COMP5212 Machine Learning Project 1 Report

Jianhao JIAO, 20475718, jjiao@ust.hk

March 12, 2018

1 Data Set

Data set	#features	#train	#test
Breast cancer	10	547	136
Diabetes	8	615	153
Digit	64	800	200
Iris	4	120	30
Wine	13	142	36

2 Notation and abbreviation

1. m_i: maximum iteration
2. c: stop criterion
3. η: learning rate
4. p: regularization term
5. a_{train}: classification accuracy on training sets
6. a_{test}: classification accuracy on testing sets
7. l_{train}: log loss on training sets
8. l_{test}: log loss on testing sets
9. $times(ms)$: execution time for each method to complete the classification task(= time for learning + time for classification on both training and testing data sets)
10. LR: logistic regression
11. NN: neural network
12. SVM_L: SVM with linear kernel
13. SVM_R: SVM with RBF kernel.

3 Logistic Regression

1. Principle

   (a) Log loss: Given a single sample with true label r^ℓ in $0, 1$, estimated probability $y^{(\ell)} \equiv P(C_1|x^{(\ell)}) = sigmoid(w^T x^{(\ell)} + w_0)$, the log loss is

   $$L(w, w_0|x^{(\ell)}) = \log P(r^{(\ell)}|x^{(\ell)}) = r^{(\ell)} \log y^{(\ell)} + (1 - r^{(\ell)}) \log(1 - y^{(\ell)})$$

   .

   (b) Cross-entropy: Given a sample datasetχ with size N, the cross-entropy is

   $$E[w, w_0|\chi] = -\sum_{\ell=1}^{N} L(w, w_0|x^{(\ell)}) = -\sum_{\ell=1}^{N} [r^{(\ell)} \log y^{(\ell)} + (1 - r^{(\ell)}) \log(1 - y^{(\ell)})]$$

   In the learning process, the cross-entropy should be minimized.

   (c) Gradient-Descent Learning:

   $$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = \eta \sum_{\ell} (r^{(\ell)} - y^{(\ell)}) x_j^{(\ell)} \quad \Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_{\ell} (r^{(\ell)} - y^{(\ell)})$$

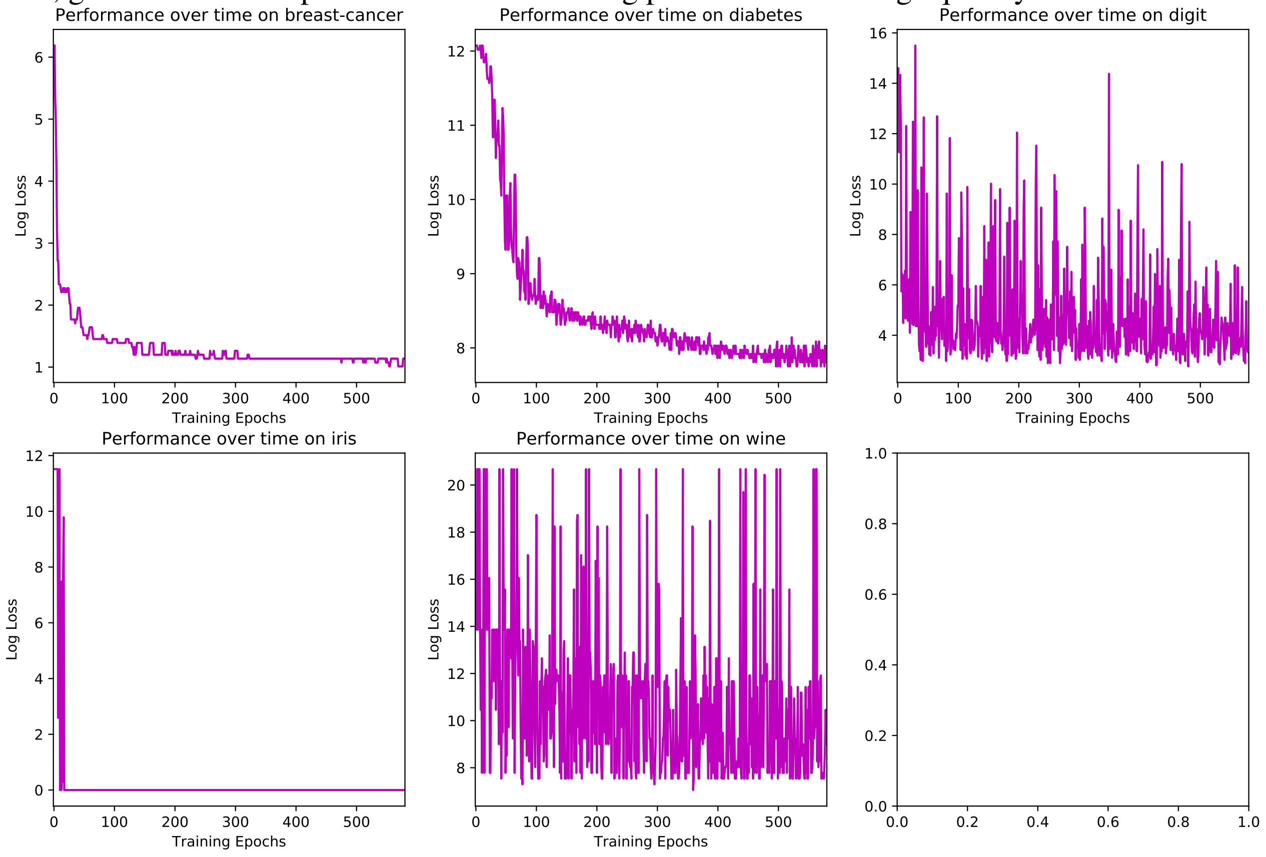
2. Experiment

   (a) Experiment Description: We try out different $\eta (< 1)$, stop criterion between iterations, maximum iterations number in the learning process on the given data sets. Furthermore, we will decrease η gradually by setting `learning_rate='optimal'` in the initialization of `SGDClassifier` class. Furthermore, the solution found may depend on the initial weights values(s.t. w, w_0) chosen randomly, we also repeat each setting for 100 times to get the average classification accuracy. Additionally, If we to observe this model's performance over time, we need to write some code to implement online learning based on this function `partial_fit()` of `SGDClassfier`. Outline of the code is below:

```
iteraitons = 30
batch_num = 20
for X, Y in zip(np.array_split(train_X, batch_num),
                np.array_split(train_Y, batch_num)):

    clf.partial_fit(X, Y, classes=classes)
    # loss
    loss.append(log_loss(train_Y, clf.predict(train_X),
                         normalize=True))
    # accuracy on train & test data
    accu_train.append(clf.score(train_X, train_Y))
    accu_test.append(clf.score(test_X, test_Y))
    # time required for learning and testing
    t_last.append(time.time() - t_begin)
print(clf.score(train_X, train_Y)*100)
print(log_loss(train_Y, clf.predict(train_X), normalize=True))
print(clf.score(test_X, test_Y)*100)
print(log_loss(test_Y, clf.predict(test_X), normalize=True))
print(np.sum(t_last)*1000, 'ms')
```

Figure 1: The performance over time of the logistic regression model on the given data sets by using the full training data sets. We can observe that on the breast-cancer, diabetes and iris data sets, gradient descent optimization in the learning process is to converge quickly.



(b) Result:

i. Classification accuracy: Table 1

ii. Performance over time (including loss and classification accuracy in the learning process: Table2, Fig.1 and Fig.2.

Table 1: Experiment results of logistic regression on different settings

Dataset	m_i	c	η	p	$a_{train}(\%)$	$a_{test}(\%)$	l_{train}	l_{test}	$time(ms)$
breast-cancer	1000	0.01	0.1	l2	96.98	96.22	0.0863	0.0725	1.448
diabetes	1000	0.01	0.1	l2	76.63	76.69	0.4837	0.5079	1.493
digit	1000	0.01	0.1	l2	84.35	85.15	4.5829	4.7096	3.504
iris	1000	0.01	0.1	l2	100.0	100.0	0.0125	0.0206	1.182
wine	1000	0.01	0.1	l2	65.62	59.14	12.2384	14.1025	1.178

Decreasing stopping criterion

Dataset	m_i	c	η	p	$a_{train}(\%)$	$a_{test}(\%)$	l_{train}	l_{test}	$time(ms)$
breast-cancer	1000	0.005	0.1	l2	96.98	96.42	0.0859	0.0722	1.484
diabetes	1000	0.005	0.1	l2	76.63	77.37	0.4781	0.5008	1.533
digit	1000	0.005	0.1	l2	84.72	85.19	4.6213	4.6844	3.453
iris	1000	0.005	0.1	l2	100.0	100.0	0.0111	0.0188	1.197
wine	1000	0.005	0.1	l2	66.85	61.19	11.5451	13.3861	1.187

Dataset	m_i	c	η	p	$a_{train}(\%)$	$a_{test}(\%)$	l_{train}	l_{test}	$time(ms)$
breast-cancer	1000	0.001	0.1	l2	97.19	96.36	0.0834	0.071	1.569
diabetes	1000	0.001	0.1	l2	76.89	77.29	0.4789	0.5032	1.590
digit	1000	0.001	0.1	l2	85.83	85.55	4.4925	4.5434	4.123
iris	1000	0.001	0.1	l2	100.0	100.0	0.0064	0.012	1.230
wine	1000	0.001	0.1	l2	66.73	59.0	12.18	14.1548	1.184

Decreasing η

Dataset	m_i	c	η	p	$a_{train}(\%)$	$a_{test}(\%)$	l_{train}	l_{test}	$time(ms)$
breast-cancer	1000	0.01	0.01	l2	95.86	96.01	0.1084	0.0986	1.565
diabetes	1000	0.01	0.01	l2	75.18	74.97	0.5115	0.5297	1.691
digit	1000	0.01	0.01	l2	85.75	85.68	1.9684	1.9145	3.419
iris	1000	0.01	0.01	l2	100.0	100.0	0.0570	0.0733	1.225
wine	1000	0.01	0.01	l2	65.63	57.97	12.7791	14.3822	1.196

Increasing η

Dataset	m_i	c	η	p	$a_{train}(\%)$	$a_{test}(\%)$	l_{train}	l_{test}	$time(ms)$
breast-cancer	1000	0.01	0.3	l2	95.05	96.42	0.0855	0.0738	1.449
diabetes	1000	0.01	0.3	l2	75.29	75.24	0.5076	0.5334	1.480
digit	1000	0.01	0.3	l2	85.36	85.16	4.9667	4.9823	3.437
iris	1000	0.01	0.3	l2	100.0	100.0	0.0047	0.0094	1.175
wine	1000	0.01	0.3	l2	65.87	61.25	11.3271	13.3838	1.215

Decreasing $\eta = \frac{1.0}{\alpha(t+t0)}(\alpha = 0.0001)$ over time

Dataset	m_i	c	η	p	$a_{train}(\%)$	$a_{test}(\%)$	l_{train}	l_{test}	$time(ms)$
breast-cancer	1000	0.01		l2	96.20	95.69	0.2705	0.2258	1.626
diabetes	1000	0.01		l2	73.16	70.56	0.7646	0.8059	2.012
digit	1000	0.01		l2	86.67	86.0	4.5237	4.7937	4.395
iris	1000	0.01		l2	100.0	99.83	0.0000	0.0087	1.200
wine	1000	0.01		l2	68.97	60.42	11.5681	13.669	1.236

Table 2: Performance over time on different data sets

Dataset	c	$a_{train}(\%)$	$a_{test}(\%)$	l_{train}	$time(ms)$
breast-cancer	0.001	96.71	96.32	1.137	455.89
diabetes	0.001	77.39	78.43	7.81	470.92
digit	0.001	90.125	87.0	3.4108	675.65
iris	0.001	100.0	100.0	0.0015	371.17
wine	0.001	73.94	63.89	8.99	389.51

Figure 2: The performance over time of the logistic regression model on the given data sets by using the full training data sets. We can observe that in the digit and wine data sets, gradient descent optimization in the learning process is opt to be fell into local minimal.

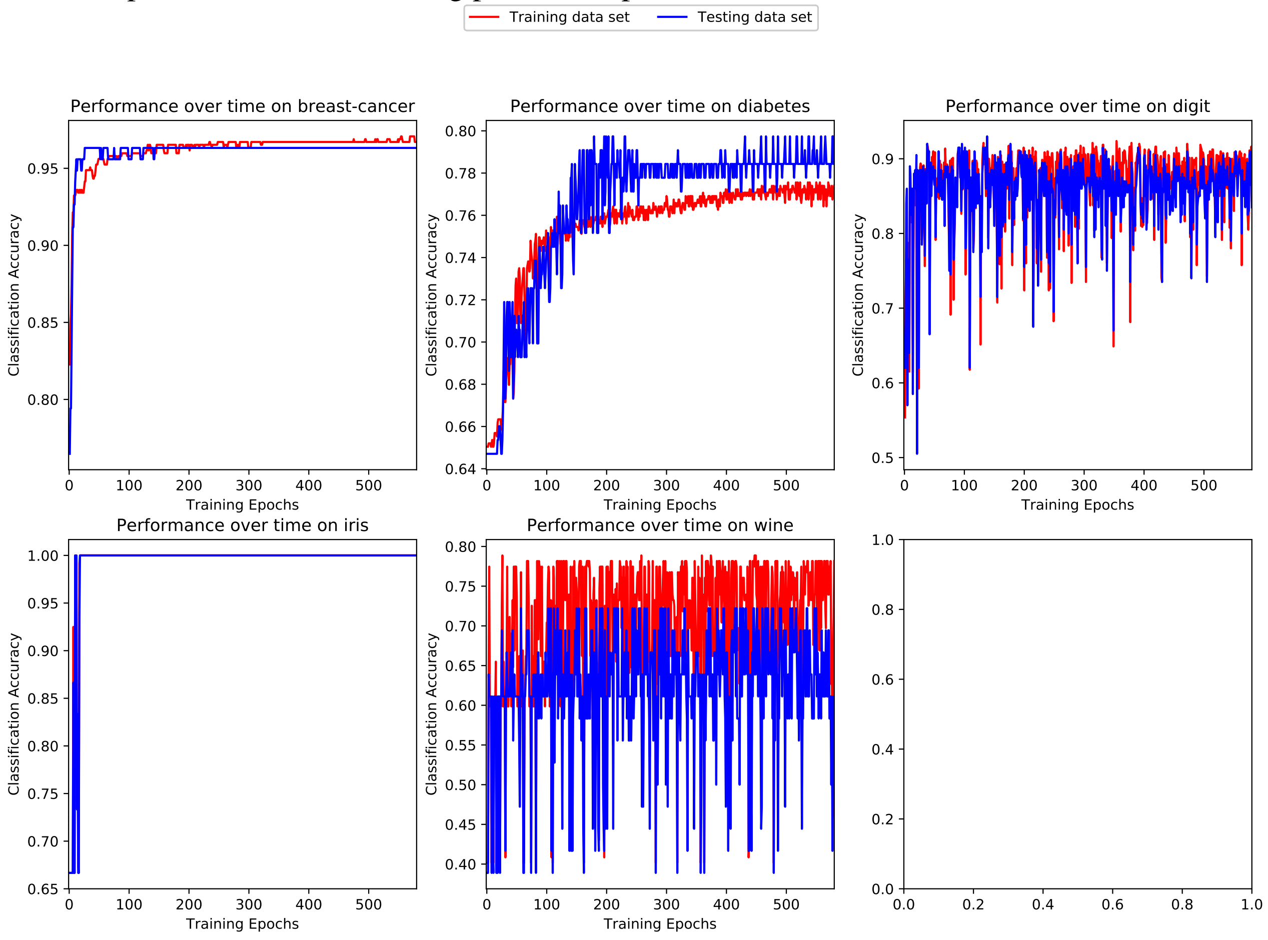
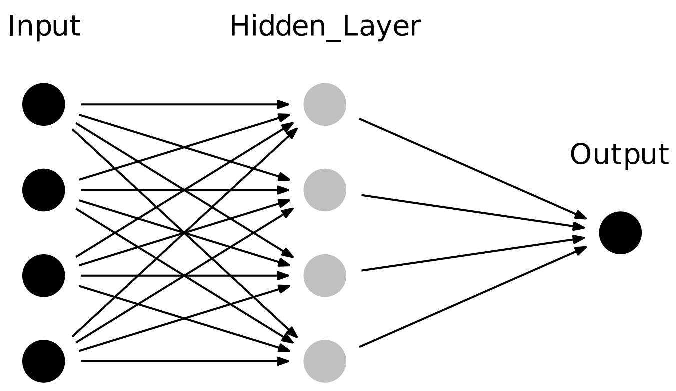


Figure 3: A tow-layer network for binary classification problem. The number of input units is 4(=feature number). The number of output units is 1, their values are the posterior to corresponding classes.



4 Single-hidden-layer Neural Network Model

1. Principle

   (a) For the problem described in the project, we need to deal with a binary classification problem with a singal-hidden-layer neural network model. The network can be visulized in Fig.1.

   (b) Input-to-hidden:

   $$z_h^{(\ell)} = sigmoid(w_h^T x^{(\ell)}) = \frac{1}{1 + \exp[-(\sum_{j=1}^{d} w_{hj} x_j^{(\ell)} + w_{h0})]}$$

   (c) Hidden-to-output: similar to Input-to-hidden:

   $$y_i^{(\ell)} = sigmoid(v_i^T z^{(\ell)}) = \frac{1}{1 + \exp[-(\sum_{h=1}^{H} v_{ih} z_h^{(\ell)} + v_{i0})]}$$

   (d) Cross-entropy: Given a sample datasetχ with size N, the cross-entropy is

   $$E[W, v|\chi] = -\sum_{\ell=1}^{N} [r^{(\ell)} \log y^{(\ell)} + (1 - r^{(\ell)}) \log(1 - y^{(\ell)})]$$

   In the learning process, the cross-entropy should be minimized.

   (e) Gradient-Descent Learning:

      i. Update rule for hidden-to-output weights:

      $$\Delta v_h = -\eta \frac{\partial E}{\partial v_h}$$

Table 3: Experiment results of neural network on the given data set with different H

Dataset	breast-cancer	diabetes	digit	iris	wine
m_i	1000	1000	1000	1000	1000
c	0.0001	0.0001	0.0001	0.0001	0.0001
η(adaptive)	0.1	0.1	0.1	0.1	0.1
p	l2	l2	l2	l2	l2
a(%)(H=1)	96.90	72.03	87.75	100.0	42.07
a(%)(H=2)	96.36	71.54	82.38	100.0	62.07
a(%)(H=3)	96.73	74.47	82.0	100.0	62.00
a(%)(H=4)	**97.45**	**75.77**	89.75	100.0	62.76
a(%)(H=5)	97.27	77.56	90.0	100.0	56.55
a(%)(H=6)	95.64	66.83	92.125	100.0	62.76
a(%)(H=7)	97.09	75.61	91.63	100.0	62.75
a(%)(H=8)	96.18	76.59	94.25	**100.0**	54.48
a(%)(H=9)	96.55	77.89	**94.00**	100.0	**68.28**
a(%)(H=10)	96.55	74.96	94.13	69.17	64.14

ii. Update rule for input-to-hidden weights: Applying the chain runle to calculate the gradient.

$$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}} = -\eta \sum_{\ell} \frac{\partial E}{\partial y^{(\ell)}} \frac{\partial y^{(\ell)}}{\partial z_h^{(\ell)}} \frac{\partial z_h^{(\ell)}}{\partial w_{hj}}$$

2. Experiment

(a) Experiment Description: As descripted above in section 3.2, we will try out different $\eta(< 1), c, m\_i$ in the learning process on the given data sets. Additionally, we also try different candidate value $H \in \{1, 2, ..., 10\}$(number of hidden units) in the cross validation to find H^* with best performance among the 10 choices of H can be found. In the implementation, we use the `MLPClassifier` classifier and set `learning_rate='adaptive'`.

(b) Result:

i. Choose H^*: We firstly select proper m_i, c, η, p, and then change implement a cross validation(randomly sampling 80% of the training instances to train a classifier and then testing it on the remaining 20%) on the given data sets with different H. Then we could determine the H^* by considering the classfication accuracy(Table 3) and loss curve(Fig.4). If we choose H^* only by the classfication accuracy in Table 3, H^* should be $4/4/8/1/9$ respectively on the corresponding data sets. However, we can observe the loss curve(Fig.4), if we choose $H^* = 9/8$ for iris, wine data sets respectively, the loss after learning should be reduced evidently, while the accuracy does not change a lot. So we choose $H^* = 4/4/9/8/9$.

ii. After choosing H^*, we test this model on the full testing data set and observe its performance(classification accuracy, loss and time required) H^*(please refer to Table 4).

Figure 4: Loss curve in the learning process on given data sets with different H

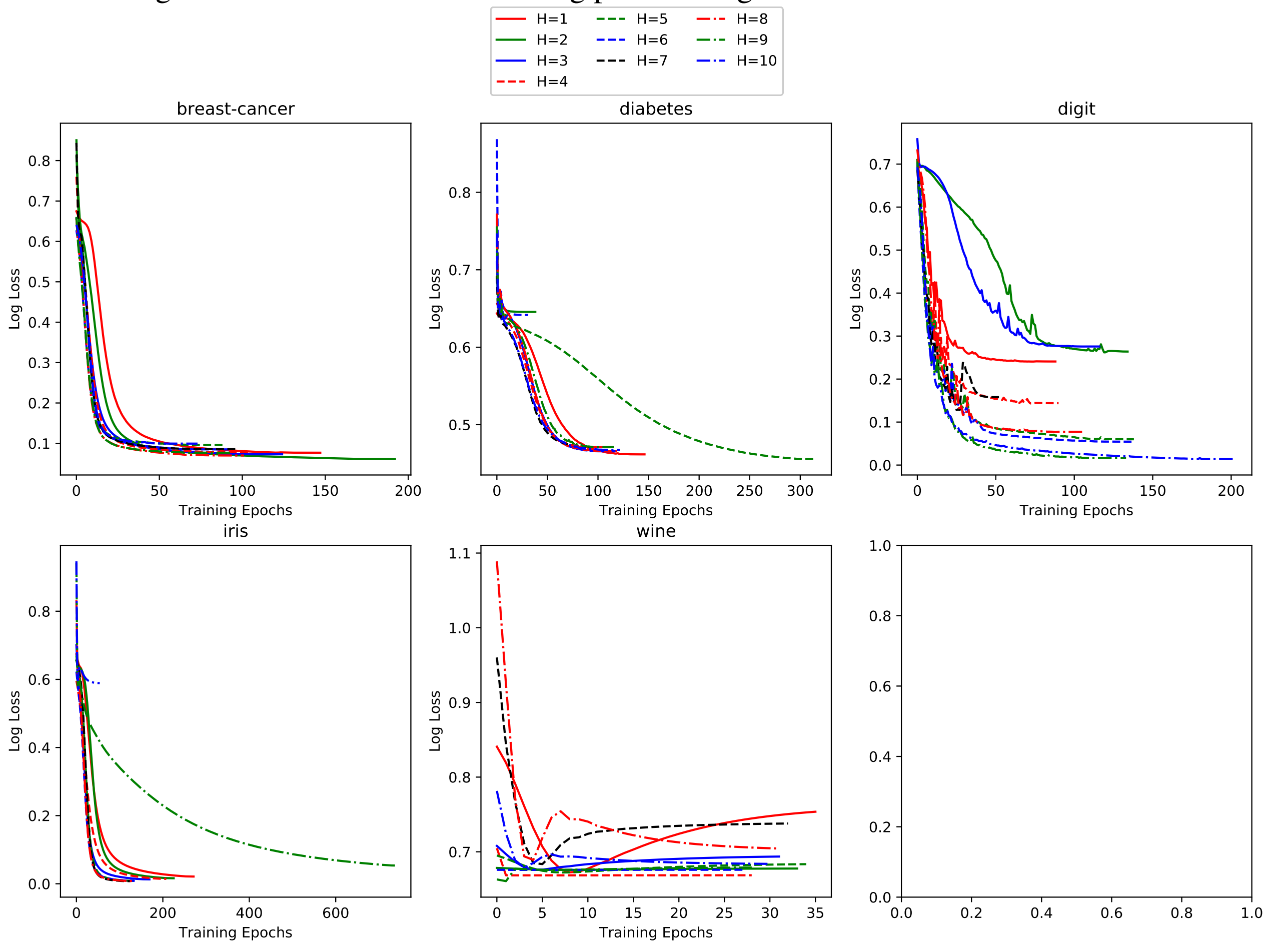
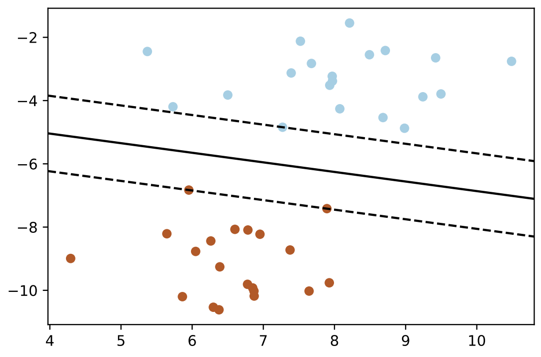


Table 4: Experiment results of neural network model with H^*

Dataset	m_i	c	η	p	H^*	$a_{train}(\%)$	$a_{test}(\%)$	l_{train}	l_{test}	$time(ms)$
breast-cancer	1000	0.001	0.1	12	3	97.06	97.06	0.09232	0.08524	27.06
diabetes	1000	0.001	0.1	12	9	79.01	79.74	0.4797	0.4977	48.97
digit	1000	0.001	0.1	12	10	95.78	95.0	0.04641	0.1104	74.49
iris	1000	0.001	0.1	12	8	100.0	100.0	0.01631	0.02091	14.76
wine	1000	0.001	0.1	12	4	68.86	61.11	0.6742	0.6701	6.671

Figure 5: A binary classification(dimension of features$= 2$) problem is visualized that it is solved by the SVM model. The central line represents a separating hyperplane. There are two red points and one blue point on the dot line, which means that they satisfy this equation: $|w^T x + w_0| = 1$ and are called closest data point. Margin is defined as the distance between the separating hyperplane and the closet points. Under the constraint of hard-margin case, we need to find a canonical optimal separating hyperplane that maximum the margin.



5 Support Vector Machine

1. Principle

   (a) For the problem described in the project, we need to deal with a binary classification problem with a SVM model to find a optimal hyperplane. It can be visulized in Fig.3.

   (b) Primal Optimization Problem:

      i. Margin can be given: $\gamma = \frac{1}{\|w\|}$. Maximizing the margin is equivalent to minimizing $\|w\|$.

      ii. Inequality constraints: For all data points in the sample $\chi = \{(x^\ell, y^\ell)\}$, we want w and w_0 to satisfy:

$$w^T x^{(\ell)} + w_0 = \begin{cases} \geqslant +1 & if \;\; y^{(\ell)} + 1 \\ \leqslant -1 & if \;\; y^{(\ell)} - 1 \end{cases}$$

      Equivalent form of inequality constraints:

$$y^{(\ell)}(w^T x^{(\ell)} + w_0) \geqslant 1$$

      iii. Primal optimization problem:

$$\text{Minimize} \;\; \frac{1}{2}\|w\|^2$$

$$\text{subject to} \;\; y^{(\ell)}(w^T x^{(\ell)} + w_0) \geqslant 1, \forall \ell$$

      Lagrangian:

$$L_p(w, w_0, \{\alpha_\ell\}) = \frac{1}{2} w^T w - w^T \sum_\ell^N \alpha_\ell y^{(\ell)} x^{(\ell)} - w_0 \sum_\ell^N \alpha_\ell y^{(\ell)} + \sum_\ell^N \alpha_\ell, \; \alpha_\ell \geqslant 0 \tag{1}$$

Table 5: Experiment results of SVM(linear kernel) on the given data set, l_{test} is defined as the cross-entropy.

Dataset	c	γ	$a_{train}(\%)$	$a_{test}(\%)$	l_{train}	l_{test}	$time(ms)$
breast-cancer	0.0001	1	97.07	97.06	0.0849	0.0711	5.088
diabetes	0.0001	0.1	76.75	78.43	0.4722	0.4967	16.841
digit	0.0001	0.001	62.25	62.0	0.6324	0.6253	104.701
iris	0.0001	0.1	100.0	100.0	0.0215	0.033	1.228
wine	0.0001	0.01	94.37	94.44	0.2801	0.3372	3.4208

iv. Solution: the optimal solution is a saddle point which minimizes L_p w.r.t the primal variables w, w_0 and maximizes L_p w.r.t the dual variables α_ℓ.

(c) Dual Optimization Problem

i. In optimization theory, it is very common and sometimes advantageous to turn a primal problem into a dual problem and then solve the latter instead. In our case, it also turns out to be more convenient to solve the dual problem (whose complexity depends on the sample size N) rather than the primal problem directly (whose complexity depends on the dimensionality d). The dual problem also makes it easy for a nonlinear extension using kernel functions.

ii. Eliminating primal variables:

$$\frac{\partial L_p}{\partial w} = 0 \Rightarrow w = \sum_{\ell}^{N} \alpha_\ell y^{(\ell)} x^{(\ell)} \tag{2}$$

$$\frac{\partial L_p}{\partial w_0} = 0 \Rightarrow \sum_{\ell}^{N} \alpha_\ell y^{(\ell)} = 0 \tag{3}$$

iii. Dual optimization problem: Plugging (2) and (3) into L_p gives the objective function L_d and the constraint:

$$\text{Minimize} \quad \sum_{\ell}^{N} \alpha_\ell - \sum_{\ell}^{N} \sum_{\ell'}^{N} \alpha_\ell \alpha_{\ell'} y^{(\ell)} y^{(\ell')} (x^{(\ell)})^T x^{(\ell')}$$

$$\text{subject to} \quad \sum_{\ell}^{N} y^{(\ell)} = 0 \text{ and } \alpha_\ell \geqslant 0, \forall \ell$$

2. Experiment

(a) SVM with linear kernel

i. Experiment Description: We will implement SVM with linear kernel for this classification problem. In the implementation, we use the `svm.SVC` classifier and set `kernel='linear', C=1.0(default).`

ii. Result: Please refer to Table 5.

(b) SVM with RBF kernel

i. Experiment Description: We will try out all candidate $\gamma \in \{1, 10^{-1}, 10^{-2}, 10^{-3}\}$ in the cross validation to find γ^* with best performance among the 4 choices of γ can be found. `svm.SVC` classifier and set `kernel='rbf', C=1.0(default).`

Table 6: Experiment results of SVM(RBF kernel) on the given data set with different γ, a is the classification accuracy on testing instances, l_{test} is defined as the cross-entropy.

Dataset	breast-cancer	diabetes	digit	iris	wine
c	0.0001	0.0001	0.0001	0.0001	0.0001
$a(\%)(\gamma = 1)/l_{test}$	**97.73**/ 0.09427	76.91/ 0.4791	51.5/ 0.6942	100.0/ 0.02232	55.86/ 0.6949
$a(\%)(\gamma = 0.1)/l_{test}$	96.00/ 0.1061	**76.26**/ 0.4779	49.38/ 0.6943	**100.0**/ 0.01753	64.14/ 0.6140
$a(\%)(\gamma = 0.01)/l_{test}$	96.36/ 0.1010	63.25/ 0.4995	92.50/ 0.07318	100.0/ 0.02836	**75.17**/ 0.5023
$a(\%)(\gamma = 0.001)/l_{test}$	94.00/ 0.1035	67.80/ 0.4990	**99.63**/ 0.01609	68.33/ 0.02089	73.79/ 0.5197

Table 7: Experiment results of SVM(RBF kernel) on the given data set with γ^*, l_{test} is defined as the cross-entropy.

Dataset	c	γ	$a_{train}(\%)$	$a_{test}(\%)$	l_{train}	l_{test}	$time(ms)$
breast-cancer	0.0001	1	98.17	96.32	0.0407	0.0928	19.2912
diabetes	0.0001	0.1	77.72	78.43	0.464	0.4996	39.5187
digit	0.0001	0.001	100.0	100.0	0.003	0.0148	140.7897
iris	0.0001	0.1	100.0	100.0	0.015	0.0222	0.6443
wine	0.0001	0.01	97.89	77.78	0.0942	0.4921	4.9481

ii. Result:

A. Choosing γ^*: We firstly select $c = 0.0001$ and RBF kernel as the kernel function, and then implement a cross validation(randomly sampling 80% of the training instances to train a classifier and then testing it on the remaining 20%) on the given data sets with different γ. Then we could determine the γ^* by considering the classfication accuracy(Table 6) and l_{test}(defined as the sum cross-entropy). If we choose γ^* only by the classfication accuracy, γ^* should be $1/1/0.001/0.1/0.01$ respectively on the corresponding data sets. However, we can observe the l_{test}, if we choose $\gamma^* = 0.1$ for diabetes data sets, the l_{test} is samller, while the accuracy does not change a lot. So we choose $\gamma^* = 1/0.1/0.001/0.1/0.01$.

B. After choosing γ^*, we test this model on the full testing data set and observe its performance(classification accuracy, loss and time required) γ^*(please refer to Table 7).

Figure 6: Confusion matrix of one result of logistic regression model with paramater:$c = 0.01, \eta = 0.1, p = l2$).

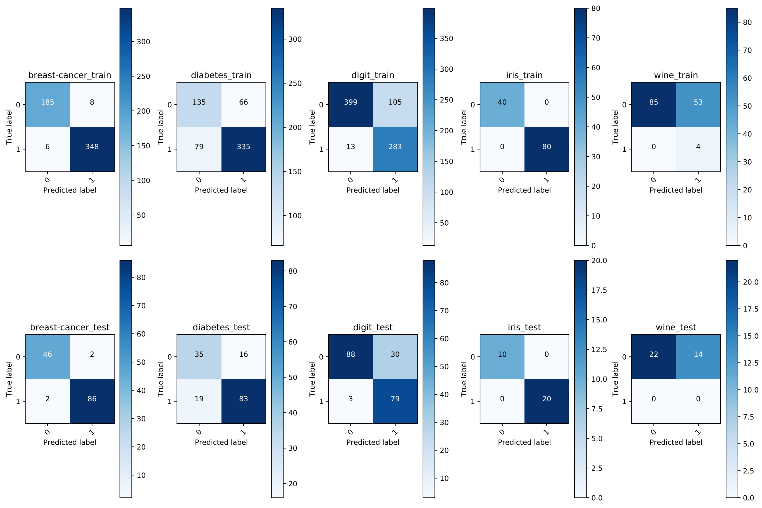
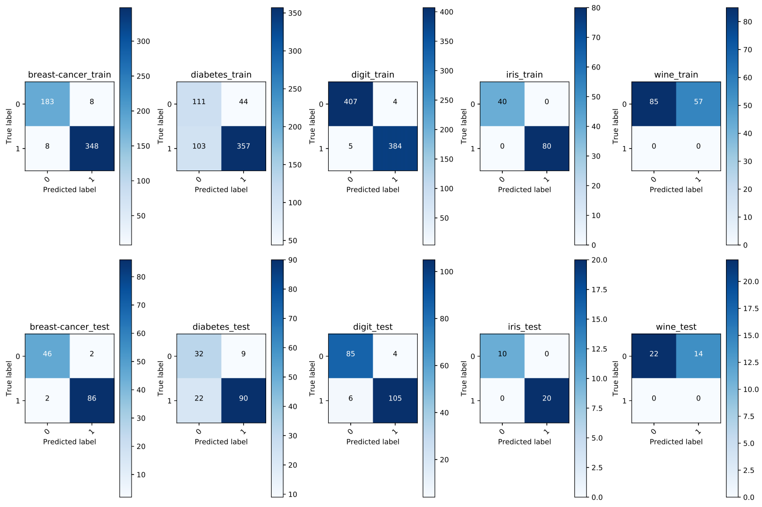


Figure 7: Confusion matrix of one result of neural network model with paramater:$c = 0.001, \eta = 0.1, H = H^*$).



6 Comparison between different models

1. Confusiton Matrix
   (a) Logistic Regression: Fig.6
   (b) Neural Network: Fig.7
   (c) SVM with linear kernel: Fig.8
   (d) SVM with RBF kernel: Fig.9
2. Accuracy Comparison: Fig.10

Figure 8: Confusion matrix of one result of SVM(linear kernel) with paramater:$c = 0.0001$.

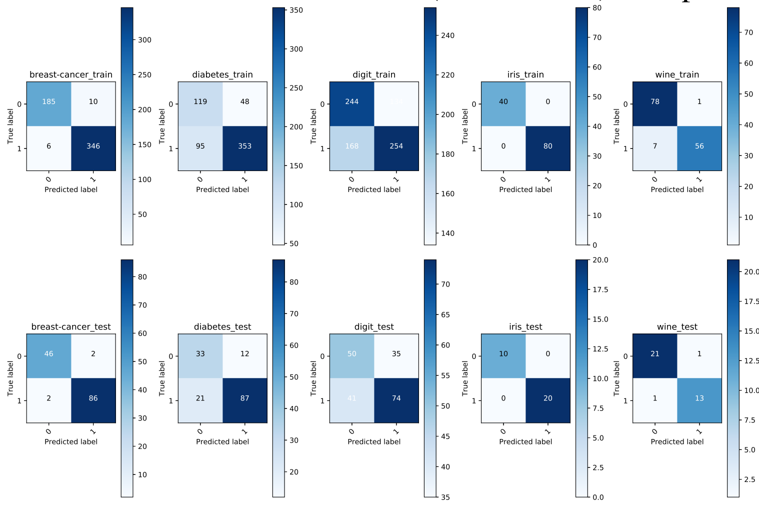


Figure 9: Confusion matrix of one result of SVM(RBF kernel) with paramater:$c = 0.0001, \gamma = \gamma^*$.

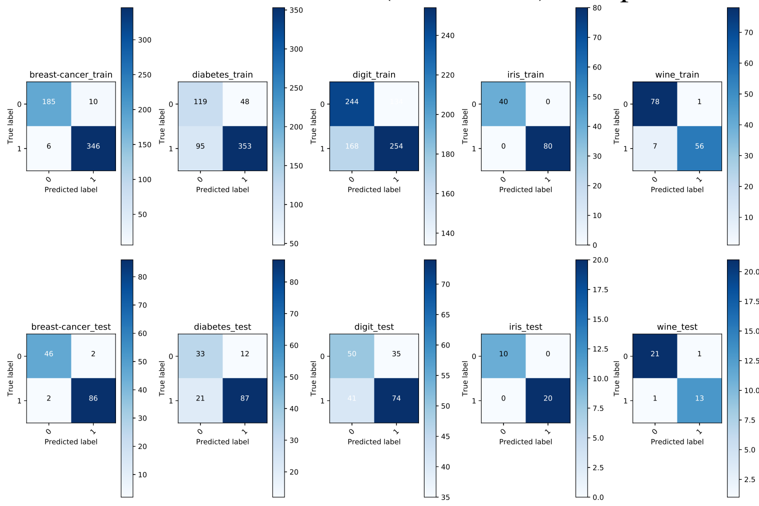
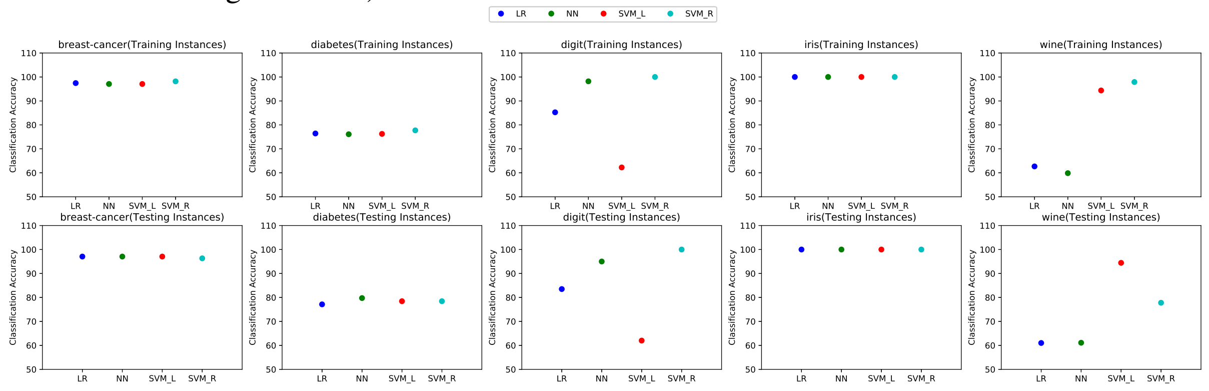


Figure 10: Accuracy comparison between different models on the given data sets(both training instances and testing instances).



7 Conclusion

According to the above figures, we can easily observe that all models have high classification accuracy on the breast-cancer, diabetes and iris data sets, especially on the breast-cancer and iris data sets. In contrast, the digit and wine date sets seem to be more challengous. On the digit data sets, SVM model with linear kernel performs lower accuracy than others, but on wine data sets(testing instances), it has higher accuracy. SVM model with RBF kernel performs high accuracy on all data sets(including training & testing instances). In summary, if the four models are sorted by classification accuracy, the order should be SVM_L < LR < NN < SVM_R. By if they are sorted by the time required, the order should be LR < NN < SVM_L < SVM_R. So we can guess that the accuracy is related to the time required.