APTS Stat Comp Assessment 2013/14

The work provided here is intended to take up to half a week to complete. Students should talk to their supervisors to find out whether or not their department requires this work as part of any formal accreditation process (APTS itself has no resources to assess or certify students). It is anticipated that departments will decide on the appropriate *level* of assessment locally, and may choose to drop some (or indeed all) of the parts, accordingly. So make sure that your supervisor or local organizer of APTS assessment has looked at the assignment before you start, and has told you which parts of it to do. In order to avoid undermining institutions' local assessment procedures the module lecturers will not respond to enquiries from students about this assignment.

A generalized linear model (GLM) relates independent response variables, y_i, to some predictors, via the following structure

$$g(\mu_i) = \mathbf{X}_i\boldsymbol{\beta} \quad \text{where} \quad \mathbb{E}(y_i) = \mu_i \quad \text{and} \quad y_i \sim \text{exponential family}$$

g is a known smooth monotonic 'link function', $\mathbf{X}$ is a model matrix, determined by the predictors (as in a linear model) and $\boldsymbol{\beta}$ is a vector of unknown parameters to be estimated. A key feature of GLMs is that the exponential family assumption implies that a known function V relates the variance and mean of y_i via $\text{var}(y_i) = V(\mu_i)\phi$. In a likelihood framework GLMs are usually estimated by a version of Newton's method in which the expected Hessian of the log likelihood replaces the actual Hessian. This approach gives rise to the method of iteratively re-weighted least squares (IRLS).

IRLS: Starting from initial guesses at μ_i and hence $\eta_i[\equiv \mathbf{X}_i\boldsymbol{\beta} \equiv g(\mu_i)]$, iterate the following steps to convergence:

1. Using the current μ_i, η_i estimates, evaluate pseudodata $z_i = g'(\mu_i)(y_i - \mu_i) + \eta_i$ and iterative weights $w_i = g'(\mu_i)^{-1}V(\mu_i)^{-0.5}$.
2. Find $\hat{\boldsymbol{\beta}}$ to minimize $\|\text{diag}(\mathbf{w})(\mathbf{z} - \mathbf{X}\boldsymbol{\beta})\|^2$. Obtain the next estimate of $\boldsymbol{\eta}$ as $\mathbf{X}\hat{\boldsymbol{\beta}}$, and hence the next $\boldsymbol{\mu}$ estimate.

At convergence the final $\hat{\boldsymbol{\beta}}$ maximizes the likelihood of the model (which is equivalent to minimizing the *deviance* defined as $D = 2\{l_s - l(\hat{\boldsymbol{\beta}})\}$, where l_s is the saturated log-likelihood for the data, and $l(\hat{\boldsymbol{\beta}})$ is the actual model log-likelihood).[1]

1. If $\mathbf{X}$ is $n \times p$, what is the leading order operations count for the IRLS algorithm?
2. Would you recommend solving the weighted least squares problem at step 2 of the IRLS by a Choleski related method or by a QR decomposition method? Explain.
3. When working with very large datasets (e.g. remote sensing data or gene expression data) and complicated model structures, the computational cost and storage requirements of fitting by IRLS can become burdensome. It would be advantageous to find fitting algorithms which had a lower leading order cost than IRLS, and required only the evaluation of $\mathbf{X}^\mathsf{T}\mathbf{x}$ for some $n-$vectors, $\mathbf{x}$. There are two obvious approaches.
   (a) At each step of the Newton iteration the expected Hessian of the negative log likelihood $\mathbf{X}^\mathsf{T}\text{diag}(\mathbf{w})^2\mathbf{X}$ could be replaced by an estimate, $n\tilde{\mathbf{X}}\widetilde{\text{diag}}(\tilde{\mathbf{w}})^2\tilde{\mathbf{X}}/M$, where $\tilde{\mathbf{X}}$ is a random sample of M rows of $\mathbf{X}$ and $\tilde{\mathbf{w}}$ is the corresponding set of rows of $\mathbf{w}$. The full gradient vector of the negative log likelihood is left un-modified at $-\mathbf{X}^\mathsf{T}(\mathbf{y} - \boldsymbol{\mu})/\{g'(\boldsymbol{\mu})V(\boldsymbol{\mu})\}$. Of course M should be much less than n.
   (b) The Newton iteration could be replaced by a quasi-Newton iteration.

[1] Chapter 2 of Wood (2006) *Generalized Additive Models:An Introduction with R*, or McCullagh and Nelder (1989) *Generalized Linear Models* contain more detailed information on GLMs.

How would the cost of these two approaches depend on n and p? Why do we expect (a) to work despite having only a rough approximation to the expected Hessian? Would using $\tilde{\mathbf{X}}$ in place of $\mathbf{X}$ in the gradient vector be likely to work? Suggest how you might find a starting Hessian for approach (b).

4. By modifying the routine `simple.glm.fit`, given below, implement method (a) from part 3, in R ($M = 1000$ is ok). Comment your routine properly, and test it (one test example is given below).

5. By modifying the routine `simple.glm.fit`, given below, implement method (b) from part 3, in R. Again, comment your routine properly, and test it.

6. Comment on the relative merits of the 3 approaches in terms of efficiency, stability and reliability.

```
simple.glm.fit <- function(y,X,family,tol=1e-7) {
## y is response, X is model matrix, family is R family
## (see documentation for ‘glm’ for further info on family
## objects). tol is convergence tolerance.
  nobs <- length(y)
  variance <- family$variance     ## extract variance function
  dev.resids <- family$dev.resids ## function to get deviance (resids)
  linkinv <- family$linkinv       ## inverse link function
  mu.eta <- family$mu.eta         ## reciprocal of link derivative
  eval(family$initialize)         ## gets initial mu - mustart, using y
  mu <- mustart;eta <- family$linkfun(mu)
  dev0 <- sum(dev.resids(y,mu,rep(1,nobs)))
  for (iter in 1:200) { ## main IRLS loop
    mu.eta.val <- mu.eta(eta)       ## reciprocal of link derivative
    z <- eta + (y - mu)/mu.eta.val       ## pseudodata
    w <- sqrt(mu.eta.val^2/variance(mu)) ## iterative (sqrt) weights
    ## solve weighted least squares problem
    Wz <- w*z;WX <- w*X
    beta <- coef(lm(Wz~WX-1))  ## could do direct QR thing here
    ## Try out step...
    d.eta <- as.numeric(X%*%beta) - eta ## the trial linear predictor step
    ok <- FALSE
    while(!ok) {
      mu <- linkinv(eta+d.eta) ## try step
      dev1 <- sum(dev.resids(y,mu,rep(1,nobs)))
      if (iter >1&&(dev1>dev0)) { ## deviance increased
        d.eta <- d.eta/2          ## so step half
        beta <- beta.old + (beta-beta.old)/2 ## apply to beta too
      } else ok <- TRUE
    }
    ## step now ok....
    eta <- eta + d.eta ## accepted linear predictor, eta
    if (abs(dev0-dev1)<tol*dev0&&iter>2) break ## convergence?
    dev0 <- dev1
    beta.old <- beta
  }
  beta ## return coeffs
} ## end of simple glm fitter

## Simple polynomial fitting example...

set.seed(1) ## simulate some data ...
n <- 100000;x <- runif(n)
f <- 0.2 * x^11 * (10 * (1 - x))^6 + 10 * (10 * x)^3 * (1 - x)^10
mu <- exp(f/5); y <- rpois(mu,mu)
X <- model.matrix(~poly(x,6))
system.time(b <- glm(y~X-1,family=poisson))
system.time(b3 <- simple.glm.fit(y,X,family=poisson()))
```