Lecture 13: Data-to-Text
Adversarial Examples
GANS / Dual Learning
Outline

- Data-to-Text
  - Task description / Approaches
  - Case study (Wiseman et al. 2017)

- Adversarial examples
  - In Vision
  - In NLP
  - Case study (Iyyer et al. 2018)

- GANS and Dual Learning
Generation is divided into modular, yet highly interdependent, decisions: (1) content planning defines which parts of the input fields or meaning representations should be selected; (2) sentence planning determines which selected fields are to be dealt with in each output sentence; and (3) surface realization generates those sentences.

Data-driven approaches have been proposed to automatically learn the individual modules. One approach first aligns records and sentences and then learns a content selection model (Duboue and McKeeown, 2002; Barzilay and Lapata, 2005). Hierarchical hidden semi-Markov generative models have also been used to first determine which facts to discuss and then to generate words from the predicates and arguments of the chosen facts (Liang et al., 2009). Sentence planning has been formulated as a supervised set partitioning problem over facts where each partition corresponds to a sentence (Barzilay and Lapata, 2006). End-to-end approaches have combined sentence planning and surface realization by using explicitly aligned sentence/meaning pairs as training data (Ratnaparkhi, 2002; Wong and Mooney, 2007; Belz, 2008; Lu and Ng, 2011). More recently, content selection and surface realization have been combined (Angeli et al., 2010; Kim and Mooney, 2010; Konstas and Lapata, 2013).

At the intersection of rule-based and statistical methods, hybrid systems aim at leveraging human contributed rules and corpus statistics (Langkilde and Knight, 1998; Soricut and Marcu, 2006; Mairesse and Walker, 2011).

Our approach is inspired by the recent success of neural language models for image captioning (Kiros et al., 2014; Karpathy and Fei-Fei, 2015; Vinyals et al., 2015; Fang et al., 2015; Xu et al., 2015), machine translation (Devlin et al., 2014; Bahdanau et al., 2015; Luong et al., 2015), and modeling conversations and dialogues (Shang et al., 2015; Wen et al., 2015; Yao et al., 2015).

Our model is most similar to Mei et al. (2016) who use an encoder-decoder style neural network model to tackle the WEATHERGOV and ROBOCUP tasks. Their architecture relies on LSTM units and an attention mechanism which reduces scalability compared to our simpler design.
Niels Henrik David Bohr (Danish: [nels ˈboɐ̯ˀ]; 7 October 1885 – 18 November 1962) was a Danish physicist who made foundational contributions to understanding atomic structure and quantum theory, for which he received the Nobel Prize in Physics in 1922.
Instead of showing raters, Could also show raters a data structure and ask them to write a sentence about it.

**Data**

name[Loch Fyne],
eatType[restaurant],
food[French],
priceRange[less than £20]
familyFriendly[yes]

The problem is that raters will usually write mundane sentences with very little variation.

*Loch Fyne is French restaurant with a price range less than £20 that is family friendly.*
E2E dataset [Novikova et al. 2016, Novikova et al. 2017]

Example image

Example of sentences raters generate

Loch Fyne is a family-friendly restaurant providing sushi at a low cost.

Loch Fyne is a Japanese family friendly restaurant catering to a budget of below £20.

Instead of showing data directly, show pictures to the rater instead. Authors show that this results in much more natural and diverse sentences.
Models

- Same types of models as for summarization can be used in data-to-text.

- Typical strategy is to “linearize” the table before feeding into encoder.
Copied Mechanisms [Gu et al. 2016, See et al. 2017]

- Allow the neural network the option of directly copying words from the source text.

Figure 3: Pointer-generator model. For each decoder timestep a generation probability $p_{gen}$ is calculated, which weights the probability of generating a word from the extended vocabulary.

For each document let the input sequence be $x^t_0$, ..., $x^T_0$. In addition, the decoder input $x^t_{gen}$ is the sigmoid function.

During training, the loss for timestep $t$ is

$$
\text{loss} = -\sum_{w \in \text{vocab}} P(w|s^t) \log p(w|s^t)\!
$$

where

- $s^t$ is the sequence of states, known as the decoder state
- $s^t$ is concatenated with the decoder input $x^t_{gen}$
- $s^t$ is calculated from the context vector $h^t_v$ which is a weighted sum of the encoder hidden states
- $h^t_v$ is an out-of-vocabulary (OOV) article.

Best viewed in color.

See et al. 2017
Data-to-Text

- Given structured data, write a textual description.

- Related to but different from summarization.

- Why do you think this task is interesting or useful?
Outline

- Data-to-Text
  - Task description / Approaches
  - Case study (Wiseman et al. 2017)

- Adversarial examples
  - In Vision
  - In NLP
  - Case study (Iyyer et al. 2018)

- GANS and Dual Learning
Adversarial Examples

- Neural models reach very high performance when the test data is iid to
  the training data.

- However, even small perturbations to the data can cause neural models
  to perform poorly
  - Sometimes model is **not robust** too small perturbations that leave
    the label **unchanged**
  - Sometimes model is **not sensitive** enough to small perturbations that
    **change** the label
Adversarial Examples [Szegedy et al. 2014, Goodfellow et al. 2015]

Let $\theta$ be the parameters of a model, $x$ the input to the model, $y$ the targets associated with $x$ (for machine learning tasks that have targets) and $J(\theta, x, y)$ be the cost used to train the neural network.

We can linearize the cost function around the current value of $\theta$, obtaining an optimal max-norm constrained perturbation of $\epsilon = \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$.

We refer to this as the "fast gradient sign method" of generating adversarial examples. Note that the required gradient can be computed efficiently using backpropagation.

We find that this method reliably causes a wide variety of models to misclassify their input. See Fig. 1 for a demonstration on ImageNet. We find that using $\epsilon = 0.007$, we cause a shallow softmax classifier to have an error rate of 99.9% with an average confidence of 79.3% on the MNIST (?). In the same setting, a maxout network misclassifies 89.4% of our adversarial examples with an average confidence of 97.6%. Similarly, using $\epsilon = 0.1$, we obtain an error rate of 87.15% and an average probability of 96.6% assigned to the incorrect labels when using a convolutional maxout network on a preprocessed version of the CIFAR-10 (Krizhevsky & Hinton, 2009) test set.

Other simple methods of generating adversarial examples are possible. For example, we also found that rotating $x$ by a small angle in the direction of the gradient reliably produces adversarial examples.

The fact that these simple, cheap algorithms are able to generate misclassified examples serves as evidence in favor of our interpretation of adversarial examples as a result of linearity. The algorithms are also useful as a way of speeding up adversarial training or even just analysis of trained networks.

Perhaps the simplest possible model we can consider is logistic regression. In this case, the fast gradient sign method is exact. We can use this case to gain some intuition for how adversarial examples are generated in a simple setting. See Fig. 2 for instructive images.

If we train a single model to recognize labels $y \in \{1, 1\}$ with $P(y = 1) = w^T x + b$ where $(z) = \log (1 + \exp(z))$ is the softplus function. We can derive a simple analytical form for training on the worst-case adversarial perturbation of $x$ rather than $x$ itself, based on gradient sign.

This is using MNIST pixel values in the interval $[0, 1]$. MNIST data does contain values other than 0 or 1, but the images are essentially binary. Each pixel roughly encodes "ink" or "no ink". This justifies expecting the classifier to be able to handle perturbations within a range of width 0.5, and indeed human observers can read such images without difficulty.

See https://github.com/lisa-lab/pylearn2/tree/master/pylearn2/scripts/papers/maxout for the preprocessing code, which yields a standard deviation of roughly 0.5.
Adversarial Examples [Goodfellow et al. 2015]

First assume an input $x$ is continuous (e.g., in vision) and designing an adversarial example for linear models:

$$\tilde{x} = x + \eta$$

perturbation such that

$$\|\eta\|_\infty < \epsilon$$

effect on (linear) model:

$$w^\top \tilde{x} = w^\top x + w^\top \eta$$

weights
Adversarial Examples [Goodfellow et al. 2015]

\[
\mathbf{w}^\top \tilde{\mathbf{x}} = \mathbf{w}^\top \mathbf{x} + \mathbf{w}^\top \eta
\]

We can maximize the effect of the perturbation by choosing:

\[
\eta = \epsilon \times \text{sign} (\mathbf{w})
\]
Adversarial Examples [Goodfellow et al. 2015]

Use same strategy applied to neural networks. Consider Taylor expansion around $x$

$$f(x + \eta) = f(x) + \frac{\partial f(x)}{\partial x} \eta + \frac{\partial^2 f}{\partial x^2} \eta^2 + \frac{\partial^3 f}{\partial x^3} \eta^3 + \ldots$$

Consider linearization of $f$ (first order approximation):

$$\mathcal{L}_w(x + \eta) \approx \mathcal{L}_w(x) + \frac{\partial \mathcal{L}_w(x)}{\partial x} \eta$$
Adversarial Examples [Goodfellow et al. 2015]

- Fast gradient sign method.
- Consider linearization of loss function (note we are differentiating with respect to the input not the weights):
  \[ L_w(x + \eta) \approx L_w(x) + \frac{\partial L_w(x)}{\partial x} \eta \]
- Choose perturbation:
  \[ \eta = \epsilon \times \text{sign} \left( \frac{\partial L_w(x)}{\partial x} \right) \]
Adversarial Examples [Szegedy et al. 2014, Goodfellow et al. 2015]

Let $\theta$ be the parameters of a model, $x$ the input to the model, $y$ the targets associated with $x$ (for machine learning tasks that have targets) and $J(\theta, x, y)$ be the cost used to train the neural network. We can linearize the cost function around the current value of $\theta$, obtaining an optimal max-norm constrained perturbation of $\epsilon = \epsilon \text{sign} (\nabla_x J(\theta, x, y))$. We refer to this as the "fast gradient sign method" of generating adversarial examples. Note that the required gradient can be computed efficiently using backpropagation.

We find that this method reliably causes a wide variety of models to misclassify their input. See Fig. 1 for a demonstration on ImageNet. We find that using $\epsilon = 0.07$, we cause a shallow softmax classifier to have an error rate of 99.9% with an average confidence of 79.3% on the MNIST (1). In the same setting, a maxout network misclassifies 89.4% of our adversarial examples with an average confidence of 97.6%. Similarly, using $\epsilon = 0.1$, we obtain an error rate of 87.15% and an average probability of 96.6% assigned to the incorrect labels when using a convolutional maxout network on a preprocessed version of the CIFAR-10 (Krizhevsky & Hinton, 2009) test set (2). Other simple methods of generating adversarial examples are possible. For example, we also found that rotating $x$ by a small angle in the direction of the gradient reliably produces adversarial examples.

The fact that these simple, cheap algorithms are able to generate misclassified examples serves as evidence in favor of our interpretation of adversarial examples as a result of linearity. The algorithms are also useful as a way of speeding up adversarial training or even just analysis of trained networks.

A DVERSARIAL TRAINING OF LINEAR MODELS VERSUS WEIGHT DECAY

Perhaps the simplest possible model we can consider is logistic regression. In this case, the fast gradient sign method is exact. We can use this case to gain some intuition for how adversarial examples are generated in a simple setting. See Fig. 2 for instructive images.

If we train a single model to recognize labels $y \in \{1, 1\}$ with $P(y = 1) = \frac{1}{1 + \exp(-\langle w, x \rangle + b)}$ where $(z) = \log (1 + \exp(z))$ is the softplus function. We can derive a simple analytical form for training on the worst-case adversarial perturbation of $x$ rather than $x$ itself, based on gradient sign.

This is using MNIST pixel values in the interval $[0, 1]$. MNIST data does contain values other than 0 or 1, but the images are essentially binary. Each pixel roughly encodes "ink" or "no ink". This justifies expecting the classifier to be able to handle perturbations within a range of width 0.5, and indeed human observers can read such images without difficulty.

See https://github.com/lisa-lab/pylearn2/tree/master/pylearn2/scripts/papers/maxout for the preprocessing code, which yields a standard deviation of roughly 0.5.
Adversarial Examples [Goodfellow et al. 2015]

- One simple way to improve robustness, train with regularizer:

\[
\tilde{\mathcal{L}}_w(x) = \alpha \mathcal{L}_w(x) + (1 - \alpha) \mathcal{L}_w(x + \epsilon \times \text{sign}\left(\frac{\partial \mathcal{L}_w(x)}{\partial x}\right))
\]

regularization hyper parameter
Adversarial Examples [Goodfellow et al. 2015]

- One simple way to improve robustness, train with regularizer:

\[
\tilde{\mathcal{L}}_w(x) = \alpha \mathcal{L}_w(x) + (1 - \alpha) \mathcal{L}_w(x + \epsilon \times \text{sign}(\frac{\partial \mathcal{L}_w(x)}{\partial x}))
\]

regularization hyper parameter

- Tricky situation since this only helps model be robust to these sorts of adversarial examples but not others
Given a question, return an extractive answer (span) by reading a passage.

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called “showers”.

What causes precipitation to fall? gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail? graupel

Where do water droplets collide with ice crystals to form precipitation? within a cloud

Figure 1: Question-answer pairs for a sample passage in the SQuAD dataset. Each of the answers is a segment of text from the passage.
SQuAD v1.1 Leaderboard

Since the release of SQuAD1.0, the community has made rapid progress, with the best models now rivaling human performance on the task. Here are the ExactMatch (EM) and F1 scores evaluated on the test set of v1.1.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Model</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
</table>
| 1    | Human Performance
Stanford University
(Rajpurkar et al. '16) | 82.304 | 91.221 |
| 2    | BERT (ensemble)
Google AI Language
| 2    | BERT (single model)
Google AI Language
https://arxiv.org/abs/1810.04805 | 85.083 | 91.835 |
| 2    | ninet (ensemble)
Microsoft Research Asia | 85.356 | 91.202 |
| 2    | ninet (ensemble)
Microsoft Research Asia | 85.954 | 91.677 |
| 3    | QANet (ensemble)
Google Brain & CMU | 84.454 | 90.490 |
| 4    | r-net (ensemble)
Microsoft Research Asia | 84.003 | 90.147 |
| 5    | QANet (ensemble)
Google Brain & CMU | 83.877 | 89.737 |
| 5    | ninet (single model)
Microsoft Research Asia | 83.468 | 90.133 |
| 5    | MARS (ensemble)
YUANFUDA research NLP | 83.982 | 89.796 |
| 6    | MARS (single model)
YUANFUDA research NLP | 83.185 | 89.547 |
| 7    | r-net+ (ensemble)
Microsoft Research Asia | 82.630 | 88.493 |
| 7    | MARS (single model)
YUANFUDA research NLP | 82.587 | 88.880 |
| 7    | Reinforced Mnemonic Reader + A2D (ensemble model) | 82.849 | 88.764 |

When test set is iid to training set, neural models perform incredibly well.

What about when test distribution deviates from training distribution?

https://rajpurkar.github.io/SQuAD-explorer/
Append an adversarial distracting sentence to the end of the passage.

Article: Super Bowl 50
Paragraph: “Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver’s Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.”
Question: “What is the name of the quarterback who was 38 in Super Bowl XXXIII?”
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

Figure 1: An example from the SQuAD dataset. The BiDAF Ensemble model originally gets the answer correct, but is fooled by the addition of an adversarial distracting sentence (in blue).
Adversarial Question Answering
[Jia and Liang, 2017]

**Article:** Super Bowl 50

**Paragraph:** “Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver’s Executive Vice President of Football Operations and General Manager. *Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*”

**Question:** “What is the name of the quarterback who was 38 in Super Bowl XXXIII?”

**Original Prediction:** John Elway

**Prediction under adversary:** Jeff Dean

<table>
<thead>
<tr>
<th>Possible Input</th>
<th>Image Classification</th>
<th>Reading Comprehension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similar Input</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semantics</th>
<th>Possible Input</th>
<th>Similar Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model’s Mistake</td>
<td>Considers the two to be different</td>
<td>Considers the two to be the same</td>
</tr>
<tr>
<td>Model Weakness</td>
<td>Overly sensitive</td>
<td>Overly stable</td>
</tr>
</tbody>
</table>

Table 1: Adversarial examples in computer vision exploit model oversensitivity to small perturbations. In contrast, our adversarial examples work because models do not realize that a small perturbation can completely change the meaning of a sentence. Images from Szegedy et al. (2014).

Figure 1: An example from the SQuAD dataset. The BiDAF Ensemble model originally gets the answer correct, but is fooled by the addition of an adversarial distracting sentence (in blue).
Adversarial Question Answering
[Jia and Liang, 2017]

**Article:** Super Bowl 50

**Paragraph:** “Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver’s Executive Vice President of Football Operations and General Manager. **Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.**”

**Question:** “What is the name of the quarterback who was 38 in Super Bowl XXXIII?”

**Original Prediction:** John Elway

**Prediction under adversary:** Jeff Dean

<table>
<thead>
<tr>
<th></th>
<th>Image Classification</th>
<th>Reading Comprehension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Possible Input</td>
<td><img src="Image" alt="" /></td>
<td>Tesla moved to the city of Chicago in 1880.</td>
</tr>
<tr>
<td>Similar Input</td>
<td><img src="Image" alt="" /></td>
<td>Tadakatsu moved to the city of Chicago in 1881.</td>
</tr>
<tr>
<td>Semantics</td>
<td>Same</td>
<td>Different</td>
</tr>
<tr>
<td>Model’s Mistake</td>
<td>Considers the two to be different</td>
<td>Considers the two to be the same</td>
</tr>
<tr>
<td>Model Weakness</td>
<td>Overly sensitive</td>
<td>Overly stable</td>
</tr>
</tbody>
</table>

Table 1: Adversarial examples in computer vision exploit model oversensitivity to small perturbations. In contrast, our adversarial examples work because models do not realize that a small perturbation can completely change the meaning of a sentence. Images from Szegedy et al. (2014).

Figure 1: An example from the SQuAD dataset. The BiDAF Ensemble model originally gets the answer correct, but is fooled by the addition of an adversarial distracting sentence (in blue).
Adversarial Question Answering
[Jia and Liang, 2017]

Article: Nikola Tesla
Paragraph: "In January 1880, two of Tesla's uncles put together enough money to help him leave Gospić for Prague where he was to study. Unfortunately, he arrived too late to enroll at Charles-Ferdinand University; he never studied Greek, a required subject; and he was illiterate in Czech, another required subject. Tesla did, however, attend lectures at the university, although, as an auditor, he did not receive grades for the courses." Question: "What city did Tesla move to in 1880?"
Answer: Prague
Model Predicts: Prague

Adversarial AddSent
What city did Tesla move to in 1880?
(Mutate question) Generate fake answer

AddAnother
What city did Tadakatsu move to in 1881?
(Step 3) Convert into statement

Adversary Adds: Tadakatsu moved the city of Chicago to in 1881.
(Step 4) Fix errors with crowdworkers, verify resulting sentences with other crowdworkers

Adversary Adds: Tadakatsu moved to the city of Chicago in 1881.
Model Predicts: Chicago

AddAny
Randomly initialize d words:

- spring attention income getting reached
- Greedily change one word
- spring attention income other reached
- Repeat many times

Adversary Adds: tesla move move other george
Model Predicts: george

Figure 2: An illustration of the ADDSENT and ADDANY adversaries.
Adversarial Question Answering
[Jia and Liang, 2017]

<table>
<thead>
<tr>
<th>Model</th>
<th>Original</th>
<th>ADDSENT</th>
<th>ADDONESENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReasoNet-E</td>
<td>81.1</td>
<td>39.4</td>
<td>49.8</td>
</tr>
<tr>
<td>SEDT-E</td>
<td>80.1</td>
<td>35.0</td>
<td>46.5</td>
</tr>
<tr>
<td>BiDAF-E</td>
<td>80.0</td>
<td>34.2</td>
<td>46.9</td>
</tr>
<tr>
<td>Mnemonic-E</td>
<td>79.1</td>
<td>46.2</td>
<td>55.3</td>
</tr>
<tr>
<td>Ruminating</td>
<td>78.8</td>
<td>37.4</td>
<td>47.7</td>
</tr>
<tr>
<td>jNet</td>
<td>78.6</td>
<td>37.9</td>
<td>47.0</td>
</tr>
<tr>
<td>Mnemonic-S</td>
<td>78.5</td>
<td>46.6</td>
<td>56.0</td>
</tr>
<tr>
<td>ReasoNet-S</td>
<td>78.2</td>
<td>39.4</td>
<td>50.3</td>
</tr>
<tr>
<td>MPCM-S</td>
<td>77.0</td>
<td>40.3</td>
<td>50.0</td>
</tr>
<tr>
<td>SEDT-S</td>
<td>76.9</td>
<td>33.9</td>
<td>44.8</td>
</tr>
<tr>
<td>RaSOR</td>
<td>76.2</td>
<td>39.5</td>
<td>49.5</td>
</tr>
<tr>
<td>BiDAF-S</td>
<td>75.5</td>
<td>34.3</td>
<td>45.7</td>
</tr>
<tr>
<td>Match-E</td>
<td>75.4</td>
<td>29.4</td>
<td>41.8</td>
</tr>
<tr>
<td>Match-S</td>
<td>71.4</td>
<td>27.3</td>
<td>39.0</td>
</tr>
<tr>
<td>DCR</td>
<td>69.3</td>
<td>37.8</td>
<td>45.1</td>
</tr>
<tr>
<td>Logistic</td>
<td>50.4</td>
<td>23.2</td>
<td>30.4</td>
</tr>
</tbody>
</table>

Table 3: ADDSENT and ADDONESENT on all sixteen models, sorted by F1 score the original examples. S = single, E = ensemble.

Table 4: Human evaluation on adversarial examples. Human accuracy drops on ADDSENT mostly due to unrelated errors; the ADDONESENT numbers show that humans are robust to adversarial sentences.
Outline

- Data-to-Text
  - Task description / Approaches
  - Case study (Wiseman et al. 2017)

- Adversarial examples
  - In Vision
  - In NLP
  - Case study (Iyyer et al. 2018)

- GANS and Dual Learning
Generative Adversarial Networks

- Very popular idea in computer vision these days (Currently less popular in NLP)

- A GAN is a type of generative model that can produce samples.

- Alternative to maximum likelihood density estimation.
Generative Modeling

Figure 1: Some generative models perform density estimation. These models take a training set of examples drawn from an unknown data-generating distribution $p_{\text{data}}$ and return an estimate of that distribution. The estimate $p_{\text{model}}$ can be evaluated for a particular value of $\mathbf{x}$ to obtain an estimate $p_{\text{model}}(\mathbf{x})$ of the true density $p_{\text{model}}(\mathbf{x})$. This figure illustrates the process for a collection of samples of one-dimensional data and a Gaussian model.

[Goodfellow 2017]
Generative Modeling

Generative Modeling

• Density estimation
• Sample generation

Figure 1: Some generative models perform density estimation. These models take a training set of examples drawn from an unknown data-generating distribution \( p_{\text{data}} \) and return an estimate of that distribution. The estimate \( p_{\text{model}} \) can be evaluated for a particular value of \( x \) to obtain an estimate \( p_{\text{model}}(x) \) of the true density \( p_{\text{data}} \).

This figure illustrates the process for a collection of samples of one-dimensional data and a Gaussian model.

Generative Modeling

• Density estimation
• Sample generation

Figure 2: Some generative models are able to generate samples from the model distribution. In this illustration of the process, we show samples from the ImageNet (Deng et al., 2009, 2010; Russakovsky et al., 2014) dataset. An ideal generative model would be able to train on examples as shown on the left and then create more examples from the same distribution as shown on the right. At present, generative models are not yet advanced enough to do this correctly for ImageNet, so for demonstration purposes this figure uses actual ImageNet data to illustrate what an ideal generative model would produce.

[Goodfellow 2017]
Generative Adversarial Networks

- Assume a dataset $M$ of images

- Adversarial Game with two players
  - Generator(G): Given noise as input, generate an image $x_G$ from the same distribution as $M$
  - Discriminator(D): Given $M$ and an image $x$ determine if $x$ is from the same distribution as $M$

- The discriminator tries to minimize its loss
- The generator tries to maximize the discriminator’s loss by producing really good images (atleast we hope)

[Goodfellow 2017]
Generative Adversarial Networks

Figure 12: The GAN framework pits two adversaries against each other in a game. Each player is represented by a differentiable function controlled by a set of parameters. Typically these functions are implemented as deep neural networks. The game plays out in two scenarios. In one scenario, training examples $x$ are randomly sampled from the training set and used as input for the first player, the discriminator, represented by the function $D$. The goal of the discriminator is to output the probability that its input is real rather than fake, under the assumption that half of the inputs it is ever shown are real and half are fake. In this first scenario, the goal of the discriminator is for $D(x)$ to be near 1. In the second scenario, inputs $z$ to the generator are randomly sampled from the model’s prior over the latent variables. The discriminator then receives input $G(z)$, a fake sample created by the generator. In this scenario, both players participate. The discriminator strives to make $D(G(z))$ approach 0 while the generative strives to make the same quantity approach 1. If both models have sufficient capacity, then the Nash equilibrium of this game corresponds to the $G(z)$ being drawn from the same distribution as the training data, and $D(x) = \frac{1}{2}$ for all $x$.

[Goodfellow 2017]
Generative Adversarial Networks

G = generator (some neural network)
D = discriminator (some neural network)
z = noise vector (input to Generator)
p_{data} = true data generating distribution

Example discriminator loss:

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

all examples from true distribution are positive
all examples from \( G \) are negative

[Goodfellow 2017]
Generative Adversarial Networks

\[ G = \text{generator (some neural network)} \]

\[ D = \text{discriminator (some neural network)} \]

\[ z = \text{noise vector (input to Generator)} \]

\[ p_{\text{data}} = \text{true data generating distribution} \]

Example generator loss:

\[
\mathcal{L}_G = -\frac{1}{2} \mathbb{E}_z \log(D(G(z)))
\]

\[ \text{all examples from } G \text{ are positive} \]

[Goodfellow 2017]
Challenges in GANs

- GANs are very unstable to train and easily degenerate.

- For instance the generator can simply “collapse” and only produce a single really good image that fools the discriminator.

- This is can occur because vanilla GANs are not really conditional models and so are not required to produce different outputs given a different input.
Figure 1: Given any two unordered image collections $X$ and $Y$, our algorithm learns to automatically “translate” an image from one into the other and vice versa: (left) Monet paintings and landscape photos from Flickr; (center) zebras and horses from ImageNet; (right) summer and winter Yosemite photos from Flickr. Example application (bottom): using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.
CycleGANs [Zhu et al. 2017]

Figure 2: Paired training data (left) consists of training examples \( \{x_i, y_i\}_{i=1}^{N} \), where the correspondence between \( x_i \) and \( y_i \) exists [22]. We instead consider unpaired training data (right), consisting of a source set \( \{x_i\}_{i=1}^{N} \) \( (x_i \in X) \) and a target set \( \{y_j\}_{j=1}^{N} \) \( (y_j \in Y) \), with no information provided as to which \( x_i \) matches which \( y_j \).
Figure 3: (a) Our model contains two mapping functions $G : X \to Y$ and $F : Y \to X$, and associated adversarial discriminators $D_Y$ and $D_X$. $D_Y$ encourages $G$ to translate $X$ into outputs indistinguishable from domain $Y$, and vice versa for $D_X$ and $F$. To further regularize the mappings, we introduce two cycle consistency losses that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \to G(x) \to F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \to F(y) \to G(F(y)) \approx y$. 

**CycleGANs** [Zhu et al. 2017]
Back to NLP

- GANs haven’t achieved breakthroughs in NLP (yet)

- However, the idea of “cycle-constraints” is very powerful.

- In particular unsupervised machine translation [Ravi and Knight 2011]
  - Given unaligned corpora in source and target languages, build a translation system :)
General Idea (Back Translation) [Sennrich et al. 2015, He et al. 2016, Lample 2017/2018]

Comos estas?  \[\rightarrow\]  ?

Spanish  \[\rightarrow\]  French

no supervision
General Idea (BackTranslation with Language Model Reward) [Sennrich et al. 2015, He et al. 2016, Lample 2017/2018]

Comos estas?

but we have a really good language model

Spanish

French

?
General Idea

Attempt 1: Run source through seq2seq and score result based on target language model.

Comos estas?

What is wrong with this?
General Idea

“Back translate” target back to source and try to recover original sentence.

Comos estas?

loss on source recovery

target language model loss

Spanish

French
Phrase/Neural Unsupervised Machine Translation
[Lample et al. 2018]

Figure 2: Comparison between supervised and unsupervised approaches on WMT'14 En-Fr, as we vary the number of parallel sentences for the supervised methods.

Table 2: **Comparison with previous approaches.** BLEU score for different models on the \(en \rightarrow fr\) and \(en \rightarrow de\) language pairs. Just using the unsupervised phrase table, and without back-translation (PBSMT (Iter. 0)), the PBSMT outperforms previous approaches. Combining PBSMT with NMT gives the best results.

<table>
<thead>
<tr>
<th>Model</th>
<th>en-fr</th>
<th>fr-en</th>
<th>en-de</th>
<th>de-en</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Artetxe et al., 2018)</td>
<td>15.1</td>
<td>15.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(Lample et al., 2018)</td>
<td>15.0</td>
<td>14.3</td>
<td>9.6</td>
<td>13.3</td>
</tr>
<tr>
<td>(Yang et al., 2018)</td>
<td>17.0</td>
<td>15.6</td>
<td>10.9</td>
<td>14.6</td>
</tr>
<tr>
<td>NMT (LSTM)</td>
<td>24.5</td>
<td>23.7</td>
<td>14.7</td>
<td>19.6</td>
</tr>
<tr>
<td>NMT (Transformer)</td>
<td>25.1</td>
<td>24.2</td>
<td>17.2</td>
<td>21.0</td>
</tr>
<tr>
<td>PBSMT (Iter. 0)</td>
<td>16.2</td>
<td>17.5</td>
<td>11.0</td>
<td>15.6</td>
</tr>
<tr>
<td>PBSMT (Iter. n)</td>
<td><strong>28.1</strong></td>
<td>27.2</td>
<td>17.9</td>
<td>22.9</td>
</tr>
<tr>
<td>NMT + PBSMT</td>
<td>27.1</td>
<td>26.3</td>
<td>17.5</td>
<td>22.1</td>
</tr>
<tr>
<td>PBSMT + NMT</td>
<td>27.6</td>
<td><strong>27.7</strong></td>
<td><strong>20.2</strong></td>
<td><strong>25.2</strong></td>
</tr>
</tbody>
</table>

**Wowwwww!!! :)**
Stay in touch

ankur.p.parikh@gmail.com