April 2021

Geometric Representation Learning

Luke Vilnis
University of Massachusetts Amherst

GEOMETRIC REPRESENTATION LEARNING

A Dissertation Presented

by

LUKE VILNIS

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2021

College of Information and Computer Sciences

GEOMETRIC REPRESENTATION LEARNING

A Dissertation Presented

by

LUKE VILNIS

Approved as to style and content by:

_____

Andrew K. McCallum, Chair

_____

Daniel R. Sheldon, Member

_____

Mohit Iyyer, Member

_____

Luc Rey-Bellet, Member

_____

James Allan, Chair of the Faculty
College of Information and Computer Sciences

ACKNOWLEDGMENTS

I cannot imagine having done a PhD with any advisor besides Andrew McCallum. Andrew is a truly creative researcher, generous with his time, a wonderful liaison to the machine learning community, a prolific fund(ing)-raiser, and a passionate engineer. The lessons I have learned from Andrew have forever shaped the kind of researcher I hope to be. Above all, his kindness is what has carried me through graduate school. Andrew is a tremendously kind, conscientious, and caring person. From the year-to-year personal tribulations that plague most students, to this final strange denouement, Andrew has always been present with empathy and understanding. His grace and inclination towards service has been sustaining and inspiring.

I would not be where I am today without Emma Tosch and Alexandre Passos. Emma was the first UMass student I met, before I even knew I wanted to go here. Our friendship and collegial discussions opened me up to a world of computer science and cemented my determination to pursue graduate school. She introduced me to Alex Passos, who saw something in me that convinced Andrew to accept me as a Master's student.

Alex Passos was a tremendous role model and mentor, possessing a staggering breadth of knowledge of machine learning and software engineering, and very generous with his time. Even before I arrived on campus he was always available to answer questions on chat. Alex impressed on me his particular approach to machine learning: a broad, mile-high view of the field, with the confidence to drill down into the details of anything required. I have tried to carry on this tradition, along with his passion for absorbing the latest and greatest research and conveying it to everyone within earshot. I hope that I have been able to impart even one-tenth of this spirit to the next generation of IESL scientists.

The bulk of my PhD work has been done with the close collaboration of Xiang (Lorraine) Li. Lorraine has been a wonderful colleague and friend through our years exploring these strange models, a creative problem solver and strong engineer. Two other collaborators stand out. Dongxu Zhang is incredibly productive, with a knack for data science an an impressive ability to quickly absorb new research ideas. In the final stretch of my PhD I have enjoyed tremendously my collaboration with Michael Boratko, a welcome source of mathematical acumen, and a wellspring of good code, good ideas, positivity, and rigor.

Aside from thesis work, my (many) years of grad school have provided the opportunity to collaborate and learn from quite a few other colleagues who merit special recognition. David Belanger, for his diligence, intellect, and mentorship. He showed me how to write machine learning papers, to dive deep into a topic, to think rigorously and collaborate productively. Arvind Neelakantan, for collaboration and friendship through multiple internships, cities, and car rides back from the department. Tommy Boucher, for countless late night research discussions in the living room, and so much more. Rajarshi Das, for his humility and kindness, for keeping abreast of the latest research, fearlessly attacking massive problems, and never leaving the lab empty. Nicholas Monath, for his indefatigable work ethic and determination. Ari Kobren, for his friendship, positive attitude, creativity, and phenomenal core strength. Haw-Shiuan Chang is an especially creative researcher, and his willingness to go off the beaten path and take on new problems with new ideas is as inspiring as it is intimidating. Emma Strubell for being a fantastic collaborator through our grueling and ultimately successful early years, and a good friend throughout.

I have been blessed to have so many other wonderful colleagues, collaborators, and friends in IESL: Rico Angell, Trapit Bansal, Javier Burroni, Caitlin Cellier, Jinho Choi, Shib Dasgupta, Andrew Drozdov, Dan Duckworth, Jeffrey Flanigan, Craig Greenberg, Neha Nayak Kennard, Brian Martin, Shikhar Murty, Sheshera Mysore, Tim O'Gorman, Harshal Pandya, Subendhu Rongali, Pedram Rooshenas, Ben Roth, Sameer Singh, Dung Thai, Pat Verga, Mike Wick, and Nishant Yadav. Ian Gemp and Aaron Schein, while not

members of IESL, were subject to enough collateral research discussion and friendship to be right at home in this list.

I have been fortunate to have had several productive industry internships. Paul Mineiro and Nikos Karampatziakis at Microsoft gave me a treasure trove of linear algebra and optimization insight and intuition. George Dahl at Google shared with me his broad and deep knowledge of, and especially intuition for, deep learning. Others at Google are too many to list here, but I owe Samy Bengio, Andrew Dai, Ian Goodfellow, Lukasz Kaiser, Quoc Le, Mohamad Norouzi, Oriol Vinyals, Ilya Sutskever, and many others who were so generous with their time and their wisdom, and often their code.

I have cherished my time with friends and collaborators at other schools: Sam Bowman, Laurent Dinh, Rob Zinkov, Ben Poole, Dzimitry Bahdanau, Alex Beutel, and too many others to name, who made internships and conferences such a singular pleasure.

My time in grad school has been greatly enriched by the fantastic professors at UMass, both through the courses they taught, their service on my committee, and their generous open-door policies for discussing research. These include Justin Domke, Arjun Guha, Akshay Krishnamurthy, Erik Learned-Miller, Subhransu Maji, Brendan O'Connor, Luc Rey-Bellet, Dan Sheldon, Philip Thomas, and Hannah Wallach.

A special thanks is due to the administrative staff, without whom none of this would be possible. Leeanne Leclerc, without whom I probably would not have been admitted. Eileen Hamel and Malaika Ross, without whom I probably would have been kicked out for any number of administrative snafus. Thanks too to Pam Mandler, Kate Moruzzi, Glenn Stowell, and Lynn Yovina, who kept IESL and CIIR running smoothly all these years.

A huge thanks to my friends, including Billy, John, and Nick, without whom I never would have made it. My family has ever been a bedrock of support through graduate school, and really all the way up since birth, if you think about it. Thank you to my parents Helen and Charles for everything.

ABSTRACT

GEOMETRIC REPRESENTATION LEARNING

FEBRUARY 2021

LUKE VILNIS

B.Sc., DUKE UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew K. McCallum

Vector embedding models are a cornerstone of modern machine learning methods for knowledge representation and reasoning. These methods aim to turn semantic questions into geometric questions by learning representations of concepts and other domain objects in a lower-dimensional vector space. In that spirit, this work advocates for density- and region-based representation learning. Embedding domain elements as geometric objects beyond a single point enables us to naturally represent breadth and polysemy, make asymmetric comparisons, answer complex queries, and provides a strong inductive bias when labeled data is scarce. We present a model for word representation using Gaussian densities, enabling asymmetric entailment judgments between concepts, and a probabilistic model for weighted transitive relations and multivariate discrete data based on a lattice of axis-aligned hyperrectangle representations (boxes). We explore the suitability of these embedding methods in different regimes of sparsity, edge weight, correlation, and independence structure, as well as extensions of the representation and different optimization

strategies. We make a theoretical investigation of the representational power of the box lattice, and propose extensions to address shortcomings in modeling difficult distributions and graphs.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS . . . . . **iv**

ABSTRACT . . . . . **vii**

LIST OF TABLES . . . . . **xiv**

LIST OF FIGURES . . . . . **xv**

CHAPTER

1. INTRODUCTION **1**

1.1 Representation, Learning, and Inference 1
1.2 Geometric Representations 3

1.2.1 Geometric Representation Learning as Structured Prediction 6
1.2.2 Event Sets, Probability and Representation Learning 7
1.2.3 Overview 8

1.3 Summary of Contributions 10
1.4 Declaration of Collaborations 11

2. PRELIMINARY BACKGROUND **12**

2.1 Notation 12
2.2 Measures 12
2.3 Probability 18
2.4 Energy-Based Models 24
2.5 Probabilistic Energy-Based Models 26
2.6 Learning 27

2.6.1 Loss functions 27
2.6.2 Learning with stochastic gradient descent 28

2.7 Applications 31

2.7.1 Word Representations 31

2.8 Matrix Factorization 34

2.8.1 Knowledge Graphs 35
2.8.2 Entailment 36
2.8.3 Density Estimation 37
2.8.4 Open-Domain Graphs 38

3. GAUSSIAN WORD REPRESENTATIONS 40

3.1 Introduction 40
3.2 Related Work 41
3.3 Background 42
3.4 Warmup: Empirical Covariances 43
3.5 Energy-Based Learning of Gaussians 44

3.5.1 Symmetric Similarity: Expected Likelihood or Probability Product Kernel 44
3.5.2 Asymmetric Similarity: KL Divergence 48
3.5.3 Uncertainty of Inner Products 50
3.5.4 Learning 51

3.6 Evaluation 52

3.6.1 Specificity and Uncertainty of Embeddings 53
3.6.2 Entailment 54
3.6.3 Directly Learning Asymmetric Relationships 56
3.6.4 Word Similarity Benchmarks 57

3.7 Later Work on Gaussian Embeddings 58

3.7.1 Mixture Gaussian Embeddings 58
3.7.2 Supervised Gaussian Knowledge Graph Embedding 59
3.7.3 Unsupervised Gaussian Graph Embedding 60

3.8 Conclusion 60

4. LATTICES AND POSETS 62

4.1 Introduction 62

4.1.1 Partial Orders and Lattices 62
4.1.2 Order Embeddings (Vector Lattice) 63

4.2 Probabilistic Order Embeddings 67
4.3 Hierarchical Density Order Embeddings 69

4.4 Hyperbolic Embeddings 70
4.4.1 Poincaré embeddings 71
4.4.2 Hyperbolic entailment cones 72
4.5 Conclusion 73
5. BOX EMBEDDINGS 75
5.1 Introduction 75
5.2 Related Work 77
5.3 Method 78
5.3.1 Correlations from Cone Measures 78
5.3.2 Box Lattices 80
5.3.3 Learning 85
5.4 Experiments 87
5.4.1 Warmup: 2D Embedding of a Toy Lattice 87
5.4.2 WordNet 90
5.4.3 Flickr Entailment Graph 92
5.5 Queries with Negated Variables 93
5.6 Conclusion 95
6. SMOOTHED LEARNING FOR BOX EMBEDDINGS 96
6.1 Introduction 96
6.2 Related Work 96
6.3 Method 97
6.3.1 Motivation: Optimization and Sparse Data 97
6.3.2 Relaxed Geometry 98
6.4 Experiments 103
6.4.1 WordNet 103
6.4.2 Imbalanced WordNet 104
6.4.3 Flickr 105
6.4.4 MovieLens 105
6.4.4.1 Initialization Sensitivity 107
6.5 Proof of Gaussian Overlap Formula 108
6.6 Conclusion and Future Work 109

7. REPRESENTATIONAL POWER OF BOX EMBEDDINGS 111

7.1 Properties of the Box Lattice 111

7.1.1 Non-Distributivity 111
7.1.2 Pseudocomplemented 112

7.2 Predicting Graphs with Box Embeddings 112

7.2.1 Box Embedding 113
7.2.2 KL Divergences and Gaussian Embeddings 114
7.2.3 Order Embeddings 115

7.3 Limitations on Representation 116

7.3.1 A Counterexample 117

7.4 Unions of Boxes 119
7.5 Additive Box Model 120

7.5.1 Universal Approximation 121

7.6 Experiments 124

7.6.1 Google Syntactic N-Grams 124

7.7 Conclusion 126

8. CONCLUSION 128

8.1 Contributions 128
8.2 Future Work 129

8.2.1 Training Methods for Box Embeddings 129
8.2.2 Deep Boxes 130
8.2.3 Flexible Representations 130
8.2.4 Inference in Large Joint Distributions 131
8.2.5 Categorical Data 131
8.2.6 Multirelational Data 132

APPENDICES

A. EXPERIMENTAL DETAILS 133

A.1 Gaussian Experiments 133
A.2 Box Experiments 133

A.2.1 WordNet Parameters 133
A.2.2 Flickr Parameters 134

A.3 Smoothed Learning Experiments 134

A.3.1 WordNet Parameters 134
A.3.2 Imbalanced WordNet Parameters 135
A.3.3 Flickr Parameters 135
A.3.4 MovieLens Parameters 136

A.4 Additive Box Experiments 136

A.4.1 Google Syntactic N-Gram Parameters 136

BIBLIOGRAPHY **138**

LIST OF TABLES

Table **Page**

2.1 Notation 13

5.3 Multi-way queries: conditional probabilities adjust when adding additional evidence or contradiction. In contrast, POE can only raise or preserve probability when conditioning. 90

5.4 Negatively correlated variables produced by the model. 90

5.5 Classification accuracy on WordNet test set. 90

5.6 KL and Pearson correlation between model and gold probability. 92

5.7 Negated variables: queries on the toy data with negated variables, calculated with Inclusion-Exclusion. 95

6.3 Classification accuracy on WordNet test set. 103

6.4 F1 scores of the box lattice, order embeddings, and our smoothed model, for different levels of label imbalance on the WordNet *mammal* subset. 105

6.5 KL and Pearson correlation between model and gold probability. 106

6.6 Performance of the smoothed model, the original box model, and several baselines on MovieLens. 107

6.7 Performance of the original box model and smoothed box model on MovieLens, as a function of different degrees of disjointness upon initialization. 108

7.3 Results on Google Syntactic N-Gram SVO classification task using single and additive box models. 126

LIST OF FIGURES

Figure **Page**

1.1 An illustration of two geometric embedding methods representing the relationship between attributes *mammal*, *herbivore*, and *rabbit*. 3

1.2 Interpretation of box embedding model in terms of events in the base probability space Ω. Each of the outcomes $\omega_1, \omega_2, \omega_3$ sampled according to the base measure P corresponds to a realization of the binary variables $P(\text{mammal}, \text{herbivore}, \text{rabbit})$ 8

1.3 Illustration of the latent organization of continuous sentence embedding variables in a variational autoencoder applied to language modeling. The outer circle represents the white Gaussian prior, while the smaller ovals represent Gaussian posteriors over sentence representations, with similar sentences given nearby representations. 9

2.1 Illustration of the product Lebesgue measure in two dimensions, approximating the area of an arbitrary set with a sequence of rectangles. 17

2.2 Example human-sourced semantic similarity scores for words. Word representation learning can automatically capture these intuitive notions from large text corpora. 32

2.3 The canonical distributed word embedding algorithm. Energy is decreased between nearby words, and increased between randomly sampled word pairs. 33

2.4 A fragment of the WordNet hypernymy graph. 35

2.5 Example negative and positive entailment relations between words. 36

3.1 Learned diagonal variances, as used in evaluation (Section 3.6), for each word, with the first letter of each word indicating the position of its mean. We project onto generalized eigenvectors between the mixture means and variance of query word *Bach*. Nearby words to *Bach* are other composers e.g. *Mozart*, which lead to similar pictures. 42

3.2 Elements of the top 100 nearest neighbor sets for chosen query words, sorted by descending variance (as measured by determinant of covariance matrix). Note that less specific and more ambiguous words have greater variance. 54

3.3 Entailment: We compare empirical and learned variances, both diagonal (D) and spherical (S). E is the dataset of [6]. Measures of similarity are symmetric (cosine between means) and asymmetric (KL) divergence for Gaussians. balAPinc is an asymmetric similarity measure specific to sparse, distributional count-based representations. 55

3.4 Synthetic experiments on embedding two simple hierarchies in two dimensions directly using KL divergence. The embedding model captures all of the hierarchical relationships present in the tree. Sibling leaves are pushed into overlapping areas by the objective function. 56

3.5 Similarity: We evaluate our learned Gaussian embeddings (L) with spherical (S) and diagonal (D) variances, on several word similarity benchmarks, compared against standard Skip-Gram (SG) embeddings on the trained on the same dataset. We evaluate Gaussian embeddings with both cosine between means (m), and cosine between the distributions themselves (d) as defined by the expected likelihood inner product. 58

4.1 Example (standard and probabilistic) order embeddings in two dimensions. In the probabilistic case, the area of each cone corresponds to a Bernoulli probability. 66

4.2 Converting densities to regions in hierarchical density order embeddings. 69

5.1 A visualization of the Bernoulli covariance formula $\mathbb{E}[AB] - \mathbb{E}[A]\mathbb{E}[B]$ for box and cone embeddings in the unit square in two dimensions, showing positive, zero, and negative correlation. 85

5.2 Representation of the toy probabilistic lattice used in Section 5.4.1. Darker color corresponds to more unary marginal probability. Edges represent specified conditional probabilities, with the rest left up to inference. The associated CPD is obtained by a weighted aggregation of leaf elements. 88

5.3 Lattice representations and conditional probabilities from POE vs. box lattice. Note how the box lattice model's lack of "anchoring" to a corner allows it vastly more expressivity in matching the ground truth CPD seen in Figure 5.2. 89

5.4 R between model and gold probabilities. 92

6.1 One-dimensional example demonstrating two disjoint indicators of intervals before and after the application of a smoothing kernel. The area under the purple product curve is proportional to the degree of overlap. 98

6.2 Comparison of different overlap functions for two boxes of width 0.3 as a function of their centers. Note that in order to achieve high overlap, the Gaussian model must drastically lower its temperature, causing vanishing gradients in the tails. 103

6.3 Distribution of probabilities in MovieLens Dataset. 107

7.1 Any box that does not overlap $A \cap B$ will lie entirely within region 1, 2, 3 or 4, and thus cannot intersect both A and B and satisfy the constraints. 118

7.2 Union of boxes model in two dimensions. 119

7.3 A demonstration of the space partitioning used by the additive box model, as well as a solution to the original counterexample. 122

7.4 Heatmaps of model output on validation set with random negatives, as a function of training iteration. The single-box model does not maintain good separation between the probability of positive and negative examples, as it cannot avoid spurious overlapping with just one box. 127

CHAPTER 1

INTRODUCTION

1.1 Representation, Learning, and Inference

A fundamental problem in artificial intelligence and machine learning is that of *representation*. Before a machine can reason over data, that data must be represented in a way that allows for useful inferences to be made, and model parameters to be fit and adjusted.

Early AI systems, such as the General Problem Solver [66], represent and reason over data symbolically — knowledge is represented as a set of abstract discrete symbols, along with rewriting rules that permit various transformations between them, with reasoning carried out by search. Learning-based systems, such as the perceptron [62], would go on to augment or replace this purely symbolic approach with manually engineered *feature-based* representations. Such manually engineered feature representations include counts of terms or word n-grams for representing language [79, 88, 77], mel-frequency cepstral coefficients for audio waveforms [58], and SIFT [57] or HOG [24] features for representing images.

In the perceptron, the prototypical learning-based binary classification model, data points are associated with fixed feature vectors, and classification decisions are scored using the inner product with a parameter vector [62]. This representation allows the parameters to be adjusted to maximize accuracy by following the gradient of an error surrogate on a labeled dataset. In probabilistic modeling, the *exponential family* fills the same role as the perceptron — unnormalized log-probabilities for events are computed by the inner product of parameters and a feature representation of the variables called the *sufficient statistics* [87].

After symbolic reasoning is augmented by manual featurization, the next step is *representation learning* or *feature learning*. Rather than relying on hand-crafted feature functions, representation learning seeks to take a very informative, but hopelessly sparse and/or high-dimensional representation of the data, and condense this into a small enough dimension that useful parameter estimation can be performed. One set of representation learning methods are the *self-supervised*, also known as *unsupervised*, algorithms. In these, complex data is broken up into pieces and representations are learned so as to make it easy to reconstruct (learning representations for one task and using them on a different one is generally known as *transfer learning*).

These self-supervised algorithms range from simple count-based models, such as distributional word representations, where words are represented by sparse vectors of counts of nearby words from a corpus [39, 20], to representations in terms of word clusters [17], to nonlinear compression models such as autoencoders and restricted Boltzmann machines [43, 75].

In addition to transfer-learning approaches to feature learning, some models learn feature representations from labeled data during supervised training. The *neural network* approaches, such as the multi-layer perceptron or embedding-based methods [78], parameterize a mapping from symbolic data to several layers of continuous representations, effectively learning the feature functions with which to represent data.

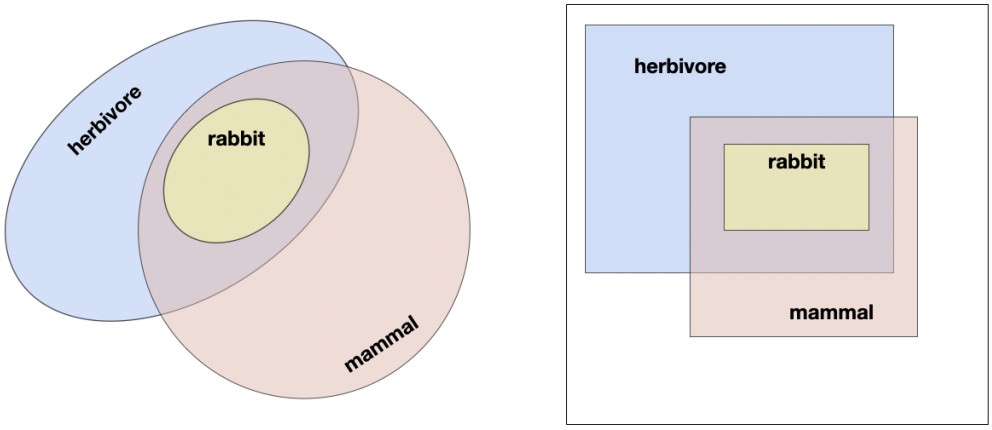
While standard feature learning focuses on learning transformations of fixed input data to best inform a prediction over output variables, an equally important type of representation is the way the output variables themselves are presented to the inference algorithm. As mentioned previously, exponential families are one such representation, where configurations of inference variables are represented as a vector of sufficient statistics and scored with a linear energy function to produce log-probabilities. The graphical modeling paradigm (in most cases) defines a subset of exponential families [87]. They allow inference over complex structures by representing them not symbolically, but with a collection

of sufficient statistics functions designed to work with algorithms such as dynamic programming. While not learning a representation of the variables of interest, the graphical modeling formalism uses particular hand-designed representations to perform consistent and tractable predictions over large sets of interacting variables.

While less heavily explored, methods for learning representation of inference variables (sometimes called *output representations*) have been studied, including learning the structure of graphical models [26], sum-product networks [34], and structured prediction energy networks [8].

This thesis will explore novel methods of representation learning, called *geometric representations*, that naturally lend themselves towards certain prediction and reasoning problems, enabling both inference and learning to take place in the same space and operate on the same objects.

1.2 Geometric Representations



(a) Gaussian embeddings (b) Box embeddings

Figure 1.1: An illustration of two geometric embedding methods representing the relationship between attributes *mammal*, *herbivore*, and *rabbit*.

The majority of machine learning operates in some way on a geometric space, usually a vector space. Classic representation learning algorithms such as *distributed word vectors* create a vector space model where neighborhoods of vectors correspond to semantically similar words. This thesis advocates moving beyond vectors, taking these geometric intuitions and designing richer ways to represent data. Rather than points we can imagine learning ovals, boxes, and cones, objects that naturally possess properties of asymmetric inclusion, specificity, breadth, transitive ordering, etc. Geometric representations are a natural way to provide an inductive bias when dealing with data that naturally possesses these relationships, in both supervised and unsupervised settings.

In this thesis, we present two novel types of geometric representations, the *Gaussian embeddings*, and the *box embeddings*.

Definition 1 (Gaussian embedding)**.** Given a set of objects S, a *Gaussian embedding* of S is a function $f : S \to \mathcal{L}^2(\mathbb{R}^d)$ which associates each object $s \in S$ with a multivariate Gaussian density function $f(s) = \phi_s(x; \mu_s, \Sigma_s)$ over $\mathbb{R}^d$.

The Gaussian embeddings [85, 2, 3, 12], represent objects as elements of a space of Gaussian distributions, with asymmetric comparisons given by KL- and alpha- divergences, and symmetric compatibility given by function-space inner products. These comparison functions leverage the geometry of e.g. the level sets of Gaussian densities to define natural geometric notions of soft subset inclusion, arising automatically from the form of the Gaussian function.

After the Gaussian embeddings, there was further work on geometric representation learning, leading to the development of models such as order embeddings [84, 55], which represent objects as components of a finite-dimensional vector lattice, as well as Poincaré embeddings [67] and hyperbolic entailment cones [32], which represent objects as regions in hyperbolic space.

In response to further development in geometric representations, especially the work in embedding objects into lattices of cone-shaped regions and probabilistic extensions thereof

[55], we developed *box embeddings* [86, 80, 56], which represent objects as axis-aligned hyperrectangles.

Definition 2 (Box embedding)**.** Given a set of objects S, and a measurable space $(\Omega_{\text{Box}}, \mathcal{E})$ with $\Omega_{\text{Box}} \subseteq \mathbb{R}^d$, a *box embedding* of S is a function $f : S \to \mathcal{E}$ which associates each object $s \in S$ with a box-shaped (axis-aligned hyperrectangular) subset of Ω_{Box},

$$x^{s,\wedge}, x^{s,\vee} \in \Omega_{\text{Box}} \subseteq \mathbb{R}^d \quad \text{and} \quad \forall j \in \{1, \ldots, d\}, \quad x_j^{s,\wedge} < x_j^{s,\vee}$$
$$f(s) = \prod_{j=1}^{d} [x_j^{s,\wedge}, x_j^{s,\vee}].$$

Further, if the underlying measurable space $(\Omega_{\text{Box}}, \mathcal{E})$ has a probability measure P, for each $s \in S$ we can define a binary random variable X_i such that $X_i = \mathbb{1}_{f(s)}$ is the indicator function of the set $f(s)$. The distribution $P(\{X_i\})$ is called a *box probability model*, and the mapping $S \to (\Omega_{\text{Box}} \to \{0, 1\})$ is called a *probabilistic box embedding* of S.

While the above definition is slightly technical, the necessary background math to understand it is provided in Chapter 2, and a more detailed exposition of box embeddings is provided in Chapter 5. In short, box embeddings associate each domain object with a box, and define a probability distribution over those objects when considered as binary attributes. The probability of each attribute is proportional to the size of its box, and co-occurrences are determined by box overlaps.

Box embeddings can easily encode asymmetric relationships between concepts through inclusion and overlap, and related properties such as transitivity. As is the intention with other geometric representations, this inductive bias leaves less to be learned from the data than with arbitrary relational models. Additionally, they provide a rich, consistent probabilistic interpretation of the learned representations, and the ability to learn and model complex interactions.

In addition to the inductive bias provided by various geometric forms, the representations discussed in this thesis can be interpreted as connected to the concept of structured prediction, and the probabilistic notions of event sets and latent variables.

1.2.1 Geometric Representation Learning as Structured Prediction

Representations in the form of richer geometric objects can lend themselves to natural asymmetric relations, which may possess transitivity (a property of commonplace relations such as partially ordered sets and lattices). These models do not only enjoy a strong inductive bias, leaving less to be learned from the data than arbitrary relational models, but at their best resemble a hybrid between embedding models and structured prediction. As noted by Vendrov et al. [84] and Li et al. [55], while the models learn sets of embeddings, these parameters obey rich structural constraints. The entire set can be thought of as one, sometimes provably consistent, structured prediction, such as an ontology in the form of a single directed acyclic graph, or a distribution over such objects.

In the case of order embeddings, or the lattice of box embeddings presented in this thesis, implications between concepts are transitive, so while the model is learned by independently sampled stochastic gradient descent updates, the final set of embeddings encodes a consistent partial order, lattice, or probability distribution. An analogy can be made here to other forms of inference which are cast as optimizing an energy function over a space of probability distributions or assignments. However, it is the geometric structure of the representation space that leads to a consistent joint prediction over the variables, rather than a more abstract set of consistency constraints. For example, when using order embeddings to predict graph edges, we predict an edge when the cone-shaped order embedding region is contained within another region. In this case, the transitivity of the subset relation between these geometric objects (convex cones) ensures that we always predict a transitive graph.

1.2.2 Event Sets, Probability and Representation Learning

An alternative motivation for, and interpretation of, geometric representations comes from the probabilistic notions of event sets and continuous latent variables.

Any introductory probability class teaches the definition of a probability distribution as a combination of a sample space Ω, a (σ-)algebra of sets $\mathcal{E}$, and a probability measure P. Each element of the sample space Ω represents the *outcome* of some idealized experiment, and given an outcome ω sampled according to P, we say that all of the *events* $E \in \mathcal{E}$, such that $\omega \in E$, have *occurred*.

The probabilistic representations discussed in this thesis, probabilistic order embeddings (POE) and the box lattice, can be interpreted through exactly this lens. That is, each box $\mathrm{Box}(X)$ corresponds to a binary random variable X defined as the indicator function of the set $\mathrm{Box}(X)$. A full probability distribution over all of the binary variables $\{X_i\}$ is thus defined by the positions of all of the latent box representations, and the base measure P from which outcomes in Ω are sampled. This corresponds to a novel form of probabilistic learning, where rather than learning the parameters of a distribution, we are inferring the underlying structure of an event space. Alternately, we are learning the support of an indicator random variable. An illustration of this correspondence is given in Figure 1.2.

A further connection between geometric representation learning and probability theory comes from the notion of continuous latent variable models. In a continuous latent variable model, e.g. the variational autoencoder [48, 71], observations in the domain of interest correspond to sets of points in a learned, smooth continuous space. Higher probability observations correspond to larger latent regions, with the entire latent space organized by similarity. Concretely, define a generative story line where we sample a Gaussian latent variable from a white prior distribution $P(Z)$, and then map that through e.g. a neural network language model decoder $P(X|Z)$ to generate a sentence, as in Bowman et al. [15]. The vectors in the latent space, distributed according to $P(Z)$, act as representations of the decoded sentence X, and the regions given high probability by posterior distribution over

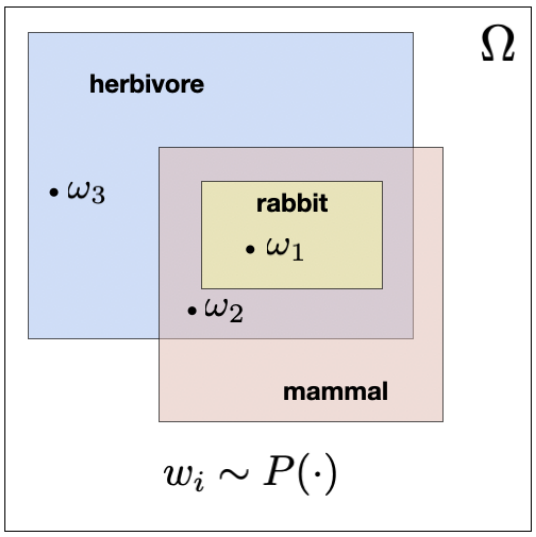


Figure 1.2: Interpretation of box embedding model in terms of events in the base probability space Ω. Each of the outcomes $\omega_1, \omega_2, \omega_3$ sampled according to the base measure P corresponds to a realization of the binary variables $P(\text{mammal}, \text{herbivore}, \text{rabbit})$

representations given a sentence, $P(Z|X)$, take the form of contiguous subsets (ellipsoidal in the case of Gaussian posterior approximators). In this case, the latent representations are not explicitly parametrized geometric objects, but rather self-organized implicit subsets. An illustration of this is given in Figure 1.3. Moreover, work in complex posterior approximations for approximate inference in continuous latent variable models [70] corresponds to learning irregular geometric shapes for the high probability regions in code space for a given observation.

1.2.3 Overview

In this thesis, we develop embedding methods that map domain objects to representations possessing richer geometric structure than the popular vector embedding models. These representations allow novel forms of reasoning to be performed directly in the embedded space, using geometric properties to express ideas of asymmetry, hierarchy, and entailment. We present a model that embeds words in the space of Gaussian distributions,

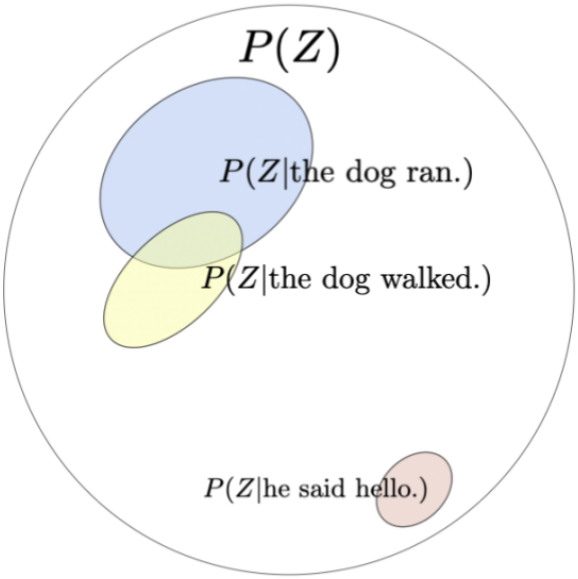


Figure 1.3: Illustration of the latent organization of continuous sentence embedding variables in a variational autoencoder applied to language modeling. The outer circle represents the white Gaussian prior, while the smaller ovals represent Gaussian posteriors over sentence representations, with similar sentences given nearby representations.

as well as probabilistic box embeddings, a model that represents objects as elements of a lattice of axis-aligned hyperrectangles. We propose two learning methods for the box lattice: a method based on surrogate functions, and an approximation inspired by Gaussian convolutions. Finally, we examine theoretical and practical properties of the box lattice and present extensions to deal with its weaknesses. The remainder of the thesis is organized as follows. In Chapter 3.3, we present some basic concepts in probability and measure theory, an abridged background on energy-based modeling and learning, and a description of some of the application areas of this thesis and baseline approaches to them. In Chapter 3, we present the Gaussian word embedding model, as well as related follow-on work using Gaussian embeddings. In Chapter 4, we give some background on order theory and lattice theory, and present order embeddings [84], Poincaré embeddings [67], and related models. In Chapter 5, we present the box lattice and the probabilistic box embedding model. In Chapter 6, we present a smoothed training method for probabilistic box embeddings.

In Chapter 7, we identify some theoretical strengths and weaknesses of box embeddings compared to other methods, identify a class of counterexample distributions that cannot be represented by box embeddings, and present an extension of the model to solve this problem.

1.3 Summary of Contributions

- A model for embedding words in the space of Gaussian distributions, allowing for easy asymmetric comparison inspired by geometric containment within level-sets of ellipsoids (3).

- A generalization of probabilistic order embeddings to arbitrary base measures and formalization in terms of random variables (4).

- Probabilistic box embeddings, a model for embedding concepts as axis-aligned hyperrectangles (boxes), with an accompanying partial order and lattice structure over the box embeddings, as well as a measure that allows the boxes to represent probability distributions over concepts (5).

- Methods for training box embeddings to model probability distributions:

  - A surrogate lower-bound of the likelihood (5)

  - A smooth relaxation of the energy function inspired by Gaussian convolutions (6)

- Theoretical examination of some representational properties of box embeddings and related models (7.3).

- Extensions of probabilistic box embeddings to handle modeling failure cases of the original model (7.5).

1.4 Declaration of Collaborations

The following work described in this thesis has been previously published and/or done in collaboration with other researchers.[1]

- Work on Gaussian embeddings, presented in Chapter 3, was previously published as Vilnis and McCallum [85].
- The work introducing box embeddings and generalizing probabilistic order embeddings, presented in Chapters 4 and 5, was done in collaboration with Xiang Li (equal contribution) and Shikhar Murti and previously published as Vilnis et al. [86].
- The work on smoothed learning for box embeddings, presented in Chapter 6, was done in collaboration with Xiang Li (equal contribution), Dongxu Zhang, and Michael Boratko and previously published as Li et al. [56].
- The ongoing work on representational power of box embeddings, presented in Chapter 7, has been done in collaboration with Michael Boratko, Dongxu Zhang, and Xiang Li.

[1] Note that all of the work presented here was done under the guidance of Professor Andrew McCallum.

CHAPTER 2

PRELIMINARY BACKGROUND

This thesis will draw on mathematics broadly from the areas of probability theory and order theory. The mathematical content of this chapter will focus mostly on probability, with definitions from order theory introduced in Chapter 4.

In order to motivate and formalize the models which are the content of this thesis, we first establish some definitions and background. Because some of our contributions are interleaved with the related work, the background will be presented in two different chapters. This chapter concerns itself with basic definitions common to many different machine learning models. The remainder of the background, concerning order theory and derived models, will be presented in Chapter 4.

2.1 Notation

Throughout the thesis, we will attempt to keep certain types of notation for common concepts consistent. This notation is presented in Table 2.1.

2.2 Measures

To introduce the small amount of probability theory we will need for this thesis, we first provide several definitions from analysis. This will allow us to define the notion of *measure*, a mathematical concept that allows us to sensibly define volumes, lengths, areas, and similar concepts for many types of sets.

The first notion we need is that of a *topological space*.

Symbol	Meaning
$P(\cdot)$	A probability measure or probability distribution.
$X : \Omega \to \Omega'$	A random variable.
$X \sim P$	The random variable X has distribution P.
$P_X(\cdot), P(X)$	The probability measure associated with the random variable X.
$P(X = x), P(X \in E)$	The probability of an event under the distribution of X.
$f_X(x)$	The probability density associated with the continuous random variable X.
$F_X(x)$	The (cumulative) distribution function of the continuous random variable X.
$p_X(x)$	The probability mass function associated with the discrete random variable X.
$\mathbb{E}[\cdot]$	The expectation of a random variable.
$\phi(x; \mu, \sigma)$	The probability density function of the normal distribution.
$\Phi(x; \mu, \sigma)$	The cumulative distribution function of the normal distribution.
$\mathcal{N}(\mu, \Sigma)$	Multivariate normal distribution.
$D_{KL}(Q\|P)$	KL divergence from P to Q.
$E(x, y, w)$	An energy function with inputs x, outputs y, and weights w.
$a \vee b$	The join of two lattice elements or maximum of two numbers.
$a \wedge b$	The meet of two lattice elements or minimum of two numbers.
a, b	Variables representing scalars.
u, v	Variables representing vectors.
U, V	Variables representing matrices.
$\langle \cdot, \cdot \rangle$	The inner product between two vectors.
$\sigma(\cdot)$	The logistic sigmoid function.

Table 2.1: Notation

Definition 3 (Topological space)**.** A *topological space* is an ordered pair $(S, \mathcal{T})$ where S is a set and $\mathcal{T}$ is a collection of subsets, called *open sets*, that satisfy the following four conditions:

1. $\varnothing \in \mathcal{T}$
2. $S \in \mathcal{T}$
3. For any finite number of sets in $\mathcal{T}$, their intersection is also in $\mathcal{T}$.
4. For any arbitrary (possibly countably or uncountably infinite) number of sets in $\mathcal{T}$, their union is also in $\mathcal{T}$.

The set $\mathcal{T}$ is called a *topology* on S. A topological space gives us a notion of the connectivity structure of a set when considered as a geometric object, that is, which subsets are nearby (share neighborhoods with) other sets.

Definition 4 (Basis)**.** Let $\mathcal{T}$ be a topology on S, and $\mathcal{B} \subset \mathcal{T}$. If $\mathcal{T}$ is the smallest topology that contains $\mathcal{B}$, then we say $\mathcal{B}$ is a *basis* for $\mathcal{T}$.

Remark. We refer to the operation to produce a topology $\mathcal{T}$ from a basis $\mathcal{B}$ as taking the *closure* of $\mathcal{B}$ under unions and finite intersections, and we call $\mathcal{T}$ *closed* under unions and finite intersections. We say that $\mathcal{T}$ is *generated* by $\mathcal{B}$.

On the real line $\mathbb{R}$, a natural topology is generated by the open intervals, corresponding to a simple linear connectivity structure.

Example (Standard topology)**.** The set of open intervals in $\mathbb{R}$, $(a, b) \coloneqq \{\, x \in \mathbb{R} \mid a < x < b \,\}$, is a basis for a topology on $\mathbb{R}$, called the *standard topology* on $\mathbb{R}$.

We can also define a topology generated by the singleton sets. This topology is most useful on finite sets, like spaces of discrete structures.

Example (Discrete topology)**.** The set of singleton subsets of any set S, $\mathcal{B} \coloneqq \{\, \{x\} \mid x \in S \,\}$, is a basis for a topology on S, called the *discrete topology* on X.

In addition to these topological concepts, which give a notion of connectivity and nearness between subsets, we introduce some concepts from measure theory, which will give us a notion of the size of various subsets. First we define a *σ-algebra*.

Definition 5 (σ-algebra)**.** A *σ-algebra* (or *σ-field*) on a set S is a collection $\mathcal{F}$ of subsets of S that satisfies the following conditions:

1. $S \in \mathcal{F}$
2. For any countable number of sets in $\mathcal{F}$, their union is also in $\mathcal{F}$.
3. For any set in $\mathcal{F}$, its complement is also in $\mathcal{F}$.

Note that this definition implies that $\mathcal{F}$ contains the empty set, and is also closed under countable intersections.

Definition 6 (Borel σ-algebra)**.** The *Borel σ-algebra* on a topological space $(S, \mathcal{T})$, which will be the only σ-algebra that we concern ourselves with in this thesis, is the smallest σ-algebra containing all of the open sets $\mathcal{T}$ of S. The elements of the σ-algebra are called *Borel sets*. That is, we generate the Borel sets by taking the closure of T under complement, countable union, and countable intersection.

Given a σ-algebra $\mathcal{F}$ on a set S, we can define a *measure*.

Definition 7 (Measure)**.** Given a set S and a σ-algebra $\mathcal{F}$ over S, a *measure* is a function μ from $\mathcal{F}$ to $\mathbb{R} \cup \{\infty\}$ satisfying the following properties:

1. **Null empty set**: $\mu(\varnothing) = 0$.
2. **Non-negativity**: $\mu(A) \geq 0$ for all $A \in \mathcal{F}$.
3. **Countable additivity**: If $\{A_i\}$ is a countable collection of pairwise disjoint sets in $\mathcal{F}$, then $\mu(\bigcup_i A_i) = \sum_i \mu(A_i)$.

Remark. An ordered pair of a set and a σ-algebra $(S, \mathcal{F})$ is called a *measurable space*.

Definition 8 (Measure space)**.** An ordered triple $(S, \mathcal{F}, \mu)$, where S is a set, $\mathcal{F}$ is a σ-algebra on S, and μ is a measure on $\mathcal{F}$, is called a *measure space*.

Definition 9 (Measurable function)**.** Let $(S, \mathcal{F}, \mu)$ be a measure space and $(T, \mathcal{G})$ be a measurable space, and f be a function from S to T. We call f a *measurable function* if $f^{-1}(B) \in \mathcal{F}$ for all $B \in \mathcal{G}$.

A *Borel measure* on a topological space $(S, \mathcal{T})$ is any measure μ defined on the Borel σ-algebra on $(S, \mathcal{T})$.

As we have previously stated, a measure corresponds to a notion of length, area, or volume. We can use this intuition to arrive at a good choice of standard measure for many sets. For example, let us define a measure on $\mathbb{R}$ with the Borel σ-algebra given by the standard topology generated by open intervals. Assign $\mu([a, b]) = b - a$ for an interval

$[a, b]$. Now, given a subset $E \subset \mathbb{R}$ let A be a countable collection of intervals whose union contains E, and C be the family of all such collections of intervals. Then,

$$\mu(E) = \inf_{A \in C} \sum_{[a,b] \in A} \mu([a, b])$$

Note that this definition agrees with our intuitive notion of length for sets that can be approximated by a union of intervals.

In a slight abuse of terminology, but not one that is important for purposes of this thesis, we will refer to this as the *Lebesgue measure* on $\mathbb{R}$ (it is actually the restriction of the Lebesgue measure to the Borel sub-σ-algebra of the Lebesgue σ-algebra, but that is a mouthful).

In order to extend measures to higher dimensional spaces than simply $\mathbb{R}$, we define the notions of *box topology* and *product measure*.

Definition 10 (Box topology)**.** Given two topological spaces $(S, \mathcal{T})$ and $(T, \mathcal{U})$, define the *box topology* on the product space $S \times T$ as the topology generated by the basis $\{ A \times B \mid A \in \mathcal{T}, B \in \mathcal{U} \}$.

The standard topology on $\mathbb{R}^n$ can be constructed as the box topology on n copies of $\mathbb{R}$ equipped with the standard topology. That is, the standard topology on $\mathbb{R}^n$ is generated by Cartesian products of open intervals in each dimension.

Definition 11 (Product measure)**.** Let $(S, \mathcal{F}, \mu)$ and $(T, \mathcal{G}, \nu)$ be measure spaces. Let $\mathcal{F} \otimes \mathcal{G}$ denote the σ-algebra on the Cartesian product set $S \times T$ which is generated by sets of the form $\{ A \times B \mid A \in \mathcal{F}, B \in \mathcal{G} \}$. Define a *product measure*, $\mu \times \nu$, on the measure space $(S \times T, \mathcal{F} \otimes \mathcal{G}, \mu \times \nu)$, as a measure satisfying

$$(\mu \times \nu)(A \times B) = \mu(A)\nu(B)$$

for all $A \in \mathcal{F}$ and $B \in \mathcal{G}$. For all measures of interest in this thesis, this requirement uniquely defines the measure.

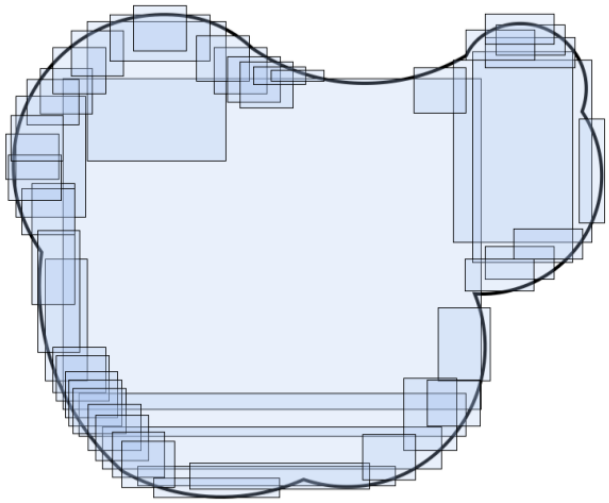


Figure 2.1: Illustration of the product Lebesgue measure in two dimensions, approximating the area of an arbitrary set with a sequence of rectangles.

A product measure can be defined on $\mathbb{R}^n$, equipped with the standard topology and associated Borel σ-algebra. As in the 1-dimensional construction, let $\mu([a_1, b_1] \times ... \times [a_n, b_n]) = (b_1 - a_1)...(b_n - a_n)$ for a product of intervals $[a_1, b_1] \times ... \times [a_n, b_n]$. Now, given a subset $E \subset \mathbb{R}$ let A be a countable collection of products of intervals whose union contains E, and C be the family of all such collections. Then,

$$\mu(E) = \inf_{A \in C} \sum_{[a_1,b_1]\times ...\times [a_n,b_n] \in A} \mu([a_1, b_1] \times ... \times [a_n, b_n])$$

In a similar minor abuse of terminology to the 1-dimensional case, we will refer to this as the *Lebesgue measure* on $\mathbb{R}^n$. Figure 2.1 demonstrates an approximation of the area of a set with a sequence of rectangles.

2.3 Probability

With our definitions from basic analysis in hand, we move on to introduce some concepts from probability theory. A *probabilistic model* is formalized in terms of a *probability space*, a set of outcomes that occur randomly. What those outcomes are and with what probability they occur is given by an ordered triple $(\Omega, \mathcal{E}, P)$, which defines a certain kind of measure space.

Definition 12 (Probability space)**.** Let $(\Omega, \mathcal{E}, P)$ be a measure space such that $\mathcal{E}$ is the Borel σ-algebra over the base set Ω with respect to some topology T, and let the measure P be *normalized* such that $P(\Omega) = 1$. Then we call Ω the *sample space*, $\mathcal{E}$ the set of *events*, and P the *probability measure*. This ordered triple is referred to as a *probability space*.

One interpretation of a probability space is as a model of a certain kind of situation. We can "run" the model to produce one of the outcomes in the sample space $\omega \in \Omega$, and we say that all of the events in $\mathcal{E}$ containing ω have occurred. The semantics of $(\Omega, \mathcal{E}, P)$ are such that with an infinite number of runs, we will see the relative frequencies of different events occur proportional to the measure given by P. This is called the *frequentist* interpretation of probability.

Remark. This assignment of probabilities to events is called a *probability distribution*, which is another term for the probability measure P, because they are both functions from events to probabilities. The only difference is we usually refer to P as a probability distribution when P is our chief object of interest, and a probability measure when we are using it to integrate some other function over the sample space, such as a random variable (introduced in the sequel).

Example (Uniform distribution)**.** Let $\Omega = [0, 1]$, $\mathcal{E}$ be the Borel σ-algebra restricted to $[0, 1]$, and P be the Lebesgue measure restricted to $\mathcal{E}$. Then the probability space $(\Omega, \mathcal{E}, P)$ describes a *uniform distribution* over $[0, 1]$. That is, for any interval $[a, b] \subset [0, 1]$, the

probability of a random element ω from the sample space falling into that interval is equal to $b - a$.

To extend and further formalize the concept of probability distributions, we introduce the notion of *random variable*.

Definition 13 (Random variable)**.** Let $(\Omega, \mathcal{E}, P)$ be a probability space, and $(\Omega', \mathcal{F})$ be a measurable space. A *random variable* is a measurable function $X : \Omega \to \Omega'$, and the probability that X takes on a value in a set $E \subset \Omega'$ is given by $P(X \in E) = P(\{\, \omega \in \Omega \mid X(\omega) \in E \,\})$.

Remark (Pushforward measure)**.** As the above definition suggests, a random variable, in addition to being a function from Ω to Ω', can also be thought of as inducing a new probability triple $(\Omega', \mathcal{F}, P)$, with the new probability measure P defined as above, and the original measure referred to as the *base measure*. This is called the *pushforward measure* and sometimes written as P_X when just re-using P would be ambiguous. Often random variables are discussed directly in terms of these measures rather than using a potentially more cumbersome definition via functions, and we say that $X \sim P$, *X has distribution P* and P is the probability distribution of X.

Example (Indicator random variables)**.** Given a probability space $(\Omega, \mathcal{E}, P)$ and an event $E \in \mathcal{E}$, an *indicator random variable* of E is a random variable $\mathbb{1}_E : \Omega \to \{0, 1\}$ that takes on the value 1 on the set E and 0 otherwise. Note that $\int_\Omega \mathbb{1}_E dP(\omega) = P(E)$. These variables play a special role in this thesis, as some of the representations learned by our models can be described as indicator random variables of certain simple events.

For the common case of real-valued random variables, the concept of a *distribution function* is useful.

Definition 14 ((Cumulative) distribution function)**.** The *distribution function*, also known as the *cumulative distribution function* or *CDF*, of a real-valued random variable X is defined as

$$F_X(x) = P(X \leq x)$$

F_X is a function $\mathbb{R} \to \mathbb{R}$ and the probability that X lies in the interval $(a, b]$ can be calculated as $P(a < x \leq b) = F_X(b) - F_X(a)$.

For any continuous random variable, we also have the notion of a *probability density function*.

Definition 15 (Probability density function)**.** The *probability density function* or *PDF* of a random variable $X : \Omega \to \Omega'$ is a function $f_X : \Omega' \to \mathbb{R}^+$ such that

$$P(X \in E) = \int_{x \in E} f_X(x) dx$$

for any subset E in Ω' and some base measure on Ω', usually taken as the standard Lebesgue measure.

We will not formally define the *Lebesgue integral* used in the above definition, but it is equivalent to the standard Riemann integral in all the applications in this thesis.

For the case of continuous real-valued random variables, the following result holds, similar to the fundamental theorem of calculus:

$$\int_a^b f_X(x) dx = F_X(b) - F_X(a)$$

Example (Gaussian random variable)**.** Define the *standard Gaussian* PDF,

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2},$$

and the Gaussian distribution function,

$$\Phi(x) = \int_{-\infty}^{x} \phi(x) dx.$$

Now given the probability triple $(\Omega, \mathcal{E}, P)$, where $\Omega = (0, 1)$, $\mathcal{E}$ is the Borel σ-algebra restricted to $(0, 1)$, and P is the Lebesgue measure restricted to $\mathcal{E}$, we define a Gaussian random variable $X : (0, 1) \to \mathbb{R}$ as

$$X(\omega) = \Phi^{-1}(\omega).$$

As we discussed earlier, we commonly push the base measure P forward through X to create the *Gaussian measure* on $\mathbb{R}$,

$$\gamma(A) = \int_{x \in A} \phi(x) dx,$$

and work with this object directly.

For discrete random variables, we instead have the notion of a *probability mass function*.

Definition 16 (Probability mass function)**.** For a discrete random variable $X : \Omega \to \Omega'$, the *probability mass function* or *PMF* is a function $p_X : \Omega' \to [0, 1]$ such that

$$p_X(x) = P(X = x).$$

Remark. As we discuss in more detail Section 2.6, in the context of probabilistic modeling, when we have a parameterized family of distributions which we are trying to fit to a certain set of observed data, the probability density (or mass) function evaluated on that data is referred to as the *likelihood function*. It is a function of the parameters and we call it the likelihood of the data given those parameters. Model selection is often done by maximizing this function.

Given a probability mass function or a probability density over a random variable X, we can define the *expectation* of a function g as

$$\mathbb{E}[g(X)] = \int_x f_X(x)g(x)dx \text{ or } \sum_x p_X(x)g(x).$$

The expectation can be used to define many interesting functions in probability theory, but one which we will use in several places throughout this thesis is the *Kullback-Liebler divergence* (KL-divergence), which is an asymmetric distance function between two random variables X and Y (or more generally, measures) having densities f_X and f_Y, with the same codomain Ω', defined as

$$D_{KL}(Y\|X) = \int_{\Omega'} f_X(\omega') \log(\frac{f_X(\omega')}{f_Y(\omega')})d\omega', \tag{2.1}$$

A similar formula holds for the case of discrete random variables,

$$D_{KL}(Y\|X) = \sum_{\Omega'} p_X(\omega') \log(\frac{p_X(\omega')}{p_Y(\omega')}),$$

This is read as the KL-divergence from $P(X)$ to $P(Y)$, and along with providing a natural distance between distributions, has several connections to inference and learning in probabilistic models.

Definition 17 (Marginal and joint probability)**.** With a probability space $(\Omega, \mathcal{E}, P)$ and a collection of random variables $X_1 : \Omega \to \Omega_1, ..., X_n : \Omega \to \Omega_n$, define the measure of a

product of events $E_1 \times ... \times E_n \in \Omega_1 \times ... \times \Omega_n$ as $P(X_1^{-1}(E_1) \cap ... \cap X_n^{-1}(E_n))$ and extend the measure to the Borel σ-algebra generated by the box topology on the entire product space. This is called the *joint distribution* of $X_1, ..., X_n$, written as $P(X_1, ..., X_n)$ or $P_{X_1 \times ... \times X_n}$. We can restrict this distribution to any subset of the variables $X_1, ..., X_n$, by slotting in the entire sample space Ω_k for the variables we want to remove. This is called *marginalizing* X_k, and the resulting distribution is called the *marginal distribution* of the remaining variables. For example, given three random variables (X_1, X_2, X_3), the marginal distribution of X_1 and X_2 is given as

$$P_{X_1 \times X_2}(E_1 \times E_2) = P(E_1 \times E_2 \times \Omega_3).$$

when applied to a product of events $E_1 \times E_2$, and extended to the rest of the Borel sets as usual. Further, if the joint distribution has a density (or mass) function, the marginal density (or mass) function can be found by integrating (summing) out the marginalized variables using Lebesgue or counting measure,

$$f_{X_1 \times X_2}(x_1, x_2) = \int_{x_3 \in \Omega_3} f_{X_1 \times X_2 \times X_3}(x_1, x_2, x_3) dx_3$$
$$p_{X_1 \times X_2}(x_1, x_2) = \sum_{x_3 \in \Omega_3} p_{X_1 \times X_2 \times X_3}(x_1, x_2, x_3)$$

Finally, we introduce the notion of *conditional probability*. Given a probability space $(\Omega, \mathcal{E}, P)$, let $A \in \mathcal{E}$ and $B \in \mathcal{E}$ be two events. Then we define the conditional probability

$$P(A|B) \coloneqq \frac{P(A \cap B)}{P(B)}.$$

When B is a set of measure zero, such as a singleton, this definition does not apply (in the continuous case). In this case, there are technical considerations beyond the scope of this chapter, but for our purposes it is enough to state the following in terms of random

variables. For a probability space $(\Omega, \mathcal{E}, P)$, and two random variables, $X : \Omega \to \Omega'$ and $Y : \Omega \to \Omega''$, push forward the measure P onto associated σ-algebras $\mathcal{F}$ and $\mathcal{G}$. Then the conditional probability $P(Y \in A|X = x)$ is a function from $\mathcal{F}$ to probability measures on $(\Omega'', \mathcal{G})$ such that for an event $A \in \mathcal{G}$,

$$\int_x f_X(x)P(Y \in A|X = x)dx = P(Y \in A).$$

Having introduced these basic definitions from probability, we move on to some concepts and applications that are more specific to machine learning and our domain in particular.

2.4 Energy-Based Models

An *energy function* assigns a real-valued numeric score to a configuration of variables. Here, the term *variables* is used in the informal sense to refer to any objects of relevance to the model, whether in-domain (e.g. a movie recommendation for a user, or a part-of-speech tag for a token), or at a higher level (e.g. the parameter vector of a classifier).

For example, a binary linear classifier that maps patterns $x \in \mathbb{R}^d$ to outcomes $y \in \{-1, 1\}$, using the parameter vector $w \in \mathbb{R}^d$, has the energy function

$$E(x, y, w) = -yw^\top x.$$

Because many energy functions are parametrized by some observed variables (like x) and model parameters (like w), we will sometimes elide the explicit w or x arguments when the context is clear. For example, in binary classification, we are given a pattern x and asked to predict a label y. In order to make these relationships clear, we could slightly abuse notation and write the binary classification energy function as

$$E(y) = -yw^\top x.$$

This notation lends itself to the interpretation that E defines a *family* of energy functions over $\{-1, 1\}$, parametrized by the choice of x and w, rather than a single energy function over all three. Either interpretation can be more useful, depending on the application.

Remark (MAP inference)**.** The most basic task to which an energy function is applied is that of *MAP inference*. In MAP inference, given the settings of some variables, we attempt to predict the settings of other variables. With an energy-based model, we do this by finding the lowest-energy configuration of our model consistent with those pre-set variables. To return to our example of the binary classifier, for a given input pattern x and parameter vector w, we predict the label y by performing energy minimization

$$\begin{aligned} y^* &= \arg\min_{y \in \{-1,1\}} E(y) \\ &= \arg\min_{y \in \{-1,1\}} -yw^\top x \\ &= \text{sign}(w^\top x). \end{aligned}$$

This gives the familiar decision rule for binary linear classification.

Remark (Structured prediction)**.** Energy-based models are often applied to *structured prediction*, defined as the task of jointly predicting multiple variables, which interact to make up a single complex object, so as to minimize an error defined jointly over those variables or that object. While the definition is not entirely rigorous, normally, when an output variable has a clear compositional structure, or once the cardinality of a discrete output space is large enough that we cannot comfortably enumerate it, we are performing structured prediction. In this thesis we will mainly concern ourselves with models of large sets of variables which interact with one another non-trivially, consistently, and in some cases are evaluated jointly. Much of this thesis concerns itself with structured prediction, although our approaches and applications may differ greatly from the usual use of this term.

2.5 Probabilistic Energy-Based Models

Given any energy function, we can define a probability density or mass function over $\mathcal{Y}$ and an associated random variable Y:

$$f_Y(y) = \frac{\exp(-E(y))}{Z},$$

where Z is the *normalization constant*, which takes an energy-weighted average over all possible outcomes

$$Z = \int_{y' \in \mathcal{Y}} \exp(-E(y'))dy.$$

Definition 18 (Exponential families)**.** The above mapping from energy-based models to probability distributions leads us to a special class of probabilistic models, called *exponential families*, can be written in terms of specifically factorized underlying energy function:

$$\begin{aligned} E(y, \theta) &= -\log h(y) - \theta^\top S(y) \\ f_Y(y) &= \frac{h(y)\exp(\theta^\top S(y))}{Z}, \end{aligned}$$

where $h(\cdot) : \mathcal{Y} \to \mathbb{R}^+$ is the *base measure*, $\theta \in \mathbb{R}^d$ is the vector of *natural parameters*, and $S(\cdot) : \mathcal{Y} \to \mathbb{R}^d$ is called the *sufficient statistics* function. In exponential families, the normalization constant Z is also known as the partition function.

Remark (Marginal inference)**.** Given a probabilistic model over many variables, we often want to compute the marginal distribution over only a few variables of interest, averaging over the other variables. The task of computing marginals is known as *marginal inference*, and is one of the chief concerns of this thesis.

2.6 Learning

2.6.1 Loss functions

In order to apply an energy function to the task of inference, in most cases there are associated parameters that must be adjusted based on observed data. This interaction between the labeled data and the energy function is mediated by a *loss function*. The loss function takes one or more sets of variable assignments, termed *training examples*, and using the current energy function produces a real-valued numeric score, the *loss*. The goal of learning is to adjust the energy function's parameters to minimize loss on the training examples. For example, the *perceptron loss* is defined, for an energy function E and training example y_i, as

$$\ell(y_i) = E(y_i) - E(y^*)$$
$$y^* = \arg\min_{y \in \mathcal{Y}} E(y).$$

Specialized to our running example of binary linear classification, we have

$$\ell(x_i, y_i, w) = \max(-y_i w^\top x_i, 0).$$

Taking the subgradient of this loss function with respect to θ, we see the familiar update rule for the binary perceptron, justifying the name, and also providing a perspective on perceptron learning as stochastic subgradient descent on a loss function, which is discussed in the next section.

A similar loss, which we use in this thesis, goes by several names, including the *structured SVM loss*. Similar to the perceptron loss, we want to give the correct value y_i a lower score than every other value, but we also want to create a positive margin of at least m between the score of the correct value and every other value

$$\ell(y_i) = \max(0, m + \max_{y \neq y_i} E(y) - E(y_i)).$$

The final family of loss functions we use in this thesis comes from the perspective of probabilistic modeling. Given an exponential family model, or any parametric distributions in general, we often want to fit the parameters to observed data by *maximizing the likelihood of the data* or, equivalently minimizing the negative log-likelihood. The latter convention fits right into our loss minimization framework.

If Y is a random variable whose probability distribution has the following density,

$$f_Y(y) = \frac{h(y)\exp(\theta^\top S(y))}{Z},$$

Then the *negative log likelihood* $-\log f_Y(y|\theta)$ for some observed data y is a loss function for fitting θ to match our observations.

A common model for probabilistic binary classification is the *Bernoulli* log-likelihood, corresponding to logistic regression in the probabilistic framework, and the *log-loss* in the energy framework.

Minimizing negative log-likelihood is also equivalent to minimizing the KL-divergence from the model to the empirical distribution of data, giving further intuition for our choice of objective function.

2.6.2 Learning with stochastic gradient descent

General supervised learning problems take the form of an optimization, with the goal of finding a parameter vector w that has low generalization error for a particular distribution of inputs and target variables $P(X, Y)$,

$$w^* = \arg\min_w \mathbb{E}[\ell(X, Y, w)].$$

This is generally done by minimizing a regularized loss over a set of training examples $\{(x_i, y_i)\}$,

$$w^* = \arg\min_w \frac{1}{n} \sum_i \ell(x_i, y_i, w) + R(w), \tag{2.2}$$

where R is some regularization function which penalizes the model for being too complex, to encourage generalization.

A common approach to solving these problems, and the one which will be used throughout this thesis, is *stochastic gradient descent* (or a similar and usually equivalent framework called *online gradient descent*).

Stochastic gradient descent (SGD) uses the structure of the regularized loss minimization equation (2.2) to derive an efficient randomized approximation to the gradient of the sum of losses term.

First, define the minibatch gradient function,

$$g(\{i_1, ..., i_k\}) = \frac{1}{k} \sum_{i \in \{i_1, ..., i_k\}} \nabla_w \ell(x_i, y_i, w),$$

which calculates the average gradient of the loss function when applied to a set of examples indexed by the indices $\{i_1, ..., i_k\}$.

Now, let I_k be a random variable following a uniform distribution over all size-k subsets of the index set $\{1, ..., n\}$. Then $g(I_k)$ is called the minibatch stochastic gradient estimator with batch size k, and it is an unbiased estimator of the gradient with respect to the entire training set, since

$$\nabla_w \frac{1}{n} \sum_i^n \ell(x_i, y_i, w) = \mathbb{E}[g(I_k)].$$

That is, by picking one or more examples at random from the training data, and computing the average gradient only with respect to those examples, we get an unbiased estimate of the true gradient of the learning problem.

Methods which use only a sequence of (possibly stochastic) gradients to optimize an objective function are known as *first-order methods* and are often preferred in machine

learning and other high dimensional problems because of their linear complexity scaling with model size. This can also be looked at as doing a repeated regularized minimization of a linear approximation of the target function.

Given such a sequence of gradients $\{g_t\}$, we obtain a sequence of parameter vectors $\{\boldsymbol{\theta}_t\}$ using a variety of *update rules* (also known as *learning algorithms*). The most basic update rule is the plain SGD update rule,

$$\begin{aligned} w_{t+1} &= \arg\min_{w} \langle -\sum_{i}^{t} \eta_i g_i, w \rangle + \frac{1}{2}\|w\|_2^2 \\ &= w_t - \eta_t g_t. \end{aligned}$$

where $\{\eta_t\}$ is a sequence of step sizes or *learning rates*. The definition in terms of $\arg\min$ shows how this can be viewed as minimization of quadratically regularized linear approximations. This type of algorithm in the context of online learning is often referred to as *follow the regularized leader* (FTRL).

A slightly more complicated update is the Adagrad update rule [27], which scales the gradients with a (usually diagonal) matrix transform, adapting to the geometry of the loss function,

$$\begin{aligned} H_t &= \operatorname{diag}\left(\left\{\sum_{i}^{t} \eta_i g_i g_i^\top\right\}^{\frac{1}{2}}\right) \\ w_{t+1} &= w_t - \eta_t H_t^{-1} g_t \end{aligned}$$

A very popular adaptive method introduced later, Adam [47], uses an additional *momentum* term, which uses a moving average of gradients, and a "de-biasing" step, as well as adaptive learning per-coordinate learning rates like Adagrad. This update is more complicated to write down.

$$
\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\
V_t &= \beta_2 V_{t-1} + (1-\beta_2) g_t g_t^\top \\
\hat{m}_t &= \frac{1}{1-\beta_1^t} m_t \\
\hat{V}_t &= \frac{1}{1-\beta_2^t} V_t \\
w_t &= w_{t-1} - \eta \hat{V}_t^{-\frac{1}{2}} \hat{m}_t
\end{aligned}
$$

We use this algorithm in the majority of our experiments.

2.7 Applications

This thesis mostly concerns itself with applications in natural language processing, knowledge representation, and knowledge base completion. For readers who are unfamiliar with these fields, we will attempt here to give a slightly more detailed description of the major applications, topics, and tasks in this thesis, along with common approaches where appropriate, to avoid duplication in the later chapters.

2.7.1 Word Representations

Fitting with the recurring theme in this thesis, the goal of *word representation* (also called *word embedding*) is to map every word type w in some vocabulary $\mathcal{V}$ to a representation, such as a sparse or dense vector, a function, or a geometric object. Once mapped to this space, the representations of the words should capture certain semantic properties in their structure and relationships to one another. Similar words are meant to have similar representations, in ways that match up with human-sourced similarity scores such as those in Figure 2.2, but with much more coverage.

Here we provide a basic example of a prediction-based method for learning word representations, popularized in the word2vec library [59], and we use this framework to develop our own word representations in Chapter 3.

(money, bank, 8.5)
(psychology, Freud, 8.21)
(media, radio, 7.42)
(drug, abuse, 6.85)
(Mars, scientist, 5.63)
(cup, object, 3.69)
(professor, cucumber, 0.31)

Figure 2.2: Example human-sourced semantic similarity scores for words. Word representation learning can automatically capture these intuitive notions from large text corpora.

Remark. Here we are talking only about representation of word *types*, not word *tokens*. A word type is an element of the vocabulary, and a word token is a particular observation of that type. That is, the word "dog" when considered as a context-independent concept is a word type. The second word in the phrase "The dog chased the ball" is a word token. The distinction is analogous to the *class* vs. *instance* distinction in object-oriented programming. Later work has focused much more heavily on in-context word token representations (cite ELMO and BERT), which we do not explore in this thesis.

In unsupervised prediction-based learning of word representations, we observe a sequence of word tokens $\{w_1, ..., w_N\} \in \mathcal{V}$, which we process into a sequence of training examples (w_i, C_i), where w_i is the i'th word token, called the *focus* word token, and $C_i = w_{i-c:i+c,\setminus i}$ is a set of *context* taken from a window of c tokens behind and c tokens ahead of w_i, excluding w_i itself.

Given these training examples, we define an energy based model on focus words and contexts. The parameters of this model will include our learned representations. Finally, we train these parameters by minimizing a loss function on the training examples. For a concrete example, we will discuss the *skip-gram* word embedding model [59].

The skip-gram model uses the negative dot product of two types of word vectors as the energy, and several variants of a negative log likelihood as the loss function. The simplest

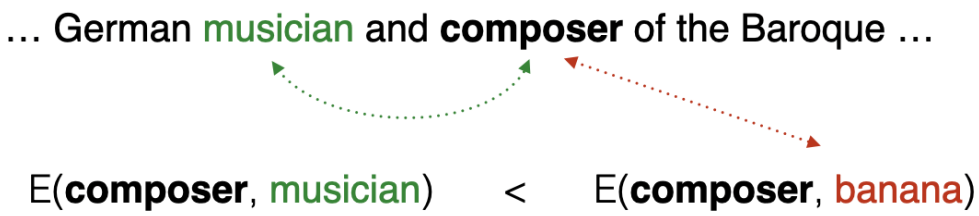


Figure 2.3: The canonical distributed word embedding algorithm. Energy is decreased between nearby words, and increased between randomly sampled word pairs.

model uses a softmax log likelihood over the entire vocabulary:

$$E(w_f, w_c) = -\langle v_{w_f}, v'_{w_c} \rangle$$
$$L(w_f, C) = -\sum_{w_c \in C} \frac{\exp(-E(w_f, w_c))}{\sum_{w' \in \mathcal{V}} \exp(-E(w_f, w'))}$$

Note that the word vectors associated with context tokens, v'_{w_c} come from a separate embedding matrix $V' \in \mathbb{R}^{|\mathcal{V}| \times d}$ than the word vectors associated with focus tokens, which come from $V \in \mathbb{R}^{|\mathcal{V}| \times d}$ The model is often approximated using a variant on *noise contrastive estimation* (cite NCE) called *negative sampling*, where the softmax over the vocabulary is replaced by a binary classification objective using observed context words for positive examples, and randomly sampled vocabulary words as negative examples, as shown in Figure 2.3. This loss function is defined as

$$L(w_f, C) = -\log \sigma(-E(w_f, w_c)) + \sum_{i=1}^{k} \log(1 - \sigma(-E(w_f, w_n^{(i)})))$$

where $w_n^{(i)}$ are sampled random elements of $\mathcal{V}$ distributed according to the negative sampling distribution $P(W_n)$, usually proportional to a fractional power of word frequency.

Upon training the skip-gram word embedding model, the context vectors V' are usually discarded, and the word vectors in the matrix V are used for representation, and can be used to solve word analogy problems through arithmetic, to find similar words, and in

downstream models to provide additional semantic information when labeled training data is scarce.

2.8 Matrix Factorization

Low-rank bilinear interaction models for collaborative filtering are another classic application of embedding-based learning. wherein each user and item are assigned embedding vectors u and v in $\mathbb{R}^d$, and the energy for a $(\text{user}, \text{item})$ pair is given by the scalar product

$$E(u, v) = -u^\top v.$$

In matrix form, this can be written as the energy function

$$E(\text{user}_i, \text{item}_j) = -(UV^\top)_{ij},$$

where $U \in \mathbb{R}^{|\text{users}| \times d}$ and $V \in \mathbb{R}^{|\text{items}| \times d}$, making $UV^\top$ a matrix in $\mathbb{R}^{|\text{users}| \times |\text{items}|}$ of rank d, the dimension of the embedding space.

There is a slight abuse of notation going on in these two definitions. In the former, the energy is defined on pairs of dense vectors $(u, v) \in \mathbb{R}^d \times \mathbb{R}^d = \mathcal{Y}$, with the embedding mapping having been already implicitly performed. In the latter, it is defined over pairs of discrete symbols $(\text{user}_i, \text{item}_j) \in \text{Users} \times \text{Items} = \mathcal{Y}$, and the embedding mapping takes the form of row-wise selection from the parameter matrices U and V. The latter presentation makes it clear that inference should only be performed over the finite set of symbols (rows), while the former makes the scalar-product nature of the energy function, and the form of the embeddings, clearer. We will sometimes switch between different presentations of an energy function, especially in the case of embeddings, when context makes the interpretation unambiguous.

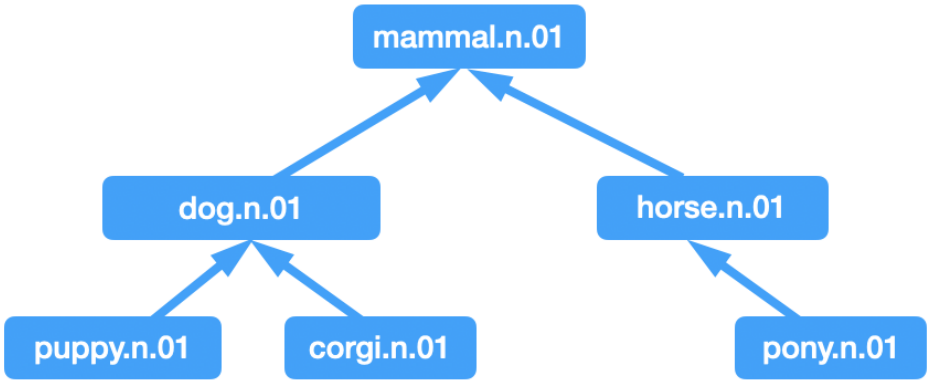


Figure 2.4: A fragment of the WordNet hypernymy graph.

Matrix factorization, like other forms of embedding-based learning, lend itself to certain types of geometric thinking. Since, when u's and v's all have the same norm, $-u^\top v$ is equivalent to $\|u - v\|_2^2$ (up to a constant), the energy function behaves much like a distance function. In this case, although the model may have been trained to score pairs (u, v), it makes sense to ask a question like "what u' is most similar to u?" and to answer it by searching through u's nearest neighbors. This sort of geometric reasoning about learned representations underlies much of the development of the models presented in this thesis.

2.8.1 Knowledge Graphs

Knowledge graphs, also called *knowledge bases* (KBs) are a term for groups of relations, ordered tuples, or alternately hypergraphs, representing database-style facts. The matrix factorization model for recommendation described in the previous section could be described as a model for predicting a single relation, (LIKES-MOVIE, User, Movie). These predictions are made by taking the model score (energy) for a (User, Movie) pair, given by the negative dot product $-u^\top v$, and predicting an edge only when the energy is below a certain threshold tuned on a development set.

An example multi-relational knowledge graph could contain relations such as BORN-IN, LOCATED-IN, WORKS-AT. Sample elements of this graph might look like (LOCATED-

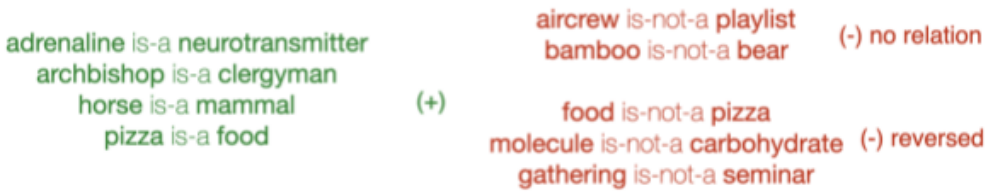


Figure 2.5: Example negative and positive entailment relations between words.

IN, Seattle, Washington) and (WORKS-AT, Bill Gates, Microsoft). These could be thought of as simply labeled edges in a graph, but relations with more than two arguments like (HAS-PARENTS, Gene, Bob, Linda) correspond to labeled hyperedges in a hypergraph. Matrix factorization and embedding models are often used for predicting this sort of multi-relational data as well.

One knowledge graph of particular interest in this thesis is the *hypernymy* graph of WordNet. This graph represents a transitive relation *Is-A* defined on different senses of dictionary words. A fragment of this is depicted in Figure 2.4 with transitive edges elided. We will discuss this example more in the next section. Furthermore, we go into greater detail about general transitive relations, especially partially ordered sets, in Chapter 4.

2.8.2 Entailment

A specific type of knowledge graph is an *entailment* graph, which is a transitive relation between interpreted as logical propositions. A generalization of this is often called *textual entailment*, which is a soft, probabilistic relaxation of the binary logical entailment relation. That is, an entailment is regarded as true if a human reading it would generally infer that it is true. *Implicature* is another word (more common in linguistics) for this soft, common-sense variety of entailment.

In this thesis we experiment with two main types of entailment. The first is between fragments of text in context, and the second is between dictionary definitions of words,

word-senses, or concepts. An example of different types of positive and negative entailment relations between words is depicted in Figure 2.5.

In the context of textual entailment between sentences or text fragments, the more general (*entailing*) text is known as simply the *text* or the *premise*, while the more specific text implied by the former is known as the *hypothesis*. Use of the term "hypothesis," with no other context implies that it is entailed by the premise, but in a context where we are trying to determine the truth of a candidate hypothesis, we might determine that candidate hypothesis to be either *entailed*, *neutral*, or *contradictory*.

In the context of entailment between words or concepts, the more general concept is known as the *hypernym*, and the more specific, the *hyponym*.

2.8.3 Density Estimation

Density estimation is the general task of modeling a probability distribution given a sample set of instantiations of the random variables. That is, we have some probabilistic model $P_{\text{model}}(X|\theta)$, parameterized by a set of weights θ, and we want to adjust these weights to model the true data distribution P_{data}, from which we have a bunch of samples x_i.

$$\theta^* = \arg\min_{\theta} -\sum_i \log P_{\text{model}}(x_i|\theta)$$

In this thesis, we focus on two types of density estimation. The first, is a generalization of the aforementioned knowledge graph models to probabilistic or *weighted* knowledge graphs, where instead of predicting the presence or absence of edges, we predict the probability of a (hyper)edge's existence. Matrix factorization models and other types of binary classifiers can be used for these density estimation tasks as well, with model scores or energies being treated as negative log probabilities of edge presence and calibrated under an appropriate loss function. An example task in this realm is weighted entailment, where

we determine the probability of a hypothesis being true given a certain premise. A more detailed description of this application is given in the next subsection.

The second type of density estimation problems we focus on are distributions over multiple binary variables, which we use to predict the probability of collections of attributes. For example, in a *market basket* task, we would like to predict which products, attributes, recommendations, or services are commonly used by the same user. So, if we were building such a model for a grocery store, we might give a high probability to a collection of products like $(\text{Chips}, \text{Cheese}, \text{Salsa}, \text{Guacamole})$, because someone is going to make nachos. While this can be cast as a weighted hypergraph model over a single relation with arbitrarily many arguments, the graph-based interpretation is often less useful in this case. Regardless, these problems are often handled with the same matrix-factorization based models as the previous case, where only pairwise probabilities are predicted and inference over clusterings is done in an ad-hoc manner.

2.8.4 Open-Domain Graphs

While matrix factorization serves as a good baseline model for many of the previous applications, in the case of (weighted) textual entailment between sentences or sentence fragments, the space of all possible sentences is far too large to assign each one its own embedding parameters. We call this case an *open-domain* task because the domain of all possible inputs is effectively infinite. What is needed in this case is a general method for mapping possibly variable-length sequences to a fixed size feature space.

While a general survey of neural network architecture is beyond the scope of this thesis, we include a description of the Long Short-Term Memory (LSTM) [44] recurrent neural network, a common method for performing this mapping.

The LSTM maps a sequence of input vectors $\{x_t\}$, such as word embeddings, to a sequence of output vectors $\{h_t\}$ using the following equations at each time step:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_i h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

where $\circ$ represents the elementwise (Hadamard) product, σ_g is the logistic sigmoid function, σ_h is the hyperbolic tangent, and c_0 and h_0 are initialized as zero vectors. The notation is suggestive of the purpose of each component. f_t, i_t and o_t are the *forget*, *input*, and *output gates*, taking on values between 0 and 1, which, combined with pointwise multiplication, allow them to control how much the c_t and h_t vectors incorporate different information sources at each time step. When used to represent an entire sequence of length T, with the final output vector h_T is used as the representation, or an average of the h_t across all time steps.

Given this vector, we simply treat it as an embedding of the sequence and can use normal embedding-based methods to make our predictions (e.g. predicting the probability of textual entailment using the negative dot product as a log-probability), i.e.

$$
P(\text{Hypothesis} = \text{true} | \text{Premise} = \text{true}) = \sigma(\langle \text{LSTM}_h(\text{Hypothesis}), \text{LSTM}_p(\text{Premise}) \rangle).
$$

CHAPTER 3

GAUSSIAN WORD REPRESENTATIONS

3.1 Introduction

In this chapter we introduce a method that moves beyond vector *point* representations to *potential functions* [1], or continuous densities in latent space. In particular we explore Gaussian function embeddings (with diagonal covariance), in which both means and variances are learned from data. Gaussians innately represent uncertainty, and provide a distance function per object. KL-divergence between Gaussian distributions is straightforward to calculate, naturally asymmetric, and has a geometric interpretation as an inclusion between families of ellipses.

There is a long line of previous work in mapping data cases to probability distributions, perhaps the most famous being radial basis functions (RBFs), used both in the kernel and neural network literature. We draw inspiration from this work to propose novel word embedding algorithms that embed words directly as Gaussian distributional potential functions in an infinite dimensional function space. This allows us to map word types not only to vectors but to soft regions in space, modeling uncertainty, inclusion, and entailment, as well as providing a rich geometry of the latent space.

After discussing related work and presenting our algorithms below we explore properties of our algorithms with multiple qualitative and quantitative evaluation on several real and synthetic datasets. We show that concept containment and specificity matches common intuition on examples concerning people, genres, foods, and others. We compare our embeddings to Skip-Gram on seven standard word similarity tasks, and evaluate the ability of our method to learn unsupervised lexical entailment. We also demonstrate that our

training method also supports new styles of supervised training that explicitly incorporate asymmetry into the objective.

We conclude with a short review of some of the extensions to Gaussian embeddings that have been proposed in the literature since the publication of the original work.

3.2 Related Work

Gaussian embeddings build on a long line of work on both distributed and distributional semantic word vectors, including distributional semantics, neural language models, and count-based language models.

Related work in probabilistic matrix factorization [63] embeds rows and columns as Gaussians, and some forms of this do provide each row and column with its own variance [76]. Given the parallels between embedding models and matrix factorization [25, 72, 54], this is relevant to our approach. However, these Bayesian methods apply Bayes' rule to observed data to infer the latent distributions, whereas our model works directly in the space of probability distributions and discriminatively trains them. This allows us to go beyond the Bayesian approach and use arbitrary (and even asymmetric) training criteria, and is more similar to methods that learn kernels [53] or function-valued neural networks such as mixture density networks [10].

Other work in multiplicative tensor factorization for word embeddings [49] and metric learning [90] learns some combinations of representations, clusters, and a distance metric jointly; however, it does not effectively learn a distance function per item. Fitting Gaussian mixture models on embeddings has been done in order to apply Fisher kernels to entire documents [22, 21]. Preliminary concurrent work from Kiyoshiyo et al. [50] describes a significantly different model similar to Bayesian matrix factorization, using a probabilistic Gaussian graphical model to define a distribution over pairs of words, and they lack quantitative experiments or evaluation.

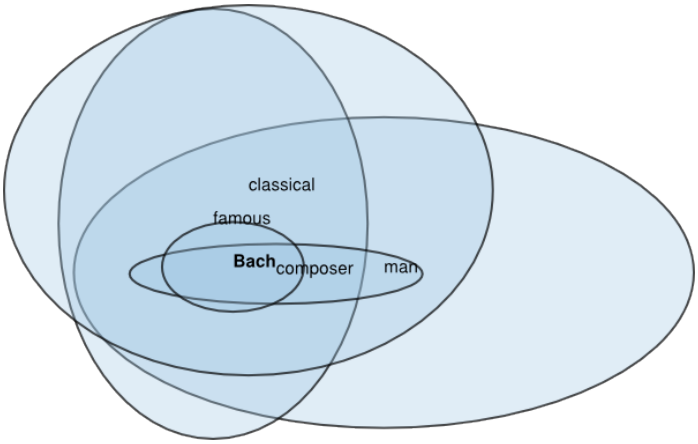


Figure 3.1: Learned diagonal variances, as used in evaluation (Section 3.6), for each word, with the first letter of each word indicating the position of its mean. We project onto generalized eigenvectors between the mixture means and variance of query word *Bach*. Nearby words to *Bach* are other composers e.g. *Mozart*, which lead to similar pictures.

In linguistic semantics, work on the *distributional inclusion hypothesis* [33], uses traditional count-based vectors to define regions in vector space [28] such that subordinate concepts are included in these regions. In fact, one strength of our proposed work is that we extend these intuitively appealing ideas (as well as the ability to use a variety of asymmetric distances between vectors) to the dense, low-dimensional distributed vectors that are now gaining popularity.

3.3 Background

As we describe in detail in Chapter 2, in unsupervised (or self-supervised) learning of word vectors, we observe a sequence of word tokens $w_1, ..., w_N$ each representing an instance of a word type (element of some vocabulary set $\mathcal{V}$), and the goal is to map each word type to a vector such that types that appear in similar contexts have similar vectors.

To achieve this, we define an energy function between word types and fit its parameters to minimize the energy between words and their observed contexts. The parameters corresponding to word types are then used as *word embeddings* or *word representations*.

In the word2vec Skip-Gram [59] word embedding model, the energy function takes the form of a negative dot product between the vectors of an observed word and an observed context $-\langle v_{w_f}, v'_{w_c} \rangle$. The loss function is a binary logistic regression negative log likelihood that treats the score of the focus word w_f and its observed context w_c as the score of a

positive example, and the score of the word and a randomly sampled context as the score of a negative example. For completeness, we reproduce this loss function here:

$$L(w_f, C) = -\log \sigma(-E(w_f, w_c)) + \sum_{i=1}^{k} \log(1 - \sigma(-E(w_f, w_n^{(i)}))),$$

where $w_n^{(i)}$ are sampled random elements of $\mathcal{V}$.

Backpropagating [75] this loss to the word vectors trains them to be predictive of their contexts, achieving the desired effect (words in similar contexts have similar vectors). In recent work, word2vec has been shown to be equivalent to factoring certain types of weighted pointwise mutual information matrices [54].

In the terminology, the contribution of this chapter is a pair of energy functions for training Gaussian distributions to represent word types.

3.4 Warmup: Empirical Covariances

Given a pre-trained set of word embeddings trained from contexts, there is a simple way to construct variances using the empirical variance of a word type's set of context vectors.

For a word w with N word vector sets $\{c(w)_i\}$ representing the words found in its contexts, and window size W, the empirical variance is

$$\Sigma_w = \frac{1}{NW} \sum_{i}^{N} \sum_{j}^{W} (c(w)_{ij} - w)(c(w)_{ij} - w)^\top$$

This is an estimator for the covariance of a distribution assuming that the mean is fixed at w. In practice, it is also necessary to add a small *ridge* term $\delta > 0$ to the diagonal of the matrix to regularize and avoid numerical problems when inverting.

However, in Section 3.6.2 we note that the distributions learned by this empirical estimator do not possess properties that we would want from Gaussian distributional embeddings, such as unsupervised entailment represented as inclusion between ellipsoids. By

discriminatively embedding our predictive vectors in the space of Gaussian distributions, we can improve this performance. Our models can learn certain forms of entailment during unsupervised training, as discussed in Section 3.6.2 and exemplified in Figure 3.1.

3.5 Energy-Based Learning of Gaussians

As discussed in Section 3.3, our architecture learns Gaussian distributional embeddings to predict words in context given the current word, and ranks these ahead of negatively sampled words. We present two energy functions to train these embeddings.

3.5.1 Symmetric Similarity: Expected Likelihood or Probability Product Kernel

While the dot product between two means of independent Gaussians is a perfectly valid measure of similarity (it is the expected dot product), it does not incorporate the covariances and would not enable us to gain any benefit from our density function representation.

The most logical next choice for a symmetric similarity function would be to take the inner product between the distributions themselves. Recall that for two (well-behaved) functions $f, g \in \mathbb{R}^n \to \mathbb{R}$, a standard choice of inner product is

$$\int_{x \in \mathbb{R}^n} f(x)g(x)dx$$

i.e. the continuous version of $\sum_i f_i g_i = \langle f, g \rangle$ for discrete vectors f and g.

This idea seems very natural, and indeed has appeared before – the idea of mapping data cases w into probability distributions (often over their contexts), and comparing them via integrals has a history under the name of the *expected likelihood* or *probability product kernel* [45].

Claim (Gaussian expected likelihood)**.** *Let $f_i(x_i; \mu_i, \Sigma_i)$ and $f_j(x_j; \mu_j, \Sigma_j)$ be probability density functions for two independent multivariate Gaussian random variables $X_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ and $X_j \sim \mathcal{N}(\mu_j, \Sigma_j)$. Then the* expected likelihood *has the explicit formula*

$$\textbf{\textit{EL}}(f_i, f_j) = \int_{x \in \mathbb{R}^n} f_i(x; \mu_i, \Sigma_i) f_j(x; \mu_j, \Sigma_j) dx = f_k(0; \mu_i - \mu_j, \Sigma_i + \Sigma_j), \tag{3.1}$$

where $f_k(0; \mu_i - \mu_j, \Sigma_i + \Sigma_j)$ is a Gaussian density function parameterized by $\mu = \mu_i - \mu_j$ and $\Sigma = \Sigma_i + \Sigma_j$, evaluated at 0.

Proof. Recall that the multivariate Gaussian PDF has the formula:

$$f(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{k}{2}} \det(\Sigma)^{\frac{1}{2}}} \exp(-\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu)).$$

First we introduce a new argument x' so that

$$\text{EL}(x', f_i, f_j) = \int_{x \in \mathbb{R}^n} f_i(x' - x; -\mu_i, \Sigma_i) f_j(x; \mu_j, \Sigma_j) dx,$$

and $E(0, f_i, f_j) = E(f_i, f_j)$, our original function of interest. Note that the density of the sum of two independent random variables is given by the convolution of the two densities.

$$f_{X+Y} = f_X * f_y.$$

So now we can see that the function $EL(x', f_i, f_j)$ is the density of the sum of the two random variables $-X_i$ and X_j, $X' = X_j - X_i$. The characteristic function of the sum of $-X_i$ and X_j is

$$\varphi_{X_j - X_i}(t) = \mathbb{E}[e^{it(X_j - X_i)}] = \mathbb{E}[e^{itX_j}]\mathbb{E}[e^{-itX_i}],$$

the product of the characteristic functions of X_j and $-X_i$.

The multivariate Gaussian distribution has the characteristic function

$$\varphi(t) = \exp(i\mu^\top t - \frac{1}{2}t^\top \Sigma t),$$

so the characteristic function of the sum is

$$\begin{aligned}\varphi_{X_j - X_i}(t) &= \exp(i\mu_j^\top t - \frac{1}{2}t^\top \Sigma_j t)\exp(-i\mu_i^\top t - \frac{1}{2}t^\top \Sigma_j t) \\ &= \exp(i(\mu_j - \mu_i)^\top t - \frac{1}{2}t^\top (\Sigma_j + \Sigma_i)t),\end{aligned}$$

which is just the characteristic function of a Gaussian distribution parameterized by $\mu = \mu_j - \mu_i$ and $\Sigma = \Sigma_i + \Sigma_j$. This means that

$$\text{EL}(x', f_i, f_j) = f_k(x'; \mu_j - \mu_i, \Sigma_i + \Sigma_j),$$

for a Gaussian density f_k, and

$$\text{EL}(f_i, f_j) = f_k(0; \mu_j - \mu_i, \Sigma_i + \Sigma_j) = f_k(0; \mu_i - \mu_j, \Sigma_i + \Sigma_j)$$

by symmetry. □

While the proof of this identity follows from the form of the characteristic function, it is a consequence of the broader fact that the Gaussian is a *stable distribution*, i.e. the sum of two Gaussian random variables is another Gaussian.

Given this similarity function between two Gaussians, and using our conventions of energy minimization, we define the expected likelihood energy as

$$E_{\text{EL}}(f_i, f_j) = -\log \text{EL}(f_i, f_j).$$

Since we aim to discriminatively train the weights of the energy function, and the expected likelihood is always positive, we work not with this quantity directly, but with its

logarithm. This has two motivations: firstly, we plan to use ranking loss, and ratios of densities and likelihoods are much more commonly worked with than differences — differences in probabilities are less interpretable than an odds ratio. Secondly, it is easier numerically, as otherwise the quantities can get exponentially small and harder to deal with.

The negative logarithm of $\mathrm{EL}(f_i, f_j)$ (in d dimensions) is

$$\begin{aligned}
&-\log f_k(0; \mu_i - \mu_j, \Sigma_i + \Sigma_j) \\
&\quad = \frac{1}{2}\log\det(\Sigma_i + \Sigma_j) + \frac{1}{2}(\mu_i - \mu_j)^\top(\Sigma_i + \Sigma_j)^{-1}(\mu_i - \mu_j) + \frac{d}{2}\log(2\pi).
\end{aligned}$$

Recalling that the gradient of the log determinant is $\frac{\partial}{\partial A}\log\det A = A^{-1}$, and the gradient $\frac{\partial}{\partial A}x^\top A^{-1}y = -A^{-\top}xy^\top A^{-\top}$ [69] we can take the gradient of this energy function with respect to the means μ and covariances Σ:

$$\begin{aligned}
\frac{\partial \log E(f_i, f_j)}{\partial \mu_i} &= -\frac{\partial \log E(f_i, f_j)}{\partial \mu_j} = \Delta_{ij} \\
\frac{\partial \log E(f_i, f_j)}{\partial \Sigma_i} &= \frac{\partial \log E(f_i, f_j)}{\partial \Sigma_j} = \frac{1}{2}((\Sigma_i + \Sigma_j)^{-1} - \Delta_{ij}\Delta_{ij}^\top) \\
&\text{where} \quad \Delta_{ij} = (\Sigma_i + \Sigma_j)^{-1}(\mu_i - \mu_j)
\end{aligned}$$

For diagonal and spherical covariances, these matrix inverses are trivial to compute, and even in the full-matrix case can be solved very efficiently for the small dimensionality common in embedding models. If the matrices have a low-rank plus diagonal structure, they can be computed and stored even more efficiently using the matrix inversion lemma.

This negative log-energy has an intuitive geometric interpretation as a regularized distance measure. Gaussians are measured as close to one another based on the distance between their means, as measured through the Mahalanobis distance defined by their joint inverse covariance. Recalling that $\log\det A + \text{const.}$ is equivalent to the log-volume of the ellipse spanned by the principle components of A, we can interpret this other term of the energy as a regularizer that prevents us from decreasing the distance by only increasing

joint variance. This combination pushes the means together while encouraging them to have more concentrated, sharply peaked distributions in order to have low energy.

3.5.2 Asymmetric Similarity: KL Divergence

In addition to the symmetric expected likelihood energy function, training Gaussian representations through KL-divergence is also a sensible choice of objective. This energy function can be used when training them to encode their context distributions, or when incorporating more explicit directional supervision, such as entailment from a knowledge base or WordNet. For asymmetric similarity, we minimize the KL divergence as our choice of energy function.

Recall the definition of KL divergence given in chapter 2,

$$D_{KL}(Y\|X) = \int_{\Omega'} f_X(\omega') \log(\frac{f_X(\omega')}{f_Y(\omega')}) d\omega', \tag{3.2}$$

between two distributions $P(X)$ and $P(Y)$ over the same domain (random variables sharing the same codomain).

Similar to the expected likelihood, the KL divergence also has a tractable closed form solution for Gaussians:

Claim (Gaussian KL-divergence)**.** *Let $f_i(x_i; \mu_i, \Sigma_i)$ and $f_j(x_j; \mu_j, \Sigma_j)$ be probability density functions for two independent multivariate Gaussian random variables $X_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ and $X_j \sim \mathcal{N}(\mu_j, \Sigma_j)$. Then the* KL-divergence *between X_i and X_j in terms of the densities f_i and f_j has the explicit formula*

$$\begin{aligned} E_{KL}(f_i, f_j) &= D_{KL}(X_i||X_j) \\ &= \frac{1}{2}\big(\log|\Sigma_j\Sigma_i^{-1}| - d + \mathrm{tr}(\Sigma_i\Sigma_j^{-1}) + (\mu_i - \mu_j)^\top \Sigma_j^{-1}(\mu_i - \mu_j)\big) \end{aligned} \tag{3.3}$$

Proof. First, we write the KL divergence in terms of expectations,

$$\begin{aligned}
D_{\mathrm{KL}}(X_i||X_j) =& \mathbb{E}[\log f_i(X_i)] - \mathbb{E}[\log f_j(X_i)] \\
=& \mathbb{E}[-\frac{1}{2}\big(\log|\Sigma_i| + (X_i-\mu_i)^\top\Sigma_i^{-1}(X_i-\mu_i) + k\log(2\pi)\big)] - \\
& \mathbb{E}[-\frac{1}{2}\big(\log|\Sigma_j| + (X_i-\mu_j)^\top\Sigma_j^{-1}(X_i-\mu_j) + k\log(2\pi)\big)] \\
=& -\frac{1}{2}\big(\log|\Sigma_i\Sigma_j^{-1}| + \mathbb{E}[(X_i-\mu_i)^\top\Sigma_i^{-1}(X_i-\mu_i)] - \\
& \mathbb{E}[(X_i-\mu_j)^\top\Sigma_j^{-1}(X_i-\mu_j)]\big).
\end{aligned}$$

The expectation of a quadratic form with respect to a random variable ϵ is given by

$$\begin{aligned}
\mathbb{E}[\epsilon^\top A\epsilon] &= \mathbb{E}[\big\langle A, \epsilon\epsilon^\top\big\rangle] \\
&= \big\langle A, \mathbb{E}[\epsilon\epsilon^\top]\big\rangle \\
&= \big\langle A, \Sigma + \mu\mu^\top\big\rangle \\
&= \mathrm{tr}(A\Sigma) + \mu^\top A\mu
\end{aligned}$$

where Σ and μ are the covariance and mean of ϵ. Plugging in this identity, we have

$$\begin{aligned}
& D_{\mathrm{KL}}(X_i||X_j) \\
&\quad = -\frac{1}{2}\big(\log|\Sigma_i\Sigma_j^{-1}| + \mathbb{E}[(X_i-\mu_i)^\top\Sigma_i^{-1}(X_i-\mu_i)] - \mathbb{E}[(X_i-\mu_j)^\top\Sigma_j^{-1}(X_i-\mu_j)]\big) \\
&\quad = -\frac{1}{2}\big(\log|\Sigma_i\Sigma_j^{-1}| + \mathrm{tr}(I) + 0^\top\Sigma_i 0 - \mathrm{tr}(\Sigma_i\Sigma_j^{-1}) - (\mu_i-\mu_j)^\top\Sigma_j^{-1}(\mu_i-\mu_j)\big) \\
&\quad = \frac{1}{2}\big(\log|\Sigma_j\Sigma_i^{-1}| - d + \mathrm{tr}(\Sigma_i\Sigma_j^{-1}) + (\mu_i-\mu_j)^\top\Sigma_j^{-1}(\mu_i-\mu_j)\big)
\end{aligned}$$

□

KL divergence is a natural energy function for representing entailment between concepts – a low KL divergence from x to y indicates that we can encode y easily as x, implying that y entails x. In the context of being used as an energy function on Gaussian

embeddings, we will denote it as E_{KL}. This can be more intuitively visualized and interpreted as a soft form of inclusion between the level sets of ellipsoids generated by the two Gaussians – if there is a relatively high expected log-likelihood ratio (negative KL), then most of the mass of y lies inside x.

Just as in the previous case, we can compute the gradients for this energy function in closed form:

$$\begin{aligned}
\frac{\partial E_{\text{KL}}(f_i, f_j)}{\partial \mu_i} &= -\frac{\partial E_{\text{KL}}(f_i, f_j)}{\partial \mu_j} = -\Delta'_{ij} \\
\frac{\partial E_{\text{KL}}(f_i, f_j)}{\partial \Sigma_i} &= \frac{1}{2}(\Sigma_i^{-1}\Sigma_j\Sigma_i^{-1} + \Delta'_{ij}\Delta_{ij}^{'\top} - \Sigma_i^{-1}) \\
\frac{\partial E_{\text{KL}}(f_i, f_j)}{\partial \Sigma_j} &= \frac{1}{2}(\Sigma_j^{-1} - \Sigma_i^{-1}) \\
&\text{where} \quad \Delta'_{ij} = \Sigma_i^{-1}(\mu_i - \mu_j)
\end{aligned}$$

using the fact that $\frac{\partial}{\partial A}\operatorname{tr}(X^\top A^{-1}Y) = -(A^{-1}YX^\top A^{-1})^\top$ and $\frac{\partial}{\partial A}\operatorname{tr}(XA) = X^\top$ [69].

3.5.3 Uncertainty of Inner Products

Another benefit of embedding objects as probability distributions is that we can look at the distribution of dot products between vectors drawn from two Gaussian representations. This distribution is not itself a one-dimensional Gaussian, but it has a finite mean and variance with a simple structure in the case where the two Gaussians are assumed independent [16]. For the distribution $P(Z)$ when $Z = X^\top Y$, we have

$$\begin{aligned}
\mu_z &= \mu_x^\top \mu_y \\
\Sigma_z &= \mu_x^\top \Sigma_x \mu_x + \mu_y^\top \Sigma_y \mu_y + \operatorname{tr}(\Sigma_x \Sigma_y)
\end{aligned}$$

this means we can find e.g. a lower or upper bound on the dot products of random samples from these distributions, that should hold some given percent of the time. Parameterizing

this energy by some number of standard deviations c, we can also get a range for the dot product as:

$$\mu_x^\top \mu_y \pm c\sqrt{\mu_x^\top \Sigma_x \mu_x + \mu_y^\top \Sigma_y \mu_y + \text{tr}(\Sigma_x \Sigma_y)}$$

We can choose c in a principled way using an (incorrect) Gaussian approximation, or more general concentration bounds such as Chebyshev's inequality.

3.5.4 Learning

To learn our model, we need to pick an energy function (EL or KL), a loss function (max-margin), and a set of positive and negative training pairs.

In our work, we use a slightly different loss function than Skip-Gram word2vec embeddings. Our energy functions take on a more limited range of values than do vector dot products, and their dynamic ranges depend in complex ways on the parameters. Both energies, negative EL and KL, can only take on positive values. The word2vec loss is a logistic binary classification loss, and it relies on being able to push the energy of positive pairs to negative infinity, and the energy of negative pairs to infinity. That is, this loss relies on the ability to adjust the energy surface in an absolute manner.

We can avoid this problem by using a loss function that only compares different energies rather than looking at their absolute magnitude — a ranking-based loss. We choose a max-margin ranking objective, similar to that used in Rank-SVM [46] or Wsabie [89], which pushes scores of positive pairs above negatives by a margin,

$$L_m(w_f, w_p, w_n) = \max(0, m - E(w_f, w_p) + E(w_f, w_n)),$$

where $m > 0$ is a positive margin, w is the focus word type, w_p is the positive context word type, and w_n is the negative context word type.

As the landscape is highly nonconvex, it is also helpful to add some regularization to the parameters.

We regularize the means and covariances differently, since they are different types of geometric objects. The means should not be allowed to grow too large, so we can add a simple hard constraint to the ℓ_2 norm:

$$\|\mu_i\|_2 \leq C, \ \ \forall i$$

However, the covariance matrices need to be kept positive definite as well as reasonably sized. This is achieved by adding a hard constraint that the eigenvalues λ_i lie within the hypercube $[m, M]^d$ for constants m and M.

$$mI \prec \Sigma_i \prec MI, \ \ \forall i$$

For diagonal covariances, this simply involves either applying the min or max function to each element of the diagonal to keep it within the hypercube, $\Sigma_{ii} \leftarrow \max(m, \min(M, \Sigma_{ii}))$.

Controlling the bottom eigenvalues of the covariance is especially important when training with expected likelihood, since the energy function includes a $\log\det$ term that can give very high scores to small covariances, dominating the rest of the energy.

We optimize the parameters using AdaGrad [27] and stochastic gradients in small mini-batches containing 20 sentences worth of tokens and contexts. More experimental details are given in the next section, along with parameter settings in Appendix A.1.

3.6 Evaluation

We evaluate the representation learning algorithms on several qualitative and quantitative tasks, including modeling asymmetric and linguistic relationships, uncertainty, and word similarity. All Gaussian experiments are conducted with 50-dimensional vectors, with diagonal variances except where noted otherwise. Unsupervised embeddings are learned on the concatenated ukWaC and WaCkypedia corpora [5], consisting of about 3 billion tokens. This matches the experimental setup used by [6], aside from leaving out the small

British National Corpus, which is not publicly available and contains only 100 million tokens. All word types that appear less than 100 times in the training set are dropped, leaving a vocabulary of approximately 280 thousand word types.

When training word2vec Skip-Gram embeddings for baselines, we follow the above training setup (50 dimensional embeddings), using our own implementation of word2vec to change as little as possible between the two models, only the loss function. We train both models with one pass over the data, using separate embeddings for the focus word types and context word types, 1 negative sample per positive example, and the same subsampling procedure as in the word2vec paper [59]. The only other difference between the two training regimes is that we use a smaller ℓ_2 regularization constraint when using the word2vec loss function, which improves performance vs. the diagonal Gaussian model which does better with "spikier" mean embeddings with larger norms (see the comment in Section 3.6.4). The original word2vec implementation uses no ℓ_2 constraint, but we saw better performance when including it in our training setup.

3.6.1 Specificity and Uncertainty of Embeddings

In Figure 3.2, we examine some of the 100 nearest neighbors of several query words as we sort from largest to smallest variance, as measured by determinant of the covariance matrix, using diagonal Gaussian embeddings. Note that more specific words, such as *joviality* and *electroclash* have smaller variance, while polysemous words or those denoting broader concepts have larger variances, such as *mix*, *mind*, and *graph*. This is not merely an artifact of higher frequency words getting more variance – when sorting by those words whose rank by frequency and rank by variance are most dissimilar, we see that genres with names like *chillout*, *avant*, and *shoegaze* overindex their variance compared to how frequent they are, since they appear in different contexts. Similarly, common emotion words like *sadness* and *sincerity* have less variance than their frequency would predict, since they

Query Word	Nearby Words, Descending Variance
rock	mix sound blue folk jazz rap avant hardcore chillout shoegaze powerpop electroclash
food	drink meal meat diet spice juice bacon soya gluten stevia
feeling	sense mind mood perception compassion sadness coldness sincerity perplexity diffidence joviality
algebra	theory graph equivalence finite predicate congruence topology quaternion symplectic homomorphism

Figure 3.2: Elements of the top 100 nearest neighbor sets for chosen query words, sorted by descending variance (as measured by determinant of covariance matrix). Note that less specific and more ambiguous words have greater variance.

have fairly fixed meanings. Another emotion word, *coldness*, is an uncommon word with a large variance due to its polysemy.

3.6.2 Entailment

The general NLP task of learning textual entailment is discussed in more detail in Chapter 2. As can be seen qualitatively in Figure 3.1, our embeddings can learn some forms of unsupervised entailment directly from the source data. We evaluate quantitatively on the Entailment dataset of [6]. Our setup is essentially the same as theirs but uses slightly less data, as mentioned in the beginning of this section. We evaluate with Average Precision and best F1 score. We include the best F1 score (by picking the optimal threshold at test) because this is used by [6], but we believe AP is better to demonstrate the correlation of various asymmetric and symmetric measures with the entailment data.

In Figure 3.3, we compare the performance of the model with variances learned jointly during embedding training by using the expected likelihood objective, with empirical variances gathered from contexts on pre-trained word2vec-style embeddings. We compare both diagonal (D) and spherical (S) variances, using both cosine similarity between means, and KL divergence. Baseline asymmetric measurements, such as the difference between the sizes of the two embeddings, did worse than the cosine. We see that KL divergence

Model	Test	Similarity	Best F1	AP
[6]	E	balAPinc	**75.1**	–
Empirical (D)	E	KL	70.05	.68
Empirical (D)	E	Cos	76.24	.71
Empirical (S)	E	KL	71.18	.69
Empirical (S)	E	Cos	76.24	.71
Learned (D)	E	KL	79.01	**.80**
Learned (D)	E	Cos	76.99	.73
Learned (S)	E	KL	**79.34**	.78
Learned (S)	E	Cos	77.36	.73

Figure 3.3: Entailment: We compare empirical and learned variances, both diagonal (D) and spherical (S). E is the dataset of [6]. Measures of similarity are symmetric (cosine between means) and asymmetric (KL) divergence for Gaussians. balAPinc is an asymmetric similarity measure specific to sparse, distributional count-based representations.

between the entailed and entailing word does not give good performance for the empirical variances, but beats the count-based balAPinc measure when used with learned variances.

For the baseline empirical model to achieve reasonable performance when using KL divergence, we regularized the covariance matrices, as the unregularized matrices had very small entries. We regularized the empirical covariance by adding a small ridge δ to the diagonal, which was tuned to maximize performance, to give the largest possible advantage to the baseline model. Interestingly, the empirical variances do worse with KL than the symmetric cosine similarity when predicting entailment. This appears to be because the empirically learned variances are so small that the choice is between either leaving them small, making it very difficult to have one Gaussian located "inside" another Gaussian, or regularizing so much that their discriminative power is washed out. Additionally, when examining the empirical variances, we noted that common words like "such," which receive very large variances in our learned model, have much smaller empirical variances relative to rarer words. A possible explanation is that the contrastive objective forces variances of commonly sampled words to spread out to avoid loss, while the empirical variance sees only "positive examples" and has no penalty for being close to many contexts at once.

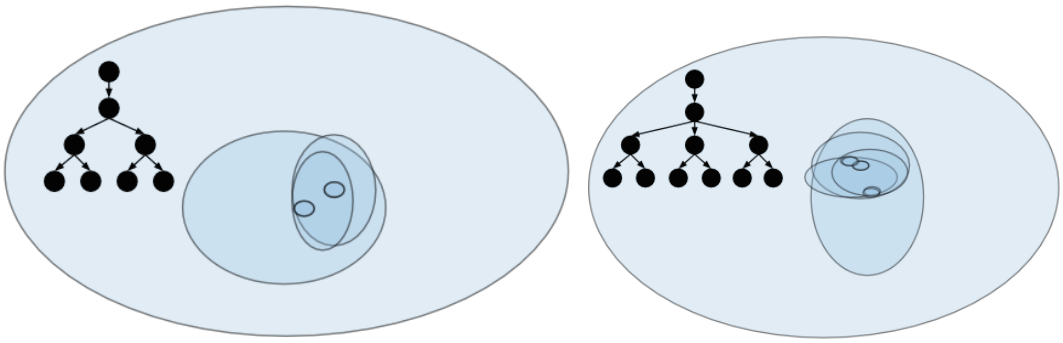


Figure 3.4: Synthetic experiments on embedding two simple hierarchies in two dimensions directly using KL divergence. The embedding model captures all of the hierarchical relationships present in the tree. Sibling leaves are pushed into overlapping areas by the objective function.

While these results indicate that we can do as well or better at unsupervised entailment than previous distributional semantic measures, we would like to move beyond purely unsupervised learning. Although certain forms of entailment can be learned in an unsupervised manner from distributional data, many entailing relationships are not present in the training text in the form of lexical substitutions that reflect the *is-a* relationship. For example, one might see phrases such as "look at that bird," "look at that eagle," "look at that dog," but rarely "look at that mammal." One appealing aspect of our models versus count-based ones is that they can be directly discriminatively trained to embed hierarchies.

3.6.3 Directly Learning Asymmetric Relationships

In Figure 3.4, we see the results of directly embedding simple tree hierarchies as Gaussians. We embed nodes as Gaussians with diagonal variances in two-dimensional space using gradient descent on the KL divergence between parents and children. We create a Gaussian for each node in the tree, and randomly initialize means. Negative contexts come from randomly sampled nodes that are neither ancestors nor descendants, while positive contexts come from ancestors or descendants using the appropriate directional KL divergence. Unlike our experiments with symmetric energy, we must use the same set of embeddings for nodes and contexts, or else the objective function will push the variances

to be unboundedly large. Our training process captures the hierarchical relationships, although leaf-level siblings are not differentiated from each other by this objective function. This is because out of all the negative examples that a leaf node can receive, only one will push it away from its sibling node.

3.6.4 Word Similarity Benchmarks

We evaluate the embeddings on seven different standard word similarity benchmarks [74, 81, 42, 61, 18, 91, 30]. A comparison to all of the state of the art word-embedding numbers for different dimensionalities as in [7] is out of the scope of this evaluation. However, we note that the overall performance of our 50-dimensional embeddings matches or beats reported numbers on these datasets for the 80-dimensional Skip-Gram vectors at *wordvectors.org* [29], as well as our own Skip-Gram implementation. Note that the numbers are not directly comparable since we use a much older version of Wikipedia (circa 2009) in our WaCkypedia dataset, but this should not give us an edge.

While it is good to sanity-check that our embedding algorithms can achieve standard measures of distributional quality, these experiments also let us compare the different types of variances (spherical and diagonal). We also compare against Skip-Gram embeddings with 100 latent dimensions, since our diagonal variances have 50 extra parameters.

We see that the embeddings with spherical covariances have an overall slight edge over the embeddings with diagonal covariances in this case, in a reversal from the entailment experiments. This could be due to the diagonal variance matrices making the embeddings more axis-aligned, making it harder to learn all the similarities and reducing model capacity. To test this theory, we plotted the absolute values of components of spherical and diagonal variance mean vectors on a q-q plot and noted a significant off-diagonal shift, indicating that diagonal variance embedding mean vectors have "spikier" distributions of components, indicating more axis-alignment.

Dataset	SG (50d)	SG (100d)	L/50/m/S	L/50/d/S	L/50/m/D	L/50/d/D
SimLex	29.39	31.13	**32.23**	29.84	**31.25**	30.50
WordSim	59.89	59.33	**65.49**	62.03	**62.12**	61.00
WordSim-S	69.86	70.19	**76.15**	73.92	**74.64**	72.79
WordSim-R	53.03	54.64	**58.96**	54.37	**54.44**	53.36
MEN	70.27	70.70	**71.31**	69.65	**71.30**	70.18
MC	63.96	66.76	**70.41**	69.17	67.01	**68.50**
RG	70.01	69.38	71.00	**74.76**	70.41	**77.00**
YP	39.34	35.76	41.50	**42.55**	36.05	**39.30**
Rel-122	49.14	51.26	**53.74**	51.09	52.28	**53.54**

Figure 3.5: Similarity: We evaluate our learned Gaussian embeddings (L) with spherical (S) and diagonal (D) variances, on several word similarity benchmarks, compared against standard Skip-Gram (SG) embeddings on the trained on the same dataset. We evaluate Gaussian embeddings with both cosine between means (m), and cosine between the distributions themselves (d) as defined by the expected likelihood inner product.

We also see that the distributions with diagonal variances benefit more from including the variance in the comparison (d) than the spherical variances. Generally, the data sets in which the cosine between distributions (d) outperforms cosine between means (m) are similar for both spherical and diagonal covariances. Using the cosine between distributions never helped when using empirical variances, so we do not include those numbers.

3.7 Later Work on Gaussian Embeddings

After the initial publication of Gaussian embeddings, other researchers were able to extend the model and apply it on some additional tasks and domains, a few representative samples of which we recap below.

3.7.1 Mixture Gaussian Embeddings

Athiwaratkun and Wilson [2] use embeddings parametrized as Gaussian mixture models to represent polysemous words with multiple distinct meanings, encoding the distribution of word w as

$$f_w(x) = \sum_{i=1}^{K} p_{w,i} \mathcal{N}(x; \mu_{w,i}, \Sigma_{w,i})$$

They use the expected likelihood / probability product kernel as an energy between word distributions in order to train the embeddings, which has a closed form in the case of Gaussian mixture models,

$$\begin{aligned}\int f(x)g(x)dx &= \sum_{i=1}^{K}\sum_{j=1}^{K} p_{f,i}p_{g,j} \int \mathcal{N}(x; \mu_{f,i}, \Sigma_{f,i})\mathcal{N}(x; \mu_{g,j}, \Sigma_{g,j})dx \\ &= \sum_{i=1}^{K}\sum_{j=1}^{K} p_{f,i}p_{g,j}\mathcal{N}(0; \mu_{f,i} - \mu_{g,j}, \Sigma_{f,i} + \Sigma_{g,j}).\end{aligned}$$

They evaluate this model on multi-sense word similarity datasets and find that it gives improved performance, with quantitative analysis showing reductions in variance of representations for polysemous words, as expected.

3.7.2 Supervised Gaussian Knowledge Graph Embedding

He et al. [40] apply Gaussians to knowledge graph embedding, modifying the popular Trans-E method [13] to use Gaussian representations of relations and entities. Given three Gaussians representing the head, the tail, and the relation, $(\mathcal{N}_h, \mathcal{N}_t, \mathcal{N}_r)$, we define the distribution of the difference between head and tail

$$\mathcal{N}_e(\mu_h - \mu_t, \Sigma_h + \Sigma_t)$$

and then calculate the KL divergence between that and the relation distribution,

$$\int \mathcal{N}(x; \mu_r, \Sigma_r) \log \frac{\mathcal{N}(x; \mu_h - \mu_t, \Sigma_h + \Sigma_t)}{\mathcal{N}(x; \mu_r, \Sigma_r)} dx$$

The authors achieve strong results on WN18 and WN11 WordNet-derived datasets for link-prediction and triplet classification, as well as the FB13 and FB15k knowledge graph completion datasets, derived from Freebase. Both datasets describe multirelational graphs, and

the authors claim that the density-based representation specifically enables them to better handle many-to-one and one-to-many relations.

3.7.3 Unsupervised Gaussian Graph Embedding

Bojchevski and Günnemann [12] introduce a methodology to learn unsupervised Gaussian representations of graphs, graph2gauss, in which the geometry of the learned representations can give information about the diversity of neighboring nodes.

The embeddings are learned using a personalized ranking approach applied to the KL divergence energy function between densities associated to each node. The objective seeks to rank each node in a k-hop neighborhood as closer in terms of KL-divergence than all of the nodes in a $k+1$-hop neighborhood.

They optimize the following objective between nodes, with $E_{ij} = D_{\mathrm{KL}}(f_j|f_i)$ representing the KL divergence from f_i to f_j, and N representing the neighborhood of a node:

$$\mathcal{L} = \sum_{i} \sum_{k<l} \sum_{j_k \in N_{ik}} \sum_{j_l \in N_{il}} \left(E_{ij_k}^2 + \exp(-E_{ij_l}) \right).$$

They apply the model to link prediction, as well as node classification on learned embeddings, and find especially that the uncertainty of the Gaussian representation helps to predict neighborhood diversity (the number of classes in a given neighborhood of a node).

3.8 Conclusion

In this chapter we introduced a method to embed word types into the space of Gaussian distributions, and learn the embeddings directly in that space. This allows us to represent words not as low-dimensional vectors, but as densities over a latent space, directly representing notions of uncertainty and enabling a richer geometry in the embedded space. We demonstrated the effectiveness of these embeddings on a linguistic task requiring asymmetric comparisons, as well as standard word similarity benchmarks, learning of synthetic hierarchies, and several qualitative examinations.

While the initial work focused on the unsupervised setting, synthetic experiments on toy hierarchies, as well as numerous follow-up work by other researchers on applications to supervised graph learning problems, point towards the suitability of geometric representation learning for supervised modeling of ordered and hierarchical structures. However, while the Gaussian embeddings hint at this sort of hierarchical or graph-like structure, there are problems when attempting to encode those structures directly as Gaussians. For example, Gaussian embeddings induce ellipsoidal regions in space, but the intersection of two such regions is not itself an ellipsoid, and it is difficult to compute its volume. In the following chapters, we will explore the mathematics of order theory, especially in vector lattices, its application to geometric representation learning for hierarchies, and the way it motivates and shapes follow-on work.

CHAPTER 4

LATTICES AND POSETS

4.1 Introduction

Given that Gaussian embeddings were inspired by the desire to find an embedding format that naturally lends itself to asymmetric comparisons, such as those inherent in *A-causes-B* or *A-is-a-B* transitive, entailment-style relations, it is appropriate that we now turn to examine the formalism of *order theory*. Order theory formalizes certain types of asymmetric relationships between objects in ways that prove valuable for designing geometric embedding models, as well as tying them into probability theory in interesting ways.

4.1.1 Partial Orders and Lattices

A non-strict *partially ordered set* (*poset*) is a pair $P, \preceq$, where P is a set, and $\preceq$ is a binary relation. For all $a, b, c \in P$,

Reflexivity: $a \preceq a$

Antisymmetry: $a \preceq b \preceq a$ implies $a = b$

Transitivity: $a \preceq b \preceq c$ implies $a \preceq c$

This generalizes the standard concept of a totally ordered set to allow some elements to be incomparable. Posets provide a good formalism for the kind of acyclic directed graph data found in many knowledge bases with transitive relations.

A *lattice* is a poset where any subset of elements has a single unique least upper bound, and greatest lower bound. In a *bounded lattice*, the set P contains two additional elements,

$\top$ (*top*), and $\bot$ (*bottom*), which denote the least upper bound and greatest lower bound of the entire set.

A lattice is equipped with two binary operations, $\vee$ (*join*), and $\wedge$ (*meet*). $a \vee b$ denotes the least upper bound of $a, b \in P$, and $a \wedge b$ denotes their greatest lower bound. A bounded lattice must satisfy these properties:

Idempotence: $a \wedge a = a \vee a = a$

Commutativity: $a \wedge b = b \wedge a$ and $a \vee b = b \vee a$

Associativity: $a \wedge b \wedge c = a \wedge (b \wedge c)$ and $(a \vee b \vee c) = a \vee (b \vee c)$

Absorption: $a \vee (a \wedge b) = a$ and $a \wedge (a \vee b) = a$

Bounded: $\bot \preceq a \preceq \top$

Note that the extended real numbers, $\mathbb{R} \cup \{-\infty, \infty\}$, form a bounded lattice (and in fact, a totally ordered set) under the $\min$ and $\max$ operations as the meet ($\wedge$) and join ($\vee$) operations. So do sets partially ordered by inclusion, with $\cap$ and $\cup$ as $\wedge$ and $\vee$. Thinking of these special cases gives the intuition for the fourth property, *absorption*.

The $\wedge$ and $\vee$ operations can be swapped, along with reversing the poset relation $\preceq$, to give a valid lattice, called the *dual lattice*. In the real numbers this just corresponds to a sign change. A *semilattice* has only a meet or join, but not both.

4.1.2 Order Embeddings (Vector Lattice)

Vendrov et al. [84] introduced a method for embedding partially ordered sets and a task, *partial order completion*, an abstract term for applications like hypernym or entailment prediction, and general learning of transitive relations (see a discussion of some of these tasks in 3.3). The goal is to learn a mapping from the partially-ordered data domain to some other partially-ordered space that will enable generalization.

Definition 19 (Order embedding)**.** Vendrov et al. [84]

A function $f : (S, \preceq_S) \to (T, \preceq_T)$ is an *order-embedding* if for all $a, b \in S$

$$a \preceq_S b \iff f(a) \preceq_T f(b)$$

Remark. While the term "order-embedding" is useful for our machine learning applications by analogy to other types of embeddings, the above construction could also simply be viewed as a standard injective poset homomorphism.

They choose T to be a vector space, and the order $\preceq_T$ to be based on the *reverse product order* on $\mathbb{R}^n_+$, which specifies for $(u, v) \in \mathbb{R}^n_+$

$$u \preceq v \iff \forall i \in \{1..n\},\ \ u_i \geq v_i, \tag{4.1}$$

so an embedding is below another in the hierarchy if all of the coordinates are larger, and 0 provides a top element. An example of this can be seen in Figure 4.1b. We will call this particular order embedding into a vector space with this product ordering *standard order embeddings*, or just *order embeddings* when the context is clear.

This ordering is optimized from examples of ordered elements and negative samples via a max-margin loss. Concretely, in the formalism of energy-based learning, Vendrov et al. [84] define an asymmetric energy function between pairs of embeddings,

$$E(u, v) = \| \max(0, v - u)\|_2^2. \tag{4.2}$$

Remark. Although the norm used in the energy function 4.2 is not explicitly specified in Vendrov et al. [84], an examination of the released code shows it to be the 2-norm.

Using this energy function, they define a loss,

$$\sum_{(u,v)\in P} E(u,v) + \sum_{(u',v')\in N} \max(0, \alpha - E(u',v')),$$

where P and N are the sets of vectors corresponding to positive and negative examples, and α is a margin hyperparameter.

Although Vendrov et al. [84] do not explicitly discuss it, their model does not just capture partial orderings, but is a standard construction of a vector (Hilbert) lattice, in which the operations of meet and join can be defined as taking the pointwise maximum and minimum of two vectors, respectively [92]. This observation is also used in [55] to generate extra constraints for training order embeddings.

As noted in the original work, these single point embeddings can be thought of as regions, i.e. the axis-aligned cone extending out from the vector towards infinity. All concepts "entailed" by a given concept must lie in this cone. We will make this precise.

Definition 20 (Cone lattice)**.** An (axis-aligned) *cone lattice* is constructed using a subset $S \subseteq \mathbb{R}^d$ along with a set of axis-aligned cones $T = \{\text{Cone}(X_i)\}_{i=1}^n$ in S, parametrized by origin vectors $\{x^{i,\vee}\}$,

$$x^{i,\vee} \in S \subseteq \mathbb{R}^d, \;\; 1 \le i \le n$$
$$\text{Cone}(X_i) = \{\, u \mid u_j \ge x_j^{i,\vee}, 1 \le i \le d, u \in S \,\}, \;\; 1 \le i \le n.$$

Define an ordering on the set of cones T using the subset relation,

$$\text{Cone}(A) \preceq \text{Cone}(B) \iff \text{Cone}(A) \subseteq \text{Cone}(B),$$

so $(T, \preceq)$ forms a poset. We can add a natural meet operation defined using set intersection,

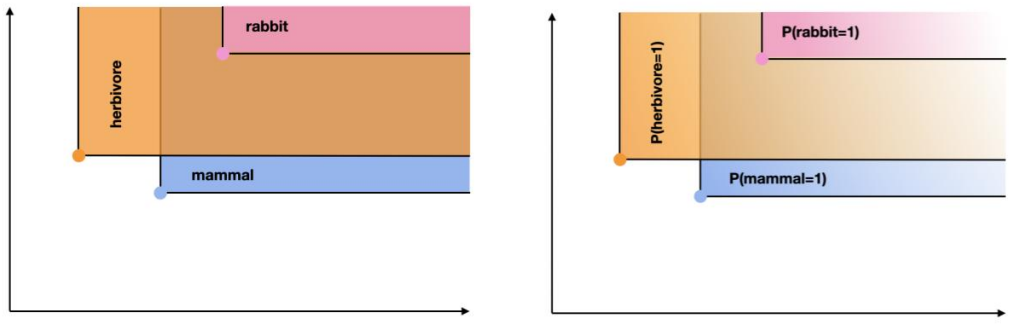
$$\begin{aligned}\text{Cone}(A) \wedge \text{Cone}(B) &= \text{Cone}(A) \cap \text{Cone}(B) = \text{Cone}(C) \\ &= \{\, u \mid u_i \geq \max(a_i^\vee, b_i^\vee) = c_i^\vee, 1 \leq i \leq d, u \in S \,\},\end{aligned}$$

so the origin vector for the meet is the pointwise maximum of the two origin vectors of the operands. The resulting structure $(T, \wedge)$ forms a meet semilattice. Finally, we can add a join operation by finding the least upper bounding cone,

$$\begin{aligned}\text{Cone}(A) \vee \text{Cone}(B) &= \text{Cone}(C) \\ &= \{\, u \mid u_i \geq \min(a_i^\vee, b_i^\vee) = c_i^\vee, 1 \leq i \leq d, u \in S \,\},\end{aligned}$$

so the origin vector for the join is the pointwise minimum of the two origin vectors of the operands. The triplet $(T, \wedge, \vee)$ is the axis-aligned cone lattice.

Remark. The original definition of standard order embeddings is given as a relation between vectors, but sometimes it is more convenient to think of the order embedding as a cone lattice. This is an isomorphic poset, and when the context is clear we will sometimes treat standard order embeddings explicitly as cones rather than vectors.



(a) Order embeddings (b) Probabilistic order embeddings

Figure 4.1: Example (standard and probabilistic) order embeddings in two dimensions. In the probabilistic case, the area of each cone corresponds to a Bernoulli probability.

4.2 Probabilistic Order Embeddings

Lai and Hockenmaier [52] built on the "region" idea to derive a probabilistic formulation (which we will refer to as POE) to model entailment probabilities in a consistent, hierarchical way, demonstrated in Figure 4.1b.

Noting that all of OE's regions obviously have the same infinite area under the standard (Lebesgue) measure of $\mathbb{R}^n_+$, they propose a probabilistic interpretation. For each poset element $a \in S$ with associated standard order embedding vector $f(a) \in \mathbb{R}^n_+$, they introduce a new Bernoulli random variable, A. The Bernoulli probabilities $P(A = 1)$ or joint Bernoulli probabilities $P(A = 1, B = 1)$, for variables (A, B) with $a^\vee = f(a)$ and $b^\vee = f(b)$, are given by the volume of these associated cones under the exponential measure:

$$P(A = 1) = \exp(-\sum_i a_i^\vee) = \int_{u \preceq a^\vee} \exp(-\|u\|_1) du$$

$$P(A = 1, B = 1) = \exp(-\|\max(a_i^\vee, b_i^\vee)\|_1)$$

where $\preceq$ is defined as in equation 4.1, and replacing sums with ℓ_1 norms for brevity since all coordinates are positive. Note that $P(A = 1, B = 1)$ can also be looked at as $P(C = 1)$ for some C associated to a cone $u \wedge v$, since the meet of two vectors is simply the intersection of their area cones. While having the intuition of measuring the areas of cones, this also automatically gives a valid probability distribution over concepts since this is just the product likelihood under a coordinatewise exponential distribution.

We can generalize this to non-exponential measures with the following formal definition of a probabilistic order embedding.

Definition 21 (Cone Probability Model and Probabilistic Order Embedding)**.** Let (Ω_{Cone}, $\mathcal{E}$, P_{Cone}) be a probability space where where $\Omega_{\text{Cone}} \subseteq \mathbb{R}^d$. Let $\{\text{Cone}(X_i)\}_{i=1}^n$ be a set of axis-aligned cones in Ω_{Cone}, as in the cone lattice (Definition 20),

$$x^{i,\vee} \in \Omega_{\text{Cone}} \subseteq \mathbb{R}^d$$

$$\text{Cone}(X_i) = \{\, u \mid u_j \geq x_j^{i,\vee}, 1 \leq i \leq d, u \in \Omega_{\text{Cone}} \,\}.$$

Further, define a set of indicator random variables $\{X_i\}$ for these cone-shaped events,

$$X_i : \Omega_{\text{Cone}} \to \{0,1\} = \mathbb{1}_{\text{Cone}(X_i)}, \;\; 1 \leq i \leq n$$

$$X_i^{-1}(\{1\}) = \text{Cone}(X_i), \;\; 1 \leq i \leq n.$$

We will denote the distribution of $X_1, ..., X_N$ as $P(X_1, ..., X_N)$, dropping the Cone subscript on P, to make it clear we are talking about the discrete pushforward measure on $\{0,1\}^N$ induced by P_{Cone} and not the measure P_{Cone} defined on a subset of $\mathbb{R}^d$. We will call this probability space and associated random variables a *cone probability model*.

If each $X_i = f(a)$ for some mapping $f : S \to (\Omega_{\text{Cone}} \to \{0,1\})$ from a finite set S to random variables in the cone probability model, we call this a *probabilistic order embedding* of S.

Remark. Note that the above definition implies that joint probabilities over multiple variables can be calculated by inclusion-exclusion with set intersection and complement over $\text{Cone}(X_i)$, e.g.

$$P(X_1 = 1, X_2 = 1, X_3 = 0) = P_{\text{Cone}}(\text{Cone}(X_1) \cap \text{Cone}(X_2) \cap \text{Cone}(X_3)^{\mathsf{c}}).$$

Example. Using the formalism of definition (21), the probabilistic order embedding of Lai and Hockenmaier [52] is a cone model $(\Omega_{\text{Cone}}, \mathcal{E}, P_{\text{Cone}})$ with $\Omega_{\text{Cone}} = \mathbb{R}_+^d$ and $P_{\text{Cone}}(E) = \int_{\omega \in E} \exp(-\|\omega\|_1) d\omega$, along with the standard order embedding mapping from domain objects to cones. We will refer to this model as *standard probabilistic order embeddings*, or just *probabilistic order embeddings* when the context is clear.

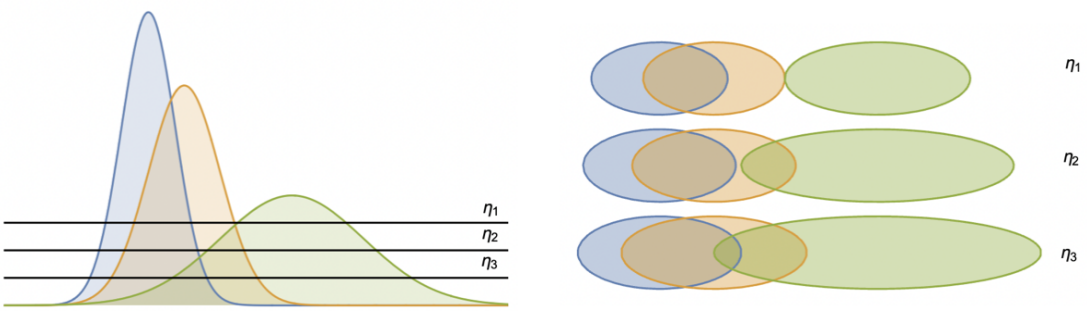


Figure 4.2: Converting densities to regions in hierarchical density order embeddings.

4.3 Hierarchical Density Order Embeddings

Athiwaratkun and Wilson [3] introduced the *hierarchical density order embeddings*, an approach to Gaussian embeddings that incorporates some of the lessons from order theory introduced by order embeddings. First, they note that a partial order on (probability) densities can be induced by looking at the set of values where the function has greater than a certain amount of density. That is, looking at certain level sets of densities, along with their interiors. Using this idea, densities can be endowed with a partial order $\preceq_\eta$, defined as

$$f \preceq_\eta g \iff \{x : f(x) > \eta\} \subseteq \{x : g(x) > \eta\}$$

for a threshold $\eta \geq 0$. This ordering is depicted in Figure 4.2. This is an elegant way to endow density functions with a partial order that uses our intuitions about level sets, considering that the normal vector lattice ordering (a function's graph being strictly above or below another function's graph) trivially makes all normalized density functions incomparable.

Remark. This definition does not technically yield a partial order on all arbitrary densities, but a *preorder*, as it does not necessarily satisfy antisymmetry (though it does for Gaussians and most other parametric distributions).

The next idea comes from the fact that the asymmetric divergences used to compare probability distributions (such as KL- and α- divergences) penalize deviation in one direction or another based on a certain rough notion of inclusion. That is, when finding the KL divergence from p to q, we have a shorter distance if the level sets of p are "inside" the level sets of q, than if the opposite is true. This can be understood by looking at the interpretation of KL divergence as encoding data from a distribution based on a code derived from a different distribution, where the inclusive ("mode-seeking") behavior leads to less distance. While the partial order defined above is theoretically pleasing, it is difficult to calculate for many distributions, including simple Gaussians. This inspires a soft surrogate to the original order. Using the intuition that divergence is higher in one direction than the other when level sets are included, the soft approximation simply uses a thresholded divergence,

$$d_\gamma(f, g) = \max(0, D(f\|g) - \gamma)$$

for some threshold $\gamma > 0$. The authors show that this modified divergence/ordering function gives state-of-the-art results when using Gaussian embeddings to model supervised partial ordering tasks such as WordNet.

4.4 Hyperbolic Embeddings

Analogs to both standard embedding models, and order-embedding-like cone embeddings, have also been defined in *hyperbolic space*. Hyperbolic space is a generalization of Euclidean space $\mathbb{R}^d$ to a manifold of constant negative curvature, a geometric property which enables it to more accurately embed tree-like graphs (in the sense of distance preservation). While hyperbolic space is not a vector space because it is not closed under addition, and thus not a vector lattice, one can define a closely related object called a *gyrovector space*.

4.4.1 Poincaré embeddings

While hyperbolic embeddings of data were previously used for modeling network topology and shortest-path routing [11, 51], the first such work to do so in the context of representation learning for the type of applications studied in this thesis (e.g. learning and representing latent entailment hierarchies) was the *Poincaré embeddings* of Nickel and Kiela [67]. These embeddings represent each variable with a point inside the *Poincaré ball*, a representation of hyperbolic space embedded in Euclidean space such that the hyperbolic distance between points increases exponentially as a function of the Euclidean distance from the origin. Specifically, the hyperbolic distance between two points u and v represented in their Euclidean coordinates within the Poincaré ball of radius 1, is given as

$$\text{arcosh}(1 + 2\frac{\|u - v\|^2}{(1 - \|u\|^2)(1 - \|v\|^2)}).$$

The authors leverage this to automatically learn concept hierarchies by noting that highly specific concepts are distant (conceptually) from many other concepts, while more general concepts are closer. This means what when training a graph embedding representing conceptual distance, embeddings of more generic concepts will organize near the center of the Poincaré ball, and more specific ones will move to the periphery. An ad-hoc asymmetric similarity function is defined as

$$\text{is-a}(u, v) = -(1 + \alpha(\|v\| - \|u\|))d(u, v),$$

where $\alpha > 0$ is a hyperparameter, and d is the hyperbolic distance. This will give a high score to is-a(u, v) and predict an entailment link if two concepts are close together (the $d(u, v)$ term), and one of them is significantly closer to the origin, indicating it is more generic.

4.4.2 Hyperbolic entailment cones

While Poincaré embeddings use an ad-hoc scoring function, along with the geometry of hyperbolic space and its Euclidean representation in the Poincaré ball, to predict hierarchy and entailment links between concepts, a more principled approach is possible. *Hyperbolic entailment cones* [32] fuse the graph-embedding benefits of hyperbolic space with the idea of representing concepts as cone-shaped regions, as in order embeddings. They note that when considering any two axis-aligned cones in d-dimensional Euclidean space (e.g. order embeddings), the volume of each that is disjoint from the other must be bounded in all but d of the cones. This implies difficulties in representing very widely branching hierarchies using cone embeddings. They define a type of cone in hyperbolic space that does not suffer from this issue, due to the exponentially expanding volume.

While a convex cone S in Euclidean space is defined as a set closed under positive linear combinations,

$$u, v \in S \implies \alpha v_1 + \beta v_2 \in S, \quad \alpha, \beta \geq 0,$$

hyperbolic entailment cones generalize this to hyperbolic space by using the exponential mapping from the tangent space. For a manifold $\mathcal{M}$ and a point on the manifold x, the *tangent bundle* is an association of each such point with its *tangent space* $T_x\mathcal{M}$. The tangent space at x is a vector space which intuitively, when considering that manifold as an embedded surface in Euclidean space, can be represented as the tangent (hyper)plane at that point.

For a manifold $\mathcal{M}$ and a point on the manifold x, the exponential map $\exp_x(v) : T_x\mathcal{M} \to \mathcal{M}$ takes a vector in the tangent space at x within a certain radius of the origin (infinite in the case of hyperbolic space) and maps it to the point on the manifold corresponding to a geodesic of the same length starting off in the same direction at the starting point x. Hyperbolic entailment cones generalize convex cones to hyperbolic space by map-

ping a standard convex cone S in the Euclidean tangent space to its image $\exp(S)$ under the exponential map.

They specifically choose a restricted family of cones in order to preserve closed form expressions for relevant quantities, choosing cones which are defined symmetrically and entirely by their apex point x. The cones can face in any direction as long as they are symmetric around a line from the origin through x, their apertures (angle at the peak) are a function only of the magnitude of the apex point, and set containment gives them a partial order such that $\|x'\| \geq \|x\|$ implies that $\operatorname{Cone} x' \subseteq \operatorname{Cone} x$.

Ganea et al. [32] show that in hyperbolic space, the angle between two (geodesic) line segments xy and $0x$ in the Poincaré ball is

$$\Xi(x, y) = \arccos\Big(\frac{\langle x, y\rangle(1 + \|\|^2) - \|x\|^2(1 + \|y\|^2)}{\|x\|\|x - y\|\sqrt{1 + \|x\|^2\|y\|^2 - 2\langle x, y\rangle}}\Big),$$

and the entailment cone with apex x in the Poincaré ball $\mathbb{D}^d$ has the closed form expression

$$C_x = \{\, y \in \mathbb{D}^d \mid \Xi(x, y) \leq \arcsin(K\frac{1 - \|x\|^2}{\|x\|}) \,\},$$

where K is a small constant related to the constraints, often set to 0.1.

The authors train these cones in a supervised manner from entailment hierarchies to contain one another by using a max-margin objective on the angles between apex points.

4.5 Conclusion

In this chapter we presented several concepts from order theory and showed their application to geometric representation learning. However, these models are not without weaknesses. The probabilistic order embeddings presented in this chapter are unable to model negatively correlated variables, as discussed in Lai and Hockenmaier [52]. In the next chapter we will introduce another probabilistic, lattice-based model, the *box lattice*,

that builds on these vector lattice based methods to increase expressivity and solve this correlation problem.

CHAPTER 5

BOX EMBEDDINGS

5.1 Introduction

In this chapter, we present a method for representing data in terms of boxes (axis-aligned hyperrectangles). This geometric representation learning algorithm explicitly parameterizes regions in latent space in order to learn a complex probabilistic interaction model for binary data.

In the introductory chapter, we discussed the similarity between learning in certain embedding models, and structured prediction. While learning the set of embeddings, the embeddings obey structural constraints in their relationships to one another, and predictions derived from those embeddings obey related constraints. While the structured prediction analogy applies best to Order Embeddings (OE), which embeds consistent partial orders, other region- and density-based representations have been proposed for the express purpose of inducing a bias towards asymmetric relationships. For example, the Gaussian Embedding (GE) model discussed in Chapter 3 aims to represent the asymmetry and uncertainty in an object's relations and attributes by means of uncertainty in the representation. However, while the space of representations is a manifold of probability distributions, the model is not truly probabilistic in that it does not model asymmetries and relations in terms of probabilities, but in terms of asymmetric comparison functions such as the originally proposed KL divergence and the recently proposed *thresholded divergences* [3].

Probabilistic models are especially compelling for modeling ontologies, entailment graphs, and knowledge graphs. Their desirable properties include an ability to remain consistent in the presence of noisy data, suitability towards semi-supervised training using

the expectations and uncertain labels present in these large-scale applications, the naturality of representing the inherent uncertainty of knowledge they store, and the ability to answer complex queries involving more than 2 variables. Note that the final one requires a true joint probabilistic model with a tractable inference procedure, not something provided by e.g. matrix factorization.

We take the dual approach to density-based embeddings and model uncertainty about relationships and attributes as explicitly probabilistic, while basing the probability on a latent space of geometric objects that obey natural structural biases for modeling transitive, asymmetric relations. The most similar work are the probabilistic order embeddings (POE) of Lai and Hockenmaier [52], which apply a probability measure to each order embedding's forward cone (the set of points greater than the embedding in each dimension), assigning a finite and normalized volume to the unbounded space. However, POE suffers severe limitations as a probabilistic model, including an inability to model negative correlations between concepts, which motivates the construction of our box lattice model.

Our model represents objects, concepts, and events as boxes, with an event's unary probability coming from the box volume and joint probabilities coming from overlaps. This contrasts with POE's approach of defining events as the forward cones of vectors, extending to infinity, integrated under a probability measure that assigns them finite volume.

One desirable property of a structured representation for ordered data, originally noted in Vendrov et al. [84] is a "slackness" shared by OE, POE, and our model: when the model predicts an "edge" or lack thereof (i.e. $P(a|b) = 0$ or 1, or a zero constraint violation in the case of OE), being exposed to that fact again will not update the model. Moreover, there are large degrees of freedom in parameter space that exhibit this slackness, giving the model the ability to embed complex structure with 0 loss when compared to models based on symmetric inner products or distances between embeddings, e.g. bilinear GLMs [23], Trans-E [13], and other embedding models which must always be pushing and pulling parameters towards and away from each other.

Our experiments demonstrate the power of our approach to probabilistic ordering-biased relational modeling. First, we investigate an instructive 2-dimensional toy dataset that both demonstrates the way the model self organizes its box event space, and enables sensible answers to queries involving arbitrary numbers of variables, despite being trained on only pairwise data. We achieve a new state of the art in denotational probability modeling on the Flickr entailment dataset [52], and a matching state-of-the-art on WordNet hypernymy [84, 60] with the concurrent work on thresholded Gaussian embedding of Athiwaratkun and Wilson [3], achieving our best results by training on additional co-occurrence expectations aggregated from leaf types.

We find that the strong empirical performance of probabilistic ordering models, and our box lattice model in particular, and their endowment of new forms of training and querying, make them a promising avenue for future research in representing structured knowledge.

5.2 Related Work

In addition to the related work in structured embeddings mentioned in the introduction, our focus on directed, transitive relational modeling and ontology induction shares much with the rich field of directed graphical models and causal modeling [68], as well as learning the structure of those models [41]. Work in undirected structure learning such the Graphical Lasso [31] is also relevant due to our desire to learn from pairwise joint/conditional probabilities and moment matrices, which are closely related in the setting of discrete variables.

Especially relevant research in Bayesian networks are applications towards learning taxonomic structure of relational data [4], although this work is often restricted towards tree-shaped ontologies, which allow efficient inference by Chu-Liu-Edmonds' algorithm [19], while we focus on arbitrary DAGs.

As our model is based on populating a latent "sample space" into boxes (products of intervals), it is especially reminiscent of the Mondrian process [73]. However, the Mondrian

process partitions the space as a high dimensional tree (a non-parametric kd-tree), while our model allows the arbitrary box placement required for DAG structure, and is much more tractable in high dimensions compared to the Mondrian's Bayesian non-parametric inference.

Embedding applications to relational learning constitute a huge field to which it is impossible to do justice, but one general difference between our approaches is that e.g. a matrix factorization model treats the embeddings as objects to score relation links with, as opposed to POE or our model in which embeddings represent subsets of probabilistic sample space which are directly integrated. They are full probabilistic models of the joint set of variables, rather than embedding-based approximations of only low-order joint and conditional probabilities. That is, any set of our parameters can answer any arbitrary probabilistic question (possibly requiring intractable computation), rather than being fixed to modeling only certain subsets of the joint.

Embedding-based learning's large advantage over the combinatorial structure learning presented by classical PGM approaches is its applicability to large-scale probability distributions containing hundreds of thousands of events or more, as in both our WordNet and Flickr experiments.

5.3 Method

We develop a probabilistic model for transitive relational data based on hyperrectangle (box) embeddings that can model both positive and negative correlations. Since this is precisely the domain for which probabilistic order embeddings (discussed in Chapter 4) were defined, we first motivate our choice to abandon these cone-based models.

5.3.1 Correlations from Cone Measures

Remark. The geometric intuition behind the following proof is well illustrated in Figure 5.1

Claim. *Let $(\Omega_{\text{Cone}}, \mathcal{E}, P)$ along with two indicator random variables A and B be a cone probability model as defined in 21. That is,*

$$\begin{aligned} a^\vee, b^\vee &\in \Omega_{\text{Cone}} \subseteq \mathbb{R}^d \\ \text{Cone}(A) &= \{\, u \mid u_i \geq a_i^\vee, 1 \leq i \leq d, u \in \Omega_{\text{Cone}} \,\} \\ \text{Cone}(B) &= \{\, u \mid u_i \geq b_i^\vee, 1 \leq i \leq d, u \in \Omega_{\text{Cone}} \,\} \\ A, B &= \mathbb{1}_{\text{Cone}(A)}, \mathbb{1}_{\text{Cone}(B)} \end{aligned}$$

for two parameter vectors $a^\vee$ and $b^\vee$. Further, assume that the base measure P is a product measure on a product space $\Omega_{\text{Cone}} \subseteq \mathbb{R}^d$ that decomposes coordinatewise into probability measures P_i. Finally, assume that the functions $F_i(x) = P_i((\infty, x) \cap \Omega_{\text{Cone}})$ are monotone increasing.

Then cov$(A, B) \geq 0$ for any settings of the model parameters. That is, cone probability models under a large class of base measures cannot model negative correlations.

Proof. For the product measure P we have

$$\begin{aligned} P(\text{Cone}(X)) &= \prod_i^d P_i([x_i^\vee, \infty) \cap \Omega_{\text{Cone}}) \\ &= \prod_i^d F_i(\infty) - F_i(x_i^\vee) \\ &= \prod_i^d 1 - F_i(x_i^\vee), \end{aligned}$$

since the axis-aligned cones decompose as a set product along with the measure.

The latter expression is just the area of the box $\prod_i^d [F_i(x_i), 1] \in [0, 1]^d$, under the standard Lebesgue (uniform) measure. Note that these boxes have only half the degrees of freedom of general boxes, since their per-dimension maximum is always 1 (intuitively, they have one end "stuck at infinity" since the cone extends to infinity.) Denote this box $\text{Box}(X)$.

Let $(\Omega_{\text{Box}}, \mathcal{F}, P_{\text{Box}})$ be a probability space with $\Omega_{\text{Box}} = [0,1]^d$ and P_{Box} equal to the standard Lebesgue measure. There is a measure-preserving bijection between the events $\text{Cone}(X)$ and $\text{Box}(X)$ because each F_i is monotone increasing and thus invertible.

So, W.L.O.G. we can consider the two boxes $\text{Box}(A)$ and $\text{Box}(B)$ corresponding to our cones in the original space, and letting $F_i(a_i^\vee) = u_i$ and $F_i(b_i^\vee) = v_i$, their intersection in the unit hypercube Ω_{Box} is $\prod_i^n [\max(u_i, v_i), 1]$.

Recall the formula for the covariance of two Bernoulli random variables:

$$\begin{aligned}
\text{cov}(A, B) &= \mathbb{E}[AB] - \mathbb{E}[A]\mathbb{E}[B] \\
&= P(\text{Cone}(A) \cap \text{Cone}(B)) - P(\text{Cone}(A))P(\text{Cone}(B)) \\
&= P(\text{Box}(A) \cap \text{Box}(B)) - P(\text{Box}(A))P(\text{Box}(B)) \\
&= \prod_i^d (1 - \max(u_i, v_i)) - \prod_i^d (1 - u_i)(1 - v_i) \geq 0
\end{aligned}$$

since the right contains all the terms of the left and can only grow smaller. □

An open question for future work is what non-product measures this claim also applies to. Note that some non-product measures, such as multivariate Gaussian, can be transformed into product measures easily (*whitening*) and the above proof would still apply. It seems probable that some measures, nonlinearly entangled across dimensions, could encode negative correlations in cone volumes. However, it is not generally tractable to integrate high-dimensional cones under arbitrary non-product measures.

5.3.2 Box Lattices

The above proof gives us intuition about the possible form of a better representation. Cones can be mapped into boxes within the unit hypercube while preserving their measure, and the lack of negative correlation between any two variables in a box probability model seems to come from the large size of the intersection of two cones, corresponding to the

positive $\mathbb{E}[AB]$ term, which dominates the covariance formula. This large intersection comes about due to "pinning" the maximum coordinate of each box to 1 in every dimension.

To remedy this, we propose to learn representations in the space of *all* boxes (axis-aligned hyperrectangles), gaining an extra degree of freedom over the cone probability model, and allowing us to create as much or as little overlap as we like between two boxes. Figure 5.1 gives a demonstration of how the amount of overlap created by different box and cone representations affects the covariance. These representations can be learned with a suitable probability measure in $\mathbb{R}^n$, the nonnegative orthant $\mathbb{R}^n_+$ (as with the original probabilistic order embedding), or directly in the unit hypercube with the standard Lebesgue measure, which we use in many of our applications.

First, we will provide a formal definition of a lattice structure over boxes.

Definition 22 (Box lattice)**.** A *box lattice* is constructed using a subset $S \subseteq \mathbb{R}^d$ along with a set of boxes (axis-aligned hyperrectangles), $T = \{\mathrm{Box}(X_i)\}_{i=1}^n$ in S, each parameterized by a pair of vectors $(x^{i,\wedge}, x^{i,\vee})$ representing the minimum and maximum coordinates of the box on each axis:

$$x^{i,\wedge}, x^{i,\vee} \in S \subseteq \mathbb{R}^d \quad \text{and} \quad \forall j \in \{1, \ldots, d\}, \quad x_j^{i,\wedge} < x_j^{i,\vee}$$

In terms of these parameters, the boxes are defined as

$$\mathrm{Box}(X_i) = \prod_{j=1}^{d} [x_j^{i,\wedge}, x_j^{i,\vee}] = \{\, u \mid x_j^{i,\wedge} \leq u_j \leq x_j^{i,\vee}, j \in \{1, \ldots, d\}, u \in S \,\}.$$

We can define an ordering on the set of boxes T using the subset relation,

$$\mathrm{Box}(A) \preceq \mathrm{Box}(B) \iff \mathrm{Box}(A) \subseteq \mathrm{Box}(B),$$

so $(T, \preceq)$ forms a poset. We can add a natural meet operation defined using set intersection,

$$\begin{aligned}
&\mathrm{Box}(A) \wedge \mathrm{Box}(B) = \mathrm{Box}(A) \cap \mathrm{Box}(B) \\
&\mathrm{Box}(A) \wedge \mathrm{Box}(B) = \bot \quad \text{if } \mathrm{Box}(A) \text{ and } \mathrm{Box}(B) \text{ disjoint, else} \\
&\mathrm{Box}(A) \wedge \mathrm{Box}(B) = \prod_{j=1}^{d} [\max(a_j^{\wedge}, b_j^{\wedge}), \min(a_j^{\vee}, b_j^{\vee})].
\end{aligned}$$

The resulting structure $(T, \wedge)$ forms a meet semilattice. Finally, we can add a join operation by finding the least upper bounding box,

$$\mathrm{Box}(A) \vee \mathrm{Box}(B) = \prod_{j=1}^{d} [\min(a_j^{\wedge}, b_j^{\wedge}), \max(a_j^{\vee}, b_j^{\vee})].$$

Note that this join operation is different from a standard union of sets because it always produces a (smallest enclosing) box, rather than an arbitrary union of hyperrectangles. The triplet $(T, \wedge, \vee)$ is the box lattice. This lattice is always bounded below, with $\bot = \emptyset$, and bounded above with $\top = S$ when S is itself a box.

This lattice, considered on its own terms as a non-probabilistic object, is strictly more general than the cone lattice in any dimension, which is proven in Section 7.1.

Remark. As mentioned above, we associate each variable X with two parameter vectors, the minimum and maximum value of the box at each dimension. Practically, for numerical reasons these are stored as a minimum and a positive offset, plus an ϵ term to prevent boxes from becoming too small and underflowing.

As our original motivation for developing boxes was to model different correlations than the cone probability model, we now apply the box lattice to define a probabilistic model. Much of this definition will follow the cone probability model (21), *mutatis mutandis*.

Definition 23 (Box probability model)**.** Let $(\Omega_{\text{Box}}, \mathcal{E}, P_{\text{Box}})$ be a probability space where $\Omega_{\text{Box}} \subseteq \mathbb{R}^d$. Let $\{\text{Box}(X_i)\}_{i=1}^N$ be a set of boxes in Ω_{Box}. As in the box lattice (Definition 22), each is parameterized by a pair of vectors

$$x^{i,\wedge}, x^{i,\vee} \in \Omega_{\text{Box}} \subseteq \mathbb{R}^d \quad \text{and} \quad \forall j \in \{1, \ldots, d\}, \quad x_j^{i,\wedge} < x_j^{i,\vee}.$$

In terms of these parameters, the boxes are defined as

$$\text{Box}(X_i) = \prod_{j=1}^{d} [x_j^{i,\wedge}, x_j^{i,\vee}].$$

Further, define a set of indicator random variables $\{X_i\}$ for these box-shaped events,

$$X_i : \Omega_{\text{Box}} \to \{0, 1\} = \mathbb{1}_{\text{Box}(X_i)}$$
$$X_i^{-1}(\{1\}) = \text{Box}(X_i).$$

We will denote the distribution of $X_1, ..., X_N$ as $P(X_1, ..., X_N)$, dropping the Box subscript on P, to make it clear we are talking about the discrete pushforward measure on $\{0, 1\}^N$ induced by P_{Box} and not the measure P_{Box} defined on a subset of $\mathbb{R}^d$. We will call this probability space and associated random variables a *box probability model*.

If each $X_i = f(a)$ for some mapping $f : S \to (\Omega_{\text{Box}} \to \{0, 1\})$ from a finite set S to random variables in the box probability model, we call this a *probabilistic box embedding* of S.

Remark. As in the case of the cone probability model, the above definition implies that joint probabilities over multiple variables can be calculated by inclusion-exclusion with set intersection and complement over $\text{Box}(X_i)$, e.g.

$$P(X_1 = 1, X_2 = 1, X_3 = 0) = P_{\text{Box}}(\text{Box}(X_1) \cap \text{Box}(X_2) \cap \text{Box}(X_3)^{\mathsf{c}}).$$

It remains to show that this representation can represent both positive and negative correlations.

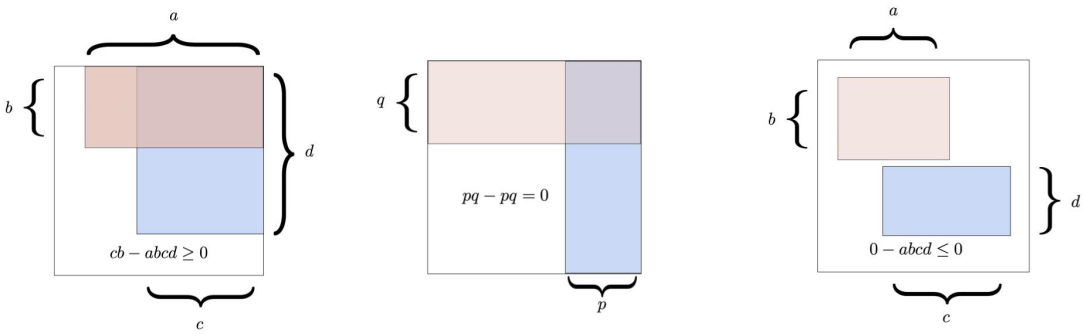
Claim. *Let* $(\Omega, \mathcal{E}, P)$ *be a probability space with* $\Omega = [0,1]^n$. *Let* $\mathrm{Box}(A)$ *and* $\mathrm{Box}(B)$ *be two boxes in* Ω*, and let* $A : \Omega \to \{0,1\} = \mathbb{1}_{Box(A)}$ *and* $B : \Omega \to \{0,1\} = \mathbb{1}_{Box(B)}$ *be indicator random variables of those events in* Ω*, that is, let* $A^{-1}(\{1\})$ *and* $B^{-1}(\{1\})$ *be products of intervals in* Ω*. There exist boxes* $\mathrm{Box}(A)$ *and* $\mathrm{Box}(B)$ *such that the correlation between the the two variables, corr*(A, B)*, can take on any value in* $[-1, 1]$.

Proof. Boxes can clearly model disjointness (exactly -1 correlation if the total volume of two disjoint boxes equals 1). Two identical boxes give their concepts exactly correlation 1. The area of the meet is continuous with respect to translations of intersecting boxes, and all other terms in correlation stay constant, so by continuity of the correlation function (and the intermediate value theorem) a box probability model can achieve all possible correlations for a pair of variables. □

This proof can be extended to boxes in $\Omega = \mathbb{R}^n$ compare $\Omega = \mathbb{R}^n$, where the base probability measure is a product measure, by the previous reduction.

A demonstration in two dimensions of the different possible correlation structures from box and cone embeddings can be found in Figure 5.1.

Limitations: Note that this model cannot perfectly describe all possible probability distributions or concepts as embedded objects. For example, the complement of a box is not a box. However, queries about complemented variables can be calculated by the Inclusion-Exclusion principle, made more efficient by the fact that all non-negated terms can be grouped and calculated exactly. We show some toy exact calculations with negated variables in Appendix 5.5. Also, note that in a knowledge graph often true complements are not required — for example *mortal* and *immortal* are not actually complements, because the concept *color* is neither.



(a) Positive covariance (b) Zero covariance (c) Negative covariance

Figure 5.1: A visualization of the Bernoulli covariance formula $\mathbb{E}[AB] - \mathbb{E}[A]\mathbb{E}[B]$ for box and cone embeddings in the unit square in two dimensions, showing positive, zero, and negative correlation.

Additionally, requiring the total probability mass covered by boxes to equal 1, or exactly matching marginal box probabilities while modeling all correlations is a difficult box-packing-type problem and not generally possible. Modeling limitations aside, the union of boxes having mass < 1 can be seen as an open-world assumption on our KB (not all points in space have corresponding concepts, yet).

5.3.3 Learning

While inference (calculation of pairwise joint, unary marginal, and pairwise conditional probabilities) in the box model is quite straightforward by taking intersections of boxes and computing volumes (and their ratios), learning does not appear easy at first glance. While the (sub)gradient of the joint probability with respect to the box embedding parameters is nonzero when boxes intersect, it has zero derivative otherwise, making it unsuitable for optimization. Instead we optimize a lower bound, which we arrive at using the join operation in the box lattice, which finds the smallest enclosing box when applied to two boxes.

For two variables A and B and their associated boxes, clearly $P_{\text{Box}}(\text{Box}(A) \vee \text{Box}(B)) \geq P_{\text{Box}}(\text{Box}(A) \cup \text{Box}(B))$, with equality only when $\text{Box}(A) = \text{Box}(B)$, so this can give us

a lower bound:

$$P_{\text{Box}}(\text{Box}(A) \wedge \text{Box}(b)) = P_{\text{Box}}(\text{Box}(A)) + P_{\text{Box}}(\text{Box}(B)) - P_{\text{Box}}(\text{Box}(A) \cup \text{Box}(B))$$
$$\geq P_{\text{Box}}(\text{Box}(A)) + P_{\text{Box}}(\text{Box}(B)) - P_{\text{Box}}(\text{Box}(A) \vee \text{Box}(B)),$$

where P_{Box} is the measure on the latent box space that gives probabilities according to the volume of the associated box. This lower bound always exists and is nonzero, even when the true joint probability is not. It is guaranteed to be nonpositive except when $\text{Box}(A)$ and $\text{Box}(B)$ intersect, in which case the true joint likelihood should be used.

While a negative bound on a probability is odd, inspecting the bound we see that its gradient will push the enclosing box to be smaller, while increasing areas of the individual boxes, until they intersect, which is a sensible learning strategy.

Since we are working with small probabilities it is advisable to negate this term and maximize the negative logarithm:

$$-\log(\delta + P_{\text{Box}}(\text{Box}(A) \vee \text{Box}(B)) - P_{\text{Box}}(\text{Box}(A)) - P_{\text{Box}}(\text{Box}(B))$$

This still has an unbounded gradient as the lower bound approaches 0, so it is also useful to add a constant $\delta > 0$ within the logarithm function to avoid numerical problems.

Since the likelihood of the full data is usually intractable to compute as a conjunction of many negations, we optimize binary conditional and unary marginal terms separately by maximum likelihood. This is called a composite likelihood method [83], and sometimes called *pseudolikelihood*, though the latter usually refers to a composite of full leave-one-out conditional distributions.

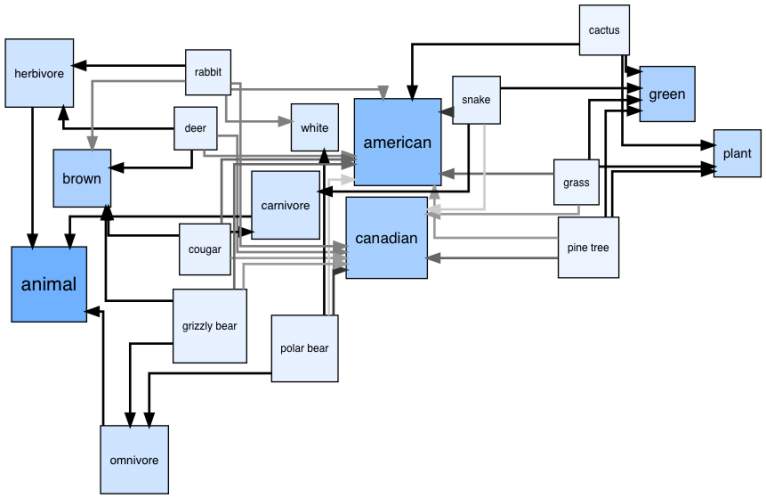
In this work, we parametrize the boxes as $(\text{min}, \Delta = \text{max} - \text{min})$, with Euclidean projections after gradient steps to keep our parameters in the unit hypercube and maintain the minimum/delta constraints.

Now that we have the ability to compute probabilities and (surrogate) gradients for arbitrary marginals in the model, and by extension conditionals, we will see specific examples in the experiments.

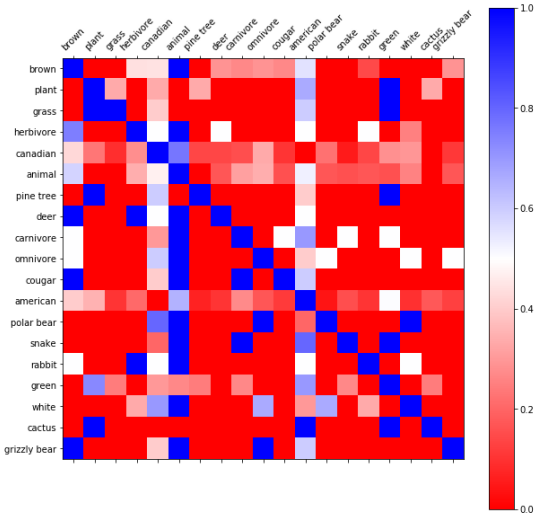
5.4 Experiments

5.4.1 Warmup: 2D Embedding of a Toy Lattice

In these experiments we use $P(V_1|V_2)$ not to denote the conditional distribution, but instead to denote the probability $P(V_1 = 1|V_2 = 1)$ by abuse of notation. We begin by investigating properties of our model in modeling a small toy problem, consisting of a small hand constructed ontology over 19 concepts, aggregated from atomic synthetic examples and some manually specified conditional probabilities, a visualization of which is provided in Figure 5.2. These manually specified examples are aggregated into a full pairwise conditional probability distribution (CPD) with missing probabilities computed using inclusion-exclusion. We model it using only 2 dimensions to enable visualization of the way the model self-organizes its latent event space, Ω_{Box}, training the model by minimize weighted cross-entropy with both the unary marginals and pairwise conditional probabilities. We also conduct a parallel experiment with POE as embedded in the unit cube, where each representation is constrained to touch the faces $x = 1, y = 1$. In Figure 5.3, we show the representation of lattice structures by POE and the box lattice model as compared to the abstract probabilistic lattice used to construct the data, shown in Figure 5.2, and compare the conditional probabilities produced by our model to the ground truth, demonstrating the richer capacity of the box model in capturing strong positive and negative correlations. In Table 5.3, we perform a series of multivariable conditional queries and demonstrate intuitive results on high-order queries containing up to 4 variables, despite the model being trained on only 2-way information.

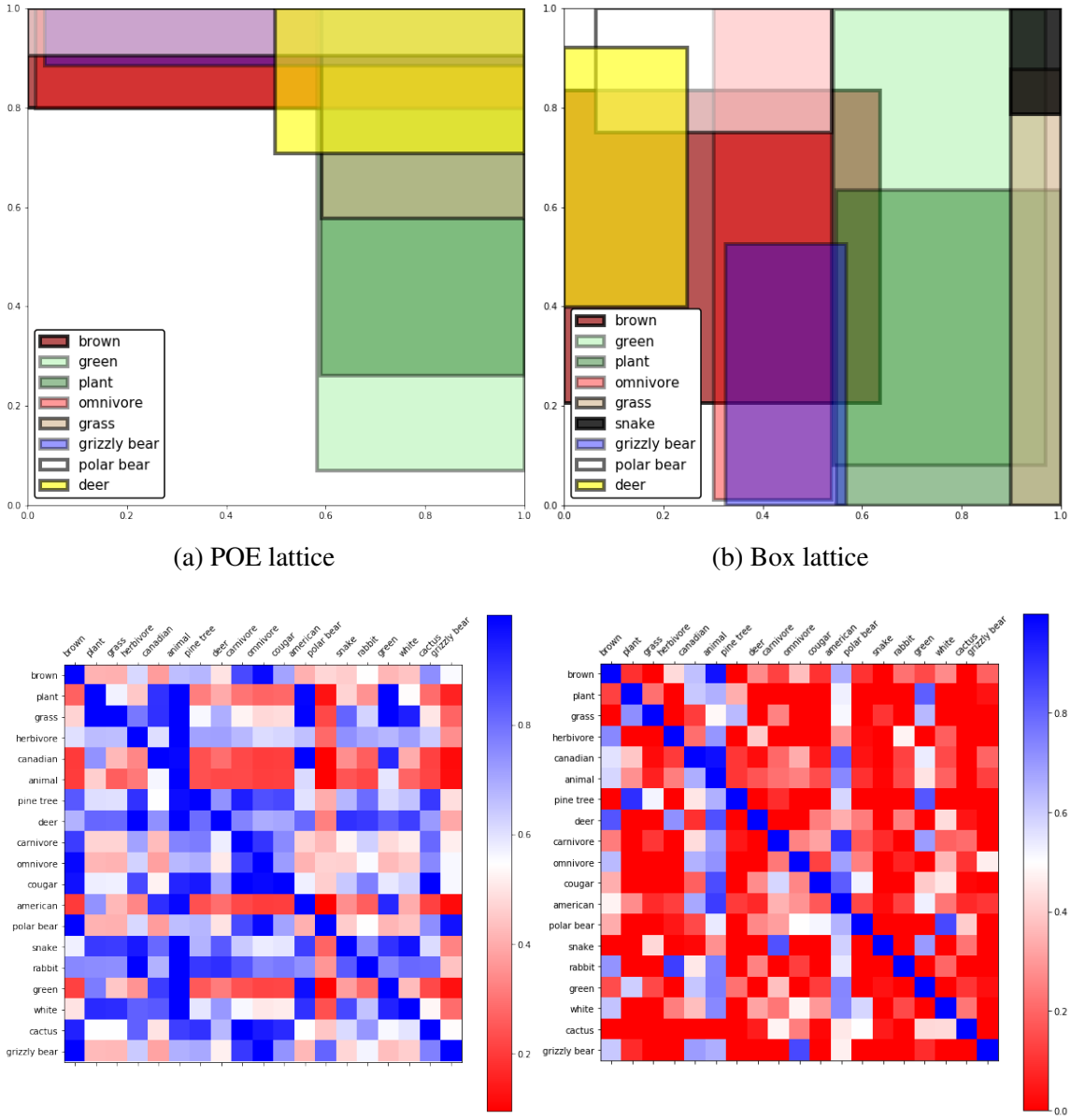


(a) Original lattice



(b) Ground truth CPD

Figure 5.2: Representation of the toy probabilistic lattice used in Section 5.4.1. Darker color corresponds to more unary marginal probability. Edges represent specified conditional probabilities, with the rest left up to inference. The associated CPD is obtained by a weighted aggregation of leaf elements.



(a) POE lattice

(b) Box lattice

(c) POE CPD

(d) Box CPD

Figure 5.3: Lattice representations and conditional probabilities from POE vs. box lattice. Note how the box lattice model's lack of "anchoring" to a corner allows it vastly more expressivity in matching the ground truth CPD seen in Figure 5.2.

P(grizzly bear | ...)		P(cactus | ...)		P(plant | ...)	
P(grizzly bear)	0.12	P(cactus)	0.10	P(plant)	0.20
omnivore	0.29	green	0.16	green	0.37
white	0.00	plant	0.39	snake	0.00
brown	0.30	american, green	0.19	carnivore	0.00
omnivore, white	0.00	plant, green, american	0.40	cactus	0.78
omnivore, brown	0.38	american, carnivore	0.00	american, cactus	0.85

Table 5.3: Multi-way queries: conditional probabilities adjust when adding additional evidence or contradiction. In contrast, POE can only raise or preserve probability when conditioning.

term1	term2
craftsman.n.02	shark.n.03
homogenized_milk.n.01	apple_juice.n.01
tongue_depresser.n.01	paintbrush.n.01
deerstalker.n.01	bathing_cap.n.01
skywriting.n.01	transcript.n.01

Table 5.4: Negatively correlated variables produced by the model.

Method	Test Accuracy %
transitive	88.2
word2gauss	86.6
OE	90.6
Li et al. [55]	91.3
DOE (KL)	**92.3**
POE	91.6
POE (100 dim)	91.7
Box	92.2
Box + CPD	**92.3**

Table 5.5: Classification accuracy on WordNet test set.

5.4.2 WordNet

We experiment on WordNet hypernym prediction, using the same train, development and test split as Vendrov et al. [84], created by randomly taking 4,000 hypernym pairs from the 837,888-edge transitive closure of the WordNet hypernym hierarchy as positive training examples for the development set, 4,000 for the test set, and using the rest as training data.

Negative training examples are created by randomly corrupting a train/development/test edge (u, v) by replacing either u or v with a randomly chosen negative node. We use their specific train/dev/test split, while Athiwaratkun and Wilson [3] use a different train/dev split with the same test set (personal communication) to examine the effect of different negative sampling techniques. We cite their best performing model, called DOE (KL).

Since our model is probabilistic, we would like a sensible value for the unary marginal probability $P(N_i = 1)$, where N_i is the random variable associated with node i. We assign these marginal probabilities by looking at the number of descendants in the training graph under a node, and normalizing over all nodes, taking $P(N_i = 1) = \frac{|\ descendants(N_i)\ |}{|\ nodes\ |}$.

Furthermore, we use the graph structure (only of the subset of edges in the training set to avoid leaking data) to augment the data with approximate conditional probabilities $P(N_i = 1|N_j = 1)$. For each leaf node, we take the set of its ancestor nodes, and increment a pairwise co-occurrence count for all pairs of variables in the set. We then aggregate these pairwise counts, and divide by the number of leaves to get an approximate binary joint probability distribution between node variables, $P(N_i = 1, N_j = 1) = \frac{|\ N_i, N_j\ co\text{-}occur\ in\ ancestor\ set\ |}{|\ leaves\ |}$. With this and the unary marginals, we can create a conditional probability table, which we prune based on the difference of $P(N_i = 1|N_j = 1)$ and $P(N_i = 1|N_j = 1)$, and add cross entropy with these conditional "soft edges" to the training objective. We refer to experiments using this additional data as Box + CPD in Table 5.5.

We use 50 dimensions in our experiments. Since our model has 2 parameters per dimension, we also perform an apples-to-apples comparison with a 100D POE model. As seen in Table 5.5, we outperform POE significantly even with this added representational power. We also observe sensible negatively correlated examples, shown in 5.4, in the trained box model, while POE cannot represent such relationships. We tune our models on the development set, with parameters documented in Appendix A.2.1. We observe that not only does our model outperform POE, it beats all previous results on WordNet, aside from the

concurrent work of Athiwaratkun and Wilson [3] (using different train/dev negative examples), the baseline POE model does as well. This indicates that probabilistic embeddings for transitive relations are a promising avenue for future work. Additionally, the ability of the model to learn from the expected "soft edges" improves it to state-of-the-art level. We expect that co-occurrence counts gathered from real textual corpora, rather than merely aggregating up the WordNet lattice, would further strengthen this effect.

5.4.3 Flickr Entailment Graph

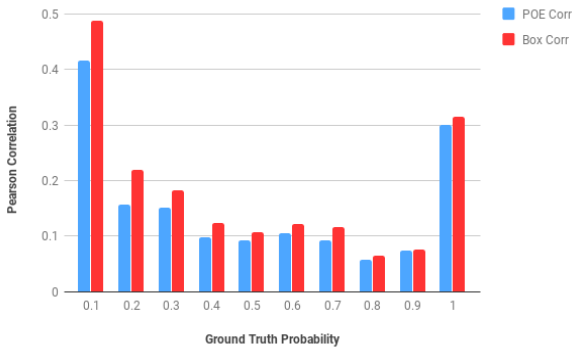


Figure 5.4: R between model and gold probabilities.

	$P(X = 1\|Y = 1)$	
Full test data	KL	Pearson R
POE	0.031	0.949
POE*	0.031	0.949
Box	**0.020**	**0.967**
Unseen pairs		
POE	0.048	0.920
POE*	0.046	0.925
Box	**0.025**	**0.957**
Unseen words		
POE	0.127	0.696
POE*	0.084	0.854
Box	**0.050**	**0.900**

Table 5.6: KL and Pearson correlation between model and gold probability.

We conduct experiments on the large-scale Flickr entailment dataset of 45 million image caption pairs. We use the exactly same train/dev/test from Lai and Hockenmaier [52]. We use a slightly different unseen word pairs and unseen words test data, obtained from the author. We include their published results and also use their published code, marked $*$, for comparison.

For these experiments, we relax our boxes from the unit hypercube to the nonnegative orthant and obtain probabilities under the exponential measure, $p(x) = \exp(-x)$. We enforce the nonnegativity constraints by clipping the LSTM-generated embedding [44] for the box minimum with a ReLU, and parametrize our Δ embeddings using a softplus activation to prevent dead units. As in Lai and Hockenmaier [52], we use 512 hidden units in our LSTM to compose sentence vectors. We then apply two single-layer feed-forward networks with 512 units applied to the final LSTM state to produce the embeddings. We tune our models on the development set, with additional details given in Appendix A.2.2.

As we can see from Table 5.6, we note large improvements in KL and Pearson correlation to the ground truth entailment probabilities. In further analysis, Figure 5.4 demonstrates that while the box model outperforms POE in nearly every regime, the highest gains come from the comparatively difficult to calibrate small entailment probabilities, indicating the greater capability of our model to produce fine-grained distinctions.

5.5 Queries with Negated Variables

Section 5.3.2 mentions that although the complement of a box is not a box, queries involving negated variables can be calculated exactly with Inclusion-Exclusion, demonstrated in Table 5.7. Note that in this table, notation of the form $P(V_1|V_2)$ does not refer to a distribution, but instead is a slight abuse of notation for the probability $P(V_1 = 1|V_2 = 1)$. While there are many more interesting and efficient approaches, we simply use the formula for calculating the volume of the union of hyperrectangles (a standard Inclusion-Exclusion formula).

This is equivalent since the intersection of complements of boxes is the complement of the union of boxes.

We are asked to compute the probability of an assignment of variables such that some take the value 1 and some take the value 0, we will call these two sets of variables T_i and F_i

$$P(T_1, ..., T_n = 1, F_1, ..., F_n = 0).$$

In the box model, we know that this probability is given by

$$P_{\text{Box}}(\text{Box}(T_1) \cap ... \cap \text{Box}(T_n) \cap \text{Box}(F_1)^c \cap ... \cap \text{Box}(F_n)^c),$$

the volume of boxes intersected with complements of boxes, computed under the measure P_{Box}.

We first intersect all of the non-negated variables into one conjunction box, $\text{Box}(T)$. We then calculate the volume of the union of T with the union of all of the boxes representing negated variables,

$$v_1 = P_{\text{Box}}(T \cup \text{Box}(F_1) \cup ... \cup \text{Box}(F_n)).$$

We can calculate the volume of this union of hyperrectangles using inclusion-exclusion.

Finally, we calculate the volume of just the negated variables boxes,

$$v_2 = P_{\text{Box}}(\text{Box}(F_1) \cup ... \cup \text{Box}(F_n)),$$

another union of hyperrectangles.

The original desired probability is then given by $v_1 - v_2$, since this measures the size of the area inside all of the true variables T_i that is also outside of the false variables F_i.

P(deer | ...)	
P(deer)	0.12
¬white	0.13
animal	0.50
¬white,animal	0.54
¬white,animal,herbivore	0.73
¬white, animal, herbivore, ¬rabbit	0.80
¬white, animal, ¬herbivore,¬rabbit	0.00

Table 5.7: Negated variables: queries on the toy data with negated variables, calculated with Inclusion-Exclusion.

5.6 Conclusion

In this chapter, we presented an embedding for probabilistic modeling based on a lattice of hyperrectangles. While showing much promise in modeling both probabilistic and non-probabilistic partial order structures, and performing probabilistic queries, the training / inference of latent boxes relies on an ad-hoc surrogate function in the disjoint case. The search for better training methods motivates the next chapter.

CHAPTER 6

SMOOTHED LEARNING FOR BOX EMBEDDINGS

6.1 Introduction

While intuitively appealing, the "hard edges" of boxes and their ability to become easily disjoint, present difficulties for gradient-based optimization: when two boxes are disjoint in the model, but have overlap in the ground truth, no gradient can flow to the model to correct the problem. This is of special concern for (pseudo-)sparse data, where many boxes should have nearly zero overlap, while others should have very high overlap. This is especially pronounced in the case of e.g. market basket models for recommendation, where most items should not be recommended, and entailment tasks, most of which are currently artificially resampled into a 1:1 ratio of positive to negative examples. To address the disjoint case, in the previous chapter we introduced an ad-hoc surrogate function. In this chapter, we will look at this problem as inspiration for a new model, based on the intuition of relaxing the hard edges of the boxes into smoothed density functions, using a Gaussian convolution with the original boxes.

We demonstrate the superiority of our approach to modeling transitive relations on WordNet, Flickr caption entailment, and a MovieLens-based market basket dataset. This modification of the original box model matches or beats existing state of the art results, while showing substantial improvements in the pseudosparse regime.

6.2 Related Work

Our approach to smoothing the energy landscape of the model using Gaussian convolution is common in mollified optimization and continuation methods, and is increasingly

making its way into machine learning models such as Mollifying Networks [38], diffusion-trained networks [64], and noisy activation functions [37].

6.3 Method

6.3.1 Motivation: Optimization and Sparse Data

Recall that in the box probability model $(\Omega_{\text{Box}}, \mathcal{E}, P_{\text{Box}}, \{X_i\})$, as defined in Chapter 5, the probability of a binary variable taking on the value 1 is given by

$$P(X_i = 1) = P_{\text{Box}}(\text{Box}(X_i)),$$

the volume of the box $\text{Box}(X_i)$, parameterized as a vector of minimum coordinates $x_i^{\wedge}$, and maximum coordinates $x_i^{\vee}$. Without loss of generality, we will consider boxes in only 1 dimension. In this case, the volume under the uniform measure is given simply as

$$P(X_i = 1) = P_{\text{Box}}(\text{Box}(X_i)) = x_i^{\vee} - x_i^{\wedge}.$$

Computing the joint probability of two variables taking the value 1 is done by first taking the intersection (or *meet* of the two boxes when viewed as elements of the box lattice), then computing the volume:

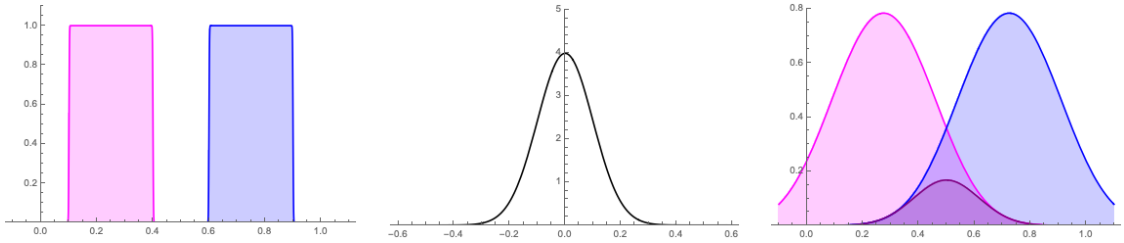
$$P(X = 1, Y = 1) = P_{\text{Box}}(\text{Box}(X) \wedge \text{Box}(Y)) = m_h(\min(x^{\vee}, y^{\vee}) - \max(x_i^{\wedge}, y^{\wedge})). \tag{6.1}$$

where $m_h(x)$ is the standard hinge function $m_h(x) = 0 \vee x = \max(0, x)$.

When using gradient-based optimization to learn box embeddings, an immediate problem identified in the original box embedding work is that when two concepts are incorrectly given as disjoint by the model, no gradient signal can flow since the meet (intersection) is exactly zero, with zero derivative, as inspection of the previous formula makes clear.

The hinge function has a large flat plateau at 0 when intervals are disjoint. This issue is especially problematic when the lattice to be embedded is (pseudo-)sparse, that is, most boxes should have very little or no intersection, since if training accidentally makes two boxes disjoint there is no way to recover with the naive measure. We previously proposed a surrogate function to optimize in this case, but we will use a more principled framework to develop alternate measures that avoid this pathology, improving both optimization and final model quality.

6.3.2 Relaxed Geometry



(a) Unsmoothed Indicators (b) Convolution Kernel (c) Smoothed w/ Overlap

Figure 6.1: One-dimensional example demonstrating two disjoint indicators of intervals before and after the application of a smoothing kernel. The area under the purple product curve is proportional to the degree of overlap.

The intuition behind our approach is that the "hard edges" of the standard box embeddings lead to unwanted gradient sparsity, and we seek a relaxation of this assumption that maintains the desirable properties of the base lattice model while enabling better optimization and preserving a geometric intuition. For ease of exposition, we will refer to 1-dimensional intervals with $\Omega_{\text{Box}} = [0, 1]$ in this section, but the results carry through from the representation of boxes as products of intervals and their volumes under the associated product measures.

The first observation is that, considering boxes as indicator functions of intervals, we can rewrite the measure of the joint probability $P(X = 1, Y = 1) = P_{\text{Box}}(\text{Box}(X) \wedge$

$\mathrm{Box}(Y)$) between intervals $\mathrm{Box}(X) = [x^\wedge, x^\vee]$ and $\mathrm{Box}(Y) = [y^\wedge, y^\vee]$ as an integral of the product of those indicators:

$$P_{\mathrm{Box}}(\mathrm{Box}(X) \wedge \mathrm{Box}(Y)) = \int\limits_{\omega \in \Omega_{\mathrm{Box}}} \mathbb{1}_{[x^\wedge, x^\vee]}(\omega) \mathbb{1}_{[y^\wedge, y^\vee]}(\omega) d\omega$$

since the product has support (and is equal to 1) only in the areas where the two intervals overlap.

A solution suggests itself in replacing these indicator functions with functions of infinite support. We elect for kernel smoothing, specifically convolution with a normalized Gaussian kernel, equivalent to an application of the diffusion equation to the original functional form of the embeddings (indicator functions) and a common approach to mollified optimization and energy smoothing [65, 38, 64]. This approach is demonstrated in one dimension in Figure 6.1.

Specifically, given $\mathrm{Box}(X) = [x^\wedge, x^\vee]$, we associate the smoothed indicator function

$$f(\omega; x^\wedge, x^\vee, \sigma^2) = \mathbb{1}_{[x^\wedge, x^\vee]}(\omega) * \phi(\omega; \sigma^2) =$$
$$\int\limits_{\omega' \in \Omega_{\mathrm{Box}}} \mathbb{1}_{[x^\wedge, x^\vee]}(\omega') \phi(\omega - \omega'; \sigma^2) d\omega' = \int_{x^\wedge}^{x^\vee} \phi(\omega - \omega'; \sigma^2) d\omega'$$

We then wish to evaluate, for two variables X and Y with associated smoothed indicators f and g,

$$\tilde{P}_\phi(\mathrm{Box}(X) \wedge \mathrm{Box}(Y)) = \int\limits_{\omega \in \Omega_{\mathrm{Box}}} f(\omega; x^\wedge, x^\vee, \sigma_1^2) g(\omega; y^\wedge, y^\vee, \sigma_2^2) d\omega \tag{6.2}$$

We denote this function $\tilde{P}_\phi(\mathrm{Box}(X) \wedge \mathrm{Box}(Y))$ with a tilde because it is not a true probability measure. We will also overload the $\tilde{P}_\phi$ function to apply to random variables as well as their associated boxes, so by e.g. $\tilde{P}_\phi(X = 1)$, we mean $\tilde{P}_\phi(\mathrm{Box}(X))$. This definition

can be extended to joint assignments of random variables using the inclusion-exclusion formula and computing positive assignments as box overlaps.

The integral 6.2 admits a closed form solution.

Proposition 1. *Let $m_\Phi(x) = \int \Phi(x)dx$ be an antiderivative of the standard normal CDF. Let $(a, b, c, d) = (x^\wedge, x^\vee, y^\wedge, y^\vee)$ for ease of notation. Then the solution to* (6.2) *is given by,*

$$\tilde{P}_\phi(\mathrm{Box}(X) \wedge \mathrm{Box}(Y)) = \sigma \left(m_\Phi(\tfrac{b-c}{\sigma}) + m_\Phi(\tfrac{a-d}{\sigma}) - m_\Phi(\tfrac{b-d}{\sigma}) - m_\Phi(\tfrac{a-c}{\sigma}) \right) \tag{6.3}$$

$$\approx \left(\rho\,\mathrm{soft}(\tfrac{b-c}{\rho}) + \rho\,\mathrm{soft}(\tfrac{a-d}{\rho}) \right) - \left(\rho\,\mathrm{soft}(\tfrac{b-d}{\rho}) + \rho\,\mathrm{soft}(\tfrac{a-c}{\rho}) \right) \tag{6.4}$$

where $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$, $\mathrm{soft}(x) = \log(1 + \exp(x))$ is the softplus function, the antiderivative of the logistic sigmoid, and $\rho = \frac{\sigma}{1.702}$.

Proof. The first line is proved in Section 6.5, the second approximation follows from the approximation of Φ by a logistic sigmoid given in Bowling et al. [14]. □

Note that, in the *zero-temperature limit*, as ρ goes to zero, we recover the formula

$$\begin{aligned}
\tilde{P}_\phi(\mathrm{Box}(X) \wedge \mathrm{Box}(Y)) &= \lim_{\rho \to 0} \left(\rho\,\mathrm{soft}(\tfrac{b-c}{\rho}) + \rho\,\mathrm{soft}(\tfrac{a-d}{\rho}) \right) - \left(\rho\,\mathrm{soft}(\tfrac{b-d}{\rho}) + \rho\,\mathrm{soft}(\tfrac{a-c}{\rho}) \right) \\
&= \left(m_h(b-c) + m_h(a-d) \right) - \left(m_h(b-d) + m_h(a-c) \right) \\
&= m_h(b \wedge d - a \vee c)
\end{aligned}$$

with equality in the last line because (a, b) and (c, d) are intervals. This last line is exactly our original equation (6.1), which is expected from convolution with a zero-bandwidth kernel (a Dirac delta function, the identity element under convolution). This is true for both the exact formula using $\int \Phi(x)dx$, and the softplus approximation.

Unfortunately, for any $\rho > 0$, multiplication of Gaussian-smoothed indicators does not give a valid meet operation on a function lattice, for the simple reason that $f^2 \neq f$, except in the case of indicator functions, violating the idempotence requirement of Section 4.1.1.

More importantly, for practical considerations, if we are to treat the outputs of $\tilde{P}_\phi$ as probabilities, the consequence is

$$\tilde{P}_\phi(X = 1|X = 1) = \frac{\tilde{P}_\phi(X = 1, X = 1)}{\tilde{P}_\phi(X = 1)} = \frac{\tilde{P}_\phi(\mathrm{Box}(X) \wedge \mathrm{Box}(X))}{\tilde{P}_\phi(\mathrm{Box}(X))} \neq 1 \tag{6.5}$$

which complicates our applications that train on conditional probabilities. However, by a modification of (6.3), we can obtain a function $\tilde{P}$ such that $\tilde{P}(\mathrm{Box}(X) \wedge \mathrm{Box}(X)) = \tilde{P}(\mathrm{Box}(X))$, while retaining the smooth optimization properties of the Gaussian model.

Recall that for the hinge function m_h and two intervals (a, b) and (c, d), we have

$$\big(m_h(b - c) + m_h(a - d)\big) - \big(m_h(b - d) + m_h(a - c)\big) = m_h(b \wedge d - a \vee c) \tag{6.6}$$

where the left hand side is the zero-temperature limit of the Gaussian model from (6.3). This identity is true of the hinge function m_h, but not the softplus function.

However, an equation with a similar functional form as (6.6) (on both the left- and right-hand sides) is true not only of the hinge function from the unsmoothed model, but also true of the softplus. For two intervals $x = (a, b)$ an $y = (c, d)$, by the commutativity of $\min$ and $\max$ with monotonic functions, we have

$$\big(\,\mathrm{soft}(b - c) \vee \mathrm{soft}(a - d)\big) \wedge \big(\,\mathrm{soft}(b - d) \vee \mathrm{soft}(a - c)\big) = \mathrm{soft}(b \wedge d - a \vee c) \tag{6.7}$$

In the zero-temperature limit, all terms in equations 6.3 and 6.7 are equivalent. However, outside of this, (6.7) is idempotent for $x = y = (a, b) = (c, d)$ (when considered as a measure of overlap, made precise in the next paragraph), while (6.3) is not.

This inspires us to define the pseudo-probabilities $\tilde{P}(X = 1)$ and $\tilde{P}(X = 1, Y = 1)$ using a normalized version of (6.7) in place of (6.3). For the interval (one-dimensional box) case, we define

$$\tilde{P}(X = 1) \propto \text{soft}(b - a)$$
$$\tilde{P}(X = 1, Y = 1) \propto \text{soft}(b \wedge d - a \vee c)$$

which satisfies the idempotence requirement, $\tilde{P}(X = 1) = \tilde{P}(X = 1, X = 1)$.

Because softplus upper-bounds the hinge function, it is capable of outputting values that are greater than 1, and therefore must be normalized. In our experiments, we use two different approaches to normalization. For experiments with a relatively small number of entities (all besides Flickr), we allow the boxes to learn unconstrained, and divide each dimension by the measured size of the global minimum and maximum $(G_i^\wedge, G_i^\vee)$ at that dimension

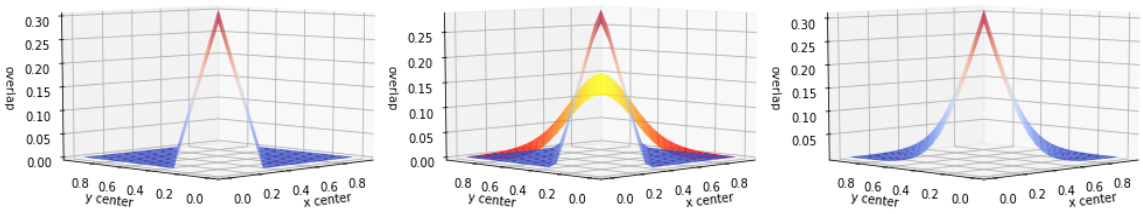
$$m_{\text{soft},i}(x) = \frac{\text{soft}(\frac{x}{\rho})}{\text{soft}(\frac{G_i^\vee - G_i^\wedge}{\rho})}$$

For data where computing these values repeatedly is infeasible, we project onto the unit hypercube and normalize by $m_{\text{soft}}(1)$. The final probability $\tilde{P}(X = 1)$ is given by the product over dimensions

$$\tilde{P}(X = 1) = \prod_i m_{\text{soft},i}(x_i^\vee - x_i^\wedge)$$
$$\tilde{P}(X = 1, Y = 1) = \prod_i m_{\text{soft},i}(x_i^\vee \wedge y_i^\vee - x_i^\wedge \vee y_i^\wedge)$$

Note that, while equivalent in the zero temperature limit to the standard uniform probability measure of the box model, this function, like the Gaussian model, is not a valid probability measure on the entire joint space of events (the lattice). However, neither is factorization of

a conditional probability table using a logistic sigmoid link function, which is commonly used for the similar tasks. Our approach retains the inductive bias of the original box model, is equivalent in the limit, and satisfies the necessary condition that $\tilde{P}(X = 1, X = 1) = \tilde{P}(X = 1)$. A comparison of the 3 different functions is given in Figure 6.2, with the softplus overlap showing much better behavior for highly disjoint boxes than the Gaussian model, while also preserving the meet property.



(a) Standard (hinge) overlap (b) Gaussian, $\sigma \in \{2, 6\}$ (c) Softplus overlap

Figure 6.2: Comparison of different overlap functions for two boxes of width 0.3 as a function of their centers. Note that in order to achieve high overlap, the Gaussian model must drastically lower its temperature, causing vanishing gradients in the tails.

6.4 Experiments

6.4.1 WordNet

Method	Test Accuracy %
transitive	88.2
word2gauss	86.6
OE	90.6
Li et al. [55]	91.3
POE	91.6
Box	92.2
Smoothed Box	92.0

Table 6.3: Classification accuracy on WordNet test set.

We perform experiments on the WordNet hypernym prediction task in order to evaluate the performance of these improvements in practice. The WordNet hypernym hierarchy contains 837,888-edges after performing the transitive closure on the direct edges in WordNet. We used the same train/dev/test split as in Vendrov et al. [84]. Positive examples are randomly chosen from the 837k edges, while negative examples are generated by swapping one of the terms to a random word in the dictionary. Experimental details are given in Appendix A.3.1.

The smoothed box model performs nearly as well as the original box lattice in terms of test accuracy[1]. While our model requires less hyper-parameter tuning than the original, we suspect that our performance would be increased on a task with a higher degree of sparsity than the 50/50 positive/negative split of the standard WordNet data, which we explore in the next section.

6.4.2 Imbalanced WordNet

In order to confirm our intuition that the smoothed box model performs better in the sparse regime, we perform further experiments using different numbers of positive and negative examples from the WordNet *mammal* subset, comparing the box lattice, our smoothed approach, and order embeddings (OE) as a baseline. The training data is the transitive reduction of this subset of the *mammal* WordNet, while the dev/test is the transitive closure of the training data. The training data contains 1,176 positive examples, and the dev and test sets contain 209 positive examples. Negative examples are generated randomly using the ratio stated in the table.

As we can see from the table, with balanced data, all models include OE baseline, Box, Smoothed Box models nearly match the full transitive closure. As the number of negative examples increases, the performance drops for the original box model, but Smoothed Box still outperforms OE and Box in all setting. This superior performance on imbalanced data

[1] Accuracy is calculated by applying the same threshold which maximized accuracy in dev set.

is important for e.g. real-world entailment graph learning, where the number of negatives greatly outweigh the positives.

Positive:Negative	Box	OE	Smoothed Box
1:1	0.9905	0.9976	**1.0**
1:2	0.8982	0.9139	**1.0**
1:6	0.6680	0.6640	**0.9561**
1:10	0.5495	0.5897	**0.8800**

Table 6.4: F1 scores of the box lattice, order embeddings, and our smoothed model, for different levels of label imbalance on the WordNet *mammal* subset.

6.4.3 Flickr

We conduct experiments on the Flickr entailment dataset. Flickr is a large-scale caption entailment dataset containing of 45 million image caption pairs. In order to perform an apples-to-apples comparison with existing results we use the exact same dataset from Vilnis et al. [86]. In this case, we do constrain the boxes to the unit cube, using the same experimental setup as Vilnis et al. [86], except we apply the softplus function before calculating the volume of the boxes.

We report KL divergence and Pearson correlation on the full test data, unseen pairs (caption pairs which are never occur in training data) and unseen captions (captions which are never occur in training data). As shown in Table 6.5, we see a slight performance gain compared to the original model, with improvements most concentrated on unseen captions.

6.4.4 MovieLens

We apply our method to a market-basket task constructed using the MovieLens dataset. Here, the task is to predict users' preference for movie A given that they liked movie B. We first collect all pairs of user-movie ratings higher than 4 points (strong preference) from the MovieLens-20M dataset. From this we further prune to just a subset of movies which have more than 100 user ratings to make sure that counting statistics are significant

	$P(x\|y)$	
Full test data	KL	Pearson R
POE	0.031	0.949
POE*	0.031	0.949
Box	0.020	0.967
Smoothed Box	**0.018**	**0.969**
Unseen pairs		
POE	0.048	0.920
POE*	0.046	0.925
Box	0.025	**0.957**
Smoothed Box	**0.024**	**0.957**
Unseen captions		
POE	0.127	0.696
POE*	0.084	0.854
Box	0.050	0.900
Smoothed Box	**0.036**	**0.917**

Table 6.5: KL and Pearson correlation between model and gold probability.

enough. This leads to 8545 movies in our dataset. We calculate the conditional probability $P(A|B) = \frac{P(A,B)}{P(B)} = \frac{\#rating(A,B)_{>4}/\#users}{\#rating(B)_{>4}/\#users}$. We randomly pick 100K conditional probabilities for training data and 10k probabilities for dev and test data [2].

We compare with several baselines: low-rank matrix factorization, complex bilinear factorization [82], and two hierarchical embedding methods, POE [52] and the Box Lattice [86]. Since the training matrix is asymmetric, we used separate embeddings for target and conditioned movies. For the complex bilinear model, we added one additional vector of parameters to capture the "imply" relation. We evaluate on the test set using KL divergence, Pearson correlation, and Spearman correlation with the ground truth probabilities.

From the results in Table 6.6, we can see that our smoothed box embedding method outperforms the original box lattice as well as all other baselines' performances, especially in Spearman correlation, the most relevant metric for recommendation, a ranking task.

[2] In the dev and test data, we also remove all the $P(A|B)$ where $P(B|A)$ appears in the training data.

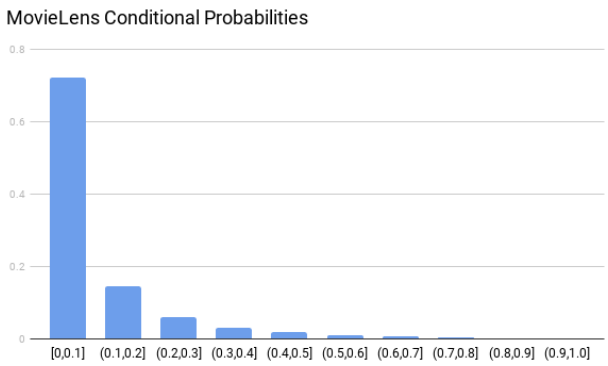


Figure 6.3: Distribution of probabilities in MovieLens Dataset.

The MovieLens dataset, while not truly sparse, has a large proportion of small probabilities which make it especially suitable for optimization by the smoothed model. The rough distribution of probabilities, in buckets of width 0.1, is shown in Figure 6.3.

	KL	Pearson R	Spearman R
Matrix Factorization	0.0173	0.8549	0.8374
Complex Bilinear Factorization	0.0141	0.8771	0.8636
POE	0.0170	0.8548	0.8511
Box	0.0147	0.8775	0.8768
Smoothed Box	**0.0138**	**0.8985**	**0.8977**

Table 6.6: Performance of the smoothed model, the original box model, and several baselines on MovieLens.

6.4.4.1 Initialization Sensitivity

We perform an additional set of experiments to determine the robustness of the smoothed box model to initialization. While the model is normally initialized randomly so that each box is a product of intervals that almost always overlaps with the other boxes, we would like to determine the models robustness to disjoint boxes in a principled way. While we can control initialization, we cannot always control the intermediate results of optimization, which may drive boxes to be disjoint, a condition from which the original, hard-edged box model may have difficulty recovering. So, parameterizing the initial distribution of

boxes with a minimum coordinate and a positive width, we adjust the width parameter so that approximately 0%, 20%, 50%, and 100% of boxes are disjoint at initialization before learning on the MovieLens dataset as usual. These results are presented in table 6.7. The smoothed model does not seem to suffer at all from disjoint initialization, while the performance of the original box model degrades significantly. From this we can speculate that part of the strength of the smoothed box model is its ability to smoothly optimize in the disjoint regime.

Approx. % Disjoint	KL		Pearson		Spearman	
	Box	Smooth	Box	Smooth	Box	Smooth
0%	0.0147	0.0138	0.8775	0.8985	0.8768	0.8977
20%	0.0172	0.0141	0.8668	0.8917	0.8608	0.8898
50%	0.0182	0.0141	0.8613	0.8908	0.8551	0.8910
100%	0.0346	0.0142	0.8401	0.8921	0.8167	0.8947

Table 6.7: Performance of the original box model and smoothed box model on MovieLens, as a function of different degrees of disjointness upon initialization.

6.5 Proof of Gaussian Overlap Formula

We wish to evaluate, for two variables X and Y, with associated smoothed indicators f and g,

$$\begin{aligned} f(x; a, b, \sigma^2) = \mathbb{1}_{[a,b]}(x) * \phi(x; \sigma^2) &= \int_{\mathbb{R}} \mathbb{1}_{[a,b]}(z)\phi(x - z; \sigma^2)dz \\ &= \int_a^b \phi(x - z; \sigma^2)dz \\ \tilde{P}_\phi(\text{Box}(X) \wedge \text{Box}(Y)) &= \int_{\mathbb{R}} f(x; a, b, \sigma_1^2)g(x; c, d, \sigma_2^2)dx \end{aligned} \tag{6.8}$$

Since the Gaussian kernel is normalized to have total integral equal to 1, so as not to change the overall areas of the boxes, the concrete formula is

$$\phi(z; \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-z^2}{2\sigma^2}}$$

Since the antiderivative of ϕ is the normal CDF, this may be recognized as the difference $\Phi(x; a, \sigma^2) - \Phi(x; b, \sigma^2)$, but this does not allow us to easily evaluate the integral of interest, which is the integral of the product of two such functions.

To evaluate (6.8), recall the identity [45, 85]

$$\int_{\mathbb{R}} \phi(x - \mu_1; \sigma_1^2)\phi(x - \mu_2; \sigma_2^2)dx = \phi(\mu_1 - \mu_2; \sigma_1^2 + \sigma_2^2) \tag{6.9}$$

For convenience, let $\tau := \frac{1}{\sqrt{\sigma_1^2+\sigma_2^2}}$. Applying Fubini's theorem and using (6.9), we have

$$\begin{aligned}
&\tilde{(}P)_\phi(\mathrm{Box}(X) \wedge \mathrm{Box}(Y)) \\
&\quad = \int_{\mathbb{R}} \int_a^b \phi(x - y; \sigma_1^2)\, dy \int_c^d \phi(x - z; \sigma_2^2)\, dz\, dx \\
&\quad = \int_c^d \int_a^b \phi(y - z; \tau^{-2})\, dy\, dz \\
&\quad = \int_c^d \int_a^b \Phi'(\tau(y - z))\tau\, dy\, dz \\
&\quad = \int_c^d \Phi(\tau(b - z)) - \Phi(\tau(a - z))\, dz \\
&\quad = \frac{-1}{\tau}(m_\Phi(\tau(b - d)) - m_\Phi(\tau(a - d)) - m_\Phi(\tau(b - c)) + m_\Phi(\tau(a - c)))
\end{aligned}$$

and therefore, with $\sigma = \tau^{-1}$,

$$\tilde{P}_\phi(\mathrm{Box}(X) \wedge \mathrm{Box}(Y)) = \sigma \left(m_\Phi(\tfrac{b-c}{\sigma}) + m_\Phi(\tfrac{a-d}{\sigma}) - m_\Phi(\tfrac{b-d}{\sigma}) - m_\Phi(\tfrac{a-c}{\sigma})\right)$$

as desired.

6.6 Conclusion and Future Work

We presented an approach to smoothing the energy and optimization landscape of probabilistic box embeddings and provided a theoretical justification for the smoothing. Due to a decreased number of hyper-parameters this model is easier to train, and, furthermore,

met or surpassed current state-of-the-art results on several interesting datasets. We further demonstrated that this model is particularly effective in the case of sparse data and more robust to poor initialization.

Tackling the learning problems presented by rich, geometrically-inspired embedding models is an open and challenging area of research, which this work is far from the last word on. This task will become even more pressing as the embedding structures become more complex, such as unions of boxes or other non-convex objects. To this end, we will continue to explore both function lattices, and constraint-based approaches to learning.

CHAPTER 7

REPRESENTATIONAL POWER OF BOX EMBEDDINGS

In this chapter we examine the representational capabilities of box embeddings, both for probabilistic and non-probabilistic data, and propose extensions to handle some failure cases. First, we examine some mathematical properties of the box lattice, especially as compared to order embeddings and box embeddings. Secondly, we investigate the suitability of box embeddings for predicting graph data. Finally, we identify a class of probability distributions which are not representable using box embeddings, propose an extension to remedy this deficiency, and test its efficacy on real-world data.

7.1 Properties of the Box Lattice

In this section, we cover some technical details about the box lattice model and its properties especially as compared to the order embedding model.

7.1.1 Non-Distributivity

A lattice is called *distributive* if the following identity holds for all members x, y, z:

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

Claim. *Order embeddings form a distributive lattice.*

Proof. This is a standard results on vector lattices shown in e.g. [92]

A non-distributive lattice is a strictly more general object, capable of modeling more objects since it does not necessarily need to fulfill the above identity for all triples x, y, z.

Claim. *The box lattice is non-distributive.*

Proof. Consider the box lattice in 1-dimension. Let $x = [0, 0.3]$, $y = [0.2, 0.6]$, and $z = [0.5, 1.0]$. Then $x \wedge (y \vee z) = [0.2, 0.3]$, but $(x \wedge y) \vee (x \wedge z) = [0, 0.6] \vee \perp = [0, 0.6]$.

This proves that the box lattice is a strict generalization of order embeddings, and not equivalent to order embeddings of any dimensionality. Additionally, our choice of an example containing disjoint elements hints at the importance of non-distributivity for our goal of modeling disjoint events.

7.1.2 Pseudocomplemented

A lattice is called *pseudocomplemented* if for every element x there exists a unique greatest element in the lattice x^* that is disjoint from x and $x \wedge x^* = \perp$. The box lattice is almost always pseudocomplemented, aside from symmetry concerns (for example, a perfectly centered cube in the 2-dimensional box lattice of side length < 1 has 4 possible equally large pseudocomplements. However any such symmetries can always be infinitesimally perturbed without breaking order structure so the box lattice is pseudocomplemented in a measure-theoretic sense. However, these pseudocomplements can be arbitrarily bad approximations of the true complement set of a box, with the worst case scenario coming from large, nearly-centered cubes.

7.2 Predicting Graphs with Box Embeddings

The box model can be viewed in two different ways. First, as a model of joint distributions $P(X_1, ..., X_N)$ over a set of binary random variables $\{X_i\}$. This approach is detailed in Chapter 5, and further examined later in this chapter. However, we are often interested in simply predicting edges in a graph or partial order. Box embeddings provide such a model, both with the box lattice, which creates a graph model based on inclusions, and with the box probability model, in which the probabilities can be thresholded to predict the edges of a DAG. That is, for a set of nodes $\{a_i\}$ we assign corresponding random variables $\{X_i\}$,

and for a threshold $\epsilon > 0$, we assign an edge from a_i to a_j if the conditional probability $P(X_j = 1|X_i = 1) \geq \epsilon$, and $P(X_j = 1|X_i = 1) > P(X = 1_i|X_j = 1)$. This gives a model of transitive directed graphs based on whether the conditional implication from the probability model meets a certain threshold. Setting $\epsilon = 1.0$ predicts an edge only if a box is fully included within another, corresponding to the partial order given by the box lattice model, or the model of Subramanian and Chakrabarti [80].

In this section, we examine the ability of box, Gaussian, and order embeddings to represent graphs. We prove that box embeddings can be used to predict consistent directed acyclic graphs through a simple thresholding procedure, but the same procedure does not work when used with Gaussian embeddings. We conduct numerical experiments giving strong evidence that this same procedure works with order embeddings as well.

7.2.1 Box Embedding

Assume we have a pairwise conditional probability table (CPD) between Bernoulli variables, that is, a matrix A such that each entry $A_{ij} = P(X_i = 1|X_j = 1)$ for a set of binary random variables $\{X_i\}$. Assume also that we have access to the unary marginals for each Bernoulli, and further that no unary marginals are exactly identical. If they are exactly identical, we can generate random independent Bernoulli parameters and their joint probability table, and take a small convex combination with that to infinitesimally perturb the statistics, so this proof is valid everywhere but on a set of measure 0 which we can approximate arbitrarily well.

Claim. *If all unary marginals are distinct, taking the elements of the pairwise CPD, removing the diagonal, and deleting an entry if $P(X_i = 1|X_j = 1) < P(X_i = 1|X_j = 1)$, that is if $A_{ij} < A_{ji}$, will result in a weighted adjacency matrix for an acyclic directed graph.*

Proof. Order the variables $X_1...X_n$ so that $P(X_i = 1) < P(X_j = 1)$ if $i < j$. Now an entry of the CPD $P(X_i = 1|X_j = 1) = P(X_i = 1, X_j = 1)/P(X_j = 1) = C_{ij}$ is less than $C_{ji} = P(X_i = 1, X_j = 1)/P(X_i = 1)$ if $P(X_i = 1) < P(X_j = 1)$. So with the variables

so ordered, if we use the CPD to create an adjacency matrix with an edge $C_{ij} = 1$ if and only if $P(X_i = 1) < P(X_j = 1)$, it will be upper triangular with 0 on the diagonal. This is a nilpotent matrix which means it is the adjacency matrix of an acyclic graph. This can be easily seen since the entries of A^k are the set of K-hop neighbors, and if this set eventually becomes empty, as in a nilpotent matrix, we have no cycles.

Since the labeling of our vertices is arbitrary, this means that our adjacency matrix created by the proposed asymmetrizing procedure is always acyclic since it is similar to an upper triangular matrix with zeros on the diagonal.

This holds as long as the unary marginals can always be ordered (which they can be except on a set of measure zero, which symmetry can be broken with a small perturbation).

7.2.2 KL Divergences and Gaussian Embeddings

Assume the same setup as section 7.2.1, however in this case the scores in the matrix come from (possibly thresholded if $A_{ij} - A_{ji} < c$) pairwise divergences between Gaussian embeddings. We demonstrate that Gaussian embeddings can not consistently generate graph structures from this same procedure as

Claim. *There exist graphs produced by the above procedure that do not lead to directed acyclic graphs if thresholded by deleting entries when $A_{ij} < A_{ji}$.*

Proof. Consider the following set of 5 2-dimensional Gaussians with diagonal covariance:

$$G_1 = \mathcal{N}(x_1; [-5, -3], \text{diag}([3, 7]))$$

$$G_2 = \mathcal{N}(x_2; [-3, 5], \text{diag}[(7, 4])$$

$$G_3 = \mathcal{N}(x_3; [-5, -6], \text{diag}([8, 1]))$$

$$G_4 = \mathcal{N}(x_4; [-7, 6], \text{diag}([5, 5]))$$

$$G_5 = \mathcal{N}(x_5; [9, 3], \text{diag}([5, 9]))$$

Applying asymmetrization and even pruning at a threshold of $c = 1$ (which is non-nilpotent and does affect edges) produces a cycle between nodes 5, 1, and 3. There are certain repeated numbers in the parameters, but this is not the cause of the issue. They are whole numbers for ease of exposition, they were randomly generated and many more examples can be created with arbitrary floating point numbers.

7.2.3 Order Embeddings

Order embeddings lie in an interesting middle ground between Gaussians and box embeddings. Interestingly, they cannot be consistently asymmetrized using the exact original energy function (4.2) given in Chapter 4. However, a small modification to the original energy function to change from the 2-norm to the 1-norm enables us to recover the desired property, i.e. they can be consistently asymmetrized into directed acyclic graphs according to the procedure in Section 7.2.1.

Claim. *Order embeddings using the energy $E(u, v) = \| \max(0, v - u)\|_2^2$ do not always result in a directed graph (even if all vectors have unique norms) if we set $A_{ij} = E(u_i, u_j)$ and select edges using the criterion $A_{ij} < A_{ji}$.*

Proof. Consider the following set of 3 3-dimensional order embeddings:

$$u_1 = (0.5, 0.6, 0.3)$$
$$u_2 = (0.8, 0.0, 0.7)$$
$$u_3 = (0.2, 0.3, 0.9).$$

Consider the matrix of pairwise energies,

$$A = \begin{pmatrix} 0 & 0.36 & 0.18 \\ 0.25 & 0 & 0.36 \\ 0.36 & 0.13 & 0 \end{pmatrix}.$$

If we set C equal to the asymmetrized version of A, we have

$$C = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

Since this matrix is not nilpotent, the graph has a cycle.

Claim. *Order embeddings using the energy $E(u, v) = \| \max(0, v - u)\|_1$ always result in a directed graph, as long as all vectors have unique 1-norms, if we set $A_{ij} = E(u_i, u_j)$ and select edges using the criterion $A_{ij} < A_{ji}$.*

Proof. Note that all order embedding vectors are in the positive orthant, so we have

$$\begin{aligned} E_1(u, v) + \|v\|_1 &= \sum_i \max(0, u_i - v_i) + \sum_i v_i \\ &= \sum_i \max(u_i, v_i) = E_1(v, u) + \|u\|_1 \\ E_1(u, v) - E_1(v, u) &= \|u\|_1 - \|v\|_1 \\ E_1(u, v) > E_1(v, u) &\iff \|u\|_1 > \|v\|_1. \end{aligned}$$

So, because all vectors have unique 1-norms, this means that if we order the graph nodes from biggest to smallest, then asymmetrization of the matrix of pairwise energies gives us nonzero entries only above the diagonal, which is a nilpotent matrix and thus we have a DAG.

7.3 Limitations on Representation

As briefly mentioned in Chapter 5, the box model cannot represent all probability distributions. The fact that each variable is associated with a box, and more generally a convex set, imposes limitations on the kind of probability distribution that can be modeled. In this

section, we will characterize one such type of distribution, and discuss extensions to the box model that enable it to model this distribution.

7.3.1 A Counterexample

When restricting the boxes to 1 or 2 dimensions, we can fairly easily come up with sets of pairwise marginal constraints between variables that are infeasible as a box model without adding more dimensions. Finding a counterexample that holds regardless of the dimension of the box model requires a bit more work. We will present the counterexample first geometrically in 2 dimensions, and then prove it in the general case.

Let A, B, and C be 3 binary random variables with the following joint distribution $P(A, B, C)$:

$$P(1,1,0) = P(1,0,1) = P(0,1,1) = \frac{1}{3}.$$

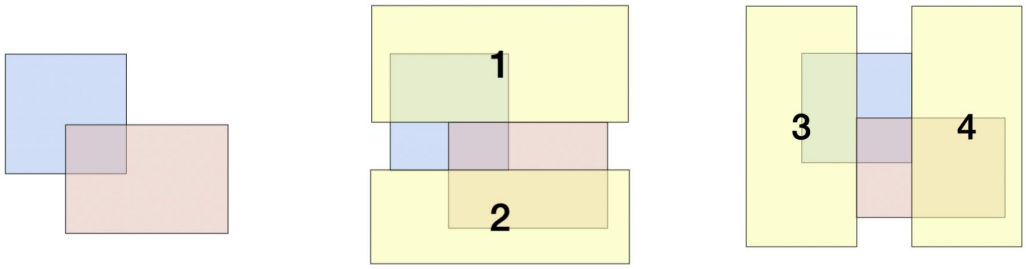
Note that this implies the constraints:

$$P_{\text{Box}}(\text{Box}(A) \cap \text{Box}(B)) = P_{\text{Box}}(\text{Box}(A) \cap \text{Box}(C)) = P_{\text{Box}}(\text{Box}(B) \cap \text{Box}(C)) = \frac{1}{3}$$
$$\text{Box}(A) \cap \text{Box}(B) \cap \text{Box}(C) = \emptyset$$

A visual proof of the impossibility of realizing these constraints in two dimensions is illustrated in Figure 7.1. In order to avoid a three-way intersection between the boxes for A, B, and C, the box for C must go in one of the four green regions, in any of which cases it cannot satisfy both pairwise marginal constraints.

As an aside, this counterexample is also used to demonstrate some representational limitations in the context of graphical models. This distribution is not representable by a Markov Random Field with only three variables, but it is representable by a factor graph.

We can generalize this counterexample and visual proof in two dimensions to box models with arbitrary numbers of dimensions in the following proposition.



(a) Sets A and B (b) Regions 1 and 2 (c) Regions 3 and 4

Figure 7.1: Any box that does not overlap $A \cap B$ will lie entirely within region 1, 2, 3 or 4, and thus cannot intersect both A and B and satisfy the constraints.

Proposition 2. *Let* $(\Omega, \mathcal{E}, P)$ *be a probability space, and let* $A, B, C : \Omega \to \{0, 1\}$ *be binary random variables such that*

$$P(A \wedge B),\ P(A \wedge C),\ P(B \wedge C) > 0$$

and

$$P(A \wedge B \wedge C) = 0.$$

Then there exists an $\epsilon > 0$ *and some event* $E \in \mathcal{E}$ *such that, for any box lattice model* Box*,*

$$|P_{\mathrm{Box}}(\mathrm{Box}(E)) - P(E)| > \epsilon.$$

Proof. Since $\mathrm{Box}(A)$ must intersect $\mathrm{Box}(B)$, we must have

$$\forall i \in \{1, \ldots, d\}, \quad a_i^{\vee} > b_i^{\wedge} \quad \text{and} \quad b_i^{\vee} > a_i^{\wedge}.$$

Similarly, since $\mathrm{Box}(A)$ must intersect $\mathrm{Box}(C)$,

$$\forall i \in \{1, \ldots, d\}, \quad c_i^{\vee} > a_i^{\wedge} \quad \text{and} \quad a_i^{\vee} > c_i^{\wedge}$$

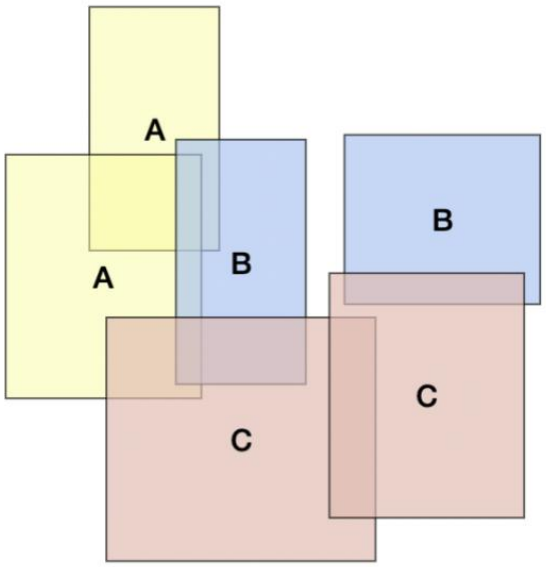


Figure 7.2: Union of boxes model in two dimensions.

and since $\mathrm{Box}(B)$ must intersect $\mathrm{Box}(C)$,

$$\forall i \in \{1, \ldots, d\}, \quad c_i^{\vee} > b_i^{\wedge} \quad \text{and} \quad b_i^{\vee} > c_i^{\wedge}.$$

Combining these constraints we find

$$\forall i \in \{1, \ldots, d\}, \quad c_i^{\vee} > \max(a_i^{\wedge}, b_i^{\wedge}) \quad \text{and} \quad c_i^{\wedge} < \min(a_i^{\vee}, b_i^{\vee}),$$

but this implies that implies that $\mathrm{Box}(C)$ intersects $\mathrm{Box}(A) \cap \mathrm{Box}(B)$. □

7.4 Unions of Boxes

The visual proof in Figure 7.1 suggests an extension of the model to remedy these limitations. We can see that the necessary overlap constraints could be achieved by allowing the variable C to correspond to a union of two different boxes, so it could overlap with both A and B separately. This allows us to define a *union of boxes* model. An example of this approach can be seen in Figure 7.2.

Definition 24 (Union of boxes model)**.** Let $(\Omega_{\text{Box}}, \mathcal{E}, P_{\text{Box}})$ be a probability space where $\Omega_{\text{Box}} \subseteq \mathbb{R}^d$. Let $\{\text{Box}(X_i)^j\}_{i \leq N, j \leq M}$ be a set of boxes in Ω_{Box}. The boxes are parameterized and defined as in Definition 23. Further, define a set of N indicator random variables, each of which is an indicator of the union of M component boxes,

$$S_i = \bigcup_{j \leq M} \text{Box}(X_i)^j, \quad X_i : \Omega_{\text{Box}} \to \{0, 1\} = \mathbb{1}_{S_i}.$$

We call the probability space and associated random variables X_i a *union of boxes model*.

This model, for a large enough number of boxes, can arbitrarily approximate any probability distribution over the X_i. For a naive construction, simply enumerate each element in the finite set of outcomes in Ω corresponding to each joint setting of the random variables $X_1, ..., X_N$. For each outcome, create a box with that volume for each of the variables whose value is 1. This can even be done in one dimension. There are of course much more efficient representations of many distributions, if we increase the dimensionality and merge contiguous boxes for the same variable.

As with the standard box model, probabilities of events can be calculated using inclusion-exclusion to calculate the volume of unions and complements of boxes under the measure $P_{\cup\text{Box}}$. However, the cost of this is exponential not only in the number of negated variables, but also the number of components, making it prohibitively computationally expensive to calculate even some simple marginal distributions.

7.5 Additive Box Model

While volume calculations in the union of boxes model require computing with unions of hyperrectangles and thus are exponentially complex, there is a way to associate multiple boxes with each variable while retaining linear complexity of volume calculation. The idea is to simply use a mixture of several box models, which we term the *additive box model*.

Definition 25 (Additive box model)**.** Let $\{(\Omega^j_{\text{Box}}, \mathcal{E}^j, P^j_{\text{Box}})\}_{j \leq M}$ be set of M probability spaces where $\Omega^j_{\text{Box}} \subseteq \mathbb{R}^d$. For each of the M component spaces, introduce a set of N boxes $\{\text{Box}(X_i)^j\}_{i \leq N}\}$ in the j'th component space. For each of these boxes, define an indicator random variable,

$$X^j_i : \Omega^j_{\text{Box}} \to \{0, 1\} = \mathbb{1}_{\text{Box}(X_i)^j}.$$

For the final part of the construction, we make use of the *disjoint* or *discriminated union* operation, written as $\sqcup$, which produces a union of two sets except that the resulting elements are tagged with the original set they come from. Finally, we define a new probability space and set of random variables,

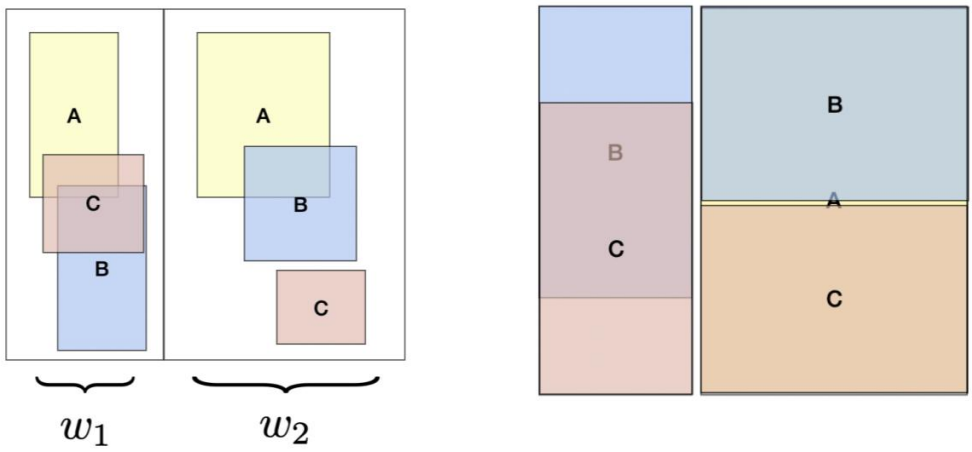
$$\begin{aligned}
\Omega_{\text{AddBox}} &= \bigsqcup_j \Omega^j_{\text{Box}} \\
X_i &= \bigsqcup_j X^j_i \\
P_{\text{AddBox}} &= \bigsqcup_j w_j P^j_{\text{Box}}
\end{aligned}$$

where w_j are a set of nonnegative weights with $\sum_j w_j = 1$. Let $\mathcal{E}$ be the smallest σ-algebra on the disjoint union such that its restriction to each component Ω^j_{Box} gives $\mathcal{E}^j$. We call the triple $(\Omega_{\text{AddBox}}, \mathcal{E}, P_{\text{AddBox}})$ and associated random variables X_i an *additive box model*.

This model can also represent arbitrary probability distributions through a similar construction to the one used for the union model, and in practice may be thought of as a union of boxes model where each component box are restricted to certain slices of the space, as demonstrated in Figure (7.3).

7.5.1 Universal Approximation

Both the additive box model and the union of boxes model can universally approximate any discrete probability distribution on a collection of binary random variables. We present



(a) Additive box model

(b) Solved counterexample

Figure 7.3: A demonstration of the space partitioning used by the additive box model, as well as a solution to the original counterexample.

here the proof in the case of the additive box model, since the union of boxes model can reproduce the additive box model by duplicating each mixture component, then scaling each one by the mixture weight and concatenating them into one model.

We first present a fairly impractical construction that allows for any joint distribution on binary variables to be perfectly reproduced by an additive box model.

Proposition 3 (Universal approximation)**.** *Let $P(X_1, ..., X_N)$ be a joint distribution over a set of binary random variables $\{X_i\}$. This gives us a pushforward measure and probability space (Ω, $\mathcal{E}$, P) where $\Omega = \{0, 1\}^N$. There exists an additive box model (Ω_{AddBox}, $\mathcal{F}$, P_{AddBox}) with variables $\{X_i'\}$ such that the pushforward measure P' on $\{0, 1\}^N$ induced by those variables matches P.*

Proof. For each outcome $\omega \in \Omega = \{0, 1\}^N$, some subset of variables X_i are equal to 1, some are equal to 0. For the former we say that ω is in the *support* of X_i. For this construction, we only need a 1-dimensional (interval) box model defined on the base sample space $[0, 1]$ under the uniform measure.

Iterating through each outcome in the sample space Ω, associate with the jth outcome ω_j such a box model $(\Omega^j_{\text{Box}}, \mathcal{E}^j, P^j_{\text{Box}}, \{X_i^{'j}\}_{i=1}^N)$. For each of the $X_i^{'j}$ that ω_j is in the support of, let $X_i^{'j} = \mathbb{1}_{[0,1]}$. Otherwise, let $X_i^{'j} = 0$. Let the mixture weight $w_j = P(\omega_j)$. Now, define the additive box model

$$\Omega_{\text{AddBox}} = \bigsqcup_j \Omega^j_{\text{Box}}$$
$$X'_i = \bigsqcup_j X_i^{'j}$$
$$P_{\text{AddBox}} = \bigsqcup_j w_j P^j_{\text{Box}}$$

where $\sqcup$ is the disjoint union. Now we can see that for any outcome ω_j, the preimage under the X'_i will pick out exactly one of the mixture components and retrieve the weight w_j corresponding to the correct probability. □

The previous construction is fairly contrived, always requiring a different mixture component for every possible event, no matter how simple the underlying dependence structure of the variables might be. We also have a separate approximation result which tracks more with the intuition of how these models can approximate complex distributions with reasonable numbers of components. Instead of directly approximating the binary distribution, we will look at approximating the preimages of the random variables.

Proposition 4 (Approximating preimages)**.** *Let $(\Omega, \mathcal{E}, P)$ be a probability space with $\Omega = [0,1]^d$, P is the standard Lebesgue measure, and $X_i : \Omega \to \{0,1\}$ be a set of binary random variables which are indicator functions of arbitrary Lebesgue-measurable subsets of Ω. Then for any $\epsilon > 0$ there exists an additive box model $(\Omega_{\text{AddBox}}, \mathcal{F}, P_{\text{AddBox}})$ with variables $\{X'_i\}$ such that for an outcome $\omega \in \{0,1\}^N$, we have $|P(\omega) - P'(\omega)| < \epsilon$ where P and P' are the pushforward measures.*

Proof. (Sketch.)

1. Approximate the Lebesgue-measurable support of each X_i with a finite union of disjoint boxes $\{A_i^j\}$.

2. Divide $[0,1]^d$ up into a disjoint union of boxes Ω_k such that no two of the boxes corresponding to the same X_i are in the same Ω_k. If a box crosses a boundary between Ω_k's, split it into two so that every box belongs to at most one Ω_k

3. For each of the Ω_k, create a mixture component and weigh that mixture component by the volume of Ω_k. The resulting additive model approximates the original P.

□

7.6 Experiments

In this section we present experiments with the additive box model in modeling language data that exhibits versions of the counterexample problem in practice.

7.6.1 Google Syntactic N-Grams

We test the additive box model on a dataset of subject/verb/object triples constructed from the Google Syntactic N-Gram Corpus [36]. The Syntactic N-gram corpus consists of parse tree fragments which appear frequently in Google books, in several different patterns. We use the *extended-biarcs* subsection of the data, and filter it down to triples with the syntactic annotations $(\text{NSUBJ}, \text{VERB}, \text{DOBJ})$ and discard stopwords and determiners to get subject/verb/object triples. The resulting dataset has 10,879,201 unique triples (words with multiple parts of speech are treated as different types) along with the count of times they appear in the corpus.

We construct a 90%/5%/5% train/validation/test split. For the train examples, the occurrence counts are used to produce a probability for each triple by normalizing over the total counts in the training data. For validation and testing, we create two different variant classification datasets. In the first, we treat the 5% splits of examples as positive, and

create an equal number of negative examples by randomly replacing a word in the triple with another word sharing the same syntactic role. In the second, we use the same positive examples but create the negative examples by sampling the replacement from a bucket of words having the same frequency as the original, with ten equally sized buckets.

The motivation for using this dataset to test out additive boxes is that it presents an example in modeling real-world data, where a version of our canonical counterexample problem might occur. Namely, there may be sets of subjects, verbs, and objects such that pairwise occurrences of them are common, but all three never show up together. For example the triple $(\text{CHICKEN}, \text{EATS}, \text{EGG})$ would never exist in a corpus, even though separately, $(\text{CHICKEN}, \text{EATS}, \cdot)$, $(\text{CHICKEN}, \cdot, \text{EGG})$, and $(\cdot, \text{EATS}, \text{EGG})$ are all plausible pairings. $(\text{MAN}, \text{BREATHES}, \text{UNDERWATER})$ is another such triple. In practice, when examining the data, we find many such examples where each pairwise probability is positive, but the triplet never occurs.

We compare the additive model to the single-box model on both the random-negatives and hard-negatives datasets. The models are trained by Bayesian hyperparameter optimization using the Weights and Biases framework [9], allotting five hours of computation time to each model and performing model selection using F1 score on the development set for the random-negatives task. More experimental details are given in Appendix A.4.1. Both models are trained and evaluated using the softbox objective on overlaps of SVO triples, with pseudo-probabilities calculated as

$$\tilde{P}((S,V,O)=1) = \prod_j \text{softplus}\big(\min(s_j^\vee, v_j^\vee, o_j^\vee) - \max(s_j^\wedge, v_j^\wedge, o_j^\wedge)\big),$$

and trained using binary cross entropy on the gold probabilities.

Results are given in Table 7.3.

We see a moderate improvement in F1 score on both datasets when using the additive box model, with larger improvements happening on the more difficult data set. The KL-

		F1	Acc	KL
Validation-Hard				
	Single	75.01	75.81	3.59E-08
	Additive	**75.87**	**76.50**	**3.08E-08**
Test-Hard				
	Single	74.97	75.76	3.40E-08
	Additive	**75.58**	**76.45**	**2.87E-08**
Validation-Random				
	Single	91.80	91.75	2.33E-08
	Additive	**92.00**	**91.93**	**2.07E-08**
Test-Random				
	Single	91.86	91.81	2.13E-08
	Additive	**92.06**	**91.99**	**1.86E-08**

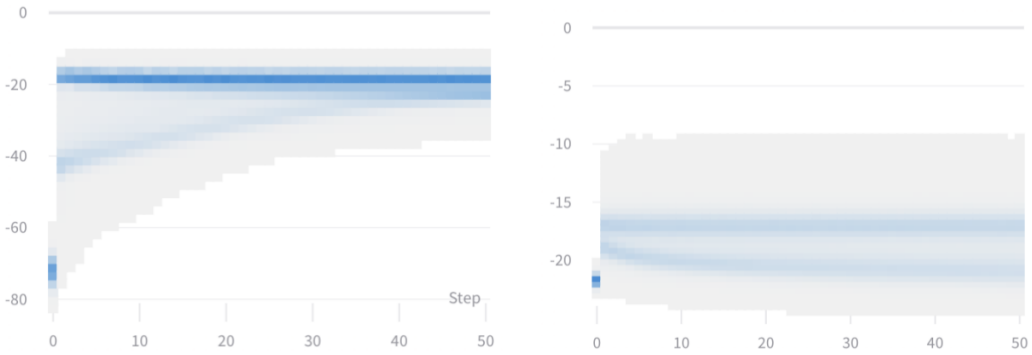
Table 7.3: Results on Google Syntactic N-Gram SVO classification task using single and additive box models.

divergence training objective is extremely low for both models, which is expected since the probabilities being regressed to are very small, making the loss hard to interpret.

We can confirm that the multi-box model is better able to create a separation between the classes by plotting histogram heatmaps as a function of training iteration of the model scores on the validation set with random negatives. The additive model properly assigns very low probabilities to negative examples, as the additive model has a clear bimodal distribution in model score on the validation set with random negatives, visible in Figure 7.4. The single box model, on the other hand, has less clear separation over time between the two classes, even as its classification performance slightly improves, it is not properly modeling the density.

7.7 Conclusion

In this chapter, we presented several properties of the box lattice and box probability models, especially regarding its ability to represent certain graphs and distributions. We characterized a class of probability distributions that are non-representable by box models of any dimension, and then introduced two new models, the union of boxes model, and



(a) Single box

(b) Additive box

Figure 7.4: Heatmaps of model output on validation set with random negatives, as a function of training iteration. The single-box model does not maintain good separation between the probability of positive and negative examples, as it cannot avoid spurious overlapping with just one box.

a practical alternative, the additive box model, to solve this counterexample. We demonstrated the ability of the additive box model to improve performance on a data set of subject/verb/object triples from the Google Syntactic N-Grams Corpus.

CHAPTER 8

CONCLUSION

8.1 Contributions

This thesis explores the nascent field of *geometric representation learning*, a method of data representation in which complex geometric objects are associated with data, and inference is performed using properties of those objects. Specifically, we introduce two models of geometric representation, the Gaussian embedding and box embedding models, and situate them within the related literature on hyperbolic embeddings, vector lattices, function space embeddings, and others.

We advocate for the use of geometric representations as an alternative to standard vector embeddings, in applications involving implicit hierarchies and order structures among domain objects, or where uncertainty over representations, multimodality, or breadth are important. Additionally, we encourage and develop the use of geometric representations based on structures in a latent event space, such as the box embeddings, as a novel approach to probabilistic modeling tasks over binary variables.

In addition to presenting the Gaussian and box embedding models, we develop a smooth learning approach for box embeddings inspired by a relaxation of the optimization problem using Gaussian convolutions, and finally an extension of the box probability model to an additive model with provable guarantees of representational fidelity even in cases where the original box embedding model would fail.

We focus our experiments on tasks in word-level entailment, weighted textual entailment, graph completion in transitive graphs, discrete density estimation, market basket problems, and common-sense modeling over language fragments. We choose these to

demonstrate the performance of geometric representations, and box embeddings in particular, on tasks that implicitly have latent hierarchical or graph structure, as well as complex density estimation tasks with latent dependencies, especially in the regime of pseudosparse data when the inductive bias is most important.

We provide a formalism for box embeddings in terms of sample spaces, σ-algebras of events, and probability measures. We provide several theoretical results about the probabilistic box embedding model and its extensions, including proofs of its ability to represent positive and negative correlation structures unrepresentable by competing methods, its ability to predict consistent graphs, identification of provably difficult counterexamples for box modeling, and their solutions.

8.2 Future Work

Geometric representation learning, and in particular probabilistic models based on representations such as box embeddings, present an exciting avenue for future work. In this vein, we have identified six directions as particularly interesting for future research.

8.2.1 Training Methods for Box Embeddings

The difficulty with training box embeddings comes from two sources. The first difficulty is that disjoint boxes create a situation where the overlap of two boxes has zero derivative with respect to the parameters. This makes it difficult to force two boxes which overlap in the ground truth together, if they do not already overlap in the model. The second difficulty is that boxes which are entirely contained within one another create a conditional probability $P(A = 1|B = 1)$ with zero derivative, and more generally, even the joint probability has a derivative which directs gradients towards simply shrinking or increasing the size of one box. This makes it difficult to force two boxes which do not overlap in the ground truth apart if they are not already somewhat disjoint.

This thesis presents two methods for training box embeddings. We introduce a method using a surrogate function to solve the first difficulty, and the second uses a smoothed approximation of the box overlap calculation which vanishes nowhere. Both methods rely on the general stochasticity during training to mitigate the second difficulty. Future work should focus on methods to removes those large subspaces of zero derivatives in the training landscape. These methods could use further smoothing of the optimization landscape, latent variable methods, or explicitly adding constraints through e.g. Lagrangian relaxation.

8.2.2 Deep Boxes

We advocate box embeddings as a replacement for vector embeddings in certain types of models and applications, where the embeddings themselves have a concrete association with variables which we want to perform inference over. However, embeddings are commonly used as features for downstream models, often processed through layers of nonlinear transformations, such as deep neural networks. Deep networks transform input embeddings into intermediate embedding vectors of hidden activations in the process of performing inference.

An analog of deep neural networks for box embeddings would transform boxes at each layer into other boxes, in a way that respects the geometry of the underlying representations. Developing deep architectures for box embeddings, and other geometric embeddings, is an exciting topic of future research.

8.2.3 Flexible Representations

In this work we introduce extensions of box embeddings to handle multiple box structured prototypes per variable, both in the intractable case of the union of boxes model, and its tractable restriction in the additive box model. Future work should investigate other types of latent geometric representations which allow for reasonable parameterizations, as well as tractable computation of quantities of interest such as overlap volumes. Products of convex (and nonconvex) polyhedra are one possible avenue for future work.

8.2.4 Inference in Large Joint Distributions

Box embeddings, when considered as a probabilistic model of binary vectors, present some interesting tradeoffs when compared to other models such as factor graphs and recurrent neural networks. Box probability models do not symmetrically represent outcomes with true assignments to random variables as opposed to false. That is, the difficulty in computing the probability of any configuration of random variables represented by box embeddings scales exponentially in the number of negated variables. Further, computing marginal probabilities over small subsets of random variables is tractable in a box model, while in a graphical model it may require the computation of normalization constants over the entire graph.

Evaluating any assignment of the random variables in a box model is equivalent to computing the volume of a union of hyperrectangles, which is an NP-hard problem. However, fully polynomial time approximation schemes do exist to solve this problem, up to additive error. Developing learning and inference methods, both MAP and marginal, in box probability models representing large sets of random variables, is a challenging and interesting avenue for future research.

8.2.5 Categorical Data

In this work, box embeddings are applied to modeling joint distributions over binary random variables. Much data of interest is modeled as categorical variables, taking on more than two values. Sometimes this data can be easily encoded in terms of binary vectors, as in multi-label prediction tasks, where a given input can be associated with any number of output labels. Other times, the encoding in terms of bit vectors is less straightforward, as in language modeling, where only one of a large set of vocabulary words should be predicted at any given time. Naively applying a box model to this, assigning one box to each word, would create a huge number of constraints among the boxes to avoid overlap. Even more importantly, naive encodings of categorical variables in terms of disjoint boxes

fail to harness the geometric properties of the box model to improve inference. We would like to be able to exploit similarities between different values of a categorical variable (e.g. similar words in a language model) to aid inference. However, it is unclear how to do this in a simple encoding of categorical data as a large set of box embeddings and binary variables with constraints.

8.2.6 Multirelational Data

Many real-world knowledge bases contain not just one relation- or edge-type, such as the hypernymy subgraph of WordNet, but many such relations. For example, WordNet itself is a hypergraph containing hypernymy as well as meronymy (part-of) relations. Furthermore, many KB relations are not transitive, and in fact may not even have the same entity type in all slots of a relation, such as $(\text{WORKS-AT}, \text{Bill Gates}, \text{Microsoft})$. It is not immediately clear how to represent all of these types of data using box embeddings or similar geometric representations. Multirelational and non-transitive extensions to the box model will be important to see geometric representations become more ubiquitous in knowledge representation.

APPENDIX A

EXPERIMENTAL DETAILS

A.1 Gaussian Experiments

All experiments with Gaussian embeddings were conducted using embeddings trained on a single iteration over the concatenated ukWaC and WaCkypedia corpora [5]. Hyperparameters were selected based on qualitative manual inspection of neighborhoods of retrieved similar embeddings, and covariance matrices. The parameters we used were:

```
dimension: 50 or 100
lambda_min: 0.1
lambda_max: 1000.0
learning_rate: 1.0
optimizer: Adagrad
```

A.2 Box Experiments

A.2.1 WordNet Parameters

Since the WordNet data has binary $0, 1$ links instead of calibrated probabilities, and the negative links are found from random negative sampling, we constrain the delta embedding to not update for negative samples during optimization. We found this was effective in preventing random negative samples from decreasing the volume of the boxes and creating artificially disjoint pairs.

The WordNet parameters that achieved best performance on the development set (whose train set performance we reported) are:

```
batch size: 800
dimension: 50
edge loss weight: 1.0
unary loss weight: 9.0
learning rate: 0.001
minimum dimension delta size: 1e-6
dimension-max regularization weight: 0.005
optimizer: Adam
```

For WordNet training with additional soft CPD edges, we use the same parameters. We also perform pruning on the generated CPD file. We only include $\langle t_1, t_2 \rangle$ pairs with probability ≥ 0.6 and the reverse pair $\langle t_2, t_1 \rangle \leq 0.4$ probability.

We tune the batch size of the model between 800 and 40000 because bigger batch size facilitates faster training. We also sweep over 1.0 to 9.0 for edge loss weight and 9.0 to 1.0 for the unary loss weight. The learning rate we tune in $\lambda \in \{0.001, 0.0001\}$. The minimum dimension delta size we tune in $\in \{0.01, 0.001, 0.0001, 0.00001, 0.000001\}$. The dimension-max regularization encourages the upper bound of box to be close 1.0 with an L1 penalty to prevent collapse. We perform parameter search in $\{0.0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$.

A.2.2 Flickr Parameters

The Flickr parameters that achieved best performance on the development set (whose train set performance we reported) are:

```
batch size: 512
dropout: 0.5
unary loss weight: 8.0
edge loss weight: 2.0
learning rate: 0.0001
minimum dimension delta size: 1e-6
optimizer: Adam
```

The LSTM parameters are initialized with Glorot initialization [35], as are the weight and bias parameters for the feedforward networks to produce the box minimums. The network to produce the Δ embedding is initialized from a uniform distribution from $[15.0, 15.50]$. We clip to zero for min embeddings (apply a ReLU), and apply a softplus to enforce the positivity and minimum dimension size constraints on the Δ embeddings.

We also sweep over 1.0 to 9.0 for edge loss weight and 9.0 to 1.0 for the unary loss weight. The learning rate $\lambda \in \{0.001, 0.0001\}$. We tried Glorot initialization with the Δ network as well, but since we wanted a high degree of overlap at the beginning of training, we simply swept over different uniform initialization ranges in $[5.0, 5.5]$, $[10.0, 10.5]$ and $[15.0, 15.5]$.

A.3 Smoothed Learning Experiments

A.3.1 WordNet Parameters

For the WordNet experiments, the model is evaluated every epoch on the development set for a large fixed number of epochs, and the best development model is used to score the test set. Baseline models are trained using the parameters of Vilnis et al. [86], with the smoothed model using hyperparameters determined on the development set.

```
batch size: 8000
dimension: 50
edge loss weight: 3.0
unary loss weight: 7.0
learning rate: 0.001
```

```
minimum dimension delta size: 1e-6
optimizer: Adam
sigma for softplus: 1.0
box min initialization: uniformly from 1e-4 to 1e-2
box delta initialization: uniformly from 0.9 to 0.99
global regularization method: l1
global regularization strength: 0.1
max training steps: 100,000
```

global regularization method and *regularization parameter* are applied to the join of all box embeddings.

A.3.2 Imbalanced WordNet Parameters

We follow the same routine as the WordNet experiments section to select best parameters. For the 12 experiments we conducted in this section, negative examples are generated randomly based on the ratio for each batch of positive examples. We do a parameter sweep for all models then choose the best result for each model as our final result.

```
batch size: 8000
dimension: 50
edge loss weight: 3.0
unary loss weight: 7.0
learning rate: 0.001
minimum dimension delta size: 1e-6
optimizer: Adam
sigma for softplus: 1.0
box min initialization: uniformly from 1e-4 to 1e-2
box delta initialization: uniformly from 0.9 to 0.99
global regularization method: l1
global regularization strength: 0.1
max training steps: 100,000
```

A.3.3 Flickr Parameters

The experimental setup uses the same architecture as Vilnis et al. [86] and [52], a single-layer LSTM that reads captions and produces a box embedding parameterized by *min* and *delta*. Embeddings are produced by feedforward networks on the output of the LSTM. The model is trained for a large fixed number of epochs, and tested on the development data at each epoch. The best development model is used to report test set score. Hyperparameters were determined on the development set.

```
batch size: 512
dropout: 0.5
dimension: 300
hidden layer size: 512
```

```
unary loss weight: 1.0
edge loss weight: 9.0
learning rate: 0.0001
optimizer: Adam
min feed forward network weight initialization: xavier_W
min feed forward network bias initialization: xavier_b
delta feed forward network weight initialization: xavier_w
delta feed forward network bias initialization:
uniform from -15 to -14.5
min parameterization: relu
delta parameterization: softplus
epoch: 10
```

A.3.4 MovieLens Parameters

For all MovieLens experiments, the model is evaluated every 50 steps on the development set, and optimization is stopped if the best development set score fails to improve after 200 steps. The best development model is used to score the test set.

```
batch size: 128
dimension: 50
learning rate: 0.001
unary loss weight: 0.001
edge loss weight: 0.999
optimizer: Adam
sigma for softplus: 1.0
box min initialization: uniformly from 1e-4 to 0.5
box delta initialization: uniformly from 0.9 to 0.999
```

A.4 Additive Box Experiments

A.4.1 Google Syntactic N-Gram Parameters

For the syntactic n-grams experiments, both the single- and additive-box models are tuned on F1 score on the development set with random negatives, each allotted 5 hours of compute time for Bayesian hyperparameter tuning using the Weights and Biases framework [9]. The following hyperparameters were used for the best performing single-box model:

```
batch size: 2^16
dimension: 62
learning rate: 0.04801
softplus_temp: 0.4099
epochs: 50
optimizer: Adam
```

For the additive box model, the following hyperparameters were selected:

```
batch size: 2^16
dimension: 38
learning rate: 0.1396
components: 8
softplus_temp: 1.249
epochs: 50
optimizer: Adam
```

BIBLIOGRAPHY

[1] M. A. Aizerman, E. A. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control,*, number 25 in Automation and Remote Control,, pages 821–837, 1964.

[2] Ben Athiwaratkun and Andrew Gordon Wilson. Multimodal word distributions. In *ACL*, 2017.

[3] Ben Athiwaratkun and Andrew Gordon Wilson. Hierarchical density order embeddings. In *International Conference on Learning Representations*, 2018.

[4] Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. Structured learning for taxonomy induction with belief propagation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1041–1051, 2014.

[5] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226, 2009.

[6] Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 23–32, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[7] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247. Association for Computational Linguistics, 2014.

[8] David Belanger and Andrew McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992, 2016.

[9] Lukas Biewald. Experiment tracking with weights and biases. 2020.

[10] Christopher M Bishop. Mixture density networks, 1994.

[11] Marián Boguná, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the internet with hyperbolic mapping. *Nature communications*, 1(1):1–8, 2010.

[12] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *ICLR*, 2018.

[13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[14] Shannon R Bowling, Mohammad T Khasawneh, Sittichai Kaewkuekool, and Byung R Cho. A logistic approximation to the cumulative normal distribution. *Journal of Industrial Engineering and Management*, 2(1), 2009.

[15] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *CoNLL*, 2016.

[16] Gerald G Brown and Herbert C Rutemiller. Means and variances of stochastic vector products with applications to random linear models. *Management Science*, 24(2): 210–216, 1977.

[17] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[18] Elia Bruni, Nam-Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *JAIR*, 2014.

[19] Y. J. Chu and T. H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 20, 1995.

[20] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[21] Stéphane Clinchant and Florent Perronnin. Textual similarity with a bag-of-embedded-words model. In *Proceedings of the 2013 Conference on the Theory of Information Retrieval*, ICTIR '13, pages 25:117–25:120, New York, NY, USA, 2013. ACM.

[22] Stéphane Clinchant and Florent Perronnin. Aggregating continuous word embeddings for information retrieval. *ACL 2013*, page 100, 2013.

[23] Michael Collins, Sanjoy Dasgupta, and Robert E Schapire. A generalization of principal components analysis to the exponential family. In *Advances in neural information processing systems*, pages 617–624, 2002.

[24] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[25] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R.A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, pages 391–407, 1990.

[26] Mathias Drton and Marloes H Maathuis. Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4:365–393, 2017.

[27] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12: 2121–2159, 2011.

[28] Katrin Erk. Representing words as regions in vector space. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 57–65. Association for Computational Linguistics, 2009.

[29] Manaal Faruqui and Chris Dyer. Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, USA, June 2014. Association for Computational Linguistics.

[30] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.

[31] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[32] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. *ICML*, 2018.

[33] Maayan Geffet and Ido Dagan. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 107–114. Association for Computational Linguistics, 2005.

[34] Robert Gens and Domingos Pedro. Learning the structure of sum-product networks. In *International conference on machine learning*, pages 873–880, 2013.

[35] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[36] Yoav Goldberg and Jon Orwant. A dataset of syntactic-ngrams over time from a very large corpus of english books. 2013.

[37] Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. Noisy activation functions. In *International Conference on Machine Learning*, pages 3059–3068, 2016.

[38] Caglar Gulcehre, Marcin Moczulski, Francesco Visin, and Yoshua Bengio. Mollifying networks. *arXiv preprint arXiv:1608.04980*, 2016.

[39] Z. S. Harris. *Mathematical Structures of Language*. Wiley, New York, 1968.

[40] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 623–632, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3794-6. doi: 10.1145/2806416.2806502.

[41] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3): 197–243, 1995.

[42] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *arXiv preprint arXiv:1408.3456*, 2014.

[43] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[44] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[45] Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844, 2004.

[46] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[47] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[48] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[49] Ryan Kiros, Richard Zemel, and Ruslan R Salakhutdinov. A multiplicative model for learning distributed text-based attribute representations. In *NIPS*, 2014.

[50] Shimaoka Kiyoshiyo, Muraoka Masayasu, Yamamoto Futo, Yotaro Watanabe, Naoaki Okazaki, and Kentaro Inui. Distribution representation of the meaning of words and phrases by a gaussian distribution. In *Language Processing Society 20th Annual Conference (In Japanese)*, 2014.

[51] Robert Kleinberg. Geographic routing using hyperbolic space. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pages 1902–1909. IEEE, 2007.

[52] Alice Lai and Julia Hockenmaier. Learning to predict denotational probabilities for modeling entailment. In *EACL*, 2017.

[53] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

[54] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

[55] Xiang Li, Luke Vilnis, and Andrew McCallum. Improved representation learning for predicting commonsense ontologies. *NIPS Workshop on Structured Prediction*, 2017.

[56] Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. Smoothing the geometry of probabilistic box embeddings. *ICLR*, 2019.

[57] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[58] Paul Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern recognition and artificial intelligence*, 116:374–388, 1976.

[59] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[60] George A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 1995.

[61] George A Miller and Walter G Charles. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28, 1991.

[62] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. 1969.

[63] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[64] Hossein Mobahi. Training recurrent neural networks by diffusion. *arXiv preprint arXiv:1601.04114*, 2016.

[65] Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

[66] Allen Newell. Report on a general problem solving program.

[67] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6338–6347. Curran Associates, Inc., 2017.

[68] Judea Pearl. *Probabilistic reasoning in intelligent systems*. 1988.

[69] Kaare Brandt Petersen. The matrix cookbook. 2006.

[70] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[71] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[72] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. 2013.

[73] Daniel M Roy and Yee W Teh. The mondrian process. In *Advances in neural information processing systems*, pages 1377–1384, 2009.

[74] Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October 1965. ISSN 0001-0782.

[75] D.E. Rumelhart, G.E. Hintont, and R.J. Williams. Learning representations by backpropagating errors. *Nature*, 323(6088):533–536, 1986.

[76] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[77] Gerard Salton. Some experiments in the generation of word and document associations. In *AFIPS Proceedings of the December 4-6, 1962, fall joint computer conference*, pages 234–250, 1962.

[78] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[79] Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[80] Sandeep Subramanian and Soumen Chakrabarti. New embedded representations and evaluation protocols for inferring transitive relations. *SIGIR 2018*, 2018.

[81] Sean R Szumlanski, Fernando Gomez, and Valerie K Sims. A new set of norms for semantic relatedness measures. In *ACL*, 2013.

[82] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080, 2016.

[83] Cristiano Varin, Nancy Reid, and David Firth. An overview of composite likelihood methods. *Statistica Sinica*, pages 5–42, 2011.

[84] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. In *ICLR*, 2016.

[85] Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. In *ICLR*, 2015.

[86] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. Probabilistic embedding of knowledge graphs with box lattice measures. In *ACL*. Association for Computational Linguistics, 2018.

[87] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2): 1–305, 2008.

[88] Warren Weaver. Translation. In *Machine translation of languages*, volume 14, pages 15–23, 1955.

[89] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, volume 11, pages 2764–2770, 2011.

[90] Eric P Xing, Michael I Jordan, Stuart Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002.

[91] Dongqiang Yang and David M. W. Powers. Verb similarity on the taxonomy of wordnet. In *In the 3rd International WordNet Conference (GWC-06), Jeju Island, Korea*, 2006.

[92] Adriaan C. Zaanen. *Introduction to Operator Theory in Riesz Spaces*. Springer Berlin Heidelberg, 1997. ISBN 9783642644870.