Unsupervised Learning 2001

Lecture 2: Latent Variable Models

Zoubin Ghahramani  
zoubin@gatsby.ucl.ac.uk

Carl Edward Rasmussen  
edward@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit  
MSc Intelligent Systems, Computer Science
The Gaussian Model (review)

\[ P(y|\mu, \Sigma) = |2\pi \Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}(y - \mu)^\top \Sigma^{-1} (y - \mu) \right\} \]

Data set \( Y = \{y_1, \ldots, y_N\} \); Likelihood is \( P(\text{data}|\text{model}) \): \( P(Y|\mu, \Sigma) = \prod_{n=1}^{N} P(y_n|\mu, \Sigma) \)

Goal: find \( \mu \) and \( \Sigma \) that maximise log likelihood:

\[ \mathcal{L} = \log \prod_{n=1}^{N} P(y_n|\mu, \Sigma) = -\frac{N}{2} \log |2\pi \Sigma| - \frac{1}{2} \sum_{n} (y_n - \mu)^\top \Sigma^{-1} (y_n - \mu) \]

Note: equivalently, minimise \(-\mathcal{L}\), which is quadratic in \( \mu \)

Procedure: take derivatives and set to zero:

\[ \frac{\partial \mathcal{L}}{\partial \mu} = 0 \quad \Rightarrow \quad \hat{\mu} = \frac{1}{N} \sum_{n} y_n \quad (\text{sample mean}) \]

\[ \frac{\partial \mathcal{L}}{\partial \Sigma} = 0 \quad \Rightarrow \quad \hat{\Sigma} = \frac{1}{N} \sum_{n} (y_n - \hat{\mu})(y_n - \hat{\mu})^\top \quad (\text{sample covariance}) \]
Three Limitations of Gaussians

- What about higher order statistical structure in the data? \[\Rightarrow\) nonlinear and hierarchical models
- What happens if there are outliers? \[\Rightarrow\) other noise models
- There are \(D(D + 1)/2\) parameters in the multi-variate Gaussian model. What if \(D\) is very large? \[\Rightarrow\) dimensionality reduction
Factor Analysis

Linear generative model: \[ y_d = \sum_{k=1}^{K} \Lambda_{dk} x_k + \epsilon_d \]
- \( x_k \) are independent \( \mathcal{N}(0, 1) \) Gaussian factors
- \( \epsilon_d \) are independent \( \mathcal{N}(0, \Psi_{dd}) \) Gaussian noise
- \( K < D \)

So, \( y \) is Gaussian with: 
\[ P(y) = \int P(x)P(y|x)dx = \mathcal{N}(0, \Lambda\Lambda^T + \Psi) \]
where \( \Lambda \) is a \( D \times K \) matrix, and \( \Psi \) is diagonal.

**Dimensionality Reduction:** Finds a low-dimensional projection of high dimensional data that captures most of the correlation structure of the data.
ML learning finds $\Lambda$ and $\Psi$ given data

parameters (corrected for symmetries):

$$DK + D - \frac{K(K - 1)}{2} < \frac{D(D + 1)}{2}$$

no closed form solution for ML params: $\mathcal{N}(0, \Lambda\Lambda^\top + \Psi)$

[Bayesian treatment would also have priors over $\Lambda$ and $\Psi$ and would average over them for prediction.]
Latent Variable Models

Explain correlations in $y$ by assuming some latent variables $x$

(e.g. objects, illumination, pose)

(e.g. object parts, surfaces)

(e.g. edges)

(retinal image, i.e. pixels)
Coding Interpretation of Factor Analysis: Coding Under a Gaussian

Remember, from Shannon’s source coding theorem:

\[ l(x) = -\log P(x) \approx -\log[p(x)\Delta] = -\log p(x) - \log \Delta \]

\[ = \frac{(x - \mu)^2}{2\sigma^2} + \frac{1}{2} \log 2\pi + \log \sigma - \log \Delta \]

Note as \( \Delta \to 0 \) then \( l(x) \to \infty \).
Coding Interpretation of Factor Analysis

Multivariate: \[ l(y) = \frac{1}{2} \sum_d \frac{(y_d - \mu_d)^2}{\sigma_d^2} + \frac{D}{2} \log 2\pi + \sum_d \log \sigma_d - D \log \Delta \]

Alternative, two-stage code...

First code the \( K \) factors: \[ l(x) = \frac{1}{2} \sum_k x_k^2 + \frac{K}{2} \log 2\pi - K \log \Delta \]

Then code the data given the factors:

\[ l(y|x) = \frac{1}{2} \sum_d \left( y_d - \sum_k \Lambda_{dk} x_k \right)^2 \frac{1}{\Psi_d^2} + \frac{D}{2} \log 2\pi + \sum_d \log \Psi_d - D \log \Delta \]

How should we choose the \( x \) ?
Principal Components Analysis

Noise variable becomes infinitesimal compared to the scale of the data: \( \Psi = \lim_{\epsilon \to 0} \epsilon I \)

Equivalently: reconstruction cost becomes infinite compared to the cost of coding the hidden units under the prior.

\[
P(x|y) = \mathcal{N}(\beta y, I - \beta \Lambda)
\]

\[
\beta = \lim_{\epsilon \to 0} \Lambda^T (\Lambda \Lambda^T + \epsilon I)^{-1} = (\Lambda^T \Lambda)^{-1} \Lambda^T
\]
Eigenvalues and Eigenvectors

$\lambda$ is an eigenvalue and $x$ is an eigenvector of $A$ if:

$$Ax = \lambda x$$

and $x$ is a unit vector ($x^T x = 1$).

**Interpretation:** the operation of $A$ in direction $x$ is a scaling by $\lambda$.

The $K$ Principal Components are the $K$ eigenvectors with the largest eigenvalues of the data covariance matrix (i.e. $K$ directions with the largest variance).

**Note:** $\Sigma$ can be decomposed:

$$\Sigma = USU^T$$

where $S$ is diag($\sigma_1^2, \ldots, \sigma_D^2$) and $U$ is a an orthonormal matrix.
Coding Interpretation of PCA

First code the $K$ factors:

$$l(x) = \frac{1}{2} \sum_k x_k^2 + \frac{K}{2} \log 2\pi - K \log \Delta$$

Then code the data given the factors:

$$l(y|x) = \frac{1}{2} \sum_d \frac{(y_d - \sum_k \Lambda_{dk} x_k)^2}{\sigma^2} + \frac{D}{2} \log 2\pi + D \log \sigma - D \log \Delta$$

Since $\sigma \to 0$ the cost of coding the factors is negligible compared to the cost of coding the data.
Mutual Information and PCA

Problem: Given $y$, find $x = Ay$ with columns of $A$ unit vectors, s.t. $I(x; y)$ is maximised (assuming that $P(x, y)$ is jointly Gaussian).

$$I(x; y) = H(x) + H(y) - H(x, y) = H(x)$$

So we want to maximize the entropy of $x$. What is the entropy of a Gaussian?

$$H(z) = -\int dz \ P(z) \ln P(z) = \frac{1}{2} \ln |\Sigma| + \frac{D}{2}(1 + \ln 2\pi)$$

Therefore we want the distribution of $x$ to have largest volume (i.e. det of covariance matrix).

$$\Sigma_x = A\Sigma_y A^\top = AUS_y U^\top A^\top$$

So, $A$ should be aligned with the columns of $U$ which are associated with the largest eigenvalues (variances).
FA vs PCA

- PCA is rotationally invariant; FA is not
- FA is noise scale invariant; PCA is not
- FA defines a probabilistic model; PCA does not
Probabilistic PCA

Linear generative model: \( y_d = \sum_{k=1}^{K} \Lambda_{dk} x_k + \epsilon_d \)

- \( x_k \) are independent \( \mathcal{N}(0, 1) \) Gaussian factors
- \( \epsilon_d \) are independent \( \mathcal{N}(0, \sigma^2) \) Gaussian noise
- \( K < D \)

I.e. factor analysis with \( \Psi = \sigma^2 I \)

Finds the same principal subspace as PCA but provides a well-defined probabilistic model.
Network Interpretations
and Encoder-Decoder Duality

![Diagram of network interpretations and encoder-decoder duality.](image)
From Supervised Learning to PCA

A linear autoencoder neural network trained to minimise squared error learns to perform PCA (Baldi & Hornik, 1989).
Gradient Methods of Learning FA

Write down negative log likelihood:

\[
\frac{1}{2} \log |\pi (\Lambda \Lambda^\top + \Psi)| + \frac{1}{2} y^\top (\Lambda \Lambda^\top + \Psi)^{-1} y
\]

Optimize w.r.t. \( \Lambda \) and \( \Psi \) (need matrix calculus) subject to constraints

We will soon see an easier way to learn latent variable models...
Limitations of Gaussian, FA and PCA models

- Gaussian, FA and PCA models are easy to understand and use in practise.

- They are a convenient way to find interesting directions in very high dimensional data sets, eg as preprocessing.

- Their problem is that they make very strong assumptions about the distribution of the data, only the mean and variance of the data are taken into account.

The class of densities which can be modelled is too restrictive.

By using mixtures of simple distributions, such as Gaussians, we can expand the class of densities greatly.
Mixtures of Gaussians – MoG

Use probabilistic mixtures of simple (eg Gaussian) density models.

Some examples where non-Gaussian densities are modelled (approximated) as a mixture of Gaussians. The red curves show the (weighted) Gaussians, and the blue curve the resulting density.

The advantage of this mixture approach, is that given enough mixture components we can model (almost) any density (as accurately as desired), but we still only need to work with the well known Gaussian form.
The MoG likelihood

Here a set of $k$ Gaussians, each with a separate mean, $\mu_i$, and covariance $\Sigma_i$ are weighted together with (non-negative) weights $\pi_i$, with the normalising condition:

\[ p_i \geq 0, \text{ and } \sum_{i=1}^{k} \pi_i = 1. \]

The probability of an observation $y^{(c)}$ under mixture component number $i$ is Gaussian:

\[ p(y^{(c)}|\mu_i, \Sigma_i) = |2\pi \Sigma_i|^{-1/2} \exp \left( -\frac{1}{2} (y^{(c)} - \mu_i)^\top \Sigma_i^{-1} (y^{(c)} - \mu_i) \right). \]

The probability of an observation $y^{(c)}$ under the entire mixture model is a weighted sum of Gaussian densities:

\[
p(y^{(c)}|\mu, \Sigma, \pi) = \sum_{i=1}^{k} \pi_i p(y^{(c)}|\mu_i, \Sigma_i)
= \sum_{i=1}^{k} \pi_i |2\pi \Sigma_i|^{-1/2} \exp \left( -\frac{1}{2} (y^{(c)} - \mu_i)^\top \Sigma_i^{-1} (y^{(c)} - \mu_i) \right),
\]

(1)
The MoG likelihood, continued

The probability of a set of \( n \) observations, \( y = \{y^{(1)}, \ldots, y^{(n)}\} \) (the likelihood):

\[
p(y|\mu, \Sigma, \pi) = \prod_{c=1}^{n} \sum_{i=1}^{k} \pi_i p(y^{(c)}|\mu_i, \Sigma_i)
\]

\[
= \prod_{c=1}^{n} \sum_{i=1}^{k} \pi_i |2\pi \Sigma_i|^{-1/2} \exp \left( -\frac{1}{2}(y^{(c)} - \mu_i)^\top \Sigma_i^{-1}(y^{(c)} - \mu_i) \right).
\]

Here, the observations are thought of as being generated *independently* from the mixture (given the parameters).

The log of the likelihood is:

\[
\log p(y|\mu, \Sigma, \pi) = \log \prod_{c=1}^{n} \sum_{i=1}^{k} \pi_i p(y^{(c)}|\mu_i, \Sigma_i) = \sum_{c=1}^{n} \log \sum_{i=1}^{k} \pi_i p(y^{(c)}|\mu_i, \Sigma_i)
\]

\[
= \sum_{c=1}^{n} \log \sum_{i=1}^{k} \pi_i (2\pi \Sigma_i)^{-1/2} \exp \left( -\frac{1}{2}(y^{(c)} - \mu_i)^\top \Sigma_i^{-1}(y^{(c)} - \mu_i) \right).
\]
Maximum likelihood (ML) training a MoG model

The log likelihood is: \[ \mathcal{L} = \sum_{c=1}^{n} \log \sum_{i=1}^{k} \pi_i p(y^{(c)} | \mu_i, \Sigma_i) \]

Its partial derivative wrt \( \theta_i = \{ \mu_i, \Sigma_i \} \) is

\[
\frac{\partial \log p(y | \pi, \mu, \Sigma)}{\partial \theta_i} = \sum_{c=1}^{n} \frac{\pi_i}{\sum_{j=1}^{k} \pi_j p(y^{(c)} | \mu_j, \Sigma_j)} \frac{\partial p(y^{(c)} | \mu_i, \Sigma_i)}{\partial \theta_i}
\]

Using the identity \( \partial p/\partial \theta = p \times \partial \log p/\partial \theta \), it can be re-written as:

\[
\frac{\partial \log p(y | \pi, \mu, \Sigma)}{\partial \theta_i} = \sum_{c=1}^{n} r_i^{(c)} \frac{\partial \log p(y^{(c)} | \mu_i, \Sigma_i)}{\partial \theta_i},
\]

where we have defined the \textit{responsibilities} of component \( i \) for data point \( c \) as:

\[
r_i^{(c)} = \frac{\pi_i p(y^{(c)} | \mu_i, \Sigma_i)}{\sum_{j=1}^{k} \pi_j p(y^{(c)} | \mu_j, \Sigma_j)}.
\]
Derivatives of log likelihood

For the means we get:

\[
\frac{\partial \log p(y|\pi, \mu, \Sigma)}{\partial \mu_i} = \sum_{c=1}^{n} r_i^{(c)} \Sigma_i^{-1} (y^{(c)} - \mu_i)
\]

and the the precisions (inverse variances):

\[
\frac{\partial \log p(y|\pi, \mu, \Sigma)}{\partial S_i} = \frac{1}{2} \sum_{c=1}^{n} r_i^{(c)} (\Sigma_i - (y^{(c)} - \mu_i)(y^{(c)} - \mu_i)^\top).
\]

Finally, the partial derivative wrt the mixing proportions is:

\[
\frac{\partial \log p(y|\pi, \mu, \Sigma)}{\partial \pi_i} = \sum_{c=1}^{n} \frac{p(y^{(c)}|\mu_i, \Sigma_i)}{\sum_{j=1}^{k} \pi_j p(y^{(c)}|\mu_j, \Sigma_j)}
\]

These equations together can be used for gradient based learning; eg taking small steps in the direction of the gradient (or using conjugate gradients).
The k-means algorithm

Assume for simplicity, that \( \pi_i = 1/k \); assume further, that \( \Sigma_i = Iz \), where \( z \to 0 \). Then the responsibilities become discrete:

\[
    r_i^{(c)} = \delta(i, \arg \max_j p(y^{(c)}|\mu_j, \Sigma_j)),
\]

being 1 for the most likely component and 0 otherwise. We can then solve directly for the means \( \mu_i \) by setting eq. (5) to zero, resulting in \( \mu_i \) being equal to the mean of the data points associated with it.

The above iterative algorithm is called k-means; it usually converges in a few iterations and it has the advantage over the gradient based method that there is no learning rate.

However, the assumptions we made are quite serious.
The EM algorithm

One could to find a procedure which jumps in parameter space, without quite as severe restrictions as k-means.

One idea is to neglect the dependency of the responsibilities on the parameters when calculating derivatives. If we do that, we can explicitly solve for the parameters, given the responsibilities.

We get the following two step algorithm:

- Evaluate the responsibilities (given the parameters), using eq. (4).

- Optimize the parameters (given the responsibilities)

\[
\mu_i = \frac{\sum_{c=1}^{n} r_i^{(c)} y^{(c)}}{\sum_{c=1}^{n} r_i^{(c)}}, \quad \text{and} \quad \Sigma_i = \frac{\sum_{c=1}^{n} r_i^{(c)} (y^{(c)} - \mu_i)(y^{(c)} - \mu_i)^\top}{\sum_{c=1}^{n} r_i^{(c)}}.
\]

In fact, we will show in the next lecture, that this procedure is guaranteed not to decrease the likelihood in each iteration. The algorithm is called the EM algorithm.
Problems

There are several problems with the new algorithms:

• slow convergence for the gradient based method

• gradient based method may develop invalid covariance matrices

• local minima; the end configuration may depend on the starting state

• how do you adjust $k$? Using the likelihood alone is no good.

• singularities; components with a single data point will have their covariance going to zero and the likelihood will tend to infinity.
Readings


