MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1567
C.B.C.L. Memo No. 133

November, 1995

Learning Fine Motion by Markov Mixtures of Experts

Marina Meilă **Michael I. Jordan**

This publication can be retrieved by anonymous ftp to publications.ai.mit.edu.

Abstract

Compliant control is a standard method for performing fine manipulation tasks, like grasping and assembly, but it requires estimation of the state of contact between the robot arm and the objects involved. Here we present a method to learn a model of the movement from measured data. The method requires little or no prior knowledge and the resulting model explicitly estimates the state of contact . The current state of contact is viewed as the hidden state variable of a discrete HMM. The control dependent transition probabilities between states are modeled as parametrized functions of the measurement. We show that their parameters can be estimated from measurements concurrently with the estimation of the parameters of the movement in each state of contact . The learning algorithm is a variant of the EM procedure. The E step is computed exactly; solving the M step exactly would require solving a set of coupled nonlinear algebraic equations in the parameters. Instead, gradient ascent is used to produce an increase in likelihood.

This report describes research done at the Dept. of Electrical Engineering and Computer Science, the Dept. of Brain and Cognitive Sciences, the Center for Biological and Computational Learning and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Dept. of Defense and by the Office of Naval Research. Michael I. Jordan is a NSF Presidential Young Investigator. The authors can be reached at M.I.T., Dept. of Brain and Cognitive Sciences, 79 Amherst St., Cambridge MA 02139, USA. E-mail: mmp@psyche.mit.edu, jordan@psyche.mit.edu

1 Introduction

For a large class of robotics tasks, such as assembly tasks or manipulation of relatively light-weight objects, under appropriate damping of the manipulator the dynamics of the objects can be neglected. For these tasks the main difficulty is in having the robot achieve its goal despite uncertainty in its position relative to the surrounding objects. Uncertainty is due to inaccurate knowledge of the geometric shapes and positions of the objects and of their physical properties (surface friction coefficients), or to positioning errors in the manipulator. The standard solution to this problem is *controlled compliance* [9]. Under compliant motion, the task is performed in stages. In each stage the robot arm maintains contact with a selected surface or feature of the environment. The stage ends when contact with the feature corresponding to the next stage is made. The goal itself is usually defined as satisfaction of a set of contact constraints rather than fixed position and force references.

Decomposing the given task into subtasks and specifying each goal or subgoal in terms of contact constraints has proven to be a particularly fertile idea, from which a fair number of approaches have evolved. But each of them have to face and solve the problem of estimating the state of contact (i.e. checking if the contact with the correct surface is achieved), a direct consequence of dealing with noisy measurements. The earliest approaches such as *guarded moves* [2] and *compliance control* [9] are problem dependent and deal only indirectly with uncertainty . *Preimage fine motion planning* [8] directly incorporates uncertainty and provides a formal problem statement and approach, but becomes computationally intractable in the case of noisy measurements. Attempting to overcome this last difficulty are the proposals of [10] *local control around a nominal path (LCNP)* and [5] *feature based programming.* The latter treats the uncertanty in position and velocity in a probabilistic framework. It associates to each state of contact the coresponding *movement model*; that is: a relationship between positions and nominal [1] and actual velocities that holds over a domain of the position-nominal velocity space. Then it constructs an observer which uses a sequence of recent measurements in order to decide which is the current movement model. All the above approaches assume prior geometrical and physical knowledge of the environment.

In this paper we present a method to *learn* a model of the environment which will serve to estimate the state of contact and to predict future positions from noisy measurements. The current movement model is viewed as the hidden state variable of a discrete HMM. The control dependent transition probabilities between states are explicitly modeled as parametrized functions of the measurement. We call this model *Markov Mixture of Experts* (MME) and show how its parameters can be estimated. The rest of the paper is organized as follows: Section 2 defines the problem, in section 3 the learning algorithm is derived, section 4 presents a simulated example and section 5 discusses other aspects relevant to the implementation and directions for further work.

[1]The nominal velocity is the velocity of the arm under the same actuator force if it was moving in free space. It is proportional in size and identical in orientation to the force vector exerted by the actuators (the *active* force). When the object is subject to reactive forces from the surfaces it is in contact with, the actual velocity does not coincide with the nominal velocity.

2 Reachability Graphs and Markov Mixtures of Experts

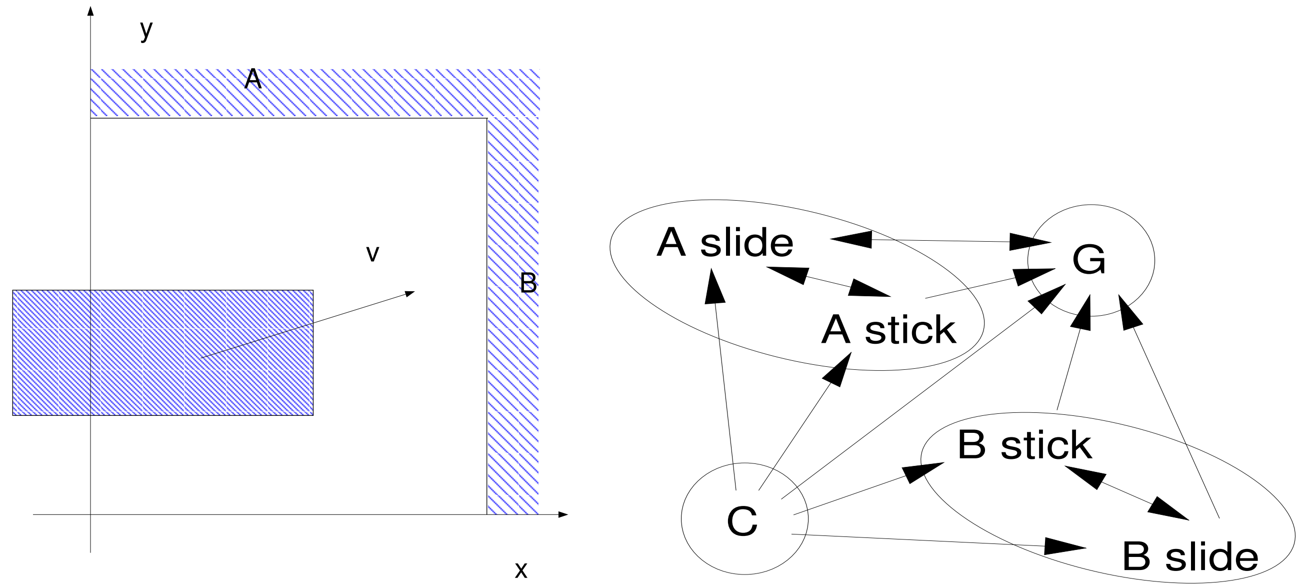
For any assembly of objects, the space of all the relative degrees of freedom of the objects in the assembly is called the *configuration space* (C-space). Every possible configuration of the assembly is represented by a unique point in the C-space and movement in the real space maps into continuous trajectories in the C-space. The sets of points corresponding to each state of contact create a partition over the C-space. Because trajectories are continuous, a point can move from a state of contact only to one of the neighboring states of contact . This can be depicted by a directed graph with vertices representing states of contact and arcs for the possible transitions between them, called the *reachability graph.* If no constraints on the velocities are imposed, then in the reachability graph each state of contact is connected to all its neighbours. But if the range of velocities restricted, the connectivity of the graph decreases and the connections are generally nonsymmetric. Figure 1 shows an example of a C-space and its reachability graph for velocities with only positive componenets. On the other hand, longer movement durations between observations could allow more than one state transition to occur, leading to an effective increase in the number of neighbours of a state.

Ideally, in the absence of noise, the states of contact and every transition through the graph can be perfectly observed. To deal with the uncertainty in the measurements, we will attach probabilities to the arcs of the graph in the following way: Let us denote by Q_i the set of configurations corresponding to state of contact i and let the movement of a point x with uniform nominal velocity v for a time ΔT be given by $x(t+\Delta T) = f^*(x, v, \Delta T)$; both x and v are vectors of same dimension as the C-space. Now, let x', v' be the noisy measurements of the true values x, v, $x \in Q_j$ and $P[x, v|x', v', j]$ the posterior distribution of x, v given the measurements and the state of contact . Then, the probability of transition to a state i from a given state j in time ΔT can be expressed as:

$$P[i|x', v', j] \;=\; \int_{\{x,v|x\in Q_j, f^*(x,v,\Delta T)\in Q_i\}} P[x, v|x', v', j]\,dx\,dv \;=\; a_{ij}(x', v') \tag{1}$$

We define a transition probability matrix $A = [a_{ji}]_{i,j=1}^{m}$ and assume a measurement noise $p[x'|q = i, x \in Q_i]$. This allows us to construct a Hidden Markov Model (HMM) with output x having a continous emission probability distribution p and where the state of contact plays the role of a hidden state variable. Our main goal is to estimate this model from observed data.

To give a general statement of the problem we will assume that all position, velocity and force measurements



(a) (b)

Figure 1: Restricted movement in a 2D space (a) and the corresponding reachability graph (b). The nodes represent movement models: C is the free space, A and B are surfaces with static and dynamic friction, G represents jamming in the corner. The velocity v has positive components.

are represented by the input vector u; an output vector y of dimensionality n_y contains the future position (which our model will learn to predict). Observations are made at integer multiples of time ΔT, indexed by $t = 0, 1, ..T$. If ΔT is a constant sampling time the dependency of the transition probability on ΔT can be ignored. For the purpose of the parameter estimation, the possible dependence between $y(t)$ and $u(t+1)$ will also be ignored, but it should be considered when the trained model is used for prediction.

Throughout the following section we will also make the assumption that the input-output dependence is described by a gaussian conditional density $p(y(t)|q(t) = k)$ with mean $f(u(t), \theta_k)$ and variance $\Sigma = \sigma^2 I$. This is equivalent to assuming that given the state of contact all noise is additive gaussian output noise, which is obviously an approximation. But this approximation will allow us to derive certain quantities in closed form.

The function $f(u, \theta_k)$ is the movement model associated with state of contact k (with θ_k its parameter vector) and q is the selector variable which takes on values from 1 to m. Sometimes we will find it useful to partition the domain of a movement model into subdomains and to represent it by a different function (i.e a different set of parameters θ_k) on each of the subdomains; when $f(u, \theta_k)$ will bear physical significance, the name *movement model* will be to them, but in general each $f(u, \theta_k)$ will be refered to as a *module*.

The evolution of q is controlled by a Markov chain which depends on u and of a set of parameters W_j:

$$a_{ij}(u(t), W_j) = Pr[q(t+1) = i|q(t) = j, u(t)] \quad t = 0, 1, \ldots$$

with

$$\sum_i a_{ij}(u, W_j) = 1 \quad j = 1, \ldots, m \;\; \forall u, W_j. \qquad (2)$$

Figure 2 depicts this architecture. It can be easily seen that this model generalizes the mixture of experts architecture [7], to which it reduces in the case where a_{ij} are independent of j (the columns of A are all equal). It becomes the model of [1] when A and f are neural networks. A model where the Markov chain transitions are implemented by a recurrent network was proposed in [3].

3 An EM Algorithm for MME

To estimate the values of the unknown parameters σ^2, W_k, θ_k, $k = 1, \ldots, m$ given the sequence of observations [2] $u_{\overline{0,T}}$, $y_{\overline{0,T}}$, $T > 0$ and a prior probability of the initial state

$$\pi_j(0) = Pr[q(0) = j].$$

the *Expectation Maximization* (EM) [4] algorithm will be used.

The EM algorithm is an iterative procedure which converges asymptotically to a local maximum of the likelihood function. It requires the introduction of unobserved variables, which, in our case will be the hidden state variables $\{q(t)\}_{t=0}^{T}$. Then it attempts to maximize a *complete* likelihood function l_c which depends on the extended set of variables.

This process is iterative, each iteration comprising two steps. The first step (Expectation or E) finds the distribution of the unobserved variables given the observed variables and the current estimates of the parameters; then it computes the expectation of l_c w.r.t. this distribution.

[2] $s_{\overline{t_1,t_2}} = \{ s(t_1), s(t_1+1), \ldots, s(t_2)\}$ denotes the sequence of values of the variable s over the time interval $t_1, \ldots, t_2$.

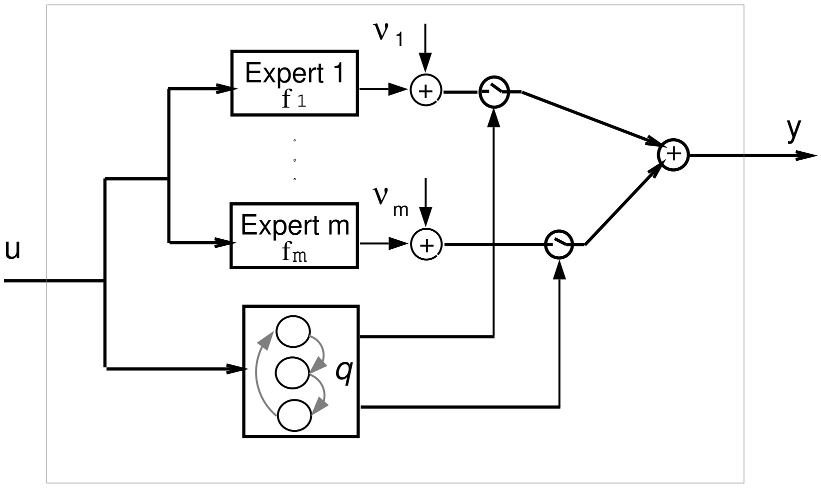


Figure 2: The Markov Mixture of Experts architecture

The Maximization (M) step reassigns to the unknown parameters the values which maximize the expected complete likelihood.

For our problem the complete likelihood is:

$$\begin{aligned} & l_c(y_{\overline{0,T}},\, q_{\overline{0,T}} \,|\, u_{\overline{0,T}},\, W,\, \theta,\, \pi(0)) \; = \\ & = \; \ln\, p\left(y_{\overline{0,T}},\, q_{\overline{0,T}} \,|\, u_{\overline{0,T}},\, W,\, \theta,\, \pi(0)\right) \qquad (3) \\ & = \; \ln \left\{ Pr[q_{\overline{0,T}} \,|\, u_{\overline{0,T}},\, W,\, \theta,\, \pi(0)] \cdot \prod_{t=0}^{T} \epsilon_{q(t)}(t) \right\} \end{aligned}$$

where

$$\epsilon_k(t) \; = \; \frac{1}{(2\pi\sigma^2)^{n_y/2}} \exp\left[-\frac{\|y(t) - f(u(t), \theta_k)\|^2}{2\sigma^2} \right] \qquad (4)$$

and $\theta = \{\theta_1,\, \theta_m\}$. The first factor does not depend on θ and u and can be written as

$$\begin{aligned} Pr[q_{\overline{0,T}} \,|\, u_{\overline{0,T}},\, W,\, \theta,\, \pi(0)] \; & = \qquad (5) \\ & = \; Pr[q_{\overline{0,T}} \,|\, u_{\overline{0,T}},\, W,\, \pi(0)] \\ & = \; \pi(0) \cdot \prod_{t=0}^{T} a_{q(t+1),q(t)} \end{aligned}$$

Let us now introduce the notations

$$\begin{aligned} z^i_{q(t)} \; & = \; \begin{cases} 1 & q(t) = i \\ 0 & \text{otherwise} \end{cases} \\ \gamma_i(t) \; & = \; Pr[q(t) = i \mid u_{\overline{0,T}},\, y_{\overline{0,T}},\, W,\, \theta,\, \sigma^2,\, \pi(0)] \quad (6) \\ \xi_{ij}(t) \; & = \; Pr[q(t) = j,\, q(t+1) = i \mid u_{\overline{0,T}}, \\ & \quad\; y_{\overline{0,T}},\, W,\, \theta,\, \sigma^2,\, \pi(0)] \\ & \quad\; \text{for} \quad t = 0, \ldots, T, \;\; i,j = 1, \ldots, m. \end{aligned}$$

The last two quantities are well-known in the HMM literature and can be computed efficiently by means of a procedure which parallels the *forward-backward* algorithm [11].

Thus, the complete likelihood l_c becomes

$$\begin{aligned} & l_c(y_{\overline{0,T}},\, q_{\overline{0,T}} \,|\, u_{\overline{0,T}},\, W,\, \theta,\, \pi(0)) \; = \qquad (7) \\ & = \ln \pi_{q(0)} + \sum_{t=0}^{T-1} \ln a_{q(t+1),q(t)}(u(t), W_{q(t)}) + \sum_{t=0}^{T} \ln \epsilon_{q(t)}(t) \\ & = \sum_{k=1}^{m} z^k_{q(0)} \ln \pi_k(0) + \sum_{t=0}^{T-1} \sum_{i,j=1}^{m} z^j_{q(t)} z^i_{q(t+1)} \ln a_{ij}(u(t), W_j) \\ & - \sum_{t=0}^{T} \sum_{k=1}^{m} z^k_{q(t)} \left[\frac{\|y(t) - f(u(t), \theta_k)\|^2}{2\sigma^2} + \frac{1}{2} \ln(2\pi\sigma_k^2)^{n_y} \right] \end{aligned}$$

Since this is a linear function of the unobserved variables represented by $z^k_{q(t)}$, taking its expectation is straightforward. Moreover, by the definitions (6), we have

$$E[z^i_{q(t)} \,|\, y_{\overline{0,T}},\, u_{\overline{0,T}},\, W,\, \theta,\, \pi(0)] \; = \; \gamma_i(t) \qquad (8)$$

$$E[z^j_{q(t)}\, z^i_{q(t+1)} \,|\, y_{\overline{0,T}},\, u_{\overline{0,T}},\, W,\, \theta,\, \pi(0)] \; = \; \xi_{ij}(t) \qquad (9)$$

for $i, j = 1, \ldots, m$.

In the M step the new estimates of the parameters are found by maximizing the *average complete log-likelihood* J, which in our case has the form

$$\begin{aligned} J(\theta, \sigma^2, W) \; & = \\ & = \; E[l_c \,|\, y_{\overline{0,T}},\, u_{\overline{0,T}},\, W,\, \theta,\, \sigma^2,\, \pi(0)] \qquad (10) \\ & = \; -\frac{1}{2\sigma^2} \sum_{t=0}^{T} \sum_{k=1}^{m} \gamma_k(t) \|y(t) - f(u(t), \theta_k)\|^2 \\ & \quad + \sum_{t=0}^{T-1} \sum_{i,j=1}^{m} \xi_{ij}(t) \ln a_{ij}(u(t), W_j) \qquad (11) \\ & \quad - \frac{T+1}{2} n_y \ln(\sigma^2) \, + \, \mathrm{C} \end{aligned}$$

where C is a constant. Since each parameter appears in only one term of J the maximization is equivalent to:

$$\theta_k^{new} \; = \; \operatorname*{argmin}_{\theta} \sum_{t=0}^{T} \gamma_k(t) \, \|y(t) - f(u(t), \theta_k)\|^2 \qquad (12)$$

$$W_j^{new} = \underset{W}{\operatorname{argmax}} \sum_{t=0}^{T-1} \sum_i \xi_{ij}(t) \ln a_{ij}(u(t), W_j) \quad (13)$$

$$\sigma^{2\,new} = \frac{\sum_{t=0}^{T} \sum_{k=0}^{m} \gamma_k(t) \,||y(t) - f(u(t), \theta_k)||^2}{n_y(T+1)} \quad (14)$$

By solving (12)-(14) a new set of values for the parameters is found and the current step of the EM procedure is completed.

The difficulty of (12) and (13) strongly depends on the form of the movement models f and of the transition matrix A. The complexity of the movement models is determined by the geometrical shape of the objects' surfaces. For planar surfaces and no rotational degrees of freedom f is linear in θ_k. Then, (12) becomes a weighted least squares problem which can be solved in closed form.

The functions in A depend both on the movement and on the noise models. Because the noise is propagated through non-linearities to the output, a closed form as in (1) may be difficult to obtain. Moreover, a correct noise model for each of the possible uncertainties is rarely available [5]. A common practical approach is to trade accuracy for computability and to parametrize A in a form which is easy to update but devoid of physical meaning. In all the cases where maximization cannot be performed exactly, one can resort to *Generalized EM* by merely *increasing* J. In particular, gradient ascent in parameter space is a technique which can replace maximization. This modification will not affect the overall convergence of the EM iteration but can significantly reduce its speed.

Because EM only finds *local* maxima of the likelihood, the initialization is important. If $f(u, \theta_k)$ correspond to physical movement models, good initial estimates for their parameters may be available. The same can apply to those components of W which bear physical significance. A complementary approach is to reduce the number of parameters by explicitly setting the probabilities of the impossible transitions to 0.

4 Simulation Results

4.1 Problem and implementation

The problem is to learn a predictive model for the movement of a point in the 2-dimensional space shown in figure 1, a. The inputs were 4-dimensional vectors of positions (x, y) and nominal velocities (v_x, v_y) and the output was the predicted position (x_{out}, y_{out}). The coordinate range was [0, 10] and the admissible velocities were confined to the upper right quadrant ($v_x, v_y \geq V_{min} > 0$) and had magnitudes (after multiplication by ΔT) between 0 and 4. The restriction in direction guaranteed that the trajectories remained in the coordinate domain. It also reflected in the topology of the reachability graph, which is not complete (has no transition to the free space from another state, for example). The magnitude range was chosen so that all the transitions in the graph have a non-negligible chance of being observed. A detailed description of the physical model is given in Appendix A.

We implemented this model by a MME. The state $q(t)$ represents the expert used up to time t and $a_{ij}(t)$ the probability of using expert i between t and $t+1$ given that $q(t) = j$.

Implementation of the experts f: The experts $f(u, \theta_k)$ were chosen to be linear in the parameters, corresponding to the piecewise linearity of the true model (see Appendix A). Linearity is achieved by introducing the additional input variables v_x/v_y, v_y/v_x, $x\,v_y/v_x$, $y\,v_x/v_y$, each of them having a new parameter as coefficient. But each additional variable affects only one of the 6 experts (see Appendix A). Therefore, by including the additional variables only in the models that depend on them, we have an insignificant increase in the number of parameters. To avoid infinite input values the the values of v_x, v_y were lower-bounded by the small constant $V_{min} = 1/50$.

Implementation of the transition probability matrix: To implement the transition matrix A we used a bank of *gating networks*, one for each column of A. Examining figure 1 it is easy to see that there exist experts that share the same final state of contact (for instance, **A stick** and **A slide** both represent movements whose final position is on surface **A**). Since transition probabilities depend only on the final position the columns of the matrix A corresponding to these experts are equal. This brings the number of distinct gating networks to 4.

The boundaries of the decision regions are curved surfaces, so that to implement each of them we used a 2 layer perceptron with softmax output, as presented in figure 3. The number of hidden units in each gating net was chosen considering the geometry of the decision regions to be learned.

4.2 Training and testing criteria

The training set consists of $N = 5000$ data points, in sequences of length $T \leq 6$, all starting in the free space. The starting positon of the sequence and the nominal velocities at each step were picked randomly. It was found that for effective learning it is necessary that the state frequencies in the training set are roughly equal. The distribution over velocities and the sequence length T were chosen so as to meet this requirement. The (x, y) values obtained by simulation were corrupted with gaussian additive noise with standard deviation σ.

In the M step, the parameters of the gating networks were updated by gradient ascent, with a fixed number of epochs for each M step. Appendix C shows how the gradient was computed. We used weighted least squares estimation for the movement models parameters. To ensure that models and gates are correctly coupled, initial values for θ are chosen around the true values. In the present case, this is not an unrealistic assumption. W was initialized with small random values. Because of the long time to convergence, only a small number of runs have been performed. The observed variance of the results over test sets was extremely small, so that the values presented here can be considered as typical.

Three criteria were used to measure the performance of the learning algorithm: experts' parameters deviation from the true values, square root of prediction MSE and hidden state misclassificaton. Because training was performed on a distribution that is not expected to appear

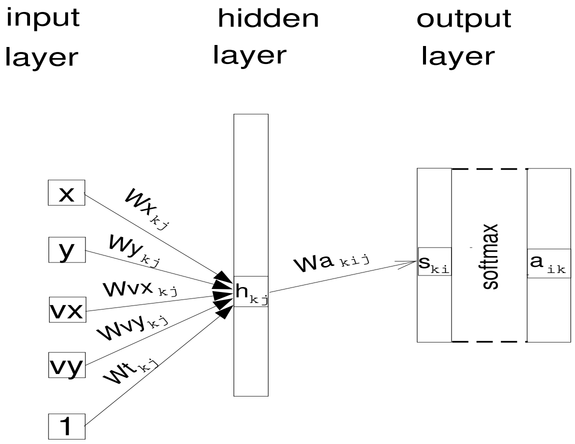


Figure 3: Gating network computing the transition probabilities a_{ik} for state of contact k.

in practice, all the models were tested on both the traininng distribution and on a distribution which is uniform over (v_x, v_y) (and therefore highly non-uniform over the states of contact), subsequently called the "uniform V distribution".

Performance comparisons were made with a ME architecture having identical experts but only one gating net. For the number of hidden units in it, various values have been tried; the results presented here represent the performance of the best model obtained. The performance of a k-nearest neighbor (k-NN) model is also shown.

4.3 Results

Learning curves. Figure 4 presents the learning curves for two MME and two ME models. The horizontal axes show the number of EM epochs. Since the number of backpropagation epochs for each M step is 2000, the number of training epochs for the ME models was scaled down accordingly. The ME models converge faster than the MME models. The differrence may be due to the choice of a fixed number of epochs in the M step; the MME model also contains more parameters in the gating nets (220 vs ME's 180 parameters). The experts contain 64 parameters.

The only model which achieves 0 misclassification error is the MME trained without noise. For training in noise, the final prediction error on the training set is the same for both architectures, but the results on test sets will show a difference favouring the MME model.

Comparison with k-NN. The k-NN method performed much worse than the other two algorithms. The following table presents a summary of the best results obtained.

Test set performance. Figure 5 presents the test set performance of MME and ME models trained with and without noise. Each test set contained $\geq$ 50,000 datapoints. The input noise added to the (x, y) values in the test sets was gaussian with variance between 0 and $(0.4)^2$. It can be seen that the ME models perform similarly w.r.t. both prediction error and state classification. The two MME models perform better than the ME models, with a significant differrence between the model trained with noise and the model trained with ideal data. The results are consistent over distributions, noise levels and performance criteria.

Table 1: **Performance ($\mathbf{MSE^{1/2}}$) for the k-NN method.** Memorized (training) set: size=11,000; noise=σ_m. Test set: size=11,000; noise=σ_t.

σ_m	k	$\sigma_t = 0$	$\sigma_t = 0.2$
0	100	0.333762	0.425333
0.2	200	0.341495	0.455458

Parameter error. In the present problem the true parameters for the experts can be computed exactly. The values of the MSE of the learned parameters in various models are shown in the following table (where σ represents the noise level in the training set):

Model	σ noise	$\mathbf{MSE}_\theta^{1/2}$
MME	0	0
MME	0.05	0.0474
MME	0.2	0.0817
ME	0	0.1889
ME	0.2	0.1889

Prediction along a trajectory. To illustrate the behaviour of the algorithm in closed loop, figure 6 presents a sample trajectory and the predictions of the MME model in open and closed loop mode. In the latter, the predicted and measured positions were averaged (with ratios 1/2) to provide the models' (x, y) input for the next time step.

The simulations show that, although input noise is not explicitly taken into account by the model, the MME architecture is tolerant to it and is able to achieve both learning and good prediction performance in noisy conditions.

The comparison with k-NN confirms that the problem is not simple and that well tailored algorithms are required to solve it. MME outperforms ME in all situations, with a small additional computational effort dur-

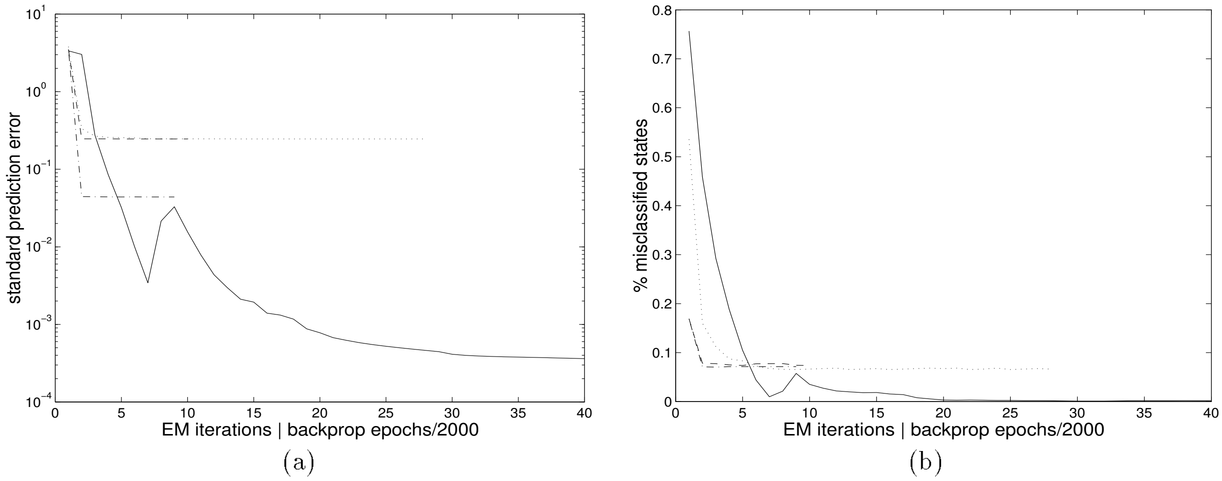


Figure 4: Prediction standard error ($\mathrm{MSE}^{1/2}$) (a) and percentage of wrong expert choices (misclassified states) (b) on the training set during learning for the MME and ME models. The abscissa was scaled according to the number of backpropagation epochs for the two models. — MME, no noise; $\cdots$ MME, noise 0.2; – · – ME, no noise; – – ME, noise 0.2.

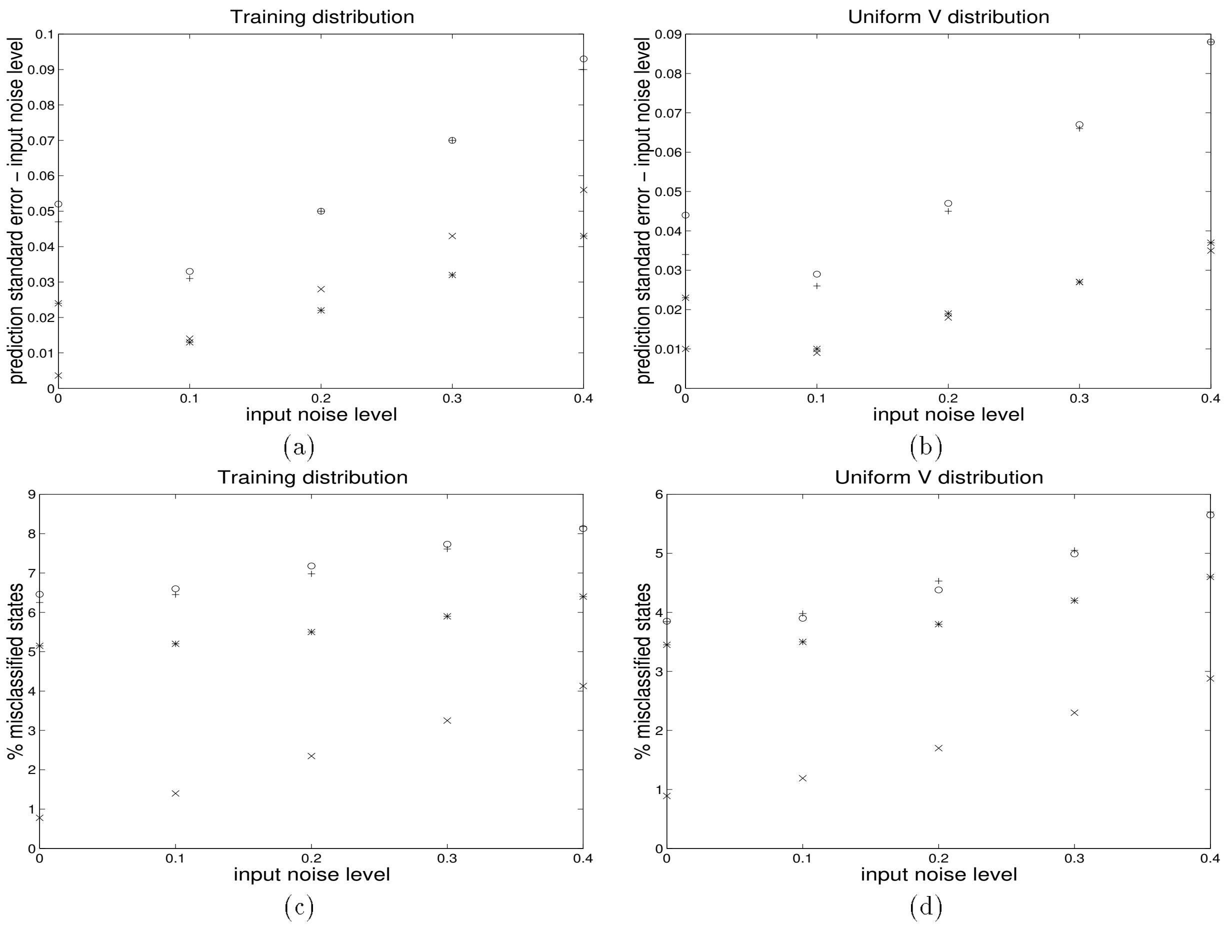


Figure 5: Test set performance of the MME and ME models for various levels of the input noise σ_{in} on two different distribuitons. (a) and (b): (Prediction MSE)$^{1/2} - \sigma_{in}$; (c) and (d): the percentage of misclassified states. $\times$ – MME, no noise; * – MME noise 0.2; $\circ$ – ME, no noise; $+$ – ME, noise 0.2.

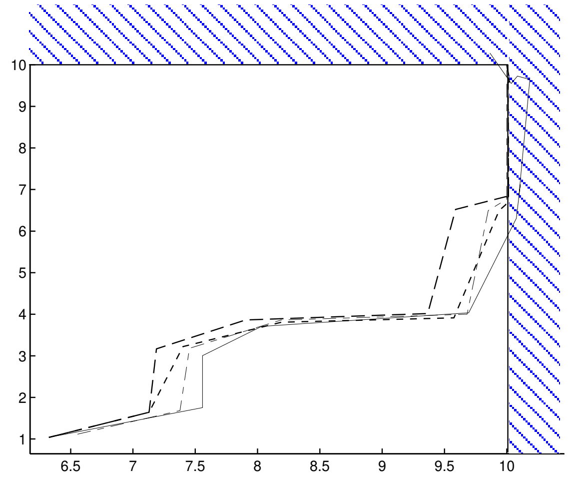


Figure 6: Sample trajectory in the (x, y) space, together with the open and closed loop predictions of the MME model. The noise level is 0.2.

ing training.This demonstrates the advantage of taking into account the time-dependencies in the data.

5 Discussion

The learning algorithm and the architecture presented here allow us to model a wide range of dynamic systems. Theoretically any finite state discrete time system can be represented in the form of a MME, and continuous state spaces can be often conveniently discretized. But the aim of using a MME is to take advantage of the natural modularity of the process we are trying to model and to divide it into subprocesses of significantly lower complexity.

In the special case when the MME reduces to a mixture of experts, the probability of a state depends only on the input u, and it is easy to see (also cf. [7]) that the gating network partitions the input space, assigning to each expert a subset of it. As the domains of the experts are generally contiguous, the mixture of experts can be viewed as a combination of local models, each being accurate in a confined region of the input space.

From the same point of view, the MME divides the data into *regimes* and constructs locally accurate models of dynamic processes corresponding to each of them. The partition performed is twofold: the dynamic component is confined to the level of the Markov chain and, at the same level, the set of input-output pairs is partitioned into subsets which are assigned to the static experts. The former is ensured by the structure of the model, the latter has to be learned during the E step of the EM algorithm.

This suggests that a critical condition for accurate learning is the correct assignment of the data points to the experts responsible for them. This is basically a clustering problem. Therefore, although the learner is provided with "input-output" pairs, it is esssentially performing an alternation of supervised and unsupervised learning tasks.

5.1 Computational aspects

Local maxima. The EM algorithm is guaranteed to converge to a local maximum of the likelihood. This may not be sufficient when the value of the likelihood at the local maximum is much smaller than the value at the global maximum. Another issue is that in some cases we may be in search of parameters with physical meaning. In these cases, we need to find the (local) maximum that is closest to the parameters of the data generating process.

The first issue can be addressed by performing several runs of the learning algorithm, each starting from a different initial point in the parameter space, and by chosing at the end the most likely solution. To address the second issue, besides validating the values of the parameters after convergence, prior knowledge about the process can be used to find a good initial estimate.

Influence of f and a_{ij}. We have imposed no conditions on the form of f and a_{ij} so far. They influence the success of learning in several ways:

1. they determine the difficulty of the Maximizaton step, as discussed in 3.

2. f influences the difficulty of the clustering problem; a linear f reduces it to gaussian clustering, whereas if the class of allowed I/O functions becomes too rich or if the mapping $\theta \to f(\cdot, \theta)$ is not one-to-one the assignment problem can become ambiguous.

3. the form of f and a_{ij} determines the number of parameters[3] and thereby the number of data points necessary for accurate learning.

4. by 2. and 3. the shape of the likelihood function is influenced and subsequently the number and distribution of the local maxima

Number of states. So far we have assumed that m, the number of experts, is known. This is not always the case in practice. Simulations suggest that starting with a large enough m could be a satisfactory strategy. Sometimes the superfluous clusters are automati-

[3] More precisely, the *complexity* of the model.

cally "voided" of data points in the process of learning. This can be attributed to the fact that, for any fixed model structure, maximizing likelihood is equivalent to minimizing the description length of the data [12].

On the other hand, the number of parameters in A increases proportionally to m^2, increasing both the computational burden and the *data complexity*. The clustering becomes also harder for a large number of alternatives. Therefore it is important to know m or to have a good upper bound on it.

Data complexity. The data complexity of a model is, loosely speaking, the number of data points required to attain a certain level of the prediction error. In the case where we can talk about true parameters, the prediction error is reflected by the accuracy of the parameters. For a more rigorous and detailed discussion of data complexity consult [6] and its references. For the purposes of this paper it is sufficient to state that for a given class of models, the data complexity increases with the number of parameters. As a consequence, an increase in m will induce a quadratic increase in the number of parameters and in the training set as well.

Simulations have shown that training this architecture requires large amounts of data. Therefore it is necessary to use prior knowledge to reduce the number of parameters. This and other uses of knowledge outside the data set for constraining the model are the topics of the next subsection.

5.2 Incorporating prior knowledge

Prior knowledge is extremely valuable in any practical problem. Although the ML paradigm doesn't provide a systematic way of incorporating it, prior knowledge can and should be used at several levels of the model.

First, it will help in finding the appropriate model structure. An important structural parameter is the number of states of the Markov chain. For instance, knowing the number of the movement models in the fine motion task has allowed us to pair each expert with a movement model, thus having simple linear experts whose parameters have a physical meaning. For systems that are not intrinsically discrete-time, prior knowledge should guide the choice for an appropriate discretization step. A too long discretization step may miss transitions between states leading thus to poor modelling, whereas a too short one will cause unnecesarily low transition probabilities between different states.

Closely related to the choice of the number of states is the choice of the functional form of the experts. This has been shown in the modeling of the robot arm motion. Prior knowledge should help find the class of functions f with the suitable complexity. This isssue, as well as the issue of the best representation for the inputs and outputs, arise in any modeling problem.

A similar selection concerns the form of the functions in A. In view of condition (2) it is often reasonable to group the transition probabilities into gating networks as we have done in section 4 thus having a vector of parameters W_j for each set of functions $\{a_{ij} | i = 1, \ldots, m\}$. If it is known a priori that certain transitions cannot occur, then their probabilities can be set to 0, thereby reducing both the complexity of the gating network and speeding up the learning process by reducing uncertainty in the state estimation. For instance, in the fine motion problem, when the state-space grows by adding new objects to the environment, the number of possible transitions from a state of contact (also called the *branching factor*) will be limited by the number of the states of contact which are within a certain distance and can be reached by a simple motion, which will grow much slowlier than m. Therefore, the number of nonzero elements in A will be approximately proportional to m. This is generally the case when transitions are local, providing another reason why locality is useful.

Finally, when the experts have physical meaning, prior knowledge should give "good" initial estimates for the experts parameters. This is important not only to decrease the learning time, but also to avoid convergence to spurious maxima of the likelihood.

Relearning. Certain aspects of relearning have been discussed at the end of the previous section. (By relearning we mean learning a set of parameters, under the assumption that another learning problem has already been solved and that the old model is identical in structure to the new one. Sometimes it is also assumed that the old and new parameter values are "close".) Since EM is a batch learning algorithm, it cannot automatically adapt to changes in the parameters of the data generating process. But knowledge exterior to the data set (that was called prior knowledge in the above paragraphs) can still be used to make relearning easier than learning from scratch. This process can be viewed as the reverse of incorporating prior knowledge: we must now (selectively) discard prior knowledge before learning from the new data begins. Obviously, the model structure remains unchanged; the question is whether to keep some of the parameter values unchanged as well. The answer is yes in the case when we know that only some of the experts or gating nets have suffered changes, in other words when the change was local. In this case the learning algorithm can be easily adapted to *locally relearn*, i.e. to reestimate only a subset of the parameters.

6 Conclusion

To conclude, the MME architecture is a generalization of both HMMs and the mixture of experts architecture. From the latter it inherits the abilty to take advantage of the natural decomposability of the processes to model, when this is the case, or to construct complex models by putting together simple local models otherwise. The embedded Markov chain allows it to account for simple dynamics in the data generation process.

The parameters of the MME can be estimated by an iterative algorithm which is a special case of the EM algorithm. As a consequence, the *a posteriori* probabilities of the hidden states are also computed. Multiple aspects pertaining to the implementation of the algorithm have been discussed. The trained model can be used as a state estimator or for prediction as the forward model in a recursive estimation algorithm.

References

[1] Y. Bengio and P. Frasconi. An input output HMM architecture. In J. D. Covan, G. Tesauro, and J. Alspector, editors, *Neural Information Processing Systems - 7*, 1995.

[2] R. Bolles and R. Paul. The use of sensory feedback in a programmable assembly system. Technical report, Stanford U., 1973.

[3] T. W. Cacciatore and S. J. Nowlan. Mixtures of controllers for jump linear and non-linear plants. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Neural Information Processing Systems - 6*, 1994.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39:1–38, 1977.

[5] B. S. Eberman. *A sequential decision approach to sensing manipulation contact features.* PhD thesis, M.I.T., 1995.

[6] Stuart Geman, Elie Bienenstock, and Rene Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[7] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[8] T. Lozano-Perez. Spatial planning: a configuration space approach. *IEEE Transactions on Computers*, 1983.

[9] M. T. Mason. Compliance and force control for computer controlled manipulation. *IEEE Trans. on Systems, Man and Cybernetics*, 1981.

[10] S. Narasimhan. *Task-level strategies for robots.* PhD thesis, M.I.T., 1994.

[11] R. L. Rabiner and B. H. Juang. An introduction to hidden Markov models. *ASSP Magazine*, 3(1):4–16, January 1986.

[12] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

A The exact movement model for the simulated example

Here the equations describing the exact movement model of the point in the C-space shown in fig.1 are given. First, some notation. The position of the point at the current time is denoted by (x, y) and its position after moving for a time ΔT with constant velocity vector (v_x, v_y) by (x^{new}, y^{new}).

The static and dynamic friction coefficients on surfaces A and B are μ_A^s, μ_A^d, μ_B^s, μ_B^d; we will consider that $x \in [0, LX]$, $y \in [0, LY]$. With the movement model notation introduced in figure 1 the movement equations are:

C:

$$\begin{aligned} x^{new} &= x + v_x \Delta T \\ y^{new} &= y + v_y \Delta T \end{aligned} \tag{15}$$

A slide:

$$\begin{aligned} x^{new} &= x + \mathrm{sgn}(v_x)\mathrm{sgn}(v_y)\mu_A^d(YL - y) \\ &\quad + \Delta T(v_x - \mu_A^d v_y) \\ y^{new} &= YL \end{aligned} \tag{16}$$

A stick:

$$x^{new} = x + v_x \frac{YL - y}{v_y} \tag{17}$$

$$y^{new} = YL \tag{18}$$

B slide:

$$\begin{aligned} x^{new} &= XL \\ y^{new} &= y + \mathrm{sgn}(v_x)\mathrm{sgn}(v_y)\mu_B^d(XL - x) \\ &\quad + \Delta T(v_y - \mu_B^d v_x) \end{aligned} \tag{19}$$

B stick:

$$x^{new} = XL \tag{20}$$

$$y^{new} = y + v_y \frac{XL - x}{v_x} \tag{21}$$

G:

$$\begin{aligned} x^{new} &= XL \\ y^{new} &= YL \end{aligned} \tag{22}$$

In the above, sgn denotes the sign function

$$\mathrm{sgn}(z) = \begin{cases} 1, & z > 0 \\ -1, & z < 0 \\ 0, & z = 0. \end{cases}$$

Equations (16-21) are slightly more general than needed for the present case. For velocities restricted to $v_x, v_y \geq V_{min} > 0$ the sgn functions are 1 and can be dropped, yielding all equations but (17 - 21) linear.

The former can also be brought to a linear form by introducing the input variables v_x/v_y, v_y/v_x, $(x.v_y)/v_x$, $(y.v_x)/v_y$. The coefficients of the extended set of variables are denoted by $\theta_x^C, \ldots \theta_{v_y}^G$ and represent the parameters of the model to be learned. The coefficients corresponding to v_x/v_y, v_y/v_x, etc. are considered to be 0 and are not subject to learning for the equations which don't contain these variables.

The domain of each movement model is the region in the 4-dimensional space where the given model is valid. This region is easily defined in terms of a set of inequalities which must be satisfied by all its points. For example, the (preimage of) free space C is the set of points for which $0 < x^{new} < XL$, $0 < y^{new} < YL$. If we define the functions:

$$\begin{aligned} d1(x, y, v_x, v_y) &= x + v_x \Delta T \\ d2(x, y, v_x, v_y) &= y + v_y \Delta T \end{aligned} \tag{23}$$

the following is an equivalent definition for the domain of C :

$$\begin{aligned} 0 < \ & d1 < XL \\ 0 < \ & d2 < YL \end{aligned}$$

but now the functions $d1$ and $d2$ can be used in inequalities defining the domains of neighboring movement models. The complete list of functions which define the domain boundaries follows.

$$
\begin{aligned}
d1 &= x + v_x \Delta T \\
d2 &= y + v_y \Delta T \\
d3 &= v_x - \mu_A^s v_y \\
d4 &= v_y - \mu_B^s v_x \\
d5 &= x + \tfrac{YL-y}{v_y} v_x \\
d6 &= y + \tfrac{XL-x}{v_x} v_y \\
d7 &= x + \mathrm{sgn}(v_x)\,\mathrm{sgn}(v_y)\,\mu_A^d (YL - y) \\
&\quad + \Delta T (v_x - \mu_A^d v_y) \\
d8 &= y + \mathrm{sgn}(v_x)\,\mathrm{sgn}(v_y)\,\mu_B^d (XL - x) \\
&\quad + \Delta T (v_y - \mu_B^d v_x)
\end{aligned}
$$

In addition, for all the movement models hold the inequalities:

$$
\begin{aligned}
0 &\le x \le XL \\
0 &\le y \le YL \\
V_{min} &\le v_x \\
V_{min} &\le v_y \\
|v| &\le V_{max}
\end{aligned}
$$

It can be seen that the model boundaries are not linear in the inputs. Even if made linear by augmenting the set of inputs as above, some domains are not convex sets and others, although convex, cannot be represented as the result of a tesselation with softmaxed hyperplanes.

B The sigmoid and softmax functions

The definition of the *sigmoid* function used in the present example is

$$\mathrm{sigmoid}(x) \equiv h(x) = \tanh(x) \equiv \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad (24)$$

where x is a real scalar. The sigmoid function maps the real line into the interval (-1,1). Its derivative w.r.t. the variable x can be expressed as

$$h'(x) = 1 - h^2(x)$$

The *softmax function* is the multivariate analog of the sigmoid. It is defined on $\mathcal{R}^n \to \mathcal{R}^m$ by:

$$\mathrm{softmax}_i(x, W) \equiv g_i(x, W) = \frac{\exp(W_i^T x)}{\sum_j \exp(W_j^T x)}, \quad i = 1, ..m$$

with W, x vectors of the same dimension.) If we make the notation $s_i = W_i^T x$, then the partial derivatives of the softmax function are given by

$$\frac{\partial g_i}{\partial s_j} = g_i(\delta_{ij} - g_j) \ \forall i, j \qquad (25)$$

where δ_{ij} represents Kronecker's symbol.

C Gradient calculations

Here we give the calculation of the derivatives used in the generalized M-step of the EM algorithm.

To obtain the new iterates of the parameters of the gating networks, one has to maximize the function given by (13) w.r.t. to W

$$J_W = \sum_{t=0}^{T-1} \sum_{ij} \xi_{ij}(t) \ln (a_{ij}(u(t), W_j))$$

In the present implementation, the functions a_{ij} are computed by a two layer perceptron with softmax output. The complete calculation is given below, and the notation is explained by figure 3.

$$a_{ik}(u(t), W) \equiv a_{ik}(t) = \mathrm{softmax}_i(h_k, \mathrm{Wa}_k)$$

$$
\begin{aligned}
h_{kj}(t) = {} & \mathrm{sigmoid}[\mathrm{Wx}_{kj} x(t) + \mathrm{Wy}_{kj} y(t) + \mathrm{Wvx}_{kj} v_x(t) \\
& + \mathrm{Wvy}_{kj} v_y(t) + \mathrm{Wt}_{kj}]
\end{aligned}
$$

where $k = 1, .., m$, $i = 1, .., m'_k$, $j = 1, ..., n_k + 1$ and m, m', n_k represent the number of states, of outputs and of hidden units for gate k, respectively. The parameter matrices and vectors $\mathrm{Wx}, \mathrm{Wy}, \mathrm{Wvx}, \mathrm{Wvy}, \mathrm{Wt}$ and Wa have the appropriate dimensions. By $j' \to k$ we denoted the fact that module j' leads to state of contact k. This notation is necessary because there can be more than one module leading to a given state of contact . (For example, modules **A stick** and **A slide** both lead to contact with surface **A**). The last unit in each hidden layer is fixed to 1 to produce the effect of a threshold in s_k.

Expressing the derivatives as in Appendix B and usig the fact that

$$\sum_i \xi_{ij}(t) = \gamma_j(t) \quad \forall t, j$$

we obtain the following formulas for the partial derivatives of J_W:

$$
\begin{aligned}
\frac{\partial J_W}{\partial W a_{kij}} &= \sum_{t,l} \frac{\partial J_W}{\partial a_{lk}(t)} \frac{\partial a_{lk}(t)}{\partial W a_{kij}} \\
&= \sum_{t,l} \frac{\sum_{j' \to k} \xi_{lj'}(t)}{a_{lk}(t)} a_{lk}(t)(\delta_{il} - a_{ik}(t)) h_{kj}(t) \\
&= \sum_t h_{kj}(t) \left[\sum_{j' \to k} \xi_{ij'}(t) - a_{ik}(t) \sum_{j' \to k} \gamma_{j'}(t) \right]
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial J_W}{\partial W x_{kj}} &= \sum_{t,l} \frac{\partial J_W}{\partial a_{lk}(t)} \frac{\partial a_{lk}(t)}{\partial h_{kj}(t)} \frac{\partial h_{kj}(t)}{\partial W x_{kj}} \\
&= \sum_t h'_{kj}(t) x(t) \sum_i W a_{kij} \cdot \\
&\quad \left[\sum_{j' \to k} \xi_{ij'}(t) - a_{ik}(t) \sum_{j' \to k} \gamma_{j'}(t) \right]
\end{aligned}
$$

Notice the bracketed term that is common to the two expressions. Assuming, for simplicity, that $j' = k$, it can be rewritten as

$$\xi_{ik}(t) - a_{ik}(t)\gamma_k(t) = \gamma_k(t) \left[\frac{\xi_{ik}(t)}{\gamma_k(t)} - a_{ik}(t) \right]$$

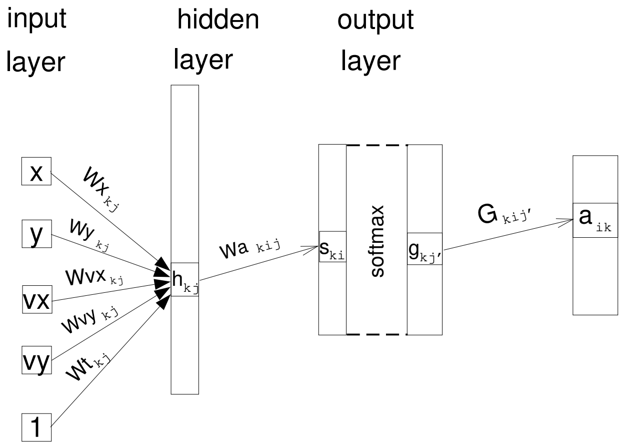


Figure 7: An alternate network for the transition probabilities. $G_{kij'}$ is 1 if $j' \to i$ and 0 otherwise.

outlining its role of weighted output error of the gating net.

The modules f are linear in the parameters, therefore their derivatives computation is straightforward.

Now we will show why the initial implementation is not fit for gradient ascent. Remember that in this representation the states of the Markov chain stood for states of contact and $a_{ik}(t) = P[q(t+1) = i|q(t) = k, u(t)]$. By $g_{kj'}(t)$ we denote the probability of using movement model j' in the time interval $(t, t+1]$ given that the state of contact at time t was k. We have that

$$a_{ik} = \sum_{j' \to i} g_{kj'} = \sum_{j'} G_{kij'} g_{kj'}$$

The structure of the network is given in figure 7. The elements of the matrix G are 1 or 0, indicating whether module j' leads to state of contact i or not.

The function to be maximized has the same form as above, with the indices i, j both running now over the states of contact . Then its gradient w.r.t. $g_{kj'}$ is

$$\frac{\partial J_W}{\partial g_{kj'}} = \frac{\xi_{ik}}{a_{ik}}$$

where i is the state of contact to which movement model j' leads (i.e. $j' \to i$). The above relationship shows that all the gates' outputs corresponding to the same state of contact receive the same error signal and therefore no competition can occur between them.