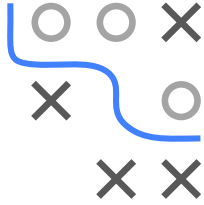
# Optimization in Machine Learning

## Bayesian Optimization
## Practical Aspects of BO

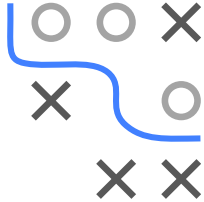


**Learning goals**

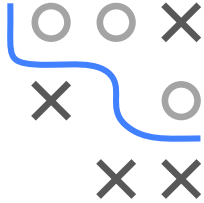
- Size of the initial design
- Optimizing the acquisition function
- When to terminate
- BO Components and robustness

# SIZE OF THE INITIAL DESIGN

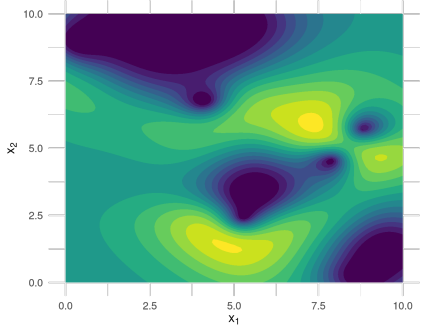
- Should not be too small
- Should not be too large
- Scale with the dimensionality of the search space
- Certain SM may impose restrictions on the lower bound of the size of the initial design
- Rule of thumb $4d$



# OPTIMIZING THE ACQUISITION FUNCTION

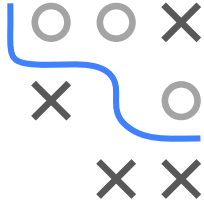


- Optimizing the acquisition function to find the next candidate point is comparably cheap
- Still can be a hard optimization problem: non-linear, multimodal
- Properly optimizing the acquisition function can be crucial for performance




Example EI landscape for a 2D problem.

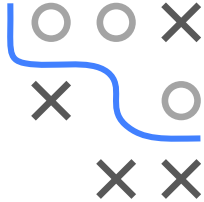
# OPTIMIZING THE ACQUISITION FUNCTION



- Optimizing the acquisition function to find the next candidate point is comparably cheap
- Still can be a hard optimization problem: non-linear, multimodal
- Properly optimizing the acquisition function can be crucial for performance
- Choice of optimizer depends on the search space
  - Numeric: L-BFGS-B with restarts, DIRECT, ...
  - Mixed: EAs, local search with restarts, ...
  - A random search can always be used but may not find a good solution (even if the budget is large)
- Sometimes, gradient-based optimizers can be used (e.g. when using a GP as SM)

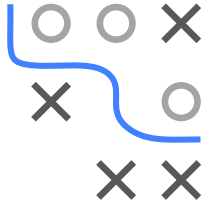
# WHEN TO TERMINATE

- After a certain number of evaluations
  - Potentially scaling the budget with the dimensionality of the search space
- After a certain runtime
- Specify a target threshold (of the objective or acquisition function)
- Based on stagnation (of the objective or acquisition function)



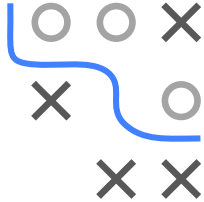
# ROBUSTNESS

A good BO implementation should be **robust**:



- Handle crashing SM
  - Especially a GP can result in errors during training (Kernel matrix not invertible, training points too close to each other)
  - Use a fallback SM (e.g., Random Forest) or catch errors and propose a new candidate uniformly at random
- Automatically detect input types (numerical, categorical; hierarchical) and choose an appropriate SM
- Have sensible defaults that work well for most scenarios

# CHOOSING THE RIGHT COMPONENTS



- BO is modular: SM, acquisition function, acquisition function optimizer
- Different choices induce different overhead, e.g., GP vs. RF, entropy based acquisition functions vs. cheap to compute ones like EI, thorough acquisition function optimization with a large budget can be expensive
- Choosing the right components usually depends on the concrete application at hand and how expensive the evaluation of the black box itself is