4M17: Practical Optimisation

Coursework Assignment 1

Cyrus Mostajeran

Technicalities:

- Assignments due by 4pm on Friday the 7th of December 2018
- Submission is via Moodle
- DPO Sessions: 4pm - 6pm Friday of week 6 & 7 (16th & 23rd November)
- Direct questions to me (csm54)

Problem 1

1. The simplest *norm approximation problem* is an unconstrained problem of the form

$$\text{minimise} \quad \|Ax - b\|, \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ are given as problem data, $x \in \mathbb{R}^n$ is the variable, and $\|\cdot\|$ is a norm on $\mathbb{R}^m$.

(a) Define the l_1, l_2, and l_∞-norms on $\mathbb{R}^m$ and write down the norm approximation problems corresponding to each of these norms. In the case of the l_2-norm, show that the problem can be expressed as an optimisation problem with a convex quadratic function with an analytic solution, which amounts to solving a linear system of equations.

Problem 1

(b) Show that the norm approximation problems corresponding to the l_1 and l_∞-norms on $\mathbb{R}^m$ can be cast as *linear programming* (LP) problems of the form

$$\begin{aligned} \min_{\tilde{x}} \quad & \tilde{c}^T \tilde{x} \\ & \tilde{A}\tilde{x} \leq \tilde{b}. \end{aligned}$$

Specify the dimensions and provide expressions for $\tilde{A}$, $\tilde{b}$, $\tilde{c}$ in terms of A and b for each of the two norms.

Problem 1

(c) In the folder Q1Data, 5 pairs of problem data (A, b) are provided for $m = 2n$ and $n = 16, 64, 256, 512, 1024$. Utilise the dual-simplex algorithm in Matlab's `linprog` functionality to solve the LP problems corresponding to l_1 and l_∞-norms, and use `linsolve` to minimise the l_2-norm. Produce a table showing the values of the minimised l_1, l_2, and l_∞ norms $\|Ax - b\|$ corresponding to each of the 5 pairs in Q1Data, as well as the running times of your algorithms in each case.

Problem 1

(c) In the folder Q1Data, 5 pairs of problem data (A, b) are provided for $m = 2n$ and $n = 16, 64, 256, 512, 1024$. Utilise the dual-simplex algorithm in Matlab's `linprog` functionality to solve the LP problems corresponding to l_1 and l_∞-norms, and use `linsolve` to minimise the l_2-norm. Produce a table showing the values of the minimised l_1, l_2, and l_∞ norms $\|Ax - b\|$ corresponding to each of the 5 pairs in Q1Data, as well as the running times of your algorithms in each case.

	min llAx-bll_1	min llAx-bll_2	min llAx-bll_inf	l_1 runtime	l_2 runtime	l_3 runtime
(A1,b1)	This	is	sort	of	what	I
(A2,b2)	have	in	mind	.	.	.
(A3,b3)	.	.	.	.	.	.
(A4,b4)	.	.	.	.	.	.
(A5,b5)	.	.	.	.	.	.

Problem 1

(d) Plot the histogram of the residuals of the norm approximation problem for the 5th pair of data `(A5,b5)` in Q1Data ($n = 1024$) for the l_1, l_2, and l_∞-norms. Briefly comment on the shape of the distribution of the residuals in each case.

Problem 2

2. In this problem, we replace the non-smooth l_1-norm cost function

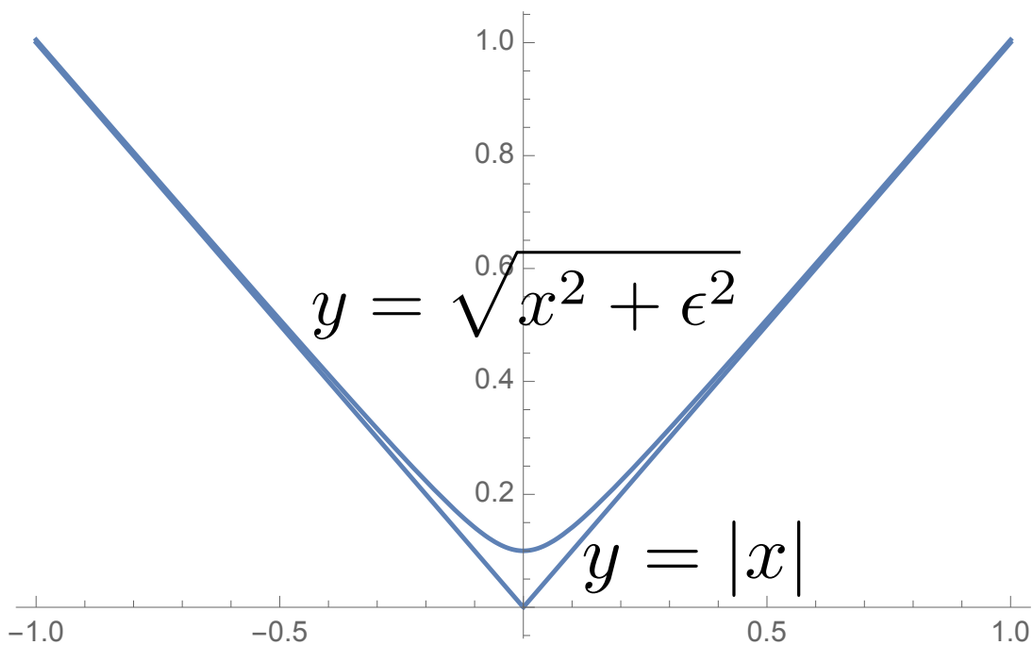
$$f(x) = \|Ax - b\|_1$$

of the previous problem with

$$f_\epsilon(x) = \sum_{i=1}^{m} \sqrt{(Ax-b)_i^2 + \epsilon^2},$$

where $\epsilon > 0$ is a small parameter.

(a) Write down expressions for the gradient $\nabla f_\epsilon(x)$ and Hessian $H = \nabla^2 f_\epsilon(x)$.



Problem 2

(b) Apply a gradient descent algorithm with backtracking linesearch to the cost function f_ϵ in (4) with $\epsilon = 0.01$ for the first two data pairs (A, b) $(n = 16, 64)$ from the previous problem, clearly stating your choice of parameters for the backtracking step. You may use $\|\nabla f_\epsilon(x)\|_2 < 1.0 \times 10^{-3}$ as the stopping criterion for your algorithm. Produce a table showing the value of the minimised cost function f_ϵ, the number of iterations taken for your algorithm to terminate, and the corresponding results obtained for the exact l_1-norm from the previous problem in each case.

Problem 2

(b) Apply a gradient descent algorithm with backtracking linesearch to the cost function f_ϵ in (4) with $\epsilon = 0.01$ for the first two data pairs (A, b) $(n = 16, 64)$ from the previous problem, clearly stating your choice of parameters for the backtracking step. You may use $\|\nabla f_\epsilon(x)\|_2 < 1.0 \times 10^{-3}$ as the stopping criterion for your algorithm. Produce a table showing the value of the minimised cost function f_ϵ, the number of iterations taken for your algorithm to terminate, and the corresponding results obtained for the exact l_1-norm from the previous problem in each case.

	min f_epsilon(x)	no. of iterations (steepest descent)	min ||Ax-b||_1	no. of iterations (l1 LP algorithm)
(A1,b1)	Something	like	this,	say.
(A2,b2)	.	.	.	.

Problem 2

Descent methods

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)} \quad \text{with } f(x^{(k+1)}) < f(x^{(k)})$$

- other notations: $x^+ = x + t\Delta x$, $x := x + t\Delta x$
- Δx is the *step*, or *search direction*; t is the *step size*, or *step length*
- from convexity, $f(x^+) < f(x)$ implies $\nabla f(x)^T \Delta x < 0$
  ($i.e.$, Δx is a *descent direction*)

General descent method.

given a starting point $x \in \mathbf{dom}\, f$.
repeat
1. Determine a descent direction Δx.
2. *Line search.* Choose a step size $t > 0$.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

Problem 2

Line search types

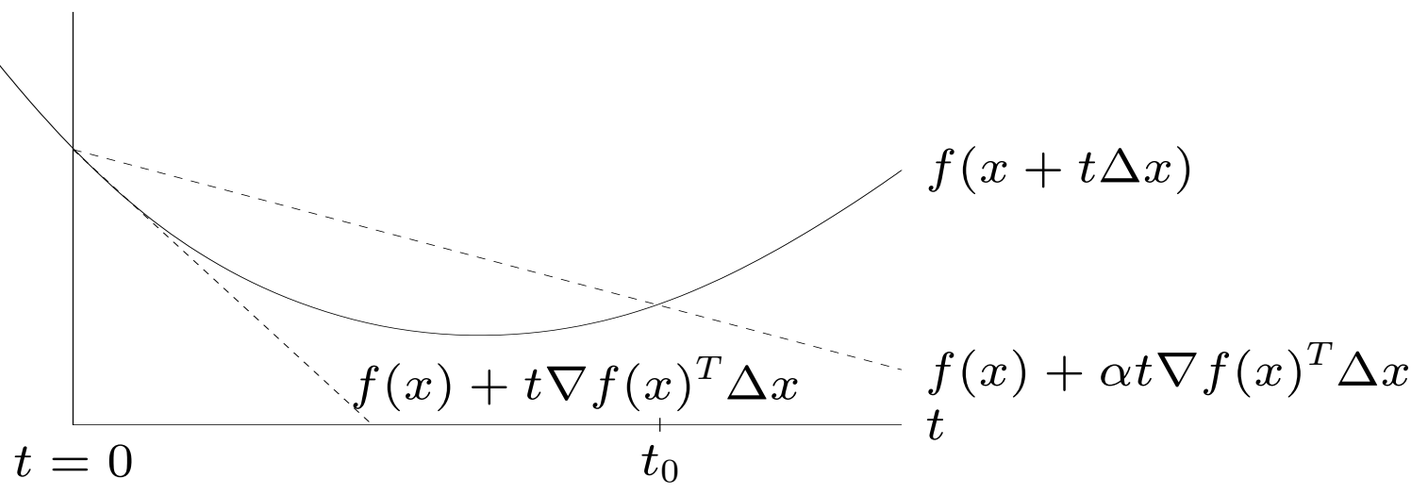
exact line search: $t = \operatorname{argmin}_{t>0} f(x + t\Delta x)$

backtracking line search (with parameters $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$)

- starting at $t = 1$, repeat $t := \beta t$ until

$$f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$$

- graphical interpretation: backtrack until $t \leq t_0$



Problem 2

Gradient descent method

general descent method with $\Delta x = -\nabla f(x)$

---

given a starting point $x \in \mathbf{dom}\, f$.
repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

- stopping criterion usually of the form $\|\nabla f(x)\|_2 \leq \epsilon$

Problem 2

(c) Analyse the dependence of the convergence rate and the value of the minimised cost function f_ϵ on ϵ by applying gradient descent to the first pair `(A1,b1)` and varying ϵ from 1.0×10^{-1} to 1.0×10^{-3}. Plot the running time of your algorithm against ϵ. What is the effect of the size of $\epsilon > 0$ on the convergence rate? Indicate the value of the minimised function f_ϵ for each choice of ϵ that you have used. What do these values suggest about the choice of $\epsilon = 0.01$ used to approximate f?

Problem 2

(d) Apply the Newton method to f_ϵ for each of the data pairs (A, b) ($n = 16, 64, 256, 512, 1024$), using Cholesky decomposition to invert the Hessian. Compare your findings with the results from part (b). Perform a more detailed convergence analysis in the case of the data pair `(A2,b2)` by comparing gradient descent with the Newton method and produce a semi-log plot of the error in the optimised function against the number of iterations in each case.

	min f_epsilon(x)	no. of iterations (Newton)	no. of iterations (grad descent)	runtime (Newton)	runtime (grad descent)
(A1,b1)	.	.	.	.	.
(A2,b2)	.	.	.	.	.
(A3,b3)	.	.	.	.	.
(A4,b4)	.	.	.	.	.
(A5,b5)	.	.	.	.	.

Problem 2

Newton's method

---

given a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.
repeat

1. *Compute the Newton step and decrement.*
   $\Delta x_{\mathrm{nt}} := -\nabla^2 f(x)^{-1} \nabla f(x)$; $\quad \lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)$.
2. *Stopping criterion.* **quit** if $\lambda^2/2 \leq \epsilon$.
3. *Line search.* Choose step size t by backtracking line search.
4. *Update.* $x := x + t\Delta x_{\mathrm{nt}}$.

---

affine invariant, $i.e.$, independent of linear changes of coordinates:

Newton iterates for $\tilde{f}(y) = f(Ty)$ with starting point $y^{(0)} = T^{-1}x^{(0)}$ are

$$y^{(k)} = T^{-1}x^{(k)}$$

Problem 2

Implementation

main effort in each iteration: evaluate derivatives and solve Newton system

$$H\Delta x = -g$$

where $H = \nabla^2 f(x)$, $g = \nabla f(x)$

via Cholesky factorization

$$H = LL^T, \qquad \Delta x_{\rm nt} = -L^{-T}L^{-1}g, \qquad \lambda(x) = \|L^{-1}g\|_2$$

- cost $(1/3)n^3$ flops for unstructured system
- cost $\ll (1/3)n^3$ if H sparse, banded

Problem 3

3. Consider the optimisation problem

$$\begin{aligned} &\text{minimise} && \|Ax - b\|_2 \\ &\text{subject to} && \mathbf{card}(x) \leq k, \end{aligned}$$

where $\mathbf{card}(x)$ denotes the number of nonzero components of $x \in \mathbb{R}^n$. Regularisation with an l_1-norm can be used as a good heuristic approach to solving this problem, whereby we solve the problem

$$\text{minimise} \quad \|Ax - b\|_2 + \gamma \|x\|_1$$

for different values of γ, finding the smallest value of γ that results in a solution with $\mathbf{card}(x) = k$. We then fix this sparsity pattern and find the value of x that minimises $\|Ax - b\|_2$.

Problem 3

(a) Replace the cost function

$$f(x) = \|Ax - b\|_2 + \gamma\|x\|_1$$

with a smooth version $f_\epsilon(x)$ by introducing a smoothened approximation to the non-smooth l_1-norm in place of $\|x\|_1$, as in Problem 2 with $\epsilon = 0.001$ this time. Write down an expression for the gradient $\nabla f_\epsilon(x)$.

Problem 3

(b) Apply your gradient descent algorithm from the previous problem to this new smooth cost function f_ϵ where (A, b) are given by `(A5,b5)` from Q1Data ($n = 1024$). You may use $\|\nabla f_\epsilon(x)\|_2 < 1.0 \times 10^{-5}$ as the stopping criterion for your algorithm. Experiment on how to tune γ to achieve a cardinality $k = \mathbf{card}(x) = 8$. For this problem you may consider an entry x_i of the vector $x \in \mathbb{R}^{1024}$ to be zero if $|x_i| < \epsilon$. State your choice of γ and specify the sparsity pattern by identifying the indices of the 8 non-zero elements in $x \in \mathbb{R}^{1024}$.

Problem 3

(c) Fix the sparsity pattern identified in part (b) and minimise $\|Ax-b\|_2$. State the value and index of each of the non-zero elements of your optimal solution x^* and evaluate $\|Ax^* - b\|_2$.