Embedding Word Tokens using a Linear Dynamical System

David Belanger  
UMass Amherst  

Sham Kakade  
Microsoft Research New England

Abstract

Low dimensional representations of words allow accurate models to be trained on limited annotated data. While most word representations are context-independent, a natural way to induce representations for words in their particular context is to perform inference over latent variables in a probabilistic model. Given the recent success of continuous vector-space word representations, we provide such an inference procedure for continuous states, where words’ representations are given by the posterior mean from Kalman filtering. Our learning algorithm is extremely scalable, operating on simple coocurrence counts, and our initial results encourage further use of continuous-state dynamical probabilistic models in NLP.

1 Introduction

A popular semi-supervised learning technique in NLP is to employ word embeddings, where every word is mapped to a low dimensional dense vector. These vectors can be learned on massive corpora and allow accurate predictive models to be trained on limited annotated data. Typically, there is a vector for every word type, e.g., ‘dog’ receives the same vector independent of its context. On the other hand, the meanings of words depends on their usage in context, so ideally we would represent each word token with a unique vector.

In response, we provide a principled probabilistic model for inferring token-level embeddings using a Gaussian linear dynamical system (LDS). This is a continuous analog of popular discrete-state unsupervised generative models used in NLP for inducing class membership for tokens. The model is trained on a very large unannotated corpus and then exact posterior inference over latent Gaussian states for each token can be performed using Kalman filtering and smoothing. With this, a natural choice of token embedding is the posterior mean. Unlike left-to-right neural network architectures for sequences, our model admits a natural backwards pass to be able to also condition on the future.

Our two-stage estimation procedure, using the method-of-moments followed by EM with the ASOS approximation of Martens, relies on simple coocurrence statistics from the corpus, which can be gathered in parallel. After that, the cost of learning is independent of the amount of training data and requires just a few EM iterations, which scale linearly with the vocabulary size. We conclude with experiments demonstrating the usefulness of our token-level vectors for part of speech tagging, obtaining competitive performance using a simple local classifier applied to tokens’ inferred states and obtaining accuracy gains when adding them as features to a lexicalized tagger. Furthermore, we show that the Kalman gain matrix from the LDS directly yields useful type-level embeddings.

2 Linear Dynamical Systems

Consider sequences of observations \( w_1, \ldots, w_T \), where each \( w_t \in \mathbb{R}^V \), and \( w_{1:t} \) denotes \( w_1, \ldots, w_t \). An LDS obeys the following generative model:

\[
\begin{align*}
    x_t &= Ax_{t-1} + \eta \\
    w_t &= Cx_t + \epsilon,
\end{align*}
\]

(1)  
(2)

where \( h \) is the dimensionality of our hidden states \( x_t \), \( \epsilon \sim N(0, D) \), \( \eta \sim N(0, Q) \), and \( x_0 \sim \pi \). The latent space for \( x \) is completely unobserved and we could choose any orientation/scale for it. Therefore, without loss of generality,
we can either fix $A = I$ or $Q = I$ (but not both), and we’ll fix $Q$. Note that the maximum eigenvalue of $A$ must be no larger than 1 in absolute value if the system is stable. In this case, $x_t$ is asymptotically mean-zero (independent of $\pi$), so $w_t$ is also mean zero. Therefore, we assume that the data we fit to has been centered.

### 2.1 Posterior Inference

Since all noise in the generative model is Gaussian, and the dynamics follow a linear evolution, any set of $x$ vectors is distributed as a multivariate Gaussian under both the prior and the posterior. Kalman filtering provides exact posterior inference for $x_t$ given $w_{1:T}$, which is exactly captured by a mean and variance. Similarly, Kalman smoothing provides the posterior for $x_t$ given all observations $w_{1:T}$. Filtering a length-$T$ sequence is $O(T(h^3 + V^3))$, which can be followed by an $O(Th^3)$ smoothing step [8].

An important property of an LDS is that the evolution of each timestep’s posterior variance under the filter/smooother does not depend on the actual values of the observations. As a result, the posterior variance either diverges to infinity or it converges to a steady-state [10]. Therefore, when computing the posterior on long time series, it is reasonable to assume that we are always in this steady-state regime.

Following Ghahramani and Roweis [8], we will use $\hat{x}_t^+$ for the mean of the posterior for $x_t$ given $w_{1:t}$ and $\bar{x}_t$ to be the posterior mean given all the data $w_{1:T}$. Under the steady state assumption, both the posterior variances given $w_{1:t}$ and given $w_{1:T}$ are fixed, time-independent quantities. The posterior means are computed using the following steady-state filtering and smoothing equations, where $K \in \mathbb{R}^{h \times V}$ is known as the Kalman gain matrix.

$$
\dot{x}_t^+ = (A - KCA)x_{t-1}^+ + Kw_t \quad (3)
$$

$$
\dot{\bar{x}}_t = J\bar{x}_{t+1} + (I - JA)\hat{x}_t. \quad (4)
$$

Finally, define $\Sigma_1 = \mathbb{E}[\hat{x}_t\hat{x}_t^\top | w_{1:(t-1)}]$ to be the steady state covariance under the posterior for each $x_t$ given its history, which can be found by solving a generalized eigenvalue problem to discover the fixed point $\Sigma_1 = A\Sigma_1 A^\top + Q$. Similarly, we define $\Sigma_0 = \mathbb{E}[\bar{x}_t\bar{x}_t^\top | w_{1:t}]$. Also, define $S = C\Sigma_1 C^\top + D$ to be the marginal variance of observations under the posterior. The optimal choice of filtering parameters (under the squared loss) is given by the following $[10]$:

$$
K = \Sigma_1 C^\top S^{-1} \quad (5)
$$

$$
J = \Sigma_0 A^\top (\Sigma_1)^{-1}. \quad (6)
$$

### 2.2 Learning: Subspace Identification

Subspace Identification (SSID) is a family of method-of-moments estimators for the LDS parameters. Like other spectral methods used in machine learning for parameter identification, it is consistent and computationally efficient, but statistically suboptimal, since it is not maximizing the likelihood of the data [11]. See Viberg [17] for details on the various SSID algorithms.

Let $r$ be a small integer. Define the $(rV) \times h$ matrix $\Gamma_r = [C; CA; CA^2; \ldots; CA^{r-1}]$, where ‘;’ denotes vertical concatenation. Next, define the covariance at lag $i$ to be $\Psi_i = \mathbb{E}_i[w_{t+i}^\top w_t]$, which is valid because we assume the data to be mean zero. Finally, we consider the $x$-$w$ covariance at lag 1: $G = \mathbb{E}_{t,x}[x_t y_t^\top]$, and the $h \times (rV)$ matrix $\Delta_r = [A^{r-1}G A^{r-2}G \ldots AG G]$.

First, observe that $\Psi_i = CA^{i-1}G$ for $i > 0$ and $\Psi_i = G^\top (A^\top)^{-i-1}C^\top$ for $i < 0$. Next, define the Hankel matrix

$$
H_r = \begin{pmatrix}
\Psi_r & \Psi_{r-1} & \Psi_{r-2} & \ldots & \Psi_1 \\
\Psi_{r+1} & \Psi_r & \Psi_{r-1} & \ldots & \Psi_2 \\
& & & \ddots & \\
\Psi_{2r-1} & \Psi_{2r-2} & \Psi_{2r-3} & \ldots & \Psi_r
\end{pmatrix}. \quad (7)
$$

Then, assuming the data was drawn from an LDS, we have that $H_r = \Gamma_r \Delta_r$. SSID employs the following steps to identify $A$ and $C$: (1) take a pass over the data to accumulate $\Psi_0, \ldots, \Psi_r$; (2) do a rank-$k$ SVD of $H_r$ to obtain $\Gamma_r$ and $\Delta_r$; (3) invoke the specific block structure of $\Gamma_r$ and $\Delta_r$ to solve for $A$ and $C$ using least-squares; and (4) fit the observation noise covariance $D$ as the variance in the observations not explained by the latent states [17]:

$$
D = \Psi_0 - C\Sigma_1 C^\top \quad (8)
$$
2.3 Learning: EM with ASOS

A local optimum of the marginal likelihood of the data can be found using expectation-maximization [8]. This alternates between performing Kalman smoothing on the data to obtain a posterior over latent states and then updating the LDS parameters by solving various least-squares problems. The parameters can be initialized effectively using SSID [15]. Such a learning procedure is conceptually simple and easy to implement, but the E step can be prohibitively expensive, since it requires performing smoothing on the entire training set.

In response, we employ the ASOS (approximate second order statistics) procedure of Martens [10]. The author proposes an elegant set of approximations to the E step, rendering its computational cost independent of the amount of training data. The procedure is based on the following facts: (1) the M step only requires aggregate second-order statistics such as \( \mathbb{E}[x_i^t w_i^T] \) and \( \mathbb{E}[x_i^t x_i^T] \), instead of values of the posterior at specific timesteps; (2) the size of each covariance-at-lag matrix \( \Psi_i \) is independent of the amount of training data; (3) covariances \( \Psi_i \) for large \( i \) can be approximated accurately by either assuming that they are zero or that they are drawn directly from an LDS with the current estimates of the model parameters; and (3) if the posterior follows a time-independent Markov relationship, e.g. in steady state, then so do the second order statistics of the posterior. To exemplify the final fact, observe that (3) and (4) directly yield:

\[
\mathbb{E}[x_i^t w_i^T] = (A - K_{ss} C A) \mathbb{E}[x_{i-1}^t w_i^T] + K_{ss} \mathbb{E}[w_i w_i^T] \tag{9}
\]

\[
\mathbb{E}[x_i^t x_i^T] = J \mathbb{E}[x_{i+1} x_i^T] + (I - JA) \mathbb{E}[x_i x_i^T]. \tag{10}
\]

Such recursions, and others defined similarly, provide a dynamic programming procedure to estimate the necessary second order statistics, given an initial estimate for the second order statistics at time horizon \( r + 1 \), the observed quantities \( \Psi_1, \ldots, \Psi_r \), and the current values for the LDS parameters.

3 LDS Model for Text

Let \( V \) be the size of our vocabulary, about 150K. We consider sequences of observations where each \( w_t \) is a \( V \)-dimensional indicator vector that is nonzero in index \( i \) if token \( t \) has index \( i \) in the vocabulary. While Gaussian noise is clearly wrong in this scenario because the generative model does not generate proper indicator vectors, we can still fit the model to data, evaluate the joint probability of observations, and perform posterior inference given observations. Therefore, Gaussian noise is worth exploring, since multinomial models with softmax link functions prevent closed-form M step updates and require expensive manipulation of normalizing constants.

Since each \( w_t \) is an indicator vector, the product \( K w_t \) in (3) is simply a column of \( K \). Therefore, we can view (3) as a simple linear recurrent neural network, where the first layer of the network maps words to type-level embeddings. A primary difference between our LDS model and a linear recurrent neural network is that the LDS has a natural backward pass (4) for integrating information from the future of a token.

Various aspects of the learning algorithms described above, such as (5), scale with \( V^3 \), which is intractable. It’s also infeasible to even instantiate a \( V \times V \) dense matrix. We now describe a series of techniques used to render parameter learning manageable in terms of both time and space. These do not apply to LDS in general; they exploit consequences of our indicator-vector observations and the high levels of sparsity typical in text cooccurrence statistics.

For indicator-vector observations, observe that \( \mathbb{E}_t[w_t w_t^T] = \text{diag}(\mu_1, \ldots, \mu_V) \), where \( \mu_i \) is the corpus-wide frequency of word type \( i \). Furthermore observe that the \((i,j)\)-th entry of \( \mathbb{E}_t[w_t w_{t+\tau}^T] \) is the frequency that word \( i \) occurs with word \( j \) \( \tau \) positions to the right. These matrices will be extremely sparse in practice. In order to obtain the second order statistics \( \Psi_i \), we must center our data, so we employ \( \Psi_i = \mathbb{E}_t[w_t w_{t+1}^T] - \mu \mu^T \). For every \( i \), this is a sparse-minus-low-rank matrix, so matrix multiplication involving \( \Psi_i \) scales with the number of nonzeros in \( \mathbb{E}_t[w_t w_{t+1}^T] \), substantially less than \( V^2 \). Furthermore, \( \mathbb{E}_t[w_t w_t + i^T] \) can be computed in parallel on chunks of the corpus, followed by a final merge.

The computational bottleneck of SSID is performing an SVD of (7). Both exact SVD and the popular randomized SVD algorithm of Halko et al. [9] can be performed using only a routine for multiplying by \( H_r \) and \( H_r^T \). Given the structure of the \( \Psi_i \), \( H_r \) is also a sparse-minus-low-rank matrix, so multiplication can be done efficiently, without instantiating a \( V \times V \) matrix. Similarly, the E-step using ASOS can be performed efficiently since it only interacts with the data via multiplication with various \( \Psi_i \).
Finally, observe that the observation noise covariance $D$ is $V \times V$. Naively, this is unmanageable computationally. However, the estimator (8) used by SSID and the one used in the M step of EM express $D$ as $\Psi_0$ minus a low rank matrix [8]. Therefore, due to the specific diagonal structure of $\Psi_0$, $D$ is fundamentally a lower dimensional object. It is manageable if we do not instantiate it as a $V \times V$ matrix and instead maintain this factored form. Finally, many tasks, such as computing the Kalman gain matrix (5) and computing the likelihood of observations require the precision matrix $D^{-1}$. This can be manipulated implicitly using the Sherman-Woodbury-Morrison formula.

4 Experiments

We train our LDS on a concatenation of three newswire corpora, the APNews portion of Gigaword, RCV1, and the New York Times from 1987-2007, totalling 2.2B tokens. The text is passed through a basic tokenizer, mapped to lower case, all numbers are replaced with a $<$num$>$ token, and all types not in the most 150K most frequent are mapped to $<$oov$. We treat the corpus as one very long time series when computing $\Psi_r$, ignoring document boundaries. We use $r = 2$ for SSID and $r = 10$ for ASOS, 200 latent dimensions, and 10 EM iterations. After 10 iterations, we found the model’s predictive accuracy diminished, and we will explore methods for preventing overfitting in future work.

Unsupervised learning of generative discrete-state sequence models for text has been shown to capture part-of-speech (POS) information accurately [6]. In response, we assess the ability of the LDS to also capture POS structure. In Table 4, we consider Penn Treebank test set accuracy using the both 12 ‘universal’ POS tags [13] and the original tags. We employ the standard train-dev-test split. In the first two columns, we apply a local classifier with token embeddings obtained by applying a Kalman smoother with a model trained using SSID (LDS-SSID), and a model trained with EM and SSID (LDS-SSID+EM). We found that tagging accuracy was poor using a simple linear classifier on the embeddings, and instead employ a two-layer neural network with sigmoid activations. We also project token embeddings to be unit norm. We contrast the performance with a traditional tagger with lexicalized features and non-local classification (Lex). In the final two columns, we add continuous features to the Lex tagger. Lex+Word2Vec uses CBOW type embeddings trained on the same corpus [11]. Lex+LDS uses token embeddings from SSID+EM. Both the decent accuracy of LDS-SSID+EM, which uses local prediction and no lexical features, and the improvements of Lex+LDS over Lex suggest that our dynamical model is capturing discriminative syntactic structure.

<table>
<thead>
<tr>
<th></th>
<th>LDS-SSID</th>
<th>LDS-SSID+EM</th>
<th>Lex</th>
<th>Lex+Word2Vec</th>
<th>Lex+LDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>universal</td>
<td>95.14</td>
<td>96.23</td>
<td>97.9</td>
<td>97.92</td>
<td>98.00</td>
</tr>
<tr>
<td>original</td>
<td>91.10</td>
<td>92.90</td>
<td>97.27</td>
<td>97.25</td>
<td>97.34</td>
</tr>
</tbody>
</table>

In Table 4 we define type-level embeddings as columns of the Kalman gain matrix, whitened with respect to the steady state covariance of the posterior. We compare the nearest neighbors to Word2Vec CBOW embeddings trained on the same corpus [11]. We find that ours captures similar semantic structure.

<table>
<thead>
<tr>
<th>query</th>
<th>LDS Kalman Gain Matrix</th>
<th>Word2Vec (CBOW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>knows safe referring</td>
<td>understands hates loves adores loathes harmless sterile healthy hopeless neutral alluding referring motioning pausing gesturing primaries nominee nomination caucus party materials contamination wastes weaponry vapors</td>
<td>understands thinks gerson reviewer loves safer unsafe render healthy hospitable awakening bunning according whipped simeon primaries principal nomination main overridings materials weaponry documentation everyman wastes</td>
</tr>
</tbody>
</table>

5 Conclusion and Future Work

Since it performs exact Bayesian inference, the Kalman filter provides an optimal low-rank linear predictor of the future, given the past, under the squared loss. Our algorithm is also fast, both for learning and inference. Our next steps are to develop a similarly scalable learning algorithm for learning a model with multinomial observations, to evaluate the model in terms of next word prediction, and to explore further applications of our token embeddings in downstream tasks.
References


