need not stop here: for instance, there may be coherent co-
variation even within individual clusters. One can capture
this variation by applying Rule 3 to get the Bayesian Clus-
tered Tensor Factorization (BCTF) (Sutskever et al., 2009)
model \((MG + G)(GM^T + G) + G\). This process is shown
in cartoon form in Figure 1.

For an example from vision, consider a matrix \(X\), where
each row is a small (e.g. 12 \(\times\) 12) patch sampled from
an image and vectorized. Image patches can be viewed as
lying near a low-dimensional subspace spanned by the low-
est frequency Fourier coefficients (Bossonaier and Sny-
der, 1986). This can be captured by the low-rank model
\(GG + G\). In a landmark paper, Olshausen and Field (1996)
found that image patches are better modeled as a linear
combination of a small number of components drawn from
a larger dictionary. In other words, \(X\) is approximated as
the product \(WA\), where each row of \(A\) is a basis function,
and \(W\) is a sparse matrix giving the linear reconstruction
coefficients for each patch. By fitting this “sparse coding”
model, they obtained a dictionary of oriented edges similar
to the receptive fields of neurons in the primary visual
cortex. If we apply Rule (5), we obtain a Bayesian ver-
sion of sparse coding, \((\exp(G) \circ G)G + G\), similar to
the model proposed by Berkes et al. (2008). Intuitively, the
latent Gaussian coefficients are multiplied elementwise by
“scale” variables to give a heavy-tailed distribution. Many
researchers have designed models to capture the depen-
dencies between these scalar variables, and such “Gaussian
scale mixture” models represent the state-of-the art for low-
level vision tasks such as denoising (Portilla et al., 2003)
and texture synthesis (Portilla and Simoncelli, 2000). One
such GSM model is that of Karklin and Lewicki (2008),
who fit a low-rank model to the scale variables. By apply-
ing Rule (1) to the sparse coding structure, we can represent
their model in our framework as \((\exp(GG + G) \circ G)G + G\).
This model has been successful at capturing higher-level
textural properties of a scene and has properties similar to
complex cells in the primary visual cortex.

Figure 2 gives several additional examples of matrix de-
composition models and highlights the relationships be-
tween them. We emphasize that our goal is not to repro-
duce existing models exactly, but to develop a formalism
powerful enough to express a wide variety of statistical as-
sumptions about the latent factors underlying the data.

We note that many of the above models are not typically
viewed as matrix decomposition structures. Describing
them as such results in a compact notation for defining
them and makes clearer the relationships between the dif-
ferent models. The above examples have in common that
complex models can be derived by incrementally adding
structure to a sequence of simpler models (in a way that
parallels the path researchers took to discover them). This
observation motivates our proposed procedures for infer-
ence and structure learning.

4 Posterior inference of component matrices

Searching over matrix decomposition structures requires a
generic and unified approach for posterior sampling of the
latent matrices. Unfortunately, for most of the structures
we consider, this posterior is complicated and multimodal,
and escaping from local modes requires carefully chosen
special-purpose sampling operators. Engineering such op-
erators for thousands of different models would be undesir-
able.

Fortunately, the compositional nature of our model space
allows us to focus the engineering effort on the relatively
small number of production rules. In particular, observe
that in a realization of the generative process, the value
of an expression depends only on the values of its sub-
expressions. This suggests the following initialization pro-
dure: when applying a production rule \(P\) to a matrix \(S\),
sample from the posterior for \(P\)’s generative model condi-
tioned on it evaluating (exactly) to \(S\). Many of our produc-
tion rules correspond to simple machine learning models
for which researchers have already expended much time
developing efficient inference algorithms:

1. **Low rank.** To apply the rule \(G \rightarrow GG + G\), we fit the
probabilistic matrix factorization (Salakhutdinov and
Mnih, 2008) model using block Gibbs sampling over
the two factors. While PMF assumes a fixed latent
dimension, we choose the dimension automatically by
placing a Poisson prior on the dimension and moving
between states of differing dimension using reversible
jump MCMC (Green, 1995).

2. **Clustering.** To apply the clustering rule to rows:
\(G \rightarrow MG + G\), or to columns: \(G \rightarrow GM^T + G\),
we perform collapsed Gibbs sampling over the cluster
assignments in a Dirichlet process mixture model.

3. **Binary factors.** To apply the rule \(G \rightarrow BG + G\) or
\(G \rightarrow GB^T + G\), we perform accelerated collapsed
Gibbs sampling (Doshi-Velez and Ghahramani, 2006)
on the binary variables in a linear-Gaussian In-
dian Buffet Process (Griffiths and Ghahramani, 2005)
model, using split-merge proposals (Meeds et al.,
2006) to escape local modes.

4. **Markov chains.** The rule \(G \rightarrow CG + G\) is equiv-
alent to estimating the state of a random walk given
noisy observations, which is done using Rauch-Tung-
Striebel (RTS) smoothing.

The remaining production rules begin with a random de-
composition of \(S\). While some of these algorithms in-
volve fitting Bayesian nonparametric models, once the di-
menisonality is chosen, the model is converted to a finite
model of fixed dimensionality (as defined in section 3). The