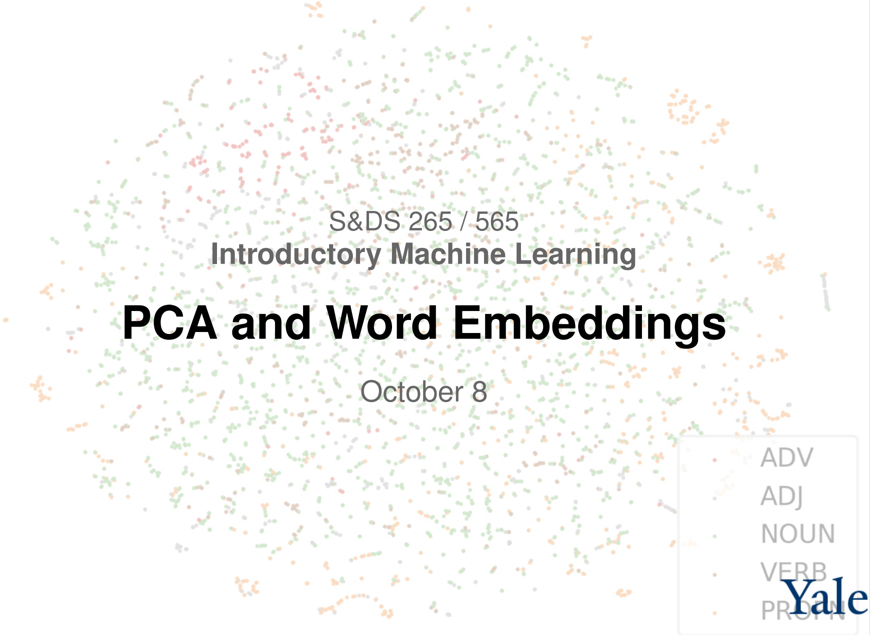
S&DS 265 / 565

Introductory Machine Learning

PCA and Word Embeddings

October 8



Plan for today

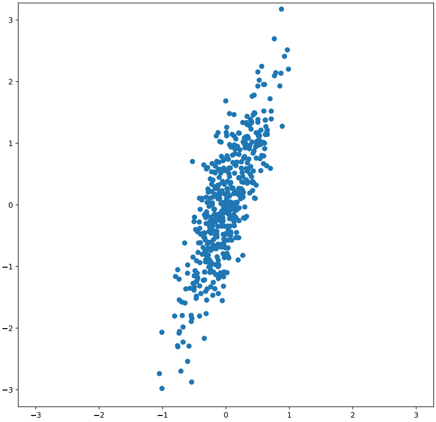
- Reminders
- Recap of PCA
- Demo notebooks
- Word embeddings

Reminders

- Assn 3 is out; due October 24
- Quiz 3 on Thursday; open at 1pm; discriminitive vs. generative models, trees, bias/variance, SGD
- Midterm next Tuesday, October 15, in class
- “Closed book, notes, computer...”
- $8\frac{1}{2} \times 11$ sheet of notes, handwritten double-sided
- Practice midterms posted on Canvas (with solutions)
- Will go over practice exams in review sessions
- Questions?

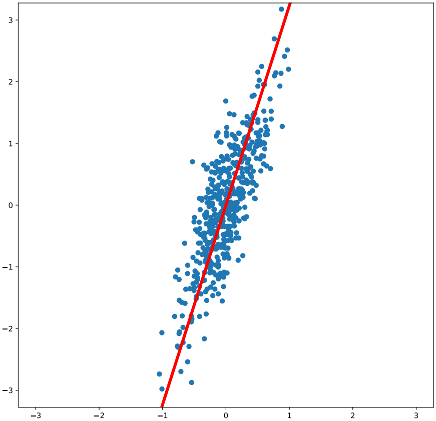
Principal Component Analysis (PCA)

PCA finds the directions of greatest variability in the data.

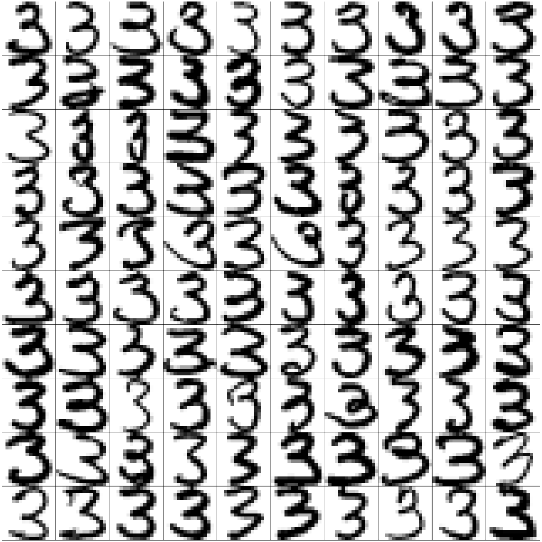


Principal Component Analysis (PCA)

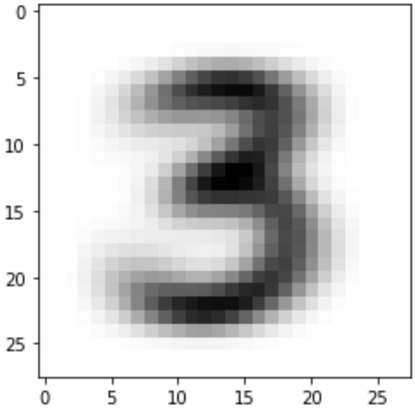
PCA finds the directions of greatest variability in the data.



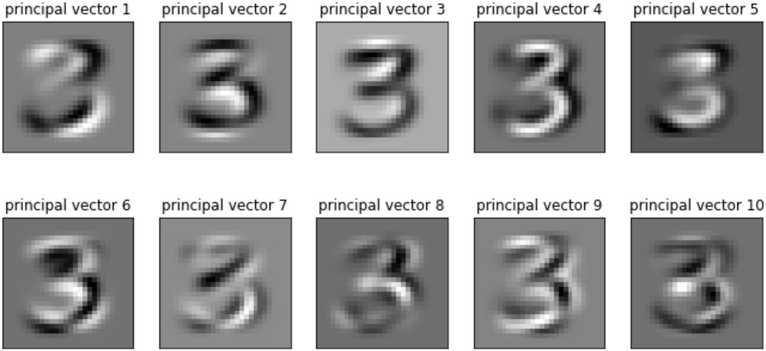
Handwritten Digits (3s)



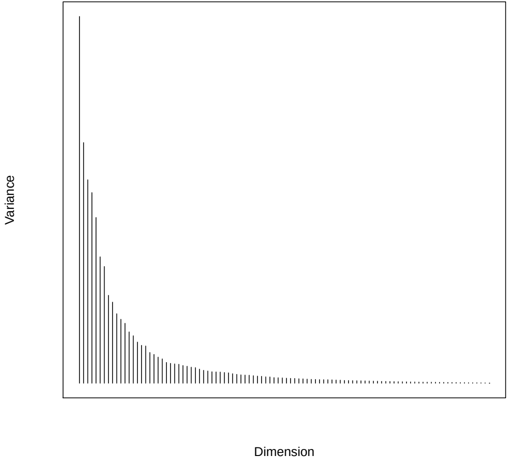
Handwritten Digits (3s) – Average



Handwritten Digits (3s) – Principal vectors

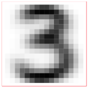
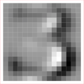
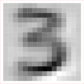


Handwritten Digits (3s) – PCA variance



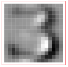
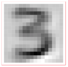
Handwritten Digits (3s)

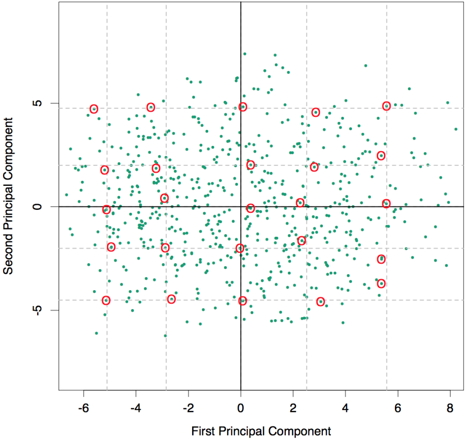
$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

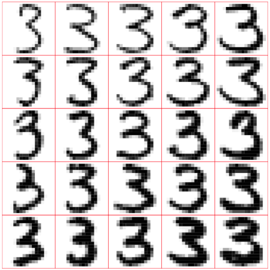
$=$  $+ \lambda_1 \cdot$  $+ \lambda_2 \cdot$  .

Handwritten Digits (3s) – Top 2 components

$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

$$= \quad$$  $$+ \lambda_1 \cdot$$  $$+ \lambda_2 \cdot$$  .





PCA: Algorithm

1. Center the data: $x_i \mapsto x_i - \overline{x}$
2. Compute the $d \times d$ sample covariance $S = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$
3. Find the first k eigenvectors of S
4. Project the data onto those k vectors

PCA: Algorithm

1. Center the data: $x_i \mapsto x_i - \frac{1}{n}\sum_{j=1}^{n} x_j = x_i - \overline{x}$

2. Compute the $d \times d$ sample covariance $S = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$. Note that
$$\frac{1}{n}\sum_i \left(x_{ij} - \overline{x}\right)^2$$
is the sample variance of jth coordinate of data.

3. Find the first k eigenvectors of S,
$$v_1, \ldots, v_k \in \mathbb{R}^d, \qquad Sv_j = \lambda_j v_j$$

4. Project the data onto those k vectors:
$$x_i \mapsto \overline{x} + (v_1^T x_i)v_1 + \ldots + (v_k^T x_i)v_k$$

Published in final edited form as:
Nature. 2008 Nov 6; 456(7218): 98–101.
Published online 2008 Aug 31. doi: 10.1038/nature07331
NIHMSID: NIHMS132060

Genes mirror geography within Europe

John Novembre,[1,2] Toby Johnson,[4,5,6] Katarzyna Bryc,[7] Zoltán Kutalik,[4,6] Adam R. Boyko,[7] Adam Auton,[7] Amit Indap,[7] Karen S. King,[8] Sven Bergmann,[4,6] Matthew R. Nelson,[8] Matthew Stephens,[2,3] and Carlos D. Bustamante[7]

Author information ▶ Copyright and License information ▶

The publisher's final edited version of this article is available at Nature
This article has been corrected. See the correction in volume 456 on page 274.
See commentary "Editorial comment should accompany hot papers online." in *Nature*, volume 455 on page 861.

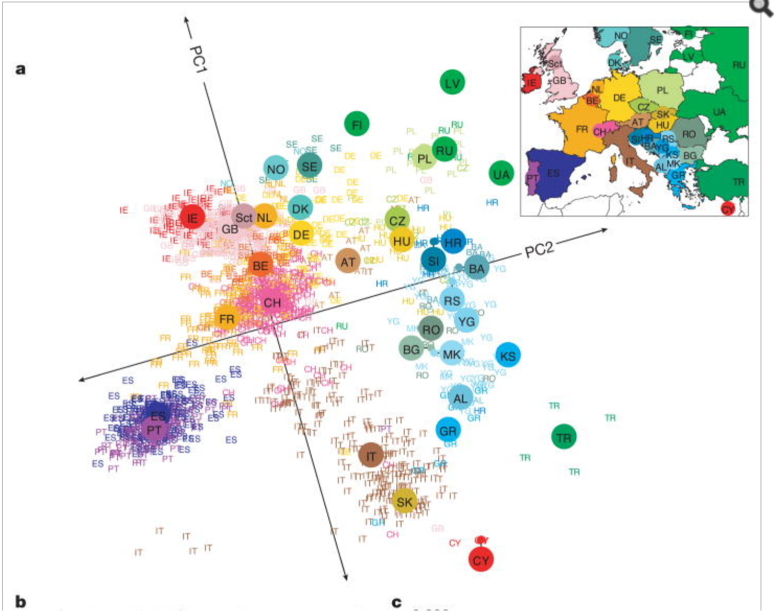
See other articles in PMC that cite the published article.

Abstract

Go to:

Understanding the genetic structure of human populations is of fundamental interest to medical, forensic and anthropological sciences. Advances in high-throughput genotyping technology have markedly improved our understanding of global patterns of human genetic variation and suggest the potential to use large samples to uncover variation among closely spaced populations[1–5]. Here we characterize genetic variation in a sample of 3,000 European individuals genotyped at over half a million variable DNA sites in the human genome. Despite low average levels of genetic differentiation among Europeans, we find a close correspondence between genetic and geographic distances; indeed, a geographical map of Europe arises naturally as an efficient two-dimensional summary of genetic variation in Europeans. The results emphasize that when mapping the genetic basis of a disease phenotype, spurious associations can arise if genetic structure is not properly accounted for. In addition, the results are relevant to the prospects of genetic ancestry testing[6]; an individual's DNA can be used to infer their geographic origin with surprising accuracy—often to within a few hundred kilometres.

https://ghr.nlm.nih.gov/primer/genomicresearch/snp



https://ghr.nlm.nih.gov/primer/genomicresearch/snp

PCA: Algorithm

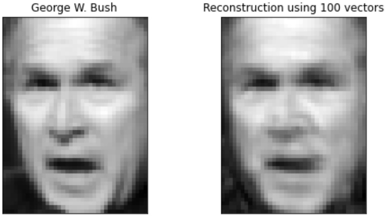
1. We can compute everything directly
2. Except for the eigenvectors
3. Let’s illustrate this in the demo notebook

Let’s go to the notebook

The number $x^T v_k$ is the amount of x that lies in the direction of the principal vector v_k. This is easily translated into Python:

In [26]:
```
v = principal_vectors.reshape(num_components, height*width)

xhat = avgimg
for k in np.arange(num_components):
    xhat = xhat + np.dot(x, v[k]) * v[k]
    plot_face_reconstruction(x, xhat, 'George W. Bush', 'Reconstruction using %d vectors' % (k+
```



Using PCA for classification or regression

- A combination of supervised learning and unsupervised learning
- Given data $\{x\}$ extract principal vectors and components
- Map each data point x_i to its principal components

$$z_i \equiv (x_i^T v_1, \ldots, x_i^T v_K)$$

- For labeled data $\{(x_i, y_i)\}$, now train a supervised learning algorithm using the transformed data $\{(z_i, y_i)\}$.

Example notebook

Flower Power: PCA and classification (30 points)

In this problem you will carry out principal components analysis and classification on the iris data. The task will be to reduce the dimension from four to two using PCA, and then to train logistic regression models on the projected data.

Summary: PCA

- PCA is an unsupervised method
- Finds directions of greatest variation in the data
- The directions are called the *principal vectors*; the weightings on the vectors are called the *principal components*
- The first few vectors may be interpretable
- Orthogonality makes interpretation difficult for the higher components
- Can be used for visualization or dimensionality reduction

Language models

- A language model is a way of assigning a probability to any sequence of words (or string of text)

$$p(w_1, \ldots, w_n)$$

Language models

- A language model is a way of assigning a probability to any sequence of words (or string of text)

$$p(w_1, \dots, w_n)$$

- By the basic rules of conditional probability we can factor this as

$$p(w_1, \dots, w_n) = p(w_1)p(w_2 \mid w_1) \dots p(w_n \mid w_1, \dots, w_{n-1})$$

Modern language models

Suppose a computer program assigns a "score" to possible next words v:

$$s(\underset{\substack{\uparrow \\ \text{possible next word}}}{v};\ \overbrace{w_1, \ldots, w_n}^{\text{previous words}})$$

https://en.wikipedia.org/wiki/Softmax_function

Modern language models

Suppose a computer program assigns a "score" to possible next words v:

$$s(\underset{\substack{\uparrow \\ \text{possible next word}}}{v};\ \overbrace{w_1, \ldots, w_n}^{\text{previous words}})$$

Can convert this to a language model by the "softmax" operation:

$$p(w \mid w_1, \ldots, w_n) = \frac{\exp(s(w; w_1, \ldots, w_n))}{\sum_{v \in V} \exp(s(v; w_1, \ldots, w_n))}$$

https://en.wikipedia.org/wiki/Softmax_function

Modern language models

Suppose a computer program assigns a "score" to possible next words v:

$$s(\underset{\substack{\uparrow \\ \text{possible next word}}}{v}; \overbrace{w_1, \ldots, w_n}^{\text{previous words}})$$

Can convert this to a language model by the "softmax" operation:

$$p(w \mid w_1, \ldots, w_n) = \frac{\exp(s(w; w_1, \ldots, w_n))}{\sum_{v \in V} \exp(s(v; w_1, \ldots, w_n))}$$

In ChatGPT, the function $s(v; w_{1:n})$ is learned on large amounts of text (unsupervised) using a type of deep neural network called a *transformer*.

---

https://en.wikipedia.org/wiki/Softmax_function

Modern language models

A language model assigns a "score" to possible next words v:

$$s(v;\ \underbrace{w_1, \ldots, w_n}_{\text{word history}}\ )$$

Modern language models

A language model assigns a "score" to possible next words v:

$$s(v;\ \underbrace{w_1, \ldots, w_n}_{\text{word history}}\ )$$

Today, we'll be working with a simple case where

$$\begin{aligned} s(v; w_1, \ldots, w_n) &= \beta_v^T \varphi(w_1, \ldots, w_n) \\ &= \beta_v^T \varphi(w_n) \\ &= \varphi(v)^T \varphi(w_n) \end{aligned}$$

Modern language models

A language model assigns a “score” to possible next words v:

$$s(v;\ \underbrace{w_1, \ldots, w_n}_{\text{word history}}\ )$$

Today, we’ll be working with a simple case where

$$\begin{aligned} s(v; w_1, \ldots, w_n) &= \beta_v^T \varphi(w_1, \ldots, w_n) \\ &= \beta_v^T \varphi(w_n) \\ &= \varphi(v)^T \varphi(w_n) \end{aligned}$$

Key intuition

- Similar words will appear with similar words
- Self-referential notion of similarity

Constructing embeddings

Language model is

$$p(w_2 \mid w_1) = \frac{\exp(\varphi(w_2)^T \varphi(w_1))}{\sum_w \exp(\varphi(w)^T \varphi(w_1))}.$$

Carry out stochastic gradient descent over the embedding vectors $\varphi \in \mathbb{R}^d$ (where $d \approx 50$–500 is chosen by hand)

This is what Mikolov et al. (2014, 2015) did at Google. With a couple of twists:

“Distributed representations of words,” (2014) “Efficient representations of words in vector space” (2015)

Constructing embeddings

`word2vec`:

- Skip-gram: predict surrounding words from current word, rather than the next word.

"Distributed representations of words," (2014) "Efficient representations of words in vector space" (2015)

Constructing embeddings

`word2vec`:

- Skip-gram: predict surrounding words from current word, rather than the next word.
- This leads to a model of nearby words $p_{\text{near}}(w_2 \mid w_1)$.

"Distributed representations of words," (2014) "Efficient representations of words in vector space" (2015)

Constructing embeddings

`word2vec`:

- Skip-gram: predict surrounding words from current word, rather than the next word.
- This leads to a model of nearby words $p_{\text{near}}(w_2 \mid w_1)$.
- Computational bottleneck is computing the denominator in the softmax.

"Distributed representations of words," (2014) "Efficient representations of words in vector space" (2015)

GloVe

Shortly after, a group at Stanford group introduced a variant called "GloVe"

- Based on a type of regression model
- More scalable with SGD

Pennington et al., "GloVe: Global vectors for word representation," (2015)

Using PCA

A closely related approach is to use PCA of pointwise mutual information (PMI):

- Form $V \times V$ matrix of pointwise mutual information values

$$\log\left(\frac{p_{\text{near}}(w_1, w_2)}{p(w_1)p(w_2)}\right)$$

- Compute top k eigenvectors $\varphi_1, \ldots, \varphi_k$
- For each word w, define embedding as

$$\varphi(w) \equiv (\varphi_{1w}, \varphi_{2w}, \ldots, \varphi_{kw})^T$$

Analogies

Leads to vector representations of words with interesting properties.

For example, analogies:

`king` is to `man` as ? is to `woman`

Analogies

Leads to vector representations of words with interesting properties.

For example, analogies:

`king` is to `man` as ? is to `woman`

`Paris` is to `France` as ? is to `Germany`

Analogies

Leads to vector representations of words with interesting properties.

For example, analogies:

`king` is to `man` as ? is to `woman`

`Paris` is to `France` as ? is to `Germany`

$$\varphi(\texttt{king}) - \varphi(\texttt{man}) \overset{?}{\approx} \varphi(\texttt{queen}) - \varphi(\texttt{woman})$$

$$\widehat{w} = \arg\min_{w} \|\varphi(\texttt{king}) - \varphi(\texttt{man}) + \varphi(\texttt{woman}) - \varphi(w)\|^2$$

Does $\widehat{w} = \texttt{queen}$?

Learned Analogies

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

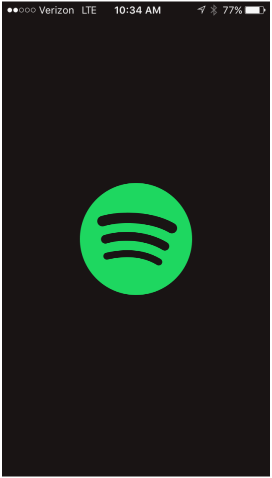
Mikolov et al., “Distributed representations of words,” (2014); “Efficient representations of words in vector space” (2015)

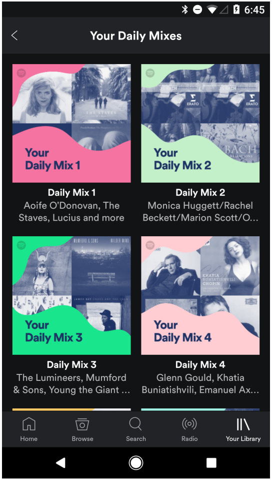
Evaluation Analogies

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

Mikolov et al., “Distributed representations of words,” (2014); “Efficient representations of words in vector space” (2015)

Recommendation via Embedding





Notebook

Let's go to the Python notebook!

Embedding / Visualization Examples

WebVectors | Similar words | Visualizations | Calculator | 2D text | Miscellaneous | Models | About

WebVectors: word embeddings online

'You shall know a word by the company it keeps.' (Firth 1957)

Enter a word to produce a list of its 10 nearest semantic associates.
English Wikipedia model will be used; for other models, visit Similar Words tab.

platypus_NOUN

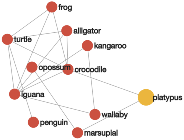
Find similar words!

Semantic associates for *platypus* (computed on English Wikipedia)

Word frequency

☑ High ☑ Medium ☐ Low

1. marsupial 0.642
2. crocodile 0.605
3. kangaroo 0.595
4. turtle 0.595
5. iguana 0.589
6. frog 0.573
7. penguin 0.572
8. wallaby 0.570
9. alligator 0.569
10. opossum 0.568



0.6 Similarity threshold ☐ Show tags

- *We show only the associates of the same part of speech as your query. All associates can be found at the Similar Words tab.*

http://vectors.nlpl.eu/explore/embeddings/en/

Summary: Word embeddings

- Word embeddings are vector representations of words, learned from cooccurrence statistics
- The models can be built using language modeling, (or regression or PCA)
- Surprising semantic relations are encoded in linear relations—for example, analogies
- Embeddings improve with more data