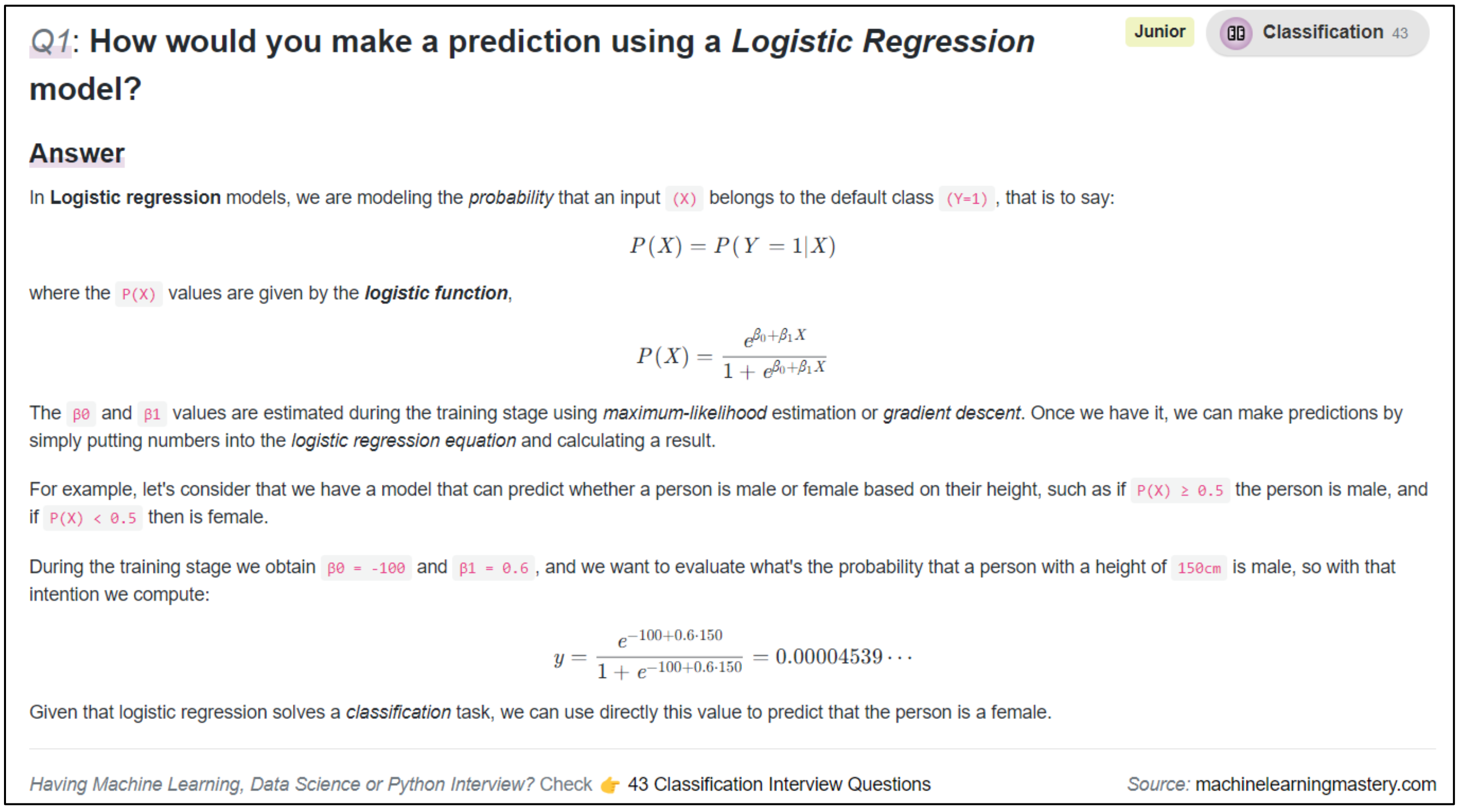
23 Logistic Regression Interview Questions (SOLVED) To Nail on ML Interview



Q1: How would you make a prediction using a *Logistic Regression* model?

Junior | Classification 43

Answer

In **Logistic regression** models, we are modeling the *probability* that an input `(X)` belongs to the default class `(Y=1)`, that is to say:

$$P(X) = P(Y = 1|X)$$

where the `P(X)` values are given by the ***logistic function***,

$$P(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

The `β0` and `β1` values are estimated during the training stage using *maximum-likelihood* estimation or *gradient descent*. Once we have it, we can make predictions by simply putting numbers into the *logistic regression equation* and calculating a result.

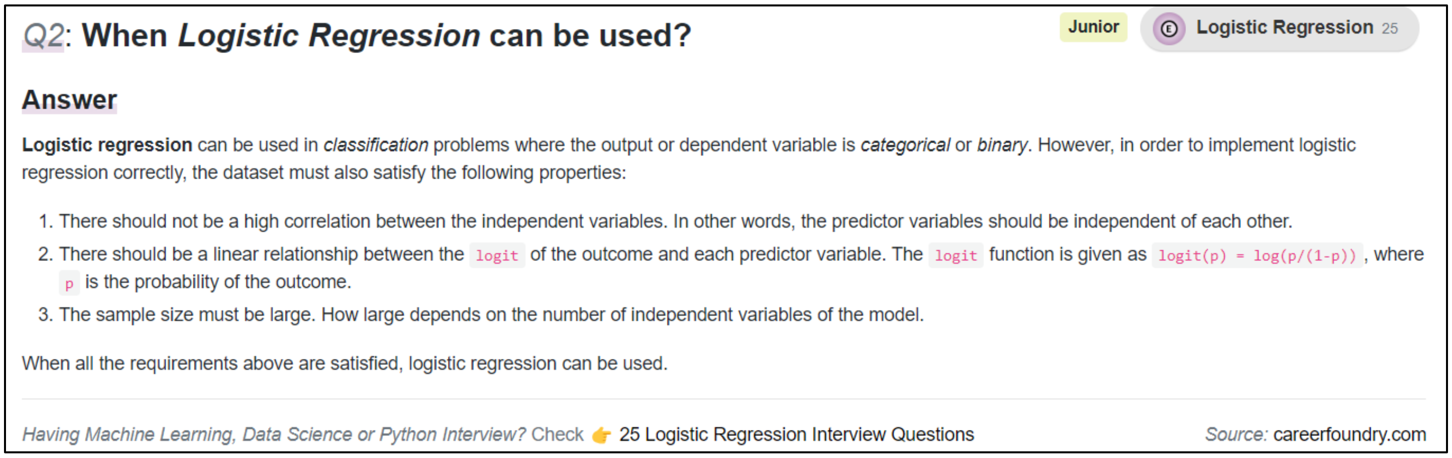
For example, let's consider that we have a model that can predict whether a person is male or female based on their height, such as if `P(X) ≥ 0.5` the person is male, and if `P(X) < 0.5` then is female.

During the training stage we obtain `β0 = -100` and `β1 = 0.6`, and we want to evaluate what's the probability that a person with a height of `150cm` is male, so with that intention we compute:

$$y = \frac{e^{-100+0.6\cdot 150}}{1 + e^{-100+0.6\cdot 150}} = 0.00004539\cdots$$

Given that logistic regression solves a *classification* task, we can use directly this value to predict that the person is a female.

Having Machine Learning, Data Science or Python Interview? Check 👉 43 Classification Interview Questions *Source:* machinelearningmastery.com



Q2: When *Logistic Regression* can be used?

Junior | Logistic Regression 25

Answer

Logistic regression can be used in *classification* problems where the output or dependent variable is *categorical* or *binary*. However, in order to implement logistic regression correctly, the dataset must also satisfy the following properties:

1. There should not be a high correlation between the independent variables. In other words, the predictor variables should be independent of each other.
2. There should be a linear relationship between the `logit` of the outcome and each predictor variable. The `logit` function is given as `logit(p) = log(p/(1-p))`, where `p` is the probability of the outcome.
3. The sample size must be large. How large depends on the number of independent variables of the model.

When all the requirements above are satisfied, logistic regression can be used.

Having Machine Learning, Data Science or Python Interview? Check 👉 25 Logistic Regression Interview Questions *Source:* careerfoundry.com

Q3: Why is Logistic Regression called *Regression* and not *Classification*?

Junior | Logistic Regression 25

Answer

Although the task we are targeting in logistic regression is a classification, *logistic regression does not actually individually classify things for you*: it just gives you probabilities (or log odds ratios in the logit form).

The only way logistic regression can actually classify stuff is if you apply a rule to the probability output. For example, you may round probabilities greater than or equal to `50%` to `1`, and probabilities less than `50%` to `0`, and that's your classification.

Having Machine Learning, Data Science or Python Interview? Check 👉 25 Logistic Regression Interview Questions *Source:* ryxcommar.com

Q4: Compare *SVM* and *Logistic Regression* in handling outliers

Mid | SVM 56

Answer

- For **Logistic Regression**, outliers can have an *unusually large effect* on the estimate of logistic regression coefficients. It will find a linear boundary if it exists to accommodate the outliers. To solve the problem of outliers, sometimes a sigmoid function is used in logistic regression.
- For **SVM**, outliers can make the decision boundary deviate severely from the optimal hyperplane. One way for SVM to get around the problem is to intrduce *slack variables*. There is a penalty involved with using slack variables, and how SVM handles outliers depends on how this penalty is imposed.

Having Machine Learning, Data Science or Python Interview? Check 👉 56 SVM Interview Questions *Source:* www.researchgate.net

Q5: How a *Logistic Regression* model is trained?

Mid | Logistic Regression 25

Answer

The **logistic model** is trained through the **logistic function**, defined as:

$$P(y) = \frac{1}{1 + e^{-wx}}$$

where `x` is the input data, `w` is the weight vector, `y` is the output label, and `P(y)` is the probability that the output label belongs to one class. If for some input we got `P(y) > 0.5`, then the predicted output is `1`, and otherwise would be `0`.

The training is based in estimate the `w` vector. For this, in each training instance we use ***Stochastic Gradient Descent*** to calculate a prediction using some initial values of the coefficients, and then calculate new coefficient values based on the error in the previous prediction. The process is repeated for a fixed number of iterations or until the model is accurate enough or cannot be made any more accurate.

Having Machine Learning, Data Science or Python Interview? Check 👉 25 Logistic Regression Interview Questions *Source:* machinelearningmastery.com

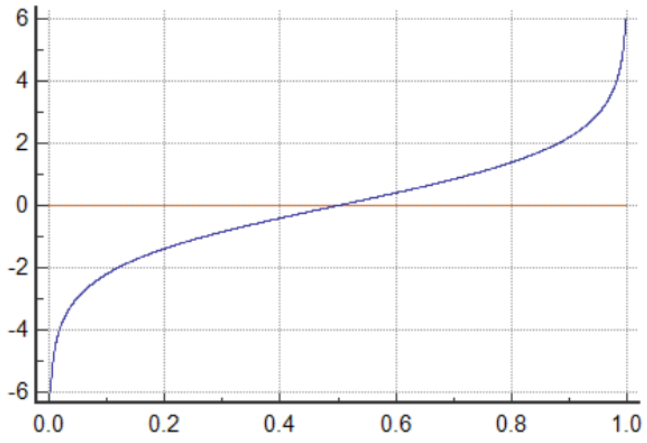
Q6: How do you use a supervised *Logistic Regression* for Classification?

Mid

Classification 43

Answer

- ***Logistic regression*** is a statistical model that utilizes ***logit*** function to model classification problems. It is a regression analysis to conduct when the dependent variable is *binary*. The **logit** function is shown below:



- Looking at the logit function, the next question that comes to mind is *how to fit that graph/equation*. The fitting of the logistic regression is done using the *maximum likelihood* function.
- In a supervised logistic regression, **features** are mapped onto the **output**. The output is usually a categorical value (which means that it is mapped with one-hot vectors or binary numbers).
- Since the **logit** function always outputs a value between `0` and `1`, it gives the **probability of the outcome**.

Q7: Provide a mathematical intuition for Logistic Regression?

Mid | Logistic Regression 25

Answer

Logistic regression can be seen as a **transformation** from linear regression to logistic regression using the logistic function, also known as the **sigmoid function** or S(x):

$$S(x) = \frac{1}{1 + e^{-x}}$$

Given the linear model:

$y = b_0 + b_1 \cdot x$

If we apply the sigmoid function to the above equation it results:

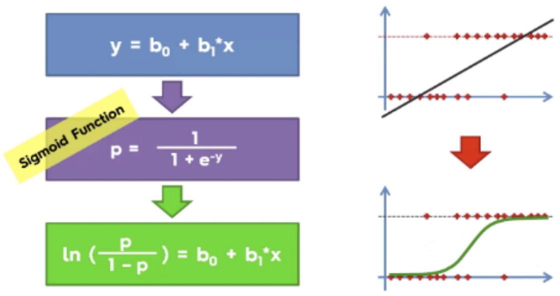
$S(y) = \frac{1}{1+e^{-y}} = p$

where `p` is the probability and it takes values between 0 and 1. If we now apply the logit function to `p`, it results:

$logit(p) = log(\frac{p}{1-p}) = b_0 + b_1 \cdot x$

The equation above represents the logistic regression. It fits a logistic curve to set of data where the dependent variable can only take the values 0 and 1.

The previous transformation can be illustrated in the following figure:



logistic function (also called the 'inverse logit').

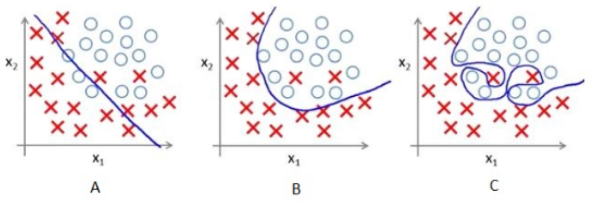
Having Machine Learning, Data Science or Python Interview? Check 👉 25 Logistic Regression Interview Questions — *Source:* medium.com

Q8: What can you infer from each of the hand drawn decision boundary of *Logistic Regression* below?

Mid | Logistic Regression 25

Problem

Also, what should we do to fix the problem of each decision boundary?



Answer

What can we infer:

- **A**: the model *underfits* the data. It will give us the maximum error compared to other two models.
- **B**: *best-fitting* model.
- **C**: the model *overfits* the data. It performs exceptionally well on training data but performs considerably worse on test data.

What can we do to fix the problem:

- **A**: *increase* the complexity of the model or *increase* the number of independent variables.
- **B**: best performing model, so we don't need to tweak anything.
- **C**: add *regularization* method to the model.

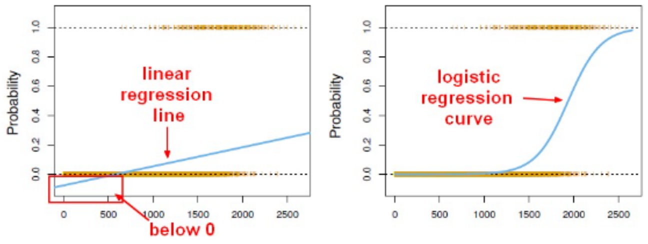
Having Machine Learning, Data Science or Python Interview? Check 👉 25 Logistic Regression Interview Questions — *Source:* www.analyticsvidhya.com

Q9: What is the difference between *Linear Regression* and *Logistic Regression*?

Mid | Logistic Regression 25

Answer

- **Linear regression output as probabilities** In linear regression, the outcome (dependent variable) is continuous. It can have any one of an infinite number of possible values. In logistic regression, the outcome (dependent variable) has only a limited number of possible values.



- **Outcome** In linear regression, the outcome (dependent variable) is continuous. It can have any one of an infinite number of possible values. In logistic regression, the outcome (dependent variable) has only a limited number of possible values.
- **The dependent variable** Logistic regression is used when the response variable is categorical in nature. For instance, yes/no, true/false, red/green/blue, 1st/2nd/3rd/4th, etc. Linear regression is used when your response variable is continuous. For instance, weight, height, number of hours, etc.
- **Equation** Linear regression gives an equation which is of the form Y = mX + C, means equation with degree 1. However, logistic regression gives an equation which is of the form Y = eX + e-X
- **Coefficient interpretation** In linear regression, the coefficient interpretation of independent variables are quite straightforward (i.e. holding all other variables constant, with a unit increase in this variable, the dependent variable is expected to increase/decrease by xxx). However, in logistic regression, depends on the family (binomial, Poisson, etc.) and link (log, logit, inverse-log, etc.) you use, the interpretation is different.
- **Error minimization technique** Linear regression uses *ordinary least squares* method to minimise the errors and arrive at a best possible fit, while logistic regression uses *maximum likelihood* method to arrive at the solution. Linear regression is usually solved by minimizing the least squares error of the model to the data, therefore large errors are penalized quadratically. Logistic regression is just the opposite. Using the logistic loss function causes large errors to be penalized to an asymptotically constant. Consider linear regression on categorical {0, 1} outcomes to see why this is a problem. If your model predicts the outcome is 38, when the truth is 1, you've lost nothing. Linear regression would try to reduce that 38, logistic wouldn't (as much)2.

Having Machine Learning, Data Science or Python Interview? Check 👉 25 Logistic Regression Interview Questions — *Source:* stackoverflow.com

Q10: What's the difference between *Softmax* and *Sigmoid* functions?

Mid | Classification 43

Answer

- **Softmax function**:
  - Is used for *multi-class* classification in logistic regression models, when we have only *one right answer* or *mutually exclusive* outputs.
  - Its probabilities sum will be `1`.
  - Is used in *different layers* of neural networks.
  - It is defined as:

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \forall \text{j} = 1\ldots\text{K}$$

- **Sigmoid function**:
  - Is used for *multi-label* classification in logistic regression models, when we have *more than one right answer* or *non-exclusive* outputs.
  - Its probabilities sum does not need to be `1`.
  - Is used as an *activation function* while building neural networks.
  - It is defined as:

$$\sigma(z_j) = \frac{e^{z_j}}{1 + e^{z_j}}$$

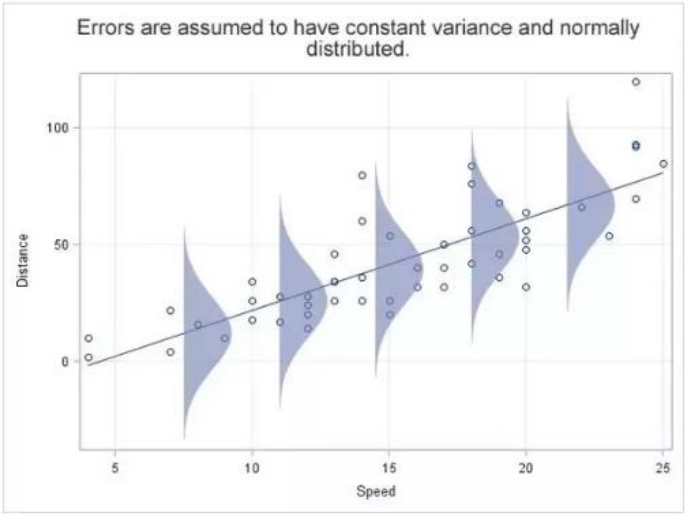
Having Machine Learning, Data Science or Python Interview? Check 👉 43 Classification Interview Questions — *Source:* glassboxmedicine.com

Q11: Why can't a *Linear Regression* be used instead of *Logistic Regression*?

Mid | Logistic Regression 25

Answer

- It is required for the independent and dependent variables to be **linear** for linear regression models, but the independent and dependent variables are **not** required to have a linear relationship in logistic functions.
- The **Linear Regression** models assume that the error terms are *normally distributed* (bell-shaped graph) whereas there are *no error terms* in **Logistic Regression** because it is assumed to follow a *Bernoulli distribution*.



- **Linear regression** has a *continuous output*. **Logistic regression** does *not* have a continuous output, rather the output is a probability between `0` and `1`. A linear regression may have an output that can go beyond `0` and `1`.

Having Machine Learning, Data Science or Python Interview? Check 👉 25 Logistic Regression Interview Questions

Source: stats.stackexchange.com

Q12: Why don't we use *Mean Squared Error* as a cost function in Logistic Regression?

Mid | Cost Function 13

Answer

- In Linear Regression, we used the **Squared Error** mechanism.
- For **Logistic Regression**, such a cost function produces a **non-convex** space with many local minimums, in which it would be very difficult to minimize the cost value and find the global minimum.

Having Machine Learning, Data Science or Python Interview? Check 👉 13 Cost Function Interview Questions

Source: towardsdatascience.com

Q13: Why is *Logistic Regression* considered a *Linear Model*?

Mid

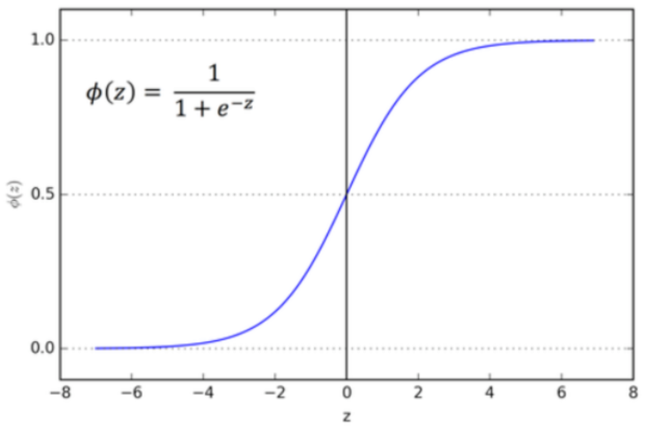
Logistic Regression 25

Answer

A model is considered linear if the **transformation of features** that is used to calculate the prediction is a **linear combination of the features**. Although Logistic Regression uses **Sigmoid function** which is a nonlinear function, the model is a generalized linear model because the outcome always depends on the sum of the **inputs and parameters**.

i.e the logit of the estimated probability response is a linear function of the predictors parameters.

$$\operatorname{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \cdots + \beta_p x_{p,i}$$



Having Machine Learning, Data Science or Python Interview? Check 👉 25 Logistic Regression Interview Questions *Source:* stats.stackexchange.com

Q14: Compare *Decision Trees* and *Logistic Regression*

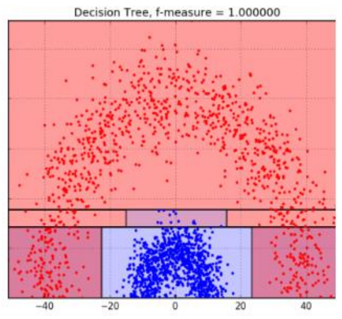
Senior

Decision Trees 49

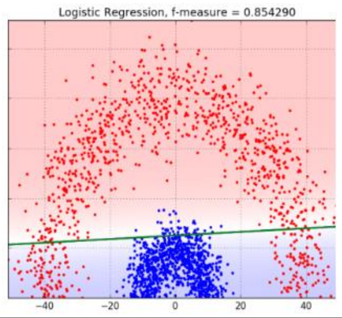
Answer

Decision Boundaries

- *Decision trees* bisect the output space into smaller and smaller regions. Even though trees fit the data better, it is prone to overfitting.

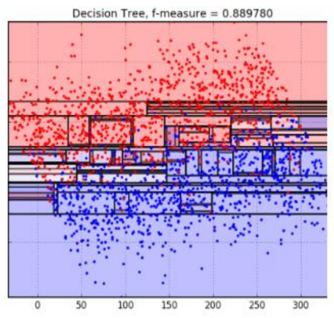


- *Logistic regression* fits a single line to divide the space exactly into two. In higher dimensions, this line would generalize to planes and hyperplanes.

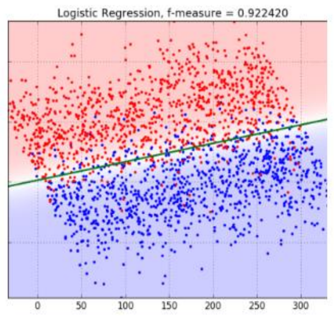


Prediction Confidence

- Each node of a *decision tree* assigns a confidence value to the entire region that it spans, leading to a patchwork appearance of confidence values.



- Prediction confidence for **logistic regression** can be computed in closed-form for any arbitrary input coordinate. This gives a more fine-grained confidence value.



Having Machine Learning, Data Science or Python Interview? Check 👉 49 Decision Trees Interview Questions *Source:* blog.bigml.com

Q15: Compare *Naive Bayes* vs with *Logistic Regression* to solve classification problems

Senior · Classification 43

Answer

- **Naive Bayes** assumes **conditional independence** among the features, which is not true in most real-life problems. So then, if for a problem these assumptions are not satisfied, the **Logistic Regression** model would be less biased and therefore, it will outperform **Naive Bayes** when given lots of training data.
- **Naive Bayes** has a time complexity of `O(log n)`, whereas the **logistic regression** is `O(n)`. So, for `n` number of features, **Naive Bayes** converges more quickly to its asymptotic estimates. Hence, it will outperform **Logistic Regression** in the case of a small training dataset.

Having Machine Learning, Data Science or Python Interview? Check 👉 43 Classification Interview Questions

Source: www.cs.cmu.edu

Q16: Explain the *Space Complexity Analysis* of *Logistic Regression*

Senior · Logistic Regression 25

Answer

The Space complexity analysis of Logistic Regression is the total amount of memory space used by the algorithm during **training** and **testing or runtime**.

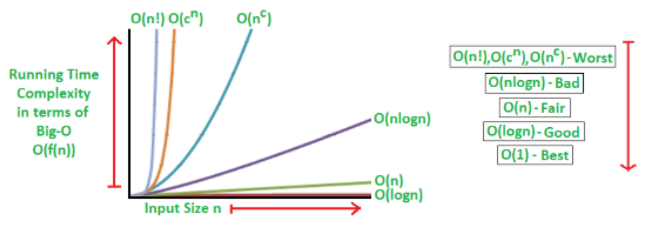
Training

During the training, four parameters are stored in the memory, and they are x, y, W, b

- To store b which is a **constant parameter**, the operation requires constant space, $O(1)$. This means that the space does not grow with the size of the data on which the algorithm is operating.
- For parameter x and y, which are both matrices of dimensions (n rows by d columns), and (n rows by 1 column) respectively. The operation to store these two matrices requires $O(nd+n)$ **space**.
- To store W, which is a vector of d columns, the operation requires $O(d)$. In summary, the space complexity analysis of Logistic Regression while training is $O(nd+n+d)$ in total.

Testing

After training the model, during the testing, W is the only parameter that needs to be stored in memory, so that other points can be classified based on the weight values. The space needed to store W is $O(d)$, because the W is a vector of d columns in dimension.



Having Machine Learning, Data Science or Python Interview? Check 👉 25 Logistic Regression Interview Questions

Source: www.analyticsvidhya.com

Q17: Explain the *Vectorized Implementation* of *Logistic Regression*?

Senior Logistic Regression 25

Answer

To make predictions on training examples, Z and the sigmoid function needs to be computed as shown below for a few training examples, in this case three.

$$\begin{aligned} z^{(1)} &= w^T x^{(1)} + b \\ a^{(1)} &= \sigma\left(z^{(1)}\right) \\ z^{(2)} &= w^T x^{(2)} + b \\ a^{(2)} &= \sigma\left(z^{(2)}\right) \\ z^{(3)} &= w^T x^{(3)} + b \\ a^{(3)} &= \sigma\left(z^{(3)}\right) \end{aligned}$$

The computation above is carried out M times, for M number of training examples.

Step 1

Define a matrix X to be the training inputs, stacked together in different columns like shown below, and this is a (N_x by M) matrix

$$X = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ x_0 & x_1 & x_2 & \ldots x_m \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Step 2

From the matrix above, we can compute

$$Z = Z^{(1)} Z^{(2)} Z^{(3)} \ldots Z^{(n)}$$

with one line of code. The matrix Z is a 1 by M matrix which is a row vector.

Step 3

Using matrix multiplication

$$Z = W^T X + b$$

we can be computed to obtain Z. This can be carried out using the matrix-vector product (np.dot) of NumPy.

Finally,

W = (N_x by 1), X is (N_x by M) and b is (1 by M), multiplication and addition gives Z (1 by M).

Q18: How can we avoid *Over-fitting* in *Logistic Regression* models?

Senior Logistic Regression 25

Answer

Regularization techniques can be used to avoid over-fitting in regression models.

Two types of regularization techniques used for logistic regression models are *Lasso Regularization, and Ridge Regularization*. Ridge and Lasso allow the regularization (*shrinking*) of coefficients. This reduces the *variance* in the model which contributes to the model not overfitting on the data. Ridge and Lasso add penalty values to the loss function as shown below:

1. **Lasso regression** adds the absolute value of the magnitude of coefficient as penalty term to the loss function as can be seen in the equation below:

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

The second term added to the loss function is a penalty term whose size depends on the *total magnitude of all the coefficients*. Lambda is a *tuning parameter* that adjusts how large a penalty there will be.

2. **Ridge** regression adds *squared magnitude* of all the coefficients as penalty term to the loss function as can be seen in the equation below:

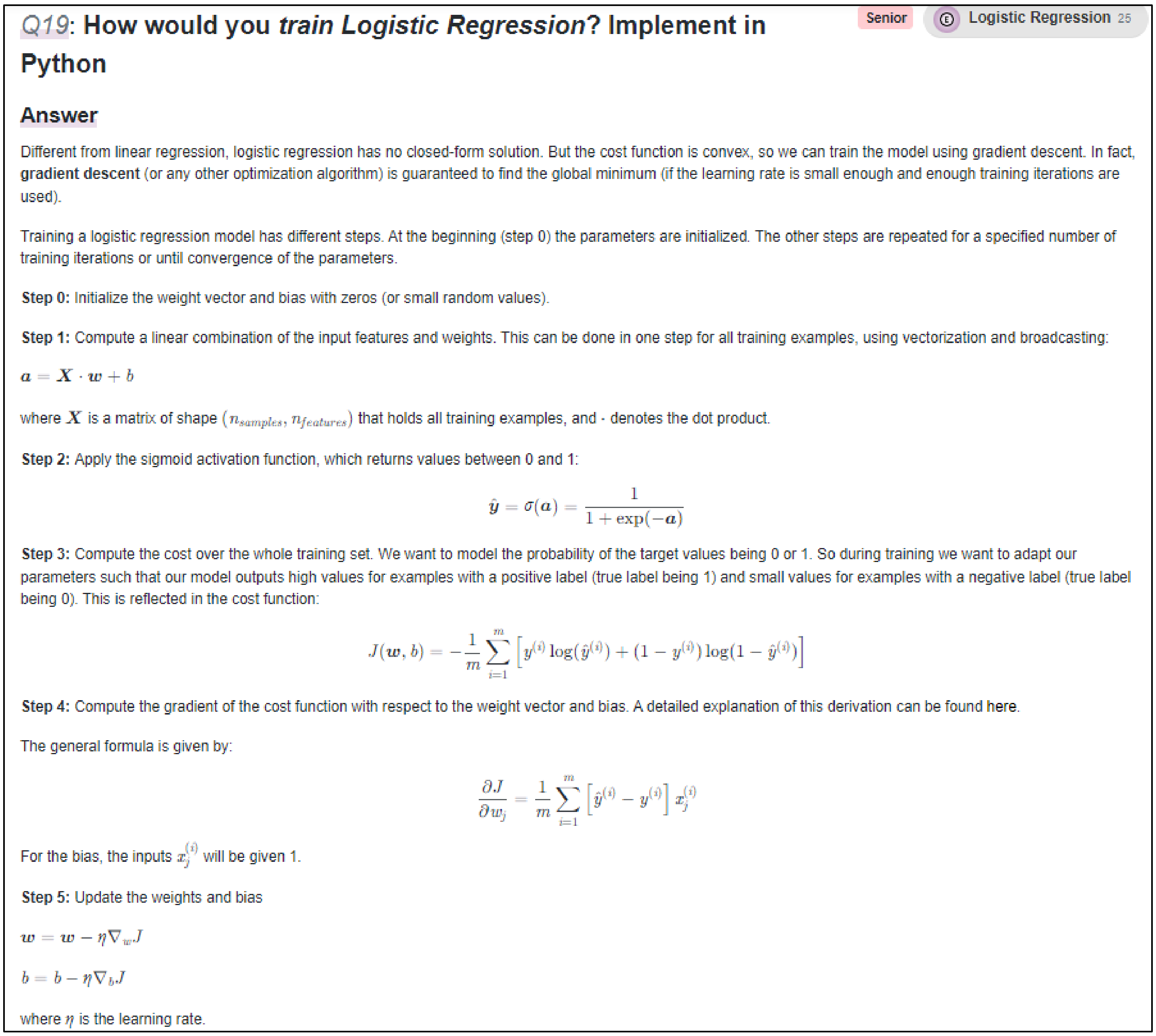
$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

The extra penalty term in this case disincentivizes including extra features. There is a balancing act between the increasing of a coefficient and the corresponding increase to the overall variance of the model.

The Ridge and Lasso act as their own feature selector were features that don't drive the predictive power of the regression to see their coefficients pushed down, while more predictive features see higher coefficients despite the added penalty.

Having Machine Learning, Data Science or Python Interview? Check 👉 25 Logistic Regression Interview Questions

Source: towardsdatascience.com



Q19: How would you *train Logistic Regression*? Implement in Python

Senior

Logistic Regression 25

Answer

Different from linear regression, logistic regression has no closed-form solution. But the cost function is convex, so we can train the model using gradient descent. In fact, **gradient descent** (or any other optimization algorithm) is guaranteed to find the global minimum (if the learning rate is small enough and enough training iterations are used).

Training a logistic regression model has different steps. At the beginning (step 0) the parameters are initialized. The other steps are repeated for a specified number of training iterations or until convergence of the parameters.

Step 0: Initialize the weight vector and bias with zeros (or small random values).

Step 1: Compute a linear combination of the input features and weights. This can be done in one step for all training examples, using vectorization and broadcasting:

$\boldsymbol{a} = \boldsymbol{X} \cdot \boldsymbol{w} + b$

where $\boldsymbol{X}$ is a matrix of shape $(n_{samples}, n_{features})$ that holds all training examples, and $\cdot$ denotes the dot product.

Step 2: Apply the sigmoid activation function, which returns values between 0 and 1:

$$\hat{\boldsymbol{y}} = \sigma(\boldsymbol{a}) = \frac{1}{1 + \exp(-\boldsymbol{a})}$$

Step 3: Compute the cost over the whole training set. We want to model the probability of the target values being 0 or 1. So during training we want to adapt our parameters such that our model outputs high values for examples with a positive label (true label being 1) and small values for examples with a negative label (true label being 0). This is reflected in the cost function:

$$J(\boldsymbol{w}, b) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

Step 4: Compute the gradient of the cost function with respect to the weight vector and bias. A detailed explanation of this derivation can be found here.

The general formula is given by:

$$\frac{\partial J}{\partial w_j} = \frac{1}{m} \sum_{i=1}^{m} \left[\hat{y}^{(i)} - y^{(i)} \right] x_j^{(i)}$$

For the bias, the inputs $x_j^{(i)}$ will be given 1.

Step 5: Update the weights and bias

$\boldsymbol{w} = \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} J$

$b = b - \eta \nabla_{b} J$

where η is the learning rate.

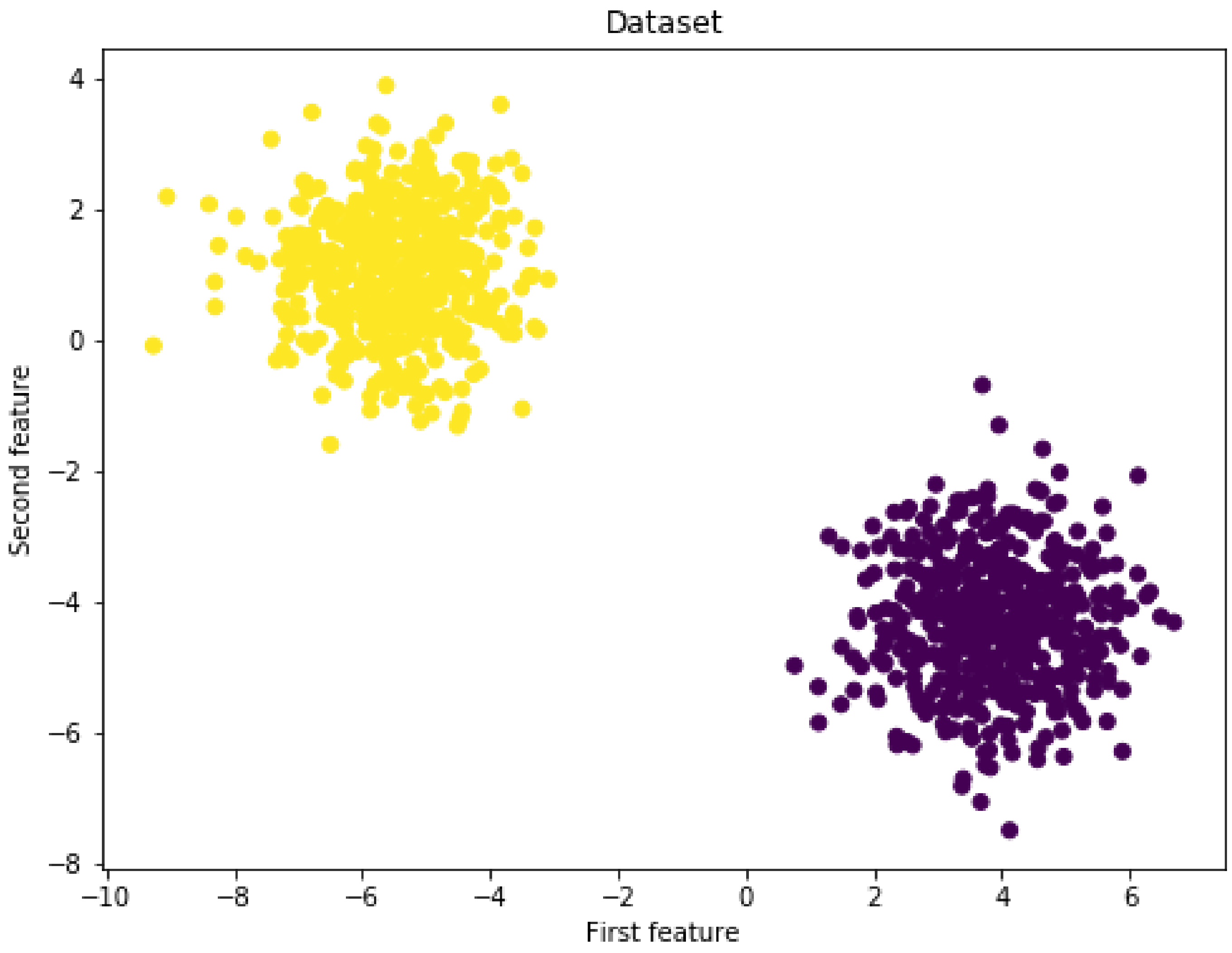
Implementation:
In Python PY

Required Libraries and Modules

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
np.random.seed(123)
```

Dataset

```
# We will perform logistic regression using a simple toy dataset of two classes
X, y_true = make_blobs(n_samples= 1000, centers=2)

fig = plt.figure(figsize=(8,6))
plt.scatter(X[:,0], X[:,1], c=y_true)
plt.title("Dataset")
plt.xlabel("First feature")
plt.ylabel("Second feature")
plt.show()
```



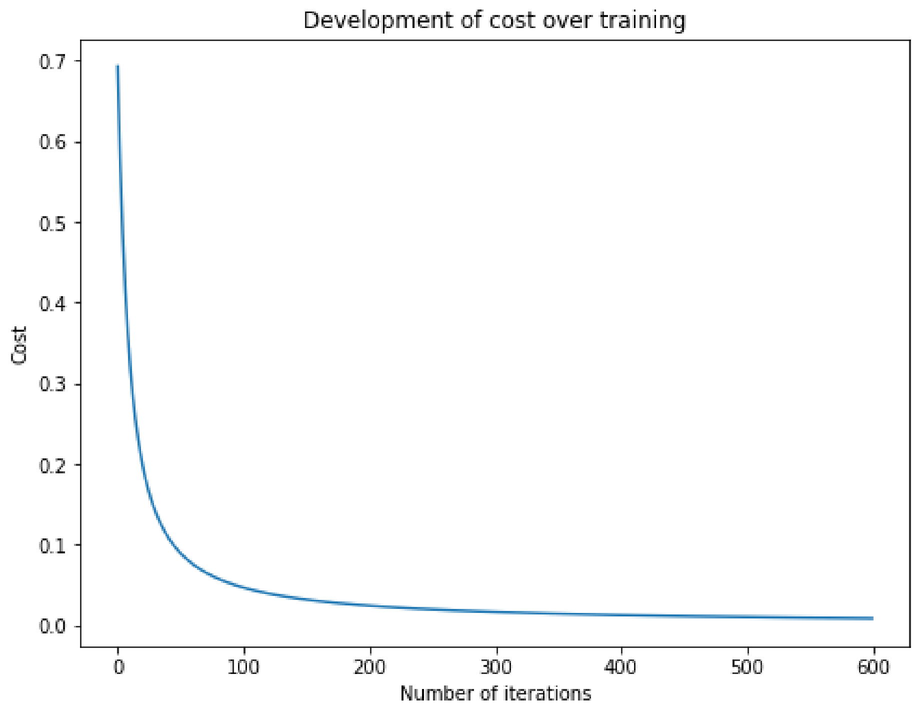
```
# Reshape targets to get column vector with shape (n_samples, 1)
y_true = y_true[:, np.newaxis]
# Split the data into a training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y_true)

print(f'Shape X_train: {X_train.shape}')
print(f'Shape y_train: {y_train.shape}')
print(f'Shape X_test: {X_test.shape}')
print(f'Shape y_test: {y_test.shape}')
Shape X_train: (750, 2)
Shape y_train: (750, 1)
Shape X_test: (250, 2)
Shape y_test: (250, 1)
```

Logistic regression class

```
class LogisticRegression:

    def __init__(self):
        pass

    def sigmoid(self, a):
        return 1 / (1 + np.exp(-a))

    def train(self, X, y_true, n_iters, learning_rate):
        """
        Trains the logistic regression model on given data X and targets y
        """
        # Step 0: Initialize the parameters
        n_samples, n_features = X.shape
        self.weights = np.zeros((n_features, 1))
        self.bias = 0
        costs = []

        for i in range(n_iters):
            # Step 1 and 2: Compute a linear combination of the input features and weights,
            # apply the sigmoid activation function
            y_predict = self.sigmoid(np.dot(X, self.weights) + self.bias)

            # Step 3: Compute the cost over the whole training set.
            cost = (- 1 / n_samples) * np.sum(y_true * np.log(y_predict) + (1 - y_true) * (np.log(1
- y_predict)))

            # Step 4: Compute the gradients
            dw = (1 / n_samples) * np.dot(X.T, (y_predict - y_true))
            db = (1 / n_samples) * np.sum(y_predict - y_true)

            # Step 5: Update the parameters
            self.weights = self.weights - learning_rate * dw
            self.bias = self.bias - learning_rate * db

            costs.append(cost)
            if i % 100 == 0:
                print(f"Cost after iteration {i}: {cost}")

        return self.weights, self.bias, costs

    def predict(self, X):
        """
        Predicts binary labels for a set of examples X.
        """
        y_predict = self.sigmoid(np.dot(X, self.weights) + self.bias)
        y_predict_labels = [1 if elem > 0.5 else 0 for elem in y_predict]

        return np.array(y_predict_labels)[:, np.newaxis]
```

Initializing and training the mode

```
regressor = LogisticRegression()
w_trained, b_trained, costs = regressor.train(X_train, y_train, n_iters=600, learning_rate=0.009)

fig = plt.figure(figsize=(8,6))
plt.plot(np.arange(600), costs)
plt.title("Development of cost over training")
plt.xlabel("Number of iterations")
plt.ylabel("Cost")
plt.show()
Cost after iteration 0: 0.6931471805599453
Cost after iteration 100: 0.046514002935609956
Cost after iteration 200: 0.02405337743999163
Cost after iteration 300: 0.016354408151412207
Cost after iteration 400: 0.012445770521974634
```



Testing the model

```
y_p_train = regressor.predict(X_train)
y_p_test = regressor.predict(X_test)

print(f"train accuracy: {100 - np.mean(np.abs(y_p_train - y_train)) * 100}%")
print(f"test accuracy: {100 - np.mean(np.abs(y_p_test - y_test))}%")
```

Source: [github.com]

Q20: Imagine that you know there are *outliers* in your data, would you use *Logistic Regression*?

Senior   Anomaly Detection 47

Answer

- **Logistic Regression:** Logistic Regression is an algorithm that will be highly *influenced* by outliers. If we have outliers in our dataset, the decision boundary could be shifted, which will lead to incorrect inferences. So using Logistic Regression wouldn't be optimal if we have outliers.
- **Tree-based model:** Algorithms like **decision trees** or **random forests** are more **robust to outliers**. This is because unlike a linear model like Logistic Regression, a tree-based model works by splitting the data into groups (repeatedly) according to whether a data point is above or below a selected threshold value on a selected feature variable. Thus, a data point with a much higher value than the rest wouldn't influence the decision of a tree-based model at all.

Having Machine Learning, Data Science or Python Interview? Check 👉 47 Anomaly Detection Interview Questions   *Source:* www.elderresearch.com

Q21: Name some advantages of using *Support Vector Machines* vs *Logistic Regression* for classification

Senior SVM 56

Answer

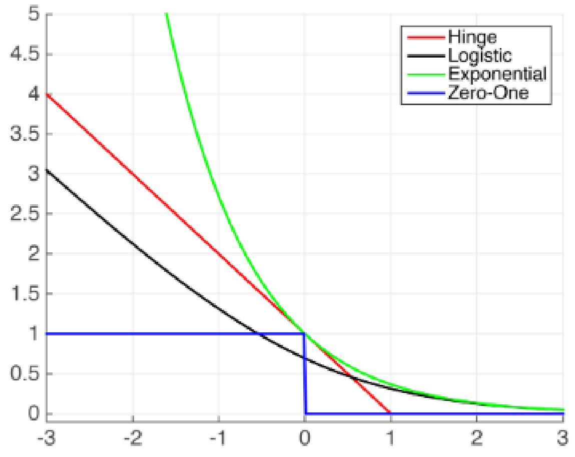
The goal of ***Support vector machine (SVM)*** and ***Logistic regression (LR)*** is to find a hyperplane that distinctly classifies the data points. The main difference between these two algorithms is in their *loss functions*: *SVM* minimizes ***hinge loss*** while *logistic regression* minimizes ***logistic loss***.

- **Hinge-Loss** - When used for Standard SVM, the loss function denotes the size of the margin between linear separator and its closest points in either class. Only differentiable everywhere with p = 2

  $\max\left[1 - h_{\mathrm{w}}(\mathbf{x}_i)y_i, 0\right]^p$

- **Log-loss** - One of the most popular loss functions in Machine Learning, since its outputs are well-calibrated probabilities.

  $\log(1 + e^{-h_{\mathrm{w}}(\mathbf{x}_i)y_i})$



Things to remember:

- The *logistic loss* diverges faster than *hinge loss*. So, in general, *SVM* will be more robust to outliers than *LR*.
- The *logistic loss* does not go to zero even if the point is classified sufficiently confidently, so this might lead to minor degradation in accuracy. *Hinger loss* can take the value of `0`, so with *SVM* we don't have this problem.
- *LR* produces probabilistic values while *SVM* produces `1` or `0`. So in a few words, *LR* makes no absolute prediction and it does not assume data is enough to give a final decision.
- *SVM* tries to maximize the margin between the closest support vectors while *LR* maximizes a class probability. Thus, *SVM* finds a solution that is as *fair as possible* for the two categories while *LR* has not this property.
- *SVM* use kernel tricks and transform datasets into rich features that can classify no *linearly separable* data while *LR* does not perform well on this kind of problems.

Q22: When would you use *SVM* vs *Logistic regression*?

Senior SVM 56

Answer

Generally, the first approach is to try with ***logistic regression*** to see how the model performs. If we found that the data can't be linearly separable or the model does not have a good performance, then we can try using ***SVM***.

We can also considerer the following rule of thumb: Given `n` number of *features* and `m` number of *training* examples:

- If `n` is less than `10,000` and `m` is less than `1,000` we can use either *logistic regression* or *SVM* with a *linear kernel*.
- If `n` is less than `1,000` and `m` is less than `10,000` we can use *SVM* with a *Gaussian*, *polynomial*, or another kernel.
- If `n` is less than `1,000`, and `m` is greater than `50,000` we first add more features manually and then can use *logistic regression* or *SVM* with a *linear kernel*.

Having Machine Learning, Data Science or Python Interview? Check 👉 56 SVM Interview Questions

Source: medium.com