Kalman Filters and Neural Networks

Valentyn Boreiko

01503954

June 2020

Abstract

In this paper, three key aspects of nonlinear stochastic dynamical systems modelling and neural networks will be presented (including the theoretical properties and derivations):

- Algorithms (based on the ideas of Kalman Filter and Particle Filter) for the optimal state estimation in stochastic dynamical systems, given the known parameters.
- Expectation-Maximization algorithm for learning the parameters of a model in the presence of incomplete data or latent (hidden) variables.
- Universal approximation theorem and Backpropagation algorithm, that will help to introduce neural networks.

As a motivation, both the application of the models to learn the human movement data will be shown, as well as the recent research development in the field. To gain a better grasp of the concepts, some simulations will be presented too.

Table of contents

Chapter 1

Neural networks and deep variational bayesian filtering (DVBF)

1.1 Application

1.1.1 Introduction

Movements of human body have time dependencies, that need to be modeled. We want to consider speed and acceleration at each time step. For this, we would need stochastic dynamical systems. Dynamic movement primitives (DMP), being such stochastic dynamical systems, have been shown to be a powerful method of representing these movements, but they do not generalize well when used in configuration or task space. One can solve this problem by mapping the data in an original space to a lower dimensional latent space. Such a mapping can be done by autoencoders, that are used to find a representation of movement in a latent feature space, in which DMP can optimally generalize.

1.1.2 Data

The CMU Graphics Lab Motion Capture Database is used to train and evaluate the model. That database is created by tracking 41 markers on human subjects during walking, jogging, etc. using a Vicon optical tracking system. The data are preprocessed using Vicon Bodybuilder to render limb joint angles, leading to a 62-dimensional feature vector. Of these, 6 values are almost constant (shoulder and finger movement, having an insignificant variance $< 10^{-26}$) and are ignored. Another set of 6 values represent the origin frame, and can be removed after all data are represented with respect to this frame. The resulting feature vector $\mathbf{y}_t$ is 50-dimensional. Before training the model, the data is normalized in every dimension to zero mean and range in $[-1,1]$. After this we take a sequence of features $\mathbf{y}_{1:T}$ and use it for the model. From now on, we will write $\mathbf{1}:\mathbf{T}$ for sequence.

1.1.3 Goal

We will focus on learning and reconstructing different movements (walking, kicking, taichi and punching were used in the original paper and we will use the data of zig-zag walking, punching, suitcase, swimming) in one model.
To give the intuition, the mapping to the 2D space (the results are from the paper) is shown below.

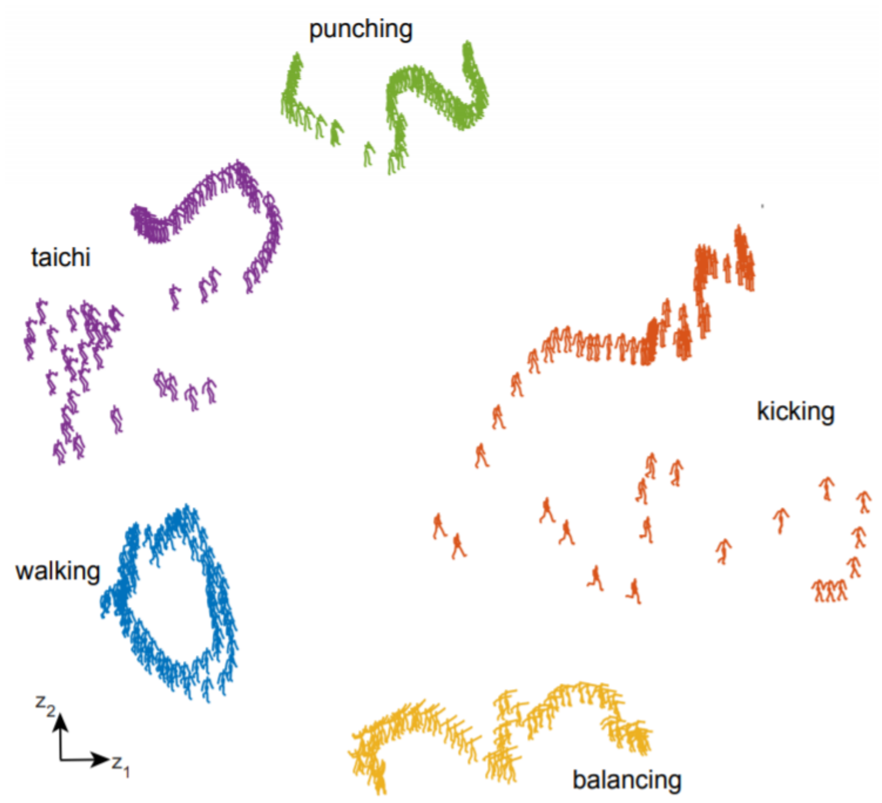

Fig. 1.1 Movement in 2D latent space

The patterns of the various generative movements can be seen. The walking is periodic, while punching is a single line in latent space. Kicking is a large-range movement, so that it has large range in the latent space, while balancing only has relative slight movement, and the range in the latent space is small.

1.2 Stochastic dynamical system

We assume, that we have for any time step t

- *observations* - they are often non-Markovian (in the sense that $\mathbb{P}(\mathbf{y}_n = y_n | \mathbf{y}_{n-1} = y_{n-1}, ..., \mathbf{y}_0 = y_0) \neq \mathbb{P}(\mathbf{y}_n = y_n | \mathbf{y}_{n-1} = y_{n-1})$), $\mathbf{y}_t \in \mathscr{Y} \subset \mathbb{R}^n_y$, and *predicted observations* ${\mathbf{y}^{\mathbf{pred}}}_t \in \mathscr{Y} \subset \mathbb{R}^n_y$,
- *external inputs* $\mathbf{q}_t \in \mathbb{R}^n_x$, which in our case are non-linear *forcing terms* $\mathbf{f}_t \in \mathbb{R}^n_x$,
- *latent variables* $\mathbf{x}_t \in \mathscr{X} \subset \mathbb{R}^n_x$, and *predicted latent variables* $\mathbf{z}_t \in \mathscr{X} \subset \mathbb{R}^{n_x}$,
- and stochastic error terms $\varepsilon_t \in \mathbb{R}^n_x$.

We are interested in learning a generative probabilistic model,

$$\mathbb{P}({\mathbf{y}^{\mathbf{pred}}}_{1:T} | \mathbf{f}_{1:T}) = \int_{\mathscr{X}^{2T}} \mathbb{P}(\varepsilon_{1:T}) \mathbb{P}({\mathbf{y}^{\mathbf{pred}}}_{1:T} | \mathbf{z}_{1:T}, \mathbf{f}_{1:T}) \mathbb{P}(\mathbf{z}_{1:T} | \varepsilon_{1:T}, \mathbf{f}_{1:T}) \, d\varepsilon_{1:T} \, d\mathbf{z}_{1:T}, \quad (1.1)$$

$\mathbb{P}({\mathbf{y}^{\mathbf{pred}}}_{1:T} | \mathbf{z}_{1:T}, \mathbf{f}_{1:T})$ is called the *emission model.* $\mathbb{P}(\mathbf{z}_{1:T} | \varepsilon_{1:T}, \mathbf{f}_{1:T})$ is called the *transition model*

The transformations *during the generation* are described in the picture below:

T.1 given $\mathbf{y}_0, \mathbf{y}_1 \in \mathscr{Y}$ - two consecutive frames, find a model to encode it to $\mathbf{x}_0, \mathbf{x}_1 \in \mathscr{X}$ in a latent space and set $\mathbf{z}_0 := \mathbf{x}_0, \mathbf{z}_1 := \mathbf{x}_1$,

T.2 given $\mathbf{z}_1, \mathbf{z}_2$ and a *goal or target frame* ($\mathbf{z}^{\mathbf{goal}}$ later on), find a model to generate a sequence $\mathbf{z}_{1:T}$ in a latent space,

T.3 given the sequence $\mathbf{z}_{1:T}$, find a model to generate a sequence ${\mathbf{y}^{\mathbf{pred}}}_{1:T}$ of the predicted movement frames.

We assume for the *transition model* the following non-linear discrete-time dynamical system:

$$\begin{aligned} \tau \ddot{\mathbf{z}}_{t+1} &= \alpha(\beta(\mathbf{z}^{\mathbf{goal}} - \mathbf{z}_t) - \dot{\mathbf{z}}_t) + \mathbf{f}_t + \varepsilon_t, \\ \dot{\mathbf{z}}_{t+1} &= \ddot{\mathbf{z}}_{t+1} \Delta t + \dot{\mathbf{z}}_t, \\ \mathbf{z}_{t+1} &= \dot{\mathbf{z}}_{t+1} \Delta t + \mathbf{z}_t, \end{aligned} \quad (1.2)$$

where Δt is a time step, τ – time constant, $\alpha > 0, \beta > 0$ – damping constants, $\mathbf{x}^{\mathbf{goal}}$ is the end state that we want to be at, $\{\mathbf{z}_0 = \mathbf{x}_0, \mathbf{z}_1 = \mathbf{x}_1\}$ is the initial condition and for each time step t we have,

$$\mathbf{f}_t = \frac{\sum_{i=1}^N \Psi_i(t) \mathbf{w}_i}{\sum_{i=1}^N \Psi_i(t)},$$

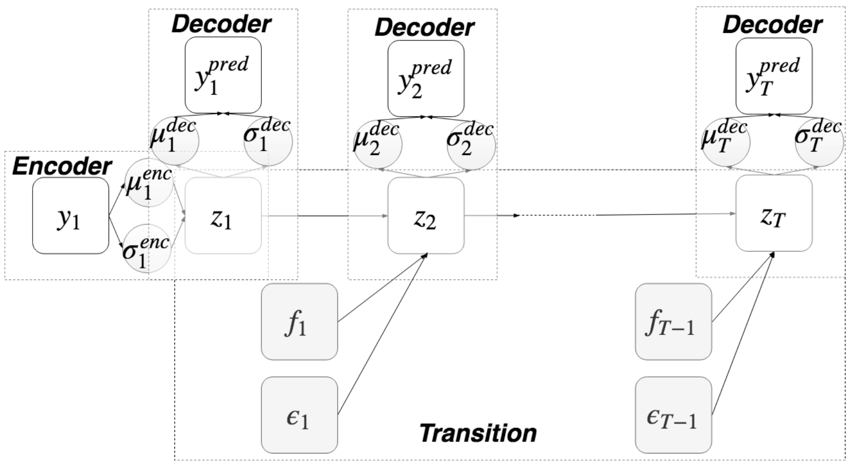

Fig. 1.2 Generation of the movement

where the $\mathbf{w_i}$ are suitable weights and

$$\Psi_i(t) = \exp\big(-\frac{(t-c_i)^2}{2\sigma_i^2}\big). \tag{1.3}$$

Therefore we assume,

$$\mathbb{P}(\mathbf{z}_{1:T}|\varepsilon_{1:T},\mathbf{f}_{1:T}) = \prod_{t=0}^{T-1} \mathbb{P}(\mathbf{z}_{t+1}|\mathbf{z_t},\varepsilon_\mathbf{t},\mathbf{f_t}). \tag{1.4}$$

An example of such a transition model is a *Linear Gaussian Model*,

Example 1.2.1 (Linear Gaussian Model)

$$\begin{aligned} \mathbf{z}_{t+1} &= A\mathbf{z}_t + B\mathbf{q}_t + w_t, \quad w_t \sim \mathcal{N}(0,\Sigma_t^w) \\ \mathbf{y}_t^{pred} &= C\mathbf{z}_t + D\mathbf{q}_t + v_t, \quad v_t \sim \mathcal{N}(0,\Sigma_t^v), \end{aligned} \tag{1.5}$$

where $\mathbf{q}_t$ are the external inputs.
The so-called *Kalman filter* solves (1.5) optimally – it is a *best linear unbiased estimator* of the state $\mathbf{z}_t$ of a linear dynamical system (1.5). In other words,

$$\mathbb{E}(\mathbf{x}_t - \mathbf{x}_t^k)^2 \leq \mathbb{E}(\mathbf{x}_t - \mathbf{x}_t^e)^2, \tag{1.6}$$

for $\mathbf{x}_t^k$ being an estimated state by kalman filter, and $\mathbf{x}_t^e$ – any other linear unbiased estimator. For the *emission model* we assume,

$$\mathbb{P}(\mathbf{y}_{1:T}^{pred}|\mathbf{z}_{1:T},\mathbf{f}_{1:T}) = \prod_{t=1}^{T} \mathbb{P}(\mathbf{y}_t^{pred}|\mathbf{z}_t), \tag{1.7}$$

Further

- for $\mathbb{P}(\mathbf{x}_t|\mathbf{y}_t)$,
$$\mathbb{P}(\mathbf{x}_t|\mathbf{y}_t) \sim \mathcal{N}(f_{\mu,\gamma}, f_{\Sigma,\gamma}), \tag{1.8}$$
where $f_{\mu,\gamma} = f_\mu(\mathbf{y}_t,\gamma), f_{\Sigma,\gamma} = f_\Sigma(\mathbf{y}_t,\gamma)$,
- for $\mathbb{P}(\varepsilon_t|\mathbf{y}_t)$,
$$\mathbb{P}(\varepsilon_t|\mathbf{y}_t) \sim \mathcal{N}(f_{\mu,\phi}, f_{\Sigma,\phi}), \tag{1.9}$$
where $f_{\mu,\phi} = f_\mu(\mathbf{y}_t,\phi), f_{\Sigma,\phi} = f_\Sigma(\mathbf{y}_t,\phi)$,
- for $\mathbb{P}(\mathbf{y}_t^{pred}|\mathbf{x}_t)$,
$$\mathbb{P}(\mathbf{y}_t^{pred}|\mathbf{z}_t) \sim \mathcal{N}(f_{\mu,\theta}, f_{\Sigma,\theta}), \tag{1.10}$$
where $f_{\mu,\theta} = f_\mu(\mathbf{z}_t,\theta), f_{\Sigma,\theta} = f_\Sigma(\mathbf{z}_t,\theta)$.

1.3 Neural networks

The functions f_μ, f_σ in the previous subsection are approximated by neural networks. The basic form of the neural network is the *feedforward neural network.*

Definition 1 ***Feedforward neural network (FNN)***
For $d,n,N_{L_1},...,N_{L_n} \in \mathbb{N}$, let σ be a non-linear activation function acting component-wise for all i,

$$\sigma \begin{pmatrix} x_1 \\ \dots \\ x_{N_{L_i}} \end{pmatrix} = \begin{pmatrix} \sigma(x_1) \\ \dots \\ \sigma(x_{N_{L_i}}) \end{pmatrix}.$$

and affine linear maps $L_i : \mathbb{R}^{N_{L_{i-1}}} \to \mathbb{R}^{N_{L_i}}, 1 \le i \le n$ with

$$L_i(x) = W_i x + b_i.$$

Then a map $\mathscr{F} : \mathbb{R}^d \to \mathbb{R}^{N_{L_n}}$, given by:

$$\mathscr{F}(x,W,b) := L_n(\sigma(L_{n-1}(\sigma(...\sigma(L_1(x)))))),$$

is called a feedforward ***neural network*** *(FNN) with $n-1$* ***hidden layers*** *of dimensions $N_{L_1},...,N_{L_{n-1}}$, and an* ***output layer*** *of dimension N_{L_n}.*

FNN can be visually represented as:

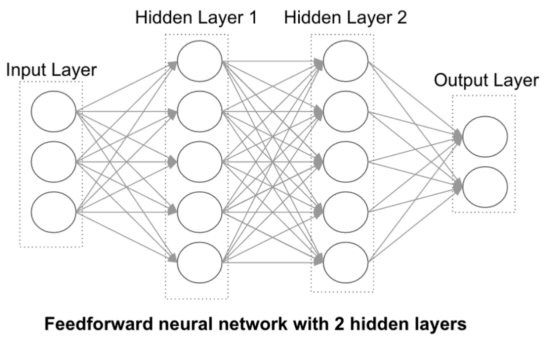

Fig. 1.3 FNN network with 2 hidden layers

Neural networks are considered interesting to many learning problems, among others because FNN can approximate arbitrary well any continuous function on a compact domain.

Theorem 1 ***Universal Approximation Theorem***
Assuming that $\sigma : \mathbb{R} \to \mathbb{R}$ is not polynomial, and taking $K \subset \mathbb{R}^d$ compact for FNN with $n \geq 2$ (i.e., at least one hidden layer), then:
For any continuous $f : K \to \mathbb{R}^{L_N}$ and any ε, there are corresponding affine linear maps with:

$$sup_{x\in K}|f(x) - \mathscr{F}(x,W,b)| \leq \varepsilon.$$

1.3.1 Quality of FNN

Given the following setting:

- FNN $\mathscr{F} : \mathbb{R}^d \to \mathbb{R}^{N_{L_n}}$
- A loss function $\mathscr{L}$ measures the *discrepancy* between the *desired output* y and the *predicted output* $\tilde{y} = \mathscr{F}(x,W,b)$,
$\mathscr{L} : \mathbb{R}^{N_{L_n}} \times \mathbb{R}^{N_{L_n}} \to \mathbb{R}_+$ (a simple choice is the Euclidian distance $\mathscr{L}(\tilde{y},y) = ||\tilde{y}-y||_2^2$).

The simplest optimization algorithm - *gradient descent* - requires only a gradient of the FNN. *Backpropagation* is using the chain rule.

Algorithm 1: Backpropagation

```
With the notation from Definition 1,
Data: $X$ - input data, $Y$ - desired data;
Forward pass,
for $i \leftarrow 0$ to $n$ do
    if $i == 0$ then
        $z_i \leftarrow X$;
    else
        $z_i \leftarrow L_i(...\sigma(L_1(X)))$;
    end
    if $i < n$ then
        $a_i \leftarrow \sigma(z_i)$;
    else
        $a_i \leftarrow \mathscr{F}(X)$;
    end
end
Backward pass,
$\delta_N \leftarrow \frac{\partial \mathscr{L}(\mathscr{F}(X,W,b),Y)}{\partial a_{N-1}}$;
$\frac{\partial \mathscr{L}(\mathscr{F}(X,W,b),Y)}{\partial b_N} \leftarrow \delta_N$;
$\frac{\partial \mathscr{L}(\mathscr{F}(X,W,b),Y)}{\partial W_N} \leftarrow \delta_N a_{N-1}^T$;
for $l \leftarrow N-1$ to $1$ do
    $\delta_l \leftarrow diag(\sigma'(z_l))W_{l+1}^T \delta_{l+1}$;
    $\frac{\partial \mathscr{L}(\mathscr{F}(X,W,b),Y)}{\partial b_n} \leftarrow \delta_n$;
    $\frac{\partial \mathscr{L}(\mathscr{F}(X,W,b),Y)}{\partial W_l} \leftarrow \delta_l a_{l-1}^T$;
end
return $\frac{\partial \mathscr{L}(\mathscr{F}(X,W,b),Y)}{\partial b_l}, \frac{\partial \mathscr{L}(\mathscr{F}(X,W,b),Y)}{\partial W_l}$, for $l \in 1,...,N$.
```

1.4 Deep variational bayesian filtering (DVBF)

Because of the assumptions (1.7), (1.4), we have:

$$\mathbb{P}(\mathbf{y}_{1:T}^{pred}|\mathbf{f}_{1:T}) = \int_{\mathscr{X}^{2T}} \mathbb{P}_\theta(\varepsilon_{1:T}) \prod_{t=1}^{T} \mathbb{P}(\mathbf{y}_t^{pred}|z_t) \prod_{t=0}^{T-1} \mathbb{P}(\mathbf{z}_{t+1}|\mathbf{z}_t, \varepsilon_t, \mathbf{f}_t)\, d\varepsilon_{1:T}\, d\mathbf{z}_{1:T}. \tag{1.11}$$

Because each $\mathbf{z}_{t+1}$ is uniquely determined by $\mathbf{z}_t, \varepsilon_t, \mathbf{f}_t$, and therefore $\mathbb{P}(\mathbf{z}_{t+1}|\mathbf{z}_t, \varepsilon_t, \mathbf{f}_t)$ is a Dirac point measure for each t, the equation (1.11) simplifies to:

$$\mathbb{P}(\mathbf{y}_{1:T}^{pred}|\mathbf{f}_{1:T}) = \int_{\mathscr{X}^T} \mathbb{P}_\theta(\varepsilon_{1:T}) \prod_{t=1}^{T} \mathbb{P}(\mathbf{y}_t^{pred}|\mathbf{z}_t)\, d\varepsilon_{1:T} = \int_{\mathscr{X}^T} \mathbb{P}_\theta(\varepsilon_{1:T}) \mathbb{P}(\mathbf{y}_{1:T}^{pred}|\mathbf{z}_{1:T})\, d\varepsilon_{1:T}. \tag{1.12}$$

This gives us the derivation of the lower bound for $\ln \mathbb{P}(\mathbf{y}_{1:T}^{pred}|\mathbf{f}_{1:T})$:

$$\begin{aligned}
\ln \mathbb{P}(\mathbf{y}_{1:T}^{pred}|\mathbf{f}_{1:T}) &= \ln \int_{\mathscr{X}^T} \mathbb{P}(\varepsilon_{1:T}) \mathbb{P}(\mathbf{y}_{1:T}^{pred}|\mathbf{z}_{1:T}) \frac{\mathbb{P}(\varepsilon_{1:T}|\mathbf{y}_{1:T})}{\mathbb{P}(\varepsilon_{1:T}|\mathbf{y}_{1:T})}\, d\varepsilon_{1:T} \\
&\geq \int_{\mathscr{X}^T} \mathbb{P}(\varepsilon_{1:T}|\mathbf{y}_{1:T}) \ln \left[\mathbb{P}(\mathbf{y}_{1:T}^{pred}|\mathbf{z}_{1:T}) \frac{\mathbb{P}(\varepsilon_{1:T})}{\mathbb{P}(\varepsilon_{1:T}|\mathbf{y}_{1:T})}\right] d\varepsilon_{1:T} \quad \text{(by Jensen's inequality)} \\
&= \mathbb{E}_{\mathbb{P}(\varepsilon_{1:T}|\mathbf{y}_{1:T})} \left[\ln(\mathbb{P}(\mathbf{y}_{1:T}^{pred}|\mathbf{z}_{1:T})) - \ln(\mathbb{P}(\varepsilon_{1:T}|\mathbf{y}_{1:T})) + \ln(\mathbb{P}(\varepsilon_{1:T}))\right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\mathbb{P}(\varepsilon_t|\mathbf{y}_t)} \ln(\mathbb{P}(\mathbf{y}_t^{pred}|\mathbf{z}_t)) - \sum_{t=1}^{T} \mathbb{E}_{\mathbb{P}(\varepsilon_t|\mathbf{y}_t)} \ln \frac{\mathbb{P}(\varepsilon_t)}{\mathbb{P}(\varepsilon_t|\mathbf{y}_t)} \\
&=: \widetilde{\mathscr{L}}(\mathbf{z}_{1:T}, \theta, \phi|\mathbf{f}_{1:T}),
\end{aligned}$$

which is then using Monte-Carlo approximation with M samples per time step t can be rewritten as follows

$$\mathscr{L}(\mathbf{z}_{1:T}, \theta, \phi|\mathbf{f}_{1:T}) := \sum_{i=1}^{N}\sum_{t=1}^{T} \ln(\mathbb{P}(\mathbf{y}_t^{pred}|\mathbf{z}_t^i)) - \sum_{i=1}^{N}\sum_{t=1}^{T} \ln \frac{\mathbb{P}(\varepsilon_t^i)}{\mathbb{P}(\varepsilon_t^i|\mathbf{y}_t)} \tag{1.13}$$

1.5 Learning

Because of the *vanishing* and *exploding gradients*, the training is less effective and takes more time for long sequences $1:T$. Therefore the sequence can be split up into overlapping subsequences and the hyperparameter l_T, which defines the length of the subsequences of $1:T$ can be optimized via grid search during training.
The isotropic Gaussian prior in the experiments was assumed.
Learning is based on the SGD with the *expected risk* defined as follows:

$$-\sum_{i=1}^{N}\sum_{t=1}^{T} \widetilde{c}_j \ln(\mathbb{P}(\mathbf{y}_t^{pred}|\mathbf{z}_t^i)) + \sum_{i=1}^{N}\sum_{t=1}^{T} \ln \frac{\mathbb{P}(\varepsilon_t^i)^{\widetilde{c}_j}}{\mathbb{P}(\varepsilon_t^i|\mathbf{y}_t)} \tag{1.14}$$

One additional learning subtask is to learn the weights $\mathbf{w}^*$ in DMP:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{T} (\mathbf{f}_i - \mathbf{f}_i^{target})^2, \tag{1.15}$$

for a sequence $(\mathbf{f}_1, \mathbf{f}_1^{target}), ..., (\mathbf{f}_T, \mathbf{f}_T^{target})$. In (1.3) we can for example choose the parameters as follows: $c_i = \Delta i, \sigma_i = \Delta\sigma$ for some σ.
Other often used optimization algorithms:

- Second order gradient descent (2GD)
- Second order stochastic gradient descent (2SGD)
- Momentum
- Nesterov Momentum

The algorithms, that have adaptive learning rates:

- AdaGrad
- RMSProp
- Adam

Second order methods, that are often used:

- Newton's Method. One major disadvantage - inversion of the matrix (complexity of $O(n^3)$) on each step
- (Nonlinear) Conjugate Gradients. Solves the issue of zig-zagging, can be used as a mini-batch methods as well [3]
- L-BFGS

During the training steps are similar to 1.2 (T.1, T.2, T.3). We need to have two sequences in a latent space though - $\mathbf{x}_{1:T}$, and $\mathbf{z}_{1:T}$:
each $\mathbf{x}_t \in \mathscr{X}$ is mapped from the corresponding frame $\mathbf{y}_t$, and then compared with the generated element $\mathbf{z}_t \in \mathscr{X}$ of sequence $\mathbf{z}_{1:T}$ to learn, how to generate this sequence $\mathbf{z}_{1:T}$ in a latent space.

1.6 Pseudo code

Algorithm 2: Training of DVBF

```
Initialize $\theta, \gamma, \phi, \mathbf{w}, i = 0, T_a > 0, \eta > 0$;
Data: $Y_{0:T}^M$ – sequences of feature vectors;
while not convergence of $\theta, \gamma, \phi, \mathbf{w}$ do
    for $t \leftarrow 0$ to $T$ do
        $Y_t^M \leftarrow$ random minibatch of $M$ sequences at time $t$;
        if $t < 1$ then
            $Y_t^M \leftarrow$ random minibatch of $M$ sequences at time 1;
            Initialize $Z_0^M \leftarrow Y_0^M, Z_1^M \leftarrow Y_1^M$;
        else
            $\begin{pmatrix} Z_{t+1}^M \\ \dot{Z}_{t+1}^M \end{pmatrix} \leftarrow A \begin{pmatrix} Z_t^M \\ \dot{Z}_t^M \end{pmatrix} + b$;
            $X_t^M \leftarrow \mathcal{N}(f_{\mu,\gamma}, f_{\sigma,\gamma})$ ;
            $\varepsilon_t^{M,pred} \leftarrow \mathcal{N}(f_{\mu,\phi}, f_{\sigma,\phi})$ ;
            $\varepsilon_t^M \leftarrow \mathcal{N}(0, \Sigma_\varepsilon)$ ;
            $Y_t^{M,pred} \leftarrow \mathcal{N}(f_{\mu,\theta}, f_{\sigma,\theta})$ ;
        end
    end
    $w_*^M \leftarrow \arg\min_{\mathbf{w}} \sum_{t=1}^T (\mathbf{f}_t(Z_t^M) - \mathbf{f}_\mathbf{t}^{\mathbf{target}}(X_t^M))^2$;
    $\widetilde{c}_i \leftarrow min(1, 0.01 + i/T_a)$;
    $g \leftarrow -\eta \nabla_{\theta,\gamma,\phi} \big( -\sum_{i=1}^N \sum_{t=1}^T \widetilde{c}_j \ln(\mathbb{P}(\mathbf{y}_t^{pred} | \mathbf{z}_t^i)) + \sum_{i=1}^N \sum_{t=1}^T \ln \frac{\mathbb{P}(\varepsilon_t^i)^{\widetilde{c}_j}}{\mathbb{P}(\varepsilon_t^i | \mathbf{y}_t)} \big)$;
    $\theta, \phi, \gamma \leftarrow$ update parameters using gradients $g$ ;
    $i \leftarrow i + 1$ ;
end
```

Chapter 2

Introduction - kalman filter and expectation maximization

2.1 Connection to neural networks and other tasks (To be improved)

For this we would refer to the extensions of the kalman filter such as *extended kalman filter* (EKF), where the state distribution is approximated by Gaussian random variables and propagated through the first-order linearization of the non-linear system, which can cause large errors in estimation of the posterior mean and variance, and *unscented kalman filter* (UKF), which has the same approximation of the state distribution, but is represented by cleverly sampled minimal set of points, which are then later passed through any *true* nonlinear system, hence they capture the posterior mean and variance accurately up to second order of Taylor expansion.
Having mentioned these two models, let's introduce three important (theoretical) areas of their applications:

- **State estimation:**
  The main task here is a state estimation of a discrete-time non-linear dynamical system:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{v}_t),$$
$$\mathbf{y}_t = h(\mathbf{x}_t, \mathbf{n}_t),$$

  where $\mathbf{u}_t$ is a know external input, $\mathbf{v}_t, \mathbf{n}_t$ - *process* and *observation* noise respectively.

- **Parameter estimation:**
  This is the main connection to neural networks, as here one tries to find non-linear mapping g, parameterized by the *weights vector* w:

$$\mathbf{y}_t = f(\mathbf{x}_t, \mathbf{w}),$$

where $\mathbf{x}_t$ is an input, $\mathbf{y}_t$ - the output.
One example here can be a result of the recent paper by S. Levine et al. [2]
In this paper authors propose a model, called "backprop kalman filter" (BKF), which is a commputation graph consisting of the kalman filter and FNN, that maps the input $\mathbf{o}_t$ to the low dimensional output $\mathbf{z}_t$, or signal for kalman filter and the observation covariance matrix $\mathbf{R}_t$. To show that the model can overcome typical difficulties in

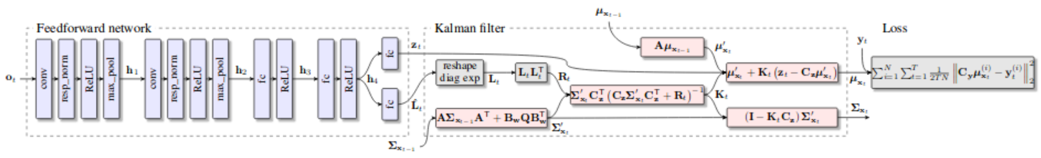
Feedforward network
$\mathbf{o}_t$
conv
resp_norm
ReLU
max_pool
$\mathbf{h}_1$
conv
resp_norm
ReLU
max_pool
$\mathbf{h}_2$
fc
ReLU
$\mathbf{h}_3$
fc
ReLU
$\mathbf{h}_4$
fc
fc
$\mathbf{z}_t$
$\hat{\mathbf{L}}_t$
Kalman filter
$\boldsymbol{\mu}_{\mathbf{x}_{t-1}}$
$\mathbf{A}\boldsymbol{\mu}_{\mathbf{x}_{t-1}}$
$\boldsymbol{\mu}'_{\mathbf{x}_t}$
reshape diag exp
$\mathbf{L}_t$
$\mathbf{L}_t\mathbf{L}_t^T$
$\mathbf{R}_t$
$\boldsymbol{\mu}'_{\mathbf{x}_t} + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}_\mathbf{z}\boldsymbol{\mu}'_{\mathbf{x}_t})$
$\boldsymbol{\Sigma}'_{\mathbf{x}_t}\mathbf{C}_\mathbf{z}^T(\mathbf{C}_\mathbf{z}\boldsymbol{\Sigma}'_{\mathbf{x}_t}\mathbf{C}_\mathbf{z}^T + \mathbf{R}_t)^{-1}$
$\mathbf{K}_t$
$\mathbf{A}\boldsymbol{\Sigma}_{\mathbf{x}_{t-1}}\mathbf{A}^T + \mathbf{B}_\mathbf{w}\mathbf{Q}\mathbf{B}_\mathbf{w}^T$
$\boldsymbol{\Sigma}'_{\mathbf{x}_t}$
$(\mathbf{I} - \mathbf{K}_t\mathbf{C}_\mathbf{z})\boldsymbol{\Sigma}'_{\mathbf{x}_t}$
$\boldsymbol{\Sigma}_{\mathbf{x}_{t-1}}$
$\boldsymbol{\mu}_{\mathbf{x}_t}$
$\boldsymbol{\Sigma}_{\mathbf{x}_t}$
Loss
$\mathbf{y}_t$
$\sum_{i=1}^N \sum_{t=1}^T \frac{1}{2TN} \left\| \mathbf{C}_\mathbf{y}\boldsymbol{\mu}_{\mathbf{x}_t}^{(i)} - \mathbf{y}_t^{(i)} \right\|_2^2$

Fig. 2.1 Backprop KF

visual state estimation, such as long-term tracking with occlusions, processing raw data, and the presence of noise, one of the synthetic tasks was to estimate the position of a red disk from raw images, given occluding disks of different colors and radii. Their trajectories follow linear-Gaussian dynamics with the Gaussian noise perturbing the motion. The difficulty is controlled by changing the number of distracting disks.

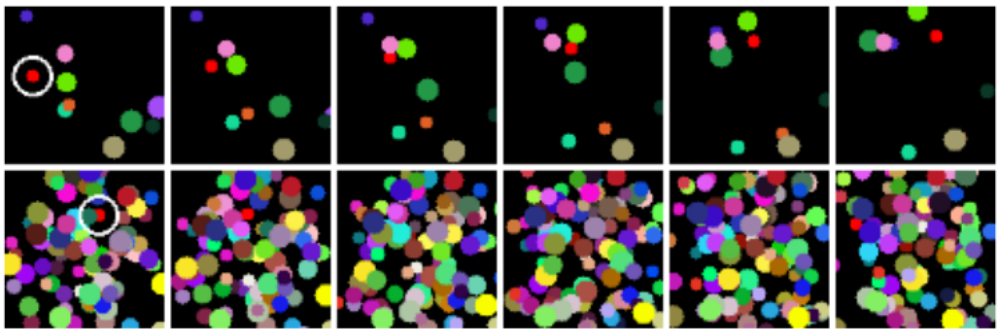

Fig. 2.2 : On this image 6 consecutive frames of two training sequences are shown (one of the low difficulty with 9 distractors, another one - of higher difficulty with 99 distractors). The task is here to track the red disk through the sequence with 100 frames

The models have high sample efficiency, considering, that they are trained only on 100 randomly sampled sequences. The results show, that BKF has outperformed other state of the art models.

Table 1: Benchmark Results

State Estimation Model	**# Parameters**	**RMS test error** $\pm\sigma$
feedforward model	7394	0.2322 ± 0.1316
piecewise KF	7397	0.1160 ± 0.0330
LSTM model (64 units)	33506	0.1407 ± 0.1154
LSTM model (128 units)	92450	0.1423 ± 0.1352
BKF (ours)	**7493**	$\mathbf{0.0537} \pm \mathbf{0.1235}$

- **Dual estimation:**
  It is a special case of the previous example, where the input $\mathbf{x}_t$ is unobserved and requires the combination of the state and parameter estimation (i.e. it is a combination of the previous both cases):

$$\begin{aligned}\mathbf{x}_{t+1} &= f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{v}_t, \mathbf{w}),\\ \mathbf{y}_t &= h(\mathbf{x}_t, \mathbf{n}_t, \mathbf{w}).\end{aligned}$$

2.2 Discrete time filtering and parameter learning

Generally our task can be described as:
Given the observed data $\mathscr{Y} = (\mathbf{y}_1, ..., \mathbf{y}_N)$, estimate for each $k \geq 1$ the state $\mathbf{x}_t$. We have three cases:

- filtering, if $k = N$,
- prediction, if $k > N$,
- smoothing, else.

To solve this tasks we need to be able to:

(T1) Estimate state optimally in a system with given parameters - where the kalman filter and extensions come in play

(T2) To learn the parameters, used in the previous step - here we would need expectation maximization

2.3 State estimation

Let for any k : $x_t, y_t, q_t, w_t, v_t \in \mathbb{R}^n$. We have then the following discrete-time dynamical system:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{q}_t) + w_t, \mathbf{y}_t = g(\mathbf{x}_t, \mathbf{q}_t) + \mathbf{v}_t, \tag{2.1}$$

where $\mathscr{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)$ - are hidden states, $\mathscr{Y}$ - observed noisy data, $\mathscr{Q} = (\mathbf{q}_1, ..., \mathbf{q}_N)$ - external inputs, and Θ - parameters of the model.
Bayesian networks alone represent a powerful tool, that is a generalization of such learning methods as neural networks, kalman filter with extensions and others.
Bayesian networks as DAG, together with markov random fields - undirected graph, both are common representatives of a probabilistic graphical model - they use graphs to encode the conditional dependencies between a set of random variables.
The applications of graphical models are, among the others, speech recognition, computer vision, casual inference, information extraction, and graphical models for protein structure.
With this notion, we can represent the system 2.1 as follows:

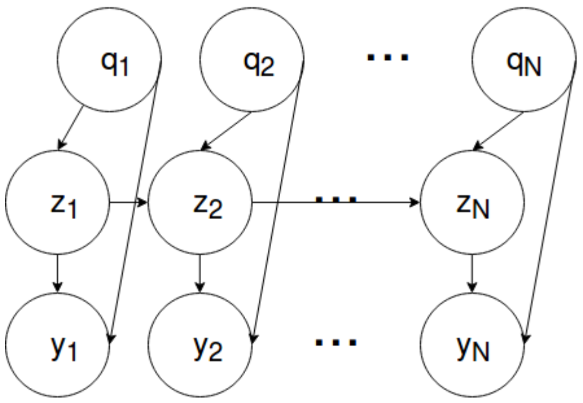

Fig. 2.4 Stochastic dynamical system as a probabilistic graphical model

As one can guess, the task is to model the conditional probability density function of the output given input.
The system is characterized by:

- For any $k \in \{1, ..., N\}$: $\mathbf{w}_t, \mathbf{v}_t \sim \mathscr{N}(0, \sigma_\mathbf{w}), \mathscr{N}(0, \sigma_\mathbf{v})$ and are i.i.d. stochastic processes

- Both $(\mathbf{x}_t)_t$ and $(\mathbf{y}_t)_t$ are stationary, in the sense, that for any $j, \tau, (t_1, ..., t_j) : F_X(\mathbf{x}_{t_1+\tau}, ..., \mathbf{x}_{t_j+\tau}) = F_X(\mathbf{x}_{t_1}, ..., \mathbf{x}_{t_j}), F_Y(\mathbf{y}_{t_1+\tau}, ..., \mathbf{y}_{t_j+\tau}) = F_Y(\mathbf{y}_{t_1}, ..., \mathbf{y}_{t_j})$,where $F_X(\mathbf{x}_{t_1}, ..., \mathbf{x}_{t_j})$ being the cumulative distribution function of the unconditional joint distribution of $(\mathbf{x}_t)_t$ at times $(t_1, ..., t_j)$

- Both f, g are assumed to be differentiable

The starting point would be a linear system - i.e., when both process and measurement functions are linear.
In this case the previous system 2.1 would be stated as:

$$\begin{aligned} \mathbf{x}_{t+1} &= A\mathbf{x}_t + B\mathbf{q}_t + \mathbf{w}_t, \\ \mathbf{y}_t &= C\mathbf{x}_t + D\mathbf{q}_t + \mathbf{v}_t, \end{aligned}$$

where $\mathbf{w}, \mathbf{v} \sim \mathcal{N}(0, \Sigma_\mathbf{w}), \mathcal{N}(0, \Sigma_\mathbf{v})$.
The kalman filter is an implementation of a Bayes' rule:
Given the prior on a state variable $p(\mathbf{x}_1)$ and the model after a noisy observation $p(y_1|x_1)$, using the Bayes' rule one sees, that in order to get go from our prior to posterior, it is needed only to multiply the likelihood from the measurement equation by the prior and to normalize:

$$p(\mathbf{x}_1|\mathbf{y}_1) = \frac{p(\mathbf{y}_1|\mathbf{x}_1)p(\mathbf{x}_1)}{p(\mathbf{y}_1)} = \frac{p(\mathbf{y}_1|\mathbf{x}_1)p(\mathbf{x}_1)}{\int_x p(\mathbf{y}_1|\mathbf{x})p(\mathbf{x})\,dx}$$

Important remarks in state estimation are:

1. if the probability distribution $p(\mathbf{x}_1)$ of initial state is Gaussian, then, since the Gaussian distribution is closed under the linear mapping and convolution, the joint probabilities of all states and outputs are Gaussian as well

2. given the previous point, one can see, that all distributions over hidden states can be completely described by their means and covariance matrices

3. then the algorithm for exactly computing the posterior mean and covariance for $\mathbf{x}_t$, given $\mathcal{Y}$, consists of forward recursion, that uses the observations $(\mathbf{y}_1, ..., \mathbf{y}_t)$ - known as a kalman filter, and backward recursion, that uses observations $(\mathbf{y}_{t+1}, ..., \mathbf{y}_T)$. The combination of both forward and backward recursions is called kalman smoother.

Even though the third point will be discussed in more detail later, we present here the summary of the computation of the means and covariances for $\mathbf{x}_t$: We start with the Gaussian belief on the previous step $\mathcal{N}(\mathbf{x}_{t-1}, \Sigma_{t-1})$, transform in into prior, according to the system dynamics, $\mathcal{N}(\mathbf{x}^*_{t-1}, \Sigma^*_{t-1})$:

$$p(\mathbf{x}_{t-1}) = \mathcal{N}(A\mathbf{x}_{t-1}, A\Sigma_{t-1}A + \Sigma_\mathbf{w})$$

After this, having the model for the noisy output $p(\mathbf{y}_t|\mathbf{x}_t) = \mathcal{N}(C\mathbf{x}_t, \Sigma_\mathbf{v})$ we condition on the given observation $\mathbf{y}_t$ and arrive to kalman filtering equation:

$$
\begin{aligned}
p(\mathbf{x}_t|\mathbf{y}_t) &= \mathcal{N}(\mathbf{x}_t, \Sigma_t),\\
&\mathbf{x}_t = \mathbf{x}^*_{t-1} + K(\mathbf{y}_t - C\mathbf{x}^*_t), \Sigma_t = (I - KC)\Sigma^*_{t-1}\\
&K = \Sigma^*_{t-1}C^T(C\Sigma^*_{t-1}C^T + R)^{-1}
\end{aligned}
$$

Why would we use the extensions of the kalman filter? The reason is simple, but the answer is slightly involved.
We consider extensions to have the state estimation in case of nonlinear f, g.
Mapping through these nonlinear functions results can result in arbitrary distributions, making the integral in Bayes' rule intractable.
The common approaches are:

1. extended kalman filter (EKF) - or local linearization of the nonlinear system around the current state, which allows to apply kalman filter to the linerization, and to give the approximately Normal state distribution after the mapping:

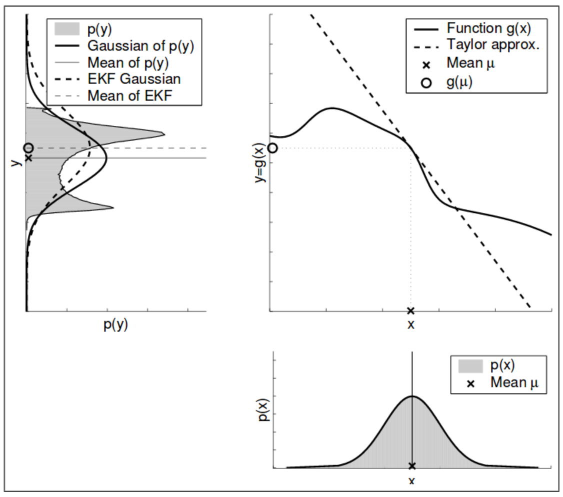

Fig. 2.5 Extended kalman filter

2. unscented kalman filter (UKF) - uses a deterministic sampling technique, called Unscented Transform, to pick the minimal set of (sigma) points around the mean, which are then later mapped through the nonlinear f, g and from which then new mean and variance are estimated:

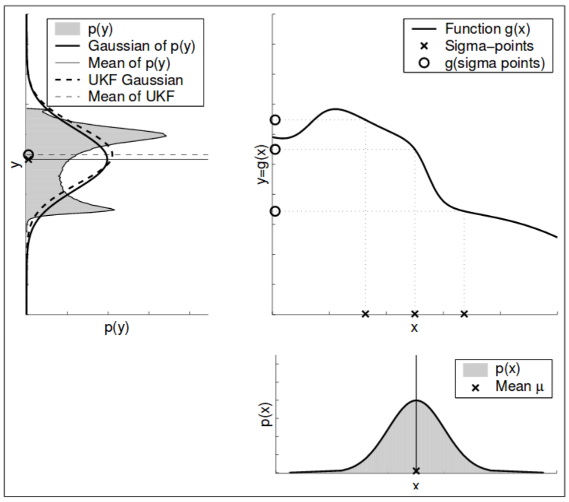

Fig. 2.6 Unscented kalman filter

3. particle filter - randomly samples the points around the mean, propagates though f, g, and re-wights at each step, using likelohood $p(\mathbf{y}|\mathbf{x})$:

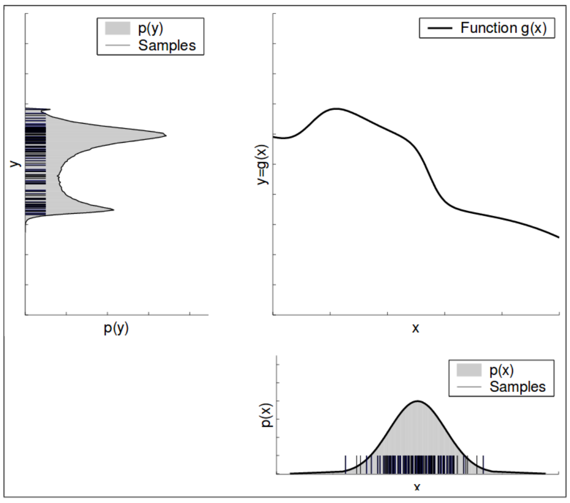

Fig. 2.7 Particle filter

2.4 Parameter learning

Our objective, is to iteratively re-estimate the parameters of the model.
Using the notation as in 2.1 we can continue with expectation maximization (EM).
In EM, the objecive can be achieved, when we maximize the likelihood of the observed data

$\mathscr{L}(\Theta) = P(\mathscr{Y}|\Theta)$ having the hidden variables $\mathscr{X}$, which is equivalent to:

$$\arg\max_{\theta \in \Theta} P(\mathscr{Y}|\Theta) = \arg\max_{\theta \in \Theta} \log P(\mathscr{Y}|\mathscr{Q},\Theta) = \log \int_X P(X,\mathscr{Y}|\mathscr{Q},\Theta)\, dX$$

First, we take any distribution $\widetilde{P}(\mathscr{X})$ over hidden states, and obtain a lower bound on $\mathscr{L}$:

$$\begin{aligned} log \int_X P(X,\mathscr{Y}|\mathscr{Q},\Theta) &= log \int_X \widetilde{P}(X) \frac{P(\mathscr{Y},X|\mathscr{Q},\Theta)}{\widetilde{P}(X)} \\ &\geq \int_X \widetilde{P}(X) log \frac{P(\mathscr{Y},X|\mathscr{Q},\Theta)}{\widetilde{P}(X)} \\ &= \int_X \widetilde{P}(X) log P(\mathscr{Y},X|\mathscr{Q},\Theta) - \int_X \widetilde{P}(X) log \widetilde{P}(X) \\ &= \mathscr{F}(\widetilde{P},\Theta) \end{aligned}$$

In terms of variational bayesian methods, $\mathscr{F}(\widetilde{P},\Theta)$ is a *free energy*, maximization of which w.r.t $\widetilde{P}$ leads to minimization of the KL-divergence (which will be defined in a moment below) between $P(\mathscr{X}|\mathscr{Y},\mathscr{Q},\Theta)$ and $\widetilde{P}(\mathscr{X})$.

Definition 2 ***Kullback–Leibler divergence (KL-divergence)*** *In our case for P and Q continuous random variables, the Kullback–Leibler divergence is defined by:*

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) log \frac{p(x)}{q(x)}\, dx,$$

where p, q - are the probability densities of P, Q.
KL-divergence thus defines thus a measure of dissimilarity between two probability distributions.
One has to mention though, that KL-divergence is not a metric, because it lacks the properties of symmetry and triangle inequality.

The following figure serves as a good intuition for the KL-divergence:

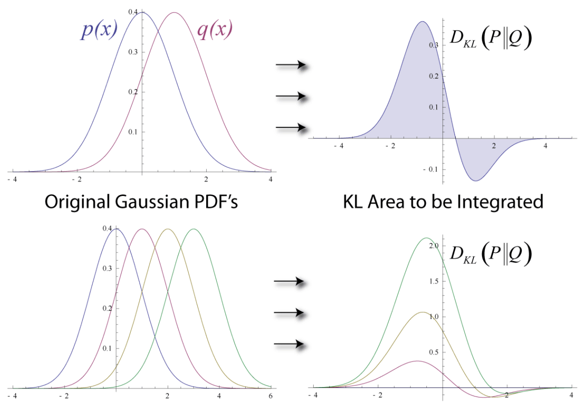

Fig. 2.8 KL-divergence between Gaussian PDF's

Having said that, we add that EM algorithm maximizies $\mathscr{F}$ in two repeating steps w.r.t only $\widetilde{P}$ and w.r.t only Θ.
In other words, starting with an initial value Θ_0, two steps are done consecutively:

$$\textbf{E-step - } \widetilde{P}_{t+1} = argmax_{\widetilde{P}} \mathscr{F}(\widetilde{P}, \Theta_t)$$
$$\textbf{M-step - } \Theta_{t+1} = argmax_{\Theta} \mathscr{F}(\widetilde{P}, \Theta_t)$$

From the remark about KL-divergence, it is intuitively clear, that E-step leads to approximation of $P(\mathscr{X}|\mathscr{Y},\mathscr{Q},\Theta_t)$ by $\widetilde{P}_t(\mathscr{X})$ as $k \to \infty$ w.r.t. KL-divergence.
It is also clear, that:

$$\Theta^*_{t+1} = argmax_{\Theta} \int_X \widetilde{P}(X) log P(\mathscr{Y}, X|\mathscr{Q}, \Theta)$$

The EM alternating steps can be geometrically be thought as a coordinate ascent:

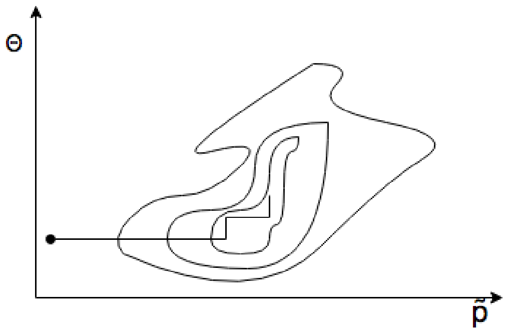

Fig. 2.9 EM as a coordinate ascent

Chapter 3

Results

As it was mentioned in the first chapter, we did the experiments on 4 different movements and present below both the loss using 2D encoding and the progression of the mapping to 2D latent space. Here the movement starts from the red cross and changes its color to green, until it reaches the green cross. One can see, that different movements have different trajectories in the latent space, punching being more straight, and zig-zag walk being more periodic.

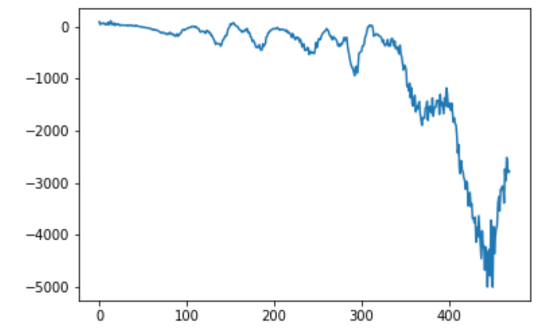
Fig. 3.1 Zig-zag walk, loss with 2D encoding

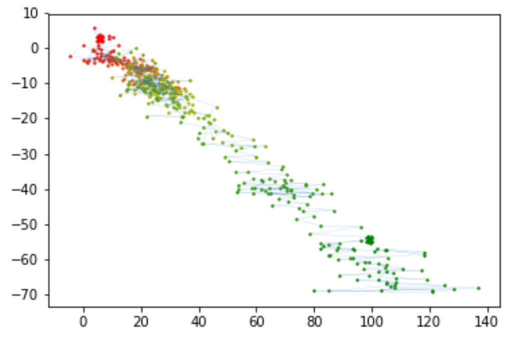
Fig. 3.2 Zig-zag walk, latent space

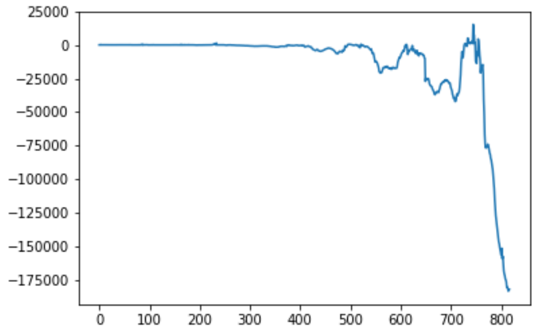
Fig. 3.3 Punching, loss with 2D encoding

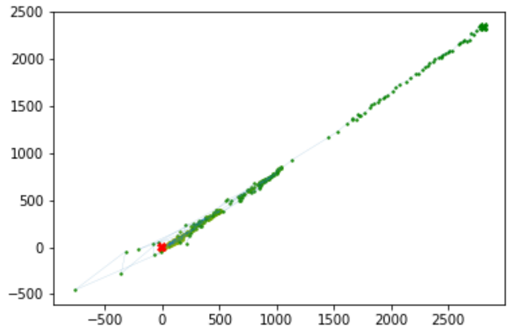
Fig. 3.4 Punching, latent space

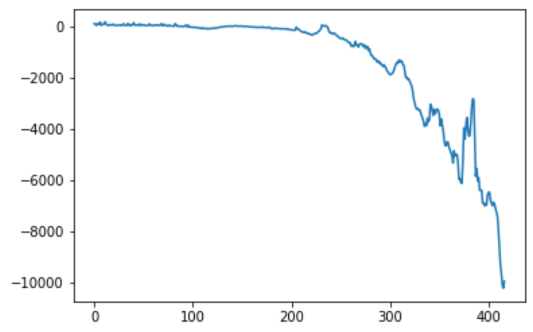
Fig. 3.5 Suitcase, loss with 2D encoding

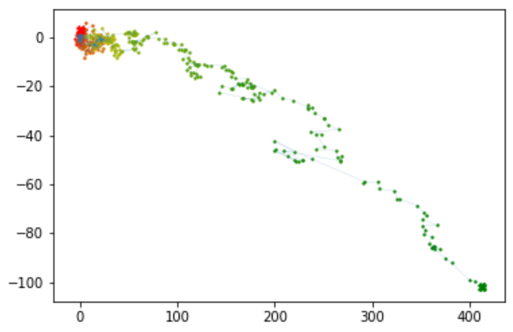
Fig. 3.6 Suitcase, latent space

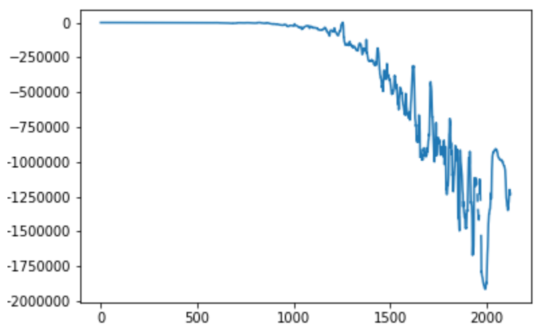
Fig. 3.7 Swimming, loss with 2D encoding

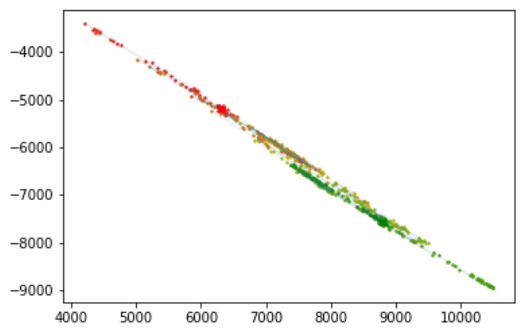
Fig. 3.8 Swimming, latent space

Bibliography

[1] Bottou et al. “The Tradeoffs of Large Scale Learning”. In: (2011).

[2] Levine et al. “Backprop KF: Learning Discriminative Deterministic State Estimators”. In: (2017).

[3] Quoc et al. “On Optimization Methods for Deep Learning”. In: (2011).

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[5] Simon Haykin. *Kalman Filtering and Neural Networks*. Wiley, 2001.

[6] Patrick van der Smagt Nutan Chen Maximilian Karl. “Dynamic Movement Primitives in Latent Space of Time-DependentVariational Autoencoders”. In: *IEEE* (2016).