Swyft & Summary Statistics

Swyft and its Simulators & Embedding Nets

- Swyft is the official implementation of the SBI code that uses TMNRE – estimates likelihood-to-evidence ratios for marginal posteriors
- Prior truncation is used (the T) which combines simulation efficiency with testability
- Based on stochastic simulators – map parameters stochastically to observational data – swyft makes it convenient to define these simulators as graphical models
- Simulators: swyft.Simulator module; build graph with the nodes being some sort of randomness at the node points
- Embedding net used: Linear layer, but also these LogRatioEstimators – this is the ratio that is described above, and there are two different versions – one for single-dimensional posteriors and one for multi-dimensional posteriors
- LogRatioEstimator_1dim inputs – num_features (size of the data vector), num_parameters (number of variables), varnames (labelling of the vectors)
- More complicated embedding nets: including optimised learning rate scheduler, fully connected layers, CNNs/RNNs
- Simulators are all classes of the form class Simulator(swyft.Simulator) – all of them involve defining some form of random variables and then defining build(self, graph) function; the simulator is then activated by saying sim = Simulator(), and then we can sample directly from there by sim.sample(N)
- The embedding net is defined by a class Network(swyft.SwyftModule), then with network = Network(), trainer = swyft.SwyftTrainer(accelerator = DEVICE), dm = swyft.SwyftDataModule(samples, batch_size = 64) and trainer.fit(network, dm) trains the network

Crafting Summary Statistics

- Summary statistics condense the data that is parsed into the model into latent spaces that are a representation of the data that will be used in ML
- Well-designed summary statistics – enhance computational efficiency, retain only the most relevant parts of the data, highlight the most interpretable sections of the data and summarise the data in a way that is specific to solving a certain problem
- Embedding networks are particularly good at crafting these summary statistics when hand-picking these summary statistics is not clear or infeasible – large volumes of data/high dimensions are mapped to low dimensional embeddings which are the summary statistics
- Good embedding networks have: automated feature learning (opposed to manually setting up them), a lot of flexibility and scalability and can include task specific information
- In SBI, embedding networks can learn summary statistics from complex/high-dimensional simulator outputs directly and reduce the overall model complexity if well designed. They can also adaptively learn summary statistics for inference tasks, rather than manually created fixed features
- Choosing the specific embedding net is often led by the type of the data, i.e. grid like data is often CNN, temporal data is often RNN etc. – also want to be careful not to overfit the data by using unnecessarily complicated embedding nets

Evaluating Embedding Nets

- We want to see how good our embedding nets are – this can be done by just analysing the posterior that we get out of it i.e. posterior coverage, accuracy compared to known results etc. – also we want this process to be more efficient than posterior estimation without an embedding net so can measure the compute etc.
- We can also measure the quality of the embedding by seeing how much relevant information the embedding preserves – one way this can be done is estimating the mutual information between the raw input and the embeddings (MINE)
- We also want embedding nets to be sensitive to perturbations (robust) – these include transformations of the data that does not change its underlying nature e.g. rotations, noise;

adding some small perturbations to the relevant data should not significantly change the embedding; repeatedly doing the same generative process should give similar embedding spaces

- We can also look at the geometric structure of the embedding space – can check if the embeddings corresponding to different data categories are well-separated, or if the embedding space naturally lies on a lower dimensional manifold within the space, so its dimension can be further reduced
- Various other tests exist, including assessing with domain specific information, looking for redundancies in the dimensions of the embedding space and just analysing the results visually