Project Report

Haidong Tian

The data set used for the final project is *Breast Cancer Wisconsin (Original)*, which comes from the online machine learning repository (*https://archive.ics.uci.edu/ml/datasets.php*). The goal is to predict whether the cancer is malignant or benign based on a given set of values of observables, which is a binary classification problem. The original data set is stored in two separate files,

- *breast-cancer-wisconsin.names*, containing the description of the data set,
- *breast-cancer-wisconsin.data*, containing the actual data points.

I. Data Preprocessing

The original data can be transformed into a proper format by running *dataPreprocessing.py*, which,

- deletes sixteen instances with missing values,
- deletes one column under the name *Sample code number* which indexes the instances,
- rearranges columns so that the first column corresponds to the responses and the rest corresponds to the values of observables,
- reassigns 1 to malignant cancer and -1 to benign cancer where 4 and 2 are assigned to malignant and benign cancer respectively in the original data set,

and,

- outputs *data_file.csv*, which contains the data set in the proper format.

In *data_file.csv*, there are 683 instances and 9 observables. All the values of observables are scaled between 1 and 10. Some sample data from *data_file.csv* are shown in Fig. 1.

Class	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses
-1	5	1	1	1	2	1	3	1	1
-1	5	4	4	5	7	10	3	2	1
-1	3	1	1	1	2	2	3	1	1
-1	6	8	8	1	3	4	3	7	1
-1	4	1	1	3	2	1	3	1	1
1	8	10	10	8	7	10	9	7	1
-1	1	1	1	1	2	10	3	1	1

Fig. 1. Sample data from *data_file.csv*.

II. Data Splitting

In *data_file.csv*, the ratio of malignant cancer to benign cancer is 35% : 65%. While this ratio is kept fixed, the data set is sampled and split into a training set containing 80% of the instances and a testing set containing 20% of the instances by running *dataSplit.py* with *data_file.csv* as the command-line argument, which generates,

- *train.csv*, containing the training set,
- *test_observables.csv*, containing the observables in the testing set,
- *test_responses.csv*, containing the responses in the testing set.

III. Model Selection

Models are evaluated by *score.py*, which takes the following as the command-line argument,

- a filename of a csv file containing true responses,
- a filename of a csv file containing predictions,

and outputs,

- accuracy,
- sensitivity.

The sensitivity is referred to the proportion of those who have malignant cancer that get predicted to have it. In the model selection process, the training data set, *train.csv*, is further sampled and split into an actual training set containing 80% of the instances and a validation set containing 20% of the instances while keeping the ratio of malignant cancer to benign cancer fixed. In each fold, models are trained on the actual training set and validated on the validation set. Suitable models are selected by comparing the average validation accuracy and the average validation sensitivity from a 10-fold validation process.

IV. Models

1. Logistic Regression

The code for logistic regression without optimization is in *logisticReg.py*, which takes the following as the command-line arguments,

- a filename of a csv file containing variable dimensions of training data points,
- a filename of a csv file containing the same dimensions of test data points minus the first column,
- an intended number of iterations,
- an intended learning rate,

and outputs,

- the time cost of running *logisticReg.py*,
- *logisticReg_predictions.csv*, containing one column of predicted values.

The intended number of iterations and the intended learning rate will be determined in the model selection process.

To address the potential issue of logistic regression taking long time to converge, the ADAM optimization algorithm can be applied, which uses both the inertia of the mean **M** and that of the variance **V** as described in the *section 13.2* of *Lecture Notes 13*. The user parameters are set as the following,

- inertia of old mean $\beta_M = 0.9$,
- inertia of old variance $\beta_V = 0.999$,
- constant to avoid division by zero $\epsilon = 10^{-8}$.

The code for ADAM is in *ADAM.py*, which takes the following as the command-line arguments,

- a filename of a csv file containing variable dimensions of training data points,
- a filename of a csv file containing the same dimensions of test data points minus the first column,
- an intended number of iterations,
- an intended learning rate,

and outputs,

- the time cost of running *ADAM.py*,
- *ADAM_predictions.csv*, containing one column of predicted values.

The intended number of iterations and the intended learning rate will be determined in the model selection process.

2. Support Vector Machines

The following kernel function is used in support vector machines,

$$kernel(\mathbf{x}, \mathbf{x}') = (\frac{1}{\gamma}\mathbf{x} \cdot \mathbf{x}')^{degree}$$

where the linear boundary is defined when $degree = 1$ and the nonlinear boundary is defined by a homogenous polynomial when $degree > 1$ and $\gamma = 100^{degree}$ is the normalization factor to keep the value of the kernel function in the range of (0, 1]. The code for support vector machines is in *homoPolySVM.py*, which takes the following as the command-line arguments,

- a filename of a csv file containing variable dimensions of training data points,
- a filename of a csv file containing the same dimensions of test data points minus the first column,
- an intended degree of the kernel function

and outputs,

- the time cost of running *homoPolySVM.py*,
- *homoPolySVM_predictions.csv*, containing one column of predicted values.

The intended degree of the kernel function will be determined in the model selection process.

3. Neural Networks

The neural networks use one hidden layer and the logistic activation function.

The code for the ADAM optimizer with a learning rate of 0.001 is in *lognnADAM.py*, which takes the following as the command-line arguments,

- a filename of a csv file containing variable dimensions of training data points,
- a filename of a csv file containing the same dimensions of test data points minus the first column,
- an intended number of hidden units,
- an intended number of epochs,

and outputs,

- the time cost of running *lognnADAM.py*,
- *lognnADAM_predictions.csv*, containing one column of predicted values.

The intended number of hidden units and the intended number of epochs will be determined in the model selection process.

The code for the SGD optimizer with a learning rate of 0.08 and zero momentum is in *lognnSGD.py*, which takes the following as the command-line arguments,

- a filename of a csv file containing variable dimensions of training data points,
- a filename of a csv file containing the same dimensions of test data points minus the first column,
- an intended number of hidden units,
- an intended number of epochs,

and outputs,

- the time cost of running *lognnSGD.py*,
- *lognnSGD_predictions.csv*, containing one column of predicted values.

The intended number of hidden units and the intended number of epochs will be determined in the model selection process.

V. Model Selection Results

Results from the 10-fold validation process can be obtained by running *nFold.py* with 10 (the intended number of folds) as the command-line argument, which,

- for logistic regression without optimization, stores, as a function of iterations (200, 400, 600, 800, 1000, 2000, 3000 and 4000 iterations) and learning rates (learning rate = 0.5, 0.1, 0.075, 0.05 and 0.01), the average validation accuracy in *logisticReg_avgValAccuracy.csv* (visualized in *logisticReg_avgValAccuracy.png*) and the average validation sensitivity in *logisticReg_avgValSensitivity.csv* (visualized in *logisticReg_avgValSensitivity.png*).
- for logistic regression with the ADAM optimizer, stores, as a function of iterations (100, 200, 400, 600, 800, 1000, 2000, and 3000 iterations) and learning rates (learning rate = 0.5, 0.1, 0.075, 0.05 and 0.01), the average validation accuracy in *ADAM_avgValAccuracy.csv* (visualized in

ADAM_avgValAccuracy.png) and the average validation sensitivity in *ADAM_avgValSensitivity.csv* (visualized in *ADAM_avgValSensitivity.png*).

- for support vector machines, stores, as a function of degree in the kernel (degree = 1, 2, 3, 4, 5, 6, 7, 8 and 9), the average validation accuracy in *homoPolySVM_avgValAccuracy.csv* (visualized in *homoPolySVM_avgValAccuracy.png*) and the average validation sensitivity in *homoPolySVM_avgValSensitivity.csv* (visualized in *homoPolySVM_avgValSensitivity.png*).
- for neural networks with the ADAM optimizer, stores, as a function of the number of hidden units (4, 5, 8, 9, 12 and 13 hidden units) and epochs (200, 400, 600, 800, 1000, 2000, 4000, 6000, 7000 and 8000 epochs), the average validation accuracy in *lognnADAM_avgValAccuracy.csv* (visualized in *lognnADAM_avgValAccuracy.png*) and the average validation sensitivity in *lognnADAM_avgValSensitivity.csv* (visualized in *lognnADAM_avgValSensitivity.png*).
- for neural networks with the SGD optimizer, stores, as a function of the number of hidden units (4, 5, 8, 9, 12 and 13 hidden units) and epochs (200, 400, 600, 800, 1000, 2000, 4000, 6000, 7000 and 8000 epochs), the average validation accuracy in *lognnSGD_avgValAccuracy.csv* (visualized in *lognnSGD_avgValAccuracy.png*) and the average validation sensitivity in *lognnSGD_avgValSensitivity.csv* (visualized in *lognnSGD_avgValSensitivity.png*).

1. Logistic Regression

The 10-fold validation results for logistic regression without optimization are shown in Fig. 2. When the learning rate is large (learning rate = 0.5), both the average validation accuracy and the average validation sensitivity oscillate a lot with the number of iterations. When the learning rate is small (learning rate = 0.01), both the average validation accuracy and the average validation sensitivity do not converge within the maximum number of iterations. For logistic regression without optimization, the optimal learning rate is 0.1 and the optimal number of iterations is 3000, where the average validation accuracy converges to 0.9633 and the average validation sensitivity converges to 0.9369, which are very close to those numbers converged to for logistic regression with the ADAM optimizer.

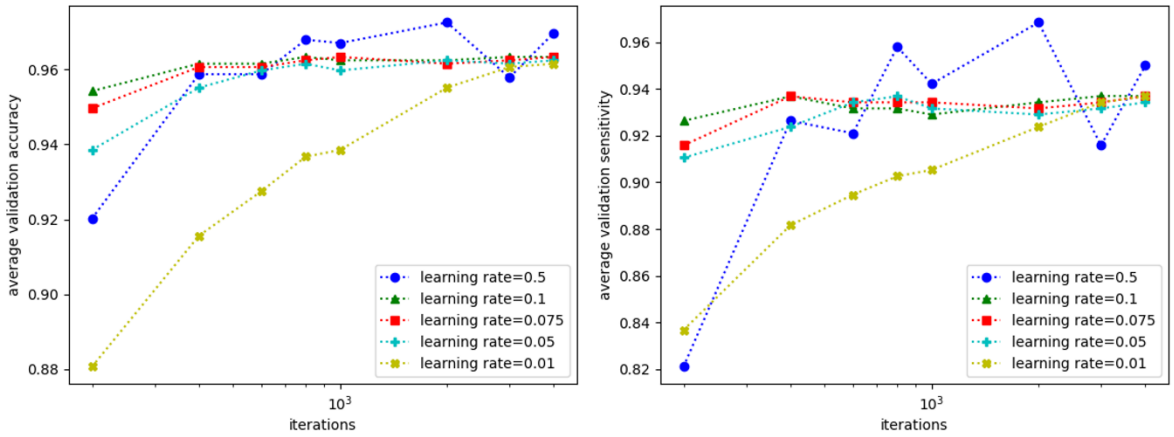

Fig. 2. 10-fold validation results for logistic regression without optimization.

The 10-fold validation results for logistic regression with the ADAM optimizer are shown in Fig. 3. When the learning rate is large (learning rate = 0.5), both the average validation accuracy and the average validation sensitivity oscillate a lot with the number of iterations. For all four other learning rates (learning rate = 0.1, 0.075, 0.05 and 0.01), the average validation accuracy converges to 0.9642 and the average validation sensitivity converges to 0.9395. When learning rate equal to 0.1 or 0.05, it only takes 600 iterations to converge, substantially fewer than that in logistic regression without optimization. As a result, it takes much less time for logistic regression with the ADAM optimizer to converge than without optimization. For logistic regression with the ADAM optimizer, the optimal learning rate is 0.1 and the optimal number of iterations is 600.

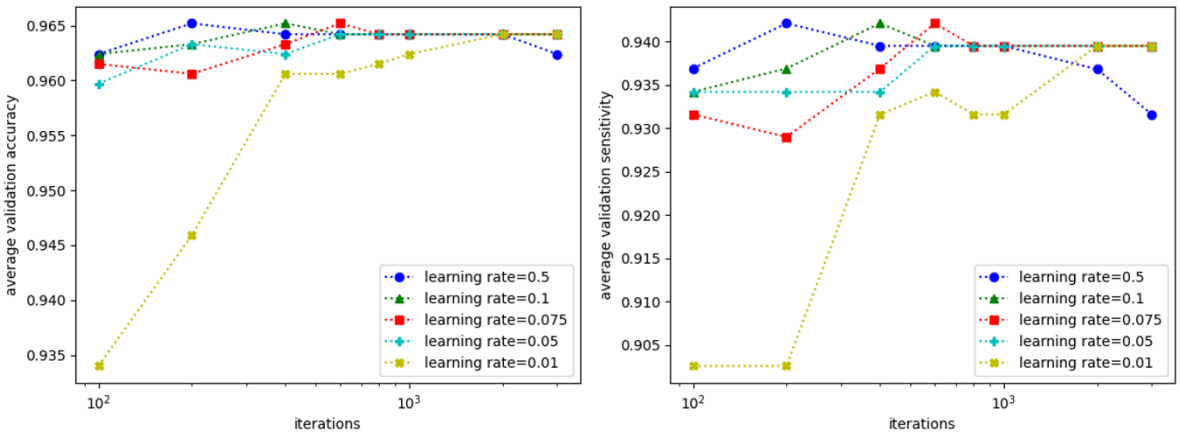

Fig.3. 10-fold validation results for logistic regression with the ADAM optimizer.

2. Support Vector Machines

The 10-fold validation results for support vector machines are shown in Fig. 4. When the degree of the polynomial is either small (degree = 1) or large (degree = 8 and 9), the average validation accuracy is low while the average validation sensitivity is high. This is because the trained models are trying to assign malignant cancer to as many instances in the validation set as possible. For support vector machines, the optimal degree of the kernel function is 3, where the average validation accuracy is 0.9367 and the average validation sensitivity is 0.8842. That both the average validation accuracy and the average validation sensitivity are low could result from hard-margin support vector machines being implemented.

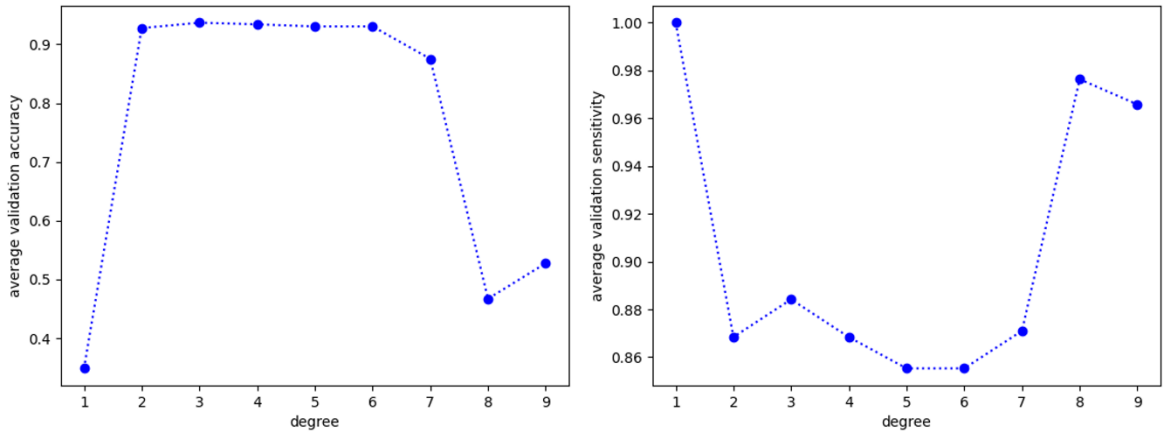

Fig.4. 10-fold validation results for support vector machines.

3. Neural Networks

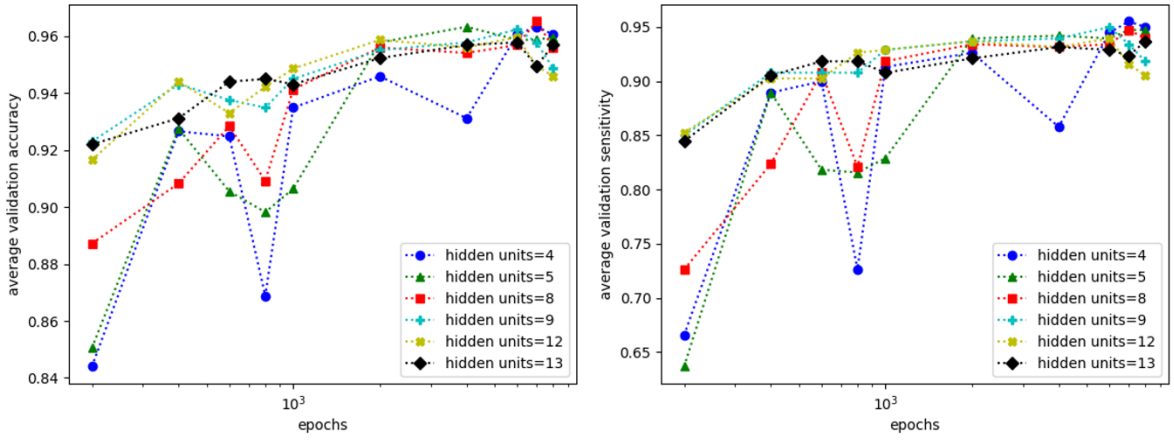

Fig. 5. 10-fold validation results for two-layer neural networks with the ADAM optimizer.

The 10-fold validation results for two-layer neural networks with the ADAM optimizer are shown in Fig. 5. If putting more weight in sensitivity, the optimal number of hidden units is 4 and the optimal number of epochs is 7000, where the average validation accuracy is 0.9633 and the average validation sensitivity is 0.9553.

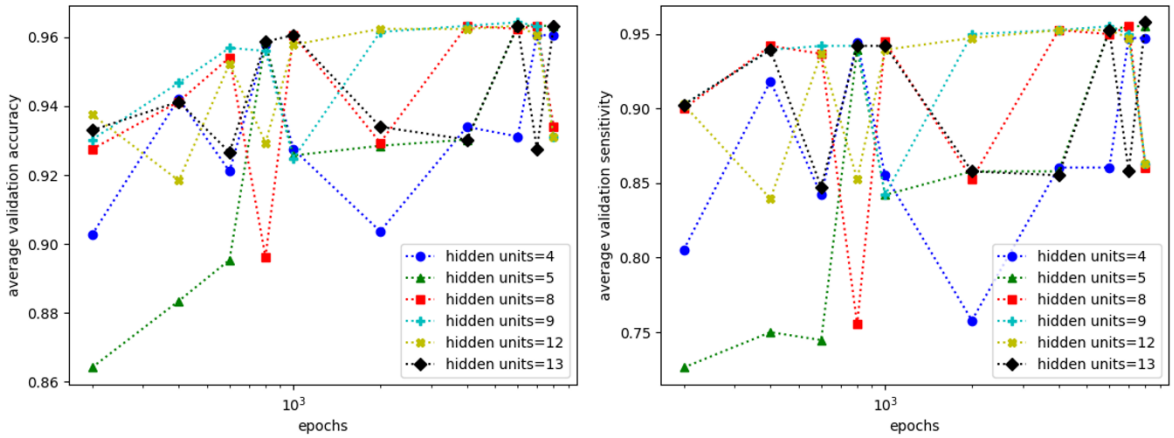

Fig. 6. 10-fold validation results for two-layer neural networks with the ADAM optimizer.

The 10-fold validation results for two-layer neural networks with the SGD optimizer are shown in Fig. 6. If putting more weight in sensitivity, the optimal number of hidden units is 13 and the optimal number of epochs is 8000, where the average validation accuracy is 0.9633 and the average validation sensitivity is 0.9579.

VI. Test Results

Models, with the optimal parameters identified in the model selection process, are trained on the entire training set and tested on the test set. The test results are summarized in Fig. 7, which, as well, are compared with the average validation accuracy and the average validation sensitivity corresponding to the optimal parameters in the model selection process. All models other than support vector machines with the homogeneous polynomial kernel can be well generated to unseen data. Logistic regression without optimization has the same test accuracy and test sensitivity as those with the ADAM optimizer, which suggests the robustness of the model. Similarly, robustness of neural networks can be seen through the same test accuracy and test sensitivity obtained with different architectures and optimizers. The validation accuracy, the validation sensitivity, the test accuracy and the test sensitivity of support vector machines

are low, which may result from hard-margin support vector machines being implemented. For this simple small data set, logistic regression works equally well as the relatively complex neural networks in terms of test sensitivity and slightly better in terms of test accuracy. However, the neural networks implemented here is only relatively complex to other models implemented. The neural networks implemented here are rather simple in terms of architectures, which may be sufficient but unlikely to be optimal for this data set.

models	average validation accuracy	average validation sensitivity	test accuracy	test sensitivity	time cost
logistic regression without optimization learning rate = 0.1 3000 iterations	0.9633	0.9369	0.9927	0.9792	55.98 s
logistic regression with the ADAM optimizer learning rate = 0.1 600 iterations	0.9642	0.9395	0.9927	0.9792	12.58 s
support vector machines degree = 3	0.9367	0.8842	0.8978	0.7917	0.4 s
neural networks with the ADAM optimizer One hidden layer 4 hidden units 7000 epochs	0.9633	0.9553	0.9854	0.9792	23.21 s
neural networks with the SGD optimizer One hidden layer 13 hidden units 8000 epochs	0.9633	0.9579	0.9854	0.9792	28.29 s

Fig. 7. test results