CS 294-115 Algorithmic Human-Robot Interaction **Fall 2016**

Lecture 7: Trajectory Optimisation Part III

Scribes: Daniel Filan and Davis Foote

7.1 Summary

This lecture will have two sections: first, an explanation of why and how to change the inner product implicitly used to define functional gradient descent, and secondly, a discussion of how student paper presentations will work, and why they will work that way.

7.2 Changing the inner product

For ease of understanding, we will think about our trajectories as discretised vectors, whose i^{th} component is the position in configuration space of the trajectory at the i^{th} point in (discrete) time. This vector will look something like

$$\xi = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \vdots \\ q_N \end{bmatrix} \tag{7.1}$$

Since we won't consider changing the start or end points of the trajectory, we leave out the initial configuration q_0 and the final configuration q_{N+1}. We should also remember that each element of this vector is itself a vector, rather than simply a scalar.

So far, we have been using the Euclidean inner product, defined as

$$\langle \xi_1, \xi_2 \rangle_E = \xi_1^\top \xi_2 = q_{1,1}^\top q_{2,1} + \cdots + q_{1,N}^\top q_{2,N} \tag{7.2}$$

Why would we want to change this? Well, an inner product defines a norm by $\|\xi\| = \sqrt{\langle \xi, \xi \rangle}$, and a norm defines a distance metric defined by $\text{distance}(\xi_1, \xi_2) = \|\xi_1 - \xi_2\|$. Therefore, we might ask ourselves whether the Euclidean metric induced by the Euclidean inner product is any good.

7.2.1 Why the Euclidean metric isn't good

Consider three trajectories of a robot with one degree of freedom:

DIAGRAM HERE

$$a = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 0 \\ 10 \\ 0 \\ 0 \end{bmatrix}, \text{ and } c = \begin{bmatrix} 0 \\ 5 \\ 10 \\ 5 \\ 0 \end{bmatrix} \tag{7.3}$$

Which is closer to a: b or c? Well,

$$\|a - b\|_E^2 = \langle b, b \rangle_E = 10 \times 10 = 100 \tag{7.4}$$

and

$$\|a - c\|_E^2 = \langle c, c \rangle_E = 5 \times 5 + 10 \times 10 + 5 \times 5 = 150 \tag{7.5}$$

Therefore, according to the Euclidean metric, b is closer to a than c is. However, intuitively, there's something wrong with this. a involves very smooth motion, b is super jerky, and c involves a bit of a jerk, but less so than b. The Euclidean metric can't notice this because it doesn't take the orderings of elements of the vector into account, and therefore can't talk about speeds, but it would be nice if we could take time into account somehow.

Can we give a more rigorous argument for why we should change metric?

7.2.1.1 Detour 1

The formula for gradient descent, as we know, is $\xi_{i+1} = \xi_i - (1/\alpha)\nabla_{\xi_i}\mathcal{U}$. But where does this come from?

Suppose instead that we approximate $\mathcal{U}$ by its first order Taylor expansion, and try to minimise that. Then, we would use the rule

$$\xi_{i+1} = \arg\min_{\xi} \left\{ \mathcal{U}[\xi_i] + \nabla_{\xi_i}\mathcal{U}^\top (\xi - \xi_i) \right\} \tag{7.6}$$

However, we know that this Taylor expansion will be off if we get far away in norm from ξ_i, so we add a regularisation term to penalise this:

$$\xi_{i+1} = \arg\min_{\xi} \left\{ \mathcal{U}[\xi_i] + \nabla_{\xi_i}\mathcal{U}^\top (\xi - \xi_i) + \frac{1}{2}\alpha\|\xi - \xi_i\|_E^2 \right\} \tag{7.7}$$

This is a quadratic problem in ξ, so we can take the gradient of the term inside the curly brackets, set the gradient to zero, and thereby solve the problem. Doing this, we get

$$0 + \nabla_{\xi_i}\mathcal{U} + \alpha(\xi - \xi_i) \tag{7.8}$$

Setting that expression to 0, we obtain

$$\xi_{i+1} = \xi_i - \frac{1}{\alpha}\nabla_{\xi_i}\mathcal{U} \tag{7.9}$$

which is exactly the formula for gradient descent.

7.2.1.2 Back to the main track

Remember the example of vectors a, b, and c? Well, gradient descent tends to go to things that are closer in norm, all else being equal. So, if b is closer to a than c is, even if c is better than b, gradient descent might well reach b first, because of how much further away c is.

What's the solution? We will change the distance metric (by changing the inner product) so that the sphere of elements of constant distance away from a becomes an ellipsoid, thereby making c closer than b.

DIAGRAM HERE

7.2.2 How to make a new inner product

All inner products are of the form $\xi_1^\top A \xi_2$, where A is a positive semi-definite matrix. Therefore, we just have to choose a matrix A. How?

Since we want our metric to like smooth deformations, let's think of other things that like that. Remember $\mathcal{U}_{\text{smooth}}$? That was defined as

$$\mathcal{U}_{\text{smooth}}[\xi] = \frac{1}{2}\int_0^T \|\xi'(t)\|^2 dt \tag{7.10}$$

When we discretise, the analogue of this is

$$\mathcal{U}_{\text{smooth}}[\xi] = \frac{1}{2}\sum_{i=0}^{N} \|q_{i+1} - q_i\|^2 \tag{7.11}$$

Suppose we compute the partial derivative of $\mathcal{U}_{\text{smooth}}$ with respect to q_i. This is

$$\frac{\partial}{\partial q_i}\mathcal{U}_{\text{smooth}} = (\nabla_\xi \mathcal{U}_{\text{smooth}})_i = -(q_{i+1} - q_i) + (q_i - q_{i-1}) = 2q_i - q_{i-1} - q_{i+1} \tag{7.12}$$

We're going to form a matrix A such that

$$\nabla_\xi \mathcal{U}_{\text{smooth}} = A\xi + c \tag{7.13}$$

where the constant c will appear because $\nabla_\xi \mathcal{U}_{\text{smooth}}$ depends on the endpoints q_0 and q_{N+1}, which don't appear in ξ. The matrix that does this is

$$A = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix} \tag{7.14}$$

This A (the Hessian of $\mathcal{U}_{\text{smooth}}$) will be what we use for our inner product:

$$\langle \xi_1, \xi_2 \rangle_A = \xi_1^\top A \xi_2 \text{ and therefore } \|\xi_1 - \xi_2\|_A^2 = (\xi_1 - \xi_2)^\top A (\xi_1 - \xi_2) \tag{7.15}$$

Going back to our motivational example, we now have that

$$
\begin{aligned}
\|b-a\|_A^2 &= \|b\|_A^2 \\
&= \begin{bmatrix} 0 & 0 & 10 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 10 \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 10 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -10 \\ 20 \\ -10 \\ 0 \end{bmatrix} \\
&= 200 \\
\|c-a\|_A^2 &= \begin{bmatrix} 0 & 5 & 10 & 5 & 0 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 5 \\ 10 \\ 5 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 5 & 10 & 5 & 0 \end{bmatrix} \begin{bmatrix} -5 \\ 0 \\ 10 \\ 0 \\ -5 \end{bmatrix} \\
&= 100
\end{aligned}
$$

So it worked!

[there was a bit here where it talked about how gradient descent would now push a three-point trajectory to become uniform, but I don't see how that works on reflection, since we don't know what the cost function is, and that could well be higher for smoother gradients]

7.2.3 How to do gradient descent now

blah blah blah