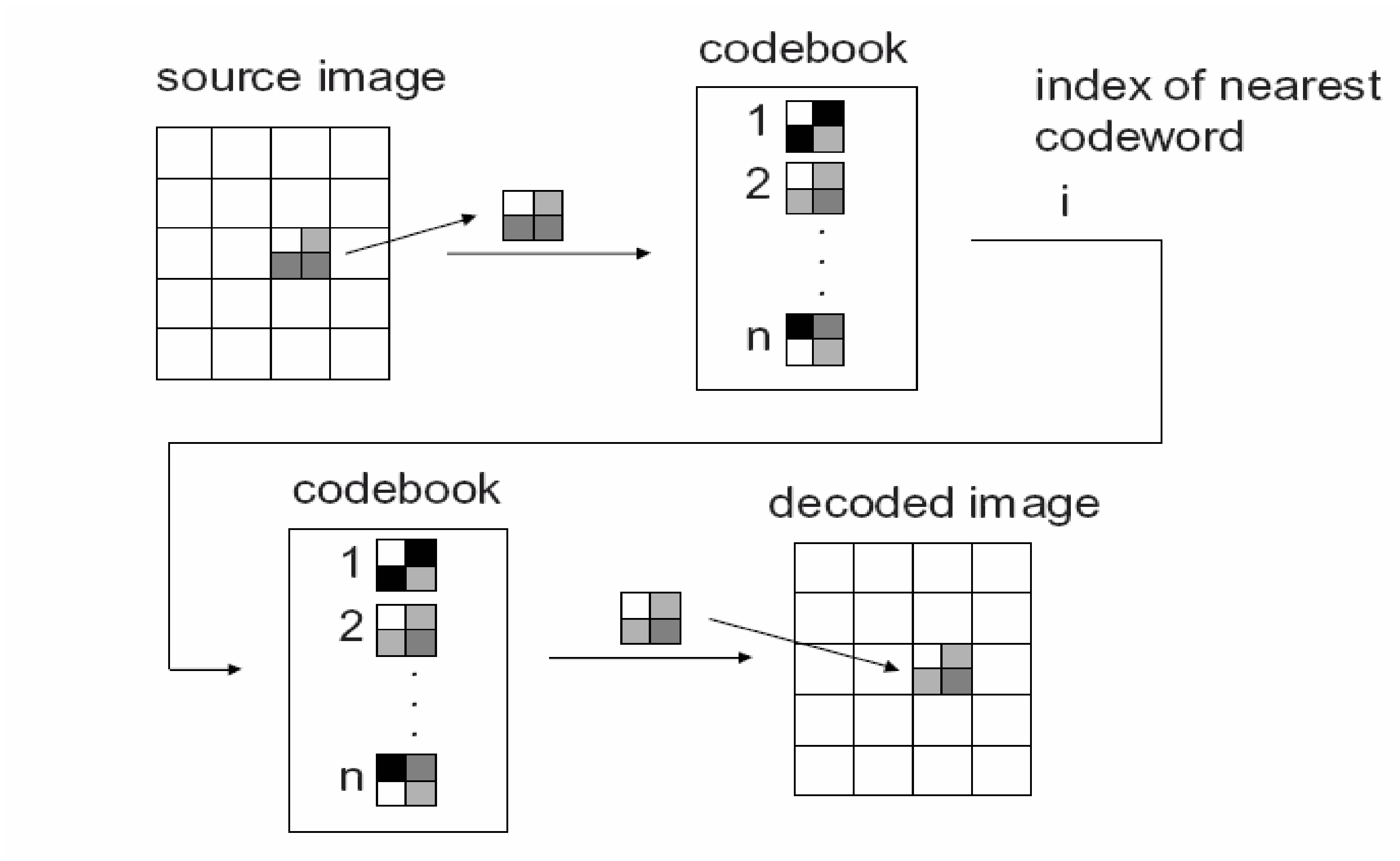
Lecture 13: Vector Quantization

Thinh Nguyen
Oregon State University

Vector Quantization



Vector

- An *a* x *b* block can be considered to be a vector of dimension *ab*.

$$\text{block} \quad \begin{array}{|c|c|} \hline w & x \\ \hline y & z \\ \hline \end{array} \quad = \quad (w,x,y,z) \text{ vector}$$

- Nearest means in terms of Euclidian distance or Euclidian squared distance. Both are equivalent.

$$\text{Distance} = \sqrt{(w_1 - w_2)^2 + (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

$$\text{Squared Distance} = (w_1 - w_2)^2 + (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2$$

- Squared distance is easier to calculate.

Vector Quantization Facts

- The image is partitioned into *a* x *b* blocks.
- The codebook has *n* representative *a* x *b* blocks called codewords, each with an index.
- Compression with fixed length codes is

$$\frac{\log_2 n}{ab} \text{ bpp}$$

- Example: *a* = *b* = 4 and *n* = 1,024
  - compression is 10/16 = .63 bpp
  - compression ratio is 8 : .63 = 12.8 : 1
- Better compression can be achieved with entropy coding of indices.

Examples

4 x 4 blocks
.63 bpp

4 x 8 blocks
.31 bpp

8 x 8 blocks
.16 bpp

Codebook size = 1,024

Scalar vs. Vector Quantization

- Pixels within a block are correlated.
  - This tends to minimize the number of codewords needed to represent the vectors well.
- More flexibility:
  - Different size blocks.
  - Different size codebooks.

Scalar vs. Vector Quantization

- Encoding:
  - Scan the *a* x *b* blocks of the image. For eachblock, find the nearest codeword in the codebook and output its index.
  - This is a “Nearest neighbor search.”
- Decoding:
  - For each index, output the codeword with that index into the destination image.
  - This is a table lookup.

The Codebook

- Both encoder and decoder must have the same codebook.
- The codebook must be useful for many images and be stored somewhere.
- The codebook must be designed properly to be effective.
- Design requires a representative training set.
- *These are major drawbacks to VQ.*

The Codebook

- Both encoder and decoder must have the same codebook.
- The codebook must be useful for many images and be stored somewhere.
- The codebook must be designed properly to be effective.
- Design requires a representative training set.
- *These are major drawbacks to VQ.*

The Codebook Design Problem

- Input: A training set *X* of vectors of dimension *d* and a number *n*. (*d* = *a* x *b* and *n* is the number of codewords)
- Ouput: *n* codewords *c(0)*, *c(1)*, ... , *c(n-1)* that minimize the distortion:

$$D = \sum_{x \in X} \| x - c(index(x)) \|^2 \quad \text{sum of squared distances}$$

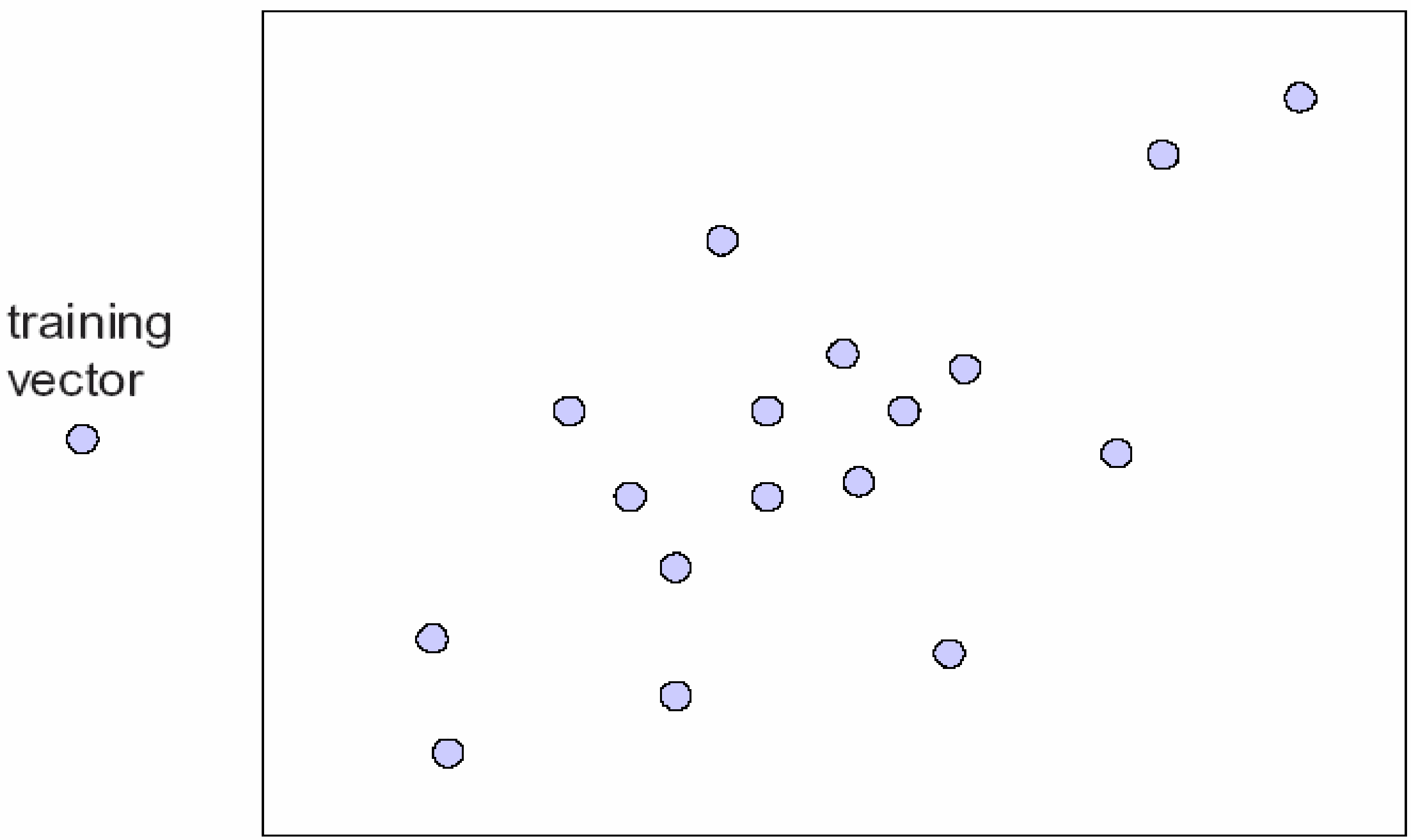
where *index(x)* is the index of the nearest codeword to *x*.

$$\|(x_0, x_1 \cdots, x_{d-1})\|^2 = x_0^{\,2} + x_1^{\,2} + \cdots + x_{d-1}^{\,2} \quad \text{squared norm}$$

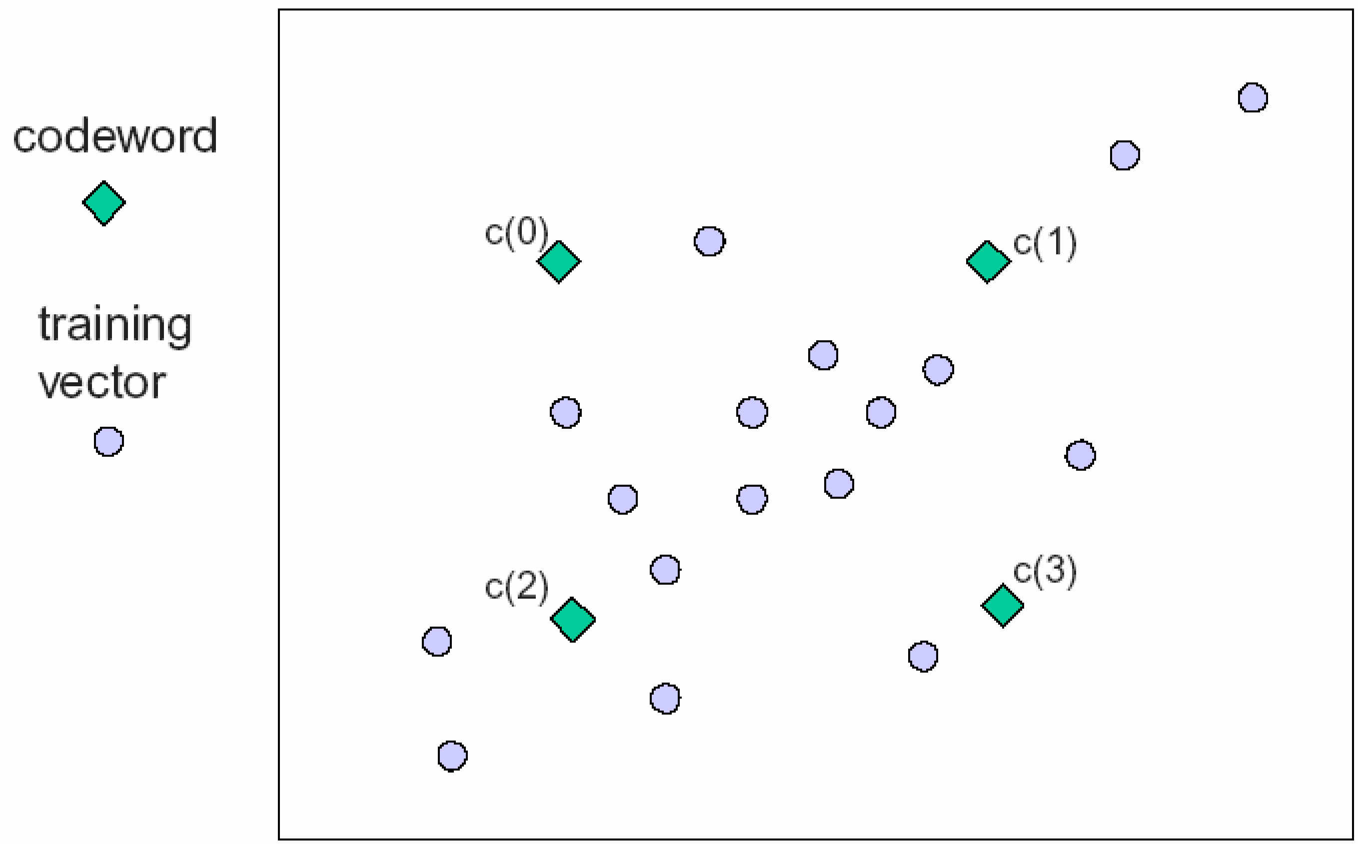
Generalized Lloyd Algorithm (GLA)

- The Generalized Lloyd Algorithm (GLA) extends the Lloyd algorithm for scalars.
  - Also called LBG after inventors Linde, Buzo, Gray (1980).
  - It can be very slow for large training sets.

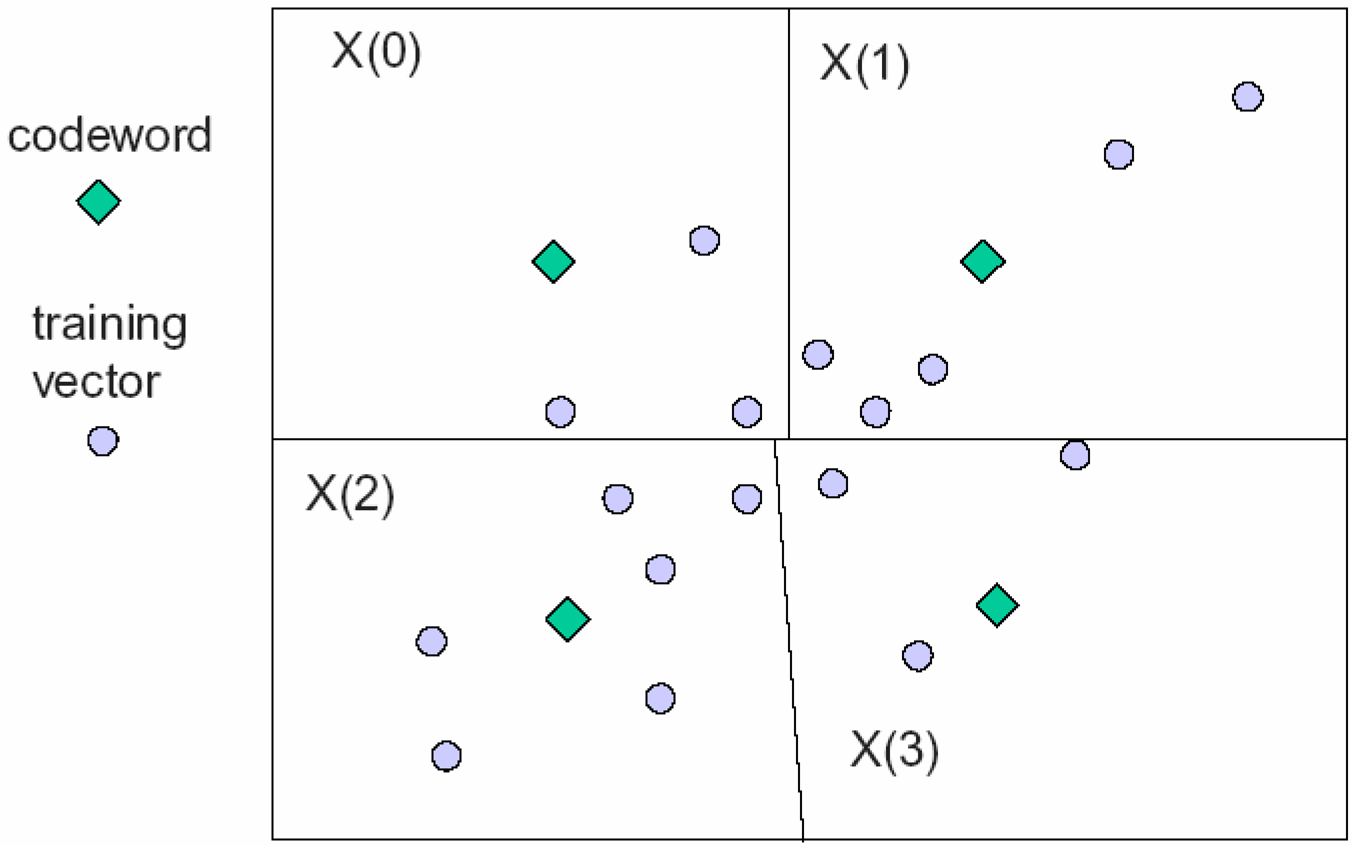
Example



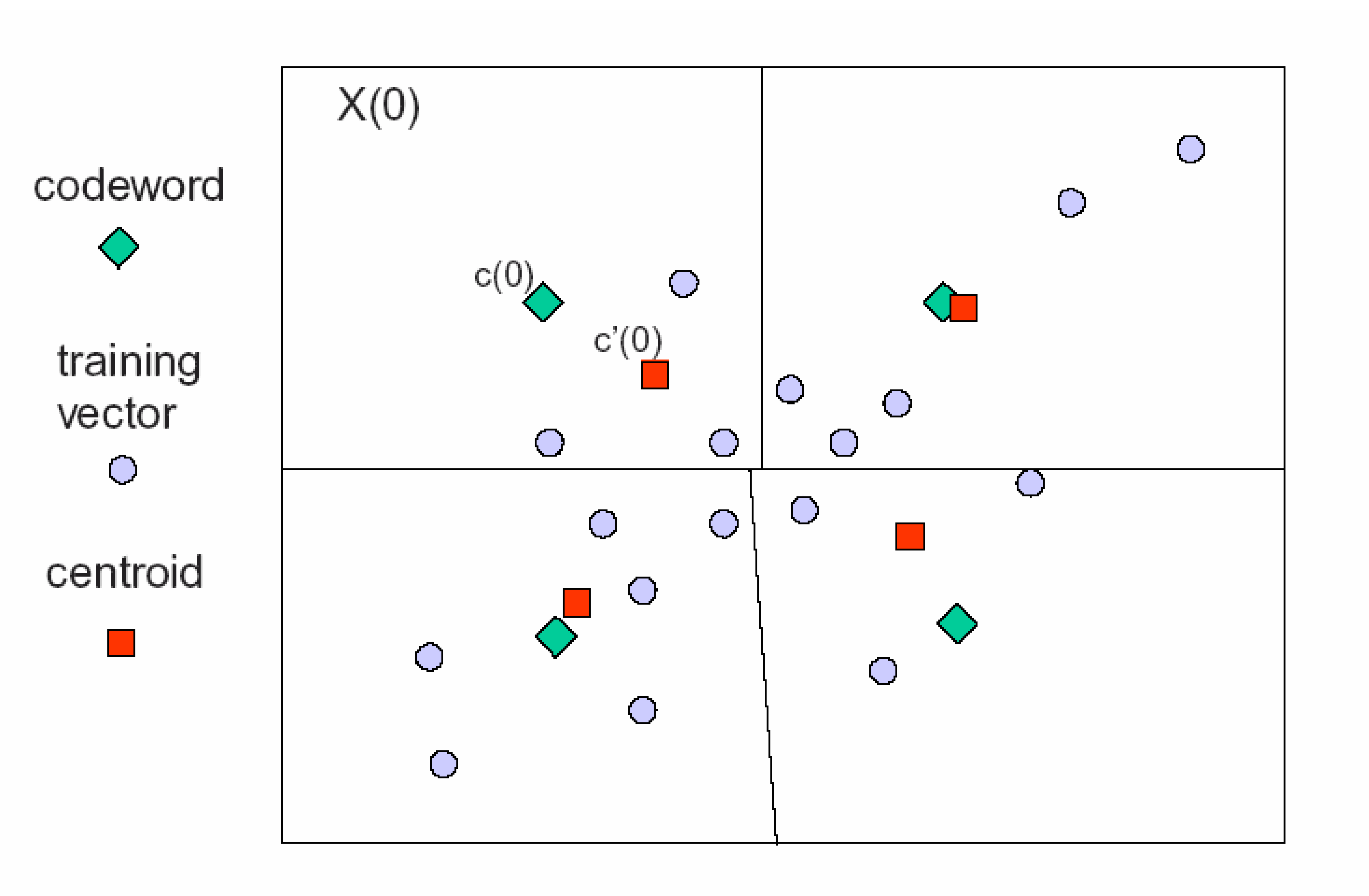
Example



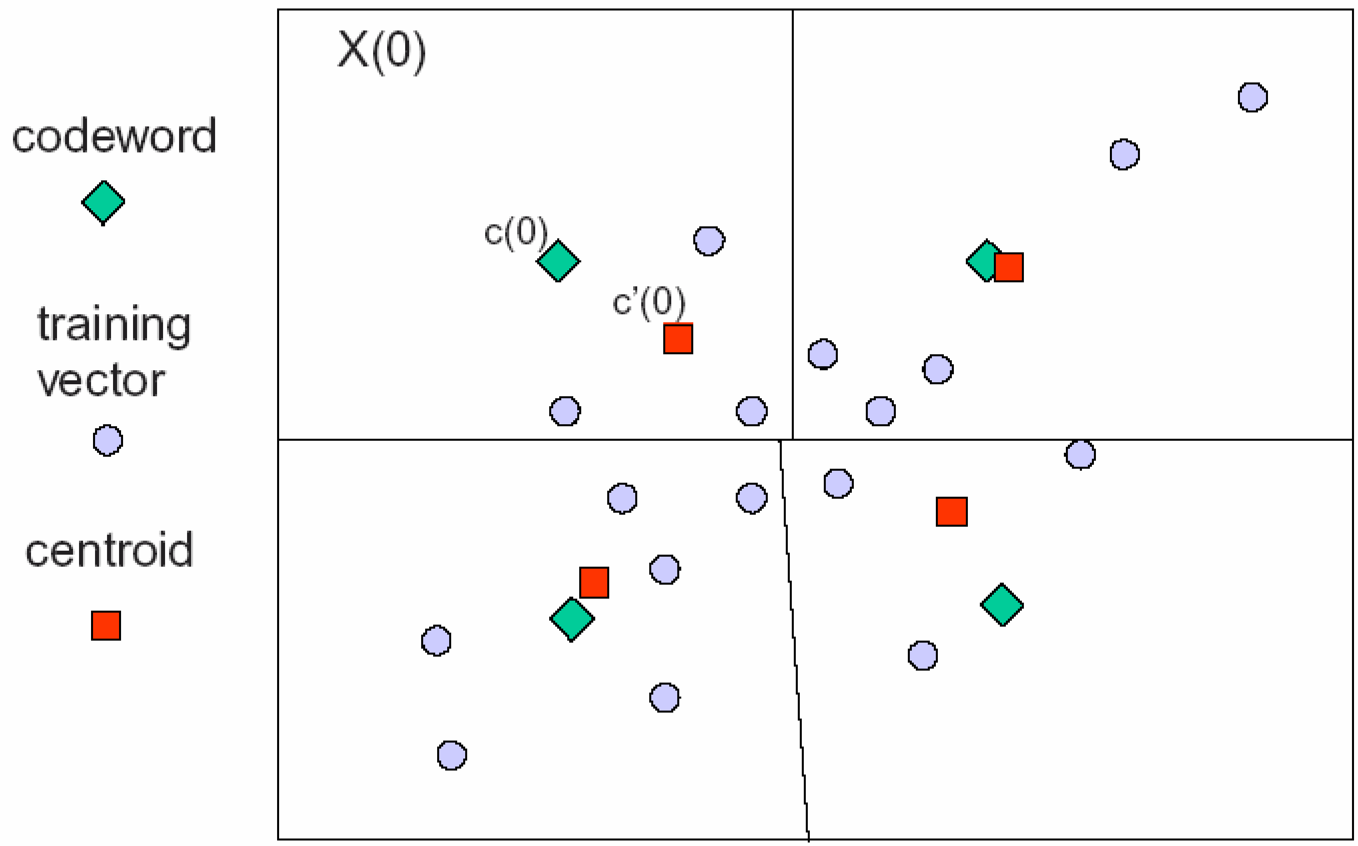
Example



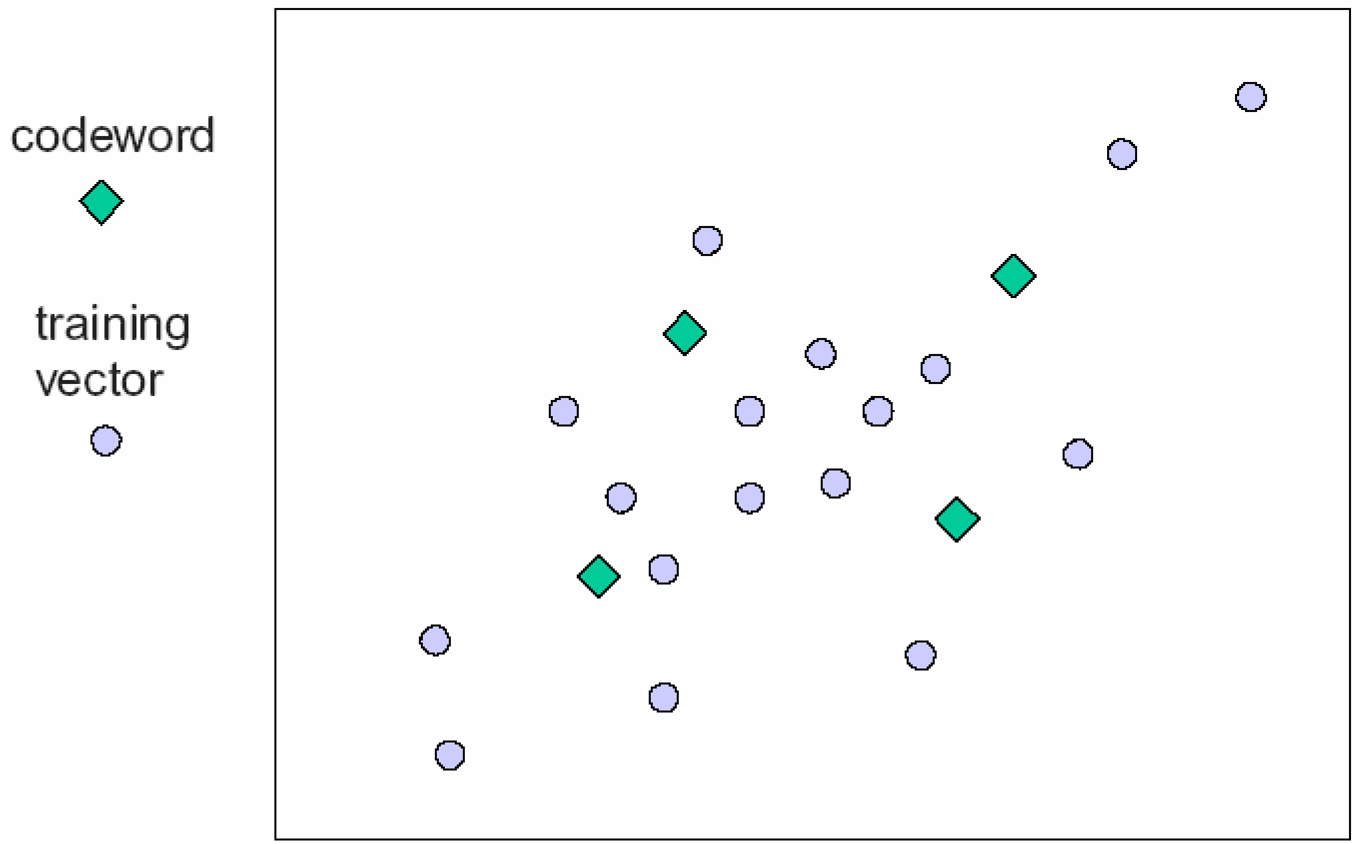
Example



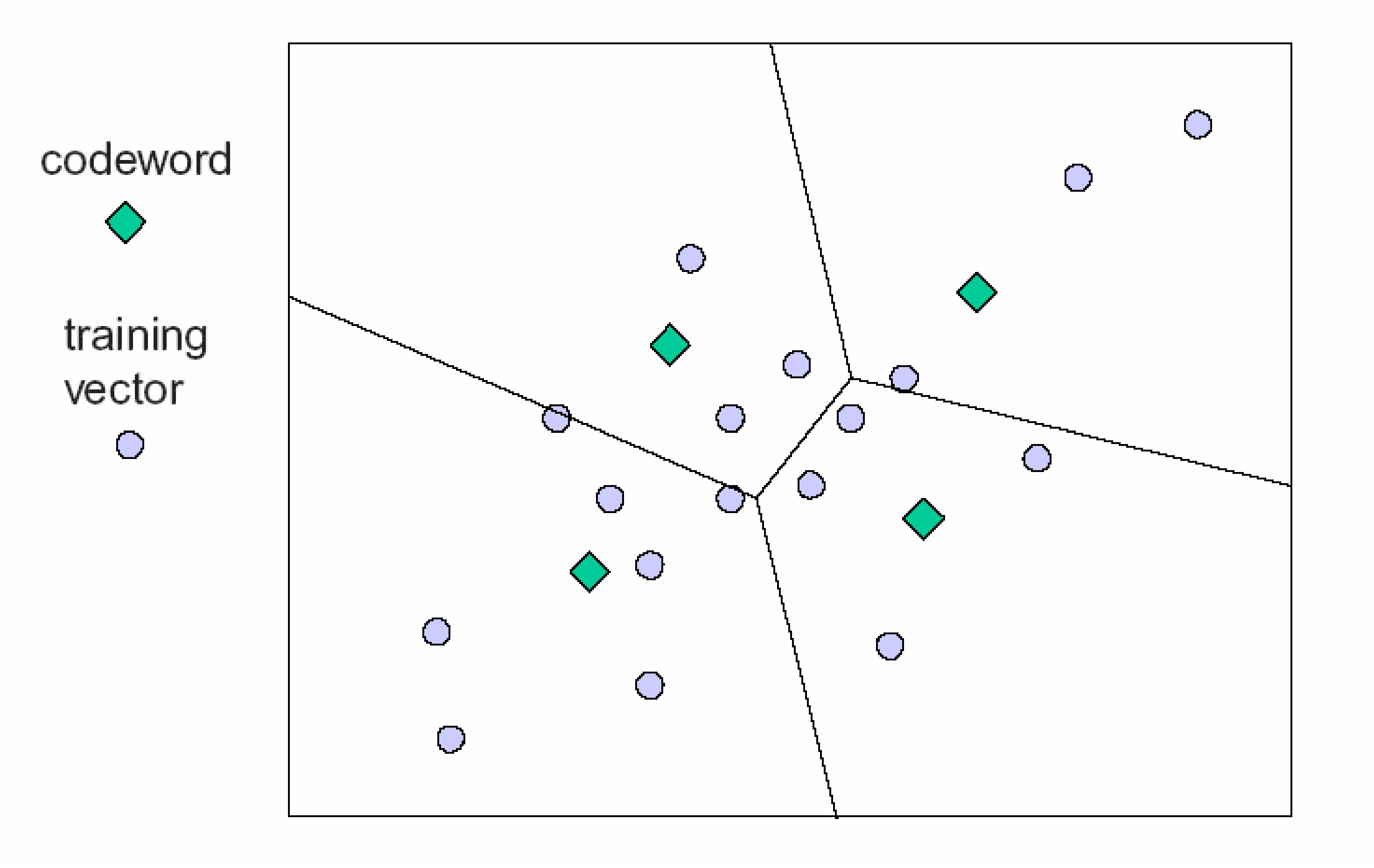
Example



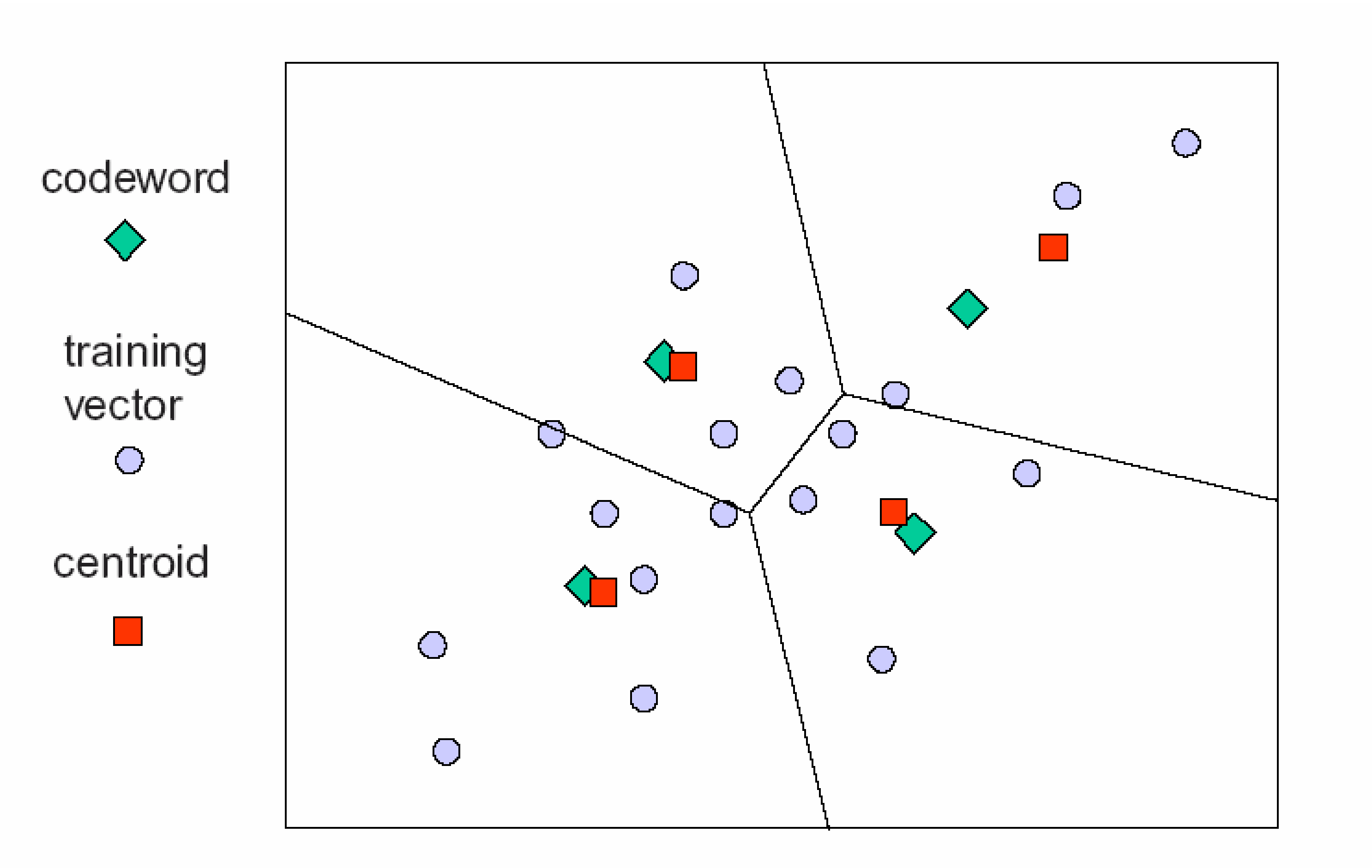
Example



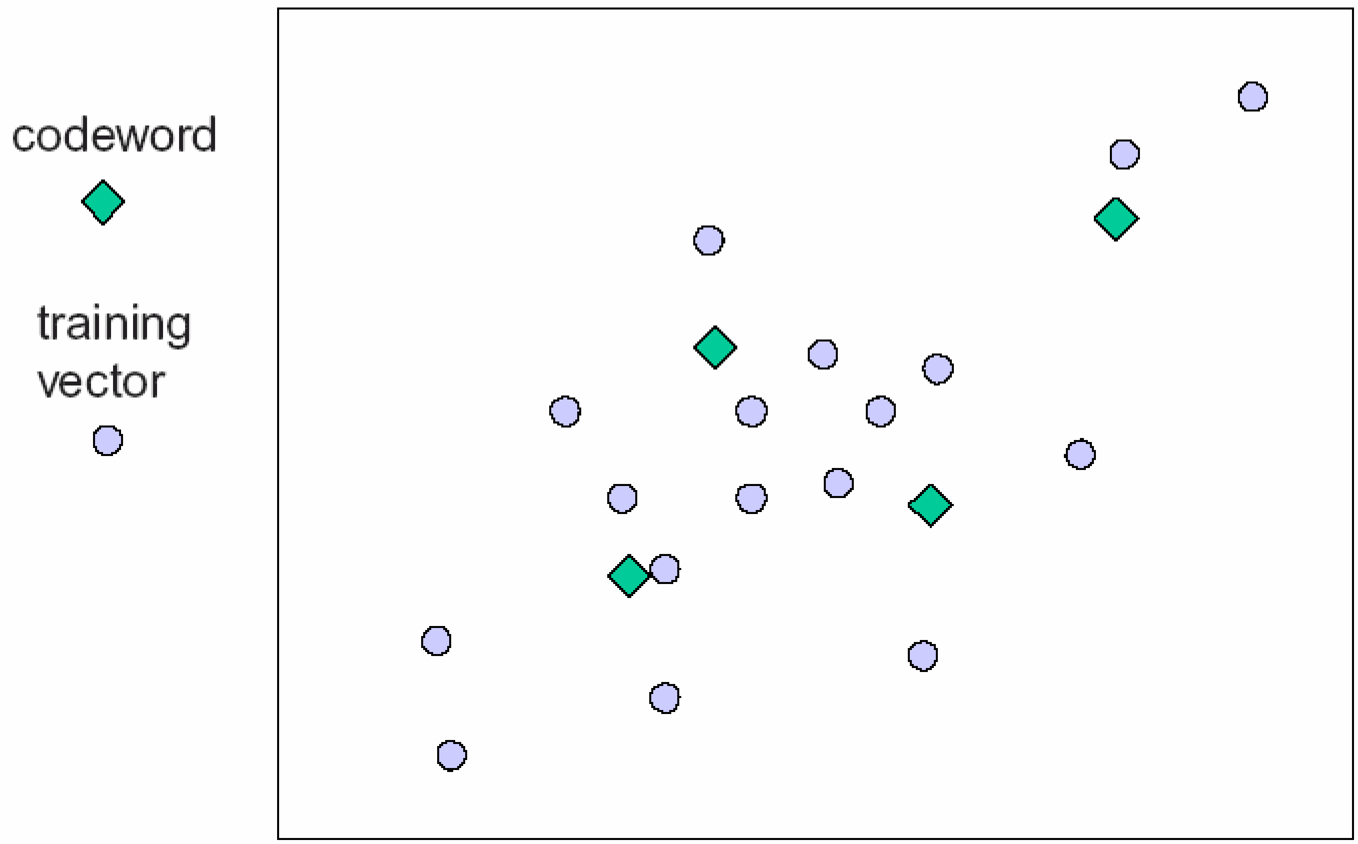
Example



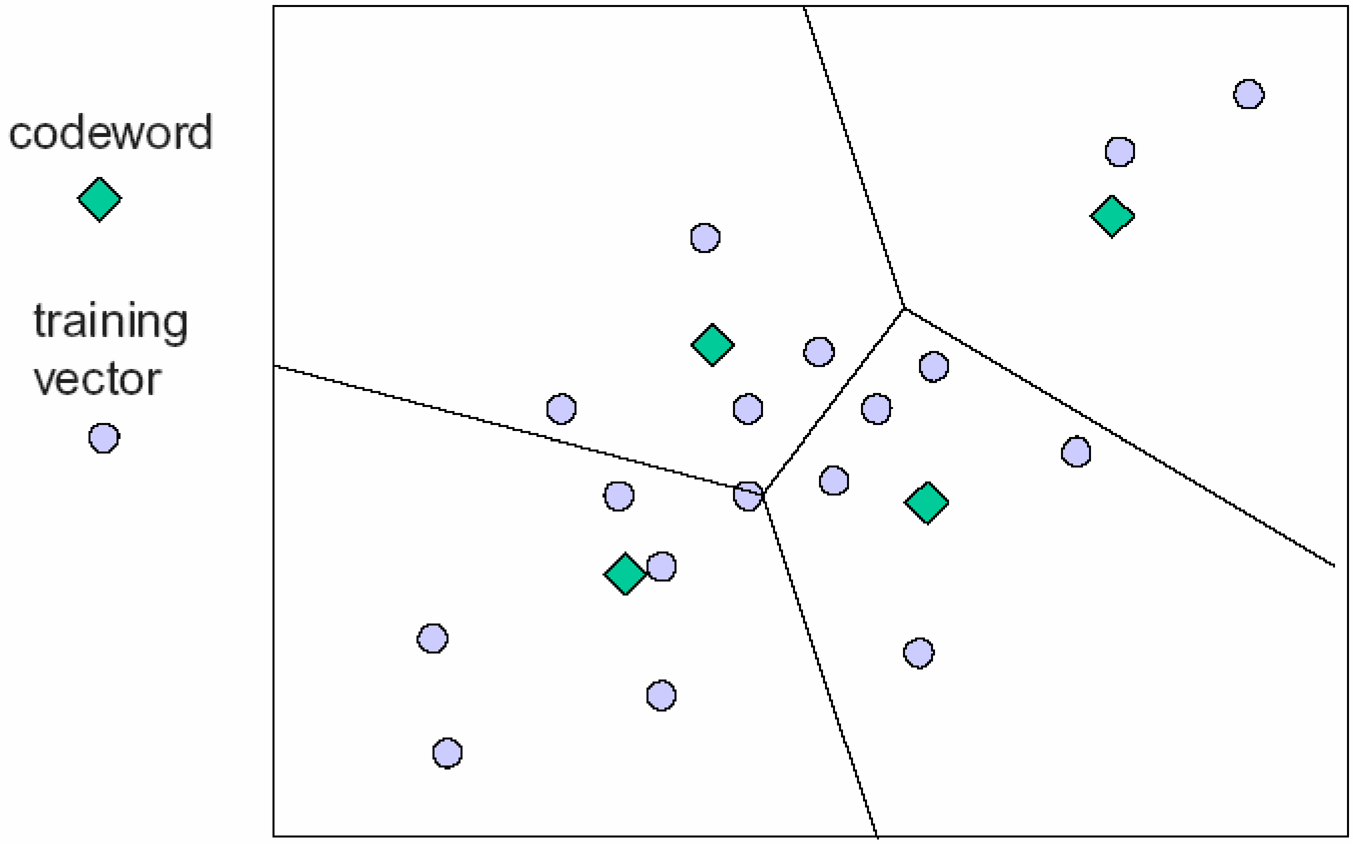
Example



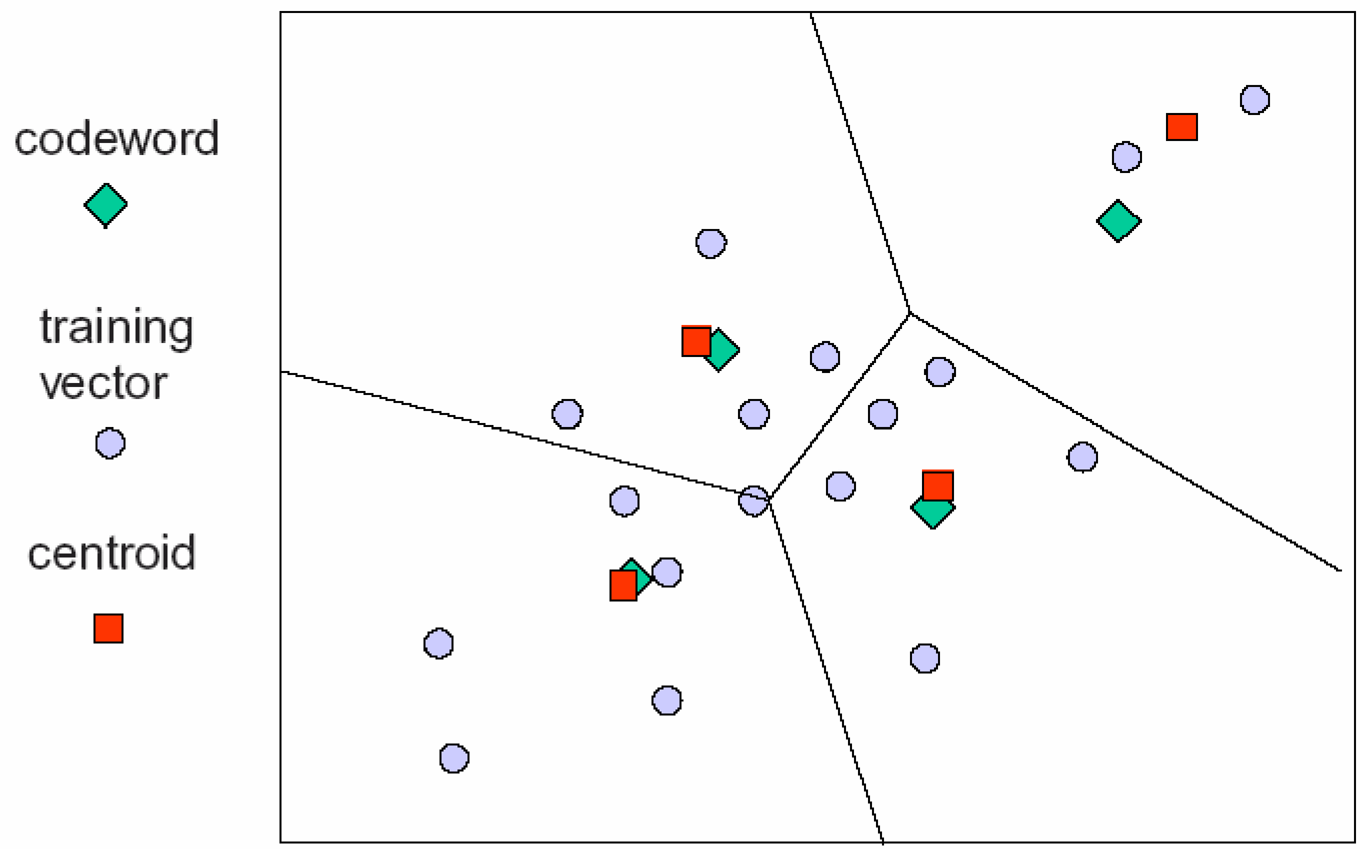
Example



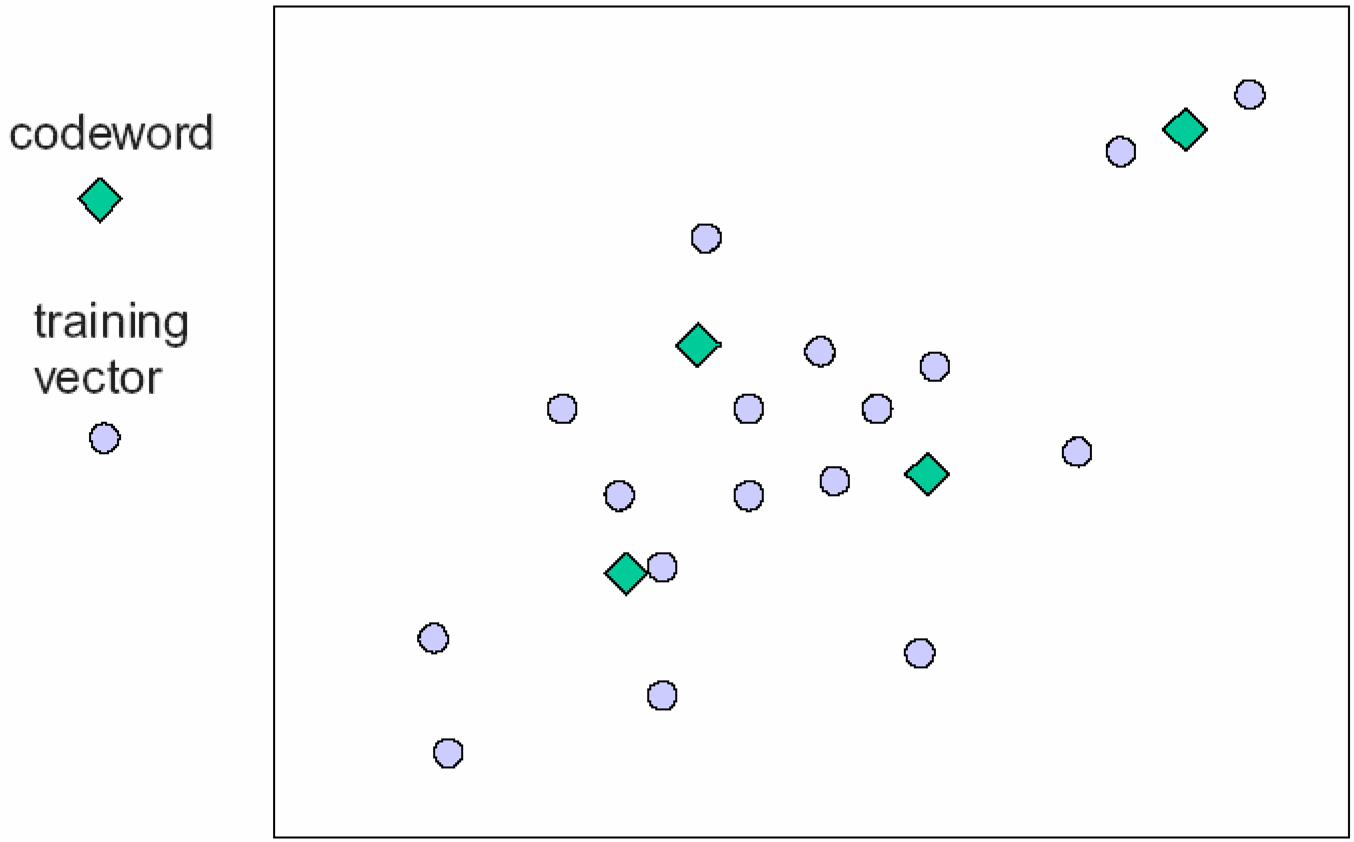
Example



Example



Example



GLA Algorithm

Choose a training set **X** and small error tolerance $\varepsilon > 0$.
Choose start codewords c(0),c(1),...,c(n-1).
Compute X(j) := {x : x is the vector in **X** closest to c(j)}.
Compute distortion D for c(0),c(1),...,c(n-1).
Repeat
  Compute new codewords:

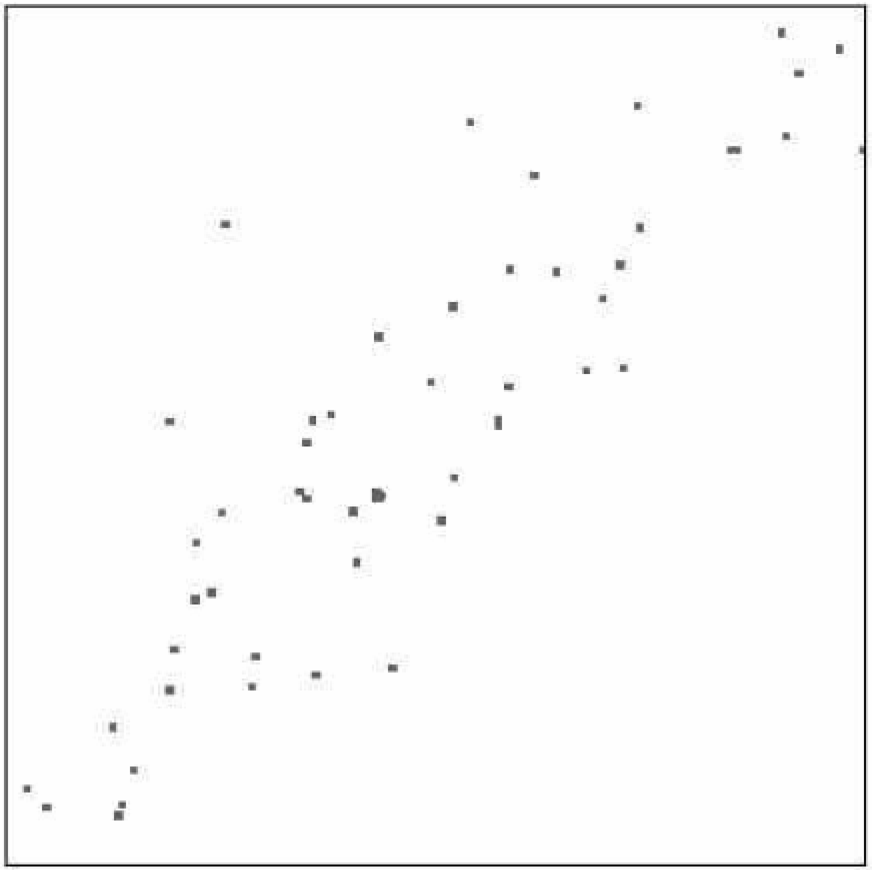
$$c'(j) := \mathrm{round}\left(\frac{1}{|X(j)|}\sum_{x \in X(j)} x\right) \qquad \text{(centroid)}$$

  Compute X'(j) = {x : x is the vector in **X** closest to c'(j)}.
  Compute distortion D' for c'(0),c'(1),...,c'(n-1).
  if $|(D - D')/D| < \varepsilon$ then quit,
  else c := c'; X := X', D := D'.
End{repeat}

Codebook

1 x 2 codewords

Note: codewords diagonally spread



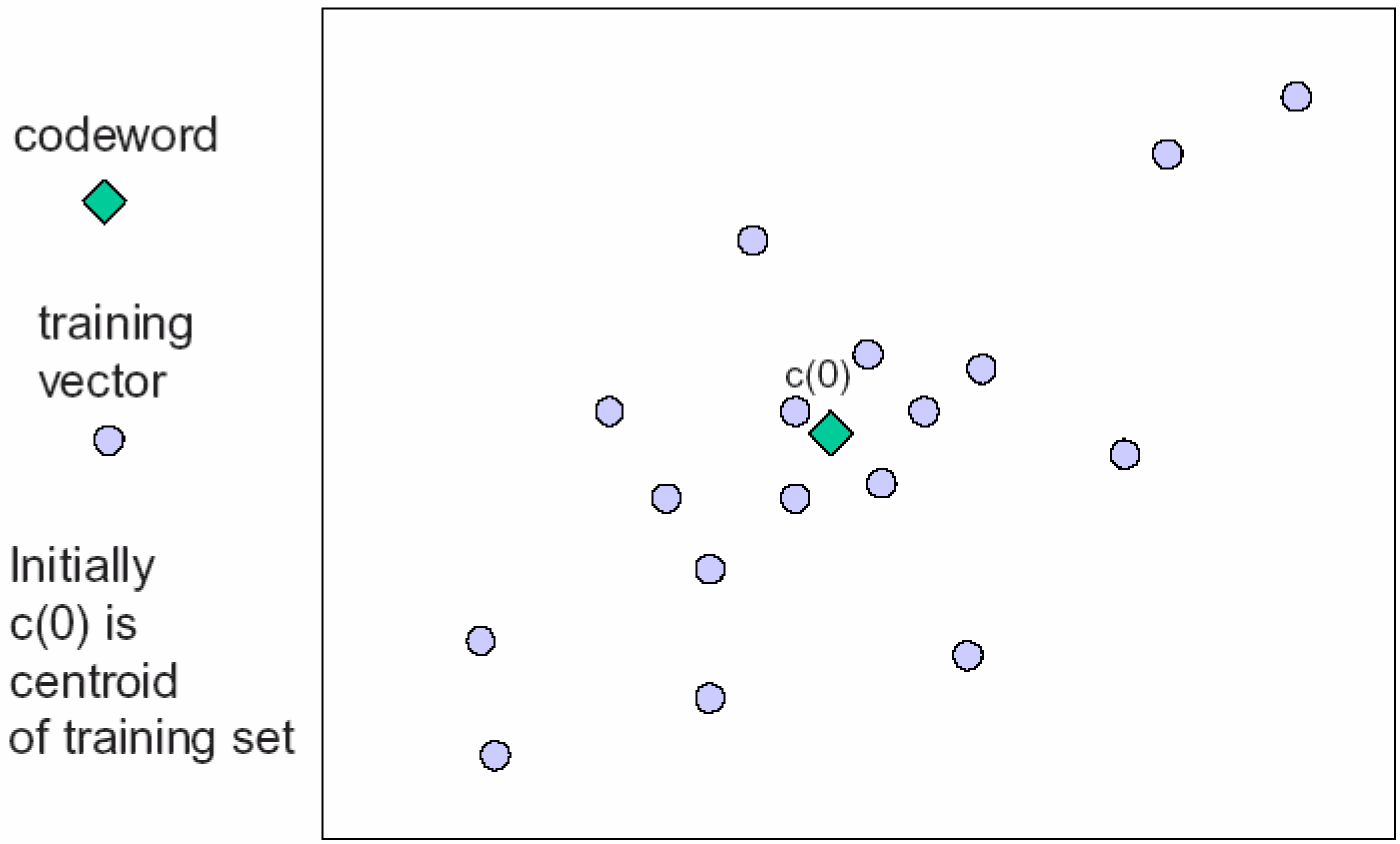
Codeword Splitting

- It is possible that a chosen codeword represents no training vectors, that is, X(j) is empty.
  - Splitting is an alternative codebook design algorithm that avoids this problem.
- Basic Idea
  - Select codeword c(j) with the greatest distortion.

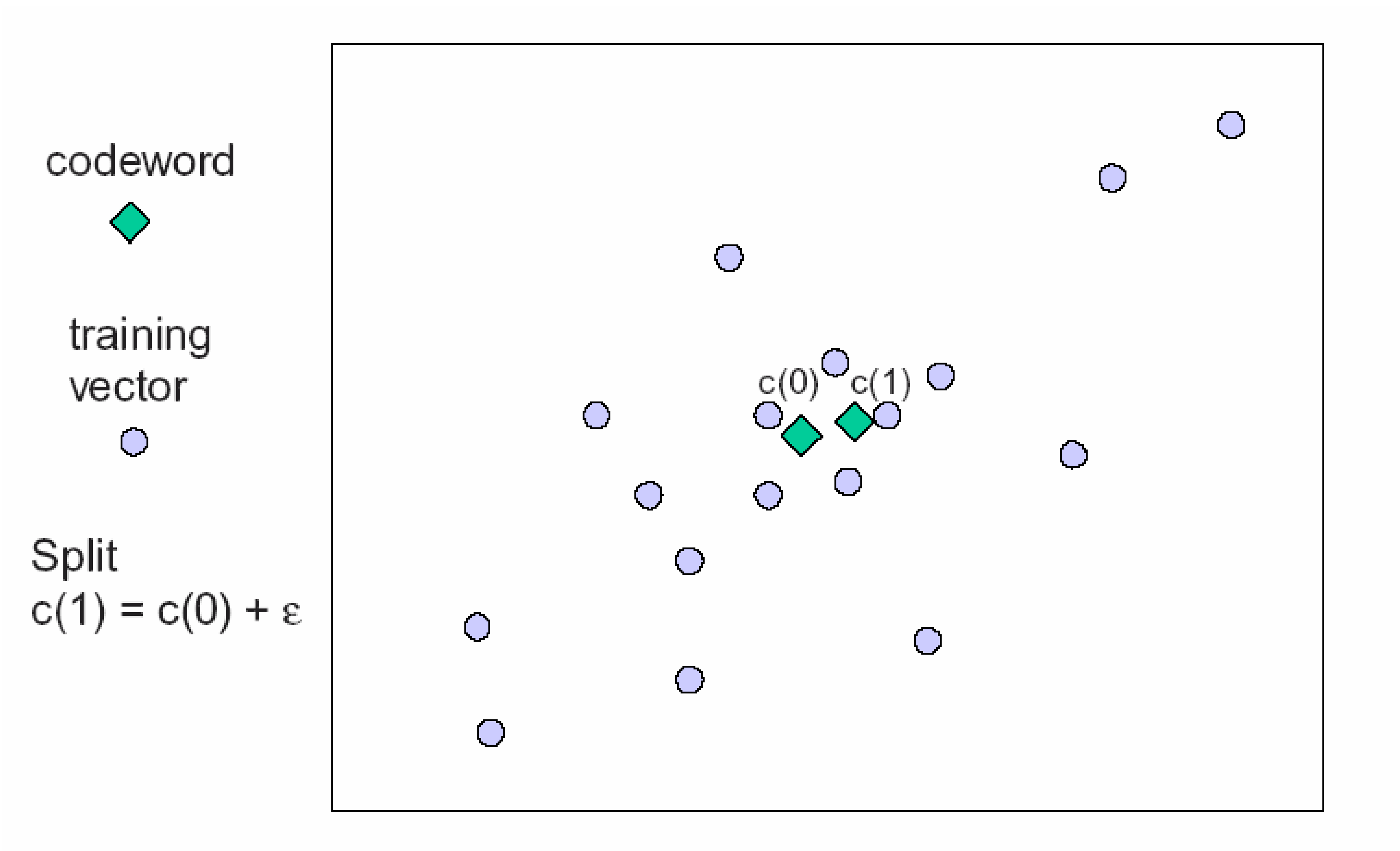
$$D(j) = \sum_{x \in X(j)} \| x - c(j) \|^2$$

  - Split it into two codewords then do the GLA.

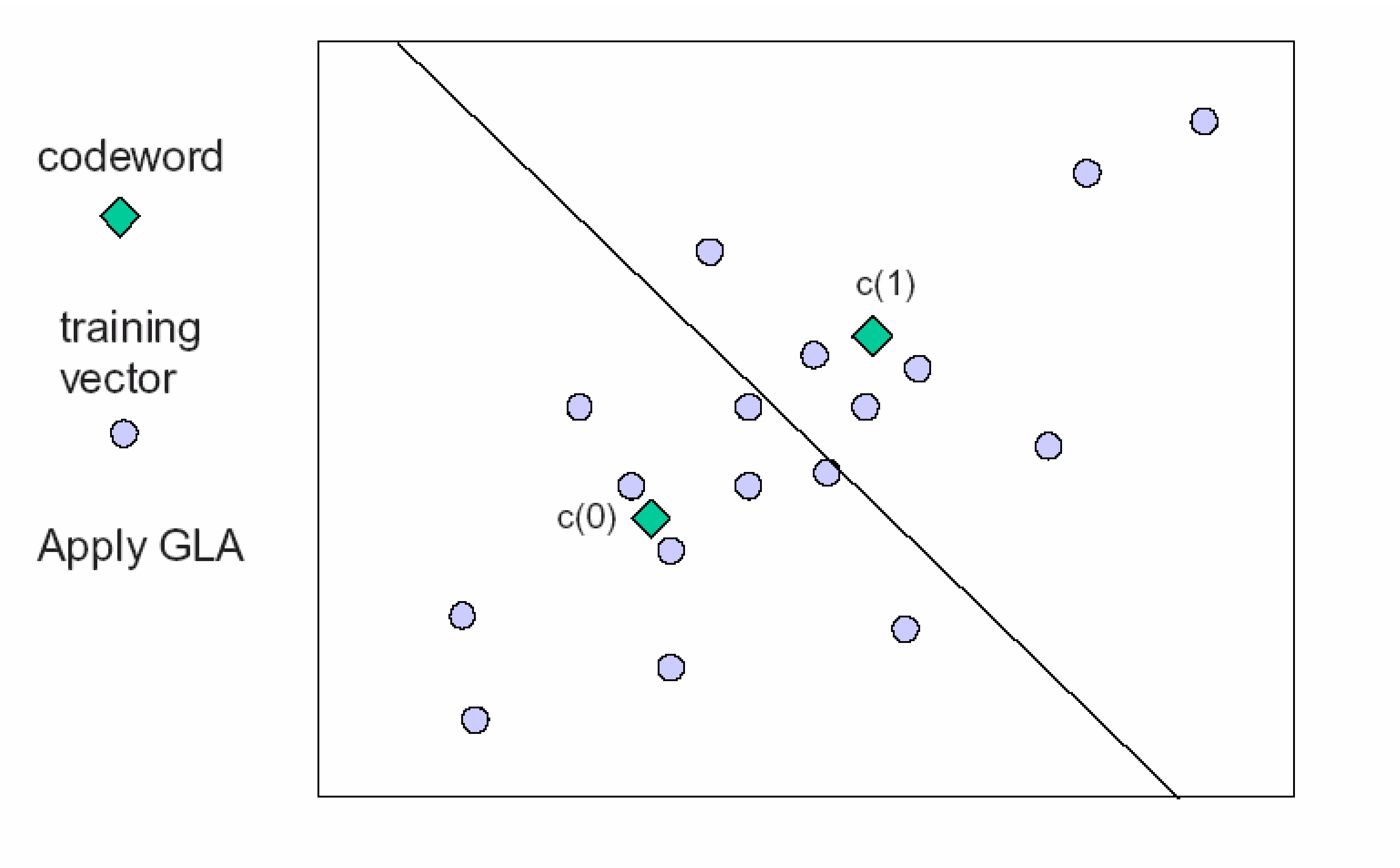
Example of Splitting



Example of Splitting



Example of Splitting

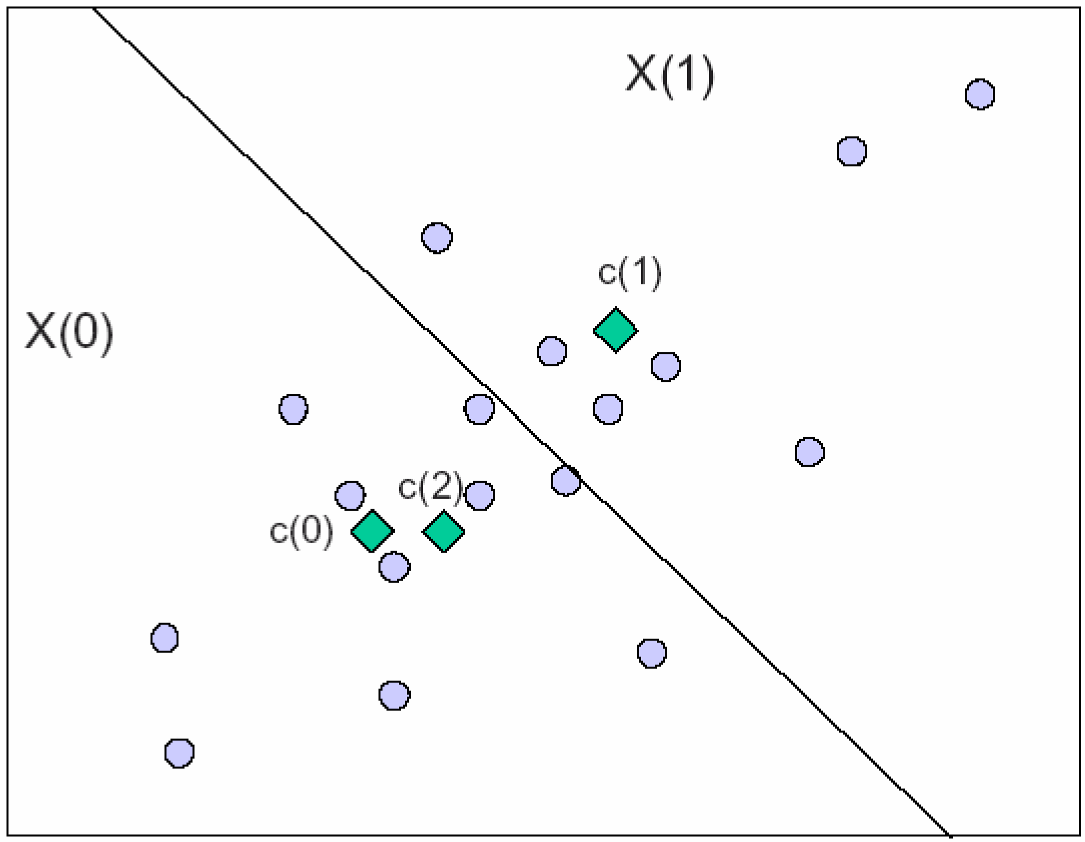


Example of Splitting

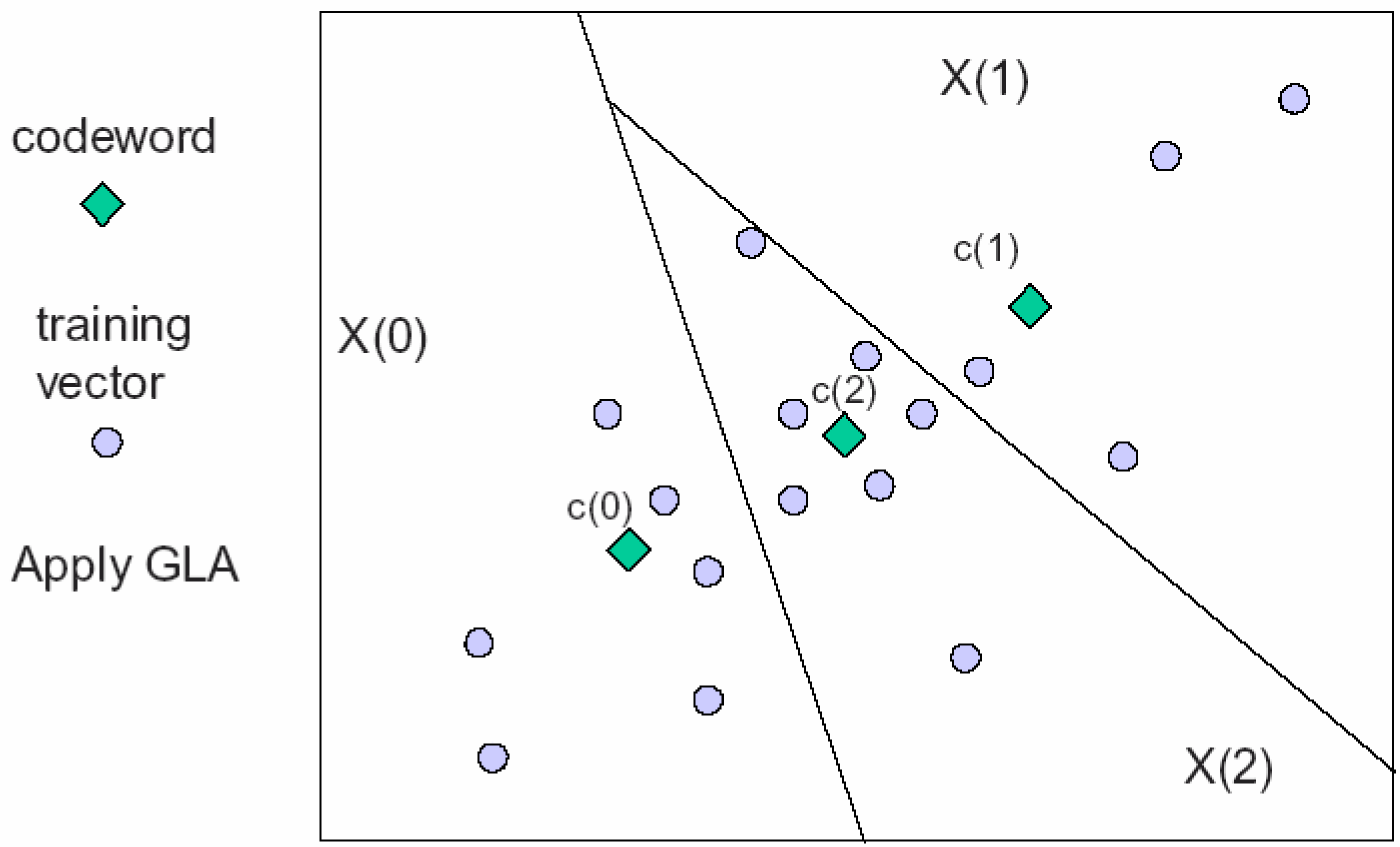
codeword

training vector

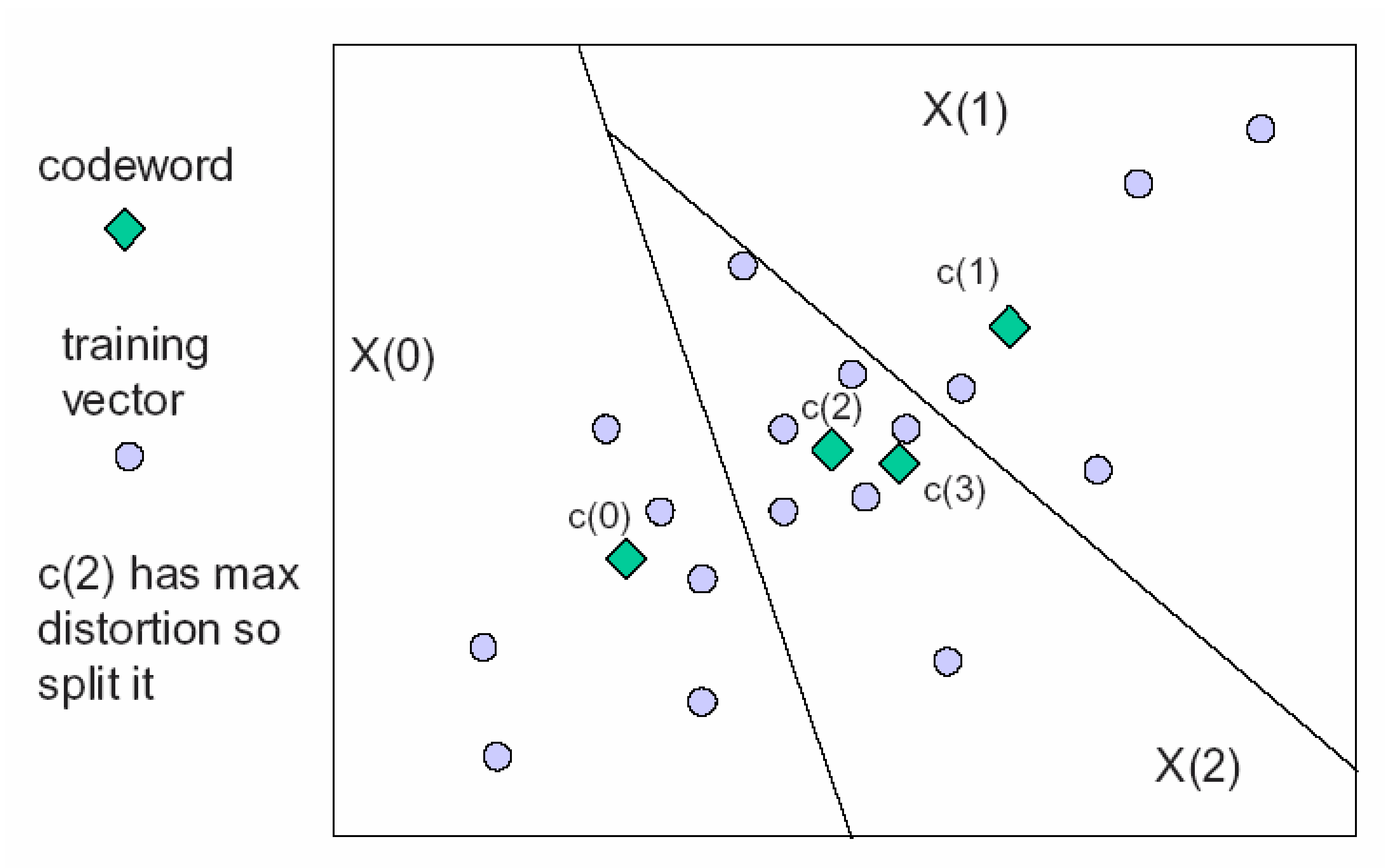
c(0) has max distortion so split it.



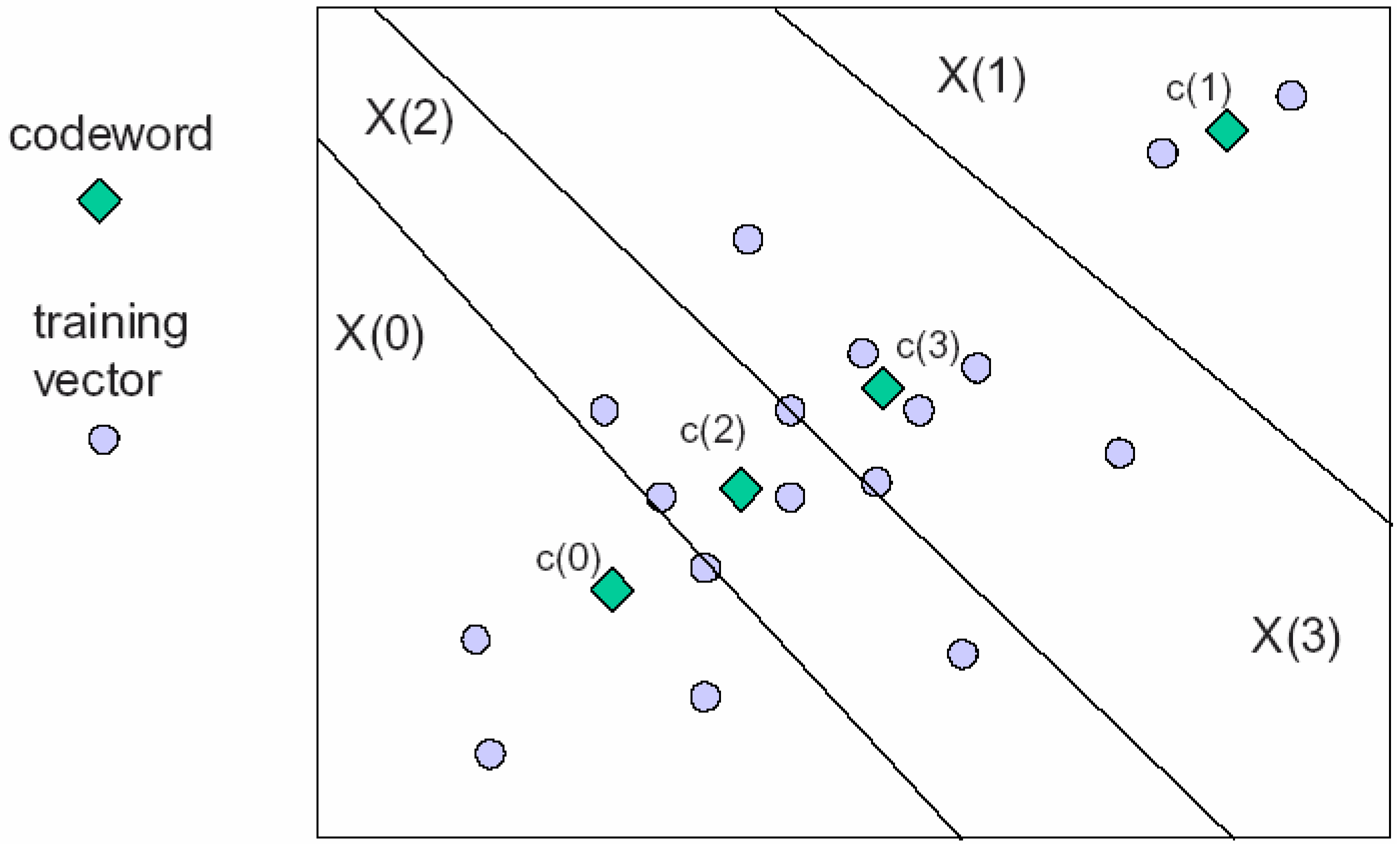
Example of Splitting



Example of Splitting



Example of Splitting



GLA Advices

- Time per iteration is dominated by the partitioning step, which is *m* nearest neighbor searches where *m* is the training set size.
  - Average time per iteration O(*m* log *n*) assuming is small.
- Training set size:
  - Training set should be at least 20 training vectors per code word to get reasonable performance.
  - Too small a training set results in "over training."
- Number of iterations can be large.