Latent Variable Models for Dimensionality Reduction

Piyush Rai

Introduction to Machine Learning (CS771A)

October 4, 2018
Recap: Latent Variable Models, ALT-OPT, and EM

- We saw that doing MLE/MAP for latent variable models is difficult in general

\[
\Theta = \arg\max_{\Theta} \log p(X|\Theta) = \arg\max_{\Theta} \log \sum_{Z} p(X, Z|\Theta) \quad \text{(if } Z \text{ is discrete)}
\]

\[
= \arg\max_{\Theta} \log \int_{Z} p(X, Z|\Theta) dZ \quad \text{(if } Z \text{ is continuous)}
\]
Recap: Latent Variable Models, ALT-OPT, and EM

- We saw that doing MLE/MAP for latent variable models is difficult in general

\[ \Theta = \arg \max_{\Theta} \log p(X|\Theta) = \arg \max_{\Theta} \log \sum_{Z} p(X, Z|\Theta) \quad \text{(if } Z \text{ is discrete)} \]

\[ = \arg \max_{\Theta} \log \int_{Z} p(X, Z|\Theta) dZ \quad \text{(if } Z \text{ is continuous)} \]

- We saw that ALT-OPT and EM can be two ways to make MLE/MAP easier in such models
Recap: Latent Variable Models, ALT-OPT, and EM

- We saw that doing MLE/MAP for latent variable models is difficult in general

\[
\Theta = \arg\max_{\Theta} \log p(X|\Theta) = \arg\max_{\Theta} \log \sum_{Z} p(X, Z|\Theta) \quad \text{(if } Z \text{ is discrete)}
\]

\[
= \arg\max_{\Theta} \log \int_{Z} p(X, Z|\Theta) dZ \quad \text{(if } Z \text{ is continuous)}
\]

- We saw that ALT-OPT and EM can be two ways to make MLE/MAP easier in such models

- At a high-level, they solve for the MLE of \( \Theta \) by solving a slightly modified problem
Recap: Latent Variable Models, ALT-OPT, and EM

- We saw that doing MLE/MAP for latent variable models is difficult in general

\[ \Theta = \arg \max_\Theta \log p(X|\Theta) = \arg \max_\Theta \log \sum_Z p(X,Z|\Theta) \quad \text{(if } Z \text{ is discrete)} \]

\[ = \arg \max_\Theta \log \int_Z p(X,Z|\Theta) dZ \quad \text{(if } Z \text{ is continuous)} \]

- We saw that ALT-OPT and EM can be two ways to make MLE/MAP easier in such models

- At a high-level, they solve for the MLE of \( \Theta \) by solving a slightly modified problem

ALT-OPT: \( \hat{\Theta} = \arg \max_\Theta \log p(X, \hat{Z}|\Theta) \)
Recap: Latent Variable Models, ALT-OPT, and EM

We saw that doing MLE/MAP for latent variable models is difficult in general

$$
\Theta = \arg \max_{\Theta} \log p(\mathbf{X}|\Theta) = \arg \max_{\Theta} \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta) \quad \text{(if } \mathbf{Z} \text{ is discrete)}
$$

$$
= \arg \max_{\Theta} \log \int_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta) d\mathbf{Z} \quad \text{(if } \mathbf{Z} \text{ is continuous)}
$$

We saw that ALT-OPT and EM can be two ways to make MLE/MAP easier in such models

At a high-level, they solve for the MLE of $\Theta$ by solving a slightly modified problem

ALT-OPT: $\hat{\Theta} = \arg \max_{\Theta} \log p(\mathbf{X}, \hat{\mathbf{Z}}|\Theta)$ (where $\hat{\mathbf{Z}}$ is a “good” estimate of $\mathbf{Z}$)
Recap: Latent Variable Models, ALT-OPT, and EM

- We saw that doing MLE/MAP for latent variable models is difficult in general

\[ \Theta = \arg \max_\Theta \log p(X|\Theta) = \arg \max_\Theta \log \sum_Z p(X, Z|\Theta) \quad \text{(if Z is discrete)} \]

\[ = \arg \max_\Theta \log \int_Z p(X, Z|\Theta) dZ \quad \text{(if Z is continuous)} \]

- We saw that ALT-OPT and EM can be two ways to make MLE/MAP easier in such models

- At a high-level, they solve for the MLE of \( \Theta \) by solving a slightly modified problem

\begin{align*}
\text{ALT-OPT:} & \quad \hat{\Theta} = \arg \max_\Theta \log p(X, \hat{Z}|\Theta) \quad \text{(where \( \hat{Z} \) is a “good” estimate of \( Z \))} \\
\text{EM:} & \quad \hat{\Theta} = \arg \max_\Theta \mathbb{E}_{p(Z|X, \Theta)}[\log p(X, Z|\Theta)]
\end{align*}
Recap: Latent Variable Models, ALT-OPT, and EM

- We saw that doing MLE/MAP for latent variable models is difficult in general

\[
\Theta = \arg \max_{\Theta} \log p(X|\Theta) = \arg \max_{\Theta} \log \sum_{Z} p(X, Z|\Theta) \quad \text{(if } Z \text{ is discrete)}
\]

\[
= \arg \max_{\Theta} \log \int_{Z} p(X, Z|\Theta) dZ \quad \text{(if } Z \text{ is continuous)}
\]

- We saw that ALT-OPT and EM can be two ways to make MLE/MAP easier in such models

- At a high-level, they solve for the MLE of \( \Theta \) by solving a slightly modified problem

ALT-OPT: \( \hat{\Theta} = \arg \max_{\Theta} \log p(X, \hat{Z}|\Theta) \) (where \( \hat{Z} \) is a “good” estimate of \( Z \))

EM: \( \hat{\Theta} = \arg \max_{\Theta} \mathbb{E}_{p(Z|X, \Theta)}[\log p(X, Z|\Theta)] \)

- But since \( Z \) and \( \Theta \) are usually “coupled”, both ALT-OPT and EM need an alternating procedure
Recap: Latent Variable Models, ALT-OPT, and EM

- We saw that doing MLE/MAP for latent variable models is difficult in general

  \[ \Theta = \arg \max_\Theta \log p(X|\Theta) = \arg \max_\Theta \log \sum_Z p(X, Z|\Theta) \quad (\text{if } Z \text{ is discrete}) \]

  \[ = \arg \max_\Theta \log \int_Z p(X, Z|\Theta) dZ \quad (\text{if } Z \text{ is continuous}) \]

- We saw that ALT-OPT and EM can be two ways to make MLE/MAP easier in such models

- At a high-level, they solve for the MLE of \( \Theta \) by solving a slightly modified problem

  ALT-OPT: \( \hat{\Theta} = \arg \max_\Theta \log p(X, \hat{Z}|\Theta) \) (where \( \hat{Z} \) is a "good" estimate of \( Z \))

  EM: \( \hat{\Theta} = \arg \max_\Theta \mathbb{E}_{p(Z|X, \Theta)}[\log p(X, Z|\Theta)] \)

- But since \( Z \) and \( \Theta \) are usually "coupled", both ALT-OPT and EM need an alternating procedure

- Instead of maximizing \( \log p(X|\Theta) \) (incomplete-data log-lik - ILL), they maximize \( \log p(X, Z|\Theta) \) (complete-data log-lik - CLL) or expected CLL
Recap: Latent Variable Models, ALT-OPT, and EM

We saw that doing MLE/MAP for latent variable models is difficult in general

\[ \Theta = \arg \max_{\Theta} \log p(X|\Theta) = \arg \max_{\Theta} \log \sum_Z p(X, Z|\Theta) \quad \text{(if } Z \text{ is discrete)} \]

\[ = \arg \max_{\Theta} \log \int_Z p(X, Z|\Theta) dZ \quad \text{(if } Z \text{ is continuous)} \]

We saw that ALT-OPT and EM can be two ways to make MLE/MAP easier in such models

At a high-level, they solve for the MLE of \( \Theta \) by solving a slightly modified problem

**ALT-OPT:** \( \hat{\Theta} = \arg \max_{\Theta} \log p(X, \hat{Z}|\Theta) \quad \text{(where } \hat{Z} \text{ is a “good” estimate of } Z) \)

**EM:** \( \hat{\Theta} = \arg \max_{\Theta} \mathbb{E}_{p(Z|X, \Theta)} [\log p(X, Z|\Theta)] \)

But since \( Z \) and \( \Theta \) are usually “coupled”, both ALT-OPT and EM need an alternating procedure

Instead of maximizing \( \log p(X|\Theta) \) (incomplete-data log-lik - ILL), they maximize \( \log p(X, Z|\Theta) \) (complete-data log-lik - CLL) or expected CLL

For most models, arg max of CLL or expected CLL is usually much easier than arg max of ILL
Recap: ALT-OPT and EM

ALT-OPT does the following

1. Initialize $\Theta = \hat{\Theta}$

2. Estimate $Z$ as $\hat{Z} = \arg \max_Z \log p(Z | X, \hat{\Theta})$

3. Estimate $\Theta$ as $\hat{\Theta} = \arg \max_\Theta \log p(X, \hat{Z} | \Theta)$

4. Go to step 2 if not converged

Step 2 (arg max) of ALT-OPT could potentially throw away a lot of information about $Z$

EM addresses it using “soft” version of ALT-OPT

1. Initialize $\Theta = \hat{\Theta}$

2. Compute the posterior distribution of $Z$, i.e., $p(Z | X, \hat{\Theta})$

3. Estimate $\Theta$ by maximizing the expected CLL $\hat{\Theta} = \mathbb{E}_{p(Z | X, \hat{\Theta})} \left[ \log p(X, Z | \Theta) \right]$

4. Go to step 2 if not converged

ALT-OPT is an approx. of EM: Replaces posterior $p(Z | X, \Theta)$ by a point distribution at its mode
Recap: ALT-OPT and EM

ALT-OPT does the following

1. Initialize $\Theta = \hat{\Theta}$
2. Estimate $Z$ as $\hat{Z} = \arg\max_Z \log p(Z|X, \hat{\Theta})$

Step 2 (arg max) of ALT-OPT could potentially throw away a lot of information about $Z$. EM addresses it using a "soft" version of ALT-OPT

1. Initialize $\Theta = \hat{\Theta}$
2. Compute the posterior distribution of $Z$, i.e., $p(Z|X, \hat{\Theta})$
3. Estimate $\Theta$ by maximizing the expected CLL $\hat{\Theta} = \mathbb{E}_{p(Z|X, \hat{\Theta})}[\log p(X, Z|\Theta)]$
4. Go to step 2 if not converged

ALT-OPT is an approximation of EM: Replaces posterior $p(Z|X, \Theta)$ by a point distribution at its mode.
Recap: ALT-OPT and EM

- ALT-OPT does the following
  1. Initialize $\Theta = \hat{\Theta}$
  2. Estimate $\mathbf{Z}$ as $\hat{\mathbf{Z}} = \arg \max_{\mathbf{Z}} \log p(\mathbf{Z}|\mathbf{X}, \hat{\Theta})$
  3. Estimate $\Theta$ as $\hat{\Theta} = \arg \max_{\Theta} \log p(\mathbf{X}, \hat{\mathbf{Z}}|\Theta)$

- Step 2 (arg max) of ALT-OPT could potentially throw away a lot of information about $\mathbf{Z}$
- EM addresses it using "soft" version of ALT-OPT
  1. Initialize $\Theta = \hat{\Theta}$
  2. Compute the posterior distribution of $\mathbf{Z}$, i.e., $p(\mathbf{Z}|\mathbf{X}, \hat{\Theta})$
  3. Estimate $\Theta$ by maximizing the expected CLL $\hat{\Theta} = \mathbb{E}_{p(\mathbf{Z}|\mathbf{X}, \hat{\Theta})}[\log p(\mathbf{X}, \hat{\mathbf{Z}}|\Theta)]$
  4. Go to step 2 if not converged

ALT-OPT is an approx. of EM: Replaces posterior $p(\mathbf{Z}|\mathbf{X}, \Theta)$ by a point distribution at its mode.
Recap: ALT-OPT and EM

ALT-OPT does the following

1. Initialize $\Theta = \hat{\Theta}$
2. Estimate $Z$ as $\hat{Z} = \arg \max_{Z} \log p(Z|X, \hat{\Theta})$
3. Estimate $\Theta$ as $\hat{\Theta} = \arg \max_{\Theta} \log p(X, \hat{Z}|\Theta)$
4. Go to step 2 if not converged

Step 2 (arg max) of ALT-OPT could potentially throw away a lot of information about $Z$

EM addresses it using "soft" version of ALT-OPT

1. Initialize $\Theta = \hat{\Theta}$
2. Compute the posterior distribution of $Z$, i.e., $p(Z|X, \hat{\Theta})$
3. Estimate $\Theta$ by maximizing the expected CLL $\hat{\Theta} = \mathbb{E}_{p(Z|X, \hat{\Theta})}[\log p(X, Z|\Theta)]$
4. Go to step 2 if not converged

ALT-OPT is an approx. of EM: Replaces posterior $p(Z|X, \Theta)$ by a point distribution at its mode
Recap: ALT-OPT and EM

- ALT-OPT does the following
  1. Initialize $\Theta = \hat{\Theta}$
  2. Estimate $Z$ as $\hat{Z} = \arg \max_Z \log p(Z | X, \hat{\Theta})$
  3. Estimate $\Theta$ as $\hat{\Theta} = \arg \max_{\Theta} \log p(X, \hat{Z} | \Theta)$
  4. Go to step 2 if not converged

- Step 2 (arg max) of ALT-OPT could potentially throw away a lot of information about $Z$

ALT-OPT is an approx. of EM: Replaces posterior $p(Z | X, \Theta)$ by a point distribution at its mode.
Recap: ALT-OPT and EM

- ALT-OPT does the following
  1. Initialize $\Theta = \hat{\Theta}$
  2. Estimate $Z$ as $\hat{Z} = \arg \max_Z \log p(Z|X, \hat{\Theta})$
  3. Estimate $\Theta$ as $\hat{\Theta} = \arg \max_{\Theta} \log p(X, \hat{Z}|\Theta)$
  4. Go to step 2 if not converged

- Step 2 (arg max) of ALT-OPT could potentially throw away a lot of information about $Z$

- EM addresses it using “soft” version of ALT-OPT
Recap: ALT-OPT and EM

ALT-OPT does the following:
1. Initialize Θ = \( \hat{\Theta} \)
2. Estimate \( Z \) as \( \hat{Z} = \arg \max_Z \log p(Z|X, \hat{\Theta}) \)
3. Estimate \( \Theta \) as \( \hat{\Theta} = \arg \max_{\Theta} \log p(X, \hat{Z}|\Theta) \)
4. Go to step 2 if not converged

Step 2 (arg max) of ALT-OPT could potentially throw away a lot of information about \( Z \).

EM addresses it using “soft” version of ALT-OPT:
1. Initialize \( \Theta = \hat{\Theta} \)
Recap: ALT-OPT and EM

- ALT-OPT does the following
  1. Initialize $\Theta = \hat{\Theta}$
  2. Estimate $Z$ as $\hat{Z} = \arg \max_Z \log p(Z|X, \hat{\Theta})$
  3. Estimate $\Theta$ as $\hat{\Theta} = \arg \max_{\Theta} \log p(X, \hat{Z}|\Theta)$
  4. Go to step 2 if not converged

- Step 2 (arg max) of ALT-OPT could potentially throw away a lot of information about $Z$

- EM addresses it using “soft” version of ALT-OPT
  1. Initialize $\Theta = \hat{\Theta}$
  2. Compute the posterior distribution of $Z$, i.e., $p(Z|X, \hat{\Theta})$
Recap: ALT-OPT and EM

ALT-OPT does the following

1. Initialize $\Theta = \hat{\Theta}$
2. Estimate $Z$ as $\hat{Z} = \arg\max_Z \log p(Z|X, \hat{\Theta})$
3. Estimate $\Theta$ as $\hat{\Theta} = \arg\max_{\Theta} \log p(X, \hat{Z}|\Theta)$
4. Go to step 2 if not converged

Step 2 (arg max) of ALT-OPT could potentially throw away a lot of information about $Z$

EM addresses it using “soft” version of ALT-OPT

1. Initialize $\Theta = \hat{\Theta}$
2. Compute the posterior distribution of $Z$, i.e., $p(Z|X, \hat{\Theta})$
3. Estimate $\Theta$ by maximizing the expected CLL $\hat{\Theta} = \mathbb{E}_{p(Z|X, \hat{\Theta})}[\log p(X, Z|\Theta)]$
Recap: ALT-OPT and EM

- ALT-OPT does the following
  1. Initialize $\Theta = \hat{\Theta}$
  2. Estimate $Z$ as $\hat{Z} = \arg \max_Z \log p(Z|X, \hat{\Theta})$
  3. Estimate $\Theta$ as $\hat{\Theta} = \arg \max_\Theta \log p(X, \hat{Z}|\Theta)$
  4. Go to step 2 if not converged

- Step 2 (arg max) of ALT-OPT could potentially throw away a lot of information about $Z$

- EM addresses it using “soft” version of ALT-OPT
  1. Initialize $\Theta = \hat{\Theta}$
  2. Compute the posterior distribution of $Z$, i.e., $p(Z|X, \hat{\Theta})$
  3. Estimate $\Theta$ by maximizing the expected CLL $\hat{\Theta} = \mathbb{E}_{p(Z|X, \hat{\Theta})}[\log p(X, Z|\Theta)]$
  4. Go to step 2 if not converged
Recap: ALT-OPT and EM

- ALT-OPT does the following
  1. Initialize $\Theta = \hat{\Theta}$
  2. Estimate $Z$ as $\hat{Z} = \arg \max_Z \log p(Z|X, \hat{\Theta})$
  3. Estimate $\Theta$ as $\hat{\Theta} = \arg \max_{\Theta} \log p(X, \hat{Z}|\Theta)$
  4. Go to step 2 if not converged

- Step 2 (arg max) of ALT-OPT could potentially throw away a lot of information about $Z$

- EM addresses it using “soft” version of ALT-OPT
  1. Initialize $\Theta = \hat{\Theta}$
  2. Compute the posterior distribution of $Z$, i.e., $p(Z|X, \hat{\Theta})$
  3. Estimate $\Theta$ by maximizing the expected CLL $\hat{\Theta} = \mathbb{E}_{p(Z|X, \hat{\Theta})}[\log p(X, Z|\Theta)]$
  4. Go to step 2 if not converged

- ALT-OPT is an approx. of EM: Replaces posterior $p(Z|X, \Theta)$ by a point distribution at its mode
Brief Detour: Generative Stories
Generative Stories..

Most probabilistic models we’ve seen can be described by an imaginative “generative story”

1. Generate all the global model parameters $\Theta$
   
   $\Theta \sim p(\Theta)$

2. For $n = 1, \ldots, N$
   
   $z_n \sim p(z \mid \Theta)$ ($z_n$ can be an observed label $y_n$ or a latent variable, e.g., cluster id)

   $x_n \sim p(x \mid z = z_n, \Theta)$ ($x_n$ is generated conditioned on $z_n$)

This procedure generates $\{(x_n, z_n)\}_{n=1}^N$ from the joint distribution

$p(x, z \mid \Theta) = p(z \mid \Theta)p(x \mid z, \Theta)$

Note: In this story, we don’t show step 1 if we aren’t using any prior distribution on $\Theta$

Note: If there are no labels or latent variables $z_n$, then we will just have $x_n \sim p(x \mid \Theta)$
Generative Stories..

• Most probabilistic models we’ve seen can be described by an imaginative “generative story”
• In this story, we first generate everything that the data depends on, and then generate the data
Generative Stories..

- Most probabilistic models we’ve seen can be described by an imaginative “generative story”
- In this story, we first generate everything that the data depends on, and then generate the data
- Here is a brief outline of what this story looks like
Most probabilistic models we’ve seen can be described by an imaginative “generative story.”

In this story, we first generate everything that the data depends on, and then generate the data.

Here is a brief outline of what this story looks like:

1. Generate all the global model parameters $\Theta$

   $$\Theta \sim p(\Theta)$$
Most probabilistic models we’ve seen can be described by an imaginative “generative story”

In this story, we first generate everything that the data depends on, and then generate the data

Here is a brief outline of what this story looks like

1. Generate all the global model parameters $\Theta$

   $$\Theta \sim p(\Theta)$$

2. For $n = 1, \ldots, N$

   $$x_n \sim p(x | z_n = z_n, \Theta)$$

   ($z_n$ can be an observed label $y_n$ or a latent variable, e.g., cluster id)

This procedure generates $\{(x_n, z_n)\}_{n=1}^N$ from the joint distribution $p(x, z | \Theta) = p(z | \Theta)p(x | z, \Theta)$

Note: In this story, we don’t show step 1 if we aren’t using any prior distribution on $\Theta$

Note: If there are no labels or latent variables $z_n$, then we will just have $x_n \sim p(x | \Theta)$
Generative Stories..

- Most probabilistic models we’ve seen can be described by an imaginative “generative story”
- In this story, we first generate everything that the data depends on, and then generate the data
- Here is a brief outline of what this story looks like
  1. Generate all the global model parameters $\Theta$
     $$\Theta \sim p(\Theta)$$
  2. For $n = 1, \ldots, N$
     $$z_n \sim p(z|\Theta)$$
     (z_n can be an observed label $y_n$ or a latent variable, e.g., cluster id)
Generative Stories..

- Most probabilistic models we’ve seen can be described by an imaginative “generative story”
- In this story, we first generate everything that the data depends on, and then generate the data
- Here is a brief outline of what this story looks like
  - Generate all the global model parameters $\Theta$
    $$\Theta \sim p(\Theta)$$
  - For $n = 1, \ldots, N$
    $$z_n \sim p(z|\Theta) \quad (z_n \text{ can be an observed label } y_n \text{ or a latent variable, e.g., cluster id})$$
    $$x_n \sim p(x|z = z_n, \Theta) \quad (x_n \text{ is generated conditioned on } z_n)$$
Generative Stories..

Most probabilistic models we’ve seen can be described by an imaginative “generative story”

In this story, we first generate everything that the data depends on, and then generate the data

Here is a brief outline of what this story looks like

1. Generate all the global model parameters $\Theta$

   $\Theta \sim p(\Theta)$

2. For $n = 1, \ldots, N$

   $z_n \sim p(z|\Theta)$ \hspace{1cm} ($z_n$ can be an observed label $y_n$ or a latent variable, e.g., cluster id)

   $x_n \sim p(x|z = z_n, \Theta)$ \hspace{1cm} ($x_n$ is generated conditioned on $z_n$)

This procedure generates $\{(x_n, z_n)\}_{n=1}^{N}$ from the joint distribution $p(x, z|\Theta) = p(z|\Theta)p(x|z, \Theta)$
Generative Stories..

- Most probabilistic models we’ve seen can be described by an imaginative “generative story”
- In this story, we first generate everything that the data depends on, and then generate the data
- Here is a brief outline of what this story looks like
  1. Generate all the global model parameters $\Theta$
     $$\Theta \sim p(\Theta)$$
  2. For $n = 1, \ldots, N$
     $$z_n \sim p(z|\Theta) \quad (z_n \text{ can be an observed label } y_n \text{ or a latent variable, e.g., cluster id})$$
     $$x_n \sim p(x|z = z_n, \Theta) \quad (x_n \text{ is generated conditioned on } z_n)$$
  3. This procedure generates $\{(x_n, z_n)\}_{n=1}^{N}$ from the joint distribution $p(x, z|\Theta) = p(z|\Theta)p(x|z, \Theta)$
  4. Note: In this story, we don’t show step 1 if we aren’t using any prior distribution on $\Theta$
Most probabilistic models we’ve seen can be described by an imaginative “generative story”

In this story, we first generate everything that the data depends on, and then generate the data

Here is a brief outline of what this story looks like

1. Generate all the global model parameters $\Theta$
   $$\Theta \sim p(\Theta)$$

2. For $n = 1, \ldots, N$
   $$z_n \sim p(z|\Theta)$$ (\textit{z}_n \text{ can be an observed label } y_n \text{ or a latent variable, e.g., cluster id})
   $$x_n \sim p(x|z = z_n, \Theta)$$ ($x_n$ is generated conditioned on $z_n$)

This procedure generates $\{(x_n, z_n)\}_{n=1}^{N}$ from the joint distribution $p(x, z|\Theta) = p(z|\Theta)p(x|z, \Theta)$

Note: In this story, we don’t show step 1 if we aren’t using any prior distribution on $\Theta$

Note: If there are no labels or latent variables $z_n$, then we will just have $x_n \sim p(x|\Theta)$
Generative Story: Some Common Examples

- Can have it if at least some part of the data is generated using a probability distribution
  (Note: Generation of global parameters \( \Theta \) not shown below)

**Generative Classification (Gaussian Class-Conditionals)**
- For \( n = 1, \ldots, N \)
  - Generate \( y_n \) as \( y_n \sim \text{multinoulli}(\pi_1, \ldots, \pi_K) \)
  - Generate \( x_n \) as \( x_n \sim \mathcal{N}(\mu_{y_n}, \Sigma_{y_n}) \)

**Gaussian Mixture Model**
- For \( n = 1, \ldots, N \)
  - Generate \( z_n \) as \( z_n \sim \text{multinoulli}(\pi_1, \ldots, \pi_K) \)
  - Generate \( x_n \) as \( x_n \sim \mathcal{N}(\mu_{z_n}, \Sigma_{z_n}) \)

**Probabilistic Dimensionality Reduction (Probabilistic PCA)**
(assuming data and latent variables to be Gaussians)
- For \( n = 1, \ldots, N \)
  - Generate \( z_n \) as \( z_n \sim \mathcal{N}(0, I_K) \)
  - Generate \( x_n \) as \( x_n \sim \mathcal{N}(Wz_n, \sigma^2 I_D) \)

**Discriminative Models for Regression/Classification**
- For \( n = 1, \ldots, N \)
  - Generate \( y_n \) as
    \[
    y_n \sim \mathcal{N}(w^Tx_n, \sigma^2) \quad \text{and} \quad y_n \sim \text{Bernoulli}(\sigma(w^Tx_n))
    \]

- The model need not have latent variables (e.g. generative classification, discriminative models)
Latent Variable Models for Dimensionality Reduction
Consider a set of observations $x_1, \ldots, x_N$, with $x_n \in \mathbb{R}^D$.
A Simple Model for Data Compression/Dimensionality-Reduction

- Consider a set of observations $x_1, \ldots, x_N$, with $x_n \in \mathbb{R}^D$
- Let’s approximate each $x_n$ by a linear combination of $K$ vectors $w_1, w_2, \ldots, w_K$ ($K \ll D$)

$$x_n \approx \sum_{k=1}^{K} z_{nk} w_k$$
Consider a set of observations $x_1, \ldots, x_N$, with $x_n \in \mathbb{R}^D$

Let’s approximate each $x_n$ by a linear combination of $K$ vectors $w_1, w_2, \ldots, w_K$ ($K \ll D$)

$$x_n \approx \sum_{k=1}^{K} z_{nk} w_k \quad \text{or} \quad x_n \approx Wz_n$$
A Simple Model for Data Compression/Dimensionality-Reduction

- Consider a set of observations $x_1, \ldots, x_N$, with $x_n \in \mathbb{R}^D$
- Let’s approximate each $x_n$ by a linear combination of $K$ vectors $w_1, w_2, \ldots, w_K$ ($K \ll D$)

$$x_n \approx \sum_{k=1}^{K} z_{nk} w_k \quad \text{or} \quad x_n \approx Wz_n$$

where $W = [w_1 \ldots w_K]$ is $D \times K$, each $w_k \in \mathbb{R}^D$, and $z_n = [z_{n1} \ldots z_{nK}] \in \mathbb{R}^K$
A Simple Model for Data Compression/Dimensionality-Reduction

- Consider a set of observations \( x_1, \ldots, x_N \), with \( x_n \in \mathbb{R}^D \)

- Let’s approximate each \( x_n \) by a linear combination of \( K \) vectors \( w_1, w_2, \ldots, w_K \) (\( K \ll D \)):
  \[
  x_n \approx \sum_{k=1}^{K} z_{nk} w_k \quad \text{or} \quad x_n \approx Wz_n
  \]

where \( W = [w_1 \ldots w_K] \) is \( D \times K \), each \( w_k \in \mathbb{R}^D \), and \( z_n = [z_{n1} \ldots z_{nK}] \in \mathbb{R}^K \)
A Simple Model for Data Compression/Dimensionality-Reduction

- Consider a set of observations \( x_1, \ldots, x_N \), with \( x_n \in \mathbb{R}^D \)
- Let’s approximate each \( x_n \) by a linear combination of \( K \) vectors \( w_1, w_2, \ldots, w_K \) \( (K \ll D) \)

\[
x_n \approx \sum_{k=1}^{K} z_{nk} w_k \quad \text{or} \quad x_n \approx W z_n
\]

where \( W = [w_1 \ldots w_K] \) is \( D \times K \), each \( w_k \in \mathbb{R}^D \), and \( z_n = [z_{n1} \ldots z_{nK}] \in \mathbb{R}^K \)
A Simple Model for Data Compression/Dimensionality-Reduction

- Consider a set of observations $x_1, \ldots, x_N$, with $x_n \in \mathbb{R}^D$
- Let’s approximate each $x_n$ by a linear combination of $K$ vectors $w_1, w_2, \ldots, w_K$ ($K \ll D$)

$$x_n \approx \sum_{k=1}^{K} z_{nk} w_k \quad \text{or} \quad x_n \approx Wz_n$$

where $W = [w_1 \ldots w_K]$ is $D \times K$, each $w_k \in \mathbb{R}^D$, and $z_n = [z_{n1} \ldots z_{nK}] \in \mathbb{R}^K$

- $z_{nk}$ tell us much of “component” $w_k$ is present in the observation $x_n$
Consider a set of observations $x_1, \ldots, x_N$, with $x_n \in \mathbb{R}^D$.

Let's approximate each $x_n$ by a linear combination of $K$ vectors $w_1, w_2, \ldots, w_K$ ($K \ll D$):

$$x_n \approx \sum_{k=1}^{K} z_{nk} w_k \quad \text{or} \quad x_n \approx Wz_n$$

where $W = [w_1 \ldots w_K]$ is $D \times K$, each $w_k \in \mathbb{R}^D$, and $z_n = [z_{n1} \ldots z_{nK}] \in \mathbb{R}^K$.

- $z_{nk}$ tell us much of “component” $w_k$ is present in the observation $x_n$.
- Can think of $z_n \in \mathbb{R}^K$ as a “compressed” latent representation of $x_n \in \mathbb{R}^D$. 

Intro to Machine Learning (CS771A)

Latent Variable Models for Dimensionality Reduction
A Simple Model for Data Compression/Dimensionality-Reduction

- Consider a set of observations $x_1, \ldots, x_N$, with $x_n \in \mathbb{R}^D$

- Let’s approximate each $x_n$ by a **linear combination** of $K$ vectors $w_1, w_2, \ldots, w_K$ ($K \ll D$)

\[
x_n \approx \sum_{k=1}^{K} z_{nk} w_k \quad \text{or} \quad x_n \approx Wz_n
\]

where $W = [w_1 \ldots w_K]$ is $D \times K$, each $w_k \in \mathbb{R}^D$, and $z_n = [z_{n1} \ldots z_{nK}] \in \mathbb{R}^K$

- $z_{nk}$ tell us much of “component” $w_k$ is present in the observation $x_n$

- Can think of $z_n \in \mathbb{R}^K$ as a “compressed” latent representation of $x_n \in \mathbb{R}^D$

- A good compression $z_n$ will be one for which $x_n$ is as close as possible to $Wz_n$
Dimensionality Reduction: The Probabilistic/Generative View

- In the linear model, we represented $x_n$ approximately as $x_n \approx Wz_n$
In the linear model, we represented $x_n$ approximately as $x_n \approx Wz_n$

The probabilistic view: Model $x_n$ by a $D$-dim Gaussian with mean vector $Wz_n$

Equivalently:

$$x_n = Wz_n + \epsilon_n$$

where $\epsilon_n \sim N(0, \sigma^2 I_D)$

Let's assume a prior $p(z_n) = N(0, I_K)$ on the latent variable $z_n$

A low-dim latent variable $z_n$ transformed to "generate" a high-dim observation $x_n$

This is a "reverse" way of thinking: A generative model for dimensionality reduction

This model is popularly known as Probabilistic Principal Component Analysis (PPCA)

The standard non-probabilistic PCA is a special case (probabilistic version has several advantages)
In the linear model, we represented $x_n$ approximately as $x_n \approx Wz_n$

The probabilistic view: Model $x_n$ by a $D$-dim Gaussian with mean vector $Wz_n$

$$p(x_n|z_n, W, \sigma^2) = \mathcal{N}(Wz_n, \sigma^2 I_D)$$

Equivalently: $x_n = Wz_n + \epsilon_n$ where $\epsilon_n \sim \mathcal{N}(0, \sigma^2 I_D)$
In the linear model, we represented $x_n$ approximately as $x_n \approx Wz_n$

The probabilistic view: Model $x_n$ by a $D$-dim Gaussian with mean vector $Wz_n$

$$p(x_n|z_n, W, \sigma^2) = \mathcal{N}(Wz_n, \sigma^2 I_D)$$

Equivalently: $x_n = Wz_n + \epsilon_n$ where $\epsilon_n \sim \mathcal{N}(0, \sigma^2 I_D)$

Let’s assume a prior $p(z_n) = \mathcal{N}(0, I_K)$ on the latent variable $z_n$
In the linear model, we represented $x_n$ approximately as $x_n \approx Wz_n$.

The probabilistic view: Model $x_n$ by a $D$-dim Gaussian with mean vector $Wz_n$.

$$p(x_n|z_n, W, \sigma^2) = \mathcal{N}(Wz_n, \sigma^2 I_D)$$

Equivalently:

$$x_n = Wz_n + \epsilon_n \quad \text{where} \quad \epsilon_n \sim \mathcal{N}(0, \sigma^2 I_D)$$

Let’s assume a prior $p(z_n) = \mathcal{N}(0, I_K)$ on the latent variable $z_n$.

A low-dim latent variable $z_n$ transformed to “generate” a high-dim observation $x_n$. 

This model is popularly known as Probabilistic Principal Component Analysis (PPCA).

The standard non-probabilistic PCA is a special case (probabilistic version has several advantages).
In the linear model, we represented $x_n$ approximately as $x_n \approx Wz_n$

The probabilistic view: Model $x_n$ by a $D$-dim Gaussian with mean vector $Wz_n$

\[
p(x_n|z_n, W, \sigma^2) = \mathcal{N}(Wz_n, \sigma^2 I_D)
\]

Equivalently: $x_n = Wz_n + \epsilon_n$ where $\epsilon_n \sim \mathcal{N}(0, \sigma^2 I_D)$

Let’s assume a prior $p(z_n) = \mathcal{N}(0, I_K)$ on the latent variable $z_n$

A low-dim latent variable $z_n$ transformed to “generate” a high-dim observation $x_n$

This is a “reverse” way of thinking: A generative model for dimensionality reduction
In the linear model, we represented $x_n$ approximately as $x_n \approx Wz_n$.

The probabilistic view: Model $x_n$ by a $D$-dim Gaussian with mean vector $Wz_n$.

$$p(x_n|z_n, W, \sigma^2) = \mathcal{N}(Wz_n, \sigma^2 I_D)$$

Equivalently: $x_n = Wz_n + \epsilon_n$ where $\epsilon_n \sim \mathcal{N}(0, \sigma^2 I_D)$

Let’s assume a prior $p(z_n) = \mathcal{N}(0, I_K)$ on the latent variable $z_n$.

A low-dim latent variable $z_n$ transformed to “generate” a high-dim observation $x_n$.

This is a “reverse” way of thinking: A generative model for dimensionality reduction.

This model is popularly known as Probabilistic Principal Component Analysis (PPCA).
Dimensionality Reduction: The Probabilistic/Generative View

- In the linear model, we represented $x_n$ approximately as $x_n \approx Wz_n$
- The probabilistic view: Model $x_n$ by a $D$-dim Gaussian with mean vector $Wz_n$

$$p(x_n|z_n, W, \sigma^2) = \mathcal{N}(Wz_n, \sigma^2 I_D)$$

Equivalently: $x_n = Wz_n + \epsilon_n$ where $\epsilon_n \sim \mathcal{N}(0, \sigma^2 I_D)$

- Let’s assume a prior $p(z_n) = \mathcal{N}(0, I_K)$ on the latent variable $z_n$
- A low-dim latent variable $z_n$ transformed to “generate” a high-dim observation $x_n$
- This is a “reverse” way of thinking: A generative model for dimensionality reduction

- This model is popularly known as Probabilistic Principal Component Analysis (PPCA)
  - The standard non-probabilistic PCA is a special case (probabilistic version has several advantages)
Some More Motivation for PPCA..

- Suppose we’re modeling $D$-dim data using a (say zero mean) Gaussian

$$p(x) = \mathcal{N}(0, \Sigma)$$

where $\Sigma$ is a $D \times D$ p.s.d. cov. matrix, $O(D^2)$ parameters needed
Some More Motivation for PPCA..

Suppose we’re modeling $D$-dim data using a (say zero mean) Gaussian

$$p(x) = \mathcal{N}(0, \Sigma)$$

where $\Sigma$ is a $D \times D$ p.s.d. cov. matrix, $O(D^2)$ parameters needed

Consider modeling the same data using PPCA:

$$p(x|z) = \mathcal{N}(Wz, \sigma^2 I_D), \quad p(z) = \mathcal{N}(0, I_K)$$

Cov. matrix is close to low-rank as $\sigma^2 \to 0$. Only $(DK + 1)$ parameters needed (nice when $D \gg N$)

PPCA = Low-rank Gaussian. Fewer parameters to learn; less chance of overfitting
Some More Motivation for PPCA.

- Suppose we’re modeling $D$-dim data using a (say zero mean) Gaussian
  \[ p(x) = N(0, \Sigma) \]
  where $\Sigma$ is a $D \times D$ p.s.d. cov. matrix, $O(D^2)$ parameters needed
- Consider modeling the same data using PPCA: $p(x|z) = N(Wz, \sigma^2 I_D)$, $p(z) = N(0, I_K)$
- For this Gaussian PPCA, the marginal distribution $p(x) = \int p(x, z)dz$ is
  \[ p(x|W, \sigma^2) = N(0, WW^T + \sigma^2 I_D) \]
  (using Gaussian marginal results)
Some More Motivation for PPCA..

- Suppose we’re modeling $D$-dim data using a (say zero mean) Gaussian
  \[ p(x) = \mathcal{N}(0, \Sigma) \]
  where $\Sigma$ is a $D \times D$ p.s.d. cov. matrix, $\mathcal{O}(D^2)$ parameters needed
- Consider modeling the same data using PPCA: $p(x|z) = \mathcal{N}(Wz, \sigma^2 I_D)$, $p(z) = \mathcal{N}(0, I_K)$
- For this Gaussian PPCA, the marginal distribution $p(x) = \int p(x, z)dz$ is
  \[ p(x|W, \sigma^2) = \mathcal{N}(0, WW^\top + \sigma^2 I_D) \] (using Gaussian marginal results)
- Cov. matrix is close to low-rank as $\sigma^2 \to 0$. Only $(DK + 1)$ parameters needed (nice when $D \gg N$)
Some More Motivation for PPCA...

- Suppose we’re modeling $D$-dim data using a (say zero mean) Gaussian
  \[ p(x) = \mathcal{N}(0, \Sigma) \]
  where $\Sigma$ is a $D \times D$ p.s.d. cov. matrix, $\mathcal{O}(D^2)$ parameters needed
- Consider modeling the same data using PPCA: $p(x|z) = \mathcal{N}(Wz, \sigma^2 I_D)$, $p(z) = \mathcal{N}(0, I_K)$
- For this Gaussian PPCA, the marginal distribution $p(x) = \int p(x, z) dz$ is
  \[ p(x|W, \sigma^2) = \mathcal{N}(0, WW^\top + \sigma^2 I_D) \]
  (using Gaussian marginal results)

  Cov. matrix is close to low-rank as $\sigma^2 \to 0$. Only $(DK + 1)$ parameters needed (nice when $D \gg N$)
  - PPCA = Low-rank Gaussian. Fewer parameters to learn; less chance of overfitting
One benefit: Once the model parameters are learned, we can even generate new data.
Benefits of Generative Models for Dimensionality Reduction

- One benefit: Once the model parameters are learned, we can even generate new data, e.g.,
  - Generate a random $z$ using the distribution $\mathcal{N}(0, I_K)$
Benefits of Generative Models for Dimensionality Reduction

- One benefit: Once the model parameters are learned, we can even generate new data, e.g.,
  - Generate a random $z$ using the distribution $\mathcal{N}(0, I_K)$
  - Generate $x$ conditioned on this $z$ from $\mathcal{N}(Wz, \sigma^2 I_D)$

Note: The above random samples are generated using a slightly more sophisticated latent variable model (VAE with ALI-BiGAN inference), not the simple PPCA (but it is similar in spirit to PPCA).

Many other benefits. For example, can do dim-red, even if $x$ has part of it as missing.
Benefits of Generative Models for Dimensionality Reduction

- One benefit: Once the model parameters are learned, we can even generate new data, e.g.,
  - Generate a random $z$ using the distribution $\mathcal{N}(0, I_K)$
  - Generate $x$ conditioned on this $z$ from $\mathcal{N}(Wz, \sigma^2 I_D)$

Note: The above random samples are generated using a slightly more sophisticated latent variable model (VAE with ALI-BiGAN inference), not the simple PPCA (but it is similar in spirit to PPCA).

Many other benefits. For example, can do dim-red, even if $x$ has part of it as missing.
One benefit: Once the model parameters are learned, we can even **generate new data**, e.g.,

- Generate a random \( z \) using the distribution \( \mathcal{N}(0, \mathbf{I}_K) \)
- Generate \( x \) conditioned on this \( z \) from \( \mathcal{N}(\mathbf{W}z, \sigma^2\mathbf{I}_D) \)

Note: The above random samples are generated using a slightly more sophisticated latent variable model (VAE with ALI-BiGAN inference), not the simple PPCA (but it is similar in spirit to PPCA).
Benefits of Generative Models for Dimensionality Reduction

- One benefit: Once the model parameters are learned, we can even generate new data, e.g.,
  - Generate a random $z$ using the distribution $\mathcal{N}(0, \mathbf{I}_K)$
  - Generate $x$ conditioned on this $z$ from $\mathcal{N}(\mathbf{W}z, \sigma^2 \mathbf{I}_D)$

Note: The above random samples are generated using a slightly more sophisticated latent variable model (VAE with ALI-BiGAN inference), not the simple PPCA (but it is similar in spirit to PPCA).

- Many other benefits. For example, can do dim-red, even if $x_n$ has part of it as missing.
Learning the PPCA Model

- Since we are doing dim-red, the goal is to “recover” $z_n$ (and $W, \sigma^2$) given $x_n$, $\forall n$.
Learning the PPCA Model

- Since we are doing dim-red, the goal is to “recover” $z_n$ (and $W, \sigma^2$) given $x_n, \forall n$
- The likelihood $p(x_n|z_n) = \mathcal{N}(Wz_n, \sigma^2 I_D)$ is Gaussian
Learning the PPCA Model

- Since we are doing dim-red, the goal is to "recover" \( z_n \) (and \( W, \sigma^2 \)) given \( x_n, \forall n \)
- The likelihood \( p(x_n|z_n) = N(Wz_n, \sigma^2 I_D) \) is Gaussian. The loss function = NLL will be
Learning the PPCA Model

- Since we are doing dim-red, the goal is to "recover" $z_n$ (and $W, \sigma^2$) given $x_n, \forall n$
- The likelihood $p(x_n|z_n) = \mathcal{N}(Wz_n, \sigma^2 I_D)$ is Gaussian. The loss function = NLL will be

$$
\mathcal{L}(Z, W, \sigma^2) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} \|x_n - Wz_n\|^2 + \frac{ND}{2} \log(2\pi\sigma^2) \quad \text{(Exercise: Verify)}
$$

Nice! So this loss is simply the reconstruction error. We can minimize it w.r.t. $Z, W, \sigma^2$
For simplicity, let’s treat $\sigma^2$ as a constant. Then the loss function will be

$$
\mathcal{L}(Z, W) = ||X - ZW^\top||_F^2
$$

Dimensionality reduction then simply boils down to solving the following problem

$$
\hat{Z}, \hat{W} = \arg \min_{Z, W} ||X - ZW^\top||_F^2 \quad \text{(Alert: This is NOT doing MLE but arg max)}
$$

Can solve it using ALT-OPT to solve it. Another (better) way will be to do a proper MLE using EM
Learning the PPCA Model

- Since we are doing dim-red, the goal is to "recover" $z_n$ (and $W, \sigma^2$) given $x_n, \forall n$
- The likelihood $p(x_n|z_n) = \mathcal{N}(Wz_n, \sigma^2 I_D)$ is Gaussian. The loss function $= \text{NLL}$ will be

$$\mathcal{L}(Z, W, \sigma^2) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} ||x_n - Wz_n||^2 + \frac{ND}{2} \log(2\pi\sigma^2) \quad \text{(Exercise: Verify)}$$

$$= \frac{1}{2\sigma^2} ||X - ZW^T||_F^2 + \frac{ND}{2} \log(2\pi\sigma^2) \quad (X: N \times D, Z: N \times K, W: D \times K)$$
Learning the PPCA Model

- Since we are doing dim-red, the goal is to "recover" $z_n$ (and $W, \sigma^2$) given $x_n, \forall n$
- The likelihood $p(x_n|z_n) = \mathcal{N}(Wz_n, \sigma^2 I_D)$ is Gaussian. The loss function = NLL will be

$$\mathcal{L}(Z, W, \sigma^2) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} ||x_n - Wz_n||^2 + \frac{ND}{2} \log(2\pi\sigma^2) \quad \text{(Exercise: Verify)}$$

$$= \frac{1}{2\sigma^2} ||X - ZW^T||_F^2 + \frac{ND}{2} \log(2\pi\sigma^2) \quad \text{ (X : N x D, Z : N x K, W : D x K)}$$

- Nice! So this loss is simply the reconstruction error. We can minimize it w.r.t. $Z, W, \sigma^2$
Learning the PPCA Model

- Since we are doing dim-red, the goal is to “recover” $z_n$ (and $W, \sigma^2$) given $x_n, \forall n$
- The likelihood $p(x_n|z_n) = \mathcal{N}(Wz_n, \sigma^2I_D)$ is Gaussian. The loss function $= \text{NLL}$ will be

$$\mathcal{L}(Z, W, \sigma^2) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} \|x_n - Wz_n\|^2 + \frac{ND}{2} \log(2\pi\sigma^2) \quad \text{(Exercise: Verify)}$$

$$= \frac{1}{2\sigma^2} \|X - ZW^\top\|_F^2 + \frac{ND}{2} \log(2\pi\sigma^2) \quad \text{($X: N \times D, Z: N \times K, W: D \times K$)}$$

- Nice! So this loss is simply the reconstruction error. We can minimize it w.r.t. $Z, W, \sigma^2$
- For simplicity, let’s treat $\sigma^2$ as a constant. Then the loss function will be

$$\mathcal{L}(Z, W) = \|X - ZW^\top\|_F^2$$
Learning the PPCA Model

- Since we are doing dim-red, the goal is to “recover” $z_n$ (and $W, \sigma^2$) given $x_n, \forall n$
- The likelihood $p(x_n|z_n) = \mathcal{N}(Wz_n, \sigma^2 I_D)$ is Gaussian. The loss function $= \text{NLL}$ will be

$$L(Z, W, \sigma^2) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} ||x_n - Wz_n||^2 + \frac{ND}{2} \log(2\pi\sigma^2)$$  

(Exercise: Verify)

$$= \frac{1}{2\sigma^2} ||X - ZW^\top||_F^2 + \frac{ND}{2} \log(2\pi\sigma^2) \quad (X : N \times D, Z : N \times K, W : D \times K)$$

- Nice! So this loss is simply the reconstruction error. We can minimize it w.r.t. $Z, W, \sigma^2$
- For simplicity, let’s treat $\sigma^2$ as a constant. Then the loss function will be

$$L(Z, W) = ||X - ZW^\top||_F^2$$

- Dimensionality reduction then simply boils down to solving the following problem

$$\{\hat{Z}, \hat{W}\} = \arg\min_{Z, W} ||X - ZW^\top||_F^2$$  

(Alert: This is NOT doing MLE but $\arg\max \sum_{n=1}^{N} \log p(x_n|z_n)$)

Intro to Machine Learning (CS771A)  
Latent Variable Models for Dimensionality Reduction  
12
Learning the PPCA Model

- Since we are doing dim-red, the goal is to “recover” $z_n$ (and $W, \sigma^2$) given $x_n, \forall n$
- The likelihood $p(x_n|z_n) = \mathcal{N}(Wz_n, \sigma^2 I_D)$ is Gaussian. The loss function = NLL will be

$$L(Z, W, \sigma^2) = \frac{1}{2\sigma^2} \sum_{n=1}^{N} ||x_n - Wz_n||^2 + \frac{ND}{2} \log(2\pi\sigma^2) \quad \text{(Exercise: Verify)}$$

$$= \frac{1}{2\sigma^2} ||X - ZW^\top||_F^2 + \frac{ND}{2} \log(2\pi\sigma^2) \quad \text{(X : N \times D, Z : N \times K, W : D \times K)}$$

- Nice! So this loss is simply the reconstruction error. We can minimize it w.r.t. $Z, W, \sigma^2$
- For simplicity, let’s treat $\sigma^2$ as a constant. Then the loss function will be

$$L(Z, W) = ||X - ZW^\top||_F^2$$

- Dimensionality reduction then simply boils down to solving the following problem

$$\{\hat{Z}, \hat{W}\} = \arg\min_{Z,W} ||X - ZW^\top||_F^2 \quad \text{(Alert: This is NOT doing MLE but arg max \sum_{n=1}^{N} \log p(x_n|z_n))}$$

- Can solve it using ALT-OPT to solve it. Another (better) way will be to do a proper MLE using EM
We saw that the PPCA problem reduced to

\[ \{ \hat{Z}, \hat{W} \} = \arg \min_{Z, W} \| X - ZW^T \|_F^2 \]

The ALT-OPT algorithm for PPCA will alternate between the following two steps:

1. Initialize \( \hat{Z} \).
2. Solve \( \hat{W} = \arg \min_W \| X - \hat{Z}W^T \|_F^2 \).
3. Solve \( \hat{Z} = \arg \min_Z \| X - Z\hat{W}^T \|_F^2 \).
4. Go to step 2 if not yet converged.

Step 2 is just like multi-output regression with \( \hat{Z} \) as feature matrix and \( X \) as label matrix.

Step is also like multi-output regression.

Note that the problem is essentially a matrix factorization of \( X \).
We saw that the PPCA problem reduced to

\[
\{ \hat{Z}, \hat{W} \} = \arg \min_{Z, W} \| X - ZW^T \|_F^2
\]

The ALT-OPT algorithm for PPCA will alternate between the following two steps:

1. Initialize \( \hat{Z} \)
2. Solve \( \hat{W} = \arg \min_W \| X - \hat{Z}W^T \|_F^2 \)
3. Solve \( \hat{Z} = \arg \min_Z \| X - Z\hat{W}^T \|_F^2 \)
4. Go to step 2 if not yet converged

Step 2 is just like multi-output regression with \( \hat{Z} \) as feature matrix and \( X \) as label matrix.
Learning PPCA via ALT-OPT

- We saw that the PPCA problem reduced to

$$\{\hat{Z}, \hat{W}\} = \arg\min_{Z,W} \|X - ZW^T\|^2_F$$

- The ALT-OPT algorithm for PPCA will alternate between the following two steps

1. Initialize $Z = \hat{Z}$
Learning PPCA via ALT-OPT

We saw that the PPCA problem reduced to

$$\{\hat{Z}, \hat{W}\} = \text{arg min}_{Z, W} ||X - ZW^T||_F^2$$

The ALT-OPT algorithm for PPCA will alternate between the following two steps:

1. Initialize $Z = \hat{Z}$
2. Solve $\hat{W} = \text{arg min}_W ||X - \hat{Z}W^T||_F^2$

Step 2 is just like multi-output regression with $\hat{Z}$ as feature matrix and $X$ as label matrix.
Learning PPCA via ALT-OPT

- We saw that the PPCA problem reduced to

\[ \{\hat{Z}, \hat{W}\} = \arg \min_{Z, W} \| X - ZW^T \|_F^2 \]

- The ALT-OPT algorithm for PPCA will alternate between the following two steps
  1. Initialize \( Z = \hat{Z} \)
  2. Solve \( \hat{W} = \arg \min_W \| X - \hat{Z}W^T \|_F^2 \)
  3. Solve \( \hat{Z} = \arg \min_Z \| X - Z\hat{W}^T \|_F^2 \)
Learning PPCA via ALT-OPT

- We saw that the PPCA problem reduced to

\[
\{\hat{Z}, \hat{W}\} = \arg \min_{Z,W} \|X - ZW^T\|_F^2
\]

- The ALT-OPT algorithm for PPCA will alternate between the following two steps

1. Initialize \( Z = \hat{Z} \)
2. Solve \( \hat{W} = \arg \min_W \|X - \hat{Z}W^T\|_F^2 \)
3. Solve \( \hat{Z} = \arg \min_Z \|X - Z\hat{W}^T\|_F^2 \)
4. Go to step 2 if not yet converged

Note that the problem is essentially a matrix factorization of \( X \)
Learning PPCA via ALT-OPT

- We saw that the PPCA problem reduced to

\[
\{ \hat{Z}, \hat{W} \} = \arg \min_{Z,W} \| X - ZW^T \|_F^2
\]

- The ALT-OPT algorithm for PPCA will alternate between the following two steps

  1. Initialize \( Z = \hat{Z} \)
  2. Solve \( \hat{W} = \arg \min_W \| X - \hat{ZW}^T \|_F^2 \)
  3. Solve \( \hat{Z} = \arg \min_Z \| X - Z\hat{W}^T \|_F^2 \)
  4. Go to step 2 if not yet converged

- Step 2 is just like multi-output regression with \( \hat{Z} \) as feature matrix and \( X \) as label matrix.
We saw that the PPCA problem reduced to
\[
\{\hat{Z}, \hat{W}\} = \arg\min_{Z,W} ||X - ZW^T||_F^2
\]

The ALT-OPT algorithm for PPCA will alternate between the following two steps
1. Initialize $Z = \hat{Z}$
2. Solve $\hat{W} = \arg\min_W ||X - \hat{Z}W^T||_F^2$
3. Solve $\hat{Z} = \arg\min_Z ||X - Z\hat{W}^T||_F^2$
4. Go to step 2 if not yet converged

Step 2 is just like multi-output regression with $\hat{Z}$ as feature matrix and $X$ as label matrix

Step is also like multi-output regression
Learning PPCA via ALT-OPT

- We saw that the PPCA problem reduced to

\[ \{ \hat{Z}, \hat{W} \} = \arg \min_{Z,W} ||X - ZW^T||^2_F \]

- The ALT-OPT algorithm for PPCA will alternate between the following two steps
  1. Initialize \( Z = \hat{Z} \)
  2. Solve \( \hat{W} = \arg \min_W ||X - \hat{Z}W^T||_F^2 \)
  3. Solve \( \hat{Z} = \arg \min_Z ||X - Z\hat{W}^T||_F^2 \)
  4. Go to step 2 if not yet converged

- Step 2 is just like multi-output regression with \( \hat{Z} \) as feature matrix and \( X \) as label matrix

- Step is also like multi-output regression

- Note that the problem is essentially a matrix factorization of \( X \)
MLE for PPCA (or why it is hard..)

To do MLE, we need to maximize \( \log p(X|W, \sigma^2) = \sum_{n=1}^{N} \log p(x_n|W, \sigma^2) \) with \( z_n \) integrated out.

\[ \log p(X|\Theta) = -N/2 \left( D \log 2\pi + \log |C| + \text{trace}(C^{-1}S) \right) \]

where \( S \) is the data covariance matrix, \( C^{-1} = \sigma^{-1}I - \sigma^{-1}WMW^\top \) and \( M = W^\top W + \sigma^{-2}I \).

The MLE solution is given by (don't worry about the proof)

\[ W_{ML} = U_K \left( L_K - \sigma_{ML}^2 I \right)^{1/2} R_{\sigma_{ML}^2} \]

\( U_K \) is \( D \times K \) matrix of top \( K \) eigvecs of \( S \), \( L_K \): \( K \times K \) diagonal matrix of top \( K \) eigvals \( \lambda_1, ..., \lambda_K \), \( R \) is a \( K \times K \) arbitrary rotation matrix (equivalent to PCA for \( R = I \) and \( \sigma^2 \rightarrow 0 \)).

Need to do eigen-decomposition of \( D \times D \) data covariance matrix. EXPENSIVE!!!
MLE for PPCA (or why it is hard..)

- To do MLE, we need to maximize \( \log p(X|W, \sigma^2) = \sum_{n=1}^{N} \log p(x_n|W, \sigma^2) \) with \( z_n \) integrated out.
- MLE on the objective \( p(x_n|W, \sigma^2) \) can be done but turns out to be a bit expensive. In particular:

\[
\log p(X|\Theta) = -\frac{N}{2} \left( D \log 2\pi + \log |C| + \text{trace}(C^{-1}S) \right)
\]

where \( S \) is the data covariance matrix, \( C^{-1} = \sigma^{-1}I - \sigma^{-1}WM^{-1}W^\top \) and \( M = W^\top W + \sigma^2 I \)

† Probabilistic Principal Component Analysis (Tipping and Bishop, 1999)
MLE for PPCA (or why it is hard..)

- To do MLE, we need to maximize $\log p(X|W, \sigma^2) = \sum_{n=1}^{N} \log p(x_n|W, \sigma^2)$ with $z_n$ integrated out.

- MLE on the objective $p(x_n|W, \sigma^2)$ can be done but turns out to be a bit expensive. In particular:

$$\log p(X|\Theta) = -\frac{N}{2} (D \log 2\pi + \log |C| + \text{trace}(C^{-1}S))$$

where $S$ is the data covariance matrix, $C^{-1} = \sigma^{-1}I - \sigma^{-1}WM^{-1}W^T$ and $M = W^TW + \sigma^2I$.

- The MLE solution is given by (don’t worry about the proof)$^\dagger$

$$W_{ML} = U_K(L_K - \sigma^2_{ML}I)^{1/2}R$$

$$\sigma^2_{ML} = \frac{1}{D-K} \sum_{k=K+1}^{D} \lambda_k$$  (noise variance = mean of “discarded” eigenvalues)

---

$^\dagger$ Probabilistic Principal Component Analysis (Tipping and Bishop, 1999)
MLE for PPCA (or why it is hard..)

- To do MLE, we need to maximize \( \log p(X|\mathbf{W}, \sigma^2) = \sum_{n=1}^{N} \log p(x_n|\mathbf{W}, \sigma^2) \) with \( z_n \) integrated out.

- MLE on the objective \( p(x_n|\mathbf{W}, \sigma^2) \) can be done but turns out to be a bit expensive. In particular:

\[
\log p(X|\Theta) = -\frac{N}{2} (D \log 2\pi + \log |C| + \text{trace}(C^{-1}S))
\]

where \( S \) is the data covariance matrix, \( C^{-1} = \sigma^{-1}I - \sigma^{-1}\mathbf{W}\mathbf{M}^{-1}\mathbf{W}^\top \) and \( \mathbf{M} = \mathbf{W}^\top \mathbf{W} + \sigma^2I \)

- The MLE solution is given by (don’t worry about the proof)†

\[
\mathbf{W}_{ML} = U_K(L_K - \sigma_{ML}^2I)^{1/2}R
\]

\[
\sigma_{ML}^2 = \frac{1}{D - K} \sum_{k=K+1}^{D} \lambda_k \quad \text{(noise variance = mean of “discarded” eigenvalues)}
\]

where \( U_K \) is \( D \times K \) matrix of top \( K \) eigvecs of \( S \),

† Probabilistic Principal Component Analysis (Tipping and Bishop, 1999)
MLE for PPCA (or why it is hard..)

To do MLE, we need to maximize \( \log p(X|W, \sigma^2) = \sum_{n=1}^{N} \log p(x_n|W, \sigma^2) \) with \( z_n \) integrated out.

MLE on the objective \( p(x_n|W, \sigma^2) \) can be done but turns out to be a bit expensive. In particular:

\[
\log p(X|\Theta) = -\frac{N}{2} (D \log 2\pi + \log |C| + \text{trace}(C^{-1}S))
\]

where \( S \) is the data covariance matrix, \( C^{-1} = \sigma^{-1}I - \sigma^{-1}WM^{-1}W^\top \) and \( M = W^\top W + \sigma^2I \)

The MLE solution is given by (don’t worry about the proof)\(^\dagger\)

\[
W_{ML} = U_K(L_K - \sigma^2_{ML}I)^{1/2}R
\]

\[
\sigma^2_{ML} = \frac{1}{D-K} \sum_{k=K+1}^{D} \lambda_k \text{ (noise variance = mean of “discarded” eigenvalues)}
\]

where \( U_K \) is \( D \times K \) matrix of top \( K \) eigvecs of \( S \), \( L_K: K \times K \) diagonal matrix of top \( K \) eigvals \( \lambda_1, \ldots, \lambda_K \),

-----

\(^\dagger\) Probabilistic Principal Component Analysis (Tipping and Bishop, 1999)
MLE for PPCA (or why it is hard..)

- To do MLE, we need to maximize \( \log p(X|W, \sigma^2) = \sum_{n=1}^{N} \log p(x_n|W, \sigma^2) \) with \( z_n \) integrated out.

- MLE on the objective \( p(x_n|W, \sigma^2) \) can be done but turns out to be a bit expensive. In particular:

  \[
  \log p(X|\Theta) = -\frac{N}{2} (D \log 2\pi + \log |C| + \text{trace}(C^{-1}S))
  \]

  where \( S \) is the data covariance matrix, \( C^{-1} = \sigma^{-1}I - \sigma^{-1}WM^{-1}W^\top \) and \( M = W^\top W + \sigma^2I \)

- The MLE solution is given by (don’t worry about the proof)\(^\dagger\)

  \[
  W_{ML} = U_K(L_K - \sigma_{ML}^2 I)^{1/2} R
  \]

  \[
  \sigma_{ML}^2 = \frac{1}{D-K} \sum_{k=K+1}^{D} \lambda_k \quad \text{(noise variance = mean of “discarded” eigenvalues)}
  \]

  where \( U_K \) is \( D \times K \) matrix of \textit{top K eigvecs} of \( S \), \( L_K: K \times K \) diagonal matrix of \textit{top K eigvals} \( \lambda_1, \ldots, \lambda_K \), \( R \) is a \( K \times K \) arbitrary rotation matrix.

---

\(^\dagger\) Probabilistic Principal Component Analysis (Tipping and Bishop, 1999)
MLE for PPCA (or why it is hard..)

- To do MLE, we need to maximize \( \log p(X|W, \sigma^2) = \sum_{n=1}^{N} \log p(x_n|W, \sigma^2) \) with \( z_n \) integrated out.

- MLE on the objective \( p(x_n|W, \sigma^2) \) can be done but turns out to be a bit expensive. In particular:

\[
\log p(X|\Theta) = -\frac{N}{2} (D \log 2\pi + \log |C| + \text{trace}(C^{-1}S))
\]

where \( S \) is the data covariance matrix, \( C^{-1} = \sigma^{-1}I - \sigma^{-1}WM^{-1}W^T \) and \( M = W^T W + \sigma^2 I \).

- The MLE solution is given by (don’t worry about the proof)\(^\dagger\)

\[
W_{ML} = U_K (L_K - \sigma_{ML}^2 I)^{1/2} R
\]

\[
\sigma_{ML}^2 = \frac{1}{D - K} \sum_{k=K+1}^{D} \lambda_k
\]

(noise variance = mean of “discarded” eigenvalues)

where \( U_K \) is \( D \times K \) matrix of top \( K \) eigvecs of \( S \), \( L_K \): \( K \times K \) diagonal matrix of top \( K \) eigvals \( \lambda_1, \ldots, \lambda_K \), \( R \) is a \( K \times K \) arbitrary rotation matrix (equivalent to PCA for \( R = I \) and \( \sigma^2 \to 0 \))

\(^\dagger\) Probabilistic Principal Component Analysis (Tipping and Bishop, 1999)
MLE for PPCA (or why it is hard..)

- To do MLE, we need to maximize \( \log p(\mathbf{X}|\mathbf{W}, \sigma^2) = \sum_{n=1}^{N} \log p(\mathbf{x}_n|\mathbf{W}, \sigma^2) \) with \( \mathbf{z}_n \) integrated out.

- MLE on the objective \( p(\mathbf{x}_n|\mathbf{W}, \sigma^2) \) can be done but turns out to be a bit expensive. In particular:

\[
\log p(\mathbf{X}|\Theta) = -\frac{N}{2} \left( D \log 2\pi + \log |\mathbf{C}| + \text{trace}(\mathbf{C}^{-1}\mathbf{S}) \right)
\]

where \( \mathbf{S} \) is the data covariance matrix, \( \mathbf{C}^{-1} = \sigma^{-1} \mathbf{I} - \sigma^{-1} \mathbf{W} \mathbf{M}^{-1} \mathbf{W}^\top \) and \( \mathbf{M} = \mathbf{W}^\top \mathbf{W} + \sigma^2 \mathbf{I} \)

- The MLE solution is given by (don’t worry about the proof)\(^\dagger\)

\[
\mathbf{W}_{ML} = \mathbf{U}_K (\mathbf{L}_K - \sigma_{ML}^2 \mathbf{I})^{1/2} \mathbf{R}
\]

\[
\sigma_{ML}^2 = \frac{1}{D-K} \sum_{k=K+1}^{D} \lambda_k \quad \text{(noise variance = mean of “discarded” eigenvalues)}
\]

where \( \mathbf{U}_K \) is \( D \times K \) matrix of top \( K \) eigvecs of \( \mathbf{S} \), \( \mathbf{L}_K \): \( K \times K \) diagonal matrix of top \( K \) eigvals \( \lambda_1, \ldots, \lambda_K \), \( \mathbf{R} \) is a \( K \times K \) arbitrary rotation matrix (equivalent to PCA for \( \mathbf{R} = \mathbf{I} \) and \( \sigma^2 \to 0 \))

- Need to do eigen-decomposition of \( D \times D \) data covariance matrix \( \mathbf{S} \). EXPENSIVE!!!

\(^\dagger\) Probabilistic Principal Component Analysis (Tipping and Bishop, 1999)
Learning PPCA via EM

Instead of maximizing the ILL log $p(X|W, \sigma^2) = \mathcal{N}(0, WW^T + \sigma^2 I_D)$, EM maximizes exp. CLL

This is done by iterating between the following two steps:

1. **E Step:** Infer the posterior $p(z_n|x_n)$ given current estimate of $\Theta = (W, \sigma^2)$ (needed for expectations)

   $$p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^T x_n, \sigma^2 M^{-1})$$

   where $M = WW^T + \sigma^2 I_K$

2. **M Step:** Maximize the expected complete data log-lik. (CLL)

   $E[\log p(X, Z|\Theta)]$ w.r.t. $\Theta$

The CLL (and expected CLL) for PPCA has a simple expression.

The CLL is

$$\log p(X, Z|W, \sigma^2) = \log \prod_{n=1}^{N} p(x_n, z_n|W, \sigma^2) = \log \prod_{n=1}^{N} p(x_n|z_n, W, \sigma^2) p(z_n)$$

$$= N \sum_{n=1}^{N} \{ \log p(x_n|z_n, W, \sigma^2) + \log p(z_n) \}$$

Using $p(x_n|z_n, W, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp \left[ -\frac{(x_n - Wz_n)^T(x_n - Wz_n)}{2\sigma^2} \right]$ and $p(z_n) \propto \exp \left[ -z_n^T z_n \right]$ and simplifying

$$\text{CLL} = -N \sum_{n=1}^{N} \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2} \sigma^2 ||x_n||^2 - \frac{1}{2} \sigma^2 z_n^T W^T x_n + \frac{1}{2} \sigma^2 \text{tr}(z_n z_n^T W^T W) + \frac{1}{2} \text{tr}(z_n z_n^T) \right\}$$

(Exercise: Verify)
Learning PPCA via EM

Instead of maximizing the ILL log \( p(X|W, \sigma^2) = \mathcal{N}(0, WW^T + \sigma^2 I_D) \), EM maximizes exp. CLL

This is done by iterating between the following two steps
Learning PPCA via EM

- Instead of maximizing the ILL log $p(X|W, \sigma^2) = \mathcal{N}(0, WW^T + \sigma^2 I_D)$, EM maximizes exp. CLL

- This is done by iterating between the following two steps

  - E Step: Infer the posterior $p(z_n|x_n)$ given current estimate of $\Theta = (W, \sigma^2)$
Learning PPCA via EM

- Instead of maximizing the ILL log $p(X|W, \sigma^2) = N(0, WW^T + \sigma^2 I_D)$, EM maximizes exp. CLL
- This is done by iterating between the following two steps
  - E Step: Infer the posterior $p(z_n|x_n)$ given current estimate of $\Theta = (W, \sigma^2)$ (needed for expectations)
Learning PPCA via EM

- Instead of maximizing the ILL log $p(X|W, \sigma^2) = \mathcal{N}(0, WW^\top + \sigma^2 I_D)$, EM maximizes exp. CLL
- This is done by iterating between the following two steps:
  - E Step: Infer the posterior $p(z_n|x_n)$ given current estimate of $\Theta = (W, \sigma^2)$ (needed for expectations)
    $$p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^\top x_n, \sigma^2 M^{-1})$$
    (where $M = W^\top W + \sigma^2 I_K$)
Learning PPCA via EM

- Instead of maximizing the ILL log $p(X|W, \sigma^2) = \mathcal{N}(0, WW^\top + \sigma^2 I_D)$, EM maximizes exp. CLL

- This is done by iterating between the following two steps
  
  - E Step: Infer the posterior $p(z_n|x_n)$ given current estimate of $\Theta = (W, \sigma^2)$ (needed for expectations)
    
    $$p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^\top x_n, \sigma^2 M^{-1})$$
    
    (where $M = W^\top W + \sigma^2 I_K$)

  - M Step: Maximize the expected complete data log-lik. (CLL) $\mathbb{E}[\log p(X, Z|\Theta)]$ w.r.t. $\Theta$
Learning PPCA via EM

- Instead of maximizing the ILL log $p(X|W, \sigma^2) = \mathcal{N}(0, WW^\top + \sigma^2 I_D)$, EM maximizes exp. CLL
- This is done by iterating between the following two steps
  - E Step: Infer the posterior $p(z_n|x_n)$ given current estimate of $\Theta = (W, \sigma^2)$ (needed for expectations)
    
    $p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^\top x_n, \sigma^2 M^{-1})$ (where $M = W^\top W + \sigma^2 I_K$)
  - M Step: Maximize the expected complete data log-lik. (CLL) $\mathbb{E}[\log p(X,Z|\Theta)]$ w.r.t. $\Theta$
- The CLL (and expected CLL) for PPCA has a simple expression.
Learning PPCA via EM

- Instead of maximizing the ILL log \( p(X|W, \sigma^2) = \mathcal{N}(0, WW^\top + \sigma^2 I_D) \), EM maximizes exp. CLL

- This is done by iterating between the following two steps
  - E Step: Infer the posterior \( p(z_n|x_n) \) given current estimate of \( \Theta = (W, \sigma^2) \) (needed for expectations)
    \[
p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^\top x_n, \sigma^2 M^{-1})
    \]
    (where \( M = W^\top W + \sigma^2 I_K \))
  - M Step: Maximize the expected complete data log-lik. (CLL) \( \mathbb{E}[\log p(X, Z|\Theta)] \) w.r.t. \( \Theta \)

- The CLL (and expected CLL) for PPCA has a simple expression. The CLL is
Learning PPCA via EM

- Instead of maximizing the ILL log $p(X|W, \sigma^2) = \mathcal{N}(0, WW^\top + \sigma^2 I_D)$, EM maximizes exp. CLL

- This is done by iterating between the following two steps
  - E Step: Infer the posterior $p(z_n|x_n)$ given current estimate of $\Theta = (W, \sigma^2)$ (needed for expectations)
    \[
    p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^\top x_n, \sigma^2 M^{-1})
    \] (where $M = W^\top W + \sigma^2 I_K$)
  - M Step: Maximize the expected complete data log-lik. (CLL) $\mathbb{E}[\log p(X, Z|\Theta)]$ w.r.t. $\Theta$

- The CLL (and expected CLL) for PPCA has a simple expression. The CLL is
  \[
  \log p(X, Z|W, \sigma^2)
  \]
Learning PPCA via EM

- Instead of maximizing the ILL \( \log p(X|W, \sigma^2) = \mathcal{N}(0, WW^\top + \sigma^2 I_D) \), EM maximizes exp. CLL

- This is done by iterating between the following two steps
  - E Step: Infer the posterior \( p(z_n|x_n) \) given current estimate of \( \Theta = (W, \sigma^2) \) (needed for expectations)
    \[
p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^\top x_n, \sigma^2 M^{-1})
    \]
    (where \( M = W^\top W + \sigma^2 I_K \))
  - M Step: Maximize the expected complete data log-lik. \( \mathbb{E}[\log p(X, Z|\Theta)] \) w.r.t. \( \Theta \)

- The CLL (and expected CLL) for PPCA has a simple expression. The CLL is
  \[
  \log p(X, Z|W, \sigma^2) = \log \prod_{n=1}^N p(x_n, z_n|W, \sigma^2)
  \]
Learning PPCA via EM

- Instead of maximizing the ILL \( \log p(X|W, \sigma^2) = \mathcal{N}(0, WW^\top + \sigma^2 I_D) \), EM maximizes exp. CLL

- This is done by iterating between the following two steps
  - E Step: Infer the posterior \( p(z_n|x_n) \) given current estimate of \( \Theta = (W, \sigma^2) \) (needed for expectations)
    \[
p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^\top x_n, \sigma^2 M^{-1}) \quad \text{(where } M = W^\top W + \sigma^2 I_K)\]
  - M Step: Maximize the expected complete data log-lik. (CLL) \( \mathbb{E}[\log p(X, Z|\Theta)] \) w.r.t. \( \Theta \)

- The CLL (and expected CLL) for PPCA has a simple expression. The CLL is
  \[
  \log p(X, Z|W, \sigma^2) = \log \prod_{n=1}^N p(x_n, z_n|W, \sigma^2) = \log \prod_{n=1}^N p(x_n|z_n, W, \sigma^2)p(z_n)
  \]
Learning PPCA via EM

- Instead of maximizing the ILL \( \log p(X|W, \sigma^2) = \mathcal{N}(0, WW^T + \sigma^2 I_D) \), EM maximizes exp. CLL

- This is done by iterating between the following two steps
  
  - **E Step:** Infer the posterior \( p(z_n|x_n) \) given current estimate of \( \Theta = (W, \sigma^2) \) (needed for expectations)
    
    \[
p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^T x_n, \sigma^2 M^{-1})
    \]
    (where \( M = W^T W + \sigma^2 I_K \))

  - **M Step:** Maximize the expected complete data log-lik. (CLL) \( \mathbb{E}[\log p(X, Z|\Theta)] \) w.r.t. \( \Theta \)

- The CLL (and expected CLL) for PPCA has a simple expression. The CLL is
  
  \[
  \log p(X, Z|W, \sigma^2) = \log \prod_{n=1}^{N} p(x_n, z_n|W, \sigma^2) = \log \prod_{n=1}^{N} p(x_n|z_n, W, \sigma^2)p(z_n) = \sum_{n=1}^{N} \{ \log p(x_n|z_n, W, \sigma^2) + \log p(z_n) \}
  \]
Learning PPCA via EM

- Instead of maximizing the ILL \( \log p(X|W, \sigma^2) = \mathcal{N}(0, WW^\top + \sigma^2 I_D) \), EM maximizes exp. CLL

- This is done by iterating between the following two steps
  - **E Step:** Infer the posterior \( p(z_n|x_n) \) given current estimate of \( \Theta = (W, \sigma^2) \) (needed for expectations)
    \[
p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^\top x_n, \sigma^2 M^{-1})
    \]
    (where \( M = WW^\top + \sigma^2 I_K \))
  - **M Step:** Maximize the expected complete data log-lik. (CLL) \( \mathbb{E}[\log p(X, Z|\Theta)] \) w.r.t. \( \Theta \)

- The CLL (and expected CLL) for PPCA has a simple expression. The CLL is
  \[
  \log p(X, Z|W, \sigma^2) = \log \prod_{n=1}^{N} p(x_n, z_n|W, \sigma^2) = \log \prod_{n=1}^{N} p(x_n|z_n, W, \sigma^2) p(z_n) = \sum_{n=1}^{N} \{ \log p(x_n|z_n, W, \sigma^2) + \log p(z_n) \}
  \]

- Using \( p(x_n|z_n, W, \sigma^2) = \frac{1}{(2\pi \sigma^2)^{D/2}} \exp \left[ -\frac{(x_n - Wz_n)^\top (x_n - Wz_n)}{2\sigma^2} \right] \)
Learning PPCA via EM

Instead of maximizing the ILL log \( p(X|W, \sigma^2) = \mathcal{N}(0, WW^T + \sigma^2 I_D) \), EM maximizes exp. CLL

This is done by iterating between the following two steps

- **E Step:** Infer the posterior \( p(z_n|x_n) \) given current estimate of \( \Theta = (W, \sigma^2) \) (needed for expectations)
  \[
p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^T x_n, \sigma^2 M^{-1}) \]  
  (where \( M = W^T W + \sigma^2 I_K \))

- **M Step:** Maximize the expected complete data log-lik. (CLL) \( \mathbb{E}[\log p(X, Z|\Theta)] \) w.r.t. \( \Theta \)

The CLL (and expected CLL) for PPCA has a simple expression. The CLL is

\[
\log p(X, Z|W, \sigma^2) = \log \prod_{n=1}^{N} p(x_n, z_n|W, \sigma^2) = \log \prod_{n=1}^{N} p(x_n|z_n, W, \sigma^2)p(z_n) = \sum_{n=1}^{N} \{ \log p(x_n|z_n, W, \sigma^2) + \log p(z_n) \}
\]

Using \( p(x_n|z_n, W, \sigma^2) = \frac{1}{(2\pi\sigma^2)^D/2} \exp \left[ -\frac{(x_n-Wz_n)^T(x_n-Wz_n)}{2\sigma^2} \right] \) and \( p(z_n) \propto \exp \left[ -\frac{z_n^T z_n}{2} \right] \)
Learning PPCA via EM

- Instead of maximizing the ILL log \( p(X|W, \sigma^2) = \mathcal{N}(0, WW^\top + \sigma^2 I_D) \), EM maximizes exp. CLL.

- This is done by iterating between the following two steps:
  - **E Step:** Infer the posterior \( p(z_n|x_n) \) given current estimate of \( \Theta = (W, \sigma^2) \) (needed for expectations)
    \[
p(z_n|x_n, W, \sigma^2) = \mathcal{N}(M^{-1}W^\top x_n, \sigma^2 M^{-1})
    \]
      (where \( M = WW^\top + \sigma^2 I_K \))
  - **M Step:** Maximize the expected complete data log-lik. (CLL) \( \mathbb{E}[\log p(X, Z|\Theta)] \) w.r.t. \( \Theta \)

- The CLL (and expected CLL) for PPCA has a simple expression. The CLL is
  \[
  \log p(X, Z|W, \sigma^2) = \log \prod_{n=1}^{N} p(x_n, z_n|W, \sigma^2) = \log \prod_{n=1}^{N} p(x_n|z_n, W, \sigma^2)p(z_n) = \sum_{n=1}^{N} \{ \log p(x_n|z_n, W, \sigma^2) + \log p(z_n) \}
  \]

- Using \( p(x_n|z_n, W, \sigma^2) = \frac{1}{(2\pi \sigma^2)^{D/2}} \exp \left[ -\frac{(x_n - Wz_n)^\top (x_n - Wz_n)}{2\sigma^2} \right] \) and \( p(z_n) \propto \exp \left[ -\frac{z_n^\top z_n}{2} \right] \) and simplifying

  \[
  \text{CLL} = -\sum_{n=1}^{N} \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2\sigma^2} ||x_n||^2 - \frac{1}{\sigma^2} z_n^\top W^\top x_n + \frac{1}{2\sigma^2} \text{tr}(z_n z_n^\top W^\top W) + \frac{1}{2} \text{tr}(z_n z_n^\top) \right\}
  \]
  (Exercise: Verify)
Learning PPCA via EM

The expected complete data log-likelihood $\mathbb{E}[\log p(\mathbf{x}, \mathbf{z} | \mathbf{w}, \sigma^2)]$

$$= - \sum_{n=1}^{N} \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2\sigma^2} ||\mathbf{x}_n||^2 - \frac{1}{\sigma^2} \mathbb{E}[\mathbf{z}_n]^\top \mathbf{w}^\top \mathbf{x}_n + \frac{1}{2\sigma^2} \text{tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top] \mathbf{w}^\top \mathbf{w}) + \frac{1}{2} \text{tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top]) \right\}$$
Learning PPCA via EM

- The expected complete data log-likelihood $\mathbb{E}[\log p(X, Z|W, \sigma^2)]$
  $$= - \sum_{n=1}^{N} \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2\sigma^2} ||x_n||^2 - \frac{1}{\sigma^2} \mathbb{E}[z_n]^\top W^\top x_n + \frac{1}{2\sigma^2} \text{tr}(\mathbb{E}[z_n z_n^\top] W^\top W) + \frac{1}{2} \text{tr}(\mathbb{E}[z_n z_n^\top]) \right\}$$

- Taking the derivative of $\mathbb{E}[\log p(X, Z|W, \sigma^2)]$ w.r.t. $W$ and setting to zero

$$W = \left[ \sum_{n=1}^{N} x_n \mathbb{E}[z_n] \right]^{-1} \left[ \sum_{n=1}^{N} \mathbb{E}[z_n z_n^\top] \right]^{-1}$$

(Exercise: verify; can also be done “online”)
Learning PPCA via EM

- The expected complete data log-likelihood \( E[\log p(X, Z|W, \sigma^2)] \)

\[
= - \sum_{n=1}^{N} \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2\sigma^2} ||x_n||^2 - \frac{1}{\sigma^2} E[z_n]^T W^T x_n + \frac{1}{2\sigma^2} \text{tr}(E[z_n z_n^T]W^T W) + \frac{1}{2} \text{tr}(E[z_n z_n^T]) \right\}
\]

- Taking the derivative of \( E[\log p(X, Z|W, \sigma^2)] \) w.r.t. \( W \) and setting to zero

\[
W = \left[ \sum_{n=1}^{N} x_n E[z_n^T] \right] \left[ \sum_{n=1}^{N} E[z_n z_n^T] \right]^{-1}
\]

(Exercise: verify; can also be done “online”)

- To compute \( W \), we need two posterior expectations \( E[z_n] \) and \( E[z_n z_n^T] \)
Learning PPCA via EM

The expected complete data log-likelihood $E[\log p(X, Z|W, \sigma^2)]$

$$= - \sum_{n=1}^{N} \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2\sigma^2} ||x_n||^2 - \frac{1}{\sigma^2} E[z_n]^T W^T x_n + \frac{1}{2\sigma^2} \text{tr}(E[z_n z_n^T]W^T W) + \frac{1}{2} \text{tr}(E[z_n z_n^T]) \right\}$$

Taking the derivative of $E[\log p(X, Z|W, \sigma^2)]$ w.r.t. $W$ and setting to zero

$$W = \left[ \sum_{n=1}^{N} x_n E[z_n]^T \right] \left[ \sum_{n=1}^{N} E[z_n z_n^T] \right]^{-1}$$

(Exercise: verify; can also be done “online”)

To compute $W$, we need two posterior expectations $E[z_n]$ and $E[z_n z_n^T]$

These can be easily obtained from the posterior $p(z_n|x_n)$ computed in E step

$$p(z_n|x_n, W) = \mathcal{N}(M^{-1}W^T x_n, \sigma^2 M^{-1})$$

where $M = W^T W + \sigma^2 I_K$
Learning PPCA via EM

- The expected complete data log-likelihood $\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$

  $$= - \sum_{n=1}^{N} \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2\sigma^2} ||\mathbf{x}_n||^2 - \frac{1}{\sigma^2} \mathbb{E}[\mathbf{z}_n]^{\top} \mathbf{W}^{\top} \mathbf{x}_n + \frac{1}{2\sigma^2} \text{tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\top}]\mathbf{W}^{\top}\mathbf{W}) + \frac{1}{2} \text{tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\top}]) \right\}$$

- Taking the derivative of $\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$ w.r.t. $\mathbf{W}$ and setting to zero

  $$\mathbf{W} = \left[ \sum_{n=1}^{N} \mathbf{x}_n \mathbb{E}[\mathbf{z}_n]^{\top} \right] \left[ \sum_{n=1}^{N} \mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\top}] \right]^{-1}$$

  (Exercise: verify; can also be done “online”)

- To compute $\mathbf{W}$, we need two posterior expectations $\mathbb{E}[\mathbf{z}_n]$ and $\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\top}]$

- These can be easily obtained from the posterior $p(\mathbf{z}_n|\mathbf{x}_n)$ computed in E step

  $$p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{W}) = \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^{\top}\mathbf{x}_n, \sigma^2\mathbf{M}^{-1})$$

  where $\mathbf{M} = \mathbf{W}^{\top}\mathbf{W} + \sigma^2 \mathbf{I}_K$

  $$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}^{-1}\mathbf{W}^{\top}\mathbf{x}_n$$
Learning PPCA via EM

- The expected complete data log-likelihood $\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$

$$= - \sum_{n=1}^{N} \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2\sigma^2} ||\mathbf{x}_n||^2 - \frac{1}{\sigma^2} \mathbb{E}[\mathbf{z}_n]^T \mathbf{W}^T \mathbf{x}_n + \frac{1}{2\sigma^2} \text{tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T] \mathbf{W}^T \mathbf{W}) + \frac{1}{2} \text{tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T]) \right\}$$

- Taking the derivative of $\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$ w.r.t. $\mathbf{W}$ and setting to zero

$$\mathbf{W} = \left[ \sum_{n=1}^{N} \mathbf{x}_n \mathbb{E}[\mathbf{z}_n]^T \right] \left[ \sum_{n=1}^{N} \mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T] \right]^{-1}$$

(Exercise: verify; can also be done “online”)

- To compute $\mathbf{W}$, we need two posterior expectations $\mathbb{E}[\mathbf{z}_n]$ and $\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T]$

- These can be easily obtained from the posterior $p(\mathbf{z}_n|\mathbf{x}_n)$ computed in E step

$$p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{W}) = \mathcal{N}(\mathbf{W}^{-1} \mathbf{W}^T \mathbf{x}_n, \sigma^2 \mathbf{M}^{-1})$$

where $\mathbf{M} = \mathbf{W}^T \mathbf{W} + \sigma^2 \mathbf{I}_K$

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}^{-1} \mathbf{W}^T \mathbf{x}_n$$

$$\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T] = \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^T + \text{cov}(\mathbf{z}_n)$$
Learning PPCA via EM

- The expected complete data log-likelihood $\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$

$$= - \sum_{n=1}^{N} \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2\sigma^2} ||\mathbf{x}_n||^2 - \frac{1}{\sigma^2} \mathbb{E}[\mathbf{z}_n]^\top \mathbf{W}^\top \mathbf{x}_n + \frac{1}{2\sigma^2} \text{tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top] \mathbf{W}^\top \mathbf{W}) + \frac{1}{2} \text{tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top]) \right\}$$

- Taking the derivative of $\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$ w.r.t. $\mathbf{W}$ and setting to zero

$$\mathbf{W} = \left[ \sum_{n=1}^{N} \mathbf{x}_n \mathbb{E}[\mathbf{z}_n]^\top \right] \left[ \sum_{n=1}^{N} \mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top] \right]^{-1}$$

(Exercise: verify; can also be done “online”)

- To compute $\mathbf{W}$, we need two posterior expectations $\mathbb{E}[\mathbf{z}_n]$ and $\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top]$

- These can be easily obtained from the posterior $p(\mathbf{z}_n|\mathbf{x}_n)$ computed in E step

$$\begin{align*}
p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{W}) &= \mathcal{N}(\mathbf{M}^{-1} \mathbf{W}^\top \mathbf{x}_n, \sigma^2 \mathbf{M}^{-1}) \quad \text{where } \mathbf{M} = \mathbf{W}^\top \mathbf{W} + \sigma^2 \mathbf{I}_K \\
\mathbb{E}[\mathbf{z}_n] &= \mathbf{M}^{-1} \mathbf{W}^\top \mathbf{x}_n \\
\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^\top] &= \mathbb{E}[\mathbf{z}_n] \mathbb{E}[\mathbf{z}_n]^\top + \text{cov}(\mathbf{z}_n) = \mathbb{E}[\mathbf{z}_n] \mathbb{E}[\mathbf{z}_n]^\top + \sigma^2 \mathbf{M}^{-1}
\end{align*}$$

Note: The noise variance $\sigma^2$ can also be estimated (take deriv., set to zero..)
Learning PPCA via EM

The expected complete data log-likelihood $\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$

$$= - \sum_{n=1}^{N} \left\{ \frac{D}{2} \log \sigma^2 + \frac{1}{2\sigma^2} ||\mathbf{x}_n||^2 - \frac{1}{\sigma^2} \mathbb{E}[\mathbf{z}_n]^\top \mathbf{W}^\top \mathbf{x}_n + \frac{1}{2\sigma^2} \text{tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^\top] \mathbf{W}^\top \mathbf{W}) + \frac{1}{2} \text{tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^\top]) \right\}$$

Taking the derivative of $\mathbb{E}[\log p(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \sigma^2)]$ w.r.t. $\mathbf{W}$ and setting to zero

$$\mathbf{W} = \left[ \sum_{n=1}^{N} \mathbf{x}_n \mathbb{E}[\mathbf{z}_n]^\top \right] \left[ \sum_{n=1}^{N} \mathbb{E}[\mathbf{z}_n\mathbf{z}_n^\top] \right]^{-1}$$

(Exercise: verify; can also be done “online”)

To compute $\mathbf{W}$, we need two posterior expectations $\mathbb{E}[\mathbf{z}_n]$ and $\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^\top]$

These can be easily obtained from the posterior $p(\mathbf{z}_n|\mathbf{x}_n)$ computed in E step

$$p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{W}) = \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^\top \mathbf{x}_n, \sigma^2 \mathbf{M}^{-1}) \quad \text{where} \quad \mathbf{M} = \mathbf{W}^\top \mathbf{W} + \sigma^2 \mathbf{I}_K$$

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}^{-1}\mathbf{W}^\top \mathbf{x}_n$$

$$\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^\top] = \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^\top + \text{cov}(\mathbf{z}_n) = \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^\top + \sigma^2 \mathbf{M}^{-1}$$

Note: The noise variance $\sigma^2$ can also be estimated (take deriv., set to zero..)
Summary: The Full EM Algorithm for PPCA

- Specify $K$, initialize $W$ and $\sigma^2$ randomly. Also center the data ($x_n = x_n - \frac{1}{N} \sum_{n=1}^{N} x_n$)
Specify \( K \), initialize \( W \) and \( \sigma^2 \) randomly. Also center the data (\( x_n = x_n - \frac{1}{N} \sum_{n=1}^{N} x_n \))

**E step:** For each \( n \), compute \( p(z_n|x_n) \) using current \( W \) and \( \sigma^2 \). Compute exp. for the M step
Specify $K$, initialize $W$ and $\sigma^2$ randomly. Also center the data ($x_n = x_n - \frac{1}{N} \sum_{n=1}^{N} x_n$)

- **E step**: For each $n$, compute $p(z_n|x_n)$ using current $W$ and $\sigma^2$. Compute exp. for the M step

$$p(z_n|x_n, W) = \mathcal{N}(M^{-1}W^T x_n, \sigma^2 M^{-1}) \quad \text{where } M = W^T W + \sigma^2 I_K$$

$$\mathbb{E}[z_n] = M^{-1}W^T x_n$$
Summary: The Full EM Algorithm for PPCA

- Specify $K$, initialize $W$ and $\sigma^2$ randomly. Also center the data ($x_n = x_n - \frac{1}{N} \sum_{n=1}^{N} x_n$)

- **E step:** For each $n$, compute $p(z_n | x_n)$ using current $W$ and $\sigma^2$. Compute exp. for the M step

\[
p(z_n | x_n, W) = \mathcal{N}(M^{-1}W^T x_n, \sigma^2 M^{-1}) \quad \text{where} \quad M = W^T W + \sigma^2 I_K
\]

\[
\mathbb{E}[z_n] = M^{-1}W^T x_n
\]

\[
\mathbb{E}[z_n z_n^T] = \text{cov}(z_n) + \mathbb{E}[z_n] \mathbb{E}[z_n]^T = \mathbb{E}[z_n] \mathbb{E}[z_n]^T + \sigma^2 M^{-1}
\]
Specify $K$, initialize $W$ and $\sigma^2$ randomly. Also center the data ($x_n = x_n - \frac{1}{N} \sum_{n=1}^{N} x_n$)

**E step:** For each $n$, compute $p(z_n|x_n)$ using current $W$ and $\sigma^2$. Compute exp. for the M step

\[
p(z_n|x_n, W) = \mathcal{N}(M^{-1}W^T x_n, \sigma^2 M^{-1}) \quad \text{where } M = W^T W + \sigma^2 I_K
\]

\[
E[z_n] = M^{-1}W^T x_n
\]

\[
E[z_n z_n^T] = \text{cov}(z_n) + E[z_n]E[z_n]^T = E[z_n]E[z_n]^T + \sigma^2 M^{-1}
\]

**M step:** Re-estimate $W$ and $\sigma^2$
Summary: The Full EM Algorithm for PPCA

- Specify $K$, initialize $W$ and $\sigma^2$ randomly. Also center the data ($x_n = x_n - \frac{1}{N} \sum_{n=1}^{N} x_n$)

- **E step:** For each $n$, compute $p(z_n|x_n)$ using current $W$ and $\sigma^2$. Compute exp. for the M step

  $$p(z_n|x_n, W) = \mathcal{N}(M^{-1}W^T x_n, \sigma^2 M^{-1})$$
  where $M = W^T W + \sigma^2 I_K$

  $$E[z_n] = M^{-1}W^T x_n$$

  $$E[z_n z_n^T] = \text{cov}(z_n) + E[z_n]E[z_n]^T = E[z_n]E[z_n]^T + \sigma^2 M^{-1}$$

- **M step:** Re-estimate $W$ and $\sigma^2$

  $$W_{new} = \left( \sum_{n=1}^{N} x_n E[z_n]^T \right) \left( \sum_{n=1}^{N} E[z_n z_n^T] \right)^{-1}$$
Summary: The Full EM Algorithm for PPCA

- Specify $K$, initialize $W$ and $\sigma^2$ randomly. Also center the data ($x_n = x_n - \frac{1}{N} \sum_{n=1}^{N} x_n$)

- **E step:** For each $n$, compute $p(z_n|x_n)$ using current $W$ and $\sigma^2$. Compute exp. for the M step

\[
p(z_n|x_n, W) = \mathcal{N}(M^{-1}W^T x_n, \sigma^2 M^{-1}) \quad \text{where} \quad M = W^T W + \sigma^2 I_K
\]

\[
E[z_n] = M^{-1}W^T x_n
\]

\[
E[z_n z_n^T] = \text{cov}(z_n) + E[z_n]E[z_n]^T = E[z_n]E[z_n]^T + \sigma^2 M^{-1}
\]

- **M step:** Re-estimate $W$ and $\sigma^2$

\[
W_{new} = \left( \sum_{n=1}^{N} x_n E[z_n]^T \right) \left( \sum_{n=1}^{N} E[z_n z_n^T] \right)^{-1}
\]

\[
\sigma_{new}^2 = \frac{1}{ND} \sum_{n=1}^{N} \left\{ ||x_n||^2 - 2E[z_n]^T W_{new} x_n + \text{tr} \left( E[z_n z_n^T] W_{new}^T W_{new} \right) \right\}
\]

Note: For $\sigma^2 = 0$, this EM algorithm can also be used to efficiently solve standard PCA (note that this EM algorithm doesn’t require any eigen-decomposition)
Summary: The Full EM Algorithm for PPCA

- Specify $K$, initialize $W$ and $\sigma^2$ randomly. Also center the data ($x_n = x_n - \frac{1}{N} \sum_{n=1}^{N} x_n$)
- **E step:** For each $n$, compute $p(z_n|x_n)$ using current $W$ and $\sigma^2$. Compute exp. for the M step
  
  \[ p(z_n|x_n, W) = \mathcal{N}(M^{-1}W^T x_n, \sigma^2 M^{-1}) \quad \text{where} \quad M = W^T W + \sigma^2 I_K \]
  
  \[ \mathbb{E}[z_n] = M^{-1} W^T x_n \]
  
  \[ \mathbb{E}[z_n z_n^T] = \text{cov}(z_n) + \mathbb{E}[z_n] \mathbb{E}[z_n]^T = \mathbb{E}[z_n] \mathbb{E}[z_n]^T + \sigma^2 M^{-1} \]

- **M step:** Re-estimate $W$ and $\sigma^2$
  
  \[ W_{new} = \left( \sum_{n=1}^{N} x_n \mathbb{E}[z_n]^T \right) \left( \sum_{n=1}^{N} \mathbb{E}[z_n z_n^T] \right)^{-1} \]
  
  \[ \sigma_{new}^2 = \frac{1}{ND} \sum_{n=1}^{N} \left\{ ||x_n||^2 - 2 \mathbb{E}[z_n]^T W_{new}^T x_n + \text{tr} \left( \mathbb{E}[z_n z_n^T] W_{new}^T W_{new} \right) \right\} \]

- Set $W = W_{new}$ and $\sigma^2 = \sigma_{new}^2$. If not converged (monitor $p(X|\Theta)$), go back to E step
Summary: The Full EM Algorithm for PPCA

- Specify $K$, initialize $W$ and $\sigma^2$ randomly. Also center the data ($x_n = x_n - \frac{1}{N} \sum_{n=1}^{N} x_n$)

- **E step:** For each $n$, compute $p(z_n|x_n)$ using current $W$ and $\sigma^2$. Compute exp. for the M step

$$p(z_n|x_n, W) = \mathcal{N}(M^{-1}W^T x_n, \sigma^2 M^{-1})$$

where $M = W^T W + \sigma^2 I_K$

$$E[z_n] = M^{-1}W^T x_n$$

$$E[z_n z_n^T] = \text{cov}(z_n) + E[z_n] E[z_n]^T = E[z_n] E[z_n]^T + \sigma^2 M^{-1}$$

- **M step:** Re-estimate $W$ and $\sigma^2$

$$W_{new} = \left[ \sum_{n=1}^{N} x_n E[z_n]^T \right] \left[ \sum_{n=1}^{N} E[z_n z_n^T] \right]^{-1}$$

$$\sigma^2_{new} = \frac{1}{ND} \sum_{n=1}^{N} \left\{ \|x_n\|^2 - 2 E[z_n]^T W_{new}^T x_n + \text{tr} \left( E[z_n z_n^T] W_{new}^T W_{new} \right) \right\}$$

- Set $W = W_{new}$ and $\sigma^2 = \sigma^2_{new}$. If not converged (monitor $p(X|\Theta)$), go back to E step

- **Note:** For $\sigma^2 = 0$, this EM algorithm can also be used to efficiently solve standard PCA (note that this EM algorithm doesn’t require any eigen-decomposition)