CSC412: Adversarial Training

David Duvenaud

Slides from Ian Goodfellow, Roger Grosse and Sebastian Nowozin
Generative Modeling

• Density estimation

Training examples

Model samples

• Sample generation
Fully Visible Belief Nets

• Explicit formula based on chain (Frey et al, 1996) rule:
  \[
  p_{\text{model}}(\mathbf{x}) = p_{\text{model}}(x_1) \prod_{i=2}^{n} p_{\text{model}}(x_i | x_1, \ldots, x_{i-1})
  \]

• Disadvantages:
  • \(O(n)\) sample generation cost
  • Generation not controlled by a latent code

PixelCNN elephants (van den Ord et al 2016)
WaveNet

Amazing quality
Sample generation slow

Two minutes to synthesize one second of audio

(Goodfellow 2016)
Change of Variables

\[ y = g(x) \Rightarrow p_x(x) = p_y(g(x)) \left| \det \left( \frac{\partial g(x)}{\partial x} \right) \right| \]

e.g. Nonlinear ICA (Hyvärinen 1999)

Disadvantages:
- Transformation must be invertible
- Latent dimension must match visible dimension

64x64 ImageNet Samples
Real NVP (Dinh et al 2016)
Variational Autoencoder
(Kingma and Welling 2013, Rezende et al 2014)

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

Disadvantages:
- Not asymptotically consistent unless \( q \) is perfect
- Samples tend to have lower quality

CIFAR-10 samples
(Kingma et al 2016)
Boltzmann Machines

\[ p(x) = \frac{1}{Z} \exp(-E(x, z)) \]

\[ Z = \sum_x \sum_z \exp(-E(x, z)) \]

- Partition function is intractable
- May be estimated with Markov chain methods
- Generating samples requires Markov chains too
GANs

- Use a latent code
- Asymptotically consistent (unlike variational methods)
- No Markov chains needed
- Often regarded as producing the best samples
  - No good way to quantify this
Generator Network

\[ x = G(z; \theta^{(G)}) \]

- Must be differentiable
- No invertibility requirement
- Trainable for any size of \( z \)
- Some guarantees require \( z \) to have higher dimension than \( x \)
- Can make \( x \) conditionally Gaussian given \( z \) but need not do so
Generative Adversarial Networks

A 1-dimensional example:

- Input distribution
- Function computed by the network
- Output distribution
Generative Adversarial Networks

\[ D(\mathbf{x}) \]

\[ \mathbf{x} \quad \text{OR} \quad \mathbf{x} = G(\mathbf{z}) \]

real-world image

generator

code vector
Generative Adversarial Networks

Updating the discriminator:

\[ D(x) \]

update the discriminator weights using backprop on the classification objective

real-world image

OR

\[ x = G(z) \]

generator

code vector
Generative Adversarial Networks

Updating the generator:

\[ D(x) \]

- backprop the derivatives, but don’t modify the discriminator weights

\[ x = G(z) \]

- flip the sign of the derivatives

- update the generator weights using backprop

\[ Z \]
Training Procedure

• Use SGD-like algorithm of choice (Adam) on two minibatches simultaneously:

  • A minibatch of training examples

  • A minibatch of generated samples

• Optional: run $k$ steps of one player for every step of the other player.
Minimax Game

\[ J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) - \frac{1}{2} \mathbb{E}_{z} \log (1 - D(G(z))) \]

\[ J^{(G)} = -J^{(D)} \]

- Equilibrium is a saddle point of the discriminator loss
- Resembles Jensen-Shannon divergence
- Generator minimizes the log-probability of the discriminator being correct
Solution

This is the canonical example of a saddle point.

There is an equilibrium, at $x = 0, y = 0$. 
Discriminator Strategy

Optimal $D(x)$ for any $p_{\text{data}}(x)$ and $p_{\text{model}}(x)$ is always

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$

Estimating this ratio using supervised learning is the key approximation mechanism used by GANs.

(Goodfellow 2016)
Non-Saturating Game

\[ J(D) = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) - \frac{1}{2} \mathbb{E}_{z} \log (1 - D(G(z))) \]

\[ J(G) = -\frac{1}{2} \mathbb{E}_{z} \log D(G(z)) \]

- Equilibrium no longer describable with a single loss
- Generator maximizes the log-probability of the discriminator being mistaken
- Heuristically motivated; generator can still learn even when discriminator successfully rejects all generator samples
DCGAN Architecture

Most “deconvs” are batch normalized

(Radford et al 2015)
DCGANs for LSUN Bedrooms

(Radford et al 2015)
Vector Space Arithmetic

Figure 15.9: A generative model has learned a distributed representation that disentangles the concept of gender from the concept of wearing glasses. If we begin with the representation of the concept of a man with glasses, then subtract the vector representing the concept of a man without glasses, and finally add the vector representing the concept of a woman without glasses, we obtain the vector representing the concept of a woman with glasses. The generative model correctly decodes all of these representation vectors to images that may be recognized as belonging to the correct class. Images reproduced with permission from Radford et al. (2015).

It is common is that one could imagine learning about each of them without having to see all the configurations of all the others. Radford et al. (2015) demonstrated that a generative model can learn a representation of images of faces, with separate directions in representation space capturing different underlying factors of variation. Figure 15.9 demonstrates that one direction in representation space corresponds to whether the person is male or female, while another corresponds to whether the person is wearing glasses. These features were discovered automatically, not fixed a priori. There is no need to have labels for the hidden unit classifiers: gradient descent on an objective function of interest naturally learns semantically interesting features, so long as the task requires such features. We can learn about the distinction between male and female, or about the presence or absence of glasses, without having to characterize all of the configurations of the other features by examples covering all of these combinations of values. This form of statistical separability is what allows one to generalize to new configurations of a person’s features that have never been seen during training.

(Radford et al, 2015)
Batch norm in $G$ can cause strong intra-batch correlation.
Non-convergence in GANs

• Exploiting convexity in function space, GAN training is theoretically guaranteed to converge if we can modify the density functions directly, but:

  • Instead, we modify $G$ (sample generation function) and $D$ (density ratio), not densities

  • We represent $G$ and $D$ as highly non-convex parametric functions

“Oscillation”: can train for a very long time, generating very many different categories of samples, without clearly generating better samples

• Mode collapse: most severe form of non-convergence
Mode Collapse

\[
\min_G \max_D \max V(G, D) \neq \max_D \min_G V(G, D)
\]

- \(D\) in inner loop: convergence to correct distribution
- \(G\) in inner loop: place all mass on most likely point

\(\text{(Metz et al 2016)}\)
Mode collapse causes low output diversity

(Reed et al, submitted to ICLR 2017)

(Reed et al 2016)
Minibatch GAN on ImageNet

(Salimans et al 2016)
Problems with Counting
Problems with Global Structure
Discrete outputs

- $G$ must be differentiable
- Cannot be differentiable if output is discrete

Possible workarounds:

- REINFORCE (Williams 1992)
- Concrete distribution (Maddison et al 2016) or Gumbel-softmax (Jang et al 2016)
- Learn distribution over continuous embeddings, decode to discrete
Can train GANs with any divergence

GAN (Jensen-Shannon)  Hellinger  Kullback-Leibler

Slide from Sebastian Nowozin
## f-GAN [Nowozin et al, 2016]

<table>
<thead>
<tr>
<th>Name</th>
<th>Output activation $g_f$</th>
<th>dom$_{f^*}$</th>
<th>Conjugate $f^*(t)$</th>
<th>$f'(1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total variation</td>
<td>$\frac{1}{2} \tanh(v)$</td>
<td>$-\frac{1}{2} \leq t \leq \frac{1}{2}$</td>
<td>$t$</td>
<td>0</td>
</tr>
<tr>
<td>Kullback-Leibler (KL)</td>
<td>$v$</td>
<td>$\mathbb{R}$</td>
<td>$\exp(t - 1)$</td>
<td>1</td>
</tr>
<tr>
<td>Reverse KL</td>
<td>$- \exp(v)$</td>
<td>$\mathbb{R}_-$</td>
<td>$-1 - \log(-t)$</td>
<td>-1</td>
</tr>
<tr>
<td>Pearson $\chi^2$</td>
<td>$v$</td>
<td>$\mathbb{R}$</td>
<td>$\frac{1}{4} t^2 + t$</td>
<td>0</td>
</tr>
<tr>
<td>Neyman $\chi^2$</td>
<td>$1 - \exp(v)$</td>
<td>$t &lt; 1$</td>
<td>$2 - 2\sqrt{1-t}$</td>
<td>0</td>
</tr>
<tr>
<td>Squared Hellinger</td>
<td>$1 - \exp(v)$</td>
<td>$t &lt; 1$</td>
<td>$\frac{t}{\frac{1-t}{1-t}} + \frac{1}{W(e^{1-t})} + t - 2$</td>
<td>0</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>$v$</td>
<td>$\mathbb{R}$</td>
<td>$- \log(2 - \exp(t))$</td>
<td>0</td>
</tr>
<tr>
<td>Jensen-Shannon</td>
<td>$\log(2) - \log(1 + \exp(-v))$</td>
<td>$t &lt; \log(2)$</td>
<td>$(1 - \pi) \log \frac{1 - \pi e^t / \pi}{1 - \pi e^t / \pi}$</td>
<td>0</td>
</tr>
<tr>
<td>Jensen-Shannon-weighted</td>
<td>$-\pi \log \pi - \log(1 + \exp(-v))$</td>
<td>$t &lt; -\pi \log \pi$</td>
<td>$- \log(1 - \exp(t))$</td>
<td>0</td>
</tr>
<tr>
<td>GAN</td>
<td>$- \log(1 + \exp(-v))$</td>
<td>$\mathbb{R}_-$</td>
<td>$\frac{1}{\alpha} (t(\alpha - 1) + 1)^{\frac{\alpha}{\alpha - 1}} - \frac{1}{\alpha}$</td>
<td>0</td>
</tr>
<tr>
<td>$\alpha$-div. ($\alpha &lt; 1, \alpha \neq 0$)</td>
<td>$\frac{1}{1-\alpha} - \log(1 + \exp(-v))$</td>
<td>$t &lt; \frac{1}{1-\alpha}$</td>
<td>$\frac{1}{\alpha} (t(\alpha - 1) + 1)^{\frac{\alpha}{\alpha - 1}} - \frac{1}{\alpha}$</td>
<td>0</td>
</tr>
<tr>
<td>$\alpha$-div. ($\alpha &gt; 1$)</td>
<td>$v$</td>
<td>$\mathbb{R}$</td>
<td>$\frac{1}{t^\alpha} (t(\alpha - 1) + 1)^{\frac{\alpha}{\alpha - 1}} - \frac{1}{t^\alpha}$</td>
<td>0</td>
</tr>
</tbody>
</table>
Loss does not seem to explain why GAN samples are sharp

Takeaway: the approximation strategy matters more than the loss

(Nowozin et al 2016)
Relation to VAEs

- Same graphical model: $z \rightarrow x$
- VAEs have an explicit likelihood: $p(x|z)$
- GANs have no explicit likelihood
  - aka implicit models, likelihood-free models
- Can use same trick for implicit $q(z|x)$
Generalizing these ideas

• Adversarial Variational Bayes. Lars Mescheder, Sebastian Nowozin, Andreas Geiger, 2017

• Learning in Implicit Generative Models. Shakir Mohamed, Balaji Lakshminarayanan, 2016

• Variational Inference using Implicit Distributions. Ferenc Huszar, 2017

• Deep and Hierarchical Implicit Models. Dustin Tran, Rajesh Ranganath, David Blei, 2017
Takeaways

• Can train a latent-variable model without specifying a likelihood function at the last layer

• This is nice because most likelihoods (e.g. spherical Gaussians on pixels) are nonsense that we only added to smooth out the objective

• Similar to move from Exact inference to MCMC to var. inf: Don’t restrict model to allow easy inference - just let a neural network clean up after.
Other uses

• Same as any other generative latent-variable model
Image to Image Translation

Figure 1: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the method on several. In each case we use the same architecture and objective, and simply train on different data.

Abstract
We investigate conditional adversarial networks as a general-purpose solution to image-to-image translation problems. These networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping. This makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations. We demonstrate that this approach is effective at synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images, among other tasks. As a community, we no longer hand-engineer our mapping functions, and this work suggests we can achieve reasonable results without hand-engineering our loss functions either.

Many problems in image processing, computer graphics, and computer vision can be posed as “translating” an input image into a corresponding output image. Just as a concept may be expressed in either English or French, a scene may be rendered as an RGB image, a gradient field, an edge map, a semantic label map, etc. In analogy to automatic language translation, we define automatic image-to-image translation as the problem of translating one possible representation of a scene into another, given sufficient training data (see Figure 1). One reason language translation is difficult is because the mapping between languages is rarely one-to-one – any given concept is easier to express in one language than another. Similarly, most image-to-image translation problems are either many-to-one (computer vision) – mapping photographs to edges, segments, or semantic labels, or one-to-many (computer graphics) – mapping labels or sparse user inputs to realistic images. Traditionally, each of these tasks has been tackled with separate, special-purpose machinery (e.g., [7,15,11,1,3,37,21,26,9,42,21,26]), despite the fact that the setting is always the same: predict pixels from pixels. Our goal in this paper is to develop a common framework for all these problems.
iGAN

youtube

(Zhu et al 2016)
Single Image Super-Resolution

original
bicubic (21.59dB/0.6423)
SRResNet (23.44dB/0.7777)
SRGAN (20.34dB/0.6562)

(Ledig et al 2016)
Semi-Supervised Classification

CIFAR-10

<table>
<thead>
<tr>
<th>Model</th>
<th>Test error rate for a given number of labeled samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1000</td>
</tr>
<tr>
<td>Ladder network [24]</td>
<td>20.40±0.47</td>
</tr>
<tr>
<td>CatGAN [14]</td>
<td>19.58±0.46</td>
</tr>
<tr>
<td>Our model</td>
<td>21.83±2.01</td>
</tr>
<tr>
<td>Ensemble of 10 of our models</td>
<td>19.22±0.54</td>
</tr>
</tbody>
</table>

SVHN

<table>
<thead>
<tr>
<th>Model</th>
<th>Percentage of incorrectly predicted test examples for a given number of labeled samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>500</td>
</tr>
<tr>
<td>DGN [21]</td>
<td>36.02±0.10</td>
</tr>
<tr>
<td>Virtual Adversarial [22]</td>
<td>24.63</td>
</tr>
<tr>
<td>Auxiliary Deep Generative Model [23]</td>
<td>22.86</td>
</tr>
<tr>
<td>Skip Deep Generative Model [23]</td>
<td>16.61±0.24</td>
</tr>
<tr>
<td>Our model</td>
<td>18.44 ± 4.8</td>
</tr>
<tr>
<td>Ensemble of 10 of our models</td>
<td>5.88 ± 1.0</td>
</tr>
</tbody>
</table>

(Salimans et al 2016)
Learning interpretable latent codes / controlling the generation process

(a) Azimuth (pose)  (b) Elevation

(c) Lighting  (d) Wide or Narrow

InfoGAN (Chen et al 2016)
PPGN for caption to image

oranges on a table next to a liquor bottle

(Nguyen et al 2016)
Class wrap-up
ML as a bag of tricks

Fast special cases:
- K-means
- Kernel Density Estimation
- SVMs
- Boosting
- Random Forests
- K-Nearest Neighbors

Extensible family:
- Mixture of Gaussians
- Latent variable models
- Gaussian processes
- Deep neural nets
- Bayesian neural nets
- ??
Regularization as a bag of tricks

Fast special cases:

• Early stopping
• Ensembling
• L2 Regularization
• Gradient noise
• Dropout

Extensible family:

• Stochastic variational inference

• Expectation-Maximization
A language of models

- Hidden Markov Models, Mixture of Gaussians, Logistic Regression

- These are simply “sentences” - examples from a language of models.

- We will try to show larger family, and point out common special cases.
AI as a bag of tricks

Russel and Norvig’s parts of AI:

• Machine learning
• Natural language processing
• Knowledge representation
• Automated reasoning
• Computer vision
• Robotics

Extensible family:

• Deep probabilistic latent-variable models + decision theory
Where are we now?

• Open research areas:
  • Optimization (especially minimax)
  • Generalizing style transfer
  • Bayesian GANs, VAEs
  • Model-based RL
  • Bayesian neural networks
  • Learning discrete latent structure
  • Learning discrete model structure
Thanks a lot!