

# 10701 Homework 2 Solutions

## 1 SVM [Hanghang, 20 points]

- 1.1 (**Kernel, 10 pts**) Given the following dataset in 1-d space (Figure 1), which consists of 3 positive data points  $\{-1, 0, 1\}$  and 3 negative data points  $\{-3, -2, 2\}$ .

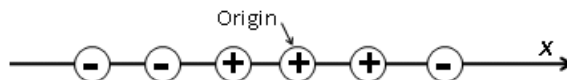


Figure 1: Dataset

- (1) Find a feature map  $\{\mathbf{R}^1 \rightarrow \mathbf{R}^2\}$ , which will map the original 1-d data points to 2-d space so that the positive set and negative set are linearly separable with each other. Plot the dataset after mapping in 2-d space.

**Solutions:**  $x \rightarrow \{x, x^2\}$ . See figure 2.

- (2) In your plot, draw the decision boundary given by hard-margin linear SVM. Mark the corresponding support vector(s).

**Solutions:** See figure 2 (supported vectors are shadowed ones).

- (3) For the feature map you choose, what is the corresponding kernel  $K(x_1, x_2)$ ?

**Solutions:**  $K(x_1, x_2) = x_1x_2 + (x_1x_2)^2$ .

- 1.2 (**Hinge Loss, 5 pts**) Given  $m$  training data points  $\{x_i, y_i\}_{i=1}^m$ , remember that the soft-margin linear SVM can be formalized as the following constrained quadratic optimization problem:

$$\begin{aligned} \operatorname{argmin}_{\{w, b\}} \quad & \frac{1}{2} w^t w + C \sum_1^m \epsilon_i \\ \text{Subject to: } & y_i(w^t x_i + b) \geq 1 - \epsilon_i \\ & \epsilon_i \geq 0 \quad \forall i \end{aligned} \tag{1}$$

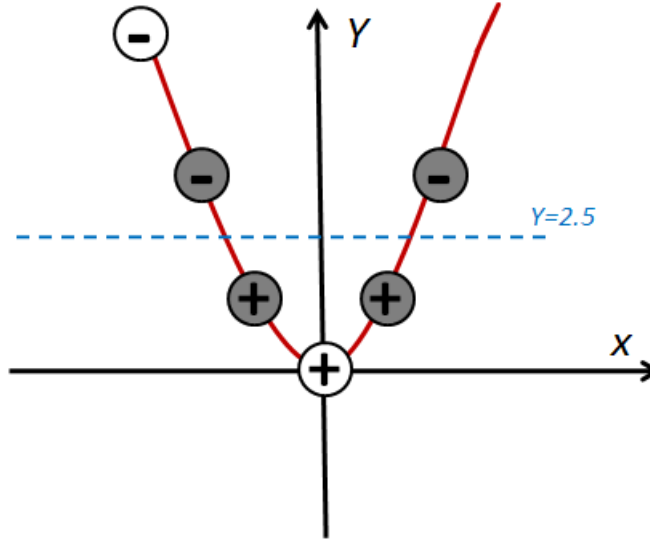


Figure 2: Feature Map

- (1) Prove that the above formulation is equivalent to the following unconstrained quadratic optimization problem:

$$\operatorname{argmin}_{\{w,b\}} w^t w + \lambda \sum_1^m \max(1 - y_i(w^t x_i + b), 0) \quad (2)$$

**Solutions:** By eq. 1, we have  $\epsilon_i \geq \max(1 - y_i(w^t x_i + b), 0)$ ,  $\forall i$ . Since we want to minimize  $\epsilon_i$ , we must have  $\epsilon_i = \max(1 - y_i(w^t x_i + b), 0)$ ,  $\forall i$ . Plus this into eq. 1, we have eq. (2).

- (2) What is your intuition for this new optimization formulation (1 or 2 sentences)?

**Solutions:** Soft-margin SVM tries to balance the simplicity of classifier (the first term) vs. the good prediction on training dataset (the second term).

- (3) What is the value for  $\lambda$  (as a function of  $C$ )?

**Solutions:**  $\lambda = 2C$ .

- 1.3 (SMO, 5 pts) Suppose we are given 4 data points in 2-d space:  $x_1 = (0, 1)$ ,  $y_1 = -1$ ;  $x_2 = (2, 0)$ ,  $y_2 = +1$ ;  $x_3 = (1, 0)$ ,  $y_3 = +1$ ; and  $x_4 =$

$(0, 2)$ ,  $y_4 = -1$ . We will use these 4 data points to train a soft-margin linear SVM. Let  $\alpha_1, \alpha_2, \alpha_3, \alpha_4$  be the Lagrange multipliers for  $x_1, x_2, x_3, x_4$  respectively. And also let the regularization parameter  $C$  be 100.

- (1) Write down the dual optimization formulation for this problem

**Solutions:**

$$\begin{aligned} \operatorname{argmax}_{\alpha_1, \alpha_2, \alpha_3, \alpha_4} & \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - 1/2(\alpha_1^2 + 4\alpha_2^2 + \alpha_3^2 + 4\alpha_4^2 + 4\alpha_2\alpha_3 + 4\alpha_1\alpha_4) \\ \text{Subject to:} & 0 \leq \alpha_1, \alpha_2, \alpha_3, \alpha_4 \leq 100 \\ & -\alpha_1 + \alpha_2 + \alpha_3 - \alpha_4 = 0 \end{aligned} \quad (3)$$

- (2) Suppose we initialize  $\alpha_1 = 5, \alpha_2 = 4, \alpha_3 = 8, \alpha_4 = 7$ . And we want to update  $\alpha_1$  and  $\alpha_4$  (keep  $\alpha_2$  and  $\alpha_3$  fixed) in the 1st iteration. Derive the update equations for  $\alpha_1$  and  $\alpha_4$  (in terms of  $\alpha_2$  and  $\alpha_3$ ). What are the values for  $\alpha_1$  and  $\alpha_4$  after update?

**Solutions:**  $\alpha_1 = \alpha_2 + \alpha_3 = 12$  and  $\alpha_4 = 0$ .

- (3) Now fix  $\alpha_1$  and  $\alpha_4$ , derive the update equations for  $\alpha_2$  and  $\alpha_3$  (in terms of  $\alpha_1$  and  $\alpha_4$ ). What are the values for  $\alpha_2$  and  $\alpha_3$  after update?

**Solutions:**  $\alpha_3 = \alpha_1 + \alpha_4 = 12$  and  $\alpha_2 = 0$ .

## 2 Neural Networks [25 Points, Mark]

### 2.1 Decision Boundaries

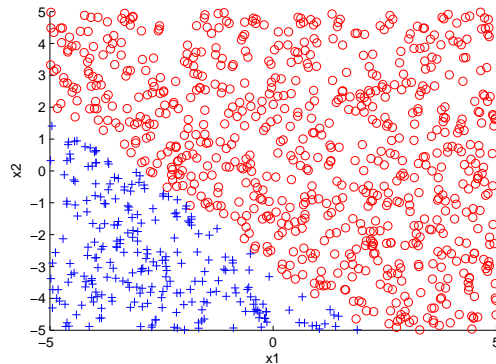


Figure 3: 2 classes of data with a linear decision boundary.

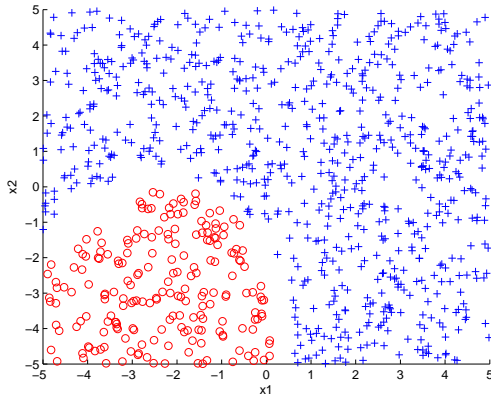


Figure 4: 2 classes of data with a non-linear decision boundary.

Neural networks can form many different types of decision boundaries for classification tasks. The boundaries can be both linear and non-linear, depending on the network structure. In this problem, we will study how different network structures affect the types of boundaries we can create. To gain some intuition, consider a simple linear boundary produced by a 1-node neural network. Our network contains 2 inputs, no hidden units, and only a single output logistic unit. Therefore, the network output is the probability that point  $x_1, x_2$  is in class one  $C_1$ :

$$\Pr(C_1|x_1, x_2) = \frac{1}{1 + \exp^{-(w_0+x_1w_1+x_2w_2)}}$$

where  $x_1, x_2$  are input points and  $w_0, w_1, w_2$  are the weights on the output node. Since the output is a probability, it will always yield a value between 0 and 1. Therefore, we choose class one if the output is  $\geq 0.5$ , and class zero if the output is  $< 0.5$ . For example, if we set the weights to  $(w_0 = 3, w_1 = 1, w_2 = 1)$ , then we will get the linear decision surface in Figure (3). The red circles indicate class one. Make sure you understand this example before continuing.

Now, if we add hidden units to our network, then we can create more complicated non-linear decision boundaries. Consider the data in Figure (4). Once again, there are two data classes but this time they are separated by a non-linear decision boundary. In this problem, we will show that a simple neural network with non-linear hidden units can learn this boundary.

In this problem we will use the 2-layer neural network as shown in Figure (5). Our network has 2 inputs, 2 hidden units, and a single output unit. The

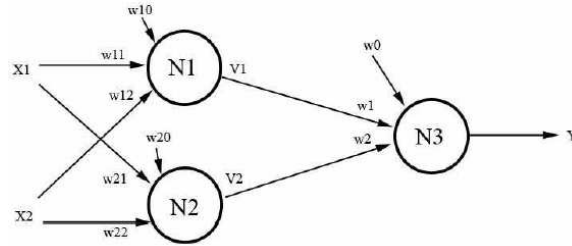


Figure 5: A 2-layer neural network with 2 hidden units and 1 output unit.

activation function of both the hidden and output units is the logistic function:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

1. (1 pt) The network contains 9 weights:  $w_0, w_1, w_2, w_{10}, w_{11}, w_{12}, w_{20}, w_{21}, w_{22}$ . Write out the hidden unit activations  $V_1$  and  $V_2$  as a function of the weights and the inputs  $X_1$  and  $X_2$ . Write out the output unit activation as a function of the weights and  $V_1$  and  $V_2$ .

**Solution:**

$$V_1 = \sigma(w_{10} + w_{11}X_1 + w_{12}X_2)$$

$$V_2 = \sigma(w_{20} + w_{21}X_1 + w_{22}X_2)$$

$$Y = \sigma(w_0 + w_1V_1 + w_2V_2)$$

2. (4 pts) Find nine weights  $w_0, w_1, w_2, w_{10}, w_{11}, w_{12}, w_{20}, w_{21}, w_{22}$  to produce the decision boundary in Figure (4). *Hint:* It may help to load the figure `neural_net_boundary.fig` into MATLAB when fitting your boundary. Be aware there is no data for this problem, your goal rather is to implement the network output function and manually play with the weights to gain intuition about the boundaries.

**Solution:**

$$w_0 = -0.1, w_1 = 1, w_2 = -1, w_{10} = 1, w_{11} = 1, w_{12} = -1, w_{20} = 2, w_{21} = 2, w_{22} = 0.1$$

3. (4 pts) The output of your network will be a number between 0 and 1 where 0.5 is the decision boundary. Using the weights you learned in the previous question, sample 2000 points uniformly in the square  $x_1 \in \{-5, 5\}, x_2 \in \{-5, 5\}$ , and classify them using your network. Plot the results using different symbols (i.e. + for class zero and O for class one) to indicate the class membership.

**Solution:**

Same as figure.

## 2.2 Backpropagation

In this problem we will use backpropagation to compute the learning rules for a standard 2-layer neural network. Our network has one hidden layer and two sets of weights (one set between the inputs and the hidden layer, and another set between the hidden layer and the output layer).

Imagine that we have  $D$  inputs,  $H$  hidden units, and  $K$  output units. The output of each unit (both hidden and output) is given by:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

Each of the  $K$  outputs is an independent classification task. With  $N$  examples in the training data, this leads to an error function for the whole network:

$$E(\mathbf{w}) = - \sum_{n=1}^N \sum_{k=1}^K \{t_{nk} \ln y_{nk} + (1 - t_{nk}) \ln (1 - y_{nk})\}$$

where  $t_{nk}$  is the training target for the  $n^{\text{th}}$  example and  $k^{\text{th}}$  output. Similarly,  $y_{nk}$  is the network response for output  $k$  when the  $n^{\text{th}}$  example is fed into the network. In this problem, we will compute the weight update rule using *stochastic gradient descent* like you did in Homework 1. The generic gradient update for a weight coming from unit (or input)  $i$  to unit  $j$  is:

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} - \eta \frac{\partial E_n}{\partial w_{ji}^{(t-1)}}$$

where  $E_n$  is the error function at a single example. Now we define the input activity of a unit  $j$  as:

$$a_j = \sum_i w_{ji} z_i$$

where  $z_i$  is the output activation of unit (or input)  $i$ . Since we are computing the update for a single example, we can simplify notation by not including  $n$  in our update rules. Unless otherwise noted, assume there are no bias weights in the units.

1. (1 pt) How many parameters are there in the network (for this question, assume units have a bias term)?

**Solution:**

$$H(D + K) + (H + K)$$

2. (1 pt) What is the derivative of the activation function  $\sigma(a)$ ?

**Solution:**

$$\sigma(a)(1 - \sigma(a))$$

3. (2 pts) Define  $\frac{\partial E_n}{\partial a_j} = \delta_j$ . Show that  $\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$

**Solution:**

$$\begin{aligned} \frac{\partial E_n}{\partial w_{ji}} &= \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \\ &= \delta_j \frac{\partial}{\partial w_{ji}} \sum_i w_{ji} z_i \\ &= \delta_j z_i \end{aligned}$$

4. (3 pts) Show that for each of the  $k$  logistic output units, that  $\delta_k = (y_k - t_k)$  when  $E_n$  is the cross-entropy error function above (for a single example).

**Solution:**

The error function for a single example  $n$  is given by:

$$E_n = - \sum_{k=1}^K \{t_k \ln y_k + (1 - t_k) \ln (1 - y_k)\}$$

Now remember that the output activation  $y_k$  is just  $\sigma(a_k)$  where  $a_k$  is the input to the output unit  $k$ . Now we need to compute  $\delta_k$  which is the derivative of  $E_n$  with respect to  $a_k$ :

$$\begin{aligned} \delta_k = \frac{\partial E_n}{\partial a_k} &= - \left\{ \frac{t_k}{\sigma(a_k)} \sigma(a_k) (1 - \sigma(a_k)) + \frac{(1 - t_k)}{(1 - \sigma(a_k))} (-1) \sigma(a_k) (1 - \sigma(a_k)) \right\} \\ &= - \left\{ t_k (1 - \sigma(a_k)) - (1 - t_k) \sigma(a_k) \right\} \\ &= - \left\{ t_k - \sigma(a_k) t_k - \sigma(a_k) + \sigma(a_k) t_k \right\} \\ &= - \left\{ t_k - \sigma(a_k) \right\} \\ &= \left\{ \sigma(a_k) - t_k \right\} \\ &= \left\{ y_k - t_k \right\} \end{aligned}$$

5. (3 pts) Let  $x_i, i = 1 \dots D$ , be input  $i$ . Compute the update rule  $w_{kj}^{(t)}$  for a weight between hidden unit  $j$  and output unit  $k$  in terms of the inputs  $x_i$  and weights.

**Solution:**

Remember that the update rule for  $w_{kj}$  is:

$$\begin{aligned} w_{kj}^{(t)} &= w_{kj}^{(t-1)} - \eta \frac{\partial E_n}{\partial w_{kj}^{(t-1)}} \\ &= w_{kj}^{(t-1)} - \eta \delta_k z_j \\ &= w_{kj}^{(t-1)} - \eta (y_k - t_k) z_j \end{aligned}$$

And now we put our answer in terms of the inputs:

$$= w_{kj}^{(t-1)} - \eta (y_k - t_k) \sigma \left( \sum_{i=1}^D w_{ji}^{t-1} x_i \right)$$

where  $y_k$  is:

$$y_k = \sigma \left( \sum_{j=1}^H w_{kj}^{t-1} \sigma \left( \sum_{i=1}^D w_{ji}^{t-1} x_i \right) \right)$$

6. (3 pts) For each of the  $j$  logistic hidden units,  $\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j}$ . Show that  $\delta_j = \sigma(a_j)(1 - \sigma(a_j)) \sum_k \delta_k w_{kj}$

**Solution:**

$$\begin{aligned} \delta_j = \frac{\partial E_n}{\partial a_j} &= \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \\ &= \sum_k \delta_k \frac{\partial a_k}{\partial a_j} \end{aligned}$$

Now remember:

$$\begin{aligned} a_k &= \sum_j w_{kj} z_j \\ a_k &= \sum_j w_{kj} \sigma(a_j) \end{aligned}$$

So taking the derivative wrt  $a_j$ :

$$\begin{aligned} a_k &= \sum_j w_{kj} z_j \\ \frac{\partial a_k}{\partial a_j} &= w_{kj} \frac{\partial}{\partial a_j} \sigma(a_j) \\ &= w_{kj} \sigma(a_j)(1 - \sigma(a_j)) \end{aligned}$$

Notice that the  $\sigma(a_j)$  terms don't depend on  $k$ , so we can substitute and pull them out of the sum:

$$\begin{aligned} \delta_j &= \sum_k \delta_k \frac{\partial a_k}{\partial a_j} \\ &= \sum_k \delta_k w_{kj} \sigma(a_j)(1 - \sigma(a_j)) \\ &= \sigma(a_j)(1 - \sigma(a_j)) \sum_k \delta_k w_{kj} \end{aligned}$$

7. (3 pts) Let  $x_i, i = 1 \dots D$ , be input  $i$ . Compute the update rule  $w_{ji}^{(t)}$  for a weight between input  $i$  and hidden unit  $j$  in terms of the inputs  $x_i$  and weights.

**Solution:**

Similar to part 5:

$$\begin{aligned} w_{ji}^{(t)} &= w_{ji}^{(t-1)} - \eta \frac{\partial E_n}{\partial w_{ji}^{(t-1)}} \\ &= w_{ji}^{(t-1)} - \eta \delta_j x_i \\ &= w_{ji}^{(t-1)} - \eta \left( \sigma(a_j)(1 - \sigma(a_j)) \sum_k \delta_k w_{kj}^{(t-1)} \right) x_i \end{aligned}$$

And now in terms of inputs  $x$  and outputs  $y_k$ :

$$\begin{aligned} &= w_{ji}^{(t-1)} - \eta \left( \sigma(a_j)(1 - \sigma(a_j)) \sum_k \delta_k w_{kj}^{(t-1)} \right) x_i \\ &= w_{ji}^{(t-1)} - \eta \left( \sigma \left( \sum_{i=1}^D w_{ji}^{(t-1)} x_i \right) (1 - \sigma \left( \sum_{i=1}^D w_{ji}^{(t-1)} x_i \right)) \sum_k \delta_k w_{kj}^{(t-1)} \right) x_i \\ &= w_{ji}^{(t-1)} - \eta \left( \sigma \left( \sum_{i=1}^D w_{ji}^{(t-1)} x_i \right) (1 - \sigma \left( \sum_{i=1}^D w_{ji}^{(t-1)} x_i \right)) \sum_k (y_k - t_k) w_{kj}^{(t-1)} \right) x_i \end{aligned}$$

### 3 Decision Trees [20 Points, Jerry]

1. (14 pts) Bob wants to try a Decision Tree algorithm on a simple dataset listed in Table 1. The dataset has 8 instances each of which has 3 attributes and a class label. The attributes are abstracted as  $X_1, X_2$ , and  $X_3$  and the class label as  $Y$ . All of them are binary variables. Bob knows the framework of the algorithm, but he needs your help on other parts (calculation, drawing, open questions, etc).

Instance	$X_1$	$X_2$	$X_3$	Class Label $Y$
1	0	0	0	0
2	0	0	1	0
3	0	1	0	1
4	0	1	1	1
5	1	0	1	1
6	1	0	1	1
7	1	1	0	0
8	1	1	0	0

Table 1: Training Dataset

Before using the Decision Tree algorithm, you first need to know the idea of *Entropy*. In information theory, *Entropy* is a measure of the uncertainty associated with a random variable. Entropy  $H(X)$  of a random variable  $X$  is defined as:

$$H(X) = - \sum_i P(X = i) \log_2 P(X = i)$$

In binary cases,

$$H(X) = -(P(X = 0) \log_2 P(X = 0) + P(X = 1) \log_2 P(X = 1))$$

- (a) (2 pts) What is the value of the Entropy of class labels,  $H(Y)$ , in the dataset?

**Solution:**

$$\begin{aligned} P(Y = 0) &= P(Y = 1) = 0.5 \\ H(Y) &= -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1 \end{aligned}$$

A decision tree is a classifier that can be drawn in a tree structure. Each internal node of the tree is associated with an attribute (from the dataset), and the branches of the node correspond to the values of the attribute. Each leaf node has a prediction on class label. We can see that when attributes in the dataset are binary, we will always get binary trees.

One of the most important parts of a Decision Tree algorithm is to pick an attribute for each node. We want to pick an attribute such that the remaining uncertainty is minimized. Bob uses the criterion of *Information Gain*, which is defined as:

$$IG(Y, X) \equiv H(Y) - H(Y | X) = H(Y) - \sum_{v \in \text{Values}(X)} \frac{|Y_v|}{|Y|} H(Y_v)$$

where  $|Y|$  is the number of instances of  $Y$ , and  $Y_v$  is the class labels of instances whose attribute  $X$  are of value  $v$ . (In other words,  $Y_v$  can be viewed as a short form of  $Y|X = v$ )

- (b) (3 pts) What are the Information Gains of  $Y$  with respect to each of the three attributes,  $IG(Y, X_1)$ ,  $IG(Y, X_2)$  and  $IG(Y, X_3)$ ? (Hint:  $\log_2 3 = 1.585$ )

**Solution:**

$$H(Y | X_1) = 1, \quad IG(Y, X_1) = 0$$

$$H(Y | X_2) = 1, \quad IG(Y, X_2) = 0$$

$$\begin{aligned} H(Y | X_3 = 0) &= -\left(\frac{3}{4} \log_2 \frac{3}{4} + \frac{1}{4} \log_2 \frac{1}{4}\right) \\ &= -\frac{1}{4} \left(3 \log 3 + 1 \log 1 - 4 \log 4\right) = 0.811 \end{aligned}$$

$$H(Y | X_3 = 1) = 0.811$$

$$H(Y | X_3) = 0.5 * H(Y | X_3 = 0) + 0.5 * H(Y | X_3 = 1) = 0.811$$

$$IG(Y, X_3) = 0.189$$

- (c) (1 pts) Which attribute should be chosen for the root according to the algorithm? (Hint: Bigger  $IG$  is better.)

**Solution:**  $X_3$

After the attribute for root is chosen, all instances are divided into two groups (or multiple groups in non-binary cases) according to their values of the chosen attribute. Then, the same algorithm is run on each group until (i) all instances in a group have the same class label, or (ii) no attributes can create multiple non-empty sub-groups.

- (d) (4 pts) By running the algorithm recursively, what does the resulting decision tree look like? Please draw it. (Hint: Each internal node should have an attribute for branching; each edge should be marked by an attribute value; and each leaf node should contain a class label as prediction. You don't need to show the calculation for this one.)

**Solution:** See graph below.

- (e) (2 pts) The Decision Tree algorithm is a heuristic method, which does not necessarily generate the simplest tree. Can you draw another decision tree of zero training error that has fewer nodes than the tree produced by the algorithm?

**Solution:** See graph below.

- (f) (1 pts) The algorithm described above can always produce a decision tree classifier with zero training error as long as there is no contradiction in the dataset. However, the test error could be non-zero. Can you give a test instance which does not contradict with the training dataset but will lead to a wrong prediction by the decision tree you plot in part (d)?

**Solution:** (1,0,0,1) or (1,1,1,0)

- (g) (1 pts) Fighting over-fitting is another important part of Decision Tree algorithms. Please list one technique that is usually used in Decision Tree algorithms. (Hint: Answer is not unique.)

**Solution:**

1. Pruning on trees using a validation dataset.
2. Rule post-pruning (C4.5 algorithm)

Instance	$X_1$	$X_2$	$X_3$	Class Label $Y$
1	0	0	1	0
2	0	0	2	0
3	0	1	3	1
4	0	1	4	1
5	1	0	5	1
6	1	0	6	1
7	1	1	7	0
8	1	1	8	0

Table 2: Training Dataset 2

2. (6 pts) Now the attribute  $X_3$  is replaced by a new attribute taking integer values as shown in Table 2.

- (a) (2 pts) What will be the new decision tree if we keep the algorithm unchanged? Please draw the tree, but details of calculation are not required.

**Solution:** See graph below.

- (b) (2 pts) Is the new decision tree good or bad? Give a one-sentence reason.

**Solution:** It is bad because it is not robust.

- (c) (2 pts) Try some other criterion for selecting the attributes. Describe your criterion and draw the resulting tree. (Hint: Answer is not unique.)

**Solution:** One example is using *Gain Ratio*, which is defined as:

$$GainRatio(Y, X) \equiv \frac{InfoGain(Y, X)}{SplitInformation(Y, X)}$$

where  $SplitInformation(Y, X) = H(X)$

However, it does not help in this example. In stead, we can use *Threshold Splits*. See Figure 6 below.

## 4 Logistic Regression and Naive Bayes (Suyash, 35 points)

1. Logistic Regression

Number of test misclassifications:

- IRLS misclassification: 4
- Gradient ascent misclassification: 12 (learning rate=1e-3)

## 2. Gaussian Naive Bayes

Number of test misclassifications: 89 Some students reported that using the same variance for both classes gives better results, with fewer misclassifications.

## 3. Variation of performance with training data

Results are shown in Figure 7.

## 4. Boolean NB and LR

Under the assumption of boolean variables  $X$  and  $Y$ , let  $P(Y = 1) = \pi$  and  $P(X_i = 1|Y = j) = \theta_{ij}$  (this is sufficient to define all probabilities in the model). Then we can write  $P(X_i|Y = j) = \theta_{ij}^{X_i} (1 - \theta_{ij})^{(1-X_i)}$ .

$$\begin{aligned} P(Y = 1|X) &= \frac{P(X|Y = 1)P(Y = 1)}{P(X|Y = 1)P(Y = 1) + P(X|Y = 0)P(Y = 0)} \\ &= \frac{1}{1 + \exp\left(\ln \frac{P(X|Y=0)P(Y=0)}{P(X|Y=1)P(Y=1)}\right)} \\ &= \frac{1}{1 + \exp\left(\ln \frac{1-\pi}{\pi} + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)}\right)} \end{aligned}$$

We now expand the second term in the denominator

$$\begin{aligned} \sum_i \ln \frac{P(X_i|Y = 0)}{P(X_i|Y = 1)} &= \sum_i \ln \frac{\theta_{i0}^{X_i} (1 - \theta_{i0})^{(1-X_i)}}{\theta_{i1}^{X_i} (1 - \theta_{i1})^{(1-X_i)}} \\ &= \sum_i X_i \ln \theta_{i0} + (1 - X_i) \ln (1 - \theta_{i0}) - X_i \ln \theta_{i1} - (1 - X_i) \ln (1 - \theta_{i1}) \\ &= \sum_i X_i \ln \frac{\theta_{i0}(1 - \theta_{i1})}{\theta_{i1}(1 - \theta_{i0})} + \ln \frac{1 - \theta_{i0}}{1 - \theta_{i1}} \end{aligned}$$

Substituting this in the earlier expression, we get

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)} \quad (4)$$

where the weights are given by

$$\begin{aligned} w_0 &= \ln \frac{1 - \pi}{\pi} + \sum_i \ln \frac{1 - \theta_{i0}}{1 - \theta_{i1}} \\ w_i &= \ln \frac{\theta_{i0}(1 - \theta_{i1})}{\theta_{i1}(1 - \theta_{i0})} \end{aligned}$$

This is a logistic probability function.

## 5. NB vs LR

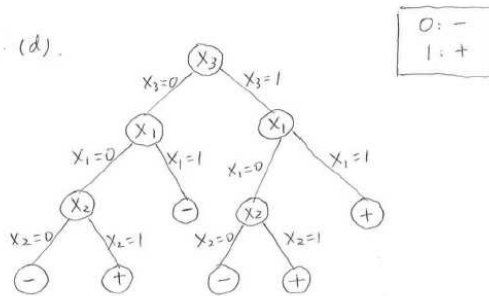
Number of test misclassifications:

- NB misclassification: 14
- Gradient ascent misclassification: 0 (learning rate= $1e-3$ )

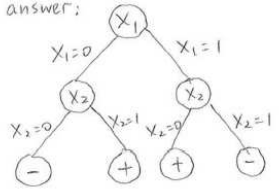
If you do not augment data with a column of 1's, then you get approximately 20 misclassifications, which could lead to the conclusion that LR here performs worse than NB. That is incorrect.

The results of the two classifications are shown in the Figures 8,9. From Figure 9, we can see that NB performs badly on the data since the features  $X_1$  and  $X_2$  are strongly correlated positively. However, NB only estimates gaussians assuming covariance between the features to be zero. The data are linearly separable, so LR can find a perfect linear separator.

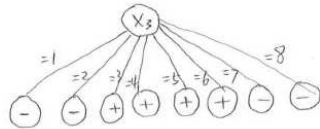
1. (d).



(e). one answer:



2. (a).



(c). one answer:

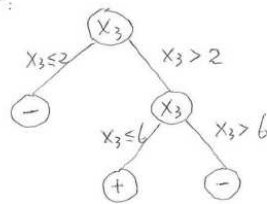


Figure 6: Decision tree

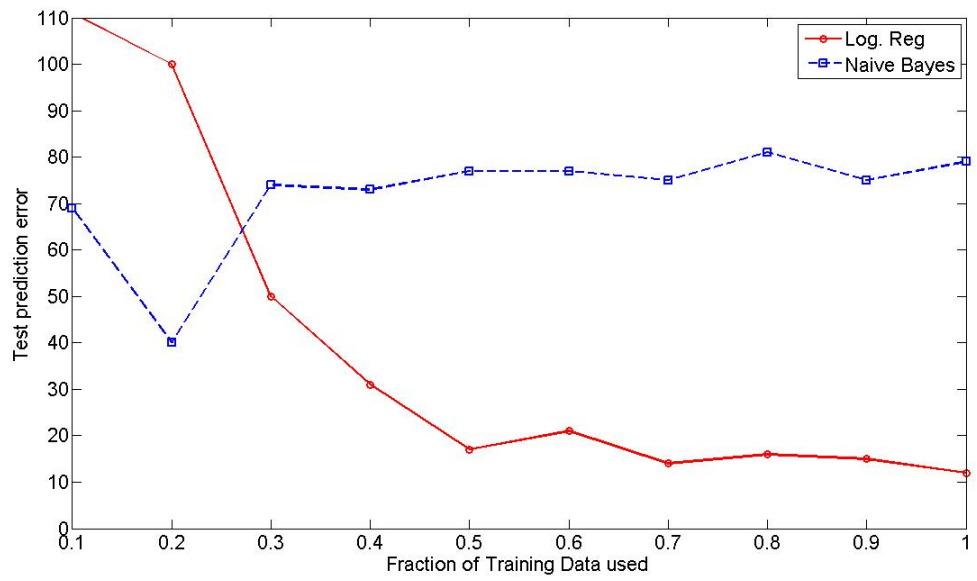


Figure 7: Variation of performance with training data

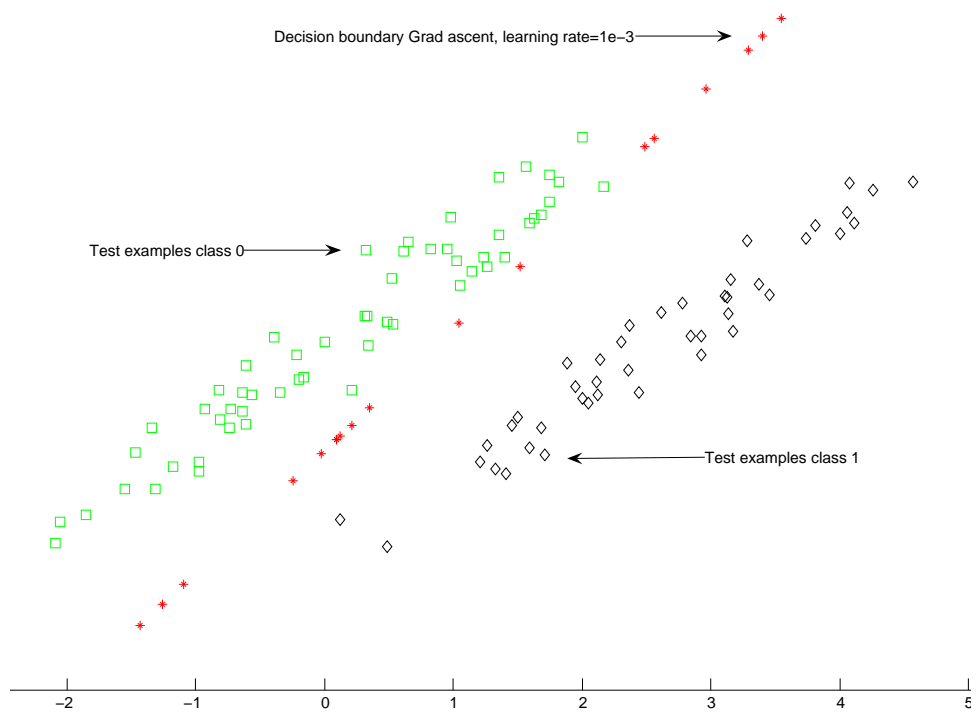


Figure 8: Decision boundary for logistic regression

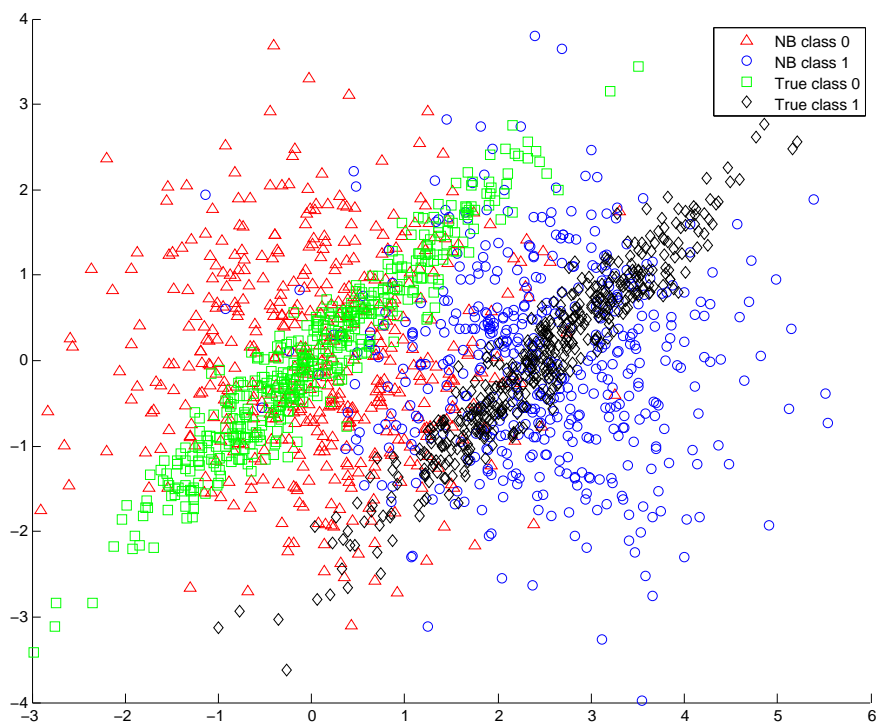


Figure 9: Estimated gaussians for NBayes and the training data