Computational Tricks

Simon Rückinger

Seminar:
Modeling, Simulation and Inference of Complex Biological Systems
(Dr. Korbinian Strimmer)
SoSe 06

9. Juni 2006

Overview

Objective

Computationally efficient algorithms for quadratic regularisation

Starting point

- A set of gene expression arrays **X**
  → large p (columns), small n (rows).
- Output value **y** we want to predict from the expression values.

Idea

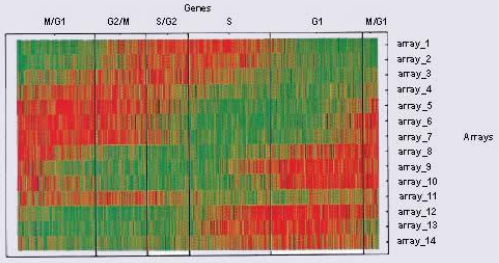
Decompose **X** in a tricky way and reduce computational effort (Matrix Inversion).

Table of contents

Data

Microarray data (p genes, n arrays)

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}$$



What can we do with gene expression data?

Output value

Usually there exists an additional measure **y** for each array (individual, time,...). This could be

- cancer class
- biological species
- survival time
- any other quantitative/qualitative measure

Making predictions

A typical goal for a statistician would be to find a model which

- shows the association between gene expression and the output value **y**
- is able to make further predictions for **y**.

Predicting y via linear regression

Ordinary least squares

Linear regression could be the model of choice for predicting **y** as a continuous measure. To get the model parameters we simply minimize the sum of squared differences between observed (y_i) and predicted values ($x_i^T \beta$).

$$\min_{\beta} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2$$

Solution for classic linear regression

This leads to the well-known solution

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

Linear regression does not work for $p > n$

Normal equations

When $p > n$, the so-called normal equations

$$X^T X \beta = X^T y$$

which lead to

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

do not have a unique solution for β. Instead they provide infinitely many solutions which

- all fit the training data perfectly
- usually are not able to make reliable predictions.

In other words: $(X^T X)$ is not invertible.

Quadratic Penalization

Ridge Regression (Hoerl *et al.*, 1970)

A classic attempt for the $p > n$ situation is to add a quadratic penalty to the OLS-criteron.

$$\min_{\beta} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \beta^T \beta$$

which leads to the solution

$$\hat{\beta} = (X^T X + \lambda \mathbb{I})^{-1} X^T y$$

Benefits

$(X^T X + \lambda \mathbb{I})$ is an invertible matrix. We get a unique solution for β.

Expensive Computation

Matrix Inversion

$(X^T X + \lambda \mathbb{I})$ is a $p \times p$ matrix. Thus the computational cost of inverting this matrix is $O(p^3)$ operations.

p is large

Typical expression arrays include between 1,000 and 20,000 genes. For 10,000 genes it means we have to invert a $10,000 \times 10,000$-matrix. This involves $O(10,000^3)$ operations.

How long does it take?

Even on a modern PC this is a matter of hours or even days.

Table of contents

Getting 'tricky'

Singular value decomposition (SVD)

Every Matrix $X \in \mathbb{R}^{p \times n}$ can be decomposited in the following way

$$X = UDV^T = RV^T$$

$$\underbrace{\begin{bmatrix} \\ X \\ \\ \end{bmatrix}}_{nxp} = \underbrace{\begin{bmatrix} \\ U \\ \\ \end{bmatrix}}_{nxn} \underbrace{\begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix}}_{nxn} \underbrace{\begin{bmatrix} \\ V \\ \\ \end{bmatrix}^T}_{pxn}$$

Getting 'tricky'

Singular value decomposition (SVD)

Every Matrix $X \in \mathbb{R}^{p \times n}$ can be decomposited in the following way

$$X = UDV^T = RV^T$$

$$\underbrace{\begin{bmatrix} \\ X \\ \\ \end{bmatrix}}_{nxp} = \overbrace{\underbrace{\begin{bmatrix} \\ U \\ \\ \end{bmatrix}}_{nxn} \underbrace{\begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix}}_{nxn}}^{R} \underbrace{\begin{bmatrix} \\ V \\ \\ \end{bmatrix}^T}_{pxn}$$

Applying SVD to ridge regression

Another solution for ridge regression

Plugging $X = RV^T$ into

$$\hat{\beta} = \underbrace{(X^T X + \lambda \mathbb{I})}_{pxp}^{-1} X^T y$$

one gets

$$\hat{\beta} = V \underbrace{(R^T R + \lambda \mathbb{I})}_{nxn}^{-1} R^T y$$

Conclusion

We can get the parameters for ridge regression by

- applying SVD to X
- using R as a 'data-matrix'
- multiplying β by V

About Singular Value Decomposition

Computational costs

Getting the parameters for ridges regression via the SVD trick involves

- reducing p variables to n variables by SVD $\rightarrow O(pn^2)$
- solving n dimensional ridge regression $\rightarrow O(n^3)$
- transforming back to p dimensions $\rightarrow O(np)$

Thus it saves inverting a $p \times p$ matrix with $O(p^3)$ operations.

Software

SVD is a standard linear algebra tool. It is implemented in most mathematical languages. In `R`

- `svd()`
- `fast.svd()` $\rightarrow$ faster for small n, large p (`library(corpcor)`)

Applying SVD to other models I

Linear predictors

Many models 'connect' the variables with the outcome through a **linear predictor**. E.g.

- logistic regression
- linear discriminant analysis
- support-vector machines

Loss functions

The parameters are estimated by minimizing a **loss function** $\sum_{i=1}^{n} L(y_i, \beta_0 + x_i^T \beta)$ which can be the

- squared error
- negative log-likelihood
- etc...

Applying SVD to other models II

Quadratic regularization

Like linear regression, all models using a linear predictor and some kind of loss function don't work properly when $p \gg n$.
This can be fixed by adding a **quadratic penalty**

$$\min_{\beta_0, \beta} \sum_{i=1}^{n} L(y_i, \beta_0 + x_i^T \beta) + \lambda \beta^T \beta.$$

The good news is

- The SVD trick can be used with all models in the above mentioned form.
- All aspects of model evaluation can be performed in this reduced space.

Reduced space computations

Theorem 1

Let $X = RV^T$ be the SVD of X, and denote by r_i the ith row of R, a vector of n predictor values for the ith observation. Consider the pair of optimization problems:

$$(\hat{\beta_0}, \hat{\beta}) = \underset{\beta_0, \beta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \beta_0 + x_i^T \beta) + \lambda \beta^T \beta$$

$$(\hat{\theta_0}, \hat{\theta}) = \underset{\theta_0, \theta \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \theta_0 + r_i^T \theta) + \lambda \theta^T \theta$$

Then $\hat{\beta_0} = \hat{\theta_0}$, and $\hat{\beta} = V\hat{\theta}$.

Proof

see Hastie *et al.*, 2004

Cross-validation I

Where to get λ from?

The regularization parameter λ is often selected by k-fold cross-validation.

How does cross-validation work?

- divide the training data into k groups of size $\frac{n}{k}$
- fit the model to $\frac{k-1}{k}$ and test it on $\frac{1}{k}$ of the data
- repeat this k seperate times
- average the results

This is done for a series of values for λ, and a preferred value is chosen.

Cross-validation II

Corollary 1

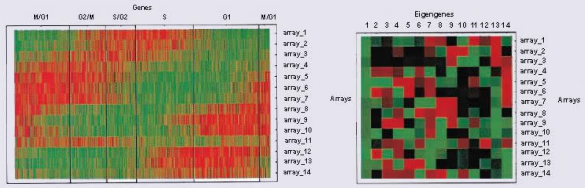
The entire model-selection process via cross-validation can be performed using a single reduced data set **R**. Hence, when we perform cross-validation, we simply sample from the rows of **R**.

Proof

Cross-validation relies on predictions $x^T\beta$, which are equivariant under orthogonal rotations. □

Illustration / Eigengenes

X→R



Eigengenes

If the columns of X are centered, the columns or R

- are the principal components of X
- also called **eigengenes**.

Derivatives I

1st/2nd derivate

If the loss function is based on a log-likelihood we can:

- perform score tests with its first derivative
- gain variances of the parameters from its second derivative (information matrix / hessian matrix)

Expensive computation

Differentation in the p-dimensional space is a waste of time.

SVD once again

SVD helps us to obtain the p-dimensional derivates from the n-dimensional versions.

Derivatives II

Corollary 2

Define

$$L(\beta) = \sum_{i=1}^{n} L(y_i, \beta_0 + x_i^T \beta), \qquad L(\theta) = \sum_{i=1}^{n} L(y_i, \beta_0 + r_i^T \theta).$$

Then with $\theta = V^T\beta \quad \Rightarrow \quad L(\beta) = L(\theta)$.
If L is differentiable, then

$$\frac{\partial L(\beta)}{\partial \beta} = V \frac{\partial L(\theta)}{\partial \theta};$$

$$\frac{\partial^2 L(\beta)}{\partial \beta \partial \beta^T} = V \frac{\partial^2 L(\theta)}{\partial \theta \partial \theta^T} V^T,$$

with the partial derivatives in the right-hand side evaluated at $\theta = V^T\beta$

Derivatives III

Proof of Corollary 2

The euqivalence

$$L(\beta) = L(\theta)$$

follows immediately from the identity

$$X = RV^T,$$

and the fact that x_i^T and r_i^T are the ith rows of **X** and **R**.
The derivatives are simple applications of the chain rule to $L(\beta) = L(\theta)$. □

Parenthesis: Woodbury matrix identity

Matrix inversion lemma

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

where

- A is $p \times p$
- U is $p \times n$
- C is $n \times n$
- V is $n \times p$

Link to SVD

For

- $A = \lambda \mathbb{I}$
- $UCV = X^TX = (VDU^T)(UDV^T) = VD^2V^T$

we see the coherency with SVD.

Table of contents

Where SVD can be applied

Classes of models

SVD is applicable to all models in the mentioned form, of which there are many.

Examples

- Generalized linear models
- The Cox proportional hazards model
- Multiple logistic regression
- Regularized linear discriminant analysis
- Neural networks

Example 1

Ramaswamy *et al.*, 2001

Multiple logistic regression in order to predict tumor class:

- $n = 144$ training tumor samples
- 14 tumor classes
- $p = 16063$ genes for each sample

Amount of time for parameter estimation

- without SVD $\longrightarrow$ 8 days
- using SVD $\longrightarrow$ 0.4 seconds

Example 2

My own decent example

Simulation of a

- gene expression data set X: $p = 2000$ genes, $n = 3$ arrays (`rnorm()`).
- continuous output value y for each array (again `rnorm()`)

Fitting a ridge regression model

- conventional $\longrightarrow$ approx. 1 minute
- with SVD $\longrightarrow$ less than a second

see 'svd.R' on course homepage

Table of contents

Summary I

Why?

Estimating parameters in models for microarray data often involves inverting large $(p \times p)$ matrices. This comes along with high computational costs.

How?

$$X = RV^T \qquad (SVD)$$

R can be used as a new data matrix with dimension $n \times n$. The p-dimensional parameter is then calculated in the following way

$$\hat{\beta} = V\hat{\theta} \qquad \beta \in \mathbb{R}^p, \theta \in \mathbb{R}^n$$

Summary II

When?

SVD works with all models involving

- a linear predictior
- some loss function
- a quadratic penalty

And...

Important aspects of model evaluation, such as

- cross-validation
- testing parameters

can be performed with the reduced data set **R**.

Does it afford?

The time saving can be enormous!

Literature

- Hastie, T. and Tibshirani, R. 2004. Efficient quadratic regularization for expression arrays. *Biostatistics* **5**:329-340.
- Alter, O., Brown, P. and Botstein, D. 2000. Singular value decomposition for genome-wide expression data processing and modelling. *Proceedings of the National Academy of Sciences, USA* **97**, 10101-10106
- Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C., Angelo, M., Ladd, C., Reich, M., Latulippe, E., Mesirov, J. *et al.*, (2001). Multiclass cancer diagnosis using tumor gene expression signature. *Proceedings of the National Academy of Sciences, USA* **98**, 15149-15154
- Hoerl, A. E., Kennard, R. W. 1970. Ridge Regression: Biased Estimation for Non-Orthogonal Problems, *Technometrics*, **12**, 55-67.