Lecture-07: Representation and Distributional Learning  (Deep Learning & AI @LMU, Germany)

Speaker: Pankaj Gupta

PhD Student (Advisor: Prof. Hinrich Schütze) CIS, University of Munich (LMU)
Research Scientist (NLP/Deep Learning), Machine Intelligence, Siemens AG | Nov 2018
Lecture Outline

- **Motivation**: Distributed Feature Representations: What/Why?
- Strategies to obtain Feature Representation via Generative Models
  - Auto-encoders
  - Restricted Boltzmann Machines (RBMs) and Deep variants (DBM)
  - Neural Autoregressive Model: NADE, DocNADE, etc.
- **Language Modeling for Distributional Semantics**:
  (Tools: Word2vec, RNN-LM, RecvNN, textTOvec, etc.)
- Metric Learning (e.g. Siamese Networks) for Textual Similarity
- **Compositionality**: Recurrent vs Recursive (overview) (already covered in lecture 05)
- Multi-task Learning (overview) (already covered in lecture 05)
Motivation: Representation Learning

¿ What is Representation Learning? ¿
¿ Why do we need Representation Learning? ¿
¿ How to form good Representations?
Motivation: Representation Learning

Traditional Machine Learning

**VISION**
- Hand-crafted features (SIFT/HOG)
- Fixed
- Your favorite classifier
- Learned
- "car"

**SPEECH**
- Hand-crafted features (MFCC)
- Fixed
- Your favorite classifier
- Learned
- "d e p"

**NLP**
- Hand-crafted features (Bag-of-words)
- Fixed
- Your favorite classifier
- Learned
- "+

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun
Motivation: Representation Learning

Hierarchical Compositionality

VISION
- pixels ➔ edge ➔ texton ➔ motif ➔ part ➔ object

SPEECH
- sample ➔ spectral band ➔ formant ➔ motif ➔ phone ➔ word

NLP
- character ➔ word ➔ NP/VP/.. ➔ clause ➔ sentence ➔ story

Feature visualization of convolutional.net trained on ImageNet from [Zeiler & Fergus 2013]

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun
Motivation: Representation Learning

Exploiting compositionality gives an exponential gain in representation power-

(1) Distributed representations / embeddings
   → feature learning

(2) Deep architectures
   → multiple levels of feature learning

Compositionality to describe the world around us efficiently
Motivation: Representation Learning

Feature Learning
Representation Learning

cuisine_italian
cuisine_wheat
content_cheese
content_tomato
content_pepper
content_wheat

Learning Algorithm (e.g., metric learning)

Food
Hungry
Burger
pizzeria
Motivation: Representation Learning

Representation Learning (also known as feature learning)
Techniques to automatically discover the representations needed
→ for feature detection that explains the data,
→ classification from raw data

Feature
→ measurable property or characteristic of a phenomenon being observed
E.g.,
- shape, size, color, etc. of objects
- content, cuisine, color, etc. of food items
- syntactic and semantic relationships in words
Motivation: Representation Learning

- RAW DATA: e.g., images, videos, text, etc.
- Representation Learning: Feature Disentangling
- Learning algorithm: Performing tasks/actions
Why: Representation Learning

Collection of similar objects, based on shared [latent] features
Why: Representation Learning

Word Analogy in distributional semantic vector space
Why: Representation Learning

Vector composition to determine semantic similarity in texts!!

HTTPS://WWW.SPRINGBOARD.COM/BLOG/INTRODUCTION-WORD-EMBEDDINGS/
Why: Representation Learning

Vector composition to determine semantic similarity in phrases or sentences!!

Country of my birth
Place where I was born
Global warming impact

A woman is behind the door
A man was front of the window
Lion is the king of jungle

HTTPS://WWW.SPRINGBOARD.COM/BLOG/INTRODUCTION-WORD-EMBEDDINGS/
Motivation: Representation Learning

How about **CONTEXT** in feature representation learning?

Food
Motivation: Representation Learning

How about CONTEXT in feature representation learning?

HALWA

meaning?
Motivation: Representation Learning

How about CONTEXT in feature representation learning?

I am very hungry, I will eat HALWA
Motivation: Representation Learning

How about CONTEXT in feature representation learning?

I am very hungry, I will eat HALWA
Motivation: Representation Learning

How about CONTEXT in feature representation learning?

Is it a good idea to eat HALWA after a meal?
Motivation: Representation Learning

How about CONTEXT in feature representation learning?

Is it a good idea to eat HALWA after a meal?
Motivation: Representation Learning

How about CONTEXT in feature representation learning?

I put a cup of sugar too much in cooking HALWA?
Why: Representation Learning?

Unsupervised pre-training and Transfer learning

Training deep networks can be challenging

→ Non-convex loss function
→ Inappropriate initializers

→ Initialisation is critical in Neural network training
→ Using appropriate initializers for better convergence

Loss function surfaces
Why: Representation Learning?

**Unsupervised pre-training and Transfer learning**

*Transfer knowledge from previous learning:*

→ Representations or features
→ Explanatory factors

Training deep networks can be challenging

↓

initialize hidden layers using UNSUPERVISED learning

↓

encode latent structure of input distribution in hidden layer

Previous learning from unlabeled data + labels for other tasks

→ *Prior: shared* underlying *explanatory factors*, in particularly between \( P(x) \) and \( P(Y|x) \)

Unsupervised pre-training and Transfer learning

transfer knowledge from previous

→ Representations
→ Explanatory factors

Example: Image recognition model

Unsupervised pre-training with unlabeled data to learn the representations of different levels of abstraction

Transfer the knowledge

Supervised Learning with available labeled data

car

... Human

Motivation: Good Representation

- **Good** features for *successful* machine learning,
  
e.g., \( \text{man} \leftrightarrow \text{human} \), \( \text{cat} \leftrightarrow \text{dog} \),
  \( \text{buy} \leftrightarrow \text{bought} \), \( \text{buy} \leftrightarrow \text{acquired} \) \( \text{Islam} \leftrightarrow \text{Christianity} \), etc.

- **Knowing** features belief about objects in *prior*,
  e.g., *features of car* \( \rightarrow \) *has_wheel, has_glasses, is_automobile, relatedto_manufacturer*, etc.

- **Handcrafting** features vs *automatic* feature learning

- Representation learning: *Estimate features / factors / causes* that explains the data \( \rightarrow \) *good representation*
  
i.e., *good representation* captures *factors of variation* that best explains the data

- *Learning representations from representations* \( \rightarrow \) **Representation learning**
  
e.g., autoencoders, RBMs, RSMs, NADE, DocNADE, iDocNADEe, generative RNNs, encoder-decoders, etc.
Motivation: Distributed Representation Learning

Local Representations vs Distributed Representations?
Motivation: Local Representation of Symbols

Consider a sequence $s$ of words: "( a cat catches a mouse )"

A set of symbols is given by, $D = \{ \text{mouse, cat, a, catches, (, ) } \}$

Given a set of symbols $D$,
a local representation maps the $i$-th symbol in $D$ to
the $i$-th unit vector $e_i$ of real values of $n$ dimension,
where $n$ is the cardinality of $D$.

Hence, the $i$-th unit vector represents the $i$-th symbol.

$$
\begin{align*}
\text{mouse} & \rightarrow e_1 = (1 \ 0 \ 0 \ 0 \ 0 \ 0)^T \\
\text{cat} & \rightarrow e_2 = (0 \ 1 \ 0 \ 0 \ 0 \ 0)^T \\
\text{a} & \rightarrow e_3 = (0 \ 0 \ 1 \ 0 \ 0 \ 0)^T \\
\text{catches} & \rightarrow e_4 = (0 \ 0 \ 0 \ 1 \ 0 \ 0)^T \\
( & \rightarrow e_5 = (0 \ 0 \ 0 \ 0 \ 1 \ 0)^T \\
) & \rightarrow e_6 = (0 \ 0 \ 0 \ 0 \ 0 \ 1)^T
\end{align*}
$$
Motivation: Local Representation of Symbols

Consider a sequence $s$ of words: “( a cat catches a mouse )”

A set of symbols is given by, $D = \{\text{mouse, cat, a, catches, (, )} \}$

Local Representations

- A sequence of vectors
- A bag-of-symbols

- A sequence of vectors representing the symbols in the sequence
- Used in recurrent neural networks
- A sequence is represented with one vector generally obtained with a weighted sum of vectors representing symbols, i.e., orderless
- SVM (used in Information Retrieval task)
Motivation: Local Representation of Symbols

Limitations of Local Representation

Local Representations

- a sequence of vectors
  - Each vector has a one-to-one mapping to a symbol
  - Too sparse, extremely inefficient for a large symbol set

- a bag-of-symbols
  - Symbolic sequences cannot be fully reconstructed
    but it is possible to know which symbols were in the sequence.
  - Does not preserve sequence order
Motivation: Local Representation of Symbols

Limitations of Local Representation

One-hot vector

No information about words semantics

Motivation: Local Representation of Symbols

Limitations of Local Representation

(i.e., dense vectors in distributed representations)

No information about words semantics

(1 0 0 0 0) · (0 1 0 0 0) = 0.80

mouse

similarity

0.0

cat

animal

size

noun

animal

size

noun

0.4 0.7 0.1

0.5 0.8 0.9
Motivation: Local Representation of Symbols

Limitations of Local Representation

No information about words semantics

How to obtain the distributed representation?

(i.e., dense vectors in distributed representations)
Motivation: Distributed Representation Learning

What is Distributed Representation Learning?

**Distributed**: “information is encoded by a multiplicity of features / factors / causes”

**Distributed representations**: 

→ vectors or tensors in metric spaces  
→ transformations of the data that *compactly* capture many different factors of variations  
→ underlying learning models are neural networks  

E.g. *Distributed word representations*:

Each word is represented as a dense and real-valued vector in low dimensional space, and each latent feature encodes syntactic and semantic relatedness information  
→ addresses the *Curse of Dimensionality*
Need for Distributed Representation Learning

The need for distributed representations

- Factor models, PCA, RBMs, Neural Nets, Sparse Coding, Deep Learning, etc.
- Each parameter influences many regions, not just local neighbors
- # of distinguishable regions grows almost exponentially with # of parameters
- GENERALIZE NON-LOCALLY TO NEVER-SEEN REGIONS

Inspired from Y. Bengio summer school, 2015
Motivation: Distributed Representation Learning

→ dense vectors in distributed representations

→ one concept represented by the dense vector
→ one dimension per property

→ enable to share similarity between more than two concepts
Motivation: Distributed Representation Learning

Power of Distributed Representations

Distributed Representation by Hinton (1984)
Representations Learning

→ Can we interpret each dimension/property?
→ Lack of interpretability in dense representations learned by Deep Learning
→ Hard to track down what’s failing

→ Hierarchical composition
→ Deep Learning is very good !!!
Distributed Representations in Deep Learning !!!
Yes it works, but how?

Representation learning: Attempts to \textit{automatically learn good features or representations}
Deep Learning: Attempts to learn \textit{multiple levels of representations} of increasing abstraction
Distributed Representation Learning in Neural Networks

➔ Key concept in neural networks: **Distributed Representation Learning**

➔ Key questions:

• How can a neural network be so effective representing objects when it has only a few hidden units (i.e. much fewer units than possible objects)?

• What is each hidden unit actually representing?

• How can a neural network generalize to objects that is has never seen before?
Distributed Representations in Deep Learning

Deep Learning = Hierarchical Compositionality

- Cascade of non-linear transformations
- Multiple layers of representations
Distributed Representations in Deep Learning

Deep Learning = Hierarchical Compositionality

– Cascade of non-linear transformations
– Multiple layers of representations

→ No single neuron “encodes” everything
→ Groups of neurons work together
Representation Learning: Supervised vs Unsupervised

**Supervised Learning**

**Data:** (x, y)
x is data, y is label

**Goal:** Learn a *function* to map x $\rightarrow$ y

**Examples:** Classification, regression, object detection, semantic segmentation, image captioning, etc.

https://www.cc.gatech.edu/classes/AY2019/cs7643_fall/
Representation Learning: Supervised vs Unsupervised

Unsupervised Learning

Data: \( x \)
Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

https://www.cc.gatech.edu/classes/AY2019/cs7643_fall/
Unsupervised Learning

**Data**: x
Just data, no labels!

**Goal**: Learn some underlying hidden *structure* of the data

**Examples**: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Principal Component Analysis (Dimensionality reduction)

https://www.cc.gatech.edu/classes/AY2019/cs7643_fall/
Unsupervised Learning

**Data:** x

Just data, no labels!

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, feature learning, density estimation, etc.

https://www.cc.gatech.edu/classes/AY2019/cs7643_fall/
Representation Learning: Supervised vs Unsupervised

Supervised Learning

**Data:** \((x, y)\)
\(x\) is data, \(y\) is label

**Goal:** Learn a *function* to map \(x \rightarrow y\)

**Examples:** Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

**Data:** \(x\)
Just data, no labels!

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, feature learning, density estimation, etc.

https://www.cc.gatech.edu/classes/AY2019/cs7643_fall/
Generative Models for Representation Learning

**Generative Classification**

→ Model $p(x, y)$; estimate $p(x|y)$ and $p(y)$
→ Use Bayes Rule to predict $y$, e.g. Naïve Bayes

**Discriminative Classification**

→ Estimate $p(y|x)$ directly, e.g. Logistic Regression, CNN, RNN, etc.

**Density Estimation**

→ Model $p(x)$, e.g. RBMs, VAEs, NADE, RSM, DocNADE, etc.
Given training data, generate new samples from same distribution

Training data ~ $p_{\text{data}}(x)$

Generated samples ~ $p_{\text{model}}(x)$

Want to learn $p_{\text{model}}(x)$ similar to $p_{\text{data}}(x)$

**Density estimation: a core problem in unsupervised learning**

Several flavors:

→ Explicit density estimation: explicitly define and solve for $p_{\text{model}}(x)$

→ Implicit density estimation: learn model that can sample from $p_{\text{model}}(x)$ w/o explicitly defining it

Training generative models can enable inference of latent representations, used as general features

Given training data, generate new samples from same distribution

Unsupervised Representation Learning to exploit tons to unlabeled data

Training data ~ $p_{\text{data}}(x)$
Generated samples ~ $p_{\text{model}}(x)$

Want to learn $p_{\text{model}}(x)$ similar to $p_{\text{data}}(x)$

Density estimation: a core problem in unsupervised learning

Several flavors:
→ Explicit density estimation: explicitly define and solve for $p_{\text{model}}(x)$
→ Implicit density estimation: learn model that can sample from $p_{\text{model}}(x)$ w/o explicitly defining it

Training generative models can enable inference of latent representations, used as general features

On causal and anticausal learning, (Janzing et al ICML 2012)

- If Ys of interest are among the causal factors of X, then
  \[ P(Y | X) = \frac{P(X | Y)P(Y)}{P(X)} \]
  is tied to P(X) and P(X | Y), and P(X) is defined in terms of P(X | Y), i.e.
- The best possible model of X (unsupervised learning) MUST involve Y as a latent factor, implicitly or explicitly.
- Representation learning SEEKS the latent variables H that explain the variations of X, making it likely to also uncover Y.

Manifolds

→ probability mass concentrates near regions that have a much smaller dimensionality than the original space where the data lives

Manifold Learning via Generative Models

Learning complex and useful data projections

Reconstruction Objective

→ train a network to learn weights such as the network can reconstruct the input, given the output (e.g., hidden vector encoding the input)!!!
Learning Distributed Word Representations (i.e., word embeddings)
Distributed word representation

→ represent (embed) words in a continuous vector space where semantically similar words are mapped to nearby points
→ describe meaning of words and sentences with vectorial representations

Idea1: “you shall judge a word by the company it keeps”
Idea2: *Distributional hypothesis:* “words have similar meaning if used in similar contexts”
Why: Distributed Word Representations

→ words sharing similar attributes are similar

E.g., *dog* is more similar to *cat* than to *car* as *dog* and *cat* share more attributes than *dog* and *car*

→ *word-to-word* matrices obtained by observing n-word windows of target words

![co-occurrence matrix](WWW.TENSORFLOW.ORG/IMAGES/LINEAR-RELATIONSHIPS.PNG)

Small corpus of 3 sentence/documents

| \( s_1 \) | a cat catches a mouse |
| \( s_2 \) | a dog eats a mouse |
| \( s_3 \) | a dog catches a cat |

\[
X = \begin{pmatrix}
2 & 2 & 2 \\
1 & 0 & 1 \\
0 & 1 & 1 \\
1 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 0
\end{pmatrix}
\]

Local representation: BoW

Distributed representation:
a word-to-word matrix considering a 1-word window of context
Why: Distributed Word Representations

→ words sharing similar attributes are similar.

E.g., *dog* is more similar to *cat* than to *car* as *dog* and *cat* share more attributes than *dog* and *car*.

→ word-to-word matrices obtained by observing n-word windows of target words

Small corpus of 3 sentence/documents

Local representation: BoW

Distributed representation: a word-to-word matrix considering a 1-word window of context
Why: Distributed Word Representations

→ words sharing similar attributes are similar.

E.g., *dog* is more similar to *cat* than to *car* as *dog* and *cat* share more attributes than *dog* and *car*.

→ word-to-word matrices obtained by observing n-word windows of target words

Small corpus of 3 sentence/documents

<table>
<thead>
<tr>
<th>s1</th>
<th>a cat catches a mouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>s2</td>
<td>a dog eats a mouse</td>
</tr>
<tr>
<td>s3</td>
<td>a dog catches a cat</td>
</tr>
</tbody>
</table>

**Local representation: BoW**

\[
X = \begin{pmatrix}
  s_1 & s_2 & s_3 \\
  a & 2 & 2 \\
  cat & 1 & 0 \\
  catches & 0 & 1 \\
  a mouse & 1 & 1 \\
  eats & 1 & 0 \\
  a dog & 0 & 1 \\
  catches a cat & 0 & 1 \\
\end{pmatrix}
\]

**Distributed representation:**

A word-to-word matrix considering a 1-word window of context

\[
\begin{pmatrix}
  a & cat & dog & mouse & catches & eats \\
  0 & 1 & 2 & 2 & 2 & 2 \\
  2 & 0 & 0 & 0 & 1 & 0 \\
  2 & 0 & 0 & 0 & 1 & 1 \\
  2 & 0 & 0 & 0 & 0 & 0 \\
  2 & 1 & 1 & 0 & 0 & 0 \\
  1 & 0 & 1 & 0 & 0 & 0 \\
\end{pmatrix}
\]
Why: Distributed Word Representations

→ words sharing similar attributes are similar.

E.g., *dog* is more similar to *cat* than to *car* as *dog* and *cat* share more attributes than *dog* and *car*.

→ word-to-word matrices obtained by observing n-word windows of target words

**co-occurrence matrix**

Small corpus of 3 sentence/documents

| s₁ | a cat catches a mouse |
| s₂ | a dog eats a mouse    |
| s₃ | a dog catches a cat   |

\[ X = \begin{pmatrix}
  a & cat & dog & mouse & catches & eats \\
  2 & 0 & 0 & 0 & 1 & 0 \\
  2 & 0 & 0 & 0 & 1 & 1 \\
  2 & 0 & 0 & 0 & 0 & 0 \\
  2 & 1 & 1 & 0 & 0 & 0 \\
  1 & 0 & 1 & 0 & 0 & 0 
\end{pmatrix} \]

Distributed representation: a word-to-word matrix considering a 1-word window of context

WWW.TENSORFLOW.ORG/IMAGES/LINEAR-RELATIONSHIPS.PNG
Why: Distributed Word Representations

- words sharing similar attributes are similar. 
E.g.,
- dog is more similar to cat than to car as dog and cat share more attributes than dog and car.

→ word-to-word matrices obtained by observing n-word windows of target words

Small corpus of 3 sentence/documents

Distributed representation: a word-to-word matrix considering a 1-word window of context

However, original co-occurrence matrix is very costly to obtain and store

→ Need compact distributed vectors
Learning Compact Distributed Word Representations

**Word2vec:** Tool to generate Word embeddings (i.e., distributed word representation) using large corpus

→ represent (embed) words in a continuous vector space

where semantically similar words are mapped to nearby points

→ uses contextual information to learn word vectors

→ neural network predicts a target word from the words surrounding it (context)

→ no explicitly co-occurrence matrix

→ no explicit association between word pairs

→ distribution of the words learned implicitly

→ compact distributed representation

→ dimensions of vectors not interpretable
Why: Power of Distributed Word Representation

Let’s TRY !!

Further reading: doc2vec, LDA, LSA, etc.
BREAK

Family of Generative Models

Approximate Density
- e.g.,
  - Variational Autoencoders,
  - RBMs, DBN, DBM,
  - RSM
  - RNN-RSM

Tractable Density
- e.g.,
  - NADE, MADE
  - DocNADE
  - iDocNADEe
  - PixelRNN/CNN

Strategies to obtaining Distributed Representations

**Family of Generative Models**

- Strategies to obtaining Distributed Representations

**Approximate Density**
- e.g.,
  - Variational Autoencoders,
  - RBMs, DBN, DBM,
  - RSM
  - RNN-RSM

**Tractable Density**
- e.g.,
  - NADE, MADE
  - DocNADE
  - iDocNADEe
  - PixelRNN/CNN

we will discuss the highlighted ones
Auto-encoders (AEs)

→ a feed-forward neural network trained to reproduce its input at the output layer

**Encoder:** compresses the input into a latent-space representation

\[ h(x) = g(Wx + b) \]

Feature-vector or representation

**Decoder:** reconstruct the input from the latent representation

\[ \hat{x} = W^T h(x) + c \]

Reconstructed input

**Loss function** (for real-valued inputs): (minimizing reconstruction error by SGD)

\[ L(\hat{x}; x) = \frac{1}{2} ||\hat{x} - x||^2 \]

Measure discrepancy b/w input and its reconstruction over training samples

Auto-encoders (AEs)

Captures useful properties of the data distribution or means to represent data or explanatory factors.

- $\text{dimension}(h) \lt \text{dimension}(x)$ → **undercomplete** i.e. learn to capture useful features
- $\text{dimension}(h) \gt \text{dimension}(X)$ → **overcomplete** i.e. learn copy input to output
- Difficult to interpret
Auto-encoders (AEs)

\[ x \rightarrow \text{Encoder} \rightarrow \hat{h} \rightarrow \text{Decoder} \rightarrow \hat{x} \]

- **Undercomplete AE** → capture useful info about data
- **Overcomplete AE** → still captures interesting features, but apply constraints on \( X \) or \( h \)

→ Difficult to interpret
Benefits of Auto-encoders (AEs)

→ a feed-forward neural network trained to reproduce its input at the output layer

Key Facts about AEs:

→ unsupervised ML algorithm, similar to PCA
→ neural network’s target output is its input
→ learn latent features/encoding of input (no manual feature engineering)
→ represent both linear and non-linear transformation in encoding
→ layered to form deep learning network, i.e., distributed representations
→ tractable / easier optimization
→ applications in denoising and dimensionality reduction (dense representation)
→ powerful non-linear (i.e., non-linear encoding and decoding) generalization of PCA

http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/
Auto-encoders (AEs)

Limitations of (regular) Autoencoders

→ Need to Conceptualize, not only compress………

Different variants to rescue……………….!!!
Auto-encoders (AEs) variants

Autoencoder variants:

1. *Denoising* Autoencoders
2. *Sparse* Autoencoders
3. *Convolutional* Autoencoders
4. *Variational* Autoencoders (VAE)
5. *Contractive* Autoencoders (CAE)
6. Stacked Autoencoders, etc…

---

Idea: *Constraint the reconstruction of an autoencoder*

……..*details later in the lecture series (Lecture on “Generative Models”)*
Family of Neural Generative Models
Family of Generative Models

- Directed graphical models
  - define prior over top-most latent representation
  - define conditionals from top latent representation to observation
    \[ p(x, h^{(1)}, h^{(2)}, h^{(3)}) = p(x|h^{(1)})p(h^{(1)}|h^{(2)})p(h^{(2)}|h^{(3)})p(h^{(3)}) \]
  - examples: variational autoencoders (VAE), generative adversarial networks (GAN), sparse coding, helmholtz machines

- Properties
  - pros: easy to sample from (ancestral sampling)
  - cons: \( p(x) \) is intractable, so hard to train

(details in Lecture on “Generative Models”)
Family of Generative Models

- Undirected graphical models
  - Define a joint energy function
    \[ E(x, h^{(1)}, h^{(2)}, h^{(3)}) = -xW^{(1)}h^{(1)} - h^{(2)}W^{(2)}h^{(3)} - h^{(3)}W^{(3)}h^{(4)} \]
  - Exponentiate and normalize
    \[ p(x, h^{(1)}, h^{(2)}, h^{(3)}) = \exp\left(-E(x, h^{(1)}, h^{(2)}, h^{(3)})\right) / Z \]
  - Examples: deep Boltzmann machines (DBM), deep energy models

- Properties
  - Pros: can compute \( p(x) \) up to a multiplicative factor (true for RBMs not general BMs)
  - Cons: hard to sample from (MCMC), \( p(x) \) is intractable, so hard to train

https://ift6135h18.wordpress.com/author/aaroncourville/page/1/
Family of Generative Models

- **Autoregressive generative models**
  - choose an ordering of the dimensions in $x$
  - define the conditionals in the product rule expression of $p(x)$
    $$p(x) = \prod_{k=1}^{D} p(x_k|x_{<k})$$
  - examples: masked autoencoder distribution estimator (MADE), pixelCNN, neural autoregressive distribution estimator (NADE), spatial LSTM, pixelRNN

- **Properties**
  - **Pros:** $p(x)$ is tractable, so easy to train, easy to sample (though slower)
  - **Cons:** doesn’t have a natural latent representation

https://ift6135h18.wordpress.com/author/aaroncourville/page/1/
Undirected (Generative) Probabilistic Graphical Models
Restricted Boltzmann Machines (RBMs)

→ undirected *probabilistic* graphical model
→ unsupervised *stochastic extractor* of *binary* features \((h)\)
→ trained using *reconstruction* objective
→ transform data into latent feature space and then reconstruct to learn data distribution
→ *Two* layers: *observed* or *visible* \((v)\) and *latent hidden* \((h)\) layer
→ both *visible* and *hidden* are *binary*
→ *energy-based models*,

Therefore, *joint probability distribution* is given by its energy function:

\[
P(v, h) = \frac{1}{Z} \exp \left\{ -E(v, h) \right\}.
\]

Further reading: [https://ift6266h15.files.wordpress.com/2015/03/chapter21.pdf](https://ift6266h15.files.wordpress.com/2015/03/chapter21.pdf)
Restricted Boltzmann Machines (RBMs)

→ Two layers: binary **visible** \((v)\) and binary **hidden** \((h)\) layer

→ **energy-based models**, 

Therefore, *joint probability distribution* is given by its energy function:

\[
P(v, h) = \frac{1}{Z} \exp \{-E(v, h)\}.
\]

→ **energy function** that parameterizes the relationship between the visible and hidden variables

\[
E(v, h) = -b^T v - c^T h - v^T W h
\]

→ normalizing constant known as the **partition function**

\[
Z = \sum_v \sum_h \exp \{-E(v, h)\}
\]

E.g., a document of 2000 unique words,

\[
2^{\text{dim}(v)} 2^{\text{dim}(h)} \approx 2^{2000} 2^{50} \approx \text{intractable}
\]

*summing over all states* \(\Rightarrow\) *summing exponential number of terms* \(\Rightarrow\) *computationally intractable* \(P(v,h)\)
Restricted Boltzmann Machines (RBMs)

→ energy-based models,

Therefore, joint probability distribution is given by its energy function:

→ energy function,

\[ E(v, h) = -b^T v - c^T h - v^T W h \]

→ partition function

\[ Z = \sum_v \sum_h \exp \{-E(v, h)\} \]

→ conditional distributions from the joint distribution

Full conditional over the hidden layer as the factorial distribution:

\[ p(h|v) = \prod_{j=1}^{n} p(h_j|v) = \prod_{j=1}^{n} \text{sigmoid}(c_j + v^T W_{.,j}) \]

Full conditional over the visible layer as the factorial distribution:

\[ p(v|h) = \prod_{i=1}^{d} p(v_i|h) = \prod_{i=1}^{d} \text{sigmoid}(b_i + W_{i,:}h) \]
Restricted Boltzmann Machines (RBMs)

→ energy-based models,
Therefore, joint probability distribution is given by its energy function:

→ energy function,

\[ E(v, h) = -b^T v - c^T h - v^T W h \]

→ partition function

\[ Z = \sum_v \sum_h \exp \{-E(v, h)\} \]

→ conditional distributions from the joint distribution

Full conditional over the hidden layer as the factorial distribution:

\[ p(h | v) = \prod_{j=1}^{n} p(h_j | v) = \prod_{j=1}^{n} \text{sigmoid}(c_j + v^T W_{:,j}) \]
Restricted Boltzmann Machines (RBMs)

- energy-based models,

Therefore, joint probability distribution is given by its energy function:

- energy function,

\[ E(v, h) = -b^T v - c^T h - v^T W h \]

- partition function

\[ Z = \sum_v \sum_h \exp \{-E(v, h)\} \]

- conditional distributions from the joint distribution

Full conditional over the hidden layer as the factorial distribution:

\[ p(h|v) = \prod_{j=1}^{n} p(h_j|v) = \prod_{j=1}^{n} \text{sigmoid}(c_j + v^T W_{:,j}) \]

encoding

\[ P(h_j = 1 | v) = \text{sigmoid} \left( c_j + v^T W_{:,j} \right) \]

j = 1, ..., 4
Restricted Boltzmann Machines (RBMs)

- Energy-based models,

Therefore, joint probability distribution is given by its energy function:

\[ E(v, h) = -b^T v - c^T h - v^T W h \]

- Energy function,

Partition function:

\[ Z = \sum_v \sum_h \exp \{ -E(v, h) \} \]

- Partition function

Conditional distributions from the joint distribution

Full conditional over the hidden layer as the factorial distribution:

\[ p(h|v) = \prod_{j=1}^n p(h_j|v) = \prod_{j=1}^n \text{sigmoid}(c_j + v^T W_{:.j}) \]

Same W matrix in encoding and decoding

Full conditional over the visible layer as the factorial distribution:

\[ p(v|h) = \prod_{i=1}^d p(v_i|h) = \prod_{i=1}^d \text{sigmoid}(b_i + W_{:.i}h) \]
Restricted Boltzmann Machines (RBMs): Illustration

https://jamesmccaffrey.wordpress.com/2017/06/02/restricted-boltzmann-machines-using-c/
Intern © Siemens AG 2017
Training RBMs

**Cost: maximize log-likelihood of the data, \( v \)**

Let us consider that we have a batch (or minibatch) of \( n \) examples taken from an i.i.d. dataset (independently and identically distributed examples) \( \{v^{(1)}, \ldots, v^{(t)}, \ldots, v^{(n)}\} \).

The log likelihood under the RBM with parameters \( b \) (visible unit biases), \( c \) (hidden unit biases) and \( W \) (interaction weights) is given by:

\[
\ell(W, b, c) = \sum_{t=1}^{n} \log P(v^{(t)})
\]

\[
= \sum_{t=1}^{n} \log \sum_{h} P(v_{n, \cdot}^{(t)}, h)
\]

\[
= \left( \sum_{t=1}^{n} \log \sum_{h} \exp \{-E(v^{(t)}, h)\} \right) - n \log Z
\]

\[
= \left( \sum_{t=1}^{n} \log \sum_{h} \exp \{-E(v^{(t)}, h)\} \right) - n \log \sum_{v, h} \exp \{-E(v, h)\}
\]

Trained efficiently using contrastive divergence (CD or PCD)

Impractical to compute the exact log-likelihood gradient

partition function (intractable)

https://ift6266h15.files.wordpress.com/2015/03/chapter21.pdf
Training RBMs

Cost: maximize log-likelihood of the data, \( v \)

Let us consider that we have a batch (or minibatch) of \( n \) examples taken from an i.i.d.
dataset (independently and identically distributed examples) \( \{v^{(1)}, \ldots, v^{(t)}, \ldots, v^{(n)}\} \).
The log likelihood under the RBM with parameters \( b \) (visible unit biases), \( c \) (hidden
unit biases) and \( W \) (interaction weights) is given by:

\[
\ell(W, b, c) = \sum_{t=1}^{n} \log P(v^{(t)})
= \sum_{t=1}^{n} \log \sum_h P(v^{(t)}_n, h)
= \left( \sum_{t=1}^{n} \log \sum_h \exp \left\{ -E(v^{(t)}, h) \right\} \right) - n \log Z
= \left( \sum_{t=1}^{n} \log \sum_h \exp \left\{ -E(v^{(t)}, h) \right\} \right) - n \log \sum_{v,h} \exp \left\{ -E(v, h) \right\}
\]

Trained efficiently using contrastive divergence (CD or PCD)

Impractical to compute the exact log-likelihood gradient

partition function (intractable)

https://ift6266h15.files.wordpress.com/2015/03/chapter21.pdf
Stacked RBMs: Deep Boltzmann Machine (DBM)

\[ P_\theta(v) = \frac{P^*(v)}{Z(\theta)} = \frac{1}{Z(\theta)} \sum_{h^1, h^2, h^3} \exp \left[ v^T W^1 h^1 + h^1^T W^2 h^2 + h^2^T W^3 h^3 \right] \]

\[ \theta = \{ W^1, W^2, W^3 \} \] model parameters

- Dependencies between hidden variables.
- All connections are undirected.
- Bottom-up and Top-down:

\[ P(h_j^3 = 1 | h^1, h^3) = \sigma \left( \sum_k W_{k,j}^3 h_k^3 + \sum_m W_{m,j}^2 h_m^1 \right) \]

Detailed lecture: [https://www.youtube.com/watch?v=MnGXXDjGNd0](https://www.youtube.com/watch?v=MnGXXDjGNd0)
Applications of RBMs

- Unsupervised pre-training and transfer learning
- Once trained, can use $W$ and biases as initial values for a neural net!

$W$ can be used to initialize neural networks
- Latent vector as features into neural network

Filter ($W$) from 1st layer: “pen-strokes”
Applications of DBMs

- Unsupervised pre-training and transfer learning
- Once trained, can use $W$ and biases as initial values for a neural net!

$W$ can be used to initialize neural networks
- Latent vector as features into neural network
RBM variants: How to model Word Counts?

→ RBM and DBM model **binary** input or real-valued input using Gaussian-RBMs

How to model count data?

Example: Text document i.e., word counts.

“a cat catches a mouse”  \[ s = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 0 \\ 0 \end{pmatrix} \]
RBM variants: Overview of Replicated Softmax (RSM)

- **Generative Model of Word Counts**
  - Family of *different-sized RBMs*
- **Energy** based undirected static topic model

\[ E(V, h) = - \sum_{i=1}^{D} \sum_{j=1}^{F} \sum_{k=1}^{K} W_{ij}^k h_j v_i^k - \sum_{i=1}^{D} \sum_{k=1}^{K} v_i^k b_i^k - \sum_{j=1}^{F} h_j a_j \]

Partition function (intractable)

\[ P(V) = \frac{1}{Z} \sum_{h} \exp(-E(V, h)), \quad Z = \sum_{V} \sum_{h} \exp(-E(V, h)) \]

\[ p(v_i^k = 1|h) = \frac{\exp(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k)}{\sum_{q=1}^{K} \exp(b_i^q + \sum_{j=1}^{F} h_j W_{ij}^q)} \]

\[ p(h_j = 1|V) = \sigma \left( a_j + \sum_{i=1}^{D} \sum_{k=1}^{K} v_i^k W_{ij}^k \right) \]

**Gupta et al. 2018, Deep Temporal-Recurrent-Replicated-Softmax for Topical Trends over Time.**
RBM variants: Overview of Replicated Softmax (RSM)

- **Generative Model of Word Counts**
- **Family of different-sized RBMs**
- **Energy** based undirected static topic model

\[
E(V, h) = - \sum_{i=1}^{D} \sum_{j=1}^{F} \sum_{k=1}^{K} W_{ij}^k h_j v_i^k - \sum_{i=1}^{D} \sum_{k=1}^{K} v_i^k b_i^k - \sum_{j=1}^{F} h_j a_j
\]

Partition function (intractable)

\[
P(V) = \frac{1}{Z} \sum_{h} \exp(-E(V, h)), \quad Z = \sum_{V} \sum_{h} \exp(-E(V, h))
\]

In RBM and RSM, \(p(v)\) is intractable !!!

\[
p(h_j = 1|V) = \sigma\left(a_j + \sum_{i=1}^{D} \sum_{k=1}^{K} v_i^k W_{ij}^k\right),
\]

\[
E(V, h) = - \sum_{j=1}^{F} \sum_{k=1}^{K} W_{ij}^k h_j \hat{v}^k - \sum_{k=1}^{K} \hat{v}^k b_i^k - D \sum_{j=1}^{F} h_j a_j
\]

Neural Autoregressive Distribution Estimator (NADE) models

Idea: Tractable log-likelihood, \( p(v) \)
Neural Autoregressive Distribution Estimator (NADE)

- **NADE**: Neural Autoregressive Distributional Estimator
- Inspired from **RBM**,
- Generative model over **binary** observations $v$
- sampling each dimension one after another

NADE is for binary data

Uria, Benigno, et al. "Neural Autoregressive Distribution Estimation"
Neural Autoregressive Distribution Estimator (NADE)

- **NADE**: Neural Autoregressive Distributional Estimator
- Inspired from RBM Family therefore, energy based
- sampling each dimension one after another

\[
p(v) = \prod_{i=1}^{D} p(v_i|v_{<i}) \text{ and computes all } p(v_i|v_{<i}) \text{ using the feed-forward architecture}
\]

\[
h_i(v_{<i}) = \text{sigm}(c + W_{:,<i}v_{<i})
\]

NADE is for binary data,

\[
p(v_i = 1|v_{<i}) = \text{sigm}(b_i + V_{:,i}h_i(v_{<i}))
\]

\[
h_{i+1}(v_{<i+1}) = \text{sigm}(c + \sum_{k<i+1} W_{:,v_k}) = \text{sigm}(W_{:,v_i} + c + \sum_{k<i} W_{:,v_k})
\]

for \(i \in \{1, \ldots, D\}\), where \(\text{sigm}(x) = 1/(1 + \exp(-x))\), \(W \in \mathbb{R}^{H \times D}\) and \(V \in \mathbb{R}^{D \times H}\) are connection parameter matrices, \(b \in \mathbb{R}^{D}\) and \(c \in \mathbb{R}^{H}\) are bias parameter vectors, \(v_{<i}\) is the subvector \([v_1, \ldots, v_{i-1}]^T\) and \(W_{:,<i}\) is a matrix made of the \(i - 1\) first columns of \(W\).

corresponds to a neural network with several parallel \(h_i(v_{<i})\) hidden layers

Advantages: Tractable \(p(v) \rightarrow\) easier to train
Neural Autoregressive Distribution Estimator (NADE)

- **NADE**: Neural Autoregressive Distributional Estimator
- Inspired from RBM Family therefore, energy based
- Sampling each dimension one after another

\[
p(v) = \prod_{i=1}^{D} p(v_i|v_{<i}) \text{ and computes all } p(v_i|v_{<i}) \text{ using the feed-forward architecture}
\]

\[
h_i(v_{<i}) = \text{sigmoid}(c + W_{:,<i}v_{<i})
\]

NADE is for binary data,

\[
p(v_i = 1|v_{<i}) = \text{sigmoid}(b_i + V_{i,:}h_i(v_{<i}))
\]

\[
h_{i+1}(v_{<i+1}) = \text{sigmoid}(c + \sum_{k<i+1} W_{:,v_k}) = \text{sigmoid}(W_{:,v_i} + c + \sum_{k<i} W_{:,v_k})
\]

For \(i \in \{1, \ldots, D\}\), where \(\text{sigmoid}(x) = 1/(1 + \exp(-x))\), \(W \in \mathbb{R}^{H \times D}\) and \(V \in \mathbb{R}^{D \times H}\) are connection parameter matrices, \(b \in \mathbb{R}^{D}\) and \(c \in \mathbb{R}^{H}\) are bias parameter vectors, \(v_{<i}\) is the subvector \([v_1, \ldots, v_{i-1}]^\top\) and \(W_{:,<i}\) is a matrix made of the \(i-1\) first columns of \(W\).

Corresponds to a neural network with several parallel \(h_i(v_{<i})\) hidden layers.

Advantages: Tractable \(p(v) \rightarrow \) easier to train
Neural Autoregressive Distribution Estimator (NADE)

- **NADE**: Neural Autoregressive Distributional Estimator
- Inspired from RBM Family therefore, energy based
- sampling each dimension one after another

\[ p(\mathbf{v}) = \prod_{i=1}^{D} p(v_i|\mathbf{v}_{<i}) \] and computes all \( p(v_i|\mathbf{v}_{<i}) \) using the feed-forward architecture

\[ h_i(\mathbf{v}_{<i}) = \text{sign}(c + W_{:,<i}\mathbf{v}_{<i}) \]

NADE is for binary data,

\[ p(v_i = 1|\mathbf{v}_{<i}) = \text{sign}(b_i + V_{:,i}.h_i(\mathbf{v}_{<i})) \]

\[ h_{i+1}(\mathbf{v}_{<i+1}) = \text{sign}(c + \sum_{k<i+1} W_{:,v_k}) = \text{sign}(W_{:,v_i} + c + \sum_{k<i} W_{:,v_k}) \]

for \( i \in \{1, \ldots, D\} \), where \( \text{sign}(x) = 1/(1 + \exp(-x)) \), \( W \in \mathbb{R}^{D \times D} \) and \( V \in \mathbb{R}^{D \times H} \) are connection parameter matrices, \( b \in \mathbb{R}^D \) and \( c \in \mathbb{R}^H \) are bias parameter vectors, \( \mathbf{v}_{<i} \) is the subvector \( [v_1, \ldots, v_{i-1}]^\top \) and \( W_{:,<i} \) is a matrix made of the \( i - 1 \) first columns of \( W \).

...corresponds to a neural network with several parallel \( h_i(\mathbf{v}_{<i}) \) hidden layers

Advantages: Tractable \( p(\mathbf{v}) \) → easier to train
Neural Autoregressive Distribution Estimator (NADE)

Training NADE

→ Ground truth values of the pixels are used for conditioning when predicting subsequent values.

→ cost: maximize log-likelihood ($\log L$)

→ optimize to maximize the logL by **stochastic gradient descent** (SGD)

\[
\mathcal{L}_{DocNADE}(\mathbf{v}) = \sum_{i=1}^{D} \log p(v_i|\mathbf{v}_{<i})
\]

where, $D$ is the number of words in document, $\mathbf{v}$ and autoregressive conditional is given by:

\[
p(v_i = 1|\mathbf{v}_{<i}) = \text{sigmoid}(b_i + \mathbf{V}_{i,:} \cdot h_i(\mathbf{v}_{<i}))
\]

where,

\[
h_i(\mathbf{v}_{<i}) = \text{sigmoid}(c + \mathbf{W}_{:,<i} \cdot \mathbf{v}_{<i})
\]
Neural Autoregressive Distribution Estimator (NADE)

Training NADE

→ Ground truth values of the pixels are used for conditioning

NADE: Alternative to RBMs with tractable \( p(\mathbf{v}) \)

→ optimize to maximize the \( \log L \) by **stochastic gradient descent** (SGD)

where, \( D \) is the number of words in document, \( \mathbf{v} \) and

an autoregressive conditional is given by:

\[
p(v_i = 1|\mathbf{v}_{<i}) = \text{sigm}(b_i + \mathbf{V}_{i,:}h_i(\mathbf{v}_{<i}))
\]

where,

\[
h_i(\mathbf{v}_{<i}) = \text{sigm}(c + \mathbf{W}_{:,<i}\mathbf{v}_{<i})
\]
Neural Autoregressive Distribution Estimator (NADE)

(Left): samples from NADE trained on a binary version of MNIST  (Middle): probabilities from which pixel was samples  (Right): Visualization of some of the rows

Neural Autoregressive Distribution Estimator (NADE)

(Left): samples from NADE trained on a binary version of MNIST  
(Middle): probabilities from which pixel was samples  
(Right): Visualization of some of the rows

NADE models binary input or real-valued input using realNADE

How to model count data in NADE architecture?
Example: Text document i.e., word counts.

"a cat catches a mouse"  \[ s = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 0 \\ 0 \end{pmatrix} \]
BREAK
NADE model **binary** input or real-valued input using realNADE

How to model count data in NADE architecture?
Example: Text document i.e., word counts.

“a cat catches a mouse”

Modeling documents in NADE → DocNADE
Neural Autoregressive Topic Model (DocNADE)

Probabilistic graphical model that *learns topics over sequences of words*

- learn topic-word distribution based on word co-occurrences
- learn distributed word representations
- compute joint distribution via autoregressive conditionals
- compute joint distribution or log-likelihood for a document, \( \mathbf{v} \) in *language modeling fashion*
- interpreted as a neural network with several parallel hidden layers
- predict the word \( v_i \), given the sequence of preceding words \( v_{<i} \)

**Modeling documents in NADE**
Neural Autoregressive Topic Model (DocNADE)

Probabilistic graphical model that *learns topics over sequences of words*

- learn **topic-word distribution** based on word co-occurrences
- learn **distributed word representations**
- compute **joint distribution** via autoregressive conditionals
- compute joint distribution or log-likelihood for a document, \( \mathbf{v} \) in *language modeling fashion*
- interpreted as a neural network with several parallel hidden layers
- **predict the word** \( v_i \), **given the sequence of preceding words** \( v_{<i} \)

**Limitations:**
- does not take into account the following words \( v_{>|i} \) in the sequence
- poor in modeling short-text documents
  (i.e., does not use pre-trained word embeddings)
Neural Autoregressive Topic Model (DocNADE)

Probabilistic graphical model that *learns topics over sequences of words*

- learn topic-word distribution based on word co-occurrences
- learn distributed word representations
- compute joint distribution via autoregressive conditionals
- compute joint distribution or log-likelihood for a document, \( \mathbf{v} \)
  in *language modeling fashion*
- interpreted as a neural network with several parallel hidden layers
- predict the word \( v_i \), given the sequence of preceding words \( v_{<i} \)

Limitations:

- does not take into account the following words \( v_{>i} \) in the sequence
- poor in modeling short-text documents due to limited context i.e., co-occurrences
Neural Autoregressive Topic Model (DocNADE)

DocNADE Formulation

- inspired by RBM, RSM and NADE models
- models the joint distribution of all words $v_i$
  $v_i \in \{1, \ldots, K\}$
  i.e., the index of the $ith$ word in the dictionary of vocabulary size $K$
- a document $v$ of size $D$ is represented as, $v = [v_1, \ldots, v_D]$
- joint distribution $p(v)$ computed via each autoregressive conditionals,

$$p(v) = \prod_{i=1}^{D} p(v_i | v_{<i})$$
Neural Autoregressive Topic Model (DocNADE)

DocNADE Formulation

- inspired by RBM, RSM and NADE models
- models the joint distribution of all words \( v_i \)
  \[ v_i \in \{1, \ldots, K\} \]
  i.e., the index of the \( i \)th word in the dictionary of vocabulary size \( K \)
- a document \( v \) of size \( D \) is represented as, \( v = [v_1, \ldots, v_D] \)
- joint distribution \( p(v) \) computed via each autoregressive conditionals,
  \[ p(v) = \prod_{i=1}^{D} p(v_i | v_{<i}) = \prod_{i=1}^{D} \frac{\exp(b_w + U_{w,:} h_i(v_{<i}))}{\sum_{w'} \exp(b_{w'} + U_{w',:} h_i(v_{<i}))} \]
  via a feed-forward neural network, where \( v_{<i} \in \{v_1, \ldots, v_{i-1}\} \)
Neural Autoregressive Topic Model (DocNADE)

DocNADE Formulation

- inspired by RBM, RSM and NADE models
- models the joint distribution of all words $v_i$
  \[ v_i \in \{1, ..., K\} \]
  i.e., the index of the $i$th word in the dictionary of vocabulary size $K$
- a document $v$ of size $D$ is represented as, \[ v = [v_1, ..., v_D] \]
- joint distribution $p(v)$ computed via each autoregressive conditionals,
  \[
p(v) = \prod_{i=1}^{D} p(v_i | v_{<i}) = \prod_{i=1}^{D} \frac{\exp(b + \sum_{w} w_i \cdot h_i(v_{<i}))}{\sum_{w'} \exp(b + \sum_{w'} w_{i} \cdot h_i(v_{<i}))}
\]
  via a feed-forward neural network, where $v_{<i} \in \{v_1, ..., v_{i-1}\}$

**Topic matrix**

\[
h_i(v_{<i}) = g(c + \sum_{k<i} W_{i,k} v_k)
\]
where, \( W \in \mathbb{R}^{H \times K} \) and \( U \in \mathbb{R}^{K \times H} \)
Neural Autoregressive Topic Model (DocNADE)

S1: Deal with stock index fall
S2: Brace for market share drop
Neural Autoregressive Topic Model (DocNADE)

S1: Deal with stock index fall
S2: Brace for market share drop
Neural Autoregressive Topic Model (DocNADE)

S1: Deal with stock index fall
S2: Brace for market share drop

Embedding aggregation

\[ \mathbf{h}_i(v_{<i}) = g(c + \sum_{k<i} \mathbf{W};v_k) \]
Neural Autoregressive Topic Model (DocNADE)

S1: Deal with stock index fall
S2: Brace for market share drop

Embedding aggregation
Neural Autoregressive Topic Model (DocNADE)

S1: Deal with stock index fall
S2: Brace for market share drop

Embedding aggregation
Neural Autoregressive Topic Model (DocNADE)

DocNADE Formulation

Properties of weight matrix, $W$

- each column-vector $W_{:, v_i}$
  - a vector for the word $v_i$
- each row-vector $W_{j,:}$
  - a distribution over vocabulary of size $K$, representing the $j$th topic
- exploit column-vector property and introduce additional matrix $E$, to incorporate pre-trained word embeddings or distributional word representations
Neural Autoregressive Topic Model (DocNADE)

DocNADE Formulation

- inspired by RBM, RSM and NADE models
- models the joint distribution of all words $v_i$
  
  $$v_i \in \{1, \ldots, K\}$$

  i.e., the index of the $i$th word in the dictionary of vocabulary size $K$
- a document $v$ of size $D$ is represented as, $v = [v_1, \ldots, v_D]$
- joint distribution $p(v)$ computed via each autoregressive conditionals,

  $$p(v) = \prod_{i=1}^{D} p(v_i | v_{<i}) = \prod_{i=1}^{D} \frac{\exp(b_w + U_{w,:} \cdot \overrightarrow{h}_i(v_{<i}))}{\sum_{w'} \exp(b_{w'} + U_{w',:} \cdot \overrightarrow{h}_i(v_{<i}))}$$

  via a feed-forward neural network, where $v_{<i} \in \{v_1, \ldots, v_{i-1}\}$

**Topic matrix**

$$\overrightarrow{h}_i(v_{<i}) = g(c + \sum_{k<i} W_{k,:} v_k)$$

where, $W \in \mathbb{R}^{H \times K}$ and $U \in \mathbb{R}^{K \times H}$

DOES NOT take into account the following words $v_{>i}$

Exploiting distributed word representations (word embeddings) + Full Context information

In biological brains, we study noisy neurons at cellular level → “biological neural network”

Like biological brains, study of noisy neurons in artificial neural networks → “artificial neural network”

Context information around words helps in determining their actual meaning !!!
Informed Document Autoregressive Topic Model with Word Embeddings

- “Lack of Context” in short-text documents, e.g., headlines, tweets, etc.
- “Lack of Context” in a corpus of few documents

Difficult to learn good representation due to:
  - small number of word co-occurrences
  - significant word non-overlap

Incoherent Topics, e.g.,
  - Topic 1: price, wall, china, fall, shares
  - Topic 2: shares, price, profits, rises, earnings

TO RESCUE: Use External/additional information, e.g., word embeddings (encodes semantic and syntactic relatedness in words)

Motivation 2: Distributional Semantics for Lack of Context

No word overlap (e.g., 1-hot-encoding)

Cosine similarity in Word Embedding space

Trading

Same topic class
DocNADE variants: Contextualized DocNADE (iDocNADE)

- incorporating full contextual information around words in a document (preceding and following words)
- boost the likelihood of each word and subsequently the document
- improved representation learning

DocNADE variants: Contextualized DocNADE (iDocNADE)

$$\mathcal{L}^{DocNADE}(v) = \sum_{i=1}^{D} \log p(v_i|v_{<i})$$

$$\overrightarrow{h_i}(v_{<i}) = g(c + \sum_{k<i} W_{i,k} v_k)$$

$$\log p(v) = \frac{1}{2} \sum_{i=1}^{D} \log p(v_i|v_{<i}) + \log p(v_i|v_{>i})$$

$$\overrightarrow{h_i}(v_{<i}) = g(c + \sum_{k<i} W_{i,k} v_k)$$

$$\overrightarrow{h_i}(v_{>i}) = g(c + \sum_{k>i} W_{i,k} v_k)$$
DocNADE variants: Contextualized DocNADE (iDocNADE)

Incomplete Context around words in DocNADE

Need for Complete Context around words in iDocNADE
DocNADE variants: DocNADE + Embedding Priors ‘e’ (DocNADEe)

DocNADE variants: DocNADE + Embedding Priors ‘e’ (DocNADEe)

- introduce weighted word embedding aggregation at each autoregressive step $k$
- $E$ as fixed prior
- topics with word embeddings
- generate a complementary textual representation (*duality*)

$$L^{DocNADE}(v) = \sum_{i=1}^{D} \log p(v_i | v_{<i})$$

$$\vec{h}^e_i(v_{<i}) = g(c + \sum_{k<i} W : , v_k)$$

DocNADE variants: DocNADE + Embedding Priors ‘e’ (DocNADEe)

- introduce weighted word embedding aggregation at each autoregressive step $k$
- $E$ as fixed prior
- topics with word embeddings
- generate a complementary textual representation (duality)

\[
\mathcal{L}_{\text{DocNADE}}(v) = \sum_{i=1}^{D} \log p(v_i|v_{<i})
\]

\[
\overline{h}_i^{e}(v_{<i}) = g(c + \sum_{k<i} W_{:,v_k} + \lambda \sum_{k<i} E_{:,v_k})
\]

Deep DocNADEs Variants with/without Embedding Priors

- Deep, multiple hidden layer architectures
- Adding new hidden layers as in a regular deep feed-forward neural network

\[
\overrightarrow{h}_i^{(d)}(v_{<i}) = g(\overrightarrow{c}^{(d)} + \mathbf{W}^{(d)} \cdot \overrightarrow{h}_i^{(d-1)}(v_{<i}))
\]

introduce embedding prior $\mathbf{E}$ in the first hidden layer, i.e.,

\[
\overrightarrow{h}_i^{e,(1)} = g(\overrightarrow{c}^{(1)} + \sum \mathbf{W}^{(1)}_{:,v_k} + \lambda \sum \mathbf{E}^{:,v_k})
\]

With the trained iDocNADEe (or DocNADE variants), the representation \( \hat{h}^e \in \mathbb{R}^H \) for a new document \( \mathbf{v}^* \) of size \( D^* \) is extracted by summing the hidden representations from the forward and backward networks to account for the context information around each word in the words’ sequence, as

\[
\hat{h}^e (\mathbf{v}^*) = g(\vec{c} + \sum_{k \leq D^*} \mathbf{W} ; , \mathbf{v}_k^* + \lambda \sum_{k \leq D^*} \mathbf{E} ; , \mathbf{v}_k^*)
\]

\[
\hat{h}^e (\mathbf{v}^*) = g(\vec{c} + \sum_{k \geq 1} \mathbf{W} ; , \mathbf{v}_k^* + \lambda \sum_{k \geq 1} \mathbf{E} ; , \mathbf{v}_k^*)
\]

Therefore, \( \hat{h}^e = \hat{h}^e (\mathbf{v}^*) + \hat{h}^e (\mathbf{v}^*) \)


Tasks:
- Information retrieval
- Document representation
- Word representation
- Text classification
- Text clustering, etc.
Tasks:
→ Information retrieval
→ Document representation
→ Word representation
→ Text classification
→ Text clustering, etc.

Table 6: 20NS dataset: The five nearest neighbors by iDocNADE. $s_i$: Cosine similarity between the word vectors from iDocNADE, for instance vectors of *jesus* and *god*. $s_g$: Cosine similarity in embedding vectors from glove.


Visualizing or interpreting filters i.e., W matrix. (column vectors $\rightarrow$ word embeddings)
### Document Representation in iDocNADE variants

<table>
<thead>
<tr>
<th>DocNADE</th>
<th>iDocNADE</th>
<th>DocNADEe</th>
</tr>
</thead>
<tbody>
<tr>
<td>beliefs, muslims, forward, alt, islam, towards, atheism, christianity, hands, opinions</td>
<td>scripture, atheists, sin, religions, christianity, lord, bible, msg, heaven, jesus</td>
<td>atheists, christianity, belief, eternal, atheism, catholic, bible, arguments, islam, religions</td>
</tr>
<tr>
<td>0.44</td>
<td>0.46</td>
<td>0.52</td>
</tr>
</tbody>
</table>

**Topics (top 10 words) of 20NS with coherence**


Tasks:
- Information retrieval
- Document representation
- Word representation
- Text classification
- Text clustering, etc.

**Visualizing or interpreting filters i.e., W matrix. (row vectors → topic information)**
Limitations:

→ Bag-of-word models
→ missing word ordering
→ missing local dynamics of the sequence

Extension(s):

→ Joint neural autoregressive topic (e.g., DocNADE) and neural language models (e.g., RNN or LSTM)
→ Introduce language concepts (e.g., word ordering, latent syntactic and semantic information) into DocNADE

Further reading: https://arxiv.org/abs/1810.03947
Compositional Distributional Semantics

Compositional models in Neural Networks: RNN-LM or Recursive Neural Network or seq2seq (Lecture-05)

**Generative Recurrent Neural Network (RNN)**

\[ L_t = -\log P(x_t|x_{t-1}, x_{t-2}, \ldots x_1) \]
Compositional Distributional Semantics

Compositional models in Neural Networks: RNN-LM or Recursive Neural Network or seq2seq (Lecture-05)

RNN-LM captures local dynamics of the sequence, i.e., word ordering, syntactic and semantic information from word co-occurrences in collocation/nearby patterns

→ RNN-LMs (or LSTM-LM) lack in capturing global semantics, i.e. long-term dependencies

→ DocNADEs capture global semantics in form of topics

Generative Recurrent Neural Network (RNN)

\[ L_t = -\log P(x_t|x_{t-1}, x_{t-2}, \ldots x_1) \]
Distibutional Representations: Local and Global Semantics

Combine or Joint training of DocNADE and LSTM-LM: \text{textTOvec}

Metric Learning for Similarity (overview)

Learn Text-Pair Representations in a Highly Structured Space

Semantic relatedness score e.g., [1-5]

Similarity metric

\[
\exp \left( -\| h_3^{(a)} - h_4^{(b)} \|_1 \right)
\]

Word Embedding layer, i.e., distributional Word vectors

Aditya Thyagarajan and Jonas Mueller. 2016. *Siamese Recurrent Architectures for Learning Sentence Similarity*
Key Take Aways

→ unsupervised learning for distributed representations in neural networks

→ pre-train and transfer learning to initialize neural networks for better convergence

→ distributed word representations encode syntactic, semantic information into vectors

→ metric learning to compute similarity over representations learned
References, Resources and Further Reading

- [https://www.cc.gatech.edu/classes/AY2019/cs7643_fall/slides/L1_intro.pdf](https://www.cc.gatech.edu/classes/AY2019/cs7643_fall/slides/L1_intro.pdf)
- [https://www.cc.gatech.edu/classes/AY2019/cs7643_fall/](https://www.cc.gatech.edu/classes/AY2019/cs7643_fall/)
- Generative models: [https://ift6135h18.wordpress.com/author/aaroncourville/page/1/](https://ift6135h18.wordpress.com/author/aaroncourville/page/1/)
- Boltzmann Machines: [https://ift6266h15.wordpress.com/category/lectures/page/1/](https://ift6266h15.wordpress.com/category/lectures/page/1/)
- [https://ift6266h16.wordpress.com/2016/03/14/lecture-19-march-17th-2016-representation-learning/](https://ift6266h16.wordpress.com/2016/03/14/lecture-19-march-17th-2016-representation-learning/)
- Generative models: [https://ift6266h15.files.wordpress.com/2015/03/chapter21.pdf](https://ift6266h15.files.wordpress.com/2015/03/chapter21.pdf)
- [https://www.cl.cam.ac.uk/~pv273/slides/UCLSlides.pdf](https://www.cl.cam.ac.uk/~pv273/slides/UCLSlides.pdf)
- [https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/](https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/)
- Plotting Samples and Filters: [http://deeplearning.net/tutorial/utilities.html#how-to-plot](http://deeplearning.net/tutorial/utilities.html#how-to-plot)
Thanks !!! Question ???

Write me, if interested in ....

firstname.lastname@siemens.com

@Linkedin: https://www.linkedin.com/in/pankaj-gupta-6b95bb17/

About my research contributions:

https://scholar.google.com/citations?user=_YjIJF0AAAAJ&hl=en