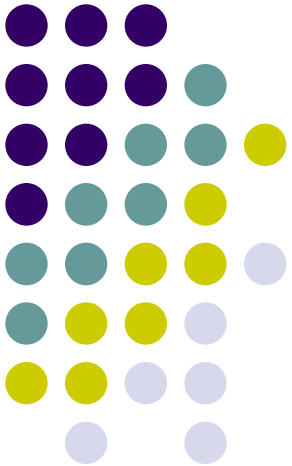
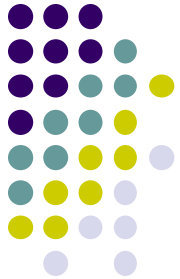
Distance and similarity (II)

Hongxin Zhang
zhx@cad.zju.edu.cn

State Key Lab of CAD&CG, ZJU
2015-03-24



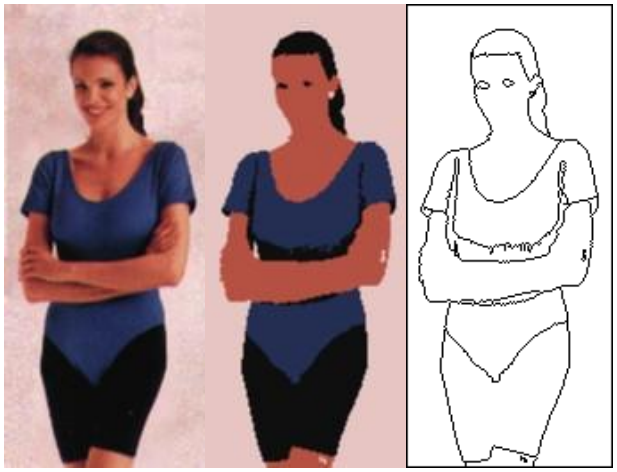
Clustering vs. Classification



- **Clustering**
  - **Instance:** $\{\mathbf{x}_i\}_{i=1}^{N}$
  - **Learn:** $<\mathbf{x}_i, t_i>$ and/or mapping from $\mathbf{x}$ to $t(\mathbf{x})$

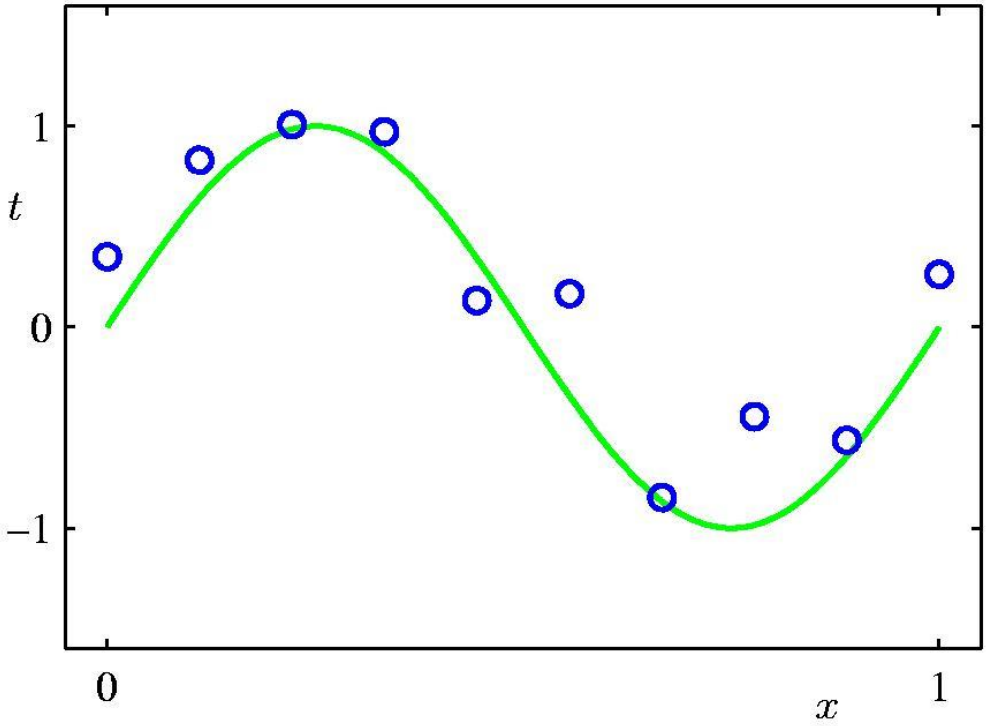
- **Classification/Regression**
  - **Instance:** $<\mathbf{x}_i, t_i>$
  - **Learn:** mapping from $\mathbf{x}$ to $t(\mathbf{x})$

Clustering and image segmentation



Mean-shift segmentation

Regression revisit: Polynomial Curve Fitting



$$\mathbf{h}(x) = \begin{pmatrix} h_0(x) \\ h_1(x) \\ \vdots \\ h_M(x) \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

$$y = \mathbf{h}(x) \cdot \mathbf{w}$$

$$\mathbf{w} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{y}$$

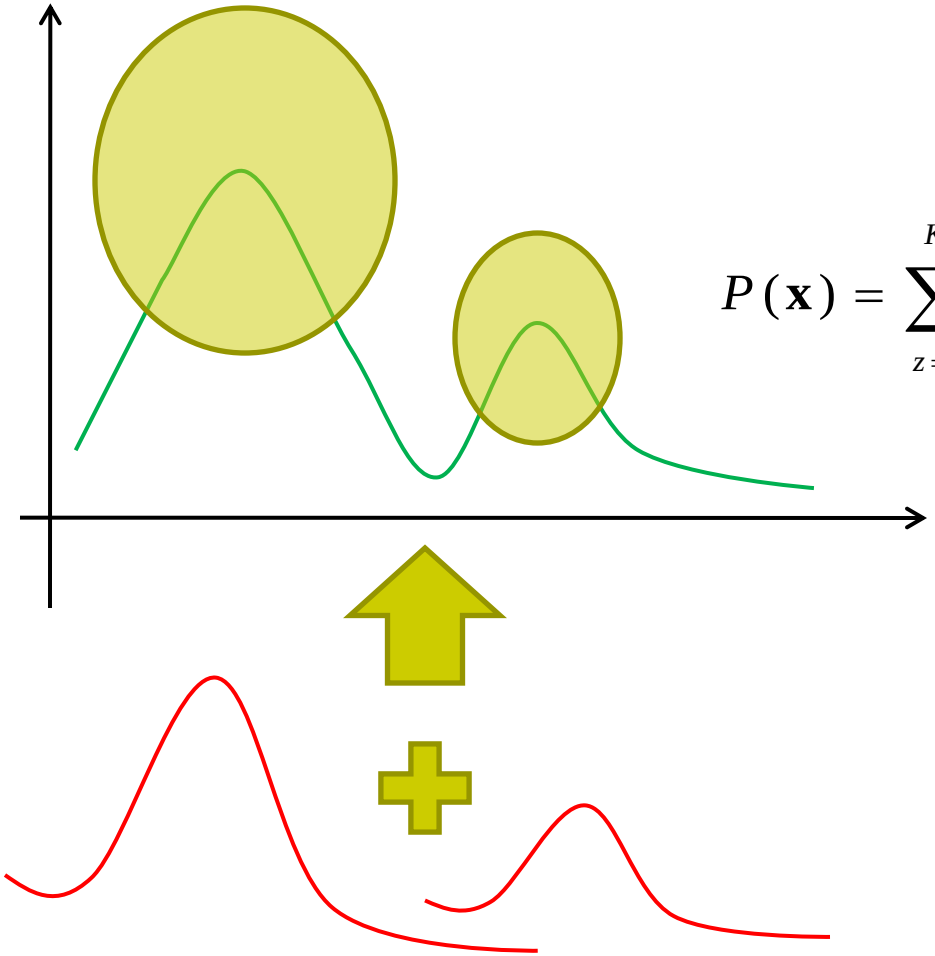
Normal equation

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

$$y(x, \mathbf{w}) = w_0 + w_1 h_1(x) + w_2 h_2(x) + \ldots + w_M h_M(x) = \sum_{j=0}^{M} w_j h_j(x)$$

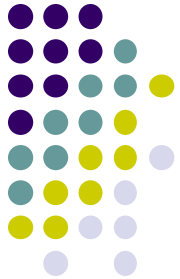
Basis function

Regression revisit: alternative approach



$$P(\mathbf{x}) = \sum_{z=1}^{K} P(Z = z \mid \pi) N(\mathbf{x} \mid \mu_z, \Sigma_z)$$

Mixtures of Gaussians



- Mixture distribution:
  - Assume *P(x)* is a mixture of K different Gaussians
  - Assume each data point, x is generated by 2-step process
    - Choose one of the K Gaussians as label z
    - Generate x according to the Gaussian $N(\mu_z, \Sigma_z)$

$$P(\mathbf{x}) = \sum_{z=1}^{K} P(Z = z \mid \pi) N(\mathbf{x} \mid \mu_z, \Sigma_z)$$

- What object function shall we optimize?
  - Maximize data likelihood

Mixtures of Gaussians (cont.)

- Multivariate Gaussian model

$$p(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\{ -\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu) \}$$

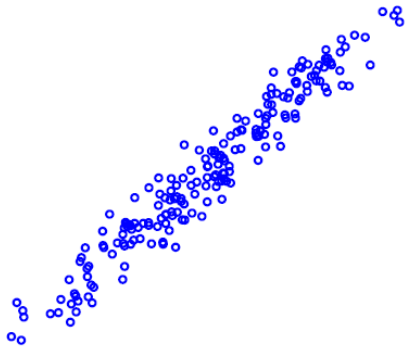
- How to generate it?

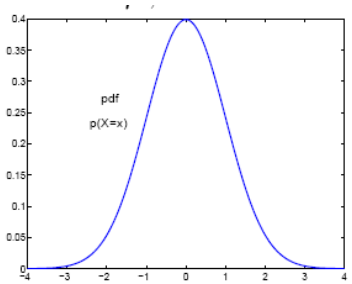
$$F_{\mu,\sigma^2}(x) = \int_{-\infty}^{x} p(z|\mu,\sigma^2)dz$$

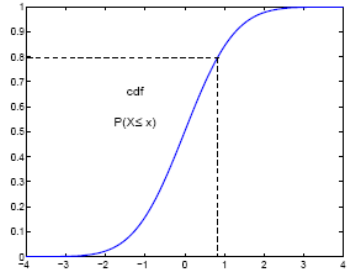
$$u \sim \text{Uniform}(0,1) \Rightarrow x = F^{-1}_{\mu,\sigma^2}(u) \sim p(x|\mu,\sigma^2)$$

$$z_i \sim p(z_i|\mu=0,\sigma^2=1), \quad \mathbf{z} = [z_1,\ldots,z_d]^T$$

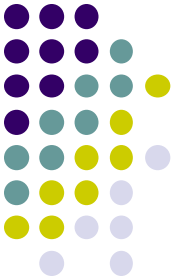
$$\mathbf{x} = \Sigma^{1/2}\mathbf{z} + \mu$$







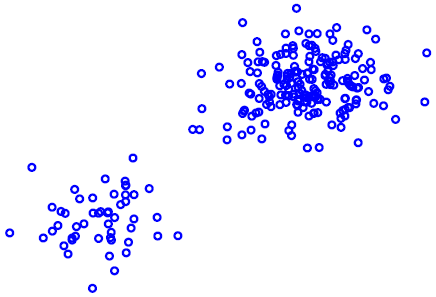
Multi-variate density estimation



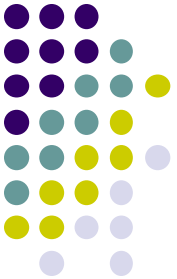
- A mixture of Gaussians model

$$p(\mathbf{x}|\theta) = \sum_{i=1}^{k} p_j \, p(\mathbf{x}|\mu_j, \Sigma_j)$$

where $\theta = \{p_1, \ldots, p_k, \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k\}$ contains all the parameters of the mixture model. $\{p_j\}$ are known as *mixing proportions or coefficients*.



Mixtures of Gaussians: Wishart distribution



- A mixture of Gaussian Model:

$$p(\mathbf{x}|\theta) = \sum_{i=1}^{k} p_j \, p(\mathbf{x}|\mu_j, \Sigma_j)$$

High dimensional parameters

$$\theta = \{p_1, \ldots, p_k, \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k\}$$

- Wishart prior

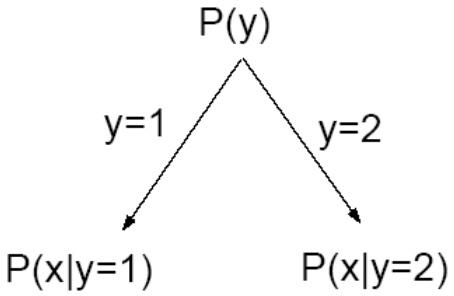
$$P(\Sigma|S, n') \propto \frac{1}{|\Sigma|^{n'/2}} \exp\left(-\frac{n'}{2}\text{Trace}(\Sigma^{-1} S)\right)$$

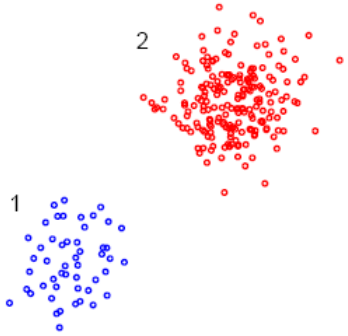
S = "prior" covariance matrix

n' = equivalent sample size

Mixture density

- Data generation process:



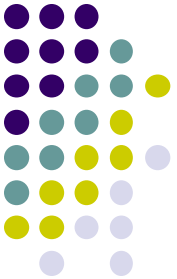


$$p(\mathbf{x}|\theta) = \sum_{j=1,2} P(y=j) \cdot p(\mathbf{x}|y=j) \quad \text{(generic mixture)}$$

$$= \sum_{j=1,2} p_j \cdot p(\mathbf{x}|\mu_j, \Sigma_j) \quad \text{(mixture of Gaussians)}$$

- Any data point $\mathbf{x}$ could have been generated in two ways

Mixture density



- If we are given just $\mathbf{x}$ we don't know which mixture component this example came from

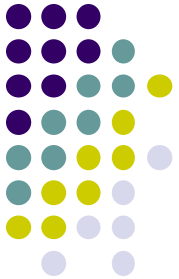
$$p(\mathbf{x}|\theta) = \sum_{j=1,2} p_j p(\mathbf{x}|\mu_j, \Sigma_j)$$

- We can evaluate the posterior probability that an observed $\mathbf{x}$ was generated from the first mixture component

$$\begin{aligned} P(y=1|\mathbf{x},\theta) &= \frac{P(y=1)\cdot p(\mathbf{x}|y=1)}{\sum_{j=1,2} P(y=j)\cdot p(\mathbf{x}|y=j)} \\ &= \frac{p_1\, p(\mathbf{x}|\mu_1,\Sigma_1)}{\sum_{j=1,2} p_j\, p(\mathbf{x}|\mu_j,\Sigma_j)} \end{aligned}$$

- This solves a *credit assignment* problem

Mixture density: posterior sampling



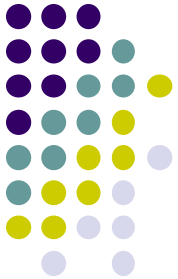
- Consider sampling $\mathbf{x}$ from the mixture density, then y from the posterior over the components given $\mathbf{x}$, and finally $\mathbf{x}'$ from the component density indicated by y:

$$\begin{aligned} \mathbf{x} &\sim p(\mathbf{x}|\theta) \\ y &\sim P(y|\mathbf{x},\theta) \\ \mathbf{x}' &\sim p(\mathbf{x}'|y,\theta) \end{aligned}$$

Is y a fair sample from the prior distribution $P(y)$?

Is $\mathbf{x}'$ a fair sample from the mixture density $p(\mathbf{x}'|\theta)$?

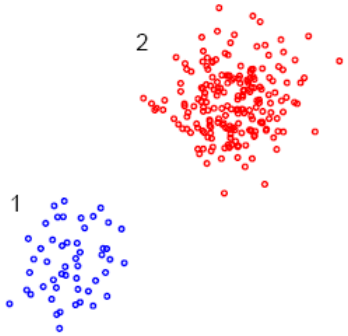
Mixture density estimation



- Suppose we want to estimate a two component mixture of Gaussians model.

$$p(\mathbf{x}|\theta) = p_1\, p(\mathbf{x}|\mu_1, \Sigma_1) + p_2\, p(\mathbf{x}|\mu_2, \Sigma_2)$$

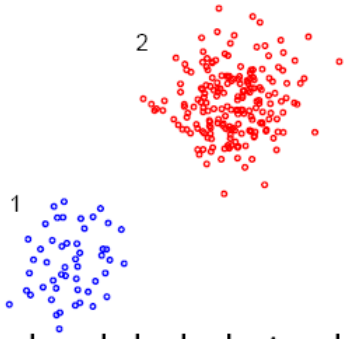
- If each example $\mathbf{x}_i$ in the training set were labeled $y_i = 1, 2$ according to which mixture component (1 or 2) had generated it, then the estimation would be easy.



- Labeled examples $\Rightarrow$ no credit assignment problem

Mixture density estimation

When examples are already assigned to mixture components (labeled), we can estimate each Gaussian independently



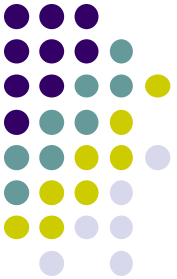
- If $\hat{n}_j$ is the number of examples labeled j, then for each $j = 1, 2$ we set

$$\hat{p}_j \leftarrow \frac{\hat{n}_j}{n}$$

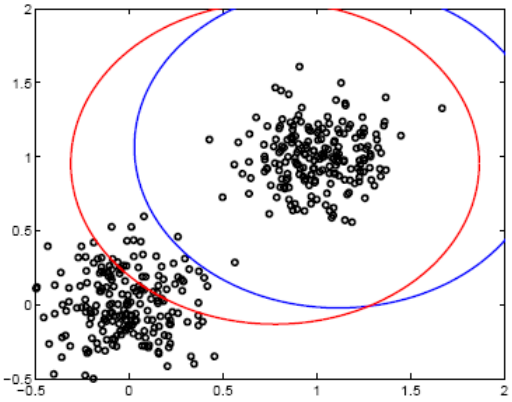
$$\hat{\mu}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i:y_i=j} \mathbf{x}_i$$

$$\hat{\Sigma}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i:y_i=j} (\mathbf{x}_i - \hat{\mu}_j)(\mathbf{x}_i - \hat{\mu}_j)^T$$

Mixture density estimation: credit assignment



- Of course we don't have such labels ... but we can guess what the labels might be based on our current mixture distribution
- We get soft labels or posterior probabilities of which Gaussian generated which example:



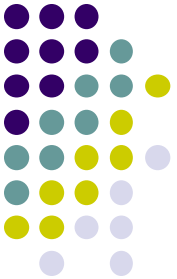
$$\hat{p}(j|i) \leftarrow P(y_i = j|\mathbf{x}_i, \theta)$$

where $\sum_{j=1,2} \hat{p}(j|i) = 1$ for all $i = 1, \ldots, n$.

- When the Gaussians are almost identical (as in the figure), $\hat{p}(1|i) \approx \hat{p}(2|i)$ for almost any available point $\mathbf{x}_i$.

  Even slight differences can help us determine how we should modify the Gaussians.

The Expectation-Maximization algorithm



E-step: softly assign examples to mixture components

$$\hat{p}(j|i) \leftarrow P(y_i = j|\mathbf{x}_i, \theta), \quad \text{for all } j = 1, 2 \text{ and } i = 1, \ldots, n$$

M-step: re-estimate the parameters (separately for the two Gaussians) based on the soft assignments.

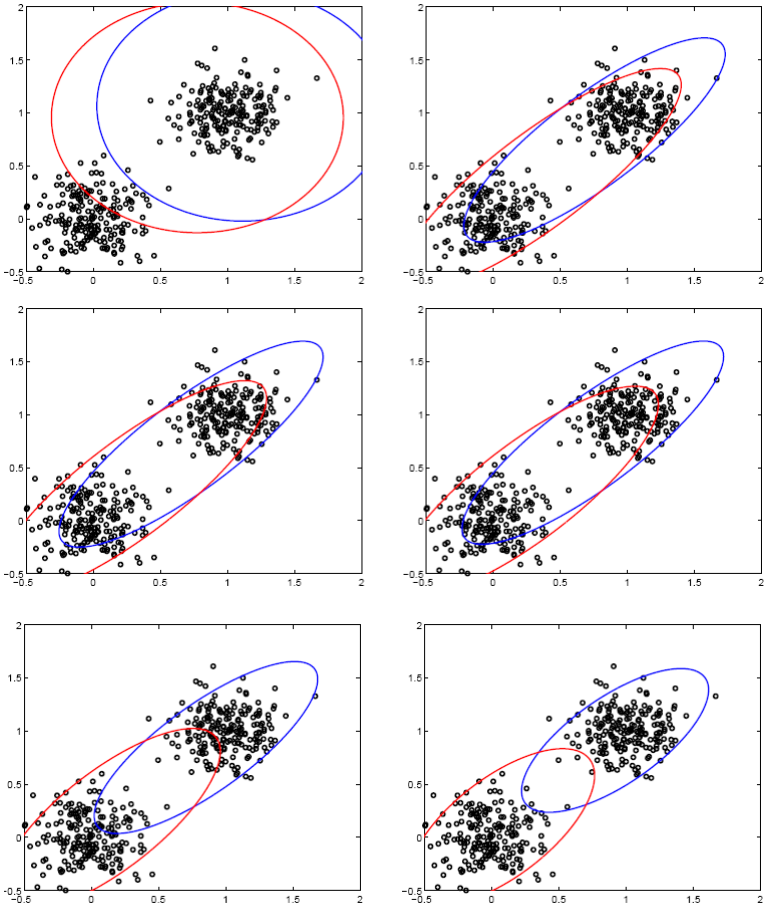
$$\hat{n}_j \leftarrow \sum_{i=1}^{n} \hat{p}(j|i) = \text{Soft } \# \text{ of examples labeled } j$$

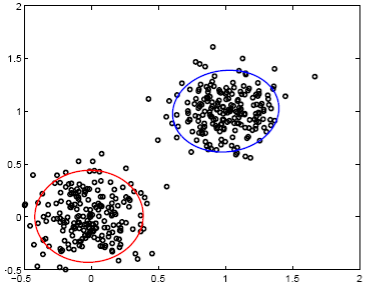
$$\hat{p}_j \leftarrow \frac{\hat{n}_j}{n}$$

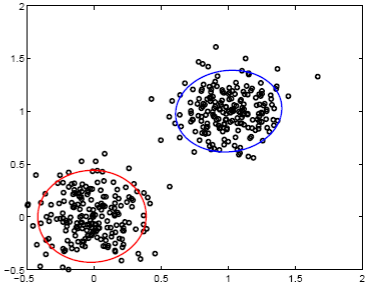
$$\hat{\mu}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i=1}^{n} \hat{p}(j|i)\, \mathbf{x}_i$$

$$\hat{\Sigma}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i=1}^{n} \hat{p}(j|i)\, (\mathbf{x}_i - \hat{\mu}_j)(\mathbf{x}_i - \hat{\mu}_j)^T$$

Mixture density estimation: example





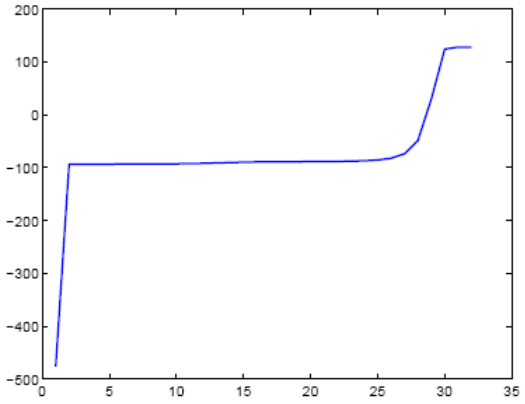


The EM-algorithm

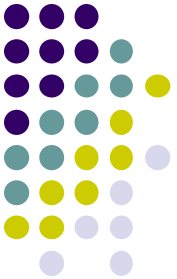
- Each iteration of the EM-algorithm *monotonically* increases the (log-)likelihood of the n training examples $\mathbf{x}_1, \ldots, \mathbf{x}_n$:

$$\log p(\text{ data } | \theta) = \sum_{i=1}^{n} \log \left(\overbrace{p_1\, p(\mathbf{x}_i | \mu_1, \Sigma_1) + p_2\, p(\mathbf{x}_i | \mu_2, \Sigma_2)}^{p(\mathbf{x}_i | \theta)} \right)$$

where $\theta = \{p_1, p_2, \mu_1, \mu_2, \Sigma_1, \Sigma_2\}$ contains all the parameters of the mixture model.



The EM algorithm



- The EM-algorithm finds a local maximum of $l(\theta; D)$

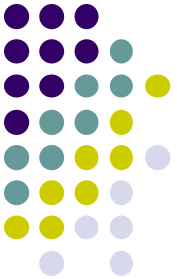
  E-step: evaluate the expected complete log-likelihood

$$
\begin{aligned}
J(\theta; \theta^{(t)}) &= \sum_{i=1}^{n} E_{j \sim P(j|\mathbf{x}_i, \theta^{(t)})} \log \left(p_j p(\mathbf{x}_i | \mu_j, \Sigma_j) \right) \\
&= \sum_{i=1}^{n} \sum_{j=1,2} P(j|\mathbf{x}_i, \theta^{(t)}) \log \left(p_j p(\mathbf{x}_i | \mu_j, \Sigma_j) \right)
\end{aligned}
$$

  M-step: find the new parameters by maximizing the expected complete log-likelihood

$$
\theta^{(t+1)} \leftarrow \arg\max_{\theta} J(\theta; \theta^{(t)})
$$

Regularized EM algorithm



- To maximize a penalized (regularized) log-likelihood

$$l'(\theta; D) = \sum_{i=1}^{n} \log p(\mathbf{x}_i|\theta) + \log p(\theta)$$

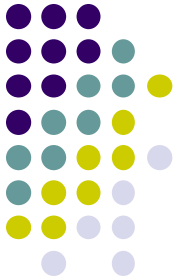
we only need to modify the M-step of the EM-algorithm.

Specifically, in the M-step, we find find θ that maximize a penalized expected complete log-likelihood:

$$\begin{aligned} J(\theta; \theta^{(t)}) &= \sum_{i=1}^{n} E_{j \sim P(j|\mathbf{x}_i, \theta^{(t)})} \log \left(p_j p(\mathbf{x}_i|\mu_j, \Sigma_j) \right) \\ &\quad + \log p(p_1, p_2) + \log p(\Sigma_1) + \log p(\Sigma_1) \end{aligned}$$

where, for example, $p(p_1, p_2)$ could be a *Dirichlet* and each $p(\Sigma_j)$ a *Wishart* prior.

Selecting the number of components

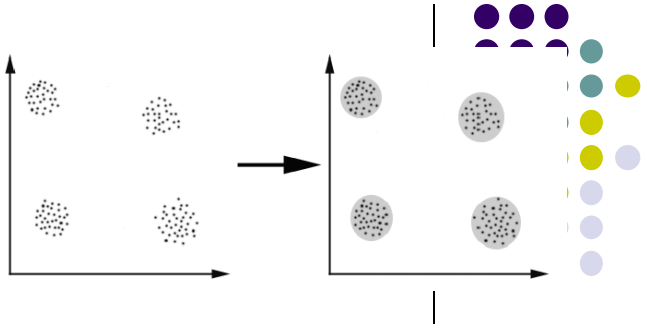


- As a simple strategy for selecting the appropriate number of mixture components, we can find k that minimize the following asymptotic approximation to the description length:

$$\text{DL} \approx -\log p(\text{data}|\hat{\theta}_k) + \frac{d_k}{2}\log(n)$$

where n is the number of training points, $\hat{\theta}_k$ is the maximum likelihood parameter estimate for the k-component mixture, and d_k is the (effective) number of parameters in the k-mixture.

K-means clustering



Given data $<x_1 \ldots x_n>$, and K, assign each x_i to one of K clusters, $C_1 \ldots C_K$, minimizing $J = \sum_{j=1}^{K} \sum_{x_i \in C_j} ||x_i - \mu_j||^2$

Where μ_j is mean over all points in cluster C_j

K-Means Algorithm:

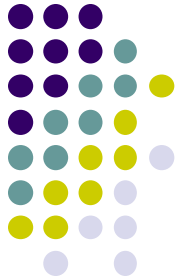
Initialize $\mu_1 \cdots \mu_K$ randomly

Repeat until convergence:

1. Assign each point x_i to the cluster with the closest mean μ_j
2. Calculate the new mean for each cluster

$$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

K-Means vs. Mixture of Gaussians

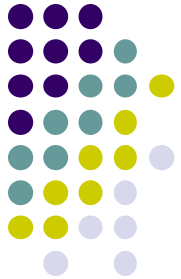


- Both are iterative algorithms to assign points to clusters

- Objective function
  - K Means: minimize $J = \sum_{j=1}^{K} \sum_{x_i \in C_j} ||x_i - \mu_j||^2$
  - MoG: maximize likelihood $P(X|\theta)$

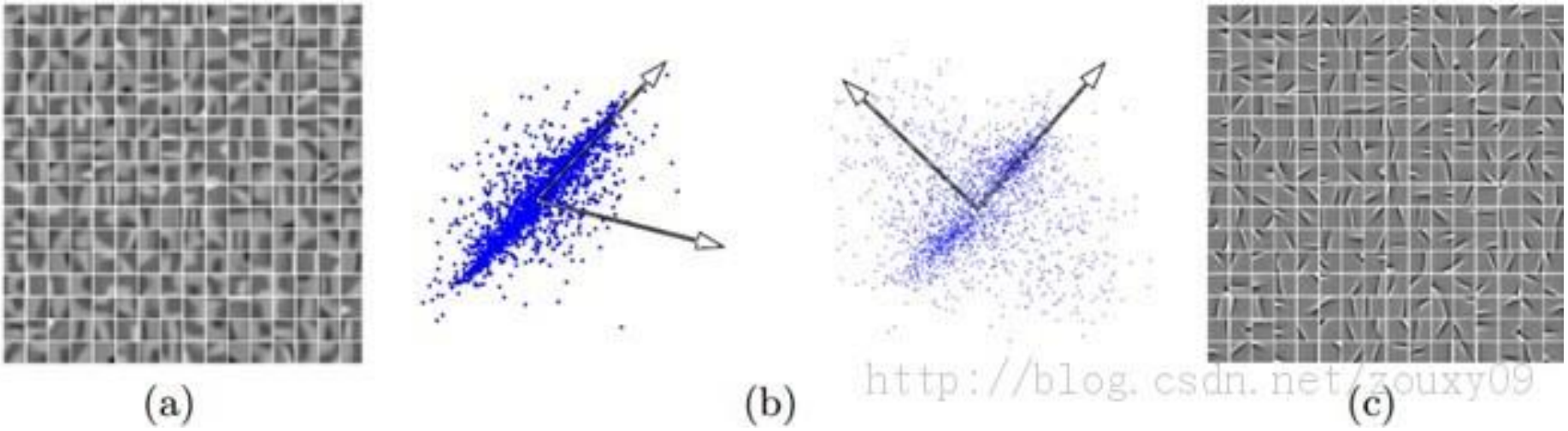
- MoG the more general formulation
  - Equivalent to K Means when $\Sigma_k = \sigma I$, and $\sigma \rightarrow 0$

Disadvantage of K-means and MOG



- The result is sensitive to the initial data
- How to determine the number of clusters

K-means and whitening



Learning Feature Representations with K-means, Adam Coates and Andrew Y. Ng. In Neural Networks: Tricks of the Trade, Reloaded, Springer LNCS, 2012

K-means and whitening

1. Normalize inputs:

$$x^{(i)} := \frac{x^{(i)} - \text{mean}(x^{(i)})}{\sqrt{\text{var}(x^{(i)}) + \epsilon_{\text{norm}}}}, \forall i$$

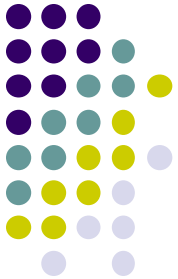
2. Whiten inputs:

$$[V, D] := \text{eig}(\text{cov}(x)); \ // \text{ So } VDV^\top = \text{cov}(x)$$
$$x^{(i)} := V(D + \epsilon_{\text{zca}} I)^{-1/2} V^\top x^{(i)}, \forall i$$

3. Loop until convergence (typically 10 iterations is enough):

$$s_j^{(i)} := \begin{cases} \mathcal{D}^{(j)\top} x^{(i)} & \text{if } j == \arg\max_l |\mathcal{D}^{(l)\top} x^{(i)}| \\ 0 & \text{otherwise.} \end{cases} \quad \forall j, i$$
$$\mathcal{D} := XS^\top + \mathcal{D}$$
$$\mathcal{D}^{(j)} := \mathcal{D}^{(j)} / ||D^{(j)}||_2 \forall j$$

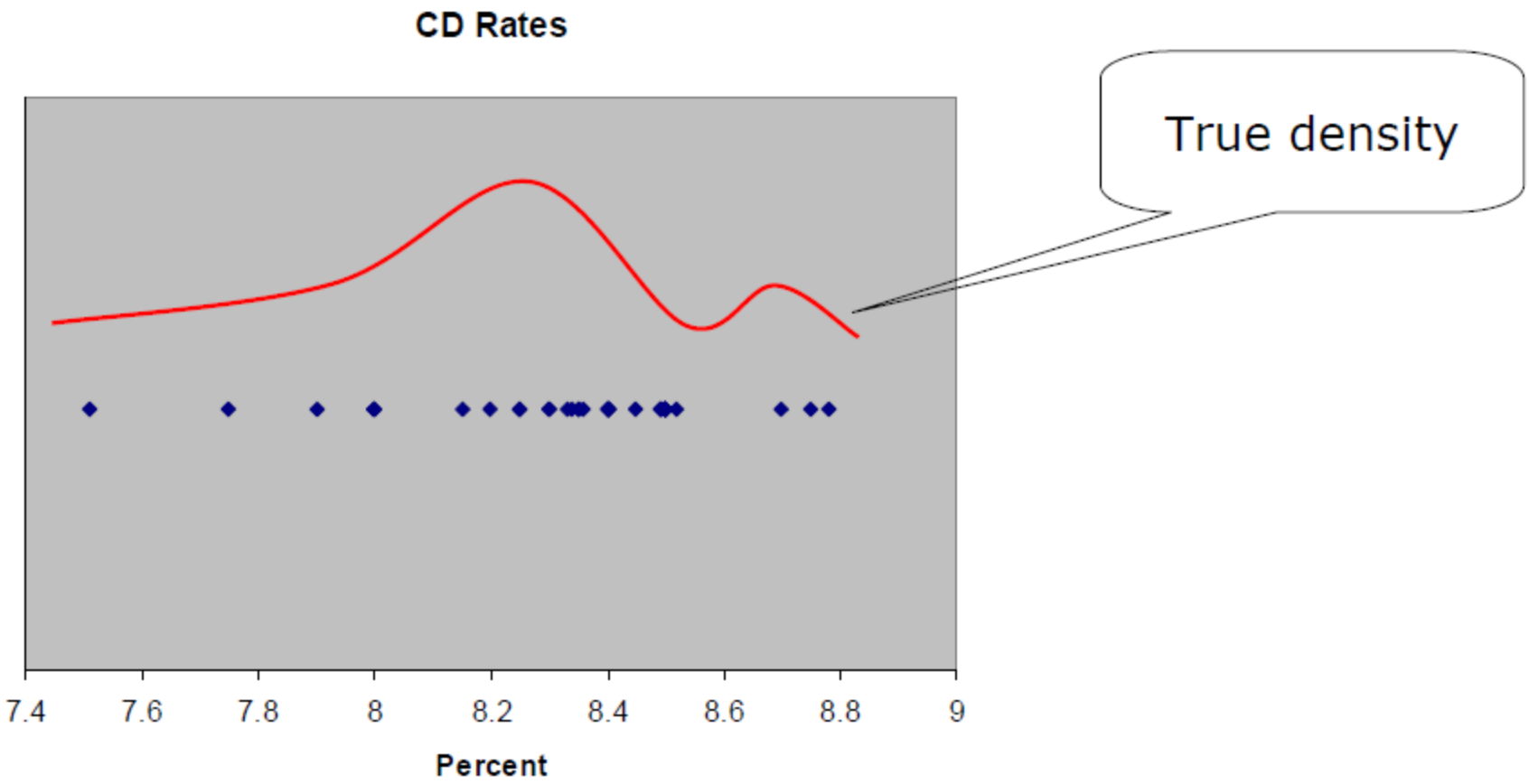
Mean shift



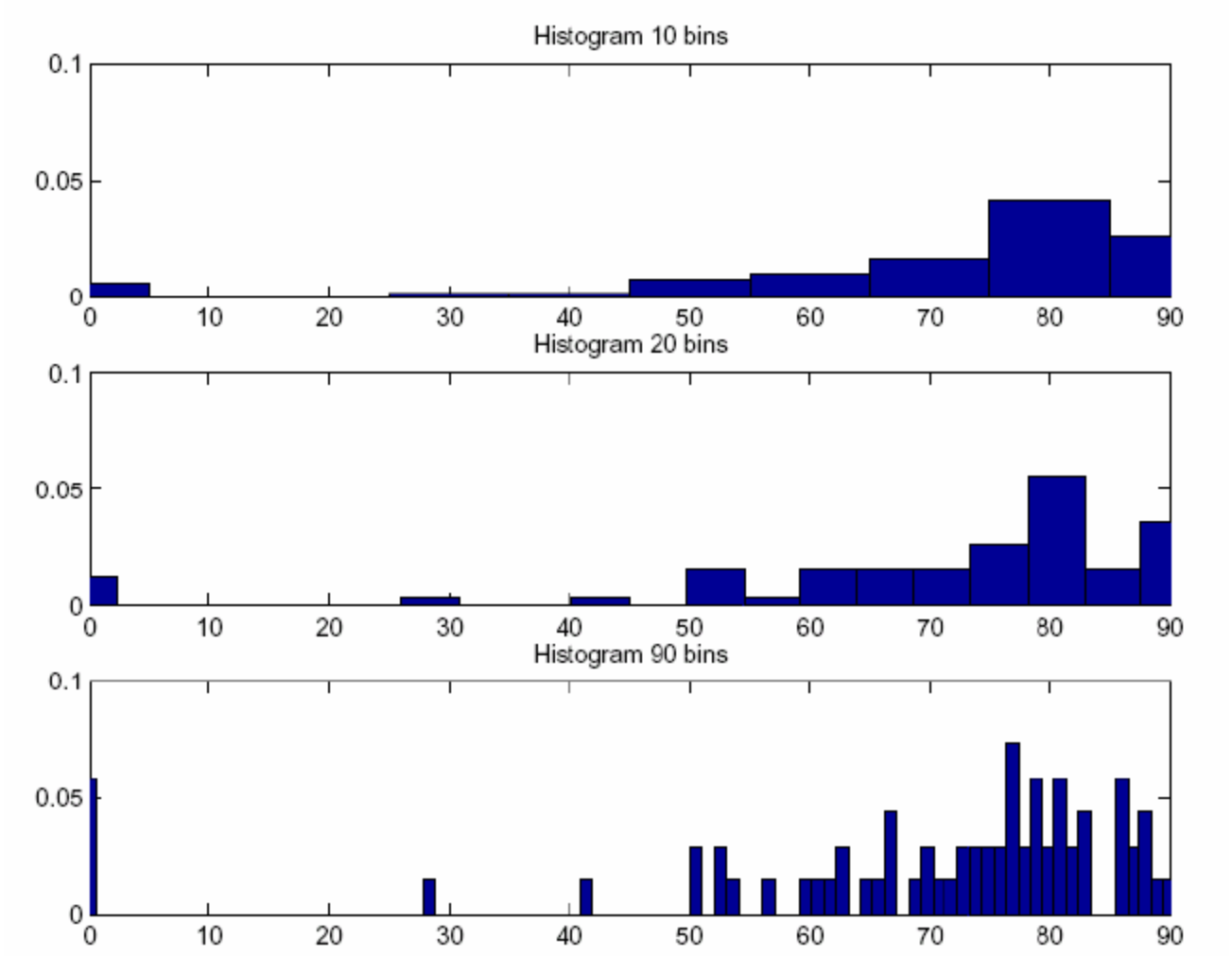
- First proposed by Fukunaga in 1970's
- Wildly used since 1998
  - In computer vision
  - And other areas

- The following several slides is mainly from:
  - http://www.cs.cornell.edu/courses/cs664/2005fa/Lectures/lecture3.pdf

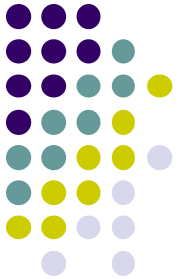
Density estimation



Histogram representation

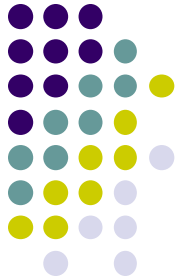


Histogram-based estimates



- You can use a variety of fitting techniques to produce a curve from a histogram
  - Lines, polynomials, splines, etc.
  - Also called regression/function approximation
  - Normalize to make this a density
- If you know quite a bit about the underlying density you can compute a good bin size
  - But that's rarely realistic in vision
  - And defeats the whole purpose of the non-parametric approach

Nearest-neighbor estimate

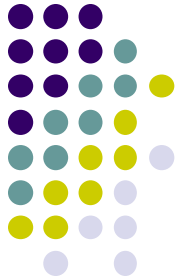


- To estimate the density, count the number of nearby data points
  - Like histogramming with sliding bins
  - Avoid bin-placement artifacts

$$\hat{p}(x) = \frac{\#\{x_i \mid \|x_i - x\| \leq \varepsilon\}}{N}$$

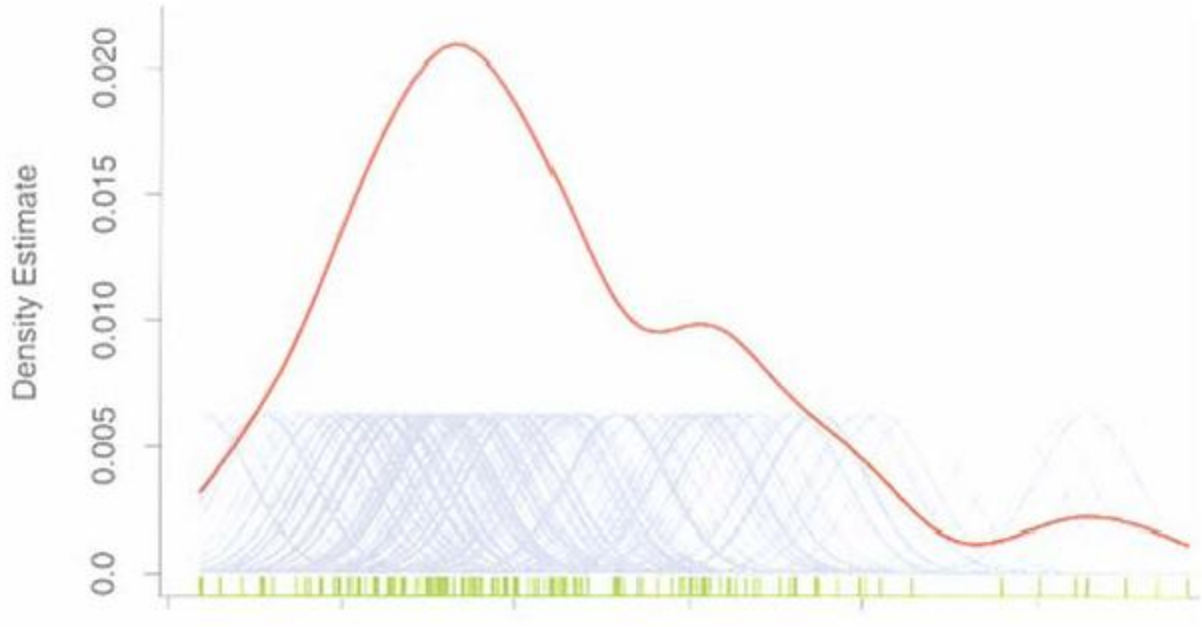
  - We can fix ε and compute this quantity, or we can fix the quantity and compute ε

Parzen estimation



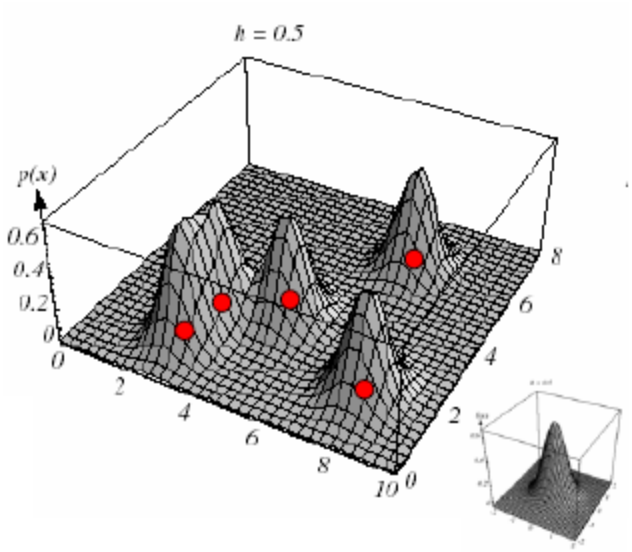
- Each observed data increases our estimate of the probability nearby
  - Simplest case: raise the probability uniformly within a fixed radius
    - Place a fixed-height “box” at each data point, add them up to get the density estimate
  - This is nearest neighbor with fixed ε
- More generally, you can use some slowly decreasing function (such as a Gaussian)
  - Called Kernel function

Parzen example



from Hastie *et al.*

Importance of scale



Mean shift algorithm

- Non-parametric method to compute the nearest mode of a distribution
  - Density increases as we get near “center”

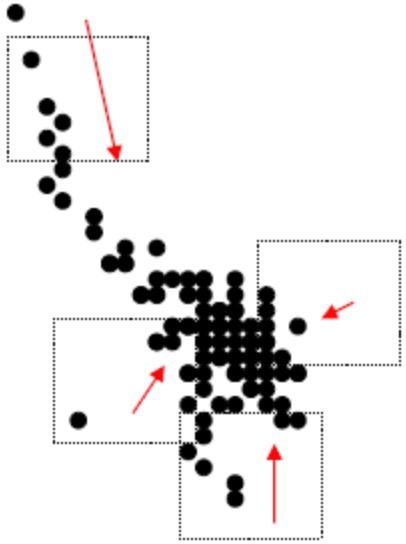
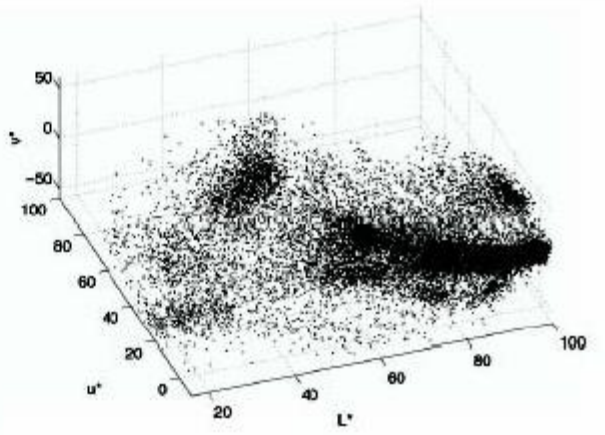
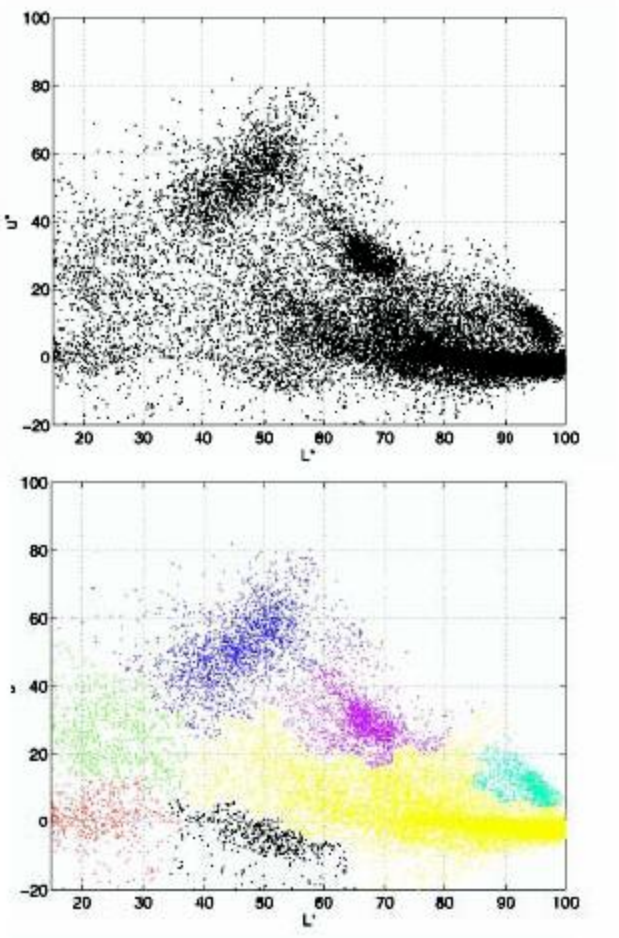
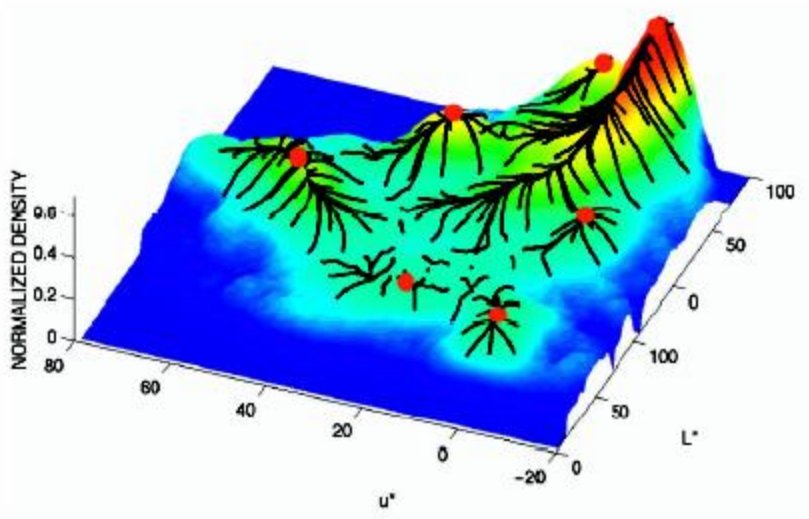


Image and histogram

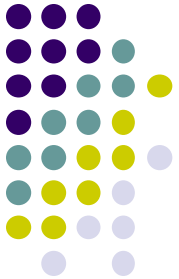


Local modes





Kernel Density Estimation



- Multivariate kernel density estimation

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} \frac{1}{h} K\left(\frac{x - x_i}{h}\right)$$

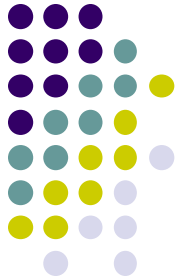
- Kernels
  - Gaussian

$$K_N = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$$

  - Epanechnikov

$$K_E = \begin{cases} 1/2c_d^{-1}(d+2)(1-\|\mathbf{x}\|^2) & \text{if } \|\mathbf{x}\| < 1 \\ 0 & \text{otherwise} \end{cases}$$

Finding Mean-Shift Vector



- Gradient computation
  - For symmetric kernel

$$\hat{\nabla} f(\mathbf{x}) = \frac{2}{nh^{d+2}} \sum_{i=1}^{n} K_N\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)\left[\frac{\sum_{i=1}^{n} \mathbf{x}_i K_N\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)}{\sum_{i=1}^{n} K_N\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)} - \mathbf{x}\right]$$

- Always converges to the local maximum!

The mean shift procedure

- Give a point x
  1. Compute the mean shift vector

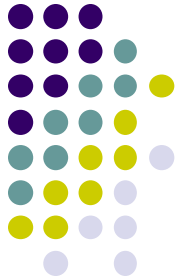
$$\hat{\nabla} f(\mathbf{x}) = \frac{2}{nh^{d+2}} \sum_{i=1}^{n} K_N\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i K_N\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)}{\sum_{i=1}^{n} K_N\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)} - \mathbf{x} \right]$$

  2. Translate density estimation window:

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \hat{\nabla} f(\mathbf{x}^{(t)})$$

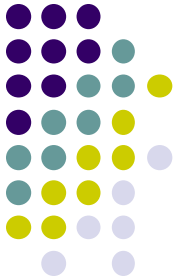
  3. Iterate steps 1. and 2. until convergence
     i.e., $\hat{\nabla} f(\mathbf{x}) \rightarrow 0$

Applications



- Pattern recognition
  - Clustering
- Image processing
  - Filtering
  - Segmentation
- Density estimation
  - Density approximation
  - Particle filter
- Mid-level application
  - Tracking
  - Background subtraction

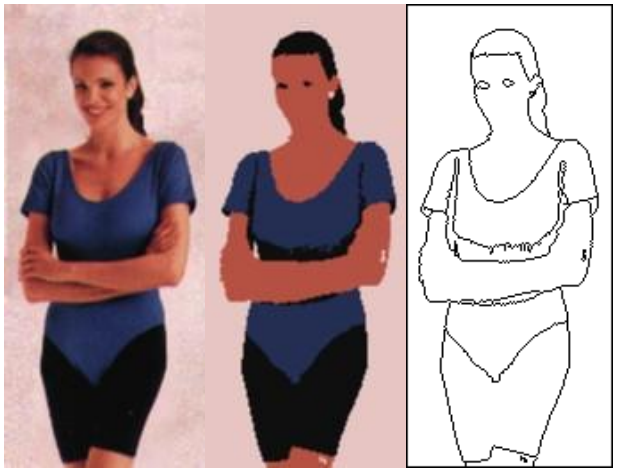
Summary



- The distance computing plays an important role in data analysis to find out
  - the suitable similarity measurement
  - the intrinsic structure of data

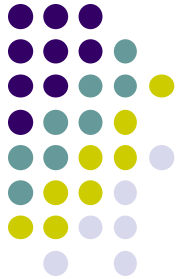
- Further reading on metric learning
- In the next lesson, we will explore more complex data with structure

Image segmentation based on mean shift



Mean-shift segmentation

Clustering



- Hierarchical clustering
  - bottom-up
- Flat clustering
  - Mixture of Gaussians
  - K-means
- Spectral based clustering

An example: ISO/BLE-charts

- ISO-Charts:
  - ISOMAP + Spectral Clustering + Stretch Minimization
- BLE-Charts:
  - Statistical Embedding + Spectral Clustering + Stretch Minimization

The End

新浪微博：@浙大张宏鑫

微信公众号：