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 15, 2014 |

# Abstract

With the advent of new RGBD sensors and their deployment in robotic systems, it is now possible to get a frame-to-frame extraction and tracking of human poses in the robot's environment. By reasoning on sequences of such poses one can learn and recognize different human activities.

The subject of this work is to devise a practical method capable of online (incremental) activity recognition. Three different approaches are taken. The first one is an implementation of an existing model which extracts representative poses from an activity and uses *Support Vector Machines* for real-time activity recognition by classifying the distribution of these poses. Several extensions are proposed, such as using sequence similarity measures for the classification. An integration into the *Robot Operating System* is developed. This results in a module capable of real-time classification but not capable of online activity recognition.

The second approach exploits the *Discriminative Sequence Back-Constraint GP-LVM*, which is a model that learns a non-linear lower-dimensional manifold for the poses, and optimizes this latent space in such a way, that the similarity of the sequences and the discriminative properties of the activity classes are captured. Online activity recognition is performed by classifying the centroid of the latent clusters belonging to the individual activity sequences. Tests on the Cornell Daily Living Activity data set reveal that this model is not capable of finding a latent space which captures all activities, because of the hard optimization problem.

The third approach is a novel method for online activity recognition based on separately learning a dense motion flow field for each activity class in its lower-dimensional representation through several *Gaussian Process Regressions*. Online recognition is done by incrementally computing and accumulating the probability that the actual pose movement corresponds to each learned flow field. The issue of finding a smooth inverse mapping from observed space to latent space is discussed and some alternatives for the dimensionality reduction are proposed.

# Zusammenfassung

Durch den Einzug von neuartigen RGBD-Sensoren und deren Einsatz in Robotersystemen, ist es heutzutage möglich Bild-zu-Bild Extraktion und Verfolgung von körperlichen Posen innerhalb der Umgebung von Roboter durchzuführen. Durch die Behandlung von Sequenzen solcher Posen kann man menschliche Aktivitäten lernen und erkennen.

Der Gegenstand dieser Arbeit ist es, eine praktische Methode für inkrementelle Aktivitätserkennung zu entwickeln. Drei verschieden Ansätze werden unternommen. Der erste ist eine Implementierung eines existierendes Modell welches repräsentative Posen von einer Aktivität extrahiert und die *Support Vector Machine* nutzt um echtzeit Aktivitätserkennung durch die Klassifizierung von der Distribution dieser Posen zu machen. Mehrere Erweiterungen werden vorgeschlagen, wie z.B. die Klassifizierung durch Ähnlichkeitsmasse für Sequenzen. Eine Integration in das *Robot Operating System* wird entwickelt. Das resultiert in einem Modul welches zwar fähig ist echtzeit Erkennung durchzuführen, aber keine inkrementelle.

Der zweite Ansatz ist es *Discriminative Sequence Back-Constraint GP-LVM* zu nutzen, welches ein Modell ist um ein nicht-lineare niedrig-dimensionale Mannigfaltigkeit für die Posen zu lernen, welche die Ähnlichkeit der Sequenzen und die separierbare Eigenschaften der Aktivitätsklassen erfasst. Inkrementelle Erkennung wird durchgeführt indem die Schwerpunkte der latenten Cluster, welche zu den einzelnen Aktivitätssequenzen gehören, klassifiziert werden. Experimente auf dem *Cornell Daily Living Activity* Datensatz zeigen, dass dieses Modell, wegen des schwierigen Optimierungsproblems, nicht fähig ist eine geeignete latente Repräsentation für alle Aktivitäten zu finden.

Der dritte Ansatz ist eine neue Methode, welche auf das Lernen eines dichten Flussfelds im latenten Raum jeder Klasse durch mehrere *Gauss-Prozesse* basiert. Die inkrementelle Erkennung wird durchgeführt durch berechnen und akkumulieren der Wahrscheinlichkeit, dass die tatsächlichen Bewegung der Pose zu einem Flussfeld korrespondiert. Das Problem eine glatte Abbildung in dem latenten Raum zu finden wird erläutert und es werden einige Alternativen vorgeschlagen um die Dimensionsreduzierung durchzuführen.

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

München, den 15 Oktober 2014 Evgeni Pavlidis

# Contents

# 1 Introduction

## 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 tasks. An example in robotics 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.

One way to do activity recognition is to infer the structure of the person (pose) in each frame and use this sequence of poses to perform classification. Earlier methods do this by extracting the pose from video data [30] and jointly inferring the activity. The problem is coupled because inferring the pose from 2D data is very difficult, and one needs also the information on the activity sequence to be able to reason about the pose.

With the advent and further development of RGBD (color and depth) sensors it is now possible to capture the depth information along with the image. Initial versions of such sensors capture the depth by projecting an infrared pattern onto the scene. By computing the warping of this pattern depth values for each pixel can be deduced. Figure 1.1 shows the ASUS Xtion PRO LIVE, which we used in this thesis. This device can output a dense depth image for each frame.

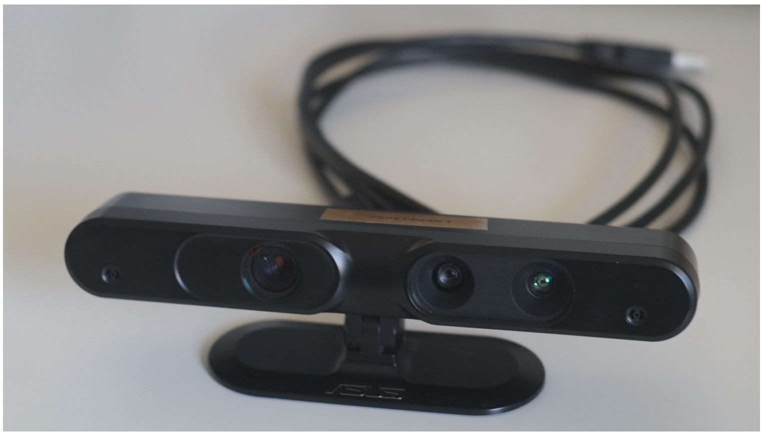

Figure 1.1: ASUS Xtion PRO LIVE

Along with robust machine learning techniques for the extraction of depth, new algorithms were devised which can detect and track the poses of persons inside the point cloud data. This pose data can be seen as the skeleton of the person. Moreover these framework work in real-time and are gradually being improved in their robustness. We used PrimeSense NITE framework which outputs the tracked skeleton of each recognized person for each frame.

This trend allows for the decoupling of the pose estimation and activity learning tasks, which make the second problem much easier. Also devising an algorithm that builds on top of robust pose estimation reduces the complexity, as we can fully ignore the pose estimation problem. Moreover skeleton data is very good as features for a pose-based algorithm. Some authors have already suggested that pose based features are much more informative for action recognition than image based ones [35]. There are already a multitude of methods working with such skeleton features that are produced by current RGBD sensors (e.g. [26], [36]).

### 1.1.1 The SPENCER project

The SPENCER Project [1] is a research project with focus on cognitive systems and social robotics. Its goal is to research different areas in the context of situation-aware robotics in populated environments, such as semantic mapping or perception, modeling and learning of social behavior. Another objective for the final robot, which will be deployed on the Amsterdam Schiphol Airport, is the capability of real-time reasoning and planning in a populated environment.

Figure 1.2 shows the final robot along with its individual sensors. Two Microsoft Kinect v2 sensors are installed which are responsible for the final pose estimation and skeleton tracking.

A final scenario, demonstrating the integration of the individual contributions, is to autonomously lead a group of passengers to a different gate in the Amsterdam Schiphol Airport.

Activity recognition can help in the perception and reasoning tasks. An example is the ability to recognize with which persons the robot can interact. Persons performing activities like *talking on the phone* or *eating* will be bad candidates, whereas persons which *read the airport map* will be good ones. An advantages of skeletal features are that no image data is being processed. Thus no personal information is saved 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.

## 1.2 Problem statement

As stated above we want to devise an algorithm which can classify different activities. We make a distinction between an action and an activity. The difference being that an

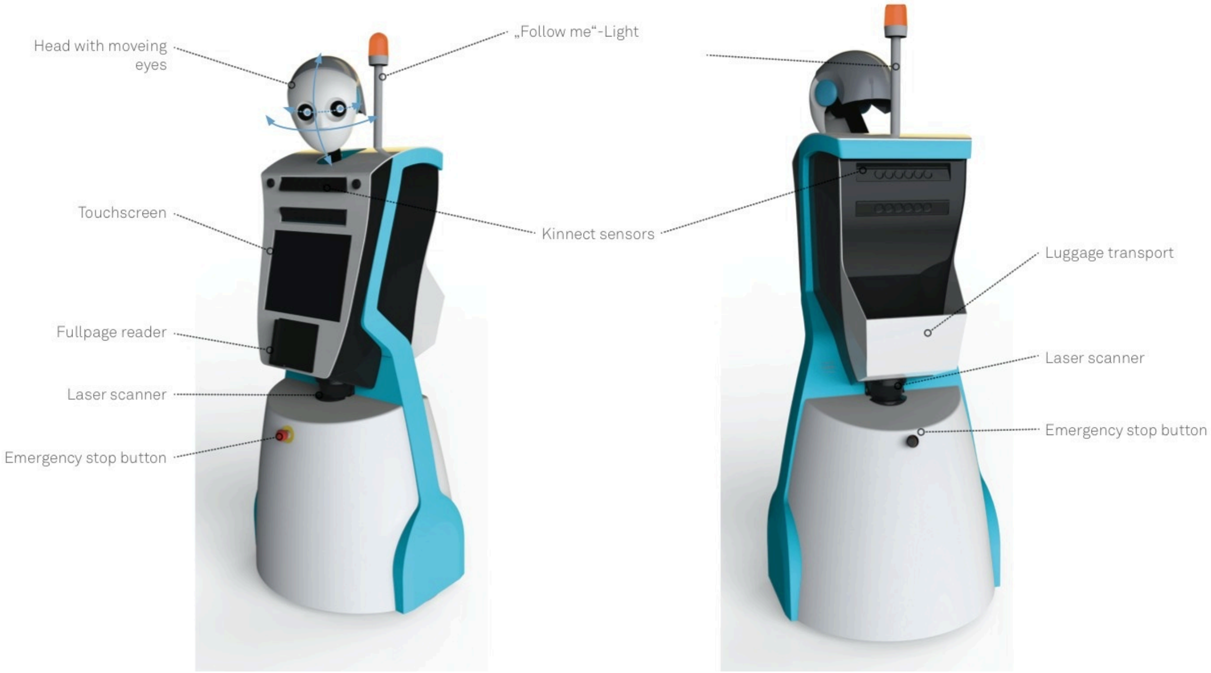

Figure 1.2: The final design of the SPENCER robot and its sensors.

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 actions several times (slicing vegetables) and which do not need to always be in the same order.

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 update its class prediction in real-time 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 an activity 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 an important prerequisite for many practical applications. In the SPENCER project for example being able to detect activities which highly deviate from a set of learned classes can help in some autonomous scenarios. A simple example could be to simply notify a supervisor in case of an anomaly. The supervisor can then go to the robot and find out what exactly is happening.

Optimally it should be possible to also recognize when an activity started and finished. This way automatic segmentation will be possible and will considerably reduce the efforts in supervised learning. 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.

Moreover we have to make sure that our algorithm can discriminate between different activities, but also will leave room for inner class variations. These variations are the result of various persons performing activities differently. A simple example is walking, where different persons have a different walking style (gait). Also varying environments will result in actions to be performed slightly differently [19].

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-dimensional.

**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. Furthermore online classification on streaming sequence data is very challenging. For this we have to devise an algorithm capable of discriminating between learned activities by only taking partial sequences into account.

The initial idea was to make use of kernel methods to model the underlying uncertainty in the activities. More specifically we wanted to compare the performance of *Gaussian Processes* – which are explained in section 2.3 – to the existing methods.

## 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 to understand the concepts described in this work.

### 1.3.1 Mathematical notation

- Matrices: uppercase : A, $\Sigma_{\mathrm{i,j}}$
- Vectors: lowercase bold : $\boldsymbol{x}, \boldsymbol{y}_{\mathrm{i}}$
- Constants: lowercase : l, n
- Parameters: lowercase Greek letters : $\gamma$, $\theta$

## 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. An emphasis is led on Gaussian Process Regression and Gaussian Process - Latent Variable Models, which are both kernel methods to perform regression and dimensionality reduction. Last the Dynamic Time Warping algorithm, which is used for sequence alignment, is explained.

**Chapters 3-5** The next three 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 [36] for classifying daily living activities. This method is then modified in way possible to capture the ordering of different sub-actions. These 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” [17]. 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 work and 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. These 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. Then a classification model can be used which takes these feature vectors and the corresponding labels as input and learns to categorize new and unobserved feature vectors by predicting a discrete value (category).

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. A prominent example is predicting the price of a real estate in a city. Given a large number of features and labeled data, for example the location, size and year of construction together with the price paid, one can predict the most likely price that a new property will cost by modeling the function between the relationship features - price. In contrast to classification, here we learn a function mapping from the features to a real value.

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) in such a way that it poorly predicts new samples, is called overfitting. This can happen if the model learns not only the data relationship 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 $\theta$ 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 manifold from which the samples are generated. The last one is called *Dimensionality Reduction* which will also be a subject in this work [3].

**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 [3].

### 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 a generative model is the *Hidden Markov Model* which models latent states of a sequence, where these states are defined by the probability of observing them and the transition probability between them. Thus the generative process is being modeled in a probabilistic way.

### 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.

### 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, in practice, is a convenient way to acquire new informative samples without letting someone go over a huge amount of data to label. In our 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 do not work 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 $\boldsymbol{y}$. It 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, by computing the kernel value, we have automatically computed the dot product in some feature space defined by this kernel. We do not have to explicitly map the input to this space. For this reason 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.

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

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. The 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 $\sigma_f^2$ represents how far away the points are located from the mean. The noise variance $\sigma_n^2$ models the strength of the noise in 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 (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.

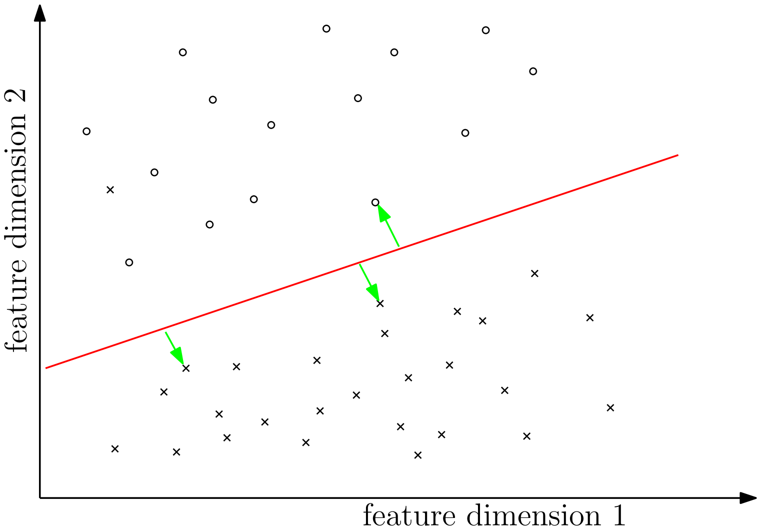

Figure 2.1: SVM decision boundary (red) between two classes (cross, circle). The support vectors are indicated in green.

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

Due to the fact that the formulation of the *SVM* works only with dot products of feature vectors, instead of finding a boundary in the original 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 original feature space, but instead could be linearly separated in some other space. If we take the *Radial Basis Function* 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 $\mu$) and it drops quickly on each side with the standard deviation $\sigma$. A plot of this distribution can be seen in Figure 2.2.

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

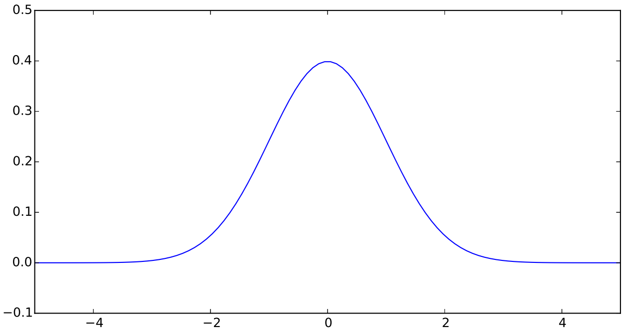

Figure 2.2: The univariate Gaussian distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$

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.

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

The two parameters of the distribution are:

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

**covariance** $\Sigma$: Representing the mutual variance for each pair of the elements of the random vector: $\Sigma_{i,j} = Cov[x_i, x_j]$

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

### 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 is the math behind them. One can use just linear algebra operations to perform many operations regarding multivariate Gaussian distributions [8].

**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 over 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 be regarded as a kernel as discussed in 2.2. Therefore a *Gaussian Process* can be interpreted as a Gaussian distribution over function space [23].

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

A Gaussian Process is 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 $f(\boldsymbol{x_{new,i}}) = y_{new,i}$. If we do not consider that the observation can be noisy, then we have a joint multivariate distribution of the form [23]:

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

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 $m \times m$ covariance matrix for the new points. The posterior can be computed by using the earlier formulas of the conditional for multivariate Gaussian distributions (see 2.3.3).

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

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 $\mathcal{O}(n) = n^3$, if we use standard methods. This is also the bottleneck of the whole algorithm and a serious drawback of Gaussian Processes. Although there are methods which reduce this limit, there is no general way to perform this operation in $\mathcal{O}(n^2)$.

### 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.

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

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

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 $\theta$. 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.

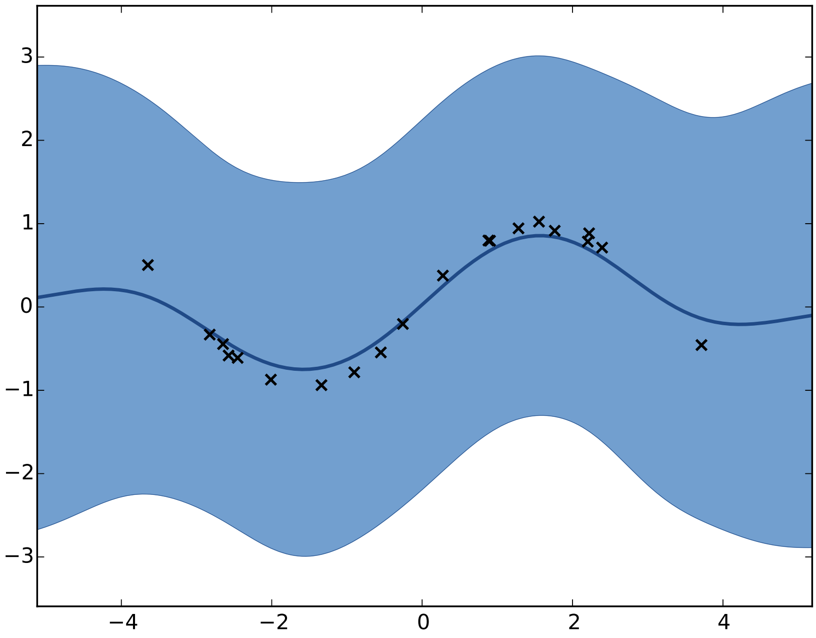

Figure 2.3: Gaussian Process Regression of the sinus function. Points were sampled with additional noise (5%). The regression was done with a RBF kernel function with length scale 1.0 and variance 1.0

Figure 2.3 shows generated samples form the sinus function with added 5% noise. We used the GPy [27] 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 lengthscale 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.

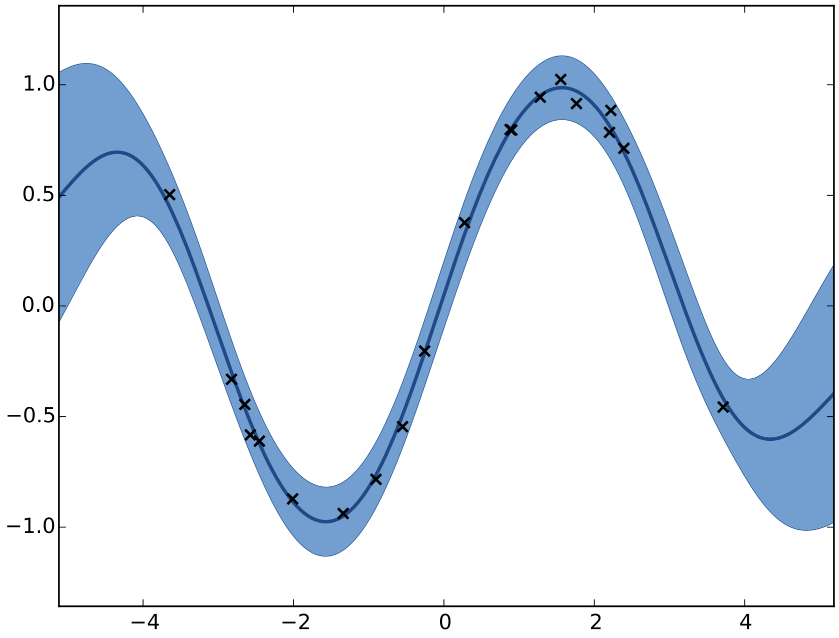

Figure 2.4: Gaussian Process Regression of the sinus function after optimization of the hyper parameters. Length scale: 0.1, Variance: 1.16

### 2.3.7 Advantages

**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.

**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.

**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.

**generative** As the *GP* can be seen as a distribution over function we can simply sample new functions from the process. Moreover different priors can be used if one has knowledge about the problem domain.

### 2.3.8 Disadvantages

**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 raises 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.

**uni-modal** Since the Gaussian distribution has only one global and local maximum it can model only one hypothesis. This is a curse but also a blessing since the math behind it is simple and unambiguous.

**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 $n$.

**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.

**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* [12] 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 still results in a good model.

This reduces the complexity to $\mathcal{O}(d^2 n)$ where $d$ is the number of the samples in the active set and $n$ is the number of overall samples. In case of classification, there is also an *IVM* method which works for multiple classes [24].

## 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 $n \times d$ matrix $Y$. 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

The *Probabilistic Principal Components Analysis (PPCA)* [29] is a linear latent variable model which combines the observed and latent space through a linear Gaussian relationship:

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

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 [13] 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 has a probabilistic interpretation [10]. 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

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

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

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 [11]:

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

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 – 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-constrained 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 variance in the data to the latent space. This problem is addressed by Lawrence et al. in the back-constrained GP-LVM [14]. A mapping 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 a number of mappings:

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

Where each mapping is a constraint for the $j$-th element of 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 [31]. 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

**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.

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

**probabilistic** Because of its probabilistic nature GP-LVM interpolation between two data sample is very natural [22]. Also it is more robust to sample noise.

**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

**no backward mapping** The idea of the GP-LVM is to learn a mapping from latent space to observation space by marginalization over the latent variables. Resulting from this is that we do not have an inverse mapping from observed space 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 `Back-constraints GP-LVM`. 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.

**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 hyperparameters, 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 dimensionality reduction is crucial for a number of models dealing with human motion (e.g. [5]). The GP-LVM preserve the distances in the mapping and are therefore suitable to model human motion with high noise in the poses.

## 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.

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

Where $\mathrm{dist}(x,y)$ 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 a real value. We consider the 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 [25].

### 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:

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

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 [19].

# 3 Approach: K-Means 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 [36].

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.

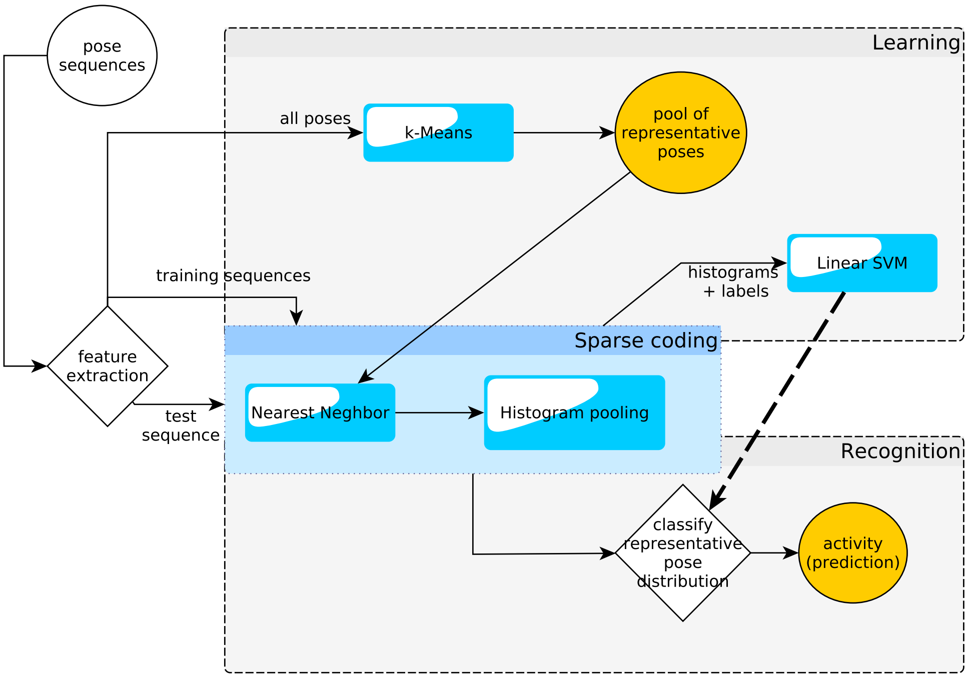

Figure 3.1: Illustration of the k-Means clustering and bag-of-features approach for activity recognition [36]. A code book is created from all poses. Recognition is performed on histograms over the sparse codes from the sequences.

## 3.1 Cornell Daily Living Activities dataset

We will use the "Cornell Activity Datasets (CAD-60)" described in [26] 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 similar. There are four persons each performing 13 activities. Examples of activities are 'talking on the phone', 'writing on whiteboard', 'drinking water', 'cooking (chopping)', 'working on computer'.

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
- Depth data

- Skeleton information: (joint position and orientation as extracted from the *NITE* module)

## 3.2 Robot Operating System

The *Robot Operating System (ROS)* [21] 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.

Each node can define a communication interface by defining message types. 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* or call *services*. The *topics* have a message type assigned to them and the *services* are defined by a request and response 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* structure and the current environment. This way real world data can be recorded and be played back later on. This is very convenient for debugging or system integration.

## 3.3 Implementation

For the implementation we used Python with the scipy and scikit-learn [18] 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 `openni_tracker` module [6]. This module reads the values of the PrimeSense NITE skeleton tracker driver and transforms the coordinates to a *ROS* specific depth camera frame. Then it publishes these transforms as a *TF* broadcast. 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 NITE module. 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 performs a classification on the sequence every two seconds. As the provided dataset is relatively large the learning time is several minutes. Parsing the data files and extracting the features takes the most part of the learning. 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 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 the euclidean distance in feature space as a measure between each mean pose. This will give a good approximation in the case that the clusters are located far away. As the *DTW* has a complexity of $\mathcal{O}(nm)$ we took every fifth element from the sequence for the calculation. Also by pre-computing the distance matrix the distance computation 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%. 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 the 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.

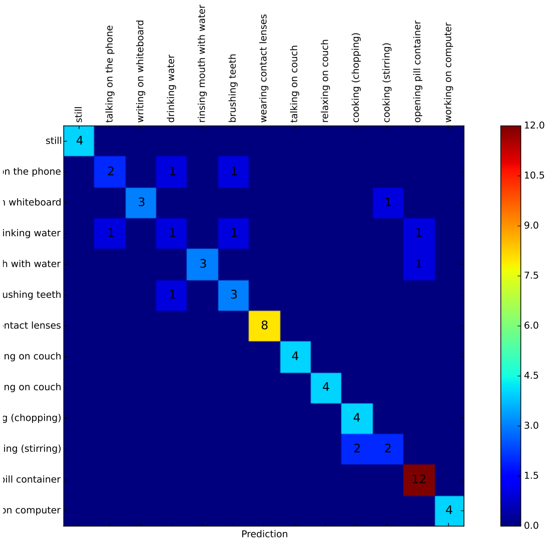

Figure 3.2: Confusion matrix: Bag-of-features approach with 128 clusters. precision 84%, recall 84%

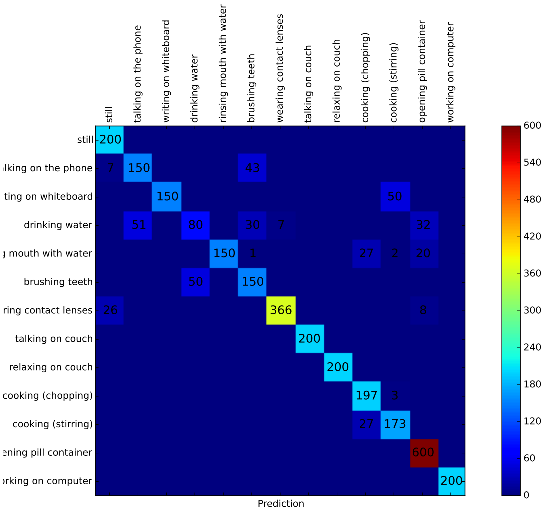

Figure 3.3: Confusion matrix: Bag-of-features approach with 128 clusters tested on intervals of 100 frames sampled 50 times from each test sequence. precision 88%, recall 88%

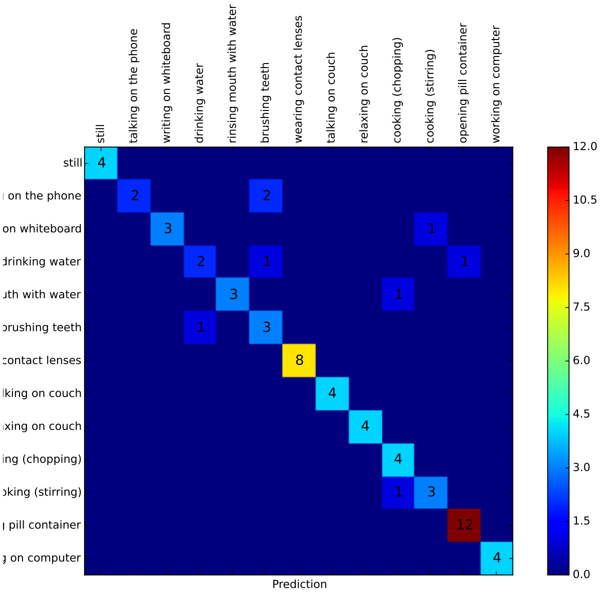

Figure 3.4: Confusion matrix: Longest common subsequence approach with 64 clusters per class. precision 90%, recall 88%

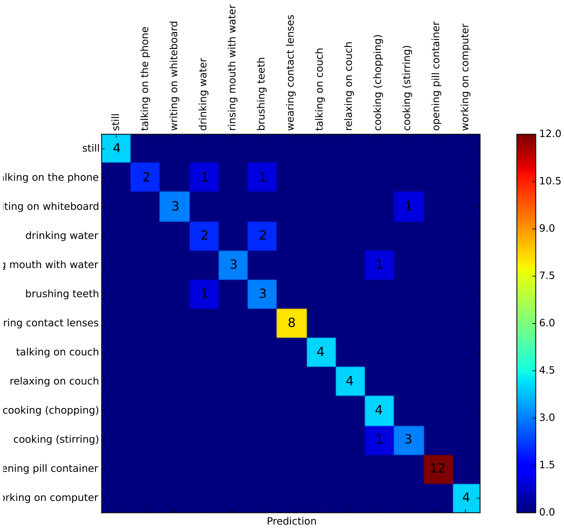

Figure 3.5: Confusion matrix: Dynamic Time Warping approach with 64 clusters per class. precision 90%, recall 88%

# 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 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. Asus XTION, Microsoft 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 will correct for the variance of skeleton lengths for 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.

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 difference vectors between two adjacent joints. An interesting approach is used in [28], 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 system, as features. This way the observation space could be reduced.

Some methods also extract temporal features (e.g. Eigenjoints [34]). 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 4.1 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.

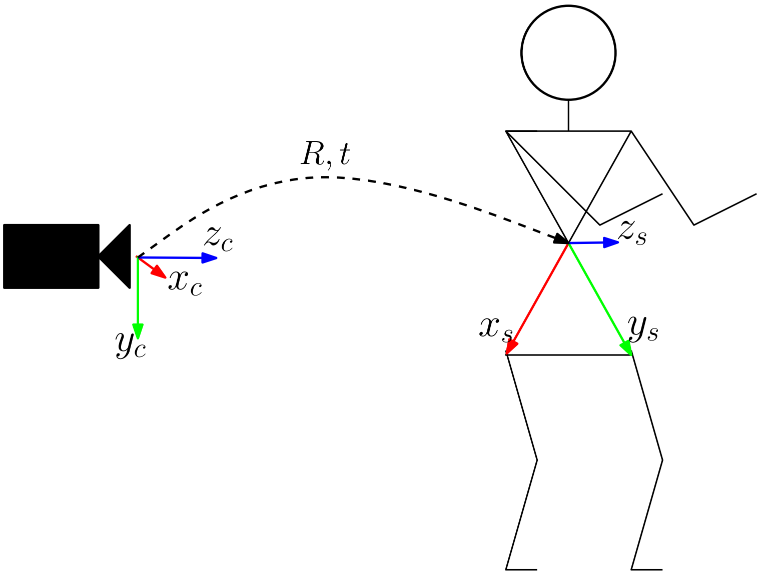

Figure 4.1: Sketch of the local skeleton frame inside the camera frame. The rotation matrix $\boldsymbol{R}$ and the translation vector $\boldsymbol{t}$ define the needed transformation to change from camera coordinates to the local skeleton coordinates

## 4.2 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 method 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 activities 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:

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

where $C_j$ is the set containing all class sequences and $|C_j|$ is the number of sequences in class $j$. The normalization factor $\min(|\boldsymbol{x}|, |\boldsymbol{x_n ew}|)$ 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 [20], 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).

### 4.2.1 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.

We tested the *DTW* using the mahalanobis and euclidean distance on the raw input data from the Cornell Activities, where we only translated the skeleton to the origin. We achieved a higher recognition rate with the mahalanobis-distance (ca. 10%). After that we also evaluated the measure on our extracted features and the recognition rate for the mahalanobis version dropped significantly. The euclidean distance performed better in this case. One possible explanation of this is that the raw features are not rotationally invariant and the euclidean distance becomes inappropriate, whereas the covariance matrix in the mahalanobis distance is able to capture this variance. Therefore the results on the raw data may be misleading. Further tests are needed to make a correct statement of the validity of a *mahalanobis based DTW* approach.

## 4.3 Discriminative Sequence Back-Constrained GP-LVM

In the paper "Discriminative Sequence Back-Constrained GP-LVM for MOCAP Based Action Recognition"[17] the authors propose a method for classifying MOCAP [2] actions.

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 [17] is illustrated in Figure 4.2

The idea is to perform a dimensionality reduction on the skeleton data, resulting in a much more 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.

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% [17].

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 captures the sequence distances.

### 4.3.1 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 interpreted as a sequence alignment kernel. This measure is to be preserved in the latent spaces.

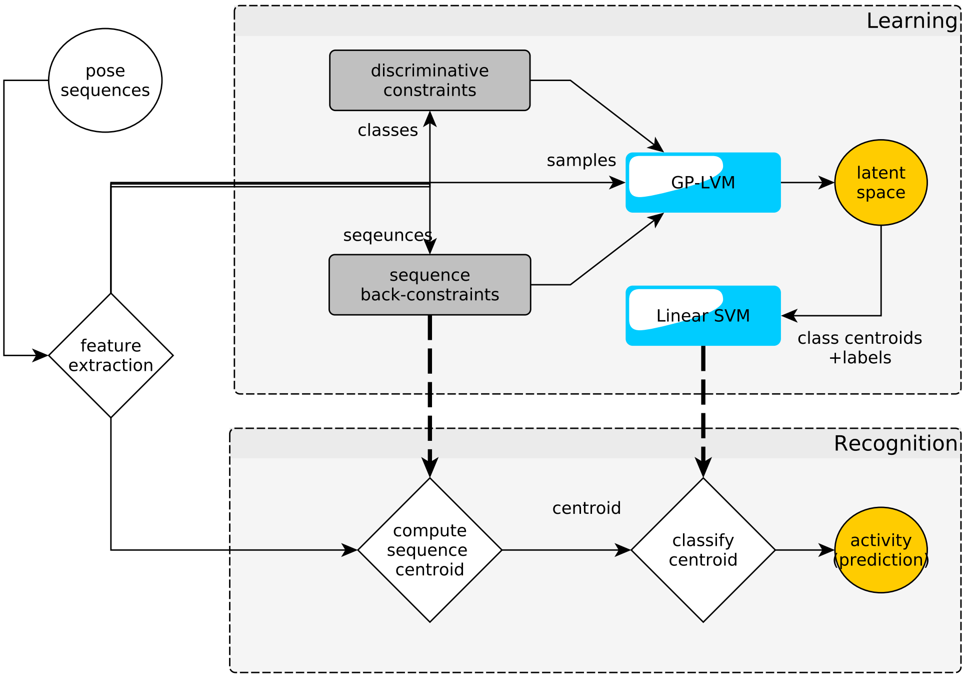

Figure 4.2: Illustration of the ”Discriminative sequence back-constrained GP-LVM” approach. Learning is done by training a GP-LVM together with the sequence-back constraints and discriminative constraints. After that an Linear SVM is trained by the class centroids and the corresponding labels. Recognition is performed by computing the centroid of a new sequence through the learned back-constraints and predicting with the SVM.

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

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

### 4.3.2 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 inner-class variance [31].

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 $\lambda$ 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 be too small and we will not gain any discrimination between the classes. Recognition is being done by applying the above mapping to the new sequence and using a SVM in the latent space.

### 4.3.3 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 sequence back-constraints. 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.

### 4.3.4 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 [17] 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. Moreover, as we do not know how long a sequence is, we have to calculate the centroid for several time intervals using a sliding window approach.

### 4.3.5 Planned extensions

1. Learn the 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 capture 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 4.2 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.

## 4.4 Implementation

As there was no publicly available source code and due to the fact that we wanted to integrate the code with ROS we choose to implement this method in Python. We used the *GPy* library which also has an *IVM* implementation of *GP-LVM*.

To implement the Discriminative GP-LVM constraints we ported 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 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.

## 4.5 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 due to the many constraints on the optimization 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 of lower dimensions. 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.

In "Exploring model selection techniques for nonlinear dimensionality reduction" [7] Harmeling argues that direct optimization of the *GP-LVM* is very difficult and also suggests to use *ISOMAP* or *LLE* for the initialization.

In the original paper [13] of the *GP-LVM* it was stated that the performance of the *GP-LVM* is strongly dependent on the initialization. It seems that, in our case, the objective function is highly non-linear and it is very likely that the optimization will find a local minimum in the vicinity of the starting position. Also using more sophisticated initialization techniques the problem of finding a good local minimum remains.

One possibility to solve this problem is to use even more sophisticated initializations. In [4] Bitzer and Williams suggest to make an initialization based on *Multidimensional Scaling*. The authors also achieve a higher likelihood on motion capture data sets using this metric in comparison to initializing with *LLE* or *ISOMAP*.

Because of the optimization problem 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 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* [33] 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 actions 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 [9]. 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 movements into account.

## 5.1 Gaussian Process Regression Flow

In [9] 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. The flow field representation is also suitable to model acceleration changes. The model works by first normalizing the trajectories in the time dimension. Then a point of the trajectory $\boldsymbol{x}$ can be defined by a three dimensional vector containing the 2D position and the progress in the time dimension. The velocity $\boldsymbol{y} = [y_x, y_y, y_t]$ in the spatio temporal domain is calculated. Then three *Gaussian Process Regressions* are used to learn the mapping from the the point $\boldsymbol{x}$ to each element of $\boldsymbol{y}$.

The learned regression can be interpreted as a dense flow field [9]:

$$\Phi(\boldsymbol{x}) = y'_x(\boldsymbol{x})\boldsymbol{i} + y'_y(\boldsymbol{x})\boldsymbol{j} + y'_t(\boldsymbol{x})\boldsymbol{k}$$

where $y'_x, y'_y, y'_t$ are the means of the *GP* regressions.

The authors also show a practical application of this method for real-time classification of trajectories of vehicles on street crossing. Recognition is performed by computing the posterior probability of the incoming sequence with the flow field for each class.

## 5.2 Gaussian Process - Latent Motion Flow

The *Gaussian Process - Latent Motion Flow* is a novel method and is inspired by the *GPRF* described in the earlier section. The difference is 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. Therefore 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 one, so we cannot fix the lengths of the trajectories. Nevertheless, resulting from the properties of *Gaussian Process Regression*, we can get a dense mean flow field and dense variances. This will allow 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 a motion flow field in the latent space for each activity. First we extract our features as described in 4.1. 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 Regression*. 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, 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*.

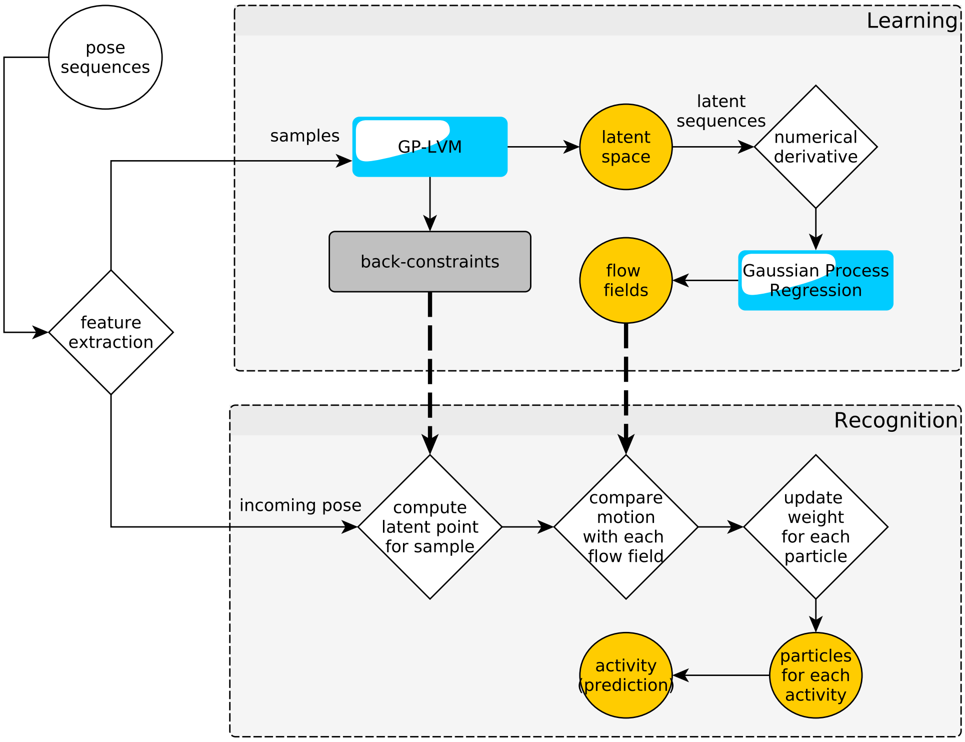

Figure 5.1: Illustration of the Gaussian Process - Latent Motion Flow approach.

Variances in the speed by which activities are performed 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 in the previous approach the *GP-LVM* handles the high-dimensional data. The regression inside the latent space, which can be interpreted as the learned motion tendencies of an activity, 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. This process can be performed for every incoming frame.

## 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 discriminative properties. 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 [32] 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 possibility of learning a unified latent space as 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 these 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.

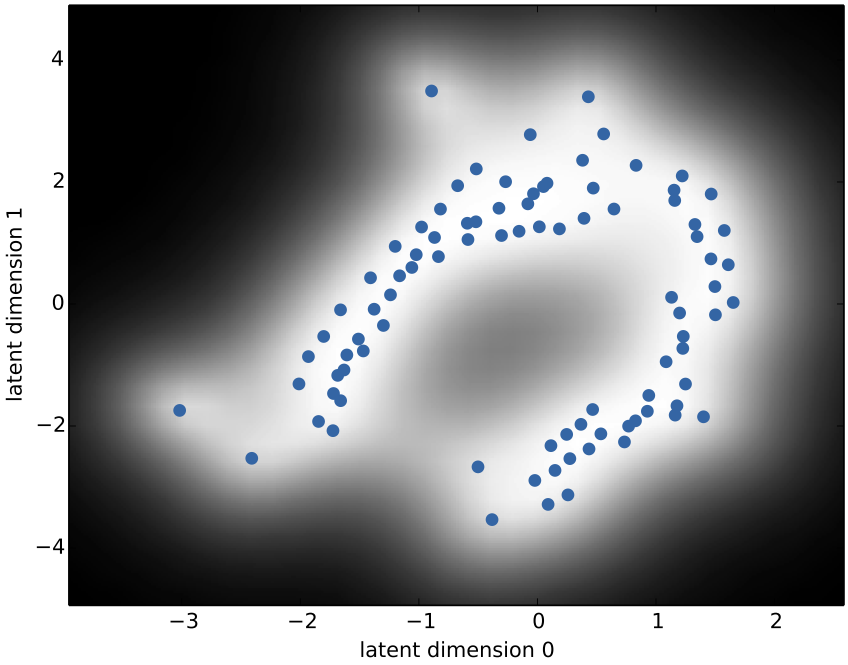

Figure 5.2: Two dimensional latent space representation of the "walking" sequence using GPy plot. The white area around the sample points represents the variance. MOCAP, subject 35, sequence 1.

One problem we encounter by learning the motion flow field from several samples is the complexity of the *Gaussian Processes*. 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. Changing the variance

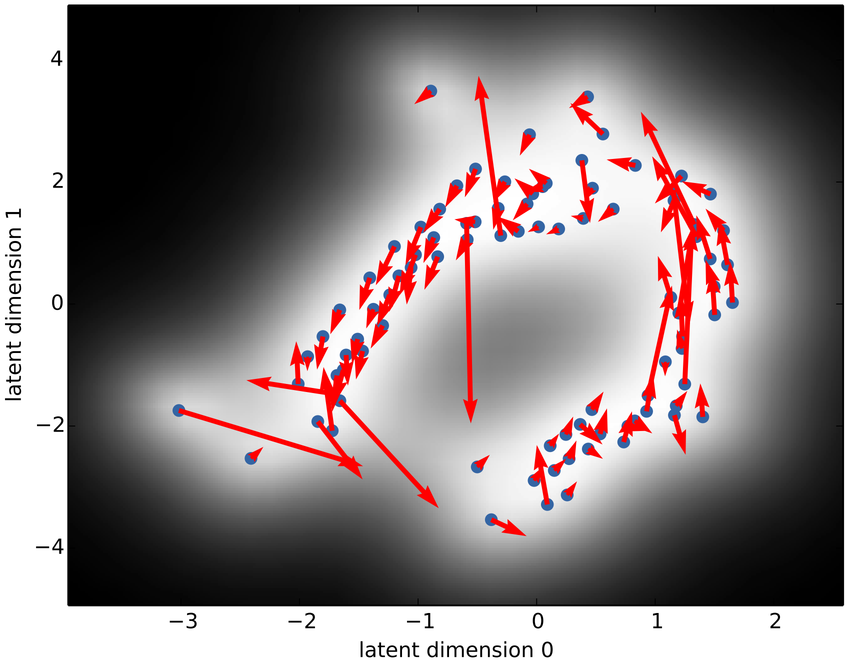

Figure 5.3: The calculated velocity (red) for each latent point. MOCAP, subject 35, sequence 1.

controls how fast the flow field looses strength when it goes outside of the main learned sample areas.

## 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.

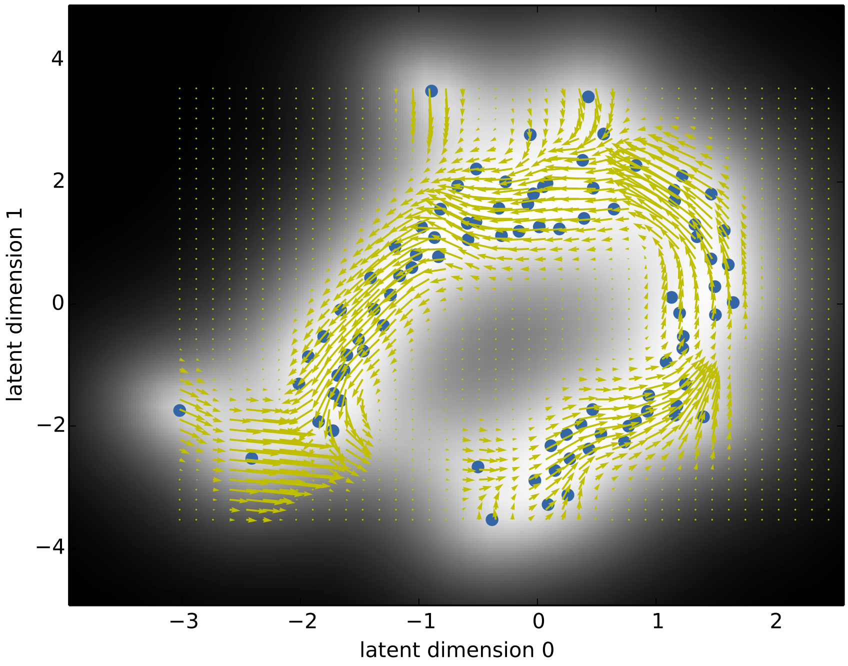

Figure 5.4: The learned flow field (yellow) from the velocities. lenthscale = 1, MOCAP, subject 35, sequence 1.

## 5.7 Advantages

**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. 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.

**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.

**Multiple Hypothesis Prediction** Since we have a *GP* representing for 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 a particle filter. Hence we can have multi-hypothesis predictions along with their probabilities.

**Modeling of cyclic activities** As this method is not trajectory based, but rather flow field based, recognition of cyclic activities is possible. Activities with repetitive motions, such as 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 defines the flow field.

**Online learning** Learning new samples can be accomplished by updating the *GP regression* with the new velocities. For this the hyperparameters have to be optimized, each time new sample points for the velocities come in. As the optimization is based on a gradient method, online learning is possible. One problem with this is that online gradient descent can get stuck in local minimum when the optimization is performed for streaming sample points. This can be mitigated by periodically re-learning the flow-fields using all samples.

**Novelty detection (anomaly detection)** In [9] the authors present the ability of the GPRF model for anomaly detection. This approach is also suitable for finding new classes as the energy value can be used to recognize novel activities. The reasoning is that if we cannot find a flow field with a big accumulated energy the activity has to be unobserved. Also with the two uncertainty values for the poses and the motion tendencies finding new activities makes the task easier.

Due to the fact that we use the *GP-LVM* and have a *GP regression*, we have two indicators for detecting unobserved data. The first one is the variance of the latent space when a new pose is mapped through the back-constraints. If it is high we know that the 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 did not 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.

**Semi-supervised segmentation of activities** With the ability to detect anomalies, described above, we could also segment a recorded sequence in known and unknown intervals. This way a supervisor can label the individual sequences. Moreover new classes of activities can be appended this way.

## 5.8 Implementation

Initially we used a non-sparse *GP-LVM* with a back-constrained *RBF* 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 a smooth function. This is a requirement to learn a robust *Gaussian Process Regression*. As the number of samples was very high, we sampled each activity sequence with a sparsity of 4. This reduced the learning time considerably but also introduced more discontinuities in the latent space.

We used the forward differences as the numerical derivative and learned the flow field by heuristically setting the length-scale with the average difference between the latent points. The initial variance was set to $10^{-3}$ as we wanted to reduce the effect on unobserved latent points.

The energy function was computed using the dot product of the motion vector with the *GP* prediction and multiplying with the variance of the *GP-LVM*.

## 5.9 Evaluation

Testing was done with only two and three 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 skeleton tracker 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. We also tested the dimensionality reduction on the Cornell Daily Living Activities data set. Unfortunately we were not able to perform an appropriate dimensionality reduction which results in a smooth mapping from observed space to latent space.

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 use a more appropriate initialization such as *PCA*, *LLE* or *ISOMAP*.

- 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.

This problem is not new. There are several publications in this area trying to change the optimization constraints of the *GP-LVM* in such a way that it results in a smoother latent space.

In [32] for example, Urtasun et al. integrate the *LLE* objective into the optimization of the *GP-LVM*. By first computing the weights in the observed space and then constraining the latent space to minimize with respect to those weights, the locality of neighboring points is preserved in the lower dimensional space. This results in a smoother latent space.

In ”Probabilistic Feature Extraction from Multivariate Time Series using Spatio-Temporal Constraints” [15] Lewandowski et al. introduce a new variant of the *GP-LVM* which uses a spatio-temporal prior on the latent points which constraints the optimization, such that the locality in respect to the samples themselves but also in respect to the time series data is preserved. This way subsequent poses in the sequence will also be located nearby in the latent space. Thus also the temporal structure of the time series is preserved. The authors also successfully tested and evaluated their method in the context of action recognition from videos which were recorded by different cameras at different angles.

The ability to also transfer the temporal relationship between the poses into the latent space is very promising, as it will help to get a smooth latent space, capable of producing a robust flow field. Unfortunately, due to time constraints and the complexity of an implementation of the spatio-temporal *GP-LVM*, we could not test this approach.

# 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 [36], which works by extracting representative poses and performs a histogram pooling over the quantized pose sequence. This method was extended using two sequence similarity measures (*LCS* and *DTW*). A *ROS* module was introduced which used the Cornell data set to perform activity recognition in real-time.

Due to the fact that the previous approach did not work online, but just classified a fixed interval of time, we tried another approach based on dimensionality reduction of the sequences and classification of the sequence centroids in latent space [17]. With this approach it was possible to classify a sequence by directly computing its centroid with the help of the *sequence back-constraints*. Unfortunately the dimensionality reduction approach for a high number of different complex activities proved to be extremely difficult.

Therefore we proposed a novel approach which first performs a dimensionality reduction for each activity class separately and then learns the flow field inside each latent space. This way the local motion tendencies of complex activities could be captured without resorting to trajectory based modeling methods. Online recognition is performed by updating each class probability with the incoming observation of the current pose. Experiments on the Cornell Activity data set revealed the issue of a non-smooth mapping between observed space and latent space, using a simple back-constrained *GP-LVM* approach. Thus it was proposed to use a more sophisticated method to perform dimensionality reduction on time series data using the *Spatio-temporal GP-LVM* [15].

## 6.2 Lessons learned

- Common dimensionality reduction for a large number of activities is intractable
- Optimization of the *GP-LVM* is very difficult and strongly depends on the initialization
- Local motion tendencies are more discriminative for complex activities then the overall dynamics

## 6.3 Contributions

- Implementation and extensions of a k-Means based approach in Python
- Implementation of a *ROS* module capable of activity recognition in real-time
- Implementation of the Discriminative GP-LVM in Python
- Implementation of the Sequence Back-constraints in Python
- A novel approach for activity recognition using latent motion flow fields
- Advantages and disadvantages of dimensionality reduction with GP-LVM in the context of activity recognition

## 6.4 Outlook

### Implementation of the GP-LMF using 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 backward mapping. One possibility to solve this problem is by implementing the *Spatio-Temporal GP-LVM* as described in [15]. Doing so the latent space will also take the temporal order of the poses into account, resulting in a smoother backward mapping (observed space to latent space).

### Adaptive GP Regression of the flow field

One issue 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. An approach that could be tested to capture this is using a non-stationary kernel.

### 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] SPENCER project - social situation-aware perception and action for cognitive robots. `http://spencer.eu`, 2013.

[2] Carnegie mellon university - CMU graphics lab - motion capture library. `http://mocap.cs.cmu.edu/`, 2014.

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

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

[5] Guoliang Fan, Xin Zhang, and Meng Ding. Gaussian process for human motion modeling: A comparative study. In *Machine Learning for Signal Processing (MLSP), 2011 IEEE International Workshop on*, pages 1–6. IEEE, 2011.

[6] Tim Field. openni_tracker - ROS wiki. `http://wiki.ros.org/openni_tracker`, 2014.

[7] Stefan Harmeling. Exploring model selection techniques for nonlinear dimensionality reduction. *University of Edinburgh, Tech. Rep. EDI-INF-RR-0960*, 2007.

[8] Phillip Hennig. Introduction to GPs II, gaussian processes winter school, sheffield, 2014.

[9] 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.

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

[11] Neil Lawrence. Latent variable models with gaussian processes, gaussian process winter school university of sheffield, UK, 2014.

[12] 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.

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

[14] 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.

[15] Michał Lewandowski, Dimitrios Makris, and Jean-Christophe Nebel. Probabilistic feature extraction from multivariate time series using spatio-temporal constraints. In Joshua Zhexue Huang, Longbing Cao, and Jaideep Srivastava, editors, *Advances in Knowledge Discovery and Data Mining*, number 6635 in Lecture Notes in Computer Science, pages 173–184. Springer Berlin Heidelberg, January 2011.

[16] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, August 2012.

[17] Valsamis Ntouskos, Panagiotis Papadakis, and Fiora Pirri. Discriminative sequence back-constrained GP-LVM for MOCAP based action recognition:. pages 87–96. SciTePress - Science and and Technology Publications, 2013.

[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, October 2011.

[19] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, June 2010.

[20] 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.

[21] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

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

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

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

[25] 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.

[26] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Human activity detection from RGBD images. *Plan, Activity, and Intent Recognition*, 64, 2011.

[27] the GPy authors. GPy: A gaussian process framework in python. `https://github.com/SheffieldML/GPy`, 2014.

[28] 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.

[29] Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[30] Pavan Turaga, Rama Chellappa, Venkatramana S. Subrahmanian, and Octavian Udrea. Machine recognition of human activities: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11):1473–1488, 2008.

[31] 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.

[32] 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.

[33] J.M. Wang, D.J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, February 2008.

[34] Xiaodong Yang and YingLi Tian. Effective 3d action recognition using EigenJoints. *Journal of Visual Communication and Image Representation*, 2013.

[35] Angela Yao, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Does human action recognition benefit from pose estimation? pages 67.1–67.11. British Machine Vision Association, 2011.

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