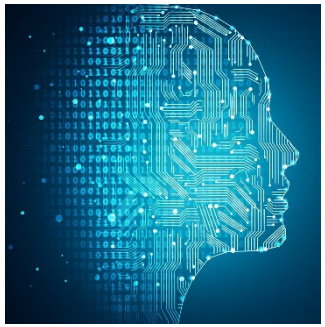
Machine Learning

Hyperparameter Optimization

Kevin Moon (kevin.moon@usu.edu)

STAT/CS 5810/6655



Hyperparameter characteristics

- Neural networks and other ML methods can have many hyperparameters
  - Architecture, learning rate, activation functions, regularization types and parameters, etc.
- These hyperparameters have different characteristics
  - Continuous
  - Discrete
  - Binary
  - Categorical
  - Bounded or unbounded
- Some are *conditional*
  - Depend on the configuration

HPO Process

1. Select the loss function and performance metrics
2. Select hyperparameters to tune and determine the (hyperparameter) optimization technique
3. Start the optimization process with a large search space determined by manual testing or domain knowledge
4. Narrow the search space based on well-performing configurations or explore new search spaces if necessary
5. Return the best-performing configuration

HPO Challenges

1. Optimization criteria is typically nonconvex and nondifferentiable
   - Hard to choose hyperparameters using GD, for example
2. Discrete and categorical hyperparameters are difficult to optimize numerically
3. The ML model is trained for each hyperparameter configuration, which can be computationally expensive
   - Data sampling (training on smaller sample sizes) is often used to speed things up

“Grad Student Descent”

- Basically, Nielsen’s guidelines
- Also called “trial and error” or “babysitting”
- The student manually tunes all hyperparameters until running out of time
- Requires a sufficient amount of experience and time
- Generally infeasible for complex problems
  - Especially those involving non-linear hyperparameter interactions
- This motivates automated methods

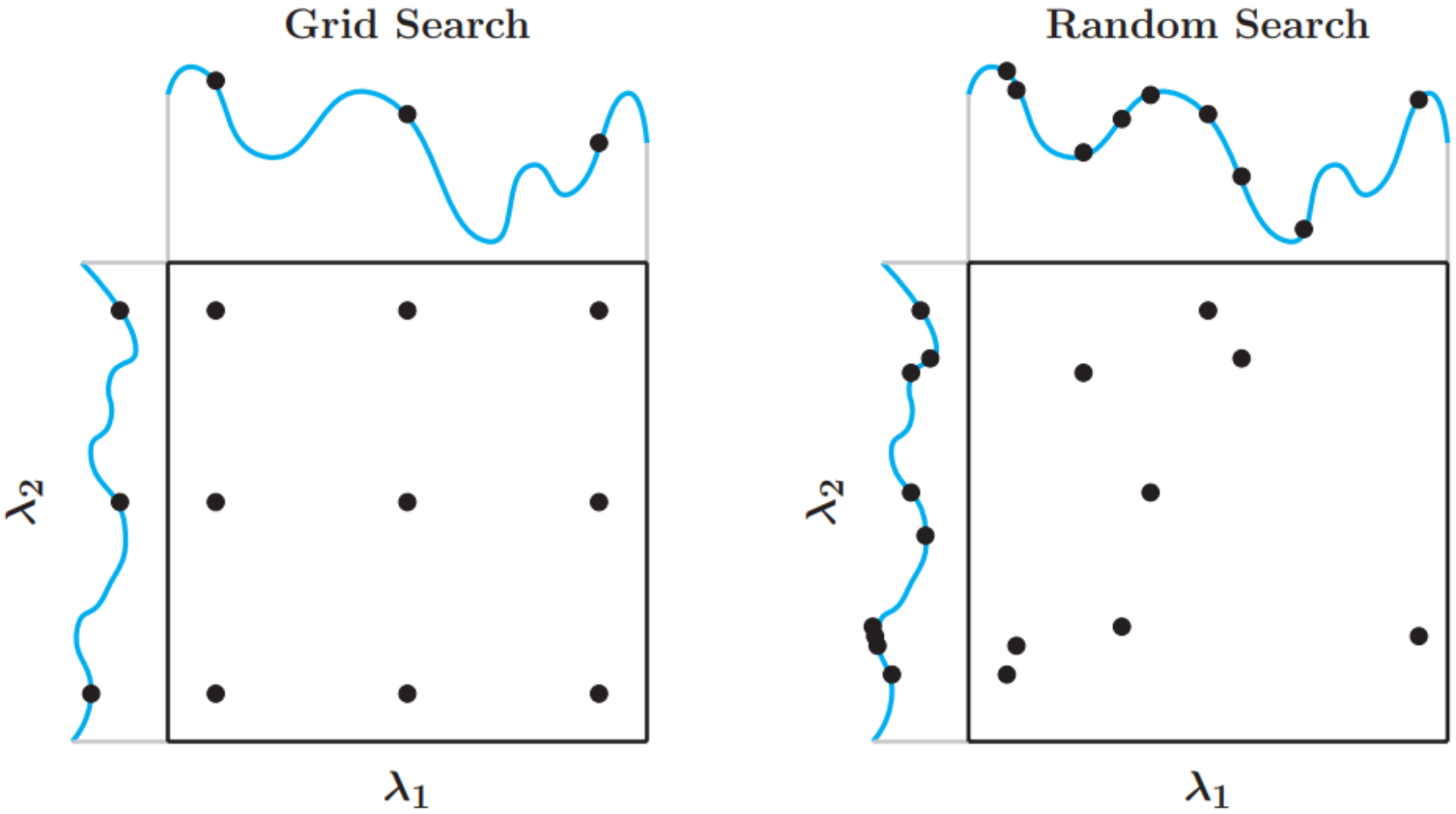
Grid Search (GS)

- One of the most commonly used methods
- Exhaustively search a grid of hyperparameter combos
- Can narrow the search space after an initial search to get better results
- **Advantages**: Simple to implement and parallelizable
- **Disadvantages**: Time-consuming and inefficient for all but categorical hyperparameters
- Time complexity: $O\left(n^k\right)$
  - $n = \#$ hyperparameter values, $k = \#$ hyperparameters

Random Search (RS)

- Similar to GS
  - Instead of testing all values, RS randomly selects a pre-defined number of samples within the bounds and tests those
- RS can explore larger search spaces than GS
- **Advantages**: Easily parallelizable and more efficient than GS
- **Disadvantages**: doesn't narrow in on good regions and inefficient with conditional hyperparameters
- Time complexity: $O(n)$

GS vs RS


Grid Search
λ_2
λ_1
Random Search
λ_2
λ_1

Bayesian Optimization (BO)

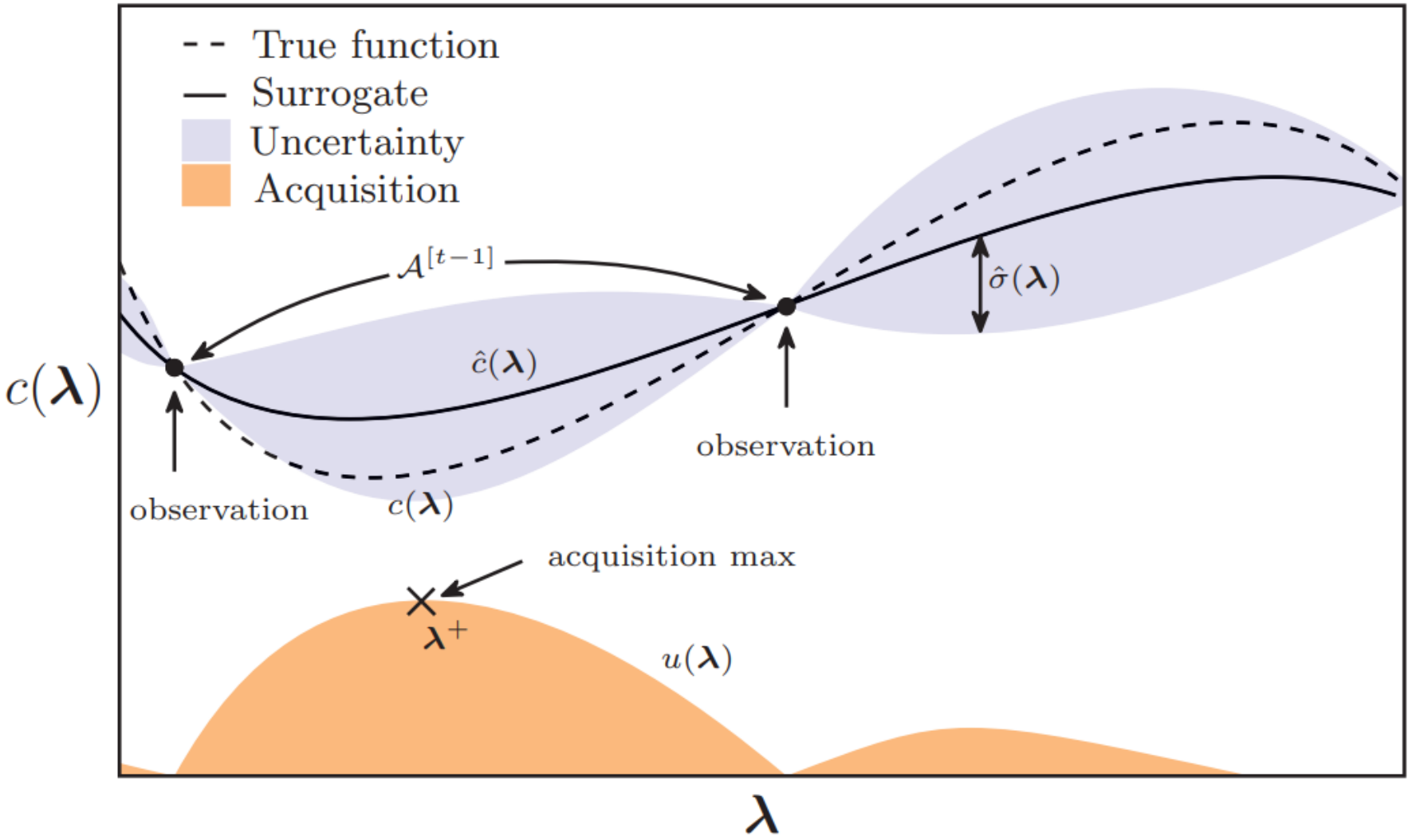
- An iterative approach that determines future evaluation points based on previously obtained results
- Based on Bayesian probability
- Two components: surrogate model and acquisition function
- Surrogate model aims to fit all currently-observed points into the loss function
- Acquisition function determines next step by balancing exploration vs exploitation
  - Exploration = look for promising areas
  - Exploitation = sample within known promising areas

Bayesian Optimization (BO)

Basic procedure:

1. Build a probabilistic surrogate model of the loss function
2. Determine potential hyperparameter values based on the surrogate model and acquisition function
3. Apply these values to the actual loss function
4. Update the surrogate model with the new results
5. Repeat steps 2-4 until maximum number of iterations

Bayesian Optimization (BO)



Bayesian Optimization (BO)

- **Advantages**: more efficient than GS and RS as they require fewer tested values and the surrogate model is often much cheaper than the entire objective function
- **Disadvantages**: not as parallelizable as it's sequential and may converge to a local min
- Common surrogate models: Gaussian process (GP), random forest (RF), and the tree Parzen estimator (TPE)
  - BO-RF also called SMAC
  - BO-RF can be used to optimize all variable types and is faster than BO-GP

Multi-fidelity optimization algorithms

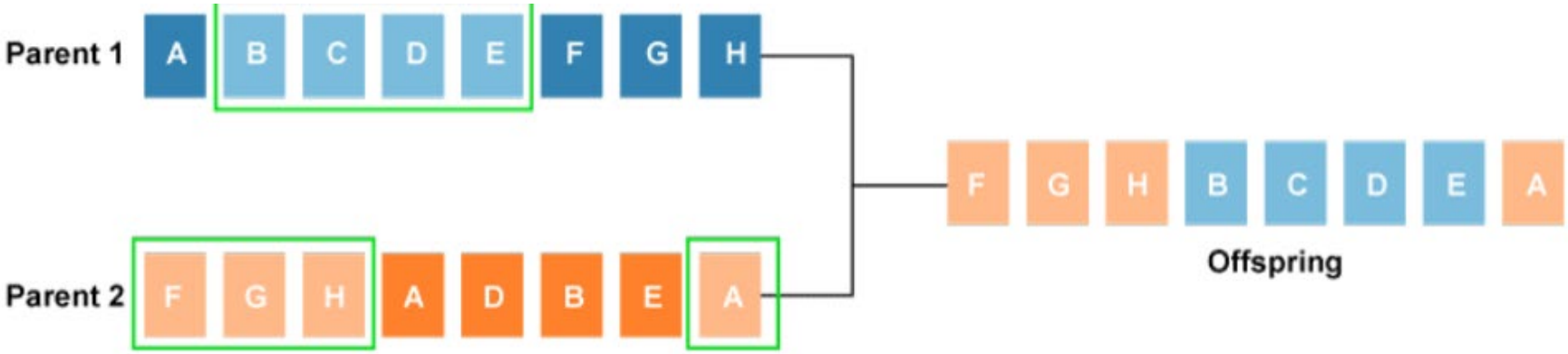
- Basic idea: do initial testing on a subset of the data
  - Use the whole data only on good configurations
- Successive halving evaluates initial hyperparameter configurations with N/n training points
  - $N =$ total training points, $n =$ # configurations
  - Throw away half of the configurations and reevaluate on $2N/n$ training points
  - Repeat until final combination
  - More efficient than RS, but has some tradeoffs
- Hyperband is an improvement over successive halving
- BOHB is a combination of Bayesian optimization and hyperband that works well
  - $O(n \log n)$ and efficient but requires small subsets to be representative

Genetic algorithm (GA)

- Iterative approach where good hyperparameters survive to the next generation
  - Mutations are introduced
- Do the following for a fixed number of iterations:
  1. Train configuration (simultaneously or one by one) and calculate their training cost
  2. Calculate the "fitness" of each configuration based on the cost. The higher the fitness, the more likely it "reproduces"
  3. Pick 2 configurations based on a probability based on their fitness and crossover the "genes" (hyperparameters) of the 2 configurations to create a "child" configuration
  4. Mutate the genes of the child
  5. Repeat steps 3-4 to create a set of "children" configurations and then repeat the process with the new configurations

Genetic Algorithms (GA)

- **Advantages**:
  - Not sensitive to initialization
  - Can give good configurations
- **Disadvantages**:
  - Introduces other hyperparameters (population size, etc.)
  - Time complexity of $O(n^2)$ and thus may be inefficient
  - Harder to parallelize than particle swarm optimization (next)



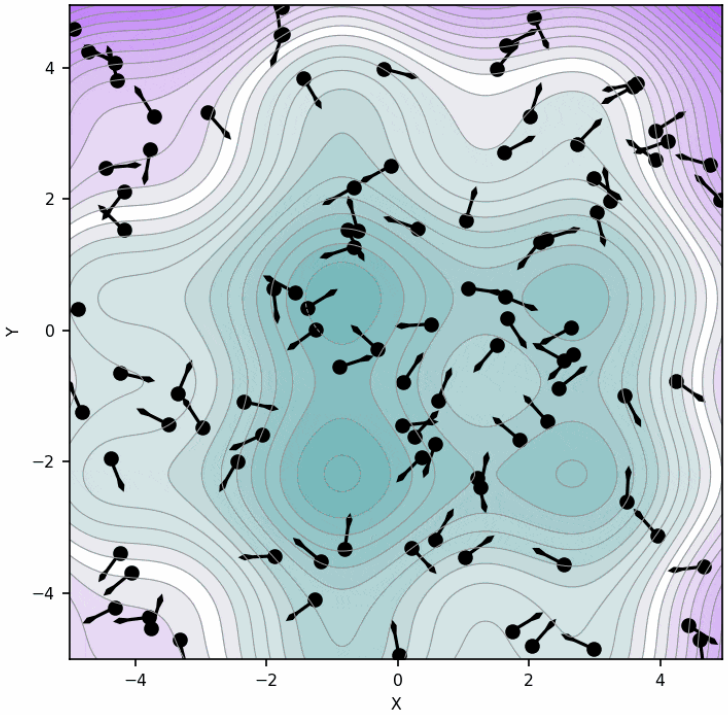
https://www.boardinfinity.com/blog/genetic-algorithm-in-machine-learning/

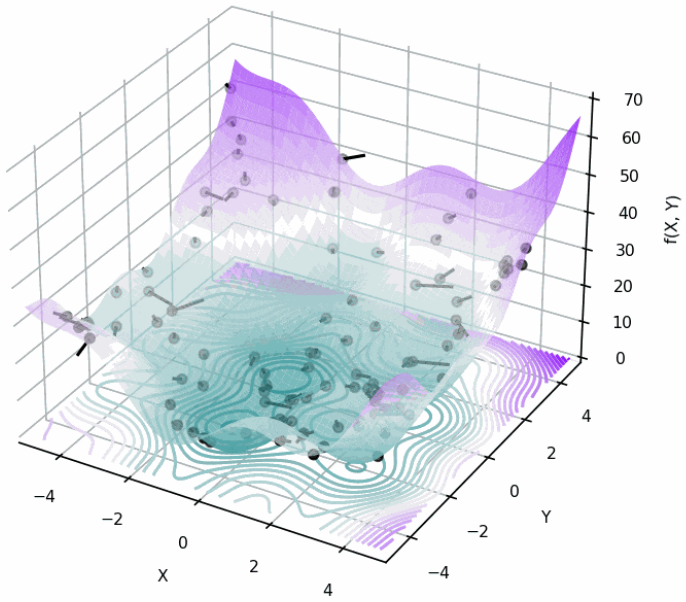
Particle Swarm Optimization (PSO)

- A group of particles semi-randomly search the hyperparameter space
- Optimal solution identified by cooperation and information sharing between particles
- **Advantages**: easier to implement than GA and often converges faster. Parallelizable
- **Disadvantages**: sensitive to initialization
- Time complexity: $O(n \log n)$

Particle Swarm Optimization (PSO)

[1/100] w:0.800 - c_1:2.000 - c_2:2.000





https://towardsdatascience.com/particle-swarm-optimization-visually-explained-46289eeb2e14

What should you use?

- You should probably at least use random search and not a grid search
- PSO seems to be a common recommendation when you have a lot of hyperparameters and computational power
- Hyperband, BOHB, and BO-RF/SMAC can be good as well

- Worth experimenting with a few and seeing which one you like

Packages

- Ray Tune contains many of these algorithms and is generally recommended
- Optuna also works well (included within Ray Tune)
- Optunity includes some of the more advanced methods like GA and PSO
- Sklearn has GS and RS built in (best used with Skorch with PyTorch)
- Other packages out there

Further reading

- Yang and Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice” https://doi.org/10.1016/j.neucom.2020.07.061
- Bischl et al. “Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges” https://arxiv.org/pdf/2107.05847.pdf
- Initialization: https://pytorch.org/docs/stable/nn.init.html