Түйіндес градиент әдісі

$$f(x) = \frac{1}{2}x^T Ax - b^T x \quad (4)$$

скаляр функциясын минимизациялайтын *x* векторын табу есебін қарастырайық, мұндағы *A* матрицасы симметриялы және оң анықталған. Оның градиенті $\nabla f = Ax - b$ нөлге тең болғанда $f(x)$ минимизацияланатындықтан, минимизациялау есебі

$$Ax = b \quad (5)$$

шешіміне эквивалент екенін көреміз.

Градиенттік әдістер бастапқы вектор x_0 бастап итерация арқылы минимизациялауды жүзеге асырады. Әрбір қайталанатын цикл *k* нақтыланған шешімді есептейді

$$x_{k+1} = x_k + \alpha_k s_k \quad (6)$$

α_k қадам ұзындығы s_k іздеу бағытында x_{k+1} мәні $f(x_{k+1})$ минимизациялайтын етіп таңдалады. Яғни, x_{k+1} (5) теңдеуді қанағаттандыруы керек:

$$A(x_k + \alpha_k s_k) = b$$

Теңдеуінің қалдығын енгізіп,

$$r_k = b - Ax_k \quad (7)$$

$\alpha_k A s_k = r_k$ аламыз. Екі жағын s_k^T-ға алдын ала көбейтіп, α_k үшін шешсек, біз келесіні аламыз

$$\alpha_k = \frac{s_k^T r_k}{s_k^T A s_k} \quad (8)$$

Бізде әлі де s_k іздеу бағытын анықтау мәселесі қалды. Интуиция бізге $s_k = -\nabla f = r_k$ таңдау керектігін айтады, өйткені бұл $f(x)$ ішіндегі ең үлкен теріс өзгерістің бағыты. Осы процедура ең жылдам түсу әдісі ретінде белгілі. Бұл танымал алгоритм емес, өйткені оның жинақталуы баяу болуы мүмкін. Неғұрлым тиімді түйіндес градиент әдісі іздеу бағытын келесідей пайдаланады

$$s_{k+1} = r_{k+1} + \beta_k s_k \quad (9)$$

β_k тұрақтысы екі тізбектес іздеу бағыты бір-бірімен түйіндес етіп таңдалады, яғни

$$s_{k+1}^T A s_k = 0$$

Түйіндес градиенттердің үлкен тартымдылығы мынада: бір түйіндес бағытта минимизациялау бұрынғы минимизацияларды жоймайды (минимизациялар бір-біріне кедергі жасамайды).

Соңғы теңдеудегі s_{k+1}орнына (9) қойылады. Осыдан аламыз

$$(r_k^T + \beta_k s_k^T) A s_k = 0$$

ол береді

$$\beta_k = -\frac{r_{k+1}^T A s_k}{s_k^T A s_k}$$

Мұнда түйіндес градиент алгоритмінің құрылымы берілген:

x_0 векторын таңдаңыз(кез келген вектор).
$r_0 = b - Ax_0$ болсын.
$s_0 \leftarrow r_0$ болсын (алдыңғы іздеу бағыты жоқ, ең тік түсу бағытын таңдаңыз).
$k = 0, 1, 2, \ldots$ көмегімен орындаңыз:

$$\alpha_k \leftarrow \frac{s_k^T r_k}{s_k^T A s_k}$$

$x_{k+1} \leftarrow x_k + \alpha_k s_k$
$r_{k+1} \leftarrow b - Ax_{k+1}$
егер$|r_{k+1}| < \varepsilon$ шығу циклі (ε – қате ауытқуы).

$$\beta_k \leftarrow -\frac{r_{k+1}^T A s_k}{s_k^T A s_k}$$

$s_{k+1} \leftarrow r_{k+1} + \beta_k s_k$

Алгоритм бойынша алынған $r_1, r_2, r_3, \ldots$ қалдық векторлары өзара ортогональ болатынын көрсетуге болады, яғни $r_i \cdot r_j = 0, \ i \neq j$. Енді n қалдық векторлардың барлық жиынын есептеу үшін жеткілікті итерацияларды орындадық делік. Келесі итерация нәтижесінде пайда болатын қалдық шешімнің алынғанын көрсететін нөлдік вектор ($r_{n+1} = 0$) болуы керек. Осылайша, түйіндес градиент алгоритмі итерациялық әдіс емес сияқты көрінеді, өйткені ол n есептеу циклінен кейін нақты шешімге жетеді. Алайда іс жүзінде түйіндес әдіс n-нен аз итерацияда қол жеткізіледі.

Түйіндес градиент әдісі кішігірім теңдеулер жиынын шешуде тура әдістермен бәсекеге қабілетті емес. Оның күші үлкен, сирек жүйелермен жұмыс істеуде (A элементтерінің көпшілігі нөлге тең) көрінеді. Айта кету керек, A алгоритмге тек оны векторға көбейту арқылы кіреді; яғни Av түрінде, мұндағы v – вектор (x_{k+1}немесе s_k). Егер A сирек болса, көбейту үшін тиімді ішкі бағдарламаны жазуға болады және А-ның өзіне емес, оның көбейтіндісін түйіндес градиент алгоритміне беруге болады.