11. Topic Models II: Beyond LDA

DS-GA 3001, Text as Data
Arthur Spirling

April 10, 2018

Housekeeping

Housekeeping

1. HW3 out now. Should have plenty of time to finish! Note academic honesty policy.

Housekeeping

1. HW3 out now. Should have plenty of time to finish! Note academic honesty policy.
2. No lecture next week (April 17).

Housekeeping

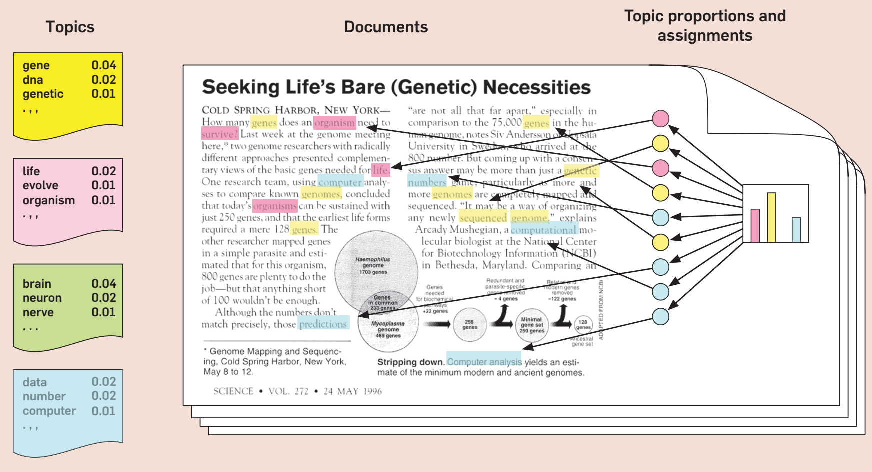
1. HW3 out now. Should have plenty of time to finish! Note academic honesty policy.
2. No lecture next week (April 17).
3. Can't do office hours today: email for an appointment.

Housekeeping

1. HW3 out now. Should have plenty of time to finish! Note academic honesty policy.
2. No lecture next week (April 17).
3. Can't do office hours today: email for an appointment.
4. Graham Neubig on neural networks at Text as Data this week.

Last time. . .

Last time. . .


Topics
Documents
Topic proportions and assignments
gene 0.04
dna 0.02
genetic 0.01
. , ,
life 0.02
evolve 0.01
organism 0.01
. , ,
brain 0.04
neuron 0.02
nerve 0.01
. . .
data 0.02
number 0.02
computer 0.01
. , ,
Seeking Life's Bare (Genetic) Necessities
COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.
Although the numbers don't match precisely, those predictions
"are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an
Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed −4 genes
Minimal gene set 250 genes
Related and modern genes removed −122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI
* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.
SCIENCE • VOL. 272 • 24 MAY 1996

Extensions and Special Cases

Extensions and Special Cases

'vanilla' LDA is a popular model.

Extensions and Special Cases

'vanilla' LDA is a popular model. But assumptions it makes are restrictive/inappropriate for some contexts...

Extensions and Special Cases

'vanilla' LDA is a popular model. But assumptions it makes are restrictive/inappropriate for some contexts...

Assn each document is mix of multiple topics.

Extensions and Special Cases

'vanilla' LDA is a popular model. But assumptions it makes are restrictive/inappropriate for some contexts...

Assn each document is mix of multiple topics.

But each document is one topic. [EAM]

Extensions and Special Cases

'vanilla' LDA is a popular model. But assumptions it makes are restrictive/inappropriate for some contexts...

Assn each document is mix of multiple topics.

But each document is one topic. [EAM]

Assn topics in documents are uncorrelated.

Extensions and Special Cases

'vanilla' LDA is a popular model. But assumptions it makes are restrictive/inappropriate for some contexts...

Assn each document is mix of multiple topics.

But each document is one topic. [EAM]

Assn topics in documents are uncorrelated.

But given topic A, we are more likely to see topic B than C. [CTM]

Extensions and Special Cases

'vanilla' LDA is a popular model. But assumptions it makes are restrictive/inappropriate for some contexts...

Assn each document is mix of multiple topics.

But each document is one topic. [EAM]

Assn topics in documents are uncorrelated.

But given topic A, we are more likely to see topic B than C. [CTM]

Assn documents are exchangeable (over time).

Extensions and Special Cases

'vanilla' LDA is a popular model. But assumptions it makes are restrictive/inappropriate for some contexts...

Assn each document is mix of multiple topics.

But each document is one topic. [EAM]

Assn topics in documents are uncorrelated.

But given topic A, we are more likely to see topic B than C. [CTM]

Assn documents are exchangeable (over time).

But time matters for what topic 'means' [DTM]

Extensions and Special Cases

'vanilla' LDA is a popular model. But assumptions it makes are restrictive/inappropriate for some contexts. . .

Assn each document is mix of multiple topics.
But each document is one topic. [EAM]

Assn topics in documents are uncorrelated.
But given topic A, we are more likely to see topic B than C. [CTM]

Assn documents are exchangeable (over time).
But time matters for what topic 'means' [DTM]

Assn no covariates

Extensions and Special Cases

'vanilla' LDA is a popular model. But assumptions it makes are restrictive/inappropriate for some contexts...

Assn each document is mix of multiple topics.
But each document is one topic. [EAM]

Assn topics in documents are uncorrelated.
But given topic A, we are more likely to see topic B than C. [CTM]

Assn documents are exchangeable (over time).
But time matters for what topic 'means' [DTM]

Assn no covariates
But topic prevalence and topic content are $f(X)$ [STM]

Lots of other ideas!

Lots of other ideas!

hierarchical LDA, pachinko allocation, nonparametric pachinko allocation,factorial LDA, gamma-poisson factorization, shared component topic models, dirichlet multinomial regression topic models, nested hierarchical dirichlet process topic model, focused topic model, inverse regression topic model, ideal point topic model, discrete innite logistic normal topic model multilingual topic model, markov topic model, relational topic model, syntactic topic model, supervised latent dirichlet allocation

Expressed Agenda Model (Grimmer, 2010)

Expressed Agenda Model (Grimmer, 2010)

Suppose each document is assigned to one topic.

Expressed Agenda Model (Grimmer, 2010)

Suppose each document is assigned to one topic.

Each author allocates a latent (to us) proportion of time to each topic.

Expressed Agenda Model (Grimmer, 2010)

Suppose each document is assigned to one topic.

Each author allocates a latent (to us) proportion of time to each topic.

$\rightarrow$ special case of LDA, and used for measuring way Senators 'express' themselves to constituents via press releases.

Notice that set of topics is same across Senators,

Expressed Agenda Model (Grimmer, 2010)

Suppose each document is assigned to one topic.

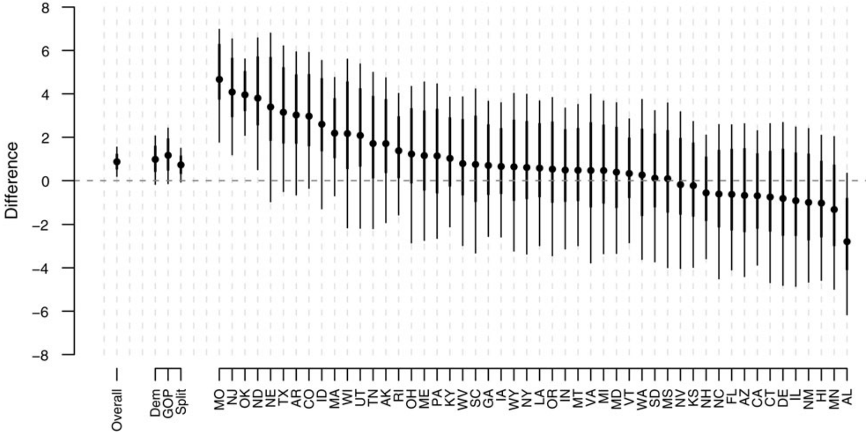
Each author allocates a latent (to us) proportion of time to each topic.

$\rightarrow$ special case of LDA, and used for measuring way Senators 'express' themselves to constituents via press releases.

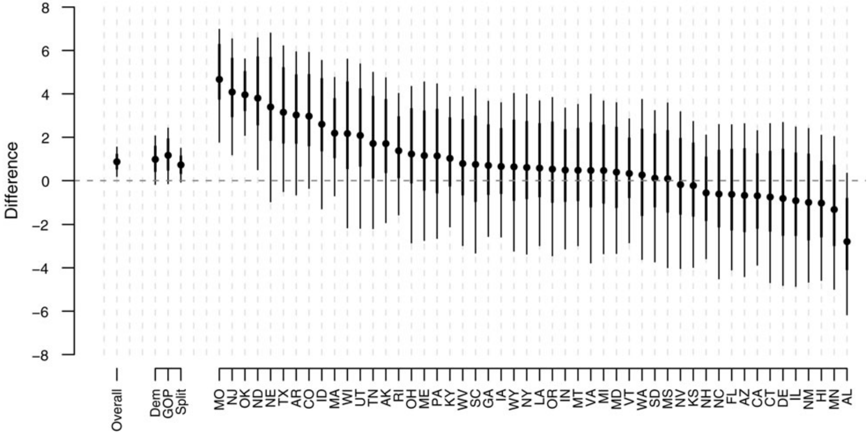
Notice that set of topics is same across Senators, but weights are allowed to vary across Senators.

Senators from same states have similar agendas

Senators from same states have similar agendas



Senators from same states have similar agendas



Senators from same states talk about more similar things than Senators from different states (generally).

Correlated Topic Model (Blei & Lafferty, 2007)

Correlated Topic Model (Blei & Lafferty, 2007)

If a news article talks about `finance` it's more likely to also talk about `law` than it is `baseball`.

Correlated Topic Model (Blei & Lafferty, 2007)

If a news article talks about `finance` it's more likely to also talk about `law` than it is `baseball`. But LDA doesn't allow this (consequence of Dirichlet prior on topic proportions).

Correlated Topic Model (Blei & Lafferty, 2007)

If a news article talks about `finance` it's more likely to also talk about `law` than it is `baseball`. But LDA doesn't allow this (consequence of Dirichlet prior on topic proportions).

The Correlated Topic Model allows for positive covariance between topics.

Correlated Topic Model (Blei & Lafferty, 2007)

If a news article talks about `finance` it's more likely to also talk about `law` than it is `baseball`. But LDA doesn't allow this (consequence of Dirichlet prior on topic proportions).

The Correlated Topic Model allows for positive covariance between topics. Does this by drawing topic proportions from a log normal.

Correlated Topic Model (Blei & Lafferty, 2007)

If a news article talks about `finance` it's more likely to also talk about `law` than it is `baseball`. But LDA doesn't allow this (consequence of Dirichlet prior on topic proportions).

The Correlated Topic Model allows for positive covariance between topics. Does this by drawing topic proportions from a log normal.

Shows improved model fit over LDA.

Correlated Topic Model (Blei & Lafferty, 2007)

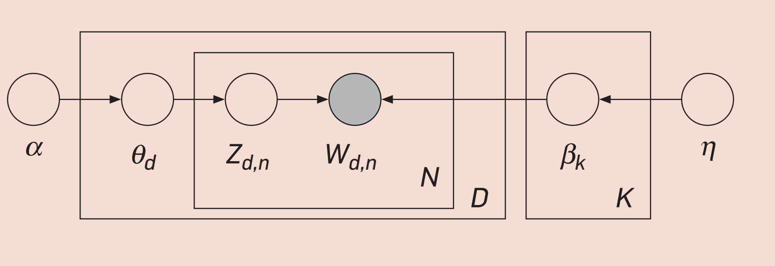
If a news article talks about `finance` it's more likely to also talk about `law` than it is `baseball`. But LDA doesn't allow this (consequence of Dirichlet prior on topic proportions).

The Correlated Topic Model allows for positive covariance between topics. Does this by drawing topic proportions from a log normal.

Shows improved model fit over LDA. BTW, note that STM (below) reduces to CTM if no covariates are specified.

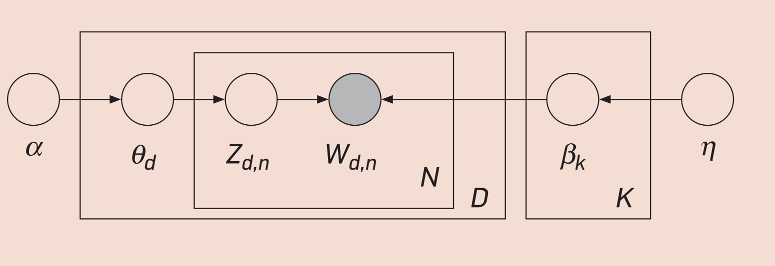
Dynamic Topic Model (Blei & Lafferty, 2006)

Recall LDA. . .



Dynamic Topic Model (Blei & Lafferty, 2006)

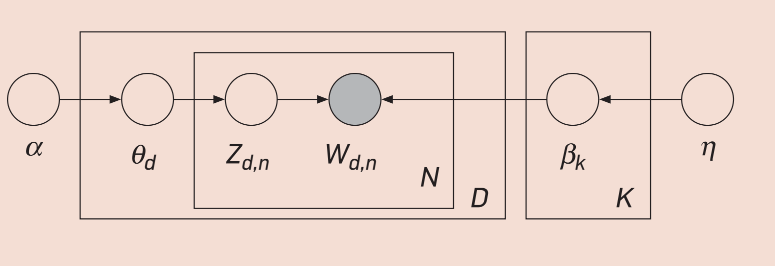
Recall LDA. . .



. . . there are multiple documents, but we don't care about their order.

Dynamic Topic Model (Blei & Lafferty, 2006)

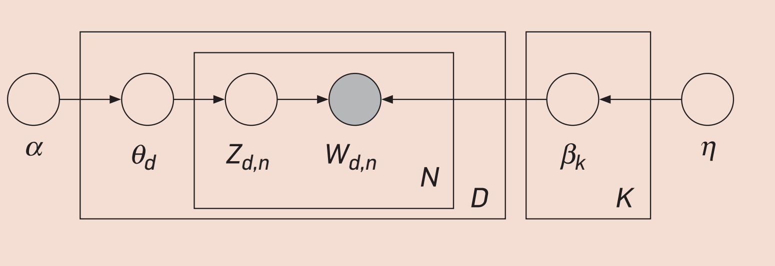
Recall LDA. . .



. . . there are multiple documents, but we don't care about their order. Our results are the 'same' regardless of how we reorder the documents and feed them to the model.

Dynamic Topic Model (Blei & Lafferty, 2006)

Recall LDA. . .

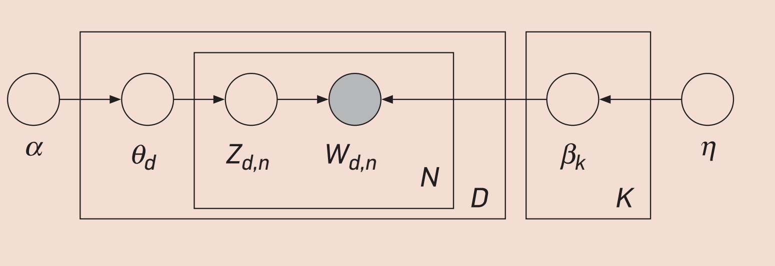


. . . there are multiple documents, but we don't care about their order. Our results are the 'same' regardless of how we reorder the documents and feed them to the model.

Dynamic Topic Model has a different model for each time period,

Dynamic Topic Model (Blei & Lafferty, 2006)

Recall LDA. . .



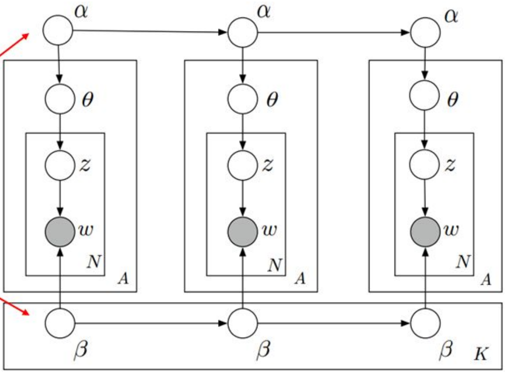
. . . there are multiple documents, but we don't care about their order. Our results are the 'same' regardless of how we reorder the documents and feed them to the model.

Dynamic Topic Model has a different model for each time period, with topics allowed to evolve over time. . .

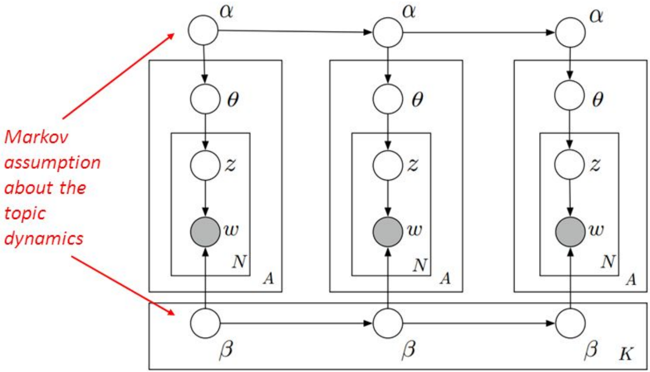
So. . .

So. . .

Markov assumption about the topic dynamics



So. . .



Now, mean parameters for the topic proportions (αs) and the what's in the topics (in terms of words, βs) are connected over time via a simple evolutionary process (West & Harrison, 1997).

How to Analyze Political Attention with Minimal Assumptions and Costs (Quinn et al, 2010)

How to Analyze Political Attention with Minimal Assumptions and Costs (Quinn et al, 2010)

What is the agenda of the Senate?

How to Analyze Political Attention with Minimal Assumptions and Costs (Quinn et al, 2010)

What is the agenda of the Senate? $\rightarrow$ dynamic topic model of over $100,000$ speeches.

How to Analyze Political Attention with Minimal Assumptions and Costs (Quinn et al, 2010)

What is the agenda of the Senate? $\rightarrow$ dynamic topic model of over $100,000$ speeches.

Assume slightly different evolution process (Dynamic Linear Model),

How to Analyze Political Attention with Minimal Assumptions and Costs (Quinn et al, 2010)

What is the agenda of the Senate? $\rightarrow$ dynamic topic model of over $100,000$ speeches.

Assume slightly different evolution process (Dynamic Linear Model), and only one topic per speech (like Grimmer).

How to Analyze Political Attention with Minimal Assumptions and Costs (Quinn et al, 2010)

What is the agenda of the Senate? $\rightarrow$ dynamic topic model of over $100,000$ speeches.

Assume slightly different evolution process (Dynamic Linear Model), and only one topic per speech (like Grimmer). Model is fit differently too.

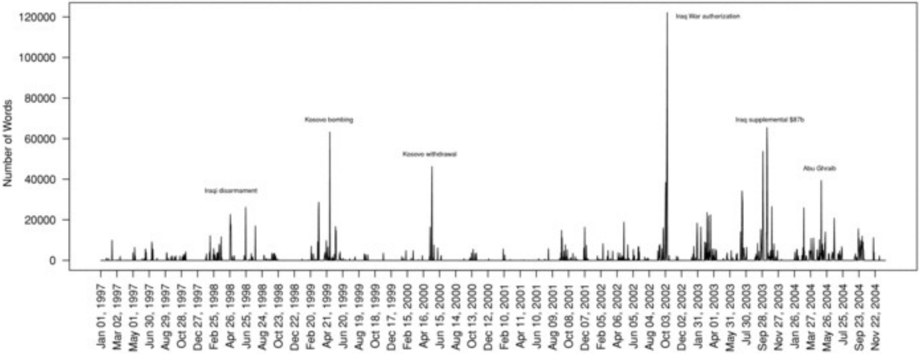
How to Analyze Political Attention with Minimal Assumptions and Costs (Quinn et al, 2010)

What is the agenda of the Senate? $\rightarrow$ dynamic topic model of over $100,000$ speeches.

Assume slightly different evolution process (Dynamic Linear Model), and only one topic per speech (like Grimmer). Model is fit differently too.

BTW, paper has a lot of validation!

Attention to Defense [Use of Force]



(b) *The Number of Words on the 'Defense [Use of Force]' Topic Per Day*

Structural Topic Model (Roberts et al.)

Structural Topic Model (Roberts et al.)

STM = LDA + contextual information.

Structural Topic Model (Roberts et al.)

STM = LDA + contextual information.

$\rightarrow$ topic prevalence varies by covariates

Structural Topic Model (Roberts et al.)

STM = LDA + contextual information.

$\rightarrow$ topic prevalence varies by covariates

e.g. women may report issues with depression more than men do.

Structural Topic Model (Roberts et al.)

STM = LDA + contextual information.

$\rightarrow$ topic prevalence varies by covariates

e.g. women may report issues with depression more than men do.

$\rightarrow$ topic content varies by covariates

Structural Topic Model (Roberts et al.)

STM = LDA + contextual information.

→ topic prevalence varies by covariates

e.g. women may report issues with depression more than men do.

→ topic content varies by covariates

e.g. young people ($X = 0$) may talk about depression differently to the way old people ($X = 1$) talk about it.

Structural Topic Model (Roberts et al.)

STM = LDA + contextual information.

$\rightarrow$ topic prevalence varies by covariates

e.g. women may report issues with depression more than men do.

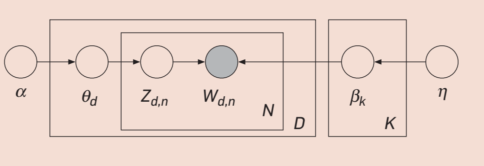
$\rightarrow$ topic content varies by covariates

e.g. young people ($X = 0$) may talk about depression differently to the way old people ($X = 1$) talk about it.

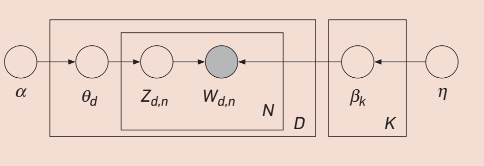
Including covariates allows for (a) more accurate estimation and (b) better interpretatability.

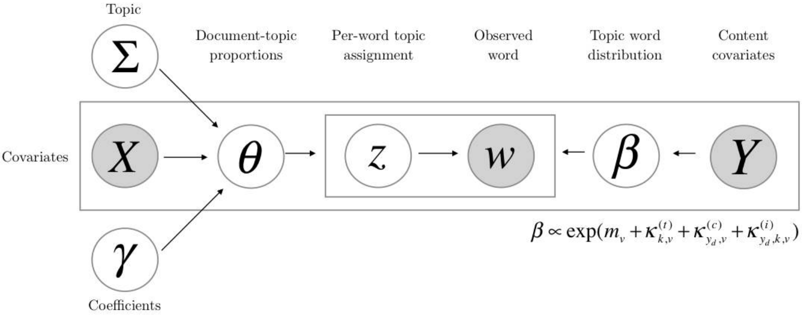
Compare: Plate Diagram

Compare: Plate Diagram



Compare: Plate Diagram





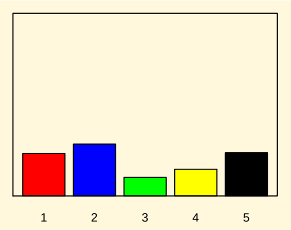
Compare: Per Document Topic Distribution (θ)

Compare: Per Document Topic Distribution (θ)

LDA: each document has some topic distribution.

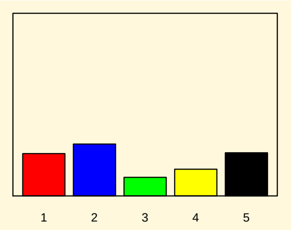
Compare: Per Document Topic Distribution (θ)

LDA: each document has some topic distribution.



Compare: Per Document Topic Distribution (θ)

LDA: each document has some topic distribution.



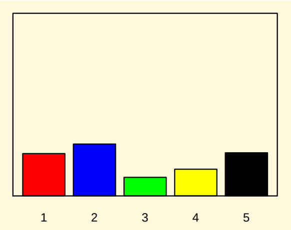
STM, that topic distribution ('prevalence') is a function of the document metadata.

Compare: Per Document Topic Distribution (θ)

LDA: each document has some topic distribution.

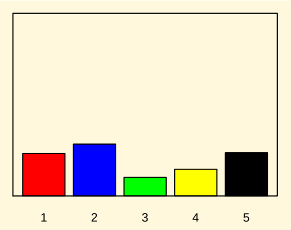
STM, that topic distribution ('prevalence') is a function of the document metadata.

e.g. perhaps male author ($X = 0$) documents have different topics relative to female ($X = 1$) author docs.



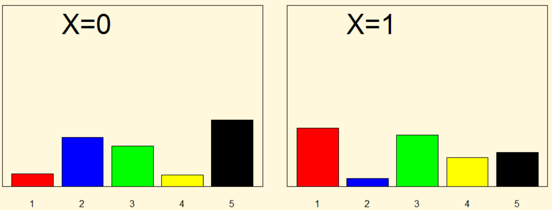
Compare: Per Document Topic Distribution (θ)

LDA: each document has some topic distribution.



STM, that topic distribution ('prevalence') is a function of the document metadata.

e.g. perhaps male author ($X = 0$) documents have different topics relative to female ($X = 1$) author docs.



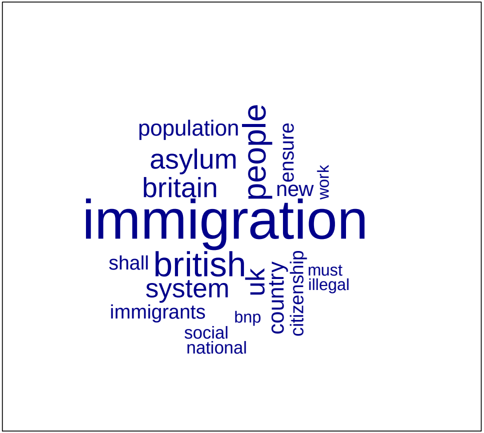
Compare: Per Topic Word Distribution (β)

Compare: Per Topic Word Distribution (β)

LDA: topic ('immigration') has a given distribution over words.

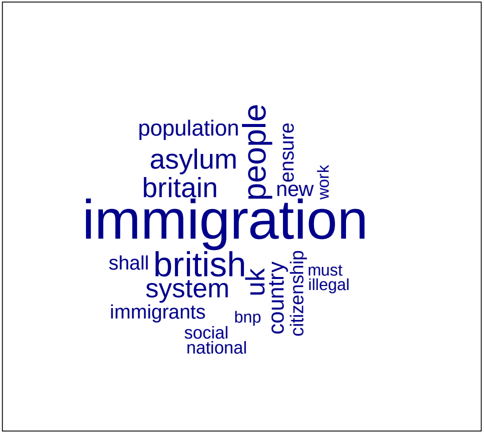
Compare: Per Topic Word Distribution (β)

LDA: topic ('immigration') has a given distribution over words.



Compare: Per Topic Word Distribution (β)

LDA: topic ('immigration') has a given distribution over words.



STM: that word distribution ('content') is a function of the document metadata.

STM: that word distribution ('content') is a function of the document metadata.

e.g. perhaps right parties ($Y = 0$) talk about a given topic differently to left ($Y = 1$) parties.

STM: that word distribution ('content') is a function of the document metadata.

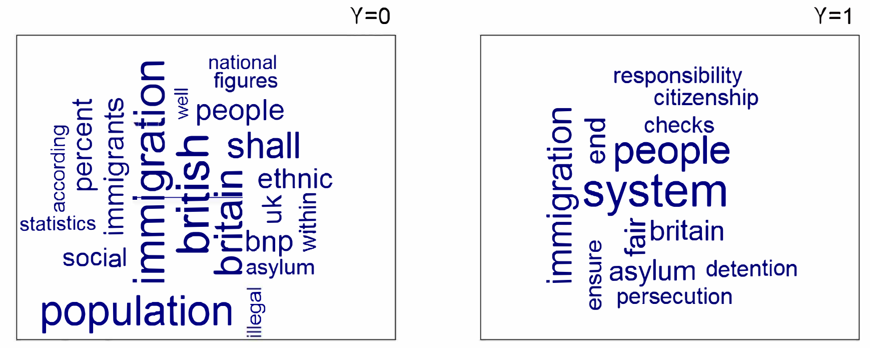
e.g. perhaps right parties ($Y = 0$) talk about a given topic differently to left ($Y = 1$) parties.

In practice, `content` needs to a single discrete variable.

STM: that word distribution ('content') is a function of the document metadata.

e.g. perhaps right parties ($Y = 0$) talk about a given topic differently to left ($Y = 1$) parties.

In practice, `content` needs to a single discrete variable.



Partner Exercise

Partner Exercise

1. In the dynamic topic model, any changes overtime are 'smooth'. Give an example of a time series problem where you want topics to change over time, but not in a smooth way.

Partner Exercise

1. In the dynamic topic model, any changes overtime are ‘smooth’. Give an example of a time series problem where you want topics to change over time, but not in a smooth way.

2. In social science, we might be interested in the dynamic topic model because both its αs and the βs are allowed to change. What types of substantive problems do changes in these two different parameters help us model?

Partner Exercise

1. In the dynamic topic model, any changes overtime are 'smooth'. Give an example of a time series problem where you want topics to change over time, but not in a smooth way.
2. In social science, we might be interested in the dynamic topic model because both its αs and the βs are allowed to change. What types of substantive problems do changes in these two different parameters help us model?
3. Are the 'effects' in the STM causal? If not, why not, and can you give a scenario where they would be?

Embeddings

Distributional Hypothesis

Distributional Hypothesis

> *"A word is characterized by the company it keeps."*
>
> -Firth, 1957

Distributional Hypothesis

> *"A word is characterized by the company it keeps."*
>
> -Firth, 1957

So: words that are used in the same contexts tend to be similar in their meaning.

Distributional Hypothesis

"A word is characterized by the company it keeps."

-Firth, 1957

So: words that are used in the same contexts tend to be similar in their meaning. How can we use this insight to understand what words mean?

Distributional Hypothesis

> *"A word is characterized by the company it keeps."*
>
> -Firth, 1957

So: words that are used in the same contexts tend to be similar in their meaning. How can we use this insight to understand what words mean?

If the word `liberal` shows up 'near' `privatization` in one country but 'near' `socialist` in another, we can learn something about those polities.

Distributional Hypothesis

> *"A word is characterized by the company it keeps."*
>
> -Firth, 1957

So: words that are used in the same contexts tend to be similar in their meaning. How can we use this insight to understand what words mean?

If the word `liberal` shows up 'near' `privatization` in one country but 'near' `socialist` in another, we can learn something about those polities.

We can systematically learn about analogies and similarities.

Approaches

Approaches

1. 'count-based' methods:

Approaches

1. 'count-based' methods: look at how often words co-occur with neighbors.

Approaches

1 'count-based' methods: look at how often words co-occur with neighbors.

e.g. Latent Semantic Analysis: it groups documents using *similar words* and words occurring in *similar documents*.

Approaches

1 'count-based' methods: look at how often words co-occur with neighbors.

e.g. Latent Semantic Analysis: it groups documents using *similar words* and words occurring in *similar documents*. There is some rank reduction to discover latent components.

Approaches

1 'count-based' methods: look at how often words co-occur with neighbors.

e.g. Latent Semantic Analysis: it groups documents using *similar words* and words occurring in *similar documents*. There is some rank reduction to discover latent components.

2 'predictive' methods:

Approaches

1 'count-based' methods: look at how often words co-occur with neighbors.

e.g. Latent Semantic Analysis: it groups documents using *similar words* and words occurring in *similar documents*. There is some rank reduction to discover latent components.

2 'predictive' methods: try to predict a word from knowing its neighbors.

Approaches

1 'count-based' methods: look at how often words co-occur with neighbors.

e.g. Latent Semantic Analysis: it groups documents using *similar words* and words occurring in *similar documents*. There is some rank reduction to discover latent components.

2 'predictive' methods: try to predict a word from knowing its neighbors. To do this, we *learn* an embedding vector that captures meaning of a given word in our corpus a numerical way.

Approaches

1. 'count-based' methods: look at how often words co-occur with neighbors.

e.g. Latent Semantic Analysis: it groups documents using *similar words* and words occurring in *similar documents*. There is some rank reduction to discover latent components.

2. 'predictive' methods: try to predict a word from knowing its neighbors. To do this, we *learn* an embedding vector that captures meaning of a given word in our corpus a numerical way.

e.g. Word2Vec:

Approaches

1 'count-based' methods: look at how often words co-occur with neighbors.

e.g. Latent Semantic Analysis: it groups documents using *similar words* and words occurring in *similar documents*. There is some rank reduction to discover latent components.

2 'predictive' methods: try to predict a word from knowing its neighbors. To do this, we *learn* an embedding vector that captures meaning of a given word in our corpus a numerical way.

e.g. Word2Vec: a powerful way to create the word vectors. Comes in two types/models, Continuous Bag of Words (CBOW) and Skip-Gram.

What's the Difference?

What's the Difference?

1. Continuous Bag of Words predicts *target* words from source *context* words.

What's the Difference?

1 Continuous Bag of Words predicts *target* words from source *context* words.

So, predicts ‘`dog`’ from ‘`the quick brown fox jumped over [ ]`’

What's the Difference?

1 Continuous Bag of Words predicts *target* words from source *context* words.

So, predicts '`dog`' from '`the quick brown fox jumped over [ ]`'

$\rightarrow$ useful in small datasets.

What's the Difference?

1 Continuous Bag of Words predicts *target* words from source *context* words.

So, predicts '`dog`' from '`the quick brown fox jumped over [ ]`'

→ useful in small datasets.

2 Skip-gram predicts source *context* words from *target* words

What's the Difference?

1. Continuous Bag of Words predicts *target* words from source *context* words.

   So, predicts '`dog`' from '`the quick brown fox jumped over [ ]`'

   $\rightarrow$ useful in small datasets.

2. Skip-gram predicts source *context* words from *target* words

   So, predicts '`the quick brown fox jumped over [ ]`' from '`dog`'.

What's the Difference?

1 Continuous Bag of Words predicts *target* words from source *context* words.

So, predicts '`dog`' from '`the quick brown fox jumped over [ ]`'

$\rightarrow$ useful in small datasets.

2 Skip-gram predicts source *context* words from *target* words

So, predicts '`the quick brown fox jumped over [ ]`' from '`dog`'.

$\rightarrow$ useful in large datasets.

Continuous Bag of Words (CBOW): A Primer

Thinking About ‘Context’

Thinking About 'Context'

Thinking About 'Context'

Lots of definitions, but common one is co-occurrence within a window.

Thinking About 'Context'

Lots of definitions, but common one is co-occurrence within a window.

Consider "`the quick brown fox jumped over the lazy dog`"...

Thinking About 'Context'

Lots of definitions, but common one is co-occurrence within a window.

Consider "`the quick brown fox jumped over the lazy dog`"...

the	quick	brown	fox	jumped	over	the	lazy	dog

Set the context window to 2.

Thinking About 'Context'

Lots of definitions, but common one is co-occurrence within a window.

Consider "`the quick brown fox jumped over the lazy dog`"...

the	quick	brown	fox	jumped	over	the	lazy	dog

Set the context window to 2.

Given that context window,

Thinking About 'Context'

Lots of definitions, but common one is co-occurrence within a window.

Consider "`the quick brown fox jumped over the lazy dog`"...

the	quick	brown	fox	jumped	over	the	lazy	dog

Set the context window to 2.

Given that context window, what words co-occur with '`fox`'?

Thinking About 'Context'

Lots of definitions, but common one is co-occurrence within a window.

Consider "`the quick brown fox jumped over the lazy dog`"...

the	quick	brown	fox	jumped	over	the	lazy	dog

Set the context window to 2.

Given that context window, what words co-occur with '`fox`'?

Well. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

Well. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

Well. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

And for ‘`over`’. . .

Well. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

And for ‘`over`’. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

Well. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

And for ‘over’. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

So here,

Well. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

And for '`over`'. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

So here, '`quick`' co-occurred with '`fox`',

Well. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

And for ‘`over`’. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

So here, ‘`quick`’ co-occurred with ‘`fox`’, but not with ‘`dog`’.

Well. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

And for 'over'. . .

the	quick	brown	fox	jumped	over	the	lazy	dog

So here, '`quick`' co-occurred with '`fox`', but not with '`dog`'.

Continuous Bag of Words (CBOW)

Continuous Bag of Words (CBOW)

We are trying to predicting the *target* given the context.

Continuous Bag of Words (CBOW)

We are trying to predicting the *target* given the context.

To keep it simple, consider predicting one target word, given one context word ("single context word CBOW")

Continuous Bag of Words (CBOW)

We are trying to predicting the *target* given the context.

To keep it simple, consider predicting one target word, given one context word ("single context word CBOW")

So, if our document was "`the quick brown fox jumped over the lazy dog`",

Continuous Bag of Words (CBOW)

We are trying to predicting the *target* given the context.

To keep it simple, consider predicting one target word, given one context word ("single context word CBOW")

So, if our document was "`the quick brown fox jumped over the lazy dog`", we start by using the as an input to predict quick as an output...

Continuous Bag of Words (CBOW)

We are trying to predicting the *target* given the context.

To keep it simple, consider predicting one target word, given one context word ("single context word CBOW")

So, if our document was "`the quick brown fox jumped over the lazy dog`", we start by using the as an input to predict quick as an output...

Then, `the` $\leftarrow$ `quick`

Continuous Bag of Words (CBOW)

We are trying to predicting the *target* given the context.

To keep it simple, consider predicting one target word, given one context word ("single context word CBOW")

So, if our document was "`the quick brown fox jumped over the lazy dog`", we start by using the as an input to predict quick as an output...

Then, `the` $\leftarrow$ `quick` and `quick` $\rightarrow$ `brown`

Continuous Bag of Words (CBOW)

We are trying to predicting the *target* given the context.

To keep it simple, consider predicting one target word, given one context word ("single context word CBOW")

So, if our document was "`the quick brown fox jumped over the lazy dog`", we start by using `the` as an input to predict `quick` as an output...

Then, `the` $\leftarrow$ `quick` and `quick` $\rightarrow$ `brown` We move on to `brown`.

Continuous Bag of Words (CBOW)

We are trying to predicting the *target* given the context.

To keep it simple, consider predicting one target word, given one context word ("single context word CBOW")

So, if our document was "`the quick brown fox jumped over the lazy dog`", we start by using the as an input to predict quick as an output...

Then, `the` $\leftarrow$ `quick` and `quick` $\rightarrow$ `brown` We move on to `brown`.

Then, `quick` $\leftarrow$ `brown` $\rightarrow$ `fox`, `brown` $\leftarrow$ `fox` $\rightarrow$ `jumped`, etc

Continuous Bag of Words (CBOW)

We are trying to predicting the *target* given the context.

To keep it simple, consider predicting one target word, given one context word ("single context word CBOW")

So, if our document was "`the quick brown fox jumped over the lazy dog`", we start by using the as an input to predict quick as an output...

Then, `the` ← `quick` and `quick` → `brown` We move on to `brown`.

Then, `quick` ← `brown` → `fox`, `brown` ← `fox` → `jumped`, etc

Until `lazy` → `dog`,

Continuous Bag of Words (CBOW)

We are trying to predicting the *target* given the context.

To keep it simple, consider predicting one target word, given one context word ("single context word CBOW")

So, if our document was "`the quick brown fox jumped over the lazy dog`", we start by using the as an input to predict quick as an output...

Then, `the` ← `quick` and `quick` → `brown` We move on to `brown`.

Then, `quick` ← `brown` → `fox`, `brown` ← `fox` → `jumped`, etc

Until `lazy` → `dog`, and ending with `lazy` ← `dog` (because we have no other context for this one)

The data

The data

Input	Output	Ob #	the	quick	brown	fox	jumped	over	the	lazy	dog

The data

Input	Output	Ob #	the	quick	brown	fox	jumped	over	the	lazy	dog
the											

The data

Input	Output	Ob #	the	quick	brown	fox	jumped	over	the	lazy	dog
the	quick	1	1	0	0	0	0	0	0	0	0

The data

Input	Output	Ob #	the	quick	brown	fox	jumped	over	the	lazy	dog
the	quick	1	1	0	0	0	0	0	0	0	0
quick	the	2	0	1	0	0	0	0	0	0	0

The data

Input	Output	Ob #	the	quick	brown	fox	jumped	over	the	lazy	dog
the	quick	1	1	0	0	0	0	0	0	0	0
quick	the	2	0	1	0	0	0	0	0	0	0
quick	brown	3	0	1	0	0	0	0	0	0	0

The data

Input	Output	Ob #	the	quick	brown	fox	jumped	over	the	lazy	dog
the	quick	1	1	0	0	0	0	0	0	0	0
quick	the	2	0	1	0	0	0	0	0	0	0
quick	brown	3	0	1	0	0	0	0	0	0	0
brown	quick	4	0	0	1	0	0	0	0	0	0
brown	fox	5	0	0	1	0	0	0	0	0	0
fox	brown	6	0	0	0	1	0	0	0	0	0
fox	jumped	7	0	0	0	1	0	0	0	0	0
jumped	fox	8	0	0	0	0	1	0	0	0	0
jumped	over	9	0	0	0	0	1	0	0	0	0
over	jumped	10	0	0	0	0	0	1	0	0	0
over	the	11	0	0	0	0	0	1	0	0	0
the	over	12	0	0	0	0	0	0	1	0	0
the	lazy	13	0	0	0	0	0	0	1	0	0
lazy	the	14	0	0	0	0	0	0	0	1	0
lazy	dog	15	0	0	0	0	0	0	0	1	0
dog	lazy	16	0	0	0	0	0	0	0	0	1

The data

Input	Output	Ob #	the	quick	brown	fox	jumped	over	the	lazy	dog
the	quick	1	1	0	0	0	0	0	0	0	0
quick	the	2	0	1	0	0	0	0	0	0	0
quick	brown	3	0	1	0	0	0	0	0	0	0
brown	quick	4	0	0	1	0	0	0	0	0	0
brown	fox	5	0	0	1	0	0	0	0	0	0
fox	brown	6	0	0	0	1	0	0	0	0	0
fox	jumped	7	0	0	0	1	0	0	0	0	0
jumped	fox	8	0	0	0	0	1	0	0	0	0
jumped	over	9	0	0	0	0	1	0	0	0	0
over	jumped	10	0	0	0	0	0	1	0	0	0
over	the	11	0	0	0	0	0	1	0	0	0
the	over	12	0	0	0	0	0	0	1	0	0
the	lazy	13	0	0	0	0	0	0	1	0	0
lazy	the	14	0	0	0	0	0	0	0	1	0
lazy	dog	15	0	0	0	0	0	0	0	1	0
dog	lazy	16	0	0	0	0	0	0	0	0	1

‘one-hot encoding’

Using the data

Using the data

Each row of this data is a 'one hot encoding' representation of the document.

Using the data

Each row of this data is a 'one hot encoding' representation of the document. Is simply a 'dummy variable' representation.

Using the data

Each row of this data is a 'one hot encoding' representation of the document. Is simply a 'dummy variable' representation.

e.g. observation 6 is [0, 0, 0, 1, 0, 0, 0, 0, 0] since it involves the use of `fox` at that point in the document.

Using the data

Each row of this data is a ‘one hot encoding’ representation of the document. Is simply a ‘dummy variable’ representation.

e.g. observation 6 is [0, 0, 0, 1, 0, 0, 0, 0, 0] since it involves the use of `fox` at that point in the document.

The binary matrix on the right is the input to a neural network.

Using the data

Each row of this data is a 'one hot encoding' representation of the document. Is simply a 'dummy variable' representation.

e.g. observation 6 is [0, 0, 0, 1, 0, 0, 0, 0, 0] since it involves the use of `fox` at that point in the document.

The binary matrix on the right is the input to a neural network. Ultimately this will produce a series of weights which are the word vector representation of the word.

Using the data

Each row of this data is a 'one hot encoding' representation of the document. Is simply a 'dummy variable' representation.

e.g. observation 6 is [0, 0, 0, 1, 0, 0, 0, 0, 0] since it involves the use of `fox` at that point in the document.

The binary matrix on the right is the input to a neural network. Ultimately this will produce a series of weights which are the word vector representation of the word.

In practice, neural nets used for the most common models are very simple linear ones, but it doesn't hurt to give a (hand-waving) overview...

Neural Nets: A Primer

The Basics

The Basics

We will extract/create linear combinations of inputs (the Xs) as 'new' features...

The Basics

We will extract/create linear combinations of inputs (the Xs) as 'new' features...

...and model Y as a non-linear function of those new features.

The Basics

We will extract/create linear combinations of inputs (the Xs) as 'new' features. . .

. . . and model Y as a non-linear function of those new features.

They are known as (artificial) neural nets (or neural networks) as they were inspired by the structure of human brains,

The Basics

We will extract/create linear combinations of inputs (the Xs) as 'new' features...

...and model Y as a non-linear function of those new features.

They are known as (artificial) neural nets (or neural networks) as they were inspired by the structure of human brains, and attempt to learn in way brains do.

The Basics

We will extract/create linear combinations of inputs (the Xs) as 'new' features...

...and model Y as a non-linear function of those new features.

They are known as (artificial) neural nets (or neural networks) as they were inspired by the structure of human brains, and attempt to learn in way brains do.

They can be made arbitrarily complex,

We will extract/create linear combinations of inputs (the Xs) as ‘new’ features. . .

. . . and model Y as a non-linear function of those new features.

They are known as (artificial) neural nets (or neural networks) as they were inspired by the structure of human brains, and attempt to learn in way brains do.

They can be made arbitrarily complex, such that they can capture almost any relationship between Y and the Xs. . . but this comes at a cost:

The Basics

We will extract/create linear combinations of inputs (the Xs) as 'new' features...

...and model Y as a non-linear function of those new features.

They are known as (artificial) neural nets (or neural networks) as they were inspired by the structure of human brains, and attempt to learn in way brains do.

They can be made arbitrarily complex, such that they can capture almost any relationship between Y and the Xs... but this comes at a cost:

(a) computational

The Basics

We will extract/create linear combinations of inputs (the Xs) as 'new' features...

...and model Y as a non-linear function of those new features.

They are known as (artificial) neural nets (or neural networks) as they were inspired by the structure of human brains, and attempt to learn in way brains do.

They can be made arbitrarily complex, such that they can capture almost any relationship between Y and the Xs...but this comes at a cost:

(a) computational—fitting the model to the data is hard computationally

The Basics

We will extract/create linear combinations of inputs (the Xs) as 'new' features...

...and model Y as a non-linear function of those new features.

They are known as (artificial) neural nets (or neural networks) as they were inspired by the structure of human brains, and attempt to learn in way brains do.

They can be made arbitrarily complex, such that they can capture almost any relationship between Y and the Xs...but this comes at a cost:

(a) computational—fitting the model to the data is hard computationally
(b) interpretation

The Basics

We will extract/create linear combinations of inputs (the Xs) as 'new' features. . .

. . . and model Y as a non-linear function of those new features.

They are known as (artificial) neural nets (or neural networks) as they were inspired by the structure of human brains, and attempt to learn in way brains do.

They can be made arbitrarily complex, such that they can capture almost any relationship between Y and the Xs. . . but this comes at a cost:

(a) computational—fitting the model to the data is hard computationally
(b) interpretation—understanding what the models says about the DGP is difficult.

The Set Up

The Set Up

We have p X-variables,

The Set Up

We have p X-variables, which could be counts in a DTM, factors, etc.

The Set Up

We have p X-variables, which could be counts in a DTM, factors, etc.
We want to predict/explain Y.

The Set Up

We have p X-variables, which could be counts in a DTM, factors, etc. We want to predict/explain Y. As usual, we'll make $\mathbf{y} = f(\mathbf{x}) + \epsilon$.

The Set Up

We have p X-variables, which could be counts in a DTM, factors, etc. We want to predict/explain Y. As usual, we'll make $\mathbf{y} = f(\mathbf{x}) + \epsilon$.

In a linear regression we would set this up as

The Set Up

We have p X-variables, which could be counts in a DTM, factors, etc. We want to predict/explain Y. As usual, we'll make $\mathbf{y} = f(\mathbf{x}) + \epsilon$.

In a linear regression we would set this up as

$$\mathbf{y} = \boldsymbol{\beta}'\mathbf{x} + \epsilon$$

The Set Up

We have p X-variables, which could be counts in a DTM, factors, etc. We want to predict/explain Y. As usual, we'll make $\mathbf{y} = f(\mathbf{x}) + \epsilon$.

In a linear regression we would set this up as

$$\mathbf{y} = \boldsymbol{\beta}'\mathbf{x} + \epsilon$$

In a projection pursuit regression (Freidman & Stuetzle), we make $f(\mathbf{x})$ a sum of non-linear (non-parametric) functions of (unidimensional projections) of $\mathbf{x}$. That is,

$$\mathbf{y} = \sum_{m=1}^{M} g_m(\beta_m'\mathbf{x}) + \epsilon$$

The Set Up

We have p X-variables, which could be counts in a DTM, factors, etc. We want to predict/explain Y. As usual, we'll make $\mathbf{y} = f(\mathbf{x}) + \epsilon$.

In a linear regression we would set this up as

$$\mathbf{y} = \boldsymbol{\beta}'\mathbf{x} + \epsilon$$

In a projection pursuit regression (Freidman & Stuetzle), we make $f(\mathbf{x})$ a sum of non-linear (non-parametric) functions of (unidimensional projections) of $\mathbf{x}$. That is,

$$\mathbf{y} = \sum_{m=1}^{M} g_m(\beta_m'\mathbf{x}) + \epsilon$$

Projection Pursuit Regression

Projection Pursuit Regression

Key component is:

Projection Pursuit Regression

Key component is:

$$\sum_{m=1}^{M} g_m(\beta_m' \mathbf{x})$$

This is an additive model, but not in the original Xs

Projection Pursuit Regression

Key component is:

$$\sum_{m=1}^{M} g_m(\beta_m' \mathbf{x})$$

This is an additive model, but not in the original Xs It's additive in the new/derived features: the $\beta_j' x$s.

Projection Pursuit Regression

Key component is:

$$\sum_{m=1}^{M} g_m(\beta'_m \mathbf{x})$$

This is an additive model, but not in the original Xs It's additive in the new/derived features: the $\beta'_j x$s.

Both the g_m and the β_js are unknown, and will be estimated (to optimize model fit).

Projection Pursuit Regression

Key component is:

$$\sum_{m=1}^{M} g_m(\beta'_m \mathbf{x})$$

This is an additive model, but not in the original Xs It's additive in the new/derived features: the $\beta'_j x$s.

Both the g_m and the β_js are unknown, and will be estimated (to optimize model fit).

To keep things simple, let's suppose that we have two Xs and that there will be one function $g \ldots$

Example

Example

Suppose the 'true' DGP for Y is

$$Y = \frac{1}{[1 + \exp(-5(\frac{X_1 + X_2}{\sqrt{2}} - 0.5))]}.$$

Example

Suppose the 'true' DGP for Y is

$$Y = \frac{1}{[1 + \exp(-5(\frac{X_1+X_2}{\sqrt{2}} - 0.5))]}.$$

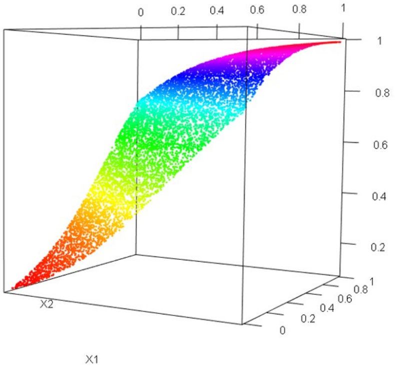
A linear function of Xs will not do a good job of approximating this!

Example

Suppose the 'true' DGP for Y is

$$Y = \frac{1}{[1 + \exp(-5(\frac{X_1 + X_2}{\sqrt{2}} - 0.5))]}.$$

A linear function of Xs will not do a good job of approximating this!



So. . .

So...

We want need a β_j' that can be 'multiplied' by the Xs in a linear way...

So...

We want need a β_j' that can be 'multiplied' by the Xs in a linear way...

Here, $\frac{X_1+X_2}{\sqrt{2}}$ is our $\beta_j' x$.

So. . .

We want need a β_j' that can be 'multiplied' by the Xs in a linear way. . .

Here, $\frac{X_1+X_2}{\sqrt{2}}$ is our $\beta_j'x$. Which implies that β_j is $(1/\sqrt{2}, 1/\sqrt{2})$.

So. . .

We want need a β_j' that can be 'multiplied' by the Xs in a linear way. . .

Here, $\frac{X_1+X_2}{\sqrt{2}}$ is our $\beta_j' x$. Which implies that β_j is $(1/\sqrt{2}, 1/\sqrt{2})$.

Then the dot product $\beta_j' x$

So. . .

We want need a β_j' that can be 'multiplied' by the Xs in a linear way. . .

Here, $\frac{X_1+X_2}{\sqrt{2}}$ is our $\beta_j' x$. Which implies that β_j is $(1/\sqrt{2}, 1/\sqrt{2})$.

Then the dot product $\beta_j' x$—our *derived* features—enters a non-linear function, g which is

So...

We want need a β_j' that can be 'multiplied' by the Xs in a linear way...

Here, $\frac{X_1+X_2}{\sqrt{2}}$ is our $\beta_j'x$. Which implies that β_j is $(1/\sqrt{2}, 1/\sqrt{2})$.

Then the dot product $\beta_j'x$—our *derived* features—enters a non-linear function, g which is

$$\frac{1}{[1+\exp(-5(\beta_j'x - 0.5))]}$$

Turns out that this idea—non-linear functions of linear combinations of Xs—allows essentially infinite flexibility:

So...

We want need a β_j' that can be 'multiplied' by the Xs in a linear way...

Here, $\frac{X_1+X_2}{\sqrt{2}}$ is our $\beta_j' x$. Which implies that β_j is $(1/\sqrt{2}, 1/\sqrt{2})$.

Then the dot product $\beta_j' x$—our *derived* features—enters a non-linear function, g which is

$$\frac{1}{[1+\exp(-5(\beta_j' x - 0.5))]}$$

Turns out that this idea—non-linear functions of linear combinations of Xs—allows essentially infinite flexibility: we can approximate anything we like (some smoothness restrictions in practice).

Example II

Example II

Suppose the 'true' DGP for Y is

$$Y = (\beta_j' x + 0.1) \sin\left(\frac{1}{\frac{\beta_j' x}{3} + 0.1}\right)$$

Example II

Suppose the 'true' DGP for Y is

$$Y = (\beta_j' x + 0.1) \sin \left(\frac{1}{\frac{\beta_j' x}{3} + 0.1} \right)$$

A linear function of Xs will not do a good job of approximating this!

Example II

Suppose the 'true' DGP for Y is

$$Y = (\beta_j' x + 0.1) \sin \left(\frac{1}{\frac{\beta_j' x}{3} + 0.1} \right)$$

A linear function of Xs will not do a good job of approximating this!

Note that X_2 has no effect on Y in this case.

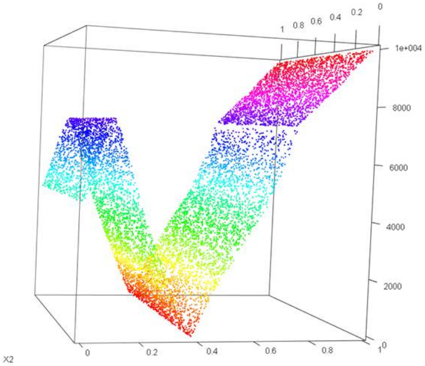
Example II

Suppose the 'true' DGP for Y is

$$Y = (\beta_j' x + 0.1) \sin\left(\frac{1}{\frac{\beta_j' x}{3} + 0.1}\right)$$

A linear function of Xs will not do a good job of approximating this!

Note that X_2 has no effect on Y in this case. So, β_j' is $(1, 0)$.



Neural Nets

Neural Nets

Neural nets are a type of non-linear statistical models—which makes them like the projection pursuit regression.

Neural Nets

Neural nets are a type of non-linear statistical models—which makes them like the projection pursuit regression.

Can be used for regression (one Y) or classification,

Neural Nets

Neural nets are a type of non-linear statistical models—which makes them like the projection pursuit regression.

Can be used for regression (one Y) or classification, in which case we need to predict $Y_1, \ldots, Y_K$ class of a given observation.

Neural Nets

Neural nets are a type of non-linear statistical models—which makes them like the projection pursuit regression.

Can be used for regression (one Y) or classification, in which case we need to predict $Y_1, \ldots, Y_K$ class of a given observation.

Infinite variety:

Neural Nets

Neural nets are a type of non-linear statistical models—which makes them like the projection pursuit regression.

Can be used for regression (one Y) or classification, in which case we need to predict $Y_1, \ldots, Y_K$ class of a given observation.

Infinite variety: (most?) basic is single layer perceptron neural net (AKA "single hidden layer back-propogation network").

Neural Nets

Neural nets are a type of non-linear statistical models—which makes them like the projection pursuit regression.

Can be used for regression (one Y) or classification, in which case we need to predict $Y_1, \ldots, Y_K$ class of a given observation.

Infinite variety: (most?) basic is single layer perceptron neural net (AKA "single hidden layer back-propogation network").

Basic idea remains: linear combinations of the Xs, which are pushed through non-linear functions to predict class membership (Y_k).

Neural Nets

Neural nets are a type of non-linear statistical models—which makes them like the projection pursuit regression.

Can be used for regression (one Y) or classification, in which case we need to predict $Y_1, \ldots, Y_K$ class of a given observation.

Infinite variety: (most?) basic is single layer perceptron neural net (AKA "single hidden layer back-propogation network").

Basic idea remains: linear combinations of the Xs, which are pushed through non-linear functions to predict class membership (Y_k).

Specifically. . .

Specifically...

Create some linear combinations of the Xs (βx).

Specifically. . .

Create some linear combinations of the Xs (βx).

Create some non-linear functions of those linear combinations (something like $\frac{1}{1+\exp(s\beta x)}$) (parametric; cf. PPR). Call those functions Z_m, and are the derived features.

Specifically...

Create some linear combinations of the Xs (βx).

Create some non-linear functions of those linear combinations (something like $\frac{1}{1+\exp(s\beta x)}$) (parametric; cf. PPR). Call those functions Z_m, and are the derived features.

Create linear functions of the Z to predict Y_k,

Specifically...

Create some linear combinations of the Xs (βx).

Create some non-linear functions of those linear combinations (something like $\frac{1}{1+\exp(s\beta x)}$) (parametric; cf. PPR). Call those functions Z_m, and are the derived features.

Create linear functions of the Z to predict Y_k, of the form $T_k = \alpha_0 + \alpha_k Z$

Specifically. . .

Create some linear combinations of the Xs (βx).

Create some non-linear functions of those linear combinations (something like $\frac{1}{1+\exp(s\beta x)}$) (parametric; cf. PPR). Call those functions Z_m, and are the derived features.

Create linear functions of the Z to predict Y_k, of the form $T_k = \alpha_0 + \alpha_k Z$

Transform those outputs into something useful for a class prediction problem,

Specifically...

Create some linear combinations of the Xs (βx).

Create some non-linear functions of those linear combinations (something like $\frac{1}{1+\exp(s\beta x)}$) (parametric; cf. PPR). Call those functions Z_m, and are the derived features.

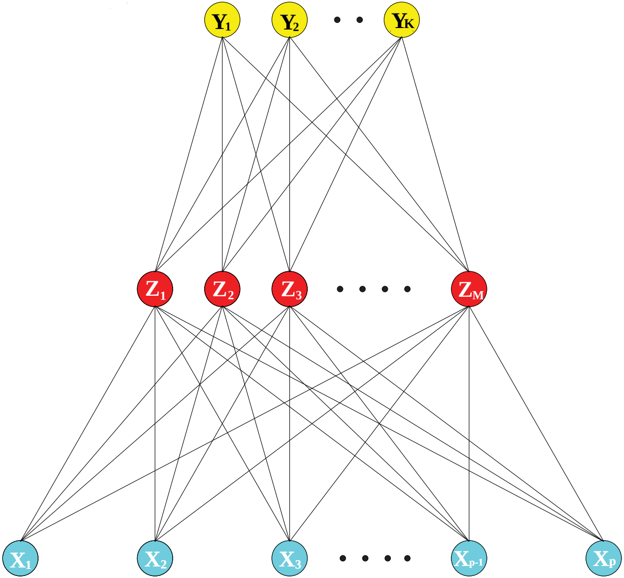
Create linear functions of the Z to predict Y_k, of the form $T_k = \alpha_0 + \alpha_k Z$

Transform those outputs into something useful for a class prediction problem, like the multinomial logit transform (called 'softmax' in this literature):

$$g_k(T_k) = \frac{\exp(T_k)}{\sum_{t=1}^{K} \exp(T_k)}$$

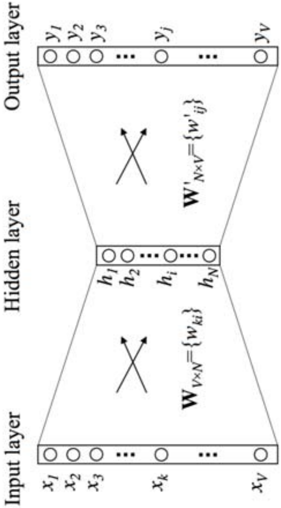
Schematic

Schematic

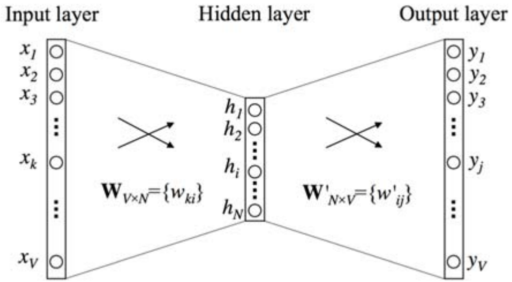


Back to Embeddings (CBOW)

Schematic of CBOW

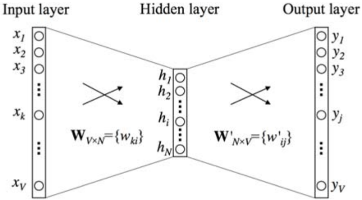


Schematic of CBOW



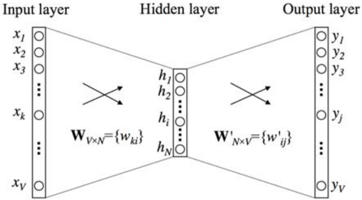
In practice...

In practice. . .



We have a vocabulary of size V: this is the size of the input 'layer' (the context words) and the size of the output 'layer' (the target words).

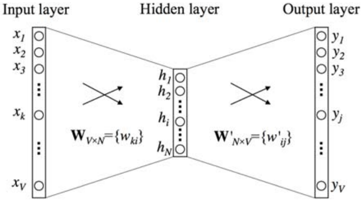
In practice. . .



We have a vocabulary of size V: this is the size of the input ‘layer’ (the context words) and the size of the output ‘layer’ (the target words).

$\rightarrow$ We want to create embedding vectors, which sum up how a given word is related to every other word.

In practice. . .

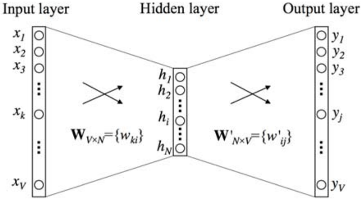


We have a vocabulary of size V: this is the size of the input 'layer' (the context words) and the size of the output 'layer' (the target words).

$\rightarrow$ We want to create embedding vectors, which sum up how a given word is related to every other word.

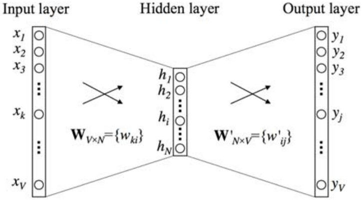
In practice. . .

In practice. . .



Our inputs (our one-hot encodings) will be transformed via a hidden layer of N 'neurons'. How many neurons?

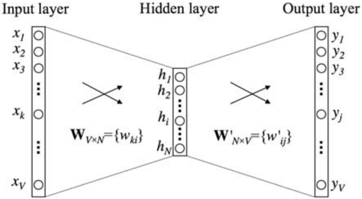
In practice...



Our inputs (our one-hot encodings) will be transformed via a hidden layer of N 'neurons'. How many neurons?

More $\rightarrow$ better at representing our inputs (bigger embedding vector).

In practice. . .

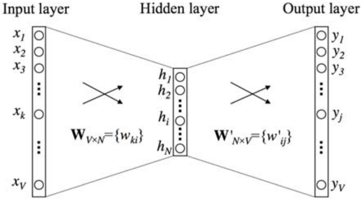


Our inputs (our one-hot encodings) will be transformed via a hidden layer of N 'neurons'. How many neurons?

More $\rightarrow$ better at representing our inputs (bigger embedding vector).

But that means we need more data.

In practice...



Our inputs (our one-hot encodings) will be transformed via a hidden layer of N 'neurons'. How many neurons?

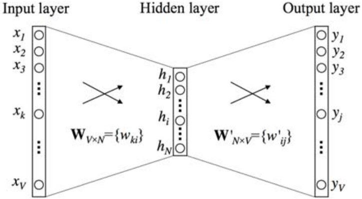
More $\rightarrow$ better at representing our inputs (bigger embedding vector).

But that means we need more data.

We go from the input layer to the hidden layer via a weight matrix W_I of size $V \times N$—each row represents a context word.

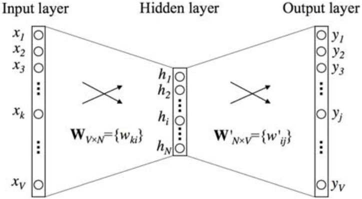
In practice. . .

In practice...



We go from the hidden layer to the output layer via a (different) weight matrix W_O of size $N \times V$—each column represents a target word.

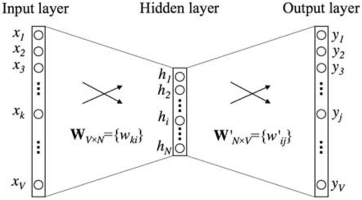
In practice. . .



We go from the hidden layer to the output layer via a (different) weight matrix W_O of size $N \times V$—each column represents a target word.

Then we'll use softmax to calculate $Pr(\text{word}_k|\text{target})$.

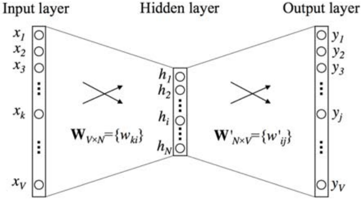
In practice. . .



We go from the hidden layer to the output layer via a (different) weight matrix W_O of size $N \times V$—each column represents a target word.

Then we'll use softmax to calculate $Pr(\text{word}_k|\text{target})$. We'll maximize this probability, and that will optimize the weight matrices.

In practice...



We go from the hidden layer to the output layer via a (different) weight matrix W_O of size $N \times V$—each column represents a target word.

Then we'll use softmax to calculate $Pr(\text{word}_k|\text{target})$. We'll maximize this probability, and that will optimize the weight matrices.

The embedding vector for a given word is a row of W_I.

Notes

Notes

CBOW is a very simple, stripped down NN:

Notes

CBOW is a very simple, stripped down NN: it doesn't actually have a non-linear transformation (just linear).

Notes

CBOW is a very simple, stripped down NN: it doesn't actually have a non-linear transformation (just linear). But lots of related models and extensions do use non-linear transformations.

Notes

CBOW is a very simple, stripped down NN: it doesn't actually have a non-linear transformation (just linear). But lots of related models and extensions do use non-linear transformations.

The softmax we want to optimize is the probability of seeing a given (target) word, given the index word we saw:

Notes

CBOW is a very simple, stripped down NN: it doesn't actually have a non-linear transformation (just linear). But lots of related models and extensions do use non-linear transformations.

The softmax we want to optimize is the probability of seeing a given (target) word, given the index word we saw:

$$\Pr(w_j | w_I) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})}$$

Notes

CBOW is a very simple, stripped down NN: it doesn't actually have a non-linear transformation (just linear). But lots of related models and extensions do use non-linear transformations.

The softmax we want to optimize is the probability of seeing a given (target) word, given the index word we saw:

$$\Pr(w_j|w_I) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})}$$

u_j is, essentially, the model representation of the word: i.e. the combination of the (weighted) input vector for a given word multiplied by the (weighted) output vector.

Notes

CBOW is a very simple, stripped down NN: it doesn't actually have a non-linear transformation (just linear). But lots of related models and extensions do use non-linear transformations.

The softmax we want to optimize is the probability of seeing a given (target) word, given the index word we saw:

$$\Pr(w_j|w_I) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})}$$

u_j is, essentially, the model representation of the word: i.e. the combination of the (weighted) input vector for a given word multiplied by the (weighted) output vector.

$\rightarrow$ probability should be one if the target always follows the context word; zero if it never does, etc.

Embedding Vectors

Embedding Vectors

Once we've optimized,

Embedding Vectors

Once we've optimized, we can extract the word specific vectors from W_I as embedding vectors (equal in length to number of neurons).

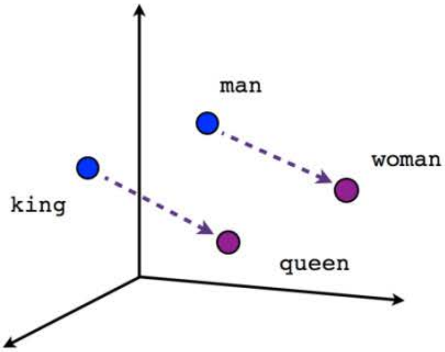
Embedding Vectors

Once we've optimized, we can extract the word specific vectors from W_I as embedding vectors (equal in length to number of neurons).
These real valued vectors can be used for analogies and related tasks,

Embedding Vectors

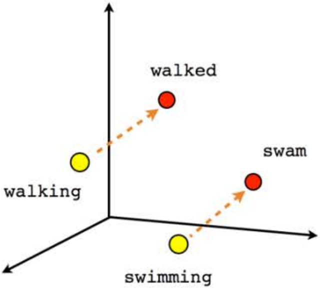
Once we've optimized, we can extract the word specific vectors from W_I as embedding vectors (equal in length to number of neurons). These real valued vectors can be used for analogies and related tasks,

$$v_{\text{queen}} - v_{\text{woman}} + v_{\text{men}} \approx v_{\text{king}}$$

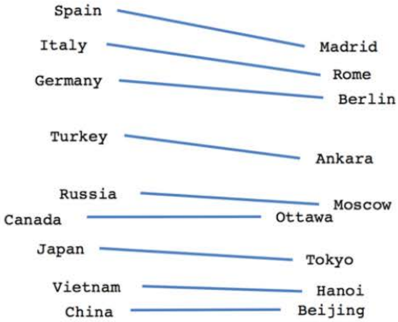


Related Tasks

Related Tasks



Verb tense



Country-Capital

Partner Exercise

Partner Exercise

1. How do we know whether a word embedding vector is a good representation or not? How could we test the merits of one particular model versus another?

2. Embeddings reflect cultural biases. How would you show this in practice given what we discussed above?

Skip-Gram

CBOW can be generalized to multiple context words.

Skip-Gram

CBOW can be generalized to multiple context words. But Skip-Gram generally more popular.

Skip-Gram

CBOW can be generalized to multiple context words. But Skip-Gram generally more popular.

The Skip-gram model predicts the context given the word. This means the output (the context) is two hot-encoded vectors (assuming two word context).

Skip-Gram

CBOW can be generalized to multiple context words. But Skip-Gram generally more popular.

The Skip-gram model predicts the context given the word. This means the output (the context) is two hot-encoded vectors (assuming two word context).

$\rightarrow$ Skip-gram is more subtle:

Skip-Gram

CBOW can be generalized to multiple context words. But Skip-Gram generally more popular.

The Skip-gram model predicts the context given the word. This means the output (the context) is two hot-encoded vectors (assuming two word context).

$\rightarrow$ Skip-gram is more subtle: contexts can be different for homographs (like 'JFK' the airport vs 'JFK' the President).

Skip-Gram

CBOW can be generalized to multiple context words. But Skip-Gram generally more popular.

The Skip-gram model predicts the context given the word. This means the output (the context) is two hot-encoded vectors (assuming two word context).

$\rightarrow$ Skip-gram is more subtle: contexts can be different for homographs (like 'JFK' the airport vs 'JFK' the President).

Shows excellent performance on many tasks (better than CBOW).

Notes II

Notes II

Why does Word2Vec work?

Notes II

Why does Word2Vec work? Area of some interest: see e.g. Levy & Goldberg in NIPS, 2014.

Notes II

Why does Word2Vec work? Area of some interest: see e.g. Levy & Goldberg in NIPS, 2014.

Producing word embedding vectors is computationally expensive, and you need a lot of data.

Notes II

Why does Word2Vec work? Area of some interest: see e.g. Levy & Goldberg in NIPS, 2014.

Producing word embedding vectors is computationally expensive, and you need a lot of data. So pre-trained embeddings (esp from Wikipedia) are available.

Notes II

Why does Word2Vec work? Area of some interest: see e.g. Levy & Goldberg in NIPS, 2014.

Producing word embedding vectors is computationally expensive, and you need a lot of data. So pre-trained embeddings (esp from Wikipedia) are available.

GloVe, Global Vectors for Word Representation, is method that works via dimension reduction of counts of co-occurrences (rather than trying to predict, as Word2Vec)

Notes II

Why does Word2Vec work? Area of some interest: see e.g. Levy & Goldberg in NIPS, 2014.

Producing word embedding vectors is computationally expensive, and you need a lot of data. So pre-trained embeddings (esp from Wikipedia) are available.

GloVe, Global Vectors for Word Representation, is method that works via dimension reduction of counts of co-occurrences (rather than trying to predict, as Word2Vec)

Can we provide framework for embeddings so we can talk about one word being statistically significantly different to another? (yes! Cho et al, 2018). And perhaps make embeddings dependent on covariates? (Rudolph et al, 2017)