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
{pjamshidi, 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: In order to change a system when it is running, systems need to be highly changeable. Typical software systems are not designed to be highly configurable. Newly selected configurations need to be configured in place. This requires a high degree of learning and adaptation.
Highly Configurable Systems In Dynamic Environments

[Credit to CMU CoBot]
Options Influence Performance
Adapt to Different Environments

TurtleBot
Adapt to Different Environments

TurtleBot

\[ 50 + 3C_1 + 15C_2 - 7C_2 \times C_3 \]
Adapt to Different Environments

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

TurtleBot

Diagram showing process flow with sections for Analyze, Plan, Monitor, and Execute.
Classic Sensitivity Analysis

TurtleBot

50 + 3*C1 + 15*C2 - 7*C2 *C3
Classic Sensitivity Analysis

TurtleBot

Measure

Data

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

50 + 3*C1 + 15*C2 - 7*C2 *C3
Classic Sensitivity Analysis

TurtleBot

Measure

Data

Learn

$50 + 3C_1 + 15C_2 - 7C_2 \cdot C_3$

50 + 3*C1 + 15*C2 - 7*C2 *C3
Classic Sensitivity Analysis

TurtleBot

Measure

Data

Learn

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

TurtleBot

Measure

Data

Learn

Optimization
Adaptation +
Reasoning +
Debugging +

50 + 3*C1 + 15*C2 - 7*C2 *C3
Measuring Performance is Expensive

25 options \times 10 \text{ values} = \mathbf{10^{25}} \text{ configurations}

[Credit to Peng Hou]
Transfer Learning
Reuse Data From Similar System

TurtleBot

Measure

Data
Reuse Data From Similar System

TurtleBot

Measure

Data

Simulator (Gazebo)
Reuse Data From Similar System

TurtleBot

Measure

Simulator (Gazebo)

Measure

Data
Reuse Data From Similar System

TurtleBot

Measure

Simulator (Gazebo)

Data

Reuse

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

Data

Reuse

Learn with TL

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

Simulator (Gazebo)

Measure

Data
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

Target function

Target samples

\( f(x) \)
Prediction with More Data but without Transfer Learning

- **Target samples**
- **Mean of prediction**
- **Variance of prediction**
- **True function**

The graph shows a linear relationship between $f(x)$ and $x$ with increasing values as $x$ increases from 1 to 13.
Prediction with Transfer Learning

Source samples
Target samples
Mean of prediction
Variance of prediction
True function

\( f(x) \)
Technical Details
Prediction With Transfer Learning

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

\[
\mu_t(x) = \mu(x) + k(x)^T(K + \sigma^2 I)^{-1}(y - \mu),
\]

\[
\sigma^2_t(x) = k(x, x) + \sigma^2 I - k(x)^T(K + \sigma^2 I)^{-1}k(x)
\]

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

\[
K := \begin{bmatrix}
k(x_1, x_1) & \ldots & k(x_1, x_t) \\
\vdots & \ddots & \vdots \\
k(x_t, x_1) & \ldots & k(x_t, x_t)
\end{bmatrix}
\]

\[
k(f, g, x, x') = k_t(f, g) \times k_{xx}(x, x'),
\]
Prediction With Transfer Learning

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

\[ \mu_t(x) = \mu(x) + k(x)^T(K + \sigma^2 I)^{-1}(y - \mu) \]
\[ \sigma^2_t(x) = k(x, x) + \sigma^2 I - k(x)^T(K + \sigma^2 I)^{-1}k(x) \]

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

\[ K := \begin{bmatrix} k(x_1, x_1) & \ldots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \ldots & k(x_t, x_t) \end{bmatrix} \]

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

https://github.com/pooyanjamshidi/transferlearning
Prediction With Transfer Learning

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

\[
\begin{align*}
\mu_t(x) &= \mu(x) + k(x)^T(K + \sigma^2 I)^{-1}(y - \mu) \\
\sigma_t^2(x) &= k(x, x) + \sigma^2 I - k(x)^T(K + \sigma^2 I)^{-1}k(x)
\end{align*}
\]

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

\[ K := \begin{bmatrix}
k(x_1, x_1) & \ldots & k(x_1, x_t) \\

\vdots & \ddots & \vdots \\

k(x_t, x_1) & \ldots & k(x_t, x_t)
\end{bmatrix} \]

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

https://github.com/pooyanjamshidi/transferlearning

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
- 3 stream processing apps
- Workload change
- NoSQL DBMS
- Workload & hardware changes
Prediction Accuracy
Performance Prediction for CoBot

Source Model
Performance Prediction for CoBot

Source Model

Target Model
Performance Prediction for CoBot

Source Model

Target Model

Prediction with 4 samples
Performance Prediction for CoBot

Source Model

Target Model

Prediction with 4 samples

Prediction with TL
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

% Source vs % Target

Budget line

Contours represent different values:
- 10
- 15
- 20
- 25
- 30
- 40
- 50
- 60
- 70
- 80
- 90
- 100

Graph indicates the relationship between budget constraints and accuracy targets.
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

C1

Target Robot

C2

C3

Source Simulator

Target Simulator
Active Learning with Transfer Learning

Iteratively find best sample points that maximize knowledge

TurtleBot

Measure

Reuse

Learn with TL

Simulator (Gazebo)

Measure

50 + 3*C1 + 15*C2 - 7*C2 *C3
Integrating Transfer Learning in MAPE-K

Contribute to Knowledge

Assist in self-optimization

Support online learning
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