UNIVERSITY OF
BIRMINGHAM

Week 3. Classification and Logistic Regression

Dr. Shuo Wang

Overview

- Classification: parametric and non-parametric
- Logistic regression algorithm
- K-Nearest Neighbours (kNN)

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, SVM, decision trees, etc.

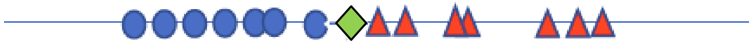
Logistic regression

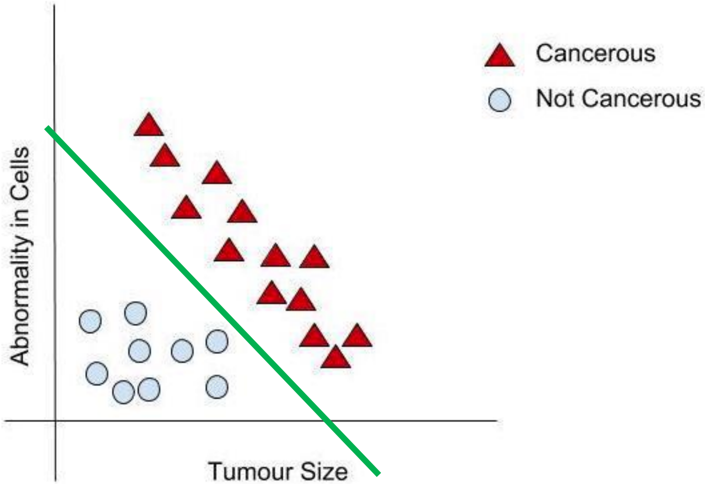
Similar to linear regression,

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

Model formulation

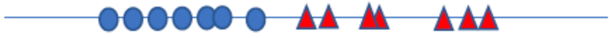
- We want to put a boundary between 2 classes.
- If x has a single attribute, we can do it with a point.
- If x has 2 attributes, we can do it with a line.
- If x has 3 attributes, we can do it with a plane.
- If x has more than 3 attributes, we can do it with a hyperplane (cannot draw it anymore).
- If the classes are linearly separable, the training error will be 0.

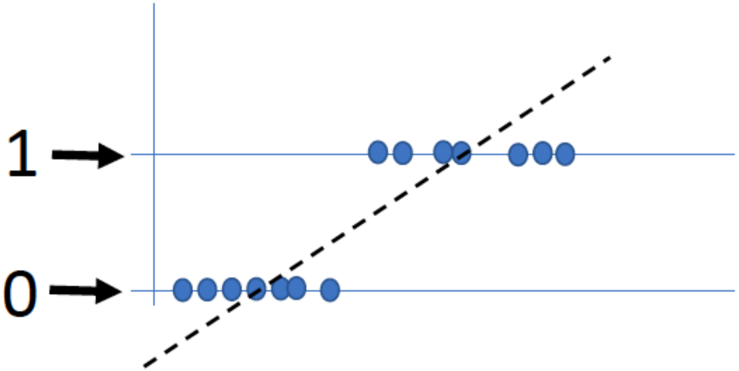




Decision boundary

Can we use linear regression to classify them?





Yes, but it might not perform very well. No ordering between categories, like there is between real numbers. We need a better model.

Model

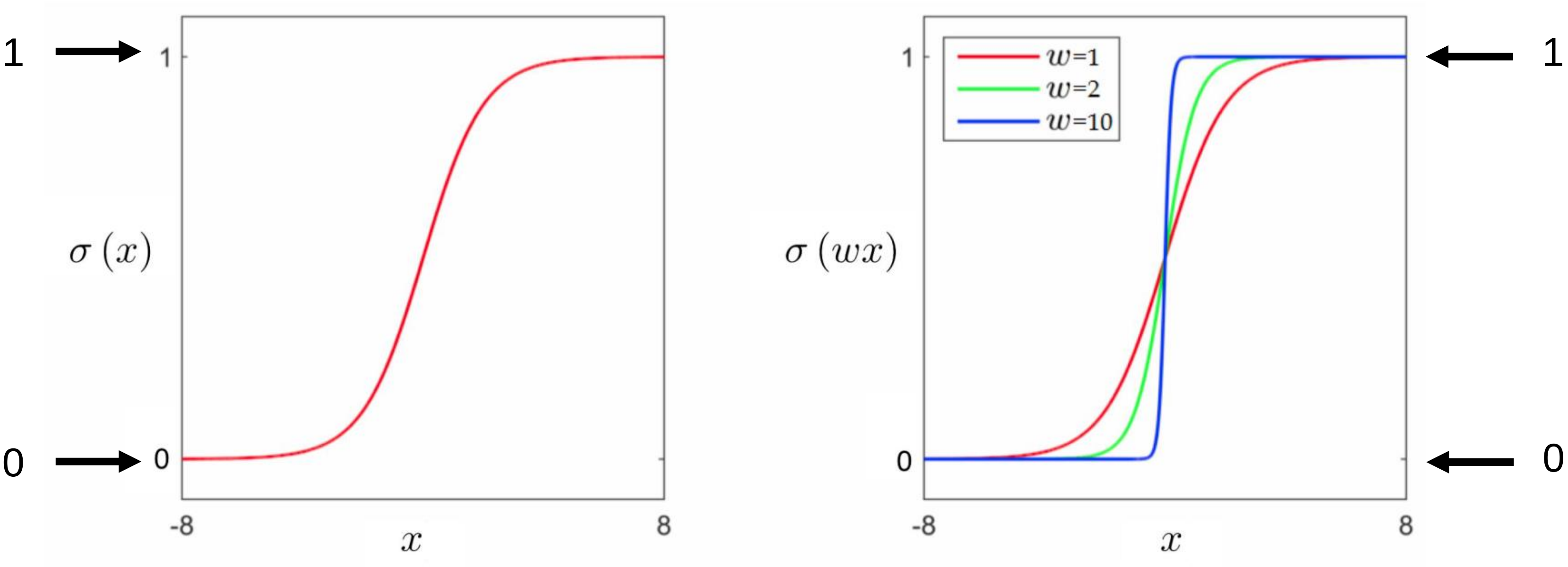
- We change the linear model slightly by passing it through a nonlinearity.
- If x has 1 attribute, we have:

$$h(x;\boldsymbol{w}) = \sigma(w_0 + w_1 x) = \frac{1}{1 + e^{-(w_0+w_1x)}}$$

- 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))

Sigmoid function



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

Model

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

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

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

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

Meaning of the sigmoid function

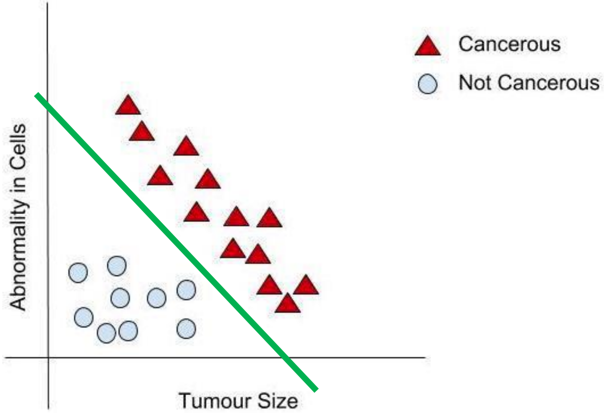
- The sigmoid function takes a single argument (note, $\boldsymbol{w}^T\boldsymbol{x}$ is one number)
- It always returns a value between 0 and 1. The meaning of this value is the probability that the label is 1.

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

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

Check your understanding

- Can you express the probability that the label is 0 using sigmoid?

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

$$\Longrightarrow 1 - \sigma(\boldsymbol{w}^T\boldsymbol{x}) = 1 - P(y = 1|\boldsymbol{x}; \boldsymbol{w}) = P(y = 0|\boldsymbol{x}; \boldsymbol{w})$$

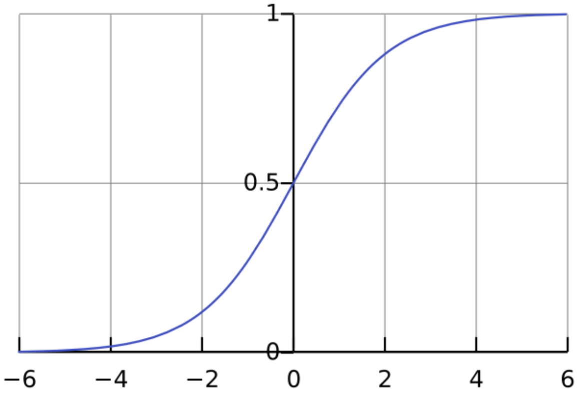
- In fact, we can write both in 1 line as:

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

y given x has a Bernoulli distribution.

Example

We are squeezing a linear function



- Suppose we have 2 input attributes, so our model is

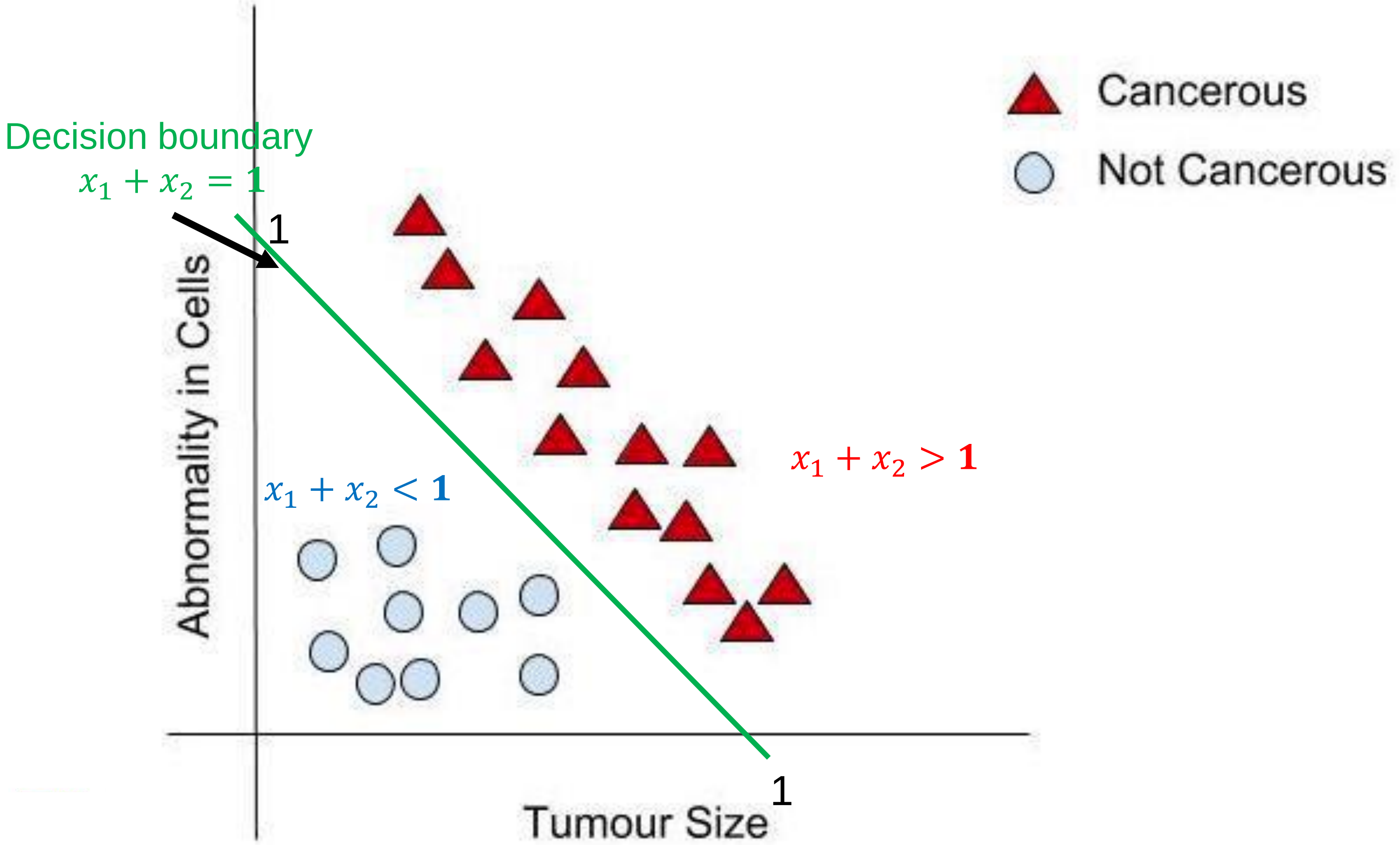
$$h(\boldsymbol{x};\boldsymbol{w}) = \boxed{\sigma(w_0 + w_1 x_1 + w_2 x_2)}$$

- Suppose we know that $w_0 = -1$, $w_1 = 1$, $w_2 = 1$.
- Q1: When do we predict 1? What is the decision boundary?

  We predict 1 precisely when $P(y = 1|\boldsymbol{x};\boldsymbol{w}) > 0.5$, which is $h(\boldsymbol{x};\boldsymbol{w}) > 0.5$.

  This is when $w_0 + w_1 x_1 + w_2 x_2 > 0$. (in this case, $-1 + x_1 + x_2$>0).

  Decision boundary is $w_0 + w_1 x_1 + w_2 x_2 = 0$. This is a line $-1 + x_1 + x_2$=0.



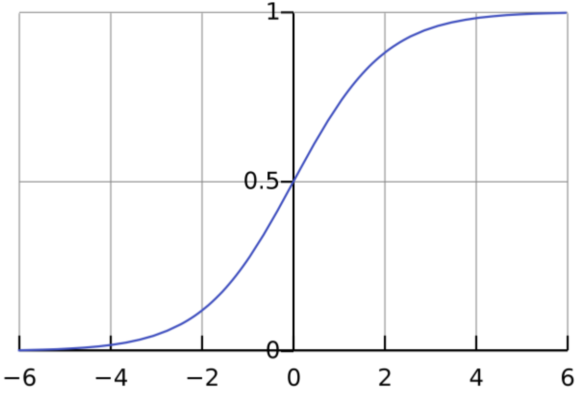
Decision boundary

$x_1 + x_2 = 1$

$x_1 + x_2 < 1$

$x_1 + x_2 > 1$

Example



- Suppose we have 2 input attributes, so our model is

$$h(\boldsymbol{x}; \boldsymbol{w}) = \sigma(w_0 + w_1 x_1 + w_2 x_2)$$

- Suppose we know that $w_0 = -1$, $w_1 = 1$, $w_2 = 1$.
- Q1: When do we predict 1? What is the decision boundary?

  We predict 1 precisely when $P(y = 1|\boldsymbol{x}; \boldsymbol{w}) > 0.5$, which is $h(\boldsymbol{x}; \boldsymbol{w}) > 0.5$.

  This is when $w_0 + w_1 x_1 + w_2 x_2 > 0$. (in this case, $-1 + x_1 + x_2$>0).

  Decision boundary is $w_0 + w_1 x_1 + w_2 x_2 = 0$. This is a line $-1 + x_1 + x_2$=0.

- Q2: Is the decision boundary of logistic regression always linear?

  Answer: yes.

Logistic regression

Similar to linear regression,

1) Model formulation

2) Cost function

3) Learning algorithm by gradient descent

Cost function - Recall

- 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): *used in linear regression*

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

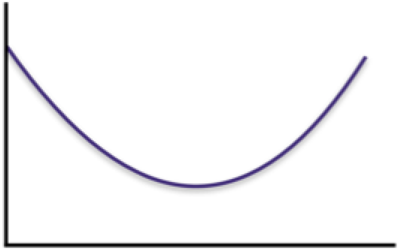
$$\boxed{g(w_0, w_1) = \frac{1}{N}\sum_{n=1}^{N} \underbrace{\left(f\left(x^{(n)}; w_0, w_1\right) - y^{(n)}\right)^2}_{\text{Loss for the n-th training example}}}$$

Empirical loss used by LR

- 0/1 loss:

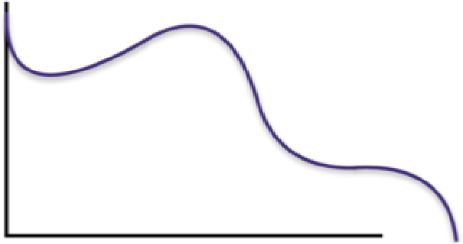
$$L_{0/1} = 0 \ if \ f(x) = y, else\ 1$$

Cost function



- MSE using sigmoid function does not work.
  - The MSE function becomes concave(not convex) – too wriggly (due to discrete output labels and bounded sigmoid output between (0,1).
  - Gradient descent does not work well on non-convex functions. (local minimum)
  - Go to https://www.desmos.com/calculator, and observe:

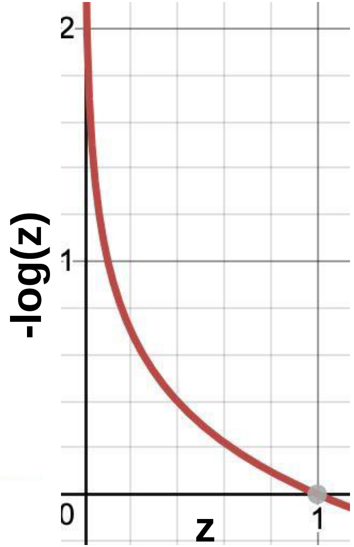
$$y=\left(1-\frac{1}{1+e^{-\left(w_0+w_1x\right)}}\right)^2$$

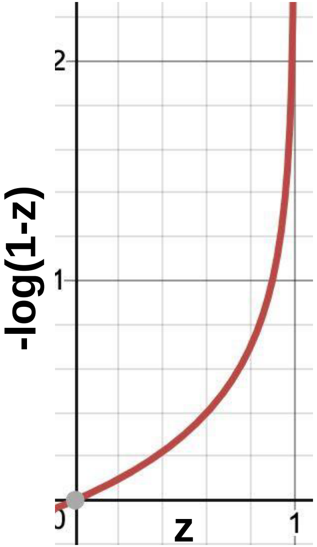


- We need a new cost function.
  - Each data point contributes a cost, and the overall cost function is the average of these.
  - The cost is a function of the free parameters of the model.

Logistic cost function

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





Overall cost:

$$g(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N} cost\big(h\big(\boldsymbol{x}^{(n)};\boldsymbol{w}\big), y^{(n)}\big)$$

Convex (easy to minimize)

Write the cost function in a single line

- $g(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N} cost\left(h\left(\boldsymbol{x}^{(n)};\boldsymbol{w}\right), y^{(n)}\right)$, where

$$cost(h(\boldsymbol{x};\boldsymbol{w}), y) = \begin{cases} -\log\left(h(\boldsymbol{x};\boldsymbol{w})\right), if\ y = 1 \\ -\log\left(1 - h(\boldsymbol{x};\boldsymbol{w})\right), if\ y = 0 \end{cases}$$

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

  This logistic loss is also called **cross-entropy** loss.

Logistic regression – what we want to do

- Given training data

$$\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \dots, \left(x^{(N)}, y^{(N)}\right), where\ y \in \{0,1\}$$

- Fit the model

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

- By minimizing the cross-entropy cost function

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

Logistic regression

Similar to linear regression,

1) Model formulation

2) Cost function

3) Learning algorithm by gradient descent

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*:

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

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

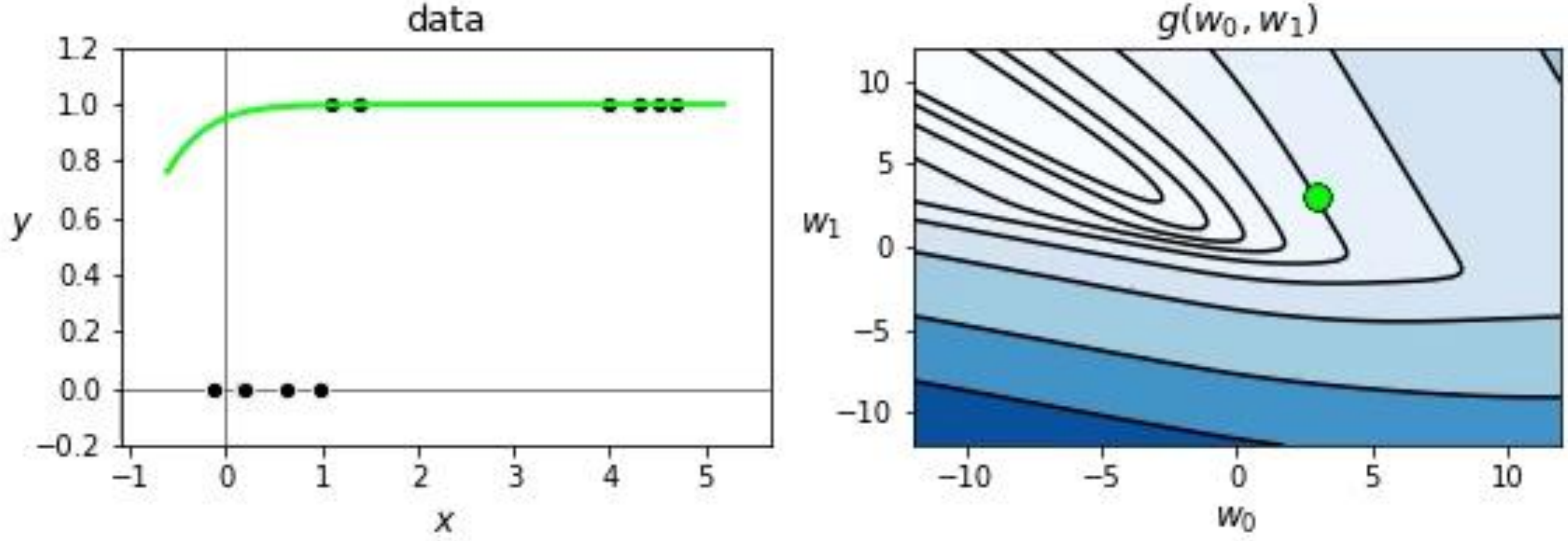
* This follows after differentiating the cost function w.r.t.w –we omit the lengthy math!

Training by gradient descent

```
While not converged
    for n = 1, 2… N  //each example in the training set
        $\boldsymbol{w} \coloneqq \boldsymbol{w} + \alpha\left(y^{(n)} - h\left(\boldsymbol{x}^{(n)}; \boldsymbol{w}\right)\right)\boldsymbol{x}^{(n)}$
Return $\boldsymbol{w}$.
```

The same, written component-wise

```
While not converged
    for n = 1, 2… N  // each example in the training set
        $w_0 \coloneqq w_0 + \alpha\left(y^{(n)} - h\left(\boldsymbol{x}^{(n)}; \boldsymbol{w}\right)\right)$
        for i = 1,…d
            $w_i \coloneqq w_i + \alpha\left(y^{(n)} - h\left(\boldsymbol{x}^{(n)}; \boldsymbol{w}\right)\right)x_i^{(n)}$
Return $\boldsymbol{w}$.
```


data
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
y
−1
0
1
2
3
4
5
x
$g(w_0, w_1)$
10
5
0
−5
−10
w_1
−10
−5
0
5
10
w_0

Extensions

- We studied logistic regression for linear binary classification
- There are extensions, such as:
    - Nonlinear logistic regression: instead of linear function inside the exp in the sigmoid, we can use polynomial functions of the input attributes
    - Multi-class logistic regression: uses a multi-valued version of sigmoid
- Details of these extensions are beyond of our scope in this module.

Examples of application of logistic regression

- Face detection: classes consist of images that contain a face and images without a face

- Sentiment analysis: classes consist of written product-reviews expressing a positive or a negative opinion

- Automatic diagnosis of medical conditions: classes consist of medical data of patients who either do or do not have a specific disease

Are we done?

- Logistic regression is not the only classifier.
- There are many others: decision trees, KNN, neural networks, SVM, etc.
- Which one is the best?

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.

Summing up

- Logistic regression is one classification approach
- It assumes a linear class-separation boundary.
- There are many other approaches (from following weeks and other modules).
- None of them can be best on all problems! = No Free Lunch