Lecture 7: Gradient descent method

COMP5930M Scientific Computation

Today

Recap

Algorithms for the initial state

- Current strategies
- Gradient descent (or steepest descent) method

The Newton algorithm

- Initial state: $\mathbf{x} = \mathbf{x}_0$
- while $|\mathbf{F}(\mathbf{x}_k)| > Tol$
  - $\mathbf{J}(\mathbf{x}_k)\delta = -\mathbf{F}(\mathbf{x}_k)$
  - $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda\delta$

- This first step is critical to the success of the algorithm

Guaranteeing convergence?

So far we have seen:

- Use *domain-knowledge* to choose $\mathbf{x}_0$
- Line-search update to improve convergence

Sometimes neither will lead to successful convergence

Gradient descent

- A solution algorithm in its own right
- Only linearly convergent
- Usually associated with minimisation problems
- Can be used to *start* Newton's method from a poor initial guess, after which we switch to standard Newton (similar idea as before when combining bisection and Newton)

Equivalence between minimisation and solving equations

- Given a system of nonlinear equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, we can find an equivalent **optimisation problem**
- Define a new function $\phi(\mathbf{x}) = |\mathbf{F}|^2 = \sum_{i=1}^{n} (F_i(\mathbf{x}))^2$

Equivalence between minimisation and solving equations

- Given a system of nonlinear equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, we can find an equivalent **optimisation problem**
- Define a new function $\phi(\mathbf{x}) = |\mathbf{F}|^2 = \sum_{i=1}^{n} (F_i(\mathbf{x}))^2$
  - At $\mathbf{x}^*$, $\mathbf{F}(\mathbf{x}^*) = \mathbf{0} \implies \phi(\mathbf{x}^*) = 0$
  - $\phi(\mathbf{x}) > 0, \quad \forall\ \mathbf{x} \neq \mathbf{x}^*$

Equivalence between minimisation and solving equations

- Given a system of nonlinear equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, we can find an equivalent **optimisation problem**
- Define a new function $\phi(\mathbf{x}) \ = \ |\mathbf{F}|^2 \ = \ \sum_{i=1}^{n} (F_i(\mathbf{x}))^2$
  - At $\mathbf{x}^*$, $\mathbf{F}(\mathbf{x}^*) = \mathbf{0} \implies \phi(\mathbf{x}^*) = 0$
  - $\phi(\mathbf{x}) > 0, \quad \forall\ \mathbf{x} \neq \mathbf{x}^*$
- Minimising ϕ is equivalent to finding a root of $\mathbf{F}$
- Optimality condition:

$$\nabla\phi(\mathbf{x}^*) = \frac{\partial}{\partial \mathbf{x}}\phi(\mathbf{x}^*) = \mathbf{0} \quad \Leftrightarrow \quad 2\, J(\mathbf{x}^*)^T \mathbf{F}(\mathbf{x}^*) = \mathbf{0}.$$

Equivalence between minimisation and solving equations

- Given a system of nonlinear equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, we can find an equivalent **optimisation problem**
- Define a new function $\phi(\mathbf{x}) = |\mathbf{F}|^2 = \sum_{i=1}^{n} (F_i(\mathbf{x}))^2$
  - At $\mathbf{x}^*$, $\mathbf{F}(\mathbf{x}^*) = \mathbf{0} \implies \phi(\mathbf{x}^*) = 0$
  - $\phi(\mathbf{x}) > 0, \quad \forall\ \mathbf{x} \neq \mathbf{x}^*$
- Minimising ϕ is equivalent to finding a root of $\mathbf{F}$
- Optimality condition:

$$\nabla\phi(\mathbf{x}^*) = \frac{\partial}{\partial \mathbf{x}}\phi(\mathbf{x}^*) = \mathbf{0} \quad \Leftrightarrow \quad 2\,J(\mathbf{x}^*)^T\mathbf{F}(\mathbf{x}^*) = \mathbf{0}.$$

Proof of gradient formula

For each j:

$$\frac{\partial}{\partial x_j}\phi(\mathbf{x}) = \frac{\partial}{\partial x_j}\sum_{i=1}^{n}(F_i(\mathbf{x}))^2 = 2\sum_{i=1}^{n}\frac{\partial F_i}{\partial x_j}(\mathbf{x})F_i(\mathbf{x})$$

$$= 2\sum_{i=1}^{n} J_{ij}(\mathbf{x})F_i(\mathbf{x}) = 2[J(\mathbf{x}^*)^T\mathbf{F}(\mathbf{x}^*)]_j$$

The gradient

- Provided ϕ is differentiable we can define the gradient at our current solution point $\mathbf{x}$
  $\nabla\phi = \left(\frac{\partial\phi}{\partial x_1}, \frac{\partial\phi}{\partial x_2}, ..., \frac{\partial\phi}{\partial x_n}\right)$
- This defines a local vector direction, at $\mathbf{x}$, along which the function $\phi(\mathbf{x})$ increases most strongly
- Conversely, $\phi(\mathbf{x})$ decreases most strongly in the opposite direction $\mathbf{d} = -\nabla\phi$

Gradient descent algorithm

- Initial state: $\mathbf{x} = \mathbf{x}_0$
- while $|\mathbf{F}(\mathbf{x}_k)| > Tol$
  - Find descent direction: $\mathbf{d} = -2\,\mathbf{J}^T(\mathbf{x}_k)\mathbf{F}(\mathbf{x}_k)$
  - Take descent step: $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}$
- We still require $\mathbf{J}(\mathbf{x}_k)$ at each iteration
- No linear system to solve, only matrix-vector multiplication
- No guarantee that $|\mathbf{F}(\mathbf{x}_{k+1})| < |\mathbf{F}(\mathbf{x}_k)|$, line-search required

Line search

- We have computed the direction to move $\mathbf{d}$ but not the distance α
  - $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\mathbf{d}$
- We can use a line search approach (see also last lecture)
  - In this case no upper limit on the distance $\alpha > 0$
  - Require descent steps $\phi(\mathbf{x}_{k+1}) < \phi(\mathbf{x}_k)$ as before
- More robust than Newton's Method, in particular for a poor initial guess
- Might converge only to a local minimum:

$$\nabla\phi(\mathbf{x}^*) = \mathbf{0}, \quad \text{but } \mathbf{F}(\mathbf{x}^*) \neq \mathbf{0}.$$

Gradient descent algorithm

- Initial state: $\mathbf{x} = \mathbf{x}_0$
- while $|\mathbf{F}(\mathbf{x}_k)| > Tol$
  - $\mathbf{d} = -2\,\mathbf{J}^T(\mathbf{x}_k)\mathbf{F}(\mathbf{x}_k)$
  - Perform line search to find optimal α
  - $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\mathbf{d}$

- We still require $\mathbf{J}(\mathbf{x}_k)$ at each iteration
- No linear system to solve
- Must perform line search for α

- Newton $\delta = -J^{-1}\mathbf{F}$
  Gradient descent $\mathbf{d} = -2J^T\mathbf{F}$

Notes on gradient descent

- Can be more aggressive
  Increasing $\alpha > 1$ to move further
- Switching to Newton's Method?
  (i) If gradient descent stalls, (ii) Once $\phi(\mathbf{x}_k) < tol_\phi$
  Switch for faster convergence
- Typical failure case when $\phi(\mathbf{x})$ has flat regions and algorithm can't find a good step size $\Rightarrow$ convergence stalls.