Autoregressive and Invertible Models

CSC2541 Fall 2016

Haider Al-Lawati
(haider.al.lawati@mail.utoronto.ca)
Min Bai
(mbai@cs.toronto.edu)
Lluís Castrejón
(castrejon@cs.toronto.edu)
Tianle Chen
(tianle.chen@mail.utoronto.ca)

Christopher Meaney
(christopher.meaney@utoronto.ca)
Jeffrey Negrea
(negrea@utstat.toronto.edu)
Jake Snell
(jsnell@cs.toronto.edu)
Bowen Xu
(xubo3@cs.toronto.edu)
Outline

If training vanilla neural nets is optimization over functions, training recurrent nets is optimization over programs.

- Recurrent neural networks
  - General Idea
  - Gated Recurrent Unit
  - Pros and Cons
  - Applications

- Invertible Models
  - Overview
  - Real NVP
  - Extensions
Recurrent Neural Networks - General Idea

- Models series data by factorizing the joint probability

\[ p(x_1, x_2, \ldots, x_n) = \prod_{k=1}^{n} p(x_k | x_1, \ldots, x_{k-1}) \]

- Summarize the information from previous observations in a sufficient statistic, \( h \)

\[ p(x_k | x_1, \ldots, x_{k-1}) = g_\theta(x_k | h_{k-1}) \]

- Loss Function: Negative Log-Likelihood

\[ -l(\theta) = - \sum_{k=1}^{n} \log g_\theta(x_k | h_{k-1}) \]
Recurrent Neural Networks - General Idea

- Autoregressive: using data from previous observations to predict next observation
- $f$ generates hidden and deterministic states $h$ given inputs $x$
- $g$ generates probability distribution (or mass) functions for next $x$ given $h$
  - For discrete $x$, $g$ contains a normalization by softmax
- $h_0$ must be initialized; it can be initialized by...
  - Sampling from some distribution
  - Learning it as an additional parameter
  - Using external information

\[
p(x_k | x_1, \ldots, x_{k-1}) = g_\theta(x_k | h_{k-1})
\]

\[
h_k = f_\theta(x_k, h_{k-1})
\]
Recurrent Neural Networks - General Idea

- Generate output, $y$, at each time step or at the end of the time series
  - Can be generated deterministically or sampled
  - May or may not be the same type as the input
  - Can model a single prediction of the next input or a joint prediction of the next $n$ inputs

\[ y_k = q_\theta(h_k) \text{ or } y_k \sim q_\theta(y_k|h_k) \]
Recurrent Neural Networks - General Idea

- Optimized via backprop through time
  - Equivalent to backprop and reverse-mode auto-differentiation
  - Costly to compute gradients for higher time steps
  - Number of application of chain rule proportional to depth of data
  - Need to store the gradient for each timestep
Simple Recurrent Neural Networks Diagram

Choices for Likelihood Functions

- For discrete output:
  - Use softmax to generate multinomial distribution from RNN output

- For continuous output:
  - Use RNN output as parameters for a chosen probability density function
Pros and Cons of Vanilla RNN

Cons:
- We cannot answer any query about the joint distribution
  - Only makes forward inference, no backwards looking inference or interpolation
- The gradient explodes/decays exponentially with time making training using first order optimization methods challenging
- Storing gradients for all time steps is memory intensive

Pros:
- Uses likelihood functions
- Can handle any length of input sequences
- Trainable on large amounts of data without needing to establish priors or structure
Bidirectional RNN

Problem: For classification you want to incorporate information from words both preceding and following:

\[ y_t = g(U[\vec{h}_t; \hat{h}_t] + c) \]

\[ h_t = f(\vec{W}_t x_t + \vec{V}_t \hat{h}_{t-1} + \vec{b}) \]

\[ \hat{h}_t = f(\vec{W}_t x_t + \vec{V}_t \hat{h}_{t+1} + \vec{b}) \]

\( h = [\vec{h}; \hat{h}] \) now represents (summarizes) the past and future around a single token.
Gated Recurrent Unit
Encoder RNNs

Convert sequential data into a compressed representation

https://www.tensorflow.org/versions/r0.10/tutorials/seq2seq/index.html
Decoder RNNs

Predict sequential data given an initial condition (inference)

https://www.tensorflow.org/versions/r0.10/tutorials/seq2seq/index.html
Application of RNNs

Recall what an RNN looks like:

Application of RNNs

Recall what an RNN looks like:

Application of RNNs

Recall what an RNN looks like:

Application of RNNs

Recall what an RNN looks like:

[Diagram of an RNN]

Application of RNNs

Recall what an RNN looks like:

What should our inputs be?
Application: Character-level language model

Predict next character given a string of characters

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Text generation

Trained on a concatenation of essays written by some startup guru using a 2-layer LSTM with 512 hidden nodes. Better than Markov Models.

The surprised in investors weren’t going to raise money. I’m not the company with the time there are all interesting quickly, don’t have to get off the same programmers. There’s a super-angel round fundraising, why do you can do. If you have a different physical investment are become in people who reduced in a startup with the way to argument the acquirer could see them just that you’re also the founders will part of users’ affords that and an alternation to the idea. [2] Don’t work at first member to see the way kids will seem in advance of a bad successful startup. And if you have to act the big company too.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Text generation

Locally the text looks correct, but upon closer inspection we find mistakes in the grammar and no coherent discourse (longer range correlations).

The surprised in investors weren’t going to raise money. I’m not the company with the time there are all interesting quickly, don’t have to get off the same programmers. There’s a super-angel round fundraising, why do you can do. If you have a different physical investment are become in people who reduced in a startup with the way to argument the acquirer could see them just that you’re also the founders will part of users’ affords that and an alternation to the idea. [2] Don’t work at first member to see the way kids will seem in advance of a bad successful startup. And if you have to act the big company too.
Text generation: Shakespeare plays

The model correctly simulates the play structure (and almost generates interesting stories)

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Text generation: Better than you at LaTeX

Dataset is Algebraic Geometry LaTeX source files. Sampled code almost compiles, sometimes chooses to omit proofs (as one does). Note that some long environments did not close, as some recurrent features are not generated.

Proof. Omitted.

Lemma 0.1. Let \( C \) be a set of the construction.

Let \( C \) be a gerber covering. Let \( \mathcal{F} \) be a quasi-coherent sheaves of \( \mathcal{O} \)-modules. We have to show that

\[ \mathcal{O}_{\mathcal{X}}(C) \]

Proof. This is an algebraic space with the composition of sheaves \( \mathcal{F} \) on \( X_{\text{node}} \), we have

\[ \mathcal{O}_X(\mathcal{F}) = \{ \text{morph} \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F}) \} \]

where \( \mathcal{G} \) defines an isomorphism \( \mathcal{F} \to \mathcal{F} \) of \( \mathcal{O} \)-modules.

Lemma 0.2. This is an integer \( \tau \) is injective.

Proof. See Spaces, Lemma ??.

Lemma 0.3. Let \( S \) be a scheme. Let \( X \) be a scheme and \( X \) is an affine open covering. Let \( U \subseteq X \) be a canonical and locally of finite type. Let \( X \) be a scheme.

Let \( X \) be a scheme which is equal to the formal complex.

The following to the construction of the lemma follows.

Let \( X \) be a scheme. Let \( X \) be a scheme covering. Let

\[ b: X \to Y' \to Y \to Y' \times_X Y \to X. \]

be a morphism of algebraic spaces over \( S \) and \( Y \).

Proof. Let \( X \) be a nonzero scheme of \( X \). Let \( X \) be an algebraic space. Let \( \mathcal{F} \) be a quasi-coherent sheaf of \( \mathcal{O}_X \)-modules. The following are equivalent:

1. \( \mathcal{F} \) is an algebraic space over \( S \).
2. \( \mathcal{X} \) is an affine open covering.

Consider a common structure on \( X \) and \( X \) the functor \( \mathcal{O}_X(U) \) which is locally of finite type.

This since \( \mathcal{F} \in \mathcal{F} \) and \( x \in \mathcal{G} \) the diagram

\[ \begin{array}{c}
\text{Spec}(R) \\
\text{Spec}(R) \\
\end{array} \]

\[ \begin{array}{c}
\text{Spec}(R) \\
\text{Spec}(R) \\
\end{array} \]

is a finite. Then \( \mathcal{G} \) is a finite type and assume \( S \) is a finite and \( \mathcal{G} \) is a finite type \( \mathcal{F} \). This is finite type diagram, and

- the composition of \( \mathcal{G} \) is a regular sequence,
- \( \mathcal{O}_X \) is a sheaf of rings.

Proof. We have that \( X = \text{Spec}(R) \) and \( \mathcal{F} \) is a finite type presentation by algebraic space. The property \( \mathcal{F} \) is a finite morphism of algebraic stacks. Then the morphism \( X \) is an open covering of \( U \).

Proof. This is clear that \( \mathcal{G} \) is a finite presentation, see Lemma ??.

A reduced space we conclude that \( U \) is an open covering of \( X \). The functor \( \mathcal{F} \) is a "fold"

\[ \mathcal{G}_{\mathcal{X}} \to \mathcal{F}_U \to \mathcal{G}_{\mathcal{X}} \to \mathcal{G}_{\mathcal{X}} \]

is an isomorphism of covering of \( \mathcal{O}_X \). If \( \mathcal{F} \) is the unique element of \( \mathcal{F} \) such that \( X \) is an isomorphism.

The property \( \mathcal{F} \) is a disjoint union of Proposition ?? and we can filtered set of presentations of scheme \( \mathcal{O}_X \)-algebras with \( \mathcal{F} \) are open of finite type over \( X \).

If \( \mathcal{F} \) is a scheme theoretic image point.

If \( \mathcal{F} \) is a finite direct sum \( \mathcal{O}_X \), then is a closed immersion, see Lemma ?? and this is a sequence of \( \mathcal{F} \) is a similar morphism.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Pixel-RNN (van den Oord et al. 2016)

Generative model of images

Pixel-RNN (van den Oord et al. 2016)

Generative model of images

Note: completions can only go one way

Figure 1. Image completions sampled from a PixelRNN.
Pixel-RNN (van den Oord et al. 2016)

Generative model of images

Teacher forcing (Doya 1993)

This is what an RNN output looks like at the beginning (compounding errors)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Teacher forcing (Doya 1993)

Use ground truth inputs during training to make training more stable and prevent exploding gradients.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Teacher forcing (Doya 1993)

At inference a sample from your posterior distribution is used.

Mismatch between training and inference!
Scheduled sampling (S. Bengio et al., 2015)

Use an schedule during training to progressively move from using GT inputs to using samples.

GT or sample depending on training step

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Initial condition

What can we use as an initial condition?

Image captioning (Kiros et al. Vinyals et al., …)

Predict caption given an image

Obtain a vector representation of an image (e.g. using a CNN)

Image captioning (Kiros et al. Vinyals et al., ...)

Predict caption given an image

Start generating text

Image captioning (Kiros et al. Vinyals et al., …)

Predict caption given an image

Code example
Invertible Models
Invertible Models

- Class of probabilistic generative models with:
  - Exact sampling \( x \sim p(x) \)
  - Exact inference \( p(z|x) = \frac{p(z)p(x|z)}{p(x)} \)
  - Exact likelihood computation \( p(x) = \int p(x|z) p(z) \, dz \)

- Relies on exploiting the change-of-variable formula for bijective functions
Generative Procedure

- Start with a latent space that is easy to sample from
  - e.g. $z \sim p_Z(z) = \mathcal{N}(0, I)$
- Pass the sampled point through a generator function
  - $x = g(z)$
- This gives us exact sampling ✓
- Similar procedure as for VAEs except the generator function is deterministic
Inference

● Choose $g$ such that the mapping is *bijective*
  ○ Each point $x$ has exactly one associated point in latent space, and vice versa
  ○ Therefore $g$ has an inverse function $f = g^{-1}$
● Given an arbitrary $x$, directly compute $z = f(x)$
Likelihood Computation

- Given a random variable $Z$ and a bijection $X = g(Z) \Leftrightarrow Z = f(X)$, the pdf of $X$ can be expressed in terms of the pdf of $Z$:

$$p_X(x) = p_Z(f(x)) \left| \det \left( \frac{\partial f(x)}{\partial x^T} \right) \right|$$

- Due to the bijection constraint, there is only one possible $z$ for a given $x$. Hence the likelihood function is reduced to the above pdf for $X = g(Z)$.

  - Exact likelihood computation

  ![Jacobian matrix](Jacobian_matrix.png)
Recall: Change of Variables for 2D

\[ Pr(z_1, z_2 \in R_z) = \iint_{R_z} p_z(z_1, z_2) \, dz_1 \, dz_2 \]

Let \( R_x \) be identical to \( R_z \) but specified in the \( x_1 - x_2 \) plane

\[ \iint_{R_x} p_z(z_1(x_1, x_2), z_2(x_1, x_2)) \left| \frac{\partial(z_1, z_2)}{\partial(x_1, x_2)} \right| \, dx_1 \, dx_2 \]

\[ Pr(z_1, z_2 \in R_z) = Pr(x_1, x_2 \in R_x) = \iint_{R_x} p_x(x_1, x_2) \, dx_1 \, dx_2 \]

\[ p_x(x_1, x_2) = p_z(z_1(x_1, x_2), z_2(x_1, x_2)) \left| \frac{\partial(z_1, z_2)}{\partial(x_1, x_2)} \right| \]
Example: Cartesian to Polar Coordinates

\[ p_{XY}(x, y) \]
\[ p_{r\theta}(r, \theta) = ? \]

\[ j = \begin{pmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -r \sin(\theta) \\ \sin(\theta) & r \cos(\theta) \end{pmatrix} \]

\[ \Rightarrow |\det j| = |r \cos^2(\theta) + r \sin^2(\theta)| = |r| \]

\[ p_{r\theta}(r, \theta) = p_{XY}(r \cos(\theta), r \sin(\theta)) |\det j| = p_{XY}(r \cos(\theta), r \sin(\theta)) r \]
Invertible Approach: Challenges

- Invertible approach gives us exact sampling, inference and likelihood computation.

- There are still three big questions to answer:
  
a. How to parameterize expressive bijective functions?
  b. How to efficiently compute a large Jacobian matrix?
  c. How to efficiently compute the determinant of a large matrix?
Parameterization of Bijective Functions

- Dinh et al. 2016 propose to use affine coupling layers:

\[
\begin{align*}
  y_{1:d} &= x_{1:d} \\
  y_{d+1:D} &= x_{d+1:D} \odot \exp \left( l(x_{1:d}) \right) + m(x_{1:d})
\end{align*}
\]

- Flexibility can be gained by increasing the complexity of \( l \) and \( m \).

\[
\begin{align*}
  x_{1:d} &= y_{1:d} \\
  x_{d+1:D} &= (y_{d+1:D} - m(y_{1:d})) \odot \exp \left( - l(y_{1:d}) \right)
\end{align*}
\]
Jacobian Computation

- The Jacobian of the transformation

\[
y_{1:d} = x_{1:d}
\]
\[
y_{d+1:D} = x_{d+1:D} \odot \exp(l(x_{1:d})) + m(x_{1:d})
\]

can be expressed as:

\[
\frac{\partial y}{\partial x^T} = \begin{bmatrix}
\mathbb{I}_d & 0 \\
\frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \text{diag}(\exp(l)) \\
x_{1:d} & x_{d+1:D}
\end{bmatrix}
\begin{bmatrix}
y_{1:d} \\
y_{d+1:D}
\end{bmatrix}
\end{bmatrix}
\]

\[
\frac{\partial y_i}{\partial x_j}
\]
Determinant Computation

- For a 2 x 2 matrix

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

- For a triangular matrix

$$\det \begin{pmatrix} a & 0 & 0 & 0 \\ b & c & 0 & 0 \\ d & e & f & 0 \\ w & x & y & z \end{pmatrix} = acfz$$
Determinant Computation

The Jacobian in our example is a triangular matrix:

$$\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \text{diag}(\exp(l)) \end{bmatrix}$$

Hence, its determinant is

$$\det \frac{\partial y}{\partial x^T} = \exp \left( \sum_j l(x_{1:d})_j \right)$$

Regardless of the lower-left block!
Masked Convolution

- For images choose $l$ and $m$ to be deep convnets
- Rather than explicitly partitioning the input, use binary mask instead:
\[
y = b \odot x + (1 - b) \odot \left( x \odot \exp \left( l(b \odot x) \right) + m(b \odot x) \right)
\]
- Masks applied in alternating checkerboard pattern

Dinh et al. 2016
Multi-scale Architecture

- Squeezing operation reduces spatial resolution and increases channels

- At each scale use the following recipe:
  - Three coupling layers with alternating checkerboards
  - Squeezing operation
  - Three coupling layers with channel-wise masking

Checkerboard and channel-wise masks are not redundant

Dinh et al. 2016
Factoring out latent variables

- Propagating full D-dimensional vector through all layers is expensive in computation and memory.
- Factor out half of the latent variables at regular intervals.

\[
\begin{align*}
  h^{(0)} &= x \\
  (z^{(i+1)}, h^{(i+1)}) &= f^{(i+1)}(h^{(i)}) \\
  z^{(L)} &= f^{(L)}(h^{(L-1)}) \\
  z &= (z^{(1)}, \ldots, z^{(L)})
\end{align*}
\]

Dinh et al. 2016
Factoring out latent variables

- Variables factored out in lower layers capture noise and low-level details.
- Those in higher layers capture more abstract concepts

Dinh et al. 2016
Samples

LSUN Tower Category

x10

Dinh et al. 2016
Latent Manifold

Dinh et al. 2016
## Compression Results

<table>
<thead>
<tr>
<th></th>
<th>CIFAR-10</th>
<th>Imagenet (32 x 32)</th>
<th>Imagenet (64 x 64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixel RNN</td>
<td>3.00</td>
<td>3.86</td>
<td>3.63</td>
</tr>
<tr>
<td>Real NVP</td>
<td>3.49</td>
<td>4.28</td>
<td>4.01</td>
</tr>
<tr>
<td>Conv DRAW</td>
<td>3.59</td>
<td>&lt; 4.40</td>
<td>&lt; 4.10</td>
</tr>
</tbody>
</table>

*bits/dim on test set*

Achieves competitive performance relative to:

- Pixel RNN (an autoregressive approach)
- Convolutional DRAW (a VAE approach)

Dinh et al. 2016
Related Work

- Real NVP (Dinh et al. 2016) was able to get good results on challenging datasets and is getting a lot of attention for it. But other previous works have also explored the role of bijective functions for density estimation.
  - Deep density models (Rippel & Adams 2013)
  - Non-linear Independent Components Estimation (Dinh et al. 2014)
  - Generalized Divisive Normalization (Ballé et al. 2015)

Recap + Comparison with Other Approaches

<table>
<thead>
<tr>
<th></th>
<th>VAE</th>
<th>Auto-regressive</th>
<th>GAN</th>
<th>Invertible Models</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exact Sampling</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Exact Inference</strong></td>
<td>❌</td>
<td>-</td>
<td>❌</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Exact Log-Likelihood</strong></td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
</tr>
</tbody>
</table>

Invertible models have desirable characteristics but have two main drawbacks:

- Must use bijective functions with tractable Jacobian determinant
- Latent dimensionality is equal to the dimensionality of input
Invertible Models + VAEs

- Highest level latent variables are most important for the content of an image.
- Instead of using a simple Gaussian prior, stack a VAE on top to get a more expressive distribution for these latent variables.
Invertible Models + Autoregressive

- Bijectivity gives another way to specify likelihood: dist. over $z_i \Leftrightarrow$ dist. over $x_i$
Invertible Models + Autoregressive

- Bijectivity gives another way to specify likelihood: dist. over $z_i \iff$ dist. over $x_i$
Invertible Models + GANs

- Replacing a GAN’s generator with a bijective function gives us inference for free:

  \[ g(z) \]
  \[ f(x) \]

- Existing work to invert the generator of a GAN (ALI of Dumoulin et al. 2016), (BiGAN of Donahue et al. 2016) but do not directly engineer the generator to be bijective.