Course Project[1]

10-424/10-624 Bayesian Methods in ML (Spring 2025)
https://www.cs.cmu.edu/~hchai2/courses/10624

1 Overview

The course project for 10-424/624 will be a guided exploration of one of the major successes of Bayesian inference in the context of machine learning: Bayesian optimization (BO). Specifically, you will be applying BO to tune the hyperparameters of various machine learning models. We have scaffolded this somewhat daunting task into three smaller components, each of which will have separate deliverables and deadlines:

1. Baseline implementation (§1.1) – due Thursday, March 27th at 11:59 PM
2. Literature review (§1.2) – due Thursday, April 10th at 11:59 PM
3. Advanced implementation and competition (§1.3) – due **Wednesday**, April 30th at 11:59 PM (during final examination period)

You will submit the deliverables for each component on Gradescope; if a component requires you to submit code, there will be a separate code submission slot. Projects in this course are completed individually. As such, you are welcome to pull from your pool of grace days in order to submit project deliverables late; you may use at most three grace days on any single deliverable.

Next, we will describe each of the components listed above in more detail.

1.1 Baseline Implementation

For this component of your project, you will use curated data of some real hyperparameter tuning task from one of three machine learning models: logistic regression, SVM or online LDA. These datasets were used in the seminal work Practical Bayesian Optimization of Machine Learning Algorithms by Snoek, et al. I *highly* encourage you to read this paper, in particular the "Empirical Analyses" section that describes these datasets.

Given some data, the performance of each method was evaluated on a three- or four-dimensional grid of hyperparameter values, defining an objective function $f(\theta)$, where θ is a multi-dimensional hyperparameter vector; note that each model has a different performance metric (again, you are encouraged to read the original paper linked above to learn more about these objective functions). The goal is to maximize the performance of the model as a function of the hyperparameters. As the cost of training and evaluating the performance of these models is relatively high, you will use precomputed grids instead. These datasets are provided to you in the handout alongside this document, in the "Project Data" directory.

You will choose one of these three tasks to work with for this component. For each task, the goal is to minimize the value in the last column given the values of the hyperparameters in the first three or four columns. Note that all of these are *minimization* problems.

We have broken this component into a sequence of smaller steps that you will complete in order. Your final deliverable will be the first section of this handout, which you must fill in using LaTeX,

[1]Compiled on Friday 4th April, 2025 at 21:46

preferably in the provided Overleaf template. You must also submit all code you write for this component in a single file titled `bo_baseline.py`. For this component, you may use `numpy`, `scipy`, `matplotlib` and any of Python's default libraries; you may not use any other packages.

Note that for this assignment, your grade for each question below will primarily be determined by the correctness of your implementations. Please ensure that your code is well-documented and comprehensible. We will also be running your code to reproduce your results: thus, **you must set the random seed you use to generate your results**. Any method of doing so is acceptable as long as you (and therefore, we) can consistently generate the same results.

1.1.1 Model fitting (55 points)

Before you can succeed with optimization, you must first determine whether you can build a useful model of the objective function. If you can't find an informative model that can make reasonable predictions, it would be foolish to try to use the model to make decisions.

1. (0 points) **Select one:** Indicate which of the three machine learning models you have chosen to work with for this component.

- ◯ Logistic Regression
- ◯ SVM
- ◯ LDA

Complete the following steps:

- Sample a set of 32 data points uniformly at random from your chosen machine learning model's dataset.
- Fit a Gaussian process (GP) model to the data you sampled using a constant mean and a squared exponential kernel. Fix the standard deviation of the noise to a small value i.e., 0.001.
- Maximize the marginal log likelihood of the data as a function of the hyperparameters: the constant mean value and the length scale and output scale of the covariance function.

2. (10 points) Report the values you learned for the hyperparameters and the corresponding marginal log likelihood.

3. (10 points) For each of the data points not in your training dataset, compute the z-score of the true value using the posterior mean and standard deviation of your GP at that location:

$$z(\mathbf{x}) = \frac{f(\mathbf{x}) - \mu_{\mathcal{D}}(\mathbf{x})}{\sqrt{k_{\mathcal{D}}(\mathbf{x}, \mathbf{x})}}$$

Plot a kernel density estimate of these z-scores (**Hint:** you may find the function `scipy.stats.gaussian_kde` helpful). If the GP is well calibrated this look something like a standard normal distribution. Is that the case for your model?

Now let's try some different models. Given a GP mean function and covariance, we can derive a score for how well those choices fit a given dataset, such as your randomly sampled dataset of 32 observations above. The most natural score would be the model evidence after integrating over the hyperparameters, but unfortunately this is intractable for Gaussian process models. A widespread and convenient approximation is called the Bayesian information criterion (BIC).

To compute the BIC we first find the values of the hyperparameters maximizing the (log) marginal likelihood:

$$\hat{\theta} = \arg\max_{\theta} \log p(y \mid X, \theta).$$

Now the BIC is a combination of two terms rewarding model fit and penalizing model complexity:

$$\text{BIC} = |\theta| \log |\mathcal{D}| - 2 \log p(y \mid X, \hat{\theta}),$$

where $|\theta|$ is the total number of hyperparameters (in this case, 3) and $|\mathcal{D}|$ is the number of observations (in this case, 32). Given a set of models, we prefer the one with the *lowest* BIC, and this score can actually be interpreted as an approximation to the (negative) log model evidence.

4. (5 points) What is the BIC score for the dataset and model you previously fit?

5. (30 points) Next, fixing the choice of mean function to be a constant mean, you will compare a few different kernels (which correspond to different models).

For each kernel function below, optimize hyperparameters (remember to also optimize the constant mean value as well). Then report the optimal hyperparameters and compute the BIC score using the same dataset you previously sampled.

- squared exponential kernel with automatic relevance determination i.e., where each dimension has its own length scale.
- Matérn kernel with $\nu = 3/2$.
- Matérn kernel with $\nu = 3/2$ and automatic relevance determination.

Which is the best model you found from among the four kernels (including the original squared exponential kernel)?

1.1.2 Bayesian optimization (45 points)

Now for the fun part: fix the model that achieves the best BIC score you computed in the previous part (along with corresponding optimal hyperparameters). You will now implement a basic Bayesian optimization procedure and observe its performance.

- Implement the *expected improvement* (EI) acquisition function, which computes a score for each possible observation location in light of the previously observed data points. Be careful to implement EI for minimization as opposed to maximization!
- Now run the following experiment:
  - Select 5 initial observations uniformly at random from the dataset.
  - Repeat the following steps 30 times:
    - Find the point $\mathbf{x}$ that maximizes the expected improvement acquisition function given the current dataset.
    - Add the observation $(x, f(x))$ to your dataset.
  - Return the final dataset (containing 35 observations)

We will evaluate the performance of EI as follows. Using the final returned dataset, identify the best point found i.e., the one with the lowest objective function value. We will score optimization performance using the "gap" measure, which for minimization is:

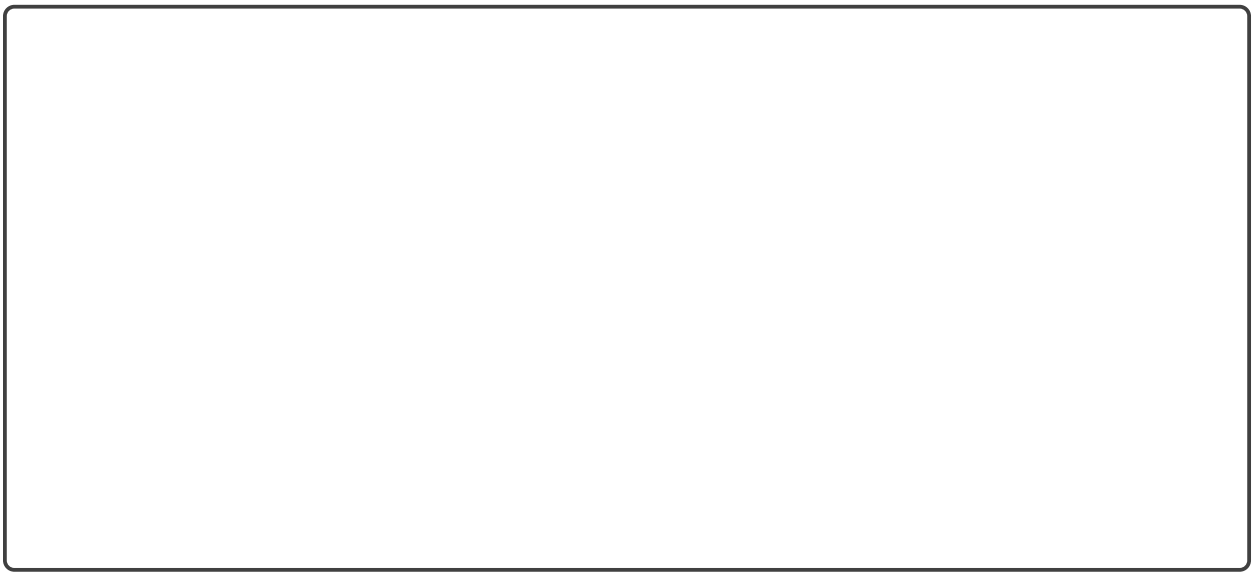
$$\text{gap} = \frac{f(\text{best initial}) - f(\text{best found})}{f(\text{best initial}) - f(\text{true minimum})}.$$

You can interpret gap as the portion of the "gap" between the best initial observation among the 5 initial data points and the optimum that we managed to traverse over the course of optimization; this gives us a normalized score between 0 and 1 for how well we did.

- Perform 20 runs of the above Bayesian optimization experiment using different random initializations, and store the entire sequence of observations for each run.
- For a baseline, we will use random search. **Using the same initializations**, implement random search (i.e., a policy that samples an unlabeled point in the domain uniformly at random). Allow random search to have a total budget of 150 observations rather than 30 (5 times the budget), and store the entire sequence of data.

6. (25 points) Make a plot of "learning curves" for both random search and EI: using just the first 30 observations for random search, plot the average gap achieved by Bayesian optimization and the average gap achieved by random search as a function of the number of observations. Plot both curves on the same axes.

7. (10 points) Compute the mean gap for EI and for random search after 30 observations, 60 observations, 90 observations, 120 observations and 150 observations.



8. (10 points) Perform a paired t-test comparing the performance of random search (using 30 observations) and EI. What is the p-value? How many observations does random search need before the p-value raises above 0.05? We can interpret this result in terms of a "speedup" of Bayesian optimization over random search.

1.2 Literature Review

Next, you will independently explore and research a few ways to improve upon EI, a relatively simple baseline for Bayesian optimization. Specifically, for this component of your project you will perform the following steps:

- Identify and read *two* papers on Bayesian optimization. Each paper must satisfy the following criteria:
  1. The paper is primarily about some method or idea *not* covered in the lectures or readings for this course e.g., Wu et al. (2017), the original paper on the knowledge gradient acquisition function would not be a valid choice.
  2. The paper is recent i.e., written within the last 10 years.
  3. The paper is published in a peer-reviewed venue, e.g., an academic conference or journal. A paper that solely exists on arXiv and does not appear anywhere else would not be a valid choice (arXiv is not a peer-reviewed platform!).
  4. The ideas in the paper are either directly applicable or easily modifiable to work in the setting of single, sequential observations of one black-box objective function. In the competition portion of this project, your implementations will only be allowed to query the objective function one location at a time and thus, the works that you research in this component must fit within that framework.
     - **Note:** this is intentionally a very restrictive requirement. Our hope is that this helps you narrow/guide your research significantly; you can typically tell just from the abstract of a paper if it is likely to satisfy this constraint.
     - As a place to start, most papers that focus on developing acquisition functions (e.g., the Wu et al. (2017) paper) or tailoring the underlying GP model specifically for Bayesian optimization will satisfy this criterion.
- Summarize the key findings of both papers. Specifically you should describe the primary contributions (what is the novel aspect of each work?), the experimental setup used to evaluate the proposed methods and any empirical/theoretical results.
- Compare and contrast the primary method or methods presented in each paper along as many relevant axes as you can come up with. The exact aspects of your discussion will vary based on the papers you select but some examples of relevant points of comparison could include
  - Computational complexity,
  - Theoretical performance guarantees,
  - Difficulty to reimplement,
  - Presence of parameters that require manual tuning, and
  - Relationships with the task of hyperparameter optimization.

  Again, the list above is not meant to be comprehensive in terms of what you can/should discuss, merely a list of starting points to get you thinking about what kinds of factors matter when deciding which method to reimplement.

- Finally, select one of these papers to do a reimplementation of for the final component of the project. Justify your choice based on the analysis you performed in the previous steps. You must clearly state your choice but you do not need to write any code for this component.

The deliverable for this component will be a 2-3 page report on your findings from the steps above; **three pages is a hard limit** but you may include an additional fourth page that contains only references. As always, you must typeset this report using LaTeX; you may use any template you wish to do so (Overleaf has some nice options to choose from).

1.3 Advanced Implementation and Competition

For the final component of your course project, you will perform a re-implementation of the method that you identified in the previous part. You will then compare the performance of your more advanced method against the simple EI baseline that you coded up in the first component. Finally, your method will be pitted against the methods your peers chose in a black-box optimization competition on a neural architecture search benchmark, where the goal is to optimize (a subset of) the hyperparameters for a fully-connected, feed-forward neural network in order to minimize the validation error on some regression task. **Crucially, this is again a minimization task** (much like the benchmarks in the first component of your project): make sure you implement your method for minimization and not maximization.

To ensure compatibility with our competition interface, your implementation must provide a function called `acq_func` that evaluates your method over a tabular set of candidate data points. Specifically, `acq_func` should take as input in order:

1. an N-by-D `numpy` array with the previously observed *locations* (e.g., `X_train`), used to condition the GP posterior;
2. an N-by-1 `numpy` array with the corresponding previously observed *objective function values* (e.g., `y_train`), also used to condition the GP posterior;
3. an M-by-D `numpy` array with the set of candidates that could be chosen as the next location to be observed (e.g., `X_cands`).

It should return **the index of the next location from the set of candidates** that your method chooses to observe, not the actual location itself.

If you wish, `acq_func` can take additional, optional arguments but our scripts will not pass any other arguments to it beyond the ones specified above. Similarly, you may include arbitrary helper functions in your implementation but our evaluation interface will only call `acq_func`.

Your deliverable for this component will be this section of the handout. In addition to answering the questions below, you must also submit all code you write for this component in a single file titled `bo_comp.py`; this includes the code for the competition as well as the code used to run the experiments detailed below. As before, you may use `numpy`, `scipy`, `matplotlib` and any of Python's default libraries in your implementation but you may not use any other packages.

Similar to the first component, your grade for each question below will primarily be determined by the correctness of your implementations. Please ensure that your code is well-documented and comprehensible. We will also be running your code to reproduce your results: thus, **you must set the random seed you use to generate your results**. Any method of doing so is acceptable as long as you (and therefore, we) can consistently generate the same results.

1.3.1 Comparison against EI (25 points)

Before comparing against your peer's methods, you should first confirm that your selected method is able to outperform the relatively simplistic EI acquisition function that you previously implemented; note that outperforming EI is not a requirement and you can still receive full credit for a correct re-implementation that does worse that EI but that would not bode particularly well for your performance in the competition...

With the same dataset and GP model that you used in 1.1.2 (i.e., the kernel function and GP hyperparameters that achieved the best BIC score), run a direct comparison between EI and your method. Specifically, you should run the following experiment 20 times:

- Select 5 initial observations uniformly at random from the dataset and use these to initialize a GP model.
- Run both EI and your chosen method for 30 iterations.
- After each iteration, compute the "gap" metric (as defined in 1.1.2) for both EI and your chosen method.
- Return the sequence of gaps achieved by EI and your method; these should be two sequences of 30 numbers each.

1. (20 points) Make a plot of learning curves for EI and your method by plotting the average gap achieved by each as a function of the number of observations. Plot both curves on the same axes.

2. (5 points) Perform a paired t-test comparing the performance of EI and your method. What is the p-value? Can you confidently conclude that your method outperforms EI? If the results are suggestive but inconclusive, you can consider either

- increasing the number of times you run the experiment above (e.g., from 20 to 30) or
- increasing the budget of observations you allot each method (e.g., from 30 to 50)

Report any such changes to the experimental routine you made, if applicable.

1.3.2 Hidden Benchmark Competition (25 points)

In the handout for this component of the project, you will find a file titled `sample_data.csv`, which contains a sample of 40 data points from the hidden benchmark. The first 7 columns are the features of the dataset and the last column contains the objective function value, which in this setting is the validation loss after 100 epochs of training.

You will fit an initial GP model to these samples, again selecting from among the kernels you compared in the first component of this project:

- squared exponential kernel,
- squared exponential kernel with automatic relevance determination i.e., where each dimension has its own length scale,
- Matérn kernel with $\nu = 3/2$,
- Matérn kernel with $\nu = 3/2$ and automatic relevance determination.

3. (15 points) For each of the kernels listed above, optimize the corresponding hyperparameters by maximizing the log marginal likelihood of the provided samples. Report the optimal hyperparameters for each model as well as the BIC score: which is the best from among the four options?

4. (10 points) Finally, using the kernel and hyperparameters you identified as optimal in the previous question, submit your code to the Gradescope assignment for this component. We will initialize each of your BO routines with the same 40 data points you were provided with for learning GP hyperparameters.

Our scripts will run your method for an additional 60 iterations of BO and report the final gap, after 100 total observations of the objective function have been made. Your score for this question will be assessed as follows:

1. If your method is able to beat a baseline random search that is allocated 160 additional observations, you will receive **6 points**.
2. If your method is in the top 50% of student submissions on our Gradescope leaderboard, you will receive **2 points**.
3. Finally, the top 3 student submissions will each receive an additional **2 points**.

You will be able to monitor your position on the Gradescope leaderboard in real-time. To limit brute-force attacks on our benchmark, **you will only have 10 total submissions to the competition**. We strongly encourage you all to debug locally as any submissions that fail to run on Gradescope will still count against your submission limit.

Good luck!