Bayesian Machine Learning - Lecture 4

Guido Sanguinetti

Institute for Adaptive and Neural Computation
School of Informatics
University of Edinburgh
gsanguin@inf.ed.ac.uk

February 26, 2015
Today’s lecture

1. Active learning
2. Active learning query strategies
3. Bayesian Optimisation
4. The GP-UCB algorithm
More types of learning

- We have considered extensively the supervised learning scenario, where data are input/output pairs.
- Often, the reason why we are interested in predictions is that outputs are difficult/expensive to collect.
- Frequently, many *unlabelled* data instances are available.
- *Semi-supervised learning* aims to improve predictions by somehow exploiting the empirical distribution of the inputs.
- *Active learning* aims to devise query strategies for selecting informative labels.
- **CAVEAT**: neither has any theoretical guarantees of working. But in practice they often do.
Active learning scenarios

- Assumption 1: there is an inexpensive mechanism to generate input points
- Assumption 2: there is an expensive mechanism to discover the label of arbitrary input points (an Oracle)
- Scenario 1: any input point is accessible (membership query synthesis)
- Scenario 2: input points come streaming, you need to choose whether to label or discard (stream-based selective sampling)
- Scenario 3: an empirical distribution of possible inputs is provided, from which we can easily sample (pool based sampling)
Illustrative diagram

Figure taken from B. Settles, Active Learning Literature Survey, available at burrsettles.com/pub/settles.activelearning.pdf
Querying by uncertainty

- We’ll focus on a classification pool based scenario for ease.
- We assume we have a probabilistic model, i.e. something which provides us with a class probability for all input points (based on the observed labels).
- An obvious criterion for selecting a query is to focus on where you know least.
- *Uncertainty sampling* selects as next query the input point for which the highest class probability is lowest.
- *Entropy sampling* selects as next query the input point for which the entropy of the predictive distribution is highest.
- The two are related (but not equivalent if more than two classes).
Querying by committee

- We assume that we have an ensemble of (probabilistic) models.
- The next query point is chosen as the one on which models disagree most.
- Closely related to model selection.
- The criteria for disagreement vary.
- *Vote entropy* maximises the entropy of a vote distribution (drawbacks?)
- *KL selection* maximises the KL-divergence between models and an average model.
Another idea is to select the point that will lead to the largest change in model parameters.

Theoretically, one should loop through all points and all possible outcomes; clearly impossible.

For models trained by gradient-based methods, one can evaluate the expectation of the likelihood gradient and choose the point with largest gradient.
Active learning proposes a dynamic world-view where learning takes place in cycles, intelligently selecting instances to query.

Bayesian Optimisation uses similar ideas to tackle optimisation.

It replaces a hard optimisation problem with an iterative approach where an easier problem is solved at every iteration.

Closely related to Reinforcement Learning (which I will not discuss in this course).
Reasons why optimisation is hard

- Optimisation is the task of finding a global optimum of a function
- Optimisation can be hard because the function has multiple local optima, or because the function argument is very high dimensional
- Optimisation can be hard also because the function is difficult to evaluate
- Example 1: global weather models map parameters and initial conditions to predicted temperature at a spot. This function needs to be computed on a supercomputer
- Example 2: hydro-solubility of a protein depends on its ammino-acid sequence. This function (of a discrete variable) is computed by engineering new proteins!
- Bayesian Optimisation can help in this case
BO key idea and terminology

- At every step of the algorithm, we have a few function evaluations and want to select a point where to evaluate the function next
- **KEY IDEA**: treat the function as a random variable, use the existing function evaluations to compute a posterior distribution over the functions, and use this distribution to select the new point
- The posterior mean is called the *surrogate function*
- The surrogate is used to create an *acquisition function* which is maximised to find the next evaluation
- The measure of success is the *cumulative regret*

\[
R_T = \frac{1}{T} \sum_{t=1}^{T} (f(x_t) - f(x^*))^2
\]

- Let’s see an example
It will come as no surprise that GPs can be used in Bayesian optimisation to construct a surrogate function.

Directly optimising the surrogate however is a bad idea (why?)

One needs a strategy to trade-off exploitation (looking around areas which you know to be promising) and exploration (checking out areas which you don’t know much about).
The GP-UCB rule

- GPs also provide uncertainty quantification (in fact, they provide full distributions on outputs at any point).
- One can trade-off exploration and exploitation by selecting regions where the function could conceivably be high, rather than regions where we expect it to be high.
- The Gaussian Process Upper Confidence Bound (GP-UCB) algorithm maximises the following acquisition function to select its next point:

\[ F(x) = E[f(x)] + \beta_t \sqrt{\text{var}(f(x))} \]

which is an upper quantile of the single point marginals.
- Surprisingly, Srinivas et al proved that this algorithm is globally convergent in probability, i.e. given \( \delta, \epsilon \) with probability \( 1 - \epsilon \) the regret will become smaller than \( \delta \).
The GP-UCB algorithm - example
The GP-UCB algorithm - example

Guido Sanguinetti  Bayesian Machine Learning - Lecture 4
The GP-UCB algorithm - example

Guido Sanguinetti

Bayesian Machine Learning - Lecture 4
The GP-UCB algorithm - example
The GP-UCB algorithm - example
The GP-UCB algorithm - example
Srinivas et al (IEEE Trans Inf Th 2012) linked the expectation of the cumulative regret to a *submodular function*

Submodular functions obey a diminishing returns rule → greedy optimisation provably works for submodular functions

The upper quantile $\beta_t$ term must be increased according to a specific schedule for the guarantees to work
Limitations of GP-UCB

- The cubic scaling of GP regression limits the dimensionality of the space, in my experience anything above 10 is fancyful.
- Optimising the acquisition function is still an NP-hard problem! In fact, the acquisition function can be nastier (more multimodal) than the original function.
- Convergence is not very fast ($O(\sqrt{T})$) → in cases where the true function is really expensive to optimise this could be too much.
Tomorrow’s lab

- Implementation of GP regression-based techniques on a toy example
- Sample functions from a GP with RBF covariance with $\alpha^2 = 4$ and $\lambda^2 = 4$ evaluated on the grid 0.1:0.1:10
- Evaluate the function $f(x) = \exp[-(x - 2)^2] + 2 \exp[-\frac{(x-6)^2}{3}]$ at the points 1:1:9, adding i.i.d Gaussian noise with variance 0.04, to get observations $y$
- Use GP regression with the previous kernel to compute the posterior function values on the grid 0.1:0.1:10
- Use GP-UCB with fixed $\beta = 2$ to optimise the function