Lagrange Multipliers
- Find stationary points of a function $f(x)$ subject to one or more constraints $g(x) = 0$
- Consider the surface $g(x) = 0$
  - The direction of increase of $f$ is $\nabla f$
  - However moving this direction may take us away from the constraint surface
  - Idea: move along component of $\nabla f$ 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 $\nabla f$ and subtract from it that part $a$ which takes it out of the surface $g$
      $\nabla_g f = \nabla f - a$

Finding the component of $\nabla f$ along $g$
- Now let us move by a distance $\delta$ along the surface $g$
  - $g(x+\delta) = g(x) + \delta \cdot \nabla g$
  - But this still lies on the surface -- so $g(x+\delta) = 0$
  - So $\delta \cdot \nabla g = 0$
  - $\Rightarrow \nabla g$ is perpendicular to motions along the surface
- But we wanted to remove any piece of $\nabla f$ that was perpendicular to $g(x) = 0$
- This will be a vector of the form $\nabla_g f = \nabla f + \lambda \nabla g$
  (For some $\lambda$

Lagrangean
- Consider the Lagrangian function
  $L(x, \lambda) = f + \lambda g$
  $\frac{\partial}{\partial x} L(x, \lambda) = \nabla f + \lambda \nabla g$
  $\frac{\partial}{\partial \lambda} L(x, \lambda) = g$
- Extremize the Lagrangian
  $\frac{\partial}{\partial x} L(x, \lambda) = \nabla f + \lambda \nabla g = 0$
  $\frac{\partial}{\partial \lambda} L(x, \lambda) = g(x) = 0$
- So this gives us both the constraint equation and the way to optimize the function on the surface.

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

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 \( \Sigma = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}}) \)
- 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 \( \mathbf{x} \) have different coordinates \( \{c_k\} \)
- 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

Approximation

- Given a dataset \( \{\mathbf{x}_i\} \) with *N* members
- Write each vector in a basis \( \{\mathbf{u}_k\} \)
- Coefficients
- Approximate each \( \mathbf{x}_i \) as sum of a variable part and a constant part
- Dimension of variable part is \( M = \sum_k c_k^2 \mathbf{u}_k^2 + \sum_k b_k \mathbf{u}_k \mathbf{u}_k \)
- Error in approximating a particular vector \( \epsilon_i = \mathbf{x}_i - \sum_k c_k \mathbf{u}_k - b_k \mathbf{u}_k \mathbf{u}_k \)
- Define sum of squares error function \( C(b) = \sum_i \left( \mathbf{x}_i - \sum_k c_k \mathbf{u}_k - b_k \mathbf{u}_k \mathbf{u}_k \right)^2 \)

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*
- 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?

Getting the parameters \( b_k \) and \( \mathbf{u}_k \)

- Evaluate \( b_k \) by setting \( \partial C(b)/\partial b_k = 0 \)
  \( b_k = \frac{1}{N} \sum_{i=1}^{N} c_k \mathbf{u}_k \)
- To get best basis vectors \( \mathbf{u}_k \) define cost function
  \( E(u) = \sum_i \left[ \sum_k \left( \mathbf{x}_i - \mathbf{u}_k \mathbf{u}_k \right) \right] + \sum_k u_k^2 \Sigma \mathbf{u}_k \mathbf{u}_k \)
  \( = \sum \mathbf{u}_k^2 \Sigma \mathbf{u}_k \mathbf{u}_k \)
- 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 \( |u_j| = 1, \ u_j \cdot u_j = \delta_{jj} \)
- How do we optimize subject to constraints?
  - Lagrange Multipliers!

Cost function with constraints:

\[
E_u = \sum_{j=1}^{N} u_j \sum_{i=1}^{N} u_i \sum_{j=1}^{N} \mu_j (u_j u_i - \delta_{ij})
\]

Can be written in the form:

\[
E_u = Tr \{ U \Sigma U \} - Tr \{ M (U U^T - I) \}
\]

\[
U = [u_1, \ldots, u_N, \ldots, u_N] \quad M = [\mu_j]
\]

- Minimizing with respect to \( u_j \)

\[(\Sigma + \Sigma') U - U (M + M') = 0 \Rightarrow \Sigma U = UM\]

- \( U \) is an orthonormal vector with columns as basis vectors
- So any set of \( U \)s and \( M \)s that satisfy \( U' \Sigma U = M \)

PCA

- Choose the simplest solution
  - \( U \) vectors in the eigenbasis of \( \Sigma \)
  - \( M \) is the diagonal matrix of eigenvalues.
- Algorithm
  1. Compute the mean of the data
     \[
x = (\sum x_i) / N
\]
  2. Compute the covariance of the data,
     \[
     \Sigma = \sum \frac{(x_i - x) (x_i - x)^T}{N}
     \]
  3. Compute eigenvectors, \( u_i \) and corresponding eigenvalues \( \lambda_i \) of \( \Sigma \)
     sorted according to the magnitude of \( \lambda_i \)
  4. For a desired approximation dimension \( M \), \( x \) can be written as
     \[
     x = \sum_{i=1}^{M} \frac{x_i}{\lambda_i} u_i + \sum_{i=M+1}^{d} \tau_i
     \]

Selecting the approximation dimension \( M \)?

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

\[
\frac{\sum \lambda_i}{\sum \lambda_i}
\]

- Two strategies:
  1. Specify the desired threshold e.g. 99%
  2. 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

Problem Introduction
• Model characterized by values of a parameter vector $\theta$
• 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)
  – Maximum A Priori Estimation (MAP)
• Talk will only touch on a vast field, but hopefully will make you familiar with the jargon.

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

Parameter Estimation
MLE and MAP

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

Maximum A Priori Estimation (MAP)
• “Bayesian approach”

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 $l(\theta)$

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 \( p(x|\theta) \).
  - 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 \( \theta \).
  - If observations in the dataset are independent the likelihood function is \( p(D|\theta) = \prod p(x_i|\theta) \).
  - Parameter estimated by maximizing the likelihood or the log with respect to \( \theta \).

- **Algorithm for approximate maximum likelihood parameter estimation when features are missing**
  - Situation:
    - Given a set of \( N \) data points \( x \) belonging to a training set \( \Delta \)
    - Data is \( d \)-dimensional
    - Some of the data points is missing features, or has poorly measurec values
    - Good data point \( x_g = \{x_1, x_2, \ldots, x_N\} \)
    - Bad data point \( x_b = \{x_1, x_2, \ldots, x_k, \ldots, x_N\} \)
  - Separate features into a good set \( D_g \) and a bad set \( D_b \)
  - Using a guess \( \theta \), fix some of the parameters, and form a likelihood function over the unknown features \( Q(\theta|\theta^*) = \epsilon \ln p(D_g|D_b \theta, D_g|D_b) \)
  - Maximize \( Q \) with respect to the unfixed values.
  - Repeat for the previously fixed values.

- **Maximum A Posteriori Estimation**
  - In MLE the estimated value of the parameter vector \( \theta \) 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, \( p(\theta) \)
  - In MAP methods instead of maximizing \( l(\theta) \) we maximize \( l(\theta)p(\theta) \)
  - In this context MLE can be viewed as finding the most likely values of \( \theta \), assuming all values are equally likely.

**Algorithm 1 (Expectation-Maximization)**

1. **initialize \( \theta^0 \), \( T \), \( t = 0 \)**
2. **while \( t < T \)**
3. **M step: \( \theta^{t+1} = \arg \max Q(\theta, \theta^t) \)
4. **E step: \( Q(\theta|\theta^t) = \epsilon \ln p(D_g|D_b \theta, D_g|D_b) \)
5. **return \( \theta^{t+1} \)**
6. **end

- 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 3.1**: The search for the best model via the EM algorithm starts with some initial state of the model parameters. \( \theta \). Then we do M-step the parameter \( \theta \) is fixed. Now, \( \theta \) is held constant and the value of \( \theta \) found which maximizes \( Q(\theta|\theta^*) \). This process repeats and more data can be found (line 1 will increase). Here the purpose that this is different from a similar search. For example, line 3 is plotted where the best value of \( \theta \) is fixed, then line 4 is plotted where the best value of \( \theta \) is fixed but for the case in normal, whereas.

**MAP methods**

- The form of the density \( p(x|\theta) \) is assumed to be known, but the value of the parameter vector \( \theta \) is not known exactly.
- Our initial knowledge about \( \theta \) is assumed to be contained in a known a priori density \( p(\theta) \).
- The rest of our knowledge about \( \theta \) is contained in a set of data \( \Delta \) of \( n \) samples \( x_1, \ldots, x_n \) drawn independently according to the unknown probability density \( p(x) \).

- **Goal**: knowing a priori estimate \( p(\theta) \) compute the posterior estimate \( p(\theta|\Delta) \)

\[
p(x|\Delta) = p(x, \theta|\Delta) = \int p(x|\theta)p(\theta|\Delta) \, d\theta
\]

By Bayes’ formula we have

\[
p(\theta|\Delta) = \frac{p(x|\theta)p(\theta|\Delta)}{\int p(x|\theta)p(\theta|\Delta) \, d\theta}
\]

and by the independence assumption

\[
p(\theta|\Delta) = \prod_{i=1}^{n} p(x_i|\theta).
\]

**Sources**

  - A classic, but a bit heavy
- **Numerical Recipes**
  - For general discussion of MLE
- The web