Transfer Learning for Improving Model Predictions in Highly Configurable Software

Pooyan Jamshidi, **Miguel Velez**, Christian Kästner,
Norbert Siegmund, Prasad Kawthekar

Carnegie Mellon University

Transfer Learning for Improving Model Predictions in Highly Configurable Software

Pooyan Jamshidi, Miguel Velez, Christian Kästner
Carnegie Mellon University, USA
{pjamshid,mvelezce,kaestner}@cs.cmu.edu

Norbert Siegmund
Bauhaus-University Weimar, Germany
norbert.siegmund@uni-weimar.de

Prasad Kawthekar
Stanford University, USA
pkawthek@stanford.edu

***Abstract*—Modern software systems are built to be used in dynamic environments using configuration capabilities to adapt to changes and external uncertainties. In a self-adaptation context, we are often interested in reasoning about the performance of the systems under different configurations. Usually, we learn a black-box model based on real measurements to predict the performance of the system given a specific configuration. However, as modern systems become more complex, there are many configuration parameters that may interact and we end up learning an exponentially large configuration space. Naturally, this does not scale when relying on real measurements in the actual changing environment. We propose a different solution: Instead of taking the measurements from the real system, we learn the model using samples from other sources, such as simulators that approximate performance of the real system at low cost. We define a cost model that transform the traditional view of model learning into a multi-objective problem that not only takes into account model accuracy but also measurements effort as well. We evaluate our cost-aware transfer learning solution using real-world configurable software including (i) a robotic system, (ii) 3 different stream processing applications, and (iii) a NoSQL database system. The experimental results demonstrate that our approach can achieve (a) a high prediction accuracy, as well as (b) a high model reliability.**

***Index Terms*—highly configurable software, machine learning, model learning, model prediction, transfer learning**

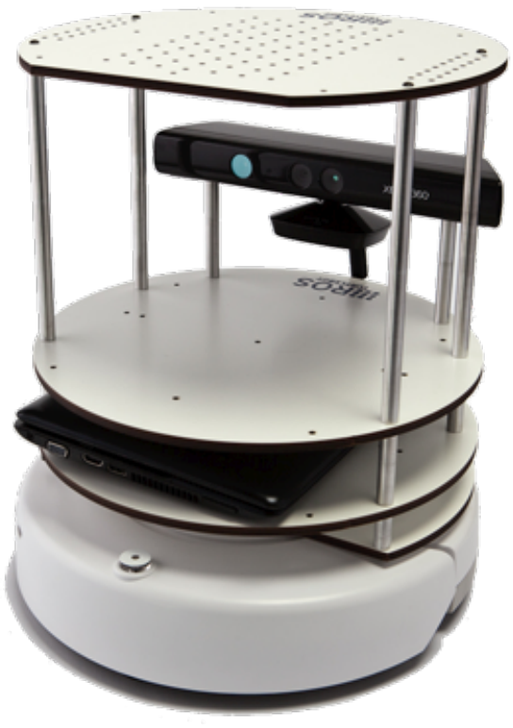
Fig. 1

order to
when it
Typic
figurabl
selected
perform
configu
their in
of learn
softwar

Highly Configurable Systems In Dynamic Environments

[Credit to CMU CoBot]

Options Influence Performance



rqt_reconfigure__Param - rqt

Dynamic Reconfigure

Filter key:

Collapse all | Expand all

- move_base
  - TrajectoryPlannerROS
  - global_costmap
  - local_costmap

/move_base/TrajectoryPlannerROS

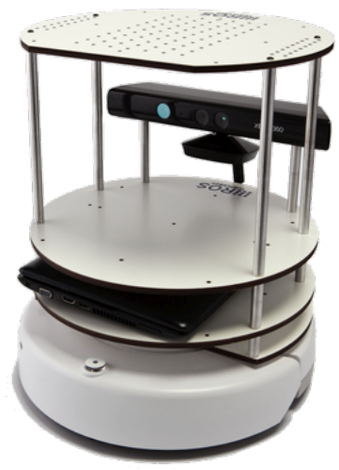
acc_lim_x	0.0	20.0	1.5
acc_lim_y	0.0	20.0	0.0
acc_lim_theta	0.0	20.0	1.2
max_vel_x	0.0	20.0	0.15
min_vel_x	0.0	20.0	0.1
max_vel_theta	0.0	20.0	1.0
min_vel_theta	-20.0	0.0	-1.0
min_in_place_vel_theta	0.0	20.0	0.4
sim_time	0.0	10.0	1.0
sim_granularity	0.0	5.0	0.05
angular_sim_granularity	0.0	1.57079632679	0.1
pdist_scale	0.0	5.0	0.5
gdist_scale	0.0	5.0	0.8
occdist_scale	0.0	5.0	0.05
oscillation_reset_dist	0.0	5.0	0.05
escape_reset_dist	0.0	5.0	0.1
escape_reset_theta	0.0	5.0	1.57079632679
vx_samples	1	300	14

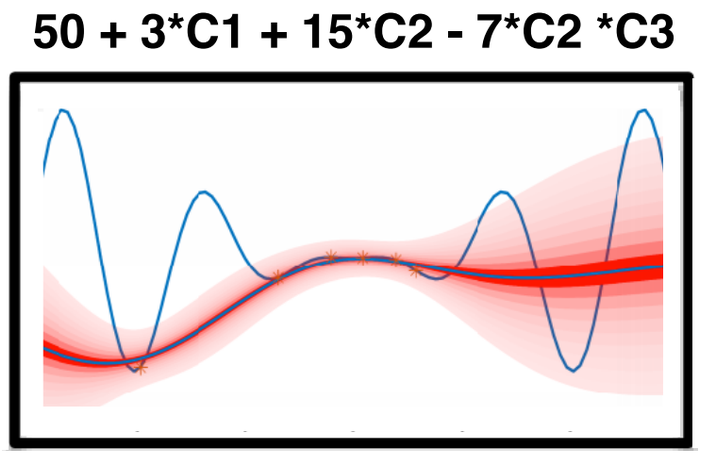
Adapt to Different Environments

TurtleBot

Adapt to Different Environments

TurtleBot

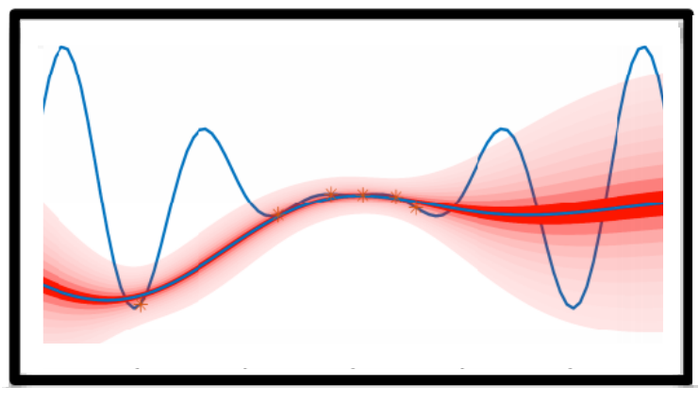


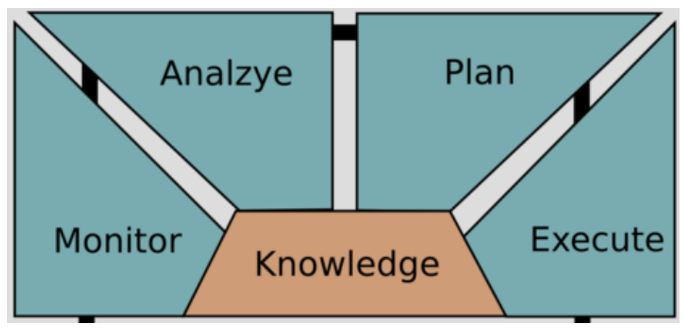


Adapt to Different Environments

TurtleBot

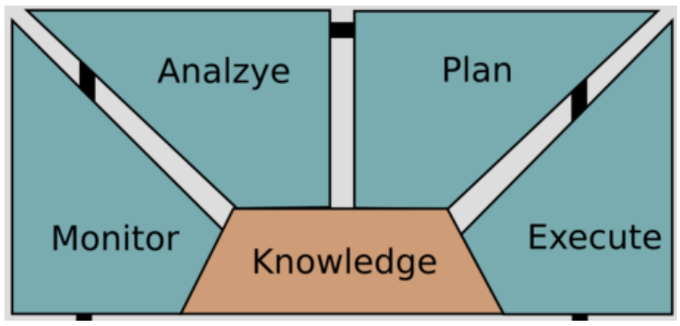
50 + 3*C1 + 15*C2 - 7*C2 *C3



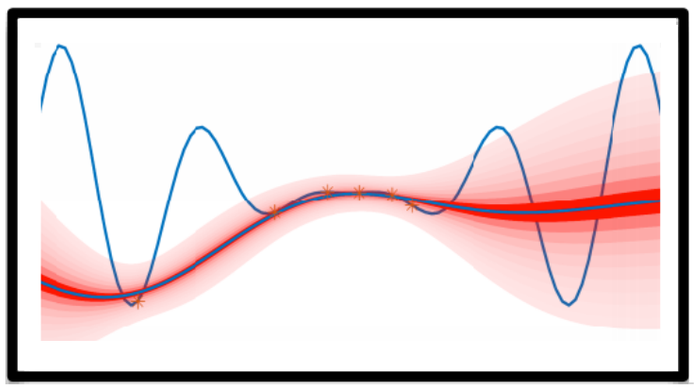


Classic Sensitivity Analysis

TurtleBot



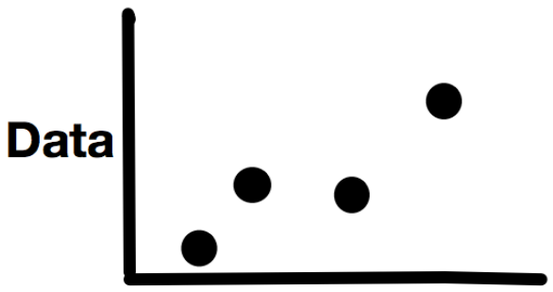
50 + 3*C1 + 15*C2 - 7*C2 *C3

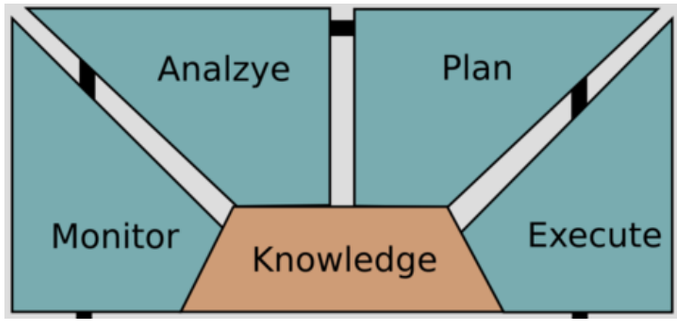


Classic Sensitivity Analysis

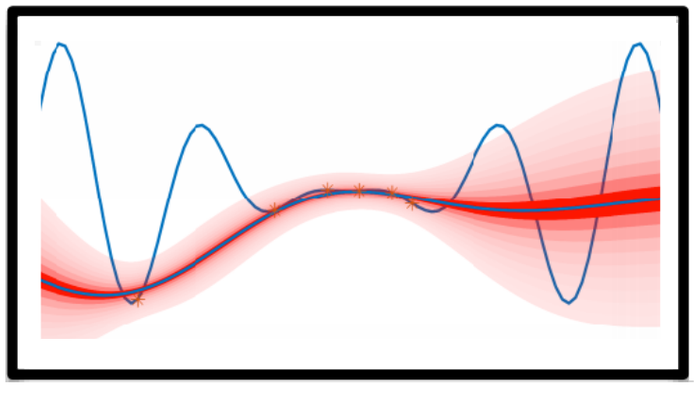
TurtleBot

Measure



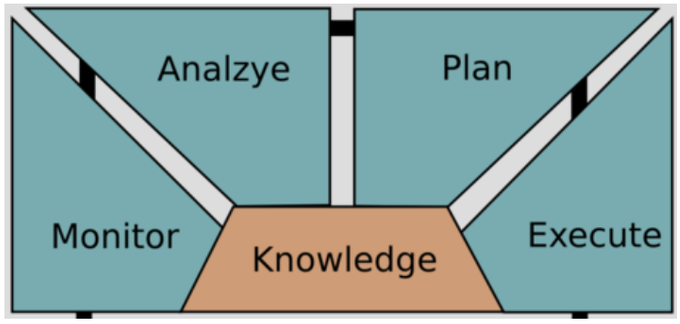


50 + 3*C1 + 15*C2 - 7*C2 *C3

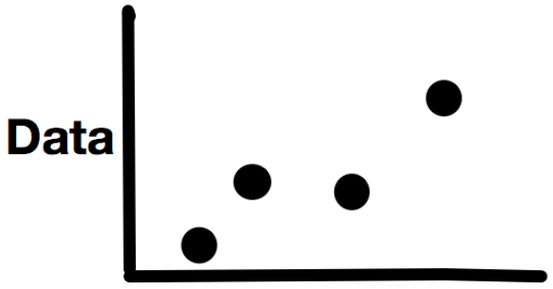


Classic Sensitivity Analysis

TurtleBot

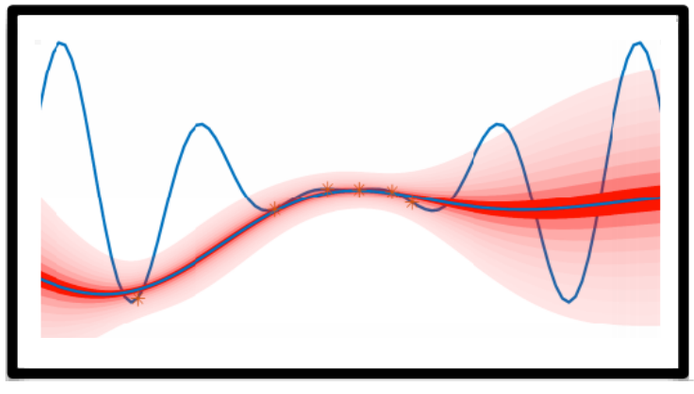


Measure



Learn

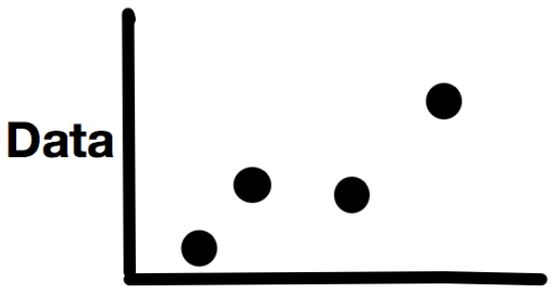
50 + 3*C1 + 15*C2 - 7*C2 *C3



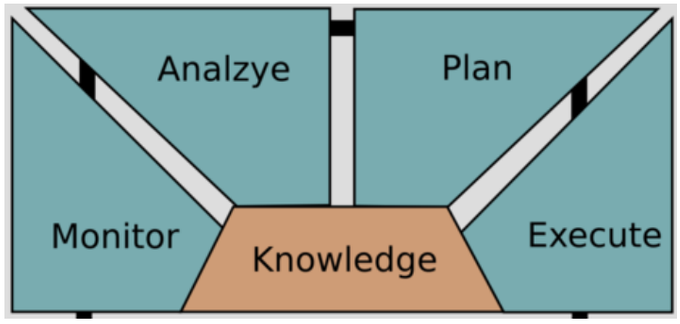
Classic Sensitivity Analysis

TurtleBot

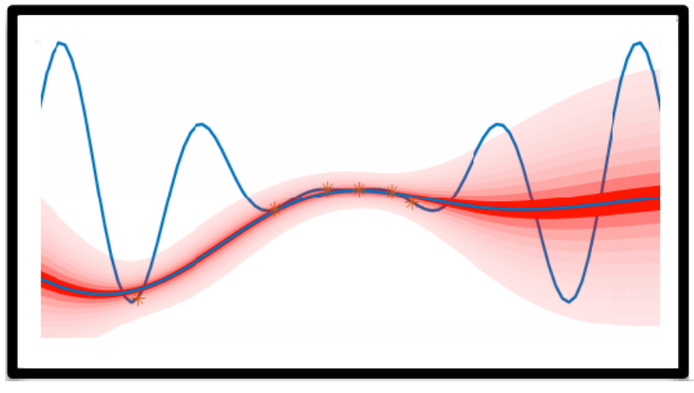
Measure



Learn



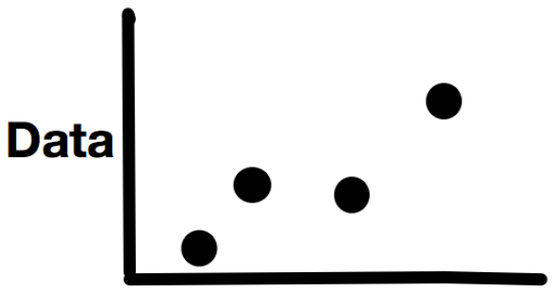
50 + 3*C1 + 15*C2 - 7*C2 *C3



Applications

TurtleBot

Measure



Learn

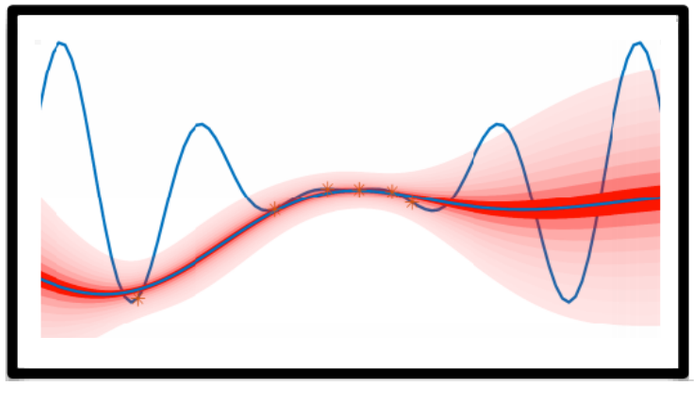
Optimization

Adaptation +

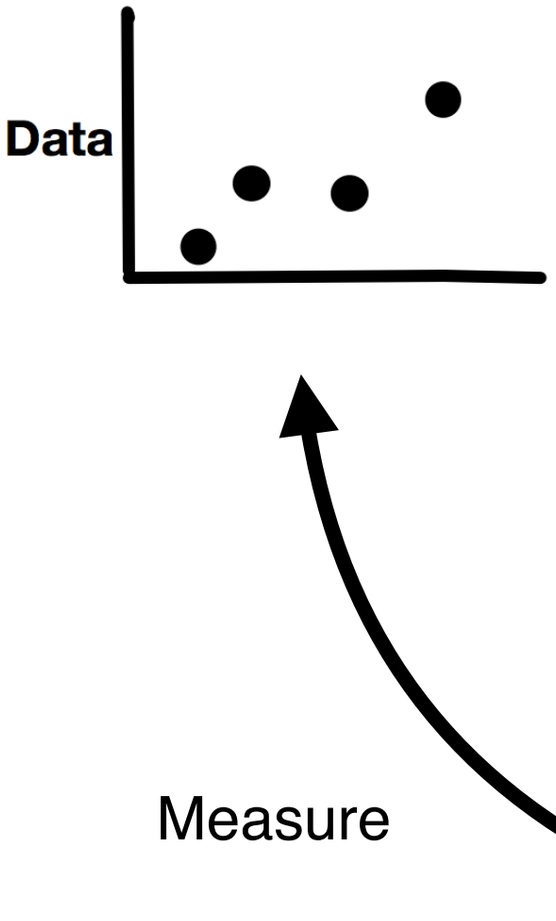
Reasoning +

Debugging +

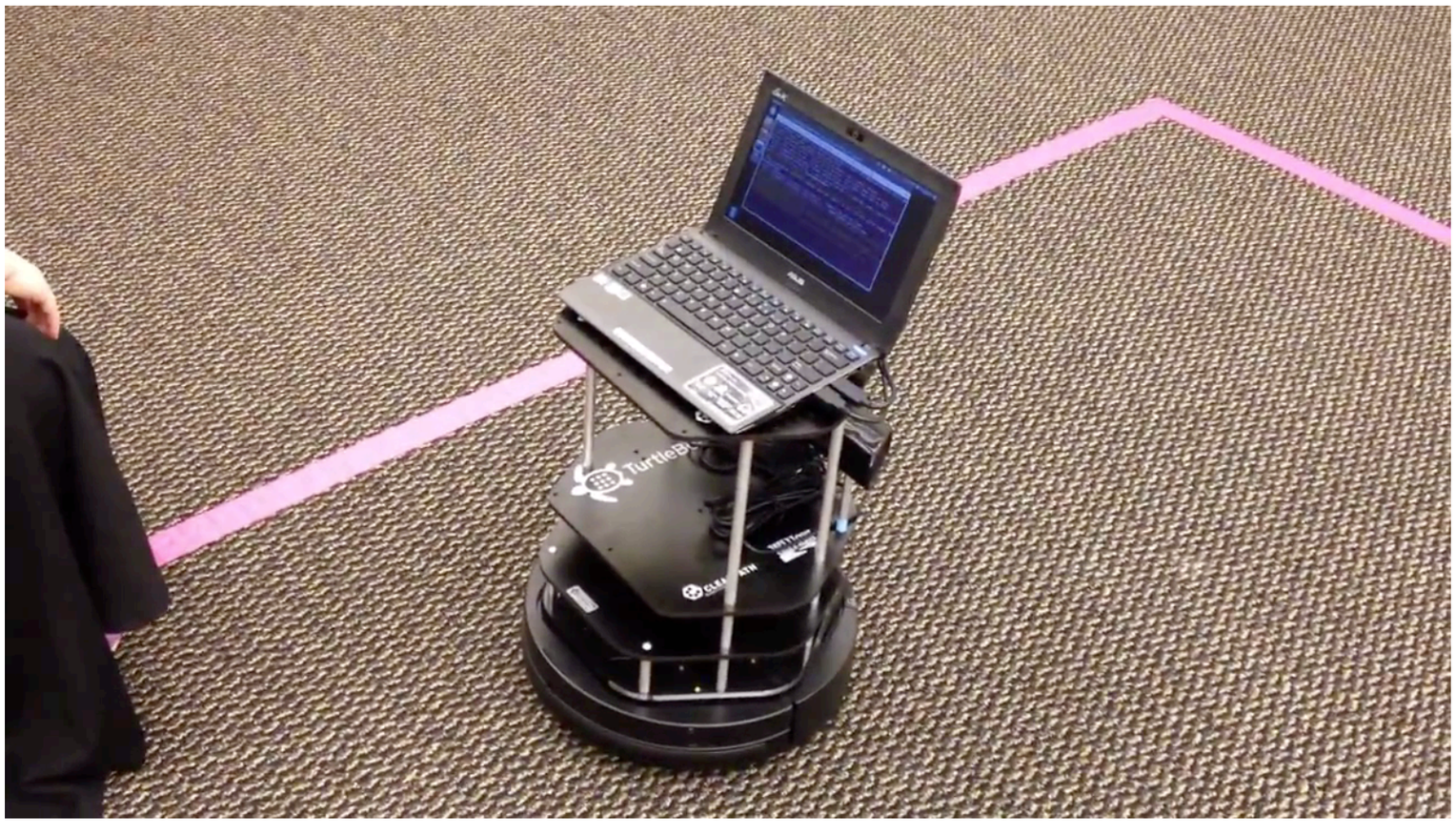
50 + 3*C1 + 15*C2 - 7*C2 *C3



Measuring Performance is Expensive



25 options × 10 values = **10^{25} configurations**



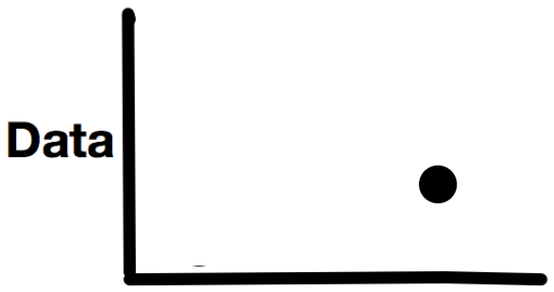
[Credit to Peng Hou]

Transfer Learning

Reuse Data From Similar System

TurtleBot

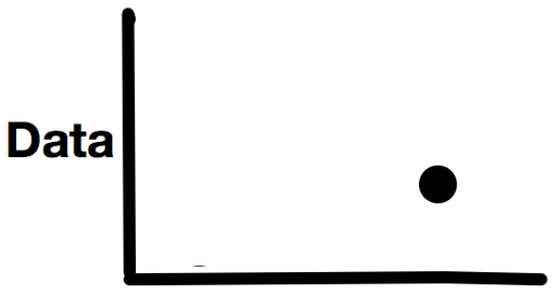
Measure



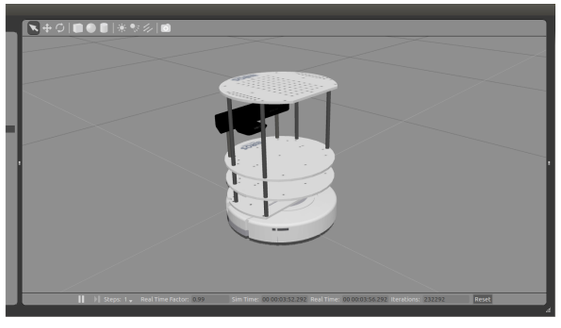
Reuse Data From Similar System

TurtleBot

Measure



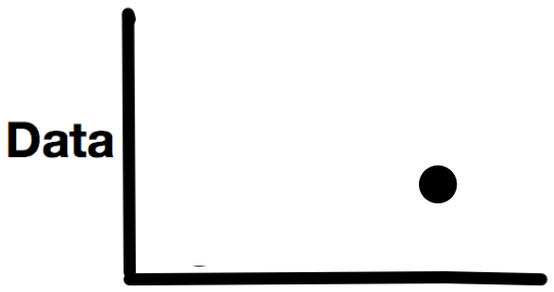
Simulator (Gazebo)



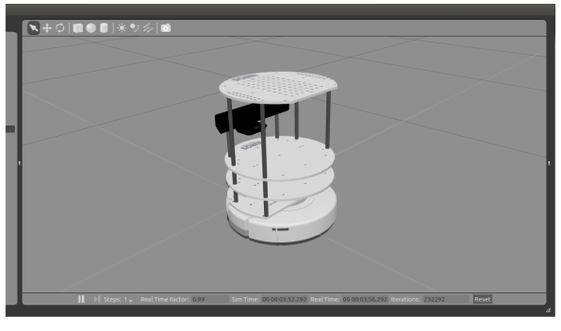
Reuse Data From Similar System

TurtleBot

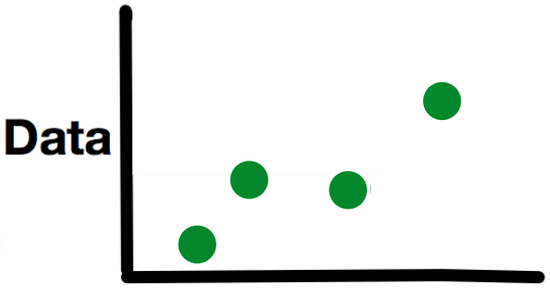
Measure



Simulator (Gazebo)



Measure

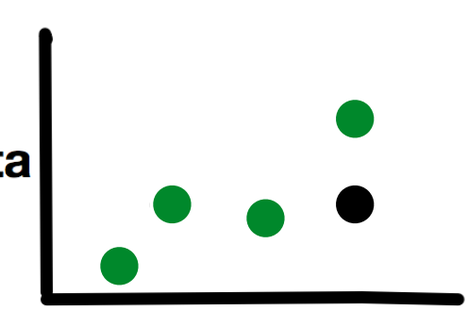


Reuse Data From Similar System

TurtleBot

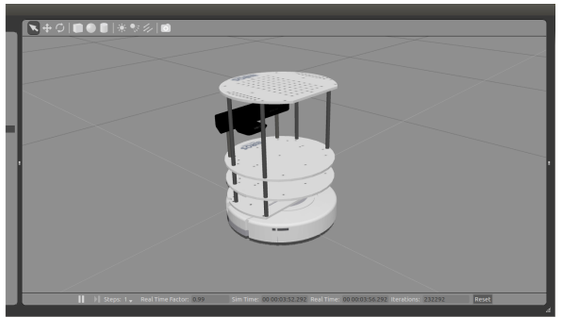
Measure

Data



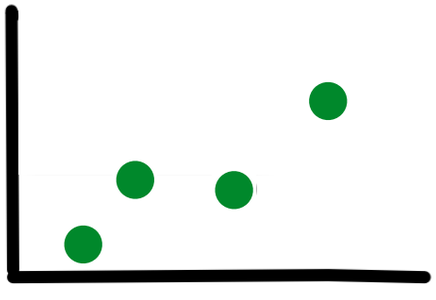
Reuse

Simulator (Gazebo)



Measure

Data

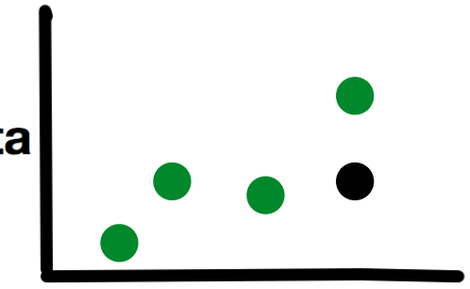


Reuse Data From Similar System

TurtleBot

Measure

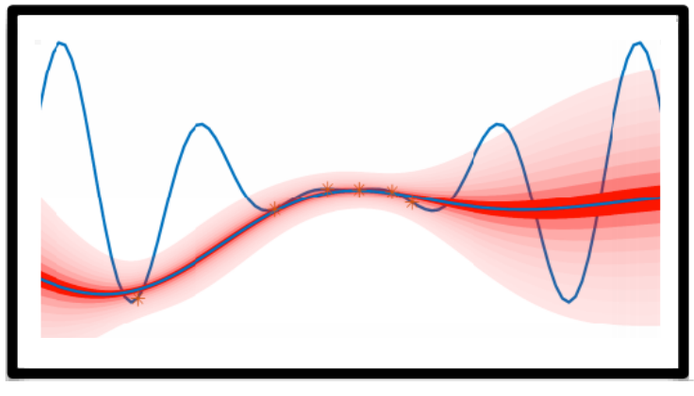
Data



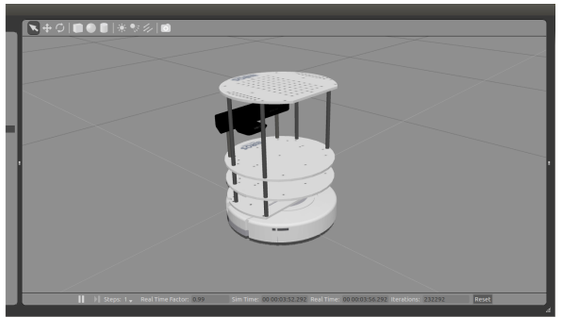
Reuse

Learn
with TL

50 + 3*C1 + 15*C2 - 7*C2 *C3

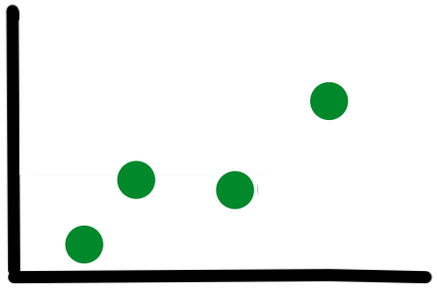


Simulator (Gazebo)



Measure

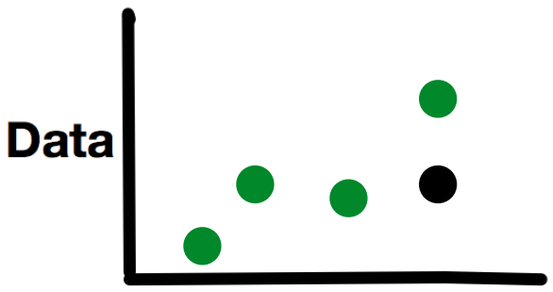
Data

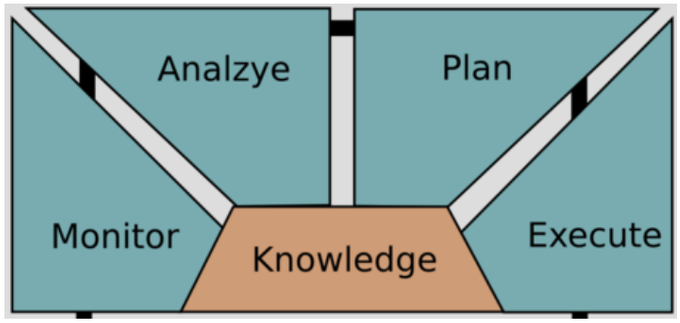


Reuse Data From Similar System

TurtleBot

Measure

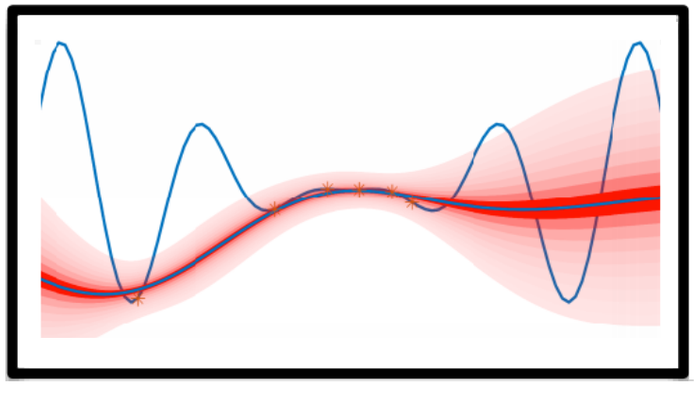




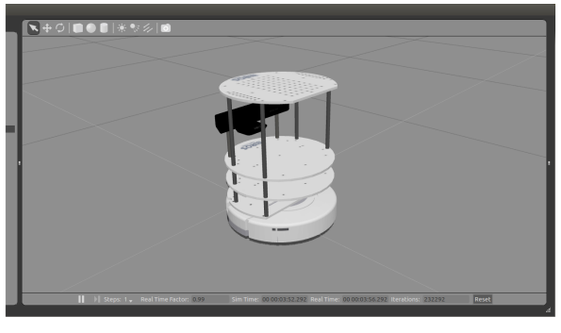
Reuse

Learn
with TL

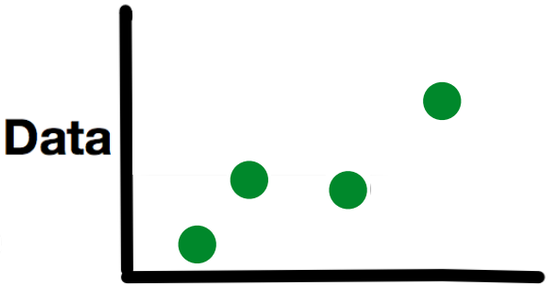
50 + 3*C1 + 15*C2 - 7*C2 *C3



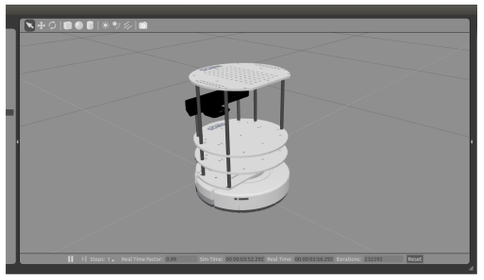
Simulator (Gazebo)



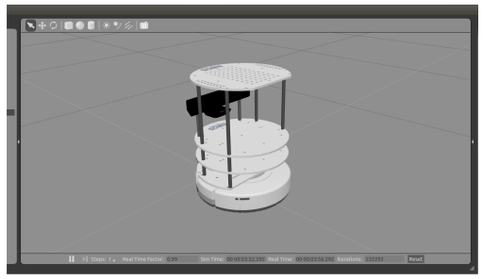
Measure

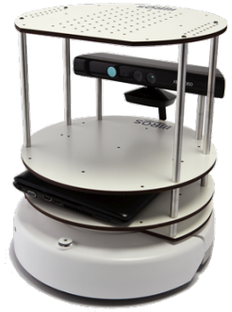


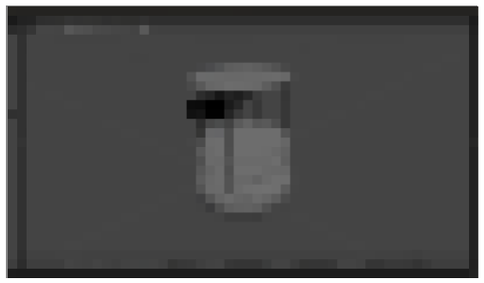
Transfer Between Different Systems



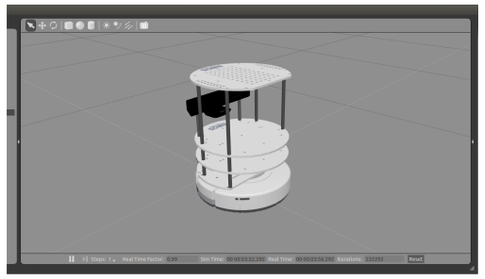
Transfer Between Different Systems

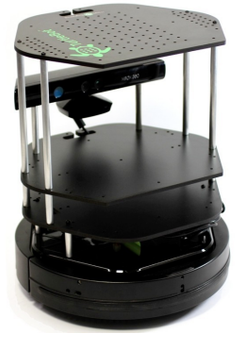




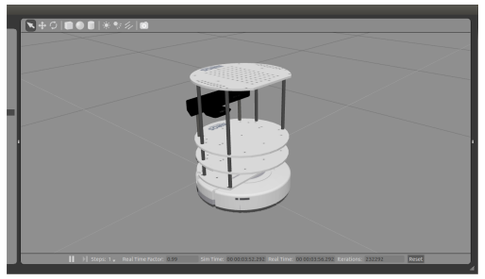


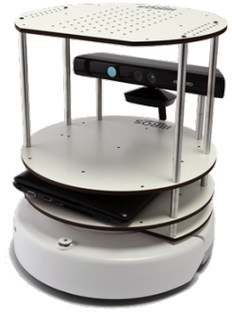
Transfer Between Different Systems





Transfer Between Different Systems



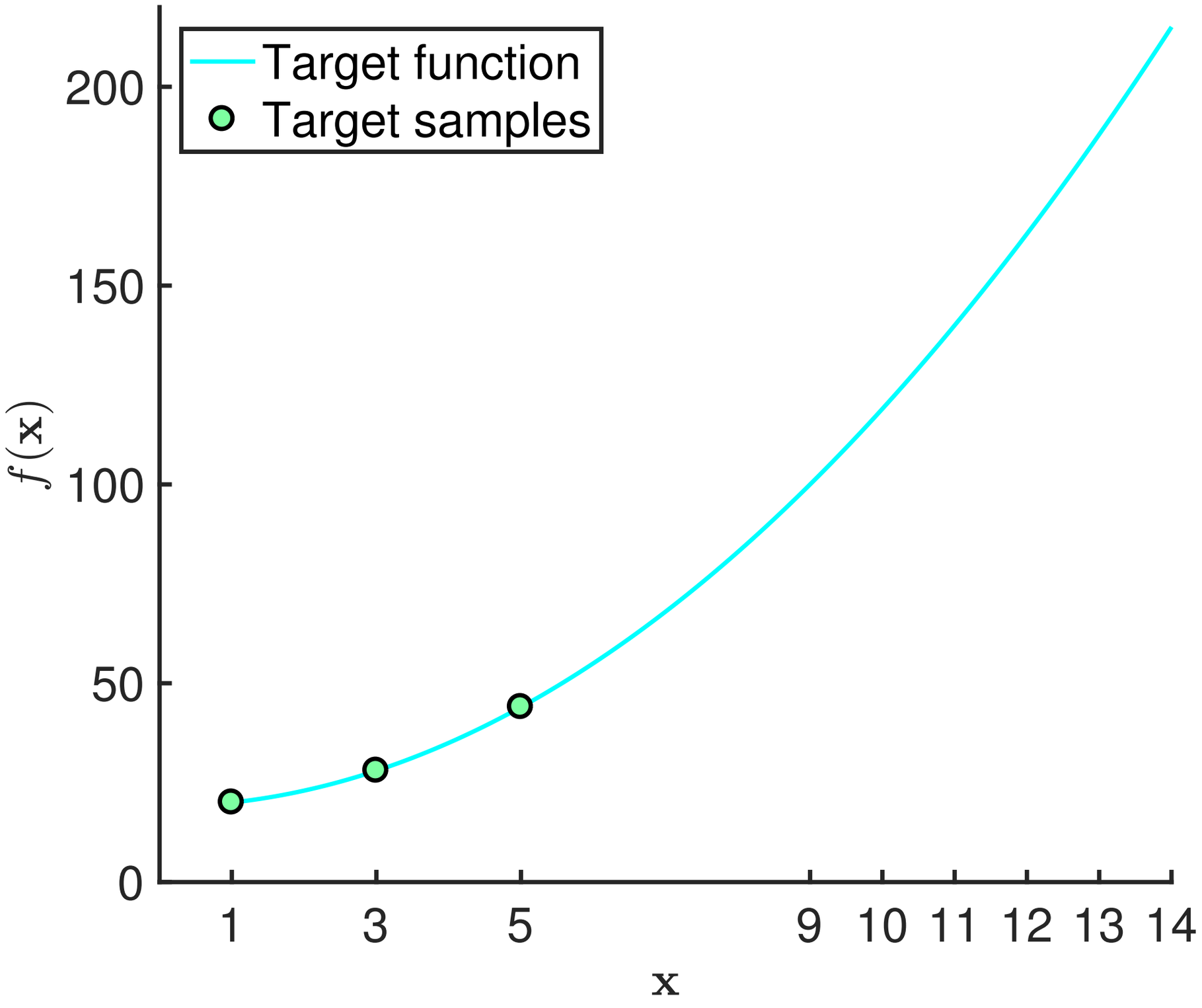


(0, 0) → (0, 10)

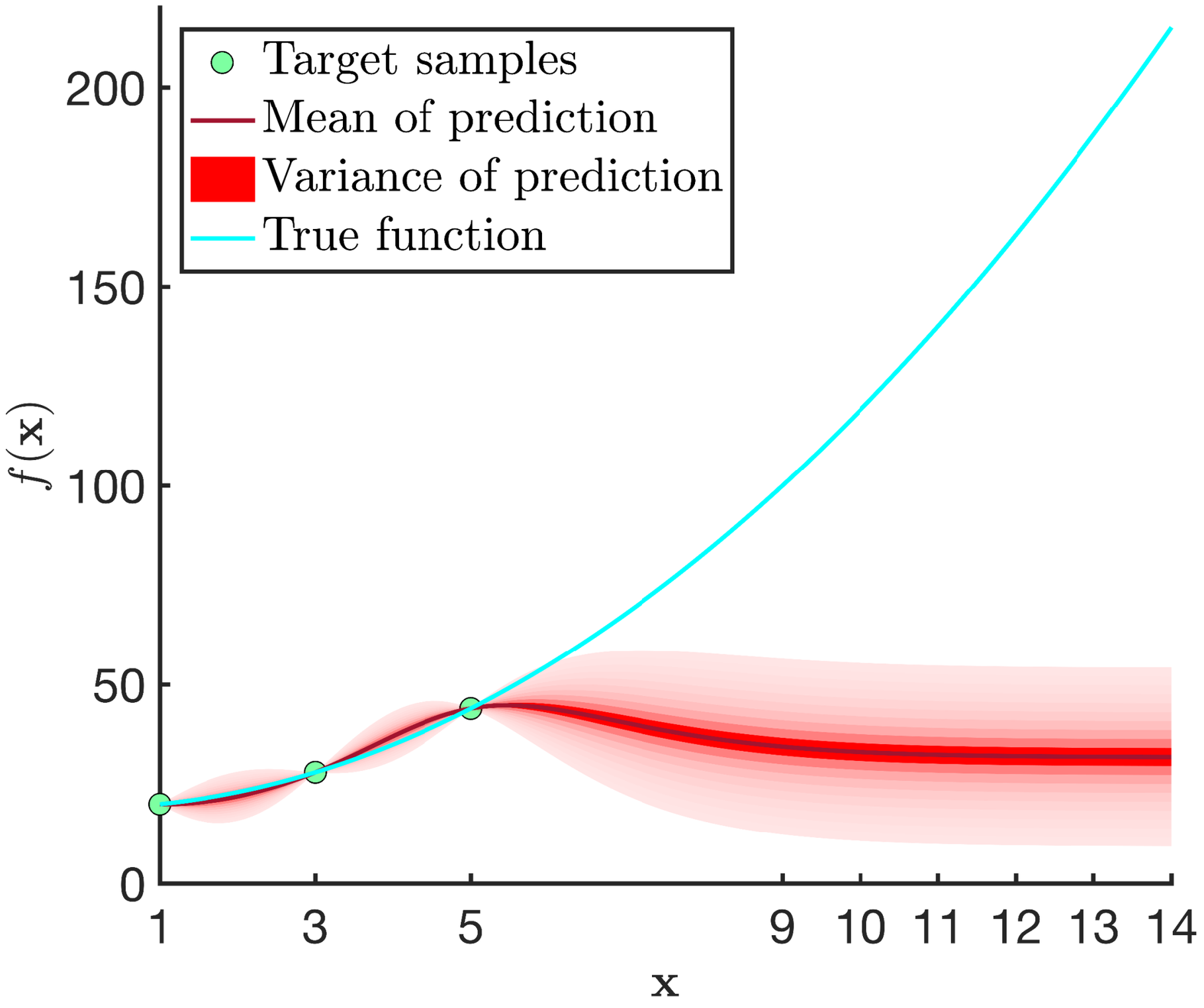
(0, 0) → (7, 12)

Exploiting Similarity

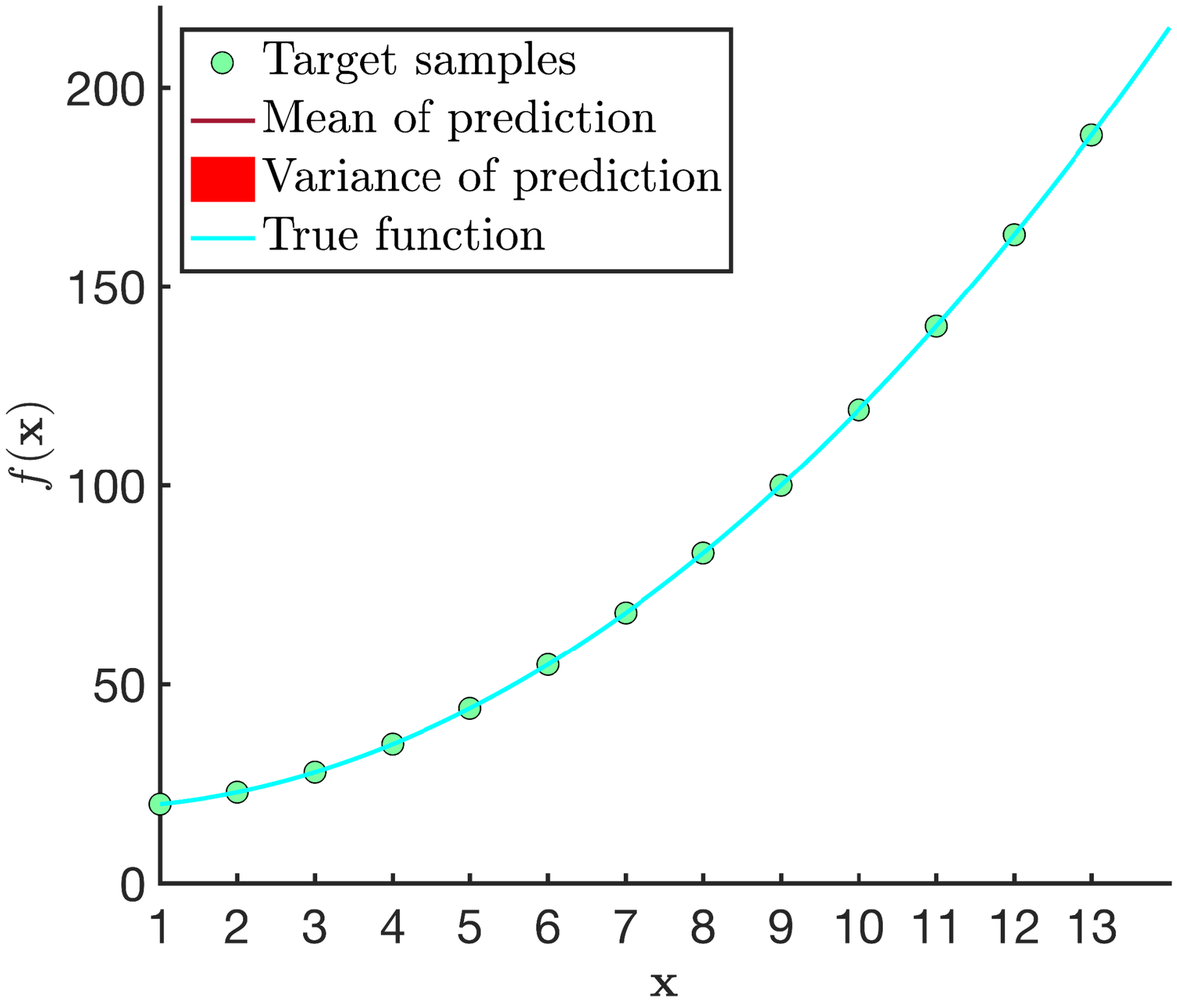
Function Prediction



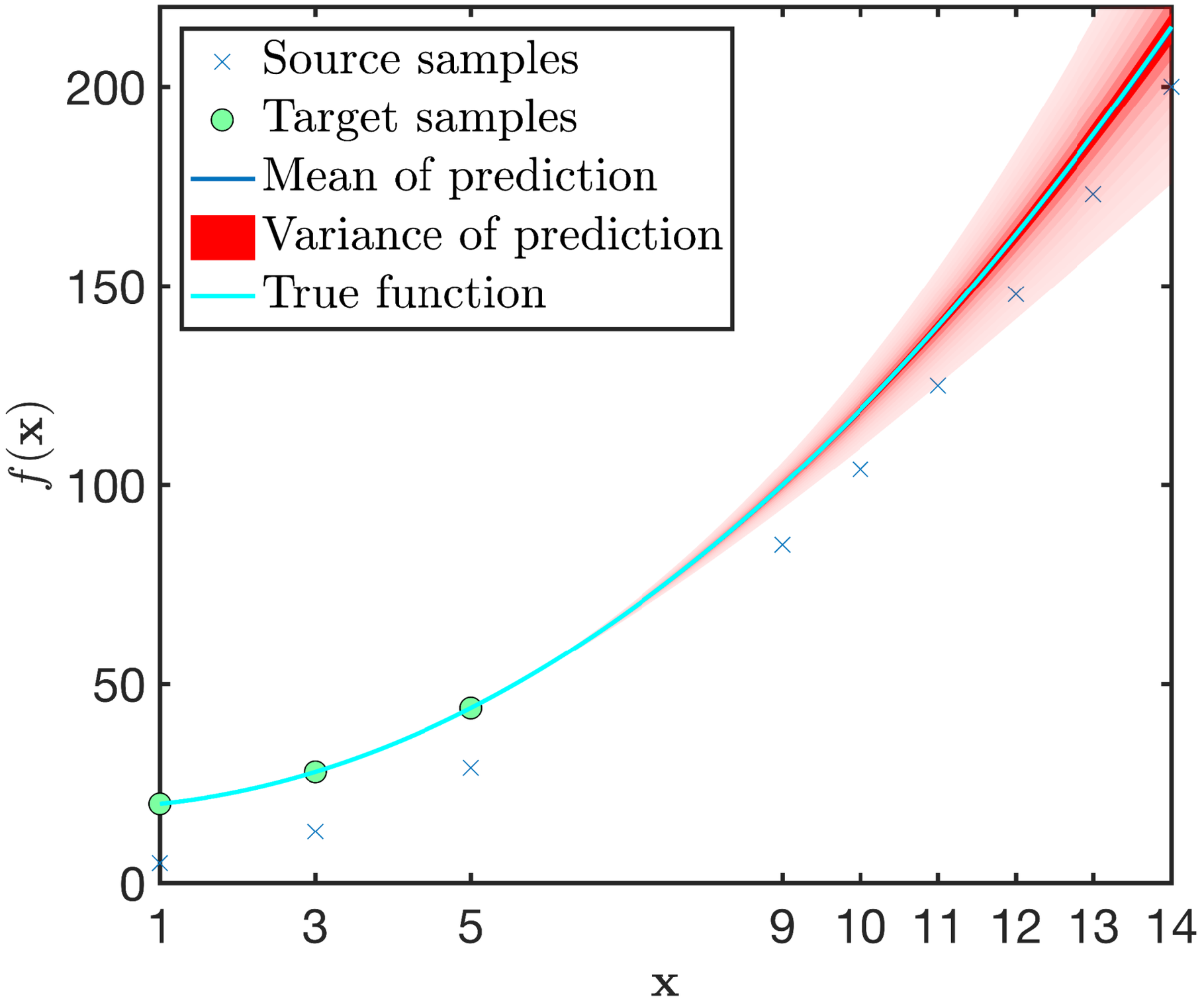
Prediction without Transfer Learning



Prediction with More Data but without Transfer Learning



Prediction with Transfer Learning



Technical Details

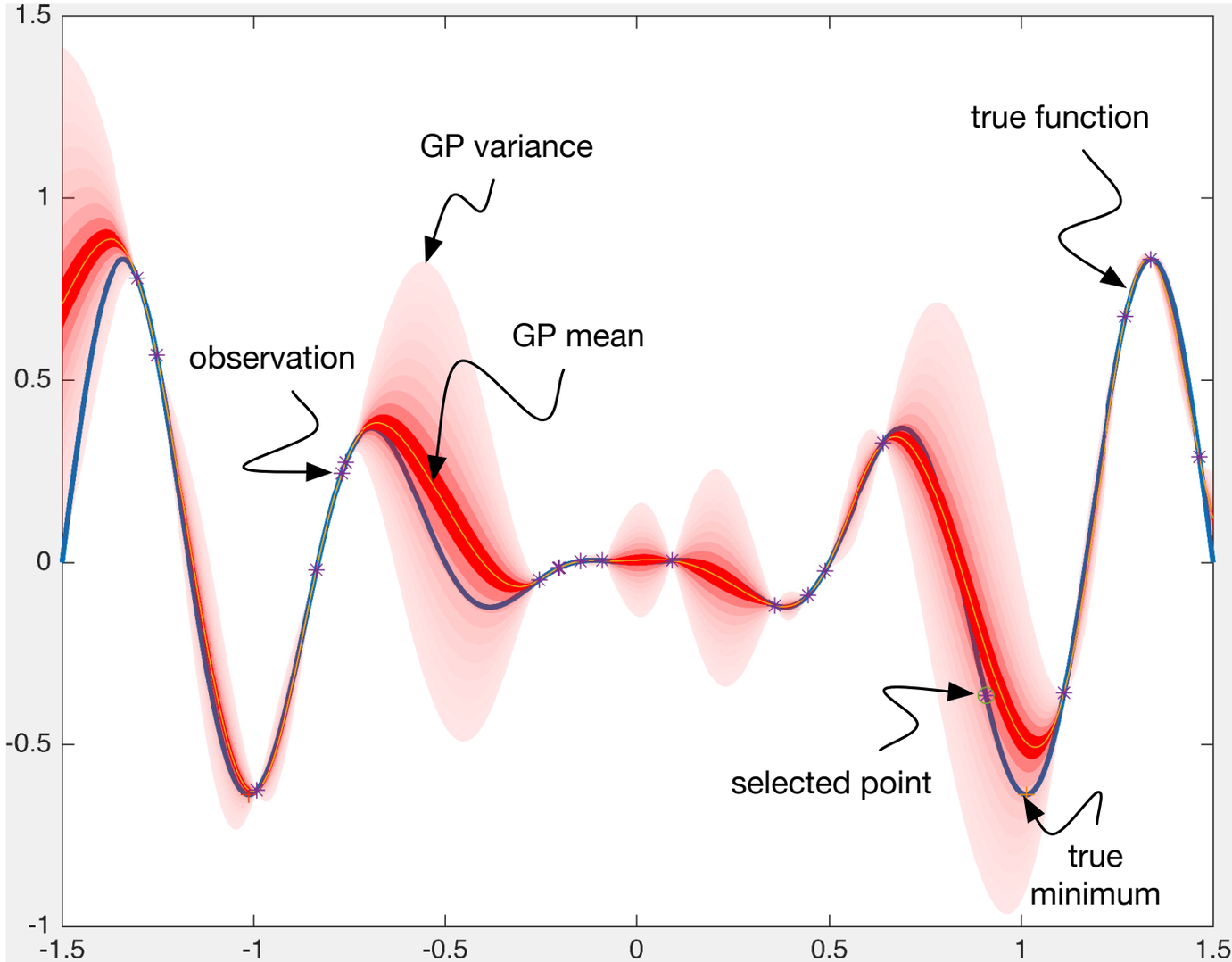
Prediction With Transfer Learning

$$y = f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')),$$

$$\mu_t(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \boldsymbol{k}(\boldsymbol{x})^\intercal (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \boldsymbol{\mu})$$
$$\sigma_t^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2 \boldsymbol{I} - \boldsymbol{k}(\boldsymbol{x})^\intercal (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}(\boldsymbol{x})$$

Motivations:

1. mean estimates + variance
2. all computation are linear algebra
3. good estimations with few data



$$\boldsymbol{K} := \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_t) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_t, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_t, \boldsymbol{x}_t) \end{bmatrix}$$

$$k(f, g, \boldsymbol{x}, \boldsymbol{x}') = k_t(f, g) \times k_{xx}(\boldsymbol{x}, \boldsymbol{x}'),$$

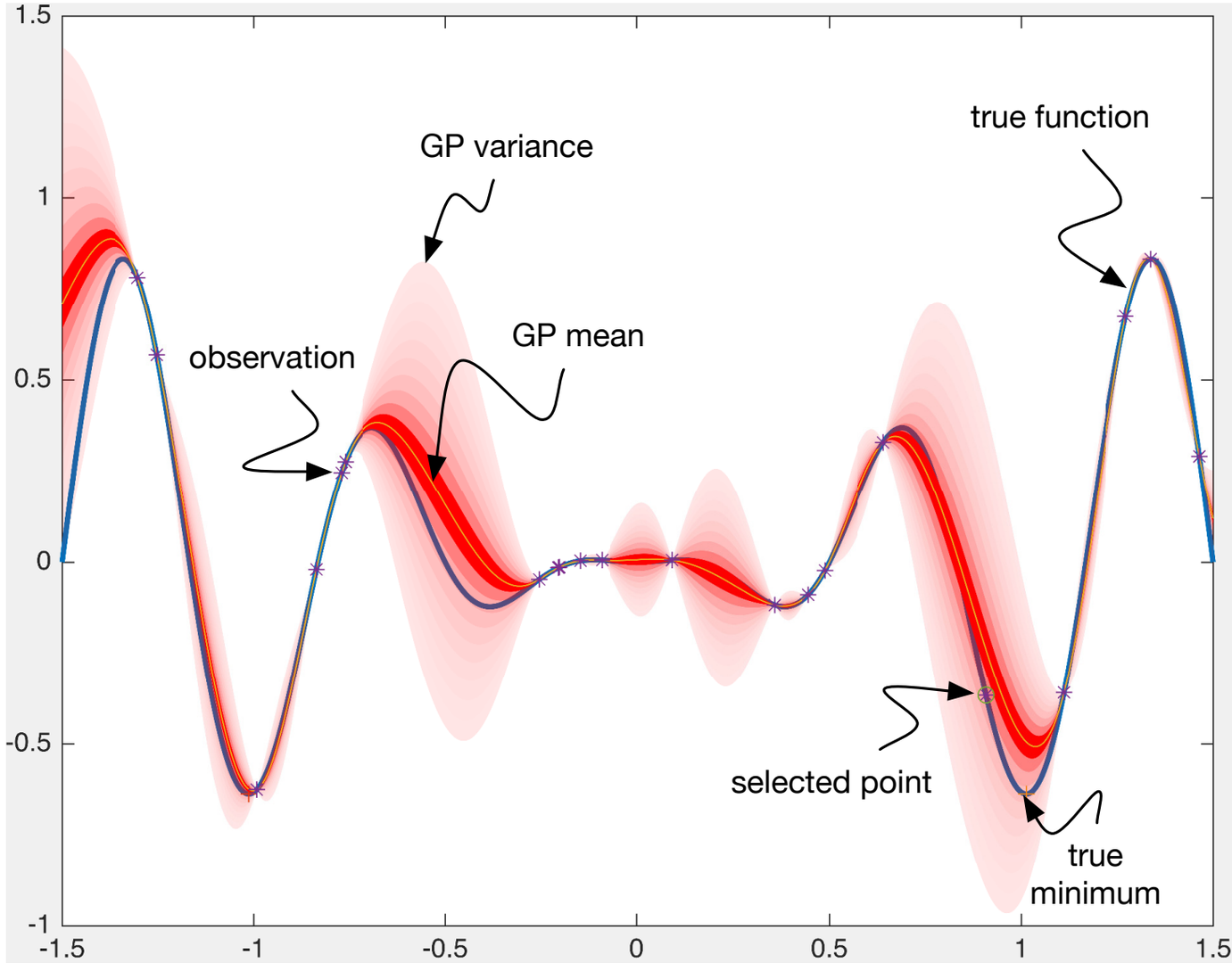
Prediction With Transfer Learning

$$y = f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')),$$

$$\mu_t(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \boldsymbol{k}(\boldsymbol{x})^\intercal (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \boldsymbol{\mu})$$
$$\sigma_t^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2 \boldsymbol{I} - \boldsymbol{k}(\boldsymbol{x})^\intercal (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}(\boldsymbol{x})$$

Motivations:

1. mean estimates + variance
2. all computation are linear algebra
3. good estimations when few data



$$\boldsymbol{K} := \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_t) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_t, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_t, \boldsymbol{x}_t) \end{bmatrix}$$

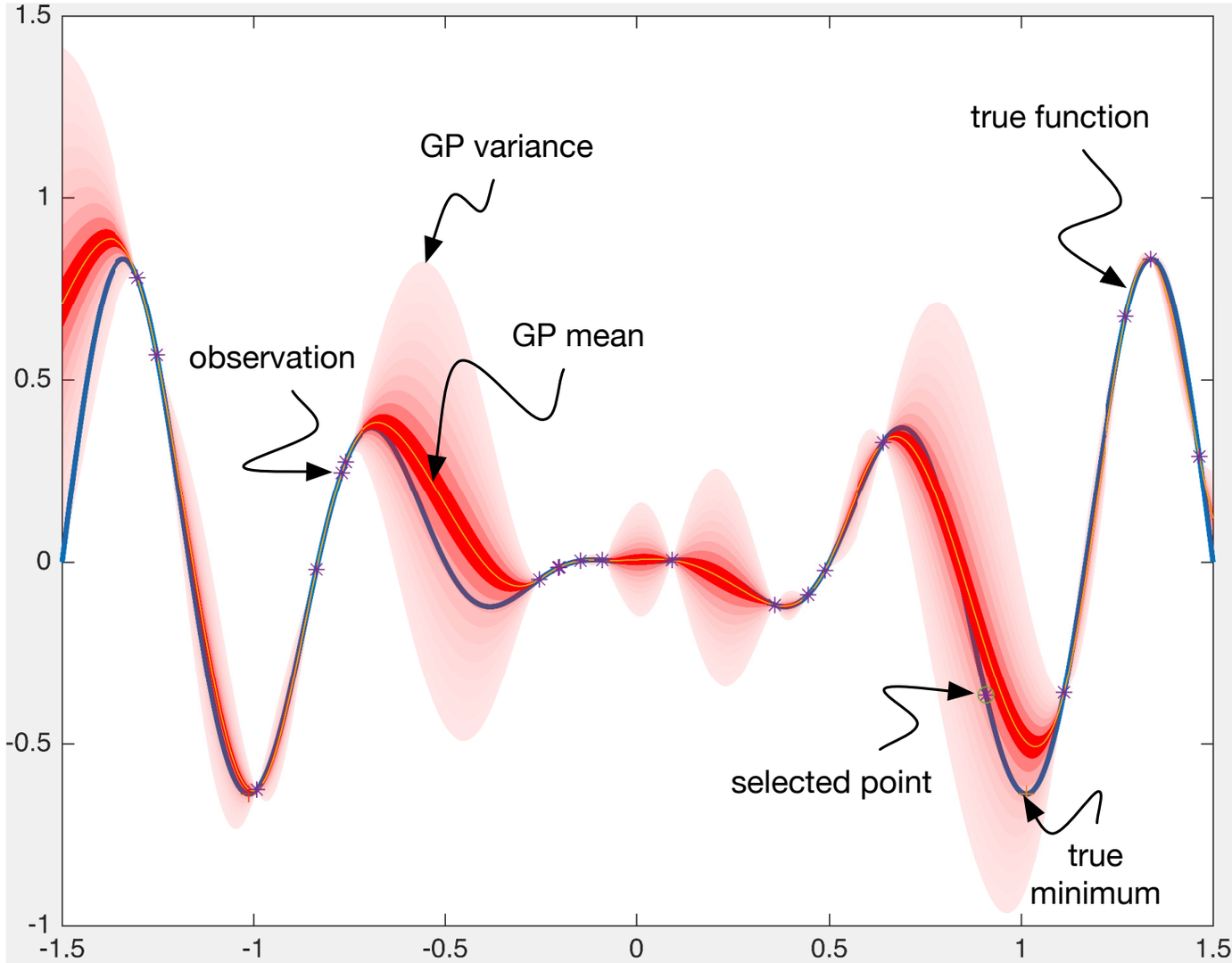
https://github.com/pooyanjamshidi/transferlearning

$$k(f, g, \boldsymbol{x}, \boldsymbol{x}') = k_t(f, g) \times k_{xx}(\boldsymbol{x}, \boldsymbol{x}'),$$

Prediction With Transfer Learning

$$y = f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')),$$

$$\mu_t(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \boldsymbol{k}(\boldsymbol{x})^\intercal (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \boldsymbol{\mu})$$
$$\sigma_t^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) + \sigma^2 \boldsymbol{I} - \boldsymbol{k}(\boldsymbol{x})^\intercal (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}(\boldsymbol{x})$$



Motivations:

1. mean estimates + variance
2. all computation are linear algebra
3. good estimations when few data

$$\boldsymbol{K} := \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_t) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_t, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_t, \boldsymbol{x}_t) \end{bmatrix}$$

https://github.com/pooyanjamshidi/transferlearning

$$k(f, g, \boldsymbol{x}, \boldsymbol{x}') = k_t(f, g) \times k_{xx}(\boldsymbol{x}, \boldsymbol{x}'),$$

pjsamshid@cs.cmu.edu

Evaluation

Case Study and Controlled Experiments

RQ1: Improve prediction accuracy?

RQ2: Tradeoffs among number of source and target samples?

RQ3: Fast enough for self-adaptive systems?

Analyzed Systems

Autonomous service robot

Environmental change

STORM

3 stream processing apps

Workload change

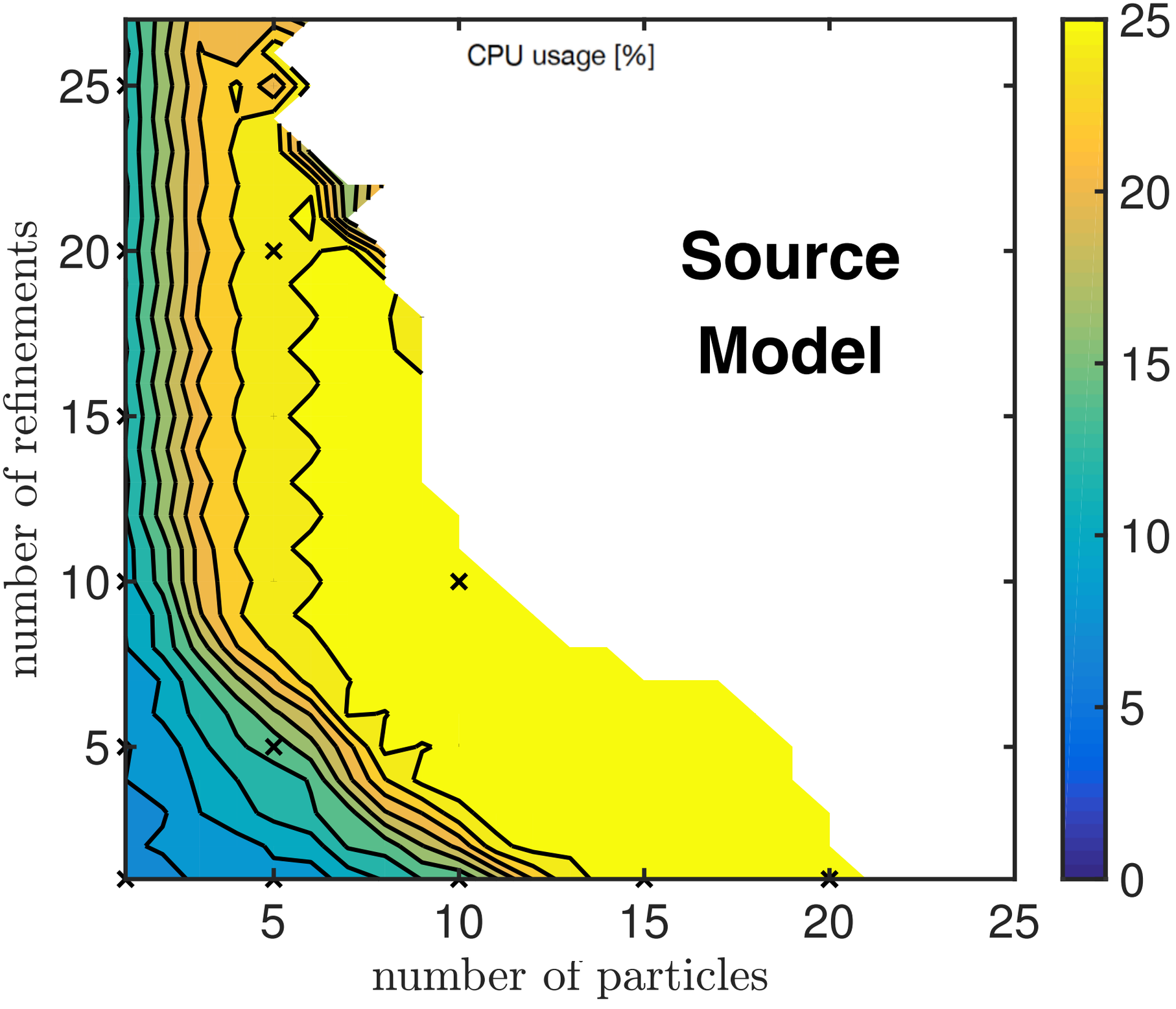
cassandra

NoSQL DBMS

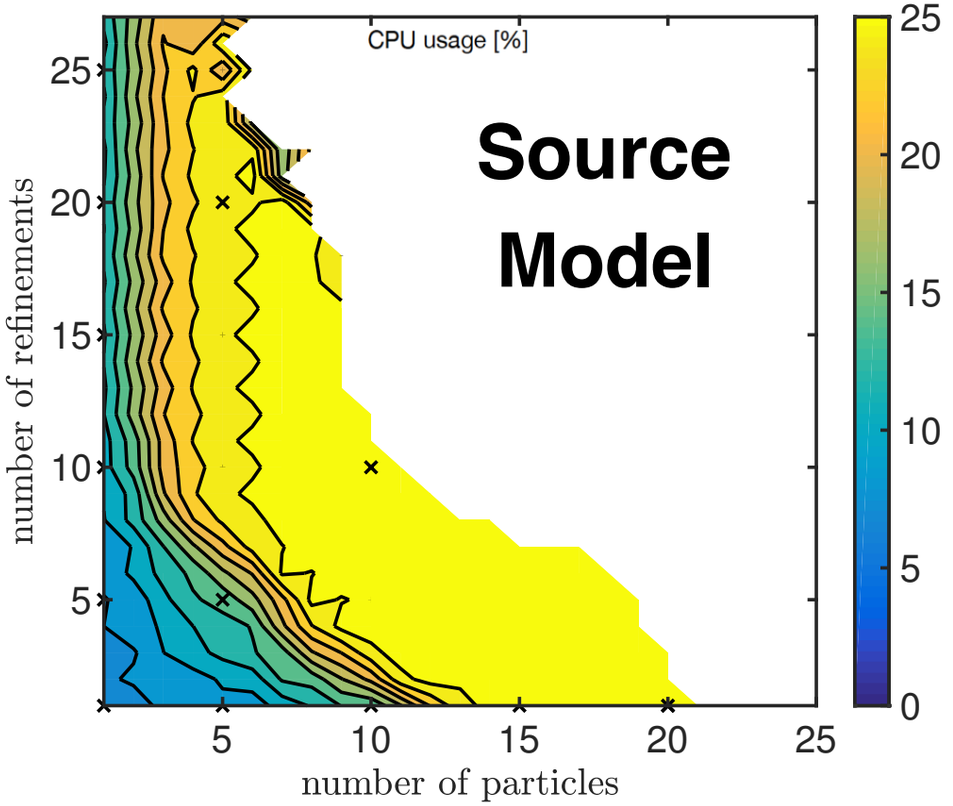
Workload & hardware changes

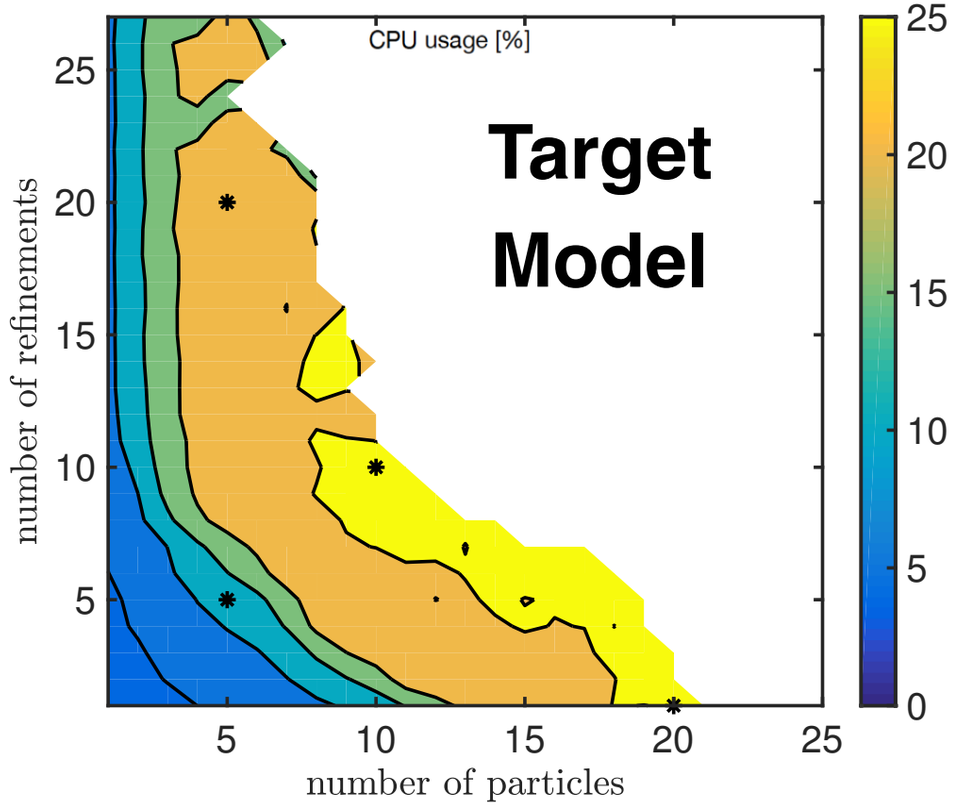
Prediction Accuracy

Performance Prediction for CoBot

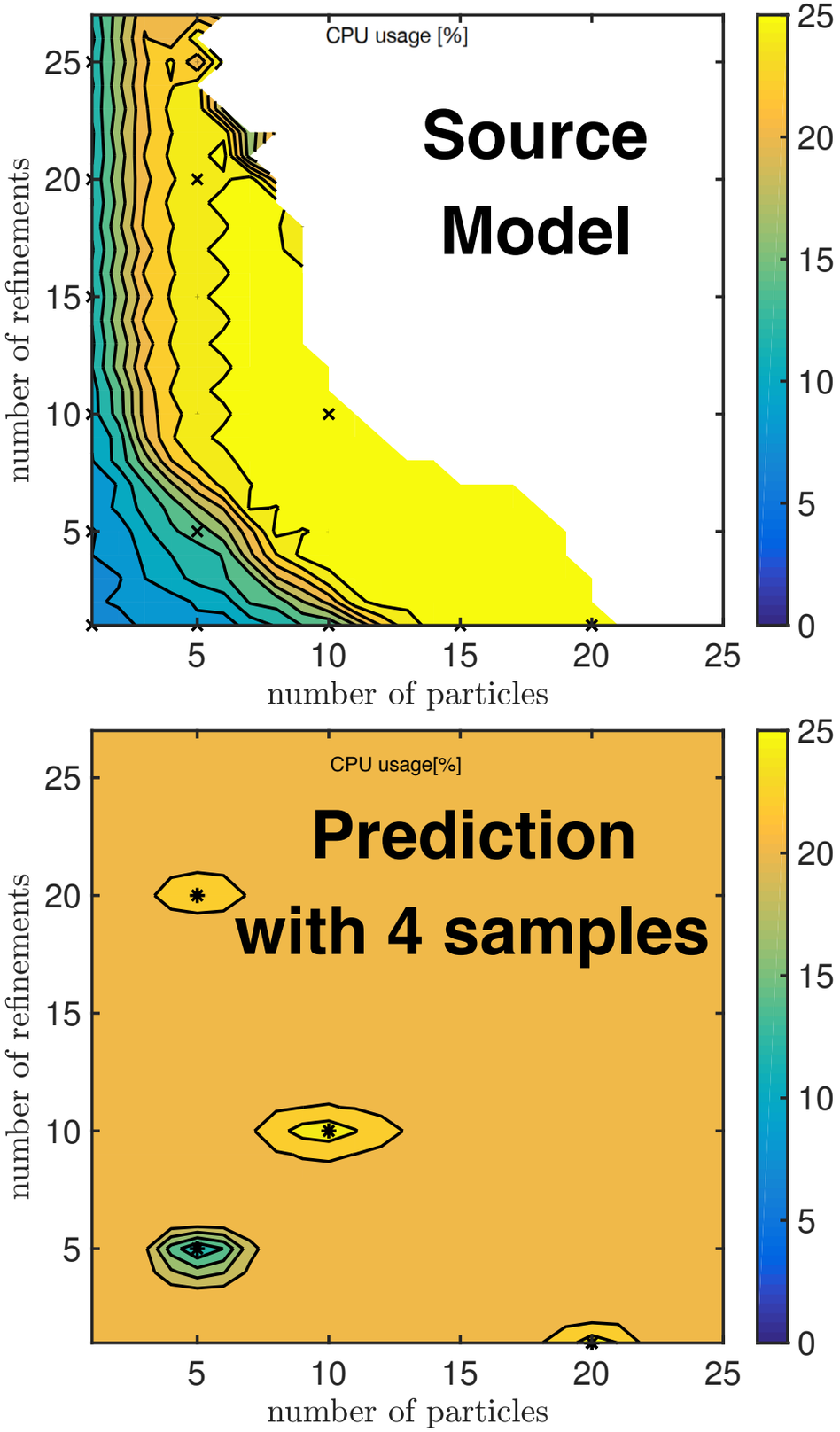


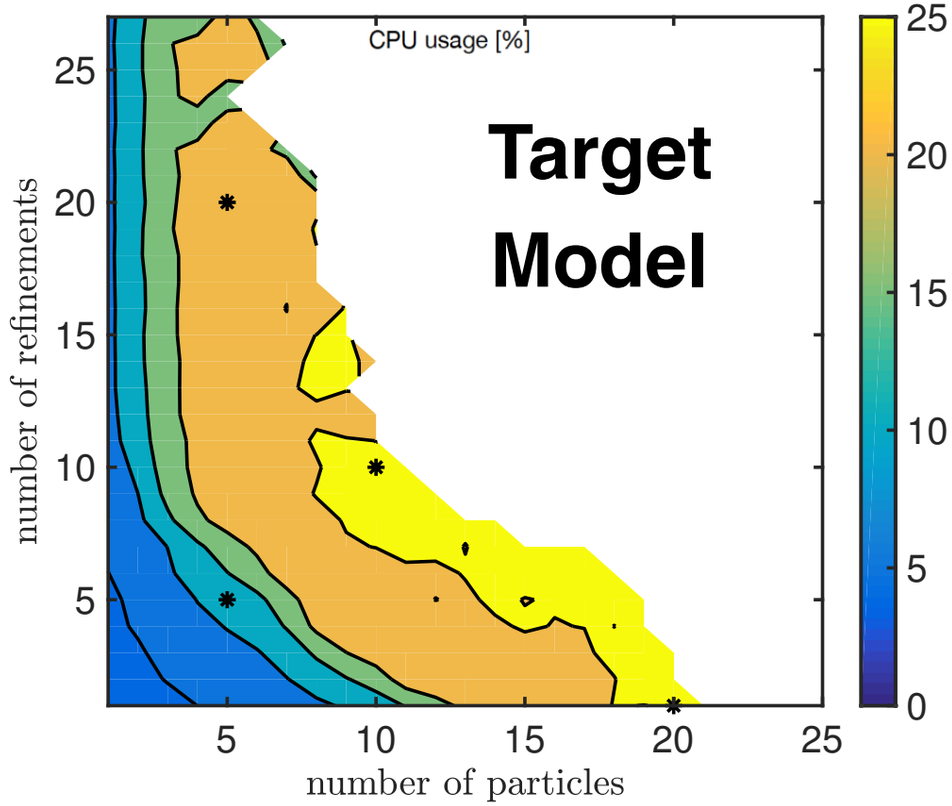
Performance Prediction for CoBot



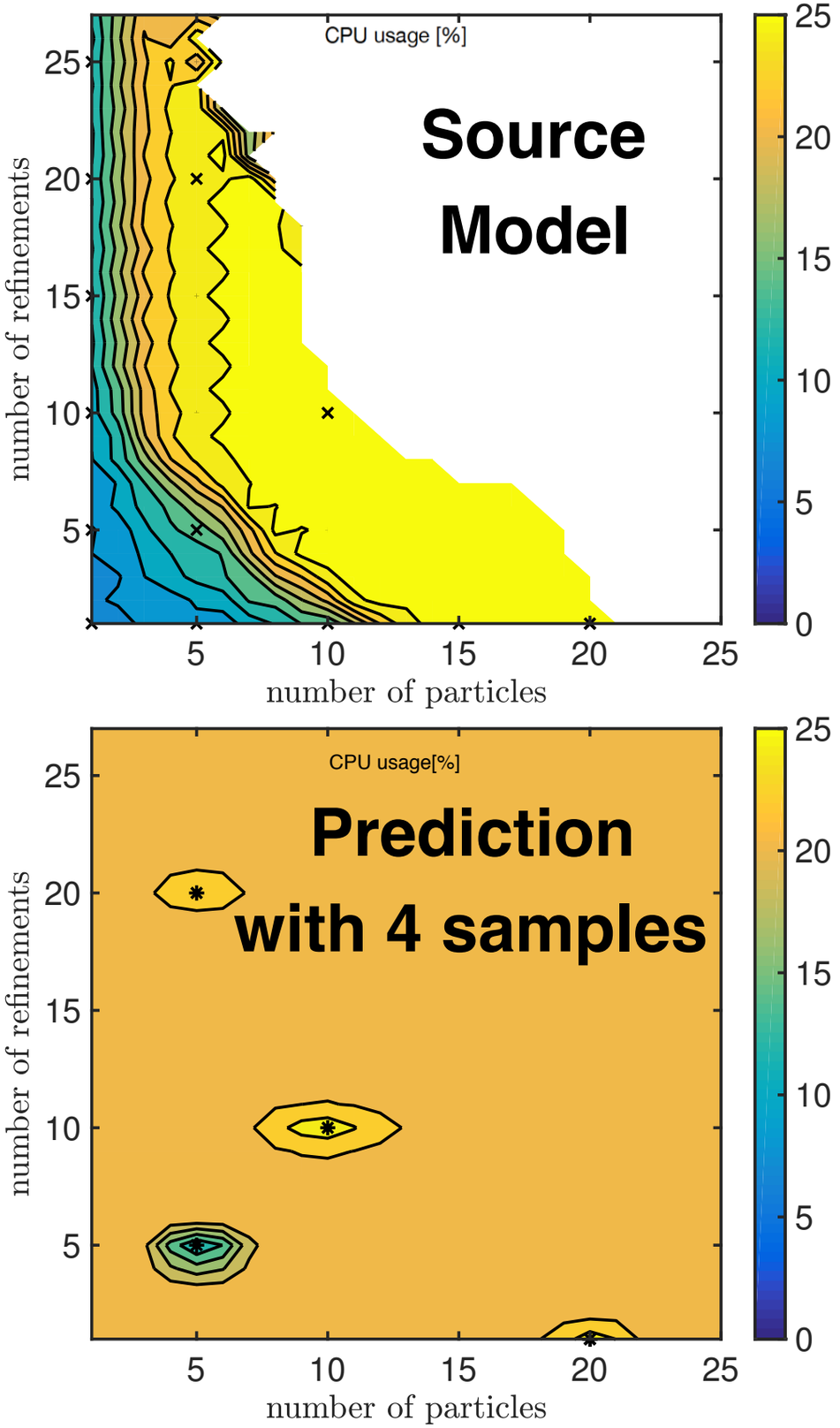


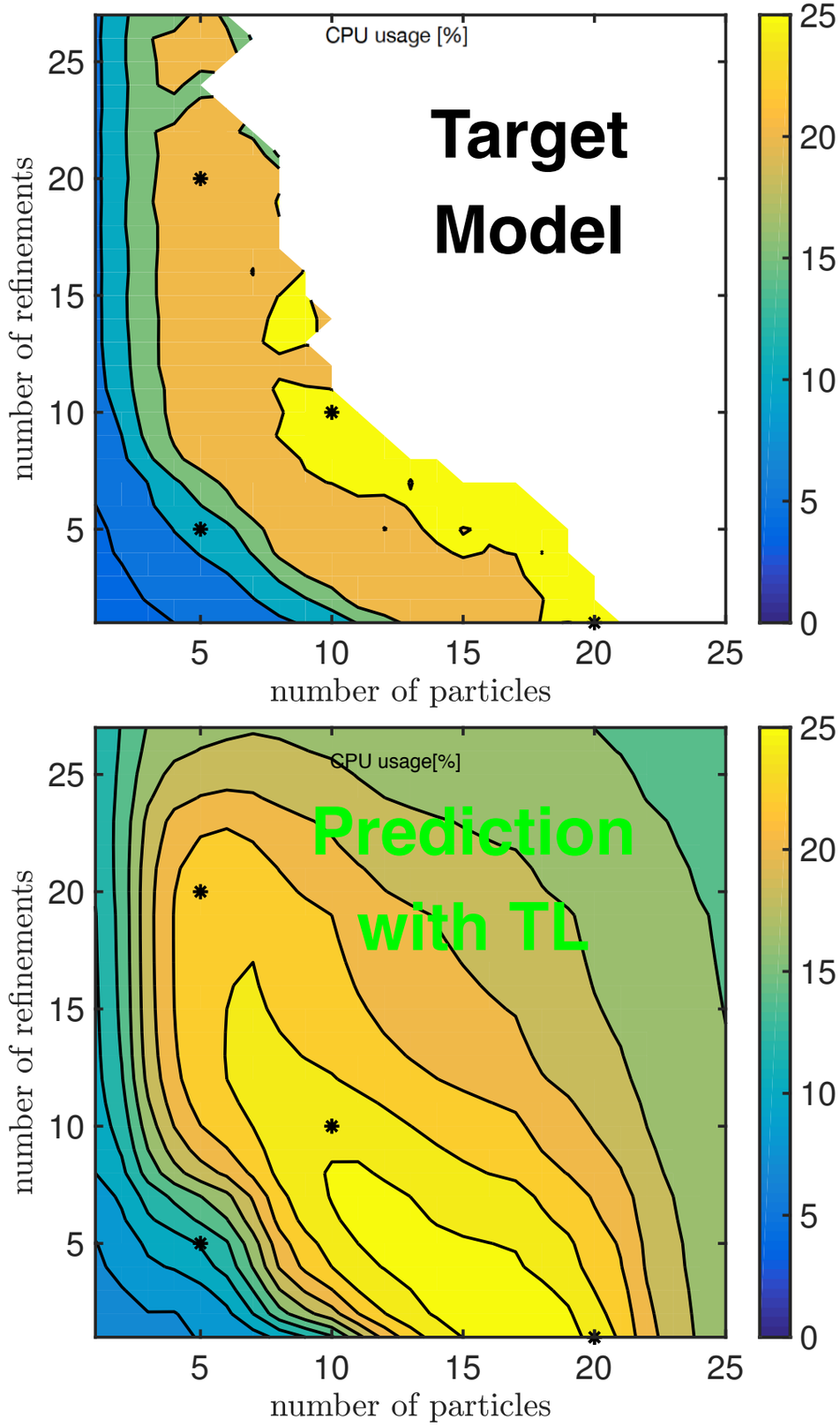
Performance Prediction for CoBot





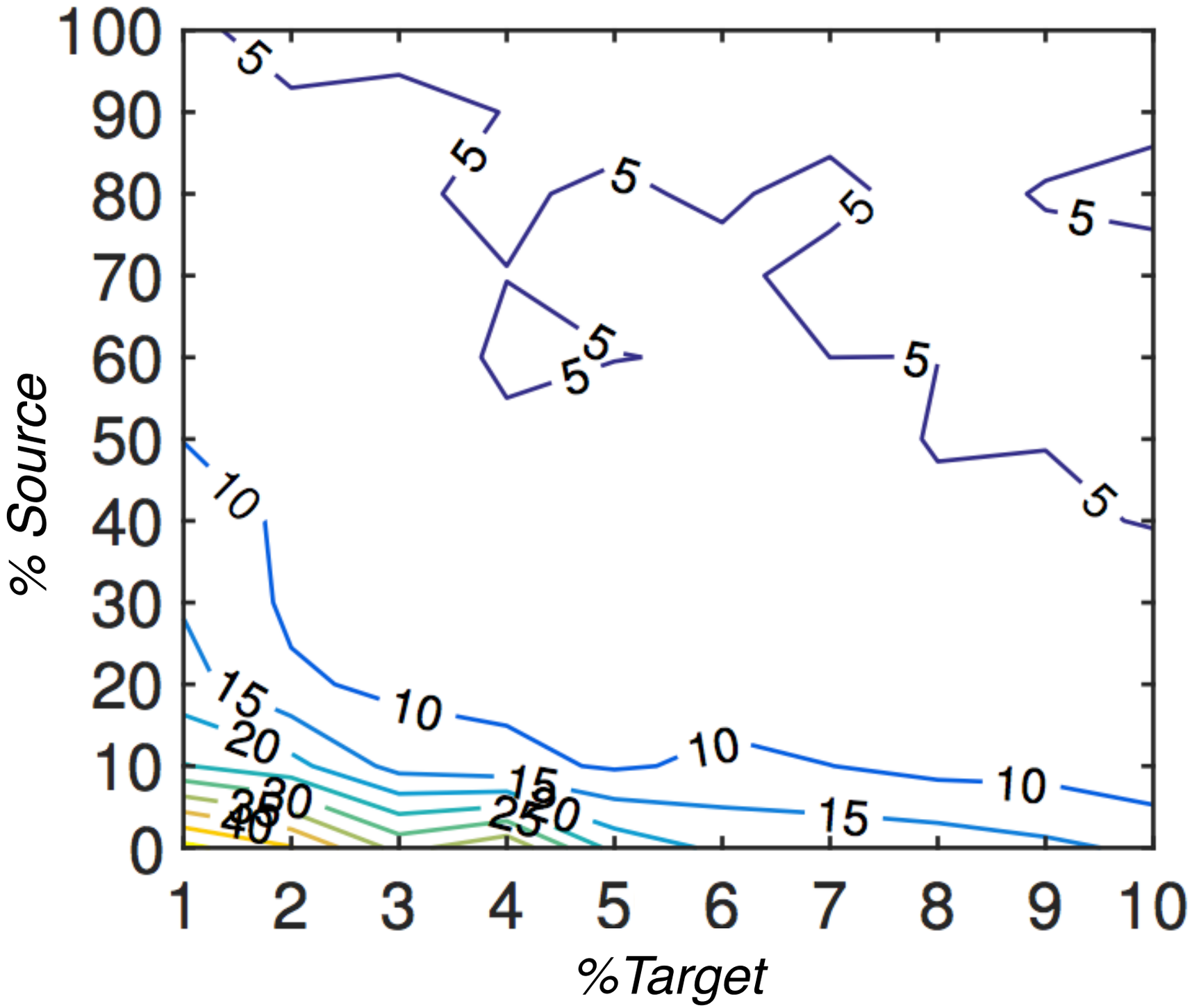
Performance Prediction for CoBot



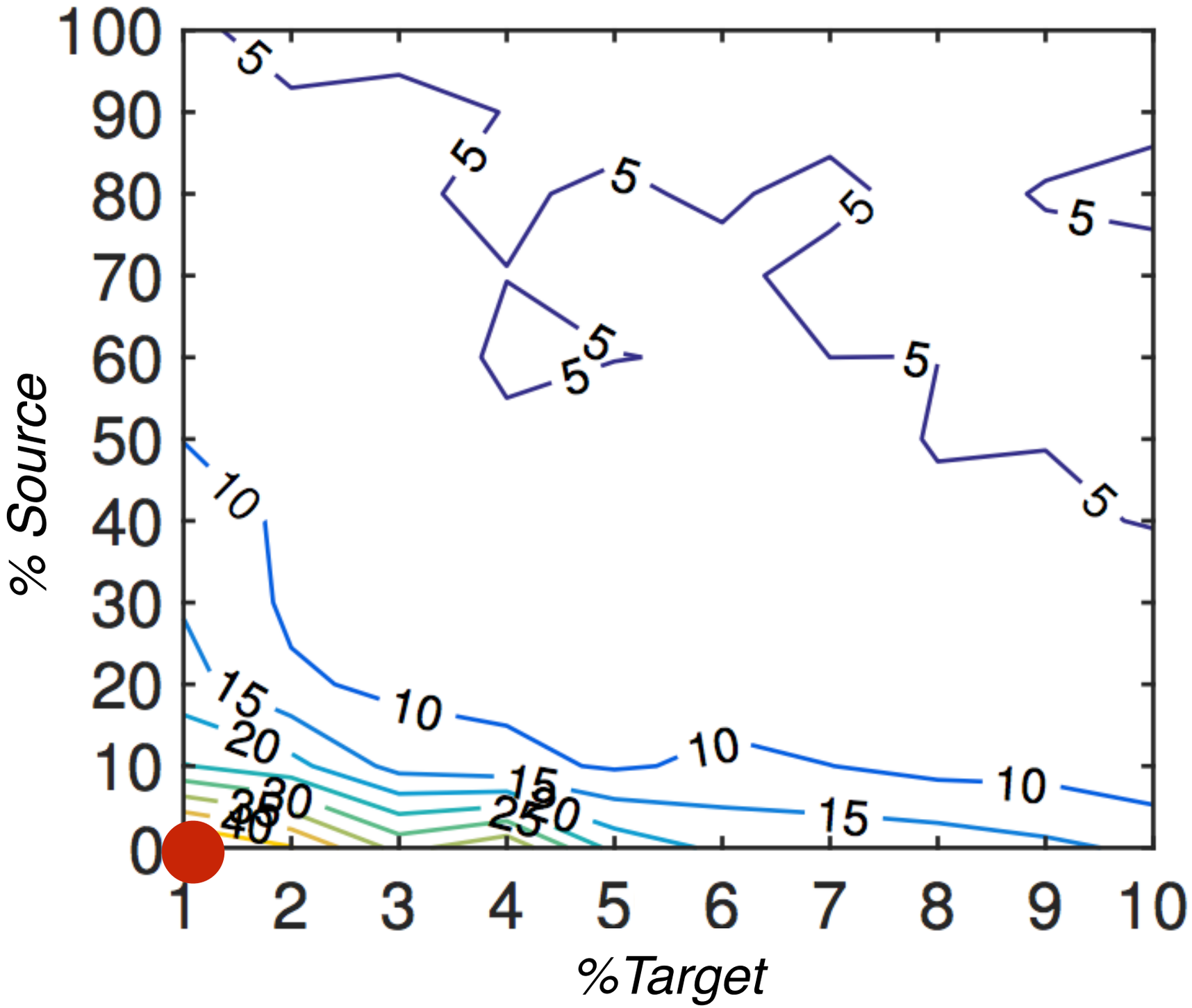


Number of Source and Target Samples

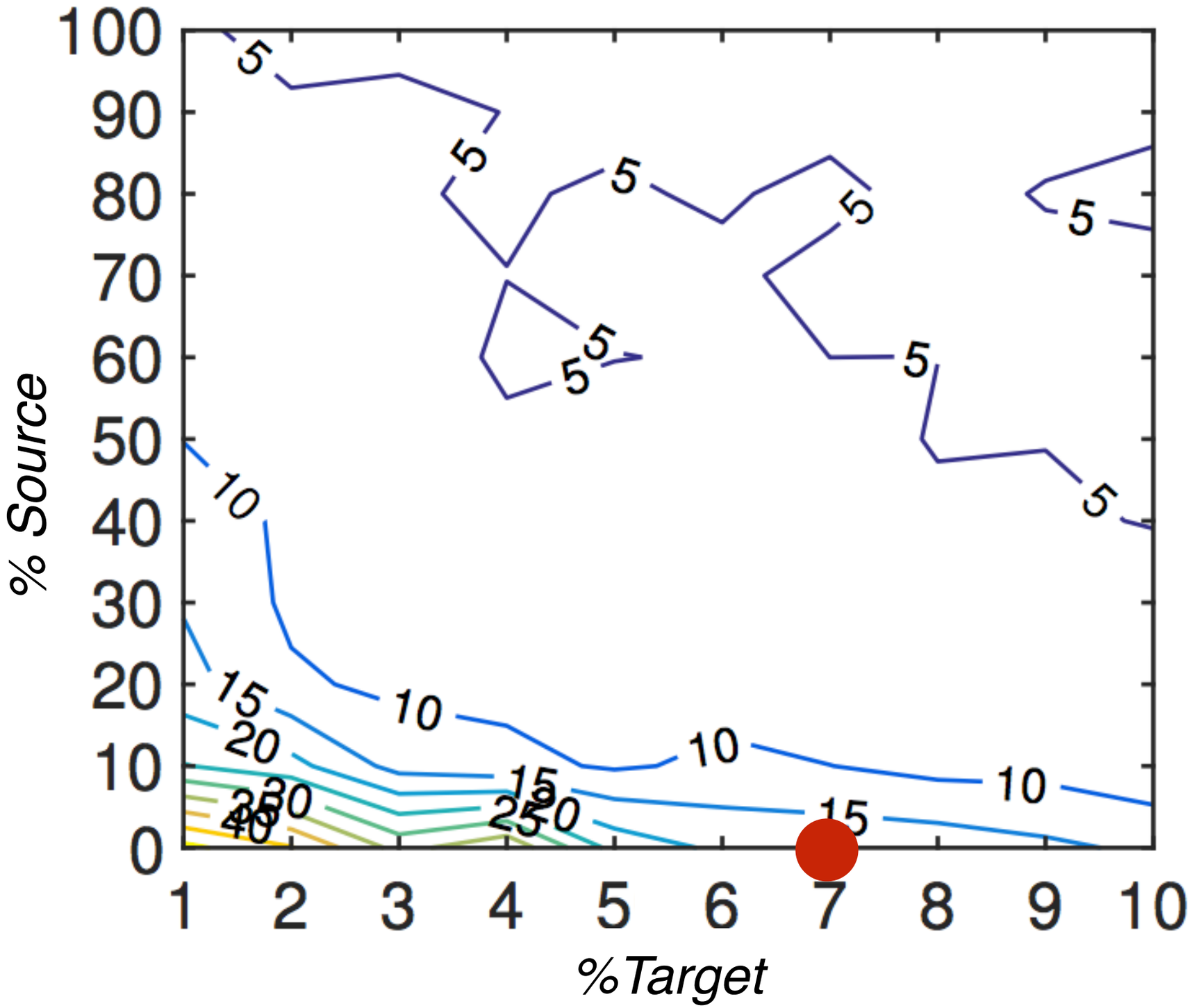
Prediction Error with Different Source and Target Samples



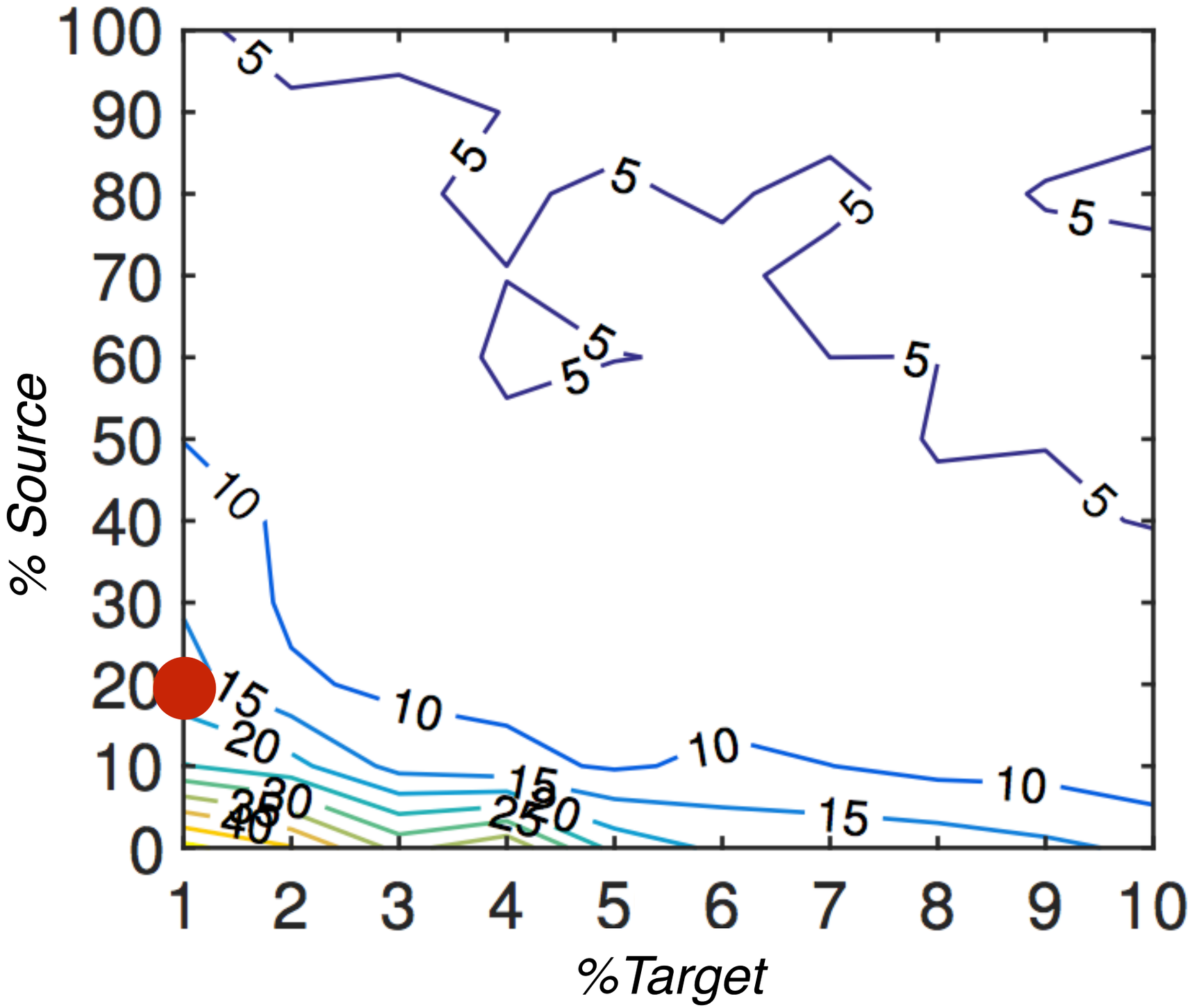
Prediction Error with Different Source and Target Samples



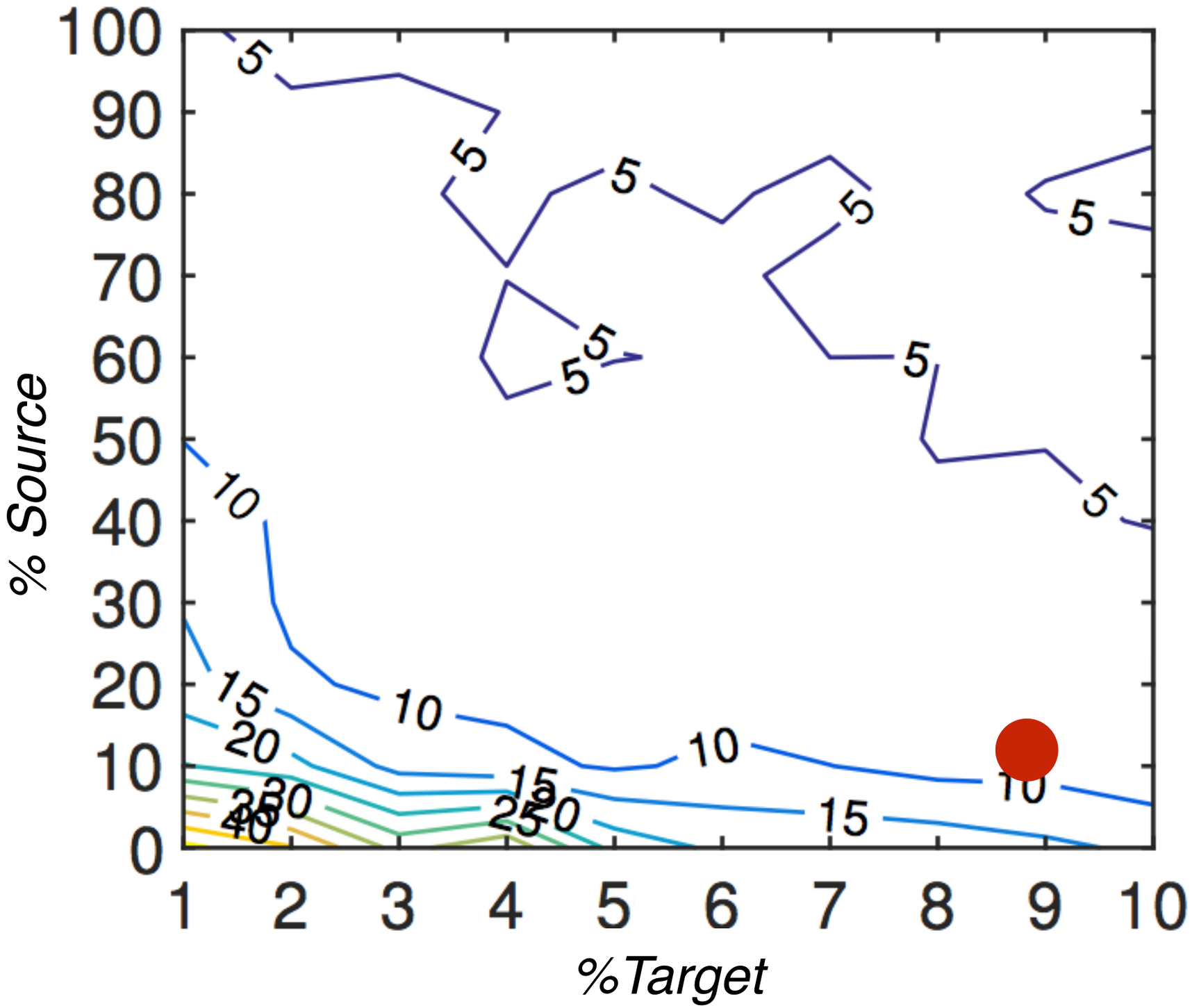
Prediction Error with Different Source and Target Samples



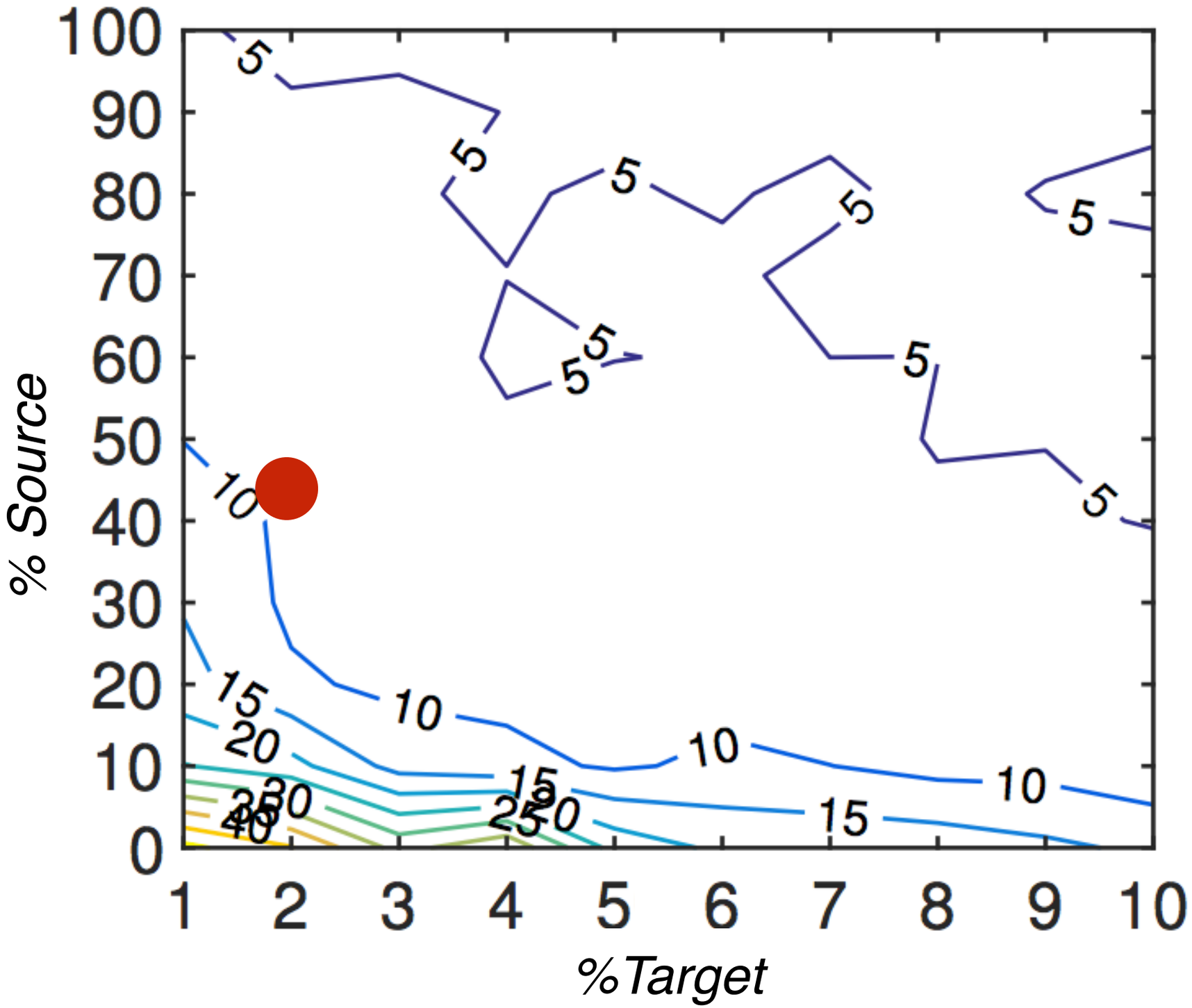
Prediction Error with Different Source and Target Samples



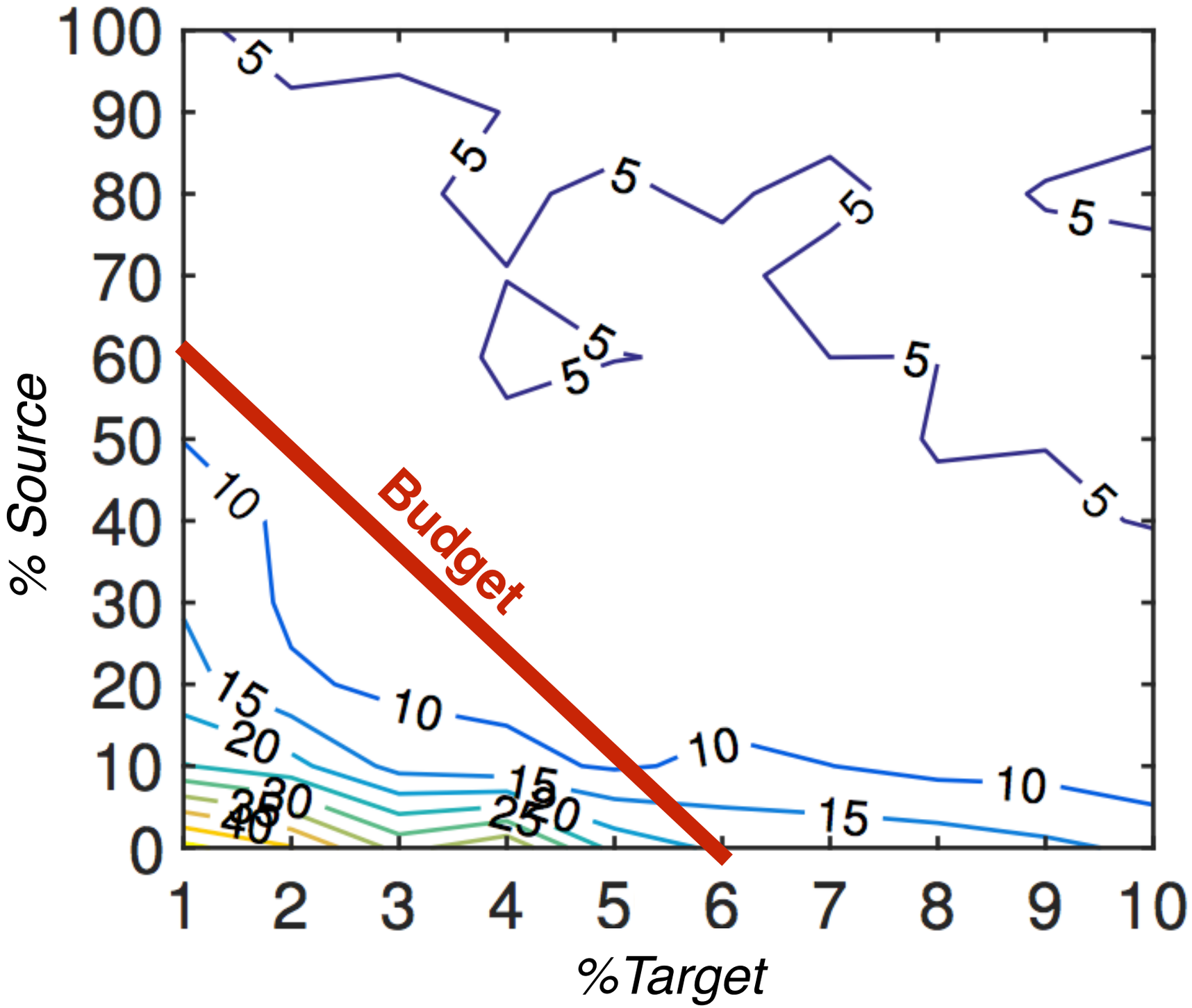
Prediction Error with Different Source and Target Samples



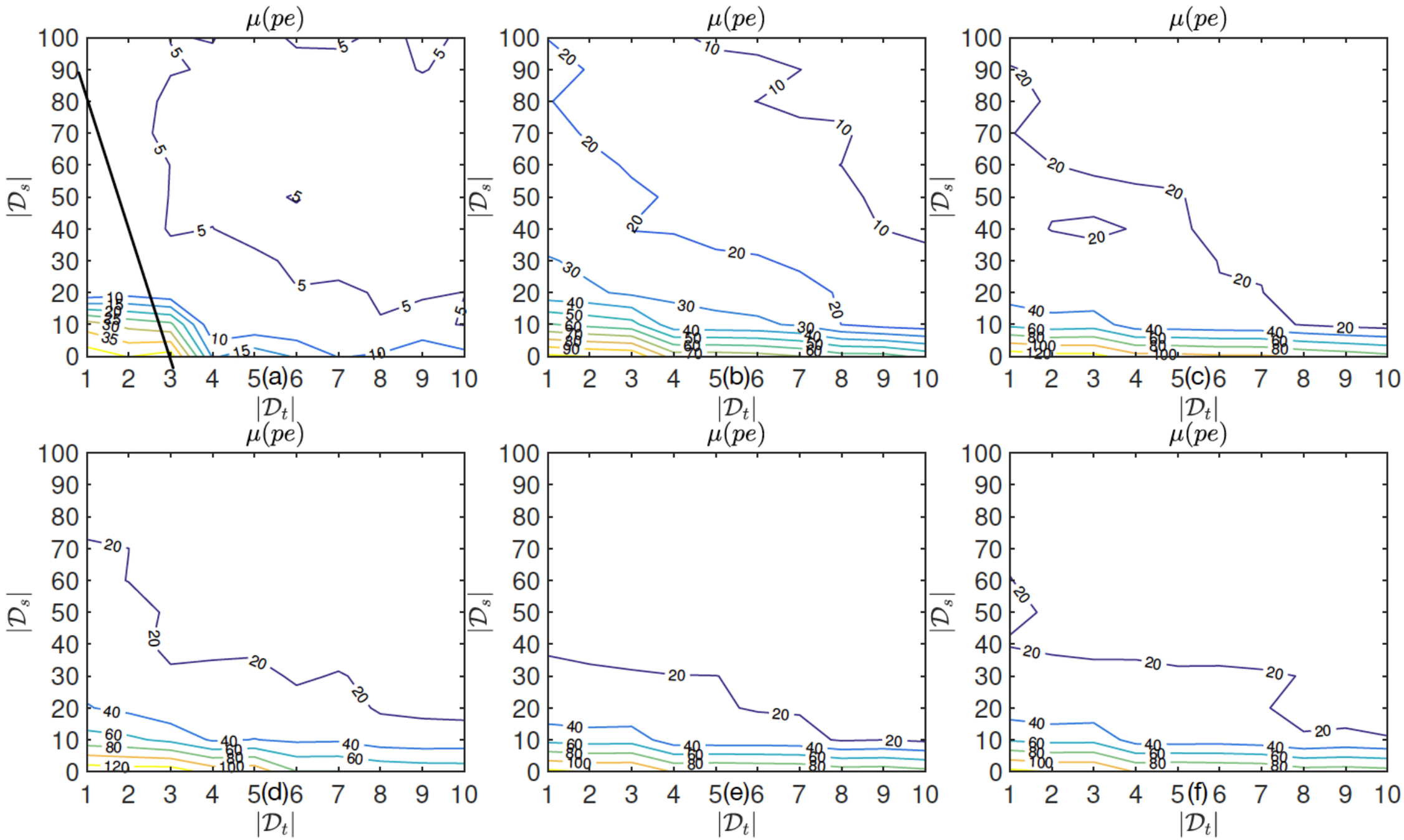
Prediction Error with Different Source and Target Samples



Accuracy and Costs



Prediction error of other systems



Applicable in Self-Adaptive Systems

Transfer Learning in Self-Adaptive Systems

Low model training overhead

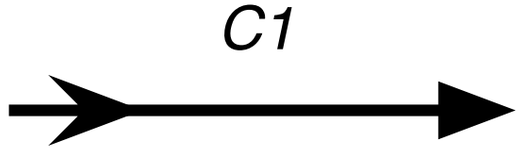
Produces accurate models

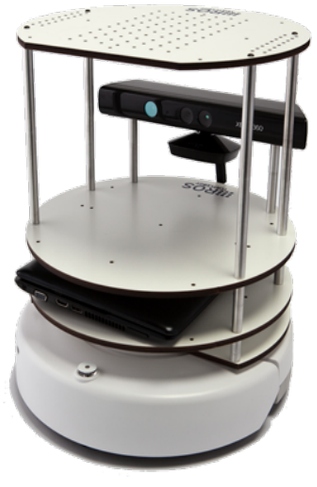
Models can improve at each adaptation cycle

Future Work, Insights, and Ideas

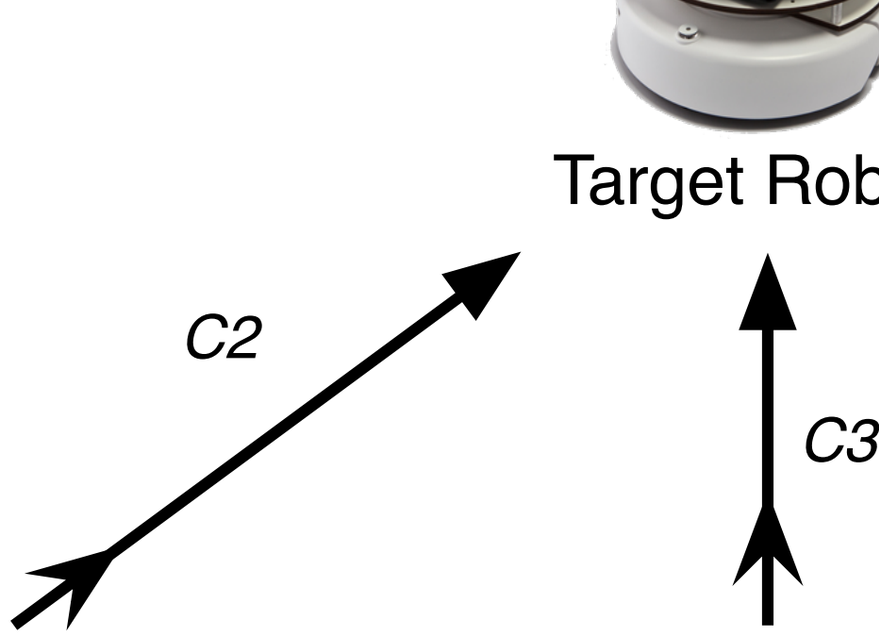
Selecting from Multiple Sources

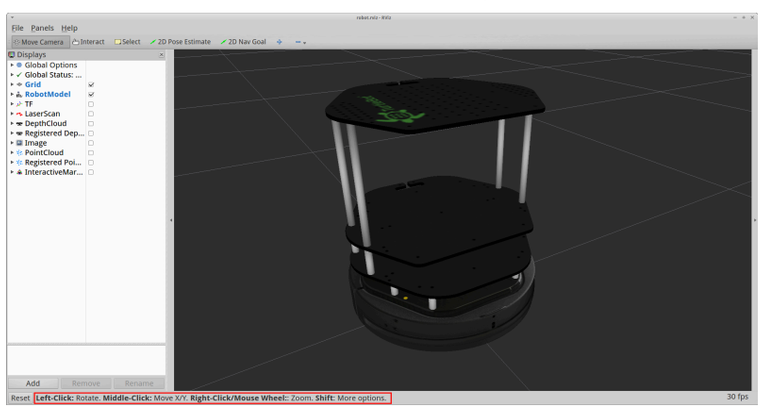
Source Robot



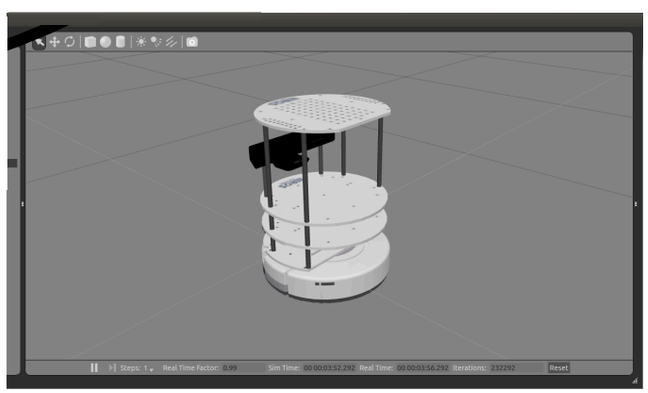


Target Robot





Source Simulator



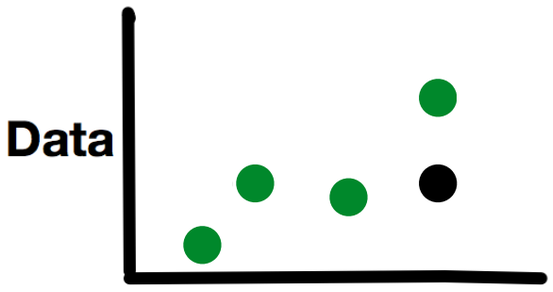
Target Simulator

Active Learning with Transfer Learning

TurtleBot

Iteratively find best sample points that maximize knowledge

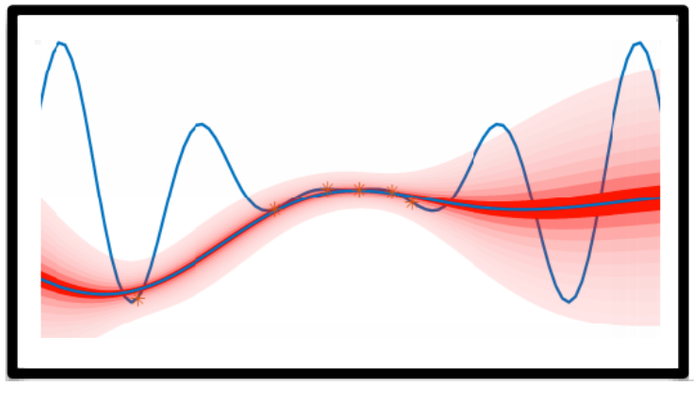
Measure



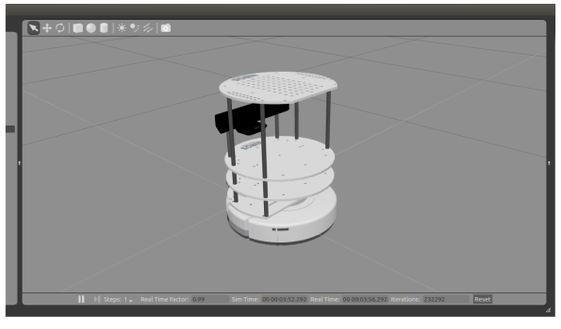
Reuse

Learn with TL

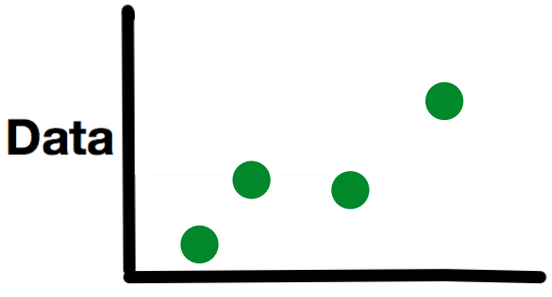
50 + 3*C1 + 15*C2 - 7*C2 *C3



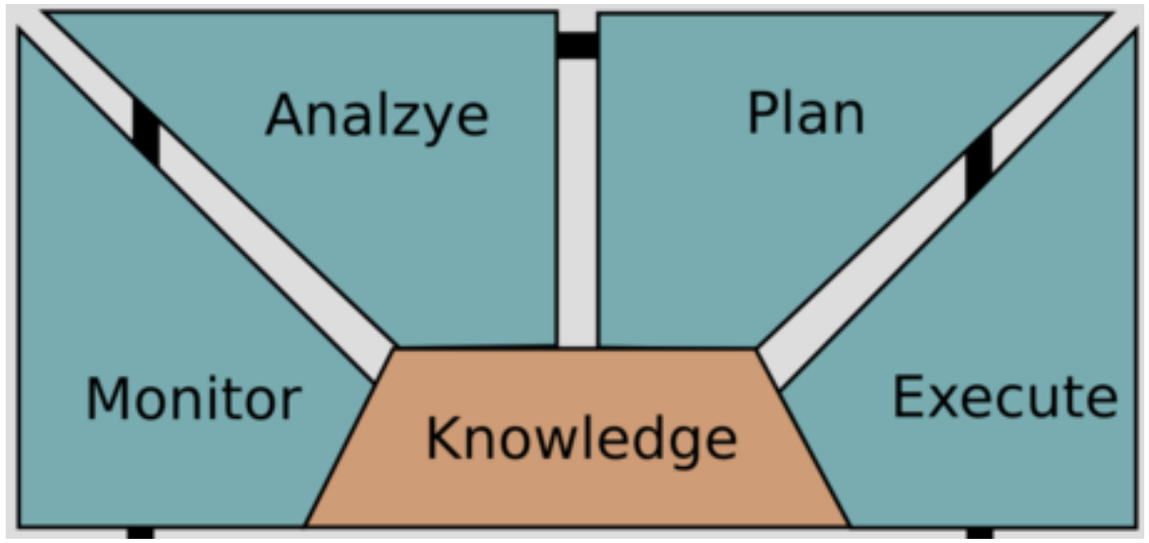
Simulator (Gazebo)



Measure



Integrating Transfer Learning in MAPE-K



Contribute to Knowledge

Assist in self-optimization

Support online learning

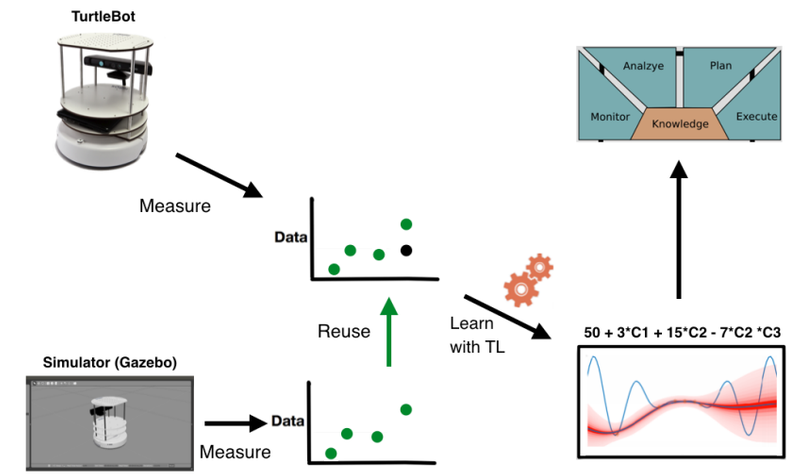
Summary

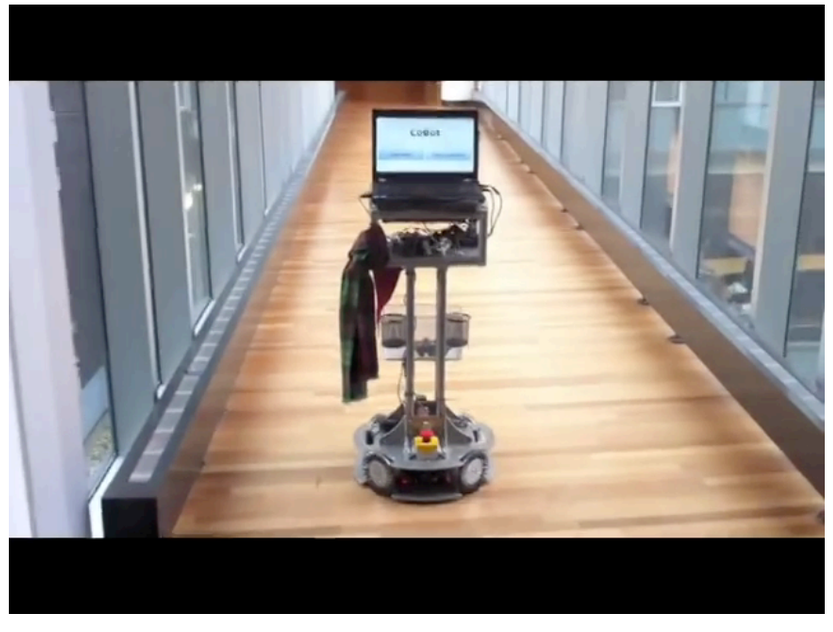
Transfer Learning for Improving Model Prediction in Highly Configurable Software

Reuse data from similar system

Improves model accuracy

Applicable in self-adaptive systems





Carnegie Mellon University

Transfer Learning Improves Sampling