Introduction: what problem are you trying to solve:

The selected implementation machine learning algorithm for this project is the Hidden Markov Models(HMM).

For the HMM there are three main groups of problems that each need to be addressed.

< This is a summary of the three main problems from this url below from authors Warakagoda, Narada D., and Norges T. Høgskole see reference for more detail http://jedlik.phy.bme.hu/~gerjanos/HMM/node6.html#SECTION00240000000000000000>

1) Given a sequence of observations what is the probability for each output.

This is addressed in the code by applying either the first-order Markov models or the second-order Markov model. . For first-order Markov model see source file Hmm.java.

2) Given a sequence of observation what is the most likely path of sequence and its output.

This is addressed in the code by applying Viterbi algorithm. . For Vitebi algorithm see source file Viterbi.java

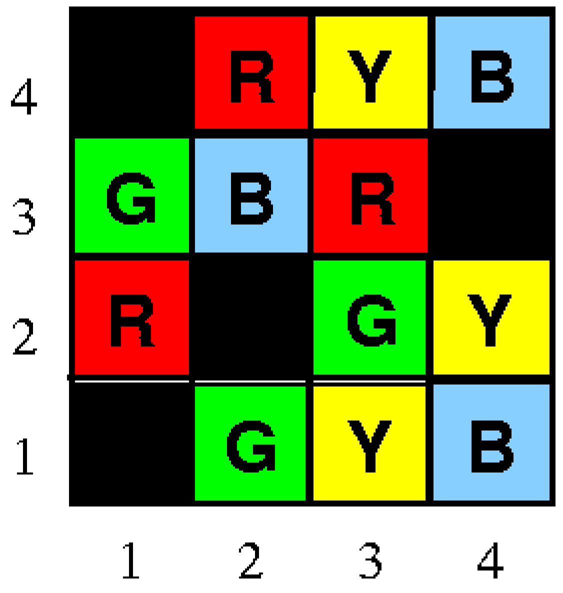
3) Given a sequence of observations what is the best model to maximize probability of output given known information.

This is addressed by the forward-backward algorithm. For forward-backward algorithm see source file fwd_bwd.java unfortunately the code is not working flawlessly yet. For this reason the user will not be able to use it.

There are three groups datasets that I have included with the project. Each group of dataset has its own problem that the HMM is trying to solve. They are under the folder dataSet then their group name. Each dataset sequence is separated by a period and two periods separate the training data from the testing data.

Group toy_robot:

This problem is trying to help the toy robot decode the path that the robot took. It is dataset is a representation of this block:



In the dataset, the states are the xy coordinates and the outputs are each of the colors(r for red, y for yellow, g for green, b for blue). The robot moves randomly up, down, left, and right to any square that is not black. If black is selected then the robot does not move and the next random move is selected. The problem that the HMM are to help deal with is that the robots sensor to detected the color are only 90% accurate. Thus in the data set 10% of the outputs are incorrectly assigned. To find the correct path is difficult since once the HMM gets off the path it has a difficult time getting back on the path. Once HMM gets off the path of the robot then it selects nearby colors that match the current output. Thus, the overall error rate is dependent on the previous selection.

This group has two files robot_no_momentum.data and robot_with_momentum.data . The file with momentum only moves randomly when it starts or hits a black square else it keeps moving in the same direction as its last move. The testing data and training data are both 200 sequences long. Each sequence is 200 moves long.

Group tracking_topics:

This group is of a dataset based on 5302 articles selected form six newsgroups(baseball, auto, guns, med, religion and windows). 1500 articles were used for training the rest for testing. The articles were formatted just to their words that they use in lowercase letters then randomly removed parts of the article and then concatenated together. In the dataset, the states are the words from the article and the outputs are the topics.

There is only one dataset file for this group file topics.data.

Group typos_no_dictionary:

This group is solving the problem of typos without using a dictionary. A typo is a near neighbor on the keyboard to the desired key. To separate the words two underscores are used(__). The state is the first letter then there is a space then the output letter. 20,000 characters are used for training while 161,000 are used for testing.

In this group there are two files one with 10% typos named typos10.data and another with 20 % typos named typos20.data .

Instructions on how to install and run your code – OS, compiler, user interface to receive network & learning parameters, where output is made:

The source code is coded in java 1.7. The executable file is runme.jar.

The output is sent to text file output.txt in the local directory of the program. The average error rate for all sequences is printed on the console line.

The only thing that the user has to selected when running the file is to pick the desire data file. This is done by feeding the executable file the data file name as is input.

An example on how to run would be open command prompt. Then change dir to Andrew_Tillmann_finial. Then type command java -jar runme.jar ./datasets/toy_robot/robot_no_momentum.data

Description of the system: not line by line, but top-down descriptions of the important parts:

The code is broken down into four java files: DataSet, Hmm, RunViterbi, Viterbi .

The file DataSet and RunViterbi is from this website:
https://www.cs.princeton.edu/courses/archive/fall06/cos402/hw/hw5/hw5.html
Very few changes have been made on the original RunVitebi file. Furthermore, no changes have been made to the DataSet.java file. These files contain no machine learning algorithms they just process the data files and run the data into the machine learning algorithms.

DataSet.java processes the data files. It finds the number of states and outputs. Furthermore, it records a string of each state and each output then putting them into separate matrixes. The last thing the file does is to fill two 2D matrixes for each state and matching output for both the training data and the testing data. The first dimension is the sequence.

RunVitebi.java takes the data form DataSet.java and runs it into the machine learning algorithms. It prints out into output.txt: the start probabilities for each state, the transition probabilities in a matrix of from state to state, the output probabilities of form state to output, then each training sequence, then each testing sequence.

The training and testing sequence are three column outputs. The first column is the real state, the next is the predicted state and the last is the given output.

Following each sequence an error rate for the sequence is printed.
RunVitebi.java also prints the average test error of all test sequences.
It is also important to note that this file has the main function in it.

I did the coding for the next to files. However, most of the comments were already there.
Hmm.java builds a first-order Hidden Markov Model. It has six public functions in the file:

1. Hmm (given data this function creates the first-order Hidden Markov model)
2. getNumStates(returns the number of states).
3. getNumOutputs(returns the number of outputs).
4. getLogStartProb(given a state it returns the probability of that state starting).
5. getLogTransProb(given fromState and toState it returns the probability of that occurrence).
6. getLogOutputProb(given state and output it return the probability of that occurrence).

   The probability functions use Laple's law of succession thus for each probability the numerator starts at one and the denominator starts at two. Furthermore, for each probability function the return value is in log format.

Vitebi.java is the vitebi algorithm to select the best path given the HMM. It has two public functions in the file:

1. Viterbi(takes HMM from Hmm.java and creates matrixes of all start, output and transition probabilities)
2. mostLikelySequence(Given output it return the state of the most like to occur, this state is found from the highest likelihood value gathered from the matrixes of start, output and transition probabilities)

LearningAlgorthm.java this file was set up so the user could select to run more than one algorithm. Right now it is designed to run either Vitebi.java or BaumWelch.java.

BaumWelch.java is the implementation of the Baum Welch algorithm that uses the forward-backward algorithm. Unfortunately, this part of the project is still a work in process. Since it is incomplete the user can not implement this algorithm towards the data set.

Ideas for enhancement:
Can finish the BaumWelch.java file.

References to sources of information that you drew on:

"CS 402: HW#5, HMM's and the Viterbi Algorithm." *CS 402: HW#5, HMM's and the Viterbi Algorithm*. N.p., n.d. Web. 15 Nov. 2013.

Warakagoda, Narada D., and Norges T. Høgskole. "A Hybrid ANN-HMM ASR System with NN Based AdaptivepreprocessingM.Sc. Thesis." *A Hybrid ANN-HMM ASR System with NN Based Adaptive Preprocessing Diploma Thesis*. Institutt for Teleteknikk Transmisjonsteknikk, 10 May 1996. Web. 15 Nov. 2013.