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

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

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

- 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 Θ by solving a slightly modified problem

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

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

- But since $\mathbf{Z}$ and Θ are usually "coupled", both ALT-OPT and EM need an alternating procedure
- Instead of maximizing $\log p(\mathbf{X}|\Theta)$ (incomplete-data log-lik - ILL), they maximize $\log p(\mathbf{X}, \mathbf{Z}|\Theta)$ (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 $\mathbf{Z}$ as $\hat{\mathbf{Z}} = \arg\max_{\mathbf{Z}} \log p(\mathbf{Z}|\mathbf{X}, \hat{\Theta})$
  3. Estimate Θ as $\hat{\Theta} = \arg\max_{\Theta} \log p(\mathbf{X}, \hat{\mathbf{Z}}|\Theta)$
  4. Go to step 2 if not converged
- Step 2 ($\arg\max$) of ALT-OPT could potentially throw away a lot of information about $\mathbf{Z}$
- 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 Θ by maximizing the expected CLL $\hat{\Theta} = \mathbb{E}_{p(\mathbf{Z}|\mathbf{X},\hat{\Theta})}[\log p(\mathbf{X}, \mathbf{Z}|\Theta)]$
  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

Brief Detour: Generative Stories

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 Θ

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

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

$$\begin{aligned} \boldsymbol{z}_n &\sim p(\boldsymbol{z}|\Theta) && (\boldsymbol{z}_n \text{ can be an observed label } y_n \text{ or a latent variable, e.g., cluster id}) \\ \boldsymbol{x}_n &\sim p(\boldsymbol{x}|\boldsymbol{z} = \boldsymbol{z}_n, \Theta) && (\boldsymbol{x}_n \text{ is generated conditioned on } \boldsymbol{z}_n) \end{aligned}$$

- This procedure generates $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}_{n=1}^N$ from the joint distribution $p(\boldsymbol{x}, \boldsymbol{z}|\Theta) = p(\boldsymbol{z}|\Theta)p(\boldsymbol{x}|\boldsymbol{z}, \Theta)$
- Note: In this story, we don't show step 1 if we aren't using any prior distribution on Θ
- Note: If there are no labels or latent variables $\boldsymbol{z}_n$, then we will just have $\boldsymbol{x}_n \sim p(\boldsymbol{x}|\Theta)$

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 Θ 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 $\boldsymbol{x}_n$ as $\boldsymbol{x}_n \sim \mathcal{N}(\mu_{y_n}, \Sigma_{y_n})$

Gaussian Mixture Model

- For $n = 1, \ldots, N$
  - Generate z_n as $z_n \sim \text{multinoulli}(\pi_1, \ldots, \pi_K)$
  - Generate $\boldsymbol{x}_n$ as $\boldsymbol{x}_n \sim \mathcal{N}(\mu_{z_n}, \Sigma_{z_n})$

Probabilistic Dimensionality Reduction (Probabilistic PCA)
(assuming data and latent variables to be Gaussians)

- For $n = 1, \ldots, N$
  - Generate $\boldsymbol{z}_n$ as $\boldsymbol{z}_n \sim \mathcal{N}(0, \mathbf{I}_K)$
  - Generate $\boldsymbol{x}_n$ as $\boldsymbol{x}_n \sim \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2\mathbf{I}_D)$

Discriminative Models for Regression/Classification

- For $n = 1, \ldots, N$
  - Generate y_n as

x not modeled

$$
\begin{aligned}
y_n &\sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2) \\
y_n &\sim \text{Bernoulli}(\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))
\end{aligned}
$$

- The model need not have latent variables (e.g. generative classification, discriminative models)

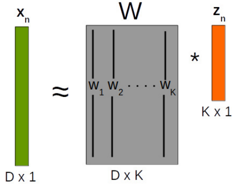
Latent Variable Models for Dimensionality Reduction

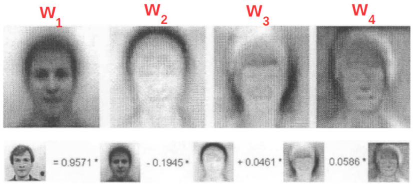
A Simple Model for Data Compression/Dimensionality-Reduction

- Consider a set of observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, with $\boldsymbol{x}_n \in \mathbb{R}^D$
- Let's approximate each $\boldsymbol{x}_n$ by a linear combination of K vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K$ $(K \ll D)$

$$\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk} \boldsymbol{w}_k \qquad \text{or} \qquad \boldsymbol{x}_n \approx \mathbf{W} \boldsymbol{z}_n$$

where $\mathbf{W} = [\boldsymbol{w}_1 \ldots \boldsymbol{w}_K]$ is $D \times K$, each $\boldsymbol{w}_k \in \mathbb{R}^D$, and $\boldsymbol{z}_n = [z_{n1} \ldots z_{nK}] \in \mathbb{R}^K$





- z_{nk} tell us much of "component" $\boldsymbol{w}_k$ is present in the observation $\boldsymbol{x}_n$
- Can think of $\boldsymbol{z}_n \in \mathbb{R}^K$ as a "compressed" latent representation of $\boldsymbol{x}_n \in \mathbb{R}^D$
- A good compression $\boldsymbol{z}_n$ will be one for which $\boldsymbol{x}_n$ is as close as possible to $\mathbf{W}\boldsymbol{z}_n$

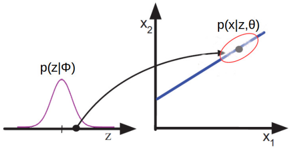
Dimensionality Reduction: The Probabilistic/Generative View

- In the linear model, we represented $\boldsymbol{x}_n$ approximately as $\boldsymbol{x}_n \approx \mathbf{W}\boldsymbol{z}_n$
- The probabilistic view: Model $\boldsymbol{x}_n$ by a D-dim Gaussian with mean vector $\mathbf{W}\boldsymbol{z}_n$

$$p(\boldsymbol{x}_n|\boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2\mathbf{I}_D)$$

$$\text{Equivalently:} \quad \boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n \quad \text{where } \epsilon_n \sim \mathcal{N}(0, \sigma^2\mathbf{I}_D)$$

- Let's assume a prior $p(\boldsymbol{z}_n) = \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$ on the latent variable $\boldsymbol{z}_n$
- A low-dim latent variable $\boldsymbol{z}_n$ transformed to "generate" a high-dim observation $\boldsymbol{x}_n$
- 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

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

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

- Consider modeling the same data using PPCA: $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\mathbf{W}\boldsymbol{z}, \sigma^2\mathbf{I}_D)$, $p(\boldsymbol{z}) = \mathcal{N}(0, \mathbf{I}_K)$
- For this Gaussian PPCA, the marginal distribution $p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{z})d\boldsymbol{z}$ is

$$\boxed{p(\boldsymbol{x}|\mathbf{W}, \sigma^2) = \mathcal{N}(0, \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}_D)} \qquad \text{(using Gaussian marginal results)}$$

- 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

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

(a) Training data (b) Random samples

  - 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 $\boldsymbol{x}_n$ has part of it as missing.

Learning the PPCA Model

- Since we are doing dim-red, the goal is to "recover" $\boldsymbol{z}_n$ (and $\mathbf{W}, \sigma^2$) given $\boldsymbol{x}_n$, $\forall n$
- The likelihood $p(\boldsymbol{x}_n|\boldsymbol{z}_n) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2\mathbf{I}_D)$ is Gaussian. The loss function = NLL will be

$$\begin{aligned} \mathcal{L}(\mathbf{Z}, \mathbf{W}, \sigma^2) &= \frac{1}{2\sigma^2}\sum_{n=1}^{N}||\boldsymbol{x}_n - \mathbf{W}\boldsymbol{z}_n||^2 + \frac{ND}{2}\log(2\pi\sigma^2) \qquad \text{(Exercise: Verify)} \\ &= \frac{1}{2\sigma^2}||\mathbf{X} - \mathbf{Z}\mathbf{W}^\top||_F^2 + \frac{ND}{2}\log(2\pi\sigma^2) \qquad (\mathbf{X}: N \times D, \mathbf{Z}: N \times K, \mathbf{W}: D \times K) \end{aligned}$$

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

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

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

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

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

Learning PPCA via ALT-OPT

- We saw that the PPCA problem reduced to

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

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

  1. Initialize $\mathbf{Z} = \hat{\mathbf{Z}}$
  2. Solve $\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} ||\mathbf{X} - \hat{\mathbf{Z}}\mathbf{W}^\top||_F^2$
  3. Solve $\hat{\mathbf{Z}} = \arg\min_{\mathbf{Z}} ||\mathbf{X} - \mathbf{Z}\hat{\mathbf{W}}^\top||_F^2$
  4. Go to step 2 if not yet converged

- Step 2 is just like multi-output regression with $\hat{\mathbf{Z}}$ as feature matrix and $\mathbf{X}$ as labal matrix
- Step is also like multi-output regression
- Note that the problem is essentially a matrix factorization of $\mathbf{X}$

MLE for PPCA (or why it is hard..)

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

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

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)[†]

$$\begin{aligned}
\mathbf{W}_{ML} &= \mathbf{U}_K(\mathbf{L}_K - \sigma^2_{ML}\mathbf{I})^{1/2}\mathbf{R} \\
\sigma^2_{ML} &= \frac{1}{D-K}\sum_{k=K+1}^{D}\lambda_k \quad \text{(noise variance = mean of ``discarded'' eigenvalues)}
\end{aligned}$$

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!!!

[†] Probabilistic Principal Component Analysis (Tipping and Bishop, 1999)

Learning PPCA via EM

- Instead of maximizing the ILL $\log p(\mathbf{X}|\mathbf{W},\sigma^2) = \mathcal{N}(0, \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}_D)$, EM maximizes exp. CLL
- This is done by iterating between the following two steps
  - E Step: Infer the posterior $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$ given current estimate of $\Theta = (\mathbf{W},\sigma^2)$ (needed for expectations)

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

  - M Step: Maximize the expected complete data log-lik. (CLL) $\mathbb{E}[\log p(\mathbf{X},\mathbf{Z}|\Theta)]$ w.r.t. Θ
- The CLL (and expected CLL) for PPCA has a simple expression. The CLL is

$$\log p(\mathbf{X},\mathbf{Z}|\mathbf{W},\sigma^2) = \log\prod_{n=1}^{N} p(\boldsymbol{x}_n,\boldsymbol{z}_n|\mathbf{W},\sigma^2) = \log\prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{z}_n,\mathbf{W},\sigma^2)p(\boldsymbol{z}_n) = \sum_{n=1}^{N}\{\log p(\boldsymbol{x}_n|\boldsymbol{z}_n,\mathbf{W},\sigma^2) + \log p(\boldsymbol{z}_n)\}$$

- Using $p(\boldsymbol{x}_n|\boldsymbol{z}_n,\mathbf{W},\sigma^2) = \frac{1}{(2\pi\sigma^2)^{D/2}}\exp\left[-\frac{(\boldsymbol{x}_n-\mathbf{W}\boldsymbol{z}_n)^\top(\boldsymbol{x}_n-\mathbf{W}\boldsymbol{z}_n)}{2\sigma^2}\right]$ and $p(\boldsymbol{z}_n)\propto\exp\left[-\frac{\boldsymbol{z}_n^\top\boldsymbol{z}_n}{2}\right]$ and simplifying

$$\text{CLL} = -\sum_{n=1}^{N}\left\{\frac{D}{2}\log\sigma^2 + \frac{1}{2\sigma^2}||\boldsymbol{x}_n||^2 - \frac{1}{\sigma^2}\boldsymbol{z}_n^\top\mathbf{W}^\top\boldsymbol{x}_n + \frac{1}{2\sigma^2}\text{tr}(\boldsymbol{z}_n\boldsymbol{z}_n^\top\mathbf{W}^\top\mathbf{W}) + \frac{1}{2}\text{tr}(\boldsymbol{z}_n\boldsymbol{z}_n^\top)\right\}$$ (Exercise: Verify)

Learning PPCA via EM

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

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

- 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

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

(Exercise: verify; can also be done "online")

- To compute $\mathbf{W}$, we need two posterior expectations $\mathbb{E}[\boldsymbol{z}_n]$ and $\mathbb{E}[\boldsymbol{z}_n\boldsymbol{z}_n^\top]$
- These can be easily obtained from the posterior $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$ computed in E step

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

- Note: The noise variance σ^2 can also be estimated (take deriv., set to zero..)

Summary: The Full EM Algorithm for PPCA

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

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

- **M step:** Re-estimate $\mathbf{W}$ and σ^2

$$\begin{aligned} \mathbf{W}_{new} &= \left[\sum_{n=1}^{N}\boldsymbol{x}_n\mathbb{E}[\boldsymbol{z}_n]^\top\right]\left[\sum_{n=1}^{N}\mathbb{E}[\boldsymbol{z}_n\boldsymbol{z}_n^\top]\right]^{-1} \\ \sigma^2_{new} &= \frac{1}{ND}\sum_{n=1}^{N}\left\{||\boldsymbol{x}_n||^2 - 2\mathbb{E}[\boldsymbol{z}_n]^\top\mathbf{W}_{new}^\top\boldsymbol{x}_n + \text{tr}\left(\mathbb{E}[\boldsymbol{z}_n\boldsymbol{z}_n^\top]\mathbf{W}_{new}^\top\mathbf{W}_{new}\right)\right\} \end{aligned}$$

- Set $\mathbf{W} = \mathbf{W}_{new}$ and $\sigma^2 = \sigma^2_{new}$. If not converged (monitor $p(\mathbf{X}|\Theta)$), 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)