Multimodal Word Distributions

Ben Athiwaratkun
Cornell University
pa338@cornell.edu

Andrew Gordon Wilson
Cornell University
andrew@cornell.edu

Abstract

Word embeddings provide point representations of words containing useful semantic information. We introduce multimodal word distributions formed from Gaussian mixtures, for multiple word meanings, entailment, and rich uncertainty information. To learn these distributions, we propose an energy-based max-margin objective. We show that the resulting approach captures uniquely expressive semantic information, and outperforms alternatives, such as word2vec skip-grams, and Gaussian embeddings, on benchmark datasets such as word similarity and entailment.

1 Introduction

To model language, we must represent words. We can imagine representing every word with a binary one-hot vector corresponding to a dictionary position. But such a representation contains no valuable semantic information: distances between word vectors represent only differences in alphabetic ordering. Modern approaches, by contrast, learn to map words with similar meanings to nearby points in a vector space (Mikolov et al., 2013a), from large datasets such as Wikipedia. These learned word embeddings have become ubiquitous in predictive tasks.

Vilnis and McCallum (2014) recently proposed an alternative view, where words are represented by a whole probability distribution instead of a deterministic point vector. Specifically, they model each word by a Gaussian distribution, and learn its mean and covariance matrix from data. This approach generalizes any deterministic point embedding, which can be fully captured by the mean vector of the Gaussian distribution. Moreover, the full distribution provides much richer information than point estimates for characterizing words, representing probability mass and uncertainty across a set of semantics.

However, since a Gaussian distribution can have only one mode, the learned uncertainty in this representation can be overly diffuse for words with multiple distinct meanings (polysemies), in order for the model to assign *some* density to any plausible semantics (Vilnis and McCallum, 2014). Moreover, the mean of the Gaussian can be pulled in many opposing directions, leading to a biased distribution that centers its mass mostly around one meaning while leaving the others not well represented.

In this paper, we propose to represent each word with an expressive multimodal distribution, for multiple distinct meanings, entailment, heavy tailed uncertainty, and enhanced interpretability. For example, one mode of the word 'bank' could overlap with distributions for words such as 'finance' and 'money', and another mode could overlap with the distributions for 'river' and 'creek'. It is our contention that such flexibility is critical for both qualitatively learning about the meanings of words, and for optimal performance on many predictive tasks.

In particular, we model each word with a mixture of Gaussians (Section 3.1). We learn all the parameters of this mixture model using a maximum margin energy-based ranking objective (Joachims, 2002; Vilnis and McCallum, 2014) (Section 3.3), where the energy function describes the affinity between a pair of words. For analytic tractability with Gaussian mixtures, we use the inner product between probability distributions in a Hilbert space, known as the expected likelihood kernel (Jebara et al., 2004), as our energy function (Section 3.4). Additionally, we propose transformations for numerical stability and initialization A.2, resulting in a robust, straightforward, and

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1645–1656
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1151

scalable learning procedure, capable of training on a corpus with billions of words in days. We show that the model is able to automatically discover multiple meanings for words (Section 4.3), and significantly outperform other alternative methods across several tasks such as word similarity and entailment (Section 4.4, 4.5, 4.7). We have made code available at http://github.com/benathi/word2gm, where we implement our model in Tensorflow (Abadi et. al, 2015).

2 Related Work

In the past decade, there has been an explosion of interest in word vector representations. word2vec, arguably the most popular word embedding, uses continuous bag of words and skip-gram models, in conjunction with negative sampling for efficient conditional probability estimation (Mikolov et al., 2013a,b). Other popular approaches use feedforward (Bengio et al., 2003) and recurrent neural network language models (Mikolov et al., 2010, 2011b; Collobert and Weston, 2008) to predict missing words in sentences, producing hidden layers that can act as word embeddings that encode semantic information. They employ conditional probability estimation techniques, including hierarchical softmax (Mikolov et al., 2011a; Mnih and Hinton, 2008; Morin and Bengio, 2005) and noise contrastive estimation (Gutmann and Hyvärinen, 2012).

A different approach to learning word embeddings is through factorization of word co-occurrence matrices such as GloVe embeddings (Pennington et al., 2014). The matrix factorization approach has been shown to have an implicit connection with skip-gram and negative sampling Levy and Goldberg (2014). Bayesian matrix factorization where row and columns are modeled as Gaussians has been explored in Salakhutdinov and Mnih (2008) and provides a different probabilistic perspective of word embeddings.

In exciting recent work, Vilnis and McCallum (2014) propose a Gaussian distribution to model each word. Their approach is significantly more expressive than typical point embeddings, with the ability to represent concepts such as *entailment*, by having the distribution for one word (e.g. 'music') encompass the distributions for sets of related words ('jazz' and 'pop'). However, with a unimodal distribution, their approach cannot capture multiple distinct meanings, much like most deterministic approaches.

Recent work has also proposed deterministic embeddings that can capture polysemies, for example through a cluster centroid of context vectors (Huang et al., 2012), or an adapted skip-gram model with an EM algorithm to learn multiple latent representations per word (Tian et al., 2014). Neelakantan et al. (2014) also extends skip-gram with multiple prototype embeddings where the number of senses per word is determined by a non-parametric approach. Liu et al. (2015) learns topical embeddings based on latent topic models where each word is associated with multiple topics. Another related work by Nalisnick and Ravi (2015) models embeddings in infinite-dimensional space where each embedding can gradually represent incremental word sense if complex meanings are observed.

Probabilistic word embeddings have only recently begun to be explored, and have so far shown great promise. In this paper, we propose, to the best of our knowledge, the first probabilistic word embedding that can capture multiple meanings. We use a Gaussian mixture model which allows for a highly expressive distributions over words. At the same time, we retain scalability and analytic tractability with an expected likelihood kernel energy function for training. The model and training procedure harmonize to learn descriptive representations of words, with superior performance on several benchmarks.

3 Methodology

In this section, we introduce our Gaussian mixture (GM) model for word representations, and present a training method to learn the parameters of the Gaussian mixture. This method uses an energy-based maximum margin objective, where we wish to maximize the similarity of distributions of nearby words in sentences. We propose an energy function that compliments the GM model by retaining analytic tractability. We also provide critical practical details for numerical stability and initialization. The code for model training and evaluation is available at http://github.com/benathi/word2gm.

3.1 Word Representation

We represent each word w in a dictionary as a Gaussian mixture with K components. Specifically, the distribution of w, f_w, is given by the density

$$\begin{aligned} f_w(\vec{x}) &= \sum_{i=1}^{K} p_{w,i}\, \mathcal{N}\left[\vec{x}; \vec{\mu}_{w,i}, \Sigma_{w,i}\right] \qquad (1) \\ &= \sum_{i=1}^{K} \frac{p_{w,i}}{\sqrt{2\pi|\Sigma_{w,i}|}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_{w,i})^{\top}\Sigma_{w,i}^{-1}(\vec{x}-\vec{\mu}_{w,i})}, \end{aligned}$$

where $\sum_{i=1}^{K} p_{w,i} = 1$. The mean vectors $\vec{\mu}_{w,i}$ represent the location of the i^{th} component of word w, and are akin to the point embeddings provided by popular approaches like `word2vec`. $p_{w,i}$ represents the component probability (mixture weight), and $\Sigma_{w,i}$ is the component covariance matrix, containing uncertainty information. Our goal is to learn all of the model parameters $\vec{\mu}_{w,i}, p_{w,i}, \Sigma_{w,i}$ from a corpus of natural sentences to extract semantic information of words. Each Gaussian component's mean vector of word w can represent one of the word's distinct meanings. For instance, one component of a polysemous word such as 'rock' should represent the meaning related to 'stone' or 'pebbles', whereas another component should represent the meaning related to music such as 'jazz' or 'pop'. Figure 1 illustrates our word embedding model, and the difference between multimodal and unimodal representations, for words with multiple meanings.

3.2 Skip-Gram

The training objective for learning $\theta = \{\vec{\mu}_{w,i}, p_{w,i}, \Sigma_{w,i}\}$ draws inspiration from the continuous skip-gram model (Mikolov et al., 2013a), where word embeddings are trained to maximize the probability of observing a word given another nearby word. This procedure follows the *distributional hypothesis* that words occurring in natural contexts tend to be semantically related. For instance, the words 'jazz' and 'music' tend to occur near one another more often than 'jazz' and 'cat'; hence, 'jazz' and 'music' are more likely to be related. The learned word representation contains useful semantic information and can be used to perform a variety of NLP tasks such as word similarity analysis, sentiment classification, modelling word analogies, or as a preprocessed input for complex system such as statistical machine translation.

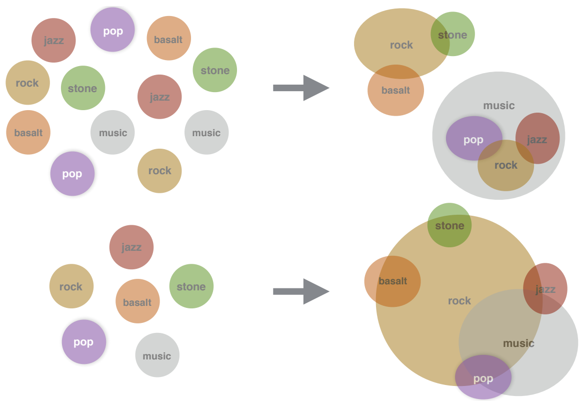

Figure 1: **Top:** A Gaussian Mixture embedding, where each component corresponds to a distinct meaning. Each Gaussian component is represented by an ellipsoid, whose center is specified by the mean vector and contour surface specified by the covariance matrix, reflecting subtleties in meaning and uncertainty. On the left, we show examples of Gaussian mixture distributions of words where Gaussian components are randomly initialized. After training, we see on the right that one component of the word 'rock' is closer to 'stone' and 'basalt', whereas the other component is closer to 'jazz' and 'pop'. We also demonstrate the entailment concept where the distribution of the more general word 'music' encapsulates words such as 'jazz', 'rock', 'pop'. **Bottom:** A Gaussian embedding model (Vilnis and McCallum, 2014). For words with multiple meanings, such as 'rock', the variance of the learned representation becomes unnecessarily large in order to assign some probability to both meanings. Moreover, the mean vector for such words can be pulled between two clusters, centering the mass of the distribution on a region which is far from certain meanings.

3.3 Energy-based Max-Margin Objective

Each sample in the objective consists of two pairs of words, (w, c) and (w, c'). w is sampled from a sentence in a corpus and c is a nearby word within a context window of length ℓ. For instance, a word $w =$ 'jazz' which occurs in the sentence 'I listen to jazz music' has context words ('I', 'listen', 'to' , 'music'). c' is a negative context word (e.g. 'airplane') obtained from random sampling.

The objective is to maximize the energy between words that occur near each other, w and c, and minimize the energy between w and its negative context c'. This approach is similar to negative sampling (Mikolov et al., 2013a,b), which contrasts the dot product between positive context pairs with negative context pairs. The energy function is a measure of similarity between distributions and will be discussed in Section 3.4.

We use a max-margin ranking objective (Joachims, 2002), used for Gaussian embeddings in Vilnis and McCallum (2014), which pushes the similarity of a word and its positive context higher than that of its negative context by a margin m:

$$\begin{aligned} L_\theta(w,c,c') = \max(0, \\ m - \log E_\theta(w,c) + \log E_\theta(w,c')) \end{aligned}$$

This objective can be minimized by mini-batch stochastic gradient descent with respect to the parameters $\theta = \{\vec{\mu}_{w,i}, p_{w,i}, \Sigma_{w,i}\}$ – the mean vectors, covariance matrices, and mixture weights – of our multimodal embedding in Eq. (1).

Word Sampling We use a word sampling scheme similar to the implementation in `word2vec` (Mikolov et al., 2013a,b) to balance the importance of frequent words and rare words. Frequent words such as 'the', 'a', 'to' are not as meaningful as relatively less frequent words such as 'dog', 'love', 'rock', and we are often more interested in learning the semantics of the less frequently observed words. We use subsampling to improve the performance of learning word vectors (Mikolov et al., 2013b). This technique discards word w_i with probability $P(w_i) = 1 - \sqrt{t/f(w_i)}$, where $f(w_i)$ is the frequency of word w_i in the training corpus and t is a frequency threshold.

To generate negative context words, each word type w_i is sampled according to a distribution $P_n(w_i) \propto U(w_i)^{3/4}$ which is a distorted version of the unigram distribution $U(w_i)$ that also serves to diminish the relative importance of frequent words. Both subsampling and the negative distribution choice are proven effective in `word2vec` training (Mikolov et al., 2013b).

3.4 Energy Function

For vector representations of words, a usual choice for similarity measure (energy function) is a dot product between two vectors. Our word representations are distributions instead of point vectors and therefore need a measure that reflects not only the point similarity, but also the uncertainty. We propose to use the *expected likelihood kernel*, which is a generalization of an inner product between vectors to an inner product between distributions (Jebara et al., 2004). That is,

$$E(f,g) = \int f(x)g(x)\,dx = \langle f,g\rangle_{L_2}$$

where $\langle\cdot,\cdot\rangle_{L_2}$ denotes the inner product in Hilbert space L_2. We choose this form of energy since it can be evaluated in a closed form given our choice of probabilistic embedding in Eq. (1).

For Gaussian mixtures f, g representing the words w_f, w_g, $f(x) = \sum_{i=1}^K p_i \mathcal{N}(x; \vec{\mu}_{f,i}, \Sigma_{f,i})$ and $g(x) = \sum_{i=1}^K q_i \mathcal{N}(x; \vec{\mu}_{g,i}, \Sigma_{g,i})$, $\sum_{i=1}^K p_i = 1$, and $\sum_{i=1}^K q_i = 1$, we find (see Section A.1) the log energy is

$$\log E_\theta(f,g) = \log \sum_{j=1}^K \sum_{i=1}^K p_i q_j e^{\xi_{i,j}} \tag{2}$$

where

$$\begin{aligned} \xi_{i,j} &\equiv \log \mathcal{N}(0; \vec{\mu}_{f,i} - \vec{\mu}_{g,j}, \Sigma_{f,i} + \Sigma_{g,j}) \\ &= -\frac{1}{2}\log\det(\Sigma_{f,i} + \Sigma_{g,j}) - \frac{D}{2}\log(2\pi) \\ -\frac{1}{2}&(\vec{\mu}_{f,i} - \vec{\mu}_{g,j})^\top (\Sigma_{f,i} + \Sigma_{g,j})^{-1} (\vec{\mu}_{f,i} - \vec{\mu}_{g,j}) \end{aligned} \tag{3}$$

We call the term $\xi_{i,j}$ partial (log) energy. Observe that this term captures the similarity between the i^{th} meaning of word w_f and the j^{th} meaning of word w_g. The total energy in Equation 2 is the sum of possible pairs of partial energies, weighted accordingly by the mixture probabilities p_i and q_j.

The term $-(\vec{\mu}_{f,i} - \vec{\mu}_{g,j})^\top (\Sigma_{f,i} + \Sigma_{g,j})^{-1} (\vec{\mu}_{f,i} - \vec{\mu}_{g,j})$ in $\xi_{i,j}$ explains the difference in mean vectors of semantic pair (w_f, i) and (w_g, j). If the semantic uncertainty (covariance) for both pairs are low, this term has more importance relative to other terms due to the inverse covariance scaling. We observe that the loss function L_θ in Section 3.3 attains a low value when $E_\theta(w,c)$ is relatively high. High values of $E_\theta(w,c)$ can be achieved when the component means across different words $\vec{\mu}_{f,i}$ and $\vec{\mu}_{g,j}$ are close together (e.g., similar point representations). High energy can also be achieved by large values of $\Sigma_{f,i}$ and $\Sigma_{g,j}$, which washes out the importance of the mean vector difference. The term $-\log\det(\Sigma_{f,i} + \Sigma_{g,j})$ serves as a regularizer that prevents the covariances from being pushed too high at the expense of learning a good mean embedding.

At the beginning of training, $\xi_{i,j}$ roughly are on the same scale among all pairs (i, j)'s. During this time, all components learn the signals from the word occurrences equally. As training progresses and the semantic representation of each mixture becomes more clear, there can be one term of $\xi_{i,j}$'s that is predominantly higher than other terms, giving rise to a semantic pair that is most related.

The negative KL divergence is another sensible choice of energy function, providing an asymmetric metric between word distributions. However, unlike the expected likelihood kernel, KL divergence does not have a closed form if the two distributions are Gaussian mixtures.

4 Experiments

We have introduced a model for multi-prototype embeddings, which expressively captures word meanings with whole probability distributions. We show that our combination of energy and objective functions, proposed in Section 3, enables one to learn interpretable multimodal distributions through unsupervised training, for describing words with multiple distinct meanings. By representing multiple distinct meanings, our model also reduces the unnecessarily large variance of a Gaussian embedding model, and has improved results on word entailment tasks.

To learn the parameters of the proposed mixture model, we train on a concatenation of two datasets: `UKWAC` (2.5 billion tokens) and `Wackypedia` (1 billion tokens) (Baroni et al., 2009). We discard words that occur fewer than 100 times in the corpus, which results in a vocabulary size of $314,129$ words. Our word sampling scheme, described at the end of Section 4.3, is similar to that of `word2vec` with one negative context word for each positive context word.

After training, we obtain learned parameters $\{\vec{\mu}_{w,i}, \Sigma_{w,i}, p_i\}_{i=1}^{K}$ for each word w. We treat the mean vector $\vec{\mu}_{w,i}$ as the embedding of the i^{th} mixture component with the covariance matrix $\Sigma_{w,i}$ representing its subtlety and uncertainty. We perform qualitative evaluation to show that our embeddings learn meaningful multi-prototype representations and compare to existing models using a quantitative evaluation on word similarity datasets and word entailment.

We name our model as Word to Gaussian Mixture (`w2gm`) in constrast to Word to Gaussian (`w2g`) (Vilnis and McCallum, 2014). Unless stated otherwise, `w2g` refers to our implementation of `w2gm` model with one mixture component.

4.1 Hyperparameters

Unless stated otherwise, we experiment with $K = 2$ components for the `w2gm` model, but we have results and discussion of $K = 3$ at the end of section 4.3. We primarily consider the spherical case for computational efficiency. We note that for diagonal or spherical covariances, the energy can be computed very efficiently since the matrix inversion would simply require $\mathcal{O}(d)$ computation instead of $\mathcal{O}(d^3)$ for a full matrix. Empirically, we have found diagonal covariance matrices become roughly spherical after training. Indeed, for these relatively high dimensional embeddings, there are sufficient degrees of freedom for the mean vectors to be learned such that the covariance matrices need not be asymmetric. Therefore, we perform all evaluations with spherical covariance models.

Models used for evaluation have dimension $D = 50$ and use context window $\ell = 10$ unless stated otherwise. We provide additional hyperparameters and training details in the supplementary material (A.2).

4.2 Similarity Measures

Since our word embeddings contain multiple vectors and uncertainty parameters per word, we use the following measures that generalizes similarity scores. These measures pick out the component pair with maximum similarity and therefore determine the meanings that are most relevant.

4.2.1 Expected Likelihood Kernel

A natural choice for a similarity score is the expected likelihood kernel, an inner product between distributions, which we discussed in Section 3.4. This metric incorporates the uncertainty from the covariance matrices in addition to the similarity between the mean vectors.

4.2.2 Maximum Cosine Similarity

This metric measures the maximum similarity of mean vectors among all pairs of mixture components between distributions f and g. That is, $d(f, g) = \max_{i,j=1,\ldots,K} \frac{\langle \boldsymbol{\mu}_{f,i}, \boldsymbol{\mu}_{g,j} \rangle}{||\boldsymbol{\mu}_{f,i}|| \cdot ||\boldsymbol{\mu}_{g,j}||}$, which corresponds to matching the meanings of f and g that are the most similar. For a Gaussian embedding, maximum similarity reduces to the usual cosine similarity.

Word	Co.	Nearest Neighbors
rock	0	basalt:1, boulder:1, boulders:0, stalagmites:0, stalactites:0, rocks:1, sand:0, quartzite:1, bedrock:0
rock	1	rock/:1, ska:0, funk:1, pop-rock:1, punk:1, indie-rock:0, band:0, indie:0, pop:1
bank	0	banks:1, mouth:1, river:1, River:0, confluence:0, waterway:1, downstream:1, upstream:0, dammed:0
bank	1	banks:0, banking:1, banker:0, Banks:1, bankas:1, Citibank:1, Interbank:1, Bankers:0, transactions:1
Apple	0	Strawberry:0, Tomato:1, Raspberry:1, Blackberry:1, Apples:0, Pineapple:1, Grape:1, Lemon:0
Apple	1	Macintosh:1, Mac:1, OS:1, Amiga:0, Compaq:0, Atari:1, PC:1, Windows:0, iMac:0
star	0	stars:0, Quaid:0, starlet:0, Dafoe:0, Stallone:0, Geena:0, Niro:0, Zeta-Jones:1, superstar:0
star	1	stars:1, brightest:0, Milky:0, constellation:1, stellar:0, nebula:1, galactic:1, supernova:1, Ophiuchus:1
cell	0	cellular:0, Nextel:0, 2-line:0, Sprint:0, phones.:1, pda:1, handset:0, handsets:1, pushbuttons:0
cell	1	cytoplasm:0, vesicle:0, cytoplasmic:1, macrophages:0, secreted:1, membrane:0, mitotic:0, endocytosis:1
left	0	After:1, back:0, finally:1, eventually:0, broke:0, joined:1, returned:1, after:1, soon:0
left	1	right-hand:0, hand:0, right:0, left-hand:0, lefthand:0, arrow:0, turn:0, righthand:0, Left:0

Word	Nearest Neighbors
rock	band, bands, Rock, indie, Stones, breakbeat, punk, electronica, funk
bank	banks, banking, trader, trading, Bank, capital, Banco, bankers, cash
Apple	Macintosh, Microsoft, Windows, Macs, Lite, Intel, Desktop, WordPerfect, Mac
star	stars, stellar, brightest, Stars, Galaxy, Stardust, eclipsing, stars., Star
cell	cells, DNA, cellular, cytoplasm, membrane, peptide, macrophages, suppressor, vesicles
left	leaving, turned, back, then, After, after, immediately, broke, end

Table 1: Nearest neighbors based on cosine similarity between the mean vectors of Gaussian components for Gaussian mixture embedding (top) (for $K = 2$) and Gaussian embedding (bottom). The notation `w:i` denotes the i^{th} mixture component of the word `w`.

4.2.3 Minimum Euclidean Distance

Cosine similarity is popular for evaluating embeddings. However, our training objective directly involves the Euclidean distance in Eq. (3), as opposed to dot product of vectors such as in `word2vec`. Therefore, we also consider the Euclidean metric: $d(f, g) = \min_{i,j=1,\dots,K} [||\boldsymbol{\mu}_{f,i} - \boldsymbol{\mu}_{g,j}||]$.

4.3 Qualitative Evaluation

In Table 1, we show examples of polysemous words and their nearest neighbors in the embedding space to demonstrate that our trained embeddings capture multiple word senses. For instance, a word such as 'rock' that could mean either 'stone' or 'rock music' should have each of its meanings represented by a distinct Gaussian component. Our results for a mixture of two Gaussians model confirm this hypothesis, where we observe that the 0^{th} component of 'rock' being related to ('basalt', 'boulders') and the 1^{st} component being related to ('indie', 'funk', 'hip-hop'). Similarly, the word `bank` has its 0^{th} component representing the river bank and the 1^{st} component representing the financial bank.

By contrast, in Table 1 (bottom), see that for Gaussian embeddings with one mixture component, nearest neighbors of polysemous words are predominantly related to a single meaning. For instance, 'rock' mostly has neighbors related to rock music and 'bank' mostly related to the financial bank. The alternative meanings of these polysemous words are not well represented in the embeddings. As a numerical example, the cosine similarity between 'rock' and 'stone' for the Gaussian representation of Vilnis and McCallum (2014) is only 0.029, much lower than the cosine similarity 0.586 between the 0^{th} component of 'rock' and 'stone' in our multimodal representation.

In cases where a word only has a single popular meaning, the mixture components can be fairly close; for instance, one component of 'stone' is close to ('stones', 'stonework', 'slab') and the other to ('carving, 'relic', 'excavated'), which reflects subtle variations in meanings. In general, the mixture can give properties such as heavy tails and more interesting unimodal characterizations of uncertainty than could be described by a single Gaussian.

Embedding Visualization We provide an interactive visualization as part of our code repository: `https://github.com/benathi/word2gm#visualization` that allows real-time queries of words' nearest neighbors (in the `embeddings` tab) for $K = 1, 2, 3$ components. We use a notation similar to that of Table 1, where a token `w:i` represents the component `i` of a word `w`. For instance, if in the $K = 2$ link we search for `bank:0`, we obtain the nearest neighbors such as `river:1, confluence:0, waterway:1`, which indicates that the 0th component of 'bank' has the meaning 'river bank'. On the other hand, searching for `bank:1` yields nearby words such as `banking:1, banker:0, ATM:0`, indicating that this component is close to the 'financial bank'. We also have a visualization of a unimodal (`w2g`) for comparison in the $K = 1$ link.

In addition, the embedding link for our Gaussian mixture model with $K = 3$ mixture components can learn three distinct meanings. For instance, each of the three components of 'cell' is close to ('keypad', 'digits'), ('incarcerated', 'inmate') or ('tissue', 'antibody'), indicating that the distribution captures the concept of 'cellphone', 'jail cell', or 'biological cell', respectively. Due to the limited number of words with more than 2 meanings, our model with $K = 3$ does not generally offer substantial performance differences to our model with $K = 2$; hence, we do not further display $K = 3$ results for compactness.

4.4 Word Similarity

We evaluate our embeddings on several standard word similarity datasets, namely, SimLex (Hill et al., 2014), WS or WordSim-353, WS-S (similarity), WS-R (relatedness) (Finkelstein et al., 2002), MEN (Bruni et al., 2014), MC (Miller and Charles, 1991), RG (Rubenstein and Goodenough, 1965), YP (Yang and Powers, 2006), MTurk(-287,-771) (Radinsky et al., 2011; Halawi et al., 2012), and RW (Luong et al., 2013). Each dataset contains a list of word pairs with a human score of how related or similar the two words are.

We calculate the Spearman correlation (Spearman, 1904) between the labels and our scores generated by the embeddings. The Spearman correlation is a rank-based correlation measure that assesses how well the scores describe the true labels.

The correlation results are shown in Table 2 using the scores generated from the expected likelihood kernel, maximum cosine similarity, and maximum Euclidean distance.

We show the results of our Gaussian mixture model and compare the performance with that of `word2vec` and the original Gaussian embedding by Vilnis and McCallum (2014). We note that our model of a unimodal Gaussian embedding `w2g` also outperforms the original model, which differs in model hyperparameters and initialization, for most datasets. Our multi-prototype model `w2gm` also performs better than skip-gram or Gaussian embedding methods on many datasets, namely, `WS, WS-R, MEN, MC, RG, YP, MT-287, RW`. The maximum cosine similarity yields the best performance on most datasets; however, the minimum Euclidean distance is a better metric for the datasets `MC` and `RW`. These results are consistent for both the single-prototype and the multi-prototype models.

We also compare out results on WordSim-353 with the multi-prototype embedding method by Huang et al. (2012) and Neelakantan et al. (2014), shown in Table 3. We observe that our single-prototype model `w2g` is competitive compared to models by Huang et al. (2012), even without using a corpus with stop words removed. This could be due to the auto-calibration of importance via the covariance learning which decrease the importance of very frequent words such as 'the', 'to', 'a', etc. Moreover, our multi-prototype model substantially outperforms the model of Huang et al. (2012) and the MSSG model of Neelakantan et al. (2014) on the WordSim-353 dataset.

4.5 Word Similarity for Polysemous Words

We use the dataset SCWS introduced by Huang et al. (2012), where word pairs are chosen to have variations in meanings of polysemous and homonymous words.

We compare our method with multiprototype models by `Huang` (Huang et al., 2012), `Tian` (Tian et al., 2014), `Chen` (Chen et al., 2014), and `MSSG` model by (Neelakantan et al., 2014). We note that `Chen` model uses an external lexical source `WordNet` that gives it an extra advantage.

We use many metrics to calculate the scores for the Spearman correlation. `MaxSim` refers to the maximum cosine similarity. `AveSim` is the average of cosine similarities with respect to the component probabilities.

In Table 4, the model `w2g` performs the best among all single-prototype models for either 50 or 200 vector dimensions. Our model `w2gm` performs competitively compared to other multi-prototype models. In SCWS, the gain in flexibility in moving to a probability density approach appears to dominate over the effects of using a multi-prototype. In most other examples, we see `w2gm` surpass `w2g`, where the multi-prototype structure is just as important for good performance as the

Dataset	sg*	w2g*	w2g/mc	w2g/el	w2g/me	w2gm/mc	w2gm/el	w2gm/me
SL	29.39	**32.23**	29.35	25.44	25.43	29.31	26.02	27.59
WS	59.89	65.49	71.53	61.51	64.04	**73.47**	62.85	66.39
WS-S	69.86	76.15	76.70	70.57	72.3	**76.73**	70.08	73.3
WS-R	53.03	58.96	68.34	54.4	55.43	**71.75**	57.98	60.13
MEN	70.27	71.31	72.58	67.81	65.53	**73.55**	68.5	67.7
MC	63.96	70.41	76.48	72.70	**80.66**	79.08	76.75	80.33
RG	70.01	71	73.30	72.29	72.12	**74.51**	71.55	73.52
YP	39.34	41.5	41.96	38.38	36.41	**45.07**	39.18	38.58
MT-287	-	-	64.79	57.5	58.31	**66.60**	57.24	60.61
MT-771	-	-	**60.86**	55.89	54.12	60.82	57.26	56.43
RW	-	-	28.78	32.34	33.16	28.62	31.64	**35.27**

Table 2: Spearman correlation for word similarity datasets. The models `sg, w2g, w2gm` denote `word2vec` skip-gram, Gaussian embedding, and Gaussian mixture embedding (K=2). The measures `mc, el, me` denote maximum cosine similarity, expected likelihood kernel, and minimum Euclidean distance. For each of `w2g` and `w2gm`, we underline the similarity metric with the best score. For each dataset, we boldface the score with the best performance across all models. The correlation scores for `sg⋆, w2g⋆` are taken from Vilnis and McCallum (2014) and correspond to cosine distance.

Model	$\rho \times 100$
Huang	64.2
Huang*	71.3
MSSG 50D	63.2
MSSG 300D	71.2
w2g	70.9
w2gm	**73.5**

Table 3: Spearman's correlation (ρ) on WordSim-353 datasets for our Word to Gaussian Mixture embeddings, as well as the multi-prototype embedding by Huang et al. (2012) and the MSSG model by Neelakantan et al. (2014). `Huang⋆` is trained using data with all stop words removed. All models have dimension $D = 50$ except for MSSG 300D with $D = 300$ which is still outperformed by our `w2gm` model.

probabilistic representation.

4.6 Reduction in Variance of Polysemous Words

One motivation for our Gaussian mixture embedding is to model word uncertainty more accurately than Gaussian embeddings, which can have overly large variances for polysemous words (in order to assign some mass to all of the distinct meanings). We see that our Gaussian mixture model does indeed reduce the variances of each component for such words. For instance, we observe that the word `rock` in `w2g` has much higher variance per dimension ($e^{-1.8} \approx 1.65$) compared to that of Gaussian components of `rock` in `w2gm` (which has variance of roughly $e^{-2.5} \approx 0.82$). We also

Model	Dimension	$\rho \times 100$
word2vec skip-gram	50	61.7
Huang-S	50	58.6
w2g	50	**64.7**
Chen-S	200	64.2
w2g	200	**66.2**
Huang-M AvgSim	50	62.8
Tian-M MaxSim	50	63.6
w2gm MaxSim	50	62.7
MSSG AvgSim	50	**64.2**
Chen-M AvgSim	200	**66.2**
w2gm MaxSim	200	65.5

Table 4: Spearman's correlation ρ on dataset SCWS. We show the results for single prototype (top) and multi-prototype (bottom) The suffix `-(S,M)` refers to single and multiple prototype models, respectively.

see, in the next section, that the Gaussian mixture model has desirable quantitative behavior for word entailment.

4.7 Word Entailment

We evaluate our embeddings on the word entailment dataset from Baroni et al. (2012). The lexical entailment between words is denoted by $w_1 \models w_2$ which means that all instances of w_1 are w_2. The entailment dataset contains positive pairs such as *aircraft* $\models$ *vehicle* and negative pairs such as *aircraft* $\not\models$ *insect*.

We generate entailment scores of word pairs and find the best threshold, measured by Average Precision (AP) or F1 score, which identifies negative versus positive entailment. We use the max-

MODEL	SCORE	BEST AP	BEST F1
W2G (5)	COS	73.1	76.4
W2G (5)	KL	73.7	76.0
W2GM (5)	COS	73.6	76.3
W2GM (5)	KL	75.7	77.9
W2G (10)	COS	73.0	76.1
W2G (10)	KL	74.2	76.1
W2GM (10)	COS	72.9	75.6
W2GM (10)	KL	74.7	76.3

Table 5: Entailment results for models `w2g` and `w2gm` with window size 5 and 10. The metrics used are the maximum cosine similarity, or the maximum negative KL divergence. We calculate the best average precision as well as the best F1 score. In most cases, `w2gm` outperforms `w2g` for describing entailment.

imum cosine similarity and the minimum KL divergence, $d(f, g) = \min_{i,j=1,\dots,K} KL(f||g)$, for entailment scores. The minimum KL divergence is similar to the maximum cosine similarity, but also incorporates the embedding uncertainty. In addition, KL divergence is an asymmetric measure, which is more suitable for certain tasks such as word entailment where a relationship is unidirectional. For instance, $w_1 \models w_2$ does not imply $w_2 \models w_1$. Indeed, *aircraft* $\models$ *vehicle* does not imply *vehicle* $\models$ *aircraft*, since all aircraft are vehicles but not all vehicles are aircraft. The difference between $KL(w_1||w_2)$ versus $KL(w_2||w_1)$ distinguishes which word distribution encompasses another distribution, as demonstrated in Figure 1.

Table 5 shows the results of our `w2gm` model versus the Gaussian embedding model `w2g`. We observe a trend for both models with window size 5 and 10 that the KL metric yields improvement (both AP and F1) over cosine similarity. In addition, `w2gm` has a better performance compared to `w2g`. The multi-prototype model estimates the meaning uncertainty better since it is no longer constrained to be unimodal, leading to better characterizations of entailment. On the other hand, the Gaussian embedding model suffers from large variance problem for polysemous words, which results in less informative word distribution and inferior entailment scores.

5 Discussion

We introduced a model that represents words with expressive multimodal distributions formed from Gaussian mixtures. To learn the properties of each mixture, we proposed an analytic energy function for combination with a maximum margin objective. The resulting embeddings capture different semantics of polysemous words, uncertainty, and entailment, and also perform favorably on word similarity benchmarks.

Elsewhere, latent probabilistic representations are proving to be exceptionally valuable, able to capture nuances such as face angles with variational autoencoders (Kingma and Welling, 2013) or subtleties in painting strokes with the InfoGAN (Chen et al., 2016). Moreover, classically deterministic deep learning architectures are now being generalized to probabilistic deep models, for full predictive distributions instead of point estimates, and significantly more expressive representations (Wilson et al., 2016b,a; Al-Shedivat et al., 2016; Gan et al., 2016; Fortunato et al., 2017).

Similarly, probabilistic word embeddings can capture a range of subtle meanings, and advance the state of the art in predictive tasks. Multimodal word distributions naturally represent our belief that words do not have single precise meanings: indeed, the shape of a word distribution can express much more semantic information than any point representation.

In the future, multimodal word distributions could open the doors to a new suite of applications in language modelling, where whole word distributions are used as inputs to new probabilistic LSTMs, or in decision functions where uncertainty matters. As part of this effort, we can explore different metrics between distributions, such as KL divergences, which would be a natural choice for order embeddings that model entailment properties. It would also be informative to explore inference over the number of components in mixture models for word distributions. Such an approach could potentially discover an unbounded number of distinct meanings for words, but also distribute the support of each word distribution to express highly nuanced meanings. Alternatively, we could imagine a dependent mixture model where the distributions over words are evolving with time and other covariates. One could also build new types of supervised language models, constructed to more fully leverage the rich information provided by word distributions.

Acknowledgements

We thank NSF IIS-1563887 for support.

References

Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. 2016. Learning scalable deep kernels with recurrent structure. *arXiv preprint arXiv:1610.08936* .

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*. pages 23–32. http://aclweb.org/anthology-new/E/E12/E12-1004.pdf.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation* 43(3):209–226. https://doi.org/10.1007/s10579-009-9081-4.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Int. Res.* 49(1):1–47.

Xi Chen, Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. pages 2172–2180.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1025–1035. http://aclweb.org/anthology/D/D14/D14-1110.pdf.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. pages 160–167.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Martín Abadi et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.* 20(1):116–131.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. 2017. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798* .

Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. 2016. Scalable bayesian learning of recurrent neural networks for language modeling. *arXiv preprint arXiv:1611.08034* .

Michael Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research* 13:307–361.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*. pages 1406–1414.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR* abs/1408.3456.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*. pages 873–882. http://www.aclweb.org/anthology/P12-1092.

Tony Jebara, Risi Kondor, and Andrew Howard. 2004. Probability product kernels. *Journal of Machine Learning Research* 5:819–844.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*. pages 133–142.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *CoRR* abs/1312.6114. http://arxiv.org/abs/1312.6114.

Y. LeCun, L. Bottou, G. Orr, and K. Muller. 1998. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 2177–2185.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*. pages 2418–2424. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9314.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*. Sofia, Bulgaria.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukás Burget, and Jan Cernocký. 2011a. Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU 2011, Waikoloa, HI, USA, December 11-15, 2011*. pages 196–201. https://doi.org/10.1109/ASRU.2011.6163930.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*. pages 1045–1048.

Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011b. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*. pages 5528–5531. https://doi.org/10.1109/ICASSP.2011.5947611.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*. pages 3111–3119.

George A. Miller and Walter G. Charles. 1991. Contextual Correlates of Semantic Similarity. *Language & Cognitive Processes* 6(1):1–28. https://doi.org/10.1080/01690969108406936.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*. pages 1081–1088.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*.

Eric T. Nalisnick and Sachin Ravi. 2015. Infinite dimensional word embeddings. *CoRR* abs/1511.05392.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1059–1069. http://aclweb.org/anthology/D/D14/D14-1113.pdf.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1532–1543. http://aclweb.org/anthology/D/D14/D14-1162.pdf.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*. WWW '11, pages 337–346.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM* 8(10):627–633.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. pages 880–887. https://doi.org/10.1145/1390156.1390267.

C. Spearman. 1904. The proof and measurement of association between two things. *American Journal of Psychology* 15:88–103.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*. pages 151–160. http://aclweb.org/anthology/C/C14/C14-1016.pdf.

Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *CoRR* abs/1412.6623.

Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing. 2016a. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*. pages 2586–2594.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. 2016b. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. pages 370–378.

Dongqiang Yang and David M. W. Powers. 2006. Verb similarity on the taxonomy of wordnet. In *In the 3rd International WordNet Conference (GWC-06), Jeju Island, Korea*.

A Supplementary Material

A.1 Derivation of Expected Likelihood Kernel

We derive the form of expected likelihood kernel for Gaussian mixtures. Let f, g be Gaussian mixture distributions representing the words w_f, w_g. That is, $f(x) = \sum_{i=1}^{K} p_i \mathcal{N}(x; \mu_{f,i}, \Sigma_{f,i})$ and $g(x) = \sum_{i=1}^{K} q_i \mathcal{N}(x; \mu_{g,i}, \Sigma_{g,i})$, $\sum_{i=1}^{K} p_i = 1$, and $\sum_{i=1}^{K} q_i = 1$.

The expected likelihood kernel is given by

$$
\begin{aligned}
E_\theta(f,g) &= \int \left(\sum_{i=1}^{K} p_i \mathcal{N}(x; \mu_{f,i}, \Sigma_{f,i}) \right) \cdot \\
&\quad \left(\sum_{j=1}^{K} q_j \mathcal{N}(x; \mu_{g,j}, \Sigma_{g,j}) \right) \, dx \\
&= \sum_{i=1}^{K} \sum_{j=1}^{K} p_i q_j \int \mathcal{N}(x; \mu_{f,i}, \Sigma_{f,i}) \cdot \mathcal{N}(x; \mu_{g,j}, \Sigma_{g,j}) \, dx \\
&= \sum_{i=1}^{K} \sum_{j=1}^{K} p_i q_j \mathcal{N}(0; \mu_{f,i} - \mu_{g,j}, \Sigma_{f,i} + \Sigma_{g,j}) \\
&= \sum_{i=1}^{K} \sum_{j=1}^{K} p_i q_j e^{\xi_{i,j}}
\end{aligned}
$$

where we note that $\int \mathcal{N}(x; \mu_i, \Sigma_i) \mathcal{N}(x; \mu_j, \Sigma_j) \; dx = \mathcal{N}(0, \mu_i - \mu_j, \Sigma_i + \Sigma_j)$ (Vilnis and McCallum, 2014) and $\xi_{i,j}$ is the log partial energy, given by equation 3.

A.2 Implementation

In this section we discuss practical details for training the proposed model.

Reduction to Diagonal Covariance

We use a diagonal Σ, in which case inverting the covariance matrix is trivial and computations are particularly efficient.

Let $\boldsymbol{d}^f, \boldsymbol{d}^g$ denote the diagonal vectors of Σ_f, Σ_g The expression for $\xi_{i,j}$ reduces to

$$
\xi_{i,j} = -\frac{1}{2} \sum_{r=1}^{D} \log(d_r^p + d_r^q)
$$

$$
-\frac{1}{2} \sum \left[(\boldsymbol{\mu}_{p,i} - \boldsymbol{\mu}_{q,j}) \circ \frac{1}{\boldsymbol{d^p} + \boldsymbol{d^q}} \circ (\boldsymbol{\mu}_{p,i} - \boldsymbol{\mu}_{q,j}) \right]
$$

where $\circ$ denotes element-wise multiplication. The spherical case which we use in all our experiments is similar since we simply replace a vector $\boldsymbol{d}$ with a single value.

Optimization Constraint and Stability

We optimize $\log \boldsymbol{d}$ since each component of diagonal vector $\boldsymbol{d}$ is constrained to be positive. Similarly, we constrain the probability p_i to be in $[0,1]$ and sum to 1 by optimizing over unconstrained scores $s_i \in (-\infty, \infty)$ and using a softmax function to convert the scores to probability $p_i = \frac{e^{s_i}}{\sum_{j=1}^{K} e^{s_j}}$.

The loss computation can be numerically unstable if elements of the diagonal covariances are very small, due to the term $\log(d_r^f + d_r^g)$ and $\frac{1}{\boldsymbol{d^q} + \boldsymbol{d^p}}$. Therefore, we add a small constant $\epsilon = 10^{-4}$ so that $d_r^f + d_r^g$ and $\boldsymbol{d}^q + \boldsymbol{d}^p$ becomes $d_r^f + d_r^g + \epsilon$ and $\boldsymbol{d^q} + \boldsymbol{d^p} + \epsilon$.

In addition, we observe that $\xi_{i,j}$ can be very small which would result in $e^{\xi_{i,j}} \approx 0$ up to machine precision. In order to stabilize the computation in eq. 2, we compute its equivalent form

$$
\log E(f,g) = \xi_{i',j'} + \log \sum_{j=1}^{K} \sum_{i=1}^{K} p_i q_j e^{\xi_{i,j} - \xi_{i',j'}}
$$

where $\xi_{i',j'} = \max_{i,j} \xi_{i,j}$.

Model Hyperparameters and Training Details

In the loss function L_θ, we use a margin $m = 1$ and a batch size of 128. We initialize the word embeddings with a uniform distribution over $[-\sqrt{\frac{3}{D}}, \sqrt{\frac{3}{D}}]$ so that the expectation of variance is 1 and the mean is zero (LeCun et al., 1998). We initialize each dimension of the diagonal matrix (or a single value for spherical case) with a constant value $v = 0.05$. We also initialize the mixture scores s_i to be 0 so that the initial probabilities are equal among all K components. We use the threshold $t = 10^{-5}$ for negative sampling, which is the recommended value for `word2vec` skip-gram on large datasets.

We also use a separate output embeddings in addition to input embeddings, similar to `word2vec` implementation (Mikolov et al., 2013a,b). That is, each word has two sets of distributions q_I and q_O, each of which is a Gaussian mixture. For a given pair of word and context (w, c), we use the input distribution q_I for w (input word) and the output distribution q_O for context c (output word). We optimize the parameters of both q_I and q_O and use the trained input distributions q_I as our final word representations.

We use mini-batch asynchronous gradient descent with Adagrad (Duchi et al., 2011) which performs adaptive learning rate for each parameter. We also experiment with Adam (Kingma and Ba, 2014) which corrects the bias in adaptive gradient update of Adagrad and is proven very popular for most recent neural network models. However, we found that it is much slower than Adagrad ($\approx$ 10 times). This is because the gradient computation of the model is relatively fast, so a complex gradient update algorithm such as Adam becomes the bottleneck in the optimization. Therefore, we choose to use Adagrad which allows us to better scale to large datasets. We use a linearly decreasing learning rate from 0.05 to 0.00001.