Read BV Chapter 10, 11

1 Equality constrained minimization

See BV Ch 10.

2 Interior point methods

See BV Ch 11.

3 Projected direction (projected gradient) method

Assume for simplicity that there are no equality constraints. Denote by \( A \) the feasible set \( A = \{ x \mid f_1(x) \leq 0 \} \). The optimization problem is \( \min_{x \in A} f_0(x) \).

Idea: start from a feasible point. Pretend the problem is unconstrained and take a step in a descent direction. If this step ends outside \( A \), stop on the boundary of \( A \), i.e. “project” the obtained point back onto \( A \).

Algorithm \texttt{ProjectedDirection}

\textbf{Input} tolerance \( tol \)

\textbf{Initialize} \( x^0 \in A, k = 0 \)

while \( ||\text{Proj}_A(x^k - \nabla f_0(x^k)) - x^k|| > tol \)
1. find direction of descent $d^k$, step $\eta^k$
2. $\tilde{x} = x^k - \eta^k d^k$ (take an unconstrained step)
3. $x^{k+1} = \text{Proj}_A(\tilde{x})$ (project back onto $A$)

The **projection operator** $\text{Proj}_A(x)$ is defined as

$$\text{Proj}_A(x) = \text{argmin}_{u \in A} ||x - u||$$  \hspace{1cm} (1)

where $x \in \mathbb{R}^n$ is a point and $A \subset \mathbb{R}^n$ is a set.

Intuition for the stopping criterion. Note that $-\nabla f_0(x^k)$ is the direction of maximum descent of $f_0$ in the absence of any constraints. There are two cases:

(i) $x^k - \nabla f_0(x^k) \notin A$, so stepping in the direction of maximum descent takes us outside $A$. Then, if projecting back falls exactly (or near) $x^k$, it means that the constraints are stopping us from further minimizing $f_0$, so we are at the constrained minimum.

(ii) $x^k - \nabla f_0(x^k) \in A$. Then, $||\text{Proj}_A(x^k - \nabla f_0(x^k)) - x^k|| = 0$ implies $\nabla f_0(x^k) = 0$ and again we are at the optimum.

This algorithm is efficient when $\text{Proj}_A(x)$ can be computed efficiently for given set of constraints.

### 3.1 Usual projection operators

**Projection of the $||.||_2$ ball**

$$z = \text{argmin}_{||z|| \leq r} ||x - z|| = \begin{cases} x, & ||x|| \leq r \\ \frac{x}{||x||}, & ||x|| > r \end{cases} \hspace{1cm} (2)$$

**Projection on the $||.||_1$ ball**

See Homework 5

**Projection on the first orthant**

$$z = \text{argmin}_{z \geq 0} ||x - z|| = \text{argmin}_{z \geq 0} ||x - z||^2 = \text{argmin}_{z \geq 0} \sum_{i=1}^{n} (x_i - z_i)^2 = (x)_+ = [\max(x_i,0)]_{i=1}^{n} \hspace{1cm} (3)$$

2
Projection on $a^\perp$ If we have a single equality constraint of the form $a^T x = 0$, with $a, x \in \mathbb{R}^n$ then the projection on $\mathcal{A} = \{ x | a^T x = 0 \}$ is obtained as follows: (i) $\frac{a}{||a||}$ is a vector of length 1 in the direction of $a$, (ii) $\frac{a^T x}{||a||}$ is the length of the projection of $x$ on $a$, (iii) we now substract from $x$ its projection on $a$ and what is left is the projection of $a^\perp$:

$$\text{Proj}_{a^\perp} x = x - \frac{a^T x}{||a||} \frac{a}{||a||} = \left( I - \frac{aa^T}{||a||^2} \right) x$$

(4)

Example. Solving the C-SVM by Projected Gradient Dual problem

$$\min_{\lambda} \frac{1}{2} \lambda^T \tilde{K} \lambda - \frac{1}{2}^T \lambda, \text{ s.t. } y^T \lambda = 0, \ 0 \prec \lambda \prec C.$$ 

Algorithm:

1. Gradient $\nabla f_0(\lambda^k) = \tilde{K} \lambda^k - 1 = g^k$ is unconstrained direction of descent
2. Project on affine constraint $d^k = \left( I - \frac{aa^T}{||a||^2} \right) g^k$ to obtain direction of descent.
3. Line search, get step size $\eta^k$, take step $\tilde{\lambda} = \lambda^k - \eta^k d^k$
4. Project on box constraints

$$\lambda^k_i = \begin{cases} 
\tilde{\lambda}_i, & \text{if } \tilde{\lambda}_i \in [0, C] \\
0, & \text{if } \tilde{\lambda}_i < 0 \\
C, & \text{if } \tilde{\lambda}_i > C 
\end{cases}$$

(5)

4 Proximal gradient methods and the Lasso

These methods are not explicitly solving constrained optimization methods; however, many inequality constrained problems can be recast as regularized unconstrained problems.

4.1 A simple $l_1$ regularized problem

$$\frac{1}{2} \arg\min_u ||x - u||^2 + \lambda ||u||_1, \lambda > 0.$$
This problem is unconstrained, but not smooth. At the optimum, we know that 0 is a subgradient of the objective, i.e. \( 0 \in \partial(||x - u||^2 + \lambda ||u||_1) = -(x - u) + \lambda \partial ||u||_1 \).

Now assume we know the solution \( u^* \), and let’s look at its components \( u_i^* \) one by one. There are two cases

- \( u_i^* \neq 0 \). In this case \( u_i^* = x_i - \lambda \text{sgn}(u_i^*) = x_i - \lambda \text{sgn}(x_i)(|x_i| - \lambda) \). To motivate the \( \frac{1}{\lambda} \) step above, reason like this: if \( u_i^* > 0 \), then \( u_i^* = x_i - \lambda \) which implies necessarily \( x_i > \lambda > 0 \); by a similar argument, \( u_i^* < 0 \) only if \( x_i < 0 \). Hence \( \text{sgn}(u_i^*) = \text{sgn}(x_i) \) (whenever \( u_i^* \neq 0 \)).

- \( u_i^* = 0 \). Then \( 0 = -x_i + 4\lambda \) with \( t \in [-1,1] \), which implies \( x_i \in [-\lambda, \lambda] \).

Putting it all together, we have that

\[
    u_i^* \equiv \text{soft}(x, \lambda) = \begin{cases} 
        0, & \text{if } x_i \in [-\lambda, \lambda] \\
        \text{sgn}(x_i)(|x_i| - \lambda), & \text{if } |x_i| > \lambda 
    \end{cases}
\]  

(6)

The above mapping is called soft thresholding of \( x \) by \( \lambda \).

More generally, the proximal mapping of a function \( f \) is defined as the operator

\[
    \text{Proxi}_f(x) = \arg\min_u (f(u) + \frac{1}{2}||x - u||^2).
\]  

(7)

The function \( \text{Proxi}_f(x) \) returns a value that aims to minimize \( f \) and to be close to \( x \). The soft-thresholding operation is the proximal mapping of the \( l_1 \) norm (times \( \lambda/2 \)).

The projection on a set \( A \) is the proximal mapping for the indicator function \( I_A \) (defined as \( I_A(x) = 0 \) if \( x \in A \) and \( I_A(x) = \infty \) for \( x \notin A \)).

Another simple problem (whose solution is left as exercise) is

\[
    \min_{\beta \in \mathbb{R}} \sum_i (y^i - x^i\beta)^2 + \lambda |\beta|.
\]  

(8)

This would be the unidimensional version of Proximal problem for the \( l_1 \) norm, were it not for the \( x^i \) variables. The solution is

\[
    \beta_j^* = \text{soft} \left( \hat{\beta}_j, \frac{\lambda}{2\alpha} \right)
\]  

(9)
where $\hat{\beta}_j$ is the usual least squares estimate given by $\hat{\beta}_j = \frac{c}{a}$, $a = \sum_i x_i^2$, $c = -\sum x_i y_i$.

4.2 Coordinate descent for the Lasso

The Lasso regression is defined as

$$\min_{\beta} \sum_{i=1}^N (y_i - \beta_0 - \beta^T x_i)^2 + \lambda ||\beta||_1.$$  \hspace{0.2cm} (10)

Where $D = \{(x^i, y^i), i = 1: N\}$ is the data, $x^i \in \mathbb{R}^n$, $y^i \in \mathbb{R}$, $\beta_0 \in \mathbb{R}$, $\beta = [\beta_1, \ldots, \beta_n]^T$ are the parameters to be estimated, and $\lambda$ is the regularization parameter.

In practice $\lambda$ is not known, and we estimate it by cross-validation. This means that we need the whole regularization path, i.e. the whole set of solutions $\beta^*(\lambda)$. Some algorithms for the Lasso obtain the regularization path implicitly, for example the Least Angle Regression (LARS) algorithm. We will not present the LARS explicitly here, but in section 4.4 we give some intuition on how LARS works. But another approach is to solve the problem (10) for a large enough set of fixed $\lambda$’s. So, for the moment we will assume that $\lambda$ is fixed.

We will also simplify the problem by ignoring $\beta_0$, which is not regularized. Note that the estimate for $\beta_0$ is the sample mean of $(y^i - \beta^T x^i)$.

Simple coordinate descent As in any coordinate descent algorithm, the idea is to optimize one $\beta_j$, keeping the others fixed.

$$\min_{\beta_j} \sum_{i=1}^N [(y_i - \sum_{k \neq j} \beta_k x_k^i) - \beta_j x_j^i]^2 + \lambda ||\beta_j|| \hspace{0.2cm} (11)$$

This is the same problem as (8) hence it can be solved in closed form. Let $a_j = 2 \sum_i (x_j^i)^2$, $c_j = 2 \sum_i x^i j r_i$. Then,

$$\beta_{j}^{new} = \text{soft}(c_j/a_j, \lambda/(2a_j))$$ \hspace{0.2cm} (12)

A good initialization is the ridge regression estimator $\beta^{ridge} = (X^T X + \lambda I)^{-1} X^T y$. This algorithm has stochastic versions, in which the coordinate $j$ is picked at random (and which are easier to analyze).
4.3 Proximal gradient for the Lasso

Proximal gradient methods have emerged as a very powerful class of methods that work well on regularized estimation problems of the type encountered in machine learning. We will exemplify this method on the (very simple case) of linear regression with $l_1$ regularization.

Assume we have to optimize a function $f$ of the form

$$ f(\beta) = \underbrace{L(\beta)}_{\text{convex, smooth}} + \underbrace{R(\theta)}_{\text{convex}} $$

where the first term is differentiable and strongly convex, while the second is convex but non-smooth. In the Lasso problem, $L(\beta)$ is the quadratic loss, and $R(\beta)$ is the regularization term $\lambda ||\beta||_1$. We assume that the proximal operator $\text{Proxi}_R(x)$ can be computed efficiently (e.g. in closed form).

First idea: approximate $f$ quadratically around the current point $\beta^k$ by

$$ \tilde{f}(z) \approx R(z) + L(\beta^k) + \underbrace{\frac{T}{\nabla L(\beta^k)} (z - \beta^k)}_{\nabla^2 L(\beta^k) \approx \frac{1}{t_k} I} + \frac{1}{2} (z - \beta^k)^T \nabla^2 L(\beta^k) (z - \beta^k). \quad (14) $$

Second idea: The Hessian is approximated by $\frac{1}{t_k} I$, where $t_k = 1/\eta^k$ will be the optimal step size (think Newton method); $\eta^k$ is found by solving analytically

$$ \min_{\eta} \|\eta (\beta^k - \beta^{k-1}) - (g^k - g^{k-1})\|^2. \quad (15) $$

This gives $\eta^k = \frac{(\beta^k - \beta^{k-1})^T (g^k - g^{k-1})}{||\beta^k - \beta^{k-1}||^2}$ which is known as the Barzilai-Borwein step (Exercise: where have you encountered this formula before?)

Third idea: is to let $\beta^{k+1}$ be the optimizer of the approximation (14), i.e.

$$ \beta^{k+1} = \arg\min_z R(z) + g^T z + \frac{1}{2t_k} ||z - \beta^k||^2 $$

$$ = \arg\min_z \frac{1}{2t_k} z [t_k R(z) + \frac{1}{2} ||z - (\beta^k - t_k g^k)||^2 $$

$$ = \text{Proxi}_{t_k R}(\beta^k - t_k g^k) \quad (18) $$

For the Lasso, the update is $\beta^{k+1} = \text{soft} (\beta^k - t_k g^k, t_k \lambda)$, where $g^k = 2X^T (X \beta^k - y)$ is the gradient of the quadratic loss. This method is not guaranteed to reduce $f$ at each iteration.
Nesterov’s methods accelerate on proximal gradient by taking the quadratic approximation around another point that $\beta^k$, as for example in

$$
\phi^k = \beta^k + \frac{k - 1}{k + 2} (\beta^k - \beta^{k-1}) \quad (19)
$$

$$
g^k = \nabla L(\phi^k) \quad (20)
$$

$$
\beta^{k+1} = \text{Prox}_{t^k R}(\phi^k - t^k g^k) \quad (21)
$$

(with $t^k$ found by e.g. line search).

### 4.4 The Lasso regularization path is piecewise linear

This is a simple argument showing that $\beta^*(\lambda)$ changes linearly with the change in $\lambda$, locally near almost every point on the regularization path.

First, note that the least squares loss in the first term of the Lasso is a quadratic function of $\beta$; let us denote it by $\frac{1}{2} \beta^T P \beta + q^T \beta + \text{constant}$.

Assume $\lambda$ is fixed, and that $S = \text{supp} \beta^*(\lambda)$ is the set of components of $\beta$ that are non-zero in this solution. Hence $\beta^*_S(\lambda)$ is the solution of

$$
\arg\min_{\beta_S} \frac{1}{2} \beta_S^T P_{S,S} \beta_S + (q_S + \lambda v)^T \beta_S \quad (22)
$$

where $v$ is a vector with $\pm 1$ elements, so that $v_j = \text{sgn} \beta_j$, $j \in S$. Thus, we can rewrite $\lambda \| \beta_S \|_1 = \lambda v^T \beta_S$. The problem (22) above is a standard least-squares problem and its solution is

$$
\beta_S = -P_{S,S}^{-1} (q_S + \lambda v) \quad (23)
$$

(provided $P_{S,S}$ is non-singular). So, as long as the support of $\beta^*(\lambda)$ doesn’t change, the solution $\beta_S$ is an affine function of $\lambda$. (We also state without proof that the number of changes in the support $S$ is finite; in fact it is $2n$.)

The LARS algorithm exploits this fact, constructing analytically the regularization path between consecutive knots (i.e. points where the support $S$ changes). It can be shown that there are no more than $2n$ knots in the Lasso solution, and that computing the whole regularization path takes about the same amount of computation as solving the full unpenalized regression.
5 A fast primal-dual algorithm for equality constrained minimization

Problem: \( \min_x f_0(x), \) s.t. \( Ax = b \) with \( A \in \mathbb{R}^{m \times n}, \ m < n. \) The dual variable is \( \nu \in \mathbb{R}^m, \) and the Lagrangian is

\[
L(x, \nu) = f_0(x) + \nu^T(Ax - b) = f_0(x) + \nu^T Ax - \nu^T b
\] (24)

An infeasible start algorithm for this problem is

**Initialize** with \( \nu^0 \in \mathbb{R}^m, \) step size \( \eta > 0 \)

- for \( k = 0, 1, \ldots \)
  1. find \( x^+ = \arg \min_x f_0(x) + \nu^T Ax \)
  2. \( \nu^{k+1} = \nu^k + \eta(Ax^+ - b) \)

This is known as the Uzawa method. Note that it does not require finding a feasible \( x^0, \) and that \( \nu \) is updated by the primal residual \( Ax - b \) as in the infeasible Newton iteration.