7/28/'10

UseR! The R User Conference 2010

An Algorithm for Unconstrained Quadratically Penalized Convex Optimization (post conference version)

Steven P. Ellis
New York State Psychiatric Institute
at Columbia University

PROBLEM

Minimize functions of form

$$F(h) = V(h) + Q(h), \quad h \in \mathbb{R}^d,$$

1. ($\mathbb{R} =$ reals; $d =$ positive integer.)
2. V is non-negative and convex.
3. V *is computationally expensive.*
4. Q is known, strictly convex, and quadratic.
5. *(Unconstrained* optimization problem)
6. Gradient, but not necessarily Hessian are available.

NONPARAMETRIC FUNCTION ESTIMATION

- Need to minimize:

$$F(h) = V(h) + \lambda h^T \mathbf{Q} h.$$

  - $\lambda > 0$ is "complexity parameter".

WAR STORY

- Work on a kernel-based survival analysis algorithm lead me to work on this optimization problem.
- At first I used BFGS, but it was very slow.
  - (Broyden, '70; Fletcher, '70; Goldfarb, '70; Shanno, '70)
  - Once I waited 19 hours for it to converge!
- Finding no software for unconstrained convex optimization (see below), I invented my own.

SOFTWARE FOR UNCONSTRAINED CONVEX OPTIMIZATION

Didn't find such software.

- CVX (`http://cvxr.com/cvx/`) is a Matlab-based modeling system for convex optimization.
  - But a developer, Michael Grant, says that CVX wasn't designed for problems such as my survival analysis problem.

“QQMM”

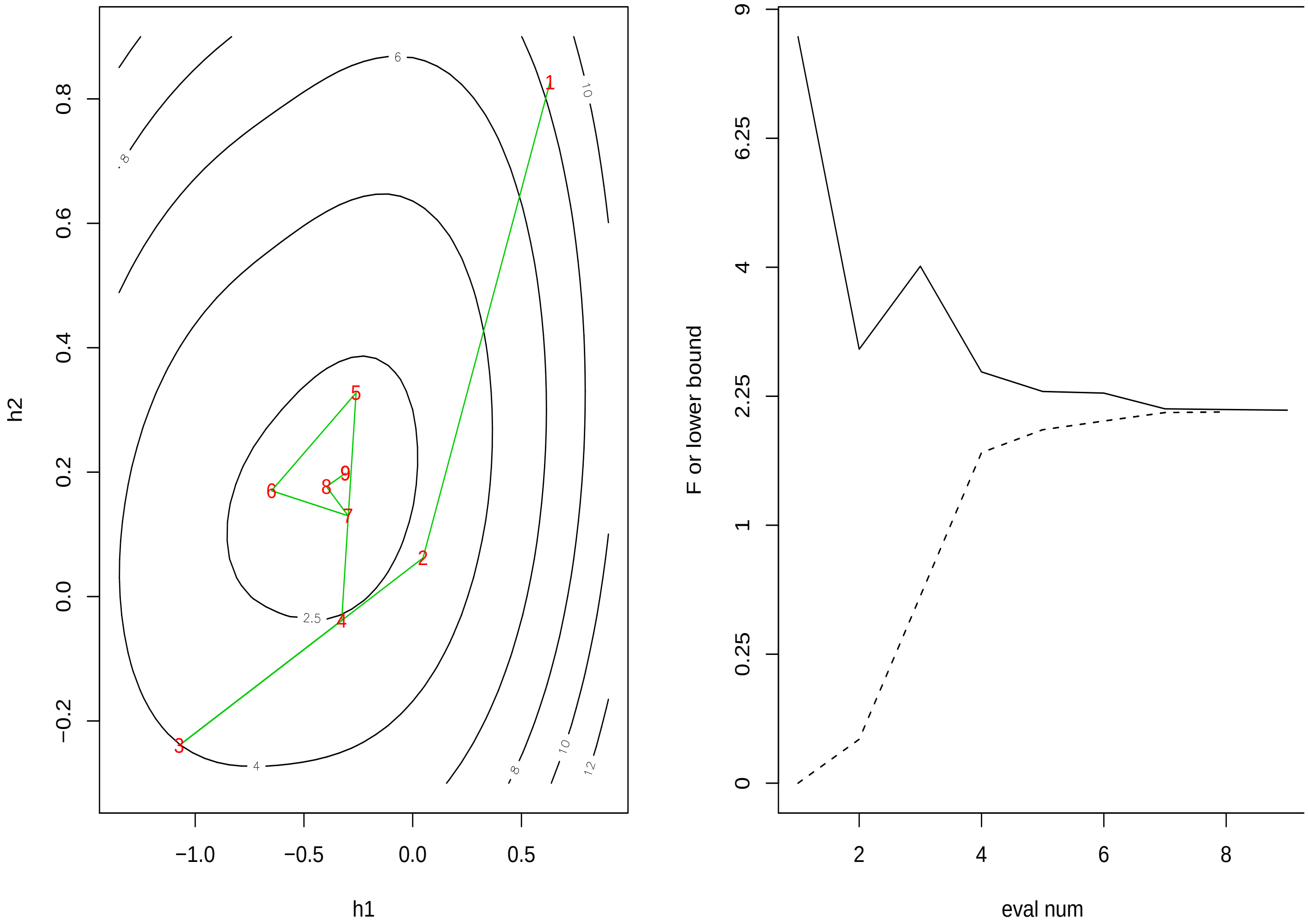
- Developed algorithm “QQMM” (“quasi-quadratic minimization with memory”; Q^2M^2) to solve problems of this type.
  - Implemented in *R*.
  - Posted on STATLIB.

Iterative descent method

- An iteration: If $h_1 \in \mathbb{R}^d$ has smallest F value found so far, compute one or more trial minimizers, h_2, until "sufficient decrease" is achieved.

- Assign $h_2 \to h_1$ to finish iteration.

- Repeat until evaluation limit is exceeded or stopping criteria are met.

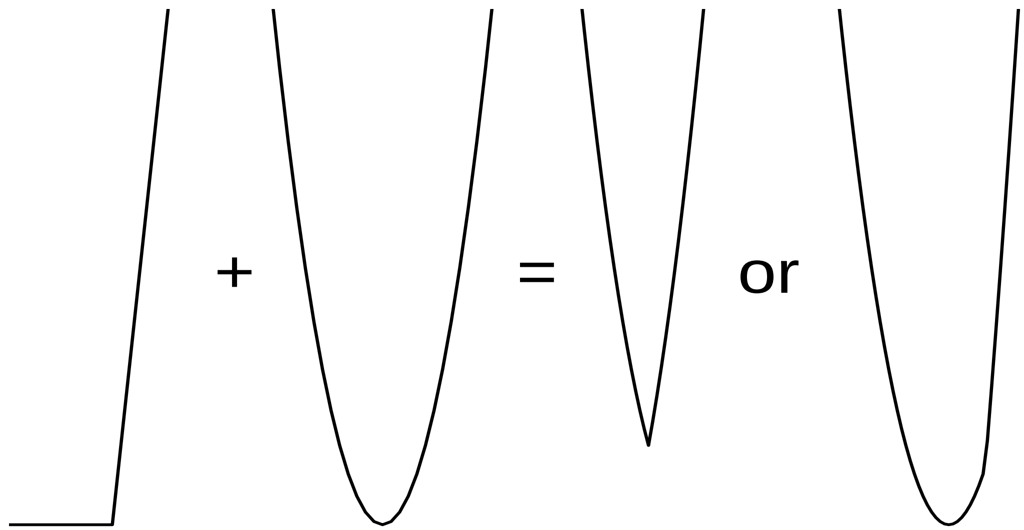


h2
h1
F or lower bound
eval num

CONVEX GLOBAL UNDERESTIMATORS

- If $h \in \mathbb{R}^d$, define a "quasi-quadratic function":

$$q_h(g) = \max\{V(h)+\nabla V(h)\cdot(g-h),\ 0\}+Q(h), \quad g \in \mathbb{R}^d$$


+
=
or

- q_h *is a convex "global underestimator" of* F*:*

$$q_h \le F.$$

- Possible trial minimand of F is the point h_2 where q_h is minimum, but that doesn't work very well.

L.U.B.'S

- If $h_{(1)}, \dots h_{(n)} \in \mathbb{R}^d$ are points visited by algorithm so far, the *least upper bound (l.u.b.)* of

$$q_{h_{(1)}}, q_{h_{(2)}}, \dots, q_{h_{(n-1)}}, q_{h_{(n)}}$$

  is their pointwise maximum:

$$F_n(h) = \max\{q_{h_{(1)}}(h), q_{h_{(2)}}(h), \dots, q_{h_{(n-1)}}(h), q_{h_{(n)}}(h)\},$$

- F_n is also a convex global underestimator of F no smaller than any $q_{h_{(i)}}$.

- The point, h_2 where F_n is minimum is probably a good trial minimizer.

- But minimizing F_n may be at least as hard as minimizing F!

- As a compromise, proceed as follows.
  - Let $h_1 = h_{(n)}$ be best trial minimizer found so far and let $h_{(1)}, \ldots h_{(n)} \in \mathbb{R}$ be points visited by algorithm so far.
- - For $i = 1, 2, \ldots, n-1$ let $q_{h_{(i)},h_1}$ be l.u.b. of $q_{h_{(i)}}$ and q_{h_1}.
    - "q double h"
    - Convex global underestimator of F.
    - Easy to minimize in closed form.

- Let $i = j$ be index in $\{1, 2, \ldots, n-1\}$ such that *minimum* value of $q_{h_{(i)},h_1}$ is *largest*.
  - I.e., no smaller than minimum value of any $q_{h_{(i)},h_1}$ $(i = 1, \ldots, n-1)$.
  - So $q_{h_{(j)},h_1}$ has a "maximin" property.
- *Let h_2 be vector at which $q_{h_{(j)},h_1}$ achieves its* minimum.
- (Actual method is slightly more careful than this.)
- If h_2 isn't better than current position, h_1, back-track.

Minimizing $q_{h_{(i)},h_1}$ *requires matrix operations.*

- Limits size of problems for which Q^2M^2 can be used to no more than, say, 4 or 5 thousand variables.

STOPPING RULE

- Trial values h_2 are minima of nonnegative global underestimators of F.

- Values of these global underestimators at corresponding h_2's are lower bounds on $\min F$.

- Store cumulative maxima of these lower bounds.

  - Let L denote current value of cumulative maximum.

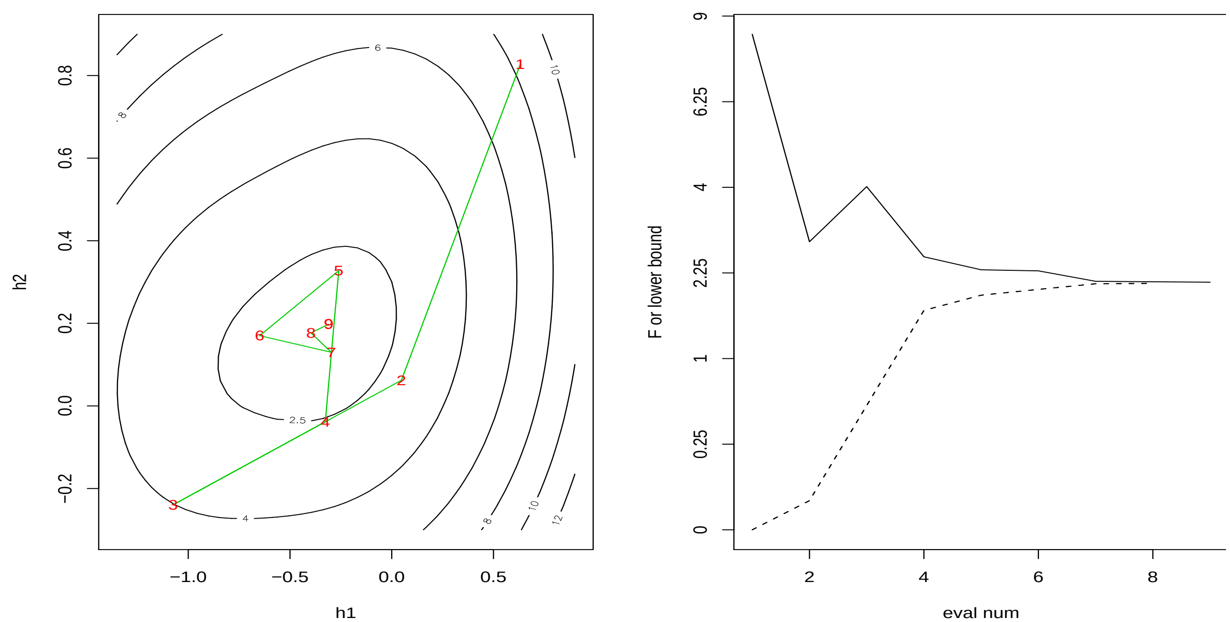
  - L is a lower bound on $\min F$.

- If h_1 is current best trial minimizer, relative difference between $F(h_1)$ and L exceeds relative difference between $F(h_1)$ and $\min F$.

$$\frac{F(h_1) - L}{L} \geq \frac{F(h_1) - \min F}{\min F}.$$

- I.e., *we can explicitly bound relative error in* $F(h_1)$ *as an estimate of* $\min F$*!*

- Choose small $\epsilon > 0$.
  - I often take $\epsilon = 0.01$.
- When upper bound on relative error first falls below threshold ϵ, STOP.
- Call this "convergence".
- Upon convergence you're guaranteed to be within ϵ of the bottom.



h2
h1
0.8
0.6
0.4
0.2
0.0
−0.2
−1.0
−0.5
0.0
0.5
6
4
2.5
8
10
12
1
2
3
4
5
6
7
8
9
F or lower bound
eval num
9
6.25
4
2.25
1
0.25
0
2
4
6
8

- Gives good control over stopping.
- That is important because . . .

STOPPING EARLY MAKES SENSE IN STATISTICAL ESTIMATION

- In statistical estimation, the function, F, depends, through V, on noisy data so:

- *In statistical estimation there's no point in taking time to achieve great accuracy in optimization.*

-

$$F(h) = V(h) + Q(h), \quad h \in \mathbb{R}^d,$$

Q^2M^2 IS SOMETIMES SLOW

- Q^2M^2 tends to close in on minimum rapidly.
- But sometimes is very slow to converge.
  - E.g., when Q is nearly singular.
  - E.g., when complexity parameter, λ, is small.
- Distribution of number of evaluations needed for convergence has long right hand tail as you vary over optimization problems.

SIMULATIONS: “PHILOSOPHY”

- If F is computationally expensive then simulations are unworkable.

- A short computation time for optimizations is desired.

- When F is computationally expensive then computation time is roughly proportional to number of function evaluations.

- Simulate computationally cheap F’s, but track *number of evaluations* not computation time.

COMPARE Q^2M^2 AND BFGS.

- Why BFGS?
  - BFGS is widely used.
    * “Default” method
  - Like Q^2M^2, BFGS uses gradient and employs vector-matrix operations.
  - Optimization maven at NYU (Michael Overton) suggested it as comparator!

– (Specifically, the “BFGS” option in the R function `optim` that was used. John C. Nash – personal communication – pointed out at the conference that other algorithms called “BFGS” are faster than the BFGS in `optim`.)

SIMULATION STRUCTURE

- I chose several relevant estimation problems.
- For each of variety of choices of complexity parameter λ use both Q^2M^2 and BFGS to fit model to randomly generated training sample and test data.
  - Either simulated data or real data randomly split into two subsamples.
- Repeat 1000 times for each choice of λ.
- Gather numbers of evaluations required and other statistics describing simulations.

- Use

  Gibbons, Olkin, Sobel ('99) *Selecting and Ordering Populations: A New Statistical Methodology*

  to select the range of λ values that, with 95% confidence, contains λ with lowest mean test error.

- Conservative to use largest λ in selected group.

SIMULATIONS: SUMMARY

- $L^{3/2}$ kernel-based regression.
  - For largest selected λ values, BFGS required 3 times as many evaluations compared to Q^2M^2.
- Penalized logistic regression: Wisconsin Breast Cancer data
  - University of California, Irvine Machine Learning Repository
  - For largest selected λ values, BFGS required 2.7 times as many evaluations compared to Q^2M^2.

- Penalized logistic regression: R's "Titanic" data.
  - For largest selected λ values, Q^2M^2required nearly twice as many evaluations compared to BFGS.
  - I.e., this time BFGS was better.
  - Hardly any penalization was required: Selected λ's were small.

SURVIVAL ANALYSIS AGAIN

- On a real data set with good choice of λ, Q^2M^2 optimized my survival analysis penalized risk function in 23 minutes.

- BFGS took:

  - 3.4 times longer with "oracle" telling BFGS when to stop.

  - 6.5 times longer without "oracle".

  - "Oracle" means using information from the "6.5 without oracle" and Q^2M^2 runs to select the number of interations at which BFGS achieves the same level of accuracy as does Q^2M^2.

CONCLUSIONS

- QQMM (Q^2M^2) is an algorithm for minimizing convex functions of form

$$F(h) = V(h) + Q(h), \quad h \in \mathbb{R}^d.$$

  - V is convex and non-negative.
  - Q is known, quadratic, strictly convex.
  - Q^2M^2 is especially appropriate when V is expensive to compute.

- Allows good control of stopping.

- Needs (sub)gradient.

- Utilizes matrix algebra. This limits maximum size of problems to no more than 4 or 5 thousand variables.

- Q^2M^2 is often quite fast, but can be slow if Q is nearly singular.