Non-convex optimization

Irène Waldspurger[*]

December 8, 2022

1 Introduction

Let us consider a general unconstrained minimization problem:

$$\boxed{\text{find } x_* \text{ such that } f(x_*) = \min_{x\in\mathbb{R}^n} f(x),}$$

for some $f : \mathbb{R}^n \to \mathbb{R}$. We assume that at least one minimizer, x_*, exists. We also assume througout the lecture that f is $\mathcal{C}^\infty$, to avoid regularity issues.

In the past lectures, you have seen many ways to find a good approximation of a minimizer, under various structural assumptions on f, but mostly under the hypothesis that f is convex. The goal of this lecture is to study what we can do when f is not convex.

1.1 Why non-convex optimization is difficult

We first try to give an intuition of why non-convex optimization is much more difficult than convex optimization.

We consider the one-dimensional case, $n = 1$. Let us imagine that we run a first-order algorithm (that is, an algorithm which can access the value of f and ∇f at any desired point, and must return an approximate minimizer based on this information only). After some time, the algorithm has queried the values of f and ∇f at several points, for instance $\left\{-3, -1, -\frac{1}{2}, \frac{3}{2}, 3\right\}$. The gathered information is represented on Figure 1.

[*]waldspurger@ceremade.dauphine.fr

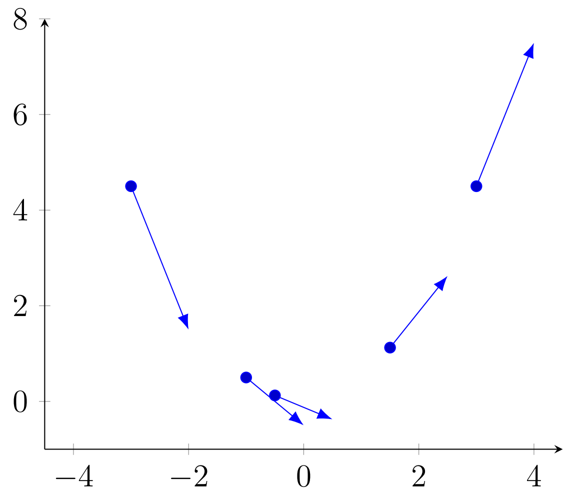

Figure 1: Values of f and ∇f at $-3, -1, -\frac{1}{2}, \frac{3}{2}, 3$.

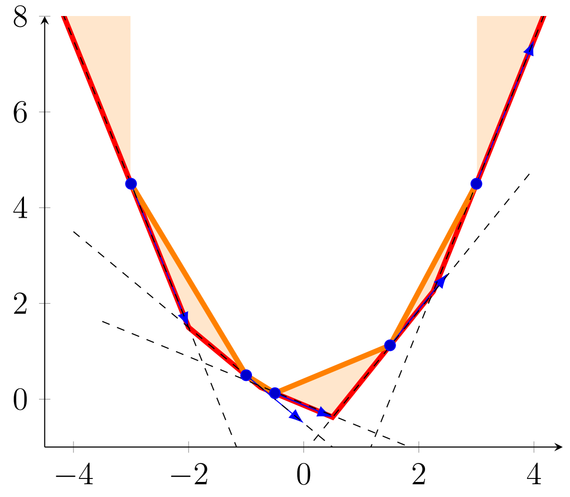

Figure 2: Upper and lower bounds on f, deduced from the information on Figure 1, under the assumption that f is convex; the graph of f must be entirely contained in the shaded zone.

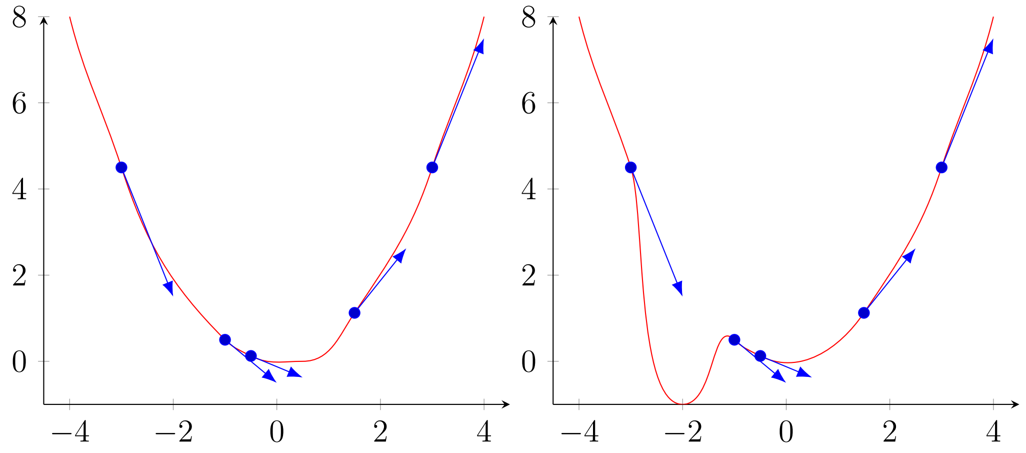

Figure 3: Two possible non-convex functions compatible with the information displayed on Figure 1.

If f is convex, this already gives significant information on the minimum and minimizer of f. Indeed, the graph of f is above its tangents, and below its chords, which provides upper and lower bounds for f, as shown on Figure 2. One can use them to estimate the minimum and minimizer of f. For instance, from the upper and lower bounds of Figure 2, one can deduce that

1. the minimum of f is between $-3/8$ and $1/8$;
2. the minimizer(s) of f belong(s) to the interval $[-1/2; 5/6]$.

In particular, from this information, one knows[1] the value of $\min f$ with precision $\frac{1}{4}$ and the minimizer with precision $\frac{2}{3}$.

But if f is not convex, this information does not allow to distinguish, for instance, the two functions plotted in Figure 3.

The function represented on the left reaches its minimum at $1/2$, and this minimum is 0. The function on the right reaches its minimum at -2, and this minimum is -1. The difference between the minimums of these two functions is 1, and the difference between the minimizers is 2.5: one cannot produce estimations for the minimal value and minimizer of f with a precision comparable to the convex setting.

[1]The minimum is in the interval $\left[-\frac{3}{8}; \frac{1}{8}\right]$. The middle point of this interval, $-\frac{1}{8}$, is therefore an approximation of $\min f$ which is at most $\frac{1}{4}$ away from the truth.

Intuitively, to compute a trustworthy approximation of $\min f$ or $\text{argmin} f$ without the convexity assumption, one needs to sample f on a fine grid. As soon as there is a “hole” in the sampling set[2], one cannot know whether the function takes large or small values in this hole, hence one cannot compute a precise estimate of $\min f$ or $\text{argmin} f$. In 1D, it may be possible to sample f on a fine grid, but if n is large, this is out of question: the number of sampling points on a fine grid grows exponentially with the dimension.

As a consequence, if f is not convex, we must give up the idea of finding an approximate minimizer. In the rest of the lecture, we will see which kind of points we can hope to find, and how.

2 Critical points

A first idea is to look for a *local minimizer* instead of a global one. It turns out that this is also out of reach, at least for pathological functions. Thus, we lower our expectations again: instead of looking for a local minimizer, we simply look for a point at which “the derivatives of f satisfy the same properties as at a local minimizer”.

Proposition 1

For any $x \in \mathbb{R}^n$, we denote $\text{Hess} f(x)$ the Hessian of f at x, that is the $n \times n$ matrix whose (i,j)-th coefficient is $\frac{\partial^2 f}{\partial x_i \partial x_j}(x)$.
If x is a local minimizer of f, then

$$\nabla f(x) = 0 \text{ and } \text{Hess} f(x) \succeq 0.$$

Almost conversely, if $\nabla f(x) = 0$ and $\text{Hess} f(x) \succ 0$, then x is a local minimizer of f.

[2]The *sampling set* is the set of points at which the algorithm queries the values of f and ∇f. In our example, it is $\{-3, -1, -1/2, 3/2, 3\}$.

Definition 1

We say that an element x of $\mathbb{R}^n$ is

- a *first-order critical point* of f if $\nabla f(x) = 0$,
- a *second-order critical point* of f if $\nabla f(x) = 0$ and $\mathrm{Hess}\, f(x) \succeq 0$.

Example

We consider the map $f : (x_1, x_2) \in \mathbb{R}^2 \to x_1^2 - x_2^2 \in \mathbb{R}$.
Its gradient and Hessian have the following formulas:

$$\forall x = (x_1, x_2) \in \mathbb{R}^2, \quad \nabla f(x) = (2x_1, -2x_2) \quad \text{and} \quad \mathrm{Hess}\, f(x) = \left(\begin{smallmatrix} 2 & 0 \\ 0 & -2 \end{smallmatrix}\right).$$

Therefore, f has a single first-order critical point, which is $(0, 0)$. This point is not a second-order critical point, because $\left(\begin{smallmatrix} 2 & 0 \\ 0 & -2 \end{smallmatrix}\right)$ is not semidefinite positive.

Although second-order critical points are not always local minimizers[3], the two notions nevertheless coincide for many functions f. In addition, it has been observed[4] that, in various interesting situations (including the training of neural networks), f has no local minimizer other than its global one, or, at least, all its local minimizers are approximate global minimizers (meaning $f(x) \approx f(x_*)$). It is thus of practical importance to be able to find critical points.

3 Convergence of gradient descent

Let us first consider the simplest first-order algorithm, gradient descent. You have studied, in the previous lectures, its convergence properties in the convex setting. How does it perform in the non-convex one?

We assume that f is L-smooth for some $L > 0$: For any $x, y \in \mathbb{R}^n$,

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y||.$$

[3]The map $(x \to x^3)$ has a second-order critical point at 0, but no local minimizer.

[4]At least numerically; theoretical justifications have been proposed, but only in quite specific settings.

We consider gradient descent with constant stepsize, equal to $1/L$: starting from an arbitrary $x_0 \in \mathbb{R}^n$, we define a sequence $(x_t)_{t\in\mathbb{N}}$ by

$$x_{t+1} = x_t - \frac{1}{L}\nabla f(x_t).$$

3.1 Convergence to a first-order critical point

Theorem 1

Let $T \in \mathbb{N}$ be fixed. We consider the following algorithm:

1. Run T steps of gradient descent, which defines a sequence $(x_0, x_1, \dots, x_T)$.
2. Compute $T_{min} = \text{argmin}_{0\le t\le T} ||\nabla f(x_t)||$ and define $\tilde{x}_T = x_{T_{min}}$.
3. Return $\tilde{x}_T$.

Then

$$||\nabla f(\tilde{x}_T)|| \le \sqrt{\frac{2L(f(x_0) - f(x_*))}{T}}.$$

We say that $\tilde{x}_T$ is a $O(1/\sqrt{T})$-approximate first-order critical point.

Proof. For all $x, h \in \mathbb{R}^n$,

$$\begin{aligned}
f(x+h) &= f(x) + \int_0^1 \langle \nabla f(x+th), h\rangle\, dt \\
&= f(x) + \int_0^1 \langle \nabla f(x) + (\nabla f(x+th) - \nabla f(x)), h\rangle\, dt \\
&= f(x) + \langle \nabla f(x), h\rangle + \int_0^1 \langle \nabla f(x+th) - \nabla f(x), h\rangle\, dt \\
&\le f(x) + \langle \nabla f(x), h\rangle + \int_0^1 ||\nabla f(x+th) - \nabla f(x)||\, ||h|| dt \\
&\qquad\qquad \text{(triangular inequality)} \\
&\le f(x) + \langle \nabla f(x), h\rangle + L\int_0^1 ||h||^2 t dt \\
&\qquad\qquad \text{(L-smoothness of f)}
\end{aligned}$$

$$= f(x) + \langle \nabla f(x), h \rangle + \frac{L}{2}||h||^2.$$

(This is a classical property of L-smooth functions.)

We apply this inequality to $x = x_t$ and $h = -\frac{1}{L}\nabla f(x_t)$:

$$\forall t \in \mathbb{N}, \quad f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}||\nabla f(x_t)||^2.$$

Consequently,

$$\begin{aligned}\sum_{t=0}^{T-1} ||\nabla f(x_t)||^2 &\leq 2L \sum_{t=0}^{T-1} (f(x_t) - f(x_{t+1})) \\ &= 2L(f(x_0) - f(x_T)) \\ &\leq 2L(f(x_0) - f(x_*)).\end{aligned}$$

Since $||\nabla f(\tilde{x}_T)|| \leq ||\nabla f(x_t)||$ for any $t \leq T$,

$$T||\nabla f(\tilde{x}_T)||^2 \leq 2L(f(x_0) - f(x_*)),$$

which implies

$$||\nabla f(\tilde{x}_T)|| \leq \sqrt{\frac{2L(f(x_0) - f(x_*))}{T}}.$$

□

3.2 Convergence to a second-order critical point

The previous theorem shows that gradient descent allows to find approximate first-order critical points, and even provides a convergence rate. For second-order critical points, the picture is more complicated.

For some choices of initial points x_0, it may happen that gradient descent does not get close to an approximate second-order critical point, even when run for an infinite number of steps. For instance, if x_0 is a first-order critical point of f, but not a second-order critical point, then

$$x_0 = x_1 = x_2 = \dots,$$

because $\nabla f(x_0) = 0$, hence gradient descent stays stuck at x_0 and never reaches a second-order critical point.

The following theorem shows that this phenomenon is very rare: for "general" initializations, it does not happen, and gradient descent converges to a second-order critical point.

Theorem 2: Lee, Simchowitz, Jordan, Recht (2016)

Let f be an L-smooth function. We assume that

- f has only a finite number of first-order critical points;
- for any $M \in \mathbb{R}$, $\{x \in \mathbb{R}^n, f(x) \leq M\}$ is bounded.

We consider gradient descent with constant stepsize $\alpha \in]0; \frac{1}{L}[$.
For almost any x_0 [a], $(x_t)_{t\in\mathbb{N}}$ converges to a second-order critical point.

[a] that is, for all x_0 outside a zero-Lebesgue measure set

Intuition of proof. The finiteness of the critical set and the boundedness of the level sets of f imply that $(x_t)_{t\in\mathbb{N}}$ converges to a first-order critical point whatever x_0. We admit this fact for simplicity.

We must show that, if x_{crit} is a first-order but not a second-order critical point of f, then $(x_t)_{t\in\mathbb{N}}$ does not converge to x_{crit}, for almost any x_0. We consider such a critical point; up to translation, we can assume that it is 0.

We make the (very) simplifying hypothesis that f is quadratic in a ball centered at 0, whose radius we call r_0:

$$\forall x \in B(0, r_0), \quad f(x) = \frac{1}{2}\langle x, Mx\rangle + \langle x, b\rangle,$$

for some $n \times n$ symmetric matrix M.

For any $x \in B(0, r_0)$, $\nabla f(x) = Mx + b$. Since 0 is a first-order critical point, we necessarily have $b = 0$. In addition, $\operatorname{Hess} f(x) = M$ for any $x \in B(0, r_0)$. The assumption that 0 is not a second-order critical point is then equivalent to the fact that $M \not\succeq 0$.

The matrix M can be diagonalized in an orthonormal basis:

$$M = U^T \begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n \end{pmatrix} U,$$

with $\lambda_1 \geq \dots \geq \lambda_n$ the eigenvalues of M and U an orthonomal matrix. Up to a change of coordinates, we can assume $U = \mathrm{Id}$. Since $M \not\succeq 0$, at least the smallest eigenvalue of M is negative: $\lambda_n < 0$.

If the sequence $(x_t)_{t\in\mathbb{N}}$ of gradient descent iterates converges to $x_{crit} = 0$, then x_t belongs to $B(0, r_0)$ for any t large enough, in which case

$$\begin{aligned} x_{t+1} &= x_t - \alpha\nabla f(x_t) \\ &= x_t - \alpha M x_t \\ &= \begin{pmatrix} \scriptstyle (1-\alpha\lambda_1)x_{t,1} \\ \vdots \\ \scriptstyle (1-\alpha\lambda_n)x_{t,n} \end{pmatrix}. \end{aligned}$$

We fix t_0 such that this relation holds for any $t \geq t_0$. Then, for any $s \in \mathbb{N}$,

$$x_{t_0+s} = \begin{pmatrix} \scriptstyle (1-\alpha\lambda_1)^s x_{t_0,1} \\ \vdots \\ \scriptstyle (1-\alpha\lambda_n)^s x_{t_0,n} \end{pmatrix}.$$

If the sequence converges to 0, all the coordinates of x_{t_0+s} must go to 0 when s goes to $+\infty$ (for any fixed t), which means that

$$\forall k \in \{1, \dots, n\}, \quad (1 - \alpha\lambda_k)^s x_{t_0,k} \overset{s\to+\infty}{\to} 0. \tag{1}$$

We have said that $\lambda_n < 0$, hence $1 < 1 - \alpha\lambda_n$ and $(1 - \alpha\lambda_n)^s \not\to 0$ when $s \to +\infty$. In order for Property (1) to hold, we must therefore have

$$x_{t_0,n} = 0.$$

To summarize, we have shown that, if $(x_t)_{t\in\mathbb{N}}$ converges to 0, then, for some t_0,

$$x_{t_0} \in \mathcal{E} \overset{def}{=} \{z \in B(0, r_0) \text{ such that } z_n = 0\}.$$

As a consequence,

$$x_0 \in (\mathrm{Id} - \alpha\nabla f)^{-t_0}(\mathcal{E}).$$

(For any map $g : \mathbb{R}^n \to \mathbb{R}^n$, we define $g^{-t_0}(\mathcal{E})$ as the set of points x such that $g^{t_0}(x) = g \circ \overset{t_0 \text{ times}}{\dots} \circ g(x) \in \mathcal{E}$.) Therefore, the set of initial points x_0 for which the gradient descent iterates may converge to 0 is included in

$$\bigcup_{t\in\mathbb{N}} (\mathrm{Id} - \alpha\nabla f)^{-t}(\mathcal{E}).$$

The set $\mathcal{E}$ has zero Lebesgue measure and one can check that $\mathrm{Id} - \alpha\nabla f$ is a diffeomorphism, hence $(\mathrm{Id} - \alpha\nabla f)^{-t}(\mathcal{E})$ has zero Lebesgue measure for any $t \in \mathbb{N}$, and the set of "problematic" initial points also has zero Lebesgue measure.

□

4 A second-order method

The theorem stated in the previous paragraph only states that gradient descent converges to a second-order critical point (for almost any initial point x_0). It does not say anything about the convergence rate. And it turns out that there are functions f for which the convergence is terribly slow.

To overcome this possible slow convergence, several strategies are possible. One of them is to add "noise" to gradient iterates from time to time, to help them get away faster from first-order critical points. The interested reader will find a description in How to escape saddle points efficiently, by C. Jin, R. Ge, P. Netrapalli, S. Kakade and M. Jordan (ICML 2017)

Another one is to explicitely exploit the information provided by second-order derivatives. This yields the family of *second-order methods*. In this section, we briefly describe one member of this family: the trust-region method.

Second-order derivatives provide local quadratic approximations of f.

Proposition 2

For any $x \in \mathbb{R}^n$,

$$f(x+h) = f(x) + \langle h, \nabla f(x) \rangle + \frac{1}{2} \langle h, \mathrm{Hess}\, f(x) h \rangle + o(||h||^2). \tag{2}$$

To define x_{t+1} from x_t, it is therefore reasonable to set

$$h_t = \underset{||h|| \leq R_t}{\mathrm{argmin}} \left(f(x_t) + \langle h, \nabla f(x_t) \rangle + \frac{1}{2} \langle h, \mathrm{Hess}\, f(x_t) h \rangle \right)$$

and $x_{t+1} = x_t + h_t$. In the definition of h_t, R_t is a positive number, the *trust radius*. Intuitively, it represents the radius of the region over which the quadratic approximation (2) is valid. Choosing it properly is important for the good behavior of the algorithm.

We provide convergence guarantees for this algorithm under the assumption that $\mathrm{Hess}\, f$ is L_2-Lipschitz for some $L_2 > 0$:

$$\forall x, y, h \in \mathbb{R}^n, \quad ||(\mathrm{Hess}\, f(x) - \mathrm{Hess}\, f(y))h|| \leq L_2 ||x - y||\, ||h||.$$

Theorem 3

Let $\epsilon > 0$ be fixed.
We run the trust-region algorithm as described above, with $R_t = \frac{\sqrt{\epsilon}}{L_2}$ for any t. We stop the algorithm if

$$\frac{||\nabla f(x_t) + \text{Hess}\, f(x_t) h_t||}{||h_t||} \leq \sqrt{\epsilon}$$

and return x_{t+1}.
For any $x_0 \in \mathbb{R}^n$, the algorithm stops after at most $O\left(\frac{L_2^2(f(x_0)-f(x_*))}{\epsilon^{3/2}}\right)$ iterations and the output x_{final} is an approximate second-order critical point, in the sense that

$$||\nabla f(x_{final})|| \lesssim \frac{\epsilon}{L_2} \quad \text{and} \quad \lambda_{\min}(\text{Hess}\, f(x_{final})) \gtrsim -\sqrt{\epsilon}.$$

(The notation "$\lesssim$" means "smaller up to a moderate multiplicative constant" and λ_{min} is the smallest eigenvalue.)

5 Example: phase retrieval

In the last part of this lecture, we give an example of a non-convex problem where it turns out that all second-order critical points are global minimizers and, moreover, it is possible to rigorously prove this fact. This example is *phase retrieval*.

In phase retrieval, one wants to recover an unknown vector $x_{true} \in \mathbb{C}^n$. Some linear maps $L_1, \dots, L_m : \mathbb{C}^n \to \mathbb{C}$ are fixed and one has access to

$$y_1 = |L_1(x_{true})|, \dots, y_m = |L_m(x_{true})|.$$

Here, the double bar, « $|.|$ » denotes the standard complex modulus. This problem is notably motivated by applications in imaging.

Since, for any $\alpha \in \mathbb{R}, k \leq m$, $|L_k(e^{i\alpha} x_{true})| = |e^{i\alpha}|\,|L_k(x_{true})| = |L_k(x_{true})|$, it is not possible to distinguish x_{true} from $e^{i\alpha} x_{true}$ from the knowledge of $y_1, \dots, y_m$. However, when $m \geq 4n$, it is possible to prove that, for almost all linear forms $L_1, \dots, L_m$, x_{true} is uniquely determined by $y_1, \dots, y_m$ up to multiplication by some unitary complex number $e^{i\alpha}$. In this case, which algorithm can recover x_{true}?

Recovering x_{true} is equivalent to finding $x \in \mathbb{C}^n$ such that

$$|L_1(x)| = y_1, \dots, |L_m(x)| = y_m.$$

The modulus is non-differentiable, but its square is, so it is simpler to rewrite these equalities as

$$|L_1(x)|^2 = y_1^2, \dots, |L_m(x)|^2 = y_m^2.$$

An intuitive idea to find such an x is to minimize the square-norm error between $(|L_1(x)|^2, \dots, |L_m(x)|^2)$ and $(y_1^2, \dots, y_m^2)$, that is

$$\mathcal{L}(x) = \sum_{k=1}^{m} \left(|L_k(x)|^2 - y_k^2 \right)^2 .$$

The function $\mathcal{L}$ is not convex. Therefore, attempting to minimize it with a first or second-order algorithm may fail: the algorithm will typically find a second-order critical point, but this critical point may not be the global minimizer x_{true}.

Numerically, it can indeed happen that the algorithm returns a point which is not close to x_{true}. However, when m is large enough compared to n and the linear maps $L_1, \dots, L_m$ are sufficiently "incoherent" with each other, it empirically seems that "bad" critical points do not exist[5].

This fact can be rigorously established, although under strong assumptions on $L_1, \dots, L_m$. Specifically, we assume that $L_1, \dots, L_m$ are generated randomly and independently according to a normal distribution (that is, for each k, the coordinates of L_k in the canonical basis are independent realizations of complex Gaussian variables with unit variance). We also assume that

$$m \geq Cn \log^3(n).$$

Theorem 4: Sun, Qu, Wright (2018)

Under the above assumptions, the second-order critical points of $\mathcal{L}$ are exactly its global minimizers $\{e^{i\alpha} x_{true}, \alpha \in \mathbb{R}\}$, with probability at least $1 - \frac{1}{m}$.

As a consequence, in this setting, it is possible to recover x_{true} by simply running gradient descent on $\mathcal{L}$, since Theorem 2 guarantees that gradient descent converges to a second-order critical point for almost any initialization.

[5] or, at least, are sufficiently rare so that a first or second-order algorithm does not find them

6 References

- Gradient descent only converges to minimizers, by J. D. Lee, M. Simchowitz, M. Jordan and B. Recht, in the Conference on Learning Theory (COLT), 2016.

- Second order optimization algorithms I, lecture notes by Y. Ye, available at http://web.stanford.edu/class/msande311/2017lecture13.pdf.

- Computing a trust region step, by J. J. Moré and D. C. Sorensen, in the SIAM journal on scientific and statistical computing, volume 4, number 3, 1983.

- A geometric analysis of phase retrieval, by J. Sun, Q. Qu and J. Wright, in Foundations of Computational Mathematics, volume 18, number 5, 2018.