Damped Newton Algorithms for Matrix Factorization with Missing Data
Aaron M. Buchanan and Andrew W. Fitzgibbon
University of Oxford
{amb, awf}@robots.ox.ac.uk

Problem Definition

Matrix factorization is central to many computer vision problems, e.g., structure from motion [8], illumination based reconstruction [5], and non-rigid model tracking [3]. In each case, the measured data are observations of the elements of a measurement matrix, denoted \( \mathbf{M} \), which is of known rank \( r \). With perfect data, in the absence of noise, the desired quantities are two smaller matrix factors:

\[
\mathbf{M} = \mathbf{X} \mathbf{A} \mathbf{B}^T,
\]

(1)

In the presence of noise, \( \mathbf{M} \) will not be observable. The factors must be recovered from the noisy version, \( \hat{\mathbf{M}} \). Assuming that the noise is isotropic Gaussian, \( \mathbf{A} \) and \( \mathbf{B} \) are the minimizers of the error function

\[
f_{\text{M}}(\mathbf{A}, \mathbf{B}) = \| \hat{\mathbf{M}} - \mathbf{A} \mathbf{B}^T \|^2_F,
\]

(2)

for which there are reliable and globally convergent algorithms based on the SVD. In practice, some elements of \( \mathbf{M} \) may not be available, for example due to occlusions, tracking failures, shadows and specularities. A weight matrix, \( \mathbf{w} \), can be incorporated to account for this:

\[
e(\mathbf{A}, \mathbf{B}) = \sum_{ij} \mathbf{w}_{ij} \| (\hat{\mathbf{M}} - \mathbf{A} \mathbf{B}^T)[ij] \|^2,
\]

(3)

where \( \odot \) is the Hadamard product (\( \mathbf{R} = \mathbf{P} \odot \mathbf{Q} \Rightarrow r_{ij} = p_{ij}q_{ij} \)) and \( \lambda_1 \) and \( \lambda_2 \) are regularizers to deal with the difficulties of local minima (consider \( \mathbf{w} = 0 \) as an extreme example). Alternating closed-form solutions is a common strategy. Note that the problem of gauge freedom (for any invertible \( \mathbf{r} \times \mathbf{r} \) matrix \( \mathbf{G} \), \( \epsilon(\mathbf{A}, \mathbf{B}) = \epsilon(\mathbf{AG}, \mathbf{BG}^T) \)) is ignored by these approaches.

Key Message

Matrix factorization with missing data remains a problem with no good solution. The State of the Art (lower left of this poster) indicates the number of algorithms proposed in the literature, all of which are based on alternation. They all perform poorly on almost any real data. Experiments (see right) show that the number of runs required to find the global minimum is often excessive, and that convergence rates are commonly glacial (see Comparison Summary to the right). Damped Newton Algorithms are a better approach, converging quickly enough that multiple random restarts find the global minimum of real problems in reasonable time. Although better than the state of the art by some way, we note that this problem is yet to enjoy a satisfactory solution.

Damped Newton Algorithms

Vectorizing the elements of the two matrix factors, the error surface \( e(\mathbf{A}, \mathbf{B}) \) becomes the function \( e(x) \) of the state vector \( x \). The Newton method iteratively updates \( x \) with the \( \delta x \) that minimizes the Taylor-series approximation

\[
e(x + \delta x) \approx e(x) + \frac{\partial e(x)}{\partial x} \delta x + \frac{1}{2} \frac{\partial^2 e(x)}{\partial x^2} \delta x \delta x^T \Rightarrow \delta x = -\mathbf{H} \mathbf{d},
\]

(4)

with the gradient \( \mathbf{d} = \frac{\partial e}{\partial x} \) and a positive definite Hessian matrix, \( \mathbf{H} = \frac{\partial^2 e}{\partial x^2} \). The sparsity structure of an example Hessian matrix for this problem is shown on the left. The upper left and lower right blocks are generated by the pure derivatives and the off-diagonal blocks by the cross derivatives. This example was generated for \( r = 3 \) with 50% known data.

Unfortunately the error in (5) is quartic in \( x \) and so quadratic approximations can be very unsuitable. Adding a scaled identity matrix to \( \mathbf{H} \) stabilizes the iterative scheme and results in the damped Newton method, similar to the Levenberg-Marquardt algorithm. Further adaptations are possible:

- **Alternation-like Regularization.** In Matlab notation, add \( \lambda \) times \( \text{tr}((\mathbf{eye}(n + m)) \odot \mathbf{r} \otimes \mathbf{r} \otimes \mathbf{r}) \), \( \mathbf{H} \), rather than \( \mathbf{I} \), to \( \mathbf{H} \).
- **Damped Newton with Line Search.** Use a line search to find the minimum in the search direction, reducing matrix inversions.
- **Hybrid Methods.** Switch between the fast initial convergence of alternation and the global efficiency of damped Newton.

Comparison Summary

Many approaches in the literature use an alternation strategy, which, although guaranteed to minimize the error at every iteration, is prone to flatlining: requiring excessive numbers of iterations before convergence. On the other hand, Newton methods are expensive per iteration, initially converge slowly, and are fractionally more difficult to program, but on average require orders of magnitude fewer iterations to reach a solution.

Experiments

The damped Newton approach provides a more powerful and more flexible framework in which to build algorithms to factorize matrices when data are missing. The theory is well understood [9] and the incorporation of priors, which is vital for satisfactory results (see below), is significantly simpler.

Conclusion

The damped Newton algorithm provides a more powerful and more flexible method in which to build algorithms to factorize matrices when data are missing. The theory is well understood [9] and the incorporation of priors, which is vital for satisfactory results (see below), is significantly simpler.

Note: a smaller rms pixel error does not imply a better solution.


Algorithm: Alternation-like PowerFactorization. Column denote the lines used by each algorithm: 
\( \text{ALT} \) = alternation; \( \text{PF} \) = PowerFactorization; \( \text{SIR} \) = Shum et al. [7]; \( \text{HHH} \) = Huynh et al. [6]; \( \text{AFAC} \) = Arora et al. [1]; \( \text{CA} \) = Guerreiro and Aguiar [4]; \( \text{RDT} \) = Brandt [2]. Notation: \( x^* \) is the \( i \)th row and \( x^j \) the \( j \)th column of matrix \( x \), with both \( x^* \) and \( x^j \) as column vectors.