AM 207: Homework 3

Verena Kaynig-Fittkau and Pavlos Protopapas
Due: 11.59 P.M. Thursday March 24th, 2016

Instructions:

- Upload your answers in an ipython notebook to Canvas.
- We will provide you imports for your ipython notebook. Please do not import additional libraries.
- Your individual submissions should use the following filenames: AM207_YOURNAME_HW3.ipynb
- Your code should be in code cells as part of your ipython notebook. Do not use a different language (or format).
- **Do not just send your code. The homework solutions should be in a report style. Be sure to add comments to your code as well as markdown cells where you describe your approach and discuss your results.**
- Please submit your notebook in an executed status, so that we can see all the results you computed. However, we will still run your code and all cells should reproduce the output when executed.
- If you have multiple files (e.g. you've added code files or images) create a tarball for all files in a single file and name it: AM207_YOURNAME_HW3.tar.gz or AM207_YOURNAME_HW3.zip

Have Fun!

In [1]:
```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set_style("white")

import time
import timeit

import scipy.stats
import pandas as pd
import pymc as pm

import re
import numpy as np
```

```
/Users/Grace/anaconda/lib/python2.7/site-packages/matplotlib/font_ma
nager.py:273: UserWarning: Matplotlib is building the font cache usi
ng fc-list. This may take a moment.
  warnings.warn('Matplotlib is building the font cache using fc-lis
t. This may take a moment.')
```

Problem 1: Employee Satisfaction Improvement

This problem is going to explore the differences between complete pooling, unpooling and partial pooling for a normal model with observed standard deviations.

You are working for a consulting firm which is trying to find a good strategy to improve employee satisfaction for their customers. Your company ran pilot studies in eight different customer companies and measured the improvement in employee satisfaction after the plan had been implemented for two years. The data you are given is the mean and standard deviation of the satisfaction improvement, measured by a survey. You follow the nature of your data by modeling the effect of the strategie with a normal model. To simplify things you can assume that the different standard deviations are an effect of different sample sizes, and that there is actually one underlying observation variance:

$$\sigma_j^2 = \frac{\sigma^2}{n_j}$$

Thus the difference in the observed standard deviations in the survey results are caused by having different numbers of survey answers n_j.

Discuss, create and compare three different solutions for your model using PYMC or another sampling method of your choice:

- complete pooling

- unpooling
- partial pooling

In [4]:
```
# Here is the data
data = np.double(np.array([[29.5,18.4],[6.3,12.7],[-3.9,15.9],[7.2,10.2
data_means = data[:,0]
data_std = data[:,1]
```

Problem 2: Motif Finding Using Gibbs

One interesting problem in bioinformatics is that of finding common subsequences of nucleotid bases (these subsequences are called motifs) that repeat themselves within larger DNA sequences. The problem is relevant for genetics because locating the positions of these motifs within the DNA sequence helps in the understanding of how genes are regulated.

Suppose that you are a biologist who is analyzing genetic material collected in a nearby asteroid. The DNA of this extraterrestrial form of life is formed as a sequence of $n_B = 5$ nucleotide bases. Let us label these extraterrestrial nucleotides and put them in a vector $\mathbf{b} = [\mathrm{M}, 0, 2, \mathrm{A}, 7]$. You have a DNA sample $\mathcal{S}$ consisting of $p = 20$ sequences of DNA, each of them with a lenght $l = 200$ nucleotide bases that you can find in this file (HW3/Sequences_new.dat). Each row of the file is a DNA sequence. You are asked to find a motif of $q = 5$ consecutive nucleotides hidden in the background that appears to repeat itself very often in the DNA you were given, but that appears only once in each of the sequences.

Let us formalize the problem. The starting positions of the motif within each sequence (our missing data in the problem) can be represented by the set of random variables:

$$\mathcal{A} = \{a_k, k = 1, \ldots, p\}$$

The motif has to start somewhere within the sequences and so for each sequence $\mathcal{S}_k$:

$$\sum_{i=1}^{l} P(a_k = i) = 1$$

What we are after is the joint distribution $P(\mathcal{A}|\mathcal{S})$ for the motif alignment (i.e., its starting position) being a_k for sequence $\mathcal{S}_k$. As shown in this paper (http://www.cs.cmu.edu/~epxing/Class/10810/readings/liu.pdf), for each sequence $\mathcal{S}_k$ (think of each sequence as a dimension of our parameter space) we can obtain the conditional probability:

$$P(a_k = i|\mathcal{A}_{\hat{k}}, \mathcal{S}) = \frac{1}{Z} \prod_{j=1}^{q} \left(\frac{\hat{\theta}_{\mathbf{j}}}{\hat{\theta}_{\mathbf{0}}} \right)^{s_{i+j}-1}$$

where $\mathcal{A}_{\hat{k}}$ refers to the alignments in all sequences other than S_k, and Z is a normalization factor. Of the other quantities, s_{x} is a vector index for the *x-th* position in the sequence, with the same length as $\hat{\theta}_{\mathbf{j}}$ and $\mathbf{b}$. It takes a value of 1 at the entry that in $\mathbf{b}$ corresponds to the base at position x and 0 for all other entries. Finally, vectors $\hat{\theta}_{\mathbf{j}}$ and $\hat{\theta}_{\mathbf{0}}$ contain respectively the probabilities of observing the q bases at the corresponding position of the current sequence, and the probabilities of finding the same bases in the background. It is important to note that the power, division, and multiplication of vectors in the equation above is performed entry-wise.

Your only task is to design a Gibbs algorithm that samples the joint probability $P(\mathcal{A}|\mathcal{S})$. Your algorithm should converge and provide the alignments of the motif and the secret motif itself. Here are some hints/tasks that should help you and that will help the TF grading your homework:

(a) Describe the equation for the conditional probabilities in your own words, and make sure you understand it before you code anything. Be as explicit as possible.

(b) Start by assigning random starting positions for the q-long motif in all sequences. Then exclude a particular sequence S_k (your current sequence) and use the remaining sequences to construct $\mathbf{\Theta}$ as a probability matrix from counting the number of times that the *i-th* base appears in position j of the motif given the current alignments. Construct also a vector $\hat{\theta}_{\mathbf{0}}$ from counting the number of times that the *i-th* base appears in the background. Note that the vectors $\hat{\theta}_{\mathbf{j}}$ are the columns of matrix $\mathbf{\Theta}$.

(c) From $\mathbf{\Theta}$ and $\theta_{\mathbf{0}}$, derive the conditional probability over all possible alignments for your current sequence and draw a sample from it. This sample will be your updated alignment for the current sequence.

(d) Iterate over all p sequences. Such iteration over sequences is only one Gibbs iteration. After enough Gibbs iterations you should start noticing that the algorithm has converged.

Problem 3: Optimizing Hand Luggage

You are going on a trip and have to optimize your hand luggage, but thanks to your cheap flight ticket, the weight of the hand lugagge is restricted and you are sure the airline will enforce the upper limit. You have a set of presents that you want to bring to the relatives you are visiting, but you have to notice that they don't all fit into your suitcase.

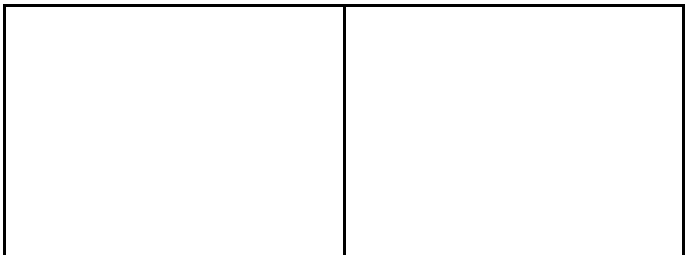
This problem is also called the knapsack problem: given a set of items, each has its weight and value, determine which items should be included into your suitcase, so that the total weight does not exceed some value W and the total value is maximal. The kind of the knapsack problem when each item can be included into the collection at most once is called the 0/1 knapsack problem. Your task is to solve this problem using simulated annealing.

Implement simulated annealing to solve this problem with the list of items below. Which ones would you pick for your suitcase? Plot and discuss your optimization scheme and results. Compare your solution to a greedy algorithm, which sorts the items by the ratio $\left|\frac{v_i}{w_i}\right|$ and puts them into the suitcase consecutively.

In [5]:
```
# here is your data:

#number_of_data_points
N = 100

#total_weight_limit
W = 3000

# weight of all the different items
w = np.array([  38,  236,  909,   73,  768,  906,  716,  646,  848,  96
        130,  973,  584,  750,  509,  391,  282,  179,  277,  255,  358
        915,  469,  908,  253,  491,  669,  926,  399,  563,  581,  216
        984,  754,  504,  479,  865,   87,  142,  394,    8,  320,  830
        535,  314,  514,  897,  317,  210,  265,  729,  654,  628,  432
        634,  457,  543,   72,  388,  455,  918,  562,  314,  516,  965
        793,  498,   44,  589,   27,  821,  337,  622,  884,  298,  467
         16,   65,  197,   26,  368,  739,  472,  904,  283,  666,  617
         23,  778,  708, 1000,  127,  280,  382,  357,  156,  934,  314
        596])

# value of all different items
v = np.array([36, 38, 30, 32, 40, 45, 45, 37, 49, 40, 44, 30, 31, 47, 4
       43, 36, 50, 36, 32, 42, 41, 37, 43, 38, 41, 42, 41, 50, 34, 37,
       43, 34, 46, 48, 30, 43, 40, 47, 37, 40, 50, 30, 42, 31, 39, 48,
       31, 32, 42, 37, 32, 40, 30, 39, 48, 36, 32, 37, 37, 46, 45, 35,
       40, 50, 46, 35, 43, 47, 48, 31, 50, 40, 30, 37, 30, 49, 47, 44,
       50, 50, 41, 36, 43, 45, 39, 32, 37, 35, 34, 35, 38, 43, 47])
```

Problem 4: Confusing Classifications

You are a graduate student conducting research in image processing. You want to test out your latest algorithm, which you decide to call "Ultra-Multilayer Hierarchical Super Convolutionary Neural Network." However, in order to test out your algorithm, you need a sizable training data set. Luckly, your advisor has generously given you over 10 GB of over one million stock photos of cats and dogs. Unfortunately, none of the images are labelled. Fantastic! You enjoy looking at photos of puppies and kittens in your spare time anyways so you decide to dedicate this entire weekend to labeling all of them.




However, after spending two hours looking over hundreds of images, these puppies and kittens are no longer looking so cute. In fact, you are starting to get disgusted at the idea of looking at another picture of these furry creatures.

At this time, you decide to tap into the "power of the crowd" by farming out the labeling task to the workers of Amazon Mechanical Turk (MTurk). You decide to hire 3 MTurk workers. However, you're not sure if these workers are reliable. So, how can you quantify the competency of the workers? In this problem you will develop a model to access the general difficulty of labeling the images of your two classes.

Classifying items in general can be hard even for humans. Some items just look alike, even if they actually are from two different classes. The difficulty in categorizing items of a specific class in relation to other classes is summarized by a confusion matrix (https://en.wikipedia.org/wiki/Confusion_matrix):

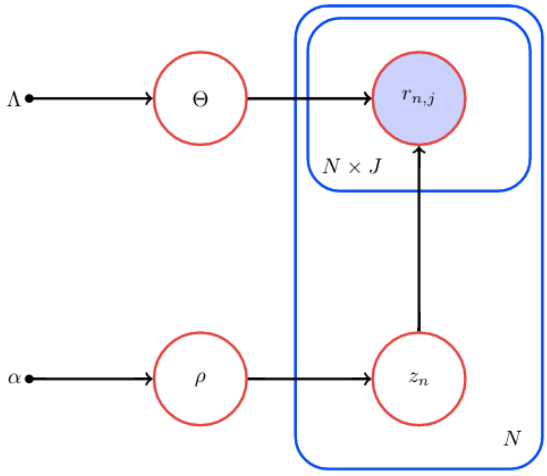
$$\Theta = \begin{pmatrix} 1.0 & 0.0 \\ 0.5 & 0.5 \end{pmatrix}$$

This confusion matrix indicates that items of class 1 are very easy to classify, and always correctly labeled, whereas items of class 2 are very hard to classify and labels are basically just random guesses between the two possibilities.

The models we discussed so far in homework and lecture were pretty low in dimensions. This problem is going to show you that the number of dimensions can grow very fast for some models, making a good sampling strategy crucial. We will only be able to explore a very minimal version of the problem, because of our limited computational resources, but it should be immediately clear how this model would scale for a greater number of classes, workers, and/or items.

Develop a Bayesian model that takes as input a set of (possible erroneous) item labelings and infer the underlying confusion matrix and the true label each data point. To make the model manageable by your laptop, use only one underlying confusion matrix (in principle different people could have different difficulties) two classes for the labels (the confusion matrix is 2x2), and a maximum of 150 data points.

The model you are after is displayed in the following dependency graph:



$$\rho \sim Dir(\alpha)$$

$$z_i \sim Multinomial(\rho)$$

$$\Lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$$

$$\Theta_{(k,:)} \sim Dir(\lambda_k)$$

$$r_{i,j} \sim Multinomial(\Theta_{(z_i,:)}),\ \forall j \in \{1, \ldots, J\}$$

You're given reports generated by the 3 workers, so $r_{i,1}$ is one label for item r_i and $r_{i,2}$ is another label given from another worker. Note that if the confusion matrix is not the identity matrix then these labels can be different because the workers make mistakes. The arrows in the diagram indicate dependence. So the labelings you observe are dependent on the confusion matrix Θ and the underlying true label z_i for each item. N is the number of data points you have and as described above you should use $N \leq 150$.

You can see from the diagram that the model uses Multinomials with Dirichlet priors.

1. Start by describing the model in terms of these distributions, what they mean and what this arrangement means for the form of the labels z_i. Discuss the meaning and influence of the hyperparameter α on the true labels of the data.
2. Discuss your selection of the hyperparameter Λ and how that influences the confusion matrix Θ.
3. Implement this Bayesian model and sample from the posterior to recover the underlying confusion matrix Θ and the distribution of the true labels ρ. Note: Θ is shared by all three workers.

In [6]:
```
# Here is your data

reports = np.load("HW3/reports.npy")
```