Session 16

Unsupervised Learning - Clustering and Dimensionality Reduction

Snorre Ralund

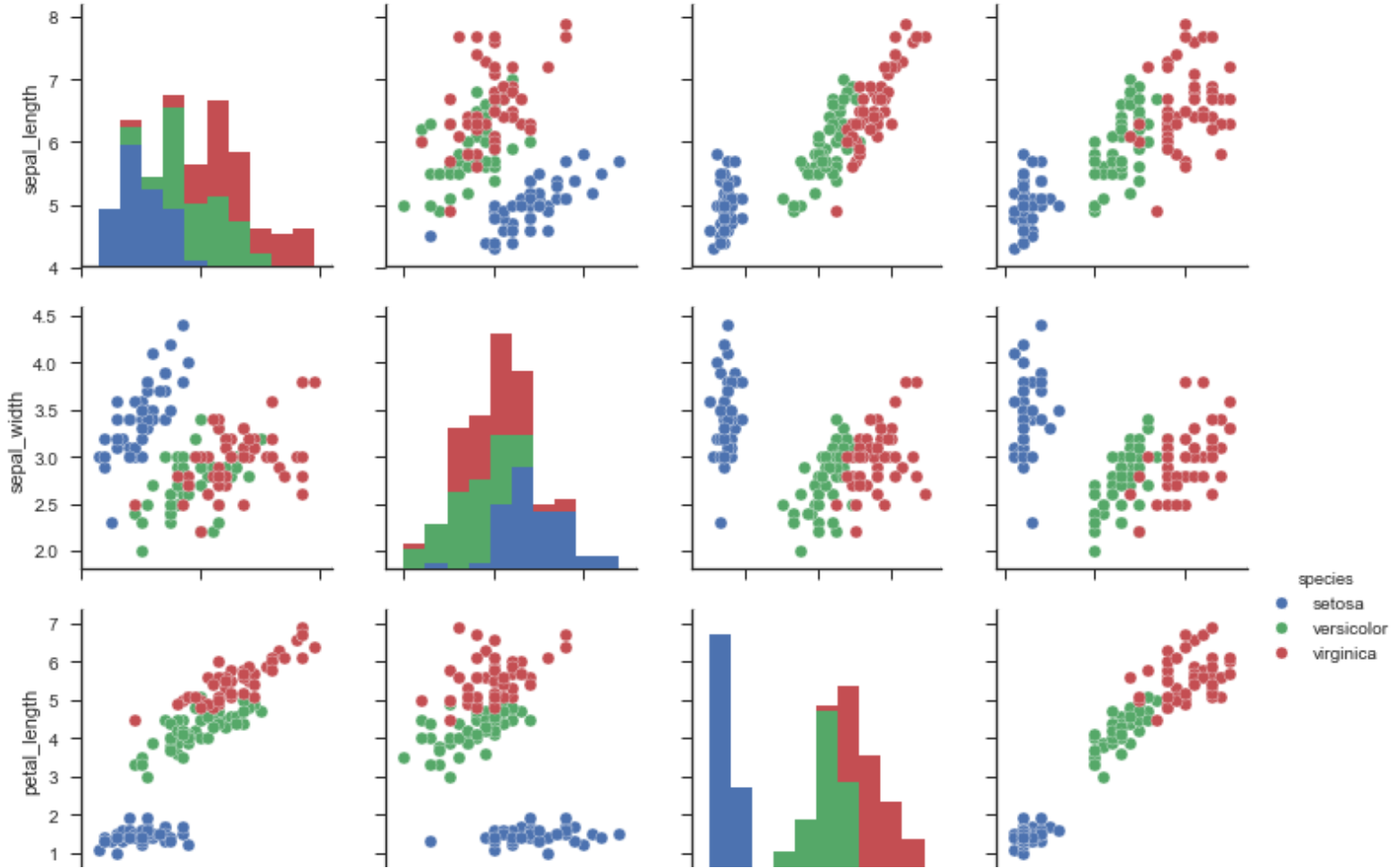
Motivation

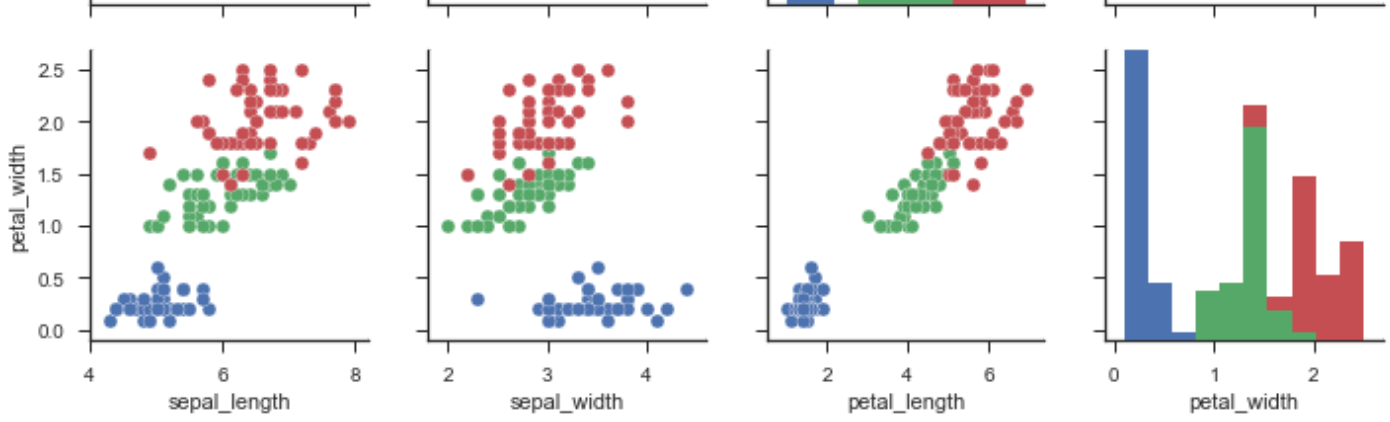
Locating Natural Clusters

Exploring Complex Patterns

In [4]:
```
%matplotlib inline
import seaborn as sns
import seaborn as sns
sns.set(style="ticks")
df = sns.load_dataset("iris")
%matplotlib inline
sns.pairplot(df, hue="species")
```

Out[4]:
```
<seaborn.axisgrid.PairGrid at 0x231dfb9f2e8>
```




petal_width
2.5
2.0
1.5
1.0
0.5
0.0
4
6
8
sepal_length
2
3
4
sepal_width
2
4
6
petal_length
0
1
2
petal_width

Motivation (2)

- A data-driven approach to learning from data.
- Leveraging big unstructured and unlabeled data.
- It is "free", and exploits the information inherent in the internal structures of the data.

Motivation(3)

- Powerful approaches for generating complex features into a supervised model.
  - Complex combinations that would be impossible for the supervised model to search for, at least in relation to the number of labels --> overfitting.
- Curse of dimensionality, that our models cannot search through the combinatorial space. After dimensionality reduction we assume that the relevant structures are kept, and that we can combine these.

Learning from unlabeled data example

- Learning from Unlabeled data + Transfer learning: https://blog.openai.com/unsupervised-sentiment-neuron/ (https://blog.openai.com/unsupervised-sentiment-neuron/)

Agenda

- Unsupervised learning for Exploratory data analysis
- Clustering algorithms from complex to categorical.
- Dimensionality reductino for exploratory analysis.
- small sidestep on interactive plots.

Supervised vs Unsupervised

Dinstinquishing the concepts

Supervised learning:

* The idea of training a model that transforms an input X to an output Y.
* Y is known and defined by the human. How to make the mapping function is not, only the constraints and architecture is defined.

Unsupervised Learning

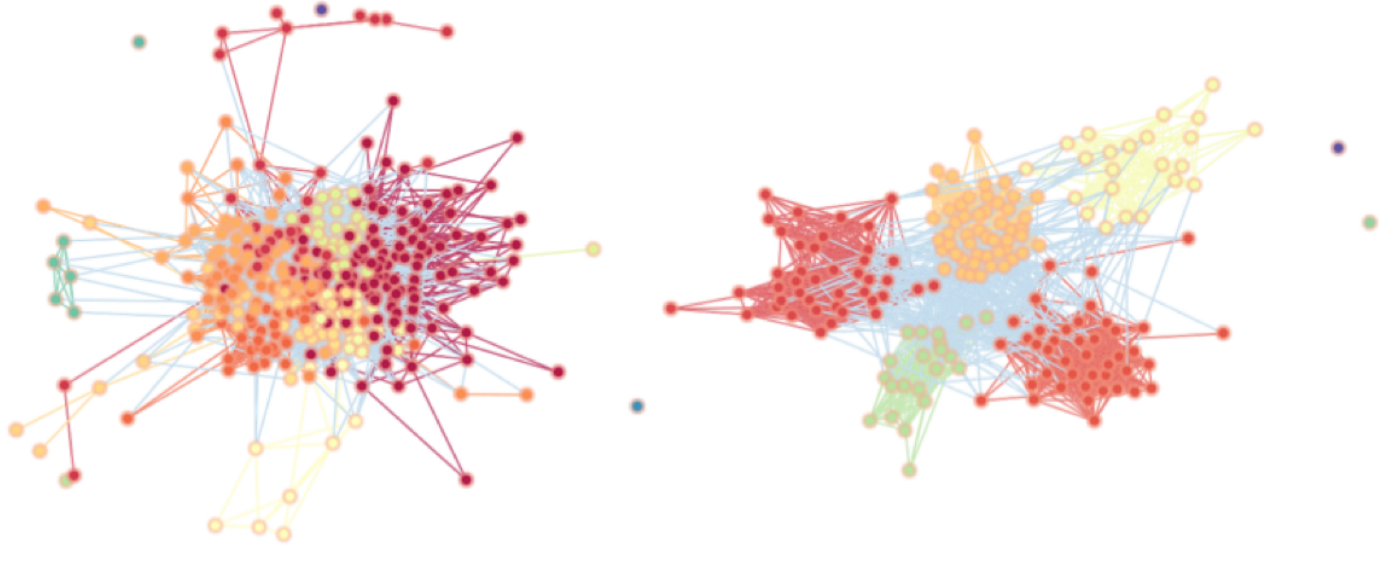
* The idea of training a model to transform an input X to an output Y.
* Y is unknown. The training of the model, is not mapped directly to a Concept or Label, but instead is defined by Internal, endogenous structure, an artificial optimization function.

Clustering

From complex multidimensional patterns to Categorical data

Networks

- You belong to a group, and that is essentially important to your way of looking at the world. Locating this group in a network means understanding your information flows, your preferences, and your main loyalties.



From complex multidimensional patterns to Categorical data (2)

Segments

- Segments of people and their preference define a recognizable *Culture*, a cultural grouping.

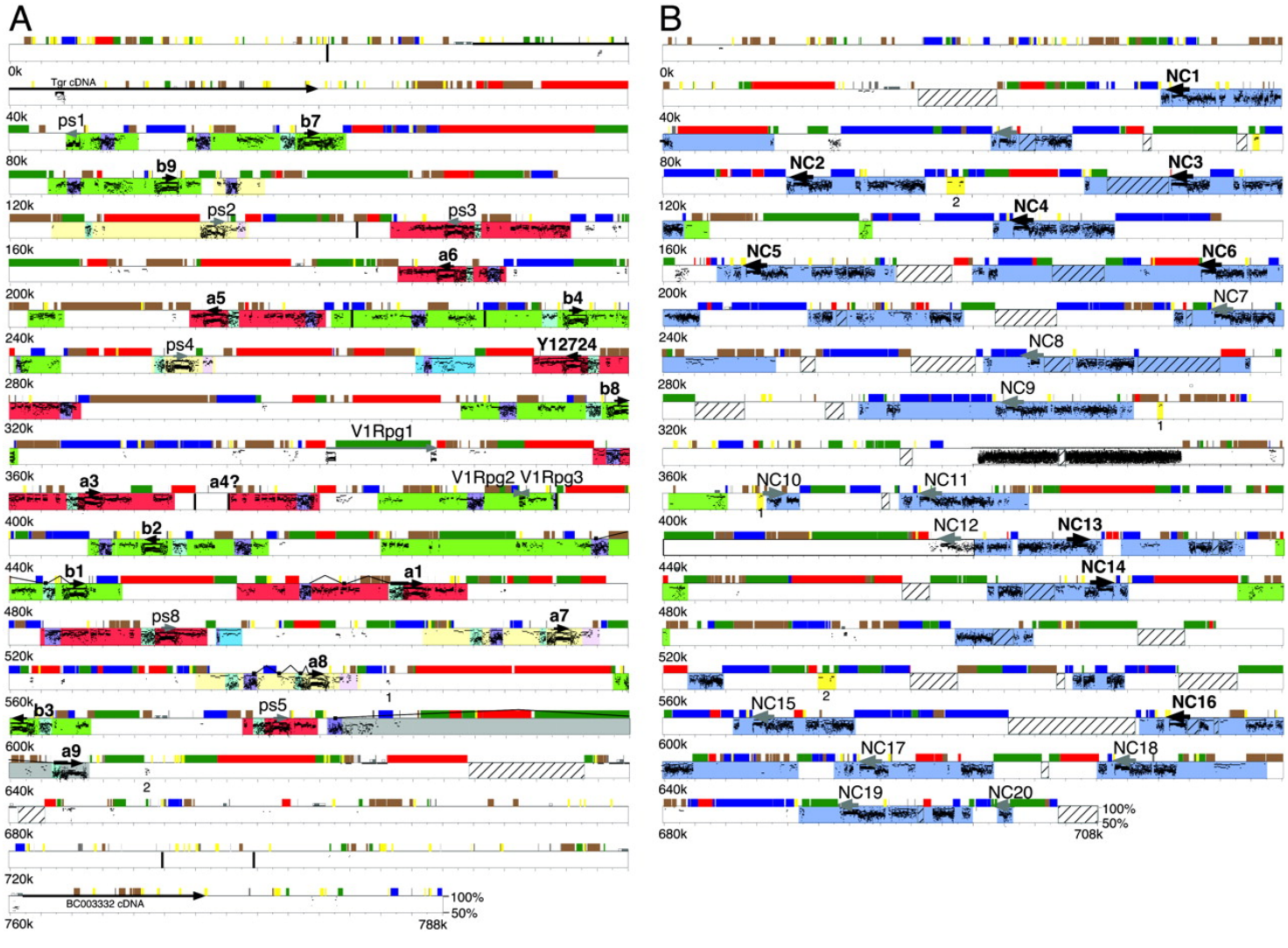
examples:

```
* Movie recommendatation systems.
* Product recommendation.
```

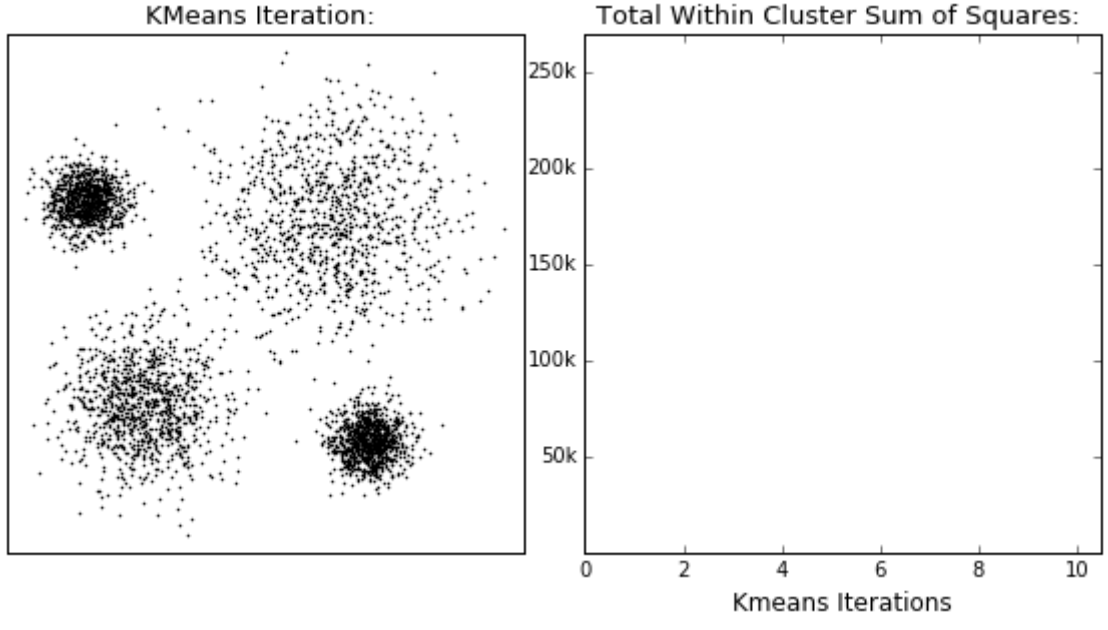
From complex multidimensional patterns to Categorical data (3)

Trajectories

- Sequence analysis: Clustering trajectories to understand archetypical careers.



An example - K-Means clustering



Credit David Sheehan

Assuming that the data can be expressed as K centers, and each point being assigned to the nearest center. We begin our search for the optimal centers.

By initializing the algorithm at K random points, we can define our first *Expectation* of the Clusters, by locating the Voronoi set of points to each initial cluster center. I.e. we assign each data point to the nearest cluster center.

The *Maximization* step, then assummes that optimal clusters will be located by moving the original cluster center to the mean of the Voronoi set.

By iterating n times we should converge on one solution.

Types of Clustering

- Hierarchical clustering: Agglomerative Linkage
- Network Clustering: Modularity based clustering.
- Hard or Soft assignment.
- Non-linear Clustering: Spectral Kmeans.

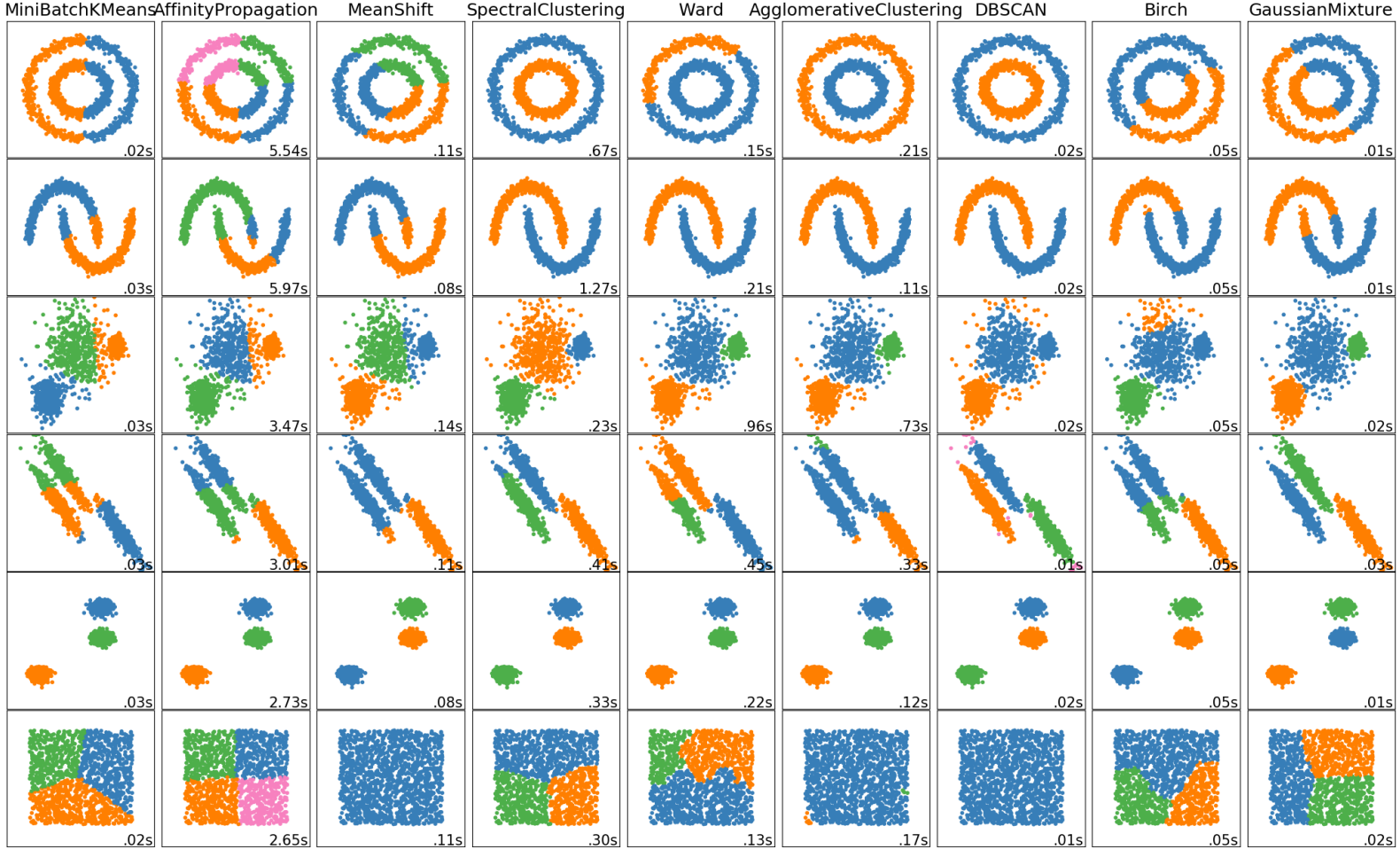
Status of unsupervised methods

As powerful and enticing as unsupervised methods are: do not let them fool you into believing that the clusters and latent dimensions are natural pr. definition, that this apriori data-driven approach is much more true than finding what you are searching for with aposteriori labels and supervised learning.

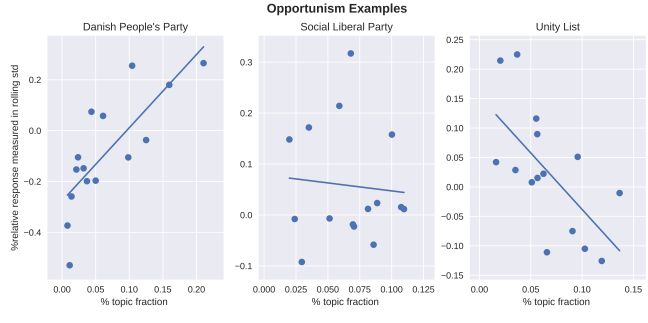
Clusters are artefacts arising from a complex interaction between the distributions and patterns in your data, and the specific algorithm and set of hyperparameters used.

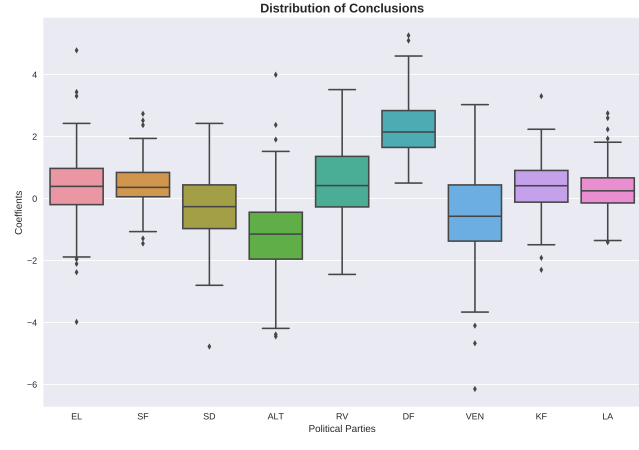
Many algorithms might have proven as useful heuristics for clustering users and consumers, for recommendation and ranking of products, or as feature generators, but without explicit evaluation on random samples using human benchmarks we can not really now ***how good*** they are, ***how fair*** or ***biased*** they are. And at most it should there for be treated as a ***prototype***.

A display of artefacts



Distribution of Conclusions





Dimensionality reduction for exploratory data analysis

Exploratory analysis in highly multidimensional data. When the scatter matrix becomes to complex, we want to visualize latent patterns in the data.

Both to understand what our model can learn, and to see if there is indeed anything to learn from a given representation.

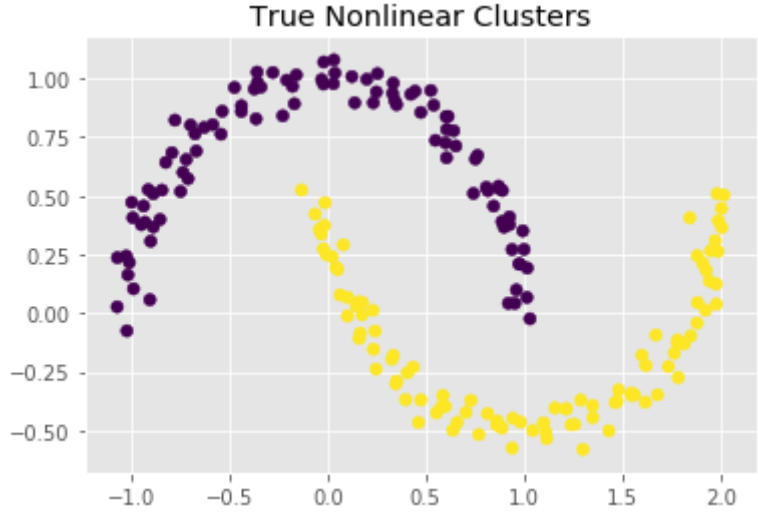
Transformations to lower dimensional space that preserves the variance.

- Methods include more classical social science tools: Preservig the structures of global and linear distance matrix.
    - Principal Component Analysis - PCA
    - NonNegativeMatrixFactorization - NNMF and
- more complex nonlinear models, Manifold Learning and Metric Learning.
    - Preserving the structures of local and non linear distance matrices.

Understanding these approaches are not within the scope of the current course.

In [9]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
from sklearn.datasets import make_moons
X, y = make_moons(200, noise=.05, random_state=0)
plt.scatter(*zip(*X),c=y,cmap=plt.cm.viridis)
plt.title('True Nonlinear Clusters')
```

Out[9]:
```
Text(0.5,1,'True Nonlinear Clusters')
```



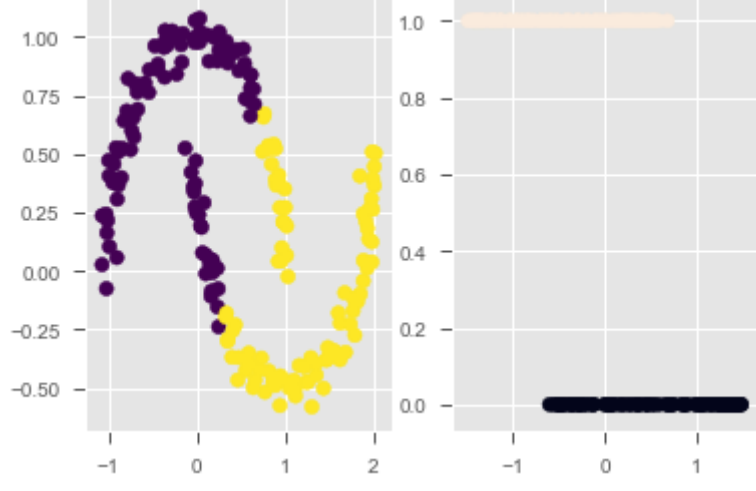
what meaning might we make of this distribution?

Is it resulting from a linear process?

Which are the natural neighbors of the points and how to extract clusters?

In [33]:
```
# We can try Our Standard Linear Approaches.
from sklearn.cluster import KMeans
labels = KMeans(2, random_state=0).fit_predict(X)
fig,axes = plt.subplots(1,2)
axes[0].scatter(X[:, 0], X[:, 1], c=labels,
            s=50, cmap='viridis');
from sklearn.decomposition import PCA
pca = PCA(n_components=1)
x = pca.fit_transform(X)
axes[1].scatter(x,y,c=y) #
```

Out[33]:
```
<matplotlib.collections.PathCollection at 0x231e666ab38>
```



Using a simple idea of centers and linear distances we cannot recover the "obvious" clusters.

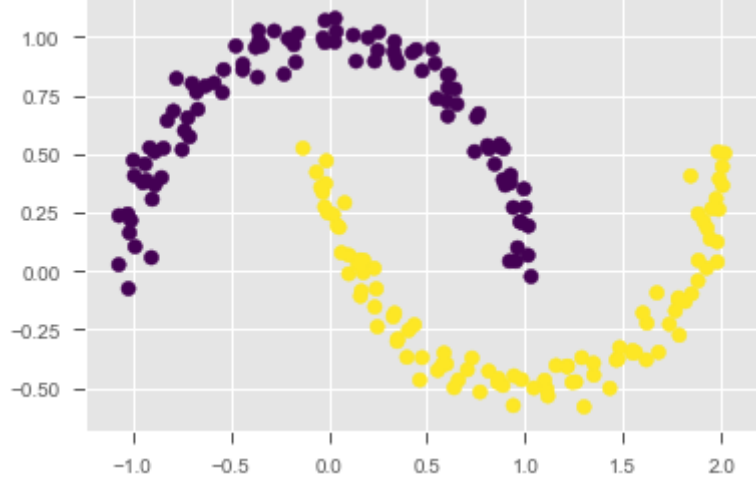
We need to redefine distances, incorporating the sparsity in the data space. I.e. the empty wholes between the two moons.

Many approaches do this by only focusing on local spaces defines by nearest neighbors.

In [34]:

```
from sklearn.cluster import SpectralClustering
model = SpectralClustering(n_clusters=2, affinity='nearest_neighbors',
                           assign_labels='kmeans')
labels = model.fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels,
            s=50, cmap='viridis');
```

```
C:\Users\jbv933\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\manifold
\spectral_embedding_.py:234: UserWarning: Graph is not fully connected, spectral embe
dding may not work as expected.
  warnings.warn("Graph is not fully connected, spectral embedding"
```



Example of an Exploratory Analysis - Spatialization

Two widely used approaches

TSNE: t-distributed stochastic neighbor embedding

```
* Widely used algorithm for exploring highdimensional data.
```

- Problem lacks interpretable global properties.

and the new player solving this issue: UMAP: Uniform Manifold Approximation and Projection

Lets us give them a spin.

First we need to construct the Distance Matrices

In [39]:

```
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
pca_embedding_idf,tsne_embedding_idf,umap_embedding_idf = np.load('tfidf_dimensionality_
reduction.npy')
pca_embedding,tsne_embedded,umap_embedding = np.load('w2vec_dimensionality_reduction.np
y')
```

In [37]:
```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/snorreralund/scraping_seminar/master/english_review_sample.csv')
```

Interactive plotting using plotly

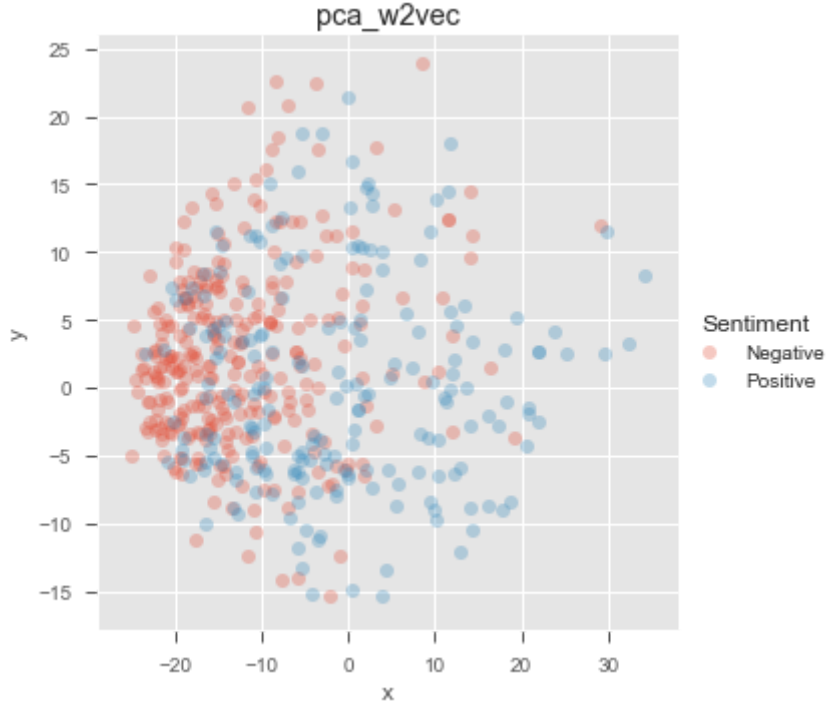
In [41]:

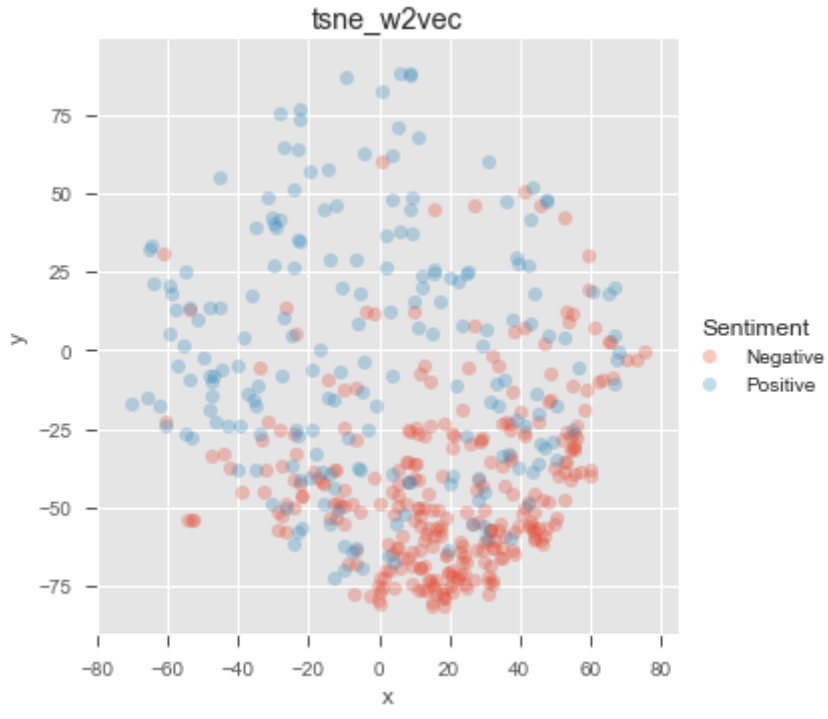
```
import plotly.offline as py # import plotly in offline mode
py.init_notebook_mode(connected=True) # initialize the offline mode, with access to the
 internet or not.
import plotly.tools as tls
tls.embed('https://plot.ly/~cufflinks/8') # embed cufflinks.
# import cufflinks and make it offline
import cufflinks as cf
cf.go_offline() # initialize cufflinks in offline mode
```

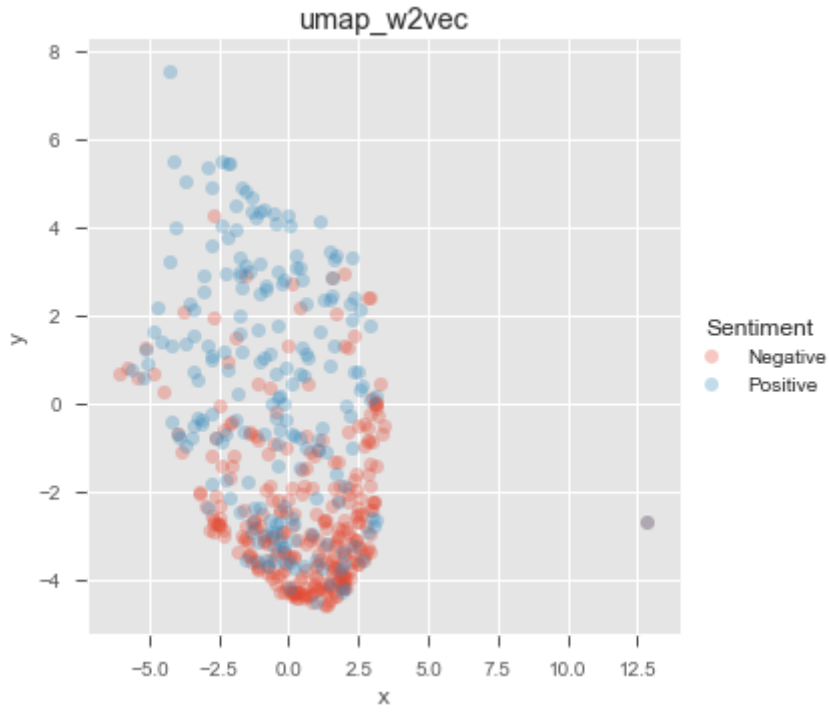
In [43]:

```
def build_hovertext(row): # function for constructing the hovertext
    """function for constructing the hovertext combining the rating value and the review
 body.
    Further more it Replaces standard newline character \n with html newline tag <br>"""
    return 'Rating:%d <br><br>Review:%s'%(row['reviewRating_ratingValue'],'<br>'.join(ro
w['reviewBody'].split('\n')))
def plot_embedding(df,embedding,title,n=1000,max_string=500,interactive=False):
    "Helper function (i.e. not a general purpose function) for plotting the data."
    df['x'] = embedding[:,0]
    df['y'] = embedding[:,1]
    df['Sentiment'] = df['reviewRating_ratingValue'].apply(lambda x: 'Positive' if x>3 e
lse 'Negative')
    df = df.groupby('reviewRating_ratingValue').apply(lambda x:x.sample(100))
    #df = make_color_scale(df,'reviewRating_ratingValue')
    df['text'] = df.apply(build_hovertext,axis=1)
    if interactive:
        df.iplot(kind='scatter',x='x',y='y',categories='reviewRating_ratingValue',text=
'text',title=title,colorscale='OrRd')#,layout=layout)
    else:
        sns.lmplot(x='x',y='y',hue='Sentiment',data=df,fit_reg=False,scatter_kws={'alph
a':0.3})
        plt.title(title)
```

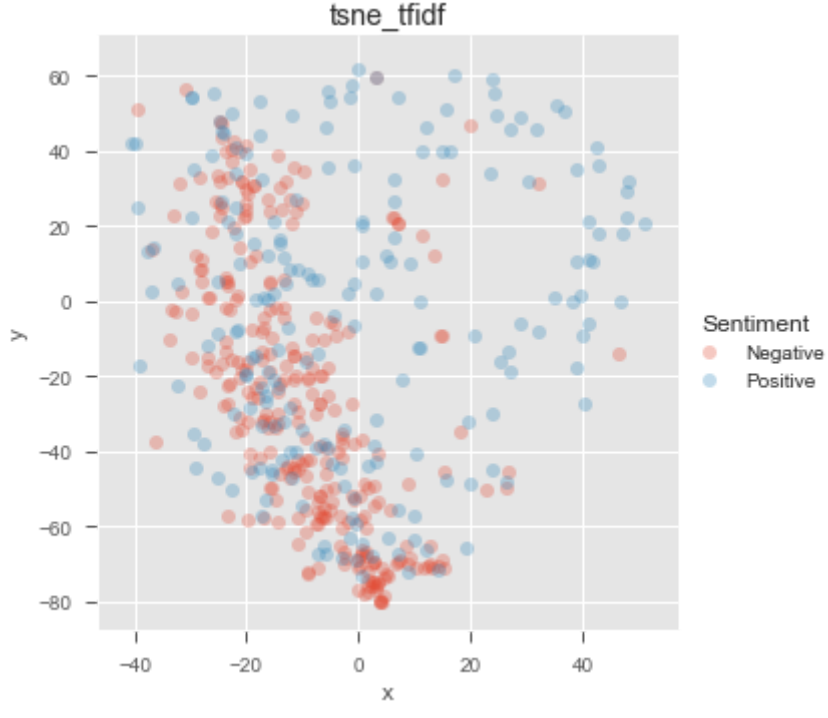
In [44]:
```
embeddings = {'pca':pca_embedding,'pca_tfidf':pca_embedding_idf
                  ,'tsne':tsne_embedded,'tsne_tfidf':tsne_embedding_idf,
                 'umap':umap_embedding,'umap_tfidf':umap_embedding_idf}
for name,embedding in enumerate(sorted(embeddings.items(),key=lambda x: len(x[0]))):
    if not '_' in name:
        name = name+'_w2vec'

    plot_embedding(df,embedding,name,interactive=False)
```




tsne_w2vec
75
50
25
0
−25
−50
−75
y
−80
−60
−40
−20
0
20
40
60
80
x
Sentiment
Negative
Positive


umap_w2vec
8
6
4
2
0
−2
−4
y
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
x
Sentiment
Negative
Positive


pca_tfidf
5
4


tsne_tfidf
60
40
20
0
−20
−40
−60
−80
y
−40
−20
0
20
40
x
Sentiment
Negative
Positive

In [46]:
```
#plot_embedding(df,umap_embedding,'umap_embedding_w2vec',interactive=True)
```

In [40]:
```
layout = {'autosize':False,
'width':900,
'height':600} # fix problem when converting to slides
```