ISTA 421 / INFO 521 – Final Project Description

Due: Monday, December 11, 5pm

Total 15% of final grade

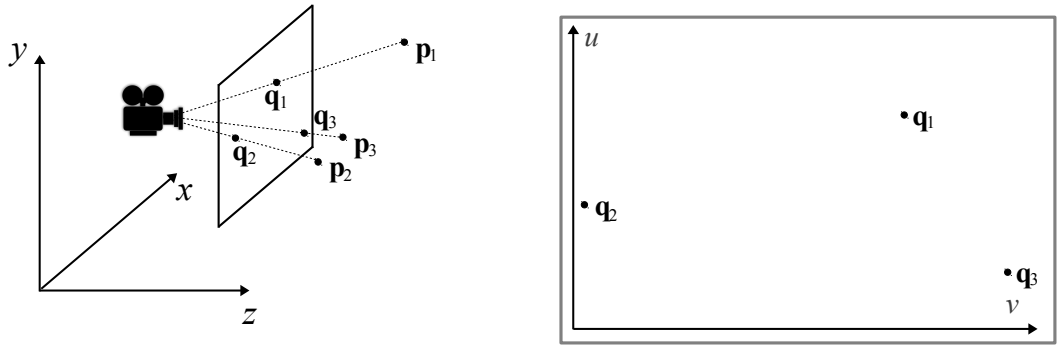
Instructions

The final course project will be worth 15% of your final grade. There are two options for the final class project. **You only need to complete ONE**:

A Inferring parameters of a 3D line from a noisy 2D image using Metropolis Hastings.

B Project of your choosing. I provide some brief instructions below, but because this is more open-ended, you **must** email me by **Friday, November 17**, with a brief description of what you plan to do so that I can ensure it will be worth the full 15% credit. If you do not talk with me beforehand, you can still submit the for this option, but you are NOT guaranteed that I will accept full 15% credit if the project is not deemed substantial enough, at my discretion.

The final project is due Monday December 11, 5pm.



(a) Graphical depiction of the projection of three points $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3 \in \mathbb{R}^3$ onto the image plane of a camera using Eqs 1 and 2.

(b) Graphical depiction of the 2D image plane with image points $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3 \in \mathbb{R}^2$ projected from points $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$.

Figure 1: Two views of projection.

Option A: Inferring parameters of a 3D line from a noisy 2D image using Metropolis Hastings

In the pinhole camera model, cameras can be represented by matrices that contain parameters for the camera, such as its position in space and its focal length.[1] This representation is useful because it allows us to obtain images of 3D points in a very simple way.

Let $\mathbf{M} \in \mathbb{R}^{3\times 4}$ be a camera matrix. Then, if $\mathbf{p} \in \mathbb{R}^3$ is a 3D point (i.e., $\mathbf{p} = [p_1, p_2, p_3]^\top, p_i \in \mathbb{R}$), its 2D image $\mathbf{q} \in \mathbb{R}^2$ is given by

$$\mathbf{q} = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\widetilde{w}} \begin{bmatrix} \widetilde{u} \\ \widetilde{v} \end{bmatrix} \tag{1}$$

where $\widetilde{u}$, $\widetilde{v}$, and $\widetilde{w}$ are defined by

$$\begin{bmatrix} \widetilde{u} \\ \widetilde{v} \\ \widetilde{w} \end{bmatrix} = \mathbf{M} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix} \tag{2}$$

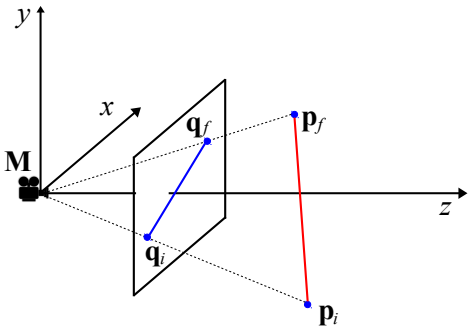
Notice the number 1 that is appended to $\mathbf{p}$ when multiplying by $\mathbf{M}$. This is called the *homogeneous* coordinate, and it is necessary for posing camera projections as matrix multiplication. This projection principle is illustrated in Figure 1

NOTE: A complete understanding of this material, while very valuable in itself (!), is not necessary to complete this problem. The only thing you are required to understand is that 3D points become 2D image pixel points by Equations 1 and 2.

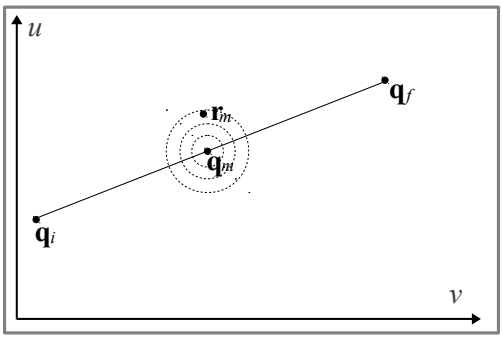
Throughout this exercise we will assume that we have a camera whose matrix is

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

[1] For those interested (*not* required for this project), details can be found in Hartley and Zisserman (2004) *Multiple View Geometry in Computer Vision*; you can also internet search for "camera matrix."



(a) A 3D line segment $\mathbf{l} = (\mathbf{p}_i, \mathbf{p}_f)$ (in red) projected onto the image *via* camera $\mathbf{M}$ (situated at the origin), producing a 2D line $\mathbf{l}_{2D} = (\mathbf{q}_i, \mathbf{q}_f)$ (in blue).



(b) Image $\mathbf{l}_{2D} = (\mathbf{q}_i, \mathbf{q}_f)$ of the projected 3D line; $\mathbf{q}_m$ is one of the sampled points along the line, which has noise, ϵ, added to it in the image-creation process, generating the observed image point $\mathbf{r}_m$; the concentric circles around $\mathbf{q}_m$ represent the contours of the 2D Gaussian noise, ϵ.

Figure 2: The noisy generative process projecting points from the line in the image

which corresponds to a camera located at world coordinates $(0, 0, 0)$, pointing down the positive z-axis, with unit focal length. What follows is the problem setup.

Let $\mathbf{l} = (\mathbf{p}_i, \mathbf{p}_f)$ be a 3D line segment with endpoints $\mathbf{p}_i, \mathbf{p}_f \in \mathbb{R}^3$. (Note: the symbol for the line segment is the bold-faced lower-case-L not the number one; also, the subscripts on the endpoints stand for "initial" and "final", respectively.) Now, consider the following generative process. We "take a picture" of $\mathbf{l}$ using camera $\mathbf{M}$, producing $\mathbf{l}_{2D} = (\mathbf{q}_i, \mathbf{q}_f)$ (that is, the $\mathbf{q}_i$ and $\mathbf{q}_f$ are obtained using Equations 1 and 2). Then we take S points along $\mathbf{l}_{2D}$ at inputs $t_1, ..., t_S$ so that $\mathbf{q}_s = \mathbf{q}_i + (\mathbf{q}_f - \mathbf{q}_i)t_s$, with $0 \leq t_s \leq 1$, $s = 1, ..., S$. Since the imaging and rendering processes are inherently noisy, we add Gaussian noise to the $\mathbf{q}_s$, which gives us $\mathbf{r}_s = \mathbf{q}_s + \epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, with $\mathbf{\Sigma}$ a 2×2 covariance matrix. Naturally, this implies that $\mathbf{r}_s | \mathbf{p}_i, \mathbf{p}_f, t_s, \mathbf{\Sigma} \sim \mathcal{N}(\mathbf{q}_s, \mathbf{\Sigma})$. See Figure 2 for an illustration of this generative process.

We will additionally place a Gaussian prior on the line (this is a different Gaussian than the Gaussian noise ϵ; the parameters here are subscripted by the bold-$\mathbf{l}$ to indicate they refer to the properties of the 3D line). This represents the prior belief about where the line, as defined by end-points $\mathbf{p}_i$ and $\mathbf{p}_f$, might be in 3D space:

$$\mathbf{p}_i, \mathbf{p}_f \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{l}, \mathbf{\Sigma}_\mathbf{l}).$$

Tasks

NOTE: Some of the details of the tasks will be made more concrete once the data is released on Monday, November 20.

All data files needed for this task will be provided in `project1-3D-line-MH/data/`

For this project option, you will need to complete the following five tasks:

1. Write a python script that, given
   - inputs $t_1, ..., t_S$

- 2D points $\mathbf{r}_1, ..., \mathbf{r}_S$
- values for $\mathbf{\Sigma}$, $\boldsymbol{\mu}_\mathbf{l}$, and $\mathbf{\Sigma}_\mathbf{l}$,

uses the Metroplis-Hastings algorithm to sample from the posterior of $\mathbf{p}_i$ and $\mathbf{p}_f$. In other words, the script should generate samples of $\mathbf{p}_i$ and $\mathbf{p}_f$ according to the posterior distribution

$$p(\mathbf{p}_i, \mathbf{p}_f | \mathbf{r}, \mathbf{t}, \mathbf{\Sigma}, \boldsymbol{\mu}_\mathbf{l}, \mathbf{\Sigma}_\mathbf{l}) \propto p(\mathbf{r} | \mathbf{p}_i, \mathbf{p}_f, \mathbf{t}, \mathbf{\Sigma}) p(\mathbf{p}_i, \mathbf{p}_f | \boldsymbol{\mu}_\mathbf{l}, \mathbf{\Sigma}_\mathbf{l}),$$

with $\mathbf{r} = \{\mathbf{r}_1, ..., \mathbf{r}_S\}$ and $\mathbf{t} = \{t_1, ..., t_S\}$.

NOTE: The prior and likelihood values can be *extremely* small, to the point that their product can result in floating-point number underflow errors – i.e., the computer cannot represent the small values and so treats them as though they were zero; this particularly becomes a hazard when you take the product of two very small numbers, which can happen when computing the posteriors in the Metropolis-Hastings acceptance ratio, which in turn can lead to "division by zero" errors (when the denominator posterior is treated as zero). A work-around is to take the log (remember, we follow the machine learning convention that this is really the *natural log*) of the acceptance ratio (be sure to carry the log completely through the ratio and products), do your numeric computations to get the log-ratio r; you can turn the final value back into the original scale by computing e^r, or keep it in the log scale (as long as you also ensure your uniform random sample is also log scale!). If you use the log scale, the python function `scipy.stats.multivariate_normal.logpdf` will be handy.

2. Use your script to find the MAP estimate of the 3D line segment using the inputs and data contained in the files `inputs.csv` and `points-2d.csv`, with $\mathbf{\Sigma}_\mathbf{l} = 10 \cdot \mathbf{I}$ (where here $\mathbf{I}$ is the 3×3 identity matrix), $\boldsymbol{\mu}_\mathbf{l} = [0, 4, 0]^\top$, and $\mathbf{\Sigma} = (0.05)^2 \cdot \mathbf{I}$ (where here $\mathbf{I}$ is the 2×2 identity matrix and $(0.05)^2$ is the scalar value of the square of 0.05). Report your results. Generate a plot like that of Figure 4.12(c), page 162 of FCML, for each parameter you are estimating, to demonstrate that you have converged to sampling from the posterior distribution. Include these plots in your results, and describe their features. By this, I mean that you should point out any trends of moving up or down in the beginning sampling phase before then varying around particular region of the parameter's space. In Figure 4.12(c), the trend upward happens very early, in the first $200 - 500$ samples or so, before it finally "converges". In this problem, your plots are likely to look a bit different than FCML Fig 4.12(c) – they will generally take more steps before convergence (maybe between steps 500 and 2000 – it will vary by parameter), and the samples won't vary quite as broadly (won't be as "jagged" as in Fig 4.12(c)), but you should see convergence.).

3. Use your script to find the Monte Carlo estimate of the mean of the posterior distribution (i.e., $\mathbb{E}_{p(\mathbf{p}_i, \mathbf{p}_f | \mathbf{r}, \mathbf{t}, \mathbf{\Sigma}, \boldsymbol{\mu}_\mathbf{l}, \mathbf{\Sigma}_\mathbf{l})} \{\mathbf{p}_i\}$ and $\mathbb{E}_{p(\mathbf{p}_i, \mathbf{p}_f | \mathbf{r}, \mathbf{t}, \mathbf{\Sigma}, \boldsymbol{\mu}_\mathbf{l}, \mathbf{\Sigma}_\mathbf{l})} \{\mathbf{p}_f\}$), using the inputs and result values from task 2. Report your results.

4. Use your script to find the Monte Carlo estimate of the predicted (2D) output point at a point at $t_* = -0.5$, using the inputs and result values from task 2. Report your results.

5. Suppose we have a second camera

$$\mathbf{M}' = \begin{bmatrix} 0 & 0 & 1 & -5 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 5 \end{bmatrix},$$

(a camera at $[5, 0, 5]^\top$ looking in the negative x-axis and unit focal length), and, consequently, a second set of observed data points $\mathbf{r}'_1, ..., \mathbf{r}'_S$, where the indices of these points correspond to the same points viewed from the first camera (as given in `points-2d.csv`). Compute the MAP estimate of the line using the same setup as in task 2, plus points found in `points-2d-2.csv` as the point viewed by the second camera. Report your results. Are the results the same? Why or why not?

Option B: Project of your choosing

You can come up with your own project idea that might involve any machine learning method, including methods we have not covered in class. This could can include contrasting 2 or more methods. You will preferably demonstrate the performance of the method(s) on real-world data, but could also use synthetic data if it does a good job demonstrating the strengths and weaknesses of the method(s). You must include in your project an evaluation of the method that involves appropriate training and testing. You will submit the following in with your final project:

1. Any data you used to train/test your method. The data in your submission must be open source (no license restrictions). If you are using data that you did not generate or collect yourself, you must clearly specify where it came from and make appropriate attribution / citations. If the data is very large and is hosted on a server, you can submit instructions for how to retrieve it; in this case, you should still include in your submission a small portion of the data on which you can still run your code, so that your submission can be run stand-alone. If the data your are interested in working with is proprietary, talk to me to come up with an alternative (one option is to create synthetic data similar to your proprietary data and use that for your final submission).
2. Code you used in your project. All code must be open source. It *is* ok for you to use code written by others as long as it is open source. However, *you* must make a significant contribution to the code configuration used in your project. **All code must be well-documented**. If you are using code that an individual or research group developed, you MUST ensure it is well-documented; if it is not, you must document it yourself, clearly describing what the component parts do. You can use larger libraries, like SciKit-Learn – in this case, you do not have to submit the code base or document it (that is sufficiently done). If you have any questions about this, be sure to talk to me before your submission. In any case, you must include instructions that explain how to use the code and the data in your submission in order to reproduce all of the results you report. One general source of data is the UC Irvine Machine Learning Repository: `https://archive.ics.uci.edu/ml/`
3. A **concise and clear** written pdf that includes the following:
   (a) Introduction describing the method (including relevant equations and what they mean, algorithm, etc.); describe the type of problem the method(s) is designed for. Include citations with summary descriptions of any relevant background material you are deriving this work from.
   (b) Procedure: the procedure you followed to generate any results,
   (c) Evaluation: describe your evaluation method and rationale for it's use,
   (d) Results: include a clear description of the results of your method; include any figures/tables of data summarizing the results. All plots/tables must be labeled.

To ensure that your proposed project will be worth the full 15% final grade, you must communicate with me (email can be sufficient; office hours is also good) by **Friday, November 17**.