LECTURE 7

COORDINATE DESCENT METHODS

- the basic idea is to split the vector x of optimization variables in blocks $x_i \in \mathbb{R}^{n_i}$ and solve the optimization problem for each block separately
- Advantages: cheaper computation and possibility to parallelize it
- Fixed points of the algorithms are not always optimal solutions of the original problem. For convergence guarantees, the required assumptions depend on the algorithm and might be stronger than those for gradient methods
- They can be interpreted from the point of view of conjugate gradient methods and methods for solving linear systems
- Connections with popular methods used nowadays for large scale problems (e.g. SGD)

ITERATIVE METHODS FOR SYSTEMS OF LINEAR EQUATIONS

$$Ax = b$$

$A \in \mathbb{R}^{n \times n}$, invertible
$a_{ii} \neq 0, \forall i$
$b \in \mathbb{R}^n$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad (i\text{-th row})$$

* A is strictly diagonally dominant if

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \forall i$$

Gauss-Seidel

- start with an estimate x^0
- each component is updated as

$$x_i^{k+1} = -\frac{1}{a_{ii}} \left[\sum_{j<i} a_{ij} x_j^{k+1} + \sum_{j>i} a_{ij} x_j^{k} - b_i \right]$$

- serial computation (to compute x_i^{k+1} we require all the previous components of x at iteration $k+1$, i.e. $x_1^{k+1}, \dots, x_{i-1}^{k+1}$)

Here is a little confused, why iff?

* • Convergence is guaranteed iff A is strictly diagonally dominant or $A \in S_{++}^n$.

- In matrix form, the iteration is:

$$x^{k+1} = -D^{-1} \left[L x^{k+1} + U x^k - b \right]$$

D

$$A = \begin{bmatrix} & U \\ L & \end{bmatrix}$$

D, L, U are diagonal, strictly lower and upper parts of A, respectively.

Jacobi • start with an estimate x^0

- the component update is

$$x_i^{k+1} = -\frac{1}{a_{ii}}\left[\sum a_{ij} x_j^k - b_j\right]$$

- parallel computation (each equation i of the system is treated separately and only variables at the previous iterations are required)
- In matrix form, the iteration is:

$$x^{k+1} = -D^{-1}(L+U)x^k + D^{-1}b$$

- Convergence here is guaranteed for strictly diagonally dominant (sufficient condition), but not for every positive definite A

Both algorithms can be written as

$$\boxed{x^{k+1} = Mx^k + Gb}$$

where $M = \begin{cases} -(I+D^{-1}L)^{-1}D^{-1}U & \text{Gauss Seidel} \\ -D^{-1}(A-D) & \text{Jacobi} \end{cases}$

- Convergence of the algorithm depends on M (note that the part depending on b has no effect on convergence properties! It can be set to 0 when studying convergence)
- Specifically, the algorithms converge iff $\rho(M) < 1$

$\rho(M)$: spectral radius

You can interpret this from the point of view of a discrete time linear system:

$x^t \to x^*$ for $t \to \infty$

iff the state matrix of the system (which here is M) has all the eigenvalues inside the unit circle.

(this holds regardless of b, which only influences x^*)

- Moreover, the rate of convergence is governed by $\rho(M)$. The higher it is, the slower the convergence

how to actually compute the spectral radius?

The COORDINATE MINIMIZATION method generalizes the Gauss-Seidel algorithm for solving linear systems (which we know is equivalent to min $f(x)$ where $f(x) = \frac{1}{2} x^T A x - bx$, when $A \in S^n_{++}$) to a generic $f(x)$.

The PARALLEL COORDINATE MINIMIZATION is instead closely related to the Jacobi algorithm.

- Convergence guarantees and rates depend on continuity and smoothness of f.
- regularization terms can be added to f to improve convergence properties (for the parallel version, the algorithm may not converge without a regularization term)

COORDINATE GRADIENT DESCENT

gradient descent: $x^{k+1} = x^k - t^k \nabla f(x^k)$

here we take for every block-coordinate i:

$$x^{k+1} = x^k - t_i^k [\nabla f(x^k)]_i \, e_i$$

where

- if $n_i = 1$ $\forall i$ (1-D coordinate), then

$e_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ ← i-th component $[\nabla f(x^k)]_i$ is the i-th component of the gradient

- if $n_i > 1$ (block-coordinates), then the corresponding components of x will be updated.

Example: $f(x) = \frac{1}{2} x^T A x - bx$, $x \in \mathbb{R}^6$

$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$ $x_1 \to i=1$, $x_2, x_3 \to i=2$, $x_4, x_5, x_6 \to i=3$

We will have 3 steps:

$i=1$ $\quad x^{k+1} = x^k - t_1^k \begin{bmatrix} a_1 x^k - b_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

$i=2$ $\quad x^{k+1} = x^k - t_2^k \begin{bmatrix} 0 \\ a_2 x^k - b_2 \\ a_3 x^k - b_3 \\ a_4 x^k - b_4 \\ 0 \\ 0 \end{bmatrix}$

$i=3$ $\quad x^{k+1} = x^k - t_3^k \begin{bmatrix} 0 \\ \vdots \\ a_5 x^k - b_5 \\ a_6 x^k - b_6 \end{bmatrix}$

a_i = i-th row of A

L-smoothness: is an upper bound of the convex property

- t_i^k can be chosen based on L_i-smoothness of the i-th block. In this case $t_i^k = \frac{1}{L_i}$.

All the criteria studied in gradient methods are also valid choices in this setting.

- The coordinates can be selected:

☑ in a "cyclic fashion": $i_0 = 1$,

$$i_{k+1} = \mathrm{mod}(k, n) + 1$$

☑ in an "essentially cyclic fashion": for some $T \geq n$, each coordinate is modified at least once in every stretch of T iterations

union $\bigcup_{j=0}^{T} \{i_{k-j}\} = \{1, 2, \ldots, n\} \quad \forall k \geq T$

☑ at random (not necessarily with equal probability)

Application to a regularized problem

$$\min_x \; h(x)$$

$$h(x) = f(x) + \lambda \Omega(x)$$

regularization parameter ($\lambda > 0$)

regularization function (assumed separable)

$$\Omega(x) = \sum_{i=1}^{n} \Omega_i(x)$$

examples:
- $\Omega(x) = \|x\|_1$, $\Omega_i(x_i) = |x_i|$
- box constraints, $\Omega_i = I_{[l_i, u_i]}(x_i)$

The update role has 2 steps (assume for simplicity $n_i = 1$)

(1) $$z_i^k = \operatorname*{argmin}_y \; (y - x_i^k)\left[\nabla f(x^k)\right]_i + \frac{1}{2\alpha_k}\|y - x_i^k\|_2^2 + \lambda \Omega_i(y)$$

$\frac{1}{2\alpha_k}$ plays the role of stepsize

This is a scalar subproblem aimed at computing a linear approximation of f along the i-th direction while explicitely accounting for Ω_i.

Notably, this problem has a closed form solution:

$$z_i^k = S_{\lambda\alpha_k}\left(x_i^k - \alpha_k \left[\nabla f(x^k)_i\right]\right)$$

where $$S_\beta(z) = \min_y \; \frac{1}{2\beta}\|y - z\|_2^2 + \Omega_i(y)$$

so no explicit search is needed.

(2) $$x^{k+1} = x^k + (z_i^k - x_i^k)\, e_i$$

Note: if $\Omega_i = 0$, the algorithm recovers the standard coordinate gradient descent.

STOCHASTIC GRADIENT DESCENT (SGD)

Where is the connection with coordinate methods?

idea of SGD algorithms

$$x^{k+1} = x^k - t^k \tilde{\nabla} f(x^k)$$

realization of a stochastic gradient approximator

Consider for example:

This looks like a coordinate gradient descent direction

$$\tilde{\nabla} f(x^k) = n \left[\nabla f(x^k)\right]_i e_i$$

Do we always have this n? use this n to make it unbiased? actually just take one sample instead of normal

where $p_i = p(I = i) = \frac{1}{n}$

Then:

UNBIASED ESTIMATE

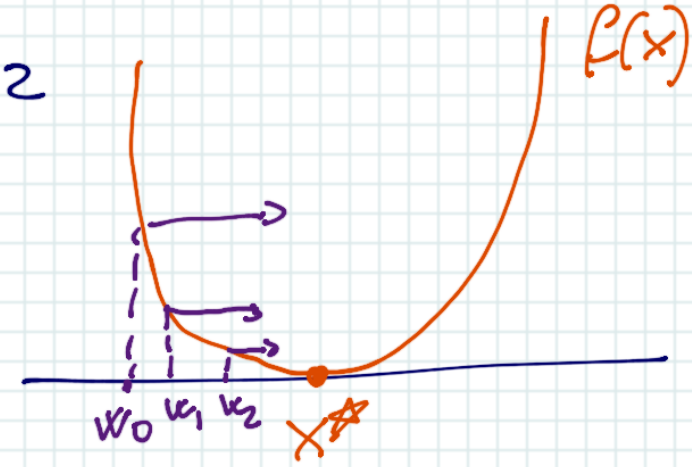
$$E\left[\tilde{\nabla} f(x^k)\right] = \frac{1}{n} \sum_{i=1}^{n} n \left[\nabla f(x^k)\right]_i e_i = \nabla f(x^k)$$

Randomized coordinate gradient descent can thus be seen as a special case of SGD methods.

However, the former has the advantage over general SGD approaches that descent in f can be guaranteed at every iteration.

SGD in action

$$f(x) = \frac{1}{4}\sum_{i=1}^{4}\left(a_i^T x - b_i\right)^2$$

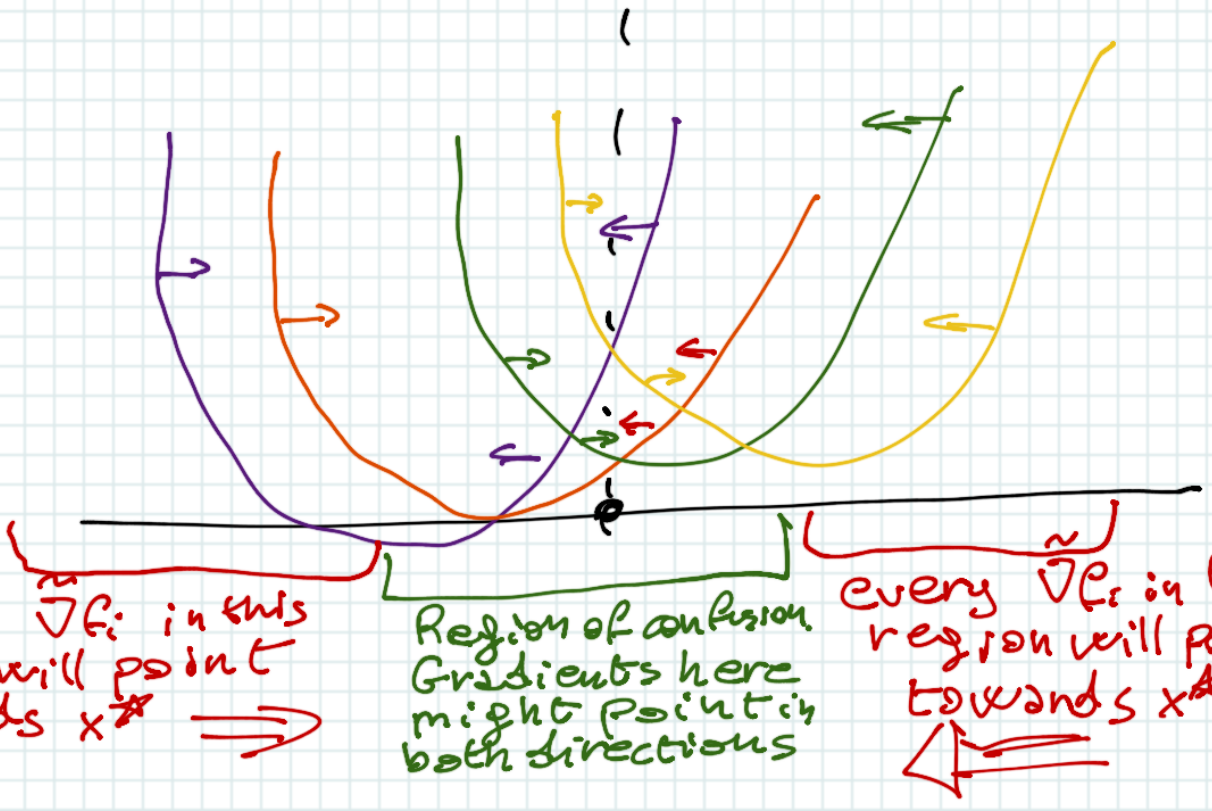


Gradient descent

SGD

$$f_1(x) = (a_1^T x - b_1)^2$$
$$f_2(x) = (a_2^T x - b_2)^2$$
$$f_3(x) = (a_3^T x - b_3)^2$$
$$f_4(x) = (a_4^T x - b_4)^2$$

x^* SGD will minimize the average value



The "confusion" is captured by the variance of the gradients:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} \| \nabla f_i(x) - \nabla f(x) \|^2$$

If $\sigma^2 = 0$ → every step goes towards x^*

If $\sigma \ll 1$ → narrow "region of confusion", most steps go in the right direction

If $\sigma \gg 1$ → the algorithm operates in a region where many steps are made away from x^*

Strategies to address variance:

- adapt step-size. Specifically, we need a decreasing step size which is fast enough initially to get in the region of x^* and then slows down to be robust to variance
- minibatch → average $K \ll n$ gradients when estimating $\tilde{\nabla} f(x^k)$. Variance is inversely proportional to the batch size.