

---

## Lecture 20

### 1 Linear Equations

Gaussian elimination solves the linear system  $Ax = b$  in time  $O(n^3)$ . If  $A$  is a symmetric positive semidefinite matrix then there are iterative methods for solving linear systems in  $A$ . Each iteration in an iterative method amounts to matrix vector multiplications, the number of iterations for convergence to a vector at distance  $\epsilon$  from the true solution for these methods is bounded in terms of the condition number  $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$ ,

(i) Steepest descent:  $O(\kappa \log 1/\epsilon)$

(ii) Conjugate gradient:  $O(\sqrt{\kappa} \log 1/\epsilon)$ .

Preconditioning tries to improve the performance of iterative methods by multiplying by a matrix  $B^{-1}$  such that the system  $B^{-1}Ax = B^{-1}b$  is ‘easy’ to solve. Trivially  $A$  is the best preconditioner for the system, but we are looking for a matrix that is ‘easy’ to invert. Intuitively finding a preconditioner is like finding a matrix that is like  $A$  but easy to invert.

We will review the iterative methods, then use a low stretch spanning tree as a preconditioners for solving Laplacian linear systems, and finally see that a preconditioner obtained by adding some edges to the tree preconditioner achieves a nearly linear time solution.

#### 1.1 Steepest Descent

The steepest descent method is the analog of finding the minima of a function by following the derivative in higher dimensions. The solution to  $Ax = b$  is the point in  $\mathbb{R}^n$  where the quadratic form  $f(x) = \frac{1}{2}x^T Ax - bx + c$  achieves the minimum value. This can be verified by expanding  $f(x)$  and setting the partial derivatives to be 0. The positive semidefinite constraint ensures that the solution to  $Ax = b$  is a local minimum for  $f(x)$  and not a saddle point.

The steepest descent method is an iterative method that starts at an arbitrary point  $x_0$  and obtains  $x_{i+1}$  from  $x_i$  by moving in direction  $r_i$  opposite to the gradient. The gradient is the direction of steepest increase for the value of  $f(x)$  locally, the goal is to minimize  $f(x)$  so we move in a direction opposite to the gradient. The update rule is,

$$x_{i+1} = x_i + \alpha_i r_i \tag{1}$$

Here  $r_i = b - Ax_i$  is the direction opposite to the gradient and we choose  $\alpha_i$  such that  $r_{i+1} \cdot r_i = 0$ . The geometric intuition for the choice of  $\alpha$  is the following: As we move in the direction  $r_i$  opposite to the gradient, the gradient starts changing, as long as the projection of the gradient onto the direction  $r_i$  is negative the value of the objective function decreases. The transition point occurs when the new gradient is orthogonal to the current direction.

The value of  $\alpha$  corresponding to the orthogonality condition can be calculated easily with some algebra,

$$\begin{aligned} r_{i+1} = b - Ax_{i+1} &= b - A(x_i + \alpha r_i) = r_i - \alpha_i A r_i \\ r_i^T \cdot r_{i+1} &= 0 \Rightarrow \alpha_i = \frac{r_i^T r_i}{r_i^T A r_i} \end{aligned} \quad (2)$$

Gradient descent is a popular strategy even for non linear problems, the computation of  $\alpha$  is simple for the linear case. An iteration of gradient descent can be implemented in time  $O(m)$  where  $m$  is the number of non zero entries in  $A$ . Each iteration requires some matrix vector multiplications, the number of iterations needed to converge to the solution is the main issue.

Consider a two dimensional example where the solution lies at the center of an ellipse with axes  $\lambda_1$  and  $\lambda_2$  and we start from some point on the boundary close to the longer axis. Convince yourself by drawing the figure that the gradient descent requires about  $\lambda_2/\lambda_1$  steps to reach the center.

In fact the gradient descent converges in  $O(\kappa)$  iterations, let  $e_i$  be the error vector  $x_i - x^*$  for iteration  $i$ . The error  $e_{i+1}$  can be expressed in terms of  $e_i$  as follows,

$$\begin{aligned} e_{i+1} &= x_{i+1} - x^* = x_i + \alpha_i(b - Ax_i) - x^* \\ &= (x_i - x^*) + \alpha_i(Ax^* - Ax_i) \\ &= (I - \alpha_i A)e_i \end{aligned} \quad (3)$$

For each iteration in the gradient descent method, we choose the optimal value  $\alpha_i$ , therefore analyzing the rate of convergence for any fixed value of  $\alpha$  suffices.

Expanding  $e_0 = \sum \beta_j v_j$  in the spectral basis for  $A$  we have the following bound on the error vector,

$$e_i = (I - \alpha A)^i e_0 = \sum_{j \in [n]} (1 - \alpha \lambda_j)^i \beta_j v_j \quad (4)$$

The rate of convergence is dominated by the value of  $\max(|1 - \alpha \lambda_{min}|, |1 - \alpha \lambda_{max}|)$  and as we are free to choose  $\alpha$  we select  $\alpha = \frac{2}{\lambda_{min} + \lambda_{max}}$  in order to make both the numbers equal to  $\frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}}$ . With this choice of  $\alpha$  we have the error bound,

$$|e_i|^2 \leq \left(1 - \frac{2\lambda_{min}}{\lambda_{max} + \lambda_{min}}\right)^{2i} |e_0|^2 \leq e^{-\frac{4i\lambda_{min}}{\lambda_{max} + \lambda_{min}}} |e_0|^2 \quad (5)$$

Choosing  $i = O(\kappa)$  we note that the Euclidean norm of the error vector decreases by a constant factor for every  $\kappa$  iterations, assuming that the initial error is bounded the method produces a solution within  $\epsilon$  of the true solution in  $O(\kappa \log 1/\epsilon)$  iterations.

The conjugate gradient performs better and converges in  $O(\sqrt{\kappa} \log 1/\epsilon)$  iterations, we will not prove this but note that the rate of convergence is controlled by the condition number.

## 1.2 Preconditioning

The notion of preconditioning involves finding matrices that are ‘easy’ to invert and ‘similar’ to  $A$ , let us make both these notions precise.

*Easy to invert:* The adjacency matrices of weighted trees can be inverted easily by back substitution. Consider the linear system  $A_T x = b$  where  $A_T$  is the adjacency matrix of a tree. A leaf in the tree corresponds to a row in the adjacency matrix having exactly one non zero entry, the equation corresponding to a leaf is of the form  $w_i x_i = b_i$ . Trees have be decomposed by removing leaves one at a time, hence the linear system can be solved by substituting the values sequentially into the remaining equations.

The inverse of a tree is a dense matrix and we do not explicitly compute the inverse. Easy to invert means that there is a linear time procedure to find  $A_T^{-1} z$  given  $z$ . The efficient inversion procedure guarantees that each step of the preconditioned gradient descent for  $A_T^{-1} A x = A_T^{-1} b$  can be implemented in  $O(m)$  time. The easy invertibility remains valid for the Laplacian matrix of trees.

*Similar to  $A$ :* The matrix  $B$  is ‘similar’ to  $A$  if multiplying by  $B^{-1}$  reduces the condition number, for example multiplying by  $A$  reduces the condition number to 1. Positive semidefinite matrices are associated with quadratic forms, if  $x^t B x \geq x^t A x$  for all  $x \in \mathbb{R}^n$  then we write,

$$B \succeq A \tag{6}$$

Note that the matrix  $B$  is positive semidefinite is equivalent to  $B \succeq 0$ . The next claim shows that establishing a matrix inequality  $kB \succeq A \succeq B$  suffices to show that the condition number of  $B^{-1}A$  is at most  $k$ ,

CLAIM 1

If  $kB \succeq A \succeq B \succeq 0$  then all the eigenvalues of  $B^{-1}A$  lies in the interval  $[1, k]$ .

PROOF: We will need two observations from linear algebra for the proof. (i) A positive semidefinite matrix has a well defined square root, this follows as  $A = \sum \lambda_i v_i v_i^T$  is diagonal in the spectral basis, the square root  $A^{1/2} = \sum \sqrt{\lambda_i} v_i v_i^T$  is well defined as all the eigenvalues are positive. (ii) The matrices  $AB$  and  $BA$  have the same eigenvalues for all  $A$  and  $B$ , one way to see this is to note that  $ABx = \lambda x$  implies that  $BA(Bx) = \lambda(Bx)$ .

It therefore suffices to prove the claim for the eigenvalues of  $B^{-1/2}AB^{-1/2}$ , we will use the Rayleigh quotient characterization of the eigenvalues,

$$\frac{x^T B^{-1/2} A B^{-1/2} x}{x^T x} = \frac{y^T A y}{y^T B y} \in [1, k] \tag{7}$$

We used the substitution  $B^{-1/2}x = y$ , the claim follows as an interval containing the Rayleigh quotients contains the eigenvalues.  $\square$

## 1.3 Preconditioning by Trees

A spanning tree  $T$  would be a good preconditioner for graph  $G$  if we could prove a graphic inequality of the form  $kL_T \succeq L_G \succeq L_T$ . Clearly  $L_G \succeq L_T$  as  $T$  is a subgraph of  $G$ , and we need to establish an inequality of the form  $kL_T \succeq L_G$ .

Inequalities of this kind are proved by the following application of the Cauchy Schwarz inequality for a path of length  $k$ ,

$$(x_1 - x_{k+1})^2 \leq k \cdot ((x_1 - x_2)^2 + (x_2 - x_3)^2 + \cdots + (x_k - x_{k+1})^2) \quad (8)$$

Rewriting as a graphic inequality for edge  $(u, v) \in G$ , we have,

$$L_{(u,v)} \preceq d_T(u, v) L_T \quad (9)$$

Adding up all the above inequalities over all the edges in  $G$  we have  $L_G \preceq st(T) L_T$  where the total stretch  $st(T) = \sum_{(u,v) \in E} d_T(u, v)$ . If the tree is weighted the expression  $d_T(u, v)$  in the inequality (9) gets replaced by  $\sum \frac{1}{w_i}$  where  $w_i$  are the weights of the edges on the path connecting  $(u, v)$  in the tree  $T$ .

A weighted spanning tree with total stretch  $\tilde{O}(m \log n)$  can be found in almost linear time as stated in the the previous lecture. The conjugate gradient method run with this choice of preconditioner converges in  $\tilde{O}(\sqrt{m})$  iterations each iteration requiring time  $O(m)$ . A linear system solver running in time  $O(m^{3/2})$  performs better than Gaussian elimination for sparse graphs. In the next lecture we will see how to improve the running time to almost linear in  $m$ .