Neural language models and word embeddings

Piotr Mirowski, Microsoft Bing
London Big-O Meetup
June 25, 2014

Ackowledgements

- AT&T Labs Research
  - Sumit Chopra (now at Facebook)
  - Srinivas Bangalore
  - Suhrid Balakrishnan
- New York University
  - Yann LeCun (now at Facebook)
- Microsoft
  - Geoffrey Zweig
  - Bhaskar Mitra
  - Abhishek Arun

Outline

- Motivations
  - Probabilistic **language models** (LMs) and **n-grams**
  - Distributional semantics
- Neural Probabilistic LMs
  - **Vector-space representation** of words
  - **Neural** probabilistic language model
  - **Log-Bilinear** (LBL) LMs
  - **Recurrent** Neural Network LMs
- Applications
  - Word representation
  - Speech recognition and machine translation
  - Sentence completion and **linguistic regularities**
- Bag-of-word-vector approaches
  - **Continuous bag-of-words** and **skip-gram** models
- Scalability with large vocabularies
  - Tree-structured LMs
  - Negative sampling

Outline

- Motivations
  - Probabilistic **language models** (LMs) and **n-grams**
  - Distributional semantics
- Neural Probabilistic LMs
  - **Vector-space representation** of words
  - **Neural** probabilistic language model
  - **Log-Bilinear** (LBL) LMs
  - **Recurrent** Neural Network LMs
- Applications
  - Word representation
  - Speech recognition and machine translation
  - Sentence completion and **linguistic regularities**
- Bag-of-word-vector approaches
  - **Continuous bag-of-words** and **skip-gram** models
- Scalability with large vocabularies
  - Tree-structured LMs
  - Negative sampling

Motivations (1): Language modeling

- Applications:
  - Speech recognition
  - Machine translation
- Language modeling aims at **quantifying** the **likelihood** of a **text** (sentence, query…)
- Score/rank candidates in **n-best list**
  - Example: HUB-4 TV Broadcast transcripts

100-best list of candidate sentences returned by the acoustic model:

Choose sentence with highest combined LM log-likelihood and acoustic model score

```
the american popular culture
americans popular culture
american popular culture
the nerds in popular culture
mayor kind popular culture
near can popular culture
the mere kind popular culture
...
```

Motivations (1): Language modeling

- Probability of a sequence of words:

$$P(W) = P(w_1, w_2, \ldots, w_{t-1}, w_T)$$

- **Conditional probability** of an upcoming word:

$$P(w_T \mid w_1, w_2, \ldots, w_{t-1})$$

- Chain rule of probability:

$$P(w_1, w_2, \ldots, w_{t-1}, w_T) = \prod_{t=1}^{T} P(w_t \mid w_1, w_2, \ldots, w_{t-1})$$

- (n-1)th order Markov assumption

$$P(w_1, w_2, \ldots, w_{t-1}, w_T) \approx \prod_{t=1}^{T} P(w_t \mid w_{t-n+1}, w_{t-n+2}, \ldots, w_{t-1})$$

- **n-grams** and **word context** of n-1 words

`the cat sat on the` **`mat`**

w_{t-5} w_{t-4} w_{t-3} w_{t-2} w_{t-1} w_t

$$P(w_t \mid \mathbf{w}_{t-5}^{t-1}) = 0.15$$

Motivations (1): Limitations of n-grams

the cat sat on the **mat** w_{t-5} w_{t-4} w_{t-3} w_{t-2} w_{t-1} w_t	$P(w_t \mid \mathbf{w}_{t-5}^{t-1}) = 0.15$
the cat sat on the **hat**	$P(w_t \mid \mathbf{w}_{t-5}^{t-1}) = 0.05$
the cat sat on the **sat**	$P(w_t \mid \mathbf{w}_{t-5}^{t-1}) = 0$

- Limitation: discrete model (each word is a token)
  - **Incomplete coverage** of the training dataset
    Vocabulary of size V words: V^n possible n-grams (exponential in n)

    my cat sat on the **mat** $P(w_t \mid \mathbf{w}_{t-5}^{t-1}) = ?$

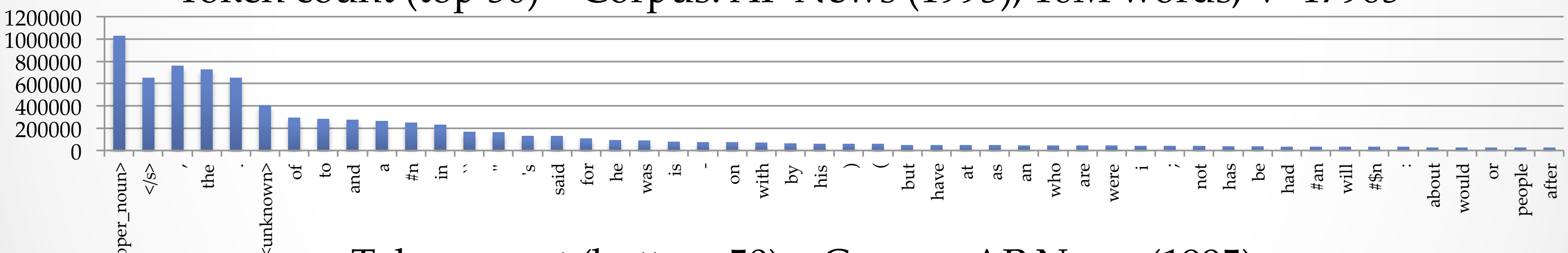
  - **No notion of semantic similarity** between word tokens

    the cat sat on the ***rug*** $P(w_t \mid \mathbf{w}_{t-5}^{t-1}) = ?$

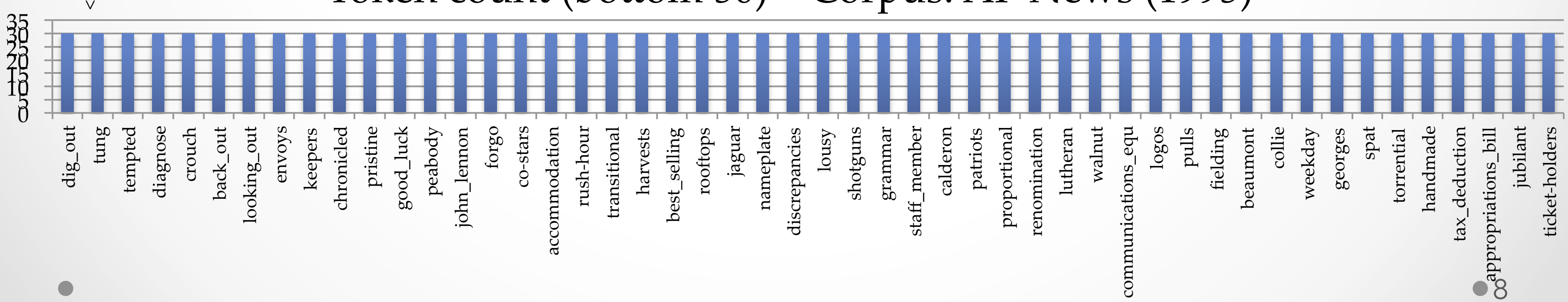
Motivations (2): Word representation

- **Bag-of-words**:
  words are tokens in vocabulary of size *V*
- Example: unigram distribution of words

Token count (top 50) – Corpus: AP News (1995), 16M words, V=17965



Token count (bottom 50) – Corpus: AP News (1995)



Motivations (2): Distributional semantics

- How to represent the **meaning of a word**?
- Using **vectors** of elements (called "features")
  - Option 1: using other words (bag-of-words representation)
  - Option 2: **learn the word representation features**
- Exploit **collocation** of words (word **context**)

the cat sat on the **mat**

w_{t-5} w_{t-4} w_{t-3} w_{t-2} w_{t-1} w_t

$$P(w_t \mid \mathbf{w}_{t-5}^{t-1}) = 0.15$$

```
[...] this article is about the cat species that is commonly kept [...]
[...] cats disambiguation . the domestic cat ( felis catus or felis [...]
[...] pet , or simply the cat when there is no need [...]
[...] to killing small prey . cat senses fit a crepuscular and [...]
[...] a social species , and cat communication includes a variety of [...]
[...] grunting ) as well as cat pheromones , and types of [...]
[...] , a hobby known as cat fancy . failure to control [...]
```

[http://en.Wikipedia.org/wiki/Cat]

Idea: combine two approaches

• • •

Learning word representations

and

learning language models

Outline

- Motivations
    - Probabilistic **language models** (LMs) and **n-grams**
    - Distributional semantics
- Neural Probabilistic LMs
    - **Vector-space representation** of words
    - **Neural** probabilistic language model
    - **Log-Bilinear** (LBL) LMs
    - **Recurrent** Neural Network LMs
- Applications
    - Word representation
    - Speech recognition and machine translation
    - Sentence completion and **linguistic regularities**
- Bag-of-word-vector approaches
    - **Continuous bag-of-words** and **skip-gram** models
- Scalability with large vocabularies
    - Tree-structured LMs
    - Negative sampling

Learning probabilistic language models

- Learn joint likelihood of training sentences under (n-1)th order Markov assumption using **n-grams**

$$P(w_1, w_2, \ldots, w_{t-1}, w_T) = \prod_{t=1}^{T} P(w_t \mid w_1, w_2, \ldots, w_{t-1}) \approx \prod_{t=1}^{T} P(w_t \mid \mathbf{w}_{t-n+1}^{t-1})$$

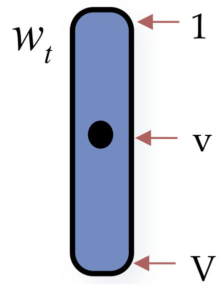
target word w_t

word history $\mathbf{w}_{t-n+1}^{t-1} = w_{t-n+1}, w_{t-n+2}, \ldots, w_{t-1}$

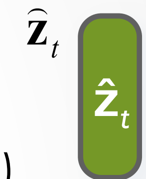
- Maximize the **log-likelihood**: $\sum_{t=1}^{T} \log P(w_t \mid \mathbf{w}_{t-n+1}^{t-1}, \boldsymbol{\theta})$
  - Assuming a **parametric model** θ

Vector-space representation of words

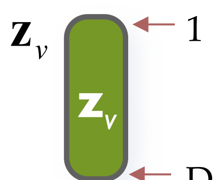
"**One-hot**" of "**one-of-V**" representation of a word token at position *t* in the text corpus, with **vocabulary of size *V***



Vector-space representation of the prediction of **target word w_t** (we predict a vector of size *D*)



Vector-space representation of any word *v* in the vocabulary using a vector of **dimension *D***



Vector-space representation of the **t^{th} word history/context**: e.g., concatenation of *n*-1 vectors of size *D*

Also called **distributed representation**

Learning continuous space language models

- Input:
  - word history (**one-hot** or **distributed representation**)
- Output:
  - target word (**one-hot** or **distributed representation**)
- **Function** that **approximates word likelihood**:
  - **Linear transform**
  - **Feed-forward neural network**
  - **Recurrent neural network**
  - **Continuous bag-of-words**
  - **Skip-gram**
  - …

Learning continuous space language models

- How do we **learn the word representations z** for each word in the vocabulary?
- How do we **learn the model** that predicts the next word or its representation $\hat{z}_t$ given a word history?

- Simultaneous learning of **model** and **representation**

Vector-space representation of words

- Compare two words using vector representations:
  - Dot product
  - Cosine similarity
  - Euclidean distance

- **Bi-Linear scoring function** at position *t*:

$$s\left(\mathbf{w}_1^{t-1}, v; \boldsymbol{\theta}\right) = s\left(\hat{\mathbf{z}}_t, v\right) = s_{\boldsymbol{\theta}}\left(v\right) = \hat{\mathbf{z}}_t^T \mathbf{z}_v + b_v$$

  - **Parametric model** θ predicts next word
  - Bias b_v for word *v* related to unigram probabilities of word *v*
  - Given a **predicted vector** $\hat{z}_t$,
    the actual predicted word is the **1-nearest neighbour of** $\hat{z}_t$

[Mnih & Hinton, 2007]

Word probabilities from vector-space representation

- Normalized probability using **softmax** function

$$P\left(w_t = v \mid \mathbf{w}_1^{t-1}\right) = \frac{e^{s(\hat{\mathbf{z}}_t, v)}}{\sum_{v'=1}^{V} e^{s(\hat{\mathbf{z}}_t, v')}}$$

- **Bi-Linear scoring function** at position *t*:

$$s\left(\mathbf{w}_1^{t-1}, v; \boldsymbol{\theta}\right) = s\left(\hat{\mathbf{z}}_t, v\right) = s_{\boldsymbol{\theta}}\left(v\right) = \hat{\mathbf{z}}_t^T \mathbf{z}_v + b_v$$

  - **Parametric model** $\boldsymbol{\theta}$ predicts next word
  - Bias b_v for word *v* related to unigram probabilities of word *v*
  - Given a **predicted vector $\hat{\mathbf{z}}_t$**,
    the actual predicted word is the **1-nearest neighbour of $\hat{\mathbf{z}}_t$**

[Mnih & Hinton, 2007]

Loss function

- **Log-likelihood model**:

$$\log P(w_1, w_2, \ldots, w_{t-1}, w_T) = \log\left(\prod_{t=1}^{T} P(w_t \mid \mathbf{w}_1^{t-1})\right) = \sum_{t=1}^{T} \log P(w_t \mid \mathbf{w}_1^{t-1})$$

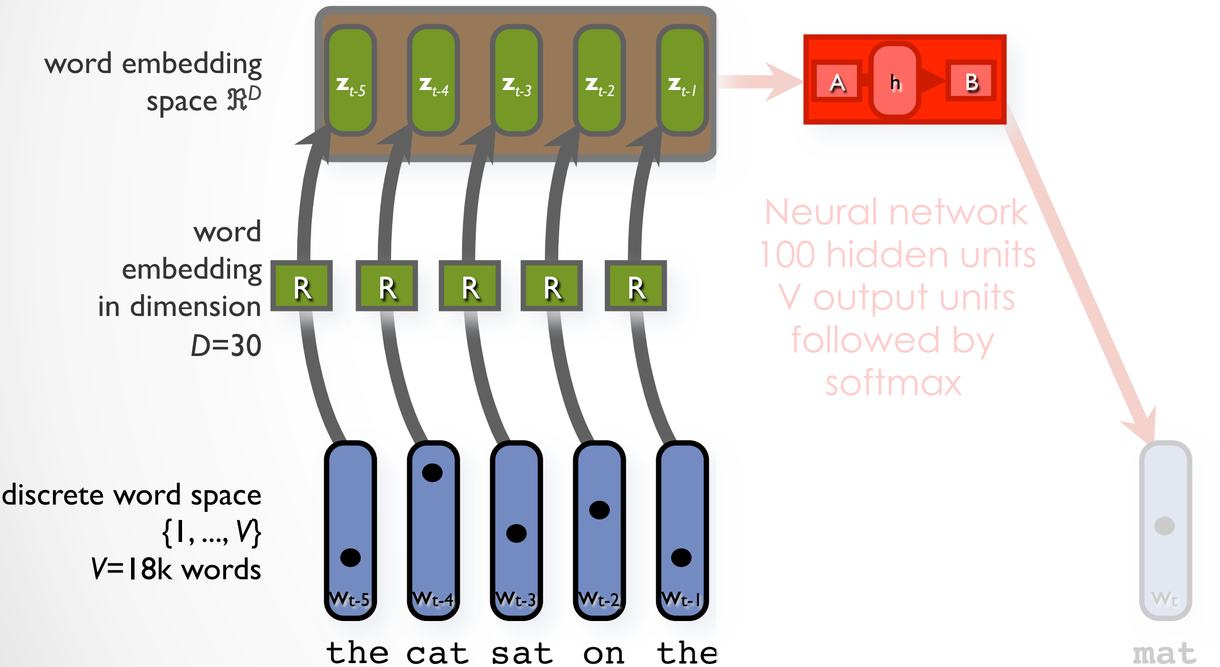
$$P\left(w_t = w \mid \mathbf{w}_1^{t-1}\right) = \frac{\boxed{e^{s_\theta(w)}}}{\boxed{\sum_{v=1}^{V} e^{s_\theta(v)}}}$$

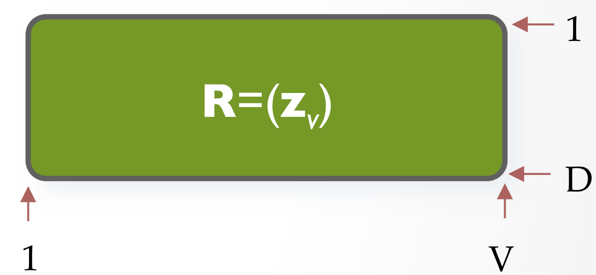
- **Loss function** to maximize:
  - Log-likelihood

$$L_t = \log P\left(w_t = w \mid \mathbf{w}_1^{t-1}\right) = \boxed{s_\theta(w)} \boxed{- \log \sum_{v=1}^{V} e^{s_\theta(v)}}$$

  - In general, loss defined as: **score of the right answer** + **normalization term**
  - Normalization term is expensive to compute

Neural Probabilistic Language Model






```
function z_hist = Embedding_FProp(model, w)
% Get the embeddings for all words in w
z_hist = model.R(:, w);
z_hist = reshape(z_hist, length(w)*model.dim_z, 1);
```

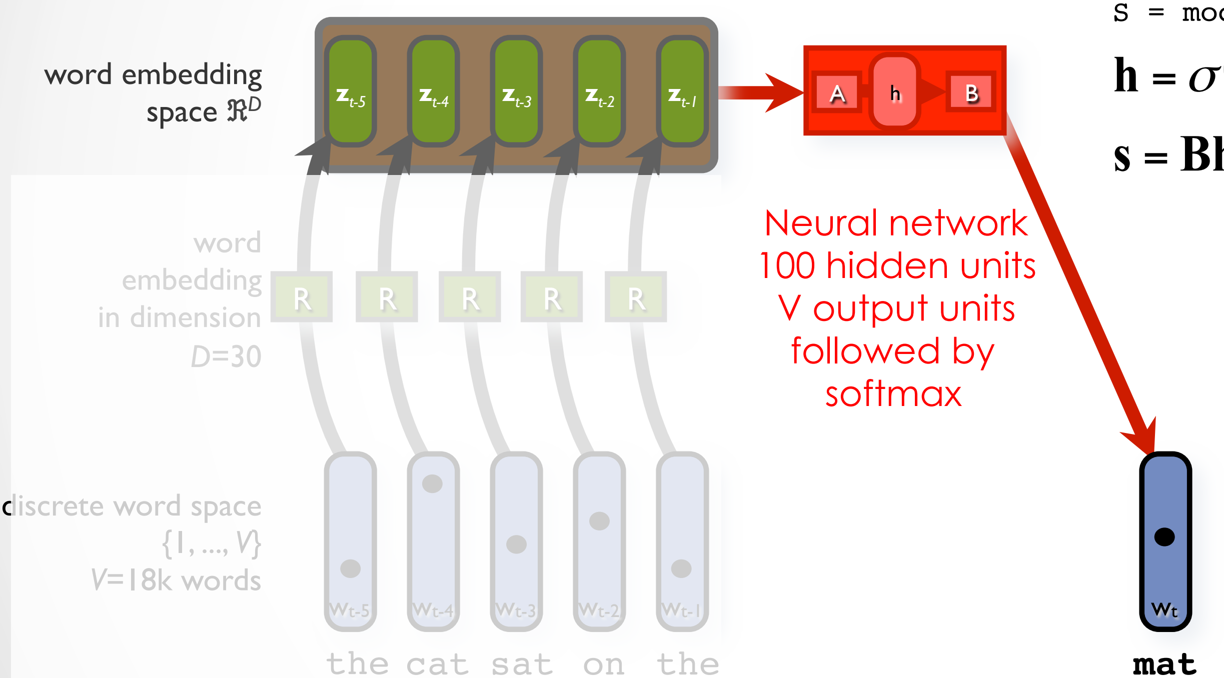
[Bengio et al, 2001, 2003; Schwenk et al, “Connectionist language modelling for large vocabulary continuous speech recognition”, *ICASSP* 2002]

Neural Probabilistic Language Model

```
function s = NeuralNet_FProp(model, z_hist)
% One hidden layer neural network
o = model.A * z_hist + model.bias_a;
h = tanh(o);
s = model.B * h + model.bias_b;
```

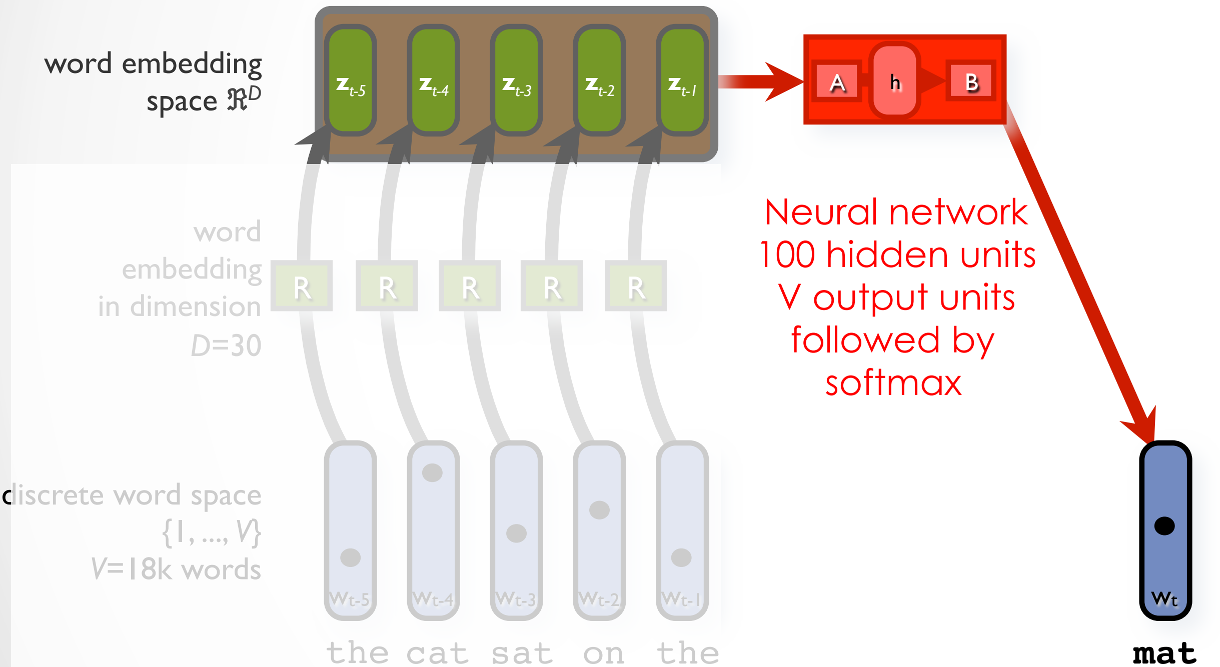
$$\mathbf{h} = \sigma\left(\mathbf{A}\mathbf{z}_{t-n+1}^{t-1} + \mathbf{b}_a\right)$$

$$\mathbf{s} = \mathbf{B}\mathbf{h} + \mathbf{b}_b$$



[Bengio et al, 2001, 2003; Schwenk et al, "Connectionist language modelling for large vocabulary continuous speech recognition", *ICASSP* 2002]

Neural Probabilistic Language Model

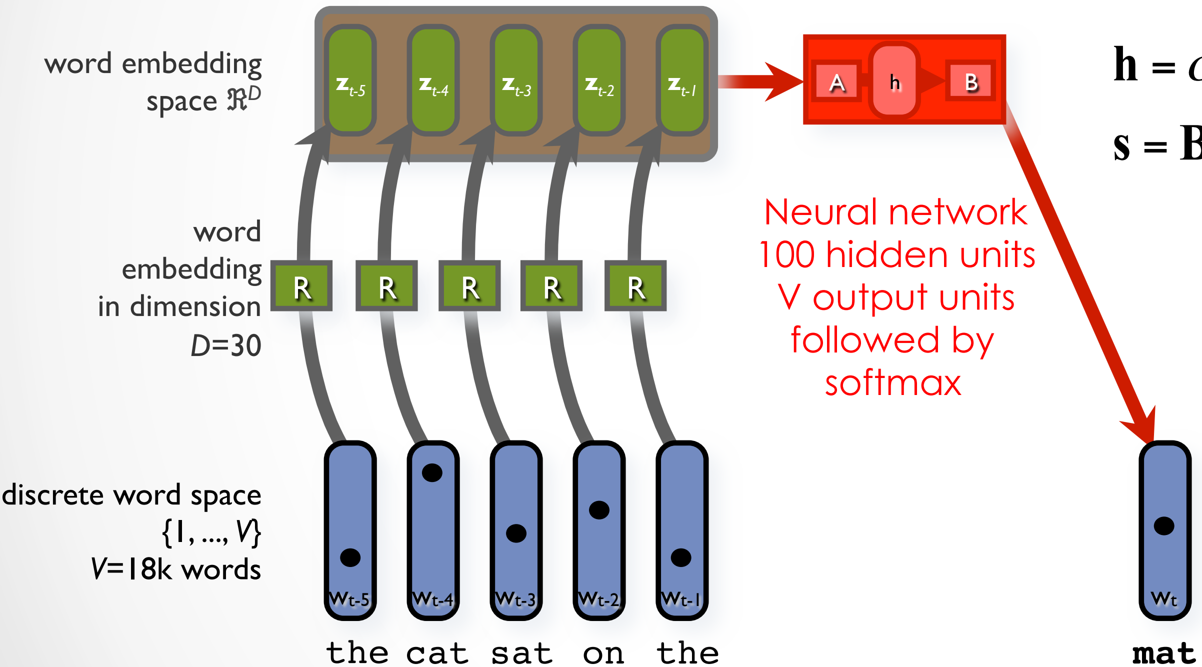



```
function p = Softmax_FProp(s)
% Probability estimation
p_num = exp(s);
p = p_num / sum(p_num);
```

$$P\left(w_t \mid \mathbf{w}_{t-n+1}^{t-1}\right) = \frac{e^{s_\theta(w_t)}}{\sum_v e^{s_\theta(v)}}$$

[Bengio et al, 2001, 2003; Schwenk et al, "Connectionist language modelling for large vocabulary continuous speech recognition", *ICASSP* 2002]

Neural Probabilistic Language Model



$$\mathbf{h} = \sigma\left(\mathbf{A}\mathbf{z}_{t-n+1}^{t-1} + \mathbf{b}_a\right)$$

$$\mathbf{s} = \mathbf{B}\mathbf{h} + \mathbf{b}_b$$

$$P\left(w_t \mid \mathbf{w}_{t-n+1}^{t-1}\right) = \frac{e^{s_{\theta}(w_t)}}{\sum_v e^{s_{\theta}(v)}}$$

Outperforms best n-grams (Class-based Kneyser-Ney back-off 5-grams) by 7%

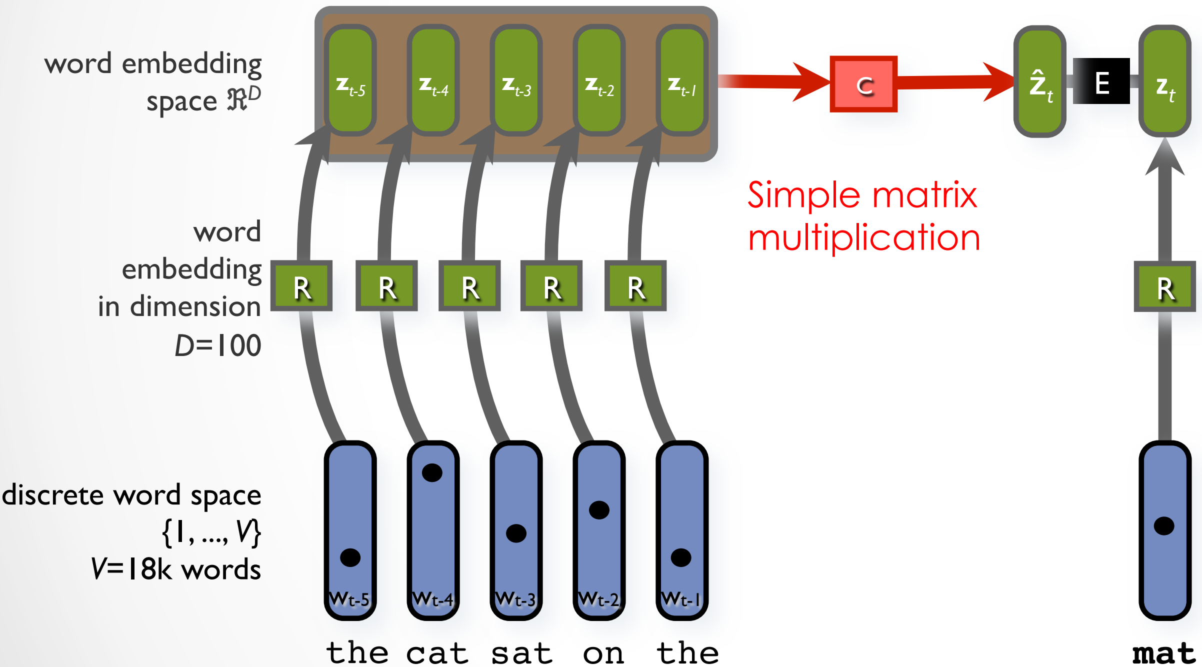
Took months to train (in 2001-2002) on AP News corpus (14M words)

Complexity: $(n\text{-}1)\times D + (n\text{-}1)\times D \times H + \boldsymbol{H\times V}$

[Bengio et al, 2001, 2003; Schwenk et al, "Connectionist language modelling for large vocabulary continuous speech recognition", *ICASSP* 2002]

Log-Bilinear Language Model

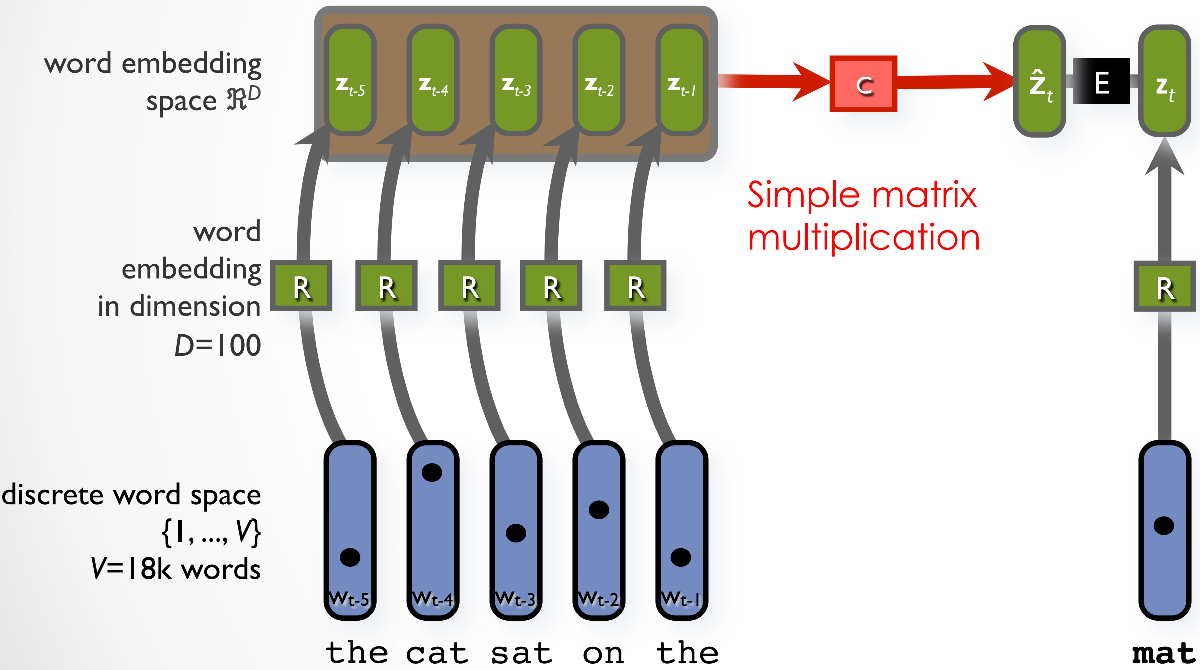
```
function z_hat = LBL_FProp(model, z_hist)
% Simple linear transform
Z_hat = model.C * z_hist + model.bias_c;
```



$$\hat{\mathbf{z}}_t = \mathbf{C}\mathbf{z}_{t-n+1}^{t-1} + \mathbf{b}_c$$

[Mnih & Hinton, 2007]

Log-Bilinear Language Model



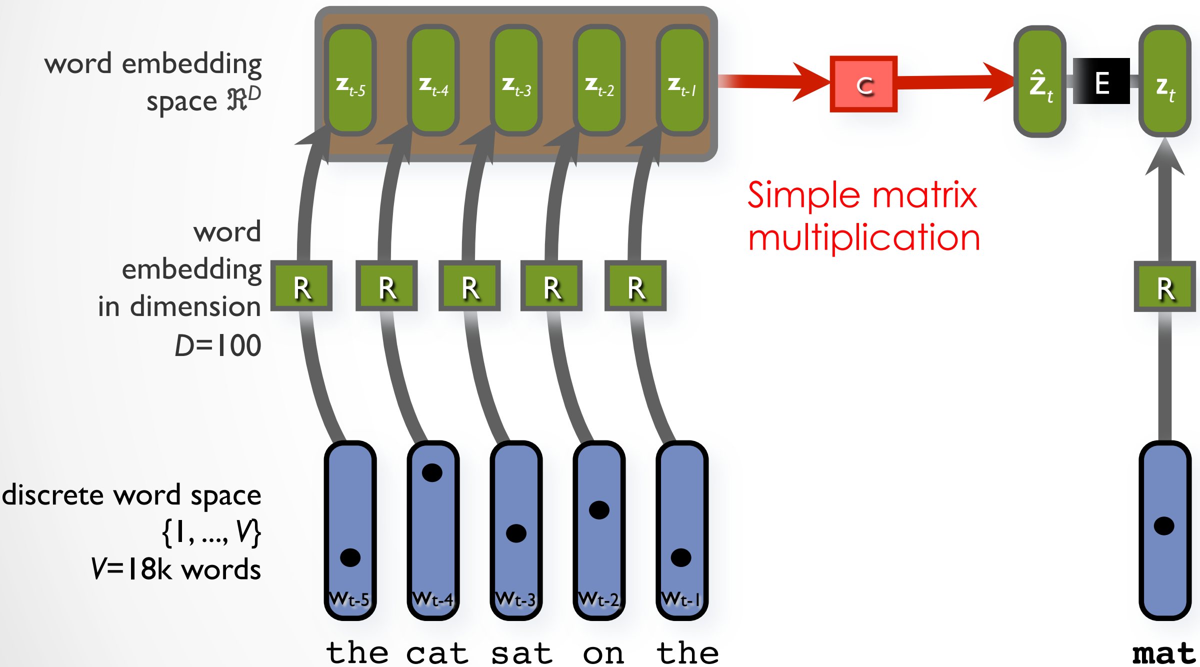

```
function s = ...
  Score_FProp(z_hat, model)
s = model.R' * z_hat + model.bias_v;
```

$$s_{\theta}(v) = \hat{\mathbf{z}}_t^T \mathbf{z}_v + b_v$$

$$P\left(w_t \mid \mathbf{w}_{t-n+1}^{t-1}\right) = \frac{e^{s_{\theta}(w_t)}}{\sum_v e^{s_{\theta}(v)}}$$

[Mnih & Hinton, 2007]

Log-Bilinear Language Model



$$\hat{\mathbf{z}}_t = \mathbf{C}\mathbf{z}_{t-n+1}^{t-1} + \mathbf{b}_c$$

$$s_{\theta}(v) = \hat{\mathbf{z}}_t^T \mathbf{z}_v + b_v$$

$$P\left(w_t \mid \mathbf{w}_{t-n+1}^{t-1}\right) = \frac{e^{s_{\theta}(w_t)}}{\sum_v e^{s_{\theta}(v)}}$$

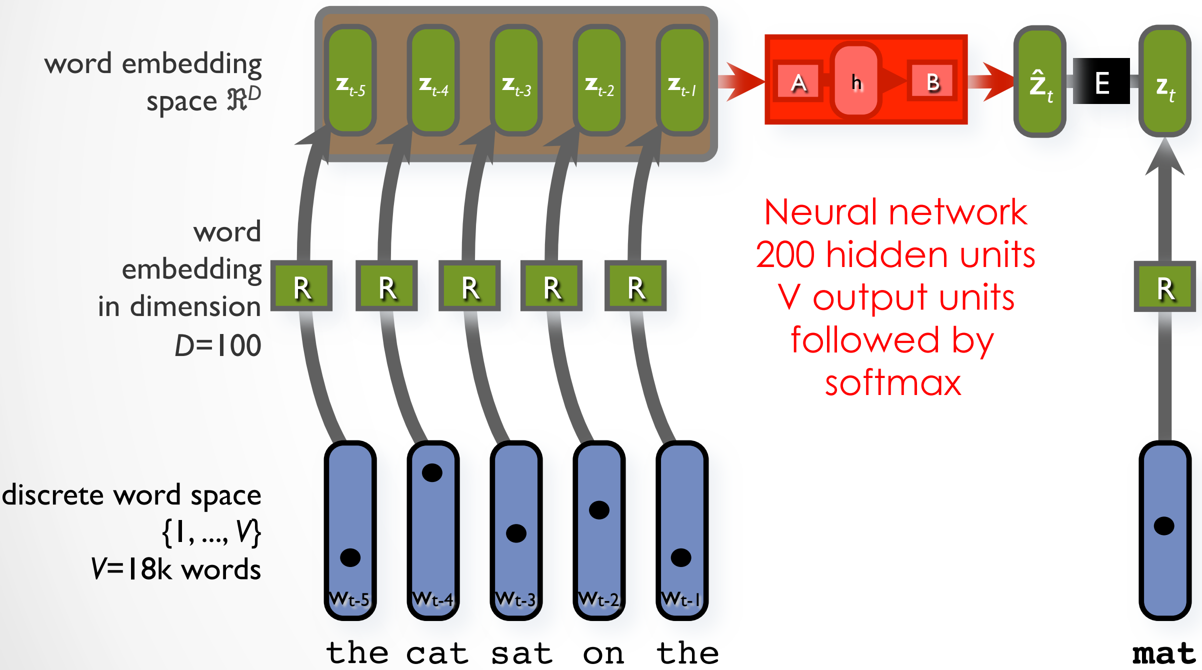
Slightly better than best n-grams (Class-based Kneyser-Ney back-off 5-grams)

Takes days to train (in 2007) on AP News corpus (14 million words)

Complexity: $(n\text{-}1)\times D + (n\text{-}1)\times D\times D +$ $\mathbf{D\times V}$

[Mnih & Hinton, 2007]

Nonlinear Log-Bilinear Language Model



$$\mathbf{h} = \sigma\left(\mathbf{A}\mathbf{z}_{t-n+1}^{t-1} + \mathbf{b}_a\right)$$

$$\hat{\mathbf{z}}_t = \mathbf{B}\mathbf{h} + \mathbf{b}_b$$

$$s_\theta(v) = \hat{\mathbf{z}}_t^T \mathbf{z}_v + b_v$$

$$P\left(w_t \mid \mathbf{w}_{t-n+1}^{t-1}\right) = \frac{e^{s_\theta(w_t)}}{\sum_v e^{s_\theta(v)}}$$

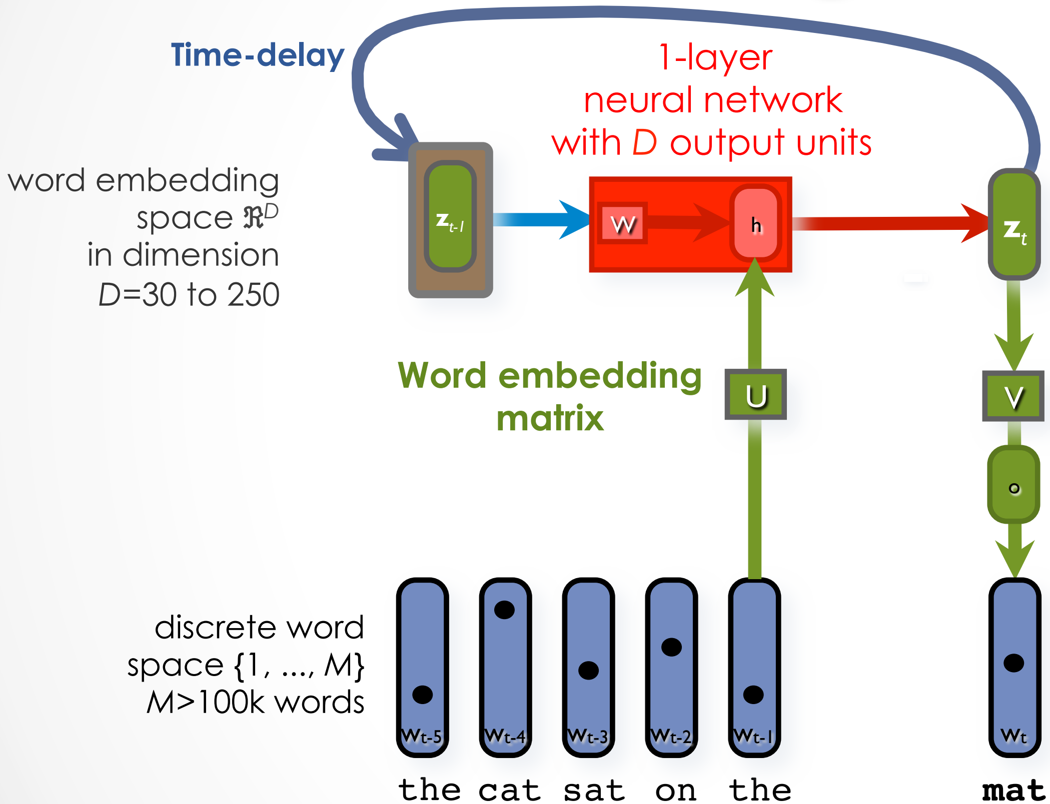
Outperforms best n-grams (Class-based Kneyser-Ney back-off 5-grams) by 24%

Took weeks to train (in 2009-2010) on AP News corpus (14M words)

Complexity: (n-1)×D + (n-1)×D×H + H×D + D×V

[Mnih & Hinton, *Neural Computation*, 2009]

Recurrent Neural Net (RNN) language model



Complexity: $D \times D + D \times D +$ ***D×V***

$$\mathbf{z}_t = \sigma\left(\mathbf{W}\mathbf{z}_{t-1} + \mathbf{U}\mathbf{w}_t\right)$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\mathbf{o} = \mathbf{V}\mathbf{z}_t$$

$$P\left(w_t \mid \mathbf{w}_{t-n+1}^{t-1}\right) = \mathbf{y}_t = \frac{e^{o(w)}}{\sum_v e^{o(v)}}$$

Handles **longer word history** (~10 words) as well as 10-gram feed-forward NNLM

Training algorithm: BPTT
Back-Propagation Through Time

[Mikolov et al, 2010, 2011]

Learning neural language models

- **Maximize the log-likelihood** of observed data, w.r.t. parameters θ of the neural language model

$$L_t = \log P\left(w_t = w \mid \mathbf{w}_1^{t-1}\right) = \boxed{s_{\boldsymbol{\theta}}(w)} \boxed{- \log \sum_{v=1}^{V} e^{s_{\boldsymbol{\theta}}(v)}}$$

$$\arg\max_{\boldsymbol{\theta}} \left(\log P\left(w_t = w \mid \mathbf{w}_1^{t-1}, \boldsymbol{\theta}\right)\right)$$

- **Parameters** θ (in a neural language model):
  - Word embedding matrix **R** and bias $\mathbf{b}_V$
  - Neural weights: **A**, $\mathbf{b}_A$, **B**, $\mathbf{b}_B$
- **Gradient descent** with learning rate η:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial L_t}{\partial \boldsymbol{\theta}}$$

Maximizing the loss function

$$P\left(w_t = w \mid \mathbf{w}_1^{t-1}\right) = \frac{\boxed{e^{s_\theta(w)}}}{\boxed{\sum_{v=1}^{V} e^{s_\theta(v)}}}$$

- **Maximum Likelihood learning:**

$$L_t = \log P\left(w_t = w \mid \mathbf{w}_1^{t-1}\right) = \boxed{s_\theta(w)} - \boxed{\log \sum_{v=1}^{V} e^{s_\theta(v)}}$$

  - Gradient of log-likelihood w.r.t. **parameters** θ:

$$\frac{\partial L_t}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} \log P\left(w_t = w \mid \mathbf{w}_1^{t-1}\right)$$

$$\frac{\partial L_t}{\partial \boldsymbol{\theta}} = \boxed{\frac{\partial}{\partial \boldsymbol{\theta}} s_\theta(w)} - \boxed{\sum_{v=1}^{V} P\left(v \mid \mathbf{w}_1^{t-1}\right) \frac{\partial}{\partial \boldsymbol{\theta}} s_\theta(v)}$$

  - Use the chain rule of gradients

Maximizing the loss function: example of LBL

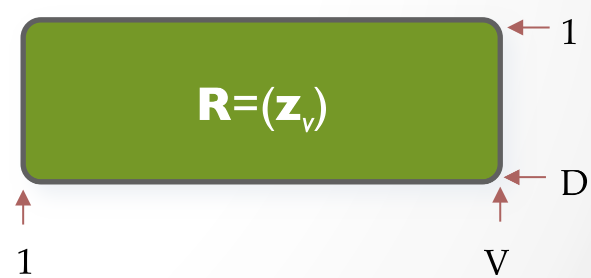
- **Maximum Likelihood learning:**

$$P\left(w_t = w \mid \mathbf{w}_1^{t-1}\right) = \frac{\boxed{e^{s_{\boldsymbol{\theta}}(w)}}}{\boxed{\sum_{v=1}^{V} e^{s_{\boldsymbol{\theta}}(v)}}} \qquad s_{\boldsymbol{\theta}}(v) = \hat{\mathbf{z}}_t^T \mathbf{z}_v + b_v$$

  - Gradient of log-likelihood w.r.t. **parameters** θ :

$$\frac{\partial L_t}{\partial \boldsymbol{\theta}} = \boxed{\frac{\partial}{\partial \boldsymbol{\theta}} s_{\boldsymbol{\theta}}(w)} - \boxed{\sum_{v=1}^{V} P\left(v \mid \mathbf{w}_1^{t-1}\right) \frac{\partial}{\partial \boldsymbol{\theta}} s_{\boldsymbol{\theta}}(v)}$$

```
function [dL_dz_hat, dL_dR, dL_dbias_v, w] = ...
  Loss_BackProp(z_hat, model, p, w)
% Gradient of loss w.r.t. word bias parameter
dL_dbias_v = -p;
dL_dbias_v(w) = 1 - p;
% Gradient of loss w.r.t. prediction of (N)LBL model
dL_dz_hat = model.R(:, w) – model.R * p;
% Gradient of loss w.r.t. vocabulary matrix R
dL_dR = –z_hat * p';
dL_dR(:, w) = z_hat * (1 – p(w));
```



  - Neural net: back-propagate gradient $\dfrac{\partial L_t}{\partial \hat{\mathbf{z}}_t}$

$$\frac{\partial L_t}{\partial \boldsymbol{\theta}} = \frac{\partial L_t}{\partial \hat{\mathbf{z}}_t} \frac{\partial \hat{\mathbf{z}}_t}{\partial \boldsymbol{\theta}}$$

Learning neural language models

Randomly choose a **mini-batch** (e.g., 1000 consecutive words)

1. Forward-propagate through **word embeddings** and through **model**
2. Estimate word likelihood (loss)
3. Back-propagate loss
4. Gradient step to update model

Stochastic Gradient Descent (SGD)

- Choice of the **learning hyperparameters**
  - Learning rate?
  - Learning rate decay?
  - Regularization (L2-norm) of the parameters?
  - Momentum term on the parameters?
- Use **cross-validation** on validation set
  - E.g., on AP News (16M words)
    - Training set: 14M words
    - Validation set: 1M words
    - Test set: 1M words

Outline

- Motivations
  - Probabilistic **language models** (LMs) and **n-grams**
  - Distributional semantics
- Neural Probabilistic LMs
  - **Vector-space representation** of words
  - **Neural** probabilistic language model
  - **Log-Bilinear** (LBL) LMs
  - **Recurrent** Neural Network LMs
- Applications
  - Word representation
  - Speech recognition and machine translation
  - Sentence completion and **linguistic regularities**
- Bag-of-word-vector approaches
  - **Continuous bag-of-words** and **skip-gram** models
- Scalability with large vocabularies
  - Tree-structured LMs
  - Negative sampling

Word embeddings obtained on Reuters

- Example of word embeddings obtained using our language model on the Reuters corpus (1.5 million words, vocabulary V=12k words), vector space of dimension D=100
- For each word, the 10 nearest neighbours in the vector space retrieved using cosine similarity:

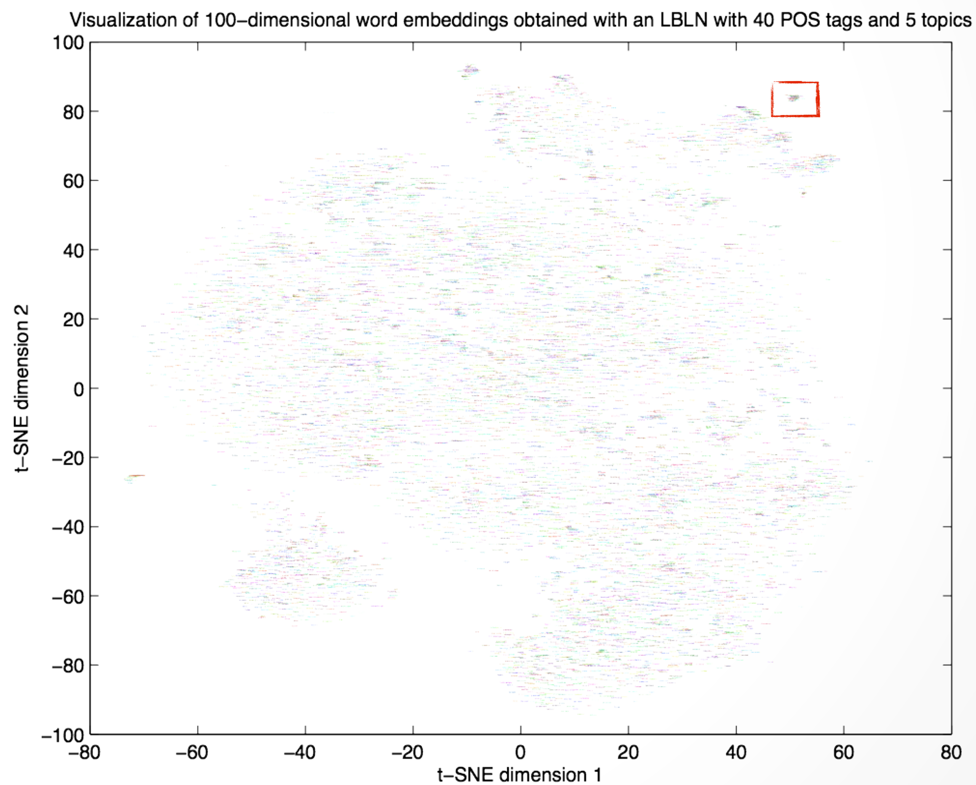
debt	aa	decrease	met	slow
financing	aaa	drop	introduced	moderate
funding	bbb	decline	rejected	lower
debts	aa-minus	rise	sought	steady
loans	b-minus	increase	supported	slowing
borrowing	a-1	fall	called	double
short-term	bb-minus	jump	charged	higher
indebtedness	a-3	surge	joined	break
long-term	bbb-minus	reduction	adopted	weaker
principal	a-plus	limit	made	stable
capital	a-minus	slump	sent	narrow

[Mirowski, Chopra, Balakrishnan and Bangalore (2010)
"Feature-rich continuous language models for speech recognition", *SLT*]

Word embeddings obtained on AP News

Example of word embeddings obtained using our LM on AP News (14M words, V=17k), D=100

The word embedding matrix R was projected in 2D by Stochastic t-SNE [Van der Maaten, *JMLR* 2008]

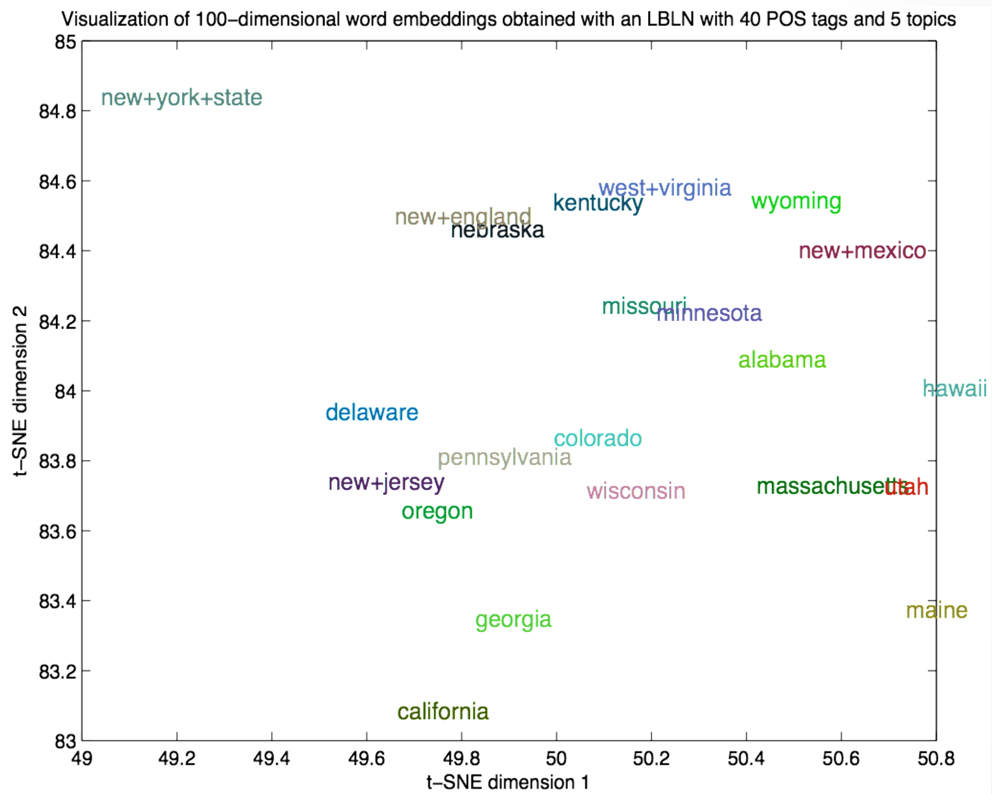


[Mirowski (2010)
"Time series modelling with hidden variables and gradient-based algorithms", *NYU PhD thesis*]

Word embeddings obtained on AP News

Example of word embeddings obtained using our LM on AP News (14M words, V=17k), D=100

The word embedding matrix R was projected in 2D by Stochastic t-SNE [Van der Maaten, *JMLR* 2008]

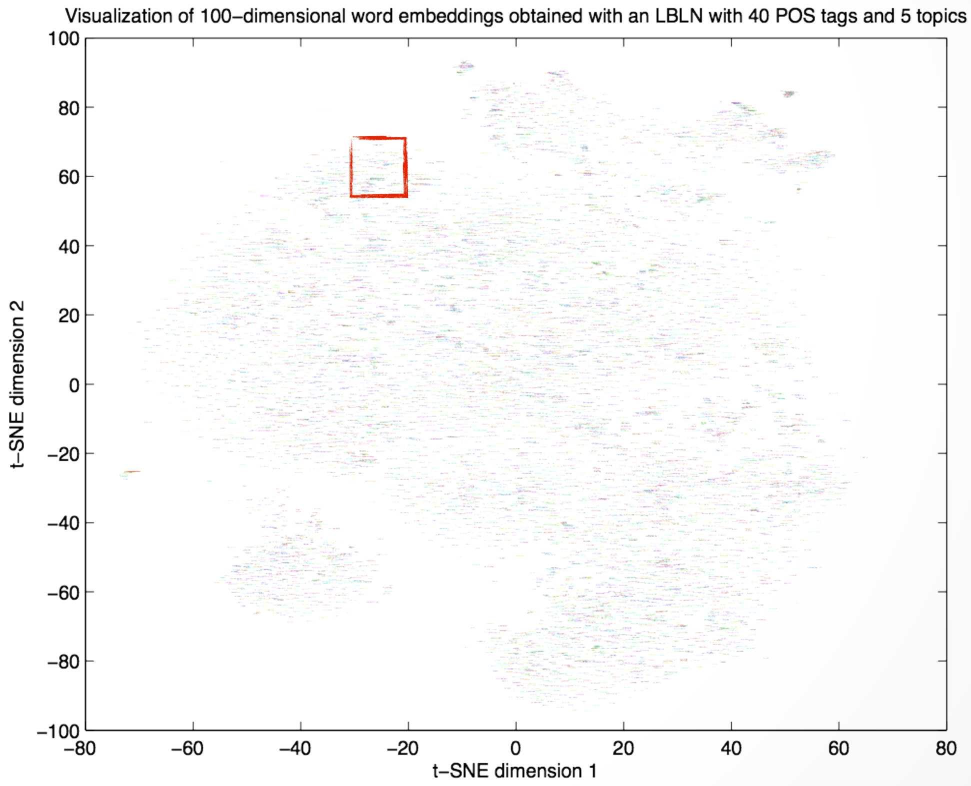


[Mirowski (2010)
"Time series modelling with hidden variables and gradient-based algorithms", *NYU PhD thesis*]

Word embeddings obtained on AP News

Example of word embeddings obtained using our LM on AP News (14M words, *V*=17k), *D*=100

The word embedding matrix R was projected in 2D by Stochastic t-SNE [Van der Maaten, *JMLR* 2008]

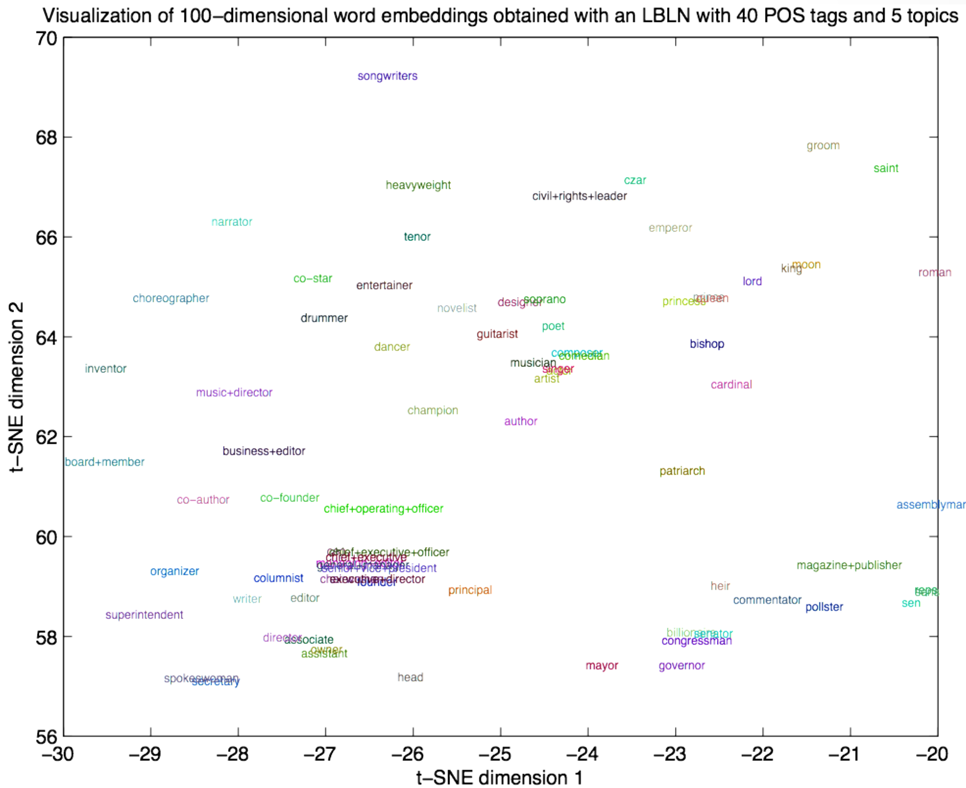


[Mirowski (2010)
"Time series modelling with hidden variables and gradient-based algorithms", *NYU PhD thesis*]

Word embeddings obtained on AP News

Example of word embeddings obtained using our LM on AP News (14M words, V=17k), D=100

The word embedding matrix R was projected in 2D by Stochastic t-SNE [Van der Maaten, *JMLR* 2008]



Performance of LBL on speech recognition

HUB-4 TV broadcast transcripts
Vocabulary V=25k
(with proper nouns & numbers)

Train on 1M words
Validate on 50k words
Test on 800 sentences

Re-rank top 100 candidate sentences, provided for each spoken sentence by a speech recognition system (acoustic model + simple trigram)

#topics	POS	Word accuracy	Method
-	-	*63.7%*	*AT&T Watson* [Goffin et al, 2005]
-	-	*63.5%*	*KN 5-grams on 100-best list*
-	-	*66.6%*	*Oracle: best of 100-best list*
-	-	*57.8%*	*Oracle: worst of 100-best list*
0	-	64.1%	Log-Bilinear models with nonlinearity and optional POS tag inputs and LDA topic model mixtures
0	F=34	64.1%	
0	F=3	64.1%	
5	-	64.2%	
5	F=34	**64.6%**	
5	F=3	**64.6%**	

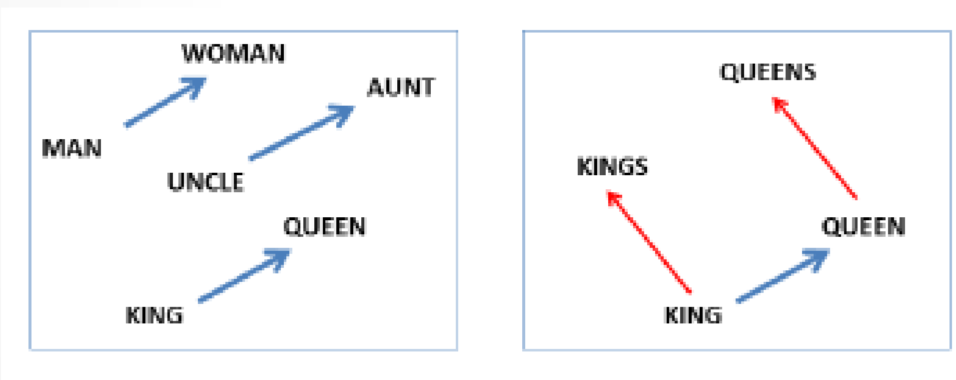
[Mirowski et al, 2010]

Syntactic and Semantic tests with RNN

Observed that word embeddings obtained by RNN-LDA have linguistic regularities "a" is to "b" as "c" is to _

Syntactic: king is to kings as queen is to **queens**

Semantic: clothing is to shirt as dish is to **bowl**



[Image credits: Mikolov et al (2013) "Efficient Estimation of Word Representation in Vector Space", *arXiv*]

Vector offset method

$$z_1 - z_2 + z_3 = \hat{z} \qquad z_v$$

cosine similarity

Microsoft Research Sentence Completion Task

- 1024 sentences with 1 missing word each
- 5 choices for each word
  - **Ground truth** and **4 impostor words**

```
That is his generous   fault, but on the whole he's a good worker.
That is his mother's   fault, but on the whole he's a good worker.
That is his successful fault, but on the whole he's a good worker.
That is his main       fault, but on the whole he's a good worker.
That is his favourite  fault, but on the whole he's a good worker.
```

- Human performance: 90% accuracy

Table 7: *Comparison and combination of models on the Microsoft Sentence Completion Challenge.*

Architecture	Accuracy [%]
4-gram [32]	39
Average LSA similarity [32]	49
Log-bilinear model [24]	54.8
RNNLMs [19]	55.4
Skip-gram	48.0
Skip-gram + RNNLMs	**58.9**

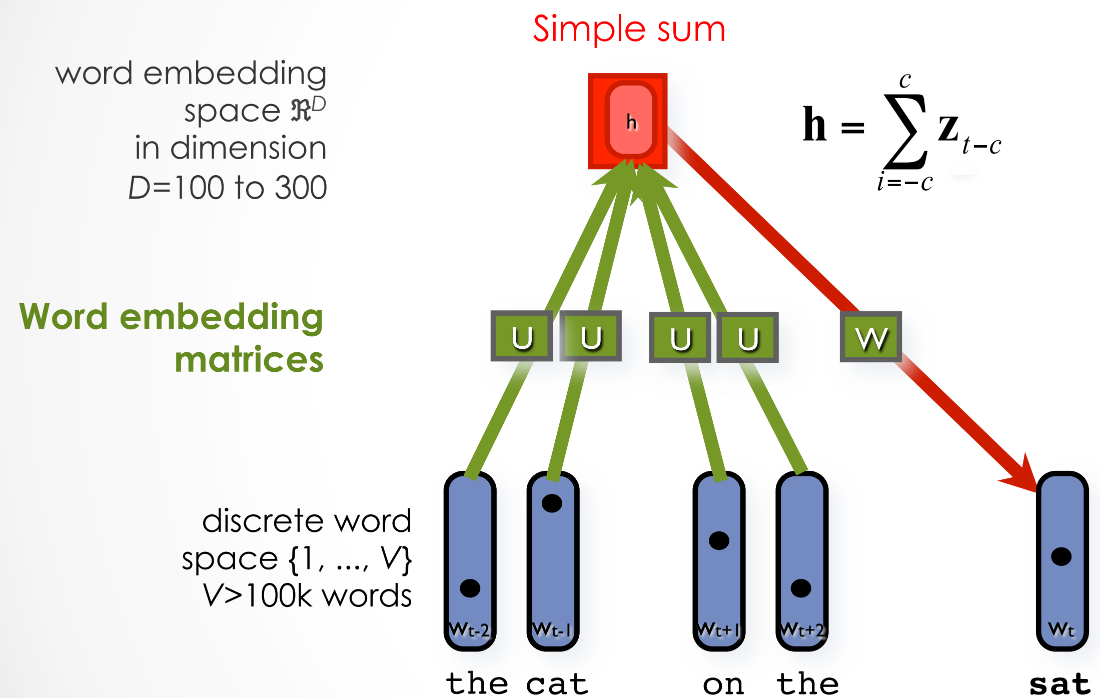
[Image credits: Mikolov et al (2013) "Efficient Estimation of Word Representation in Vector Space", *arXiv*]

[Zweig & Burges, 2011; Mikolov et al, 2013a; http://research.microsoft.com/apps/pubs/default.aspx?id=157031]

Outline

- Motivations
  - Probabilistic **language models** (LMs) and **n-grams**
  - Distributional semantics
- Neural Probabilistic LMs
  - **Vector-space representation** of words
  - **Neural** probabilistic language model
  - **Log-Bilinear** (LBL) LMs
  - **Recurrent** Neural Network LMs
- Applications
  - Word representation
  - Speech recognition and machine translation
  - Sentence completion and **linguistic regularities**
- Bag-of-word-vector approaches
  - **Continuous bag-of-words** and **skip-gram** models
- Scalability with large vocabularies
  - Tree-structured LMs
  - Negative sampling

Continuous Bag-of-Words



$$\mathbf{h} = \sum_{i=-c}^{c} \mathbf{z}_{t-c}$$

$$\mathbf{o} = \mathbf{W}\mathbf{h}$$

$$P\left(w_t \mid \mathbf{w}_{t-c}^{t-1}, \mathbf{w}_{t+1}^{t+c}\right) = \frac{e^{o(w)}}{\sum_v e^{o(v)}}$$

Extremely efficient estimation of **word embeddings** in matrix **U** **without a Language Model**.
Can be used as input to neural LM.
Enables much larger datasets, e.g., Google News (6B words, V=1M)

Complexity: 2C×D + ***D*×*V***

Complexity: 2C×D + ***D*×log(*V*)** (**hierarchical softmax using tree factorization**)

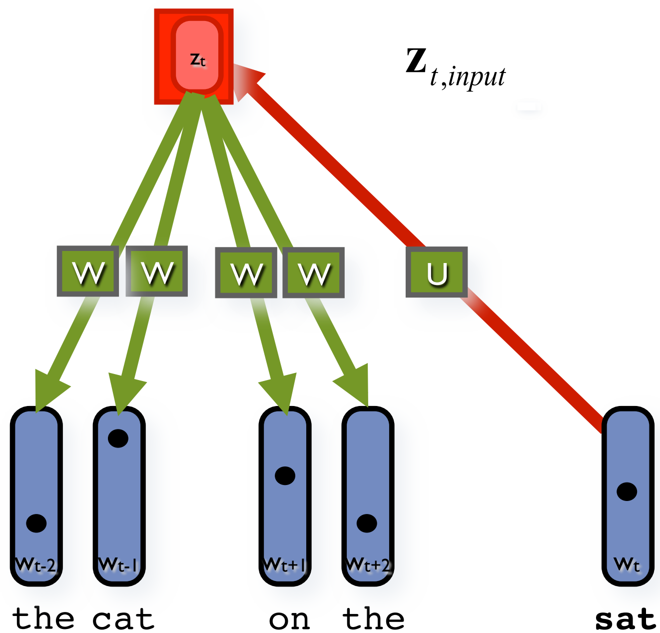
[Mikolov et al, 2013a; Mnih & Kavukcuoglu, 2013; http://code.google.com/p/word2vec]

Skip-gram

word embedding space $\Re^D$ in dimension D=100 to 1000

Word embedding matrices

discrete word space {1, ..., V} V>100k words



$$s_{\theta}(v,c) = \mathbf{z}_{v,output}^{T}\mathbf{z}_{t,input}$$

$$P(w_{t+c} \mid w_t) = \frac{e^{s_{\theta}(w,c)}}{\sum_v e^{s_{\theta}(v,c)}}$$

Extremely efficient estimation of **word embeddings** in matrix **U** **without a Language Model**.
Can be used as input to neural LM.
Enables much larger datasets, e.g., Google News (33B words, V=1M)

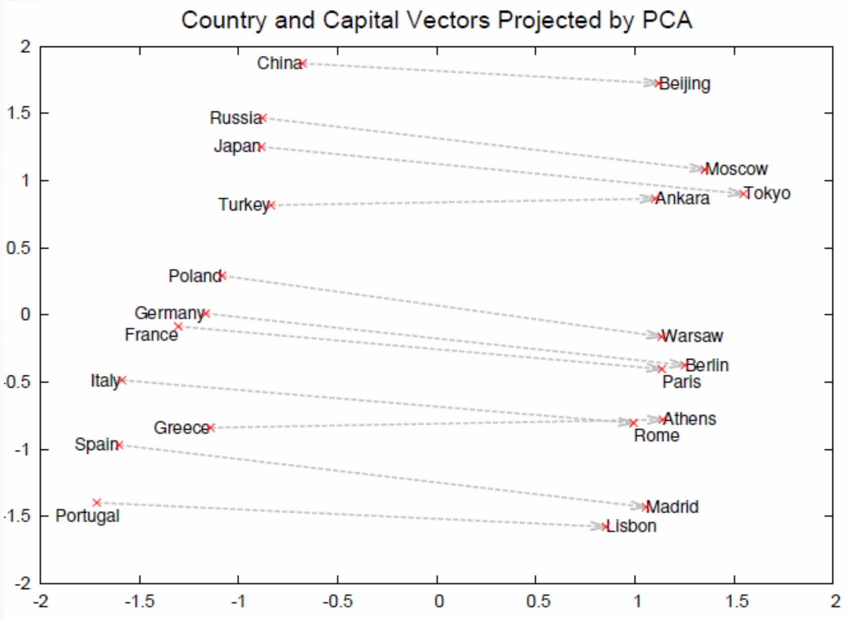
Complexity: 2C×D + **2C×D×V**

Complexity: 2C×D + **2C×D×log(V)** (**hierarchical softmax using tree factorization**)

Complexity: 2C×D + **2C×D×(k+1)** (**negative sampling with k negative examples**)

[Mikolov et al, 2013a, 2013b; Mnih & Kavukcuoglu, 2013; http://code.google.com/p/word2vec]

Vector-space word representation without LM



[Image credits: Mikolov et al (2013) "Distributed Representations of Words and Phrases and their Compositionality", *NIPS*]

Word and phrase representation learned by skip-gram **exhibit linear structure** that enables **analogies with vector arithmetics**.

This is **due to training objective**, input and output (before softmax) are in **linear relationship**.

The sum of vectors in the loss function is the sum of log-probabilities (or log of product of probabilities), i.e., comparable to the AND function.

[Mikolov et al, 2013a, 2013b; http://code.google.com/p/word2vec]

Examples of Word2Vec embeddings

Example of word embeddings obtained using Word2Vec on the 3.2B word Wikipedia:

- Vocabulary *V*=2M
- Continuous vector space *D*=200
- Trained using CBOW

debt	aa	decrease	met	slow	france	jesus	xbox
debts	aaarm	increase	meeting	slower	marseille	christ	playstation
repayments	samavat	increases	meet	fast	french	resurrection	wii
repayment	obukhovskii	decreased	meets	slowing	nantes	savior	xbla
monetary	emerlec	greatly	had	slows	vichy	miscl	wiiware
payments	gunss	decreasing	welcomed	slowed	paris	crucified	gamecube
repay	dekhen	increased	insisted	faster	bordeaux	god	nintendo
mortgage	minizini	decreases	acquainted	sluggish	aubagne	apostles	kinect
repaid	bf	reduces	satisfied	quicker	vend	apostle	dsiware
refinancing	mortardepth	reduce	first	pace	vienne	bickertonite	eshop
bailouts	ee	increasing	persuaded	slowly	toulouse	pretribulational	dreamcast

[Mikolov et al, 2013a, 2013b; http://code.google.com/p/word2vec]

Semantic-syntactic word evaluation task

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

[Image credits: Mikolov et al (2013) "Efficient Estimation of Word Representation in Vector Space", *arXiv*]

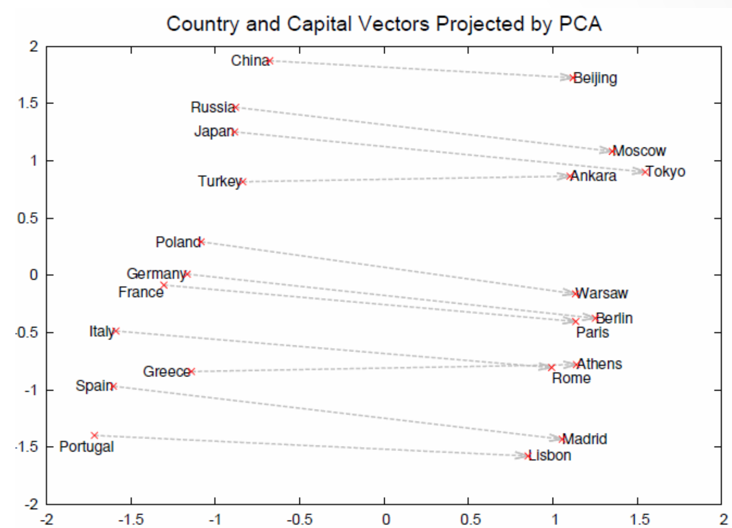
[Mikolov et al, 2013a, 2013b; http://code.google.com/p/word2vec]

Performance on the semantic-syntactic task

Table 4: *Comparison of publicly available word vectors on the Semantic-Syntactic Word Relationship test set, and word vectors from our models. Full vocabularies are used.*

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

[Image credits: Mikolov et al (2013) "Efficient Estimation of Word Representation in Vector Space", *arXiv*]



[Image credits: Mikolov et al (2013) "Distributed Representations of Words and Phrases and their Compositionality", *NIPS*]

Word and phrase representation learned by skip-gram
exhibit linear structure that enables analogies with vector arithmetics.
Due to training objective, input and output (before softmax) in linear relationship.
Sum of vectors is like sum of log-probabilities, i.e. log of product of probabilities, i.e., AND function.

[Mikolov et al, 2013a, 2013b; http://code.google.com/p/word2vec]

Outline

- Motivations
  - Probabilistic **language models** (LMs) and **n-grams**
  - Distributional semantics
- Neural Probabilistic LMs
  - **Vector-space representation** of words
  - **Neural** probabilistic language model
  - **Log-Bilinear** (LBL) LMs
  - **Recurrent** Neural Network LMs
- Applications
  - Word representation
  - Speech recognition and machine translation
  - Sentence completion and **linguistic regularities**
- Bag-of-word-vector approaches
  - **Continuous bag-of-words** and **skip-gram** models
- Scalability with large vocabularies
  - Tree-structured LMs
  - Negative sampling

Computational bottleneck of large vocabularies

target word	$w(t)$
word history	$\mathbf{w}_1^{t-1}$
scoring function	$s_v(t) = s\left(\mathbf{w}_1^{t-1}, v\right)$
softmax	$g(s_v) = \dfrac{e^{s_v}}{\sum_{v'=1}^{V} e^{s_{v'}}}$

$$P\left(w_t = v \mid \mathbf{w}_1^{t-1}\right) = g\left(s_v(t)\right)$$

- Bulk of computation at **word prediction** and at **input word embedding** layers
- **Training can take days or even weeks**
- Large vocabularies:
  - AP News (14M words; V=17k)
  - HUB-4 (1M words; V=25k)
  - Google News (6B words, V=1M)
  - Wikipedia (3.2B, V=2M)
- Strategies to compress output softmax

Hierarchical softmax by grouping words

target word $w(t)$

word history $\mathbf{w}_1^{t-1}$

scoring function $s_{\boldsymbol{\theta}}(v) = s\left(\mathbf{w}_1^{t-1}, v; \boldsymbol{\theta}\right)$

softmax $g\left(s_{\boldsymbol{\theta}}(v)\right) = \dfrac{e^{s_{\boldsymbol{\theta}}(v)}}{\sum_{v'=1}^{V} e^{s_{\boldsymbol{\theta}}(v')}}$

$$P\left(w_t = v \mid \mathbf{w}_1^{t-1}\right) = g\left(s_{\boldsymbol{\theta}}(v)\right)$$

$$P\left(w_t = v \mid \mathbf{w}_1^{t-1}\right) = \boxed{P\left(c \mid \mathbf{w}_1^{t-1}\right)} \times \boxed{P\left(v \mid \mathbf{w}_1^{t-1}, c\right)}$$

$$P\left(w_t = v \mid \mathbf{w}_1^{t-1}\right) = \boxed{g\left(s_{\boldsymbol{\theta}}(c)\right)} \times \boxed{g\left(s_{\boldsymbol{\theta}}(c, v)\right)}$$

- Group words into **disjoint classes**:
  - E.g., 20 classes with frequency binning
  - Use unigram frequency
  - Top 5% words ("the") go to class 1
  - Following 5% words go to class 2
- Factorize word probability into:
  - **Class probability**
  - **Class-conditional word probability**
- Speed-up factor:
  - O(|V|) to O(|C|+max|VC|)

[Mikolov et al, 2011, Auli et al, 2013]

Hierarchical softmax by grouping words

target word $w(t)$

word history $\mathbf{w}_1^{t-1}$

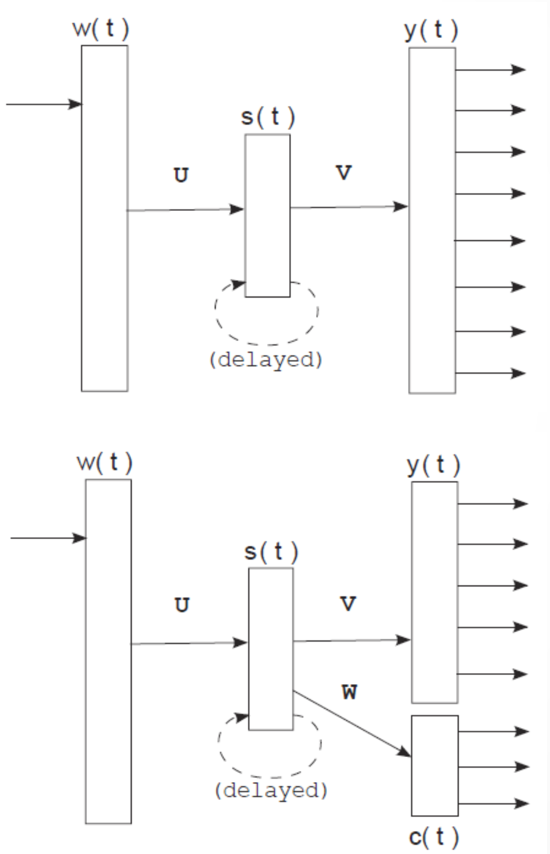
scoring function $s_{\boldsymbol{\theta}}(v) = s\left(\mathbf{w}_1^{t-1}, v; \boldsymbol{\theta}\right)$

softmax $g\left(s_{\boldsymbol{\theta}}(v)\right) = \dfrac{e^{s_{\boldsymbol{\theta}}(v)}}{\sum_{v'=1}^{V} e^{s_{\boldsymbol{\theta}}(v')}}$

$$P\left(w_t = v \mid \mathbf{w}_1^{t-1}\right) = g\left(s_{\boldsymbol{\theta}}(v)\right)$$

$$P\left(w_t = v \mid \mathbf{w}_1^{t-1}\right) = \boxed{P\left(c \mid \mathbf{w}_1^{t-1}\right)} \times \boxed{P\left(v \mid \mathbf{w}_1^{t-1}, c\right)}$$

$$P\left(w_t = v \mid \mathbf{w}_1^{t-1}\right) = \boxed{g\left(s_{\boldsymbol{\theta}}(c)\right)} \times \boxed{g\left(s_{\boldsymbol{\theta}}(c, v)\right)}$$



[Image credits: Mikolov et al (2011) "Extensions of Recurrent Neural Network Language Model", *ICASSP*]

[Mikolov et al, 2011, Auli et al, 2013]

Negative sampling

- Probability estimation as binary classification problem:
  - **Positive examples (data)** vs. **negative examples (noise)**
  - **Scaling factor** *k*: **noisy samples** k times more likely than data samples
    - **Noise distribution**: based on **unigram word probabilities**

$$P\left(D=1 \mid w, \mathbf{w}_1^{t-1}\right)=\frac{\boxed{e^{s_\theta(w)}}}{\boxed{e^{s_\theta(w)}}+\boxed{kP_{noise}(w)}}$$

$$P\left(D=1 \mid w, \mathbf{w}_1^{t-1}\right)=\sigma\left(s_\theta(w)\right)$$

- **Negative sampling**
  - Remove normalization term in probabilities

$$L_t{}'=\boxed{\log\sigma\left(s_\theta(w)\right)}+\boxed{\sum_{i=1}^{k} E_{P_{noise}}\left[\log\sigma\left(-s_\theta(v_i)\right)\right]}$$

- Compare to **Maximum Likelihood learning**:

$$L_t=\boxed{s_\theta(w)}-\boxed{\log\sum_{v=1}^{V} e^{s_\theta(v)}}$$

[Mnih & Teh, 2012; Mnih & Kavukcuoglu, 2013; Mikolov et al, 2013a, 2013b]

Speed-up over full softmax

LBL with **full softmax**, trained on APNews data, **14M words**, **V=17k** **7days**

Skip-gram (context 5) with phrases, trained using **negative sampling**, on Google data, **33G** words, **V=692k + phrases** **1 day**

Model (training time)	Redmond	Havel	ninjutsu	graffiti	capitulate
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona karate	cheesecake gossip dioramas	abdicate accede rearm
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -	gunfire emotion impunity	- - -
Mnih (100d) (7 days)	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	- - -	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-Phrase (1000d, 1 day)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	ninja martial arts swordsmanship	spray paint grafitti taggers	capitulation capitulated capitulating

[Image credits: Mikolov et al (2013) "Distributed Representations of Words and Phrases and their Compositionality", *NIPS*]

LBL (2-gram, 100d) with **full softmax**, **1 day**

LBL (2-gram, 100d) with **noise contrastive estimation** **1.5 hours**

RNN (100d) with **50-class hierarchical softmax** **0.5 hours** (own experience)

Training algorithm	Number of samples	Test PPL	Training time (h)
ML		163.5	21
NCE	1	192.5	1.5
NCE	5	172.6	1.5
NCE	25	163.1	1.5
NCE	100	159.1	1.5
RNN (HS)	50 classes	145.4	0.5

Penn TreeBank data (900k words, V=10k)

[Image credits: Mnih & Teh (2012) "A fast and simple algorithm for training neura probabilistic language models", *ICML*]

[Mnih & Teh, 2012; Mikolov et al, 2010-2012, 2013b]

Thank you!

- Further references: following this slide
- Basic (N)LBL Matlab code: available on demand
- Contact: piotr.mirowski@computer.org

References

- **Basic n-grams with smoothing and backtracking (no word vector representation):**

  - S. Katz, (1987)
    "Estimation of probabilities from sparse data for the language model component of a speech recognizer",
    IEEE Transactions on Acoustics, Speech and Signal Processing,
    vol. ASSP-35, no. 3, pp. 400–401
    https://www.mscs.mu.edu/~cstruble/moodle/file.php/3/papers/01165125.pdf
  - S. F. Chen and J. Goodman (1996)
    "An empirical study of smoothing techniques for language modelling",
    ACL
    http://acl.ldc.upenn.edu/P/P96/P96-1041.pdf?origin=publication_detail
  - A. Stolcke (2002)
    "SRILM - an extensible language modeling toolkit"
    ICSLP, pp. 901–904
    http://my.fit.edu/~vkepuska/ece5527/Projects/Fall2011/Sundaresan,%20Venkata%20Subramanyan/srilm/doc/paper.pdf

References

- **Neural network language models:**
  - Y. Bengio, R. Ducharme, P. Vincent and J.-L. Jauvin (2001, 2003)
    "A Neural Probabilistic Language Model",
    NIPS (2000) 13:933-938
    J. Machine Learning Research (2003) 3:1137-115
    http://www.iro.umontreal.ca/~lisa/pointeurs/BengioDucharmeVincentJauvin_jmlr.pdf
  - F. Morin and Y. Bengio (2005)
    "Hierarchical probabilistic neural network language model",
    AISTATS
    http://core.kmi.open.ac.uk/download/pdf/22017.pdf#page=255
  - Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, J.-L. Gauvain (2006)
    "Neural Probabilistic Language Models",
    Innovations in Machine Learning, vol. 194, pp 137-186
    http://rd.springer.com/chapter/10.1007/3-540-33486-6_6

References

- **Linear and/or nonlinear (neural network-based) language models**:
  - A. Mnih and G. Hinton (2007)
    "Three new graphical models for statistical language modelling",
    ICML, pp. 641–648, http://www.cs.utoronto.ca/~hinton/absps/threenew.pdf
  - A. Mnih, Y. Zhang, and G. Hinton (2009)
    "Improving a statistical language model through non-linear prediction",
    Neurocomputing, vol. 72, no. 7-9, pp. 1414 – 1418
    http://www.sciencedirect.com/science/article/pii/S0925231209000083
  - A. Mnih and Y.-W. Teh (2012)
    "A fast and simple algorithm for training neural probabilistic language models"
    ICML, http://arxiv.org/pdf/1206.6426
  - A. Mnih and K. Kavukcuoglu (2013)
    "Learning word embeddings efficiently with noise-contrastive estimation"
    NIPS
    http://papers.nips.cc/paper/5165-learning-word-embeddings-efficiently-with-noise-contrastive-estimation.pdf

References

- **Recurrent neural networks (long-term memory of word context):**
  - Tomas Mikolov, M Karafiat, J Cernocky, S Khudanpur (2010) "Recurrent neural network-based language model" *Interspeech*
  - T. Mikolov, S. Kombrink, L. Burger, J. Cernocky and S. Khudanpur (2011) "Extensions of Recurrent Neural Network Language Model" *ICASSP*
  - Tomas Mikolov and Geoff Zweig (2012) "Context-dependent Recurrent Neural Network Language Model" *IEEE Speech Language Technologies*
  - Tomas Mikolov, Wen-Tau Yih and Geoffrey Zweig (2013) "Linguistic Regularities in Continuous SpaceWord Representations" *NAACL-HLT* https://www.aclweb.org/anthology/N/N13/N13-1090.pdf
  - http://research.microsoft.com/en-us/projects/rnn/default.aspx

References

- **Applications:**

  - P. Mirowski, S. Chopra, S. Balakrishnan and S. Bangalore (2010)
    "Feature-rich continuous language models for speech recognition",
    SLT
  - G. Zweig and C. Burges (2011)
    "The Microsoft Research Sentence Completion Challenge"
    MSR Technical Report MSR-TR-2011-129
  - http://research.microsoft.com/apps/pubs/default.aspx?id=157031
  - M. Auli, M. Galley, C. Quirk and G. Zweig (2013)
    "Joint Language and Translation Modeling with Recurrent Neural Networks"
    EMNLP
  - K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi and D. Yu (2013)
    "Recurrent Neural Networks for Language Understanding"
    Interspeech

References

- **Continuous Bags of Words, Skip-Grams, Word2Vec:**
  - Tomas Mikolov et al (2013)
    "Efficient Estimation of Word Representation in Vector Space"
    arXiv.1301.3781v3
  - Tomas Mikolov et al (2013)
    "Distributed Representation of Words and Phrases and their Compositionality"
    arXiv.1310.4546v1, *NIPS*
  - http://code.google.com/p/word2vec

Probabilistic Language Models

- Goal:
  score sentences according to their likelihood
  - Machine Translation:
    - *P*(**high** winds tonight) > *P*(**large** winds tonight)
  - Spell Correction
    - The office is about fifteen **minuets** from my house
    - *P*(about fifteen **minutes** from) > *P*(about fifteen **minuets** from)
  - Speech Recognition
    - *P*(**I saw a van**) >> *P*(**eyes awe of an**)
    - Re-ranking n-best lists of sentences produced by an acoustic model, taking the best
- Secondary goal:
  sentence completion or generation

Example of a bigram language model

Training data

There is a big house
I buy a house
They buy the new house

Model

p(big | a) = 0.5
p(is | there) = 1
p(buy | they) = 1
p(house | a) = 0.5
p(buy | i) = 1
p(a | buy) = 0.5
p(new | the) = 1
p(house | big) = 1
p(the | buy) = 0.5
p(a | is) = 1
p(house | new) = 1
p(they | < s >) = .333

Test data

S1:
they buy a big house
*P(S1) = 0.333 * 1 * 0.5 * 0.5 * 1*
P(S1) = 0.0833

S2:
*they buy **a new** house*
P(S2) = ?

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-1})$$

Slide courtesy of Abhishek Arun

Intuitive view of perplexity

- How well can we predict next word?

I always order pizza with cheese and ____

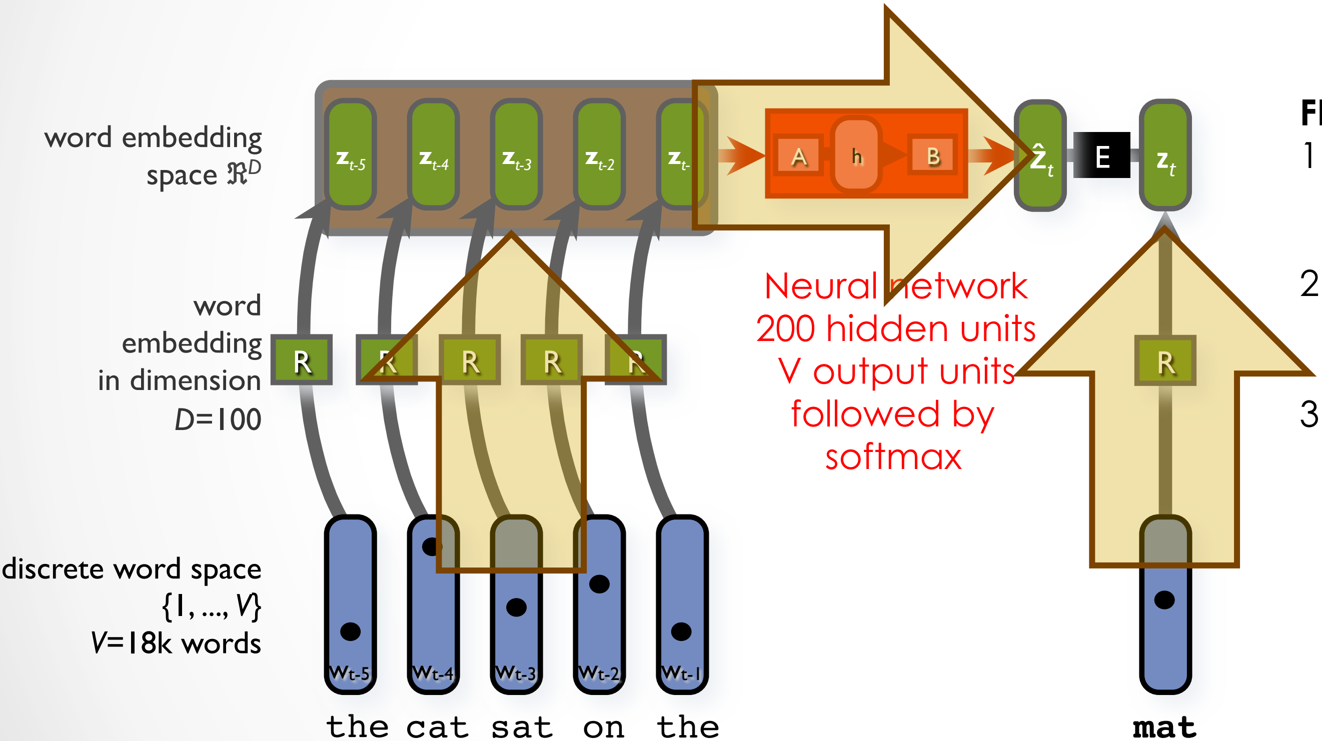
The 33rd President of the US was ____

I saw a ____

mushrooms 0.1
pepperoni 0.1
anchovies 0.01
….
fried rice 0.0001
….
and 1e-100

  - A random predictor would give each word probability 1/V where V is the size of the vocabulary
  - A better model of a text should assign a higher probability to the word that actually occurs
- Perplexity:
  - "how many words are likely to happen, given the context"
  - Perplexity of 1 means that the model recites the text by heart
  - Perplexity of V means that the model produces uniform random guesses
  - The lower the perplexity, the better the language model

Nonlinear Log-Bilinear Language Model

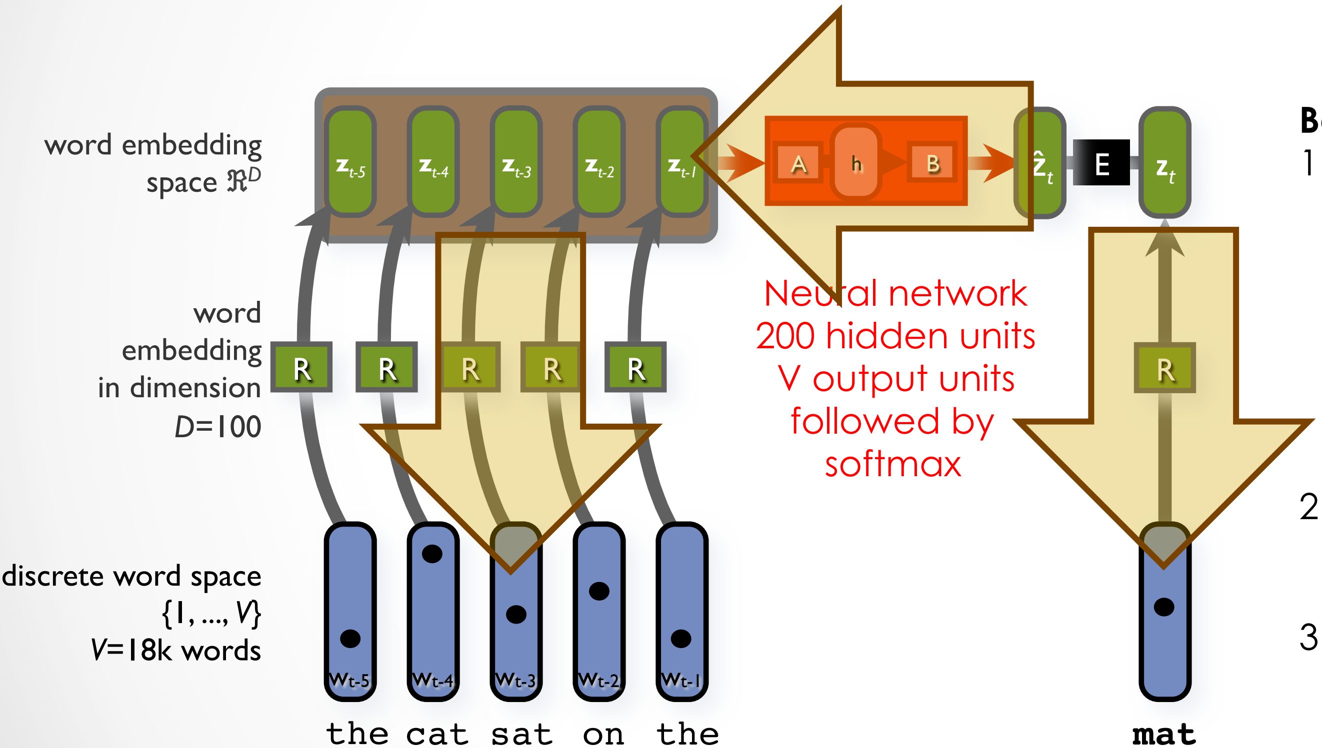


FProp

1. Look-up embeddings of the words in the n-gram using **R**
2. Forward propagate through the neural net
3. Look-up ALL vocabulary words using **R** and compute energy and probabilities (**computationally expensive**)

[Mnih & Hinton, *Neural Computation*, 2009]

Nonlinear Log-Bilinear Language Model

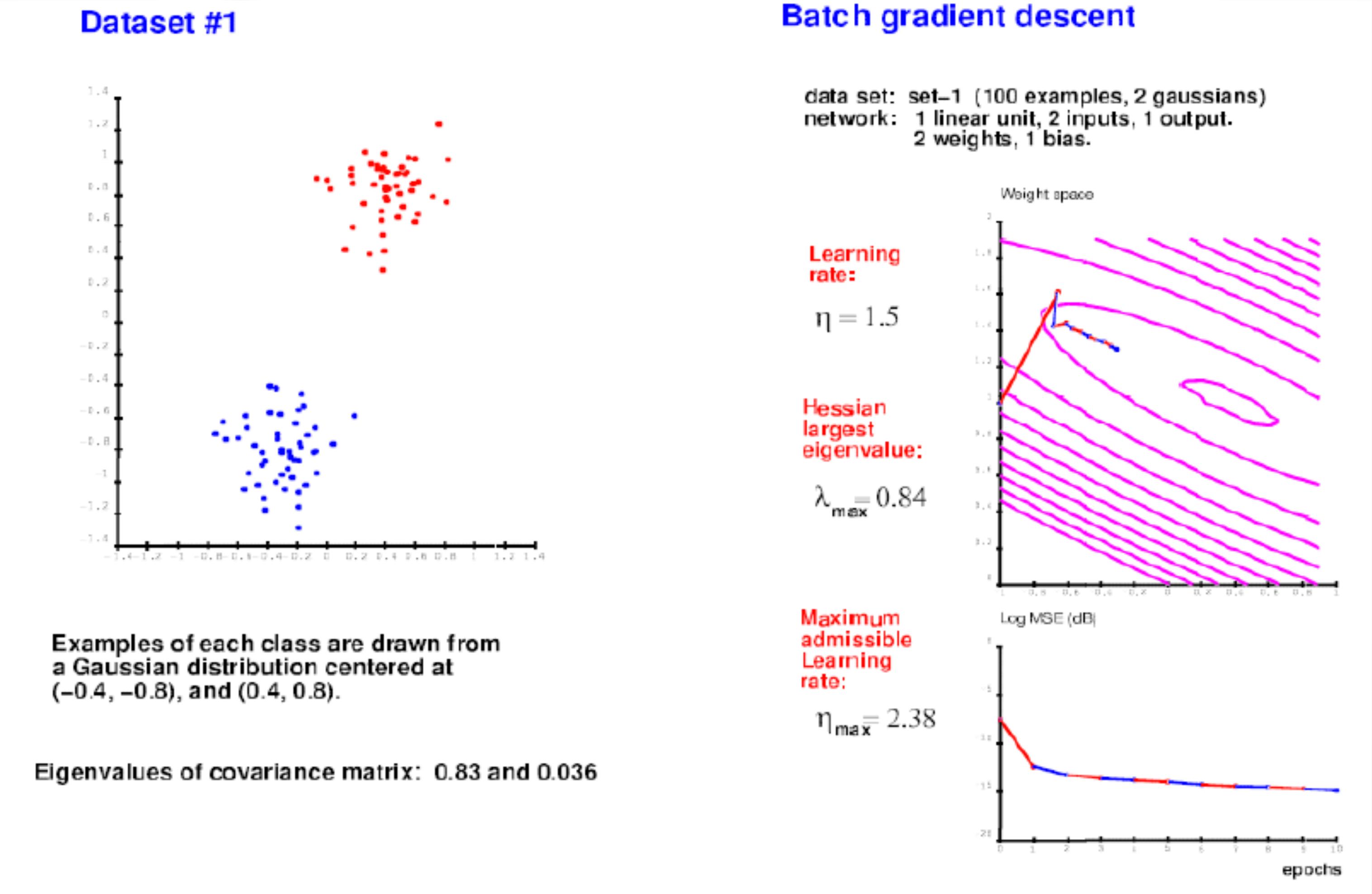


BackProp

1. Compute gradients of loss w.r.t. output of the neural net, back-propagate through neural net layers **B** and **A** (**computationally expensive**)
2. Back-propagate further down to word embeddings **R**
3. Compute gradients of loss w.r.t. words of all vocabulary, back-propagate to **R**

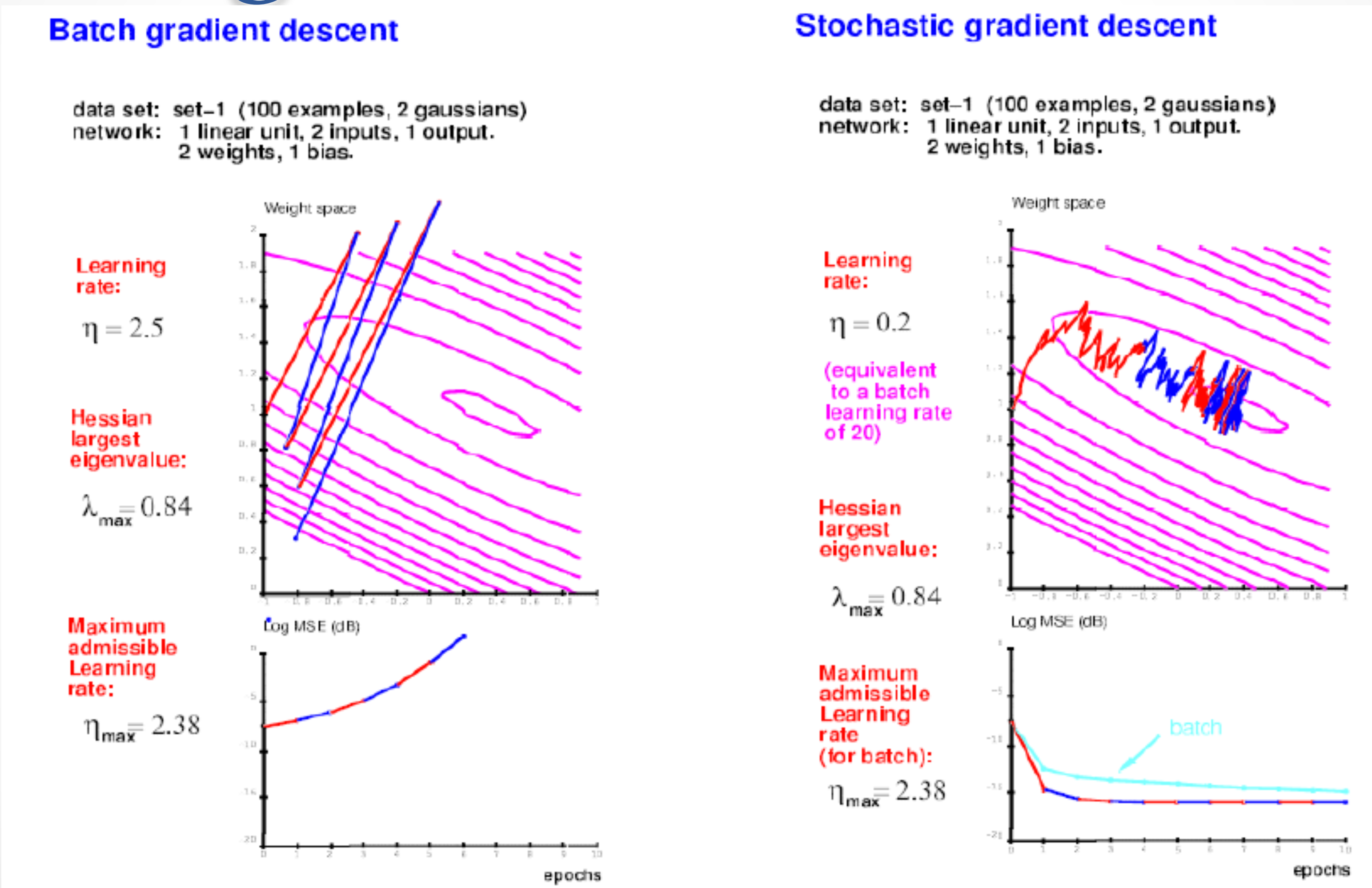
[Mnih & Hinton, *Neural Computation*, 2009]

Stochastic gradient descent



[LeCun et al, "Efficient BackProp", *Neural Networks: Tricks of the Trade*, 1998;
Bottou, "Stochastic Learning", *Slides from a talk in Tubingen*, 2003]

Stochastic gradient descent



[LeCun et al, "Efficient BackProp", *Neural Networks: Tricks of the Trade*, 1998;
Bottou, "Stochastic Learning", *Slides from a talk in Tubingen*, 2003]

Perplexity of RNN language models

	Penn Corpus	
Model	NN	NN+KN
KN5 (baseline)	-	141
feedforward NN	141	118
RNN trained by BP	137	113
RNN trained by BPTT	123	106

Penn TreeBank
V=10k vocabulary
Train on 900k words
Validate on 80k words
Test on 80k words

Model	Test ppx
Kneyser-Ney back-off 5-grams	123.3
Nonlinear LBL (100d) [Mnih & Hinton, 2009, using our implementation]	104.4
NLBL (100d) + 5 topics LDA [Mirowski, 2010, using our implementation]	98.5
RNN (200d) + 40 topics LDA [Mikolov & Zweig, 2012, using RNN toolbox]	86.9

AP News
V=17k vocabulary
Train on 14M words
Validate on 1M words
Test on 1M words

[Mirowski, 2010; Mikolov & Zweig, 2012;
RNN toolbox: http://research.microsoft.com/en-us/projects/rnn/default.aspx]

Semantic-syntactic word evaluation task

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

[Image credits: Mikolov et al (2013) "Efficient Estimation of Word Representation in Vector Space", *arXiv*]

[Mikolov et al, 2013a, 2013b; http://code.google.com/p/word2vec]

Semantic-syntactic word evaluation task

Table 4: *Comparison of publicly available word vectors on the Semantic-Syntactic Word Relationship test set, and word vectors from our models. Full vocabularies are used.*

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

[Image credits: Mikolov et al (2013) "Efficient Estimation of Word Representation in Vector Space", *arXiv*]

[Mikolov et al, 2013a, 2013b; http://code.google.com/p/word2vec]

Noise-Contrastive Estimation

- **Conditional probability of word** w in the data:

$$P\left(w_t = w \mid \mathbf{w}_1^{t-1}\right) = \frac{\boxed{e^{s_\theta(w)}}}{\boxed{\sum_{v=1}^{V} e^{s_\theta(v)}}}$$

- **Conditional probability that word** w comes from **data D** and **not from the noise distribution**:

$$P\left(D = 1 \mid w, \mathbf{w}_1^{t-1}\right) = \frac{\boxed{P_d^{\mathbf{w}_1^{t-1}}(w)}}{\boxed{P_d^{\mathbf{w}_1^{t-1}}(w)} + \boxed{kP_{noise}(w)}} \qquad P\left(D = 1 \mid w, \mathbf{w}_1^{t-1}\right) = \frac{\boxed{e^{s_\theta(w)}}}{\boxed{e^{s_\theta(w)}} + \boxed{kP_{noise}(w)}}$$

  - Auxiliary binary classification problem:
    - **Positive examples (data)** vs. **negative examples (noise)**
  - **Scaling factor** *k*: **noisy samples** k times more likely than data samples
    - **Noise distribution**: based on **unigram word probabilities**
  - Empirically, model can cope with un-normalized probabilities:

$$P_d^{\mathbf{w}_1^{t-1}}(w) \leftarrow P\left(w \mid \mathbf{w}_1^{t-1}, \theta\right) \approx e^{s_\theta(w)}$$

[Mnih & Teh, 2012; Mnih & Kavukcuoglu, 2013; Mikolov et al, 2013a, 2013b]

Noise-Contrastive Estimation

- **Conditional probability that word** w comes from **data D** and **not from the noise distribution**:

$$P\left(D=1 \mid w, \mathbf{w}_1^{t-1}\right)=\frac{\boxed{e^{s_\theta(w)}}}{\boxed{e^{s_\theta(w)}}+\boxed{kP_{noise}(w)}}$$

  - Auxiliary binary classification problem:
    - **Positive examples (data)** vs. **negative examples (noise)**
  - **Scaling factor** *k*: **noisy samples** k times more likely than data samples
    - **Noise distribution**: based on **unigram word probabilities**
  - Introduce log of difference between:
    - **score of word** w **under data distribution**
    - and **unigram distribution score of word** w

$$\Delta s_\theta(w)=\boxed{s_\theta(w)}\boxed{-\log kP_{noise}(w)}$$

$$P\left(D=1 \mid w, \mathbf{w}_1^{t-1}\right)=\sigma\left(\Delta s_\theta(w)\right)$$

$$\sigma(x)=\frac{1}{1+e^{-x}}$$

[Mnih & Teh, 2012; Mnih & Kavukcuoglu, 2013; Mikolov et al, 2013a, 2013b]

Noise-Contrastive Estimation

$$P\left(D=1 \mid w, \mathbf{w}_1^{t-1}\right)=\frac{\boxed{P_d^{\mathbf{w}_1^{t-1}}(w)}}{\boxed{P_d^{\mathbf{w}_1^{t-1}}(w)}+\boxed{kP_{noise}(w)}} \qquad P\left(D=1 \mid w, \mathbf{w}_1^{t-1}\right)=\frac{\boxed{e^{s_\theta(w)}}}{\boxed{e^{s_\theta(w)}}+\boxed{kP_{noise}(w)}}$$

- **New loss function** to maximize:

$$L_t' = \boxed{E_{P_d^{\mathbf{w}_1^{t-1}}}}\left[\log P\left(D=1 \mid w, \mathbf{w}_1^{t-1}\right)\right] + \boxed{kE_{P_{noise}}}\left[\log P\left(D=0 \mid w, \mathbf{w}_1^{t-1}\right)\right]$$

$$\frac{\partial L_t'}{\partial \boldsymbol{\theta}} = \left(1-\sigma\left(\Delta s_{\boldsymbol{\theta}}(w)\right)\right)\frac{\partial}{\partial \boldsymbol{\theta}} s_{\boldsymbol{\theta}}(w) - \sum_{i=1}^{k} \sigma\left(\Delta s_{\boldsymbol{\theta}}(v_i)\right)\frac{\partial}{\partial \boldsymbol{\theta}} s_{\boldsymbol{\theta}}(v_i)$$

- Compare to **Maximum Likelihood learning**:

$$\frac{\partial L_t}{\partial \boldsymbol{\theta}} = \boxed{\frac{\partial}{\partial \boldsymbol{\theta}} s_{\boldsymbol{\theta}}(w)} - \boxed{\sum_{v=1}^{V} P\left(v \mid \mathbf{w}_1^{t-1}\right)\frac{\partial}{\partial \boldsymbol{\theta}} s_{\boldsymbol{\theta}}(v)}$$

[Mnih & Teh, 2012; Mnih & Kavukcuoglu, 2013; Mikolov et al, 2013a, 2013b]