Summary. We present an evaluation of different techniques for the estimation of forces and torques measured by a single six-axis force/torque sensor placed along the kinematic chain of a humanoid robot arm. In order to retrieve the external forces and detect possible contact situations, the internal forces must be estimated. The prediction performance of an analytically derived dynamic model as well as two supervised machine learning techniques, namely Least Squares Support Vector Machines and Neural Networks, are investigated on this problem. The performance are evaluated on the normalized mean square error (NMSE) and the comparison is made with respect to the dimension of the training set, the information contained in the input space and, finally, using a Euclidean subsampling strategy.

Key words: Force sensing, machine learning, humanoid robotics.

1

Learning to Exploit Proximal Force Sensing: a Comparison Approach

Matteo Fumagalli, Arjan Gijsberts, Serena Ivaldi, Lorenzo Jamone, Giorgio Metta, Lorenzo Natale, Francesco Nori, and Giulio Sandini

Robotics, Brain and Cognitive Science Department
Italian Institute of Technology
Via Morego, 30 Genoa 16163
E-mail: {name.surname}@iit.it

1.1 Introduction

Emerging applications require robots to act safely in dynamic, unstructured environments. In order to avoid damaging the robot and the surrounding environment (physical objects and/or interacting agents), special sensors are usually applied to detect contact situations [21, 1]. Classical approaches to manipulation exploit a force/torque (FT) sensor placed on the end-effector, where most of the interactions occur. External forces acting on the other parts of the arm, however, cannot be measured with this configuration. Furthermore, it may not be feasible to put the sensor on the end-effector, due to its size or weight. An alternative solution is to place the sensor at the base of the manipulator or along the kinematic chain (e.g. after the shoulder) [19, 6]. In this case, the FT sensor measures both external and internal forces, the latter being the ones depending on gravitational, Coriolis and inertial forces. This solution allows the robot to detect interaction with the environment not only on the end-effector (e.g. voluntarily touching or grasping an object), but on the whole arm (e.g. hitting unexpected obstacles, being stopped by a human agent during motion). In order to accurately detect the external contribution of the forces, the manipulator dynamics must be compensated, i.e. the internal forces must be known, modeled or estimated.

Multiple approaches can be used for the estimation of these internal forces. Firstly, the functional estimation can be done using an analytical model describing the physics of the system, or at least its most significant properties. Model-based estimation strongly relies on the availability of a (mathematical) model of the robot [20], and is recommended only if the kinematic and dynamic parameters are known or identifiable with high accuracy. To this purpose, rigid multi-body dynamic modeling is generally used and some or all the parameters are identified [24] in order to improve the model accuracy. Within this context, the overall model accuracy is primarily limited by the

(potentially nonlinear) effects which the model does not explicitly take into account (e.g. gearbox backlash). Alternatively, supervised machine learning approaches can be used to approximate the internal dynamic model from a set of training examples. This approach may be preferred when explicitly modeling all possible nonlinear effects is cumbersome [25]. The main drawback of supervised learning methods is the need for collecting a rich and significant training set. Furthermore, it may be necessary to perform the training phase offline, due to the high computational requirements of these learning methods. In contrast, the model-based approach only needs to identify a small set of significant parameters; this identification technique requires much fewer data and computational resources and therefore can typically be performed efficiently online.

In this chapter, we investigate an analytical model and two supervised machine learning methods (Least Squares Support Vector Machines and Feed-forward Neural Networks) for the estimation of internal forces in a robotic arm, which is equipped with a six-axis FT sensor inside the kinematic chain. It seems reasonable to assume, however, that the results can be generalized to similar problems in robotics (i.e. problems related to the estimation of the dynamical parameters of a kinematic chain).

Firstly, we focus our attention on the amount of training data necessary to obtain accurate predictions. The qualitative measure for the prediction is the average Normalized Mean Square Error (NMSE) for the forces and torques in three dimensions. In our framework, the minimum amount of external forces that the robot can detect is proportional to the magnitude of this estimation error; this value is critical in order to have safe interaction with the environment. We expect machine learning methods to benefit from larger data sets; on the other hand, model based techniques should be more insensitive to the size of the training set.

Secondly, we pose our interest in understanding how the type of supplied data influences the estimation error. In particular, we verify empirically the usefulness of velocity and acceleration measurements when estimating (relatively complex) dynamical models. This issue is particularly important in the field of humanoid robotics, where smooth motions are usually preferred. As a consequence, velocities and accelerations need to satisfy some smoothness requirements, which prevent data from being completely random[1]. Therefore, successfully exploiting velocity and acceleration data is difficult, especially without specific sensors dedicated to their measurement.

At last, we analyze the effect of sampling distribution on the generalization performance. This analysis can give information about the way training

[1] Typical identification techniques make strong assumptions on the supplied data set. Machine learning techniques assume sufficiently distributed samples that cover the variability of the underlying function. Model based approaches assume persistently exciting conditions (see [10] for a definition).

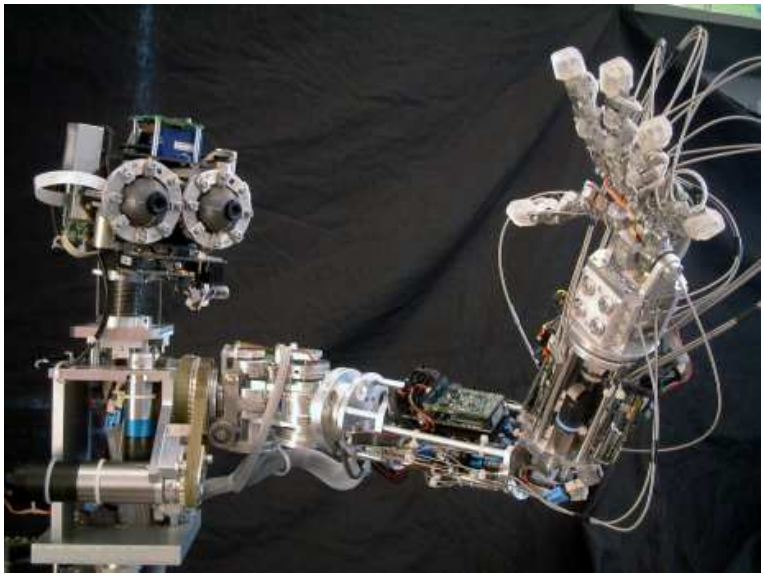

Fig. 1.1. The humanoid robot James.

data should be gathered or subsampled to practical dimensions from a larger training set.

The chapter is organized as follows. In Section 1.2 the robotic platform and the estimation problem are described. Section 1.3 describes three different approaches for internal forces estimation: an analytical model with identified parameters and the two machine learning methods. Experimental results are reported and discussed in Section 1.4. Section 1.5 contains the conclusions.

1.2 Robot Setup and Problem Formulation

This work has been carried out on the humanoid robot James [8]. James is a humanoid torso, consisting of a 7 DOF head, a 7 DOF left arm and a 8 DOF left hand, with the overall size of a 10 years old boy (cf. Fig. 1.1). Among the 7 degrees of freedom of the arm (3 for the shoulder, 1 for the elbow and 3 for the wrist), only 4 (the shoulder and elbow) have been considered in this work[2]. At the top of the upper arm, just below the shoulder, a single 6-axis FT sensor (ATI mini45 [15]) is placed (see Fig. 2(a)). This solution has been chosen because most of the space in the upper and forearm is occupied by the motors actuating the wrist, elbow and fingers, and by the DSP boards used to

[2] The justification for this simplification is that state of the wrist (position, velocity and acceleration) only has a minor effect on the internal forces, due to the relatively negligible amount of mass after the wrist (i.e. the hand mass) with respect to the whole arm.

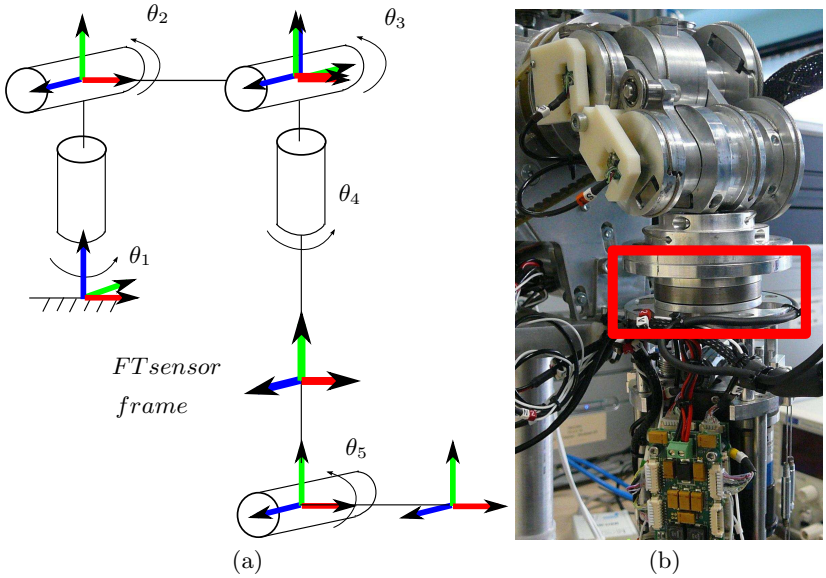

Fig. 1.2. (a) The reference frames for James' arm. Of the five joints, only four are independently actuated, θ_3 and θ_4 being mechanically coupled. (b) Detail of James' upper-arm, showing the FT sensor and its placement (red square).

control them. Furthermore, this placement enables the robot to detect both internal forces due to arm motion and external forces due to contacts between the arm/hand and the environment.

As explained before, the FT sensor measures both internal and external forces, the latter being the ones to be determined for interaction control purpose (e.g. obstacle detection and avoidance). The whole arm surface is taken into account for possible contact points (so the contact may happen in any point on the arm, not only on the end-effector). In the following we will discuss the retrieval of the external forces and the consequent need to estimate the internal ones.

Let us consider an open kinematic chain with n degrees of freedom. Let $\mathbf{q} \in \mathbb{R}^n$ be the generalized coordinates describing the pose of the kinematic chain. The FT sensor measurement will be denoted $\mathbf{x} = [\mathbf{f}^\top, \boldsymbol{\tau}^\top]^\top \in \mathbb{R}^6$. As previously said, this quantity contains both external and internal forces $\mathbf{f} \in \mathbb{R}^3$ and torques $\boldsymbol{\tau} \in \mathbb{R}^3$. Specifically we have:

$$\mathbf{x} = \mathbf{x}_I + \mathbf{x}_E \ , \tag{1.1}$$

where $\mathbf{x}_I$ and $\mathbf{x}_E$ refer to the internal and external forces/torques, respectively. More precisely, neglecting the effect of the elasticity of the transmissions and defining $\mathbf{f}_E, \boldsymbol{\tau}_E$ as the external forces and torques applied at the contact point, equation (1.1) can be expanded as follows (see [20] for details on the derivations):

$$\begin{bmatrix}\mathbf{f}\\ \boldsymbol{\tau}\end{bmatrix} = \underbrace{M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q})}_{\mathbf{x}_I} + \underbrace{T(\mathbf{q},\mathbf{d})\begin{bmatrix}\mathbf{f}_E\\ \boldsymbol{\tau}_E\end{bmatrix}}_{\mathbf{x}_E} \ , \tag{1.2}$$

where M, C and G are the inertial, Coriolis and gravity matrices of the dynamic system equations, and T is a roto-translation matrix describing the transformation of the external forces from the contact point reference frame to the sensor reference frame, with $\mathbf{d}$ being the distance vector from the contact point to the sensor.

Whenever the robot interacts with an external object, a non-null external force component $\mathbf{x}_E$ arises: in order to detect a collision or a contact, $\mathbf{x}_E$ must be identified from the sensor measurements $\mathbf{x}$. Practically, the identification can be performed by subtracting the internal forces ($\mathbf{x}_I$) from the measured ones ($\mathbf{x}$):

$$\mathbf{x}_E = \mathbf{x} - \mathbf{x}_I \ , \tag{1.3}$$

which yields an indirect measurement of the external forces and torques[3]. Then, the vector $\mathbf{x}_I$ must be computed from the model, or derived from experimental data. When the robot moves freely in its workspace, the sensor only perceives the internal components of forces and torques (i.e. $\mathbf{x}_I = \mathbf{x}$). These components only depend on position, velocity and acceleration of the joints. The problem of retrieving $\mathbf{x}_E$ is therefore reduced to the estimation of the internal forces and torques, i.e. the mapping from $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ to $\mathbf{f}, \boldsymbol{\tau}$:

$$\mathbf{x}_I = f(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}). \tag{1.4}$$

1.3 Proposed Approaches

Two distinct ways to identify $f(\cdot)$ in equation (1.4) are: (1) deriving it analytically, (2) approximating it using a set of examples (i.e. machine learning). In the latter case, the learning algorithm is agnostic to the underlying dynamics model that is used to produce the examples. One advantage of this approach is that nonlinear effects do not need to be explicitly modeled, as these are learned implicitly by the algorithm. In this section, we will detail the model-based approach and two machine learning algorithms, namely Least Squares Support Vector Machines and Neural Networks.

Remark 1. It is worth discussing some issues related to the noise effecting the measurement equation (1.4) which is at the base of all the proposed identification methods. Among the different contributions, the F/T sensor itself

[3] Notice that $\mathbf{x}_E$ does not correspond to the real external forces $\mathbf{f}_E$ and torques $\boldsymbol{\tau}_E$, but to their projection on the sensor, i.e. $\mathbf{x}_E = T(\mathbf{q},\mathbf{d})[\mathbf{f}_E^\top, \boldsymbol{\tau}_E^\top]^\top$. To retrieve the real external forces and torques, we must also know the distance $\mathbf{d}$ from the contact point to the sensor. In the future, we plan to mount a full body sensing skin [5], which will provide the necessary feedback to detect the contact location.

is a primary source of noise (see the specifications in [15]) but it is not the only one. As a matter of fact, also the velocity and acceleration measurements are subject to numerical inaccuracies, as these are computed using first and second order numerical differentiation of the position measurements. These derivatives are estimated based on the difference between the samples at time t and $t - W$, i.e.

$$\dot{\mathbf{q}}_t = \frac{\mathbf{q}_t - \mathbf{q}_{t-W}}{W\Delta T} \ , \ \ddot{\mathbf{q}}_t = \frac{\dot{\mathbf{q}}_t - \dot{\mathbf{q}}_{t-W}}{W\Delta T} \quad \text{for } t = W, \ldots, \infty \ . \tag{1.1}$$

This computation is performed on the DSP boards embedded in the robot arm at 1 kHz rate, i.e. $\Delta T = 1$ ms. The window length W has been set to 35, i.e. $W\Delta T = 35$ms[4]. Other sources of noise include communication delays effecting the synchronization of sensory measurements and reliability of the position measurements in presence of elasticity in the actuation design (elastic tendons and rubber transmission belts). The complex interaction of these various sources of noise makes it very difficult to characterize the overall system noise and therefore a complete analysis will be left outside the scope of the current paper.

1.3.1 Model-Based Approach

Let us consider a robotic manipulator with n degrees of freedom and links, and a force/torque sensor located in the middle of the kinematic chain, immediately after one of the joints. As already pointed out, the sensor will measure both internal ($\mathbf{x}_I$) and external ($\mathbf{x}_E$) force/torque component acting on the following links. In this section, we assume that the FT sensor measures only the internal forces, i.e. $\mathbf{x}_E \equiv \mathbf{0}$. Therefore, (1.2) can be written as:

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) \ . \tag{1.2}$$

Starting from this formulation, we derive a model based approach for estimating the parameters in (1.2). In order to tune this model, a set of parameters that best fits the force/torque acquisition, given a certain data set of joint positions $\mathbf{q}$, velocities $\dot{\mathbf{q}}$ and accelerations $\ddot{\mathbf{q}}$, needs to be found. Equation (1.2) is written as the linear product of a matrix $D(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ and a vector $\boldsymbol{\eta}$ (see [10] for details). The matrix $D(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ depends solely on the joint positions, velocities and accelerations, whereas $\boldsymbol{\eta}$ contains the dynamical parameters that we would like to estimate. Practically:

$$\mathbf{x} = \begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = D(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\eta} \ , \tag{1.3}$$

where $\boldsymbol{\eta}$ has a complex structure that can be formalized as follows:

[4] This window length has been chosen specifically to low-pass filter the position measurements and the computed velocities, whilst maintaining sufficient accuracy.

$$\boldsymbol{\eta} = [\psi,\ \Psi]^\top \quad .$$

The row vectors ψ and Ψ depend only on the system dynamical parameters and have the following structure:

$$\psi = \left[\, m_1\varphi \cdots\ m_n\varphi \,\right] \in \mathbb{R}^{n^\psi} \tag{1.4}$$

$$\Psi = \left[\, \mathbf{r}_1 \cdots\ \mathbf{r}_n \,\right] \in \mathbb{R}^{n^\Psi} \quad , \tag{1.5}$$

where

$$\varphi = \left[\, \mathbf{l}_1^\top \cdots\ \mathbf{l}_n^\top,\, \mathbf{c}_1^\top \cdots\ \mathbf{c}_n^\top \,\right] \in \mathbb{R}^{n^\varphi} \tag{1.6}$$

$$\mathbf{r}_i = \left[\, \mathbf{s}_{i,1} \cdots\ \mathbf{s}_{i,n},\, I_i^1 \cdots\ I_i^6 \,\right] \tag{1.7}$$

$$\mathbf{s}_{i,j} = \left[\, m_i\varphi_j^2\ m_i\varphi_j\varphi_{j+1} \cdots\ m_i\varphi_j\varphi_{n_\varphi} \,\right] \quad . \tag{1.8}$$

For each link $i = 1, \ldots, n$, $\mathbf{l}_i \in \mathbb{R}^3$ is the vector representing the lengths of the link in the x, y and z directions with respect to the previous joint's reference frame, $\mathbf{c}_i \in \mathbb{R}^3$ is the vector of the center of mass of each link, with respect to the same reference frame[5]. Further, $m_i \in \mathbb{R}$ and $I_i^k \in \mathbb{R}$ are the mass and inertial parameters of each link for $k = 1, \ldots, 6$. Interestingly, equation (1.3) can be further simplified to the form $\mathbf{x} = \hat{D}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\phi}$, where $\boldsymbol{\phi}$ is the minimum set of identifiable parameters, i.e. a linear combination of the elements of vector $\boldsymbol{\eta}$ (see [10] for details).

The vector $\boldsymbol{\phi}$ of the system dynamical parameters can be often retrieved from an accurate model of the robot (e.g. CAD drawings), but this procedure is typically neither feasible nor accurate. Different ways for identifying the dynamic parameters can be found in the literature, but their discussion is out of the scope of this work (the interested reader should refer to [6, 10]). Here we focus on a technique based on a weighted linear least squares solution. Let us define a weighting diagonal matrix ω containing the variances of each component of force (f_x, f_y, f_z) and torque (τ_x, τ_y, τ_z):

$$\omega = \begin{bmatrix} \frac{1}{\sigma^2_{f_x}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma^2_{f_y}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma^2_{\tau_z}} \end{bmatrix} \quad . \tag{1.9}$$

At the i^{th} time instant, we measure the system position ($\mathbf{q}_i$), velocity ($\dot{\mathbf{q}}_i$) and acceleration ($\ddot{\mathbf{q}}_i$) and the associated force sensor output $\mathbf{x}_i$. After ℓ time samples, a possible estimation for the vector $\boldsymbol{\phi}$ of dynamical parameters is

[5] Each kinematic chain link has an associated reference frame, defined by the Denavit-Hartenberg convention [3] . All the dynamic and kinematic quantities of each link (center of mass, inertia, lengths, etc.) refer to the associated reference frame.

given by the vector $\boldsymbol{\phi}^\circ$ which minimizes the (weighted) norm of the error vectors $(\mathbf{x}_i - \hat{D}_i\boldsymbol{\phi})$, i.e.:

$$\boldsymbol{\phi}^\circ = \arg\min_{\boldsymbol{\phi}} \sum_{i=1}^{\ell} (\mathbf{x}_i - \hat{D}_i\boldsymbol{\phi})^\top \omega (\mathbf{x}_i - \hat{D}_i\boldsymbol{\phi}) \ . \tag{1.10}$$

where we defined $\hat{D}_i = \hat{D}(\mathbf{q}_i, \dot{\mathbf{q}}_i, \ddot{\mathbf{q}}_i)$. The explicit solution is given by:

$$\boldsymbol{\phi}^\circ = \Delta^\dagger_\Omega \mathbf{Y} = \left[\Delta^\top \Omega \Delta\right]^{-1} \Delta^\top \Omega \mathbf{Y} \ , \tag{1.11}$$

where $\Omega = diag(\omega)$ and

$$\Delta = \begin{bmatrix} \hat{D}_1 \\ \hat{D}_2 \\ \vdots \\ \hat{D}_\ell \end{bmatrix} \ \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_\ell \end{bmatrix} \ . \tag{1.12}$$

Remark 2. Given the discussion above, learning the optimal parameters value $\boldsymbol{\phi}^\circ$ consists in a matrix inversion. With simple algebraic simplifications, it can be proved that the dimension of the matrix to be inverted does not depend on the number of acquired data but only on the dimension of the vector $\boldsymbol{\phi}$. Similarly, once the model has been trained, the model prediction (the prediction of $\mathbf{x}$ given $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$) consists in evaluating $\hat{D}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ and the product $\hat{D}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\phi}^\circ$. Therefore, the computational complexity of the prediction depends mainly on the evaluation of the matrix $\hat{D}$ (in our example represented by ~ 1700 multiplications, ~ 700 sums and 8 sine/cosine evaluations).

1.3.2 Least Squares Support Vector Machines for Regression

Least Squares Support Vector Machines (LS-SVMs) belong to the class of kernel methods, which use a positive definite kernel function to estimate a linear approximator in a (usually) high-dimensional feature space [23]. Its formulation shares similarities with the Support Vector Machine for Regression (SVR) [22]. Let us define the data set $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^{\ell}$, where inputs $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding outputs $y_i \in \mathbb{R}$ for $i = 1, \ldots, \ell$. LS-SVM estimates a linear decision function of the form $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b$, where b is a bias term and $\phi(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^f$ maps samples from the input space into a (usually) high-dimensional feature space. The weight vector $\mathbf{w}$ and bias b are chosen such that both the squared norm of $\mathbf{w}$ and the sum of the squared errors $\epsilon_i = y_i - f(\mathbf{x}_i)$ are minimized. This is described by the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}C\sum_{i=1}^{\ell}\epsilon_i^2 \\ \text{subject to} \quad & y_i = \langle \mathbf{x}_i, \mathbf{w} \rangle + b + \epsilon_i \qquad \text{for } 1 \le i \le \ell \ , \end{aligned} \tag{1.13}$$

where C is a regularization constant. Standard application of the Lagrange method yields the dual optimization problem [23]:

$$\text{maximize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}C\sum_{i=1}^{\ell}\epsilon_i^2 - \sum_{i=1}^{\ell}\alpha_i\left(\langle\mathbf{x}_i,\mathbf{w}\rangle + b + \epsilon_i - y_i\right) \quad . \tag{1.14}$$

Here $\alpha_i \in \mathbb{R}$ are the Lagrange multipliers associated with each sample. Using this dual formulation, the decision function can be rewritten as $f(\mathbf{x}) = \sum_{i=1}^{\ell}\alpha_i\langle\phi(\mathbf{x}_i),\phi(\mathbf{x})\rangle + b$. One particular advantage of this expansion is that the solution is described in terms of inner products with respect to the training samples $\mathbf{x}_i$. Hence, a kernel function $k(\mathbf{x}_i,\mathbf{x}_j) = \langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle$ can be used to implicitly map the data into the feature space. Given a kernel matrix $K = \{k(\mathbf{x}_i,\mathbf{x}_j)\}_{i,j=1}^{\ell}$, the solution to the optimization problem in (1.14) is given by a system of linear equations:

$$\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix} = \begin{bmatrix}K + C^{-1}I & \mathbf{1}\\ \mathbf{1}^T & 0\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{y}\\ 0\end{bmatrix} \quad . \tag{1.15}$$

Note that solving the LS-SVM optimization problem reduces to a $(\ell+1)\times(\ell+1)$ matrix inversion, which in return can be solved efficiently using Cholesky decomposition [2]. Another advantage of LS-SVM over other kernel methods (e.g. SVR), is that the Leave-One-Out (LOO) error can be computed exactly using a single training run on the complete data set [2]. It is important to note that the final generalization performance of the LS-SVM is strongly dependent on the selection of both C and the kernel function. For our experiments, we consider the commonly used Radial Base Function (RBF) kernel $k(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2)$, where parameter γ tunes the radius of the Gaussian. A grid search on the range $C \in \left[2^0, 2^1, \ldots, 2^{16}\right]$ and $\gamma \in \left[2^{-11}, 2^{-10}, \ldots, 2^1\right]$ is used to select "optimal" hyperparameters, where the generalization performance of each configuration is estimated using the LOO error on the training set. Furthermore, we considered the cases that $\mathbf{x} = \{\mathbf{q}, [\mathbf{q},\dot{\mathbf{q}}], [\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}], [\mathbf{q},\ddot{\mathbf{q}}]\}$. As the output y is limited to scalar values, a distinct machine has to be trained for each output dimension, such that $y = \{f_x, f_y, f_z, \tau_x, \tau_y, \tau_z\}$.

Remark 3. Though LS-SVM has several advantageous properties with respect to SVM, one apparent disadvantage is that it does not produce sparse models. Input samples $\mathbf{x}_i$ can only be removed from the kernel expansion when $\alpha_i = 0$, which in return is only the case if $\epsilon_i = 0$. As a result, on practical problems all input samples will be included in the model. This reflects negatively on the prediction time. For m output dimensions and assuming the RBF kernel function, the prediction of an n dimensional input vector requires $m\left(\ell(n+1)+1\right)$ sums, $m\ell(n+2)$ products, and $m\ell$ exponentials; where ℓ is the size of the data set the m distinct machines has been trained on.

Remark 4. The scope of this work is limited to batch learning on small to medium sized data sets. Nguyen-Tuong et al. (cf. [18] in this volume) have

shown that – on a similar learning problem – Gaussian Processes Regression (GPR) commonly outperforms other methods in this particular context. It is worth noting that the performance of LS-SVM can be expected to be similar to GPR, as both methods share some similarities in the approximation function[6].

1.3.3 Neural Networks

Lastly, a multiple input - multiple output one-hidden-layer (OHL) feed-forward neural network (NN) is chosen as the second machine learning method, for its generalization and approximation capabilities [9], and denoising property when dealing with experimental data. More specifically, we constrain the approximation function to take on a fixed, parameterized structure, that is a ν "neurons", n inputs, m outputs neural network, $\hat{\boldsymbol{\mu}}(\cdot, \mathbf{w})$, with sigmoidal activation functions σ in both hidden and output layer[7] :

$$\hat{\boldsymbol{\mu}}(\tilde{\mathbf{x}}, \mathbf{w}) = col\left(\tilde{\boldsymbol{\sigma}}_j\left[\sum_{h=1}^{\nu} c_{hj}\sigma(\tilde{\mathbf{x}}, \boldsymbol{\kappa}_h) + b_j\right], j = 1, \ldots, m\right) \tag{1.16}$$

where $\hat{\boldsymbol{\mu}}(\cdot, \mathbf{w}) : \mathbb{R}^n \times \mathbb{R}^W \mapsto \mathbb{R}^m$, $c_{hj}, b_j \in \mathbb{R}, \boldsymbol{\kappa}_h \in \mathbb{R}^{n+1}, j = 1, \ldots, m$, being ν the number of *neurons* constituting the network. The vector $\mathbf{w} \in \mathbb{R}^W, W = (n+1)\nu + (\nu+1)2m$ collects all the parameters to be optimized. The notations $\tilde{\boldsymbol{\sigma}}$ and $\tilde{\mathbf{x}}$ account for the output and input normalization.[8] Furthermore, we trained four different type of networks, with $\mathbf{x} = \{\mathbf{q}, [\mathbf{q}, \dot{\mathbf{q}}], [\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}], [\mathbf{q}, \ddot{\mathbf{q}}]\}$ and $n = \{4, 8, 12\}$ respectively, for different number of neurons $\nu = 5, 20, 50, 100, 150$ and different training sets. The number of outputs was always fixed to 6 (forces and torques). The training algorithm is based on the well known Levenberg-Marquardt (LM) algorithm [12, 13]. The criterion for training the network (that is to find the optimal parameters $\mathbf{w}^\circ$) is to minimize the mean square error between the estimated and the measured data:

$$\text{minimize } \Phi(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{\ell} \boldsymbol{\epsilon}_i^\top(\mathbf{w})\boldsymbol{\epsilon}_i(\mathbf{w}) \ , \tag{1.17}$$

where $\boldsymbol{\epsilon}_i(\mathbf{w}) = \mathbf{y}_i - \hat{\boldsymbol{\mu}}(\tilde{\mathbf{x}}_i, \mathbf{w})$ is the error between the measured and the predicted data. Once all the partial derivatives of the error function Φ are

[6] The main differences between both methods are that LS-SVM includes a bias term and requires less assumptions on the distribution of the data.

[7] We chose a sigmoidal output layer (instead of a classical linear output layer) since it naturally generates bounded values within a specific range, which are consistent with the output ranges after data normalization. This choice allows to remove signal constraints and not to take care of the possibility that the network generates inconsistent values.

[8] The input variables are normalized from their original range to $[-1, 1]$, while the network outputs are scaled from $[-1, 1]$ (the output range of a sigmoidal tanh-based neural network) to the forces and torques real ranges.

back-propagated, the weights update equation is applied:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - [J^\top(\mathbf{w}_k)J(\mathbf{w}_k) + \mu I]^{-1} J^\top(\mathbf{w}_k)\boldsymbol{\epsilon}(\mathbf{w}_k) \ , \tag{1.18}$$

where $\boldsymbol{\epsilon}(\mathbf{w}_k) = [\boldsymbol{\epsilon}_0(\mathbf{w}_k), \ldots, \boldsymbol{\epsilon}_{N-1}(\mathbf{w}_k)]$, and $J(\mathbf{w}_k) \in \mathbb{R}^{N \times W}$ is the Jacobian matrix of the errors with respect to the parameters of the NN:

$$J(\mathbf{w}) = \begin{bmatrix} \frac{\partial \boldsymbol{\epsilon}_0}{\partial w_0} & \frac{\partial \boldsymbol{\epsilon}_0}{\partial w_1} & \cdots & \frac{\partial \boldsymbol{\epsilon}_0}{\partial w_{W-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{\epsilon}_{N-1}}{\partial w_0} & \frac{\partial \boldsymbol{\epsilon}_{N-1}}{\partial w_1} & \cdots & \frac{\partial \boldsymbol{\epsilon}_{N-1}}{\partial w_{W-1}} \end{bmatrix} . \tag{1.19}$$

The parameter μ, adjusted iteratively, balances the LM between a steepest descent and a Gauss-Newton algorithm. To improve the training performance we used the Nguyen-Widrow (NW) method [17] to initialize the network (see also [7, 14]).

Remark 5. The proposed neural network training is designed for batch learning, and generically the estimate improves with the growth of both training set and number of parameters[9]. Since the training is performed offline, in the prediction phase the computation is quite fast, consisting only of a single forward pass of the network. More precisely, given the number of neurons ν for a OHL neural network with n inputs, m outputs, sigmoidal activation functions (hyperbolic tangent $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$) in the hidden and output layer, the necessary operations are $\nu(n + m)$ products, $\nu(n+m+1)+m+2(\nu+m)$ sums, $2(\nu+m)$ exponentials and $\nu+m$ divisions, and the flops count is linear with ν. As an example, for 20 neurons, 4 inputs, 6 outputs and both layers with the usual hyperbolic tangent, the flops count (computed with the Lightspeed Matlab toolbox v.2.2. [16]) is 2766.

1.4 Results and Discussion

The three previously discussed methods have been evaluated experimentally on a common data set that has been gathered during a sequence of random arm movements, performed in joint space. Every movement brings the arm from a starting position $\mathbf{q}_\mathrm{s} \in \mathbb{R}^4$ to a final position $\mathbf{q}_\mathrm{f} \in \mathbb{R}^4$, which subsequently becomes the starting position for the next movement. Each of these positions is defined by a vector of joint angles, which are chosen randomly using a uniform distribution within the admissible range of the respective

[9] The number of parameters usually depend on three factors: the complexity of the function to be approximated (i.e. a very smooth function requires fewer neurons than a highly varying one, as more basis function are necessary to approximate the irregular changes of the latter), the dimension of the training set and the quality of the training set (i.e. to which extent the training set is representative for the variable space).

	$q[^\circ]$				$\dot{q}[^\circ/s]$				$\ddot{q}[^\circ/s^2]$			
joint #	0	1	2	3	0	1	2	3	0	1	2	3
max	150	60	30	70	18	15	20	15	103	76	248	34
min	50	-100	-60	10	-39	-49	-34	-23	-135	-85	-158	-38

Table 1.1. Value ranges of the arm joint positions, velocities and accelerations.

joint. Joint velocity profiles during motion are *bell-shaped* with a predefined maximum velocity, which causes the absolute velocities to vary from zero to the maximum value during any movement. Trivially, the sign of the velocity depends on the direction of the motion. The joint accelerations (i.e. actual slope of the velocity profiles) depend on the distance between $\mathbf{q}_\mathrm{s}$ and $\mathbf{q}_\mathrm{f}$, since the time duration of the movement is kept constant. Joint positions, velocities and accelerations were retrieved from the DSP boards at 50 Hz. Velocities and accelerations were computed via numerical differentiation on the DSP boards at a higher frequency (1 kHz). A simple collision avoidance strategy was used during the experimental data acquisition, in order to ensure that the arm would not collide with the body or the environment.

The complete data set of 40000 samples has been shuffled and split in two equal parts. The set of the first 20000 samples is used for training and is subsequently subsampled to obtain smaller sized training sets, whereas the second half is used as a common test set. The reported performance measure on the test set is the average Normalized Mean Squared Error (NMSE) over all 6 output dimensions, where the NMSE is defined as the mean squared error divided by the variance of the respective output dimension. For the two machine learning approaches, the input dimensions have been rescaled (see original ranges Table 1.1) to be approximately within the range $[-1, +1]$, based on the maximum and minimum values found in each input dimension in the training set.

1.4.1 Number of Training Samples

In this initial experiment we measured the performance of each method when increasing the number of training samples. The results in Fig. 1.3 show clearly that the two learning methods have a strong dependency on the size of the training set. As more samples become available, they consistently continue to improve performance, eventually outperforming the model-based approach by an order of magnitude.

Interestingly, the model-based approach appears to perform at a constant level, regardless of the number of samples. This is confirmed by further analysis on even smaller data sets, as demonstrated in Fig. 1.4. When considering only the joint positions, it shows the remarkable capability of achieving acceptable performance using only 5 training samples. This means that the model-based approach is the preferred approach when only very few samples are available. The machine learning methods require many more samples to achieve similar

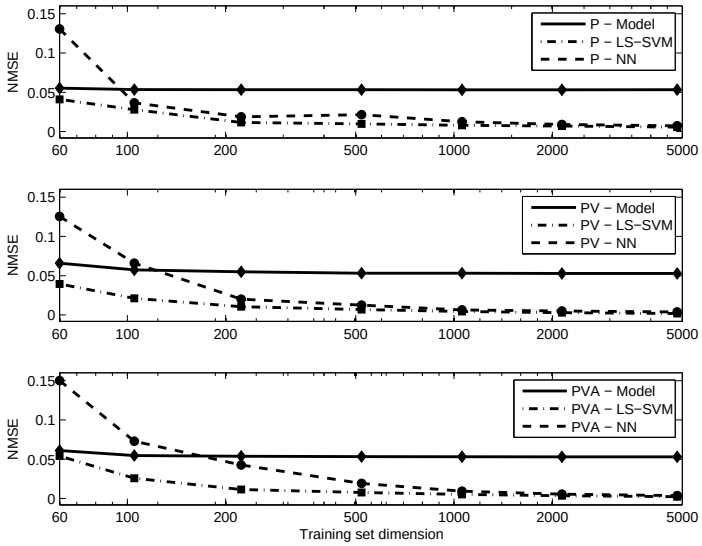

Fig. 1.3. Comparison of the three methods on random training subsets of increasing dimension and three different input spaces. P denotes the input space containing only joint positions ($\mathbf{q} \in \mathbb{R}^4$), PV contains both joint positions and velocities ($\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^8$), and PVA contains joint positions, velocities and accelerations ($\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^{12}$).

performance. This is not surprising considering that the model based approach takes advantage of additional information implicit in the structure of the model.

1.4.2 Contribution of Velocity and Acceleration on the Estimation

Another observation (Fig. 1.5) is that inclusion of joint velocities and accelerations does not always improve the generalization performance when training is done on a small number of samples. Intuitively, one might expect that adding relevant information could only improve the estimation. However, learning methods require an increasing amount of training samples to make effective use of this additional information (i.e. the *curse of dimensionality* [4]). This affects particularly the learning methods, since these need to construct their model based solely on training data. Fig. 1.5 shows that both LS-SVM and NN eventually use joint velocities to improve their predictions, given a sufficiently large training set.

Joint accelerations, however, do not improve prediction performance in any of the cases (cf. Fig. 1.6). This is probably due to the low signal to noise ratio for the acceleration and, in first place, to the robotic setup used to obtain

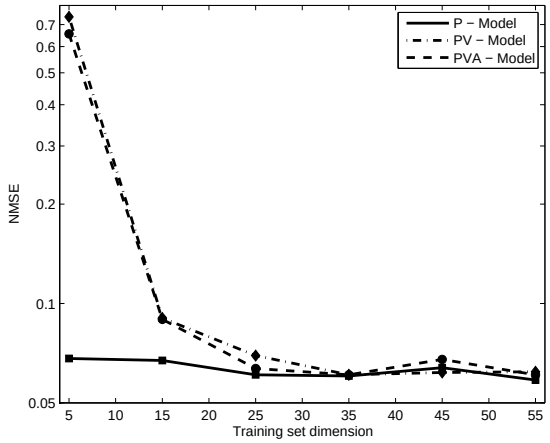

Fig. 1.4. Performance of the model-based method on very small training sets.

the data set. In particular, the joint accelerations were not measured directly but were derived from positions. This causes the acceleration measurements to be much less precise and reliable than those for joint velocities and positions. Furthermore, the range of accelerations is relatively small[10] and within this range we observed that the contribution of $M(\mathbf{q})\ddot{\mathbf{q}}$ in equation (1.2) is relatively small compared to the contribution of the other terms ($C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ and $G(\mathbf{q})$).

1.4.3 Selective Subsampling

The data set we acquired is characterized by the fact that there is an abundance of training samples. Thus far, we have used a uniform random subsampling strategy, as to ensure that the qualitative properties of the subset approximate those of the original data set. However, with such an abundance of samples it is likely that the original data set is *oversampled* (i.e. additional samples do not further increase the generalization performance) and contains samples that are (nearly) identical to each other. This similarity of input samples particularly affects LS-SVM, as this method describes the prediction function in terms of inner products with respect to all training samples.

We can guarantee a certain "sparsity" of the training set by taking a subset, such that the inter-sample distance is at least a threshold t. Let us define a distance measure $D(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)\Sigma^{-1}(\mathbf{x}_i - \mathbf{x}_j)}$, where $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^n$ and Σ is an $n \times n$ matrix containing the variances of all input dimensions on

[10] The chosen motors produce limited torques, which reflects into relatively low accelerations.

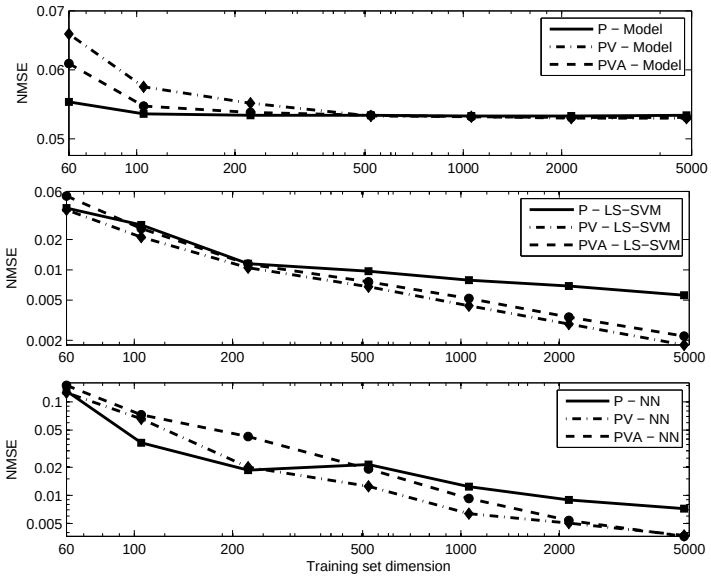

Fig. 1.5. Comparison of the performance for all methods with increasing input spaces (i.e. P, PV and PVA; defined as in Fig. 1.3). Note that both axes are in logarithmic scale to accentuate differences in final performance.

its diagonal. This measure coincides with the Euclidean distance in the standardized input space. In order to construct a Euclidean subset $\mathcal{E}_t$, we iterate over a permutation of the original data set $\mathcal{S}$ using index $i = 1, \ldots, \ell$ and append only those samples to $\mathcal{E}_t$, for which $\min D(\mathbf{x}_i, \mathbf{x}) \geq t \; \forall \mathbf{x} \in \mathcal{E}_t$.

Fig. 1.7 shows the prediction performance of LS-SVM with random and Euclidean subsampling. The Euclidean subsets were generated by varying the threshold t, such that the size of the subsets were nearly identical to each of the random subsets. Whether selective subsampling based on Euclidean distance outperforms random subsampling depends on the input space that is used to determine the inter-sample distance, and the size of the training set. When this distance is determined solely based on the joint positions, then Euclidean subsampling results in a significant improvement for small data sets. In contrast, random subsampling performs better than Euclidean subsampling based on joint positions, velocities and accelerations. It is our belief that this difference is due to the Euclidean strategy attempting to create a uniform sampling distribution in all dimensions under consideration, effectively forming subsets that contain a wide range of velocities and accelerations (besides positions). Given the relatively low velocities and accelerations of the robot, the force and torques are primarily caused by gravity. In return, gravity is only dependent on the joint positions of the robot. It is therefore beneficial,

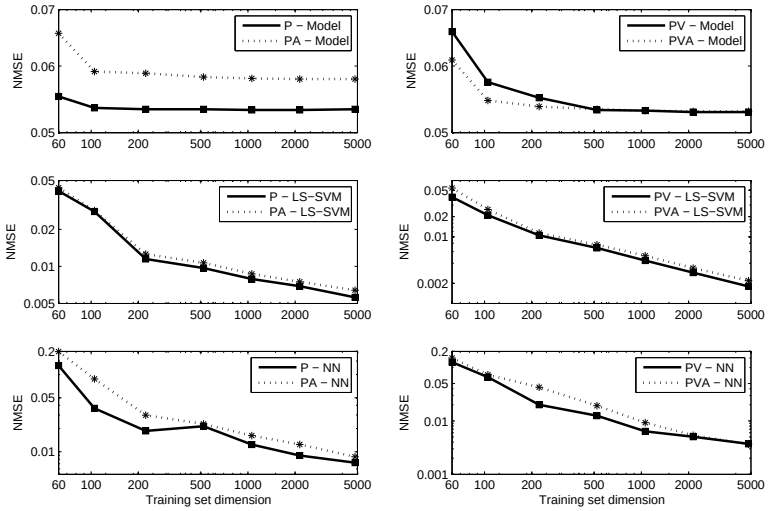

Fig. 1.6. Performance after inclusion of joint accelerations. The figures on the left hand side compare the performance on P and PA (defined analogously to P, PV and PVA in Fig. 1.3), while those the right hand side compare PV and PVA.

for limited training sets, to select those samples that help LS-SVM to model this gravity component.

Further, we can note that the different subsampling strategies perform nearly identically for large training sets. This can easily explained by the fact that the size of $\mathcal{E}_t$ is inversely proportional to the chosen distance threshold t and, by definition, $\mathcal{E}_t$ becomes a random permutation of $\mathcal{S}$ as t approaches zero. In short, for large data sets, and thus a small inter-sample threshold, the Euclidean and random subsets have very similar sample distributions and therefore similar performance.

1.5 Conclusions

In this paper, we presented and compared different approaches to force/torque estimation from a FT sensor. Experimental data-driven learning methods are proposed and compared with respect to the classical model-based technique, and advantages and disadvantages of both types of approaches are discussed. Learning algorithms outperform the rigid body dynamic model in terms of prediction accuracy, given that a sufficient amount of training data is available. The generalization performance of these methods improves steadily as more training samples become available. LS-SVM converges slightly faster

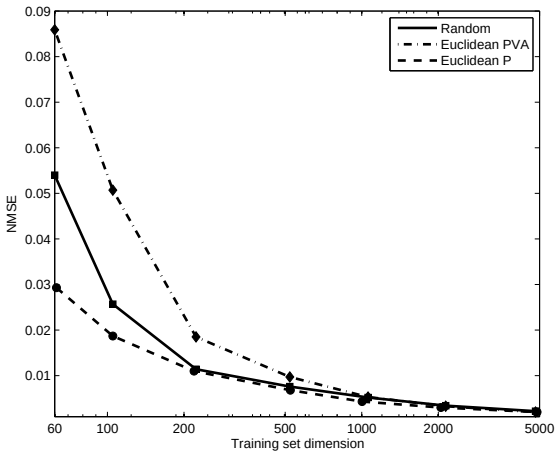

Fig. 1.7. Comparison of random and selective subsampling based on standardized Euclidean distance. *Euclidean P* and *Euclidean PVA* denote subsampling based on Euclidean distance thresholds $t = \{1.35, 1.15, 0.88, 0.65, 0.5, 0.35, 0.18\}$ using position inputs and thresholds $t = \{6.0, 5.3, 4.5, 3.7, 3.1, 2.5, 1.8\}$ using position-velocity-acceleration inputs, respectively.

than Neural Networks, but their final performance on large data sets is nearly identical. The model-based method, on the other hand, requires very few samples (in order of ten) to achieve acceptable predictions, and in practice requires very few samples if the estimate relies only on joint positions. Furthermore, we tested the relevance of velocity and acceleration information when learning the dynamic equation which describes the FT measurement. It was observed that machine learning methods improve their prediction when including velocity data at the cost of requiring larger training sets. On the contrary, acceleration does not significantly improve performance and the reason for this incongruence has to be found in the low signal to noise ratio (positions are double differentiated to obtain accelerations). We also evaluated the impact of training set pre-processing by defining a suitable selective subsampling strategy: a Euclidean distance metric was introduced and the resulting training set was compared with a random sampling one. For LS-SVM, the resulting spatial distribution of the training samples improves the estimation for small data sets, while its beneficial effect disappear with increasing the size of the training set.

Acknowledgment

This work was supported by European Commission projects RobotCub (IST-004370), ITALK (ICT-214668) and CHRIS (ICT-215805).

References

1. S. Haddadin A. De Luca, A. Albu-Schaffer and G. Hirzinger. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1623–1630, 2006.
2. G. C. Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *IJCNN-2006: Proceedings of the International Joint Conference on Neural Networks*, pages 1661–1668, Vancouver, BC, Canada, July 2006.
3. J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, 23:215–221, 1955.
4. Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.
5. G. Metta G. Cannata, M. Maggiali and Sandini. An embedded artificial skin for humanoid robots. In *Proc. of IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pages 434–438, 2008.
6. S. Dubowsky G. Liu, K. Iagnemma and G. Morel. A base force/torque sensor approach to robot manipulator inertial parameter estimation. In *Robotics and Automation, 1998. ICRA 1998. Proceedings of the 1998IEEE International Conference on*, 1998.
7. M. T. Hagan and M. B. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5:989–993, 1994.
8. L. Jamone, F. Nori, G. Metta, and G. Sandini. James: A humanoid robot acting over an unstructured world. In *International Conference on Humanoid Robots*, Genova, Italy, 2006.
9. M. Stinchombe K. Hornik and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.
10. Krzysztof Kozlowski. *Modelling and Identification in Robotics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.
11. M. Lagarde, P. Andry, P. Gaussier, S. Boucenna, and L. Hafemeister. Proprioception and imitation: on the road to agent individuation. In *From Motor to Interaction Learning in Robots*. Springer, 2009.
12. K. Levenberg. A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.
13. D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.
14. S. Ivaldi M. Fumagalli L. Jamone F. Nori L. Natale G. Metta and M. Baglietto. Estimation of forces and torques in a humanoid arm: comparison of model based and offline/online learning techniques. In *Submitted to the 48th IEEE Conference on Decision and Control*, 2009.
15. ATI mini45. 6 axes f/t sensor. `http://www.ati-ia.com/products/ft/ft_models.aspx?id=Mini45`.
16. Tom Minka. Lightspeed toolbox. `http://research.microsoft.com/en-us/um/people/minka/software/lightspeed/`.
17. D. Nguyen and B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In *Proc. of the Int. Joint Conference on Neural Networks*, volume 3, pages 21–26, June 1990.
18. D. Nguyen-Tuong, M. Seeger, and J. Peters. Real-time local gp model learning. In *From Motor to Interaction Learning in Robots*. Springer, 2009.

19. J.H. Chung S. Lu and S.A. Velinsky. Human-robot collision detection and identification based on wrist and base force/torque sensors. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3796–3801, April 2005.
20. L. Sciavicco and B. Siciliano. *Modeling and control of robot manipulators*. MacGraw-Hill, 1996.
21. M. Shinya and K. Kazuhiro. Collision detection system for manipulator based on adaptive impedance control law. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1080–1085, 2003.
22. A.J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.
23. J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific Publishing Co. Pte Ltd., Singapore, 2002.
24. J. Swevers, C. Ganseman, D.B. Tukel, J. De Schutter, and H. Van Brussel. Optimal robot excitation and identification. *IEEE Trans. on Robotics and Automation*, 3(5):730–740, 1997.
25. J. Ting, M. Mistry, J. Peters, S. Schaal, and J. Nakanishi. A bayesian approach to nonlinear parameter identification for rigid body dynamics. *Robotics: Science and Systems (RSS)*, 2006.