Convex optimization - MVA 2017/2018
Homework 3

Rémi Lespinet

1 Interior Points Method

If $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^p \to \mathbb{R}^n$, then

$$\nabla(f \circ g)(x) = \mathrm{D}g(x)^T \nabla f(g(x))$$

and if $f : \mathbb{R}^n \to \mathbb{R}^q$ and $g : \mathbb{R}^p \to \mathbb{R}^n$, then

$$\mathrm{D}(f \circ g)(x) = \mathrm{D}f(g(x))\mathrm{D}g(x)$$

- Q1: We have

$$\nabla\phi(x) = Qx + p$$

and

$$\frac{\partial B}{\partial x_j}(x) = -\frac{1}{x_j}$$

Let Ψ be the function

$$\begin{array}{rccc} \Psi: & \mathbb{R}^N & \to & \mathbb{R}^N \\ & (x_1, \dots, x_N) & \mapsto & \left(\frac{1}{x_1}, \dots, \frac{1}{x_N}\right) \end{array}$$

We can write

$$\nabla B(x) = -\Psi(x)$$

If we note f the function

$$\begin{array}{rccc} f: & \mathbb{R}^d & \to & \mathbb{R}^N \\ & x & \mapsto & b - Ax \end{array}$$

It's jacobian is given by

$$\mathrm{D}f(x) = -A$$

Thus

$$\boxed{\nabla\phi_t(x) = t(Qx + p) + A^T\Psi(b - Ax)}$$

- Q2: Obviously,

$$\nabla^2 B(x) = \mathrm{D}\Psi(x) = \mathrm{diag}\left[-\frac{1}{x_1^2}, \dots, -\frac{1}{x_d^2}\right]$$

Let Ψ be the function

$$\begin{array}{rccc} \Theta\colon & \mathbb{R}^N & \to & \mathbb{R}^N \\ & (x_1,\dots,x_N) & \mapsto & \operatorname{diag}\left(\frac{1}{x_1^2},\dots,\frac{1}{x_N^2}\right) \end{array}$$

We can rewrite the precedent expression as

$$\nabla^2 B(x) = \mathrm{D}\Psi(x) = -\Theta(x)$$

Then

$$\mathrm{D}(\Psi \circ f)(x) = \Theta(b - Ax)A$$

Hence,

$$\boxed{\nabla^2 \phi_t(x) = tQ + A^T \Theta(b - Ax)A}$$

2 Support Vector Machine Problem

- Q1: For the primal problem (SVM-P), we can take

$$z = (1+\epsilon,\dots,1+\epsilon)$$

and

$$w = (0,\dots,0)$$

For the dual problem (SVM-D), we can take

$$\lambda = (\frac{1}{2\tau n},\dots,\frac{1}{2\tau n})$$

- Q2:

For the primal, we obtain

$$X = \begin{bmatrix} w_1 \\ \vdots \\ w_d \\ z_1 \\ \vdots \\ z_N \end{bmatrix} \quad P = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \frac{1}{\tau n} \\ \vdots \\ \frac{1}{\tau n} \end{bmatrix} \quad Q = \left[\begin{array}{c|c} \overbrace{\begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix}}^{d} & \overbrace{\begin{matrix} 0 \end{matrix}}^{n} \\ \hline \begin{matrix} 0 \end{matrix} & \begin{matrix} 0 \end{matrix} \end{array}\right]$$

and,

$$A = \left[\begin{array}{c|c} \overbrace{\begin{matrix} -y_1x_1 \\ -y_2x_2 \\ -y_3x_3 \\ \vdots \\ -y_Nx_N \end{matrix}}^{d} & \overbrace{\begin{matrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \\ & & & & -1 \end{matrix}}^{n} \\ \hline \begin{matrix} \\ \\ \\ 0 \\ \\ \\ \end{matrix} & \begin{matrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \\ & & & & -1 \end{matrix} \end{array}\right] \quad B = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \\ -1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

For the dual, we notice that this is equivalent to

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|\sum_{i=1}^{N} \lambda_i y_i x_i\|_2^2 - \mathbf{1}^T\lambda \\ &\text{subject to} \quad 0 \leq \lambda \leq \frac{1}{\tau n} \end{aligned}$$

Hence,

$$X = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{N-1} \\ \lambda_N \end{bmatrix} \quad P = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \\ -1 \end{bmatrix}$$

we have $Q = S^T S$ with

$$S = \left[\begin{array}{c|c|c|c|c} y_1x_1 & x_2y_2 & x_3y_3 & \dots & x_Ny_N \\ & & & & \end{array}\right]$$

Indeed,

$$S\lambda = \sum_{i=1}^{N} \lambda_i x_i y_i$$

and,

$$A = \left[\begin{array}{c} \overbrace{\begin{matrix} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & -1 & \\ & & & & -1 \end{matrix}}^{n} \\ \hline \begin{matrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 1 \end{matrix} \end{array}\right] \begin{matrix} \} n \\ \\ \} n \end{matrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \frac{1}{\tau n} \\ \frac{1}{\tau n} \\ \vdots \\ \frac{1}{\tau n} \\ \frac{1}{\tau n} \end{bmatrix}$$

TODO : we should exploit the structure

- Q3: **Effect of τ**

The figure 1 shows the performance of the classifier on the Iris dataset (considering only the 2 classes *Iris-versicolor* and *Iris-virginica*) as a function of τ. Each sample has a probability $p = 0.8$ (Bernoulli distribution) of being selected for the training, $mu = 10$, $tol = 0.01$. For the curve to be smoother, I ran the procedure 20 times and average the peformance ratio obtained on these 20 iterations.

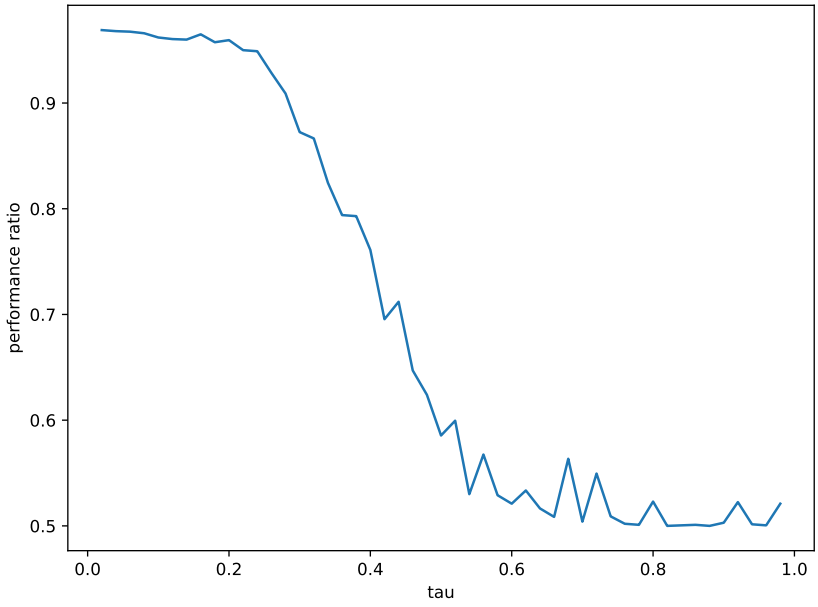

FIGURE 1 – Performance of the classifier as a function of τ, obtained via the procedure described above

I've also generated data in $\mathbb{R}^2$ using a mixture of gaussian, the result of the decision boundary produced by the SVM classifier on these data is shown in the figure 2. In both cases, we see that τ has a significant impact on the performance of the classifier (in term of classification error), this is expected, because setting a high value of τ gives more importance to the regularity term $\frac{1}{2}\|w\|^2$ and less to $\frac{1}{N}\sum_{i=1}^{N} z_i$ in the primal problem, and this is the most important term for the classifier to perform well.

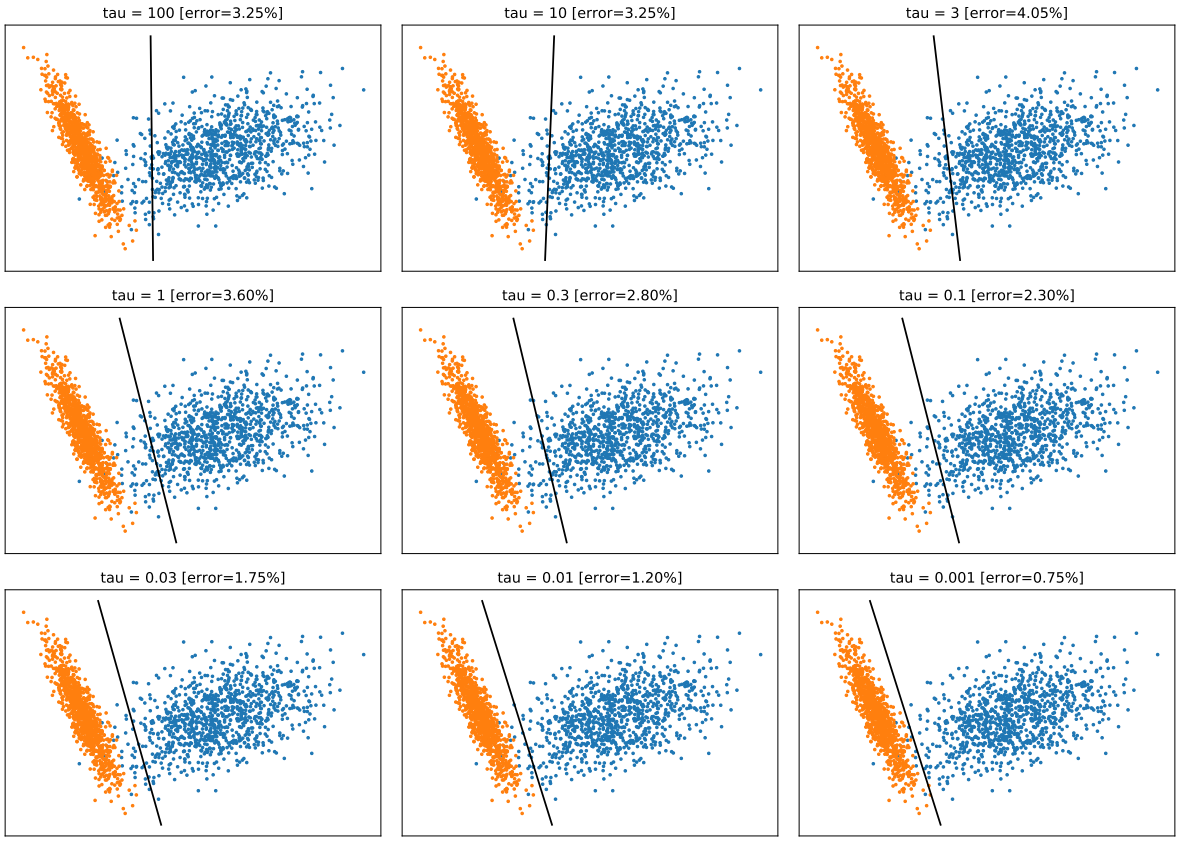

FIGURE 2 – Representation of the boundary produced by the SVM classifier on a gaussian mixture for different values of τ, classifier trained on 400 points (200 blue, 200 orange) and evaluated for 2000 points (1000 blue, 1000 orange), $\mu = 10$, $tol = 10^{-3}$

• Q4: The duality gaps for the primal and the dual problem, for different values of μ are represented in the figure 3.

The figure shows that the number of barrier steps, decreases as μ increases. This is of course not free, since we are multiplying t by a higher value at each iteration of the barrier, the log barrier is relaxed and the Newton method called at each iteration is searching in a much larger space, which requires more Newton iterations. We are effectively delegating the complexity to the inner Newton calls.

The table 1 presents the total number of step needed to solve the primal and the dual SVM problems on the Iris dataset for the Damped Newton method and the Newton method with line search and a tolerance $\epsilon = 10^{-5}$ and $\tau = 0.01$. We see that the total number of Newton iteration does not decrease as fast as the number of barrier steps, which confirms the previous analysis.

We can also notice that the Newton method with backtracking line search requires less iteration to converge compared to the Damped Newton method in our case. Since Newton steps of both method have roughly the same costs (the complexity added by the backtracking part can be resumed to computing the function at several points, which in our case can be done in $O(n^2)$. This cost is negligeable when compared to solving $\nabla^2 f(x)\Delta x = \nabla f(x)$ (even if it could potentially be multiplied by a high number of backtracking iteration, in practice, this is never the case)). Hence the Newton Method with backtracking search is performing faster than the Damped Newton when training a SVM classifier on the Iris dataset.

	$\mu = 2$	$\mu = 15$	$\mu = 50$	$\mu = 100$
Damped Newton (Primal)	323	270	275	273
Damped Newton (Dual)	281	224	231	228
Newton Line search (Primal)	165	60	57	69
Newton Line search (Dual)	158	62	53	57

TABLE 1 – Total number of iterations needed to train the SVM classifier on the Iris dataset, $tol = 10^{-5}$, $\tau = 0.01$

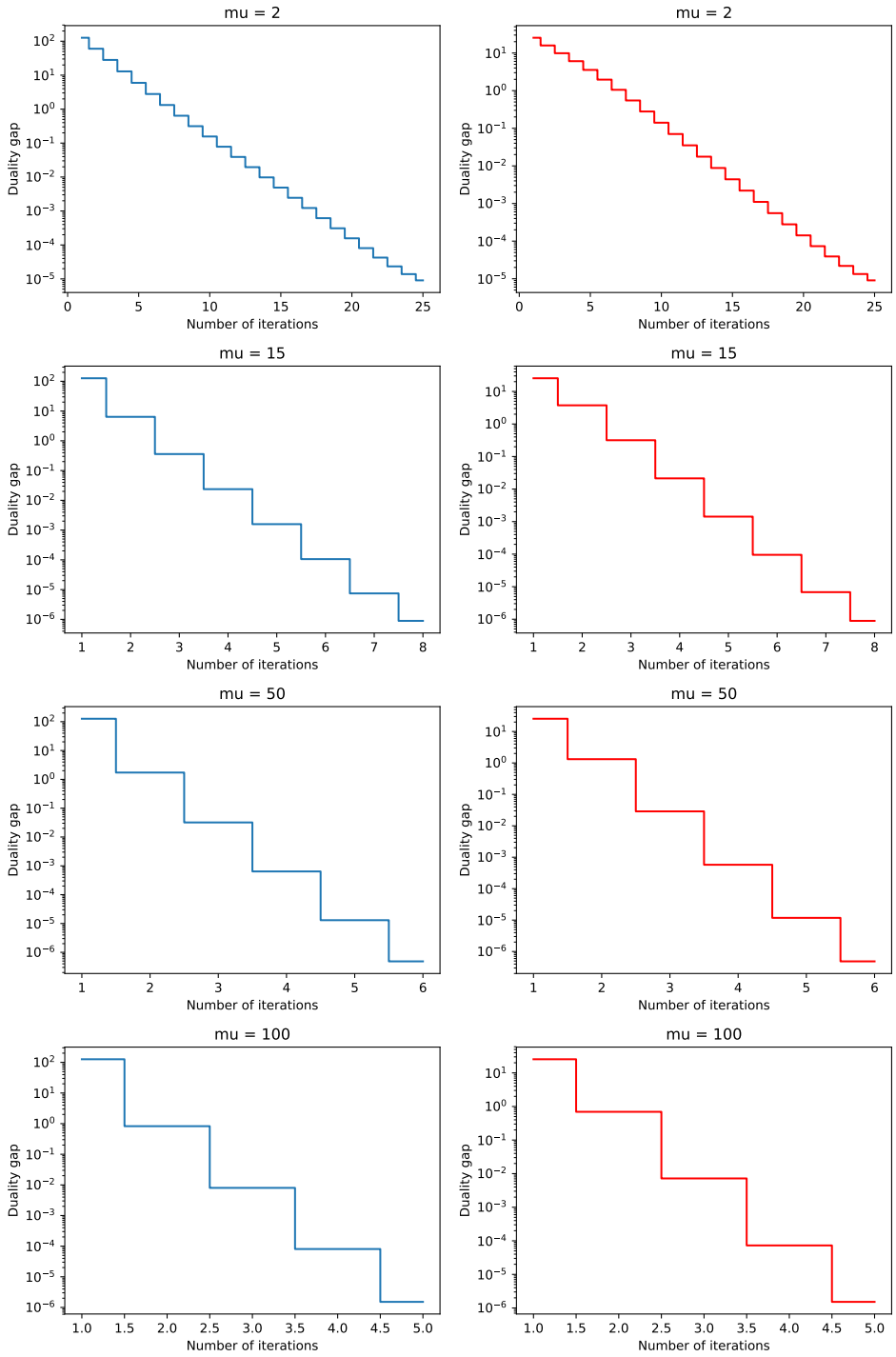

FIGURE 3 – Representation of the duality gap of the primal and the dual problem (in semilog scale) for different values of μ (2, 15, 50, 100), $\epsilon = 10^{-5}$ $\tau = 0.1$