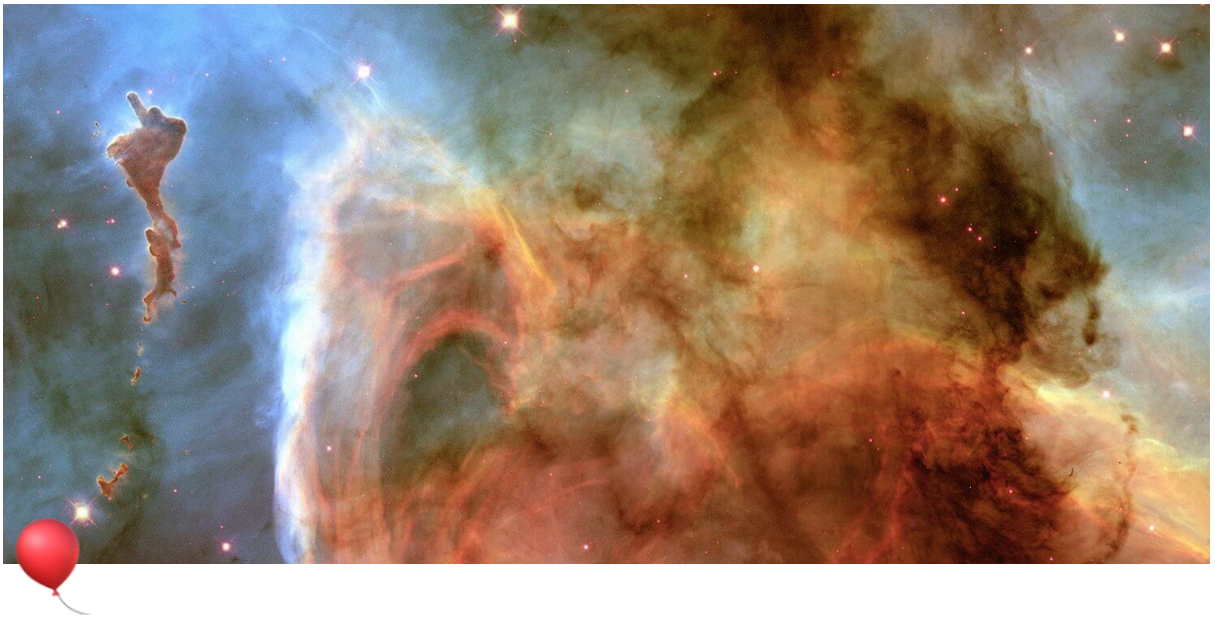


Classification and Logistic Regression

Some concepts

Recall:

- Classification: predict categorical labels, e.g. spam detection.
- Regression: predict real values, e.g. stock price prediction.

Some learning algorithms:

- Linear regression: a linear and parametric model for regression problems.
- Logistic regression: a linear and parametric model for classification problems *(contrary to its name!)*
- K-Nearest Neighbours (KNN) : a non-parametric model that can be used for both classification and regression problems.

Parametric and Non-parametric Models

Parametric models:

- A model that summarizes data with a finite set of parameters.
- Make assumptions on data distributions.
- E.g. linear/logistic regression, neural networks

Non-parametric models:

- A model that cannot be characterized by a bounded set of parameters.
- No assumptions on data distributions.
- E.g. instance-based learning that generate hypotheses using training examples, including kNN, decision trees, etc.

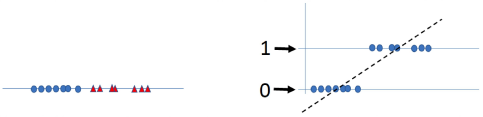
Classification problems:

Logistic regression

Similar to linear regression,

- 1) Model formulation
- 2) Cost function
- 3) Learning algorithm by gradient descent

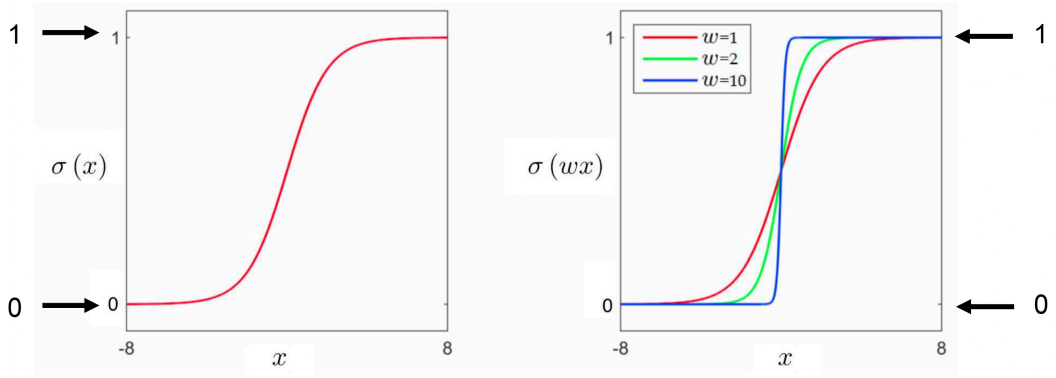
Can we use linear regression to classify them?



Sigmoid function

$$y = \frac{1}{1 + e^{-x}}$$

Visualize this function



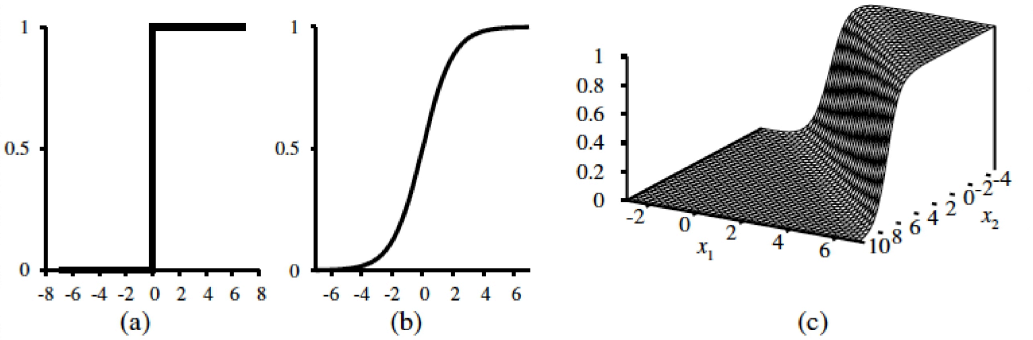
- It is a smoothed version of a step function (e.g. 0 for negative numbers and +1 for positive numbers).
- Also seen in neural networks.

As $\lim_{x \to \infty} e^{-x} \to 0$.
$\therefore y = \frac{1}{1+e^{-x}} \to 1$

When $\lim_{x \to -\infty} e^{-x} \to \infty$.
$\therefore y = \frac{1}{1+e^{-x}} \to \frac{1}{\infty} \to 0$

This is proof why the sigmoid function only returns values between 0 and 1 because these are its limits as $x \to \infty \quad and \quad x \to -\infty$

Sigmoid function



Step function	Univariate sigmoid function	Multivariate sigmoid function
Discontinuous	Continuous	Continuous
Non-differentiable	Differentiable	Differentiable
Hard to be optimised		

Just highlighting how discrete functions are differentiable

and how

continuous functions are non-differentiable

Model (univariate case)

- We change the linear model slightly by passing it through a nonlinearity.
- Composite function: a linear function (inner) embedded in the sigmoid function (outer).
- If x has 1 attribute, we have:

$$h(x; \boldsymbol{w}) = \sigma(w_0 + w_1 x) = \frac{1}{1 + e^{-(w_0 + w_1 x)}}$$

- The function $\sigma(u) = \frac{1}{1+e^{-u}}$ is called sigmoid function or logistic function.

Go to https://www.desmos.com/calculator
Type: y = 1/1+exp(-(w_0+w_1*x))

Model (general case)

- If **x** has d attributes, we have:

$h(\boldsymbol{x}; \boldsymbol{w}) = \sigma(w_0 + w_1 x_1 + \cdots + w_d x_d) = \frac{1}{1+e^{-(w^T x)}}$, where

All components of $\boldsymbol{w}$ are free parameters.

$$\boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \cdots \\ w_d \end{pmatrix}, \boldsymbol{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \cdots \\ x_d \end{pmatrix} \in R^d$$

- The sigmoid function takes a single argument (note, $\boldsymbol{w}^T \boldsymbol{x}$ is one number)

(^) Literally exactly same thing as linear regression from other lecture just with the sigmoid function

Meaning of the sigmoid function

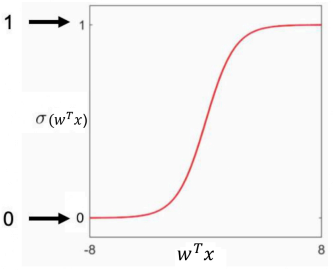
- It always returns a value between 0 and 1. The meaning of this value is the probability that the label is 1.

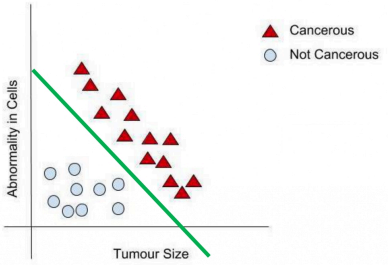
$$h = \sigma(\boldsymbol{w}^T\boldsymbol{x}) = P(y = 1|\boldsymbol{x}; \boldsymbol{w})$$

  If it is smaller than 0.5, then we predict label 0.

  If it is larger than 0.5, then we predict label 1.

- There is a slim chance that the sigmoid outputs exactly 0.5. The set of all possible inputs for which this happens is called the decision boundary.

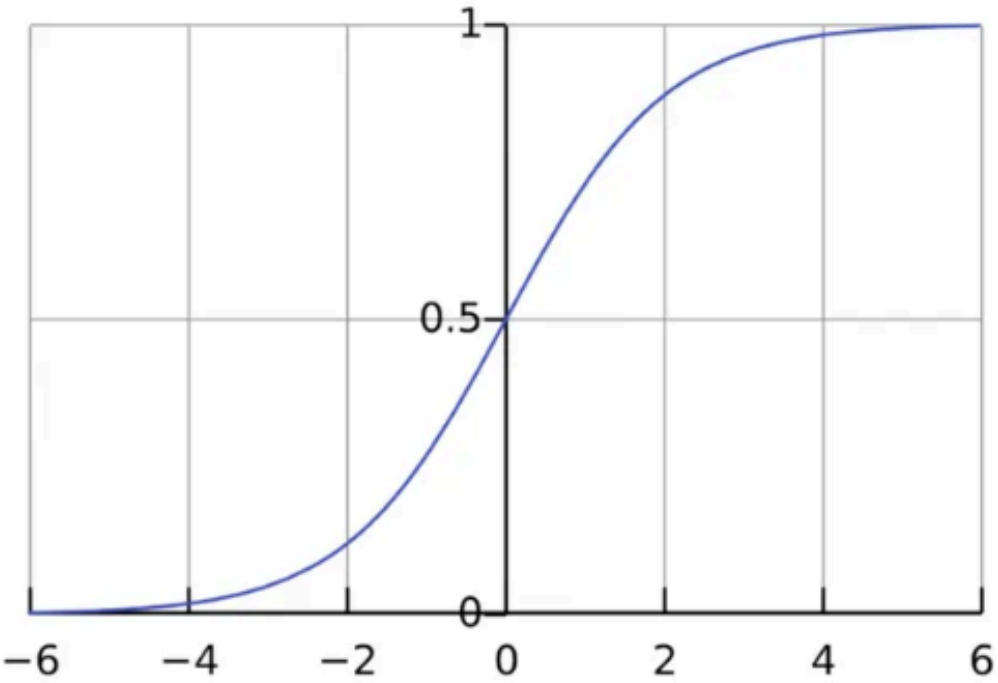




Decision boundary

And what is a"label"?

Because the sigmoid function is for logistic problems which tackles classification problems, the labels represent categories for the discrete data



Is the decision boundary of logistic regression always linear?

Loss function

- The loss expresses an error, so it must be always non-negative.
- Absolute value loss (L1 loss):

$$L1 = |f(x) - y|$$

- Mean squared error loss (L2 loss):

$$L2 = (f(x) - y)^2$$

$$g(w_0, w_1) = \frac{1}{N}\sum_{n=1}^{N}(f(x^{(n)}; w_0, w_1) - y^{(n)})^2$$

Empirical loss used by LR

Loss for the n-th training example

- 0/1 loss:

$$L_{0/1} = 0 \; if \; f(x) = y, else \; 1$$

UNIVERSITY OF BIRMINGHAM

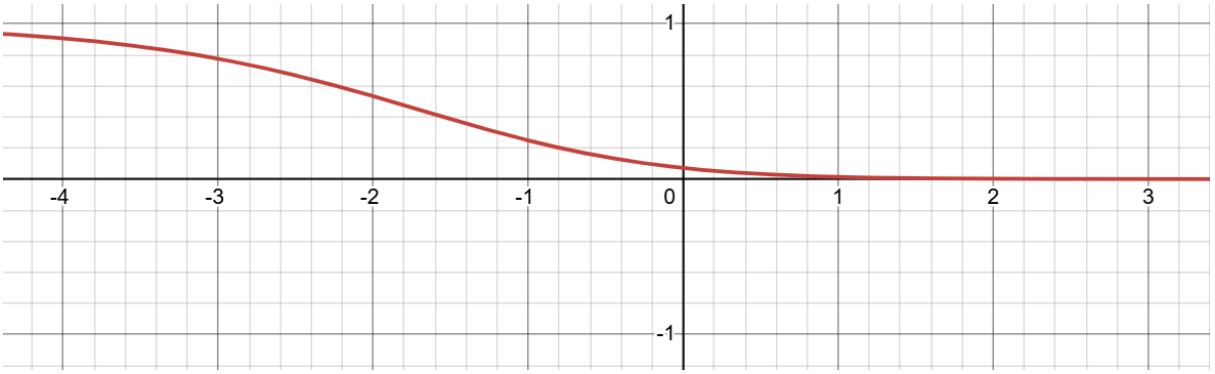
Cost function

- MSE using sigmoid function does not work.
  - The MSE function becomes non-convex – bounded sigmoid output between (0,1).
  - Gradient descent does not work well on non-convex functions. (local minimum, vanishing gradients)
  - Go to https://www.desmos.com/calculator, and observe: when label = 1,

$$y = \left(1 - \frac{1}{1 + e^{-(w_0 + w_1 x)}}\right)^2$$

- We need a new cost function.
  - Each data point contributes a cost, and the overall cost function is the average of these.
  - The cost is a function of the free parameters of the model.

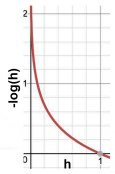
MSE = Mean Squared Error



Logistic cost function

y=ln x 

For each $(\boldsymbol{x}, y)$ pair, $cost(h(\boldsymbol{x};\boldsymbol{w}), y) = \begin{cases} -\log\big(h(\boldsymbol{x};\boldsymbol{w})\big), if\ y = 1 \\ -\log\big(1 - h(\boldsymbol{x};\boldsymbol{w})\big), if\ y = 0 \end{cases}$



Overall cost:
$g(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N} cost(h(\boldsymbol{x}^{(n)};\boldsymbol{w}), y^{(n)})$
Convex (easy to minimize)

Write the cost function in a single line

- $g(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N} cost(h(\boldsymbol{x}^{(n)};\boldsymbol{w}), y^{(n)})$, where
$$cost(h(\boldsymbol{x};\boldsymbol{w}), y) = \begin{cases} -\log\big(h(\boldsymbol{x};\boldsymbol{w})\big), if\ y = 1 \\ -\log\big(1 - h(\boldsymbol{x};\boldsymbol{w})\big), if\ y = 0 \end{cases}$$
- $g(\boldsymbol{w}) = -\frac{1}{N}\sum_{n=1}^{N}(y^{(n)} \log h(\boldsymbol{x}^{(n)};\boldsymbol{w}) + (1 - y^{(n)}) \log(1 - h(\boldsymbol{x}^{(n)};\boldsymbol{w})))$
This logistic loss is also called **cross-entropy** loss.

$g(\boldsymbol{w}) = -\frac{1}{N}\sum_{n=1}^{N}(y^{(n)} \log h(\boldsymbol{x}^{(n)};\boldsymbol{w}) + (1 - y^{(n)}) \log(1 - h(\boldsymbol{x}^{(n)};\boldsymbol{w})))$
This logistic loss is also called **cross-entropy** loss.

- ***When the actual value $y^{(n)}$ is 1 then the second part of the function is "ignored"***
  - ***It just becomes 0***
    - ***Effectively negating it and making it not contribute to the sum.***
- ***And when the actual value $y^{(n)}$ is 0 then the first part becomes 0***
  - ***So that part instead is effectively negated.***
  - ***This allows the function to go from 2 lines/ 2 parts to one line.***

Logistic regression – what we want to do

- Given training data
$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)}), where\ y \in \{0,1\}$$
- Fit the model
$$y = h(\boldsymbol{x};\boldsymbol{w}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$$
- By minimizing the cross-entropy cost function
$$g(\boldsymbol{w}) = -\frac{1}{N}\sum_{n=1}^{N}(y^{(n)} \log h(\boldsymbol{x}^{(n)};\boldsymbol{w}) + (1 - y^{(n)}) \log(1 - h(\boldsymbol{x}^{(n)};\boldsymbol{w})))$$

Training by gradient descent

- We use gradient descent (again like linear regression) to minimize the cost function, i.e. to find the best weight values $\boldsymbol{w}$.
- The gradient vector is*:

$$\nabla g(\boldsymbol{w}) = -\big(y^{(n)} - h(\boldsymbol{x}^{(n)};\boldsymbol{w})\big)\,\boldsymbol{x}^{(n)}, where\ \boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_d \end{pmatrix}, \boldsymbol{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_d \end{pmatrix} \in R^d$$

- We plug this into the general gradient descent algorithm given last week.

* This follows after differentiating the cost function w.r.t.w –we omit the lengthy math!

$g(w) = -\frac{1}{N}\sum_{n=1}^{N} y^{(n)} \log(h(x^{(n)}; w)) + (1 - y^{(n)}) log(1 - h(x^{(n)}; w))$
$\Delta g(w)$:

$h(x^{(n)}; w) = \sigma(z) = \frac{1}{1+e^{-w^T x^{(n)}}}$
where $z = w^T x^{(n)}$

$\frac{d}{dz} log z = \frac{1}{z}$

$\frac{d}{dw} \log(h(x^{(n)}; w)) = \frac{1}{h(x^{(n)}; w)} * \frac{d}{dw} h(x^{(n)}; w)$

$\frac{d}{dw} \log(1 - h(x^{(n)}; w)) = \frac{1}{1-h(x^{(n)}; w)} * \frac{d}{dw}(1 - h(x^{(n)}; w))$
Since $\frac{d}{dw}(1 - h(x^{(n)}; w)) = -\frac{d}{dw} h(x^{(n)}; w)$,

$\frac{d}{dw} \log(1 - h(x^{(n)}; w)) = -\frac{1}{1-h(x^{(n)}; w)} * \frac{d}{dw} h(x^{(n)}; w)$

The function $h(x^{(n)}; w)$ is the sigmoid, so its derivative is:
Let $h(x^{(n)}; w) = \sigma(z)$
$\sigma(z) = \frac{1}{1+e^{-x}}$
$\sigma(z) = (1 + e^{-x})^{-1}$
$\Delta\sigma(z) = -1 * -e^{-x}(1 + e^{-x})^{-2}$
$\Delta\sigma(z) = e^{-x}(1 + e^{-x})^{-2}$
Now we need to write e^{-x} in terms of $\sigma(z)$
Rationalise $\sigma(z)$:
$\frac{1}{1+e^{-z}} * \frac{1-e^{-z}}{1-e^{-z}}$
$\therefore \frac{1-e^{-z}}{(1+e^{-z})(1-e^{-z})}$
$\therefore \frac{1-\sigma(z)}{\sigma(z)}$
$\therefore \Delta\sigma(z) = \frac{1-\sigma(z)}{\sigma(z)} * \sigma(z)^2$

$\therefore \frac{d}{dz}\sigma(z) = \sigma(z)(1 - \sigma(z))$

Since $z = w^T x^{(n)}$:

$$h(x^{(n)};w) = \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\frac{d}{dw}h(x^{(n)};w) = h(x^{(n)};w)(1-h(x^{(n)};w))x^{(n)}$$

Now we've dealt with everything separately, let's do the differentiation of the full thing

First term:

$$y^{(n)}\log(h(x^{(n)};w))$$

Derivative:

$$y^{(n)}\frac{h(x^{(n)};w)}{h(x^{(n)};w)}(1-h(x^{(n)};w))x^{(n)}$$

Second term:

$$(1-y^{(n)})\log(1-h(x^{(n)};w))$$

Derivative:

$$-(1-y^{(n)})\frac{h(x^{(n)};w)}{1-h(x^{(n)};w)}(1-h(x^{(n)};w))x^{(n)}$$

Full thing:

$$-\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)}\frac{h(x^{(n)};w)}{h(x^{(n)};w)}(1-h(x^{(n)};w))x^{(n)} - (1-y^{(n)})h(x^{(n)};w)\right]$$

Simplify:

$$-\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)}(1-h(x^{(n)};w))x^{(n)} - (1-y^{(n)})h(x^{(n)};w)\right]$$

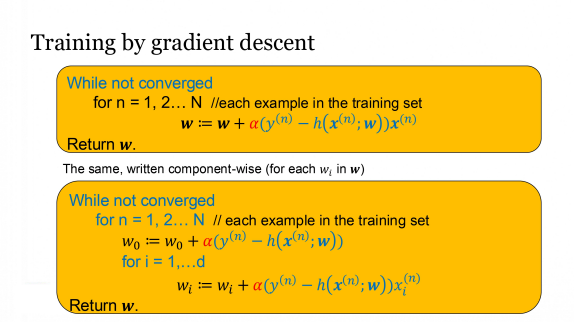
$$-\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)}x^{(n)} - y^{(n)}h(x^{(n)};w)x^{(n)} - h(x^{(n)};w)x^{(n)} + y^{(n)}h(x^{(n)};w)x^{(n)}\right]$$

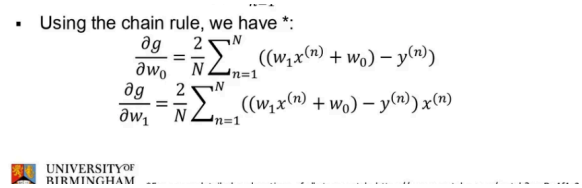
$$-\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)}x^{(n)} - h(x^{(n)};w)x^{(n)}\right]$$

$$-\frac{1}{N}\sum_{n=1}^{N}\left[(y^{(n)} - h(x^{(n)};w))x^{(n)}\right]$$

$\therefore$

$$\Delta g(w) = -(y^{(n)} - h(x^{(n)};w))x^{(n)}$$





This image shows the reason why the w_0 term doesn't have an $x_i^{(n)}$ next to it

No Free Lunch (NFL)

Simply to say:

- No single machine learning algorithm is universally the best-performing algorithm for all problems.

Theorem(Wolpert; also Hume 200 years ago):

- Given any distribution that generates the *x* of *S*, and any training set of size *N*. For any learner *A*,

$$\frac{1}{|F|}\sum_{f\in F} err(A(S) on\ task\ f) = \frac{1}{2}$$

Implication:

- If learner A1 is better than learner A2 for a task *f*, then there is another task g for which learner A2 is better than learner A1.
- So we need to know many learning methods & try them on the task at hand.

S: training set; h: a classifier; H: set of all classifiers; A: learner that chooses h from H based on S; f: task that A tries to learn; F: set of all possible tasks; err(h on task f): generalisation error.