Week 4, Lecture 7 - Dimensionality Reduction: PCA and NMF

Aaron Meyer

Outline

- Administrative Issues
- Decomposition methods
  - Factor analysis
  - Principal components analysis
  - Non-negative matrix factorization

Dealing with many variables

- So far we've largely concentrated on cases in which we have relatively large numbers of measurements for a few variables
  - This is frequently refered to as $n > p$
- Two other extremes are imporant
  - Many observations and many variables
  - Many variables but few observations ($p > n$)

Dealing with many variables

Usually when we're dealing with many variables, we don't have a great understanding of how they relate to each other

- E.g. if gene X is high, we can't be sure that will mean gene Y will be too
- If we had these relationships, we could reduce the data
  - E.g. if we had variables to tell us it's 3 pm in Los Angeles, we don't need one to say it's daytime

Dimensionality Reduction

Generate a low-dimensional encoding of a high-dimensional space

Purposes:

- Data compression / visualization
- Robustness to noise and uncertainty
- Potentially easier to interpret

Bonus: Many of the other methods from the class can be applied after dimensionality reduction with little or no adjustment!

Matrix Factorization

Many (most?) dimensionality reduction methods involve matrix factorization

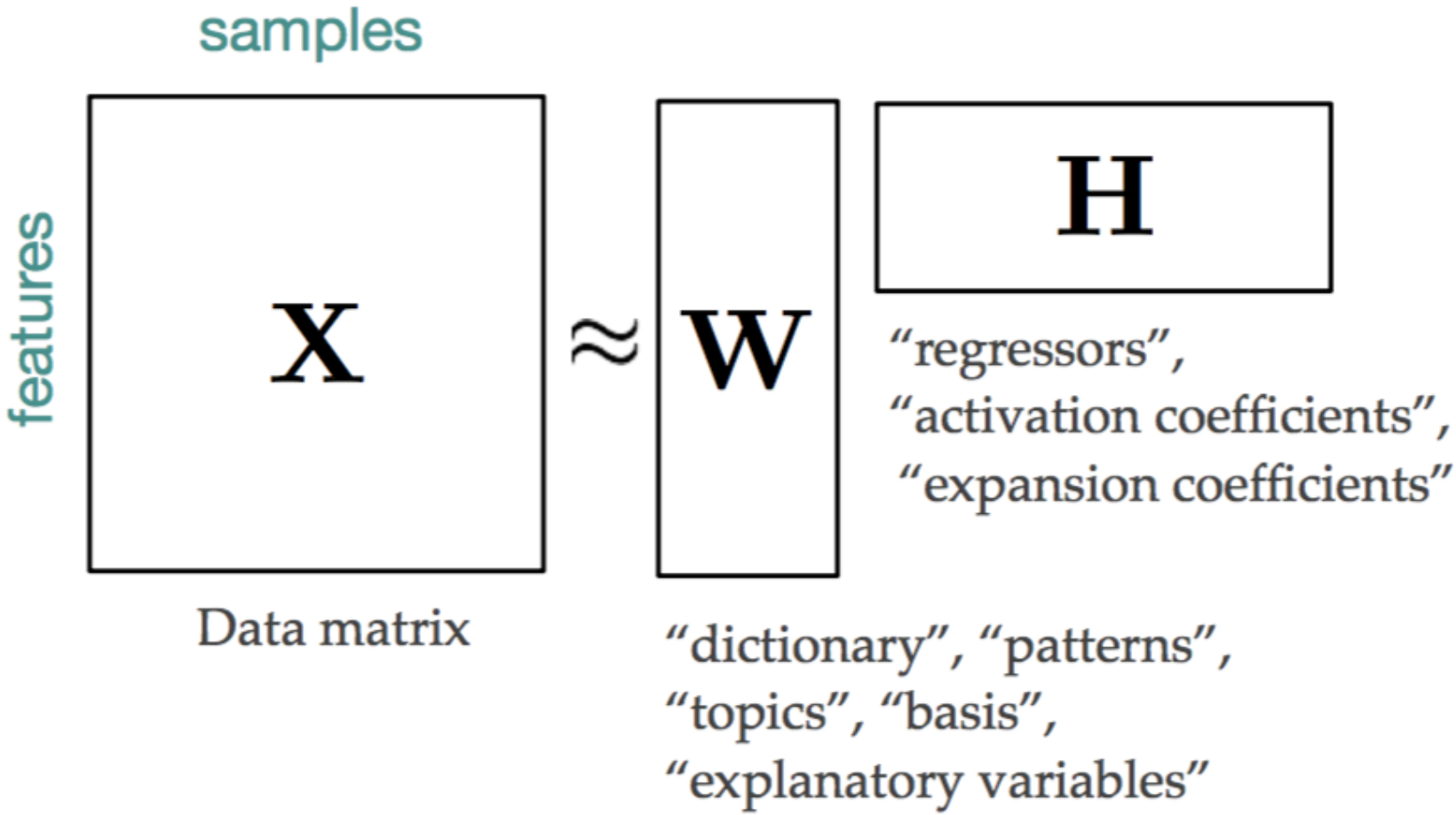
Basic Idea: Find two (or more) matrices whose product best approximate the original matrix

Low rank approximation to original $N \times M$ matrix:

$$\mathbf{X} \approx \mathbf{W}\mathbf{H}^T$$

where $\mathbf{W}$ is $N \times R$, $\mathbf{H}^T$ is $M \times R$, and $R \ll N$.

Matrix Factorization



Generalization of many methods (e.g., SVD, QR, CUR, Truncated SVD, etc.)

Aside - What should R be?

$$\mathbf{X} \approx \mathbf{W}\mathbf{H}^T$$

where $\mathbf{W}$ is $M \times R$, $\mathbf{H}^T$ is $M \times R$, and $R \ll N$.

Matrix Factorization

Matrix factorization is also compression

Figure: http://www.aaronschlegel.com/image-compression-principal-component-analysis/

Factor Analysis

Matrix factorization is also compression

Figure: http://www.aaronschlegel.com/image-compression-principal-component-analysis/

Factor Analysis

Matrix factorization is also compression

Figure: http://www.aaronschlegel.com/image-compression-principal-component-analysis/

Examples from bioengineering

Process control

- Large bioreactor runs may be recorded in a database, along with a variety of measurements from those runs
- We may be interested in how those different runs varied, and how each factor relates to one another
- Plotting a compressed version of that data can indicate when an anomolous change is present

Examples from bioengineering

Mutational processes

- Anytime multiple contributory factors give rise to a phenomena, matrix factorization can separate them out
- Will talk about this in greater detail

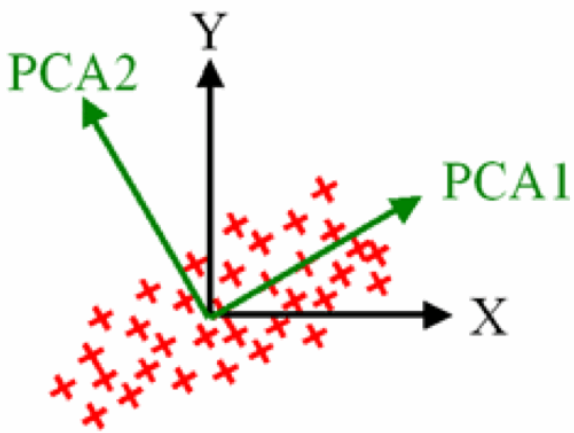
Cell heterogeneity

- Enormous interest in understanding how cells are similar or different
- Answer to this can be in millions of different ways
- But cells often follow *programs*

Principal Components Analysis

Application of matrix factorization

- Each principal component (PC) is linear combination of **uncorrelated** attributes / features'
- Ordered in terms of variance
- kth PC is orthogonal to all previous PCs
- Reduce dimensionality while maintaining maximal variance



Principal Components Analysis

BOARD

Methods to calculate PCA

- Iterative computation
  - More robust with high numbers of variables
  - Slower to calculate
- NIPALS (Non-linear iterative partial least squares)
  - Able to efficiently calculate a few PCs at once
  - Breaks down for high numbers of variables (large p)

Practical Notes

PCA

- Implemented within `sklearn.decomposition.PCA`
  - `PCA.fit_transform(X)` fits the model to `X`, then provides the data in principal component space
  - `PCA.components_` provides the "loadings matrix", or directions of maximum variance
  - `PCA.explained_variance_` provides the amount of variance explained by each component

PCA

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA

iris = datasets.load_iris()

X = iris.data
y = iris.target
target_names = iris.target_names

pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)

# Print PC1 loadings
print(pca.components_[:, 0])
# ...
```

PCA

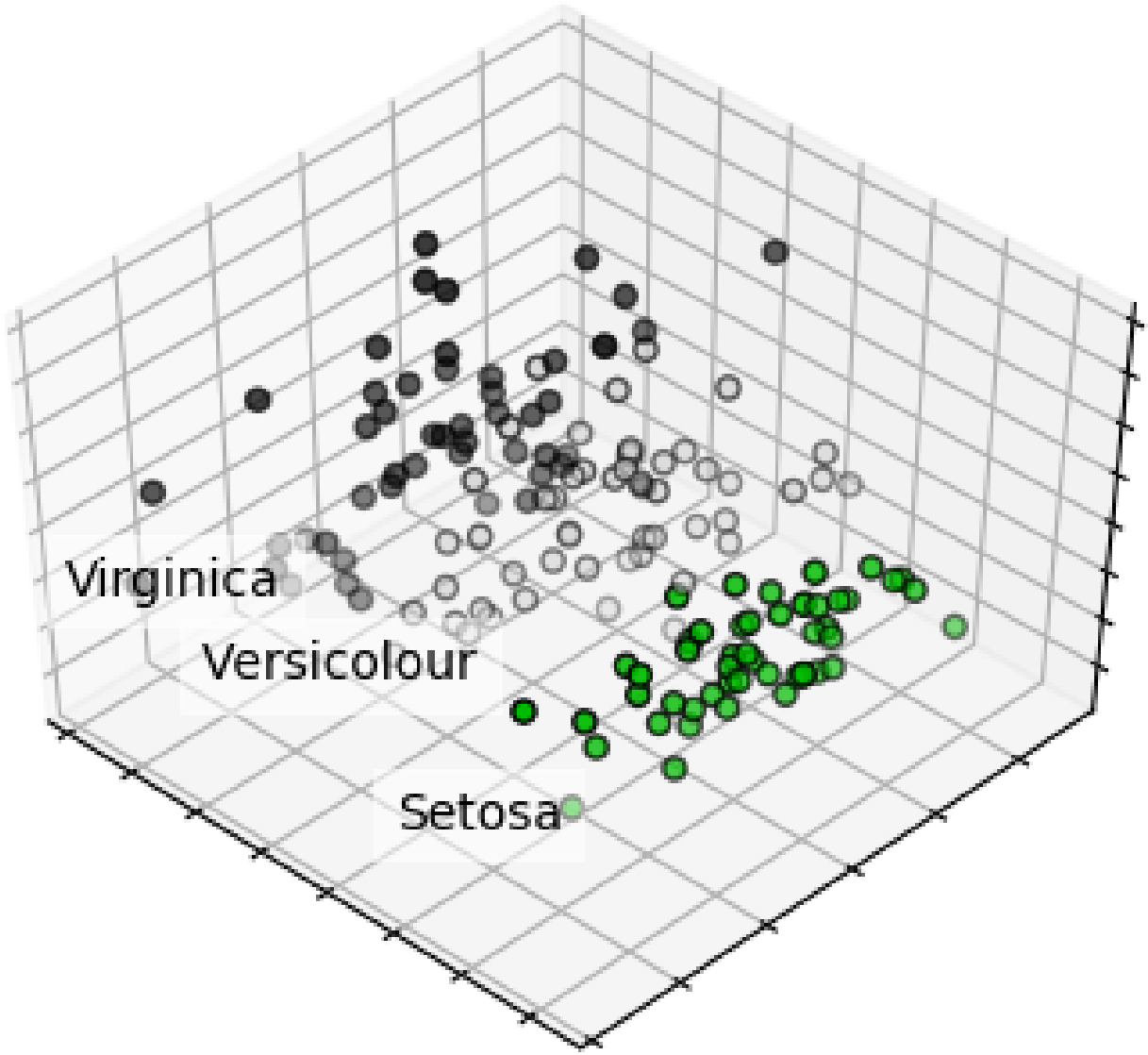
```
# ...
pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)

# Print PC1 loadings
print(pca.components_[:, 0])

# Print PC1 scores
print(X_r[:, 0])

# Percentage of variance explained for each component
print(pca.explained_variance_ratio_)
# [ 0.92461621  0.05301557]
```

PCA



Non-negative matrix factorization

Like PCA, except the coefficients in the linear combination must be non-negative

- Forcing positive coefficients implies an additive combination of basis parts to reconstruct whole
- Generally leads to zeros for factors that don't contribute

Non-negative matrix factorization

The answer you get will always depend on the error metric, starting point, and search method

BOARD

What is significant about this?

- The update rule is multiplicative instead of additive
- In the initial values for W and H are non-negative, then W and H can never become negative
- This guarantees a non-negative factorization
- Will converge to a local maxima
  - Therefore starting point matters

Non-negative matrix factorization

The answer you get will always depend on the error metric, starting point, and search method

- Another approach is to find the gradient across all the variables in the matrix
- Called coordinate descent, and is usually faster
- Not going to go through implementation
- Will also converge to a local maxima

NMF application: Netflix

Suppose Alice rates Inception 4 stars. **What led to this rating?**

NMF application: Netflix

480,000 users x 17,700 movies Test data: most recent ratings

Training data

user	movie	date	score
1	21	5/7/02	1
1	213	8/2/04	5
2	345	3/6/01	4
2	123	5/1/05	4
2	768	7/15/02	3
3	76	1/22/01	5
4	45	8/3/00	4
5	568	9/10/05	1
5	342	3/5/03	2
5	234	12/28/00	2
6	76	8/11/02	5
6	56	6/15/03	4

Test data

user	movie	date	score
1	62	1/6/05	?
1	96	9/13/04	?
2	7	8/18/05	?
2	3	11/22/05	?
3	47	6/13/02	?
3	15	8/12/01	?
4	41	9/1/00	?
4	28	8/27/05	?
5	93	4/4/05	?
5	74	7/16/03	?
6	69	2/14/04	?
6	83	10/3/03	?

What do you think is done with missing values?

NMF application: Netflix

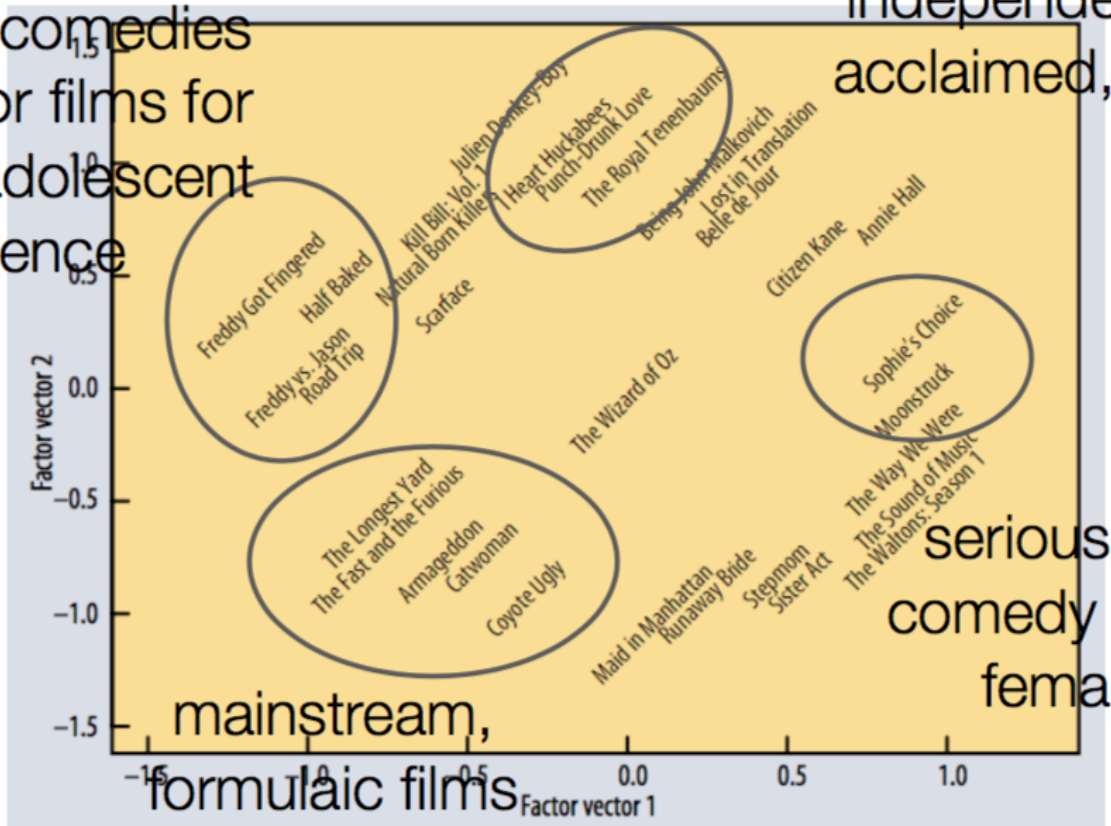
- More generally, additional baseline predictors include:
  - A factor that allows Alice's rating to (linearly) depend on the (square root of the) number of days since her first rating. (For example, have you ever noticed that you become a harsher critic over time?)
  - A factor that allows Alice's rating to depend on the number of days since the movie's first rating by anyone. (If you're one of the first people to watch it, maybe it's because you're a huge fan and really excited to see it on DVD, so you'll tend to rate it higher.)
  - A factor that allows Alice's rating to depend on the number of people who have rated Inception. (Maybe Alice is a hipster who hates being part of the crowd.)
  - A factor that allows Alice's rating to depend on the movie's overall rating.
  - (Plus a bunch of others.)

NMF application: Netflix

And, in fact, modeling these biases turned out to be fairly important: in their paper describing their final solution to the Netflix Prize, Bell and Koren write that:

Of the numerous new algorithmic contributions, I would like to highlight one – those humble baseline predictors (or biases), which capture main effects in the data. While the literature mostly concentrates on the more sophisticated algorithmic aspects, we have learned that an accurate treatment of main effects is probably at least as signficant as coming up with modeling breakthroughs.

NMF application: Netflix



NMF application: Mutational Processes in Cancer

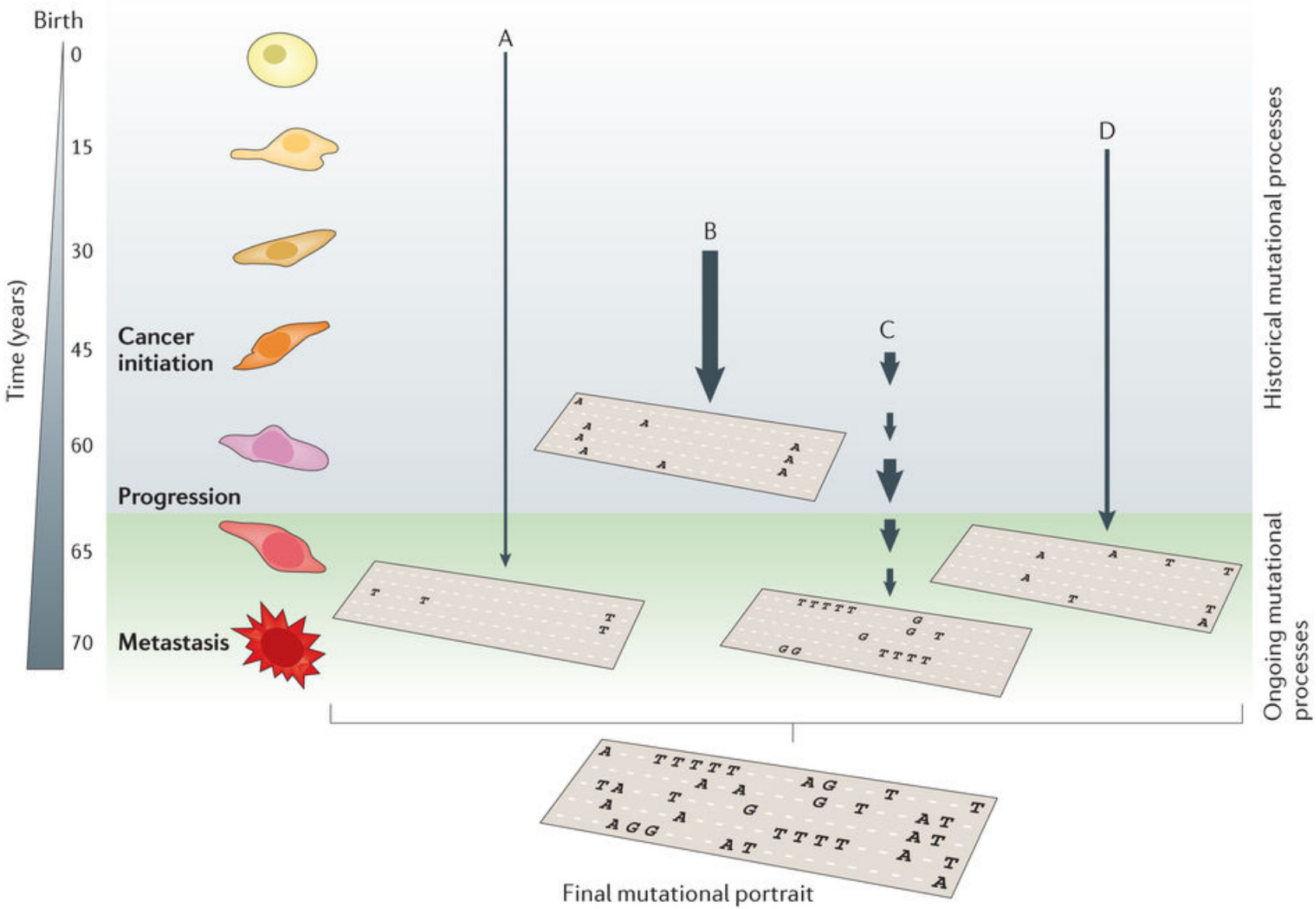


Figure: Helleday et al, Nat Rev Gen, 2014

NMF application: Mutational Processes in Cancer

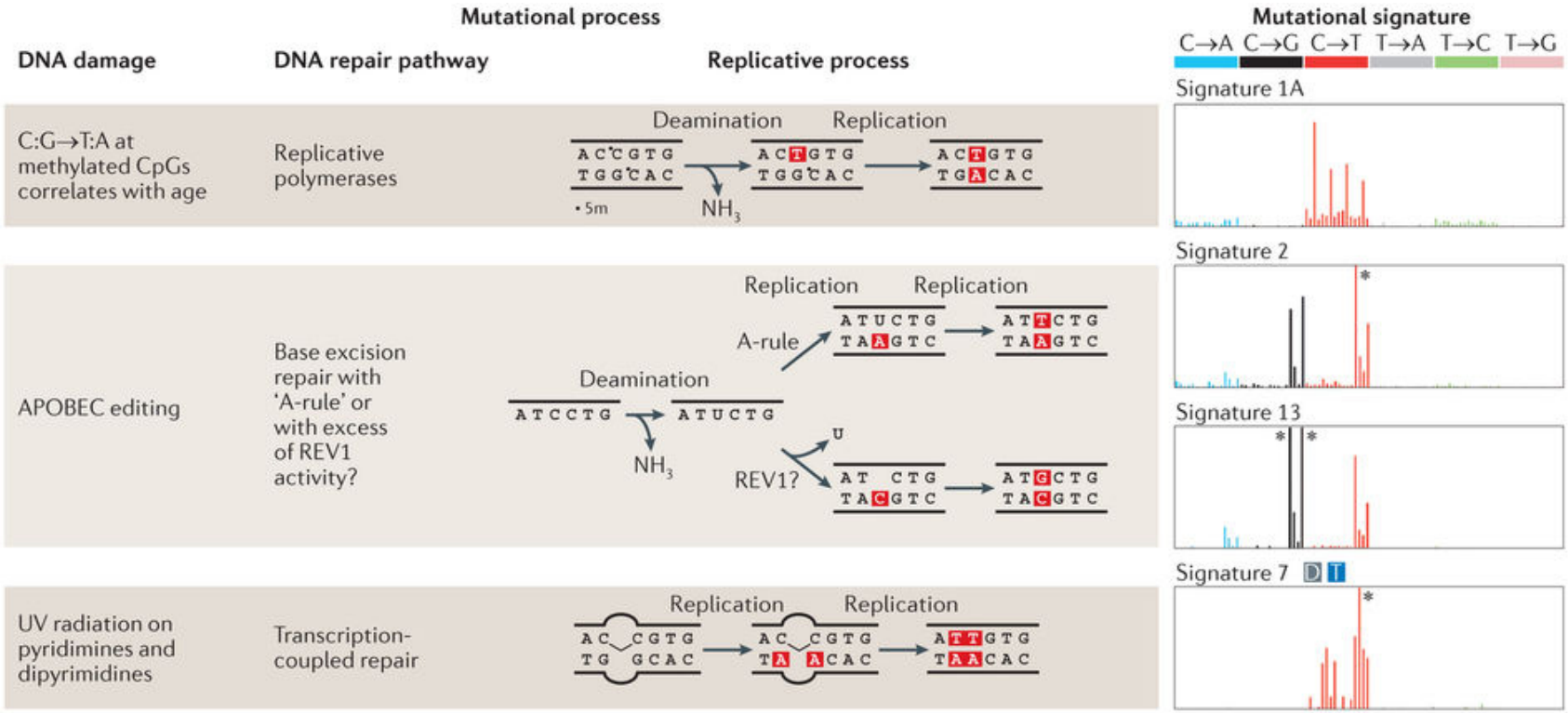


Figure: Helleday et al, Nat Rev Gen, 2014

NMF application: Mutational Processes in Cancer

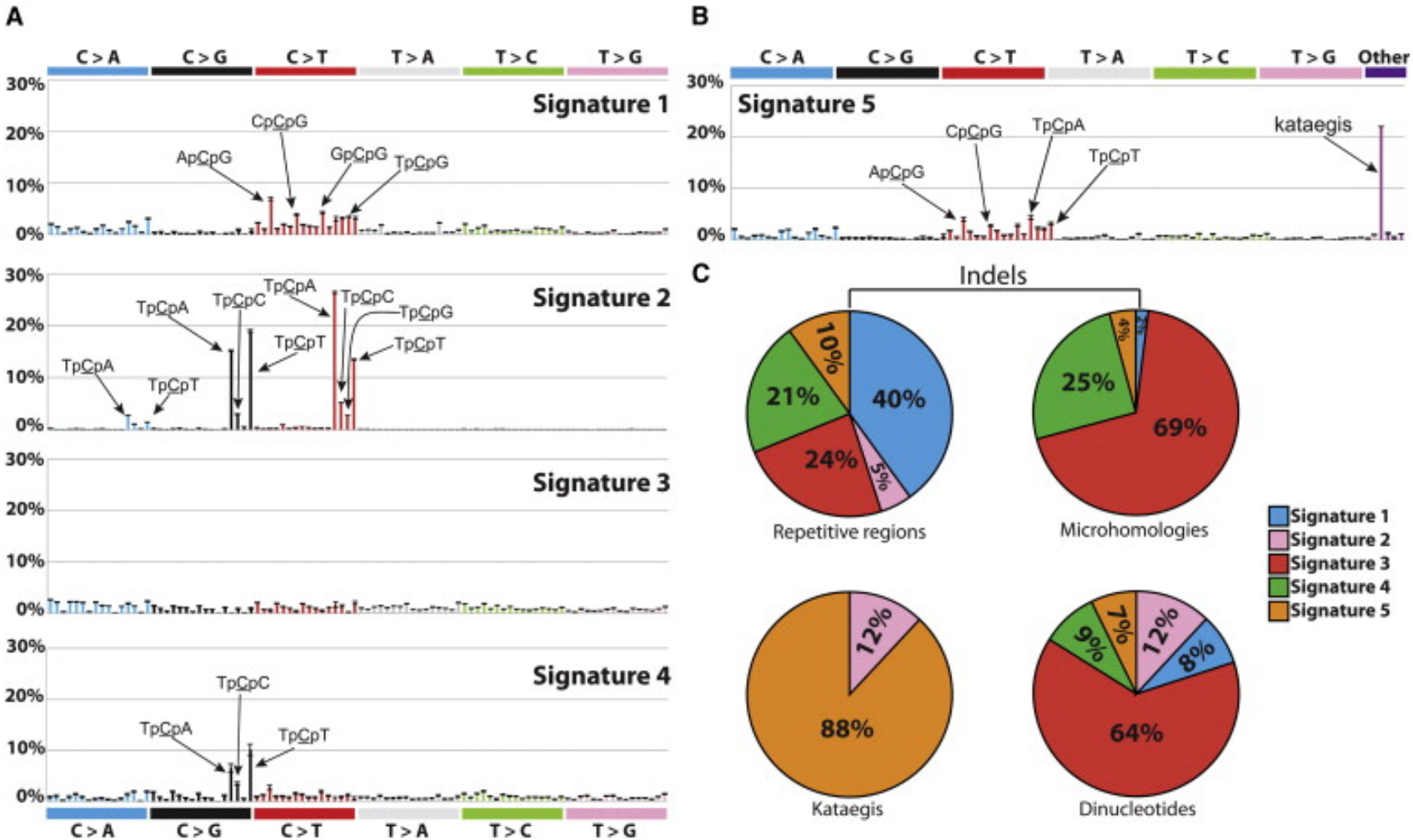


Figure: Alexandrov et al, Cell Rep, 2013

NMF application: Mutational Processes in Cancer

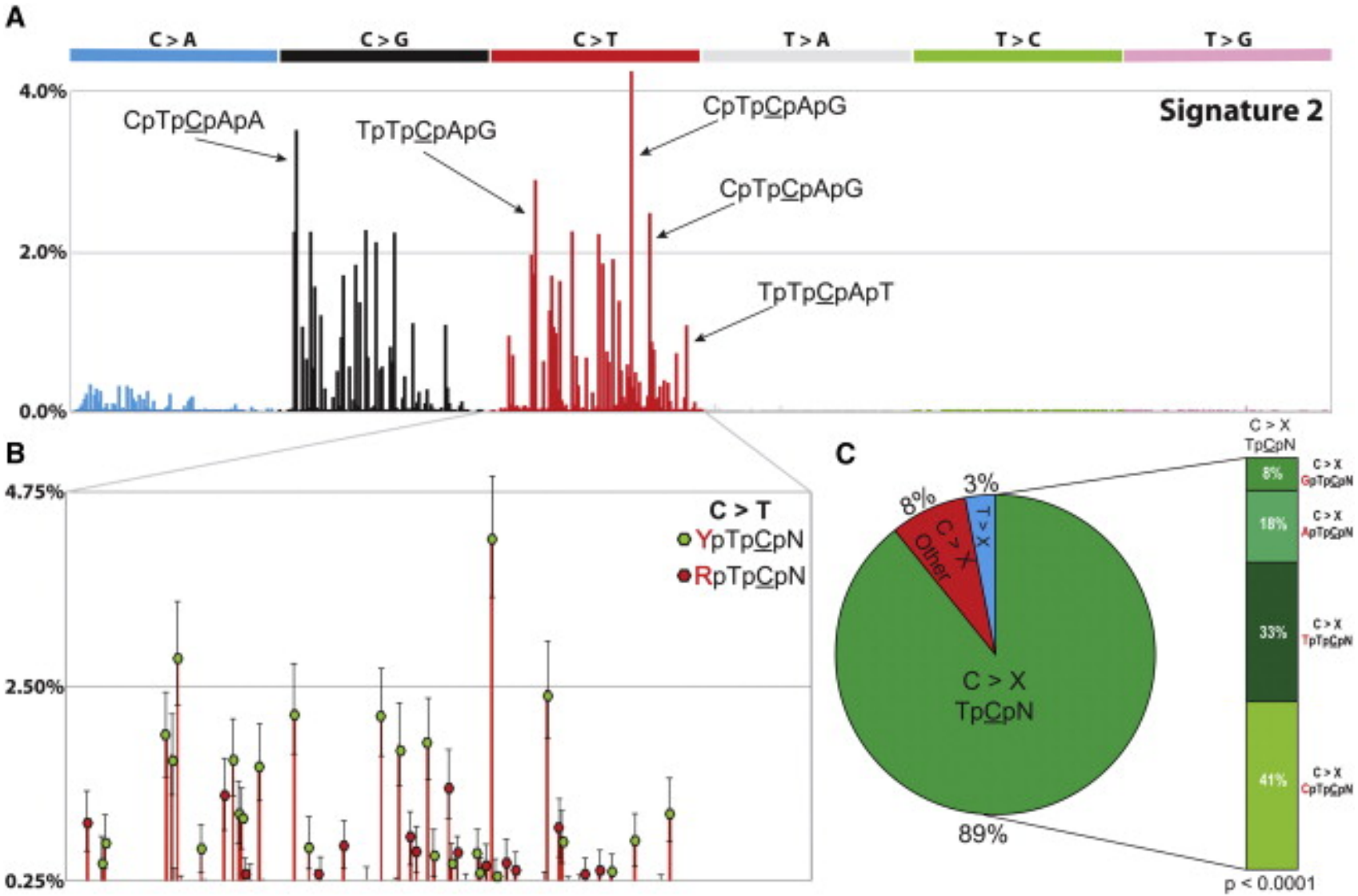


Figure: Alexandrov et al, Cell Rep, 2013

Practical Notes - NMF

- Implemented within `sklearn.decomposition.NMF`.
  - `n_components`: number of components
  - `init`: how to initialize the search
  - `solver`: 'cd' for coordinate descent, or 'mu' for multiplicative update
  - `l1_ratio`: Can regularize fit
- Provides:
  - `NMF.components_`: components x features matrix
  - Returns transformed data through `NMF.fit_transform()`

Summary

PCA

- Preserves the covariation within a dataset
- Therefore mostly preserves axes of maximal variation
- Number of components can vary—in practice more than 2 or 3 rarely helpful

NMF

- Explains the dataset through factoring into two **non-negative** matrices
- Much more stable and well-specified reconstruction when assumptions are appropriate
- Excellent for separating out additive factors

Closing

As always, selection of the appropriate method depends upon the question being asked.