ORIE 779: Functional Data Analysis

From last meeting

Finished Strangeness of high dimensional space

FLD - Expanding dimension view

Revisited ICA (from discrimination viewpoint)

- Toy examples
- Corpora Collosa Data

ICA for Corpora Collosa Data

Typically found outliers

Tried several approaches (looking for “separation of classes”):

- PCA start
- Reduced subspace
- Project to sphere
- Different non-linearities (tanh and gauss)

ICA for Corpora Collosa Data (cont.)

Idea for improvement: find “directions to minimize kurtosis”

(not absolute value of kurtosis)

Implementation (short of recoding ICA):

1. Look in all 20 ICA directions (for some choice of opt’s)
2. Compute kurtosis for each
3. Sort in increasing kurtosis order

ICA for Corpora Collosa Data (cont.)

Attempts:

a. Absolute Kurtosis, random start [combined graphic]:

- all kurtoses > 0, found *no* “useful directions”

b. Absolute Kurtosis, PC start [combined graphic]:

- found a bimodal direction (discovered earlier)
- and a 2nd direction with kurtosis < 0
- “Start” is still an important issue

ICA for Corpora Collosa Data (cont.)

c. Tanh, random start [combined graphic]:

- found 4 directions with kurtosis < 0
- none give “magic bullet” discrimination
- maybe “4 together” (e.g. input to CART) can do well?

d. Tanh, PC start [combined graphic]:

- OK, but not so good as (c)

ICA for Corpora Collosa Data (cont.)

e. Gaus, random start [combined graphic]:

  - similar to above

f. Gaus, PC start [combined graphic]:

  - again 4 directions with strongly negative kurtosis
  - quite different directions from those in (c)?

ICA for Corpora Collosa Data (cont.)

Some conclusions and ideas:

i. ICA is a very promising method

ii. Starting point is critical (and poorly understood)

iii. Would like to try explicitly minimizing kurtosis

Support Vector Machines

Mechanics described last time by Rommel Regis

One type of motivation:

Polynomial Embedding

Polynomial Embedding

Aizerman, Braverman and Rozoner (1964) *Automation and Remote Control*, **15**, 821-837.

Motivating idea: extend "scope" of linear discrimination,

by adding "nonlinear components" to data

(better use of name "nonlinear discrimination"????)

E.g. In 1d, "linear separation" splits the domain

$$\{x : x \in \Re\}$$

into only 2 parts [toy graphic]

Polynomial Embedding (cont.)

But in the “quadratic embedded domain”

$$\{(x, x^2) : x \in \Re\} \subset \Re^2$$

linear separation can give 3 parts [toy graphic]

- original data space lies in 1d manifold
- very sparse region of $\Re^2$
- curvature of manifold gives better linear separation
- can have *any* 2 break points (2 points $\Rightarrow$ line)

Polynomial Embedding (cont.)

Stronger effects for higher order polynomial embedding:

E.g. for cubic, $\left\{\left(x, x^2, x^3\right) : x \in \Re\right\} \subset \Re^3$

linear separation can give 4 parts (or fewer) [toy graphic]

- original space lies in 1d manifold, even sparser in $\Re^3$
- higher d curvature gives improved linear separation
- can have *any* 3 break points (3 points $\Rightarrow$ plane)?
- relatively few “interesting separating planes”

Polynomial Embedding (cont.)

General View: for original data matrix:

$$\begin{pmatrix} x_{11} & & x_{1n} \\ \vdots & \cdots & \vdots \\ x_{d1} & & x_{dn} \end{pmatrix}$$

“add rows”:

$$\begin{pmatrix} x_{11} & & x_{1n} \\ \vdots & & \vdots \\ x_{d1} & & x_{dn} \\ x_{11}^2 & \cdots & x_{1n}^2 \\ \vdots & & \vdots \\ x_{d1}^2 & & x_{dn}^2 \\ x_{11}x_{21} & & x_{1n}x_{2n} \\ \vdots & & \vdots \end{pmatrix}$$

Polynomial Embedding (cont.)

Fisher Linear Discrimination: Choose Class 1 for $\underrightarrow{x}^{0}$ when:

$$\underrightarrow{x}^{0^t} \hat{\Sigma}^{w^{-1}} \left(\underrightarrow{\overline{X}}^{(1)} - \underrightarrow{\overline{X}}^{(2)} \right) \leq \frac{1}{2} \left(\underrightarrow{\overline{X}}^{(1)} + \underrightarrow{\overline{X}}^{(2)} \right) \hat{\Sigma}^{w^{-1}} \left(\underrightarrow{\overline{X}}^{(1)} - \underrightarrow{\overline{X}}^{(2)} \right)$$

in *embedded* space.

- image of class boundaries in original space is *nonlinear*
- allows much more *complicated* class regions
- can also do Gaussian Likelihood Ratio (or others)

Polynomial Embedding Toy Examples

E.g. 1: Parallel Clouds

- PC1 always bad (finds “embedded greatest var.” only)
- FLD stays good
- GLR OK discrimination at data, but ∃ overfitting problems

Polynomial Embedding Toy Examples (cont.)

E.g. 2: Split X

- FLD rapidly improves with higher degree

- GLR always good, but never “ellipse around blues”???

- Should apply ICA first??? Or perhaps instead???

Polynomial Embedding Toy Examples (cont.)

E.g. 3: Donut

- FLD: poor for low degree, then good, no overfit

- GLR: best with no embed, “square shape” for overfitting?

Polynomial Embedding (cont.)

Drawback to polynomial embedding:

- too many extra terms create spurious structure
- i.e. have “overfitting”
- High Dimension Low Sample Size problems worse

Kernel Machines

Idea: replace polynomials by other "nonlinear functions"

e.g. 1: "sigmoid functions" from neural nets

e.g. 2: "radial basis functions" – Gaussian kernels

Related to "kernel density estimation" (smoothed histogram)

Kernel Machines (cont.)

Radial basis functions: at some "grid points" $\underrightarrow{g}_1, \ldots, \underrightarrow{g}_k$,

For a "bandwidth" (i.e. standard deviation) σ,

Consider (d dim'al) functions: $\varphi_\sigma\left(\underrightarrow{x} - \underrightarrow{g}_1\right), \ldots, \varphi_\sigma\left(\underrightarrow{x} - \underrightarrow{g}_k\right)$

Replace data matrix with:

$$\begin{pmatrix} \varphi_\sigma\left(\underrightarrow{X}_1 - \underrightarrow{g}_1\right) & & \varphi_\sigma\left(\underrightarrow{X}_n - \underrightarrow{g}_1\right) \\ \vdots & \cdots & \vdots \\ \varphi_\sigma\left(\underrightarrow{X}_1 - \underrightarrow{g}_k\right) & & \varphi_\sigma\left(\underrightarrow{X}_n - \underrightarrow{g}_k\right) \end{pmatrix}$$

Kernel Machines (cont.)

For discrimination: work in radial basis function domain,

With new data vector $\underrightarrow{X}_0$ represented by:
$$\begin{pmatrix} \varphi_\sigma\left(\underrightarrow{X}_0 - \underrightarrow{g}_1\right) \\ \vdots \\ \varphi_\sigma\left(\underrightarrow{X}_0 - \underrightarrow{g}_1\right) \end{pmatrix}$$

Kernel Machines (cont.)

Toy Examples:

E.g. 1: Parallel Clouds – good at data, poor outside

E.g. 3: Split X – OK at data, strange outside

E.g. 5: Donut – mostly good (slight mistake for one kernel)

Main lesson: generally good in regions with data,
unpredictable results where data are sparse

Kernel Machines (cont.)

E.g. 7: Checkerboard

- Kernel embedding (FLD or GLR) is excellent
- While polynomials (FLD – GLR) lack flexibility
- Lower degree is worse

Kernel Machines (cont.)

Note: Gaussian Likelihood Ratio had frequent numerical failure

Important point for kernel machines:

High Dimension Low Sample Size problems get worse

This is motivation for “Support Vector Machines”

Kernel Machines (cont.)

$\exists$ generalizations of this idea to other types of analysis,

and some clever computational ideas.

E.g. “Kernel based, nonlinear Principal Components Analysis”

Schölkopf, Smola and Müller (1998) “Nonlinear component analysis as a kernel eigenvalue problem”, *Neural Computation*, **10**, 1299-1319.

Support Vector Machines

Classical References:

Vapnik (1982) *Estimation of dependences based on empirical data*, Springer (Russian version, 1979)

Boser, Guyon & Vapnik (1992) in *Fifth Annual Workshop on Computational Learning Theory*, ACM.

Vapnik (1995) *The nature of statistical learning theory*, Springer.

Recommended tutorial:

Burges (1998) A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, **2**, 955-974, see also web site:

http://citeseer.nj.nec.com/burges98tutorial.html

Support Vector Machines (cont.)

Motivation: High Dimension Low Sample Size discrimination

(e.g. from doing a nonlinear embedding)

$\exists$ a tendency towards major *over-fitting* problems

Toy Example:

In 1st dimension: Class 1: $N(2,0.8)$ Class 2: $N(-2,0.8)$
$(n=20$ of each, and threw in 4 "outliers")

In dimensions $2,...,d$: independent $N(0,1)$

Support Vector Machines (cont.)

Toy Example: for linear discrimination:

Top: Proj'n onto (2-d) subspace generated by 1st unit vector (- -)
and Discrimination direction vector (----) (shows angle)

For "reproducible (over new data sets) discrimination":

Want these "near each other", i.e. small angle

Bottom: 1-d projections, and smoothed histograms

Support Vector Machines (cont.)

Lessons from Fisher Linear Discrimination Toy Example:

- Great angle for $d = 1$, but substantial overlap
- OK angle for $d = 2,...,10$, still significant overlap
- Angle gets very bad for $d = 11,...,18$, but overlap declines
- No overlap for $d \geq 23$ (perfect discrimination!?!?)
- Completely nonreproducible (with new data)
- Thus useless for real discrimination

Support Vector Machines (cont.)

Main Goal of Support Vector Machines:

Achieve a trade off between:

Discrimination quality for data at hand

vs.

Reproducibility with new data

Approaches:

1. Regularization (bound on “generaliz’n”, via “complexity”)
2. Quadratic Programming (general’n of Linear Prog.)

Support Vector Machines (cont.)

Implementation: Matlab code from:

http://www.isis.ecs.soton.ac.uk/resources/svminfo/

(Caution: must use class labels ± 1)

Many others web available, e.g. see:

http://www.kernel-machines.org/software.html

Support Vector Machines (cont.)

Caution: SVM is not robust, instead "feels outliers"

- reason is "higher penalty for data farther from plane"
- note "jumping effect" – nonlinear min'ing artifact????

Can get strange results in "indeterminate case":

- generally good, stable answer
- but hardly inside data at "crossing point"?

Support Vector Machines (cont.)

Possible weakness: can be “strongly driven by a few points”

- huge “range of chosen hyperplanes”
- but all are “pretty good discriminators”
- only happens when “whole range is OK”????

Support Vector Machines (cont.)

Revisit toy examples (from Polynomial Embedding):

E.g. Parallel Clouds:

- SVM and FLD very comparable

E.g. Split X:

- SVM & FLD fairly comparable
- SVM had worse overfitting at cubic (could fix via C????)

Support Vector Machines (cont.)

E.g. Donut:

- SVM & FLD fairly comparable
- SVM gives better “cutoffs” at higher degrees
- since non-elliptical data, in high degree embedded space

E.g. Checkerboard – Kernel embedding

- SVM gives better boundaries than FLD
- But not so good as GLR

General Conclusion about Discrimination

"There Ain't No Such Thing As a Free Lunch"

I.e. each method can be:

- Great
- Very Poor

Depending on context, and data set at hand.

Thus useful to understand, and to have a big bag of tricks.

Validation for Discrimination

How “well” does a method work?

Theoretical Answer: for a random point from the underlying distributions, what is the probability of “correct classification”

Naïve Empirical Answer: proportion of training data correctly classified

Problem 1: tendency towards “too optimistic”

Problem 2: Way off for overfitting (e.g. HDLSS) [toy example]

Validation for Discrimination (cont.)

Better empirical answers: Cross-Validation

Simplest version:

- Use ½ the data to “train”, i.e. construct the discrim’n rule
- Use the other ½ to “assess”, i.e. compute error rate
- Unbiased est. of prob. of correct classification
- But get error rate for “wrong sample size”

Validation for Discrimination (cont.)

Cross-validation (cont.)

More sophisticated version: Leave – One - Out

- Train with all but one data point
- Use that for assessment
- Repeat for each data point
- Still unbiased est. of prob. of correct classification
- Much closer to correct sample size

Alternate view of Discrimination

Neural Networks:

“The Language” in

Duda, R. O., Hart, P. E. and Stork, D. G. (2001) *Pattern Classification*, Wiley.

Advantages:

- Broad framework that “includes everything”
- Excellent results in some situations

Drawbacks:

- Only a “black box”
- Usually no insight as to “why?”
- “Good discrimination may not be “learning about data”

Wish I had more time for:

1. PCA time series – chemometrics data
2. ICA in discrimination
3. In vector space, orthogonal basis introduction
4. Fourier basis 3-22-01
5. Legendre basis
6. Tensor product Fourier Legendre basis
7. Zernike basis
8. Revisit cornea data?   (compare “raw image” with “fit images”, fiddle with Cornean power map? (do this at home?), use Figure from LMTZ paper, see directories D:\DellInspiron7000\SW30\Docs\Steve and D:\DellInspiron7000\SW30\Pictures)
9. Elliptical Fourier bases 4-05-01
10. Complex plane representation (no simple real valued basis)
11. Corpora Collosa Approximation
12. Support Vector Machines
13. Polynomial Embedding

14. Micro-Array Data analysis
15. Normal KerCli discrimination (in Cornean/demo)

Old material had no time for:

CC Sequential ICA vs. Simultaneous ICA 3-22-01: 3-5

ICA and Projection Pursuit 3-22-01: 6-13

Poly embedding: 4-19-01: pg. 2-13

Kernel machines: 4-19-01: pg. 14-22

SVM: 4-19-01: pg. 23-27, 5-2-01: pg. 2-20

Validation of discrimination: 5-2-01: pg. 21-32