Gesture Recognition Portfolios for Personalization

Angela Yao[1] *
[1]ETH Zürich
yaoa@vision.ee.ethz.ch

Luc Van Gool[1,2]
[2]KU Leuven
vangool@esat.kuleuven.be

Pushmeet Kohli[3]
[3]Microsoft Research Cambridge
pkohli@microsoft.com

Abstract

Human gestures, similar to speech and handwriting, are often unique to the individual. Training a generic classifier applicable to everyone can be very difficult and as such, it has become a standard to use personalized classifiers in speech and handwriting recognition. In this paper, we address the problem of personalization in the context of gesture recognition, and propose a novel and extremely efficient way of doing personalization. Unlike conventional personalization methods which learn a single classifier that later gets adapted, our approach learns a set (portfolio) of classifiers during training, one of which is selected for each test subject based on the personalization data. We formulate classifier personalization as a selection problem and propose several algorithms to compute the set of candidate classifiers. Our experiments show that such an approach is much more efficient than adapting the classifier parameters but can still achieve comparable or better results.

1. Introduction

The last few years have seen tremendous interest in developing "natural" user interfaces. These interfaces do not use keyboards or mice, which have been the dominant modes of human-computer interaction over the last few decades. Instead, they infer user intent through verbal commands and body gestures. An important milestone in the progress of natural user interfaces was the launch of Microsoft Kinect [18]. In fact, the combination of cheap depth cameras and real-time human pose estimation algorithms [7, 18] has sparked a trend of adding gesture-based control into all sorts of consumer electronics and hobbyist projects. Although Kinect has enabled the estimation of body part movements, the interface developer is still left with the challenge of assigning meaning to the movements.

A number of gesture recognition methods have been proposed in the literature [15, 6, 13], including some which operate on the human pose estimated from the Kinect [15, 6]. Most of these methods follow a standard classification paradigm. During training, samples are collected and a classifier is learned; during testing, end-users use this classifier, which may or may not generalize to his or her gestures. Gestures, however, like handwriting and speech, are inherently unique to an individual [6]. For instance, people in western societies usually perform a "Greet" gesture by shaking hands, while people from eastern societies, such as Japan, greet by bowing, or in the case of India, by performing "Namaste"[1]. Even if the same body movement is associated with a gesture, there are still significant variations in the speed and magnitude of movements among individuals. Finally, in a learning based approach, the system associates the gestures with movements natural to the people in the training dataset, but these movements may not be natural to the end-user.

One way to resolve the difficulties of learning a generic classifier is to personalize the classifier to every user. Typically, personalization involves learning the general classifier's parameters on the training data and then adapting these parameters to give the best performance on some "personalization" set – extra training samples collected from the intended user [11, 19, 21]. This type of user adaptation is used extensively in speech [12, 17] and handwriting recognition [3, 11], with the obvious tradeoff between the amount of personalization data (and subsequent accuracy) versus the inconvenience to the user. Depending on the extent of adaptation, the personalization step may involve a computationally expensive search for user parameters.

Personalization solves a problem that is conceptually similar to transfer learning and domain adaptation, where the target application, domain, or data distribution differs from the original training data. Solutions typically involve feature transformations between domains [16, 8] or adjustment of the classifier parameters [10, 19, 21, 5]. Many of these cases assume that significant amounts of data exists in the target domain, albeit partially [19, 16] or completely unlabeled [21, 8]. Such an assumption is unrealistic for gesture recognition, since collecting large amounts of personalization data is highly inconvenient for a target user and may be impossible after system deployment.

In this paper, we propose a novel and extremely efficient method of classifier personalization. We target applications with limited computational resources, e.g. a mobile app, requiring very little personalization data, since its collection can be onerous for the end user. Unlike traditional personal-

*This work began while Angela Yao was an intern at MSR Cambridge.

[1]palms pressed together, with fingers pointed upwards in front of chest

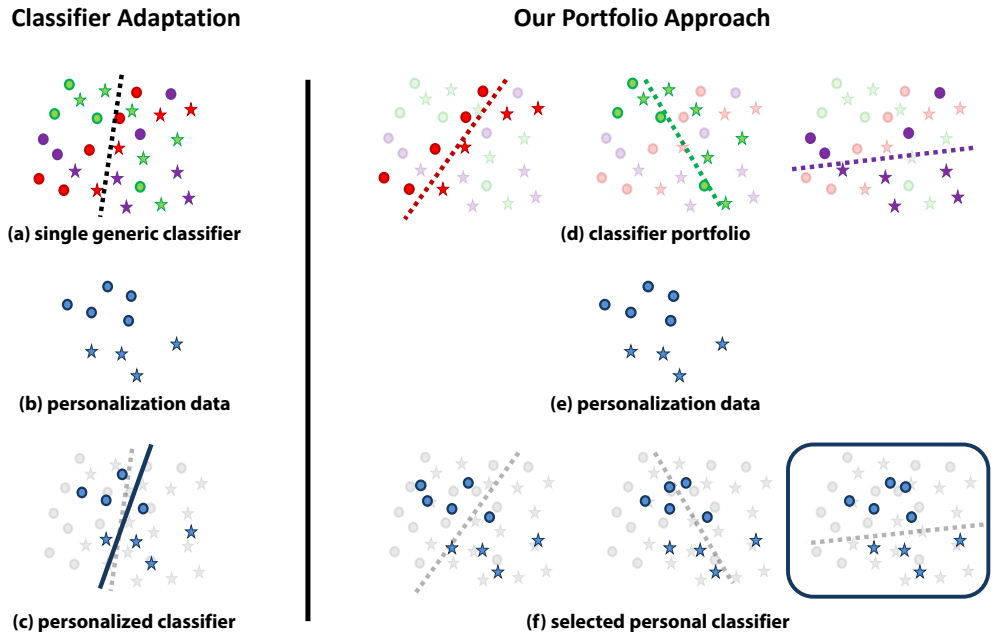


Figure 1. We compare standard personalization ((a) to (c)) with our proposed approach of selecting a personal classifier from a portfolio ((d) to (f)). The different shapes, *i.e.* the circles and stars, represent different gesture classes, while the colours represent different users. In standard personalization, one (a) learns a generic classifier on the training data, (b) collects personalization data and (c) then uses it as a regularizer for adjusting the original classifier parameters. Such approaches are undesirable as they may require large amounts of personalization data, be computationally expensive, or infeasible after system deployment. Our proposal is to (d) learn a portfolio of classifiers on the training data such that at least one classifier is well-suited to any given user. At run time, (e) personalization data is used to evaluate all classifiers in the portfolio and (f) to select the best for the target user. (Best viewed in colour.)

ization, which learn a single classifier that later gets adapted, our method learns a set, or *portfolio*, of classifiers during training. The training procedure ensures that at least one classifier is well-suited to any given user, so that during personalization, one simply evaluates all classifiers in the set and choose the best performing one (overview in Figure 1).

The key contributions of this paper are (1) formulation of classifier personalization as a selection problem, (2) proposal of methods for computing sets of candidate personalized classifiers, and (3) evaluation of the benefits of personalization on two datasets. While we use gesture recognition as the target application, the method is applicable for adapting any classifier to be user-specific.

2. Personalized Gesture Recognition

We formulate gesture recognition as a classification problem. For a set of data/label pairs $\{(x,y)|x \in \mathcal{X}, y \in \mathcal{Y}\}$, one can find a mapping f with parameters θ to make predictions $\hat{y} = f(x,\theta)$. In the standard classification paradigm, one learns an optimal θ_g^* on the training set $\mathcal{G} = \{(x_s^i, y_s^i) \mid i \in 1...K, s \in 1...M\}$ with K data/label pairs from M training subjects. Following a maximum likelihood estimate framework, θ_g^* is determined by:

$$\theta_g^* = \underset{\theta}{\operatorname{argmax}}\, L(\theta) = \underset{\theta}{\operatorname{argmax}} \sum_{s,i} \log\left(p(y_s^i|x_s^i;\theta)\right). \quad (1)$$

For a target user r, θ_g^* is applicable only if the user's underlying distribution $p_r(x,y)$ is the same as the training data $p_g(x,y)$. By factoring $p(x,y) = p(x|y) \cdot p(y)$, one can see that variations arise from differing $p(x|y)$ or $p(y)$. $p(x|y)$ reflects different physical movements that people associate with a given gesture, while $p(y)$ is the prior probability of observing a gesture being performed by an individual. In the case of gesture recognition, the need for personalization may come from both types of variation.

2.1. Subject- and Instance-Specific Reweighting

One can adapt the generic classifier of Eq. (1) into a user-specific classifier targeting user r by emphasizing the training instances coming from subjects more similar to the target user. More formally, a user-specific θ_r^* which is optimal for subject r can be determined by:

$$\theta_r^* = \underset{\theta}{\operatorname{argmax}} \sum_{s,i} d\big(r,s\big) \cdot \log\left(p(y_s^i|x_s^i;\theta)\right). \quad (2)$$

The function $d(r,s)$ encodes the similarity between user r and training subject s; its value is large if r and s are similar in terms of some attributes (e.g. age, gender, height *etc.*) that could affect the movements users associate with certain gestures. The reweighting has the effect of biasing the likelihood so that the learned classifier is especially good

at recognizing gestures from training subjects similar to the target user; one can also extend such a concept to make $d(\cdot)$ instance specific. Such an approach to personalization is intuitive and is analogous to the instance weighting approaches used in natural language parsing [1, 10, 5]. Unfortunately, it is impractical when extended to gesture recognition for a number of reasons. Firstly, it is unclear which user attributes are useful for measuring the similarity $d(\cdot)$. Secondly, the values of these attributes for the training subjects and or end-user may be unavailable. Finally, the optimal form that function $d(\cdot)$ ought to take is unclear.

2.2. Personalized Training Set

In some scenarios, the system can access some additional personalization data $\mathcal{P}_r = \{(x_r^j, y_r^j) \mid j \in 1...N\}$ data from the target user r. This might be collected from the user before he or she starts using the recognizer for the first time and is usually very limited in quantity, *i.e.* $N \ll M \times K$. By having access to the personalization data, one can then attempt to model $p(x|y)$. Previous works [11, 21] have proposed the use of $\mathcal{P}_r$ as an additional regularization term in the likelihood, *i.e.*

$$\theta_r^* = \underset{\theta}{\operatorname{argmax}}\, L(\theta) + \lambda_f \cdot \sum_i \log\left(p(y_r^i|x_r^i;\theta)\right), \quad (3)$$

where λ_f serves as a regularizing constant. The above problem formulation, which we will refer to as ***full personalization***, involves a complete re-maximization of the original conditional likelihood along with a personalized conditional likelihood. One way to simplify such a maximization is to assume that θ_r^* is close to θ_g^* and can be approximated as

$$\theta_r^* \approx \underset{\theta}{\operatorname{argmax}} \sum_i \log\left(p(y_r^i|x_r^i;\theta)\right) - \lambda_a \cdot ||\theta_g^* - \theta||. \quad (4)$$

The summation term maximizes only the personalized conditional likelihood, while the second term prevents large deviations from θ_g^*, with some regularizing constant λ_a. We refer to this formulation as ***adaptive personalization***. Note that depending on the nature of the cost function and classifier in question, such a maximization can still be computationally expensive. Furthermore, with very few personalization samples, estimates of the personalized conditional likelihood may still be unreliable.

3. Learning a Portfolio of Classifiers

We now present our method that side-steps explicitly maximizing the personalized likelihood function in Eqs. (3,4) at personalization time. We learn a set or portfolio of classifiers at training time and choose the one that performs best on the personalization samples. Let Θ denote the parameters of a portfolio of J classifiers *i.e.* $\Theta = \{\theta_j | j = 1...J\}$, where each θ_j is the parameter of classifier j.

3.1. Attribute-Based Portfolios

One way of creating a gesture recognition portfolio is to learn specialized classifiers where each classifier targets a range of attribute values. For example, if height were a relevant attribute, one could fill the portfolio with some classifiers tuned for short subjects and others for tall subjects. A simple way of integrating attributes is to weight the loss of each classifier according to the targeted attribute value. More formally, let each training instance x_s^i be associated with attribute value a_s^i; its weighting $w_{s,i}$ is determined by $m_j(\cdot)$, where m_j could be adjusted to emphasize different values of a for each classifier j.

Algorithm 1 Learning of an attribute-based portfolio

initialize: $\Theta = \{\}$
for $k = 1 \rightarrow J$ **do**
1. $w_{s,i} := m_j(a_{s,i})$
2. $\theta_j^* = \underset{\theta}{\operatorname{argmax}} \sum_{s,i} w_{s,i} \cdot \log\left(p(y_s^i|x_s^i;\theta)\right)$
3. $\Theta \leftarrow \Theta \cup \theta_j^*$

end for

Based on Algorithm 1, one can learn a portfolio of classifiers which may target height, age, gender *etc*. which may affect the movements users associate with certain gestures. For one or two relevant attributes, this is a feasible solution, but for more attributes, Algorithm 1 no longer scales, particularly if one combines the attributes, e.g. learning classifiers dedicated to middle-age, tall, Caucasian women. As such, we propose a more direct approach in Section 3.2.

3.2. Likelihood-Based Portfolios

Our goal is to learn the optimal parameters Θ^* for the portfolio such that for a target user r, there exists $\theta_j \in \Theta$ which is an adequate classifier, *i.e.* maximize $p(y|x;\theta_j)$. We begin with the likelihood in Eq. (1), and define a portfolio objective $L_P(\Theta)$ as the sum, over training subjects, of the *maximum* likelihood for one such classifier in the portfolio on the instances of each subject. More formally,

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}}\, L_P(\Theta) = \underset{\Theta}{\operatorname{argmax}} \sum_s L_s(\Theta), \quad (5)$$

where the subject-specific likelihood $L_s(\Theta)$ is defined as

$$L_s(\Theta) := \max_{\theta_j \in \Theta} \sum_i \log\left(p(y_s^i|x_s^i;\theta_j)\right). \quad (6)$$

Such a formulation is related to the Multiple Choice Learning framework [9], which used a coordinate descent procedure to minimize the set loss.

Greedy Minimization The portfolio objective defined in Eq. (5) is a supermodular set function [14]. Maximization of

super-modular functions are in general NP-hard, even if the set of all solutions is finite. We can use the greedy procedure of Algorithm 2 to approximate a solution. In each iteration of Algorithm 2, the log likelihood l_s from the best classifier so far in the existing portfolio is determined for every subject. A new classifier with parameter θ_k^* is then learned based on the max of the new log likelihood and l_s.

Algorithm 2 Greedy maximization of portfolio objective

initialize:

$\theta_1^* = \underset{\theta}{\operatorname{argmax}} \sum_{s,i} \log\left(p(y_s^i|x_s^i;\theta)\right)$

$\Theta = \{\theta_1^*\}$

for $k = 2 \rightarrow J$ **do**

1. **for all** s **do**
   $l_s := \max_{\theta_j \in \Theta} \sum_i \log\left(p(y_s^i|x_s^i;\theta_j)\right)$
   end for
2. $\theta_k^* = \underset{\theta}{\operatorname{argmax}} \sum_s \left(\max\left\{\sum_i \log\left(p(y_s^i|x_s^i;\theta)\right), l_s\right\}\right)$
3. $\Theta \leftarrow \Theta \cup \theta_k^*$

end for

3.3. Robust Likelihood-Based Portfolios

Note that l_s considers only the *max* log likelihood among the J classifiers in the portfolio for each subject, *i.e.* only one classifier needs to maximize the likelihood on the instances from any subject. The remaining classifiers can have any likelihood and not affect the overall portfolio objective. Such an objective implicitly assumes that we are somehow able to isolate the best classifier for any given subject at test time. This is a reasonable assumption if there are a large number of instances in the personalization set. In this case, we could select the best classifier from the portfolio for each subject with very high confidence by observing the classifier accuracy on the personalization instances.

In real world personalization scenarios, the personalization set is usually quite small; the low number of personalization instances may lead to the selection of a sub-optimal classifier. This observation necessitates a robust version of the portfolio objective, to encourage *every* classifier in the portfolio to have an acceptable log likelihood on each subject. As such, we introduce a threshold ν which encourages all classifiers in the portfolio to have an acceptable log likelihood for every subject, *i.e.*

$$\theta_k^* = \underset{\theta}{\operatorname{argmax}} \sum_s \left(\max\left\{\sum_i \log\left(p(y_s^i|x_s^i;\theta)\right), \min\{l_s, \nu\}\right\}\right) \tag{7}$$

We can simplify the maximization of θ_k^* with a surrogate function that takes the form of a weighted sum of the instances of individual subjects:

$$\theta_k^* = \underset{\theta}{\operatorname{argmax}} \sum_{s,i} w_s \cdot \log\left(p(y_s^i|x_s^i;\theta)\right), \tag{8}$$

$$\text{where } w_s = \frac{\kappa_s}{\min\{l_s, \nu\}} \tag{9}$$

For learning the new predictor θ_k^*, Eq. (8) assigns for each subject a weight inversely proportional to an upper bounded l_s in the current portfolio. Such a weighting is conceptually similar to the user-specific classifier θ_r^* of Eq. (2), but has the added advantage of not having to estimate $d(\cdot)$ by taking a direct likelihood maximization approach. Note that the formulation of Eq. (9) gives all instances of each subject the same weight. Ideally, one would like to adjust the weights according the loss of individual instances per subject, *i.e.* generalize w_s to $w_{s,i}$:

$$w_{s,i} = \frac{\kappa_{s,i}}{\min\{l_{s,i}, \nu\}}, \tag{10}$$

$$\text{where } l_{s,i} := \max_{\theta_j \in \Theta} \log\left(p(y_s^i|x_s^i,\theta_j)\right), \tag{11}$$

resulting in Algorithm 3.

Algorithm 3 Robust greedy maximization

initialize:

$\theta_1^* = \underset{\theta}{\operatorname{argmax}} \sum_{s,i} \log\left(p(y_s^i|x_s^i;\theta)\right)$

$\Theta = \{\theta_1^*\}$

for $k = 2 \rightarrow J$ **do**

1. **for all** s, i **do**
   $l_{s,i} := \max_{\theta_j \in \Theta} \log\left(p(y_s^i|x_s^i;\theta_j)\right)$
   end for
2. $w_{s,i} = \frac{\kappa_{s,i}}{\min\{l_{s,i},\nu\}}$
3. $\theta_k^* = \underset{\theta}{\operatorname{argmax}} \sum_{s,i} w_{s,i} \cdot \log\left(p(y_s^i|x_s^i;\theta)\right)$
4. $\Theta \leftarrow \Theta \cup \theta_k^*$

end for

4. Base Classifier

Our proposed personalization method is general enough that it can be adapted to any type of classifier learned by maximizing a likelihood term. We have chosen a random forest classifier as our base classifier, as they are fast and efficient for training, and more importantly, for testing, making them well-suited for real-time applications. Random forests have been shown to work well for action and gesture recognition in the past, either through a voting framework [20] or by direct classification [6].

We use the direct classification approach of [6] with parameter $\theta = \{\mathcal{F}\}$, where $\mathcal{F}$ is a random forest. Each tree f in the forest $\mathcal{F}$, at time t, classifies input feature data $\mathbf{x}_t = \{x_{t'}, x_{t'+1}, ..., x_t\}$ (composed of observations from

Table 1. Example gestures from the MSRC-12 (left) and 2013 ChaLearn Gesture Challenge (right).

time t' to t) into gesture class $\hat{y}_t$ from the set of possible gesture classes $\mathcal{Y}$. The ensemble of classifications from each tree f in the forest approximates a posterior class distribution

$$p_{\mathcal{F}}(\hat{y} = y|\mathbf{x}_t) = \frac{1}{|\mathcal{F}|} \sum_f \hat{p}_{f,l}(\hat{y}_t^f = y|\mathbf{x}_t) \qquad (12)$$

over $\mathcal{Y}$, where $\hat{p}_{f,l}(\cdot)$ is the distribution stored at leaf node l of tree f that the sample $\mathbf{x}_t$ goes to during testing. The trees themselves are learned according to the standard random forest framework [2], using information gain as the split criterion and empirical class probabilities in the entropy measure. The classified gesture $\hat{y}_t$ at time t is a maximum posterior value. We evaluate recognition performance in the same way as [6] with a balanced F-score.

The computational cost of learning a random forest is directly related to the number of potential splits b evaluated at each node. The cost of each potential split evaluation is in turn directly related to the number of training data samples, as well as the cost of computing potential information gain, *i.e.* $C_{\Delta H}$. For a tree of depth $d+1$ constructed from $M \cdot K$ training samples, the overall computational cost, when it is a full binary tree, can be estimated as

$$C = (M \cdot K) \cdot d \cdot b + \ 2^d \cdot b \cdot C_{\Delta H}, \qquad (13)$$

where the first term is the cost of evaluating all feature splits on all data samples, and the second term corresponds to the cost of computing the information gain of these splits.

4.1. Portfolios of Random Forests

In learning a portfolio of random forest classifiers as per Algorithm 3, one can directly apply w_s or $w_{s,i}$ to the split nodes and influence tree structure and or to the leaf nodes to influence the class posterior. At a split node with data samples X, the empirical class probability $p(y|X)$ can be replaced with $\rho_y(X)$:

$$\rho_y(X) = \sum_X \frac{w_{s,i} \cdot \mathbb{I}(y_{s,i} = y)}{w_{s,i}}. \qquad (14)$$

$\rho_y(X)$ is a normalized summation of the sample weights over all samples in X, where $\mathbb{I}$ is an indicator function equal to 1 if the predicate is true and 0 otherwise. $\rho_y(X)$ can also be similarly substituted at the leaf nodes to adjust the posterior class distribution.

The computational cost of learning the portfolio of random forests is high; for a portfolio of J classifiers, the cost is J times that of Eq. (13). The learning phase, however, can be done offline, before deployment. The benefit comes during personalization, since one only needs to evaluate the N personalization samples at d nodes. The resulting personalization cost is shown in the third row of Table 2.

Type	Personalization Cost
full	$C_f = (M \cdot K + N) \cdot d \cdot b + 2^d \cdot b \cdot C_{\Delta H}$
adaptive	$C_a = N \cdot d \cdot b_a + 2^d \cdot b_a \cdot C_{\Delta H}$
portfolio	$C_p = N \cdot d \cdot J$

Table 2. Comparison of personalization costs (per tree). Such a cost breakdown highlights the efficiency of our proposed portfolio method, especially since $b \gg b_a \gg J$.

4.2. Personalized Baselines

Full Personalization We can incorporate personalization data into the random forest classifier in two different ways. To use $\mathcal{P}_r$ as a regularization term in the likelihood, as per Eq. (3), we learn new forests on a combined training set of the original training data and the personalization data. The regularizing constant λ_f is proportional to the number of times that the personalization data is used when constructing the trees. The computational cost of full personalization is the same as that of learning a forest with $(M \cdot K + N)$ data samples (see first row of Table 2).

Adaptive Personalization To only maximize the personalized conditional likelihood, as per Eq. (4), we adapt the forests learned on the original training data $\mathcal{G}$ to the personalization data $\mathcal{P}_r$ by adjusting the split and leaf nodes. Split nodes are updated by doing a local search around the original threshold, *i.e.* evaluating the new N personalization samples with b_a potential splits and replacing the original split with the the new split which yields the highest information gain on the personalization samples. Leaf nodes can be updated with a weighted mixture of the original and new class posterior from the personalization samples. The regularization constant λ_a is then proportional to the neighbourhood size in which we search for the new threshold as well as the mixing weight of the new label distribution. The computational cost of adaptive personalization has a similar form as learning a forest with N data samples (see Table 2).

5. Experiments

We test our method on two different gesture datasets, both recorded by the Microsoft Kinect. The first is the MSRC-

12 Kinect Gesture Dataset [6], with 12 gestures (see left of Table 1) collected from 30 subjects. In total, there are ∼4900 gesture instances. The second dataset is the 2013 Chalearn Gesture Challenge [4], with 20 gestures (see right of Table 1) collected from 36 subjects. We experiment with the training and valuation data, with ∼11000 instances.

In both datasets, we temporally segment the gesture instances[2] and normalize the segments into 100 and 60 frames for MSRC-12 and ChaLearn respectively. We concatenate the coordinates of the body joints (3 x 20 joints) over the length of the normalized segment as input features. In all experiments, we use a leave-one-subject-out cross validation scheme, and report the mean F-scores over 100 runs.

5.1. Base Classifier

We train forests of 5 trees of maximum depth 5, with 1024 feature tests (b) in the regular training and full personalization and 400 feature tests (b_a) in the adaptive personalization. For MSRC-12, the base classifier has a mean F-score of 0.77 with stdev of 0.09 across classes and 0.07 across subjects. For ChaLearn, which is more difficult than MSRC-12, the base classifier has an average F-score of 0.35 with stdev of 0.18 across classes and 0.10 across subjects. Of the 20 classes, several are subtle hand gestures that are very difficult to identify through skeletal data alone and the F-scores vary widely across the different gestures.

5.2. Personalized Baseline Performance

We plot the mean F-score versus the number of personalization instances for the full and adaptive personalization baselines in Figures 2(a) and 2(b) respectively for MSRC-12 and Figure 2(c)) for ChaLearn. For both baselines, the F-score increases with the number of personalization instances, though less so for ChaLearn than MSRC-12.

Full personalization The parameter λ_f is the number of times that personalization instances are used for constructing the trees and it controls the extent of personalization. For MSRC-12, when $\lambda_f = 1$, personalization has little effect. At $\lambda_f = 1000$, the personalization instances dominate and the classifier tends to over-fit to these instances, hence the very steep growth curve and the lower F-scores than $\lambda_f = 100$. For ChaLearn, the effects of personalization are more dramatic, hence the large jump in F-score for even one personalization instance at $\lambda_f = 1$. Further increase in λ_f has little effect, with over-fitting occurring already with $\lambda_f = 100$, suggesting that the users are highly individual in the way they gesture from each other, but consistent amongst amongst themselves.

Adaptive personalization The two values in λ_a correspond to the neighbourhood search size of the updated threshold and the weight of the personalization instances for determining the label posterior. For both MSRC-12 and ChaLearn, having a smaller neighbourhood size and a larger weighting for the personalization samples is more preferable *i.e.* $\lambda_a = (0.1, 1000)$, though the F-scores are not as high as full personalization. One key difference is that the adaptive personalization, which was almost comparable to the full personalization with MSRC-12, does not improve F-scores much for ChaLearn, again supporting the conclusion that ChaLearn subjects gesture very differently.

[2]On MSRC-12, segments are approximated based on the labelled action points. On ChaLearn, ground truth segments are provided.

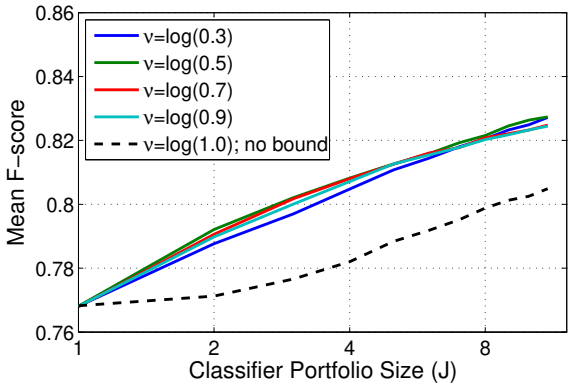


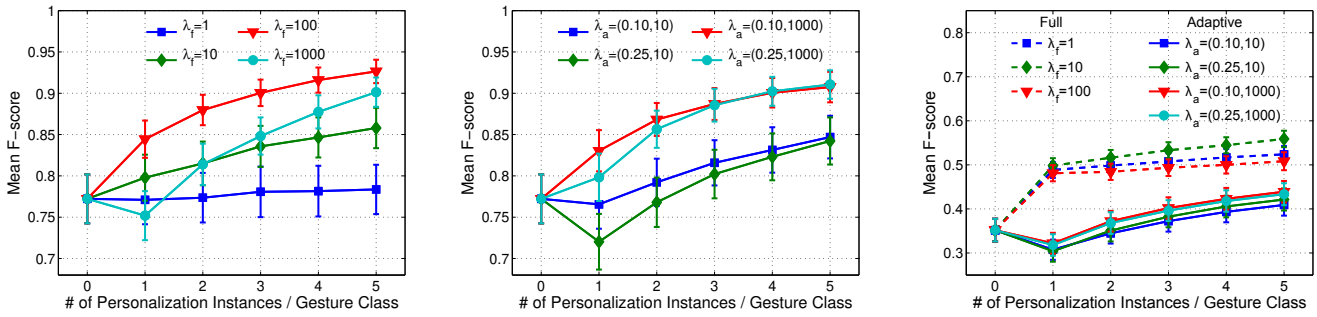
Figure 4. Mean F-scores for MSRC-12, $\kappa_{s,i} = 5$ for differing values of ν. The value of ν has little effect, but without any bounds, *i.e.* when ν=log(1), the performance deteriorates significantly.

5.3. Portfolio Performance

Robust versus non-robust portfolios We first compare the greedy minimization of Algorithm 2 with the robust version of Algorithm 3. In Figure 4, we plot the mean F-score for different values of ν, as applied to Eq. 10. The F-scores are not sensitive to ν; as long as some bound is applied, the maximization is robust. However, there is a significant deterioration when there is no bound, *i.e.* when $\nu = \log(1)$ and is equivalent to Alg. 2. For all subsequent experiments, we use the robust greedy maximization.

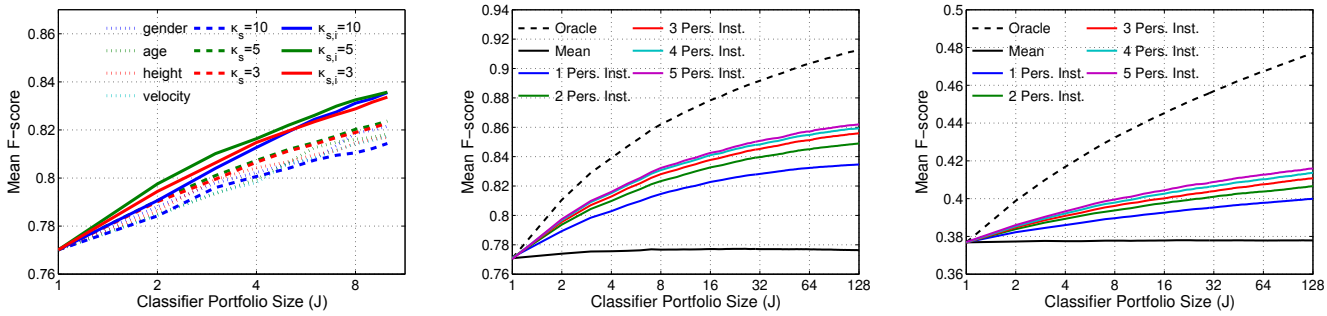
Comparison of portfolio types In Figure 3(a), the MSRC-12 mean F-score is plotted with respect to portfolio size for the attribute-based (see Alg. 1), subject-based (see Alg. 3, Eq. 9) and instance-based portfolios (see Alg. 3, Eq. 10). For each attribute-based portfolio, m_j takes the form of a step functions, emphasizing a subgroup of the subjects. Subjects were separated into either two (according to gender) or three (according to age, height, gesture velocity) subgroups. For the subject- and instance-based portfolios, we set $\nu = log(0.8)$ and vary κ_s and $\kappa_{s,i}$. We find that the attribute-based portfolios have similar F-scores as the subject-based portfolios, while the instance-based portfolios are slightly better, with $\kappa_{s,i} = 5$ being the best.

Since the ChaLearn dataset does not provide any user attributes, we compare only the subject- and instance-based portfolios; we find that the subject-based portfolios have slightly higher F-scores, with $\kappa_s = 3$ being best. The reason



(a) MSRC-12 Full Personalization (b) MSRC-12 Adaptive Personalization (c) ChaLearn Personalization Baselines

Figure 2. Personalized baseline F-scores for MSRC-12(a,b) and ChaLearn(c). In (a) and (c), λ_f is the number of times that personalization instances are used for ***full personalization*** (see Eq. (3)). In (b) and (c), λ_a determines the extent of ***adaptive personalization*** (see Eq. (4)). The first value of λ_a is the search neighbourhood size for the updated threshold in the split nodes, as a fraction of the entire range of possible threshold values. The second value is the number of times the personalization instances are weighted for determining the leaf node label posterior. For both personalization baselines, mean F-score increases with the number of personalization instances both datasets. With MSRC-12, adaptive personalization performs comparably with full personalization, but worse for ChaLearn.



(a) MSRC-12 Comparison of Portfolio Methods (b) MSRC-12 Classifier Selection, $\kappa_{s,i}=5$ (c) ChaLearn Classifier Selection, $\kappa_s=3$

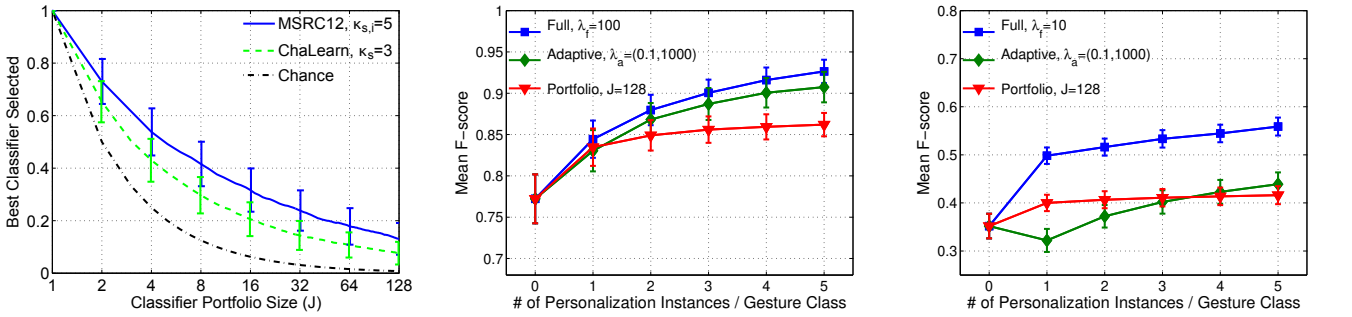
Figure 3. Portfolio F-scores (best viewed in colour). In (a), the portfolio methods are compared, each using five personalization instances per gesture class; the attribute-based (dotted lines) and subject-based (dashed lines) portfolios have similar performance, while the instance-based (solid lines) is slightly better. In (b) and (c), the selected, oracle (best possible classifier) and mean (over entire portfolio) F-scores are plotted for increasing portfolio sizes. F-score increases with more personalization instances since the classifier selection becomes more reliable.

for this is most likely due to the extreme variation in F-scores across both subjects and gesture classes, thus leading to overfitting to specific problematic sequences. We note that our results are incomparable to those in [4], where the proposed methods perform combined gesture and speech recognition.

Portfolio size and classifier selection Figures 3(b) and (c) plots the mean F-score with respect to portfolio size for different number of personalization instances for MSRC-12 and ChaLearn respectively. For both the selected classifier and the oracle, *i.e.* best possible classifier in the portfolio given ground truth, performance increases and slowly saturates with portfolio size J. The increase for ChaLearn, however, is much slower than for MSRC-12, most likely due to the difficulty of the dataset. Finally, the increase in performance can be attributed to the selection of suitable classifiers rather than training better classifiers, the mean performance of the classifiers in the portfolio stays relatively constant.

Selecting a suitable classifier for a given user becomes increasingly more reliable with more personalization instances, hence the increase in F-score. The efficacy of using personalization instances to select the best classifier for both datasets is shown in Figure 5(a). As the portfolio size increases, it becomes more and more difficult to select the best possible classifier, but the fraction of times that it is selected is still higher than chance for both datasets.

Comparison of portfolios to personalization baselines Finally, we compare the best personalization baseline results to the portfolio results in Figures 5(b) and (c). At the expense of more computational power, the personalization baselines do perform better, especially the full personalization. With very few personalization instances (1 or 2), however, our proposed portfolio method can already achieve comparable or even better performance than the adaptive personalization though with significantly higher efficiency. Given the trade-off of performance and user convenience, it is clear that the portfolio method is ideal for applications with limited



(a) Selection of the Best Classifier (b) MSRC-12 Personalization Baselines vs. Portfolio (c) ChaLearn Personalization Baselines vs. Portfolio

Figure 5. In (a), the fraction that the best classifier is selected decreases as the portfolio size increases, but is still above chance for both datasets. Personalization baselines and portfolio F-scores (b,c); the strength of the portfolio method comes when there are very few (1-3) personalization instances. (Best viewed in colour).

computational power and very few personalization instances.

6. Conclusion

We have shown a very efficient way of personalizing gesture recognition. In particular, our method is well suited for applications with limited computation resources at run time, since it does not require any re-optimization of the classification parameters. We have shown that it is possible to achieve good results with very few personalization instances (as little as 1 to 5), though the exact trade-off between personalization accuracy and convenience to the end user, *i.e.* the amount of personalization data that needs to be collected, is most likely system-dependent. Furthermore, as gesture recognition systems are typically designed for interactive applications, there is also the possibility of not only the classifier adapting to the user, but also for the user to adapt to the system. It remains an open research question as to how to guide the user to learn to gesture within the confines of an existing system.

Acknowledgements Partly funded by the European Union Framework Seven project ReMeDi (grant 610902).

References

[1] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *ICML*, 2007.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] S. Connell and A. Jain. Writer adaptation for online handwriting recognition. *PAMI*, 24(3):329–346, 2002.

[4] S. Escalera, J. Gonzalez, X. Baro, M. Reyes, O. Lopes, I. Guyon, V. Athitsos, and H. J. Escalante. Multimodal gesture recognition challenge. In *ICMI*, 2013.

[5] G. Foster, C. Goutte, and R. Kuhn. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Empirical Methods in NLP*, 2010.

[6] S. Fothergill, H. Mentis, P. Kohli, and S. Nowozin. Instructing people for training gestural interactive systems. In *CHI*, 2012.

[7] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real time motion capture using a single time-of-flight camera. In *CVPR*, 2010.

[8] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition. In *ICCV*, 2011.

[9] A. Guzman-Rivera, D. Batra, and P. Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In *NIPS*, 2012.

[10] J. Jiang and C. Zhai. Instance weighting for domain adaptation in NLP. In *ACL*, 2007.

[11] W. Kienzle and K. Chellapilla. Personalized handwriting recognition via biased regularization. In *ICML*, 2006.

[12] C. Leggetter and P. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density HMMs. *Computer speech and language*, 9(2):171, 1995.

[13] S. Mitra and T. Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Trans. on*, 37(3):311–324, 2007.

[14] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[15] M. Raptis, D. Kirovski, and H. Hoppe. Real-time classification of dance gestures from skeleton animation. In *Eurographics Symposium on Computer Animation*, 2011.

[16] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, 2010.

[17] K. Shinoda and C.-H. Lee. A structural Bayes approach to speaker adaptation. *Speech and Audio Processing, IEEE Transactions on*, 9(3):276–287, 2001.

[18] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. In *PAMI*, 2013.

[19] J. Yang, R. Yan, and A. G. Hauptmann. Cross-domain video concept detection using adaptive SVMs. In *of ACM Int. Conf. on Multimedia*, 2007.

[20] A. Yao, J. Gall, and L. Van Gool. Coupled action recognition and pose estimation from multiple views. *IJCV*, 2012.

[21] C. Zhang, R. Hamid, and Z. Zhang. Taylor expansion based classifier adaptation: Application to person detection. In *CVPR*, 2008.