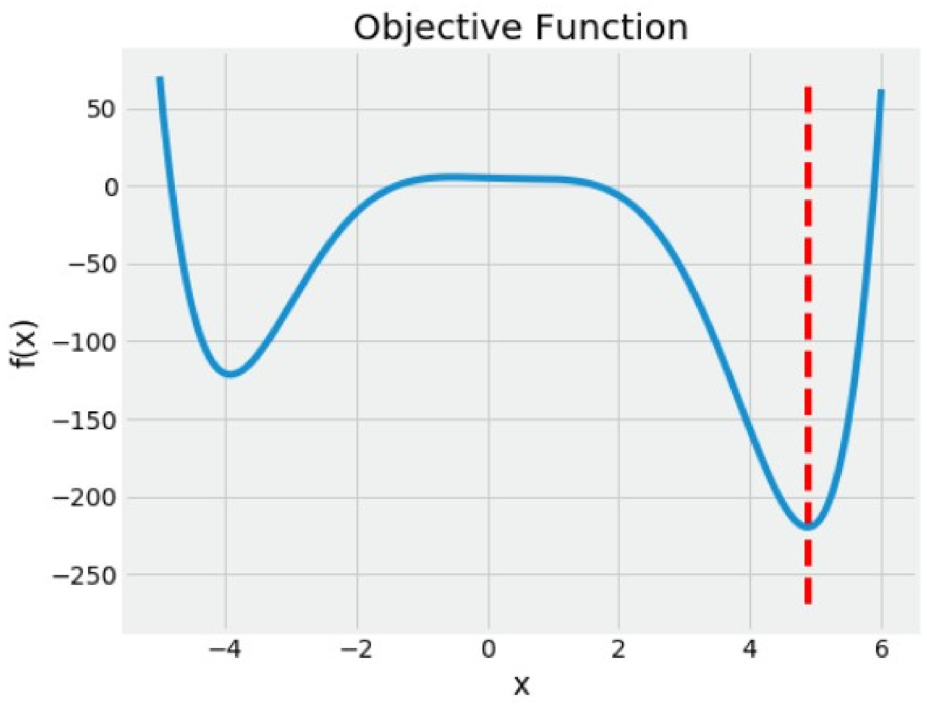
Machine learning
Lecture 8

```
Minimum of -219.8012 occurs at 4.8779
```



Optimization of hyperparameters.

Marcin Wolter

IFJ PAN

9 December 2020

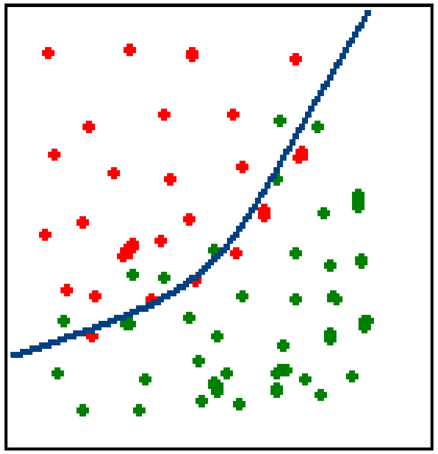
Your projects

- Examples of last year projects:
  - https://github.com/marcinwolter/MachineLearnin2019_projects
  - Andrii_Fedorchuk.ipynb – based on habits and hobby guess a gender
  - Clustering.py – cluster similar molecules
  - PCA.ipynb – nice use of PCA for chemistry
  - SvitlanaPastukh.ipynb – deep learning, image recognition
  - tau-_μμμ project.ipynb – high energy physics contest
- Interesting public datasets and notebooks analyzing them:
  - https://www.kaggle.com/datasets
  - https://www.kaggle.com/notebooks

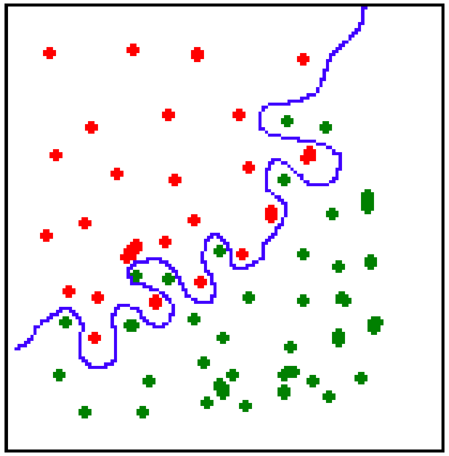
Good luck in finding a nice topic for your project!

Overtraining

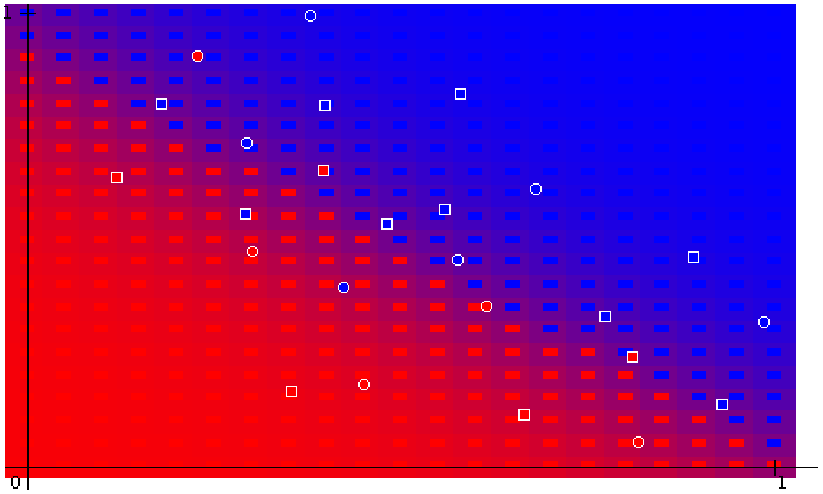
- **Overtraining** – algorithm "learns" the particular events, not the rules.
- This effect appears for all ML algorithms.
- Remedy – checking with another, independent dataset.

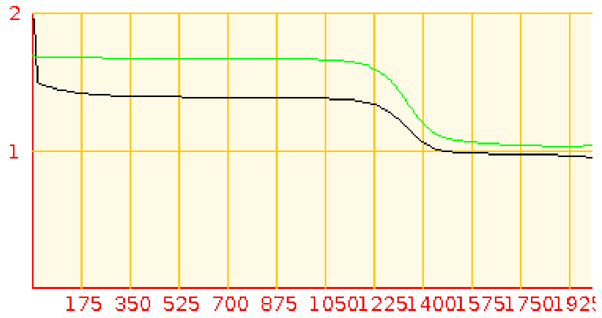


Correct



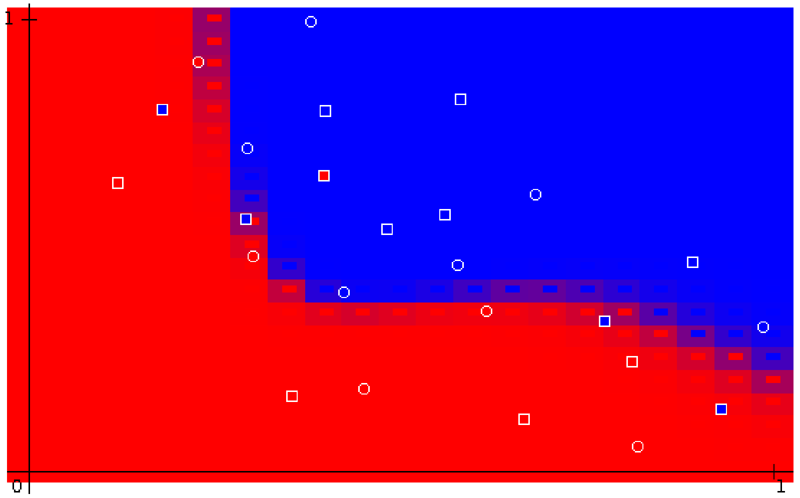
Overtraining

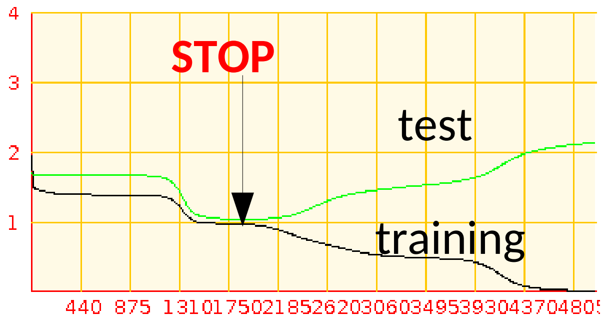




● 🟥 Training sample

■ 🟥 Test sample

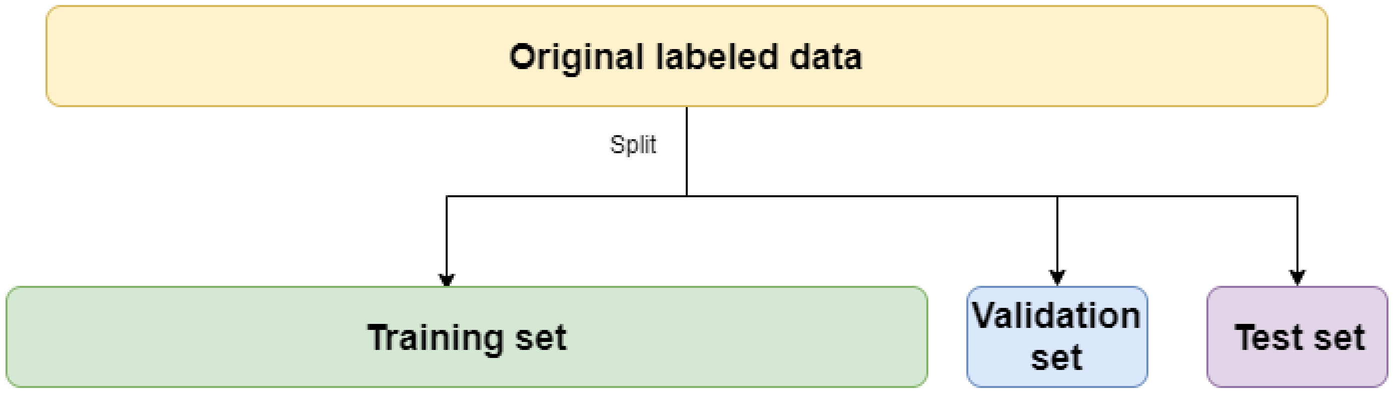




Example of using Neural Network.

How to train a ML algorithm?

- How to avoid **overtraining** while learning?
- Should we use one sample for **training** and another for **validating**?
- Then we increase the error – we use just a part of data for training.
- **Second remark:** to avoid ovetraining and find the performance of the trained algorithm we should use one more, **third data sample** to measure the final performance of the ML algorithm.



- How to **optimize the hyperparameters** of the ML algorithm (number of trees and their depth for BDT, number of hidden layers, nodes for Neural Network)?

Hyperparameter optimization

- Nearly each ML method has few hyperparameters (structure of the Neural Net, number of trees and their depth for BDT etc).
- They should be optimized for a given problem.
- **Task: for a given data sample find a set of hyperparameters, that the estimated error of the given method is minimized.**
- Looks like a typical minimization problem (fitting like), but:
  - Getting each measurement is costly
  - High noise
  - We can get the value of the minimized function (so our error) in the pont **x** of the hyperparameter space, but we can't get the differential easily.

How does it work in practice?

- Straight line fitting

  $y(x, w) = w_0 + w_1 x$ fit to the data.

1) Gaussian prior, no data used

2) First data point. We find the likelihood based on this point (left plot) and multiply: priori*likelihood. We get the posterior distribution (right plot).

3) We add the second point and repeat the procedure.

4) Adding all the points one by one.

Remark: data are noisy.

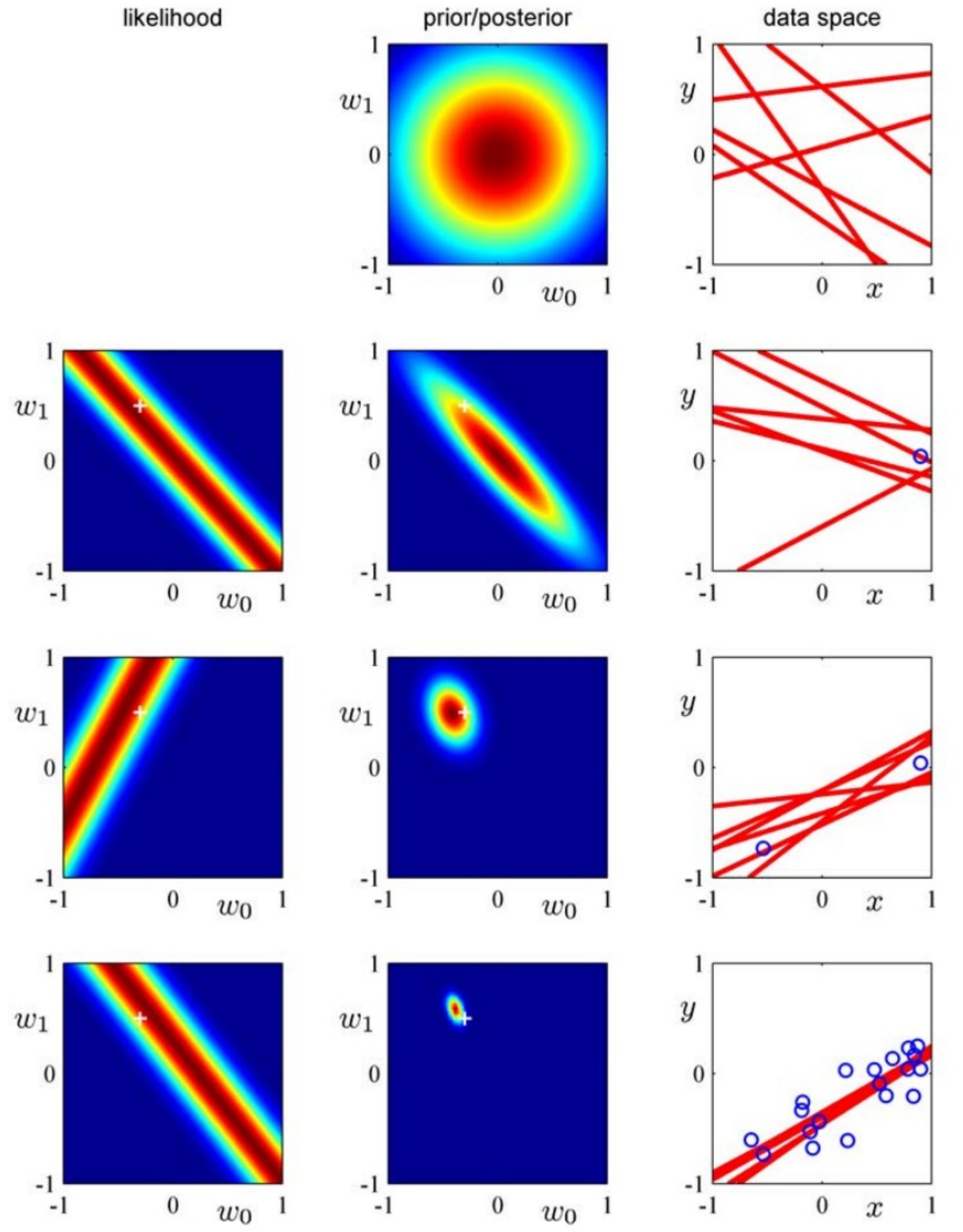
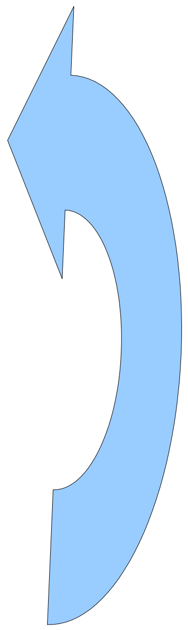


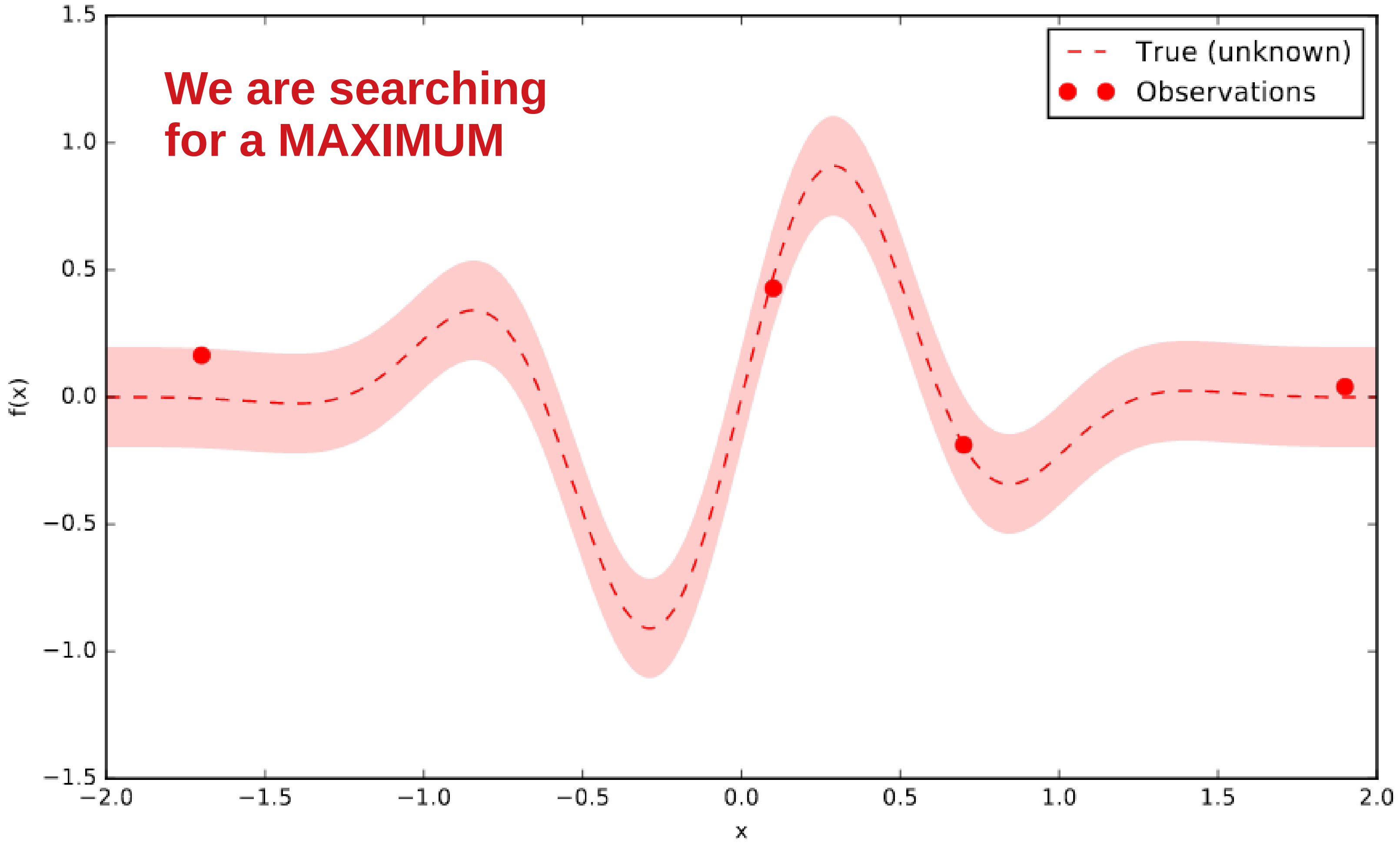
Illustration of sequential Bayesian learning for a simple linear model of the form $y(x, \mathbf{w}) = w_0 + w_1 x$. A detailed description of this figure is given in the text.

Optimization of hyperparameters

- How to optimize:
  - „Grid search” - scan over all possible values of parameters.
  - „Random search”
  - Some type of fitting…
- Popular method is the **„bayesian optimization”**
  - Build the probability model
  - Take „a priori” distributions of parameters
  - Find, for which point in the hyperparameter space you can maximally improve your model
  - Find the value of error
  - Find the „a posteriori” probability distribution
  - Repeat



Starting point

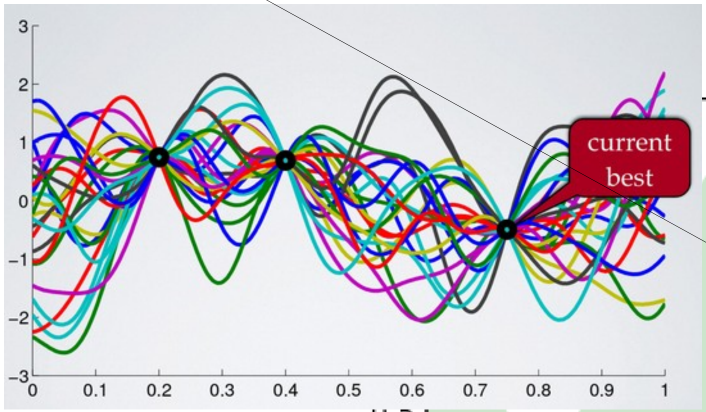


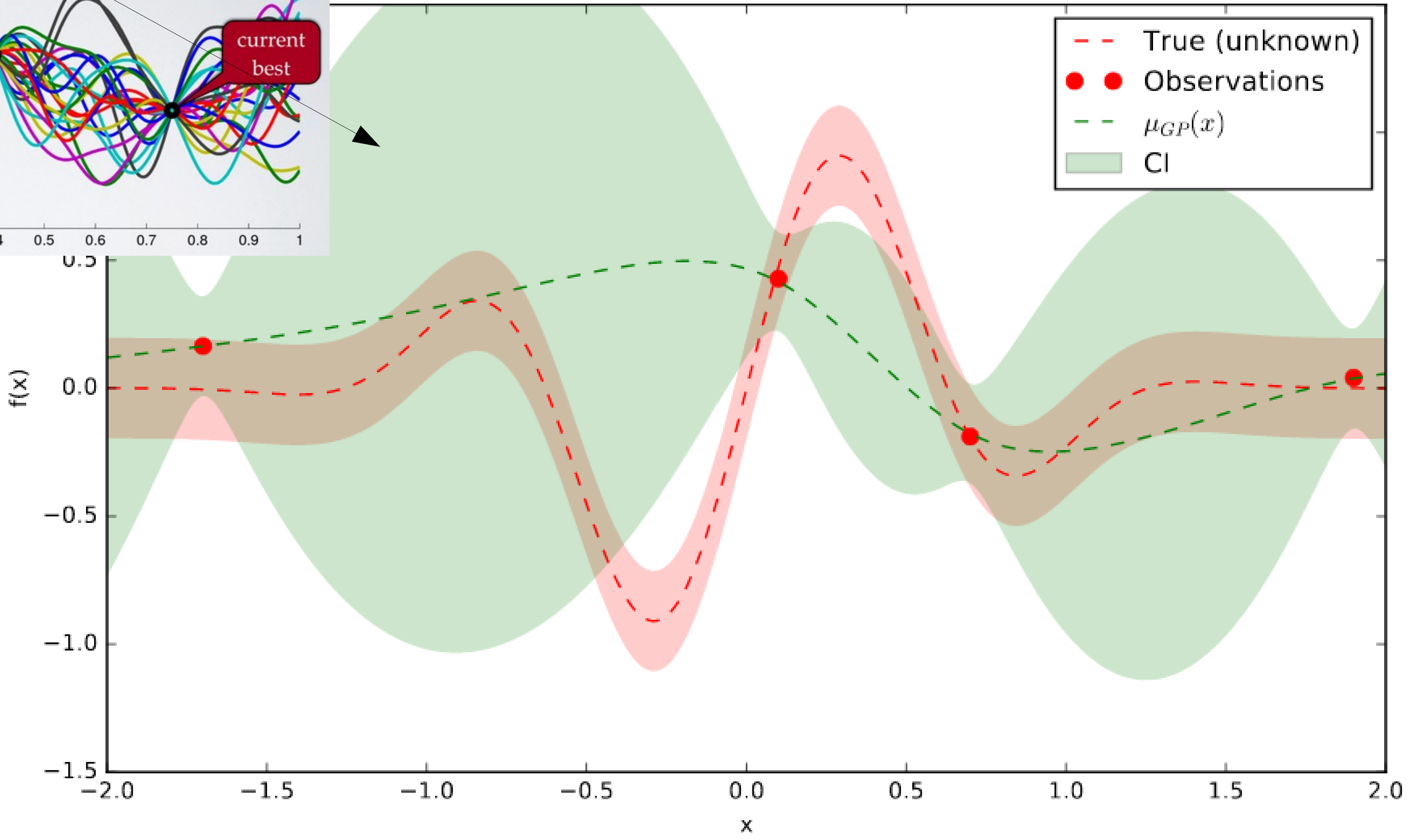
Unknown function (with noise), four observations.
Where should we do the next costly probing?

https://www.iro.umontreal.ca/~bengioy/cifar/NCAP2014-summerschool/slides/Ryan_adams_140814_bayesopt_ncap.pdf

A posteriori distribution

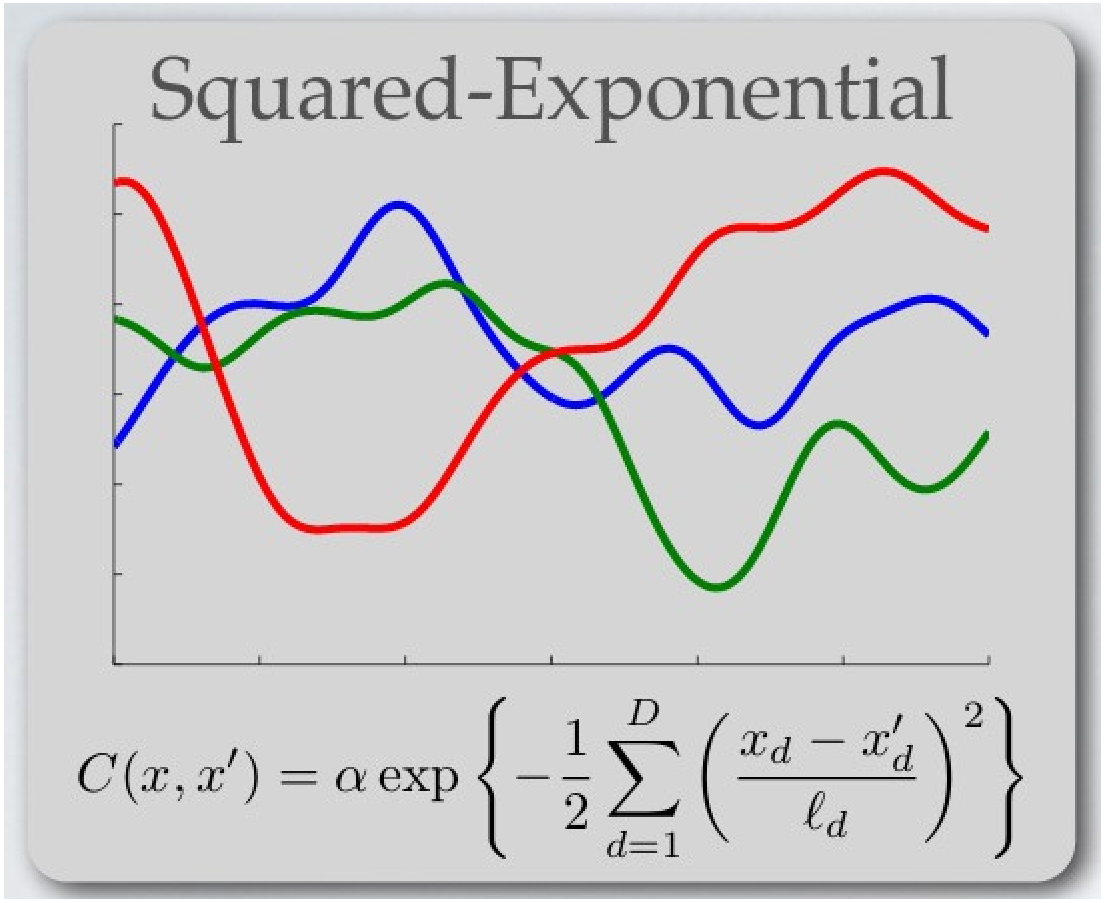
A set of functions





The a posteriori distribution of possible functions, those functions could generate the observed data points.

A posteriori functions – Gaussian Processes (GP)



$$C(x, x') = \alpha \exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_d - x'_d}{\ell_d}\right)^2\right\}$$

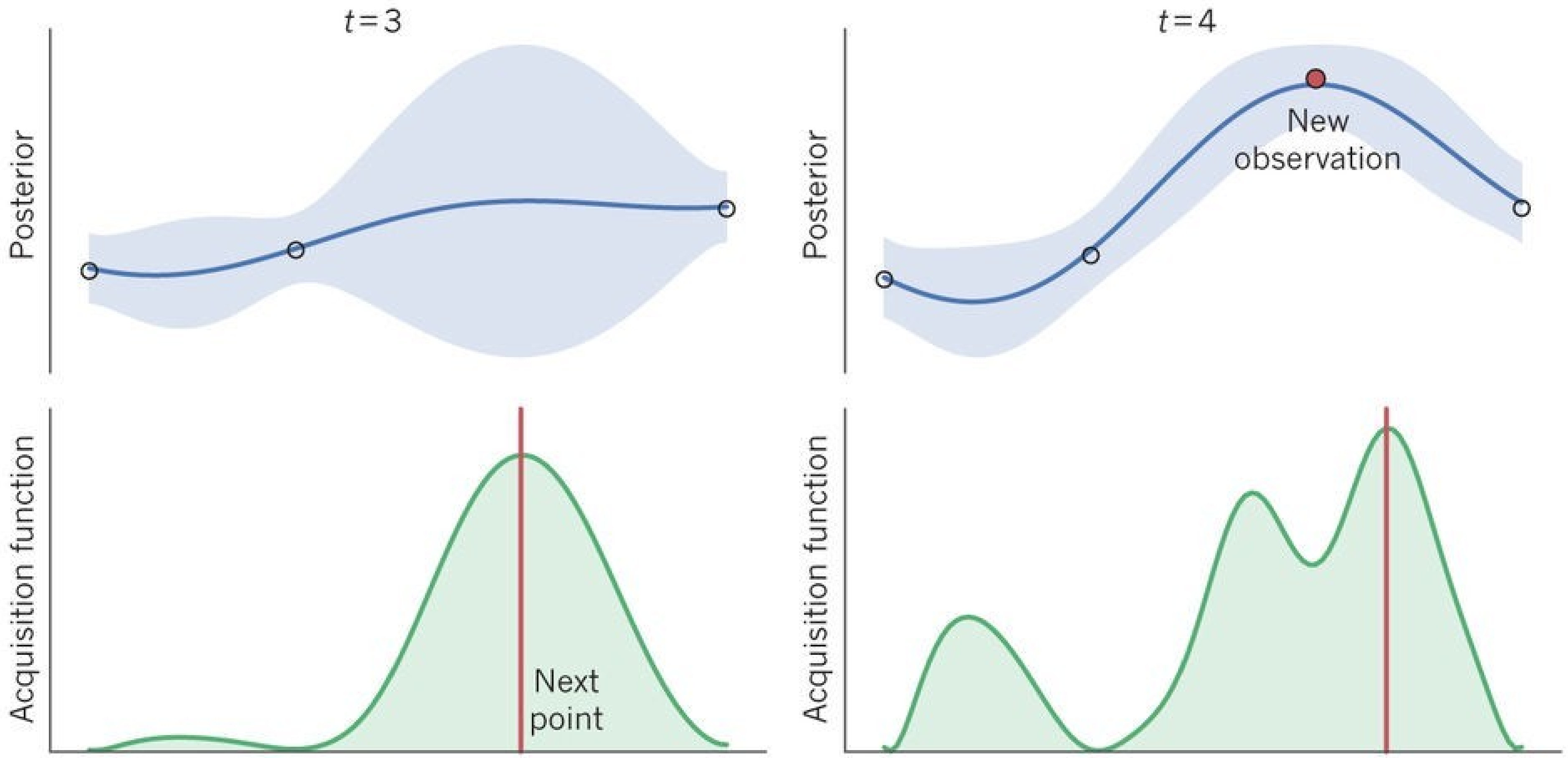
These functions should be somehow parametrized, for example they could be Gaussian functions.

Acquisition function

- Posterior GP (Gaussian Processes) give us the mean of GP functions $\mu(x)$ and their expected variation $\sigma^2(x)$.
  - **Exploration** – searching for huge variation
  - **Exploitation** – searching for a smallest/greatest (depends on sign and convention) value of mean $\mu(x)$
- The acquisition policy has to balance these two approaches

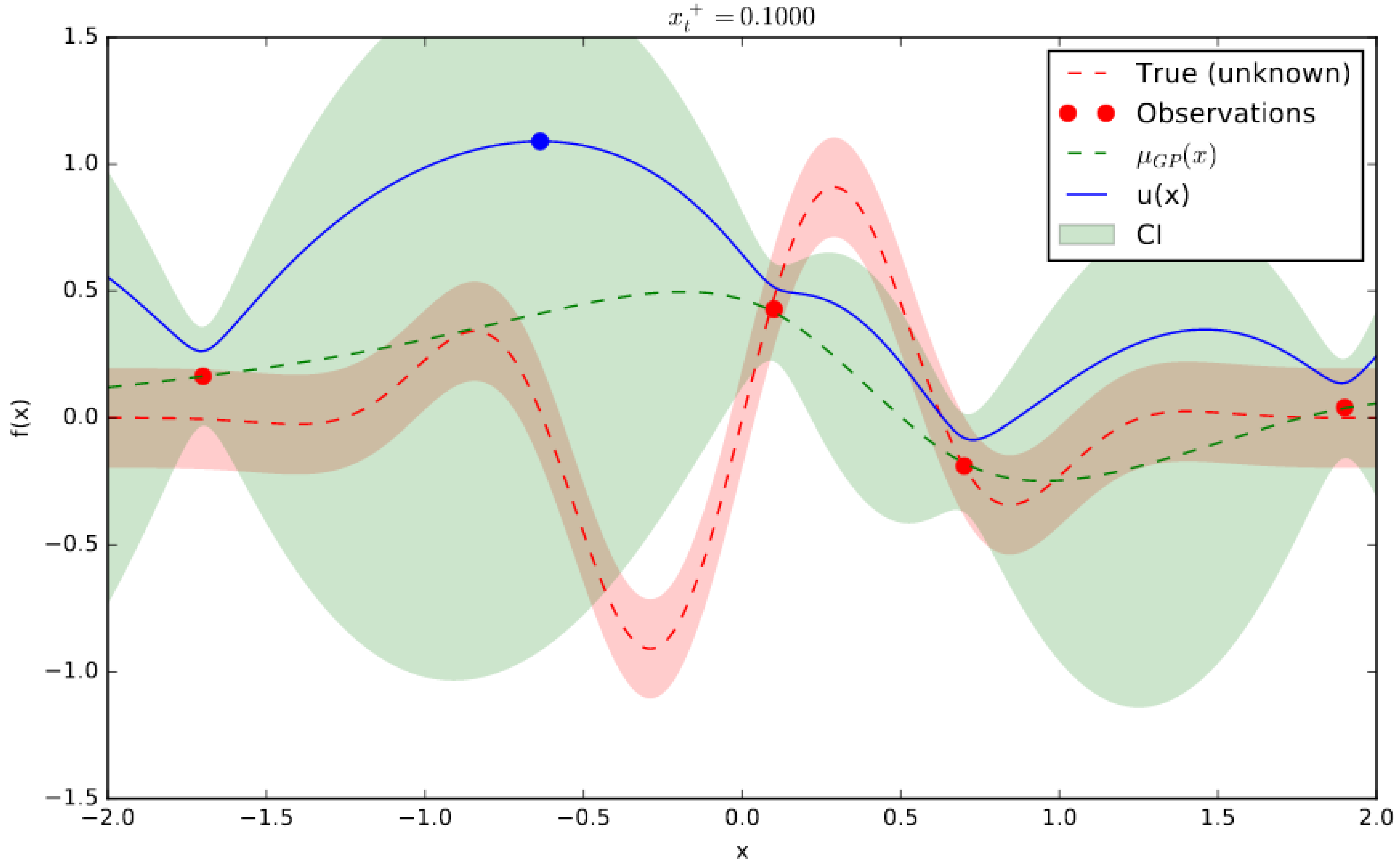
Where to put the next point?

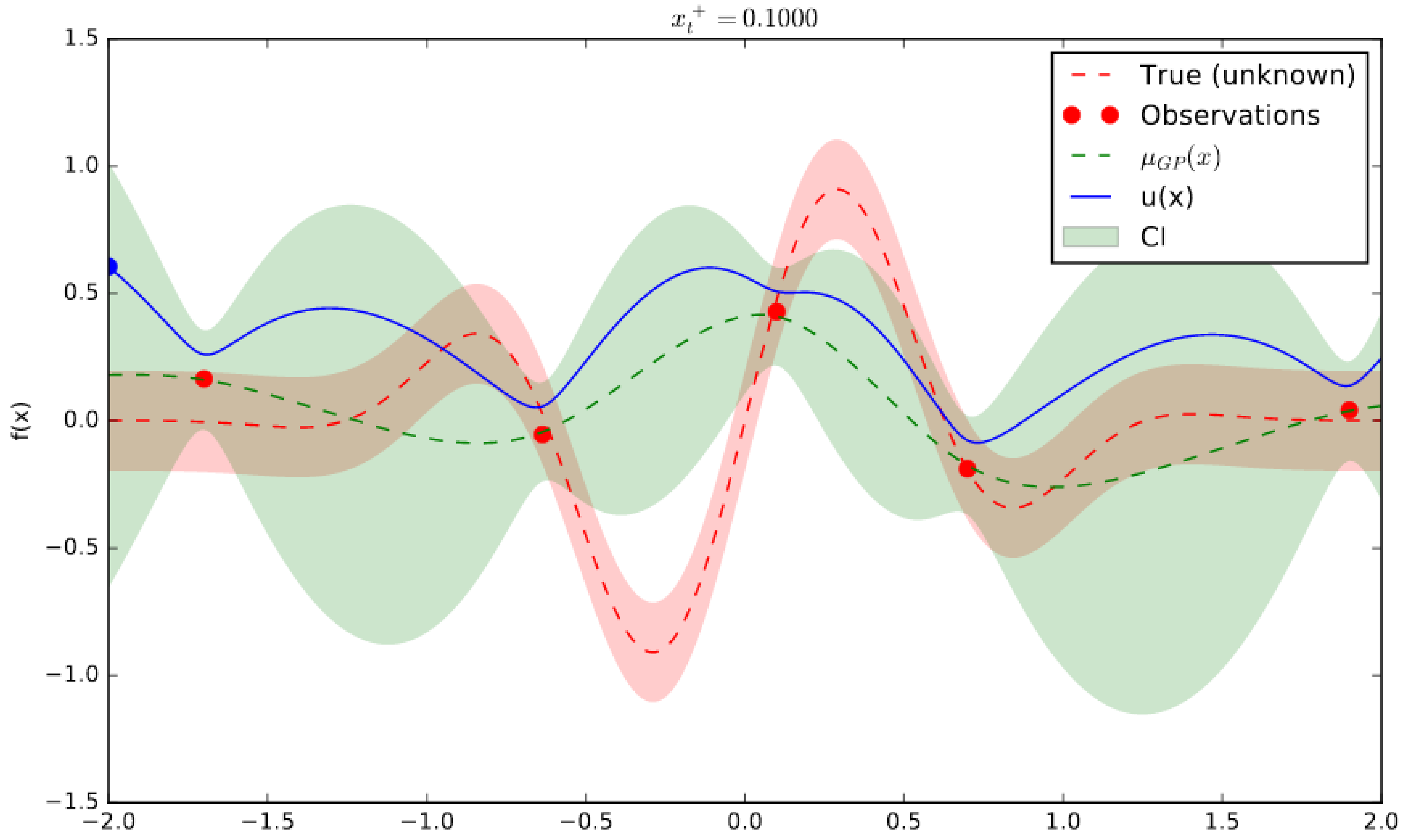
- Our next chosen point(x) should has high mean (exploitation) & high variance (exploration).



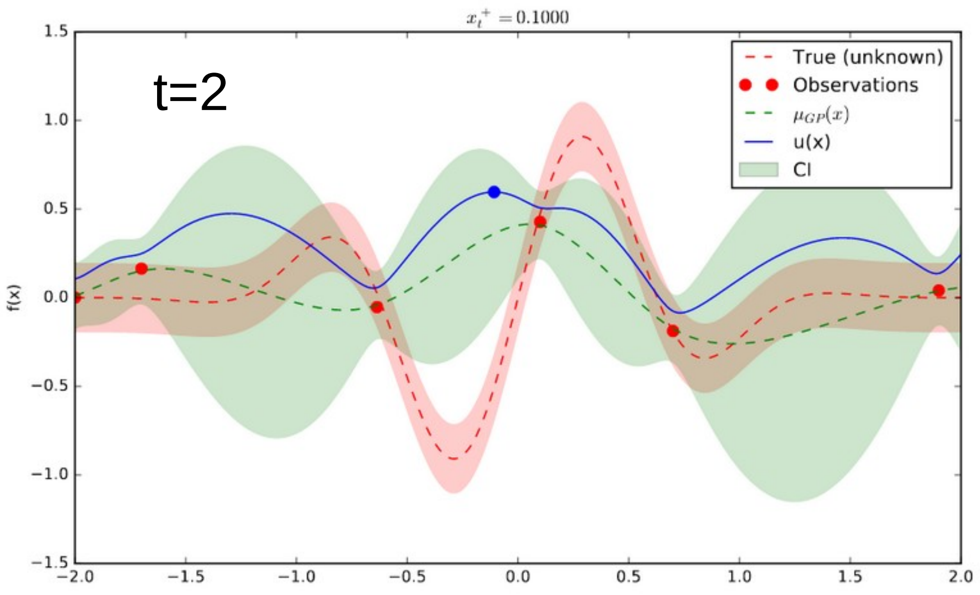
We choose next x

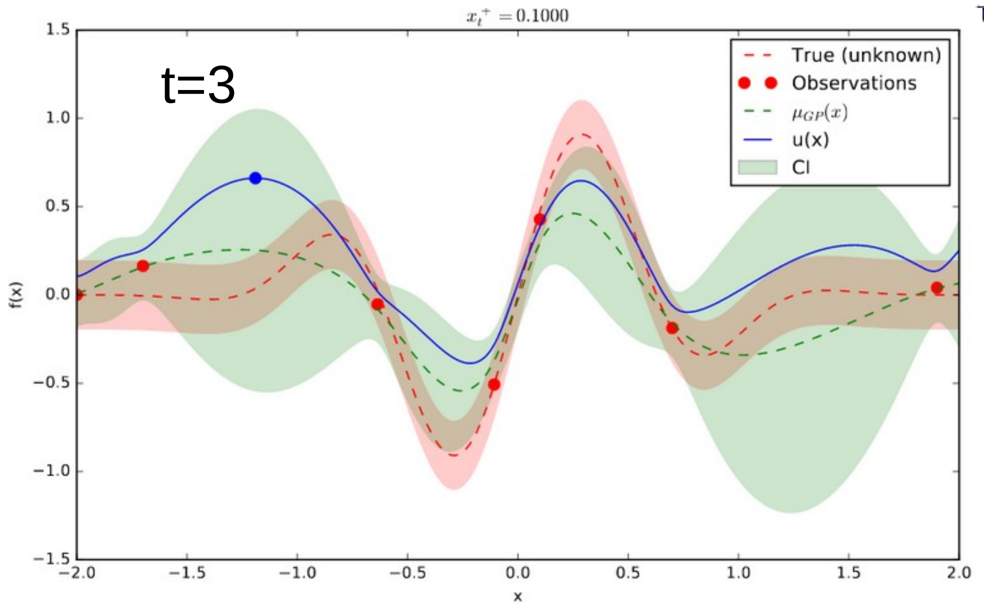
u(x) – acqusition function
(finding maximum)

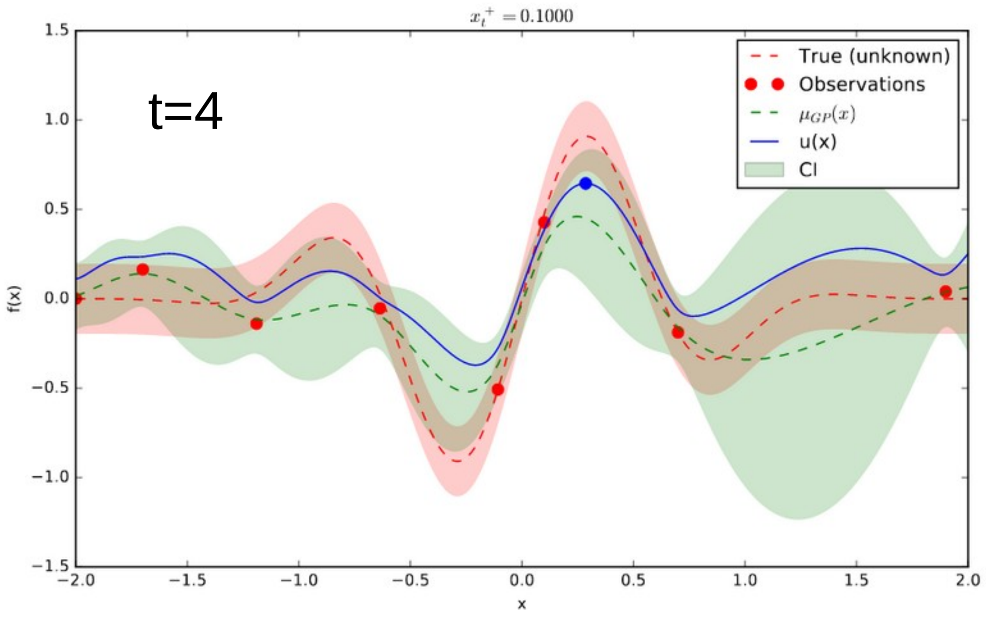


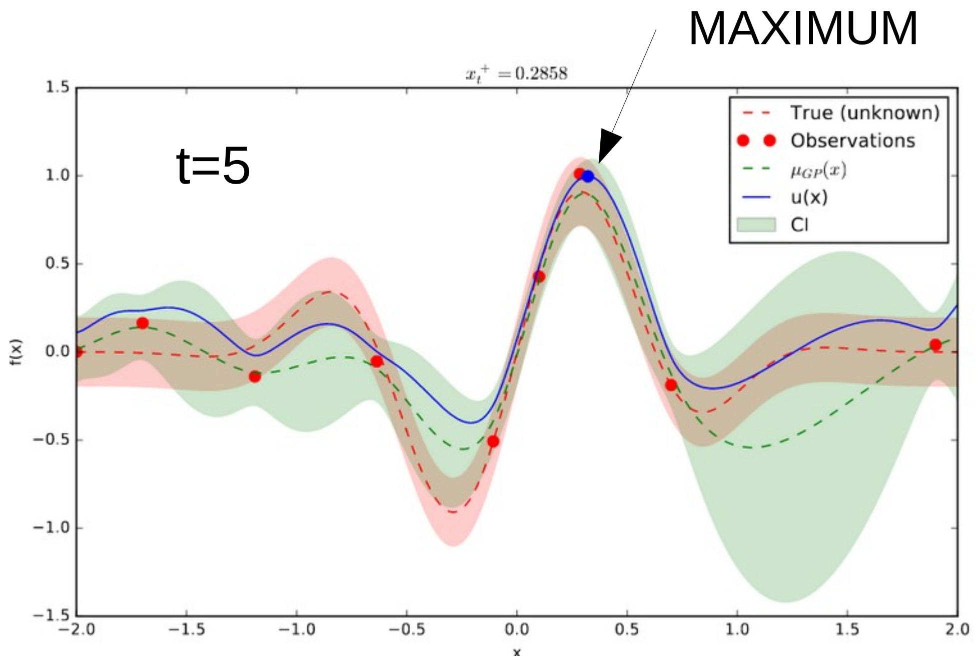


Dokonujemy próbkowania i powtarzamy procedurę...


$x_t^+ = 0.1000$
t=2
True (unknown)
Observations
$\mu_{GP}(x)$
u(x)
CI
f(x)


$x_t^+ = 0.1000$
t=3
True (unknown)
Observations
$\mu_{GP}(x)$
u(x)
CI
f(x)
x


$x_t^+ = 0.1000$
t=4
True (unknown)
Observations
$\mu_{GP}(x)$
u(x)
CI
f(x)
x


MAXIMUM
$x_t^+ = 0.2858$
t=5
True (unknown)
Observations
$\mu_{GP}(x)$
u(x)
CI
f(x)
x

Limitations

- Bayesian optimization depends on the parameters chosen
- On the acquisition function
- On the prior selected....
- It’s **sequential.**
- Implementations:
  - Tree of Parzen Estimators (TPE) used by the HyperOpt package https://github.com/hyperopt/hyperopt).
  - OPTUNA package https://optuna.org/ Quite advanced, grid search, random sampling,TPE i Covariance Matrix Adaptation Evolution Strategy CMA-ES algorithms
  - Many more packages...

OPTUNA vs. HYPEROPT comparison:
https://neptune.ai/blog/optuna-vs-hyperopt

Examples

- Simple example with HYPEROPT & OPTUNA:

  https://github.com/marcinwolter/MachineLearning2020/blob/main/hyperopt_optuna_demo.ipynb

- Optimization of digits reading MNIST with HYPEROPT and OPTUNA:

  https://github.com/marcinwolter/MachineLearning2020/blob/main/mnist_mlp_minimal_hyperopt.ipynb

  https://github.com/marcinwolter/MachineLearning2020/blob/main/mnist_mlp_minimal_optuna.ipynb

Summary

- We have learned today how to optimize the hyperparameters!