Principal Components Analysis MLE, EM and MAP

CMSC 828D

Fundamentals of Computer Vision

Fall 2000

Outline

- Lagrange Multipliers
- Principal Components Analysis
- Review of parameter estimation.
- Notation and Problem Definition
- Maximum Likelihood Estimation
- Difficulties
- Bayesian view
- Maximum A Posteriori Estimation
- Algorithms: Expectation Maximization

Lagrange Multipliers

- Find stationary points of a function $f(\mathbf{x})$ subject to one or more constraints $g(\mathbf{x}) = 0$
- Consider the surface $g(\mathbf{x})=0$
  - The direction of increase of f is ∇f
  - However moving this direction may take us away from the constraint surface
  - **Idea**: move along component of ∇f *along the surface.*
  - Denote this component as $\nabla_g f$
  - At the extremum point this function will be stationary

$$\nabla_g f = 0$$

  - How to get $\nabla_g f$?
  - Take ∇f and subtract from it that part $\mathbf{a}$ which takes it out of the surface g

$$\nabla_g f = \nabla f - \mathbf{a}$$

Finding the component of ∇f along g

- Now let us move by a distance δ along the surface **g**
  - $g(\mathbf{x}+\boldsymbol{\delta})=g(\mathbf{x})+ (\boldsymbol{\delta} \cdot \nabla \mathbf{g})$
  - But this still lies on the surface -- so $g(\mathbf{x}+\boldsymbol{\delta})=0$
  - So $\boldsymbol{\delta} \cdot \nabla g=0$
  - $\Rightarrow$ ∇g is perpendicular to motions along the surface
- But we wanted to remove any piece of ∇f that was perpendicular to $g(\mathbf{x})=0$
- This will be a vector of the form

$$\nabla_g f = \nabla f + \lambda \nabla g$$

(For some λ)

Lagrangian

- Consider the Lagrangian function

$$L(\mathbf{x}, \lambda) = f + \lambda g$$

$$\frac{\partial}{\partial \mathbf{x}} L(\mathbf{x}, \lambda) = \nabla f + \lambda \nabla g, \qquad \frac{\partial}{\partial \lambda} L(\mathbf{x}, \lambda) = g$$

- Extremize the Lagrangian

$$\frac{\partial}{\partial \mathbf{x}} L(\mathbf{x}, \lambda) = \nabla f + \lambda \nabla g = 0, \qquad \frac{\partial}{\partial \lambda} L(\mathbf{x}, \lambda) = g(\mathbf{x}) = 0$$

- So this gives us **both the constraint equation** and **the way to optimize the function on the surface**.

Principal Components Analysis

Key technique in dealing with data

- Data Reduction
  - Experimental measurements produce lots of data
  - Scientists postulate lots of hypotheses as to what factors affect data. Create overly complex models
  - Goal: find factors that affect data most and create small models
- Knowledge discovery
  - Collect lots of data
  - Are there patterns hidden in the collected data that can help us develop a model and understanding?
  - Can we use this understanding to classify a new piece of data?
- Applications: Almost all computer vision
  - Especially face recognition, tracking, pattern recognition… etc.

Basics

- Record data
- d dimensional data vector $\mathbf{x}$
- Record N observations
- Mean $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$
- Covariance $\mathbf{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})'(\mathbf{x}_i - \bar{\mathbf{x}})$
- Problem: d can be very large
  - *"megapixel camera"* d>1 million (values of the intensity at the pixels)
  - Image is a point in a d dimensional space
- Need a way to capture the information in the data but using very few "coordinates"

Principal Components Analysis

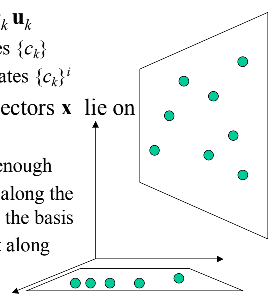
- Consider a vector $\mathbf{x}$ that lies in a d dimensional linear space.
- Let vectors $\mathbf{u}_k$, $k=1,\ldots,d$ define a basis in the space

$$\mathbf{x}=\sum c_k \mathbf{u}_k$$

$\mathbf{x}$ is characterized by d coordinates $\{c_k\}$

Different $\mathbf{x}_i$ have different coordinates $\{c_k\}^i$

- Now consider a case that the vectors $\mathbf{x}$ lie on a lower dimensional manifold
  - Smaller number of coordinates enough
  - For small d, if points are spread along the axes it may be easy to recognize the basis
  - For larger d and if points are not along axes it is harder
  - Need mathematical tools

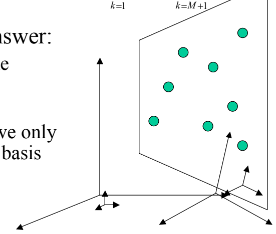


Dimension Reduction

- Expressing the points using the basis vectors along the axes, we still need all the coordinates to describe the $\mathbf{x}_i$
- However if we had an alternate basis we need only two variables and a constant to describe the points.
- Complexity of most algorithms is a power of d

$$\mathbf{x} = \sum_{k=1}^{M} c_k\mathbf{u}_k + \sum_{k=M+1}^{d} b_k\mathbf{u}_k$$

- Mathematical questions to answer:
  - **Best Basis**: How to find out the basis that is best lined up with the data?
  - **Approximation question**: If we only wanted the best k dimensional basis how do we select it?
  - How do we account for noise?



Approximation

- Given a dataset $\{\mathbf{x}^i\}$ with N members
- Write each vector in a basis $\{\mathbf{u}_k\}$
- Coefficients $c_k^i = \mathbf{x}'^i\mathbf{u}_k$
- Approximate each $\mathbf{x}^i$ as sum of a variable part and a constant part and
- Dimension of variable part is M $\mathbf{x}^i = \sum_{k=1}^{d} c_k^i\mathbf{u}_k \simeq \sum_{k=1}^{M} c_k^i\mathbf{u}_k + \sum_{k=M+1}^{d} b_k\mathbf{u}_k$
- Error in approximting a particular vector

$$\boldsymbol{\varepsilon}^i = \mathbf{x}^i - \sum_{k=1}^{M} c_k^i\mathbf{u}_k - \sum_{k=M+1}^{d} b_k\mathbf{u}_k = \mathbf{x}^i - \sum_{k=1}^{M} c_k^i\mathbf{u}_k - \sum_{k=M+1}^{d} b_k\mathbf{u}_k = \sum_{k=M+1}^{d}(c_k^i - b_k)\mathbf{u}_k$$

- Define sum of squares error function C

$$C(b_k) = \sum_{i=1}^{N}\left[\sum_{k=M+1}^{d}(c_k^i - b_k)\mathbf{u}_k\right]^2$$

Getting the parameters b_k and $\mathbf{u}_k$

- Evaluate b_k by setting $\partial C/\partial b_k=0$

$$b_k = \frac{1}{N}\sum_{i=1}^{N} c_k^i = \mathbf{u}'_k\,\bar{\mathbf{x}}$$

- To get best basis vectors $\mathbf{u}_k$ define cost function

$$\begin{aligned} E_M &= \sum_{i=1}^{N}\sum_{k=M+1}^{d}\left[(\mathbf{x}^i - \bar{\mathbf{x}})\cdot\mathbf{u}_k\right]^2 \\ &= \sum_{k=M+1}^{d}\mathbf{u}_k'\left[\sum_{i=1}^{N}\left[(\mathbf{x}^i - \bar{\mathbf{x}})(\mathbf{x}^i - \bar{\mathbf{x}})'\right]\right]\mathbf{u}_k \\ &= \sum_{k=M+1}^{d}\mathbf{u}_k'\mathbf{\Sigma}\mathbf{u}_k \end{aligned}$$

- Minimize E with respect to $\mathbf{u}_k$
- However the expression is homogeneous in $\mathbf{u}_k$
  - Obvious solution is $\mathbf{u}_k=0$

Finding the best basis

- To avoid the trivial solution we need a constraint
- Basis vectors have unit magnitude $||\mathbf{u}_k||=1$, $\mathbf{u}_j \cdot \mathbf{u}_k=\delta_{jk}$
- How do we optimize subject to constraints?
  - Lagrange Multipliers! Cost function with constraints:

$$E_M = \sum_{k=M+1}^{d} \mathbf{u}_k{}'\boldsymbol{\Sigma}\mathbf{u}_k - \sum_{j=M+1}^{N}\sum_{k=M+1}^{N} \mu_{jk}\left(\mathbf{u}_j{}'\mathbf{u}_k - \delta_{jk}\right)$$

Can be written in the form:

$$E_M = Tr\{\mathbf{U}'\boldsymbol{\Sigma}\mathbf{U}\} - Tr\{\mathbf{M}(\mathbf{U}'\mathbf{U}-\mathbf{I})\}$$

$$\mathbf{U} = [\mathbf{u}_{M+1} | \mathbf{u}_{M+2} | \cdots | \mathbf{u}_d] \qquad \mathbf{M} = [\mu_{jk}]$$

- Minimizing with respect to $\mathbf{u}_k$

$$(\boldsymbol{\Sigma}+\boldsymbol{\Sigma}')\mathbf{U} - \mathbf{U}(\mathbf{M}+\mathbf{M}') = 0 \quad \Rightarrow \quad \boldsymbol{\Sigma}\mathbf{U} = \mathbf{U}\mathbf{M}$$

- **U** is an orthonormal vector with columns as basis vectors
- So any set of **U***s* and **M***s* that satisfy $\mathbf{U}^t\boldsymbol{\Sigma}\mathbf{U} = \mathbf{M}$

PCA

- Choose the simplest solution
  - **U** vectors in the eigenbasis of $\boldsymbol{\Sigma}$
  - **M** is the diagonal matrix of eigenvalues.
- Algorithm
  1. Compute the mean of the data

$$\mathbf{x}^{-} = (\textstyle\sum_i \mathbf{x}_i)/N$$

  2. Compute the covariance of the data,

$$\boldsymbol{\Sigma} = \textstyle\sum_i (\mathbf{x}^i - \mathbf{x}^{-})(\mathbf{x}^i - \mathbf{x}^{-})'$$

  3. Compute eigenvectors, $\mathbf{u}_i$ and corresponding eigenvalues λ_i of $\boldsymbol{\Sigma}$ sorted according to the magnitude of λ_i
  4. For a desired approximation dimension M, $\mathbf{x}^i$ can be written as

$$\mathbf{x}^i \simeq \sum_{k=1}^{M} c_k^i \mathbf{u}_k + \sum_{k=M+1}^{d} \overline{\mathbf{x}}_k$$

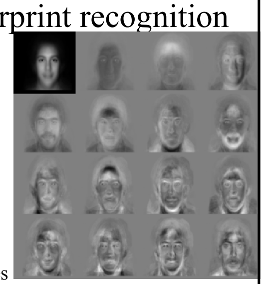
Selecting the approximation dimension *M*?

- The proportion of variance in the data captured when we truncate at a given M is

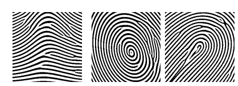
$$\text{Proportion of variance captured} = \frac{\sum_{i=1}^{M}\lambda_i}{\sum_{i=1}^{d}\lambda_i}$$

- Two strategies:
  - 1st: Specify the desired threshold e.g. 99%
  - 2nd: Look at the magnitudes of $\lambda_i / \lambda_{i+1}$
    - In some problems it will exhibit a sharp value at some value of i
    - *"Intrinsic dimension"* of the problem

Application: Face/fingerprint recognition

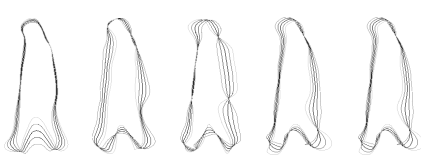


- 128 faces at 64x64 resolution for training
  - $d = 4096$
  - Perform PCA choosing 1st 20 modes (16 shown beside)
  - Approximate new faces using these
  - Greater than 95% accuracy claimed on a database of 7000 faces
- Also used for fingerprint storage and recognition
- If interested check http://c3iwww.epfl.ch/projects_activities/jmv/fingerprint_identification.html



Pedestrian shapes from PCA modes

- Problem: track moving pedestrians from a moving camera.
- Solution: generate PCA modes ("eigenvectors") from Pedestrian shapes
- Track pedestrian shapes in new images by searching for variations in PCA modes



Movie

- From Philomin et al 2000

Summary

Principal Components Analysis (PCA) exploits the redundancy in multivariate data. Allows one to:

- Determine (relationships) in the variables
- Reduce the dimensionality of the data set without a significant loss of information

Parameter Estimation
MLE and MAP

Problem Introduction

- Model characterized by values of a parameter vector $\boldsymbol{\theta}$
- Have several observations of a process that we think follows this model
- Using this observation set as "training data" we want to find the most probable values of the parameters
- Observations have errors and are assumed to follow a probability distribution
- **Two Approaches:**
  - ***Maximum Likelihood Estimation (MLE)***
    - ***Expectation Maximization Algorithm***
  - ***Maximum A Priori Estimation (MAP)***
    - ***"Bayesian approach"***
- Talk will only touch on a vast field, but hopefully will make you familiar with the jargon.

Notation

- parameter vector being estimated $\boldsymbol{\theta}$
- **a** test value to be compared
- *E.g., if* $N(\mu, \sigma)$ 1-D normal distribution

$$N(\mu,\sigma)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

•Parameters to be estimated μ, σ

•d dimensional data with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$

$$N(\boldsymbol{\mu},\boldsymbol{\Sigma})=\frac{1}{\sqrt{(2\pi)^d\,|\boldsymbol{\Sigma}|}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

•Parameters to be estimated $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$

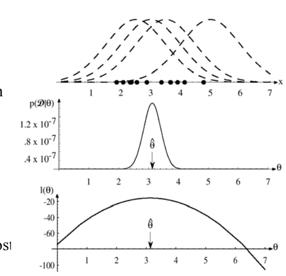
•Data set on which the estimation is based $\mathcal{D}$

Maximum Likelihood Estimation

- Use a set of N data points $\mathbf{x}^i$ belonging to a training set $\mathcal{D}$, and assumed to be drawn independently from the probability density $p(\mathbf{x}|\boldsymbol{\theta})$ to estimate $\boldsymbol{\theta}$
- Because observations are independent $p(\mathcal{D}|\theta)=\prod_{i=1}^{N} p(\mathbf{x}_i\,|\,\theta)$
- Likelihood of $\boldsymbol{\theta}$ with respect to the samples in $\mathcal{D}$, is $p(\mathcal{D}|\boldsymbol{\theta})$
  - probability that the set of observations in the dataset would have occurred, given the parameters $\boldsymbol{\theta}$
- Maximum likelihood estimate, $\boldsymbol{\theta}$^ is the value of $\boldsymbol{\theta}$ that maximizes this probability.
- Estimation problem: treat $p(\mathcal{D}|\boldsymbol{\theta})$ as a function of $\boldsymbol{\theta}$ and find value that maximizes it.

Log Likelihood Function

- Probabilities are positive.
- Logarithm is a monotonic increasing function
- So, maxima of the likelihood function will occur at the same values as its logarithm
- Easier to work with
  - Converts products to sums
  - Shrinks big numbers and small numbers to *O(1)*
  - Easier to differentiate resulting cost function
- Denoted $l(\boldsymbol{\theta})$



$$l(\theta)=\ln p(\mathcal{D}|\theta)=\sum^{N}\ln p(x^k\,|\,\theta)$$

$$\hat{\theta}=\arg\max_{\theta} l(\theta)$$

$$\nabla_\theta \ln p(\mathcal{D}|\theta)=\sum^{N}\nabla_\theta \ln p(x^k\,|\,\theta)=0$$

Estimate can be a local minimum or a global minimum

Maximum Likelihood Estimation

- Summary
  - Given a dataset whose elements are assumed to be distributed according to a probability distribution $p(\mathbf{x}|\boldsymbol{\theta})$
  - Create the likelihood function for the data set that shows the probability that the data set could have come out of the assumed probability distribution with given parameters $\boldsymbol{\theta}$.
  - If observations in the dataset are independent the likelihood function is $p(\mathcal{D}|\theta) = \prod_{i=1}^{N} p(\mathbf{x}_i|\theta)$
  - Using the log of the likelihood function is often more convenient.
  - Parameter estimated by maximizing the likelihood or the log with respect to $\boldsymbol{\theta}$

Expectation Maximization

- Algorithm for approximate maximum likelihood parameter estimation when features are missing
- Situation:
  - Given a set of N data points $\mathbf{x}^i$ belonging to a training set Δ
  - Data is d dimensional
  - Some of the data points is missing features, or has poorly measurec values
  - Good data point $\mathbf{x}_g = \{x_1, x_2, \ldots, x_N\}$
  - Bad data point $\mathbf{x}_b = \{x_1, x_2, \ldots, x_k, \ldots, x_N\}$
- Separate features into a good set $\mathcal{D}_g$ and a bad set $\mathcal{D}_b$
- Using a guess $\boldsymbol{\theta}$, fix some of the parameters, and form a likelihood function over the unknown features

$$Q(\theta;\theta^i) = \varepsilon\left[\ln p\left(\mathcal{D}_g, \mathcal{D}_b; \theta \mid \mathcal{D}_g; \theta^i\right)\right]$$

Maximize Q with respect to the unfixed values.
Fix the found values
Repeat for the previously fixed values

Algorithm 1 (Expectation-Maximization)

$$
\begin{array}{l}
\underline{\textbf{begin}}\ \underline{\textbf{initialize}}\ \boldsymbol{\theta}^0, T, i = 0 \\
\quad\quad \underline{\textbf{do}}\ i \leftarrow i + 1 \\
\quad\quad\quad \textbf{E step} : \text{compute } Q(\boldsymbol{\theta};\ \boldsymbol{\theta}^i) \\
\quad\quad\quad \textbf{M step} : \boldsymbol{\theta}^{i+1} \leftarrow \arg\max_{\theta} Q(\boldsymbol{\theta};\ \boldsymbol{\theta}^i) \\
\quad\quad \underline{\textbf{until}}\ Q(\boldsymbol{\theta}^{i+1};\ \boldsymbol{\theta}^i) - Q(\boldsymbol{\theta}^i;\ \boldsymbol{\theta}^{i-1}) \le T \\
\quad \underline{\textbf{return}}\ \hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}^{i+1} \\
\underline{\textbf{end}}
\end{array}
$$

- Sometimes we prefer to apply the EM, even when there are no missing features
- Q may be simpler to optimize
- Get an approx. MLE solution

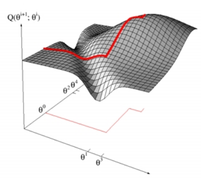


Figure 3.5: The search for the best model via the EM algorithm starts with some initial value of the model parameters, θ^0. Then, via the **M step** the optimal θ^1 is found. Next, θ^1 is held constant and the value θ^2 found which optimizes $Q(\cdot,\cdot)$. This process iterates until no value of θ can be found that will increase $Q(\cdot,\cdot)$. Note in particular that this is different from a gradient search. For example here θ^1 is the global optimum (given fixed θ^0), and would not necessarily have been found via gradient search. (In this illustration, $Q(\cdot,\cdot)$ is shown symmetric in its arguments; this need not be the case in general, however.)

Maximum A Posteriori Estimation

- In MLE the estimated value of the parameter vector $\hat{\boldsymbol{\theta}}$ is not taken to be a random variable.
- This is against the philosophy of "Bayesian" methods
- Everything is random
- We have an estimate of a "prior" probability
- We make a measurement
- Based on this measurement we convert/update the prior probability to a "posterior" one.
- Thus we are given a prior probability for the parameters, $p(\boldsymbol{\theta})$
- In MAP methods instead of maximizing $l(\boldsymbol{\theta})$ we maximize $l(\boldsymbol{\theta})p(\boldsymbol{\theta})$
- In this context MLE can be viewed as finding the most likely values of $\boldsymbol{\theta}$, assuming all values are equally likely

MAP methods

- The form of the density $p(\mathbf{x}|\boldsymbol{\theta})$ is assumed to be known, but the value of the parameter vector $\boldsymbol{\theta}$ is not known exactly.
- Our initial knowledge about $\boldsymbol{\theta}$ is assumed to be contained in a known a priori density $p(\boldsymbol{\theta})$.
- The rest of our knowledge about $\boldsymbol{\theta}$ is contained in a set $\mathcal{D}$ of n samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ drawn independently according to the unknown probability density $p(\mathbf{x})$.

- Goal: knowing a priori estimate $p(\boldsymbol{\theta})$ compute the posterior estimate $p(\boldsymbol{\theta}|\mathcal{D})$

$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}, \boldsymbol{\theta}|\mathcal{D})\ d\boldsymbol{\theta}, \ = \int p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\ d\boldsymbol{\theta}.$$

By Bayes' formula we have

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})\ d\boldsymbol{\theta}},$$

and by the independence assumption

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta}).$$

Sources

- Christopher Bishop, "Neural Networks for Pattern Recognition", Clarendon Press, 1995.
- R.O. Duda, Hart (and D. Stork), 1973 (new edition expected in 2000.)
  - A classic, but a bit heavy
- Numerical Recipes
  - For general discussion of MLE
- The web