Multi dimensional code

1 Lagrangian

The objective function we are mini-maximizing is given by the Lagrangian:

$$\mathbb{E}\left[g(\nabla\phi(X))\right] - \mathbb{E}\left[e^{g(Y)}\right]$$

for $X \sim \mu$, $Y \sim \nu$, two random variables in $\mathbb{R}^d$.

Our sample based approximation, given $x_1, \dots, x_N$ i.i.d. samples from μ and $y_1, \dots, y_M$ i.i.d. samples from ν, is:

$$\sum_{i=1}^{N} n_i g(\nabla\phi(x_i)) - \sum_{j=1}^{M} m_j e^{g(y_j)} + P(\phi, g)$$

where P is a penalization term, and the *weights* n_i, m_j are given by:

1. $n_i = \frac{1}{N}$ and $m_i = \frac{1}{M}$, in data science applications (where we really observe the samples)
2. n_i, m_j to be normalized intensities in the case of black and white pictures (Lagrangian is easily extensible to colored pictures).

The function $g : \mathbb{R}^d \to \mathbb{R}$, $\phi : \mathbb{R}^d \to \mathbb{R}$ and thus $\nabla\phi : \mathbb{R}^d \to R^d$. Note that $\nabla\phi$ should really be denoted as $\nabla_x\phi$ to indicate we are taking a gradient w.r.t. the argument of ϕ, unlike what we will do in the implicit gradient descent scheme. Similarly, the dummy variable used for g is y, and hence the gradient of g w.r.t. its argument will be denoted as $\nabla_y g$.

2 Functional space

Let's treat the case of a generic multidimensional normal distribution first.

We will parametrize g and ϕ using quadratic functions, and Gaussians. The most general case is given by:

$$\begin{cases} \phi(x) = \frac{1}{2}x^\top(I+S)x + v\cdot x - \sum_{p=1}^{P} d_p \exp\left(-\frac{1}{2}||T_p(x-m_p)||^2\right) \\ g(z) = \frac{1}{2}z^\top Lz + w\cdot z + c + \sum_{q=1}^{Q} e_q \exp\left(-\frac{1}{2}||W_q(z-c_q)||^2\right) \end{cases}$$

S, L are symmetric matrices of $\mathbb{R}^d$, T_p, W_q upper triangular matrices (check positivity, and how to enforce it) of $\mathbb{R}^d$, v, m_p, w, c_q are vectors of $\mathbb{R}^d$ and c, d_p, e_q are scalars.

$\nabla_x\phi$ is thus given by:

$$\nabla_x\phi(x) = (I+S)x + v + \sum_{p=1}^{P} d_p T_p^\top T_p(x-m_p)\exp\left(-\frac{1}{2}||T_p(x-m_p)||^2\right)$$

2.1 More details about the parameters

The subsection below will **not** be the way we will compute the derivatives w.r.t. to the parameters, as it would be too laborious and dimension dependent.

It is just included as a reference.

We will use the vector a to denote all parameters for ϕ, and b for g.

If we choose P generic Gaussians for ϕ, we need

$$\frac{d(d+1)}{2} + d + P\left(1 + \frac{d(d+1)}{2} + d\right)$$

coefficients for a;

- $a_0, \ldots, a_{d(d+1)/2-1}$ for S
- $a_{d(d+1)/2}, \ldots, a_{d(d+1)/2+d-1}$ for v
- For $p = 1, \ldots, P$:
  - $a_{p(\frac{d(d+1)}{2}+d)+p-1}$ for d_p
  - $a_{p(\frac{d(d+1)}{2}+d)+p}, \ldots, a_{p(\frac{d(d+1)}{2}+d)+p-1+\frac{d(d+1)}{2}}$ for T_p
  - $a_{p(\frac{d(d+1)}{2}+d)+p+\frac{d(d+1)}{2}}, \ldots, a_{p+(\frac{d(d+1)}{2}+d)+p-1+\frac{d(d+1)}{2}+d}$ for m_p

Similarly, if we choose Q generic Gaussians for g, we need

$$\frac{d(d+1)}{2} + d + Q\left(1 + \frac{d(d+1)}{2} + d\right) + 1$$

coefficients for b;

- $b_0, \ldots, b_{d(d+1)/2-1}$ for L
- $b_{d(d+1)/2}, \ldots, b_{d(d+1)/2+d-1}$ for w
- $b_{d(d+1)/2+d}$ for c
- For $q = 1, \ldots, Q$:
  - $b_{q(\frac{d(d+1)}{2}+d)+q}$ for e_q
  - $b_{q(\frac{d(d+1)}{2}+d)+q+1}, \ldots, b_{q(\frac{d(d+1)}{2}+d)+q+\frac{d(d+1)}{2}}$ for W_q
  - $b_{q(\frac{d(d+1)}{2}+d)+q+\frac{d(d+1)}{2}+1}, \ldots, b_{q(\frac{d(d+1)}{2}+d)+q+\frac{d(d+1)}{2}+d}$ for c_q

3 Implicit Gradient descent

3.1 Laborious way

When doing an implicit gradient descent on the Lagrangian, we need to compute the twisted Gradient and Hessian, which are given by derivatives w.r.t. to the coefficients a, b. If we do it coefficient by coefficient, we would get:

$$\partial_{a_k} L = \sum_i n_i \nabla_y g(\nabla_x \phi(x_i)) \cdot \partial_{a_k} \nabla_x \phi(x_i) + \partial_{a_k} P$$

$$\partial_{b_n} L = \sum_i^N n_i \partial_{b_n} g(\nabla_x \phi(x_i)) - \sum_j m_i \partial_{b_n} g(y_j) e^{g(y_j)} + \partial_{b_n} P$$

The Hessian is given by:

$$\partial_{a_k a_l} L = \sum_i n_i \nabla_y g(\nabla_x \phi(x_i)) \cdot \partial_{a_k a_l} \nabla_x \phi(x_i) + [\partial_{a_l} \nabla_x \phi(x_i)]^T \nabla_y^2 g(\nabla_x \phi(x_i)) \partial_{a_k} \nabla_x \phi(x_i) + \partial_{a_k a_l} P$$

$$\partial_{a_k b_n} L = \sum_i n_i \partial_{b_n} \nabla_y g(\nabla_x \phi(x_i)) \cdot \partial_{a_k} \nabla_x \phi(x_i) + \partial_{a_k b_n} P$$

$$\partial_{b_n b_m} L = \sum_i n_i \partial_{b_n b_m} g(\nabla_x \phi(x_i)) - \sum_j m_i \left[\partial_{b_n} g(y_j) \partial_{b_m} g(y_j) + \partial_{b_n b_m} g(y_j)\right] e^{g(y_j)} + \partial_{b_n b_m} P$$

Hence we are 'only' required to compute the quantities involving derivatives of ϕ, g w.r.t. x, y, a, b and plug them in.

Again, doing it variable by variable is quite laborious; even though a lot of these derivatives are similar to each other, the computation becomes dimension dependent.

3.2 Tensor calculus, Einstein summation

Instead, we can get rid of this dimension dependency by taking derivatives w.r.t. to vectors and matrices.

For example,

$$\nabla_a L = \begin{pmatrix} \nabla_S L \\ \nabla_v L \\ \nabla_{d_1} L \\ \nabla_{T_1} L \\ \nabla_{m_1} L \\ \vdots \\ \nabla_{d_P} L \\ \nabla_{T_P} L \\ \nabla_{m_P} L \end{pmatrix}$$

All the derivatives involving $\nabla_{d_p} L, \nabla_{T_p} L, \nabla_{m_p} L$ are similar, so we only really need to compute the 5 first ones in this example.

Note that the derivatives w.r.t. the matrices will be **flattened** to vectors.

The structure is very similar with the other first and second derivatives. In general, the rule is: For a variable A only involving coefficients of a ($A = S, v, d_p, T_p, m_p$) and a B only using coefficients of b ($B = L, w, c, e_q, W_q, c_q$), then:

$$\nabla_A L = \sum_i n_i \nabla_y g(\nabla_x \phi(x_i)) \nabla_A \nabla_x \phi(x_i) + \nabla_A P$$

With of course a clear abuse of notation. What I really mean here is that we need to construct:

- A tensor of n_i of dimension N
- A tensor of $\nabla_y g(\nabla_x \phi(x_i))$ of dimensions $N \times d$
- A tensor $\nabla_A \nabla_x \phi(x_i)$ of dimensions $N \times d \times |A|$
- The contractions [1] of the tensors are natural: Contract the dimensions d of $\nabla_y g(\nabla_x \phi(x_i))$ and $\nabla_A \nabla_x \phi(x_i)$, then contract the dimensions N of all tensors. What is left is a tensor of dimension $|A|$, as expected.

Similarly:

$$\nabla_B L = \sum_i^N n_i \nabla_B g(\nabla_x \phi(x_i)) - \sum_j m_i \nabla_B g(y_j) e^{g(y_j)} + \nabla_B P$$

Is of dimension $|B|$. This is because:

[1] contract means multiply element-wise then sum

- n_i of dimension N
- $\nabla_B g(\nabla_x \phi(x_i)), \nabla_B g(y_j)$ is of dimension $N \times |B|$
- $e^{g(y_j)}$ is of dimension N
- Contract on the dimension N appropriately to get a dimension $|B|$ tensor.

This notation might appear cumbersome, but will really come in handy for the Hessian part;

$$\nabla_{AA'} L = \sum_i n_i \left(\nabla_y g(\nabla_x \phi(x_i)) \nabla_{AA'} \nabla_x \phi(x_i) + [\nabla_{A'} \nabla_x \phi(x_i)]^T \nabla_y^2 g(\nabla_x \phi(x_i)) \nabla_A \nabla_x \phi(x_i) \right) + \nabla_{AA'} P$$

- n_i of dimension N
- $\nabla_y g(\nabla_x \phi(x_i))$ is of dimension $N \times d$
- $\nabla_{AA'} \nabla_x \phi(x_i)$ is of dimension $N \times d \times |A| \times |A'|$
- $\nabla_A \nabla_x \phi(x_i)$ is of dimension $N \times d \times |A|$ (same if replace A by A')
- $\nabla_y^2 g(\nabla_x \phi(x_i))$ is of dimension $N \times d \times d$
- Now we have to be careful. Obviously all dimensions N gets contracted together, and in the end.
- The first term $\nabla_y g(\nabla_x \phi(x_i)) \nabla_{AA'} \nabla_x \phi(x_i)$ gets contracted on the d dimension, to get an $N \times |A| \times |A'|$ tensor.
- The second term $[\nabla_{A'} \nabla_x \phi(x_i)]^T \nabla_y^2 g(\nabla_x \phi(x_i)) \nabla_A \nabla_x \phi(x_i)$ means that: We contract the d dimension of $\nabla_{A'} \nabla_x \phi(x_i)$ with the first d dimension of $\nabla_y^2 g(\nabla_x \phi(x_i))$, and the d dimension of $\nabla_A \nabla_x \phi(x_i)$ with the second d dimension of $\nabla_y^2 g(\nabla_x \phi(x_i))$.
- The end result will also be a tensor of dimensions $N \times |A| \times |A'|$. Contracted on N with n_i yields the desired $|A| \times |A'|$ matrix.

For the cross term:

$$\nabla_{AB} L = \sum_i n_i \nabla_B \nabla_y g(\nabla_x \phi(x_i)) \nabla_A \nabla_x \phi(x_i) + \nabla_{AB} P$$

- n_i of dimension N
- $\nabla_B \nabla_y g(\nabla_x \phi(x_i))$ is of dimension $N \times d \times |B|$
- $\nabla_A \nabla_x \phi(x_i)$ is of dimension $N \times d \times |A|$
- This one is easy: Contract the d dimension together, then all the N dimension. You get an $|A| \times |B|$ matrix.

Finally, for the last term:

$$\nabla_{BB'} L = \sum_i n_i \nabla_{BB'} g(\nabla_x \phi(x_i)) - \sum_j m_i \left[\nabla_B g(y_j) \nabla_{B'} g(y_j) + \nabla_{BB'} g(y_j) \right] e^{g(y_j)} + \nabla_{BB'} P$$

- n_i of dimension N
- $\nabla_{BB'} g(\nabla_x \phi(x_i))$ is of dimension $N \times |B| \times |B'|$
- Contract on N to get a $|B| \times |B'|$
- $\nabla_A \nabla_x \phi(x_i)$ is of dimension $N \times d \times |A|$
- m_i of dimension M
- $e^{g(y_j)}$ of dimension M

- $\nabla_B g(y_j)$ of dimension $M \times |B|$ (similar for $|B'|$)
- Contract all on M only to get a $|B| \times |B'|$
- $\nabla_{BB'} g(y_j)$ of dimension $M \times |B| \times |B'|$, which contracted on M with m_i and $e^{g(y_j)}$ gives a $|B| \times |B'|$

The command np.einsum in Python will do all the contraction job efficiently for us. For example, if I have the operation $[\nabla_{A'} \nabla_x \phi(x_i)]^T \nabla_y^2 g(\nabla_x \phi(x_i)) \nabla_A \nabla_x \phi(x_i)$ to do correctly, I will type:

$$\text{np.einsum('ijk ,ijl ,ilm} \rightarrow \text{mk'}, \nabla_{A'} \nabla_x \phi(x_i), \nabla_y^2 g(\nabla_x \phi(x_i)), \nabla_A \nabla_x \phi(x_i))$$

This means the first tensor $\nabla_{A'} \nabla_x \phi(x_i)$ will have its axis $N \times d \times |A'|$ be referred respectively as i, j, k. Similarly to the other inputs.

All common indices that do not appear on the right of the arrow are contracted. In the end, I'm asking for a tensor $'mk'$, which corresponds to $|A| \times |A'|$ with my dummy variables convention.

3.3 Derivatives

All of this sounds complicated but is very easy to automate, since the structure is always the same. The hard and annoying part is obviously to compute the partials of $\nabla_x \phi$, g, $\nabla_y g$ w.r.t. the parameters A, B.

More specifically, we need:

- $\nabla_A \nabla_x \phi(x_i)$ of dimension $N \times d \times |A|$
- $\nabla_y g(\nabla \phi(x_i))$ of dimension $N \times d$
- $\nabla_B g(\nabla \phi(x_i))$ of dimension $N \times |B|$
- $\nabla_B g(y_j)$ of dimension $M \times |B|$
- $\nabla_{AA'} \nabla_x \phi(x_i)$ of dimension $N \times d \times |A| \times |A'|$
- $\nabla_{yy} g(\nabla \phi(x_i))$ of dimension $N \times d \times d$
- $\nabla_B \nabla_y g(\nabla \phi(x_i))$ of dimension $N \times d \times |B|$
- $\nabla_{BB'} g(\nabla_x \phi(x_i))$of dimension $N \times |B| \times |B'|$
- $\nabla_{BB'} g(y_j)$of dimension $M \times |B| \times |B'|$

for $A = S, v, d_p, T_p, m_p$ and $B = L, w, c, e_q, W_q, c_q$. Nearly all of them are straightforward, except those involving T_p, W_q, and to a lesser extent m_p, e_q.