Adaptive Proximal Gradient Method for Constrained Matrix Factorization

Peter Melchior · Rémy Joseph · Fred Moolekamp

Abstract The Proximal Gradient Method (PGM) is a robust and efficient way to minimize the sum of a smooth convex function $f$ and a non-differentiable convex function $r$. It determines the sizes of gradient steps according to the Lipschitz constant of the gradient of $f$. For many problems in data analysis, the Lipschitz constants are expensive or impossible to compute analytically because they depend on details of the experimental setup and the noise properties of the data. Adaptive optimization methods like ADAGRAD choose step sizes according to on-the-fly estimates of the Hessian of $f$. As quasi-Newton methods, they generally outperform first-order gradient methods like PGM and adjust step sizes iteratively and with low computational cost. We propose an iterative proximal quasi-Newton algorithm, ADAPROX, that utilizes the adaptive schemes of ADAM and its variants (AMSGRAD, ADAMX, PADAM) and works for arbitrary proxable penalty functions $r$. In test cases for Constrained Matrix Factorization we demonstrate the advantages of ADAPROX in fidelity and performance over PGM, especially when factorization components are poorly balanced. The python implementation of the algorithm presented here is available as an open-source package at https://github.com/pmelchior/proxmin.

Keywords Optimization · Quasi-Newton · Proximal Algorithms · Non-negative Matrix Factorization

1 Introduction

Many problems in data analysis seek to minimize $f(x) + r(x)$, where $x \in \mathbb{R}^d$, $f$ is a smooth convex loss function with a Lipschitz-continuous gradient, and $r$ is a potentially non-convex function.
penalty function that regularizes the solution. For a large class of penalties it is beneficial to access them through their proximal operators (Moreau 1965)

\[ \text{prox}_{\alpha r}(x) \equiv \arg\min_z \left\{ r(z) + \frac{1}{2\alpha} \|z - x\|^2_2 \right\} \] (1)

because it enables a forward-backward scheme, where at iteration \( t \) a step in the direction of \( \nabla f \) is followed by the application of the proximal operator:

\[ x_{t+1} = \text{prox}_{\alpha t r}(x_t - \alpha_t \nabla f(x_t)) \]. (2)

If step size \( \alpha_t \in (0, 2/L) \) with \( L \) being the Lipschitz constant of \( \nabla f \), the sequence converges to the minimum of \( f + r \). This algorithm is known as Proximal Gradient Method (PGM, e.g. Parikh and Boyd 2014).

It is straightforward to compute the Lipschitz constants for simple problems, but more complex experimental designs can make that computation non-analytic or very expensive. As an example, consider the linear inverse problem with i.i.d. Gaussian errors,

\[ f(x) = \frac{1}{2} \|Ax - y\|^2_2 \] (3)

which has \( L = \|A^\top A\|_s \). In image analysis the matrix \( A \) typically encodes resampling and convolution operations, so that \( L \) may be expensive to compute. In Section 4, we are going to discuss the matrix factorization problem, where \( L \) needs to be computed at every iteration and may not even be known in closed form. We therefore seek a proximal gradient method whose step sizes can be set without invoking Lipschitz constants. Although adaptive step size schedules have been proposed for special cases (e.g. \( \ell_1 \)-penalized solutions, Loris et al 2009), a more efficient approach for several classes of regularizers replaces PGM by a proximal quasi-Newton scheme (Becker and Fadili 2012; Becker et al 2019). We seek a generalization of this approach that is fully agnostic about the proxable regularizer and that exploits the robustness and efficiency of the quasi-Newton optimizer ADAM (Kingma and Ba 2015).

2 Adaptive gradient methods

One possible approach for adaptive gradient updates amounts to replacing a simple gradient step with

\[ x_{t+1} = x_t - \alpha_t \frac{m_t}{\sqrt{v_t}} \] (4)

where \( \alpha_t \) are externally provided step sizes, potentially varying at every step \( t \); and \( m_t \) and \( v_t \) are estimates of the mean and variance of \( g \equiv \nabla f(x) \), respectively. This scheme has two effects: it adjusts the step size for every dimension, and it renders the updates steps independent of the actual amplitude of \( g \), effectively sidestepping the problem of having to compute Lipschitz constants.

ADAGrad (Duchi et al 2011), one of the first algorithms to use the scheme, was designed for online optimization with sparse gradients and therefore sums up \( g^2 \) from all previous iterations as \( v_t \), RMSProp (Hinton et al 2012) and ADAM maintain the general form of

\[ \text{In this work, we denote the element-wise } 2\text{-norm and the spectral norm, i.e. the largest eigenvalue, as } \|\cdot\|_2 \text{ and } \|\cdot\|_s, \text{ respectively.} \]
Table 1: Choices to accumulate mean and variance of $g \equiv \nabla f(x)$ for the algorithms discussed in this work, typically via intermediate variables $m_\ell$, $v_\ell$, and $\hat{v}_\ell$. Steps sizes $\alpha_t$ are usually set to $\alpha/\sqrt{t}$ for provable convergence, but in practice often follow a different schedule, including constant steps. PGM uses $\alpha_t \in (0, 2/L_\ell)$, usually $1/L_\ell$. Constants $\beta_1$ or scheduled $\beta_1, \beta_2$ are from $[0, 1)$, $\epsilon > 0$, and $p \in (0, 1/2)$.

| Name       | Mean estimate | Variance estimate |
|------------|---------------|-------------------|
| SGD, PGM   | $m_t$         | $\hat{g}_t$      | $v_t$ | $\hat{v}_t$ | $\psi_t$ |
| ADA\textsc{Grad} | $m_t$ | $g_t$ | $v_t$ | $\hat{v}_t$ | $\psi_t$ |
| ADAM       | $\beta_1 m_{\ell-1} + (1 - \beta_1) g_t$ | $\beta_1 v_{\ell-1} + (1 - \beta_2) g_t^2$ | $\beta_1 \hat{v}_{\ell-1} + (1 - \beta_2) \hat{g}_t^2$ | $\max(\hat{v}_t, v_t)$ | $\sqrt{v_t}$ |
| AMSGrad    | $\beta_1 m_{\ell-1} + (1 - \beta_1) g_t$ | $\beta_1 v_{\ell-1} + (1 - \beta_2) g_t^2$ | $\max(\hat{v}_t, v_t)$ | $\sqrt{v_t}$ |
| ADAMX      | $\beta_1 m_{\ell-1} + (1 - \beta_1) g_t$ | $\beta_1 v_{\ell-1} + (1 - \beta_2) g_t^2$ | $\max(\hat{v}_t, v_t)$ | $\sqrt{v_t}$ |
| PADAM      | $\beta_1 m_{\ell-1} + (1 - \beta_1) g_t$ | $\beta_1 v_{\ell-1} + (1 - \beta_2) g_t^2$ | $\max(\hat{v}_t, v_t)$ | $\sqrt{v_t}$ |

Equation 4 but replace the moment accumulation with exponential moving averages, which has proven very popular and successful in practice. More recently, flaws in the original convergence proof of Kingma and Ba (2015) have triggered a series of minor modifications to the exact form of the $v_\ell$ term, e.g. AMSGrad (Reddi et al 2018), PAdam (Chen and Gu 2018), and ADAMX (Phuong and Phong 2019). To better clarify the corresponding choices, we rewrite Equation 4 as

$$x_{t+1} = x_t - \frac{\alpha}{\psi_2} \frac{\phi(g_1, \ldots, g_t)}{\psi(g_1^2, \ldots, g_t^2)}$$

and list the choices for $\phi$ and $\psi$ of each algorithm in Table 1.

3 Adaptive proximal gradient methods

We seek a forward-backward splitting method like Equation 2 that uses the adaptive gradient update scheme of Equation 5. The introduction of $\psi$ updates every dimension $j$ of $x$ with a different effective learning rate $\alpha_t/\psi_{t,j}$, which is equivalent to introducing a variance-normalizing metric $H_j$ for the parameter space. If $\psi$ is an approximation of the Hessian of $f$, the update corresponds to a proximal quasi-Newton scheme (Becker and Fadili 2012; Tran-Dinh et al 2015) of the form

$$x_{t+1} = \prox_{\alpha_t}^{H_t} \left( x_t - \alpha_t \frac{\phi(g_1, \ldots, g_t)}{\psi(g_1^2, \ldots, g_t^2)} \right),$$

with a variable-metric proximal operator

$$\prox_{\alpha_t}^{H}(x) \equiv \arginf_z \left\{ r(z) + \frac{1}{2\alpha} \|z - x\|_H^2 \right\} \text{ and } \|x\|_H = x^\top H x.$$

We refer to the regular proximal operator in an adaptive scheme without considering the variable metric, the results would be feasible but not optimal.

Duchi et al (2011) already introduced a variable-metric projection $P_{10}^{H}$ of the updated parameter $x_{t+1}$ onto a convex subset $\mathcal{S} \subset \mathbb{R}^d$. In this work, it is strikingly close to Equation 6, but differs in two aspects. First, ADA\textsc{Grad} uses $\phi_t = g_t$, i.e. the exact instantaneous gradient direction. Later methods have adopted moving averages, which constitutes an inexact proximal gradient method but does not affect convergence as long as $\phi_t - g_t$ decreases as...
Algorithm 1 Adaptive Proximal Gradient Method (ADAPROX)

1: procedure ADAPROX(x1; V(f); prox, (f); [α], [β]; ρ)
2: for t = 1, 2, . . . do
3:     g = V(f)(x)
4:     φt = φ(γ1, . . . , γl) # First moment function from Table 1
5:     ψt = ψ(γ2, . . . , γl) # Second moment function from Table 1
6:     xk+1 = xk + αt φt/ψt
7:     Ht = Diag(ψt)
8:     γt = αt/ max(ψt)
9:     zt = xk+1
10:     for τ = 1, 2, . . . do
11:         zt+1 = proxγt, 1 Ht (zt + (αt/βt)ψt(zt − ξt−1)) # Equation 7
12:         if ∥zt+1 − zt∥ < ϵ∥zt+1∥ then break
13:     xk+1 = zt+1
14:     if ∥xk+1 − xk∥ < ϵ∥xk+1∥ then break

$O(1/\delta^2)$ for any $\delta > 0$ (Schmidt et al. 2011). Second, projection operators are only a special class of proximal operators, namely those for the indicator function of any convex subset $S \subset \mathbb{R}^d$. A limitation to indicator functions is not fundamental, lifting it allows for adaptive schemes with regularizers that impose e.g. sparsity or low-rankness of the solutions.

This leads to the question how to solve Equation 7. Becker and Fadili (2012) showed that if $H \equiv D + uu^T$ with a diagonal $D$ and an arbitrary $u \in \mathbb{R}^d$, $\text{prox}^H(x)$ can be replaced with $\text{prox}_{\rho H^{-1/2}}(D^{1/2}x - v)$. The offset $v$ needs to be found through a line search that involves $\text{prox}_{\rho H^{-1/2}}$, which itself may be expensive to compute even if $\text{prox}_{\rho H}$ is efficient. Becker et al. (2019) demonstrated how to perform this computation more directly for several classes of common regularizers, which can lead to substantial performance gains.

We propose a more direct approach that is entirely agnostic about the regularizer. Because the $H$-norm part is differentiable with gradient $\frac{1}{\alpha}H(z - x)$ and Lipschitz constant $L_H = \frac{1}{\alpha} \sqrt{\|H^T H\|_2}$, the minimizer of Equation 7 for a given $x$ can be found with PGM:

$$z_{t+1} = \text{prox}_{\gamma t, 1} \left( z_t - \frac{\gamma t}{\alpha} H(z_t - x) \right) \text{ for } t = 1, 2, \ldots$$ (8)

The step size of the sub-problem is as usual $\gamma \in (0, 2/L_H)$. Duchi et al. (2011) and Tran-Dinh et al. (2015) showed that it is often sufficient, and much more efficient in high-dimensional settings, to diagonalize the metric: $H_t = \text{Diag}(\psi_t)$. With corresponding step sizes $\gamma_t = \alpha_t/\max(\psi_t)$, the proximal sub-iterations to achieve optimality read as

$$z_{t+1} = \text{prox}_{\gamma t, 1} \left( z_t - \frac{1}{\max(\psi_t)} \text{Diag}(\psi_t)(z_t - \xi_{t+1}) \right) \text{ for } t = 1, 2, \ldots,$$ (9)

where $\xi_{t+1}$ denotes the unconstrained parameter after gradient update from Equation 5. Once the desired level of convergence of the $z$-sequence is reached, $x_{t+1} \leftarrow z_{t+1}$. In essence, the PGM sub-iterations link ordinary unscaled proximal operators to quasi-Newton updates.

The entire algorithm is listed as Algorithm 1.

4 Applications to Constrained Matrix Factorization

With matrix factorization, one seeks to find factors $A$ and $S$ to approximate a matrix $Y$ by minimizing e.g. the loss

$$f(x) = \frac{1}{2} \|AS - Y\|_2^2.$$ (10)
For this bilinear problem the Lipschitz constants \( \|A^\top A\|_s \) and \( \|S S^\top\|_s \) have to be recomputed at every iteration \( t \). It becomes more complicated if data are affected by heteroscedastic or correlated noise. The loss function generalizes to

\[
f(x) = \frac{1}{2} (AS - Y)^\top \Sigma^{-1} (AS - Y)
\]

with an inverse covariance matrix \( \Sigma^{-1} \). As we have shown (Melchior et al 2018, section 2), the Lipschitz constants remain analytic but involve products of block-diagonal representations of \( A \) and \( S \) with \( \Sigma^{-1} \), which require spectral norms for very large matrices. A similar complication arises in online optimization or data fusion applications because not every batch or data set has an equal amount of information on all parameters. The global loss function for matrix factorization

\[
f(x) = \frac{1}{2} \sum_l (P_l AS - Y_l)^\top \Sigma_l^{-1} (P_l AS - Y_l)
\]

has gradients with complicated structure depending on the degradation operators \( P_l \) and noise properties \( \Sigma_l \) of data set \( Y_l \). In particular, the naïve estimate \( L = \sum_l L_l \) is an upper bound for the global Lipschitz constant that is applicable only in the unrealistic case that the data sets provide identical information about the parameters. The resulting step sizes will be under-estimated and thus slow down the convergence of the optimization. These applications should therefore benefit from our proposed adaptive proximal scheme.

4.1 Non-negative and Mixture Matrix Factorization

We consider the canonical non-negative matrix factorization (NMF), i.e. the parameterization and loss function from Equation 10 with the penalty function

\[
t_+ (x) = \begin{cases} 0 & \text{if } x_i \geq 0 \forall i, \\ \infty & \text{else} \end{cases}
\]

for both matrix factors \( A \) and \( S \). It provides a prototypical example of an efficient proximal operator \( \text{prox}_+ (x) = \max(0, x) \), i.e. an element-wise thresholding operator. The test data has \( C = 100 \) observations with Gaussian i.i.d. noise of a mixture model of \( K = 3 \) sinusoidal components \( \in \mathbb{R}^{50} \) and is shown in Figure 1. We also run a variant of NMF, dubbed MixMF, that is additionally constrained to impose the mixture-model characteristic of these data, i.e. \( \sum_l A_{ck} = 1 \forall c \). The correspondent proximal operator is the projection operator onto the simplex, \( \text{prox}_\text{unity} (x) = \frac{|x|}{\sum |x_i|} \), and is applied to every row \( A_c \) for \( c = \{1, \ldots, C\} \) to normalize the contributions of all components.

We compare the performance in terms of final loss and number of evaluations and proximal evaluations for PGM and ADA\textsc{prox} with the adaptive schemes listed in Table 1. The initial values for \( A \) and \( S \) are drawn from \( \mathcal{U}(0, 1) \). For PGM, we compute the analytic Lipschitz constants at every step. For ADA\textsc{prox}, we choose the step sizes by considering the amplitude of the elements of \( A \) and \( S \), which are of order unity. In the first run, we set them conservatively to \( \alpha = 0.01 \), in the second run more aggressively to \( \alpha = 0.1 \). In both runs the step sizes are kept constant. The results are shown in Figure 2 and summarized in Table 2.

Even with the conservative step sizes \( \alpha = 0.01 \), ADA\textsc{prox}–\text{AMSGrad} outperforms PGM in terms of final loss and number of iterations for both problems. For \( \alpha = 0.1 \), every adaptive scheme outperforms PGM on the NMF problem, but they all show mild to
**Fig. 1** NMF test data of $K = 3$ sinusoidal components $S_k \in \mathbb{R}^{50}$ (left), observed 100 times with different mixing weights and i.i.d. Gaussian noise of $\sigma_n = 0.02$ (center). The best-fit result of ADAProx-AMSGrad with $\alpha = 0.1$, rescaled to a maximum of 1, is shown in the right panel. The test data are publicly available in the code repository.

**Fig. 2** Loss for the NMF and MixMF problems of PGM and ADAProx with different step sizes and adaptive optimization schemes from Table 1. Following the recommendation by Kingma and Ba (2015), we set $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$; for PAdam, we set $p = 0.125$ as recommended by Chen and Gu (2018). With fixed step sizes $\alpha$, AMSGrad and ADAMX behave identically, the latter is thus not shown. Solutions are considered converged when the relative deviation of $A$ and $S$ between subsequent iterations is $< 10^{-4}$. 
Table 2 Performance for the NMF (top) and MixMF (bottom) problem of PGM and ADAProx with different adaptive optimization schemes. See Figure 2 for details. We list the number of iterations and the average number of proximal sub-iterations per iteration for (A, S), respectively.

| Name            | Final Loss | Iterations | Sub-Iterations | Final Loss | Iterations | Sub-Iterations |
|-----------------|------------|------------|----------------|------------|------------|----------------|
| NMF α = 0.01    |            |            |                |            |            |                |
| PGM             | 0.97261    | 541        | (1.1)          | 0.97261    | 541        | (1.1)          |
| ADAProx-ADAM    | 0.97121    | 663        | (1.39, 1.99)   | 0.96585    | 677        | (1.82, 1.98)   |
| ADAProx-PADAM   | 0.97811    | 600        | (1.92, 1.92)   | 0.96722    | 352        | (1.98, 1.98)   |
| ADAProx-AMSGRAD | 0.96928    | 405        | (1.97, 1.96)   | 0.96645    | 299        | (2.00, 2.00)   |
| MixMF α = 0.01  |            |            |                |            |            |                |
| PGM             | 1.0193     | 444        | (1.1)          | 1.0193     | 444        | (1.1)          |
| ADAProx-ADAM    | 1.0208     | 756        | (12.2, 1.99)   | 1.3738     | 1000*      | (16.5, 1.74)   |
| ADAProx-PADAM   | 1.0208     | 528        | (2.74, 1.94)   | 1.0227     | 286        | (3.07, 1.87)   |
| ADAProx-AMSGRAD | 1.0191     | 375        | (9.95, 1.99)   | 1.3436     | 1000*      | (11.1, 1.41)   |

| Name            | Final Loss | Iterations | Sub-Iterations | Final Loss | Iterations | Sub-Iterations |
|-----------------|------------|------------|----------------|------------|------------|----------------|
| NMF α = 0.1     |            |            |                |            |            |                |
| PGM             | 0.97261    | 541        | (1.1)          | 0.97261    | 541        | (1.1)          |
| ADAProx-ADAM    | 0.97121    | 663        | (1.39, 1.99)   | 0.96585    | 677        | (1.82, 1.98)   |
| ADAProx-PADAM   | 0.97811    | 600        | (1.92, 1.92)   | 0.96722    | 352        | (1.98, 1.98)   |
| ADAProx-AMSGRAD | 0.96928    | 405        | (1.97, 1.96)   | 0.96645    | 299        | (2.00, 2.00)   |

* indicates non-convergence after 1000 steps

prominent oscillations on the MixMF problem. We find empirically that for AMSGrad and PADAM, but not for ADAM, this behavior can be mitigated by reducing the moving average parameters. Values of $\beta_1 \approx 0.5$ and $\beta_2 \approx 0.8$ appear useful compromises between maintaining memory of previous gradients and adjusting to the newly constrained state.

Unsurprisingly, the solvers for the MixMF problem require more proximal sub-iterations for A than for S or for either in the NMF problem, where $\ll 2$ iterations of Equation 9 is sufficient. That low number is due to the per-element thresholding of $\text{prox}_{+}$. If an element $x_i < 0$, it will be projected to 0 on the first interation of $\text{prox}_{+}$. Because $H$ is diagonal, no other element is affected in the second iteration, the thesholding operation comes to the same result, and the sub-problem terminates after two iterations. The mixture-model constraint, on the other hand, affects all elements of A and therefore requires multiple passes to converge to the optimal solution. It is intriguing that ADAProx-PADAM requires fewer calls of $\text{prox}_{\text{unity}+}$. We do not have an explanation for this behavior.

Repeating the tests with different random seeds we establish these algorithm traits.

- Differences in the final loss between PGM and ADAProx are small. Adaptive schemes can reach convergence in fewer iterations.
- ADAProx-ADAM shows good performance with small step sizes, but is the most unstable scheme overall, possibly related to the concerns raised by Reddi et al (2018).
- ADAProx-AMSGRAD shows some instability with larger step sizes; reducing $\beta_1$ and $\beta_2$ is beneficial.
- ADAProx-PADAM, using $p = [0.1, 0.25]$, shows fast initial drops in the loss for large step sizes and converges quickly but to a slightly inferior final loss.

4.2 Astronomical Source Separation

Modern astronomical surveys have become so sensitive that they can reveal many astrophysical sources, primarily stars in our Galaxy as well as distant galaxies, in any region of the sky. The density of these sources is so high that they now routinely overlap, which complicates their measurement. Performing robust and, ideally, accurate source separation has thus become a critical task.
Fig. 3 Astronomical source separation example. A false-color composite of a 5-band image data cube (left) comprising $K = 7$ circular Gaussian sources with band-dependent Gaussian additive noise; the CMF model of this scene from ADAPROX-PADAM (center); individual components shown in Figure 5) and its residuals (right). The left and center panel use an inverse hyperbolic sine stretch to increase the dynamical range; the right panel uses a linear stretch. The test data are publicly available in the code repository.

We present a simplified test case that captures the main characteristic of source separation in astronomical surveys (see Melchior et al (2018) for a full implementation). The data set is comprised of $30 \times 30$ pixel images, observed in $C = 5$ different optical filter bands, and affected by filter-dependent uncorrelated Gaussian background noise. The problem is identical to hyperspectral unmixing with $C$ frequency bins, where both endmember spectra and spatial footprints are unknown. Using the definitions from Equation 12, and interpreting every filter as an independent observation $l$, we set $\Sigma_{l^{-1}} = \sigma_l^{-2}I$. The degradation operator $P_l$ amounts to a simple projection of the hyperspectral data cube to a single observed filter band. For the sake of simplicity and unlike actual observations, no convolution degrades the spatial resolution of the images.

We distribute $K = 7$ two-dimensional circular Gaussian-shaped sources randomly in the image, with sizes $\sigma_k$ ranging from 1 to 10 pixels. Their integrated fluxes scale with the size, $F_k \propto \sigma_k^2$, as is approximately observed for astronomical sources. The example multi-band image is shown in the left panel of Figure 3.

Since all astronomical sources are expected to be emitters of light, we impose non-negativity constraints on $A$ and $S$. To aid the source separation, we also add an $\ell_0$ penalty for $S$, whose proximal operator is the element-wise hard thresholding operator

$$\text{prox}_{\lambda \ell_0}(x) = \begin{cases} x & \text{if } |x| > \lambda \\ 0 & \text{else} \end{cases}$$

and then normalize the sum of the pixels with $\text{prox}_{\text{unity},+}$. This source separation problem is amenable to solution by PGM (e.g. Rapin et al 2013), which allows us to compare to ADAPROX solutions. In either case, we initialize the individual components with circular Gaussians, whose centers and sizes are randomized by up to $\sigma_k/4$ and 50%, respectively; their per-band amplitudes are taken from the noisy images at the assumed center. This approach mimics a data processing pipeline that performs the initial object detection and characterization to warm-start the source separation method.

For PGM, we again compute the analytic Lipschitz constants at every iteration to set the step sizes. For ADAPROX, we follow the general logic of Section 4.1 and set them relative to their typical amplitude. As the spatial distributions are normalized to unity, their mean amplitude is $\approx 10^{-3}$, and we decide on a more conservative setting by adjusting the
step sizes to 1%, i.e. \( \alpha^{(5)} = 10^{-5} \). The per-band amplitudes \( A_k \), however, are different by a factor of \( \approx 100 \) between the brightest and the faintest source. Unlike in PGM, we are free to set them differently for every component and chose \( \alpha^{(A_k)} = \frac{0.1}{c} \sum c A_{ck} \), reflecting our expectation that the initial amplitudes can have errors on the order of 10%.

The resulting losses are shown in Figure 4 and summarized in Table 3. It is evident that ADAProx can match or outperform PGM in terms of the final loss within a similar number of iterations. ADAProx-PADAM yields the best result, albeit with a slower convergence, but only after adjusting \( p = 0.45 \). With the recommended \( p \approx 0.125 \), the first few steps move far away from the initial \( A \) and \( S \), which means the solver largely ignores the reasonable starting positions. At \( p = 0.5 \), PADAM is identical to AMSGrad. Intermediate values of \( p \) appear to compromise between rapid initial improvement of the loss with PADAM and the robust performance that characterizes AMSGrad at smaller step sizes, in accordance with our observations in Section 4.1.

With the given constraints, ADAProx requires only 1 to 2 proximal sub-iterations. By avoiding the computation of the spectral norm for the Lipschitz constants in PGM, ADAProx exhibits much lower runtimes, which more than outweighs the computational cost of the adaptive schemes and the extra evaluations of the proximal operators.

The visual inspection of the individual components (Figure 5) of the best-fitting ADAProx-PADAM model confirms that the colors and shapes improve noticeably from well-chosen initial parameters. The spatial distributions reveal the impact of noise but also the effect of the \( \ell_0 \) penalty, which promotes configurations with few non-zero pixels.

---

### Table 3 Performance for the astronomy CMF problem problem of PGM and ADAProx. See Figure 4 for details. We list the number of iterations and the number of proximal sub-iterations per iteration for \((A, S)\), respectively. The runtime is for a single CPU on a recent Apple MacBook Pro.

| Name               | Final Loss | Iterations | Sub-Iterations | Runtime [s] |
|--------------------|------------|------------|----------------|-------------|
| PGM                | 2538.4     | 91         | (1, 1)         | 6.3         |
| ADAProx-Adam       | 2884.8     | 78         | (1.97, 2.00)   | 0.17        |
| ADAProx-PADAM      | 1398.2     | 167        | (1.19, 2.13)   | 0.38        |
| ADAProx-AMSGrad    | 1883.9     | 94         | (1.51, 2.00)   | 0.12        |
Fig. 5 Individual components of the model shown in the middle panel of Figure 3. The images use an inverse hyperbolic sine stretch adjusted for each component.
5 Summary

We present an adaptive proximal gradient method, ADA\textsc{prox}, which enables constrained convex optimization using the gradient updates of the recently proposed unconstrained method \textsc{Adam} and its variants \textsc{AMSGrad}, \textsc{AdamX} and \textsc{PADAM}. We solve the arising proximal quasi-Newton iteration by classical proximal gradient sub-iterations. The scheme is applicable to arbitrary proxable penalty functions. The cost of our proposed method arises from the need to compute and store moving averages of the first and second moment of the gradient of $f$ as well as multiple computations of the proximal mapping. Its benefits lie in the exploitation of the Hessian estimate to compute step sizes, which avoids the computation of Lipschitz constants traditionally required for the proximal gradient method. \textsc{ADAPROX} is thus beneficial in cases when the Lipschitz constants cannot be calculated analytically or efficiently, e.g. in non-linear models or signal processing problems with complicated observation design.

We demonstrate in three variants of the non-negative matrix factorization problem that \textsc{ADAPROX}, in particular with the \textsc{AMSGrad} and \textsc{PADAM} schemes, outperforms \textsc{PGM} in terms of final loss, number of iterations, and runtime. \textsc{ADAPROX} requires that step sizes for each parameter are set in advance. We find that relative step sizes on the order of 1% to 10% of the typical amplitude of the parameters work well in practice.

The python implementation of the algorithms presented here are available as an open-source package at https://github.com/pmelchior/proxmin.

References

Becker S, Fadili J (2012) A quasi-newton proximal splitting method. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ (eds) Advances in Neural Information Processing Systems 25, Curran Associates, Inc., pp 2618–2626, URL http://papers.nips.cc/paper/4523-a-quasi-newton-proximal-splitting-method.pdf

Becker S, Fadili J, Ochs P (2019) On Quasi-Newton Forward-Backward splitting: Proximal calculus and convergence. SIAM journal on optimization: a publication of the Society for Industrial and Applied Mathematics pp 2445–2481, DOI 10.1137/18M1167152, URL https://doi.org/10.1137/18M1167152

Chen J, Gu Q (2018) Closing the generalization gap of adaptive gradient methods in training deep neural networks URL http://arxiv.org/abs/1806.06763, 1806.06763

Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research: JMLR 12(Jul):2121–2159, URL http://www.jmlr.org/papers/v12/duchi11a.html

Hinton G, Srivastava N, Swersky K (2012) Neural networks for machine learning. Tech. rep., URL https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, URL http://arxiv.org/abs/1412.6980

Loris I, Bertero M, De Mol C, Zanella R, Zanni L (2009) Accelerating gradient projection methods for $\ell_1$-constrained signal recovery by steplength selection rules. Applied and computational harmonic analysis 27(2):247–254, DOI 10.1016/j.acha.2009.02.003, URL http://www.sciencedirect.com/science/article/pii/S10635203090000219

Melchior P, Moolekamp F, Jerdee M, Armstrong R, Sun AL, Bosch J, Lupton R (2018) scarlet: Source separation in multi-band images by constrained matrix factorization. Astronomy and Computing 24:129–142, DOI 10.1016/j.ascom.2018.07.001, URL http://www.sciencedirect.com/science/article/pii/0221333718300301

Moreau JJ (1965) Proximité et dualité dans un espace hilbertien. Bulletin de la Société Mathématique de France 93:273–299, DOI 10.24033/bsmf.1625, URL http://www.numdam.org/item/BSMF_1965__93__273_0

Parikh N, Boyd S (2014) Proximal algorithms. Foundations and Trends® in Optimization 1(3):127–239, DOI 10.1561/2400000003, URL http://dx.doi.org/10.1561/2400000003
Phuong TT, Phong LT (2019) On the convergence proof of AMSGrad and a new version DOI 10.1109/ACCESS.2019.2916341, URL http://arxiv.org/abs/1904.03590, 1904.03590
Rapin J, Bobin J, Larue A, Starck J (2013) Sparse and non-negative bss for noisy data. IEEE Transactions on Signal Processing 61(22):5620–5632, DOI 10.1109/TSP.2013.2279358
Reddi SJ, Kale S, Kumar S (2018) On the convergence of adam and beyond. URL https://openreview.net/pdf?id=ryQu7f-RZ
Schmidt M, Roux NL, Bach FR (2011) Convergence rates of inexact Proximal-Gradient methods for convex optimization. In: Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F, Weinberger KQ (eds) Advances in Neural Information Processing Systems 24, Curran Associates, Inc., pp 1458–1466, URL http://papers.nips.cc/paper/4452-convergence-rates-of-inexact-proximal-gradient-methods-for-convex-optimization.pdf
Tran-Dinh Q, Kyrillidis A, Cevher V (2015) Composite self-concordant minimization. Journal of machine learning research: JMLR URL http://www.jmlr.org/papers/volume16/trandihn15a/trandihn15a.pdf