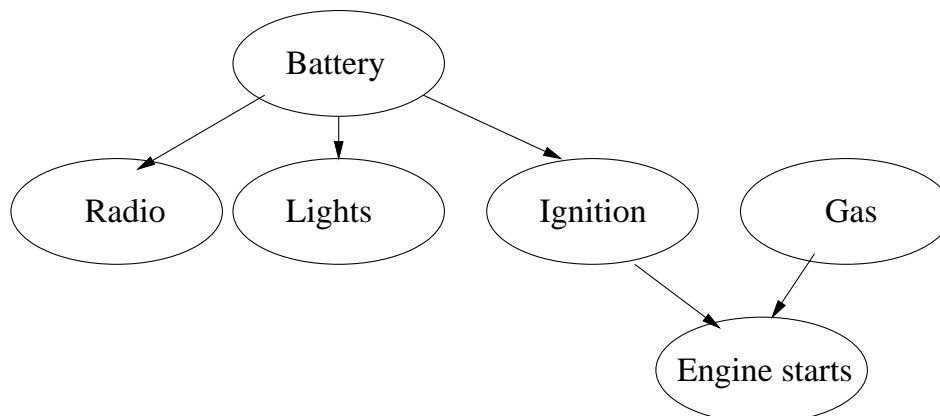

Problem assignment 11
Due: Tuesday, December 6, 2017

Learning

Problem 1. Learning parameters of the Bayesian belief network

Assume the Bayesian belief network for the diagnosis of the engine problem in the figure below.



All variables in the network are binary with true, false values.

Assume we have collected the data from different car repair shops in the neighborhood and we want to learn the parameters of the network. The data are given as vectors of values for every variable in the network. The data for variables are given in the following order:

(Battery, Radio, Lights, Ignition, Gas, EngineStarts).

The values are encoded using binary values: 1 for True and 0 for False. Your dataset is as follows:

```

1 1 1 1 1 1
1 1 1 0 1 0
1 0 1 1 1 1
1 0 1 1 1 0
0 1 0 0 1 0
1 1 1 0 0 0
1 1 1 1 1 1
1 1 0 1 1 1
1 1 0 1 1 1
1 1 0 1 0 1
1 0 0 1 1 1
0 0 0 1 1 1
1 1 1 0 1 0
1 1 1 1 0 0
1 0 1 1 0 1
1 0 1 0 1 0
1 1 1 1 1 1
1 0 1 1 1 0
1 1 0 1 0 1
0 0 1 0 0 0

```

Compute the ML estimates of the following probability parameters of the BBN network:

- $P(Gas)$
- $P(Lights|Battery = T)$
- $P(Lights|Battery = F)$
- $P(Ignition|Battery = T)$
- $P(EngineStarts|Ignition = T, Gas = T)$
- $P(EngineStarts|Ignition = T, Gas = F)$

Problem 2. Classification of handwritten digits with a logistic regression model

In this problem we use a logistic regression model to learn and to discriminate between the two digits (3 and 5) that can appear on the input. The input consists of a two dimensional array of 1s and 0s representing the grid of pixels and their colors (black and white), e.g.

```
# #####
    ##
    ##
    #####
        ###
#      ##
# ##   ##
    #####
```

Our objective is to train the model to classify digits 3 and 5 from the set of handwritten digit patterns and their labels. The data are divided into two sets:

- **training set** used for training the model (files *digit_x.dat*, *digit_y.dat*) ;
- **testing set** used for testing the performance of the learned model (files *digit_x_test.dat* and *digit_y_test.dat*).

The data are available on the class web page. The data in files are organized in rows. In file *digit_x.dat* rows represent the binary description of either a digit 3 or 5. File *digit_y.dat* reflects its corresponding label, 1 for 3 and 0 for 5. Test files *digit_x_test.dat* and *digit_y_test.dat* have the same structure. There are 100 sample digits in the training, and 400 in the testing files respectively.

To simplify the data access we have prepared the following C/C++ functions for you (included in file *main11.c*):

- ***init_digits()*** which initializes the digit system by loading digits them into DIGIT structures and initializing random number generator. Datasets are represented as arrays of DIGITS and are stored in *testing_set* and *training_set* arrays.
- ***get_random_training_digit()*** returns a random training example from the training data set. It is returned in the DIGIT structure.
- ***get_random_testing_digit()*** returns a random example from the testing data set.
- ***get_nth_training_digit(int nth)*** returns *nth* training example from the training data set. There are 100 samples in the training set.
- ***get_nth_testing_digit(int nth)*** returns *nth* example from the testing data set. There are 400 samples in the testing set.
- ***print_digit(DIGIT)*** which prints a digit as seen above.

Part a. In the logistic regression model we express the distribution of outputs $p(y = 1|\mathbf{x}, \mathbf{w})$ as:

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp\left(-\left[w_0 + \sum_{j=1}^k w_j x_j\right]\right)},$$

where \exp stands for the exponential function, and \mathbf{w} is a vector of weights w_0, w_1, \dots, w_{64} , such that w_0 corresponds to the bias weight and w_1, \dots, w_{64} to the weights for digit pixels. x_j is the j -th component of the input vector \mathbf{x} .

Write a C/C++ function for computing the probability of $y = 1$ given the set of weights \mathbf{w} and the input vector \mathbf{x} , $p(y = 1|\mathbf{x}, \mathbf{w})$. Definitions in file *main11.h* give you appropriate structures.

Part b. In the logistic regression we want to maximize the likelihood of the predictions. The optimization of the likelihood can be performed using the online algorithm with gradient-based updates. The $(i + 1)$ th on-line update of a weight w_j with a data sample $\langle \mathbf{x}, y \rangle$, is

$$w_j^{(i+1)} = w_j^{(i)} + \alpha(i + 1) \cdot \left[y - p(y = 1|\mathbf{x}, \mathbf{w}^{(i)}) \right] \cdot x_j,$$

where $\alpha(i + 1)$ is the learning rate that changes with the update step, $x_j^{(j)}$ is the j th component of the input vector \mathbf{x} .

Implement the online gradient-based procedure: *online_gradient_descent(int iterations)* for updating the weight parameters of the logistic regression model. Your program should start from zero weights (all weights \mathbf{w} set to 0 at the beginning). To update weights use the annealed learning rate $\alpha(i) = \frac{1}{2\sqrt{i}}$ where i indexes the i th update step. Repeat the update for 1500 steps.

To learn the weights use digits from the training set only. In every step pick samples randomly using function *get_random_training_digit()*.

Part c. The model (weights) learned in part b can be used to classify digits. The output label $\{0, 1\}$ for an input \mathbf{x} can be obtained using the following simple rule:

If $p(y = 1|\mathbf{x}, \mathbf{w}) \geq 0.5$ then output 1
else output 0.


Write C/C++ function *int classify(DIGIT digit)* for classifying input digits.

Part d. In this part we are interested in analyzing the quality of the learned logistic regression model. To test the quality we use average misclassification error. The average misclassification error for the dataset with N samples is defined as:

$$Error_{AMC} = \frac{\# \text{ of misclassified digits}}{N}.$$

Write C/C++ function to compute the average misclassification error of the model for both the training and testing datasets. Use functions *get_nth_training_digit(int nth)* and

`get_nth_testing_digit(int nth)` to access the data. The dataset sizes are given by constants `NUM_TEST_DIGS` 400 and `NUM_TRAIN_DIGS` 100.

Part e. Write and submit `main11.c` program that (1) learns the weights based on the training data set (as described above) and (2) tests the quality of the resulting model according to part d. Run the program multiple times and  (1) the weights learned and (2) average training and testing misclassification error. Remark: Although the resulting models can differ because of the random choice of the training examples you should be able to see models with the average misclassification error for the testing data below 0.15 most of the time.

Part f. Give the mean misclassification errors for both the training and test data in your report. Which misclassification error is higher? Training or testing? Can you explain it?

Part g. Optional (Extra credit). You may want to experiment a bit with number of updates and learning rate schedules. For example, try to learn using only 100, 200, 500 updates before returning the weights. In terms of learning schedule you can try e.g. $\alpha(i) = \frac{1}{i}$. Report results of your experiments and any hypothesis or conjectures about the performances.