A one phase IPM for non-convex optimization

Oliver Hinder, Yinyu Ye

June 16, 2017

Abstract

Solver

1 Main algorithm

$$D_\lambda(x, y) = \frac{\|\nabla L(x, y)\|_\infty}{\sqrt{\|y\|_\infty + 1}}$$

$$E_\mu(x, y, s) = \max\{D(x, y), \|Sy - \mu\|_\infty\}$$

Algorithm 1 One phase primal-dual IPM

function IPM(x, y)
- $\lambda \leftarrow ...$
- $f_\lambda(x) := f(x) + \lambda\|x\|^2$
- **for** $k = 1, ...\infty$ **do**
  - Form primal schur complement and factorize.
  - **if** matrix is PD **then**
    - Stable-correction
  - **else**
    - Stable-trust-region-step
  - **end if**
  - **for** $i = 1, ..., p(\#corrections)$ **do**
    - **if** $D_\lambda(x, y) \le \min\{\mu, \theta\}$ & $\frac{Sy}{\mu} \in [\beta_2, 1/\beta_2]$ **then**
      - Aggressive-correction
      - $\lambda \leftarrow ...$
    - **else**
      - Stable-correction
    - **end if**
  - **end for**
- **end for**

end function

1.1 Trust region

Algorithm 2 Stable-trust-region-step

```
function STABLE-TRUST-REGION-STEP(f, a, x, y, r)
    for j = 1, ..., ∞ do
        (d_x, d_y, M^+) ← APPROX-PRIMAL-DUAL-TRUST-REGION(f, a, x, y, r)
        x^+, y^+, α^+ ← stable-line-search(...)
        if α^+ > α_min then
            if j = 1 & α = 1 then
                r^+ ← 10r
            else if α^+ < α_small then
                r^+ ← r/2
            else
                r^+ ← ||d_x||_2
            end if
            break
        end if
        r^+ ← r^+/8
    end for
    return (x^+, y^+, M^+, r^+)
end function
```

Algorithm 3 Stable-trust-region-step

```
function STABLE-TRUST-REGION-IPOPT-STYLE(f, a, x, y, r)
    for j = 1, ..., ∞ do
        Factorize M with ... δ ...
        if inertia is good then
            ...
        end if
        x^+, y^+, α^+ ← stable-line-search(...)
    end for
    return (x^+, y^+, M^+, r^+)
end function
```

1.2 Line searches

```
function MOVE(f, a, x, y, d_x, d_y, η, α_P)
    x^+ ← x + α_P d_x
    μ^+ ← (1 − ηα_P)μ
    θ^+ ← (1 − ηα_P)θ
    s^+ ← a(x^+) + θ(s^1 − a(x^1))
    α_D ← arg max_{α∈[0,1]} α s.t. S^+(y + d_y α_D)/μ ∈ [eβ_1, e/β_1]
    y^+ ← y + α_D d_y
end function
```

Algorithm 4 Aggressive line search

```
function AGGRESSIVE-LINE-SEARCH(f, a, x, y, d_x, d_y)
    η ← 1
    α_P ← FRACTIONTOBOUNDARY(s, d_s)
    for i = 1, ..., ∞ do
        x^+, y^+, status ← MOVE(f, a, x, y, d_x, d_y, η, α_P)
        if status = feasible & Function value does not increase too much then
            return x^+, y^+, s^+
        else
            α_P ← α_P/2
        end if
    end for
end function
```

Algorithm 5 Stable line search

```
function STABLE-LINE-SEARCH(f, a, x, y, s, d_x, d_y, d_sη)
    η ← 0
    α_P ← FRACTIONTOBOUNDARY(y, s, d_y, d_s)
    for i = 1, ..., ∞ do
        x^+, y^+, status ← MOVE(f, a, x, y, d_x, d_y, η, α_P)
        if status = feasible then
            if sufficient progress on merit function then
                return x^+, y^+
            else

            end if
        else
            α_P ← α_P/2
        end if
        ...
    end for
end function
```

2 Scrap paper

3 Log barrier sub-problems

This paper is concerned with the following problem:

$$\min f(x) - \mu \log(s) + \frac{1}{2} d_x^T D_x d_x + \frac{1}{2} d_s^T D_s d_s \tag{1a}$$

$$a(x) - s = r\mu \tag{1b}$$

$$s \geq 0 \tag{1c}$$

The KKT conditions for (1) are:

$$\nabla_x \mathcal{L}(x, y) = \nabla f(x) + D_x d_x - \nabla a(x)^T y = 0 \tag{2a}$$

$$\mathcal{C}_\mu(s, y) = Ys - \mu e = 0 \tag{2b}$$

$$\mathcal{P}_\mu(x, s) = a(x) - s - \mu r = 0 \tag{2c}$$

$$s, y \geq 0 \tag{2d}$$

Where the Lagrangian $\mathcal{L}(x, y) := f(x) - y^T a(x)$.

We combine the log barrier merit function and the complementary conditions as follows:

$$\phi(x, y) = \psi(x) + \zeta(x, y) \tag{3}$$

With:

$$\zeta(x, y) = \frac{\|\mathcal{C}(x, y)\|_\infty^3}{\mu^2}$$

We now introduces models to locally approximate these merit functions $\nabla_x \mathcal{L}(x, y)$, ψ, $\mathcal{C}$ and ϕ respectively. To describe our approximations of a function f around the point (x, y) we use the function $\tilde{\Delta}^f_{(x,y)}(u, v)$ to denote the predicted increase in the function f at the new point $(x + u, y + v)$. Observe that we use different approximations depending on the choice of function f.

We use a typical linear approximate of $\nabla_x \mathcal{L}(x, y)$ as follows:

$$\tilde{\Delta}^{\nabla_x \mathcal{L}}_{(x,y)}(d_x, d_y) = \nabla_{x,x} L(x, y) d_x + \nabla a(x) d_y \tag{4}$$

The following function $\tilde{\Delta}^{\psi}_{(x,y)}(u)$ is an approximation of the function $\psi(x)$ at the point (x, y) and predicts how much the function ψ changes as we change the current from x to $x + u$.

$$\tilde{\Delta}^{\psi}_{(x,y)}(u) = \frac{1}{2} u^T M(x, y) u + \nabla \psi(x)^T u \tag{5}$$

With:

$$M(x, y) = \nabla^2 \mathcal{L}(x, y) + \sum_i \frac{y_i}{a(x)} \nabla a(x)^T \nabla a(x) \tag{6}$$

Note that if we set $y_i = \frac{\mu}{s_i}$ then $M(x, y) = \nabla^2 \psi(x)$ and $\tilde{\Delta}^{\psi}_{(x,y)}$ becomes the second order taylor approximation of ψ at the point x. Thus we can think of $\tilde{\Delta}^{\psi}_{(x,y)}(u)$ as a primal-dual approximation of the function ψ.

We can also build a model of the $\zeta(x, y)$ as follows:

$$\tilde{\Delta}^{\zeta}_{(x,y)}(d_x, d_y) = \frac{\|Sy + Y d_s + S d_y - \mu e\|_\infty^3 - \|\mathcal{C}(x, y)\|_\infty^3}{\mu^2} \tag{7}$$

With S a diagonal matrix containing entries of $a(x)$ and $d_s = \nabla a(x) d_x$. This model $\tilde{\Delta}^{\mathcal{C}}_{(x,y)}$ corresponds to the typical primal-dual linear model of $\mathcal{C}$ i.e. $C(x + d_x, y + d_y) \approx Sy + Y d_s + S d_y - \mu e$.

With S and Y contain the diagonal elements of $a(x)$ and y respectively.

This allows us to approximate the change in the function ϕ at the point (x, y) as follows:

$$\tilde{\Delta}^{\phi}_{(x,y)}(d_x, d_y) = \tilde{\Delta}^{\psi}_{(x,y)}(d_x) + \tilde{\Delta}^{\zeta}_{(x,y)}(d_x, d_y) \tag{8}$$

We say an iterate (x, y) satisfies approximate complementary if $(x, y) \in \mathcal{Q}_\mu$ where $\mathcal{Q}_\mu$ is defined as follows:

$$\mathcal{Q}_\mu = \left\{ (x, y) \in R^n \times R^m : a(x) > 0, y > 0, \|\mathcal{C}(x, y)\|_\infty \leq \frac{\mu}{2} \right\} \tag{9}$$

We say the point (x, y) is a μ-scaled KKT point if $(x, y) \in \mathcal{T}_\mu$ where:

$$\mathcal{T}_\mu = \{ (x, y) \in \mathcal{Q}_\mu : \|\nabla \mathcal{L}(x, y)\| \leq \mu(\|y\|_1 + 1) \} \tag{10}$$

In which case the algorithm terminates.

4 Algorithm

Let S, Y denote the diagonal matrices with entries of s and y respectively. We can linearize (2) at the iterate (x, y, s) as follows:

$$\begin{bmatrix} \nabla^2 \mathcal{L}(\hat{x}, \hat{y}) + D_x & -\nabla a(\hat{x})^T & 0 \\ \nabla a(\hat{x}) & 0 & -I \\ 0 & \hat{S} & \hat{Y} + D_s \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L}(x, y) \\ \mathcal{P}_\mu(x, s) \\ \mathcal{C}_\mu(s, y) \end{bmatrix} \tag{11}$$

Which is equivalent to solving:

$$\begin{bmatrix} \nabla^2 \mathcal{L}(\hat{x}, \hat{y}) + \nabla a(x)^T D_s \nabla a(x) + D_x & \nabla a(\hat{x})^T \\ \nabla a(\hat{x}) & -(\hat{Y} + D_s)^{-1} \hat{S} \end{bmatrix} \begin{bmatrix} d_x \\ -d_y \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L}(x, y) \\ \mathcal{P}_\mu(x, s) + (\hat{Y} + D_s)^{-1} \mathcal{C}_\mu(s, y) \end{bmatrix} \tag{12}$$

One can also solve this system by solving the Schur complement:

$$(\nabla^2 \mathcal{L}(\hat{x}, \hat{y}) + \nabla a(\hat{x})^T (\hat{Y} + D_s) \hat{S}^{-1} \nabla a(\hat{x}) + D_x) d_x = -\nabla \mathcal{L}(x, \mu S^{-1} e) - \nabla a(\hat{x})^T \hat{Y} \hat{S}^{-1} \mathcal{P}_\mu(x, s)$$

Observe that (**??**) may be singular or correspond to a direction that makes the log barrier objective worse. To rectify this problem we compute the direction as follows:

$$d_x = \arg \min_{\|u\|_2 \le r} \tilde{\Delta}^{\psi}_{(x,y)}(u) \tag{13a}$$

$$d_s = \nabla a(x) d_x \tag{13b}$$

$$d_y = -S^{-1} \left(Y d_s + \mathcal{C}(x, y) \right) \tag{13c}$$

******CAREFUL WITH SIGNS i.e. should be $d_s = -\nabla a(x) d_x$, $d_y = -S^{-1} \left(Y d_s + \mathcal{C}(x, y) \right)$ ***************
It is well-known from trust region literature that there exists some $\delta \in [0, \infty)$ such that:

$$(M(x, y) + \delta I) d_x = -\nabla \psi(x) \tag{14}$$

Furthermore, by re-arranging this equation we can deduce that (d_x, d_y, d_s) satisfies a perturbed version of (**??**):

$$\begin{bmatrix} \nabla^2 \mathcal{L}(x, y) + \delta I & -\nabla a(x)^T & 0 \\ -\nabla a(x) & 0 & I \\ 0 & S & Y \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_s \end{bmatrix} = - \begin{bmatrix} \nabla \mathcal{L}(x, y) \\ 0 \\ \mathcal{C}(x, y) \end{bmatrix} \tag{15}$$

Algorithm 6 Primal-dual trust region step

function PRIMAL-DUAL-TRUST-REGION(x, y, r) $**** \in ****$

$$d_x \in \arg \min_{\|u\| \le r} \tilde{\Delta}^{\psi}_{(x,y)}(u) \tag{16a}$$

$$d_s = \nabla a(x) d_x \tag{16b}$$

$$S = \mathrm{Diag}(a(x)) \tag{16c}$$

$$d_y = -S^{-1} \left(Y d_s + \mathcal{C}(x, y) \right) \tag{16d}$$

$(x^+, y^+) \leftarrow (x + d_x, y + d_y)$
return(x^+, y^+, d_x, d_y)
end function

Our complete algorithm is summarized as follows:

Algorithm 7 Primal-dual non-convex interior point algorithm

function NON-CONVEX-IPM(x^1, y^1)
  for $k = 1, ..., \infty$ **do**
    $r \leftarrow R(y^k)$
    repeat
      $(x^+, y^+, d_x, d_y) \leftarrow$ PRIMAL-DUAL-TRUST-REGION(x^k, y^k, r)
      if $(x^+, y^+) \in \mathcal{Q}_\mu$ **then**
        if $(x^+, y^+) \in \mathcal{T}_\mu$ **then**
          return (x^+, y^+)
        end if
      end if
      $r \leftarrow r/2$
    until $\phi(x^+) > \phi(x^k) + \frac{1}{2}\tilde{\Delta}^{\phi}_{(x^k, y^k)}(d_x, d_y)$
    $x^k \leftarrow x^+$
    $y^k \leftarrow y^+$
  end for
end function

5 Delta computation

Algorithm 8 Delta

$\lambda_{lb} = 0$, $\lambda_{ub} = \delta_{\max} = \|H\|_F^2$, δ_{k-1} ▷ lower and upper bounds on minimum eigenvalue
Try $\delta = 0$, if succeeds, trial solve with this delta. If step size is small skip to trust region step.
$\delta = \delta_{k-1}$
if $\delta = 0$ **then**
  $\delta = \delta_{\min}$
end if
for $i = 1, ..., \infty$ **do**
  Break if inertia correct and update λ_{lb} and λ_{ub}.
  $\delta = \delta 100$
end for
Trust region
$R = \|d_x^{k-1}\|_2$
for $i = 1, ..., \infty$ **do**
  Compute trust region with R
  If trust region is too accurate increase radius size
  If step unsuccessful decrease radius size
  Prevent oscillation
end for