Machine Learning (CS 181): 14. Mixture Models

David C. Parkes and Sasha Rush

Spring 2017

Contents

Contents

Review: Unsupervised Learning

- Data $D = \{\mathbf{x}\}_{i=1}^{n}$. No target values.
- Typical goals: understand data, summarize data, identify concepts.
- Last lecture: clustering
  - K-means (simple approach, but inflexible— linear decision boundaries)
  - HAC (flexible, provides dendrograms, but poor performance in high dimensions)

Application: K-means on Image Data

CIFAR-100 color. 50,000 images. 32 x 32 x 3 (RGB), each 0-255.

(a) Cluster Centers (b) Cluster 1 (c) Cluster 2 (d) Cluster 3 (e) Cluster 4

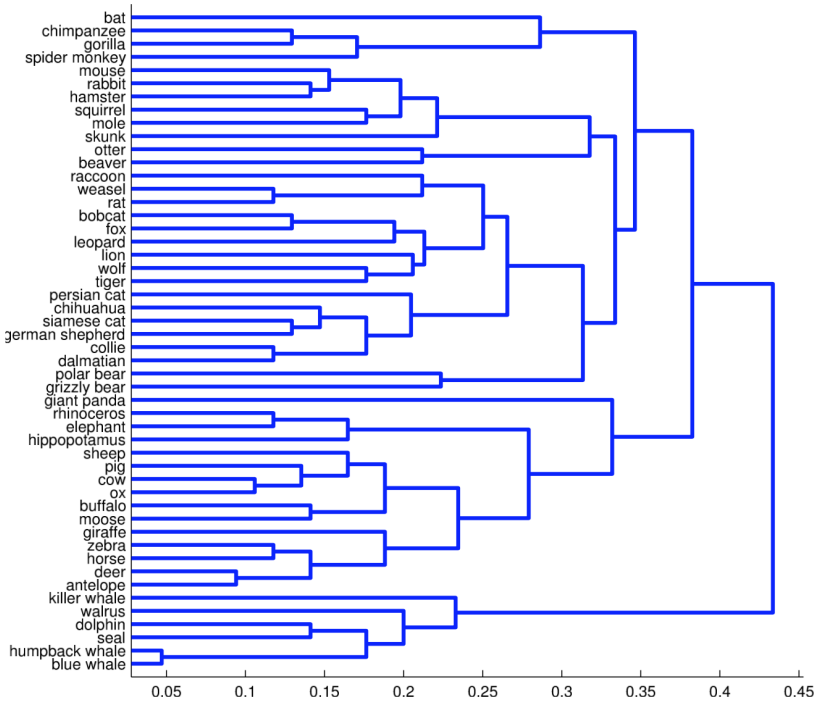
(f) Cluster 5 (g) Cluster 6 (h) Cluster 7 (i) Cluster 8 (j) Cluster 9 (k) Cluster 10

(l) Cluster 11 (m) Cluster 12 (n) Cluster 13 (o) Cluster 14 (p) Cluster 15 (q) Cluster 16

(R.Adams; $K = 16$. Clusters pick up on low-freq color variations)

Application: HAC on Animal Data

Data set of 50 animals, 85 binary features (e.g., longneck, water, smelly)



Contents

A concern about K-means, HAC

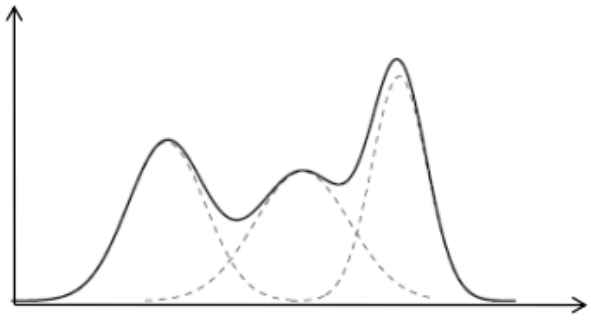
Both seem a bit ad hoc:

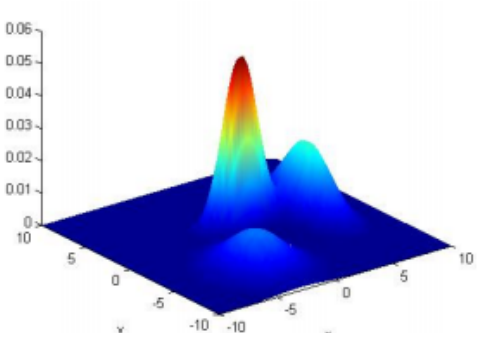
- K-means: finds an assignment of examples to clusters, and cluster prototypes to minimize the total, Euclidean distance between examples and cluster prototypes.
- HAC: adopts a pointwise distance between examples, (e.g., Euclidean or edit distance), and a "group linkage" distance (e.g., min, max, average ...).

Why are these distance measures the right way to think about what makes a good clustering?

How else can we cluster data?

- We can view data as coming from a]mixture over different components of a distribution.





D. Ogreniyorum ($m = 1, m = 2$)

- Use data D to estimate the parameters of this "mixture model." Each component will correspond to a class, or cluster. Given this, can predict the most likely cluster (or class) for each example.

Mixture Models

- Observed data (given): $\mathbf{x}_i \in \mathbb{R}^m$ (features)
- Represent the class of each example as $\mathbf{z}_i$, a one-hot vector. This is latent (unobserved.)
- Generative model (parameters $\mathbf{w}$): first sample a class, then conditioned on the class, generate features

$$p(\mathbf{x}, \mathbf{z}; \mathbf{w}) = p(\mathbf{z}; \mathbf{w}) p(\mathbf{x} \,|\, \mathbf{z}; \mathbf{w})$$

- Given parameters $\mathbf{w}$, we can predict a class via Bayes rule

$$p(\mathbf{z} \,|\, \mathbf{x}; \mathbf{w}) \propto p(\mathbf{x} \,|\, \mathbf{z}; \mathbf{w}) p(\mathbf{z}; \mathbf{w})$$

The Gaussian Mixture Model

- Observed data (given): $\mathbf{x}_i \in \mathbb{R}^m$
- Latent variable (not given): $\mathbf{z}_i \in \{C_k\}_{k=1}^c$, for c clusters
- Class distribution (generalized Bernoulli):

$$p(\mathbf{z} = C_k) = \theta_k, \quad \text{for } k \in \{1, \ldots, c\}$$

- Class conditional distribution (Normal):

$$p(\mathbf{x}|\mathbf{z} = C_k) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \Sigma_k), \quad \text{for } k \in \{1, \ldots, c\}$$

- Parameters of the model: $\boldsymbol{\theta}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^c$
- Given estimated parameters, predict a class via Bayes rule

$$p(\mathbf{z} = C_k \,|\, \mathbf{x}) \propto \theta_k \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Contents

Maximum Likelihood Estimation

- Observed data (given) $\mathbf{x}_i \in \mathbb{R}^m$. Latent variable $\mathbf{z}_i$ (not given)
- First, suppose we see $\mathbf{z}_i$ in the data. We be able to compute the complete-data log likelihood:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n} \ln(p(\mathbf{x}_i, \mathbf{z}_i; \mathbf{w}))$$

- But: $\mathbf{z}_i$s are latent. Cannot maximize this! Rather, we need to maximize the log likelihood on observed data:

$$\sum_{i=1}^{n} \ln(p(\mathbf{x}_i; \mathbf{w})) = \sum_{i=1}^{n} \ln \left(\sum_{k=1}^{c} \theta_k \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

Aside: What if we did know the class labels?

Can write out the probability:

$$p(\mathbf{x}_i, \mathbf{z}_i) = p(\mathbf{z}_i) p(\mathbf{x}_i \mid \mathbf{z}_i)$$

$$p(\mathbf{z}_i) = \prod_{k=1}^{c} \theta_k^{z_{ik}}, \quad p(\mathbf{x}_i \mid \mathbf{z}_i) = \prod_{k=1}^{c} \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{ik}}$$

The complete-data log likelihood is:

$$\sum_{i=1}^{n} \ln(p(\mathbf{x}_i, \mathbf{z}_i; \mathbf{w})) = \sum_{i=1}^{n} \sum_{k=1}^{c} z_{ik} \ln \theta_k + \sum_{i=1}^{n} \sum_{k=1}^{c} z_{ik} \ln \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

We could solve this MLE problem analytically (c.f., HW2):

$$\hat{\theta}_k = \frac{n_k}{n}, \quad \hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i=1}^{n} z_{ik} \mathbf{x}_i, \quad \hat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k} \sum_{i=1}^{n} z_{ik} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top$$

where $n_k = \sum_i z_{ik}$. Very nice!

The Estimation Problem for Gaussian Mixture Model

But we don't know the $\mathbf{z}_i$ values. Rather, we need to maximize:

$$\sum_{i=1}^{n} \ln(p(\mathbf{x}_i; \mathbf{w})) = \sum_{i=1}^{n} \ln \left(\sum_{k=1}^{c} \theta_k \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right),$$

where we marginalize out over latent classes.

This does not have an analytical solution. The log of the sum prevents it decomposing by parameters $\boldsymbol{\theta}$ and $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$.

Even computing a gradient is tricky (note: $\frac{d}{dx} \ln(f(x)) = \frac{1}{f(x)} \frac{df}{dx}$, we end up with a sum of fractions).

Contents

How to proceed?

Iteratively! Guess the class assignments for the data given current parameters, and then improve the estimate of parameters.

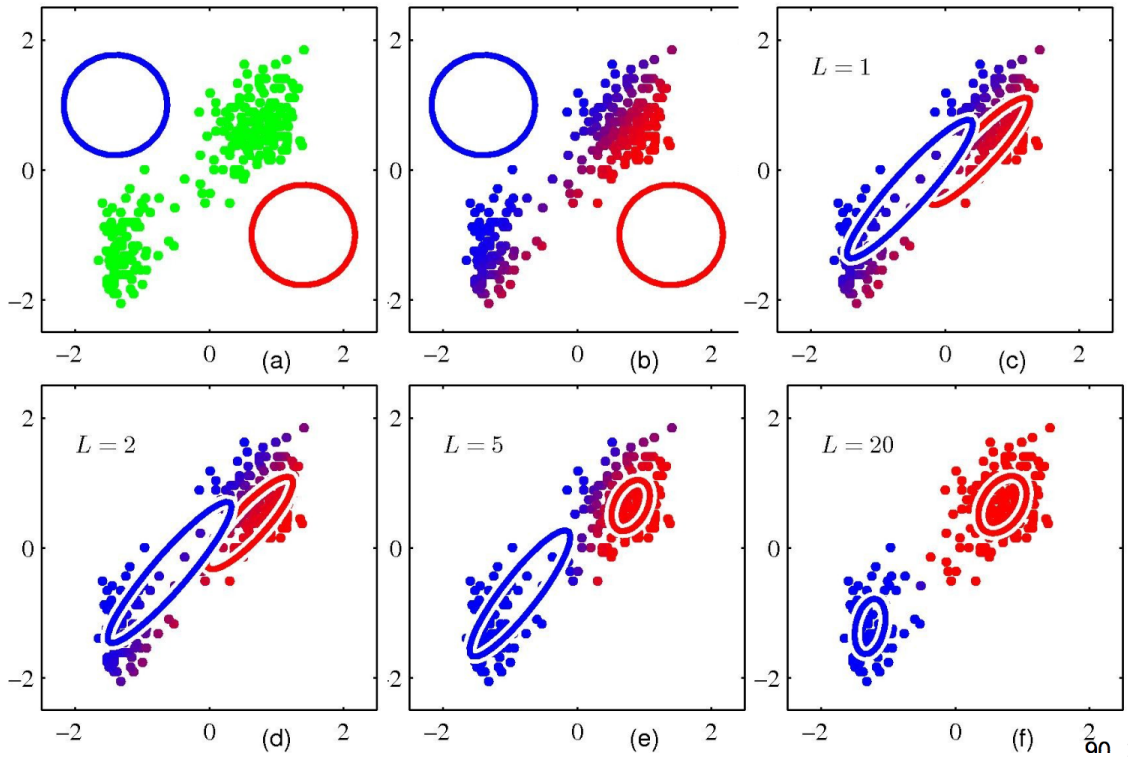
Initialize parameters, then repeat the following two steps:

- (E step) Use the current parameters to predict the assignments $\mathbf{z}_i$, in particular, a distribution on the class for each example.
- (M step) Update the parameters: maximize the "expected complete data log likelihood" using the predicted assignments.

This is the EM algorithm. It is a very powerful, general method to maximize likelihood for models with latent variables. M-step often has an analytical solution.

Illustration: EM on Mixture of Gaussians

(Bishop. Old Faithful data. 1 st. dev. contours. (b) E step. (c) M step; 2, 5, 20 interations.)



The Expected Complete-Data Log Likelihood

(E-step): Given current parameters, estimate "soft class assignments"

$$p(z_i = C_k \,|\, \mathbf{x}_i) = q_{ik} \propto \theta_k \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

the probability that example $\mathbf{x}_i$ is assigned to class k.

Recall $\mathcal{L}(\mathbf{w})$ is complete-data log likelihood. Define the expected complete-data log likelihood as

$$\begin{aligned}\mathbf{E}_{\mathbf{Z}}[\mathcal{L}(\mathbf{w})] &= \mathbf{E}_{\mathbf{Z}}[\sum_{i=1}^{n} \ln(p(\mathbf{x}_i, \mathbf{z}_i; \mathbf{w}))] \\ &= \sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik} \ln \theta_k + \sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik} \ln \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).\end{aligned}$$

The M-Step: Use the Soft Assignment

For complete-data log likelihood, we had the following MLE estimator:

$$\hat{\theta}_k = \frac{n_k}{n}, \quad \hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik}\mathbf{x}_i, \quad \hat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top$$

where $n_k = \sum_i z_{ik}$.

Computing our M-step leads to the following modification:

$$\hat{\theta}_k = \frac{n_k}{n}, \quad \hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i=1}^{n} q_{ik}\mathbf{x}_i, \quad \hat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k}\sum_{i=1}^{n} q_{ik}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top$$

where $n_k = \sum_i q_{ik}$. Still has a nice, analytical solution!

The Expectation Maximization algorithm

Initialize parameters. Then repeat:

- Expectation step: estimate class distribution conditioned on $\mathbf{x}_i$, $p(\mathbf{z}_i \mid \mathbf{x}_i)$, for each latent variable. This is the posterior over the latent variables $\mathbf{z}_i$ conditioned on the observed data $\mathbf{x}_i$, and computed with the current model parameters.
- Maximization step: update parameters $\boldsymbol{\theta}$, $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, to maximize the expected complete-data log likelihood.

Until convergence. Alternate between predicting the class for each example, and updating the parameters of the model.

A general and powerful idea: not specific to mixture of Gaussians (see this for topic models in next class.)

Convergence of EM algorithm

- At each step, the EM algorithm adjusts the parameter values to improve the likelihood of the (observed) data:

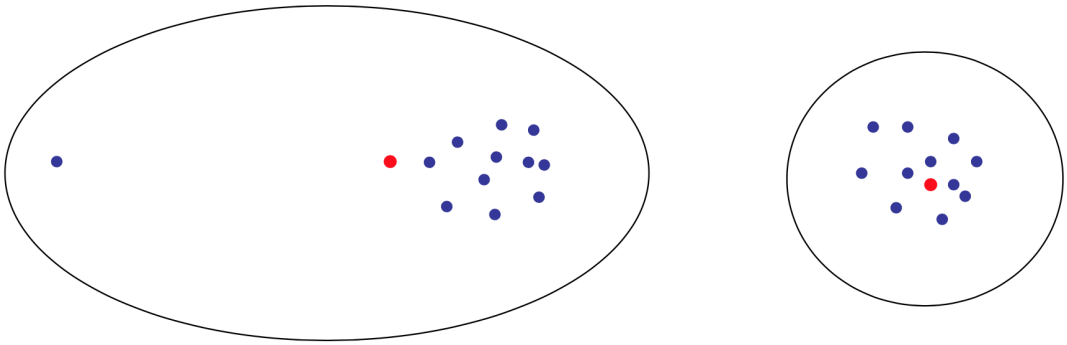
$$p(D; \mathbf{w}^{(0)}) < p(D; \mathbf{w}^{(1)}) < \ldots <$$

- This means that the EM algorithm will converge to a local optimum. Can be usefully combined with random restarts.

Global vs Local optima

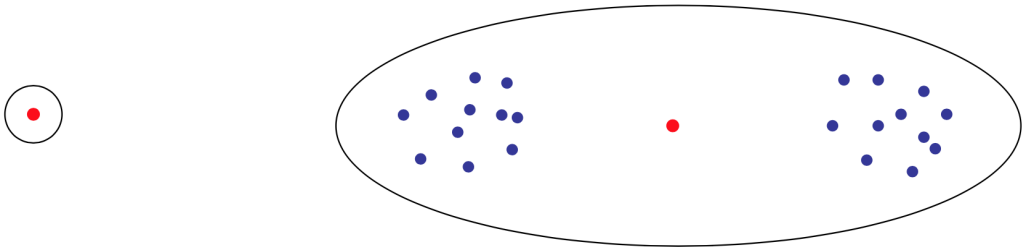
Example: Global vs Local optima

Global optimum ($c = 2$):



Example: Global vs Local optima

Local optimum ($c = 2$):



Note: Initialization of EM for Mixture of Gaussians

E-step:

$$p(z_i = C_k \,|\, \mathbf{x}_i) = q_{ik} \propto \theta_k \, \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

M-step:

$$\hat{\theta}_k = \frac{n_k}{n}, \quad \hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i=1}^{n} q_{ik} \mathbf{x}_i, \quad \hat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k} \sum_{i=1}^{n} q_{ik} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top$$

where $n_k = \sum_i q_{ik}$.

Need asymmetric initialization. What happens if:

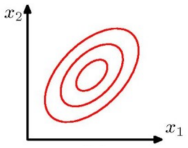
- class probability $\hat{\theta}_k = 1/c$, and $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ equal for each class?
- class probability $\hat{\theta}_1 > \max_{j \neq 1} \hat{\theta}_j$, and $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ equal for each class?

Contents

Variations on Mixture of Gaussians model

- Mixture of Gaussians (General)

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

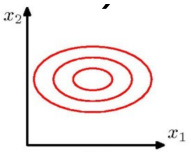


O(m^2) *parameters*

- Mixture of Gaussians (Diagonal)

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

where $\boldsymbol{\Sigma}_k$ is diagonal.

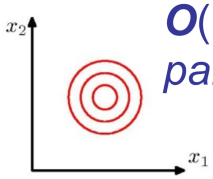


O(m) *parameters*

- Mixture of Gaussians (isotropic, or spherical)

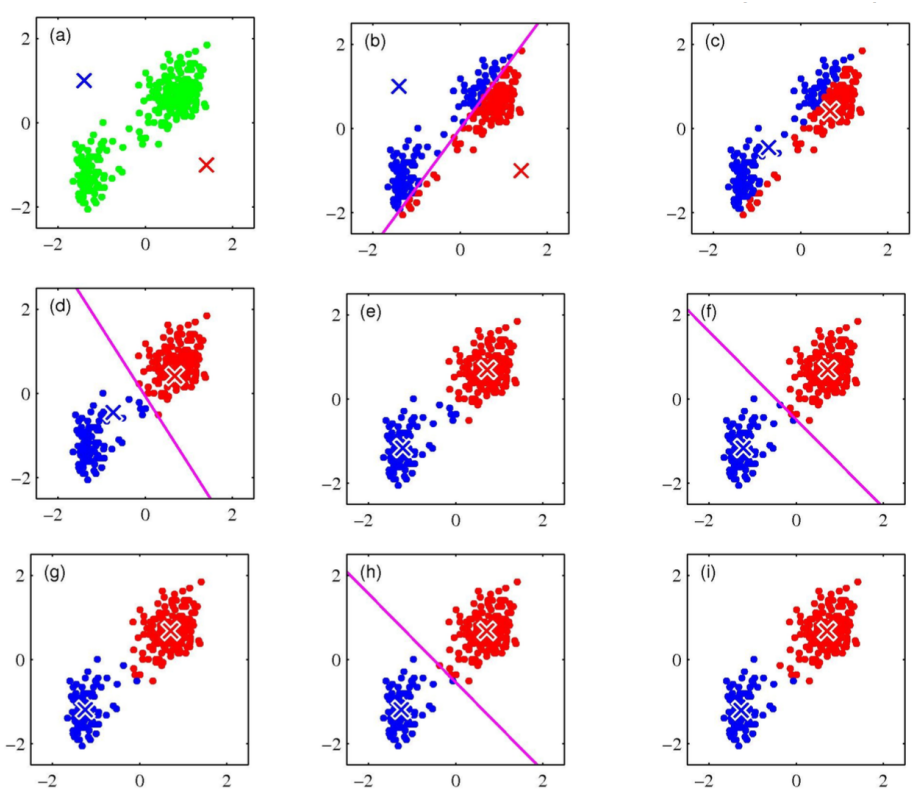
$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \sigma^2\mathbf{I}),$$

where $\mathbf{I}$ is the identity matrix. (Linear decision boundaries.)



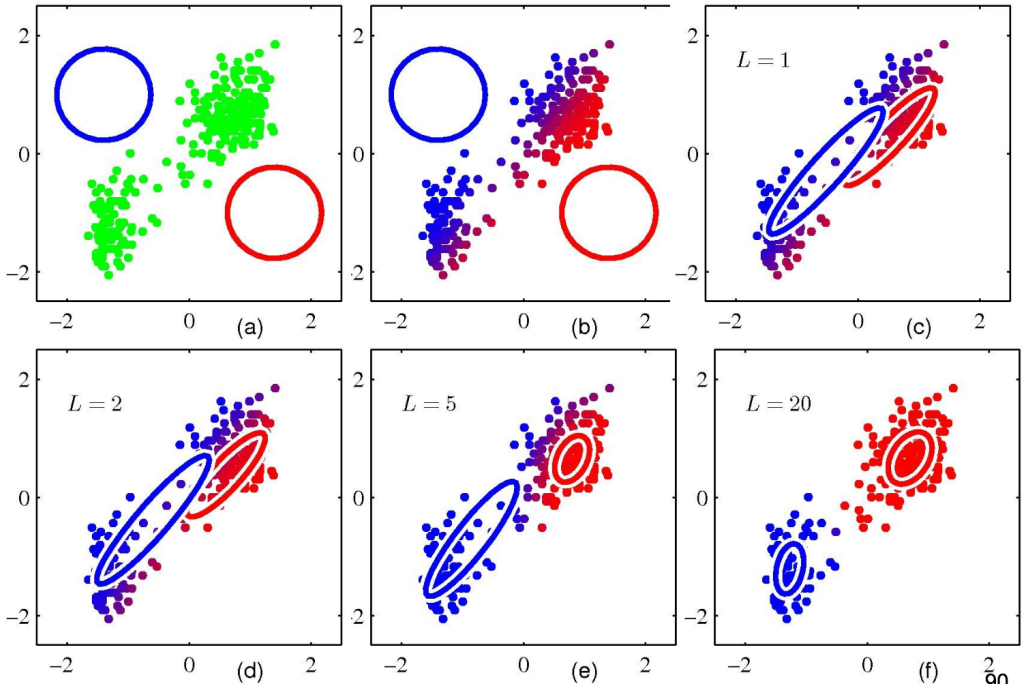
O(m) *parameters*

Recall: K-Means on Old Faithful Eruptions (Bishop)



Compare: EM (Gaussian Mixture) on Old Faithful

(Bishop. Old Faithful data. 1 st. dev. contours. (b) E step. (c) M step; 2, 5, 20 interations.)



Connecting EM and K-Means (Lloyd's algorithm)

K-means:

- (E-step): assign each example to the closest prototype
- (M-step): for each k, update each prototype to the centroid of assigned examples

This is a special case of EM:

- Class probabilities fixed at $1/c$; spherical Gaussians, each with the same, fixed covariance $\epsilon\mathbf{I}$, for small $\epsilon > 0$.
- (E-step) produces a hard assignment (why?)
- (M-step) the centroid maximizes the expected complete-data log likelihood given spherical noise

See that K-means is the limit of EM on Gaussian mixture in which variance parameter $\epsilon \to 0$.

Next class

- Naive Bayes with latent variables (i.e., clustering with discrete features).
- Topic models: each example is a proportion of different 'topics,' and each topic has a topic-conditional distribution on features.
- Graphical representations: a convenient language with which to describe probabilistic models.

Summary

- A mixture model describes a distribution as the average over a number of component distributions.
- The Gaussian mixture model is useful for clustering.
- The MLE problem has no analytical solution because the class variables are latent (unobserved). Ugly expression for log likelihood.
- But: by guessing classes, we can repeatedly update parameters to maximize the expected, complete-data log likelihood.
- This is an application of the EM algorithm. A useful, widely-applicable method for MLE estimation.