Special Topics

PHYS 453

Dr Daugherity

ACU
ABILENE CHRISTIAN UNIVERSITY


ML-309:
UNSUPERVISED LEARNING
THEJENKINSCOMIC

Clustering

Problem #1

What can we do without training data?

Can we **learn** anything?

One goal is to look for natural clusters in unlabeled data. We can then group similar data, and even (kinda) classify new data as belonging to a certain group.

Clustering

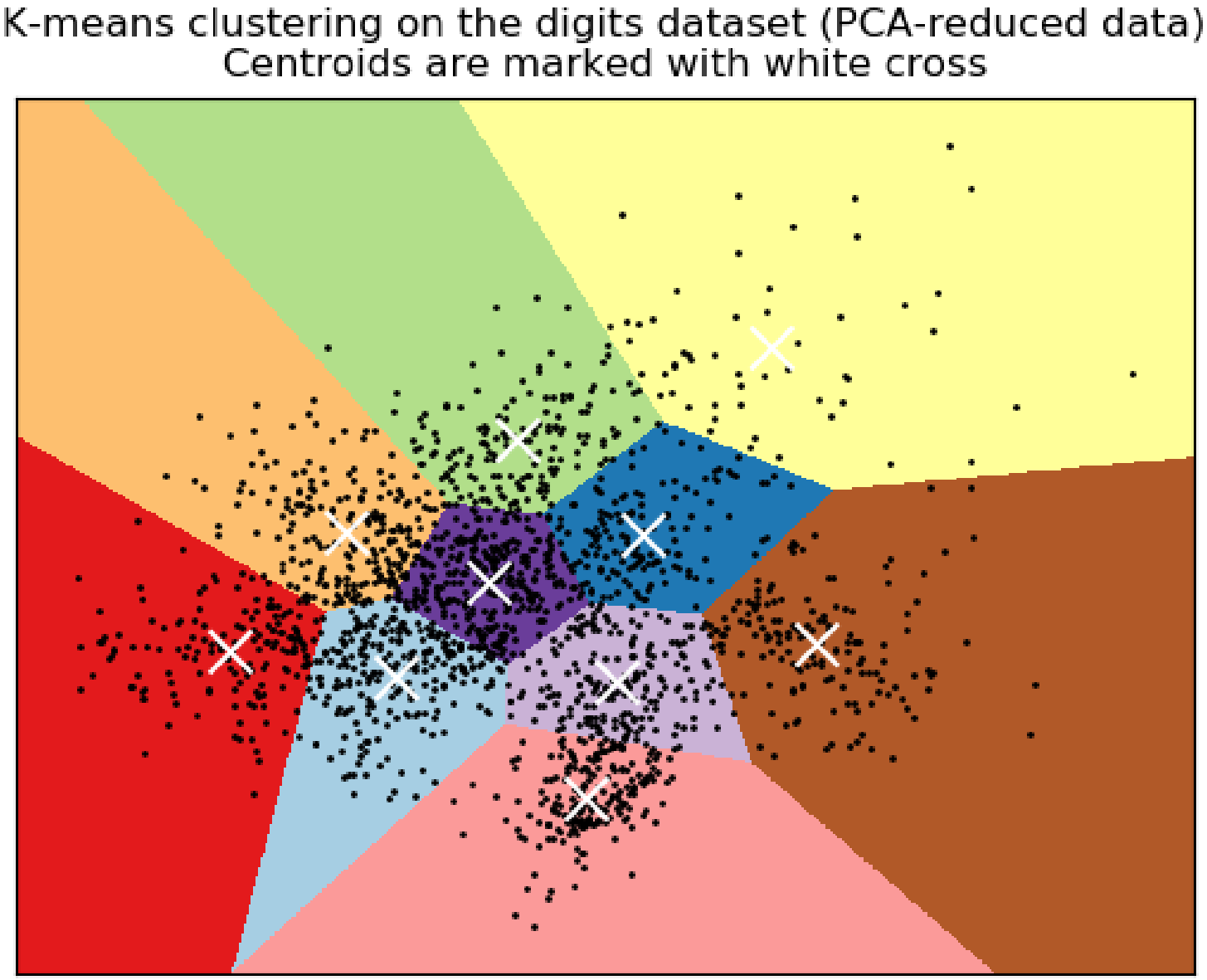
Clustering code

References:

- User's Guide http://scikit-learn.org/stable/modules/clustering.html
- Intro to ML Chapter 3: https://github.com/amueller/introduction_to_ml_with_python/blob/master/03-unsupervised-learning.ipynb
- Thoughtful ML: https://github.com/thoughtfulml/examples-in-python/tree/master/em-clustering

k Means Clustering

http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html



k Means Clustering

Algorithm:

choose k

randomly choose k data points for each cluster

REPEAT

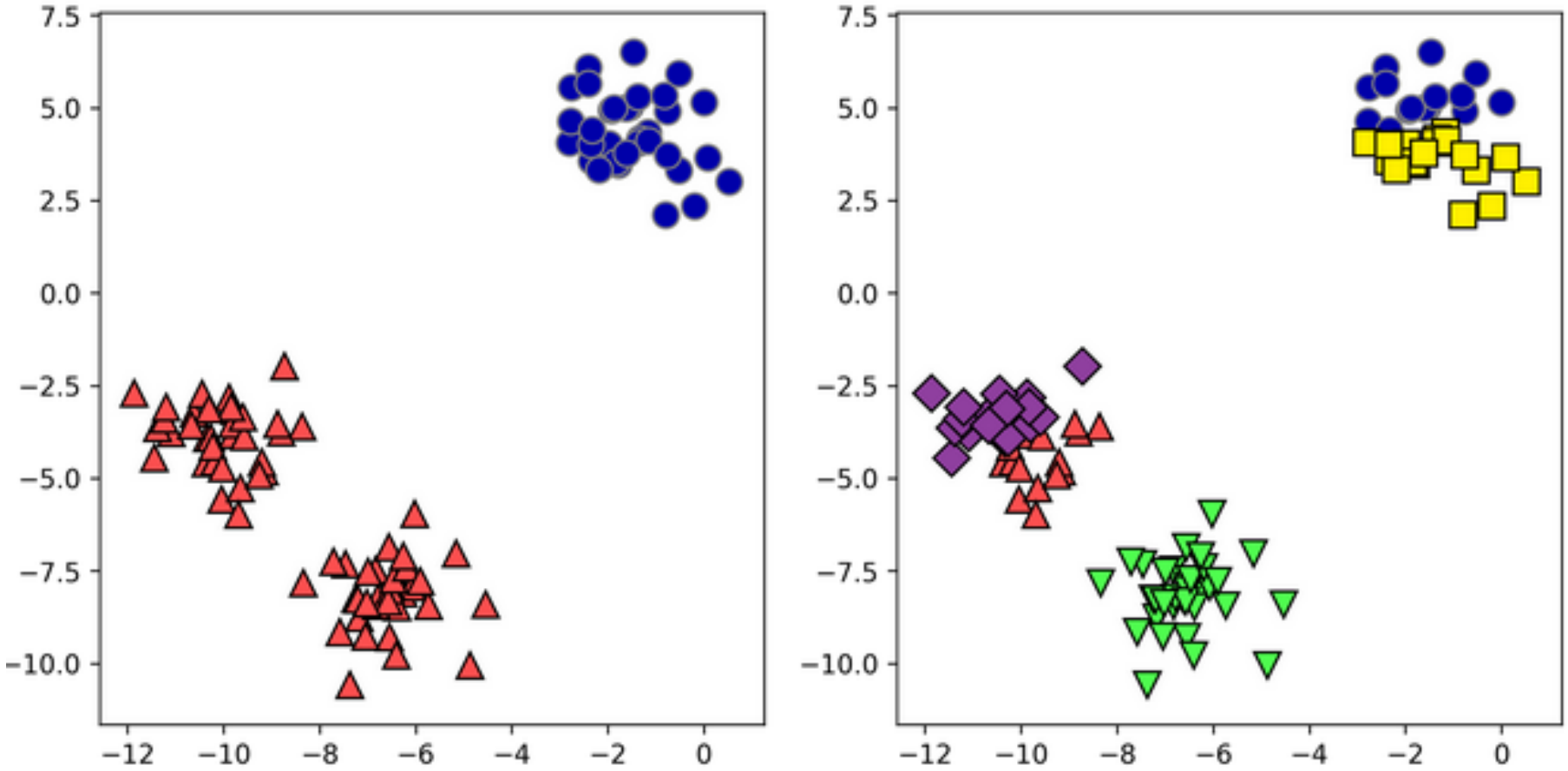
find mean of each cluster

assign every data point to nearest cluster

UNTIL DONE

k Means Clustering

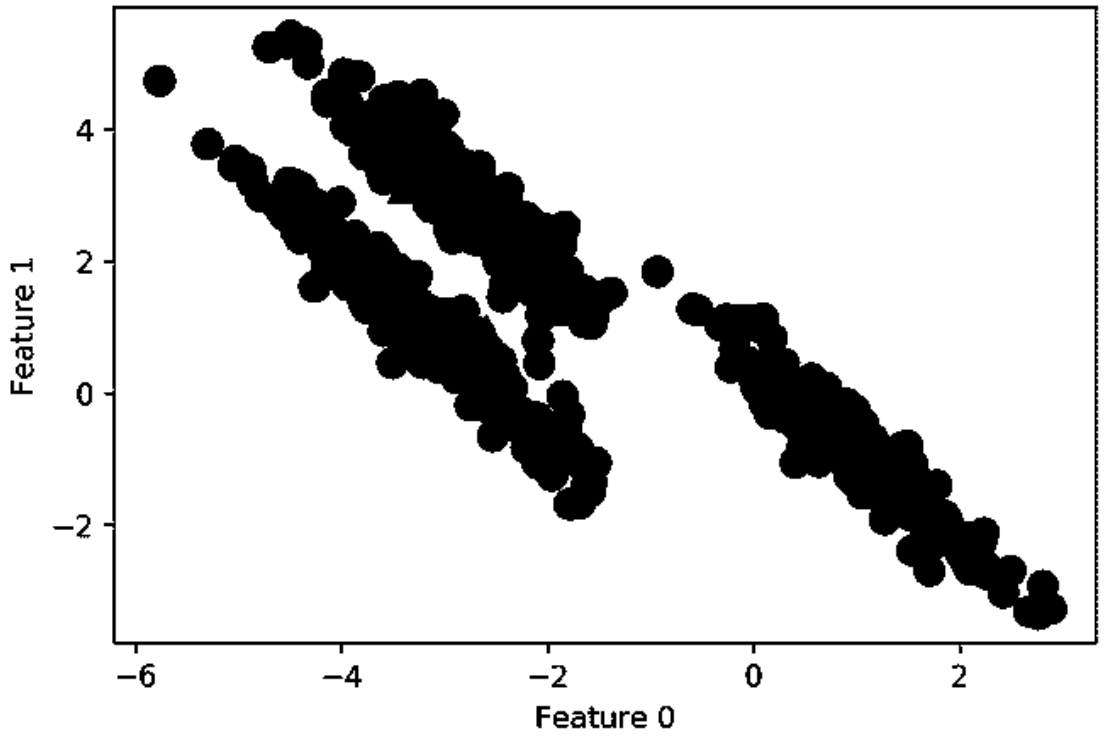
https://github.com/amueller/introduction_to_ml_with_python/blob/master/03-unsupervised-learning.ipynb

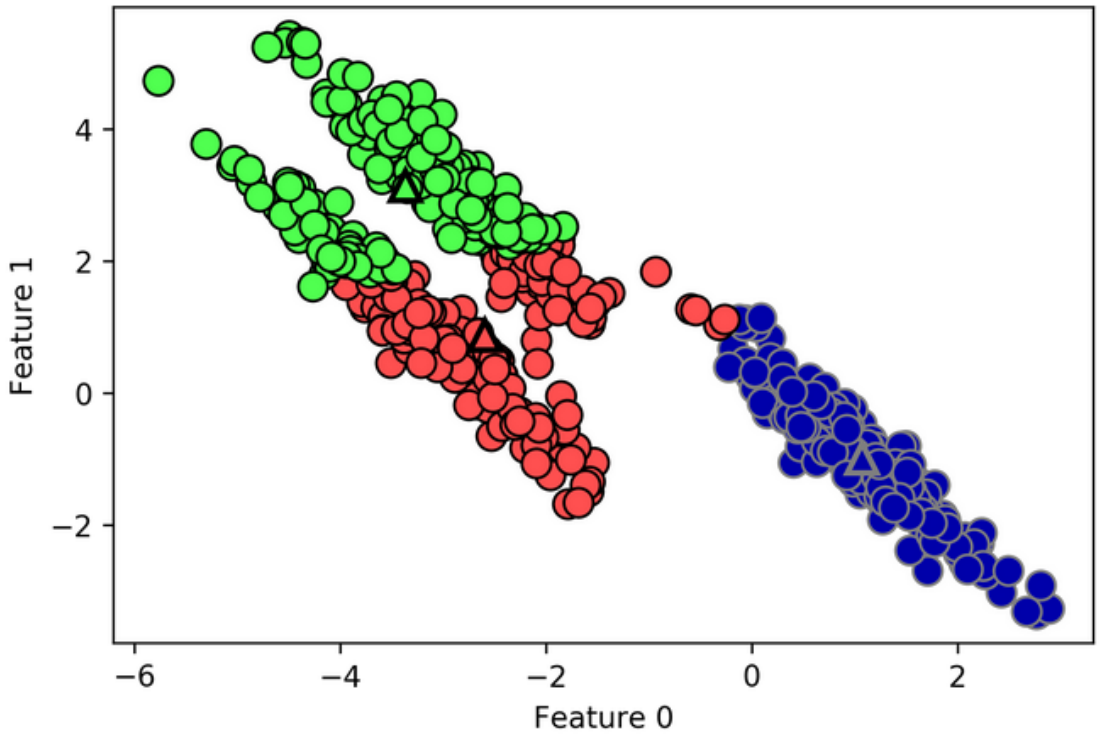


Wrong k values!

k Means Clustering

https://github.com/amueller/introduction_to_ml_with_python/blob/master/03-unsupervised-learning.ipynb

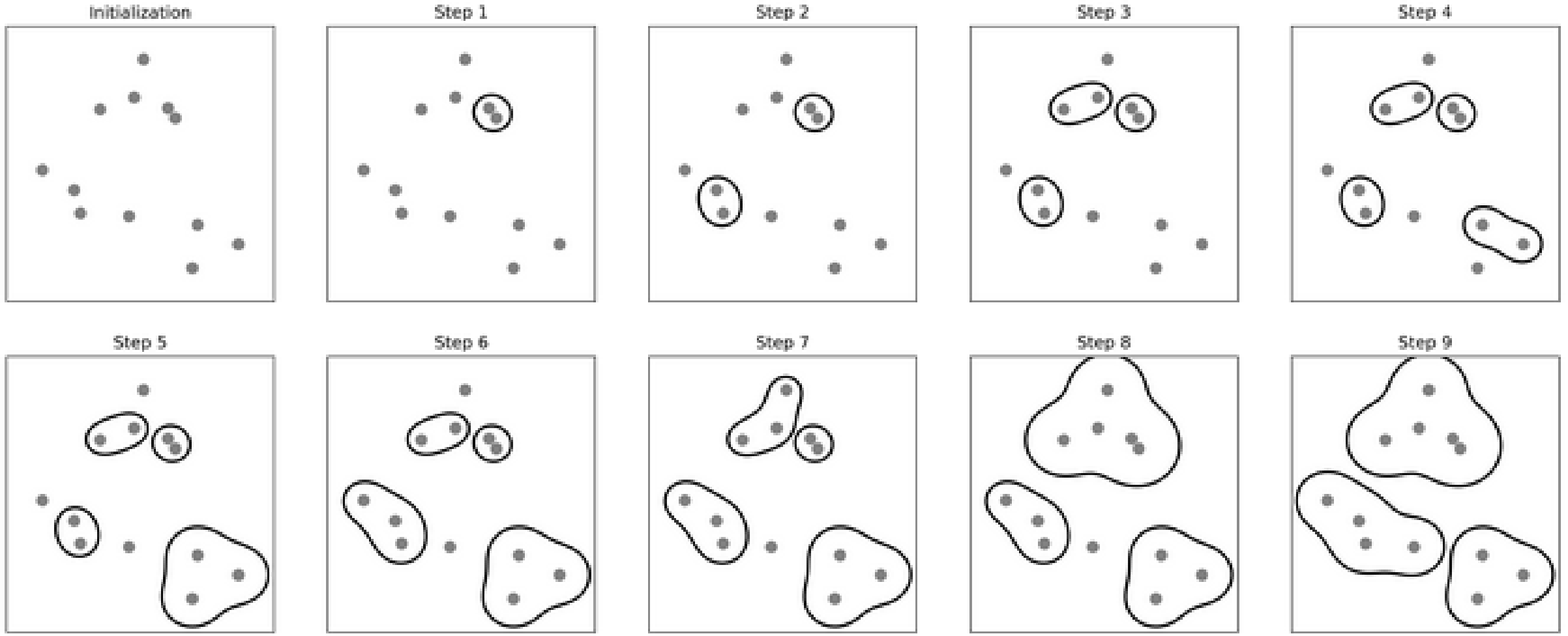




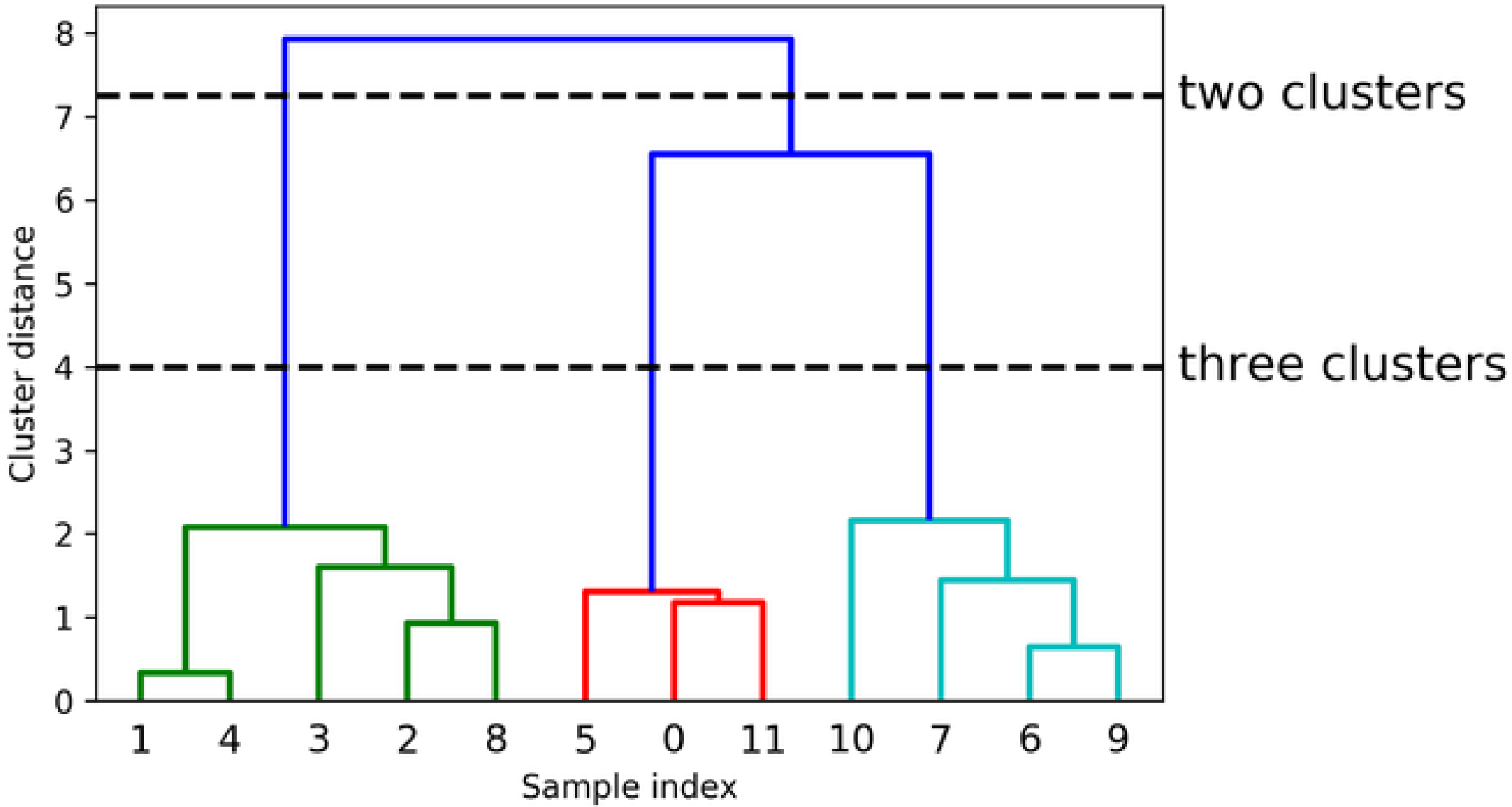
fails with non-spherical clusters

Agglomerative Clustering

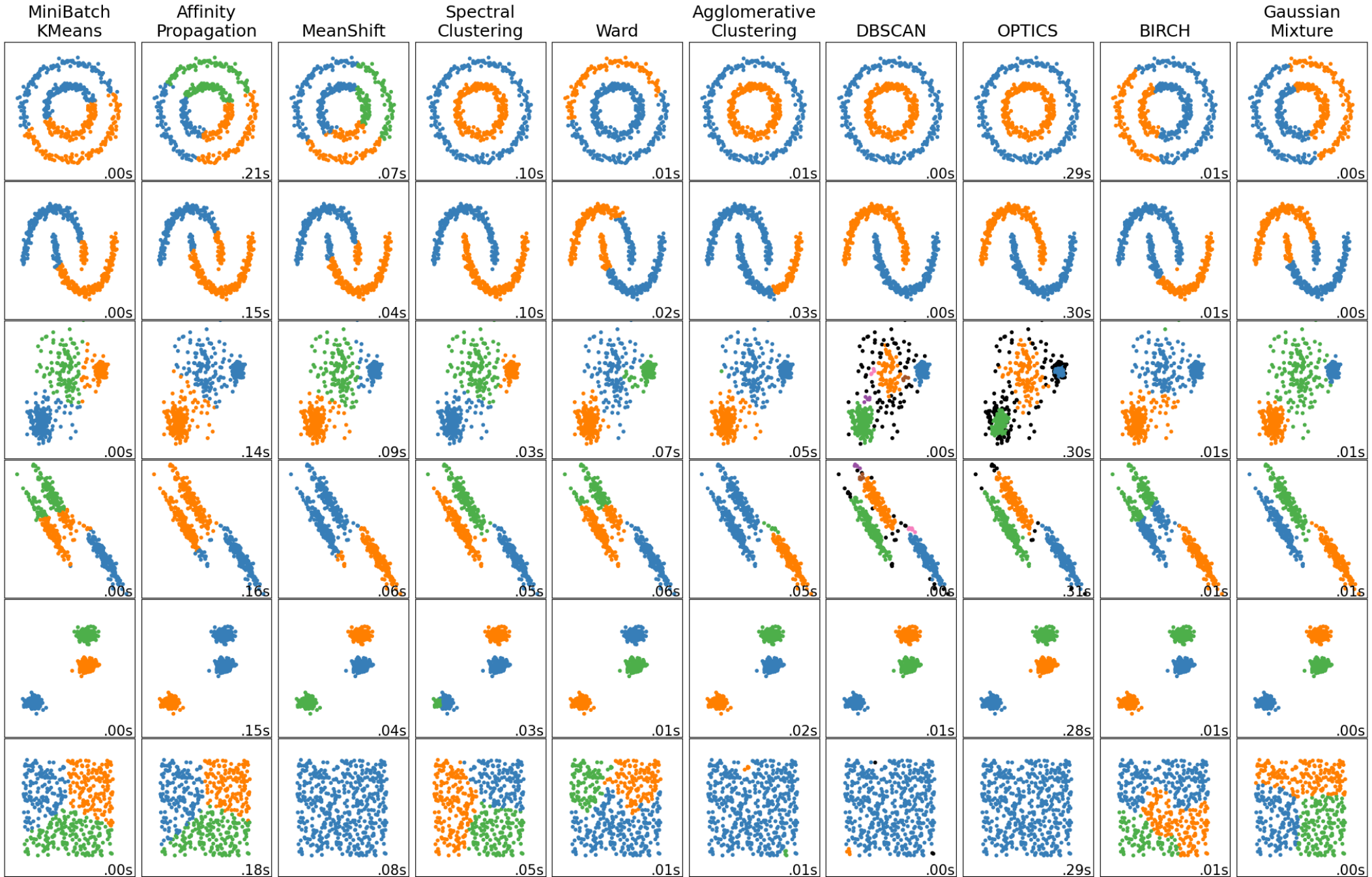
- Every point starts as its own cluster
- Merge nearby clusters until done.



Agglomerative Clustering



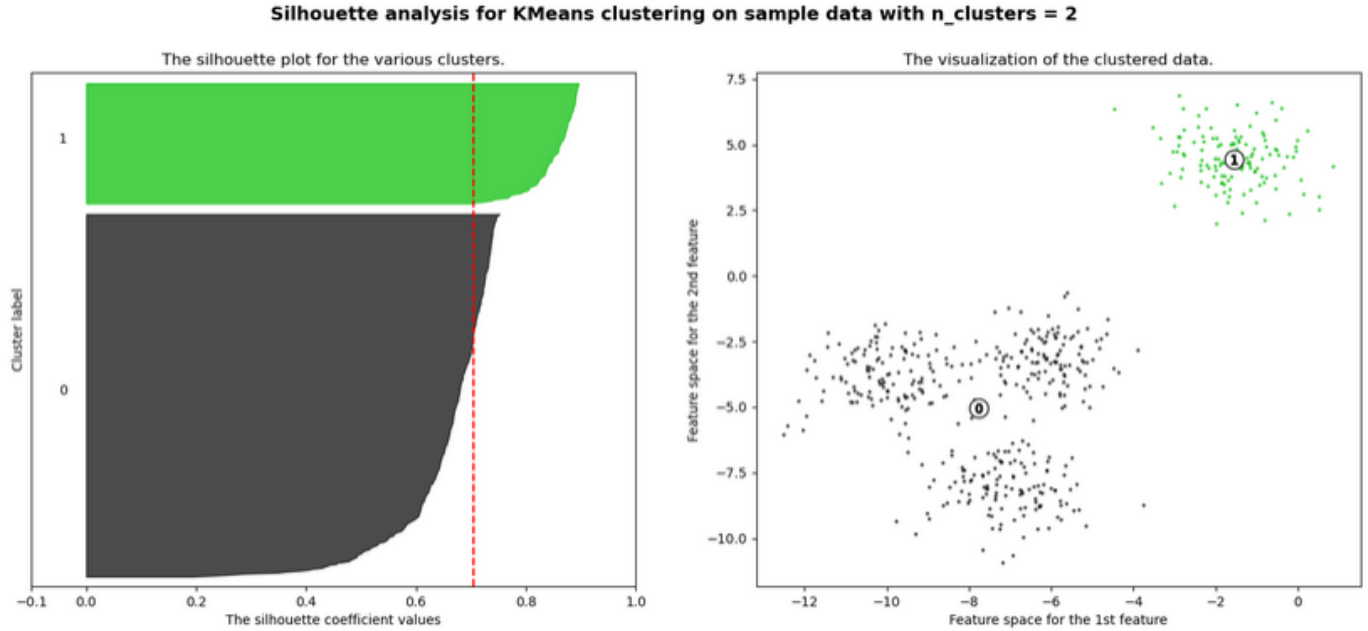
https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

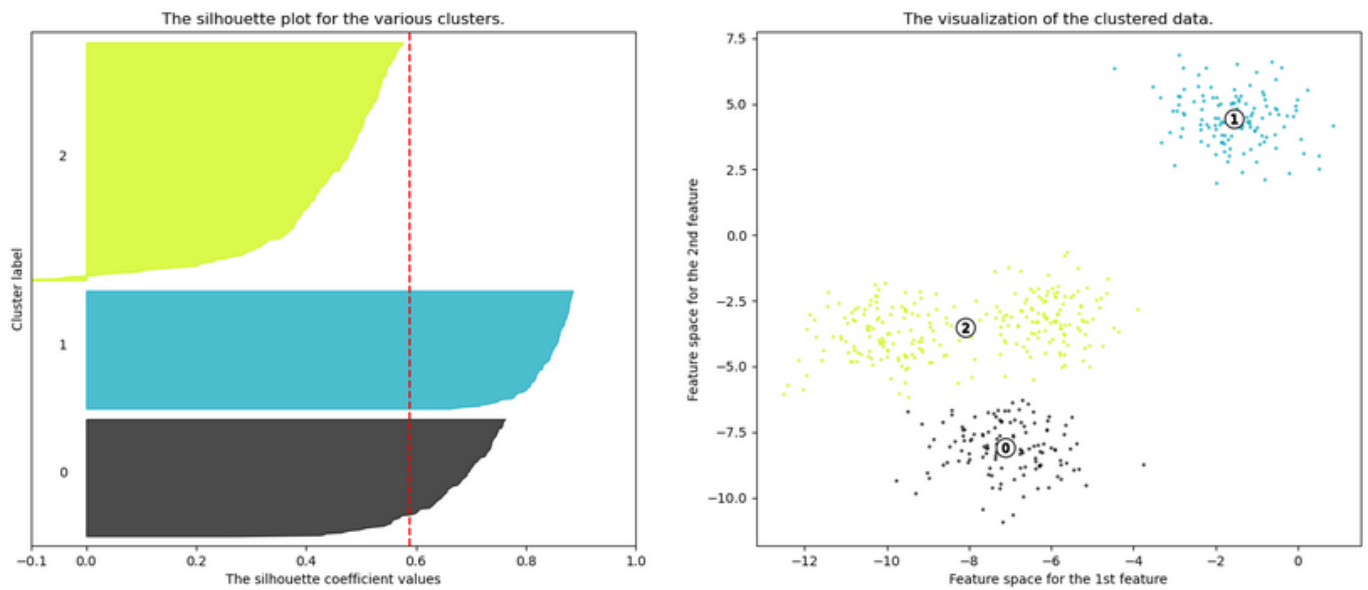


https://scikit-learn.org/stable/modules/clustering.html

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters, inductive	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry, inductive	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry, inductive	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry, transductive	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, transductive	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances, transductive	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes, outlier removal, transductive	Distances between nearest points
OPTICS	minimum cluster membership	Very large `n_samples`, large `n_clusters`	Non-flat geometry, uneven cluster sizes, variable cluster density, outlier removal, transductive	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation, inductive	Mahalanobis distances to centers
BIRCH	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction, inductive	Euclidean distance between points
Bisecting K-Means	number of clusters	Very large `n_samples`, medium `n_clusters`	General-purpose, even cluster size, flat geometry, no empty clusters, inductive, hierarchical	Distances between points

https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html#sphx-glr-auto-examples-cluster-plot-kmeans-silhouette-analysis-py





PCA

Problem #2

- Given a set of features, is it possible to optimize them **before** classifying?
- Equivalent to dimensionality reduction: can I get rid of features to “compress” the data?
- I could even make a 2D representation of any data…

There are a set of preprocessing tricks we can use

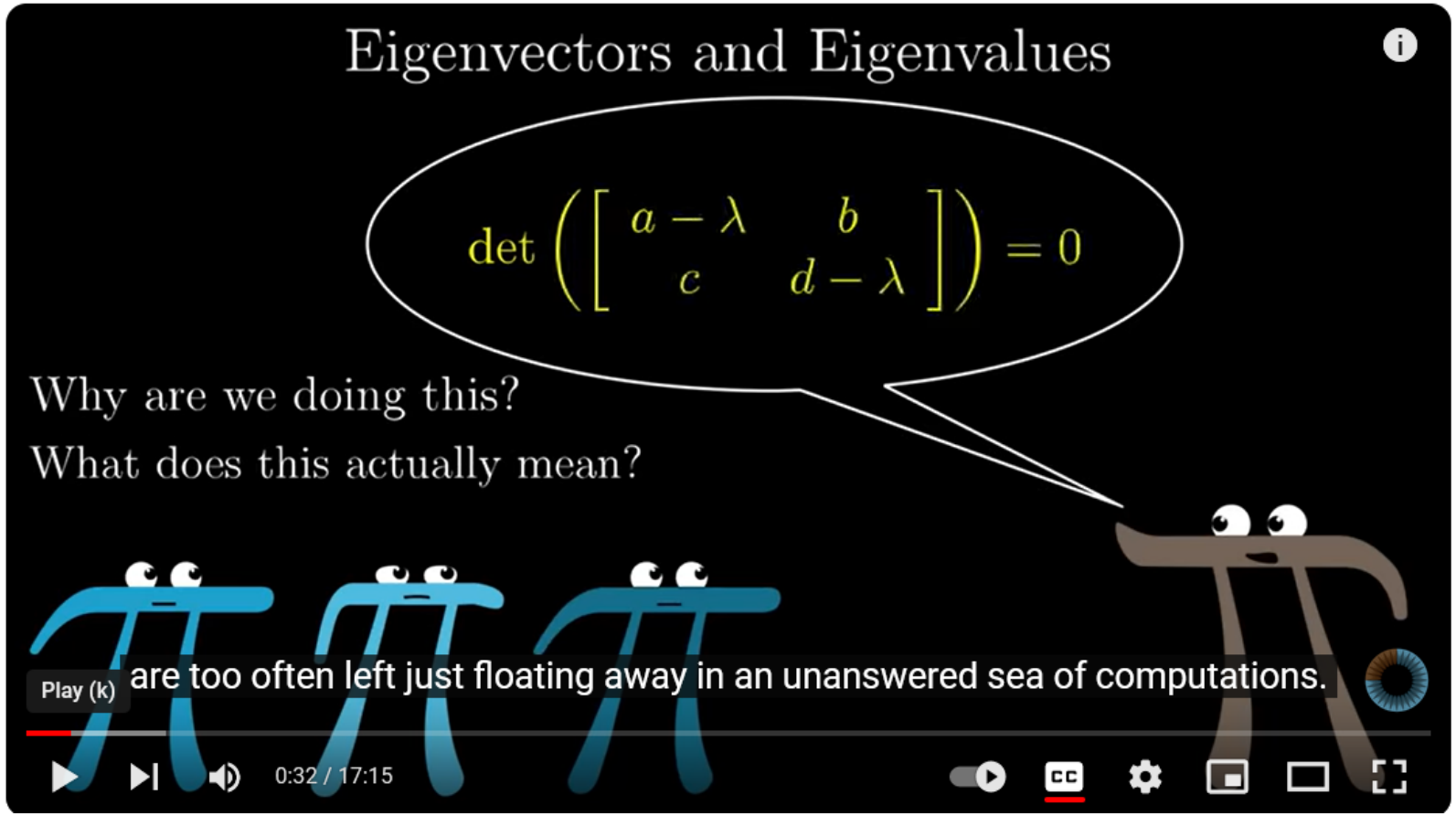
Principal Component Analysis

The algorithm is surprisingly easy:

1. normalize the data
2. construct covariance matrix
3. find eigenvalues and eigenvectors

- "Principal Components" are the eigenvectors in order from largest to smallest eigenvalues
- You can discard the less important eigenvectors to reduce the number of dimensions
- *Maximizes* local variance

https://www.youtube.com/watch?v=PFDu9oVAE-g



Eigenvectors and eigenvalues | Chapter 14, Essence of linear algebra

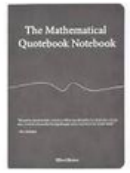
4.5M views 7 years ago 3Blue1Brown series S1 E14

A visual understanding of eigenvectors, eigenvalues, and the usefulness of an eigenbasis.
Help fund future projects: / 3blue1brown ...more

3Blue1Brown
6.13M subscribers

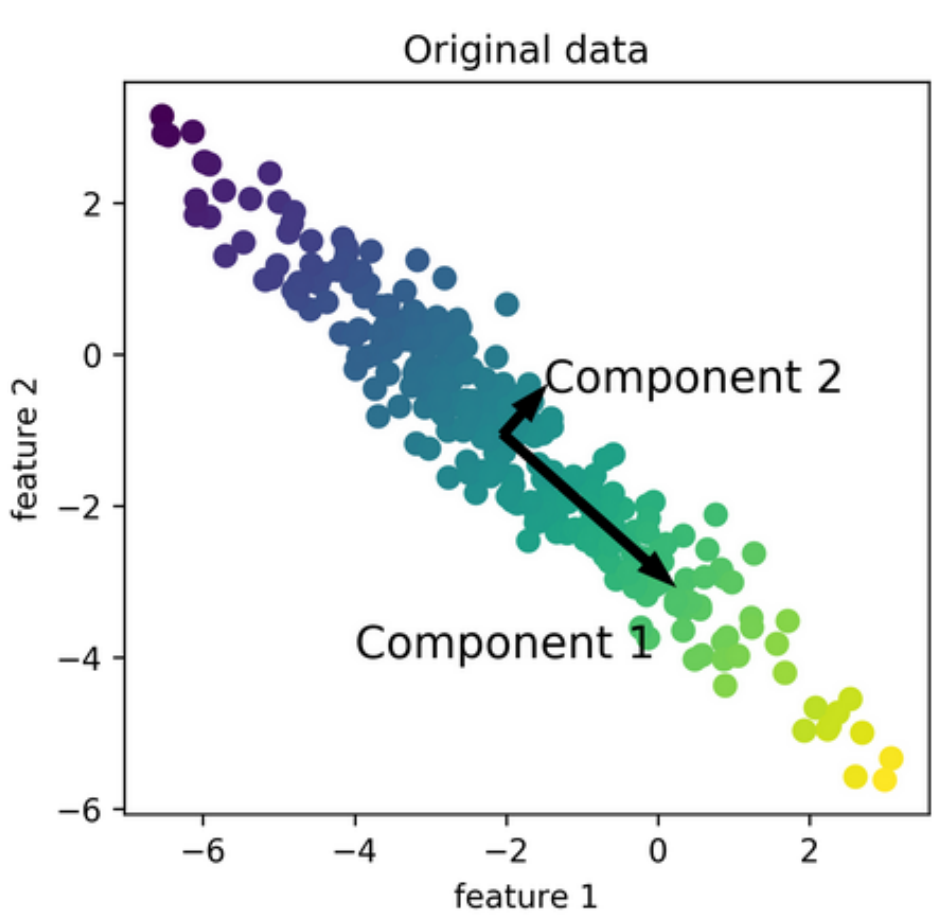
90K

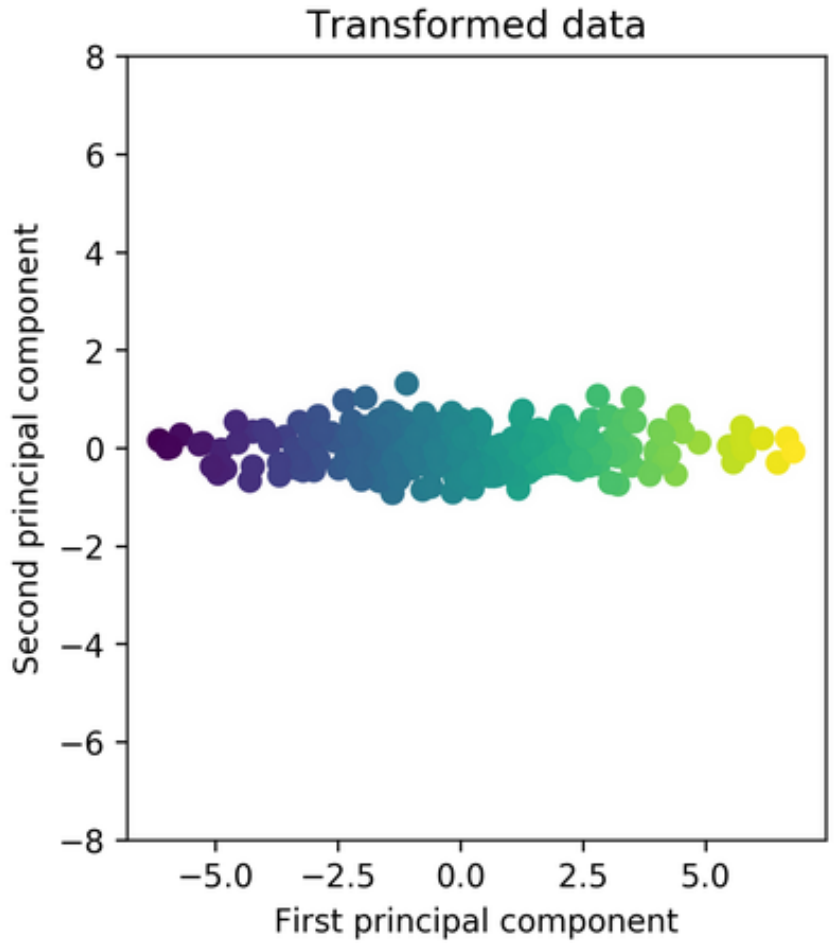
Shop the 3Blue1Brown store



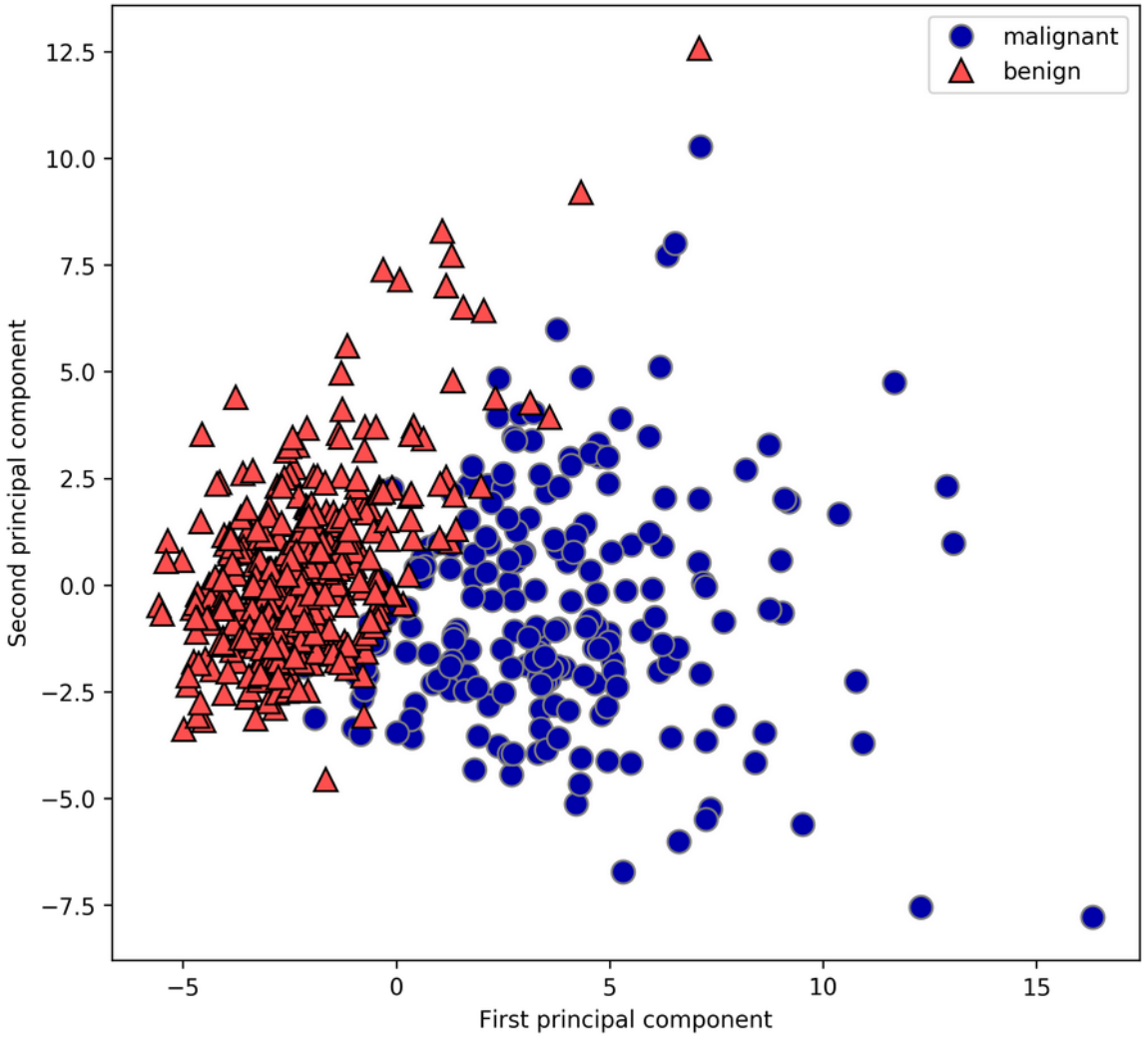


$20.00 store.dftba....
$17.29 store.dftba....
$29.00 store.dftba....
$12.56 store.dftba....

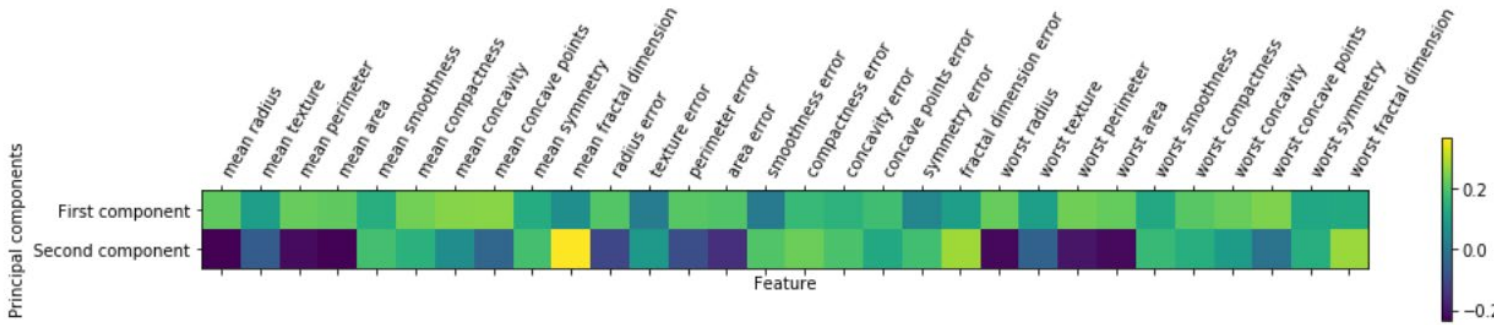

Original data
feature 2
feature 1
Component 2
Component 1
2
0
−2
−4
−6
−6
−4
−2
0
2


Transformed data
Second principal component
First principal component
8
6
4
2
0
−2
−4
−6
−8
−5.0
−2.5
0.0
2.5
5.0

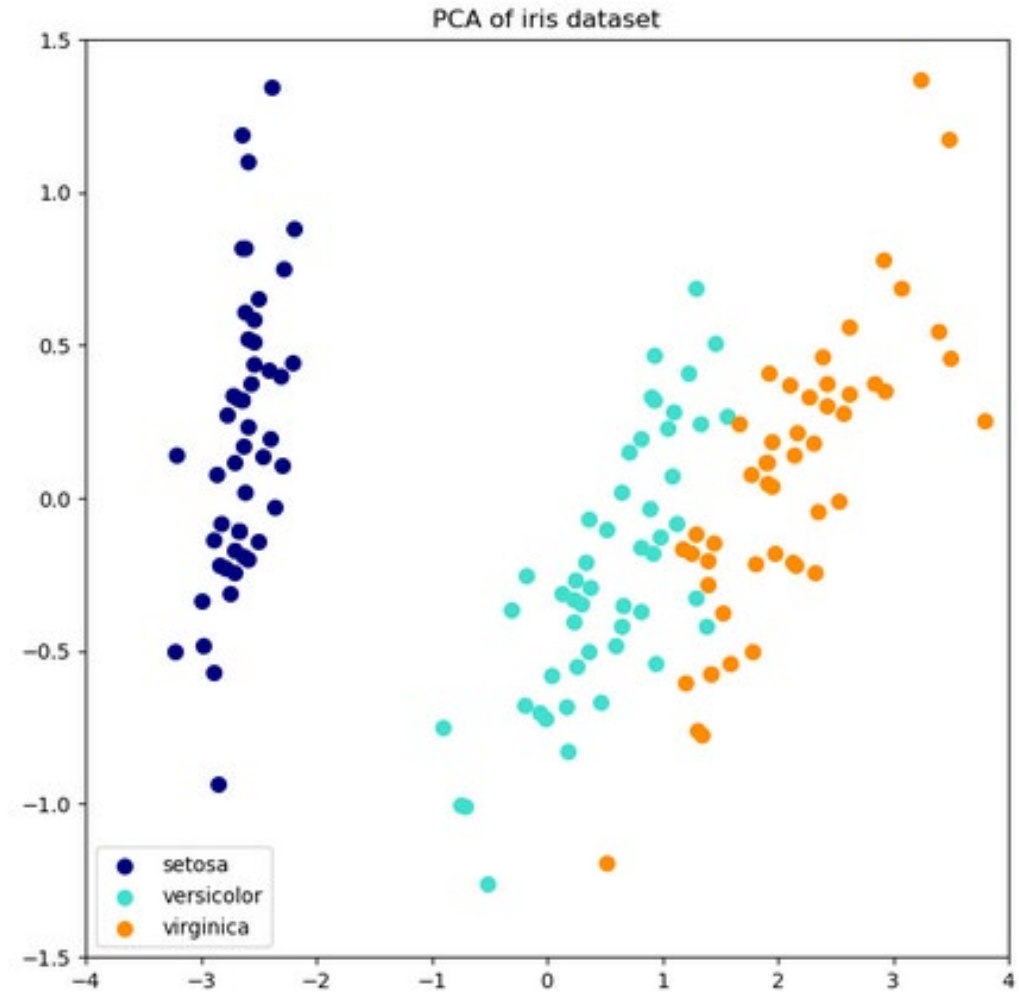
https://github.com/amueller/introduction_to_ml_with_python/blob/master/03-unsupervised-learning.ipynb



The cancer data set has 30 features. PCA lets us visualize a 2D representation of the data.



http://scikit-learn.org/stable/auto_examples/decomposition/plot_incremental_pca.html




```
pca.components_
```

```
array([[ 0.36138659, -0.08452251,  0.85667061,  0.3582892 ],
       [ 0.65658877,  0.73016143, -0.17337266, -0.07548102]])
```

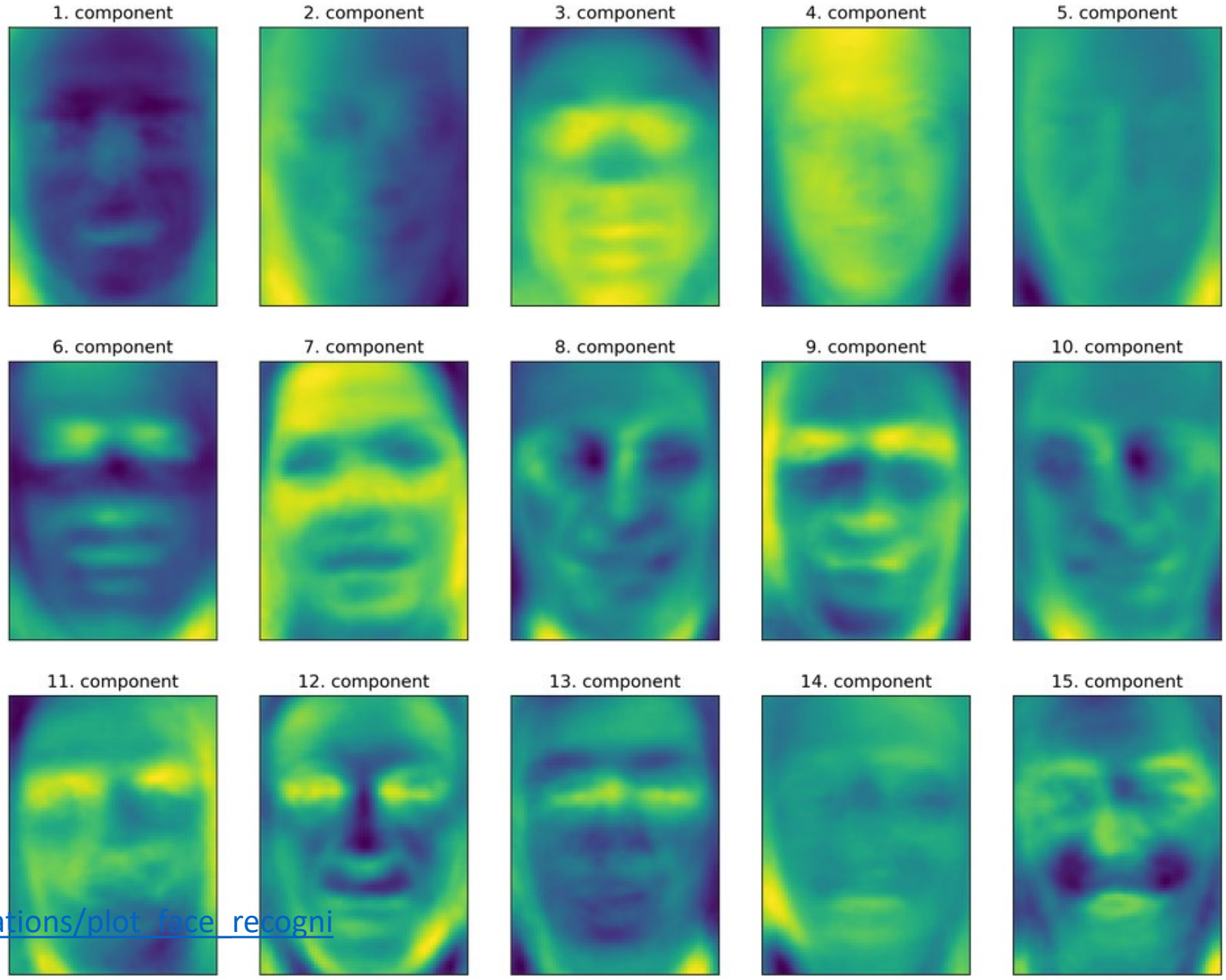
Eigenfaces!

If we plot the biggest PCA components for face images we get the eigenfaces!

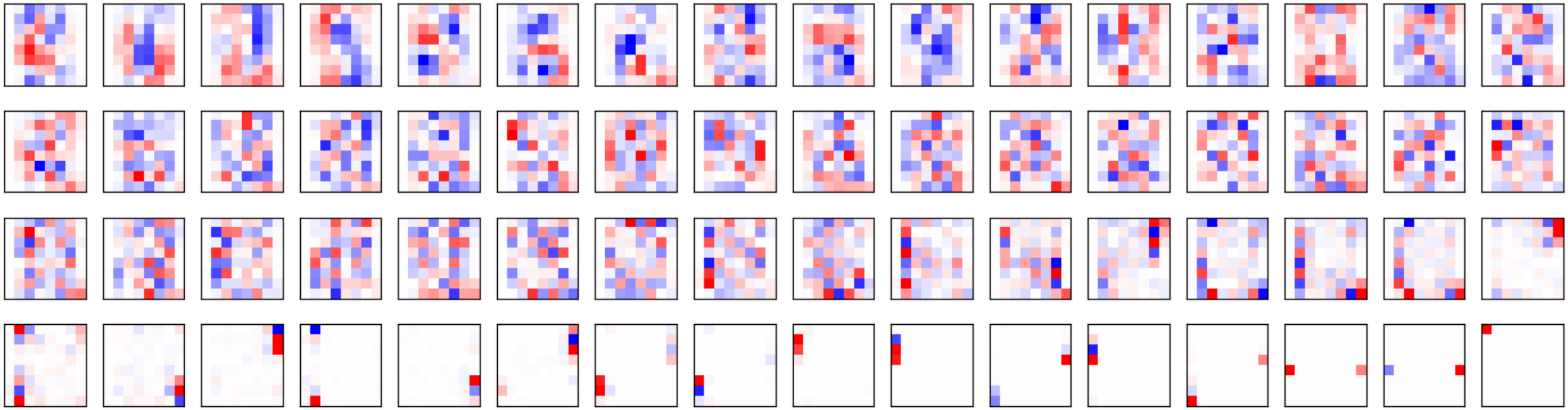
Every face could be represented as a linear combination of these.

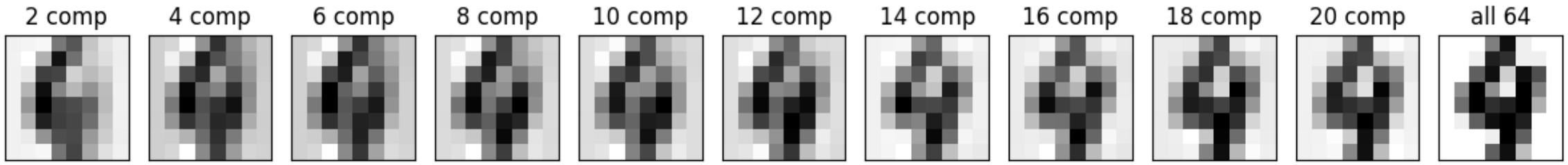
We can also use this trick for completing partial images. Fit the eigenfaces to the portion we've got then predict missing piece

https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html



Eigendigits





Manifold Learning

Manifold Learning

http://scikit-learn.org/stable/modules/manifold.html

Can we do even better? Transform the features in a way that makes the (unlabeled) data form clusters in simple 2D or 3D space.

These tricks are implemented in sklearn as “Manifold Learning”. Let’s look at one of the newer and most magical methods.

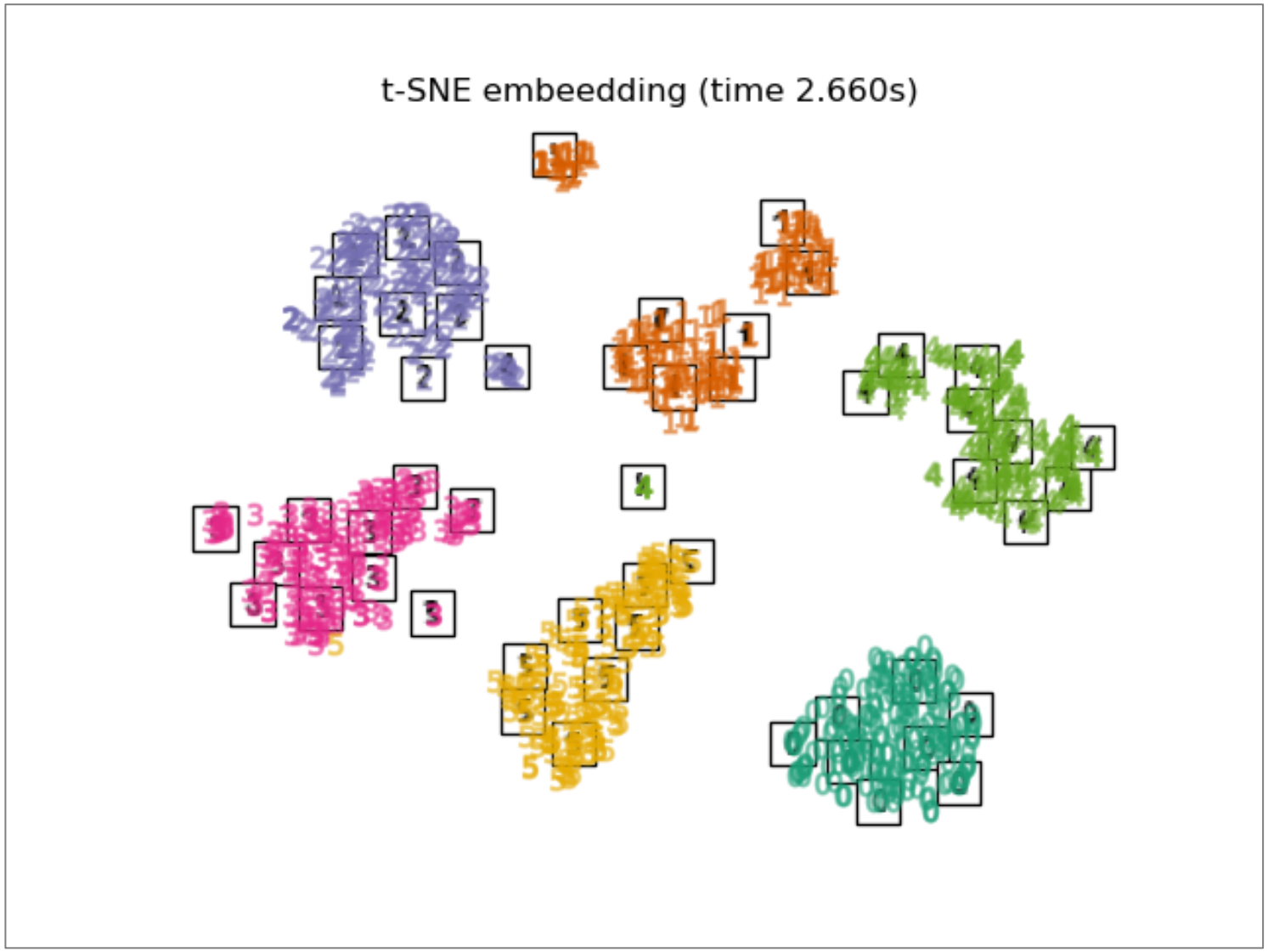
tSNE

t-distributed Stochastic Neighbor Embedding

Imagine we want to construct a 2D map of high-dimensional points. We can do this by attaching “springs” to each point in 2D space where the springs get smaller for points that are close neighbors

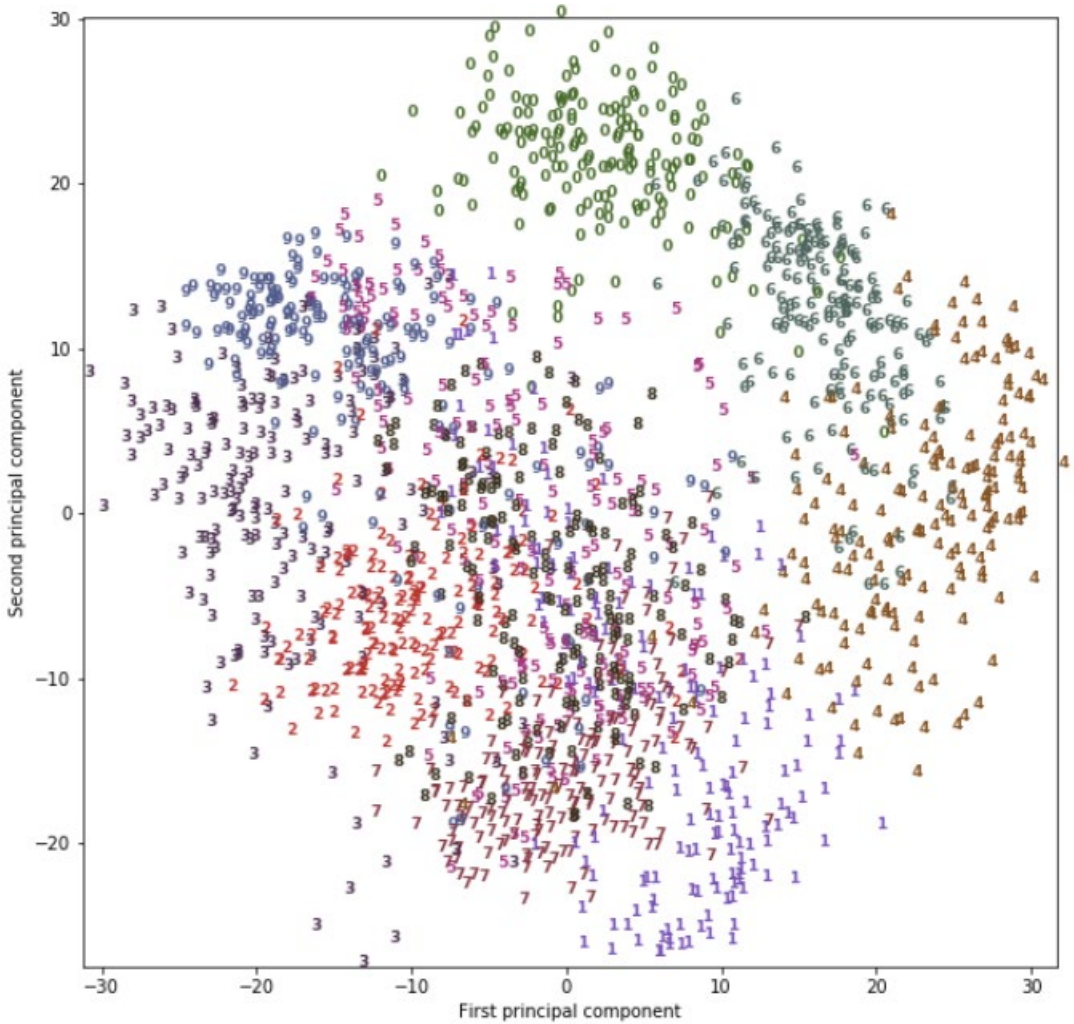
- https://distill.pub/2016/misread-tsne/
- https://www.youtube.com/watch?v=RJVL80Gg3lA&list=UUtXKDgv1AVoG88PLl8nGXmw (algorithm starts at 10:30)

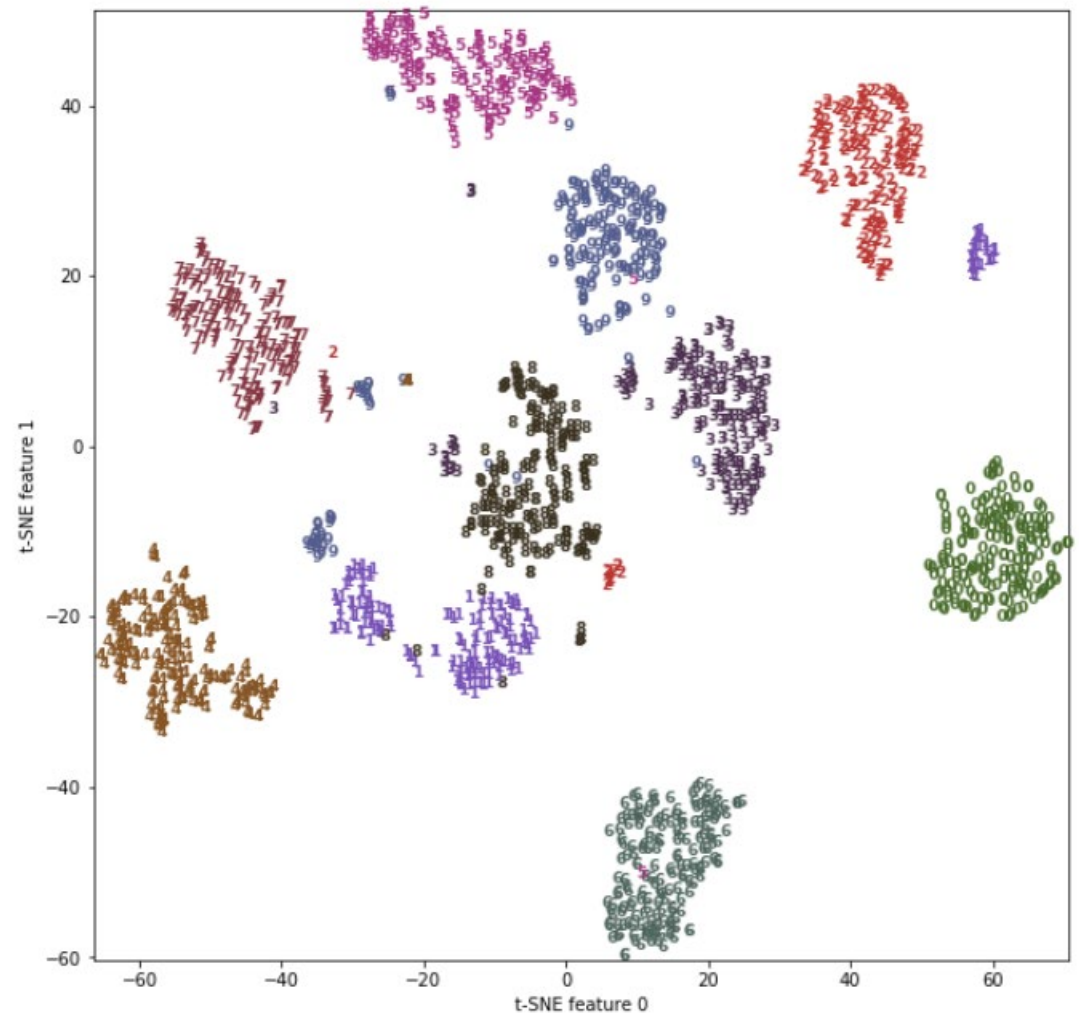
https://scikit-learn.org/stable/auto_examples/manifold/plot_lle_digits.html



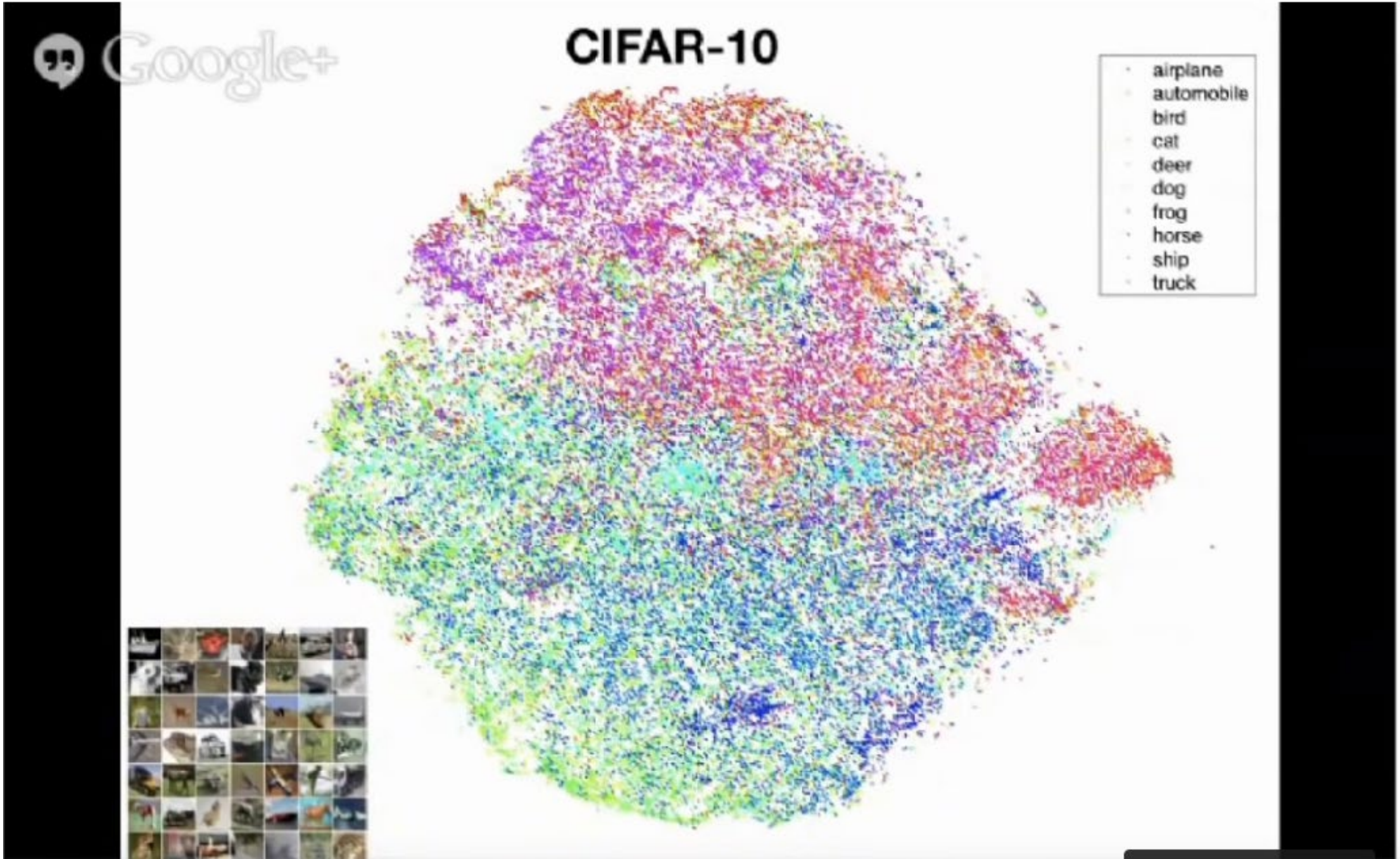
T-SNE on the digits dataset (only using digits 0-5) makes a 2D plot showing clusters of "similar" without knowing the right answers!

https://github.com/amueller/introduction_to_ml_with_python/blob/master/03-unsupervised-learning.ipynb





https://www.youtube.com/watch?v=RJVL80Gg3lA



Stochastic Search

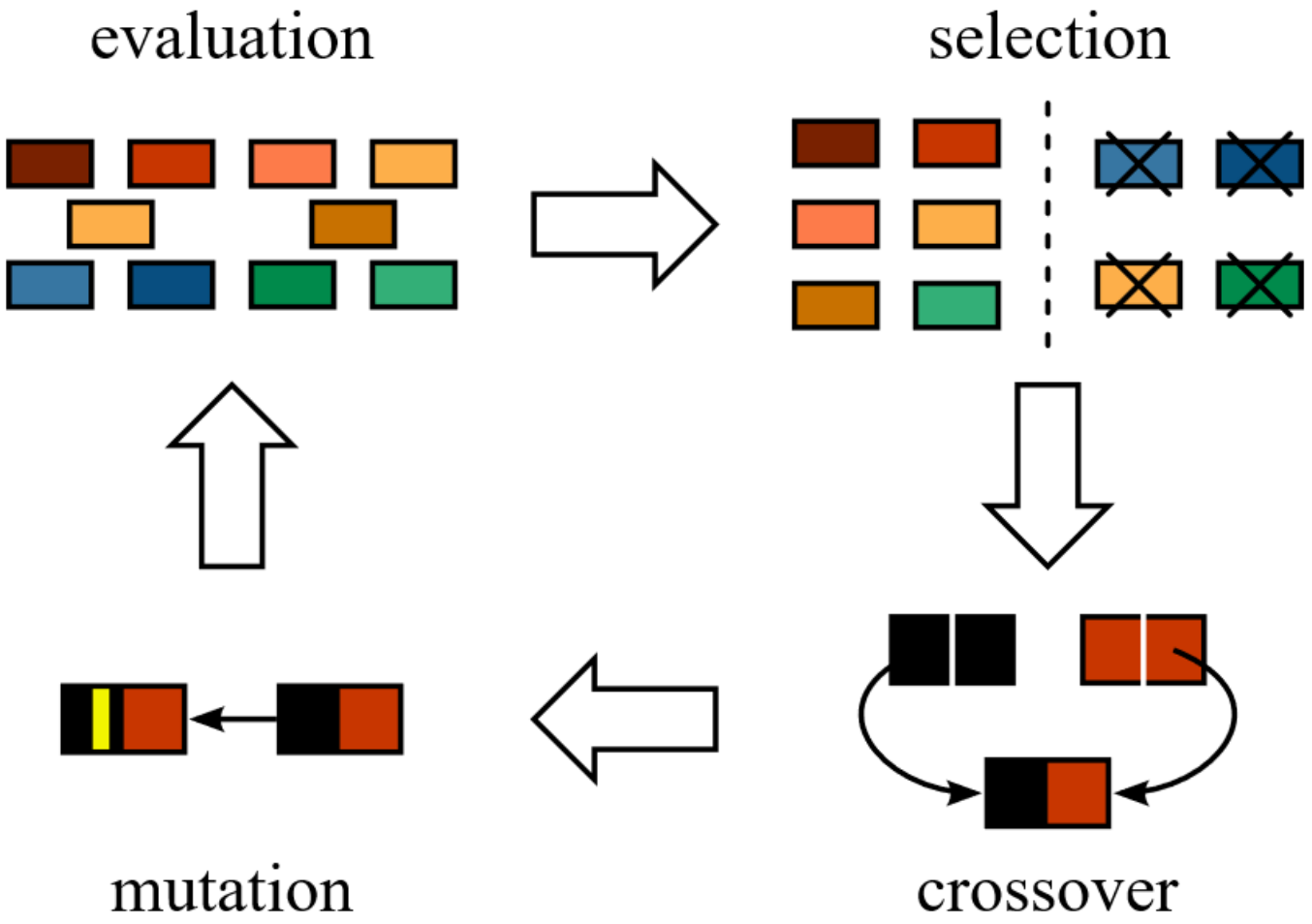
Problem #3

Can we do any training when we don't know the "right" answer?

If we have a given set of parameters (e.g. neural network weights, decision tree thresholds, etc.) we can try to find the optimal set that maximizes some metric

CGP Grey: How Machines Learn
https://www.youtube.com/watch?v=R9OHn5ZF4Uo

Genetic Breeding Algorithms



A neural network car with genetic weights (no backprop!)
https://www.youtube.com/watch?v=0Str0Rdkxxo

Genetic Breeding Algorithms

https://gplearn.readthedocs.io/en/stable/intro.html

Gplearn (a package that plays nicely with sklearn) represents arbitrary equations as trees:

$$y = X_0^2 - 3 \times X_1 + 0.5$$

This could be re-written as:

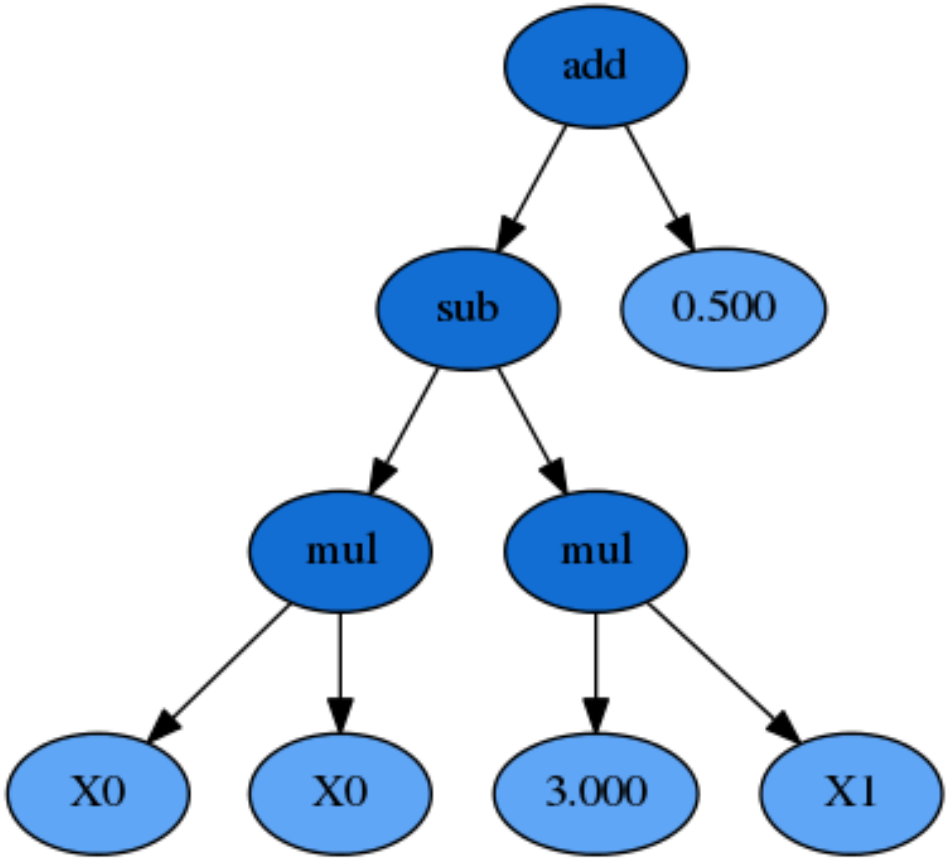
$$y = X_0 \times X_0 - 3 \times X_1 + 0.5$$

Or as a LISP symbolic expression (S-expression) representation which uses prefix-notation, and happens to be very common in GP, as:

$$y = (+(-(\times X_0 X_0)(\times 3 X_1))0.5)$$

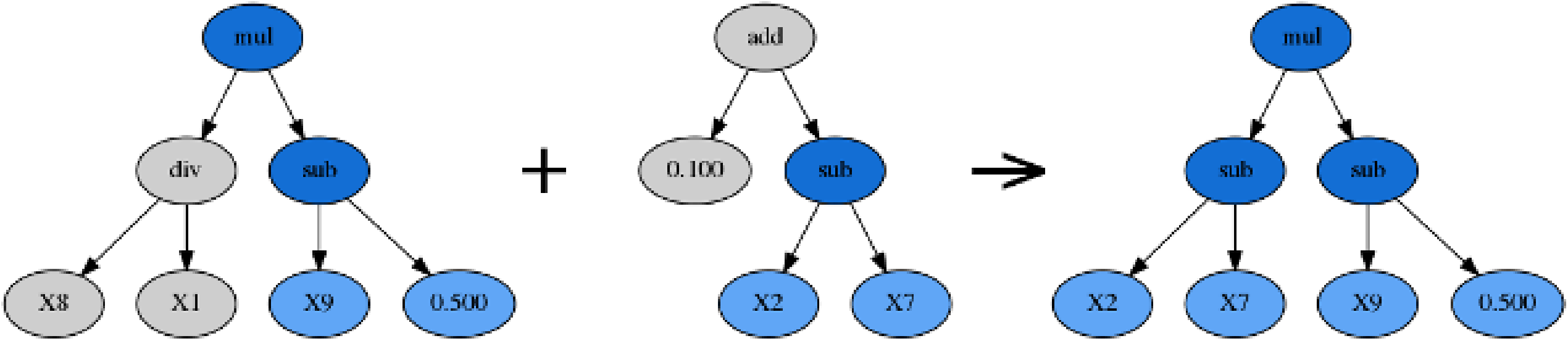
Or, since we're working in python here, let's express this as a numpy formula:

```
y = np.add(np.subtract(np.multiply(X0, X0), np.multiply(3., X1)), 0.5)
```



Genetic Breeding Algorithms

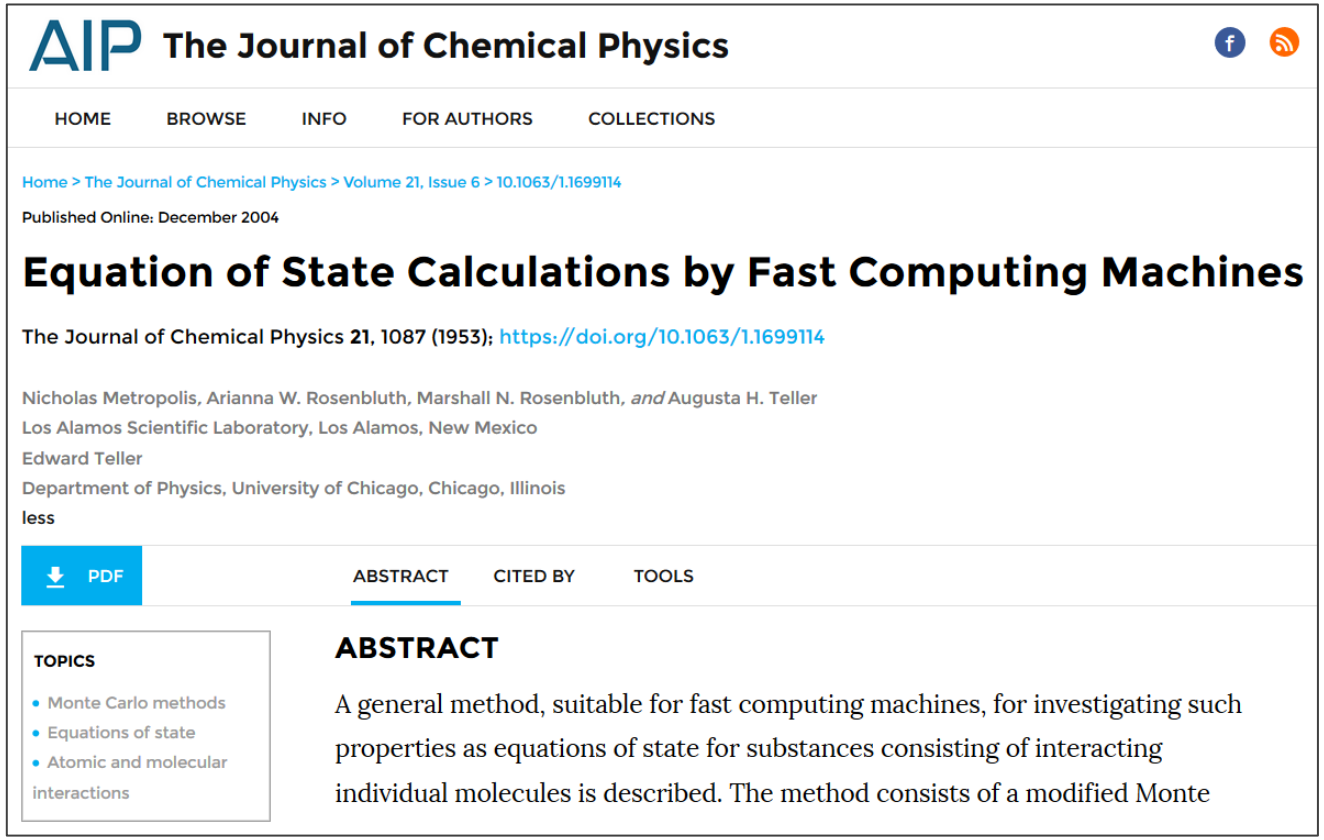
https://gplearn.readthedocs.io/en/stable/intro.html



Then it randomly evolves the trees to discover an equation that represents the data

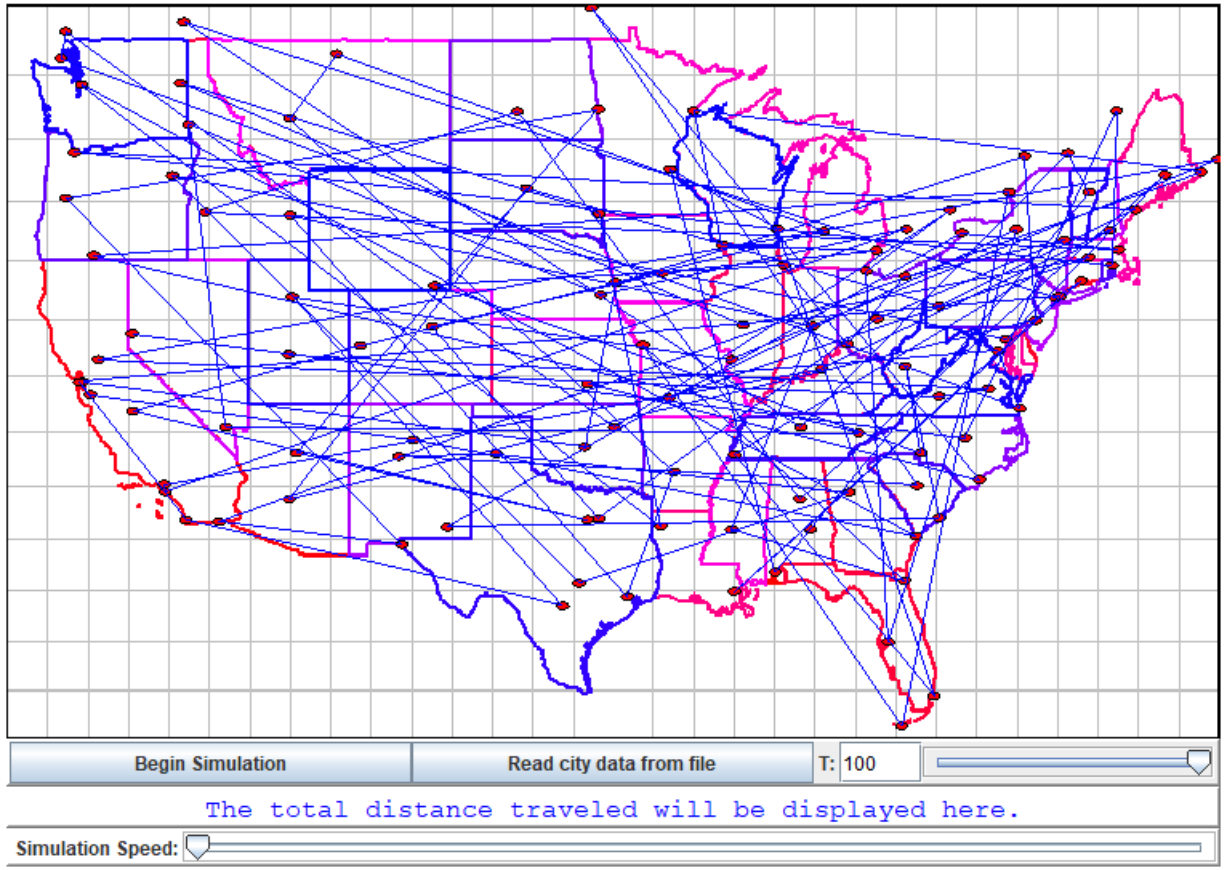
Simulated Annealing

Another stochastic search uses the **METROPOLIS ALGORITHM** to look for an optimal state



Simulated Annealing

https://www.compadre.org/osp/items/detail.cfm?ID=11538



#	City Name	Long. (deg)	Lat. (deg)
1	"Columbia, ...	-81.03333333	34
2	"Albuquerqu...	-106.65	35.08333333
3	"Amarillo, ...	-101.8333333	35.18333333
4	"Atlanta, G...	-84.38333333	33.75
5	"Austin, Te...	-97.73333333	30.26666667
6	"Baker, Ore. "	-117.8333333	44.78333333
7	"Baltimore,...	-76.63333333	39.3
8	"Bangor, Ma...	-68.78333333	44.8
9	"Birmingham...	-86.83333333	33.5
10	"Bismarck, ...	-100.7833333	46.8
11	"Boise, Ida...	-116.2166667	43.6
12	"Boston, Ma...	-71.08333333	42.35
13	"Buffalo, N...	-78.83333333	42.91666667
14	"Carlsbad, ...	-104.25	32.43333333
15	"Charleston...	-79.93333333	32.78333333
16	"Charleston...	-81.63333333	38.35
17	"Charlotte,...	-80.83333333	35.23333333
18	"Cheyenne, ...	-104.8666667	41.15
19	"Chicago, I...	-87.61666667	41.83333333
20	"Cincinnati...	-84.5	39.13333333
21	"Cleveland,...	-81.61666667	41.46666667
22	"Albany, N....	-73.75	42.66666667
23	"Columbus, ...	-83.01666667	40
24	"Dallas, Te...	-96.76666667	32.76666667
25	"Denver, Co...	-105	39.75
26	"Des Moines...	-93.61666667	41.58333333
27	"Detroit, M...	-83.05	42.33333333
28	"Dubuque, I...	-90.66666667	42.51666667
29	"Duluth, Mi...	-92.08333333	46.81666667
30	"Eastport, ...	-67	44.9
31	"El Centro,...	-115.55	32.63333333
32	"El Paso, T...	-106.4833333	31.76666667
33	"Eugene, Or...	-123.0833333	44.05
34	"Fargo, N.D. "	-96.8	46.86666667
35	"Flagstaff,...	-111.6833333	35.21666667
36	"Fort Worth...	-97.31666667	32.71666667
37	"Fresno, Ca...	-119.8	36.73333333
38	"Grand Junc...	-108.55	39.08333333
39	"Grand Rapi...	-85.66666667	42.96666667
40	"Havre, Mon...	-109.7166667	48.55
41	"Helena, Mo...	-112.0333333	46.58333333
42	"Hot Spring...	-93.05	34.51666667
43	"Houston, T...	-95.35	29.75