COMP/EECE 7/8740 Neural Networks

Topics:

Logistic regression

- Logistic regression
- Underfitting or overfitting problems and
- Solutions

Md Zahangir Alom
Department of Computer Science
University of Memphis, TN

What is Logistic Regression ?

- This type of statistical model is often used for **classification and predictive analytics**.
- Logistic regression **estimates the probability of an event occurring**, such as voted or didn't vote, based on a given dataset of independent variables.
  - Since the outcome is a probability, **the dependent variable is bounded between 0 and 1**.
- Similar to linear regression, logistic regression is also used to estimate the relationship between a dependent variable and one or more independent variables, but it is used to **make a prediction about a categorical variable versus a continuous one**.
  - A categorical variable can be true or false, yes or no, 1 or 0, et cetera.

Linear versus Logistic Regression

- The unit of measure also differs from linear regression as it produces a probability, **but the logit function transforms the S-curve into straight line**
- Both models are used in regression analysis to make predictions about future outcomes, **linear regression is typically easier to understand**
- Linear regression also does not require as large of a sample size as **logistic regression needs an adequate sample** to represent values across all the response categories.
- Without a larger, representative samples, the model may not have sufficient statistical power to detect a significant effect.

Types of logistic regression

- There are **three types of logistic regression models**, which are defined based on categorical response.
  - **Binary logistic regression:** In this approach, the response or dependent variable is dichotomous in nature—i.e. it has only two possible outcomes (e.g. 0 or 1). Some popular examples of its use include predicting if an e-mail is spam or not spam or if a tumor is malignant or not malignant (most popular type).
  - **Multinomial logistic regression:** In this type of logistic regression model, the dependent variable has three or more possible outcomes; however, these values have no specified order. For example, a multinomial logistic regression model can help the studio to determine the strength of influence a person's age, gender, and dating status may have on the type of film that they prefer.
  - **Ordinal logistic regression:** This type of logistic regression model is leveraged when the response variable has three or more possible outcome, but in this case, these values do have a defined order. Examples of ordinal responses include grading scales from A to F or rating scales from 1 to 5.

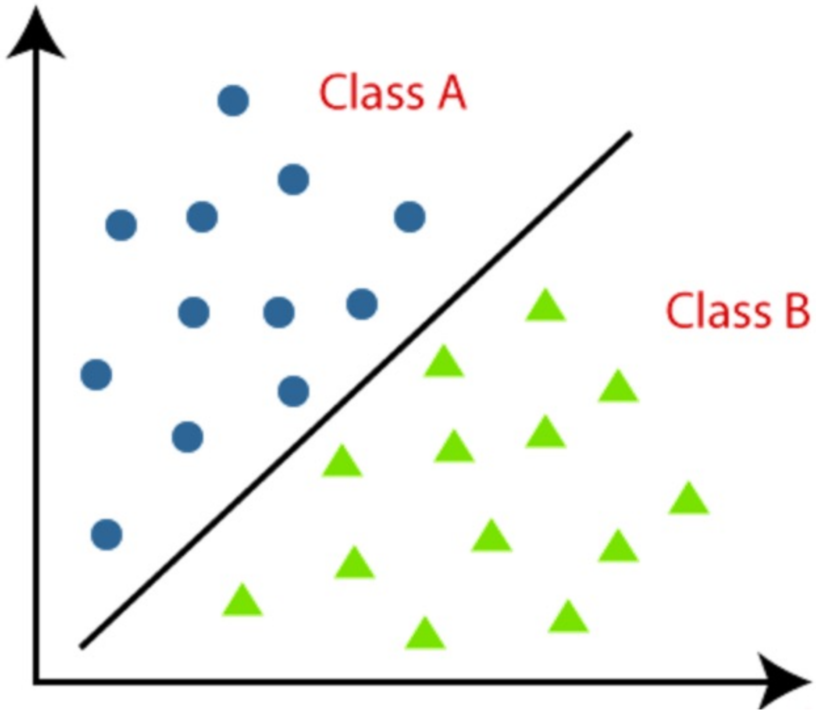
Logistic regression in machine learning

- In Machine Learning (ML), logistic regression belongs to the family of supervised learning models.
- The **negative log likelihood used as the loss function**, using the process of gradient descent to find the global maximum.
- Logistic regression can also **be prone to overfitting**, particularly when there is a high number of predictor variables within the model.
- **Regularization is typically used to penalize parameters large coefficients** when the model suffers from high dimensionality.

CLASSIFICATION

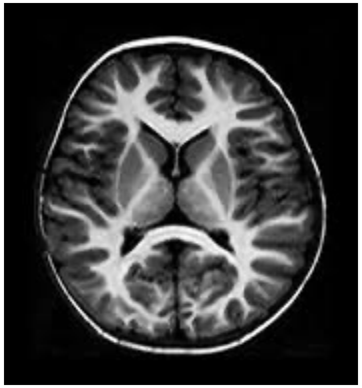
Classification

- Target value – 0 or 1 – 2 Classes

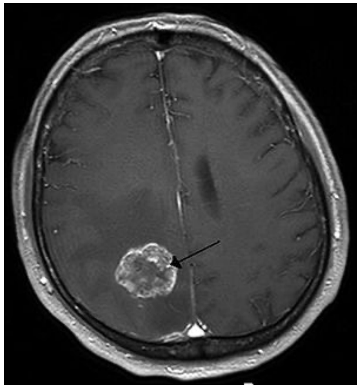


Some Examples

- Email: Spam / Not Spam
- Online Transaction: Fraudulent (Yes/No)
- Tumor: No/Yes



No



Yes

Target Variable ‘Y’

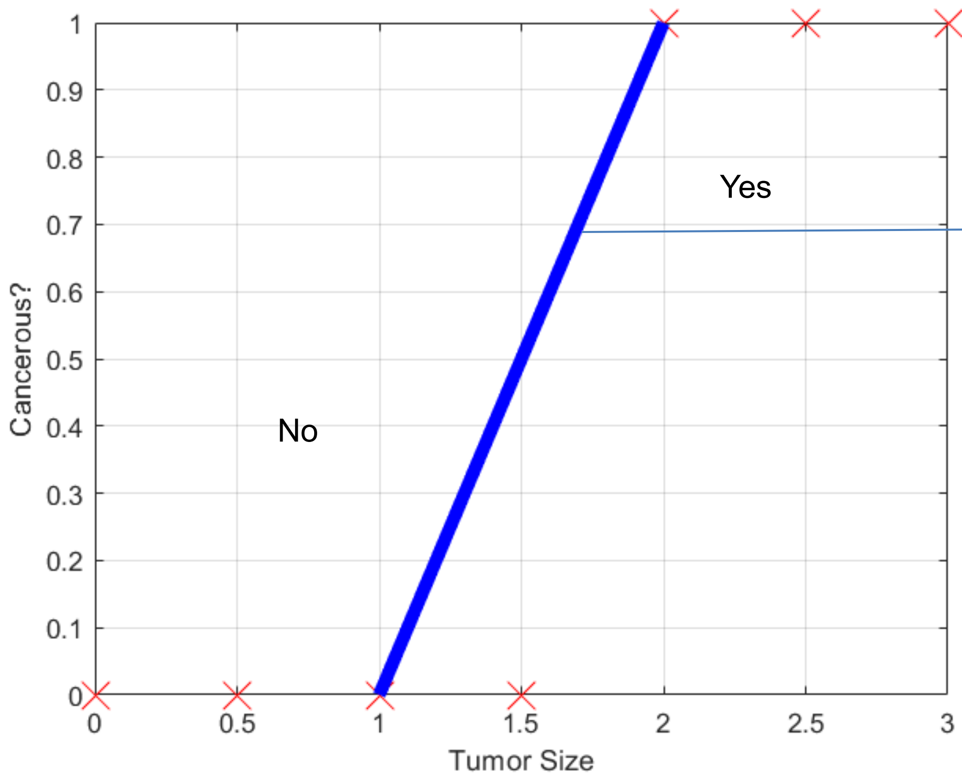
- For 2-class problem
  - $y \in \{0,1\}$

0 : Negative Class : No Tumor

1 : Positive Class : Tumor

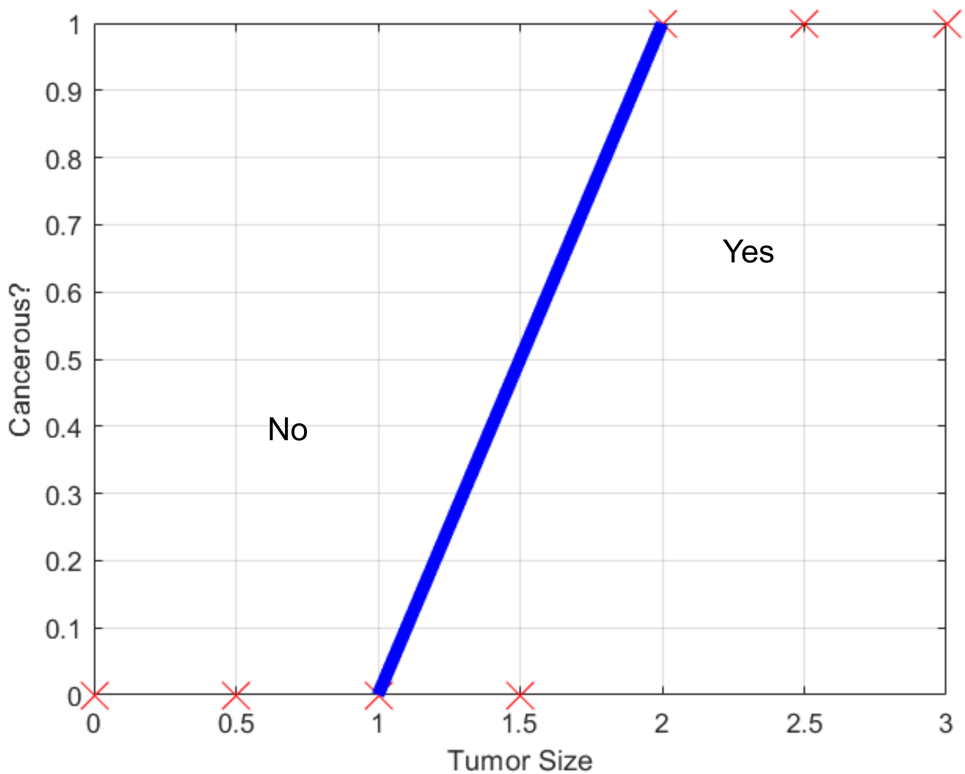
- For 4-class problem
  - $y \in \{0,1,2,3\}$

Let’s start with 2 class variable



$h_\theta(x) = \theta_0 + \theta_1 x$?

Conversion of Regression to Classification?

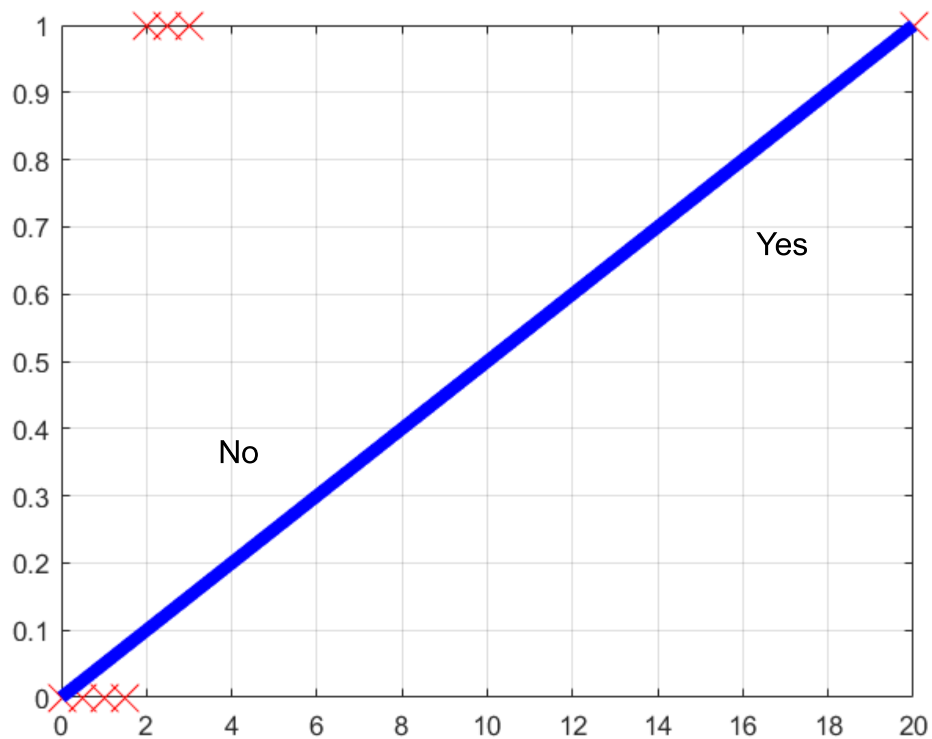


$h_\theta(x) = x - 1$

if $h_\theta(x) \geq 0.5$ → Predict y ($\hat{y}$)=1

$h_\theta(x) < 0.5$ → Predict y ($\hat{y}$)=0

Let's add one more training example?



Poor performance in this scenario

"Adding one example has reduced the performance considerably"

"Other funny things can happen too"

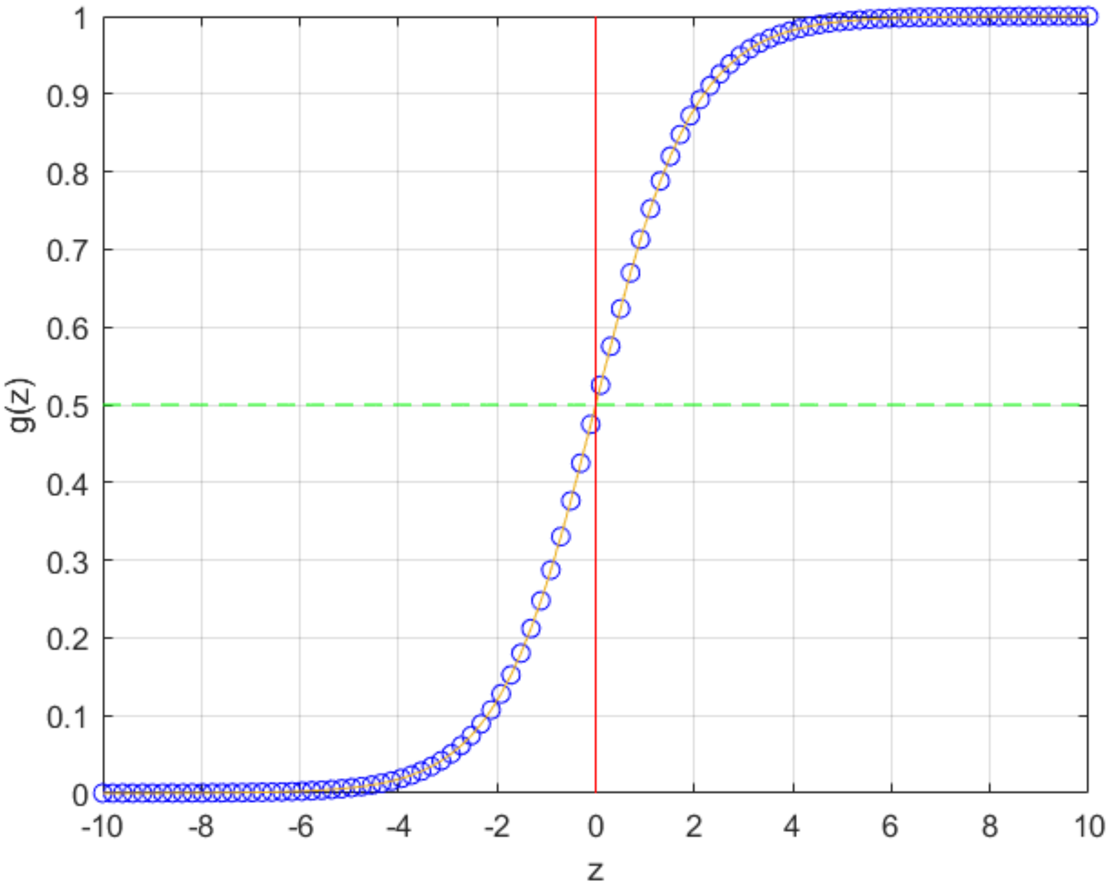
'h' should be technically 0 or 1 but there is a possibility that 'h' can be a greater value as $\theta_0 + \theta_1 x$

Logistic regression

- Don't be confused by the name: It's designed for **classification**
- Classification Algorithm: $0 \leq h_\theta(x) \leq 1$
- Hypothesis:
  - $h_\theta(x) = \theta x^T$ - Linear Regression
  - $h_\theta(x) = g(\theta x^T)$ - Logistic Regression
  - g ?

We have to modify the linear regression equation slightly to find the solution for logistic regression

Sigmoid function



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

Interpretation of Hypothesis

- $h_\theta(x)$: estimated probability that $y = 1$ for the input 'x'
- Let's say, $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$ = $\begin{bmatrix} 1 \\ Tumor\ Size \end{bmatrix}$ and let's say our algorithm predicts

  $h_\theta(x)$ = 0.7, then 70% of chance of tumor being cancerous

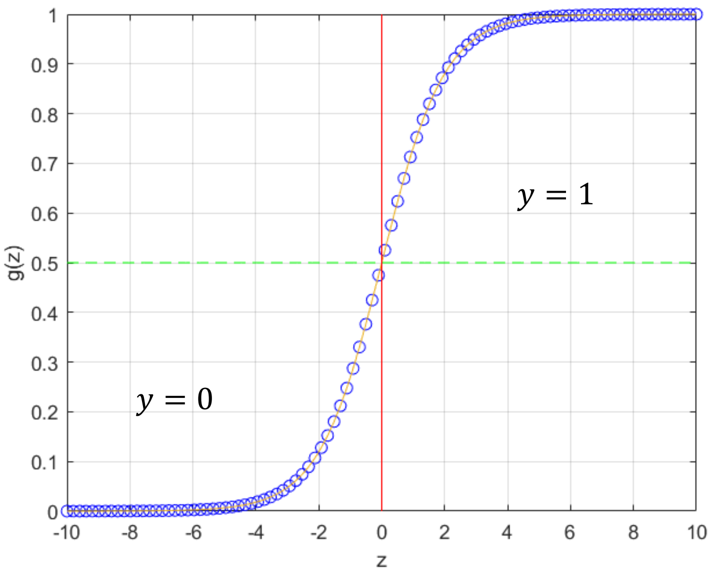
- $h_\theta(x) = \mathrm{P}(\mathrm{y} = 1 \mid \mathrm{x}\,; \theta)$ → Probability that y=1 given 'x' estimated by θ
- $\mathrm{P}(\mathrm{y} = 0 \mid \mathrm{x}\,; \theta)$ = $1 - \mathrm{P}(\mathrm{y} = 1 \mid \mathrm{x}\,; \theta)$

Decision Boundary

- $h_\theta(x) = g(\theta x^T) = \mathrm{P}(\mathrm{y} = 1 \mid \mathrm{x}\,; \theta)$

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

- $\hat{y} = 1$ if $h_\theta(x) \geq 0.5$
- $\hat{y} = 0$ if $h_\theta(x) < 0.5$

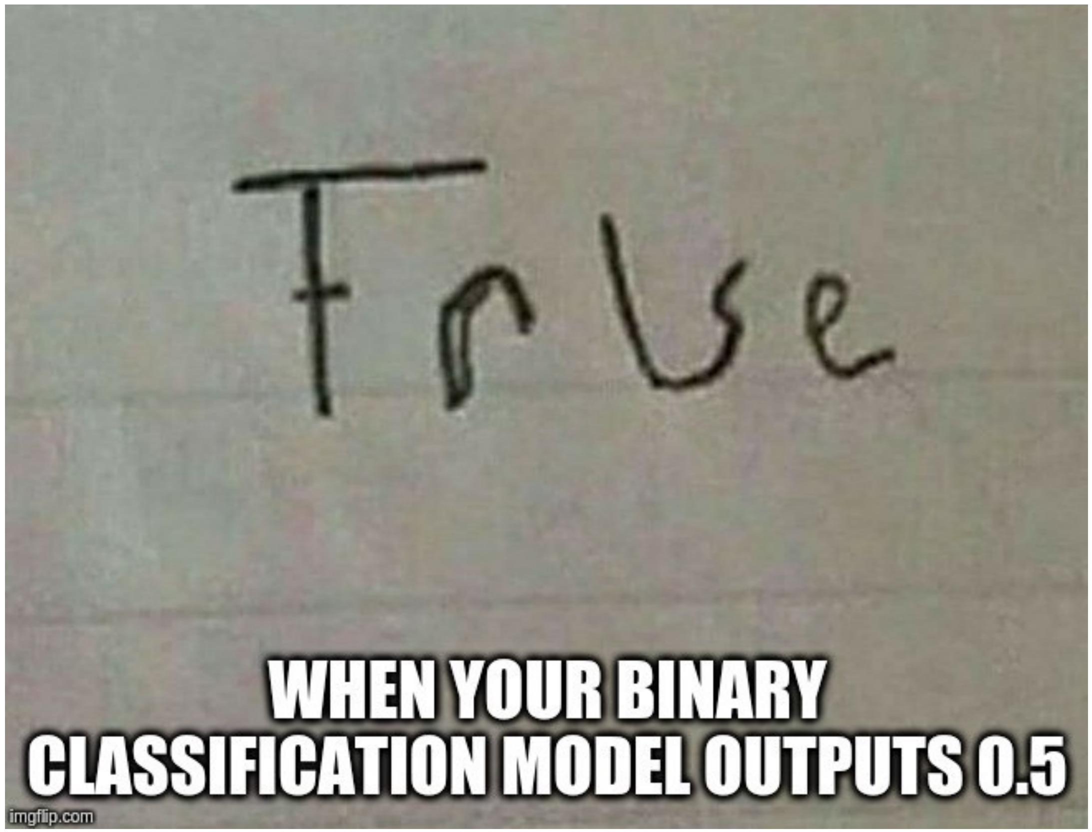


- $g(z) \geq 0.5$ whenever $z \geq 0$
- $g(\theta x^T) \geq 0.5$ whenever $\theta x^T \geq 0$

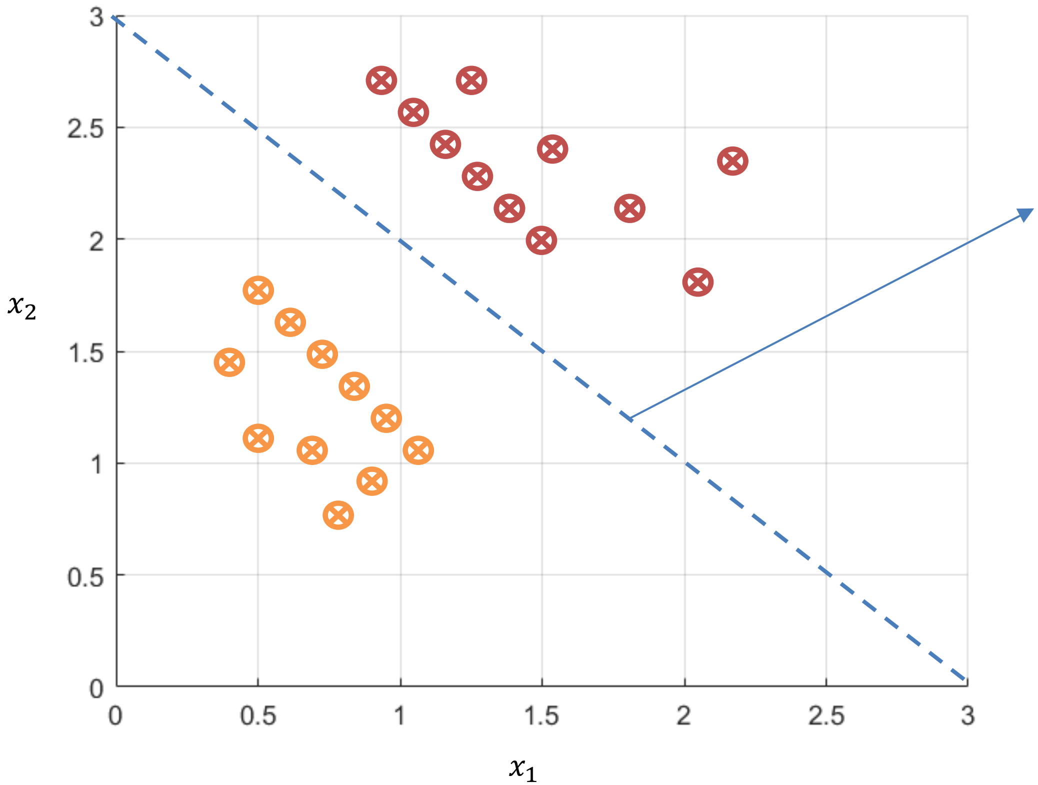
Predicts $\mathrm{y} = 1$ whenever $\theta x^T \geq 0$

Predicts $\mathrm{y} = 0$ whenever $\theta x^T < 0$

When the Hypothesis value is 0.5



Decision Boundary



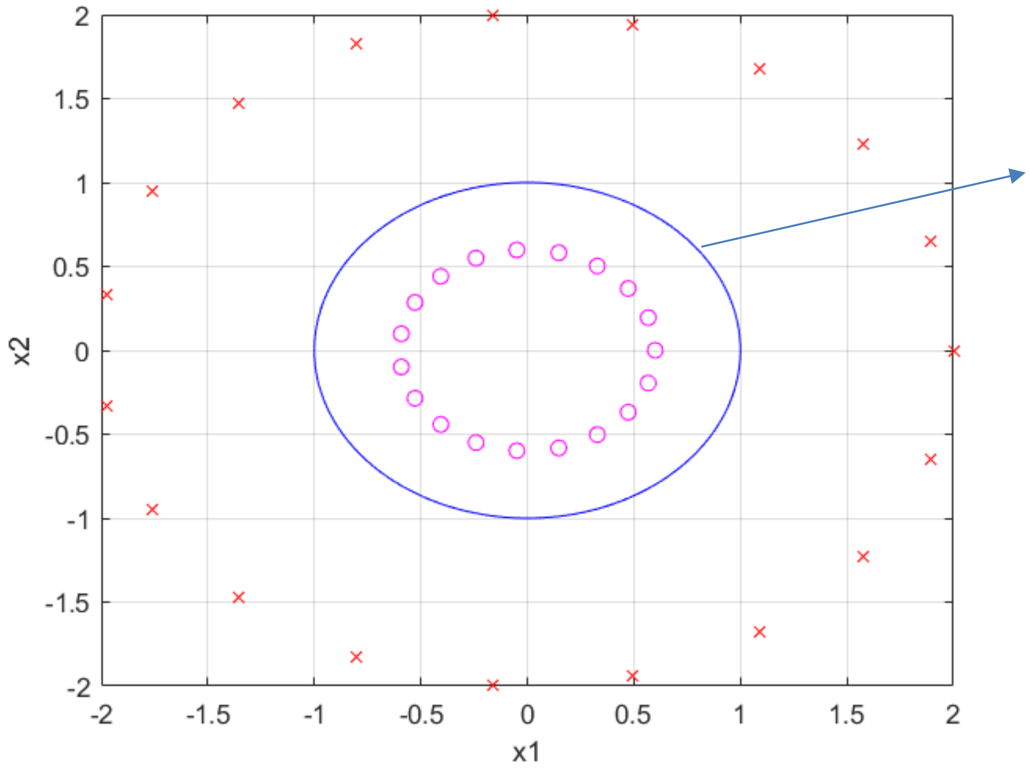
${}_{\theta}(x) = \mathrm{g}(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

θ_0= -3, θ_1= 1, θ_2=1

- $\hat{y} = 1$ if -3+x_1+x_2 $\geq$ 0
- $\hat{y} = 0$ if -3+x_1+x_2 < 0
- If x_1+x_2 $\geq$ 3 –> $\hat{y} = 1$
- If x_1+x_2 $<$ 3 –> $\hat{y} = 0$

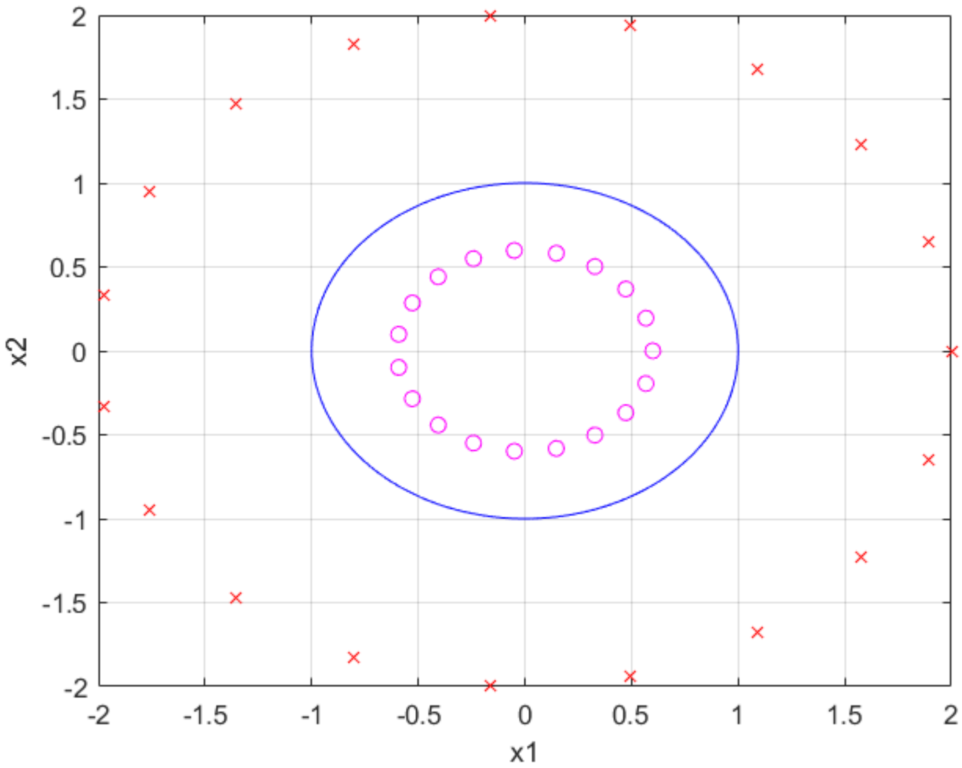
18

Non-linear Decision Boundary



How can we make this happen?

Non-linear Decision Boundary



<u>Hypothesis:</u>

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 {x_1}^2 + \theta_4 {x_2}^2)$$

- θ_0 = -1, θ_1=0, θ_2=0, θ_3=1, θ_4=1

We can add higher order polynomial features as needed

${x_1}^2 + {x_2}^2 \geq 1$ –> $\hat{y}$=1

${x_1}^2 + {x_2}^2 < 1$ –> $\hat{y}$=0

Logistic regression

- Logistic regression can be used to **determine very complex boundaries** if necessary

- Problem definition
  – Training Set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \ldots\ldots\ldots\ldots (x^{(m)}, y^{(m)})\}$
  – $x \in [x_0, x_1, x_2, x_3 \ldots \ldots \ldots x_n]$
  – $y \in [0,1]$
  – $h_\theta(x) = \frac{1}{1+e^{-\theta x^T}}$

- How to choose θ?

Linear regression cost function

- $J = \frac{1}{2m}\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right)^2$ - from previous class

Can this be applied for logistic regression?

- Answer is yes and no (depends)
- Mean square error performs **reasonably well but not consistent**
- Mean square error are used for certain classification problems, however, **there is a possibility of local minima** , Not a convex function always!
- Gradient descent for classification using **mean square error(MSE) might not converge to global minima**

Logistic regression cost function

$$P(y \mid x) = \begin{cases} \hat{p} & \text{if } y = 1 \\ 1 - \hat{p} & \text{if } y = 0 \end{cases}$$

Max/Min occours at points where derivative is equal to 0:

$$\frac{\partial P(y \mid x)}{\partial p} = 0$$

Our examples can only be part of one class at a time either 1 or 0 so,

- when $y = 1$ then $\hat{p}$ should be the output and $1 - \hat{p}$ should be ignored,
- similarly, when $y = 0$ then $1 - \hat{p}$ should be the output and $\hat{p}$ should be ignored.

We can combine this as follows:

$$\boxed{P(y \mid x) = \hat{p}^y (1 - \hat{p})^{1-y}}$$

if $y = 1$ then:

$$P(y \mid x) = \hat{p}^y (1 - \hat{p})^{1-y} = \hat{p}^1 (1 - \hat{p})^{1-1} = \hat{\mathbf{p}}$$

if $y = 0$ then:

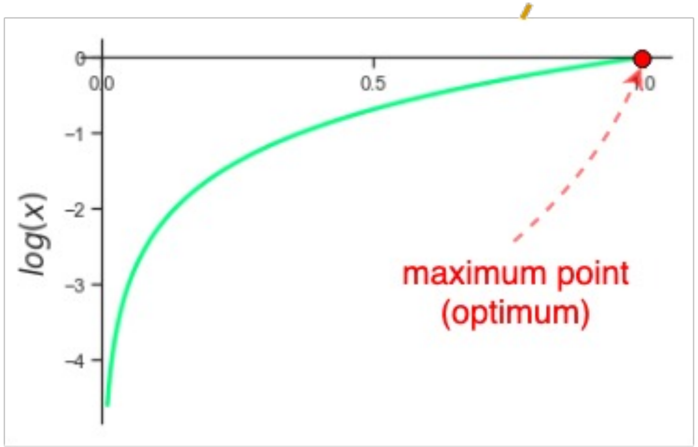
$$P(y \mid x) = \hat{p}^y (1 - \hat{p})^{1-y} = \hat{p}^0 (1 - \hat{p})^{1-0} = \mathbf{1 - \hat{p}}$$

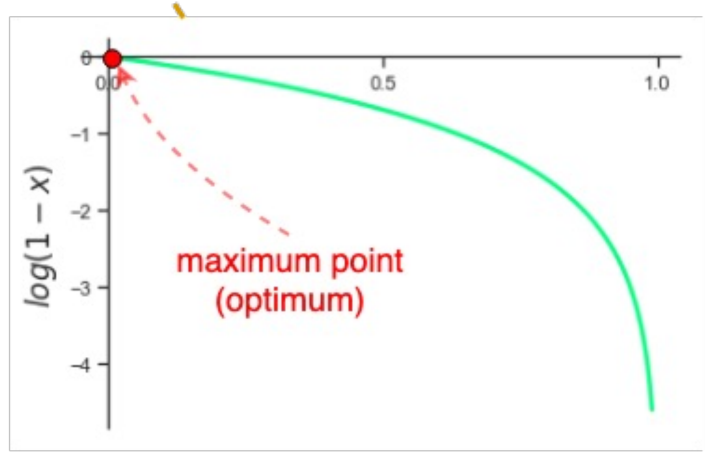
Logistic regression cost function

Recall the properties of natural log:

- $\log(a * b) = log(a) + \log(b)$
- $log(a^x = x\log(a))$

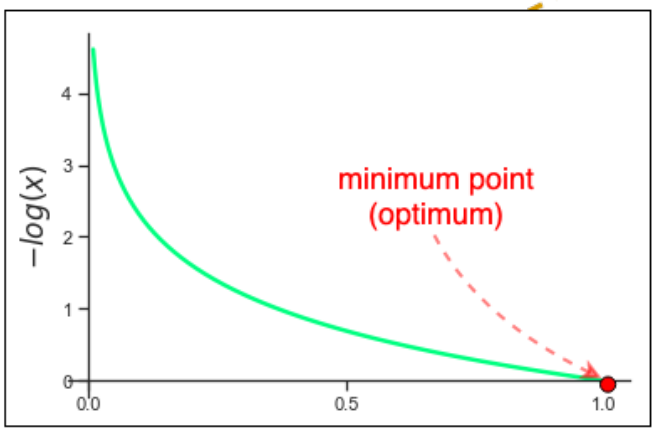
$$\begin{aligned}\log(P(y \mid x) &= \log(\hat{p}^y(1-\hat{p})^{(1-y)}) \\ &= \log(\hat{p}^y) + \log((1-\hat{p})^{(1-y)}) \\ &= \mathbf{y}\log(\hat{\mathbf{p}}) + (\mathbf{1}-\mathbf{y})\log(\mathbf{1}-\hat{\mathbf{p}})\end{aligned}$$

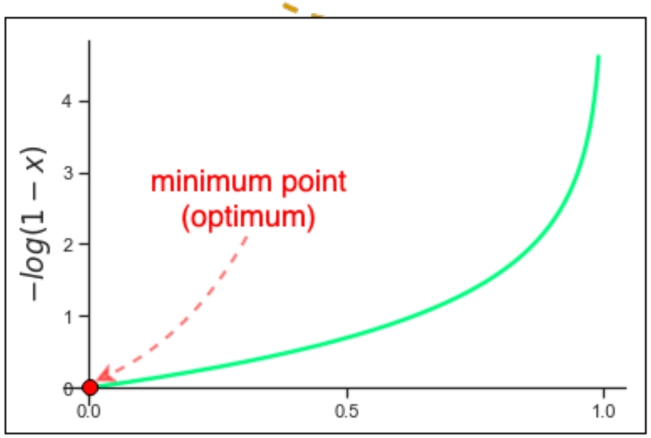




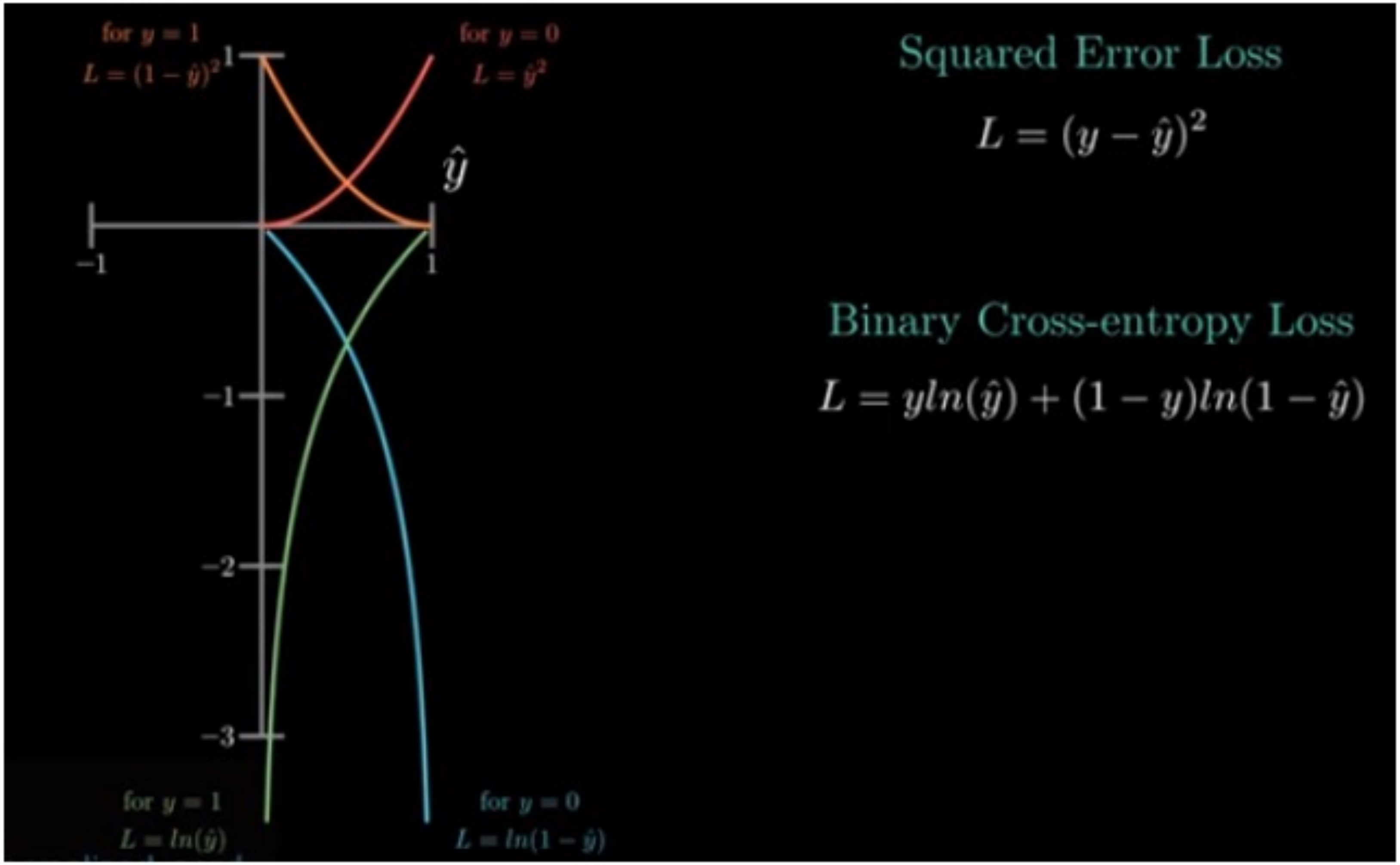
Logistic regression cost function

$$
\begin{aligned}
Binary\,Cross\,Entropy\,Loss(y, \hat{p}) &= -\log(P(y \mid x)) \\
&= -(y \log(\hat{p}) + (1 - y) \log(1 - \hat{p})) \\
&= \mathbf{-y \log(\hat{p}) - (1 - y) \log(1 - \hat{p})}
\end{aligned}
$$





Logistic regression cost function



Logistic regression cost function

$$\mathcal{L}(\hat{p}^{(1)}, \hat{p}^{(2)}, \cdots, \hat{p}^{(m)} \mid (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})) = \prod_{i=1}^{m} \hat{p}^{(i)\,y^{(i)}} (1 - \hat{p}^{(i)})^{1-y^{(i)}}$$

This is the ***likelihood($\mathcal{L}$)*** function, where we are trying to maximize the *each* probability, $\hat{p}$, given the examples $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})$.

$\prod_{i=1}^{m}$ represents product over examples 1 to m

$$\begin{aligned}
\log\left(\mathcal{L}(\hat{p}^{(1)}, \hat{p}^{(2)}, \cdots, \hat{p}^{(m)} \mid (x^{(1)}, y^{(1)}, (x^{(2)}, y^{(2)}, \cdots, (x^{(m)}, y^{(m)}))\right) &= \log\left(\prod_{i=1}^{m} \hat{p}^{(i)\,y^{(i)}} (1 - \hat{p}^{(i)})^{1-y^{(i)}}\right) \\
&= \sum_{i=1}^{m} \log\left(\hat{p}^{(i)\,y^{(i)}} (1 - \hat{p}^{(i)})^{1-y^{(i)}}\right) \\
&= \sum_{i=1}^{m} y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})
\end{aligned}$$

This is called the ***log-likelihood*** function

Logistic regression cost function

$$\text{Binary Cross-Entropy Cost} = \frac{1}{m}\left(-\log\left(\mathcal{L}(\hat{p}^{(1)}, \hat{p}^{(2)}, \cdots, \hat{p}^{(m)})\right)\right)$$
$$= \frac{\mathbf{1}}{\mathbf{m}}\sum_{\mathbf{i=1}}^{\mathbf{m}} -\mathbf{y}^{(\mathbf{i})}\log(\hat{\mathbf{p}}^{(\mathbf{i})}) - (\mathbf{1}-\mathbf{y}^{(\mathbf{i})})\log(\mathbf{1}-\hat{\mathbf{p}}^{(\mathbf{i})})$$

$$-\log\left(\mathcal{L}(\hat{p}^{(1)}, \hat{p}^{(2)}, \cdots, \hat{p}^{(m)})\right) = -\sum_{i=1}^{m} y^{(i)}\log(\hat{p}^{(i)}) + (1-y^{(i)})\log(1-\hat{p}^{(i)})$$
$$= \sum_{i=1}^{m} -y^{(i)}\log(\hat{p}^{(i)}) - (1-y^{(i)})\log(1-\hat{p}^{(i)})$$

Logistic regression cost function

- $Cost\left(h_\theta\left(x^i\right) - y^i\right) = -y\log\left(h_\theta(x)\right) - (1-y)\log\left(1 - h_\theta(x)\right)$

Since y is either going to be 0 or 1

- $\boldsymbol{J} = \frac{1}{m}\left[\sum_{i=1}^{M} -y^i\, log\left(h_\theta\left(x^i\right)\right) - (1-y^i)\, log\left(1 - h_\theta\left(x^i\right)\right)\right]$

Principle of Maximum Likelihood

- $\boldsymbol{J} = \frac{-1}{m}\left[\sum_{i=1}^{M} y^i\, log\left(h_\theta\left(x^i\right)\right) + (1-y^i)\, log\left(1 - h_\theta\left(x^i\right)\right)\right]$

Gradient Descent(GD)

- To fit parameters θ
- To make a prediction :
  - $h_\theta(x)= \frac{1}{1+e^{-\theta x^T}}$
- $J = \frac{-1}{m}[\sum_{i=1}^{M} y^i \, log\left(h_\theta\left(x^i\right)\right) + (1 - y^i)\, log\left(1 - h_\theta\left(x^i\right)\right)]$
- Minimize $J(\theta)$

Gradient Descent – Update θ

- $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} \boldsymbol{J}(\theta)$ - Similarly update all θ_j

- $\frac{\partial}{\partial \theta_j} \boldsymbol{J}(\theta) = \frac{-1}{m} \frac{\partial}{\partial \theta_j} [\sum_{i=1}^{M} y^i \, log\left(h_\theta\left(x^i\right)\right) + (1 - y^i) \, log\left(1 - h_\theta\left(x^i\right)\right)]$

Gradient Descent Equation

- $\frac{\partial}{\partial \theta_j} \boldsymbol{J}(\theta) = \frac{1}{m} \sum_{i=1}^{M} \left(h_\theta\left(x^i\right) - y^i \right) x^i$

Exactly the same as Linear Regression except the

$h_\theta\left(x^i\right)$

Logistic Regression Steps

- Load the dataset: Input Features and Target Variable

- Normalize Features
  - $\hat{x} = \frac{x - \mu}{\sigma}$ or any other methods
  - Add $\boldsymbol{x_0} = 1$ to all samples (Normalized $\hat{x}$)
  - new Dimensions of $\boldsymbol{x}$ – M x (n+1)
- Implement cost function after computing hypothesis
  - $h_\theta(\mathrm{x}) = g(\theta x^T), g(z) = \frac{1}{1+e^{-z}}$
  - $J = \frac{-1}{m}[\sum_{i=1}^{M} y^i \, log\left(h_\theta\left(x^i\right)\right) + (1 - y^i) \, log\left(1 - h_\theta\left(x^i\right)\right)]$
  - No need of for loops in this scenario too
- Implement gradient descent function
  - $\boldsymbol{\theta} = \theta - \alpha\delta$
  - $\delta = \frac{1}{m}\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right) x^i$
- Compute $h_\theta(\mathrm{x})$ with updated θ
- Threshold $h_\theta(\mathrm{x})$ with 0.5 and determine $\hat{y} \in \{0,1\}$
- Evaluate the performance

Logistic regression for multi-class variable

- Email Folder: Work, Personal, Spam, Ads

$y = 1$ $y = 2$ $y = 3$ $y = 4$

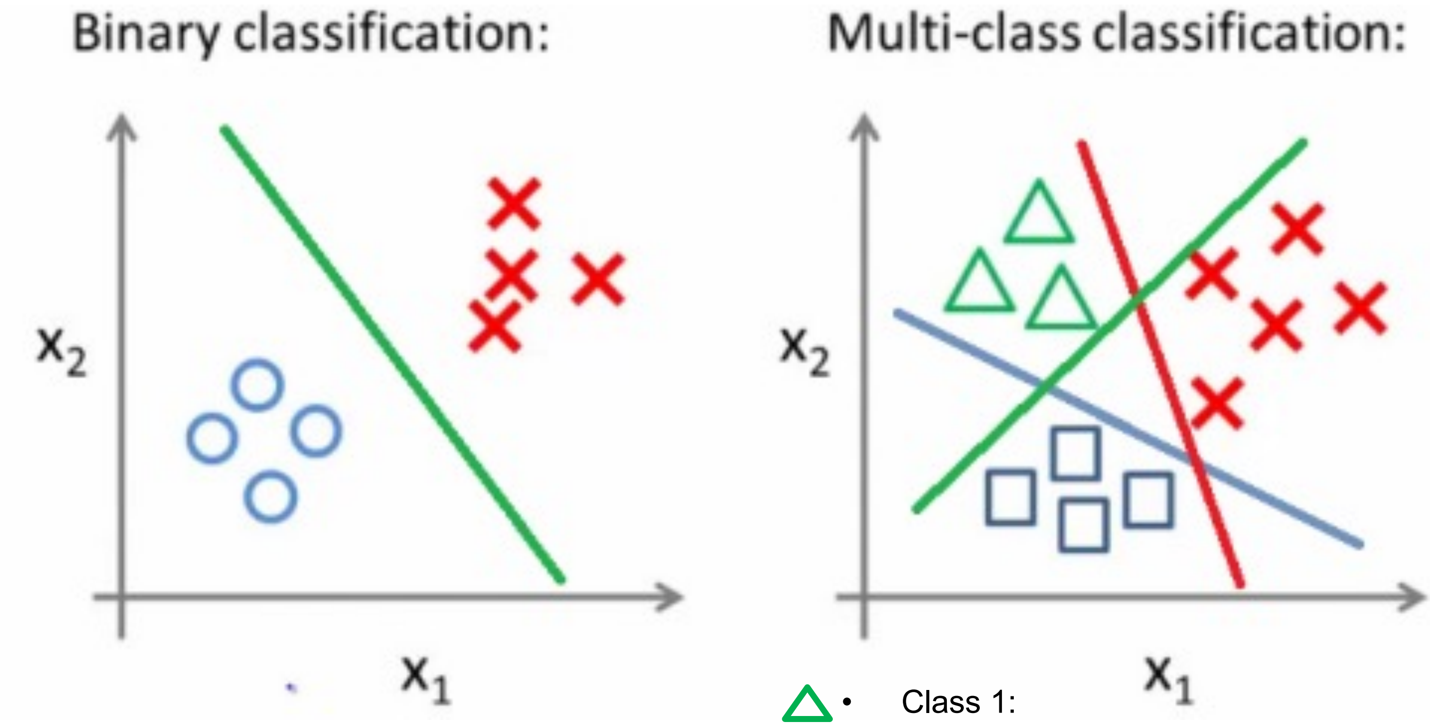
- Medical Imaging: No Cancer, Benign, Malignant

$y = 1$ $y = 2$ $y = 3$

- Weather: Sunny, Cloudy, Rain, Snow

$y = 1$ $y = 2$ $y = 3$ $y = 4$

Binary vs. Multi-Class Classification



- Class 1:
- Class 2: X
- Class 3:

9/8/22 35

Multi-Class Classification

- 3 Binary classification approaches:
  - Where class 1 is positive, classes 2 & 3 are negative
  - Where class 2 is positive, classes 1 & 3 are negative
  - Where class 3 is positive, classes 1 & 2 are negative

One vs. All

- Train a logistic regression classifier $h_\theta(x^i)$ for each class 'i' to predict the probability that $\mathrm{y} = i$
- On a new input x, to make a prediction, pick the class 'i' that maximize =>$max_i\ (h_\theta{}^i(\mathrm{x}))$
  - In this case, '3' classifiers – Pick the one with highest probability

PROBLEM OF UNDERFITTING /OVERFITTING

Regularization with **Ridge, Lasso, and Elastic Net Regressions**

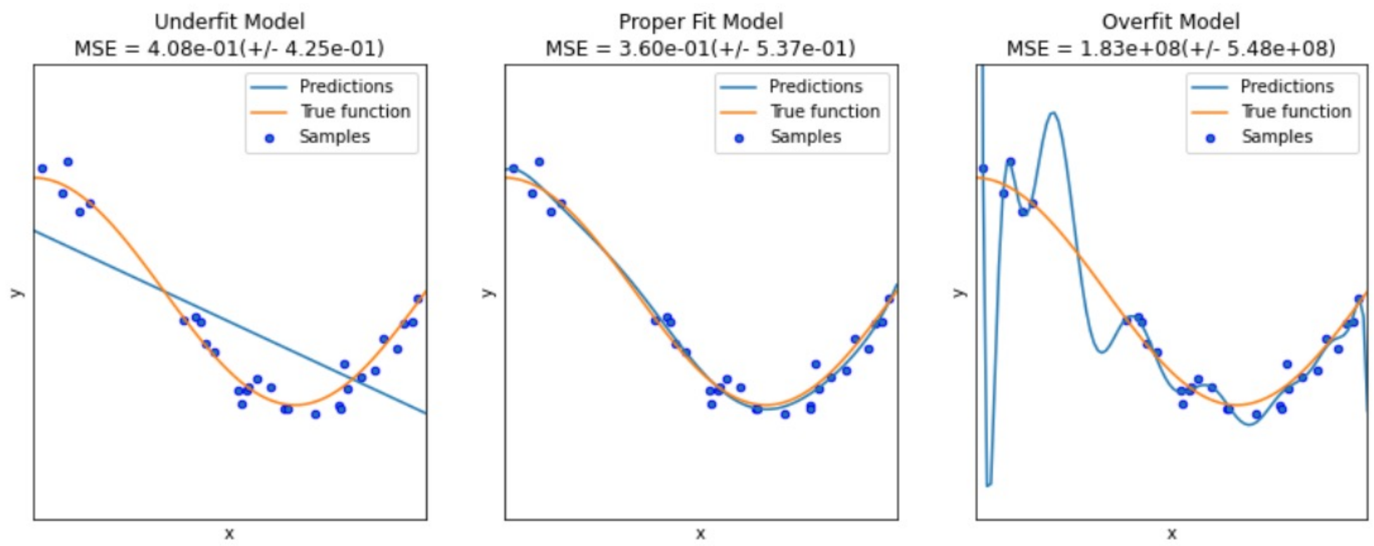
- ***What is Regularization***: Techniques for combating overfitting and improving training.
- A common phenomenon referred to as “overfitting”. Overfitting has **a polar opposite called underfitting**
- In technical terms, overfitting means the model you built **has more parameters than the data can justify**
- You may be asking *what does all this have to do with regularization?*
  – The answer is **everything**.

Measure to address overfitting

- Reduce number of features
  - Choose manually
  - Feature selection algorithm (Later in the course)
- Regularization
  - Keep all features, but **reduce magnitude/values of $\boldsymbol{\theta_j}$**
  - When we have a lot of features, each feature can make an important contribution to predict ‘y’
  - This method would help us preserve all features
- Overview of the differences in 3 common regularization techniques:
  - Ridge
  - Lasso, and
  - Elastic Net.

Regularization with Ridge, Lasso, and Elastic Net Regressions

- **Adding the right amount of bias to a model can help** make more accurate predictions by desensitizing it to some of the noise in the training data.



In the plots above the **blue dots are sample data points taken from the real-world**. The distance those samples are from the **yellow line, "True Function", is called the "noise" of the data.** The distance of the sample points to the blue line is referred to as the "error" of our model.

https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Ridge regression : regularized linear regression cost function

- $J = \frac{1}{\mathbf{2}m}\left[\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right)^2 + \lambda\sum_{j=1}^{n}\theta_j{}^2\right]$

Regularization Parameter

Shrink all parameters - reduces over reliance on any feature

- By convention, regularization is done only for θ_1 to θ_n and not for θ_o

- Small values for parameters:
  - "Simpler" Hypothesis
  - Less prone to overfitting

Ridge regression : regularized linear regression cost function

- $J = \frac{1}{2m}\left[\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right)^2 + \lambda \sum_{j=1}^{n} \theta_j{}^2\right]$

Keeping the parameters small to avoid overfitting

Fit the training set

Regularization Parameter: Trade off

- λ is adjusted to control the value of θ and minimal error
- Let's say, λ = 10^3, $\theta_j \cong 0$, $j = 1,2, \ldots \mathrm{n}$ which would mean $h_\theta(x)$=θ_0 and end up in underfitting
- λ should be chosen carefully

Ridge regression : regularized linear regression

- Regularized Linear Regression
  - $J = \frac{1}{2m}\left[\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right)^2 + \lambda \sum_{j=1}^{n} \theta_j{}^2\right]$
- Gradient descent without regularization
  - $\boldsymbol{\theta} = \theta - \alpha\delta$
  - $\delta = \frac{1}{m}\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right) x^i$

$\boldsymbol{h_\theta(x)=\theta x^T}$

- Gradient descent with regularization $\boldsymbol{x_0} = 1$
  - $\boldsymbol{\theta_0} = \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right) x_0^i$
  - $\boldsymbol{\theta_j} = \theta_j - \frac{\alpha}{m}\left[\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right) x_j^i + \lambda\theta_j\right]$
  - $\boldsymbol{\theta_j} = \theta_j\left[1 - \frac{\alpha\lambda}{m}\right] - \frac{\alpha}{m}\left[\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right) x_j^i\right]$

Ridge regression : regularized logistic regression

- Regularized Logistic Regression
  - $J = \frac{-1}{m}[\sum_{i=1}^{M} y^i \log\left(h_\theta\left(x^i\right)\right) + (1-y^i)\log\left(1-h_\theta\left(x^i\right)\right)]$
    $+[\frac{\lambda}{2m}\sum_{j=1}^{n} {\theta_j}^2]$
- Gradient descent without regularization
  - $\boldsymbol{\theta} = \theta - \alpha\delta$
  - $\delta = \frac{1}{m}\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right) x^i$

$$\boldsymbol{h_\theta(x)} = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{e}^{-\boldsymbol{\theta}\boldsymbol{x}^T}}$$

$$x_0 = 1$$

- Gradient descent with regularization
  - $\boldsymbol{\theta_0} = \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right) x_0^i$
  - $\boldsymbol{\theta_j} = \theta_j - \frac{\alpha}{m}\left[\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right) x_j^i + \lambda\theta_j\right]$
  - $\boldsymbol{\theta_j} = \theta_j\left[1 - \frac{\alpha\lambda}{m}\right] - \frac{\alpha}{m}\left[\sum_{i=1}^{M}\left(h_\theta\left(x^i\right) - y^i\right) x_j^i\right]$

Ridge regression : regularization technique

- This function **penalizes your model for having too many or too large predictors**
- The objective of Ridge regression is to reduce the effect of these predictors to **decrease the chance of overfitting your data.**
- If we were to set $\lambda = 0$ then this would be a normal linear regression.
- The most common use of Ridge regression is to be *preemptive* in **addressing overfitting concerns**
- Ridge regression is a good tool for handling multicollinearity *when you must keep all your predictors*.

Ridge regression works well if there are many predictors of about the same magnitude. This means all predictors have similar power to predict the target value.

Ridge regression : reference codes

```
from sklearn.model_selection import Ridge

# Create an array of alpha values to test
alphas = np.logspace(-1, 1.5, 500,base=10)

# Create a Ridge regression model instance
ridge = Ridge(random_state=0, max_iter=10000,alpha=alphas)

# Create dictionary key,value pair of alpha values
tuned_parameters = [{'alpha': alphas}]

# Specify number of folds for cross_validation
n_folds = 5

# Create grid search instance using desired variables
clf_ridge = GridSearchCV(ridge, tuned_parameters, cv=5, refit=False)
clf_ridge.fit(x_train_scaled, y_train)
ridge_scores = clf_ridge.cv_results_['mean_test_score']

# Plot the Figure
plt.figure().set_size_inches(8, 6)
plt.plot(alphas, ridge_scores)
plt.xlabel('Alpha Value')
plt.ylabel('Cross Validation Score')
plt.title('Ridge Regression Alpha Demonstration')
plt.axvline(clf_ridge.best_params_['alpha'], color='black',
linestyle='--')
print(f'The optimal alpha value is:
{clf_ridge.best_params_["alpha"]}')
```

Lasso Regression
: regularization techniques

- **Lasso regression** the equation below and thinking to yourself "that looks almost identical to Ridge regression.".
- Lasso differs from Ridge regression by summing the **absolute value of the predictors (m_j)** instead of summing the squared values.

$$\text{cost\_function\_lasso} = \sum_{i=1}^{n} (y_i - \sum_{j=1}^{k} (m_j x_{ij}) - b)^2 + \lambda \sum_{j=1}^{p} | m_j |$$

Lasso Cost Function — Notice the lambda (λ) multiplied by the sum of the absolute value of the predictors

- Lasso is an acronym that stands for "Least Absolute Shrinkage and Selection Operator." ***Due to the penalty term not being squared, some values can reach 0***. *When a predictor coefficient (m_j) reaches 0 that predictor does not affect the model.*
- Lasso tends to do well **if there are few significant predictors and the** magnitudes of the others are close to zero**.**

Lasso regression : reference codes

```
from sklearn.model_selection import GridSearchCV

# Create an array of alpha values to test
# Start np.linspace value is 10**-10 because a value of 0 throws
warnings
alphas = np.logspace(-10, 1, 1000,base=10)

# Create dictionary key,value pair of alpha values
tuned_parameters = [{'alpha': alphas}]

# Specify number of folds for cross_validation
n_folds = 5

# Create grid search instance using desired variables
clf_lasso = GridSearchCV(lasso, tuned_parameters, cv=5, refit=True)
clf_lasso.fit(x_train_scaled, y_train)
lasso_scores = clf_lasso.cv_results_['mean_test_score']

# Plot the results
plt.figure().set_size_inches(8, 6)
plt.plot(alphas, lasso_scores)
plt.xlabel('Alpha Value')
plt.ylabel('Model CV Score')
plt.title('Lasso Regression Alpha Demonstration')
plt.axvline(clf_lasso.best_params_['alpha'], color='black',
linestyle='--')
print(f'The optimal alpha value is :
{clf_lasso.best_params_["alpha"]}')
```

Elastic Net Regression
: regularization technique

- **Elastic Net regression** was created as a critique of Lasso regression.
- **Elastic Net** is a combination of both Lasso and Ridge regressions. .

$$argmin_{\beta} \sum_{i} (y_i - \boldsymbol{\beta}' \boldsymbol{x}_i)^2 + \lambda_1 \sum_{k=1}^{K} |\beta_k| + \lambda_2 \sum_{k=1}^{K} \beta_k^2$$

The λ_1 is Lasso penalty (L1) and λ_2 is the Ridge regression penalty (L2)

Elastic Net Regression
: regularization technique

- As you can see in the picture above there are now two λ terms. λ_1 is the “alpha” value for the Lasso part of the regression and λ_2 is the “alpha” value for the Ridge regression equation.

- When using sci-kit learn’s Elastic Net regression the alpha term is a ratio of λ_1:λ_2

- Setting the ratio values:
  - **ratio = 0 it acts as a Ridge regression, and**
  - **when the ratio = 1 it acts as a Lasso regression.**
  - **Any value between 0 and 1 is a combination of Ridge and Lasso regression**

Summary : regularization technique

- ***When do I use Regularization:***

  - Ridge regression — Ridge regression works well if there are many large predictors of about the same value.

  - Lasso — Few significant predictors and the magnitudes of the others are close to zero

  - Elastic Net — Mixture of both Ridge and Lasso

- ***How do I use Regularization:***
  - Split and Standardize the data (only standardize the model inputs and not the output)
  - Decide which regression technique Ridge, Lasso, or Elastic Net you wish to perform.
  - Use **GridSearchCV** to optimize the hyper-parameter alpha
  - Build your model with optimized alpha and make predictions!

Summary

- Logistic regression
- Overfitting and under fitting problems
- Solutions with regularization techniques

- What's next ?
  - Support Vector Machine (SVM)
  - Kernelization approaches