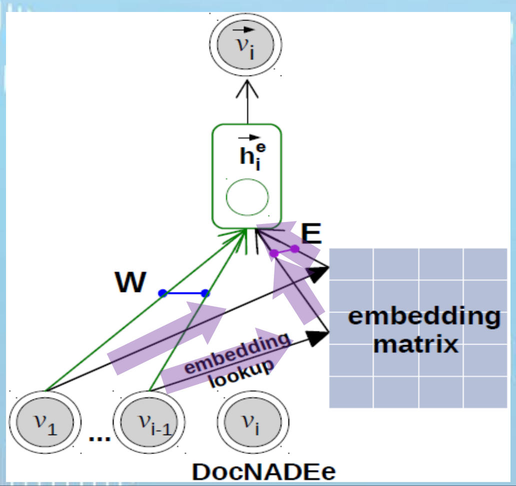


Lecture-07: Representation and Distributional Learning (Deep Learning & AI @LMU, Germany)

***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*)

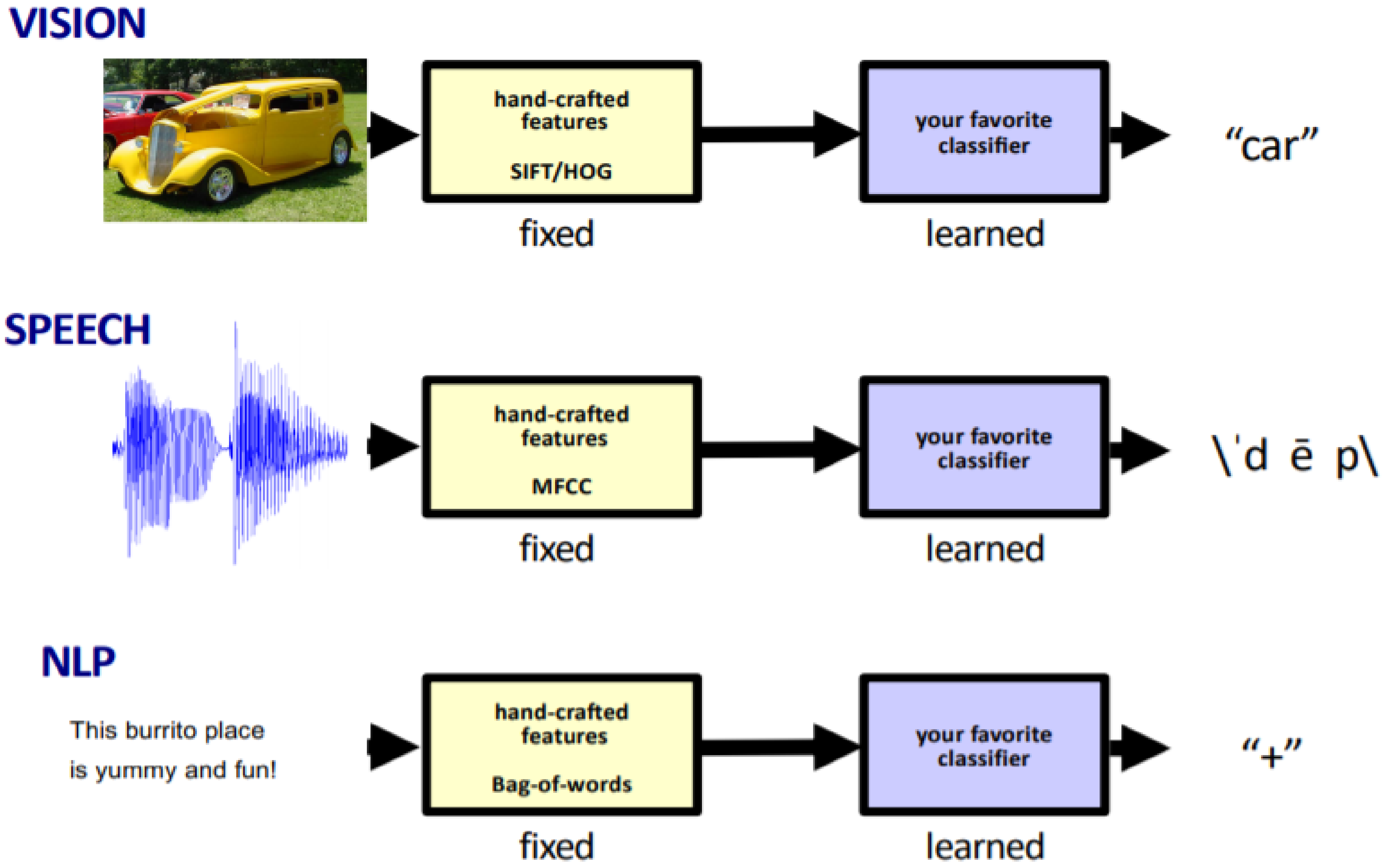
What is Representation Learning ?

Why do we need Representation Learning ?

How to form good Representations?

Motivation: Representation Learning

Traditional Machine Learning



Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Motivation: Representation Learning

Hierarchical Compositionality

VISION

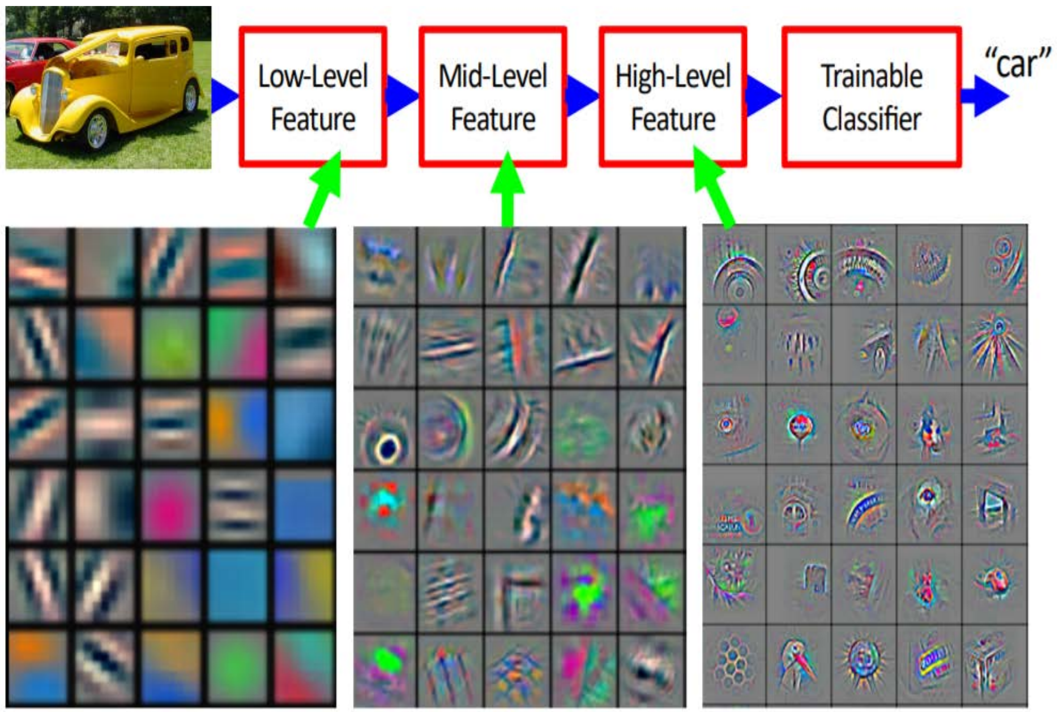
pixels ▶ edge ▶ texton ▶ motif ▶ part ▶ object

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-

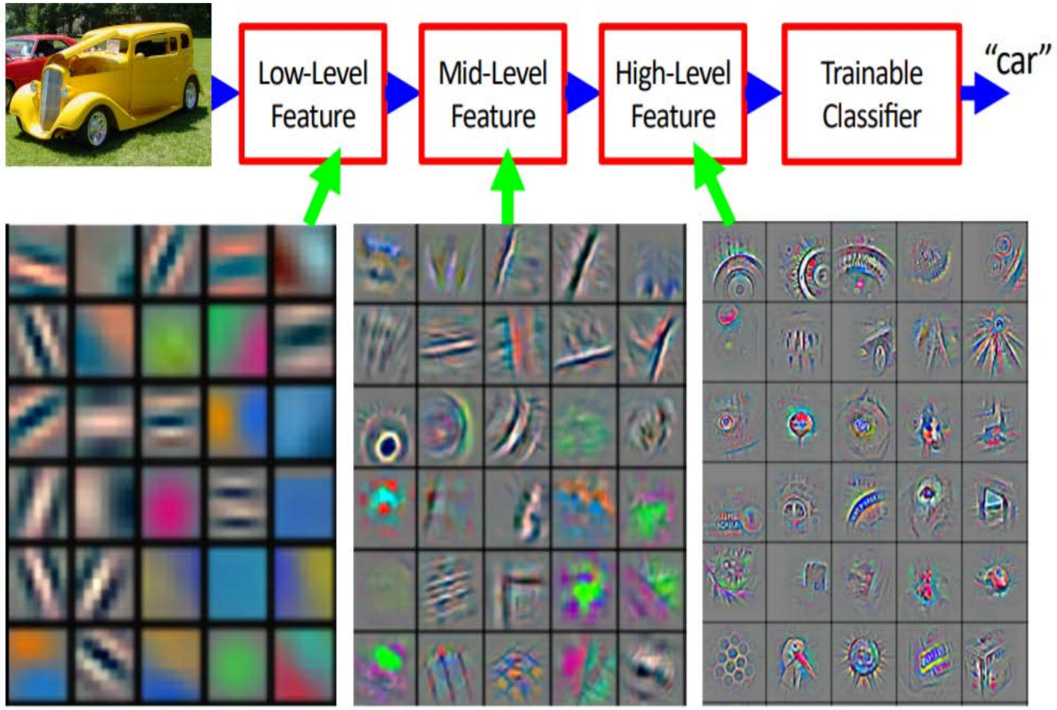
(1) Distributed representations / embeddings

→ *feature learning*

(2) Deep architectures

→ *multiple* levels of feature learning

Compositionality to describe the world around us efficiently



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

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Motivation: Representation Learning

Feature Learning

Representation Learning

cuisine_italian

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*,

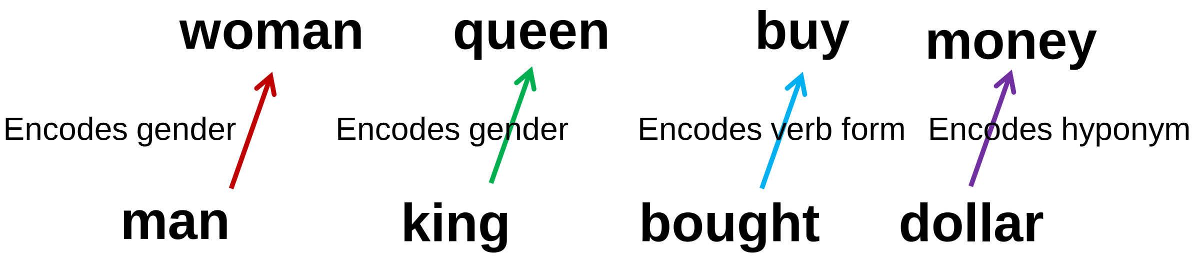
→ classification from raw data

Feature

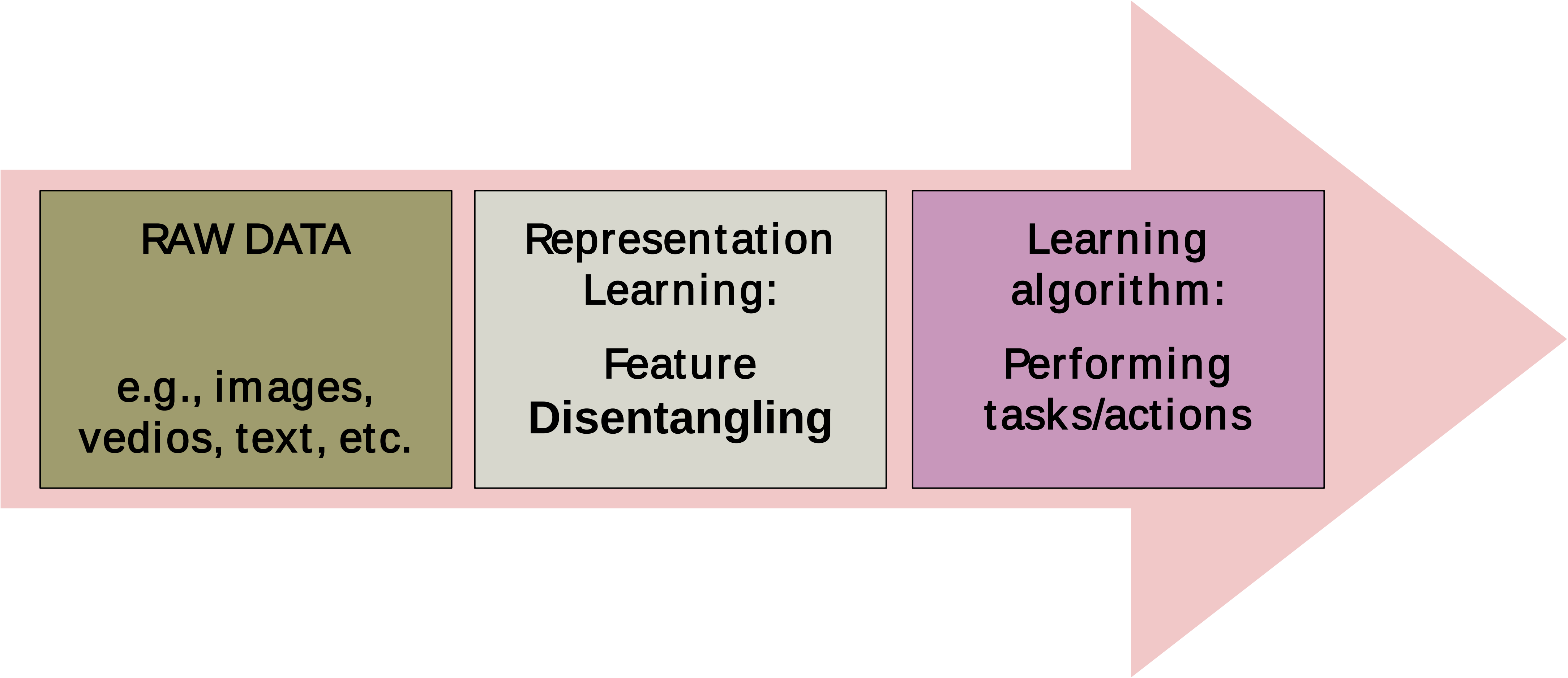
→ measurable property or characteristic of a phenomenon being observed

E.g.,

- *shape*, *size*, *color*, etc. of objects
- *content*, *cuisine*, *color*, etc. of food items
- *syntactic* and *semantic* relationships in words



Motivation: Representation Learning

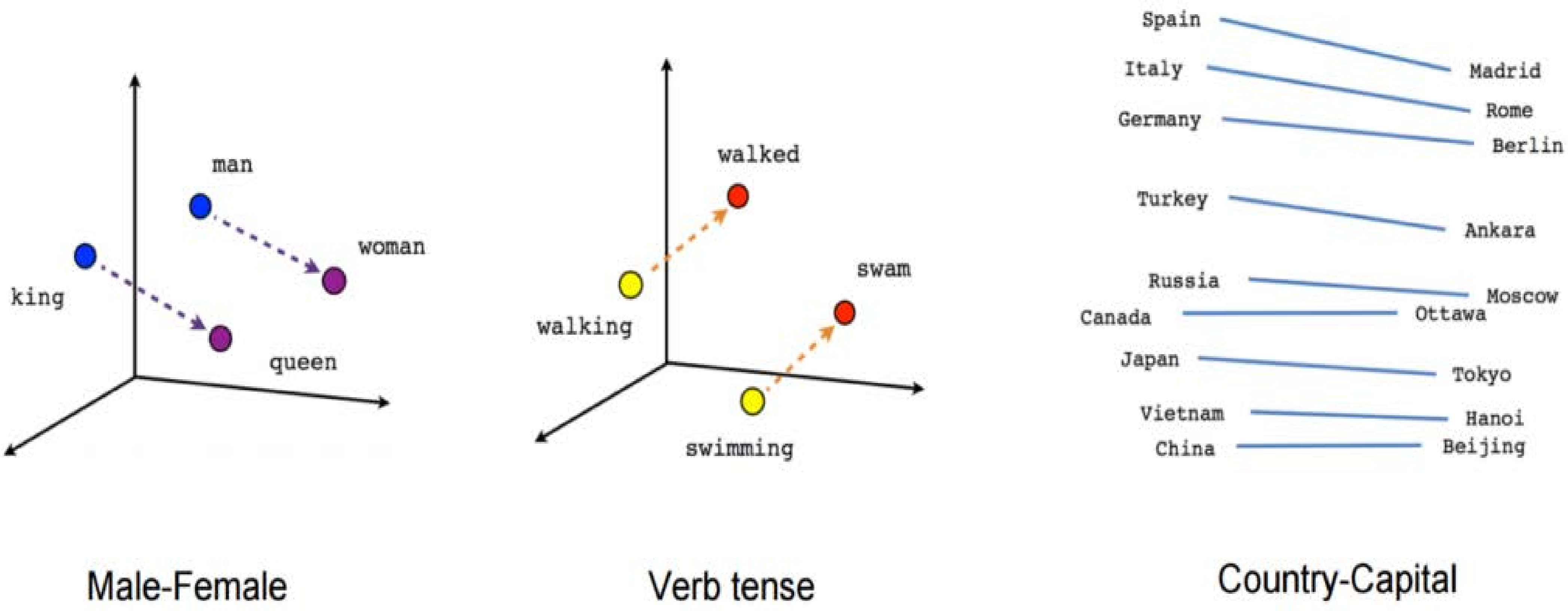


Why: Representation Learning

Collection of similar objects, based on shared [latent] features

Why: Representation Learning

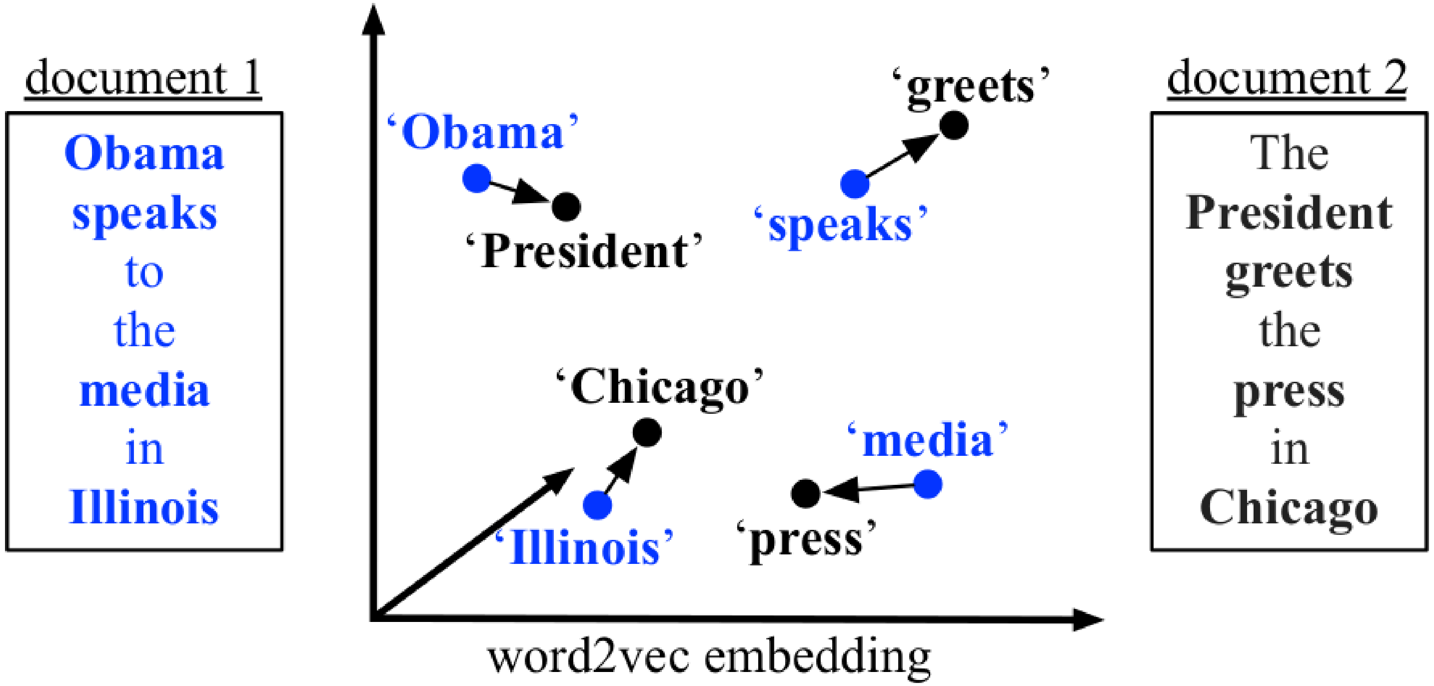
Word Analogy in distributional semantic vector space



WWW.TENSORFLOW.ORG/IMAGES/LINEAR-RELATIONSHIPS.PNG

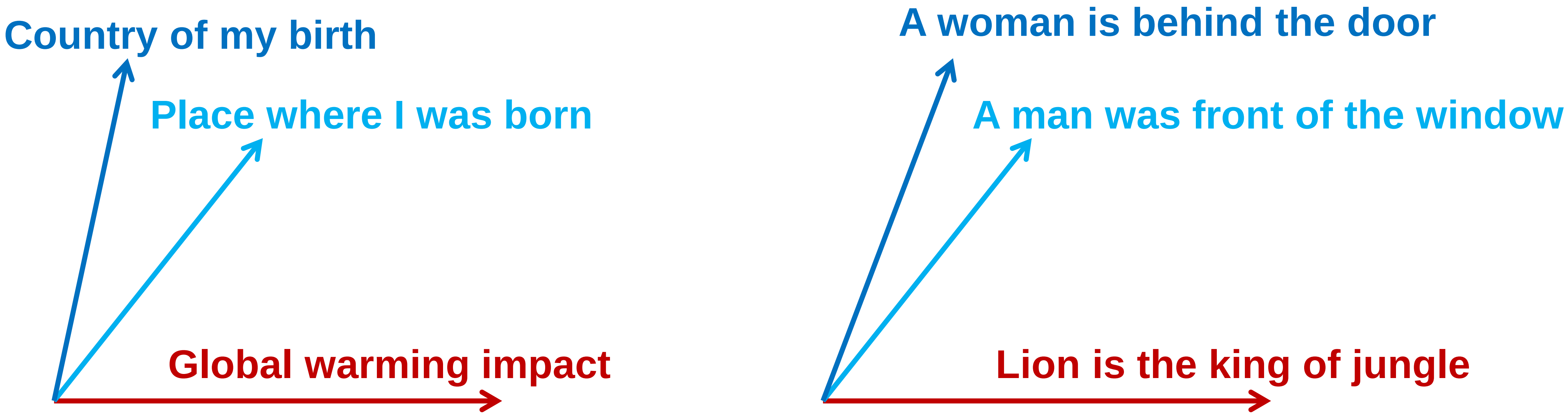
Why: Representation Learning

Vector composition to determine semantic similarity in texts !!



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

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



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

How about CONTEXT in feature representation learning?

Food

Motivation: Representation Learning

How about CONTEXT in feature representation learning?

meaning ?

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?

food

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?

desert

Is it a good idea to *eat* HALWA after a *meal* ?

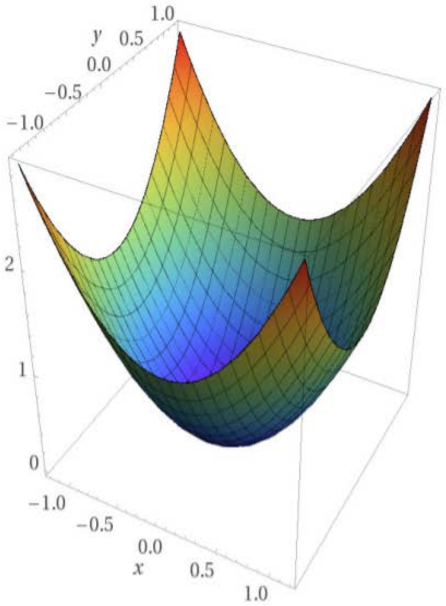
Motivation: Representation Learning

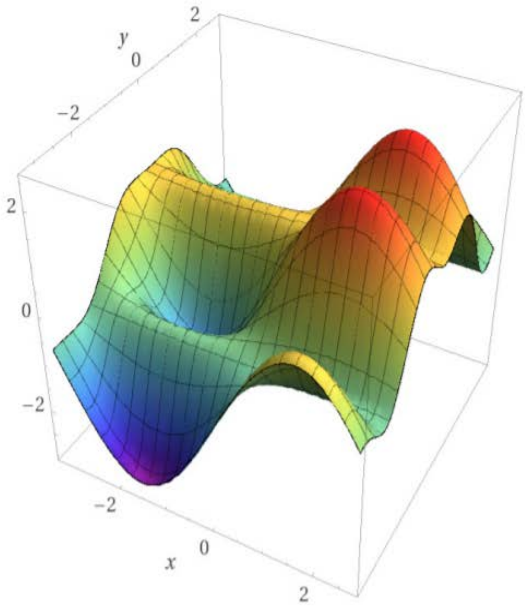
How about CONTEXT in feature representation learning?

desert

I put a cup of ***sugar*** too much in ***cooking*** **HALWA ?**

Unsupervised pre-training and Transfer learning





Loss function surfaces

Training deep networks can be challenging

?

→*Non-convex loss function*

→ *in-appropriate initializers*

→ Initialisation is cricitial in Neural network training

→ Using appropriate initializers for better convergence

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)

http://www.iro.umontreal.ca/~bengioy/ift6266/H16/representation-learning.pdf

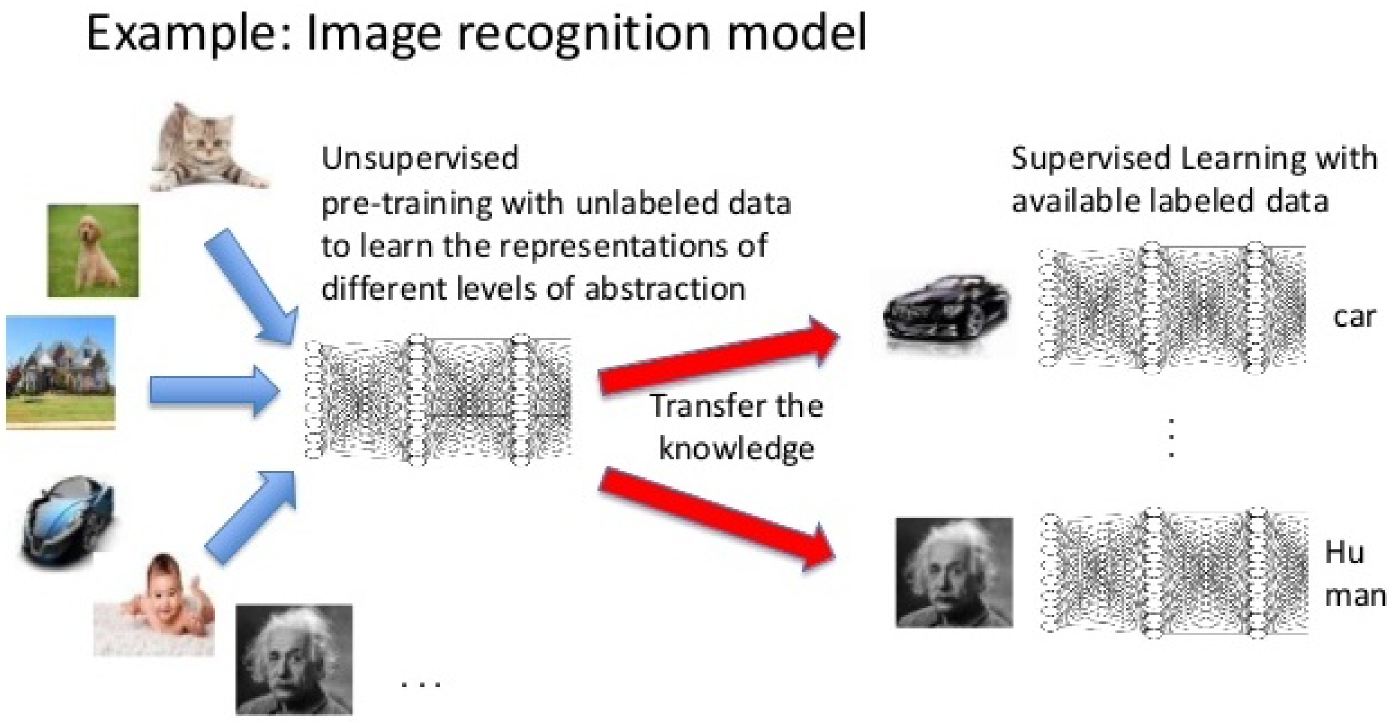
Why: Representation Learning?

Unsupervised pre-training and Transfer learning

transfer knowledge from previous

→ Representations

→ Explanatory factors



http://www.iro.umontreal.ca/~bengioy/ift6266/H16/representation-learning.pdf

Motivation: Good Representation

- Good features for *successful* machine learning,
  e.g., *man* ←→ *human*, *cat* ←→ *dog*,
  buy ←→ *bought*, *buy* ←→ *acquired* *Islam* ←→ *Christianity*, etc

- Knowing features belief about objects in prior,
  e.g., ***features of car*** → *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 → good representation
  i.e., *good representation* captures factors of variation that best explains the data

- *Learning representations from representations* → **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** = {*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.

$$\mathrm{mouse} \rightarrow \mathbf{e}_1 = (1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0)^T$$
$$\mathrm{cat} \rightarrow \mathbf{e}_2 = (0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0)^T$$
$$\mathrm{a} \rightarrow \mathbf{e}_3 = (0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0)^T$$
$$\mathrm{catches} \rightarrow \mathbf{e}_4 = (0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0)^T$$
$$(\rightarrow \mathbf{e}_5 = (0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0)^T$$
$$) \rightarrow \mathbf{e}_6 = (0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1)^T$$

Motivation: Local Representation of Symbols

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

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

$$\mathbf{s} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Local Representations

$$\text{mouse} \rightarrow \mathbf{e}_1 = (1\ 0\ 0\ 0\ 0\ 0)^T$$
$$\text{cat} \rightarrow \mathbf{e}_2 = (0\ 1\ 0\ 0\ 0\ 0)^T$$
$$\text{a} \rightarrow \mathbf{e}_3 = (0\ 0\ 1\ 0\ 0\ 0)^T$$
$$\text{catches} \rightarrow \mathbf{e}_4 = (0\ 0\ 0\ 1\ 0\ 0)^T$$
$$(\rightarrow \mathbf{e}_5 = (0\ 0\ 0\ 0\ 1\ 0)^T$$
$$) \rightarrow \mathbf{e}_6 = (0\ 0\ 0\ 0\ 0\ 1)^T$$

a sequence of vectors

→ a sequence of vectors representing the symbols in the sequence

→ used in recurrent neural networks

a bag-of-symbols

→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)

$$\mathbf{s} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Motivation: Local Representation of Symbols

Limitations of Local Representation

Local Representations

a sequence of vectors

$$\mathbf{s} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

→ Each vector has a one-to-one mapping to a symbol

→ Too sparse, extremely inefficient for a large symbol set

a bag-of-symbols

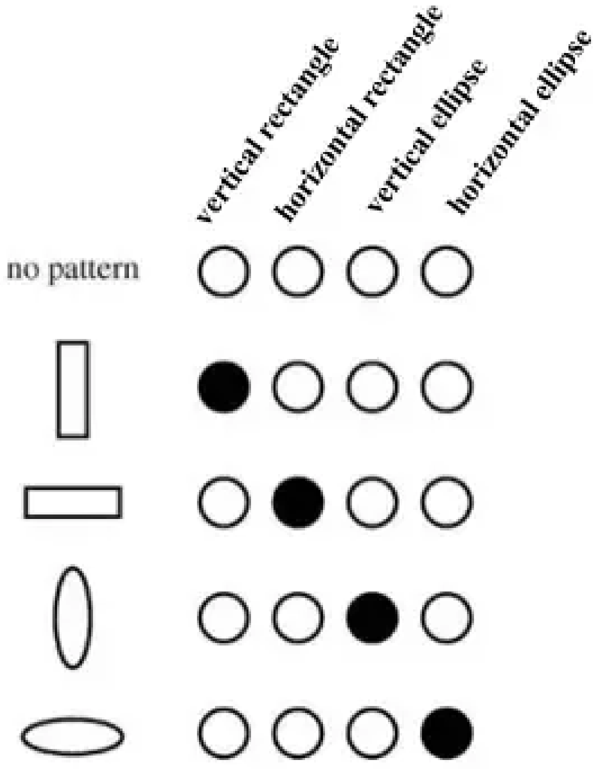
$$\mathbf{s} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

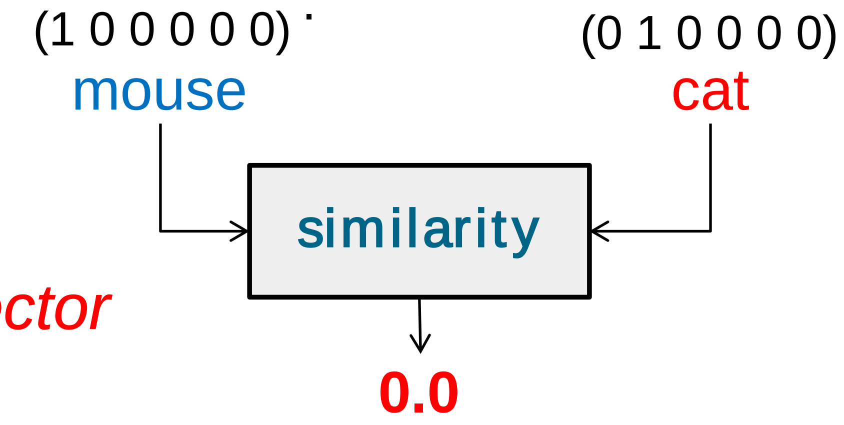
→ symbolic sequences cannot be fully reconstructed

but it is possible to know which symbols were in the sequence.

→ does not preserve sequence order

Limitations of Local Representation





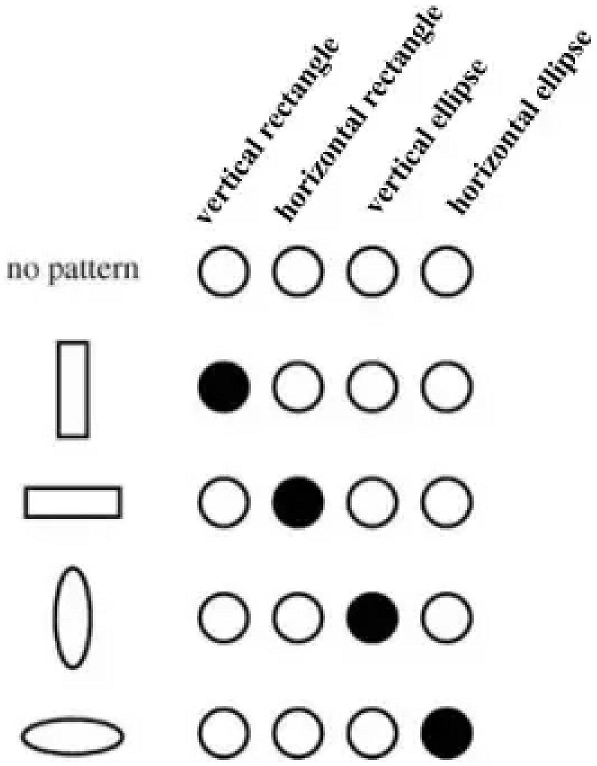
One-hot vector

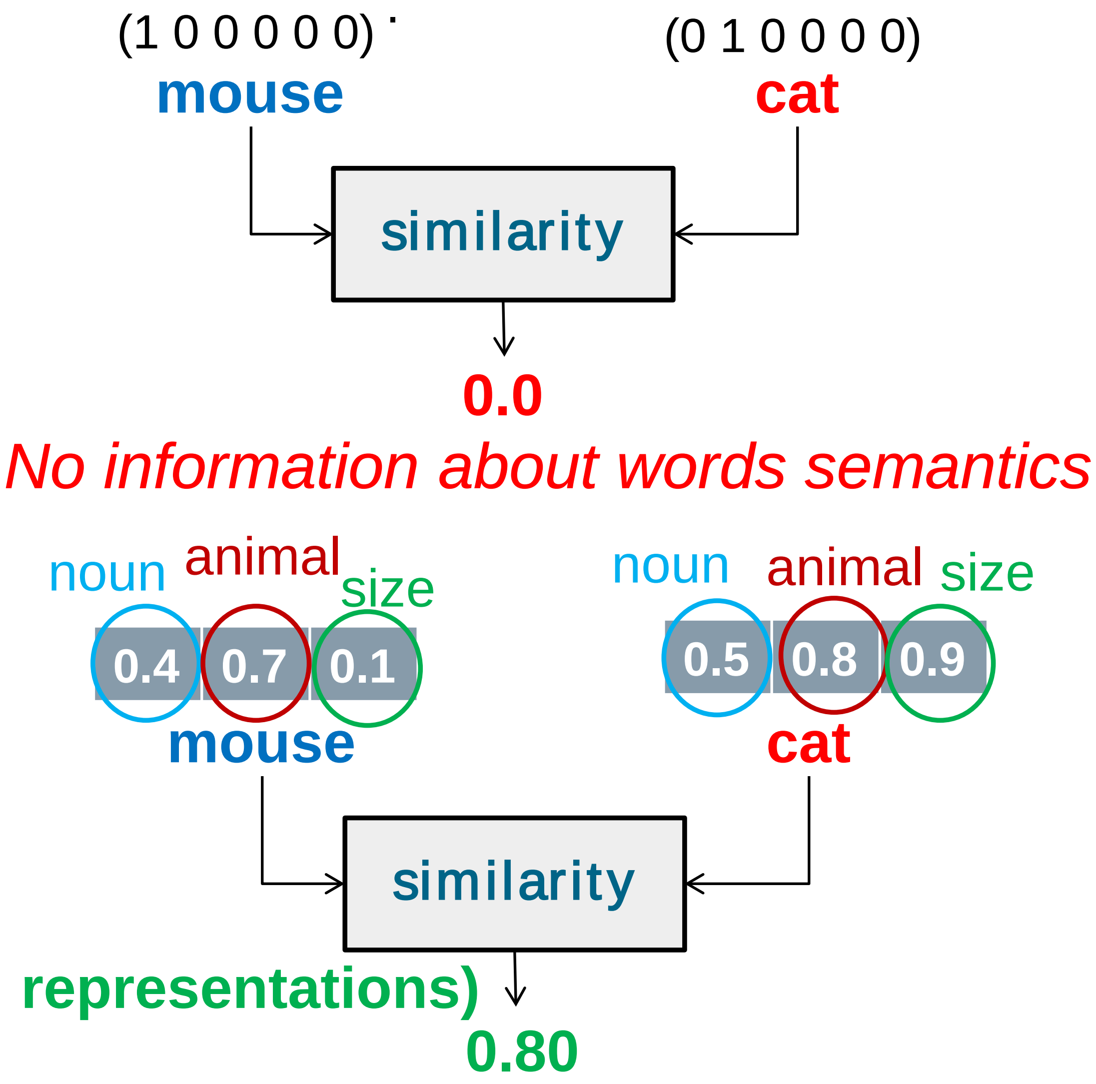
No information about words semantics

https://www.cc.gatech.edu/classes/AY2019/cs7643_fall/slides/L1_intro.pdf

Motivation: Local Representation of Symbols

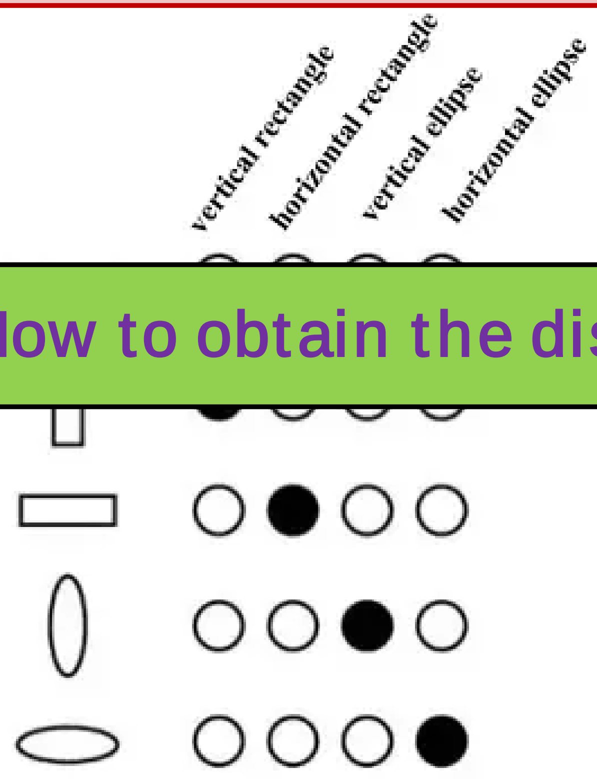
Limitations of Local Representation



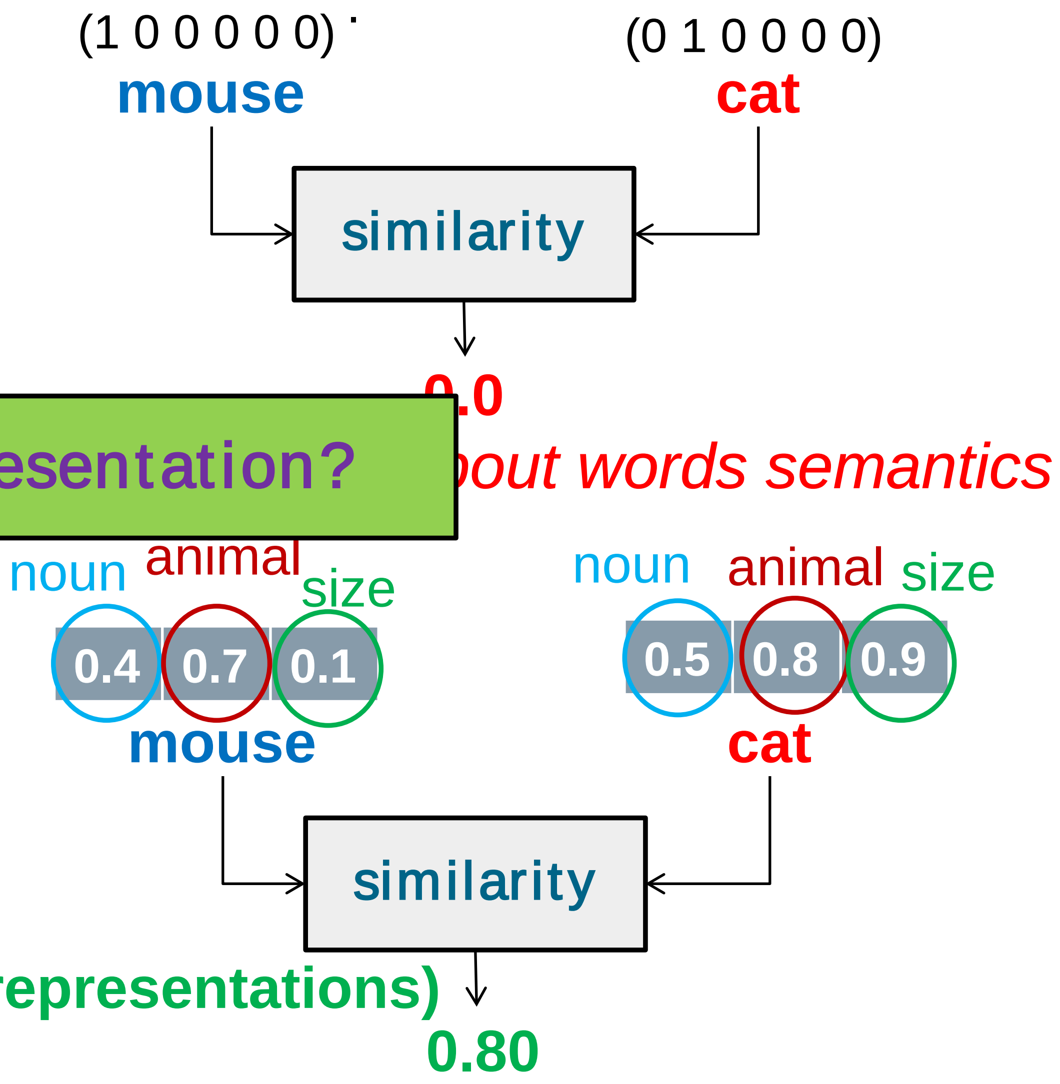


(i.e., dense vectors in distributed representations)

Limitations of Local Representation



How to obtain the distributed representation?



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

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

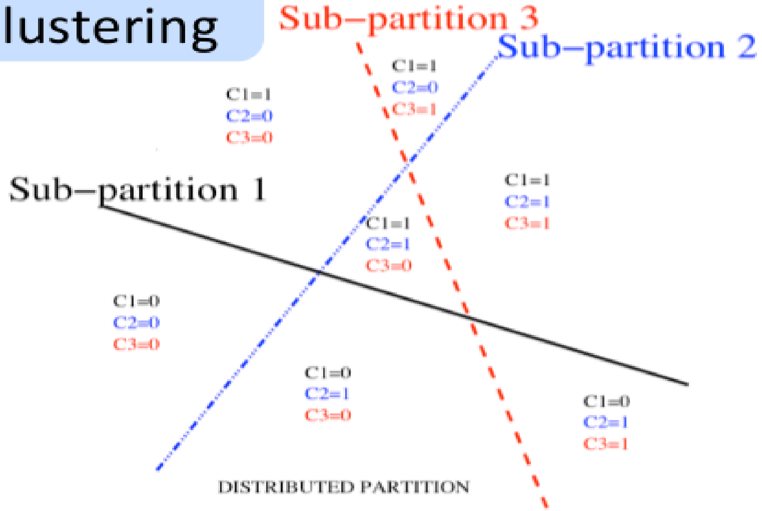
→ addresses the *Curse of Dimensionality*

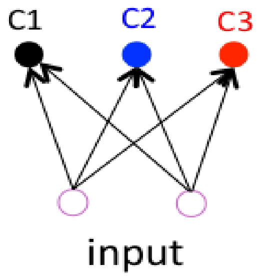
Need for Distributed Representation Learning

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**

Multi-Clustering





Non-mutually exclusive features/ attributes create a combinatorially large set of distinguiable configurations

Inspired from Y. Bengio summer school, 2015

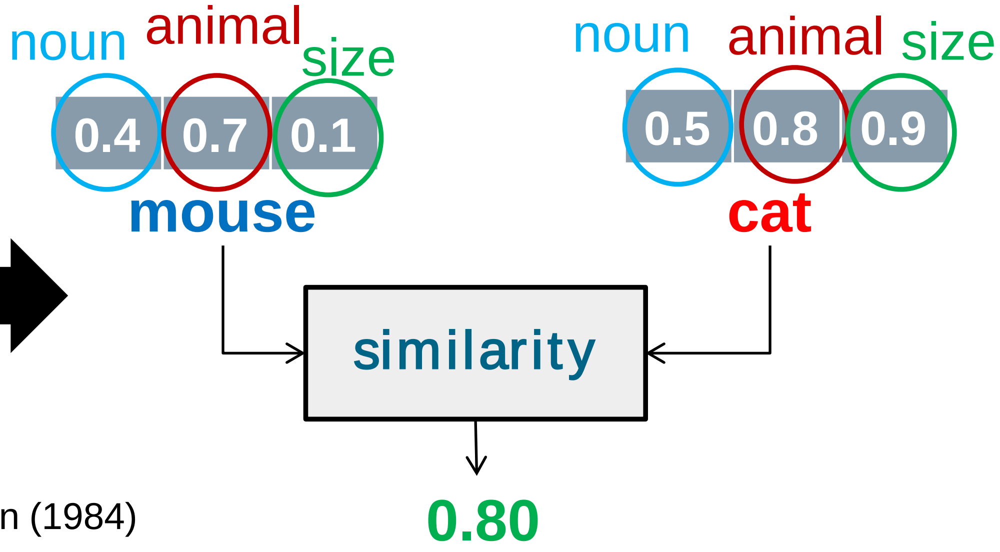
Motivation: Distributed Representation Learning

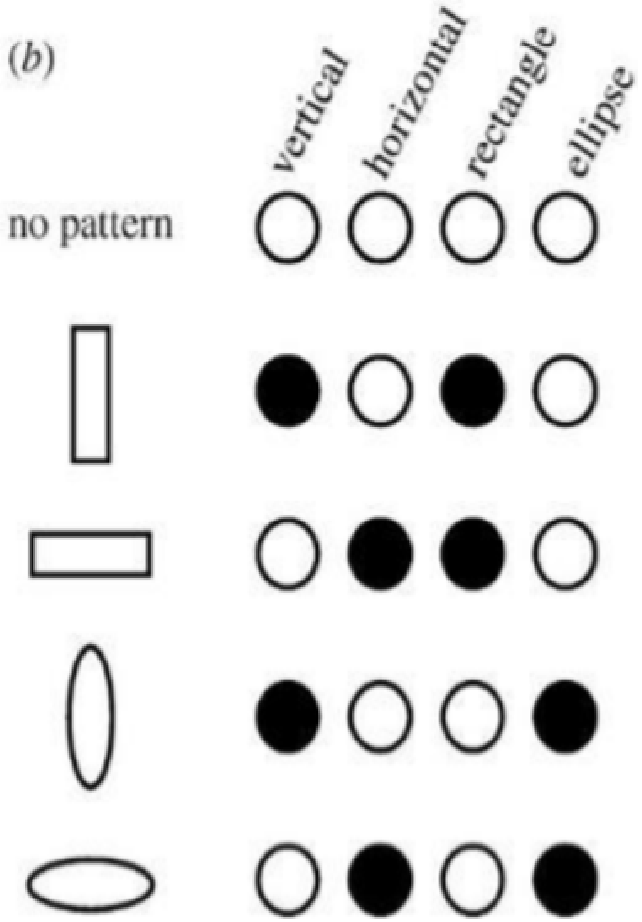
→ dense vectors in distributed representations

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

→ enable to share similarity between more than two concepts

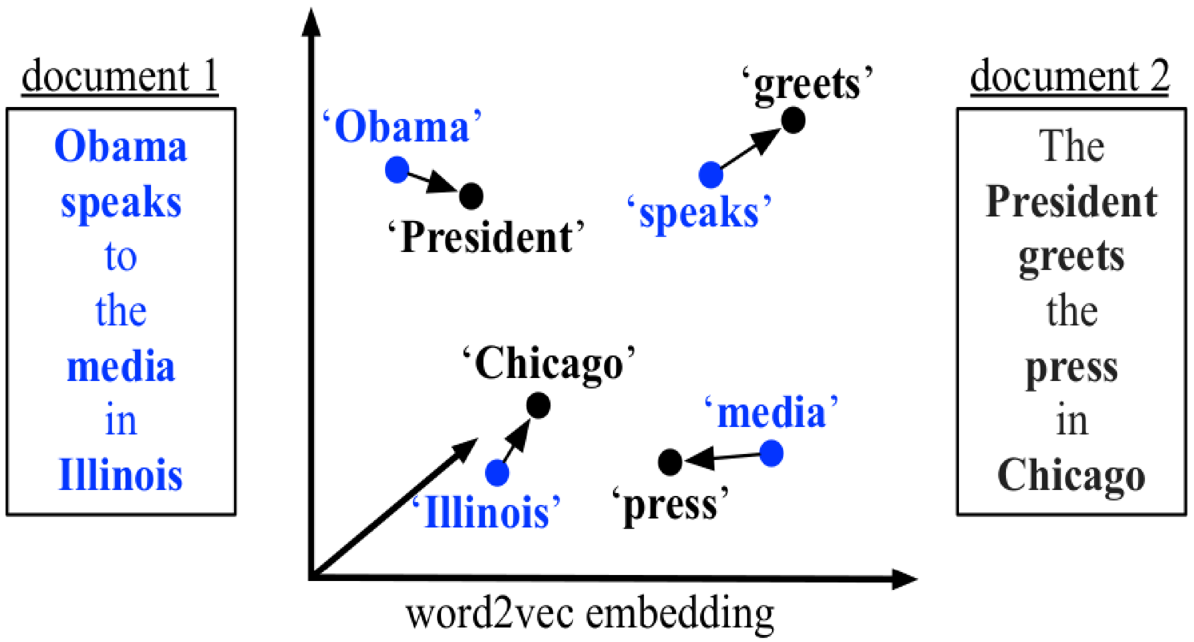
Power of Distributed Representations

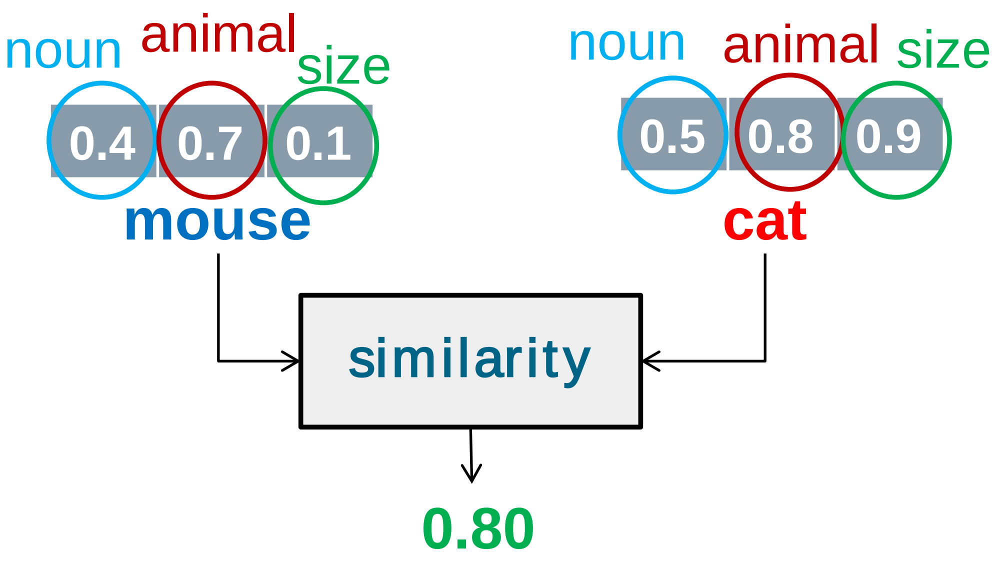




Distributed Representation by Hinton (1984)

Power of Distributed Representations





Distributed Representation by Hinton (1984)

Representations Learning

Representation Learning

components

interpretability

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

compositionality

→ Hierarchical composition
→ Deep Learning is very good !!!

Distributed Representations in Deep Leaning !!!

Yes it works, but how?

Representation learning: Attempts to ***automatically learn good features or representations***

Deep Learning: Attempts to learn ***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)?

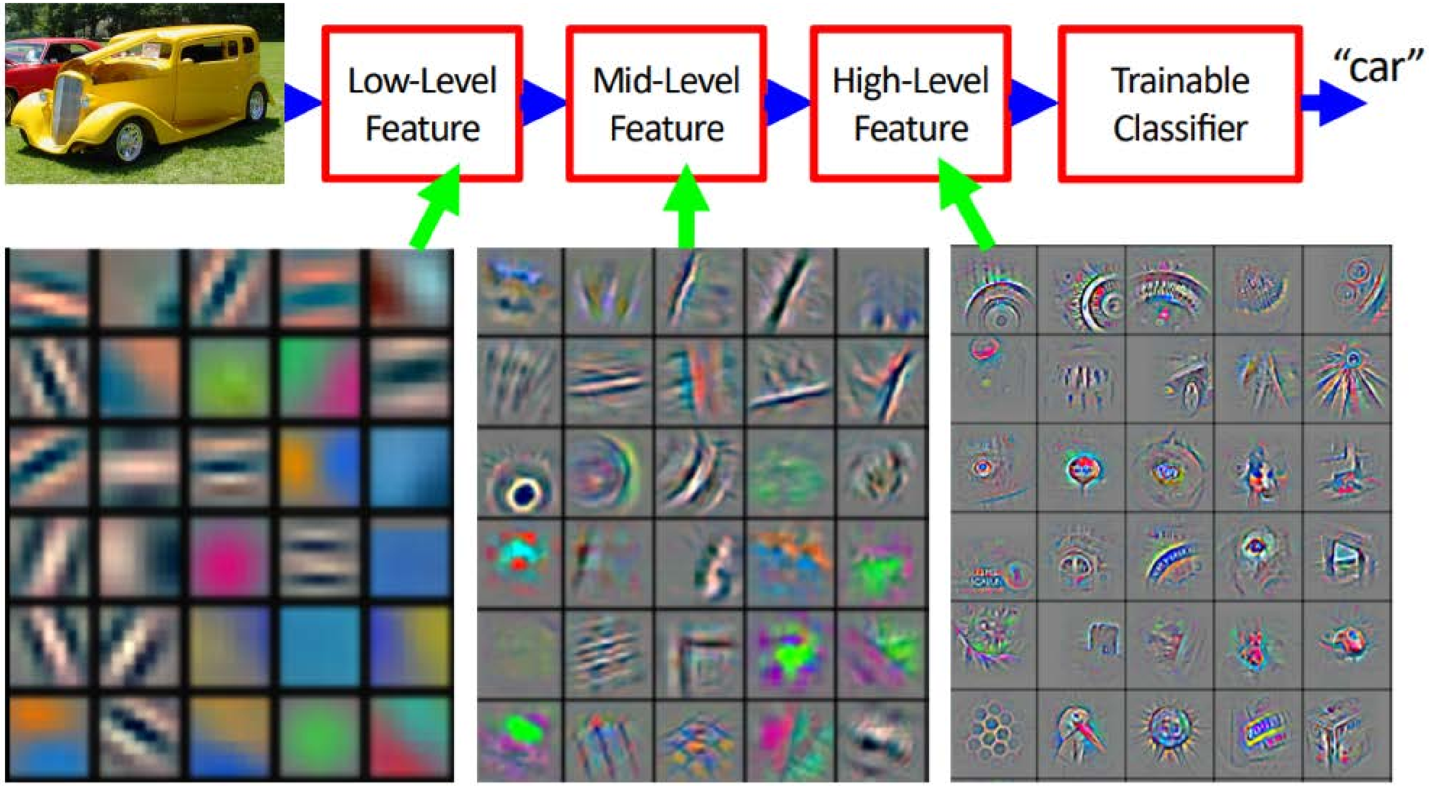
• What is each hidden unit actually representing?

• How can a neural network generalize to objects that is has never seen before?

Deep Learning = Hierarchical Compositionality

– Cascade of non-linear transformations

– Multiple layers of representations

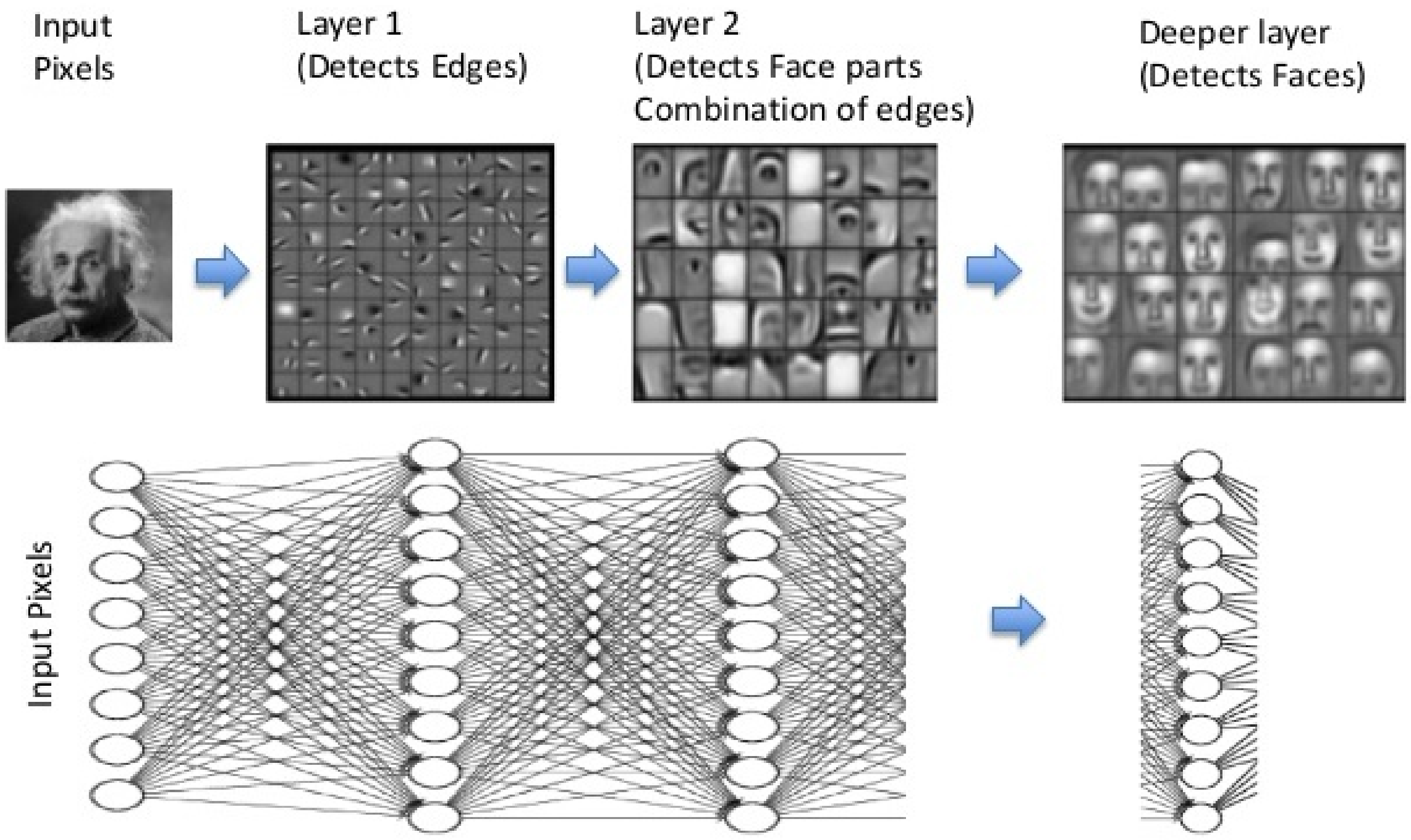


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

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 → y

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

→ Cat

Classification

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

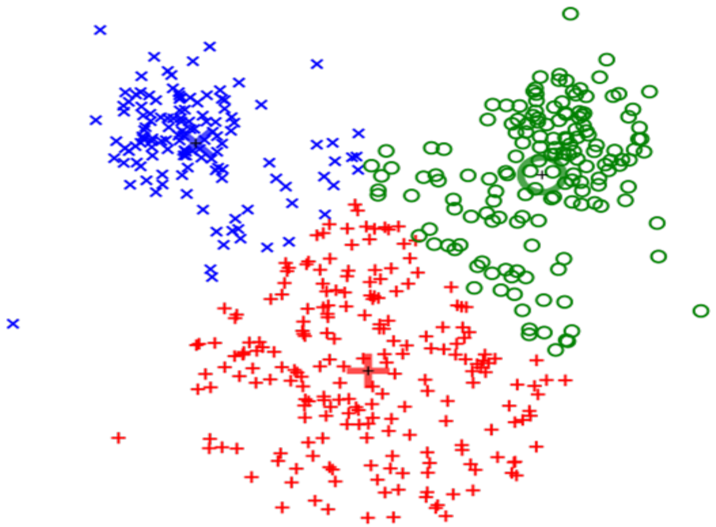
Representation Learning: Supervised vs Unsupervised

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.



K-means clustering

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

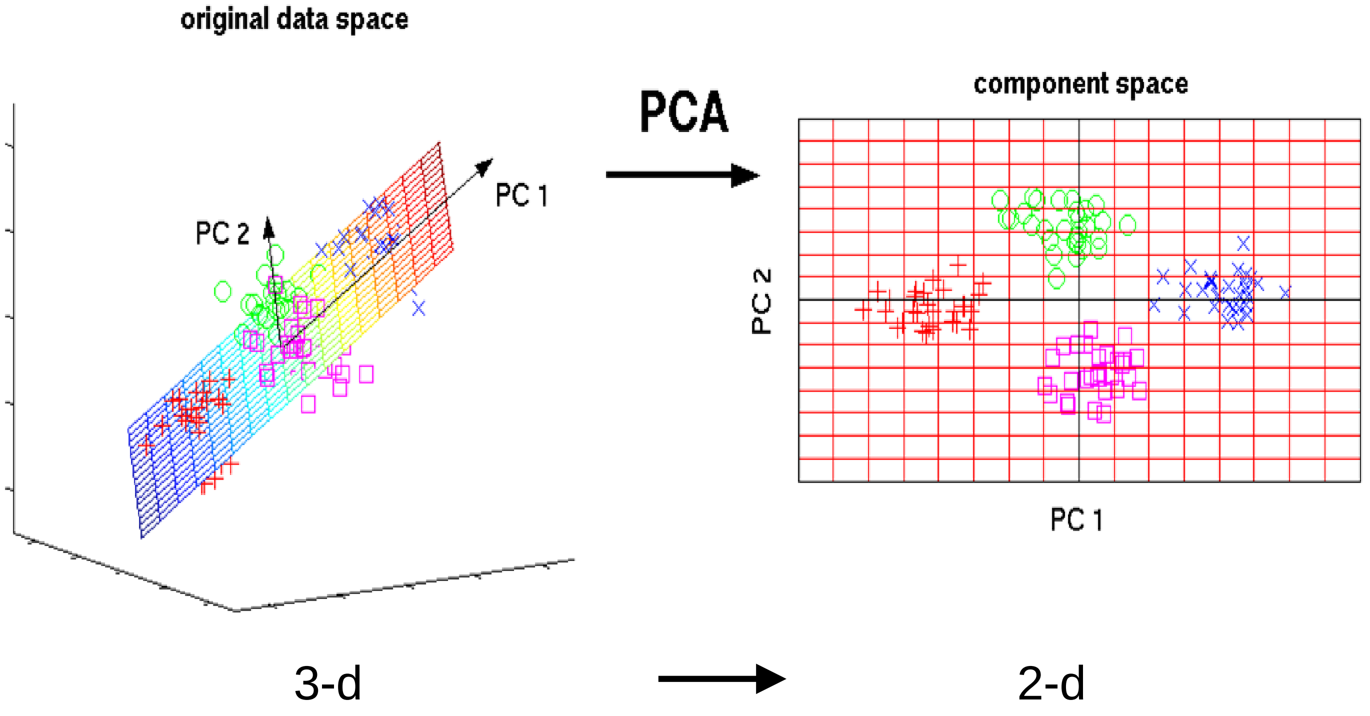
Representation Learning: Supervised vs Unsupervised

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.



Principal Component Analysis
(Dimensionality reduction)

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

Representation Learning: Supervised vs Unsupervised

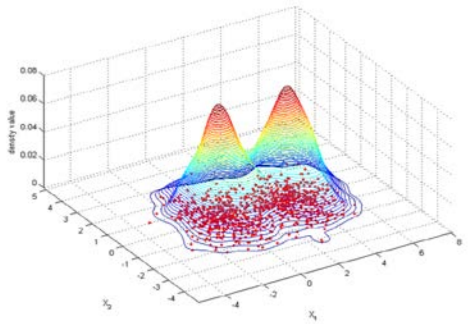
Unsupervised Learning

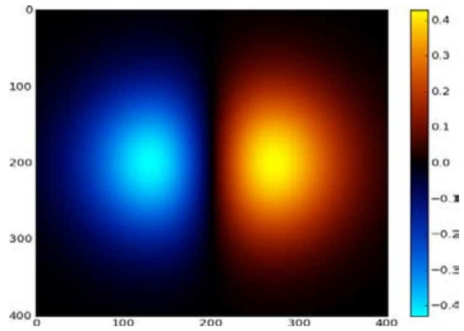
Data: x
Just data, no labels!

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

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

1-d density estimation





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 -> 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.*

Lecture 04, 05

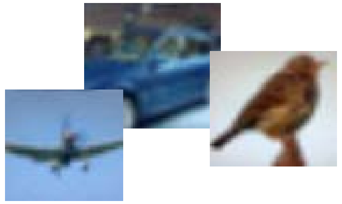
Density Estimation

→ Model **p(x)**, e.g. RBMs, VAEs, NADE, RSM, DocNADE, etc.

This lecture

[Deep] Generative Models for Representation Learning

Given training data, generate new samples from same distribution



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

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

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

Density estimation: a core problem in unsupervised learning

Several flavors:

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

→Implicit density estimation: learn model that can sample from $p_{model}(x)$ 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 exploits tons to unlabeled data

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

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

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

Density estimation: a core problem in unsupervised learning

Several flavors:

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

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

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

Why Latent Factors & Unsupervised Representation Learning? Because of Causality.

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.

http://www.iro.umontreal.ca/~bengioy/ift6266/H16/representation-learning.pdf

TO DO: Motivation: image manifold

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

http://www.iro.umontreal.ca/~bengioy/ift6266/H16/representation-learning.pdf

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)!!!

http://www.iro.umontreal.ca/~bengioy/ift6266/H16/representation-learning.pdf

BREAK

Learning Distributed Word Representations

(i.e., word embeddings)

Distributional Language Semantics: Tools for Language Modeling

SIEMENS

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

s_1	*a cat catches a mouse*
s_2	*a dog eats a mouse*
s_3	*a dog catches a cat*

Small corpus of 3 sentence/documents

$$X = \begin{array}{c} \\ a \\ cat \\ dog \\ mouse \\ catches \\ eats \end{array} \begin{array}{c} \begin{array}{ccc} s_1 & s_2 & s_3 \end{array} \\ \begin{pmatrix} 2 & 2 & 2 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array} = \begin{array}{c} \\ a \\ cat \\ dog \\ mouse \\ catches \\ eats \end{array} \begin{array}{c} \begin{array}{cccccc} a & cat & dog & mouse & catches & eats \end{array} \\ \begin{pmatrix} 0 & 1 & 2 & 2 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Local representation: BoW

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

s_1	$\boxed{a}$ $\boxed{cat}$ *catches a mouse*
s_2	*a dog eats a mouse*
s_3	*a dog catches a cat*

Small corpus of 3 sentence/documents

co-occurrence matrix

$$X = \begin{array}{c} \\ a \\ cat \\ dog \\ mouse \\ catches \\ eats \end{array} \begin{array}{c} \begin{array}{ccc} s_1 & s_2 & s_3 \end{array} \\ \begin{pmatrix} 2 & 2 & 2 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array} = \begin{array}{c} \\ \boxed{a} \\ cat \\ dog \\ mouse \\ catches \\ eats \end{array} \begin{array}{c} \begin{array}{cccccc} a & cat & dog & mouse & catches & eats \end{array} \\ \begin{pmatrix} 0 & \boxed{1} & 2 & 2 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Local representation: BoW

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

co-occurrence matrix

s_1	*a cat catches a mouse*
s_2	*a dog eats a mouse*
s_3	*a dog catches a cat*

Small corpus of 3 sentence/documents

$$X = \begin{array}{c} \\ a \\ cat \\ dog \\ mouse \\ catches \\ eats \end{array} \begin{array}{c} \begin{array}{ccc} s_1 & s_2 & s_3 \end{array} \\ \begin{pmatrix} 2 & 2 & 2 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array} = \begin{array}{c} \\ \boxed{a} \\ cat \\ dog \\ mouse \\ catches \\ eats \end{array} \begin{array}{c} \begin{array}{cccccc} a & cat & dog & mouse & catches & eats \end{array} \\ \begin{pmatrix} 0 & 1 & \boxed{2} & 2 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Local representation: BoW

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

co-occurrence matrix

s_1	$a\ cat\ \boxed{catches}\ a\ mouse$
s_2	$a\ dog\ eats\ a\ mouse$
s_3	$a\ dog\ \boxed{catches}\ a\ cat$

Small corpus of 3 sentence/documents

$$X = \begin{array}{c} \\ a \\ cat \\ dog \\ mouse \\ catches \\ eats \end{array} \begin{array}{cccccc} a & cat & dog & mouse & catches & eats \\ \hline \end{array}$$

$$X = \begin{array}{c|cccccc} & a & cat & dog & mouse & catches & eats \\ \hline a & 0 & 1 & 2 & 2 & 2 & 2 \\ cat & 2 & 0 & 0 & 0 & 1 & 0 \\ dog & 2 & 0 & 0 & 0 & 1 & 1 \\ mouse & 2 & 0 & 0 & 0 & 0 & 0 \\ catches & 2 & 1 & 1 & 0 & 0 & 0 \\ eats & 1 & 0 & 1 & 0 & 0 & 0 \end{array}$$

similar in distributed representation space

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

→ w

E.g

→ w

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

➔ Need compact distributed vectors

s_1	$a\ cat\ \boxed{catches}\ a\ mouse$
s_2	$a\ dog\ eats\ a\ mouse$
s_3	$a\ dog\ \boxed{catches}\ a\ cat$

$$X = \begin{array}{c} \\ a \\ cat \\ dog \\ mouse \\ catches \\ eats \end{array} \begin{array}{c} \begin{array}{cccccc} a & cat & dog & mouse & catches & eats \end{array} \\ \begin{pmatrix} 0 & 1 & 2 & 2 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Small corpus of 3 sentence/documents

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

WWW.TENSORFLOW.ORG/IMAGES/LINEAR-RELATIONSHIPS.PNG

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

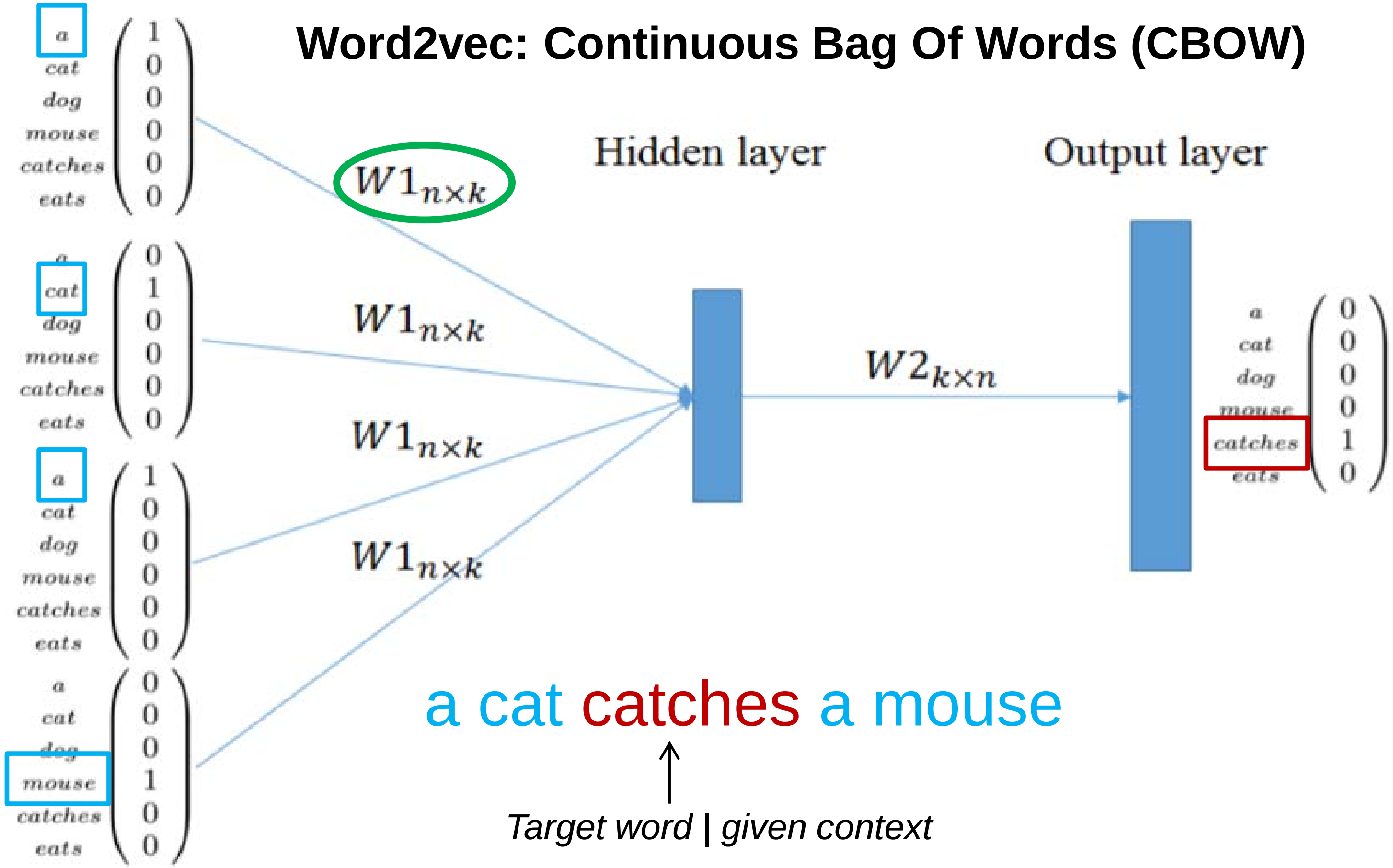
→ no explicit association between word pairs

→ *distribution of the words* learned **implicitly**

→ **compact distributed representation**

→ dimensions of vectors not interpretable

Word2vec: Continuous Bag Of Words (CBOW)

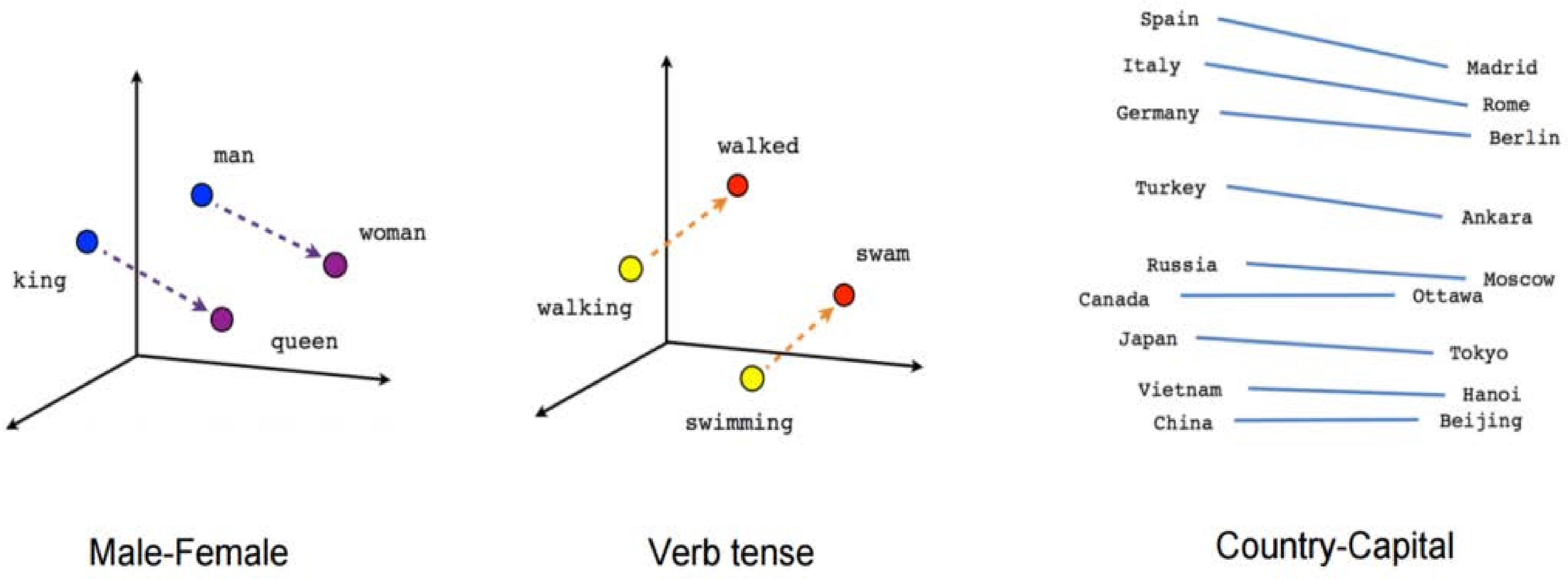


a cat catches a mouse

↑

Target word | given context

Why: Power of Distributed Word Representation



Word analogy using vector operation: http://bionlp-www.utu.fi/wv_demo/ *Let's TRY !!*

Further reading: doc2vec, LDA, LSA, etc.

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

[Deep] Generative Models for Representation Learning

we will discuss the highlighted ones

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

Auto-encoders (AEs)

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

$\boxed{\textbf{Encoder:}}$ compresses the input into a latent-space representation

maps from feature space back into input space

feature-vector or representation

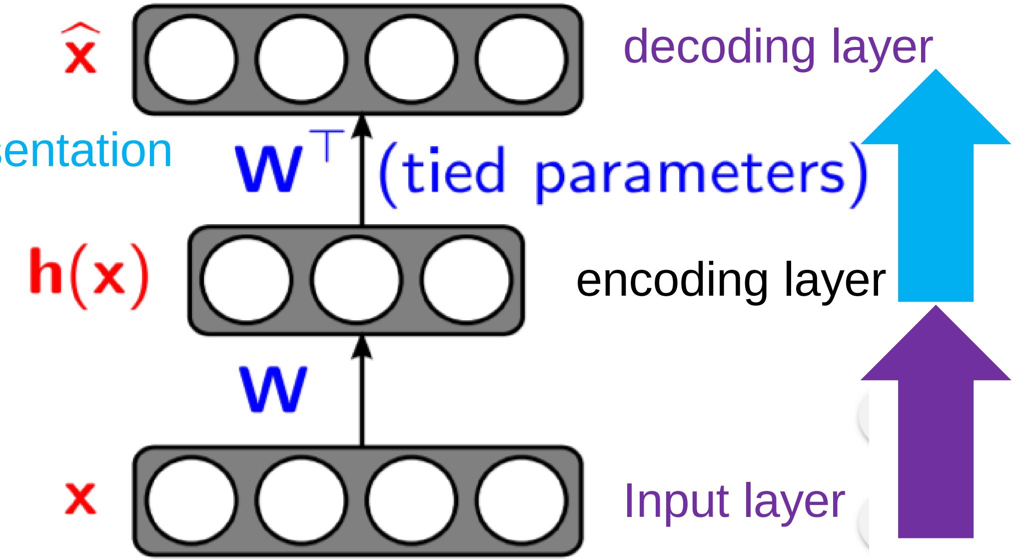
$$h(x) = g(\mathbf{W}x + b)$$

input

$\boxed{\textbf{Decoder:}}$ reconstruct the input from the latent representation

Reconstructed input →

$$\widehat{x} = \mathbf{W}^\top h(x) + c$$



Loss function (for real-valued inputs):

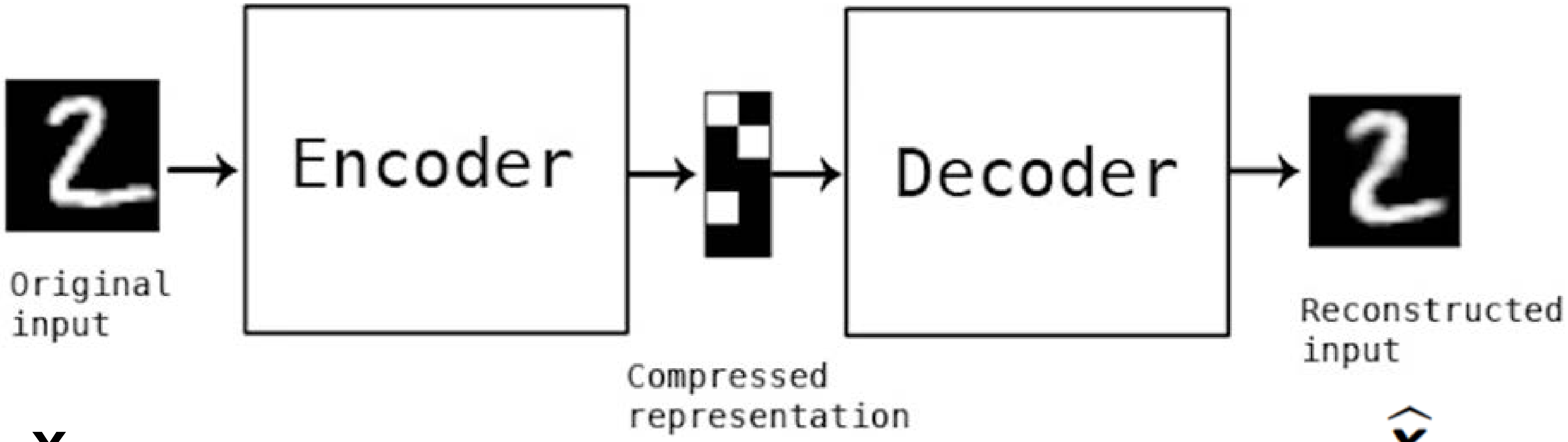
(minimizing reconstruction error by SGD)

$$L(\widehat{x}; x) = \frac{1}{2}\|\widehat{x} - x\|^2$$

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

https://andre-martins.github.io/docs/dsl2018/lecture_06.pdf (LeCun, 1987; Bourlard and Kamp, 1988; Hinton and Zemel, 1994)

Auto-encoders (AEs)



$\mathbf{x}$ *trained to copy input to output* $\mathbf{h}$ $\hat{\mathbf{x}}$

(representation encoding input)

compression

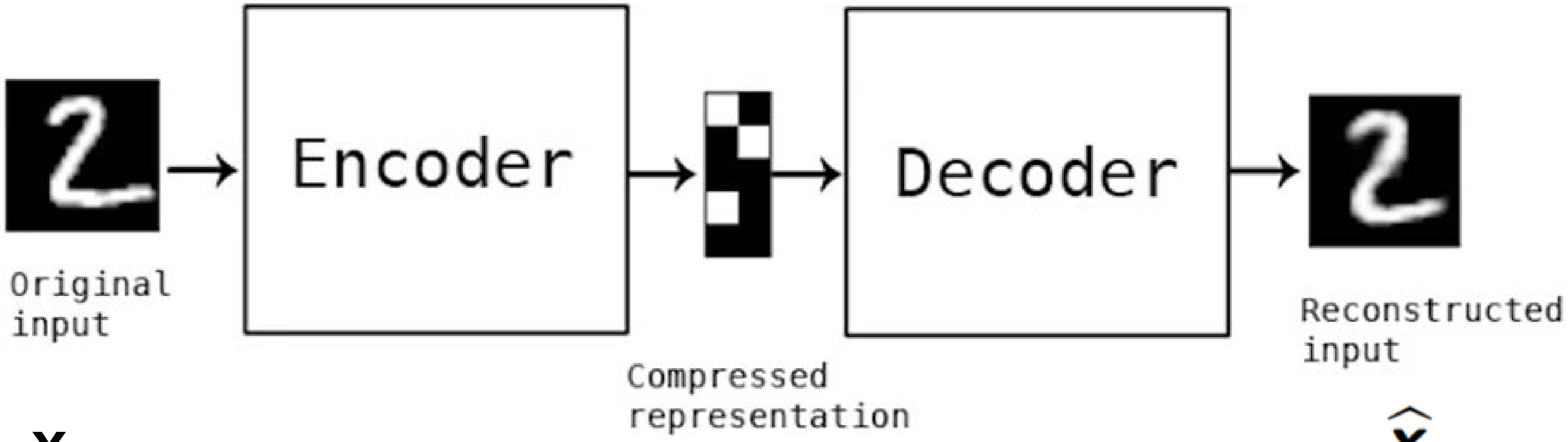
captures useful properties of the data distribution **or** *means to represent data* **or** *explanatory factors*

→ *dimension(h) < dimension(x)* ➔ ***undercomplete i.e. learn to capture useful features***

→ *dimension(h) > dimension(X)* ➔ ***overcomplete*** *i.e. learn copy input to output*

→ *Difficult to interpret*

Auto-encoders (AEs)



$\mathbf{x}$ *trained to copy input to output* $\mathbf{h}$ $\hat{\mathbf{x}}$

(representation encoding input)

compression

Undercomplete AE → capture useful info about data

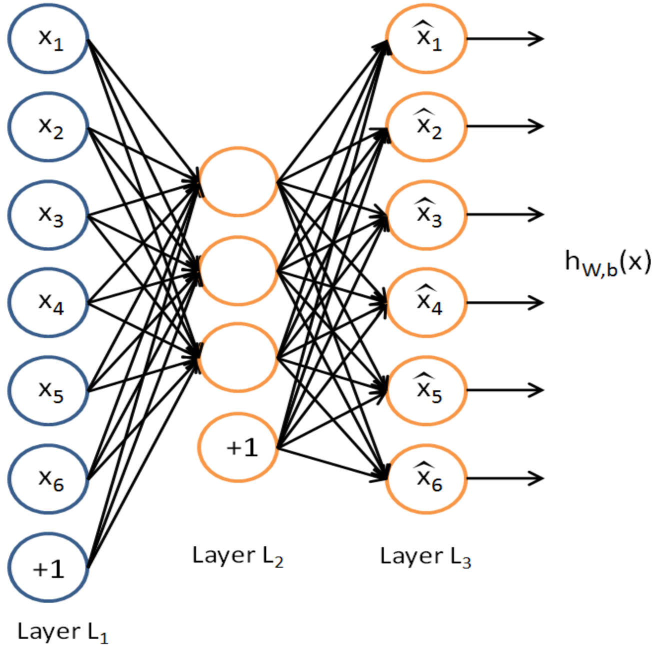
Overcomplete AE → still captures interesting features, but apply constraints on X or h

→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/

Limitations of (regular) Autoencoders

→ Need to Conceptualize, not only compress…….

Different variants to rescue……………….!!!

Offline

Autoencoder variants:

1. *Denoising* Autoencoders
2. *Sparse* Autoencoders

Idea: *Constraint the reconstruction of an autoencoder*

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

……..details later in the lecture series (Lecture on "Generative Models")

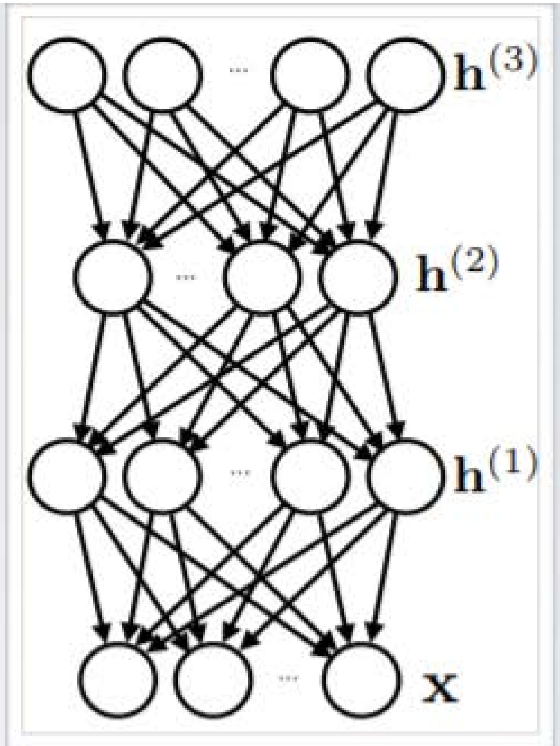
Family of Neural Generative Models

Family of Generative Models

- **Directed graphical models**
  - define prior over top-most latent representation
  - define conditionals from top latent representation to observation

$$p(\mathbf{x}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = p(\mathbf{x}|\mathbf{h}^{(1)})p(\mathbf{h}^{(1)}|\mathbf{h}^{(2)})p(\mathbf{h}^{(2)}|\mathbf{h}^{(3)})p(\mathbf{h}^{(3)})$$

  - examples: variational autoencoders (VAE), generative adversarial networks (GAN), sparse coding, helmholtz machines



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

(details in Lecture on "Generative Models")

https://ift6135h18.wordpress.com/author/aaroncourville/page/1/

Family of Generative Models

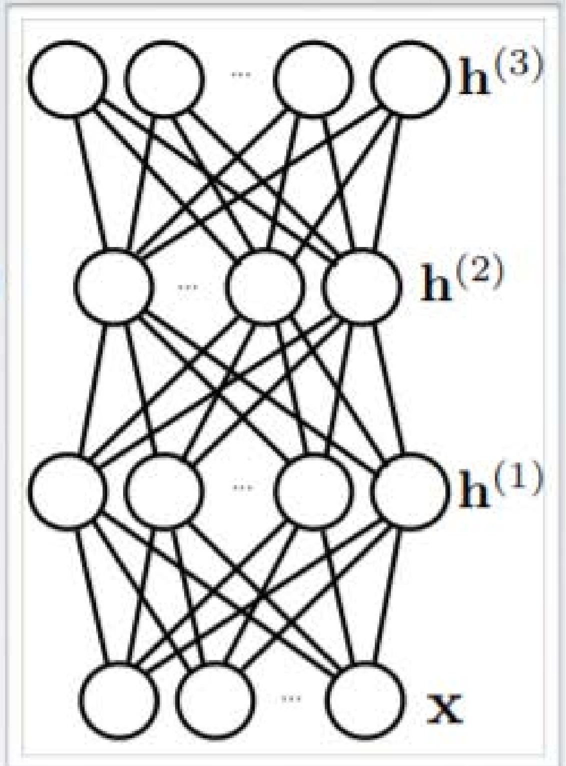
- **Undirected graphical models**
  - define a joint energy function

$$E(\mathbf{x},\mathbf{h}^{(1)},\mathbf{h}^{(2)},\mathbf{h}^{(3)}) = -\mathbf{x}\mathbf{W}^{(1)}\mathbf{h}^{(1)} - \mathbf{h}^{(2)}\mathbf{W}^{(2)}\mathbf{h}^{(3)} - \mathbf{h}^{(3)}\mathbf{W}^{(3)}\mathbf{h}^{(4)}$$

  - exponentiate and normalize

$$p(\mathbf{x},\mathbf{h}^{(1)},\mathbf{h}^{(2)},\mathbf{h}^{(3)}) = \exp\left(-E(\mathbf{x},\mathbf{h}^{(1)},\mathbf{h}^{(2)},\mathbf{h}^{(3)})\right)/Z$$

  - examples: deep Boltzmann machines (DBM), deep energy models



- Properties
  - *pros:* can compute $p(\mathbf{x})$ up to a multiplicative factor (true for RBMs not general BMs)
  - *cons:* hard to sample from (MCMC), $p(\mathbf{x})$ is intractable, so hard to train

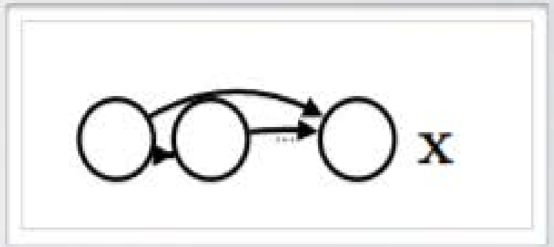
https://ift6135h18.wordpress.com/author/aaroncourville/page/1/

Family of Generative Models

- **Autoregressive generative models**
  - choose an ordering of the dimensions in $\mathbf{x}$
  - define the conditionals in the product rule expression of $p(\mathbf{x})$

$$p(\mathbf{x}) = \prod_{k=1}^{D} p(x_k | \mathbf{x}_{<k})$$

  - examples: masked autoencoder distribution estimator (MADE), pixelCNN
    neural autoregressive distribution estimator (NADE), spatial LSTM, pixelRNN



- Properties
  - *pros:* $p(\mathbf{x})$ 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

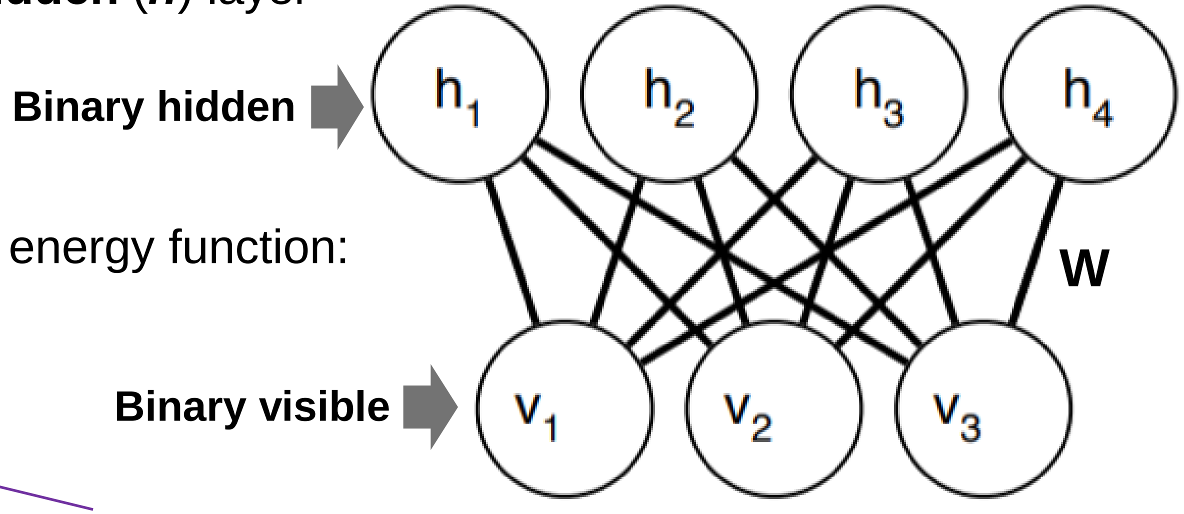
→ transform data into latent feature space and then reconstruct to learn data distribution

→ *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:



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

partition function **energy** **binary visible** **binary hidden features**

Further reading: https://ift6266h15.files.wordpress.com/2015/03/chapter21.pdf

Restricted Boltzmann Machines (RBMs)

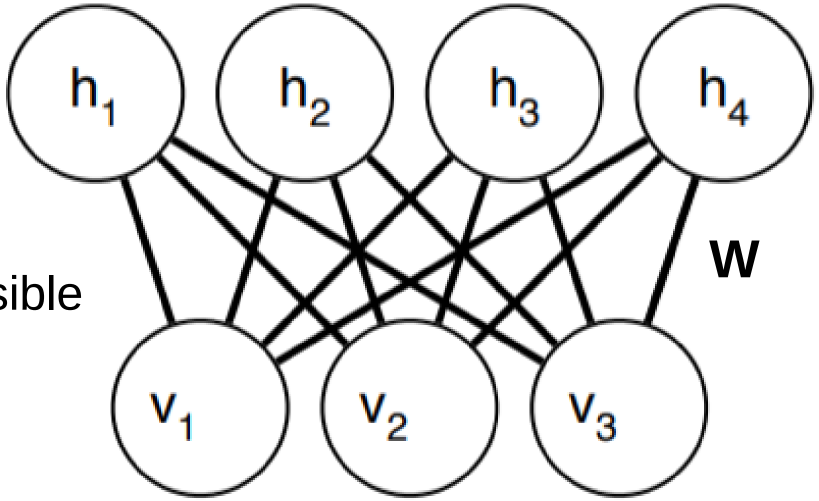
→ Two layers: binary **visible** (***v***) and binary **hidden** (***h***) layer

→ ***energy-based models,***

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

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

partition function **energy** **binary visible** **binary hidden features**



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

$$E(\boldsymbol{v},\boldsymbol{h}) = -\boldsymbol{b}^\top\boldsymbol{v} - \boldsymbol{c}^\top\boldsymbol{h} - \boldsymbol{v}^\top\boldsymbol{W}\boldsymbol{h}$$

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

$$Z = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}}\exp\{-E(\boldsymbol{v},\boldsymbol{h})\}$$

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

$2^{\mathrm{dim}(v)}\,2^{\mathrm{dim}(h)} \Rightarrow 2^{2000}\,2^{50} \Rightarrow$ **intractable**

summing over all states ➔ summing exponential number of terms ➔ computationally **intractable** P(**v**,**h**)

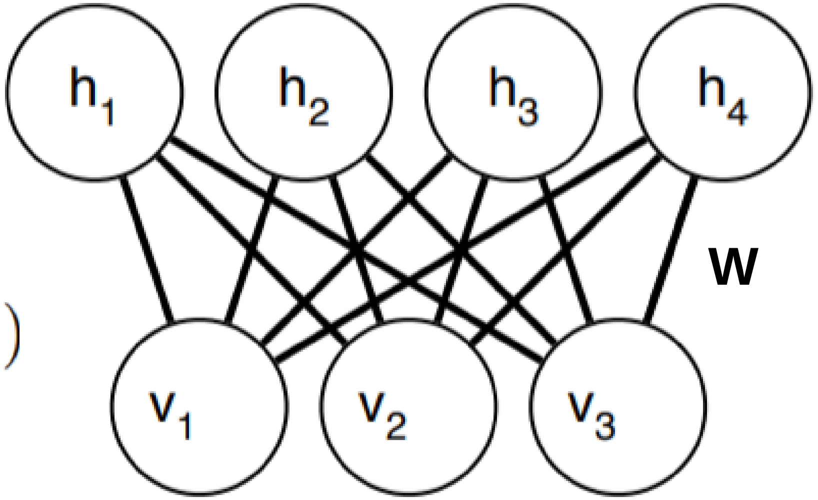
Restricted Boltzmann Machines (RBMs)

→ **energy-based models,**

Therefore, *joint probability distribution* is given by its energy function: $P(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \exp\{-E(\boldsymbol{v}, \boldsymbol{h})\}$.

→ **energy function,** $E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}$

→ **partition function** $Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\{-E(\boldsymbol{v}, \boldsymbol{h})\}$



→ **conditional distributions** from the joint distribution

Full conditional over the **hidden layer** as the factorial distribution:

$$p(\mathbf{h}|\mathbf{v}) = \Pi_{j=1}^{n} p(h_j|\mathbf{v}) = \Pi_{j=1}^{n} \text{sigmoid}(c_j + \mathbf{v}^T W_{:,j})$$

Full conditional over the **visible layer** as the factorial distribution:

$$p(\mathbf{v}|\mathbf{h}) = \Pi_{i=1}^{d} p(v_i|\mathbf{h}) = \Pi_{i=1}^{d} \text{sigmoid}(b_i + W_{i,:}\mathbf{h})$$

→ **energy-based models,**

Therefore, *joint probability distribution* is given by its energy function: $P(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{Z}\exp\{-E(\boldsymbol{v},\boldsymbol{h})\}.$

→ **energy function,** $E(\boldsymbol{v},\boldsymbol{h}) = -\boldsymbol{b}^\top\boldsymbol{v} - \boldsymbol{c}^\top\boldsymbol{h} - \boldsymbol{v}^\top\boldsymbol{W}\boldsymbol{h}$

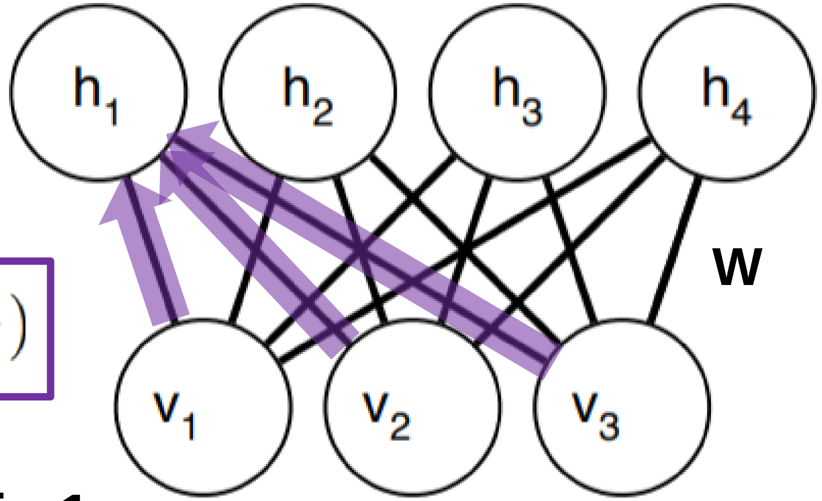
→ **partition function** $Z = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}}\exp\{-E(\boldsymbol{v},\boldsymbol{h})\}$

→ **conditional distributions** from the joint distribution

Full conditional over the **hidden layer** as the factorial distribution:

$$\boxed{p(\mathbf{h}|\mathbf{v}) = \Pi_{j=1}^{n} p(h_j|\mathbf{v}) = \Pi_{j=1}^{n}\mathrm{sigmoid}(c_j + \mathbf{v}^T W_{:,j})}$$

encoding



j =1

$$\boxed{P(h_j = 1 \mid \boldsymbol{v}) = \mathrm{sigmoid}\left(c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right)}$$

→ **energy-based models,**

Therefore, *joint probability distribution* is given by its energy function: $P(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \exp\{-E(\boldsymbol{v}, \boldsymbol{h})\}.$

→ **energy function,** $E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}$

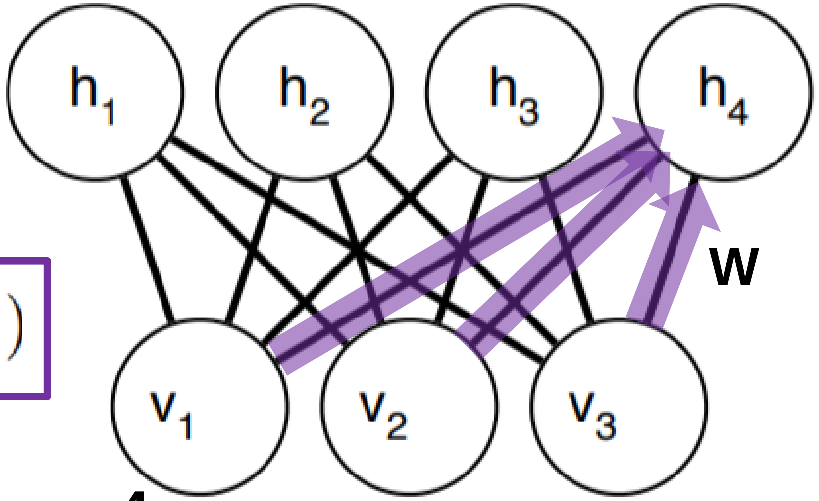
→ **partition function** $Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\{-E(\boldsymbol{v}, \boldsymbol{h})\}$

→ **conditional distributions** from the joint distribution

Full conditional over the **hidden layer** as the factorial distribution:

$$\boxed{p(\mathbf{h}|\mathbf{v}) = \Pi_{j=1}^{n} p(h_j|\mathbf{v}) = \Pi_{j=1}^{n} \mathrm{sigmoid}(c_j + \mathbf{v}^T W_{:,j})}$$

encoding



j =1,…, 4

$$\boxed{P(h_j = 1 \mid \boldsymbol{v}) = \mathrm{sigmoid}\left(c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right)}$$

Restricted Boltzmann Machines (RBMs)

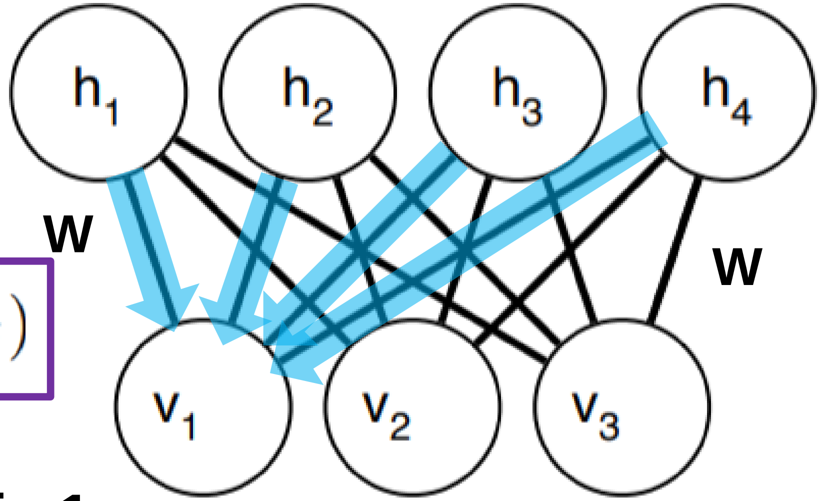
→ **energy-based models,**

Therefore, *joint probability distribution* is given by its energy function: $P(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{Z}\exp\{-E(\boldsymbol{v},\boldsymbol{h})\}.$

→ **energy function,** $E(\boldsymbol{v},\boldsymbol{h}) = -\boldsymbol{b}^\top\boldsymbol{v} - \boldsymbol{c}^\top\boldsymbol{h} - \boldsymbol{v}^\top\boldsymbol{W}\boldsymbol{h}$

→ **partition function** $Z = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}}\exp\{-E(\boldsymbol{v},\boldsymbol{h})\}$

Same W matrix in encoding and decoding



→ **conditional distributions** from the joint distribution

Full conditional over the **hidden layer** as the factorial distribution:

$$\boxed{p(\mathbf{h}|\mathbf{v}) = \Pi_{j=1}^{n} p(h_j|\mathbf{v}) = \Pi_{j=1}^{n}\text{sigmoid}(c_j + \mathbf{v}^T W_{:,j})}$$

encoding

Full conditional over the **visible layer** as the factorial distribution:

i =1

$$\boxed{p(\mathbf{v}|\mathbf{h}) = \Pi_{i=1}^{d} p(v_i|\mathbf{h}) = \Pi_{i=1}^{d}\text{sigmoid}(b_i + W_{i,:}\mathbf{h})}$$

$$\boxed{p(v_i|\mathbf{h}) = \text{sigmoid}(b_i + W_{i,:}\mathbf{h})}$$

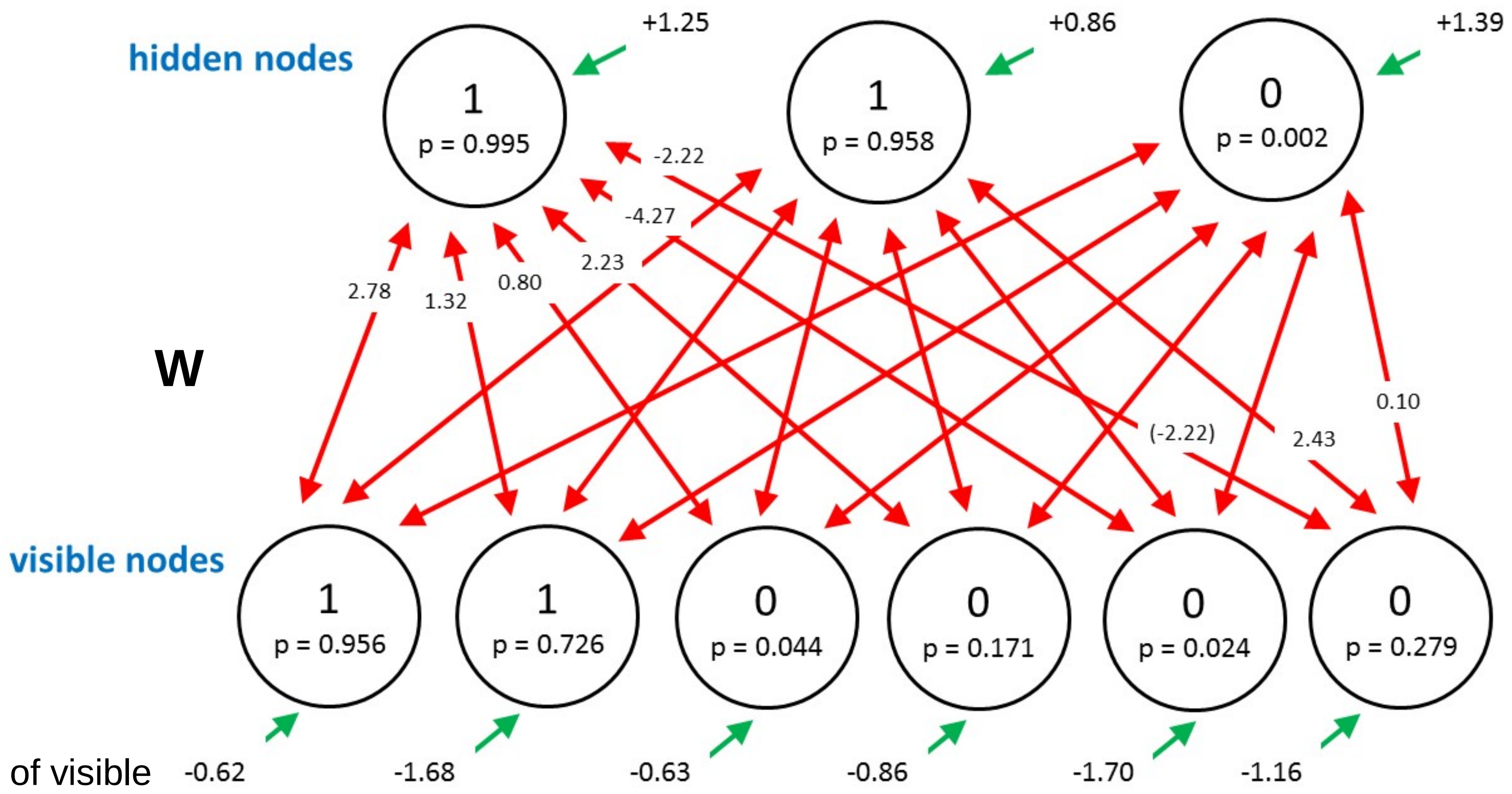
Decoding/reconstruction

Restricted Boltzmann Machines (RBMs): Illustration

Indoor-sport *sport* *outdoor-sport*

b, bias of hidden

W



c, bias of visible

Chess **Squash** **Hockey** **Tennis** **Cricket** **Football**

https://jamesmccaffrey.wordpress.com/2017/06/02/restricted-boltzmann-machines-using-c/

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) $\{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(t)}, \ldots, \boldsymbol{v}^{(n)}\}$. The log likelihood under the RBM with parameters $\boldsymbol{b}$ (visible unit biases), $\boldsymbol{c}$ (hidden unit biases) and $\boldsymbol{W}$ (interaction weights) is given by:

$$
\begin{aligned}
\ell(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}) &= \boxed{\sum_{t=1}^{n} \log P(\boldsymbol{v}^{(t)})} \\
&= \sum_{t=1}^{n} \log \sum_{\boldsymbol{h}} P(\boldsymbol{v}^{(t)}_{n,:}, \boldsymbol{h}) \\
&= \left(\sum_{t=1}^{n} \log \sum_{\boldsymbol{h}} \exp\left\{ -E(\boldsymbol{v}^{(t)}, \boldsymbol{h}) \right\} \right) - n \log \boxed{Z} \\
&= \left(\sum_{t=1}^{n} \log \sum_{\boldsymbol{h}} \exp\left\{ -E(\boldsymbol{v}^{(t)}, \boldsymbol{h}) \right\} \right) - n \log \sum_{\boldsymbol{v}, \boldsymbol{h}} \exp\{ -E(\boldsymbol{v}, \boldsymbol{h}) \}
\end{aligned}
$$

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

Cost: maximize log-likelihood of the data, v

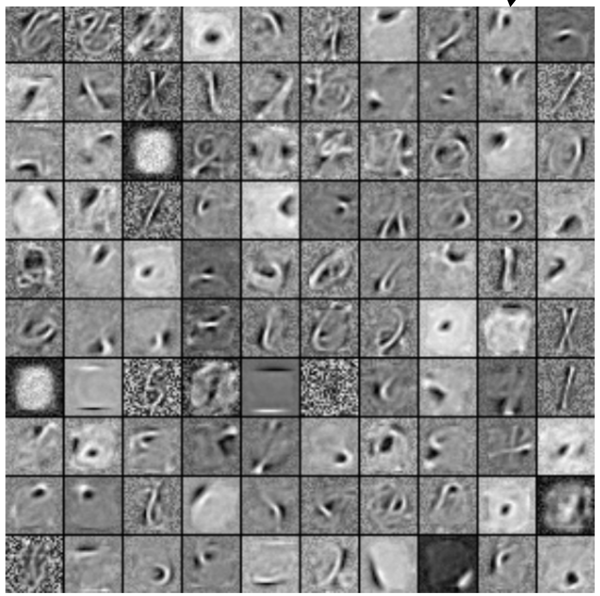
Let us consider that we have a batch (or minibatch) of n examples taken from an i.i.d dataset (independently and identically distributed examples) $\{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(t)}, \ldots, \boldsymbol{v}^{(n)}\}$. The log likelihood under the RBM with parameters $\boldsymbol{b}$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

$$
\begin{aligned}
\ell(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}) &= \boxed{\sum_{t=1}^{n} \log P(\boldsymbol{v}^{(t)})} \\
&= \sum_{t=1}^{n} \log \sum_{\boldsymbol{h}} P(\boldsymbol{v}^{(t)}_{n,:}, \boldsymbol{h}) \\
&= \left(\sum_{t=1}^{n} \log \sum_{\boldsymbol{h}} \exp\left\{-E(\boldsymbol{v}^{(t)}, \boldsymbol{h})\right\} \right) - n \log \boxed{Z} \\
&= \left(\sum_{t=1}^{n} \log \sum_{\boldsymbol{h}} \exp\left\{-E(\boldsymbol{v}^{(t)}, \boldsymbol{h})\right\} \right) - n \log \sum_{\boldsymbol{v},\boldsymbol{h}} \exp\left\{-E(\boldsymbol{v}, \boldsymbol{h})\right\}
\end{aligned}
$$

Trained efficiently using contrastive divergence (CD or PCD)

Impractical to compute the exact log-likelihood gradient

partition function (intractable)



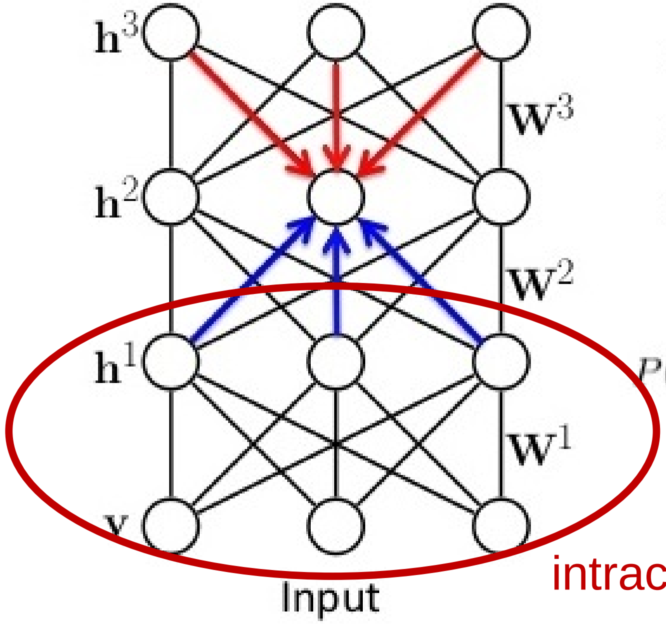
https://ift6266h15.files.wordpress.com/2015/03/chapter21.pdf

Stacked RBMs: Deep Boltzmann Machine (DBM)

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

Deep Boltzmann Machine

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

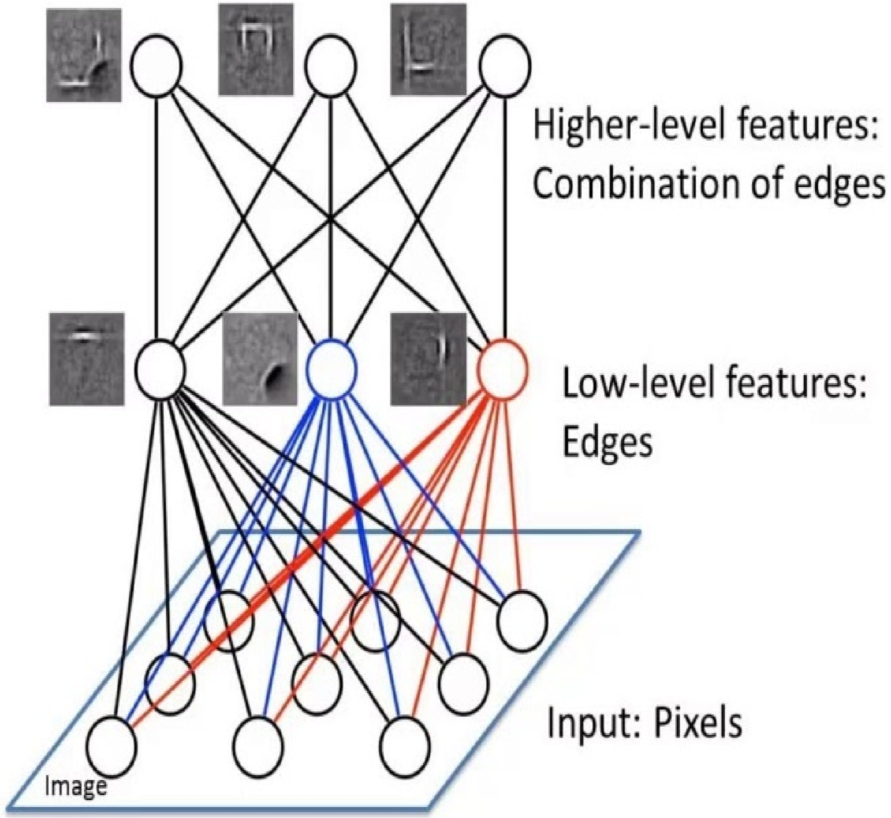


intractable

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

$$P(h_j^2 = 1|\mathbf{h}^1,\mathbf{h}^3) = \sigma\left(\sum_k W^3_{kj} h^3_k + \sum_m W^2_{mj} h^1_m\right)$$

Top-down Bottom-up

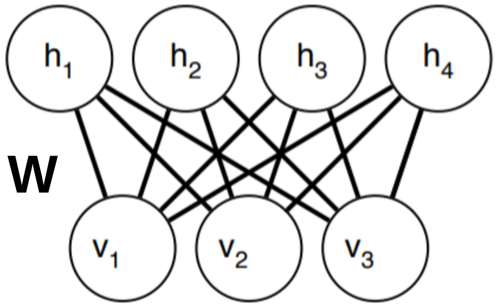


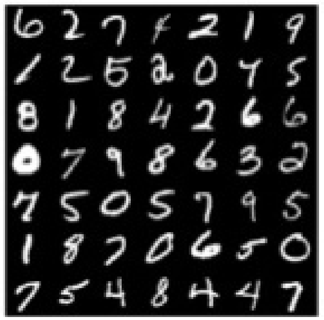
Detailed lecture: https://www.youtube.com/watch?v=MnGXXDjGNd0

Applications of RBMs

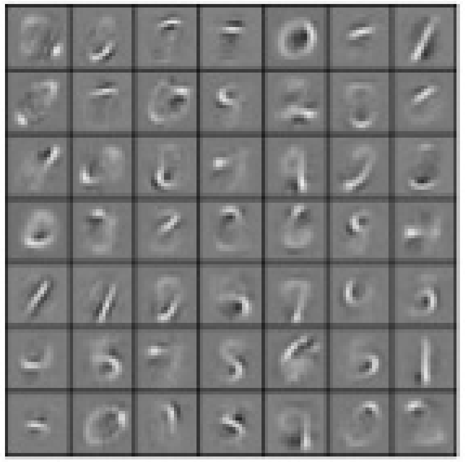
→ Unsupervised pre-training and transfer learning

→ Once trained, can use **W** and **biases** as initial values for a neural net!





Training examples



Filter (**W**) from 1st layer: “pen-strokes”

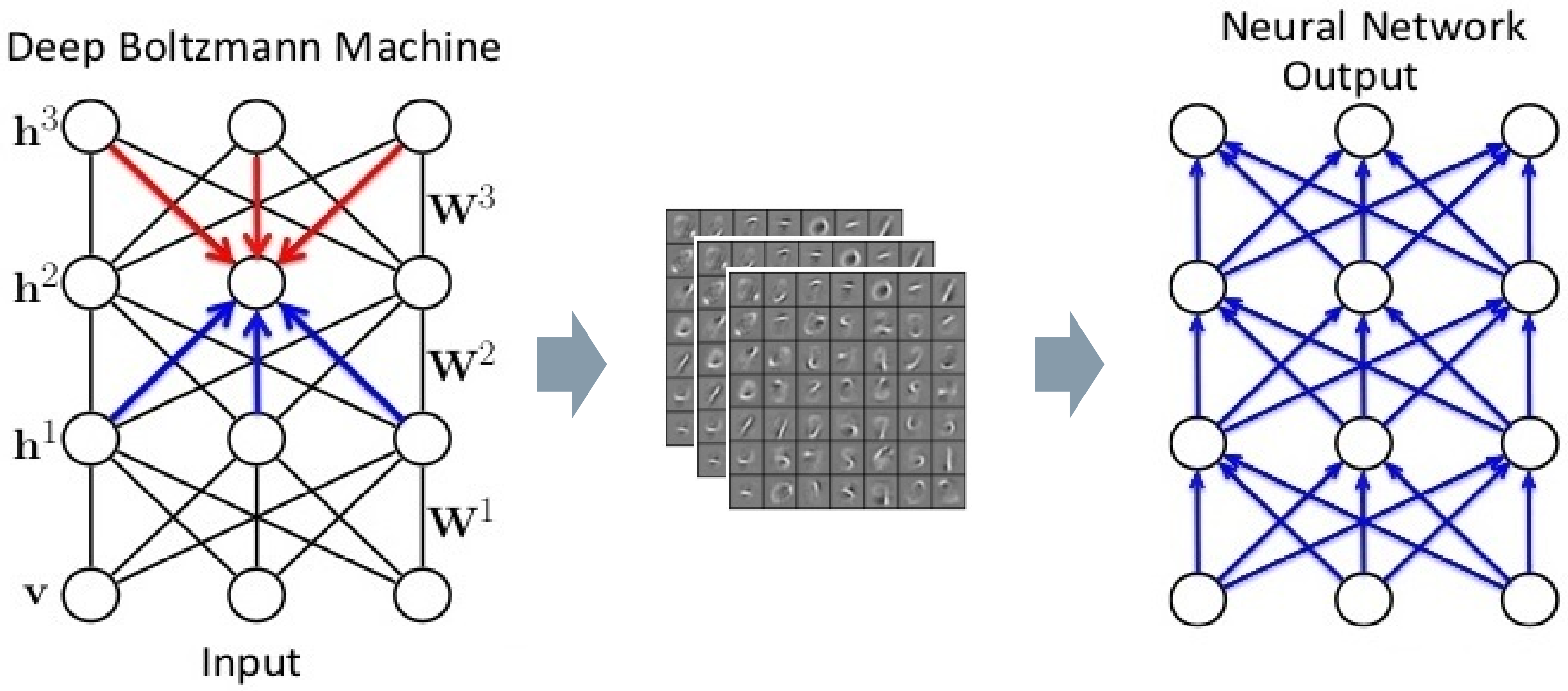
→ **W can be used to initialize neural networks**

→ latent vector as features into neural network

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

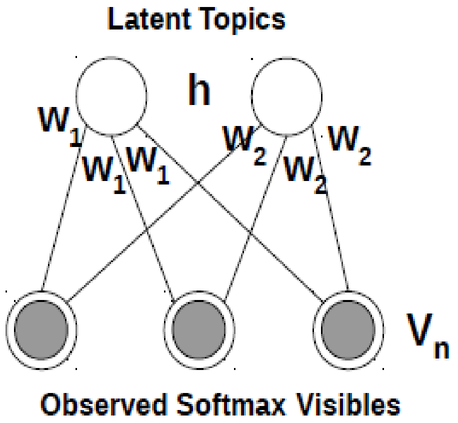
? **How to model count data?**
Example: Text document i.e., word counts.

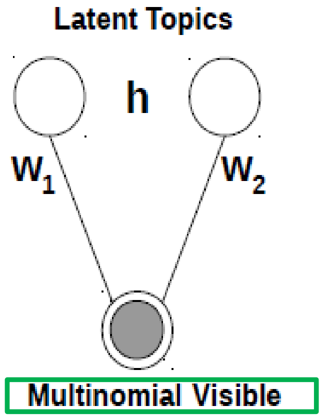
"a cat catches a mouse" $\mathbf{s} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}$

RBM variants: Overview of Replicated Softmax (RSM)

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

F: #hidden units,
K: vocabulary size





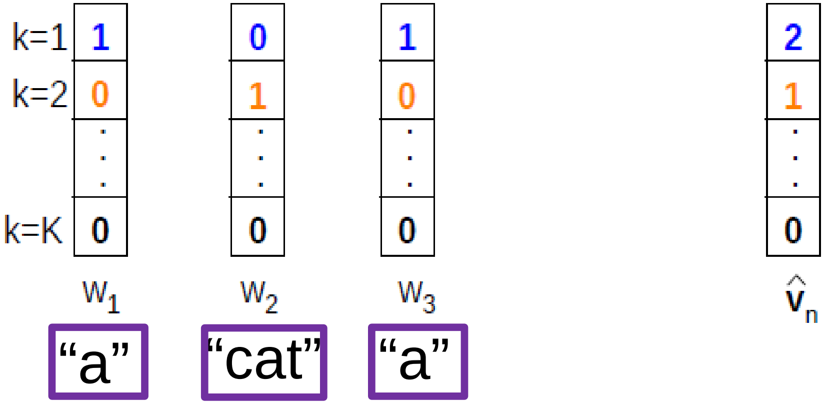
$$E(\mathbf{V},\mathbf{h}) = -\sum_{i=1}^{D}\sum_{j=1}^{F}\sum_{k=1}^{K} W_{ij}^{k} h_j v_i^k - \sum_{i=1}^{D}\sum_{k=1}^{K} v_i^k b_i^k - \sum_{j=1}^{F} h_j a_j$$

partition function (intractable)

$$P(\mathbf{V}) = \frac{1}{\mathcal{Z}}\sum_{\mathbf{h}} \exp\left(-E(\mathbf{V},\mathbf{h})\right), \quad \mathcal{Z} = \sum_{\mathbf{V}}\sum_{\mathbf{h}} \exp\left(-E(\mathbf{V},\mathbf{h})\right)$$

$$\boxed{p(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k\right)}{\sum_{q=1}^{K}\exp\left(b_i^q + \sum_{j=1}^{F} h_j W_{ij}^q\right)}}$$

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



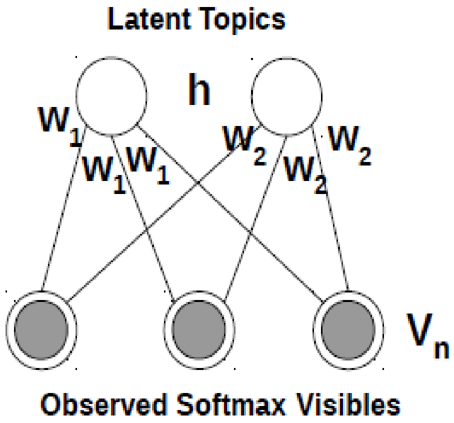
$$E(\mathbf{V},\mathbf{h}) = -\sum_{j=1}^{F}\sum_{k=1}^{K} W_j^k h_j \hat{v}^k - \sum_{k=1}^{K} \hat{v}^k b^k - D\sum_{j=1}^{F} h_j a_j$$

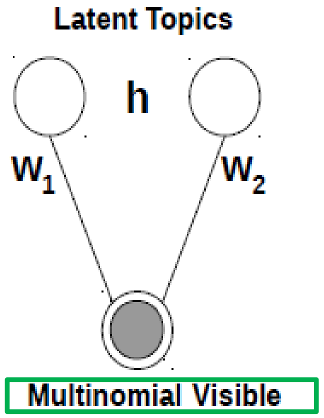
***Gupta** et al. 2018, Deep Temporal-Recurrent-Replicated-Softmax for Topical Trends over Time.*

RBM variants: Overview of Replicated Softmax (RSM)

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

F: #hidden units,
K: vocabulary size

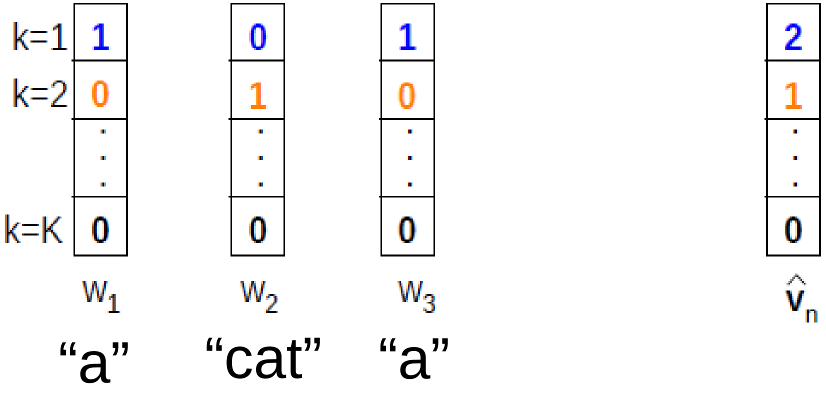




$$E(\mathbf{V},\mathbf{h}) = -\sum_{i=1}^{D}\sum_{j=1}^{F}\sum_{k=1}^{K} W_{ij}^{k} h_j v_i^k - \sum_{i=1}^{D}\sum_{k=1}^{K} v_i^k b_i^k - \sum_{j=1}^{F} h_j a_j$$

partition function (intractable)

$$P(\mathbf{V}) = \frac{1}{\mathcal{Z}}\sum_{\mathbf{h}} \exp\left(-E(\mathbf{V},\mathbf{h})\right), \quad \mathcal{Z} = \sum_{\mathbf{V}}\sum_{\mathbf{h}} \exp\left(-E(\mathbf{V},\mathbf{h})\right)$$



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

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

$$E(\mathbf{V},\mathbf{h}) = -\sum_{j=1}^{F}\sum_{k=1}^{K} W_j^k h_j \hat{v}^k - \sum_{k=1}^{K} \hat{v}^k b^k - D\sum_{j=1}^{F} h_j a_j$$

***Gupta** et al. 2018, Deep Temporal-Recurrent-Replicated-Softmax for Topical Trends over Time.*

Neural Autoregressive Distribution Estimator (NADE) models

Idea: Tractable log-likelihood, p(**v**)

LMU SIEMENS

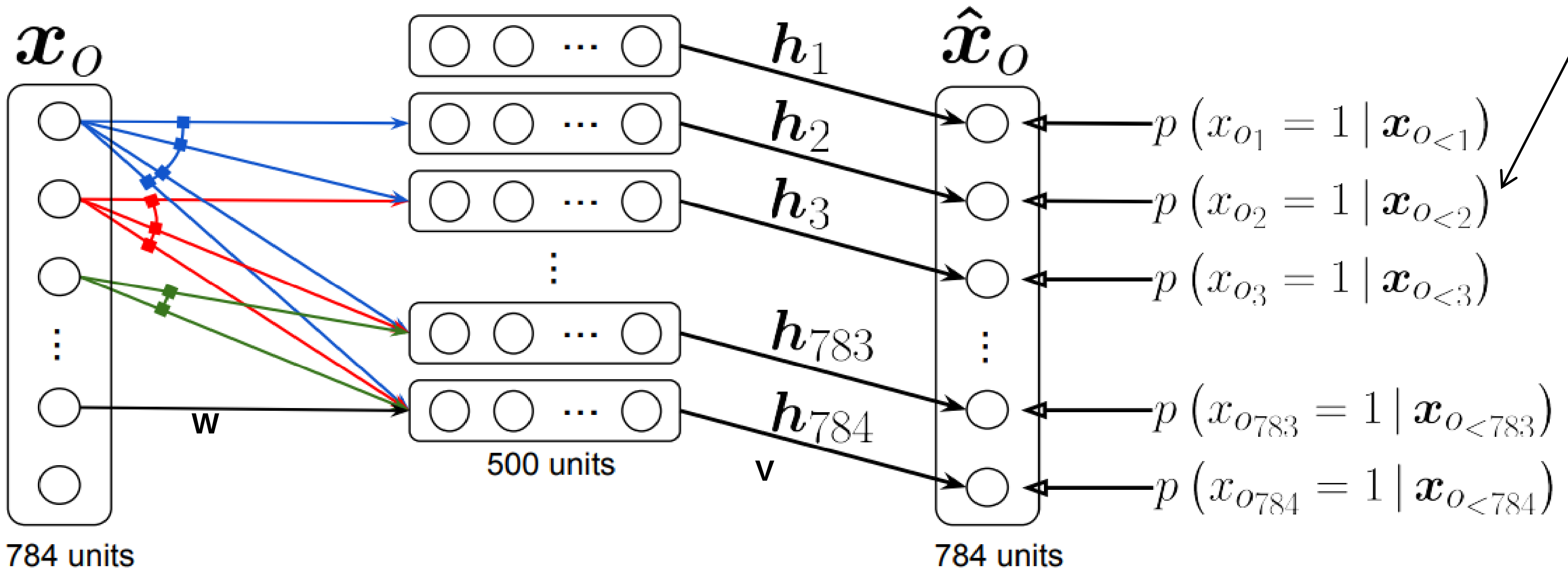
Neural Autoregressive Distribution Estimator (NADE)

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

given previous variables

in the ordering

Some ordering of **v**



binary

NADE is for binary data

Uria, Benigno, et al. "Neural Autoregressive Distribution Estimation“

Neural Autoregressive Distribution Estimator (NADE)

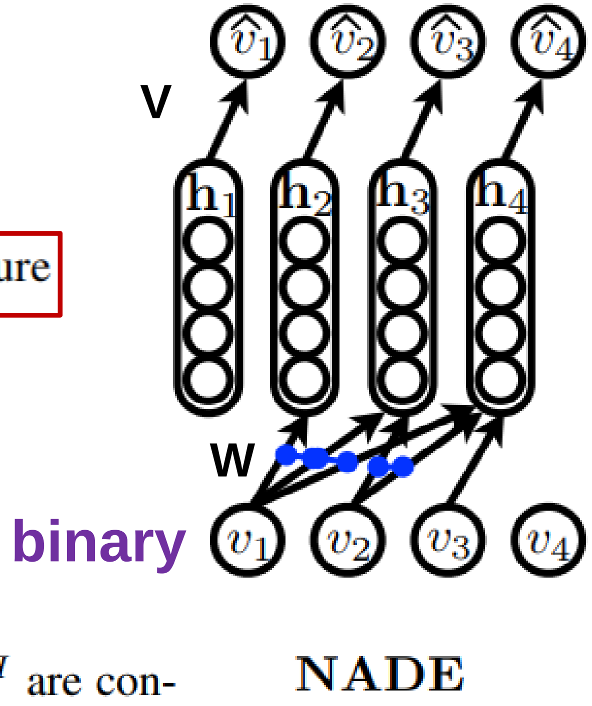
- **NADE:** Neural Autoregressive Distributional Estimator
- Inspired from RBM Family therefore, energy based
- sampling each dimension one after another

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

$$\mathbf{h}_i(\mathbf{v}_{<i}) = \text{sigm}\left(\mathbf{c} + \mathbf{W}_{:,<i}\mathbf{v}_{<i}\right)$$

NADE is for binary data, $p(v_i = 1|\mathbf{v}_{<i}) = \text{sigm}\left(b_i + \mathbf{V}_{i,:}\mathbf{h}_i(\mathbf{v}_{<i})\right)$

$$\mathbf{h}_{i+1}(\mathbf{v}_{<i+1}) = \text{sigm}(\mathbf{c} + \sum_{k<i+1} W_{:,v_k}) = \text{sigm}(\mathbf{W}_{:,v_i} + \mathbf{c} + \sum_{k<i} \mathbf{W}_{:,v_k})$$



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

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

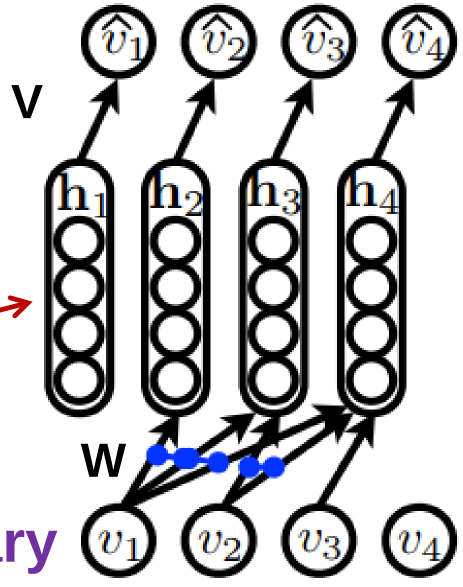
Advantages: Tractable **p(v)** → easier to train

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

$$\boxed{\mathbf{h}_i(\mathbf{v}_{<i}) = \mathrm{sigm}\left(\mathbf{c} + \mathbf{W}_{:,<i}\mathbf{v}_{<i}\right)}$$



NADE is for binary data, $p(v_i = 1|\mathbf{v}_{<i}) = \mathrm{sigm}\left(b_i + \mathbf{V}_{i,:}\mathbf{h}_i(\mathbf{v}_{<i})\right)$

$$\mathbf{h}_{i+1}(\mathbf{v}_{<i+1}) = \mathrm{sigm}(\mathbf{c} + \sum_{k<i+1} W_{:,v_k}) = \mathrm{sigm}(\mathbf{W}_{:,v_i} + \mathbf{c} + \sum_{k<i} \mathbf{W}_{:,v_k})$$

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

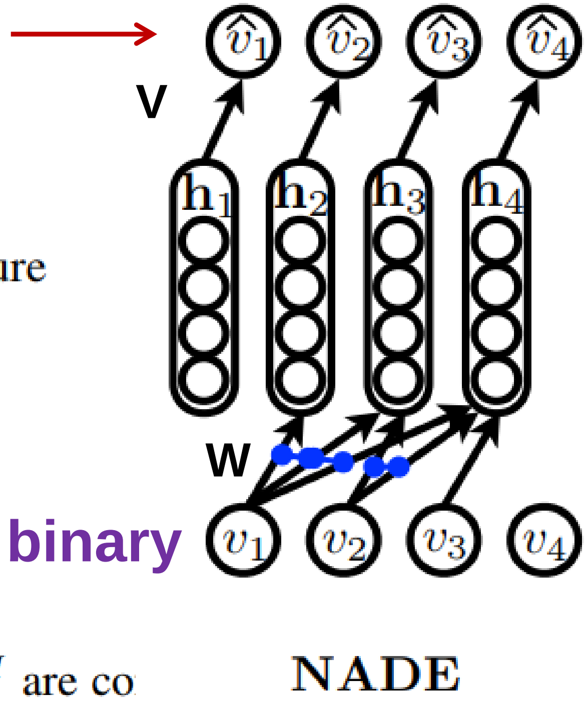
corresponds to a neural network with several parallel $\mathbf{h}_i(\mathbf{v}_{<i})$ hidden layers

Advantages: Tractable **p(v)** → 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



$$\mathbf{h}_i(\mathbf{v}_{<i}) = \mathrm{sigm}\left(\mathbf{c} + \mathbf{W}_{:,<i}\mathbf{v}_{<i}\right)$$

NADE is for binary data, $\boxed{p(v_i = 1|\mathbf{v}_{<i}) = \mathrm{sigm}\left(b_i + \mathbf{V}_{i,:}\mathbf{h}_i(\mathbf{v}_{<i})\right)}$

$$\mathbf{h}_{i+1}(\mathbf{v}_{<i+1}) = \mathrm{sigm}(\mathbf{c} + \sum_{k<i+1} W_{:,v_k}) = \mathrm{sigm}(\mathbf{W}_{:,v_i} + \mathbf{c} + \sum_{k<i} \mathbf{W}_{:,v_k})$$

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

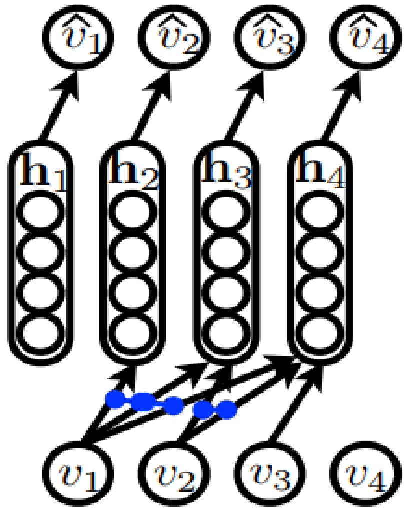
corresponds to a neural network with several parallel $\mathbf{h}_i(\mathbf{v}_{<i})$ hidden layers

Advantages: Tractable **p(v)** → easier to train

Neural Autoregressive Distribution Estimator (NADE)

Training NADE

$$p(v_i = 1|\mathbf{v}_{<i}) = \mathrm{sigm}\left(b_i + \mathbf{V}_{i,:}\mathbf{h}_i(\mathbf{v}_{<i})\right)$$



NADE

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

→ *cost*: maximize log-likelihood (logL)

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

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

where, D is the number of words in document, **v** and autoregressive conditional is given by:

$$p(v_i = 1|\mathbf{v}_{<i}) = \mathrm{sigm}\left(b_i + \mathbf{V}_{i,:}\mathbf{h}_i(\mathbf{v}_{<i})\right)$$

where, $\mathbf{h}_i(\mathbf{v}_{<i}) = \mathrm{sigm}\left(\mathbf{c} + \mathbf{W}_{:,<i}\mathbf{v}_{<i}\right)$

Neural Autoregressive Distribution Estimator (NADE)

Training NADE

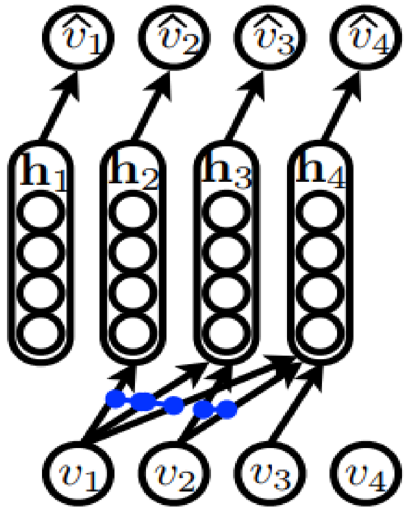
$$p(v_i = 1|\mathbf{v}_{<i}) = \mathrm{sigm}\left(b_i + \mathbf{V}_{i,:}\mathbf{h}_i(\mathbf{v}_{<i})\right)$$

→ Ground truth values of the pixels are used for conditioning

NADE: Alternative to RBMs with tractable p(v)

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

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



NADE

where, D is the number of words in document, **v** and

an autoregressive conditional is given by:

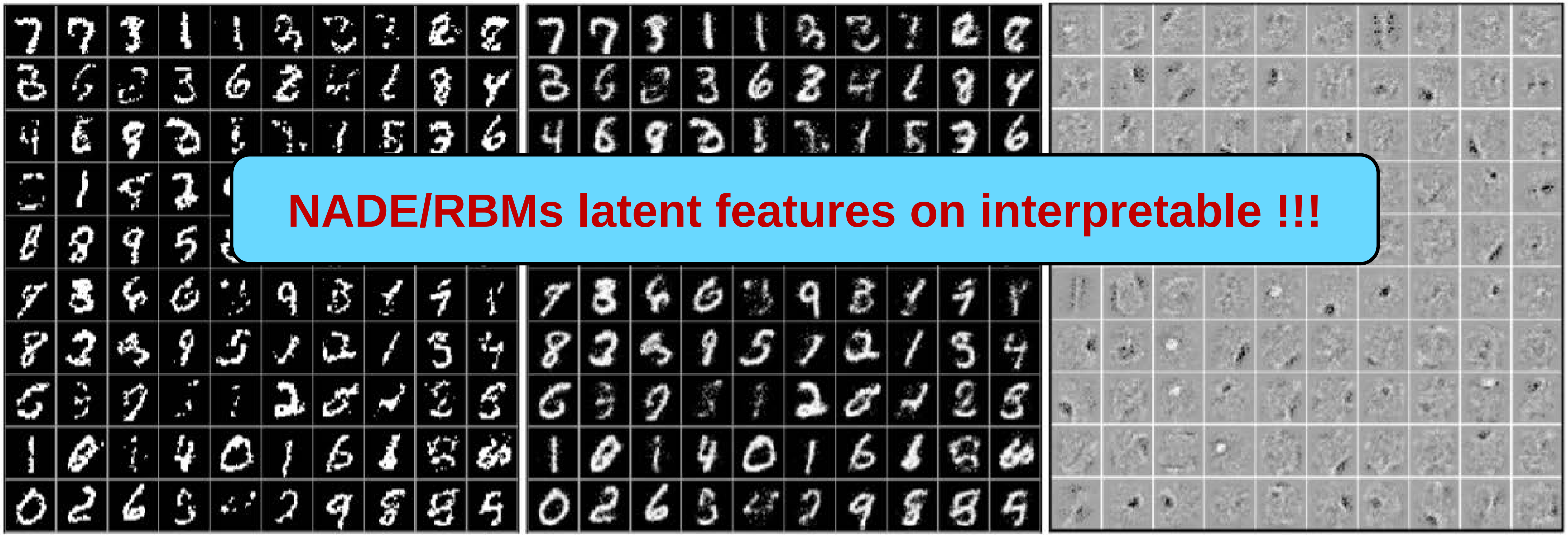
$$p(v_i = 1|\mathbf{v}_{<i}) = \mathrm{sigm}\left(b_i + \mathbf{V}_{i,:}\mathbf{h}_i(\mathbf{v}_{<i})\right)$$ where, $$\mathbf{h}_i(\mathbf{v}_{<i}) = \mathrm{sigm}\left(\mathbf{c} + \mathbf{W}_{:,<i}\mathbf{v}_{<i}\right)$$

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

https://ift6135h18.files.wordpress.com/2018/04/autoregressive_gen.pdf

Neural Autoregressive Distribution Estimator (NADE)



NADE/RBMs latent features on interpretable !!!

(**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

https://ift6135h18.files.wordpress.com/2018/04/autoregressive_gen.pdf

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" $\mathbf{s} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}$

Modeling documents in NADE → **DocNADE**

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" $\mathbf{s} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}$

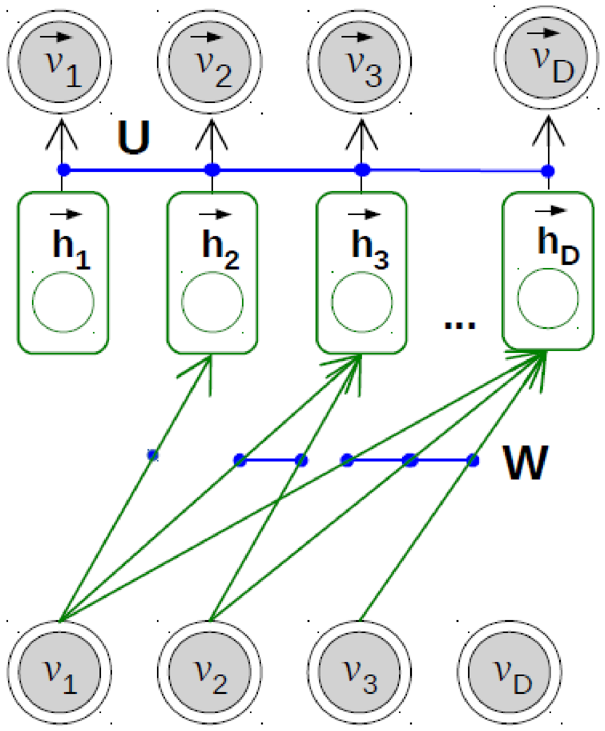
Modeling documents in NADE → **DocNADE**

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, **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

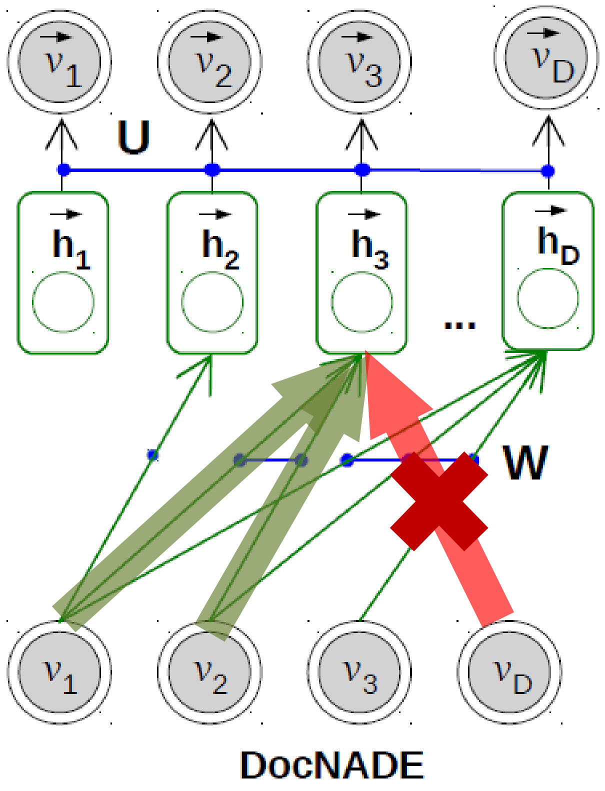


DocNADE

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, **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 $v_{<i}$**



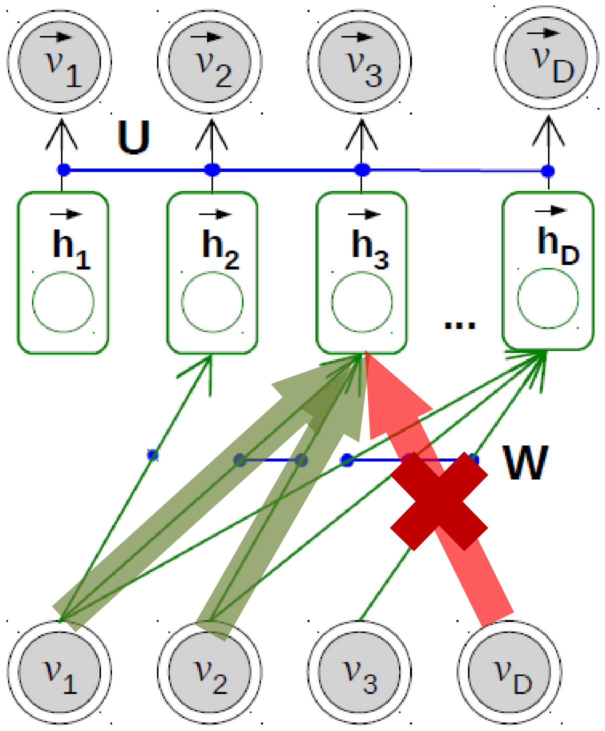
Limitations:

- ➢ does not take into account the following words $\mathbf{v}_{>i}$ 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, **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 $v_{<i}$**



DocNADE

Limitations:

- does not take into account the following words $\mathbf{v}_{>i}$ 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

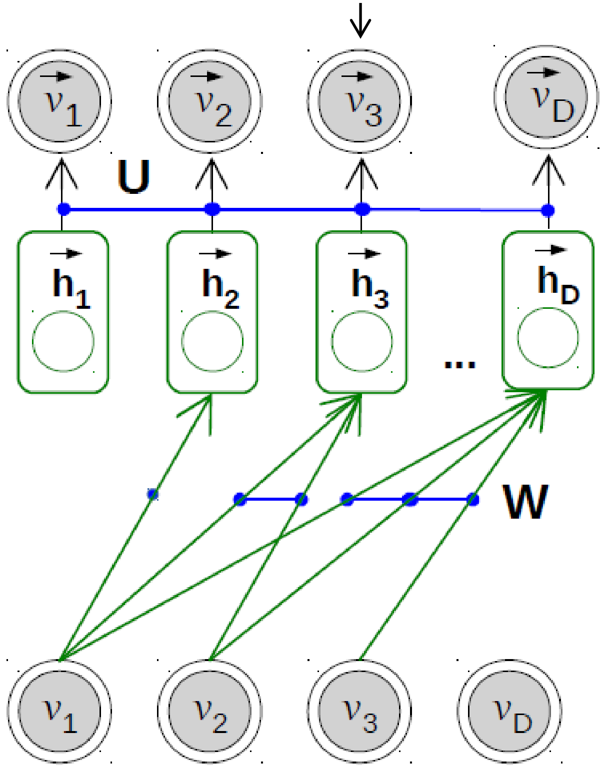
$$v_i \in \{1, ..., K\}$$

i.e., the index of the *ith* word in the dictionary of vocabulary size K

- a document **v** of size D is represented as, $\mathbf{v} = [v_1, ..., v_D]$
- joint distribution p(**v**) computed via each autoregressive conditionals,

$$p(\mathbf{v}) = \prod_{i=1}^{D} p(v_i|\mathbf{v}_{<i})$$

autoregressive conditional, $p(\mathbf{v}_{i=3} \mid \mathbf{v}_{<3})$



DocNADE

Neural Autoregressive Topic Model (DocNADE)

DocNADE Formulation

- inspired by RBM, RSM and NADE models
- models the joint distribution of all words v_i

$$v_i \in \{1, \ldots, K\}$$

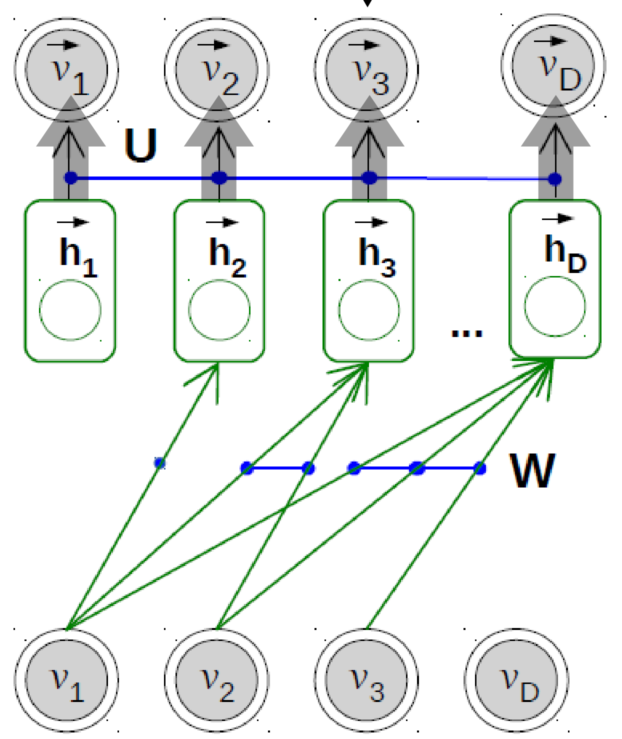
i.e., the index of the *ith* word in the dictionary of vocabulary size K

- a document **v** of size D is represented as, $\mathbf{v} = [v_1, \ldots, v_D]$
- joint distribution p(**v**) computed via each autoregressive conditionals,

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

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

autoregressive conditional, p($\mathbf{v}_{i=3}$ | $\mathbf{v}_{<3}$)



DocNADE

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, $\mathbf{v} = [v_1, \ldots, v_D]$
- joint distribution p(**v**) computed via each autoregressive conditionals,

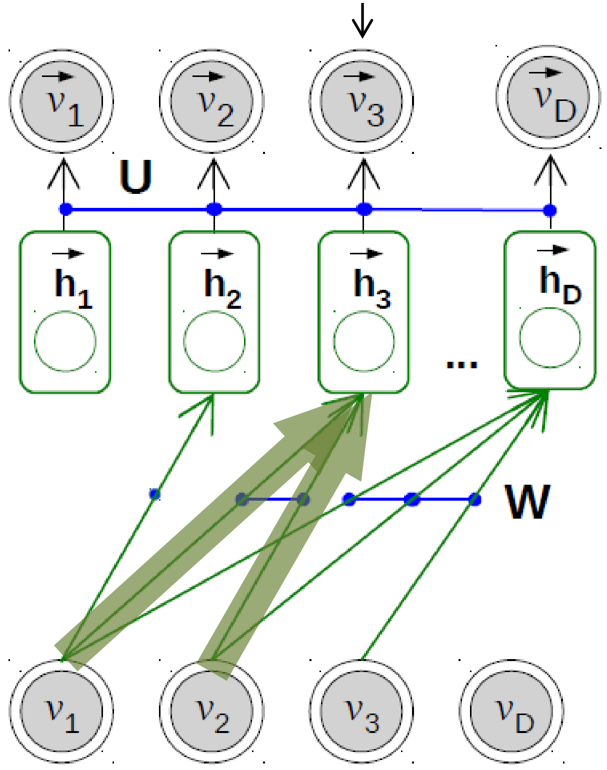
$$p(\mathbf{v}) = \prod_{i=1}^{D} p(v_i|\mathbf{v}_{<i}) = \prod_{i=1}^{D} \frac{\exp(b_w + \mathbf{U}_{w,:}\overrightarrow{\mathbf{h}}_i(\mathbf{v}_{<i}))}{\sum_{w'} \exp(b_{w'} + \mathbf{U}_{w',:}\overrightarrow{\mathbf{h}}_i(\mathbf{v}_{<i}))}$$

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

$$\overrightarrow{\mathbf{h}}_i(\mathbf{v}_{<i}) = g(\mathbf{c} + \sum_{k<i} \mathbf{W}_{:,v_k})$$

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

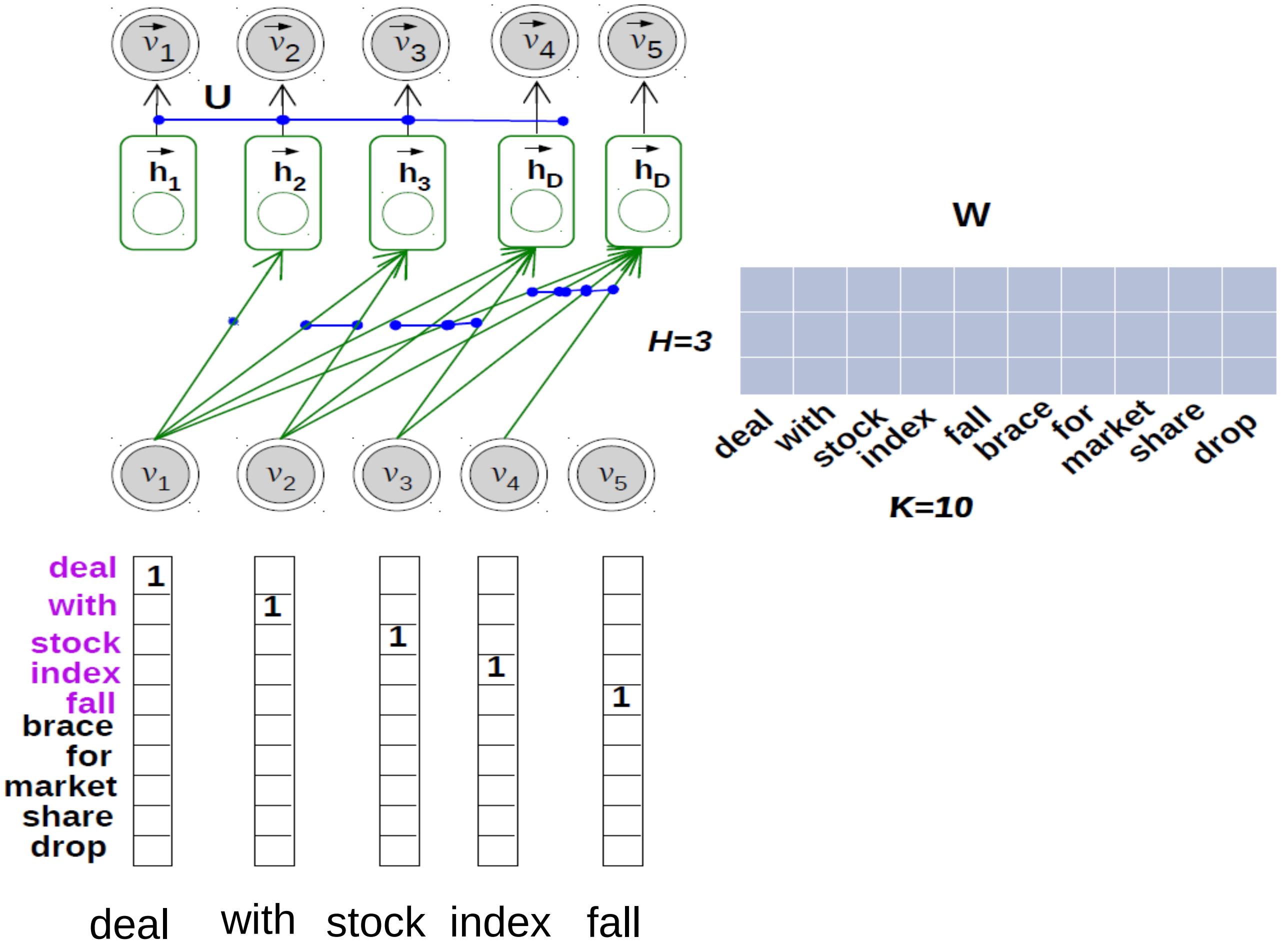
autoregressive conditional, p($\mathbf{v}_{i=3}$ | $\mathbf{v}_{<3}$)



DocNADE

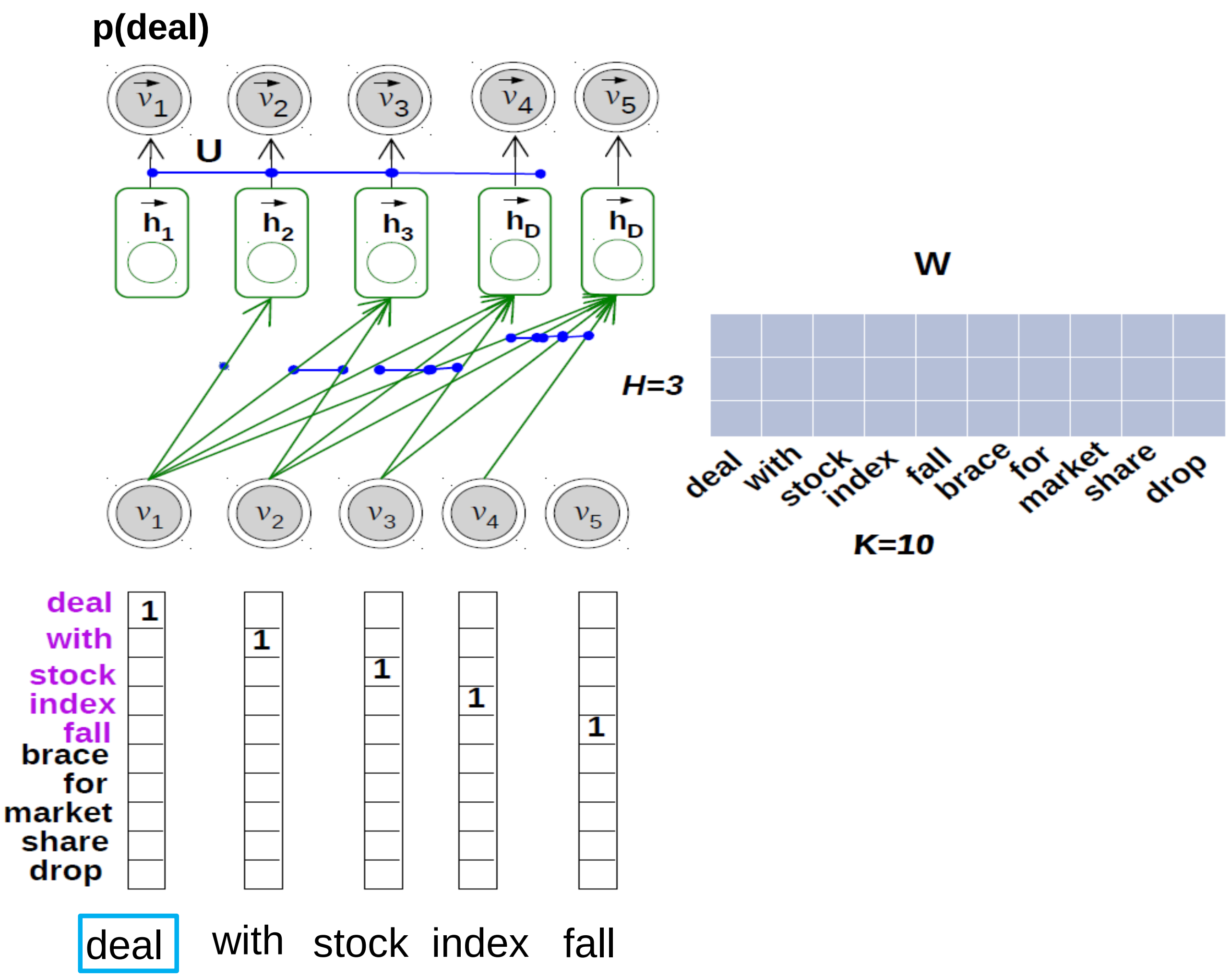
Neural Autoregressive Topic Model (DocNADE)

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



Neural Autoregressive Topic Model (DocNADE)

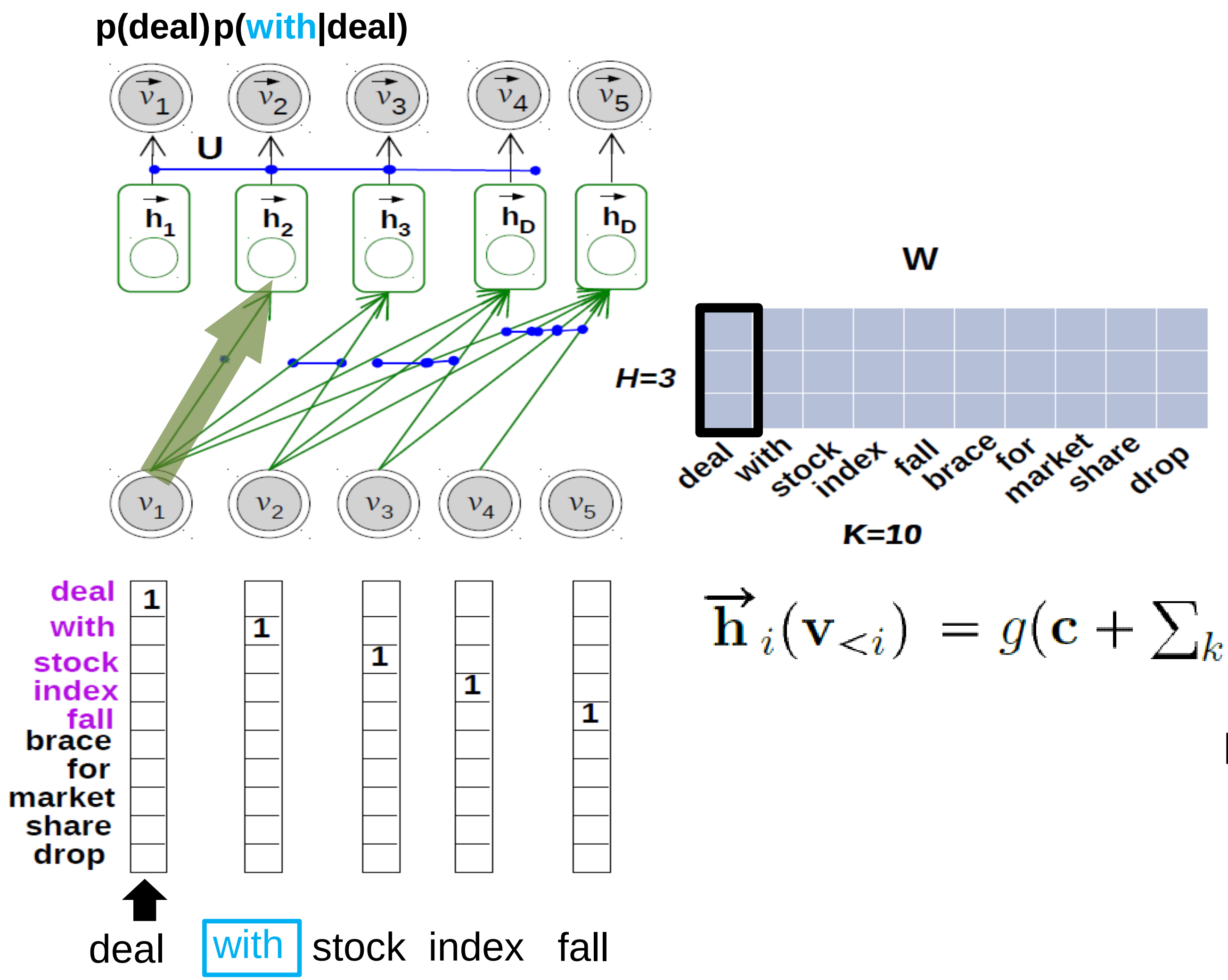
S1: *Deal with stock index fall*
S2: *Brace for market share drop*



Neural Autoregressive Topic Model (DocNADE)

S1: *Deal with stock index fall*

S2: *Brace for market share drop*



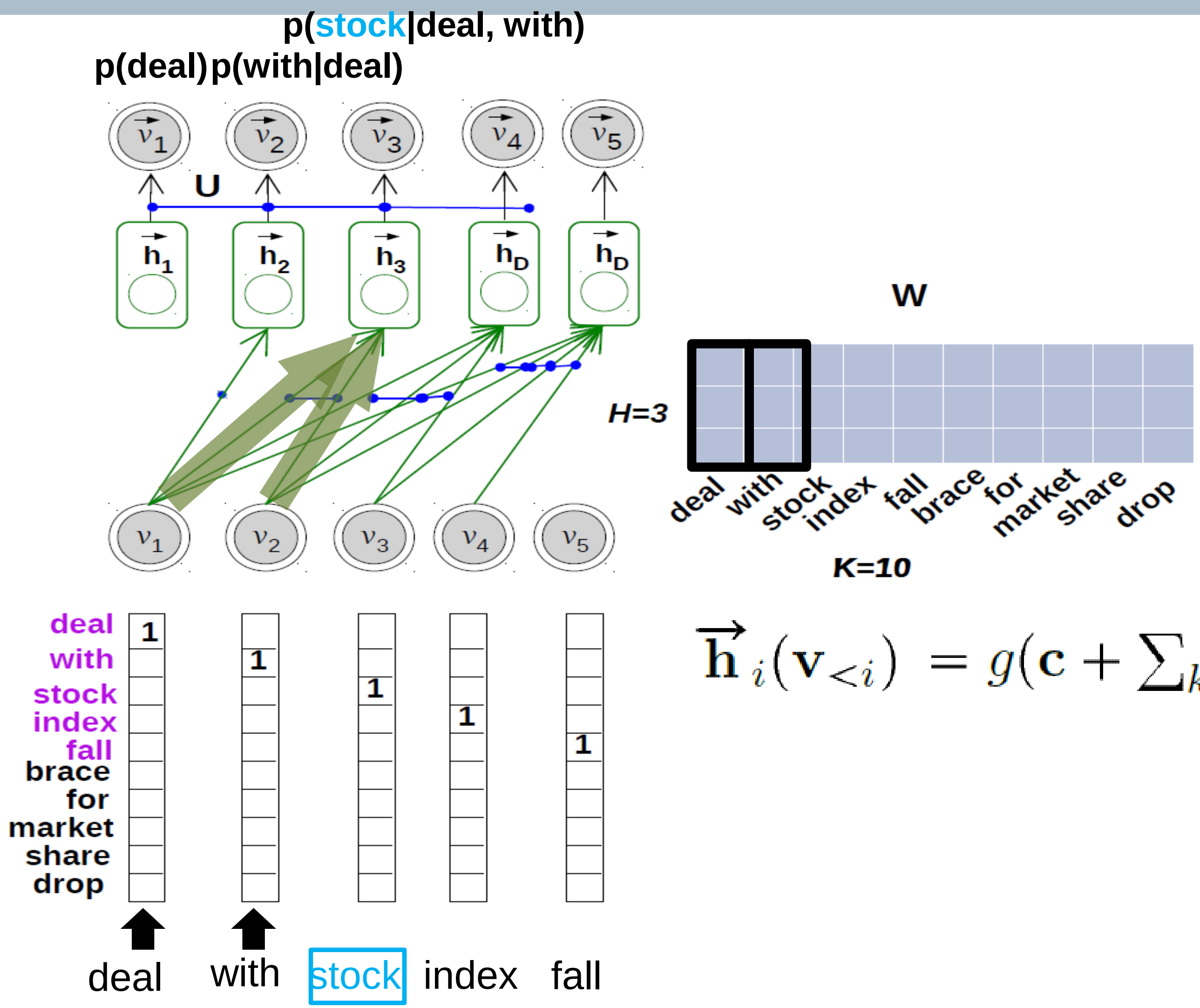
$$\vec{\mathbf{h}}_i(\mathbf{v}_{<i}) = g(\mathbf{c} + \textstyle\sum_{k<i} \mathbf{W}_{:,v_k})$$

Embedding aggregation

Neural Autoregressive Topic Model (DocNADE)

S1: *Deal with stock index fall*

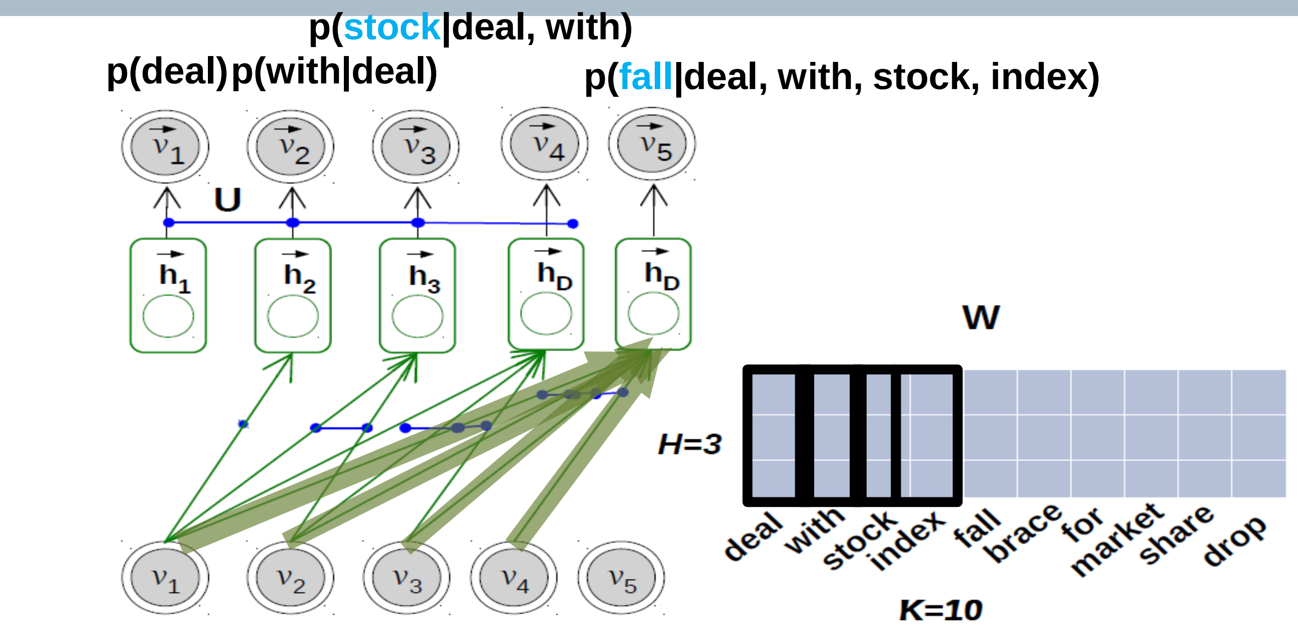
S2: *Brace for market share drop*

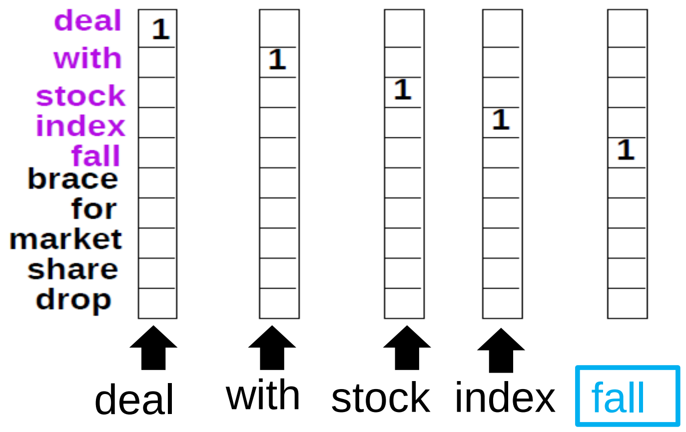


$$\vec{\mathbf{h}}_i(\mathbf{v}_{<i}) = g(\mathbf{c} + \textstyle\sum_{k<i} \mathbf{W}_{:,v_k})$$

Embedding aggregation

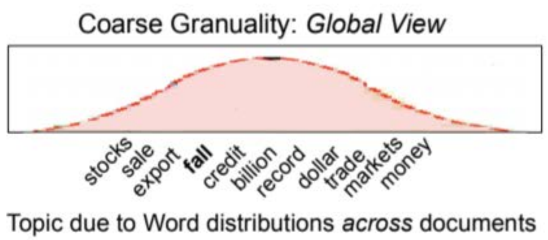
Neural Autoregressive Topic Model (DocNADE)

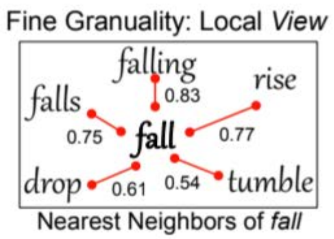




S1: *Deal with stock index fall*

S2: *Brace for market share drop*





$$\vec{\mathbf{h}}_i(\mathbf{v}_{<i}) = g(\mathbf{c} + \textstyle\sum_{k<i} \mathbf{W}_{:,v_k})$$

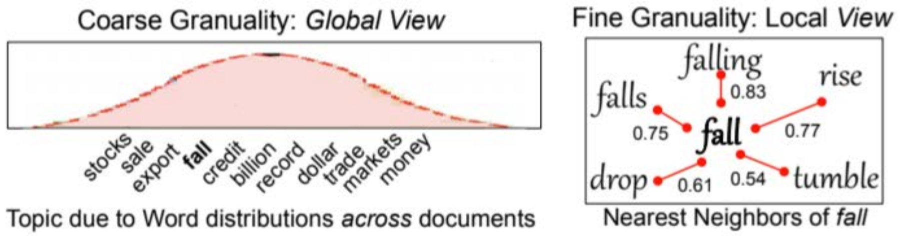
Embedding aggregation

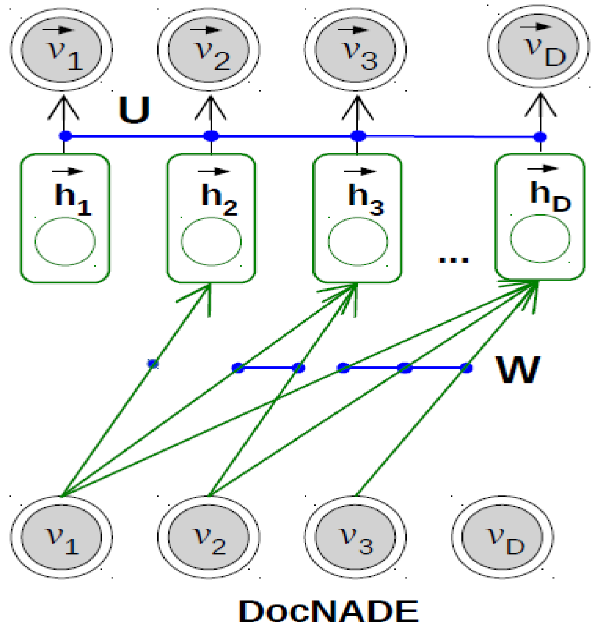
DocNADE Formulation

Properties of weight matrix, **W**

➢ each *column-vector* $\mathbf{W}_{:,vi}$

→ a vector for the word v_i





➢ each *row-vector* $\mathbf{W}_{j,:}$

→ a distribution over vocabulary of size K,

representing the ***jth*** 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 *ith* word in the dictionary of vocabulary size K

- a document $\mathbf{v}$ of size D is represented as, $\mathbf{v} = [v_1, \ldots, v_D]$
- joint distribution p(**v**) computed via each autoregressive conditionals,

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

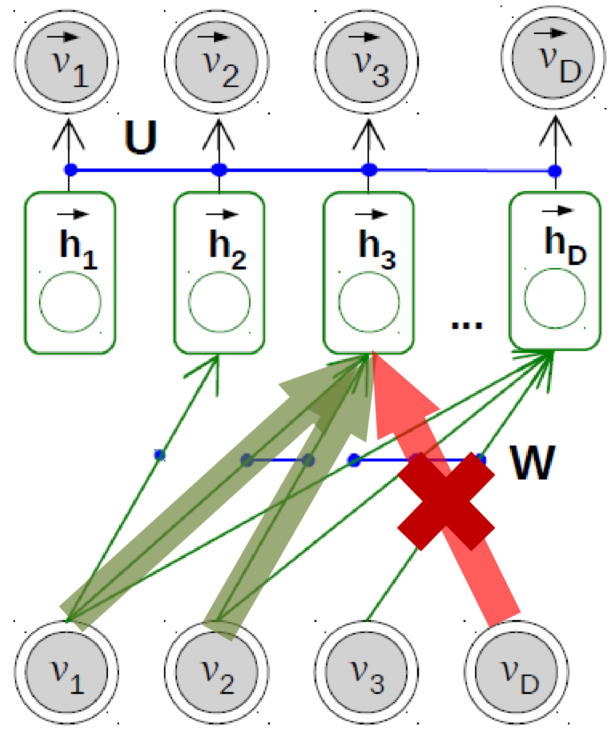
via a *feed-forward neural network*, where $\mathbf{v}_{<i} \in \{v_1, \ldots, v_{i-1}\}$

$$\overrightarrow{\mathbf{h}}_i(\mathbf{v}_{<i}) = g(\mathbf{c} + \textstyle\sum_{k<i} \mathbf{W}_{:,v_k})$$

Topic matrix

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

autoregressive conditional, p($\mathbf{v}_{i=3}$ | $\mathbf{v}_{<3}$)



DocNADE

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

Gupta et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior

DocNADE extensions………

Exploiting distributed word representations (word embeddings)

+

Full Context information

Checkout the backup slides, if interested !!!

Gupta et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior

Document Representation in iDocNADE variants

book			jesus			windows			gun		
neighbors	s_i	s_g	*neighbors*	s_i	s_g	*neighbors*	s_i	s_g	*neighbors*	s_i	s_g
books	.61	.84	christ	.86	.83	dos	.74	.34	guns	.72	.79
reference	.52	.51	god	.78	.63	files	.63	.36	firearms	.63	.63
published	.46	.74	christians	.74	.49	version	.59	.43	criminal	.63	.33
reading	.45	.54	faith	.71	.51	file	.59	.36	crime	.62	.42
author	.44	.77	bible	.71	.51	unix	.52	.47	police	.61	.43

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.

Tasks:
→Information retrieval
→ document representation
→ word representation
→ text classification
→ text clustering, etc.

Visualizing or interpreting filters i.e., W matrix. (column vectors →word embeddings)

Gupta et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior

Document Representation in iDocNADE variants

Checkout the backup slides!!!

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

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)

Gupta et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior

SIEMENS

Document Representation in iDocNADE variants

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)

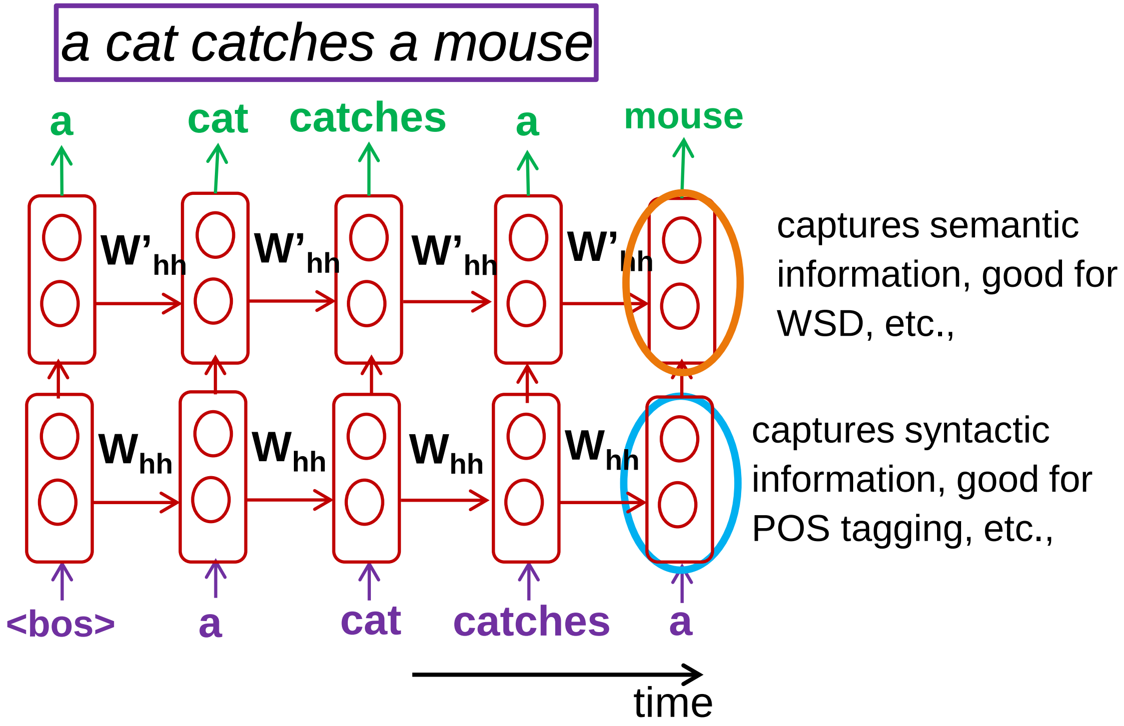
→ Introduce **language concepts** (e.g., word ordering, latent syntactic and semantic information) into DocNADE

Tasks:
→Information retrieval
→ document representation
→ word representation
→ text classification
→ text clustering, etc.

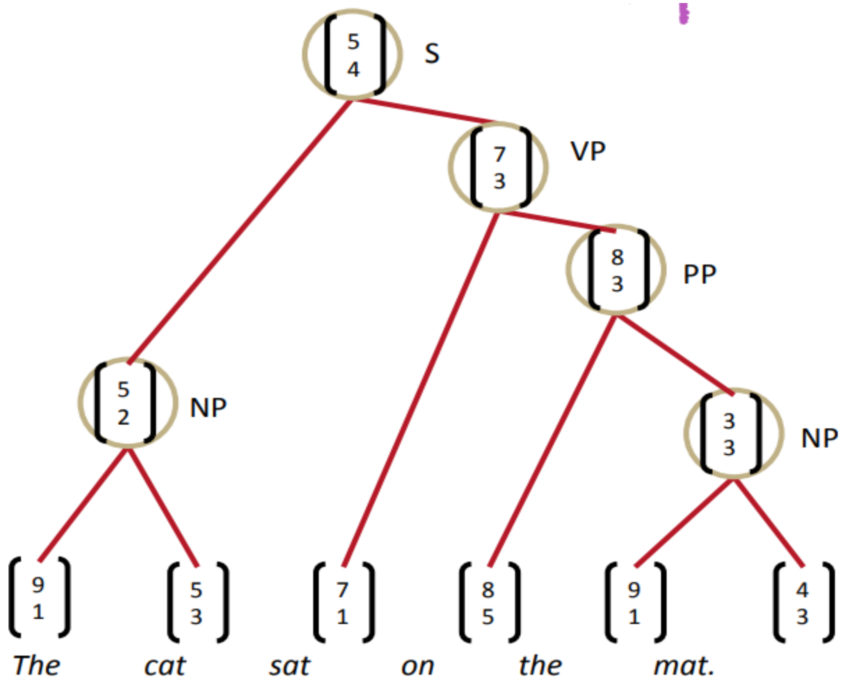
Further reading: https://arxiv.org/abs/1810.03947

Compositional Distributional Semantics

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



Generative Recurrent Neural Network (RNN)



Recursive Neural Network

$$L_t = -\log P(x_t | x_{t-1}, x_{t-2}, .. x_1)$$

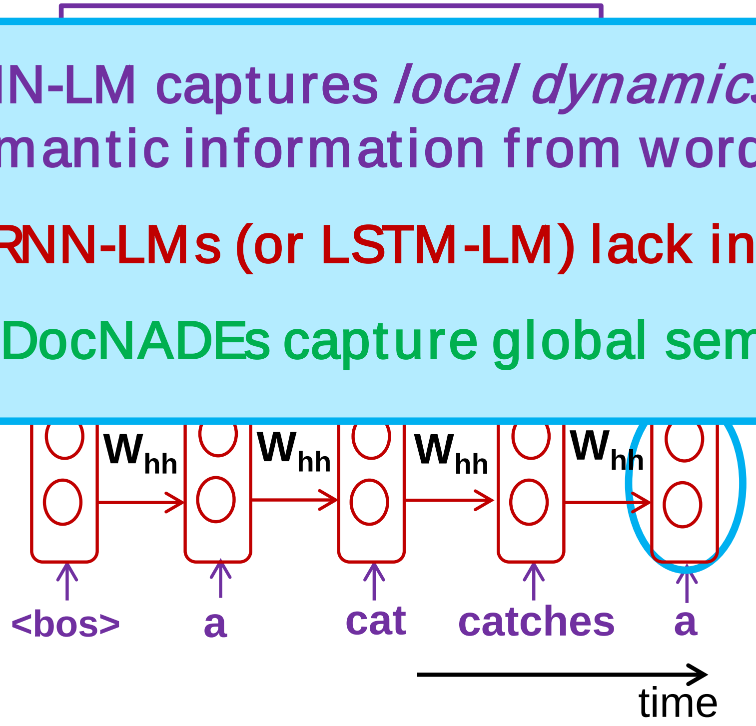
Compositional Distributional Semantics

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

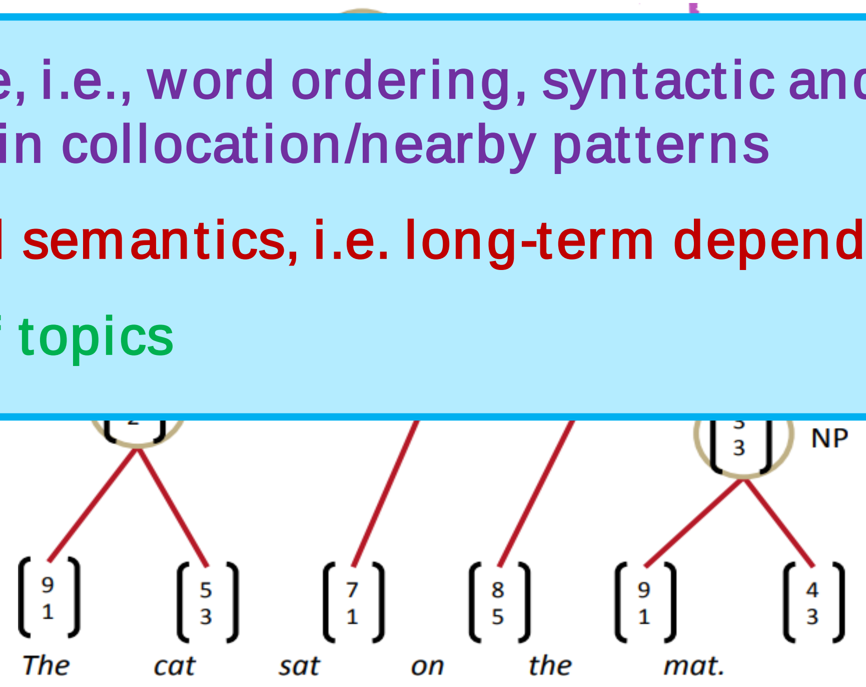
→ DocNADEs capture global semantics in form of topics



captures syntactic information, good for POS tagging, etc.,

Generative Recurrent Neural Network (RNN)

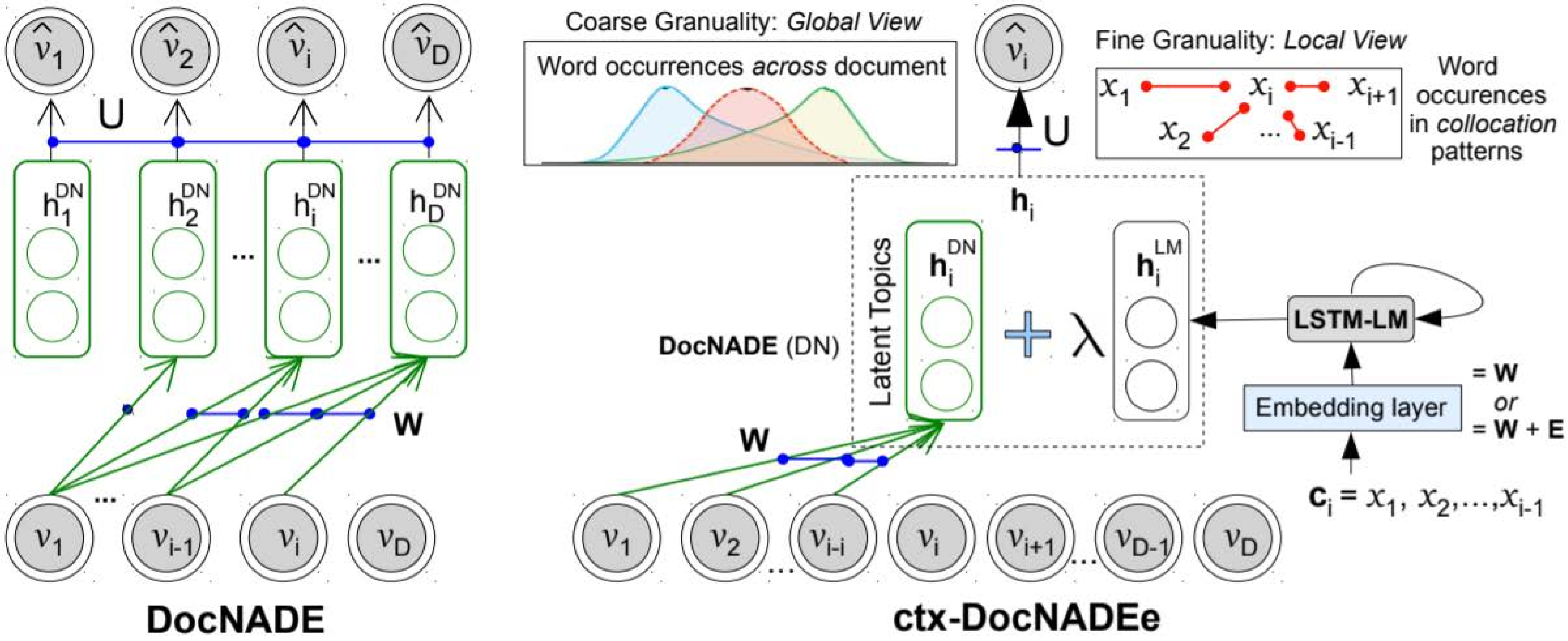
$$L_t = -\log P(x_t | x_{t-1}, x_{t-2}, .. x_1)$$



Recursive Neural Network

Distributional Representations: Local and Global Semantics

Combine or Joint training of DocNADE and LSTM-LM: textTOvec



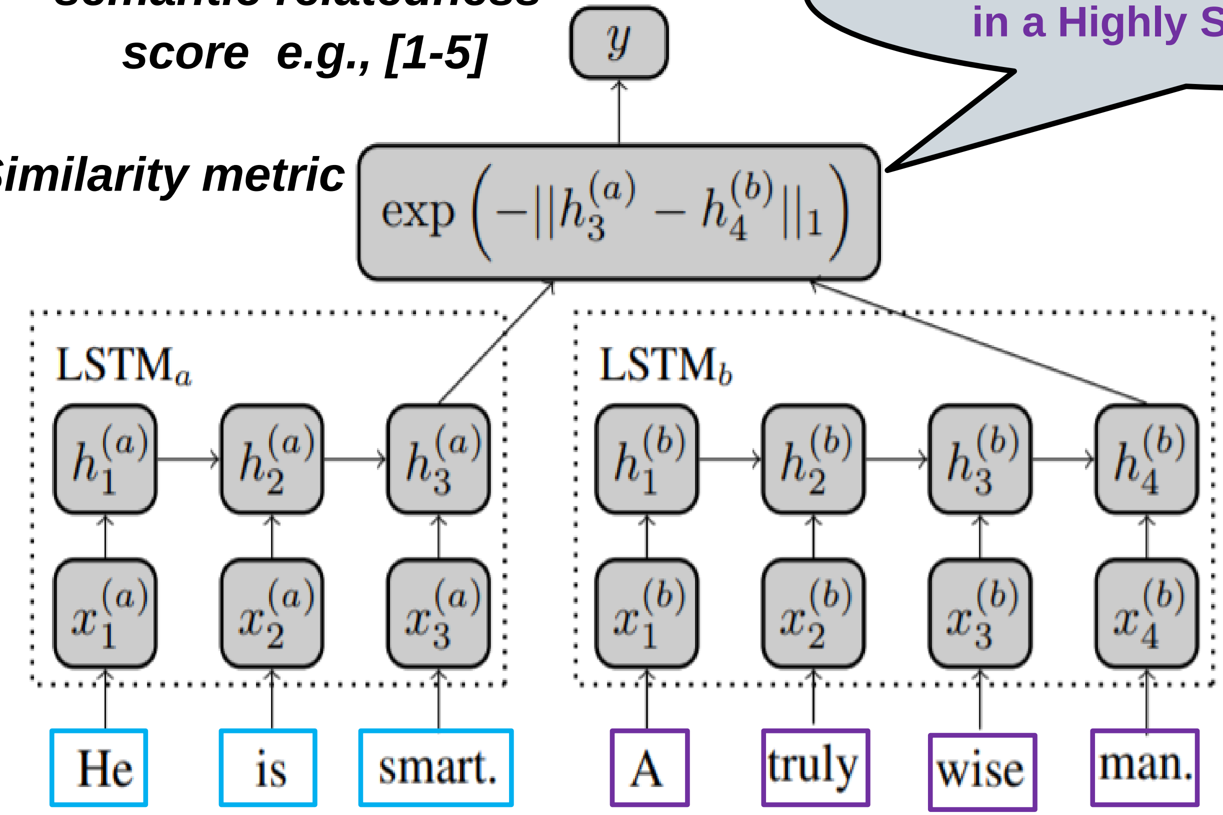
Further reading: **Gupta** et al, 2018. *textTOvec:* https://arxiv.org/abs/1810.03947

Metric Learning for Similarity (overview)

Learn *Text-Pair Representations* in a Highly Structured Space

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

Similarity metric



gold predicted

Sentence pair	G	M
A little girl is looking at a woman in costume. A young girl is looking at a woman in costume.	4.7	4.8
A person is performing tricks on a motorcycle. The performer is tricking a person on a motorcycle.	2.6	2.9
Someone is pouring ingredients into a pot. A man is removing vegetables from a pot.	2.4	2.5
Nobody is pouring ingredients into a pot. Someone is pouring ingredients into a pot.	3.5	3.7

***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/
- Generative models: https://ift6135h18.wordpress.com/author/aaroncourville/page/1/
- Boltzmann Machines: http://www.iro.umontreal.ca/~bengioy/ift6266/H12/html/contents.html#contents-en
- Boltzmann Machines: https://ift6266h15.wordpress.com/category/lectures/page/1/
- http://www.iro.umontreal.ca/~bengioy/ift6266/H16/representation-learning.pdf
- 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
- http://cse.iitkgp.ac.in/~sudeshna/courses/DL17/Autoencoder-15-Mar-17.pdf
- https://www.cl.cam.ac.uk/~pv273/slides/UCLSlides.pdf
- Autoregressive networks: https://ift6135h18.files.wordpress.com/2018/04/autoregressive_gen.pdf
- 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
- iDocNADEe: *Gupta* et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior
- ctx-DocANDEe (textTOvec): *Gupta* et al, 2018. textTOvec: https://arxiv.org/abs/1810.03947
- Siamese LSTM: *Gupta* et al, 2018. *Replicated Siamese LSTM in Ticketing System for Similarity Learning and Retrieval in Asymmetric Texts*

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

Backup slides, if interested !!!

Informed Document Autoregressive Topic Model with Word Embeddings

Motivation1: Contextual Information

Source Text		Sense/Topic
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"

Training Contexts (Preceding + Following)		Sense/Topic of "neurons"
Like biological brains, study of noisy + in artificial neural networks	→	"biological neural network"
Like biological brains, study of noisy + in artificial neural networks	→	"artificial neural network"

Context information around words helps in determining their actual meaning !!!

Gupta et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior

Informed Document Autoregressive Topic Model with Word Embeddings

SIEMENS

Motivation2: Distributional Semantics for Lack of Context

- **"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.,

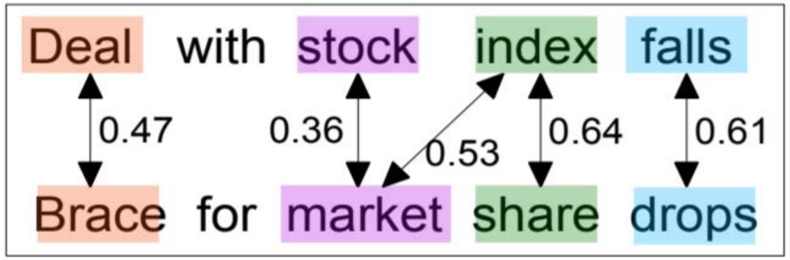
Topic1: price, ***wall***, ***china***, fall, shares — **incoherent**

Topic2: **shares, price, profits, rises, earnings** — **coherent**

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

No word overlap

(e.g., 1-hot-encoding)



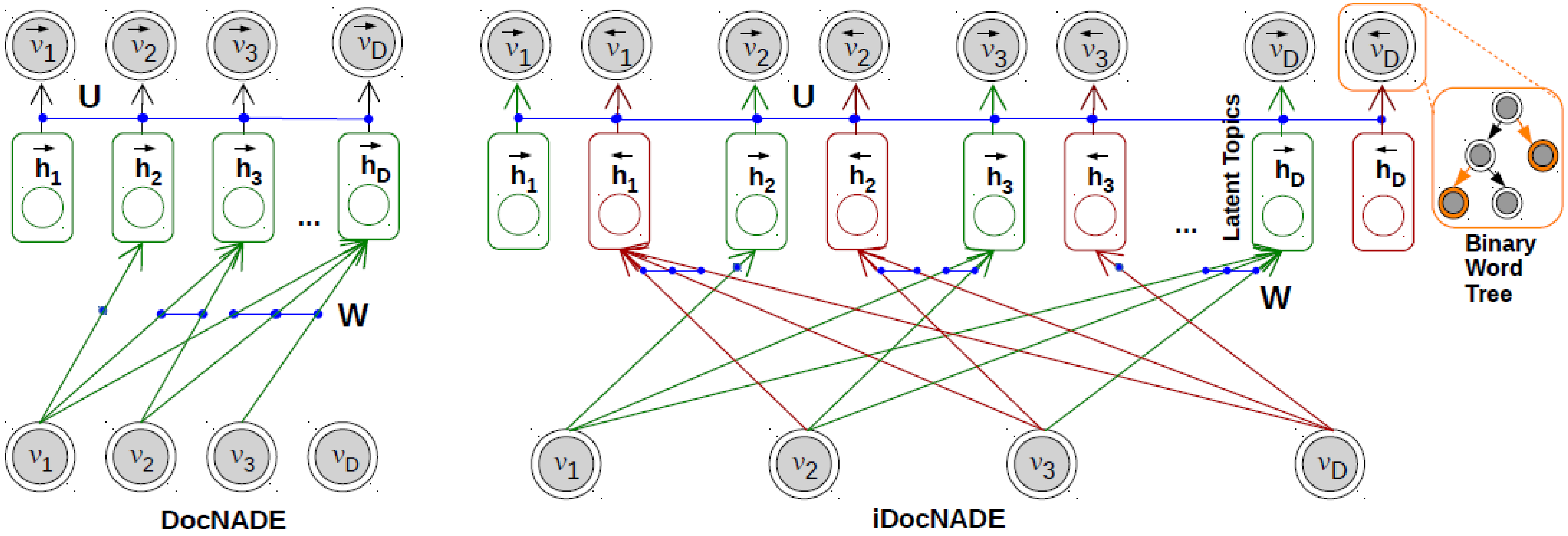
Cosine similarity in Word Embedding space

→ **trading**

→ **trading**

Same topic class

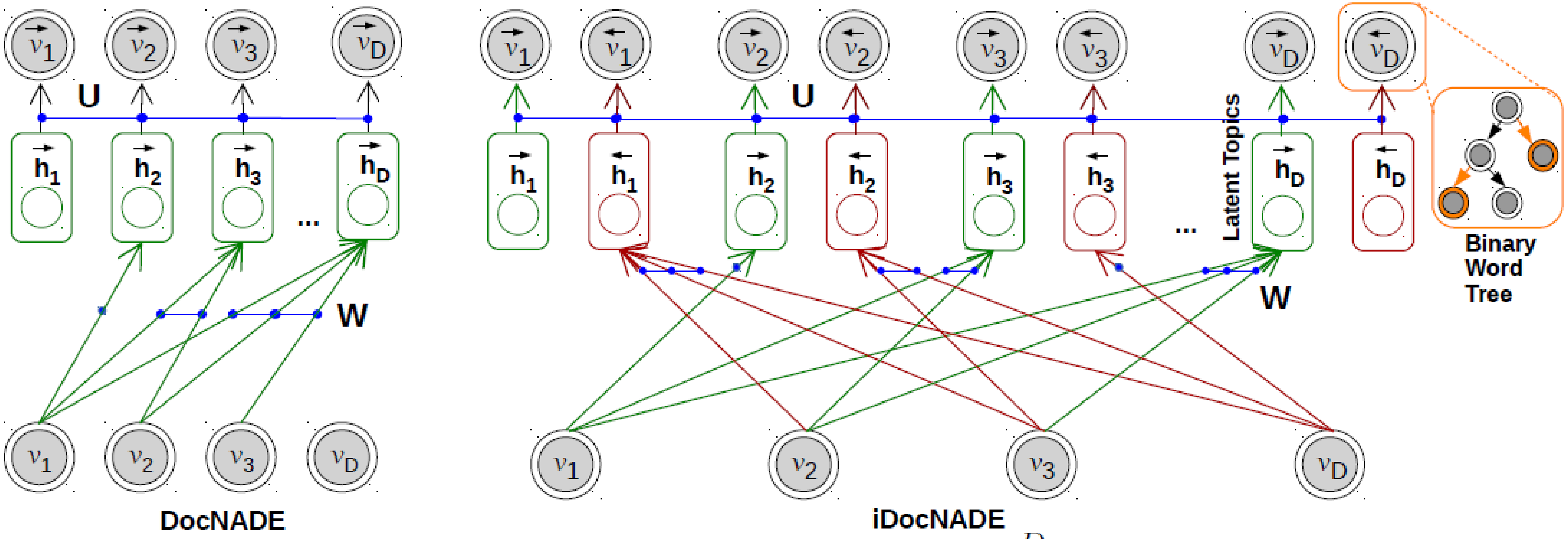
DocNADE variants: Contextualized DocNADE (iDocNADE)



- **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

Gupta et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior

DocNADE variants: Contextualized DocNADE (iDocNADE)



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

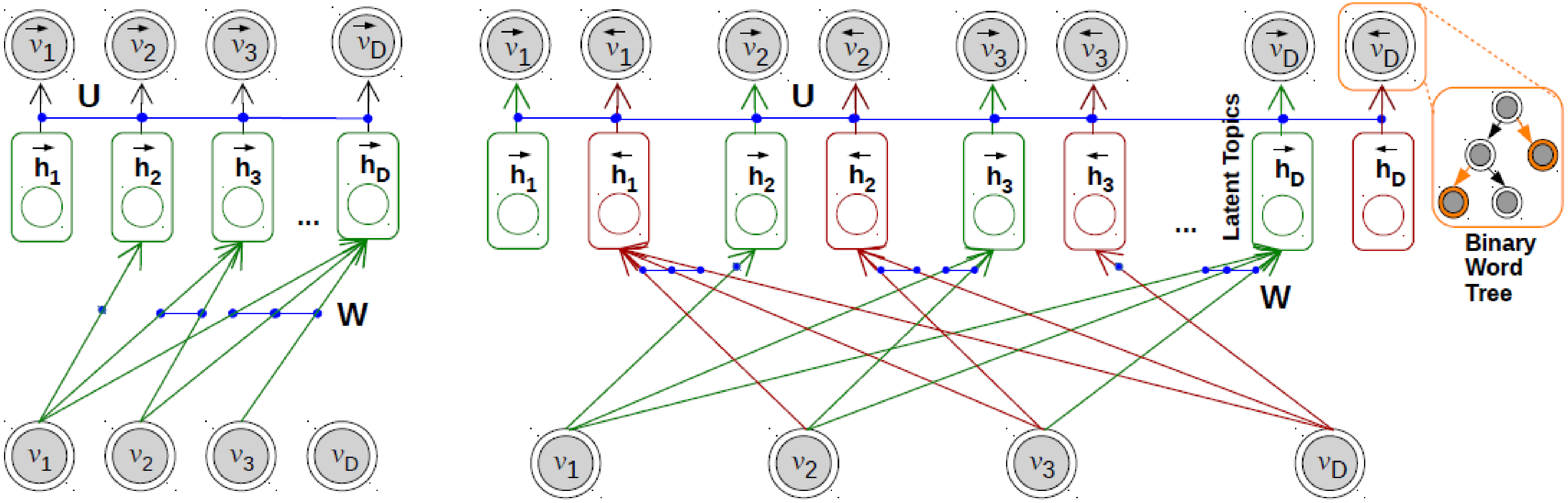
$$\overrightarrow{\mathbf{h}}_i(\mathbf{v}_{<i}) = g(\mathbf{c} + \textstyle\sum_{k<i} \mathbf{W}_{:,v_k})$$

$$\log p(\mathbf{v}) = \frac{1}{2}\sum_{i=1}^{D} \underbrace{\log p(v_i|\mathbf{v}_{<i})}_{\text{forward}} + \underbrace{\log p(v_i|\mathbf{v}_{>i})}_{\text{backward}}$$

$$\overrightarrow{\mathbf{h}}_i(\mathbf{v}_{<i}) = g(\overrightarrow{\mathbf{c}} + \sum_{k<i} \mathbf{W}_{:,v_k})$$

$$\overleftarrow{\mathbf{h}}_i(\mathbf{v}_{>i}) = g(\overleftarrow{\mathbf{c}} + \sum_{k>i} \mathbf{W}_{:,v_k})$$

DocNADE variants: Contextualized DocNADE (iDocNADE)



DocNADE iDocNADE

Incomplete Context around words in DocNADE

Need for

Complete Context around words in iDocNADE

$\mathcal{L}^{Do}$

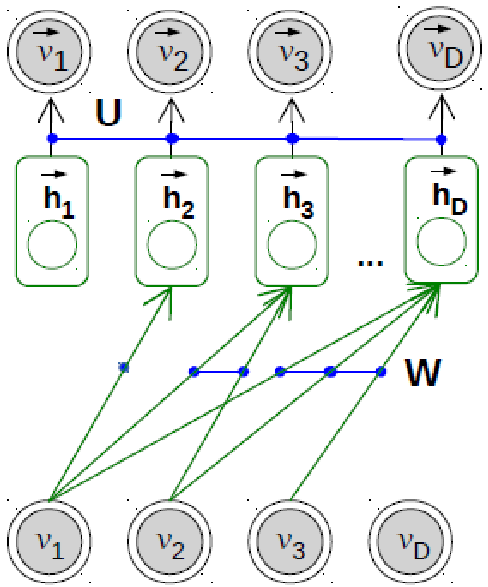
$+\underbrace{\log p(v_i|\mathbf{v}_{>i})}_{\text{backward}}$

$\overrightarrow{\mathbf{h}}_i(\mathbf{v}$

$_{:,v_k})$

$\overleftarrow{\mathbf{h}}_i(\mathbf{v}_{>i}) = g(\overleftarrow{\mathbf{c}} + \sum_{k>i} \mathbf{W}_{:,v_k})$

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

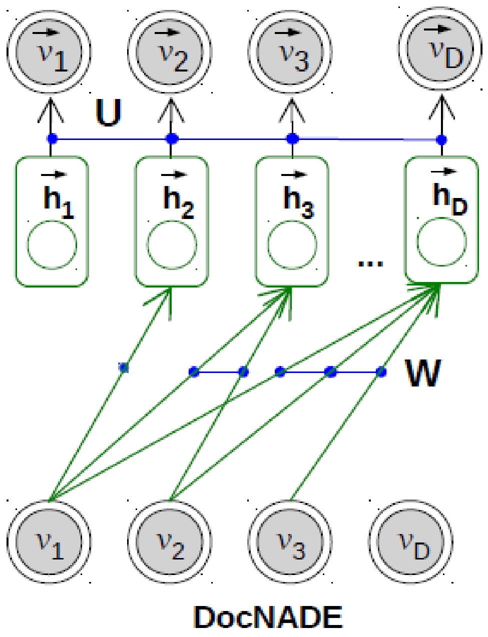


$$\mathcal{L}^{DocNADE}(\mathbf{v}) = \sum_{i=1}^{D} \log p(v_i|\mathbf{v}_{<i})$$

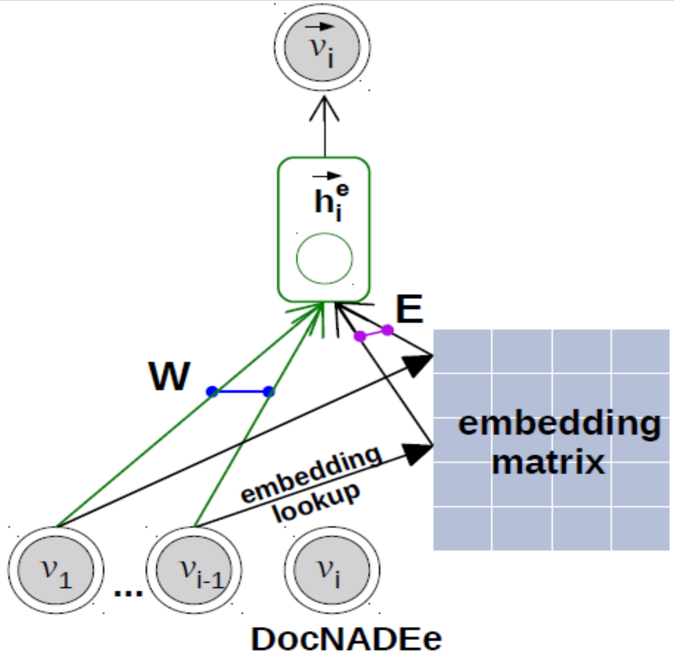
- $\overrightarrow{\mathbf{h}}_i(\mathbf{v}_{<i}) = g(\mathbf{c} + \sum_{k<i} \mathbf{W}_{:,v_k})$

Gupta et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior

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*)

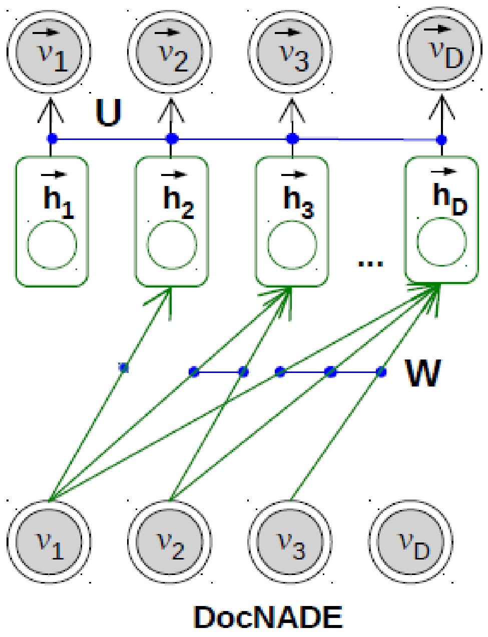


$$\mathcal{L}^{DocNADE}(\mathbf{v}) = \sum_{i=1}^{D} \log p(v_i|\mathbf{v}_{<i})$$

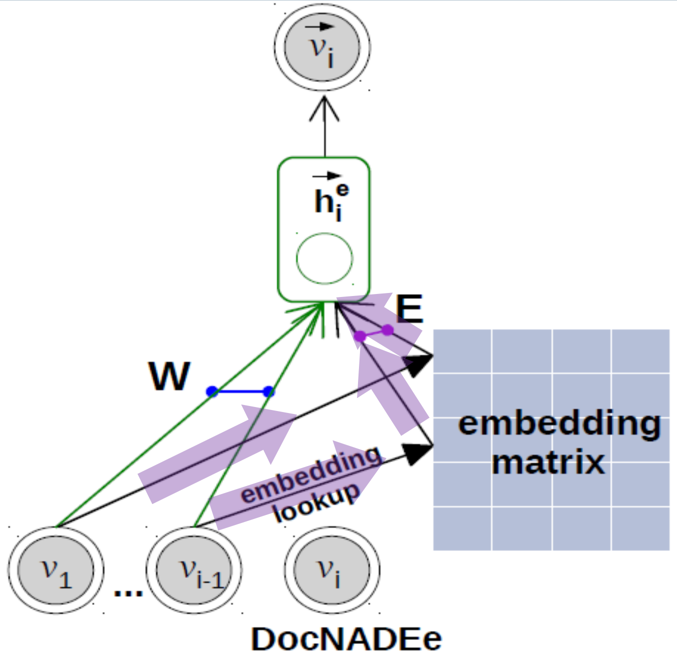
$$\overrightarrow{\mathbf{h}}_i(\mathbf{v}_{<i}) = g(\mathbf{c} + \textstyle\sum_{k<i} \mathbf{W}_{:,v_k})$$

Gupta et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior

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*)



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

$$\overrightarrow{\mathbf{h}}_i(\mathbf{v}_{<i}) = g(\mathbf{c} + \textstyle\sum_{k<i} \mathbf{W}_{:,v_k})$$

$$\overrightarrow{\mathbf{h}}^{e}_i(\mathbf{v}_{<i}) = g(\overrightarrow{\mathbf{c}} + \sum_{k<i} \mathbf{W}_{:,v_k} + \lambda \sum_{k<i} \mathbf{E}_{:,v_k})$$

Gupta et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior

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

$$\overrightarrow{\mathbf{h}}_i^{(d)}(\mathbf{v}_{<i}) = g(\overrightarrow{\mathbf{c}}^{(d)} + \mathbf{W}^{(d)} \cdot \overrightarrow{\mathbf{h}}_i^{(d-1)}(\mathbf{v}_{<i}))$$

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

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

DeepVairant1	DeepVairant2	DeepVairant3	DeepVairant4
DeepDNE	iDeepDNE	DeepDNEe	iDeepDNEe

Gupta et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior

Document Representation in iDocNADE variants

With the trained *iDocNADEe* (or *DocNADE* variants), the representation ($\overleftrightarrow{\mathfrak{h}}^e \in \mathbb{R}^H$) 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

$$\overrightarrow{\mathbf{h}}^e(\mathbf{v}^*) = g(\overrightarrow{\mathbf{c}} + \sum_{k \leqslant D^*} \mathbf{W}_{:,v_k^*} + \lambda \sum_{k \leqslant D^*} \mathbf{E}_{:,v_k^*})$$

$$\overleftarrow{\mathbf{h}}^e(\mathbf{v}^*) = g(\overleftarrow{\mathbf{c}} + \sum_{k \geqslant 1} \mathbf{W}_{:,v_k^*} + \lambda \sum_{k \geqslant 1} \mathbf{E}_{:,v_k^*})$$

Therefore; $\overleftrightarrow{\mathfrak{h}}^e = \overrightarrow{\mathbf{h}}^e(\mathbf{v}^*) + \overleftarrow{\mathbf{h}}^e(\mathbf{v}^*)$

Tasks:
→Information retrieval
→ document representation
→ word representation
→ text classification
→ text clustering, etc.

Gupta et al, 2018. Document Informed Neural Autoregressive Topic Models with Distributional Prior