CSE 546 Machine Learning, Autumn 2013
Homework 2

Shrainik Jain - 1323338

1 Boosting

MACHINE LEARNING

HOMEWORK - 2

SHRAINIK JAIN - 132 333 B

1) $H(x) = sgn\left\{\sum_{t=1}^{T} \alpha_t h_t(x)\right\} = sgn\{f(x)\}$

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

$\epsilon_{Training}$ = Probability of wrong prediction over the distribution of weighted data points

$= P_{i \sim D_t}[H_t(x_i) \neq y_i]$

= Number of training point where we predicted incorrectly / Total number of training points

$$\epsilon_{Train} = \frac{\sum_{j=1}^{N} \mathbb{1}\{H_t(x^j) \neq y^j\}}{N} \rightarrow (1)$$

← i.e. 1, when $H(x^j) \neq y^j$; 0, otherwise.

we make an error when

$y^j = 1$ and $f(x^j) \leq 0$ { sgn is -1 }

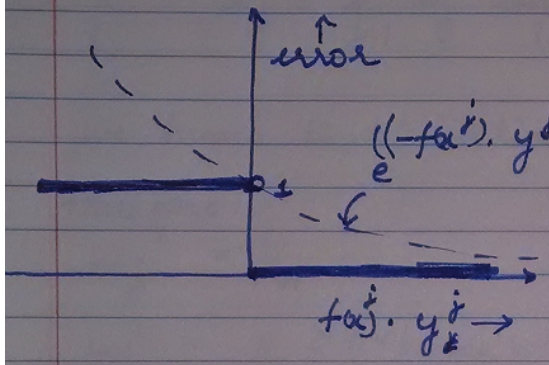
or

$y^j = -1$ and $f(x^j) \geq 0$ { sgn is +1 }

This implies that we make an error when

$$f(x^j) \cdot y^j \leq 0$$

i.e. error for a single point $= \begin{cases} 1, & f(x^j).y^j < 0 \\ 0, & \text{otherwise} \end{cases}$



in other words

$$\epsilon_{Train} = \frac{1}{N}\sum_{j=1}^{N} (\text{error for point } j)$$

$$= \frac{1}{N}\sum_{j=1}^{N} \mathbb{1}(f(x^j).y^j) < 0) \quad \hookrightarrow (2)$$

The above step function for the error of classifying j^{th} point is bounded by $\exp(-f(x^j).y^j)$ because:

$$\exp(-f(x^j).y^j) > 1 \quad \text{for } -f(x^j).y^j < 0$$

$$1 \geq \exp(-f(x^j).y^j) \geq 0 \quad \text{for } -f(x^j).y^j \geq 0$$

i.e.

$$\exp(-f(x^j).y^j) \geq f(x^j).y^j \quad \text{for all } (x^j, y^j) \quad \hookrightarrow (3)$$

combining equations (1), (2) & (3) we get:

$$\epsilon_{Training} = \frac{1}{N}\sum_{j=1}^{N} \mathbb{1}\{H(x^j) \neq y^j\} \leq \frac{1}{N}\sum_{j=1}^{N} \exp(-f(x^j).y^j)$$

Q1. 2) $$w_j^{(t+1)} = \frac{w_j^{(t)} \exp(-\alpha_t y^i h_t(x^i))}{Z_t}$$

$$Z_t = \sum_{j=1}^{N} w_j^{(t)} \exp(-\alpha_t y^i h_t(x^i))$$

we know that initial $w_{j=1 \text{ to } N}^{1} = 1/N$ } equal weights

therefore:

$$w_j^1 = 1/N$$

$$w_j^2 = 1/N \left(\frac{\exp(-\alpha_1 y^i h_1(x^i))}{Z_1} \right)$$

$$\vdots$$

$$w_j^{(t+1)} = 1/N \left(\frac{\exp(\alpha_1 y^i h_1(x^i))}{Z_1} \right) \left(\frac{\exp(-\alpha_2 y^i h_2(x^i))}{Z_2} \right) \cdots \left(\frac{\exp(-\alpha_t y^i h_t(x^i))}{Z_t} \right)$$

$$w_j^{(t+1)} = 1/N \; \frac{\exp\left(-\sum_{i=1}^{t} \alpha_i y^i h_i(x^i)\right)}{\prod_{i=1}^{t} Z_i}$$

now Sum of all weights at each iteration is 1.

So, $\sum_{j=1}^{N} w_j^{t+1} = 1$

$$\therefore \frac{1}{N}\sum_{j=1}^{N}\left(\frac{\prod_{t=1}^{T}\exp(-\alpha_t y^j h_t(x^j))}{\prod_{t=1}^{T} Z_t}\right) = 1$$

$$\frac{1}{N}\sum_{j=1}^{N}\exp\left(\sum_{t=1}^{T}(-\alpha_t y^j h_t(x^j))\right) = \prod_{t=1}^{T} Z_t$$

$$\frac{1}{N}\sum_{j=1}^{N}\exp\left(-y^j\left(\sum_{t=1}^{T}\alpha_t h_t(x^j)\right)\right) = \prod_{t=1}^{T} Z_t$$

· also $\sum_{t=1}^{T}\alpha_t h_t(x^j) = f(x_j)$

$$\therefore \frac{1}{N}\sum_{j=1}^{N}\exp(-y^j \cdot f(x^j)) = \prod_{t=1}^{T} Z_t \cdot$$

Q.E.D.

Q1) 3) (a) $\epsilon_t = \sum_{j=1}^{m} w_j^t \, \mathbb{1}\{h_t(x^j) \neq y^j\}$

$$z_t = (1-\epsilon_t)\exp(-\alpha_t) + \epsilon_t \exp(\alpha_t)$$

α_t for z_t max $\Rightarrow \frac{\partial z_t}{\partial \alpha_t} = 0$

$$\Rightarrow -(1-\epsilon_t)\exp(-\alpha_t) + \epsilon_t \exp(\alpha_t) = 0$$

$$\Rightarrow (1-\epsilon_t)\exp(-\alpha_t) = \epsilon_t \exp(\alpha_t)$$

$$\Rightarrow \ln(1-\epsilon_t) - \alpha_t = \ln(\epsilon_t) + \alpha_t$$

$$\Rightarrow \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) = 2\alpha_t$$

$$\boxed{\hat{\alpha}_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)}$$

$$z_t^{opt} = (1-\epsilon_t)\exp\left(-\frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\right) + \epsilon_t \exp\left(\frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\right)$$

$$\Rightarrow (1-\epsilon_t)\cdot\left(\frac{1-\epsilon_t}{\epsilon_t}\right)^{-1/2} + \epsilon_t\left(\frac{1-\epsilon_t}{\epsilon_t}\right)^{1/2}$$

$$\Rightarrow (1-\epsilon_t)^{1/2}\,\epsilon_t^{1/2} + \epsilon_t^{1/2}(1-\epsilon_t)^{1/2}$$

$$\Rightarrow \boxed{Z_t^{opt} = 2\sqrt{\epsilon_t(1-\epsilon_t)}} \quad \text{Q.E.D.}$$

Q1 3) (b) $\epsilon_t = \frac{1}{2} - \gamma_t$

$$\Rightarrow z_t^{opt} = 2\sqrt{(\tfrac{1}{2}-\gamma_t)(1-(\tfrac{1}{2}-\gamma_t))}$$

$$\Rightarrow \frac{2}{2}\sqrt{(1-2\gamma_t)(2\gamma_t-1)}$$

$$z_t^{opt} \Rightarrow \sqrt{1-4\gamma_t^2}$$

also, $z_t^{opt} = (1-\epsilon_t)\exp(-\gamma_t^2)$

$$z_t \leq z_t^{opt} = e^{\frac{1}{2}\ln(1-4\gamma_t^2)}$$

also, since $\log(1-x) \leq -x$ for $0 \leq x \leq 1$

$$\ln(1-4\gamma_t^2) \leq -4\gamma_t^2$$

$\Uparrow$ because $0 \leq \gamma_t \leq \frac{1}{2}$

$$\gamma_t^2 \leq \frac{1}{4}$$

$$0 \leq 4\gamma_t^2 < 1$$

therefore $\Rightarrow$ $z_t^{opt} \leq e^{\frac{1}{2}(-4\gamma_t^2)} = e^{-2\gamma_t^2}$

$\therefore z_t^{opt} \leq \exp(-2\gamma_t^2)$

$\therefore$ $z_t \leq \exp(-2\gamma_t^2)$ Q.E.D.

$$\therefore \epsilon_{Training} \leq \prod_{t=1}^{T} Z_t \leq \exp\left(-2 \sum_{t=1}^{T} \gamma_t^2\right)$$

Q 1. 3) c) if each classifier is better than random,

then $\epsilon_T < 1/2$

$$\epsilon_T = 1/2 - \gamma_t$$

for some $\gamma_t < 1/2$

this implies $\exists \gamma = \min(\gamma_1, \gamma_2 \ldots \gamma_t)$

$$\therefore \epsilon_{Training} \leq \exp\left(-2 \sum_{t=1}^{T} \gamma_t^2\right)$$

since $\forall \gamma_t \quad \gamma \leq \gamma_t$

$$\exp(-2\gamma_t^2) \leq \exp(-2\gamma^2)$$

$$\therefore \epsilon_{Train} \leq \exp(-2\gamma^2) \cdot \exp(-2\gamma^2) \ldots \ldots \exp(-2\gamma^2)$$

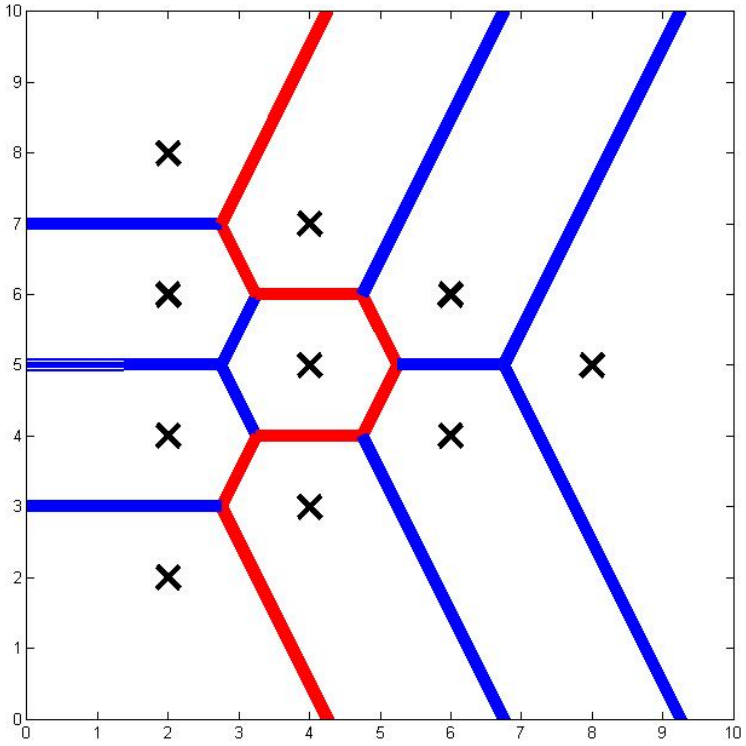
T times

$$\epsilon_{Train} \leq \exp(-2T\gamma^2)$$

Q. E. D.

2 KNN

The following figure is the Voronoi diagram for the dataset. The Redline is the decision boundary, i.e. all points left of it are positive.



Q2 1) Attached as image.

2) (8,5)
All others change the decision boundary if removed.

3). Error by removing points one at a time and classifying it based on other data points.

Let the points be numbered 1 to 10 as

positive ⇒ 1 ⇒ (2,2) | 2 ⇒ (2,4) | 3 ⇒ (2,6) | 4 ⇒ (2,8) | 5 ⇒ (4,5)
negative ⇒ 6 ⇒ (4,3) | 7 ⇒ (4,7) | 8 ⇒ (6,4) | 9 ⇒ (6,6) | 10 ⇒ (8,5)

removing point 1 ⇒. ← negative
closest points ⇒ (2, 5, 6.) ⇒ prediction
↑ positive ↑ positive
positive
Error = 0

removing point 2 ⇒ distance = 2 distance 2.2360
closest points ⇒ (1, 3, 5, 6)
↑ positive ↑ negative ↑ } irrespective of choice of the 3 points
prediction
prediction = positive
Error = 0

removing point 3 ⇒ ← distance = 2 → distance = 2.236
closest points (2, 4, 5, 7)
positive ← negative
prediction = positive
Error = 0

removing point 4 ⇒ negative

closest points (3, 5, 7)

positive prediction = positive

error = 0

removing point 5 ⇒

negative positive

closest points = (6, 7, 2, 3, 8, 9)

distance = 2 distance 2.236

error = 1

removing point 6 ⇒ distance = 2 distance = 2.236

negative

closest points = (5, 1, 2, 8)

positive

prediction = positive

error = 1 } as actual value is negative

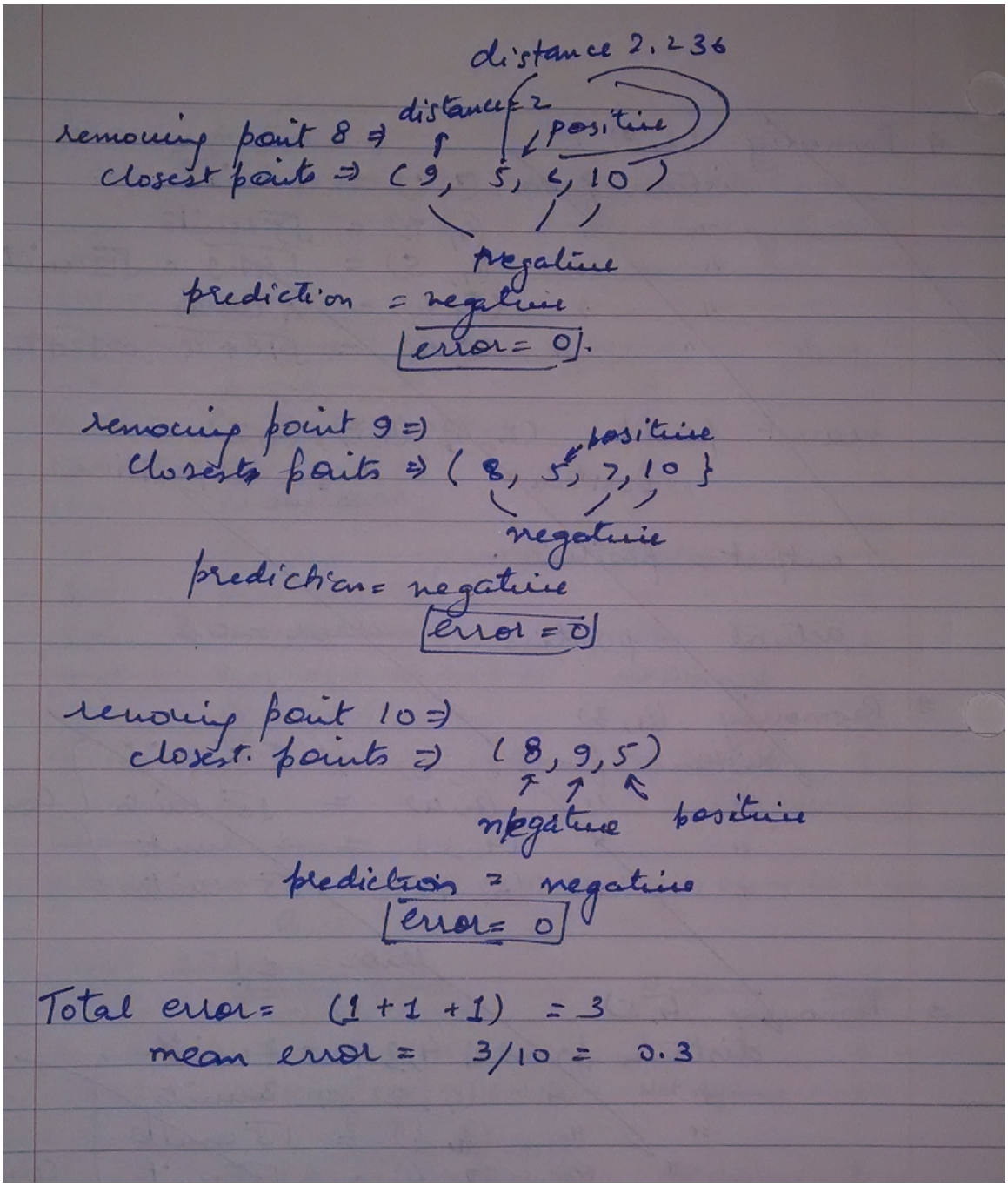
removing point 7 ⇒ distance 2 distance 2.236

closest points ⇒ (5, 3, 4, 9)

positive negative

prediction = positive

error = 1 Actual value = negative


distance 2.236
removing point 8 ⇒
distance=2
positive
closest points ⇒ (9, 5, 7, 10)
negative
prediction = negative
error = 0.
removing point 9 ⇒
positive
closest points ⇒ (8, 5, 7, 10)
negative
prediction = negative
error = 0
removing point 10 ⇒
closest points ⇒ (8, 9, 5)
negative
positive
prediction = negative
error = 0
Total error = (1 + 1 + 1) = 3
mean error = 3/10 = 0.3

Q2. a) Removing feature 1:-

Data points ⇒ (2, +) (4, +) (6, +) (8, +) (5, +)
(3, −) (7, −) (4, −) (6, −) (5, −)

LOOCV iterations:

Point to remove	Closest Point	Output	Error.
(2, +)	(3, −); (4, −); (4, +)	−	1
(4, +)	(4, −); (3, −); (5, −); (5, +)	−	1
(6, +)	(6, −); (5, +); (5, −)	−	1
(8, +)	(7, −); (6, −); (6, +)	−	1
(5, +)	(5, −); (4, −); (6, −); (4, +); (6, +)	Not Defined	1
(3, −)	(2, +); (4, +); (4, −)	+	1
(7, −)	(6, +); (6, −); (8, +)	+	1
(4, −)	(4, +); (3, −); (5, −); (5, +)	Not Defined	1
(6, −)	(6, +); (5, +); (5, −)	+	1
(5, −)	(5, +); (4, −); (4, +); (6, −); (6, +)	Not Defined	1

Total error = 10

Removing feature 2:

Data points ⇒ (2, +) (2, +) (2, +) (2, +) (4, +) (4, −) (4, −) (6, −) (6, −) (8, −)

LOOCV Iterations.

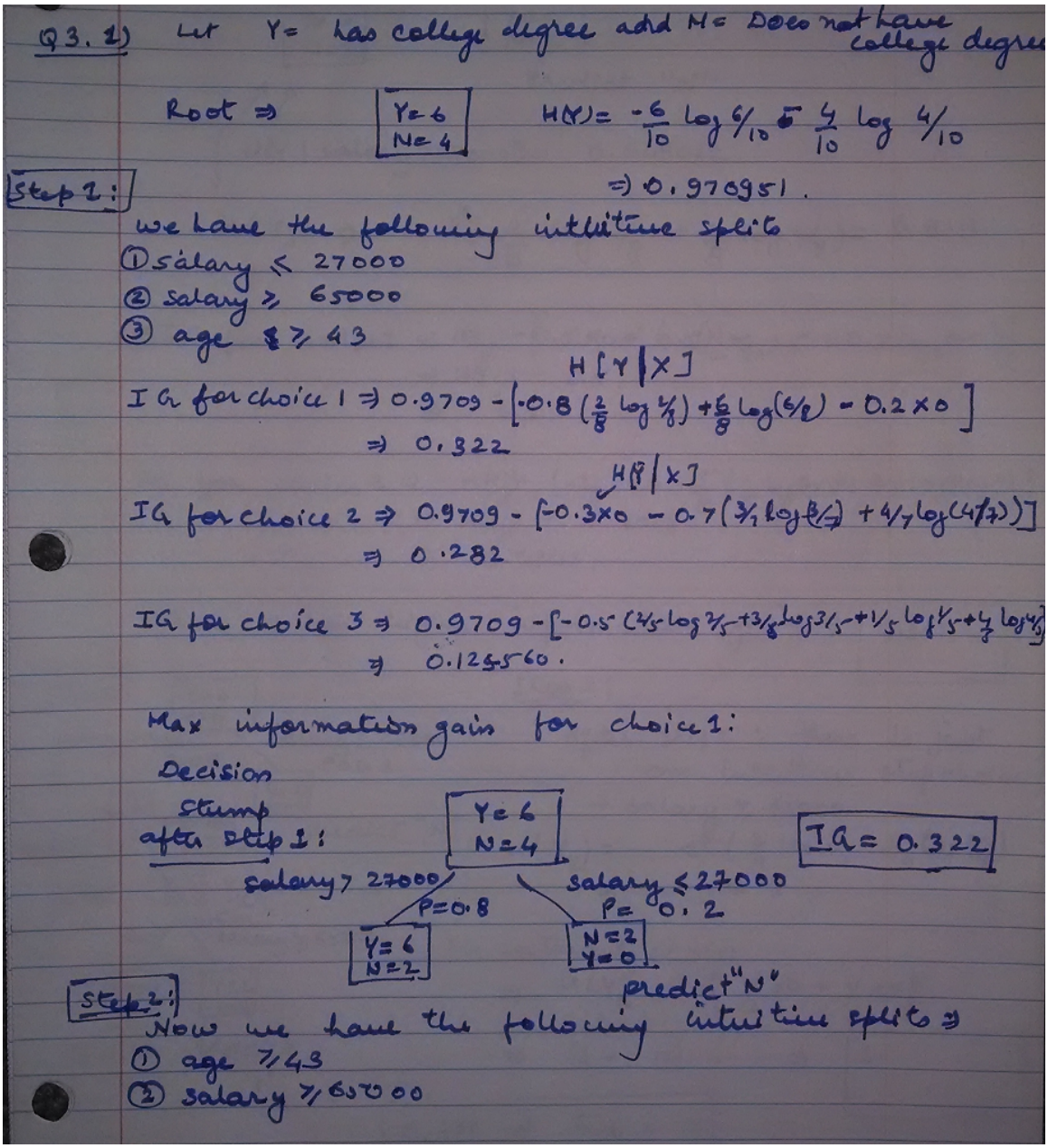
Point to remove	Closest points	Output	Error
(2, +)	(2, +); (2, +); (2, +)	+	0
same for other points at (2, +)		+	0
(4, +)	(4, −); (4, −); [(2, +)] ×4; (6, −); (6, −) (4 points with (2, +))	Not defined	1
(4, −)	(4, +); (4, +); [(2, +)] ×4; (6, −); (6, −)	Not defined	1
(4, −)	← Same →	Not defined	1
(6, −)	(6, −); (4, −); (4, −); (4, +); (8, −);	−	0
(6, −)	← same →	−	0
(8, −)	(6, −); (6, −); (4, −); (4, −); (4, +)	−	0

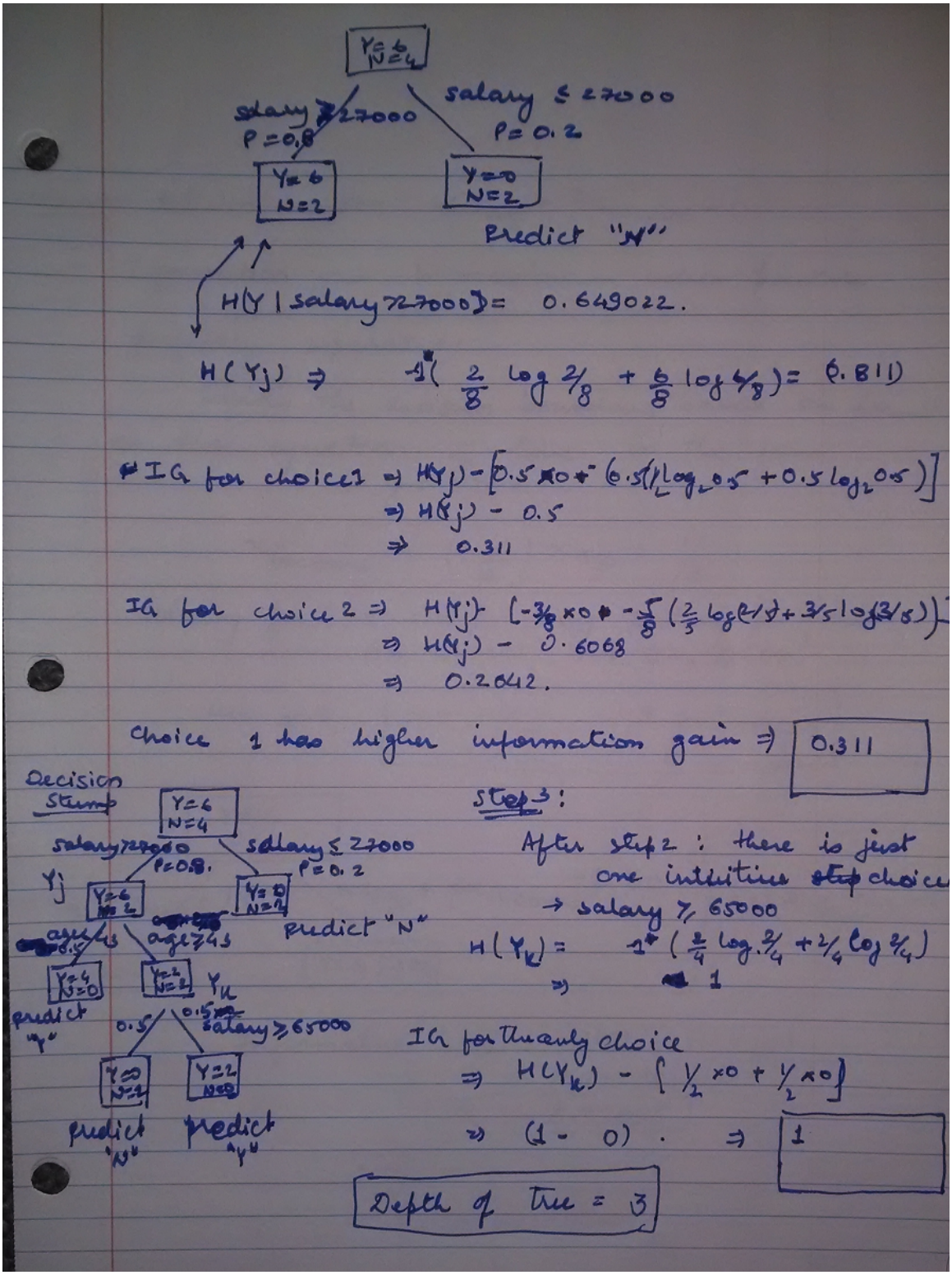
Total ⇒ 3

THEREFORE, we can safely eliminate feature 2.

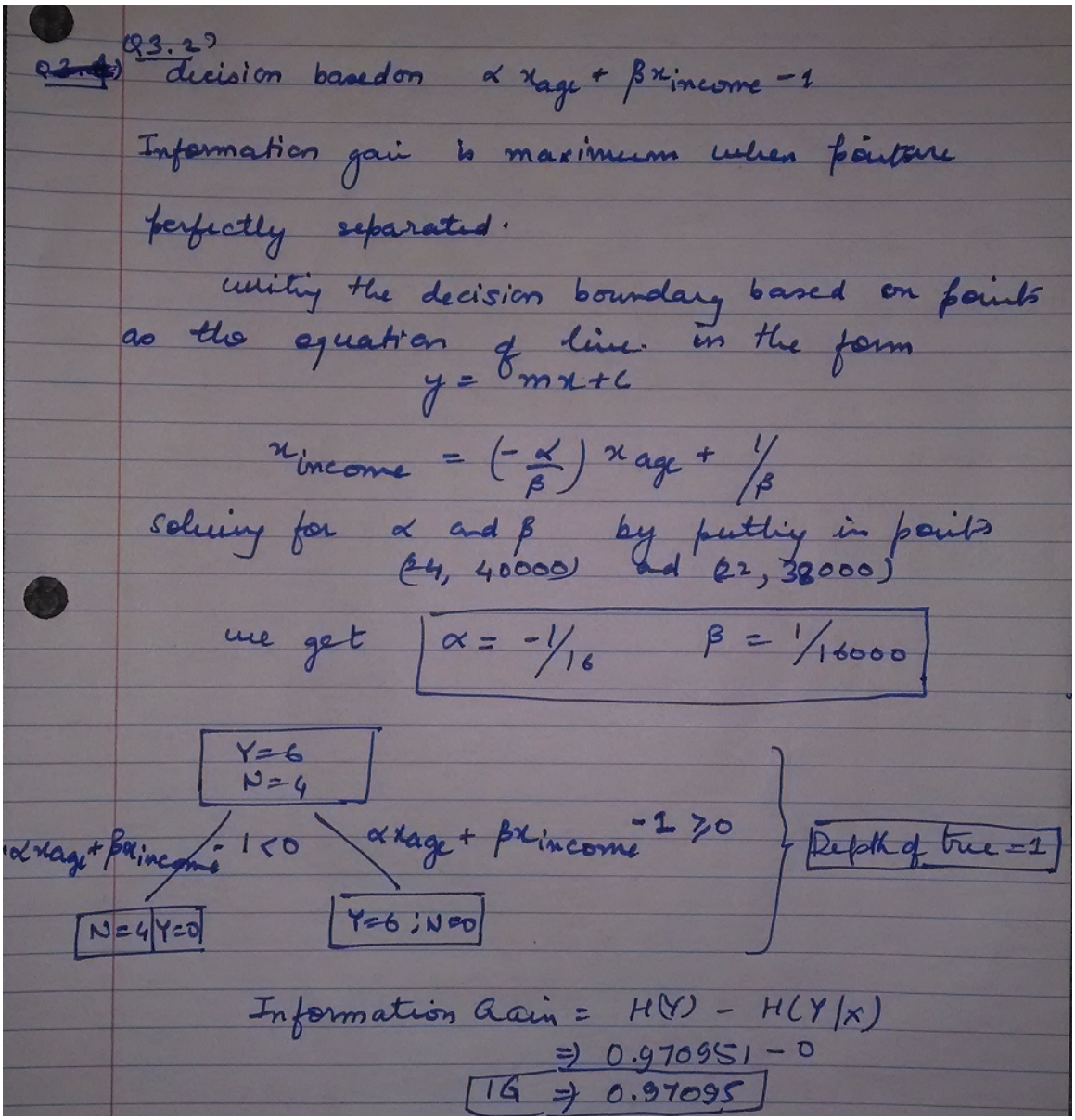
3 Decision Trees

3.1




Y=6 N=4
salary > 27000
P=0.8
salary ≤ 27000
P=0.2
Y=6 N=2
Y=0 N=2
Predict "N"
H(Y | salary > 27000) = 0.649022.
H(Yj) ⇒ -1*(2/8 log 2/8 + 6/8 log 6/8) = (0.811)
* IG for choice1 ⇒ H(Yj) - [0.5 x 0 + 0.5(1/2 log2 0.5 + 0.5 log2 0.5)]
⇒ H(Yj) - 0.5
⇒ 0.311
IG for choice 2 ⇒ H(Yj) - [3/8 x 0 + 5/8 (2/5 log(2/5) + 3/5 log(3/5))]
⇒ H(Yj) - 0.6068
⇒ 0.2042.
choice 1 has higher information gain ⇒ 0.311
Decision Stump
Y=6 N=4
salary > 27000
P=0.8
salary ≤ 27000
P=0.2
Yj
Y=6 N=2
Y=0 N=4
Predict "N"
age ≤ 45
age > 45
Y=4 N=0
Y=2 N=2
YL
predict "Y"
0.5
0.5
salary ≥ 65000
Y=0 N=2
Y=2 N=0
predict "N"
predict "Y"
Step 3 :
After step 2 : there is just one intuitive choice
→ salary ≥ 65000
H(YL) = -1* (2/4 log 2/4 + 2/4 log 2/4)
⇒ 1
IG for the only choice
⇒ H(YL) - [1/2 x 0 + 1/2 x 0]
⇒ (1 - 0) . ⇒ 1
Depth of tree = 3

3.2



3.3

Advantages of multivariate decision trees:

- More than one feature can be tested for per decision, this helps getting a way better predictor when the data points are linearly separable.
- Training error is lower as more complex models are allowed (decision based on multiple

variables and rather than a single variable).

- Multivariate decision trees will have, on an average, smaller tree size.

Disadvantages of multivariate decision trees:

- The computation cost (CPU Time) of the learning algorithm is very high (imagine running linear regression again and again for each decision node), as a lot of features need to be tested to find out the maximum information gain per step. While the univariate algorithm requires considering only one feature at a time.
- On smaller datasets, multivariate trees tend to over-fit a lot because of standard deviation in the data points. This leads to higher test errors. Multivariate trees require a lot of data-points which might or might not be present.

4 Programming Question

4.1 Unprocessed Dataset

No question in this part.

4.2 Processed Dataset (that you will use...)

No question in this part.

4.3 Accessing and Processing the Data

No question in this part.

4.4 Batch Gradient Descent

1. Actual weight update step:

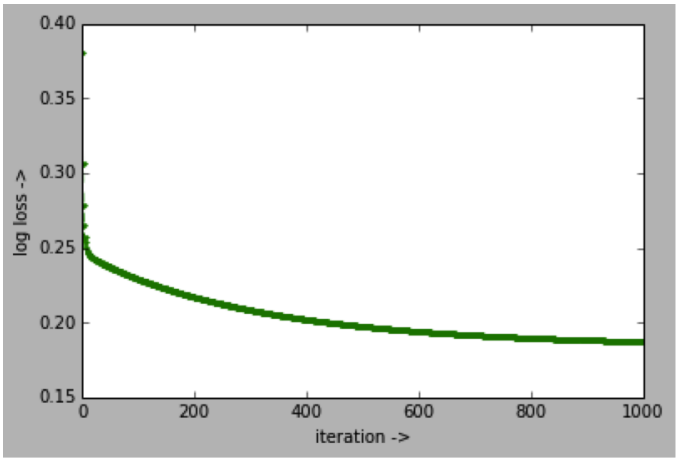
$$w_i^{t+1} = w_i^t + \eta(-\lambda w_i^t + \sum_{j=1}^{N} x_i^j (y^j - \frac{e^{w_0 + \sum w_i x_i^j}}{1 + e^{w_0 + \sum w_i x_i^j}})) \tag{1}$$

   Weight update step in python:

```
#(Y is Nx1, X is Nx(D+1), W is (D+1)x1)

yj_minus_p = Y - (exp(X.dot(W))/(1+ exp(X.dot(W))))
W[0] = W[0] + (eta / N) * sum(yj_minus_p)
W[1:] = (1 - eta * lmbd) * W[1:] + eta * (array(yj_minus_p).dot(X[:,1:]) / N)
```

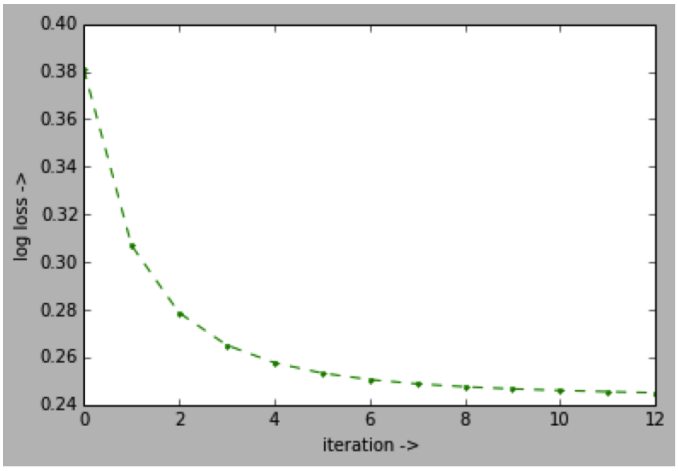
2. (a) Log loss versus number of iterations:



(b) SSE for batch gradient descent with 1000 iterations: 54.0

3. (a) Number of iterations with the stopping criteria: 13

(b) Log loss versus Iterations for gradient descent with the stop criteria:



(c) SSE for batch gradient descent with the stop criteria: 54.0

4.5 Stochastic Gradient Descent

1. Actual weight update step:

$$w_i^{t+1} = w_i^t + \eta(-\lambda w_i^t + x_i^j(y^j - \frac{e^{w_0+\sum w_i x_i^j}}{1+e^{w_0+\sum w_i x_i^j}})) \tag{2}$$

Weight update step in python:

```
#(Y is Nx1, X is Nx(D+1), W is (D+1)x1)

yj_minus_p = Y[j] - (1-1/(1+exp(X[j,:].dot(W))))
```

```
W[0] = W[0] + (eta) * yj_minus_p
W[1:] = (1 - eta * lmbd) * W[1:] + eta * (yj_minus_p * (X[j,1:]))
```

2. (a) l_2 norm for:
   - $\lambda = 0$: 1.92503456434
   - $\lambda = 0.3$: 0.283520413611

   (b) SSE for $\lambda = 0.3$: 54.0

   (c) Feature weights for:
   - $INTERCEPT$: -3.10616785425
   - $DEPTH$: 0.109353101677
   - $POSITION$: -0.006094751226

3. After 5 iterations, log loss with:
   - Stochastic Descent: 0.197392978738
   - Gradient Descent: 0.257743788383

   Stochastic Descent seems to converge faster.

4.6 Class Imbalance

1. For predictions made by SGD running with one pass over data, and $\lambda = 0.3$ and $\eta = 0.1$:
   - Precision and Recall for class 0: 0.946, 1.0
   - Precision and Recall for class 1: 0.0, 0.0

2. For predictions made by batch gradient descent running for 10000 iterations over the oversampled data and with $\lambda = 0.3$ and $\eta = 0.01$:
   - Precision and Recall for class 0: 0.94140625, 0.509513742
   - Precision and Recall for class 1: 0.04918032, 0.444444444