Chapter 9
Time-Variant Factorization Models

All of our proposed factorization models so far do not model any time-variance within factors. In this chapter, we develop a non-parametric approach that allows to model changes in time for each factor. We will focus on the model itself which is generic and not limited to any optimization task like ranking, classification or regression. Even though, factor models for context-aware ranking can benefit from these extensions, this work is not limited to the ranking task, but is more general. That is why we describe the time-aware factorization models for typical problems instead of limiting the discussion to context-aware ranking.

9.1 Introduction

Factorization models (FM) are the basis for many popular methods in machine learning, including maximum margin matrix factorization (Srebro et al, 2005), (higher-order) singular value decomposition (Lathauwer et al, 2000) or principal component analysis. They subsume many models like Tucker decomposition (Tucker, 1966), Parallel factor analysis/ PARAFAC (Harshman, 1970; Carroll and Chang, 1970) or matrix factorization. The usual assumption for factorization models is that the factors are independent of time. In this chapter, we extend general FM to the case where factors are time-variant.

Analogously to the general idea of factorization models, we decompose each time-variant factor into a time independent part and a time-variant part. The time-variant part is a set of basis functions that is modelled explicitly. Instead of using the same predefined set of time-variant functions for each factor, we generate these functions from the data using a non-parametric kernel approach. This is done per factor because in factorization problems among the variables the time points with (latent) observations differ and also the number of observations differ. Our generated functions make sure that the free model parameters that have to be estimated are placed corresponding to the time points with observations of this variable – e.g. regions with more observations get more parameters. This leads both to a closer fit and less overfitting because the number of basis functions can be reduced and related to the number of observations.

S. Rendle: Context-Aware Ranking with Factorization Models, SCI 330, pp. 137–153.
springerlink.com

First, we shortly introduce the general problem of modelling relations over high-dimensional categorical domains with factorization models. Then, we introduce our time-variant factorization model and show the decomposition of the time-variant factors into free parameters and a set of basis functions. Then we develop a general probabilistic model for this decomposition. This is used to derive simple basis functions like time-invariant biases and single point estimates. After that we introduce our generating approach that is based on a probabilistic kernel between two points of time. Using this kernel the time-variant basis functions can be generated by Gibbs sampling. Finally, we evaluate this approach on a real world dataset and four artificial datasets with varying characteristics.

Besides the rich literature on time-independent factorization models, there is only little research on time-variant settings. Recently, a time-variant matrix factorization model for collaborative filtering has been introduced (Koren, 2009). Koren investigates empirically several time models including bins, drift and a spline[1]. In contrast to this, we introduce a generic model for factorization models (including higher-order $m > 2$) and develop a method for generating the basis functions from the data.

9.2 Problem

Factorization models can model relations over categorical domains. For instance, the matrix factorization model in chapter 6 models the interactions between users and items – e.g. how much a user likes an item. It has been shown empirically (Koren, 2009) that this problem is time-variant and that e.g. a user's taste changes over time.

9.2.1 Time-Variant Relations

We use the notation of chapter 3: Let $X_1, \ldots, X_m$ be the domains of m categorical variables, i.e. $X_i = \{x_1^i, \ldots, x_{|X_i|}^i\}$. Depending on the problem, variables with the same domain are possible – e.g. for representing symmetric matrices like adjacency graphs.

A time-variant relation Y over the categorical domains $X_1, \ldots, X_m$ can be represented as a function y:

$$y : X_1 \times \ldots \times X_m \times \mathbb{R} \to T \tag{9.1}$$

where T is the target domain, e.g. $T \subseteq \mathbb{R}$ for regression problems or $T = \{-1, 1\}$ for classification. Or alternatively in tensor notation:

$$Y : \mathbb{R} \to T^{|X_1| \times \ldots \times |X_m|} \tag{9.2}$$

We will use both notations $y_{x_1,\ldots,x_m}(t)$ and $y(x_1, \ldots, x_m, t)$.

[1] This spline approach is related to our kernel method using a special kernel and sampling independently of the observations.

9.2.2 Sparseness

We assume that only a small part $S \subset (X_1 \times \ldots X_m \times \mathbb{R})$ of the relation Y is observed. Sparseness is assumed both in the dimensions of the variables and even more in time. That means for each variable instance $x \in X$ there are only limited points in time where the relation Y is observed. These are the points $T_x := \{t | \hat{f}_T^x(t) > 0\}$ where the empirical distribution $\hat{f}_T^x$ is non-zero:

$$\hat{f}_T^x(t) := \frac{|(X_1 \times \ldots \{x\} \times \ldots X_m \times \{t\}) \cap S|}{|(X_1 \times \ldots \{x\} \times \ldots X_m \times \mathbb{R}) \cap S|} \tag{9.3}$$

We assume that each variable instance x is observed at least once and thus the denominator does not vanish. Furthermore, the empirical distribution over a relation instance $\mathbf{x} = (x_1, \ldots, x_m)$ has only very small or no support as most relation instances are never observed. Note that this makes the prediction task very hard, because it means that time-variance is not observed directly on the relation instances but is hidden within variable interactions. Thus the problem is more difficult than typical time series problems.

9.2.3 Context-Aware Ranking

Now, we will briefly show how context-aware ranking fits into this setting.

Problem Setting

For ranking within a time-aware problem (eq. (9.1)), the context would be:

$$\mathscr{C} = X_1 \times \ldots \times X_{m-1} \times \mathbb{R} \tag{9.4}$$

And the target is to find a ranking on X_m given a context $\mathbf{c} \in \mathscr{C}$:

$$\succ \subset \mathscr{C} \times X_m^2 \tag{9.5}$$

Modelling

Like discussed in section 3.4, the ranking can be expressed by the function y (eq. (9.1)). But now, y contains a variable (the time) that is not finite and thus the factorization models of chapter 5 cannot be applied directly. Thus, in this chapter we derive time-aware factorization models that can handle time. The rest of this chapter focuses on this task.

Learning

The learning task is then to find a function y that generates a ranking $\succ^y$ which satisfies the infered training data D_S. E.g. by using BCR-OPT and BCR-LEARN.

Sparseness

The semantics of S for the general case (see section 9.2.2) is different from the training data in context-aware ranking. The reason is, that for context-aware ranking there are no direct observations of y itself. Instead, training data is given on pairs within D_S which are direct training data for $\succ$ and thus D_S is indirect training data for y. Nevertheless, we will never need S from now on, but only $\hat{f}^x_T$. This can be defined for context-aware ranking as:

$$\hat{f}^x_T(t) := \begin{cases} \dfrac{|(\mathscr{C} \times \{x\} \times X_m \times \{t\}) \cap D_S| + |(\mathscr{C} \times X_m \times \{x\} \times \{t\}) \cap D_S|}{|(\mathscr{C} \times \{x\} \times X_m \times \mathbb{R}) \cap D_S| + |(\mathscr{C} \times X_m \times \{x\} \times \mathbb{R}) \cap D_S|} & \text{if } x \in X_m \\ \dfrac{|(X_1 \times \ldots \{x\} \times \ldots X_m^2 \times \{t\}) \cap D_S|}{|(X_1 \times \ldots \{x\} \times \ldots X_m^2 \times \mathbb{R}) \cap D_S|} & \text{else} \end{cases} \tag{9.6}$$

With these definitions, also context-aware ranking can be applied to the general model on which we will focus from now on.

9.3 Time-Variant Factorization Models

In the following, we discuss approaches to model $Y(t)$ by a factorization model $\hat{Y}(t)$. First, we introduce the time-variant factorization models and derive a general way of modelling time-variant factors by decomposing them into latent parameters Θ and a predefined time-structure $\mathscr{H}(t)$.

9.3.1 Time-Variant Tucker Decomposition

Based on the example of Tucker decomposition (TD), we introduce time-variant factor models. The TD model of a tensor of mode m is defined as:

$$\hat{Y} := \mathscr{B} \times_1 V^1 \times_2 \ldots \times_m V^m \tag{9.7}$$

$$\mathscr{B} \in \mathbb{R}^{k_1,\ldots,k_m}, \quad V^i \in \mathbb{R}^{|X_i| \times k_i} \tag{9.8}$$

Where V^i is the factorization matrix for a variable X_i. For each variable instance $x \in X_i$, V^i contains one row with k_i values that describe the factors of x. It is very important to note that these factors are never observed, but have to be estimated during the learning phase of the model. TD subsumes a variety of factorization models including PARAFAC or matrix factorization. See chapter 5 for more details about TD and PARAFAC.

A time-variant TD is modelled by making the factors depending on time:

$$\hat{Y}(t) := \mathscr{B}(t) \times_1 V^1(t) \times_2 \ldots \times_m V^m(t) \tag{9.9}$$

$$\mathscr{B} : \mathbb{R} \to \mathbb{R}^{k_1,\ldots,k_m}, \quad V^i : \mathbb{R} \to \mathbb{R}^{|X_i| \times k_i} \tag{9.10}$$

The core tensor $\mathscr{B}$ is usually chosen fixed (e.g. diagonal for PARAFAC/ MF) or for TD it can be computed directly from orthonormal factorization matrices. Thus, we will concentrate on modelling time variant factorization matrices $V^i(t)$. As the modelling is identical for all $i \in \{1,\ldots,m\}$ and to shorten notation, we will drop the index and write $V(t)$ and X respectively. Recall that neither the factors nor the time-variance on variable instances is ever directly observed.

9.3.2 Time-Variant Factor Matrices

Let V be a factorization matrix for a variable X. The task is to model changes in time of $V(t)$. Our idea is to decompose $V(t)$ into value estimates Θ and general time-variant functions $\mathscr{H}(t)$.

$$\hat{V}(t) := \Theta \otimes^* \mathscr{H}(t) \tag{9.11}$$

where Θ is a parameter tensor:

$$\Theta \in \mathbb{R}^{|X| \times k \times l} \tag{9.12}$$

and $\mathscr{H}$ contains l basis functions that are assumed to describe the general-time dependencies of the problem. In general, these time dependencies can be individual per variable instance and factorization feature:

$$\mathscr{H}: \quad \mathbb{R} \to \mathbb{R}^{|X| \times k \times l} \tag{9.13}$$

And $\otimes^*$ is defined as the multiplication and contraction operation that multiplies entries with identical index of X and k and contracts l:

$$\hat{v}_{x,f}(t) := \sum_{i=1}^{l} h_{x,f,i}(t)\, \theta_{x,f,i} \tag{9.14}$$

In this model, Θ are the latent parameters to be estimated and $\mathscr{H}$ is a predefined tensor of functions that is modelled explicitly. Thus with this approach, the task of modelling $V(t)$ is reduced to model $\mathscr{H}(t)$. In the following, we will describe how to obtain models for each factor $v_{x,f}(t) \in V(t)$ – i.e. by defining the basis functions $h_{x,f,i}$ in the set $H_{x,f} := \{h_{x,f,1},\ldots,h_{x,f,l}\}$. To shorten notation, we will skip all indices x, f whenever possible and write $v(t)$.

9.4 Models for Time-Variant Factors

First, we investigate the general model for $v(t)$ that is described by a set H of time-variant basis functions. Next, we derive a probabilistic model that can express any bounded factor. Based on the assumption that the values of a factor at two similar points should be similar, we develop a non-parametric kernel based model that generates the functions in H based on the observed data in S.

9.4.1 General Decomposition Model

The general model for a time-variant factor is:

$$\hat{v}(t) := \sum_{h \in H} h(t)\, \theta_h \tag{9.15}$$

Where H is the set of basis functions for modelling the factor v.

Expressiveness

In the general case, H is not restricted:

$$H^{\text{general}} \subseteq \{h : \mathbb{R} \to \mathbb{R}\} \tag{9.16}$$

Obviously, without a restriction on H, the model in eq. (9.15) can express any function $v(t)$. This can easily be seen by:

$$H := \{v(t)\}, \quad \theta_1 := 1 \quad \Rightarrow\ \hat{v}(t) = v(t) \tag{9.17}$$

For sure this shows only the theoretical expressiveness as v is not known in practice.

Modelling Approaches

An example for H is the set of trigonometric functions:

$$H^{\text{trig}} := \{\sin(nt), \cos(nt) \,|\, n \in \mathbb{N}\} \tag{9.18}$$

with this eq. (9.15) becomes a fourier series which is known to approximate any function well within a fixed interval. Our approach is different from that. Rather than taking the same fixed set of functions for all factors, we generate the functions from the data. To derive these functions, we first formulate a probabilistic model.

9.4.2 Probabilistic Model

For the probabilistic analysis we use the following random variables:

- $T \in \mathbb{R}$ is the random variable for time – t is the realization.
- $V \in \mathbb{R}$ is the random variable of the factor – v is the realization.
- $C \in \{c_1, \ldots, c_l\}$ is the random variable over a finite set of features for explaining the time behaviour of V.

Keep in mind, that v is a shortcut for $v_{x,f}$ and also C is depending on (x, f) – i.e. $C_{x,f}$.

9.4.2.1 General Model for Bounded Factors

Similar to the algebraic decomposition of value estimation and time, we assume conditional independence between factor V and time T given C that explains the time variant behaviour:

$$p(v,t|c) \stackrel{!}{=} p(v|c)\, p(t|c) \tag{9.19}$$

We will show in the following how to relate this model to the general model in eq. (9.15). Afterwards, our kernel model will be a specialization of this general probabilistic model.

Expectation Value $E(v|t)$

It is well known that the (conditional) expectation $E(v|t)$ of a random variate is minimizing the squared prediction error for any variable v at a predetermined point of time t. Hence, a straightforward way of modelling the time-variant factor is:

$$\hat{v}(t) := E(v|t) \tag{9.20}$$

This expectation value can be expressed as a function of C.

Lemma 9.1. *With eq. (9.19), the expected value for v at time point t is:*

$$E(v|t) = \sum_{c \in C} E(v|c)\, p(c|t) \tag{9.21}$$

Proof. With the definition of expectation we have:

$$E(v|t) := \int_{\mathbb{R}} v\, p(v|t)\, dv \tag{9.22}$$

Next, $p(v|t)$ is transformed by marginalization with C:

$$\begin{aligned} p(v|t) &= \sum_{c \in C} p(v,c|t) = \sum_{c \in C} \frac{p(v,t|c)\, p(c)}{p(t)} \\ &\stackrel{(*)}{=} \sum_{c \in C} \frac{p(v|c)\, p(t|c)\, p(c)}{p(t)} = \sum_{c \in C} p(v|c)\, p(c|t) \end{aligned} \tag{9.23}$$

At $(*)$ the conditional independence of v and t given the time-effect variable c is used. Substituting $p(v|t)$ in the expectation leads to:

$$\begin{aligned} E(v|t) &= \int_{\mathbb{R}} v \sum_{c \in C} p(v|c)\, p(c|t)\, dv \\ &= \sum_{c \in C} p(c|t) \underbrace{\int_{\mathbb{R}} v\, p(v|c)\, dv}_{E(v|c)} = \sum_{c \in C} E(v|c)\, p(c|t) \end{aligned} \tag{9.24}$$

Probabilistic Model

In total, the probabilistic model for the factor v is:

$$\hat{v}(t) = \sum_{c \in C} E(v|c)\, p(c|t) \tag{9.25}$$

As discussed before, no parameter in our latent model is directly observed, so $\theta_c := E(v|c)$ has to be estimated in the model's learning phase:

$$\hat{v}(t) = \sum_{c \in C} \theta_c\, p(c|t) \tag{9.26}$$

Comparing this to eq. (9.15) makes the semantics clear: the variables C with the distribution $p_{C|T}$ correspond to H and θ_h is the expectation value $E(v|c)$. Thus, given $p_{C|T}$, the function set H is defined as:

$$H^{\text{prob}}(p) := \{p(c_1|t), \ldots, p(c_l|t)\} \tag{9.27}$$

Expressiveness

With the probabilistic model, H is restricted to all conditional probability distributions over C:

$$H^{\text{prob}} \subseteq \left\{ p \,\middle|\, \forall t : \sum_{c \in C} p(c|t) = 1, \forall c : p(c|t) \geq 0 \right\}$$

This model can express any time variant factor $v(t)$ with a finite lower ($\eta_l := \min_t v(t)$) and upper bound ($\eta_u := \max_t v(t)$). This can be seen by using the following example.

$$\begin{aligned}
C &:= \{c_1, c_2, c_3\} \\
p(c_1|t) &:= \frac{0.5}{\eta_u - \eta_l}(v(t) - \eta_l), & \theta_1 &:= 2\,(\eta_u - \eta_l) \\
p(c_2|t) &:= 0.5 - p(c_1|t), & \theta_2 &:= 0 \\
p(c_3|t) &:= 0.5, & \theta_3 &:= 2\,\eta_l
\end{aligned}$$

Now p is a probability distribution over C for each t and $\hat{v}(t) = v(t)$. Again, this only shows the theoretical expressiveness of this model.

9.4.2.2 Basic Probabilistic Models

In the following, we derive concrete models for $p_{C|T}$. To distinguish the distributions, we will use their random variables as index (e.g. $p_{C|T}$) and use f for densities and p distributions over discrete variables.

Single Point Estimates

Recall, that for each variable instance $x \in X$ there are only limited points in time where the relation Y is observed (see eq. (9.3)). At these time points $t \in T_x$ the latent variable $v_{x,f}(t)$ can be estimated. Using one variable c for each time point $t \in T_x$ a point estimate for $v(t)$ can be made:

$$|C| := |T_x|, \quad \phi : C \to T_x, \quad \phi \text{ bijective} \tag{9.28}$$

$$\forall c \in C : p(c|t) := \delta(\phi(c) = t) \tag{9.29}$$

where ϕ is the placement of the basis function δ via C on the time domain. Applying this to equation (9.26) leads to:

$$v(t) = \delta(t \in T_x)\, \theta_{\phi^{-1}(t)} \tag{9.30}$$

That means for each time with an observation of x, a time depending latent feature is learned such that $\hat{y}$ (see eq. (9.1)) can be reconstructed at that point of time. But (i) no estimations for non observed points of time for this variable can be made (besides the trivial estimate 0). Especially forecasting is not possible. And (ii) no relationships between the values of v at two close points of time are made. Thus, single point estimates might suffer from overfitting. In general this model is useful to model noise on the observations.

Constant effects/ Bias

Often, the largest part of a latent feature is time-independent – i.e. the time-variance is centered around a fixed bias. Modelling time-independence is done by:

$$C := \{c\}, \quad p(c|t) := 1 \tag{9.31}$$

Mixture effects

Mostly, an effect does not appear isolated but mixed with other effects. Two effects $(C_1, p(C_1|t))$ and $(C_2, p(C_2|t))$ can be mixed by linearly combining both into a new effect $(C, p(C|t))$:

$$C := C_1 \cup C_2 \tag{9.32}$$

$$\forall c_i \in C : \; p(c_i|t) := \begin{cases} \alpha\, p(c_i|t), & \text{if } c_i \in C_1 \\ (1-\alpha)\, p(c_i|t), & \text{if } c_i \in C_2 \end{cases}$$

where $\alpha \in [0,1]$ defines the weight for the combination. For sure, also more than two effects can be mixed – e.g. by recursively applying eq. (9.32). For example a reasonable mixture is a combination of a bias, noise (point estimates) and our non-parametric model that we describe next.

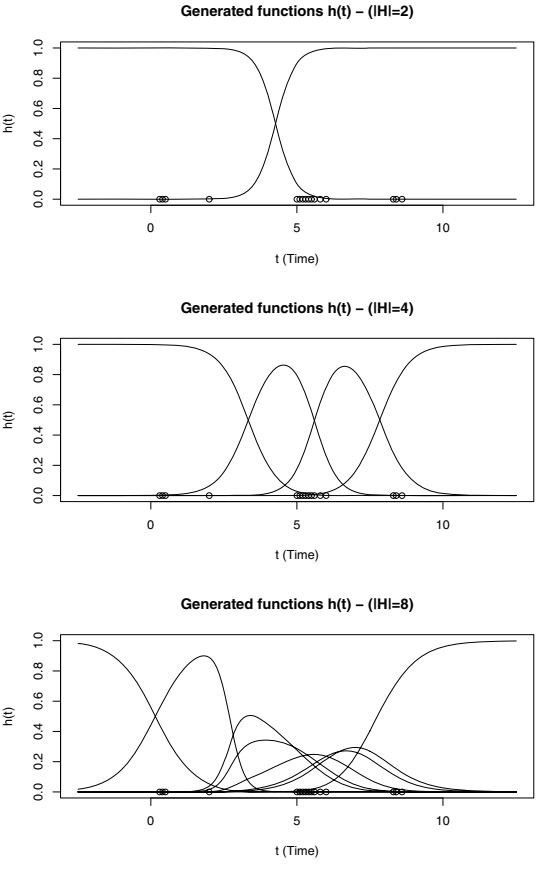

Fig. 9.1 Basis functions H_v for a variable instance x generated by our non-parametric method. This method is based on a kernel function (here Gaussian kernel) and the observed time points of this variable instance. Here, these time points are plotted as dots on the time-axis. As the factor v should be approximated by $|H|$ basis functions, more complexity is placed to regions of time with more (indirect) observations.

9.4.3 Non-parametric Method for Generating Time-Variant Basis Functions

In this section, we derive a method to generate for each variable instance x continuous functions H_x from the observed data S. The model is based on the idea of a continuous dependency K of two points of time – we call this dependency the 'kernel'. We will present how to model $p_{C|T} \cong H$ only with this kernel assumption using the data to 'place' the variables C in regions with many observations (see figure 9.1).

Continuous Time Dependency

The time-dependency of two points of time t_1 and t_2 is given as the density $K(\alpha)$ over their difference $\alpha = t_1 - t_2$. We denote this time dependency as K because we will refer to it as a (time) *kernel*.

Examples for time kernels are:

$$K^{\text{Gauss}}(\alpha) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\alpha^2}{2\sigma^2}\right) \tag{9.33}$$

$$K^{\text{Exp}}(\alpha) := \delta(\alpha > 0)\, \lambda e^{-\lambda \alpha} \tag{9.34}$$

$$K^{\text{Identity}}(\alpha) := \delta(\alpha = 0) \tag{9.35}$$

After a kernel has been chosen, it can be used to define the conditionals between time t and the latent time variables C[2]:

$$f_{T|C}(t|c) := K(t - \phi(c)) \tag{9.36}$$

Again, a mapping $\phi : C \to \mathbb{R}$ from time effect variables to the time domain is necessary. We will later show how to obtain this mapping by Gibbs sampling. With this definition, the exponential kernel can be seen as a causal kernel, because the support of K is only in the future of $\phi(c)$ – that means c influences only the future. E.g. Poisson processes have the exponential kernel as density. The identity kernel leads to single point estimates as described in eq. (9.30).

Kernel-based Model for $p_{C|T}$

We can express $p_{C|T}$ by $f_{T|C}$ using the Bayes theorem:

$$p_{C|T}(c|t) = \frac{f_{T|C}(t|c)\, p_C(c)}{\sum_{c' \in C} f_{T|C}(t|c')\, p_C(c')} \tag{9.37}$$

Using the kernel definition and assuming p_C to be uniformly distributed results in:

$$p_{C|T}(c|t) = \frac{1}{Z(t)} K(t - \phi(c)) \tag{9.38}$$
$$\text{with} \quad Z(t) := \sum_{c \in C} K(t - \phi(c))$$

which defines the basic functions H (eq. (9.27)). An example can be found in figure 9.2. Here we have three variables that have been placed on time points $2, 4, 7$. For each time t on the t-axis, the y-axis shows the probabilistic influence $p_{C|T}(c|t)$ (i.e. the weight) of each variable c on $v(t)$. In this figure, the basis functions $h_1(t) := p(c_1|t)$, $h_2(t) := p(c_2|t)$ and $h_3(t) := p(c_3|t)$ are stacked.

Placement ϕ of Latent Variables

Next, we show how to derive the placement of C on the time domain. It is based on sampling $|C|$ points of time using the empirical distribution over time $\hat{f}_T$ and

[2] As C is a finite set and K is usually a density, one cannot use K directly to express $p_{C|T}$

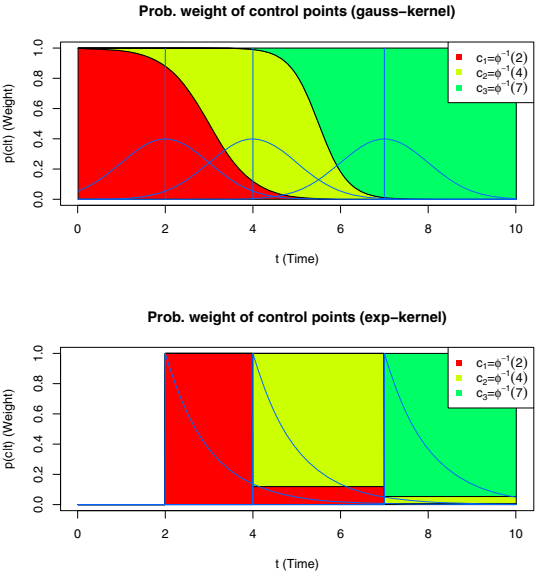

Fig. 9.2 Relationship between the kernel K and $p_{C|T}$ (which defines the basis functions H). There are three time variables c_1, c_2, c_3 placed at time $2, 4, 7$ with their corresponding kernel. The two figures show $p_{C|T}$ for the Gaussian kernel (*top*) and for an exponential kernel (*bottom*). The shaded areas show the weight of the time variables c_1, c_2, c_3. At all points of time $p_{C|T}$ is a probability distribution over C. An example for generated functions H is shown in figure 9.1.

the kernel. For each variable c, one sample t_c is drawn that defines the target of the mapping $\phi(c) := t_c$.

Let T_C be the continuous random variable to describe the placement of C on the time domain. By marginalization, one can express T_C by:

$$\begin{aligned} f_{T_C}(t_c) &= \int_{\mathbb{R}} f_{T_C,T}(t_c,t)\,dt \\ &= \int_{\mathbb{R}} f_{T_C|T}(t_c|t)\, f_T(t)\,dt \end{aligned} \tag{9.39}$$

Sampling directly from the joint probability distribution $f_{T_C,T}$ is not possible as it is unknown. Instead sampling from eq. (9.39) is possible, because the probability of f_T is empirically observed by $\hat{f}_T$ (eq. (9.3)) and the conditional $f_{T_C|T}$ is defined by the kernel K according to eq. (9.36). Thus:

$$f_{T_C}(t_c) \approx \int_{\mathbb{R}} K(t - t_c)\, \hat{f}_T(t)\,dt \tag{9.40}$$

Sampling from f_{T_C} can be performed by Markov Chain Monte Carlo methods (MCMC), i.e. a Gibbs sampler: (1) Sample t^* from $\hat{f}_T$; (2) Sample t_c from $K(t^* - t_c)$. Note that with eq. (9.38) and eq. (9.27), this corresponds to sampling functions h.

Generation Method

The complete procedure for generating for each variable instance x individual basis functions H_x is:

1. Choose a kernel function K.
2. Sample the mapping $\phi : C \to T$ with eq. (9.40)
3. The basis functions H_x are defined by eq. (9.38) and eq. (9.27).

This approach automatically places most parameters Θ to the regions with most observations $\hat{f}_T^x$ (see figure 9.1). The sampling process adapts automatically to the kernel – e.g. with an identity kernel the Gibbs sampler corresponds to bootstrapping. Furthermore, it is possible to use less parameters Θ for variables with only little observations by sampling less functions[3]. This leads to a better generalization capability and prevents overfitting.

The limitation of the kernel approach is that it does not infer from patterns in the past to patterns in the future. E.g. if a sine-like curve has been learned within the region of observations, this shape is not applied to the future. Instead, because of the model assumption that values are similar to close points, the curve would converge to the estimation θ of the last parameter – under the assumption of a decreasing kernel like exponential or Gaussian; another kernel e.g. a damped sine-kernel would behave differently and might correctly repeat the sine shape.

Expressiveness and Relation to ARIMA

If the kernel function can be chosen for each factor freely, because of eq. (9.37) the expressiveness is the same as the general probabilistic model in eq. (9.26).

If we limit the kernel to a fixed one, like Gaussian or exponential, it is still possible to keep the expressiveness of an arbitrary $p_{C|T}$ as long as we use an infinite number of parameters C and allow a free placement ϕ. For this proof, we quantise time $t = i\Delta t$ ($i \in \mathbb{Z}$, $\phi : C \to \mathbb{Z}$) and get from eq. (9.26) and eq. (9.38):

$$v(i\Delta t) = \sum_{c \in C} \theta_c \, g(i\Delta t - \phi(c)\Delta t) \tag{9.41}$$

with:

$$g(i\Delta t - \phi(c)\Delta t) := \frac{1}{Z(i\Delta t)} K(i\Delta t - \phi(c)\Delta t) \tag{9.42}$$

Applying the Z-transform and using its property of linearity and time shifting, this equation becomes:

$$v(z) = g(z)\, q(z) \tag{9.43}$$

$$q(z) = \sum_{c \in C} \theta_c \, z^{-\phi(c)\Delta t} \tag{9.44}$$

Eq. (9.44) is a Laurent-series which converges to a holomorphic function on (a possibly empty) annulus around 0. Any function holomorphic on an annulus has a

[3] In our evaluation, we sample $\#_x^\gamma$ times, where $\gamma \in [0,1]$ is a hyperparameter that controls the number of draws and $\#_x$ is the number of observations for the variable instance x.

convergent Laurent-series with respect to this annulus. Thus any (discrete) time dependency such that the Z-transform is holomorphic on an annulus can be represented by a sum (9.41) with infinitely countable C. In this sense, a model of this type is general enough.

A popular and generic time dependency of an output $v(t)$ on its input is the Box-Jenkins model (Ljung, 1999).

$$v(z) = \frac{c(z)/d(z)}{1 - a(z)/b(z)} g(z) \tag{9.45}$$

where $a(z), b(z), c(z), b(z)$ are given by Laurent series, and thus $\frac{c(z)/d(z)}{1-a(z)/b(z)}$ is rational. This model widely subsumes popular time series models including AR, ARMA and ARIMA. Because any rational function is holomorphic, $h(z) = \frac{c(z)/d(z)}{1-a(z)/b(z)}$ holds, and we conclude that our model covers all major time dependency of $v(t)$ as far as we allow C infinite.

9.5 Evaluation

We study the effectiveness of our method on regression problems – i.e. the square loss with L2-regularization is minimized:

$$\underset{\Theta_1,\ldots,\Theta_m}{\operatorname{argmin}} \sum_{(x_1,\ldots,x_m,t)\in S} (y_{x_1,\ldots,x_m}(t) - \hat{y}_{x_1,\ldots,x_m}(t))^2 + \Lambda \left(\sum_{i=1}^{m} ||\Theta_i||_F^2 \right) \tag{9.46}$$

We optimize this function by stochastic gradient descent which is known to work well on typical factorization problems (Koren, 2009). As factorization model we use PARAFAC which corresponds to the popular MF for cases with more than two modes.

9.5.1 Experimental Setup

9.5.1.1 Datasets

To get a deeper understanding of the effectiveness of our model, we use both synthetic and real-world data. The synthetic data is generated by the following PARAFAC-model.

$$y_{x_1,\ldots,x_m}(t) := \sum_{f=1}^{k} \prod_{i=1}^{m} v^i_{x_i,f}(t) + \varepsilon_{x_1,\ldots,x_m}(t) \tag{9.47}$$

Where $\varepsilon_{x_1,\ldots,x_m}(t) \sim N(0, 0.1)$ is Gaussian noise. To prevent outliers from dominating the evaluation, all values of y are truncated to $[-5, 5]$ – i.e. we cut the tails of the output distribution. Each of the factors $v^i_{x_i,f}$ of the generating model has its own time variance. We consider two types of time variances:

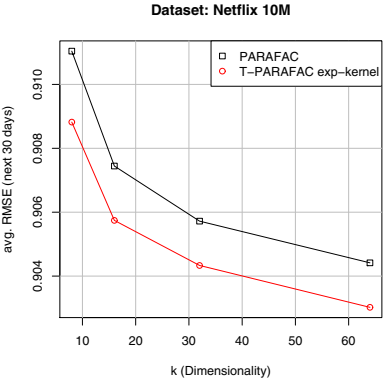

Fig. 9.3 Netflix 10M dataset: PARAFAC vs. T-PARAFAC (exp-kernel) with an increasing number of factorization dimensions (k). The evaluation was made for the last year in the dataset: for each month one model is learned on all past data and the next month is evaluated. The figure shows the average of these 12 RMSE scores per model type and dimensionality.

- **Stationary:** The generating functions are sines.

$$v^i_{x,f} := a^i_{x,f} + b^i_{x,f} \sin\left(\frac{2\pi c^i_{x,f} t}{t_{\max}} + d^i_{x,f}\right)$$

  Where $a^i_{x,f} \sim N(0,1)$, $b^i_{x,f} \sim N(0,0.5)$, $c^i_{x,f} \sim \text{Exp}(1)$, $d^i_{x,f} \sim N(0,\pi)$. Note that this corresponds to an ARMA model without damping, i.e. an ARMA model with strong memory and many parameters.
- **Trend:** The generating functions are linear trends:

$$v^i_{x,f} := a^i_{x,f} + b^i_{x,f} t$$

  Where $a^i_{x,f} \sim N(0,0.75)$ and $b^i_{x,f} \sim N(0,0.5/t_{\max})$.

We have generated 4 datasets: two 2-mode datasets with $|X_1| = |X_2| = 1000$ and two 3-mode datasets with $|X_1| = |X_2| = |X_3| = 100$. For each dataset $t_{\max} = 1000$ and we sampled $|S| = 100,000$ observations. The stationary models have $k = 4$ and the trend models $k = 8$ factors.

As real-world dataset we use a subset of the Netflix dataset with $|S| \approx 10,000,000$, $|X_{\text{user}}| = 40,000$ and $|X_{\text{item}}| = 5000$.

9.5.1.2 Evaluation Protocol

A forecasting problem is set up by splitting the datasets by time t_s. All data before t_s (i.e. $[-\infty, t_s)$) is put into the training set S. And the data S_{test} of the time-span $[t_s, t_s + 30)$ is used for evaluation. The model is trained on S and the RMSE on S_{test} is measured. By moving t_s with a step length of 30, we have independent test sets and overlapping training sets of increasing size. For the artificial datasets we increased t_s from 30 to 960 with the step length of 30. For the Netflix dataset we evaluated the last 12 months starting from day 1842.

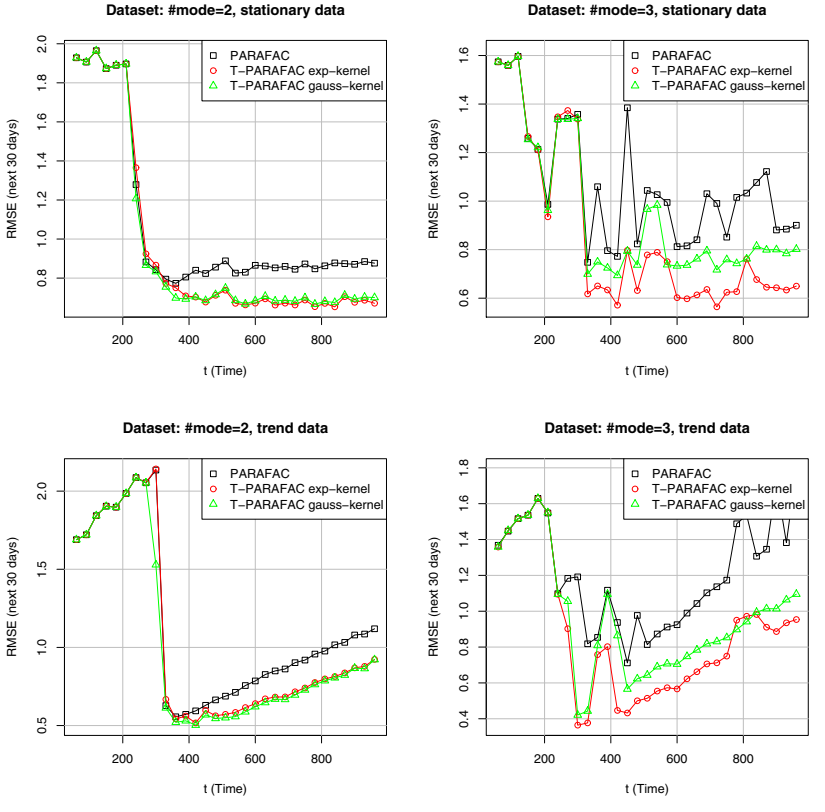

Fig. 9.4 Artificial data: Models are trained on a history from time 0 to time t. With increasing training data, T-PARAFAC (gaussian and exponential kernel) outperforms the standard PARAFAC model.

For the artificial datasets we run the models with several combinations of hyperparameters (k was fixed to the dimensions of the generating process) and the model selection at time t was to choose the hyperparameter combination that was best in the previous one[4]. For Netflix, the hyperparameters were chosen just once per holdout at the first time-split.

9.5.2 Results

Figure 9.3 compares on the Netflix dataset the average forecasting RMSE for the PARAFAC and time-variant PARAFAC (T-PARAFAC) model by varying the factorization dimensionality. As you can see, T-PARAFAC outperforms PARAFAC on

[4] Note that this corresponds to parameter selection using a holdout split of the previous 30 days.

all dimensionalities. This observation matches with previous results on this dataset (Koren, 2009).

Figure 9.4 shows how the next-month forecasting RMSE develops over time under heavily time-variant data. On all datasets, the models need about 250 time steps before enough data is accumulated to identify the factors. After that the T-PARAFAC outperforms PARAFAC. On the stationary data, increasing data helps the T-PARAFAC models to get more stable features whereas normal PARAFAC alternates. On trend data, after identifying the factors, all models suffer from more data. Still, T-PARAFAC has a better performance than PARAFAC as it seems to capture some of the time dependencies. We assume, that the reason why T-PARAFAC cannot fully capture the trend is the limited amount of data and that the trend is not explicitly modelled because the assumption of our model is that the future factors are similar to the last ones.

9.6 Conclusion

In this chapter, we have introduced a general model for time-variant factorization. The general model relies on decomposing a time-variant factor into a set of basis functions and time-invariant parameters. Based on the assumption that at two close points of time, the values of the factor are close, we have developed a method for generating the basis function for each variable instance individually. The generating process is based on a kernel that defines closeness and the time points of observations of the variable instance. We have discussed the expressiveness of this approach and the relations to ARMA models. In future work, we want to investigate how to apply time-patterns learned within a factor to enhance extrapolation to regions without observations. Furthermore, we want to apply the time-aware factor model to context-aware ranking problems.

References

Carroll, J., Chang, J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. Psychometrika 35, 283–319 (1970)

Harshman, R.A.: Foundations of the parafac procedure: models and conditions for an 'exploratory' multimodal factor analysis. UCLA Working Papers in Phonetics, 1–84 (1970)

Koren, Y.: Collaborative filtering with temporal dynamics. In: KDD 2009: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 447–456. ACM, New York (2009)

Lathauwer, L.D., Moor, B.D., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. 21(4), 1253–1278 (2000)

Ljung, L. (ed.): System identification (2nd ed.): theory for the user. Prentice Hall PTR, Englewood Cliffs (1999)

Srebro, N., Rennie, J.D.M., Jaakola, T.S.: Maximum-margin matrix factorization. In: Advances in Neural Information Processing Systems, vol. 17, pp. 1329–1336. MIT Press, Cambridge (2005)

Tucker, L.: Some mathematical notes on three-mode factor analysis. Psychometrika 31, 279–311 (1966)