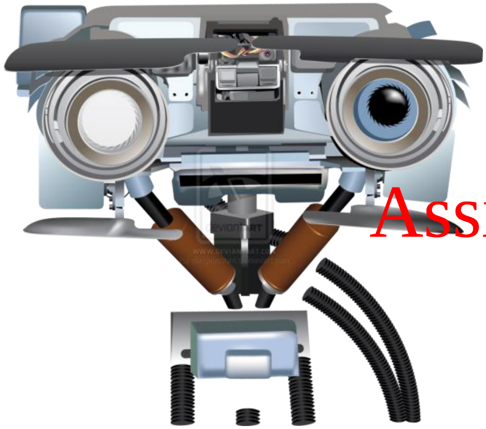
Announcements

Reminder: ps2 due tonight at midnight (Boston)

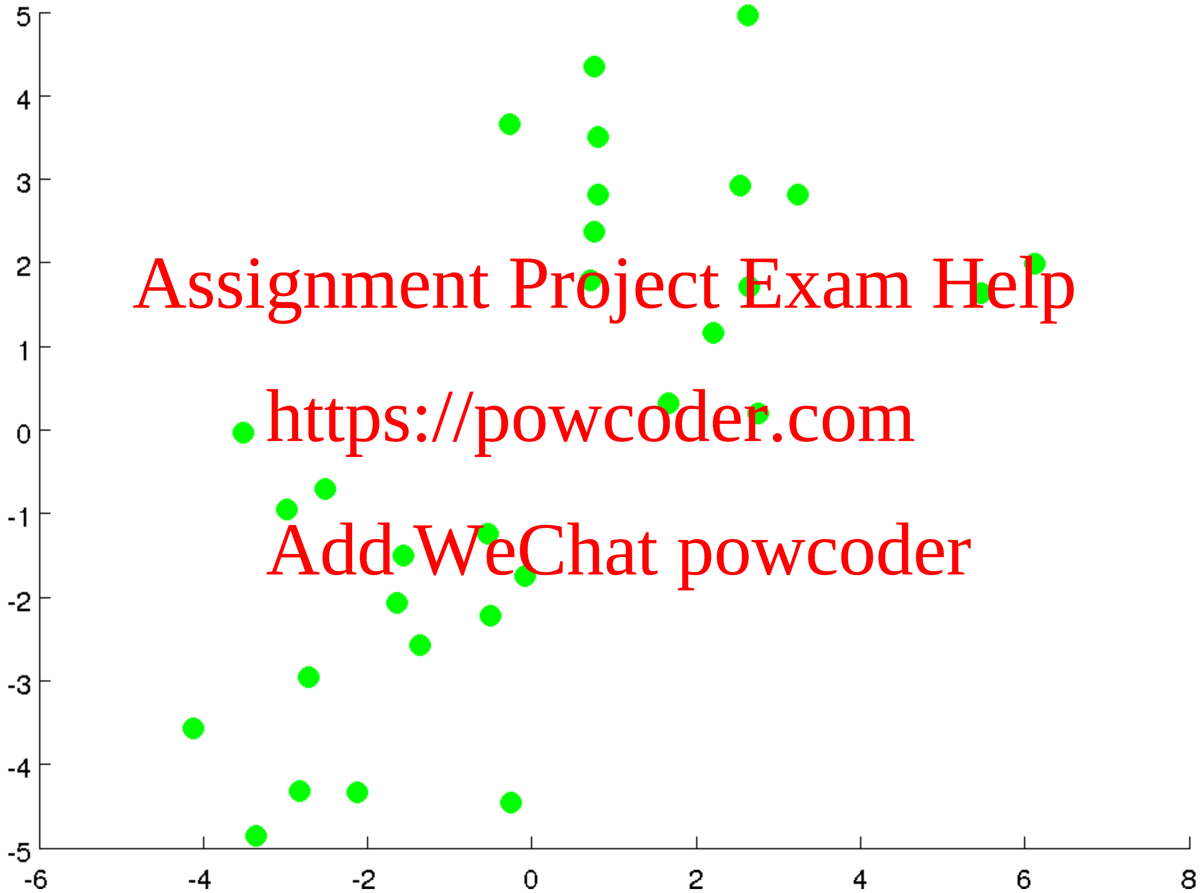
- Self-Grading form for ps1 out tomorrow 9/25 (1 week to turn in)

- Self-Grading form for ps2 out Monday 9/28 (1 week to turn in)

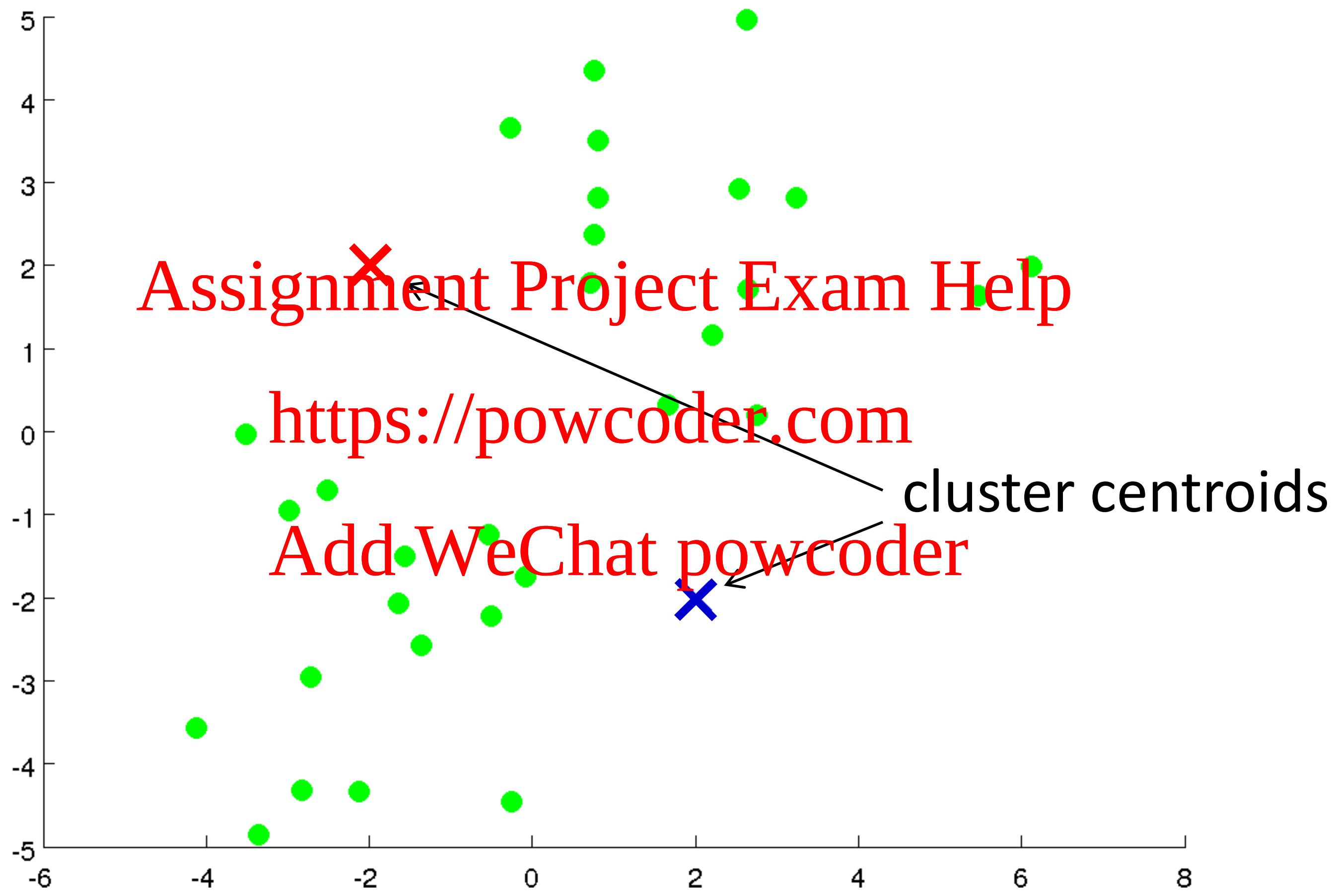


Unsupervised Learning II

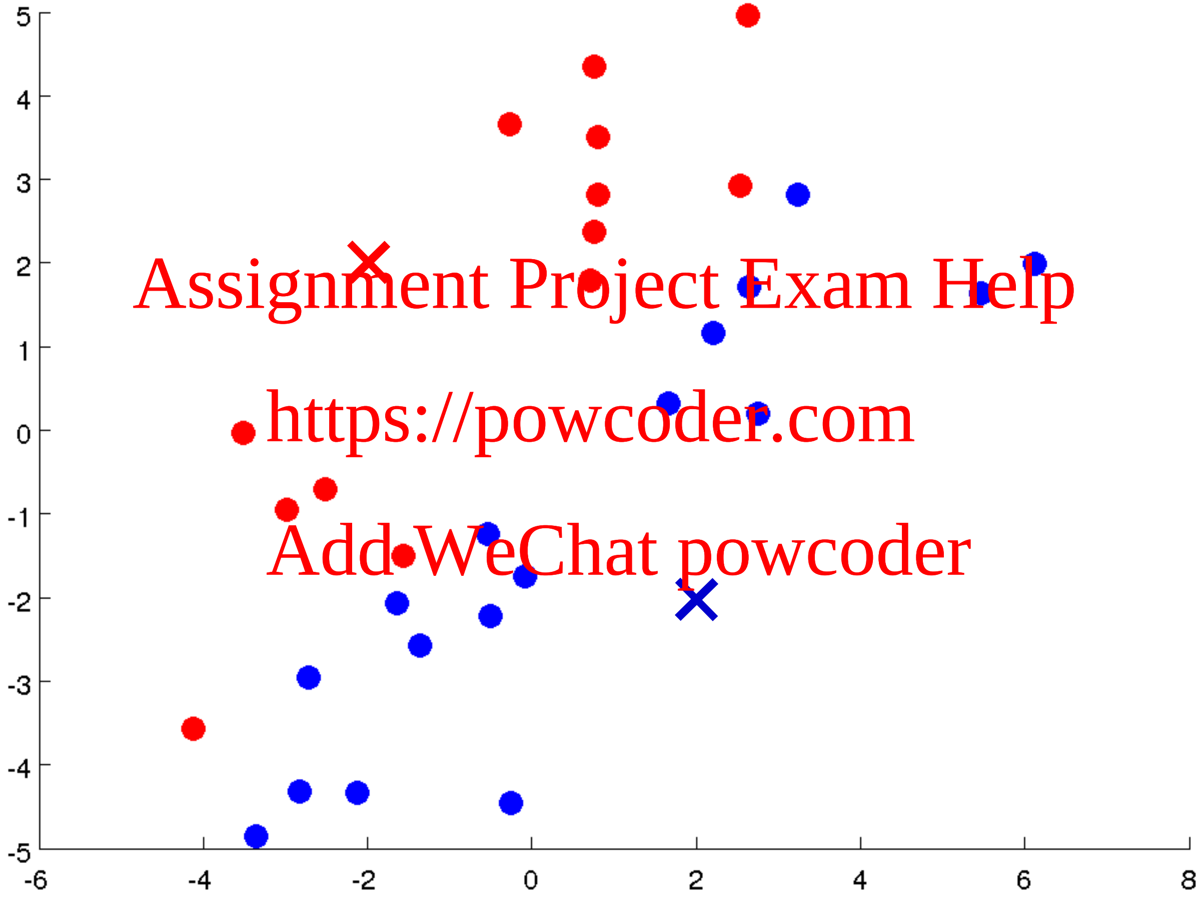
Agglomerative Clustering



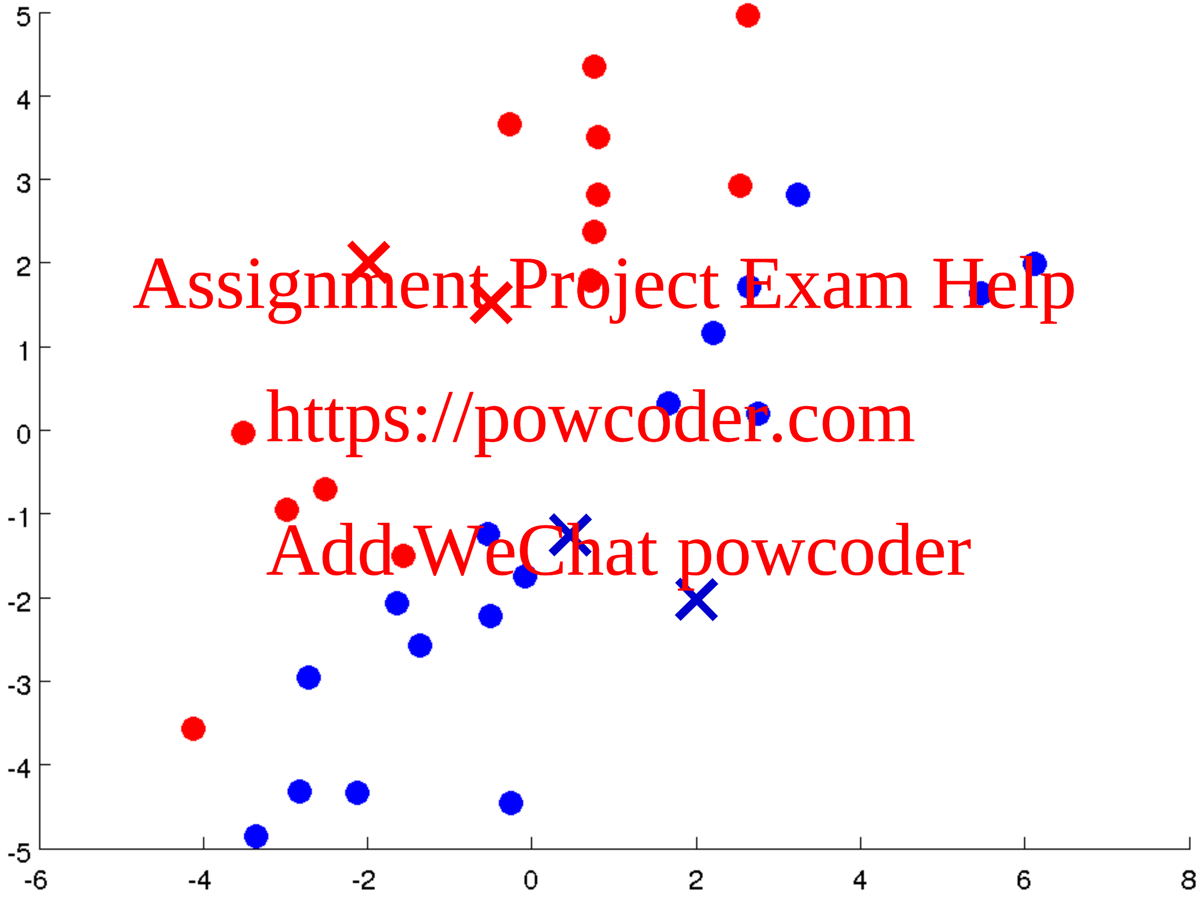
slide credit: Andrew Ng



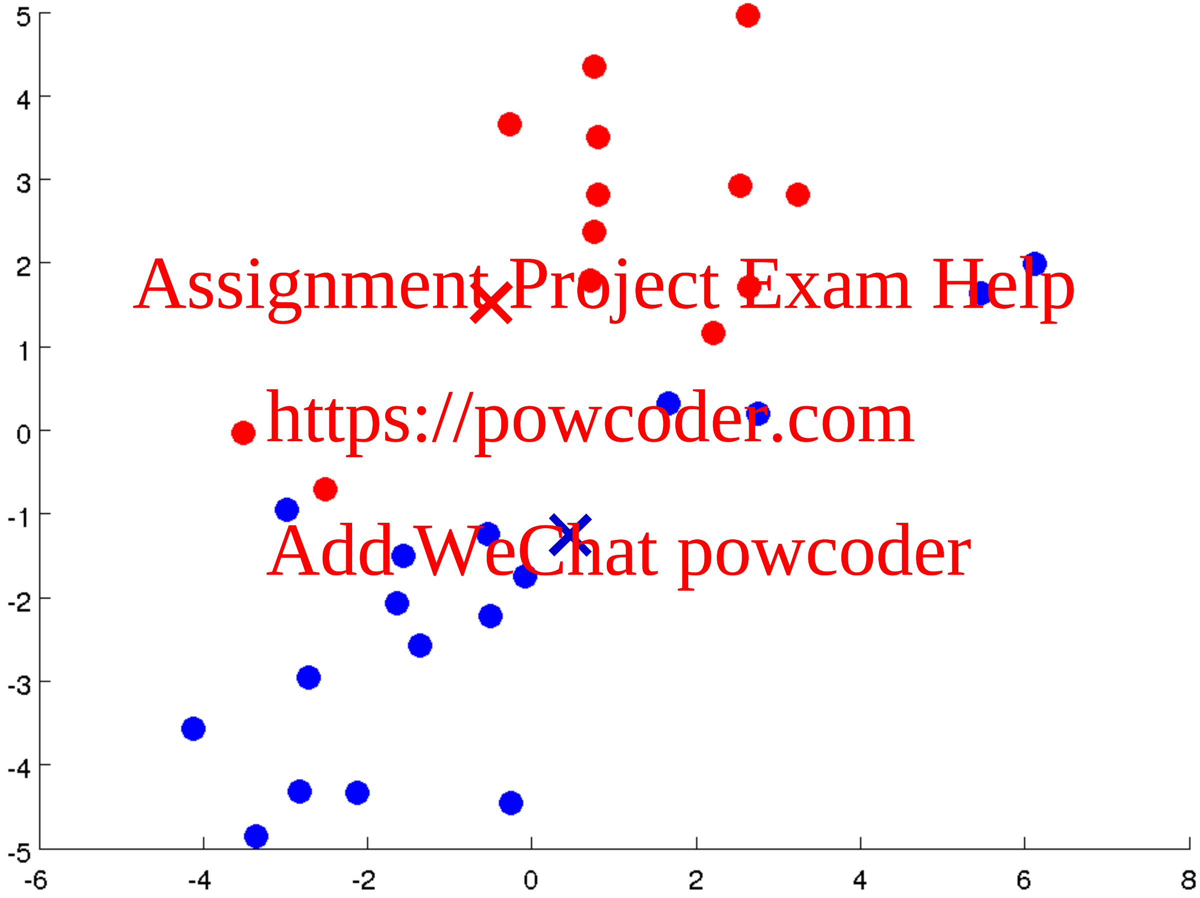
slide credit: Andrew Ng



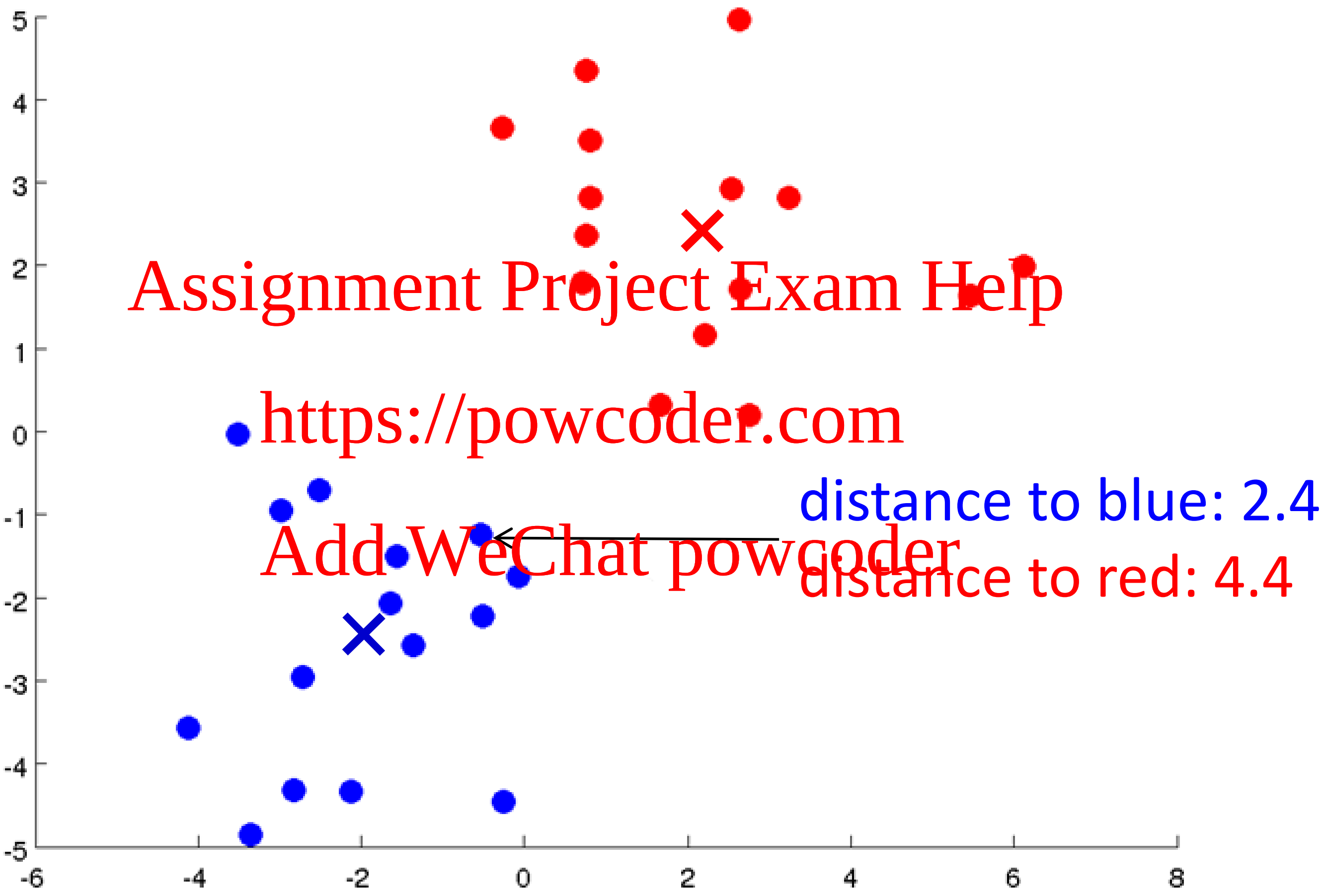
slide credit: Andrew Ng



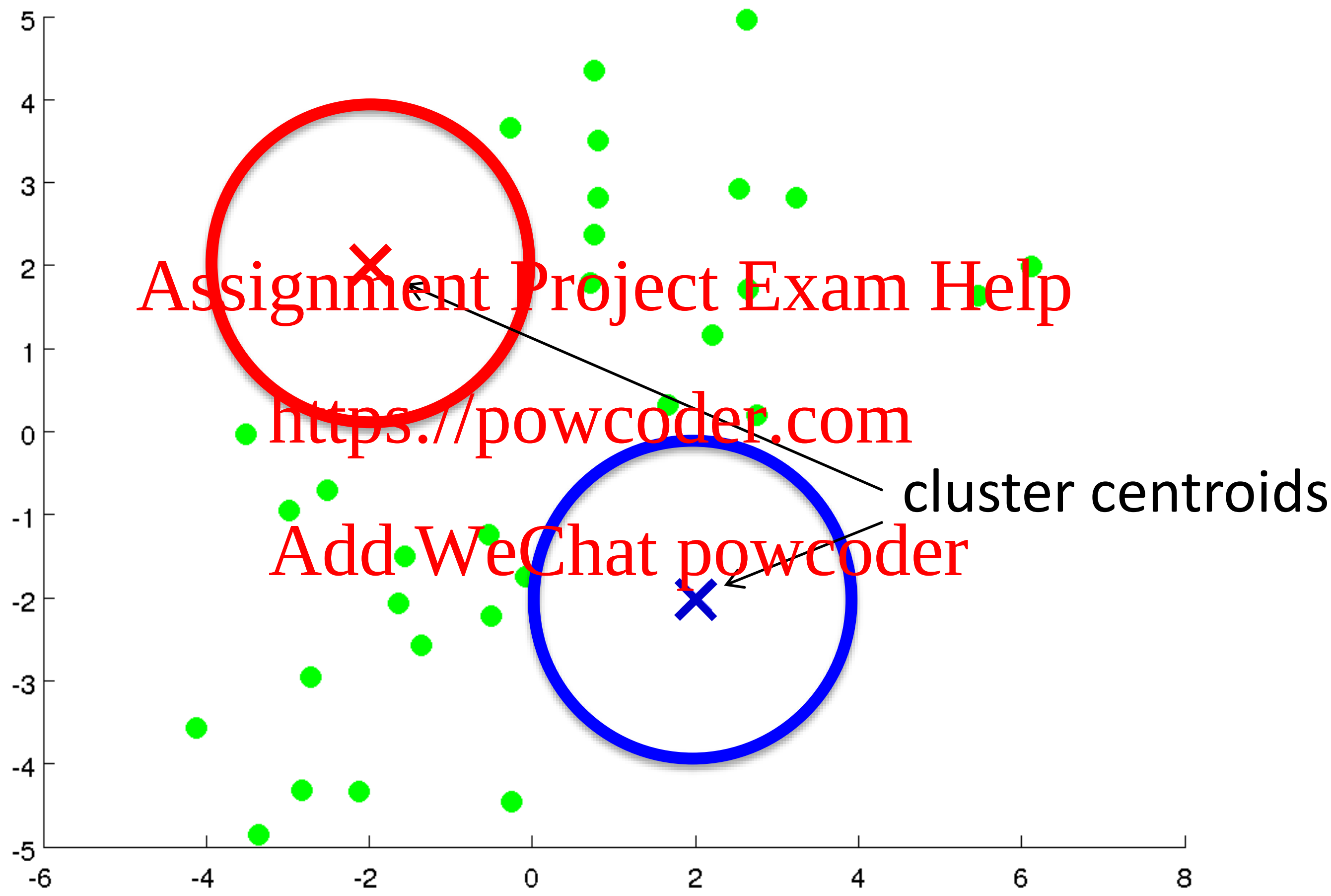
slide credit: Andrew Ng



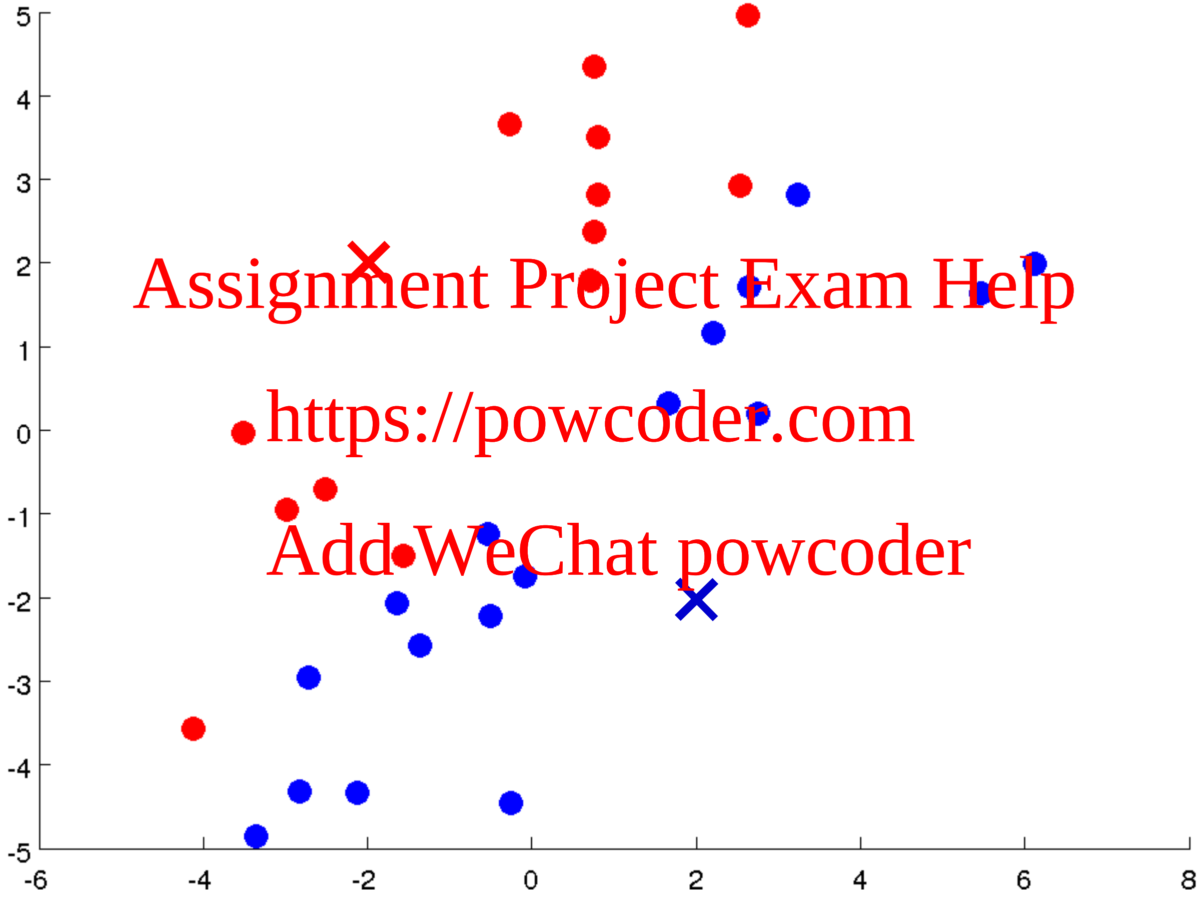
slide credit: Andrew Ng



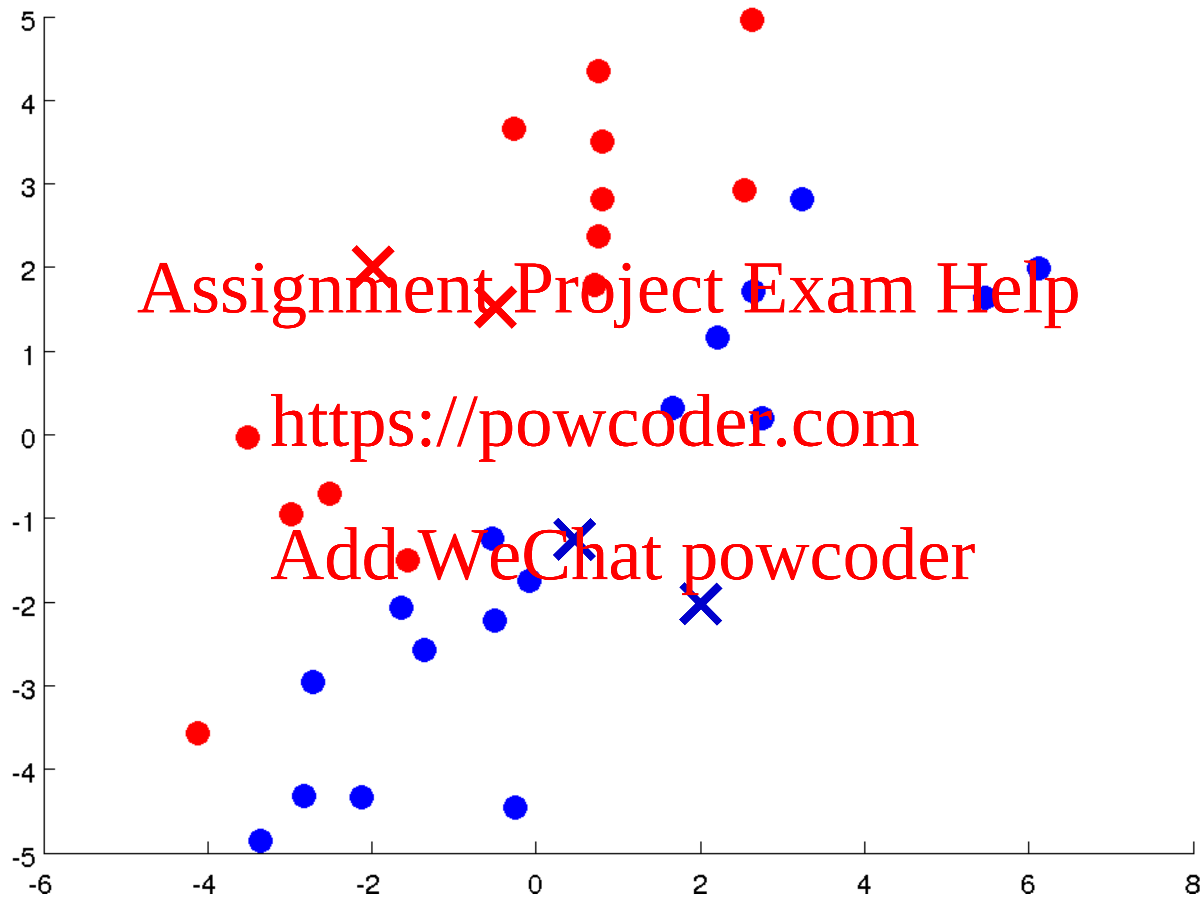
slide credit: Andrew Ng



slide credit: Andrew Ng

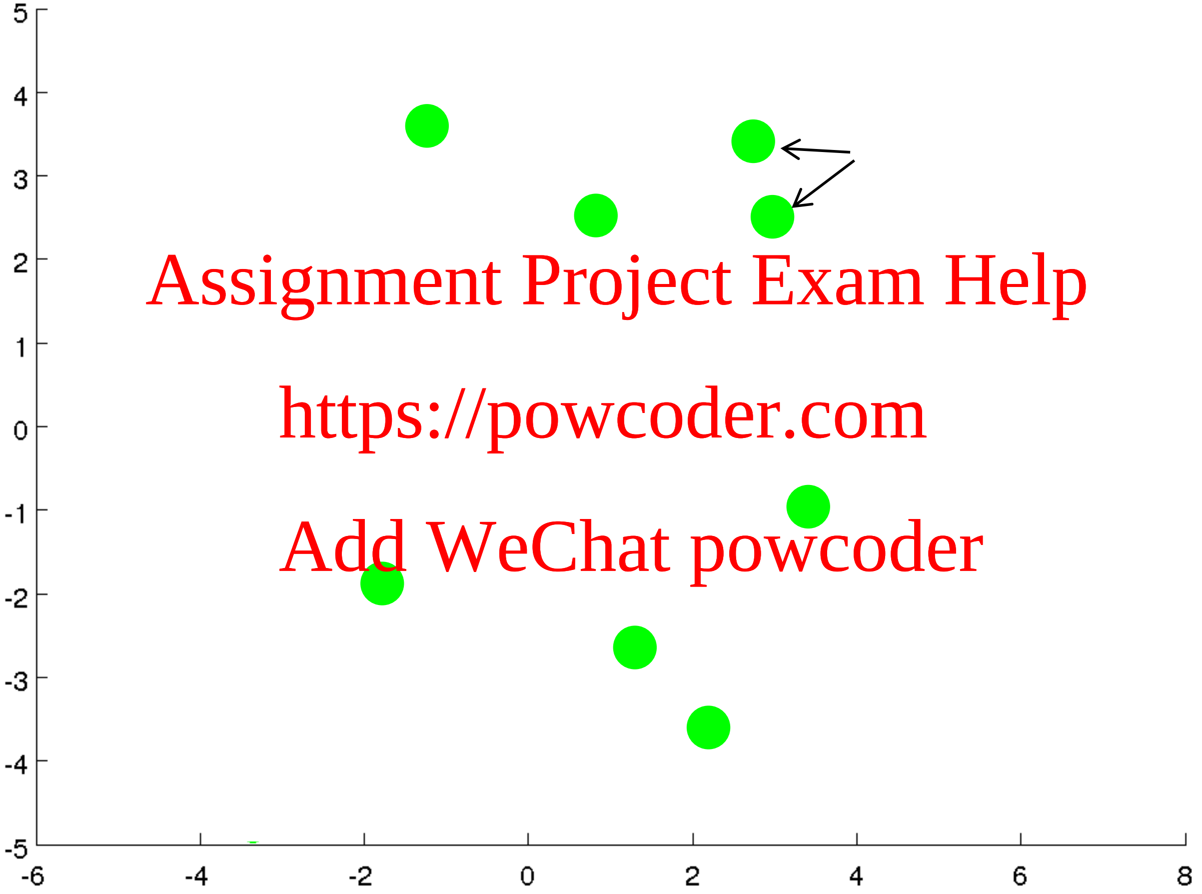


slide credit: Andrew Ng

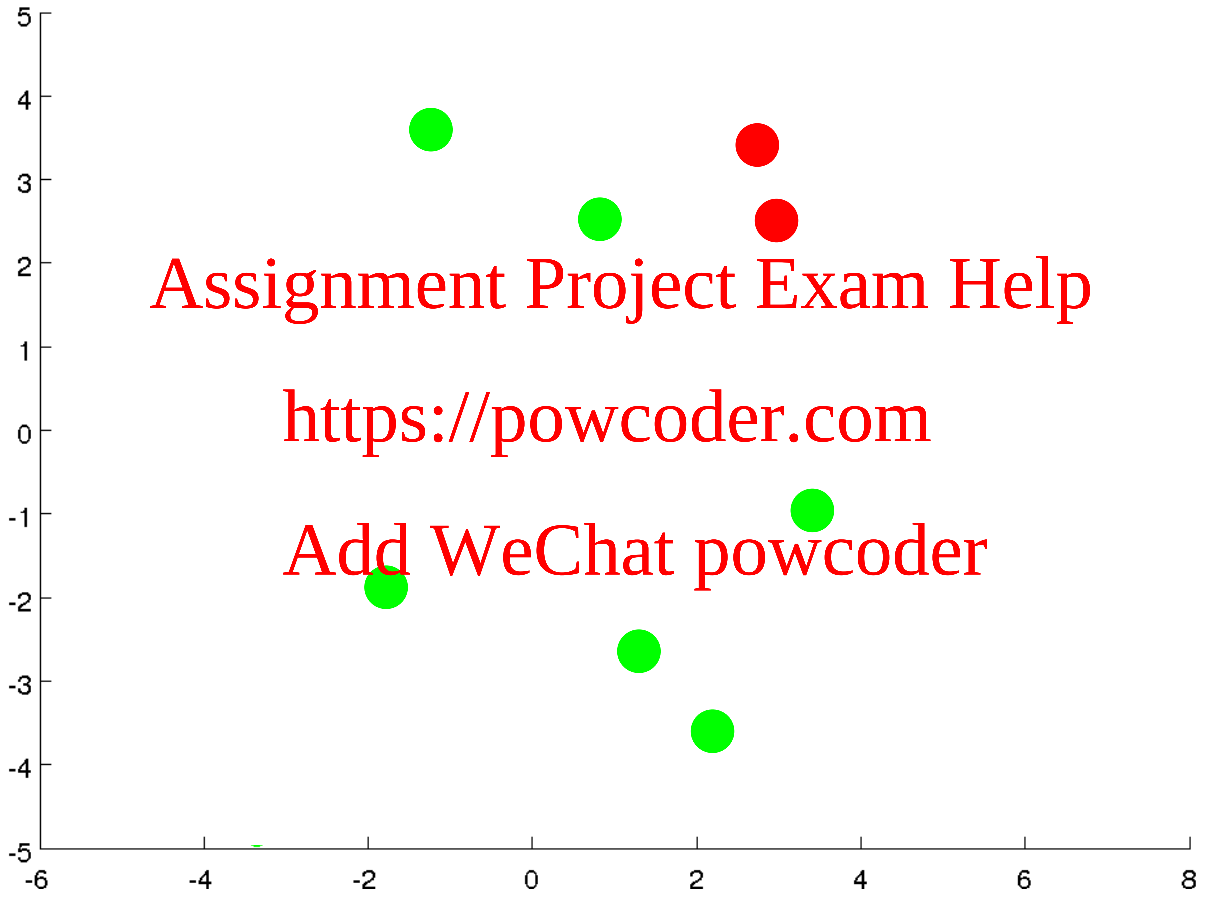


slide credit: Andrew Ng

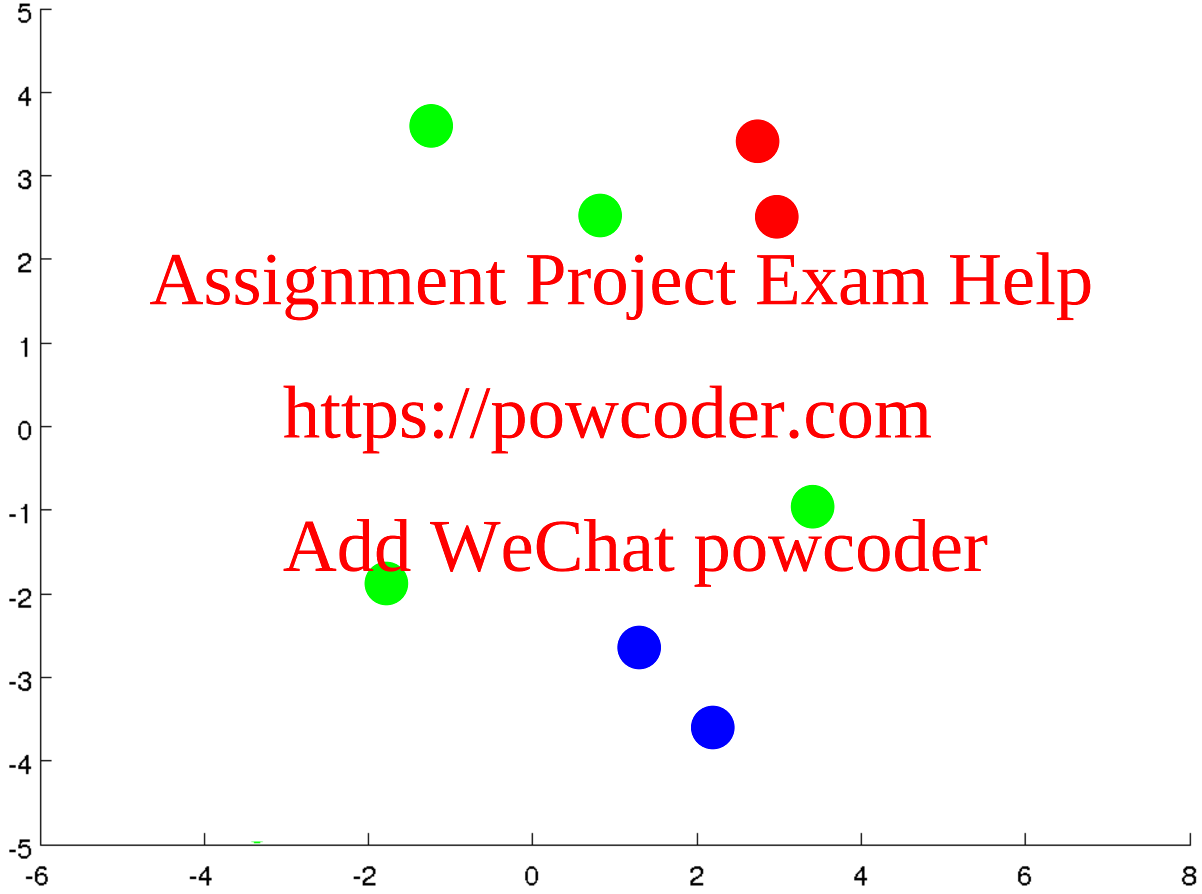
Iteratively combine the two closest points



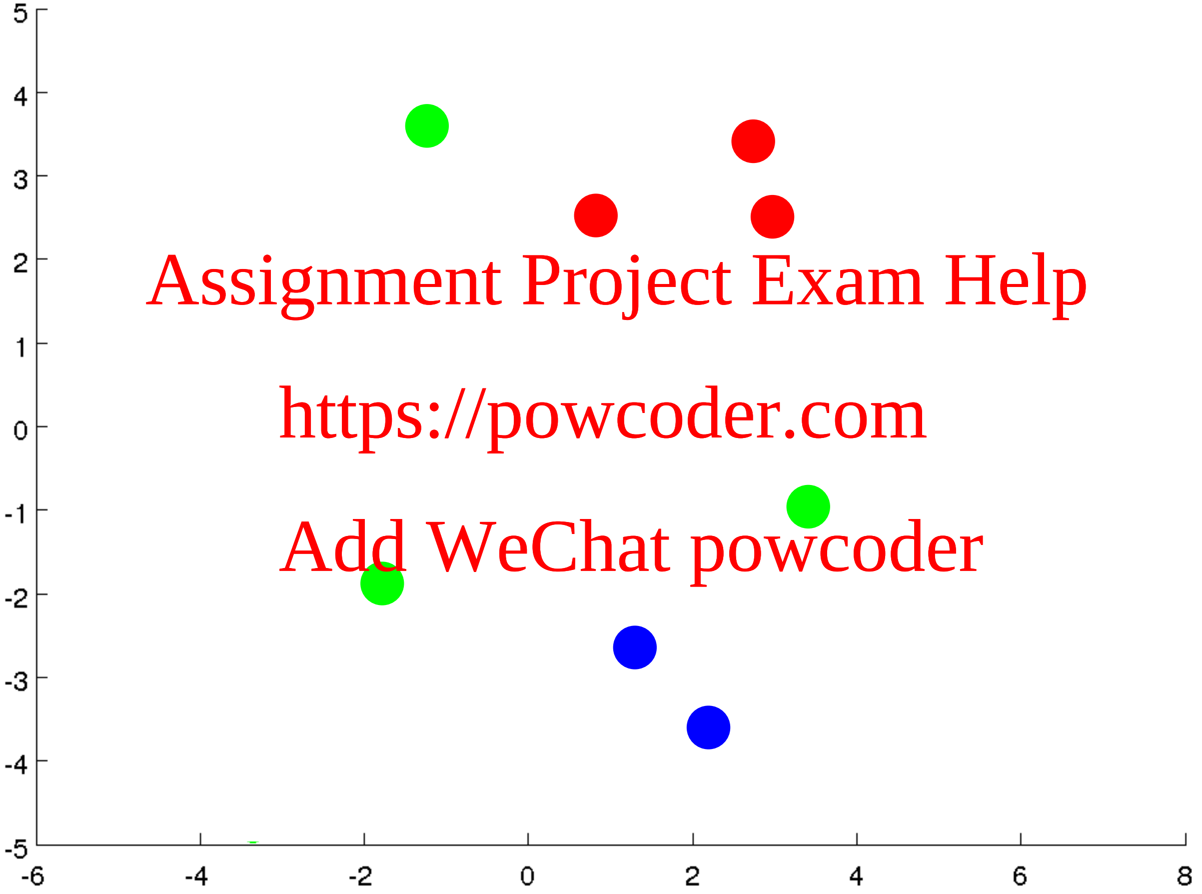
Iteratively combine the two closest points



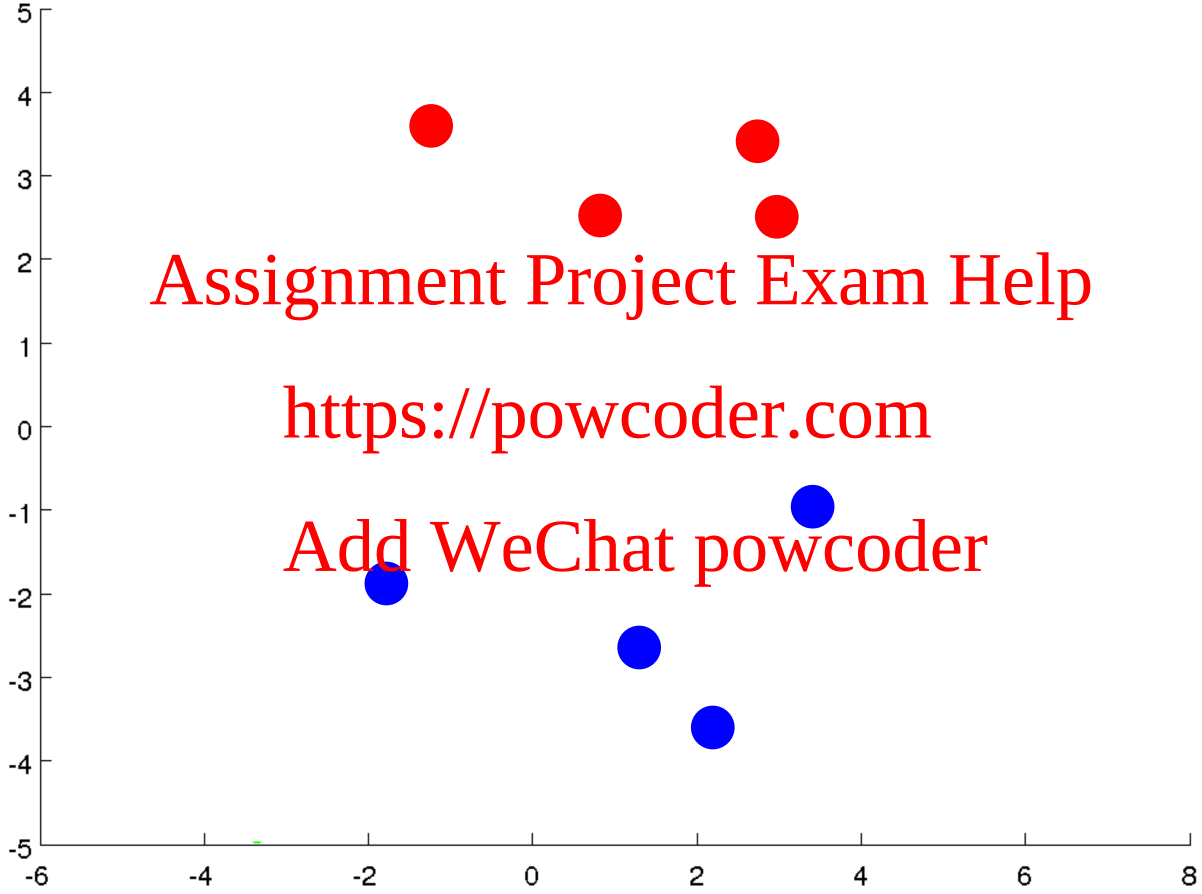
Iteratively combine the two closest points



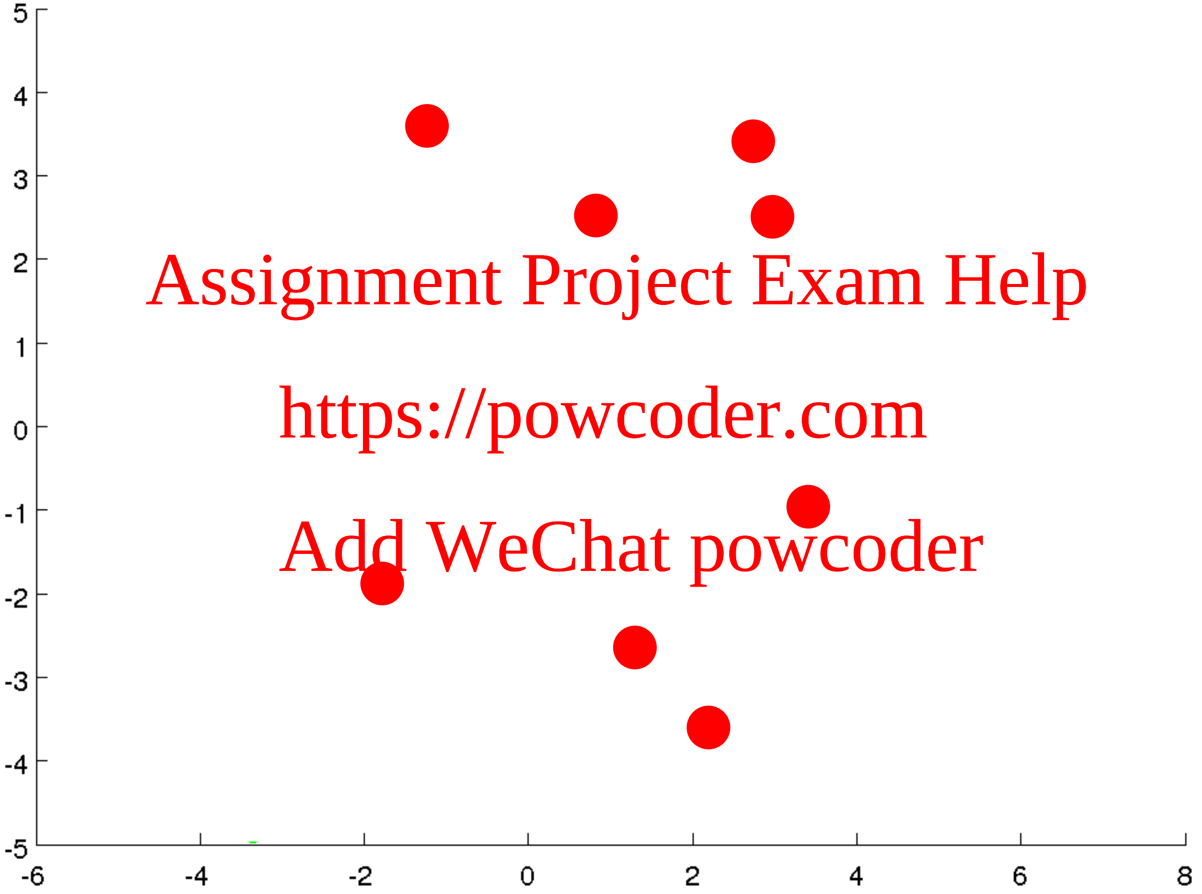
Iteratively combine the two closest points



Iteratively combine the two closest points



Iteratively combine the two closest points



Agglomerative Clustering Example

(bottom-up clustering)

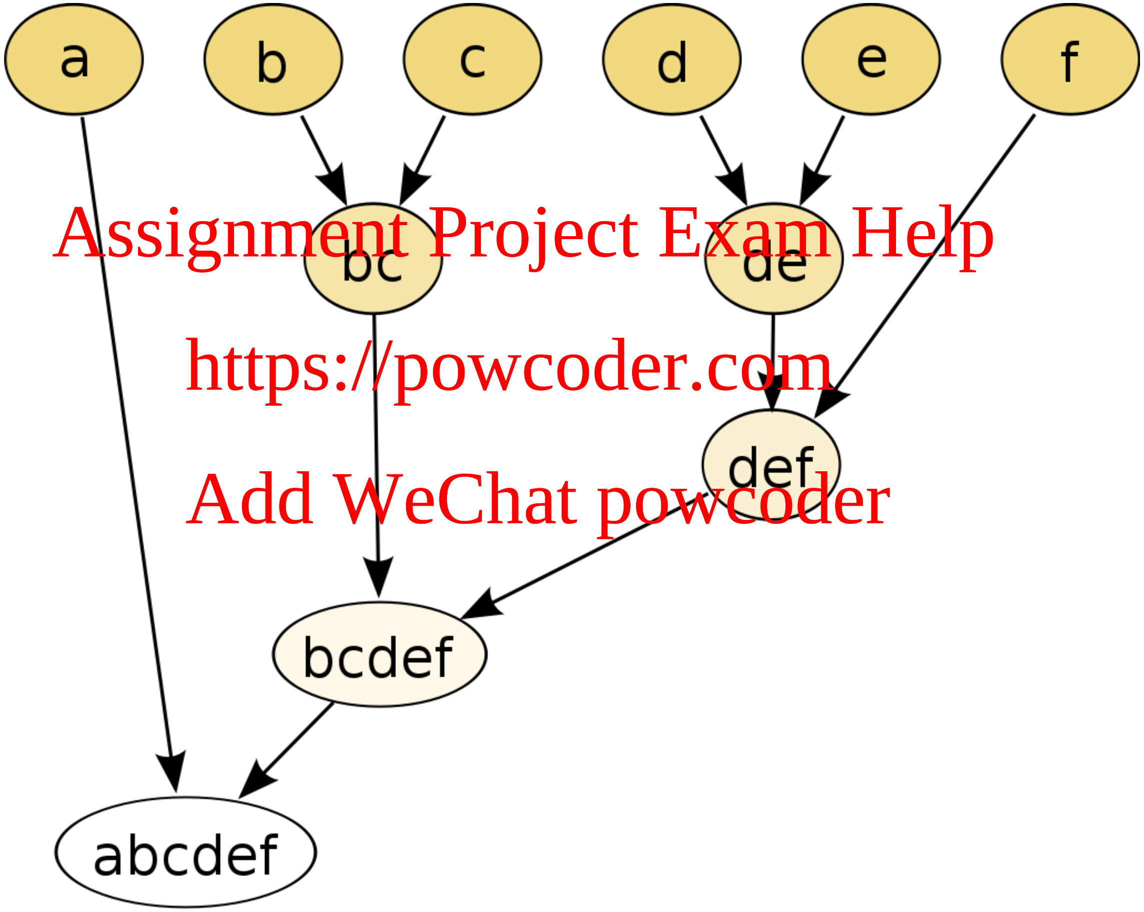
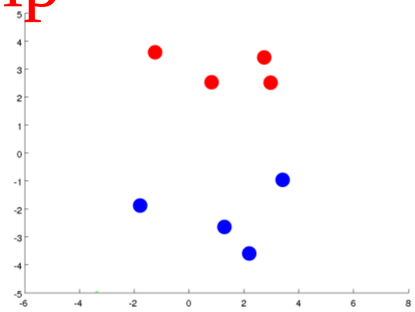


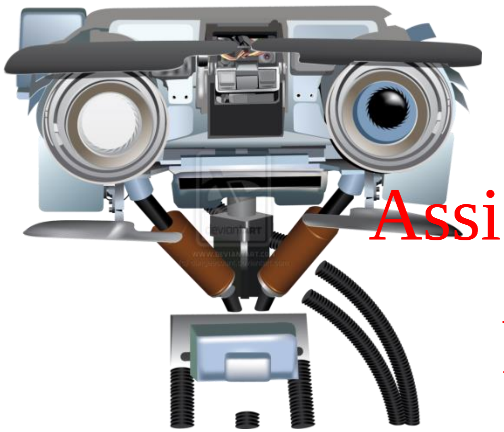
Image source: https://en.wikipedia.org/wiki/Hierarchical_clustering

When do we stop combining?

- Select based on prior knowledge or task performance (e.g. you know there are two categories of data)



- Choose cost threshold to stop combining



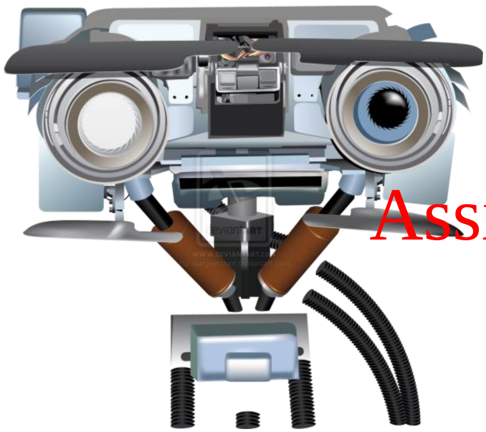
Unsupervised Learning II

Continuous Latent Variables

Today

- **Applications of clustering**: vector quantization, data compression

- **Continuous latent variables**: principal component analysis



Unsupervised Learning II

Applications of Clustering

Application of Clustering: Vector Quantization

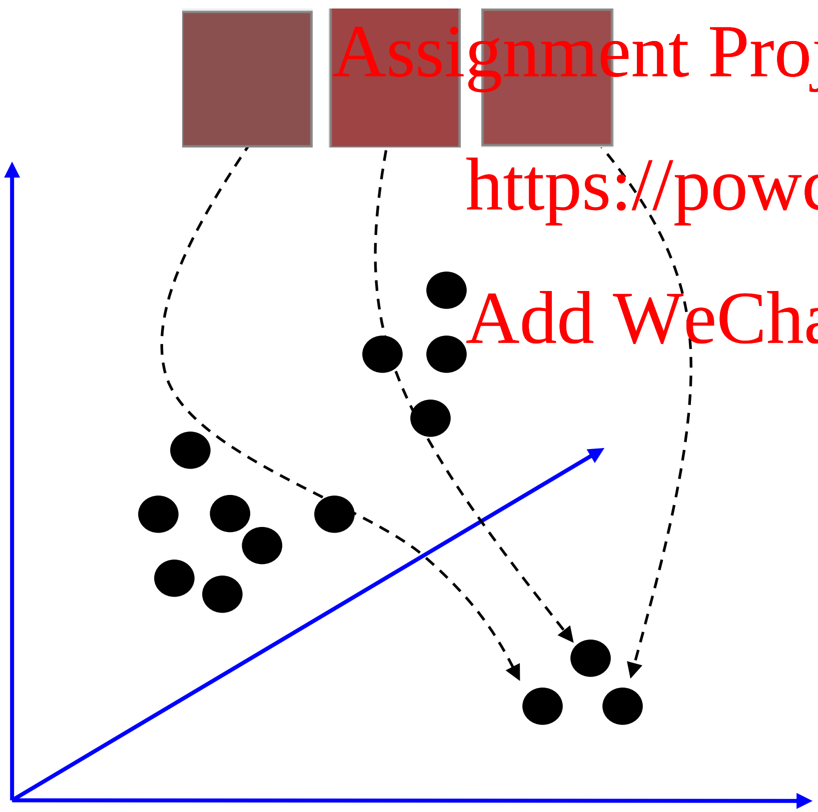
$$x^{(i)} = \begin{bmatrix} 138 \\ 80 \\ 79 \end{bmatrix} \quad \rightarrow \quad \mu_{c^i}$$

- Compress an image using clustering
- Each {R, G, B} pixel value is an input vector $x^{(i)}$ (255 x 255 x 255 possible values)
- Cluster into K clusters (using k-means)
- Replace each vector by its cluster's index $c^{(i)}$ (K possible values)
- For display, show the mean μ_{c^i}

Vector quantization: color values

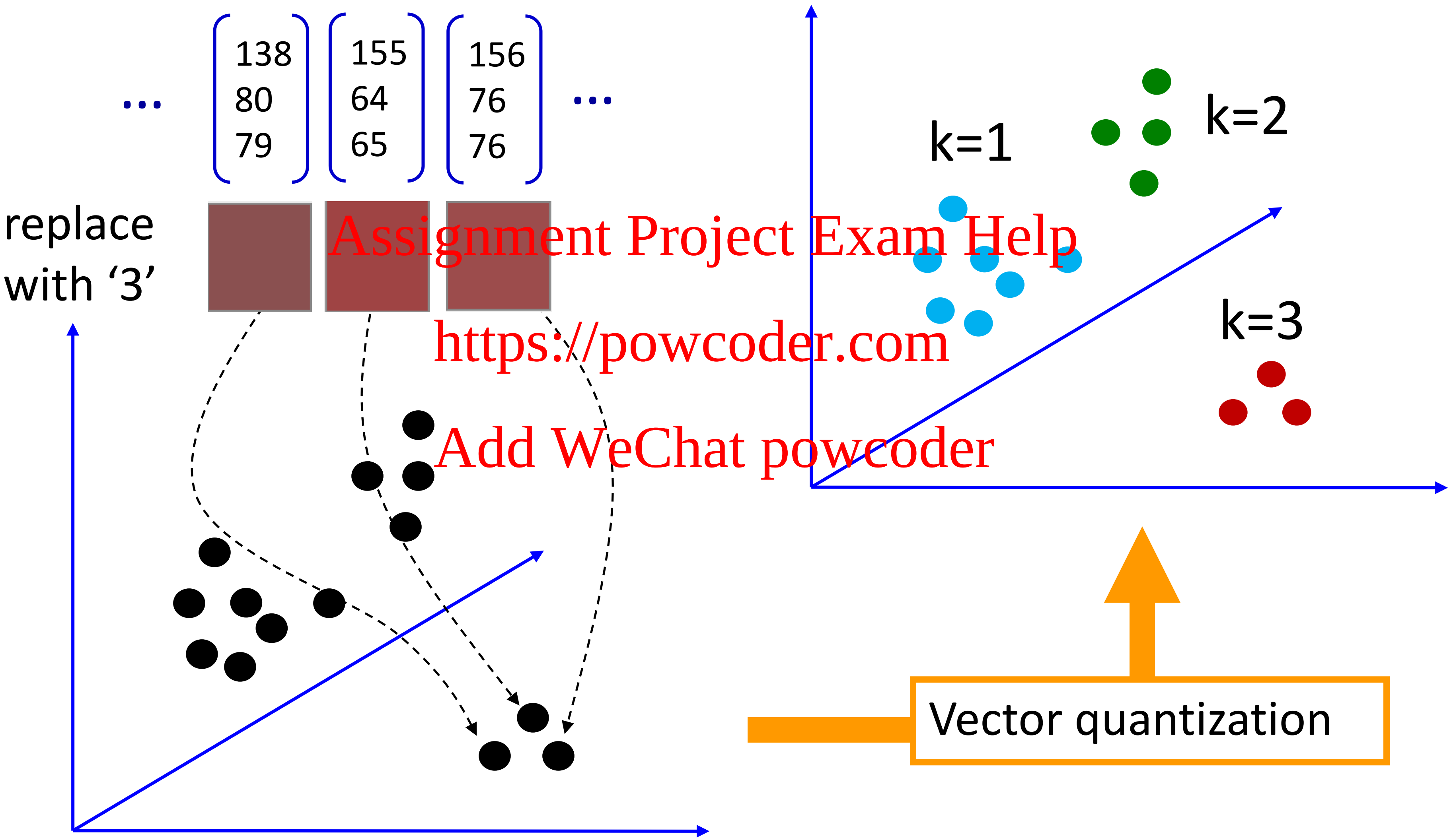
Example: R, G, B vectors

$$\cdots \begin{pmatrix} 138 \\ 80 \\ 79 \end{pmatrix} \begin{pmatrix} 155 \\ 64 \\ 65 \end{pmatrix} \begin{pmatrix} 156 \\ 76 \\ 76 \end{pmatrix} \cdots$$



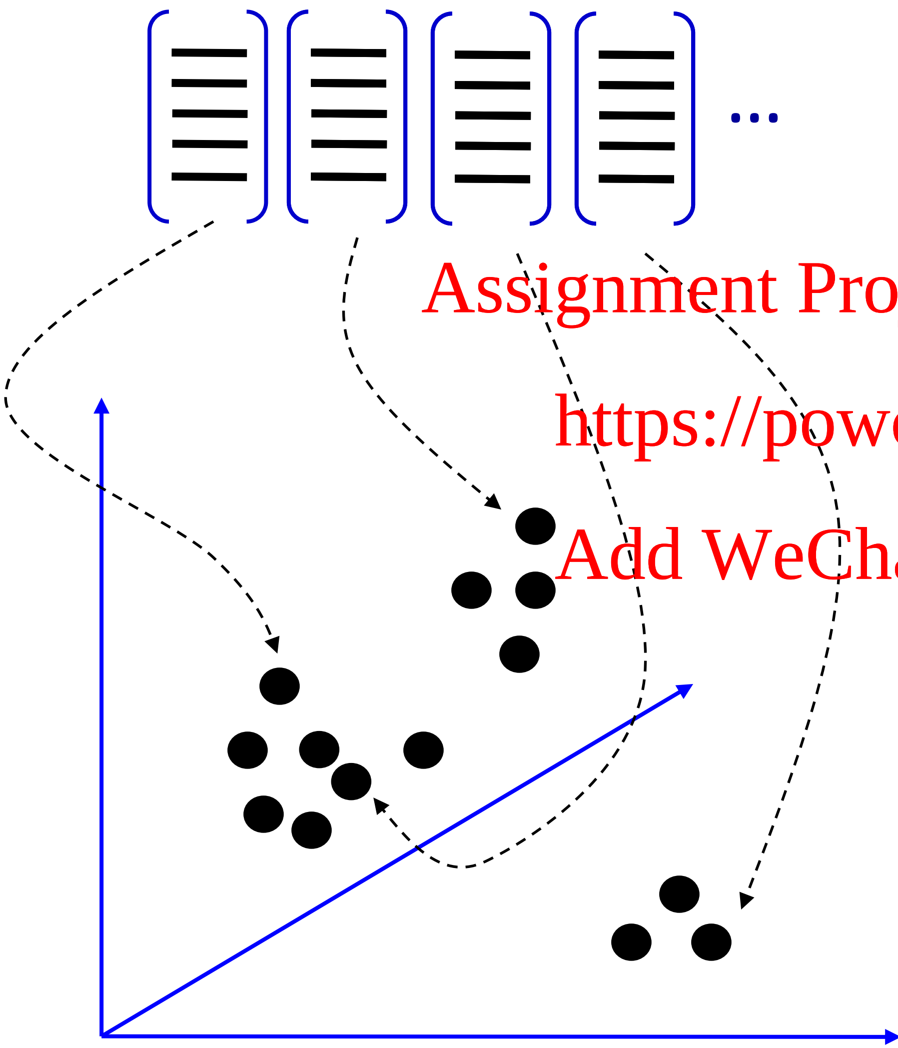
Slide credit: Josef Sivic

Vector quantization: color values



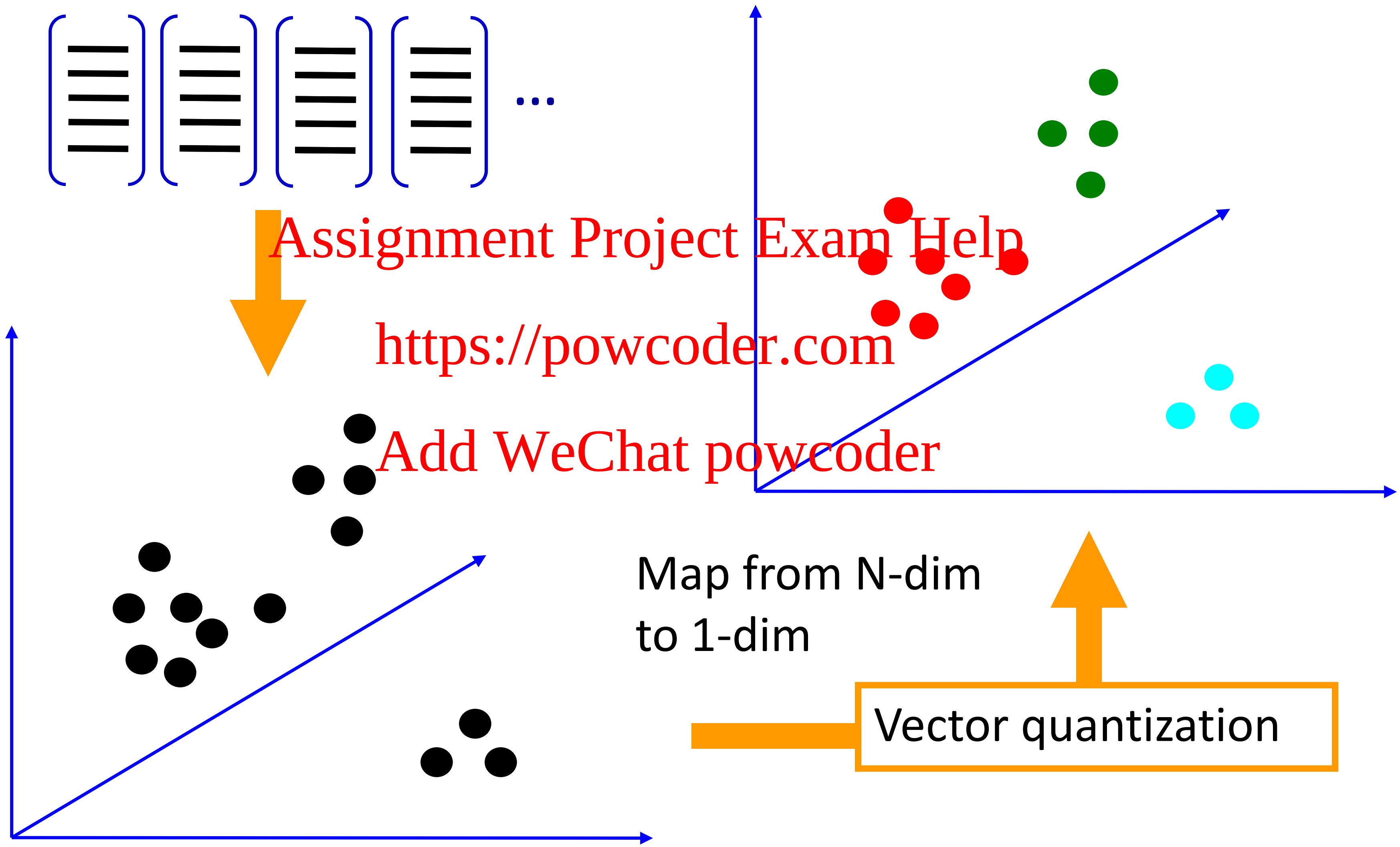
Slide credit: Josef Sivic

Vector quantization: general case



Slide credit: Josef Sivic

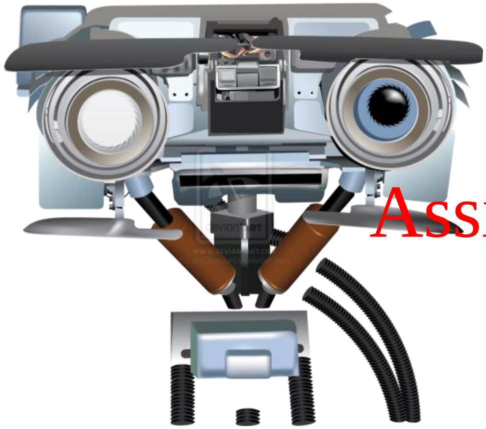
Vector quantization: general case



Slide credit: Josef Sivic

K-Means for Image Compression

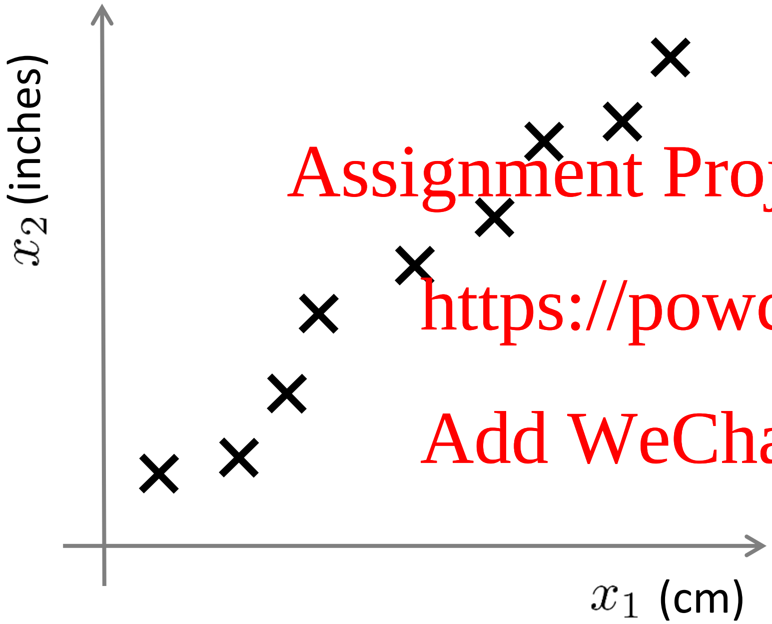
Figure 9.3 Two examples of the application of the K-means clustering algorithm to image segmentation showing the initial images together with their K-means segmentations obtained using various values of K. This also illustrates of the use of vector quantization for data compression, in which smaller values of K give higher compression at the expense of poorer image quality.



Unsupervised Learning II

Continuous Latent Variables

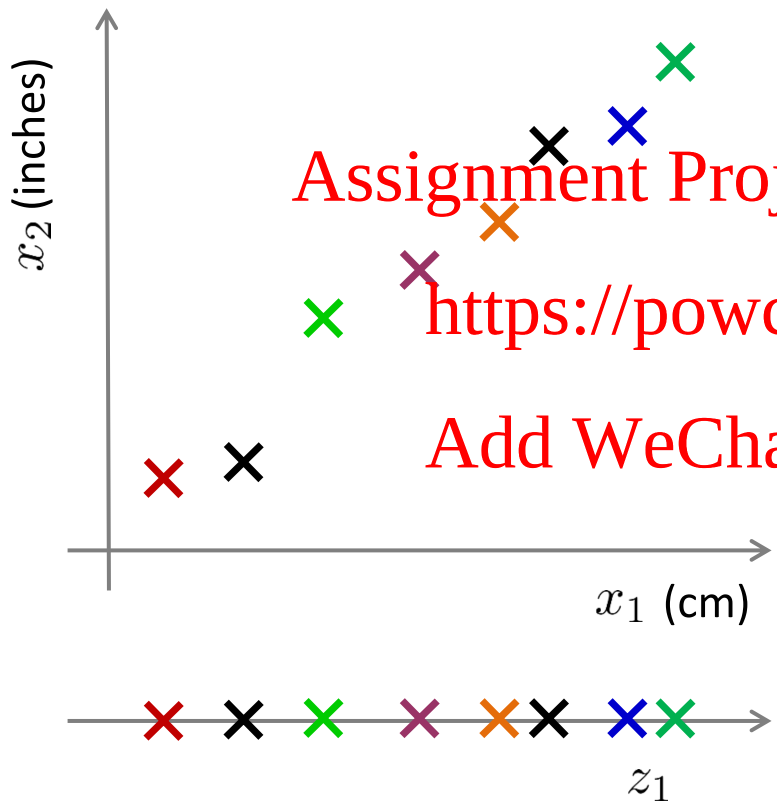
Data Compression



Reduce data from 2D to 1D

Vector quantization?

Data Compression: hidden dimension

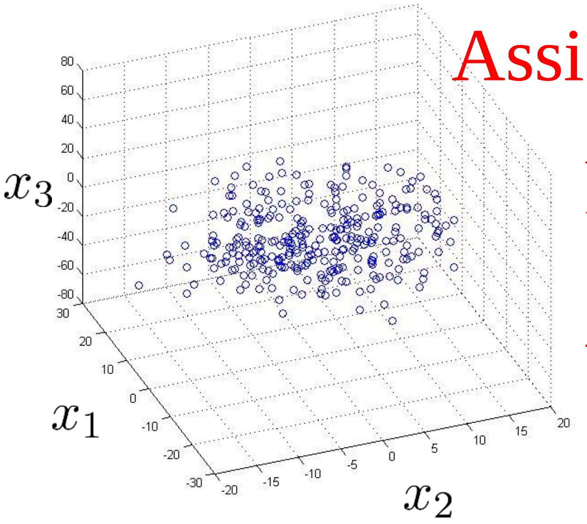


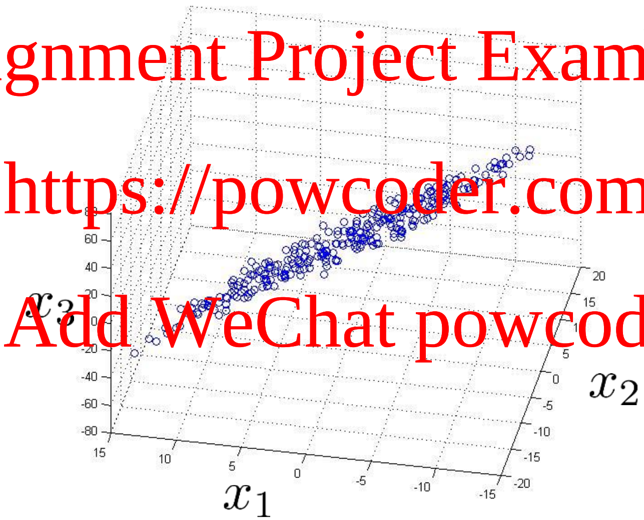
Reduce data from 2D to 1D

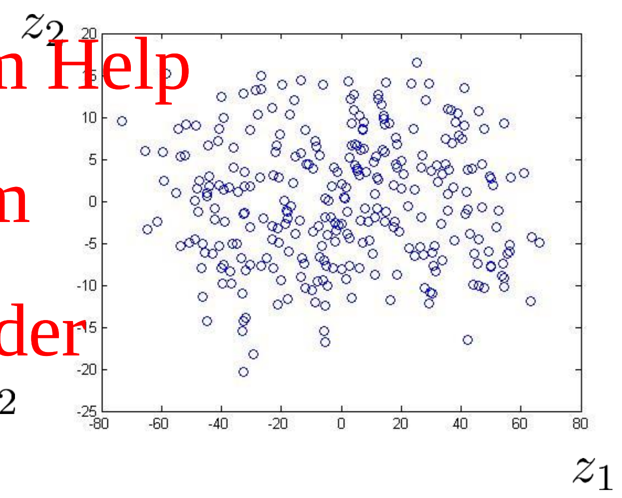
$$x^{(1)} \rightarrow z^{(1)}$$
$$x^{(2)} \rightarrow z^{(2)}$$
$$\vdots$$
$$x^{(m)} \rightarrow z^{(m)}$$

Data Compression

Reduce data from 3D to 2D







Data Visualization

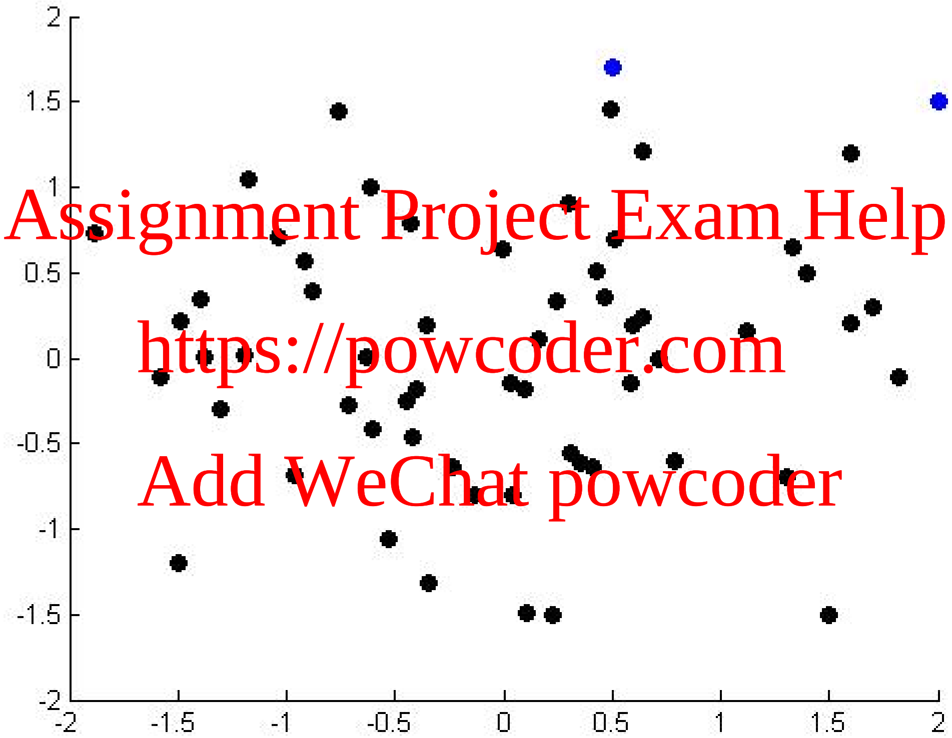
Country	GDP (trillions of US$)	Per capita GDP (thousands of intl. $)	Human Develop-ment Index	Life expectancy	Poverty Index (Gini as percentage)	Mean household income (thousands of US$)	…
Canada	1.577	39.17	0.908	80.7	32.6	67.293	…
China	5.878	7.54	0.687	73	46.9	10.22	…
India	1.632	3.41	0.547	64.7	36.8	0.735	…
Russia	1.48	19.84	0.755	65.5	39.9	0.72	…
Singapore	0.223	56.69	0.866	80	42.5	67.1	…
USA	14.527	46.86	0.91	78.3	40.8	84.3	…
…	…	…	…	…	…	…	

[resources from en.wikipedia.org]

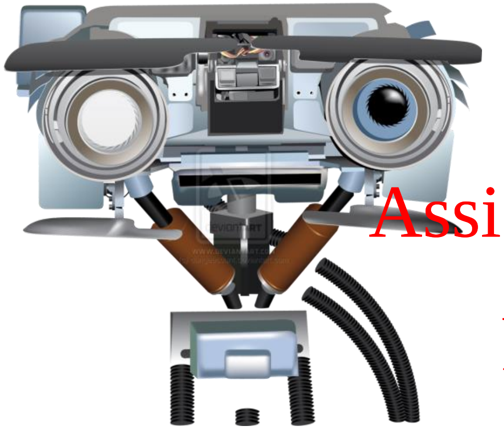
Data Visualization

Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
…	…	…

Data Visualization



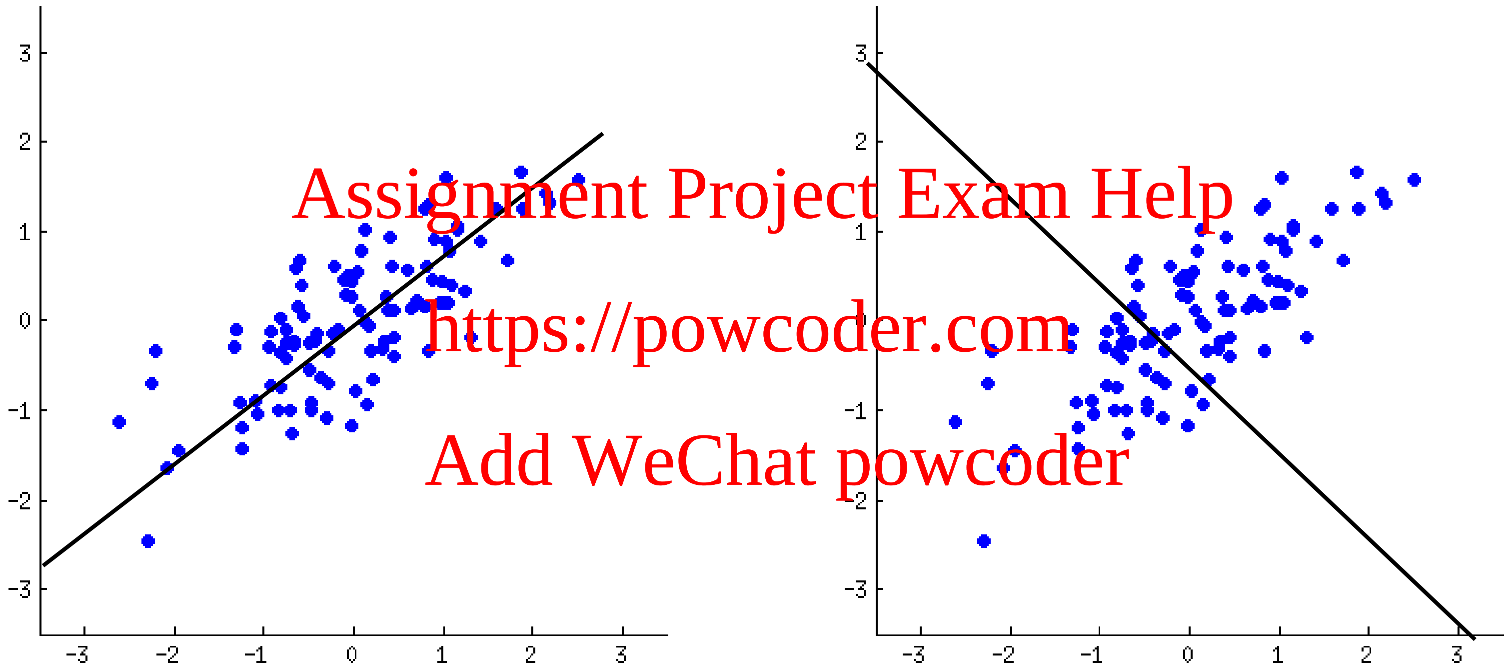
1



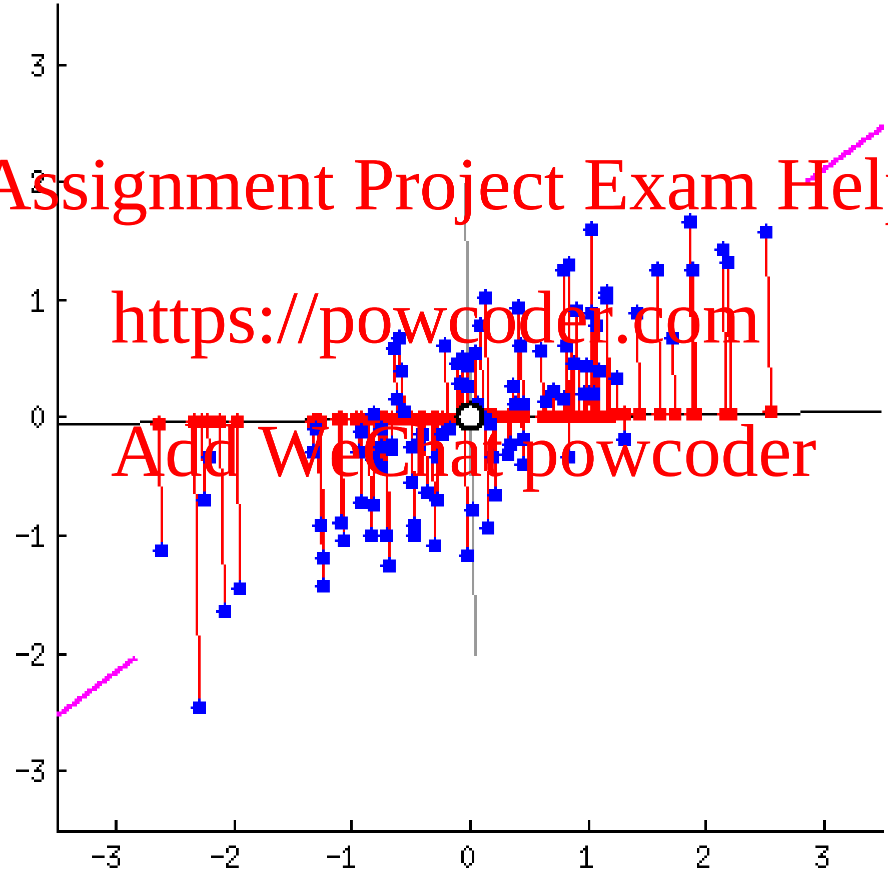
Unsupervised Learning II

Principal Component Analysis

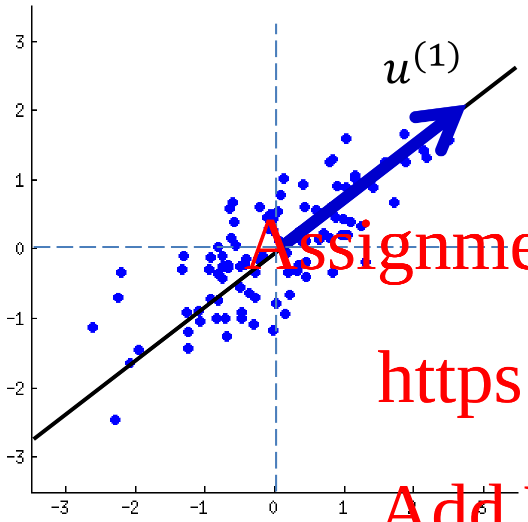
How to choose lower-dim subspace?



Minimize “error”

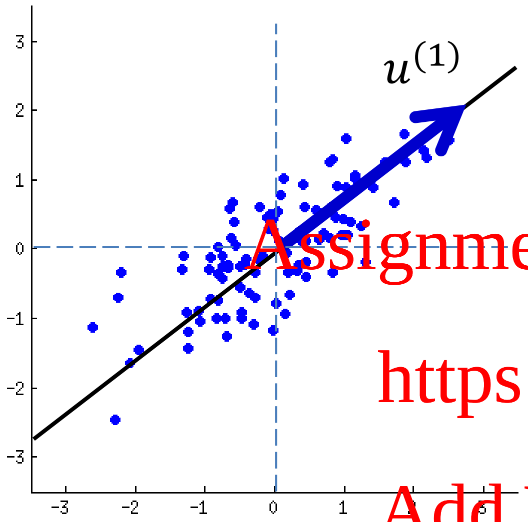


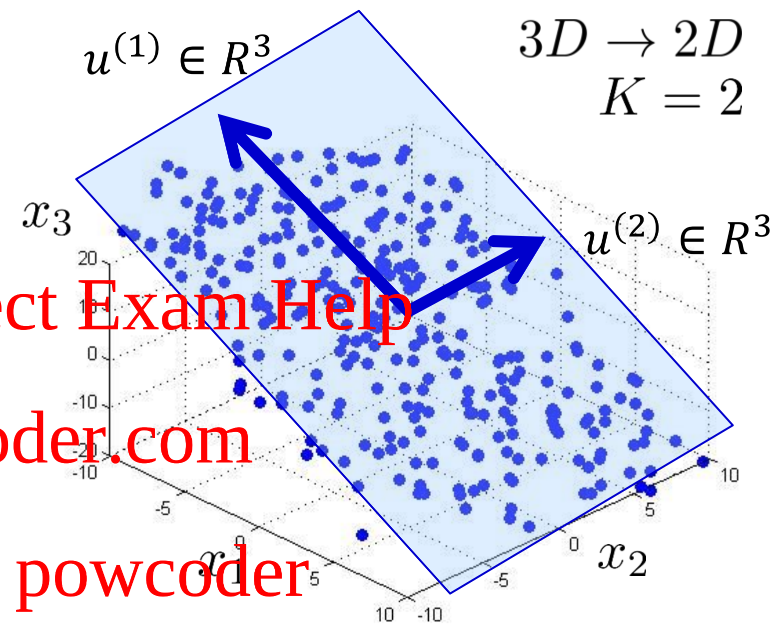
Choose subspace with minimal “information loss”



Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)}$) onto which to project the data, so as to minimize the projection error.

Choose subspace with minimal “information loss”

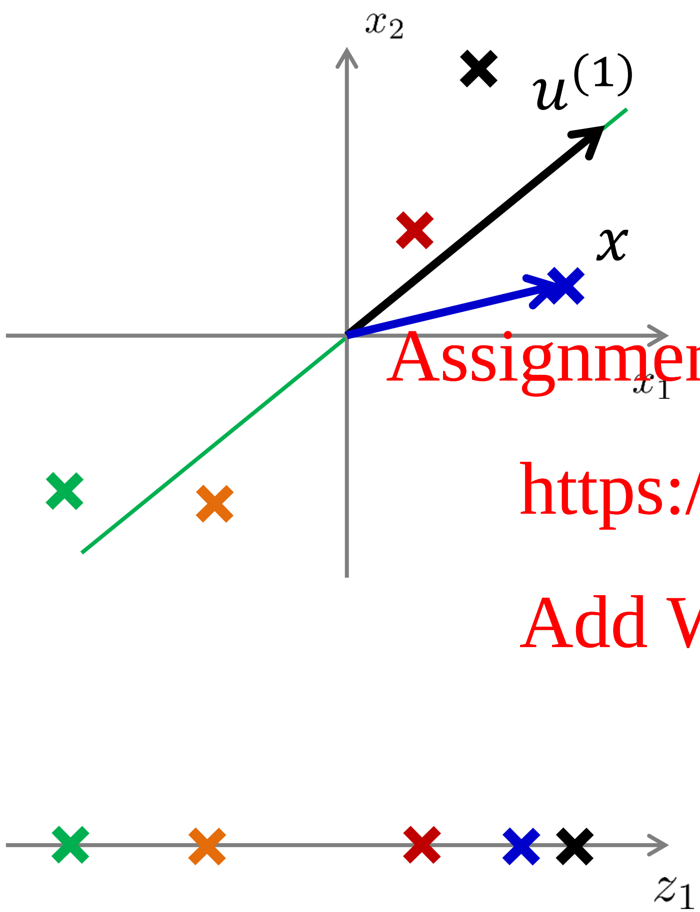




Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)}$) onto which to project the data, so as to minimize the projection error.

Reduce from n-dimension to K-dimension: Find K vectors $u^{(1)}, u^{(2)}, \dots, u^{(K)}$ onto which to project the data so as to minimize the projection error.

Principal Components Analysis

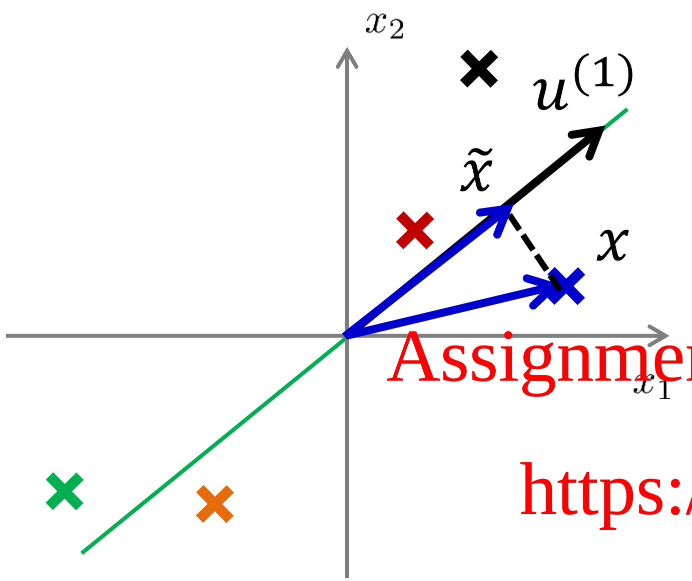


Find orthonormal basis vectors

$$\mathrm{U} = [u^{(1)} \quad \ldots \quad u^{(K)}], \text{ where } \mathrm{K} \ll \mathrm{n}$$

$$z = U^T x, \quad z_k = {u^{(k)}}^T x$$

Principal Components Analysis



Find orthonormal basis vectors

$$\mathrm{U} = [u^{(1)} \quad \ldots \quad u^{(K)}], \text{ where } \mathrm{K} \ll \mathrm{n}$$

$$z = U^T x, \quad z_k = {u^{(k)}}^T x$$

Reconstructed data point

$$\tilde{x} = \sum_{k=1}^{K} z_k u^{(k)}$$

Cost function: reconstruction error

$$J = \frac{1}{m} \sum_{i=1}^{m} \left\| \tilde{x}^i - x^i \right\|^2$$

Want: $\min_{U} J$

PCA Solution

- The solution turns out to be the first K eigenvectors of the data covariance matrix (see Bishop 12.1 for details)

- Closed-form, use Singular Value Decomposition (SVD) on covariance matrix

- Other PCA formulations
  - can derive via maximizing variance of projected data
  - probabilistic formulation of PCA possible, or the similar *factor analysis*, see Bishop 8.1.4

PCA Algorithm

Normalize features (ensure every feature has zero mean) and optionally scale feature

Compute "covariance matrix" Σ:

Sigma = $$\frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T$$

Compute its "eigenvectors":


```
[U,S,V] = svd(Sigma);
```

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Keep first K eigenvectors and project to get new features **z**

```
Ureduce = U(:,1:K);
z = Ureduce'*x;
```

PCA Algorithm

Data preprocessing

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

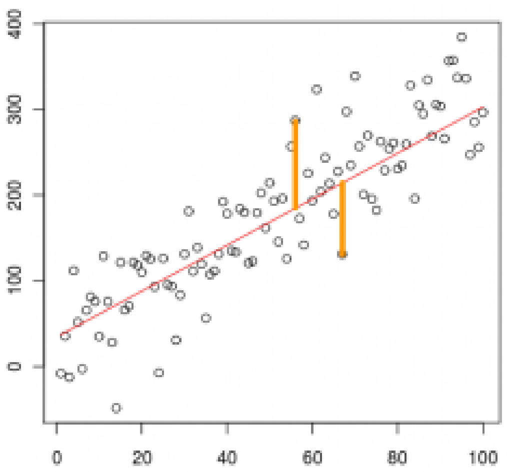
Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

If different features on different scales (e.g., $x_1 =$ size of house, $x_2 =$ number of bedrooms), scale features to have comparable range of values.

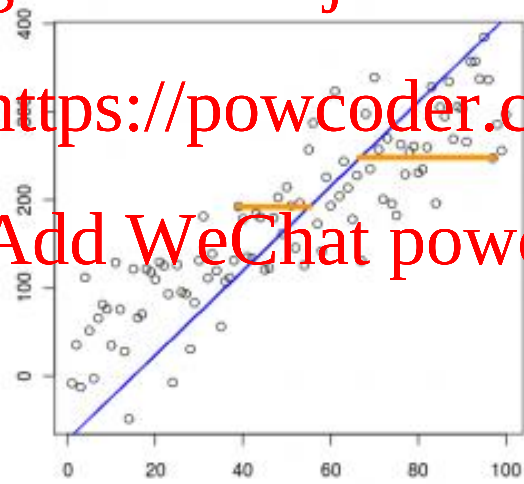
PCA is not linear regression

- There is no “output” in PCA, all dimensions are equal

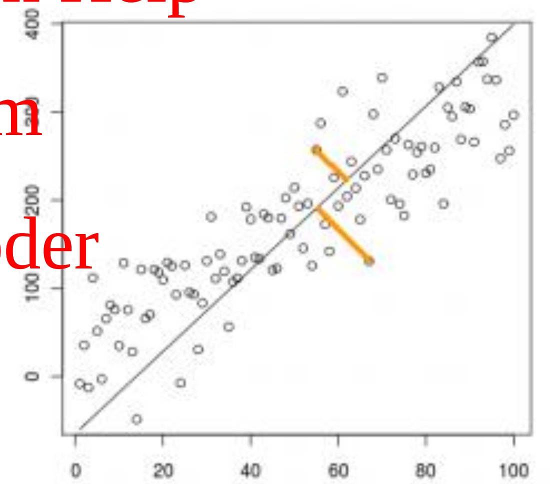
Regression $x_1 \rightarrow x_2$



Regression $x_2 \rightarrow x_1$



PCA



Choosing k (number of principal components)

Average squared projection error:

Total variation in the data:

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01 \qquad (1\%)$$

"99% of variance is retained"

Choosing k (number of principal components)

```
[U,S,V] = svd(Sigma)
```

Pick smallest value of k for which

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{m} S_{ii}} \geq 0.99$$

(99% of variance retained)

Good use of PCA

- Compression
  - Reduce memory/disk needed to store data
  - Speed up learning algorithm

- Visualization

Bad use of PCA: To prevent overfitting

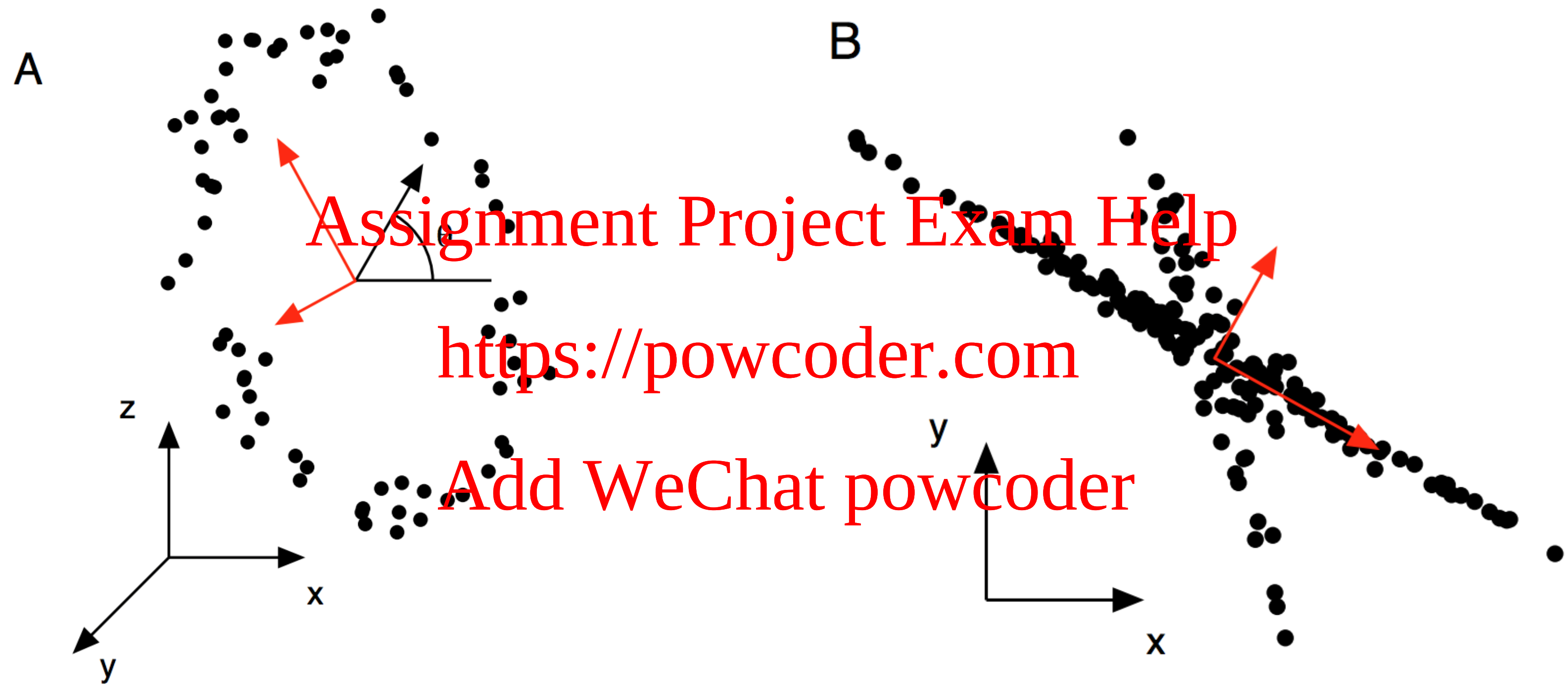
Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$.

Thus, fewer features, less likely to overfit.

This might work OK, but isn't a good way to address overfitting. Use regularization instead.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

When does PCA fail?



(a) Tracking a person on a ferris wheel (black dots). All dynamics can be described by the phase of the wheel θ, a non-linear combination of the naïve basis.

(b) Non-Gaussian distributed data and non-orthogonal axes cause PCA to fail. The axes with the largest variance do not correspond to the appropriate answer.

Shlens, A Tutorial on Principal Component Analysis

Next Class

Neural Networks I: Feed-forward Nets:

artificial neuron, MLP, sigmoid units; neuroscience inspiration; output vs hidden layers; linear vs nonlinear networks; feed-forward neural networks

Reading: Bishop 5.1-5.3