Crime Statistics and ML

A “standard” ML perspective

Can we predict crime?
Can we prevent crime?
And if we can do either, what are the right measures of effectiveness?

A (slightly) more nuanced set of questions

What if our predictions are only effective for some types of crime?

For some types of neighborhoods?

What features are *acceptable* to use in predicting crime?

How are these features/labels gathered?
What if they are gathered in an uneven manner?

And what will be done with these predictions?

Would most of our concerns be mitigated by:

Removing demographic information from a dataset?

What is the concern here?

In the US, policing, arresting, charging, and convicting for certain crimes has been applied to different populations at *very* unequal rates.

E.g., illicit drug use is charged at much higher rates for minority persons

Moreover, crimes charged at higher rates for certain demographics have been deigned more dangerous than similar ones charged in other demographics

E.g. crack cocaine having higher legal penalties than powder cocaine

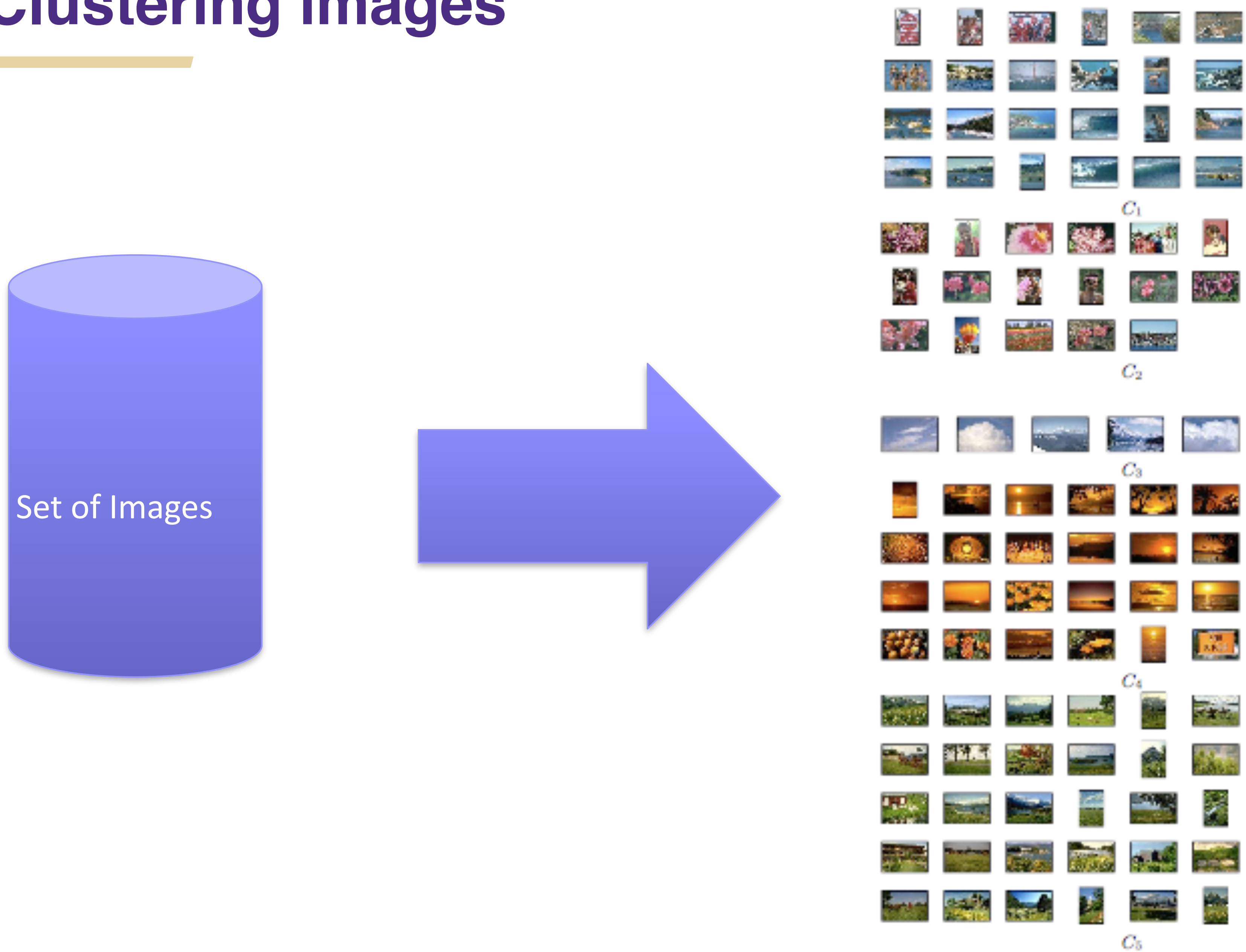
The racial and class makeup of neighborhoods today are the result of decades and centuries of design by lawmakers.

Clustering K-means

UNIVERSITY of WASHINGTON

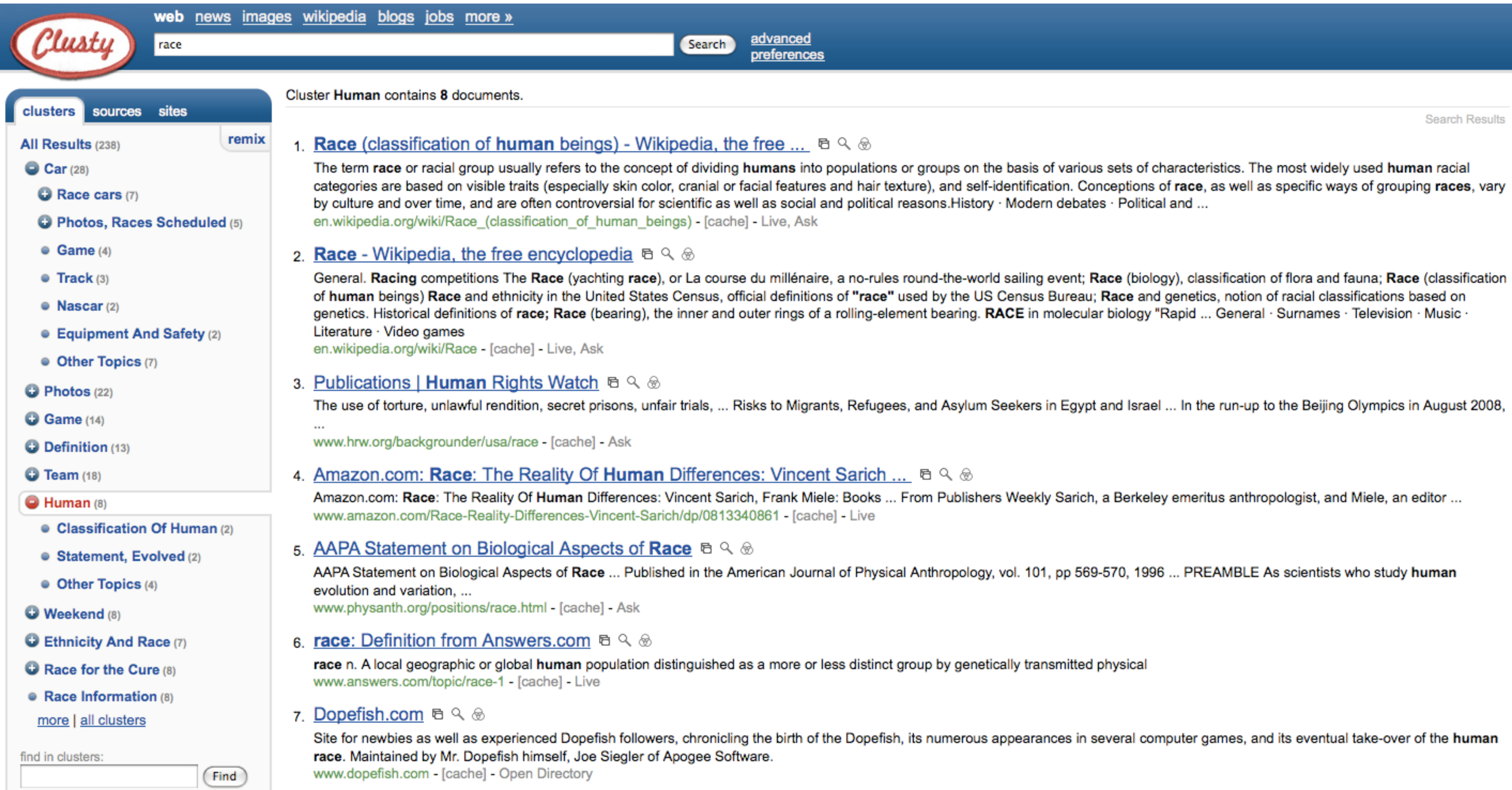
©Kevin Jamieson 2018

Clustering images

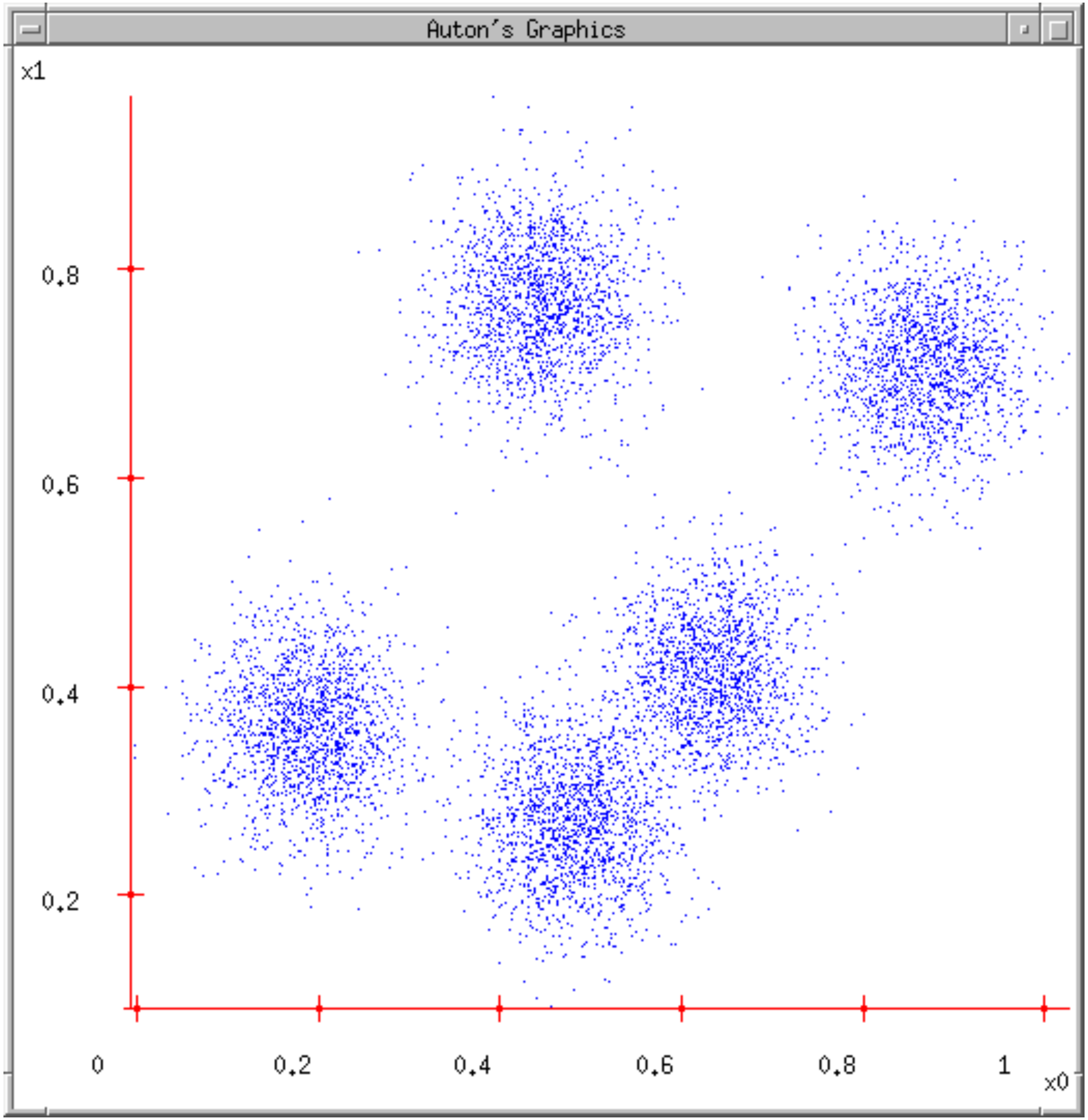


[Goldberger et al.]

Clustering web search results

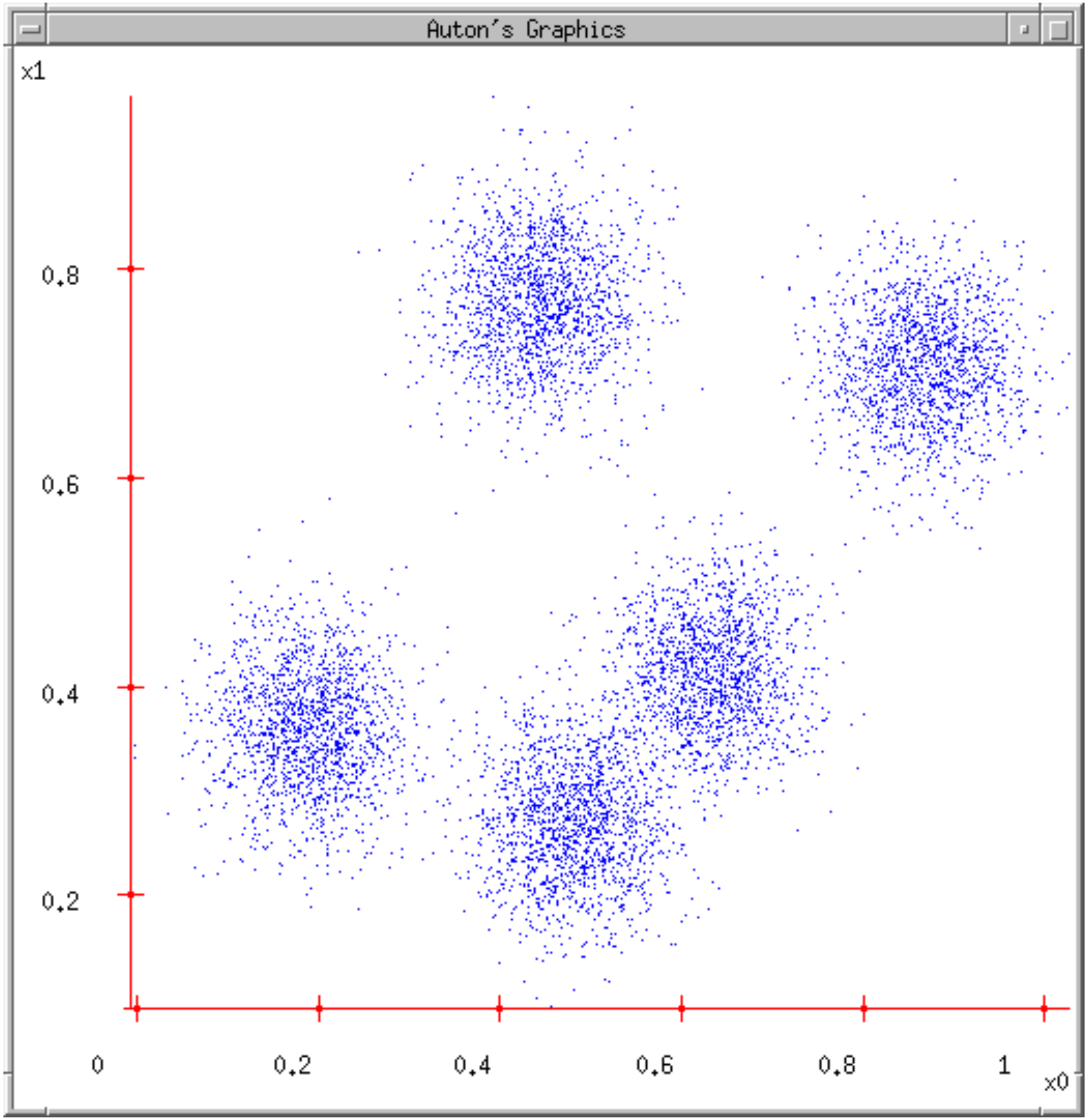


Some Data



Clustering

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Pick clusters to minimize some objective fn.



Clustering

1. Fix a # of clusters *(e.g. k=5)*
2. Choose/Assign each point x_j to $C(j) \in \{1,\ldots,k\}$
    1. Sometimes, pick centers $\boldsymbol{\mu}_1, \ldots \boldsymbol{\mu}_k$

    To minimize

$$F(\mu, C)$$

K-means refers to optimizing this objective:

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

K-means refers to optimizing this objective:

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

How to minimize this quantity?

K-means refers to optimizing this objective:

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

How to minimize this quantity?

NP-Hard to minimize exactly. :(

K-means refers to optimizing this objective:

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

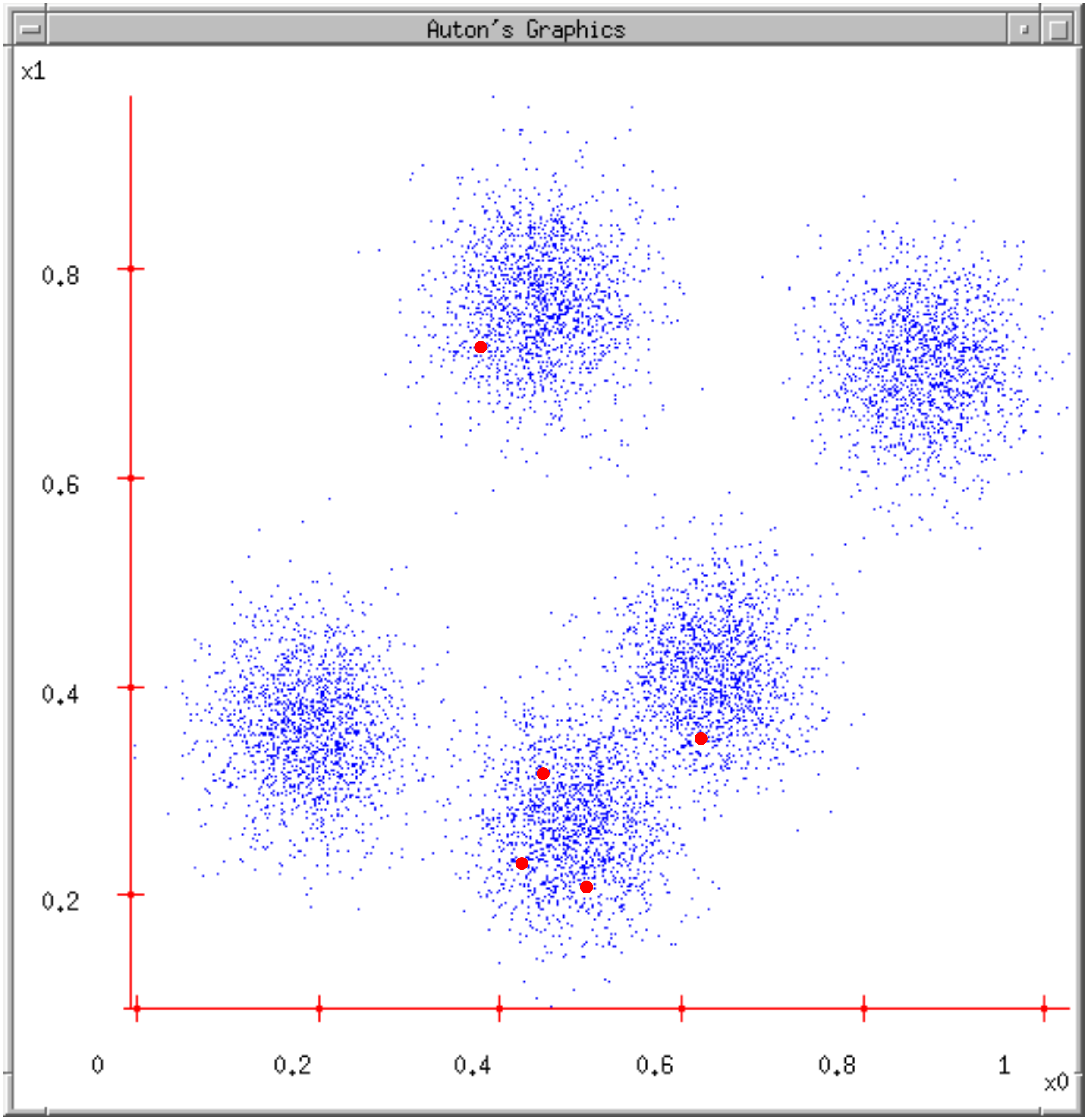
How to minimize this quantity?

NP-Hard to minimize exactly. :(

But, several natural algorithms work well in practice!

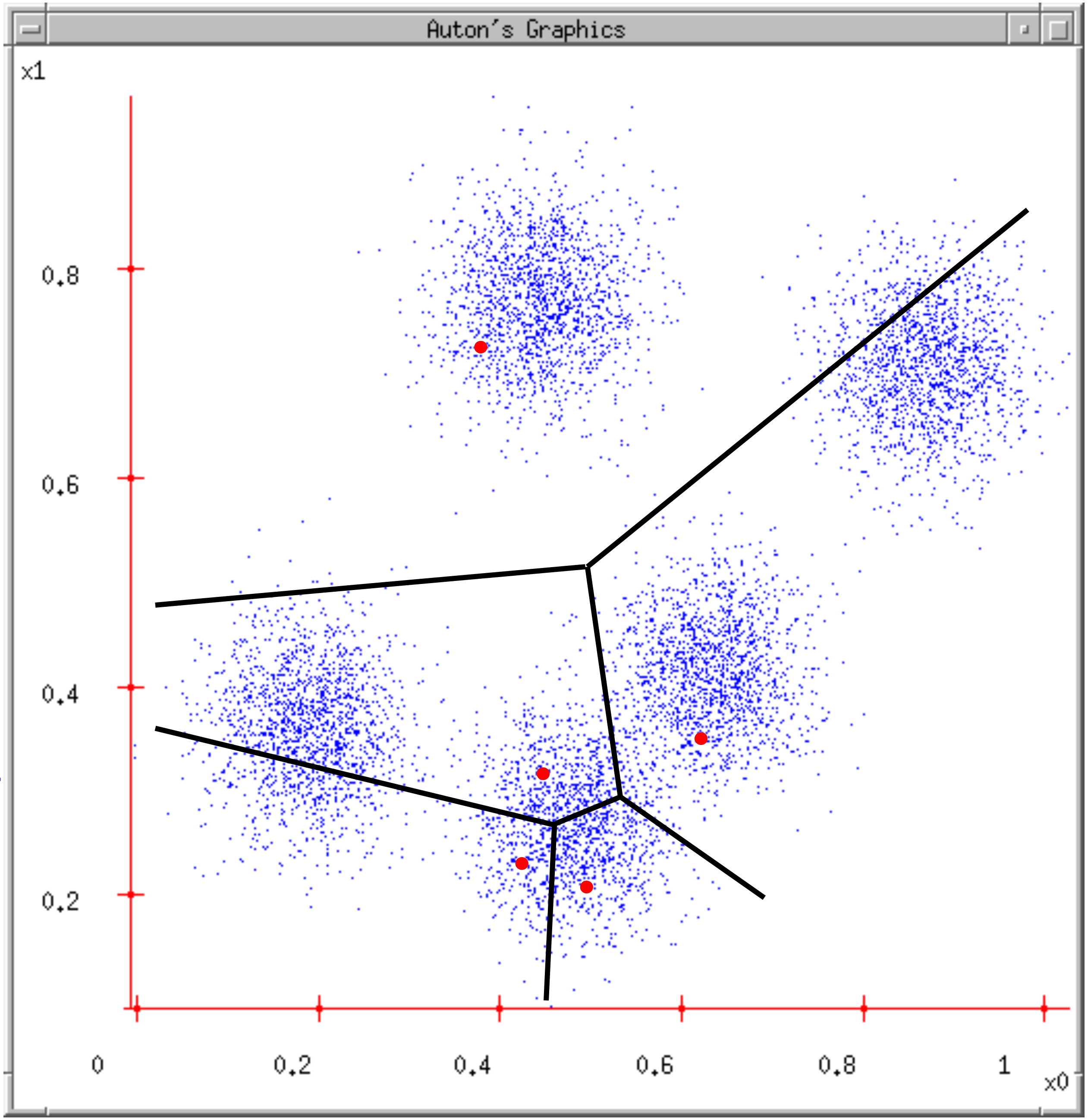
Lloyd's algorithm

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



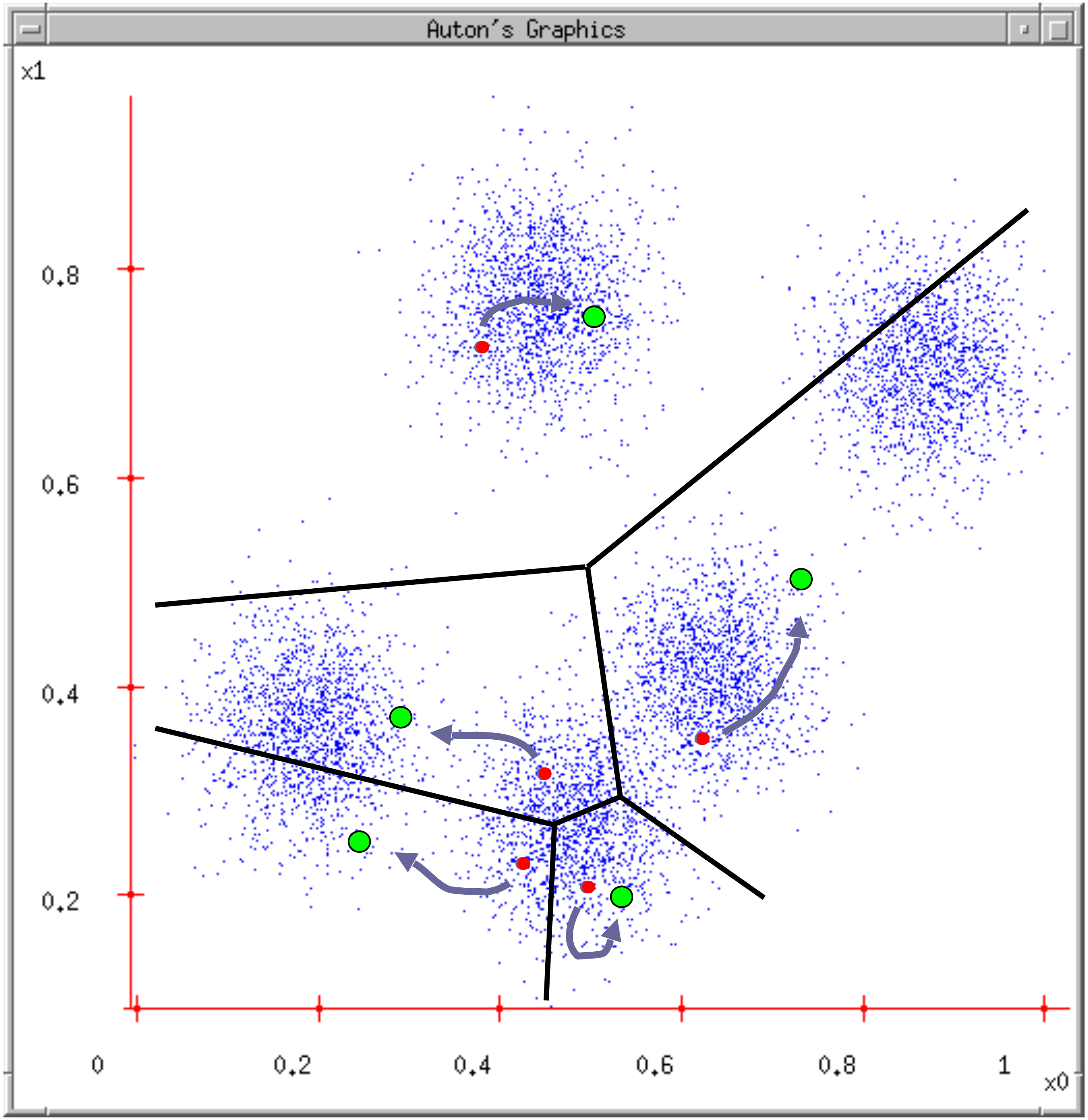
Lloyd's algorithm

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Each center "owns" a set of datapoints)



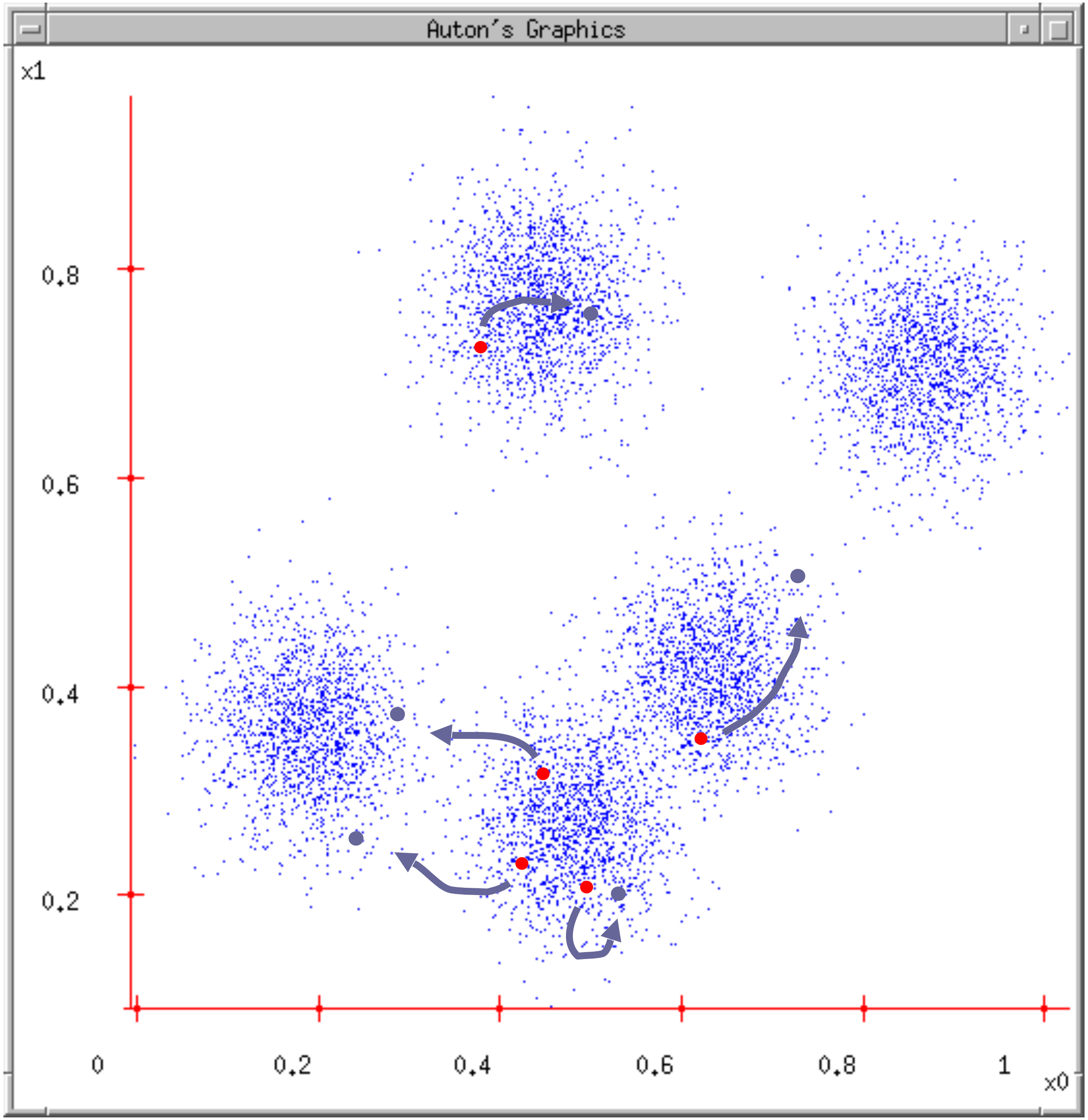
Lloyd’s algorithm

1. Ask user how many clusters they’d like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it’s closest to.
4. Each Center finds the centroid of the points it owns



Lloyd's algorithm

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



Lloyd's algorithm

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!

Lloyd's algorithm

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!

$$C^{(t)}(j) \leftarrow \arg\min_i ||\mu_i - x_j||^2$$

$$\mu_i^{(t+1)} \leftarrow \arg\min_\mu \sum_{j:C(j)=i} ||\mu - x_j||^2$$

Does Lloyd's algorithm converge??? Part 1

$$\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

First, fix **μ**

minimize w.r.t C

Does Lloyd's algorithm converge??? Part 2

$$\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

First, fix **μ**

minimize w.r.t C

Then, fix C

minimize w.r.t **μ**

$F(\mu, C)$ decreases each step $\Rightarrow$ the algorithm doesn't cycle

Only $\binom{n}{k} \approx n^k$ configuarations $\Rightarrow$ converges in finite # iterations

A cool application of k-means clustering: compression

Vector Quantization, Fisher Vectors

Vector Quantization (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
3. Represent each patch as a code word.

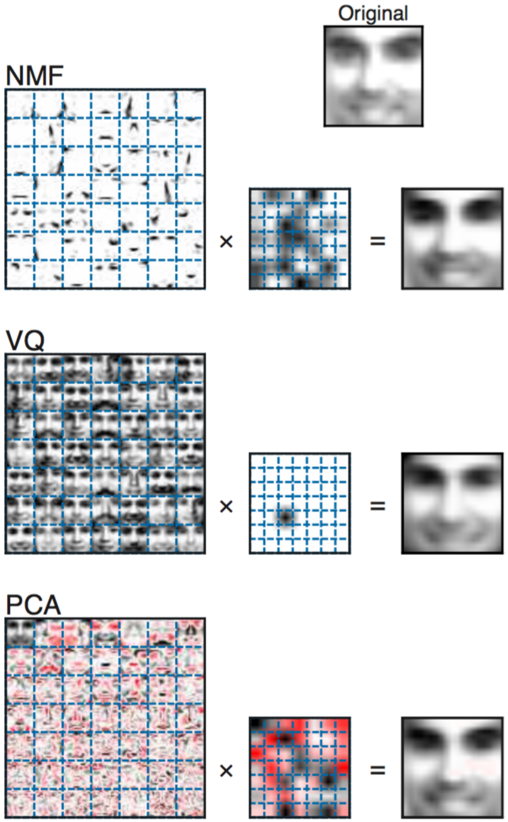
FIGURE 14.9. *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a* 1024×1024 *grayscale image at 8 bits per pixel. The center image is the result of* 2×2 *block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*

Vector Quantization, Fisher Vectors

Vector Quantization (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
   (k = # of codewords, center is code!)
3. Represent each patch as a code word.

FIGURE 14.9. *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a* 1024×1024 *grayscale image at 8 bits per pixel. The center image is the result of* 2×2 *block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*

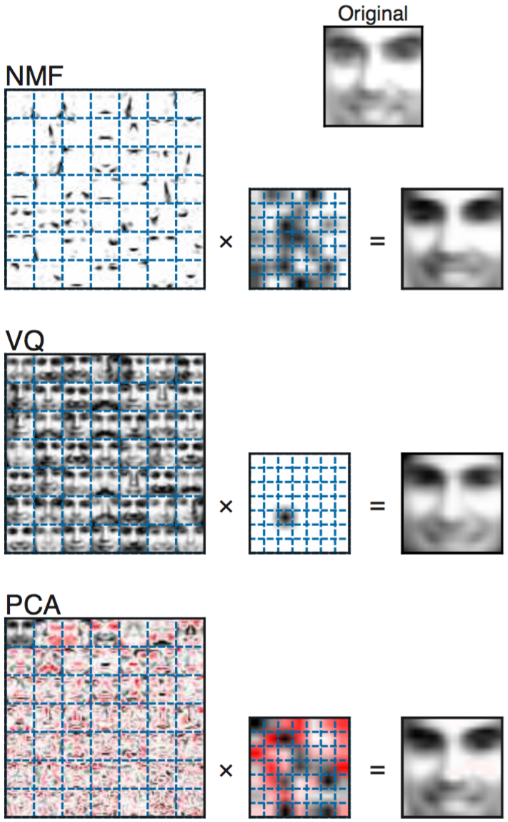


Vector Quantization, Fisher Vectors

Vector Quantization (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
   (k = # of codewords, center is code!)
3. Represent each patch as a code word.

FIGURE 14.9. *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024×1024 grayscale image at 8 bits per pixel. The center image is the result of 2 × 2 block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*



Coates, Ng, *Learning Feature Representations with K-means*, 2012

When to use K-means, or something else?

What sort of groupings are desired?

- Nonoverlapping, similar diameter clusters: k-means may work well
- Otherwise, might want another objective function (spectral, k-median, k-mode, …)

One bad case for k-means

K-means summary

A clustering objective

minimize average L2 distance to centers of clusters

Lloyd's algorithm: a greedy heuristic for minimizing it

Will converge in finite time, may not find global minimum

Good for finding similar width, nonoverlapping clusters

Sensitive to initial center selection, and random may not be the best a priori

See k-means++, *The Advantages of Careful Seeding,* Arthur and Vassilvitskii

Principal Component Analysis

UNIVERSITY of WASHINGTON

Linear projections

Given $x_1, \ldots, x_n \in \mathbb{R}^d$, for $q \ll d$ find a compressed representation with $\lambda_1, \ldots, \lambda_n \in \mathbb{R}^q$ such that $x_i \approx \mu + \mathbf{V}_q \lambda_i$ and $\mathbf{V}_q^T \mathbf{V}_q = I$

$$\min_{\mu, \mathbf{V}_q, \{\lambda_i\}_i} \sum_{i=1}^{n} \|x_i - \mu - \mathbf{V}_q \lambda_i\|_2^2$$

PCA

Data dependent dimensionality reduction

- Useful for
  - Visualization
  - Interpretation
  - Compression
  - Understanding "intrinsic dimension"

	kale	taco bell	sashimi	pop tarts
Alice	10	1	2	7
Bob	7	2	1	10
Carolyn	2	9	7	3
Dave	3	6	10	2

Figure credit:
- Karlin
- Roughgarden + Va
- Benedetto
- Novembre et al
- Alex Williams
- Sandipan Dey
- Victor Lavenko

PCA

Claim:

Each row can be expressed approximately as

$$x_i \approx \bar{x} + a_{i1}\vec{v_1} + a_{i2}\vec{v_2}$$

$$\vec{v_1} = [3 \quad -3 \quad -3 \quad 3]$$

$$\vec{v_2} = [1 \quad -1 \quad 1 \quad -1]$$

	kale	taco bell	sashimi	pop tarts
Alice	10	1	2	7
Bob	7	2	1	10
Carolyn	2	9	7	3
Dave	3	6	10	2

$$\bar{x} = [5.5 \quad 4.5 \quad 5 \quad 5.5]$$

Figure credit:
Karlin
Roughgarden + Va
Benedetto
Novembre et al
Alex Williams
Sandipan Dey
Victor Lavenko

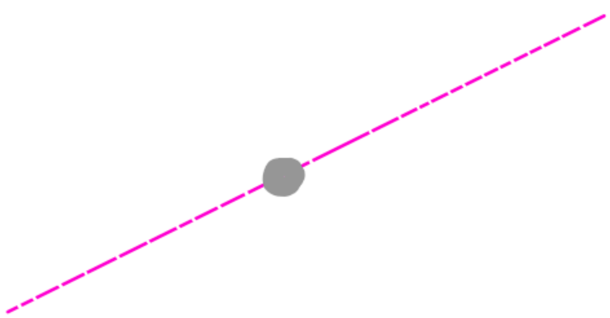
PCA in one dimension

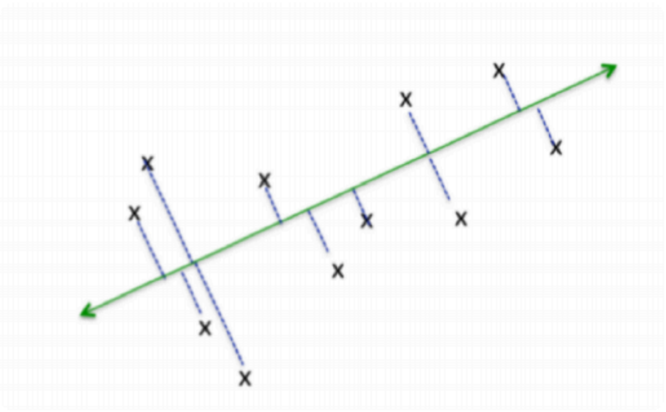
Goal: find a k < d-dimensional representation of X

For k = 1:

Choose $\vec{v} \in \mathbb{R}^d, ||v|| = 1$
to minimize

$$\frac{1}{n}\sum_{i=1}^{n} dist(x_i, \text{line defined by } \vec{v})$$





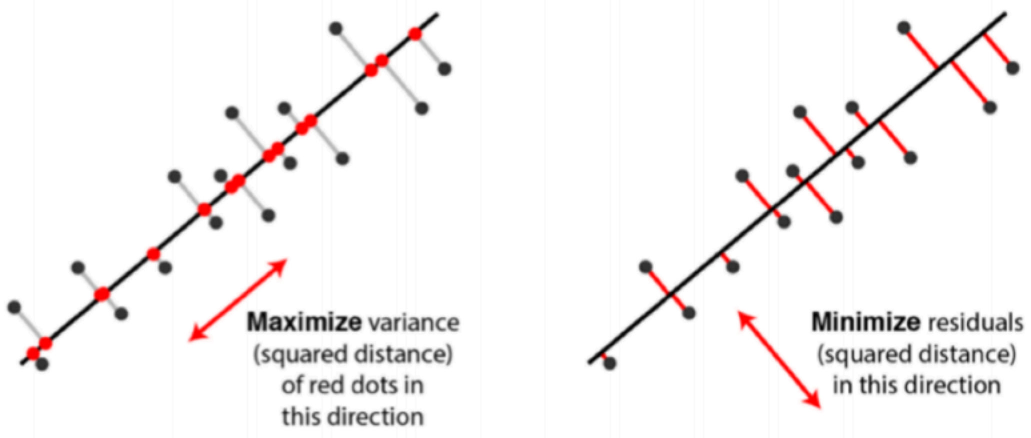
PCA in one dimension, 2 equivalent views

Goal: find a k < d-dimensional representation of X

For k = 1:

Choose $\vec{v} \in \mathbb{R}^d, ||v|| = 1$
to minimize

$$\frac{1}{n}\sum_{i=1}^{n} dist(x_i, \text{line defined by } \vec{v})$$



Two equivalent views of principal component analysis.

PCA: a high-fidelity linear projection

Given $x_1, \ldots, x_n \in \mathbb{R}^d$, for $q \ll d$ find a compressed representation with $\lambda_1, \ldots, \lambda_n \in \mathbb{R}^q$ such that $x_i \approx \mu + \mathbf{V}_q \lambda_i$ and $\mathbf{V}_q^T \mathbf{V}_q = I$

$$\min_{\mu, \mathbf{V}_q, \{\lambda_i\}_i} \sum_{i=1}^{n} \|x_i - \mu - \mathbf{V}_q \lambda_i\|_2^2$$

Fix $\mathbf{V}_q$ and solve for μ, λ_i:

$$\mu = \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$\lambda_i = \mathbf{V}_q^T (x_i - \bar{x})$$

Which gives us:

$$\min_{\mathbf{V}_q} \sum_{i=1}^{N} ||(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^T (x_i - \bar{x})||^2.$$

$\mathbf{V}_q \mathbf{V}_q^T$ is a *projection matrix* that minimizes error in basis of size q

PCA: a high-fidelity linear projection

$$\sum_{i=1}^{N} ||(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^T (x_i - \bar{x})||_2^2$$

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

$$\mathbf{V}_q^T \mathbf{V}_q = I_q$$

$$\min_{\mathbf{V}_q} \sum_{i=1}^{N} ||(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^T (x_i - \bar{x})||_2^2 = \min_{\mathbf{V}_q} Tr(\Sigma) - Tr(\mathbf{V}_q^T \Sigma \mathbf{V}_q)$$

Eigenvalue decomposition

$\mathbf{V}_q$ are the first q eigenvectors of Σ

Minimize reconstruction error and capture the most variance in your data.

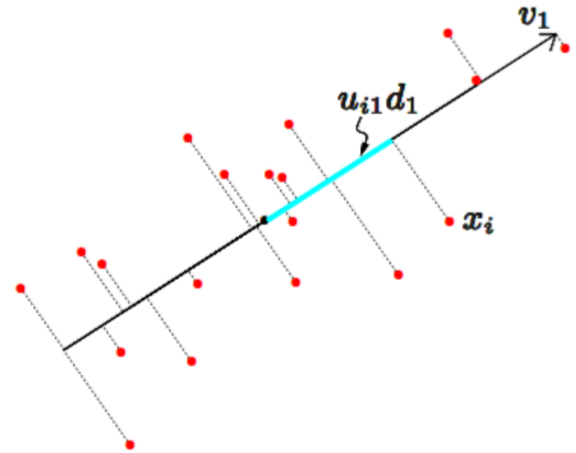
PCA: a high-fidelity linear projection

Given $x_i \in \mathbb{R}^d$ and some $q < d$ consider

$$\min_{\mathbf{V}_q} \sum_{i=1}^{N} ||(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^T (x_i - \bar{x})||^2.$$

where $\mathbf{V}_q = [v_1, v_2, \ldots, v_q]$ is orthonormal:

$$\mathbf{V}_q^T \mathbf{V}_q = I_q$$



$\mathbf{V}_q$ are the first q eigenvectors of Σ

$\mathbf{V}_q$ are the first q *principal components*

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

Principal Component Analysis (PCA) projects $(\mathbf{X} - \mathbf{1}\bar{x}^T)$ down onto $\mathbf{V}_q$

$$(\mathbf{X} - \mathbf{1}\bar{x}^T)\mathbf{V}_q = \mathbf{U}_q \text{diag}(d_1, \ldots, d_q) \qquad \mathbf{U}_q^T \mathbf{U}_q = I_q$$

PCA Algorithm

PCA

input
A matrix of m examples $X \in \mathbb{R}^{m,d}$
number of components n
if $(m > d)$
$A = X^\top X$
Let $\mathbf{u}_1, \ldots, \mathbf{u}_n$ be the eigenvectors of A with largest eigenvalues
else
$B = X X^\top$
Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the eigenvectors of B with largest eigenvalues
for $i = 1, \ldots, n$ set $\mathbf{u}_i = \frac{1}{\|X^\top \mathbf{v}_i\|} X^\top \mathbf{v}_i$
output: $\mathbf{u}_1, \ldots, \mathbf{u}_n$

How do we compute the principal components?

1. Power iteration
2. **Solving for a singular value decomposition (SVD)**

Singular Value Decomposition (SVD)

Theorem (SVD): Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{S} \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.

$$\mathbf{A}^T\mathbf{A}v_i =$$

$$\mathbf{A}\mathbf{A}^T u_i =$$

Singular Value Decomposition (SVD)

Theorem (SVD): Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{S} \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.

$$\mathbf{A}^T\mathbf{A}v_i = \mathbf{S}_{i,i}^2 v_i$$

$$\mathbf{A}\mathbf{A}^T u_i = \mathbf{S}_{i,i}^2 u_i$$

$\mathbf{V}$ are the first r eigenvectors of $\mathbf{A}^T\mathbf{A}$ with eigenvalues $\text{diag}(\mathbf{S})$
$\mathbf{U}$ are the first r eigenvectors of $\mathbf{A}\mathbf{A}^T$ with eigenvalues $\text{diag}(\mathbf{S})$

Computational complexity of SVD

Theorem (SVD): Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $\mathbf{A} = \mathbf{USV}^T$ where $\mathbf{S} \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.

at most r singular values

irrelevant | n - m | last columns of U

Computing the remaining economy-sized SVD takes time O(n m r)

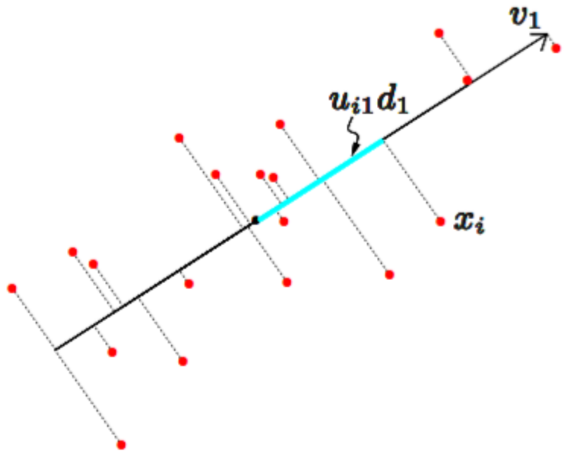
Linear projections

Given $x_i \in \mathbb{R}^d$ and some $q < d$ consider

$$\min_{\mathbf{V}_q} \sum_{i=1}^{N} ||(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^T (x_i - \bar{x})||^2.$$

where $\mathbf{V}_q = [v_1, v_2, \dots, v_q]$ is orthonormal:

$$\mathbf{V}_q^T \mathbf{V}_q = I_q$$



$\mathbf{V}_q$ are the first q eigenvectors of Σ

$\mathbf{V}_q$ are the first q *principal components*

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

Principal Component Analysis (PCA) projects $(\mathbf{X} - \mathbf{1}\bar{x}^T)$ down onto $\mathbf{V}_q$

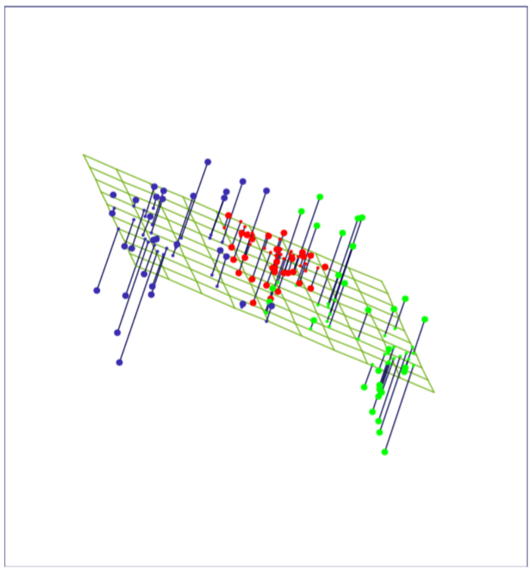
$$(\mathbf{X} - \mathbf{1}\bar{x}^T)\mathbf{V}_q = \mathbf{U}_q \text{diag}(d_1, \dots, d_q) \qquad \mathbf{U}_q^T \mathbf{U}_q = I_q$$

Singular Value Decomposition defined as

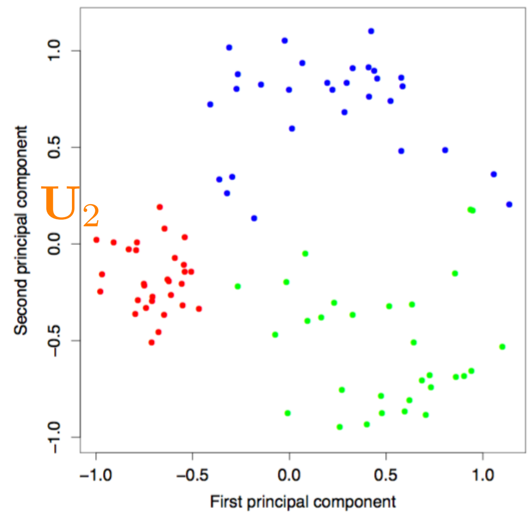
$$\mathbf{X} - \mathbf{1}\bar{x}^T = \mathbf{USV}^T$$

Dimensionality reduction

$\mathbf{V}_q$ are the first q eigenvectors of Σ and *SVD* $\mathbf{X} - \mathbf{1}\bar{x}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T$



$\mathbf{X} - \mathbf{1}\bar{x}^T$

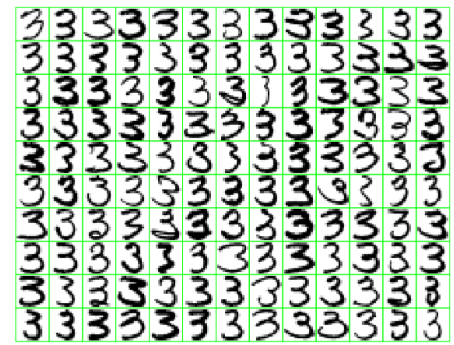


$\mathbf{U}_1$

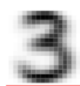
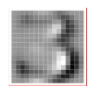
Dimensionality reduction

$\mathbf{V}_q$ are the first q eigenvectors of Σ and *SVD* $\mathbf{X} - \mathbf{1}\bar{x}^T = \mathbf{USV}^T$

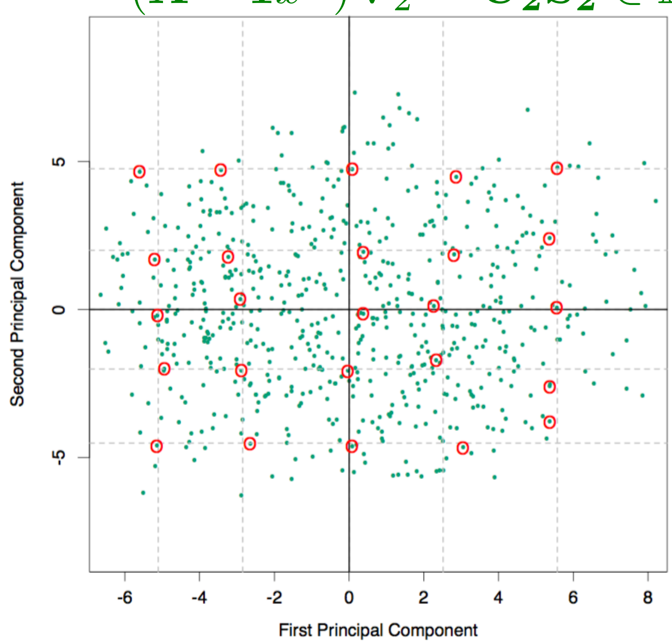
Handwritten 3's, 16x16 pixel image so that $x_i \in \mathbb{R}^{256}$

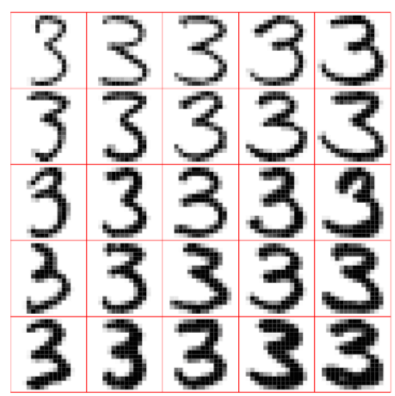


$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

=  $+ \lambda_1 \cdot$  $+ \lambda_2 \cdot$ .

$$(\mathbf{X} - \mathbf{1}\bar{x}^T)\mathbf{V}_2 = \mathbf{U_2 S_2} \in \mathbb{R}^{n \times 2}$$





$\mathrm{diag}(\mathbf{S})$

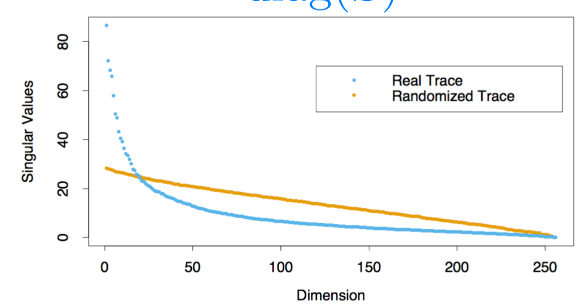


FIGURE 14.24. *The 256 singular values for the digitized threes, compared to those for a randomized version of the data (each column of* **X** *was scrambled).*

PCA Algorithm

PCA

input

A matrix of m examples $X \in \mathbb{R}^{m,d}$

number of components n

if $(m > d)$

$A = X^\top X$

Let $\mathbf{u}_1, \ldots, \mathbf{u}_n$ be the eigenvectors of A with largest eigenvalues

else

$B = X X^\top$

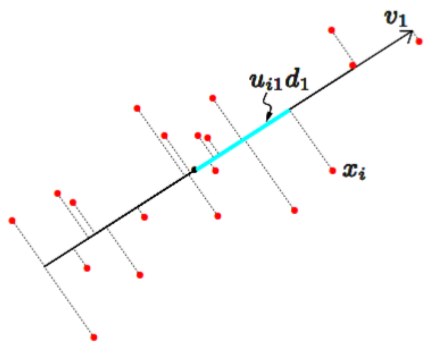
Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the eigenvectors of B with largest eigenvalues

for $i = 1, \ldots, n$ set $\mathbf{u}_i = \frac{1}{\|X^\top \mathbf{v}_i\|} X^\top \mathbf{v}_i$

output: $\mathbf{u}_1, \ldots, \mathbf{u}_n$

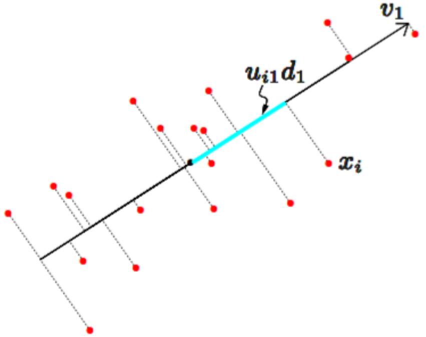
Power method - one at a time

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T \qquad v_* = \arg\max_v \, v^T \Sigma v$$



Power method - one at a time

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T \qquad v_* = \arg\max_v v^T \Sigma v$$



$$z_0 \sim \mathcal{N}(0, I)$$

Iterate: $$z_{t+1} = \frac{\Sigma z_t}{\|\Sigma z_t\|_2}$$

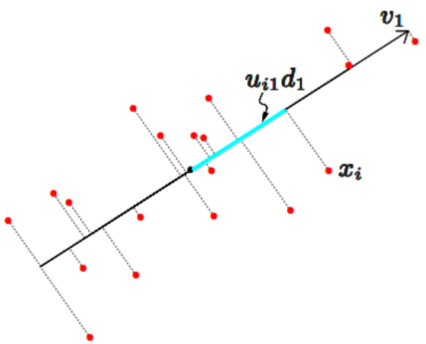
To analyze write: $$\Sigma = \mathbf{V}\mathbf{D}\mathbf{V}^T \qquad z_t =: \mathbf{V}\alpha_t$$

$$\alpha_{t+1} = \mathbf{V}^T z_{t+1} = \frac{\mathbf{V}^T \Sigma z_t}{\|\Sigma z_t\|} = \frac{\mathbf{D}\alpha_t}{\|\mathbf{D}\alpha_t\|} = \frac{\mathbf{D}^2\alpha_{t-1}}{\|\mathbf{D}^2\alpha_{t-1}\|} = \frac{\mathbf{D}^t\alpha_0}{\|\mathbf{D}^t\alpha_0\|}$$

$$\mathbf{D}^t = (\mathbf{D}_{1,1})^t(\mathbf{D}/\mathbf{D}_{1,1})^t \to (\mathbf{D}_{1,1})^t \mathbf{e}_1 \mathbf{e}_1^T \text{ since } \mathbf{D}_{i,i}/\mathbf{D}_{1,1} < 1$$

Power method - one at a time

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T \qquad v_* = \arg\max_v v^T \Sigma v$$



$$z_0 \sim \mathcal{N}(0, I)$$

Iterate: $z_{t+1} = \frac{\Sigma z_t}{\|\Sigma z_t\|_2}$

To analyze write: $\Sigma = \mathbf{V}\mathbf{D}\mathbf{V}^T \qquad z_t =: \mathbf{V}\alpha_t$