

S&DS 265 / 565
Introductory Machine Learning

PCA and Word Embeddings

October 10

ADV
ADJ
NOUN
VERB
PRON
Yale

Plan for today

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

Reminders

- Assn 3 is out; due October 26
- Quiz 3 on Thursday; open at 1pm, closes Friday at 6pm
- Midterm next Tuesday, October 17, 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?

Review sessions

Regina: Thursday, October 12, 8-9pm

Hannah: Friday, October 13, 5-6pm

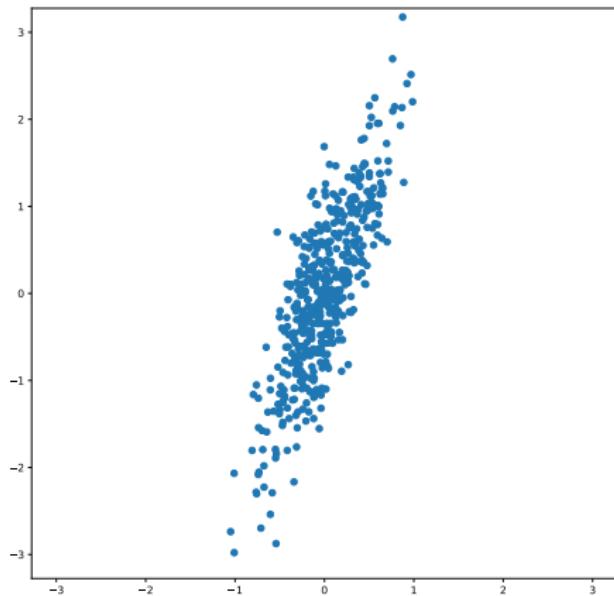
Kaylee: Saturday, October 14, 5-6pm

Awni: Sunday, October 15, 2-3pm

Held in Kline Tower (KT) 211

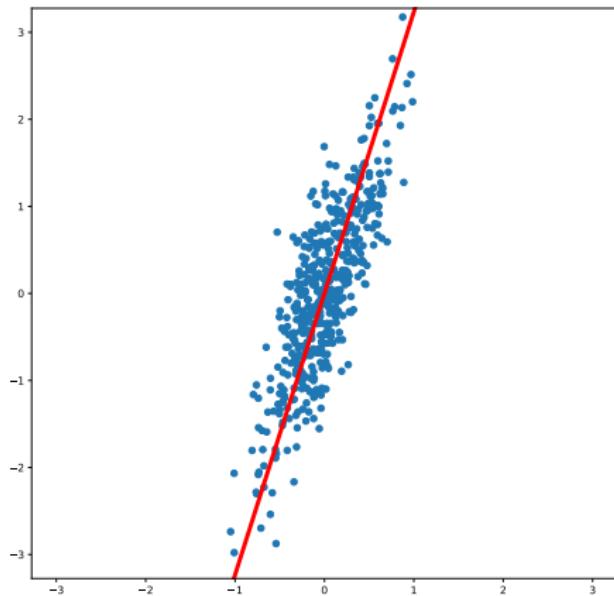
Principal Component Analysis (PCA)

PCA finds the directions of greatest variability in the data.



Principal Component Analysis (PCA)

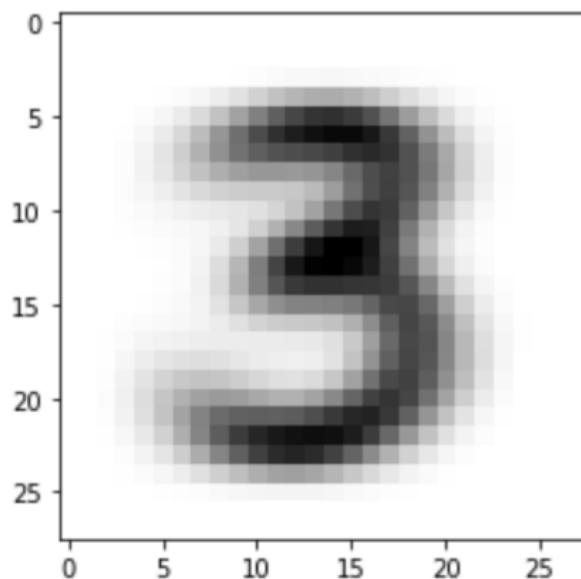
PCA finds the directions of greatest variability in the data.



Handwritten Digits (3s)

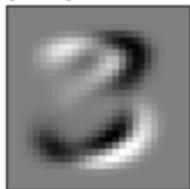
3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3

Handwritten Digits (3s) – Average

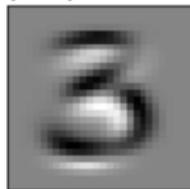


Handwritten Digits (3s) – Principal vectors

principal vector 1



principal vector 2



principal vector 3



principal vector 4



principal vector 5



principal vector 6



principal vector 7



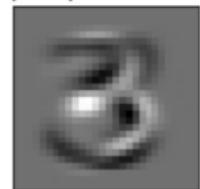
principal vector 8



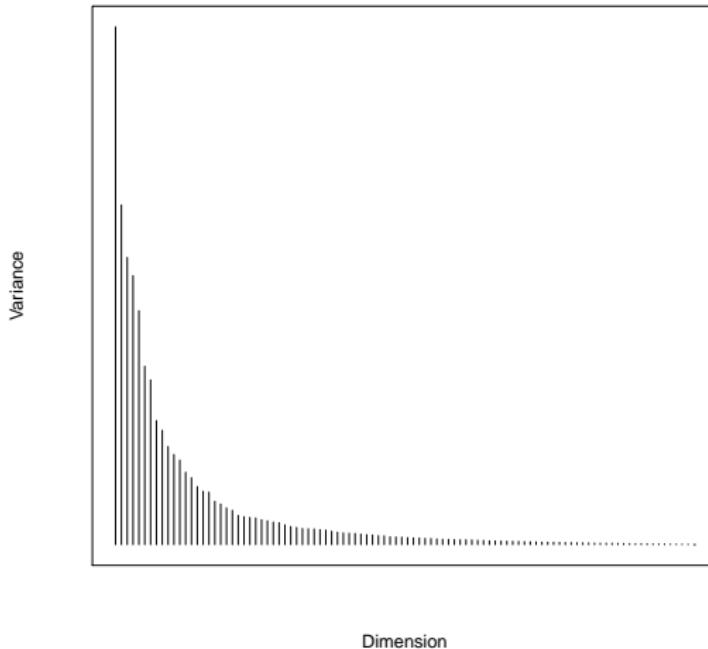
principal vector 9



principal vector 10



Handwritten Digits (3s) – PCA variance

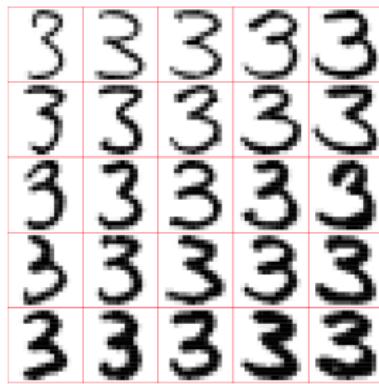
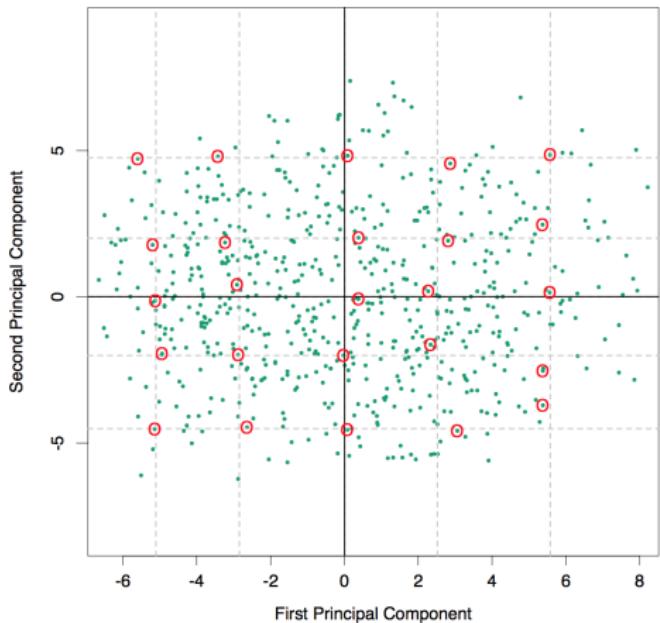


Handwritten Digits (3s)

$$\begin{aligned}\hat{f}(\lambda) &= \bar{x} + \lambda_1 v_1 + \lambda_2 v_2 \\ &= \boxed{3} + \lambda_1 \cdot \boxed{3} + \lambda_2 \cdot \boxed{3}.\end{aligned}$$

Handwritten Digits (3s) – Top 2 components

$$\begin{aligned}\hat{f}(\lambda) &= \bar{x} + \lambda_1 v_1 + \lambda_2 v_2 \\ &= \boxed{\text{3}} + \lambda_1 \cdot \boxed{\text{3}} + \lambda_2 \cdot \boxed{\text{3}}.\end{aligned}$$



PCA: Algorithm

- ① Center the data: $x_i \mapsto x_i - \bar{x}$
- 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: $x_i \mapsto x_i - \frac{1}{n} \sum_{j=1}^n x_j = x_i - \bar{x}$
- ② Compute the $d \times d$ sample covariance $S = \frac{1}{n} \sum_{i=1}^n x_i x_i^T$. Note that

$$\frac{1}{n} \sum_i (x_{ij} - \bar{x})^2$$

is the sample variance of j th coordinate of data.

- ③ Find the first k eigenvectors of S ,

$$v_1, \dots, v_k \in \mathbb{R}^d, \quad S v_j = \lambda_j v_j$$

- ④ Project the data onto those k vectors:

$$x_i \mapsto \bar{x} + (v_1^T x_i) v_1 + \dots + (v_k^T x_i) v_k$$

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+1))
```

George W. Bush



Reconstruction using 100 vectors



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

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

- 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)

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

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

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

Modern language models

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

$$s(v; \overbrace{w_1, \dots, w_n}^{\text{previous words}})$$

↑
possible next word

Modern language models

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

$$s(v; \overbrace{w_1, \dots, w_n}^{\text{previous words}})$$

↑
possible next word

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

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

Modern language models

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

$$s(v; \overbrace{w_1, \dots, w_n}^{\text{previous words}})$$

↑
possible next word

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

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

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*.

Modern language models

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

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

Modern language models

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

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

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

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

Modern language models

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

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

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

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

Key intuition

- Words that have similar neighbors will be similar
- Self-referential notion of similarity

Constructing embeddings

Language model is

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

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

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

Constructing embeddings

Heuristics used:

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

Constructing embeddings

Heuristics used:

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

Constructing embeddings

Heuristics used:

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

Constructing embeddings

Heuristics used:

- Skip-gram: predict surrounding words from current word, rather than the next word.
- This leads to a model of nearby words $p_{\text{near}}(w_2 | w_1)$.
- Second is computational. The bottleneck is computing the denominator in the softmax.
- Use “negative sampling”: Approximation

$$\begin{aligned} & \sum_w \exp(v(w)^T v(w_1)) \\ & \approx \exp(v(w_2)^T v(w_1)) + \sum_{\text{random } w} \exp(v(w)^T v(w_1)) \end{aligned}$$

GloVe

Shortly after, a group at Stanford group introduced a variant called “GloVe”

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

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 v_1, \dots, v_k
- For each word w , define embedding as

$$v(w) \equiv (v_{1w}, v_{2w}, \dots, v_{kw})^T$$

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

$$v(\text{king}) - v(\text{man}) \stackrel{?}{\approx} v(\text{queen}) - v(\text{woman})$$

$$\hat{w} = \arg \min_w \|v(\text{king}) - v(\text{man}) + v(\text{woman}) - v(w)\|^2$$

Does $\hat{w} = \text{queen}$?

Learned Analogies

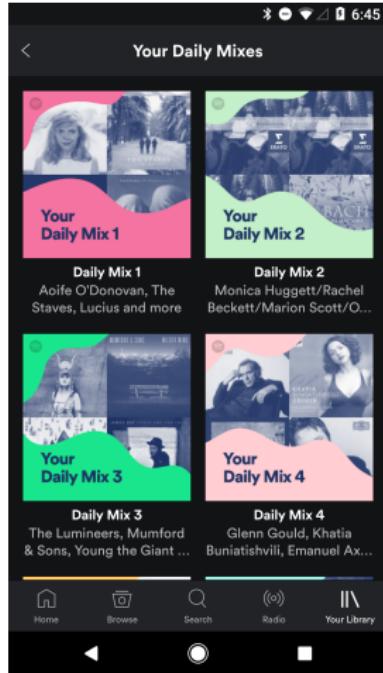
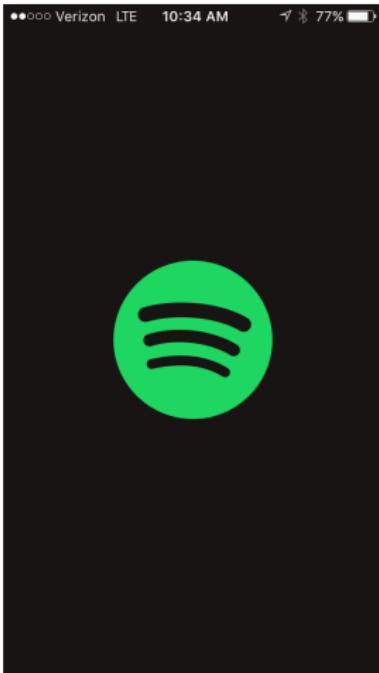
Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

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

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

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

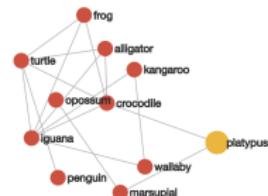
Find similar words!

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