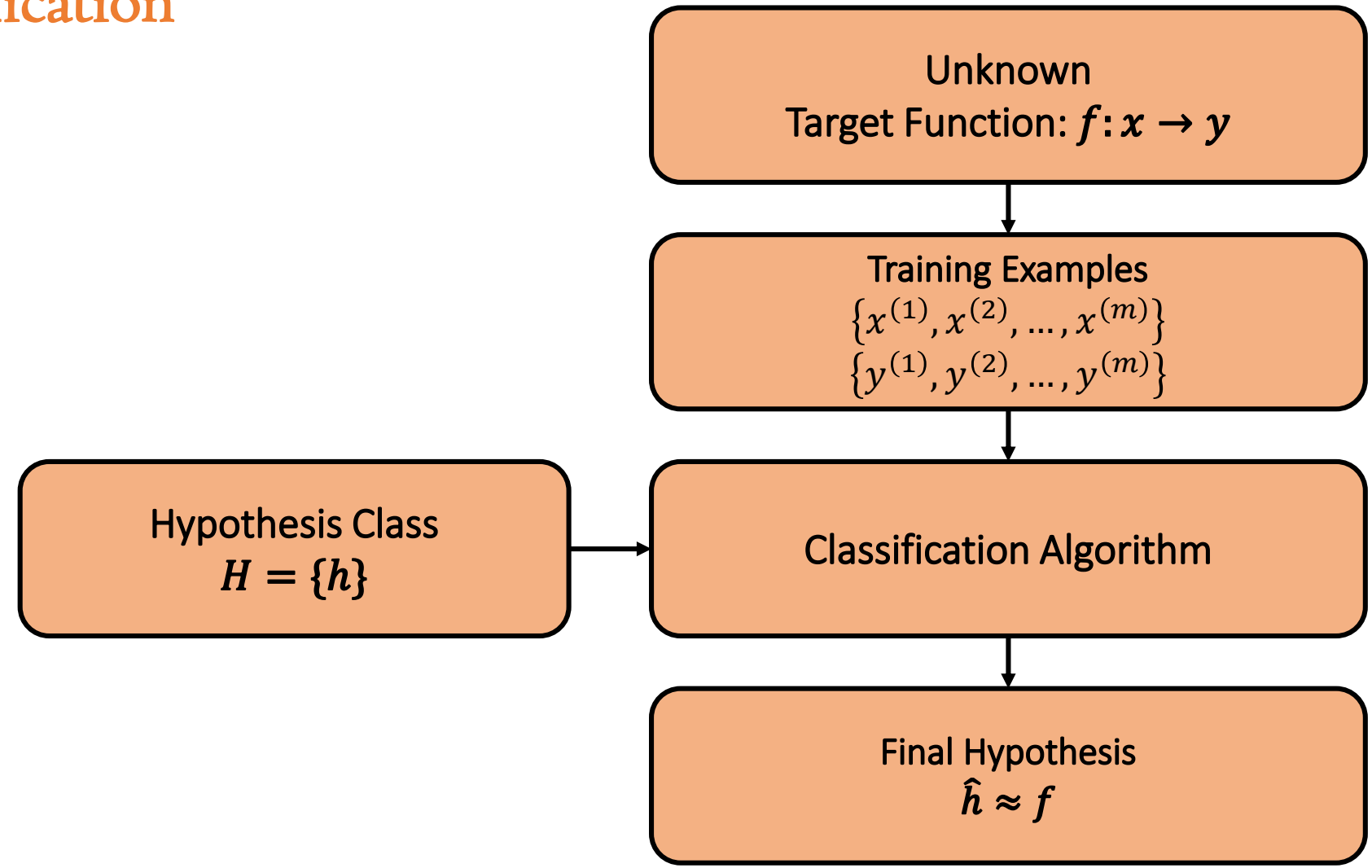
Machine Learning in Healthcare

#C19 Neural Networks III: Hyperparameters tuning

Technion-IIT, Haifa, Israel

Assist. Prof. Joachim Behar
Biomedical Engineering Faculty
Technion-IIT

Classification



Hyperparameters tuning

Hyperparameters versus parameters

- **Parameters:** are the parameters that define the model and that we want to learn through the training process with gradient descent.
  - $\forall\, l \in [1{:}\,L], W^{[l]}, b^{[l]}$
- **Hyperparameters:** other parameters that are set before the learning process is started but that have an influence on the classifier performances.
  - E.g. α, L
  - Need to search for these hyperparameters values to ensure a good model architecture.

Hyperparameters

- With SVM we have a few typical hyperparameters C and γ.
- However, ***lots*** of hyperparameters in deep learning models…

Symbol	
α	Learning rate.
β	Momentum
p	Mini batch size
K	Number of iterations for gradient descent.
$n_h^{[l]}$	Number of hidden units of the lth layer.
L	Number of layers in a neural network.
$g^{[l]}$	Activation function for layer l.
k	Learning rate decay
	Features scaling method
	Other model specific hyperparameters (e.g. convolution kernel width in CNN.)

Hyperparameters

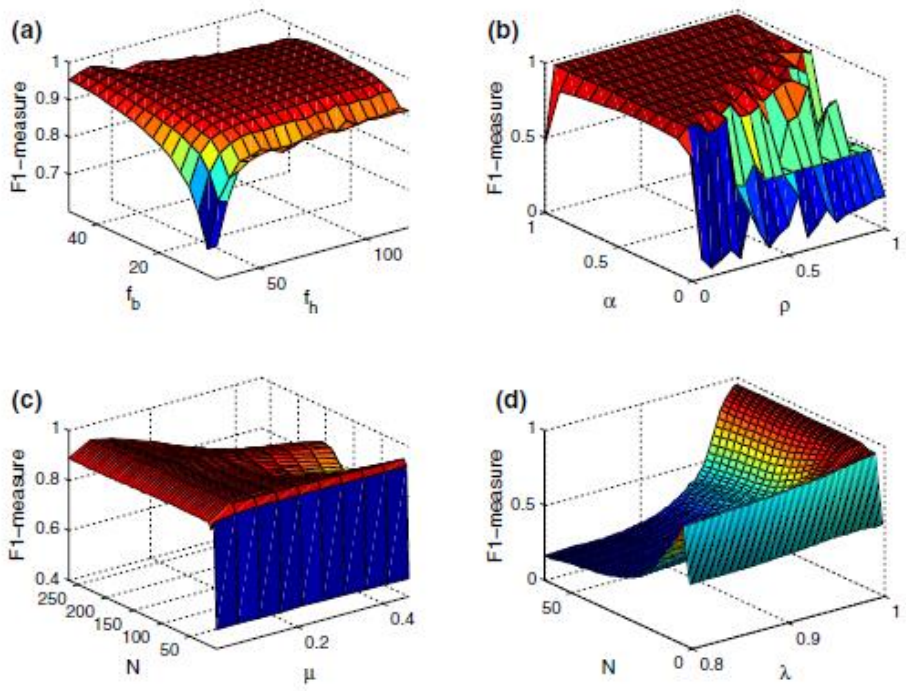
- We will denote θ the set of hyperparameters we want to optimize for.
- Our goal is to find the values of θ that gives the best performance on the validation set.
- How do we find the best configuration of these hyperparameters in such a high dimensional search space?
  - Baby sitting,
  - Grid search,
  - Random search,
  - Bayesian optimization.

Babysitting

- Also known as the “Grad Student Descent”!
  - Iterate sequentially,
  - Manual.
- Usually keep doing that until you are running out of time for your assignment!
- This is meaningful in an initial stage of development to get a feel of what hyperparameters are particularly important for example. But then you need to quickly move to some more clever search algorithms.

Grid search

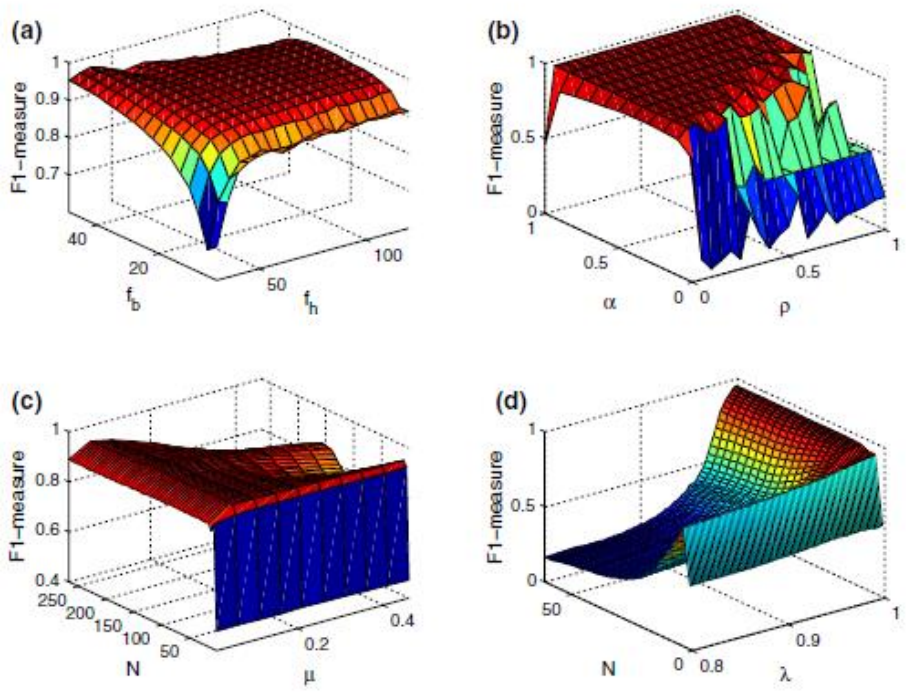
- The “just try everything” approach.
- Grid-search steps:
  - Define the p hyper-parameters,
  - For each one, define the range of possible values,
  - Search all possible configurations and report the performance.



Behar, Joachim, et al. "A comparison of single channel fetal ECG extraction methods." Annals of biomedical engineering 42.6 (2014): 1340-1353.

Grid search

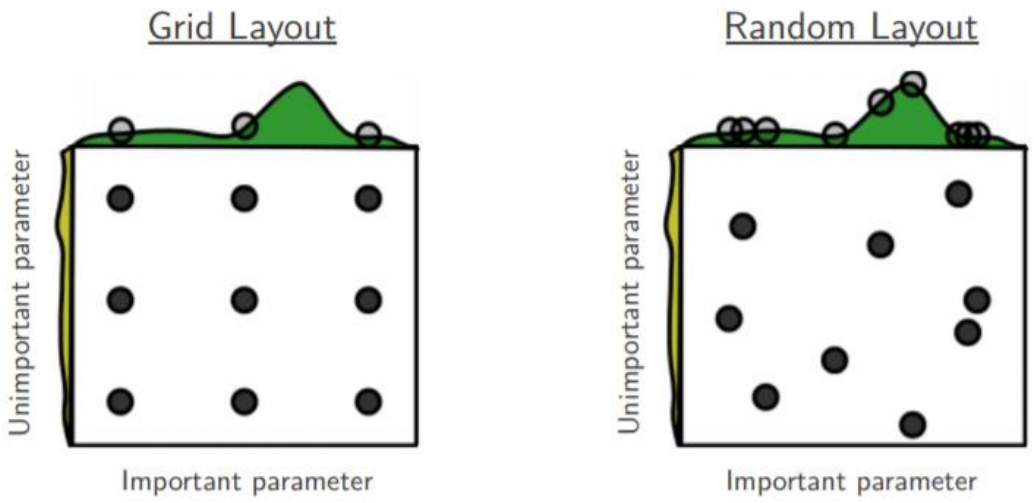
- Pluses
  - Will find a good combination of hyperparameters.
  - Can perform the search in parallel.
- Minuses
  - Does not take into account the computation history.
  - Search space increases exponentially with the number of dimension: *curse of dimensionality*.
    - E.g. p hyperparameters, search of 5 values in a given range will require 5^p iterations.



Behar, Joachim, et al. "A comparison of single channel fetal ECG extraction methods." Annals of biomedical engineering 42.6 (2014): 1340-1353.

Random search

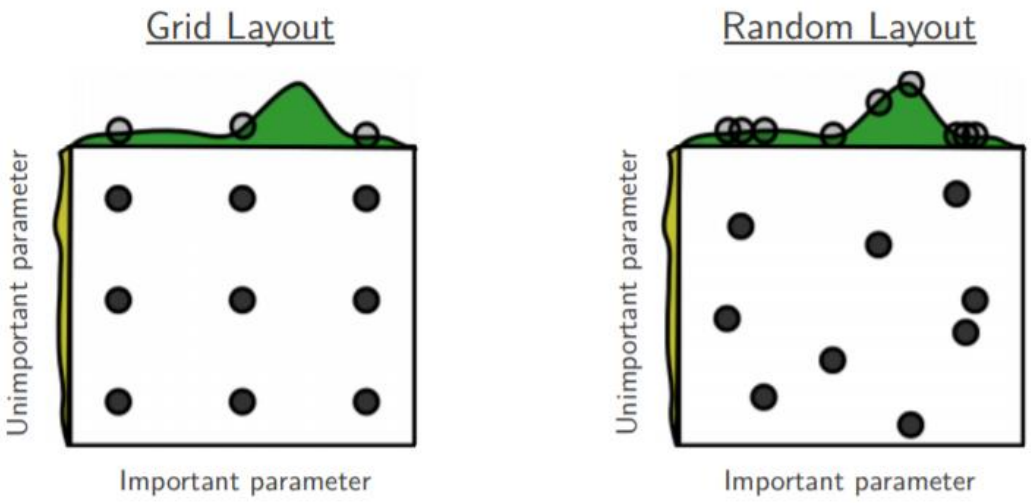
- Random-search steps:
  - Define the p hyper-parameters,
  - For each one, define the range of possible values,
  - ~~Search all possible configurations~~ sample randomly from the hyperparameters space and report the performance.



Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." Journal of Machine Learning Research 13.Feb (2012): 281-305.

Random search

- Pluses
  - Explore the hyperparameters space more widely in a given number of iterations than grid search. This will help find a good configuration with fewer iteration.
  - Enables the inclusion of prior knowledge by specifying the distribution from which you are sampling from.
  - Can perform the search in parallel.
- Minuses
  - Still does not take into account the computation history.



Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." Journal of Machine Learning Research 13.Feb (2012): 281-305.

Hyperparameters versus parameters

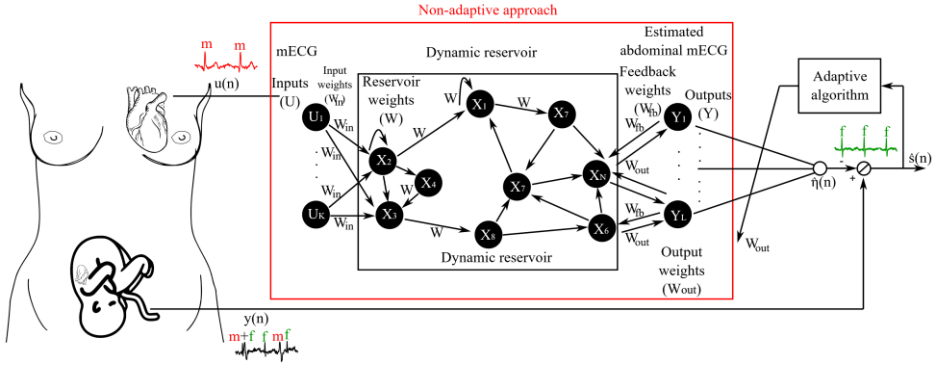
- Random search:

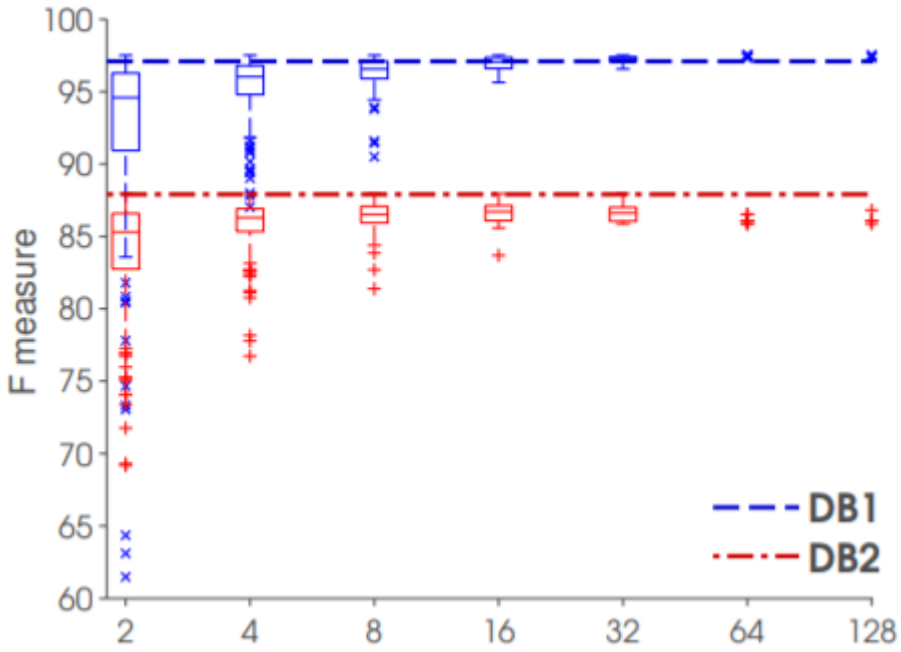
Table 1: Parameters search range and optimal parameters found for the preprocessing step and the ESN. GS: grid search. RS: random search. $\mathcal{U}$: uniform distribution.

Parameter	Search: GS (step size)	Search: RS	Optimal parameters: GS	Optimal parameters: RS
Low pass filter cut-off, f_b	[1, 49] (3)	$\mathcal{U} \sim [1, 50]$	20	26
High pass filter cut-off, f_h	[50, 120], (5)	$\mathcal{U} \sim [50, 120]$	95	87
Leakage, α	[0, 1] (0.1)	$\mathcal{U} \sim [0, 1]$	0.4	0.974
Spectral radius of W, ρ	[0, 1] (0.1)	$\mathcal{U} \sim [0, 1]$	0.4	0.821
Units in the reservoir, M	[10, 250] (20)	$\mathcal{U} \sim [10, 250]$	90	135
Scaling of W_{in}, γ	1	$\mathcal{U} \sim [0, 1]$	1	0.622
Seed value, s^1	–	$\mathcal{U} \sim [0, 10000]$	–	1588

Table 2: Performance comparison of the optimal parameters obtained by grid search (GS), random search (RS, best across 32 iterations ± 1 standard error), and template subtraction (TS).

Method — Statistics	DB_1 ESN-GS	DB_1 ESN-RS	DB_1 TS	DB_2 ESN-GS	DB_2 ESN-RS	DB_2 TS
Se	97.1	97.3 ± 0.29	90.3	87.6	87.6 ± 0.73	86.4
PPV	97.3	97.5 ± 0.28	90.0	86.5	85.5 ± 0.53	85.2
F_1	97.2	97.4 ± 0.27	90.1	87.9	86.5 ± 0.62	85.8





Behar, Joachim, et al. "An echo state neural network for foetal ECG extraction optimised by random search." Proc. Adv. Neural Inf. Process. Syst (2013).

Bayesian optimization

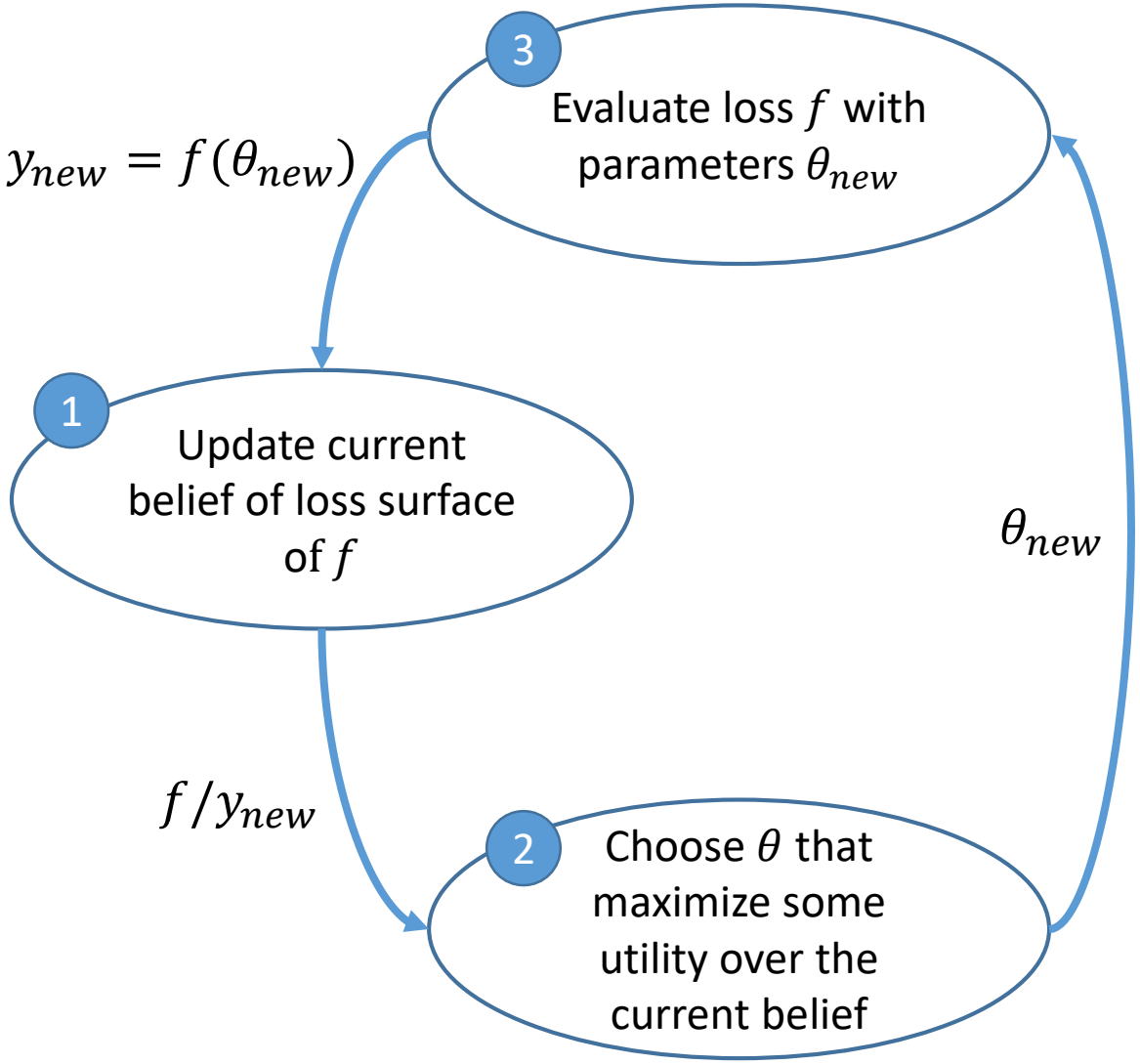
- Bayesian optimization: probability model for the loss function and sequentially move in that region of better performance.
  - It is a type of sequential model-based optimization (SMBO)
- We define the loss function $f(\theta)$ given a dataset D.
  - We seek $\theta^* = argmin_{\theta \in \chi}(f(\theta))$
  - We can evaluate f for any θ but we do not have an easy functional form for it or gradients.
  - We seek to use Gaussian Processes (GP) to estimate f based on the points $\theta_{1:n}$ it was explicitly evaluated for and return the marginal means and variances.
- Bayesian optimization steps:
  - Use previous observations of the loss f to evaluate it,
  - Find the next (optimal) point to sample f for.

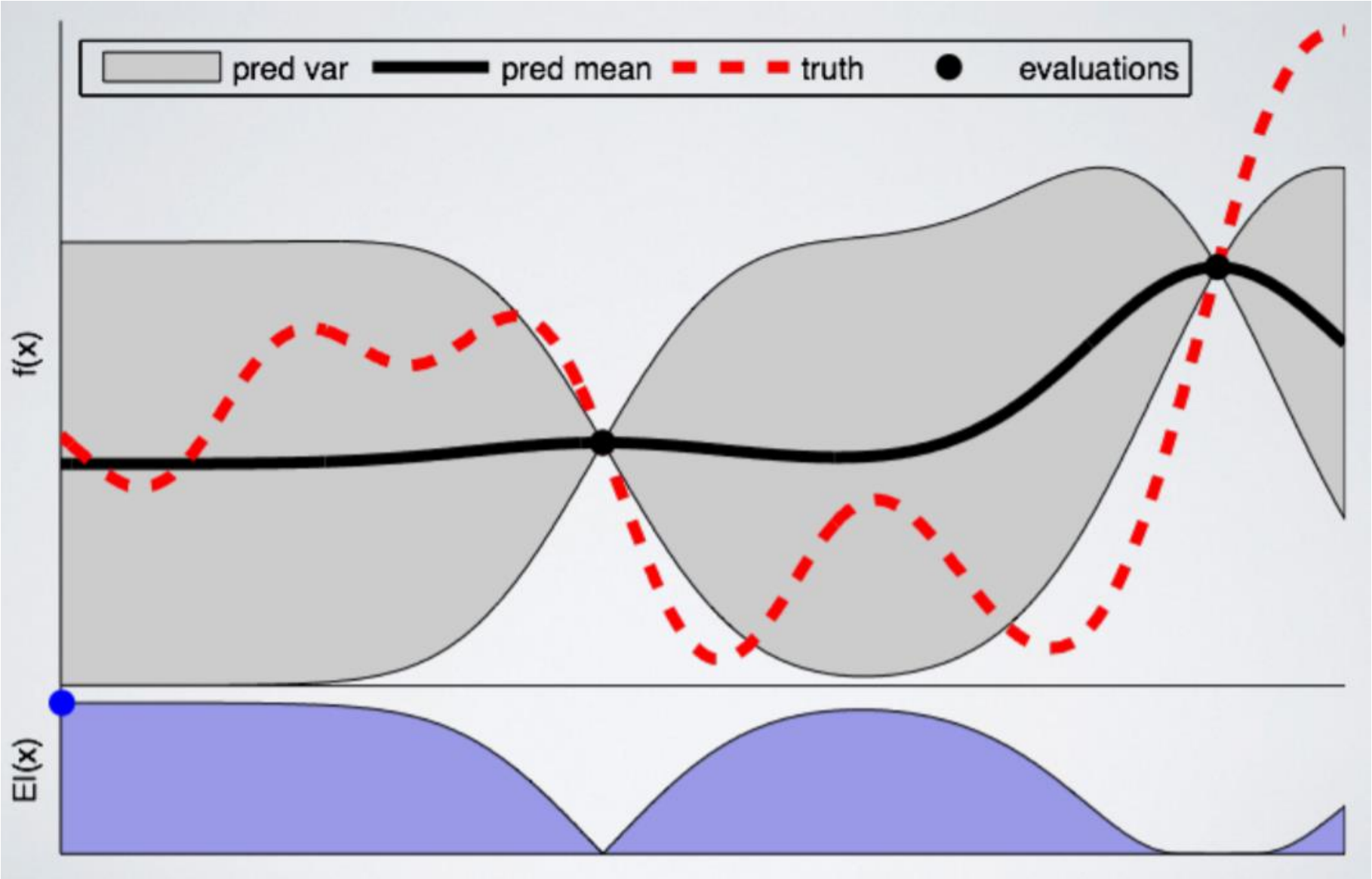
Bayesian optimization

- In Bayesian optimization, we are building a probabilistic model for the performance metric f. Implicitly this is introducing computational overhead.
- So when does it makes sense to invest in this approach versus grid/random search?
  - The number of hyperparameters is very high.
  - It is computationally very expensive to evaluate $f(\theta)$ for a single point θ (and thus implicitly the computational overhead is not such a high cost.)

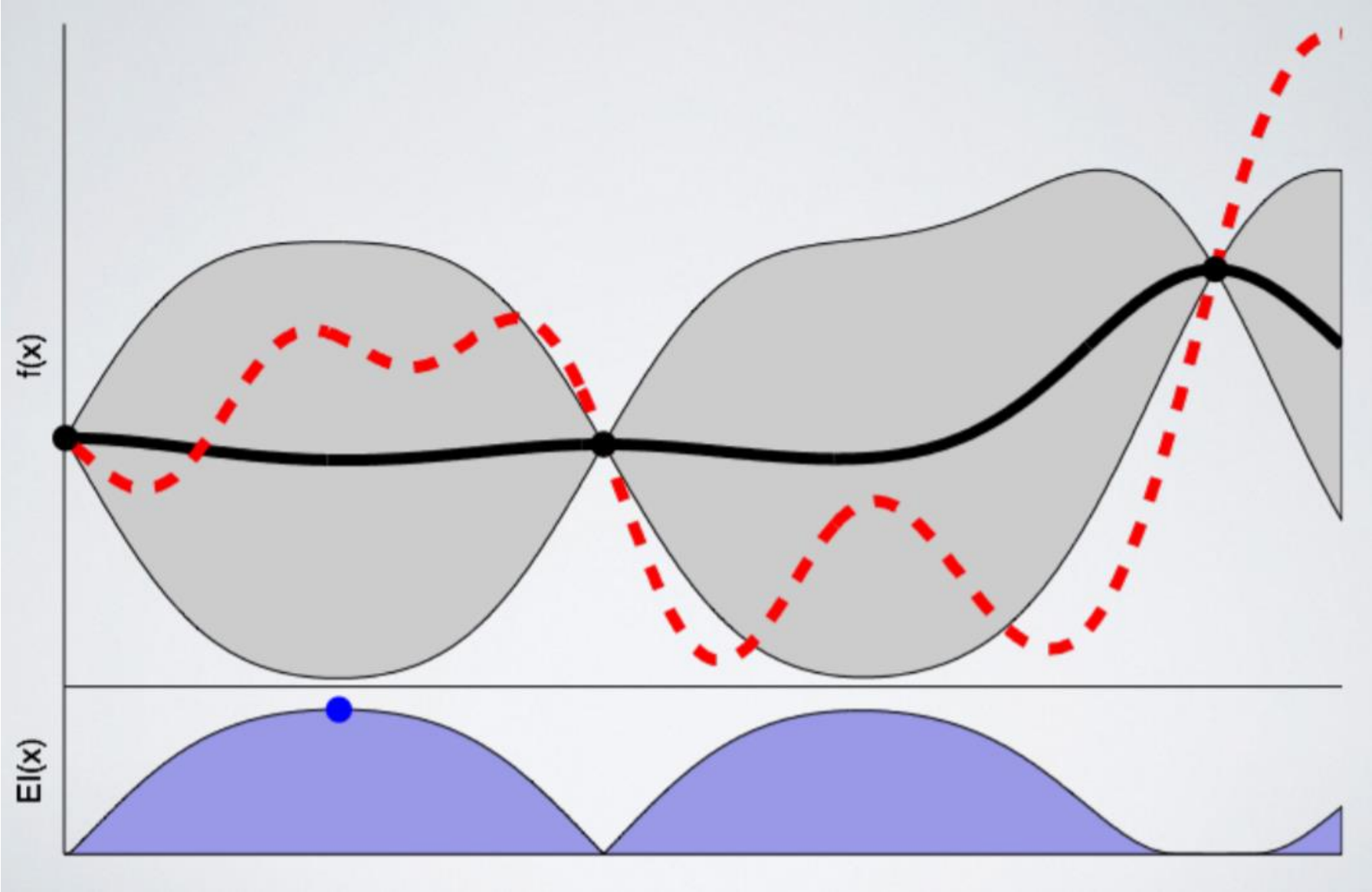
Bayesian optimization

- Steps:
  1. Using the points that were evaluated $\theta_{1:n}$, compute the posterior expectation of the loss f.
  2. Choose new point θ_{new} to sample by maximizing some utility of the expectation of f. The utility specifies which regions of the domain of f are optimal to sample from.
  3. Evaluate f at a new point θ_{new}.
- Gaussian processes used to represent the loss function f and evaluate its utility.

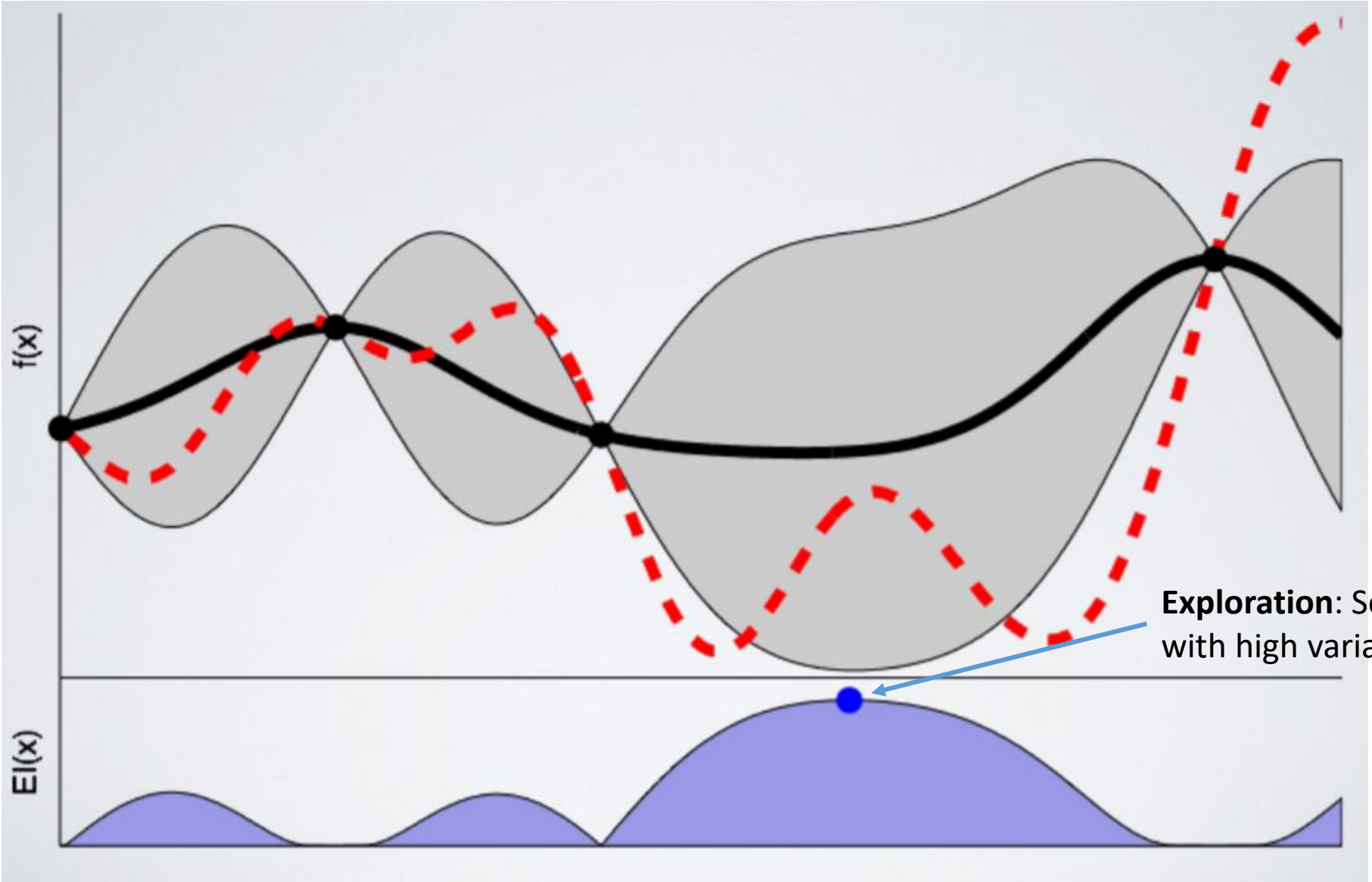




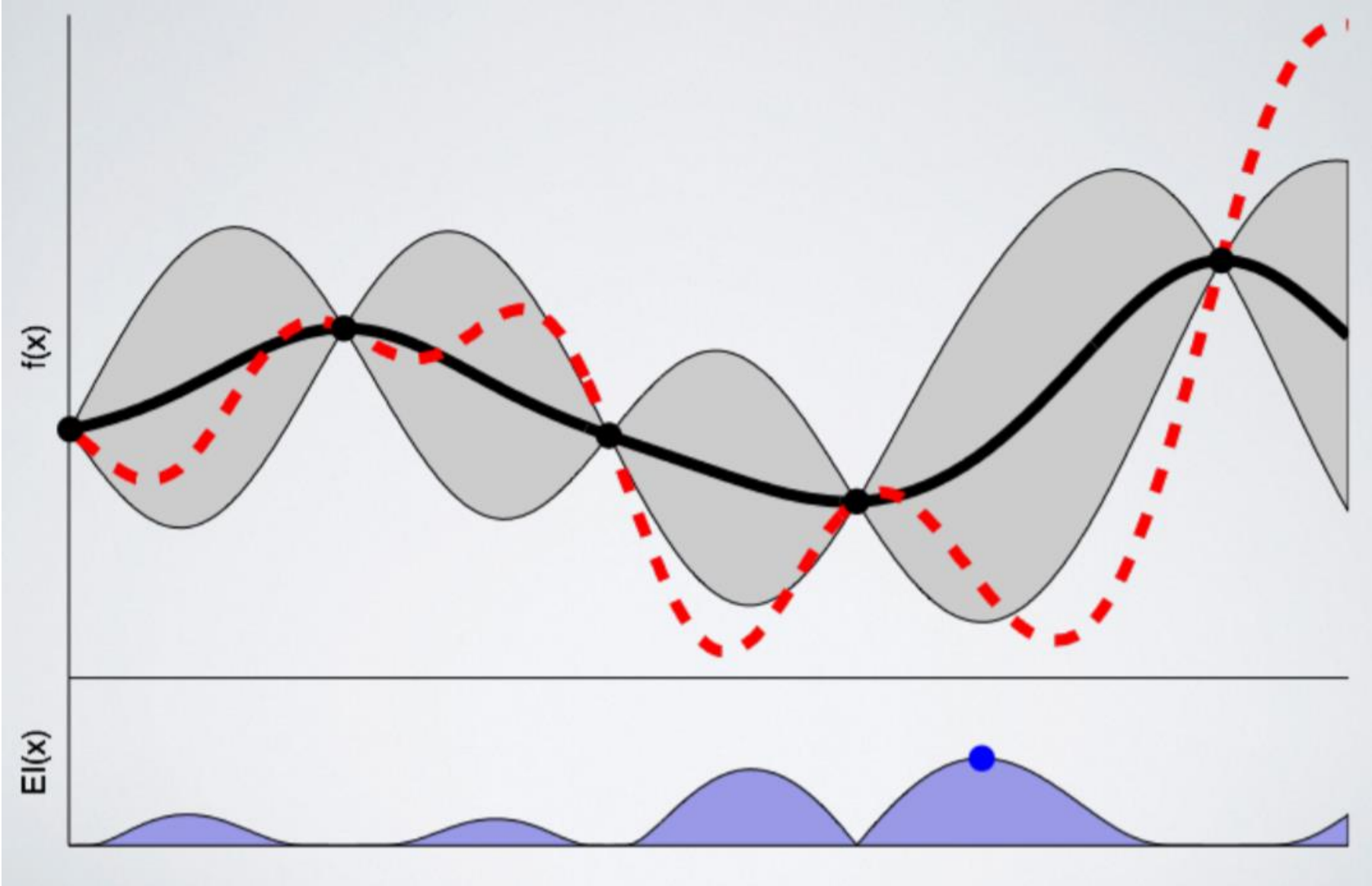
(Example from Ryan P. Adams)



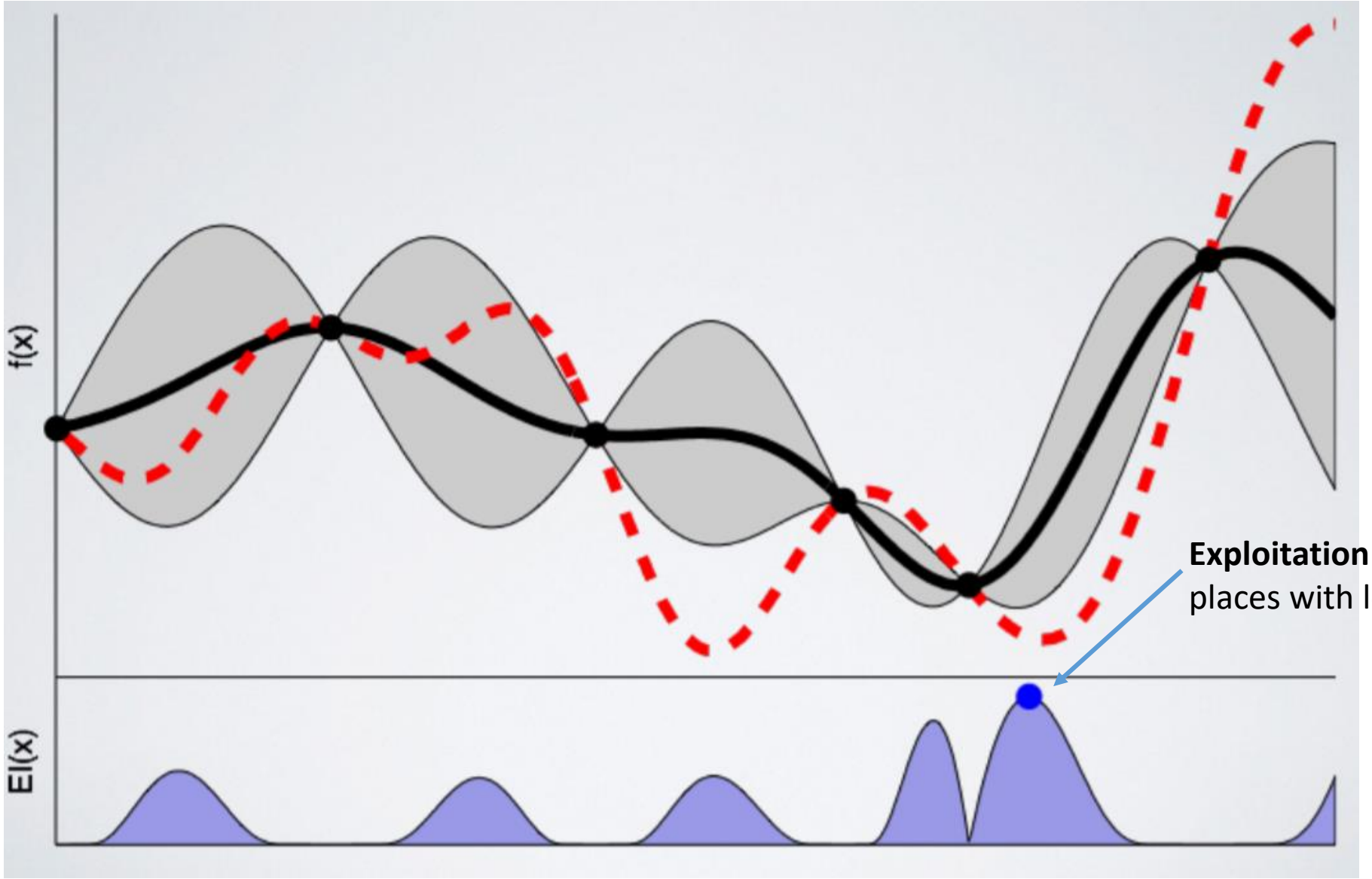
(Example from Ryan P. Adams)



(Example from Ryan P. Adams)

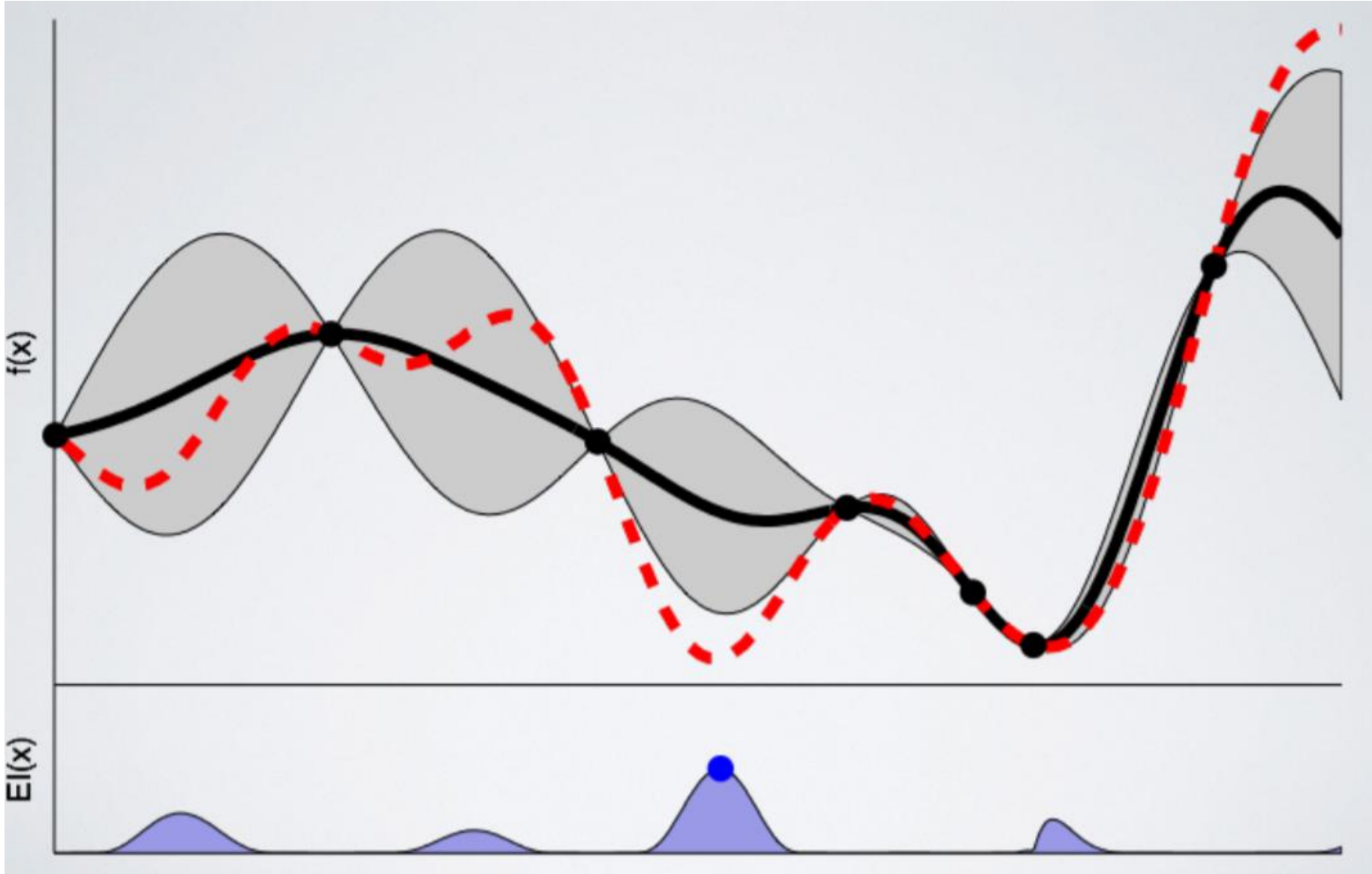


(Example from Ryan P. Adams)

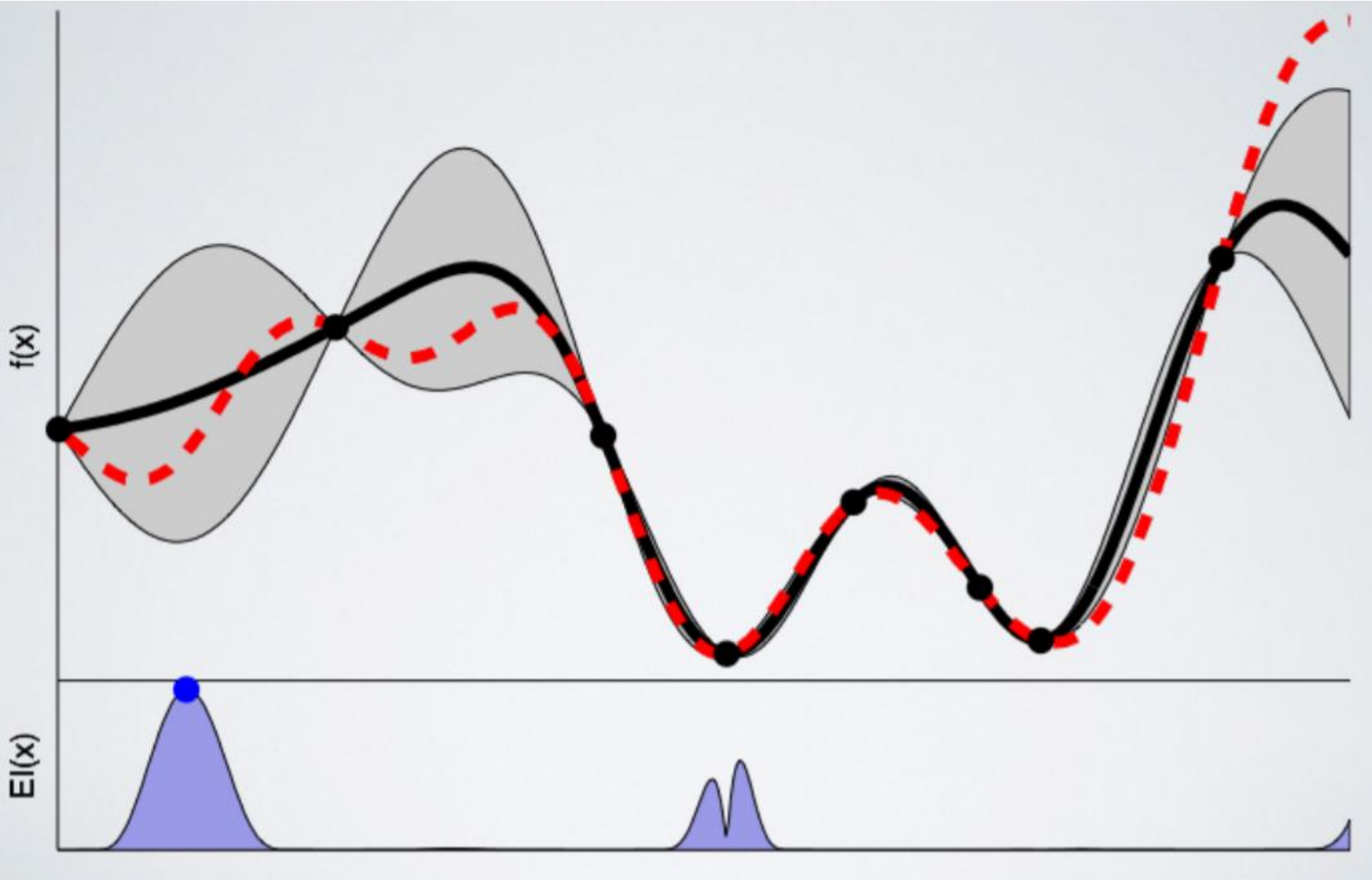


Exploitation: Seek places with low mean.

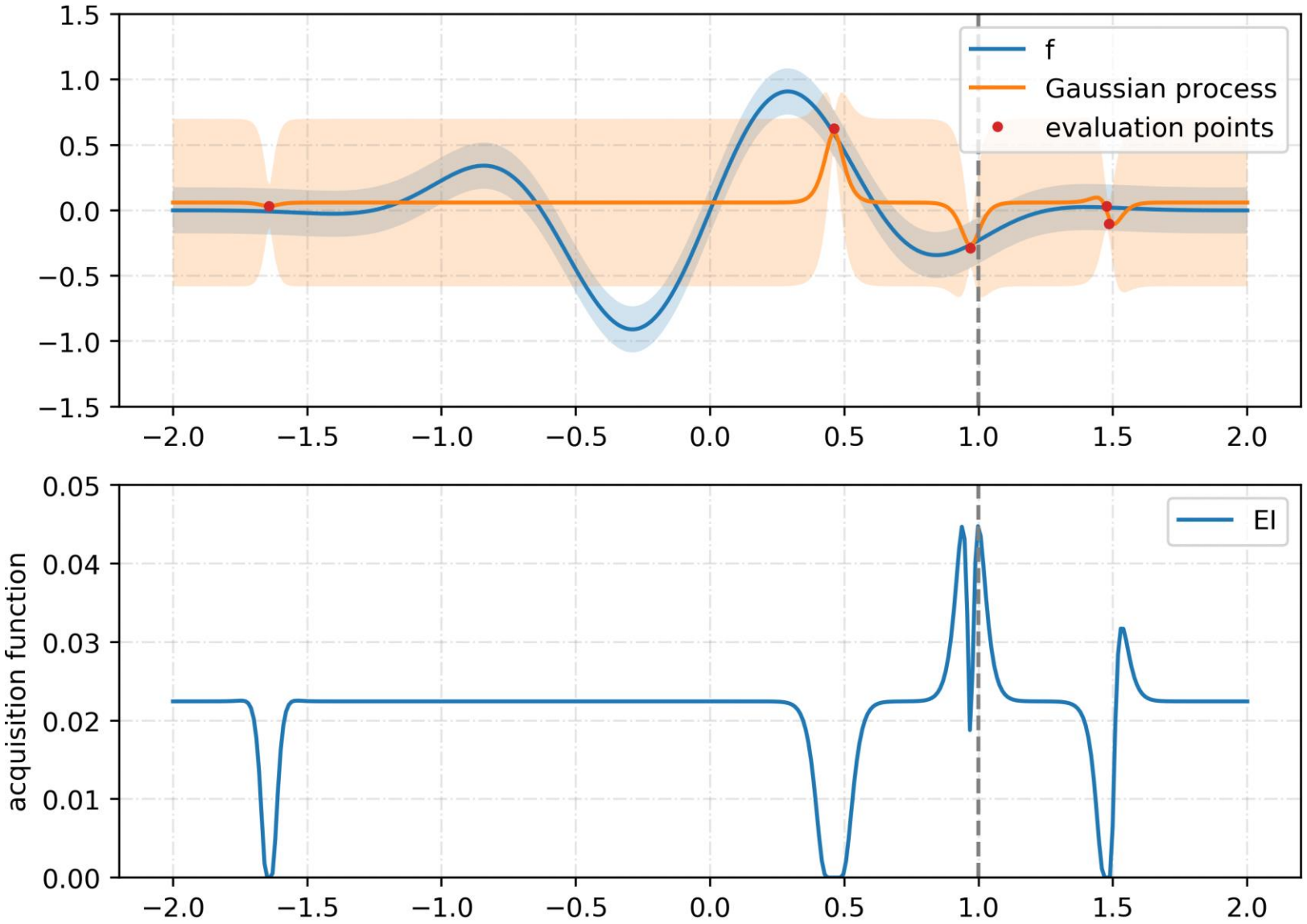
(Example from Ryan P. Adams)



(Example from Ryan P. Adams)



(Example from Ryan P. Adams)



https://medium.com/criteo-labs/hyper-parameter-optimization-algorithms-2fe447525903

Recommendations

- Development phase: baby sitting.
- Optimization phase:
  - Random search: If you have the computing resources train many models at the same time and look for the best (on your validation set). This will give you a good baseline of hyperparameters tuning.
  - Use Bayesian optimization or such *Sequential Model Based Optimization.*

Take home

- Hyperparameters versus parameters.
- Hyperparameters tuning
  - Lots of hyperparameters in NN
  - Different methods to search the hyperparameters space:
    - Babysitting,
    - Grid-search,
    - Random search,
    - Bayesian optimization.
- These exists other optimization methods that we did not covered (e.g. Evolutionary optimization).

References

[1] Andrew Ng, Coursera, Neural Networks and Deep Learning. Coursera.
[2] Initializing neural network: https://www.deeplearning.ai/ai-notes/initialization/
[3] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747(2016).
[4] Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." Journal of Machine Learning Research 13.Feb (2012): 281-305.
[5] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.
[6] Thomas Huijskens - Bayesian optimisation with scikit-learn
https://www.youtube.com/watch?v=jtRPxRnOXnk
https://thuijskens.github.io/2016/12/29/bayesian-optimisation/
[7] Adams, Ryan P. "A tutorial on Bayesian optimization for machine learning." Harvard University (2014). https://www.cs.toronto.edu/~rgrosse/courses/csc411_f18/tutorials/tut8_adams_slides.pdf

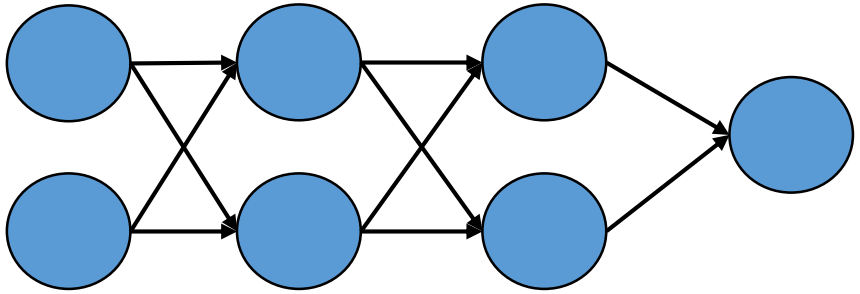
Agenda

- Parameters and hyperparameters choices.
- Vanishing and exploding gradient.

Vanishing and exploding gradient

Vanishing and exploding gradient

- What is it about?
  - Vanishing: very small gradients develops resulting in parts of the network not to update itself so essentially stopping to learn.
  - Exploding: very large gradients develops resulting in large updates to the network parameters making it unstable.
- When does it happen?
  - Typical problem when training deep network.
- Intuition:
  - Assuming linear activation functions.
  - Forward propagation: $\hat{y} = W^{[L]}W^{[L-1]} \dots W^{[1]}X$
  - Assuming $\forall l \in [\![1, L]\!], W^{[l]} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ then $\hat{y} \propto 2^L$
- Thus if $L \gg 1$ and $W^{[l]} > I$ then the value will tend to "explode".
- If $L \gg 1$ and $W^{[l]} < I$ then the value will tend to "vanish".



Vanishing and exploding gradient

- Similar idea with backpropagation and the gradient:
  - Large gradients will lead to large updates in the weights, these updates becomes larger and larger as we get closer to the earlier layers. This leads the NN to become very unstable and “explode”.
  - The gradient becomes very small as we approach the first layers and this will lead to small update to their weights which means that these layers do not update/learn anymore. The gradient “vanished”.
- How to address this issue?
  - Re-design the network model,
  - Use Long Short-Term Memory networks (for RNN),
  - Use Gradient clipping,
  - Use weight regularization,
  - ResNet
  - …