Machine Learning

Supervised Learning

Question 1

10 / 10 pts

Supervised learning is when we give our learning algorithm the right answer *y* for each example to learn from. Which is an example of supervised learning?

- (●) spam filtering

  For instance, emails labeled as "spam" or "not spam" are examples used for training a supervised learning algorithm. The trained algorithm will then be able to predict with some degree of accuracy whether an unseen email is spam or not.

- ( ) predicting the average age of a group of customers

Question 2

10 / 10 pts

Which are the two common types of supervised learning? (Choose two)

- [x] Classification

  Classification predicts from among a limited set of categories (also called classes). These could be a limited set of numbers or labels such as "cat" or "dog".

- [x] Regression

  Regression predicts a number among potentially infinitely possible numbers.

- [ ] Clustering

Question 3

10 / 10 pts

Which of these is a type of unsupervised learning?

- [ ] Regression
- [x] Clustering

  Clustering groups data into groups or clusters based on how similar each item (such as a hospital patient or shopping customer) are to each other.

- [ ] Categorization

Linear Regression

Question 1 — 10 / 10 pts

For linear regression, the model is represented by $f_{w,b}(x) = wx + b$. Which of the following is the output or "target" variable?

- x
- ✓ y

  this notation is usually used for output or target.
- $\hat{y}$
- m

Question 2 — 10 / 10 pts

For linear regression, the model is $f_{w,b}(x) = wx + b$.

Which of the following are the inputs, or features, that are fed into the model and with which the model is expected to make a prediction?

- m
- w and b
- (x, y)
- ✓ x

  The x, the input features, are fed into the model to generate a prediction $f_{w,b}(x) = wx + b$

Question 3

10 / 10 pts

For linear regression, if you find parameters w and b so that $J(w, b)$ is very close to zero, what can you conclude?

- (selected) The selected values of the parameters ww and bb cause the algorithm to fit the training set really well.

  When the cost is small, this means that the model fits the training set well.

- The selected values of the parameters ww and bb cause the algorithm to fit the training set really poorly.

- This is never possible -- there must be a bug in the code.

Question 4

10 / 10 pts

Which of the following statement is correct for Gradient Descent method?

- Correct! Gradient Descent can always reach to a local minimum with a small enough fixed learning rate.
- It can only be used for linear regression model
- Correct! Can be used for multi-variable linear regression
- Correct! A large choice of learning rate could cause divergence.

Cost function for logistic regression

Question 1 10 / 10 pts

$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right)$$

In this lecture series, "cost" and "loss" have distinct meanings. Which one applies to a single training example?

- (selected) Loss

  In these lectures, loss is calculated on a single training example. It is worth noting that this definition is not universal. Other lecture series may have a different definition.

- Cost
- Both Loss and Cost
- Neither Loss nor Cost

Question 2 10 / 10 pts

For simplified Cost function, if the label $y^{(i)} = 0$, then what does this expression simplify to?

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -y^{(i)} \log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) - \left(1 - y^{(i)}\right) \log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)$$

- $-\log(1 - f_{\vec{w},b}(\mathbf{x}^{(i)})) - log(1 - f_{\vec{w},b}(\mathbf{x}^{(i)}))$
- $\log(f_{\vec{w},b}(\mathbf{x}^{(i)}))$
- (selected) $-\log(1 - f_{\vec{w},b}(\mathbf{x}^{(i)}))$

  When $y^{(i)} = 0$, the first term reduces to zero.

- $\log(1 - f_{\vec{w},b}(\mathbf{x}^{(i)})) + log(1 - f_{\vec{w},b}(\mathbf{x}^{(i)}))$

Gradient descent for logistic regression

Question 1

10 / 10 pts

Repeat for j = 1,2, … , n {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b); \qquad \frac{\partial}{\partial w_j} J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) x^{(i)}$$

$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b); \qquad \frac{\partial}{\partial b} J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)$$

} simultaneous updates

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$$

Which of the following two statements is a more accurate statement about gradient descent for logistic regression?

- The update steps look like the update steps for linear regression, but the definition of $f_{\vec{w},b}(\mathbf{x}^{(i)})$ is different.

  For logistic regression, $f_{\vec{w},b}(\mathbf{x}^{(i)})$ is the sigmoid function instead of a straight line.

- The update steps are identical to the update steps for linear regression.

Over Fit

Question 1

10 / 10 pts

Which of the following can address overfitting?

- [ ] Remove a random set of training examples
- [x] Apply regularization

  Regularization is used to reduce overfitting.
- [x] Select a subset of the more relevant features.

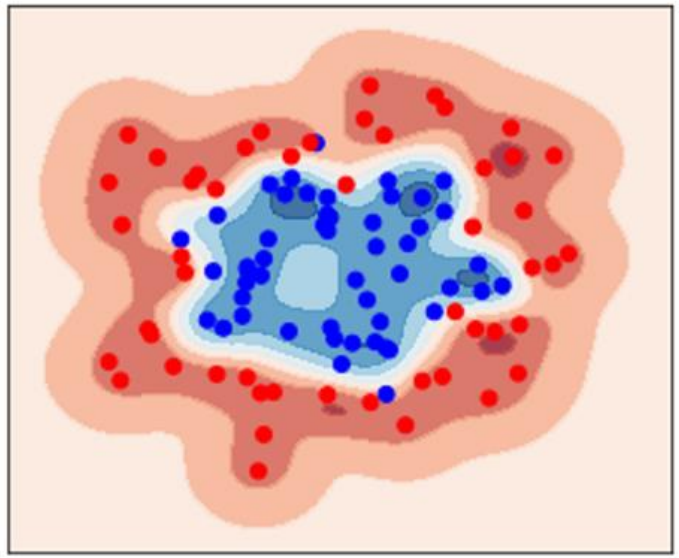
  If the model trains on the more relevant features, and not on the less useful features, it may generalize better to new examples.
- [x] Collect more training data

  If the model trains on more data, it may generalize better to new examples.

Question 2

10 / 10 pts

You fit logistic regression with polynomial features to a dataset, and your model looks like this.



What would you conclude? (Pick one)

Correct!

- [x] The model has high variance (overfit). Thus, adding data is likely to help

  The model has high variance (it overfits the training data). Adding data (more training examples) can help.

Question 3

10 / 10 pts

$$J(\vec{w}, b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(x^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} {w_j}^2$$

Suppose you have a regularized linear regression model. If you increase the regularization parameter λ, what do you expect to happen to the parameters $w_1, w_2, \ldots, w_n$?

- [ ] This will increase the size of the parameters $w_1, w_2, \ldots, w_n$

Correct!

- [x] This will reduce the size of the parameters $w_1, w_2, \ldots, w_n$

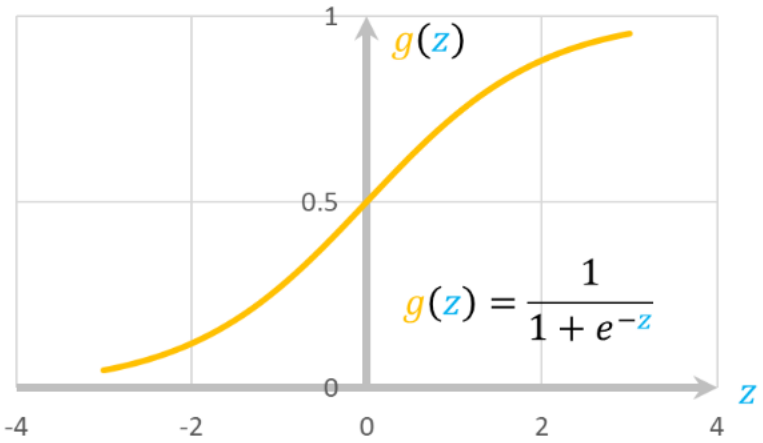
Regularization reduces overfitting by reducing the size of the parameters $w_1, w_2, \ldots w_n$.

Classification

Question 2

10 / 10 pts

Recall the sigmoid function is $g(z) = \frac{1}{1+e^{-z}}$



If z is a large positive number, then:

- $g(z)$ is near negative one (-1)
- $g(z)$ is near one (1)

  Say $z = +100$. So e^{-z} is then e^{-100}, a really small positive number. So, $g(z) = \frac{1}{1+\text{a small positive number}}$ which is close to 1
- $g(z)$ is near zero (0)

Question 1

10 / 10 pts

Which is an example of a classification task?

- Based on a patient's blood pressure, determine how much blood pressure medication (a dosage measured in milligrams) the patient should be prescribed.
- Based on a patient's age and blood pressure, determine how much blood pressure medication (measured in milligrams) the patient should be prescribed.
- Based on the size of each tumor, determine if each tumor is malignant (cancerous) or not.

  This task predicts one of two classes, malignant or not malignant.

Question 3

10 / 10 pts

A cat photo classification model predicts 1 if it's a cat, and 0 if it's not a cat. For a particular photograph, the logistic regression model outputs $g(z)$ (a number between 0 and 1). Which of these would be a reasonable criteria to decide whether to predict if it's a cat?

- Predict it is a cat if $g(z) = 0.5$
- $g(z) >= 0.5$

  Think of $g(z)$ as the probability that the photo is of a cat. When this number is at or above the threshold of 0.5, predict that it is a cat.
- Predict it is a cat if $g(z) > 0.7$
- Predict it is a cat if $g(z) < 0.5$

K-Nearest Neighbor

Question 1

10 / 10 pts

k-NN algorithm does more computation on test time rather than train time.

- (●) True

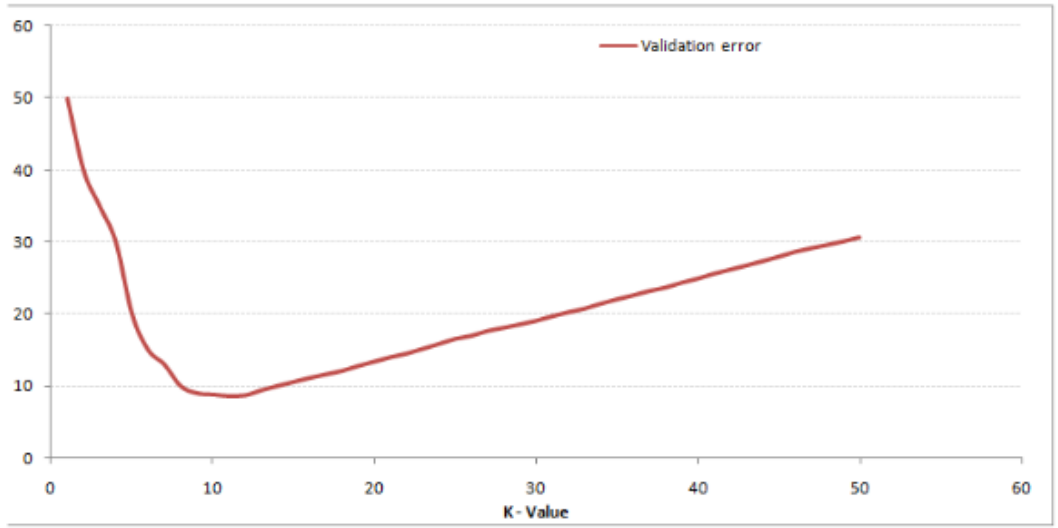
  In the testing phase, a test point is classified by assigning the label which are most frequent among the *k* training samples nearest to that query point – hence higher computation. That requires computing the distances and sorting them out.

- ( ) False

Question 2

10 / 10 pts

In the image below, which would be the best value for k assuming that the algorithm you are using is k-Nearest Neighbor.



- ( ) 1
- ( ) 50
- (●) 10

  Validation error is the least when the value of k is 10. So it is best to use this value of k

- ( ) 20

曼哈顿距离（Manhattan Distance）是一种距离度量方法，用于计算两个点在网格状路径上的距离。它得名于曼哈顿市的街道布局，因为在曼哈顿，街道是呈方格状的，人们只能沿着水平或垂直方向移动。

在二维空间中，曼哈顿距离的计算公式是：

$$\text{Manhattan Distance} = |x_2 - x_1| + |y_2 - y_1|$$

欧几里得距离（Euclidean Distance）是计算两个点之间的"直线"距离的度量方法，通常用于连续空间中。它基于勾股定理，给出了两点间的最短距离，也被称为"直线距离"或"线性距离"。

在二维空间中，两个点 (x_1, y_1) 和 (x_2, y_2) 之间的欧几里得距离公式为：

$$\text{Euclidean Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

在三维空间中，若两个点的坐标为 (x_1, y_1, z_1) 和 (x_2, y_2, z_2)，欧几里得距离的公式是：

$$\text{Euclidean Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

汉明距离是指两个相同长度的二进制字符串中不同位的数量。

Question 5 10 / 10 pts

Which of the following distances can be used as a metric in KNN method?

- (●) all of the above

  depending on the problem at hand, all the distances can be used.

- ( ) Hamming distance
- ( ) Manhattan distance
- ( ) Euclidean distance

Question 6

10 / 10 pts

check all the statements which are true for the KNN method.

- [x] KNN has high bias for larger K values

  This is correct. As K increases the complexity of the decision boundaries decreases and hence the KNN bias increases.

- [ ] KNN learns more about the nature of the training data if we use a larger training set.

- [x] KNN can be applied to both classification and regression problems

  That is correct. KNN can be applied to both problems.

- [ ] Euclidean distances is the only distance metric which is being used in KNN.

Question 7

10 / 10 pts

Which of the following will be true about k in KNN in terms of variance?

- ( ) None of these
- ( ) it depends.
- ( ) When you increase the k the variance increases.
- (•) When you decrease the k the variance will increases.

Question 8

10 / 10 pts

Unlike most machine learning algorithm, KNN works worse if all the features have the same scale.

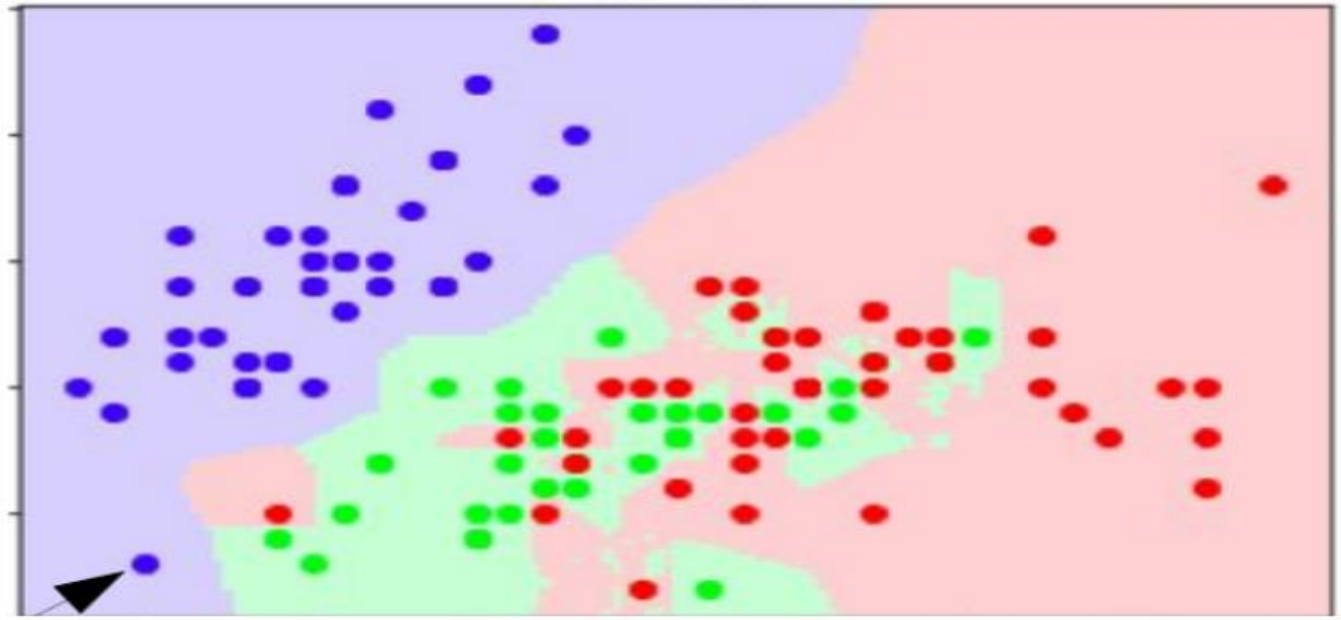
- ( ) True
- (•) False

  KNN measures distances between points therefore it is extra sensitive to the scale of the training data features.

Question 9

10 / 10 pts

What can you say about the following classification problem if it was done with KNN method?



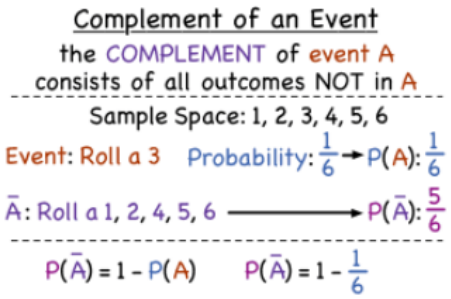
- [ ] The decision boundary is not piece-wise linear.
- [x] It would have high level of error against new set of data.

  That is correct, it has high variance so its error will be high against new set of data even though the training data is modeled perfectly.

- [ ] It should have a high bias against the training data.
- [x] Its K should be very small because the complexity of the decision boundaries is high.

Bayesian Classifiers

Question 1 — 10 / 10 pts



If $P(A) = 0.20$, $P(B|A) = 0.60$ and $P(B|\bar{A}) = 0.25$, then $P(A|B) = ?$

- 0.6250
- 0.3750

$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$, everything is given except $P(B)$. Using marginal probability $P(B) = P(B|A) \times P(A) + P(B|\bar{A}) \times P(\bar{A}) = 0.6 \times 0.2 + 0.25 \times (1 - 0.2) = 0.32$. Therefore $P(A|B) = \frac{0.6 \times 0.2}{0.32} = 0.375$

Question 2 — 10 / 10 pts

A virus has infected 1.8% of a population. A test detects this virus 95% of the time when it is actually present, but it returns a false positive 3% of the time when the virus is not present.
If a person selected at random from this population tests positive for the virus, what is the probability that this person is actually infected? [Round to the nearest percent.]

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(B) = P(B|A)P(A) + P(B|\bar{A})P(\bar{A})$$

- 37%

If you pay attention to the wording that is used, what you need to calculate is the following conditional probability. We know the test have come up positive so that is the known event.

P (infected | test is positive) = ?

Using Bayesian theorem this is equivalent to:

P (infected | test is positive) = P (test is positive | infected) x P (infected) /P (test is positive)

Among these the two on nominator are already give:

P (test is positive | infected) = 0.95

P (infected) = 0.018

In order to calculate the denominator we need to use marginal probability.

P (test is positive) = P (test is positive | infected) x P (infected) +P (test is positive | not infected) x P (not infected)

P (test is positive) = 0.95 x 0.018 + 0.03 x 0.982 = 0.04656

Finally :

P (infected | test is positive) = 0.95 x 0.018 / 0.04656 ≈ 0.37

Question 3

10 / 10 p

$$P(y = K|\vec{x}) = \frac{P(\vec{x}|y = K)P(y = K)}{P(\vec{x})}$$

$P(y = K|\vec{x})$, $P(\vec{x}|y = K)$, $P(y = K)$ and $P(\vec{x})$ are called posterior, likelihood, Prior and Evidence probabilities

Which statement is correct about the "prior" probability assumption in Bayesian classifier?

- $P(y = K) = \frac{1}{K}$, which means the probability of belonging to a category has to be the same for all K classes.
- None of the above
- It is a probability which must be given prior to collecting the data
- **(selected)** It can be $P(y = K) = \frac{m_K}{m}$, with m_K being the total number of training samples which belong to the category K, and m being the total number of training samples.

The prior assumption in Bayesian classifiers is that the probability of belonging to a specific class K is equal to the total number of training data

Question 4

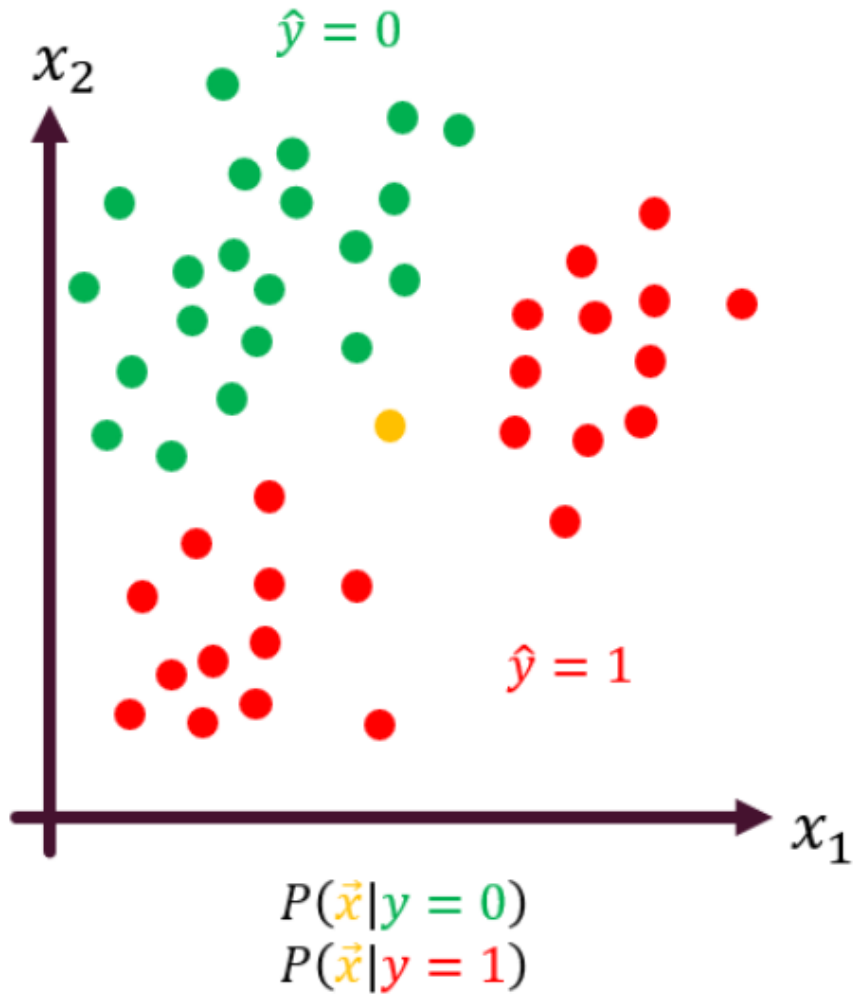
10 / 10 pts

In Bayesian classifiers the best choice of likelihood probability always comes from assuming the Gaussian probability density distribution.

- True
- **(selected)** False

Even though for most problem we do make this assumption but it is not correct to say that Gaussian distribution is "always" the best assumption. Similar to the case of customers visit hours to a restaurant, which we discussed in the class, one should look at the training data and see if Gaussian distribution is a good fit first.

In the following categorical example, what is the best assumption for likelihood probabilities of each class?



- $P(\vec{x}|y = 0)$ is best to be modeled by a two dimensional bimodal distribution while $P(\vec{x}|y = 1)$ is best to be modeled by a two dimensional Gaussian distribution.

- **$P(\vec{x}|y = 0)$ is best to be modeled by a two dimensional Gaussian distribution while $P(\vec{x}|y = 1)$ is best to be modeled by a two dimensional bimodal distribution.**

  That is correct. As it can be seen for category 0 most of the data are concentrated around a single space, but for category 1 the data are concentrated at two different spaces.

- Both $P(\vec{x}|y = 0)$ and $P(\vec{x}|y = 1)$ are best to be modeled by two dimensional Gaussian distribution.

- Both $P(\vec{x}|y = 0)$ and $P(\vec{x}|y = 1)$ are best to be modeled by two dimensional bimodal distributions.

Question 6

10 / 10 pts

All the features in Gaussian Naive Bayes are assumed to be uncorrelated.

- True
- False

This is incorrect. The features are assumed to be independent. Independence and uncorrelation are two different things.

Question 7

10 / 10 pts

All the features in Gaussian Naive Bayes are assumed to be independent.

- True

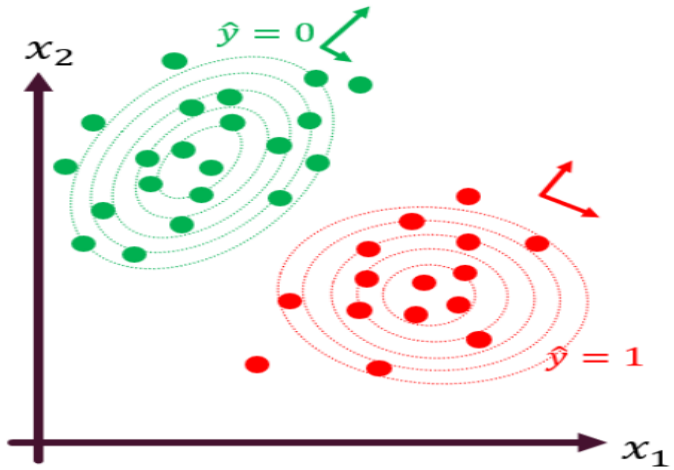
That is correct. Gaussian Naive Bayes classifiers assume all the features are independent.

- False

Question 8

10 / 10 pts

If the underlying probabilistic model of our data is a multivariate normal distribution, which one of the following has the highest chance to fit our training data the best?



- Gaussian Naive Bayes
- A Quadratic Discriminant Analysis.

That is correct. Considering the fact that in general the features are not independent and the fact that different classes might have different covariance matrices the most accurate form of analysis is the QDA. Both LDA and Naive Bayes make extra assumptions which could increase the inaccuracy of the analysis.

- A Linear Discriminant Analysis.
- None of the above

Neural Networks

Question 1 — 10 / 10 pts

Which of these are terms used to refer to components of an artificial neural network?

- [ ] axon
- [x] activation function

  Yes, an activation is the number calculated by a neuron (and "activations" in the figure above is a vector that is output by a layer that contains multiple neurons)
- [x] neurons

  Yes, a neuron is a part of a neural network
- [x] layers

  Yes, a layer is a grouping of neurons in a neural network

Question 2 — 10 / 10 pts

True/False? Neural networks take inspiration from, but do not very accurately mimic, how neurons in a biological brain learn.

- (•) True

  Artificial neural networks use a very simplified mathematical model of what a biological neuron does.
- ( ) False

Question 3 — 10 / 10 pts

For a neural network, what is the expression for calculating the activation of the third neuron in layer 2?

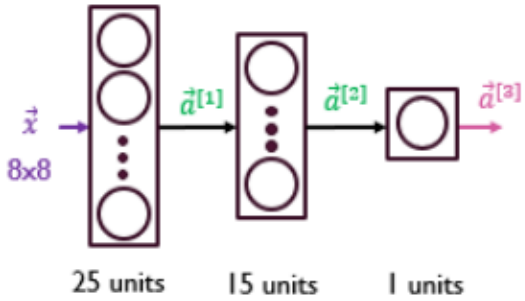
- ( ) $a_3^{[2]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[2]} + b_2^{[3]})$
- ( ) $a_3^{[2]} = g(\vec{w}_3^{[2]} \cdot \vec{a}^{[2]} + b_3^{[2]})$
- ( ) $a_3^{[2]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[1]} + b_2^{[3]})$
- (•) $a_3^{[2]} = g(\vec{w}_3^{[2]} \cdot \vec{a}^{[1]} + b_3^{[2]})$

  Yes! The superscript [2] refers to layer 2. The subscript 3 refers to the neuron in that layer. The input to layer 2 is the activation vector from layer 1.

Question 4

10 / 10 pts

For the handwriting recognition task discussed in lecture, what is the output $a_1^{[3]}$?



- A vector of several numbers that take values between 0 and 1
- A vector of several numbers, each of which is either exactly 0 or 1
- A number that is either exactly 0 or 1, comprising the network's prediction
- **The estimated probability that the input image is of a number 1, a number that ranges from 0 to 1.**

Yes! The neural network outputs a single number between 0 and 1.

Question 5

10 / 10 pts

How many layers the following neural network has?
model = Sequential([Dense(units=25, activation="sigmoid"),Dense(units=15, activation="sigmoid"),Dense(units=10, activation="sigmoid"),Dense(units=1, activation="sigmoid")])

- **4**
- 1
- 25
- 10

Question 6

10 / 10 pts

How do you define the second layer of a neural network that has 4 neurons and a sigmoid activation?

- Dense(layer=2, units=4, activation = 'sigmoid')
- **Dense(units=4, activation='sigmoid')**

Yes! This will have 4 neurons and a sigmoid activation.

- Dense(units=4)
- Dense(units=[4], activation=['sigmoid'])

Question 7

10 / 10 pts

For which type of task would you use the binary cross entropy loss function?

`

model.compile(loss=BinaryCrossentropy())

`

- regression tasks (tasks that predict a number)
- BinaryCrossentropy() should not be used for any task.
- binary classification (classification with exactly 2 classes)

  Yes! Binary cross entropy, which we've also referred to as logistic loss, is used for classifying between two classes (two categories).

- A classification task that has 3 or more classes (categories)

Question 8

10 / 10 pts

Here is code that you saw in the lecture:

```
model = Sequential([
    Dense(units=25, activation='sigmoid'),
    Dense(units=15, activation='sigmoid'),
    Dense(units=1, activation='sigmoid')
    ])
```

model.compile(loss=BinaryCrossentropy())

model.fit(X,y,epochs=100)

`

Which line of code updates the network parameters in order to reduce the cost?

- model.compile(loss=BinaryCrossentropy())
- model.fit(X,y,epochs=100)

  Yes! The third step of model training is to train the model on data in order to minimize the loss (and the cost)

- model = Sequential([...])
- None of the above -- this code does not update the network parameters.

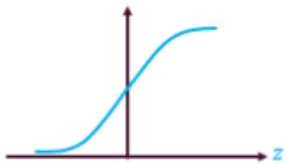
Question 9

10 / 10 pts

Which of the following activation functions is the most common choice for the hidden layers of a neural network?

$z = \vec{w}.\vec{x} + b$

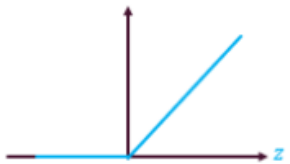
Sigmoid



$g(z) = \frac{1}{1+e^{-z}}$

$0 < g(z) < 1$

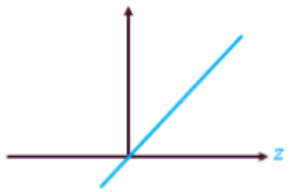
ReLU (Rectified Linear Unit)



$g(z) = \max(0, z)$

$0 \leq g(z)$

Linear activation function



$g(z) = z = \vec{w}.\vec{x} + b$

- Linear
- Most hidden layers do not use any activation function
- Sigmoid
- ReLU (rectified linear unit)

Yes! A ReLU is most often used because it is faster to train compared to the sigmoid. This is because the ReLU is only flat on one side (the left side) whereas the sigmoid goes flat (horizontal, slope approaching zero) on both sides of the curve.

Question 10

10 / 10 pts

For the task of predicting housing prices, which activation functions could you choose for the output layer? Choose the 2 options that apply.

- Sigmoid
- linear

Yes! A linear activation function can be used for a regression task where the output can be both negative and positive, but it's also possible to use it for a task where the output is 0 or greater (like with house prices).

- ReLU

Yes! ReLU outputs values 0 or greater, and housing prices are positive values.

Question 11

10 / 10 pts

A neural network with many layers but no activation function (in the hidden layers) is not effective; that's why we should instead use the linear activation function in every hidden layer.

- True
- False

Yes! A neural network with many layers but no activation function is not effective. A linear activation is the same as "no activation function".

Question 12

10 / 10 pts

- For K different category each category we will define a different z parameter.

$$z_j = \vec{w}_j . \vec{x} + b_j \qquad j = 1, \ldots, K$$

- Then the probability of the feature belonging to each category can be modeled as.

$$a_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} = \frac{e^{z_j}}{e^{z_1} + e^{z_2} + \cdots + e^{z_K}} = \mathrm{P}(y = j|\vec{x})$$

For a multiclass classification task that has 4 possible outputs, the sum of all the activations adds up to 1. For a multiclass classification task that has 3 possible outputs, the sum of all the activations should add up to ....

- More than 1
- 1

Yes! The sum of all the softmax activations should add up to 1. One way to see this is that if $e^{z_1} = 10, e^{z_2} = 20, e^{z_3} = 30$, then the sum of $a_1 + a_2 + a_3$ is equal to $\frac{e^{z_1}+e^{z_2}+e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}$ which is 1.

- Less than 1
- It will vary, depending on the input x.

Question 13

10 / 10 pts

- For Logistic Regression:

$$a_1 = g(z) = \frac{1}{1+e^{-z}} = \mathrm{P}(y=1|\vec{x})$$
$$a_2 = 1 - a_1 \qquad = \mathrm{P}(y=0|\vec{x})$$

- We defined the loss as follow:

$$Loss = \begin{cases} -\log(a_1) & if \quad y=1 \\ -\log(1-a_1) & if \quad y=0 \end{cases}$$

$$Loss = \begin{cases} -\log(a_1) & if \quad y=1 \\ -\log(a_2) & if \quad y=0 \end{cases}$$

- For softmax regression:

$$a_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} = \mathrm{P}(y=j|\vec{x}) \qquad j = 1, \ldots, K$$

- Similarly, the loss will be defined as:

$$Loss = \begin{cases} -\log(a_1) & if \quad y=1 \\ -\log(a_2) & if \quad y=2 \\ \vdots & \\ -\log(a_K) & if \quad y=K \end{cases}$$

- This is called Crossentropy loss.

For multiclass classification, the cross entropy loss is used for training the model. If there are 4 possible classes for the output, and for a particular training example, the true class of the example is class 3 (y=3), then what does the cross entropy loss simplify to? [Hint: This loss should get smaller when a_3 gets larger.]

- $\frac{z_3}{z_1+z_2+z_3+z_4}$
- z_3
- $\frac{-log(a_1)+-log(a_2)+-log(a_3)+-log(a_4)}{4}$
- **(selected)** $-log(a_3)$

Correct. When the true label is 3, then the cross entropy loss for that training example is just the negative of the log of the activation for the third neuron of the softmax. All other terms of the cross entropy loss equation $(-log(a_1), -log(a_2), and - log(a_4))$ are ignored

Question 14

10 / 10 pts

model

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
model = Sequential([
  Dense(units=25, activation='relu')
  Dense(units=15, activation='relu')
  Dense(units=10, activation='linear') )]
```

loss

```
from tensorflow.keras.losses import
  SparseCategoricalCrossentropy
model.compile(...,loss=SparseCategoricalCrossentropy(from_logits=True) )
```

fit

```
model.fit(X,Y,epochs=100)
```

predict

```
logits = model(X)
f_x = tf.nn.softmax(logits)
```

For multiclass classification, the recommended way to implement softmax regression is to set from_logits=True in the loss function, and also to define the model's output layer with...

- **(selected)** a 'linear' activation

Yes! Set the output as linear, because the loss function handles the calculation of the softmax with a more numerically stable method.

- a 'softmax' activation

Python Practice

Question 1

10 / 10 pts

What does Python print as it executes the following sequence of statements?

```
a = 'Honda'
b = 'Audi'
print(a + b)
```

- Honda Audi
- HondaAudi
- 'Honda Audi'
- 'HondaAudi'

Question 4

10 / 10 pts

Using the list defined below, what does Python print when these statements are executed?

```
L = ['Ford', 'Chevrolet', 'Toyota', 'Nissan', 'Tesla']
for x in L:
    print('Item:', x)
```

- Ford
  Chevrolet
  Toyota
  Nissan
  Tesla
- There would be a syntax error
- Item: Ford
  Item: Chevrolet
  Item: Toyota
  Item: Nissan
  Item: Tesla
- 1 Ford
  2 Chevrolet
  3 Toyota
  4 Nissan
  5 Tesla

Question 5

10 / 10 pts

What is the output of the following `display()` function?

```
def display(**kwargs):
    for i in kwargs:
        print(i)

display(emp="Kelly", salary=9000)
```

- No answer text provided.
- No answer text provided.
- TypeError
- Kelly
  9000
- ('emp', 'Kelly')
  ('salary', 9000)
- No answer text provided.
- emp
  salary

To **accept Variable Length of Keyword Arguments**, i.e., To create functions that take `n` number of **Keyword arguments** we use `**kwargs` (prefix a parameter name with a double asterisk `**`).

Question 6

10 / 10 pts

What is the output of the following `display_person()` function call

```
def display_person(*args):
    for i in args:
        print(i)

display_person(name="Emma", age="25")
```

- Emma
  25
- name
  age
- TypeError

To **accept Variable Length of Keyword Arguments**, i.e., To create functions that take n number of Keyword arguments we use `**kwargs` (prefix a parameter name with a double asterisk `**`).

Question 7

10 / 10 pts

Choose the correct function declaration of `fun1()` so that we can execute the following function call successfully

- def fun1(args*)
- def fun1(**kwargs)
- No, it is not possible in Python
- **def fun1(*data)** ✓

To accept multiple values or if the number of arguments is unknown, we can add `*` before the parameter name to accept arbitrary arguments. i.e.,
To **accept Variable Length of Positional Arguments**, i.e., To create functions that take n number of Positional arguments we use `*args` (prefix a parameter name with an asterisk `*`).

Question 8

10 / 10 pts

Python function always returns a value

- **True** ✓

If you do not include any `return` statement in function, it automatically returns `None`. So, in Python function always returns a value.

- False

Question 9

10 / 10 pts

Which of the following keywords are used to create a loop in Python?

- do
- loop
- **while** ✓
- foreach
- **for** ✓

Question 10

10 / 10 pts

Find the output of the code given below.

```
# Code snippet starts
import math
def sqr(a):
    return a*a
def root(a):
    return math.sqrt(a)
def calc(a):
    l = []
    l.append(sqr(a))
    l.append(int(root(a)))
    return l[1]
print(calc(5))
```

- (●) 2
- ( ) 1.2
- ( ) 2.532
- ( ) 4

HW1

```
# UNQ_C1
# GRADED FUNCTION: compute_cost

def compute_cost(x, y, w, b):
    """
    Computes the cost function for linear regression.

    Args:
        x (ndarray): Shape (m,) Input to the model (Population of cities)
        y (ndarray): Shape (m,) Label (Actual profits for the cities)
        w, b (scalar): Parameters of the model

    Returns
        total_cost (float): The cost of using w,b as the parameters for linear regression
                   to fit the data points in x and y
    """
    # number of training examples
    m = x.shape[0]

    # You need to return this variable correctly
    total_cost = 0

    ### START CODE HERE ###
    # For each example
    cost_sum = 0
    for i in range(m):
        f_wb = w * x[i] + b
        cost_i = (f_wb - y[i]) ** 2
        cost_sum += cost_i
    total_cost = (1/(2 * m)) * cost_sum
    ### END CODE HERE ###

    return total_cost
```

```
from re import M
# UNQ_C2
# GRADED FUNCTION: compute_gradient
def compute_gradient(x, y, w, b):
    """
    Computes the gradient for linear regression
    Args:
      x (ndarray): Shape (m,) Input to the model (Population of cities)
      y (ndarray): Shape (m,) Label (Actual profits for the cities)
      w, b (scalar): Parameters of the model
    Returns
      dj_dw (scalar): The gradient of the cost w.r.t. the parameters w
      dj_db (scalar): The gradient of the cost w.r.t. the parameter b
     """

    # Number of training examples
    m = x.shape[0]

    # You need to return the following variables correctly
    dj_dw = 0
    dj_db = 0

    ### START CODE HERE ###
    # For each example
    for i in range(m):
        f_wb = w * x[i] + b
        dj_dw_i = (f_wb - y[i]) * x[i]
        dj_db_i = f_wb - y[i]
        dj_dw += dj_dw_i
        dj_db += dj_db_i
    dj_dw = dj_dw / m
    dj_db = dj_db / m
    ### END CODE HERE ###


    # s = ?
    # dj_db = ?
    # dj_dw = ?
    # return dj_db, dj_dw

    return dj_dw, dj_db
```

```
            running gradient descent
    """

    # number of training examples
    m = len(x)

    # An array to store cost J and w's at each iteration — primarily for graphing later
    J_history = []
    w_history = []
    w = copy.deepcopy(w_in)   #avoid modifying global w within function
    b = b_in

    for i in range(num_iters):

        # Calculate the gradient and update the parameters
        dj_dw, dj_db = gradient_function(x, y, w, b )

        # Update Parameters using w, b, alpha and gradient
        w = w - alpha * dj_dw
        b = b - alpha * dj_db

        # Save cost J at each iteration
        if i<100000:           # prevent resource exhaustion
            cost =  cost_function(x, y, w, b)
            J_history.append(cost)

        # Print cost every at intervals 10 times or as many iterations if < 10
        if i% math.ceil(num_iters/10) == 0:
            w_history.append(w)
            print(f"Iteration {i:4}: Cost {float(J_history[-1]):8.2f}   ")

    return w, b, J_history, w_history #return w and J,w history for graphing
```

HW2

```
"""

### START CODE HERE ###
g = 1/(1+np.exp(-z))

### END SOLUTION ###

return g
```

```
### START CODE HERE ###
# Hint: Use the sigmoid() you defined
# e.g. f = sigmoid(z)
f = sigmoid(np.dot(X, w) + b)
total_cost = -(1/m)*np.sum(y*np.log(f) + (1-y)*np.log(1-f))

### END CODE HERE ###

return total_cost
```

```
        ### START CODE HERE ###
#         for i in range(m):
#             z_wb = None
#             for j in range(n):
#                 z_wb += None
#             z_wb += None
#             f_wb = None

#             dj_db_i = None
#             dj_db += None

#             for j in range(n):
#                 dj_dw[j] = None
        for i in range(m):
            z_wb = np.dot(X[i], w) + b
            f_wb = sigmoid(z_wb)
            dj_db_i = f_wb - y[i]
            dj_db += dj_db_i
            for j in range(n):
                dj_dw[j] += dj_db_i * X[i,j]
        dj_db = dj_db / m
        dj_dw = dj_dw / m

        ### END CODE HERE ###


        return dj_db, dj_dw
```

```
    ### START CODE HERE ###
    z = np.dot(X, w) + b
    f = sigmoid(z)
    p = 1 / (1 + np.exp(-z))
    p = np.where(p >= 0.5, 1, 0)

    ### END CODE HERE ###
    return p
```

```
    dj_db, dj_dw = compute_gradient(X, y, w, b)

    ### START CODE HERE ###
    dj_dw += (lambda_/m)*w

    ### END CODE HERE ###

    return dj_db, dj_dw
```

```
    ### START CODE HERE ###

    for i in range(n):
        reg_cost += w[i]**2
    reg_cost = (lambda_/(2*m))*reg_cost
    total_cost = cost_without_reg + reg_cost

    ### END CODE HERE ###

    # Add the regularization cost to get the total cost


    return total_cost
```