Ethan Wang
305319001

CS148 HW3

1 A) Entropy = $-[p(0) \times \log p(0) + p(1) \times \log p(1)]$

$p(0) = 3/8 \quad p(1) = 5/8$

Entropy = $-[(3/8 \times \log_2(3/8)) + (5/8 \times \log_2(5/8))]$

Entropy = 0.954

B) CHEST PAIN Information Gain

YES - E = 0   NO - E = 0.811   Gain = 0.954 - 0.4055

$(\frac{1}{2} \times 0) + (\frac{1}{2} \times 0.811) = 0.4055$ = 0.5485

MALE Information Gain

YES - Entropy = 1   NO - E = 0.811   Gain = 0.954 - 0.9055

$(\frac{1}{2} \times 1) + (\frac{1}{2} \times 0.811) = 0.9055$ = 0.0485

SMOKES Information Gain

YES - Entropy = 0.722   NO - E = 0.918   Gain = 0.954 - 0.7955

$(5/8 \times 0.722) + (3/8 \times 0.918) = 0.7955$ = 0.1585

EXERCISES Information Gain

YES - E = 0.971   NO - E = 0   Gain = 0.954 - 0.6069

$(5/8 \times 0.971) + (3/8 \times 0) = 0.6069$ = 0.3471

I would split on CHEST PAIN, since it has highest information gain. (0.5485)

C) ① Stop splitting into more regions if all instances in a region belong in the same class

② Stop splitting if all of the information gains are negative/less than a certain threshold

③ Stop splitting if the number of instances in a region falls below the pre-defined threshold

④ Stop splitting if the total number of regions exceeds a pre-defined threshold

⑤ Stop splitting if the points in a region are independent of the predictors

D) Standardizing or normalizing shouldn't be necessary for decision trees as such transformations will not affect how a decision tree is created. We look at entropy/information gain, neither of which are affected by standardization/normalization.

E) Decision trees are generally robust to outliers. We split based on observed entropy or Gini Index. Neither of these measurements should really be affected by outliers, meaning that the decision trees can still work fine with a few outliers.

2

$w = 0$

2 a) x_1 $w = y_1 x_1$

x_2 $w = y_1 x_1$

x_3 $w = y_1 x_1 + y_3 x_3$

x_4 $w = y_1 x_1 + y_3 x_3 + y_4 x_4$

x_5 $w = y_1 x_1 + y_3 x_3 + y_4 x_4$

After the training epoch, $w = y_1 x_1 + y_3 x_3 + y_4 x_4$

b) Based on my answer from part a, I got that $w = y_1 x_1 + y_3 x_3 + y_4 x_4$

$w = \langle 1, 1, 0 \rangle - \langle 1, 1, -3 \rangle + \langle 1, 3, -1 \rangle$ with the values from the table

$w = \langle 1, 3, 2 \rangle$

Does $y_i = \text{sgn}(w^T x_i)$? If yes, then the prediction is correct (+1)

If no, then the prediction is wrong (-1)

$(1\,1\,0)\begin{pmatrix}1\\3\\2\end{pmatrix} = 4$ $y_1 = \text{sgn}(4)$

$+1 = 1$ ✓

Model predicts $x_1 = 1$, which is correct.

c) The activation function for this Perceptron model is called sgn, whereas in class the professor said that traditionally a step function is used (for the logistic regression model).
If the perceptron in this problem used the sigmoid activation function, it would basically be the same as the model discussed during class.

3. a) For the hidden layers, we could use sigmoid, tanh, ReLU (and its variations), Softplus, or Swish. Sigmoid outputs a value between 0 and 1, which makes sense for binary classification. ReLU can regulate the input by eliminating negative inputs and acting as a linear function for non-negative inputs. The variations of ReLU have the same basic idea but have minor advantages (ie Exponential ReLU is differentiable at all points. tanh is a compressed version of sigmoid, so it follows the same idea but has outputs between -1 and 1 instead and can be a bit faster. This would still work fine for this binary classification. Softplus and Swish are similar to ReLU so they would also work fine, but they are typically used for large neural networks. All of the activation functions listed above could work in the hidden layer(s) of a binary classification neural network; it really depends on the context of the problem. Personally, I would try using ReLU or sigmoid first, then possibly experiment with other functions.

For the output layer, we want outputs to be binary for the sake of binary classification. Activation functions such as sigmoid or logistic would probably work best since they output from 0 to 1. A modified version of tanh that outputs from 0 to 1 may be used as well. Personally, I would try using sigmoid for a binary classification problem and then experiment with other activation functions if the results are unsatisfactory.

b) $X_1 = 2 \quad X_2 = -3 \quad y = 1$

$W_{11} = 0.9 \quad W_{12} = 0.4 \quad W_{21} = -1.5 \quad W_{22} = -0.7 \quad W_{31} = -0.2 \quad W_{32} = 1.6$

$W_{10} = 0 \quad W_{20} = 0 \quad W_{30} = 0$

Neuron 1: $W_1 = (X_1 \cdot W_{11}) + (X_2 \cdot W_{12})$

$= (2 \cdot 0.9) + (-3 \cdot 0.4) = 0.6 \to \text{ReLU} \to 0.6$

Neuron 2: $W_2 = (X_1 \cdot W_{21}) + (X_2 \cdot W_{22})$

$= (2 \cdot -1.5) + (-3 \cdot -0.7) = -0.9 \to \text{sigmoid} \to 0.289$

$y = (W_1 \cdot W_{31}) + (W_2 \cdot W_{32}) = (0.6 \times -0.2) + (0.289 \cdot 1.6)$

$y = 0.34 \qquad \text{Output} = 0.34$

c) Binary Cross-Entropy Loss Function

$$\mathcal{L}(W; X, Y) = -\sum_i [y_i \log p_i + (1-y_i)\log(1-p_i)]$$

$$\mathcal{L} = -[(1)(\log_2 0.34) + (1-1)(\log 0.66)] = 1.5564$$

Basically : $\mathcal{L} = -\ln(\hat{y})$ $\qquad \dfrac{\partial \mathcal{L}}{\partial \hat{y}} = -\dfrac{1}{\hat{y}}$ $\qquad -2.94$

d) $$\mathcal{L} = -\ln\Big(\underset{W_{31}}{-0.2} \cdot \underbrace{\max(0, \underset{W_{11}}{0.9} \cdot \underset{X_1}{2} \underset{X_2}{-3} W_{12})}_{\text{Neuron 1 (ReLU)}} + \underset{W_{32}}{1.6} \cdot \underbrace{\frac{1}{1+e^{-(-1.5\cdot 2 + -0.7 \cdot -3)}}}_{\text{Neuron 2 (Sigmoid)}}\Big)$$

Sigmoid

$$\mathcal{L} = -\ln(-0.2 \cdot \max(0, 1.8 - 3W_{12}) + 0.4625)$$

If $W_{12} < 0.6$: $\mathcal{L} = -\ln(0.6x + 0.1025)$

$W_{12} \geq 0.6$: $\mathcal{L} = -\ln(0.4625)$

From part b we see that $W_{12} = 0.4$

$$\mathcal{L} = -\ln(0.6x + 0.1025)$$

$$\frac{\partial \mathcal{L}}{\partial W_{12}} = \frac{1}{0.6W_{12} + 0.1025} \cdot 0.6$$

-1.75

e) The network has a total of 9 parameters (6 weights + 3 bias terms).

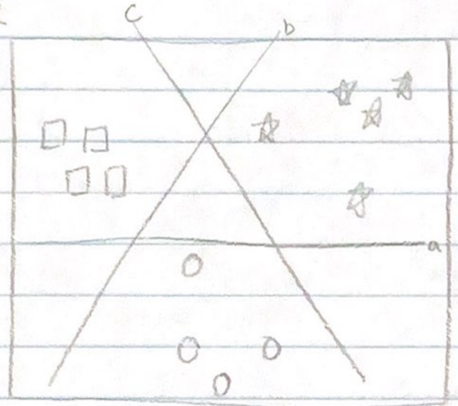
4. a) In logistic regression, the number of parameters is :

# of features + 1 (bias term) = 20 + 1 = 21 parameters

Total number of parameters = 5 (classes) × 21 = 105

We need 105 total parameters for a multi-class classification problem with 5 classes and 20 features.

b) OVR

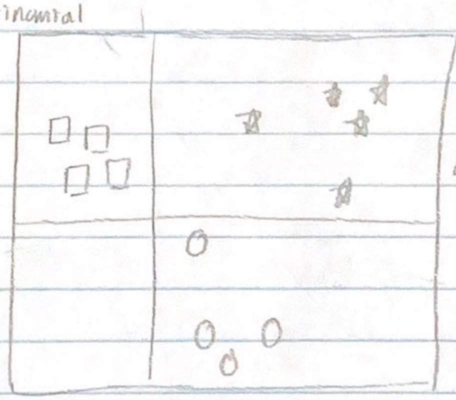


a - separates circles from squares + stars

b - separates squares from circles + stars

c - separates stars from circles + squares

Multinomial



Use the square as the reference group

Line 1 : predict y ≠ square from y = square

Line 2 : predict y = circle from y ≠ square

square vs star

and

square vs circles

Note : I realize my sketches of the multi-class classification aren't perfect, but hopefully you get the general idea.

3
4 > Neural Network

ReLU - piecewise straight line

tanh - much smoother

5. 1) C

Decision Tree shouldn't generate a curve like A or B. In general, Decision trees should have very clear-cut boundaries

Decision Trees make straight, hard boundaries

2) D

Random Forest should look similar to decision tree, but the boundaries are a bit less clear cut as many decision trees are averaged out to get the final model

Neural Network is a curve

3) B

ReLU is more of a piecewise straight line, and this is reflected in the decision boundary of B

4) A

tanh is a much smoother function than ReLU, and this is reflected in the decision boundary.