Lecture 18: The sequential design method for modelling–Active learning

So far in the course, we have designed the experiment and the modeling tools separately. While this is the standard procedure, it could be quite prohibitive in terms of data efficiency. For example, imagine your task as mapping out a region of two-dimensional space w.r.to. a measured variable. This could be mapping out a phase diagram of materials with changes to composition and temperature and measuring if we obtained a periodic structure at any given combination or not. Or perhaps designing a material that requires us to apply a specific rate of cooling and application of tensile stress to measure anisotropy. In both the cases above we are dealing with a similar problem as in modeling so a reasonable choice would be to define a grid or one of the randomization design methods discussed so far to collect data and fit it to our model of choice. In the two cases we discussed above, this approach turns out to be a data-inefficient and sometimes intractable approach. One possible reason for this is that measured responses have some spatial correlations that result in redundant information gained from nearby samples. For example, in the case of phase mapping, it is reasonable to expect that samples prepared under similar temperatures and compositions tend to have similar properties thus a query in the two-dimensional space not only reveals a binary label for the location queried but also for the nearby points. A better strategy for the experimental design would be to perform the modeling and data collection in an iterative fashion. In particular, devising a strategy to sequentially query our design space such that after each iteration, we only sample location that has a high likelihood of improving our model prediction capabilities. In the literature, this method is called *active learning* which we define in this lecture and discuss the inner workings in detail in the next class. Our knowledge of local correlations between measured outputs can be encoded using prior knowledge thus we take a Bayesian approach. The Gaussian process is a suitable model choice for this because of its ability to predict the response and uncertainty around each point.

Suppose we have collected the following data set so far at locations $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t\}$ and corresponding measured responses as $\mathbf{y} = \{y_1, y_2, \ldots, y_t\}$. Assuming that the underlying function mapping y to any $\mathbf{x}$ is given by f such that $y \sim f + \epsilon, \epsilon\, \mathcal{N}(0, \sigma)$. Furthermore, we have selected to model f using a GP such that $f \sim \mathcal{N}(\mu, K)$ for a given mean and covariance/kernel function. In the active learning paradigm, our goal is to collect a set of B data points at locations that are going to improve the model we are trying to fit. At each iteration, we query a *selector* that tells us a set of locations $\mathbf{x}$ available for selection. The job of the selector is to keep track of labeled and unlabelled points and provide a set of points for which we want to evaluate the *utility* of being selected. A simple selector would be to return all the points that have not been selected so far or use a more complicated techniques such as a nearest neighbor group where only points that are not the nearest neighbors of any of the labeled points are returned. Once we have selected the points to evaluate, we compute the utility of adding any point from the eligible points to our data repository. Given that our model is a probabilistic one, one approach is to compute uncertainty at each prediction and use a score such as marginal entropy defined as follows:

$$H[y|x, D] = -\sum_i p(y = i|x, D) \log(p(y = i|x, D))$$

$$x^* = \text{argmax}\, H[y|x, D]$$

These are called *query strategies* wherein we select query points that either maximize or minimize score functions. A query strategy such as the maximizer of marginal entropy shown above returns a set of points we can query in our experiment and update our model and data set object, and selector. Putting it all together, we obtain the following pipeline for active learning:

Algorithm 1: Active learning algorithm

Data: $\mathbf{X}, \mathbf{y}, B$
Result: $\{\mathbf{x}, y\}_t^{t+B}$
while *budget not expired* **do**
    x_u = selector($\mathbf{X}, \mathbf{y}$) ;
    x^* = query strategy($\mathbf{X}, \mathbf{y}, x_u$) ;
    y^* = experiment(x^*) ;
    $\mathbf{X} = [\mathbf{X}, x^*], \mathbf{y} = [\mathbf{y}, y^*]$;

Figure 1 illustrates this using a simple example in a regression setting. The query strategy is defined as the one that maximizes the standard deviation of prediction around each point in the domain. The selector is the trivial unlabeled selector that returns all the points that have not been labeled within a linear grid of the domain. The active learning procedure (also referred to as a campaign)

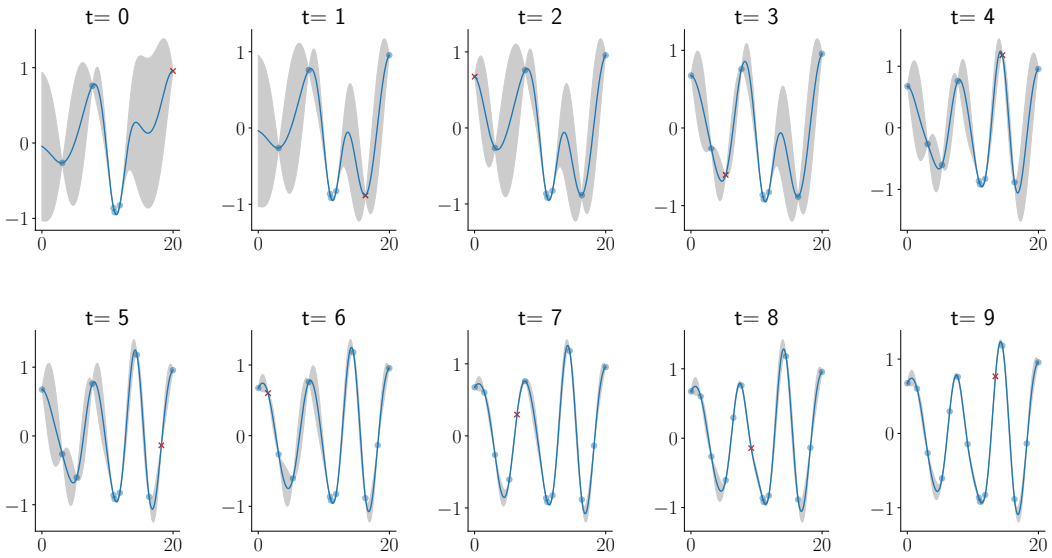

Figure 1: Active learning in regression

started with just four points drawn in blue circles in the subplot titled $t = 0$ and the query strategy returned the point depicted in a red cross mark in the same subplot. This procedure is repeated for the next 10 iterations collecting more data as we go along and improving the model both in terms of predictions and uncertainty.