Introduction to Machine Learning **Fall Semester, 2014**

Recitation 1: October 26th

Lecturer: Regev Schweiger *Scribe: Yishay Mansour*

1.1 K-Means

In the k-means algorithm, in each iteration we have two actions:

Assign: Sets each point to its closest center:

$$C_j^t = \arg\min_i \|\mathbf{x}_j - \mu_i^{t-1}\|^2, S_i^t = \{j | C_j^t = i\}$$

Update: Minimizes F by re-computing the centers:

$$\mu_i^t = (1/|S_i^t|) \sum_{j \in S_i^t} \mathbf{x}_j$$

The algorithm has an objective function F, where

$$F((\mu_1, \ldots, \mu_k), (S_1, \ldots, S_k)) = \sum_{i=1}^{k} \sum_{j \in S_i} \|\mathbf{x}_j - \mu_i\|_2^2$$

The value of F (as a function of the centers and cluster assignments) decreases with each iteration, until it stops (Figure 1.1). Theoretically we are not guaranteed convergence because we might loop between configurations of identical F value .

More importantly, we might have a bad solution (see the example of 3-means in Figure 1.2).

How can we overcome the convergence problem? We can select a few random starting points and select the best (the one that has the lowest observed F). The dependency on the number of clusters is illustrated in Figure 1.3.

Another issue is *overfitting* - while increasing k decreases F on the data we see, future data may not behave well according to the k clusters we see.

An example for using the k-means: We have a picture with 512×512 pixels, each 24 bits (i.e., each has 8 bits for each color). We would like to do a compression to 4 bits per pixel. We can view the input as 2^{18} 3-dimensional vectors (the colors of each pixel). We run a 16-means algorithms on this input. When the algorithm ends we have 16 clusters, and each pixel belongs to a cluster. Now we give each pixel the name of the cluster, and for each cluster we keep its center. The total size in only $4 \cdot 2^{18} + 16 \cdot 24$ versus $24 \cdot 2^{18}$ before.

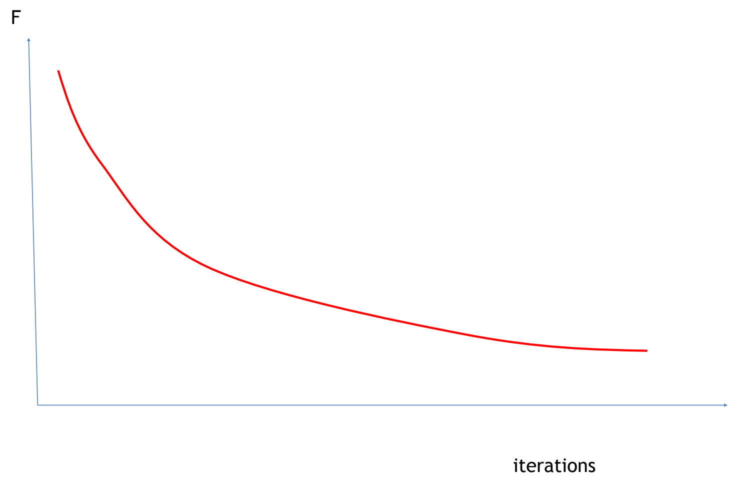

Figure 1.1: Objective function F as a function of the number of iterations

1.2 k-Nearest Neighbors

The input we have are points and their classification, i.e., $(\mathbf{x}, y)$. The goal is to compute a function $f(\mathbf{x})$ and hopefully $f(\mathbf{x}) \approx y$. For binary classification (as in the lecture) we can set $f(\mathbf{x}) = majority(\mathbf{x}_{[1]}, \ldots, \mathbf{x}_{[k]})$. For categorical classification we can use plurality (instead of majority), i.e., $f(x) = \arg\max_c \{\mathbf{x}_{[j]} | y_{[j]} = c\}$. For continuous values we can set $f(x) = (1/k) \sum_{j=1}^{k} y_{[j]}$.

In order to make the problem sound, we need to select a loss function. For binary prediction, an intuitive loss function is the $0-1$-loss, where $L(a,b) = 0$ iff $a = b$ and otherwise $L(a,b) = 1$. We can also have a quadratic loss $L^2(a,b) = (a-b)^2$. We set $L_f(\mathbf{x}, y) = L(f(\mathbf{x}), y)$, i.e., $a = f(\mathbf{x})$ and $b = y$. The rest of the discussion refers to the continuous case.

To illustrate the influence of k, assume that the samples $(\mathbf{x}, y)$ are drawn from a specific joint distribution of the random variables $(\mathbf{X}, Y)$. Our goal is to select the function f that minimizes

$$\mathrm{E}_{\mathbf{X},Y}\left[L_f(\mathbf{x}, y)\right] = \mathrm{E}_{\mathbf{X}}\left[\mathrm{E}_{Y|\mathbf{X}}\left[(f(\mathbf{x}) - y)^2\right]\right]$$

The optimal function is $\hat{f}$ such that $\hat{f}(\mathbf{x}) = E[y|\mathbf{x}]$ (exercise). However, we can not use this in practice, since we do not know the distribution of $(\mathbf{x}, y)$. We can view the k-NN as an approximation of $\hat{f}$.

The difference from $\hat{f}$ is in two places: (1) We use points near x rather than $\mathbf{x}$ itself; (2) We use the empirical average based only on a few points (k) rather than the underlying

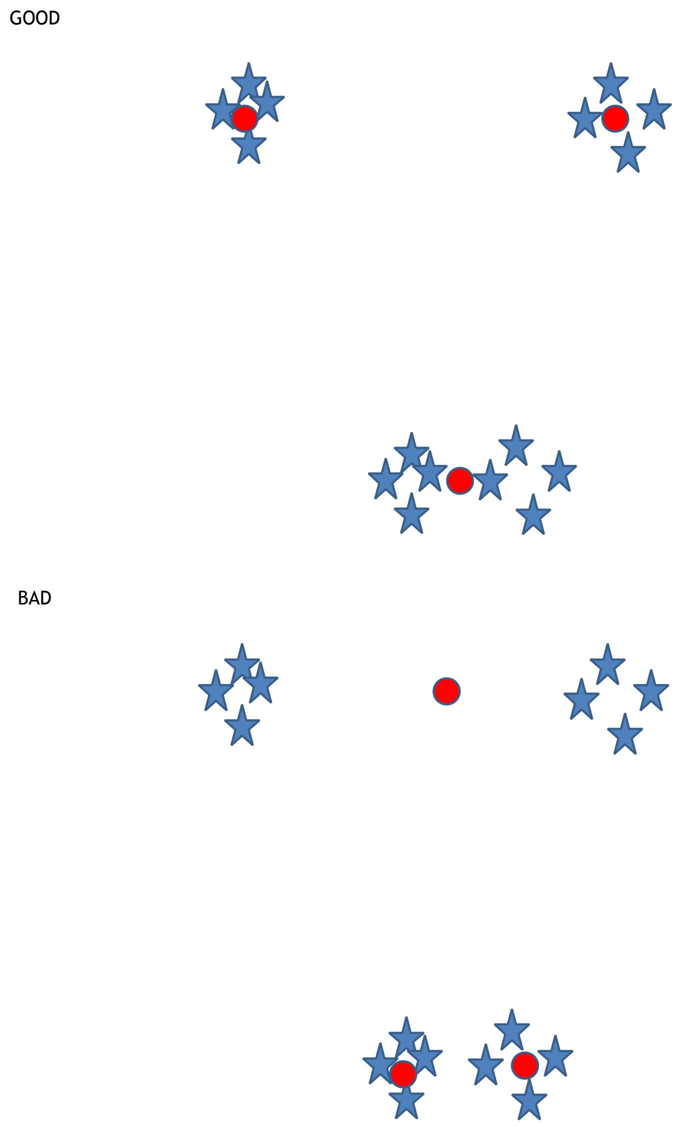

Figure 1.2: Bad and good solutions for 3 clusters

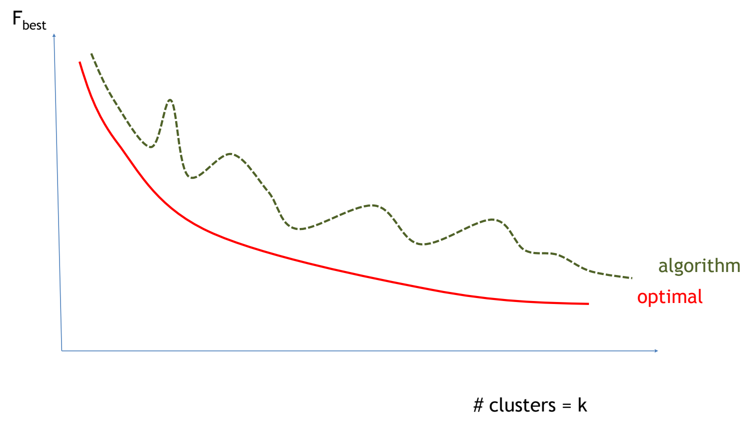

Figure 1.3: Dependency of F on the number of clusters, optimal vs. algorithm

average of the conditional distribution $(y|\mathbf{x})$.

For a very large k (e.g. $k = n$), the error is large, since we are grouping together very different examples, hurting the approximation (1) above.

For a small k (e.g. $k = 1$), we approximate the conditional average based on a very small sample set, hurting approximation (2) above and thereby fitting also noise.