COS 424: Interacting with Data

Lecturer: Rob Schapire and David Blei
Scribe: Indraneel Mukherjee

Lecture # 8
March 1, 2007

In the previous lecture we saw how Support Vector Machines (SVMs) serve as effective tools in classifying data by obtaining a hyperplane that separates the positive and negative examples by the maximum margin. We also saw how projecting the data into a higher dimensional space and linearly separating the projected data allowed us to separate the examples by polynomial surfaces in case the data was not linearly separable. Further the use of a kernel allowed us to carry out computations in high dimensions without suffering more than a logarithmic (in the degree of the polynomial) overhead in time complexity.

1 Case study of SVM (Rob)

We wrap up our discussion of SVMs by analyzing how well they are able to classify handwritten digits, represented as 16×16 arrays of pixel-intensities. The training set consists of 2000 examples, and the test-set is of size 7300.

According to Rob's report, the linear classifier misclassified 340 training examples, a polynomial kernel of degree 2 achieved a training-error of only 4, erring on what are evidently outliers (fig. 1), while a degree 3 polynomial perfectly classified the training set.

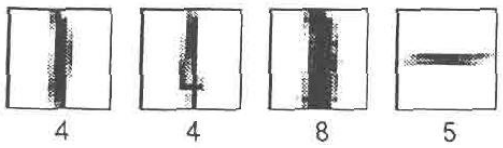


Figure 1: The outliers misclassified by degree 2 kernel

In general, the simplicity vs training-accuracy tradeoff makes the task of finding the optimum degree of the kernel-polynomial difficult. A trial and error approach could consist of splitting our collection of training examples into a test set and a training set, and trying out SVM's with various degree kernels to find the best fit. However, this leads to wastage of data.

For this problem, however, we could find, among the kernels that suffer negligible training error, the one that minimizes the explicit estimate $(R/\delta)^2$ of the VC-dimension(*simplicity*) of linear classifiers achieving margin δ on examples of norm at most R. A plot of this estimate against different degrees is shown in figure 2, and the optimal degree can be read off from the graph. Since the given problem is not a binary-classification task, we have to consider the possibility that the optimum degree could vary with the digits(fig. 3). In that case, on a test example, we run the SVMs for each digit separately and break ties by choosing the prediction that maximizes confidence, i.e. margin.

The table in figure 4 compares the performance of SVMs having degree chosen to minimize $(R/\delta)^2$ with those having other degrees for different digits. In each case, the best performing degrees (enclosed in squares) are the ones which are chosen as above.

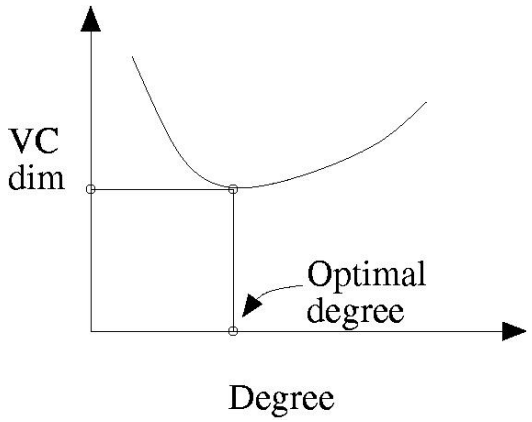


Figure 2:

To sum up, SVM's are highly popular since they are the state-of-the-art for many problems of practical interest. So even though the algorithms for finding SVM's are complicated, they have been well-studied.

2 Clustering (David)

Till now we have been looking at the problem of supervised learning; namely, we are presented with a number of *labelled* training examples which we use to output a hypothesis that guesses the correct label when presented with a new unlabelled test example. The training set is typically small, since labelled data is expensive to generate. In the unsupervised learning setting, there is an abundance of unlabelled training examples, and we strive to gain some understanding of the data. Generally learning problems can be broadly classified as supervised or unsupervised learning, although there are certain problems combining features of both settings.

Understanding unlabelled examples essentially consists of segmenting the data into groups of similar points. For instance, we might want to group customers according to purchase histories, genes according to expression profiles, web-search results according to topics, MySpace users according to interests, etc. . This could be useful, for instance, in automatically organizing the data, discovering hidden structures, or representing high-dimensional data in a low-dimensional space.

2.1 Clustering Setup

We will assume our data consists of a set $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of points in $\mathbb{R}^p$, each point $\mathbf{x}_n$ being represented as $\mathbf{x}_n = \langle x_{n,1}, \ldots, x_{n,p} \rangle$. We will also need a notion of *distance* $d(\mathbf{x}_n, \mathbf{x}_m)$ between data-points. We want to segment the data into k groups $\{z_1, \ldots, z_N\}$ where $z_i \in \{1, \ldots, K\}$, for some choice of k (how k is chosen will be discussed later).

Before discussing clustering algorithms in the general setup, we look at a motivating example. Consider the problem of clustering the collection of 500 2-dimensional points shown in figure 5.

Squared Euclidean distance (squared to simplify computations), defined as $d(\mathbf{x}_n, \mathbf{x}_m) = \sum_{i=1}^{p} (x_{n,i} - x_{m,i})^2 = \|x_n - x_m\|^2$, seems a reasonable candidate for the distance function,

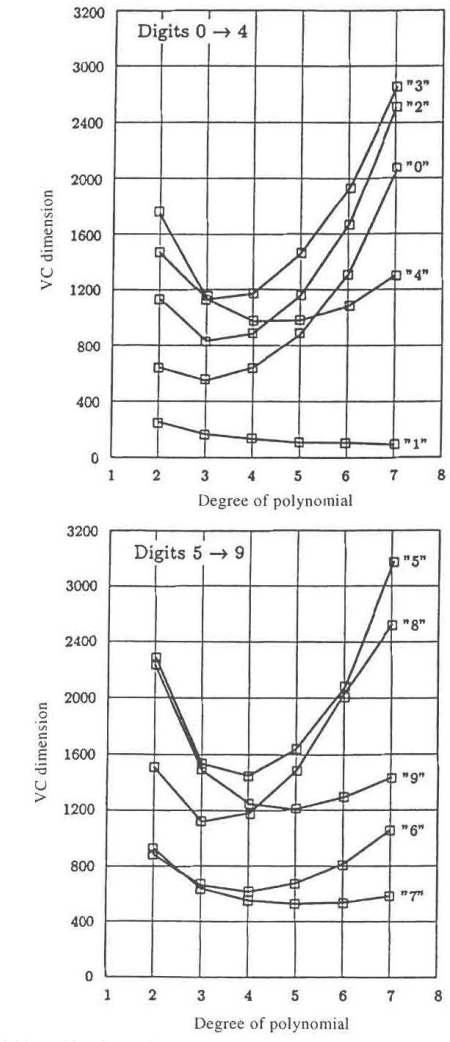


Figure 3:

either by visual inspection, or from the fact that the samples lie in the Euclidean plane. Our goal is to group the data into k groups, for an appropriate choice of k. Automatically choosing k is complicated. To keep things simple, we will assume a value of 4. While there are different ways in which one may use the data and distances to cluster, we begin with the simplest algorithm: k-means.

2.2 k-means

The basic idea in k-means is to describe each cluster by its *mean value*, i.e. the point obtained by coordinate-wise summing up the points within a cluster and taking average. Pseudocode for the algorithm follows:

1. Initialization

    - Data are $\mathbf{x}_{1:N}$

Table 12.8. Experiments on choosing the best degree of polynomial[a]

	Chosen Classifier			Number of Test Errors						
Digit	deg.	dim.	$h_{\text{est.}}$	1	2	3	4	5	6	7
0	3	$\sim 10^6$	530	36	14	**11**	11	11	12	17
1	7	$\sim 10^{16}$	101	17	15	14	11	10	10	**10**
2	3	$\sim 10^6$	842	53	32	**28**	26	28	27	32
3	3	$\sim 10^6$	1157	57	25	**22**	22	22	22	23
4	4	$\sim 10^9$	962	50	32	32	**30**	30	29	33
5	3	$\sim 10^6$	1090	37	20	**22**	24	24	26	28
6	4	$\sim 10^9$	626	23	12	12	**15**	17	17	19
7	5	$\sim 10^{12}$	530	25	15	12	10	**11**	13	14
8	4	$\sim 10^9$	1445	71	33	28	**24**	28	32	34
9	5	$\sim 10^{12}$	1226	51	18	15	11	**11**	12	15

[a]The boxes indicate the chosen order of a polynomial.

Figure 4:

- Choose initial cluster means $\mathbf{m}_{1:k}$ (same dimension as data).

2. Repeat

   (a) Assign each data point to its closest mean

   $$z_n = \arg \min_{i \in \{1,\ldots,k\}} d(\mathbf{x}_n, \mathbf{m}_i)$$

   (b) Compute each cluster mean to be the coordinate-wise average over data points assigned to that cluster,

   $$\mathbf{m}_k = \frac{1}{N_k} \sum_{\{n\,:\,z_n = k\}} \mathbf{x}_n$$

3. Until assignments $\mathbf{z}_{1:N}$ do not change

The positions of the means in different iterations of a sample run of the algorithm on the above dataset is shown below.

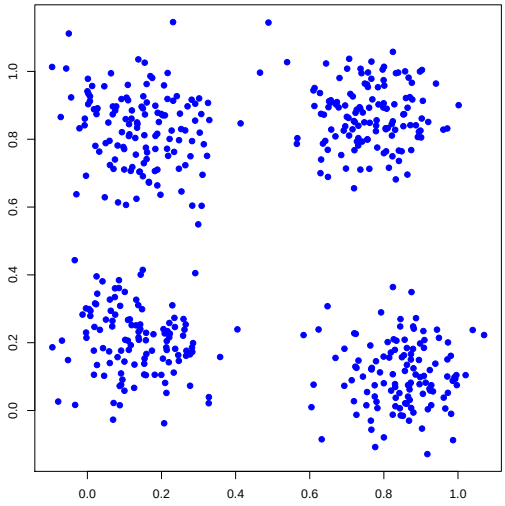
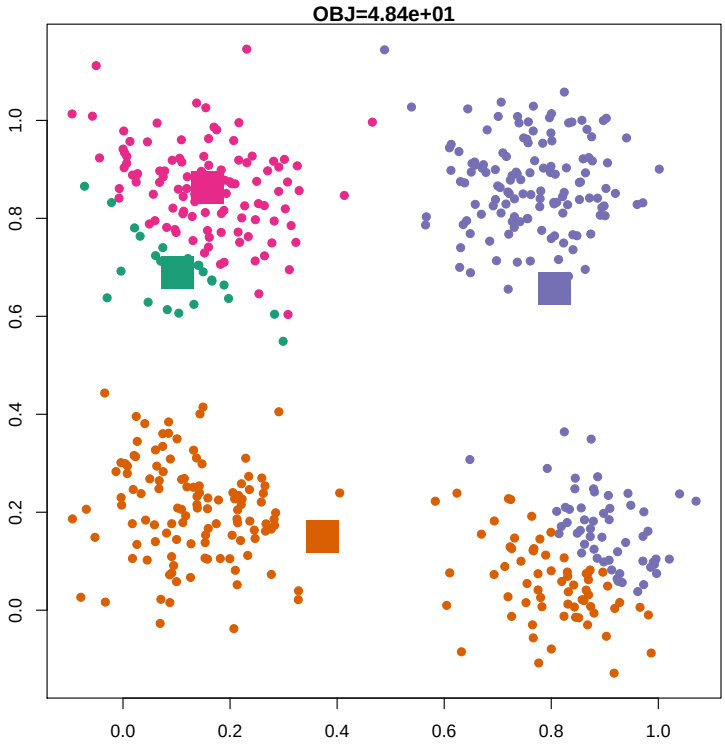
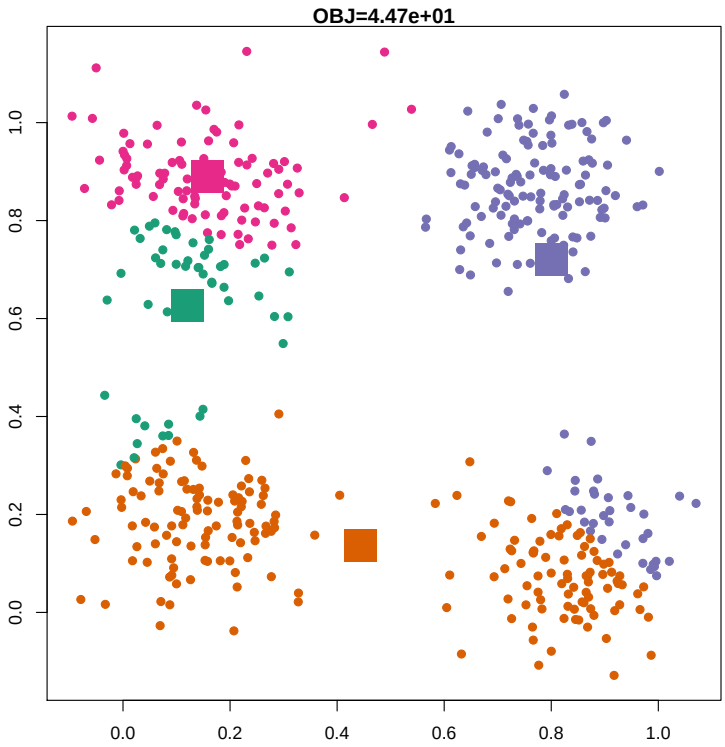
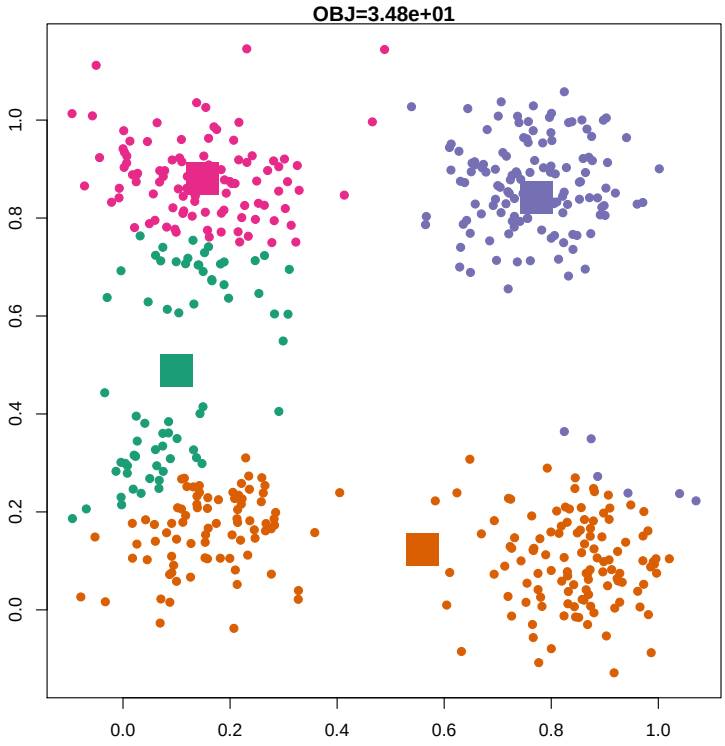
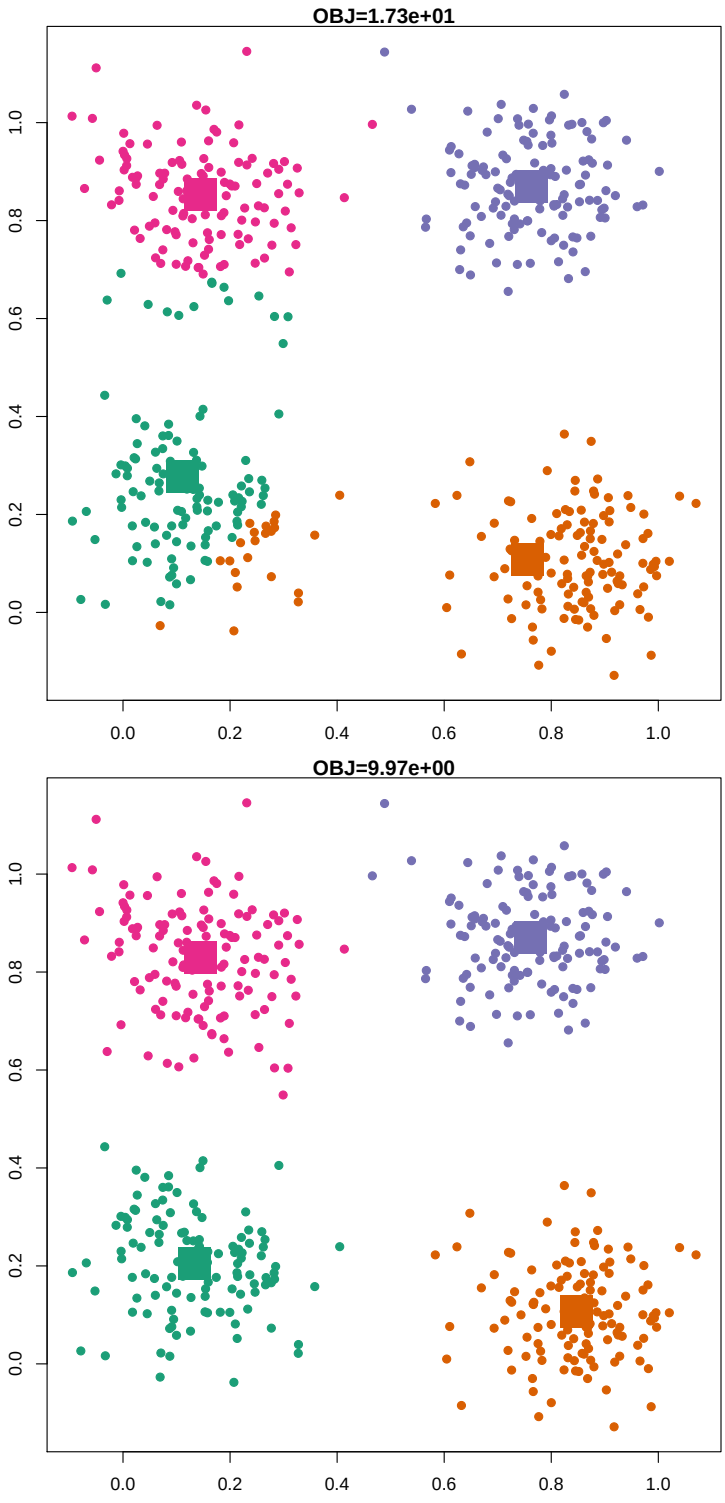


Figure 5:

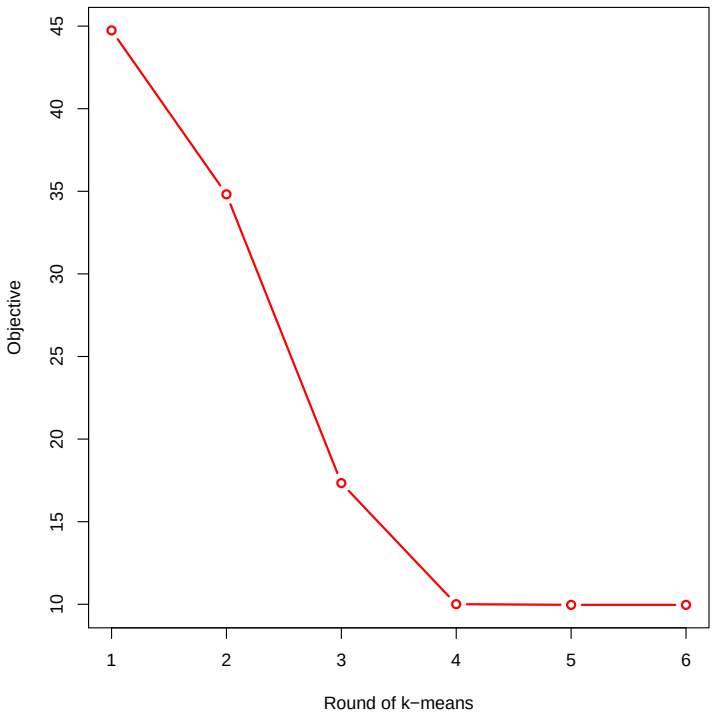



OBJ=4.47e+01
0.0
0.2
0.4
0.6
0.8
1.0


OBJ=3.48e+01
0.0
0.2
0.4
0.6
0.8
1.0



Intuitively, it is clear that we produce increasingly better clusterings as the algorithm proceeds. This can be captured formally by measuring the quality of a particular clustering by the k-means objective function $F(z_{1:N}, \mathbf{m}_{1:k}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{m}_{z_n}\|^2$. The changing values of the objective function in the sample run is written at the top of the respective slides, and can be seen to be steadily decreasing. A plot of the same can be seen below.



2.3 k-means is Coordinate Descent

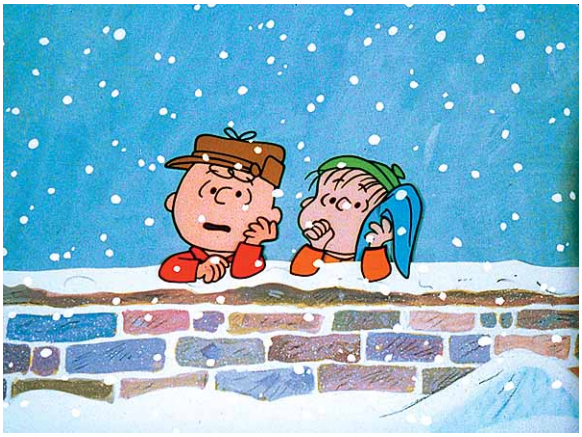
We could think of k-means as trying to minimize the objective function $F(z_{1:N}, \mathbf{m}_{1:k}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{m}_{z_n}\|^2$. It is easy to see that step $2(a)$ of the algorithm minimizes F if the centroids of the clusters are fixed to be $\mathbf{m}_{1:k}$: each point contributes (independently of other points) its distance to its assigned mean to the sum, and choosing the closest mean minimizes this contribution.

On the other hand, if the assignments $z_{1:N}$ are fixed, minimizing F is equivalent to minimizing $\sum_{i:z_i=t} \|x_i - \mathbf{m}_t\|^2$ for each cluster $t \in [k]$ separately. Note this expression is exactly the negative log-likelihood of the points labelled t by $\mathbf{z}$, assuming the points are distributed according to a Gaussian distribution centered at $\mathbf{m}_t$. This is an important connection that motivates the choice of the objective function and which will be explored later. From the first homework assignment, we know that the choice of center maximizing the likelihood is the empirical mean of the samples, which is chosen in step $2(b)$ of the algorithm.

Thus k-means strives to minimize its objective function by fixing different subsets of its coordinates, and minimizing the resulting simplified objective function. Algorithms adhering to this general paradigm form a family referred to as coordinate descent algorithms. These are used when the objective function is non-convex, and the solution they converge to, which depends on the initial choice of the cluster-means, are local minima. Multiple restarts with different initial choices are often necessary to find the global minimum.

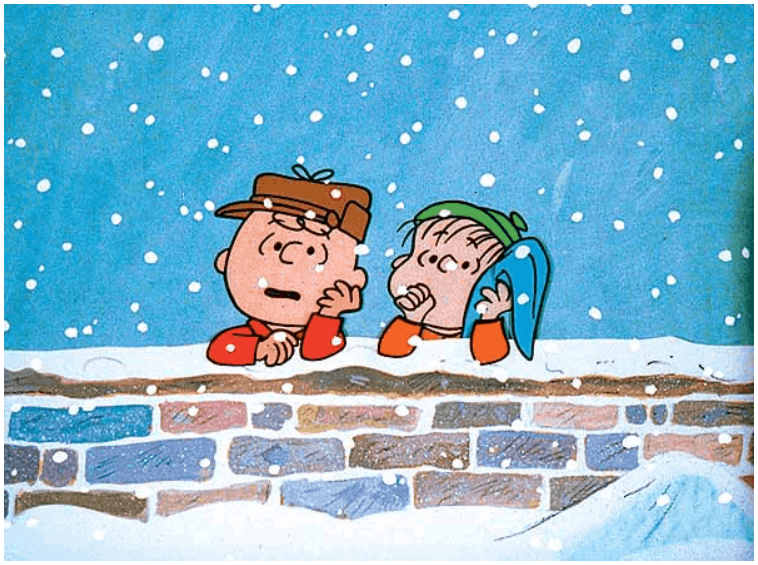
2.4 Compressing Images: an application

An image consists of a grid of pixels $\mathbf{x}_n$. A pixel consists of a triple of real numbers, corresponding to proportions of red, green and blue in the color that it represents.

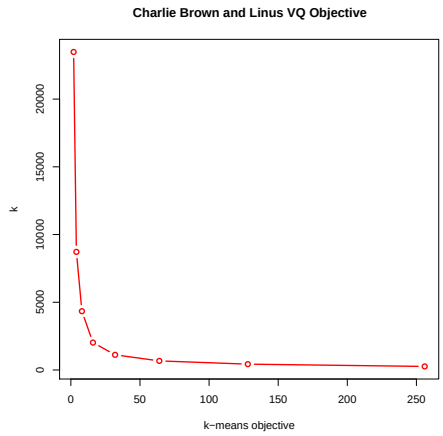


For example, the above 1024×1024 image is a collection of 1048576 triples, which requires 3M of storage. A possible means of compression is vector quantization, where we decide to approximate the colors in an image by a small set of colors, called the *codebook*, and replace each pixel $\mathbf{x}_n$ in the compressed image by the index z_n of the color in the codebook "closest" to it. For instance, if we choose a codebook of size $k = 100$, then we'd need only 7 bits per pixel for the compressed image, along with the 100×3 bits for the codebook, for a total storage of only 897K.

Given a notion of closeness, the problem reduces to finding the codebook of some size k. Thinking of the $\mathbf{x}_n$ as 3-dim points, and letting squared euclidean norm be a measure of closeness of colors, the centers $\mathbf{m}$ found by the k-means algorithm applied to $\mathbf{x}_n$ could serve as the codebook. In this case, the choice of the size k represents a tradeoff between the extent of compression obtained and the quality of the compressed image. Applying k-means to achieve compression by means of vector quantization to the above image, for $k = 2, 8, 32, 128$, is shown below (in that order).



Notice how the quality of the compression improves rapidly initially, but is hardly perceptible later on, although the value of k, and hence the compressed size, is increasing exponentially. A plot showing how the distortion, as measured by the k-means objective function, changes with k confirms our observations.



2.5 k-medoids

In many practical settings, when the data is multivariate discrete, like customer purchase histories, or is only positive, such as time spent on a web-page, Euclidean norm is no longer an appropriate measure of the distance. Further, as in the case of purchase histories, the mean of a cluster may be meaningless. k-medoids is an algorithm which works on an arbitrary metric d on the data points, and associates each cluster with its "most typical" member, namely the one that minimizes the sum of distances from all the other points in that cluster. Pseudocode for k-medoids is given below:

1. Initialization
    - Data are $\mathbf{x}_{1:N}$
    - Choose initial cluster identities $\mathbf{m}_{1:k}$

2. Repeat

   (a) Assign each data point to its closest center

   $$z_n = \arg\min_{i \in \{1,\ldots,k\}} d(\mathbf{x}_n, \mathbf{m}_i)$$

   (b) For each cluster, find the data point in that cluster that is closest to the other points in that cluster

   $$i_k = \arg\min_{\{n\,:\,z_n=k\}} \sum_{\{m\,:\,z_m=k\}} d(\mathbf{x}_n, \mathbf{x}_m)$$

   (c) Set each cluster center equal to their closest data points

   $$m_k = \mathbf{x}_{i_k}$$

3. Until assignments $\mathbf{z}_{1:N}$ do not change

2.6 Choosing k

Choosing k is a nagging problem in cluster analysis. Sometimes the problem determines k, e.g. a certain required compression in the vector quantization of an image, clustering customers for an available number of salespeople in a business, etc. . We try to choose a "natural" value for the number of clusters, but in general this notion is not well-defined.