Christopher Hase
`christopher_hase@g.harvard.edu`
CS281-F17

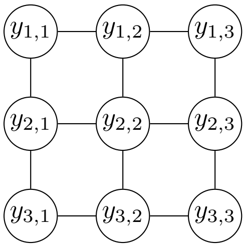
Assignment #4

Due: Monday 5:00pm,
November 13, 2017

Collaborators: None

Graphical Models for Denoising

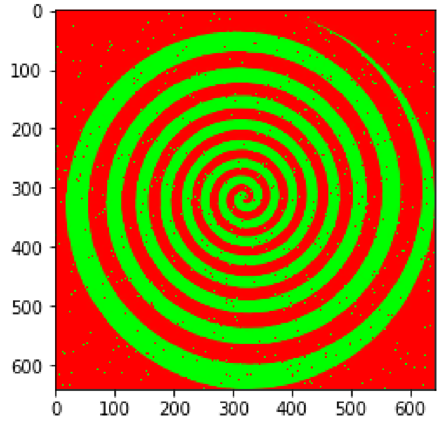
We have seen several variants of the grid-structured Ising model where we have a binary-valued variable at each position of a grid. Here we consider the grid Potts model which has the same graphical model structure, but instead with multiple labels K at each node of the undirected graph $y_{i,j} \in \{1, \dots, K\}$.



In particular consider a conditional Potts model for image denoising. The input x will consist of a picture with pixels x_{ij}, where each pixel is one of K different colors and has been perturbed by random noise. Each random variable y_{ij} represents the color we think each pixel should take after denoising. Unary potentials represent the prior belief for that pixel based on its observed value. Neighbor potentials enforce smoothness of the labeling. Specifically, $\theta(y_{ij} = k) = 10 * \delta(x_{ij} = k)$, and for all neighboring pixels $n \in \{(i-1,j), (i+1,j), (i,j-1), (i,j+1)\}$,

$$\theta(y_{i,j}, y_n) = \begin{cases} 10 & y_{i,j} = y_n \\ 2 & |y_{i,j} - y_n| = 1 \\ 0 & o.w. \end{cases}$$

This is obviously a simplified and discretized view of the true denoising problem, but it gives us a reasonable place to start. As an example consider the problem with $K = 2$ and noise over the image of a spiral.

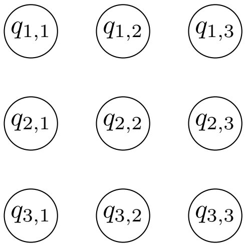


[Note: for the example problems we will show k=1 as red, k=2 as green, and k=3 as blue. We will represent this as the last dimension of the image in a one-hot encoding, so $x[i,j,0] = 1$ for red, $x[i,j,1] = 1$ for green, and $x[i,j,2] = 1$ for blue. Here red is "close" to green and green is close to blue, but red is not close to blue. This is not supposed to be physically true, just part of the problem.]

Problem 1 (Variational Inference for Denoising, 30pts)

For the problems below we have provided a set of example images in the form of numpy arrays including a small sample, a flag, a bullseye, and the large spiral above.

1. First as a sanity check, consider the 3x3 image small with $K = 2$. Compute using brute-force the true posterior marginal probability $p(y_{i,j}|x)$ of any node.
2. Now consider a variational-inference based approach to this problem. Using mean-field factorization, with $q(y)$ factored to each node of the graph, derive local mean field updates.



3. Implement these mean-field updates with a synchronous schedule in PyTorch/numpy. (This means that all parameters are updated with expectations from the previous time step.). Run for 30 epochs starting with q set to a uniform distribution. Graph the results on the small images and compare to the brute-force approach. Compare the variational values to the exact marginals for the small example. Note: running on the spiral example will likely require a fast/matrix implementation.
4. Implement Loopy Belief Propagation with a synchronous or non-synchronous schedule in PyTorch/Numpy following the algorithm given in Murphy (Section 22.2.2). Run for 30 epochs using the starting conditions in in Algorithm 22.1. Compare to the mean field approach.
5. (Optional) Write out the Integer Linear Programming formulation for the maximum assignment problem. What is the advantage of mean field compared to the ILP approach?
6. (Optional) Install the PuLP toolkit in python. Implement the ILP formulation for this problem. Compare your solution for the smaller images.

1.1. These are the posterior marginals for $y_{ij} = 1 \; \forall \; i, j$ for the small image:

```
[[ 1.          1.          1.        ]
 [ 0.99999998  1.          0.99999998]
 [ 0.00252378  0.00248464  0.00252378]]
```

These are the posterior marginals for $y_{ij} = 2 \; \forall \; i, j$ for the small image:

```
[[  5.79905571e-12   1.95516261e-15   5.79905571e-12]
 [  1.51930154e-08   6.56942344e-12   1.51930154e-08]
 [  9.97476218e-01   9.97515359e-01   9.97476218e-01]]
```

Note that there are some issues with rounding here.

1.2. Let l be the epoch number. I am assuming we have $K = 3$ colors for this part. We use a mean-field factorization to approximate the joint posterior $p(\mathbf{y}|\mathbf{x})$, and the mean field updates can be written as:

$$q_{ij}^{(l+1)}(y_{ij}) = \frac{\exp\left(\theta(y_{ij}) + \sum\limits_{n\in \text{nbr}_{ij}} \theta(y_{ij},1)q_n^{(l)}(1) + \theta(y_{ij},2)q_n^{(l)}(2) + \theta(y_{ij},3)q_n^{(l)}(3)\right)}{\sum\limits_{k=1}^{3} \exp\left(\theta(k) + \sum\limits_{n\in \text{nbr}_{ij}} \theta(k,1)q_n^{(l)}(1) + \theta(k,2)q_n^{(l)}(2) + \theta(k,3)q_n^{(l)}(3)\right)}$$

We need to make this update $\forall\ y_{ij} \in \{1,2,3\}$ and $\forall\ i,j$.

1.3. For the comparison of the denoised images to the originals, see the next page.

Here are the variational values for the posterior marginals for $y_{ij} = 1\ \forall\ i,j$ for the small image:

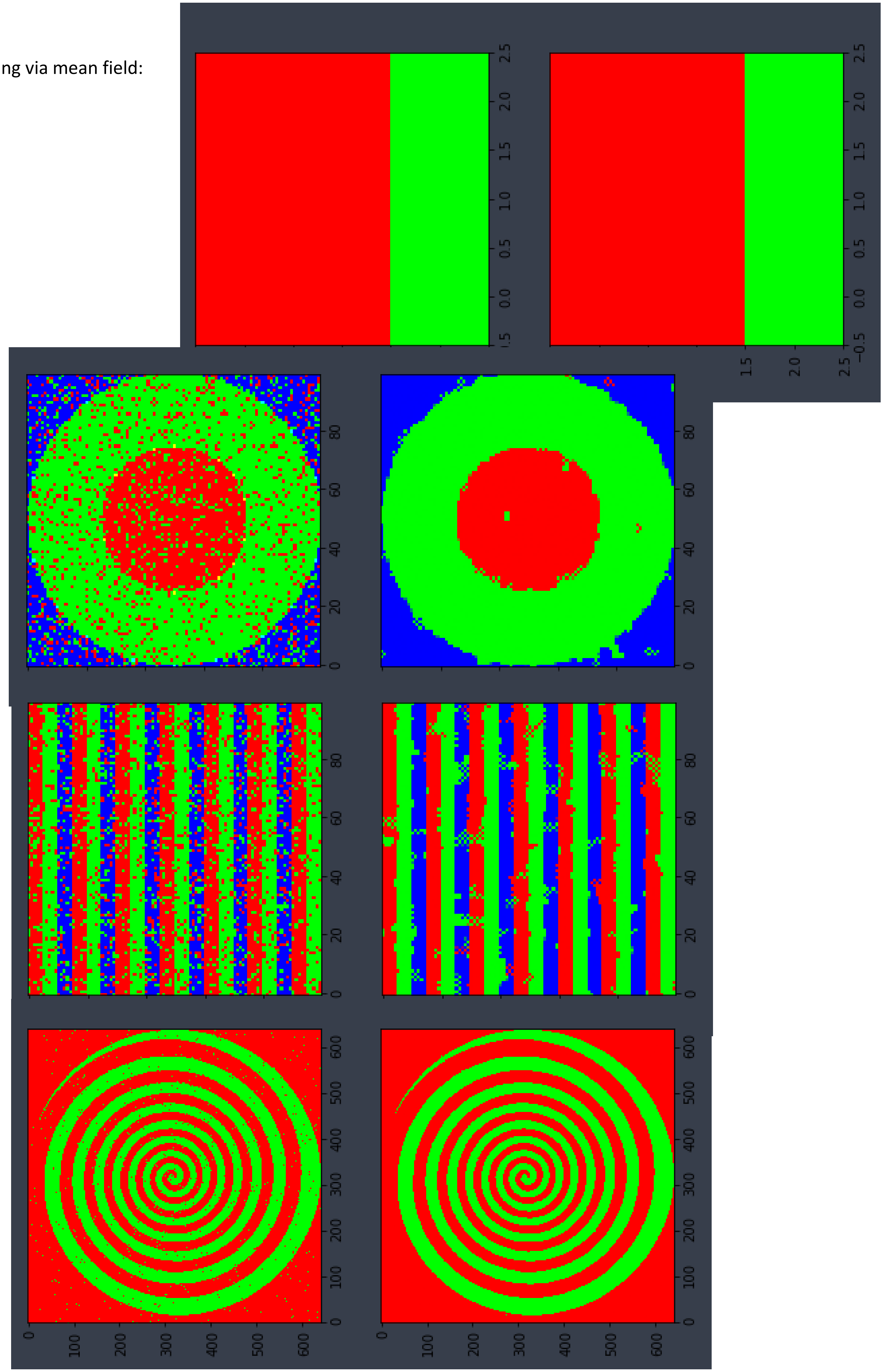
```
[[  1.00000000e+00   1.00000000e+00   1.00000000e+00]
 [  9.99999985e-01   1.00000000e+00   9.99999985e-01]
 [  4.53978686e-05   1.52521210e-08   4.53978686e-05]]
```

Here are the variational values for the posterior marginals for $y_{ij} = 2\ \forall\ i,j$ for the small image:

```
[[  5.10909027e-12   1.71390843e-15   5.10909027e-12]
 [  1.52189208e-08   5.10909027e-12   1.52189208e-08]
 [  9.99954600e-01   9.99999985e-01   9.99954600e-01]]
```

Note that there are some issues with rounding here. The the variational values are close to the exact marginals.

Denoising via mean field:



Modeling users and jokes with a Bayesian latent bilinear model

The next two questions will develop Bayesian inference methods for the simplest version of the latent bilinear model you used to model jokes ratings in HW3. The data set we'll use is the same as in HW3, a modified and preprocessed variant of the Jester data set. However, to make things easier (and to make being Bayesian more worthwhile) **we'll only use subsampling to 10% of the training data**. The other ratings will form your test set.

The model

The model is the same as in HW3, but with Gaussian priors on the latent parameter matrices U and V. Let $r_{i,j} \in \{1, 2, 3, 4, 5\}$ be the rating of user i on joke j. A latent linear model introduces a vector $u_i \in \mathbb{R}^K$ for each user and a vector $v_j \in \mathbb{R}^K$ for each joke. Then, each rating is modeled as a noisy version of the appropriate inner product. Specifically,

$$r_{i,j} \sim \mathcal{N}(u_i^T v_j, \sigma_\epsilon^2).$$

Fix σ_ϵ^2 to be 1.0, and start with $K = 2$. We put independent Gaussian priors on each element of U and V:

$$U_{i,k} \sim \mathcal{N}(0, \sigma_U^2 = 5)$$

$$V_{i,k} \sim \mathcal{N}(0, \sigma_V^2 = 5)$$

Problem 2 (Stochastic Variational Inference, 30pts)

Recall that variational inference optimizes a lower bound on the log marginal likelihood (integrating out parameters θ), like so:

$$\begin{aligned}\log p(x) &= \log \int p(x,\theta)d\theta = \log \int p(x|\theta)p(\theta)d\theta && (1)\\ &= \log \int \frac{q_\lambda(\theta)}{q_\lambda(\theta)} p(x|\theta)p(\theta)d\theta = \log \mathbb{E}_{q_\lambda} \frac{1}{q(\theta)} p(x|\theta)p(\theta)d\theta && (2)\\ &\geq \mathbb{E}_{q_\lambda} \log \left[\frac{1}{q_\lambda(\theta)} p(x|\theta)p(\theta) \right] = \underbrace{-\mathbb{E}_{q_\lambda} \log q_\lambda(\theta)}_{\text{entropy}} + \underbrace{\mathbb{E}_{q_\lambda} \log p(\theta)}_{\text{prior}} + \underbrace{\mathbb{E}_{q_\lambda} \log p(x|\theta)}_{\text{likelihood}} = \mathcal{L}(\lambda) && (3)\end{aligned}$$

In this case, $\theta = U, V$ and $x = R$:

$$\mathcal{L}(\lambda) = -\mathbb{E}_{q_\lambda} \log q_\lambda(U,V) + \mathbb{E}_{q_\lambda} \log p(U,V) + \sum_{n=1}^{N} \mathbb{E}_{q_\lambda} \log p(r_n|U,V) \quad (4)$$

This is a general formula that works for many different priors, likelihoods and variational approximations. Here we will keep things simple and choose $q(U,V)$ to be a Gaussian factorized over every single entry of each matrix for U and V, e.g. the same form as the prior. Thus our variational parameters will consist of a mean and variance for each entry in U and V: $\lambda_{ik}^{(\mu U)}$, $\lambda_{ik}^{(\sigma^2 U)}$, $\lambda_{jk}^{(\mu V)}$, and $\lambda_{jk}^{(\sigma^2 V)}$.

1. Derive the expression for the KL divergence between two univariate Gaussians.

2. Exploiting the conditional independence of the model, we can write the variational objective (which we want to maximize) as:

   $$\mathcal{L}(\lambda) = -KL(q_\lambda(U) \mid\mid p(U)) - KL(q_\lambda(V) \mid\mid p(V)) + \sum_{n=1}^{N} \mathbb{E}_{q_\lambda} \log p(r_n|U,V)$$

   Simplify the first two terms of this model to get a closed form expression.

3. The third term is the likelihood of the data under an expectation wrt the variational parameters. Assume that we approximate this term using a single sample of rows $\tilde{u}_i$ and $\tilde{v}_j$ for each rating $r_{i,j}$. Write out the full objective with this approximation for the last term.

4. Unfortunately this is difficult to optimize, since the sampled variables depend on the variational parameters λ. An alternative method, known as *reparameterization*, replaces expectation of the form $\mathbb{E}_{X\sim\mathcal{N}(\mu,\sigma)}[f(X)]$, in terms of $\mathbb{E}_{Z\sim\mathcal{N}(0,1)}[f(Z\sigma+\mu)]$. Rewrite the objective in this form using sampled dummy variables $\tilde{z}$ (and no $\tilde{u}_i$ or $\tilde{v}_j$).

5. Using PyTorch, set up this model using `nn.Embedding` for the variational parameters. For numerical stability, store the log of the variance in the embedding table, and also initialize this table with very low values, e.g. `logvar.weight.data = -1000`. For $K = 2$, optimize the variational parameters for 10 epochs over the sampled data. Use Adam with learning rate 0.001.

   Plot the training and test-set log-likelihood as a function of the number of epochs, as well as the marginal likelihood lower bound. That is to say: at the end of each epoch, evaluate the log of the average predictive probability of all ratings in the training and test sets using 100 samples from q(U,V). The lower bound is the sum of entropy, prior and likelihood terms, while the training-set and test-set likelihoods only use the likelihood term.

6. Fit your variational model for $K = 1$ to $K = 10$, and plot the training-set log-likelihood, test-set log-likelihood, and lower bound for each value of K. How do the shapes of these curves differ?

2.1. $X \sim \mathcal{N}(\mu, \sigma^2)$ has PDF p.
q is a univariate Gaussian PDF parameterized by mean θ and variance τ^2.

$$KL(p||q) = \int_{-\infty}^{\infty} \log\left(\frac{p(x)}{q(x)}\right) p(x)dx$$

$$= \int_{-\infty}^{\infty} \log\big(p(x)\big)p(x)dx - \int_{-\infty}^{\infty} \log\big(q(x)\big)p(x)dx$$

Focusing on the first term for now, we have:

$$\int_{-\infty}^{\infty} \log\big(p(x)\big)p(x)dx = \int_{-\infty}^{\infty} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)\right)p(x)dx$$

$$= -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\int_{-\infty}^{\infty}(x^2 - 2x\mu + \mu^2)p(x)dx$$

$$= -\frac{1}{2}\log(2\pi\sigma^2) - \frac{E(X^2) - 2\mu E(X) + \mu^2}{2\sigma^2}$$

$$= -\frac{1}{2}\log(2\pi\sigma^2) - \frac{\sigma^2 + \mu^2 - 2\mu^2 + \mu^2}{2\sigma^2}$$

$$= -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2}$$

Now we focus on the second term:

$$\int_{-\infty}^{\infty} \log\big(q(x)\big)p(x)dx = \int_{-\infty}^{\infty} \log\left(\frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\frac{1}{2\tau^2}(x-\theta)^2\right)\right)p(x)dx$$

$$= -\frac{1}{2}\log(2\pi\tau^2) - \frac{1}{2\tau^2}\int_{-\infty}^{\infty}(x^2 - 2x\theta + \theta^2)p(x)dx$$

$$= -\frac{1}{2}\log(2\pi\tau^2) - \frac{E(X^2) - 2\theta E(X) + \theta^2}{2\tau^2}$$

$$= -\frac{1}{2}\log(2\pi\tau^2) - \frac{\sigma^2 + \mu^2 - 2\theta\mu + \theta^2}{2\tau^2}$$

$$= -\frac{1}{2}\log(2\pi\tau^2) - \frac{\sigma^2 + (\mu - \theta)^2}{2\tau^2}$$

Putting the two terms together, we have:

$$KL(p||q) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2} + \frac{1}{2}\log(2\pi\tau^2) + \frac{\sigma^2 + (\mu-\theta)^2}{2\tau^2}$$

$$= \frac{1}{2}\left(\log\left(\frac{\tau^2}{\sigma^2}\right) + \frac{\sigma^2 + (\mu - \theta)^2}{\tau^2} - 1 \right)$$

2.2. Let N be the number of users.
Let M be the number of jokes.
Let K be the number of latent parameters for each user/joke.
Let B be the set of all (user, joke) pairs in the training data for which there are ratings. Then $|B|$ is the number of ratings in the training data.
Let $U \in \mathbb{R}^{N \times K}$ be the random latent parameter matrix for the users.
Let $V \in \mathbb{R}^{M \times K}$ be the random latent parameter matrix for the jokes.
$q_{U_{ik}}$ is a univariate Gaussian PDF parameterized by mean $\lambda_{ik}^{(\mu U)}$ and variance $\lambda_{ik}^{(\sigma^2 U)}$ $\forall\ i, k$. These are our approximations to the posteriors of the entries in U.
$q_{V_{jk}}$ is a univariate Gaussian PDF parameterized by mean $\lambda_{jk}^{(\mu V)}$ and variance $\lambda_{ik}^{(\sigma^2 V)}$ $\forall\ j, k$. These are our approximations to the posteriors of the entries in V.
$p_{U_{ik}}$ is a univariate Gaussian PDF parameterized by mean 0 and variance σ_U^2 $\forall\ i, k$. These are the priors of the entries in U.
$p_{V_{jk}}$ is a univariate Gaussian PDF parameterized by mean 0 and variance σ_V^2 $\forall\ j, k$. These are the priors of the entries in V.

Let $\boldsymbol{\lambda}$ be the set of all variational parameters.

$$q_U(U) = \prod_{i=1}^{N} \prod_{k=1}^{K} q_{U_{ik}}(U_{ik})$$
$$p_U(U) = \prod_{i=1}^{N} \prod_{k=1}^{K} p_{U_{ik}}(U_{ik})$$
$$q_V(V) = \prod_{j=1}^{M} \prod_{k=1}^{K} q_{V_{jk}}(V_{jk})$$
$$p_V(V) = \prod_{j=1}^{M} \prod_{k=1}^{K} p_{V_{jk}}(V_{jk})$$
$$q_{UV}(U, V) = q_U(U) \cdot q_V(V)$$
$$p_{UV}(U, V) = p_U(U) \cdot p_V(V)$$

We want to simplify the first two terms in the variational objective $\mathcal{L}(\boldsymbol{\lambda})$.

$$KL\big(q_U(U) || p_U(U)\big) = E_{q_U}\left(\log\left(\frac{q_U(U)}{p_U(U)} \right) \right)$$

$$= E_{q_U}\left(\log\left(\frac{\prod_{i=1}^{N} \prod_{k=1}^{K} q_{U_{ik}}(U_{ik})}{\prod_{i=1}^{N} \prod_{k=1}^{K} p_{U_{ik}}(U_{ik})} \right) \right)$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} E_{q_{U_{ik}}}\left(\log\left(\frac{q_{U_{ik}}(U_{ik})}{p_{U_{ik}}(U_{ik})} \right) \right)$$

$$= \sum_{i=1}^{N}\sum_{k=1}^{K} KL(q_{U_{ik}}(U_{ik})||p_{U_{ik}}(U_{ik})$$

Here we see that we have a sum of KL divergences between univariate Gaussians, which we know the expressions for from part 2.1. Thus, we have:

$$\sum_{i=1}^{N}\sum_{k=1}^{K} KL(q_{U_{ik}}(U_{ik})||p_{U_{ik}}(U_{ik}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}\left(\log\left(\frac{\sigma_U^2}{\lambda_{ik}^{(\sigma^2 U)}}\right) + \frac{\lambda_{ik}^{(\sigma^2 U)} + \left(\lambda_{ik}^{(\mu U)}\right)^2}{\sigma_U^2} - 1\right).$$

We see an analogous result for $KL\big(q_V(V)||p_V(V)\big)$. That is, we have:

$$KL\big(q_V(V)||p_V(V)\big) = \frac{1}{2}\sum_{j=1}^{M}\sum_{k=1}^{K}\left(\log\left(\frac{\sigma_V^2}{\lambda_{jk}^{(\sigma^2 V)}}\right) + \frac{\lambda_{jk}^{(\sigma^2 V)} + \left(\lambda_{jk}^{(\mu V)}\right)^2}{\sigma_V^2} - 1\right)$$

2.3. The final term in $\mathcal{L}(\boldsymbol{\lambda})$ can be written as:

$$\sum_{(i,j)\in B} E_{q_{UV}}\Big(\log\big(\mathcal{N}(r_{ij}|U_i^T V_j, \sigma_\epsilon^2)\big)\Big)$$

According to the problem statement, we are going to approximate this with:

$$\sum_{(i,j)\in B} \log\big(\mathcal{N}(r_{ij}|\tilde{u}_i^T\tilde{v}_j, \sigma_\epsilon^2)\big)$$

$$= -\frac{|B|}{2}\log(2\pi\sigma_\epsilon^2) - \frac{1}{2\sigma_\epsilon^2}\sum_{(i,j)\in B}(r_{ij} - \tilde{u}_i^T\tilde{v}_j)^2$$

Now we can write our full objective as:

$$\mathcal{L}(\boldsymbol{\lambda}) \approx -\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}\left(\log\left(\frac{\sigma_U^2}{\lambda_{ik}^{(\sigma^2 U)}}\right) + \frac{\lambda_{ik}^{(\sigma^2 U)} + \left(\lambda_{ik}^{(\mu U)}\right)^2}{\sigma_U^2} - 1\right)$$

$$-\frac{1}{2}\sum_{j=1}^{M}\sum_{k=1}^{K}\left(\log\left(\frac{\sigma_V^2}{\lambda_{jk}^{(\sigma^2 V)}}\right) + \frac{\lambda_{jk}^{(\sigma^2 V)} + \left(\lambda_{jk}^{(\mu V)}\right)^2}{\sigma_V^2} - 1\right)$$

$$-\frac{|B|}{2}\log(2\pi\sigma_\epsilon^2) - \frac{1}{2\sigma_\epsilon^2}\sum_{(i,j)\in B}(r_{ij} - \tilde{u}_i^T\tilde{v}_j)^2$$

2.4. Note that $\lambda_{ik}^{(\sigma U)} = \sqrt{\lambda_{ik}^{(\sigma^2 U)}}$ and $\lambda_{jk}^{(\sigma V)} = \sqrt{\lambda_{jk}^{(\sigma^2 V)}}$

$Z \sim \mathcal{N}(0,1)$

$U_{ik} = Z\lambda_{ik}^{(\sigma U)} + \lambda_{ik}^{(\mu U)} \ \forall \ i, k$

$V_{jk} = Z\lambda_{jk}^{(\sigma V)} + \lambda_{jk}^{(\mu V)} \ \forall \ j, k$

We will use the notations $\tilde{z}_{U_{ik}}$ and $\tilde{z}_{V_{jk}}$ for the purposes of distinguishing samples that were drawn from the distribution of Z.

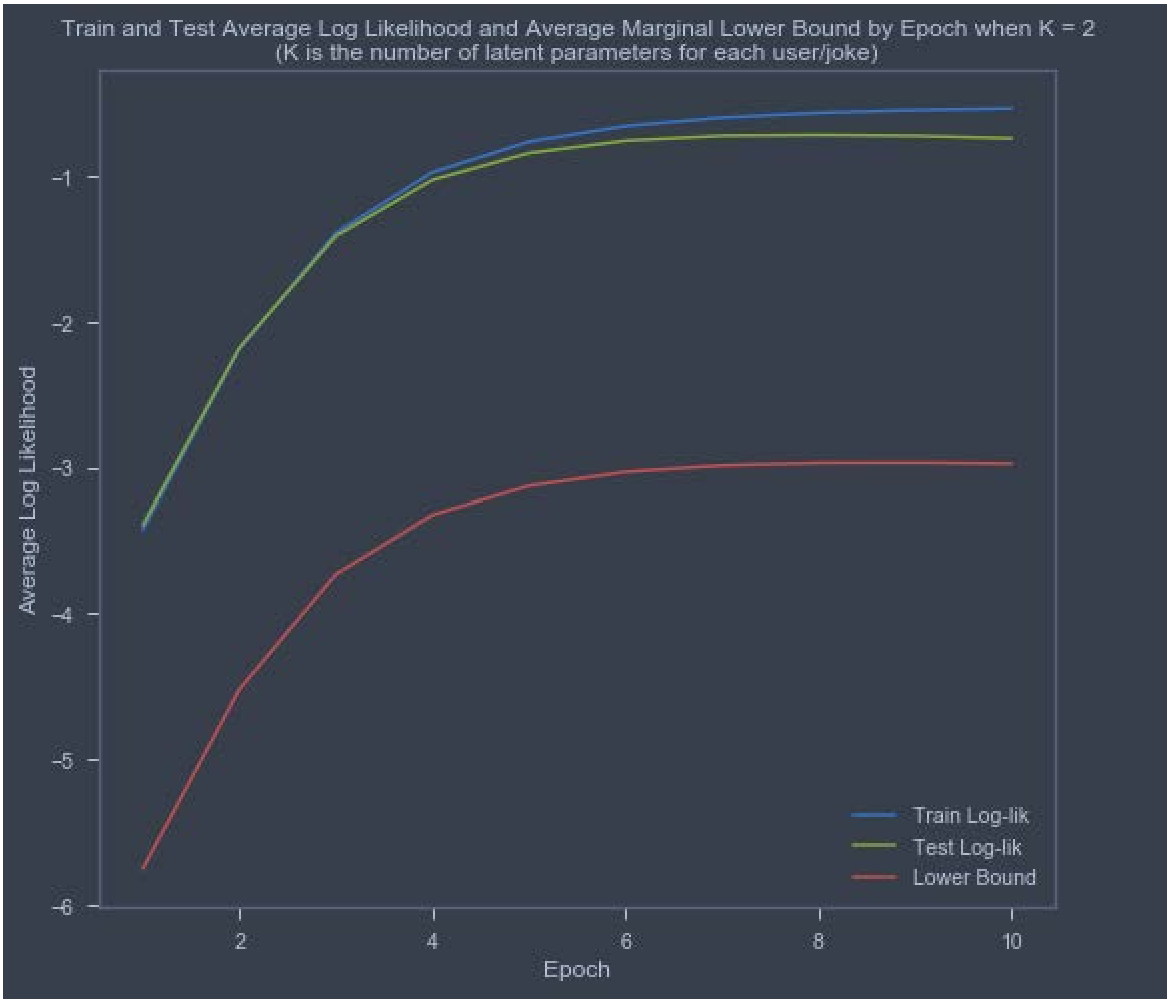
We can re-write our full objective with alterations to the third term to reflect our reparameterization as:

$$\mathcal{L}(\boldsymbol{\lambda}) \approx -\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}\left(\log\left(\frac{\sigma_U^2}{\lambda_{ik}^{(\sigma^2 U)}}\right) + \frac{\lambda_{ik}^{(\sigma^2 U)} + \left(\lambda_{ik}^{(\mu U)}\right)^2}{\sigma_U^2} - 1\right)$$

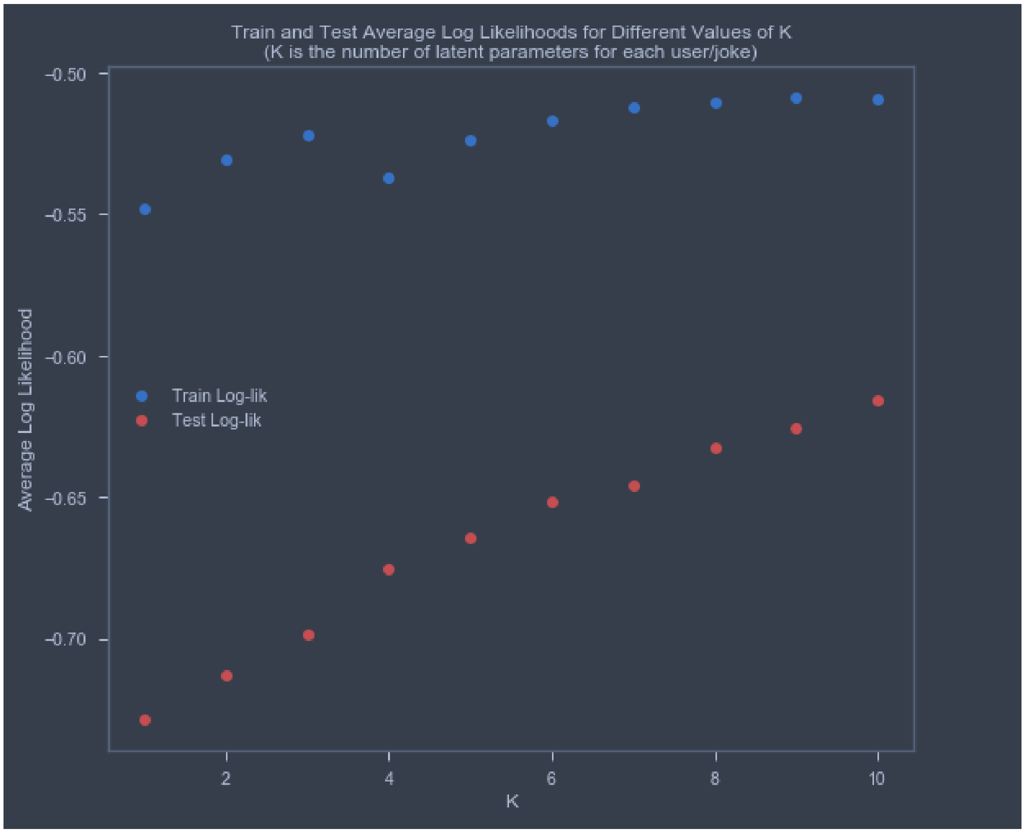
$$-\frac{1}{2}\sum_{j=1}^{M}\sum_{k=1}^{K}\left(\log\left(\frac{\sigma_V^2}{\lambda_{jk}^{(\sigma^2 V)}}\right) + \frac{\lambda_{jk}^{(\sigma^2 V)} + \left(\lambda_{jk}^{(\mu V)}\right)^2}{\sigma_V^2} - 1\right)$$

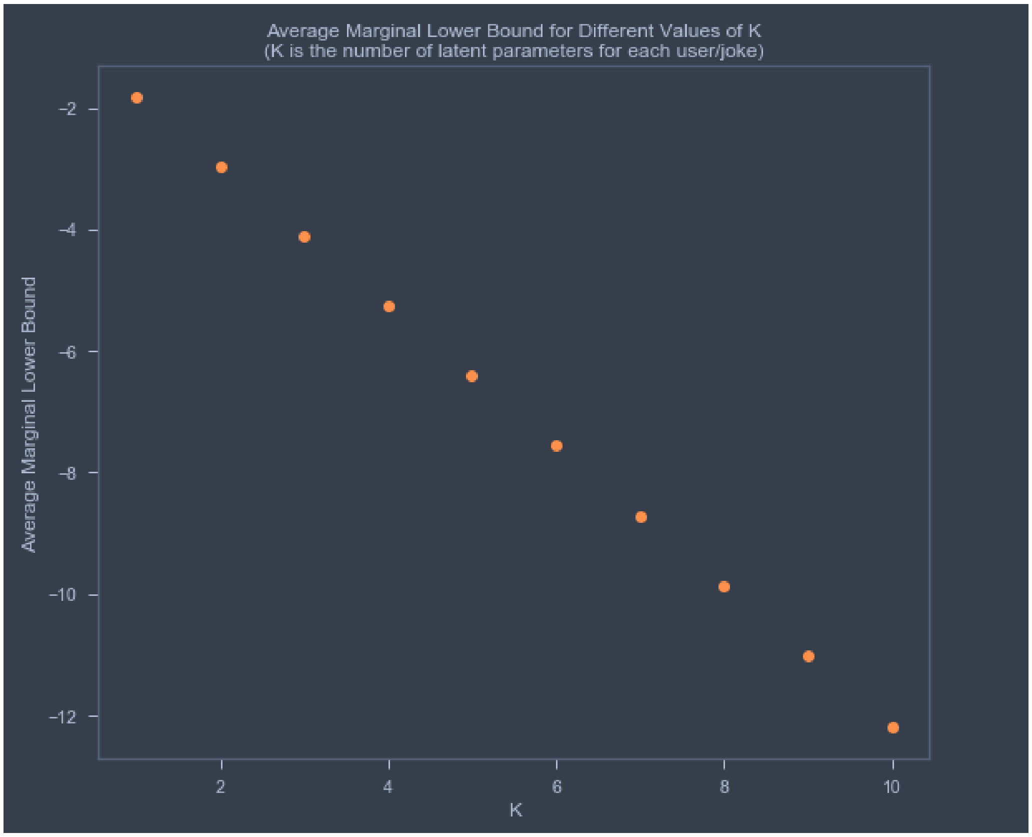
$$-\frac{|B|}{2}\log(2\pi\sigma_\epsilon^2) - \frac{1}{2\sigma_\epsilon^2}\sum_{(i,j)\in B}\left(r_{ij} - \sum_{k=1}^{K}\left(\tilde{z}_{U_{ik}}\lambda_{ik}^{(\sigma U)} + \lambda_{ik}^{(\mu U)}\right)\left(\tilde{z}_{V_{jk}}\lambda_{jk}^{(\sigma V)} + \lambda_{jk}^{(\mu V)}\right)\right)^2$$

2.5.


Train and Test Average Log Likelihood and Average Marginal Lower Bound by Epoch when K = 2
(K is the number of latent parameters for each user/joke)
Average Log Likelihood
-1
-2
-3
-4
-5
-6
Epoch
2
4
6
8
10
Train Log-lik
Test Log-lik
Lower Bound

2.6. The shapes of the train and test average log-likelihood curves are pretty much the same-- the log-likelihoods increase as K increases. The average marginal likelihood lower bound decreases as K increases. This makes sense since increasing K increases the number of latent parameters to learn for each user and each joke.





Problem 3 (Gibbs Sampling, 25pts)

.

In this problem we will consider a different sampling-based approach for estimating the posterior.

1. Write down the conditional equations for U and V. That is to say, write their conditional distributions, conditioned on all the other variables as well as the training data:

$$p(U_i \mid V, R)$$

$$p(V_j \mid U, R)$$

   Because the model is bi-linear, these updates should have fairly simple forms. Here, we mean U_i to mean the latent parameters corresponding to the ith user, and V_j to mean those for the jth joke.

2. A Gibbs sampler is an alternative model for computing the posteriors of intractable models. The method works by repeatedly alternating between drawing samples of U conditioned on V, and then samples of V conditioned on U. (We will derive in greater detail in coming classes).

   Give the pseudocode for running this algorithm using the posterior equations from above.

3. Run the Gibbs sampler for 100 steps (i.e. update both U and V 100 times). Plot the training and test-set log-likelihood as a function of the number of steps through your training set. That is, use all previous samples of U, V to evaluate the predictive probability of all ratings.

4. One reason to be Bayesian is that you don't have to worry about overfitting. Run your Gibbs sampler for $K = 1$ to $K = 10$, and plot the training and test-set log-likelihood for each value of K. How do the shapes of these curves differ from the curves you saw when doing maximum likelihood estimation in HW3?

3.1. Let $\mathbf{U}_i \sim \mathcal{N}(\mathbf{0}, \sigma_U^2\mathbf{I})$ with $\mathbf{U}_i \in \mathbb{R}^K$ be the prior for the latent parameters for user i $\forall$ $i \in \{1, \ldots, N\}$.
Let $\mathbf{V}_j \sim \mathcal{N}(\mathbf{0}, \sigma_V^2\mathbf{I})$ with $\mathbf{V}_j \in \mathbb{R}^K$ be the prior for the latent parameters for joke j $\forall$ $j \in \{1, \ldots, M\}$.
Let $\mathbf{U} \in \mathbb{R}^{N\times K}$ be a matrix with $\mathbf{U}_i$ as its ith row $\forall$ i.
Let $\mathbf{V} \in \mathbb{R}^{M\times K}$ be a matrix with $\mathbf{V}_j$ as its jth row $\forall$ j.
Let $\mathbf{V}^{(i)}$ be a matrix with K columns and its rows as the rows in $\mathbf{V}$ corresponding to the jokes rated by user i $\forall$ i.
Let $\mathbf{U}^{(j)}$ be a matrix with K columns and its rows as the rows $\mathbf{U}$ corresponding to the users who rated joke j $\forall$ j.
Let $\mathbf{R} \in \mathbb{R}^{N\times M}$ be a matrix with its ijth entry as R_{ij}, the rating by the ith user for the jth joke, $\forall$ i, j. $R_{ij}|\mathbf{U}_i, \mathbf{V}_j \sim \mathcal{N}(\mathbf{U}_i^T\mathbf{V}_j, \sigma_\epsilon^2)$.
Let $\mathbf{R}^{(i)}$ be the vector of ratings submitted by user i $\forall$ i. The structure of the graphical model and properties of Gaussians allow us to write that $\mathbf{R}^{(i)}|\mathbf{U}_i, \mathbf{V}^{(i)} \sim \mathcal{N}(\mathbf{V}^{(i)}\mathbf{U}_i, \sigma_\epsilon^2\mathbf{I})$.
Let $\mathbf{R}^{(j)'}$ be the vector of ratings submitted by users for joke j $\forall$ j. The structure of the graphical model and properties of Gaussians allow us to write that $\mathbf{R}^{(j)'}|\mathbf{U}^{(j)}, \mathbf{V}_j \sim \mathcal{N}(\mathbf{U}^{(j)}\mathbf{V}_j, \sigma_\epsilon^2\mathbf{I})$.

Now we can write the posteriors for $\mathbf{U}_i$ and $\mathbf{V}_j$.

$p(\mathbf{u}_i|\mathbf{r}^{(i)}, \mathbf{v}^{(i)}) \propto p(\mathbf{r}^{(i)}|\mathbf{u}_i, \mathbf{v}^{(i)}) \cdot p(\mathbf{u}_i|\mathbf{v}^{(i)})$

$= p(\mathbf{r}^{(i)}|\mathbf{u}_i, \mathbf{v}^{(i)}) \cdot p(\mathbf{u}_i)$

$= \mathcal{N}(\mathbf{r}^{(i)}|\mathbf{v}^{(i)}\mathbf{u}_i, \sigma_\epsilon^2\mathbf{I}) \cdot \mathcal{N}(\mathbf{u}_i|\mathbf{0}, \sigma_U^2\mathbf{I})$

From here, using results from The Matrix Cookbook in section 8.1.8, we have the following:

$$\mathbf{U}_i | \mathbf{R}^{(i)}, \mathbf{V}^{(i)} \sim \mathcal{N}(\boldsymbol{\theta}_i, \boldsymbol{\Psi}_i)$$

$$\boldsymbol{\Psi}_i = \left(\frac{1}{\sigma_U^2} \mathbf{I} + \frac{1}{\sigma_\epsilon^2} \mathbf{V}^{(i)T} \mathbf{V}^{(i)} \right)^{-1}$$

$$\boldsymbol{\theta}_i = \frac{1}{\sigma_\epsilon^2} \boldsymbol{\Psi}_i \mathbf{V}^{(i)T} \mathbf{R}^{(i)}$$

$$p(\mathbf{v}_j | \mathbf{r}^{(j)'}, \mathbf{u}^{(j)}) \propto p(\mathbf{r}^{(j)'} | \mathbf{v}_j, \mathbf{u}^{(j)}) \cdot p(\mathbf{v}_j | \mathbf{u}^{(j)})$$

$$= p(\mathbf{r}^{(j)'} | \mathbf{v}_j, \mathbf{u}^{(j)}) \cdot p(\mathbf{v}_j)$$

$$= \mathcal{N}(\mathbf{r}^{(j)'} | \mathbf{u}^{(j)} \mathbf{v}_j, \sigma_\epsilon^2 \mathbf{I}) \cdot \mathcal{N}(\mathbf{v}_j | \mathbf{0}, \sigma_V^2 \mathbf{I})$$

From here, using results from The Matrix Cookbook in section 8.1.8, we have the following:

$$\mathbf{V}_j | \mathbf{R}^{(j)'}, \mathbf{U}^{(j)} \sim \mathcal{N}(\boldsymbol{\alpha}_j, \boldsymbol{\beta}_j)$$

$$\boldsymbol{\beta}_j = \left(\frac{1}{\sigma_V^2} \mathbf{I} + \frac{1}{\sigma_\epsilon^2} \mathbf{U}^{(j)T} \mathbf{U}^{(j)} \right)^{-1}$$

$$\boldsymbol{\alpha}_j = \frac{1}{\sigma_\epsilon^2} \boldsymbol{\beta}_j \mathbf{U}^{(j)T} \mathbf{R}^{(j)'}$$

3.2. Pseudocode:

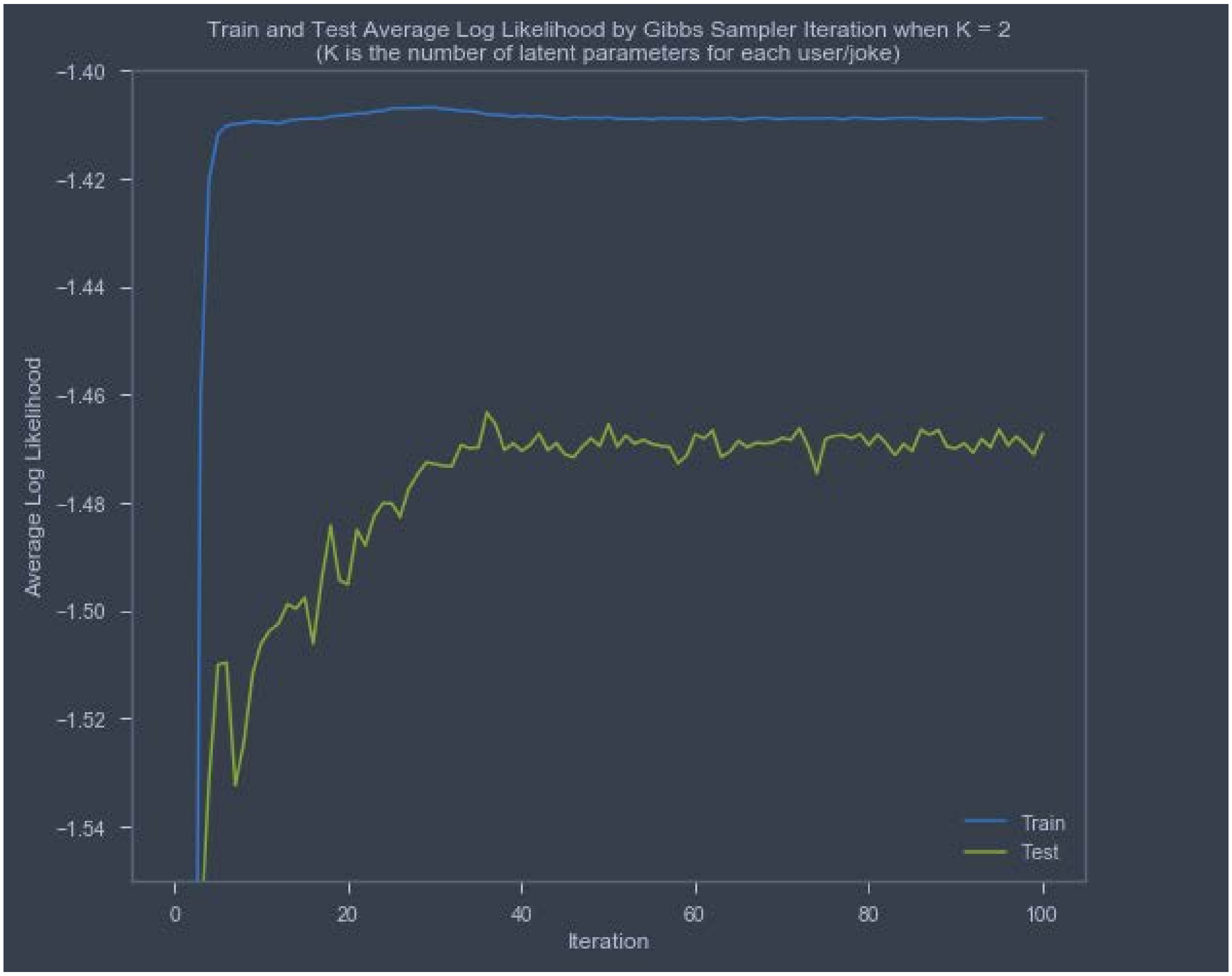
(a) Initialize values for $\mathbf{V}_j$ by sampling once from its prior $\forall\, j$.

Repeat for some number of steps:

(b) Using the values for the samples in the previous step (after step (a), samples come from the posteriors for $\mathbf{V}_j$ $\forall\, j$) and the ratings, sample once from the posterior for $\mathbf{U}_i$ $\forall\, i$.

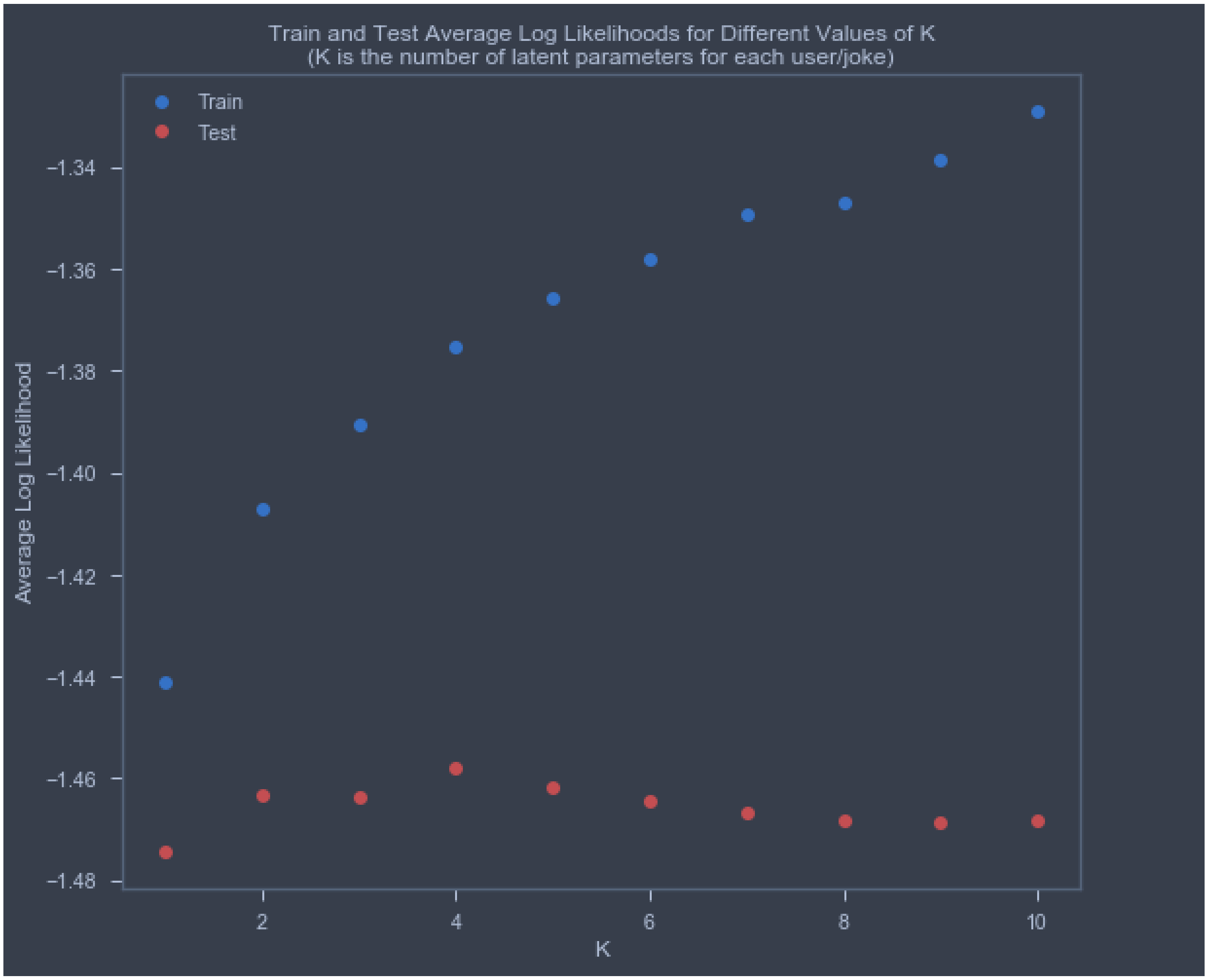
(c) Using the values for the samples in step (b) and the ratings, sample once from the posterior for $\mathbf{V}_j$ $\forall\, j$.

3.3. Note that I plotted average log likelihoods instead of log likelihoods.



3.4. Note that I plotted average log likelihoods instead of log likelihoods.

When doing maximum likelihood estimation in HW3, we saw small improvements in the performance of the model on the training and testing sets as we increased K. Here, when taking a Bayesian approach and using Gibbs sampling, we see small improvements in performance on the training set as we increase K. For the testing set, we see small improvements in performance up through K=4, but then we see small drop-offs in performance as we increase K. These observations are at odds with what we would expect to see when comparing a Bayesian approach to a maximum likelihood approach. However, if we were to increase K further, I think we would be far more concerned about a high degree of overfitting when using the maximum likelihood approach than when using the Bayesian approach with Gibbs sampling.


Train and Test Average Log Likelihoods for Different Values of K
(K is the number of latent parameters for each user/joke)
● Train
● Test
Average Log Likelihood
K