CS 475-675 Machine Learning: Midterm 1

Fall 2021

150 points.

Name (print): Dimitri Lezcano

JHED: dlezcan1

If you think a question is unclear or multiple answers are reasonable, please write a brief explanation of your answer, to be safe. Also, show your work if you want wrong answers to have a chance at some credit: it lets us see how much you understood.

This exam is open-book: permitted materials include textbooks, personal notes, lecture material, recitation material, past assignments, the course Piazza, and scholarly articles and papers. Other materials are otherwise not permitted. It is not permitted to discuss or share questions or solutions of this exam with any person, via any form of communication, other than the course instructors. It is not permitted to solicit or use any solutions to past exams for this course.

Declaration:

I have neither given nor received any unauthorized aid on this exam. In particular, I have not spoken to any other student about any part of this exam. The work contained herein is wholly my own. I understand that violation of these rules, including using an unauthorized aid, copying from another person, or discussing this exam with another person in any way, may result in my receiving a 0 on this exam.

Signature Date

Good luck!

True/False (50 points)

For each question, circle (or otherwise clearly indicate) either True or False. If False, explain why.
2 points for correct True/False answer, -2 points for incorrect True/False answer, 3 points for a correct explanation, 0 points for an incorrect explanation.

1) Minimizing the exponential loss is a good way of creating a classifier robust to outliers.
True False
Explanation:

False

Take (x, y) to be an outlier pair $\implies yf(x) < 0 \implies \exp(-yf(x)) > 0$. Therefore, say we have a f that correctly classifies 99 points and the point (x, y) is largely misclassified (i.e. $y_{100} = 1, f(x_{100}) = -100$). Then we would see that

$$\sum_{i=1}^{100} \exp(-y_i f(x_i)) \approx \exp(-y_{100} f(x_{100}))$$

$\because \forall i < 100 \exp(-y_i f(x_i)) < 1 \implies \sum_{i=1}^{99} \exp(-y_i f(x_i)) < 100 << \exp(100) = \exp(-y_{100} f(x_{100})$ $\therefore$ the loss is dominated by the outlier and therefore, exponential loss is not robust to outliers.

2) Minimising the 0-1 loss on the training data can never achieve good out of sample prediction performance.
True False
Explanation:

False

While 0-1 loss does not generalize to all datasets, if we have a distribution

$$p(y \mid x) = \begin{cases} 1 & y = 1 \\ 0 & y = 0 \end{cases}$$

then a classifier would only need to learn (from any sampled dataset of this distribution) to classify all inputs as $f(x) = \hat{y} = 1$ which can be learned using 0-1 loss.

3) Naive Bayes is a linear classifier.
True False
Explanation:

True

The decision boundary of a Naive Bayes classifier between two classes Y_i and Y_m is given by (using independence properties of Naive Bayes)

$$\log \frac{p(Y_i \mid X)}{p(Y_m \mid X)} = \log \frac{p(Y_i)}{p(Y_m)} + \sum_j \log \frac{p(X_j \mid Y_i)}{p(X_j \mid Y_m)} = constant + \sum_j g_j(X_j)$$

therefore, yields a linear decision boundary (in g_j), so Naive Bayes is a linear classifier.

4) Decreasing the bias of a predictor must necessarily increase its variance.
True False
Explanation:

False

Suppose $f_{true}(x) = x^2$ is the "true" function we are attempting to learn. When we attempt to learn the function using $\hat{f_1}(x;\beta) = ax + b$, there will be a significant amount of bias and very low variance from sampled data from the "true" function. However, if we add more parameters to decrease the bias of the predictor in the form of $\hat{f_2}(x;\beta) = ax^2 + bx + c$, we will not only decrease the bias of the predictor, but we will also decrease its variance since we are able to perfectly learn the function from a decent amount of sampled data.

5) $\mathbb{E}[\mathbb{E}[A \mid B]] = \mathbb{E}[A]$.
True False
Explanation:

True

$$\begin{aligned}
\mathbb{E}\left[A\right] &= \sum_a ap(a) \\
\mathbb{E}\left[\mathbb{E}\left[A \mid B\right]\right] &= \sum_b \sum_a ap(a \mid b) \\
&= \sum_a \sum_b ap(a \mid b) \because \text{summations commute} \\
&= \sum_a a \sum_b p(a \mid b) \because a \perp b \\
&= \sum_a ap(a) = \mathbb{E}\left[A\right]
\end{aligned}$$

6) Rosenblatt's Perceptron algorithm can be modified to solve multiclass classification problems, provided any pair of classes is linearly separable.
True False
Explanation:

True

For each class Y_l (say $l = 1, \ldots, k$), define a perceptron with parameters β_l such that $f_l(x) = \text{sign}(\beta_l^T x)$ (including bias term in dot product) be the decision rule for x to be classified as Y_l or not Y_l. So here, we have k perceptron classifiers, f_l, that determines whether each is either in a specific class or not. Since all of the classes are linearly separable, the class Y_l is linearly separable with not Y_l classes. So we train each perceptron to discriminate what is Y_l and what is not Y_l for $l = 1, \ldots, k$, since these classes are linearly separable with what is Y_l and what is not Y_l.

7) Naive Bayes is a special case of a Markov random field.
True False
Explanation:

True

Naive Bayes assumes $X_i \perp\!\!\!\perp X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_k \mid Y \; \forall i = 1, \ldots, k$. Take a graphical model, $\mathcal{G}$ where Y is the root node and X_i are all children of Y. In other words $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{X_1, \ldots, X_k, Y\}$ are the vertices and $\mathcal{E} = \{(X_i, Y) \mid i = 1, \ldots, k\}$ are the pair-wise edges of $\mathcal{G}$. Therefore, the maximal cliques of this graph are all of size 2 and we have the independences, $X_i \perp\!\!\!\perp X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_k \mid Y \; \forall i = 1, \ldots, k$, the same independence as Naive Bayes. Therefore by Hammersley-Clifford theorem, Naive Bayes factorizes w.r.t $\mathcal{G} \iff$ Naive Bayes obeys the global and local markov properties $\iff$ Naive Bayes is an MRF.

8) If $A \perp\!\!\!\perp C$ then either $A \perp\!\!\!\perp B$ or $B \perp\!\!\!\perp C$.
True False
Explanation:

False

B	A	C	$p(A, B, C)$
0	0	0	1/8
0	0	1	1/4
0	1	0	1/4
0	1	1	1/4
1	0	0	1/8
1	0	1	0
1	1	0	0
1	1	1	0

Here, we have that $p(A) = 1/2 = p(C)$ and $p(B) = 1/8$. From the above distribution, we have that $p(A = 1, B = 1) = 0 \neq p(A = 1)p(B = 1) = \frac{1}{2} \times \frac{1}{8}$. Similarly for $p(B = 1, C = 1) \neq p(B = 1)p(C = 1)$. However, if we consider $p(A, C)$:

A	C	$p(A, C)$	$p(A)p(C)$
0	0	1/4	1/4
0	1	1/4	1/4
1	0	1/4	1/4
1	1	1/4	1/4

therefore $A \perp\!\!\!\perp C$ and $A \not\perp\!\!\!\perp B$ nor $B \not\perp\!\!\!\perp C$.

9) A support vector machine applied to a linearly separable dataset must have at least 2 support vectors.
True False
Explanation:

True

The goal for an SVM is to maximize the margin between two classes. Therefore, in order to calculate a margin, at the minimum, it will need one vector from each class that are the closest to the (linear) decision boundary to define the margin. Otherwise, the margin may not be maximized should the SVM only have one support vector (i.e. maximum between a single support vector is ∞).

10) Logistic regression is a special case of the projection pursuit model with $M = 1$.
True False
Explanation:

False

Projection pursuit models: M different linear combinations of features $\beta_m^T \vec{x}$ to learn a complex non-linear link functions and add the results

$$f(\vec{x}) = \sum_{m=1}^{M} g_m(\beta_m^T \vec{x})$$

for a set of *unrestricted* $g_m(\cdot)$.

Here, logistic regression is actually a restricted moment model which learns $f(\vec{x}) = g(\vec{x}; \beta)$ where g is *fixed*, not unrestricted. Therefore, we have that logisitc regression is NOT a projection pursuit model.

Multiple Part Questions (100 points)

11) Probability And Likelihood (25 points)

(i) Show that if $A \perp\!\!\!\perp B \cup D \mid C$ then $A \perp\!\!\!\perp D \mid C$. You may want to use the fact that if $X \perp\!\!\!\perp Y \mid Z$ then $p(X, Y \mid Z) = p(X \mid Z)p(Y \mid Z)$.

(ii) Consider the following density function (for non-negative x): $f(x; \lambda) = \lambda \exp\{-\lambda x\}$. Derive the maximum likelihood estimate of its parameter λ.

(iii) What function of x does λ correspond to?

(iv) If, instead of solving for λ in closed form, we instead opted to use stochastic gradient descent to update our guess $\lambda_{(t)}$ to obtain $\lambda_{(t+1)}$, using the ith row of data x_i, and a learning parameter ρ, what would our update be?

Solution:

(i)

$$\begin{aligned}
p(A, D \mid C) &= \sum_b p(A, B = b, D \mid C) \\
&= \sum_b p(A \mid C)p(B = b, D \mid C) \because A \perp\!\!\!\perp B \cup D \mid C \\
&= p(A \mid C)\sum_b p(B = b, D \mid C) \\
&= p(A \mid C)p(D \mid C) \implies A \perp\!\!\!\perp D \mid C
\end{aligned}$$

(ii)

$$\begin{aligned}
\log \mathcal{L}_{[D]}(\lambda) &= \sum_i \log(f(x_i; \lambda)) \\
&= \sum_i -\lambda x_i + \log \lambda \\
\frac{\partial}{\partial \lambda} \log \mathcal{L}_{[D]} &= \sum_i -x_i + \frac{1}{\lambda} = 0 \implies \\
-\left(\sum_i x_i\right) + \frac{N}{\lambda} &= 0 \\
\frac{1}{\lambda} &= \frac{1}{N}\sum_i x_i \\
\lambda &= \left(\frac{1}{N}\sum_i x_i\right)^{-1}
\end{aligned}$$

(iii) λ_{MLE} corresponds to the inverse of the mean value of x.

(iv) Using (ii), we would find

$$\begin{aligned}
\frac{\partial}{\partial \lambda_i} \log \mathcal{L}_{(x_i)}(\lambda_i) &= -x_i + \frac{1}{\lambda_i} \\
\lambda_{i+1} &= \lambda_i - \rho \left(\frac{\partial}{\partial \lambda_i} \log \mathcal{L}_{(x_i)}(\lambda_i) \right) \\
&= \lambda_i - \rho \left(-x_i + \frac{1}{\lambda_i} \right)
\end{aligned}$$

12) **Classification (25 points)**

Consider the *Slightly Less Naive Bayes* classifier on features $X_1, \ldots, X_k$ and outcome Y, defined by the following independence restrictions:

$$\begin{aligned}
X_1 &\perp\!\!\!\perp X_3, \ldots, X_k \mid X_2, Y \\
X_k &\perp\!\!\!\perp X_1, \ldots, X_{k-2} \mid X_{k-1}, Y \\
X_i &\perp\!\!\!\perp X_1, \ldots, X_{i-2}, X_{i+2}, \ldots, X_k \mid X_{i-1}, X_{i+1}, Y
\end{aligned}$$

(i) Write down the joint likelihood $\mathcal{L}_{[D]} \equiv \prod_{i=1}^{n} p(\vec{x}_i, y_i)$ for this model. You may want to think about the MRF factorization corresponding to the above independences.

(ii) Write down the predictive model $p(Y \mid \vec{X})$ for this classifier.

(iii) Write down the decision boundary for classes Y_j and Y_m: $\log \frac{p(Y_j|\vec{X})}{p(Y_m|\vec{X})}$. Is this a linear decision boundary in $\vec{X}$?

(iv) Is the Naive Bayes model a submodel of the Slightly Less Naive Bayes model? Recall that a model is a set of distributions (that satisfy some independence restrictions in this case), and a submodel is a subset. In other words, all distributions in a submodel (subset) are also in a supermodel (superset). That is, elements of a submodel satisfy restrictions in a supermodel.

Hint: think about the graphoid axioms, and how Naive Bayes independences compare to Slightly Less Naive Bayes independences.

Solution:

(i) Note that the distribution $p(\vec{X} \mid Y)$ can be modelled as an MRF chain for

$$X_1 \mid Y — X_2 \mid Y — \ldots — X_{k-1} \mid Y — X_k \mid Y$$

Now, considering a single sample $(\vec{x_i}, y_i)$ and using the MRF factorization of maximal cliques for a

$$\begin{aligned}
p(\vec{x_i}, y_i) &= p(y_i) p(\vec{x_i} \mid y_i) \\
&= p(y_i) \frac{1}{Z} \prod_{j=1}^{k} \phi_{j|y_i}(x_i^{(j)} \mid y_i) \prod_{j=1}^{k-1} \phi_{j,j+1|y_i}(x_i^{(j)}, x_i^{(j+1)} \mid y_i) \\
\mathcal{L}_{[D]} &= \prod_{i=1}^{n} p(\vec{x_i}, y_i) \\
&= \prod_{i=1}^{n} \frac{1}{Z} p(y_i) \prod_{j=1}^{k} \phi_{j|y_i}(x_i^{(j)} \mid y_i) \prod_{j=1}^{k-1} \phi_{j,j+1|y_i}(x_i^{(j)}, x_i^{(j+1)} \mid y_i)
\end{aligned}$$

(ii) Using Bayes' rule,

$$\begin{aligned}
p(Y \mid \vec{X}) &= \frac{p(\vec{X} \mid Y)p(Y)}{\sum_y p(\vec{X} \mid Y = y)} \\
&= p(Y)\frac{\frac{1}{Z}\prod_{j=1}^{k}\phi_{j|Y}(X_j \mid Y)\prod_{j=1}^{k-1}\phi_{j,j+1|Y}(X_j, X_{j+1} \mid Y)}{\sum_y \frac{1}{Z}\prod_{j=1}^{k}\phi_{j|y}(X_j \mid Y = y)\prod_{j=1}^{k-1}\phi_{j,j+1|y}(X_j, X_{j+1} \mid Y = y)} \\
p(Y \mid X) &= p(Y)\frac{\prod_{j=1}^{k}\phi_{j|Y}(X_j \mid Y)\prod_{j=1}^{k-1}\phi_{j,j+1|Y}(X_j, X_{j+1} \mid Y)}{\sum_y \prod_{j=1}^{k}\phi_{j|y}(X_j \mid Y = y)\prod_{j=1}^{k-1}\phi_{j,j+1|y}(X_j, X_{j+1} \mid Y = y)}
\end{aligned}$$

(iii) Using (ii) and changing $Y_j \to Y_l$,

$$\begin{aligned}
\log\frac{p(Y_l \mid \vec{X})}{p(Y_m \mid \vec{X})} &= \log\left(\frac{p(Y_l)}{p(Y_m)}\frac{\prod_{j=1}^{k}\phi_{j|Y_l}(X_j \mid Y_l)\prod_{j=1}^{k-1}\phi_{j,j+1|Y_l}(X_j, X_{j+1} \mid Y_l)}{\prod_{j=1}^{k}\phi_{j|Y_m}(X_j \mid Y_m)\prod_{j=1}^{k-1}\phi_{j,j+1|Y_m}(X_j, X_{j+1} \mid Y_m)}\right) \\
&= \log\left(\frac{p(Y_l)}{p(Y_m)}\prod_{j=1}^{k}\frac{\phi_{j|Y_l}(X_j \mid Y_l)}{\phi_{j|Y_m}(X_j \mid Y_m)}\prod_{j=1}^{k-1}\frac{\phi_{j,j+1|Y_l}(X_j, X_{j+1} \mid Y_l)}{\phi_{j,j+1|Y_m}(X_j, X_{j+1} \mid Y_m)}\right) \\
&= \log\frac{p(Y_l)}{p(Y_m)} + \sum_{j=1}^{k}\log\frac{\phi_{j|Y_l}(X_j \mid Y_l)}{\phi_{j|Y_m}(X_j \mid Y_m)} + \sum_{j=1}^{k-1}\log\frac{\phi_{j,j+1|Y_l}(X_j, X_{j+1} \mid Y_l)}{\phi_{j,j+1|Y_m}(X_j, X_{j+1} \mid Y_m)} \\
&= \log\frac{p(Y_l)}{p(Y_m)} + \sum_{j=1}^{k}\big(\log\phi_{j|Y_l}(X_j \mid Y_l) - \phi_{j|Y_m}(X_j \mid Y_m)\big)\sum_{j=1}^{k-1}(\log\phi_{j,j+1|Y_l}(X_j, X_{j+1} \mid Y_l) \\
&\quad - \phi_{j,j+1|Y_m}(X_j, X_{j+1} \mid Y_m))
\end{aligned}$$

The above decision boundary is linear in $\vec{X}$ if $\forall j = 1, \ldots, k, \forall Y$ we have that $\log\phi_{j|Y}$ and $\log\phi_{j,j+1|Y}$ is linear in $\vec{X}$.

(iv) Naive Bayes' assumes

$$\begin{aligned}
&X_1 \perp\!\!\!\perp X_2, \ldots, X_k \mid Y \\
&X_k \perp\!\!\!\perp X_1, \ldots, X_{k-2} \mid Y \\
&X_i \perp\!\!\!\perp X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_k \mid Y \ \forall i = 2, \ldots, k-1
\end{aligned}$$

Given that we know using the graphoid axiom, *weak union*, $A \perp\!\!\!\perp B, C \mid D \implies A \perp\!\!\!\perp B \mid C, D$, we have that

$$\begin{aligned}
&X_1 \perp\!\!\!\perp X_2, \ldots, X_k \mid Y \implies X_i \perp\!\!\!\perp X_3, \ldots, X_k \mid X_2, Y \\
&X_k \perp\!\!\!\perp X_1, \ldots, X_{k-2} \mid Y \implies X_k \perp\!\!\!\perp X_1, \ldots, X_{k-3} \mid X_{k-2}, Y \\
&X_i \perp\!\!\!\perp X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_k \mid Y \implies X_i \perp\!\!\!\perp X_1, \ldots, X_{i-2}, X_{i+2}, \ldots, X_k \mid X_{i-1}, X_{i+1}, Y \\
&\quad \forall i = 2, \ldots, k-1
\end{aligned}$$

So Naive Bayes is a submodel of Slightly Less Naive Bayes.

13) **Neural Network Models (25 points)**

Consider a neural network with K inputs $\vec{X}$, and output Y.

(i) Assume there are $M > 10$ hidden layers with one node each. The first hidden node V_1 uses the sigmoid activation function $\sigma(x) = 1/(1+e^{-x})$ applied to a linear combination of $\vec{X}$ given by $\vec{\beta}_0^T\vec{X}$, all subsequent hidden nodes V_m ($m = 2, \ldots, M$) use a sigmoid activation function applied to a weighted combination of the value of the previous hidden node V_{m-1} and a bias term ($V_m = \text{sigmoid}(\beta_{m0} + \beta_{m1}V_{m-1})$), with the output Y being given by $\text{sigmoid}(\beta_{M0} + \beta_{M1}V_M)$.

Describe how the gradients with respect to $\vec{\beta}_0$ differ between layers. Justify your answer. Hint: calculate the derivative of $\sigma(x)$, and think about what will happen by chain rule of differentiation if a set of $\sigma(x)$ functions are composed.

(ii) Now replace all $\sigma(x)$ activation functions in the model in (i) by ReLU activation functions.

How does the gradient of the output with respect to $\vec{\beta}_0$ compare with that same gradient in (i)?

(iii) Assume there is a single hidden layer with K nodes, each with a $ReLU(x)$ activation function, and each feature in $\vec{X}$ is connected to exactly one hidden node. In other words, each hidden node V_k ($k = 1, \ldots, K$) is equal to $ReLU(\beta_k X_k)$. Furthermore, assume the output Y is given by $\sigma(\vec{\beta}^T(V_1, \ldots, V_K))$. Is this model equivalent to the logistic regression model? Explain.

(iv) As before, assume there is a single hidden layer with K nodes, each with a $\sigma(x)$ activation function. However, now each hidden node V_k ($k = 1, \ldots, K$) is given by $\sigma()$ applied to a weighted combination of $\vec{X}$, where weights for each V_k are *shared.* The outcome model is given by a weighted combination of $\vec{\alpha}^T(V_1, \ldots, V_K)$. Is this model sufficiently general to be able to approximate the logistic regression model? Explain.

Solution:

(i) First, note that $\frac{d\sigma}{dx} = \frac{d}{dx}(1+e^{-x})^{-1} = -e^{-x}(1+e^{-x})^{-2} = -\sigma(x)(1-\sigma(x))$.

Second, using the above, note that

$$\frac{\partial V_m}{\partial V_{m-1}} = \frac{\partial}{\partial V_{m-1}}\sigma(\beta_{m0} + \beta_{m1}V_{m-1}) = \beta_{m1}V_m(1 - V_m)$$

Therefore if we consider just looking at the bias terms,

$$\frac{\partial V_M}{\partial \beta_{m0}} = \frac{\partial V_M}{\partial V_{M-1}}\frac{\partial V_{M-1}}{\partial \beta_{m0}} = \frac{\partial V_m}{\partial \beta_{m0}}\prod_{j=m+1}^{M}\frac{\partial V_j}{\partial V_{j-1}}$$

$$= V_m(1 - V_m)\prod_{j=m+1}^{M}\beta_{j1}V_j(1 - V_j)$$

Now since V_j is a sigmoid, $0 \leq V_j(1-V_j) \leq \frac{1}{2} < 1$, therefore, as m gets smaller (closer to 1, closer to input layer) we see that the product term becomes smaller $\because$

$$\prod_{j=m+1}^{M} V_j(1-V_j) \leq \prod_{j=m+1}^{M} \frac{1}{2} \leq \left(\frac{1}{2}\right)^{M-m-1}$$

Similarly,

$$\frac{\partial V_M}{\partial \beta_{m1}} = \frac{\partial V_m}{\partial \beta_{m1}} \prod_{j=m+1}^{M} \frac{\partial V_j}{\partial V_{j-1}}$$

So we would have that as $m \to 1$, the proportionality of the gradients decays to zero $(\prod_{j=m+1}^{M} \frac{\partial V_j}{\partial V_{j-1}})$. This is also known as the *vanishing gradient* problem.

(ii) First, note that

$$\frac{d\,\mathrm{ReLU}(x)}{dx} = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Second, note that

$$\frac{\partial V_j}{\partial V_{j-1}} = \frac{\partial}{\partial V_{j-1}} \mathrm{ReLU}(\beta_{j0} + \beta_{j1}V_{j-1}) = \mathbb{I}_{>0}(\beta_{j0} + \beta_{j1}V_{j-1})\beta_{j1}$$

Therefore if we consider just looking at the bias terms,

$$\begin{aligned} \frac{\partial V_M}{\partial \beta_{m0}} &= \frac{\partial V_M}{\partial V_{M-1}} \frac{\partial V_{M-1}}{\partial \beta_{m0}} = \frac{\partial V_m}{\partial \beta_{m0}} \prod_{j=m+1}^{M} \frac{\partial V_j}{\partial V_{j-1}} \\ &= \mathbb{I}_{>0}(\beta_{m0} + \beta_{m1}V_{m-1}) \prod_{j=m+1}^{M} \mathbb{I}_{>0}(\beta_{j0} + \beta_{j1}V_{j-1})\beta_{j1} \end{aligned}$$

where for parameters that need to be updated, the indicator function for each of these higher layers would be 1, which would mean that the update is $\propto \prod_{j=m+1}^{M} \beta_{j1}$, rather than any decay terms. Similarly, just as in (i), the gradients for β_{m1} would have the same proportionality constant of 1.

(iii) We have here that $Y = \sigma(\beta^T \mathrm{ReLU}(X))$ while logistic regression $Y_{\log} = \sigma(\beta^T X)$, which does not have a non-linear activation (ReLU). In more mathematical terms

$$\begin{aligned} \sigma(\beta^T \mathrm{ReLU}(X)) &= \sigma(\beta^T X)\ \forall X \text{ iff} \\ \exp(-\beta^T \mathrm{ReLU}(X)) &= \exp(-\beta^T X)\ \forall X \text{ iff} \\ \beta^T \mathrm{ReLU}(X) &= \beta^T X\ \forall X \text{ iff} \\ \mathrm{ReLU}(X) &= X\ \forall X \text{ which is a contradiction} \end{aligned}$$

So no, this is not a logistic regression model.

(iv) The model's output is represented by $Y = \sum_{k=1}^{K} \alpha_k \sigma(\beta^T X)$. If we set $\alpha_1 = 1$ and $\alpha_{k>1} = 0$, we have that $Y = \sigma(\beta^T X) = Y_{\log}$, so the model is sufficiently general approximate logistic regression.

14) **AdaBoost and Exponential Loss (25 points)**

Assume a one-dimensional dataset with $x = \langle -1, 0, 1\rangle$ and $y = \langle -1, +1, -1\rangle$. Consider three weak classifiers:

$$h_1(x) = \begin{cases} 1 & \text{if } x > \frac{1}{2} \\ -1 & \text{otherwise} \end{cases}, \qquad h_2(x) = \begin{cases} 1 & \text{if } x > -\frac{1}{2} \\ -1 & \text{otherwise} \end{cases}, \qquad h_3(x) = 1.$$

(1) Show your work for the first 2 iterations of AdaBoost. In the table below, replace the "?" in the table below with the value of the quantity at iteration t. For the ε_t column, put its error rate (remember to use the correct row weights). For the h column, put the index of the weak learner with the best error rate at the current iteration. For the α_t column, put the update parameter appropriately calculated from ε_t. In addition, calculate the distribution over examples at the second iteration $D_2(1), D_2(2), D_2(3)$. Use the natural logarithm in your calculations (log with base e).

t	h_t	α_t	ε_t	$D_t(1)$	$D_t(2)$	$D_t(3)$
1	?	?	?	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
2	?	?	?	?	?	?

(2) Without performing an explicit calculation, make an argument for what the classifier will be that AdaBoost outputs in this case after 100 iterations? Explain.

(3) In general, getting near-perfect training accuracy in machine learning leads to over-fitting. However, AdaBoost can get perfect training accuracy without overfitting. Give a brief justification for why that is the case.

(4) The logistic regression likelihood (with outcome labels $y \in \{1, -1\}$) is equal to $\prod_{i=1}^{n} \frac{1}{1+\exp(-y_i\vec{\beta}\vec{x}_i)}$. Show that maximizing the likelihood is equivalent to minimizing the exponential loss: $\sum_{i=1}^{n} \exp(-y_i f(\vec{x}_i))$ for some $f(.)$.

Solution:

(1)

t	h_t	ε_t	α_t	$D_t(1)$	$D_t(2)$	$D_t(3)$
1	2	1/3	0.3466	1/3	1/3	1/3
2	2	1/3	0.3466	1/4	1/4	1/2
3	2	1/3	0.3466	1/6	1/6	2/3

(2) What we can see here is that $f_{Adaboost}(x) = h_2(x)$ will end up being the classifier after 100 iterations. What we can see here is that h_2 correctly classifies $x = -1 \& x = 0$ and misclassifies $x = +1$. $h_1 \& h_3$ both misclassify $x = +1$ as well as misclassify one of either $x = 0$ or $x = -1$. Therefore, h_2 will always be picked as the best weak learner therefore since its loss will always out-perform h_1 and h_3 since it's error rate will always be less as $x = +1$ will continued to get weighted heavier, but all of the classifiers misclassify $x = +1$. Therefore, no other learners will be selected by Adaboost, so $h_2(x)$ will be the learned classifier.

(3) AdaBoost minimizes the exponential loss of the classifier which focuses very heavily on misclassified data, increasing generalization error.

(4)

$$\begin{aligned}
\text{argmax}_\beta \prod_i \frac{1}{1+\exp(-y_i\beta x_i)} &= \text{argmin}_\beta \prod_i (1+\exp(-y_i\beta x_i)) \\
&= \text{argmin} \log \prod_i (1+\exp(-y_i\beta x_i)) \\
&= \text{argmin} \sum_i \log(1+\exp(-y_i\beta x_i)) \\
&\approx \text{argmin} \sum_i \exp(-y_i\beta x_i) \text{ using a first-order approximation.}
\end{aligned}$$

Therefore we see that for $f(x) = \beta x$, this is equivalent as minimizing exponential loss (for logistic regression fits that are within a first-order approximation).