Introduction to Generative Models

Piyush Rai

Machine Learning (CS771A)

Sept 23, 2016
Generative Model

- Defines a probabilistic way that could have “generated” the data

\[ p(z|\Phi) \rightarrow p(x|z,\theta) \rightarrow x \]

- Each observation \( x_n \) is assumed to be associated with a latent variable \( z_n \) (we can think of \( z_n \) as a compact/compressed “encoding” of \( x_n \))

- \( z_n \) is assumed to be a random variable with some prior distribution \( p(z|\phi) \)

- Assume another data distribution \( p(x|z,\theta) \) that can “generate” \( x \) given \( z \)

- What \( x \) and \( z \) “look like”, and the form of the distributions \( p(z|\phi), p(x|z,\theta) \) will be problem-specific (we will soon look at some examples)

- \( \{\theta,\phi\} \) are the unknown model parameters

- The goal will be to learn \( \{\theta,\phi\} \) and \( z_n \)'s, given the observed data
Generative Model

- Defines a **probabilistic way** that could have “generated” the data

\[ p(z|\Phi) \quad \xrightarrow{\text{z}} \quad p(x|z,\theta) \quad \xrightarrow{\text{x}} \quad 4 \]

- Each observation \( x_n \) is assumed to be associated with a **latent variable** \( z_n \) (we can think of \( z_n \) as a compact/compressed “encoding” of \( x_n \))
Generative Model

- Defines a **probabilistic way** that could have “generated” the data

- Each observation $x_n$ is assumed to be associated with a **latent variable** $z_n$ (we can think of $z_n$ as a compact/compressed “encoding” of $x_n$)

- $z_n$ is assumed to be a **random variable** with some **prior distribution** $p(z|\phi)$

![Diagram]

Generative Model

- Defines a **probabilistic way** that could have “generated” the data

\[ p(z|\Phi) \quad \text{p}(x|z,\theta) \quad z \quad x \]

- Each observation \( x_n \) is assumed to be associated with a latent variable \( z_n \) (we can think of \( z_n \) as a compact/compressed “encoding” of \( x_n \))

- \( z_n \) is assumed to be a random variable with some prior distribution \( p(z|\phi) \)

- Assume another data distribution \( p(x|z,\theta) \) that can “generate” \( x \) given \( z \)
**Generative Model**

- Defines a **probabilistic way** that could have “generated” the data

![Diagram of generative model]

- Each observation $x_n$ is assumed to be associated with a **latent variable** $z_n$ (we can think of $z_n$ as a compact/compressed “encoding” of $x_n$)

- $z_n$ is assumed to be a **random variable** with some **prior distribution** $p(z|\phi)$

- Assume another **data distribution** $p(x|z,\theta)$ that can “generate” $x$ given $z$

- What $x$ and $z$ “look like”, and the form of the distributions $p(z|\phi), p(x|z,\theta)$ will be problem-specific (we will soon look at some examples)
Generative Model

- Defines a **probabilistic way** that could have “generated” the data

Each observation \( x_n \) is assumed to be associated with a **latent variable** \( z_n \) (we can think of \( z_n \) as a compact/compressed “encoding” of \( x_n \))

- \( z_n \) is assumed to be a **random variable** with some **prior distribution** \( p(z|\phi) \)

- Assume another **data distribution** \( p(x|z,\theta) \) that can “generate” \( x \) given \( z \)

- What \( x \) and \( z \) “look like”, and the form of the distributions \( p(z|\phi), p(x|z,\theta) \) will be problem-specific (we will soon look at some examples)

- \( \{\theta, \phi\} \) are the unknown model parameters
Generative Model

- Defines a **probabilistic way** that could have “generated” the data

![Diagram of generative model]

- Each observation $x_n$ is assumed to be associated with a **latent variable** $z_n$ (we can think of $z_n$ as a compact/compressed “encoding” of $x_n$)

- $z_n$ is assumed to be a **random variable** with some **prior distribution** $p(z|\phi)$

- Assume another **data distribution** $p(x|z,\theta)$ that can “generate” $x$ given $z$

- What $x$ and $z$ “look like”, and the form of the distributions $p(z|\phi), p(x|z,\theta)$ will be problem-specific (we will soon look at some examples)

- $\{\theta, \phi\}$ are the unknown model parameters

- The goal will be to learn $\{\theta, \phi\}$ and $z_n$’s, given the observed data
Generative models can be described using a “generative story” for the data.

1. First, draw a random latent variable $z_n \sim p(z|\phi)$ from the prior on $z$.
2. Given $z_n$, now generate $x_n \sim p(x|z,\theta)$ from the data distribution.

Such models usually have two types of variables: “local” and “global.”
- Each $z_n$ is a “local” variable (specific to the data point $x_n$).
- $(\phi, \theta)$ are global variables (shared by all the data points).

We may be interested in learning the global vars, or local vars, or both. Usually it’s possible to infer the global vars from local vars (or vice-versa).
Generative models can be described using a “generative story” for the data.

The “generative story” of each observation $x_n$, $\forall n$.

Machine Learning (CS771A)
Generative models can be described using a “generative story” for the data.

The “generative story” of each observation $x_n$, $\forall n$

- First draw a random latent variable $z_n \sim p(z|\phi)$ from the prior on $z$.
Generative models can be described using a “generative story” for the data.

The “generative story” of each observation $x_n, \forall n$

- First draw a random latent variable $z_n \sim p(z|\phi)$ from the prior on $z$
- Given $z_n$, now generate $x_n$ as $x_n \sim p(x|\theta, z_n)$ from the data distribution
Generative models can be described using a “generative story” for the data.

The “generative story” of each observation $x_n, \forall n$

- First draw a random latent variable $z_n \sim p(z|\phi)$ from the prior on $z$
- Given $z_n$, now generate $x_n$ as $x_n \sim p(x|\theta, z_n)$ from the data distribution

Such models usually have two types of variables: “local” and “global”
Generative models can be described using a “generative story” for the data.

The “generative story” of each observation $x_n, \forall n$

- First draw a random latent variable $z_n \sim p(z|\phi)$ from the prior on $z$
- Given $z_n$, now generate $x_n$ as $x_n \sim p(x|\theta, z_n)$ from the data distribution

Such models usually have two types of variables: “local” and “global”

- Each $z_n$ is a “local” variable (specific to the data point $x_n$)
Generative models can be described using a “generative story” for the data.

The “generative story” of each observation $x_n$, $\forall n$

- First draw a random latent variable $z_n \sim p(z|\phi)$ from the prior on $z$
- Given $z_n$, now generate $x_n$ as $x_n \sim p(x|\theta, z_n)$ from the data distribution.

Such models usually have two types of variables: “local” and “global”

- Each $z_n$ is a “local” variable (specific to the data point $x_n$)
- $(\phi, \theta)$ are global variables (shared by all the data points)
Generative models can be described using a “generative story” for the data.

The “generative story” of each observation $x_n, \forall n$
- First draw a random latent variable $z_n \sim p(z|\phi)$ from the prior on $z$.
- Given $z_n$, now generate $x_n$ as $x_n \sim p(x|\theta, z_n)$ from the data distribution.

Such models usually have two types of variables: “local” and “global”.
- Each $z_n$ is a “local” variable (specific to the data point $x_n$).
- $(\phi, \theta)$ are global variables (shared by all the data points).
- We may be interested in learning the global vars, or local vars, or both.
Generative models can be described using a “generative story” for the data.

The “generative story” of each observation $x_n$, $\forall n$:
- First draw a random latent variable $z_n \sim p(z|\phi)$ from the prior on $z$.
- Given $z_n$, now generate $x_n$ as $x_n \sim p(x|\theta, z_n)$ from the data distribution.

Such models usually have two types of variables: “local” and “global”:
- Each $z_n$ is a “local” variable (specific to the data point $x_n$).
- $(\phi, \theta)$ are global variables (shared by all the data points).
- We may be interested in learning the global vars, or local vars, or both.
- Usually it’s possible to infer the global vars from local vars (or vice-versa).
Why Generative Models?

- A proper, probabilistic way to think about the data generation process

![Diagram showing generative model process](http://torch.ch/blog/2015/11/13/gan.html)
Why Generative Models?

- A proper, probabilistic way to think about the data generation process

- Allows modeling different types of data (real, binary, count, etc.) by changing the data distribution \( p(x|\theta, z) \) appropriately

- Can synthesize or "hallucinate" new data using an already learned model

- Generate a "random" \( z \) from \( p(z|\phi) \) and generate \( x \) from \( p(x|\theta, z) \)

- Allows handling missing data (by treating missing data also as latent variable)

Hallucinated faces pic courtesy: http://torch.ch/blog/2015/11/13/gan.html
Why Generative Models?

- A proper, probabilistic way to think about the data generation process

- Allows modeling different types of data (real, binary, count, etc.) by changing the data distribution $p(x|\theta, z)$ appropriately

- Can synthesize or “hallucinate” new data using an already learned model
  - Generate a “random” $z$ from $p(z|\phi)$ and generate $x$ from $p(x|\theta, z)$

Hallucinated faces pic courtesy: http://torch.ch/blog/2015/11/13/gan.html
Why Generative Models?

- A proper, probabilistic way to think about the data generation process

- Allows modeling different types of data (real, binary, count, etc.) by changing the data distribution $p(x|\theta, z)$ appropriately

- Can synthesize or “hallucinate” new data using an already learned model
  - Generate a “random” $z$ from $p(z|\phi)$ and generate $x$ from $p(x|\theta, z)$

- Allows handling missing data (by treating missing data also as latent variable)

Hallucinated faces pic courtesy: http://torch.ch/blog/2015/11/13/gan.html
Some “Canonical” Generative Models

- Mixture model (used in clustering and probability density estimation)
Some “Canonical” Generative Models

- Mixture model (used in clustering and probability density estimation)
- Latent factor model (used in dimensionality reduction)
Some “Canonical” Generative Models

- Mixture model (used in clustering and probability density estimation)

![Mixture model diagram]

- Latent factor model (used in dimensionality reduction)

![Latent factor model diagram]

- Can even combine these (e.g., mixture of latent factor models)
Example: Mixture Model

- Assume data \( \{x_n\}_{n=1}^N \) was generated from a mixture of \( K \) distributions

In a mixture model, \( z \) is discrete so \( p(z|\phi) \) is a multinomial distribution.

The data distribution \( p(x|\theta_z) \) depends on the type of data being modeled.

Mixture models can model complex distributions as superposition of simpler distributions (can be used for density estimation, as well as clustering).
Example: Mixture Model

- Assume data \( \{x_n\}_{n=1}^N \) was generated from a mixture of \( K \) distributions

\[
\begin{align*}
\text{Suppose these } K \text{ distributions are } & \quad p(x|\theta_1), \ldots, p(x|\theta_K) \\
\end{align*}
\]

- In a mixture model, \( z \) is discrete so \( p(z|\phi) \) is a multinomial distribution

Mixture models can model complex distributions as superposition of simpler distributions (can be used for density estimation, as well as clustering).
Example: Mixture Model

- Assume data $\{x_n\}_{n=1}^N$ was generated from a mixture of $K$ distributions

- Suppose these $K$ distributions are $p(x|\theta_1), \ldots, p(x|\theta_K)$

- Don't know which of the $K$ distributions each $x_n$ was generated from
Example: Mixture Model

- Assume data \( \{x_n\}_{n=1}^N \) was generated from a mixture of \( K \) distributions.

- Suppose these \( K \) distributions are \( p(x|\theta_1), \ldots, p(x|\theta_K) \).

- Don’t know which of the \( K \) distributions each \( x_n \) was generated from.

- Consider the following generative story for each \( x_n, n = 1, 2, \ldots, N \).
Example: Mixture Model

- Assume data \( \{x_n\}_{n=1}^N \) was generated from a mixture of \( K \) distributions

- Suppose these \( K \) distributions are \( p(x|\theta_1), \ldots, p(x|\theta_K) \)

- Don't know which of the \( K \) distributions each \( x_n \) was generated from

- Consider the following generative story for each \( x_n, n = 1, 2, \ldots, N \)
  - First choose a mixture component \( z_n \in \{1, 2, \ldots, K\} \) as \( z_n \sim p(z|\phi) \)
Example: Mixture Model

- Assume data \( \{x_n\}_{n=1}^{N} \) was generated from a mixture of \( K \) distributions

- Suppose these \( K \) distributions are \( p(x|\theta_1), \ldots, p(x|\theta_K) \)

- Don't know which of the \( K \) distributions each \( x_n \) was generated from

- Consider the following generative story for each \( x_n, n = 1, 2, \ldots, N \)
  - First choose a mixture component \( z_n \in \{1, 2, \ldots, K\} \) as \( z_n \sim p(z|\phi) \)
  - Now generate \( x_n \) from the mixture component no. \( z_n \) as \( x_n \sim p(x|\theta_{z_n}) \)
Example: Mixture Model

- Assume data \( \{x_n\}_{n=1}^{N} \) was generated from a mixture of \( K \) distributions

\[
\text{Suppose these } K \text{ distributions are } p(x|\theta_1), \ldots, p(x|\theta_K)
\]

- Don't know which of the \( K \) distributions each \( x_n \) was generated from

- Consider the following generative story for each \( x_n, n = 1, 2, \ldots, N \)
  - First choose a mixture component \( z_n \in \{1, 2, \ldots, K\} \) as \( z_n \sim p(z|\phi) \)
  - Now generate \( x_n \) from the mixture component no. \( z_n \) as \( x_n \sim p(x|\theta_{z_n}) \)

- In a mixture model, \( z \) is discrete so \( p(z|\phi) \) is a multinomial distribution
Example: Mixture Model

- Assume data \( \{x_n\}_{n=1}^N \) was generated from a mixture of \( K \) distributions

- Suppose these \( K \) distributions are \( p(x|\theta_1), \ldots, p(x|\theta_K) \)

- Don’t know which of the \( K \) distributions each \( x_n \) was generated from

- Consider the following generative story for each \( x_n, n = 1, 2, \ldots, N \)
  - First choose a mixture component \( z_n \in \{1, 2, \ldots, K\} \) as \( z_n \sim p(z|\phi) \)
  - Now generate \( x_n \) from the mixture component no. \( z_n \) as \( x_n \sim p(x|\theta_{z_n}) \)

- In a mixture model, \( z \) is discrete so \( p(z|\phi) \) is a multinomial distribution

- The data distribution \( p(x|\theta_{z_n}) \) depends on the type of data being modeled
Example: Mixture Model

- Assume data \( \{x_n\}_{n=1}^N \) was generated from a mixture of \( K \) distributions

- Suppose these \( K \) distributions are \( p(x|\theta_1), \ldots, p(x|\theta_K) \)

- Don't know which of the \( K \) distributions each \( x_n \) was generated from

- Consider the following generative story for each \( x_n, \ n = 1, 2, \ldots, N \)
  - First choose a mixture component \( z_n \in \{1, 2, \ldots, K\} \) as \( z_n \sim p(z|\phi) \)
  - Now generate \( x_n \) from the mixture component no. \( z_n \) as \( x_n \sim p(x|\theta_{z_n}) \)

- In a mixture model, \( z \) is discrete so \( p(z|\phi) \) is a multinomial distribution

- The data distribution \( p(x|\theta_{z_n}) \) depends on the type of data being modeled

- Mixture models can model complex distributions as superposition of simpler distributions (can be used for density estimation, as well as clustering).
Example: Latent Factor Model

Assume each $D$-dim $x_n$ generated from a $K$-dim latent factor $z_n$ ($K \ll D$)
Example: Latent Factor Model

- Assume each $D$-dim $x_n$ generated from a $K$-dim latent factor $z_n$ ($K \ll D$)

- Consider the following generative story for each $x_n$, $n = 1, 2, \ldots, N$
Example: Latent Factor Model

- Assume each $D$-dim $x_n$ generated from a $K$-dim latent factor $z_n$ ($K \ll D$)

- Consider the following generative story for each $x_n$, $n = 1, 2, \ldots, N$
  - First generate $z_n$ from a $K$-dim distr. as $z_n \sim p(z|\phi)$
Example: Latent Factor Model

- Assume each $D$-dim $x_n$ generated from a $K$-dim latent factor $z_n$ ($K \ll D$)

- Consider the following generative story for each $x_n$, $n = 1, 2, \ldots, N$
  - First generate $z_n$ from a $K$-dim distr. as $z_n \sim p(z|\phi)$
  - Now generate $x_n$ from a $D$-dim distr. as $x_n \sim p(x|z_n, \theta)$
Example: Latent Factor Model

- Assume each $D$-dim $x_n$ generated from a $K$-dim latent factor $z_n$ ($K \ll D$)

- Consider the following generative story for each $x_n$, $n = 1, 2, \ldots, N$
  - First generate $z_n$ from a $K$-dim distr. as $z_n \sim p(z|\phi)$
  - Now generate $x_n$ from a $D$-dim distr. as $x_n \sim p(x|z_n, \theta)$

- When $p(z|\phi)$ and $p(x|z_n, \theta)$ are Gaussian distributions, this basic generative model is called factor analysis or probabilistic PCA
Example: Latent Factor Model

- Assume each \( D \)-dim \( x_n \) generated from a \( K \)-dim latent factor \( z_n \) (\( K \ll D \))

- Consider the following generative story for each \( x_n, n = 1, 2, \ldots, N \)
  - First generate \( z_n \) from a \( K \)-dim distr. as \( z_n \sim p(z|\phi) \)
  - Now generate \( x_n \) from a \( D \)-dim distr. as \( x_n \sim p(x|z_n, \theta) \)

- When \( p(z|\phi) \) and \( p(x|z_n, \theta) \) are Gaussian distributions, this basic generative model is called **factor analysis** or **probabilistic PCA**

- The choice of \( p(z|\phi) \) and \( p(x|z_n, \theta) \) in general will be problem dependent

Machine Learning (CS771A)
Example: Latent Factor Model

- Assume each $D$-dim $x_n$ generated from a $K$-dim latent factor $z_n$ ($K \ll D$)

Consider the following generative story for each $x_n$, $n = 1, 2, \ldots, N$

- First generate $z_n$ from a $K$-dim distr. as $z_n \sim p(z|\phi)$
- Now generate $x_n$ from a $D$-dim distr. as $x_n \sim p(x|z_n, \theta)$

When $p(z|\phi)$ and $p(x|z_n, \theta)$ are Gaussian distributions, this basic generative model is called factor analysis or probabilistic PCA

- The choice of $p(z|\phi)$ and $p(x|z_n, \theta)$ in general will be problem dependent
- Many recent advances in generative models (e.g., deep generative models, generative adversarial networks, etc) are based on these basic principles
Going Forward..

- We will look at, in more detail, some specific generative models

  - Gaussian mixture model (for clustering and density estimation)
  - Factor Analysis and Probabilistic PCA (for dimensionality reduction)

- We will also look at how to do parameter estimation in such models

  - One common approach is to perform MLE/MAP
  - However, presence of latent variables $z$ makes MLE/MAP hard
  - Reason: Since $z$ is a random variable, we must sum over all possible values of $z$ when doing MLE/MAP for the model parameters $\theta, \phi$

$$\log p(x|\theta, \phi) = \log \sum_z p(x|z, \theta) p(z|\phi)$$ (Log can't go inside the summation!)

- Expectation Maximization (EM) algorithm gives a way to solve the problem

  - Basic idea in EM: Instead of summing over all possibilities of $z$, make a "guess" $\tilde{z}$ and maximize

$$\log p(x, \tilde{z}|\theta, \phi)$$ w.r.t. $\theta, \phi$ to learn $\theta, \phi$. Use these values of $\theta, \phi$ to refine your guess $\tilde{z}$ and repeat until convergence.
Going Forward..

- We will look at, in more detail, some specific generative models
  - Gaussian mixture model (for clustering and density estimation)

Expectation Maximization (EM) algorithm gives a way to solve the problem.

Basic idea in EM: Instead of summing over all possibilities of $z$, make a "guess" $\tilde{z}$ and maximize

$$\log p(x, \tilde{z} | \theta, \varphi)$$

w.r.t. $\theta, \varphi$ to learn $\theta, \varphi$. Use these values of $\theta, \varphi$ to refine your guess $\tilde{z}$ and repeat until convergence.
We will look at, in more detail, some specific generative models

- Gaussian mixture model (for clustering and density estimation)
- Factor Analysis and Probabilistic PCA (for dimensionality reduction)

Expectation Maximization (EM) algorithm gives a way to solve the problem. Basic idea in EM: Instead of summing over all possibilities of $z$, make a "guess" $\tilde{z}$ and maximize

$$\log p(x, \tilde{z} | \theta, \phi)$$

w.r.t. $\theta, \phi$ to learn $\theta, \phi$. Use these values of $\theta, \phi$ to refine your guess $\tilde{z}$ and repeat until convergence.
Going Forward..

- We will look at, in more detail, some specific generative models
  - Gaussian mixture model (for clustering and density estimation)
  - Factor Analysis and Probabilistic PCA (for dimensionality reduction)
- We will also look at how to do parameter estimation in such models
We will look at, in more detail, some specific generative models:

- Gaussian mixture model (for clustering and density estimation)
- Factor Analysis and Probabilistic PCA (for dimensionality reduction)

We will also look at how to do parameter estimation in such models:

- One common approach is to perform MLE/MAP
Going Forward..

- We will look at, in more detail, some specific generative models
  - Gaussian mixture model (for clustering and density estimation)
  - Factor Analysis and Probabilistic PCA (for dimensionality reduction)
- We will also look at how to do parameter estimation in such models
  - One common approach is to perform MLE/MAP
  - However, presence of latent variables $z$ makes MLE/MAP hard
Going Forward..

- We will look at, in more detail, some specific generative models
  - Gaussian mixture model (for clustering and density estimation)
  - Factor Analysis and Probabilistic PCA (for dimensionality reduction)
- We will also look at how to do parameter estimation in such models
  - One common approach is to perform MLE/MAP
  - However, presence of latent variables $z$ makes MLE/MAP hard
  - Reason: Since $z$ is a random variable, we must sum over all possible values of $z$ when doing MLE/MAP for the model parameters $\theta, \phi$

$$\log p(x|\theta, \phi) = \log \sum_z p(x|z, \theta)p(z|\phi) \quad \text{(Log can’t go inside the summation!)}$$
Going Forward..

- We will look at, in more detail, some specific generative models
  - Gaussian mixture model (for clustering and density estimation)
  - Factor Analysis and Probabilistic PCA (for dimensionality reduction)
- We will also look at how to do parameter estimation in such models
  - One common approach is to perform MLE/MAP
  - However, presence of latent variables $z$ makes MLE/MAP hard
  - Reason: Since $z$ is a random variable, we must sum over all possible values of $z$ when doing MLE/MAP for the model parameters $\theta, \phi$

$$
\log p(x|\theta, \phi) = \log \sum_z p(x|z, \theta)p(z|\phi) \quad \text{(Log can’t go inside the summation!)}
$$
- **Expectation Maximization (EM) algorithm** gives a way to solve the problem
Going Forward..

- We will look at, in more detail, some specific generative models
  - Gaussian mixture model (for clustering and density estimation)
  - Factor Analysis and Probabilistic PCA (for dimensionality reduction)

- We will also look at how to do parameter estimation in such models
  - One common approach is to perform MLE/MAP
  - However, presence of latent variables $z$ makes MLE/MAP hard
  - Reason: Since $z$ is a random variable, we must sum over all possible values of $z$ when doing MLE/MAP for the model parameters $\theta, \phi$
    \[
    \log p(x|\theta, \phi) = \log \sum_z p(x|z, \theta)p(z|\phi) \quad \text{(Log can't go inside the summation!)}
    \]
  - Expectation Maximization (EM) algorithm gives a way to solve the problem
    - Basic idea in EM: Instead of summing over all possibilities of $z$, make a “guess” $\tilde{z}$ and maximize $\log p(x, \tilde{z}|\theta, \phi)$ w.r.t. $\theta, \phi$ to learn $\theta, \phi$. Use these values of $\theta, \phi$ to refine your guess $\tilde{z}$ and repeat until convergence.
Gaussian Mixture Model (GMM)

- Assume the data is generated from a mixture of $K$ Gaussians

- Each Gaussian represents a “cluster” in the data
Gaussian Mixture Model (GMM)

- Assume the data is generated from a mixture of $K$ Gaussians

- Each Gaussian represents a “cluster” in the data

- The distribution $p(x)$ will be a weighted a mixture of $K$ Gaussians

$$p(x) = \sum_z p(x, z) = \sum_z p(z)p(x|z) = \sum_{k=1}^{K} p(z = k)p(x, z = k) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k)$$
Gaussian Mixture Model (GMM)

- Assume the data is generated from a mixture of $K$ Gaussians

Each Gaussian represents a “cluster” in the data

The distribution $p(x)$ will be a weighted a mixture of $K$ Gaussians

$$p(x) = \sum_z p(x, z) = \sum_z p(z)p(x|z) = \sum_{k=1}^K p(z = k)p(x, z = k) = \sum_{k=1}^K \pi_k N(x|\mu_k, \Sigma_k)$$

where $\pi_k$’s are the mixing weights: $\sum_{k=1}^K \pi_k = 1, \pi_k \geq 0$ (intuitively, $\pi_k = p(z = k)$ is the fraction of data generated by the $k$-th distribution)
Gaussian Mixture Model (GMM)

- Assume the data is generated from a mixture of $K$ Gaussians

Each Gaussian represents a “cluster” in the data

The distribution $p(x)$ will be a weighted a mixture of $K$ Gaussians

$$p(x) = \sum_z p(x, z) = \sum_z p(z)p(x|z) = \sum_{k=1}^K p(z = k)p(x, z = k) = \sum_{k=1}^K \pi_k \mathcal{N}(x | \mu_k, \Sigma_k)$$

where $\pi_k$’s are the mixing weights: $\sum_{k=1}^K \pi_k = 1, \pi_k \geq 0$ (intuitively, $\pi_k = p(z = k)$ is the fraction of data generated by the $k$-th distribution)

- The goal is to learn the params $\{\mu_k, \Sigma_k\}_{k=1}^K$ of these $K$ Gaussians, the mixing weights $\{\pi_k\}_{k=1}^K$, and/or the cluster assignment $z_n$ of each $x_n$
Gaussian Mixture Model (GMM)

- Assume the data is generated from a mixture of $K$ Gaussians

- Each Gaussian represents a “cluster” in the data

- The distribution $p(x)$ will be a weighted a mixture of $K$ Gaussians

$$p(x) = \sum_z p(x, z) = \sum_z p(z)p(x|z) = \sum_{k=1}^{K} p(z = k)p(x, z = k) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k)$$

  where $\pi_k$'s are the **mixing weights**: $\sum_{k=1}^{K} \pi_k = 1, \pi_k \geq 0$ (intuitively, $\pi_k = p(z = k)$ is the fraction of data generated by the $k$-th distribution)

- The goal is to learn the params $\{\mu_k, \Sigma_k\}_{k=1}^{K}$ of these $K$ Gaussians, the mixing weights $\{\pi_k\}_{k=1}^{K}$, and/or the cluster assignment $z_n$ of each $x_n$

- GMM in many ways improves over $K$-means clustering
GMM Clustering: Pictorially

Some synthetically generated data (top-left) generated from a mixture of 3 overlapping Gaussians (top-right).

Notice the “mixed” colored points in the overlapping regions in the final clustering.
Next Class

- GMM in more detail. Extensions of GMM.
- Parameter estimation in GMM
- The Expectation Maximization (EM) algorithm