Clustering

Professor Ameet Talwalkar

Outline

Schedule and Final

Upcoming Schedule

- Today: Last day of class
- Next Monday (3/13): No class – I will hold office hours in my office (BH 4531F)
- Next Wednesday (3/15): Final Exam, HW6 due

Final Exam

- Cumulative but with more emphasis on new material
- 8 short questions (recall that midterm had 6 short and 3 long questions)
- Should be much shorter than midterm, but of equal difficulty
- Focus on major concepts (e.g., MLE, primal and dual formulations of SVM, gradient descent, etc.)

Outline

Neural Networks – Basic Idea

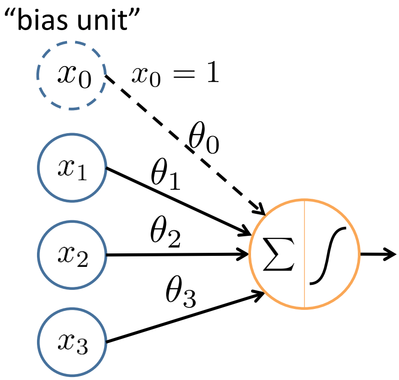
Learning nonlinear basis functions and classifiers

- Hidden layers are nonlinear mappings from input features to new representation
- Output layers use the new representations for classification and regression

Learning parameters

- Backpropogation = efficient algorithm for (stochastic) gradient descent
- Can write down explicit updates via chain rule of calculus

Single Node

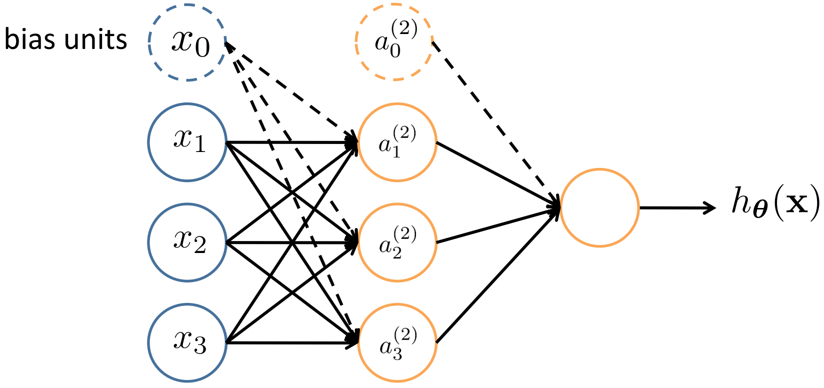


$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(\boldsymbol{\theta}^\intercal \mathbf{x}\right) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\intercal \mathbf{x}}}$$

Sigmoid (logistic) activation function: $g(z) = \frac{1}{1 + e^{-z}}$

Neural Network



Layer 1
(Input Layer)

Layer 2
(Hidden Layer)

Layer 3
(Output Layer)

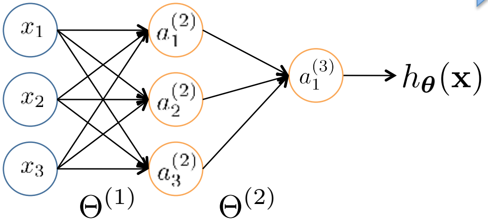
Vectorization

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3\right) = g\left(z_1^{(2)}\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3\right) = g\left(z_2^{(2)}\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3\right) = g\left(z_3^{(2)}\right)$$

$$h_\Theta(\mathbf{x}) = g\left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}\right) = g\left(z_1^{(3)}\right)$$



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$$

$$h_\Theta(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

Learning in NN: Backpropagation

- Similar to the perceptron learning algorithm, we cycle through our examples
  - If the output of the network is correct, no changes are made
  - If there is an error, weights are adjusted to reduce the error

- We are just performing (stochastic) gradient descent!

Based on slide by T. Finin, M. desJardins, L Getoor, R. Par 32

Optimizing the Neural Network

$$J(\Theta) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik}\log(h_\Theta(\mathbf{x}_i))_k + (1-y_{ik})\log\Big(1-(h_\Theta(\mathbf{x}_i))_k\Big)\right]$$
$$+\frac{\lambda}{2n}\sum_{l=1}^{L-1}\sum_{i=1}^{s_{l-1}}\sum_{j=1}^{s_l}\left(\Theta_{ji}^{(l)}\right)^2$$

Solve via: $\min_{\Theta} J(\Theta)$

Unlike before, $J(\Theta)$ is not convex, so GD on a neural net yields a local optimum

$$\frac{\partial}{\partial\Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)}\,\delta_i^{(l+1)}$$ (ignoring λ; if λ = 0)

Forward Propagation

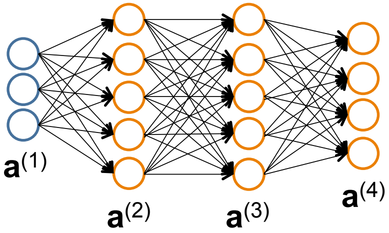
Backpropagation

Forward Propagation

- Given one labeled training instance ($\mathbf{x}$, y):

<u>Forward Propagation</u>

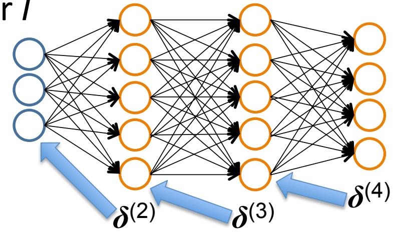
- $\mathbf{a}^{(1)} = \mathbf{x}$
- $\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{a}^{(1)}$
- $\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$ [add $a_0^{(2)}$]
- $\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$
- $\mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$ [add $a_0^{(3)}$]
- $\mathbf{z}^{(4)} = \Theta^{(3)}\mathbf{a}^{(3)}$
- $\mathbf{a}^{(4)} = h_\Theta(\mathbf{x}) = g(\mathbf{z}^{(4)})$



Backpropagation: Gradient Computation

Let $\delta_j^{(l)}$ = "error" of node j in layer l

(#layers $L = 4$)



Element-wise product .*

<u>Backpropagation</u>

- $\delta^{(4)} = \mathbf{a}^{(4)} - \mathbf{y}$
- $\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} \;.\!*\; g'(\mathbf{z}^{(3)})$
- $\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} \;.\!*\; g'(\mathbf{z}^{(2)})$
- (No $\delta^{(1)}$)

$g'(\mathbf{z}^{(3)}) = \mathbf{a}^{(3)} \;.\!*\; (1-\mathbf{a}^{(3)})$

$g'(\mathbf{z}^{(2)}) = \mathbf{a}^{(2)} \;.\!*\; (1-\mathbf{a}^{(2)})$

Training a Neural Network via Gradient Descent with Backprop

```
Given: training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
Initialize all $\Theta^{(l)}$ randomly (NOT to 0!)
Loop // each iteration is called an epoch
    Set $\Delta_{ij}^{(l)} = 0 \quad \forall l, i, j$          (Used to accumulate gradient)
    For each training instance $(\mathbf{x}_i, y_i)$:
        Set $\mathbf{a}^{(1)} = \mathbf{x}_i$
        Compute $\{\mathbf{a}^{(2)}, \ldots, \mathbf{a}^{(L)}\}$ via forward propagation
        Compute $\boldsymbol{\delta}^{(L)} = \mathbf{a}^{(L)} - y_i$
        Compute errors $\{\boldsymbol{\delta}^{(L-1)}, \ldots, \boldsymbol{\delta}^{(2)}\}$
        Compute gradients $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$
    Compute avg regularized gradient $D_{ij}^{(l)} = \begin{cases} \frac{1}{n}\Delta_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)} & \text{if } j \neq 0 \\ \frac{1}{n}\Delta_{ij}^{(l)} & \text{otherwise} \end{cases}$
    Update weights via gradient step $\Theta_{ij}^{(l)} = \Theta_{ij}^{(l)} - \alpha D_{ij}^{(l)}$
Until weights converge or max #epochs is reached
```

Backpropagation

Based on slide by Andrew Ng

Summary of the course so far

Supervised learning has been our focus

- Setup: given a training dataset $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, we learn a function $h(\boldsymbol{x})$ to predict $\boldsymbol{x}$'s true value y (i.e., regression or classification)
- Linear vs. nonlinear features
  1. Linear: $h(\boldsymbol{x})$ depends on $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$
  2. Nonlinear: $h(\boldsymbol{x})$ depends on $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$, where ϕ is either explicit or depends on a kernel function $k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)$
- Loss function
  1. Squared loss: least square for regression (minimizing residual sum of errors)
  2. Logistic loss: logistic regression
  3. Exponential loss: AdaBoost
  4. Margin-based loss: support vector machines
- Principles of estimation
  1. Point estimate: maximum likelihood, regularized likelihood

cont'd

- Optimization
  1. Methods: gradient descent, Newton method
  2. Convex optimization: global optimum vs. local optimum
  3. Lagrange duality: primal and dual formulation
- Learning theory
  1. Difference between training error and generalization error
  2. Overfitting, bias and variance tradeoff
  3. Regularization: various regularized models

Supervised versus Unsupervised Learning

Supervised Learning from labeled observations

- Labels 'teach' algorithm to learn mapping from observations to labels
- Classification, Regression

Supervised versus Unsupervised Learning

Supervised Learning from labeled observations

- Labels 'teach' algorithm to learn mapping from observations to labels
- Classification, Regression

Unsupervised Learning from unlabeled observations

- Learning algorithm must find latent structure from features alone
- Can be goal in itself (discover hidden patterns, exploratory analysis)
- Can be means to an end (preprocessing for supervised task)
- Clustering (Today)

Outline

Clustering

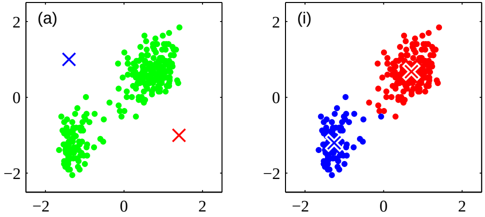
Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^{N}$ and K, we want to output:

Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output:

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.

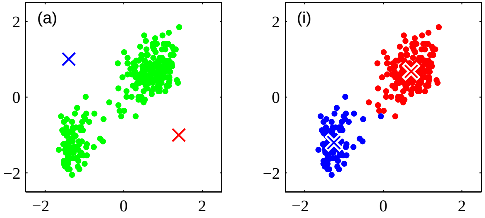


Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output:

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership

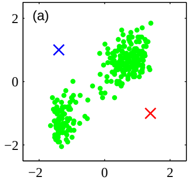
Toy Example Cluster data into two clusters.

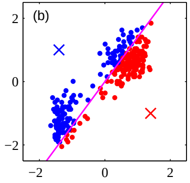


Applications

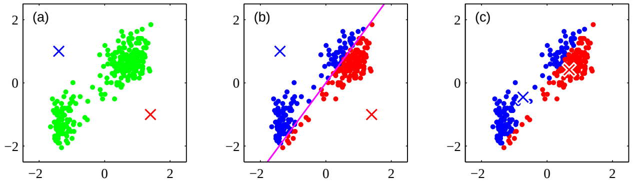
- Identify communities within social networks
- Find topics in news stories
- Group similiar sequences into gene families

K-means example

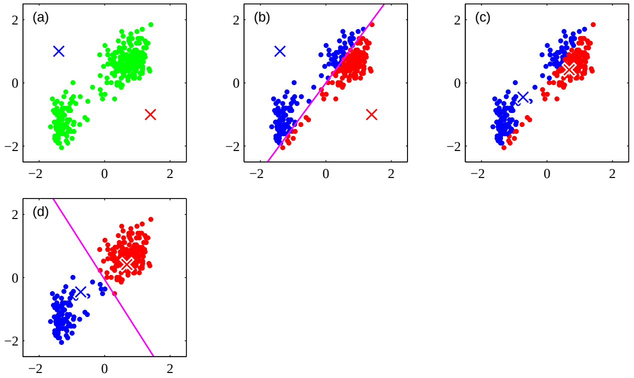




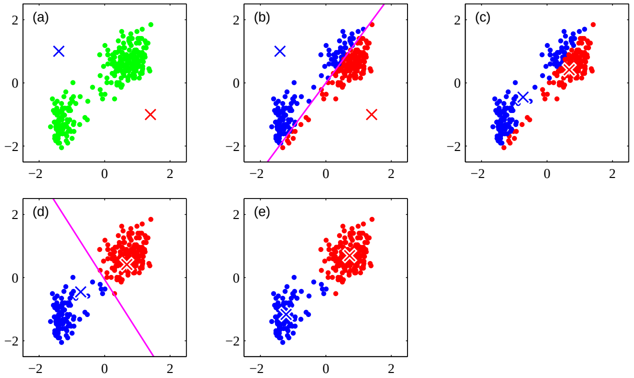
K-means example



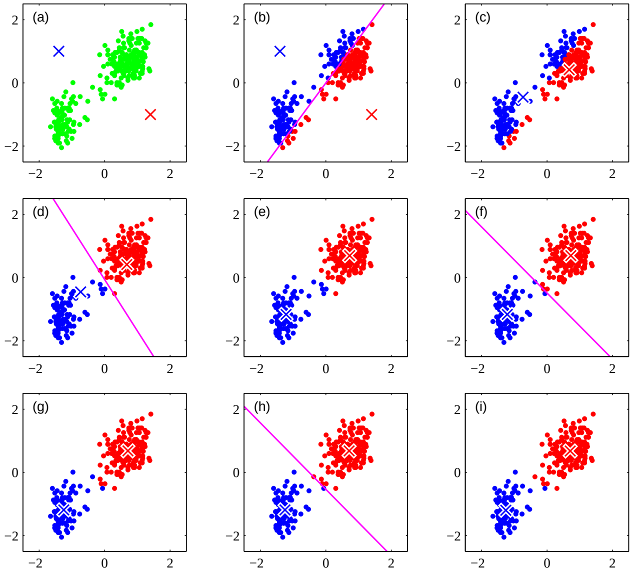
K-means example



K-means example



K-means example



K-means clustering

Intuition Data points assigned to cluster k should be near prototype $\boldsymbol{\mu}_k$

K-means clustering

Intuition Data points assigned to cluster k should be near prototype $\boldsymbol{\mu}_k$

Distortion measure (clustering objective function, cost function)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\boldsymbol{x}_n) = k$$

Algorithm

Minimize distortion Alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values

Algorithm

Minimize distortion Alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values
- **Step 1** Fix $\{\boldsymbol{\mu}_k\}$ and minimize over $\{r_{nk}\}$, to get this assignment:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

Algorithm

Minimize distortion Alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values
- **Step 1** Fix $\{\boldsymbol{\mu}_k\}$ and minimize over $\{r_{nk}\}$, to get this assignment:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- **Step 2** Fix $\{r_{nk}\}$ and minimize over $\{\boldsymbol{\mu}_k\}$ to get this update:

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\boldsymbol{x}_n}{\sum_n r_{nk}}$$

- **Step 3** Return to Step 1 unless stopping criterion is met

Remarks

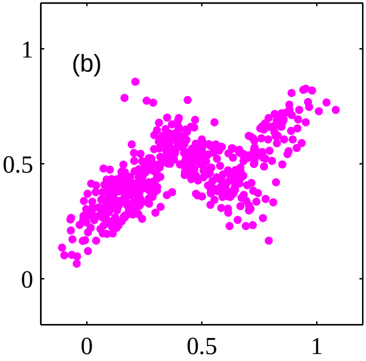
- Prototype $\boldsymbol{\mu}_k$ is the mean of points assigned to cluster k, hence 'K-means'
- The procedure reduces J in both Step 1 and Step 2 and thus makes improvements on each iteration
- No guarantee we find the global solution
- Quality of local optimum depends on initial values at Step 0
- k-means$++$ is a principled approximation algorithm

Probabilistic interpretation of clustering?

How can we model $p(\boldsymbol{x})$ to reflect our intuition that points stay close to their cluster centers?

Probabilistic interpretation of clustering?

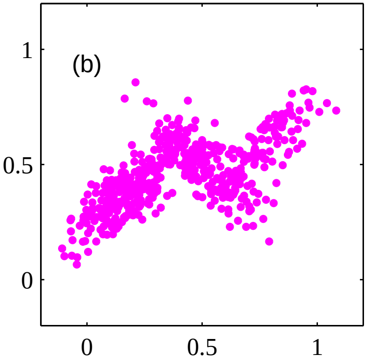
How can we model $p(\boldsymbol{x})$ to reflect our intuition that points stay close to their cluster centers?



- Points seem to form 3 clusters

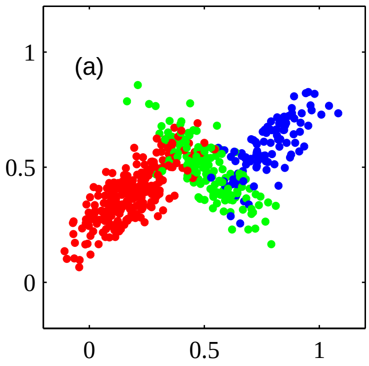
Probabilistic interpretation of clustering?

How can we model $p(\boldsymbol{x})$ to reflect our intuition that points stay close to their cluster centers?



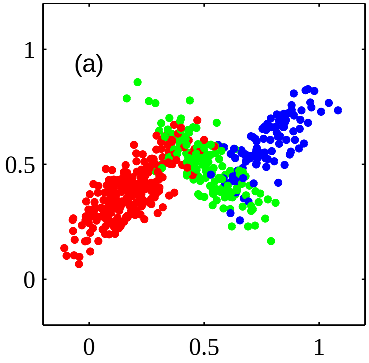
- Points seem to form 3 clusters
- We cannot model $p(\boldsymbol{x})$ with simple and known distributions
- E.g., the data is not a Guassian b/c we have 3 distinct concentrated regions

Gaussian mixture models: intuition



- Model *each* region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)

Gaussian mixture models: intuition



- Model *each* region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)
- We don't know *cluster assignments* (label), *parameters* of Gaussians, or *mixture components*!
- Must learn from *unlabeled* data $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^{N}$

Gaussian mixture models: formal definition

GMM has the following density function for $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- K: number of Gaussians — they are called mixture components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of k-th component

Gaussian mixture models: formal definition

GMM has the following density function for $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- K: number of Gaussians — they are called mixture components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of k-th component
- ω_k: mixture weights (or priors) represent how much each component contributes to final distribution. They satisfy 2 properties:

Gaussian mixture models: formal definition

GMM has the following density function for $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- K: number of Gaussians — they are called mixture components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of k-th component
- ω_k: mixture weights (or priors) represent how much each component contributes to final distribution. They satisfy 2 properties:

$$\forall\ k,\ \omega_k > 0, \quad \text{and} \quad \sum_k \omega_k = 1$$

These properties ensure $p(\boldsymbol{x})$ is in fact a probability density function

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\boldsymbol{x}, z) = p(z)p(\boldsymbol{x}|z)$$

where z is a discrete random variable taking values between 1 and K.

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\boldsymbol{x}, z) = p(z)p(\boldsymbol{x}|z)$$

where z is a discrete random variable taking values between 1 and K. Denote

$$\omega_k = p(z = k)$$

Now, assume the conditional distributions are Gaussian distributions

$$p(\boldsymbol{x}|z = k) = N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\boldsymbol{x}, z) = p(z)p(\boldsymbol{x}|z)$$

where z is a discrete random variable taking values between 1 and K. Denote

$$\omega_k = p(z = k)$$

Now, assume the conditional distributions are Gaussian distributions

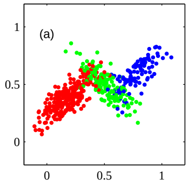
$$p(\boldsymbol{x}|z = k) = N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Then, the marginal distribution of $\boldsymbol{x}$ is

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Namely, the Gaussian mixture model

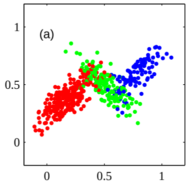
GMMs: example



The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

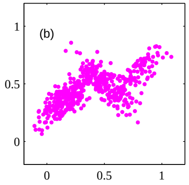
$$p(\boldsymbol{x}|z = red) = N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x}|z = blue) = N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x}|z = green) = N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

GMMs: example



The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

$$p(\boldsymbol{x}|z = red) = N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x}|z = blue) = N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x}|z = green) = N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$



The marginal distribution is thus

$$p(\boldsymbol{x}) = p(red)N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + p(blue)N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) + p(green)N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

Parameter estimation for Gaussian mixture models

The parameters in GMMs are

Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$

Let's first consider the simple/unrealistic case where *we have labels* z

Define $\mathcal{D}' = \{\boldsymbol{x}_n, z_n\}_{n=1}^N$

- $\mathcal{D}'$ is the *complete* data
- $\mathcal{D}$ the *incomplete* data

How can we learn our parameters?

Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$

Let's first consider the simple/unrealistic case where *we have labels* z

Define $\mathcal{D}' = \{\boldsymbol{x}_n, z_n\}_{n=1}^N$

- $\mathcal{D}'$ is the *complete* data
- $\mathcal{D}$ the *incomplete* data

How can we learn our parameters?

Given $\mathcal{D}'$, the maximum likelihood estimation of the $\boldsymbol{\theta}$ is given by

$$\boldsymbol{\theta} = \arg\max \log \mathcal{D}' = \sum_n \log p(\boldsymbol{x}_n, z_n)$$

Parameter estimation for GMMs: complete data

The *complete* likelihood is decomposable

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n)p(\boldsymbol{x}_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(\boldsymbol{x}_n|z_n)$$

where we have grouped data by its values z_n.

Parameter estimation for GMMs: complete data

The *complete* likelihood is decomposable

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n)p(\boldsymbol{x}_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(\boldsymbol{x}_n|z_n)$$

where we have grouped data by its values z_n.

Let $\gamma_{nk} \in \{0, 1\}$ be a binary variable that indicates whether $z_n = k$:

Parameter estimation for GMMs: complete data

The *complete* likelihood is decomposable

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n)p(\boldsymbol{x}_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(\boldsymbol{x}_n|z_n)$$

where we have grouped data by its values z_n.

Let $\gamma_{nk} \in \{0,1\}$ be a binary variable that indicates whether $z_n = k$:

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log p(z=k)p(\boldsymbol{x}_n|z=k)$$

Parameter estimation for GMMs: complete data

The *complete* likelihood is decomposable

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n)p(\boldsymbol{x}_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(\boldsymbol{x}_n|z_n)$$

where we have grouped data by its values z_n.

Let $\gamma_{nk} \in \{0,1\}$ be a binary variable that indicates whether $z_n = k$:

$$\begin{aligned}\sum_n \log p(\boldsymbol{x}_n, z_n) &= \sum_k \sum_n \gamma_{nk} \log p(z=k)p(\boldsymbol{x}_n|z=k) \\ &= \sum_k \sum_n \gamma_{nk} \left[\log \omega_k + \log N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]\end{aligned}$$

Parameter estimation for GMMs: complete data

From our previous discussion, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \left[\log \omega_k + \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]$$

Regrouping, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Parameter estimation for GMMs: complete data

From our previous discussion, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \left[\log \omega_k + \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]$$

Regrouping, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

The term inside the braces depends on k-th component's parameters. It is now easy to show that (left as an exercise) the MLE is:

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

What's the intuition?

Intuition

Since γ_{nk} is binary, the previous solution is nothing but

- ω_k: fraction of total data points whose z_n is k
  - note that $\sum_k \sum_n \gamma_{nk} = N$
- $\boldsymbol{\mu}_k$: mean of all data points whose z_n is k
- $\boldsymbol{\Sigma}_k$: covariance of all data points whose z_n is k

Intuition

Since γ_{nk} is binary, the previous solution is nothing but

- ω_k: fraction of total data points whose z_n is k
  - note that $\sum_k \sum_n \gamma_{nk} = N$
- $\boldsymbol{\mu}_k$: mean of all data points whose z_n is k
- $\boldsymbol{\Sigma}_k$: covariance of all data points whose z_n is k

This intuition will help us develop an algorithm for estimating $\boldsymbol{\theta}$ when we do not know z_n (incomplete data)

Parameter estimation for GMMs: incomplete data

When z_n is not given, we can guess it via the posterior probability

$$p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^{K} p(\boldsymbol{x}_n|z_n = k')p(z_n = k')}$$

Parameter estimation for GMMs: incomplete data

When z_n is not given, we can guess it via the posterior probability

$$p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^{K} p(\boldsymbol{x}_n|z_n = k')p(z_n = k')}$$

To compute the posterior probability, we need to know the parameters $\boldsymbol{\theta}$!

Let's pretend we know the value of the parameters so we can compute the posterior probability.

How is that going to help us?

Estimation with soft γ_{nk}

We define $\gamma_{nk} = p(z_n = k|\boldsymbol{x}_n)$

Estimation with soft γ_{nk}

We define $\gamma_{nk} = p(z_n = k|\boldsymbol{x}_n)$

- Recall that γ_{nk} was previously binary
- Now it's a "soft" assignment of $\boldsymbol{x}_n$ to k-th component
- Each $\boldsymbol{x}_n$ is assigned to a component fractionally according to $p(z_n = k|\boldsymbol{x}_n)$

Estimation with soft γ_{nk}

We define $\gamma_{nk} = p(z_n = k|\boldsymbol{x}_n)$

- Recall that γ_{nk} was previously binary
- Now it's a "soft" assignment of $\boldsymbol{x}_n$ to k-th component
- Each $\boldsymbol{x}_n$ is assigned to a component fractionally according to $p(z_n = k|\boldsymbol{x}_n)$

We now get the same expression for the MLE as before!

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

But remember, we're 'cheating' by using $\boldsymbol{\theta}$ to compute γ_{nk}!

Iterative procedure

Alternate between estimating γ_{nk} and computing parameters

- Step 0: initialize $\boldsymbol{\theta}$ with some values (random or otherwise)
- Step 1: compute γ_{nk} using the current $\boldsymbol{\theta}$
- Step 2: update $\boldsymbol{\theta}$ using the just computed γ_{nk}
- Step 3: go back to Step 1

Iterative procedure

Alternate between estimating γ_{nk} and computing parameters

- Step 0: initialize $\boldsymbol{\theta}$ with some values (random or otherwise)
- Step 1: compute γ_{nk} using the current $\boldsymbol{\theta}$
- Step 2: update $\boldsymbol{\theta}$ using the just computed γ_{nk}
- Step 3: go back to Step 1

This is an example of the *EM algorithm* — a powerful procedure for model estimation with hidden/latent variables

Iterative procedure

Alternate between estimating γ_{nk} and computing parameters

- Step 0: initialize $\boldsymbol{\theta}$ with some values (random or otherwise)
- Step 1: compute γ_{nk} using the current $\boldsymbol{\theta}$
- Step 2: update $\boldsymbol{\theta}$ using the just computed γ_{nk}
- Step 3: go back to Step 1

This is an example of the *EM algorithm* — a powerful procedure for model estimation with hidden/latent variables

Connection with K-means?

Iterative procedure

Alternate between estimating γ_{nk} and computing parameters

- Step 0: initialize $\boldsymbol{\theta}$ with some values (random or otherwise)
- Step 1: compute γ_{nk} using the current $\boldsymbol{\theta}$
- Step 2: update $\boldsymbol{\theta}$ using the just computed γ_{nk}
- Step 3: go back to Step 1

This is an example of the *EM algorithm* — a powerful procedure for model estimation with hidden/latent variables

Connection with K-means?

- GMMs provide probabilistic interpretation for K-means
- K-means is "hard" GMM or GMMs is "soft" K-means
- Posterior γ_{nk} provides a probabilistic assignment for $\boldsymbol{x}_n$ to cluster k