Coordinate descent

Alexandre Gramfort

alexandre.gramfort@inria.fr

Master 2 Data Science, Univ. Paris Saclay
Optimisation for Data Science
course based on notes from Olivier Fercoq.

Table of Contents

1. Exact coordinate descent
2. Coordinate gradient descent
3. Proximal coordinate descent
4. Applications to ML estimators

Why coordinate descent for datascience?

So far you have seen first order method:

- gradient descent
- proximal gradient descent
- accelerated gradient descent

You'll also see with me

- Newton methods
- quasi-Newton methods

Coordinate descent (CD) has received a lot of attention in ML/stats over the last 10 years. It's state-of-the-art techniques on a number of learning problems, as CD applies in this settings (not as general as gradient descent). It's what R GLMNET package and Scikit-Learn Lasso / Elastic-Net / LinearSVC estimators use.

Coordinate wise optimization

We work in finite dimension $\mathbb{R}^n$ (think n parameters to optimize)

Coordinate descent is **extremely simple**

Idea: minimize one coordinate at a time (keeping the other fixed)

Question: Given convex, differentiable $f : \mathbb{R}^n \to \mathbb{R}$, if we are at a point x such that $f(x)$ is minimized along each coordinate axis, have we found a global minimizer?

i.e., does $f(x + dU_i) \geq f(x) \; \forall d \in \mathbb{R}, \; \forall i \Rightarrow f(x) = \min_z f(z)$?

where $U_i = (0, \dots, 1, \dots, n) \in \mathbb{R}^n$ is the ith canonical basis vector.

Coordinate wise optimization

$f(x + dU_i) \geq f(x), \forall d \in \mathbb{R}$ implies that

d >0 or d<0 or d=0 $\quad \dfrac{\partial f}{\partial x^{(i)}}(x) = 0$

which implies

$$\nabla f(x) = \left(\frac{\partial f}{\partial x^{(1)}}(x), \ldots, \frac{\partial f}{\partial x^{(n)}}(x) \right) = 0$$

OK for f smooth and convex !

Table of Contents

1. Exact coordinate descent
2. Coordinate gradient descent
3. Proximal coordinate descent
4. Applications to ML estimators

Exact coordinate descent

Objective: $\min_{x \in \mathbb{R}^n} f(x)$

Initialisation: $x_0 = (x_0^{(1)}, \ldots, x_0^{(n)})$.

Algorithm:

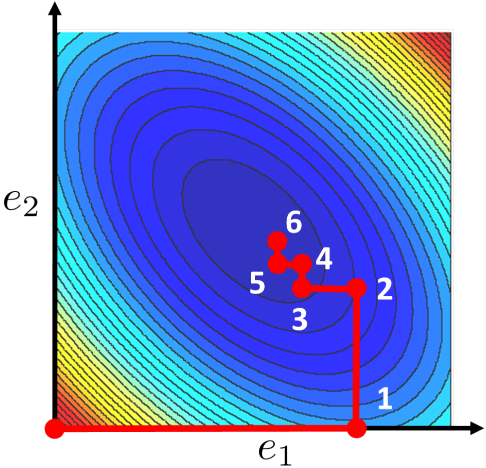
$$\text{Choose } l = (k \mod n) + 1 \quad \text{(cyclic rule)}$$

$$\begin{cases} x_{k+1}^{(i)} = \arg\min_{z \in \mathbb{R}} f(x_k^{(1)}, \ldots, x_k^{(l-1)}, z, x_k^{(l+1)}, \ldots, x_k^{(n)}) & \text{if } i = l \\ x_{k+1}^{(i)} = x_k^{(i)} & \text{if } i \neq l \end{cases}$$

Note: The order of cycle through coordinates is arbitrary, can use any permutation of $1, 2, \ldots, n$.

Note: We just have to solve 1D optimization problems but a lot of them...

Example



Coordinate descent on a 2D problem

Example: Linear regression

Let $f(x) = \frac{1}{2}\|y - Ax\|^2$, where $y \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ is the design matrix with columns $A_1, \ldots, A_n$ (one per feature)

Consider minimizing over $x^{(i)}$, with all $x^{(j)}$, $j \neq i$ fixed:

$$0 = \nabla_i f(x) = A_i^\top (Ax - y) = A_i^\top (A_i x^{(i)} + A_{-i} x^{(-i)} - y)$$

i.e., we take:

$$x^{(i)} = \frac{A_i^\top (y - A_{-i} x^{(-i)})}{A_i^\top A_i}$$

Repeat these update by cycling over coordinates

$\rightarrow$ notebook

Example: Linear regression

Note that doing:

$$x^{(i)} = \frac{A_i^\top (y - A_{-i}x^{(-i)})}{A_i^\top A_i}$$

is equivalent to:

$$x^{(i)} \leftarrow x^{(i)} + \frac{A_i^\top r}{A_i^\top A_i}$$

where $r = y - Ax$ is the current *residual*. If current r is available the cost of an update is $O(m)$. Updating r is also $O(m)$ so full pass/epoch on coordinates is $O(mn)$ as for gradient descent.

Convergence of exact coordinate descent

Proposition (Warga (1963))

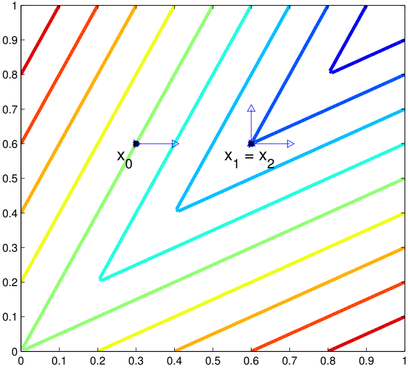
Assume that

- f *is continuously differentiable*
- f *is strictly convex*
- *there exists* $x_* \in \arg\min_{x \in \mathcal{X}} f(x)$

then the exact coordinate descent method converges to x_*.

Counter-example: convex nonsmooth

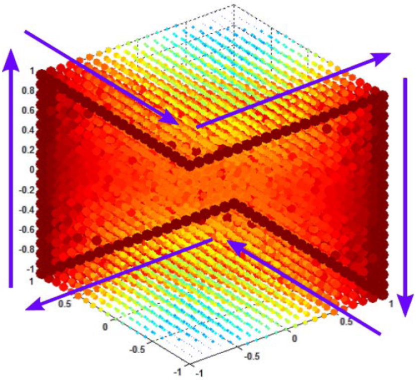
What if f is convex and non-smooth?



$$f(x^{(1)}, x^{(2)}) = |x^{(1)} - x^{(2)}| - \min(x^{(1)}, x^{(2)})$$

Counter-example: smooth nonconvex

What is f is smooth and non-convex? (Example due to Powell)



$$f(x^{(1)}, x^{(2)}, x^{(3)}) = \\ -(x^{(1)}x^{(2)} + x^{(2)}x^{(3)} + x^{(3)}x^{(1)}) + \sum_{i=1}^{3} \max(0, |x^{(i)}| - 1)^2$$

Adaboost

y_j = label
h_j = weak classifier
Minimise the exponential loss:

$$f(x) = \sum_{j=1}^{m} \exp(-y_j h_j^\top x).$$

Algorithm:

- Select the variable i_{k+1} such that $i_{k+1} = \arg\max_i |\nabla_i f(x_k)|$ (greedy rule a.k.a. Gauss-Southwell rule, requires to compute the full gradient at each iteration)
- Perform exact coordinate descent along coordinate i_{k+1}

If $y_i \in \{-1, 1\}$ and $h_j \in \{-1, 0, 1\}^n$: closed form formulas

Table of Contents

1. Exact coordinate descent
2. Coordinate gradient descent
3. Proximal coordinate descent
4. Applications to ML estimators

Motivation

- A 1D optimisation problem to solve at each iteration: This may be expensive
- We may solve it approximately since we've got plenty of iterations left
- We will do one single gradient step in the 1D problem

Coordinate gradient descent

Parameters: $\gamma_1, \ldots, \gamma_n > 0$

Algorithm:

Choose $i_{k+1} \in \{1, \ldots, n\}$

$$
\begin{cases}
x_{k+1}^{(i)} = x_k^{(i)} - \gamma_i \nabla_i f(x_k) & \text{if } i = i_{k+1} \\
x_{k+1}^{(i)} = x_k^{(i)} & \text{if } i \neq i_{k+1}
\end{cases}
$$

Coordinate gradient descent

Parameters: $\gamma_1, \ldots, \gamma_n > 0$

Algorithm:

Choose $i_{k+1} \in \{1, \ldots, n\}$

$$\begin{cases} x_{k+1}^{(i)} = x_k^{(i)} - \gamma_i \nabla_i f(x_k) & \text{if } i = i_{k+1} \\ x_{k+1}^{(i)} = x_k^{(i)} & \text{if } i \neq i_{k+1} \end{cases}$$

Choice of γ: coordinate-wise Lipschitz constant i.e. Lipschitz constant of

$$g_{i,x} : X_i \to \mathbb{R}$$
$$h \mapsto f(x + U_i h) = f(x^{(1)}, \ldots, x^{(i-1)}, x^{(i)} + h, x^{(i+1)}, \ldots, x^{(n)})$$

We will denote $L_i = L(\nabla g_{i,x})$ this Lipschitz constant.

Convergence speed

Assume f is convex; ∇f is Lipschitz continuous; $\forall i$, $\gamma_i = \frac{1}{L_i}$.

Proposition (Beck and Tetruashvili (2013))

If $i_{k+1} = (k \mod n) + 1$, *then*

$$f(x_{k+1}) - f(x_*) \leq 4L_{\max}(1 + n^3 L_{\max}^2 / L_{\min}^2) \frac{R^2(x_0)}{k + 8/n}$$

where $R^2(x_0) = \max_{x,y \in \mathcal{X}} \{\|x - y\| : f(y) \leq f(x) \leq f(x_0)\}$, $L_{\max} = \max_i L_i$ *and* $L_{\min} = \min_i L_i$.

Note: n^3 can be prohibitive in high dimension. Due to pathological cases of the cyclic rule this bound is very very pessimistic (cf. linear regression).

Convergence speed with randomization

Assume f is convex; ∇f is Lipschitz continuous; $\forall i$, $\gamma_i = \frac{1}{L_i}$.

Proposition (Nesterov (2012))

If i_{k+1} is randomly generated, independently of $i_1, \ldots, i_k$ and $\forall i \in \{1, \ldots, n\}$, $\mathbb{P}(i_{k+1} = i) = \frac{1}{n}$, then

$$\mathbb{E}[f(x_{k+1}) - f(x_*)] \leq \frac{n}{k+n}\Big((1 - \frac{1}{n})(f(x_0) - f(x_*)) + \frac{1}{2}\|x_* - x_0\|_L^2\Big)$$

where $\|x\|_L^2 = \sum_{i=1}^n L_i \|x^{(i)}\|_2^2$.

Note: As the algorithm is now stochastic the bounds are given in expectation.

Comparison with gradient descent

The iteration complexity of the gradient descent method is

$$f(x_{k+1}) - f(x_*) \leq \frac{L(\nabla f)}{2(k+1)} \|x_* - x_0\|_2^2$$

To get an ϵ-solution (*i.e.*, such that $f(x_k) - f(x_*) \leq \epsilon$), we need at most $\frac{L(\nabla f)}{2\epsilon} \|x_* - x_0\|_2^2$ iterations.

while for coordinate descent we need (omitting randomization)

$$\frac{n}{\epsilon}\Big((1 - \frac{1}{n})(f(x_0) - f(x_*)) + \frac{1}{2}\|x_* - x_0\|_L^2\Big)$$

iterations.

Comparison with gradient descent

How do the cost of iterations compare?

Let C the cost of one GD iteration and c the cost of one CD iteration.

Back to least square: C is the cost of computing $\nabla f(x) = A^\top(Ax - b)$ which means $C = O(\mathrm{nnz}(A))$ or $C = O(mn)$ for a dense matrix.

We have for CD, $\nabla_i f(x) = U_i^\top A^\top(Ax - b)$ and with smart residual updates $c = O(\mathrm{nnz}(A))/n$ or $C = O(m)$ for a dense matrix. So

$$c \approx C/n$$

Comparison with gradient descent

Let's recall number of iterations for CD:

$$\frac{n}{\epsilon}\Big((1-\frac{1}{n})(f(x_0)-f(x_*))+\frac{1}{2}\|x_*-x_0\|_L^2\Big)$$

- $f(x_0)-f(x_*)\leq\frac{L(\nabla f)}{2}\|x_0-x_*\|_2^2$ and it may happen that $f(x_0)-f(x_*)\ll\frac{L(\nabla f)}{2}\|x_0-x_*\|_2^2$
- $L(\nabla f)=\lambda_{\max}(A^\top A)$ and $L_i=a_i^\top a_i$ with $a_i=AU_i$. We always have $L_i\leq L(\nabla f)$ and it may happen that $L_i=O(L(\nabla f)/n)$.
- So in the quadratic case, $C_{CD}\leq C_{GD}$ and we may have $C_{CD}=O(C_{GD}/n)$.
- Explains the results in the notebook...

Table of Contents

1 Exact coordinate descent

2 Coordinate gradient descent

3 Proximal coordinate descent

4 Applications to ML estimators

CD for composite separable problem?

Let us consider:

$$F(x) = f(x) + \sum_{i=1}^{n} g_i(x^{(i)}) \enspace ,$$

with

- f convex, differentiable
- each g_i convex

The non-smooth part is here separable.
Question: Does

$$F(x + dU_i) \geq F(x) \; \forall d \in \mathbb{R}, \; \forall i \overset{?}{\Rightarrow} F(x) = \min_z F(z)$$

CD for composite separable problem?

$$F(y) - F(x) \geq \nabla f(x)^\top (y - x) + \sum_{i=1}^{n} (g_i(y^{(i)}) - g_i(x^{(i)}))$$

$$\text{should be} = \quad \geq \sum_{i=1}^{n} \underbrace{\left[\nabla_i f(x)(y^{(i)} - x^{(i)}) + (g_i(y^{(i)}) - g_i(x^{(i)}))\right]}_{\geq 0}$$

$$\geq 0$$

This suggests that it should work ...

Proximal coordinate descent

Parameters: $\gamma_1, \ldots, \gamma_n > 0$

Algorithm:

Choose $i_{k+1} \in \{1, \ldots, n\}$

$$\begin{cases} x_{k+1}^{(i)} = \mathrm{prox}_{\gamma_i, g_i}\left(x_k^{(i)} - \gamma_i \nabla_i f(x_k)\right) & \text{if } i = i_{k+1} \\ x_{k+1}^{(i)} = x_k^{(i)} & \text{if } i \neq i_{k+1} \end{cases}$$

$$\mathrm{prox}_{\gamma, g}(y) = \arg\min_{x \in \mathbb{R}^n} g(x) + \frac{1}{2}\|x - y\|^2_{\gamma^{-1}}$$
$$\mathrm{prox}_{\gamma_i, g_i}(y) = \arg\min_{x \in \mathbb{R}} g_i(x) + \frac{1}{2\gamma_i}(x - y)^2$$

$\rightarrow$ proximal operators for $g(x) = \lambda|x|$, $g(x) = \lambda\|x\|_2^2$ and $g(x) = \mathbb{I}_{[0,1]}(x)$.

Convergence speed

We want to minimize $F = f + g$.
Assume f and g are convex; ∇f is Lipschitz continuous;
$\forall i,\ \gamma_i = \frac{1}{L_i}$.

Proposition (Richtárik and Takáč (2014))

If i_{k+1} is randomly generated, independently of $i_1, \dots, i_k$ and $\forall i \in \{1, \dots, n\}$, $\mathbb{P}(i_{k+1} = i) = \frac{1}{n}$, then

$$\mathbb{E}[F(x_{k+1}) - F(x_*)] \leq \frac{n}{k+n}\Big((1-\frac{1}{n})(F(x_0) - F(x_*)) + \frac{1}{2}\|x_* - x_0\|_L^2\Big)$$

Note: One obtain the same rate as for non-composite objectives.
$\rightarrow$ cf. Proof in lecture notes.

Table of Contents

1. Exact coordinate descent
2. Coordinate gradient descent
3. Proximal coordinate descent
4. Applications to ML estimators

Regression and classification under sparsity constraints

$$\min_{x \in \mathbb{R}^n} F(x) = \min_{x \in \mathbb{R}^n} f(x) + \sum_{i=1}^{n} g_i(x^{(i)})$$

- Lasso: $f(x) = \frac{1}{2}\|y - Ax\|^2$ and $g(x) = \|x\|_1 = \sum_i |x^{(i)}|$
- ℓ_1 log. reg.: $f(x) = \log(\exp(-y \odot Ax) + 1)$ and $g(x) = \|x\|_1$ where $\odot$ is the elementwise product (Hadamard product).
- Box-constrained regression $f(x) = \frac{1}{2}\|y - Ax\|^2$ s.t. $\|x\|_\infty \leq \kappa$
- Non-negative least squares (NNLS) $f(x) = \frac{1}{2}\|y - Ax\|^2$ s.t. $x^{(i)} \geq 0$

Note: Generally the regularizer is separable non-smooth and the data fit is smooth.

$\rightarrow$ write full algorithm for NNLS and Lasso

Multi-output regression under sparsity constraints

Multi-task Lasso (k tasks):

$$\min_{x \in \mathbb{R}^{n \times k}} F(x) = \min_{x \in \mathbb{R}^{n \times k}} \frac{1}{2} \|Y - Ax\|_{Fro}^2 + \sum_{i=1}^{n} \|x^{(i,\cdot)}\|_2$$

where $x^{(i,\cdot)}$ is the ith row of matrix x.

Note: Here the g is still separable yet blocks of coordinates are updated at each iteration (it's block proximal coordinate descent). First convergence proof due to Tseng (2001).

Support vector machines

Coordinate descent can be applied to the SVM in the dual. If the primal with ($y_i \in \{-1, 1\}$) reads:

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} C \sum_{i=1}^{n} \max(0, 1 - y_i(z_i^\top w + b)) + \frac{1}{2}\|w\|_2^2$$

the classical dual of SVM for binary classification is given by:

$$\max_{\alpha \in \mathbb{R}^n} -\frac{1}{2}\alpha^T Q \alpha + \mathbf{1}^\top \alpha \text{ s.t. } y^T \alpha = 0 \text{ and } 0 \leq \alpha \leq C\mathbf{1}$$

with $Q_{ij} = y_i y_j z_i^\top z_j$.
Note: Here w is the normal to the separating hyperplane and b is the intercept.
$\rightarrow$ Derive the dual from the primal writing the Lagrangian and KKT optimality conditions.

Support vector machines with SMO

The dual reads:

$$\max_{\alpha \mathbb{R}^n} -\frac{1}{2}\alpha^T Q\alpha + \mathbf{1}^\top \alpha \text{ s.t. } y^T\alpha = 0 \text{ and } 0 \leq \alpha \leq C\mathbf{1}$$

Sequential minimal optimization or SMO (Platt, 1998) is a blockwise coordinate descent in blocks of 2. Instead of cycling, it chooses the next block greedily.

Note: This does not meet separability assumptions for convergence we have just seen.

Note: This is what is implemented in Scikit-Learn SVC and SVR estimators that use internally the libsvm C++ library.

Support vector machines with SDCA

If one does not fit an intercept b the primal reads:

$$\min_{w \in \mathbb{R}^p} C \sum_{i=1}^{n} \max(0, 1 - y_i z_i^\top w) + \frac{1}{2}\|w\|_2^2$$

and a dual formulation becomes:

$$\max_{\alpha \in \mathbb{R}^n} -\frac{1}{2}\alpha^T Q \alpha + \mathbf{1}^\top \alpha - \mathbb{I}_{[0,C]^n}(\alpha).$$

Proximal coordinate ascent applies to this problem. When using the stochastic approach this algorithm is called Stochastic Dual Coordinate Ascent (SDCA).

Note: This is what is implemented in Scikit-Learn LinearSVC when using parameter *dual=True*. It uses internally the liblinear C++ library.

$\rightarrow$ Write an implementation of SDCA.

Support vector machines with SDCA

Proposition (Shalev-Shwartz and Zhang (2013))

Let us define a primal point $w_k = Z^\top \mathit{Diag}(y)\, \alpha_k$, *where* $(\alpha_k)_{k \geq 0}$ *is generated by SDCA. The duality gap satisfies for all* $K \geq n$,

$$\mathbb{E}\Big[\frac{1}{K} \sum_{k=K}^{2K-1} P(w_k) - D(\alpha_k)\Big] \leq$$

$$\frac{n}{K+n}\Big((1 - \frac{1}{n})(D(\alpha_*) - D(\alpha_0)) + \frac{1}{2}\|\alpha_* - \alpha_0\|_L^2\Big) + \frac{n}{2K} C^2 \sum_{i=1}^{n} L_i$$

where $\forall i, L_i = y_i^2 \|z_i\|^2$.

$\rightarrow$ cf. Proof in lecture notes.

Graphical Lasso

Let $A \in \mathbb{R}^{n \times p}$, where rows are independent Gaussian observations drawn from $N(0, \Sigma)$,
The graphical Lasso estimator (Banerjee et al., 2007, Friedman et al., 2007) reads:

$$\min_{\Theta \in \mathbb{R}^{p \times p}} - \log \det \Theta + \operatorname{tr} S\Theta + \lambda \|\Theta\|_1$$

where $\|\Theta\|_1 = \sum_{ij} |\Theta_{ij}|$.

It provides an estimate of Σ^{-1} (precision matrix) when $S = A^\top A / n$ is the empirical covariance.

Graphical Lasso

Stationarity conditions:

$$-\Theta^{-1} + S + \lambda\Gamma = 0$$

where $\Gamma_{ij} \in \partial|\Theta_{ij}|$. Posing $W = \Theta^{-1}$. It is possible to do a coordinate descent on W. See Friedman et al. (2007).

Note: With $\lambda = 0$ one recovers the maximum likelihood estimator.
Note: This is implemented the *GraphLasso* estimator in Scikit-Learn or in the glasso package in R.