Machine Learning BLG454E, May, 27 2020 120mins, Fake Exam ☺
Duration: 100 minutes.

*Closed books and notes. Write your answers neatly **in the space provided for them**. Write your name on each sheet. Good Luck ☺*

Q1	Q2	Q3	Q4	Q5	Q6	Q7	TOTAL
10	**4**	**14**	**20**	**30**	**2**	**30**	**100**

Question 1 [10 points —very easy]

a) Point out two key differences between classification and regression machine learning problems. [2 points]

b) Given a dataset $D = \{\boldsymbol{x^i}, y^i\}_{i=1}^{50}$, where $\boldsymbol{x^i} \in \mathbb{R}^d$ denotes the feature vector for sample i and its corresponding class label $y^i \in \{-1,1\}$, we train a soft perceptron model on D, using:

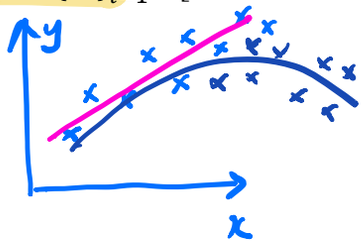
- 5-fold (5F) cross-validation strategy, and
- 10-fold (10F) cross-validation strategy.

We will learn the same model parameters using 5-F and 10F: true or false? [2 points]

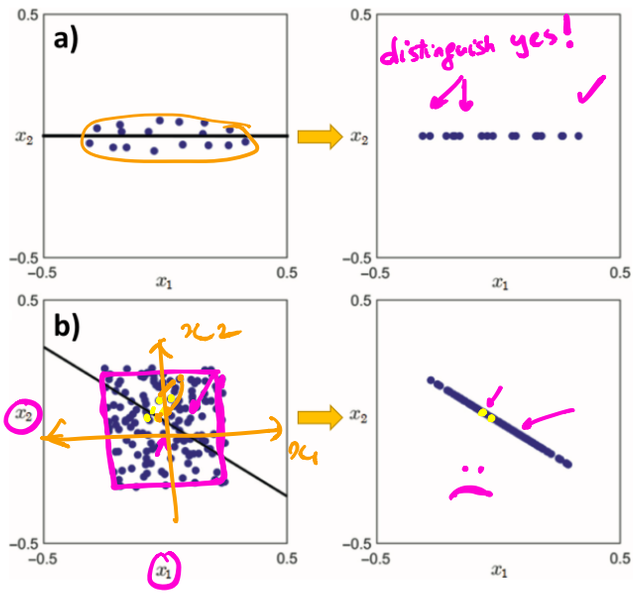
The classification accuracy using 5F and 10F will be identical: true or false? [2 points]

c) Given that we have only 50 samples (25 in class 1 and 25 in class 2), which cross-validation strategy is more suited to train the linear soft perceptron: 5F or 10F? Justify your choice. [3 points]

d) Given a particular training dataset $\{\boldsymbol{x^i}, y^i\}_{i=1}^{n}$ where $\boldsymbol{x^i} \in \mathbb{R}^d$ denotes the feature vector for sample i and its corresponding target value $y^i \in \mathbb{R}$, how can we know if a linear or nonlinear regression model is best to choose to predict the outputs $\{y^i\}_{i=1}^{n}$ from the inputs $\{\boldsymbol{x^i}\}_{i=1}^{n}$? [*For simplicity, we hypothesize that each sample has a single feature.*] [1 point]



Question 2 [4 points —easy]



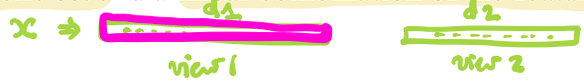
Each dataset (a) and (b) is mapped onto a lower dimensional space using PCA. The black line represents the first principle component (or eigenvector).

If you were to choose a dimensionality reduction technique for each dataset to train a linear classifier, would PCA be a good option? **Justify your choice for each dataset.**

Question 3 [14 points —medium → difficult]

In many computer vision systems, the same object can be observed at varying viewpoints or even by different sensors, which brings in the challenging demand for recognizing objects from distinct even heterogeneous views.

Canonical correlation analysis –CCA– is a means of assessing the relationship between two datasets of features. The idea is to study the correlation between a linear combination of the features in one set and a linear combination of the features in another set.



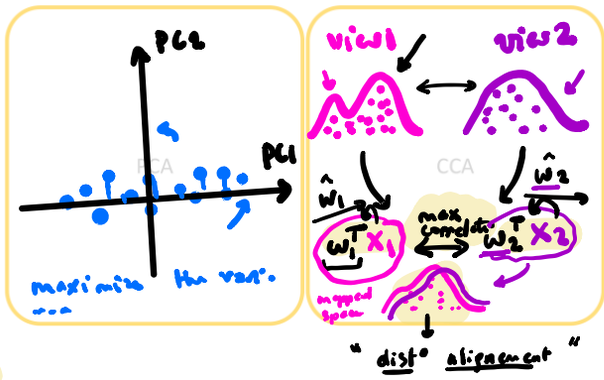
CCA attempts to find a common subspace where the samples from two views are most correlated. Formally, let $\mathcal{S}$ represent the samples from two views: $\mathcal{S} = \{(\mathbf{x}_{11}, \mathbf{x}_{12}), \ldots, (\mathbf{x}_{n1}, \mathbf{x}_{n2})\}$, where $\mathbf{x}_{ij} \in \mathbb{R}^{p_j}$ $j = 1, 2$, represents the ith sample from the jth view of p_j dimension. Two matrices $\mathbf{X}_1 = [\mathbf{x}_{11}, \ldots, \mathbf{x}_{n1}]$ and $\mathbf{X}_2 = [\mathbf{x}_{12}, \ldots, \mathbf{x}_{n2}]$ are defined to represent the data from the two views. Two linear transforms $\mathbf{w}_1$ and $\mathbf{w}_2$ are obtained to respectively project the samples from two views into the common subspace, by maximizing the correlation between $\mathbf{w}_1^T\mathbf{X}_1$ and $\mathbf{w}_2^T\mathbf{X}_2$ as below:

$$\max_{\mathbf{w}_1,\mathbf{w}_2} \mathbf{w}_1^T\mathbf{X}_1\mathbf{X}_2^T\mathbf{w}_2 \quad s.t. \ \mathbf{w}_1^T\mathbf{X}_1\mathbf{X}_1^T\mathbf{w}_1 = 1 \ \mathbf{w}_2^T\mathbf{X}_2\mathbf{X}_2^T\mathbf{w}_2 = 1. \quad (1)$$

With the Lagrange multiplier, Eq. (1) can be solved by resorting to the eigenvalue decomposition.

[Kan et al. TPAMI 2016]

1) Illustrate how PCA works 2) Illustrate how CCA works



3) Point one similarity and one difference between PCA and CCA

Let note: $\mathbf{C}_{12} = \mathbf{X}_1\mathbf{X}_2^T$ denotes the cross-covariance matrix of datasets $\mathbf{X}_1$ and $\mathbf{X}_2$. $\mathbf{C}_{11} = \mathbf{X}_1\mathbf{X}_1^T$ denotes the covariance matrix of $\mathbf{X}_1$.

4) Given the following derivation rules, rewrite (1) using $\mathbf{C}_{12}$ and solve the primal Lagrangian optimization problem of equation (1)

$$\frac{\partial \mathbf{x}^T\mathbf{A}\mathbf{y}}{\partial \mathbf{x}} = \mathbf{A}y \qquad \frac{\partial \mathbf{x}^T\mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = \mathbf{A}x \qquad \frac{\partial \mathbf{x}^T\mathbf{A}\mathbf{y}}{\partial \mathbf{y}} = \mathbf{A}^T x$$

5) Solving this constrained system will lead to: $\mathbf{C}_{22}^{-1}\mathbf{C}_{21}\mathbf{C}_{11}^{-1}\mathbf{C}_{12}\mathbf{w}_2 = \alpha^2\mathbf{w}_2$

Geometrically, what does the optimal vector $\mathbf{w}_2$ represent?

$$\max_{w_1,w_2} w_1^T C_{12} w_2 \quad s.t. \ \alpha_1 f(w_1) = w_1^T C_{11} w_1 - 1 = 0, \ \alpha_2 g(w_2) = w_2^T C_{22} w_2 - 1 = 0$$

$$\mathcal{L}_p(w_1, w_2) = w_1^T C_{12} w_2 - \alpha_1 w_1^T C_{11} w_1 - \alpha_2 w_2^T C_{22} w_2 + 2$$

$$\textcircled{1} \ \frac{\partial \mathcal{L}_p}{\partial w_1} = C_{12} w_2 - 2\alpha_1 C_{11} w_1 = 0$$

$$\textcircled{2} \ \frac{\partial \mathcal{L}_p}{\partial w_2} = C_{21} w_1 - 2\alpha_2 C_{22} w_2 = 0$$

$$\begin{cases} C_{12} w_2 = 2\alpha_1 C_{11} w_1 & \times w_1^T \\ C_{21} w_1 = 2\alpha_2 C_{22} w_2 & \times w_2^T \end{cases} \qquad \begin{cases} w_1^T C_{11} w_1 = 1 \\ w_2^T C_{22} w_2 = 1 \end{cases}$$

$$w_1^T C_{12} w_2 = 2\alpha_1 = w_2^T C_{21} w_1 = 2\alpha_2 \Rightarrow \alpha_1 = \alpha_2 = \alpha \Rightarrow \alpha = 2\alpha$$

$$\begin{cases} C_{12} w_2 = \alpha C_{11} w_1 \\ C_{21} w_1 = \alpha C_{22} w_2 \end{cases} \Rightarrow \frac{1}{\alpha} C_{11}^{-1} C_{12} w_2 = w_1 \Rightarrow C_{22}^{-1} C_{21} w_1 = \alpha w_2$$

$$C_{22}^{-1} C_{21} C_{11}^{-1} C_{12} w_2 = \alpha^2 w_2$$

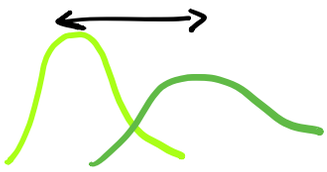
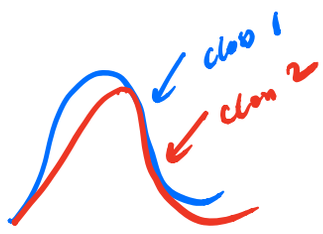
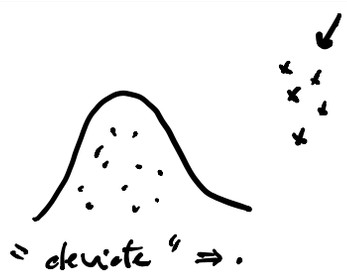
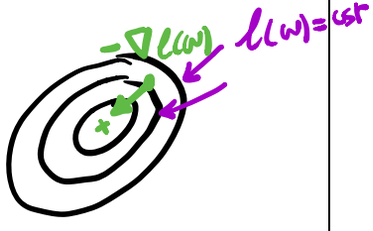
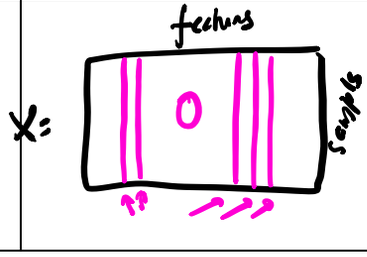
$$C_{22}^{-1} C_{21} C_{11}^{-1} C_{12} w_2 = \alpha^2 w_2 \Rightarrow w_2 = \alpha^2 w_2$$

w_2 ↓ eigen

$Ax = \lambda x$

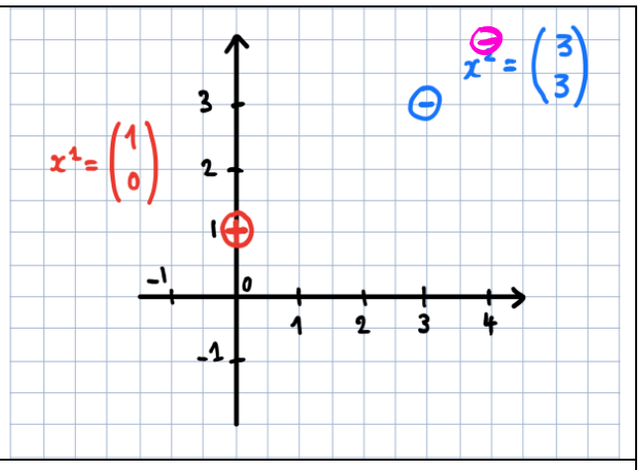
Question 4 [20 points —easy]

Define the following terms and illustrate the concept in each box.

Concept	Simple definition	Illustration
Data fracture (or shift)		
Overlapping classes		
Outliers		
Level curves of a loss function to minimize and **directionality** of the loss gradient		
Sparse feature selection		

Question 5 [30 points —medium]

a) Given the two following training samples, provide below a step-by-step solution to estimate the optimal parameters ($\boldsymbol{w}$ and b) of the hyperplane separating the two classes. [20 points]



Step-by-step solution:

Step 1:

$$\begin{cases} x^1 = \begin{pmatrix}1\\0\end{pmatrix} & x^2 = \begin{pmatrix}3\\3\end{pmatrix} \\ y_1 = 1 & y_2 = -1 \end{cases}$$

① Define the primal Lagrangian:

$$\mathcal{L}_p = \sum_{i=1}^{2} \alpha_i - \frac{1}{2} \sum_{i=1}^{2}\sum_{j=1}^{2} \alpha_i \alpha_j y_i y_j x^{iT} x^j$$

(1,1) (1,2) (2,1) (2,2)

$$= \alpha_1 + \alpha_2 - \frac{1}{2}\left[\alpha_1^2 - 3\alpha_1\alpha_2 - 3\alpha_2\alpha_1 + 18\alpha_2^2\right]$$

$$\mathcal{L}_p = \alpha_1 + \alpha_2 - \frac{1}{2}\left[\alpha_1^2 + 18\alpha_2^2 - 6\alpha_1\alpha_2\right]$$

② Define the dual Lagrangian to derive using the primal constraint:

- primal constraint $\sum_{i=1}^{n} \alpha_i y_i = 0 \Rightarrow \alpha_1 - \alpha_2 = 0 \Rightarrow \alpha_1 = \alpha_2$

↓ substitute and define dual Lagrangian

$$\mathcal{L}_d = \alpha_1 + \alpha_1 - \frac{1}{2}\left[\alpha_1^2 + 18\alpha_1^2 - 6\alpha_1^2\right]$$

$$\mathcal{L}_d = 2\alpha_1 - \frac{13}{2}\alpha_1^2$$

③ Deriving dual Lagrangian wrt dual variable(s):

$$\frac{\partial \mathcal{L}_d}{\partial \alpha_1} = 2 - 13\alpha_1 = 0 \Rightarrow \alpha_1 = \frac{2}{13}$$

→ $\alpha_1 = \alpha_2 = 2/13$ ⇒ both are support vectors.

④ $w^* = \sum_{i=1}^{2} \alpha_i y_i x^i = \frac{2}{13}\left(\begin{bmatrix}1\\0\end{bmatrix} - \begin{bmatrix}3\\3\end{bmatrix}\right) = \begin{bmatrix}-4/13\\-6/13\end{bmatrix}$

⑤ $b^* = ?$

$y_i (w^* \begin{bmatrix}1\\0\end{bmatrix} + b^*) = 1$

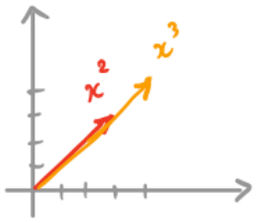
$y_i = 1$

$\Rightarrow -4/13 + b^* = 1$

$\Rightarrow b^* = 17/13$

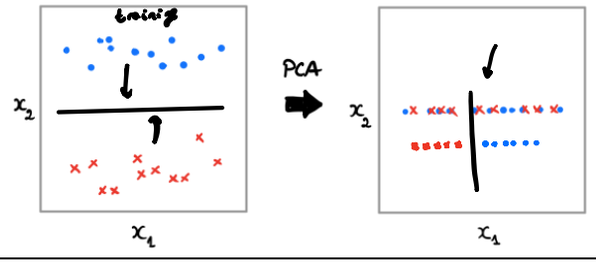
b) If we add a third training point $x^3 = [4\ 4]^T$ with a negative label, will that impact the

Answer (b):

hyperplane estimated using points x^1 *and* x^2? Justify. [5 points] 	To get a FULL mark, you need to also mention that x^3 and x^2 are colinear ⇒ redundant samples ⇒ will not affect SVM.
c) Explain how to classify the point $x^{test} = [-1\ 0]^T$ using the estimate model. What is the predicted label of x^{test}? [5 points]	**Answer (c):** $x^{tst} \Rightarrow \text{sign}(w^* x^{tst^T} + b^*)$ $\text{sign}\left(\underbrace{4/13 + 17/13}\right) > 0$

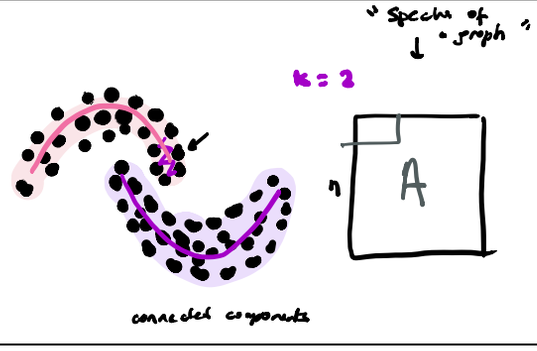
Question 6 [2 points —easy]

Would PCA be a good dimension reduction method to choose for boosting the classification of these two datasets? Justify your choice. [2 points]

	**Choice and justification:**

Question 7 [30 points —medium➔ difficult] (Read Annex)

Read the annex and answer the following questions in the specified space.

1) Illustrate a data distribution where k-means will fail and spectral clustering will work. Each sample will be represented by a dot.	
2) Point out one difference between k-means and spectral clustering	
3) Point out one similarity between k-means and spectral clustering	

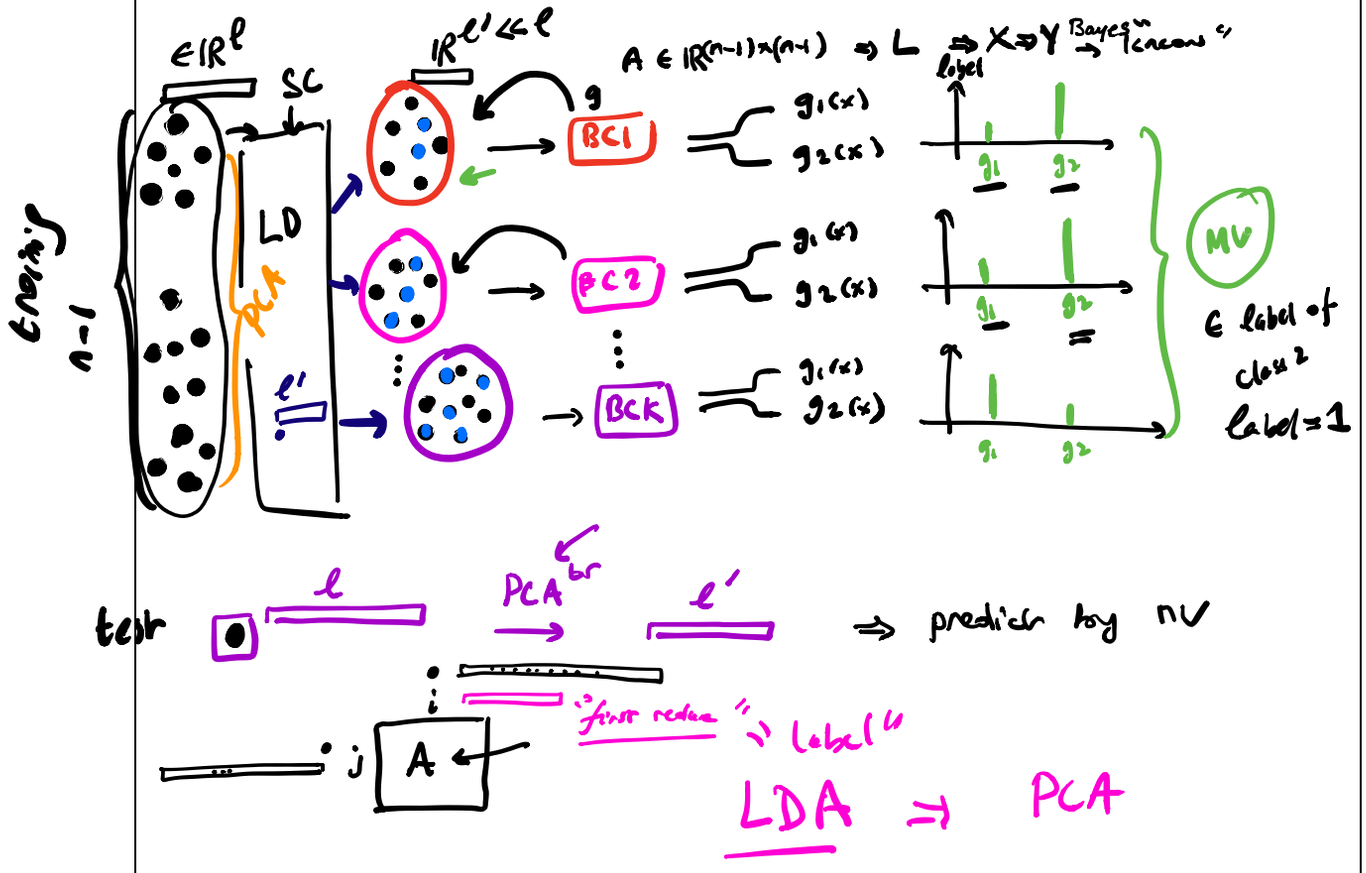
4) What is the **core idea** of spectral clustering? [Hint: think about the original features and how they are transformed.]	

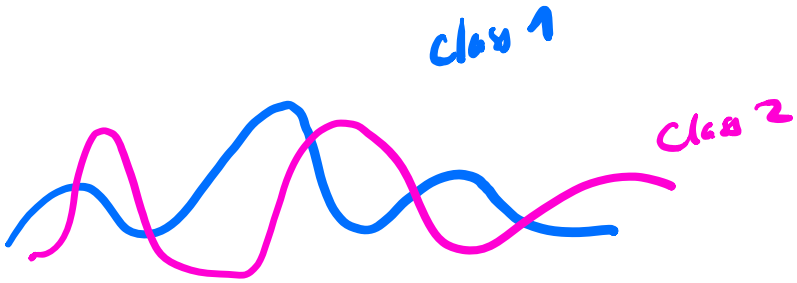
Problem to solve:

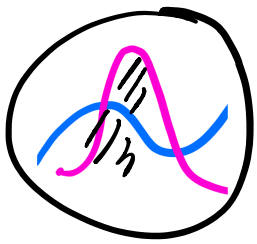
Given a dataset $\mathbf{S} \in \mathbb{R}^{n \times l}$ with n samples, each sample encoded by l features, and has a label $y \in \{0,1\}$. Our ultimate goal is to cluster the training samples into K clusters and train an ensemble of K **Bayes classifiers** for a target classification task. To this aim, we will leverage **dimensionality reduction** to overcome the curse of dimensionality and we use **spectral clustering** to cluster training samples into K clusters to train the ensemble classifier.

- We aim to first reduce the dimensionality of the samples to $l' \ll l$ using PCA.
- Next, using the reduced data, we partition the training set into K clusters and train K logistic classifiers.
- We then train a bayes classifier for each cluster, independently.

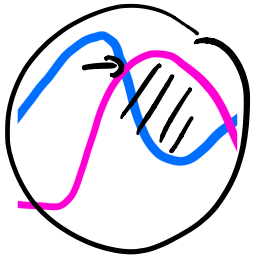
1) Define in table all your mathematical symbols.
2) Using **leave-one-out cross-validation**, write a pseudo-code including all the steps for training and testing samples. All mathematical notations should be specified even for the ~~logistic~~ classifier and PCA. You cannot use "built-in functions".







→ BC



→ BC

On Spectral Clustering: Analysis and an algorithm

Andrew Y. Ng
CS Division
U.C. Berkeley
ang@cs.berkeley.edu

Michael I. Jordan
CS Div. & Dept. of Stat.
U.C. Berkeley
jordan@cs.berkeley.edu

Yair Weiss
School of CS & Engr.
The Hebrew Univ.
yweiss@cs.huji.ac.il

Abstract

Despite many empirical successes of *spectral clustering* methods—algorithms that cluster points using eigenvectors of matrices derived from the data—there are several unresolved issues. First, there are a wide variety of algorithms that use the eigenvectors in slightly different ways. Second, many of these algorithms have no proof that they will actually compute a reasonable clustering. In this paper, we present a simple spectral clustering algorithm that can be implemented using a few lines of Matlab. Using tools from matrix perturbation theory, we analyze the algorithm, and give conditions under which it can be expected to do well. We also show surprisingly good experimental results on a number of challenging clustering problems.

1 Introduction

The task of finding good clusters has been the focus of considerable research in machine learning and pattern recognition. For clustering points in $\mathbb{R}^n$—a main application focus of this paper—one standard approach is based on generative models, in which algorithms such as EM are used to learn a mixture density. These approaches suffer from several drawbacks. First, to use parametric density estimators, harsh simplifying assumptions usually need to be made (e.g., that the density of each cluster is Gaussian). Second, the log likelihood can have many local minima and therefore multiple restarts are required to find a good solution using iterative algorithms. Algorithms such as K-means have similar problems.

A promising alternative that has recently emerged in a number of fields is to use *spectral methods* for clustering. Here, one uses the top eigenvectors of a matrix derived from the distance between points. Such algorithms have been successfully used in many applications including computer vision and VLSI design [5, 1]. But despite their empirical successes, different authors still disagree on exactly which eigenvectors to use and how to derive clusters from them (see [11] for a review). Also, the analysis of these algorithms, which we briefly review below, has tended to focus on simplified algorithms that only use one eigenvector at a time.

One line of analysis makes the link to *spectral graph partitioning*, in which the sec-

ond eigenvector of a graph's Laplacian is used to define a semi-optimal cut. Here, the eigenvector is seen as a solving a relaxation of an NP-hard discrete graph partitioning problem [3], and it can be shown that cuts based on the second eigenvector give a guaranteed approximation to the optimal cut [9, 3]. This analysis can be extended to clustering by building a weighted graph in which nodes correspond to datapoints and edges are related to the distance between the points. Since the majority of analyses in spectral graph partitioning appear to deal with partitioning the graph into exactly two parts, these methods are then typically applied recursively to find k clusters (e.g. [9]). Experimentally it has been observed that using more eigenvectors and directly computing a k way partitioning is better (e.g. [5, 1]).

Here, we build upon the recent work of Weiss [11] and Meila and Shi [6], who analyzed algorithms that use k eigenvectors simultaneously in simple settings. We propose a particular manner to use the k eigenvectors simultaneously, and give conditions under which the algorithm can be expected to do well.

2 Algorithm

Given a set of points $S = \{s_1, \ldots, s_n\}$ in $\mathbb{R}^l$ that we want to cluster into k subsets:

1. Form the affinity matrix $A \in \mathbb{R}^{n \times n}$ defined by $A_{ij} = \exp(-\|s_i - s_j\|^2/2\sigma^2)$ if $i \neq j$, and $A_{ii} = 0$.
2. Define D to be the diagonal matrix whose (i,i)-element is the sum of A's i-th row, and construct the matrix $L = D^{-1/2}AD^{-1/2}$.
3. Find $x_1, x_2, \ldots, x_k$, the k largest eigenvectors of L (chosen to be orthogonal to each other in the case of repeated eigenvalues), and form the matrix $X = [x_1 x_2 \ldots x_k] \in \mathbb{R}^{n \times k}$ by stacking the eigenvectors in columns.
4. Form the matrix Y from X by renormalizing each of X's rows to have unit length (i.e. $Y_{ij} = X_{ij}/(\sum_j X_{ij}^2)^{1/2}$).
5. Treating each row of Y as a point in $\mathbb{R}^k$, cluster them into k clusters via K-means or any other algorithm (that attempts to minimize distortion).
6. Finally, assign the original point s_i to cluster j if and only if row i of the matrix Y was assigned to cluster j.

Here, the scaling parameter σ^2 controls how rapidly the affinity A_{ij} falls off with the distance between s_i and s_j, and we will later describe a method for choosing it automatically. We also note that this is only one of a large family of possible algorithms, and later discuss some related methods (e.g., [6]).

At first sight, this algorithm seems to make little sense. Since we run K-means in step 5, why not just apply K-means directly to the data? Figure 1e shows an example. The natural clusters in $\mathbb{R}^2$ do not correspond to convex regions, and K-means run directly finds the unsatisfactory clustering in Figure 1i. But once we map the points to $\mathbb{R}^k$ (Y's rows), they form tight clusters (Figure 1h) from which our method obtains the good clustering shown in Figure 1e. We note that the clusters in Figure 1h lie at 90° to each other relative to the origin (cf. [8]).

[1] Readers familiar with spectral graph theory [3] may be more familiar with the Laplacian $I - L$. But as replacing L with $I - L$ would complicate our later discussion, and only changes the eigenvalues (from λ_i to $1 - \lambda_i$) and not the eigenvectors, we instead use L.

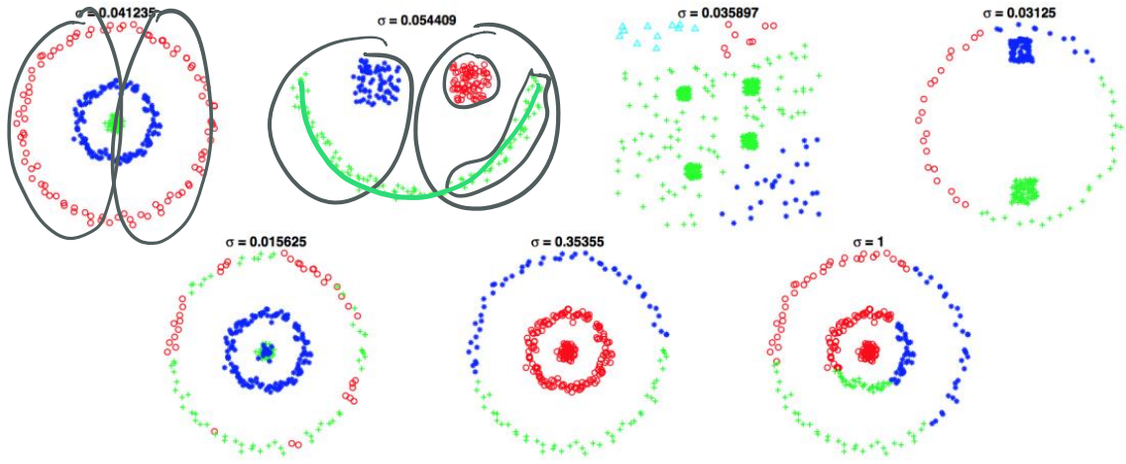


Figure 1: **Spectral clustering without local scaling** (using the NJW algorithm.) *Top row: When the data incorporates multiple scales standard spectral clustering fails. Note, that the optimal σ for each example (displayed on each figure) turned out to be different. Bottom row: Clustering results for the top-left point-set with different values of σ. This highlights the high impact σ has on the clustering quality. In all the examples, the number of groups was set manually. The data points were normalized to occupy the $[-1, 1]^2$ space.*