Accelerating Fixed Point Algorithms with Many Parameters

Michael Karsh

UCLA Department of Statistics

November 17, 2011
Introduction

▸ Purpose of this Dissertation:
▸ Evaluate Convergence Acceleration Methods on Dataset With Large Number of Parameters

▸ Motivation:
▸ EM Algorithm Slow on London Deaths Data
▸ Try Convergence Acceleration on a Genetic Dataset which will have a Large Number of Parameters
Terms Key to This Dissertation

- Fixed Point $x$ of Function $F$
- Point satisfying $x = F(x)$
- Fixed Point Algorithm
- $x_{n+1} = F(x_n)$
Point of Attraction

- Point \( x_\infty \) such that if \( x_\infty \in D \), there will be \( S \subset D \) such that if \( x_n \in S, x_{n+1} \in D \)

- \( \lim_{n \to \infty} x_n = x_\infty \)

- If function continuous, point of attraction is fixed point of function
Optimization

- Maximize or Minimize $f$
- Set $f'$ equal to 0
- Find fixed point of $G(x) = x - A(x)f'(x)$ for invertible matrix $A$
Newton and Scoring

- Newton: Find fixed point of $G(x) = x - (f''(x))^{-1}f'(x)$
- Scoring: Find fixed point of $G(x) = x + (E(f''(x)))^{-1}f'(x)$
- Scoring: Find fixed point of $G(x) = x - (E(f'(x)f'(x)))^{-1}f'(x)$
Application to Nonlinear Least Squares

- Let $h_i$ predict $y_i$ based on $x$
- Let $z \approx x$
- Find fixed point of $G(x) \approx x - 4(\sum_i h'_i(z))^{-1}\sum_i (h'_i(z)(x - z))$
Application to Iteratively Reweighted Least Squares

- Let $A$ be a matrix which when multiplied by $x$ approximates $y$
- Let $W$ be a matrix which may weight different errors differently
- Find fixed point of $\arg\min_x (y^{(k)} - Ax)^T W(x^{(k)}) (y^{(k)} - Ax)$
Minorization Maximization

- Statisticians generally want to maximize likelihood
- Minorization: Choose $g$ such that $g(x_n|x_n) = f(x_n)$ and for every $x \ g(x|x_n) \leq f(x)$
- Maximization: Set $x_{n+1} = \text{arg max}_x g(x|x_n)$
Majorization Minimization

- Statisticians generally want to minimize sums of squared errors.
- Majorization: Choose \( g \) such that \( g(x_n|x_n) = f(x_n) \) and for every \( x \), \( g(x|x_n) \geq f(x) \).
- Minimization: Set \( x_{n+1} = \arg\min_x g(x|x_n) \).
EM Algorithm: Minorization to Maximize Likelihood

- **Minorization: E-step:** $Q(x, x_n) = E(\ln f(x|x_n))$
- **Maximization: M-step:** $x_{n+1} = \arg\max_x Q(x, x_n)$
Iterative Proportional Fitting: Minorization to Maximize Likelihood

- Minorization of Likelihood Given Column Entries: Divide Current Row Sums by Desired Row Sums and Multiply this Result by Row Entries
- Minorization of Likelihood Given Row Entries: Divide Current Column Sums by Desired Column Sums and Multiply this Result by Column Entries
- Maximization of Likelihood: Repeat Procedure Until Obtain Desired Row and Column Entries
Multidimensional Scaling: Majorization to Minimize Sums of
Squares

- Given dissimilarities $\delta_{i,j}$ between points $i$ and $j$ and weights $w_{i,j}$ of errors $i,j$
- Choose distances to $d_{i,j}$ to minimize $\sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} (\delta_{i,j} - d_{i,j})^2$
- Majorization: $\sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} \delta_{i,j}^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} (d_{i,j} | d_{i,j,k})^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} \delta_{i,j} (d_{i,j} | d_{i,j,k})$
- Minimization: $d_{i,j,k+1} = \arg \min_{d_{i,j}} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} \delta_{i,j}^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} (d_{i,j} | d_{i,j,k})^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} \delta_{i,j} (d_{i,j} | d_{i,j,k})$
Block Relaxation

- Each iteration takes a number of steps equal to the number of parameters instead of just 1 step as Newton and Scoring do or just 2 steps as Majorization Minimization and Minorization Maximization do.

- Maximize or Minimize Function with respect to 1 parameter at a time holding all other parameters constant.
Example of Block Relaxation: Alternating Least Squares

- Model Response Variables Based on Explanatory Variables
- Model Explanatory Variables Based on Response Variables
- Repeat This Process
Example of Block Relaxation: Coordinate Descent

- Two Types: Free Steering and Cyclic
- Free Steering: Select One Possible Update For All Coordinates Before Going Onto Next Set of Updates
- Cyclic: Update One Coordinate At A Time While Holding Values of All Other Coordinates Constant
Definitions

- Uniformly Compact: A Map Mapping the Whole Space to a Compact Subset of the Space
- Upper Semicontinuous (Closed): Pick a Set of Points Converging to a Limit. Pick Points from their Images under the Map such that these Points have a Limit. Then this Limit is in the Image of the Limit of the original Points under the Map.
- To Find Desirable Points, if a Point is Desirable, Stop. Otherwise Pick Point from Image of Current Point under Map. Repeat Until Desirable Point.
Zangwill’s Theorem

- Zangwill: If a map is uniformly compact and upper semicontinuous and the real-valued evaluation function is less for each point in the image of the original point than it is for the original point, then all limit points of the mapping process are desirable points.

- Meyer: If the real-valued evaluation function is less for each point in the image of the original point than it is for the original point, then successive points from the mapping process get closer and closer to each other.
Assume Map is Differentiable at Fixed Point

If Derivative has Absolute Value Between 0 and 1, Convergence Linear

If Derivative has Absolute Value 1, Convergence Sublinear

If Derivative has Absolute Value 0, Convergence Superlinear

Newton’s Method, If It Converges, Does So Superlinearly, (In Fact It Does So Quadratically)

EM Algorithm and Alternating Least Squares Converge Linearly
Long vs. Short Sequences

- While it is possible to transform a long sequence into another long sequence, what is far more useful is to transform a short sequence into another short sequence.

- One sequence transformation that does this is Aitken’s $\Delta^2$:

$$y_n = \frac{x_n x_{n+2} - x_{x+1}^2}{x_{n+2} - 2x_{n+1} + x_n}$$
Definitions Key to Understanding Convergence Acceleration

- **Rate of Convergence**: \( \lim_{n \to \infty} \frac{||x_{n+1} - x^*||}{||x_n - x^*||} \)

- **Accelerate Convergence**: transform sequence to sequence that converges faster

- **Converge Faster**: \( \lim_{n \to \infty} \frac{||y_n - x^*||}{||x_n - x^*||} = 0 \)

- **Translative**: Adding constant to each member of sequence, each member of transformed sequence, and limit does not change limiting ratio.

- **Homogeneous**: Multiplying each member of sequence, each member of transformed sequence, and limit by constant does not change limiting ratio.

- **Quasi-Linear**: Translative and Homogeneous
Generalized Remanence

- Set of sequences all of which have the same limit such that:
  - No member of any sequence in the set equals the limit
  - All sequences are equal up to a point
  - Beyond this point all but one sequence are equal up to another point
  - Beyond this point all but two sequences are equal up to a third point
  - Beyond this point all but three are equal up to a fourth point
  - and so on.
- No sequence transformation can accelerate convergence of all sequences in set
- Set of all logarithmically convergent sequences satisfies generalized remanence
Evaluation of Sequence Transformation

- Synchronous Process: A sequence transformation with the same rate of convergence as the original sequence which over the long run is closer to converging than the original sequence by a constant.

- If set of sequences satisfies generalized remanence, goal for sequence transformation: synchronous process.

- Problem: limiting constant factor closer to convergence may not exist.

- Contractive sequence: Beyond certain iteration closer to converging by AT LEAST a certain constant factor.

- Goal with sequence transformation: either faster rate of convergence or synchronous process or contractive sequence.
Examples of Methods to Accelerate Convergence

- Epsilon Algorithms
- versions of Aitken’s $\Delta^2$
- Polynomial Methods
- Squared Polynomial Methods
- Compact Recursive Projection Algorithms
Epsilon Algorithms

- **Scalar Epsilon Algorithm:** $\varepsilon_0^{(n)} = 0$ $\varepsilon_0^{(n)} = s_n$

  $\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \frac{1}{\varepsilon_{k}^{(n+1)} - \varepsilon_{k}^{(n)}}$

- **Vector Epsilon Algorithm:** $\varepsilon_0^{(n)} = 0$ $\varepsilon_0^{(n)} = s_n$

  $\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \frac{\varepsilon_{k}^{(n+1)} - \varepsilon_{k}^{(n)}}{(\varepsilon_{k}^{(n+1)} - \varepsilon_{k}^{(n)}) \cdot (\varepsilon_{k}^{(n+1)} - \varepsilon_{k}^{(n)})}$

- **Topological Epsilon Algorithm:** $\varepsilon_0^{(n)} = 0$ $\varepsilon_0^{(n)} = s_n$

  $\varepsilon_{2k+1}^{(n)} = \varepsilon_{2k-1}^{(n+1)} + \frac{y}{y \cdot \Delta \varepsilon_{2k}} \varepsilon_{2k+2}^{(n)} = \varepsilon_{2k}^{(n+1)} + \frac{\Delta \varepsilon_{2k}^{(n)}}{\Delta \varepsilon_{2k+1} \cdot \Delta \varepsilon_{2k}}$
Aitken’s $\Delta^2$

- Ramsay shows how Aitken’s $\Delta^2$ can accelerate convergence by decelerating oscillations of sequences which alternate between being above and below the optimal value as well as by accelerating convergence of sequences which are consistently on one side of the optimal value.

- Scalar version:
  \[ y_n = \frac{x_{n+2}x_n-x_{n+1}^2}{x_{n+2}-2x_{n+1}+x_n} \]

- 1st vector version:
  \[ y_{i+2} = x_{i+2} + \frac{(x_{i+2}-x_{i+1}).(x_{i+2}-2x_{i+1}+x_i)}{||x_{i+2}-2x_{i+1}+x_i||^2} \]

- 2nd vector version:
  \[ y_{i+2} = x_{i+2} + \frac{(x_{i+2}-x_{i+1}).(x_{i+1}-x_i)(x_{i+2}-x_{i+1})}{(x_{i+1}-x_i)(x_{i+2}-2x_{i+1}+x_i)} \]

- 3rd vector version:
  \[ y_{i+2} = x_{i+2} + \frac{||x_{i+2}-x_{i+1}||(x_{i+2}-x_{i+1})}{||x_{i+2}-x_{i+1}|| - ||x_i+1-x_i||} \]
### Modified Minimal Polynomial Extrapolation (MMPE): \( y_i^n = y_{i+1} \)

### Minimal Polynomial Extrapolation (MPE): \( y_i^n = \Delta x_{n+i} \)

### Reduced Rank Extrapolation (RRE): \( y_i^n = \Delta^2 x_{n+i} \)
Squared Polynomial Methods

- Double coefficient to first difference to get new coefficient to first difference.
- Square coefficient to first difference to get coefficient to second difference and add this result.
- To get coefficient for hybrid method: take square root of product of coefficients to MPE and RRE and add square root of one minus ratio of RRE coefficient to MPE coefficient.
- Again double coefficient to multiply by first difference and subtract this result and square coefficient to multiply by second difference and add this result.
Compact Recursive Projection Algorithms

- **Main version:**
  \[ y_0^{(i)} = x_i \quad y_k^{(i)} = y_{k-1}^{(i)} - \frac{(z_k \cdot y_{k-1}^{(i)})y_{k-1}^{i+1}}{(z_k \cdot y_{k-1}^{(i+1)})} \]

- **First variant:**
  \[ y_0^{(i)} = x_i \quad y_k^{(i)} = y_{k-1}^{(i)} - \frac{(z_k \cdot y_{k-1}^{(i+1)})(y_k^{i} - y_{k-1}^{i+1})}{(z_k \cdot y_{k-1}^{(i)})} \]

- **Second variant:**
  \[ y_0^{(i)} = x_i \quad y_k^{(i)} = y_{k-1}^{(i)} - \frac{(z_k \cdot y_{k-1}^{(i+1)})y_{k-1}^{i+1}}{(z_k \cdot (y_{k-1}^{(i+1)} - y_{k-1}^{i}))} - y_{k-1}^{i+1} \]
Conversion of Scalar Methods to Matrix Methods

- Matrix Methods Take Into Account Relations Between Parameters Which Scalar Methods Do Not

- For This Reason We Want to Convert Scalar Methods to Matrix Methods

- To Do This Change Division to Multiplication by the Inverse of a Matrix

- Fast Way to Take Inverse of Matrix: Moore-Penrose Inverse

- Moore-Penrose Inverse:

\[
(I - V(U^T U)^{-1} U^T)^{-1} = I + V(U^T U - U^T V)^{-1} U^T
\]
Results on London Deaths Data

- unaccelerated EM takes 2517 iterations
- scalar methods (except for vector and topological epsilon algorithms which take 2587) also take 2517
- De Leeuw in linear article and the Haifa group achieve convergence in anywhere from 2 iterations for 4-step RRE to 56 iterations for 3-step RRE
- De Leeuw in linear article and the Haifa group do not achieve convergence at all using 2-step RRE or 4-step MPE
- Varadhan and Roland find only slight convergence acceleration using polynomial method but greater acceleration using squared polynomial methods ranging from 572 iterations for SqRRE to 244 iterations for SqMPE
- Moore-Penrose inverse matrix method accelerates convergence to 230 iterations
Results on Ekman Color Circle

- unaccelerated MDS takes 1260 iterations
- scalar methods (except for vector and topological epsilon algorithms which take 447) accelerate to 440
- De Leeuw in rate article achieves convergence in 519 iterations using the basic algorithm
- De Leeuw in rate article achieves convergence acceleration to between 90 and 270 iterations
- Moore-Penrose inverse matrix method accelerates convergence to 90 iterations
Results on Data with Equal Dissimilarities

- unaccelerated MDS takes 206 iterations
- scalar methods (except for vector and topological epsilon algorithms which take 217) also take 206
- De Leeuw in linear article and the Haifa group article achieves convergence acceleration to between 5 and 63 iterations
- De Leeuw in rate article accelerates convergence to between 8 and 58 iterations.
- Moore-Penrose inverse matrix method accelerates convergence to 36 iterations
Results on Morse Code Signals

- unaccelerated MDS takes 35 iterations
- scalar methods (except for epsilon algorithms which accelerate to 21) also take 35
- De Leeuw in rate article takes 1214 iterations using basic algorithm
- De Leeuw in rate article accelerates convergence to between 187 and 655 iterations.
- Moore-Penrose inverse matrix method takes 35 iterations
Conclusions

- If a set of sequences satisfies generalized remanence, no sequence transformation can accelerate convergence of all sequences in the set.
- This does not mean that no sequence transformation will ever accelerate convergence of any sequence in the set.
- Each of the previous examples may be in sets satisfying generalized remanence.
- In most cases scalar methods did not accelerate.
- In most cases matrix methods, such as those in De Leeuw’s papers and those using the Moore-Penrose Inverse accelerate convergence.
- This may be due to the fact that scalar methods do not take into account correlations between parameters, whereas matrix methods do.
Examples of Genetic Datasets

- 1000 Individuals of CEU, CHB, JPT, or YRI ancestry with 13262 SNP Markers
- 324 Individuals of CEU, YRI, MEX, ASW with 13298 SNP Markers
- 912 Individuals from New York City of Northwestern European, Southeastern European, or Ashkenazi Jewish Population with 9378 SNP Markers

For this dissertation I used two different datasets:

- A Raw Dataset from Affymetrix with 131036 Parameters
- A Pre-Processed Dataset David Alexander Used With 22000 Parameters
David Alexander’s Dataset

- Select 10000 SNP Markers
- Select 1000 Individuals
- Assume 2 Populations
- Base Selections on Allowing No More Than 5 Percent of Genotypes to be Missing
- Base Selections on Requiring Minimum 200 kilobase Pair Separation
- Missing Data Requires EM Which is Slow
- Block Relaxation Faster
Raw Dataset from Affymetrix

- 100 Genotypes Checked between this Dataset and David Alexander’s Dataset Same
- 99 Percent Confidence that at least 95 Percent are Same
- Data Consists of Following Items for each SNP Marker
  - Means, Variances, Covariances Between Reference and Sample for BB, AB, and AA
  - Fragment Length and Type (for one Type Fragment Length is Going to be 0 since the SNP Marker is Going to be of the other Type)
  - Which DNA Bases The Fragment Consists of (A, C, G, T)
Block Relaxation Algorithm for Processing Raw Data
(Continued on Two Slides After This One)

- Estimate Raw Effect of A Allele by normalizing difference between sample means for AA and AB. Do likewise for B allele.
- Take log base 2 of this normalization.
- Take the means of logs of normalizations to estimate raw copy number.
- Log-Sum Method: Take log base 2 of sums of normalizations.
- Sum-Log Method: Take the sum of the logs base 2 of normalizations.
- Obtain scaled effect by subtracting differences between means from raw effect and multiplying that by ratios of sample variances to reference variances.
- Obtain copy number differences by subtracting scaled effects from raw effects.
- Model copy number differences on fragment lengths.
- Re-estimate means by multiplying scaled effects by ratios of reference variances and dividing that by ratios of sample variances. Subtract this result from raw effect. Add AB sample mean.
Procedure for Estimating Copy Number States

- Possible Copy Number States are 0 for homozygous deletion, 1 for hemizygous deletion, 2 for no change, 3 for single amplification, 4 for multiple amplification
- These are categories so they have to be discrete
- For both Log-Sum and Sum-Log Methods:
  - Raise 2 to Results for Log-Sum and Sum-Log Methods
  - Round this result to Nearest Integer Between 0 and 4
Determining Decay Rates

- Determine Probability of Going from One Copy Number State to Another on Successive SNP Markers
- Put These Probabilities into a Posterior Stochastic Matrix
- Have a Prior Stochastic Matrix assuming a 96 Percent Probability of Staying in the Same State and a 1 Percent Probability of Changing States
- Estimate Decay Amount by Multiplying inverse of difference between Identity matrix and Prior Probability Matrix by Difference between Posterior and Prior Probability Matrices
- Estimate Decay Rate by Multiplying log of Posterior Transition Probability by Inverse of Distances Between States
- Re-Estimate Decay Amounts by Raising e to Products Decay Rates and Distances Between States
- Re-Estimate Posterior Transition Matrix by Adding Decay Amounts to the Product of the Prior Transition Matrix and the Difference Between the Identity Matrix and the Decay Amount
Results of Scalar Methods

- The Algorithm Described on the Previous Three Slides Converges in 298 Iterations on Raw Data
- Results of Scalar Methods
- Epsilon Algorithms Run Out of Memory
- All Other Scalar Methods Run Into Either Stagnation or Near Breakdown
- Since Varadhan and Roland Successfully Combine A Method They Find Resulting in Near Breakdown with a Method They Find Resulting in Stagnation, I decided to Try This
- Did not Work Still Resulted in Stagnation or Near Breakdown
- This Dataset is Unsuitable for Scalar Methods Since Many Parameters Categorical (Can Only Take On Discrete Sets of Values)
Results of Matrix Methods

- Run out of Memory on Raw Data
- Results On David Alexander’s Processed Data
- Most Scalar Methods take 298 Iterations to Converge
- Moore-Penrose subsumes any discrete variable into a continuous variable without compromising the integrity of the data
- Most Matrix Methods using Moore-Penrose Inverse take 50 Iterations to Converge
Conclusions

- Datasets in which some parameters can only take on discrete values cannot be accelerated by scalar methods.
- Genetic data highly correlated so needs matrix methods.
- Matrix methods run out of memory if dataset too large.
- As a possible future project:
  - Figure ways to use memory more efficiently in matrix methods.
  - Reduce size of dataset either via random sampling or as David Alexander did.
- For exceptionally large datasets it might be possible to use distributed processing across several computers or some other strategy, such as using more powerful machines to resolve the memory limitations.