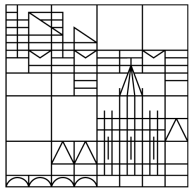
Universität Konstanz



Lecture 12 Mean shift, PCA , and anomaly detection

Dr. Hanhe Lin

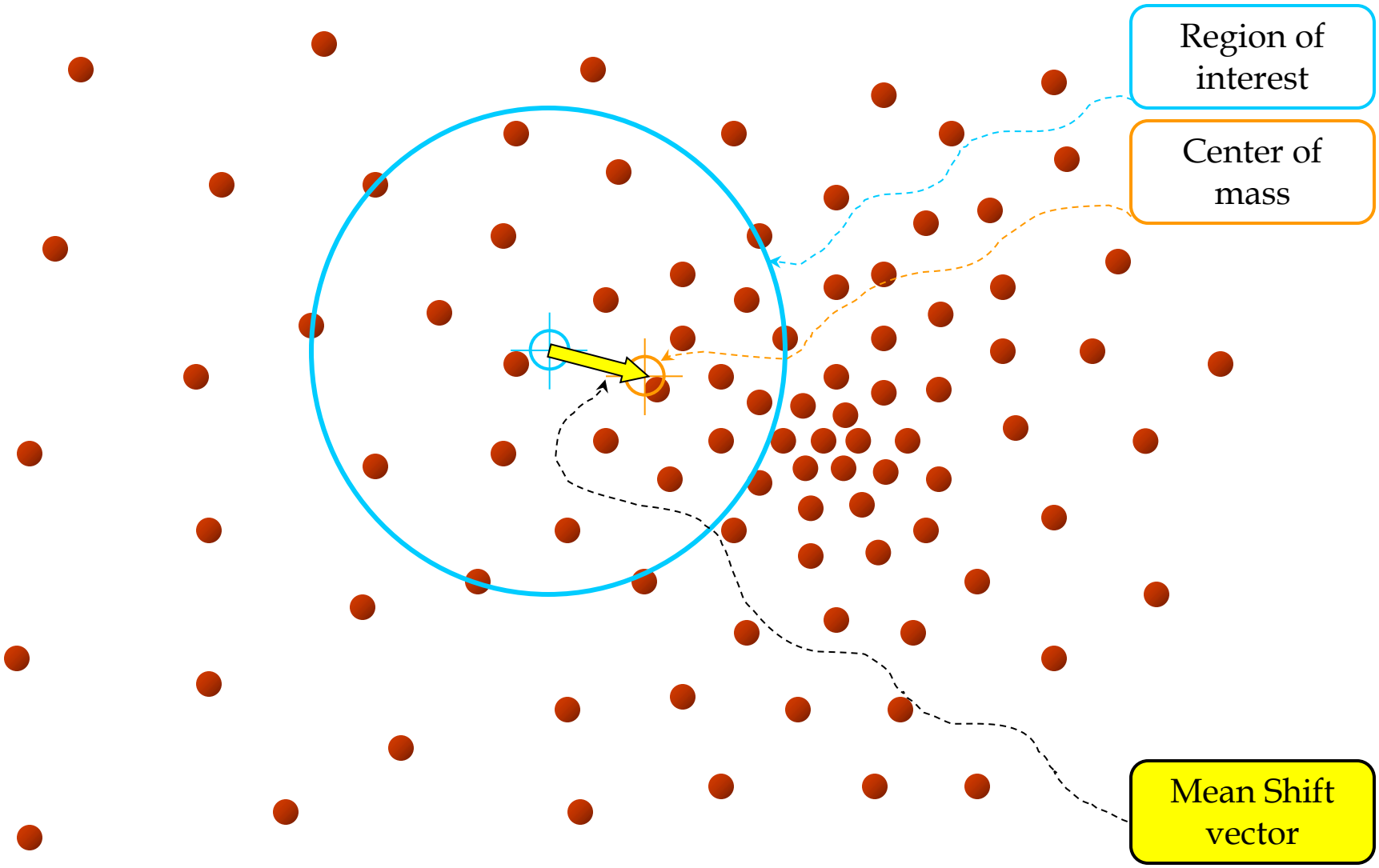
Dept. of Computer and Information Science

University of Konstanz

Outline

- Mean shift
- Dimensionality reduction
  - Principal Component Analysis (PCA)
- Anomaly detection
  - Univariate/Multivariate Gaussian distribution
  - Gaussian mixture model (GMM)
  - One-class Support Vector Machine

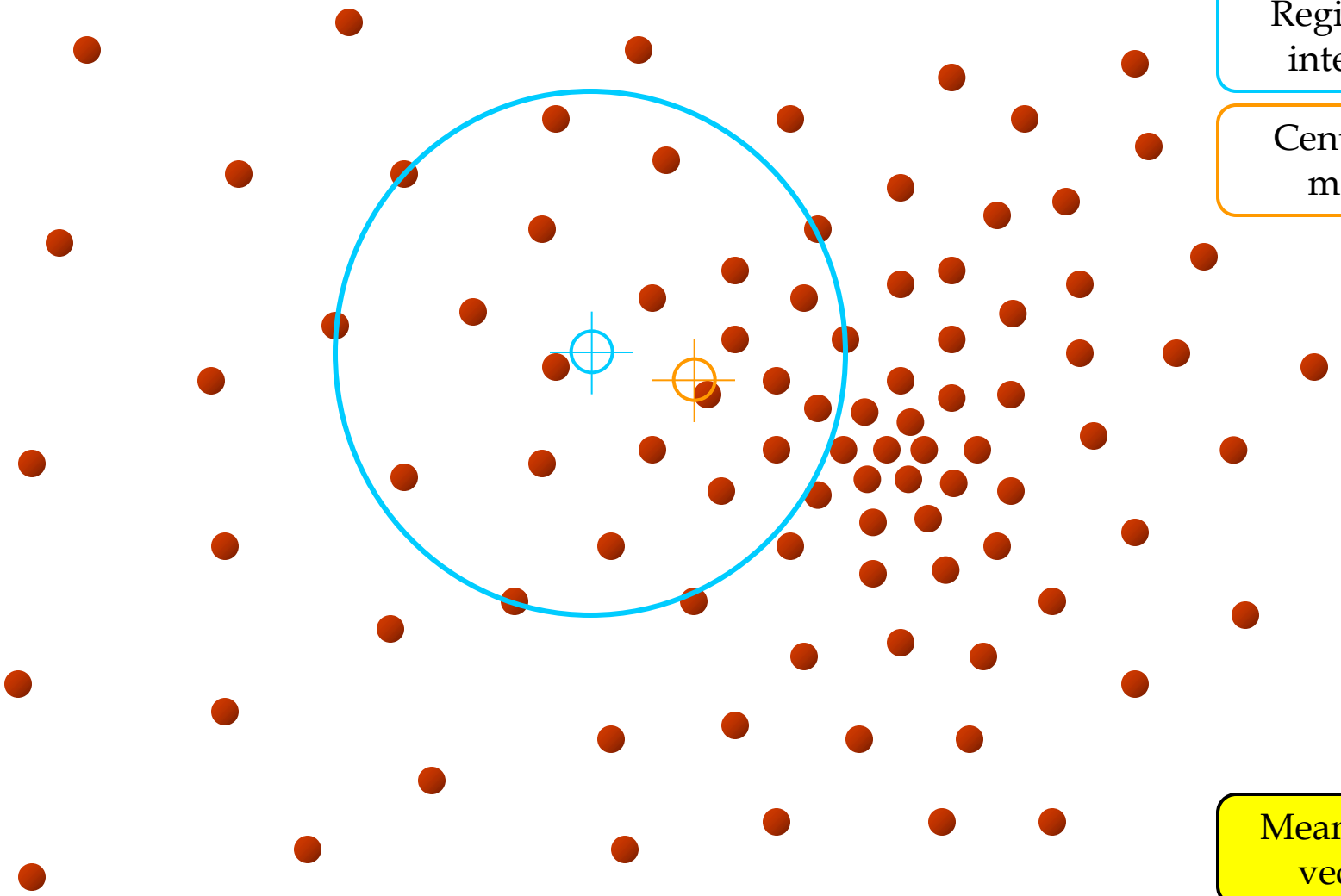
Mean shift – intuition



Objective : Find the densest region

Distribution of identical billiard balls

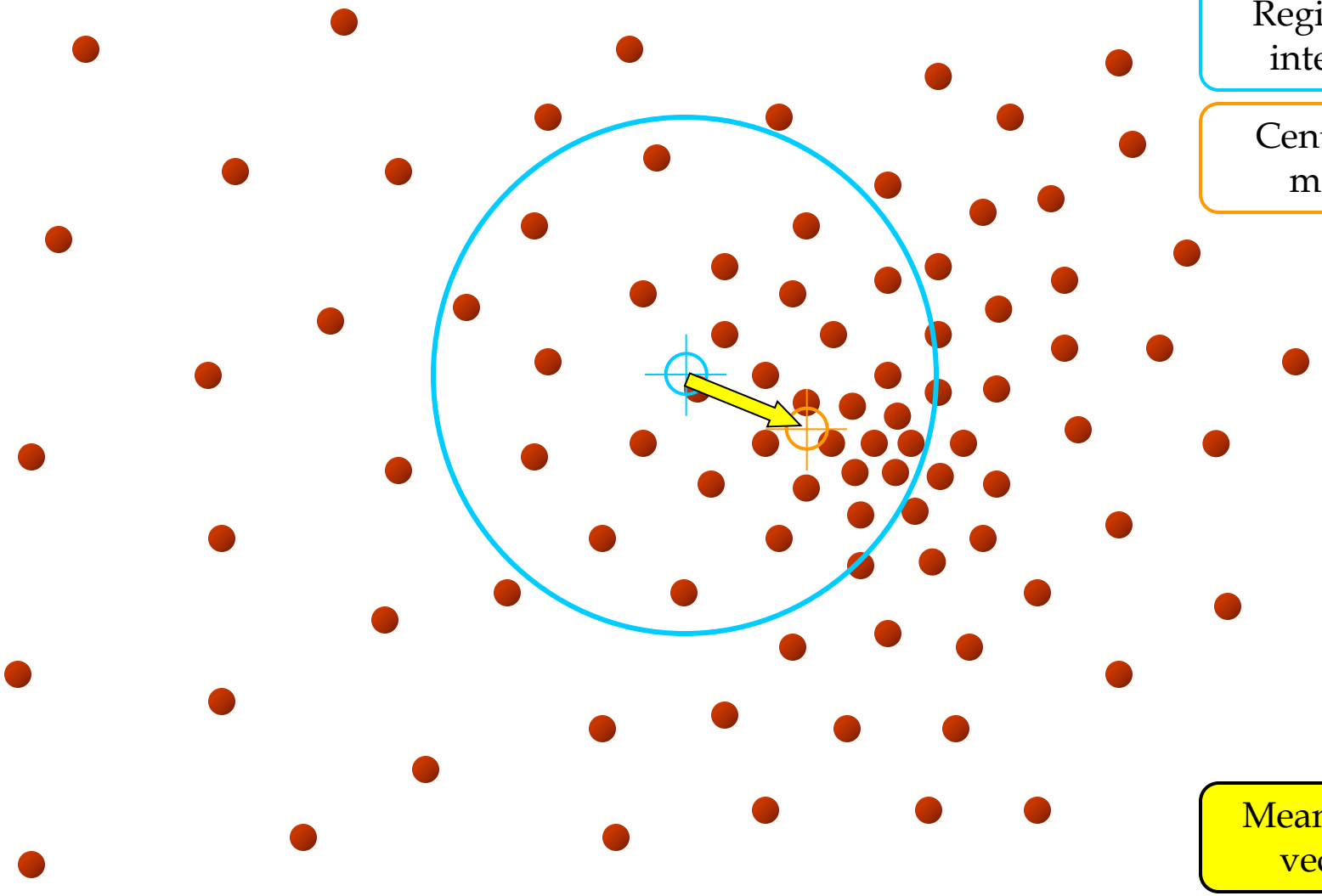
Mean shift – intuition



Objective : Find the densest region

Distribution of identical billiard balls

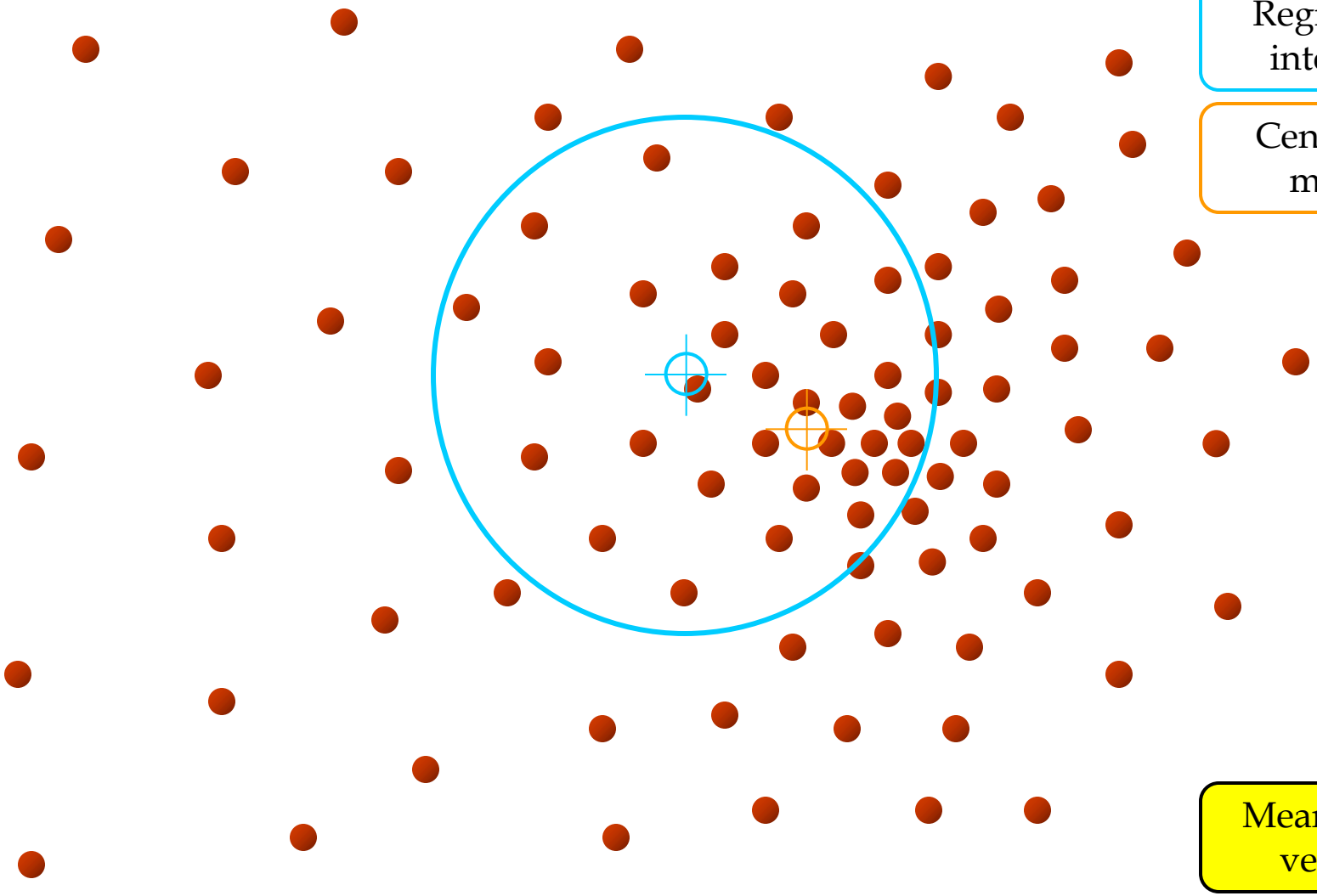
Mean shift – intuition



Objective : Find the densest region

Distribution of identical billiard balls

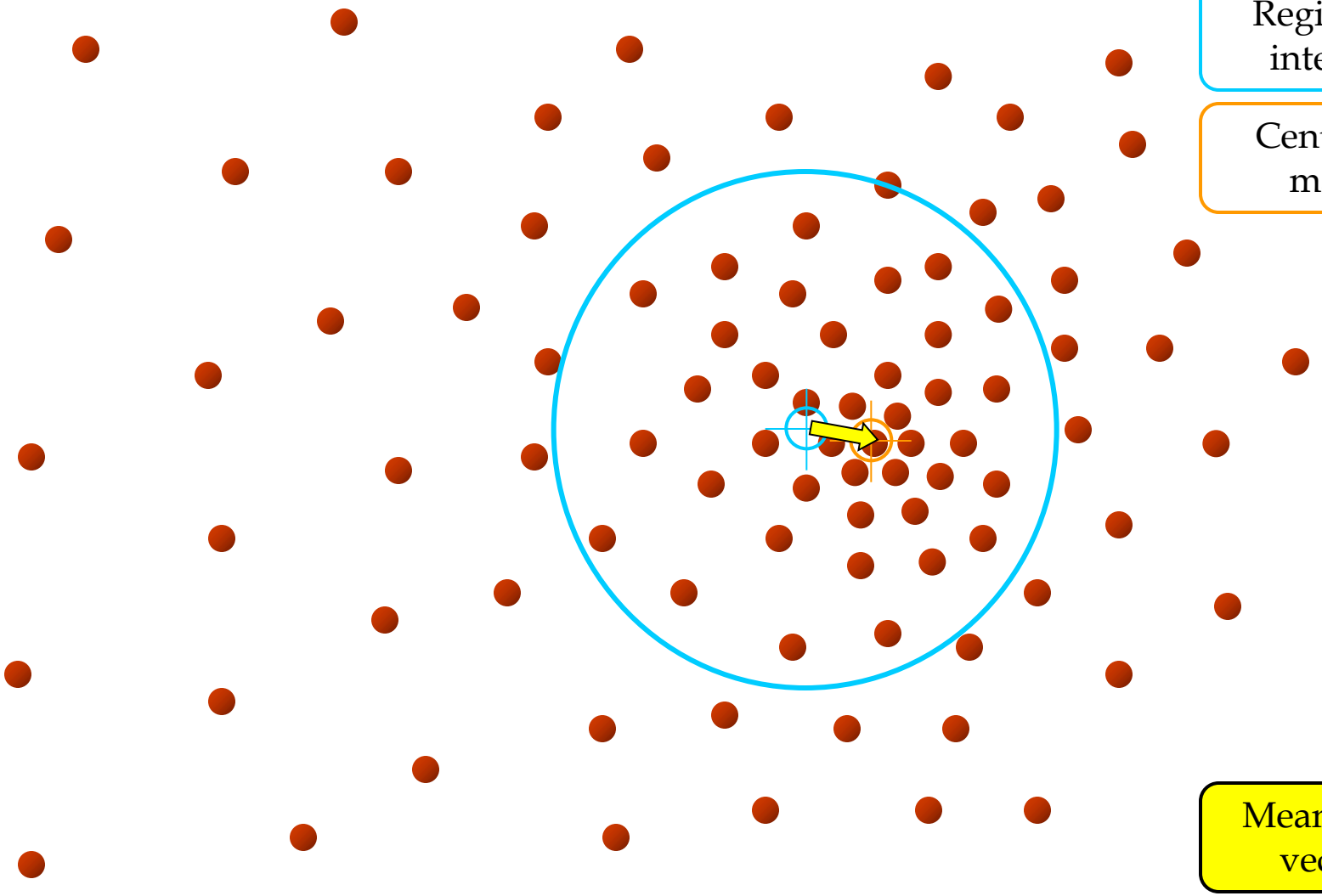
Mean shift – intuition



Objective : Find the densest region

Distribution of identical billiard balls

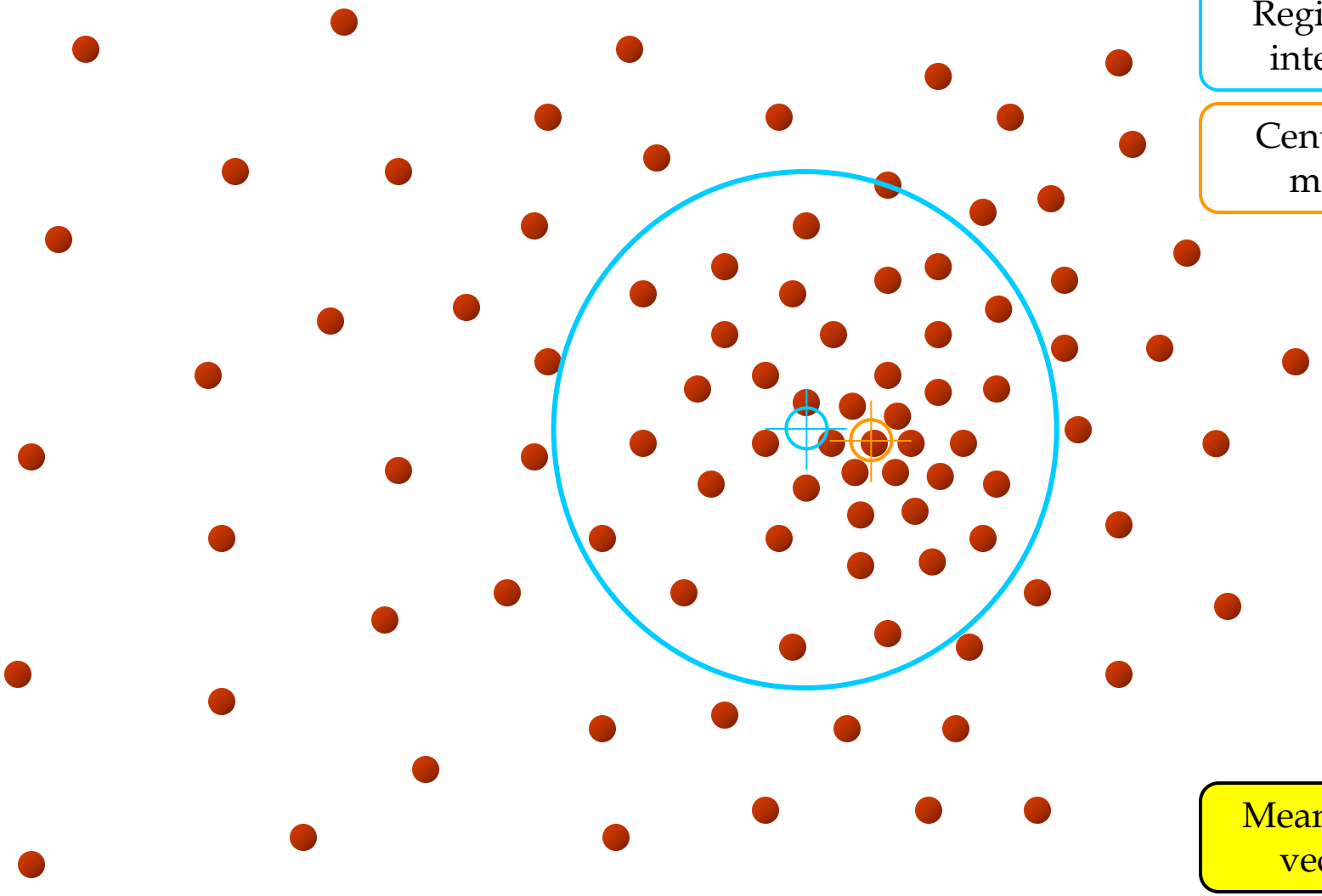
Mean shift – intuition



Objective : Find the densest region

Distribution of identical billiard balls

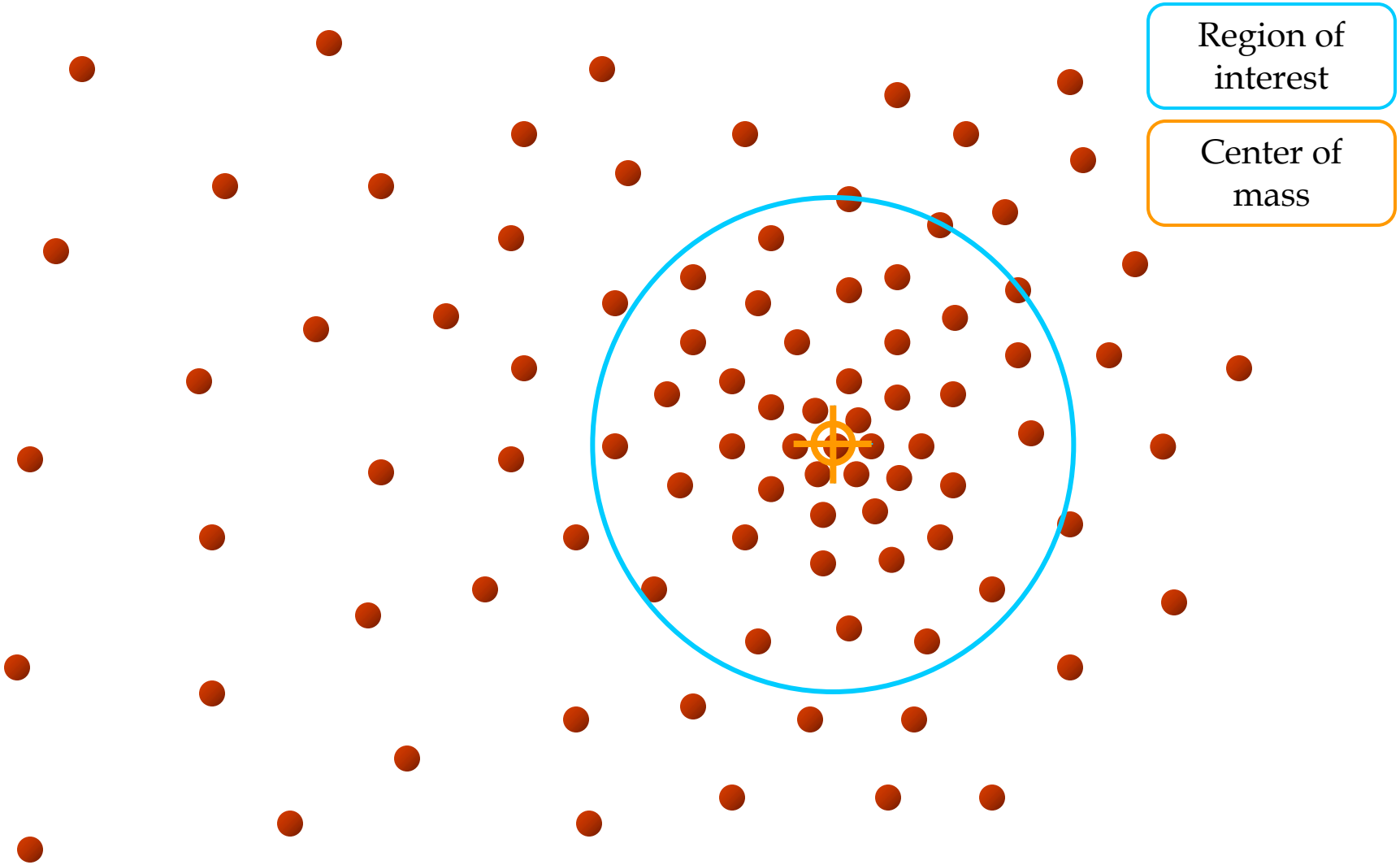
Mean shift – intuition



Objective : Find the densest region

Distribution of identical billiard balls

Mean shift – intuition



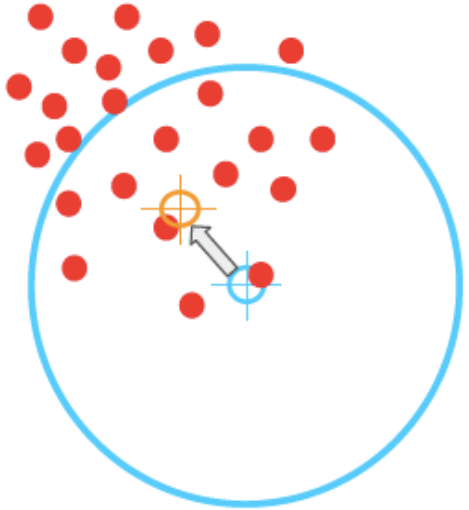
Objective : Find the densest region

Distribution of identical billiard balls

Mean shift

- Mean shift is a procedure for locating the maxima of a density function given discrete data sampled from that function (mode-seeking algorithm)

- Algorithm:
  - Input: random initialize centroid and a fixed window radius *r*
  - Repeat until converge:
    - Compute the mean of points in the window
    - Shift the centroid to the new mean



- It is guaranteed to move toward the direction of maximum increase in the density

- Application: clustering, tracking, ...

Mean shift – model representation

Suppose the current mean is μ, and $\left\{ x^{(1)}, \quad x^{(2)}, \quad \ldots, \quad x^{(m)} \right\}$ be the data that are in the window radius r, the mean shift vector is given by:

$$m_r(\mu) = \frac{\sum_{i=1}^{m} k(\mu, x^{(i)}) x^{(i)}}{\sum_{i=1}^{m} k(\mu, x^{(i)})} - \mu$$

Kernel function

Mean shift procedure:

- Compute the mean shift vector $m_r(\mu)$
- Shift to the new mean $\mu := \mu + m_r(\mu)$

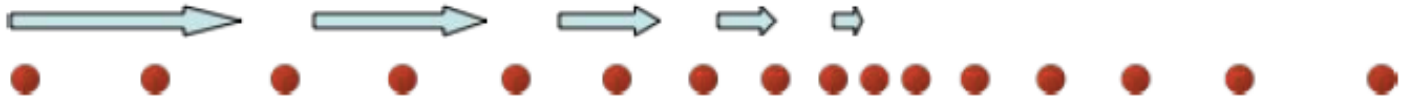
Common kernels

- Flat kernel: $k(x^{(i)}, x^{(j)}) = \begin{cases} 1 & if \ \left\| x^{(i)} - x^{(j)} \right\| \le h \\ 0 & otherwise \end{cases}$
- Gaussian kernel: $k(x^{(i)}, x^{(j)}) = \exp(-\frac{\left\| x^{(i)} - x^{(j)} \right\|^2}{2\sigma^2})$
- …

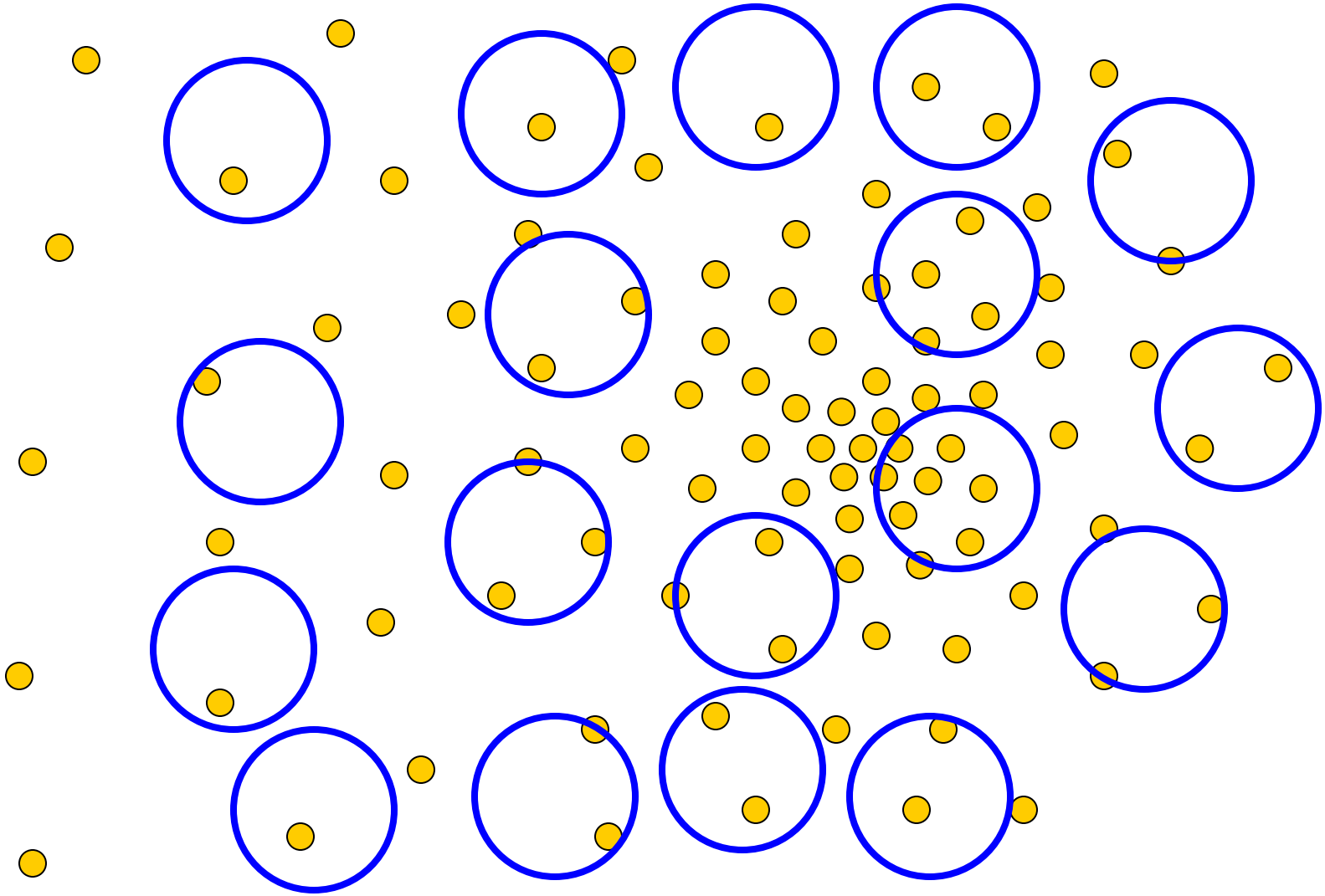
Properties of mean shift

- Adaptive gradient ascent
  - Automatic convergence speed - the mean shift vector size depends on the gradient itself
  - Near maxima, the steps are small and refined

- Convergence is guaranteed for very small steps only



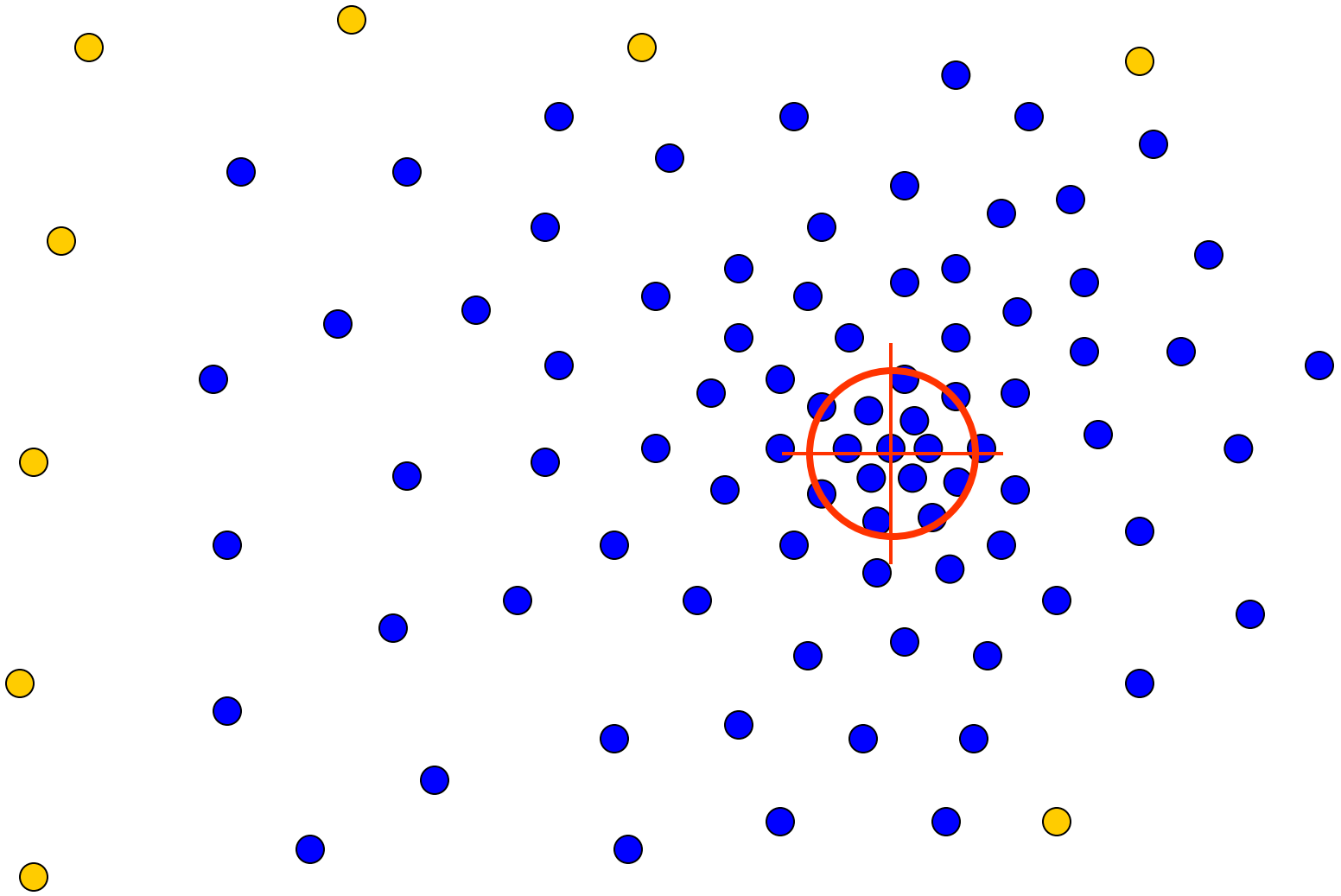
Mean shift clustering - intuition



Tessellate the space with windows

Run the procedure in parallel

Mean shift clustering - intuition

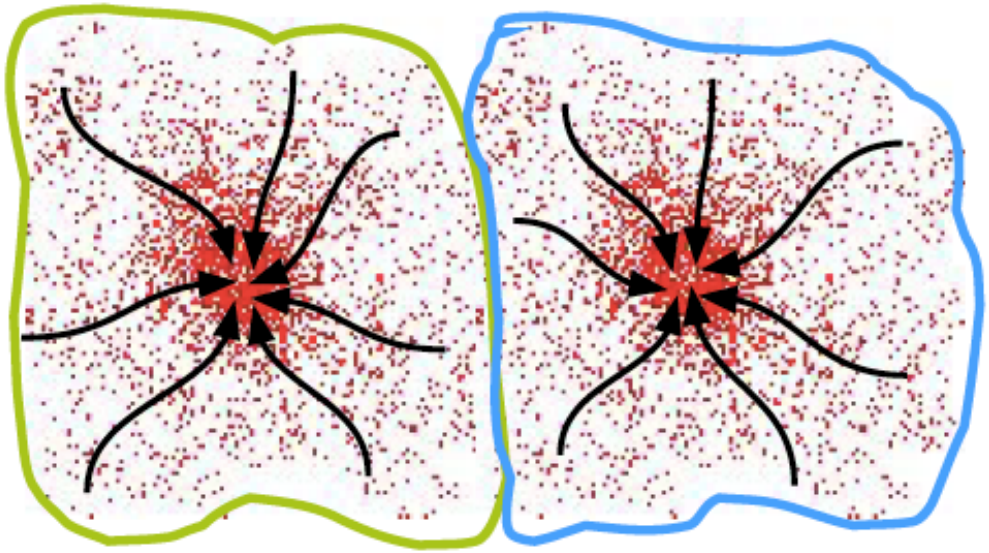


The blue data points were traversed by the windows towards the mode

Mean shift clustering

- Attraction basin: the region for which all trajectories lead to the same mode

- Clustering: all data points in the attraction basin of a mode



Mean shift clustering

Algorithm

- Starting on the data points, run mean shift procedure to find the stationary points of the density function
- Prune those points by retaining only the local maxima

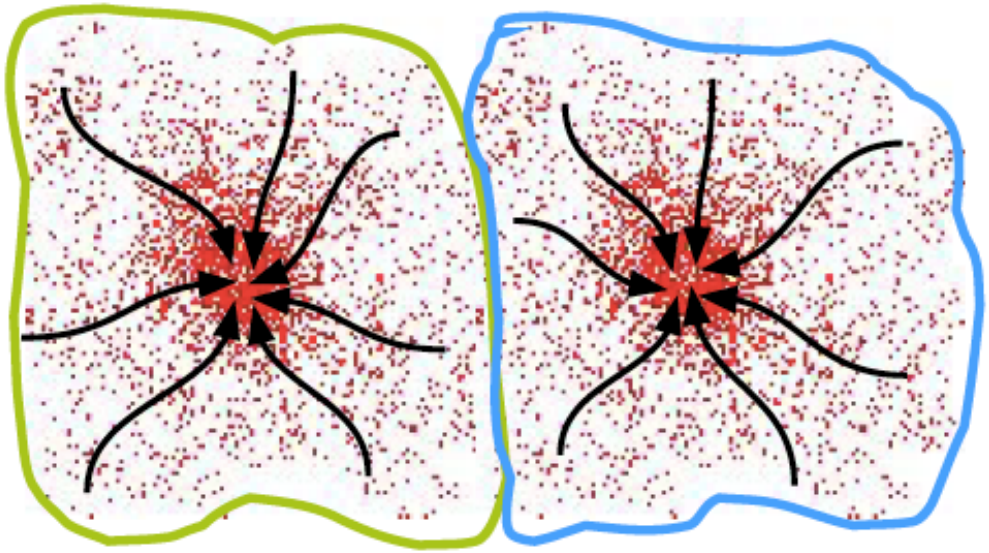
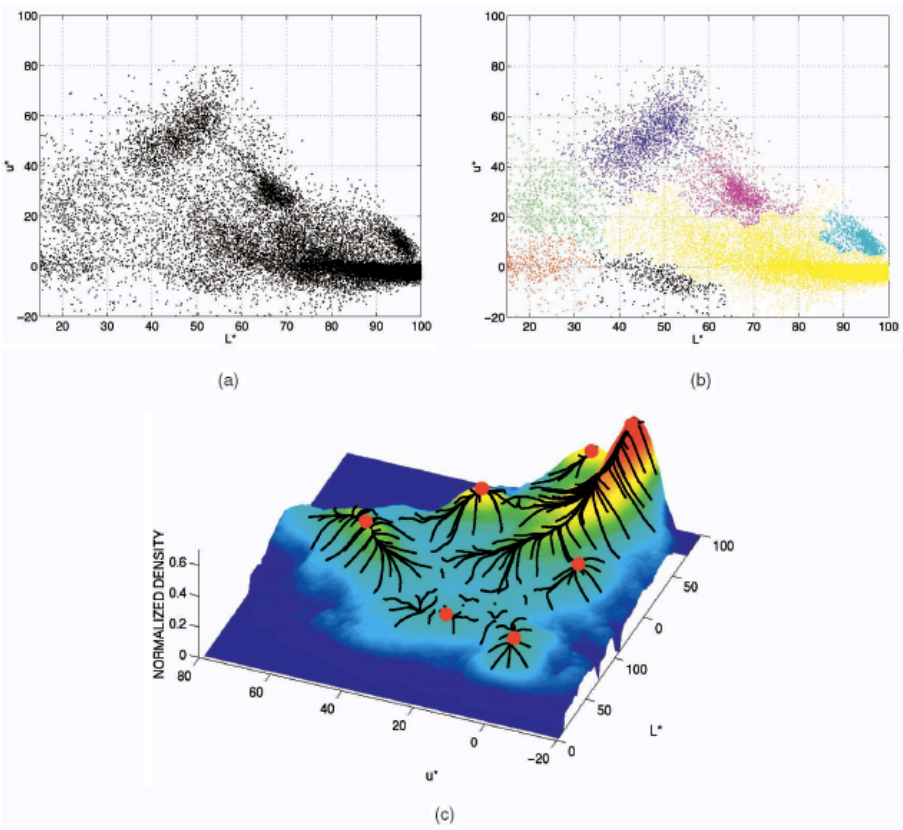
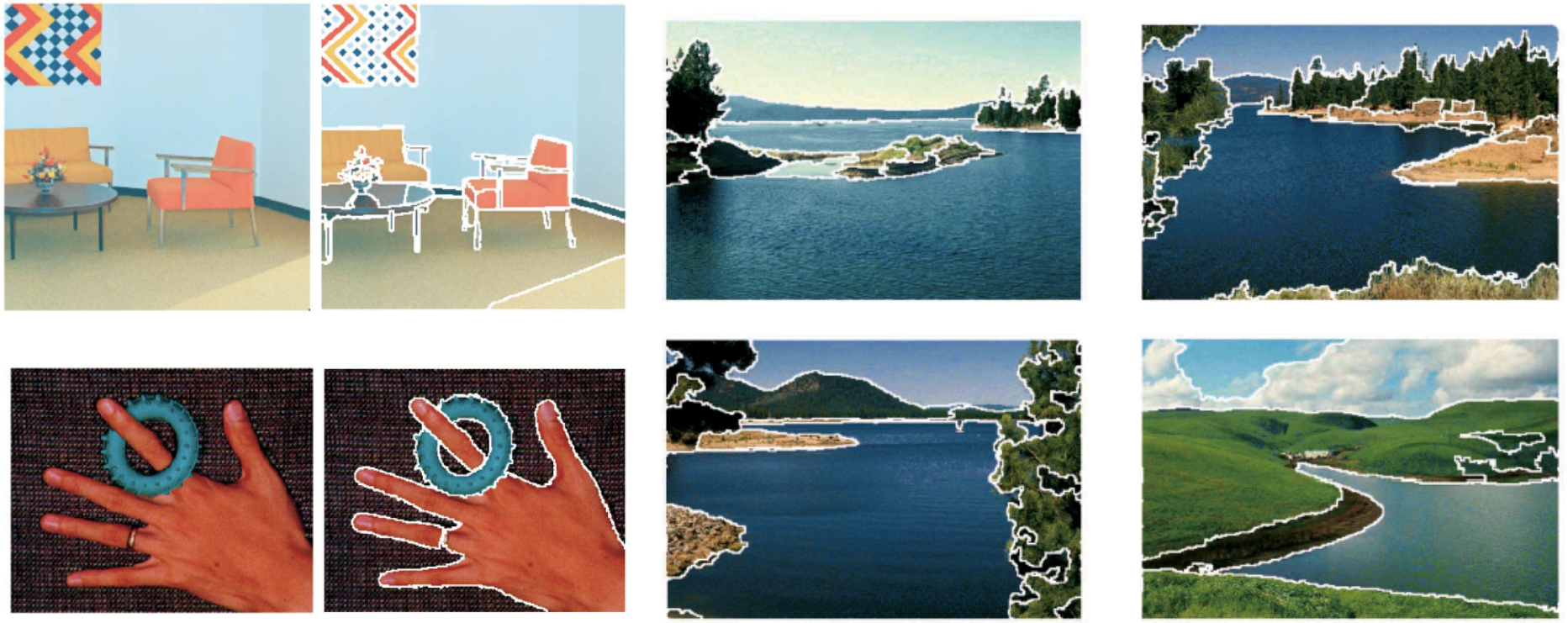


Image segmentation by mean shift clustering

- Find features (color, gradients, texture, etc.)
- Initialize windows at individual pixel locations
- Perform mean shift for each window until converge
- Merge windows that end up the same mode

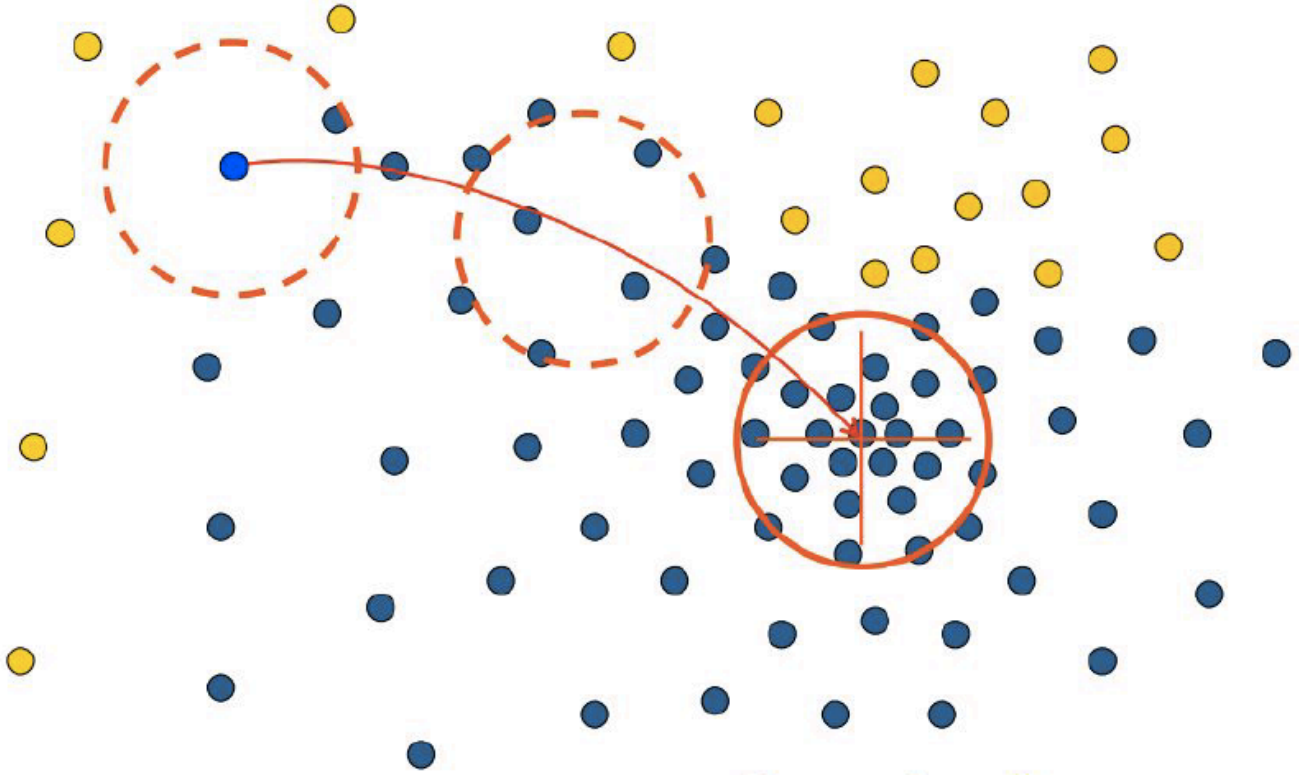


Segmentation results



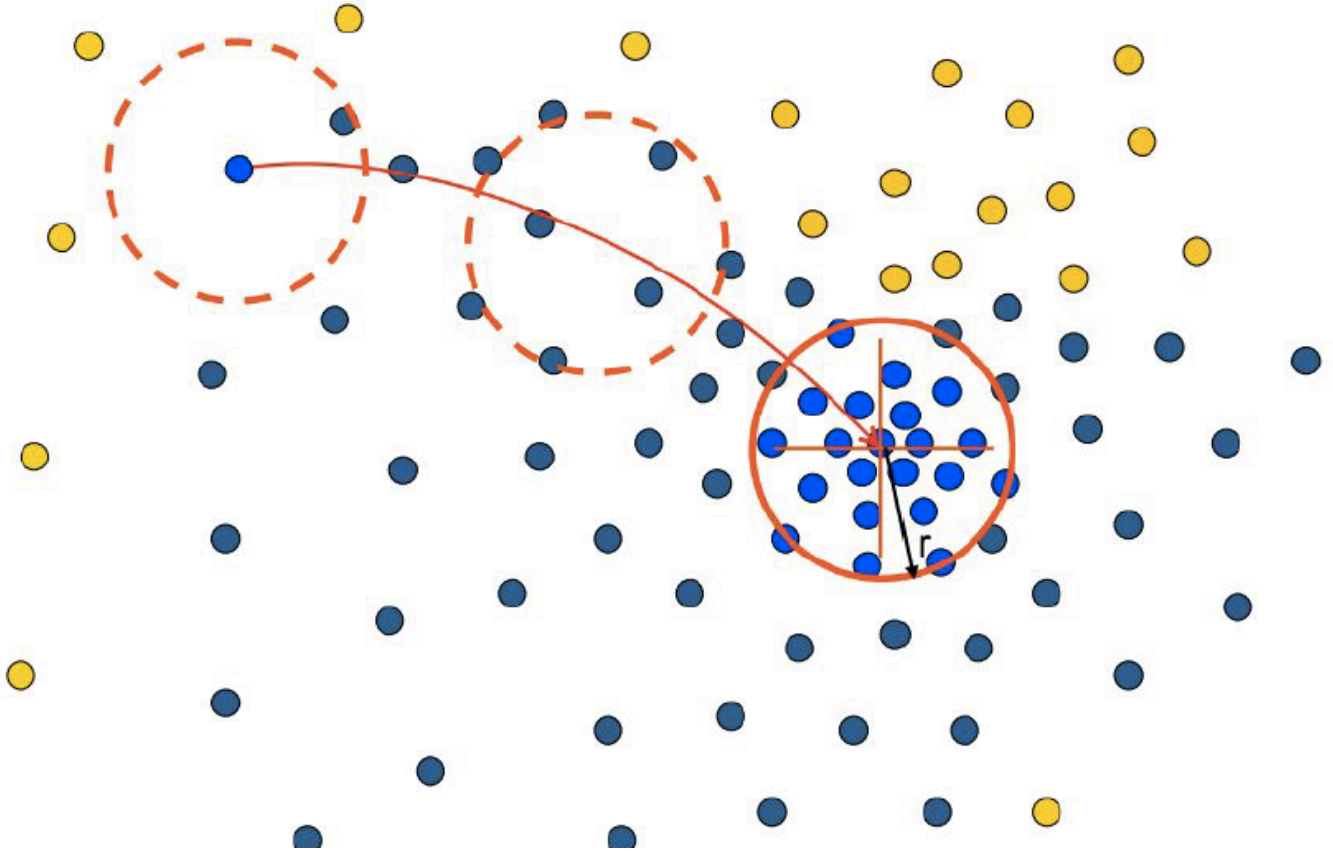
Results from: Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

Problem: computational intensive



- Need to shift many windows
- Many computations will be redundant

Speedups



- Assign all points within radius r of end point to the mode

Summary of mean shift clustering

Pros:

- General, application-independent tool, simple to implement
- Model-free, doesn’t assume any prior shape on data clusters
- Parameters free, only window radius
- Doesn't need to select number of cluster
- Robust to outliers

Cons:

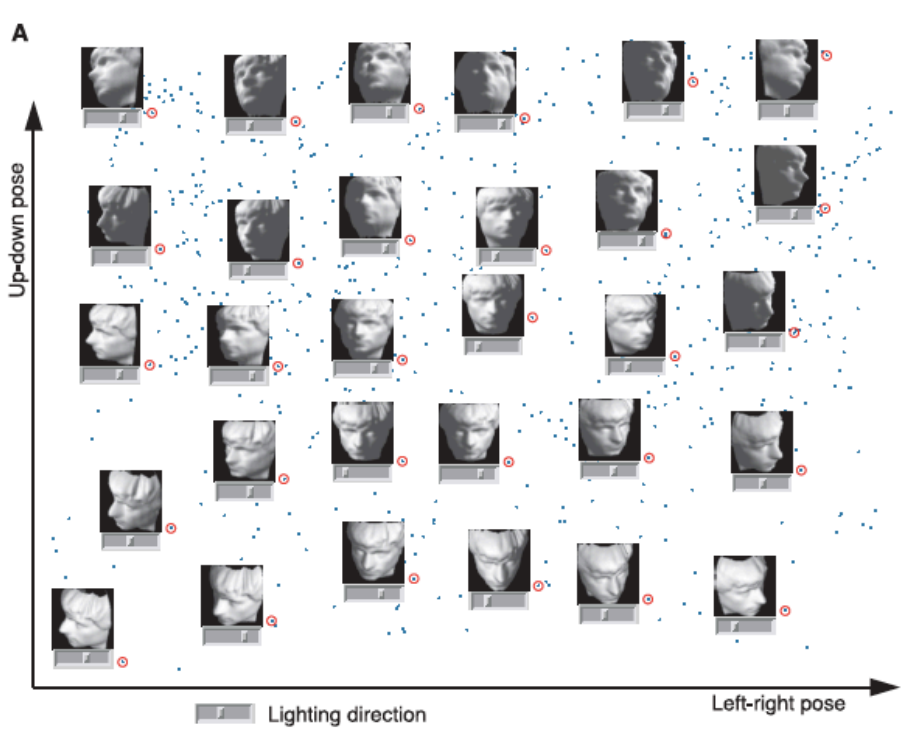
- Output depends on window radius, which is not trivial
  - Inappropriate window size can cause modes to be merged, or generate additional “shallow” modes
- Computational intensive

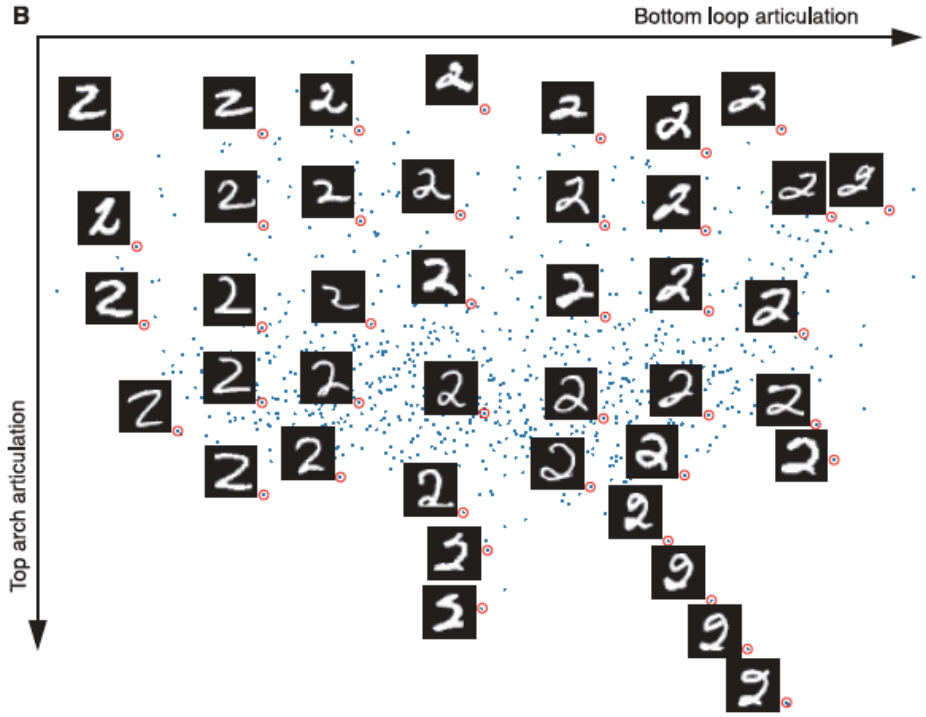
Curse of dimensionality

- Data become sparse with increase of dimensionality
- Redundant features, more noise
- Hard to interpret and visualize
- Hard to store and process data (computationally challenging)

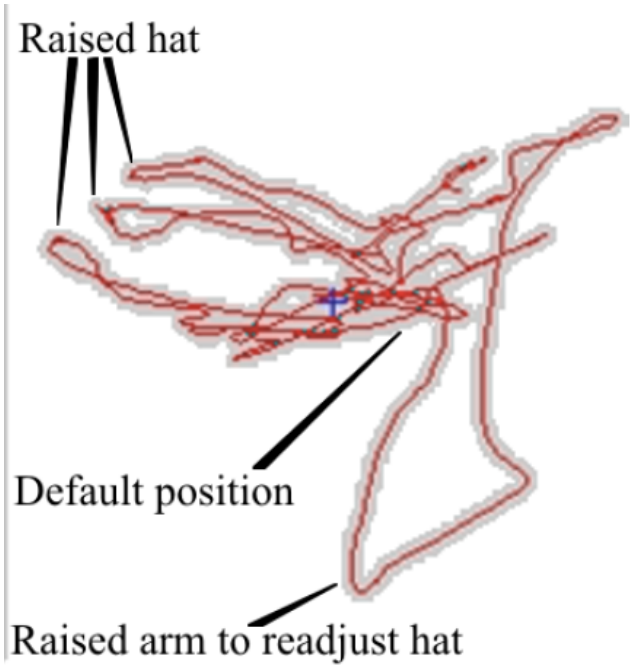
Dimensionality reduction

Given a set of $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ with n features $(m << n)$, dimensionality reduction aims to find a set of with k features, where $k << n$, and preserve some properties of the original data set, such as variance, distance, etc





Dimensionality reduction



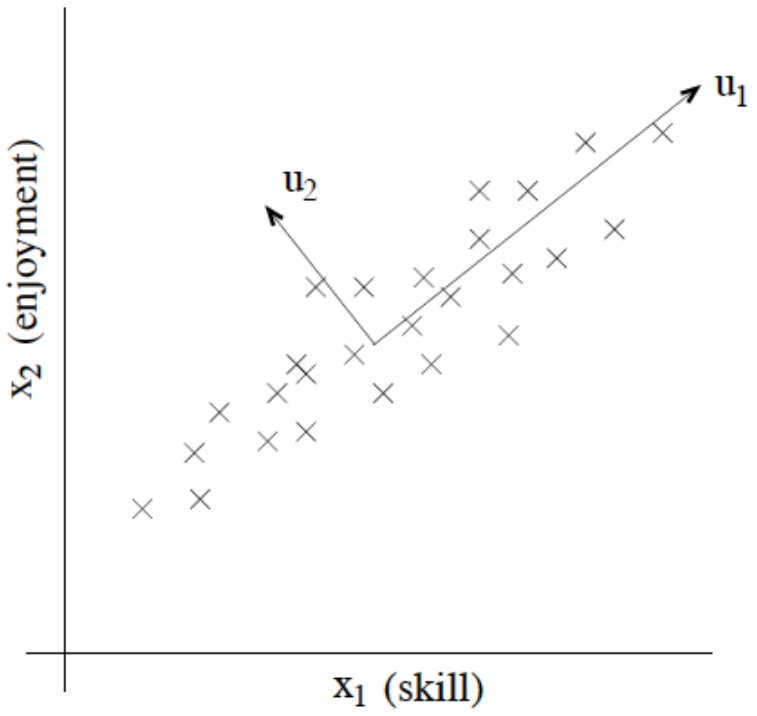
Dimensionality reduction

- Dimensionality reduction can be applied:
  - Compression
  - Visualization
  - Face detection
  - ...

- Dimensionality reduction algorithms can be divided into:
  - Linear: PCA, Locality Preserving Projections (LPP), …
  - Non-linear: Isometric feature mapping (Isomap), Laplacian Eigenmaps (LE), t-Distributed Stochastic Neighbor Embedding (t-SNE), ...

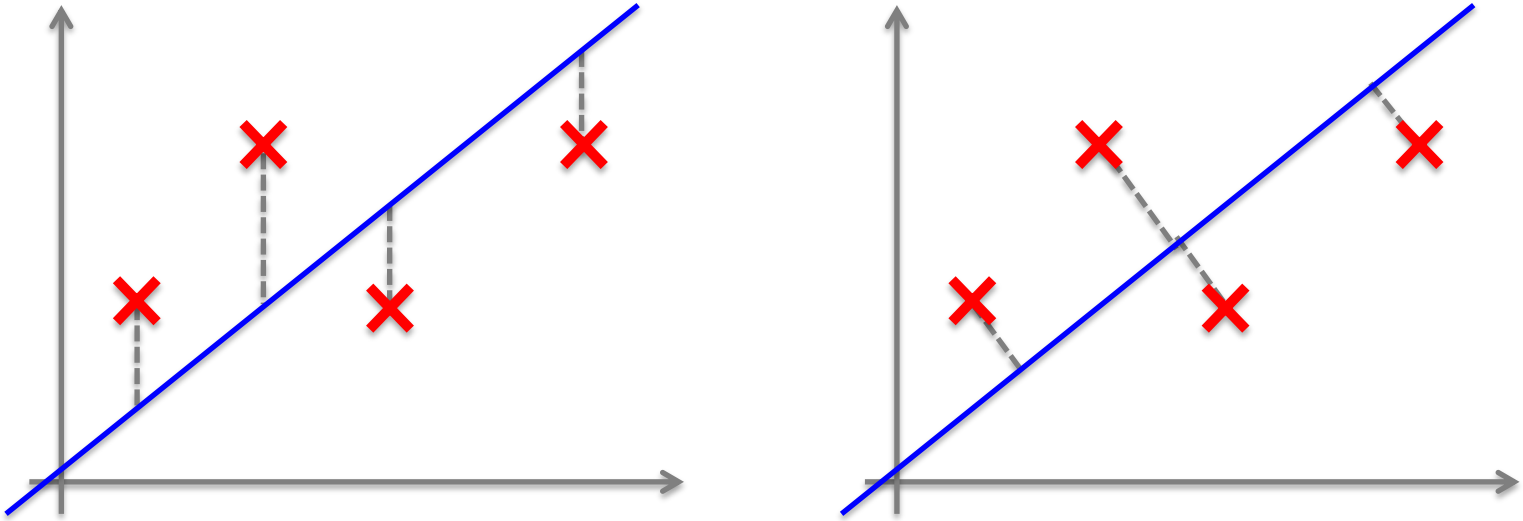
PCA – intuition

- Consider a dataset from a survey of pilots for radio-controlled helicopters, where x_1 is the pilot skill, and x_2 captures how much he/she enjoys flying
- The two attributes are strongly correlated, i.e., truly enjoy flying become good pilots
- We try to find some diagonal axis capturing the intrinsic piloting "karma" of a person, with only a small amount of NOISE lying off this axis



PCA

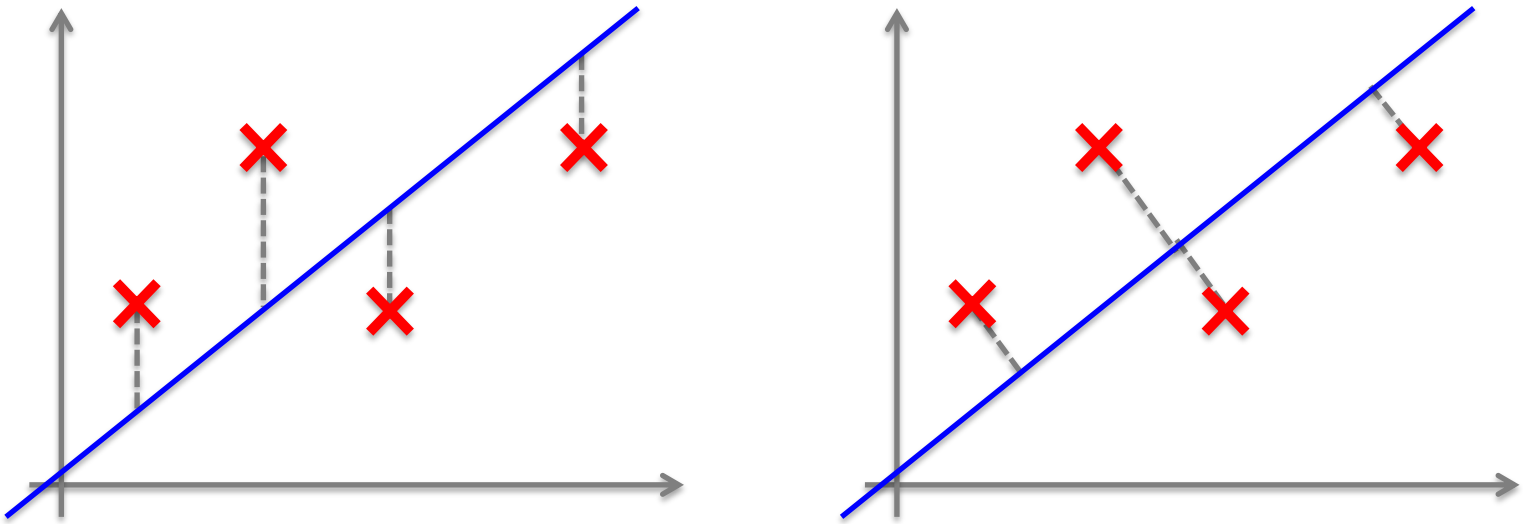
Objective: to reduce from n-dimension to k-dimension, PCA aims to find k unit vectors $u^{(1)},u^{(2)},\ldots,u^{(k)}$ onto which to project the data, so as to minimize the projection error



Which one is projection?

PCA

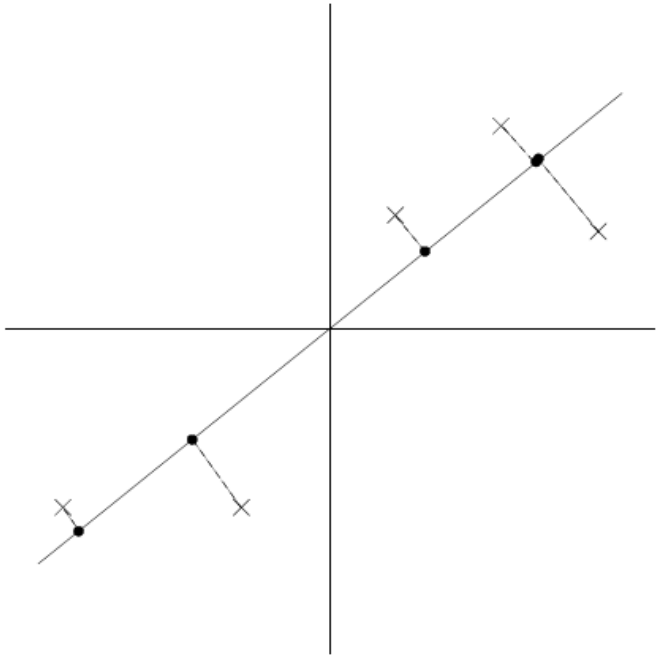
Objective: to reduce from n-dimension to k-dimension, PCA aims to find k unit vectors $u^{(1)},u^{(2)},\ldots,u^{(k)}$ onto which to project the data, so as to minimize the projection error



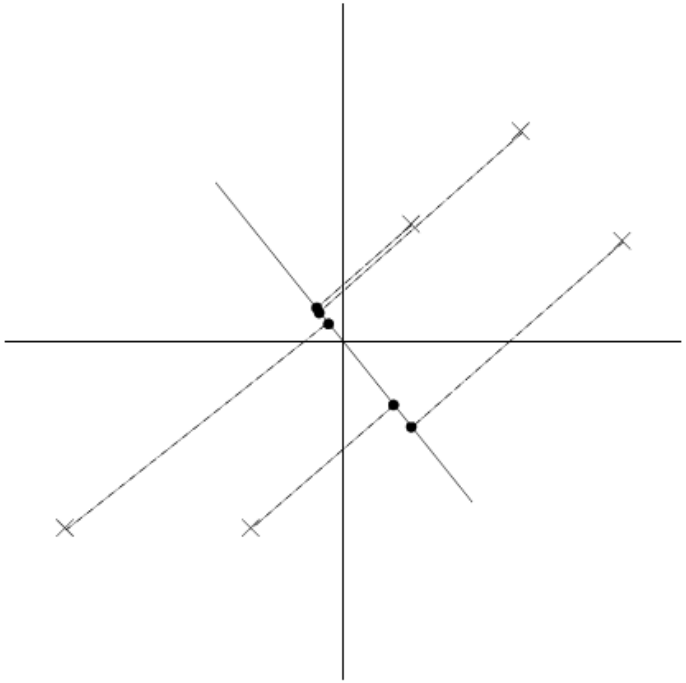
Which one is projection?

Note the difference between PCA and linear regression!

Q: which project is better?



Large variance, small projection error



Small variance, large projection error

Data preprocessing

- Normalize mean and variance
  1. Let $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$
  2. Replace each $x^{(i)}$ with $x^{(i)} - \mu$
  3. Let $\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)})^2$
  4. Replace each $x_j^{(i)}$ with $x_j^{(i)}/\sigma_j$

- Step 3 an 4 may be omitted if we have a prior knowledge that the different features are all on the same scale

PCA

- Optimization objective:

$$\max_u \frac{1}{m}\sum_{i=1}^{m}((x^{(i)})^T u)^2 \quad \text{s.t.} \|u\| = 1$$

$$\max_u u^T \underbrace{\left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)}(x^{(i)})^T\right)}_{\Sigma} u \quad \text{s.t.} \|u\| = 1$$

- Using Lagrange multiplier λ, one may solve the following eigenvalue problem:

$$\Sigma u = \lambda u$$

- Extend the first principal component to k principal components:

$$\Sigma U = \lambda U$$

PCA

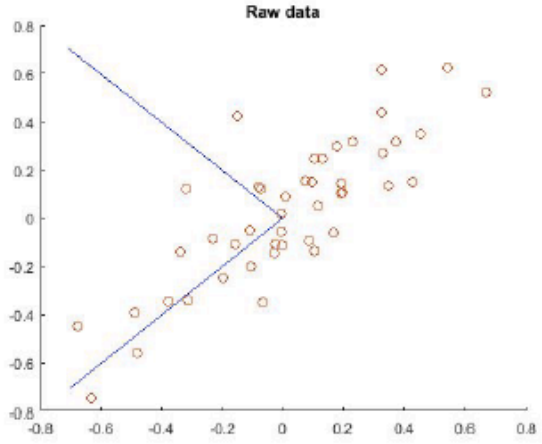
- Algorithm:
  - Preprocess data, i.e., zero mean and scale variance if need
  - Compute the covariance matrix of normalized data
  - Compute the eigenvectors and eigenvalues of the covariance matrix
  - The first top k eigenvectors correspond to the first k principal components
  - Project the data into the k-dimensional subspace

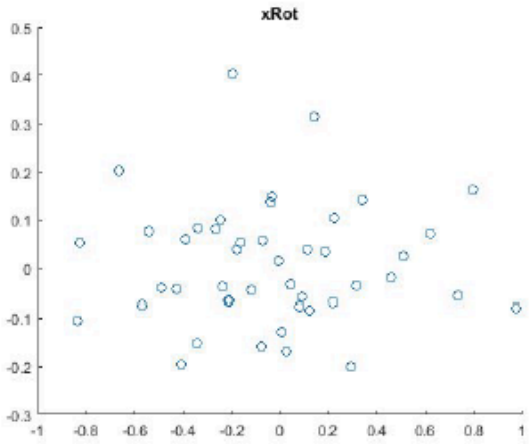
Reconstruction from compressed representation

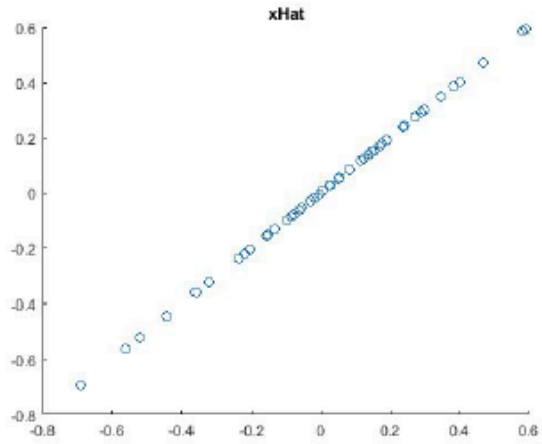
$$x_{approx}^{(i)} = UU^T x^{(i)}$$

xHat = u(:,1:k)*u(:,1:k)' * x;

Note: keep mean, variance, and eigenvalue for future use!







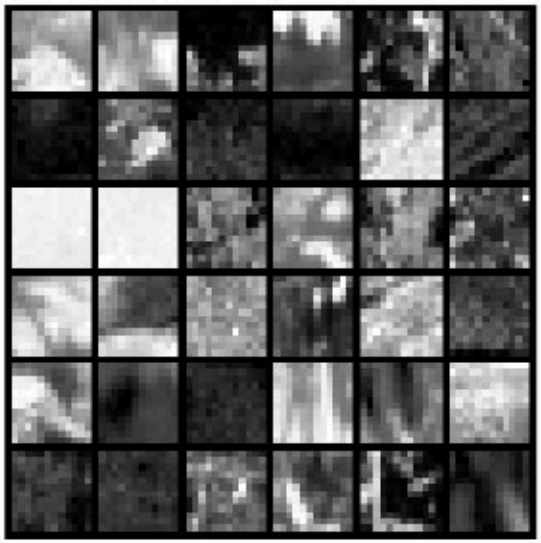
k = 1

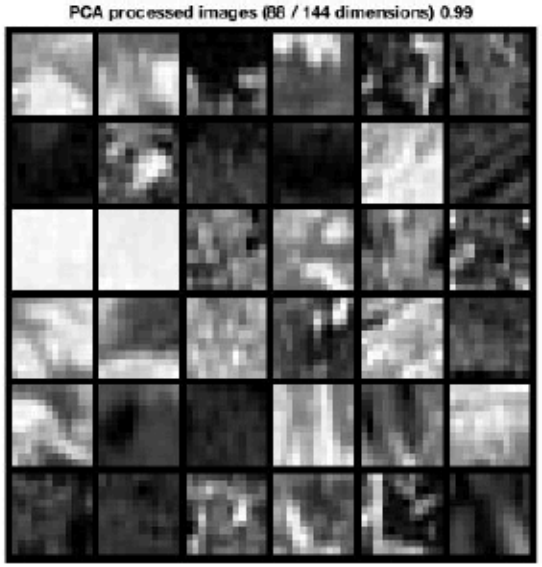
Number of components k to retain

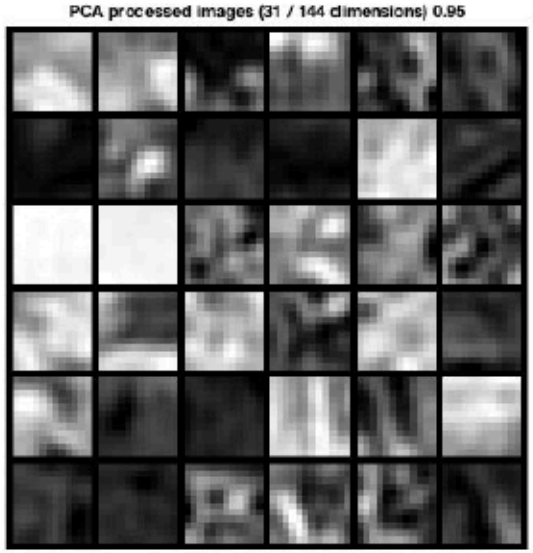
- If k is too large, we won't be compressing the data much
- if k is too small, we may lose too much information
- To decide how to set k, we will look at the "percentage of variance retained" for different values of k
- Let $\lambda_1, \lambda_2, \cdots, \lambda_n$ be the eigenvalues of covariance matrix (sorted in descending order), if we retain k principal components, the percentage of variance retained is given by:

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{i=1}^{n} \lambda_i} \geq \delta$$

Number of components k to retain







PCA summary

- Visualization ($k = 2$ or 3)
- Compression (k is obtained by computing the percentage of variance retained)
  - Reduce memory/disk needed to store data
  - Reduce correlation between features
  - Speed up learning algorithm
- Noise reduction - Eigenface [Turk 1991]
- Suggestion: before implementing PCA, first try running whatever you want to do with the original/raw data. Only if that doesn't do what you want, then implement PCA and consider using the projected data

t-SNE visualization of CNN code

- Extracted 4096-dimensional fc7 VGG features from 50,000 images
- Apply t-SNE to compute a 2-dimensional embedding
- t-SNE arranges images that have a similar features nearby in the embedding

Anomaly detection

- Anomalies and outliers are essentially the same thing: objects that are different from most other objects
- Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior [Chandola 2009]
- Historically, the field of statistics tried to find and remove outliers as a way to improve analyses.
- There are now many fields where the outliers/anomalies are the objects of greatest interest

Causes of anomalies

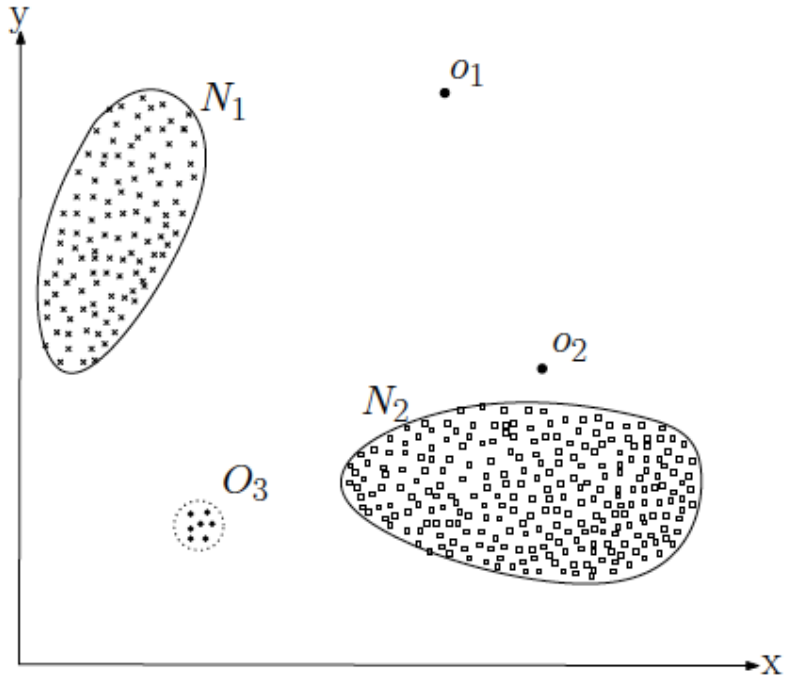
- Data from different class of object or underlying mechanism
  - Disease vs. non-disease
  - Fraud vs. non-fraud
- Natural variation
  - Tails on a Gaussian distribution
- Data measurement and collection errors

Structure of anomalies

- Point anomalies
- Contextual anomalies
- Collective anomalies

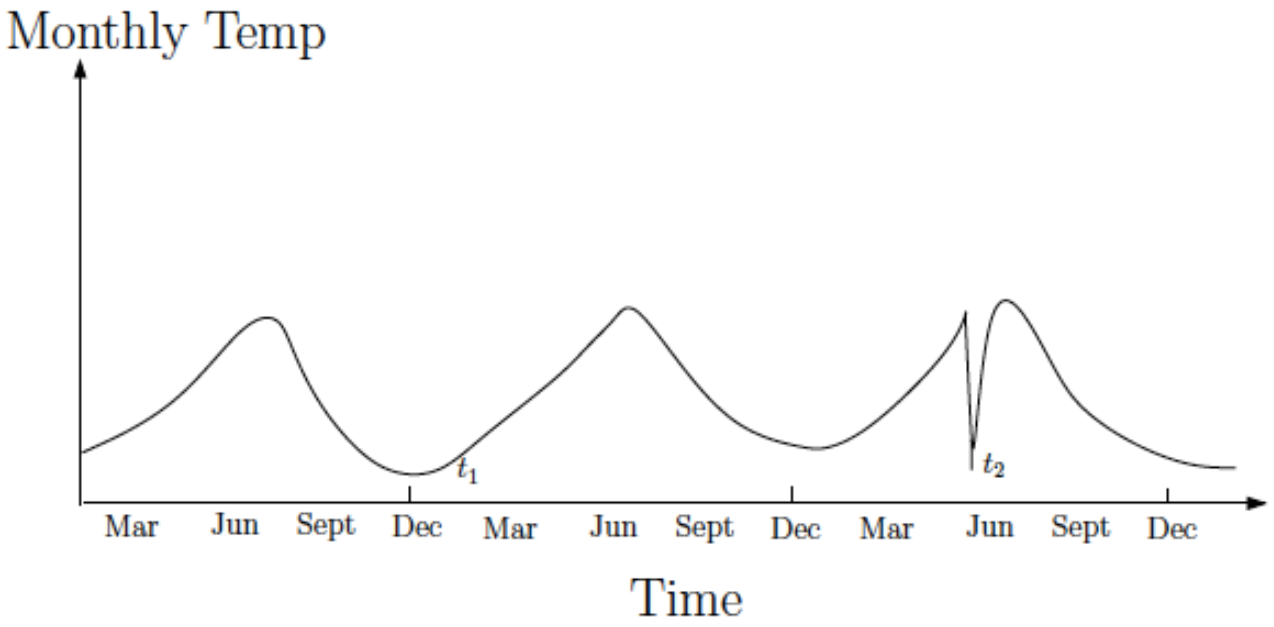
Points anomalies

- An individual data instance is anomalous with respect to the data



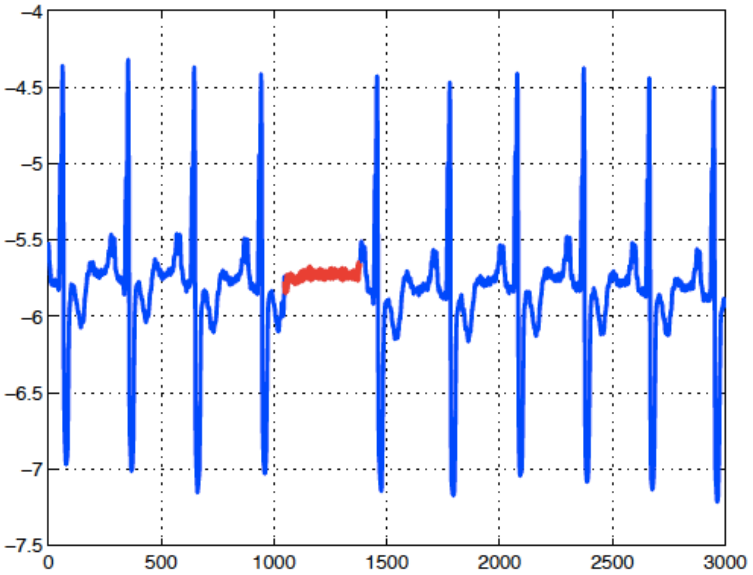
Contextual anomalies

- An individual data instance is anomalous within a context
- Requires a notion of context
- Also called “conditional anomalies”



Collective anomalies

- A collection of related data instances is anomalous
- Requires a relationship among data instances
  - Sequential data
  - Spatial data
  - Graph data
- The individual instances within a collective anomaly are not anomalous by themselves



Applications of anomaly detection

- Network intrusion
- Insurance/credit card fraud
- Healthcare informatics/medical diagnostics
- Image processing/video surveillance
- …

Fraud detection

- Detection of criminal activities occurring in commercial organizations.
- Malicious users might be:
  - Employees
  - Actual customers
  - Someone posing as a customer (identity theft)
- Types of fraud
  - Credit card fraud
  - Insurance claim fraud
  - Mobile/cell phone fraud
- Challenges
  - Fast and accurate real-time detection
  - Misclassification cost is very high

Healthcare informatics

- Detect anomalous patient records
  - Indicate disease outbreaks, instrumentation errors, etc.

- Key challenges
  - Only normal labels available
  - Misclassification cost is very high
  - Data can be complex: spatial-temporal

Image processing

- Detecting outliers in an image monitored over time
- Detecting anomalous regions within an image
- Used in
  - video surveillance
  - satellite image analysis
- Key Challenges
  - Detecting collective anomalies
  - Data sets are very large
  - Adaptive to concept drift and real-time detection

Use of data labels in anomaly detection

- Supervised anomaly detection
  - Labels available for both normal data and anomalies
  - Similar to classification with high class imbalance

- Semi-supervised anomaly detection
  - Labels available only for normal data

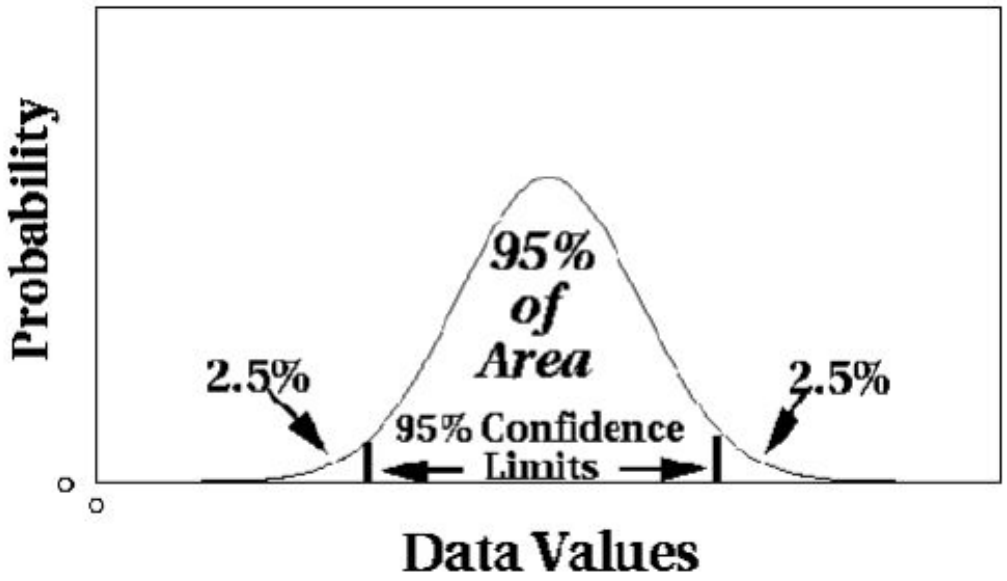
- Unsupervised anomaly detection (Common in application)
  - No labels assumed
  - Based on the assumption that anomalies are very rare compared to normal data

Unsupervised anomaly detection

- No labels available
- Based on assumption that anomalies are very rare compared to “normal” data
- General steps
  - Build a profile of “normal” behavior
    - Summary statistics for overall population
    - Model of multivariate data distribution
  - Use the “normal” profile to detect anomalies
    - Anomalies are observations whose characteristics differ significantly from the normal profile

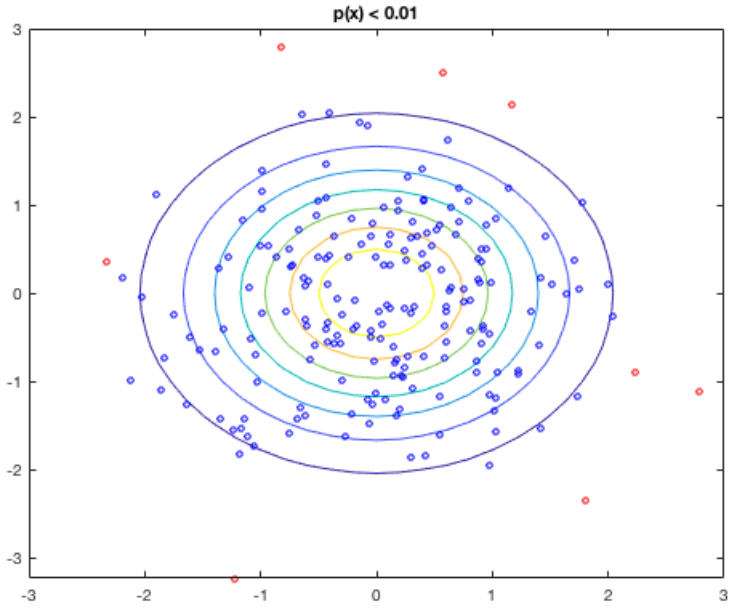
Univariate Gaussian distribution

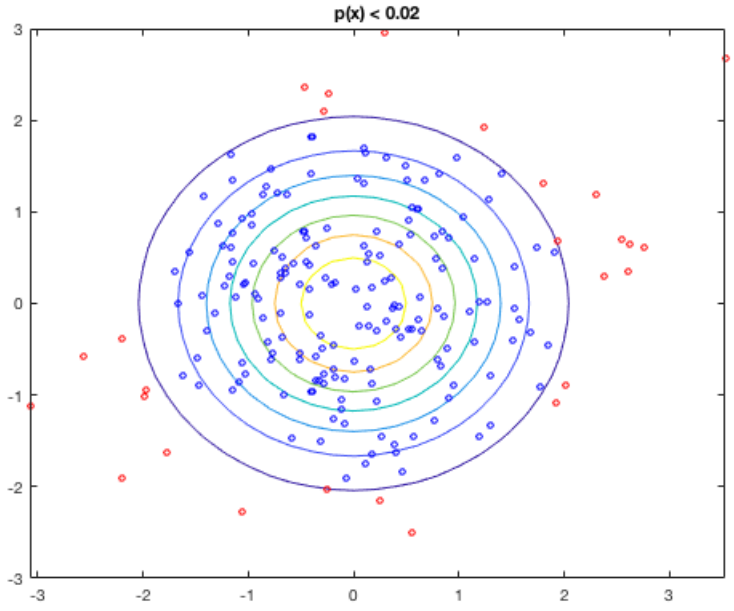
- Anomalies are defined by z-score > threshold



Multivariate Gaussian distribution

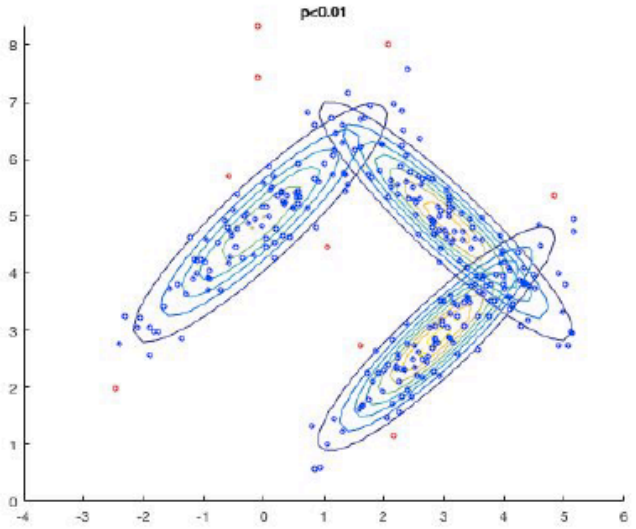
- If the distribution is well defined by multivariate Gaussian distribution
- Outliers are defined by probability < threshold or Mahalanobis distance > threshold



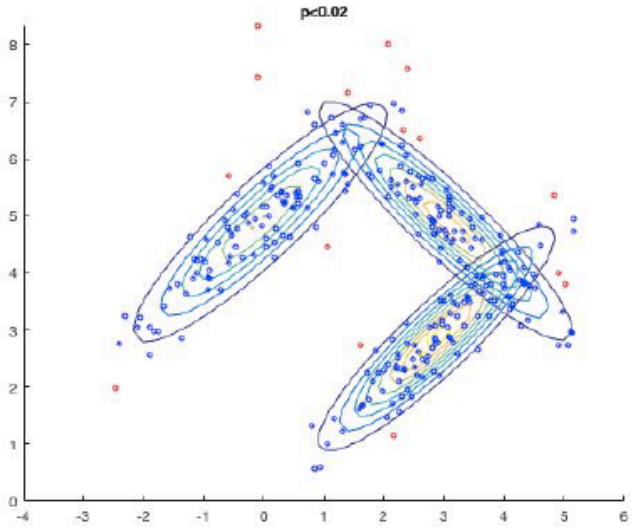


GMM

- Select number of Gaussian models k, run the GMM algorithm to obtain the optimal parameters



- For a new data, if its probability for each Gaussian model is less than a threshold, it is detected as an anomaly



Statistical anomaly detection

- Pros
  - Statistical tests are well-understood and well-validated
  - Quantitative measure of degree to which object is an outlier
- Cons
  - Data may be hard to model parametrically
    - Multiple modes
    - Variable density
  - In high dimensions, data may be insufficient to estimate true distribution

One-Class Support Vector Machine (OCSVM)

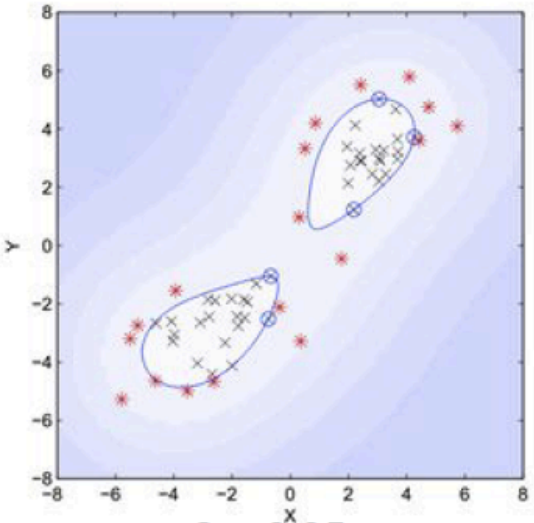
- Given a set of unlabelled train data $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$, OCSVM [Scholkopf 2001] aims to find an optimal separating function $f(x) = w \cdot \Phi(x) - \rho$ to contain most of the training data in a compact region.
- To obtain the optimal parameter w and ρ, one can solve the following quadratic programming problem:

$$\min_{w,\rho} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i - \rho$$

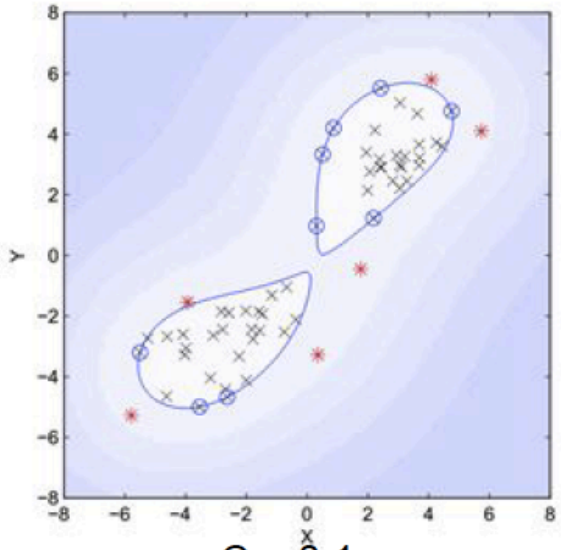
$$\text{s.t. } w \cdot \Phi(x) \geq \rho - \xi_i, \ \xi_i \geq 0$$

- OCSVM is similar to SVM, so most SVM optimization can be applied to it straightforwardly.

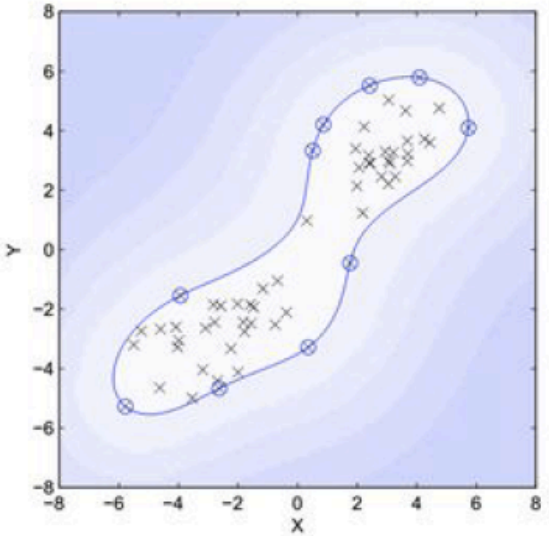
OCSVM



C = 0.05



C = 0.1



C = 0.4

OCSVM anomaly detection

- Pros:
  - Non-linear by kernel method
  - Quantitative measure of degree to which object is an outlier
- Cons:
  - Time consuming when training set is quite large
  - Hard to choose parameters, C and σ

Evaluating an anomaly detection system

- Aircraft engines motivating example: 10,000 good (normal) engines, 20 flawed engines (anomalous)
  - Training set: 6,000 good engines (y=0)
  - Validation set: 2000 good engines (y=0), 10 anomalous (y=1)
  - Test set: 2000 good engines (y=0), 10 anomalous (y=1)

- Possible evaluation metrics to choose threshold:
  - Precision/Recall
  - F1-measure
  - ROC curve

Anomaly detection vs. supervised learning

- Very small number of positive examples (y=1)
- Large number of negative examples
- Many different "types" of anomalies. Hard for any examples what the anomalies look like
- Future anomalies may look nothing like any of the anomalous

- Large number of positive and negative examples
- Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples are likely to be similar to ones in training set