Lecture 7

W. Hebisch

November 30, 2021

1 Self-concordant functions

Recall from last lecture: we say that convex $f : I \to \mathbb{R}$ (where I is an interval) is self-concordant when

$$|f'''(x)| \leq 2f''(x)^{3/2}$$

for all x in domain of f. We say that convex multivariate f is self-concordant when restriction of f to any line is self-concordant.

In Newton method we use scalar product $\langle h_1, h_2\rangle_x = \langle \nabla^2 f(x)h_1, h_2\rangle$ dependent on x which is better adapted to f, than usual scalar product. We will assume that values of self-concordant function f go to infinity when arguments go to boundary of the domain. This ensures that self-concordant function is defined on maximal possible domain. Let $W_x = \{y : \|y - x\|_x < 1\}$.

1.1 Main estimate

Lemma 1.1 *Let f be as above. f is defined on W_x and for $\|h\|_x < 1$ we have*

$$f(x) + \langle\nabla f(x), h\rangle + \phi(-\|h\|_x) \leq f(x + h) \leq f(x) + \langle\nabla f(x), h\rangle + \phi(\|h\|_x)$$

where $\phi(s) = -log(1 - s) - s = \sum_{i=2}^{\infty} \frac{s^i}{i}$. Moreover,

$$(1 + \|h\|_x)^{-2}\nabla^2 f(x) \leq \nabla^2 f(x + h) \leq (1 - \|h\|_x)^{-2}\nabla^2 f(x).$$

Lower bounds remain valid as long as $x + h$ is in domain of f.

1.2 Newton method for self-concordant functions

Our main estimate means that nondegenerate f is well conditioned on compact subsets of W_x. In particular this implies uniform speed of convergence of Newton method. More precisely, recall that gradient of f at x with respect to norm $\|\cdot\|_x$ is given by

$$(\nabla^2 f(x))^{-1}\nabla f(x)$$

Put

$$\lambda(f, x) = \|(\nabla^2 f(x))^{-1}\nabla f(x)\|_x = \langle\nabla^2 f(x))^{-1}\nabla f(x), \nabla f(x)\rangle^{1/2}.$$

When $\lambda(f, x)$ is small (say smaller than $\frac{1}{2}$) we have fast (quadratic) convergence. When $\lambda(f, x)$ is bounded from below, then using step staying in W_x we can still get some fixed decay of objective function.

To stay in domain of f it is natural to use damped Newton method, that is put

$$x_{i+1} = x_i - \frac{1}{1 + \lambda(f, x_i)} (\nabla^2 f(x_i))^{-1} \nabla f(x_i)$$

Lemma 1.2 *We have*

$$f(x_i) - f(x_{i+1}) \geq \lambda(f, x_i) - \log(1 + \lambda(f, x_i))$$

Note: For $\lambda(f, x_i) = 1$ this predicts decay approximately by 0.3068, for $\lambda(f, x_i) = \frac{1}{2}$ we get 0.0945.

Example: Let $g(x) = -\gamma x - \log(1 - x) - x$. Comparing derivative of g to 0 we see that g attains minimal value at $x = \gamma/(1 + \gamma)$. Also $g'(0) = -\gamma$, $g''(0) = 1$, $\lambda(g, 0) = \gamma$ and damped Newton method started in $x_0 = 0$ will get minimal value of g in single step and decay of objective function is exactly equal to estimate from the lemma. In particular, single step estimate can not be improved and any other choice of step depending only on λ will lead to worse estimate.

Proof of the lemma: Lemma essentially follows from example and main estimate. Namely, let $w = (\nabla^2 f(x_i))^{-1} \nabla f(x_i)$, $u = w/\|w\|_{x_i}$. Put $\gamma = \langle \nabla f(x_i), u \rangle$ and $g(t) = -\gamma t - \log(1 - t) - t$.

We have

$$\lambda(f, x_i) = \|w\|_{x_i} = \langle \nabla f(x_i), (\nabla^2 f(x_i))^{-1} \nabla f(x_i) \rangle^{1/2} = \langle \nabla f(x_i), w \rangle^{1/2}$$

so

$$\gamma = \langle \nabla f(x_i), u \rangle = \frac{\langle \nabla f(x_i), w \rangle}{\|w\|_{x_i}} = \frac{\lambda(f, x_i)^2}{\lambda(f, x_i)} = \lambda(f, x_i).$$

Hence, $\lambda(g, 0) = \gamma = \|w\|_{x_i} = \lambda(f, x_i)$. By main estimate

$$f(x_i + tu) \leq f(x_i) + g(t).$$

However, in damped Newton method $x_{i+1} = x_i + tu$ with

$$t = \frac{\|w\|_{x_i}}{1 + \lambda(f, x_i)} = \frac{\gamma}{1 + \gamma}$$

which is the same t as value produced by damped Newton method applied to g.

For local convergence we have:

Lemma 1.3 *When x_{i+1} is given by damped Newton method*

$$\lambda(f, x_{i+1}) \leq 2\lambda(f, x_i)^2.$$

When x_{i+1} is given by standard Newton method

$$\lambda(f, x_{i+1}) \leq \left(\frac{\lambda(f, x_i)}{1 + \lambda(f, x_i)} \right)^2$$

In particular, damped Newton method has $\lambda(f, x_{i+1}) < \lambda(f, x_i)$ when $\lambda(f, x_i) < \frac{1}{2}$, while standard Newton when $\lambda(f, x_i) < \frac{3-\sqrt{5}}{2} \approx 0.3819$.

The result are quite satisfactory: when $\lambda(f, x_i)$ is large we get steady decay of objective function, once $\lambda(f, x_i) < \frac{1}{2}$ local convergence takes over. But there is a little troubling aspect: decrease of objective function is rather small when say $\frac{1}{2} < \lambda(f, x_i) < 2$. We will see later that this region is particularly interesting for applications.

We can not get better decay of f, but naively we could hope that small number of steps will transition from large $\lambda(f, x_i)$ to quadratic convergence. Below we show by example that this is not the case: we can get many steps when $\lambda(f, x_i)$ is quite close to 1.

Example: Let $f(x) = -\log(x) + \varepsilon x^2$. It is self-concordant and attains minimum at $x_\infty = \frac{1}{\sqrt{2\varepsilon}}$. When $x_0 = 1$ we have $f'(x_0) = -1 + 2\varepsilon$, $f''(x_0) = 1 + 2\varepsilon$. Easy calculation shows that pure Newton method makes step slightly smaller than 1. More complicated calculation shows that damped Newton method makes step slightly smaller than $\frac{1}{2}$. After single step we may rescale our function so we get problem like original, only with changed ε. So we may get several steps like above. In case of pure Newton method improvement of objective function is approximately $\log(2) \approx 0.693$. In case of damped Newton method improvement is approximately $\log(\frac{3}{2}) \approx 0.405$. So theoretically (with very small ε) we may have very large number of steps with only moderate improvement in each step. Practically, we have here numbers of widely varying magnitude and numerical accuracy will limit number of steps.

This was example in dimension one, but we can add arbitrarily many irrelevant variables. If we add quadratic term in extra variables with minimum at 0, we get true multidimensional problem, which however have the same convergence behaviour as our problem in dimension one.

Note that from point of view of classical convergence theory f is very badly conditioned and our result based on self-concordance is much better.

1.3 What next

Self-concordant functions are rather special, but we will see that they appear as barrier functions in interior point methods for constrained problems. In particular we will use the results for interior point method of solving linear programming problems. Now, interior point methods also work quite well for several nonlinear problems. However, before constrained optimization we shall study some different topics, so there will be a little discontinuity here.

1.4 Further reading 1

Stephen Boyd, Lieven Vandenberghe, Convex Optimization, chapter 9.

A. Nemirovski, INTERIOR POINT POLYNOMIAL TIME METHODS IN CONVEX PROGRAMMING, lecture notes, chapter 2.

Yurii Nesterov, Introductory lectures on convex optimization, Springer 2004, chapter 4.1.

2 Conjugate direction methods

We already know about gradient descent and Newton method.

Gradient descent may require very large number of iterations, but per iteration cost is small.

Newton method typically converges in smaller number of iterations, but in each iteration we need to solve linear system of equations involving second derivative.

Conjugate direction methods have per iteration cost slightly larger than gradient descent, but should converge faster. In this sense are intermediate between gradient descent and Newton method.

Remark: In practice we care about conjugate gradient method, but theory is nicer when we make things slightly more general that is consider conjugate direction methods.

Conjugate direction methods originally were introduced as storage efficient method of exact solving of some linear systems. More precisely we know that for positive A minimization of

$$\frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle$$

is equivalent to solving

$$Ax = b.$$

Original conjugate direction methods minimized quadratic form, in at most n steps reaching exact solution where n is dimension of the problem.

Later it was observed that stopping method earlier one can get approximate solution and frequently one can get good solution in relatively small number of steps.

Conjugate direction methods were generalized to nonlinear problem. In such case method in no longer convergent in finite number of steps.

Let A be a positive definite matrix.

Definition. We say that a sequence of vectors d_i is A-orthogonal (conjugate) if and only if for $i \neq j$

$$\langle Ad_i, d_j\rangle = 0$$

Recall standard linear algebra:

Lemma 2.1 *If vectors d_i are A-conjugate, then they are linearly independent.*

Remark: To better see that this is standard result we can introduce new scalar product by the formula

$$\langle x, y\rangle_A = \langle Ax, y\rangle.$$

Then A-orthogonal simply means orthogonal with respect to this new scalar product.

A-conjugate vectors help in optimizing $f(x) = \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle$.

Algorithm:

1. Take arbitrary x_0.
2. For i starting at 1 repeat: $x_i = x_{i-1} + \alpha_i d_i$ where α_i minimizes f on the line $x_{i-1} + \alpha_i d_i$.

First we deal with one dimensional optimization. Put

$$\phi(s) = f(x_{i-1} + sd_i).$$

We have

$$\phi'(s) = \langle A(x_{i-1} + sd_i) - b, d_i\rangle.$$

Writing $r_i = Ax_{i-1} - b$ we have

$$\phi'(s) = \langle r_i, d_i\rangle + s\langle Ad_i, d_i\rangle$$

so solving for $\phi'(\alpha_i) = 0$ we get

$$\alpha_i = \frac{-\langle r_i, d_i\rangle}{\langle Ad_i, d_i\rangle}$$

Lemma 2.2 *Let* $x_* = \operatorname{argmin}_{x\in H} f(x)$ *where* $H = x_0 + \operatorname{lin}\{d_1, \dots, d_i\}$. *Then* $x_* = x_i$.

Proof: For any $x \in H$ we have $x - x_0 = \sum_{j=1}^{i} \beta_j d_j$. By A-orthogonality

$$\langle A(x - x_0), d_j\rangle = \beta_j \langle Ad_j, d_j\rangle$$

so

$$\langle A(x - x_0), x - x_0\rangle = \sum_{j=1}^{i} \beta_j^2 \langle Ad_j, d_j\rangle$$

Now

$$f(x) = \frac{1}{2}\langle A((x - x_0) + x_0), (x - x_0) + x_0\rangle - \langle b, (x - x_0) + x_0\rangle$$

$$= \frac{1}{2}\langle A(x - x_0), x - x_0\rangle + \langle A(x - x_0), x_0\rangle - \langle b, x - x_0\rangle + \frac{1}{2}\langle Ax_0, x_0\rangle - \langle b, x_0\rangle$$

$$= \frac{1}{2}\left(\sum_{j=1}^{i} \beta_j^2 \langle Ad_j, d_j\rangle\right) + \sum_{j=1}^{i} \beta_j \langle Ad_j, x_0\rangle - \sum_{j=1}^{i} \beta_j \langle b, d_j\rangle$$

$$+\frac{1}{2}\langle Ax_0, x_0\rangle - \langle b, x_0\rangle$$

$$= c + \sum_{j=1}^{i} \phi_j(\beta_j)$$

where $c = \frac{1}{2}\langle Ax_0, x_0\rangle - \langle b, x_0\rangle$ and

$$\phi_j(\beta_j) = \frac{1}{2}\beta_j^2\langle Ad_j, d_j\rangle + \beta_j(\langle Ad_j, x_0\rangle - \langle b, d_j\rangle)$$

so $f(x)$ is sum of functions of separate variables. Clearly for such functions optimizing in turn with respect to each variable gives optimal point. But this is exactly what our algorithm is doing.

Lemma 2.3 $\langle r_{i+1}, d_j\rangle = 0$ *for* $j = 1, \ldots, i$.

Proof: This follows from optimality of x_i on H: derivative of f at x_i in direction of d_j is 0. But

$$\partial_s f(x_i + sd_j) = \langle A(x_i + sd_j), d_j\rangle - \langle b, d_j\rangle$$

$$= \langle Ax_i - b, d_j\rangle + s\langle Ad_j, d_j\rangle = \langle r_{i+1}, d_j\rangle + s\langle Ad_j, d_j\rangle.$$

and when $s = 0$ this gives the claim.

2.1 Gram-Schmidt orthogonalization

Now it remains to find sequence of A-orthogonal vectors. There are many such sequences. General procedure is called Gram-Schmidt orthogonalization and works as follows:

- start from arbitrary basis $\{v_i\}$
- take $d_1 = v_1$
- iteratively, given $d_1, \ldots, d_i$ compute $c_{i+1,j} = \langle v_{i+1}, d_j\rangle / \langle d_j, d_j\rangle$ and put $d_{i+1} = v_{i+1} - \sum_{j=1}^{i} c_j d_j$

This is easy to program, but requires rather large number of operations and large memory.

Remark: In our case we would need to use $\langle x, y\rangle_A$ which significantly increases cost of computation.

2.2 Conjugate gradient method

Fortunately one A-orthogonal sequence of vectors is easy to find. Corresponding algorithm is called conjugate gradient method. We put $d_1 = -r_1$ and

$$d_{i+1} = -r_{i+1} + \beta_i d_i$$

where β_i is such that d_{i+1} and d_i are A-orthogonal, that is

$$\beta_i = \frac{\langle Ar_{i+1}, d_i\rangle}{\langle Ad_i, d_i\rangle}.$$

Lemma 2.4 *Assume that $r_1, \ldots, r_i$ are nonzero. Then*

$$\mathrm{lin}\{d_1, \ldots, d_i\} = \mathrm{lin}\{r_1, Ar_1, \ldots, A^{i-1}r_1\}$$

and d_j, $j = 1, \ldots, i$ are A-orthogonal.

Proof: By induction. Denote space on the right hand side above by V_i and on the left hand side by W_i. Clearly $V_1 = W_1$ and sequence consistiong of single vector is A-orthogonal. So we may assume that claim is valid for i and need to prove it for $i+1$. To prove that d_j are A-orthogonal it enough to prove that d_{i+1} is A-orthogonal to W_i. By definition of d_{i+1} we know that d_{i+1} is A-orthogonal to d_i. So it is enough to show that d_{i+1} is A-orthogonal to W_{i-1}.

Now

$$d_{i+1} = -r_{i+1} + \beta_i d_i$$

since d_i is A-orthogonal to W_{i-1} it is enough to show that r_{i+1} is A-orthogonal to W_{i-1}. For $v \in W_{i-1}$ we have

$$\langle Ar_{i+1}, v\rangle = \langle r_{i+1}, Av\rangle.$$

However,

$$Av \in AW_{i-1} = AV_{i-1} \subset V_i = W_i$$

where we used inductive assumption. Also, by inductive assumption d_j, $j = 1, \ldots, i$ are A-orthogonal so by previous lemma r_{i+1} is orthogonal to W_i. Hence

$$\langle r_{i+1}, Av\rangle = 0$$

which shows A-orthogonality of d_j, $j = 1, \ldots, i, i+1$.

It remains to show equality $V_{i+1} = W_{i+1}$. Clearly $AV_i \subset V_{i+1}$. We have $x_i - x_0 \in W_i$, so by inductive assumption $x_i - x_0 \in V_i$. Hence

$$r_{i+1} = Ax_i - b = A(x_i - x_0) + Ax_0 - b = A(x_i - x_0) + r_1 \in V_{i+1}$$

and $d_{i+1} = -r_{i+1} + \beta_i d_i \in V_{i+1}$, so $W_{i+1} \subset V_{i+1}$. But we proved that r_{i+1} is orthogonal to W_i, so since r_{i+1} is nonzero we have $r_{i+1} \notin W_i = V_i$, so r_{i+1} and V_i span V_{i+1}, so $V_{i+1} = W_{i+1}$.

A priori single step of conjugate gradient method needs two application of matrix A to vector: one to compute $r_{i+1} = Ax_i - b$ and second to compute Ad_i.

We can simplify computation of conjugate gradient method as follows: $-\langle r_i, d_i\rangle = \langle r_i, r_i\rangle - \beta_{i-1}\langle r_i, d_{i-1}\rangle = \langle r_i, r_i\rangle$ so

$$\alpha_i = \frac{\langle r_i, r_i\rangle}{\langle Ad_i, d_i\rangle}$$

$$r_{i+1} = r_i + \alpha_i Ad_i$$

$$\beta_i = \frac{\langle r_{i+1}, Ad_i\rangle}{\langle Ad_i, d_i\rangle}$$

$$d_{i+1} = -r_{i+1} + \beta_i d_i$$

which needs one multiplication of vector by matrix, 3 scalar products and 2 vector linear combinations.

Recall that we say that a method is first order method if

$$x_i \in x_0 + \mathrm{lin}\{\nabla f(x_0), \ldots, \nabla f(x_{i-1})\}.$$

Lemma 2.5 *Assume f is quadratic as above, $V_i = \mathrm{lin}\{r_1, Ar_1, \ldots, A^{i-1}r_1\}$. For any first order method $x_i \in x_0 + V_i$.*

Proof: We have

$$\nabla f(x) = Ax - b = Ax_0 - b + A(x - x_0) = r_1 + A(x - x_0).$$

Now the claim follows by induction, if $x_i \in x_0 + V_i$, then

$$\nabla f(x_i) \in r_1 + AV_i \subset V_{i+1}$$

so

$$\mathrm{lin}\{\nabla f(x_0), \ldots, \nabla f(x_{i-1})\} \subset V_i$$

which gives the claim.

Note: $V_0 = \{0\}$ so $x_0 \in x_0 + V_0$ which gives starting point for induction.

Together with previous lemma this means that for quadratic functions conjugate gradient method is the best first order method.

Remark: Best here means that we get smallest value of goal function. There are different criteria, for example smallness of $\|\nabla f(x)\|$ or distance between x and optimal point x_∞.

Now we will explore some consequences. Let

$$P(t) = \sum_{j=0}^{i-1} \gamma_j t^j$$

be a polynomial of degree $i - 1$. Let

$$y_i = x_0 + P(A)r_1$$

Similar to previous argument y_i is obtained by in step i of a first order method, so error is at least as big as error of conjugate gradient method, that is

$$\langle A(x_i - x_\infty), x_i - x_\infty \rangle \le \langle A(y_i - x_\infty), y_i - x_\infty \rangle.$$

But $r_1 = Ax_0 - b$ and $b = Ax_\infty$ so we can write

$$y_i - x_\infty = x_0 + P(A)(Ax_0 - Ax_\infty) - x_\infty = (I + AP(A))(x_0 - x_\infty)$$

so

$$\langle A(x_i - x_\infty), x_i - x_\infty \rangle \le \langle A(I + AP(A))(x_0 - x_\infty), (I + AP(A))(x_0 - x_\infty) \rangle.$$

From this one can derive more specific estimate, we skip detail of proof, and just give the result.

Lemma 2.6 *Assume that A has $n-m$ eigenvalues in interval $[a,b]$ and remaining m eigenvalues are bigger than b. Then*

$$E(x_{m+1}) \leq \left(\frac{b-a}{b+a}\right)^2 E(x_0)$$

where

$$E(x) = \langle A(x - x_\infty), x - x_\infty \rangle$$

is error at x.

Note that what matters above are distinct eigenvalues. In particular, when A has only $m+1$ eigenvalues, than we get optimum after $m+1$ steps.

There are cases when A has small number of distinct eigenvalues, but in general we expect distinct eigenvalues. In fact, in problem 5.2 you should compute eigenvalues of matrix A corresponding to simple differential problem. This matrix has distinct eigenvalues and "spread out". While problem 5.2 is simplified so that eigenvalues of A are easily computable similar behaviour appears in other differential problems. Such problems appear in many engineering tasks.

2.3 Preconditioning conjugate gradient

To improve conditioning we can use transformation

$$x = C^{-1}y$$

In terms of y convergence depends on eigenvalues of $(C^{-1})^T AC^{-1}$. When $W = (C^T C)^{-1}$ is approximate inverse of A, then new problem is well conditioned (note that WA and $(C^{-1})^T AC^{-1}$ have the same eigenvalues).

It is important that there is no need to explicitly compute transformed problem. Namely, denoting by $x_i = C^{-1}y_i$ and using $'$ to denote other transformed quantities in y variables we have

$$A' = (C^{-1})^T AC^{-1},$$

$$b' = (C^{-1})^T b$$

$$r'_k = A'y_{k-1} - b' = A'Cx_{k-1} - b' = (C^{-1})^T (Ax_{k-1} - b).$$

Writing $z_k = Ax_{k-1} - b$ and transforming r'_k to x variables we compute

$$r_k = C^{-1}r'_k = (C^T C)^{-1}(Ax_{k-1} - b) = W(Ax_{k-1} - b) = Wz_k.$$

Next,

$$\langle r'_k, r'_k \rangle = \langle W(Ax_{k-1} - b), (Ax_{k-1} - b) \rangle = \langle Wz_k, z_k \rangle$$

so we can compute this in x coordinates. Similarly $d_1 = C^{-1}d'_1 = -C^{-1}r'_1 = -r_1$ and

$$d_{i+1} = C^{-1}d'_{i+1} = -C^{-1}r'_i + \beta'_i C^{-1}d'_i = -r_k + \beta'_i d_i,$$

$$\langle A'd'_i, d'_i\rangle = \langle AC^{-1}d'_i, C^{-1}d'_i\rangle = \langle Ad_i, d_i\rangle,$$
$$\langle r'_{i+1}, A'd'_i\rangle = \langle r_{i+1}, Ad_i\rangle$$

so indeed we can compute scalar products giving α'_i and β'_i in terms of x variables. Finally

$$\begin{aligned} z_{i+1} = Ax_i - b = AC^{-1}y_i - b &= AC^{-1}(y_{i-1} + \alpha'_i d'_i) - b \\ = \alpha'_i AC^{-1}d'_i + AC^{-1}y_{i-1} - b &= \alpha'_i Ad_i + Ax_{i-1} - b \\ &= z_i + \alpha'_i Ad_i \end{aligned}$$

so we get whole iteration.

Note that there is no need to explicitly compute W: all we need is ability to compute Wz_k which we can do by solving equations.

2.4 Nonquadratic conjugate gradient methods

When f is not a quadratic function, then there are several possible ways to generalize conjugate gradient method. First, note that our $r_j = \nabla f(x_{j-1})$. As before we use $x_i = x_{i-1} + \alpha_i d_i$ with $d_1 = -r_1$ and $d_{i+1} = -r_{i+1} + \beta_i d_i$. To choose search direction, we need rule for β_i. Two popular choices are Fletcher-Rieves formula

$$\beta_i = \frac{\|\nabla f(x_i)\|^2}{\|\nabla f(x_{i-1})\|^2}$$

and Polak-Ribiere formula

$$\beta_i = \frac{\langle \nabla f(x_i), \nabla f(x_i) - \nabla f(x_{i-1})\rangle}{\|\nabla f(x_{i-1})\|^2}.$$

Both formulas give the same results for quadratic functions, but in practice Polak-Ribiere formula seem to give better results for nonquadratic ones. To determine α_i we need to perform line search. Backtracking line search would give bad results for quadratic functions, so we need better one. Evaluating f at 3 points on line and using quadratic interpolation to find approximate minimum gives exact result for quadratic functions and seem to behave well in general.

For nonquadratic functions search directions are no longer orthogonal and after large number of steps may be quite suboptimal. Usual way to handle this is to use restarts, that is from time to time take $\beta_i = 0$.

Close to optimal point one can show that conjugate gradient method with restarts after n steps (where n is dimension of the space) behaves similar to Newton method, more precisely n steps of conjugate gradient method do similar work as one step of Newton method.

This is easy to see at intuitive level: close to optimal point our function behaves very similar to quadratic function and we could use conjugate gradient to compute $f''(x_i)^{-1}\nabla f(x_i)$. Using f gives slightly different result, but resulting convergence as fast as Newton method (but we need n steps of conjugate gradient to perform one Newton step).

When f'' is well conditioned, then more frequent restarts lead to faster convergence: method behaves like Newton method with approximate inverse of f''.

2.5 Quasi-Newton methods

In the next lecture we will look at different approach to methods intermediate between gradient descent and Newton method, namely at quasi-Newton methods. We will also look at Nesterov acceleration and momentum. Here we just remark that conjugate gradient method may be viewed as variant of momentum.

2.6 Further reading 2

David G. Luenberger, Yinyu Ye, Linear and Nonlinear Programming, chapter 9.

Jorge Nocedal, Stephen J. Wright, Numerical Optimization, chapter 5.