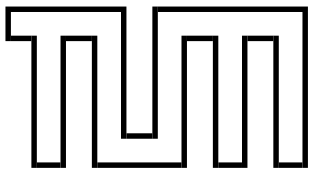


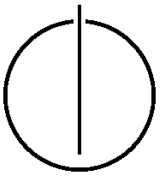
FAKULTÄT FÜR INFORMATIK

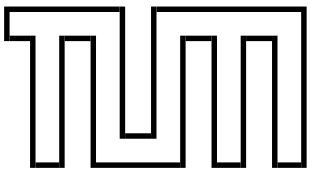
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Online Activity Recognition through Kernel Methods

Evgeni Pavlidis





FAKULTÄT FÜR INFORMATIK

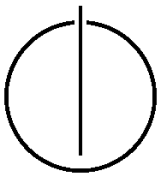
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Online Activity Recognition through Kernel Methods

Echtzeit Aktivitätserkennung mittels Kernelmethoden

Author:	Evgeni Pavlidis
Supervisor:	Prof. Dr. Daniel Cremers
Advisor:	Dr. Rudolph Triebel
Date:	October 13, 2014



Abstract

English abstract

Zusammenfassung

German Abstract

Ich versichere, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den October 13, 2014 Evgeni Pavlidis

Contents

1 Introduction

cite:software packages and tools used

cite:datasets (mocap, daily activities, ms activities)

Check bibliography style and data!!!

1.1 Motivation

Activity recognition is a big research field in computer vision, machine learning and robotics. Being able to infer what human actors are doing helps in many practical robotic tasks. An example is doing short-time prediction for collision-avoidance. Moreover in social robotics it is crucial to know what humans are doing when reasoning about the current state of the robot's environment.

To do pose based activity recognition the human pose has to be inferred for each frame. With the advent and further development of RGBD (color and depth) sensors it is now possible to perform skeletal tracking of persons. Initial versions of such sensor capture the depth by projecting an infrared pattern and computing the depth values for each pixel by the warping. Figure `fig:xtion` shows the Asus XTION.

This allows us to decouple pose estimation and activity learning, which make the problem a bit easier.

For real robotic tasks it is very important that the activity recognition is online i.e. runs in real-time.

Advantages of skeletal features are that no person sensitive information has to be processed by the learning algorithm. In the context of the SPENCER project this is an important prerequisite. Another advantage is also that the skeletal features are very informative for activity recognition. Also beginning to concept an algorithm that builds on top of robust pose estimation reduces the complexity, as we can fully ignore the pose estimation problem.

1.1.1 The SPENCER project

Figure 1.1

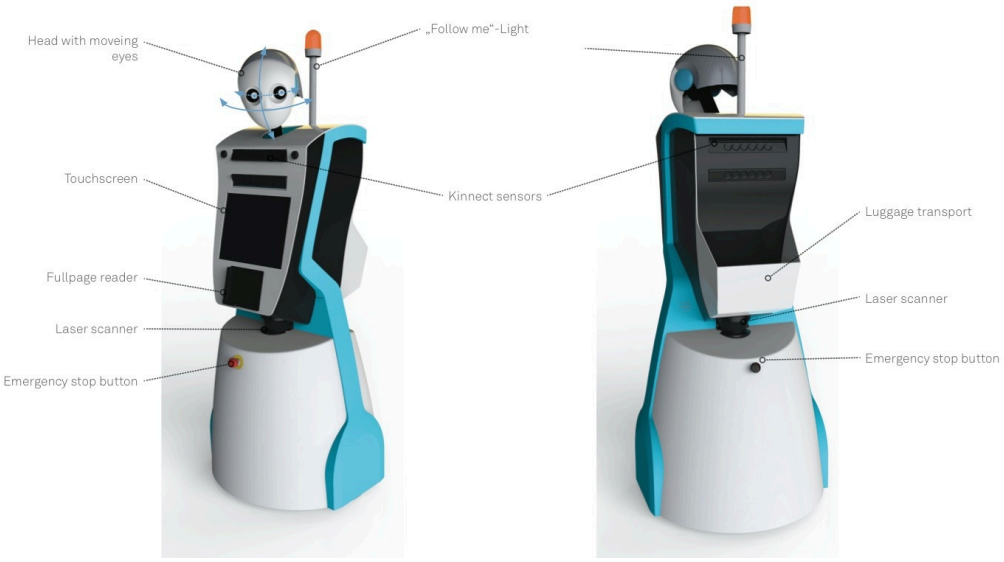


Figure 1.1: The SPENCER robot and its sensors.

make a distinction between online recognition and online learning !!! maybe change online to real-time

make a distinctio between action ar activity

1.2 Problem statement

As stated above we want to devise an algorithm which can classify different activities. These activities can have different nature and should not be confused with actions. The difference being that an action is something that is brief, such as waving or sitting down. An activity can span over a longer time period and can be defined by several sub-actions. An example is cooking where the person can perform some action several times (slicing vegetables) and does not need to always have the same order. Also for online activity recognition we have to classify incomplete data consisting of subsequences. This makes the problem more challenging and reduces the pool of methods that can be used for the classification.

We identify three requirements for a practical activity recognition system:

1. Online recognition
   The algorithm has to be able to do real-time classification as the sequence is progressing. In the SPENCER project this is a hard requirement. The method used should be fast enough to classify the current activity without much lag.

2. Classification of incomplete sequences
   This follows directly from the first requirement. There should be no assumptions about the activities regarding completeness, length or periodicity. Activities should be classified without knowing when they started or ended. This is due to

the fact that most likely the robot will not have the ability to track each person all the time. Therefore it is imperative to be able to recognize activities only by seeing a part of it.

3. Novelty detection
   The algorithm should be able to recognize unobserved activities. First this property will allow for the detection of anomalous activities, which is a important prerequisite many practical applications. In the SPENCER event for example being able to detect activities which highly deviate from a set of learned activities can help in the scenario, as a supervisor can be notified to find out what is exactly is happening.

Optimally it should be possible to also recognize when an unknown activity started and finished. This way automatic segmentation will be possible and will considerably reduce the supervised learning time. In combination with active learning this will greatly reduce training time for practical applications. Unfortunately this problem is very difficult and, to the authors knowledge, there exist no robust method to perform a segmentation of activities.

There are two problems we have to deal with when designing a robust activity recognition algorithm.

High-dimensional input data One big issue with skeleton based activity recognition is that each pose in each frame is defined by a high-dimensional vector. So comparing poses is a non-trivial problem. We thus have to find a way to either deal with this high-dimensional feature space, or reduce this space to a lower one.

Classification of sequences Another big challenge in activity recognition is classification of time series data. In contrast to the simpler sample model, here we have to classify sequences. Therefore appropriate models have to be implemented that take the dynamics into account or use sequence similarity measures.

Evaluate Gaussian processes against different ml algorithms for activity recognition

1.3 Prerequisites and notation

We assume a basic understanding in *Linear Algebra* and *Probability theory* . Although a high-level overview on machine learning is given in chapter 2, deeper knowledge in this field will help understand the rest of this work.

1.3.1 Mathematical notation

- Matrices uppercase
- Vectors lowercase bold
- Constants lowercase
- Parameters lowercase greek letters

1.4 Outline

Introduction This chapter introduced the topic of this work. The motivation and the problem statement are explained.

Background The second chapter summarizes some basic concepts and models that are prerequisites for our approaches. It begins with an overview of machine learning and introduces kernel methods with the *Support Vector Machines* as an example. After that the multivariate Gaussian distribution is described emphasis is led on Gaussian Process Regression and Gaussian Process - Latent Variable Models, which is an unsupervised learning method for dimensionality reduction. Last the Dynamic Time Warping algorithm, which is used for sequence alignment, is explained.

Chapters 3-5 The next two chapters present the approaches that we used to perform online activity recognition.

Approach: K-Means clustering The third chapter presents an implementation of a bag-of-features approach proposed in [19] for classifying daily living activities. This method is then modified in way possible to capture the ordering of different sub-actions. The described methods are then evaluated and contrasted to each other.

Approach: Discriminative sequence back-constrained GP The second approach is an implementation of the "Discriminative Sequence Back-constrained GP-LVM" [1]. The motivation behind this method is explained. Then the method is evaluated on the Cornell Daily Living Activities data set.

Approach: Gaussian Process - Latent Motion Flow The fifth chapter introduces a novel approach for online activity recognition. The approach is inspired by the *Gaussian Process Regression Flow* and models a dense motion flow field inside latent space for each activity. The advantages and problems of this model are discussed.

Results and Outlook The last chapter summarizes the results of the three approaches and gives a brief outlook of future improvements.

2 Background

This chapter introduces some basic concepts needed to understand the proposed approaches. First a high-level overview is given on machine learning and its terminology. Then the Kernel function is explained along with the *Support Vector Machine* - a kernelized learning method. Following is an explanation of *Gaussian Processes*, their different interpretations and properties. After that the *Gaussian Process - Latent Variable Model* is being introduced along with some extensions for learning a backward mapping and optimizing it for discrimination in the case of multiple classes. Last two *Sequence similarities measures* are presented which are used in our implementations.

2.1 Machine Learning

2.1.1 Supervised learning

Supervised learning is the task of classification or regression when the data is labeled i.e. we have the ground truth of every sample. First several features are extracted from the input data. This features should capture the most informative elements of the data. The algorithm then takes the labeled samples (features plus labels) and infers the model parameters (or hyperparameters) accordingly.

There are two distinct cases in supervised learning:

1. **Classification**

   Classification is the task of learning which category a sample belongs to. A prominent example is spam filtering. By taking a large number of emails which are labeled either as spam or as ham (regular email), the algorithm deduces a model which can classify unknown samples into these two categories. As the input data, in this example, is text different features can be extracted. One possibility is to take common words that are included in spam and regular messages and define the feature vector, so that the first element counts the occurrences of the first word and so on.

2. **Regression**

   Regression is a terminus in machine learning and can be understood as function approximation. Here the domain of the sample's label is continuous. An example would be predicting

In most cases we search for a good model that explains the data we have. Parametric models, for example, have a pre-defined model which is parametrized. An example is linear regression where we model a function with the sum of the individually weighted feature elements. The different weights are the parameters. These parameters are learned, such that the model is a good fit for the training data. When searching for an appropriate model it is also important that we try to capture the underlying relationship without compromising the generalization property, which is the ability of the model to correctly predict unseen samples. The case that an algorithm learns the relationship of the data that is used to train the model (training data) but poorly predicts new samples is called overfitting. This means that the model learns not only the data but also fits the noise.

Very often the parameter search is done by maximizing the probability of the data given the model parameters.

$$\arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}) = \arg\max_{\boldsymbol{\theta}} \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}$$

where θ are the model parameters and $\mathcal{D}$ is the data. $p(\boldsymbol{\theta}|\mathcal{D})$ is the posterior which is proportional to the likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ times the prior $p(\boldsymbol{\theta})$.

2.1.2 Unsupervised learning

In contrast to supervised learning in unsupervised learning we have no labeled data i.e. there is no supervisor giving each sample a category (classification) or a value (regression). In this case we can only derive properties of the generation process. Therefore we try to detect patterns in the unlabeled data. These pattern may be clusters of similarity or a lower dimensional generative manifold from which the samples are generated. The last one is called *Dimensionality Reduction* which will be also a subject in this work. [2]

K-means algorithm

An example of an unsupervised learning method for finding a pre-determined number of clusters k in given data is the *k-means* method. The idea is that we first fix the number of clusters and choose k points randomly in the space, which represent a guess of the cluster means (center of mass). After that we try to move these points, such that they align with the real data's k centers of mass. This is done by iterating between two steps:

1. Assign each point $\boldsymbol{x}$ to the closest centroid (cluster mean)
2. Find new centroids by computing the mean of all assigned points for each cluster k

Doing so it is guaranteed that the algorithm will converge, although it could be in a local minimum. [2]

2.1.3 Generative models

Generative methods model the underlying process which generates the data. In Bayesian terms we model the posterior by modeling the likelihood and the prior. Thus more data is needed to find an appropriate model. On the other side the model is very flexible and many attributes have a natural interpretation. An example of this is

generative model example

2.1.4 Discriminative models

A discriminative model is only concerned with modeling the actual posterior. This way fewer samples are needed to find an appropriate model. On the other hand by not taking the likelihood into account the model's ability to generalize unseen data is worse. For this reason discriminative methods are more susceptible to overfitting.

2.1.5 Online learning

Algorithms which can be gradually optimized towards a good solution using streaming batches of samples are considered to do online learning. This means that the model can be updated gradually towards a good solution without having seen all data. Such algorithms are very convenient, as they allow to quickly adapt to the needed data. In context to activity recognition, for example, online learning will allow for the model to improve as new activities are performed and simultaneously labeled. In contrast to online learning online recognition means that the algorithm works in real-time and fast recognition is possible.

2.1.6 Active learning

Very often the bottleneck of powerful supervised learning techniques is that they rely on a large number of correctly labeled data. Since labeling has to be performed by a human it is very difficult and costly to label large amount of data. By identifying more important samples by their information ability of selecting a good model, it is possible to learn a good model using only a subset of all the samples. Letting the algorithm select such samples and query only their labels from a human, who is now actively participating in the learning loop, is called active learning.

Active learning is in practice a convenient way to acquire new informative samples without letting someone go over a huge amount of data to label. In out context, active learning can be used to find activities where the model is uncertain about and query those from a supervisor. This way only relevant activities, which will improve the models ability to perform recognition, will be labeled and learned.

2.2 Kernel methods

Many machine learning algorithms work not with the features directly but instead use only the dot product between features. The dot product between two vectors can be seen as a measure of similarity.

2.2.1 A space defined by sample similarity

Suppose we have n sample points $\boldsymbol{x_i}$ of dimensionality d: $\boldsymbol{x_i} \in \mathcal{R}^d$. When extracting features we try to capture the most characteristic properties of the data for each sample. Let us say that we want to extract m features. Then we have a vector $\boldsymbol{z_i} \in \mathcal{R}^m$ which represents each sample. This means that learning is done in a feature space of dimensionality m. Another space, where we can reason about the data is a similarity space. Suppose we have a function $k(\boldsymbol{x}, \boldsymbol{y})$ which measures the similarity between point $\boldsymbol{x}$ and point $\boldsymbol{y}$, then we can define a vector $\boldsymbol{s}$ of similarities for a new point

This similarity measure is also called a *kernel function*.

A kernel defines a similarity measure between two points $\boldsymbol{x}$ and $\boldsymbol{y}$. The kernel function can be defined as the dot product between two feature vectors.

$$k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$$

where $\phi(\boldsymbol{x})$ is a mapping from the input space (raw data) to a feature space. Note that the dimensionality of the input space and feature space do not have to be the same. Moreover one can define a feature space with infinite dimensions.

If a machine learning algorithm is formulated only in terms of the dot product of two feature vectors, this term can be directly exchanged with a kernel. Therefore we can by computing the kernel value, we can compute the dot product in some feature space, where we do not have to explicitly map the input to this space. Therefore we can work in feature spaces which are high- and even infinite-dimensional. This is called the *kernel trick*.

2.2.2 The Radial Basis Function

An often used kernel is the *Radial Basis Function* also known as the *Gaussian* or the *Squared exponential* kernel for one dimension.

$$k(\boldsymbol{x}, \boldsymbol{y}) = rbf(\boldsymbol{x}, \boldsymbol{y}) = \sigma_f^2 \exp(-\frac{1}{2l^2}\|\boldsymbol{x} - \boldsymbol{y}\|^2) + \sigma_n^2 \delta_{xy}$$

The parameters of kernel functions are called *hyperparameters* as they do not represent direct parameter for the model. It can be interpreted as a measure which gives strong values for points located in the near vicinity of a certain test point. This hyperparameter l represents the length scale which defines how strong the points effect each other, or the strength of the effect a point has with its relative distance. The signal variance σ_n^2

represents how far away the points are located from the mean. The noise variance σ_n^2 captures how strong the noise is of the generative process.

2.2.3 Support Vector Machines

Suppose we have data which is linearly separable. If we have only two features we can draw all samples in a 2D plot. This is shown in Figure 2.1. In this case the *best* line that can separate both classes should be as far apart from all samples as possible, i.e. the gap between both classes should be as large as possible. This line can be defined by the samples that are nearest to it. These samples are called support vectors as they are sufficient to span the boundary. For this reason *SVM* is also called a sparse method as one only needs the support vectors to define the classification boundary. For higher dimensional feature spaces the same idea holds, but instead of having a line we have a plane (a hyperplane) which dissects the space in two parts. As the *SVM* models the boundary between each class without considering any generative process it is a discriminative model.

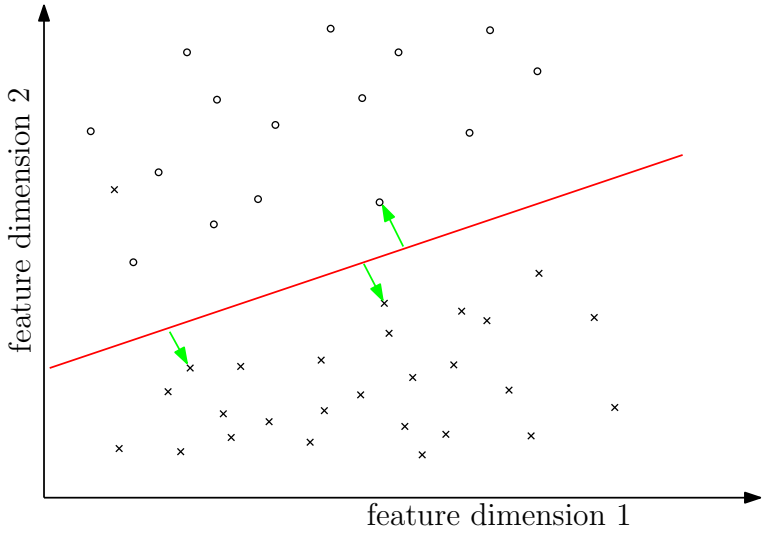


Figure 2.1: SVM decision boundary (red) between two classes (cross, circle). The support vectors are indicated in green.

The assumption that the data is linearly separable can be relaxed in two ways:

As the formulation of the *SVM* works only with dot products of feature vectors, instead of finding a boundary in the feature space we can use the kernel trick to project the data into some other features space. This way the data may not be linearly separable in the feature space, but instead could be linearly separated in some other space. If we take the *Radial Basis Funciton* for example the feature space has infinite dimensions and thus the data can be linearly separated.

We can also allow for a small subset of samples to cross the boundary without compromising its discriminative properties. This is called the *soft-margin SVM*.

The theory behind *SVM* and the fact that the support vectors can be found by optimizing a convex function make this method a very robust way to do classification. For this reason there are multiple implementations of *SVMs* which are very popular and are used very often in practical applications.

2.3 Gaussian Processes

2.3.1 Univariate Gaussian distribution

In the one dimensional case the Gaussian distribution is well known and understood. Moreover many processes in nature can be modeled with this distribution. It is also called the Normal distribution. The probability of an event is very high on a certain "point" (its mean value μ) and it drops quickly on each side with the standard deviation σ. A plot of this distribution can be seen in Figure 2.2.

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma^2}}$$

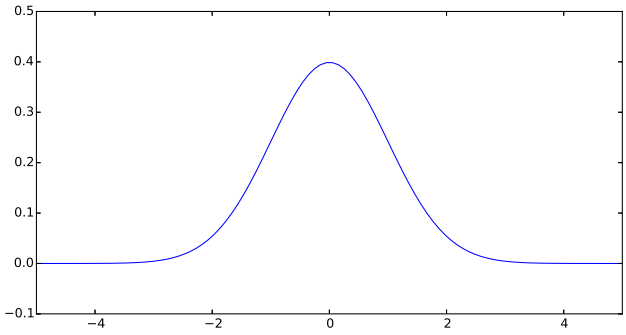


Figure 2.2: The univariate Gaussian distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$

One disadvantage of this distribution which we can see from the above formula is that it can model only one hypothesis. This is also the case for the Gaussian distributions of multiple (multivariat Gaussian distribution) and infinite (Gaussian process) dimensions.

2.3.2 Multivariate Gaussian distribution

The multivariat Gaussian distribution is the generalization of the Gaussian distribution in higher dimensions.

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

The two parameters of the distribution are:

mean $\boldsymbol{\mu} = E[x]$ Representing the most probable vector

covariance $\boldsymbol{\Sigma}$ Representing the mutual variance for each pair of the elements of the random vector: $\boldsymbol{\Sigma}_{ij} = Cov[x_i, x_j]$

The exponent is the mahalanobis distance, which measures the distance of a point to the ellipsoid defined by the covariance matrix.

cite Gaussian Winter School slides Philipp Hennig Gaussian Process Summer School 2014

cite rasmussen

2.3.3 Properties of multivariate Gaussian distributions

Aside for being an appropriate model for many processes occurring in nature, Gaussian distributions are also very nice to work with. One reason GPs are straightforward to work with is the math behind them. It is just linear algebra operations.

Linear maps for Gaussian distributions

If $\boldsymbol{x}$ is a Gaussian random vector:

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu_x}, \Sigma_x)$$

and $y = Ax + b$ then:

$$\boldsymbol{y} \sim \mathcal{N}(A\boldsymbol{\mu_x} + b, A\Sigma_x A^T)$$

We see that a linear map of a Gaussian distributed random variable is also Gaussian.

The marginal and conditional of multivariate Gaussian distributions

If $\boldsymbol{x}$ and $\boldsymbol{y}$ are jointly Gaussian, then we have:

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu_x} \\ \boldsymbol{\mu_y} \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{xy}^T & \Sigma_y \end{bmatrix} \right)$$

The marginal of $\boldsymbol{x}$ is just the part that is defined for it without considering the rest of the mean or covariance matrix, and is thus also a multivariate Gaussian.

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu_x}, \Sigma_x)$$

The conditional of $\boldsymbol{x}$ given $\boldsymbol{y}$ is also Gaussian:

$$\boldsymbol{x}|\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu_{x|y}}, \Sigma_{x|y})$$

where the mean is computed to $\boldsymbol{\mu_{x|y}} = \boldsymbol{\mu_x} + \Sigma_{xy}\Sigma_y^-1(\boldsymbol{y} - \boldsymbol{\mu_y})$ and the variance is $\Sigma_{x|y} = \Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^T$.

2.3.4 From multivariate distributions to Gaussian Processes

Consider the multivariate Gaussian distribution. If we want to model the distribution of a discrete function defined over a finite interval, we can treat each element of the vector $\boldsymbol{x}$ as a point of the function. Thus we can view the multivariate Gaussian distribution as a probability density function over the function space. Letting the dimensionality d go to infinity (the distance between each point goes to zero) we can model continuous functions.

In this case the mean is a point in function space, thus a function:

$$m(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{x}] = f(x)$$

And because of the fact that we now have infinite dimensions the covariance can be seen as an "*infinite matrix*", thus a function of two elements:

$$Cov(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{y}) - m(\boldsymbol{y}))]$$

This can also regarded as a kernel as discussed in `Kernel Methods`. Therefore a *Gaussian Process* can be interpreted as a Gaussian distribution over function space.[12]

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

A Gaussian Process is therefore defined by its mean and covariance functions. The mean is the most probable function and the variance defines the second moment of the distribution.

2.3.5 Prediction

With *Gaussian Processes* we do not learn a model, but instead we have a probability over infinitely many models with the mean being the most probable one. The kernel function defines the relation between two elements of the function.

Prediction with a *Gaussian Process* is simply a matter of computing the posterior *Gaussian Process* given a prior *Gaussian Process* and the data. The marginalization property is what makes this feasible as it lets us compute with a finite part of the covariance function – which can be seen as a covariance matrix. We can discard the infinite part of the mean and "covariance matrix" and work only on the parts where we have real data.

Suppose we have m new points X_{new} and we want to predict the values $ f({$x_{new,i}$}) = $y_{new,i}$. If we do not consider that the observation can be noisy, then we have a joint multivariate distribution of the form **[?]**:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y_{new}} \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}\right)$$

Here K is the $n \times n$ covariance matrix evaluated for all training point pairs, K_* is an $m \times n$ matrix for all combinations between training and new points, K_{**} is a $m \times m$ co-

variance matrix for the new points. The posterior can be computed by using the earlier formulas of the conditional for multivariate Gaussian distributions (see `Properties of Gaussian distributions`).

$$\boldsymbol{y_{new}}|X_{new}, X, \boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{\mu_{new}}, K_{new}\right)$$

Where $\boldsymbol{\mu_{new}} = K_* K^{-1} \boldsymbol{y}$ is the posterior mean and $K_{new} = K - K_* K^{-1} K_*^T$ is the posterior variance. This distribution is called the /predictive distribution.

The case where we want to also model noise is similar, except that we have a changed covariance matrix $K_{\text{with noise}} = K + \sigma^2 I$.

We see that we have to invert the covariance matrix K, which is of dimensions $n \times n$. Therefore this operation has a runtime complexity of $\mathcal{O}(n) = n^3$ which is also the bottleneck of the whole algorithm and a serious drawback of Gaussian Processes.

2.3.6 Learning the hyper-parameters

A *Gaussian Process* is a non-parametric model and is governed by the hyperparameters of the used kernel. In the case of a *GP* the training phase is different than in parametric models, where the model parameters are inferred from the data. Training in the case of GPs means finding good hyperparameter for the kernel, by reducing the marginal log-likelihood in respect to the data by variational optimization. The marginal likelihood over each function value $\boldsymbol{f}$ can be written as:

We have the following *Gaussian Process*, again we set the mean to zero:

$$f(\boldsymbol{y}) \sim \mathcal{GP}(\mathbf{0}, k(\boldsymbol{x}, \boldsymbol{x'}))$$

For the marginal likelihood we only need the finite part of the covariance hence we have to maximize the likelihood of a multivariate Gaussian.

$$p(\boldsymbol{y}|0, K) = \frac{1}{(2\pi)^{\frac{n}{2}} |K|^{\frac{1}{2}}} \exp(-\frac{1}{2} \boldsymbol{y}^T K^{-1} \boldsymbol{y})$$

which results in the following log-likelihood:

$$\log \mathcal{L}(\boldsymbol{y}, K, \boldsymbol{\theta}) = \frac{1}{2} \log(|K|) - \frac{y^T K^{-1} y}{2} - \frac{n}{2} \log 2\pi$$

where K is dependent on $\boldsymbol{\theta}$. This log-likelihood is to be maximized in respect to θ. This means the minimization of the following energy function.

$$E(\boldsymbol{\theta}) = \frac{y^T K^{-1} y}{2} + \frac{1}{2} \log(|K|)$$

The first term can be interpreted as the data fit term, which tries to explain the data with the best possible covariance. The second term is a regularizer of the covariance.

In contrast to parametric models Gaussian processes are less prune to overfitting because of the covariance regularizer term.

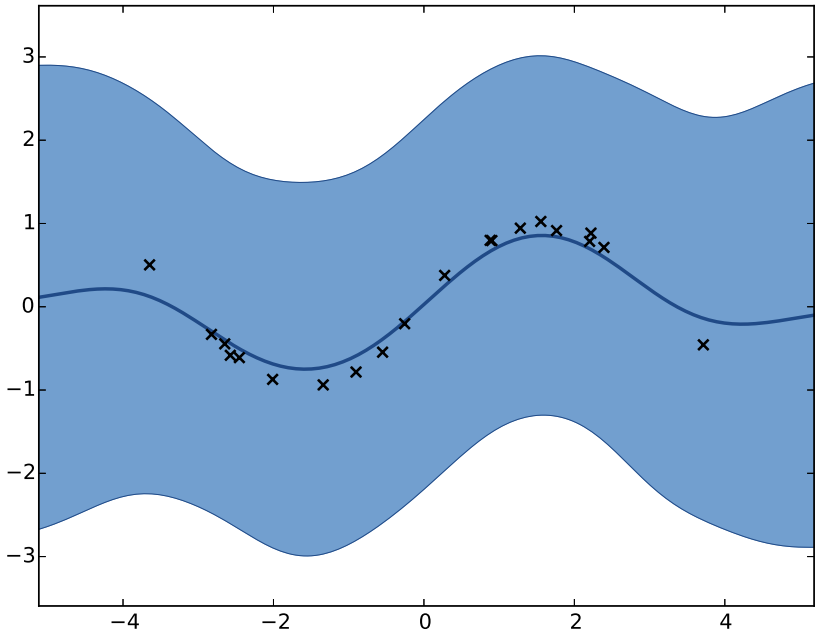


Figure 2.3: Gaussian Process Regression of the sinus. Sample points were sampled with additional 5% noise. The regression was done with an RBF kernel function with length scale 1.0 and variance 1.0.

Figure 2.3 shows generated samples form the sinus function with added 5% noise. We used the GPy library to plot the confidence intervals. The mean of the *Gaussian Process* regression is indicated by the blue line. The blue area represents the standard deviation for the point above and below the mean function value. An *RBF* kernel was used. First initial guess of the hyperparameters was 1.0 for the length scale and 1.0 for the variance. We see that the unoptimized hyperparameters result in a good mean value. But the variance is too big and has to be improved by optimizing the log-likelihood in respect to the sampled points.

Figure 2.4 shows the mean and the confidence after the optimization.

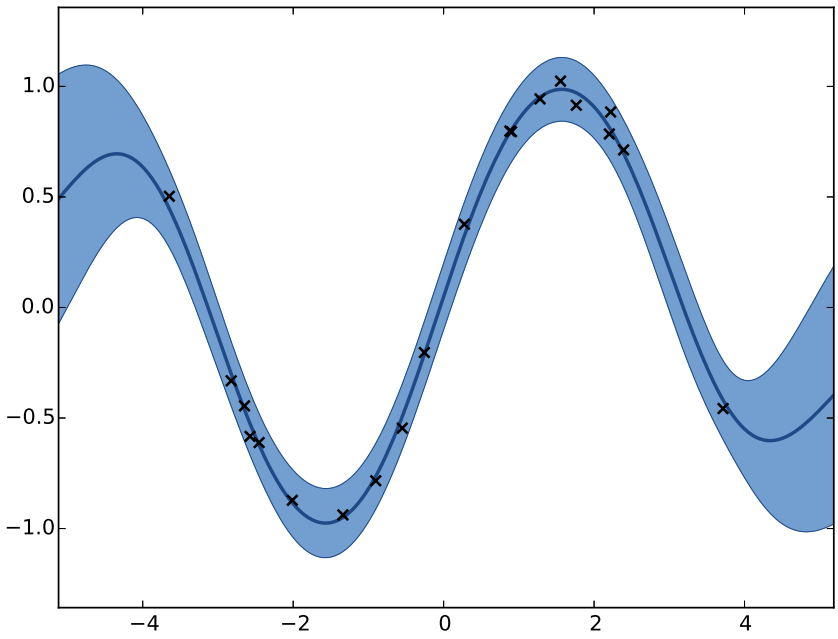


Figure 2.4: Gaussian Process Regression of the sinus function after optimization of the hyper parameters. Length scale: 0.1, Variance: 1.16.

2.3.7 Advantages

1. linear algebra operations As we have seen earlier the conditional and marginals are all Gaussian distributions themselves and thus they can be computed using simple linear algebra operations.

2. non parametric When using a parametric model one has to make sure that the chosen model is sufficiently complex to fit the data but at the same time is not too complex that it will overfitt the training data. This is a very hard task and is in most cases done through cross-validation of the model with an independent validation set. As discussed above GPs are less prune to overfitting and therefore we do not need to reduce the training data to create a validation set. Moreover we do not have to specify the underlying model.

3. probabilistic Being a probabilistic model *GPs* have Bayesian interpretation. The hyperparameters can be interpreted. The lenghtscale controls how much neighboring points contribute to the covariance of the function. The signal variance represents the expected distance from the mean. Due to the variance they also

have a notion of uncertainty. This in turn allows *Gaussian Process* to be used for active learning.

4. generative As the *GP* models the likelihood and has a prior. Moreover different priors can be used if one has knowledge about the problem domain.

2.3.8 Disadvantages

1. susceptible to outliers One big problem of the Gaussian distribution is that it has the assumption that the noise is Gaussian. When this assumption does not hold and we have several outlier it either shifts the mean un-proportionally to itself or raise the variance to be able to explain the outliers. The student-t distribution, for example, is robust against outliers but is much harder to deal with, because the operations required to compute a posterior (marginals, conditionals) are no more simple linear algebra operations.

2. unimodal Since the Gaussian distribution is concave it can model only one hypothesis. This is a curse but also a blessing since the math behind it is simple and unambiguous.

3. high memory complexity Because of the fact the a *Gaussian Process* has to remember all sample points the memory increases proportionally to the sample size n.

4. high computational complexity A run-time complexity of $\mathcal{O}(n^3)$ is a serious drawback, as it is not practical to use *Gaussian Process* with data which has many samples.

5. non-convex optimization of the hyper-parameters The optimization of the hyper-parameters is difficult, and it is likely that it will not find the optimum solution. This means that in many cases one has to set these hyperparameters by hand or use heuristics to be able to find good starting values.

2.3.9 Sparse Methods

As the computation cost for inverting the covariance matrix is cubic, there are some methods which approximate the solution. One of these methods is the *Informative Vector Machine* [7] where a subset of samples is selected by maximum entropy. This samples are used as an active set, which can explain the rest of the data, and using it will still result in a good model.

This reduces the complexity to $\mathcal{O}(d^2 n)$ where d is the number of the active set and n is the number samples. In case of classification, there is also an *IVM* method which works for multiple classes.[13]

2.4 Gaussian Process - Latent Variable Model

Latent Variable Models are unsupervised learning models to perform dimensionality reduction from an observed space $\boldsymbol{y} \in \mathcal{R}^d$ to a latent space $\boldsymbol{x} \in \mathcal{R}^l$. We can combine n samples of the observed space in the matrix $n \times d$ Y. We want to find the latent points, i.e. the $n \times l$ matrix X where $l < d$.

2.4.1 Probabilistic Principal Components Analysis

cite ppca
cite gp
winter
school

The *Probabilistic Principal Components Analysis (PPCA)* is a linear latent variable model which combines the observed and latent space through a linear Gaussian relationships:

$$\boldsymbol{y}_i = W\boldsymbol{x}_i + n_i$$

here the i-th observation sample $\boldsymbol{y}_i$ is a linear map of the i-th latent point $\boldsymbol{x}_i$ plus added noise n_i. We take a Gaussian prior for the noise with an isotropic covariance:

$$n_i \sim \mathcal{N}(0, \sigma^2 I)$$

If we assume independent, identically distributed samples we get the following conditional over all samples:

$$p(Y|X, W) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{y}_i|W\boldsymbol{x}_i, \sigma^2 I)$$

Further we assume a Gaussian prior over the latent space (again with iid. assumption):

$$p(X) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{x}_i|0, I)$$

With these priors we can integrate out the latent points:

$$p(Y|W) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{y}_i|0, WW^T + \sigma^2 I)$$

The linear parameters can be found by maximizing this likelihood. This is equivalent to the closed form solution using *Singular Value Decomposition*.

2.4.2 Dual Probabilistic Principal Components Analysis

In [8] Lawrence noticed that, instead of integrating all latent variables, it is possible to integrate out all parameters .

We assume now that W has a Gaussian prior:

$$P(W) = \prod_{i=1}^{d} \mathcal{N}(\boldsymbol{w}_i|0, \boldsymbol{I})$$

Here d is the dimensionality of the observed space. Then all parameters are integrated out, leading to:

$$p(Y|X) = \prod_{i=1}^{d} \mathcal{N}(\boldsymbol{y}_i|0, XX^T + \sigma^2 I)$$

This is stated in the paper as the dual representation of the previous approach and is called the *Dual Probabilistic PCA*.

Lawrence also noted that the covariance matrix can be interpreted as a linear kernel $XX^T + \sigma^2 I = K$.

$$P(Y|X) = \prod_{i=1}^{d} \mathcal{N}(\boldsymbol{y}_i|0, K)$$

Thus the *Dual Probabilistic PCA* can be interpreted as product of *Gaussian Processes* with a linear kernel.

By exchanging the linear kernel with a non-linear one, we automatically have a technique for non-linear dimensionality reduction, which is has a probabilistic interpretation [6]. The general non-linear model is called the *Gaussian Process - Latent Variable Model*. The *Gaussian Processes* learn a mapping from latent space to observed space ($\mathcal{N}(\boldsymbol{y}_i|0, K)$).

Using the trace properties

$$tr(a) = a \text{ , a is a scalar}$$

$$tr(AB) = tr(BA)$$

we can change the mahalanobis distance term $\boldsymbol{y}_i^T K^{-1} \boldsymbol{y}_i = tr(\boldsymbol{y}_i^T K^{-1} \boldsymbol{y}_i) = tr(K^{-1} \boldsymbol{y}_i \boldsymbol{y}_i^T)$

The log likelihood is, thus:

$$logp(Y|X) \propto -\frac{p}{2} \log K - \frac{1}{2} tr(K^{-1} Y Y^T)$$

In the general case, when there is no linear kernel any more, this equation cannot be solved in closed form. Therefore we have to do gradient based optimization. But marginalizing over all latent space samples means that we have to include these in the optimization. This fact makes the optimization problem very hard as the dimensionality is high-dimensional – number of samples n times number of latent dimensions l + hyperparameters – and, for general problems, has many local minima.

As initially proposed the standard way of initializing the latent space is using *Principal Components Analysis*.

2.4.3 Back-constraints GP-LVM

One problem with this model is that it does not preserve local distances in the latent space. This is because it tries to explain the data by moving distant samples from the observed space also far apart in the latent space. This can be explained by the fact the method tries to transfer the variances in the data to the latent space. This problem is addressed by Lawrence et al. in the back-constrained GP-LVM [9]. A mapping $g_i(\boldsymbol{y}_i) = \boldsymbol{x}_i$ is introduced which constrains the points in latent space to be more near if they are also near in observed space. Instead of optimizing directly on X the back-constrained GP-LVM optimizes the mapping with a number of mappings:

$$x_{i,j} = g_j(\boldsymbol{y}_i, \boldsymbol{\gamma})$$

Where each mapping is a constraint for the j-th element of the every latent point x_i with some parameters $\boldsymbol{\gamma}$. A proposed constraint in the paper is the RBF kernel $k(\boldsymbol{x}, \boldsymbol{y})$:

$$g_j(\boldsymbol{y}_n, A, l, \sigma) = \sum_{i=1}^{n} a_{j,i} k(\boldsymbol{y}_n, \boldsymbol{y}_i)$$

This can be interpreted as a *Radial Basis Funciton Network*. Here the parameters are the weights for the individual samples $a_{j,i}$ and the hyperparameters of the kernel. Instead of optimizing over the latent space we now have to optimize with respect to these parameters.

Having this back-constraints also gives us a mapping from observed space to latent space which can be used to project a new sample into the latent space without costly maximum likelihood estimates.

2.4.4 Discriminative GP-LVM

Another improvement in the context of classification in latent space is the Discriminative GP-LVM [16]. Using a *General Discriminant Analysis* criterion a prior is being enforced on the latent space which ensures that samples from one class are more clustered and different classes are more separated. This is done by maximizing the between-class separability and minimizing the within-class variability while optimizing the log likelihood of the GP-LVM.

2.4.5 Advantages

1. non-linear The GP-LVM performs a non-linear dimensionality reduction and is therefore suitable for many applications where using linear methods do not give good results.

2. generative New points from the latent space have can be mapped to observation space. Thus it is possible to generate and simulate data.

3. probabilistic Because of its probabilistic nature GP-LVM interpolation between two data sample is very natural. [11] Also it is more robust to sample noise.

4. uncertainties The properties of *Gaussian Process* transfer also directly to this method. We have a measure of how certain the algorithm is inside the latent space. Points that are located in the vicinity of observed samples will have a lower variance. Points far away will have high variance and thus a big uncertainty.

2.4.6 Disadvantages

1. No mapping from observation space to latent space The idea of the GP-LVM is to learn a mapping from latent space to observation space by marginalization over the latent space. Resulting from this is that we do not have an inverse mapping into the latent space. This fact may be of no importance for character modeling and motion interpolation but in our case it is crucial. An inverse mapping can be computed by using the back-constrained GP-LVM described in 2.4.3. However one should also keep in mind that using back-constraints inherently changes the latent space as it employs an additional constraint on the mapping.

2. Very hard optimization problem Resulting from the disadvantages of Gaussian Process regarding the optimization of the hyper-parameters the GP-LVM is also very hard to optimize as its objective function is non-convex. But in the case of GP-LVM we have a much larger optimization space due to the fact the we do not optimize only the hyper-parameters, of the mapping Gaussian Process, but also the latent space itself.

   This in fact is the biggest problem as it limits its use on real world data, because for more complex manifold structures there will likely be many local minima. For this reason it is crucial to choose a good initialization. Examples are PCA, Local Linear Embedding or ISOMAP.

2.4.7 GP-LVM for human motion modeling

As the space of human motion is high-dimensional (spatio-temporal) dimensionality reduction is crucial for a number of models dealing with human motion (e.g. [**?**]). The GP-LVM preserve the distances in the mapping and are therefore suitable to model human motion with high noise of the poses.

Analogy LVM ¡-¿ marionettes

see Urtasun DGPLVM Newest addition is [4]

2.5 Sequence similarity measures

2.5.1 Dynamic Time Warping

The *Dynamic Time Warping* is an algorithm which tries to find a minimal path between two sequences where the path can be warped in the time dimension. The sequences can be of arbitrary length.

The recursive definition – excluding some corner cases – reveals the workings of this method.

$$\text{dtw}_{x,y}(i,j) = \text{dist}(x_i, y_j) + \min \begin{cases} \text{dtw}_{x,y}(i-1,j) \\ \text{dtw}_{x,y}(i,j-1) \\ \text{dtw}_{x,y}(i-1,j-1) \end{cases}$$

Where dist(x, y) is a distance function which tells how close two points are, and i and j are the element indices for the first and second sequence. The DTW can be computed with dynamic programming and has a runtime complexity of $\mathcal{O}(nm)$ where n, m are the lengths of the two sequences.

It is closely relates to the *Longest Common Subsequence*, where, instead of minimizing the total warping cost between both sequences, we try to maximize a common subsequence.

Since we are not interested in the path itself but in the cost of the minimal path we define the DTW as a mapping from two time series to an real value. We consider DTW to be a distance which is not entirely correct as the triangle inequality does not hold. Nevertheless it gives us a notion of how similar two time series are and since it is non-negative ($d(x, y) >= 0$), symmetric ($d(x, y) = d(y, x)$) and respects the identity property ($d(x, x) = 0$) it can be used to define a meaningful, be it not formally correct, kernel. [14]

2.5.2 Longest Common Subsequence

The *Longest Common Subsequence* algorithm finds the biggest non-consecutive subsequence that is contained inside two sequences. Its recursive definition is:

$$\text{lcs}_{x,y}(i,j) = \begin{cases} \text{lcs}_{x,y}(i-1,j-1) + 1 \text{ if } x_i = y_i \\ \max(\text{lcs}_{x,y}(i-1,j), \text{lcs}_{x,y}(i,j-1)) \text{ otherwise} \end{cases}$$

This can be implemented using dynamic programming and has a run-time complexity of $\mathcal{O}(nm)$ where m and n are the lengths of the sequences. Several algorithms exist which reduce this complexity by making some kind of assumptions about the data .

cite source survey LCS

3 Approach: KMeans clustering approach

As a starting point, we choose to re-implement a working method with a good performance on this data set. Therefore we choose an existing algorithm based on the *bag-of-features* approach published in 2012 [19].

The idea is illustrated in Figure 3.1:

- Define features which capture the structure in a time instant along with the local displacement of the skeleton. There are two types of features that are extracted. First the structural configuration is captured by the difference vector between each joint pair. Second the local motion is captured by the difference vector for frame t and $t-1$ for each joint. This way the feature represents the current configuration and the current motion performed for every frame. The feature vector is of size 360.

- From all poses find the most k prevalent ones. This means clustering the feature space and finding the mean vectors for each cluster. This is done by the K-Means method.

- Quantize each activity by these poses. For each activity, each frame is being mapped to a cluster mean by nearest neighbor. Doing so we have a sequence of the mean poses for each activity. This step together with the previous one is also known as sparse-coding.

- Compute a fixed sized vector that represents the distribution of each mean pose. By computing the histogram over the previous sequence and normalizing we capture the occurrence of each pose representing the feature clusters (bag-of-features).

- Perform classification using this new feature vector. Using a linear *Support Vector Machine* we learn the activities along with their corresponding labels.

The above algorithm works very well in practice. This can be explained by the fact that the mean poses are very distinct for different activities. This means that they capture the most discriminative poses of the activity which can be robustly recognized.

This methods solves the problem of high-dimensionality of the poses by sparse-coding. The sequence classification issue is solved by histogram pooling.

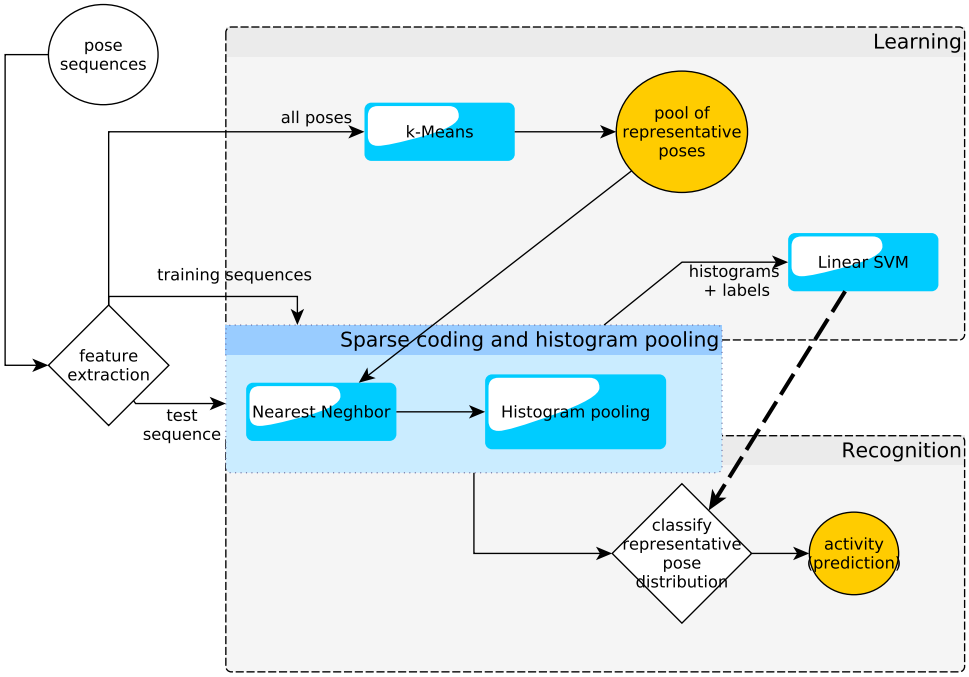


Figure 3.1: Illustration of the k-Means clustering and bag-of-features approach for activity recognition [19]. A code book is created from all poses. Recognition is performed on histograms over the sparse codes from the sequences.

3.1 Cornell Daily Living Activities dataset

We will use the "Cornell Activity Datasets (CAD-60 & CAD-120)"[1] to learn and evaluate the performance of our implementation. This dataset is challenging as it contains complex daily living activities, some of which are very close together. There are four persons each performing 13 activities. The activities *brushing teeth*, /brushing mouth

One person in the data set is left handed and therefore the recognition ability drops considerably in this case. One way to make the method more robust for this case is to also learn the mirrored data. We do not use this approach as we wanted to compare our extensions with the original paper.

The data set consist of an sequence of frames which include:

- Image data
- RGBD data

[1]Human Activity Detection from RGBD Images, Jaeyong Sung, Colin Ponce, Bart Selman, Ashutosh Saxena. In AAAI workshop on Pattern, Activity and Intent Recognition (PAIR), 2011.

- Skeleton information: (joint position and orientation)
- annotated meta information (e.g. activity)

3.2 Robot Operating System (ROS)

The *Robot Operating System* is a middleware which is intended to consolidate and define a layer for the implementation of complex robotic systems. It has a variety of drivers for different sensors and actors and defines a *node* based interface between different sub-modules. cite ROS

Each node can define a communication interface by defining message types, topics and services. This way a complex system is split in several small nodes and, because of this modularization, it is easier to add, exchange, work on and test different parts and functionality. The nodes can communicate using either pre-defined *topics* which have a message type or *services* which can also have some own defined type. A node can subscribe to a topic and each message that is then published on this topic will result in a callback.

ROS also implements a system of transformations between different coordinate systems. Such coordinate frames can be defined for joints, sensors or objects in the environment. This system is called *TF*. Nodes can broadcast such transforms and other modules can listen to these broadcasts. Also transforms which are not directly specified are calculated by ROS and can be listened to.

A *bag* file captures all messages which are being send along with the whole topic net. This way real world data can be recorded and be played back. This is very convenient for debugging or system integration.

3.3 Implementation

For the implementation we used Python with the scipy and scikit-learn libraries. For K-Means we used the mini-batch implementation which is expected to perform worse than the passive variant, but also is much faster. As described in the paper we also used a linear SVM with an *RBF* kernel for classification.

3.4 Integration into ROS

For real time extraction of the skeleton we used the $\text{openni}_{\text{tracker}}$ module. /todo{cite} This module reads the values of the OpenNI nite /todo{cite} skeleton tracker driver and transforms the coordinates to a ros specific depth camera frame. Then it publishes these transforms as a TF message. Because of the fact that the TF broadcasts of the joints are not synchronized we modified the module to publish the pose as an atomic message containing the skeleton positions for each frame. We also did not use any

transformation, as we wanted to use the Cornell data set which is recorded with the raw data coming from the RGBD sensor. This way we could test the performance of the algorithm for online recognition without tedious creation of a new data set.

We publish the pose on the topic `/openni_tracker/pose` having the message type of an array of float32.

We implemented a new ros module called $activity_{recognition}$ which subscribes to the above topic saves a number of poses and every three seconds performs a classification on the sequence. As the provided dataset is relatively large the learning time is several minutes. The most time takes to parse the data files and extract the features. As we did not want to do this every time, we serialized a learned model and loaded it every time the module starts. This way it is also possible to load different learned activities and begin the recognition without waiting for the model to be re-learned.

3.5 Shortcomings

The skeleton tracking is very noisy. We observed very big variations between subsequent frames. Therefore we performed a discrete Gaussian filter smoothing for each sequence. Unfortunately the recognition rate did not improve in our tests.

We observed that the number of prevalent poses is not sufficient to capture the variances inside some classes. For this reason we performed K-Means for each class of activities separately and used the ball-tree nearest neighbor algorithm to quantize the sequences for the recognition. With this it is more likely that same activities will fall to the same representative poses as they are more evenly distributed between the classes. Moreover this allows us to extract more mean poses as the K-Means algorithm has to run only on the samples of each class separately.

The *bag-of-features* approach performs very well but it does not capture the order of the underlying poses. Instead by performing histogram pooling, it has a notion of how prevalent each pose is for every activity.

To circumvent this we modified the method to classify with the *Longest Common Subsequence (LCS)* algorithm. Instead of performing a histogram pooling we classify each quantized sequence using the average *LCS* distance for each class. The standard algorithm for the *LCS* for two sequences is implemented, just like in the case of the *DTW*, with dynamic programming. As described earlier there exist more complex algorithms which reduce the run-time complexity. In our case this was not needed as we already know that different activities will, for the most part, contain different poses. For this reason we can simply remove all elements which are not in the intersection of both sequences as a pre-processing step. This allowed the algorithm to run in real time.

A second idea was to compare the sequences using *Dynamic Time Warping*. For this we chose as a measure between each mean pose the euclidean distance in feature space, which will give a good approximation in the case that the clusters are located far away. As the *DTW* has a complexity of $\mathcal{O}(n*m)$ we took every fifth element from the sequence

for the calculation. Also by pre-computing the distance matrix the distance operation is a simple look-up operation and the algorithms was fast enough.

Regardless of the extensions one serious drawback of this approach is that only a fixed time interval can be classified. There is no way to robustly recognize transitions between different activities. For this reason we tried another approach which uses *GP-LVM* to reduce the feature space and can find the centroid for an activity in this space. This method is described in Chapter 4.

3.6 Evaluation

We performed 4-fold cross validation using each person as test data and the other three persons for training. With our implementation we achieved a comparable precision rate of 84% and recall rate of 84% as stated in the original paper. The confusion matrix is shown in Figure 3.2. We can see that the algorithm is confused by some similar classes, such as ”talking on the phone”, ”drinking water” and ”brushing teeth”.

We also tested the prediction rate using only 100 frames for the prediction. By uniformly sampling 50 intervals from each test sequence the algorithm achieved a surprising average accuracy and precision of 88%. The confusion matrix is shown in Figure 3.3 This can be explained by the fact that we sample 100 frames 50 times from the same test activity.

Using the *LCS* measure we achieved a precision rate of 90% and an accuracy of 88%. For the code book we extracted 64 clusters from each activity class, resulting in $64 * 13 = 832$ representative poses. The confusion matrix in Figure 3.4 indicates that the recognition for three difficult classes (talking on the pone, drinking water, brushing teeth) is better. This can be attributed to the fact the the *LCS* also discriminates using the order of the clustered poses instead of only relying on their distribution.

A comparable result was also achieved using the *DTW* distance (Figure 3.5).

It can thus be argued that the most discriminative information for the classification task is inside the powerful features that are extracted and the representative poses produced by the clustering. By only knowing these poses classification of complex activities is possible.

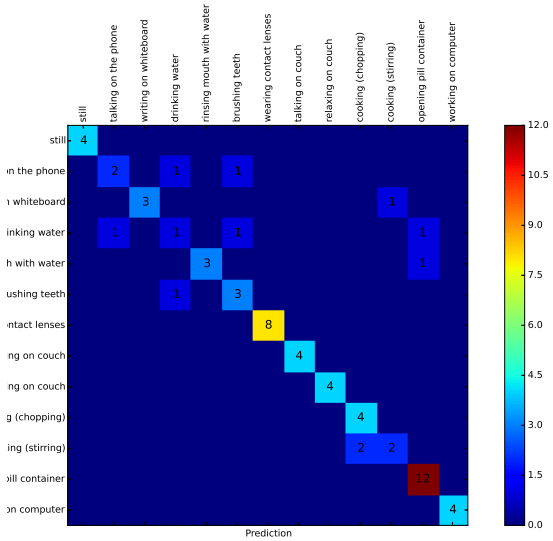


Figure 3.2: Confusion matrix: Bag-of-features approach with 128 clusters. precision 84%, recall 84%

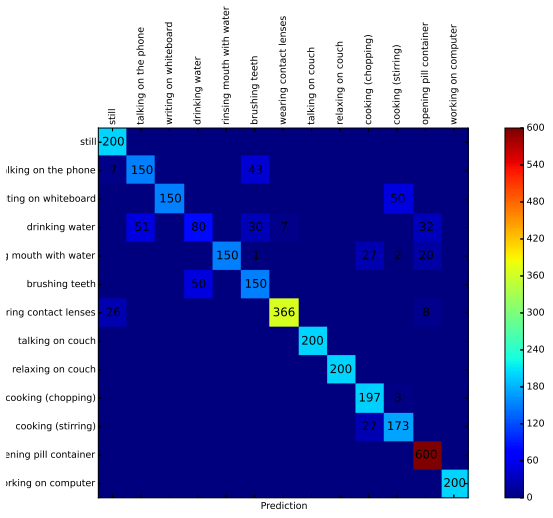


Figure 3.3: Confusion matrix: Bag-of-features approach with 128 clusters tested on intervals of 100 frames sampled 50 times from each test sequence. precision 88%, recall 88%

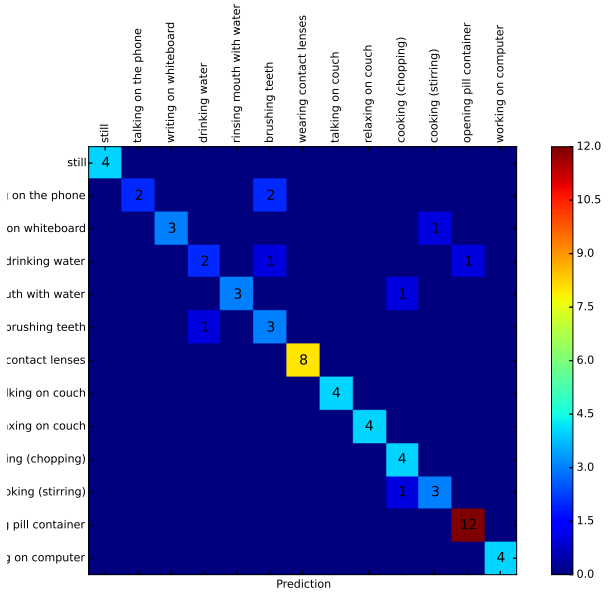


Figure 3.4: Confusion matrix: Longest common subsequence approach with 64 clusters per class. precision 90%, recall 88%

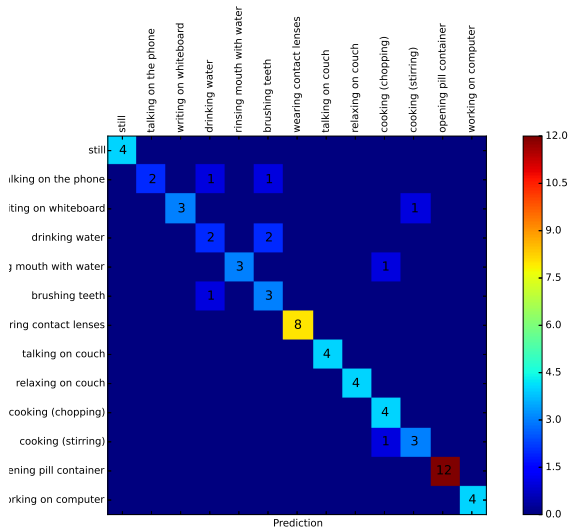


Figure 3.5: Confusion matrix: Dynamic Time Warping approach with 64 clusters per class. precision 90%, recall 88%

4 Approach: Discriminative Sequence Back-Constrained GP-LVM

As discussed earlier the simple *bag-of-features* approach has its limitations as it is not capable of identifying activity transitions. To deal with this problem we choose to implement another algorithm, capable of classifying a sequence in real time and inherently taking the alignment of the sequences into account.

4.1 Discriminative Sequence Back-Constrained GP-LVM

In the paper "Discriminative Sequence Back-Constrained GP-LVM for MOCAP Based Action Recognition"[1] the authors propose a method for classifying MOCAP actions.

cite mocap

The MOCAP database consists of a large number of different activities performed by human actors and recorded using motion capture devices. The information recorded is comparable to the skeleton representation but contains more data and is virtually noise free.

The method proposed in [1] is illustrated in Figure 4.1

The idea is to perform a dimensionality reduction on the skeleton data, resulting in a much compact representation. By introducing a *DTW* based sequence alignment kernel similarity measures can be defined for the activities. By using this similarity measure for the sequences in the observed space and constraining the optimization to preserve this measure the local distances between the sequences are transferred into the latent space. The latent points of similar sequences are thus located nearby and the centroids of similar activities are distributed more close to each other. Then instead of using back-constraints to map a single pose sample into the latent space, we can compute the centroid in the latent space directly from a sequence of poses.

The sequence back-constraints have two advantages:

First all the sequences have a meaningful clustering in the latent space and thus the mean (centroid) of each sequence is a good representation.

Second by also learning the back-constraint it is possible to calculate the centroid of a sequence in the latent space directly without maximizing a likelihood. This in turn is being used to infer the centroid for an activity in the real-time classification for actions.

The authors validated this approach on the MOCAP dataset using 7 different actions (Run, Walk, Jump, Throw-Toss, Sit-Stand, Box, Dance) and achieved an average recognition rate of 72.9%.

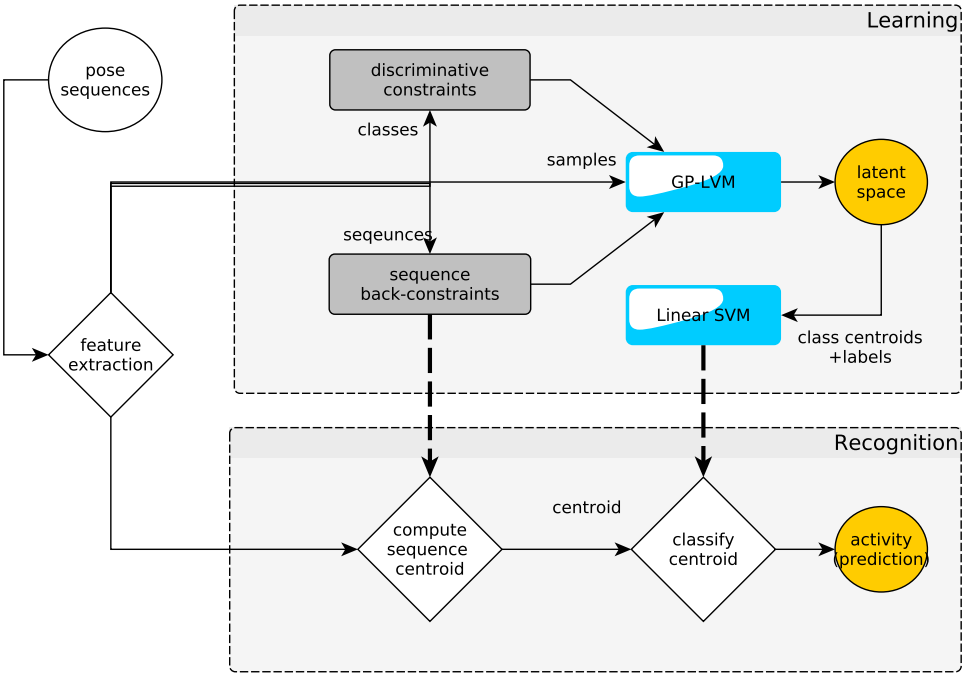


Figure 4.1: Illustration of the "Discriminative sequence back-constrained GP-LVM" approach. Learning is done by training a GP-LVM together with the sequence-back constraints and discriminative constraints. After that an Linear SVM is trained by the class centroids and the corresponding labels. Recognition is performed by computing the centroid of a new sequence through the learned back-constraints and predicting with the SVM.

The issue of the high-dimensional input data is solved by the dimensionality reduction with the *GP-LVM*. The issue of classifying sequences is resolved with the sequence alignment kernel, which is used to change the latent space, such that it also represents sequence distances.

4.1.1 Sequence back-constraints

The mapping is defined as a linear combination of the *DTW* distance between every other sequence. For every latent dimension q we have:

$$g_q(Y_s) = \sum_{m=1}^{S} a_{mq} k(Y_s, Y_m)$$

where the similarity measure is $k(Y_s, Y_m) = \gamma e^{\mathrm{DTW}(Y_s, Y_m)}$. This measure can be in-

terpreted as a sequence alignment kernel. The measure is to be preserved in the latent spaces.

$$g_q(Y_s) = \mu_{sq} = \frac{1}{L_s} \sum_{n \in J_s} x_{nq}$$

Therefore we need to perform a constrained optimization for the *GP-LVM*.

4.1.2 Discriminative GP-LVM

Furthermore, by applying the Discriminative GP-LVM we ensure that poses of different activities are separated from each other and poses from similar activities are located closer together. This ensures that the centroid of an activity is more informative and thus discriminative. The Discriminative GP-LVM works by minimizing the between class similarity and maximizing the innner-class variance [16].

The two criteria for optimization are:

- The distance between the classes

$$S_b = \sum_{i=1}^{l} \frac{n_i}{n} (\boldsymbol{\mu_i} - \boldsymbol{\mu})(\boldsymbol{\mu_i} - \boldsymbol{\mu})^T$$

where n is the number of samples, n_i is the number of samples for class i and l is the number of classes. Furthermore $\boldsymbol{\mu_i}$ is the mean of the class and $\boldsymbol{\mu}$ is the mean across all classes.

- The variance within each class

$$S_w = \frac{1}{n} \sum_{i=1}^{l} \sum_{j=1}^{n_i} \frac{n_i}{n} (\boldsymbol{x_{i,j}} - \boldsymbol{\mu_i})(\boldsymbol{x_{i,j}} - \boldsymbol{\mu_i})^T$$

where $\boldsymbol{x_{i,j}}$ is the j-th sample from class i.

The variance within each class $|S_w|$ should be minimized and the distances between the classes $|S_b|$ should be maximized. This is done by combining both condition into a sole criterion and maximizing it:

$$J(X) = tr(S_w^{-1} S_b)$$

This criterion is added to the likelihood of the GP-LVM with a parameter λ which decides how much weight the discrimination should take in the optimization. If we make the model more discriminative we could break the learning of the non-linear manifold. On the other hand if the value is small the contribution will bee to minor and we will not gain any discrimination between the classes. Recognition is being done by applying the mapping above to the new sequence and using a SVM in the latent space.

4.1.3 Advantages

This methods maps each pose from every activity inside the same latent space, which ensures that the mapping captures a non-linear manifold which is categorized by all activities. Recognition can be done in real time by using the learned back constrained. The centroid in the latent space is being calculated for the whole sequence and classified by the SVM. Also incomplete trajectories can be classified. When there is an activity transition the centroid will cross the decision boundary of the *SVM* and be naturally classified to the new corresponding activity.

4.1.4 Shortcomings

As all activities are modeled inside one latent space it is very difficult to find a non-linear mapping from latent to observed space. The standard approach for optimization in the *GP-LVM* is using the *Scaled Conjugate Gradient* method. As the optimization for *GP-LVM* is determined by the above similarity measure and the discriminative criterion finding a good minimum is very difficult. It is thus highly likely that performing a gradient optimization will be stuck in an local minimum. The authors in [1] argue that initializing with a more sophisticated dimensionality reduction technique is a necessity. In their work they use the *ISOMAP* and the *Locally Linear Embedding* methods.

Also one problem with the real-time recognition is that determining when exactly an activity has ended/begun is very difficult. Also as we do not know how long a sequence is we have to calculate the centroid for several time frames using a sliding window approach.

4.1.5 Extensions:

1. Learn pose together with local motion to capture dynamics The GP-LVM learns a mapping for each pose but does not consider velocities and accelerations. If we take a pose along with its first and second moments as the high-dimensional space we allow for the temporal displacements to be also modeled. The latent space will represents the pose along with the local motions and the DTW kernel in the constraint will also captures the dynamics of the activity. Due to the difficult optimization and the high complexity of the data set we could not find a good local minimum with this approach.

2. Use mahalanobis for the DTW As described in section `Dynamic Time Warping with Mahalanobis Distance` we wanted to use a modified version of the *DTW* for learning the sequence back-constraints. But due to our tests the mahalanobis inspired *DTW* did not perform any better for our chosen features.

4.2 Feature extraction

Regardless of the chosen algorithm the features used for learning will have a big impact on the performance of the model. Therefore it is imperative to extract discriminative features from the skeleton data.

We get the joint positions and the angles between them in the camera frame defined by the used depth camera (.e.g Microsoft's Kinect).When extracting features we have to make sure that we have view invariant features of the skeleton. We want these data in the frame of the skeleton.

One way to achieve scale invariance is to normalize all link lengths in respect to the torso link. This correct for variances of skeleton lengths in different persons. To make the pose view invariant we have to define a local skeleton frame which captures the skeletons *orientation* in the world coordinate system.

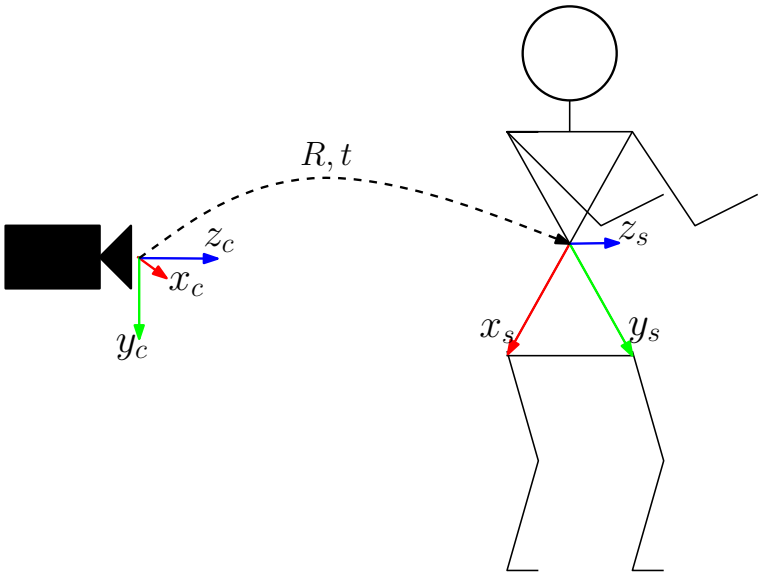


Figure 4.2: Sketch of the local skeleton frame inside the camera frame. The rotation matrix $\boldsymbol{R}$ and the translation vector $\boldsymbol{t}$ define the needed transformation to change from camera coordinates to the local skeleton coordinates

Another way to achieve view invariance is to not consider the 3D points of the joints all together but instead to take only relative features. These can be, for example the angles or distances between two adjacent joints. An interesting approach is used in [15], which is to define a polar coordinate frame for each joint and use only two angles, which define the orientation of the joint in a polar coordinate frame, as features. This way we also reduce the observation space.

As discussed in `Related Work` many methods also make the extracted temporal

features (e.g. Eigenjoints). However since we want to include the dynamics in our model we do not extract such features explicitly.

We selected a 3D point cloud of the joints in the skeletons own coordinate frame as features. The reason for this is that we believe the 3D point cloud to be more linear than relative features, which in turn will help when optimizing the model. Figure 4.2 shows this approach. We chose the two vectors – torso to right hip and torso to left hip – to define our local coordinate system. By normalizing and computing the cross product we have also the third vector which points to the walking direction of the skeleton.

4.3 Implementation

cite GPy

As there was no publicly available source code and we wanted to integrate the code with ROS we choose to implement this method in Python. We used the *GPy* library from the ... Sheffield University .

To implement the Discriminative GP-LVM constraints we ported the code from Prof. Urtasun's matlab code to Python and integrating it with *GPy*.

Doing so we encountered several problems with the current numpy and scipy libraries dealing with sparse matrices. As of now there is no way to perform a fast multiplication of sparse matrices and of block-diagonal matrices. The only solution to this is by manually implementing an algorithm. But doing so one looses all the advantages of the BLAS and LAPACK integration of numpy.

To enforce the sequence back-constraints we implemented a constrained optimization by adding Lagrangians to the objective function. This way the weight parameters for each sequence alignment kernel were learned.

4.4 Dynamic time warping with mahalanobis distance

The Dynamic Time Warping algorithm is a prominent and very effective choice for computing the similarity between two sequences. Using this measure as a sequence alignment kernel the methods aligns similar sequences closer to each other. The effectiveness of the recognition is determined by the accuracy of this alignment kernel.

The issue with this approach, in the context of activity recognition, is how to define the distance metric between two poses. This metric is crucial for the *DTW* to find an optimal path. Popular choices for the distance function is the euclidean distance, if 3D points are used as features and which we used in our implementation, and the angular distance for angles. The problem with these two distances is that they are just the sum of the individual feature differences. As the dimensionality grows this metric becomes less informative.

In the case of human poses we have a certain notion of which poses are similar and which are far apart. Maybe this is due to the fact that we inherently know – or classify – to which activity the pose corresponds to and have therefore some notion of closeness

with respect to an activity which cannot be approximated with the euclidean distance. Poses from different activities will most likely also seem to be more or less similar depending on how similar the actions are.

One idea to transfer this knowledge is by using the Mahalanobis distance instead of the euclidean distance when computing the similarity of two pose sequences. By computing the covariance for each activity we have some notion of the variance across all feature dimensions for a specific class. This way we can capture – to some extent – the variability for each class. Now we can compute a similarity measure with a new sequence x_new for each class and each sample of this class. Thus we can define a notion of measure between a class and a new sample by:

$$s(j, \boldsymbol{x_{new}}) = \frac{1}{|C_j|} \sum_{\boldsymbol{x} \in C_j} \frac{\text{DTW}_{\text{mahalanobis}(\boldsymbol{\Sigma_j^{-1}})}(\boldsymbol{x}, \boldsymbol{x_{new}})}{min(|\boldsymbol{x_i}|, |\boldsymbol{x_{new}}|)}$$

where C_j is the set containing all class sequences and $|C_j|$ is the number of sequences in class j. The normalization factor $min(|\boldsymbol{x_i}|, |\boldsymbol{x_new}|)$ makes sure that the minimum cost computed by the DTW is proportional to the smallest sequence.

This way the distance error is distributed by a way defined by the variance across each dimension.

A similar idea was also proposed in the context of handwritten signature verification in [10], which uses just one covariance matrix. The covariance matrix is determined such that, just like in the case of Discriminant GP-LVM, it maximizes the variability between classes and minimizes the difference for samples in the same class. In contrast to our approach the overall covariance matrix may define a more meaningful and discriminative measure but it is also more difficult to update when performing online learning and when learning a new class (novelty detection).

4.4.1 Implementation and evaluation

We wrote a simple version of the Dynamic Time Warping in Python using dynamic programming and following the recursive definition in chapter 2.5.1. As the variance for some feature dimensions can be zero the constructed covariance matrix does not have full rank and thus cannot be inverted. We mitigate this problem with an approximation of the inverse by computing the pseudoinverse.

4.5 Evaluation

Our tests on the Cornell Daily Living Activity data were unsuccessful as the optimization failed to find a discriminative latent space. We believe that the many constraints on the optimization and the highly variant data is very hard to optimize.

Another reason for this could be that the activities in the Cornell data set are more complex. The MOCAP data represents action which could be described more easily

with a non-linear manifold. In contrast most daily living activities consist of several and, also in their inherent structure, different actions. The *DTW* measure is therefore not suitable to capture the similarity between two complex activities.

It can be argued that the performance of the *GP-LVM* is strongly dependent on the initialization. This was stated also in the original [**?**]. It seems that the objective function is highly-nonlinear and it is very likely that the optimization will find a local minimum in the vicinity of the starting position.

For these reasons we choose to implement a new model based on motion flow fields which will be learned for each activity separately.

5 Approach: GP-Latent Motion Flow

It can be argued that the mean poses computed in the *bag-of-features* method capture the most probable pose and motion tendencies of an activity. The good performance of the algorithm can be attributed to this fact. This can be explained by the local motion descriptor and the structural descriptor which are good representations for the current pose. We want to come up with an algorithm which performs dimensionality reduction, like in the case of the *Discriminative sequence back-constrained GP-LVM* but also captures the these motion tendencies for each activity class.

Many models use *GP-LVM* to reduce the high dimensional space into fewer dimension. These approaches make the problem more feasible but the issue remains how to do classification for time-series data. Human motions are mostly characterized by the dynamics of the model (temporal dimension). So we have to compare trajectories in the latent space. One idea is to learn a *Gaussian Process Dynamical Model* for each activity. This way we will have a function of the trajectory. This method is very powerful but using it will very likely not give good results in our case, due to the fact that more complex activities do not necessarily resemble the same trajectory. If we take the activity "cooking" as an example, there is no main trajectory that is being followed. Moreover this activity is defined by its local motion tendencies and sub-actions. Also a trajectory based approach will not be able to model cyclic action inside an activity which can be repeated an undetermined number of times.

One idea to solve these problems is to learn a motion flow field inside the latent space. This can be done with a similar method to the *Gaussian Process Regression Flow (GPRF)* proposed in [5]. The classification can be done using first and second order dynamics which should give better results. Going further the activity itself is characterized by the first and second moments of the trajectory function. By explicitly modeling the velocity of the trajectory we can take changes in the joint movement into account.

5.1 Gaussian Process Regression Flow

In [**?**] the authors propose a new method to model trajectories called *Gaussian Process Regression Flow*. The idea is to model a trajectory using a dense flow field inside the spatio-temporal domain.

They also show a practical application of this method for real-time classification of trajectories of vehicles on street crossing.

5.2 GP-Latent Motion Flow

The GP-LMF method is inspired by this model. With the difference that we do not use the spatio-temporal domain but only the spatial domain for the latent space. The reason being that we do not have starting and ending positions for each activity and also the lengths can be variable. For this reason it is very difficult to normalize with respect to the time dimension. On top of that we also want to recognize an activity which is being interrupted by another activity, so we cannot fix the lengths of the trajectories. Nevertheless, resulting from the properties of Gaussian Process regression, we have also a dense mean flow field and dense variances. This will allows us perform efficient and robust online recognition in the latent space.

The method is illustrated in Figure 5.1. The idea is to learn a *GP-LVM* mapping together with motion flow field in the latent space for each activity. First we extract our features as described in `Feature Extraction`. Then we perform a dimensionality reduction for each class and learn the backward mapping. Having the sequence in the latent space we compute the numerical derivative for each point and learn it through a *Gaussian Process*. This *GP* represents our flow field. In practice we learn a separate *GP* for each latent dimension. Each activity has its own flow field. Recognition and prediction is done by calculating the energy of the currently moving point, i.e. the incoming pose mapped to the latent space, with each different field. The field with the minimum energy represents the most probable activity as the point follows more closely its "current" of motion.

This model is attractive for two reasons. First real-time classification of incomplete trajectories is possible. Incomplete not only in the sense of the first part of an activity but any interval of an activity, which could be also somewhere in the middle of the sequence. Second it is possible to do online learning by simply adding the new class as a new flow field to the pool of GP regressions. It is very difficult to adjust other models, which rely on the mapping between latent space to observation space, for online learning, because of the problem that we can get stuck in a local minimum when optimizing the parameters of the *GP-LVM*.

Variances in the speed of performing an activity can be modeled by giving the point in the latent space a mass which can be adjusted in real time. When a point has greater mass then it needs more energy to be propagated through the flow field (the overall activity is slower) and vice versa.

As we use the *GP-LVM* and a *GP regression* way we have two indicators for recognizing unobserved data. The first one is the variance of the back-constraint mapping. If it is high we know that current sample is far away from the observed ones. The second is the variance of the *Gaussian Process Regression*. If this value is high we know that we didn't see any sample in the latent space with the current motion. Therefore, with this two indicators, we have a notion of how new a sample and its current motion are.

Another advantage of this method is that activities with repetitive motions, such as

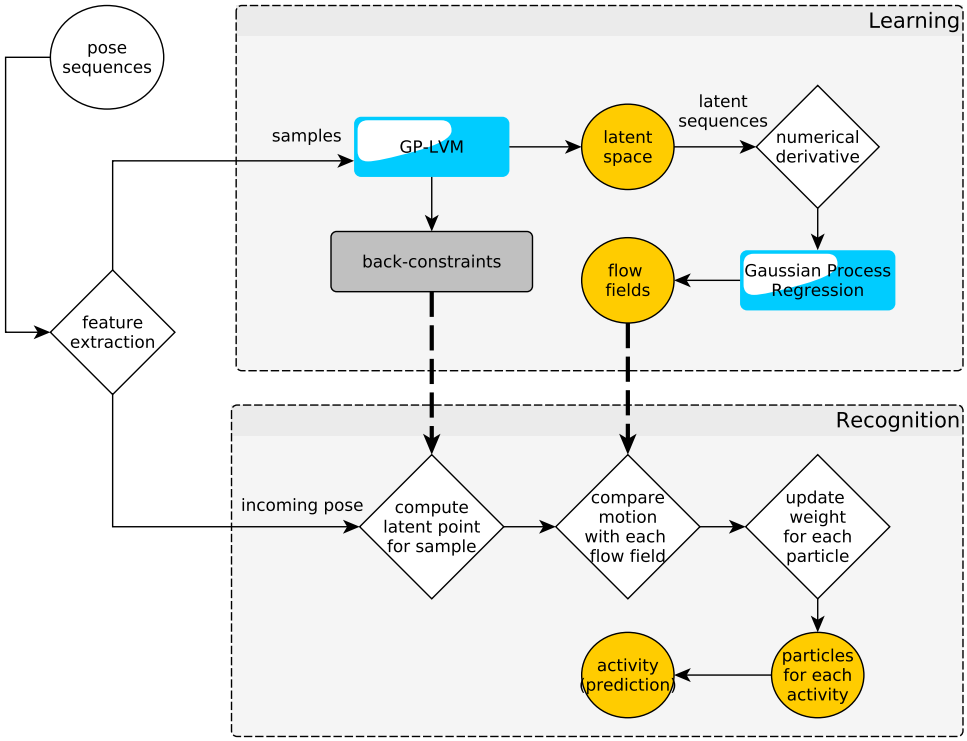


Figure 5.1: Illustration of the Gaussian Process - Latent Motion Flow approach.

walking or running, can be learned without using periodic kernels or without resorting to model them explicitly. Repetitive motions can be seen as just multiple samples of the same motion which define the flow field.

As in the previous approach the *GP-LVM* handles the high-dimensional data. The regression inside the latent space is then used to classify sequences in real time.

5.3 Recognition

The first approach for computing the energy for a new sequence and each flow field was to compute the dot product of the actual gradient with the vector from the flow field at the latent point mapped from the back-constrained mapping. We wanted then to perform online classification by comparing which flow field results in the minimum energy. This proved to be a bad measure as unobserved poses will map to a small area inside the latent space and be located very close to each other. This can be explained by the fact the they do not lie on the learned manifold and therefore will have small variation inside the latent space. This means also that if we compute the energy for these points it will be very small.

For this reason we changed our measure and choose to perform recognition by computing the energy that is accumulated along the current. Inspired by the particle filter method our recognition approach was to have a particle for the latent space of each activity. In every time step the particle is being updated with the above described probability. Then all particles are normalized. This way we ensure that the particle represents the probability that the current action is being performed. If it respects the flow field it will accumulate more weight and due to the normalization the other particles will become smaller. Thus the weight of the particle represents the probability of the activity for each time step.

5.4 Requirement: Smooth latent space

It is imperative for the *Gaussian Process* regression that the latent space is smooth. If this is not the case than different poses which are located nearby can have completely different gradients. This in turn will break the similarity property of the *RBF* kernel and the flow field would have no As the *GP-LVM* preserves distances rather than locality, it is very likely that there will be no smooth mapping in the latent space. One approach to constrain the latent space to be smooth is using the back-constraints.

5.5 Learning the flow field

The initial idea was to learn a general dimensionality reduction for a high number of varying activities and work with only one latent space. But as we saw in Chapter 4 such an approach is very difficult to realize, because of the difficult optimization task. Another problem is that the it is very difficult to learn a smooth mapping in the latent space. This is described more deeply in [18] where the authors try to incorporate the optimization criterion of *Locally Linear Embedding* together with the a back-constrained *Gaussian Process Dynamical Model*. As this approach needs also prior knowledge and is very complex we decided to learn each activity separately. Future work should deal with the possibilities of learning a unified latent space at it will allow us to learn different flow fields in the same space and we will not have to perform a heuristic normalization of the different flow fields.

Figure 5.2 shows the latent space representation of the walking activity of subject 35 in the MOCAP database. The dimensionality reduction was performed using a *RBF* backward mapping with lengthscale 1.0. The computed velocities are shown in Figure 5.3. From this velocities we learn a flow field (Figure 5.4). We use a very small variance for the *RBF* kernel in the *GP* regression of the velocities to bring the strength of the flow field near zero outside of the observed samples. This way when we perform activity recognition samples which deviate from the learned flow will result in no velocities.

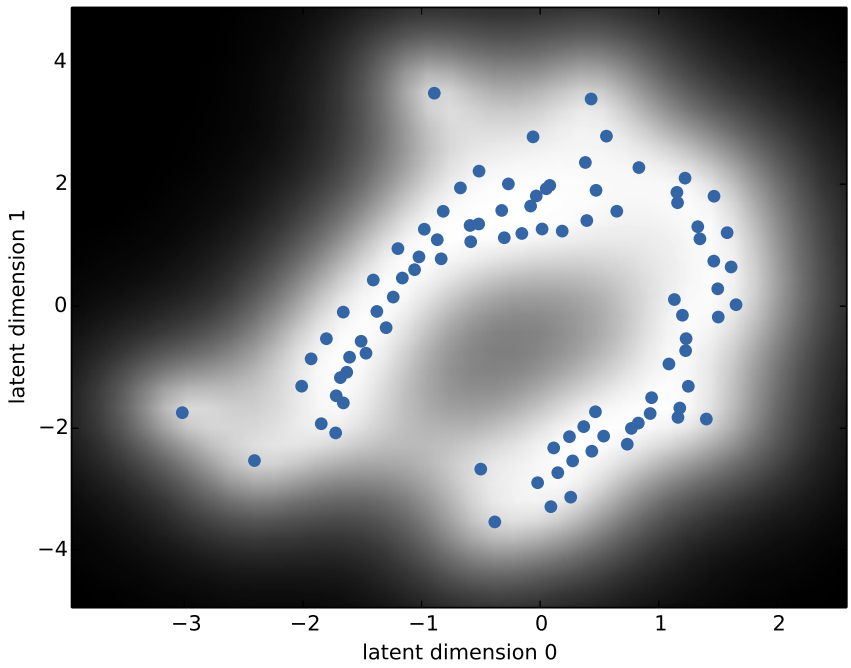


Figure 5.2: Two dimensional latent space representation of the "walking" sequence using GPy plot. The white area around the sample points represents the variance. MOCAP, subject 35, sequence 1.

One problem we encounter by learning the motion flow field from several samples is complexity of the *Gaussian Process*. There are two solutions for this. The first one is to use a sparse GP model. The second one is to sample points from all samples and use only those that are most suitable for the regression. If we take *IVM* as the sparse GP model both approaches can be seen as equivalent as the *IVM* will automatically take the most informative samples.

5.5.1 Effects of the hyperparameters

Changing the *lengthscale* defines how much each point is contributing to the regression process. It can be interpreted as a smoothness factor which governs how strong the interpolation of the flow field is performed on the latent points.

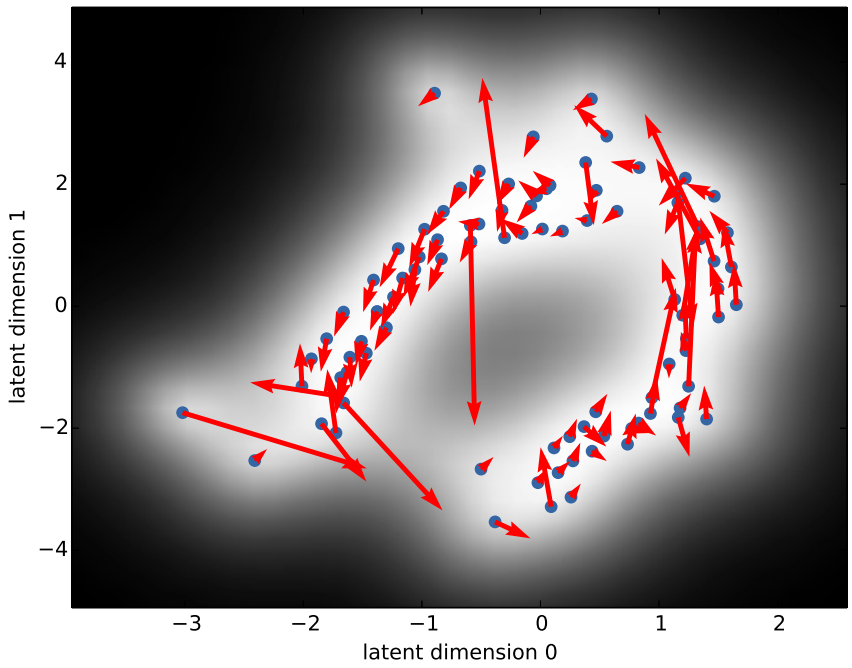


Figure 5.3: The calculated velocity (red) for each latent point. MOCAP, subject 35, sequence 1.

5.6 Interpretation

The proposed model has a natural interpretation. A point represents a pose in latent space and an activity is a trajectory in time inside the same space. With the flow field we learn the motion tendencies for each pose. When performing recognition we let the current point traverse each separate field and compute the accumulated energy. If we consider that the point has a mass we can model the speed at which activities are being performed. This way we can recognize when a point leaves an activity, which represents a *motion current*, and passes over to some other activity.

5.7 Advantages

5.7.1 Recognition

The current activity is being mapped into the latent space, through the learned back-constraints. The recognition is being performed solely in the latent space. By propagating the current position by each flow field we can calculate the next possible pose.

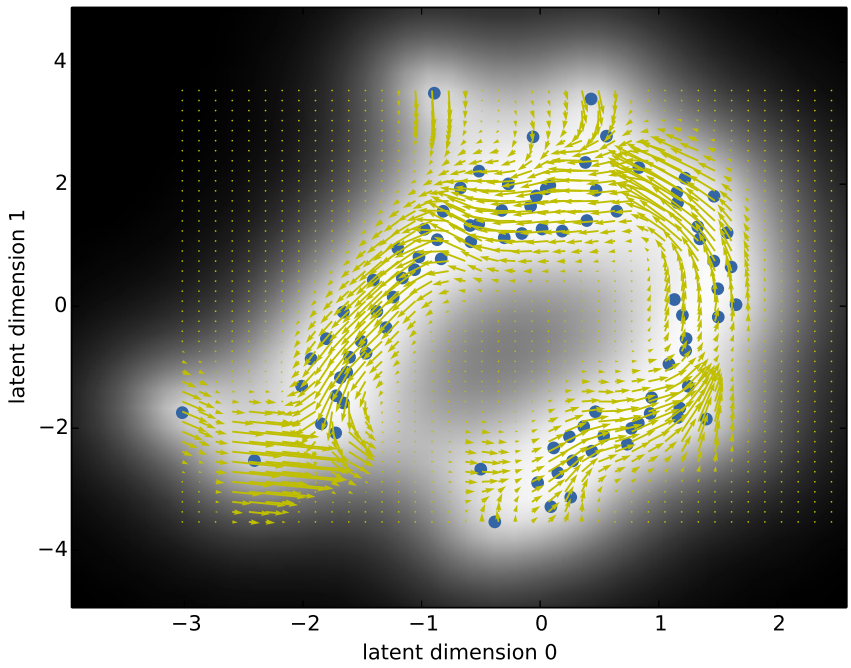


Figure 5.4: The learned flow field (yellow) from the velocities. lenthscale = 1, MOCAP, subject 35, sequence 1.

By comparing the similarity considering the variances we have a measure of how well the current activity resembles each flow field e.g. learned activity. By doing this for each separate activity class we can maintain a probability how likely it is that the current motion tendencies resemble each learned class. As this probability is update in each time step, online recognition is possible. Moreover when the activity changes the corresponding flow field will accumulate more weight.

5.7.2 Prediction

If we have detected the activity predicting is simply a matter of propagating the pose through the flow field by taking the mean of the GP. This way short-term predictions are possible. Also the prediction is updated every time the point changes its position in the latent space.

5.7.3 Multiple Hypothesis Prediction

Since we have a GP representing our flow field we can predict future point positions with the mean value. Moreover also having informative variances we can sample several possible trajectories. This can be accomplished using an particle filter. Hence we can have multi-hypothesis predictions along with their probabilities.

5.7.4 Online learning

Learning new samples can be accomplished by updating the *GP regression* with the new velocities.

5.7.5 Active learning

5.7.6 Novelty detection (anomaly detection)

In [**?**] the authors present the ability of the GPRF model for anomaly detection. This approach is also suitable for finding new classes as the above energy value can be used to recognize novel activities. The reasoning is that if we cannot find an flow field with a small energy the activity has to be unobserved. Also with the two uncertainty values for the poses and the motion finding new activities makes the task easier.

5.7.7 Modeling of cyclic activities

As this method is not trajectory based, but rather flow field based, recognition of cyclic activities is possible.

5.8 Implementation

We used non-sparse *GP-LVM* with a back-constrained mapping to learn each class of activities. The back-constrained has two purposes. The first is that we need a backward mapping to map new poses to the latent space. The second is that the latent space must be smooth.

5.9 Evaluation

Performing a non-linear dimensionality reduction is no easy task. Testing was done with only two dimensions as it easier to visualize the latent space and the resulting flow fields. A latent space with higher dimension will naturally make the reduction more robust and the field will have a more natural interpretation but due to the fact that we model each latent dimension separately the flow field looses its discriminative effect in higher dimensions.

At first we concentrated our efforts for learning with the MOCAP data. In theory the data collected from the kinect should be equivalent. One difference is the high noise in the pose estimation, but due to the fact that the GP-LVM preserves distances rather than locality this problem is mitigated to a certain degree. For most activities, such as walking, running and jumping it was possible to learn a representative flow field. For more complex actions the optimization could not find a smooth mapping. This could also be due to the used *RBF* kernel for the backward mapping.

Unfortunately we were not able to perform an appropriate dimensionality reduction. This has several reasons:

- Due to the fact that we want to also learn a backward mapping (observed space to latent space) we have to initialize the latent space with this mapping. For this reason we could not define a more appropriate initialization such as *PCA* or *LLE*.
- The Cornell dataset contains many activities with many samples. Using the *IVM* variant helps in some respect to circumvent this problem but also introduces an approximation.
- The back-constraints were not enough to constrain the latent space to be smooth.

[18] [3] [17]

The author in Exploring model selection techniques for nonlinear dimensionality reduction also suggest to use ISOMAP or LLE to initialize the GPLVM and argues that direct optimization of the GP-LVM is very difficult.

Probabilistic Feature Extraction from Multivariate Time Series using Spatio-Temporal Constraints

6 Conclusions and Outlook

6.1 Summary

In this thesis we covered the issue of performing online activity recognition from skeletal features. We began by re-implementing an existing method, which works by extracting representative poses and performs a histogram pooling over the quantized pose sequence.

6.2 Lessons learned

- Dimensionality reduction for all activities is very difficult (also with extra constraints)
- Dynamics is a good measure for classification of human activities

6.3 Contributions

- Advantages and Disadvantages of dimensionality reduction with GP-LVM for human motion in the context of activity recognition
- Implementation of the Discriminative GP-LVM with python

We ported the matlab code provided by Prof. Urtasun into python and integrated it with the GPy library

- Implementation of the Sequence Back-constraints

We used Lagrangians to implement a constrained optimization of the likelihood function

- A novel approach for activity recognition

With the *GP-LMF* we presented a novel method for online recognition of complex activities.

6.4 Outlook

6.4.1 Implementation of the GP-LMF using spatio-temporal GP-LVM

cite spatio temporal GP-LVM

As described earlier the GP-LMF approach failed, due to the fact that the optimization of the GP-LVM with back-constrained did not result in a smooth latent space. One possibility to solve this problem is by implementing the *Spatio-Temporal GP-LVM* as described in .

6.4.2 Adaptive GP Regression of the flow field

One problem of the proposed *GP-LMF* method, besides the difficulty to obtain a smooth mapping, is that the *RBF* kernel is stationary. This means that the lengthscale is defined over the whole latent space. We would like this lengthscale to adapt to the curvature of the flow field. This way small and large motion tendencies can be learned.

6.4.3 Semi-supervised activity learning by automatic segmentation of activities

If the problem of a smooth latent space can be solved the *GP-LMF* method can be used to perform an automatic segmentation of observed and unobserved activities. As discussed earlier, since we have two good indicators of the uncertainty of both, the poses we see and the motion tendencies, we can segment a time interval into ”known” and ”unknown” activities. In conjunction with online learning, this will greatly reduce the time this method needs to learn appropriate flow fields for a number of classes.

List of Figures

Bibliography

[1] Discriminative sequence back-constrained GP-LVM for MOCAP based action recognition:. pages 87–96. SciTePress - Science and and Technology Publications, 2013.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, August 2006.

[3] Sebastian Bitzer and Christopher Williams. Kick-starting GPLVM optimization via a connection to metric MDS. 2011.

[4] Yun Jiang and Ashutosh Saxena. Modeling high-dimensional humans for activity anticipation using gaussian process latent crfs. In *Robotics: Science and Systems, RSS*, 2014.

[5] Kihwan Kim, Dongryeol Lee, and Irfan Essa. Gaussian process regression flow for analysis of motion trajectories. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1164–1171. IEEE, 2011.

[6] Neil Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *The Journal of Machine Learning Research*, 6:1783–1816, 2005.

[7] Neil Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse gaussian process methods: The informative vector machine. *Advances in neural information processing systems*, pages 625–632, 2003.

[8] Neil D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Nips*, volume 2, page 5, 2003.

[9] Neil D. Lawrence and Joaquin Quiñonero-Candela. Local distance preservation in the GP-LVM through back constraints. In *Proceedings of the 23rd international conference on Machine learning*, pages 513–520. ACM, 2006.

[10] Yu Qiao, Xingxing Wang, and Chunjing Xu. Learning mahalanobis distance for DTW based online signature verification. In *Information and Automation (ICIA), 2011 IEEE International Conference on*, pages 333–338. IEEE, 2011.

[11] Sébastien Quirion, Chantale Duchesne, Denis Laurendeau, and Mario Marchand. Comparing GPLVM approaches for dimensionality reduction in character animation. 2008.

[12] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. University Press Group Limited, January 2006.

[13] Matthias Seeger and Michael I. Jordan. Sparse gaussian process classification with multiple classes. Technical report, Citeseer, 2004.

[14] Hiroshi Shimodaira, Ken-ichi Noma, Mitsuru Nakai, and Shigeki Sagayama. Dynamic time-alignment kernel in support vector machine. In *NIPS*, volume 14, pages 921–928, 2001.

[15] Ilias Theodorakopoulos, Dimitris Kastaniotis, George Economou, and Spiros Fotopoulos. Pose-based human action recognition via sparse representation in dissimilarity space. *Journal of Visual Communication and Image Representation*, 25(1):12–23, January 2014.

[16] Raquel Urtasun and Trevor Darrell. Discriminative gaussian process latent variable model for classification. In *Proceedings of the 24th international conference on Machine learning*, pages 927–934. ACM, 2007.

[17] Raquel Urtasun, David J. Fleet, and Pascal Fua. 3d people tracking with gaussian process dynamical models. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 238–245. IEEE, 2006.

[18] Raquel Urtasun, David J. Fleet, and Neil D. Lawrence. Modeling human locomotion with topologically constrained latent variable models. In *Human Motion–Understanding, Modeling, Capture and Animation*, pages 104–118. Springer, 2007.

[19] Chenyang Zhang and Yingli Tian. RGB-d camera-based daily living activity recognition. *Journal of Computer Vision and Image Processing*, 2(4):12, 2012.