HOMEWORK D
INFERENCE AND OPTIMIZATION[1]

CMU 10-607: COMPUTATIONAL FUNDAMENTALS FOR MACHINE LEARNING (FALL 2018)

https://piazza.com/cmu/fall2018/10606607

OUT: Dec. 04, 2018

DUE: Dec. 08, 2018 11:59 PM

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section on the course site for more information: http://www.cs.cmu.edu/~mgormley/courses/606-607-f18/about.html#7-academic-integrity-policies
- **Late Submission Policy:** See the late submission policy here: http://www.cs.cmu.edu/~mgormley/courses/606-607-f18/about.html#6-general-policies
- **Submitting your work to Gradescope:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using Gradescope (https://gradescope.com/). Please use the provided template. Submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Alternatively, submissions can be written in LaTeX. Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. Each derivation/proof should be completed on a separate page. For short answer questions you **should not** include your work in your solution. If you include your work in your solutions, your assignment may not be graded correctly by our AI assisted grader.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For LaTeXusers, use ■ and ●for shaded boxes and circles, and don't change anything else.

[1]Compiled on Tuesday 4th December, 2018 at 14:45

Instructions for Specific Problem Types

For "Select One" questions, please fill in the appropriate bubble completely:

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie

○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie

● Noam Chomsky

For "Select all that apply" questions, please fill in all appropriate squares completely:

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

□ I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

■ I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

Fill in the blank: What is the course number?

10-607 10-707

1 Variable Elimination (Programming) [5 pts]

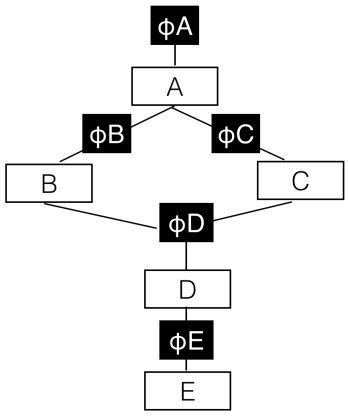


Figure 1.1: Bayes Net depicting variables and their dependencies.

Figure 1.1 depicts a factor graph for a given Bayes Net, and we are interested in finding the probability distribution P(E). You are given a partial implementation of Variable Elimination in `variable_elimination.py` for computing marginal probability of P(E) including the factor graph data structure.

Specifically, you are provided with the following classes:

- class Variable This class allows one to define Bayes Net variables. On initialization the variable object can be given a name and a domain of values. This list of domain values can be added to or deleted from in support of an incremental specification of the variable domain. The variable also has a set and get value method. These set a value for the variable that can be used by the factor class.
- class Factor This class allows one to define a factor specified by a table of values. On initialization the variables the factor is over is specified. This must be a list of variables. This list of variables cannot be changed once the constraint object is created. Once created the factor can be incrementally initialized with a list of values. To interact with the factor object one first sets the value of each variable in its scope (using the variable's set_value method), then one can set or get the value of the factor (a number) on those fixed values of the variables in its scope. Initially, one creates a factor object for every conditional probability table in the bayes-net. Then one initializes the factor by iteratively setting the values of all of the factor's variables and then adding the factor's numeric value using the add_value method.
- class FactorGraphClass This class allows one to put factors and variables together to form a Factor Graph. It serves as a convenient place to store all of the factors and variables associated with a Bayes Net in one place. It also has some utility routines to, e.g,., find all of the factors a variable is involved in.

Please implement the following methods.

1. function Naive_VE(Net, QueryVar): Please implement the naive version of the algorithm to find the distribution P(E) ie. compute the joint distribution of all variables for a given query E.
2. function get_bfs_ordering(Factors, QueryVar) which uses BFS and provides the order for variable elimination. Provide an assertion within the function to ensure the order is what you expect it should be. Return an ordered list of Variable objects.

3. function __collapse(Net, QueryVar): collapsing function that removes the variable X, its neighboring factors and adds in a new factor as appropriate.
4. function __variable_elimination(Net, QueryVar). Your final implementation should accept a factor graph and a query variable (i.e. the one for which we want the marginal distribution).
5. What would the appropriate assertions be for the following cases? Please provide your answers here as well as include them within the code.

   (a) **[1 pts]** Assertion about the size of the resulting factor after collapsing

   ```
   assert len(factors) == input_factor_len - len(f_eliminated) + 1
   ```

   (b) **[1 pts]** Assertion about the neighbors of the resulting factor after collapsing

   ```
   assert newFactorScope == factors[-1].get_scope()
   ```

   (c) **[1 pts]** Assertion about each factor summing to one.

   ```
   assert abs(sum(newfactor.value)/len(newfactor.get_scope()) - 1) < 1e-6
   ```

   Now you will be profiling code for execution time using cProfile. Profiling your code is often a useful step in understanding your code from a time point of view. As mentioned in the documentation "cProfile and profile provide deterministic profiling of Python programs. A profile is a set of statistics that describes how often and for how long various parts of the program executed.". You can find the documentation here - https://docs.python.org/2/library/profile.html. Cprofile provides the following statistics:

   - ncalls: for the number of calls,

- tottime: for the total time spent in the given function (and excluding time made in calls to sub-functions)
- cumtime :is the cumulative time spent in this and all subfunctions (from invocation till exit). This figure is accurate even for recursive functions.
- percall: is the quotient of cumtime divided by primitive calls
- filename:lineno(function) provides the respective data of each function

cProfile can be run using:

```
python -m cProfile [-o output_file] [-s sort_order] myscript.py
-o writes the profile results to a file instead of to stdout
```

-s specifies one of the sort_stats() sort values to sort the output by. This only applies when -o is not supplied.

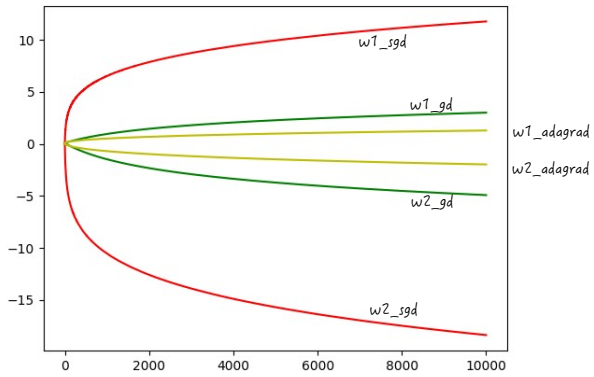
6. **[2 pts]** Profile the time of the function calls to Naive_VE and VE methods. Remember the method Naive_VE is where you have computed the joint probability distribution over all the variables. Report the total time of both functions.

```
Ordered by: standard name

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     1    0.000    0.000    0.001    0.001 variable_elimination1.py:572(__variable_elimination)
     1    0.001    0.001    0.001    0.001 variable_elimination1.py:528(Naive_VE)


    VE tottime: 0.000
    Naive_VE tottime: 0.001
```

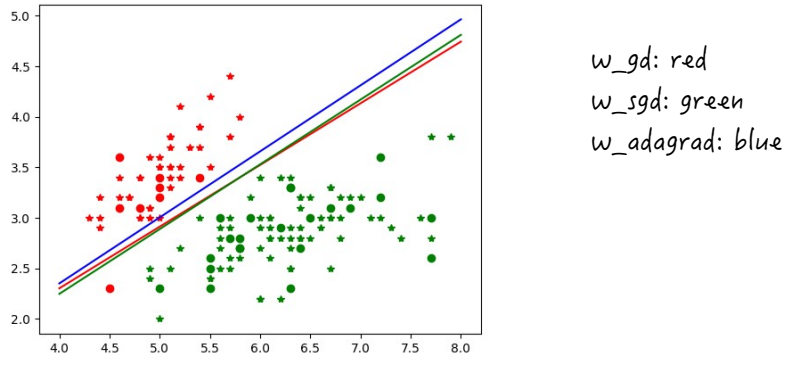
2 Unconstrained Loss Optimization for ML (Programming) [15 pts]

In this question, you will implement *Gradient Descent*, *Stochastic Gradient Descent*, and *AdaGrad*. Please code the update rules for each of these optimization techniques in the `loss_optimization.py` file provided to you. After implementing the update rules, please answer the following questions:

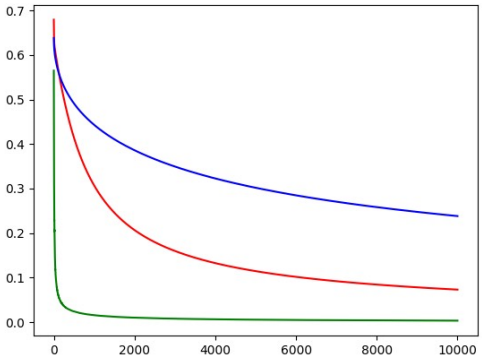
1. **[3 pts]** Create a plot for the 2D parameter space $\mathbf{w}_1, \mathbf{w}_2$ corresponding to the non-bias weight parameters. These are the second and third elements in the weight vector since the first element $\mathbf{w}_0$ corresponds to the bias term. In this plot, show the trajectory of the weight parameters captured at the end of each epoch as they are updated during the training process. The plot should include three trajectories, one for each of the three parameter update rules. Include a legend indicating which trajectory corresponds to which optimizer.



2. **[3 pts]** Plot the two-dimensional data with the positive datapoints (label=1) in green and negative datapoints (label=0) in red. Display the train and test datapoints using different market shapes. Also include in the plot the classifier decision boundary learned by the three different optimizers. Include a legend indicating which decision boundary corresponds to which optimizer.

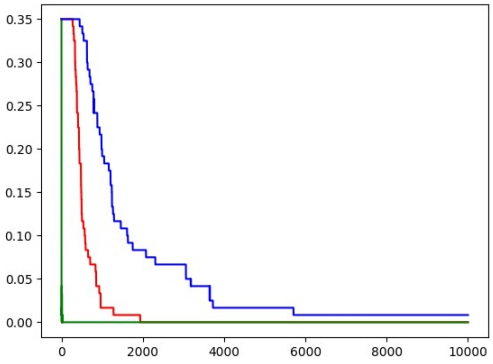


3. **[3 pts]** Plot the curve for train logistic loss versus train epoch for each of the three optimizers in a single plot. Include a legend indicating which curve corresponds to which optimizer. You can obtain the numbers required for the plot when you run the script after coding the three update rules.



w_gd: red
w_sgd: green
w_adagrad: blue

4. **[3 pts]** Plot the curve for train error versus train epoch for each of the three optimizers in a single plot. Include a legend indicating which curve corresponds to which optimizer. You can obtain the numbers required for the plot when you run the script after coding the three update rules.



w_gd: red
w_sgd: green
w_adagrad: blue

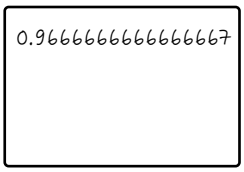
5. **[1 pts]** What is the accuracy obtained by the gradient descent optimizer on the test data?

0.9666666666666667

6. **[1 pts]** What is the accuracy obtained by the stochastic gradient descent optimizer on the test data?

0.9666666666666667

7. **[1 pts]** What is the accuracy obtained by the Adagrad optimizer on the test data?


0.9666666666666667

3 Constrained Loss Optimization for ML (Theory) [6 pts]

State True/False for the statements provided below. Provide a reason for your choice is the box accompanying each statement.

1. **[2 pts]** Simplex algorithm is an example of an interior point method for convex optimization. **Select one:**

   ○ True

   ● False

   Simplex algorithm and interior point are two different methods, while interior point method is more advanced with polynomial complexity.

2. **[2 pts]** In linear programming i.e. optimization problems with a linear objective and linear equality or inequality constraints, the optimal point lies at the boundary of the feasible set if the feasible set is bounded. Assume that all inequalities in the considered linear programming problem are non-strict i.e. of the form $\mathbf{Ax} \leq \mathbf{b}$. Feasible set is the set of all points that satisfy the constraints of the linear programming problem. **Select one:**

   ● True

   ○ False

   The optimal point is located at a vertex (corner) of the feasible region.

3. **[2 pts]** In linear models such as linear regression or logistic regression, stochastic gradient descent involves iteratively performing a gradient descent update on the scalar weight parameter associated with a randomly chosen feature. **Select one:**

   ○ True

   ● False

   SGD randomize the mini batches (training sample) instead of features.

4 Submission [1 pts]

1. **[1 pts]** In addition to the PDF submission with your written answers and plots, please zip the files `variable_elimination.py` and `loss_optimization.py` directly (do NOT place them in a folder and then zip), name the zipped file `<andrewid>-10607-hwd.zip`, and submit the zipped folder on Gradescope under the assignment `Homework D (Programming)`. This assignment is autograded and will assign points based on automated tests that run on your code. Your code file submissions should follow the provided code template files. In particular, they should NOT include any additional Python imports such as `matplotlib` which might make it fail on the Autograder. Please maintain any code for creating your plots in separate files. You do not have to submit your plotting code on Gradescope. Have you made the code submission on Gradescope? **Select one:**

   ● Yes

   ○ No

5 Collaboration Policy

After you have completed all other components of this assignment, report your answers to the collaboration policy questions detailed in the Academic Integrity Policies found here.

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details including names of people who helped you and the exact nature of help you received.

2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details including names of people you helped and the exact nature of help you offered.

3. Did you find or come across code that implements any part of this assignment? If so, include full details including the source of the code and how you used it in the assignment.