Non-negative matrix factorization (NMF)

UNSUPERVISED LEARNING IN PYTHON

Benjamin Wilson
Director of Research at lateral.io

Non-negative matrix factorization

- NMF = "non-negative matrix factorization"
- Dimension reduction technique
- NMF models are interpretable (unlike PCA)
- Easy to interpret means easy to explain!
- However, all sample features must be non-negative (>= 0)

Interpretable parts

- NMF expresses documents as combinations of topics (or "themes")

DataCamp
DataCamp is the first and foremost leader in Data Science Education offering skill-based training, pioneering technical innovation
...

≈ 0.6 *

```
python
r
program
function
method
…
```

+ 0.5 *

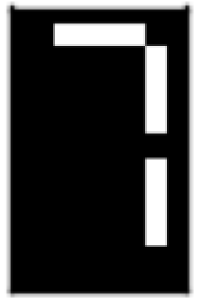
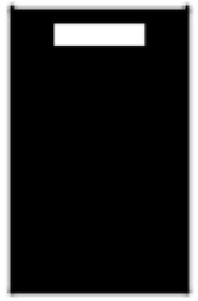
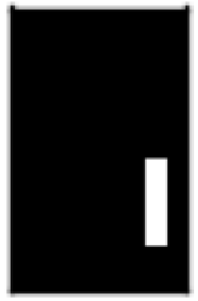
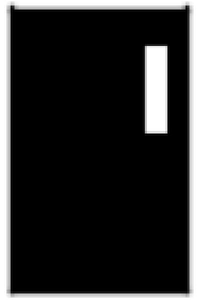
```
data
analysis
cluster
statistics
mean
…
```

+ 0.7 *

```
learn
teaching
lesson
lessons
course
…
```

Interpretable parts

- NMF expresses images as combinations of patterns

 ≈ 0.98 *  + 0.91 *  + 0.94 * 

Using scikit-learn NMF

- Follows `fit()` / `transform()` pattern
- Must specify number of components e.g. `NMF(n_components=2)`
- Works with NumPy arrays and with `csr_matrix`

Example word-frequency array

- Word frequency array, 4 words, many documents
- Measure presence of words in each document using "tf-idf"
  - "tf" = frequency of word in document
  - "idf" reduces influence of frequent words

	course	datacamp	potato	the
document0	0.2,	0.3,	0.0,	0.1
document1	0.0,	0.0,	0.4,	0.1
...		...		

Example usage of NMF

- `samples` is the word-frequency array

```
from sklearn.decomposition import NMF
model = NMF(n_components=2)
model.fit(samples)
```

```
NMF(n_components=2)
```

```
nmf_features = model.transform(samples)
```

NMF components

- NMF has components
- ... just like PCA has principal components
- Dimension of components = dimension of samples
- Entries are non-negative

```
print(model.components_)
```

```
[[ 0.01  0.    2.13  0.54]
 [ 0.99  1.47  0.    0.5 ]]
```

NMF features

- NMF feature values are non-negative
- Can be used to reconstruct the samples
- ... combine feature values with components

```
print(nmf_features)
```

```
[[ 0.    0.2 ]
 [ 0.19  0.  ]
  ...
 [ 0.15  0.12]]
```

Reconstruction of a sample

```python
print(samples[i,:])
```

```
[ 0.12  0.18  0.32  0.14]
```

```python
print(nmf_features[i,:])
```

```
[ 0.15  0.12]
```

model.components_

```
  0.15 *     [[ 0.01   0.    2.13   0.54  ]
+ 0.12 *      [ 0.99   1.47  0.     0.5   ]]
-------------------------------------------------
             [ 0.1203  0.1764  0.3195  0.141 ]
```

reconstruction of sample

Sample reconstruction

- Multiply components by feature values, and add up
- Can also be expressed as a product of matrices
- This is the "**M**atrix **F**actorization" in "NMF"

NMF fits to non-negative data only

- Word frequencies in each document
- Images encoded as arrays
- Audio spectrograms
- Purchase histories on e-commerce sites
- ... and many more!

Let's practice!

UNSUPERVISED LEARNING IN PYTHON

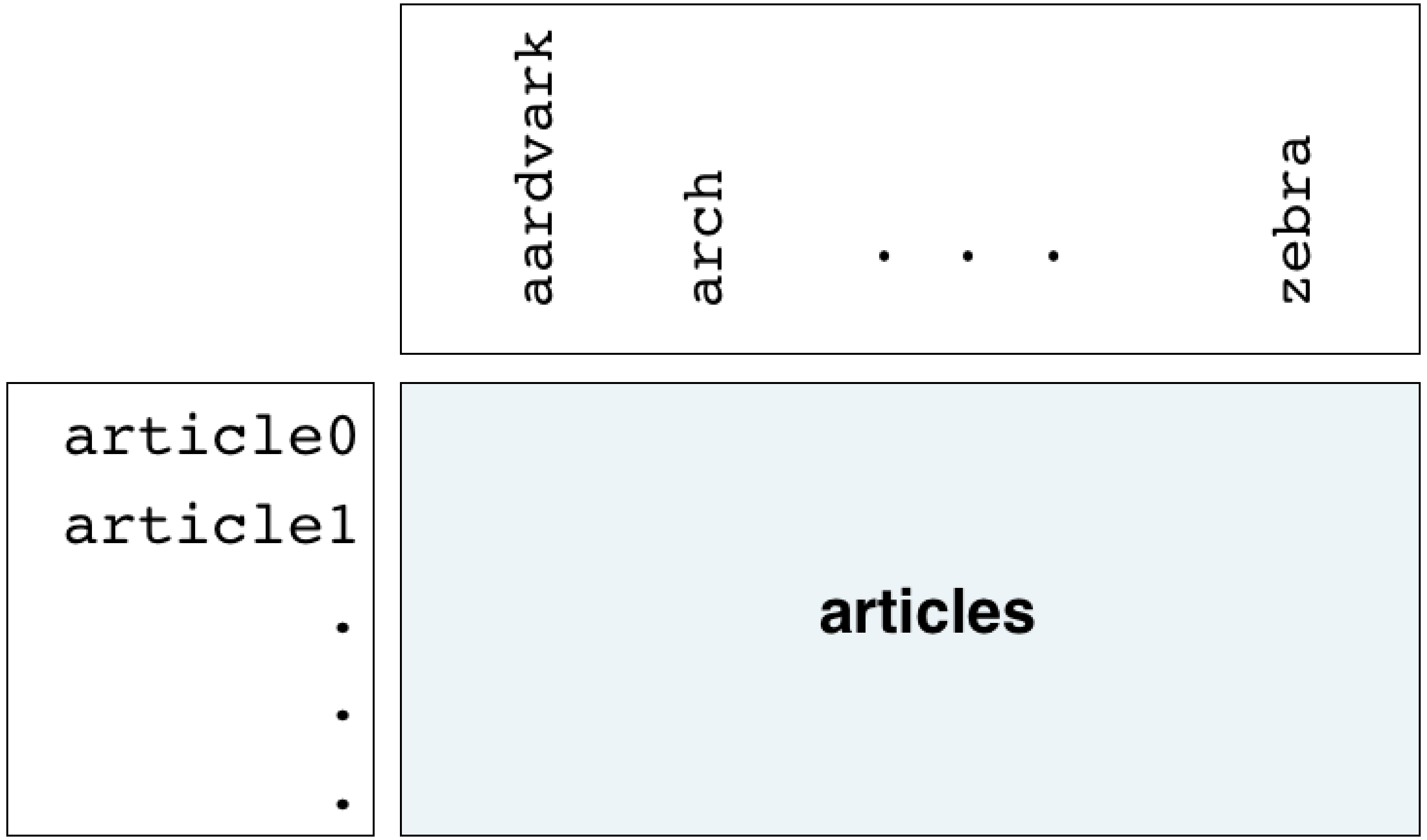
NMF learns interpretable parts

UNSUPERVISED LEARNING IN PYTHON

Benjamin Wilson
Director of Research at lateral.io

Example: NMF learns interpretable parts

- Word-frequency array articles (tf-idf)
- 20,000 scientific articles (rows)
- 800 words (columns)



Applying NMF to the articles

```
print(articles.shape)
```

```
(20000, 800)
```

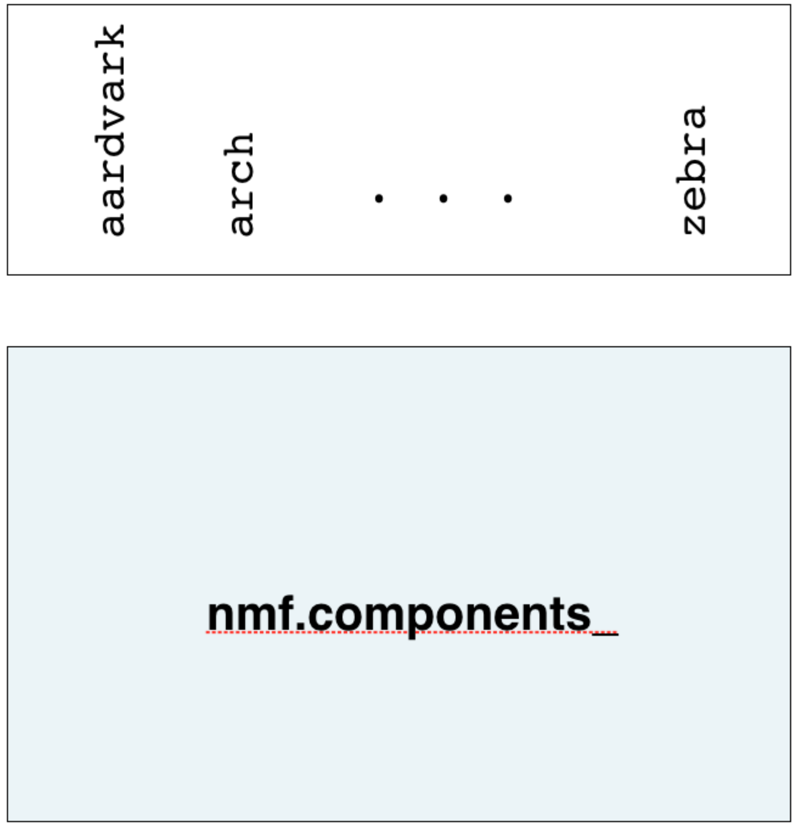
```
from sklearn.decomposition import NMF
nmf = NMF(n_components=10)
nmf.fit(articles)
```

```
NMF(n_components=10)
```

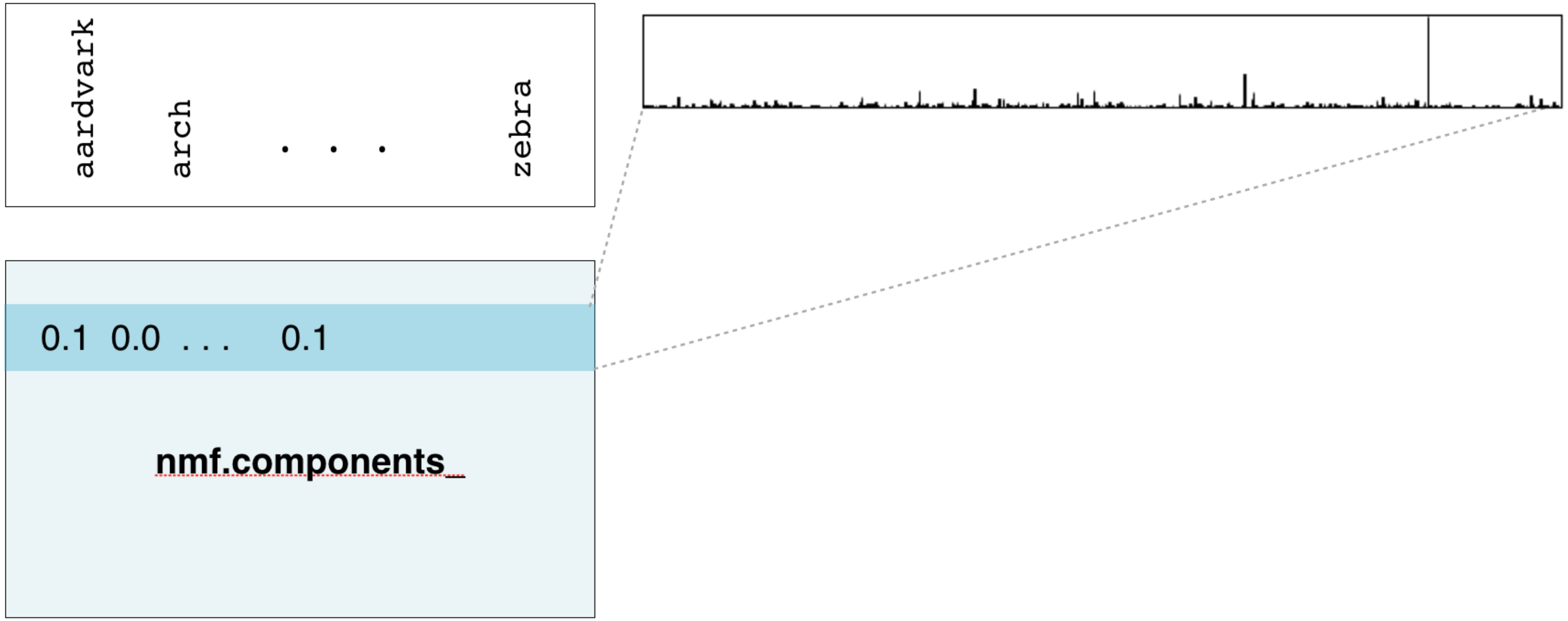
```
print(nmf.components_.shape)
```

```
(10, 800)
```

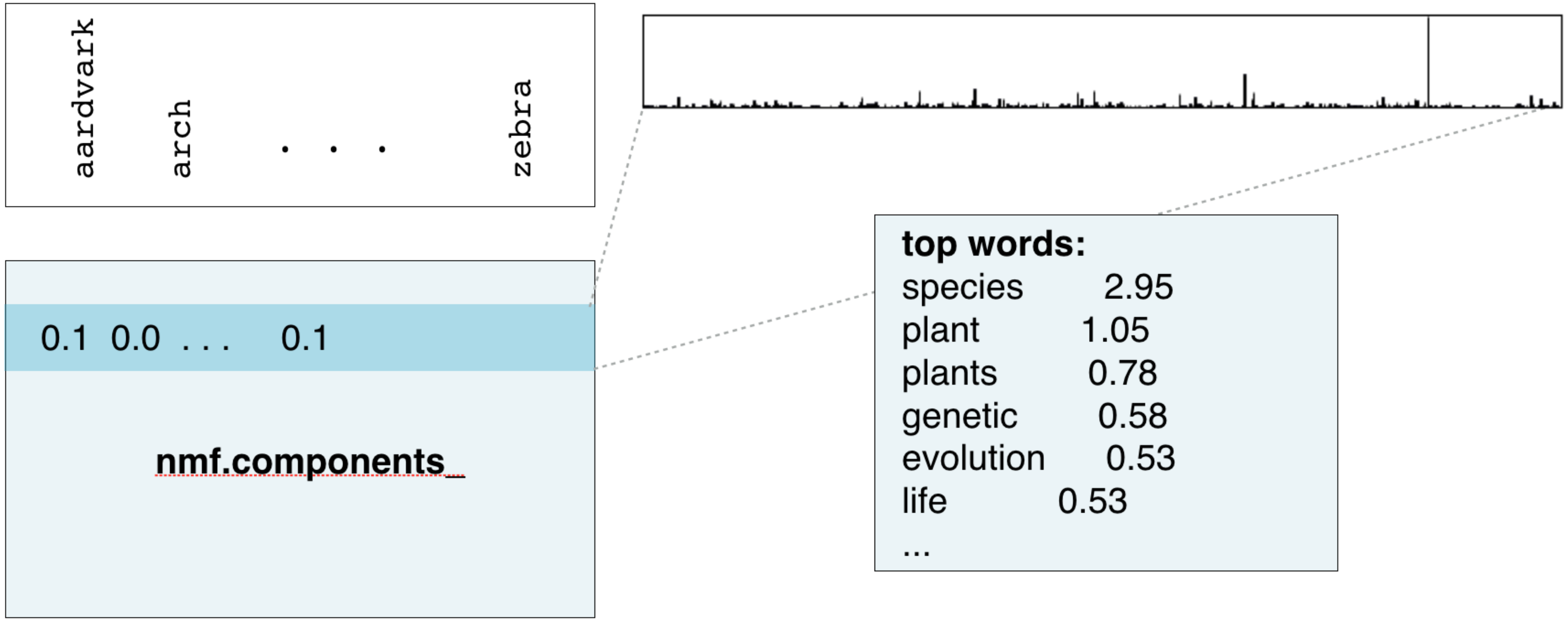
NMF components are topics



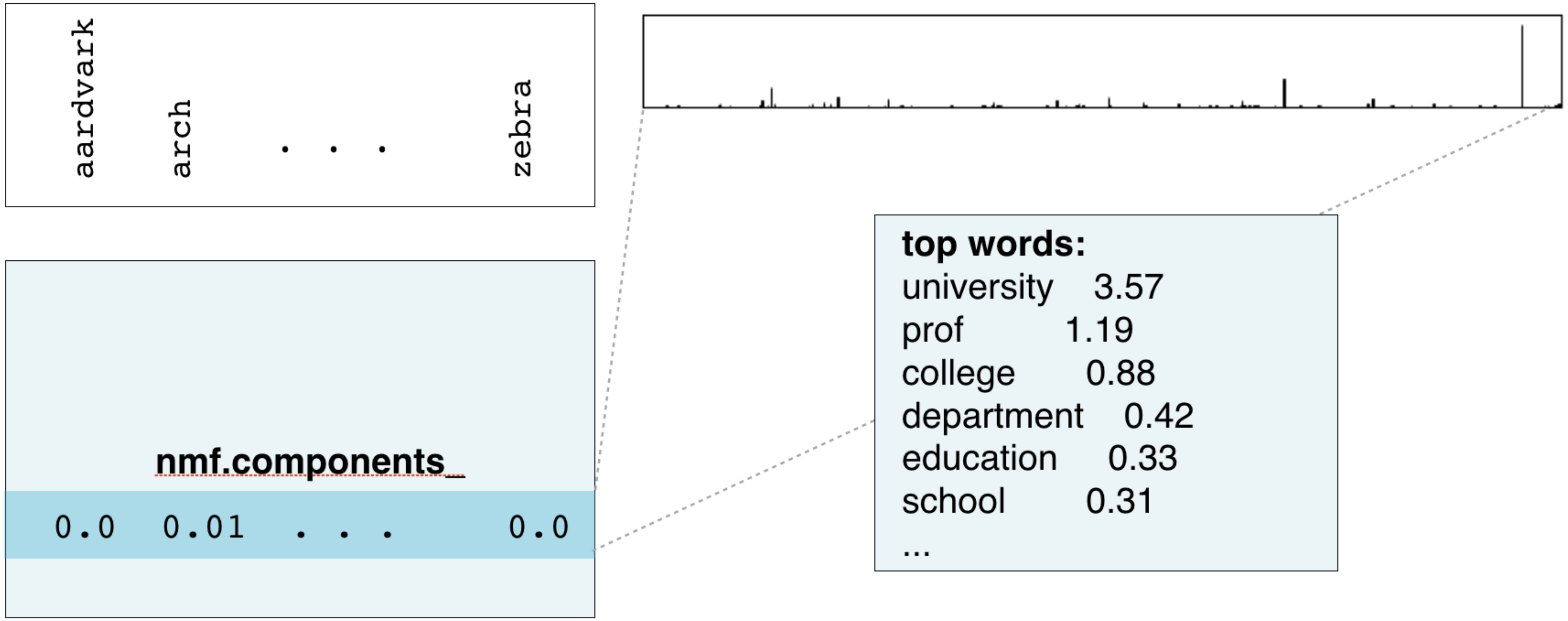
NMF components are topics



NMF components are topics

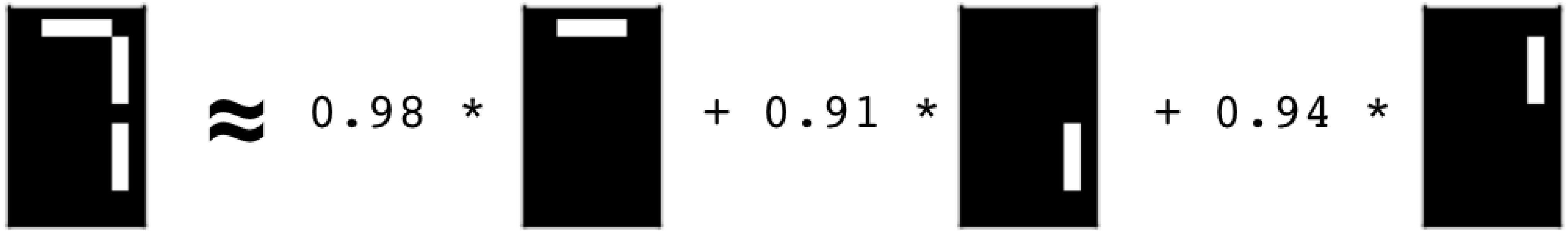


NMF components are topics



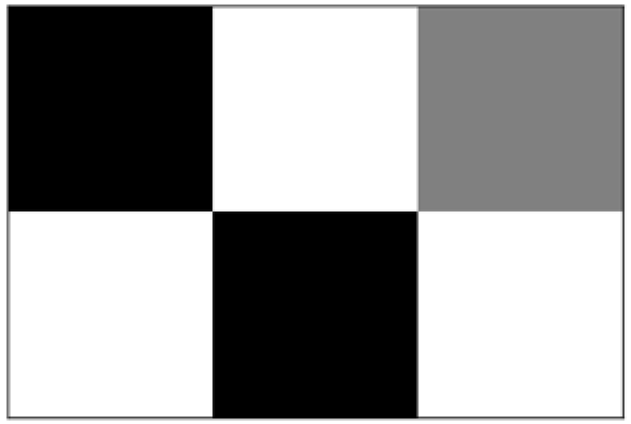
NMF components

- For documents:
  - NMF components represent topics
  - NMF features combine topics into documents
- For images, NMF components are parts of images



Grayscale images

- "Grayscale" image = no colors, only shades of gray
- Measure pixel brightness
- Represent with value between 0 and 1 (0 is black)
- Convert to 2D array



```
[[ 0.   1.   0.5]
 [ 1.   0.   1. ]]
```

Grayscale image example

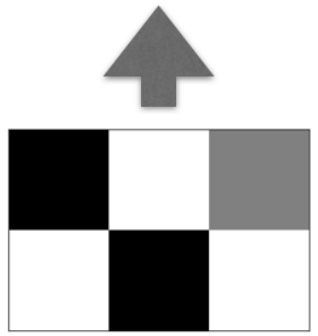
- An 8×8 grayscale image of the moon, written as an array

```
[[ 0.   0.   0.   0.   0.   0.   0.   0. ]
 [ 0.   0.   0.   0.7  0.8  0.   0.   0. ]
 [ 0.   0.   0.8  0.8  0.9  1.   0.   0. ]
 [ 0.   0.7  0.9  0.9  1.   1.   1.   0. ]
 [ 0.   0.8  0.9  1.   1.   1.   1.   0. ]
 [ 0.   0.   0.9  1.   1.   1.   0.   0. ]
 [ 0.   0.   0.   0.9  1.   0.   0.   0. ]
 [ 0.   0.   0.   0.   0.   0.   0.   0. ]]
```

Grayscale images as flat arrays

- Enumerate the entries
- Row-by-row
- From left to right, top to bottom

```
[[ 0.   1.   0.5]
 [ 1.   0.   1. ]]
```

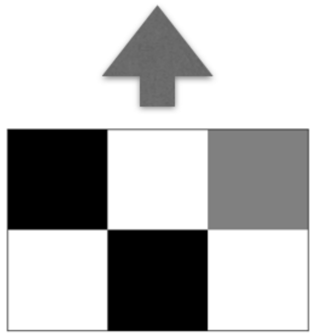


Grayscale images as flat arrays

- Enumerate the entries
- Row-by-row
- From left to right, top to bottom

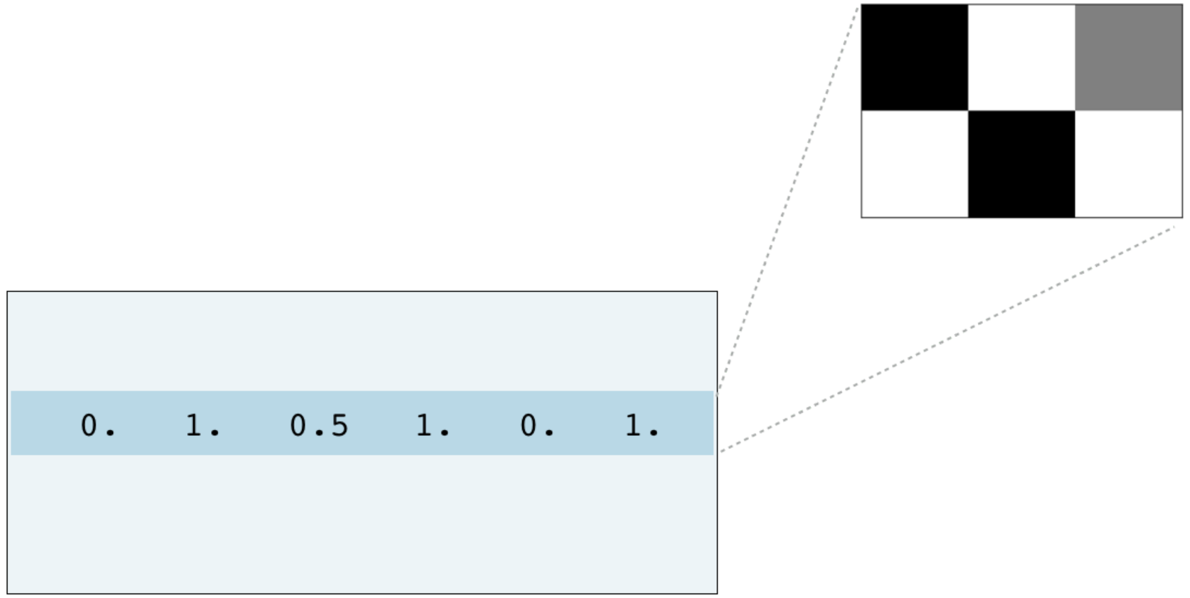
```
[[ 0.   1.   0.5]
 [ 1.   0.   1. ]]
```

```
[ 0.   1.   0.5   1.   0.   1. ]
```



Encoding a collection of images

- Collection of images of the same size
- Encode as 2D array
- Each row corresponds to an image
- Each column corresponds to a pixel
- ... can apply NMF!



Visualizing samples

```
print(sample)
```

```
[ 0.   1.   0.5  1.   0.   1. ]
```

```
bitmap = sample.reshape((2, 3))
print(bitmap)
```

```
[[ 0.   1.   0.5]
 [ 1.   0.   1. ]]
```

```
from matplotlib import pyplot as plt
plt.imshow(bitmap, cmap='gray', interpolation='nearest')
plt.show()
```

Let's practice!

UNSUPERVISED LEARNING IN PYTHON

Building recommender systems using NMF

UNSUPERVISED LEARNING IN PYTHON

Benjamin Wilson
Director of Research at lateral.io

Finding similar articles

- Engineer at a large online newspaper
- Task: recommend articles similar to article being read by customer
- Similar articles should have similar topics

Strategy

- Apply NMF to the word-frequency array
- NMF feature values describe the topics
- ... so similar documents have similar NMF feature values
- Compare NMF feature values?

Apply NMF to the word-frequency array

- `articles` is a word frequency array

```
from sklearn.decomposition import NMF
nmf = NMF(n_components=6)
nmf_features = nmf.fit_transform(articles)
```

Strategy

- Apply NMF to the word-frequency array
- NMF feature values describe the topics
- ... so similar documents have similar NMF feature values
- Compare NMF feature values?

Versions of articles

- Different versions of the same document have same topic proportions
- ... exact feature values may be different!

strong version

Dog bites man!
Attack by terrible
canine leaves man
paralyzed...

Versions of articles

- Different versions of the same document have same topic proportions
- ... exact feature values may be different!
- E.g. because one version uses many meaningless words
-

strong version

Dog bites man!
Attack by terrible
canine leaves man
paralyzed...

weak version

You may have heard,
unfortunately it seems
that a dog has perhaps
bitten a man ...

Versions of articles

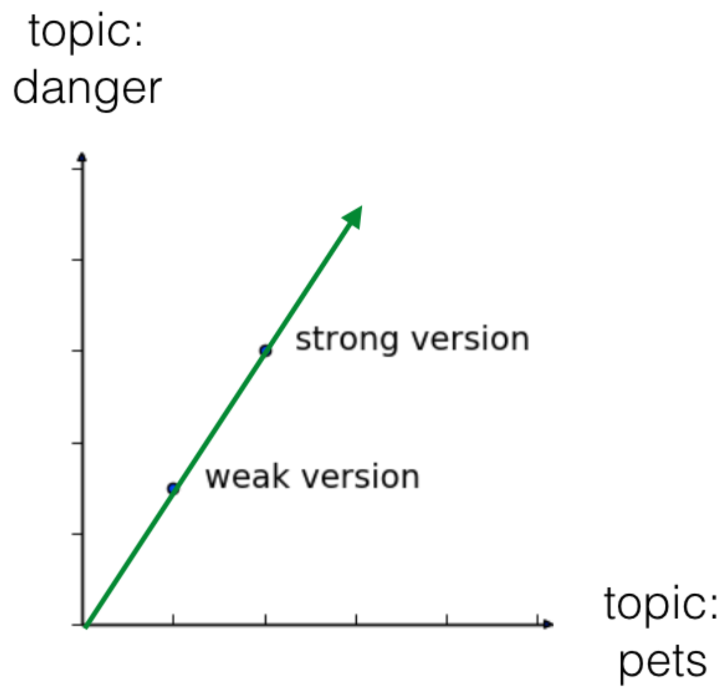
- Different versions of the same document have same topic proportions
- ... exact feature values may be different!
- E.g. because one version uses many meaningless words
- But all versions lie on the same line through the origin

strong version

Dog bites man! Attack by terrible canine leaves man paralyzed...

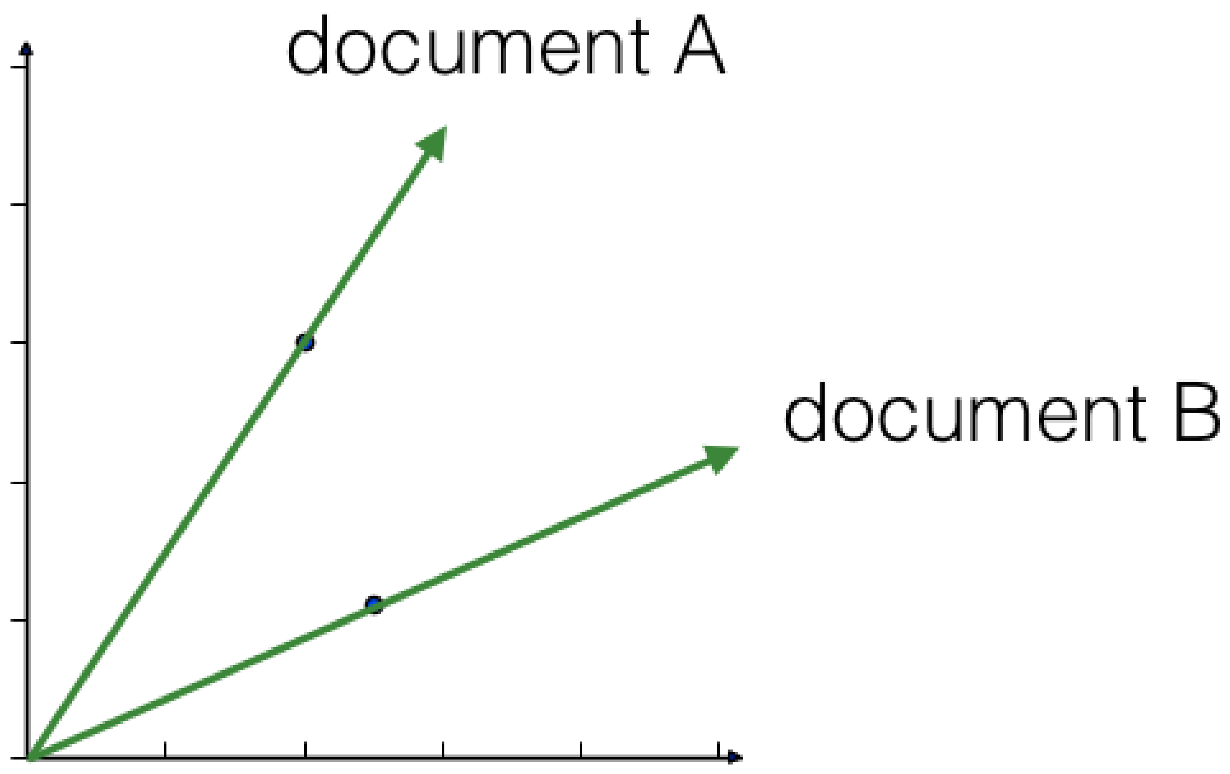
weak version

You may have heard, unfortunately it seems that a dog has perhaps bitten a man ...



Cosine similarity

- Uses the angle between the lines
- Higher values means more similar
- Maximum value is 1, when angle is 0 degrees



Calculating the cosine similarities

```
from sklearn.preprocessing import normalize
norm_features = normalize(nmf_features)
# if has index 23
current_article = norm_features[23,:]
similarities = norm_features.dot(current_article)
print(similarities)
```

```
[ 0.7150569   0.26349967 ..., 0.20323616  0.05047817]
```

DataFrames and labels

- Label similarities with the article titles, using a DataFrame
- Titles given as a list: `titles`

```
import pandas as pd
norm_features = normalize(nmf_features)
df = pd.DataFrame(norm_features, index=titles)
current_article = df.loc['Dog bites man']
similarities = df.dot(current_article)
```

DataFrames and labels

```
print(similarities.nlargest())
```

```
Dog bites man                             1.000000
Hound mauls cat                           0.979946
Pets go wild!                             0.979708
Dachshunds are dangerous                  0.949641
Our streets are no longer safe            0.900474
dtype: float64
```

Let's practice!

UNSUPERVISED LEARNING IN PYTHON

Final thoughts

UNSUPERVISED LEARNING IN PYTHON

Benjamin Wilson
Director of Research at lateral.io

Congratulations!

UNSUPERVISED LEARNING IN PYTHON