**EE 381V: Large Scale Optimization** **Fall 2012**

# Lecture 9 — September 27

*Lecturer: Caramanis & Sanghavi* *Scribe: Alan Bernstein, Lark Kwon Choi*

Note that the main reference for this lecture is Nocedal & Wright (Chapter 5 and 6).

## 9.1 Topics Covered Last Time

- Newton's Method
- Self-Concordant functions

In the last lecture, we covered Newton's Method and Self-Concordant functions, which are used as barrier functions for constrained optimization. This lecture considers Newton's Method for large scale problems.

## 9.2 Newton Method for Large Scale Optimization

The Newton step, $\Delta x_{\text{nt}}$, is a solution to the minimization of a quadratic approximating function $\hat{f}(x + \Delta x_{\text{nt}})$:

$$\hat{f}(x + \Delta x_{\text{nt}}) \triangleq f(x) + \nabla f(x)^T \Delta x_{\text{nt}} + \Delta x_{\text{nt}}^T \nabla^2 f(x) \Delta x_{\text{nt}}.$$

and also the solution to the linear system:

$$\nabla^2 f(x) \Delta x_{\text{nt}} = \nabla f(x).$$

The idea behind adapting Newton's Method comes from steepest descent, where the step direction is determined using a different norm, $||\cdot||_B$,

$$\Delta x_{\text{sd}} = -B \nabla f(x),$$

where $B$ is some positive definite matrix. Now, in Newton's method, a new, optimal, norm is chosen at each time step:

$$\Delta x_{\text{k}} = -B_k \Delta f(x).$$

Note that for $B_k = \nabla^2 f(x_k)^{-1}$, the optimal choice, this is equivalent to the Newton step. But this choice requires computing and inverting the Hessian; is there a better choice that converges faster than a fixed-norm approach, but is computationally cheaper than an optimal approach?

There are many *variable metric* methods for solving this problem. These use a different norm at each step, to avoid calculation of second derivatives and simplify the calculation of the search direction. The basic idea of variable metric methods is to iteratively construct a good approximation to the inverse Hessian by building a sequence of matrices. Among many methods, this lecture focuses on two main ideas:

1. Conjugate Gradient
   The conjugate gradient method is used to solve large linear system of equations and nonlinear optimization problems (*linear* and *nonlinear conjugate gradient* methods, respectively). The conjugate gradient method is the most widely used iterative method for solving $Ax = b$, with $A \succ 0$ and can be extended to non-quadratic unconstrained minimization. It is a little less reliable than exact Newton methods, but can handle very large problems.

2. Approximate Solution
   Approximation of the solution in a way that makes sense for the Hessian, as opposed to, e.g., least-squares. Finding $B_k$ in the Quasi-Newton methods come in two main flavors. One is the *Davidon-Fletcher Powell (DFP)* algorithm (sometimes referred to as simply Fletcher-Powell). The other is *Broyden-Fletcher-Goldfarb-Shanno (BFGS)*. DFP finds a solution that is symmetric, positive definite and closest to the current approximate value of $B_k$. BFGS was derived from DFP, which was popular until BFGS was introduced. Instead of approximating the Hessian, $B_k$, BFGS approximates its inverse, $H_k$. Rather than solving a linear system to get the search direction, it executes a matrix multiply.

## 9.3 Conjugate Gradient & Solving Linear Systems

The conjugate gradient method is an iterative method for solving a linear system of equations

$$Ax = b,$$

where A is symmetric and positive definite. This is a natural assumption when $A$ is the Hessian of a convex function. Note that this is equivalent to the following minimization problem:

$$\min_x \phi(x) = \tfrac{1}{2} x^T A x - bx,$$

since the problem is convex, and the optimality condition $\nabla\phi(x) = 0$ gives

$$\nabla\phi(x_k) \equiv Ax_k - b = r_k \equiv 0,$$

at a point $x_k$, $\nabla\phi(x_k) = Ax_k - b$, which is the residual error. We denote this as $r_k = Ax_k - b$.

**Definition 1. (Conjugacy of vectors)** *A set of nonzero vectors $\{p_0, p_1, ..., p_l\}$ is called conjugate with respect to the symmetric positive semidefinite matrix $A$ if*

$$p_i^T A p_j = 0, \quad i \neq j. \tag{9.1}$$

A set of conjugate vectors can be used to sequentially solve a linear system. Given a starting point $x_0$, and conjugate vectors $\{p_i\}$,

$$x_{k+1} = x_k + \eta_k p_k, \tag{9.2}$$

where $\eta_k$ is the one-dimensional minimizer of the quadratic function $\phi$, given by

$$\eta_k = \frac{-r_k^T p_k}{p_k^T A p_k}. \tag{9.3}$$

We refer to this as the "baby conjugate gradient" method, or B-CG. A basic result from linear algebra is that:

**Theorem 9.1.** *For any $x_0 \in R^n$ the sequency $\{x_k\}$ generated by 9.2, 9.3 converges to the solution $x^*$ of the linear system in at most $n$ steps.*

**Proof:** Since the direction $p_i$ are linearly independent, they must span the whole space $R^n$. Hence,

$$x^* - x_0 = \sigma_0 p_0 + \sigma_1 p_1 + ... + \sigma_{n-1} p_{n-1},$$

for some choice of scalars $\sigma_k$. From 9.1, we obtain

$$\sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k}. \tag{9.4}$$

The coefficient $\sigma_k$ coincide with the step length $\eta_k$ generated by the formula 9.3.
Then,

$$x_k = x_0 + \eta_0 p_0 + \eta_1 p_1 + ... + \eta_{k-1} p_{k-1}.$$

By premultiplying this expression by $P_k^T A$ and using the conjugacy property,

$$p_k^T A(x_k - x_0) = 0,$$

finally, we have

$$p_k^T A(x^* - x_0) = p_k^T A(x^* - x_k) = p_k^T (b - Ax_k) = -p_k^T r_k.$$

By comparing above equations 9.3 and 9.4, $\sigma_k = \eta_k$, giving the result. □

**Exercise 1.** *Show that Baby Conjugate Gradient is coordinate descent with exact line search.*

**Proof:** Let's consider a simple case. Suppose $A$ is symmetric. By the spectral theorem, $A$ is diagonalizable. Suppose $p_i = e_i$. Then, the contours of the function $\phi(\cdot)$ are ellipses whose axes are aligned with the coordinate directions. We can find the minimizer of this function by performing one-dimensional minimizations along the coordinate directions $e_1, e_2, ..., e_n$ in turn. When we change the problem by using new variable as $\hat{x} = T^{-1}x$, with n×n matrix $T$ defined by $T = [p_0, p_1, ..., p_{n-1}]$, and a set of conjugate directions with respect with respect to A,$\{p_0, p_1, ..., p_{n-1}\}$, then, the quadratic function $\phi$ becomes

$$\hat{\phi}(\hat{x}) = \phi(T\hat{x}) = \frac{1}{2}\hat{x}^T(T^TAT)\hat{x} = (T^Tb)^T\hat{x}.$$

By the conjugacy property 9.1, the matrix $T^TAT$ is diagonal. Hence we can find the minimizing value of $\hat{\phi}$ by performing $n$ one-deimesional minimizations along the coordinate directions of $\hat{x}$, which is corresponding to the direction $p_i$ in x-space. Therefore, The Baby Conjugate Gradient method is coordinate descent with exact line search in a new coordinate system. □

**Exercise 2.** *Show that $p_i$ is the $i^{th}$ eigenvector of A.*

**Proof:** We can prove above by showing $r_k^Tp_i = 0$, for $i = 0, ..., k-1$. Since $\eta_k$ is always the one dimensional minimizer, we have immediately that $r_1^Tp_0 = 0$. By using induction hypothesis and the conjugacy of $p_i$, we can conclude that $r_k^Tp_i = 0$. □

**Note 1.** *To apply the above idea, we need to compute the complete set of eigenvectors. This might be harder than simply solving the entire system to begin with. In addition, it is not practical for large scale applications. There is a missing step; finding $\{p_k\}$, the conjugate directions, with minimal computational requirements.*

In generating its set of conjugate vectors, the conjugate gradient method picks a new vector sequentially. Namely, each direction $p_k$ is chosen to be a linear combination of negative residual $-r_k$ and the previous direction $p_{k-1}$ as follows,

$$p_k = -r_k + \beta_k p_{k-1}. \tag{9.5}$$

**Exercise 3.** *Show that*

$$\beta_k = \frac{r_k^TAp_{k-1}}{p_{k-1}^TAp_{k-1}}. \tag{9.6}$$

**Proof:** The scalar $\beta_k$ is to be determined by the requirement that $p_{k-1}$ and $p_k$ must be conjugate with respect to A. By premultiplying 9.5 by $p_{k-1}^TA$ and imposing the condition $p_{k-1}^TAp_k = 0$, we obtain 9.6. □

### 9.3.1 Conjugate Gradient for analysis

- Conjugate Gradient Preliminary Version

We add a step to compute $p_{k+1}$. This version of the algorithm is useful for analysis. We will introduce a second version which is useful for implementation.

Initialize:

$$\begin{aligned} r_0 &= Ax_0 - b, \\ p_0 &= -r_0, \\ k &= 0. \end{aligned}$$

while $r_k \neq 0$:

$$\begin{aligned} \eta_k &= \frac{-r_k^T p_k}{p_k^T A p_k}, \\ x_{k+1} &= x_k + \eta_k p_k, \\ r_{k+1} &= Ax_{k+1} - b, \\ \beta_{k+1} &= \frac{r_{k+1}^T A p_k}{p_k^T A p_k}, \\ p_{k+1} &= -r_{k+1} + \beta_{k+1} p_k. \end{aligned} \tag{9.7}$$

end(while)

Note that the algorithm depends on matrix-vector multiplications, but no matrix inversions. We can show that this algorithm has several properties:

**Theorem 9.2.** *Suppose $r_k \neq 0$, then the following four properties hold:*

1. *$r_k \perp r_i$, i.e., $r_k^T r_i = 0$, for $i = 0, 1, ..., k-1$,*
2. *$\text{span}\{r_0, r_1, ..., r_k\} = \text{span}\{r_0, Ar_0, ..., A^k r_0\}$,*
3. *$\text{span}\{p_0, p_1, ..., p_k\} = \text{span}\{r_0, Ar_0, ..., A^k r_0\}$,*
4. *$\{p_i\}$ conjugate with respect to $A$, i.e., $p_k^T A p_i = 0$, for $i = 0, 1, ..., k-1$.*

**Proof:** First, let's prove Theorem 9.2-1. Because the direction set is conjugate, $r_k^T p_i = 0$ for all $i = 0, 1, ..., k-1$ and any $k = 1, 2, ...n-1$. By rearranging 9.7, we find that $p_i = -r_i + \beta_i p_{i-1}$, so that $r_i \in \text{span}\{p_i, p_{i-1}\}$ for all $i = 1, 2, ..., k-1$. Hence, we conclude that $r_k^T r_i = 0$ for all $i = 1, ..., k-1$.
Other proofs are by induction. Theorem 9.2-2 and 9.2-3 hold for $k = 0$, and 9.2-4 holds for $k = 1$. Assuming these properties are true for some $k$, then we show that they hold for

$k + 1$. First, we show that the set on the left-hand side is included in the set on the right side. From induction hypothesis,

$$r_k \in \text{span}\{r_0, Ar_0, ..., A^k r_0\}, \qquad p_k \in \text{span}\{r_0, Ar_0, ..., A^k r_0\}.$$

By multiplying the second expression by A,

$$Ap_k \in \text{span}\{Ar_0, ..., A^{k+1} r_0\}.$$

Then, $r_{k+1} \in \text{span}\{r_0, Ar_0, ..., A^{k+1} r_0\}$ and by combining this with the induction hypothesis of Theorem 9.2-2, we conclude $\text{span}\{r_0, r_1, ..., r_k\} = \text{span}\{r_0, Ar_0, ..., A^k r_0\}$. To prove the reverse inclusion, we use induction hypothesis of Theorem 9.2-3 to deduce that

$$A^{k+1} r_0 = A(A^k r_0 \in \text{span}\{Ar_0, ..., A^{k+1} r_0\}.$$

Since $Ap_i = \frac{r_{i+1} - r_i}{\eta}$, it follows that

$$A^{k+1} r_0 \in \text{span}\{r_0, ..., r^{k+}, r^{k+1}\}.$$

Similarly, via the induction hypothesis of Theorem 9.2-2, the Theorem 9.2-2 continues to hold when $k$ is replaced by $k + 1$.
We can prove Theorem 9.2-3 by the following argument:

$$\begin{aligned}
&\text{span}\{p_0, p_1, ..., p_k, p_{k+1}\} \\
&\qquad = \text{span}\{p_0, p_1, ..., p_k, r_{k+1}\} \\
&\qquad = \text{span}\{r_0, Ar_0, ..., A^k r_0, r_{k+1}\} \\
&\qquad = \text{span}\{r_0, r_1, ..., r_k, r_{k+1}\} \\
&\qquad = \text{span}\{r_0, Ar_0, ..., A^{k+1} r_0\}.
\end{aligned}$$

Next, we prove Theorem 9.2-4 by multiplying 9.7 by $Ap_i, i = 0, 1, ..., k$ we obtain that

$$p_{k+1}^T Ap_i = -r_{k+1}^T Ap_i + \beta_{i+1} p_k^T Ap_i. \tag{9.8}$$

By the definition of $\beta_k$, the right hand of 9.8 vanished when $i = k$. In addition, since $p_0, p_1, ..., p_k$ are conjugate, $r_{k+1}^T p_i = 0$ for $i = 0, 1, ..., k - 1$. By repeatedly applying Theorem 9.2-3, we can find that

$$Ap_i \in \text{span}\{Ar_0, A^2 r_0, ..., A^{i+1} r_0\} \subset \text{span}\{p_0, p_1, ..., p_{i+1}\}.$$

Therefore, we can deduce that $r_{k+1}^T Ap_i = 0$ for $i = 0, 1, ..., k - 1$. Consequently, the first and second term in the right side of 9.8 vanishes, and we conclude that $p_k^T Ap_i = 0$, for $i = 0, 1, ..., k - 1$.

$\square$

- Conjugate Gradient Practical Version

Another version of conjugate gradient, which differs only in two steps:

$$\eta_k = \frac{r_k^T r_k}{p_k^T A p_k}, \qquad \beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}.$$

**Theorem 9.3.** *If $A$ has only $d$ distinct eigenvalues ($A$ is possibly still full rank), then the conjugate gradient algorithm terminates at the exact solution in at most $d$ steps.*

**Theorem 9.4.** *If $A$ has eigenvalues $0 \leq \lambda_1 \leq ... \leq \lambda_k$ then at iteration $k+1$,*

$$||x_{k+1} - x^*||_{\mathrm{A}}^2 \leq \left(\frac{\lambda_{n-k} - \lambda_i}{\lambda_{n-k} + \lambda_1}\right)^2 ||x_0 - x^*||_{\mathrm{A}}^2.$$

**Theorem 9.5.** *If $A$ is positive semidefinite with condition number, then $\kappa(A) = \frac{M}{m}$,*

$$||x_k - x^*||_{\mathrm{A}} \leq 2\left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^k ||x_0 - x^*||_{\mathrm{A}}.$$

**Remark 1.** *These results come from important property. The conjugate gradient algorithm produces a solution over an "increasing" sequence of affine spaces:*

$$x_0 + \mathrm{span}\{p_0, ..., p_n\} = x_0 + \underbrace{\mathrm{span}\{r_0, Ar_0, A^2 r_0, ..., A^n r_0\}}_{\mathcal{K}_n(A, r_0)}.$$

**Definition 2.** *$\mathcal{K}_n(A, r_0)$ is a* Krylov subspace.

When the distribution of the eigenvalues of $A$ has certain favorable features, the algorithm will identify the solution in many fewer than $n$ iterations. From theorem 9.2-3, we have that

$$x_{k+1} = x_0 + \eta_0 p_0 + ... + \eta_k p_k = x_0 + \gamma_0 r_0 + \gamma_1 A r_0 + ... + \gamma_k A^k r_0. \tag{9.9}$$

We can define $P_k^*$ to be a polynomial of degree $k$ with coefficients $\gamma_0, \gamma_0, ..., \gamma_k$, and we have

$$P_k^*(A) = \gamma_0 I + \gamma_1 A + ... + \gamma_k A^k.$$

Then, 9.9 can be $x_{k+1} = x_0 + P_k^*(A) r_0$. Among all possible methods whose first $k$ steps are restricted to the Krylov subspace $\mathcal{K}_n(r_0, k)$, Conjugate Gradient does the best job of minimizing the distance to the solution after $k$ stpes, when the distance is measured the weighted norm defined by $||Z||_{\mathrm{A}}^2 = Z^T A Z$. Using this norm we get

$$\tfrac{1}{2}||x - x^*||_A^2 = \tfrac{1}{2}(x - x^*)^T A (x - x^*) = \phi(x) - \phi(^*).$$

This states that $x_{k+1}$ minimizes $\phi$, and hence $||x - x^*||_A^2$, over the set $x_0 + \eta_0 p_0 + ... + \eta_k p_k$. It follows the polynomial $P_k^*$ solves the problem, $\min_{P_k} ||x_0 + P_k(A) r_0 - x^*||_A$. Let $0 \leq$

$\lambda_1 \leq ... \leq \lambda_n$ be the eigenvalues of $A$, and let $v_1, v_2, ..., v_n$ be the corresponding orthonormal eigenvectors. Since these eigenvectors span the whole space $R^n$, we can write for some coefficients $\xi_i$ that

$$||x_{k+1} - x^*||_A^2 = \sum \lambda_i \left[I + \lambda_i P_k^*(\lambda_i)\right]^2 \xi_i^2.$$

Since the polynomial $P_k^*$ generated by the Conjugate Gradient method is optimal, we have the following expressions

$$\begin{aligned} ||x_{k+1} - x^*||_A^2 &= \min_{P_k} \sum \lambda_i \left[I + \lambda_i P_k^*(\lambda_i)\right]^2 \xi_i^2 \\ &\leq \min_{P_k} \max_{1\leq i\leq n} \left[I + \lambda_i P_k^*(\lambda_i)\right]^2 \left(\sum \lambda_i \xi_i^2\right) \\ &\leq \min_{P_k} \max_{1\leq i\leq n} \left[I + \lambda_i P_k^*(\lambda_i)\right]^2 ||x_0 - x^*||_A^2, \end{aligned}$$

and finally we can quantify the convergence rate of the Conjugate Gradient method by estimating the nonnegative scalar quantity

$$\min_{P_k} \max_{1\leq i\leq n} \left[I + \lambda_i P_k^*(\lambda_i)\right]^2. \tag{9.10}$$

This means we search for a polynomial $P_k$ that makes $\left[I + \lambda_i P_k^*(\lambda_i)\right]^2$ as small as possible.

**Note 2.** *Theorem 9.3 is strongest, but less useful, because it depends on a significant constraint on A, which is difficult to achieve. In particular, a tiny perturbation would destroy that property, where as it would not affect assumptions for Theorems 9.4 and 9.5. These theorems are in decreasing order of strength, in that the first depends on having the most information about the matrix A.*

## 9.4 Nonlinear Conjugate Gradient

We have studied linear conjugate gradient to solve $Ax = b$ or minimize $\phi(x) = \frac{1}{2}x^TAx - b^Tx$. Now, we extend this to a general algorithm for minimizing nonlinear functions. This extension is possible via two changes:

1. Observe that $r_k = Ax - b$ is the gradient of the quadratic function, so just use $r_k = \nabla\phi(x_k)$ or $r_k = \nabla f(x_k)$
2. The linear case essentially used exact line search to find $\beta_k$, but now we need to use something suitable for nonlinear functions, like backtracking line search.

The modified algorithm is as follows:

```
Initialize:
                $f_0 = f(x_0), \nabla f_0 = \nabla f(x_0),$
                $p_0 = -\nabla f_0,$
                $k = 0.$
while $\nabla f_k \neq 0,$
                Compute $\eta_k$, set $x_{k+1} = x_k + \eta_k p_k,$
                Evaluate $\nabla f_{k+1},$
                $\beta_{k+1}^{FR} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k},$
                $p_{k+1} = -\Delta f_{k+1} + \beta_{k+1} p_k,$
                $k = k + 1.$
end(while)
```

Where one of several well-known method for finding $\beta_{k+1}$, the Fletcher-Reeves method, has been used.
Each iteration requires only evaluation of the objective function and its gradient. No matrix operations are performed and storage of just a few vectors is required. To complete the specification of the nonlinear conjugate gradient algorithm, we need to be more precise about the choice of line search parameters, and there are computational issues:

**Issue 1.** *The line search used must guarantee that we have a descent method. For this, we need the* strong Wolfe conditions*:*

1. $f(x_k + \eta_k p_k) \leq f(x_k) + \alpha \eta_k \nabla f(x_k)^T p_k,$
2. $\left|\nabla f(x_k + \eta_k p_k)^T p_k\right| \leq \tilde{\alpha} \left|\nabla f(x_k)^T p_k\right|$, *where* $0 < \alpha < \tilde{\alpha} < \frac{1}{2}$.

   *Any line search procedure that yields an $\eta_k$ satisfying the above conditions will ensure that all directions $p_k$ are descent directions for the function $f$.*

**Issue 2.** *Computational issues:*

1. *The algorithm may not converge in $n$ steps*
2. *Progress might be very small under some conditions.*

*There are some pitfalls, which are covered in the homework.*

**Example 1.** *Consider the non-convex function $f(x)$ below, known as the Rosenbrock function. It has a unique minimizer $x^* = (1,1)$ and in a neighborhood of this point, the Hessian is positive definite. For $x_{init} = (-1.2, 1)$, the convergence and error vs iteration are shown in Figure 9.1.*

$$f(x) = 100(x_x - x_1^2)^2 + (1 - x_1)^2.$$

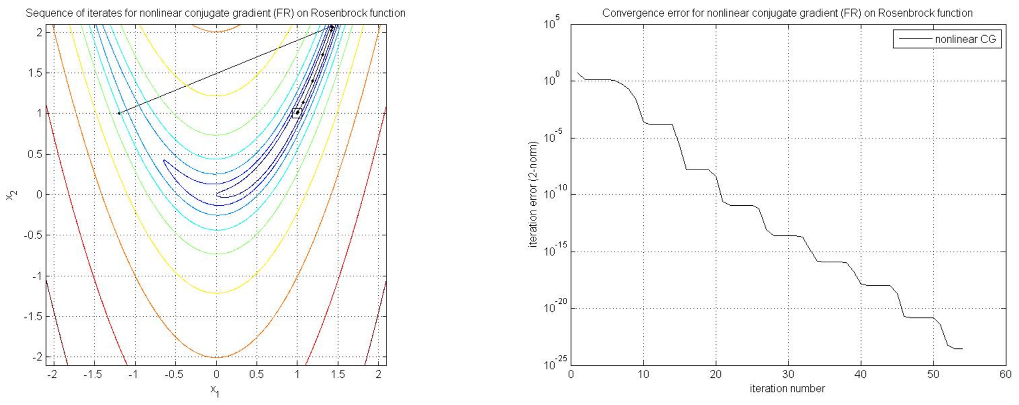

**Figure 9.1.** Convergence and error vs iteration for the Rosenbrock function using the nonlinear Conjugate Gradient method.

## 9.5 BFGS & DFP

1. Up until now, we have covered the first section of the lecture: how to exactly or approximately solve the equality constraint, $\nabla^2 f(x)\Delta x_{nt} = -\nabla f(x)$, for Newton's method, in a computationally inexpensive way, and how to adapt that technique nonlinear problems.

2. In the following, we discuss Quasi-Newton methods: $B_k \Delta x = -\nabla f$.

BFGS (Broyden-Fletcher-Goldfarb-Shanno) is derived in a natural way from DFP (Davidon-Fletcher-Powell), which was popular until BFGS was developed. The key idea is that $B_k$ should be an "approximate Hessian" in some sense. Specifically, $B_{k+1}$ is chosen so that $\nabla \hat{f}_{k+1}$ agrees with $\nabla f$ at $x_k$ and $x_{k+1}$. What conditions does this impose on $B_k$? For agreement at $x_{k+1}$, none, as this is achieved by construction. For agreement at $x_{k+1}$, this implies that $B_{k+1}(x_{k+1} - x_k) = (\nabla f(x_{k+1}) - \nabla f(x_k))$. This leads to the update:

$$\begin{aligned} s_k &\triangleq x_{k+1} - x_k, \\ y_k &\triangleq \nabla f(x_{k+1}) - \nabla f(x_k), \\ B_{k+1} &\triangleq \text{argmin}||B - B_k||, \\ &\quad \text{s.t. } Bs_k = y_k, B \succ 0. \end{aligned}$$

Different norms here correspond to different Quasi-Newton methods. This appears to be another difficult problem to solve, but if we use a weighted Frobenius norm for $||\cdot||$, then this new optimization problem can be solved in closed-form, and the updates are easy to compute.

In the next lecture, we will focus on methods for approximating Newton's method by obtaining recursive approximations to the Hessian at each step.