Feature Extraction

Ludwig Krippahl

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Summary

- Feature Extraction
- Principal Component Analysis
- Self Organizing Maps
- Extraction and reduction with SOM (toy example)

Feature Extraction

Feature Extraction

- General idea: derive useful features from data
  - Image patches
  - Sound frequencies
  - Types of words
- Transform data into a more useful data set
  - Many specific solutions for specific problems
- But one very used general solution: PCA

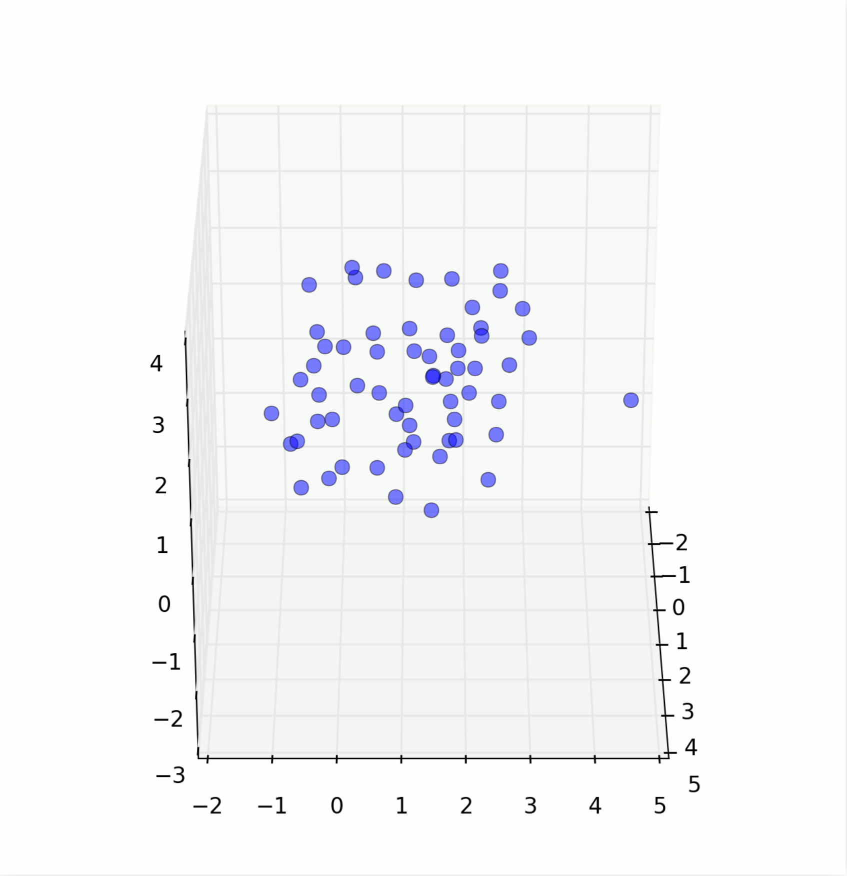
PCA

Principal Component Analysis

- Main idea: find orthogonal base so that
  - First vector aligns with the greatest dispersion of points
  - Subsequent (orthogonal) align with the remaining dispersion
- We can reduce dimensionality keeping only the k first

PCA

- Illustrate with simple data set:



Based on Sebastian Raschka's demo: Implementing a Principal Component Analysis (PCA) in Python step by step

PCA

- First we need the scatter matrix:

$$m = \frac{1}{n} \sum_{k=1}^{n} x_k$$

$$S = \sum_{k=1}^{n} (x_k - m)(x_k - m)^T$$

- Scatter matrix is a multiple of the covariance matrix, with the value at i, j the covariance of x_i to x_j

- Then we find the eigenvectors of the scatter matrix

- An eigenvector of A is a v such that

$$Av = \lambda v$$

- It's a vector in the direction the matrix transformation "stretches" the coordinates

PCA

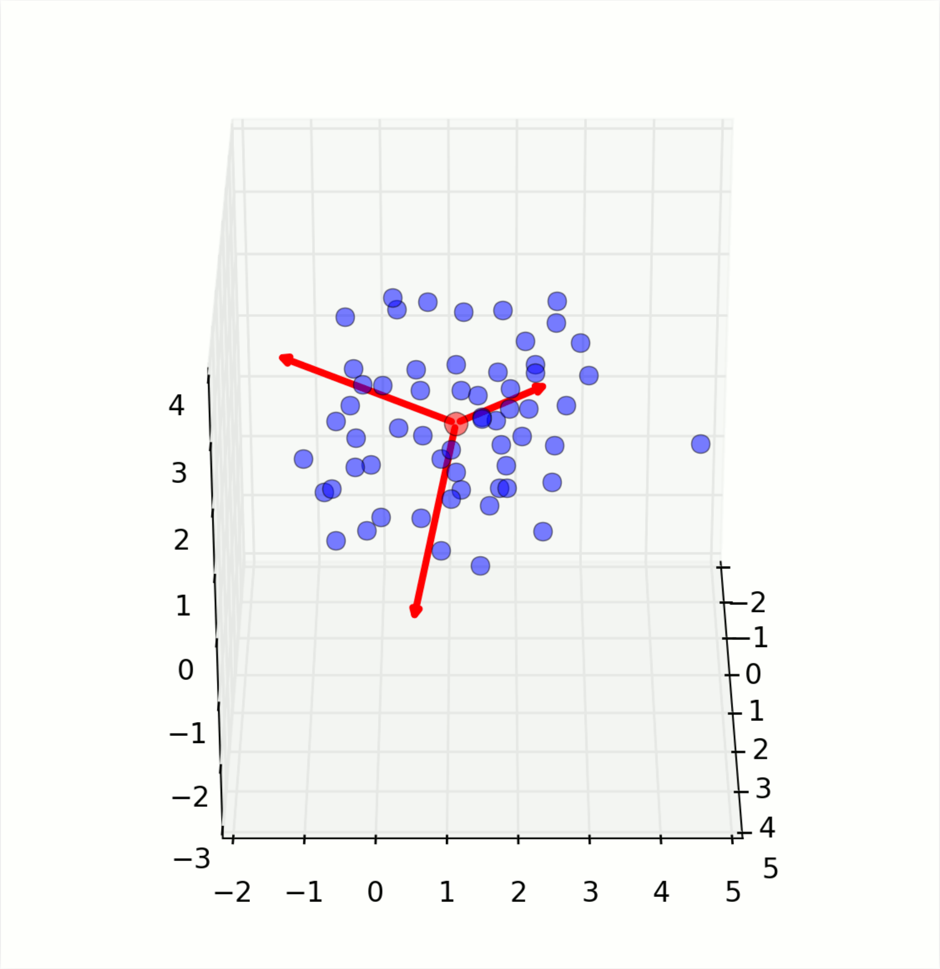
```
import numpy as np
mean_v = np.mean(data,axis=0)
scatter = np.zeros((3,3))
for i in range(data.shape[0]):
    centered = (data[i,:] - mean_v).reshape(3,1)
    scatter += centered.dot(centered.T)

print(mean_v)
[ 1.07726488  1.11609716  1.03600411]
print(scatter)
 [[ 110.10604771   39.91266264   52.3183266 ]
  [  39.91266264   80.68947748   34.48293948]
  [  52.3183266    34.48293948   97.58136923]]

eig_vals, eig_vecs = np.linalg.eig(scatter)
print(eig_vals)
[ 183.57291365   51.00423734   53.79974343]
print(eig_vecs)
[[ 0.66718409  0.72273622  0.18032676]
 [ 0.45619248 -0.20507368 -0.8659291 ]
 [ 0.58885805 -0.65999783  0.46652873]]
```

PCA

- Eigenvectors



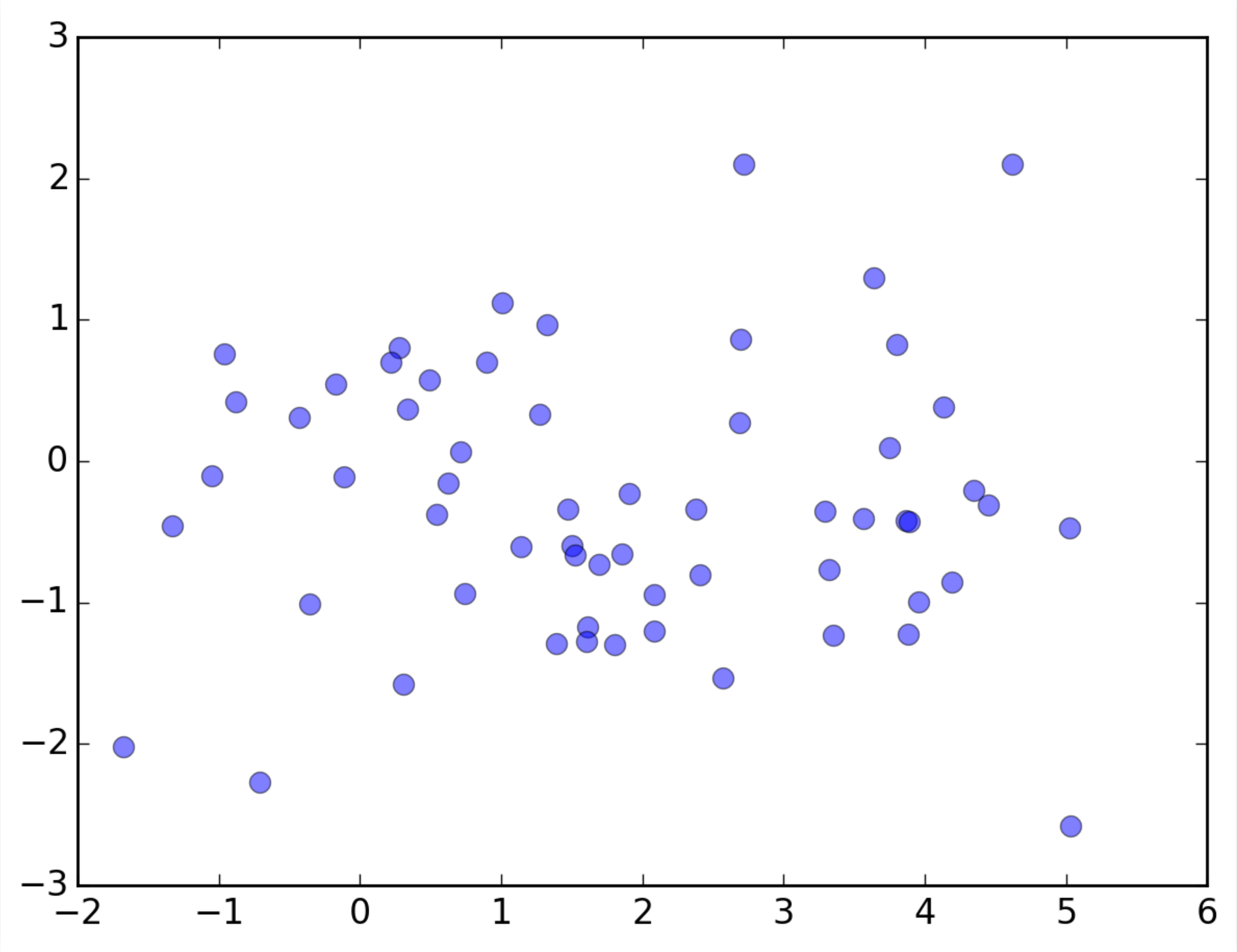
PCA

- Select the two eigenvectors with the largest (absolute) eigenvalues and project the data

```
fig = plt.figure(figsize=(7,7))
transf = np.vstack((eig_vecs[:,0],eig_vecs[:,2]))
t_data = transf.dot(data.T)
plt.plot(t_data[0,:], t_data[1,:],'o', markersize=7, color='blue', alpha=0.5)
plt.gca().set_aspect('equal', adjustable='box')
plt.savefig('L16-transf.png',dpi=200,bbox_inches='tight')
plt.close()
```

PCA

- Projected data



PCA with Scikit-Learn

- `sklearn.decomposition.PCA`
- Note: points in rows, features in columns
- Select the two eigenvectors with the largest (absolute) eigenvalues and project the data

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(data)
print(pca.components_)
t_data = pca.transform(data)
```

```
[[-0.66718409 -0.45619248 -0.58885805]
 [ 0.18032676 -0.8659291   0.46652873]]
```

Self Organizing Maps

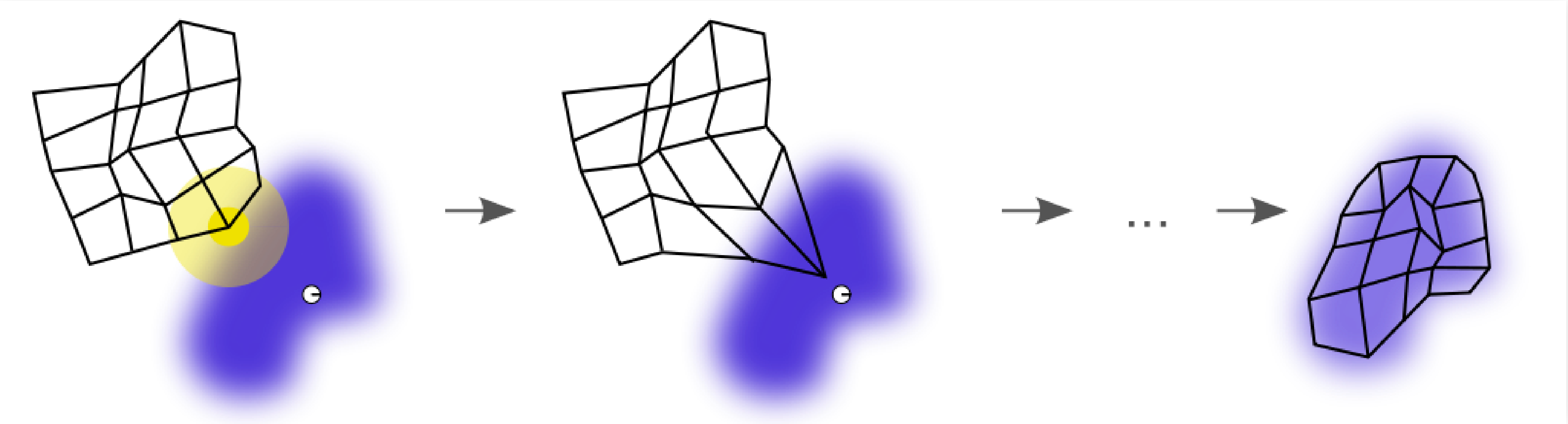
Self Organizing Map

- Artificial Neural Network
- Input dimension equal to the input space
- Neurons disposed in a 2D matrix
- Two distance measures: between neurons and neuron to vector
- Training:

- Coefficients start with small random numbers (or 2 principal components, random examples, ...)
- Find Best Matching Unit (BMU) to each example
- Adjust BMU and neurons closest in the matrix (rate decreasing with matrix distance)
- Learning coefficient decreases monotonically

Self Organizing Map

- 2D matrix mapped to ND space



Source: en.wikipedia.org/wiki/Self-organizing_map

SOM Example: colors

- Using the minisom library:
  - https://github.com/JustGlowing/minisom
- Based on http://www.pymvpa.org/examples/som.html

```
colors = np.array(
         [[0., 0., 0.],
          [0., 0., 1.],
  ...
          [.5, .5, .5],
          [.66, .66, .66]])
color_names = \
        ['black', 'blue', 'darkblue', 'skyblue',
         'greyblue', 'lilac', 'green', 'red',
         'cyan', 'violet', 'yellow', 'white',
         'darkgrey', 'mediumgrey', 'lightgrey']
```

SOM Example: colors

- Training the SOM

```
class MiniSom:
  def __init__(self, x, y, input_len, sigma=1.0,
               learning_rate=0.5, ...):
    """
    Initializes a Self Organizing Maps.
    x,y - dimensions of the SOM
    input_len - number of the elements of the vectors in input
    sigma - spread of the neighborhood function (Gaussian)
    learning_rate - initial learning rate
    """

  def random_weights_init(self, data):
    """ Initializes the weights of the
        SOM picking random samples from data
    """

  def train_batch(self, data, num_iteration):
    """ Trains using all the vectors in data sequentially
    """
```

SOM Example: colors

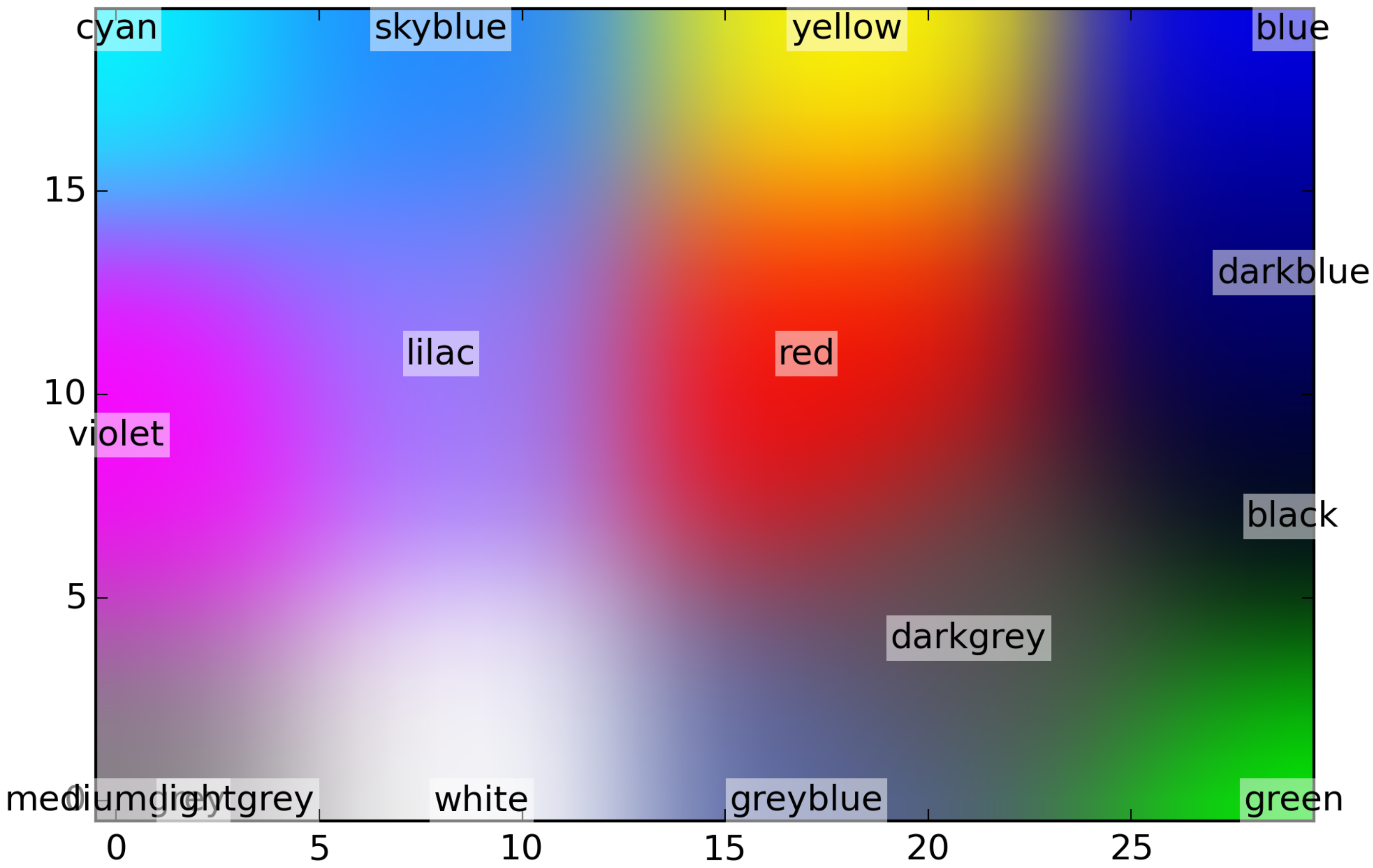
- Training the SOM

```
from minisom import MiniSom
import matplotlib.pyplot as plt
import numpy as np

plt.figure(1, figsize=(7.5, 5), frameon=False)
som = MiniSom(20, 30, 3, learning_rate=0.5, sigma = 2)
som.random_weights_init(colors)
som.train_batch(colors,10000)

for ix in range(len(colors)):
    winner = som.winner(colors[ix])
    plt.text(winner[1], winner[0], color_names[ix], ha='center',
             va='center',bbox=dict(facecolor='white', alpha=0.5, lw=0))
plt.imshow(som.weights, origin='lower')
plt.savefig('L6-colors.png',dpi=300)
plt.close()
```

SOM


cyan
skyblue
yellow
blue
darkblue
lilac
red
violet
black
darkgrey
white
greyblue
green
15
10
5
0
5
10
15
20
25

SOM Example

SOM Example

The problem

- Examine country progress from indicators: Gapminder data,
  - http://www.gapminder.org
- There is one file per indicator (.xlsx), with one row per country and one column per year
- We need to organize yearly data by country:
  - GDP per capita, Life expectancy, Infant Mortality, Employment over 15

SOM Example

The problem

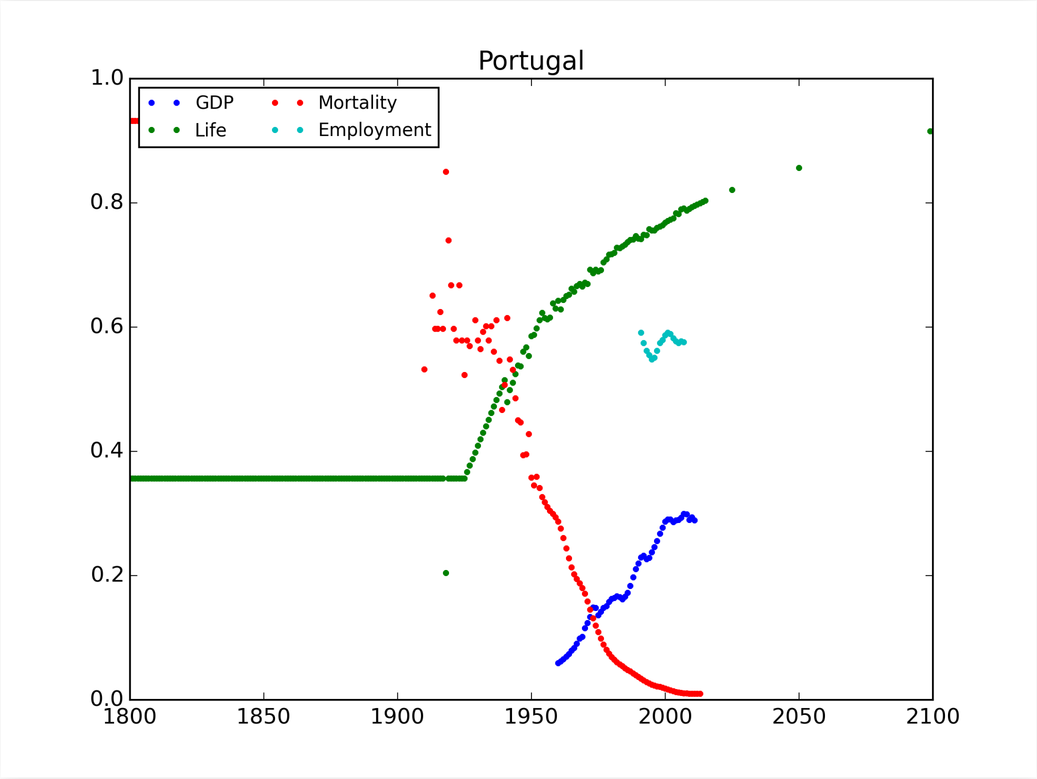
- Available data

BK25 | 28,799

	A	AR	AS	AT	AU	AV	AW	AX	AY
1	Life expectancy with projections	1807	1808	1809	1810	1811	1812	1813	1814
2	Afghanistan	28,139273	28,129027	28,11878	28,108533	28,098287	28,08804	28,077793	28,067547
3	Albania	35,4	35,4	35,4	35,4	35,4	35,4	35,4	35,4
4	Algeria	28,8224	28,8224	28,8224	28,8224	28,8224	28,8224	28,8224	28,8224
5	American Samoa								
6	Andorra								
7	Angola	26,98	26,98	26,98	26,98	26,98	26,98	26,98	26,98
8	Anguilla								
9	Antigua and Barbuda	33,536	33,536	33,536	33,536	33,536	33,536	33,536	33,536
10	Argentina	33,2	33,2	33,2	33,2	33,2	33,2	33,2	33,2
11	Armenia	33,995	33,995	33,995	33,995	33,995	33,995	33,995	33,995
12	Aruba	34,419	34,419	34,419	34,419	34,419	34,419	34,419	34,419
13	Australia	34,05	34,05	34,05	34,05	34,05	34,05	34,05	34,05
14	Austria	34,4	34,4	34,4	34,4	34,4	34,4	34,4	34,4
15	Azerbaijan	29,165	29,165	29,165	29,165	29,165	29,165	29,165	29,165

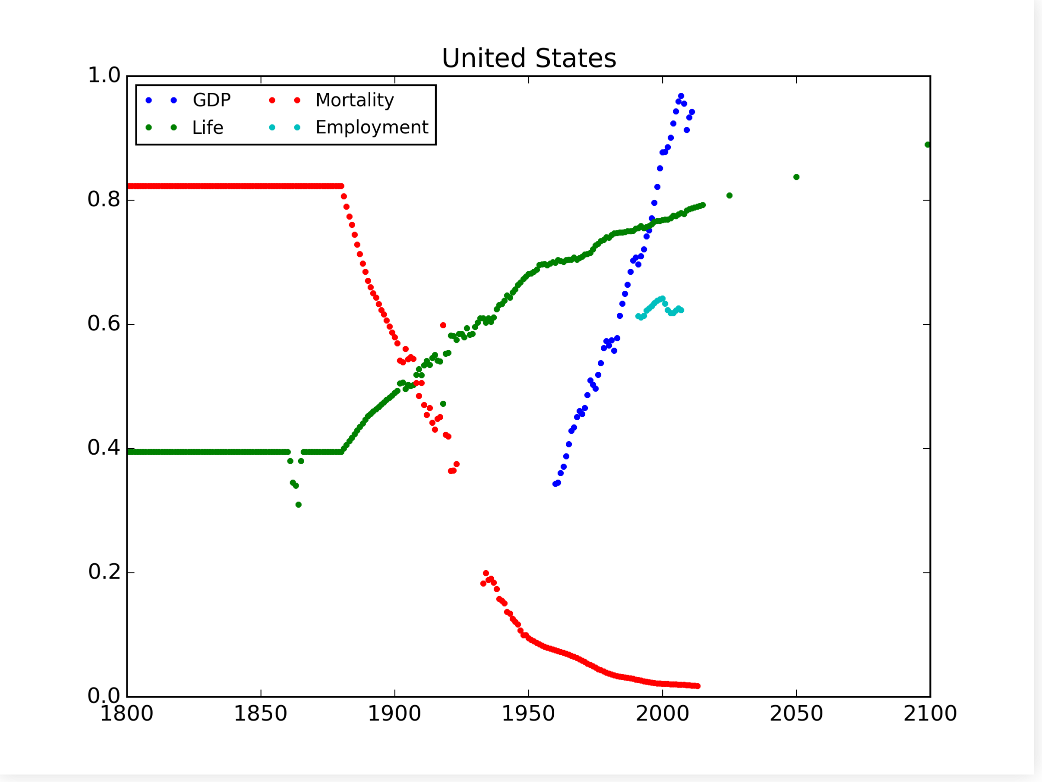
SOM Example

- Data is very diverse
  - Different number of points for different series
  - Different number of points for different countries



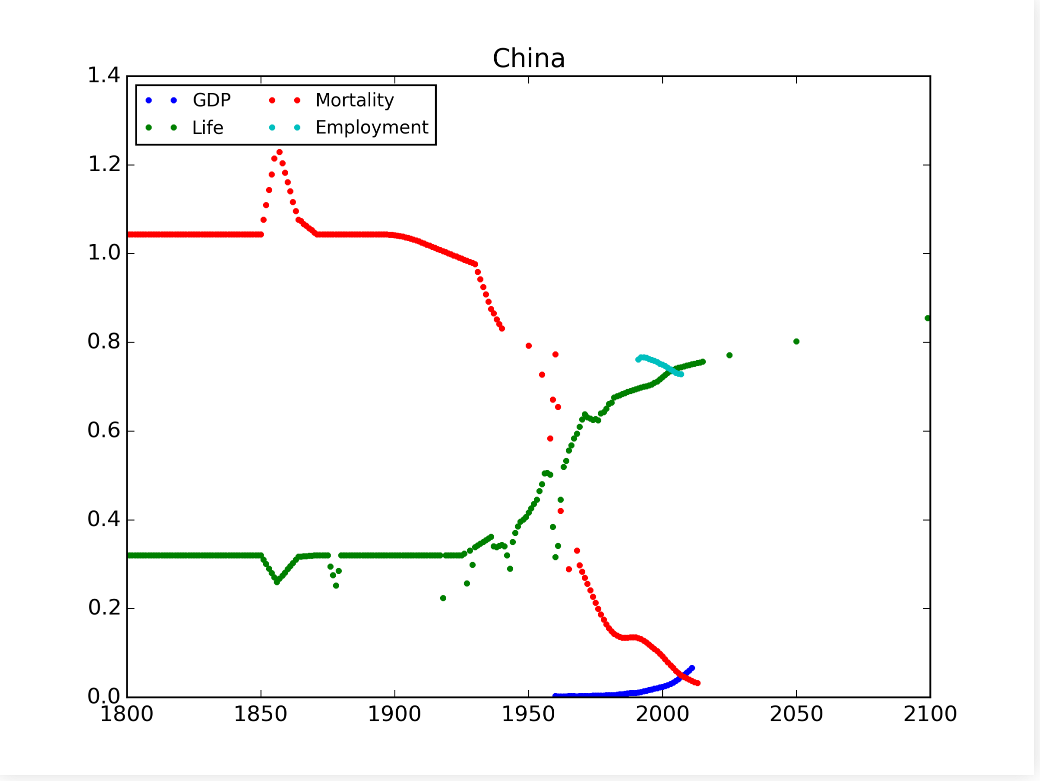
SOM Example

- Data is very diverse
  - Different number of points for different series
  - Different number of points for different countries



SOM Example

- Data is very diverse
  - Different number of points for different series
  - Different number of points for different countries



SOM Example

The problem

- How to organize countries by progress?

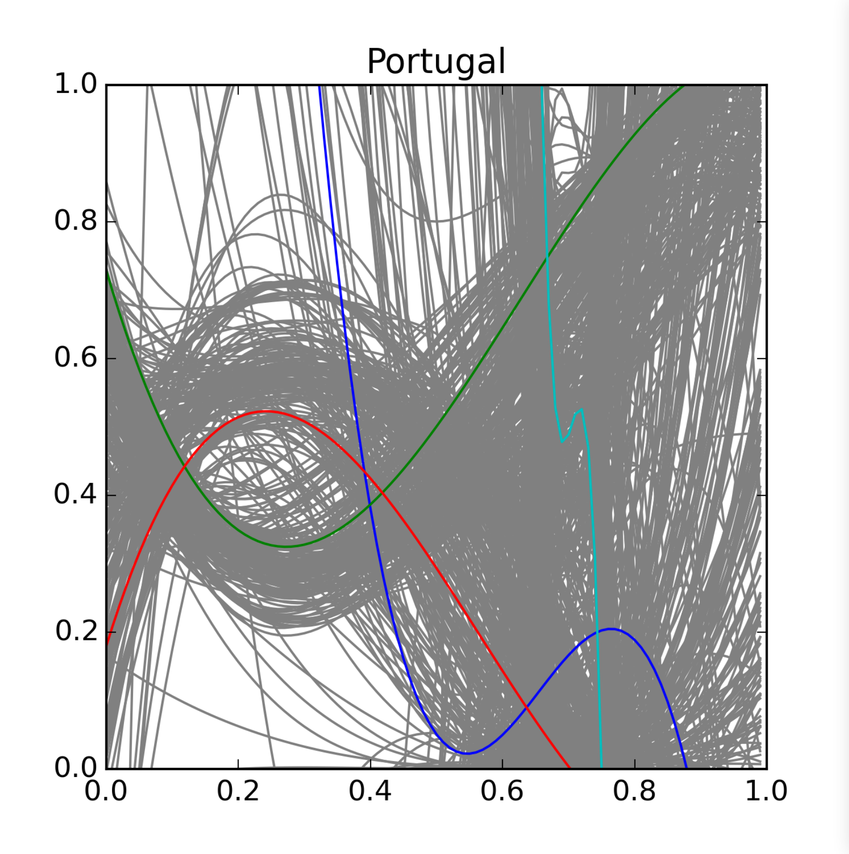
The solution

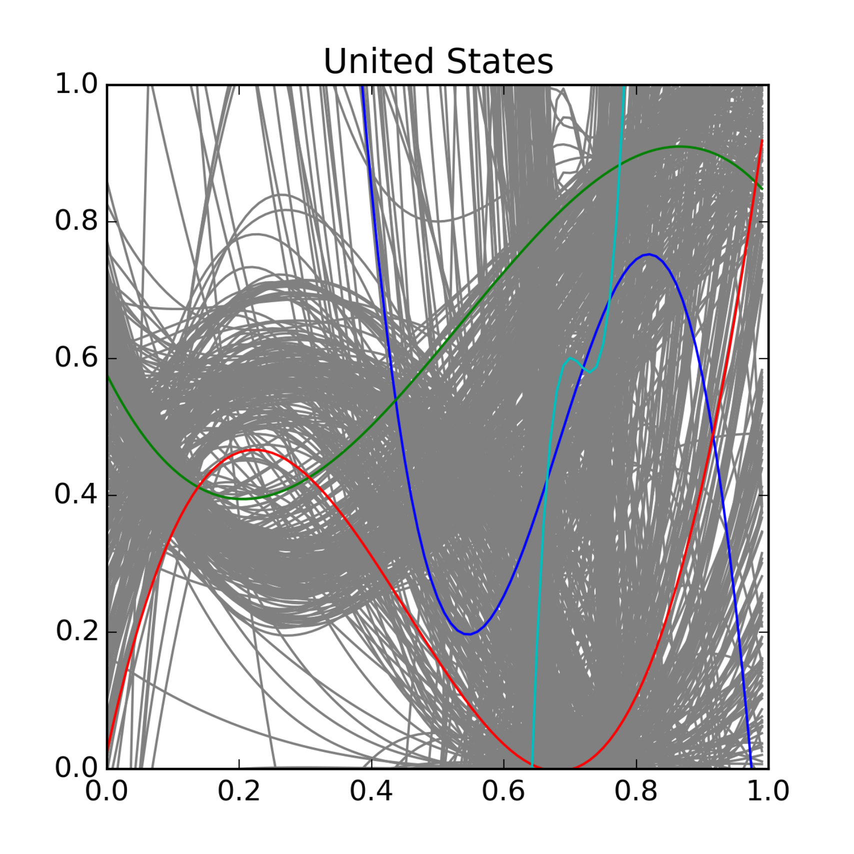
- First, feature extraction:

- Compute polynomial regression for each series
- Describe each country by a set of coefficients
- Degree 3, 4 indicators, 16 dimensions

SOM Example

- Compute polynomial regression for each series





SOM Example

The problem

- How to organize countries by progress?

The solution

- First, feature extraction:
  - Compute polynomial regression for each series
  - Describe each country by a set of coefficients
  - Degree 3, 4 indicators, 16 dimensions
- Second, project into lower dimensions
  - Train SOM
  - Label countries in SOM

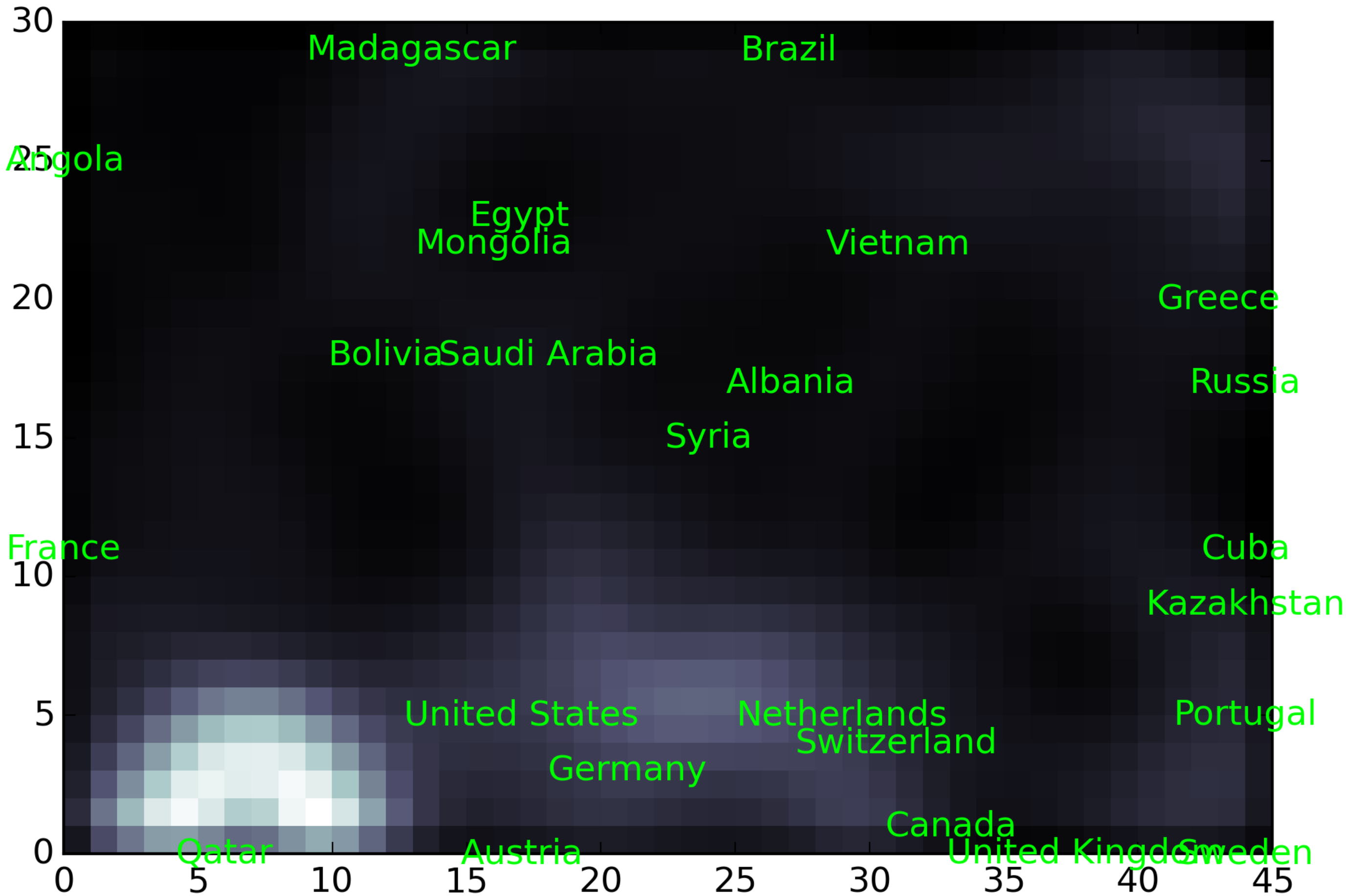
SOM Example

- High dimension descriptors
  - Each country has 4 x 4 = 16 values (Four coefficients, four data sets)
- Project everything in 2 dimensions
  - Self organizing maps

```
som = MiniSom(30, 45, features, learning_rate=0.5, sigma = 2)
som.random_weights_init(descs)
som.train_batch(descs,10000)
to_plot = open('countries_to_plot.txt').readlines()
for ix in range(len(to_plot)):
    to_plot[ix]=to_plot[ix].strip()

plt.figure(1, figsize=(7.5, 5), frameon=False)
plt.bone()
plt.pcolor(som.distance_map()) # average dist. to neighs.
for ix in range(len(descs)):
    if countries[ix].name in to_plot:
        winner = som.winner(descs[ix])
        plt.text(winner[1], winner[0], countries[ix].name,
                 ha='center', va='center',color='lime')
plt.savefig('L6-countries_som.png',dpi=300)
plt.close()
```

SOM Example



Madagascar
Brazil
Angola
Egypt
Mongolia
Vietnam
Greece
Bolivia
Saudi Arabia
Albania
Russia
Syria
France
Cuba
Kazakhstan
United States
Netherlands
Portugal
Switzerland
Germany
Canada
Qatar
Austria
United Kingdom
Sweden
0
5
10
15
20
25
30
35
40
45

Summary

Feature Extraction

Summary

- Feature extraction
  - Not all data sets have "nice" feature vectors
- PCA and Self organizing maps
  - Useful methods for projecting data into fewer dimensions
- Countries example:
  - "messy" data, feature extraction with regression
  - High dimensions, dimensional reduction with SOM for visualization

Further reading

- PCA with Scikit-Learn,
  - http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
- Wikipedia: Self Organizing Map