A Linear Dynamical System Model For Text

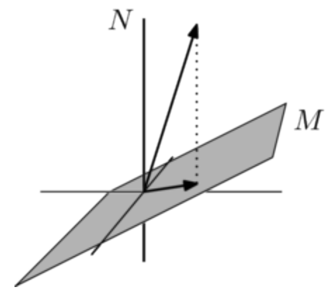
David Belanger (UMass Amherst)
Sham Kakade (Microsoft Research)

UMASS CS
SCHOOL OF COMPUTER SCIENCE

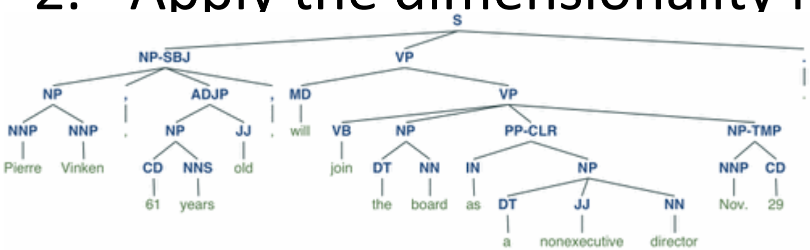
Microsoft Research

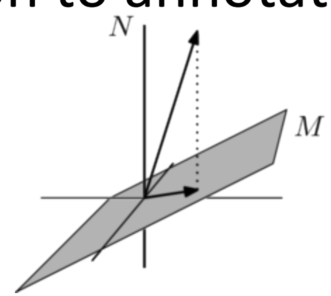
Semi-Supervised Learning in NLP

1. Do unsupervised learning that induces some reduced-dimensionality representation of text.

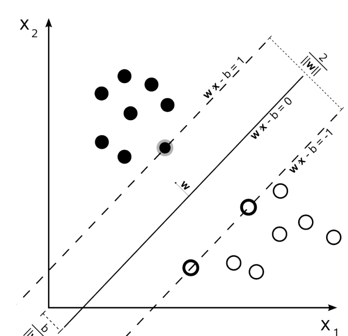


2. Apply the dimensionality reduction to annotated data.



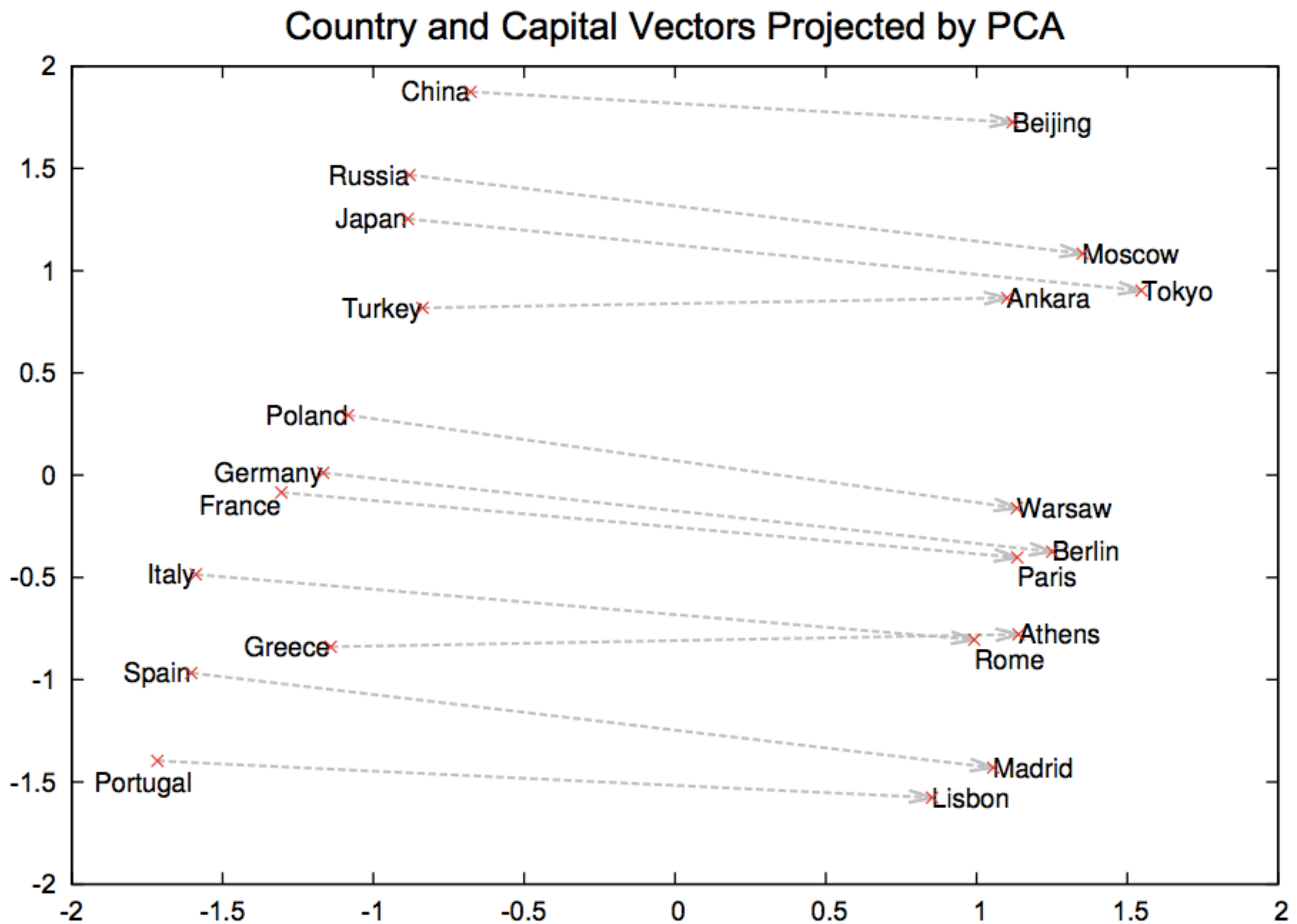


3. Do supervised learning on the mapped data.



Word Embeddings

Map each word to a low dimensional vector



(Bengio et al. 2003; Mikolov et al., 2013; …)

Word Tokens vs. Word Types

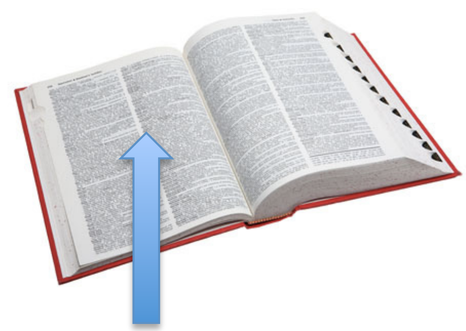
Types:

What you look up in the dictionary.

Tokens:

Words in context.

“The dog ran.”



Token

Type

Word embeddings are typically at the type level

Our Work

Goal:
Token Embeddings

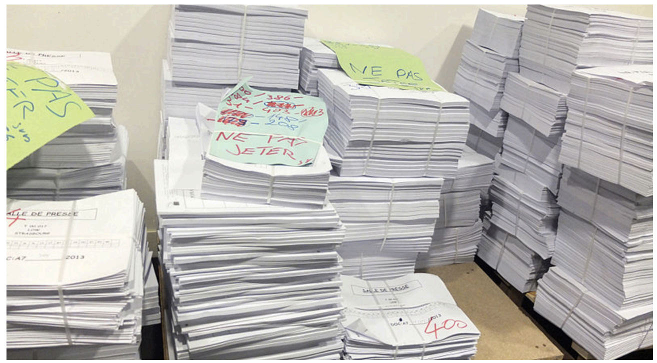
We should embed word *tokens* in context.

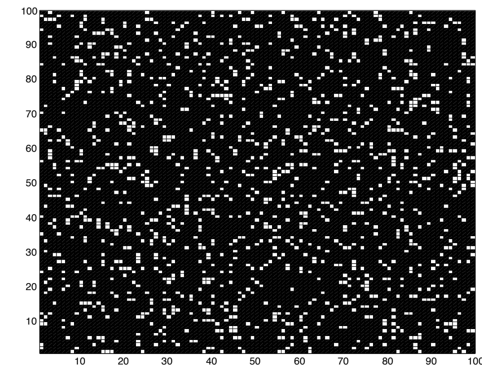
"Bank of England" vs. "River Bank Cafe"

"chair of the department" vs. "chair at the dinner table"

Consideration:
Advantages of Type-Level Training

Computational:





Training data compressed to sparse co-occurrence counts.

Size of matrix is independent of size of corpus!

Statistical:

$$P(w) = \frac{\#(w)}{N} \quad \Rightarrow \quad P(w) = \frac{\#(w+\alpha)}{N+\alpha V}$$

Smoothing is difficult in token-level training.

Consideration: Latent-Variable Sequence Modeling

- Many word embedding methods consider sliding windows or bags of words.
- Text is structured as a sequence. Ideally our token embedding method would model this structure.
- The latent state yields dimensionality reduction

Our General Method

1) Learn a generative model for text sequences with a vector-valued latent variable for every token.
2) At test time, obtain token embeddings using posterior inference over these latent variables.

Related Work

- Latent-state sequence models trained at token level:
  - HMMs w/ Baum-Welch (Rabiner, 1986)
  - RNN language model (Mikolov et al., 2010)
  - Neural language model (Bengio et al., 2003)

- Sequence model with type-level training, but no dimensionality reduction:
  - Ngram language models

- Type-level training of word embeddings, but not a sequence model:
  - Glove (Pennington, et al., 2014)
  - PPMI factorization (Levy and Goldberg, 2014)
  - CCA (Dhillon et al., 2012, Stratos et al. 2015)

- Token-level training, but not a sequence model:
  - Word2Vec (Mikolov et al., 2013) and variants

- Type-level training of sequence model, but requires third-order statistics:
  - Spectral learning of HMMs (Hsu et al., 2008)

Linear Dynamical Systems

Gaussian Linear Dynamical System

Generative model:

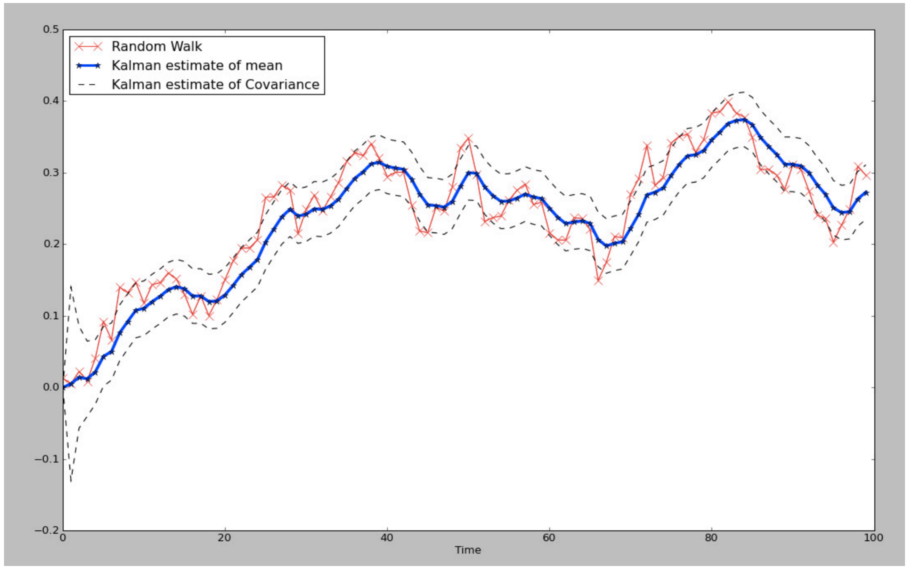
latent states $x_t = \boxed{A x_{t-1}} + \eta$

observations $w_t = \boxed{C x_t} + \epsilon,$

$$\boxed{\epsilon \sim N(0, D), \eta \sim N(0, Q)}$$

Kalman Filter

- *Exact, Efficient* posterior inference for latent states.



(r-bloggers.com)

- Maintains mean and variance for every timestep.
- Cubic in relevant dimensions.
- Forward and backward passes.

Steady State Kalman Filter

Fact 1:

The Kalman filter's update to the posterior variances doesn't depend on the actual observations.

Fact 2:

This variance reaches a steady state value quickly.

Exact Kalman Filter

$$
\cancel{
\begin{aligned}
\hat{x}_t^{t-1} &= A\hat{x}_{t-1}^{t-1} \\
S_t^{t-1} &= AS_{t-1}^{t-1}A^\top + Q \\
K_t &= S_t^{t-1}C^\top(CS_{t-1}^{t-1}C^\top + D)^{-1} \\
\hat{x}_t^t &= \hat{x}_{t-1}^{t-1} + K_t(w_t - C\hat{x}_t^{t-1}) \\
S_t^{t} &= S_t^{t-1} - K_tCS_t^{t-1}
\end{aligned}
}
$$

Kalman Filter
w/ Steady State Assumption

$$\hat{x}_t^t = (A - K_{ss}CA)\hat{x}_{t-1}^{t-1} + K_{ss}w_t$$

Steady-State Filtering

Precompute

Posterior mean at t-1, given observations including t-1.

$$\hat{x}_t^t = (A - K_{ss}CA)\hat{x}_{t-1}^{t-1} + K_{ss}w_t$$

Posterior mean at t, given observations including t.

Kalman Gain Matrix

Steady-State Backwards Pass (Kalman Smoothing)

$$\bar{x}_t = J_{ss}\bar{x}_{t+1} + (I - J_{ss}A)\hat{x}_t$$

Doesn't depend on observation dimension. Fast.

LDS for Text

Gaussian Likelihood for Words?

One-hot encoding $[0, \ldots, 1, \ldots 0]$

"CAT"

Effect of using Gaussian Likelihood

CAN DO	CAN NOT DO
Perform Posterior Inference	Generate Text
Evaluate Probability of Observation	
Fit Model Very Quickly	

Relationship to RNN Language Model

$$\hat{x}_t^t = (A - K_{ss}CA)\hat{x}_{t-1}^{t-1} + K_{ss}w_t$$

Product with one-hot vector = word embedding lookup

Kalman filter updates

=

RNN language model updates with no non-linearities

Text-LDS vs. RNN Language Model

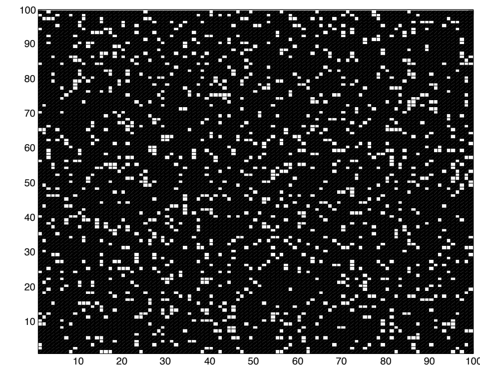
	Pros	Cons
LDS	• Fast learning (this paper) • Backwards Pass	• Can't generate text from it. • Perplexity uninterpretable
RNN-LM	• longer-term memory	• slow training • difficult to tune stepsizes, etc.

Spoiler Alert:
We speed up RNN training by initializing with LDS parameters.

Learning the LDS Parameters

Type-Level Sufficient Statistics

$$\Psi_i = \mathbb{E}_t[w_{t+i} w_t^\top]$$



$$[\Psi_i]_{jk} = \frac{\#\left(\text{word}_k \text{ i positions to the right of word}_j\right)}{N}$$

Collect in single (parallelizable) pass over corpus.

Spectral learning of HMMs uses *third* order moments

$$\mathbb{E}_t[w_{t+2} \otimes w_{t+1} \otimes w_t]$$ difficult to estimate!

Learning Algorithm 1: Subspace Identification (Method of Moments)

(Van Overschee & De Moor, 1996)

Step 1: Construct Big, Sparse Hankel Matrix

$$H_r = \begin{pmatrix} \Psi_r & \Psi_{r-1} & \Psi_{r-2} & \dots & \Psi_1 \\ \Psi_{r+1} & \Psi_r & \Psi_{r-1} & \dots & \Psi_2 \\ \dots & & & & \\ \Psi_{2r-1} & \Psi_{2r-2} & \Psi_{r-3} & \dots & \Psi_r \end{pmatrix}$$

Step 2: (Randomized) SVD (Halko and Tropp, 2009)

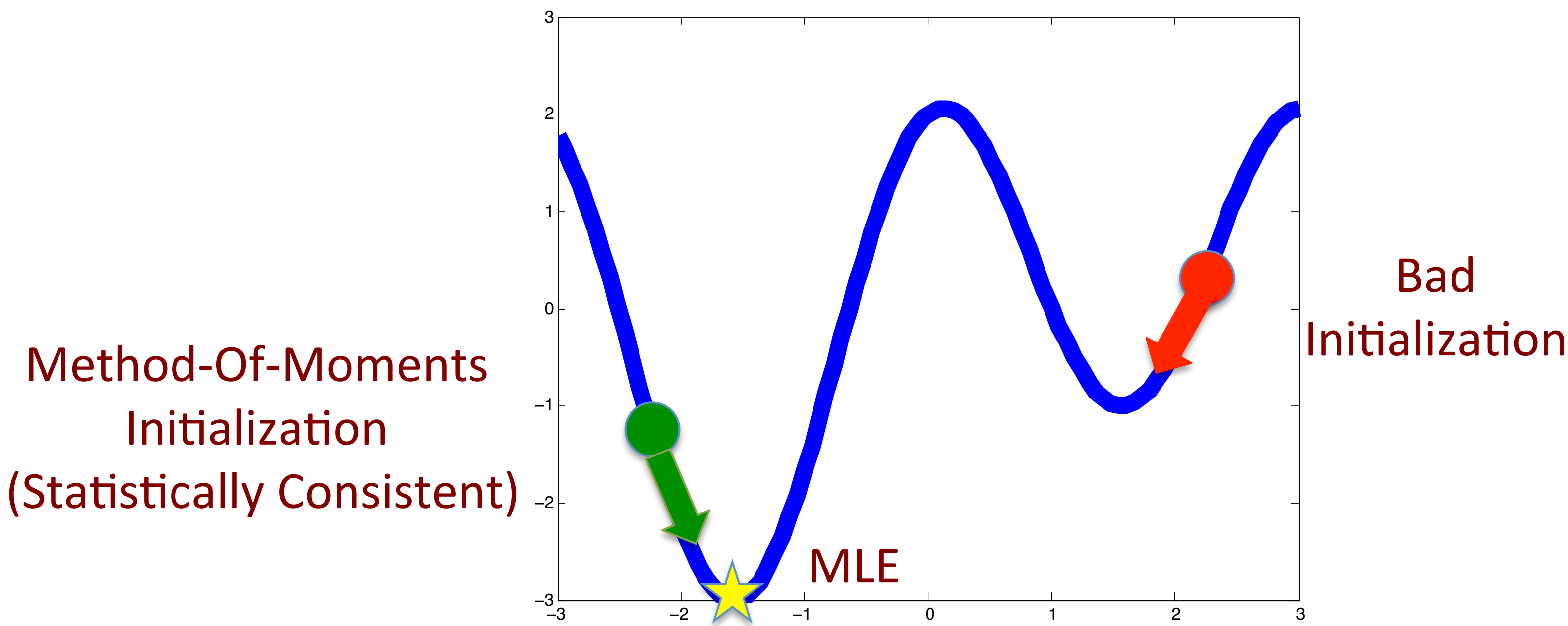
$$H_r = \Gamma_r \Delta_r$$

PROS	CONS
Fast, Non-Iterative	Statistically Suboptimal
Statistically Consistent	

Two-Stage Estimation

Meta-Algorithm:

1) Initialize parameters
2) Do local search on likelihood surface using EM (because MLE is statistically optimal)



Learning Algorithm 2: Expectation-Maximization (Initialized With Subspace ID)

~~E-Step = Posterior inference over the corpus~~

M-Step = Two easy least-squares problems

Slow. Not at type-level.

ASOS E-Step (Martens, 2010)

Observation 1:

The M-step is least-squares, so all we need from the E step are time-averaged second order statistics.

$$\mathbb{E}[\hat{x}_t w_t^\top],\ \mathbb{E}[\hat{x}_t \hat{x}_t^\top],\ \mathbb{E}[\hat{x}_{t+1} \hat{x}_t^\top]$$

Observation 2:

If the posterior follows a Markov relationship (Kalman Filter), then so do the time-averaged second order statistics.

Example Markov relationship $$x_t = Ax_{t-1} + b_t$$

Markov relationship on second-order statistics $$\mathbb{E}[x_t w_t^\top] = A\mathbb{E}[x_{t-1} w_t^\top] + \mathbb{E}[b_t w_t^\top]$$

Observation 3:

Using Ψ_i, we can Kalman filter + smooth second-order statistics matrices directly!

Recap

So far: how to handle very large corpora.

Next: how to handle large vocabularies by exploiting the specific structure of one-hot data.

High Dimensional Observations

$$\begin{aligned} x_t &= Ax_{t-1} + \eta \\ w_t &= Cx_t + \epsilon, \end{aligned}$$

$$\epsilon \sim N(0, D), \eta \sim N(0, Q)$$

Can't even store a V x V matrix!

Option 1: Use diagonal approximation.

Option 2: Exploit specific functional form of MLE for D

MLE for Noise Covariance

μ = vector of word frequencies

$$\Psi_0 = \mathbb{E}_t[w_t w_t^\top] = \mathrm{diag}(\mu) - \mu\mu^\top$$

MLE noise covariance is diagonal-minus-low-rank:

$$I - \mu^{\frac{1}{2}} \mu^{\frac{1}{2}^\top} + \left[CM^\top\right] B \left[E^\top M\right]^\top$$

But we need the *inverse* covariance all over the place…

Sherman-Woodbury-Morrison to the rescue!

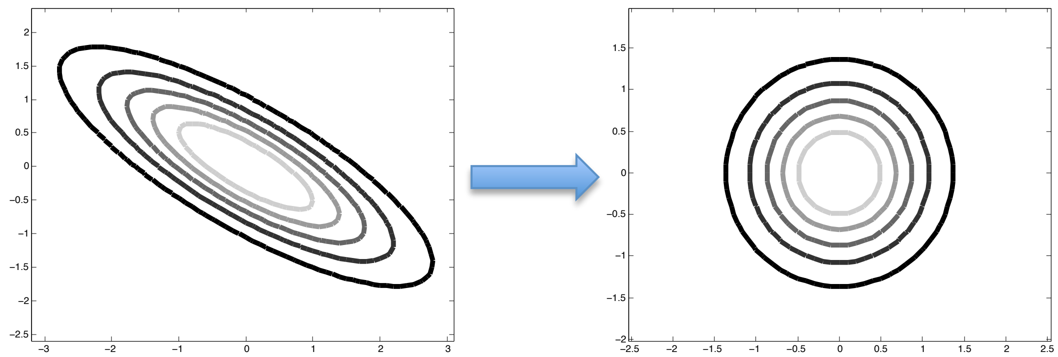
More Linear Algebra Tricks (see paper)

- Whiten the data for SSID using unigram frequencies.
- Account for rank deficiency of the one-hot observations.

Obtaining Token Embeddings using the LDS

Train Time:

1. Train the LDS
2. Find posterior latent covariance on training data
3. Transform LDS so that training latent covariance is spherical



Test Time:

1. Run Kalman smoothing per-sentence to get posterior over latent states.
2. Token Embedding = Posterior Mean

Experiments

LDS Transition Dynamics

The transition matrix A converts right singular vectors into left singular vectors. Are these interpretable?

Right Singular Vector	Left Singular Vector
chris mike steve jason tim jeff bobby ian greg adam tom phil nick brian ron	evans anderson harris robinson smith phillips collins murray murphy
brooklyn art science harlem princeton manhattan wimbledon hartford arts greenwich advertising massachusetts	symphony journal briefing street harbor beach birthday medal avenue bay innings box park district
salt chicken pepper chocolate butter cheese cream sauce bread sugar thick	chicken cream pepper sauce cheese chocolate salt butter bread sweet
policemen helicopters soldiers suspects demonstrators guards iraqis personnel	remained expressed recommended denied remains feels gets resumed is sparked

WSJ Part of Speech Tagging

Method:
Local classification using
dense features per token.

Word2Vec	LDS-SSID	LDS-EM
92.58	**83.00**	**94.30**

Remarks:

1. SSID performs poorly on its own.
2. The LDS sequence model outperforms Word2Vec

WSJ Part of Speech Tagging

Method:

Structured prediction using

dense + lexicalized features

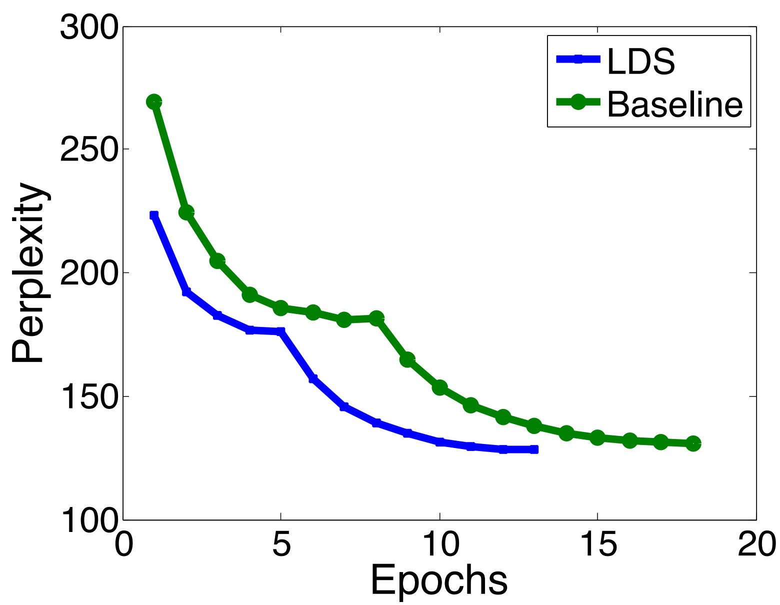
Lex	Lex + LDS-EM	Lex + Word2Vec
97.28	**97.32**	**97.35**

Remarks:

LDS sequence modeling unnecessary when performing global structured prediction.

RNN Initialization

- The Kalman Filter updates are identical to the those of an RNN with no non-linearities.
- Non-linear RNN training with SGD is slow.
- Initialize the RNN with LDS parameters!



Conclusion

- We obtain context-dependent word embeddings by performing posterior inference in an LDS.
- You can learn continuous latent state sequence models using only type-level statistics!
- Our LDS is a simple, scalable alternative to an RNN. Usefulness:
  - Current work: initialize RNN with LDS parameters.
  - Future: use within variational latent-variable RNN frameworks.
- Code coming soon. Check my website.

Questions?

Learning Algorithm: Overview

Step 1: Gather $\Psi_i = \mathbb{E}_t[w_{t+i} w_t^\top]$

Step 2: Estimate LDS parameters using Subspace Identification (Method of Moments)

Step 3: Perform about 50 iterations of EM to refine parameters.

Steps 2 and 3 only operate on Ψ_i

NER Tagging

Method:

Structured Prediction using

Dense + Lexicalized Features

Lex	Lex + Brown	Lex + Word2Vec	Lex + LDS-EM
89.3	**89.8**	**90.0**	**89.9**

Remarks:

Similar gain as established benchmarks.

Subspace ID (continued)

Step 2: SVD

$$H_r = \Gamma_r \Delta_r$$

Step 3: Use Nested Structure to Recover A and C using Least Squares

$$\Gamma_r = \left[C\ ;\ CA\ ;\ CA^2\ ;\ \ldots\ ;\ CA^{r-1}\right]$$

$$\Delta_r = \left[A^{r-1}G\ A^{r-2}G \ldots AG\ G\right]$$

LDS on Projected Words

- Step 1:

  Train type-level word embeddings using some existing algorithm.

- Step 2:

  Project the unsupervised training corpus.

- Step 3:

  Fit an LDS on the projected data.

LDS on Projected Words

- Advantages
  - Gaussian assumption is more reasonable.
  - Linear algebra tricks are unnecessary for scalability
- Problems
  - Still can't generate text from it.
  - Vulnerable to choice of embeddings.

LDS on Projected Words

New random variable:

$$Mw_t$$

Covariance of projection = projection of covariance:

$$\mathbb{E}_t[Mw_t(Mw_t)^\top] = M\mathbb{E}_t[w_t w_t^\top]M^\top$$

Motivation:
EM vs. SGD

- Tuning learning rate schedules for non-convex problems is annoying and difficult.
- EM takes big batch steps on the likelihood.

	Word2Vec	LDS-SSID	LDS-EM
Universal	95.00	89.26	96.44
Penn	92.58	83.00	94.30

	Lex	Lex + LDS-EM	Lex + Word2Vec
Universal	97.97	98.05	98.02
Penn	97.28	97.32	97.35

Neural Language Model

(Mnih and Hinton, 2007)

Represent context as linear combination of context words' embeddings

$$\hat{r} = \sum_{i=1}^{n-1} C_i r_{w_i}$$

Word probability is log-bilinear

$$P(w_n = w | w_{1:n-1}) = \frac{\exp(\hat{r}^T r_w + b_w)}{\sum_j \exp(\hat{r}^T r_j + b_j)}$$

ASOS

(Martens, 2010)

Step 0:

Collect empirical covariances at various lags

$$\Psi_i = \mathbb{E}_t[w_{t+i} w_t^\top]$$

Step 1:

Approximate covariances at high lags by assuming that they are drawn from the current model parameters.

Step 2:

Run a Kalman filter on the second order statistics directly.

Step 3:

Use the estimated covariances at lag = 0 to perform the M step.

Learning Algorithm: Overview

Gather Sufficient Statistics

$$\Psi_i = \mathbb{E}_t[w_{t+i}w_t^\top]$$

Subspace Identification

Motivation: Using Co-Occurrence Counts

- Learning is *independent of corpus size.*
- Can apply type-level smoothing.

Consideration: Sequence Model

Method Based on a Sequence Model?

Yes	No
Brown Clusters	Word2Vec
Recurrent Neural Networks	Glove
POS Induction with HMMs	CCA