Generalized additive models for gigadata: modelling the UK black smoke network daily data. Appendices B and C

B Convergence properties

Here we show that the section 4.1 iteration is guaranteed to converge when $\mathbf{W}$ is independent of $\boldsymbol{\beta}$ (e.g. for a Gaussian with identity link, gamma with log link or Poisson with square root link). The notation follows sections 3 and 4.

Theorem 1. *Let* $V(\boldsymbol{\beta}, \boldsymbol{\lambda}) = D(\boldsymbol{\beta})/\phi + \boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}_\lambda\boldsymbol{\beta} + \log|\mathbf{X}^{\mathsf{T}}\mathbf{W}\mathbf{X}/\phi + \mathbf{S}_\lambda| - \log|\mathbf{S}_\lambda|_+$ *have a single (minimum) turning point and* $\mathbf{W}$ *be independent of* $\boldsymbol{\beta}$*. Then the iteration of section 4.1 converges to the minimum of* V*.*

Proof. Under the stated assumptions $\partial V/\partial\boldsymbol{\beta}$ coincides with the partial derivative of the penalized least squares objective minimized to compute $\hat{\boldsymbol{\beta}}_\lambda$. Since the step, $\Delta\boldsymbol{\beta}$ to $\hat{\boldsymbol{\beta}}_\lambda$ is hence minus the product of a positive definite matrix with $\partial V/\partial\boldsymbol{\beta}$, $\Delta\boldsymbol{\beta}$ is a descent direction for V (see e.g. Wood, 2015, §5.1.1). Since V only depends on $\boldsymbol{\beta}$ via $D(\boldsymbol{\beta})/\phi + \boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}_\lambda\boldsymbol{\beta}$, then step 3 of the section 4.1 iteration is equivalent to testing that V has decreased. Furthermore it is readily checked that $\mathrm{d}\mathcal{V}/\mathrm{d}\boldsymbol{\rho}$ given in (8) coincides with $\partial V/\partial\boldsymbol{\rho}$. Since $\boldsymbol{\Delta}$ is again given by minus the product of a positive definite matrix and $\partial V/\partial\boldsymbol{\rho}$, it is therefore a descent direction for V. Given the coincidence of derivatives, step 7 of the section 4.1 iteration is equivalent to applying the same test to V, which will guarantee reduction of V if it has a single minimum (and will often do so when it is multimodal). Hence each step of the algorithm is guaranteed to reduce V until its minimum is reached. □

Obviously the strong assumption that $\mathbf{W}$ is independent of $\boldsymbol{\beta}$ is only sufficient, rather than necessary, for convergence. The proof suggests that convergence should occur whenever the derivatives of $D(\boldsymbol{\beta})/\phi + \boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}_\lambda\boldsymbol{\beta}$ and V are 'close', but does not preclude convergence when this is not the case. Note that V is a version of the Laplace approximate marginal likelihood (e.g. Wood, 2011) but with the Hessian replaced by its expected value.

Practically the section 4.1 iteration typically takes around the same number of steps as a performance iteration in which the working REML score is fully optimized at each step: that is 5-20 for most cases but often around 10-40 for binary data. Since both the $\boldsymbol{\beta}$ updates and $\boldsymbol{\rho}$ updates are based on Newton methods, which tend towards quadratic convergence, there is rather little dependence of iteration number on tolerance, at least below the relative tolerances of around 10^{-6} used here (and little to be gained by loosening that tolerance). Small sample sizes *can* promote increased numbers of iterations (in both the old and new methods), as any identifiability/collinearity problems are then at their most acute, while the quadratic approximation on which each Newton step is based can also be poor on the scale of the Newton step at small sample sizes. There is little reason to expect sensitivity of the number of fitting steps to the basis dimensions used, and simulation experiments tend to back this up.

C Multi-core computing

Several computational details cannot be ignored in multi-core computing, if algorithms are to scale well with the number of cores used. These are briefly discussed here. We consider only the case of shared memory multi-core machines — that is the kind of architecture typical of a server or desktop workstation with 2 to 100 cores, rather than the distributed memory architecture of a supercomputer or GPU.

1. Current computers are memory bandwidth limited. While a floating point operation may take only 1 or 2 core cycles, retrieving the data on which to perform the operation may take 10 times as

long. In consequence CPUs have a small amount of very fast access cache memory available as a buffer between main memory and the CPU cores. There are big performance gains to be had from structuring algorithms to re-use data already in cache. For numerical linear algebra this means constructing algorithms in a block oriented manner, where most floating point work consists of matrix-matrix multiplication. To see why this matter consider two operations with equal floating point operation count: Forming $\mathbf{Ay}$ where $\mathbf{A}$ is 1000×1000 and $\mathbf{y}$ is a vector, and forming $\mathbf{BC}$ where $\mathbf{B}$ and $\mathbf{C}$ are both 100×100 matrices. The former involves no re-use of the elements of $\mathbf{A}$, whereas the latter involves multiple re-use of the elements of $\mathbf{B}$ and $\mathbf{C}$.

2. Many modern CPUs use some form of 'Hyper-threading' where each physical core appears to the operating system as 2 cores. The idea is that two programme threads will often be performing tasks using different parts of the core, and hence can run at the same time on the same core. In floating point intensive computations all threads spend most of their time using the floating point unit, and there is no performance gain, but rather a loss, from hyper-threading. It is therefore usually better to disable it.

3. A multi-core CPU can run at a higher speed without overheating when only a few cores are in use, as opposed to all being in use. Most CPUs exploit this and run faster under low core loading. In consequence using 10 cores of a 10 core CPU can never be 10 times as fast as using one core, even with perfectly scalable code.

4. Under high energy efficiency settings it is possible for a thread doing relatively little work to fail to increase the speed of its core, relative to cores running more intensive threads. This can lead to the paradoxical situation in which the low work thread actually becomes the rate limiting one. Setting the CPU control policy to favour high performance will remove the problem (note that for modern CPUs the core speed is controlled in hardware and no longer directly by the operating system).

A further complication is the advent of non-uniform memory access (NUMA). When a machine has multiple CPUs (each with multiple cores), then memory is often arranged in blocks associated with a CPU. A CPU can still access all memory, but access is fastest from its associated block. In the work reported here we were unable to make good use of NUMA.

References

Wood, S. N. (2011). Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 73*(1), 3–36.

Wood, S. N. (2015). *Core Statistics*. Cambridge University Press.