Conjugate direction methods

- When applied to quadratics of n variables, they converge in at most n steps.
- Usual implementations: need only gradient. No need to use Hessian.
- More complicated than steepest descent algorithm.

Conjugate vectors (§10.1)

- Given $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$, symmetric.
- Two vectors $\boldsymbol{d}^{(1)}$ and $\boldsymbol{d}^{(2)}$ are $\boldsymbol{Q}$-conjugate if $\boldsymbol{d}^{(1)T}\boldsymbol{Q}\boldsymbol{d}^{(2)} = 0$.
- The vectors $\boldsymbol{d}^{(1)}, \ldots, \boldsymbol{d}^{(m)}$ are $\boldsymbol{Q}$-conjugate if every pair of them are $\boldsymbol{Q}$-conjugate.
- If $\boldsymbol{Q} = \boldsymbol{I}$, conjugacy reduces to orthogonality.

Example:

- Let
$$\boldsymbol{Q} = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix}.$$
- Consider
$$\boldsymbol{d}^{(0)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \qquad \boldsymbol{d}^{(1)} = \begin{bmatrix} 1 \\ 0 \\ -3 \end{bmatrix}, \qquad \boldsymbol{d}^{(2)} = \begin{bmatrix} 1 \\ 4 \\ -3 \end{bmatrix}.$$
- The above vectors are $\boldsymbol{Q}$-conjugate.
- There are many sets of vectors that are $\boldsymbol{Q}$-conjugate.

Lemma (10.1): Suppose $\boldsymbol{Q} > 0$, $n \times n$. If the nonzero vectors $\boldsymbol{d}^{(0)}, \ldots, \boldsymbol{d}^{(k)}$ are $\boldsymbol{Q}$-conjugate, then they are linearly independent.

Proof:

- Suppose $\alpha_0, \ldots, \alpha_k$ satisfy
$$\alpha_0\boldsymbol{d}^{(0)} + \cdots + \alpha_k\boldsymbol{d}^{(k)} = \boldsymbol{0}.$$
- Want to show that $\alpha_0 = \cdots = \alpha_k = 0$.
- Premultiply equation by $\boldsymbol{d}^{(j)T}\boldsymbol{Q}$ to get
$$\alpha_j\boldsymbol{d}^{(j)T}\boldsymbol{Q}\boldsymbol{d}^{(j)} = 0.$$
- Since $\boldsymbol{Q} > 0$, we deduce that $\alpha_j = 0$.

Conjugate direction algorithm (§10.2)

- Consider the algorithm
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)},$$
where, as usual,
$$\alpha_k = \arg\min_{\alpha \geq 0} f(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}).$$

- Apply to quadratic:
$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{b},$$

- Recall formula for α_k in this case:
$$\alpha_k = -\frac{\boldsymbol{d}^{(k)T} \boldsymbol{g}^{(k)}}{\boldsymbol{d}^{(k)T} \boldsymbol{Q} \boldsymbol{d}^{(k)}}.$$

- Conjugate direction algorithm: the directions $\boldsymbol{d}^{(0)}, \boldsymbol{d}^{(1)}, \ldots$ are $\boldsymbol{Q}$-conjugate.

- The above defines a *family* of algorithms.

- Theorem (10.1): In a conjugate direction algorithm, we have
$$\boldsymbol{x}^{(n)} = \boldsymbol{x}^*$$
regardless of what $\boldsymbol{x}^{(0)}$ we start with.

Proof of theorem:

- Want to show $\boldsymbol{Q}\boldsymbol{x}^{(n)} = \boldsymbol{b}$. We have
$$\begin{aligned}
\boldsymbol{x}^{(n)} &= \boldsymbol{x}^{(n-1)} + \alpha_{n-1}\boldsymbol{d}^{(n-1)} \\
&= \boldsymbol{x}^{(n-2)} + \alpha_{n-2}\boldsymbol{d}^{(n-2)} + \alpha_{n-1}\boldsymbol{d}^{(n-1)} \\
&\vdots \\
&= \boldsymbol{x}^{(0)} + \alpha_0 \boldsymbol{d}^{(0)} + \cdots + \alpha_{n-1}\boldsymbol{d}^{(n-1)}.
\end{aligned}$$
Hence,
$$\boldsymbol{x}^{(n)} - \boldsymbol{x}^{(0)} = \alpha_0 \boldsymbol{d}^{(0)} + \cdots + \alpha_{n-1}\boldsymbol{d}^{(n-1)}$$

- Premultiply both sides by $\boldsymbol{d}^{(k)T}\boldsymbol{Q}$, where $0 \leq k \leq n-1$. All terms on the right hand side will vanish, except the kth.

- So

$$\begin{aligned}
&\boldsymbol{d}^{(k)T}\boldsymbol{Q}(\boldsymbol{x}^{(n)} - \boldsymbol{x}^{(0)}) \\
&\quad = \alpha_k \boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{d}^{(k)} \\
&\quad = -\boldsymbol{d}^{(k)T}\boldsymbol{g}^{(k)} \quad \text{by } \alpha_k \text{ formula} \\
&\quad = -\boldsymbol{d}^{(k)T}(\boldsymbol{Q}\boldsymbol{x}^{(k)} - \boldsymbol{b}) \\
&\quad = -\boldsymbol{d}^{(k)T}\boldsymbol{Q}(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{*}) \\
&\quad = -\boldsymbol{d}^{(k)T}\boldsymbol{Q}(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(0)} + \boldsymbol{x}^{(0)} - \boldsymbol{x}^{*}) \\
&\quad = -\boldsymbol{d}^{(k)T}\boldsymbol{Q}(\boldsymbol{x}^{(0)} - \boldsymbol{x}^{*}).
\end{aligned}$$

- Hence,

$$\begin{aligned}
\boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{x}^{(n)} &= \boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{x}^{*} \\
&= \boldsymbol{d}^{(k)T}\boldsymbol{b}.
\end{aligned}$$

- The equation

$$\boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{x}^{(n)} = \boldsymbol{d}^{(k)T}\boldsymbol{b}$$

  holds for $k = 0, \dots, n-1$.

- Because $\boldsymbol{d}^{(0)}, \dots, \boldsymbol{d}^{(n-1)}$ are linearly independent, we deduce that $\boldsymbol{Q}\boldsymbol{x}^{(n)} = \boldsymbol{b}$.

- We already know that $\boldsymbol{g}^{(k+1)T}\boldsymbol{d}^{(k)} = 0$.

- Lemma (10.2): In the conjugate direction algorithm,

$$\boldsymbol{g}^{(k+1)T}\boldsymbol{d}^{(i)} = 0$$

  for all $k = 0, \dots, n-1$, and $0 \le i \le k$.

- Proof: later.

- Interpretation:

$$f(\boldsymbol{x}^{(k+1)}) = \min_{a_0,\dots,a_k} f\left(\boldsymbol{x}^{(0)} + \sum_{i=0}^{k} a_i \boldsymbol{d}^{(i)}\right).$$

  Not only is α_k the best step size at the kth step, it is the best step size “overall.”

- Consider two iterations of the algorithm

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$$

  where $\boldsymbol{d}^{(0)}$ and $\boldsymbol{d}^{(1)}$ are given $\boldsymbol{Q}$-conjugate vectors.

- We know that because
$$f(\boldsymbol{x}^{(2)}) = \min_{\alpha} f(\boldsymbol{x}^{(1)} + \alpha \boldsymbol{d}^{(1)}),$$
we have $\boldsymbol{g}^{(2)T}\boldsymbol{d}^{(1)} = 0$.
- What additional information does $\boldsymbol{g}^{(2)T}\boldsymbol{d}^{(0)} = 0$ correspond to?
- Consider the function
$$\bar{\phi}(a_0, a_1) = f(\boldsymbol{x}^{(0)} + a_0\boldsymbol{d}^{(0)} + a_1\boldsymbol{d}^{(1)})$$
- Note that $\bar{\phi}(\alpha_0, \alpha_1) = f(\boldsymbol{x}^{(2)})$.
- By chain rule,
$$\nabla\bar{\phi}(\alpha_0, \alpha_1) = \begin{bmatrix} \boldsymbol{g}^{(2)T}\boldsymbol{d}^{(0)} \\ \boldsymbol{g}^{(2)T}\boldsymbol{d}^{(1)} \end{bmatrix}.$$
- Hence, $\boldsymbol{g}^{(2)T}\boldsymbol{d}^{(i)} = 0$ for $i = 0, 1$ corresponds to the FONC for the function $\bar{\phi}$.
- We have
$$\bar{\phi}(\alpha_0, \alpha_1) = \min_{a_0, a_1} \bar{\phi}(a_0, a_1).$$
- Note that after $k+1$ steps of the algorithm, the point $\boldsymbol{x}^{(k+1)}$ lies on the set
$$\mathcal{S}_k = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x} = \boldsymbol{x}^{(0)} + \boldsymbol{v},\ \boldsymbol{v} \in \mathcal{V}_k\}$$
where $\mathcal{V}_k = \text{span}[\boldsymbol{d}^{(0)}, \ldots, \boldsymbol{d}^{(k)}]$.
- The previous lemma tells us that
$$f(\boldsymbol{x}^{(k+1)}) = \min_{\boldsymbol{x} \in \mathcal{S}_k} f(\boldsymbol{x}).$$
- The subspace $\mathcal{V}_k$ is "expanding" as k increases.
- Eventually, it will expand so much that the global minimizer lies inside $\mathcal{S}_k$.
- At that time, $\boldsymbol{x}^{(k+1)}$ will be the minimizer!

Proof of "expanding subspace" lemma:

- To prove the lemma, we use induction on k.
- For $k = 0$, the lemma is true because $\boldsymbol{g}^{(1)T}\boldsymbol{d}^{(0)} = 0$ as we know from before.
- Assume true for $k-1$; i.e., $\boldsymbol{g}^{(k)T}\boldsymbol{d}^{(i)} = 0$ for $i = 0, \ldots, k-1$.

- Consider k. We already know that $\boldsymbol{g}^{(k+1)T}\boldsymbol{d}^{(k)} = 0$. So, it remains to show that $\boldsymbol{g}^{(k+1)T}\boldsymbol{d}^{(i)} = 0$ for $i < k$.

- Now,
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}.$$
Premultiplying by $\boldsymbol{Q}$ and subtracting $\boldsymbol{b}$, we obtain
$$\boldsymbol{g}^{(k+1)} = \boldsymbol{g}^{(k)} + \alpha_k \boldsymbol{Q}\boldsymbol{d}^{(k)}.$$

- For $i < k$, we have
$$\begin{aligned} \boldsymbol{g}^{(k+1)T}\boldsymbol{d}^{(i)} &= (\boldsymbol{g}^{(k)} + \alpha_k \boldsymbol{Q}\boldsymbol{d}^{(k)})^T \boldsymbol{d}^{(i)} \\ &= \boldsymbol{g}^{(k)T}\boldsymbol{d}^{(i)} + \alpha_k \boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{d}^{(i)} \\ &= 0, \end{aligned}$$
where $\boldsymbol{g}^{(k)T}\boldsymbol{d}^{(i)} = 0$ by induction hypothesis, and $\boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{d}^{(i)} = 0$ by $\boldsymbol{Q}$-conjugacy.

- Done!

Generating conjugate directions

- Conjugate direction algorithm:
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)},$$
where
$$\alpha_k = \operatorname*{arg\,min}_{\alpha \geq 0} f(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)})$$
$\boldsymbol{d}^{(0)}, \boldsymbol{d}^{(1)}, \ldots$ are $\boldsymbol{Q}$-conjugate.

- How do we generate the directions $\boldsymbol{d}^{(0)}, \boldsymbol{d}^{(1)}, \ldots$?

- For each k, we generate $\boldsymbol{d}^{(k+1)}$ based on current and past data. For example, $\boldsymbol{d}^{(k)}$, $\boldsymbol{g}^{(k)}$, and $\boldsymbol{g}^{(k+1)}$.

- We study two methods for generating successive directions $\boldsymbol{d}^{(0)}, \boldsymbol{d}^{(1)}, \ldots$:
  - Conjugate gradient method
  - Quasi-Newton method

Conjugate gradient algorithm (§10.3)

- Algorithm:
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)},$$
where
$$\alpha_k = \arg\min_{\alpha \geq 0} f(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}).$$
- We need a way to generate the $\boldsymbol{d}^{(k)}$ such that for a quadratic, they are $\boldsymbol{Q}$-conjugate.
- Conjugate gradient method: use gradient to generate $\boldsymbol{d}^{(k)}$.
- Update $\boldsymbol{d}^{(k)}$ according to formula:
$$\boldsymbol{d}^{(k+1)} = -\boldsymbol{g}^{(k+1)} + \beta_k \boldsymbol{d}^{(k)},$$
where by convention we take $\boldsymbol{d}^{(-1)} = \mathbf{0}$ (i.e., start with $\boldsymbol{d}^{(0)} = -\boldsymbol{g}^{(0)}$).
- The scalar β_k is computed using a formula involving $\boldsymbol{g}^{(k)}$, $\boldsymbol{g}^{(k+1)}$, and $\boldsymbol{d}^{(k)}$.

Easy way to compute β_k

- We need $\boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{d}^{(k+1)} = 0$.
- Hence,
$$0 = \boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{d}^{(k+1)} = -\boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{g}^{(k+1)} + \beta_k \boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{d}^{(k)}.$$
- We obtain
$$\beta_k = \frac{\boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{g}^{(k+1)}}{\boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{d}^{(k)}}.$$
- The above formula not immediately useful because it involves $\boldsymbol{Q}$. (How to apply to non-quadratics?)

Useful formulas for β_k:

- Hestenes-Stiefel formula:
$$\beta_k = \frac{\boldsymbol{g}^{(k+1)T}[\boldsymbol{g}^{(k+1)} - \boldsymbol{g}^{(k)}]}{\boldsymbol{d}^{(k)T}[\boldsymbol{g}^{(k+1)} - \boldsymbol{g}^{(k)}]}$$
- Polak-Ribiere formula:
$$\beta_k = \frac{\boldsymbol{g}^{(k+1)T}[\boldsymbol{g}^{(k+1)} - \boldsymbol{g}^{(k)}]}{\boldsymbol{g}^{(k)T}\boldsymbol{g}^{(k)}}.$$

- Fletcher-Reeves formula:

$$\beta_k = \frac{\boldsymbol{g}^{(k+1)T}\boldsymbol{g}^{(k+1)}}{\boldsymbol{g}^{(k)T}\boldsymbol{g}^{(k)}}$$

- The previous three formulas all lead to conjugate direction algorithms (i.e., the resulting directions are $\boldsymbol{Q}$-conjugate when applied to a quadratic with Hessian $\boldsymbol{Q}$). See book for proof.

- The conjugate gradient algorithm using the above formulas for β_k can be applied to any function f.

- If f is a quadratic, all the three formulas are equivalent.

- If f is not a quadratic, the algorithm will not usually reach the solution in n steps.

- For general f, the formulas have different performance. Performance highly dependent on f.

- If using sloppy line search, Hestenes-Stiefel formula is recommended.

- Modifications are possible. For example, Powell's formula (modification of Polak-Ribiere):

$$\beta_k = \max\left[0, \frac{\boldsymbol{g}^{(k+1)T}[\boldsymbol{g}^{(k+1)} - \boldsymbol{g}^{(k)}]}{\boldsymbol{g}^{(k)T}\boldsymbol{g}^{(k)}}\right].$$

