

# 10701 Machine Learning, Fall 2011

## Homework 1 solutions

October 12, 2011

### 1 Decision trees

#### 1.1 Information gain and entropy

##### 1.1.1 To show $IG(X;Y)=IG(Y;X)$

From the definition, we have

$$\begin{aligned}IG(X;Y) &= H(X) - H(X | Y) \\&= - \sum_{x=1}^k p(X = x) \log p(X = x) + \sum_{y=1}^k p(Y = y) H(X | Y = y) \\&= - \sum_{x=1}^k p(X = x) \log p(X = x) + \sum_{y=1}^k p(Y = y) \sum_{x=1}^k p(X = x | Y = y) \log p(X = x | Y = y) \\&= - \sum_{x=1}^k p(X = x) \log p(X = x) + \sum_{y=1}^k \sum_{x=1}^k p(X = x, Y = y) \log p(X = x | Y = y)\end{aligned}$$

We know that by the law of total probability (this term was not introduced in recitation, but the rule was),

$$P(X = x) = \sum_{y=1}^k P(X = x, Y = y) \tag{1}$$

Substituting this in the expression above, we get

$$\begin{aligned}IG(X;Y) &= - \sum_{x=1}^k \sum_{y=1}^k p(X = x, Y = y) \log p(X = x) + \sum_{y=1}^k \sum_{x=1}^k p(X = x, Y = y) \log p(X = x | Y = y) \\&= \sum_{x=1}^k \sum_{y=1}^k p(X = x, Y = y) (\log p(X = x | Y = y) - \log p(X = x)) \\&= \sum_{x=1}^k \sum_{y=1}^k p(X = x, Y = y) (\log p(X = x, Y = y) - \log p(Y = y) - \log p(X = x)) \\&= \sum_{x=1}^k \sum_{y=1}^k p(X = x, Y = y) \log \frac{p(X = x, Y = y)}{p(Y = y) p(X = x)}\end{aligned}$$

Therefore we have that

$$IG(X;Y) = \sum_{x=1}^k \sum_{y=1}^k p(X=x, Y=y) \log \frac{p(X=x, Y=y)}{p(Y=y)p(X=x)} \quad (2)$$

Since the expression in Equation 2 is symmetric w.r.t X and Y, interchanging them does not affect its value. Therefore  $IG(X;Y) = IG(Y;X)$ , i.e, information gain is symmetric.

**1.1.2 Show that  $IG(X;Y) = H(X) + H(Y) - H(X,Y)$**

We know by definition that  $IG(X;Y) = H(X) - H(X|Y)$ . Therefore it is sufficient to prove that  $H(X|Y) = H(X,Y) - H(Y)$  or alternatively that  $H(X|Y) + H(Y) = H(X,Y)$ . We have that

$$\begin{aligned} H(X|Y) + H(Y) &= - \sum_{y=1}^k p(Y=y) \sum_{x=1}^k p(X=x|Y=y) \log p(X=x|Y=y) + H(Y) \\ &= - \sum_{y=1}^k \sum_{x=1}^k p(X=x, Y=y) \log p(X=x|Y=y) - \sum_{y=1}^k p(Y=y) \log p(Y=y) \\ &= - \sum_{y=1}^k \sum_{x=1}^k p(X=x, Y=y) \log p(X=x|Y=y) \\ &\quad - \sum_{y=1}^k \sum_{x=1}^k p(X=x, Y=y) \log p(Y=y) \\ &= - \sum_{y=1}^k \sum_{x=1}^k p(X=x, Y=y) [\log p(X=x|Y=y) + \log p(Y=y)] \\ &= - \sum_{y=1}^k \sum_{x=1}^k p(X=x, Y=y) [\log p(X=x|Y=y) p(Y=y)] \\ &= - \sum_{y=1}^k \sum_{x=1}^k p(X=x, Y=y) \log p(X=x, Y=y) \\ &= H(X,Y) \end{aligned}$$

Therefore  $IG(X;Y) = H(X) + H(Y) - H(X,Y)$

**1.1.3 Show that  $IG(X;Y) = H(X,Y) - H(X|Y) - H(Y|X)$**

For this part, we use the definition of the joint entropy to claim that  $H(Y,X) = H(X,Y)$  by symmetry of the definition. In part 2, we showed that  $H(X|Y) + H(Y) = H(X,Y)$  or equivalently, that

$$H(Y) = H(X,Y) - H(X|Y) \quad (3)$$

Replacing Y by X and X by Y in Equation 3 and using the result that  $H(Y,X) = H(X,Y)$ , we get that

$$H(X) = H(X,Y) - H(Y|X) \quad (4)$$

Substituting the results from Equations 3 and 4 in the result from part 2, we get that

$$\begin{aligned}
 IG(X;Y) &= H(X) + H(Y) - H(X,Y) \\
 &= [H(X,Y) - H(Y|X)] + [H(X,Y) - H(X|Y)] - H(X,Y) \\
 &= H(X,Y) - H(X|Y) - H(Y|X)
 \end{aligned}$$

Therefore  $IG(X;Y) = H(X,Y) - H(X|Y) - H(Y|X)$ .

## 1.2 Building decision trees with Information gain

### 1.2.1 How to use IG to decide a split

The procedure for using IG to decide whether to split on weather or road traffic is:

- Compute the entropy of the distribution of accident rate at the root node.
- To split according to weather, partition the data based on the value of the weather attribute.
- Compute the weighted entropy of the accident rate distributions at the two child nodes, with the weight given by the number of data points in the node.
- Compute the information gain due to weather as the difference between the root entropy and the weighted children entropy.
- Repeat the process for splitting along road traffic to determine information gain due to road traffic.
- Choose the attribute which gives a larger information gain.

### 1.2.2 Calculations of the split

At the root, the distribution of accident rate is [62 High,69 Low]. For convenience, we will represent High as '+' and Low as '-' in the following calculation.

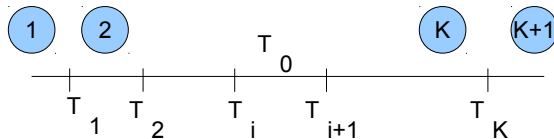
$$\begin{aligned}
 H(root) &= H(62+,69-) \\
 &= -\frac{62}{131} \log_2 \frac{62}{131} - \frac{69}{131} \log_2 \frac{69}{131} \\
 &= 0.998
 \end{aligned}$$

If we split on weather being Sunny or Rainy, the two child nodes we get have an accident rate distribution which is [30 +,53 -] for Sunny and [32 +,16 -] for Rainy. Therefore we have  $H(Sunny) = 0.944$  and  $H(Rainy) = 0.918$  So, we can compute  $IG(Weather)$  as

$$\begin{aligned}
 IG(Weather) &= H(root) - \left( \frac{83}{131} H(Sunny) + \frac{48}{131} H(Rainy) \right) \\
 &= 0.998 - 0.934 \\
 &= 0.064
 \end{aligned}$$

If we split on road traffic being Heavy or Light, the two child nodes we get have an accident rate distribution which is [37 +,27 -] for Heavy and [25 +,42 -] for Light. Therefore we have

Figure 1: Range of temperature values in sorted form. The blue circles indicate the  $K+1$  regions into the which the temperature values partition the data



$H(Heavy) = 0.982$  and  $H(Light) = 0.953$  So, we can compute  $IG(Traffic)$  as

$$\begin{aligned} IG(Traffic) &= H(root) - \left( \frac{64}{131}H(Heavy) + \frac{67}{131}H(Light) \right) \\ &= 0.998 - 0.967 \\ &= 0.031 \end{aligned}$$

The calculations show that  $IG(Weather) > IG(Traffic)$ . Therefore we should split by the Weather attribute first.

### 1.2.3 Continuous Temperature attribute

Figure 1.2.3 shows how the  $K$  sorted temperature values divide the real line and correspondingly, the data. This divides the line into  $K+1$  regions, indicated by the filled blue circles. Putting a threshold in the leftmost or the rightmost region induces a trivial partition on the data (where all data belongs to one side of the threshold. In the following part we will show that such a partition of the data is uninformative). Therefore we only need to consider thresholds belong to the  $K-1$  regions labeled  $2, 3, \dots, K$ .

Consider a typical region between 2 and  $K$ , say the region  $[T_i, T_{i+1})$ . If  $T_0$  indicates a typical threshold within the region, then it is clear that any  $T_0 \in [T_i, T_{i+1})$  induces the same partition on the training data. For convenience, we will say that the threshold of interest in the region  $[T_i, T_{i+1})$  is  $\frac{T_i + T_{i+1}}{2}$ .

If the sorted values of the temperature variable are  $T_1, T_2, \dots, T_K$ , then we only need to consider the  $K-1$  values  $\frac{T_1 + T_2}{2}, \dots, \frac{T_{K-1} + T_K}{2}$  as thresholds for the potential split.

### 1.2.4 Duplicate or one-valued attribute

A decision tree would not be affected by a duplicate or a single-valued variable.

**One-valued attribute** Suppose a split at a one-valued attribute is being considered at any stage in the tree formation. In this case, all the data will be associated with a single child node which would be identical to the root node for the split. Therefore the information gain for that split would be zero, making the one-valued attribute the least likely candidate for a split.

**Duplicate attribute** Suppose  $A_1$  and  $A_2$  are the attributes which are duplicates of each other. If neither of them are in the decision tree and are being considered for a split, we can see that their duplicate nature means that they will induce the same partition on the data after the split. As a result, their information gain will be the same. Any one of them can be chosen for the split.

If  $A_1$  (say) is already in the tree and  $A_2$  is being considered for a split, the root node of the split already contains a value (or a range of values) of  $A_1$ . If  $A_1$  takes a single value at the root, then

A2 must also take a single value and from the previous case, we know that the information gain due to A2 would be zero. If A1 takes a range of values at the root node of the split, then A2 will have non-negative information gain and could allow further improvement in the training error of the decision tree. (Note: This could have been accomplished by further splitting on A1 as well).

## 2 Linear regression

### 2.1 1-D Regression

(a)

The squared-error loss function is  $J(a, b) = \sum_i (y_i - ax_i - b)^2$ .

(b)

To find  $\arg \min_{a,b} J(a, b)$ , we differentiate  $J(a, b)$  w.r.t.  $a, b$ .

$$\begin{aligned} \frac{d}{da} J(a, b) &= \sum_i -2(y_i - ax_i - b)x_i \\ \frac{d}{db} J(a, b) &= \sum_i -2(y_i - ax_i - b). \end{aligned}$$

Setting both differentials to zero yields a system of 2 equations:

$$\begin{aligned} a &= \frac{(\sum_i y_i x_i) - (b \sum_i x_i)}{\sum_i x_i^2} \\ b &= \frac{(\sum_i y_i) - (a \sum_i x_i)}{N}. \end{aligned}$$

Substituting  $b$  into  $a$ , we obtain the minimizer in  $a$ ,

$$\begin{aligned} a^* &= \frac{(\sum_i y_i x_i) - (\sum_i x_i) \frac{(\sum_i y_i) - (a^* \sum_i x_i)}{N}}{\sum_i x_i^2} \\ a^* &= \frac{N(\sum_i y_i x_i) - (\sum_i x_i)(\sum_i y_i)}{N(\sum_i x_i^2) - (\sum_i x_i)^2} \end{aligned}$$

whereas substituting  $a$  into  $b$  gives the minimizer in  $b$ ,

$$\begin{aligned} b^* &= \frac{(\sum_i y_i) - (\sum_i x_i) \frac{(\sum_i y_i x_i) - (b^* \sum_i x_i)}{\sum_i x_i^2}}{N} \\ b^* &= \frac{(\sum_i x_i^2)(\sum_i y_i) - (\sum_i x_i)(\sum_i y_i x_i)}{N(\sum_i x_i^2) - (\sum_i x_i)^2}. \end{aligned}$$

(c)

The log-likelihood function is

$$\ell(a, b) = \sum_i \log \left( \frac{1}{\sqrt{2\pi}\sigma} \right) - \frac{(y_i - ax_i - b)^2}{2\sigma^2},$$

implying that the MLE is

$$\arg \max_{a,b} \ell(a,b) = \arg \max_{a,b} - \sum_i (y_i - ax_i - b)^2.$$

(d)

We have that

$$\arg \max_{a,b} \ell(a,b) = \arg \max_{a,b} - \sum_i (y_i - ax_i - b)^2 = \arg \min_{a,b} \sum_i (y_i - ax_i - b)^2 = \arg \min_{a,b} J(a,b).$$

In other words, the least square estimate in (a) is equal to the MLE estimate in (c).

(e)

Let

$$\mathbf{X} = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

and note that the inverse formula for  $2 \times 2$  matrices is

$$\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}^{-1} = \frac{1}{\alpha\delta - \beta\gamma} \begin{bmatrix} \delta & -\beta \\ -\gamma & \alpha \end{bmatrix}.$$

Thus, the matrix form solution for  $a, b$  is

$$\begin{aligned} \begin{bmatrix} a^* \\ b^* \end{bmatrix} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \\ &= \begin{bmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & N \end{bmatrix}^{-1} \begin{bmatrix} \sum_i x_i y_i \\ \sum_i y_i \end{bmatrix} \\ &= \frac{1}{N(\sum_i x_i^2) - (\sum_i x_i)^2} \begin{bmatrix} N & -\sum_i x_i \\ -\sum_i x_i & \sum_i x_i^2 \end{bmatrix} \begin{bmatrix} \sum_i x_i y_i \\ \sum_i y_i \end{bmatrix} \\ &= \frac{1}{N(\sum_i x_i^2) - (\sum_i x_i)^2} \begin{bmatrix} N(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i) \\ -(\sum_i x_i)(\sum_i x_i y_i) + (\sum_i x_i^2)(\sum_i y_i) \end{bmatrix} \end{aligned}$$

which is equal to the solution in (b).

## 2.2 Regularization: Ridge and Lasso Regression

(a)

We minimize  $J_R(\beta)$  by setting its gradient w.r.t.  $\beta$  to zero:

$$\begin{aligned}
\nabla J_R(\beta) &= \nabla \left[ (\mathbf{X}\beta - \mathbf{y})^\top (\mathbf{X}\beta - \mathbf{y}) + \lambda \beta^\top \beta \right] = 0 \\
2\mathbf{X}^\top (\mathbf{X}\beta - \mathbf{y}) + 2\lambda\beta &= 0 \\
(\mathbf{X}^\top \mathbf{X} + \lambda \mathbb{I}) \beta &= \mathbf{X}^\top \mathbf{y} \\
\beta^* &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbb{I})^{-1} \mathbf{X}^\top \mathbf{y}
\end{aligned}$$

(b)

Assuming  $\mathbf{X}^\top \mathbf{X} = \mathbb{I}$ , we minimize  $J_L(\beta)$  by setting its gradient w.r.t.  $\beta$  to zero:

$$\begin{aligned}
\nabla J_R(\beta) &= \nabla \left[ (\mathbf{X}\beta - \mathbf{y})^\top (\mathbf{X}\beta - \mathbf{y}) + \lambda \|\beta\|_1 \right] = 0 \\
2\mathbf{X}^\top (\mathbf{X}\beta - \mathbf{y}) + \lambda \nabla \|\beta\|_1 &= 0 \\
2\beta + \lambda \nabla \|\beta\|_1 &= 2\mathbf{X}^\top \mathbf{y}
\end{aligned}$$

Let us consider the derivative w.r.t.  $\beta_a$  for some  $1 \leq a \leq M$ :

$$2\beta_a + \lambda \frac{d}{d\beta_a} |\beta_a| = 2 \sum_i x_{i,a} y_i$$

There are 3 cases to consider: either the optimal value of  $\beta_a$  is  $> 0$ ,  $< 0$ , or  $= 0$ . If we assume the optimal  $\beta_a > 0$ , then  $\frac{d}{d\beta_a} |\beta_a| = 1$  and we have

$$\begin{aligned}
2\beta_a + \lambda &= 2 \sum_i x_{i,a} y_i \\
\beta_a &= \left( \sum_i x_{i,a} y_i \right) - \frac{\lambda}{2}
\end{aligned}$$

provided that  $(\sum_i x_{i,a} y_i) > \frac{\lambda}{2}$ . If we assume the optimal  $\beta_a < 0$ , then  $\frac{d}{d\beta_a} |\beta_a| = -1$  and we have

$$\begin{aligned}
2\beta_a - \lambda &= 2 \sum_i x_{i,a} y_i \\
\beta_a &= \left( \sum_i x_{i,a} y_i \right) + \frac{\lambda}{2}
\end{aligned}$$

provided that  $(\sum_i x_{i,a} y_i) < -\frac{\lambda}{2}$ . For the final case where the optimal  $\beta_a = 0$ , there is no closed form solution to the gradient  $\frac{d}{d\beta_a} |\beta_a|$ . We observe this corresponds to situations where  $-\frac{\lambda}{2} \leq (\sum_i x_{i,a} y_i) \leq \frac{\lambda}{2}$ . In other words,  $-\frac{\lambda}{2} \leq (\sum_i x_{i,a} y_i) \leq \frac{\lambda}{2}$  implies  $\beta_a = 0$ .

Hence the optimal  $\beta^*$  is

$$\beta_a^* = \begin{cases} \left( \sum_i x_{i,a} y_i \right) - \frac{\lambda}{2} & \text{if } (\sum_i x_{i,a} y_i) > \frac{\lambda}{2} \\ \left( \sum_i x_{i,a} y_i \right) + \frac{\lambda}{2} & \text{if } (\sum_i x_{i,a} y_i) < -\frac{\lambda}{2} \\ 0 & \text{otherwise.} \end{cases}$$

(c)

Assuming  $\mathbf{X}^\top \mathbf{X} = \mathbb{I}$ , vanilla linear regression implies  $\beta_a^* = \sum_i x_{i,a} y_i$ , ridge regression implies  $\beta_a^* = \frac{1}{\lambda} \sum_i x_{i,a} y_i$ , and Lasso regression implies

$$\beta_a^* = \begin{cases} (\sum_i x_{i,a} y_i) - \frac{\lambda}{2} & \text{if } (\sum_i x_{i,a} y_i) > \frac{\lambda}{2} \\ (\sum_i x_{i,a} y_i) + \frac{\lambda}{2} & \text{if } (\sum_i x_{i,a} y_i) < -\frac{\lambda}{2} \\ 0 & \text{otherwise.} \end{cases}$$

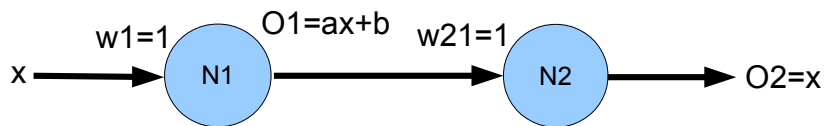
With respect to vanilla linear regression, ridge regression shrinks the  $\beta^*$  estimates by a factor of  $\frac{1}{\lambda}$ , whereas Lasso regression translates the  $\beta^*$  estimates by a distance  $\frac{\lambda}{2}$  towards zero.

### 3 Neural Networks [25pt, Bin]

#### 3.1 Representation

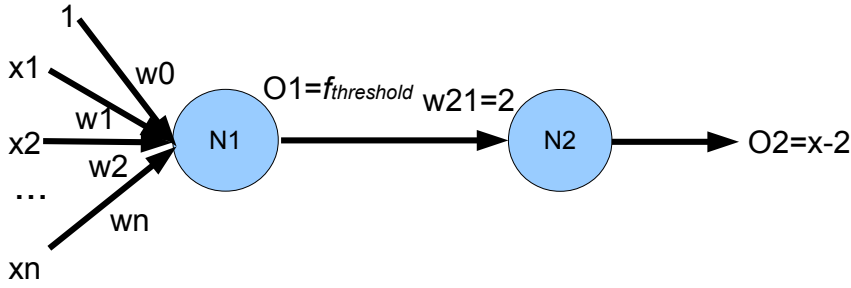
(1) Suppose the function is  $y = ax + b$ .

$$o_2(x) = 1 \times o_1(x) = 1 \times (a(x \times 1) + b) = ax + b$$



- (2) Cannot find one, since both activation functions are linear. We could not generate non-linear functions such as polynomials of degree two.
- (3) Cannot find one. 1) If the function at the first layer is the hard threshold function, the final output will only contain two values 0 and 1. 2) If both the functions at the first layer and the second layer are linear functions, the output function would still be linear. 3) If the functions at the first layer are hard threshold functions, and the function at the second layer is the linear function, the final output would still be piecewise constant. In neither of the above case, can we find any neural net that generates the hinge loss function.
- (4) Let's denote the hard threshold function as  $f_{threshold}$ .

$$\begin{aligned} o_2(x) &= (2 \times o_1(w_0 + \sum_i w_i x_i)) - 2 = (2 \times f_{threshold}(w_0 + \sum_i w_i x_i)) - 2 \\ &= 2 \times 1 - 2 = 0 \text{ if } w_0 + \sum_i w_i x_i \geq 0 \\ &= 2 \times 0 - 2 = -2 \text{ otherwise} \end{aligned}$$



(5) Suppose the function is

$$\begin{aligned}
 y &= b_0 \text{ if } x < a_0 \\
 &= b_i \text{ if } a_{i-1} \leq x < a_i
 \end{aligned}$$

for an infinite increasing sequence,  $A = a_0, a_1, \dots, a_i, \dots$ , where  $a_i < a_{i+1}$ .

If  $x < a_0$ . For all  $o_{1j}$ , where  $j \geq 0$ , we have  $a_j > x$ . Therefore,  $x - a_j < 0$ . So,  $o_{1j} = 0$ . Then,

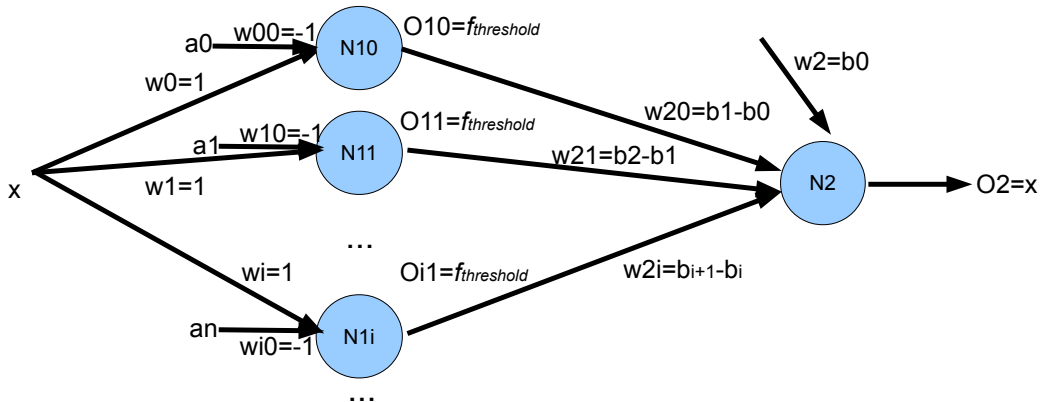
$$o_2 = w_2 + \sum_i w_{2i} \times o_{1i} = b_0 + \sum_i w_{2i} \times 0 = b_0$$

If  $a_{i-1} \leq x < a_i$ .

- For all  $o_{1j}$ , where  $j < i$ , we have  $a_j \leq x$ . Therefore,  $x - a_j \geq 0$ . So  $o_{1j} = 1$ .
- For all  $o_{1j}$ , where  $j \geq i$ , we have  $a_j > x$ . Therefore,  $x - a_j < 0$ . So  $o_{1j} = 0$ .

Then,

$$\begin{aligned}
 o_2 &= w_2 + \sum_i w_{2i} \times o_{1i} \\
 &= w_2 + \sum_{j < i} w_{2j} \times o_{1j} + \sum_{j \geq i} w_{2j} \times o_{1j} \\
 &= b_0 + \sum_{j < i} w_{2j} \times 1 + \sum_{j \geq i} w_{2j} \times 0 \\
 &= b_0 + \sum_{j < i} b_{j+1} - b_j \\
 &= b_i
 \end{aligned}$$



### 3.2 Decision Boundary

(1) 1-KNN, decision tree and Naive Bayes can learn the decision boundary, but logistic regression cannot – because they can only learn linear decision boundary.

(2)

$$o_1 = \frac{1}{1 + \exp(-(w_{10} + w_{11}x_1 + w_{12}x_2))},$$

$$o_2 = \frac{1}{1 + \exp(-(w_{20} + w_{21}x_1 + w_{22}x_2))}.$$

(3) Decide  $\text{sgn}(o_3 - 1/2)$ .

(4) There is no set of weights can learn the correct decision boundary. This is because now the two-layer (even any number of layer) NN collapse into a one-layer NN with a logistic unit. Hence, the classifier becomes a logistic regression classifier, which can only learn linear boundaries.

### 3.3 Cross-Entropy and Noisy Label

(1) We see that

$$\begin{aligned} \mathbf{W}_{\text{MLE}} &= \arg \max_{\mathbf{w}} \prod_{i=1}^N p(t^i | x^i; \mathbf{w}) \\ &= \arg \max_{\mathbf{w}} \sum_{i=1}^N \ln p(t^i | x^i; \mathbf{w}) \\ &= \arg \max_{\mathbf{w}} \sum_{i=1}^N (t^i \ln y^i + (1 - t^i) \ln(1 - y^i)) \\ &= \arg \min_{\mathbf{w}} E[W]. \end{aligned}$$

(2) The error function considering noisy label is

$$E'[W] = - \sum_{i=1}^N (t^i \ln(y^i(1 - \epsilon) + (1 - y^i)\epsilon) + (1 - t^i) \ln((1 - y^i)(1 - \epsilon) + y^i\epsilon))$$

When  $\epsilon = 0$ ,

$$\begin{aligned} E'[W] &= - \sum_{i=1}^N (t^i \ln(y^i(1 - \epsilon) + (1 - y^i)\epsilon) + (1 - t^i) \ln((1 - y^i)(1 - \epsilon) + y^i\epsilon)) \\ &= - \sum_{i=1}^N (t^i \ln(y^i \times 1 + (1 - y^i) \times 0) + (1 - t^i) \ln((1 - y^i) \times 1 + y^i \times 0)) \\ &= - \sum_{i=1}^N (t^i \ln y^i + (1 - t^i) \ln(1 - y^i)) \\ &= E[W] \end{aligned}$$

Consider the extreme case in which the labels are always incorrect. If using the original error function, back-propagation would learn a completely incorrect model. But if we use the new model, and set  $\epsilon$  to be 1, back-propagation would actually try to learn a model that classifies the data opposite to the original label, which is what we are actually looking for.

## 4 Logistic Regression and Naive Bayes [25 points, Suyash]

- Number of misclassified examples on the test set:
  - IRLS: 4
  - Gradient ascent: 13
- Decision boundary of logistic regression is  $w_0 + \sum_i X_i w_i = 0$ . The decision boundary is linear.
- Number of misclassified examples on the test set: 89. If you set both class variances to be the same, then your accuracy improves and you only get about 20 misclassified examples.
- Assume  $Y$  is Bernoulli with parameter  $\pi = P(Y = 1)$ , and  $P(X_i|Y = j)$  is Bernoulli with parameter  $\theta_{ij}$ .

Then we have that  $P(X_i|Y = j) = \theta_{ij}^{X_i} (1 - \theta_{ij})^{(1-X_i)}$

The conditional probability for Naive Bayes classifier could be written as

$$P_{NB}(Y = 1|X) = \frac{P(Y = 1) \prod_{i=1}^n P(X_i|Y = 1)}{P(Y = 0) \prod_{i=1}^n P(X_i|Y = 0) + P(Y = 1) \prod_{i=1}^n P(X_i|Y = 1)} \quad (5)$$

$$= \frac{1}{1 + \frac{P(Y=0) \prod_{i=1}^n P(X_i|Y=0)}{P(Y=1) \prod_{i=1}^n P(X_i|Y=1)}} \quad (6)$$

$$= \frac{1}{1 + \exp\left(\ln\left(\frac{P(Y=0) \prod_{i=1}^n P(X_i|Y=0)}{P(Y=1) \prod_{i=1}^n P(X_i|Y=1)}\right)\right)} \quad (7)$$

$$= \frac{1}{1 + \exp\left(\ln\left(\frac{P(Y=0)}{P(Y=1)}\right) + \sum_{i=1}^n \ln\left(\frac{P(X_i|Y=0)}{P(X_i|Y=1)}\right)\right)} \quad (8)$$

$$= \frac{1}{1 + \exp\left(\ln\left(\frac{1-\pi}{\pi}\right) + \sum_{i=1}^n X_i \ln\left(\frac{\theta_{i0}}{\theta_{i1}}\right) + (1 - X_i) \ln\left(\frac{1-\theta_{i0}}{1-\theta_{i1}}\right)\right)} \quad (9)$$

$$= \frac{1}{1 + \exp\left(\ln\left(\frac{1-\pi}{\pi}\right) + \sum_{i=1}^n \ln\left(\frac{1-\theta_{i0}}{1-\theta_{i1}}\right) + \sum_{i=1}^n X_i \ln\left(\frac{\theta_{i0}(1-\theta_{i1})}{\theta_{i1}(1-\theta_{i0})}\right)\right)} \quad (10)$$

$$(11)$$

Now, we could define

$$w_0 = \ln\left(\frac{1-\pi}{\pi}\right) + \sum_{i=1}^n \ln\left(\frac{1-\theta_{i0}}{1-\theta_{i1}}\right) \quad (12)$$

$$w_1 = \ln\left(\frac{\theta_{i0}(1-\theta_{i1})}{\theta_{i1}(1-\theta_{i0})}\right) \quad (13)$$

Then we have

$$P_{NB}(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n X_i w_i)} = P_{LR}(Y = 1|X) \quad (14)$$

---

**Note:** Some students said that since the boolean variable can be represented as 1 and 0, they form a special case of the continuous analysis done in class. However, this is not exactly true. When  $X_i$  is boolean, we must have that  $P(X_i = 0) + P(X_i = 1) = 1$  since the boolean variable can only take two values. So, if the probability of  $X_i$  were to have a gaussian form, we would need

$$\frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-(1-\mu)^2}{2\sigma^2} + \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-(0-\mu)^2}{2\sigma^2} = 1$$

It is not obvious that there exist  $\mu$  and  $\sigma$  which satisfy this constraint and the constraint that the two classes must have the same variance (as we assumed in class).

---

5. For logistic regression, the conditional probability  $P(Y = 1|X)$  is shown in figure 2(a), and the decision boundary is shown in figure 2(b). The number of misclassified examples is 0. For naive Bayes, the conditional probability  $P(Y = 1|X)$  is shown in figure 3(a), and

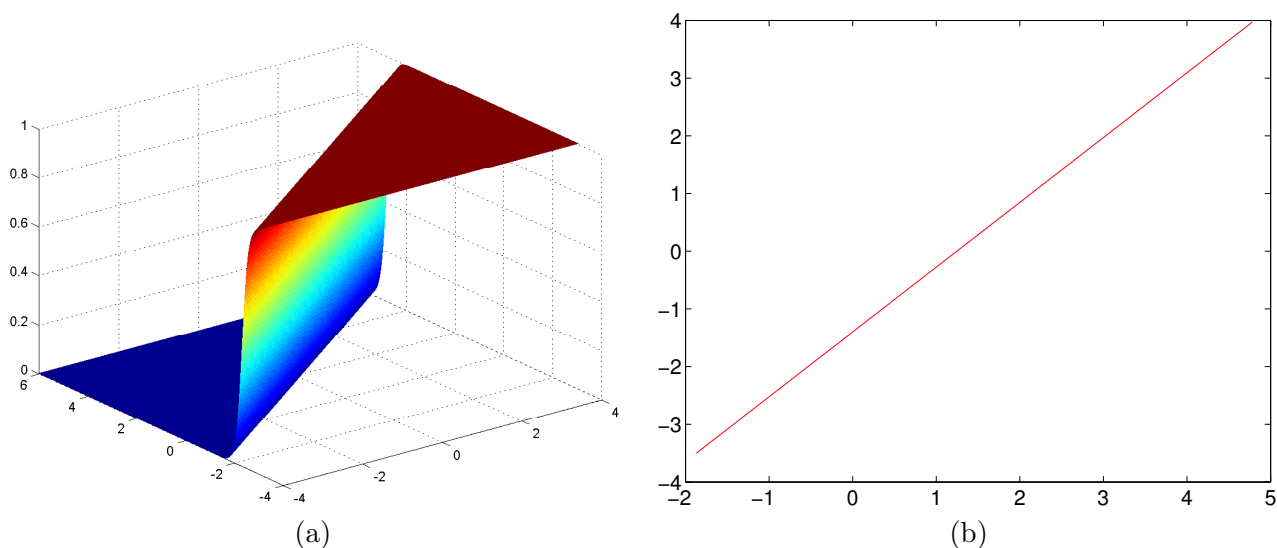


Figure 2: (a) Conditional probability of logistic regression (b) decision boundary of logistic regression.

the decision boundary is shown in figure 3(b). The number of misclassified examples is 16. Clearly, logistic regression could separate the test data correctly, while naive Bayes cannot. The decision boundary of logistic regression is linear, while the decision boundary for naive Bayes in this question is quadratic, i.e., non-linear. The plot of test data is shown in figure 4, where we clearly see that the 2 classes of data are linearly separable. The Naive Bayes assumption that the features are independent (given the class label) are violated by this data, where the features are strongly correlated.

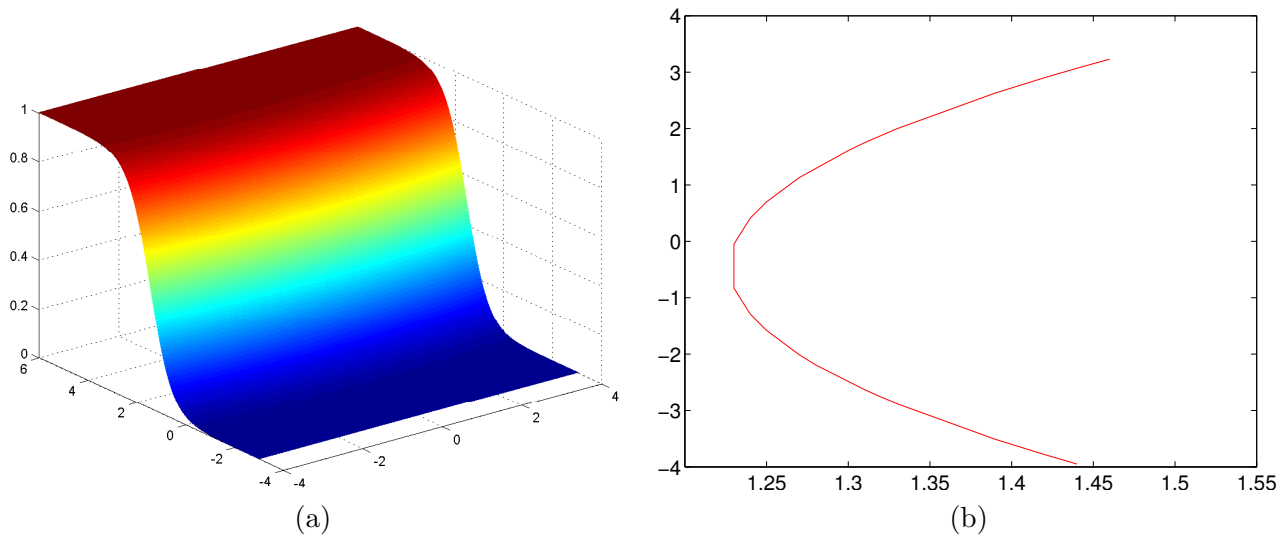


Figure 3: (a) Conditional probability of naive Bayes (b) decision boundary of naive Bayes.

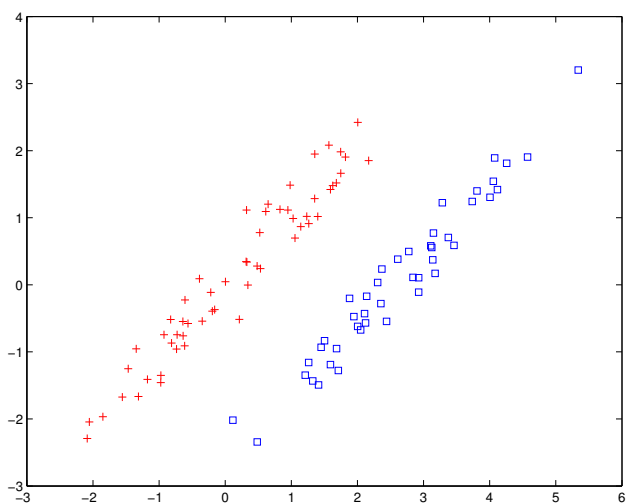


Figure 4: Test data distribution in q4\_5.mat.