

Transferring Multi-device Localization Models using Latent Multi-task Learning

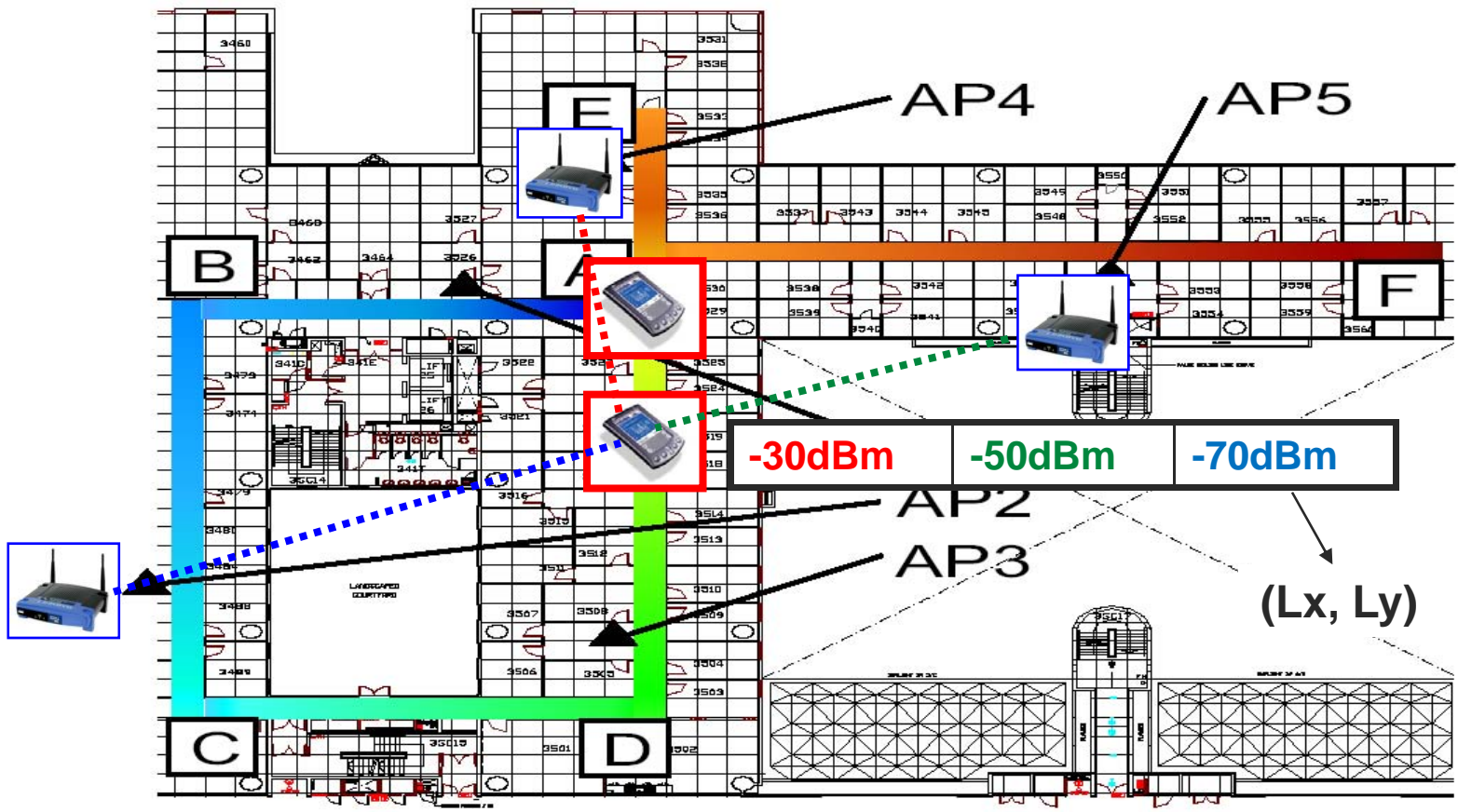
Vincent W. Zheng, Sinno J. Pan, Qiang Yang, Jeffrey J. Pan
Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Present in the 23rd AAAI Conference
on Artificial Intelligence (AAAI-08).
Chicago, Illinois, USA. July 2008.

<http://www.cse.ust.hk/~vincenz>



Signal-strength-based Localization



Learning-based Location Estimation

- Two phases: **training** and **localization**
- **Training phase** – collect a set of signal-location pairs and learn a mapping function

(AP1, AP2, AP3)	(Lx, Ly)
(-30, -50, -70) dBm	(2, 7) m
(-40, -45, -75) dBm	(3, 11) m
.....	...

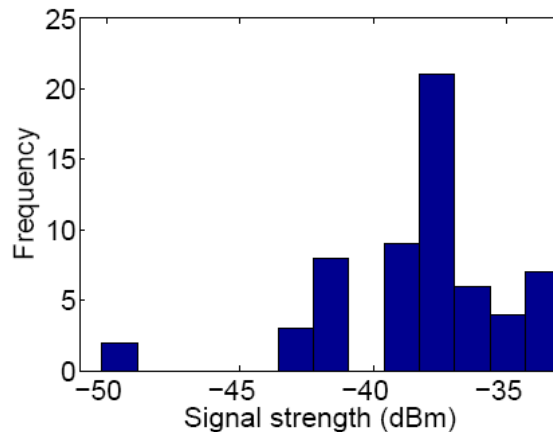


Mapping function
from signal to location
 $F : S \rightarrow L$

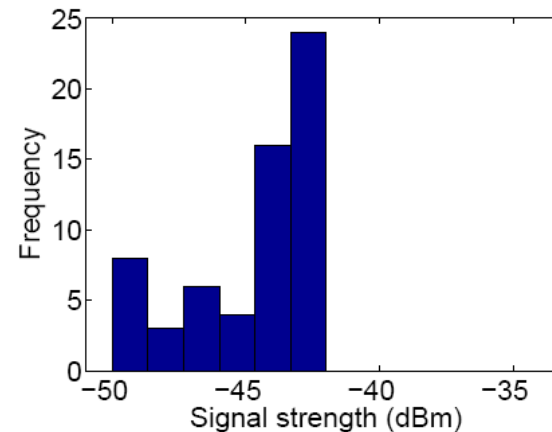
- **Localization phase** – given a signal **S**, predict its most probable location $L = F(S)$

Our focus: signal variation over device

- Traditional localization methods assume:
 - Training and testing data have **same** distributions
 - Use **same (or, at least very similar)** device in training-testing
- What if using different devices?
 - **Different** signal sensing capacities, **different** signal distributions



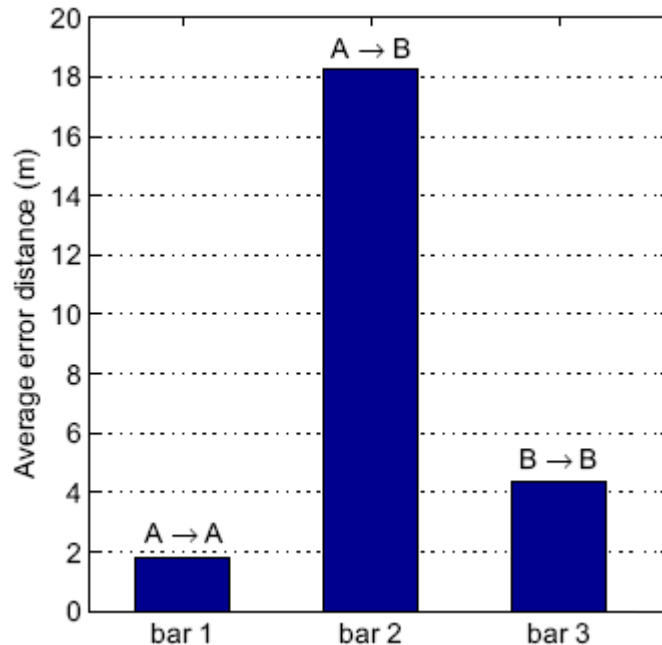
(a) WiFi signal at device 1



(b) WiFi signal at device 2

Motivation: Why transferring across devices?

- Necessity for considering multi-device problem



Device A (**Source device**)

Device B (**Target device**)

“ $X \rightarrow Y$ ”:

X for training, Y for testing

Measurement:

Average error distance* (unit: meter)

* The *smaller*, the *better*.

Related works

- Most of existing works did not consider the multi-device problem
 - Bahl & Padmanabhan 2000 – K-Nearest-Neighbors
 - Nguyen, Jordan, & Sinopoli 2005 – Kernel learning
 - Ferris, Fox, & Lawrence 2007 – Gaussian process
- Few work address the multi-device problem
 - Haeberlen *et al.* 2002 – **Linear fit** between devices' signals
For **each AP-dimension**, they try to fit a linear model:
$$RSS_B = c_1 * RSS_A + c_2$$
where RSS_A , RSS_B – signal strength for device A and B;
 c_1 , c_2 – linear model parameter.

Transferring Localization Models over Devices

- The Question:
 - Given some labeled data from a **source device**, how to use as few labeled data as possible to calibrate a **target device**?
- Our Solution: Multi-task Learning (MTL)
 - The idea behind multi-task learning is to pool together the data from all related tasks, such that the learning on each task can benefit from the data enrichment
 - **Learning on Multiple devices = Multiple tasks**
so that, the learning in **target device** can benefit from the labeled data information in **source device**

Our Solution: Latent Multi-task Learning

- Many existing MTL methods assume
 - The hypotheses learned from the **original feature** space for related tasks can be similar
 - Consider regression in MTL
$$f_t(z) = \langle w_t, z \rangle + b$$

i.e. w_t 's to be similar for the tasks ($t=1,2,\dots$)
 - However, in multi-device problem, w_t 's can be quite different
- **Latent** Multi-task Learning (LatentMTL)
 - We only require that multiple tasks may share similar parameter structure in some **feature space**

$$f_t(z) = \langle w_t, \varphi_t(z) \rangle + b, \quad w_t = w_0 + v_t$$

Overview of the Algorithm

■ Our Multi-task learning

- Input: $D_{\text{src}} + D_{\text{tar}}$
- Output: labels for D_{tst}
- Training phase
 1. Learn the latent space
i.e. ϕ_t for task t , $t=1, \dots, T$
 2. Learn the localization model
i.e. (w_t, b) for task t , $t=1, \dots, T$
- Localization phase
 1. Predict labels for D_{tst} by using
 ϕ_{tar} and (w_{tar}, b)

■ Single-task learning

- Input: D_{tar}
- Output: labels for D_{tst}
- Training phase
 1. Learn the localization model
i.e. (w, b) for D_{tar}
- Localization phase
 1. Predict labels for D_{tst} by using
 (w, b)

We aim to benefit from exploiting the **latent space** and **source device data!**

Learning with LatentMTL

- The objective (encoded in SVR) is to optimize

$$\min J(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b, \varphi_t) =$$

$$\underbrace{\sum_{t=1}^T \pi_t \sum_{i=1}^{n_t} (\xi_{it} + \xi_{it}^*)}_{\text{Loss}} + \underbrace{\frac{\lambda_1}{T} \sum_{t=1}^T \|\mathbf{v}_t\|^2}_{\text{Regularize Uniqueness}} + \underbrace{\lambda_2 \|\mathbf{w}_0\|^2}_{\text{Maximize Margin}} + \underbrace{\frac{\lambda_3}{T} \sum_{t=1}^T \Omega(\varphi_t)}_{\text{Mapping Function Complexity}}$$

$$s.t. \quad \begin{cases} y_{it} - (\mathbf{w}_0 + \mathbf{v}_t) \cdot \varphi_t(\mathbf{x}_{it}) - b \leq \varepsilon + \xi_{it} \\ (\mathbf{w}_0 + \mathbf{v}_t) \cdot \varphi_t(\mathbf{x}_{it}) + b - y_{it} \leq \varepsilon + \xi_{it}^* \\ \xi_{it}, \xi_{it}^* \geq 0 \end{cases}$$

Constraints for ε -SVR

Learning with LatentMTL (Cont')

■ Optimization Issues

- Convex function with bilinear constraints
- **Not global optimal**, but each sub-problem is convex
 - Sub-problem 1: learning regression weights
 - Sub-problem 2: learning feature mapping functions

■ Alternating Optimization

- **Step 1**: Fix feature mapping functions, solve sub-problem 1
- **Step 2**: Fix regression weights, solve sub-problem 2
- Iterate the two steps until **converged***

* c.f. Theorem 2 for convergence proof

Learning with LatentMTL (Cont')

■ Alternating Optimization

○ Step 1

$$\begin{aligned} \min \tilde{J}(\mathbf{w}, \xi_{it}, \xi_{it}^*) &= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{t=1}^T \pi_t \sum_{i=1}^{n_t} (\xi_{it} + \xi_{it}^*) \\ \text{s.t.} \quad &\begin{cases} y_{it} - \mathbf{w} \cdot \phi((\varphi(\mathbf{x}_{it}), t)) - b \leq \varepsilon + \xi_{it} \\ \mathbf{w} \cdot \phi((\varphi(\mathbf{x}_{it}), t)) + b - y_{it} \leq \varepsilon + \xi_{it}^* \\ \xi_{it}, \xi_{it}^* \geq 0 \end{cases} \end{aligned}$$

○ Step 2

$$\begin{aligned} \min \hat{J}(\varphi_t) &= \tilde{\varphi}_t' \cdot I \cdot \tilde{\varphi}_t \\ \text{s.t.} \quad &A_1 \leq B\tilde{\varphi}_t \leq A_2 \end{aligned}$$

○ Iterate the two steps until **converged**

○ **Both steps are quadratic programming problems!**

The Algorithm for LatentMTL

- **Input:** labeled D_{src}, D_{tar}^l , unlabeled D_{tar}^{tst} ; optionally unlabeled D_{tar}^u
- **Output:** labels for test data D_{tar}^{tst}
- Training phase
 - **Initialize** the latent space dimension and feature mappings
 - **Alternating optimization** to minimize the objective function, which involves using input data
 - **Derive** the feature mappings $\varphi_t(\cdot)$ and regression weights (w_t, b) after convergence
- Localization phase
 - Map the test data D_{tar}^{tst} to a latent feature space, and **predict** the labels by $f_{tar}(z) = \langle w_{tar}, \varphi_{tar}(z) \rangle + b$

[Experimental Setup]



WiFi Access Point (AP)

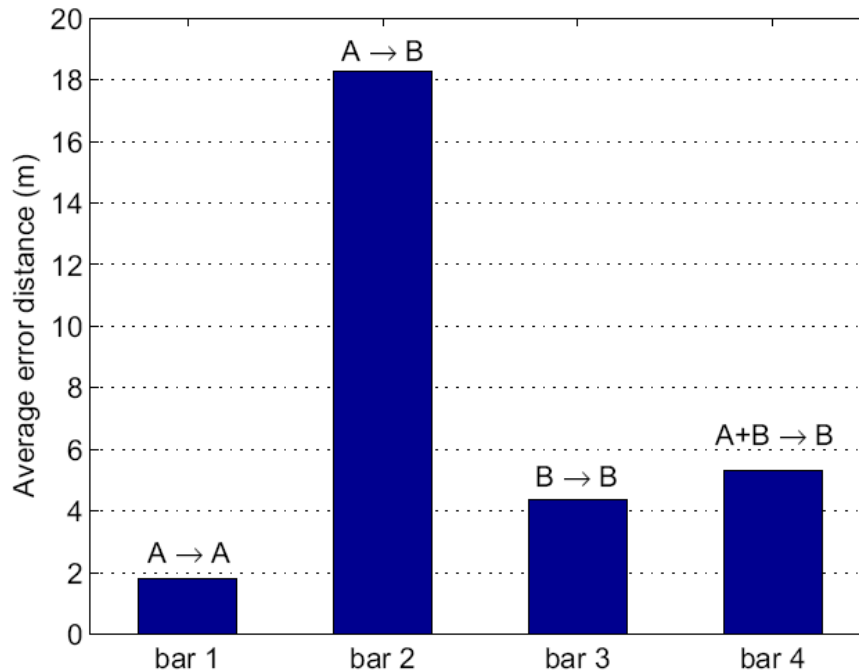
Academic building
64 x 50 m²

Two different laptops

User moves in the building

Experimental Results

- Necessity for considering multi-device problem



Device A (Source device)

Device B (Target device)

“ $X \rightarrow Y$ ”:

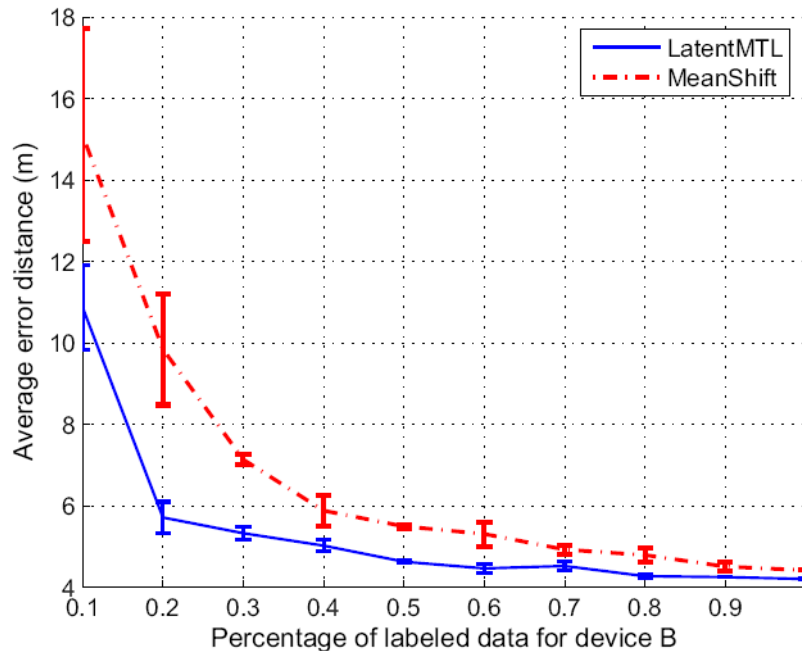
X for training, Y for testing.

Measurement:

Average error distance (meter)
(the smaller, the better).

Experimental Results (Cont')

■ LatentMTL vs. Linear-Fit (Haeberlen *et al.* 2002)



Linear-Fit:

Collect labeled data for both devices.

For each AP-dimension, fit a linear model:

$$RSS_B = c_1 * RSS_A + c_2$$

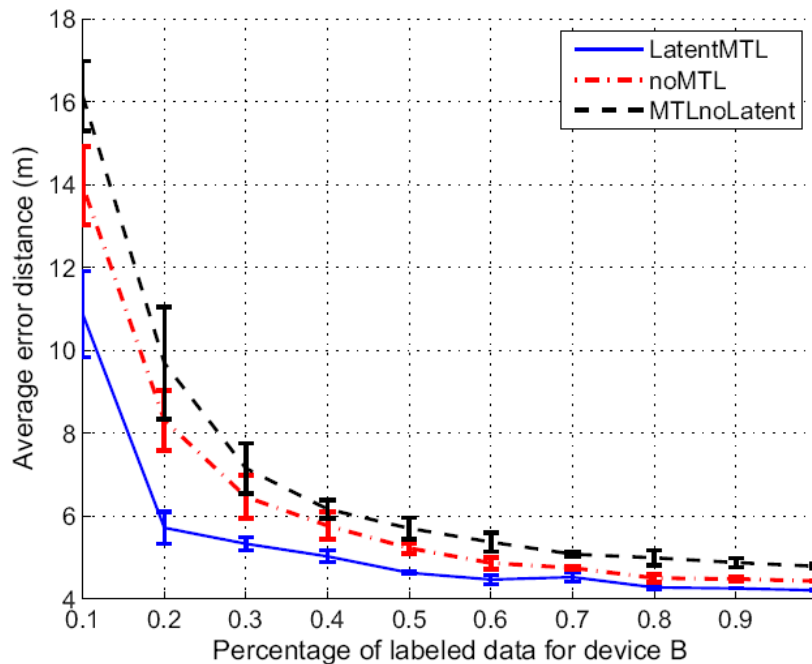
where

RSS_A , RSS_B – signal strength for device A and B;

c_1 , c_2 – linear model parameter.

Experimental Results (Cont')

LatentMTL vs. noMTL & MTLnoLatent



noMTL:

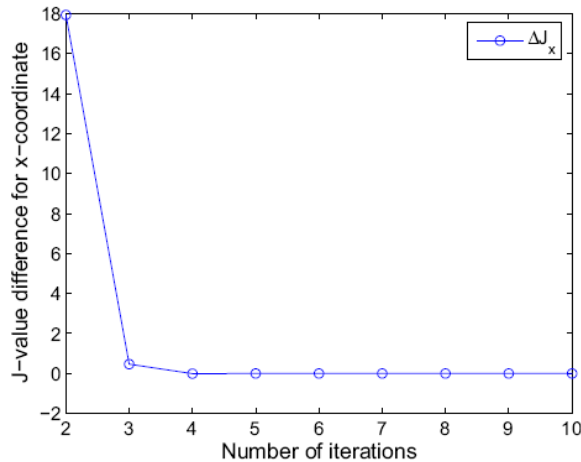
Use only some device B's data for training and testing.

MTLnoLatent:

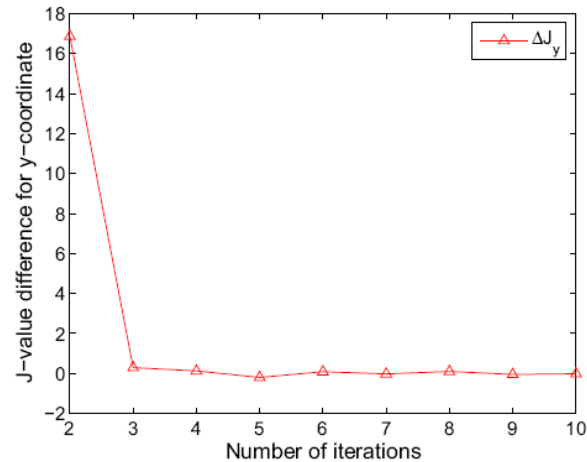
Use device A and device B's data for training, but without exploiting the latent space.

Experimental Results (Cont')

Convergence of LatentMTL



(a) J_x -value difference ΔJ_x



(b) J_y -value difference ΔJ_y

J_x , J_y – the objective function value for both x- and y- coordinates in a 2-D physical location space.

Conclusion and Future Work

■ The Algorithm

Conclusion

- Signal variation over device
- Multiple device = multiple tasks (latent multi-task learning)
- When calibrating new device, benefit from existing devices
- Alternating optimization, easily solved

■ Algorithm level

Future Work

- Extend to semi-supervised setting
- Exploit nonlinear feature mapping function form

■ Application level

- Learn with more different devices together

[The End]

Thank You
Questions?