Logistic Regression

Linear Regression → cont. values of y

↓

regression algorithm

↓

regression problem (stock market analysis)

classification : y belongs to one of certain discrete classes.

↓

y → spam (0)

y → not spam (1)

↓

y → Cat (0)

y → Horse (1)

y → Dog (2)

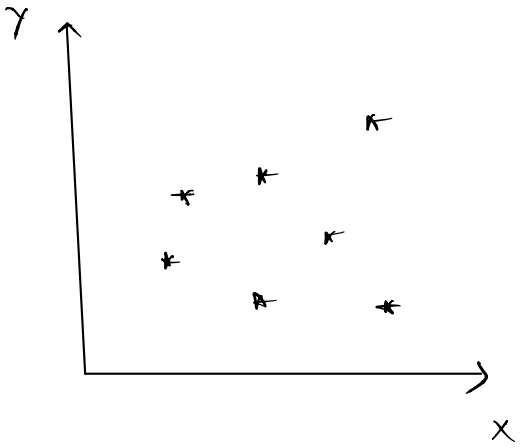
*) dealing with binary classification in this lecture

↓

only 2 classes to choose from

↓

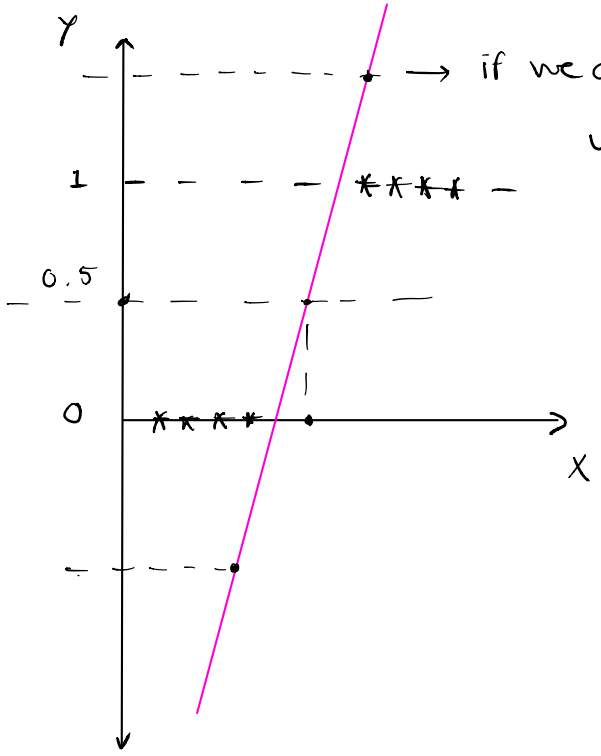
One way of doing this is by extending the already learnt linear regression algorithm.



→ Continuous in nature

↓

Distributed through the domain



→ if we apply linear regression to it, it will give the corresponding cont. that need to be tuned to get categorical output.

↓

One way of doing that is to bring a cutoff above which if predicted, y belongs to class 1, otherwise class 0.

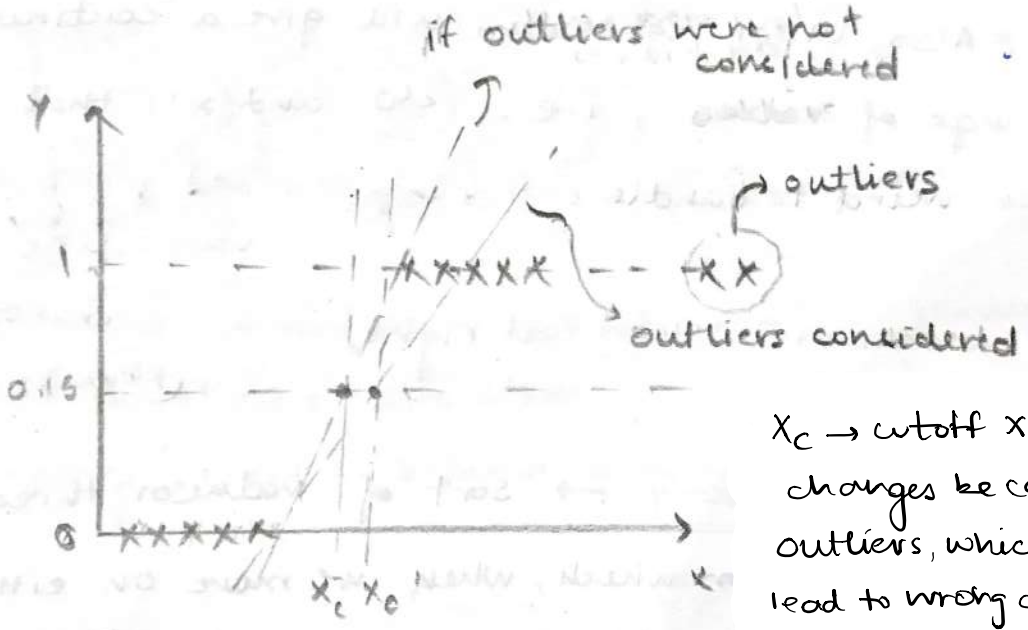
↓

$y_{pre} > 0.5 \rightarrow 1$

$y_{pre} < 0.5 \rightarrow 0$

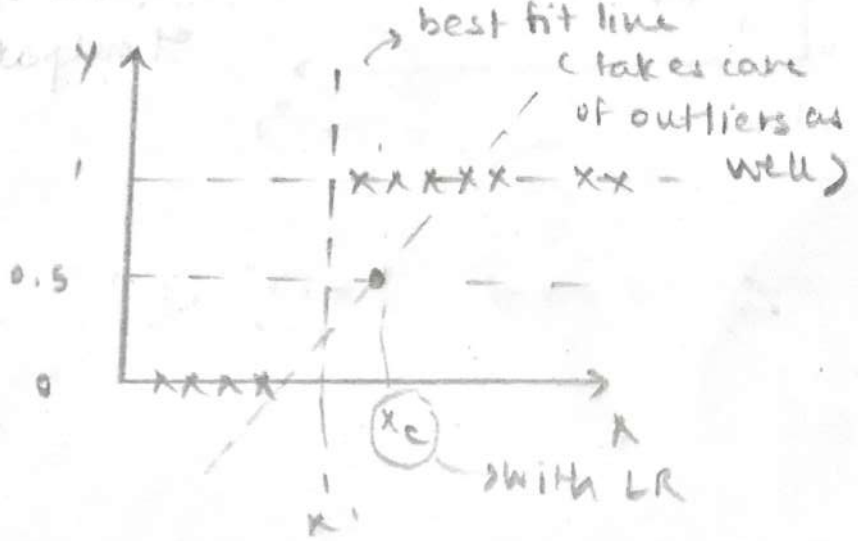
} example

One problem with this approach is the effect of outliers brought in by the line given by linear regression, which affects x_c and hence the accuracies of our classifications.



$X_c \rightarrow$ cutoff x

changes because of outliers, which may lead to wrong classification



→ linear regression would pick something which would actually be distorted due to the presence of outliers.

(92) * Also, linear regression will give a continous range of values, i.e. (<<0) and (>>1), that will be weird to handle.

↓ (does not feel right)

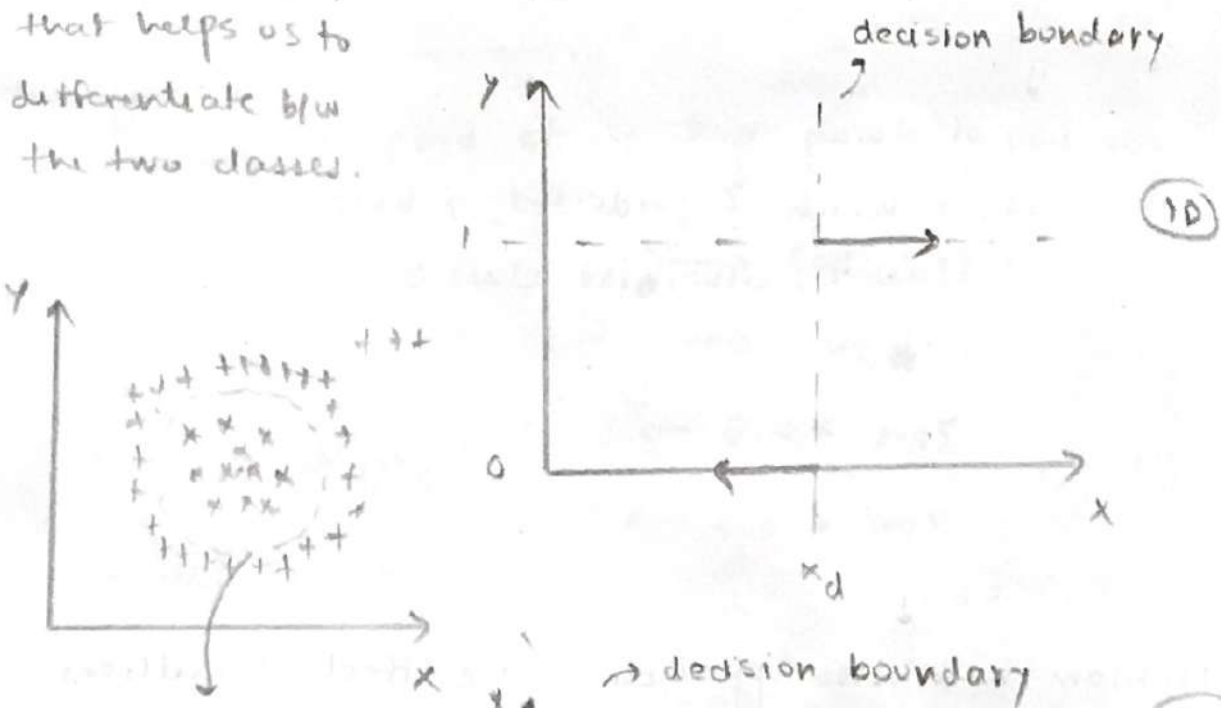
↓ values like −300 or 100 might not make any sense

* Decision boundary → sort of value or threshold from which, when we move on either direction, we reach a class.

or

crossing that threshold classifies to a class

the boundary that helps us to differentiate b/w the two classes.



decision boundary

↓ non linear decision boundary

↓ output is represented by different symbols of output.

* logistic Regression

↓

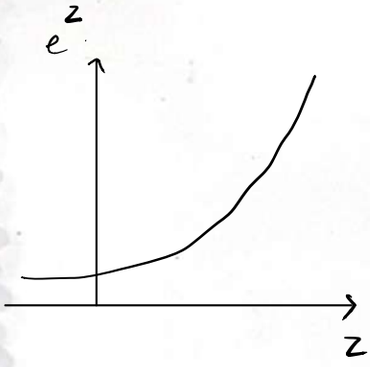
classification algorithm

↓

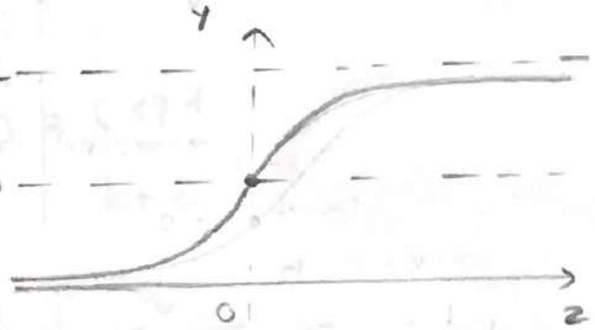
(i) extreme values in a dataset (outliers) should not hurt our classification

(ii) values should be between 0 and 1

(•) sigmoid function:



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



$z \to -\infty$, $e^{-z} \to \infty$

$s(z) \to 0$

$z \to \infty \to e^{-z} \to 0$

$s(z) \to 1$

$z \to 0$, $e^{-0} \to 1$

$s(z) = 0.5$

↓

Sigmoid never reaches 0 or 1, rather approaches them, as they act like two horizontal asymptotes.

↓

nice curve with symmetry on both sides of origin, and stays between 0 and 1

↓

exploit sigmoid for logistic regression ← brings all values in the 0, 1 range

↓

that can be thought of as probability as well.

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

$g(x) = mx + c$ → linear regression line

or

$m_1x_1 + m_2x_2 + m_3x_3 + c$ } in case of 3 features }

$g(x) = m^T \cdot x$

$x \rightarrow (n+1) \times 1$

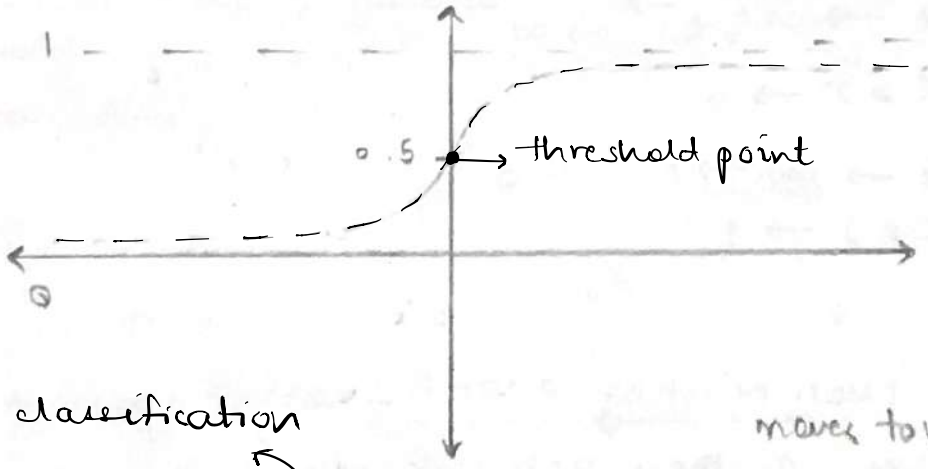
$m^T \rightarrow (n+1) \times 1$

$$h(x) = \frac{1}{1 + e^{-m^T \cdot x}}$$

The main point of using sigmoid is the fact that it handles the value issue from linear regression

↓

And brings all values in (0,1) range.



used for classification

$h(x) \geq 0.5 \rightarrow y_{pre} \rightarrow 1$

$h(x) < 0.5$

↓

$y_{pre} \rightarrow 0$

moves towards 0 or 1 very fast

↓

so whenever we give it an input, it is very likely, it will land either near 0 or 1

$$\frac{1}{1 + e^{-m^T x}} \quad (= z) \quad > 0.5$$

$z \geq 0 \rightarrow S(z) \geq 0.5$

$z < 0 \rightarrow S(z) < 0.5$

$m^T x > 0 \rightarrow h(x) > 0.5$

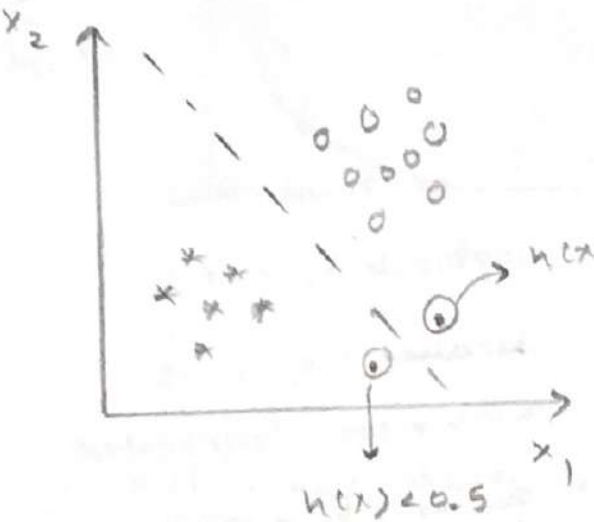
$y_{pre} \rightarrow 1$

$m^T x < 0 \rightarrow h(x) < 0.5$

$y_{pre} \rightarrow 0$

$$s(z) = \frac{1}{1+e^{-z}}, \quad s(z) > 0.5 \text{ if } z > 0$$

$h(x) = s(m^T \cdot x) = \frac{1}{1+e^{-m^T \cdot x}}$



$\downarrow$

$h(x) > 0.5$ if $m^T \cdot x > 0$.

$\downarrow$

Our prediction function is non-linear but our decision boundary is linear

$$h(x) = \frac{1}{1+e^{-m^T \cdot x}} = s(m^T x)$$

↓

Hypothesis function

↓

The task now becomes to find the optimal values of m, such that error is minimized (finding optimal parameters)

If we choose squared error as was the case with linear regression, then.

$$E(h(x), y) = \sum_{i=1}^{N} (y^i - h(x^i))^2$$

↓

basically dependent on output and actual ys.

↓

can't use this approach to sigmoid, since now the function is comparatively complicated and might have multiple local minimas

↓

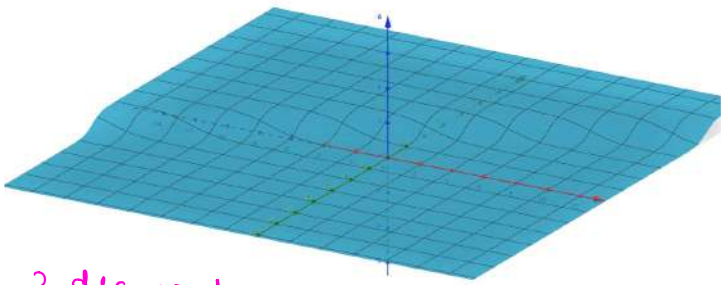
Did not have this problem with linear regression because $m^T \cdot x$ is a linear function.

↓

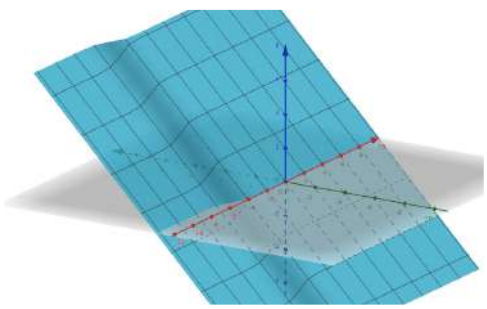
∴ error function was of 2nd degree, hence one minima, and also easy to differentiate.

↓

Hence wherever we start, we reach the best possible answer



2 degree hypothesis fn.



error fn. with one degree sigmoid

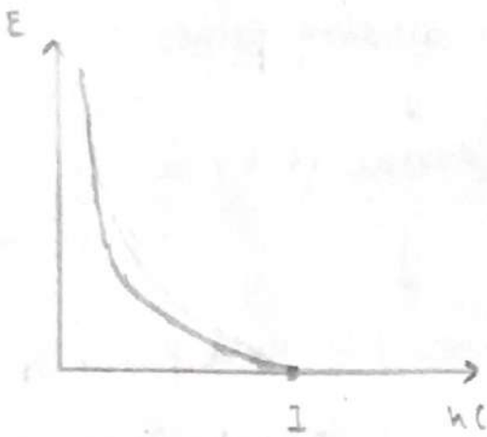
$E(h(x^i), y^i) \longrightarrow -\log(h(x^i)) \quad y = 1$

$-\log(1 - h(x^i)) \quad y = 0.$

↓

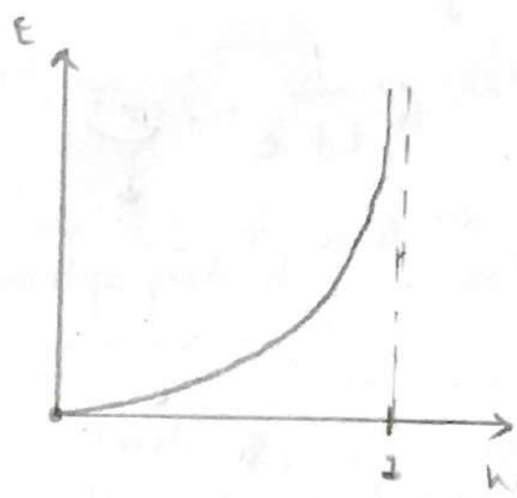
when dealing with binary classification

$h(x) \to 0$
$E \to \infty$
$h(x) \to 1$
$E \to 0$



if $y = 1$,
$h(x^i) = 1$
↓
error should be zero
↓
our error function does that

* if actual answer is 1 and we predict 0, so we are penalising a lot (because exponential inc.)
↓
(grows very fast)



$-\log(1 - h(x^i)) \quad y = 0$

$h(x) \to 0$
$E(x) \to 0$

$h(x) \to 1$
$E(x) \to \infty$

So here we are saying that if the true value is 0 or 1, the predicted should be close to the corresponding value, else heavy cost will be incurred.

combining, to form only one eqn for error function. we get

$$E(h(x^i), y^i) = \sum_{i=1}^{m} -y^i \log(h(x^i)) - (1-y^i)\log(1-h(x^i))$$

↓

for all data points

↓

we divide it by (m)

↓

no. of data points

↓

to get error per data point (avg error)

$$= \frac{1}{m}\sum_{i=1}^{m} -y^i \log(h(x^i)) - (1-y^i)\log(1-h(x^i))$$

↓

happens to be convex, with single minima.

↓

where $h(x^i) = \frac{1}{1+e^{-m^T \cdot x}}$

↓

finding appropriate values of m, using gradient descent

↓

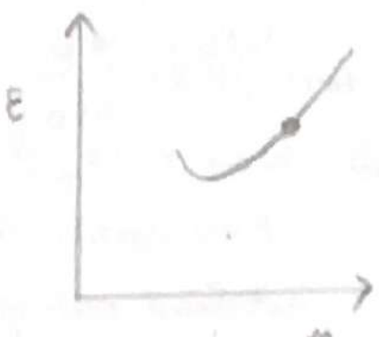
minimizing the error fn.

↓

starting with random values for the parameters

↓

find $\frac{\partial E}{\partial m_j}$ and tuning m_j



$$m_j' = m_j - \alpha\left(\frac{\partial E}{\partial m_j}\right)$$

→ we modify E^i (error for one sample) to ease calculation of gradient

$$E^i = -y^i \log(h(x^i)) - (1-y^i)\log(1-h(x^i))$$

$$\log(h(x^i)) = \log\left(\frac{1}{1+e^{-m^T x^i}}\right)$$

$$= \log 1 - \log(1+e^{-m^T x^i})$$

$$= -\log(1+e^{-m^T x^i})$$

$$\log(1-h(x^i)) = \log\left(1 - \frac{1}{1+e^{-m^T x^i}}\right)$$

$$= \log\left(\frac{e^{-m^T x^i}}{1+e^{-m^T x^i}}\right)$$

$$= \log(e^{-m^T x^i}) - \log(1+e^{-m^T x^i})$$

$$E^i = -y^i(-\log(1+e^{-m^T x^i})) - (1-y^i)(\log(e^{-m^T x^i}) - \log(1+e^{-m^T x^i}))$$

$$= y^i \log(1+e^{-m^T x^i}) - \log e^{-m^T x^i} + \log(1+e^{-m^T x^i}) + y^i \log(e^{-m^T x^i}) - y^i \log(1+e^{-m^T x^i})$$

$$E^i = m^T x^i - y^i m^T x^i + \log\left(1 + e^{-m^T x^i}\right)$$

$$E^i = \log\left(e^{m^T x_i}\right) - y^i m^T x^i + \log\left(1 + e^{-m^T x_i}\right)$$

$$E^i = \log\left(1 + e^{m^T x_i}\right) - y^i m^T x^i$$

$$\frac{\partial E^i}{\partial m_J} = -y^i x_J^i + \frac{e^{m^T x_i}\, x_J^i}{1 + e^{m^T x_i}}$$

$$\frac{\partial E^i}{\partial m_J} = -y^i x_J^i + \frac{x_J^i}{e^{-m^T x_J^i} + 1}$$

$$\frac{\partial E^i}{\partial m_J} = -y^i x_J^i + x_J^i\, h(x^i)$$

$$\frac{\partial E^i}{\partial m_J} = x_J^i\left(-y^i + h(x^i)\right)$$

$$\frac{\partial E(m)}{\partial m_J} = -\frac{1}{N} \sum_{i=1}^{N} \left(y^i - h(x^i)\right) x_J^i$$

(102) * multiclass logistic regression

using logistic regression to reach an hypothesis function which gives us a value b/w 0 and 1 for any x^i.

↓

higher the value of $h(x^i)$, more the probability that the value will be 1

↓

since the output is b/w 0 and 1, we sometimes use it as probability

↓

$h(x) \rightarrow P(y=1)$ → higher the probability, more are the chances we are taking the correct decision

↓

0.0001

↓

pretty much sure that the class is 0

↓

similarly $P(y=0)$, $1-h(x)$

↓

handling this approach from binary to multi class problems

↓

tagging folders or mails etc

↓

not possible to integrate with the method derived for binary classification

↓

$>0.5 \rightarrow 1$

$<0.5 \rightarrow 0$

approach 1 : one vs rest

$y \rightarrow$ cat, Rat, Dog

↓

building 3 logistic regression models

↓

model 1 $\rightarrow$ $y =$ [cat, not a cat] $\rightarrow$ using sigmoid to train hypothesis for this modified data, and accordingly getting decision boundaries using gradient descent

↓ ↓

1 0

↙

In training data, whenever we have cat label, it becomes a 1, and other class labels are treated as 0.

↓

Model is created for each class and all of them will have their separate hypothesis function.

↓

After training, when given a new data point, each hypothesis function will give its value as to if what is the probability of the data point belonging to the classes through each model. One with maximum probability for a class is then chosen as the predicted label.

*) since the hypothesis function operates on different optimal values, different parameters, the sum of probabilities is not meant to be equal to 1.

↓

One way to handle this is to take weighted average of all probabilities, and then that should lead to the sum being equal to one.

approach 2 : multinomial logistic regression

↓

train 1 model with lot more parameters

↓

K classes (numbered 1 to K)

↓

$$P(y = J) = \frac{e^{m_J X}}{\sum_{i=1}^{k} e^{m_i X}}$$ → softmax function

↓

So for each class we will have a corresponding vector m and their training will take place accordingly.

↓

For a new data point, we find the softmax value for each class, and the one with maximum is selected.

↓

learning $n * k$ parameters together for 1 model.

↓

whereas in one vs rest approach, we train k models, all independent, each with n parameters.

*) finding complex boundaries and regularization

The decision boundary for binary classification using sigmoid have been linear. As with linear regression, to get more complex boundaries, we can make use of dummy features.

↓

might lead to overfitting.

↓

Regularization is a technique used to avoid overfitting; wherein the cost function is modified to have a direct dependency on summation of parameters.

↓

$$\text{cost} = \left(\Sigma -y^i \log(h(x^i)) - (1-y^i)\log(1-h(x^i))\right) + (\lambda)\left(\Sigma (m_1)^d\right)$$

Regularization factor

↓

if $\lambda = 0$, no added cost, leading to overfitting case.

↓

if $\lambda = 1$, very high importance to parameter values, leading to underfitting → another hyperparameter to tune.

↓

associates a cost for including a certain feature.

↓

If it reduces the first one a lot, then the higher value on regularization can be justified.

$\sum m_i \rightarrow l_1$ regularization

$\sum (m_i)^2 \rightarrow l_2$ regularization (generally used)

↓

No regularization on an intercept, adds no dependency on any feature.