GEORGE BROWN COLLEGE

SCHOOL OF COMPUTER TECHNOLOGY

AASD 4001
Mathematical Concepts for Machine Learning

Lecture 2

Reza Moslemi, Ph.D., P.Eng.

Session 1

- Mathematics of Natural Language Processing algorithms
- Regression algorithms

Mathematics of Natural Language Processing algorithms

What is Natural Language Processing (NLP)?

- Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language. NLP draws from many disciplines, including computer science and computational linguistics, in its pursuit to fill the gap between human communication and computer understanding.

Why NLP?

- There is a wealth of audio- and text-based resources. If we can use NLP to our advantage, we gain worthy information.

- Spam/ham detection for fraud prevention in text messages/emails
- An online business analyzing reviews on its products
- An automaker using NLP to learn the public feedback to its latest vehicle line
- Analyzing social media trends, what people like, what they discuss most, etc.

Mathematics of Natural Language Processing algorithms (cont'd)

Building blocks of human language are words, but machine learning algorithms usually work with vectors of features.

$$\text{Text} \xrightarrow{\hspace{6em}} \text{vector of features}$$

Let's see how it can be done.

Mathematics of Natural Language Processing algorithms (cont'd)

Consider the following 2 phrases:

- College Student
- College Professor

We can use their word count to make a vector of the features (words)

- Let's gather all the words in both phrases: (college, student, professor)
- College Student -> (1,1,0)
- College Professor -> (1,0,1)
- These vectors, representing a document as a vector of words, are often known as bag of words

$$BOW_{Matrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Mathematics of Natural Language Processing algorithms (cont'd)

It is common to improve the bag of words by using the TF-IDF (Term Frequency – Inverse Document Frequency) method.

- TF: importance of the term in the document
- IDF: importance of the term in all documents (corpus)

TF-IDF is a popular scoring approach used to weigh terms for NLP tasks because it assigns a value to a term according to its importance in a document scaled by its importance across all documents in your corpus, which mathematically eliminates naturally occurring words, and selects words that are more descriptive of your text.

TF = The number of times a word appears in a document divided by the total number of words in the document

- e.g.: when a 30-word document contains the term "student" 5 times, the TF for the word 'student' is 5/30=1/6.

Mathematics of Natural Language Processing algorithms (cont'd)

IDF = $log\left(\frac{N}{df_x}\right)$

df_x: number of documents containing x

N: total number of documents

- e.g.: Let's assume the size of the corpus is 100 documents. If there are 20 documents that contain the term "student", then the IDF is:
  $log\left(\frac{100}{20}\right) = log5 = 0.70$

Mathematically, TF-IDF ($W_{x,y}$) of a word x in a document y is obtained from:

$$W_{x,y} = TF_{x,y} \times IDF_x$$

Mathematics of Natural Language Processing algorithms (cont'd)

Using bag of words with TF-IDF, we end up with a sparse matrix:

$$\begin{pmatrix} 1.0 & 0 & 5.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.0 & 0 & 0 & 0 & 0 & 11.0 & 0 \\ 0 & 0 & 0 & 0 & 9.0 & 0 & 0 & 0 \\ 0 & 0 & 6.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7.0 & 0 & 0 & 0 & 0 \\ 2.0 & 0 & 0 & 0 & 0 & 10.0 & 0 & 0 \\ 0 & 0 & 0 & 8.0 & 0 & 0 & 0 & 0 \\ 0 & 4.0 & 0 & 0 & 0 & 0 & 0 & 12.0 \end{pmatrix}$$

Where each row represents a phrase and each column represents a given word, with the TF-IDF being the value of the cell.

Regression algorithms

In machine learning, regression algorithms are supervised models for estimating the relationships between a dependent variable and one or more independent variables.

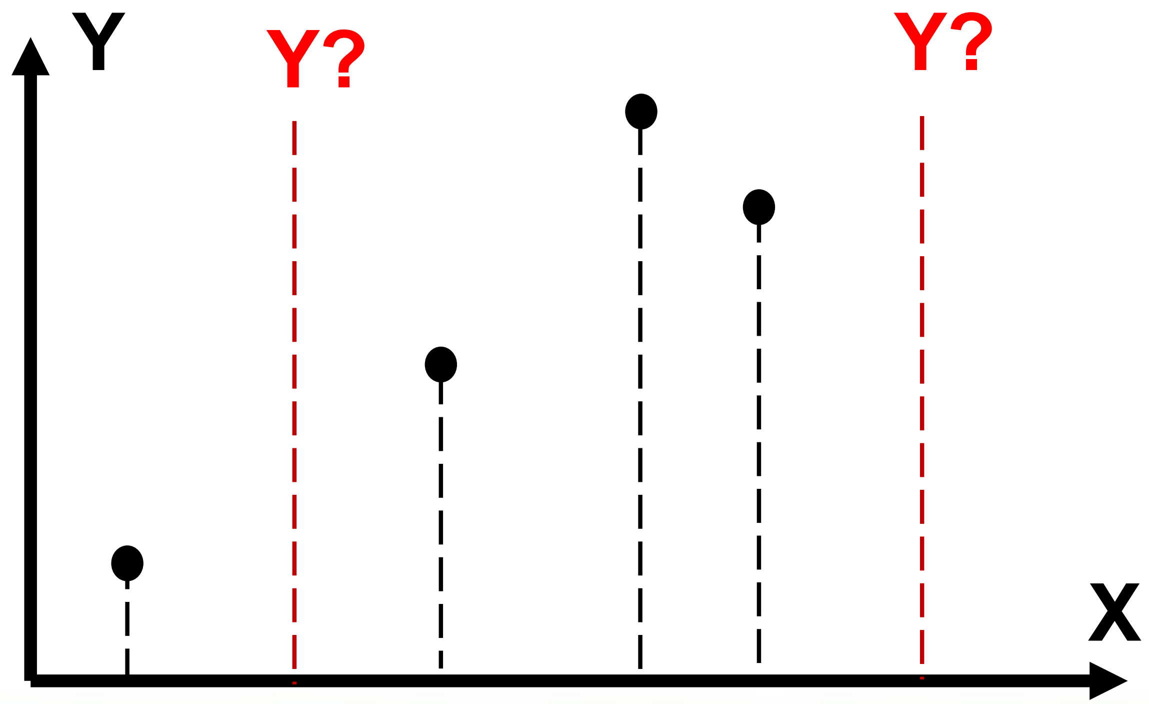
- Linear Regression
- Logistic Regression (for classification)
- Etc.

What is a supervised learning model?

Regression algorithms (cont'd)

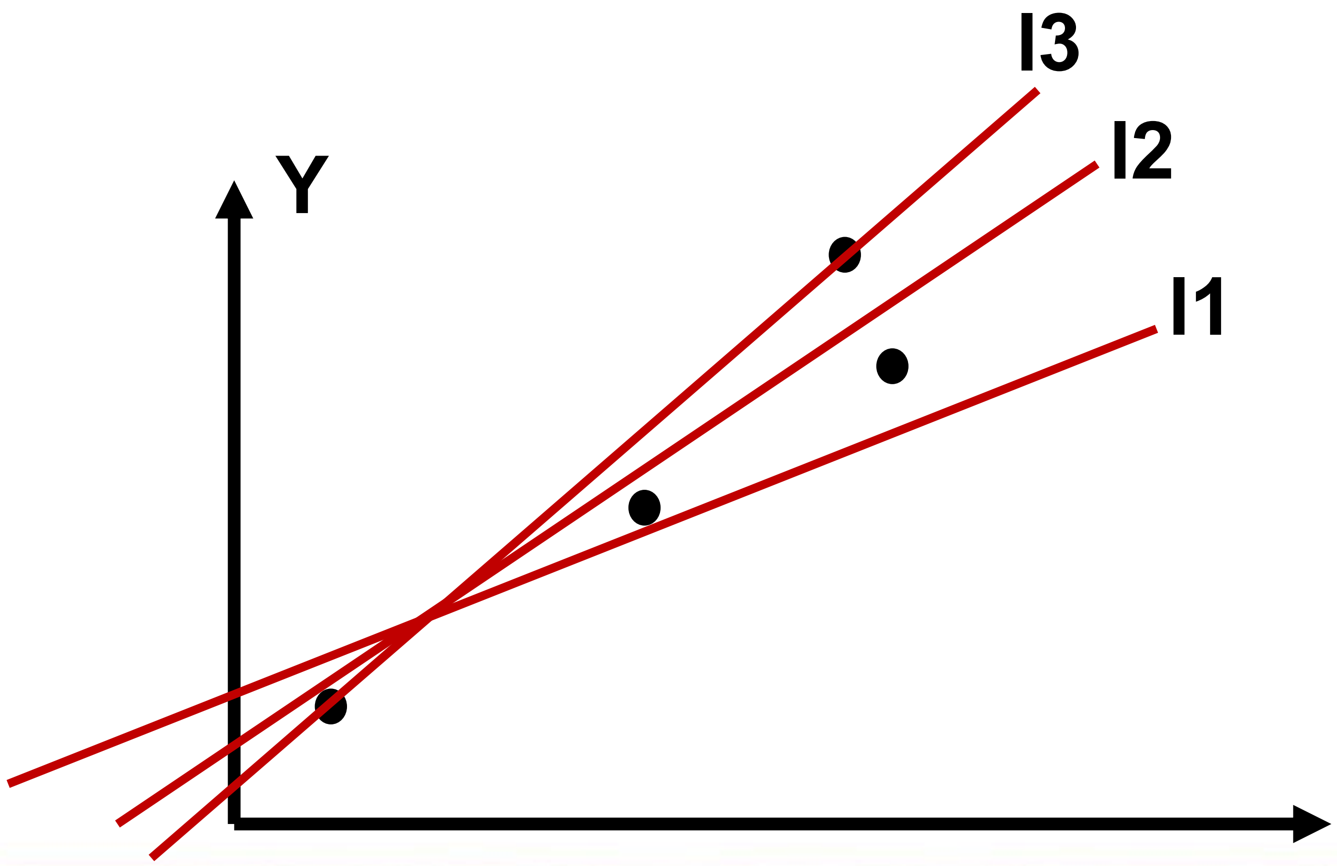
Linear regression is

Let's look at this in 2D:



Regression algorithms (cont'd)

But there could be an infinite number of lines.



Regression algorithms (cont'd)

Which one to choose?

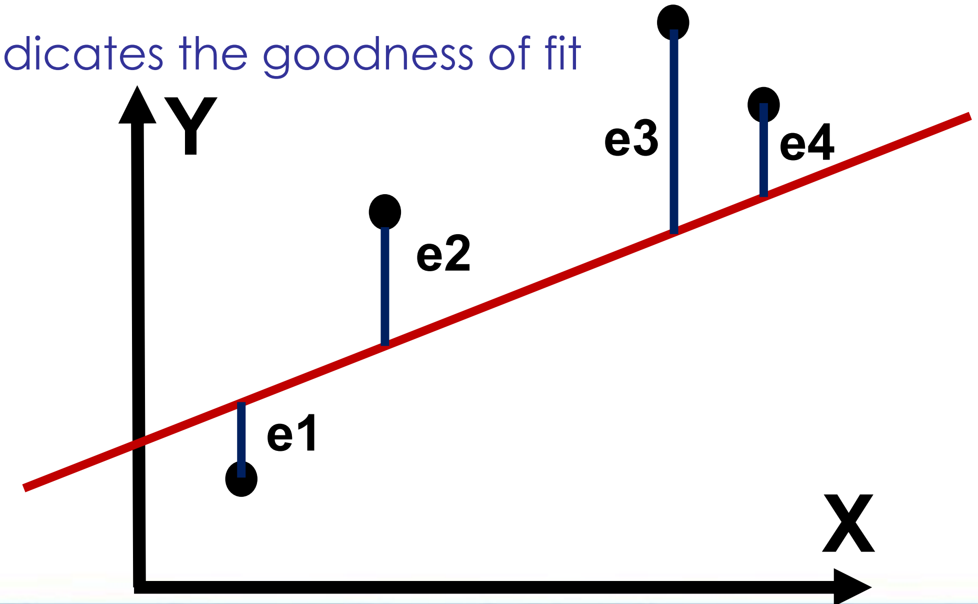
Constraints: ground truth, intercept, etc.

We will adopt a least square *loss function*

- A metric to minimize the overall error (sum of squared residuals)
  - $LSLF = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ where y_i and $\hat{y}_i$ are the actual and estimated y-values at x_i, respectively.
- R-squared (Coef. of determination) indicates the goodness of fit

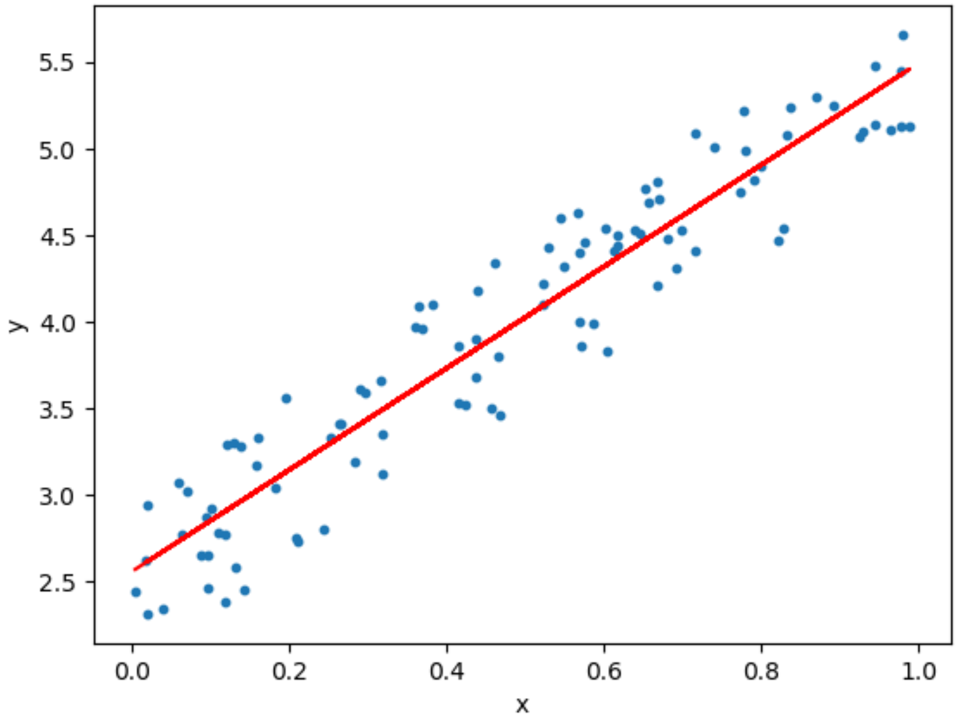
$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

- R-squared lies between 0 and 1
  - $R^2 = 0$ means a very bad fit
  - $R^2 = 1$ means an excellent fit
- Other metrics: RMSE, MAE, MAPE



Regression algorithms (cont'd)

This idea can be readily extended to *n* points.



Let's move to jupyter and open "Linear Regression.ipynb" to continue.

Regression algorithms (cont'd)

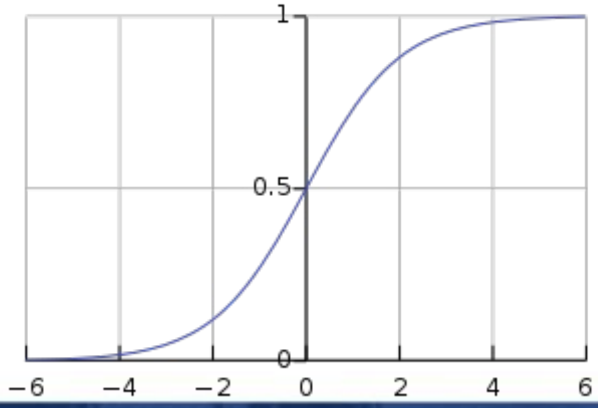
Logistic regression (logit regression) :

The first 2 are Binary classification (two classes only, usually 0 and 1) whereas the last example is multiclass classification (Multinomial Logistic/Multi-class Logistic Regression using Softmax function instead of Sigmoid)

Logistic regression employs the so-called logistic function (sigmoid):

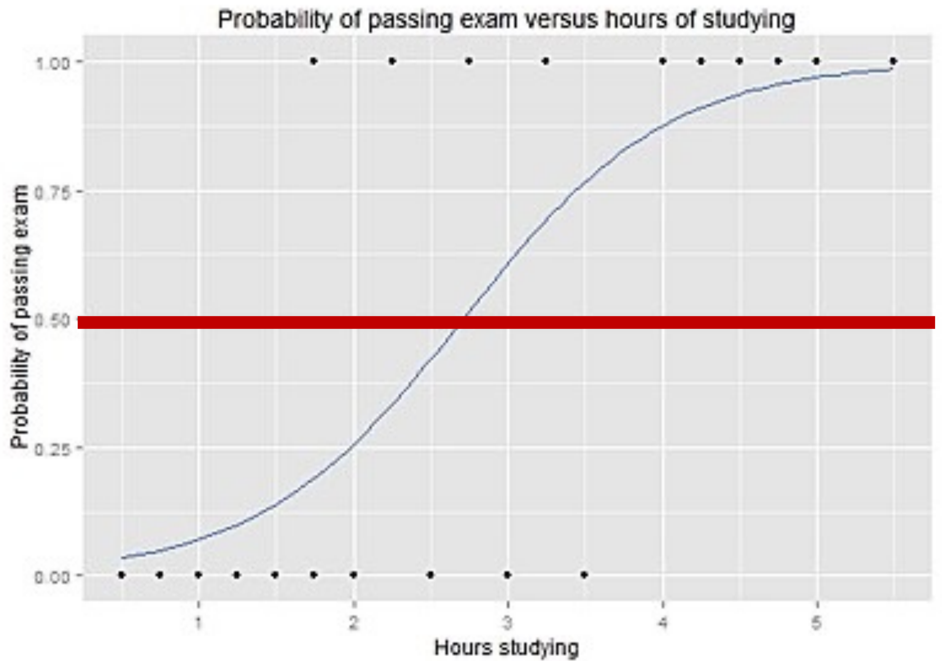
$$p = \frac{1}{1 + e^{-l}}$$

$$logit: l = \ln(\frac{p}{1 - p})$$



Regression algorithms (cont'd)

Using logistic function, it can predict values that lie between 0 and 1. Then, using a threshold (usually 0.5) it can classify into 2 classes. Logistic regression has discreet outputs as opposed to linear regression. Dependent variable is categorical.



In order to evaluate the quality of a logistic regression model, we use the so-called *confusion matrix (error matrix)*.

Regression algorithms (cont'd)

A confusion matrix is given below:

		Actual Class	
		P	N
Predicted Class	P	**TP**	FP
	N	FN	**TN**

where:

P = Positive; N = Negative

TP = True Positive; FP = False Positive; TN = True Negative; FN = False Negative

Regression algorithms (cont'd)

An example of a confusion matrix for disease diagnosis for 200 patients:

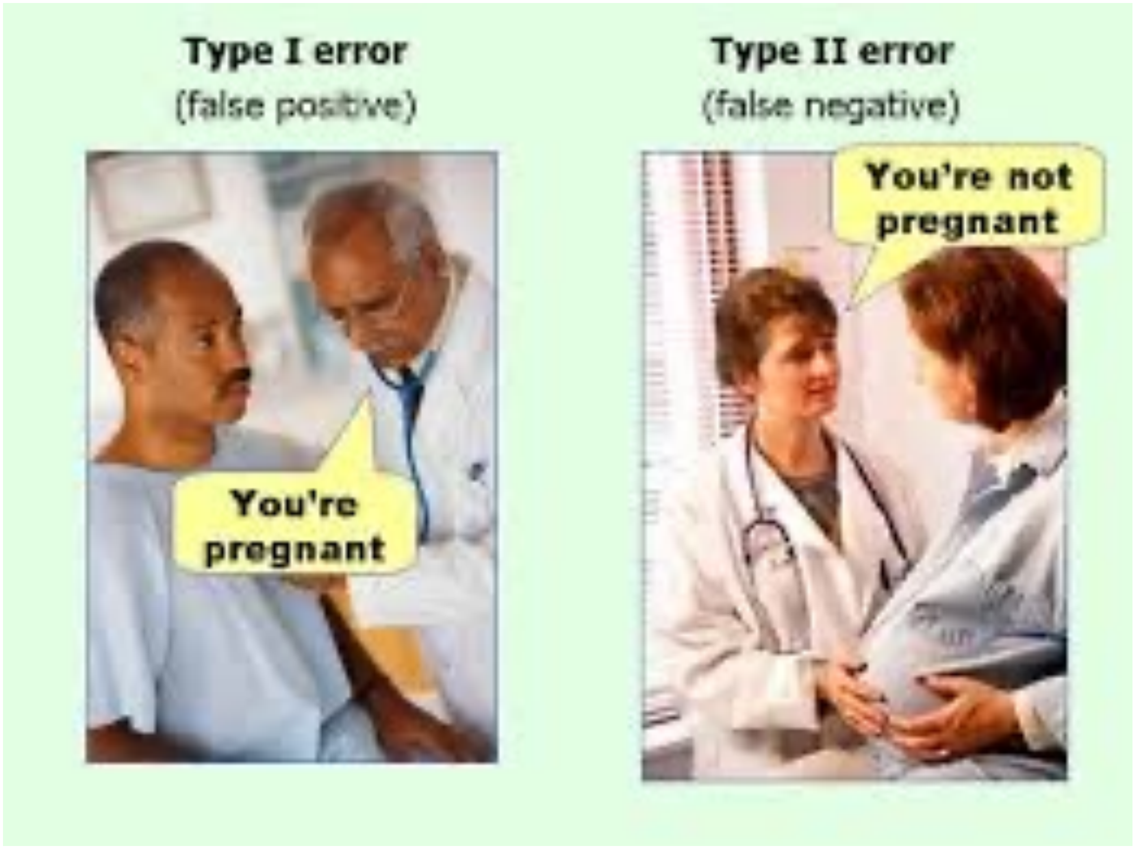
n = 200 patients		Actual Class	
		P	N
Predicted Class	P	**40**	15
	N	5	**140**

A couple of definitions:

- Accuracy: (TP+TN)/TOTAL = 180/200 = 90%
- Misclassification error: (FP+FN)/TOTAL = 20/200 = 10%

Regression algorithms (cont'd)

We try to avoid FP and FN predictions:



Let's move to jupyter and open "Logistic Regression.ipynb" to discuss logistic regression in more detail.