CPSC 540 Tutorial

Reza Babanezhad

rezababa@cs.ubc.ca

Outline

- EM
- Robust PCA algorithm
- Fun with ML
- Question for you!

EM Algorithm

- X is observed
- Y is unobserved
- ϴ is parameter whose estimation is easier with considering Y!

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta})\, d\mathbf{y}$$

- E-Step:

$$\begin{aligned} Q(\boldsymbol{\theta}|\widehat{\boldsymbol{\theta}}^{(t)}) &\equiv E[\log p(\mathbf{y}, \boldsymbol{\theta}|\mathbf{x})|\mathbf{x}, \widehat{\boldsymbol{\theta}}^{(t)}] \\ &\propto \log p(\boldsymbol{\theta}) + E[\log p(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta})|\mathbf{x}, \widehat{\boldsymbol{\theta}}^{(t)}] \\ &= \log p(\boldsymbol{\theta}) + \int p(\mathbf{y}|\mathbf{x}, \widehat{\boldsymbol{\theta}}^{(t)}) \log p(\mathbf{y}, \mathbf{x}|\boldsymbol{\theta})\, d\mathbf{y} \end{aligned}$$

here we are doing MAP estimation

- M-Step:

$$\widehat{\boldsymbol{\theta}}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\widehat{\boldsymbol{\theta}}^{(t)})$$

EM for GMM

- Multivariate Normal Distribution

$$p(x_i|\mu, \Sigma_k) = (2\pi)^{-\frac{d}{2}}|\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu)\right)$$

- Mixture distribution

$$p(x_i) = \sum_{k=1}^{K} p(z_i = k|\pi)p(x_i|\mu_k, \Sigma_k),$$

- Probability of each cluster:

$$p(z_i = k) = \pi_k$$

GMM

- E-Step:
  - Log likelihood:

$$\mathcal{L}(X, Z) = \log \prod_{k=1}^{K} \prod_{i=1}^{N} \Big(P(z_i = k \mid \pi) \, P(x_i \mid z_i = k, \mu_k, \Sigma_k) \Big)^{\mathbb{I}(z_i = k \mid \theta^t, X)}$$

$$= \sum_{k=1}^{K} \sum_{i=1}^{N} \mathbb{I}[z_i = k] \Big[\log \pi_k + \log P(x_i \mid z_i = k, \mu_k, \Sigma_k) \Big]$$

If we assume we are at step t+1, & we know our parameters values at step t:

GMM

$$E_{Z|\mu^t,\pi^t,\Sigma^t,x}[\mathcal{L}(X,Z)] = \sum_{k=1}^{K}\sum_{i=1}^{N} E_{Z|\mu^t,\Sigma^t,\pi^t,x}[I[z_i=k]][\log\pi^k + \log P(x|\mu_k,\Sigma_k)]$$

$$E_{Z|\theta^t,x}[I[z_i=k]] = P(z_i=k|\theta^t,x) = \tau_k^i$$

Using Bayes rule

$$\tau_k^i = \frac{P(x_i|z_i=k,\mu_k^t,\Sigma_k^t)P(z_i=k|\pi^t)}{\sum_{k'=1}^{K} P(x_i|z_i=k',\mu_{k'}^t,\Sigma_{k'}^t)P(z_i=k'|\pi^t)} = \frac{\mathcal{N}(\mu_k^t,\Sigma_k^t)\pi_k^t}{\sum_{k'=1}^{K}\mathcal{N}(\mu_{k'}^t,\Sigma_{k'}^t)\pi_{k'}^t}$$

$$Q_{\theta|\theta^t} = \sum_{k=1}^{K}\sum_{i=1}^{N} \tau_k^i \left[\log \pi_k + \log \mathcal{N}(\mu_k, \Sigma_k)\right]$$

M-step: We have an extra condition that $\sum_{k=1}^{K} \pi_k = 1$

we have to consider this when optimizing w.r.t. π

$$\log \mathcal{N}(\mu_k, \Sigma_k) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\log|\Sigma| - \frac{1}{2}\operatorname{tr}(S_k \Sigma_k^{-1})$$

$$S_k = (X-\mu_k)^T(X-\mu_k) = \sum_{i}^{N}(x_i-\mu_k)^2$$

$$\frac{\partial Q^{t+1}}{\partial \mu_k} = \frac{\partial \left(\sum_{i=1}^{N} \tau_k^i (x_i - \mu_k)^T \Sigma^{-1} (x_i - \mu_k) \right)}{\partial \mu_k} = 0$$

$$\Rightarrow \frac{\partial Q^T}{\partial \mu_k} = 2 \sum_{i=1}^{N} \tau_i^k (x_i - \mu_k) \Sigma_k^{-1} = 0$$

$$\Rightarrow \sum_{i=1}^{N} \tau_i^k x_i - \sum_{i=1}^{N} \tau_i^k \mu_k = 0 \Rightarrow \mu_k^{t+1} = \frac{\sum_{i=1}^{N} \tau_i^k x_i}{\sum_{i=1}^{N} \tau_i^k}$$

$$\frac{\partial Q}{\partial \Sigma_k^{-1}} = \frac{\partial \sum_{i=1}^{N} \tau_i^k \left(\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}tr\left(S_i^k \Sigma_k^{-1}\right)\right)}{\partial \Sigma_k^{-1}} = 0$$

$$S_i^k = (x_i - \mu_k^{t+1})^T (x_i - \mu_k^{t+1})$$

$$\Rightarrow \frac{\partial Q}{\partial \Sigma^{-1}} = \sum_{i=1}^{N} \frac{\tau_i^k}{2}\left(+\Sigma - S_i^k\right) = 0$$

$$\Rightarrow \sum \tau_i^k \Sigma_k = \sum \tau_i^k S_i^k \Rightarrow \Sigma_k^{t+1} = \frac{\sum_i^N \tau_i^k S_i^k}{\sum_i^N \tau_i^k}$$

$$\frac{\partial Q}{\partial \pi_k} = \frac{\partial \sum_{i=1}^{N} \tau_k^i (\log \pi_k)}{\partial \pi_k} = 0 \quad s.t. \sum_{k'=1}^{K} \pi_{k'} = 1$$

$$\Rightarrow \frac{\partial Q}{\partial \pi_k} = \frac{\partial \left[\sum_{i=1}^{N} \tau_k^i (\log \pi_k) + \lambda \left(1 - \sum_{k'=1}^{K} \pi_{k'}\right) \right]}{\partial \pi_k} = 0$$

$$\Rightarrow \frac{\partial Q}{\partial \pi_k} = \sum_{i=1}^{N} \frac{\tau_k^i}{\pi_k} - \lambda = 0 \Rightarrow \pi_k = \frac{\sum_{i=1}^{N} \tau_k^i}{\lambda}$$

$$\sum_{k=1}^{K} \pi_k = 1 \rightarrow \sum_{k=1}^{K} \frac{\sum_{i=1}^{N} \tau_k^i}{\lambda} = 1 \Rightarrow \lambda = \sum_{k=1}^{K} \sum_{i=1}^{N} \tau_k^i$$

EM for Semi-supervised learning

- Binary Naïve Bayes Classifier

Let (x_i, y_i) be sample data, $i \in \{1 \dots N\}$, $y_i \in \{1, \dots C\}$

$x^i \in R^d$, $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$, $x_{ij} \in \{0, 1\}$ $j \in [1, d]$

Parameters (π, θ), $P(y = c | \pi) = \pi_c$, $P(x_{ij} = 1 | y_i = c, \theta) = \theta_{jc}$

$$P(x, y | \theta, \pi) = P(y_i | \pi) \prod_{j=1}^{d} P(x_{ij} | \theta_j)$$

likelihood: $$P(X, y | \theta, \pi) = \prod_{i=1}^{N} \prod_{c=1}^{C} \pi_c^{I(y_i = c)} \prod_{j=1}^{d} \theta_{jc}^{x_j^i I(y_i = c)} (1 - \theta_{jc})^{1 - x_j^i I(y_i = c)}$$

$$\theta_{jc} = \frac{N_{jc}}{N_c}, \quad \theta_c = \frac{N_c}{N}, \quad N_c = \sum I(y_i = c), \quad N_{jc} = \sum I[y_i = c, x_{ij} = 1]$$

EM for semi-supervised learning

- Now assume we have small labeled data set {X_L, y_L} and a large unlabelled data set {X_U}.
- Assume all variables are binary and we want to use Naïve bayes for classification
- Let N be the size of labelled set and M of unlabelled
- Parameter set: $\theta = \{\theta_1, \theta_{01}, \theta_{11}, \theta_{02}, \theta_{12}, \ldots, \theta_{0d}, \theta_{1d}\}$
- We want to drive EM algorithms step which treat y_U as hidden variables.

$$P(y_L, X_L, y_U, X_U) = \prod_{i=1}^{N} P[x_i, y_i | \theta] \prod_{m=1}^{M} P(y_m, x_m | \theta]$$

We don't know y_u. So treat it as hidden variables and marginalize over all y_u.

$$P(y_L, X_L, X_u) = \sum_{y_1 \in \{0,T\}} \sum_{y_2 \in \{0,T\}} \cdots \sum_{y_M \in \{0,1\}} \prod_{i=1}^{N} P(y_i, x_i | \theta) \prod_{m=1}^{M} P(y_m, x_m | \theta)$$

E-step:

$$\mathcal{L} = P(y_L, x_L, X_u \mid \theta) = \prod_{i=1}^{N} P(y_i, x_i \mid \theta) \prod_{m=1}^{M} \left[\sum_{y_c \in \{0,1\}} P(y_m, x_m \mid \theta) \right]$$

$$\log \mathcal{L} = \sum_{i=1}^{N} \log P(y_i, x_i \mid \theta) + \sum_{m=1}^{M} \log \sum_{y_m \in \{0,1\}} P(y_m, x_m \mid \theta)$$

$$\geq \sum_{i=1}^{N} \log P(y_i, x_i \mid \theta) + \sum_{m=1}^{M} \sum_{y_m \in \{0,1\}} P(y_m \mid x_m, \theta^t) \log P(y_m, x_m \mid \theta)$$

$$= Q(\theta \mid \theta^t)$$

at step t!

$$\gamma_{m0}^{t} = P(y_m = 0 \mid x_m, \theta^t) = \frac{P(x_m \mid y_m = 0, \theta^t)\, P(y_m = 0 \mid \theta^t)}{\sum_{y_m \in \{0,1\}} P(x_m \mid y_m, \theta^t)\, P(y_m \mid \theta^t)}$$

We can define r_{m1}^{t} similarly;

$$P(y_i \mid x_i, \theta^t) \propto P(y_i, x_i \mid \theta^t) = P(y_i \mid \theta^t) \prod_{j=1}^{d} P(x_{ij} \mid y_i, \theta^T)$$

$$= (\theta_1^t)^{y_i} (1 - \theta_1^t)^{1 - y_i} \prod_{j=1}^{d} (\theta_{y_i j}^t)^{x_i} (1 - \theta_{y_i j})^{1 - x_i}$$

M- STEP:

For a single point we have:

$$\log P(y_i, x_i \mid \theta) = \log P(y_i \mid \theta_1) + \sum_{i=1}^{d} \log\left(P\left(x_{ij} \mid y_i, \theta_{y_i, j}\right)\right)$$

$$\frac{\partial \log P(y_i, x_i \mid \theta)}{\partial \theta_1} = \frac{\partial \log P(y_i \mid \theta_1)}{\partial \theta_1} + \frac{\partial c}{\partial \theta_1} \rightarrow 0$$

$$= \frac{\partial \log \theta_1^{y_i} (1-\theta_1)^{1-y_i}}{\partial \theta_1} = \frac{\partial \left[y_i \log \theta_1 + (1-y_i) \log (1-\theta_1) \right]}{\partial \theta_1}$$

$$= \frac{y_i}{\theta_1} - \frac{1-y_i}{1-\theta_1}$$

Now we take derivite of $Q(\theta, \theta^t)$ w.r.t. θ_1

$$\frac{\partial Q}{\partial \theta_1} = \sum_{i=1}^{N} \frac{\partial \log P(y_i, x_i | \theta)}{\partial \theta_1} + \sum_{m=1}^{M} \sum_{y_m \in \{0,1\}} r_{m y_m}^t \frac{\partial \log P(y_m, x_m | \theta)}{\partial \theta_1}$$

$$= \sum_{i=1}^{N} \frac{\partial \log P(y_i, x_i | \theta)}{\partial \theta_1} + \sum_{m=1}^{M} r_{m0}^t \frac{\partial \log P(y_m = 0, x_m | \theta)}{\partial \theta_1}$$

$$+ \sum_{m=1}^{M} r_{m1}^t \frac{\partial \log P(y_m = 1, x_m | \theta)}{\partial \theta_1}$$

$$\frac{\partial Q}{\partial \theta_1} = \sum_{i=1}^{N}\left[\frac{y_i}{\theta_1} - \frac{1-y_i}{1-\theta_1}\right] + \sum_{m=1}^{M} r_{m0}^{t}\left[\frac{0}{\theta_1} - \frac{1-0}{1-\theta_1}\right] + \sum_{m=1}^{T} r_{m1}^{t}\left[\frac{1}{\theta_1} - \frac{1-1}{1-\theta_1}\right]$$

$$= \sum_{i=1}^{N}\left[\frac{y_i}{\theta_1} - \frac{1-y_i}{1-\theta_1}\right] - \sum_{m=1}^{M}\frac{r_{m0}^{t}}{1-\theta_1} + \sum_{m=1}^{M}\frac{r_{m1}^{t}}{\theta_1}$$

$$= \frac{\sum_{i=1}^{N} y_i + \sum_{m=1}^{M} r_{m1}^{t}}{\theta_1} - \frac{\sum_{i=1}^{N}(1-y_i) + \sum_{m=1}^{M} r_{m0}^{t}}{1-\theta_1}$$

$$= \frac{N_1 + R_1^{t}}{\theta_1} - \frac{N_0 + R_0^{t}}{1-\theta_1}, \quad N_j = \sum_{i=1}^{N} I[y_i == j], \quad R_j^{t} = \sum_{m=1}^{M} r_{mj}^{t}$$

$$j \in \{0,1\}$$

Setting $\frac{\partial Q}{\partial \theta_1} = 0$ we get

$$\frac{\theta_1}{1-\theta_1} = \frac{N_1 + R_1^t}{N_0 + R_0^t} \Rightarrow \theta_1 = \frac{N_1 + R_1^t}{N_0 + N_1 + R_1^t + R_0^t} = \frac{N_1 + R_1}{N + M}$$

$$\frac{\partial \log P(x_i, y_i \mid \theta)}{\partial \theta_{1j}} = \frac{\partial \log P(y_i \mid \theta)}{\partial \theta_{1j}} + \frac{\sum_{j=1}^{d} \partial \log P(x_{ij} \mid y_i, \theta)}{\partial \theta_{1j}}$$

$$= \frac{\partial \log P(x_{ij} \mid y_i, \theta)}{\partial \theta_{1j}} = \frac{\partial \log \theta_{1j}^{x_i \mathbb{I}[y_i=1]} (1-\theta_{1j})^{1-x_i \mathbb{I}[y_i=1]}}{\partial \theta_{ij}}$$

$$= \frac{\partial}{\partial \theta_{1j}} \left\{ x_i y_i \log \theta_{1j} + (1 - x_i y_i) \log (1 - \theta_{1j}) \right.$$

$$\frac{\partial \log P(x_i, y_i \mid \theta)}{\partial \theta_{1j}} = \frac{x_i y_i}{\theta_{1j}} - \frac{1 - x_i y_i}{1 - \theta_{1j}}$$

Similarly we can do for θ_{0j} :

$$\frac{\partial \log P(x_i, y_i \mid \theta)}{\partial \theta_{0j}} = \frac{x_i (1 - y_i)}{\theta_{0j}} - \frac{1 - x_i (1 - y_i)}{1 - \theta_{0j}}$$

Similar to θ_1 if we compute $\frac{\partial Q}{\partial \theta_{1j}}$ and $\frac{\partial Q}{\partial \theta_{0j}}$ and set it to zero we get:

$$\theta_{1j} = \frac{N_{11}^{j} + R_{11}^{j}}{N_1 + R_1}, \quad \theta_{0j} = \frac{N_{01}^{j} + R_{01}^{j}}{N_0 + R_0}$$

$$N_{11} = \sum_{i=1}^{N} I(y_i = 1)\, I(x_{ij} = 1), \quad N_{01} = \sum_{i=1}^{N} I(y_i = 0)\, I(x_{ij} = 1)$$

$$R_{11} = \sum_{i=1}^{M} r_{i1}\, I(x_{ij} = 1) \qquad R_{01} = \sum_{i=1}^{M} r_{i0}\, I(x_{ij} = 1)$$

Robust PCA

- Suppose we are given a large data matrix M and we may know it can decompose it to a low rank matrix L and a sparse matrix S:

$$M = L + S \qquad M \in \mathbb{R}^{m \times n}$$

- So the question is how can we compute L and S in a tractable manner?

Some Application

- Video Surveillance: Given a sequence of surveillance video frames, we often need to identify activities that stand out from the background. If we stack the video frames as columns of a matrix M, then the low-rank component L naturally corresponds to the stationary background and the sparse component S captures the moving objects in the foreground. However, each image frame has thousands or tens of thousands of pixels, and each video fragment contains hundreds or thousands of frames.

- Face Recognition: Images of a human’s face can be well-approximated by a low-dimensional subspace. Being able to correctly retrieve this subspace is crucial in many applications such as face recognition and alignment. However, realistic face images often suffer from self-shadowing, specularities, or saturations in brightness, which make this a difficult task and subsequently compromise the recognition performance

More application

- Latent Semantic Indexing. Web search engines often need to analyze and index the content of an enormous corpus of documents. A popular scheme is the Latent Semantic Indexing (LSI). The basic idea is to gather a document-versus-term matrix M whose entries typically encode the relevance of a term (or a word) to a document such as the frequency it appears in the document (e.g. the TF/IDF). PCA (or SVD) has traditionally been used to decompose the matrix as a low-rank part plus a residual, which is not necessarily sparse (as we would like). If we were able to decompose M as a sum of a low-rank component L and a sparse component S, then L could capture common words used in all the documents while S captures the few key words that best distinguish each document from others.

Problem formulation as optimization

- Traditional PCA

$$\min \|M - L\|_F \quad \text{s.t.} \quad \operatorname{rank}(L) \leq k$$

- Robust PCA

$$\min \|L\|_* + \lambda \|S\|_1 \quad \text{s.t.} \quad M = L + S$$

$$\|L\|_* := \sum_i \sigma_i(L), \quad \sigma_i \geq 0$$

$\sigma_i(L)$ → singular value of L

Singular Value Decomposition

let $M \in \mathbb{R}^{m \times n}$ & $n \leq m$

We can decompose M in a way s.t.

$$M = U \Sigma V^T \quad \& \quad V^T V = I, \quad U^T U = I$$

unitary matrix

$U \in \mathbb{R}^{m \times r}$ $\Sigma \in \mathbb{R}^{r \times r}$, $V \in \mathbb{R}^{n \times r}$

Σ is a diagonal matrix of $\sigma_1 \geq \sigma_2 \geq \sigma_3 \ldots \geq 0$

Robust PCA optimization

- The objective and constraint are convex ☺
- Multiple ways to formulate it
  - Two possible one:

1- $\min \|L\|_* + \lambda\|S\|_1 + \langle Y, M-S-L\rangle + \frac{1}{2\mu}\|M-L-F\|_F^2$

2- or simpler one:

$$\min \|L\|_* + \lambda\|S\|_1 + \frac{1}{2\mu}\|M-L-F\|_F^2$$

$\hookrightarrow$ Assignment is about this one!

Solving the optimization problem

- Finding the lambda:

In theory a good $\lambda = \frac{1}{\sqrt{0.1m}}$

- Block coordinate descent:

If we have $\min_{w,v} g(w) + f(v) + h(w,v)$

we can optimize w.r.t. each w & v separately and use a new value of one to update the other one!

Solving the optimization problem

$$\min_{W,V} f(V) + g(W) + h(V, W)$$

$$W^{t+1} = \arg\min_{W} g(W) + h(V^t, W)$$

$$V^{t+1} = \arg\min f(V) + h(V, W^{t+1})$$

So in Assignment Q4 you can update S and L in 2 steps!

- Now the problem is dealing with nuclear norm and L1 norm
  - For updating S we just need to deal with L1 norm and like previous assignment we use soft-threshold:

$$S^{t+1} = prox_{\alpha_t, \|S\|_1}\left[S^t - \alpha_t(M - L^t - S^t)\right]$$

- For updating L with nuclear norm, let's consider the following problem:

$$L^{t+1} = \mathrm{prox}_{\|L\|_*}(L^t) = \underset{L}{\mathrm{argmin}} \left\{ \frac{1}{2}\|L - L^t\|_F^2 + \lambda\|L\|_* \right\}$$

- It defines proximal operator in matrix domain with Frobenius norm and nuclear norm!

In-exact intuition and solution!

- Let $L^t = U\Sigma^t V^T$ $\quad \Sigma^t = diag(\sigma_1^t, \dots, \sigma_r^t)$

and we want to find L^* s.t. $L^* = U\Sigma V^T$, $\Sigma = diag(\sigma_1, \dots, \sigma_r)$

so:

$$\underset{L}{\operatorname{argmin}} \{ \|L - L^t\|_F^2 + \lambda \|L\|_* \}$$

$$\equiv \underset{\Sigma}{\operatorname{argmin}} \left\{ \sum_{i=1}^{r} \|\sigma_i - \sigma_i^t\|^2 + \lambda \sum \sigma_i \right\}$$

- Now we can optimize w.r.t. each σ_i.

So $\arg\min_{\sigma_i} \left\{ \frac{1}{2} \| \sigma_i - \sigma_i^t \|^2 + \lambda \sigma_i \right\}$

taking gradient w.r.t. σ_i :

$$\sigma_i - \sigma_i^t + \lambda = 0 \Rightarrow \sigma_i = \sigma_i^t - \lambda$$

but $\sigma_i \geq 0 \Rightarrow \sigma_i^{t+1} = \max\{0, \sigma_i^t - \lambda\}$

↳ It is similar to soft_treshold but on σ_i!

- We can recover the L:

$$L^{t+1} = U \Sigma^{t+1} V^T$$

Exact solution for L

- All objective functions are convex
- Nuclear norm is convex in matrix domain!

Let $L^* = \operatorname{argmin} \left\{ \frac{1}{2} \| L - L^t \|_F^2 + \lambda \| L \|_* \right\}$

Optimality condition:

$$0 \in L^* - L^t + \lambda \underbrace{\partial \| L^* \|_*}_{\text{subgradient of nuclear norm}}$$

Lemma for sub-gradient in matrix space

If $X = U\Sigma V^T$ and $X \in R^{m\times n}$,

$$\partial \|X\|_* = \{UV^T + W : W \in R^{m\times n}, U^T W = 0, WV = 0, \|W\|_F \le 1\}$$

Claim: If $L^t = U_0 \Sigma_0^t V_0^T + U_1 \Sigma_1^t V_1^T$

where U_0, Σ_0 (resp. U_1, Σ_1) are singular vectors corresponding with singular value $> \lambda$ (resp. $\le \lambda$). we have

$$L^* = U_0(\Sigma_0 - \lambda I) V_0^T, \quad W = \lambda^{-1} U_1 \Sigma_1^t V_1^T$$

$$0 = L^* - L^t + \lambda \partial \| L^* \|_*$$

$$L^t - L^* = \lambda U_0 V_0^T + U_1 \Sigma_1^t V_1^T = \lambda (U_0 V_0^T + W)$$

$U_0^T W = 0$, $W V_0 = 0$, $\operatorname{diag}(\Sigma_1^t) \leq \lambda \Rightarrow \|W\| \leq 1$

So: $L^t - L^* \in \lambda \partial \| L^* \|_*$

Note: $L^* = U_0 (\Sigma_0 - \lambda I) V_0^t = U (\Sigma_0^t - \lambda I) V$

So $L^* = U(\text{soft\_threshold}(\ldots))V$

- Termination Condition:
  - Noise On Signal Ratio (NOSR)

$$NOSR = \frac{\| M - L - S \|_F}{\| M \|_F} \leq \delta$$

Fun Time ☺

- Image segmentation
  - You are given an image and asked as a AI expert to segment the image based on the meaningful object existing in the image.

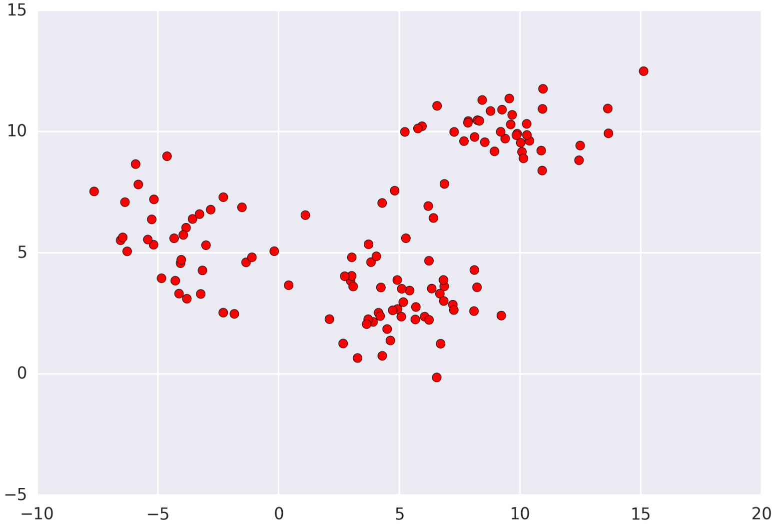
  - How do you do that?

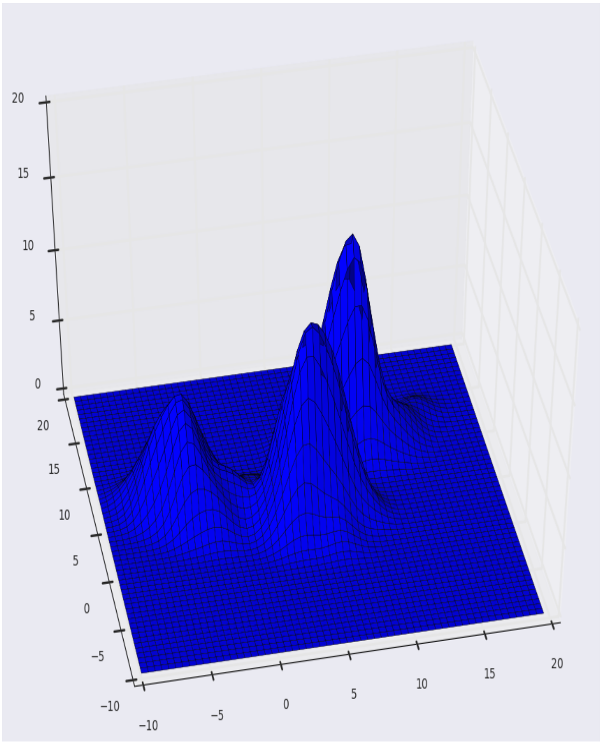
Mean-Shift Clustering

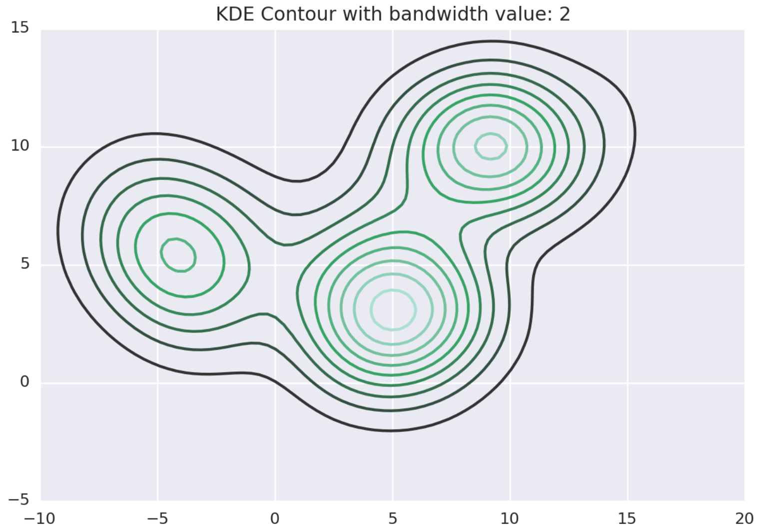
- The problem is a clustering problem so we are looking for a clustering algorithm.
- K-means is a possible answer but you need to know the means and number of cluster already which makes it hard to apply for any arbitrary Image
- An alternative algorithm is Mean-Shift clustering.
  - Basic Idea
  - Estimate a density over the data point using kernel density estimation
    - For this you need to pick kernel parameter: for example bandwidth
  - Find the pick or modes of the density as clusters mean or center.
  - Assign each point to the closest or appropriate center

Mean-Shift Clustering

- Gaussian Kernel Density Estimation







Mean-Shift Clustering

- How can we find modes?
- This is the KDE (assuming the kernel is normalized)

$$f(\mathbf{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) \qquad K(\mathbf{x}) = c_{k,d}\,k(\|\mathbf{x}\|^2)$$

- To find peaks or modes as usual take gradient and set it to 0!

$$\begin{aligned}
\nabla f(\mathbf{x}) &= \frac{2c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}(\mathbf{x}_i-\mathbf{x})\,g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\\
&= \underbrace{\frac{2c_{k,d}}{nh^{d+2}}\left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right]}_{\text{Kernelizing w.r.t. to } G(x)}\underbrace{\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i\, g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}-\mathbf{x}\right]}_{m(x)}
\end{aligned}$$

Mean-Shift Clustering

- Mean shift vector m(x) always moves toward increasing f(x)

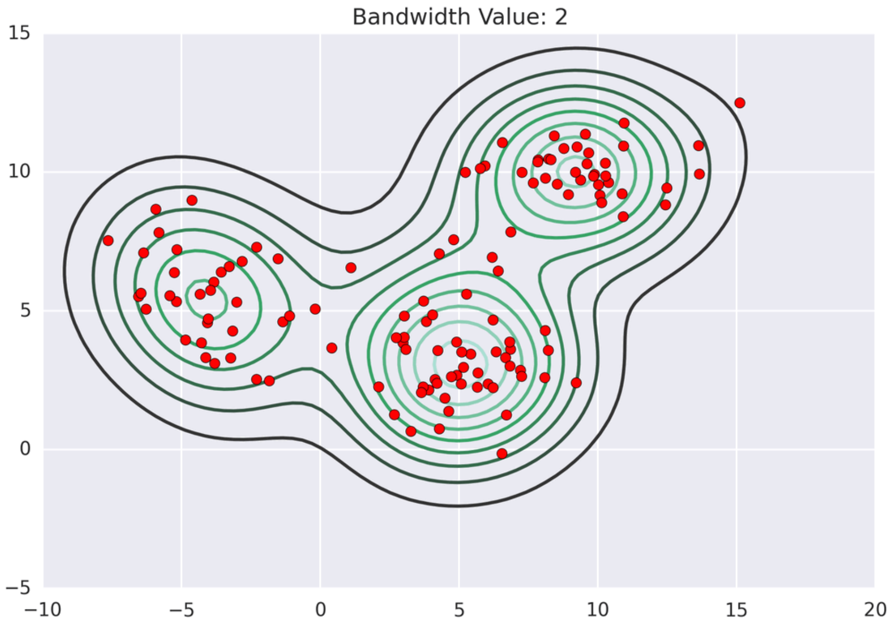
$$\mathbf{m}_h(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x}$$

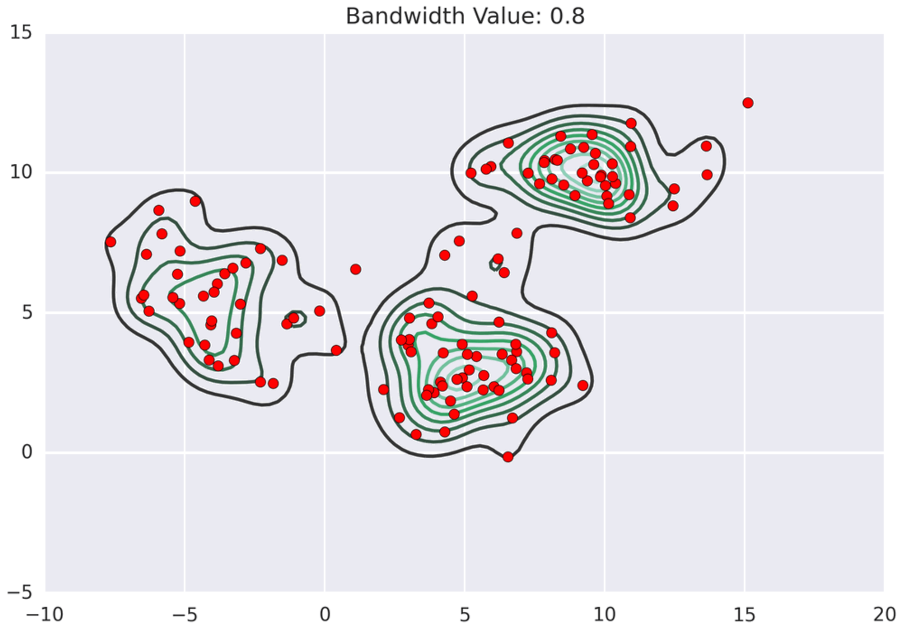
- Mean shift procedure
  - Compute $m_h(x^t)$
  - Move : $x^{t+1} = x^t + \alpha\, m_h(x^t)$

Do this for all points!

Mean-Shift Clustering

- Cluster each point based on the mode that point moves toward!





Q 4 U

- We can use kernel methods when the data sample size is small and our feature set or base function set is huge. But in big data era we don’t have a “small” sample size. How can we use kernels in this settings?
- Think about it and we may discus about it in my next tutorial or maybe Mark will do in class!