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

Vincent Wenchen Zheng, Sinno Jialin Pan, Qiang Yang and Jeffrey Junfeng Pan
Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, Hong Kong
{vincentz,sinnopan,qyang,panjf}@cse.ust.hk

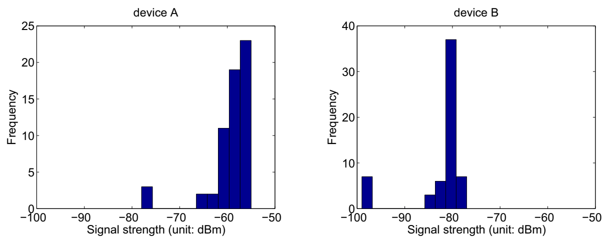
Abstract

In this paper, we propose a latent multi-task learning algorithm to solve the multi-device indoor localization problem. Traditional indoor localization systems often assume that the collected signal data distributions are fixed, and thus the localization model learned on one device can be used on other devices without adaptation. However, by empirically studying the signal variation over different devices, we found this assumption to be invalid in practice. To solve this problem, we treat multiple devices as multiple learning tasks, and propose a multi-task learning algorithm. Different from algorithms assuming that the hypotheses learned from the original data space for related tasks can be similar, we only require the hypotheses learned in a latent feature space are similar. To establish our algorithm, we employ an alternating optimization approach to iteratively learn feature mappings and multi-task regression models for the devices. We apply our latent multi-task learning algorithm to real-world indoor localization data and demonstrate its effectiveness.

Introduction

With the increasing availability of 802.11 Wireless LAN, indoor localization using wireless signal strength information has attracted more and more interest from research and industrial communities (Bahl & Padmanabhan 2000; Nguyen, Jordan, & Sinopoli 2005; Ferris, Fox, & Lawrence 2007). Learning based WiFi localization methods generally work in two phases: in an *offline* phase, a mobile device moving around the wireless environment is used to collect wireless signals from various access points (APs). Then, the received signal strength (RSS) values, together with the location information, are used as the training data to learn a statistical localization model. In an *online* phase, the learned localization model is used to infer the locations according to the real-time RSS values.

A major drawback of traditional localization methods is that they assume the collected signal data distributions to be fixed so that the localization model learned on one device can be used for other devices. However, this is not always feasible. In practice, different data collection devices may have different signal sensing capacities and yield different data distributions. To illustrate this, we empirically studied the signal variation over different devices. As shown in Figure 1, the signals collected by two devices can be quite different even at a same location. This directly causes the traditional localization models to fail if we use one device's data for training and another device's data for testing. Because it is expensive to recollect a large amount of data for calibrating the new device, a more practical method would be to collect a small amount of data on the new device, and integrate them with a large amount of data collected before on other devices. If we can train an accurate model on the combined data, it would save much human effort. This problem is referred to as a *multi-device localization* problem. An intuitive solution to this problem is to see the signal variation over devices as simple Gaussian mean shift, and use a linear model to fit this shift from the data collected by both devices (Haeberlen *et al.* 2002). However, as we will show in the "Experiments" section, such simple method may not work well in complex indoor environments. Observing that although the devices may be different from each other, the learning tasks on these devices are related since they all try to learn a mapping function from a signal space to a *same* location space. This motivates us to model the multi-device localization as a multi-task learning problem (Caruana 1997).



(a) RSS at device A (b) RSS at device B

Figure 1: Signal variations over two different devices.

Multi-task learning was proposed to exploit the task relatedness for improving the learning performance (Caruana 1997; Evgeniou & Pontil 2004). The idea behind multi-task learning is that it pools together the data from all the related

*This research is supported by NEC Lab China (NECLC05/06.EG01) and Hong Kong CAG Grant HKBU1/05C.

tasks, such that the learning on each task can benefit from the data enrichment, especially when each task can only have a small number of labeled data for training. This coincides with our motivation by only collecting a small number of labeled data by the new device, and trying to utilize the data from other devices for model learning. Many existing multi-task learning methods assume that the hypotheses learned from the original feature space for related tasks can be similar. This potentially requires the data distributions for related tasks to be similar in the high-dimensional feature space. However, in our multi-device localization problem, the data distributions are expected to be quite different from each other. Therefore, we extend the multi-task learning for multi-device localization problem by only requiring that the hypotheses learned from a *latent* feature space are similar. In other words, we look for appropriate feature mappings, by which we can map different devices' data to a well-defined low-dimensional feature space. In this latent space, new device can benefit from integrating the data collected before by other devices to train a localization model.

The contributions of our work are as follows. For WiFi localization, we develop a novel solution for calibrating a new device by making use of data collected before on other devices, thus saving a great deal of data recollection effort. For machine learning, we develop a latent multi-task learning algorithm for better exploiting the task relatedness. In our method, we employ an *alternating optimization* (Bezdek & Hathaway 2003) approach to iteratively learn the latent feature mappings and the multi-task regression function. We demonstrate our method's effectiveness on real data sets.

Related Work

Indoor Localization

Recent learning-based indoor localization methods include (Bahl & Padmanabhan 2000; Nguyen, Jordan, & Sinopoli 2005; Ferris, Fox, & Lawrence 2007). Different from localization systems using signal propagation models for location estimation, learning based methods use statistical learning algorithms to train a mapping function from a signal space to a location space. For example, location estimation is done in (Nguyen, Jordan, & Sinopoli 2005) through kernel learning and (Ferris, Fox, & Lawrence 2007) through Gaussian-process latent variable model. Few of the previous works considered the signal-data distribution variation across different devices. Haeberlen et al. (Haeberlen *et al.* 2002) treated signal variation as a Gaussian mean-value shift, and used a linear model to fit the RSS values on source and target devices. However, as we show in our experimental studies, such an obvious adaptation method does not work well in a complex indoor environment.

Multi-task Learning

Multi-task learning (Caruana 1997) jointly learns a set of related tasks together so that the tasks can benefit each other. To better understand this, some theoretical work for multi-task learning was done by (Baxter 2000; Ben-David & Schuller 2003). Recent works on multi-task learning include (Bakker & Heskes 2003; Evgeniou & Pontil 2004), *etc.* An assumption for these works is that the hypotheses learned from the original high-dimensional feature space for related tasks are similar. However, this assumption may not hold in practice. If the data distributions for related tasks are very different, the hypotheses learned from the original feature space may differ a lot from each other. We consider this case and only require the hypotheses learned in a *latent* feature space are similar. We show that this latent space can be found through regularization techniques. Our work shares some similarities with recent work on latent structure multi-task learning (Ando & Zhang 2005; Argyriou, Evgeniou, & Pontil 2007; Argyriou *et al.* 2008). However, most of these works assumed that different tasks share a common feature mapping and their objective was to find this mapping. When the tasks have different kinds of features, for example, learning English and French, using a common feature mapping for both tasks can be inappropriate. We attempt to establish a general multi-task learning framework which can uncover an appropriate latent feature mapping for each task, so that our framework can be used not only in multi-device localization but also some other areas with tasks possibly having different features.

Problem Description

Consider a two-dimensional indoor localization problem. In the environment, assume that there are m access points (APs), which periodically send out wireless signals. A user walking in the environment with a device can measure the received signal strength (RSS) sent from the m APs. Each RSS vector $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_m^{(i)})'$ is a data instance, and its label $\mathbf{y}^{(i)} = (y_1^{(i)}, y_2^{(i)})$ is a 2-D location coordinate vector in $\mathbb{R}^2$. In latent multi-task learning for multi-device localization, we're given a *source* device, by which we have collected a large amount of labeled signal data $D_{src} = \{(\mathbf{x}_{src}^{(i)}, y_{src,j}^{(i)})\}$, $i = 1, ..., n_s, j = 1, 2$. Here $\mathbf{x}_{src}^{(i)}$ is a signal vector drawn from a distribution $\mathcal{P}$ in a feature space $\mathcal{X}_s$. $y_{src,j}^{(i)}$ is a corresponding location coordinate. Our objective is to use D_{src} to help calibrate a *target* device, by which we will only collect a small amount of labeled signal data $D_{tar}^l = \{\mathbf{x}_{tar}^{(i)}, y_{tar,j}^{(i)}\}$, $i = 1, ..., n_t^l, j = 1, 2$. Here $n_t^l \ll n_s$. $\mathbf{x}_{tar}^{(i)}$ is a signal vector drawn from a distribution $\mathcal{Q}$ in a feature space $\mathcal{X}_t$. Distributions $\mathcal{Q}$ and $\mathcal{P}$ are different. The dimensions for $\mathbf{x}_{src}$ and $\mathbf{x}_{tar}$ need not be the same. Optionally, some unlabeled data for the target device $X_{tar}^u = \{\mathbf{x}_{tar}^{(i)}\}$, $i = n_t^l + 1, ..., n_t^l + n_t^u$, can also be made available. Finally, we have a test data set from target device $D_{tar}^{tst} = \{\mathbf{x}_{tar}^{tst(i)}, y_{tar,j}^{tst(i)}\}$, $i = 1, ..., n_t^{tst}, j = 1, 2$. We will predict the labels for the test data in D_{tar}^{tst} by making use of D_{tar}^l, D_{src} and optionally X_{tar}^u.

Latent Multi-task Learning

In latent multi-task learning for multi-device localization, we treat multiple devices as multiple learning tasks. In this paper, we consider $T = 2$ tasks[1] with a source device and a

[1]Learning with more than 2 tasks can be a natural extension.

target device. However, larger T can also be handled by our framework. We are interested in finding appropriate feature mappings φ_{src} and φ_{tar}, with which we can map data from both source and target devices' data X_{src} and X_{tar} into a k-dimensional latent feature space $\mathcal{X}_m$ where the the learned hypotheses are similar. As in (Evgeniou & Pontil 2004), we model the hypotheses similarity (or, task relatedness) by exploiting the shared structure by the hypotheses. More specifically, we consider a regression problem: $f(\mathbf{z}) = \langle \mathbf{w}, \mathbf{z} \rangle + b$, where $\mathbf{z} \in \mathcal{X}_m$ are RSS vectors, $f(\cdot)$ outputs the locations, and $\langle \cdot, \cdot \rangle$ denotes a dot product. Each task t, $t = 1, ..., T$, has a hypothesis parameterized as $\mathbf{w}_t$[2]. These $\mathbf{w}_t$ share a common structure $\mathbf{w}_0$ by

$$\mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t, \quad for \quad t = 1, ..., T.$$

$\mathbf{v}_t$ denotes the difference for each task t. For latent multi-task learning, our aim is to find appropriate feature mapping functions φ_t, $t = 1, ..., T$, such that the regression loss is minimized across T tasks and the task hypotheses are similar in $\mathcal{X}_m$, *i.e.* $\mathbf{v}_t$ is "small".

Based on this, we formulate our latent multi-task learning method under a soft-margin support vector regression (SVR) framework (Smola & Schölkopf 2004) as follows:

$$\begin{array}{l}\min J(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b, \varphi_t) = \\ \sum\limits_{t=1}^{T} \pi_t \sum\limits_{i=1}^{n_t} (\xi_{it} + \xi_{it}^*) + \frac{\lambda_1}{T} \sum\limits_{t=1}^{T} \|\mathbf{v}_t\|^2 + \lambda_2 \|\mathbf{w}_0\|^2 + \frac{\lambda_3}{T} \sum\limits_{t=1}^{T} \Omega(\varphi_t) \\ s.t. \quad \left\{ \begin{array}{l} y_{it} - (\mathbf{w}_0 + \mathbf{v}_t) \cdot \varphi_t(\mathbf{x}_{it}) - b \le \varepsilon + \xi_{it} \\ (\mathbf{w}_0 + \mathbf{v}_t) \cdot \varphi_t(\mathbf{x}_{it}) + b - y_{it} \le \varepsilon + \xi_{it}^* \\ \qquad \xi_{it}, \xi_{it}^* \ge 0 \end{array} \right. . \end{array} \tag{1}$$

From left to right in Equation (1), we explain each term:

- In the *first* term, the inner summation denotes the regression loss for task t, with ξ_{it} and ξ_{it}^* as slack variables measuring the errors. n_t is the number of labeled data for task t. The outer summation computes the weighted loss over all the T tasks with weight parameters π_t for each task t. Intuitively, minimizing the first term equals to minimizing the localization error for the T devices.
- In the *second* term, minimizing $\|\mathbf{v}_t\|^2$ regularizes the dissimilarity among the task hypotheses in the latent feature space $\varphi_t(\mathbf{x})$.
- In the *third* term, minimizing $\|\mathbf{w}_0\|^2$ corresponds to maximizing the margin of the learned models to provide the generalization ability. Here, λ_1 and λ_2 are positive regularization parameters. Generally λ_1 is larger than λ_2 to force the task hypotheses to be similar.
- In the *fourth* term, $\Omega(\varphi_t)$ denotes the complexity of mapping function φ_t. Minimizing it is to regularize the mapping function complexity to prevent overfit. To make our problem tractable, we consider φ_t as a linear transformation, and leave the nonlinear case for future work. With a slight abuse of notation, we let $\varphi_t(\mathbf{x}) = \varphi_t \mathbf{x}$, where φ_t is a $k \times d$ matrix. k is the dimension of the latent space and $k < d$. We interpret the complexity function $\Omega(\varphi_t)$ as the *Frobenius norm, i.e.* $\Omega(\varphi_t) = \|\varphi_t\|_F^2$.
- The *constraints* follow the routine of the standard ϵ-SVR (Smola & Schölkopf 2004), with b as the bias term and ϵ as the tolerance parameter.

[2]Here we fix b across tasks for computation simplicity. However, a varying b can be addressed in an extension of this work.

Notice that Equation (1) consists of a convex objective function and bilinear constraints, the whole optimization problem is not convex. However, separately considering the parameters $(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b)$ and φ_t, we can reduce Equation 1 to a convex optimization problem. Motivated by this, we employ an alternating optimization approach to solve the problem, by iteratively optimizing the parameters $(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b)$ with fixed φ_t and optimizing φ_t with fixed $(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b)$. Therefore, our latent multi-learning algorithm consists of two steps: the first step learns the parameters of the regression function and the second step learns low-dimensional feature mapping functions for all the tasks.

Learning the Regression Function

Given mapping functions φ_t, $t = 1, .., T$, we re-formulate Equation (1) as a standard ε-SVR problem by feature mapping. Notice that the regression functions are used to predict the location labels $\{y_{it}\}$ for signal data $\{\mathbf{x}_{it}\}$, this step corresponds to *offline training* localization model in a latent feature space $\varphi_t(\mathbf{x})$.

First, we consider the function $f_t(\varphi_t(\mathbf{x})) = \mathbf{w}_t \cdot \varphi_t(\mathbf{x}) + b$, $t = 1...T$. This function can be identified by a real-valued function $F : X \times \{1, ..., T\} \to \mathbb{R}$ with $F(\varphi_t(\mathbf{x}), t) = f_t(\varphi_t(\mathbf{x}))$, which takes $((\varphi_t(\mathbf{x}), t), y)$ as training examples. Then, we can define a feature mapping on $(\varphi_t(\mathbf{x}), t)$ as:

$$\phi((\varphi_t(\mathbf{x}), t)) = \left(\frac{\varphi_t(\mathbf{x})}{\sqrt{\mu}}, \underbrace{\mathbf{0}, ..., \mathbf{0}}_{t-1}, \varphi_t(\mathbf{x}), \underbrace{\mathbf{0}, ..., \mathbf{0}}_{T-t} \right), \tag{2}$$

where $\mathbf{0} \in \mathbb{R}^d$ is a zero vector, and $\mu = \frac{T\lambda_2}{\lambda_1}$.

Theorem 1. *The latent multi-task learning in Equation (1) can be re-formulated as a standard ε-SVR problem:*

$$\begin{array}{l}\min \tilde{J}(\boldsymbol{w}, \xi_{it}, \xi_{it}^*) = \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum\limits_{t=1}^{T} \pi_t \sum\limits_{i=1}^{n_t} (\xi_{it} + \xi_{it}^*) \\ s.t. \quad \left\{ \begin{array}{l} y_{it} - \boldsymbol{w} \cdot \phi((\varphi(\boldsymbol{x}_{it}), t)) - b \le \varepsilon + \xi_{it} \\ \boldsymbol{w} \cdot \phi((\varphi(\boldsymbol{x}_{it}), t)) + b - y_{it} \le \varepsilon + \xi_{it}^* \\ \qquad \xi_{it}, \xi_{it}^* \ge 0 \end{array} \right. , \end{array} \tag{3}$$

where $C = \frac{T}{2\lambda_1}$, and $\boldsymbol{w} = (\sqrt{\mu} w_0, v_1, ..., v_T)$.

Proof. Details are given in the Appendix. □

By deriving the dual of Equation (3) (details are given in the Appendix), we can have the optimal solution for the regression function as follows:

$$f^*(\mathbf{x}) = \sum_{t=1}^{T} \sum_{i=1}^{n_t} (\alpha_{it} - \alpha_{it}^*) K(\mathbf{x}_{it}, \mathbf{x}) + b, \tag{4}$$

where $K(\mathbf{x}_{is}, \mathbf{x}_{jt}) = (\frac{1}{\mu} + \delta_{st}) \varphi_s(\mathbf{x}_{is}) \cdot \varphi_t(\mathbf{x}_{jt})$. δ_{st} is an indicator function, which equals to 1 if $s = t$, and 0 otherwise. Based on $f^*(\mathbf{x})$, we can derive the parameters ($\mathbf{w}_0$, $\mathbf{v}_t$, ξ_{it}, ξ_{it}^*, b) for the use in next step.

Algorithm 1 Latent Multi-task Learning (LatentMTL)

```
Input: Labeled source device data $D_{src}$, labeled target device data $D^l_{tar}$, unlabeled target device data $D^{tst}_{tar}$; optionally, also some unlabeled target device data $D^u_{tar}$
Output: Labels for $D^{tst}_{tar}$
begin
  Offline Model Training:
  1: Initialize the latent space dimensionality $k$ and the feature mapping functions $\varphi_t$;
  2: repeat
  3:     Learn the regression function $f^*(\mathbf{x})$ in the latent feature space defined by $\varphi_t$, as shown in Eq.(3)
  4:     Learn the latent space by optimizing the feature mapping functions $\varphi_t$, as shown in Eq.(5);
  5: until In Eq.(1), $J$'s change is less than a threshold $\zeta$
  Online Location Estimation:
  1: For each $\mathbf{x}_i \in D^{tst}_{tar}$, output its location label by $f^*(\mathbf{x})$.
end
```

Learning the Latent Feature Space

Given the regression function parameters $(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi^*_{it}, b)$ from the previous step, we can reduce Equation (1) to

$$\begin{aligned} &\min \hat{J}(\varphi_t) = \sum_{t=1}^{T} \|\varphi_t\|_F^2 \\ &s.t. \quad \begin{cases} y_{it} - \mathbf{w}_t \cdot \varphi_t \mathbf{x}_{it} - b \le \varepsilon + \xi_{it} \\ \mathbf{w}_t \cdot \varphi_t \mathbf{x}_{it} + b - y_{it} \le \varepsilon + \xi^*_{it} \end{cases} . \end{aligned} \tag{5}$$

By optimizing over the feature mapping functions φ_t, we find the common latent space for the previous step's offline localization model training.

We show Equation (5) can be re-formulated as a standard Quadratic Programming (QP) problem. First, we vectorize the mapping function matrix φ_t as $\tilde{\varphi}_t = [\varphi'_{t1}, ..., \varphi'_{td}]'$, with each $k \times 1$ vector φ_{ti} is the i^{th} column of the matrix φ_t. Hence,

$$\begin{aligned} &\mathbf{w}_t \cdot \varphi_t \mathbf{x}_{it} = (\mathbf{w}'_t \varphi_t)\mathbf{x}_{it} \\ &= [\mathbf{w}'_t \varphi_{t1}, ..., \mathbf{w}'_t \varphi_{t1}]\mathbf{x}_{it} = [\mathbf{x}^1_{it}\mathbf{w}'_t, ..., \mathbf{x}^d_{it}\mathbf{w}'_t]\tilde{\varphi}_t. \end{aligned}$$

Define the data for task t as X_t, a $n \times d$ matrix with each row as a data point $\mathbf{x}_{it}$. Then, the conditions in Equation (5) can be re-formulated as

$$A_1 \le B\tilde{\varphi}_t \le A_2,$$

where A_1 is a $n_t \times 1$ vector with each element $A^i_1 = y_{it} - b - (\varepsilon + \xi_{it})$. Similarly, A_2 is a $n_t \times 1$ vector with each element $A^i_2 = y_{it} - b + (\varepsilon + \xi^*_{it})$. B is a Kronecker product over data matrix X_t and the transposed regression weight $\mathbf{w}'_t$, *i.e.* $B = X_t \otimes \mathbf{w}'_t$. Both A_1, A_2 and B can calculated from previous section by solving the standard SVR problem. Notice that the minimization in Equation (5) is independent among each task t, we can re-formulate it as a QP problem:

$$\begin{aligned} &\min \hat{J}(\varphi_t) = \tilde{\varphi}'_t \cdot I \cdot \tilde{\varphi}_t \\ &s.t. \quad A_1 \le B\tilde{\varphi}_t \le A_2 \ . \end{aligned} \tag{6}$$

Any QP solver can be used to derive the optimal solution.

Theorem 2. *Convergence: the latent multi-task learning algorithm is guaranteed to converge to a local optimal solution in a finite number of iterations.*

Proof. Observing that Equation (1) consists of a convex objective function and bilinear constraints, we find that the optimization problem is not convex. So we use an alternating optimization approach to iteratively minimize the J-value. As shown in (Bezdek & Hathaway 2003), the alternating optimization procedure can guarantee the algorithm to converge to local optimum in a finite number of iterations. $\square$

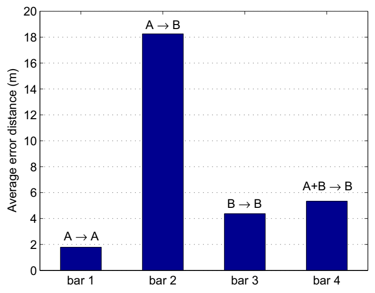
Experiments

In this section, we study the benefits of transferring multi-device localization models using our latent multi-task learning algorithm. Our experiments were set up in a $64m \times 50m$ department office area. The calibration data were collected by two laptops with different wireless adapters, denoted as A and B. Our goal was to calibrate target device B with the help of source device A. We collected training and test data in total 107 locations. At each location, we collected 20 samples for each device. For each device, we randomly splitted 50% of the data for training, and the rest for testing. The collected signal vector has 118 dimensions, and the latent space dimension k is set to be 60. We tested different k-values and found that $k = 60$ seems to give the best results. When k is too large, the computation cost is too high. When k is too small, the performance may suffer due to information loss. Since the localization problem is a regression problem, we follow (Ferris, Fox, & Lawrence 2007) to report the average *error distance*[3] and the standard deviation.

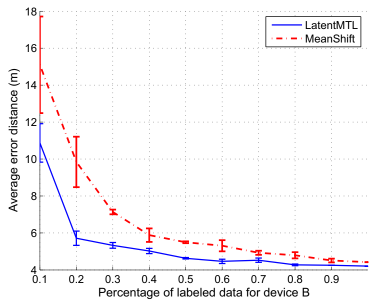
First of all, we would like to confirm our motivation for considering multi-device localization. Figure 2(a) shows the statistical results of five trials. In this figure, bar 1 corresponds to learning an SVR model on device A's training data, and then test the model on device A's test data. The average error distance is $1.79 \pm 0.08m$. Bar 2 in the figure corresponds to training an SVR model on device A's training data, and test on device B's test data. We get an average error distance of $18.25 \pm 1.82m$. Compared to bar 1, such a performance is far from satisfactory[4]. This testifies our observation that the data distributions on different devices are quite different. Thus we need to conduct adaptation between devices. Our next question is: how well our *LatentMTL* method can perform? To answer this question, we use 100% of the device B's training data to train an SVR model, and test this model on device B's test data. As shown in bar 3 in Figure 2(a), we get an average error distance of $4.37 \pm 0.05m$. Note that this error distance derived from device B is much larger than that from device A as shown

[3]Error distance is calculated by the euclidean distance between a predicted location and its ground truth value. The average error distance is the mean of the error distances over all the test data. In localization, we attempt to minimize the error distance.

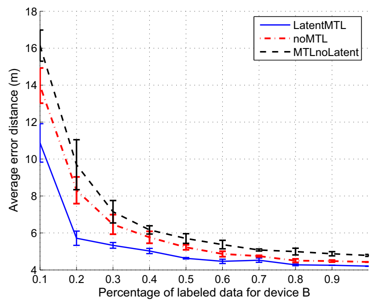
[4]We get similar result using device B's data for training and using device A's data for testing. Therefore, in the rest of the paper, we only consider adapting from device A (as source device) to device B (as target device).



(a) Empirical study of the need for device adaptation.



(b) LMTL-MeanShift



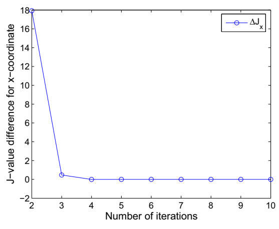
(c) LatentMTL vs. noMTL & MTLnoLatent

Figure 2: Experimental results.

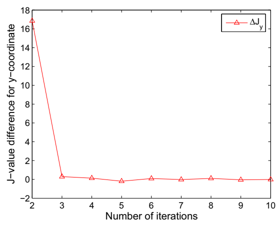
by bar 1. From Figure 1, we can know the reason clearly: while device A detects a signal strength value of -60dBm, device B measures the same signal as -80dBm from a same AP at a same location. This means that device B has a lower signal sensing capacity than device A, which testifies our argument that different devices have different capacities for signal sensing. In bar 4, we show that our method can be close to the result in bar 3 with $5.33 \pm 0.16m$, by using only 30% of device B's training data and full device A's training data.

Secondly, we compare our *LatentMTL* method with a baseline method known as *MeanShift* (Haeberlen *et al.* 2002), to show that the signal variation is not a simple Gaussian mean shift. The basic assumption for *MeanShift* is that the effect of hardware variation on received signal strength is linear. Hence, for each AP signal, a linear model $c(i) = c_1 \cdot i + c_2$ can be designed to fit the RSS value $c(i)$ on a target device, using the RSS value i on a source device. In the above equation, c_1 and c_2 are model parameters, which can be computed by least square fit on both devices' data collected from a same set of locations. We vary the number of device B's labeled data for fitting the linear model. And after fitting the model, we apply SVR to learn a localization model for device B by using both the existing labeled data and the fitted data, in order to simulate *MeanShift*. As shown in Figure 2(b) by a five-trial test, our *LatentMTL* consistently outperforms the *MeanShift* method. That is because in a complex indoor environment, the linear assumption does not hold for the *MeanShift* method. We also observe that our *LatentMTL* algorithm quickly converges as the number of device B's data increases. Therefore, practically, we only need to collect at most 30% of the data to calibrate a new device, which gives a large reduction of the calibration effort compared to recollecting all the data over the whole region for a new device.

Thirdly, we design experiments to show the benefits of using latent multi-task learning for multi-device localization. We compare our method with two baselines. The first baseline algorithm corresponds to not using multi-task learning (*noMTL*) in localization, which means that we use only device B's available labeled data for training. The second baseline corresponds to using multi-task learning in the original feature space (*MTLnoLatent*), which means that we perform multi-task learning on the data from both devices without any latent feature mapping. The second baseline corresponds to the approach given in (Evgeniou & Pontil 2004). As shown in Figure 2(c) in a five-trial test, our *LatentMTL* consistently outperforms the two baselines on increasing size of training data. It is interesting to observe that *noMTL* is consistently better than *MTLnoLatent*. This is because data from the two devices can have very different distributions in the original feature space, such that in this space their learned hypotheses may not be similar. Using traditional multi-task learning in this setting may harm the performance. Such observation testifies our motivation of only requiring the learned hypotheses to be similar in a latent feature space.



(a) J_x-value difference ΔJ_x



(b) J_y-value difference ΔJ_y

Figure 3: Convergence of *latentMTL* algorithm.

Finally, we show the convergence property of our algorithm. In our algorithm, we need to separately train a model on both x and y coordinates for a 2-D location space. By making use of all of device A's training data and 10% of device B's training data, we record the J-value as J_x and J_y for each iteration i. The convergence is measured by the J-value difference through $\Delta J_x = J_x(i+1) - J_x(i)$ and $\Delta J_y = J_y(i+1) - J_y(i)$. As shown in Figure 3, our *latentMTL* algorithm quickly converges on this data set.

We also test the running time of *LatentMTL* on our Pentium 4 PC (2.13GHz, 1G RAM). Solving Equation 1 with 1,470 118-dimensional data samples requires less than 10 CPU minutes using MATLAB software. Since a new device only needs to be calibrated once, such running time for training a localization model on a device is practical.

Conclusion and Future Work

In this paper, we empirically study the signal variations over different devices, and propose a latent multi-task learning algorithm for transferring multi-device localization models. We formulate our model as an optimization problem with convex objective function and bilinear constraints, and employ an alternating optimization approach to solve it. The contribution of our work is that, we only require that the learned hypotheses (i.e. the localization models) to be similar in a latent space, which broadens the application range of multi-task learning. We apply our algorithm to the real-world multi-device localization problem, where our $LatentMTL$ algorithm compares favorably with other baseline algorithms.

For our future work, we plan to extend our algorithm to support semi-supervised learning and other types of learning in order to save calibration effort. In addition, we wish to exploit kernelized version of our mapping function to fit the nonlinear data.

References

Ando, R. K., and Zhang, T. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.* 6:1817–1853.

Argyriou, A.; Micchelli, C. A.; Pontil, M.; and Ying, Y. 2008. A spectral regularization framework for multi-task structure learning. In Platt, J.; Koller, D.; Singer, Y.; and Roweis, S., eds., *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press.

Argyriou, A.; Evgeniou, T.; and Pontil, M. 2007. Multi-task feature learning. In Schölkopf, B.; Platt, J.; and Hoffman, T., eds., *Advances in Neural Information Processing Systems 19*. Cambridge, MA: MIT Press. 41–48.

Bahl, P., and Padmanabhan, V. 2000. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the Conference on Computer Communications*, volume 2, 775–784.

Bakker, B., and Heskes, T. 2003. Task clustering and gating for bayesian multitask learning. *J. Mach. Learn. Res.* 4:83–99.

Baxter, J. 2000. A model of inductive bias learning. *Journal of Artificial Intelligence Research* 12:149–198.

Ben-David, S., and Schuller, R. 2003. Exploiting task relatedness for multiple task learning. In *Proceedings of the Sixteenth Annual Conference on Learning Theory*.

Bezdek, J. C., and Hathaway, R. J. 2003. Convergence of alternating optimization. *Neural, Parallel Sci. Comput.* 11(4):351–368.

Caruana, R. 1997. Multitask learning. *Machine Learning* 28(1):41–75.

Evgeniou, T., and Pontil, M. 2004. Regularized multi–task learning. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 109–117. New York, NY, USA: ACM.

Ferris, B.; Fox, D.; and Lawrence, N. 2007. Wifi-slam using gaussian process latent variable models. In *International Joint Conferences on Artificial Intelligence*, 2480–2485.

Haeberlen, A.; Flannery, E.; Ladd, A. M.; Rudys, A.; Wallach, D. S.; and Kavraki, L. E. 2002. Practical robust localization over large-scale 802.11 wireless networks. In *MOBICOM'02*.

Nguyen, X.; Jordan, M. I.; and Sinopoli, B. 2005. A kernel-based learning approach to ad hoc sensor network localization. *ACM Trans. Sen. Netw.* 1(1):134–152.

Smola, A. J., and Schölkopf, B. 2004. A tutorial on support vector regression. *Statistics and Computing* 14(3):199–222.

Appendix

Proof for Theorem 1

Consider $\mathbf{w} = (\sqrt{\mu}w_0, v_1, ..., v_T)$. Hence, $\mathbf{w}\cdot\phi((\varphi(\mathbf{x}), t)) = (\mathbf{w}_0 + \mathbf{v}_t)\cdot\varphi(\mathbf{x})$, and $\|\mathbf{w}\|^2 = \sum_{t=1}^{T}\|\mathbf{v}_t\|^2 + \mu\|\mathbf{w}_0\|^2$. Given φ_t, the term $\frac{\lambda_3}{T}\sum_{t=1}^{T}\Omega(\varphi_t)$ is constant. Hence,

$$
\begin{aligned}
&\min J \doteq \sum_{t=1}^{T}\pi_t\sum_{i=1}^{n_t}(\xi_{it}+\xi_{it}^*) + \frac{\lambda_1}{T}\sum_{t=1}^{T}\|\mathbf{v}_t\|^2 + \lambda_2\|\mathbf{w}_0\|^2 \\
&\doteq \frac{T}{2\lambda_1}\sum_{t=1}^{T}\pi_t\sum_{i=1}^{n_t}(\xi_{it}+\xi_{it}^*) + \frac{1}{2}\left(\sum_{t=1}^{T}\|\mathbf{v}_t\|^2 + \frac{T\lambda_2}{\lambda_1}\|\mathbf{w}_0\|^2\right) \\
&= C\sum_{t=1}^{T}\pi_t\sum_{i=1}^{n_t}(\xi_{it}+\xi_{it}^*) + \frac{1}{2}\|\mathbf{w}\|^2.
\end{aligned}
$$

Derivation for Equation (3)'s Dual

Denote $\phi((\varphi(\mathbf{x}_{it}), t))$ as $\hat{\mathbf{x}}_{it}$. The Lagrangian for Eq.(3) is

$$
\begin{aligned}
L &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{t=1}^{T}\pi_t\sum_{i=1}^{n_t}(\xi_{it}+\xi_{it}^*) \\
&+ \sum_{t=1}^{T}\sum_{i=1}^{n_t}\alpha_{it}(y_{it} - \mathbf{w}\cdot\hat{\mathbf{x}}_{it} - b - \varepsilon - \xi_{it}) + \sum_{t=1}^{T}\sum_{i=1}^{n_t}\beta_{it}\xi_{it} \\
&+ \sum_{t=1}^{T}\sum_{i=1}^{n_t}\alpha_{it}^*(\mathbf{w}\cdot\hat{\mathbf{x}}_{it} + b - y_{it} - \varepsilon - \xi_{it}) + \sum_{t=1}^{T}\sum_{i=1}^{n_t}\beta_{it}^*\xi_{it}^*.
\end{aligned}
$$

Let $\frac{\partial L}{\partial \mathbf{w}} = 0$, $\frac{\partial L}{\partial \xi_{it}^{(*)}} = 0$, and $\frac{\partial L}{\partial b} = 0$. We solve these equations and plug the solutions back to L. Then, we have the dual for Equation (3) as follows:

$$
\begin{aligned}
&\max_{\alpha_{it},\alpha_{it}^*} -\varepsilon\sum_{t=1}^{T}\sum_{i=1}^{n_t}(\alpha_{it}+\alpha_{it}^*) + \sum_{t=1}^{T}\sum_{i=1}^{n_t}y_{it}(\alpha_{it}-\alpha_{it}^*) \\
&-\frac{1}{2}\sum_{s=1}^{T}\sum_{i=1}^{n_s}\sum_{t=1}^{T}\sum_{j=1}^{n_t}(\alpha_{is}-\alpha_{is}^*)(\alpha_{jt}-\alpha_{jt}^*)K(\mathbf{x}_{is},\mathbf{x}_{jt}) \\
&\quad s.t. \quad \sum_{t=1}^{T}\sum_{i=1}^{n_t}(\alpha_{it}-\alpha_{it}^*) = 0 \quad and \quad \alpha_{it},\alpha_{it}^* \in [0, C\pi_t]\ .
\end{aligned}
\tag{7}
$$