Lab 8

EN.520.612.01.FA20 Machine Learning for Signal Processing

Generating Images using Probabilistic PCA

In this lab, we will implement a probabilistic version of PCA. We will also use the transformation matrix obtained by this method to generate never before seen samples of handwritten digits using the MNIST Dataset.

1. Understanding pPCA

PCA can also be expressed as the maximum likelihood solution of a probabilistic latent variable model. This reformulation of PCA, known as probabilistic PCA (pPCA), has several advantages compared with conventional PCA. To read more, look at Chapter 12 of Bishop's Pattern Recognition and Machine Learning.

The Probabilistic PCA (pPCA) model and corresponding EM-algorithm are defined as follows:

- Assuming centered data, given a D-dimensional data-point $\mathbf{x}_n$ in the training data as a column vector, we define

  $$p(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n|\mathbf{W}\mathbf{z}_n, \sigma^2\mathbf{I}_D) \tag{1}$$

  where $\mathbf{z}_n$ is a k-dimensional latent Gaussian variable with unit-diagonal covariance, there is D-dimensional noise with covariance $\sigma^2\mathbf{I}$. In mathematical terms, this means that $p(\mathbf{z}_n) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}_k)$.

- We can derive the EM algorithm for pPCA by following the general framework; we write down the complete-data log likelihood and take its expectation with respect to the posterior distribution of the latent distribution evaluated using old parameter values. Maximization of this expected complete-data log likelihood then yields the new parameter values. Because the data points are assumed independent, the complete-data log likelihood function for the entirety of the data $\mathbf{X}$ takes the form

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

  Using the fact that a D-dimensional Gaussian distribution is defined as

  $$\mathcal{N}(\mathbf{x}|\mu, \mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}}\exp\left(-0.5(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right) \tag{3}$$

  and remember we have zero-mean centered data ($\mu = \mathbf{0}$) and our covariance is $\mathbf{\Sigma} = \sigma^2\mathbf{I}_D$.

- Solving eqn. (2) by plugging in the closed-form of a Gaussian distribution, we obtain the Expectation-Step as follows -

  $$\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I} \tag{4}$$
  $$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}^{-1}\mathbf{W}^T\mathbf{x}_n \tag{5}$$
  $$\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T] = \sigma^2\mathbf{M}^{-1} + \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^T \tag{6}$$

here, we use the old $\mathbf{W}$ and σ values, assuming they're fixed for the E step.

- In the M-step, we keep the posterior statistics fixed, and maximize w.r.t $\mathbf{W}$ and σ, giving us

$$\mathbf{W}_{\text{new}} = \left[\sum_{n=1}^{N} \mathbf{x}_n \mathbb{E}[\mathbf{z}_n]^T\right] \left[\sum_{n=1}^{N} \mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T]\right]^{-1} \tag{7}$$

$$\sigma^2_{\text{new}} = \frac{1}{ND} \sum_{n=1}^{N} \{||\mathbf{x}_n||^2 - 2\mathbb{E}[\mathbf{z}_n]^T \mathbf{W}^T_{\text{new}} \mathbf{x}_n + \text{Tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] \mathbf{W}^T_{\text{new}} \mathbf{W}_{\text{new}})\} \tag{8}$$

2. Generating New Images

Now that we have the building blocks for probabilistic PCA, we can move on to something quite interesting that the formulation allows us; by randomly sampling $\mathbf{z}$s from a k-dimensional Gaussian distribution, we can generate *entirely new* images that we've never seen before. For this lab, we will use the MNIST Dataset that we've used before. As before, you can use the provided `loadMNISTImage.m` and `loadMNISTData.m` files to load in the data.
The algorithm to generate new samples is as follows

1. Define $\mathbf{X}^{(i)}$ as all training images that belong to class $i = 0, 1, \ldots, 9$. For example, all images belonging to class 0 would result in $\mathbf{X}^{(0)} \in \mathbb{R}^{784 \times 5923}$, as there are 5923 training images of digit 0.
2. We will perform the following steps separately for each of the 10 classes of images we have -
   (a) Center the input data $\mathbf{X}^{(\mathbf{i})}$.
   (b) Initialize the weights $\mathbf{W}^{(i)}$and $\sigma^{(i)}$s randomly.
   (c) Experiment with both $k = 50$ and $k = 100$.
   (d) Calculate the initial expected log-likelihood with randomly initialized weights.
   (e) Use the E-step to obtain the mean and variance of the hidden variables $\mathbf{z}_n$.
   (f) Use the M-step to obtain new values for $\mathbf{W}^{(i)}$ and $\sigma^{(i)}$.
   (g) Repeat steps (d), (e) and (f), until the log-likelihood converges. A good sanity check for your algorithm is ensuring log-likelihood increases in each step.
   (h) Once you've obtained your final $\mathbf{W}^{(i)}$ and $\sigma^{(i)}$, we can move on to generating new unseen images from class i.
   (i) Sample a point $\mathbf{z}_{\text{gen}}$ from your k-dimensional Gaussian distribution (in MATLAB, you can use `randn()`).
   (j) Your generated image is given by $\mathbf{x}_{\text{gen}} = \mathbf{W}^{(i)} \mathbf{z}_{\text{gen}}$.
   (k) For each of the 10 classes, generate 25 new images each, and plot them as a 5×5 grid. Note that once you've calculated $\mathbf{W}^{(i)}$ for one class, you can keep reusing it to generate new images by sampling new $\mathbf{z}_{\text{gen}}$. Generating images should be a very fast operation. Return your grids of generated images with your submission. Your code should output 10 5×5 grids of generated images.
   (l) Also note that you will have to add the means of each class back to the generated images for the images to look more understandable when you plot them.

Some Tips for faster computations

Deriving the linear algebra that goes into these computational tricks is beyond the scope of this class. However, feel free to use these in your code to speed up the EM Algorithm.

1. The inverse of matrix $\mathbf{M}$ can be calculated much faster by using the Cholesky decomposition of $\mathbf{M}$, therefore

$$\mathbf{U} = \text{chol}(\mathbf{M}) \tag{9}$$

$$\mathbf{V} = \text{inv}(\mathbf{U}) \tag{10}$$

$$\mathbf{M}^{-1} = \mathbf{V}\mathbf{V}^T \tag{11}$$

2. Instead of calculating the log-likelihood at each step, we're going to approximate the log-likelihood so that it is much faster to calculate, and we don't need to invert the covariance matrix over and over again. The approximate log-likelihood for the system at any step is given by

$$\log p(\mathbf{X}|\mathbf{W}, \sigma^2) = -\frac{N}{2}\left[D\log(2\pi) + \log|\mathbf{M}| + \text{Tr}(S\mathbf{M}^{-1})\right] \tag{12}$$

where the sample covariance S is calculated as

$$S = \sum_{n=1}^{N} ||\mathbf{x}_n||^2 \tag{13}$$

For the other two terms, first, let's define the following

$$\mathbf{T} = (\mathbf{U}^T)^{-1}(\mathbf{W}^T\mathbf{X}) \tag{14}$$

and therefore

$$\text{Tr}(S\mathbf{M}^{-1}) = \frac{S - \sum_{i,j}|\mathbf{T_{ij}}|^2}{N\sigma^2} \tag{15}$$

$$\text{and } \log|\mathbf{M}| = 2\ \text{sum}(\log(\text{diag}(\mathbf{U}))) + (D-k)\log(\sigma^2) \tag{16}$$

Python:
https://colab.research.google.com/drive/11nsrrqIXb1GdEG0ifuA3ijGItLJVK0gn?usp=sharing