Non-linear Matrix Factorization with Gaussian Processes

Neil D. Lawrence NEILL@CS.MAN.AC.UK
School of Computer Science, University of Manchester, Kilburn Building, Oxford Road, M13 9PL, U.K.

Raquel Urtasun RURTASUN@ICSI.BERKELEY.EDU
ICSI and EECS, UC Berkeley, 1947 Center Street, Berkeley, CA 94704

Abstract

A popular approach to collaborative filtering is matrix factorization. In this paper we develop a non-linear probabilistic matrix factorization using Gaussian process latent variable models. We use stochastic gradient descent (SGD) to optimize the model. SGD allows us to apply Gaussian processes to data sets with millions of observations without approximate methods. We apply our approach to benchmark movie recommender data sets. The results show better than previous state-of-the-art performance.

1. Introduction

Collaborative filtering is the process of filtering information from different viewpoints. A particular use of the approach is the prediction of user tastes. Given a body of works, such as books or movies, what does a given user's quality rating of one item say about their likely rating for another? There are two dominant approaches to collaborative filtering. The *neighborhood* approach involves expressing a similarity measure between the items to be rated. When a prediction for a new item is required, the similarity between the new item and previously rated items is computed. The new prediction is given by a normalized similarity-weighted sum of the rated items. The alternative approach is the *latent factor* approach, which typically involves a low rank approximation. There is evidence (Bell & Koren, 2007) that both these approaches to collaborative filtering are important for good performance on real data.

In this paper we develop a non-linear extension of the latent factor approach to collaborative filtering. We show how a probabilistic matrix factorization is equivalent to probabilistic principal component analysis. We then extend this model in a non-linear way to give a probabilistic non-linear matrix factorization. Moreover, our formula for predicting a new test rating from previously rated items turns out to be strikingly similar to the weighted similarity equations used in neighborhood approaches.

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

Our model therefore embodies a combined approach to collaborative filtering. Heuristic algorithms exhibiting this combined approach to collaborative filtering have already been suggested (see *e.g.* Koren, 2008). Our algorithm emerges naturally through Gaussian process prediction. We show how the parameters of the algorithm can all be learned using maximum likelihood via stochastic gradient descent.

The latent factor approach to collaborative filtering sees the data as a sparsely populated matrix of ratings. For a data set with N items and D users we take this matrix to be $\mathbf{Y} \in \Re^{N\times D}$. The objective is to factorize $\mathbf{Y}$ into a lower rank form, $\mathbf{Y} \approx \mathbf{U}^\top\mathbf{V}$ (see *e.g.* Rennie & Srebro, 2005; DeCoste, 2006), where $\mathbf{U} \in \Re^{q\times N}$ and $\mathbf{V} \in \Re^{q\times D}$. Prediction can then be done by estimating $(\mathbf{U}, \mathbf{V})$ from the training data and computing the resulting approximation to $\mathbf{Y}$. In this paper we consider a non-linear generalization of this approach.

To guide algorithmic development, we favor a probabilistic perspective, we therefore consider the very natural probabilistic interpretation of this factorization given by Salakhutdinov and Mnih (2008b) they referred to as *probabilistic matrix factorization* (PMF). Their probabilistic model takes the form

$$p\left(\mathbf{Y}|\mathbf{U},\mathbf{V},\sigma^2\right) = \prod_{i=1}^{N} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{V}^\top\mathbf{u}_{:,i},\sigma^2\mathbf{I}\right).$$

where $\mathbf{u}_{:,i}$ is the ith column of $\mathbf{U}$ and $\mathbf{y}_{i,:}$ is a column vector taken from the ith row of $\mathbf{Y}$ containing ratings of the ith item from the users. In practice $\mathbf{y}_{i,:}$ will have many missing values, but we will ignore this aspect for the moment.

In PMF we are modeling the matrix $\mathbf{Y}$ as a noise corrupted low rank matrix. The mean of the distribution is given by the matrix factorization, $\mathbf{U}^\top\mathbf{V}$, and the noise is taken to be Gaussian with variance σ^2. In PMF a Gaussian prior is placed over $\mathbf{U}$, $p\left(\mathbf{U}\right) = \prod_{i=1}^{N}\prod_{j=1}^{q}\mathcal{N}\left(u_{j,i}|0,\alpha_u^{-1}\right)$ and $\mathbf{V}$, $p\left(\mathbf{V}\right) = \prod_{i=1}^{D}\prod_{j=1}^{q}\mathcal{N}\left(v_{j,i}|0,\alpha_v^{-1}\right)$. Ideally, the marginal likelihood of the model would be calculated, but in practice this is not tractable. Instead, in their original paper, Salakhutdinov and Mnih suggest *maximum a posteriori* (MAP) approximations.

It is well known that SVD of a data matrix is directly equivalent to principal component analysis (PCA). In this case we consider $\mathbf{Y}$ to be a *design* matrix, with each row representing a separate data point. Salakhutdinov and Mnih consider their model to be a probabilistic generalization of SVD, it therefore makes sense to ask the question, what is the relationship between PMF and probabilistic PCA (PPCA, Tipping & Bishop, 1999)?

2. PMF is Bayesian PCA

It turns out that the unconstrained PMF is probabilistically equivalent to Bayesian PCA (Bishop, 1999a). This is a little easier to see with a small change of notation. Consider a matrix of latent variables, $\mathbf{X} \equiv \mathbf{U}^\top \in \Re^{N\times q}$ and a mapping matrix which goes from the latent space to the observed data space, $\mathbf{W} \equiv \mathbf{V}^\top \in \Re^{D\times q}$. Using this new notation,

$$p\left(\mathbf{Y}|\mathbf{W},\mathbf{X},\sigma^2\right) = \prod_{i=1}^{N}\mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:},\sigma^2\mathbf{I}\right), \quad (1)$$

the likelihood has a familiar look. It is a multi-output linear regression from a q dimensional feature matrix $\mathbf{X}$ to matrix targets $\mathbf{Y}$. Placing a prior over $\mathbf{X}$,

$$p\left(\mathbf{X}\right) = \prod_{i=1}^{N}\prod_{j=1}^{q}\mathcal{N}\left(x_{i,j}|0,\alpha_x^{-1}\right),$$

and marginalizing leads to

$$p\left(\mathbf{Y}|\mathbf{W},\sigma^2,\alpha_x\right) = \prod_{i=1}^{N}\mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{0},\alpha_x^{-1}\mathbf{W}\mathbf{W}^\top+\sigma^2\mathbf{I}\right).$$

When optimizing with respect to parameters we can absorb α_x into $\mathbf{W}$ leaving the likelihood function associated with PPCA (Tipping & Bishop, 1999). Unfortunately, attempting to marginalize $\mathbf{W}$ is now intractable. Instead, we take a step back and consider marginalizing $\mathbf{W}$ *instead of* $\mathbf{X}$. Taking the prior over $\mathbf{W}$,

$$p\left(\mathbf{W}\right) = \prod_{i=1}^{D}\prod_{j=1}^{q}\mathcal{N}\left(w_{i,j}|0,\alpha_w^{-1}\right)$$

and combining with the likelihood we recover the following marginal likelihood:

$$p\left(\mathbf{Y}|\mathbf{X},\sigma^2,\alpha_w\right) = \prod_{j=1}^{D}\mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0},\alpha_w^{-1}\mathbf{X}\mathbf{X}^\top+\sigma^2\mathbf{I}\right).$$

This is the marginal likelihood of a Bayesian linear regression model with multiple outputs. However, we are not given the input matrix, $\mathbf{X}$, but we optimize with respect to it. Optimization with respect to these inputs is *also* equivalent to probabilistic PCA. It is known as dual probabilistic PCA (DPPCA, Lawrence, 2005).

If we were to marginalize both $\mathbf{X}$ and $\mathbf{W}$ we would recover Bayesian PCA. This is not analytically tractable, but the Laplace approximation (Bishop, 1999a; Minka, 2001) and variational approaches (Bishop, 1999b) have been proposed. Salakhutdinov and Mnih (2008a) used Gibbs sampling to deal with the intractabilities.

The equivalences of the models we've laid out above imply we can deal with marginalization of either $\mathbf{X}$ or $\mathbf{W}$ analytically, leaving us to optimize the resulting marginal likelihood with respect to the remaining matrix *and* the hyper parameters of the model.

2.1. Missing Values

Both models we've described are Gaussian models with a particular covariance structure. This means that marginalizing over missing values is straightforward. Consider a Gaussian distribution over a vector $\mathbf{y}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, $\mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\mu},\boldsymbol{\Sigma}\right)$. We represent an observed subset of $\mathbf{y}$ by $\mathbf{y}_\mathbf{i}$, where $\mathbf{i}$ represents the indexes of the observed values. Marginalizing the missing values leads to a Gaussian of the form $\mathbf{y}_\mathbf{i} \sim \mathcal{N}\left(\boldsymbol{\mu}_\mathbf{i},\boldsymbol{\Sigma}_{\mathbf{i},\mathbf{i}}\right)$, where $\boldsymbol{\mu}_\mathbf{i}$ and $\boldsymbol{\Sigma}_{\mathbf{i},\mathbf{i}}$ represent the mean vector with the rows (and for $\boldsymbol{\Sigma}$ columns) associated with the unobserved elements of $\mathbf{y}$ removed. This implies that when the data matrix is sparse (as is typically the case in collaborative filtering) the likelihoods have the form

$$p\left(\mathbf{Y}|\mathbf{W},\sigma^2,\alpha_x\right) = \prod_{i=1}^{N}\mathcal{N}\left(\mathbf{y}_{i,\mathbf{j}_i}|\mathbf{0},\alpha_x^{-1}\mathbf{W}_{\mathbf{j}_i,:}\mathbf{W}_{\mathbf{j}_i,:}^\top+\sigma^2\mathbf{I}\right)$$

and

$$p\left(\mathbf{Y}|\mathbf{X},\sigma^2,\alpha_w\right) = \prod_{j=1}^{D}\mathcal{N}\left(\mathbf{y}_{\mathbf{i}_j,j}|\mathbf{0},\alpha_w^{-1}\mathbf{X}_{\mathbf{i}_j,:}\mathbf{X}_{\mathbf{i}_j,:}^\top+\sigma^2\mathbf{I}\right). \quad (2)$$

Which of these likelihoods should we select? Either the likelihood in the PPCA form or the DPPCA form can be used. Whilst there are sensible statistical arguments for selecting the likelihood with less parameters (which this is will depend on whether there are more users or items), the maximum likelihood solutions for the fully observed case are equivalent. We will proceed with the DPPCA form, though, as typically there are more users than items in collaborative filtering, and in this case DPPCA has fewer parameters. However, it should be borne in mind that the ideas that follow apply equally to the PPCA form, but with the roles of users and items reversed (there is a duality between the likelihoods that should have become apparent by now).

When $\mathbf{Y}$ is fully observed, a global maximum for (2) with respect to $\mathbf{X}$ and σ^2 can be found through an eigenvalue problem. When there are missing values, Tipping and Bishop (1999) suggested an expectation-maximization (EM) algorithm. However, when D is very large EM is computationally too expensive and it makes sense to consider stochastic gradient descent.

2.2. Stochastic Gradient Descent

Rather than maximizing the likelihood through an EM algorithm we propose presenting ratings for each user one at a time and computing the gradient of the log likelihood for that user. The parameters $\mathbf{X}$, α_w and σ^2 can then be updated by the gradients for that user, and a new user is presented. The objective is to maximize the log likelihood, conversely we minimize the negative log likelihood. For the jth user this is given by

$$E_j(\mathbf{X}) = \frac{N_j}{2}\log|\mathbf{C}_j| + \frac{1}{2}\left(\mathbf{y}_{\mathbf{i}_j,j}^{\top}\mathbf{C}_j^{-1}\mathbf{y}_{\mathbf{i}_j,j}\right) + \text{const.},$$

where $\mathbf{C}_j = \alpha_w^{-1}\mathbf{X}_{\mathbf{i}_j,:}\mathbf{X}_{\mathbf{i}_j,:}^{\top} + \sigma^2\mathbf{I}$ and N_j is the number of items rated by user j. The gradient with respect to $\mathbf{X}$ can be computed as

$$\frac{\mathrm{d}E_j(\mathbf{X})}{\mathrm{d}\mathbf{X}_{\mathbf{i_j},:}} = -\mathbf{G}\mathbf{X}_{\mathbf{i_j},:}$$

with $\mathbf{G} = \left(\mathbf{C}_j^{-1}\mathbf{y}_{\mathbf{i}_j,j}\mathbf{y}_{\mathbf{i}_j,j}^{\top}\mathbf{C}_j^{-1} - \mathbf{C}_j^{-1}\right)$. Gradients with respect to σ^2 and α_w can also be found. The computational complexity of the gradient computation is $O(N_j^3)$ or, if $q < N_j$ the matrix inversion lemma,

$$\mathbf{C}^{-1} = \sigma^{-2}\mathbf{I} - \sigma^{-4}\mathbf{X}\left(\alpha_w\mathbf{I}\sigma^{-2} + \mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{X}^{\top},$$

can be used to give a complexity of $O(q^3)$ for the inversion.

3. Non-Linear PMF via GP-LVMs

We have already highlighted the fact that probabilistic matrix factorization, with the parameters $\mathbf{W}$ marginalized is a Bayesian multi-output regression model in which we optimize with respect to the inputs to the regression. This type of model is equivalent to probabilistic PCA. However, it also belongs to a larger class of models called Gaussian process latent variable models (GP-LVM). Lawrence (2005) showed how the matrix $\mathbf{C}$ has an interpretation as a Gaussian process (GP) covariance matrix. The GP associated with the covariance function $\mathbf{C} = \alpha_w^{-1}\mathbf{X}\mathbf{X}^{\top} + \sigma^2\mathbf{I}$ is a linear model. However, by replacing the inner product matrix, $\mathbf{X}\mathbf{X}^{\top}$, by a Mercer kernel the model becomes a non-linear GP model. Maximization of the log likelihood can no longer be done through an eigenvalue problem, but it is straightforward to apply stochastic gradient descent in the manner described above.

The regression model from (1) can be written as a product of univariate Gaussian distributions,

$$p\left(\mathbf{Y}|\mathbf{W},\mathbf{X},\sigma^2\right) = \prod_{j=1}^{D}\prod_{i=1}^{N}\mathcal{N}\left(y_{i,j}|f_j\left(\mathbf{x}_{i,:}\right),\sigma^2\mathbf{I}\right),$$

where the mean of each Gaussian is given by the inner product $f_j\left(\mathbf{x}_{i,:}\right) = \mathbf{w}_{j,:}^{\top}\mathbf{x}_{i,:}$. Probabilistic PCA can be recovered by marginalizing either $\mathbf{W}$ or $\mathbf{X}$. The GP-LVM is recovered by recognizing that we can place the prior distribution *directly* over the function $f(\cdot)$ through a Gaussian process (Rasmussen & Williams, 2006).

A Gaussian process (GP) can be thought of as a probability distribution for generating functions. The GP is specified by a mean and a covariance function. For any given set of observations of the function, $\mathbf{f}$, the joint distribution over those observations is Gaussian. Restricting ourselves to GPs with a zero mean function, they are distributed as $p(\mathbf{f}|\,\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0},\mathbf{K})$, where $\mathbf{K}$ represents the covariance function. The covariance function is made up of elements, $k(\mathbf{x}_{i,:},\mathbf{x}_{j,:})$ that encode the degree of correlation between two samples, f_i, f_j from $\mathbf{f}$ as a function of the inputs associated with those samples, $\mathbf{x}_{i,:}$ and $\mathbf{x}_{j,:}$. For a covariance function to be valid, it has to lead to a positive semi-definite matrix $\mathbf{K}$ for all valid inputs to the function. In practice that means that valid covariance functions have to be positive definite functions, *i.e.* the class of valid covariance functions is the same as the class of Mercer kernels (Schölkopf & Smola, 2001). A linear regression model is a GP in which the covariance function is taken to be $k\left(\mathbf{x}_{i,:},\mathbf{x}_{j,:}\right) = \mathbf{x}_{i,:}^{\top}\mathbf{x}_{j,:}$.

A widely used covariance function that gives a prior over non-linear functions is known as the RBF covariance,

$$k\left(\mathbf{x}_{\ell,:}, \mathbf{x}_{i,:}\right) = \alpha_m \exp\left(-\frac{\gamma_m}{2}||\mathbf{x}_{\ell,:} - \mathbf{x}_{i,:}||^2\right).$$

This covariance can be substituted directly for the linear covariance function in (2) giving the following probabilistic model,

$$p\left(\mathbf{Y}|\mathbf{X}, \sigma^2, \boldsymbol{\theta}\right) = \prod_{j=1}^{D} \mathcal{N}\left(\mathbf{y}_{\mathbf{i}_j,j}|\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I}\right).$$

where $\boldsymbol{\theta}$ are the parameters of the covariance function. Alternative covariance functions can also be considered, but in this paper we focus only on the RBF and linear covariance functions.

3.1. Prediction of User Rating: Relations to Neighborhood Approaches

The parameters that need to be stored after model training are the latent variables, $\mathbf{X}$, and the parameters of the covariance function. When a prediction is required for a new item, these parameters are combined using the standard formula for prediction by Gaussian processes (Rasmussen & Williams, 2006). The mean of a user j's prediction for item ℓ is given by

$$\boldsymbol{\mu}_{\ell,j} = \mathbf{s}^\top \mathbf{y}_{\mathbf{i}_j,:}, \qquad (3)$$

where we have defined

$$\mathbf{s} = \left(\mathbf{K}_{\mathbf{i}_j,\mathbf{i}_j} + \sigma^2\mathbf{I}\right)^{-1} \mathbf{k}_{\mathbf{i}_j,\ell}.$$

In other words, the mean prediction for an unseen item for the user is given by a weighted sum of the user's rated items. This has a very similar feel to neighborhood models for collaborative filtering, where the neighborhood is based on item similarity. In this approach to collaborative filtering, similarity scores are made between the test item and the user's rated items. These are often based on correlation measures. Prediction of the test item is made by summing the user's rated items weighted by their normalized similarities to the test item.

For the GP-LVM approach, the neighborhood is being constructed in the latent space[1], $\mathbf{X}$. Consider the RBF covariance function: $k_{\ell,i} = \alpha_m \exp\left(-\frac{\gamma_m}{2}||\mathbf{x}_{\ell,:} - \mathbf{x}_{i,:}||^2\right)$. As a user's rated items become better separated in the latent space, then $\mathbf{K}_{\mathbf{i}_j,\mathbf{i}_j}$ becomes a constant diagonal matrix and $\mathbf{s}$ becomes a scaled version of $\mathbf{k}_{\mathbf{i}_j,\ell}$. Now note that the elements of $\mathbf{k}_{\mathbf{i}_j,\ell}$ *are* just scaled similarities between the test rating and the user's rated items. For this case our prediction is functionally identical to the neighborhood approach to collaborative filtering.

Note that the model can rapidly deal with new users. A prediction for a new user can be made using (3) without any change to the parameters of the system.

An additional advantage of the Gaussian process is that it provides a full posterior distribution over the predictions. This allows us to compute a variance for the prediction,

$$\varsigma_{\ell,j} = k_{\ell,\ell} + \sigma^2 - \mathbf{k}_{\mathbf{i}_j,\ell}^\top \left(\mathbf{K}_{\mathbf{i}_j,\mathbf{i}_j} + \sigma^2\mathbf{I}\right)^{-1} \mathbf{k}_{\mathbf{i}_j,\ell}.$$

The variance could be helpful in determining whether or not to propose the item to the user: it expresses the confidence with which the model is estimating the user's preference. As we will see in the experiments there is a strong correlation between the variance and the number of items the user has rated.

4. Experimental Evaluation

We consider three data sets to assess the quality of the GP-LVM for collaborative filtering. The data sets are widely used benchmarks for collaborative filtering, each containing a set of item ratings from different users. These data sets are the EachMovie, the 1M MovieLens, and the recently released 10M MovieLens data[2]. We don't show results on Netflix. State-of-the-art on Netflix typically involves model combinations and bespoke algorithm modification to account for the particular peculiarities of the data (Salakhutdinov & Mnih, 2008b; Salakhutdinov & Mnih, 2008a; Bell & Koren, 2007; Bell et al., 2008). This makes it an interesting challenge, but perhaps a less good evaluator of generic algorithms for collaborative filtering. Two of the data sets we consider (EachMovie and 1M MovieLens) have been widely used benchmarks, allowing us to compare our approach to a range of other approaches. There are currently no other published results on the third data set (the recently released 10M MovieLens), but we expect it will also become a well used benchmark and our result serves to show that our algorithm is easily extended to larger systems.

For the 1M MovieLens and EachMovie data sets, we mimic the setup used by Marlin (2004a), allowing us to compare directly to the best published results which include: the user rating profile (URP, Marlin, 2004b), Attitude (Marlin, 2004a), maximum margin matrix factorization (MMMF, Rennie & Srebro, 2005), the approach of (Park & Pennock, 2007) that combines collaborative filtering with item proximities, and en-

[1]By reversing the roles of $\mathbf{W}$ and $\mathbf{X}$ we could also develop a user-orientated neighborhood model.

[2]See `http://www.grouplens.org/`.

sembles of MMMF (DeCoste, 2006). To our knowledge the results of DeCoste (2006) were the best reported to date on the EachMovie and 1M MovieLens databases.

We explore the importance of the dimensionality of the latent space, q; how performance varies against the available training data and the effect of covariance function choice on the results. For training the models we always use stochastic gradient descent (SGD). Parameters of the stochastic gradient descent algorithm were "tuned" on the 100k MovieLens data[3] (results not presented) and were fixed for the three presented data sets. For all our experiments we used 10 epochs of SGD, with a learning rate of 1×10^{-4} and a momentum of 0.9. Latent points were initialized by drawing from a zero mean spherical Gaussian with standard deviation 1×10^-3. All covariance parameters were initialized as 1 except noise variance which was fixed to 5 and bias variance which was fixed to 0.11. Our code is available from `http://www.cs.man.ac.uk/~neill/collab/`.

4.1. Marlin's setup

In his thesis Marlin (2004a) defines two types of generalization, "weak" and "strong". *Weak* generalization is a single step process which involves filling-in missing data in the rating matrix. *Strong* generalization is a two-stage process, where the models are trained on one set of users and the test predictions are on a disjoint set of users. The learner is given sample ratings of those users, but may not use those ratings until after the initial model is constructed.

Marlin used the normalized mean absolute error (NMAE) as an error measure so that random guessing produces a score of 1. NMAE is computed by normalizing the NMAE by a factor that depends on the range of the ratings[4]. The factor is 1.6 for MovieLens data set (*i.e.*, ratings varying $\{1, \cdots, 5\}$), and 1.944 for the EachMovie data set (*i.e.* ratings varying $\{1, \cdots, 6\}$).

EachMovie Data The EachMovie data set contains 2.6 million ratings for 1,648 movies and 74,424 users. The ratings range $\{1, 2, \cdots, 6\}$. Following Marlin (2004a) users with fewer than 20 ratings were discarded; This left us with 36,656 users. We selected 30,000 randomly that were used for "weak" generalization, while the remaining 6,656 users were used for the strong generalization. We report results averaged over the same 3 partitions originally used in Marlin (2004a) and replicated by Rennie and Srebro (2005); DeCoste (2006); Park and Pennock (2007), where one movie is withheld to construct the test set.

Table 1. **EachMovie**. Our approach outperforms the URP and Attitude algorithms (Marlin, 2004a), the MMMF (Rennie & Srebro, 2005), E-MMMF of (DeCoste, 2006), and the Item-based approach of (Park & Pennock, 2007).

	Weak NMAE	Strong NMAE
URP	0.4422 ± 0.0008	0.4557 ± 0.0008
Attitude	0.4520 ± 0.016	0.4550 ± 0.0023
MMMF	0.4397 ± 0.0006	0.4341 ± 0.0025
Item	0.4382 ± 0.0009	0.4365 ± 0.0024
E-MMMF	0.4287 ± 0.0023	0.4301 ± 0.0035
Ours Linear	$\mathbf{0.4209 \pm 0.0017}$	$\mathbf{0.4171 \pm 0.0054}$
Ours RBF	$\mathbf{0.4179 \pm 0.0018}$	$\mathbf{0.4134 \pm 0.0049}$

Table 1 shows the NMAE for the different baselines as well as our approach. Note that our approach, with either a linear or an RBF kernel, outperforms significantly all the baselines. Importantly, our approach uses latent dimensions much smaller than the 100D used in MMMF and E-MMMF. In particular, for weak and strong generalization we use 20D latent spaces for the RBF and linear models. Larger dimensional spaces resulted in better performance especially for the strong generalization, since the latent space is trained using more data, and thus is less prone to overfitting.

Experiments showing the influence of the latent dimensionality are shown below. The Weak RMSE[5] of our approach when using a 20D latent space is ($\mathbf{1.1110} \pm 0.0028$) for the RBF kernel and ($\mathbf{1.1118} \pm 0.0022$) for linear. The Strong RMSE of our approach when using a 20D latent space is ($\mathbf{1.0981} \pm 0.0077$) for the RBF kernel and ($\mathbf{1.1008} \pm 0.0080$) for the linear kernel.

1M MovieLens The 1M MovieLens data consists of 1 million ratings for 6,040 users, and 3,952 movies, with ratings ranging $\{1, 2, \cdots, 5\}$. 5,000 users were used for weak generalization and the remaining 1,040 for strong generalization. As before, we reported results averaged over the same 3 partitions used in (Marlin, 2004a; Rennie & Srebro, 2005; DeCoste, 2006;

[3]The tuning process was very coarse: we always used momentum 0.9 and 10 epochs of SGD. The learning rate was then logarithmically decreased from 0.1 until results seemed to be converging.

[4]The motivation for this factor is that it is the mean absolute error that would be obtained if the ratings were uniformly distributed and the prediction was uniform random guessing. A much better prediction in this case would be to guess the median. For MovieLens the prediction would be 3 and it would lead to a score of 1.2. A still better "dumb" score could be obtained by predicting the median of the observed data. However, the use of the factors 1.6 and 1.944 has become somewhat standardized, so we follow this practice.

[5]Note that for EachMovie RMSE error is higher than for MovieLens as the range of the ratings is larger.

Table 2. **1M MovieLens**. Our approach outperforms the URP and Attitude algorithms (Marlin, 2004a), the MMMF (Rennie & Srebro, 2005), E-MMMF (DeCoste, 2006), and the Item-based approach (Park & Pennock, 2007) when using a non-linear latent space.

	Weak NMAE	Strong NMAE
URP	0.4341 ± 0.0023	0.4444 ± 0.0032
Attitude	0.4320 ± 0.0055	0.4375 ± 0.0028
MMMF	0.4156 ± 0.0037	0.4203 ± 0.0138
Item	0.4096 ± 0.0029	0.4113 ± 0.104
E-MMMF	0.4029 ± 0.0027	0.4071 ± 0.0093
Ours linear	0.4052 ± 0.0011	**0.4071 ± 0.0081**
Ours RBF	**0.4026 ± 0.0020**	**0.3994 ± 0.0145**

Park & Pennock, 2007).

Table 2 shows the NMAE for the baselines as well as for our approach. Using an RBF covariance function our approach results in the best performance for both weak and strong generalization. As before, the latent space required by our approach is small. In particular, for weak generalization we use a 10D latent space for the RBF and 11D for the linear model, and for strong generalization a 14D latent space for the non-linear model and 15D for the linear one. The weak RMSE of our approach is (**0.8801** ± 0.0082) for a 12D RBF model and (**0.8791** ± 0.0080) for a 14D linear model. The Strong RMSE of our approach is (**0.8748** ± 0.0268) for a 15D RBF model and (**0.8775** ± 0.0239) for a 11D linear model.

As for the EachMovie data set, the method that previously performed best among the baselines is the ensembles of MMMF (DeCoste, 2006) that combines predictions from multiple MMMF models. In particular predictions from 100 models created using bagging and multiple random weight seeds are combined by plurality voting, voting by averaging and voting by confidence. Our approach could also benefit from averaging the predictions from multiple models. We set up a small experiment where we combine by averaging the predictions of 11 models whose latent space dimensionality ranges 5 to 15. The NMAE error for Weak NMAE was reduced to (**0.3987** ± 0.0013).

A further point of interest is the fact that with our approach, unlike most of the baselines, the strong NMAE is often smaller than the weak NMAE. This might look surprising, however the data set partitioning means that the models learned for strong generalization have seen more data (*i.e.*, 30,000 for EachMovie and 5,000 for 1M MovieLens) than the weak generalization. This combined with the fact that all user specific parameters are marginalized in the model leads to improved

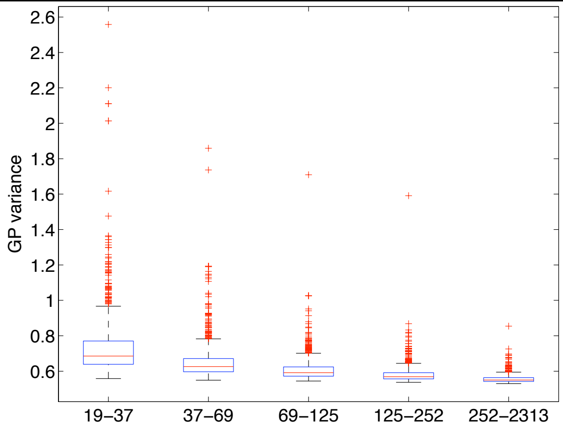

Figure 1. **GP variance:** as a function of the number of movies rated, for a 10D latent space learned on 1M MovieLens Weak. The variance of the GP is a good indicator of the uncertainty in the model, its value decreases with the amount of movies rated.

performance.

Figure 1 shows the variance of the GP as a function of the movies rated for a 10D RBF model learned for 1M MovieLens Weak. Note that the variance of the GP is a good indicator of the uncertainty in the model, its value decreases as the amount of movies rated increases.

4.2. Latent dimensionality and training set size

To test the response of our models to increasing data set size we also conduct experiments in a different setting than Marlin. We generate different training set sizes by splitting different percentages of the data as training and testing.

Figure 2 depicts NMAE and RMSE errors average over 5 random partitions as a function of the dimensionality of the latent space. As expected our model's predictions are more accurate when more data is used for training. Small latent dimensionalities are preferred for small training set sizes (*e.g.* 30%) to avoid overfitting. When using more data, higher dimensional latent spaces result in better performance.

4.3. Using different kernels

Figure 3 shows NMAEs when using different kernels for the MovieLens and EachMovie databases. Note than in general non-linear models (*e.g.* RBF) outperform linear models, showing the benefit of our approach with respect to other approaches that assume that the mapping is linear (*e.g.* Rennie & Srebro, 2005; DeCoste, 2006; Salakhutdinov & Mnih, 2008b; Salakhutdinov & Mnih, 2008a.

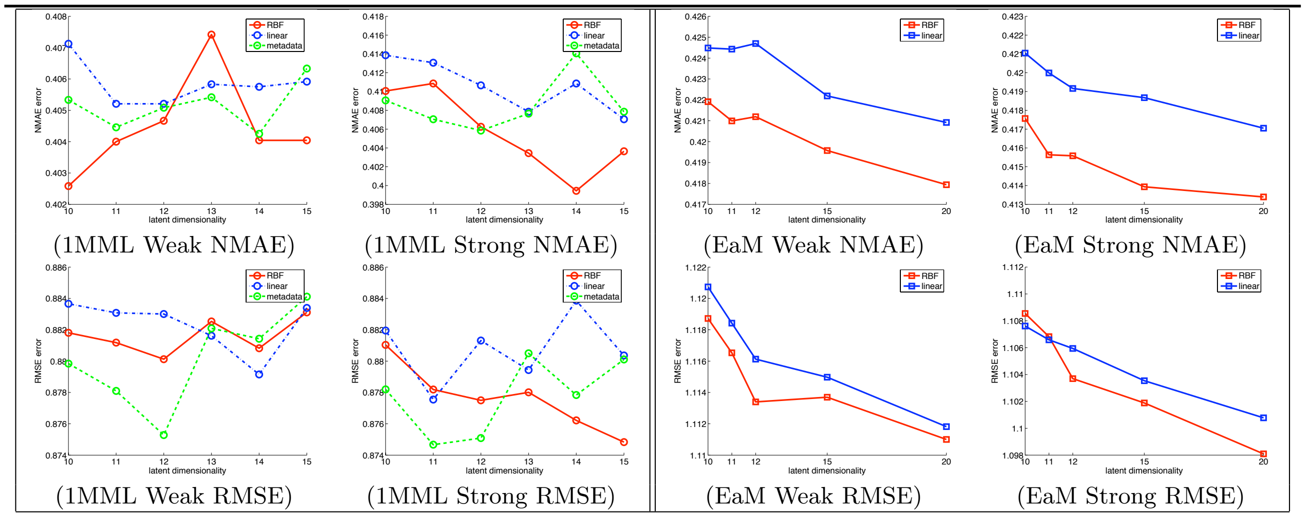

Figure 3. **1M MovieLens (1MML) and EachMovie (EaM):** NMAE and RMSE errors for different kernels. Note that in general non-linear latent spaces result in better performance.

4.4. Larger database

10M MovieLens We also perform experiments on a larger data set. The 10M MovieLens data set consists of 10 million ratings for 71,567 users and 10,681 movies, with ratings ranging $\{1, 2, \cdots, 5\}$. We use the r_a and r_b partitions provided with the database, that split the data into a training and testing, so that they are 10 ratings per user in the test set. This database was made available on 9th January 2009, so there are currently no other published results to compare with. results. Our approach gave an RMSE of ($\mathbf{0.8740} \pm 0.0278$) using a 10 dimensional latent space.

4.5. Adding Movie Meta-Data

The backbone of the GP-LVM is a Gaussian process regression model, in which we are optimizing with respect to input values to maximize the likelihood of the data. For the case of movies, there is additional data about the movie that we might want to include, for example with the 1M MovieLens data, there is information about the genre of the movie (*e.g.*, comedy and western). This can be encoded in a binary vector that defines the genre for each movie in the database. We can include this information in the kernel matrix. If the meta-data for a particular movie is given in a vector $\mathbf{m}_{i,:}$ then a covariance function can be created from the meta-data,

$$k_m\left(\mathbf{m}_{i,:}, \mathbf{m}_{j,:}\right) = \alpha_m \exp\left(-\frac{\gamma_m}{2}||\mathbf{m}_i - \mathbf{m}_j||^2\right).$$

This can be combined with the covariance function defined in $\mathbf{x}$-space through a tensor product,

$$k_{i,j} = k_m\left(\mathbf{m}_{i,:}, \mathbf{m}_{j,:}\right) k_x\left(\mathbf{x}_{i,:}, \mathbf{x}_{j,:}\right).$$

We now optimize the parameters of the meta-data covariance function along with the $\mathbf{X}$ and the parameters of the latent data covariance function.

The dashed green line in Figure 3 shows the NMAE and RMSE when using meta-data. No significant improvement in performance is seen with the meta-data, but conversely it hasn't significantly harmed the performance. In practice the use of meta-data may be important when, for example, a new movie is released and ratings data is not yet available for it.

5. Conclusions

We have shown how probabilistic matrix factorization is equivalent to Bayesian probabilistic PCA. This inspired a new algorithm which allowed for non-linear extensions in the manner of the Gaussian process latent variable model. The predictions from the model have a very similar form to neighborhood based methods for collaborative filtering. We tested the resulting algorithm on two popular movie databases. Our approach outperformed all existing published results for these data. We briefly considered extensions to the model involving incorporating side information (meta-data) about movie genre. We didn't see a significant difference in the performance given meta-data, but speculate that such data may be useful for new movies with few ratings.

As well as pushing forward the boundary of the state-the-art performance for the data sets we've studied, our model provides probabilistic predictions. An item/user dependent variance can be computed giving the confidence with which the model is prepared to make the predictions.

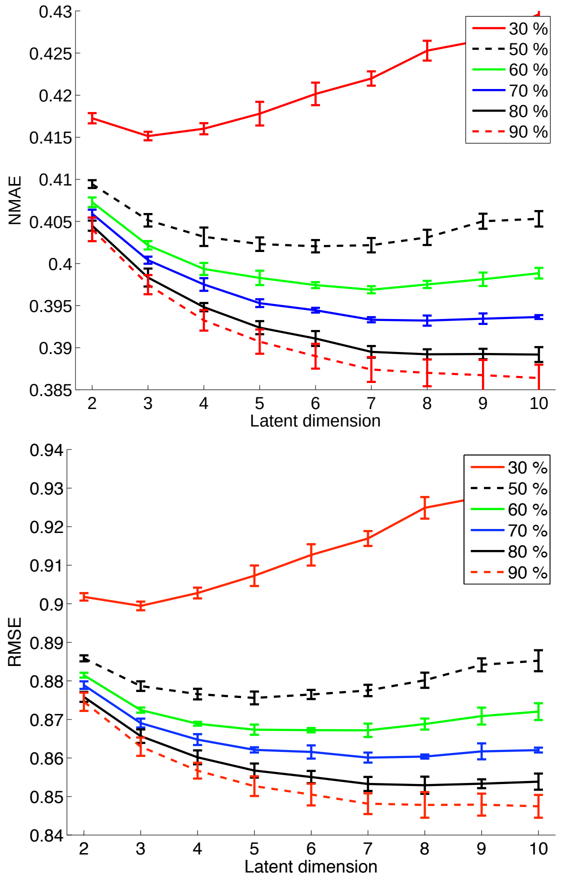

Figure 2. **1M MovieLens:** NMAE and RMSE errors as a function of the latent space dimensionality for different percentages of the database used as training, i.e., 30-90 %.

References

Bell, R., & Koren, Y. (2007). Lessons from the Netflix prize challenge. *SIGKDD Explorations 9*, 75–79.

Bell, R. M., Koren, Y., & Volinsky, C. (2008). The BellKor 2008 solution to the netflix prize. Available from `http://www.research.att.com/~volinsky/netflix/`.

Bishop, C. M. (1999a). Bayesian PCA. *Advances in Neural Information Processing Systems* (pp. 482–388). Cambridge, MA: MIT Press.

Bishop, C. M. (1999b). Variational principal components. *Proceedings Ninth International Conference on Artificial Neural Networks, ICANN'99* (pp. 509–514).

DeCoste, D. (2006). Collaborative prediction using ensembles of maximum margin matrix factorization. *Proceedings of the International Conference in Machine Learning* (pp. 249–256). Omnipress.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 426–434).

Lawrence, N. D. (2005). Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, *6*, 1783–1816.

Marlin, B. (2004a). Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto.

Marlin, B. (2004b). Modeling user rating profiles for collaborative filtering. *Advances in Neural Information Processing Systems* (pp. 627–634). Cambridge, MA: MIT Press.

Minka, T. P. (2001). Automatic choice of dimensionality for PCA. *Advances in Neural Information Processing Systems* (pp. 598–604). Cambridge, MA: MIT Press.

Park, S.-T., & Pennock, D. M. (2007). Applying collaborative filtering techniques to movie search for better ranking and browsing. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Cambridge, MA: MIT Press.

Rennie, J. D. M., & Srebro, N. (2005). Fast maximum margin factorization for collaborative prediction. *Proceedings of the International Conference in Machine Learning* (pp. 713–719).

Salakhutdinov, R., & Mnih, A. (2008a). Bayesian probabilistic matrix factorization using MCMC. *Proceedings of the International Conference in Machine Learning* (pp. 872–879). Omnipress.

Salakhutdinov, R., & Mnih, A. (2008b). Probabilistic matrix factorization. *Advances in Neural Information Processing Systems* (pp. 1257–1264). Cambridge, MA: MIT Press.

Schölkopf, B., & Smola, A. J. (2001). *Learning with kernels*. Cambridge, MA: MIT Press.

Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society, B*, *6*, 611–622.