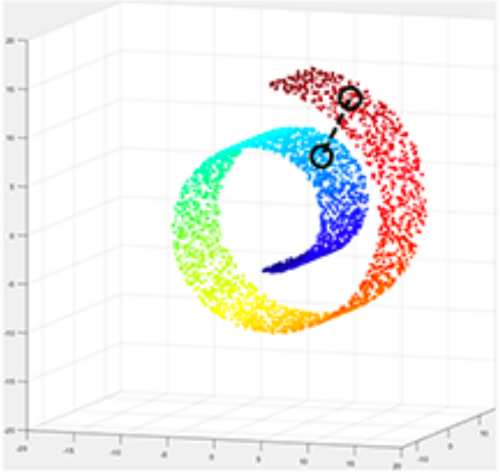
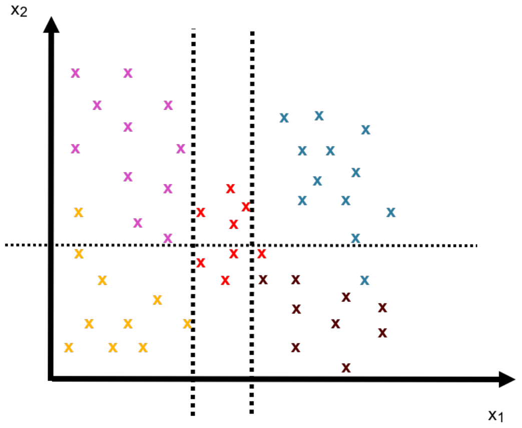
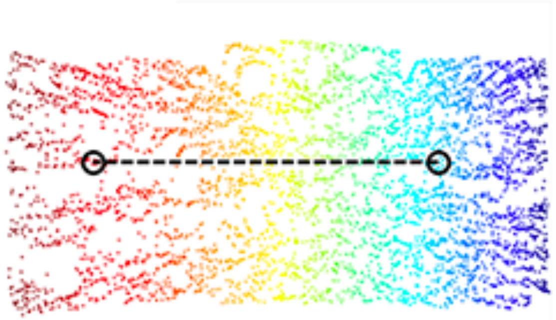
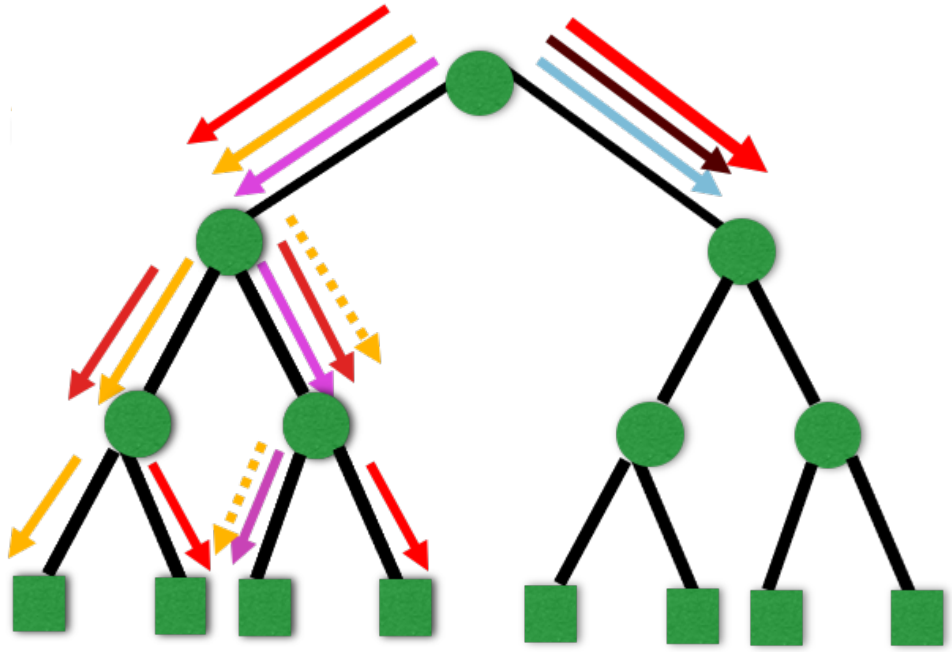
Machine Learning for Biomedical/Healthcare Applications









Laplacian Eigenmaps and Decision Trees

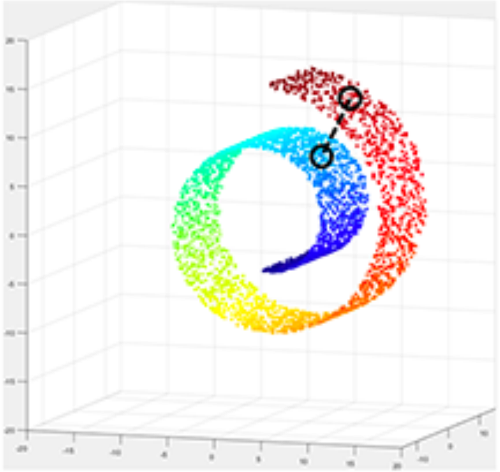
Dr Emma Robinson

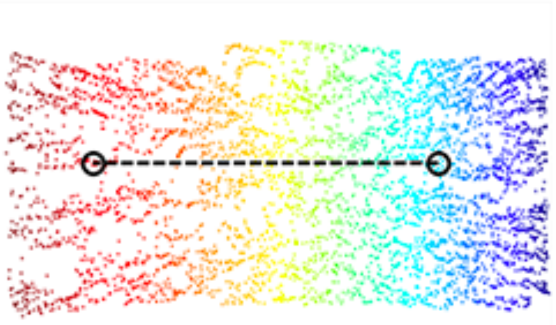
Learning Objectives

1. Gain an **intuition for the reasoning behind manifold learning** and define the conditions, under which it is needed, over linear techniques such as PCA
2. Learn how to implement **Laplacian Eigenmaps** from scratch and scikit learn
3. Be able to define what is meant by a weak learner, a **decision stump** a **decision tree**
4. Be able to define and use **weak learning rules: Information Gain and Gini Index**,
5. Learn step by step how to construct a **decision tree classifier from scratch** and using scikit-learn

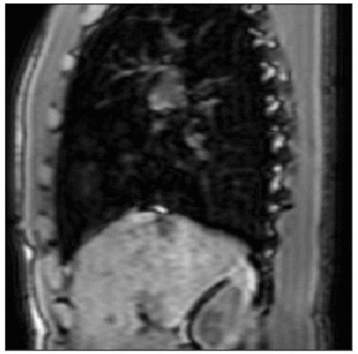
Recap – Manifold learning

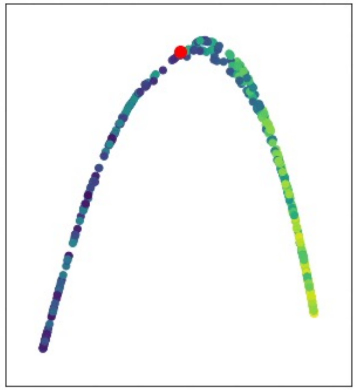
- To learn to unroll data assumed to lie on a curved manifold in high-dimensional space





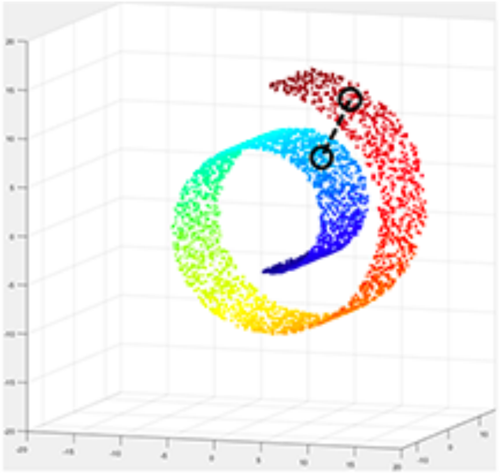
- In the real world high dimensional data often lies on a lower-dimensional subspace
- Reflecting the physical laws that explain it's behaviour
- i.e. subsequent photos of a panoramic photo or a cine of lung/heart motion will relate to each other by laws of motion

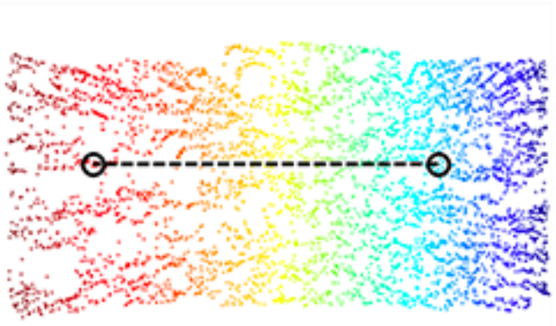




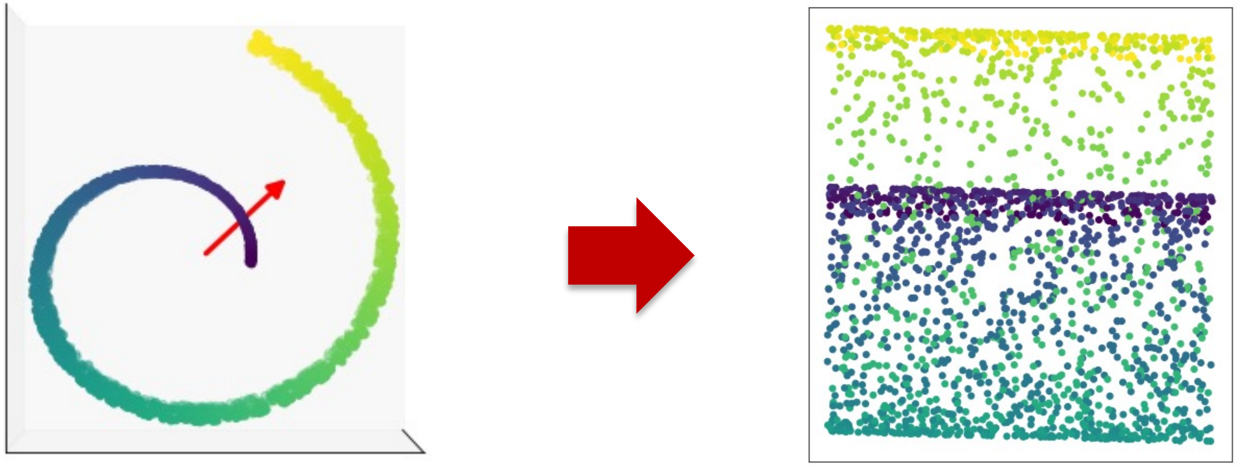
Recap – Manifold learning

- To learn to unroll data assumed to lie on a curved manifold in high-dimensional space





- Look at the similarity of *neighbouring data points only*
- Not the whole data distribution unlike PCA



Exercise 1 - applying PCA to the swiss role data set

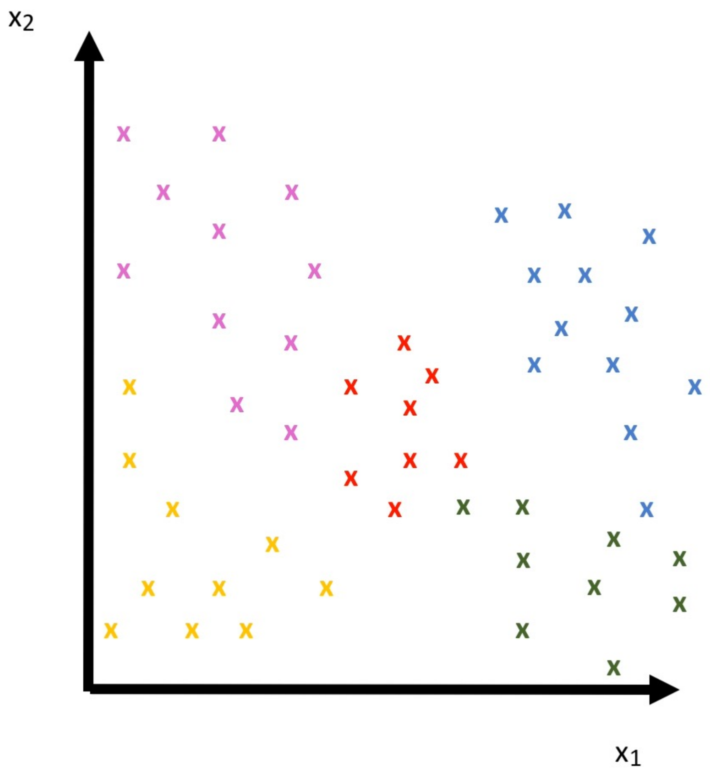
Today manifold learning (first hour)

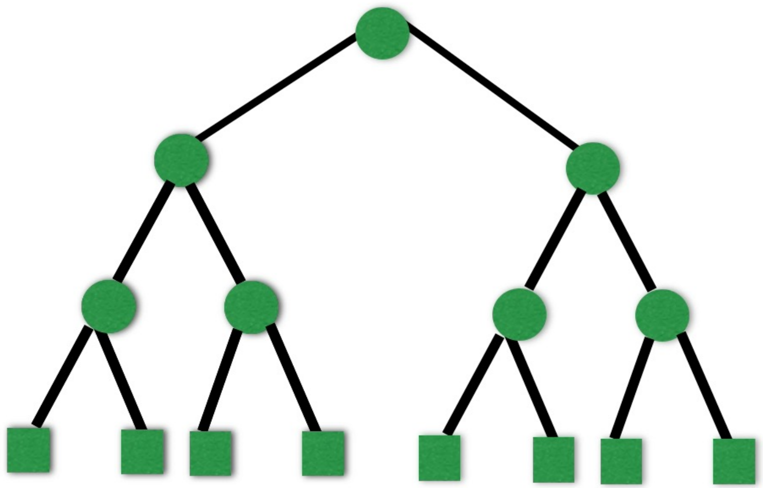
- Implement Laplacian Eigenmaps (spectral clustering) for swiss roll problem
  6.1.Laplacian_Eigenmaps.ipynb

  1. Estimate *symmetric* k_NN graph
     - Get nearest neighbours by estimating difference using sum of square differences
     - Binarise
     - Symmetrise
  2. Estimate Degree $\mathbf{D}_{\mathrm{ii}} = \sum_j \boldsymbol{A}_{\boldsymbol{ij}}$
  3. Estimate Laplacian $\boldsymbol{L} = \mathbf{D} - \mathbf{A}$
  4. Implement in scikit learn

Recap decision trees

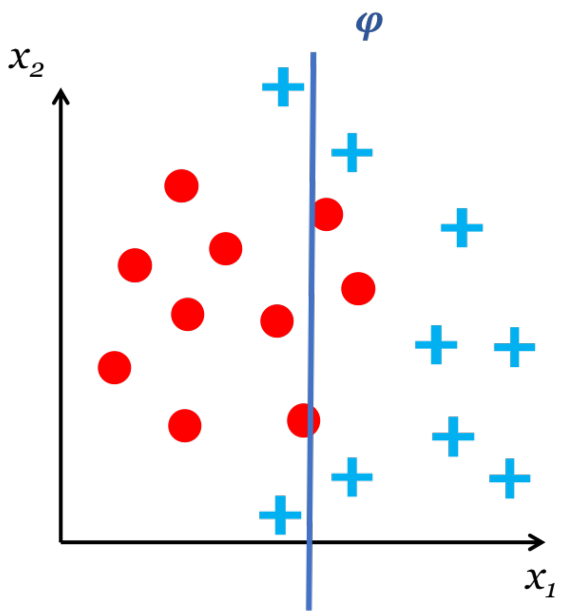
- Simple ‘axis-aligned’ learning rules based on thresholding individuals features
- Non-linear
- Interpretable



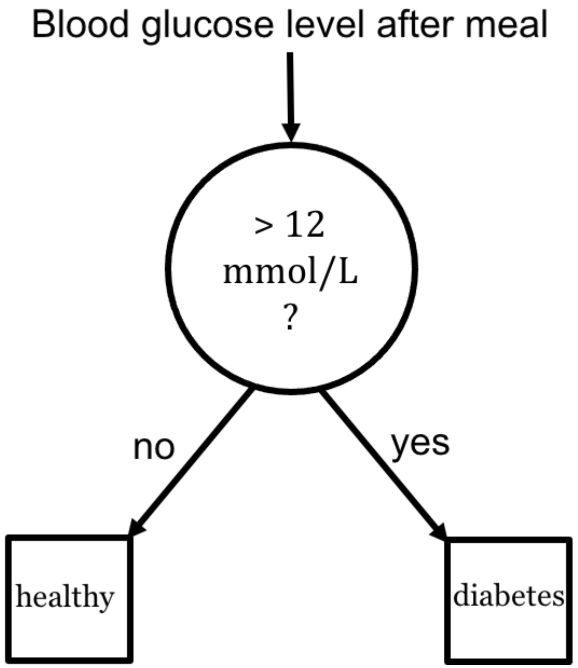


Today – implement decision stump learning in python

- **6.2.Decision_Trees.ipy**
- Try different features and thresholds
- Calculate gini index
- Weight by proportion of data in each branch



$$\begin{array}{cc} & \text{feature} \quad \text{class} \\ \cancel{dataset} = & \begin{pmatrix} 1 & 0 \\ 2 & 0 \\ 3 & 0 \\ 4 & 0 \\ 5 & 0 \\ 6 & 1 \\ 7 & 1 \\ 8 & 1 \\ 9 & 1 \\ 10 & 1 \end{pmatrix} \end{array}$$



Tutorial: Decision tree/stump - Optimisation:

$$dataset = \begin{pmatrix} 1 & 0 \\ 2 & 0 \\ 3 & 0 \\ 4 & 0 \\ 5 & 0 \\ 6 & 1 \\ 7 & 1 \\ 8 & 1 \\ 9 & 1 \\ 10 & 1 \end{pmatrix}$$

At each tree node j:

For each possible feature k:

I.e. try thresholding on each feature value

For each possible threshold τ on this feature:

Evaluate function $I(S_j, k)$

This estimates the cost of splitting training data into left/right* components (S_j^1, S_j^1)
*: leaf 1, leaf 2

Chose the best feature (k_{opt}) corresponding to optimum cost (I_{opt}) and threshold τ_{opt}

Split data according to this function

until termination criterion is met

REPEAT*

Assign leaf nodes majority label of training examples

Important points to remember!

1. The total data coming into the node needs to be split into two subsets – one for each branch
2. **Try $I(S_j, k)$ on each value of the column X[:,k]** to find your optimum τ
3. The **labels are in the last column of the data**
4. **Gini index:** Indicates how mixed the classes are following the split.

$$I(S_j) = \sum_i \frac{|S_j^i|}{|S_j|} Gini_i \qquad \text{where} \qquad \text{Gini} = 1 - \sum_{y_k \in Y} p(y_k)^2$$

- $p(y_k)$ is proportion at given node that are of class k ;
- Final score is weighted by proportion of total samples in that branch

I Laplacian Eigenmaps

Open 6.1.Laplacian_Eigenmaps.ipynb ; To do

1. Exercise 2 - Create a *symmetric* k-Nearest neighbour graph (30 mins)
2. Exercise 3 - implement the Laplacian Eigenmaps embedding of the swiss roll data set (10 mins)
3. Exercise 4 - Implementing Laplacian Eigenmaps with Scikit Learn (10 mins)

II Decision stumps and Trees

Open **6.2.Decision_Trees.ipynb** ; To do

1. Exercise 1 building a decision stump classifier from scratch (30-45 mins)
2. (optional) Exercise 2: building and testing a complete decision tree (15 mins)
3. Exercise 3 - Running Decision Trees with Scikit-Learn (15 mins)
    1. (optional) Compare against the decision tree built in Exercise 2

Additional exercises

- Try constructing a regression tree from scratch; using the above classification tree as the basis but:
  - creating a new MSE cost, and
  - editing the prediction function accordingly (to fit constant function to mean);
- Try it out the following toy dataset (Taken from: http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html#sphx-glr-auto-examples-tree-plot-tree-regression-py)

```
# Create a random dataset
rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - rng.rand(16))
```

- Compare your result