Karlsruher Institut für Technologie

Neuronale Netze

Learning Vector Quantization and Related Techniques

Joshua Winebarger

13.12.2011

Kohonen Maps

- Central question:
  - Can we find a lower-dimensional representation of the data which preserves the relations between patterns in the input?
- Method:
  - Unsupervised competitive learning in a two-dimensional neural network
- Results:
  - Line, square, or cube providing a mapping of the data
  - Emphasis on visual presentation of data
  - Applications in classification, automatic control, image and speech processing
- Do there exist other methods for doing this?

Karlsruher Institut für Technologie

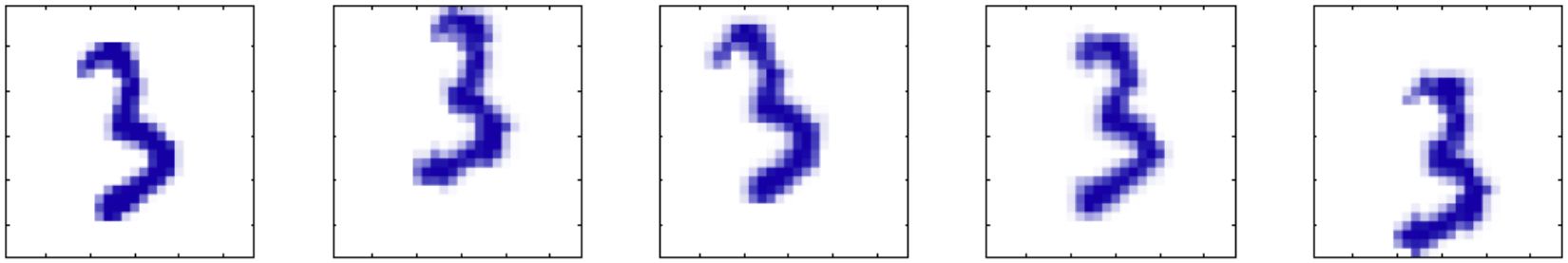
Dutch-English Word Mapping

DEMO 1

Karlsruher Institut für Technologie

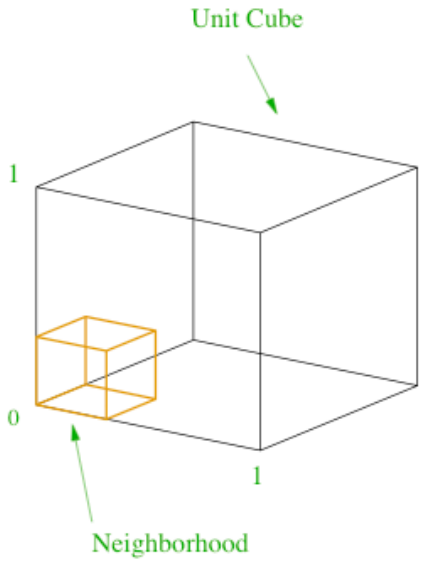
PRINCIPAL COMPONENTS ANALYSIS

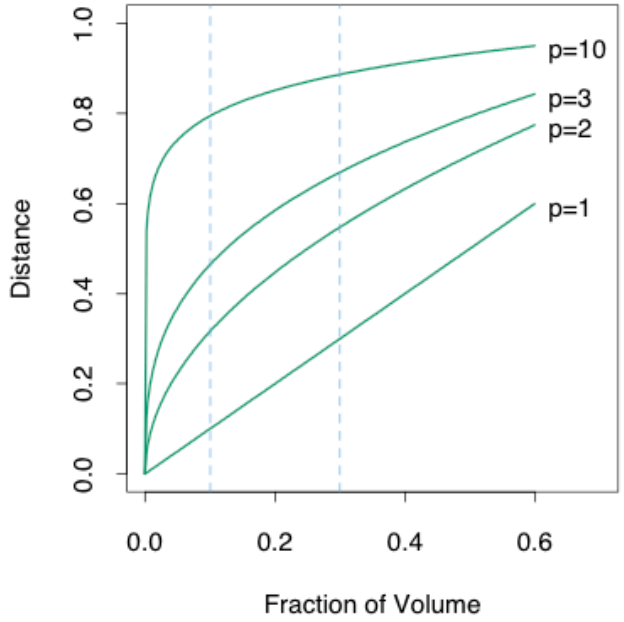
Motivating Example



- Consider hundreds of 100x100-pixel images variable in displacement, scale, and rotation (latent variables)
- Dimension of latent variables much smaller than image dimension
- How to recover latent variables?

Curse of Dimensionality





- Consider p-dimensional unit hypercube containing observations
- Suppose a neighborhood capturing a fraction *r* of observations
  - i.e. a fraction *r* of the unit volume
- Expected edge length will be $e(r)=r^{1/p}$
- In ten dimensions:
  - $e_{10}(0.01)=0.63$
  - $e_{10}(0.1)=0.80$

Curse of Dimensionality

- Distance functions lose their usefulness in high dimensionality

$$V_{sphere} = \frac{2r^d \pi^{d/2}}{d\Gamma(d/2)}$$

$$V_{cube} = (2r)^d$$

$$\lim_{d \to 0} \frac{r_{\max} - r_{\min}}{r_{\min}} = 0$$

$$R = \frac{V_{sphere}}{V_{cube}} = \frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)}$$

$$\lim_{d \to \infty} R = 0$$

Principal Components Analysis

- Invented by Karl Pearson in 1901
- Also known as:
  - Karhunen-Loève transform in information theory
  - Hotelling transform in image analysis
  - Latent Semantic Analysis in text processing
- Linear transformation to a new coordinate system
- New variables -- principal components
  - linear functions of the original variables
  - Uncorrelated
  - Greatest variance by any projection of the data comes to lie on the first coordinate
    - Second greatest variance on the second coordinate etc.

Principal Components Analysis

- Aims
  1. Find a set of *K* orthogonal vectors in data space accounting for as much of the data's variance as possible
  2. Projection of data from original *D*-dim space to *K*-dim space spanned by these vectors
  3. Retain as much of the intrinsic information in the data as possible
- Results
  - Typically M<<N → reduced data much easier to handle in searching for clusters
  - Guarantees in terms of minimizing least squares error in the new approximation

Why preserve variance of data?

- Selecting for vectors spanning the data in the directions of highest variance
- Equivalent to maximizing the information content of output projection where it has a gaussian distribution
- Information Theory
  - Shannon entropy quantifies the expected value of information contained in a message

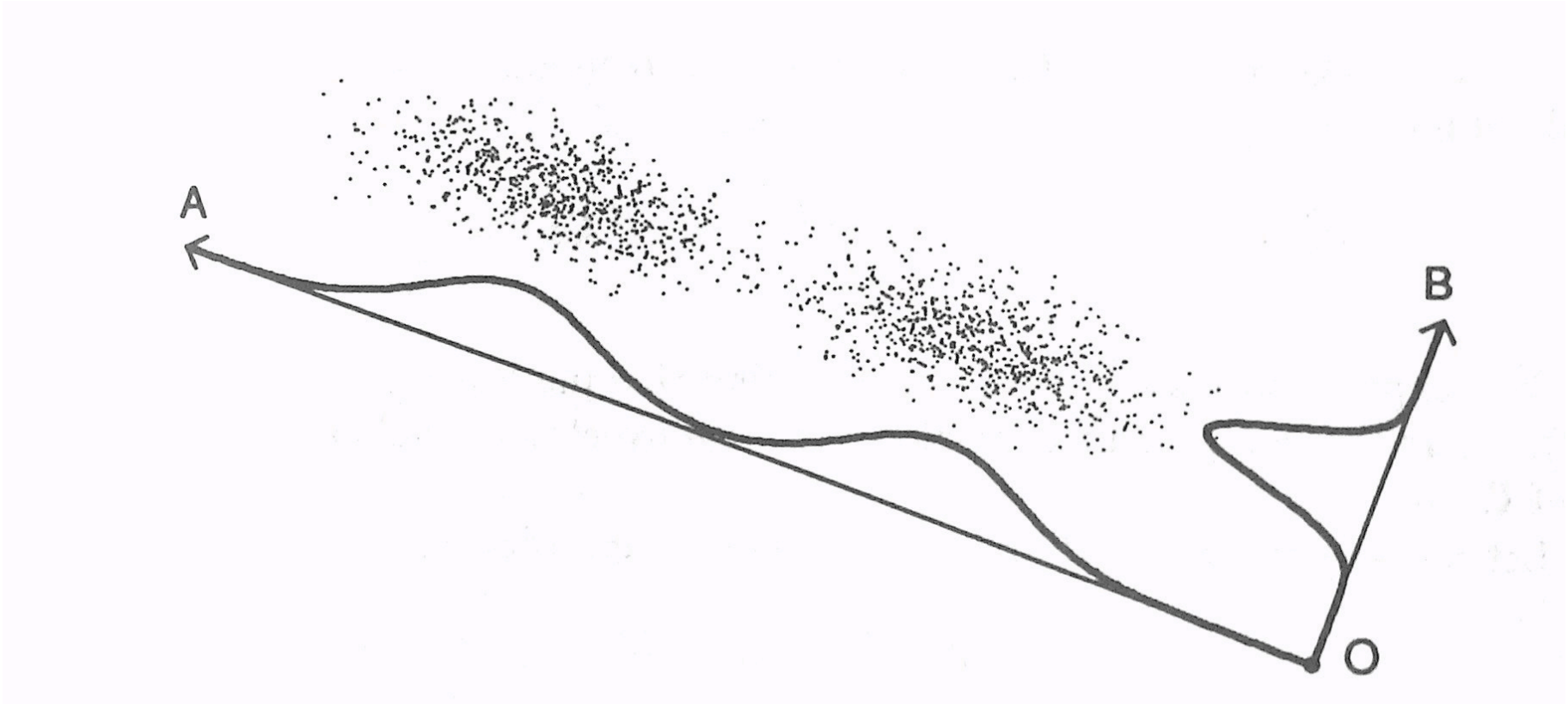
$$H(X) = -\int_{-\infty}^{\infty} p(x) \log\{p(x)\} dx$$

  - Which for a gaussian is:

$$\tfrac{1}{2} \ln\left(2\pi e \sigma^2\right)$$

  - Entropy or information content depends on variance of data

Example



Derivation

- Consider *N* data points $x_1,\ldots,x_N$ in R^D
  - *D* x *N* matrix *X*
- We wish to find an orthogonal linear mapping onto a lower dimensional subspace R^K: $R^D \rightarrow R^K$ such that the variance in of the data in the new space is maximal
- Mapping given by *D* x *K* matrix *U* with $U^T U = I_k$
  - Columns of *U* are orthogonal and have unit length
- Mapping is given as $y = U^T x_n$
- Mean of projected data given by $U^T \bar{x}$
- Variance of projected data is:

$$\frac{1}{N}\sum_n \left(U^T x_n - U^T \bar{x}\right)\left(U^T x_n - U^T \bar{x}\right)^T = U^T S U \qquad S = \frac{1}{N}\sum_n \left(x_n - \bar{x}\right)\left(x_n - \bar{x}\right)^T$$

- Total variance: $trace\left(U^T S U U^T S U\right) \quad (1)$

Derivation (Cont.'d)

- We can also find the projection U onto R^K s.t. the mean squared distance between data and its projection back from the new space is minimum
- Center the data first by subtracting a matrix of columns of means
- Then minimize: $R = \left\| UU^T \left(X - \bar{X} \right) - \left(X - \bar{X} \right) \right\|_2$
- This measure is equivalent to:

$$trace\left(RR^T\right)$$

$$= trace\left(UU^T SUU^T\right) - 2trace\left(UU^T S\right) + trace\left(S\right)$$

$$= trace\left(UU^T S\right) + H$$

- Using trace(AB) = trace(BA) and UU^T being idempotent
- Thus maximize $trace\left(UU^T S\right) \quad (2)$

PCA Algorithm (Informal)

- First principal component u_1 taken along direction of maximum variance
- Second principal component u_2 lies in subspaceperpendicular to the first
  - Taken as direction of maximum variance in this subspace
- Third principal component taken in direction of max. variance in subspace perpendicular to both u_1 and u_2

PCA Algorithm

- Computation of principal components u_i
  - The minimization of (1) and maximization of (2) are equivalent to setting the principal components to the *K* eigenvectors corresponding to the *K* largest eigenvalues of the sample covariance matrix *S*
- For centered data this reduces to finding the eigenvectors of *C*:

$$C = E\left\{XX^T\right\}$$

$$\sigma_x^2 = E\left\{\left(X^T x\right)^2\right\} = E\left\{\left(x^T XX^T x\right)^2\right\} = x^T Cx = \sum_\alpha \lambda^\alpha x_\alpha^2$$

PCA Algorithm Proof

- Proof for k^{th} principal component:
  - Variance along direction of a unit vector x:

$$\sigma_x^2 = E\left\{\left(X^T x\right)^2\right\} = E\left\{\left(x^T X X^T x\right)^2\right\} = x^T C x = \sum_\alpha \lambda^\alpha x_\alpha^2$$

  - Where x_α is the component of x along the eigenvector c^α belonging to the eigenvalue λ^α of C
  - Take eigenvalues in decreasing order:
    - $\lambda^1 \geq \lambda^2 \geq \ldots \geq \lambda^N$ where $\lambda^1 = \lambda_{max}$
  - Assume princpal components 1 to k-1 are along the first k-1 eigenvector directions (Induction)
  - u_k is constrained to be perpendicular to these directions
  - Therefore $x_1 \ldots x_{k-1} = 0$
  - Maximize σ_x^2 subject to this condition with $|x|=1$ and thus $\Sigma_\alpha x_\alpha^2 = 1$

$$x_j = \begin{cases} \pm 1 & \text{if } j = k \\ 0 & \text{otherwise} \end{cases}$$

  - Therefore the kth principal component is along the kth eigenvector
  - Further, $\sigma_x^2 = \lambda^k$ when x is along u^k

Time for computation

- Principal computational cost is from computation of eigenvectors
- We could compute a full singular value decomposition (SVD) giving all eigenvectors
  - ~$O(D^3)$
- We could iteratively calculate each next largest eigenvector:
  - ~$O(KD^2)$

Karlsruher Institut für Technologie

Dimensionality Reduction with PCA

DEMO 2

Karlsruher Institut für Technologie

K-MEANS CLUSTERING

Cluster Analysis

- Group collections of objects into subsets such that those within each cluster are more closely related to one another than objects assigned to different clusters
- Used to form descriptive statistics
  - Ascertain whether or not data consists of a set of distinct subgroups
- Depends on notion of degree of similarity between objects
- i.e. a way to find similarities in data
- Types of clustering algorithms*:
  - Combinatorial
  - Mixture Modeling
  - Mode seekers

*Elements of Statistical Learning

Combinatorial Clustering

- Assume *N* observations $x_1,\ldots x_N$
- Suppose a fixed number of clusters $K < N$
- Each cluster assigned a label: *k* in {1, … , N}
- (Typically) each observation belongs to only one cluster
  - This implies a mapping $k=C(i)$
  - We seek the mapping $C^*(i)$ that minimizes a loss function based on dissimilarities
- Aim is to partition the data into K clusters
- Result is a partitioning of the data space into Voronoi cells

Loss Functions for Clustering

- Within-cluster or *intra-class* scatter
  - Characterizes closeness of observations within the same cluster

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k} d(x_i, x_{i'}).$$

- Total scatter: $$T = \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} d_{ii'} = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\left(\sum_{C(i')=k} d_{ii'} + \sum_{C(i')\neq k} d_{ii'}\right)$$
- Decomposes to: $$T = W(C) + B(C)$$
- Interclass scatter: $$B(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')\neq k} d_{ii'}$$
- Minimize *W*(*C*) is equivalent to maximizing *B*(*C*)

K-means algorithm (Lloyd's Algorithm)

- Finding *C*(i)* by enumeration is too time-consuming
- Instead use *iterative greedy descent*
  - Convergence to a local optima
- Dissimilarity measure $d(x_i, x_{i'})$ or $d\left(x_i, \bar{x}_k\right)$
  - Choose Euclidean distance: $d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = ||x_i - x_{i'}||^2$
- Minimize $$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} ||x_i - x_{i'}||^2$$
$$= \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2,$$
- Where: $N_k = \sum_{i=1}^{N} I(C(i) = k)$.
- i.e. minimize W(C) by assigning observations to clusters to minimize average dissimilarity of observations from cluster mean

K-means algorithm (Cont.'d)

- Minimize $W(C)$ with respect to class assignement $C(i)$ and means
- Perform cyclic descent:
  1. Fix means, optimize W(C) w.r.t. C(i)
  2. Fix C(i), minimize W(C) w.r.t. means
  3. Repeat until no change in class assignment or means
- ***Lloyd's Algorithm***:
  1. **Classify:** Assign each observation *i* to the nearest mean*:*

$$C(i) = \underset{1 \le k \le K}{\operatorname{argmin}} \left\| x_i - \mu_k \right\|^2$$

  2. **Recenter:** For each class *k*, compute a new centroid as the mean of the updated class assignments:

$$\mu_k = \frac{\sum_{i:C(i)=k} x_i}{\sum_{i:C(i)=k} 1}$$

  3. **Repeat until stopping criteria fulfilled**

Karlsruher Institut für Technologie

K-Means

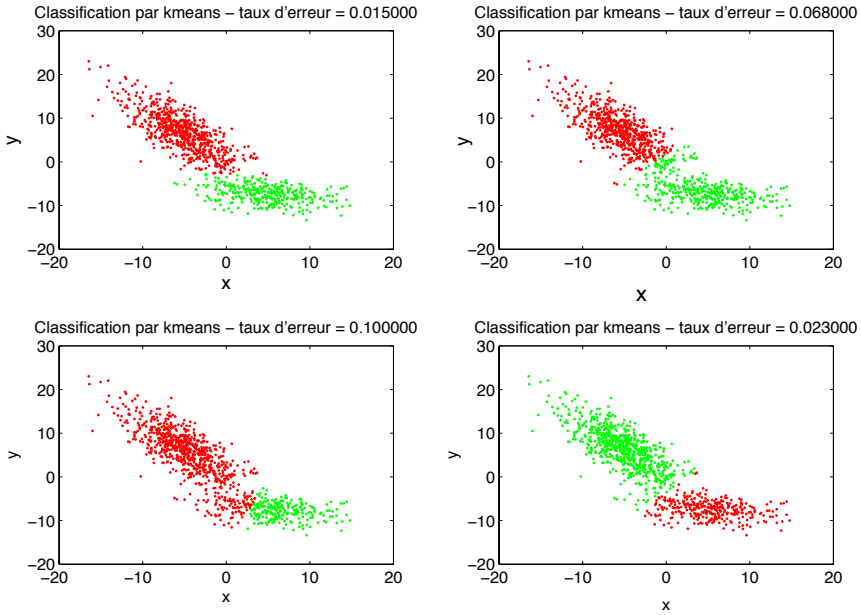
DEMO 3

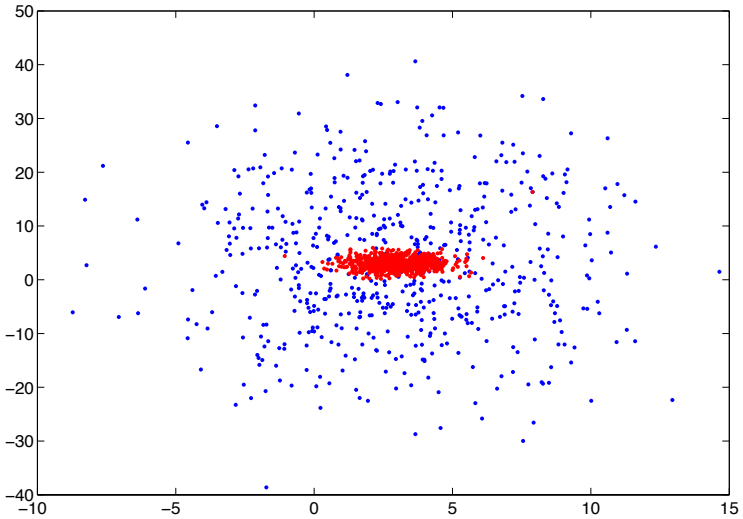
Karlsruher Institut für Technologie

K-Means Matlab Demo

DEMO 3B

Analysis





- K-means seeks to make the size of each cluster approximately the same
- Membership based on location of centroids
- Number of centers fixed in advance – what number?
  - Minimize Schwarz Criterion: $W(C)+\boldsymbol{\lambda mk}\log(\boldsymbol{R})$

Convergence

- **Theorem**: During the course of the k-means algorithm, the loss function monotonically decreases
- **Proof**
  - Let $\mu_1^{(t)},\ldots,\mu_k^{(t)}$ be the k centroids at iteration t
  - Let $C_1^{(t)},\ldots,C_k^{(t)}$ be the clusters at iteration t
  - Step 1 assigns each data point to its closest center, therefore:
    - $\text{loss}(C_1^{(t+1)},\ldots,C_k^{(t+1)};\ \mu_1^{(t)},\ldots,\ \mu_k^{(t)}) \leq \text{loss}(C_1^{(t)},\ldots,C_k^{(t)};\ \mu_1^{(t)},\ldots,\ \mu_k^{(t)})$
  - Step 2 re-centers the data at its mean, giving another reduction in loss:
    - Because $\text{loss}(C;\mu) = \text{loss}(C;\text{mean}(C)) + |C|\bullet||\mu\text{-mean}(C)||^2$
    - $\text{loss}(C_1^{(t+1)},\ldots,C_k^{(t+1)};\ \mu_1^{(t+1)},\ldots,\ \mu_k^{(t+1)}) \leq \text{loss}(C_1^{(t+1)},\ldots,C_k^{(t+1)};\ \mu_1^{(t)},\ldots,\ \mu_k^{(t)})$

Extension – Fuzzy K-Means

- Developed by Dunn in 1973
- Continuous degrees of belonging to classes
- Minimize fuzzy intra-class distances

$$J_m = \sum_{i=1}^{N}\sum_{k=1}^{K} u_{ik}^m \left\| x_i - \mu_k \right\|^2$$

- Algorithm*:

1. Choose initial cluster prototypes
2. Compute degree of membership for all *x* in all clusters *k:*

$$u_{ij}^m = \left[\sum_{j=1}^{K} \left(\frac{\left\| x_i - \mu_k \right\|^2}{\left\| x_i - \mu_j \right\|^2} \right)^{2/m-1} \right]^{-1}$$

3. Compute new centers

$$c_k = \frac{\sum_{i=1}^{N} \left(u_{ik}\right)^m x_i}{\sum_{i=1}^{N} \left(u_{ik}\right)^m}$$

4. Repeat 2 & 3

*« A Fuzzy Clustering Model of Data and Fuzzy c-Means »

Karlsruher Institut für Technologie

Fuzzy k-means

DEMO 4

GMM vs K-Means

- Replace means (centroids) with gaussian mixtures
  - (μ_k, Σ_k)
- Expectation-maximization algorithm for training
  - Similar to k-means algorithm
  - Solve problem with hidden information
- Classification may be done by assigning sample to closest mean
- No strict sense of cluster membership
- K-means equivalent to using spherical covariance matrices of equal size for centroid
- K-means could be used to initialize clusters for GMM

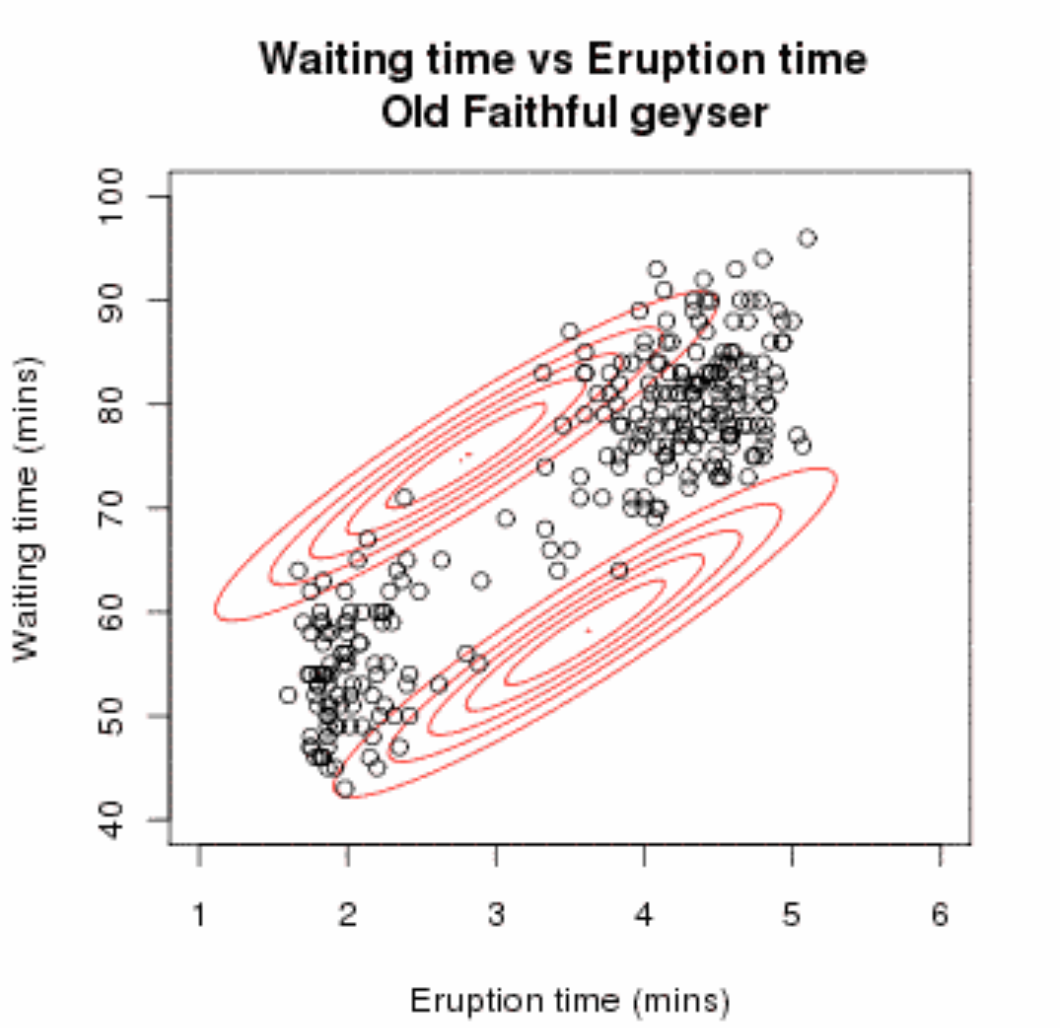
EM Algorithm for GMM

- Coordinate ascent
- Related idea of partial membership (think fuzzy k-means) (class membership probability)
- **Intialization:** Assume initial
  $p(k|i)^{(0)}$, $\mu_k^{(0)}$, and $\Sigma_k^{(0)}$
- **E-Step: Update membership probabilities**

$$p^{(t)}(k \mid n) = \frac{\pi_k^{(t)} g\left(x_i; \mu_k^{(i)}, \Sigma_k^{(i)}\right)}{\sum_{k=1}^{K} \pi_k^{(t)} g\left(x_i; \mu_k^{(i)}, \Sigma_k^{(i)}\right)}$$

- **M-Step: Update Gaussians and prior probabilites**

GMM Example


Waiting time vs Eruption time
Old Faithful geyser
Waiting time (mins)
40
50
60
70
80
90
100
1
2
3
4
5
6
Eruption time (mins)

VECTOR QUANTIZATION

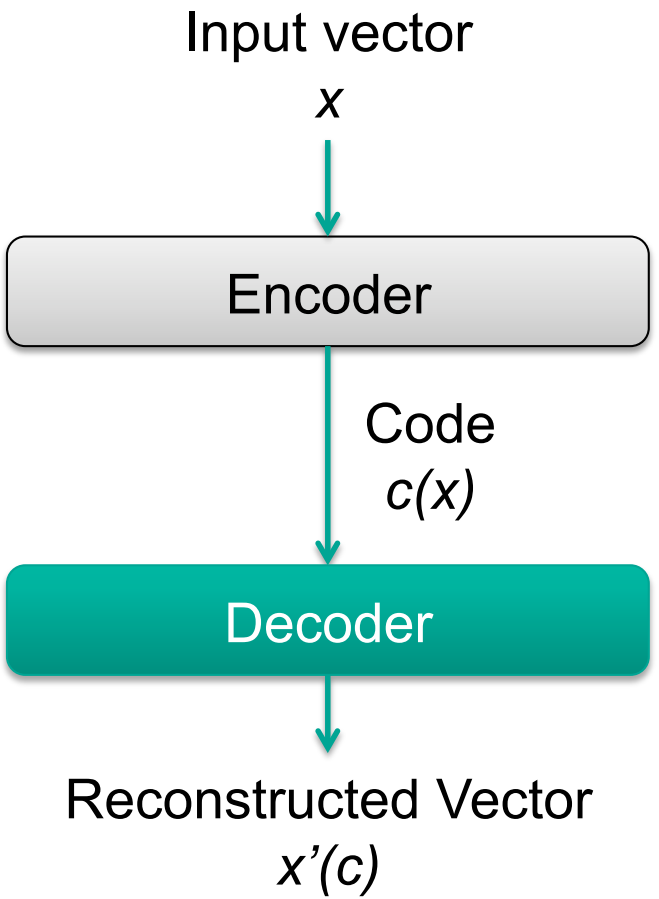
Vector Quantization (VQ)

- VQ Theory:
  - Approximate the data space with a smaller number of vectors
  - Categorize a set of input vectors *x* into *M* classes
  - Each class has an associated **prototype vector**
    - Set of all prototype vectors is called a ***codebook***
  - Represent any vector by its class
    - Finding appropriate class:
      - Identify nearest prototype vector
  - Similar to competitive learning:
    - U_i are prototype vectors
    - Find class by winner: $\|u_c - x\| \leq \|u_i - x\| \text{ for all } i$

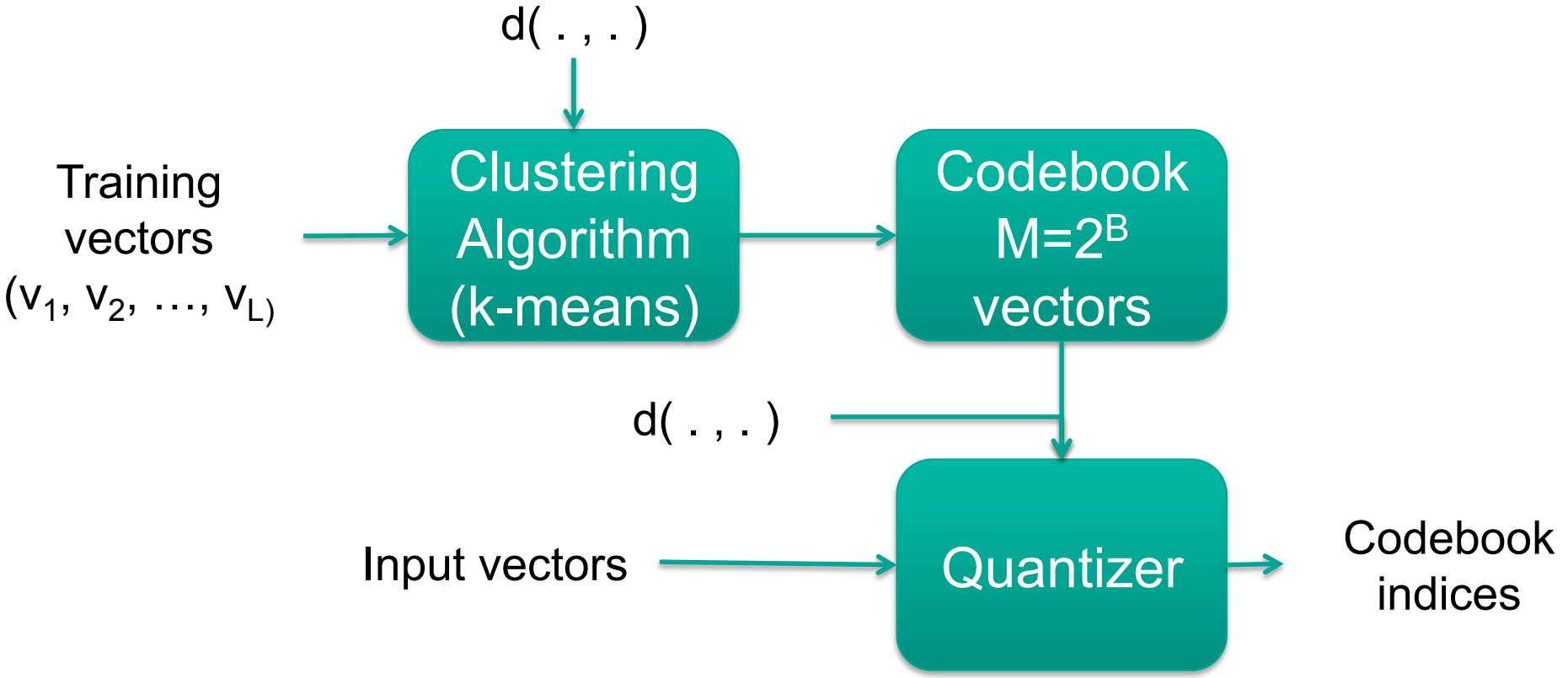
Applications of VQ

- Multimedia compression for storage and transmission
- Dimensionality reduction
- Classification
  - Ex.: Tokenization of speech frames

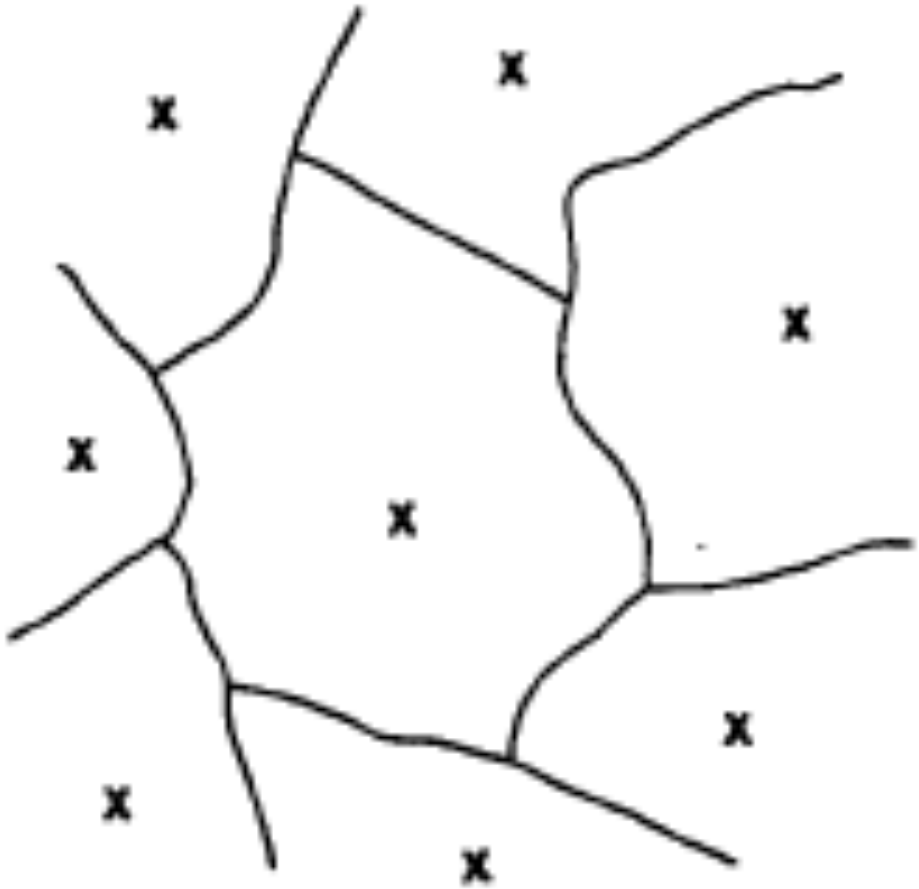
Encoder-Decoder Model



Basic VQ Training and Classification



Voronoi Regions



Derivation of VQ Algorithm

- Distortion measures *d*

1. Most common is squared-error:

$$d(x,x') = \sum_{i=0}^{k} |x_i - x_i'|^2$$

2. Itakura, Saito, and Chaffee

$$d(x,x') = (x - x')R(x)(x - x')^T$$

- Optimality
  - Let X = (X_0, … , X_k)
  - Expected distortion with respect to underlying distribution:
    - D(q) = E{ d(X,q(X)) }
  - Given quantizer *q* with codebook vectors $U = \{ u_i ; i=1, \ldots, M \}$ yielding a partition $S = \{ S_i ; i= 1, \ldots, M \}$

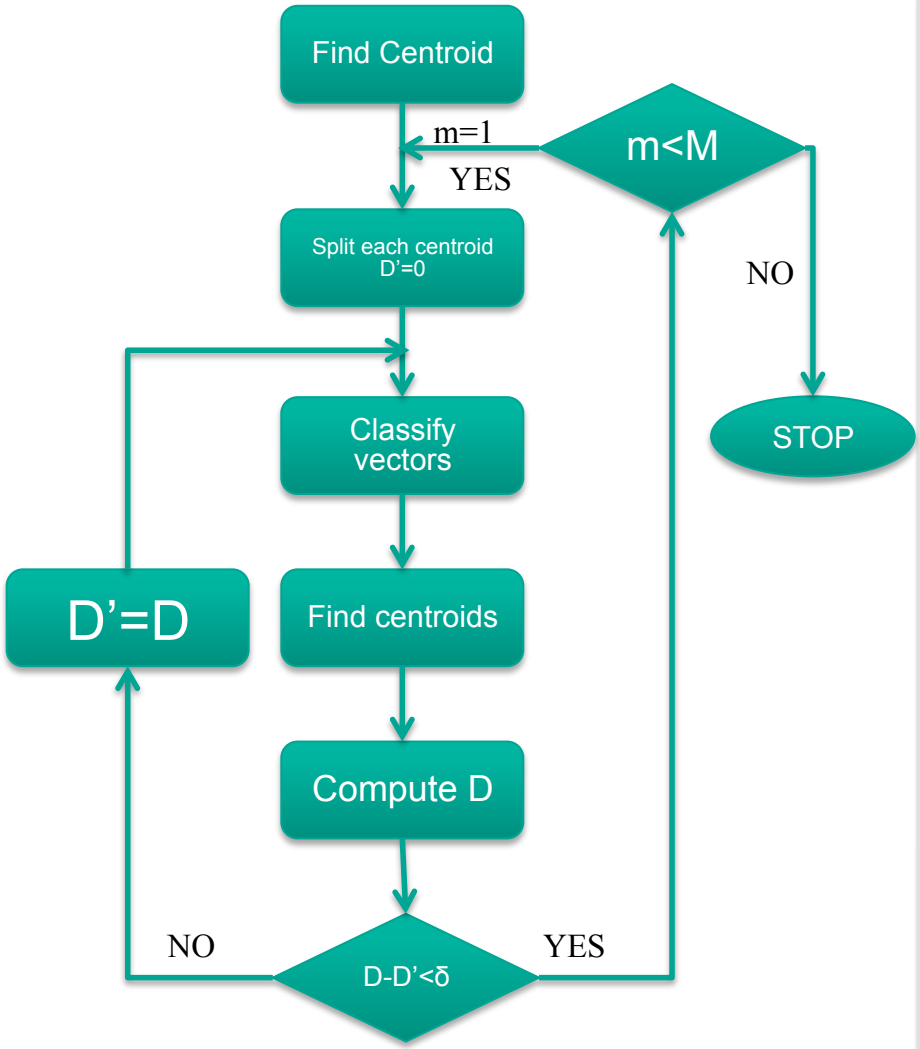
$$D(\{U,S\}) = E\{d(X,q(X))\}$$

$$= \sum_{i=1}^{M} E\{d(X,u_i) \mid X \in S_i\}$$

  - N-level quantizer is optimal if it minimizes the expected distortion
  - Result is k-means algorithm

Iterative VQ Algorithm

- Can design M-vector codebook in stages
- Procedure
  - First a 1-vector codebook
  - Split to initialize search for 2-vector
  - Continue splitting until M-vector codebook
- Algorithm
  1. Design a 1-vector codebook $U_1 = u_1$
  2. Double size of codebook by splitting U_n according to the rule:
     1. $U_n^+ = U_n(1+\varepsilon)$
     2. $U_n^- = U_n(1-\varepsilon)$
  3. Use the K-means algorithm to get the best centroids for the two codebooks
  4. Merge U_n^+ and U_n^- to make U_{n+1}
  5. Repeat steps 2 through 4



Supervised SOM

- SOM is typically an unsupervised process
- Classification accuracy can be improved if class information used in the learning phase:
- Form input vectors of two parts:
  - x_s the data
  - x_u the class information
  - $x = [\ x_s^T\ ,\ x_u^T\ ]^T$ then used as input to SOM
- Enhanced class-separation
- Recognition phase:
  - Only the x_s part is compared with weights

Karlsruher Institut für Technologie

LEARGNING VECTOR QUANTIZATION

Learning Vector Quantization

- Kohonen suggested a **supervised** form of VQ called LVQ
- Class of related algorithms: LVQ1, LVQ2, LVQ3, and OLVQ
- VQ and SOM are unsupervised clustering and learning
- LVQ uses supervised learning, but with no spatial order of codebook vectors

Optimal Decision

- Optimal decision discussed in framework of Bayes theory of probability
- Assume all samples of x are derived from a finite set of classes $\{ S_k \}$ with overlapping distributions
- $P(S_k)$: *a priori* probability of classes S_k
- $p(x \mid x \text{ in } S_k)$: conditional prob. Density of x on S_k
- Discriminant functions:

$$\delta_k(x) = p(x \mid x \in S_k) P(S_k) \quad \text{(I)}$$

- Rate of misclassification minimized if a sample x is classified by:

$$\delta_c(x) = \max_k \{\delta_k(x)\} \quad \text{(II)}$$

LVQ Approach

- Approach:
  - We assign a subset of codebook vectors to each class S_k
  - Then we search for the codebook vector m_i closest to x
  - x is classified as same class as closest m_i
  - Only codebook vectors near edge of class borders are important
    - A good approximation of $p(x \mid x \text{ in } S_k)$ is not necessary everywhere
    - Place the m_i into signal space to minimize average expected misclassification probability

LVQ1

- Assume several codebook vectors assigned to each class of x values and that x is assigned the class of the nearest m_i
- Let the index of the winning codebook vector c be:

$$c = \underset{i}{\operatorname{argmin}} \left\{ \| x - m_i \| \right\}$$

- Let $x(t)$ be an input sample
- Let $m_i(t)$ be the sequential values of the m_i in the discrete-time domain
- Start with properly defined initial values
- Apply reward-punishment learning rule for each x:
- Samples applied cyclically

$$m_c(t+1) = m_c(t) + \alpha(t)\left[x(t) - m_c(t)\right]$$

if x and m_c belong to the same class

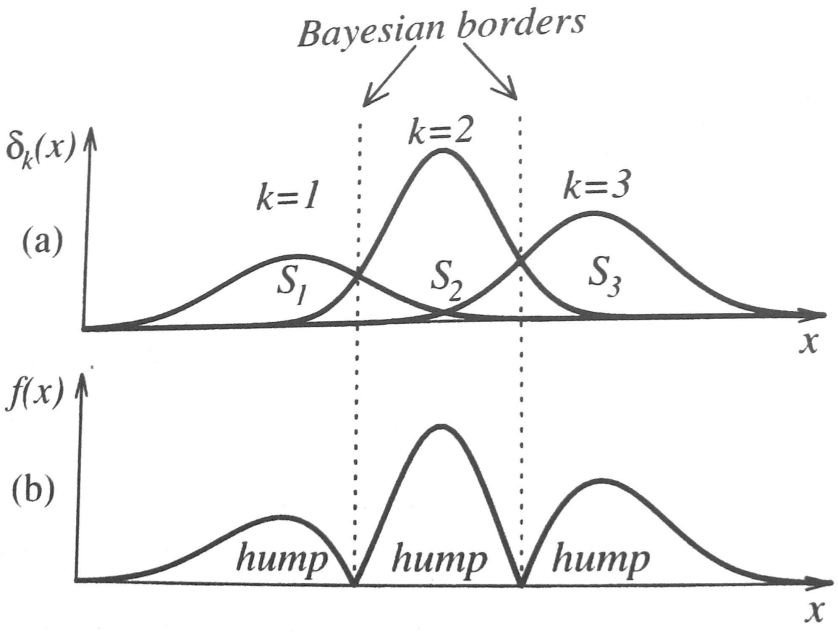
$$\text{(III)} \quad m_c(t+1) = m_c(t) - \alpha(t)\left[x(t) - m_c(t)\right]$$

if x and m_c belong to different classes

$$m_c(t+1) = m_i(t) \text{ for } i \neq c$$

LVQ1 – Derivation

- Assume we want to approximate a density function $f(x)$ with the LVQ
- Let optimal decision (Bayesian) borders be defined by equations (I) and (II) (previous slide)
  - These borders divide signal space into class regions B_k s.t. misclassification is minimized
- $f(x)$ has the form:



for $x \in B_k$ and $h \neq k$

$$f(x) = p(x \mid x \in S_k) P(S_k) - \max_h \{ p(x \mid x \in S_h) P(S_h) \}$$

LVQ1 – Derivation

- Use VQ to define point density of *m* approximating *f*(x)
- Optimal values found by minimizing average expected quantization error *E*

$$\nabla_{m_i} E = -2 \int \delta_{ci} \cdot \left(x - m_i \right) p\left(x \right) dx$$

- Only winner should be updated:
- Gradient step:

$$m_i\left(t+1\right) = m_i\left(t\right) - \lambda \cdot \nabla_{m_i(t)} E$$

$$= m_i\left(t\right) - \lambda \cdot \nabla_{m_i(t)} 2\delta_{ci} \left[x\left(t\right) - m_i\left(t\right) \right]$$

- Replace *f*(*x*) with *p*(*x*)
- With some derivation:

$$\nabla_{m_i(t)} E = -2\delta_{ci} \left[x\left(t\right) - m_i\left(t\right) \right] \text{ if } x\left(t\right) \in S_k$$

$$\nabla_{m_i(t)} E = -2\delta_{ci} \left[x\left(t\right) - m_i\left(t\right) \right] \text{ if } x\left(t\right) \in S_r$$

- Where *r* is the runner-up class

$$r = \underset{h}{\operatorname{argmax}} \left\{ p\left(x \mid x \in S_h \right) P\left(S_h \right) \right\}$$

LVQ1 – Derivation

- Rewrite $\alpha(t)=2\lambda$:

$$
\text{(IV)}\quad
\begin{aligned}
&m_c(t+1) = m_c(t) + \alpha(t)\left[x(t) - m_c(t)\right] \text{ if } x(t) \in B_k \text{ and } x(t) \in S_k \\
&m_c(t+1) = m_c(t) - \alpha(t)\left[x(t) - m_c(t)\right] \text{ if } x(t) \in B_k \text{ and } x(t) \in S_r \\
&m_c(t+1) = m_c(t) \text{ if } x(t) \in B_k \text{ and } x(t) \in S_h, h \neq r \\
&m_i(t+1) = m_i(t) \text{ if } i \neq c
\end{aligned}
$$

- Notes:
  - In (III) the « punishment » correction made every time *x* miscalssified
  - In (IV) it is made only if *x* is in the runner-up class
  - If *x* in neither class, no change is made
  - LVQ2 and LVQ3 are closer to (IV) than LVQ1

OLVQ1

- Determine the optimal learning correction factor α(t) for fastest convergence
- Derivation
  - Express (III) in the form
  - $m_c(t+1) = [\ 1 - s(t)\alpha_c(t)\ ]m_c(t)+s(t)\alpha_c(t)x(t)$
    - where $s(t)$=+1 if classification is correct, and $s(t)$=-1 if incorrect
  - m_c(t+1) contains a trace of $x(t)$
  - Traces of earlier $x(t)$ are contained in the term $m_c(t)$
  - Magnitude of last trace of $x(t)$ scaled by factor $\alpha_c(t)$
  - During this step the trace of $x(t-1)$ has become scaled down by $[\ 1 - s(t)\alpha_c(t)\ ]\ \alpha_c(t-1)$
  - Stipulate identical scaling: $\alpha_c(t) = [\ 1 - s(t)\alpha_c(t)\ ]\ \alpha_c(t-1)$
  - Hold for all t, induction: all traces collected up to time t of earlier $x(t)$ will be saceld by an equal amount at the end
  - Optimal values of α(t) determined by recursion:

$$\alpha_c(t) = \frac{\alpha_c(t-1)}{1+s(t)\alpha_c(t-1)}$$

LVQ2

- Differentially shift the decision borders toward the Bayesian limits
- Identical classification decision with LVQ1
- Difference(s):
  1. Two codebook vectors m_i and m_j updated simultaneously
     1. m_i and m_j are nearest neighbors of x
  2. x must fall into a « window » defined around midplane of m_i and m_j
     - Assume d_i and d_j are Euclidean distances of x from m_i and m_j
     - x falls in a window w if

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s, \text{ where } s = \frac{1-w}{1+w}$$

     - Window of 0.2 to 0.3 is recommended

LVQ2.1

- Improvement over LVQ2
- Allows either m_i or m_j to be the closes codebook vector

$$m_i(t+1) = m_i(t) - \alpha(t)\left[x(t) - m_i(t)\right]$$
$$m_j(t+1) = m_j(t) + \alpha(t)\left[x(t) - m_j(t)\right]$$

- Where:
  - m_i and m_j are the two closest codebook vectors to x,
  - x and m_j belong to the same class
  - x and m_i belong to different classes
  - x must fall into the ‘window’

LVQ3

- Improvement over LVQ2
- Introduce correction to ensure that m_i continue approximating the class distributions ($f(x)$), at least roughly
- Algorithm:

$$m_i(t+1) = m_i(t) - \alpha(t)\left[x(t) - m_i(t)\right]$$

$$m_j(t+1) = m_j(t) + \alpha(t)\left[x(t) - m_j(t)\right]$$

- Where:
  - m_i and m_j are the two closest codebook vectors to x
  - x and m_j belong to the same class
  - x and m_i belong to different classes
  - x falls into the window

$$m_k(t+1) = m_k(t) + \varepsilon\alpha(t)\left[x(t) - m_k(t)\right]$$

$$\text{for } k \in \{i, j\}, \text{ if } x, m_i, \text{ and } m_j \text{ belong to the same class}$$

Differences between LVQ1, LVQ2, and LVQ3

- LVQ1 and LVQ3 are more robust
  - Codebook vectors assume stationary values over extended learning periods
- LVQ1 can be optimized for quick convergence
- LVQ2: Relative distances of codebook vectors from class borders optimized
  - No guarantee of optimal vector placement to describe forms of class distributions
  - Therefore use a small value of learning rate and limited number of training steps

Initialization of Codebook Vectors

- Iterative assignment
  - Due to fact that class distributions are unknown
  - Final placement of codebook vectors not known until end of learning
  - Distance and optimal numbers cannot be pre-determined
- Practical step:
  - Start with same number of codebook vectors in each class
  - Upper limit to total number of codebook vectors: time and compute power available
- Determine min. number for codebook vectors per class:
  - Medians of shortest distances between codebook vectors should be somewhat smaller than the standard deviations of input samples in all respective classes
- Initial values of codebook vectors:
  - Use first samples of training data picked from respective classes
  - These samples must pass a K-Nearest-Neighbor test of tentaive classification

Learning & Stopping

Optimal Learning

- Begin learning with OLVQ1 for fast convergence
  - # steps: ~30 to 50 times total number of codebook vectors
- Continue with other algorithms with a low initial learning rate

Stopping Rule

- NN algorithms may « overlearn »
  - Over-specialization to the data
- As with NN:
- Divide data into *training*, *validation*, and *test* sets
- Test against validation set after every training iteration
- Stop training using some heuristic on the performance on the validation set

Karlsruher Institut für Technologie

SUMMARY

Summary

- SOM
  - Dimensionality Reduction
  - Spatial Representation
  - Unsupervised Competitive Learning
- PCA
  - Dimensionality Reduction
  - Spatial Representation
- VQ
  - Uses K-Means algorithm
  - Finding prototypes from data
- K-Means
  - Constrained case of EM algorithm for GMM
- Supervised SOM
  - Unsupervised learning using labels
- LVQ
  - Supervised learning for VQ
  - Finding prototypes from data

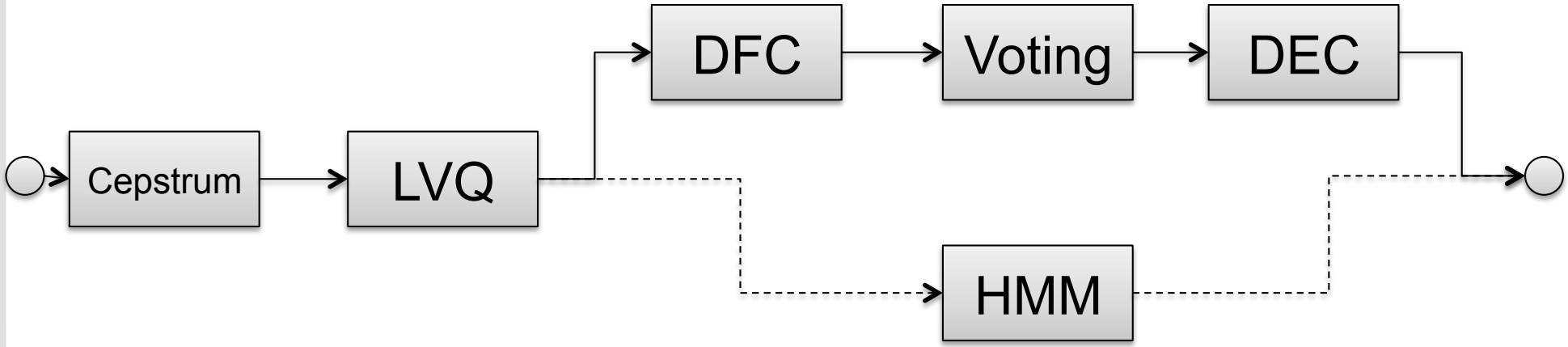
Karlsruher Institut für Technologie

APPLICATIONS

Application: Transcription of Continuous Speech

- Computation of Short-time Cepstrum
  - Elimination of peaks due to harmonics
- Conversion of Cepstra to Quasiphonemes
  - LVQ
    - Used to assign an acoustic label every 10ms
- Correct for coarticulation effects
  - Use context-depended quasiphoneme grammar
  - DFC --- Dynamically Focusing Context
- Merge quasiphones to phonemes / Decoding Quasiphone Sequences into Phones
  - Voting
    - Consider *n* successive labels
    - Determine meta-label by majority of these *n* labels
    - Label the *n* labels with this meta-label
    - Heuristic rules to ensure no overlap of decision, proper number of phonemes
  - HMM
- Dynamically Expanding Context
  - Symbolic method for correcting phonemic errors and translate phonemes to orthographic text

Application: Transcription of Continuous Speech



- Results:
  - Three Finnish speakers
  - Four repetitions of a set of 311 words
    - Three repetitions for training, one for testing (leave-one-out principal)
  - 95.6% Correctness

Application: VQ For Speech Compression

- Assume we require a codebook with about 1024 unique spectral vectors
  - 25 variants for each of the 40 basic speech units
  - Need 10-bit number to represent an arbitrary spectral vector
- Assume a rate of 100 spectral vectors per seond
- Then a total bit rate of 1000 bps is required to transmit a speech signal
- This is about 1/16th the rate for a non-compressed signal
- VQ Representation of speech can be very efficient

Application: VQ for Speech Compression

Advantages

- Reduced storage space
- Reduced computation for determining spectral similarity (lookup table)
- Recognition through discrete representation possible

Disadvantages

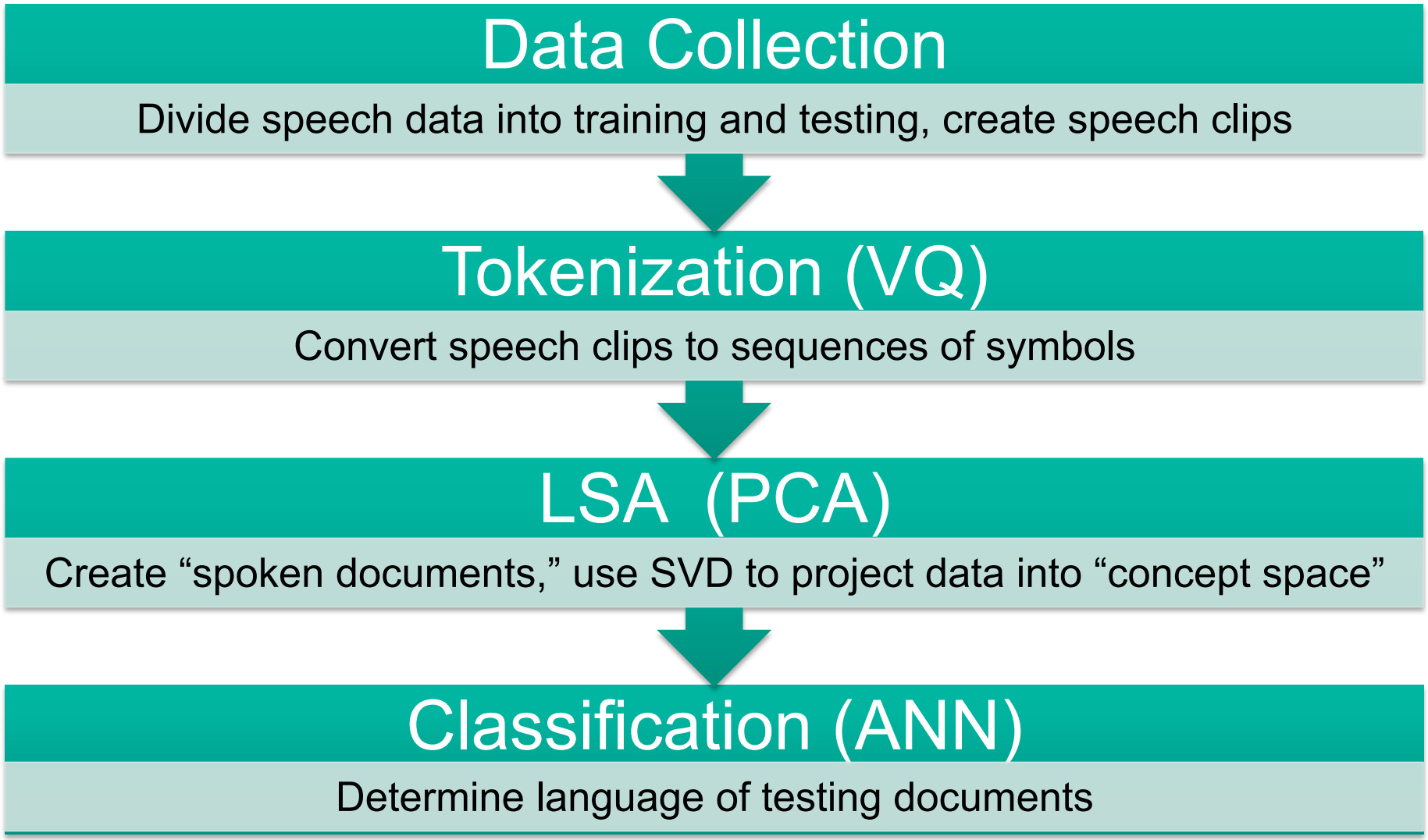
- Inherent spectral distortion
  - Quantization error decreases with size of codebook
- Storage for codebook vectors is often non-trivial

Application: Spoken Language Identification

- Design a system for identifying the language of a clip of speech based on prior examples of languages
- Supervised learning: train classifier with speech representing two or more languages
  - Tokenize speech (VQ)
  - Create « spoken documents »
  - Use text categorization techniques to classify each document (PCA)
    - Projection into lower-dimensional « concept space »
  - Classification

Example Application: Spoken Language Identification

Karlsruher Institut für Technologie



Example Application: Spoken Language Identification

- Division of data into training and testing set
- Creation of spoken documents
  - Creation of 30-second clips (documents)
  - 45 millisecond frames
  - Computation of linear prediction coefficients (LPC) cepstrum
  - 6-bit VQ: 64 centroids → 64 symbols
    - K-means clustering on 5 minutes of randomly selected speech from each language
  - Computation of symbolic co-occurrence statistics
    - 4094 Bigrams
- Term document matrix
  - Training: 4094x1400
  - Testing: 4094x400
  - Weighting
- Principal Components Analysis → projection into concept space
  - Reduces noise and sparsity
  - Eases comparisons
- Training of classifiers and testing
  - Artificial Neural Network, K-nearest neighbor, Mean-similarity