

# CSCD 409 Scientific Programming

## Module 5d: Projection Pursuit

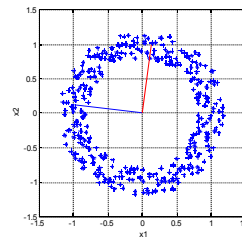
© 2009-2016, Paul Schimpf

The road analogy was borrowed from Jonathon Shlens, Center for Neural Science, New York University

## Orthogonal Basis

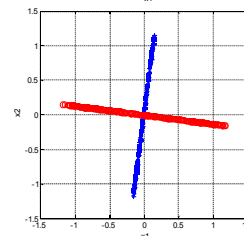
- PCA restricts the search for a new basis to orthogonal directions

- and to components that are linear combinations of the original measurement basis
- this makes the problem solvable with linear algebra
- but it doesn't always work



- Consider this data

- clearly, both variables can largely be predicted from a single variable (angle)
- $x_1 = \cos(\theta)$
- $x_2 = \sin(\theta)$
- but PCA can't figure that out

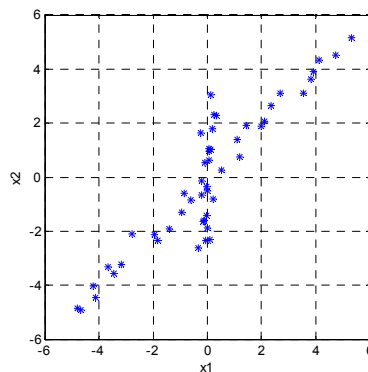


## An Analogy?

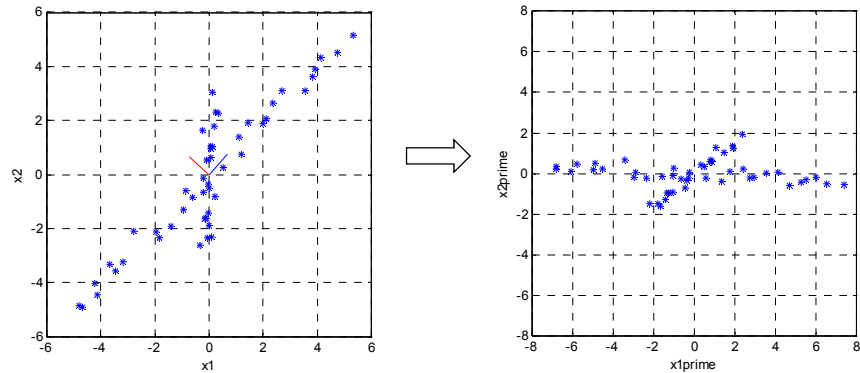
- **PCA operates the way one might explore a town**
  - drive down the longest road through town first
  - when you see the second longest road, drive down that
  - PCA requires that each new road explored be perpendicular to the previous (in fact, perpendicular to all previous, so the analogy breaks down a bit there, but hopefully you get the idea)
  - However, not all towns (data) are built up along perpendicular roads
- **A refinement: Kernel PCA**
  - data are preconditioned by some nonlinear kernel before processing
  - in the previous example, by first expressing the data in terms of new variables,  $\theta$  and  $r$ , which requires nonlinear (trigonometric) operations (I'll ask you to pursue a slightly different idea in the H/W)
  - this requires some a-priori knowledge of the system
- **Projection Pursuit / Independent Component Analysis**
  - requires that the projected data be statistically independent, NOT that they be projected onto orthogonal axes (linearly independent)
  - $P(AB) = P(A) P(B)$

## Let's Visualize This Problem

- **Given this data set, where would you eyeball the important axes of variability?**
  - and where do you think PCA would put them?



## Here's what PCA does



- PCA will express the variations along two axis that are orthogonal
  - whereas it appears that expressing the data as variations along two non-orthogonal axis might be useful here
  - especially if there were more dimensions

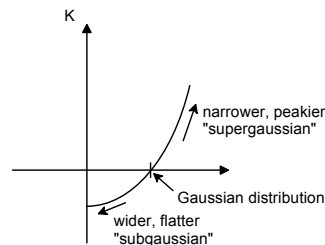
## Projection Pursuit

- Given  $n$  data samples, each of dimension  $m$
- Start with a data set whose variables (axes) are uncorrelated (via PCA)
- For each dimension (axis)
  - Generate a randomly-oriented  $m$ -dimensional unit vector, as a trial axis (call this the unmixing vector,  $w$ )
  - Project all data points onto  $w$
  - Measure the likely "independence" of that projected data ->Measure the kurtosis
  - Rotate  $w$  in a direction that increases the "independence" and keep doing so until it no longer increases
  - Record the final  $w$  as the basis vector for the current dimension
  - Remove the data projections onto that final  $w$  from the dataset, leaving the residuals to work on for the next axis (which needn't be orthogonal in the original space)

## Measure Independence?

- **Mixtures of independent signals tend to have Gaussian distributions**

- in fact, the Central Limit Theorem guarantees this (eventually)
- so we will look for projections that produce non-Gaussian distributions
- Kurtosis is a measure of non-Gaussian'ness
- so, maximize the Kurtosis (or perhaps its square)



as the spread gets large and flat, this ratio  $\rightarrow 1$

as the spread gets small and peaky, this ratio  $\rightarrow \text{inf}$

for a Gaussian distribution, the first term is 3

Def'n of Kurtosis

$$K = \frac{\frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^4}{\left( \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 \right)^2} - 3$$

variance

\*There is a Matlab Kurtosis function

look up: Independent component analysis--axis project independent data

## About the last step (Removal)

- We have m mixtures, each containing n samples

- matrix X is (m x n)

- We project them onto w:

$$y = w^T X \quad (1 \times m) (m \times n) = (1 \times n) \quad \text{Basically a scalar projection.}$$

- the result is a collection of n scalars (*scalar* projection) representing the variation along new basis vector w
- (this projected data is what we are typically after if it represents some kind of source signal)
- we now want to remove that variation from the dataset X
- IOW, we want to subtract a *vector* projection

$X_{\text{new}} = X - wy = X - ww^T X$  (Gram-Schmidt orthogonalization)

$X_{\text{new}}$  = Modified data we are using for the next iteration

- This projection and removal requires that the original variables be uncorrelated

- this is why the first step is to transform our data using PCA

## Another way to look at it

- Another way to think about **PCA pre-processing**
  - we are looking for a separation of these data into sets that are statistically independent
    - $P(xy) = P(x) P(y)$ , which requires that
    - $E[x^p y^q] = E[x^p] E[y^q]$  for all positive powers  $p$  and  $q$
  - PCA produces a set of signals that are *uncorrelated*, and thus *linearly independent*, which is a weaker property in that it only requires that
    - $E[xy] = E[x] E[y]$  (i.e., only for  $p=1$  and  $q=1$ )
  - because we are looking for a set of variables (axes) that have at least that property, it makes sense to start with an uncorrelated set, which we can obtain with PCA pre-processing

## Kurtosis Calculation

$$K = \frac{\frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^4}{\left( \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 \right)^2} - 3$$

Don't need to calculate this if the data is zero-mean

Don't need to calculate this if all we care about is maximization (IOW, we can ignore this if we don't care what the actual Kurtosis is, and don't care about sub-Gaussian signals)

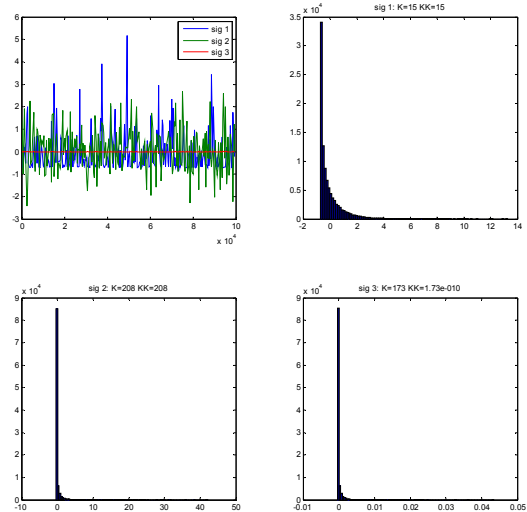
We don't need this if we normalize our projected data to unit variance

- More about the denominator:
  - Maybe you don't want to force your trial projections to unit variance, because you want to preserve variations in amplitude for the unmixed sources
  - In that case there is still some reason to ignore the denominator ...

## K vs. 4th Moment (numerator)

- Suppose we have the following mixture of signals

- K is full Kurtosis, KK is numerator of Kurtosis
- which two signals would you want to extract first?
- note that K is a *normalizing* measurement of "peakiness"
- it is the same regardless of signal amplitude (variance)
- KK is larger for signals with larger variance



## Searching for w

- How do we decide how to rotate w to increase KK?

- K now refers to just the numerator, which is  $E[y^4]$
- start with a random guess for w, make sure it has unit amplitude
- we want to compute the slope of K and move in an uphill direction

\*Gradient ascent

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \quad w_{next} = w + \eta \frac{\Delta K}{\Delta w}$$

Parameter to control the step size that we'll allow the end user to control. The user will be able to judge refinements by looking at a history of how K evolves.  $w_{next}$  should also be normalized to unit amplitude (by dividing it by its norm)

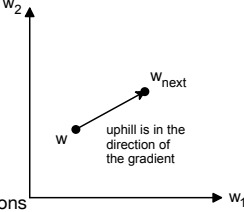
- a constant  $\eta$  still means smaller steps as we approach the peak of a well-behaved (smooth) function (why?)
- in this case it is possible to derive, analytically, a formula for the gradient of K, but that requires some knowledge of multivariate calculus
- so next we look at a cruder but more general approach of *probing* the function to estimate the gradient

## Gradient Ascent by Probing

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \quad w_{next} = w + \eta \frac{\Delta K}{\Delta w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + \eta \begin{bmatrix} \Delta K / \Delta w_1 \\ \Delta K / \Delta w_2 \end{bmatrix}$$

$$\frac{\Delta K}{\Delta w_1} = \frac{K\left(y\left(w + \begin{bmatrix} h \\ 0 \end{bmatrix}\right)\right) - K(y(w))}{h} \quad \frac{\Delta K}{\Delta w_2} = \frac{K\left(y\left(w + \begin{bmatrix} 0 \\ h \end{bmatrix}\right)\right) - K(y(w))}{h}$$

$y$  is a function of  $w$ .  $y$  is a projection. Will have to do this for  $h$ -dimensions



- **For each dimension of  $w$  ( $m=2$  shown)**
  - add a small (relative to 1) offset,  $h$ , to that dimension (I call this is the probe size, also a parameter that we will pass in)
  - recompute  $K$  using that  $w$  as the projection:  $y(w)=w^T X$
- **Estimate the gradient vector from the changes in  $K$  divided by the change in  $w$  (which is  $h$ )**
- **Take a step in that direction**
- **Stop when the increase in  $K$  falls below some relative change:  $\text{abs}((K_{\text{new}} - K) / K) < \text{tol}$**

## Does it work?

- **See sound example**
- **Reproducing this result is the term project for graduate students**
- **Feel free to use an analytic formula for the gradient of the "kurtosis" instead of probing**
  - this is not hard to derive if you're comfortable with vector calculus (manipulating derivatives in vectors and matrices)
  - if you're not comfortable in that area of mathematics, but know how to find the derivative of a 1-dimensional polynomial ...
  - you can still derive this by starting with a small example (say  $m \times n = 2 \times 3$ ), expanding  $w$  and  $X$  into their components, finding the derivative, and then collecting terms back into matrix-vector notation