LVMs for Dimensionality Reduction

CS771: Introduction to Machine Learning

Piyush Rai

Plan

- A latent variable model for dimensionality reduction
  - Probabilistic PCA
- Expectation maximization (EM) algorithm for MLE for PPCA

Probabilistic PCA (PPCA)

- Assume obs $\boldsymbol{x}_n \in \mathbb{R}^D$ as a linear mapping of a latent var $\boldsymbol{z}_n \in \mathbb{R}^K$ + Gaussian noise

$D \times 1$ offset

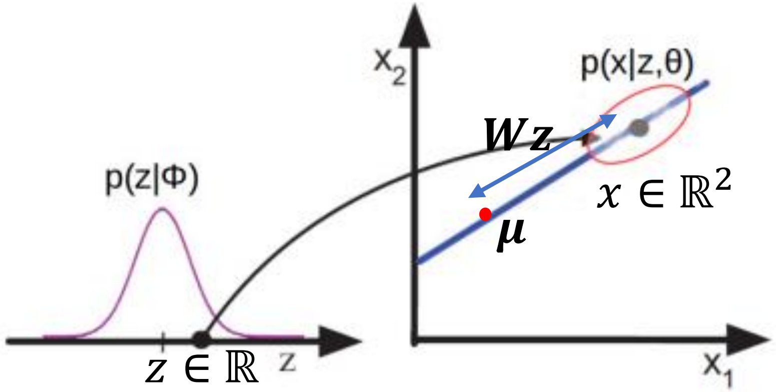
$D \times K$ matrix

$$\boldsymbol{x}_n = \boldsymbol{\mu} + \boldsymbol{W}\boldsymbol{z}_n + \boldsymbol{\epsilon}_n$$

Drawn from a zero-mean D-dim Gaussian $\mathcal{N}(\boldsymbol{0}, \sigma^2 I_D)$

- Equivalent to saying $p(\boldsymbol{x}_n | \boldsymbol{z}_n, \boldsymbol{\mu}, \boldsymbol{W}, \sigma^2) = \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{W}\boldsymbol{z}_n, \sigma^2 I_D)$
- Assume a zero-mean Gaussian prior on $\boldsymbol{z}_n$, so $p(\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{0}, I_K)$

A "reverse" (generative) way of thinking: first generate a low-dim latent variable z_n and then map it to generate the high-dim observation x_n



Need to estimate fewer parameters ($DK + D + 1$ as opposed to $O(D^2)$)

Thus PPCA does a low-rank approximation of the covariance matrix

- Joint distr. of $\boldsymbol{x}_n$ and $\boldsymbol{z}_n$ is Gaussian (since $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ and $p(\boldsymbol{z}_n)$ are individually Gaussian) and the marginal distribution of $\boldsymbol{x}_n$ will be Gaussian

$$p(\boldsymbol{x}_n | \boldsymbol{W}, \sigma^2) = N(\boldsymbol{x}_n | \boldsymbol{\mu}, \boldsymbol{W}\boldsymbol{W}^\top + \sigma^2 I_D)$$

As $\sigma^2 \to 0$, the covariance become approx low-rank (rank K) and only $DK + D + 1$ params needed, as opposed to $O(D^2)$ for the full covariance

Benefits of Generative Models for Dim-Red

- One benefit: Once model parameters are learned, we can even generate new data, e.g.,
  - Generate a random $\boldsymbol{z}_n$ from $\mathcal{N}(\mathbf{0}, I_K)$
  - Generate $\boldsymbol{x}_n$ condition on $\boldsymbol{z}_n$ from $\mathcal{N}(\boldsymbol{\mu} + \boldsymbol{W}\boldsymbol{z}_n, \sigma^2 I_D)$

(a) Training data

(b) Random samples

Generated using a more sophisticated generative model, not PPCA (but similar in formulation)

- Many other benefits. For example, can do dim-red, even if $\boldsymbol{x}_n$ has part of it as missing.

Learning PPCA using EM

- The ILL is $p(\boldsymbol{x}_n|\boldsymbol{\mu},\boldsymbol{W},\sigma^2) = N(\boldsymbol{x}_n\,|\boldsymbol{\mu},\boldsymbol{W}\boldsymbol{W}^\top + \sigma^2 I_D)$ with $\boldsymbol{z}_n$ integrated out
- Ignoring $\boldsymbol{\mu}$ for notational simplicity, ILL is $p(\boldsymbol{x}_n|\boldsymbol{W},\sigma^2) = N(\boldsymbol{x}_n\,|\boldsymbol{0},\boldsymbol{W}\boldsymbol{W}^\top + \sigma^2 I_D)$
- Can maximize ILL but requires solving eigen-decomposition (PRML: 12.2.1)
- EM will instead maximize expected CLL, with CLL given by

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

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

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

- Expected CLL will need $\mathbb{E}[\boldsymbol{z}_n]$ and $\mathbb{E}[\boldsymbol{z}_n\boldsymbol{z}_n^\top]$ where the expectations are w.r.t. the conditional posterior of $\boldsymbol{z}_n$

Learning PPCA using EM

Using $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ and $p(\boldsymbol{z}_n)$ and Gaussian reverse conditional property

Ignoring the μ parameter

- The EM algo for PPCA alternates between two steps
  - Compute conditional posterior of $\boldsymbol{z}_n$ given parameters $\Theta = (\mathbf{W}, \sigma^2)$

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

  - Maximize the expected CLL $\mathbb{E}[\log p(\boldsymbol{X}, \boldsymbol{Z}|\mathbf{W}, \sigma^2)]$ w.r.t. Θ

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

- Taking derivative of expected CLL w.r.t. $\mathbf{W}$ and setting to zero gives

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

Can likewise estimate σ^2 as well

- Required expectations can be found from the conditional posterior of $\boldsymbol{z}_n$

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

Full EM algo 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)

Other Generative Models for Dim-Red

- Factor Analysis is similar to PPCA except that the noise covariance of a diagonal matrix instead of $\sigma^2 I$

- Can use a mixture of probabilistic PCA for nonlinear dimensionality reduction
  - Data assumed to come from a mixture of low-rank Gaussians
  - Each low-rank Gaussian is a PPCA model
  - Basically does clustering + dimensionality reduction in each cluster

- Variational auto-encoders (VAE): z_n to x_n mapping is defined by deep neural net

Will look at VAE and GAN briefly when talking about deep learning

- Generative adversarial networks (GAN) are models that can only generate
  - Some variants of GANs (e.g., bi-directional GAN) can also be used to learn z_n from x_n

Coming up next

- Deep neural networks