

15780: GRADUATE AI (SPRING 2017)

Homework 3: Probabilistic Modeling and Game Theory  
(Solutions)

Release: March 29, 2017,  
Due: April 12, 2017

# 1 Maximum Likelihood Estimation [30 points]

## 1.1 MLE for Uniform Distribution with Two Parameters [10 points]

Given a collection of observed (independent) data points  $X = \{x^{(1)}, \dots, x^{(m)}\}$  from a uniform distribution  $U(\alpha, \beta)$ , where

$$p(x) = \begin{cases} \frac{1}{\beta - \alpha}, & \text{if } \alpha \leq x \leq \beta \\ 0, & \text{otherwise,} \end{cases}$$

derive the maximum likelihood estimators of  $\alpha$  and  $\beta$ , which maximize the probability of observing  $X$ .

**Solution:** If any of the data points is not in  $[\alpha, \beta]$  (i.e.,  $\exists x \in X$  s.t.  $x \notin [\alpha, \beta]$ ), then the probability of observing  $X$  is 0. If all the data points are within  $[\alpha, \beta]$ , then the probability of observing  $X$  is  $\left(\frac{1}{\beta - \alpha}\right)^m$ . To maximize this probability, we want to minimize  $\beta - \alpha$  while keeping all the data points in  $[\alpha, \beta]$  (i.e.,  $\forall x \in X, x \in [\alpha, \beta]$ ). That is, we want to find the minimum interval including all the data points. The interval  $[\min(X), \max(X)]$  is such an interval. This is because, if  $\alpha > \min(X)$  or  $\beta < \max(X)$ , then there exists  $x \in X$  such that  $x \notin [\alpha, \beta]$ . Thus, the maximum likelihood estimators of  $\alpha$  and  $\beta$  are  $\hat{\alpha} = \min(X)$  and  $\hat{\beta} = \max(X)$ .

## 1.2 MLE for Uniform Distribution with One Parameter [10 points]

Given a collection of observed (independent) data points  $X = \{x^{(1)}, \dots, x^{(m)}\}$  from a uniform distribution  $U(-2\alpha, \alpha)$ , derive the maximum likelihood estimators of  $\alpha$ , which maximizes the probability of observing  $X$ .

**Solution:** If any of the data points is not in  $[-2\alpha, \alpha]$  (i.e.,  $\exists x \in X$  s.t.  $x \notin [-2\alpha, \alpha]$ ), then the probability of observing  $X$  is 0. If all the data points are within  $[-2\alpha, \alpha]$ , then the probability of observing  $X$  is  $\left(\frac{1}{3\alpha}\right)^m$ . To maximize this probability, we want to find the minimum  $\alpha$  such that all the data points are in  $[-2\alpha, \alpha]$ . All the data points are in  $[-2\alpha, \alpha]$  if and only if  $\alpha \geq \frac{-\min(X)}{2}$  ( $\Leftrightarrow -2\alpha \leq \min(X)$ ) and  $\alpha \geq \max(X)$ . Thus, the maximum likelihood estimators of  $\alpha$  is  $\hat{\alpha} = \max(\max(X), \frac{-\min(X)}{2})$ .

## 1.3 MLE for Uniform Distribution with Prior [10 points]

Given a collection of observed (independent) data points  $X = \{x^{(1)}, \dots, x^{(m)}\}$  from a uniform distribution  $U(0, e^\alpha)$  where  $\alpha$  follows a prior distribution

$$p(\alpha) \propto e^{-\alpha^2},$$

derive the estimator of  $\alpha$  that maximizes the posteriori probability  $p(\alpha|X)$ . (**Hint:** use  $p(\alpha|X) \propto p(X|\alpha)p(\alpha)$ ).

**Solution:** If any of the data points is not in  $[0, e^\alpha]$  (i.e.,  $\exists x \in X$  s.t.  $x \notin [0, e^\alpha]$ ), then  $p(X|\alpha) = 0$ , and from the hint,  $p(\alpha|X) = 0$ . If all the data points are within  $[0, e^\alpha]$ , then

$$p(\alpha|X) \propto p(X|\alpha)p(\alpha) \propto \left(\frac{1}{e^\alpha}\right)^m e^{-\alpha^2} = e^{-\alpha^2 - m\alpha}. \quad (1)$$

Since  $\exp(x)$  is a monotonically increasing function, we want to find  $\alpha$  maximizing  $f(\alpha) = -\alpha^2 - m\alpha$ , while keeping all the data points in  $[0, e^\alpha]$ . Note that  $f(\alpha)$  is a quadratic function maximized at  $\alpha = -m/2$ .

If  $\max(X) \leq e^{-m/2}$ , then  $\alpha = -m/2$  maximizes  $f(\alpha)$  while satisfying  $\forall x \in X, x \in [0, e^\alpha]$ . If  $\max(X) > e^{-m/2}$ , then  $\alpha = \log(\max(X))$  maximizes  $f(\alpha)$  while satisfying  $\forall x \in X, x \in [0, e^\alpha]$ . Hence,

$$\hat{\alpha} = \begin{cases} -m/2, & \text{if } \max(X) \leq e^{-m/2} \\ \log(\max(X)), & \text{otherwise,} \end{cases}$$

is the estimator of  $\alpha$  maximizing  $p(\alpha|X)$ .

## 2 Equilibria [30 points]

### 2.1 Iterated Elimination of Strictly Dominated Strategies [15 points]

One method of simplifying the search for Nash equilibria is through the iterated elimination of strictly dominated strategies. We say that a player's pure strategy  $s'_i$  is strictly dominated by another pure  $s_i$  if  $\forall s_{-i} \in S_{-i}, u_i(s'_i, s_{-i}) < u_i(s_i, s_{-i})$ . In other words, if there exist two pure strategies,  $s_1$  and  $s_2$ , for player  $i$  such that no matter the (possibly mixed) strategies of the other players it is always in player  $i$ 's interest to play  $s_1$  over  $s_2$ , then  $s_1$  dominates  $s_2$ .

The iterated elimination of strictly dominated strategies proceeds by eliminating one strictly dominated strategy per round. It then iteratively continues until there are no more dominated strategies to eliminate. For example, iterated elimination of strictly dominated strategies on the following game proceeds as follows.

	North	East	South	West
Top	2,3	1,-1	4,0	3,-3
Middle	7,2	-2,0	5,2	6,7
Bottom	8,2	0,1	6,-1	4,0

- Column eliminates East, as playing North is strictly better.
- Row eliminates Top, as playing either Middle or Bottom is strictly better now that Column has eliminated East.
- Column eliminates South, as playing West is strictly better now.
- No more strategies can be eliminated; this leaves Row: [Middle, Bottom] and Column: [North, West] as the surviving strategies.

Prove that if iterated elimination of dominated strategies eliminates all but one of the strategies of each player, then there is a unique Nash equilibrium in the game. You may find Nash's Theorem useful here.

**Solution:** First, we prove that iterative elimination of strictly dominated strategies never removes an action that is in the support of any Nash equilibrium.

AFSOC that an action  $s_i^*$  in a Nash equilibrium  $s^*$  is removed, and, furthermore, that it is the first action in any Nash equilibrium that is removed. This means that there exists some  $s_i$  such that  $u_i(s_i, s'_{-i}) > u_i(s_i^*, s'_{-i}), \forall s'_{-i} \in S_{-i}$  at this time. However, we know that because  $s_i^*$  is the first action in any Nash equilibrium to be removed, the rest of the actions in the support of  $s^*$  are still in  $S_{-i}$  at this time, which means that  $u_i(s_i, s_{-i}^*) > u_i(s_i^*, s_{-i}^*)$ , which contradicts our assumption that  $s^*$  is a Nash equilibrium.

Now, by the hint and Nash's theorem, we are done: because there must exist at least one Nash equilibrium, and because IESDS never removes an action that is in the support of any Nash equilibrium, then this must be the unique Nash equilibrium in the game.

## 2.2 Correlated Equilibria [15 points]

Let  $p_1, \dots, p_n$  be probability distributions representing  $n$  correlated equilibria in the same 2-player game. Prove that any convex combination of  $p_1, \dots, p_n$  is also a correlated equilibrium.

**Solution:** Let's consider the utilities of the first player. The argument is symmetric for the second player. Let  $p = \sum_i a_i p_i$  be a convex combination of  $p_1, \dots, p_n$ . Since  $p_i$  is a correlated equilibrium (CE), for all  $s'_1 \in S$ ,

$$\begin{aligned} \sum_{s_2 \in A} p_i(s_1, s_2) u_1(s_1, s_2) &\geq \sum_{s_2 \in A} p_i(s_1, s_2) u(s'_1, s_2) \\ \Rightarrow \sum_{s_2 \in A} a_i p_i(s_1, s_2) u_1(s_1, s_2) &\geq \sum_{s_2 \in A} a_i p_i(s_1, s_2) u(s'_1, s_2) \end{aligned}$$

By summing the previous inequality over all distributions  $p_1, \dots, p_n$ , we obtain

$$\begin{aligned} \sum_{s_2 \in A} p(s_1, s_2) u_1(s_1, s_2) &= \sum_{s_2 \in A} \left( \sum_{i=1}^n a_i p_i(s_1, s_2) \right) u_1(s_1, s_2) \\ &= \sum_{i=1}^n \sum_{s_2 \in A} a_i p_i(s_1, s_2) u_1(s_1, s_2) \\ &\geq \sum_{i=1}^n \sum_{s_2 \in A} a_i p_i(s_1, s_2) u_1(s'_1, s_2) \\ &= \sum_{s_2 \in A} \left( \sum_{i=1}^n a_i p_i(s_1, s_2) \right) u_1(s'_1, s_2) \\ &= \sum_{s_2 \in A} p(s_1, s_2) u_1(s'_1, s_2) \end{aligned}$$

This shows that  $p$  is a CE, as desired.

## 3 Metropolis-Hastings Implementation [10 points]

**This portion is not graded on autolab.**

We will implement the Metropolis-Hastings algorithms to generate samples from the unnormalized distribution

$$\tilde{p}(x) = e^{-x^2} + 1.3e^{-(x-2)^2}$$

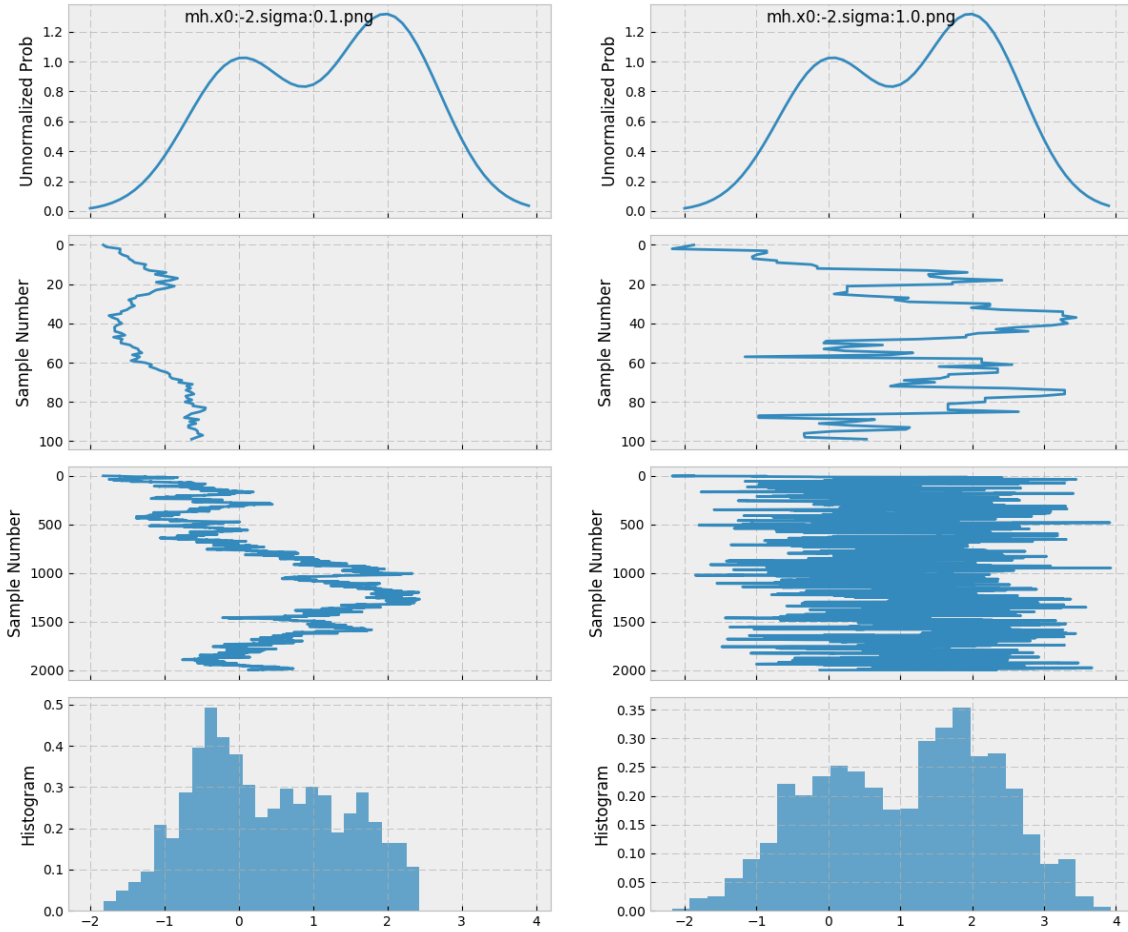
with a Gaussian sampling function with the centered at the previous point with some standard deviation  $\sigma$ .

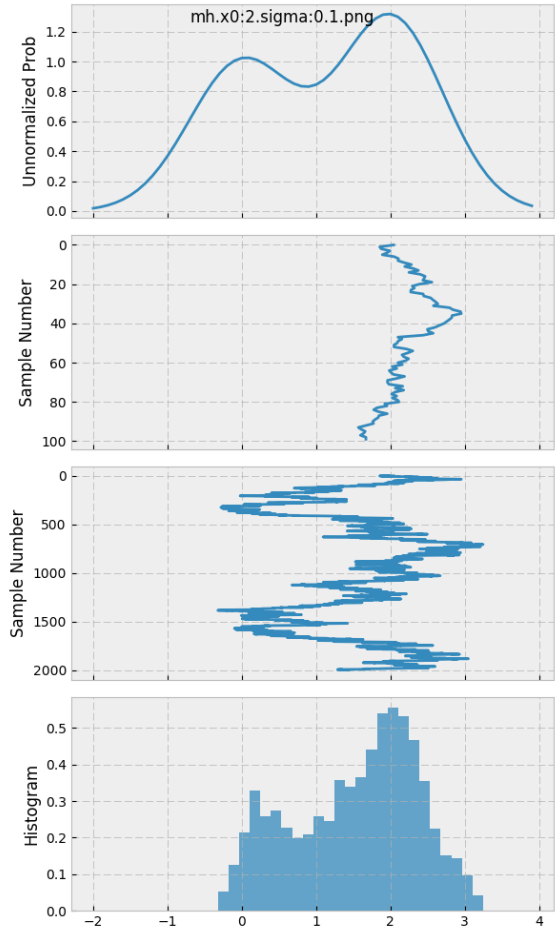
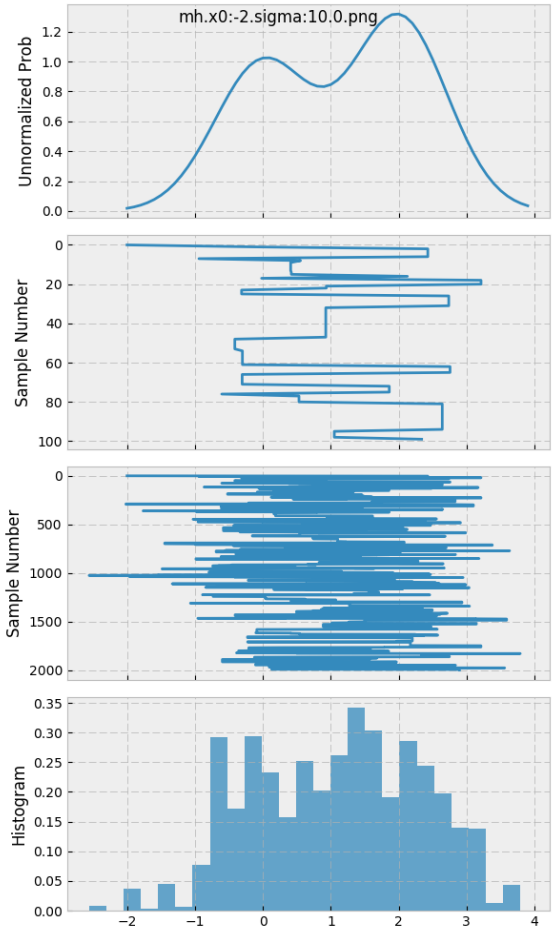
- (5 points) Run your implementation for every combination of  $x_0 \in \{-2, 2\}$  and  $\sigma \in \{0.1, 1.0, 10\}$ . Include plots (1. samples over time and 2. a histogram) for all of these in your writeup.
- (3 points) How and why does  $\sigma$  impact the samples? For all three values, you should state what happens and why it happens.
- (2 points) What is the impact of  $x_0$  on the samples?

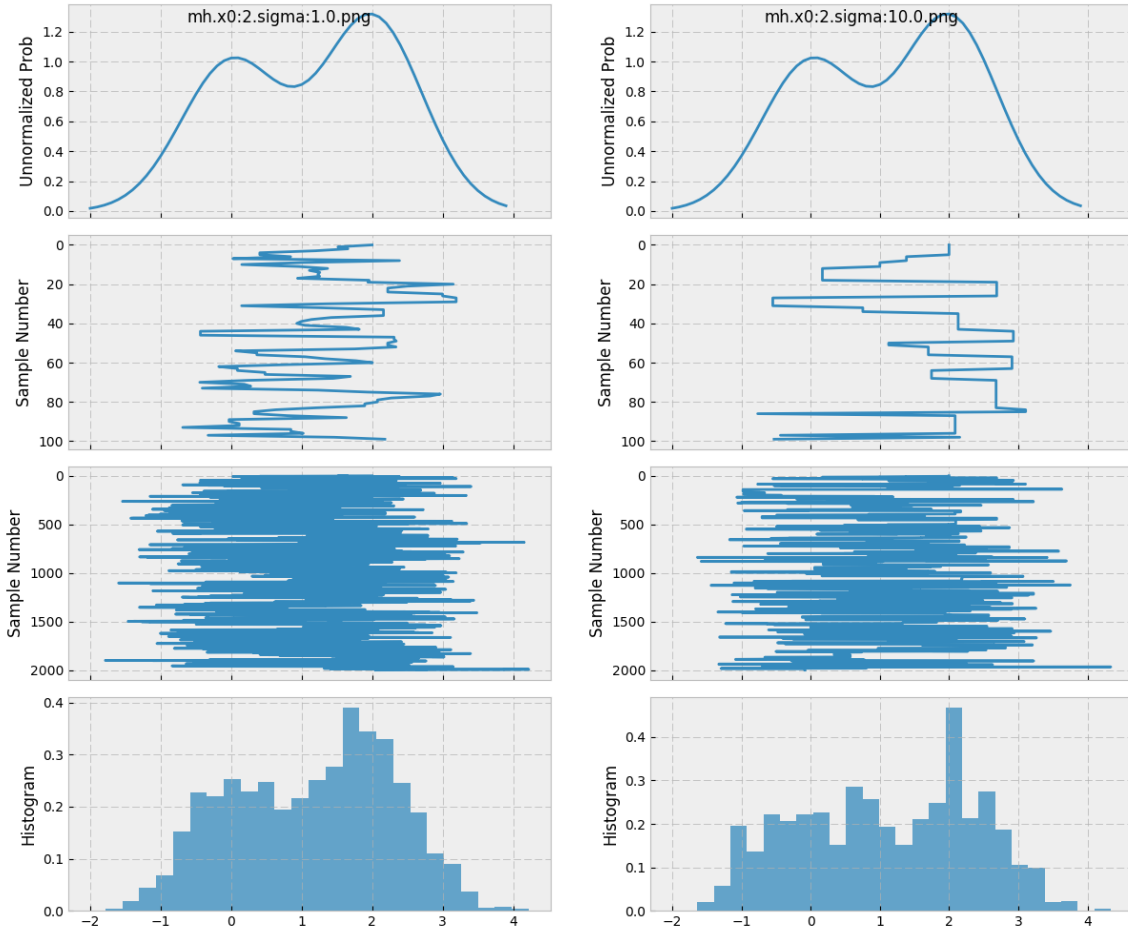
### 3.1 Getting started

Download the handout source code and edit the `mh.py` source file. This file contains a `main` method that will run your implementation and generate sampling plots and histograms. In this file, finish the Metropolis-Hastings implementation in the `mh` function from the description on page 22 of Slides 16 (Probabilistic Modeling III: MCMC).

**Solution:**







## 4 Gibbs Sampling [30 points]

For this problem you will implement the Gibbs sampling algorithm to sample values for the variables in a Bayesian network given a set of evidence variables. In particular, you will implement the following function, in the included file `gibbs.py`:

```
def gibbs(vars, cpts, evidence, n_steps=1000):
```

This function takes as input:

- a dictionary mapping each variable in the Bayes net to the set of values it can take on,
- a list of conditional probability tables (CPTs) for each variable, which are instances of the `CPT` class (more details on this shortly),
- a dictionary mapping each variable we observe (evidence variable) to the observed value, and

- (optionally) the number of steps we want to run Gibbs sampling for—at each step, we sample a new value for a single variable that is not in the evidence set.

The function should return a dictionary mapping every variable to its sampled (including the evidence variables).

The basis for the Bayes net representation we will use is the `CPT` class included in the `gibbs.py` file. You won't have to edit this class at all, or even necessarily understand all the code in this class, but you will need to understand how to use the class. This class describes a conditional probability table for a particular variable in the Bayes net. Each `CPT` object contains 1) a Python dictionary that maps each variable in the CPT to a list of its possible values, and 2) a dictionary mapping each possible instantiation of the variables to the conditional probability of the variable of interest given the remaining variables (i.e., given all of its parents). Let's look at a simple example: if we initialize `CPT` by the following:

```
c = CPT({"x1": [0, 1], "x2": ["TRUE", "FALSE"], "x3": ["RED", "BLUE", "GREEN"]})
```

this will create a factor of three variables, “x1”, “x2”, and “x3,” where the first variable can take on values 0 or 1, the second can be “TRUE” or “FALSE”, and the third can take on values “RED”, “GREEN”, or “BLUE” (note that in this problem, unlike the lecture notes, variables can take on more than two values, but this really introduces no added complexity). Note that there is no requirement that the list of possible values for each factor be numbers: indeed, for the real-world Bayes net example, these will typically be lists of strings, but this should not change your code at all. If you want to access this list of variables at any point, use the class member `c.variables`.

To get the probability of a variable `v` given all its parents and children, you can “call” the `CPT` object by passing in the variables as keyword arguments. For example, you can write

```
c(v=0, par1=1, par2=1)
```

Note that you must provide arguments for all the variables in the `CPT` for `v`; otherwise you will get an error. However, you may pass in additional variables; the function simply ignores them. For example,

```
#same as c(v=0, par1=1, par2=1)
c(v=0, par1=1, par2=1, unrelated_v=1)
```

This is particularly useful, because you can do something like the following

```
all_vars = {"v1":0, "v2":0, "v3":1, "v4":0, "v5":1, "v6":0, "v7":1, "v8":1}
c(**all_vars)
```

Thus, even if only a small subset of the variables are in a particular `CPT`, the syntax above allows us to easily find the probability of a particular vertex given its parents and children simply by passing in the instantiation of all variables into its `CPT`.

We will test your code on the ALARM Bayes net<sup>1</sup>. This network has been provided to you in the file `alarm.xmlbif` and code for parsing it has been provided in `gibbs.py`. We will test your code on several test cases where we examine if your code correctly samples a particular variable given a set of parent

---

<sup>1</sup>I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine, pages 247-256. Springer-Verlag, 1989.



variables. To see if it correctly samples a variable, we will run your function several times with the same inputs and compare the number of times you sample each value for that variable to the expected conditional distribution of the variable given the evidence variables (as generated by our solution Gibbs sampling code). Three example test cases have been provided in `gibbs.py`. Your code should run efficiently; each test case should run in under two minutes. Note: If you find that you only fail one or two test cases on Autolab, try uploading and running your code again, just in case it failed the tests by chance (but is actually correct).

**Hint:** You may want to start by thinking about how to **easily** express the probability of a particular variable taking on a value given the value of all other variables in terms of the conditional probabilities of the form in the `CPT` class. This could help you avoid writing complicated code and make your code more efficient.

## 5 Submitting to Autolab

Create a tar file containing your writeup and the completed `gibbs.py` modules for the programming problems. Make sure that your tar has these files at the root and not in a subdirectory. Use the following commands from a directory with your files to create a `handin.tar.gz` file for submission.

```
$ ls
gibbs.py  writeup.pdf  [...]
$ tar cvzf handin.tar.gz writeup.pdf gibbs.py
a writeup.pdf
a gibbs.py
$ ls
handin.tar.gz  gibbs.py  writeup.pdf  [...]
```