1}, \ldots, x_1)$$
maximum likelihood

to fit the model to the empirical data distribution, maximize the likelihood of the true data examples

likelihood: \[ p(\mathbf{x}) = \prod_{j=1}^{M} p(x_j | \mathbf{x}_{<j}) \]

optimize the parameters to assign high (log) probability to the true data examples

learning: \[ \theta^* = \arg\max_{\theta} \log p(\mathbf{x}) \]

logarithm for numerical stability

auto-regressive conditionals
models

can parameterize conditional distributions using a **recurrent neural network**

unrolling auto-regressive generation from an RNN

"teacher forcing"

**Deep Learning**, Goodfellow et al., 2016

(chapter 10)
models

can parameterize conditional distributions using a **recurrent neural network**

The Unreasonable Effectiveness of Recurrent Neural Networks, *Karpathy*, 2015

Pixel Recurrent Neural Networks, *van den Oord et al.*, 2016
models can also condition on a local window using convolutional neural networks

PixelCNN

Pixel Recurrent Neural Networks, \textit{van den Oord et al.}, 2016

Conditional Image Generation with PixelCNN Decoders, \textit{van den Oord et al.}, 2016

output distributions

need to choose a form for the conditional output distribution, i.e. how do we express \( p(x_j|x_1, \ldots, x_{j-1}) \)?

model the data as \textbf{categorical} variables

\longrightarrow \textit{multinomial output}

model the data as \textbf{continuous} variables

\longrightarrow \text{Gaussian, logistic, etc. output}
example applications

text

images

occluded
completions
original

Pixel Recurrent Neural Networks, van den Oord et al., 2016

speech

WaveNet: A Generative Model for Raw Audio, van den Oord et al., 2016
recap: auto-regressive models

$$p(x_6|x_5, x_4, x_3, x_2, x_1)$$

model conditional distributions to auto-regress to missing values

**Pros**
- tractable and straightforward to evaluate the (log) likelihood
- great at capturing details
- superior quantitative performance

**Cons**
- difficult to capture "high-level" global structure
- need to impose conditioning order
- sequential sampling is computationally expensive
EXPLICIT \hspace{5mm} \textsc{latent} \hspace{5mm} \textsc{variable} \hspace{5mm} \textsc{models}
reality **generates** sensory stimuli from underlying latent phenomena

**REALITY**
- matter
- energy
- forces
- etc.

**STIMULI**

**PERCEPTION**
- object identities
- object locations
- communication
- etc.

can use *latent variables* to help model these phenomena
probabilistic graphical models provide a framework for modeling relationships between random variables

**PLATE NOTATION**

- **observed variable**
  - \( x \)

- **unobserved (latent) variable**
  - \( x \)

- **directed relationship**
  - \( x \rightarrow y \)

- **undirected relationship**
  - \( x \xrightarrow{} y \)

- **set of variables**
  - \( \{x\} \)
  - \( N \)
review exercise:
represent an auto-regressive model of 3 random variables with plate notation
comparing auto-regressive models and latent variable models

auto-regressive model

latent variable model
example: undirected latent variable model

restricted Boltzmann machine

cut for time
example: **directed** latent variable model
example: **directed latent variable model**

**GENERATIVE MODEL**

\[ p(x, z) = p(x|z)p(z) \]

1. sample \( z \) from \( p(z) \)
2. use \( z \) samples to sample \( x \) from \( p(x|z) \)

**intuitive example**

- object \( \sim p(\text{objects}) \)
- lighting \( \sim p(\text{lighting}) \)
- background \( \sim p(\text{bg}) \)

**GENERATION**

![Diagram of directed latent variable model with nodes for z, x, and N, and arrows indicating the joint, conditional, and prior distributions.](image)
example: directed latent variable model

Posterior Inference

\[ p(z|x) = \frac{p(x,z)}{p(x)} \]

use Bayes’ rule (def. of cond. prob.) provides conditional distribution over latent variables

intuitive example

what is the probability that I am observing a cat given these pixel observations?

\[ p(\text{cat} | \text{observation}) = \frac{p(\text{cat} | \text{observation}) p(\text{cat})}{p(\text{observation})} \]
**example: directed latent variable model**

Model Evaluation

![Diagram](image.png)

\[ p(x) = \int p(x, z) dz \]

*MARGINALIZATION*

to evaluate the likelihood of an observation, we need to *marginalize* over all latent variables

i.e. consider all possible underlying states

**intuitive example**

how likely is this observation under my model? (what is the probability of observing this?)

for all objects, lighting, backgrounds, etc.:

how plausible is this example?
example: **directed latent variable model**

to fit the model, we want to evaluate the *marginal* (log) likelihood of the data

\[
\theta^* = \arg\max_{\theta} \log p(x)
\]

however, this is generally *intractable*, due to the integration over latent variables

\[
p(x) = \int p(x, z) dz
\]

*integration in high-dimensions*
variational inference

**main idea**

instead of optimizing the (log) likelihood, optimize a *lower bound* on it

introduce an *approximate posterior*, then minimize KL-divergence to the true posterior

\[
q^*(z|x) = \arg\min_q D_{KL}(q(z|x)||p(z|x))
\]

evaluating KL-divergence involves evaluating \(p(z|x)\), instead maximize \(\mathcal{L}\):

\[
q^*(z|x) = \arg\max_q \mathcal{L}
\]

where \(\mathcal{L}\) is the evidence lower bound (ELBO), defined as \(\mathcal{L} \equiv \mathbb{E}_{z \sim q(z|x)} [\log p(x, z) - \log q(z|x)]\)

\(\mathcal{L}\) provides a lower bound on \(\log p(x)\), so we can use \(\mathcal{L}\) to (approximately) fit the model

\[
\tilde{\theta}^* = \arg\max_{\theta} \mathcal{L}
\]
interpreting the ELBO

we can write the ELBO as

\[ L \equiv \mathbb{E}_{z \sim q(z|x)} [\log p(x, z) - \log q(z|x)] \]

\[ = \mathbb{E}_{z \sim q(z|x)} [\log p(x|z)p(z) - \log q(z|x)] \]

\[ = \mathbb{E}_{z \sim q(z|x)} [\log p(x|z) + \log p(z) - \log q(z|x)] \]

\[ = \mathbb{E}_{z \sim q(z|x)} [\log p(x|z)] - D_{KL}(q(z|x)||p(z)) \]

\[ \text{reconstruction} \quad \text{regularization} \]

\( q(z|x) \) is optimized to represent the data while staying close to the prior

many connections to compression, information theory

resembles the “auto-encoding” framework
variational **inference** involves optimizing the approximate posterior for each data example

\[
q^*(z|x) = \arg\max_q \mathcal{L}
\]

can be solved using gradient ascent and (stochastic) backpropagation / REINFORCE, but can be computationally expensive

can instead **amortize** inference over data examples by learning a separate **inference model** to output approximate posterior estimates

"**variational auto-encoder**"
hierarchical latent variable models

A Recurrent Latent Variable Model for Sequential Data, Chung et al., 2015

Learning Hierarchical Features from Generative Models, Zhao et al., 2017

Improving Variational Inference with Inverse Auto-regressive Flow, Kingma et al., 2016

sequential latent variable models

Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data, Karl et al., 2016

Learning to Infer, Marino et al., 2017

iterative inference models
introducing latent variables to a generative model generally makes evaluating the (log) likelihood intractable

need to consider all possible “hypotheses” to evaluate (marginal) likelihood of the “outcome”
change of variables

under certain conditions, we can use the change of variables formula to exactly evaluate the log likelihood

consider a variable in one dimension: $x \sim \text{Uniform}(0, 1)$

then let $y$ be an affine transformation of $x$, e.g. $y = 2x + 1$.

to conserve probability mass, $p(y) = p(x) \left| \frac{dx}{dy} \right|$
change of variables

in higher dimensions, conservation of probability mass generalizes to

\[
p(y) = p(x) \left| \det \frac{dx}{dy} \right| = p(x) |\text{det } J^{-1}| \\
\]

where \( J \) is the Jacobian matrix of the transformation, \( J = \frac{dy}{dx} \)

\( |\text{det } J^{-1}| \) expresses the local distortion in volume from the linear transformation

"law of the unconscious statistician" (LOTUS)

can evaluate the probability from one variable’s distribution by evaluating the probability of a transformed variable and the volume transformation

for certain classes of transformations, this is \textit{tractable} to evaluate
change of variables

to use the change of variables formula, we need to evaluate $|\det J^{-1}|$

for an arbitrary $N \times N$ Jacobian matrix, this is worst case $O(N^3)$

restricting the transformations to those with diagonal or triangular inverse Jacobians allows us to compute $|\det J^{-1}|$ in $O(N)$. 

$\rightarrow$ product of diagonal entries
change of variables

can transform the data into a space that is easier to model

Density Estimation Using Real NVP, Dinh et al., 2016
change of variables for variational inference: \textit{normalizing flows}

use more complex approximate posterior, but evaluate a simpler distribution

\[
q(z|x)
\]

transform \( q(z|x) \)

chain together multiple transforms to get more expressive model

\[
\text{target distribution:}
\]

\text{transforms}
transforms

additive coupling layer  Dinh et al., 2014

\[ y_{1:d} = x_{1:d} \]
\[ y_{d+1:D} = x_{d+1:D} + f(x_{1:d}) \]

planar flow  Rezende & Mohamed, 2015

\[ y = x + f(x) \circ g(h(x)^T x + b(x)) \]

affine coupling layer  Dinh et al., 2016

\[ y_{1:d} = x_{1:d} \]
\[ y_{d+1:D} = x_{d+1:D} \circ \exp(f(x_{1:d})) + g(x_{1:d}) \]

inverse auto-regressive flow (IAF)  Kingma et al., 2016

\[ y = \frac{x - f(x)}{\exp(g(x))} \]

masked auto-regressive flow (MAF)  Papamakarios et al., 2017

\[ y = x \circ \exp(g(x)) + f(x) \]
recent work

NICE: Non-linear Independent Components Estimation  
Dinh et al., 2014

Density Estimation Using Real NVP  
Dinh et al., 2016

Variational Inference with Normalizing Flows  
Rezende & Mohamed, 2015

Improving Variational Inference with Inverse Autoregressive Flow  
Kingma et al., 2016
recap: explicit latent variable models

Pros
- can capture abstract variables, good for semi supervised learning
- relatively fast sampling / training
- theoretical foundations from info. theory

Cons
- likelihood evaluation / inference often intractable
- requires additional assumptions on latent variables
- difficult to capture details
IMPLICIT LATENT VARIABLE MODELS
instead of using an explicit probability density, learn a model that defines an implicit density

specify a stochastic procedure for generating the data that does not require an explicit likelihood evaluation
Generative Stochastic Networks (GSNs)

Deep Generative Stochastic Networks Trainable by Backprop, Bengio et al., 2013

train an auto-encoder to learn Monte Carlo sampling transitions

the generative distribution is \textit{implicitly} defined by this transition
estimate density ratio through \textit{Bayesian two-sample test}

data distribution $p(\hat{x})$ \quad generated distribution $p(\tilde{x})$

\[
\frac{p(\hat{x})}{p(\tilde{x})} = \frac{p(x|\text{data})}{p(x|\text{gen.})}
\]

\[
\frac{p(\hat{x})}{p(\tilde{x})} = \frac{p(\text{data}|x)p(x)/p(\text{data})}{p(\text{gen.}|x)p(x)/p(\text{gen.})} \quad \text{(Bayes’ rule)}
\]

\[
\frac{p(\hat{x})}{p(\tilde{x})} = \frac{p(\text{data}|x)}{p(\text{gen.}|x)} \quad \text{(assuming equal dist. prob.)}
\]

density estimation becomes a sample discrimination task
Generative Adversarial Networks (GANs)

\[
\hat{x} \sim p(\hat{x}) \quad \rightarrow \quad D(x) \quad \leftarrow \quad \tilde{x} \sim p(\tilde{x})
\]

Data

Discriminator

Generator

learn the discriminator:

\[
p(\text{data}|x) = D(x) \quad \quad p(\text{gen.}|x) = 1 - D(x)
\]

Bernoulli outcome: \( y \in \{\text{data, gen.}\} \)

\[
\log p(y|x) = \log D(\hat{x}) + \log(1 - D(\tilde{x}))
\]

two-sample criterion:

\[
\min_G \max_D \mathbb{E}_{p(\hat{x})} [\log D(\hat{x})] + \mathbb{E}_{p(\tilde{x})} [\log(1 - D(\tilde{x}))]
\]

Goodfellow, 2016

Mohamed, 2016
Generative Adversarial Networks (GANs)

\[
\begin{align*}
\hat{x} & \sim p(\hat{x}) \\
\tilde{x} & \sim p(\tilde{x}) \\
D(x) & \quad \text{Data} \\
G(z) & \quad \text{Generator}
\end{align*}
\]

two-sample criterion:

\[
\min_G \max_D \mathbb{E}_{p(\hat{x})} \left[ \log D(\hat{x}) \right] + \mathbb{E}_{p(\tilde{x})} \left[ \log(1 - D(\tilde{x})) \right]
\]

in practice:

\[
\max_D \mathbb{E}_{p(\hat{x})} \left[ \log D(\hat{x}) \right] + \mathbb{E}_{p(\tilde{x})} \left[ \log(1 - D(\tilde{x})) \right]
\]

\[
\max_G \mathbb{E}_{p(\tilde{x})} \left[ \log D(\tilde{x}) \right]
\]

Goodfellow, 2016

Generative Adversarial Networks, Goodfellow et al., 2014
explicit models tend to cover the entire data manifold, but are constrained

implicit models tend to capture part of the data manifold, but can neglect other parts

→ "mode collapse"
Generative Adversarial Networks (GANs)

GANs can be difficult to optimize

<table>
<thead>
<tr>
<th>DCGAN</th>
<th>LSGAN</th>
<th>WGAN (clipping)</th>
<th>WGAN-GP (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline $(G: \text{DCGAN}, D: \text{DCGAN})$</td>
<td><img src="baseline" alt="Images" /></td>
<td><img src="wgan_clipping" alt="Images" /></td>
<td><img src="wgan_gp" alt="Images" /></td>
</tr>
<tr>
<td>$G$: No BN and a constant number of filters, $D$: DCGAN</td>
<td><img src="no_bn" alt="Images" /></td>
<td><img src="wgan_clipping" alt="Images" /></td>
<td><img src="wgan_gp" alt="Images" /></td>
</tr>
<tr>
<td>$G$: 4-layer 512-dim ReLU MLP, $D$: DCGAN</td>
<td><img src="relu_mlp" alt="Images" /></td>
<td><img src="wgan_clipping" alt="Images" /></td>
<td><img src="wgan_gp" alt="Images" /></td>
</tr>
<tr>
<td>No normalization in either $G$ or $D$</td>
<td><img src="no_normalization" alt="Images" /></td>
<td><img src="wgan_clipping" alt="Images" /></td>
<td><img src="wgan_gp" alt="Images" /></td>
</tr>
<tr>
<td>Gated multiplicative nonlinearities everywhere in $G$ and $D$</td>
<td><img src="gated_multiplicative" alt="Images" /></td>
<td><img src="wgan_clipping" alt="Images" /></td>
<td><img src="wgan_gp" alt="Images" /></td>
</tr>
<tr>
<td>tanh nonlinearities everywhere in $G$ and $D$</td>
<td><img src="tanh_nonlinearities" alt="Images" /></td>
<td><img src="wgan_clipping" alt="Images" /></td>
<td><img src="wgan_gp" alt="Images" /></td>
</tr>
<tr>
<td>101-layer ResNet $G$ and $D$</td>
<td><img src="resnet_g" alt="Images" /></td>
<td><img src="wgan_clipping" alt="Images" /></td>
<td><img src="wgan_gp" alt="Images" /></td>
</tr>
</tbody>
</table>

**Improved Training of Wasserstein GANs**, Gulrajani et al., 2017
without an explicit likelihood, it is difficult to quantify the performance

**inception score**

use a pre-trained Inception v3 model to quantify class and distribution entropy

$$IS(G) = \exp \left( \mathbb{E}_{p(\tilde{x})} D_{KL}(p(y|\tilde{x})||p(y)) \right)$$

$p(y|\tilde{x})$ is the class distribution for a given image

- should be highly peaked (low entropy)

$p(y) = \int p(y|\tilde{x}) d\tilde{x}$ is the marginal class distribution

- want this to be uniform (high entropy)

---

*Improved Techniques for Training GANs*, Salimans et al., 2016
*A Note on the Inception Score*, Barratt & Sharma, 2018
extensions: Wasserstein GAN

under an “ideal” discriminator, the generator minimizes the Jensen-Shannon divergence

$$D_{JS}(p(\hat{x})||p(\tilde{x})) = \frac{1}{2} D_{KL}(p(\hat{x})||\frac{1}{2}(p(\hat{x}) + p(\tilde{x}))) + \frac{1}{2} D_{KL}(p(\tilde{x})||\frac{1}{2}(p(\hat{x}) + p(\tilde{x})))$$

however, this metric can be discontinuous, making it difficult to train

$$\theta$$ is a gen. model parameter

can instead use the Wasserstein (Earth Mover’s) distance (which is continuous and diff. almost everywhere):

$$W(p(\hat{x}), p(\tilde{x})) = \inf_{\gamma \in \Pi(p(\hat{x}), p(\tilde{x}))} \mathbb{E}_{(\hat{x}, \tilde{x}) \sim \gamma} [||\hat{x} - \tilde{x}||]$$

think of it as the “minimum cost of transporting points between two distributions”

intractable to actually evaluate Wasserstein distance, but by constraining the discriminator, can evaluate

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{p(\hat{x})} [D(\hat{x})] - \mathbb{E}_{p(\tilde{x})} [D(\tilde{x})]$$

$\mathcal{D}$ is the set of Lipschitz functions, which can be enforced through weight clipping or gradient penalty

Wasserstein GANs, Arjovksy et al., 2017

Improved Training of Wasserstein GANs, Gulrajani et al., 2017
extensions: inference

can we also learn to infer a latent representation?

Adversarially Learned Inference, Dumoulin et al., 2017

Adversarial Feature Learning, Donahue et al., 2017
applications

image to image translation

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Zhu et al., 2017

experimental simulation

Learning Particle Physics by Example, de Oliveira et al., 2017

interpretable representations

InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, Chen et al., 2016

music synthesis

MIDINET: A CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORK FOR SYMBOLIC-DOMAIN MUSIC GENERATION, Yang et al., 2017

StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, Zhang et al., 2016

text to image synthesis
recap: implicit latent variable models

\[ \hat{x} \sim p(\hat{x}) \quad \rightarrow \quad D(x) \quad \leftarrow \quad \tilde{x} \sim p(\tilde{x}) \]

**Pros**
- able to learn flexible models
- requires fewer modeling assumptions
- capable of learning latent representation

**Cons**
- difficult to evaluate
- sensitive, difficult to optimize
- can be difficult to incorporate model assumptions
DISCUSSION
generative models: what are they good for?

generative models model the data distribution

1. can generate and simulate data

2. can extract structure from data
generative models: what’s next?

applying generative models to new forms of data

incorporating generative models into complementary learning systems