Research questions

Ferran Alet
Learning and Intelligent Systems, CSAIL-MIT
alet@mit.edu

Description

This series of problems allows us to see if we are a good research match. These problems are meant to be somewhere in between a Machine Learning class and research you would do with us (papers 1.2,1.3 and problem 2.5 being the closest to us); giving you a sense of the types of problems you would face. Second, they will allow us to to see if you need more Machine Learning/programming/math knowledge before starting, which we could certainly help you with.

General instructions

- **You should aim to spend between 10 and 15 hours over the course of at most 10 days.**
- **You should discuss one paper (section 1) and solve one problem (section 2). Aim to finish them, but you're not expected to; we're at least as interested in your thought processes as your actual answers.**
- Feel free to email me if you have any questions, both about the problems or the projects.
- You should code in python, using all the packages that you want. In case you use Deep Learning, preferably use PyTorch, but Keras and Tensorflow are also fine.
- Expected problem difficulty is marked with stars(*,** or ***). It's perfectly fine to try the easiest problem in each section, specially if you're freshman or sophomore.
- If you feel one star (*) problems are too hard, send us an email. We are in the first iteration of this PDF so we haven't calibrated yet.
- You should aim for simple yet effective solutions and short, clear code.
- You are encouraged to read all available resources (such as Stack Overflow). However, you cannot ask any person (except me) for help, including posting a question online.
- After you have spent 10-15 hours, send me an email with your work.

1 Discussing a paper

Choose one paper and answer the questions.

1.1 A Probabilistic Data-driven model for planar pushing (*)

1. Read the paper
2. In your opinion, what are the 1-2 citations that are most important to this paper? Why?
3. What do you like about this paper? What do you think could be improved?
4. Why do we model $\Delta x, \Delta y, \Delta\theta$ and not the $x_{new}, y_{new}, \theta_{new}$?
5. Why is the standard GP doing well in NMSE but doing poorly in NLPD?
6. Explain figure 11 and what it is trying to determine. Propose another experiment, using only available data, (possibly similar to the current experiment, but no need) that computes the same quantity in a more direct way. There is no single right answer.

1.2 Modular Multitask Reinforcement Learning with Policy Sketches (**)

1. Read the paper
2. In your opinion, what are the 1-2 citations that are most important to this paper? Why?
3. What do you like about this paper? What do you think could be improved?
4. What is the role of curriculum learning?
5. Look at the code to find the size of the map in the crafting environment. Suppose you start in the middle of the grid and an object is in a corner; how many steps do you need to get there if you act optimally? Give a rough estimate of the expected number of steps you would need to get there by moving randomly.
6. In the cliff environment, imagine you have mastered actions North, East, West but have never seen the 'South' primitive. Now you receive a policy sketch which also contains action 'South', which of the following do you think is true and why?
   (a) North-South-East-West is faster to learn
   (b) South-North-West-East is faster to learn
   (c) Both equally are equally easy to learn

1.3 Neural Relational Inference(**)

We recommend being familiar with graph neural networks before reading this paper.

1. Read the paper
2. In your opinion, what are the 1-2 citations that are most important to this paper? Why?
3. What do you like about this paper? What do you think could be improved?
4. Give 1-2 other examples of environments that could be modeled using this paper.
5. Look at table 1. Why do think that the gap supervised-unsupervised is bigger for charges than for springs?
6. Instead of using a Graph Neural Network to predict the edge between two nodes, we could train an LSTM that receives the concatenation of both states for all timesteps and outputs the edge label probabilities. What is the main flaw in this approach?
7. Look at the results of the static algorithm for Kuramoto.
   (a) Do you notice anything weird?
   (b) What do you think is (approximately) the maximum (not minimum) error that can be achieved in Kuramoto?

2 Solving a problem

Choose one problem.

2.1 2D world(*)

1. Build a 2D grid where each square can be either empty or full. Code a generator that builds a random grid with a fraction f of full squares.
2. Implement a BFS to find a path from one free square to another moving up-down-left-right.
3. Given a grid and the moves done by an agent (sequence of squares) find all potential goals consistent with the agent moving optimally.

2.2 Binary MNIST (*)

From MNIST, you can construct lots of binary MNIST datasets by only looking at images with either of two labels; removing 80% of the data. For example, you could only look at 1s and 2s and build a 1-vs-2 binary classifier.

1. Build 25 datasets on X vs Y for every (X,Y) pair where X is even and Y is odd. Train one binary classifier for each dataset.
2. Discuss, but don't implement, how you would get the missing 2-way classifiers where X,Y have the same parity using the trained 25 binary classifiers.
3. Implement a 10-way classifier (the usual one) using the trained 25 binary classifiers; without any extra parameters nor retraining.
   Is it better to use the binary output or the logistic/softmax output?
4. Implement a 10-way classifier (the usual one) using less than 12 trained binary classifiers; without any extra parameters nor retraining.

2.3 Pixel MNIST (**)

MNIST images are 28x28=784 pixels; but not all pixels are created equal. In this problem we want to classify MNIST digits by only looking at a few pixels.

1. Design and implement an MNIST classifier that only gets to see 10 pixels.
   Which pixels to look for should be decided by the program. For example, a simple solution would be to try 100 random sets of 10 pixels, train a softmax classifier for each set, and pick the one performing the best in a validation set.
2. Design and implement an MNIST classifier that sequentially queries 10 pixels: picks a pixel, observes it, picks another pixel, observes it,..., and makes a decision after 10 pixels. Therefore, the queried pixels will be image dependent.

2.4 Math mini-problems(**)

Warning: problems in this section are not necessarily in increasing order of difficulty.

1. Solve question 1 in subsection 2.5.
2. Solve question 2 in subsection 2.5.
3. Let T_1, T_2 be exponential distributions with timescales (inverse means) α, β. Compute:
   (a) $\mathbb{E}[T_1 - T_2]$
   (b) Probability that $T_1 > 2T_2$.
4. What are second order optimization methods? Why aren't they used to optimize Neural Networks?
5. Imagine you test a robotic system 5 times and it fails 2 times.
   (a) Give the confidence intervals for its true probability of success.
   (b) Given a uniform prior [0,1] over its probability of success, compute the probability that the true probability of success is at least 75%. You can solve this question programatically if you don't know how to solve it with math.
6. In an experiment we are building shapes composed of 4 pieces joined to a central square with magnets, see figure 1. We have 4 types of pieces and 4 copies of each piece (enough to build any shape). Solve the following question, both with a small python program and a short math computation
   (a) How many different shapes can we build if the orientation of the shape matters?
   (b) Notice that some shapes in the previous question are the same up to a rotation of 90,180 or 270 degrees. How many shapes can we build taking these symmetries into account?

2.5 Meta-learning; permuted MNIST (***)

Permuted MNIST is a meta-learning problem where datasets are generated by picking MNIST images, permuting the pixels with the same permutation for all images, and keeping the labels.

1. Let S_n be the set of permutations of n elements ($|S_n| = n!$). As you probably know permutations can be composed; for example, in S_3 if you have the permutation $p = \{1 \rightarrow$

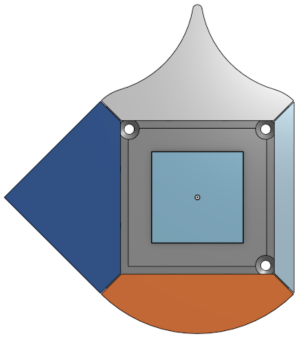

Figure 1: Sample figure with one piece of each type (see last question in subsection 2.4)

$2, 2 \to 3, 3 \to 1\}$, usually denoted $(2, 3, 1)$, then $p \circ p = (3, 1, 2)$. Find a small set of basic permutations that span S_n, i.e. a set of permutations that can be composed to generate all permutations.

2. Given the basis you found in the previous question, what is the maximum number of compositions needed to span any element in S_n? If that number is not $O(n)$, find another set that allows you to do it in $O(n)$.
3. How would you approach the permuted MNIST problem? Write a description, but only implement it if you don't get to solve question 6.
4. Read Modular meta-learning
5. How would you approach the permuted MNIST problem with MOMA? How many modules do you think are needed and why?
6. Implement it (using the MOMA code in https://sites.google.com/view/modular-meta-learning or from scratch) and explain the differences between your expectations and what you ultimately found out.