# Information Retrieval & Data Mining [COMP0084]

*Introduction to machine learning*

Vasileios Lampos

Computer Science, UCL

# Preliminaries

- In this lecture:
  - Data mining; association rule mining (apriori algorithm)
  - Introduction to machine learning; supervised learning (regression, classification), unsupervised learning (clustering) with examples

- Useful additional reads:
  - Chapters 2, 4 of "*Web Data Mining*" by Bing Liu (2006) — cs.uic.edu/~liub/WebMiningBook.html
  - Chapters 3, 4, 14 of "*The Elements of Statistical Learning*" by Hastie, Tibshirani, and Friedman (2008) — hastie.su.domains/ElemStatLearn/
  - Chapter 5 of "*Speech and language processing*" (SLP) by Jurafsky and Martin (2021) — web.stanford.edu/~jurafsky/slp3/
  - More advanced reading: Paper on estimating influenza prevalence based on Web search activity by Lampos, Miller et al. — nature.com/articles/srep12760

- Some slides were adapted from Bing Liu's course — cs.uic.edu/~liub/teach/cs583-fall-21/cs583.html

# Data mining — Definition

- **Data mining** is the process of discovering (*mining*) useful patterns from or conducting inferences based on various types of *data* sources such as structured information repositories (e.g. databases), text, images, sound, video, and so on.
- **Multi-disciplinary**: machine learning (or AI more broadly), statistics, databases, information retrieval — *but the distinction between machine learning and data mining is becoming increasingly difficult, especially from an applications perspective.*
- Strong research community: Knowledge Discovery and Data Mining or **KDD** — kdd.org
- Why? Gaining knowledge from a database is not as simple as conducting database queries
- Applications include marketing, recommendations, scientific data analysis, and *any task involving large amounts of data*

# Data mining — Association rule mining

- Today: a quick look into **Association rule mining** / **learning** — *perhaps the most important task proposed and studied by the data mining community*
- Introduced by Agrawal, Imielinski, and Swami in 1993 — dl.acm.org/doi/pdf/10.1145/170035.170072
- Applicable on categorical / discrete data (e.g. product categories, movies, songs)
- Just a good, old algorithm! No machine learning involved here...
- Initially used for market basket analysis to understand how products purchased by customers are related, e.g.

*spaghetti* → *basil* [**support** = 0.1%, **confidence** = 25%]

# Association rule mining — Notation & definitions

**market basket transactions**

$t_1$ : {almonds, cashews, pistachios}

$t_2$ : {almonds, bananas}

$\dots$

$t_n$ : {cashews, oranges, pistachios}

# Association rule mining — Notation & definitions

**market basket transactions**

$t_1$ : {almonds, cashews, pistachios}
$t_2$ : {almonds, bananas}
$\ldots$
$t_n$ : {cashews, oranges, pistachios}

- A set of all the $m$ **items**, $I = \{i_1, i_2, \ldots, i_m\}$
  — e.g. "almonds" is an item

# Association rule mining — Notation & definitions

**market basket transactions**

$t_1$ : {almonds, cashews, pistachios}
$t_2$ : {almonds, bananas}
$\dots$
$t_n$ : {cashews, oranges, pistachios}

- A set of all the $m$ **items**, $I = \{i_1, i_2, \dots, i_m\}$
  — e.g. "almonds" is an item
- A set of all the $n$ **transactions**, $T = \{t_1, t_2, \dots, t_n\}$

# Association rule mining — Notation & definitions

**market basket transactions**

$t_1$ : {almonds, cashews, pistachios}
$t_2$ : {almonds, bananas}
$\ldots$
$t_n$ : {cashews, oranges, pistachios}

- A set of all the $m$ **items**, $I = \{i_1, i_2, \ldots, i_m\}$
  — e.g. "almonds" is an item
- A set of all the $n$ **transactions**, $T = \{t_1, t_2, \ldots, t_n\}$
- A transaction $t_i$ is a set of items, and hence $t_i \subseteq I$

# Association rule mining — Notation & definitions

**market basket transactions**

$t_1$ : {almonds, cashews, pistachios}

$t_2$ : {almonds, bananas}

$\dots$

$t_n$ : {cashews, oranges, pistachios}

# Association rule mining — Notation & definitions

**market basket transactions**

$t_1$ : {almonds, cashews, pistachios}
$t_2$ : {almonds, bananas}
$\ldots$
$t_n$ : {cashews, oranges, pistachios}

- An **itemset** is a set of items
  – e.g. $X = \{\text{almonds, cashews}\}$

# Association rule mining — Notation & definitions

**market basket transactions**

$t_1$ : {almonds, cashews, pistachios}
$t_2$ : {almonds, bananas}
$\dots$
$t_n$ : {cashews, oranges, pistachios}

- An **itemset** is a set of items
  – e.g. $X = \{\text{almonds}, \text{ cashews}\}$
- A $k$-**itemset** is an itemset with $k$ items
  – e.g. $X = \{\text{almonds}, \text{ cashews}, \text{ pistachios}\}$ is a 3-itemset

# Association rule mining — Notation & definitions

**market basket transactions**

$t_1$ : {almonds, cashews, pistachios}
$t_2$ : {almonds, bananas}
$\ldots$
$t_n$ : {cashews, oranges, pistachios}

- An **itemset** is a set of items
  – e.g. $X = \{\text{almonds}, \text{cashews}\}$
- A $k$-**itemset** is an itemset with $k$ items
  – e.g. $X = \{\text{almonds}, \text{cashews}, \text{pistachios}\}$ is a 3-itemset
- A transaction $t_i$ contains **itemset** $X$ if $X \subseteq t_i$

# Association rule mining — Notation & definitions

**market basket transactions**

| |
|---|
| $t_1$ : {almonds, cashews, pistachios} |
| $t_2$ : {almonds, bananas} |
| $\dots$ |
| $t_n$ : {cashews, oranges, pistachios} |

- An **itemset** is a set of items
  – e.g. $X = \{\text{almonds}, \text{ cashews}\}$
- A $k$-**itemset** is an itemset with $k$ items
  – e.g. $X = \{\text{almonds}, \text{ cashews}, \text{ pistachios}\}$ is a 3-itemset
- A transaction $t_i$ contains **itemset** $X$ if $X \subseteq t_i$
- An **association rule** between itemsets $X, Y$ is an implication of the form:

$$X \rightarrow Y, \text{ where } X, Y \subset I, \text{ and } X \cap Y = \emptyset$$

# Association rule mining — Support & confidence

- **Association rule**: $X \rightarrow Y$
  - a pattern present in our data that we want to "mine" $\Longrightarrow$ **data mining**
  - when $X$ occurs, $Y$ occurs with a certain *support* and *confidence*

- **support** $= \dfrac{(X \cup Y)\,.\,\text{count}}{n}$
  - $n$ transactions
  - support $= \Pr(X \cup Y) =$ probability that a transaction contains both itemsets $X$ and $Y$
  - how many times $X$ and $Y$ appear together in all $(n)$ transactions in $T$ divided by $n$

# Association rule mining — Support & confidence

- Association rule: $X \to Y$
  – when $X$ occurs, $Y$ occurs with a certain *support* and *confidence*

- **support** $= \dfrac{(X \cup Y)\,.\,\text{count}}{n} = \Pr(X \cup Y)$

- **confidence** $= \dfrac{(X \cup Y)\,.\,\text{count}}{X\,.\,\text{count}}$
  – confidence $= \Pr(Y \mid X) =$ conditional probability that a transaction that contains $X$ will also contain $Y$
  – how many times a transaction that contains $X$ also contains $Y$ divided by the number of transactions that contain $X$

- **Association rule mining goal:** Find all association rules $(X \rightarrow Y)$ that satisfy a pre-specified (*by us!*) **minimum support** (also abbreviated as **`minsup`**) and **minimum confidence** (**`minconf`**)

- Key properties for this data mining task
  – **Completeness**, i.e. we need to identify all possible rules
    Note that $X \rightarrow Y$ and $Y \rightarrow X$ are different rules. **Why?**
  – Mining with data on hard disk (*because it is not always feasible to load everything in memory*)

# Association rule mining — An example

$t_1$ : {almonds, cashews, pistachios}
$t_2$ : {almonds, bananas}
$t_3$ : {apples, bananas}
$t_4$ : {almonds, bananas, cashews}
$t_5$ : {almonds, bananas, cashews, oranges, pistachios}
$t_6$ : {cashews, oranges, pistachios}
$t_7$ : {cashews, oranges, pistachios}

# Association rule mining — An example

- Toy database with 7 supermarket transactions

$t_1$ : {almonds, cashews, pistachios}
$t_2$ : {almonds, bananas}
$t_3$ : {apples, bananas}
$t_4$ : {almonds, bananas, cashews}
$t_5$ : {almonds, bananas, cashews, oranges, pistachios}
$t_6$ : {cashews, oranges, pistachios}
$t_7$ : {cashews, oranges, pistachios}

# Association rule mining — An example

- Toy database with 7 supermarket transactions
- Let's set our association rule mining goals:
  — **`minsup`** = 30% and **`minconf`** = 80%

$t_1$ : {almonds, cashews, pistachios}
$t_2$ : {almonds, bananas}
$t_3$ : {apples, bananas}
$t_4$ : {almonds, bananas, cashews}
$t_5$ : {almonds, bananas, cashews, oranges, pistachios}
$t_6$ : {cashews, oranges, pistachios}
$t_7$ : {cashews, oranges, pistachios}

# Association rule mining — An example

- Toy database with 7 supermarket transactions
- Let's set our association rule mining goals:
  — **`minsup`** = 30% and **`minconf`** = 80%
- **Frequent itemset** examples:
  — {almonds, cashews} with support 3/7 (> **`minsup`**)
  — {cashews, pistachios} with support 4/7
  — {cashews, oranges, pistachios} with support 3/7

$t_1$ : {almonds, cashews, pistachios}
$t_2$ : {almonds, bananas}
$t_3$ : {apples, bananas}
$t_4$ : {almonds, bananas, cashews}
$t_5$ : {almonds, bananas, cashews, oranges, pistachios}
$t_6$ : {cashews, oranges, pistachios}
$t_7$ : {cashews, oranges, pistachios}

# Association rule mining — An example

- Toy database with 7 supermarket transactions
- Let's set our association rule mining goals:
  — **`minsup`** = 30% and **`minconf`** = 80%

$t_1$ : {almonds, cashews, pistachios}
$t_2$ : {almonds, bananas}
$t_3$ : {apples, bananas}
$t_4$ : {almonds, bananas, cashews}
$t_5$ : {almonds, bananas, cashews, oranges, pistachios}
$t_6$ : {cashews, oranges, pistachios}
$t_7$ : {cashews, oranges, pistachios}

- **Frequent itemset** examples:
  — {almonds, cashews} with support 3/7 (> **`minsup`**)
  — {cashews, pistachios} with support 4/7
  — {cashews, oranges, pistachios} with support 3/7
- **Association rule candidates** from the above frequent itemsets
  — almonds → cashews has a confidence of 3/4 (< **`minconf`**, *rejected*)
  — pistachios → cashews has a confidence of 4/4 (> **`minconf`**, *accepted*)
  — {cashews, oranges} → pistachios has a confidence of 3/3 (> **`minconf`**, *accepted*)

# Association rule mining — Algorithms

- Large number of different association rule mining algorithms deploying different strategies to solve this task
- Algorithms can differ in their computational efficiency, data structures that are required, memory requirements
- But their **output can only be the same**:
  - Given a transaction data set $T$, **`minsup`**, and **`minconf`**, the set of association rules in $T$ is uniquely determined.
- Foundational algorithm for association rule mining: **Apriori**

# Association rule mining — Apriori

- **Apriori** is perhaps the most popular algorithm in data mining
- "*Apriori*" $\Longrightarrow$ because it uses "*prior*" knowledge of frequent itemsets
- Proposed by Agrawal and Srikant in 1994 — vldb.org/conf/1994/P487.pdf (> 30,000 citations)
- Apriori is a 2-step algorithm:
  - first, find all the itemsets with a minimum support (a.k.a. *frequent itemsets*) in a database of transactions
  - then, use the identified frequent itemsets to generate association rules

# Apriori — Identify frequent itemsets

# Apriori — Identify frequent itemsets

- The key idea of Apriori is the **downward closure property** or commonly also referred to as the “**Apriori property**”:
  – Any subset of a frequent itemset is also a frequent itemset
  = Any subset of an itemset whose support is ≥ **`minsup`** has also support ≥ **`minsup`**

# Apriori — Identify frequent itemsets

- The key idea of Apriori is the **downward closure property** or commonly also referred to as the "**Apriori property**":

  — Any subset of a frequent itemset is also a frequent itemset

  = Any subset of an itemset whose support is ≥ **`minsup`** has also support ≥ **`minsup`**

- If the itemset {a, b, c, d} with 4 items is frequent, then its $\left(2^4 - 2\right) = 14$ non-empty sub-itemsets (*subsets*) will also be frequent.

  Just for clarity, these are: {a}, {b}, {c}, {d}, {a, b}, {a, c}, {a, d}, {b, c}, {b, d}, {c, d}, {a, b, c}, {a, b, d}, {a, c, d}, and {b, c, d}

# Apriori — Identify frequent itemsets

- The key idea of Apriori is the **downward closure property** or commonly also referred to as the "**Apriori property**":
  — Any subset of a frequent itemset is also a frequent itemset
  = Any subset of an itemset whose support is ≥ **`minsup`** has also support ≥ **`minsup`**

- If the itemset {a, b, c, d} with 4 items is frequent, then its $\left(2^4 - 2\right) = 14$ non-empty sub-itemsets (*subsets*) will also be frequent.
  Just for clarity, these are: {a}, {b}, {c}, {d}, {a, b}, {a, c}, {a, d}, {b, c}, {b, d}, {c, d}, {a, b, c}, {a, b, d}, {a, c, d}, and {b, c, d}

- **Contraposition:** Reversing the above property, if an itemset is **not frequent**, then any of its **supersets cannot be frequent** since they will be containing something that is not frequent

# Apriori — The gist of the algorithm

- Apriori is an iterative algorithm
  - given a minimum support
  - find all frequent 1-itemsets (denoted by `F[1]` in the source code**)
  - use those to find all frequent 2-itemsets, and so on
    > `C[2]` is a list of frequent 2-itemset candidates based on `F[1]`
    > `F[2]` $\subseteq$ `C[2]` is a list with the frequent 2-itemsets
  - **key algorithmic principle**: in each iteration $k$ of the algorithm only consider itemsets that contain some frequent $(k-1)$-itemset

*** might be useful while going through the slides*

# Apriori — An important detail (*item ordering*)

- Items should always be sorted according to a sorting scheme — i.e. lexicographic order
- This order will be used throughout the algorithm as it helps to reduce redundant passes and comparisons on the data
- For example, the frequent itemset {a, b, c, d} is identical to the frequent itemsets {c, d, a, b} or {b, a, d, c} — we only need to deal with {a, b, c, d} once!

# Apriori — Pseudocode of the algorithm (*part 1*)

```
% T: all the transactions, MINSUP: frequent itemset minimum support
function apriori(T, MINSUP):
  % C[1] count of 1-itemsets, n transactions in T
  C[1], n ← initial-pass(T)
  % F[1] is the set of frequent 1-itemsets
  F[1] ← {f | f in C[1] AND f.count/n ≥ MINSUP}
  for k = 2; F[k-1] ≠ ∅; k++:
    % use the (k-1)—itemsets to generate k-itemset candidates, C[k]
    C[k] ← generate-candidates(F[k-1])
    for each transaction t in T:
      for each candidate c in C[k]:
        if c is in t:
          c.count++
    F[k] ← {c in C[k] | c.count/n ≥ MINSUP}

  return F
```

# Apriori — Pseudocode of the algorithm (*part 1*)

main function

```
% T: all the transactions, MINSUP: frequent itemset minimum support
function apriori(T, MINSUP):
  % C[1] count of 1-itemsets, n transactions in T
  C[1], n ← initial-pass(T)
  % F[1] is the set of frequent 1-itemsets
  F[1] ← {f | f in C[1] AND f.count/n ≥ MINSUP}
  for k = 2; F[k-1] ≠ ∅; k++:
    % use the (k-1)—itemsets to generate k-itemset candidates, C[k]
    C[k] ← generate-candidates(F[k-1])
    for each transaction t in T:
      for each candidate c in C[k]:
        if c is in t:
          c.count++
    F[k] ← {c in C[k] | c.count/n ≥ MINSUP}

  return F
```

subroutine

# Apriori — Candidate itemset generation

- The **`generate-candidates`** function takes the $(k-1)$-frequent itemsets, denoted by `F[k-1]` in the source code, and returns a superset of $k$-frequent itemset candidates, denoted by `C[k]`

- It implements the following two operations:
  - **Join**: generate all possible candidate $k$-itemsets, `C[k]`, based on the $(k-1)$-frequent itemsets, `F[k-1]`
  - **Prune**: remove those candidates in `C[k]` that cannot be frequent, i.e. if a candidate itemset has a subset of items that is not already identified as a frequent itemset it should be removed

# Apriori — Pseudocode of the algorithm (*part 2*)

```
% using frequent (k-1)-itemsets generate frequent k-itemset candidates
function generate-candidates(F[k-1]):
  C[k] ← ∅
  for every f1, f2 in F[k-1] where:
    a = f1 - f2 AND                        % set difference
    b = f2 - f1 AND                        % set difference
    (a AND b) are both of size 1 AND       % f1 and f2 differ by 1 element
    a < b do:                              % lexicographic comparison
      c ← {f1,b}                           % frequent k-itemset candidate
      C[k] ← {C[k],c}                      % add c to candidate itemsets
      for each (k-1)-subset s of c do:
        if s not in F[k-1]:
          delete c from C[k]               % pruning non-frequent candidates

  return C[k]                              % return candidate itemsets
```

# Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

Let’s use Apriori to identify all frequent itemsets with minimum support of 30%

# Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

```
C[1]:{almonds:4/7, apples:1/7, bananas:4/7, cashews:5/7, oranges:3/7,
      pistachios:4/7}

F[1]:{almonds, bananas, cashews, oranges, pistachios}

C[2]:{ {almonds, bananas}:3/7,     {almonds, cashews}:3/7,
       {almonds, oranges}:1/7,     {almonds, pistachios}:2/7,
       {bananas, cashews}:2/7,     {bananas, oranges}:1/7,
       {bananas, pistachios}:1/7,  {cashews, oranges}:3/7,
       {cashews, pistachios}:4/7,  {oranges, pistachios}:3/7 }
```

# Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

```
C[2]:{ {almonds, bananas}:3/7,     {almonds, cashews}:3/7,
       {almonds, oranges}:1/7,     {almonds, pistachios}:2/7,
       {bananas, cashews}:2/7,     {bananas, oranges}:1/7,
       {bananas, pistachios}:1/7,  {cashews, oranges}:3/7,
       {cashews, pistachios}:4/7,  {oranges, pistachios}:3/7 }

F[2]:{ {almonds, bananas}, {almonds, cashews}, {cashews, oranges},
       {cashews, pistachios}, {oranges, pistachios} }
```

# Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

```
F[2]:{ {almonds, bananas}, {almonds, cashews}, {cashews, oranges},
       {cashews, pistachios}, {oranges, pistachios} }

C[3]:{ {almonds, bananas, cashews}:2/7,
       {cashews, oranges, pistachios}:3/7 }
```

# Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

```
F[2]:{ {almonds, bananas}, {almonds, cashews}, {cashews, oranges},
       {cashews, pistachios}, {oranges, pistachios} }

C[3]:{ {almonds, bananas, cashews}:2/7,
       {cashews, oranges, pistachios}:3/7 }
```

*** Incorrect ***

# Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

```
F[2]:{ {almonds, bananas}, {almonds, cashews}, {cashews, oranges},
       {cashews, pistachios}, {oranges, pistachios} }
```

~~`C[3]:{ {almonds, bananas, cashews}:2/7, {cashews, oranges, pistachios}:3/7 }`~~

*** Incorrect ***

```
C[3]:{ {cashews, oranges, pistachios}:3/7 }
```

entry `{almonds, bananas, cashews}` will be pruned because `{bananas, cashews}` is not in `F[2]`

```
F[3]:{ {cashews, oranges, pistachios} }
```

# Apriori — An example

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

Apriori identified the following *frequent* itemsets with a minimum support of 30%:

```
F[1]:{almonds:4/7, bananas:4/7, cashews:5/7, oranges:3/7, pistachios:4/7}

F[2]:{ {almonds, bananas}:3/7,     {almonds, cashews}:3/7,
       {cashews, oranges}:3/7,     {cashews, pistachios}:4/7,
       {oranges, pistachios}:3/7 }

F[3]:{ {cashews, oranges, pistachios}:3/7 }
```

# Apriori — Generating association rules from frequent itemsets

- Frequent itemsets do not directly provide association rules
- For each frequent itemset $F$
  For each non-empty subset $A$ of $F$ (*no repetitions*)
  - $B = F - A$
  - $A \to B$ is an association rule if confidence $(A \to B) \geq \texttt{minconf}$

$$\text{support}\,(A \to B) = \text{support}\,(A \cup B) = \text{support}\,(F)$$

$$\text{confidence}\,(A \to B) = \frac{\text{support}\,(A \cup B)}{\text{support}\,(A)}$$

# Apriori — Generating association rules (*example*)

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

**`minsup`** = 30%, **`minconf`** = 80%, let's use `F[3]:{ {cashews, oranges, pistachios}:3/7 }`

```
A = {{cashews, oranges}, {cashews, pistachios}, {oranges, pistachios},
     {cashews}, {oranges}, {pistachios}}
```

| A → B | | confidence |
|---|---|---|
| {cashews, oranges} | → pistachios | confidence = 1 |
| {cashews, pistachios} | → oranges | confidence = 0.75 |
| {oranges, pistachios} | → cashews | confidence = 1 |
| cashews | → {oranges, pistachios} | confidence = 0.6 |
| oranges | → {cashews, pistachios} | confidence = 1 |
| pistachios | → {cashews, oranges} | confidence = 0.75 |

# Apriori — Generating association rules (*example*)

```
t[1]: {almonds, cashews, pistachios}
t[2]: {almonds, bananas}
t[3]: {apples, bananas}
t[4]: {almonds, bananas, cashews}
t[5]: {almonds, bananas, cashews, oranges, pistachios}
t[6]: {cashews, oranges, pistachios}
t[7]: {cashews, oranges, pistachios}
```

**`minsup`** = 30%, **`minconf`** = 80%, let's use `F[3]:{ {cashews, oranges, pistachios}:3/7 }`

```
A = {{cashews, oranges}, {cashews, pistachios}, {oranges, pistachios},
     {cashews}, {oranges}, {pistachios}}
```

| A → B | | confidence |
|---|---|---|
| {cashews, oranges} | → pistachios | confidence = 1 |
| {cashews, pistachios} | → oranges | confidence = 0.75 |
| {oranges, pistachios} | → cashews | confidence = 1 |
| cashews | → {oranges, pistachios} | confidence = 0.6 |
| oranges | → {cashews, pistachios} | confidence = 1 |
| pistachios | → {cashews, oranges} | confidence = 0.75 |

# Apriori — Generating association rules from frequent itemsets

- To obtain an association rule $A \rightarrow B$, we need to compute the quantities: support $(A \cup B)$ and support $(A)$
- This information has already been recorded during itemset generation. Therefore, there is no need to access the raw transaction data any longer.
- Not as time consuming a frequent itemset generation, although there are efficient algorithms to generate association rules as well

# The (*very*) basics of machine learning

— supervised learning (*regression*, *classification*)
— unsupervised learning (*clustering*)

# Machine learning

- Arthur Samuel (IBM, 1959): "*Machine learning is the field of study that gives the computer the ability to learn* (**a task**) *without being explicitly programmed.*"
  — credited for coining the term
  — although we are still explicitly programming them to learn!

- Tom Mitchell (CMU, 1998): "A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$."
  — more formal definition
  — learning from experience (*observations*, *data*)

# Notational conventions for this lecture

$x \in \mathbb{R}$ denotes a real-valued scalar

$\mathbf{x} \in \mathbb{R}^n$ denotes a real-value vector with $n$ elements

$\mathbf{X} \in \mathbb{R}^{n \times m}$ denotes a real-valued matrix with $n$ rows and $m$ columns

$\mathbf{y} \in \mathbb{R}^m$ denotes $m$ instances of a real valued response (or target) variable

$\hat{\mathbf{y}} \in \mathbb{R}^m$ denotes $m$ inferences of a real valued response variable

$\|\mathbf{x}\|_k = \left( \sum_{i=1}^{n} |x_i|^k \right)^{\frac{1}{k}}$ denotes the $L_k$-norm $\in \mathbb{R}$ of $\mathbf{x}$

# Learning from experience

- Experience is something tangible, i.e. an observation and eventually a data point, something that can take a numeric form
- $\mathbf{x}_i$ denotes a numeric interpretation of an input
  $y_i$ denotes a numeric interpretation of an output

$< \mathbf{x}_i, y_i >$ is an observation / sample

# Learning from experience

- Experience is something tangible, i.e. an observation and eventually a data point, something that can take a numeric form
- $\mathbf{x}_i$ denotes a numeric interpretation of an input
  $y_i$ denotes a numeric interpretation of an output

$< \mathbf{x}_i, y_i >$ is an observation / sample

$$
\begin{array}{l}
< \mathbf{x}_1, y_1 > \\
< \mathbf{x}_2, y_2 > \\
\dots \\
< \mathbf{x}_n, y_n >
\end{array}
$$

# Learning from experience

- Experience is something tangible, i.e. an observation and eventually a data point, something that can take a numeric form
- $\mathbf{x}_i$ denotes a numeric interpretation of an input
  $y_i$ denotes a numeric interpretation of an output

$< \mathbf{x}_i, y_i >$ is an observation / sample

$$< \mathbf{x}_1, y_1 >$$
$$< \mathbf{x}_2, y_2 >$$
$$\dots$$
$$< \mathbf{x}_n, y_n >$$

training

# Learning from experience

- Experience is something tangible, i.e. an observation and eventually a data point, something that can take a numeric form
- $\mathbf{x}_i$ denotes a numeric interpretation of an input
  $y_i$ denotes a numeric interpretation of an output

$< \mathbf{x}_i, y_i >$ is an observation / sample

# Learning from experience

- Experience is something tangible, i.e. an observation and eventually a data point, something that can take a numeric form
- $\mathbf{x}_i$ denotes a numeric interpretation of an input
  $y_i$ denotes a numeric interpretation of an output

$< \mathbf{x}_i, y_i >$ is an observation / sample

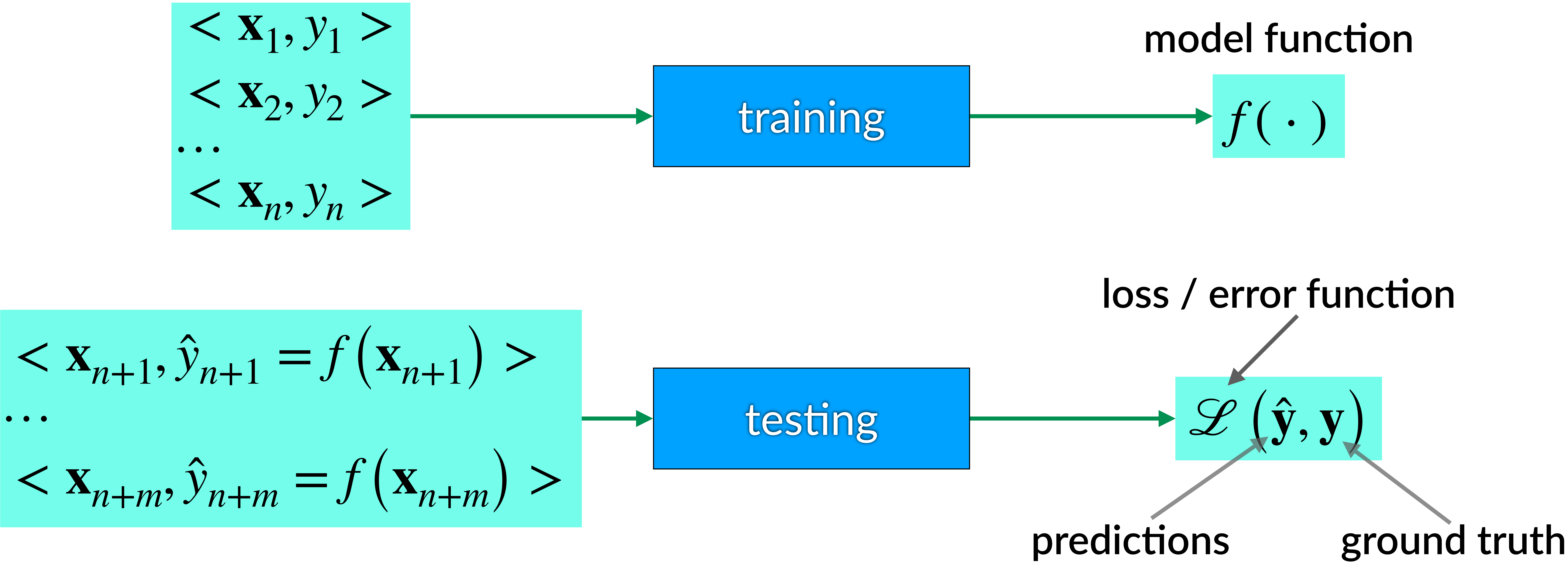
$$
\begin{array}{l}
< \mathbf{x}_1, y_1 > \\
< \mathbf{x}_2, y_2 > \\
\ldots \\
< \mathbf{x}_n, y_n >
\end{array}
\longrightarrow \boxed{\text{training}} \longrightarrow f(\cdot)
$$

model function

$$
\begin{array}{l}
< \mathbf{x}_{n+1}, \hat{y}_{n+1} = f\left(\mathbf{x}_{n+1}\right) > \\
\ldots \\
< \mathbf{x}_{n+m}, \hat{y}_{n+m} = f\left(\mathbf{x}_{n+m}\right) >
\end{array}
$$

- Experience is something tangible, i.e. an observation and eventually a data point, something that can take a numeric form
- $\mathbf{x}_i$ denotes a numeric interpretation of an input
  $y_i$ denotes a numeric interpretation of an output

$< \mathbf{x}_i, y_i >$ is an observation / sample



# Learning from experience

- If $\mathscr{L}\left(\hat{\mathbf{y}}, \mathbf{y}\right)$ is “relatively” small, then our model might be learning from experience
- But what makes an error “relatively” small? We need to have a solid reference loss value.

**Question**

We are classifying photos of cats and dogs. A classifier sees a photo of either a cat or a dog and makes a binary decision: does the photo show a cat or a dog? In total, the classifier classifies $1{,}000$ photos. It makes the correct classification $96\,\%$ of the times, i.e. it classifies correctly $960$ out of the $1{,}000$ photos. **Is the accuracy of the classifier good?**

- If $\mathscr{L}\left(\hat{\mathbf{y}}, \mathbf{y}\right)$ is “relatively” small, then our model might be learning from experience
- But what makes an error “relatively” small? We need to have a solid reference loss value.

**Question**

We are classifying photos of cats and dogs. A classifier sees a photo of either a cat or a dog and makes a binary decision: does the photo show a cat or a dog? In total, the classifier classifies $1{,}000$ photos. It makes the correct classification $96\,\%$ of the times, i.e. it classifies correctly $960$ out of the $1{,}000$ photos. **Is the accuracy of the classifier good?**

From these $1{,}000$ photos, we actually know that only $20$ are showing cats. ***Is the accuracy of classifier good?***

# Common machine learning categorisation

- **Supervised learning**
  Learn a mapping $f$ from inputs $\mathbf{X}$ to outputs $\mathbf{y}$ – also can be expressed by $f : \mathbf{X} \to \mathbf{y}$
  – $\mathbf{X}$ are also called features, observations, covariates, predictors
  – $\mathbf{y}$ are also called labels, targets, responses, ground truth
  – $< \mathbf{X}, \mathbf{y} >$ can also be referred to as observations or samples

- **Unsupervised learning**
  No outputs associated with the input $\mathbf{X}$ – the task becomes to discover an underlying structure or patterns in $\mathbf{X}$

- **Reinforcement learning**
  The system or agent has to learn how to interact with its environment
  Policy: which action to take in response to an input $\mathbf{X}$
  Different from supervised learning because no definitive responses are given
  Only rewards – *learning with a critic as opposed to learning with a teacher*

# Common machine learning categorisation

- **Supervised learning**
  Learn a mapping $f$ from inputs $\mathbf{X}$ to outputs $\mathbf{y}$ – also can be expressed by $f : \mathbf{X} \to \mathbf{y}$
  – $\mathbf{X}$ are also called features, observations, covariates, predictors
  – $\mathbf{y}$ are also called labels, targets, responses, ground truth
  – $< \mathbf{X}, \mathbf{y} >$ can also be referred to as observations or samples
- **Unsupervised learning**
  No outputs associated with the input $\mathbf{X}$ – the task becomes to discover an underlying structure or patterns in $\mathbf{X}$
- **Reinforcement learning**
  The system or agent has to learn how to interact with its environment
  Policy: which action to take in response to an input $\mathbf{X}$
  Different from supervised learning because no definitive responses are given
  Only rewards – *learning with a critic as opposed to learning with a teacher*

# Supervised learning

- **Regression**
  estimate / predict a continuous output / target variable

  i.e. learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \mathbb{R}^{n}$

  Examples: predict a time series trend (finance, climate, etc.), estimate the prevalence of an infectious disease disease in epidemiology

- **Classification**
  estimate a set of $C$ unordered (and mutually exclusive) labels / classes

  i.e. learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \{1, 2, \ \dots, C\}$

  Examples: detect spam email, medical imaging, text classification, language models

- **Regression**
  estimate / predict a continuous output / target variable

  i.e. learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \mathbb{R}^{n}$

  Examples: predict a time series trend (finance, climate, etc.), estimate the prevalence of an infectious disease disease in epidemiology

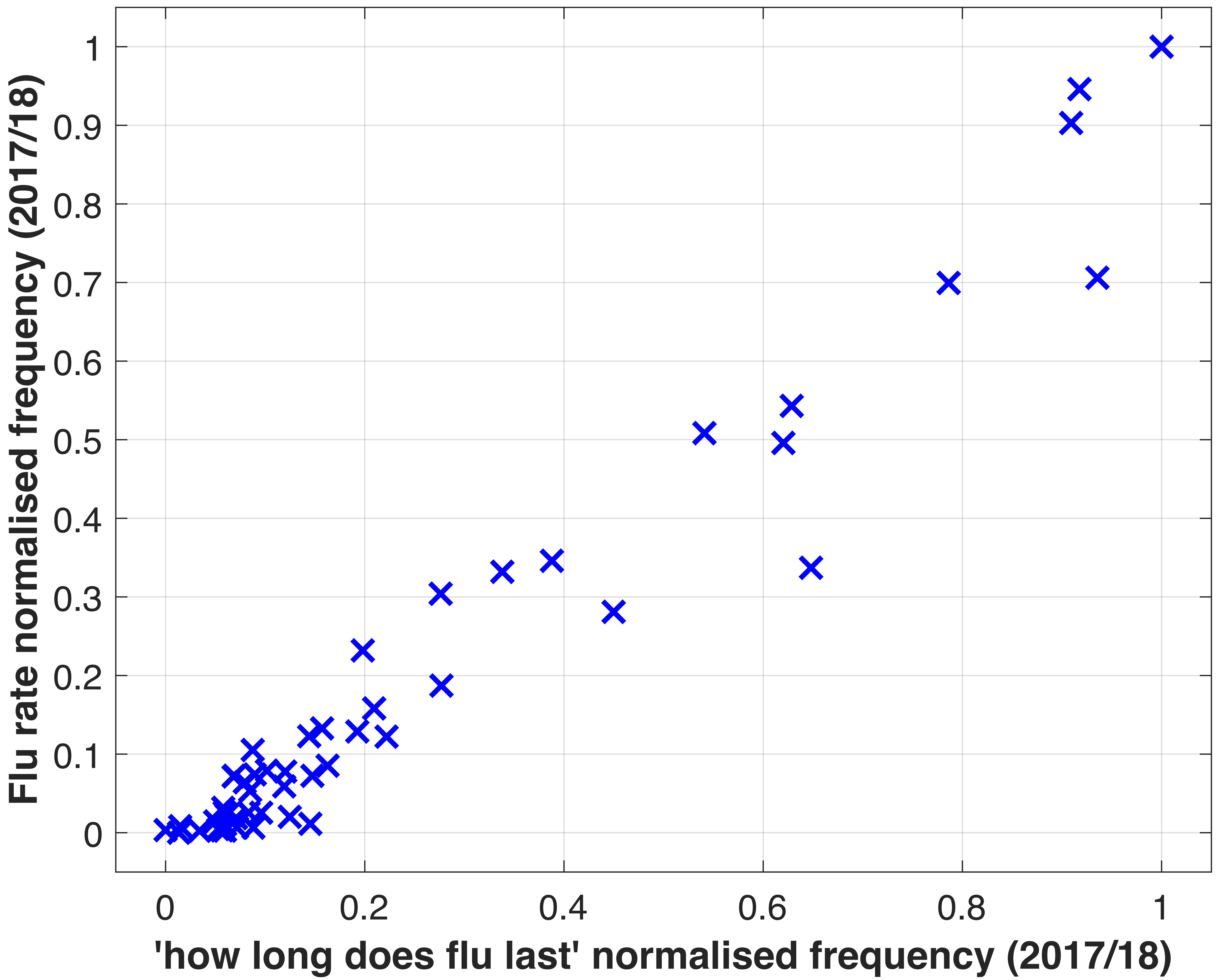
- **Classification**
  estimate a set of $C$ unordered (and mutually exclusive) labels / classes

  i.e. learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \{1, 2, \ldots, C\}$

  Examples: detect spam email, medical imaging, text classification, language models

# Supervised learning — Regression

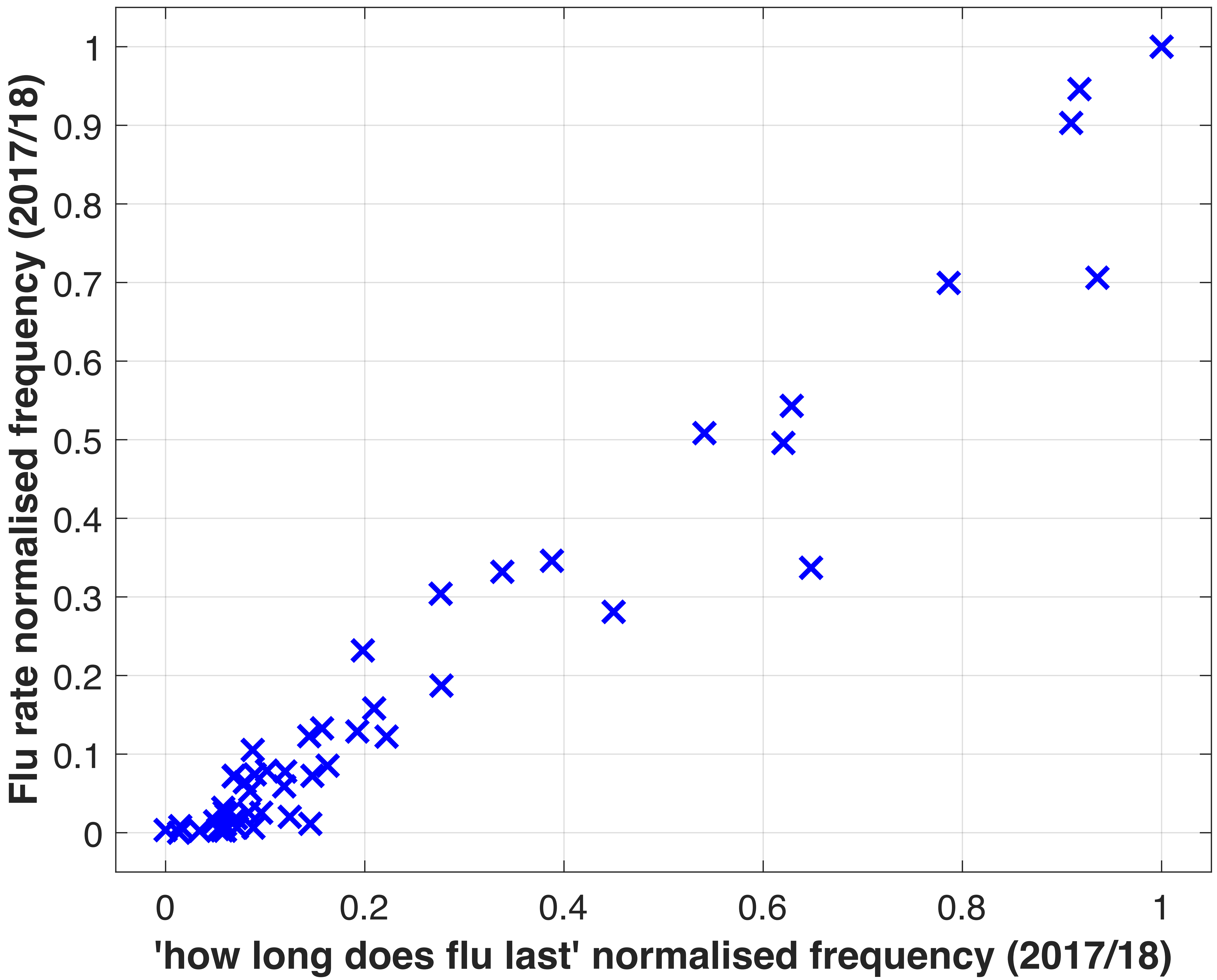
- Estimate the **prevalence of influenza-like illness** in England based on the **frequency of the search query** “*how long does flu last*”



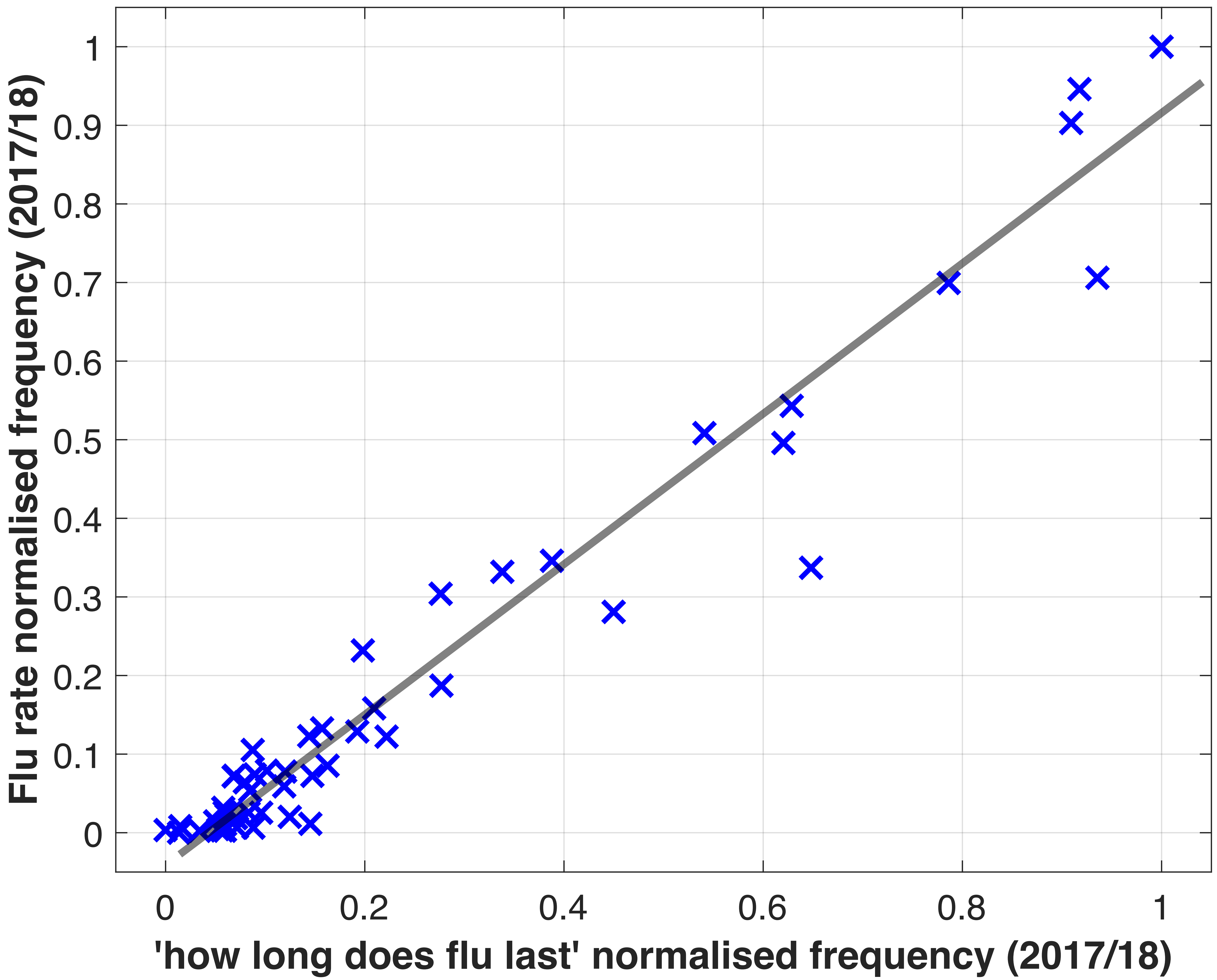

**Data:** dropbox.com/s/rgyg190whw26qrj/data-COMP0084-intro-to-ml.zip?dl=0

# Supervised learning — Regression

- Estimate the **prevalence of influenza-like illness** in England based on the **frequency of the search query** “*how long does flu last*”
- Linearly related, bivariate correlation of $0.975$.
  Question: *What is the maximum possible correlation?*



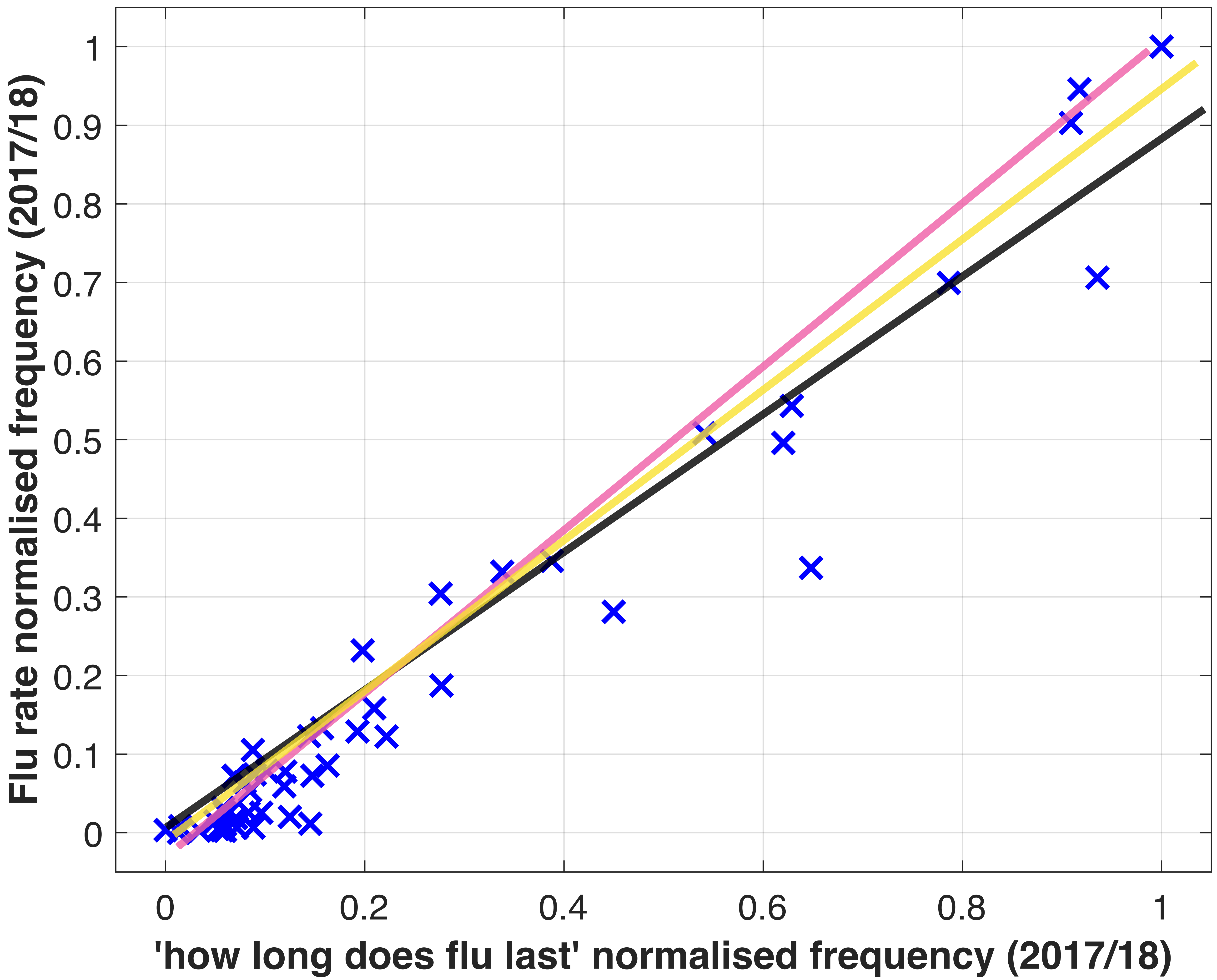
- Estimate the **prevalence of influenza-like illness** in England based on the **frequency of the search query** “*how long does flu last*”
- Linearly related, bivariate correlation of $0.975$.
  Question: *What is the maximum possible correlation?*
- Can we capture this relationship with a straight line? We should be able to.




**Data:** dropbox.com/s/rgyg190whw26qrj/data-COMP0084-intro-to-ml.zip?dl=0

# Supervised learning — Regression

- Estimate the **prevalence of influenza-like illness** in England based on the **frequency of the search query** "*how long does flu last*"
- Linearly related, bivariate correlation of $0.975$.
  Question: *What is the maximum possible correlation?*
- Can we capture this relationship with a straight line? We should be able to.
- Which line is the "best" though?




**Data:** dropbox.com/s/rgyg190whw26qrj/data-COMP0084-intro-to-ml.zip?dl=0

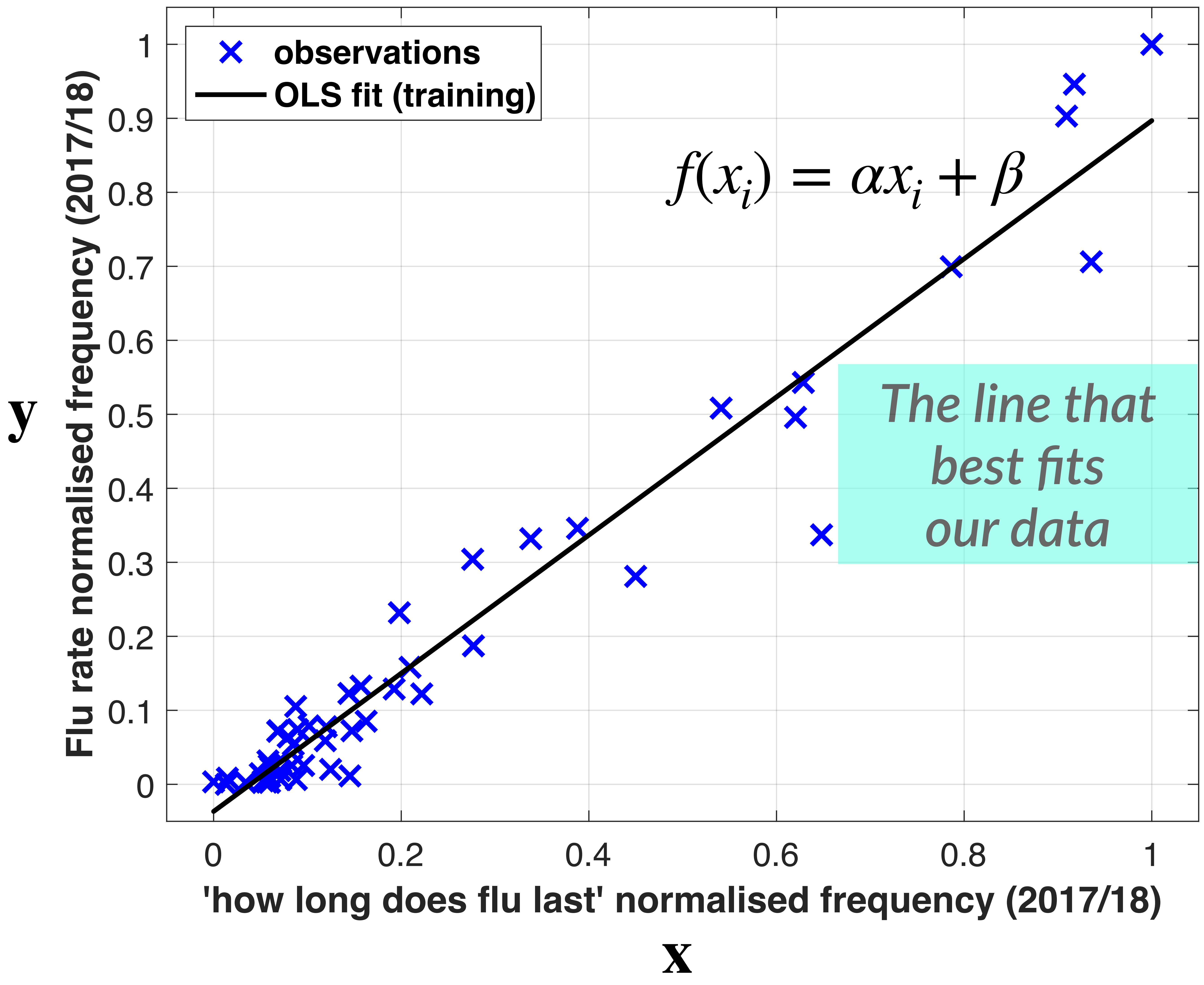
# Supervised learning — Ordinary least squares (*linear*) regression

Let's recap and provide a few more details about our regression task:

- our prediction target / response variable $\mathbf{y}$ denotes the weekly influenza-like illness prevalence in England from September 2017 until the end of August 2018
- our input or observation $\mathbf{x}$ denotes the corresponding weekly frequency of the search query "how long does flu last" (Google) for the same time period and location
- We want to learn a linear mapping $f$ from the input $\mathbf{x}$ to the output $\mathbf{y}$ based on our current observations, i.e. for a weekly query frequency $x_i$, $f(x_i) = \hat{y}_i = \alpha x_i + \beta \approx y_i$
- This linear mapping has two unknown hyper-parameters: $\{\alpha, \beta\}$
- Find a line that best fits to our observations

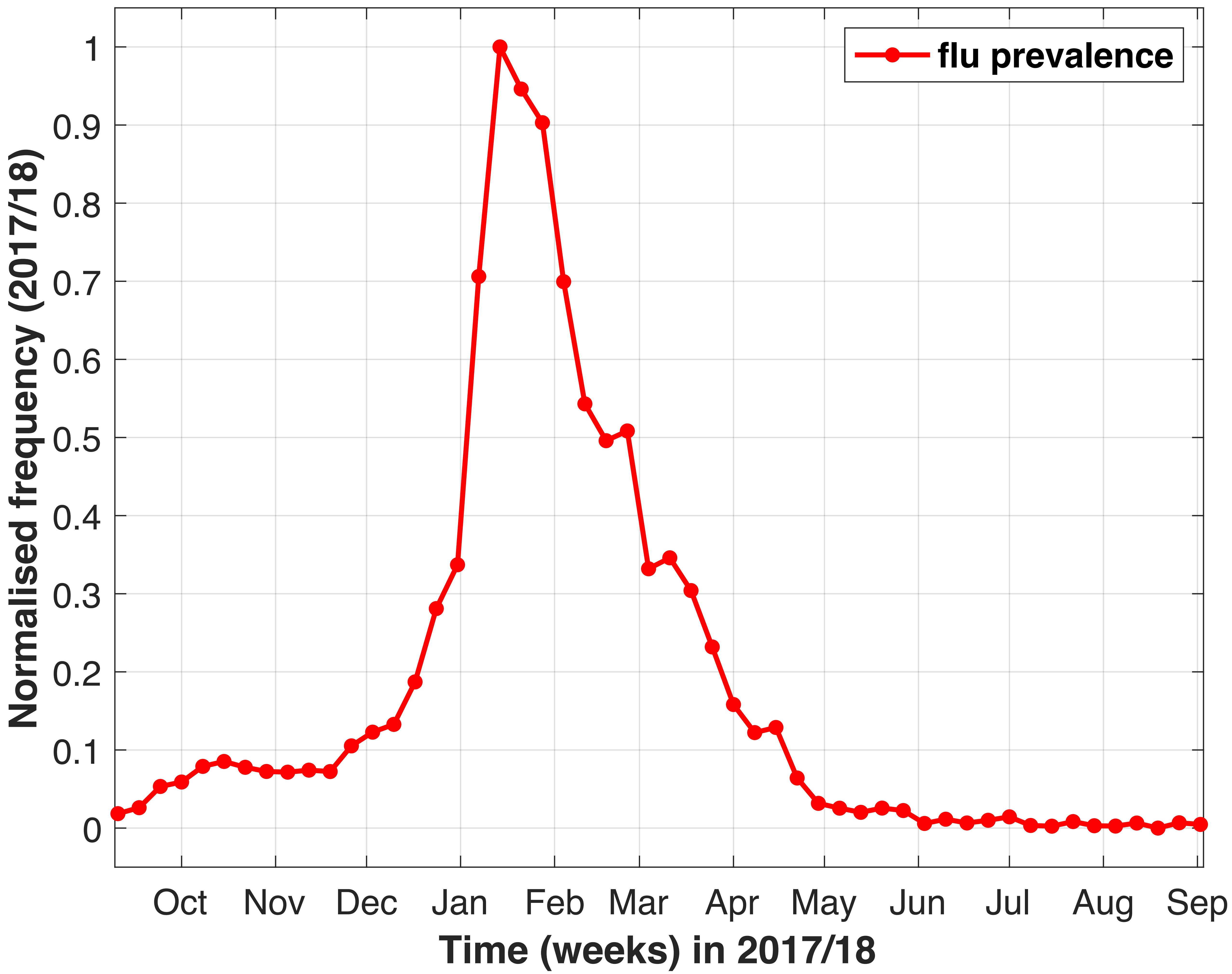
# Supervised learning — Ordinary least squares (OLS; *linear*) regression

- Estimate the **prevalence of influenza-like illness** in England based on the **frequency of the search query** "*how long does flu last*"
- Linearly related, bivariate correlation of $0.975$.
  Question: *What is the maximum possible correlation?*
- Can we capture this relationship with a straight line? We should be able to.
- Which line is the "best" though?



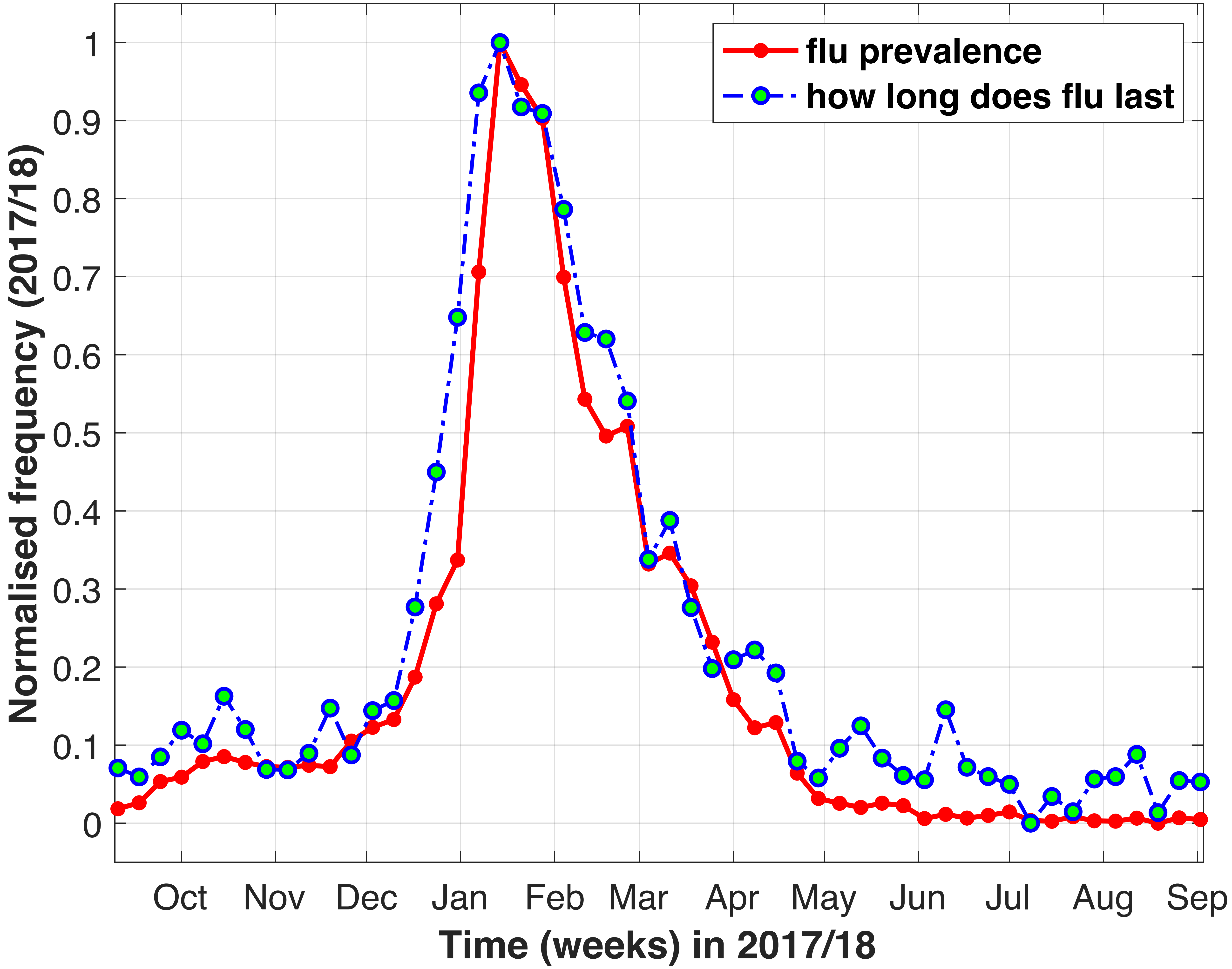
# Supervised learning — OLS regression, *alternative point of view*

- $\mathbf{y}$ ~ weekly flu prevalence



# Supervised learning — OLS regression, *alternative point of view*

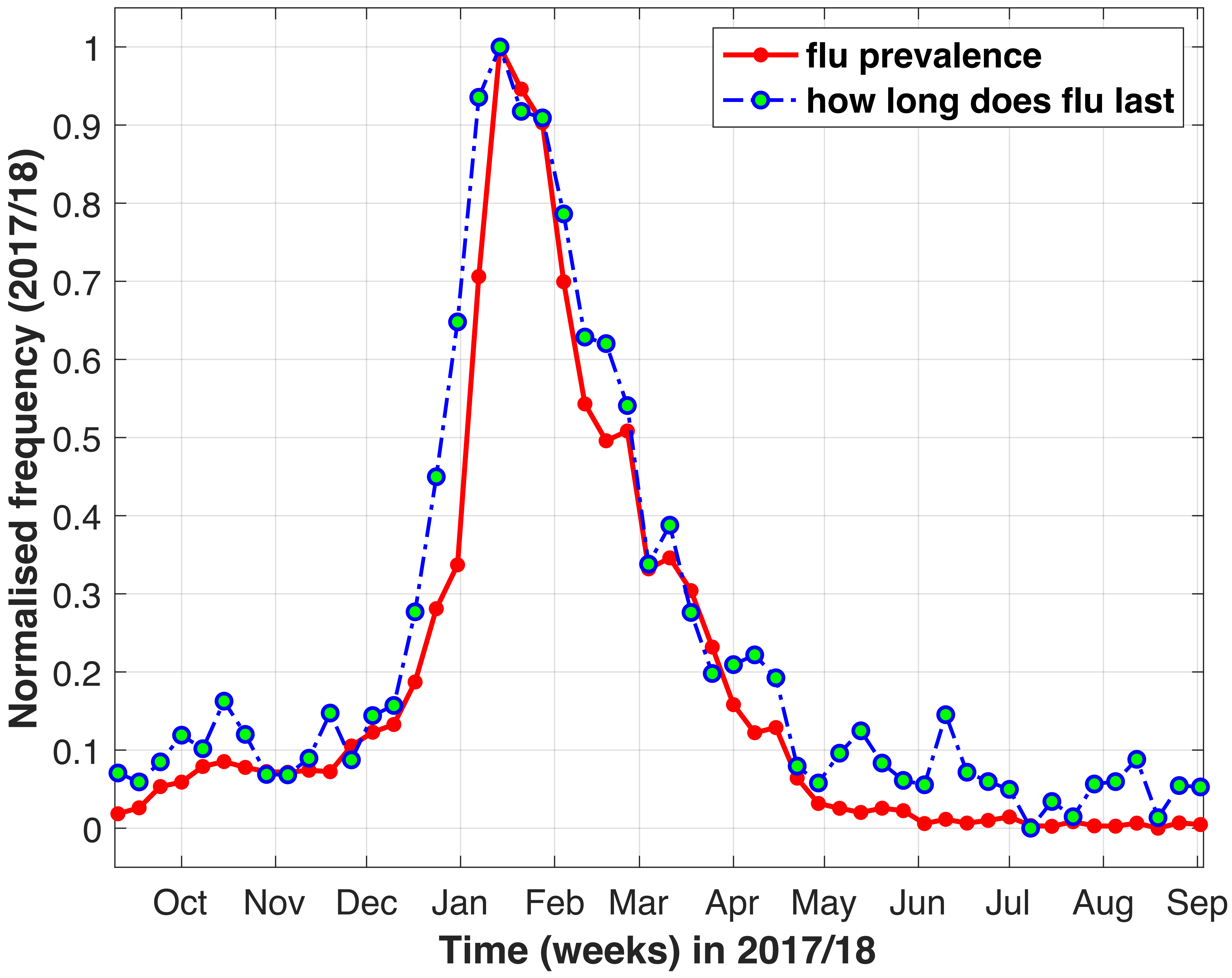
- $\mathbf{y}$ ~ weekly flu prevalence
- $\mathbf{x}$ ~ weekly search frequency of "*how long does flu last*"



# Supervised learning — OLS regression, *alternative point of view*

- $\mathbf{y}$ ~ weekly flu prevalence
- $\mathbf{x}$ ~ weekly search frequency of "*how long does flu last*"
- $f : \mathbf{x} \rightarrow \mathbf{y}$ such that
  $f(x_i) = \hat{y}_i = \alpha x_i + \beta \approx y_i$

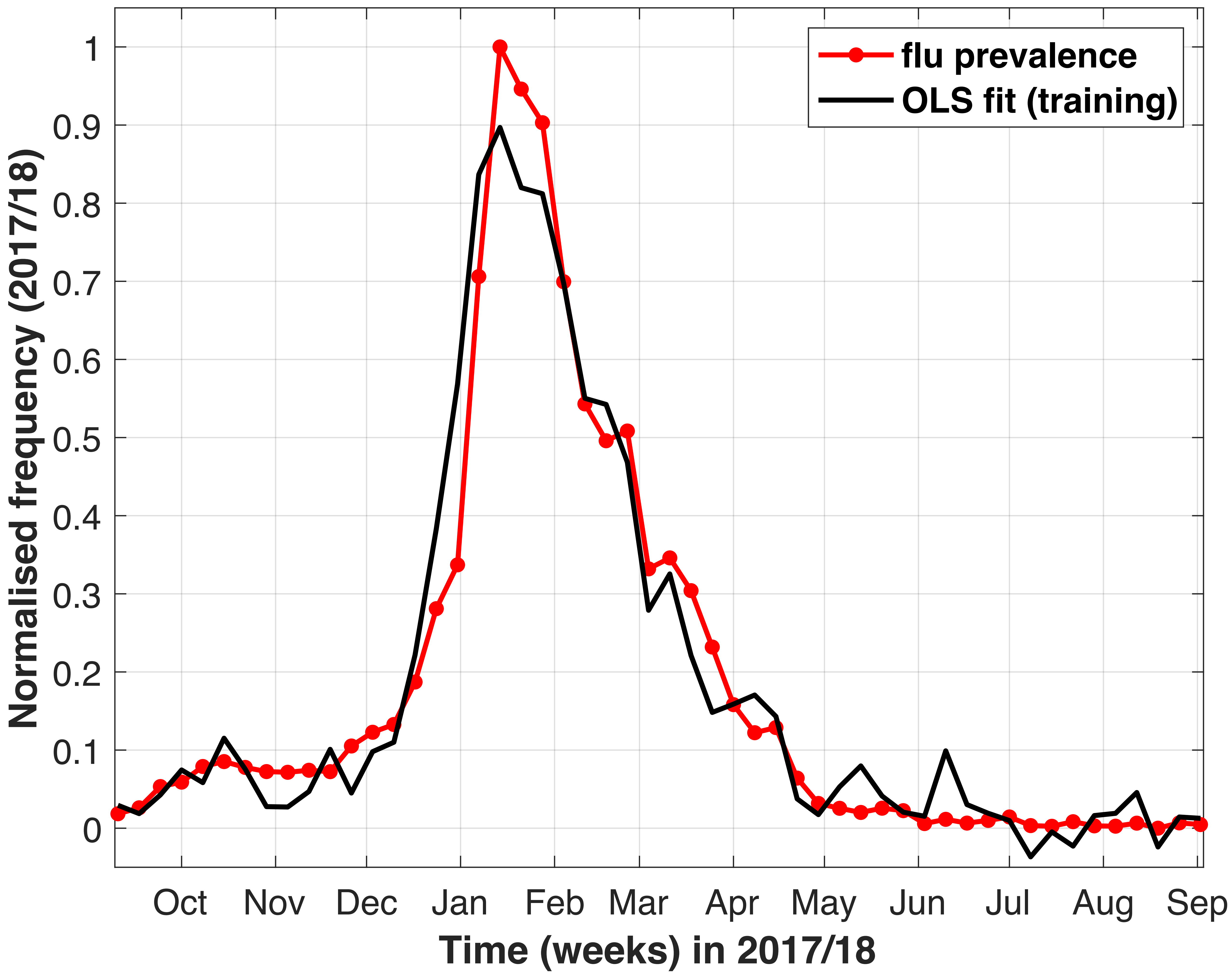
  Solve this using OLS regression.



# Supervised learning — OLS regression, *alternative point of view*

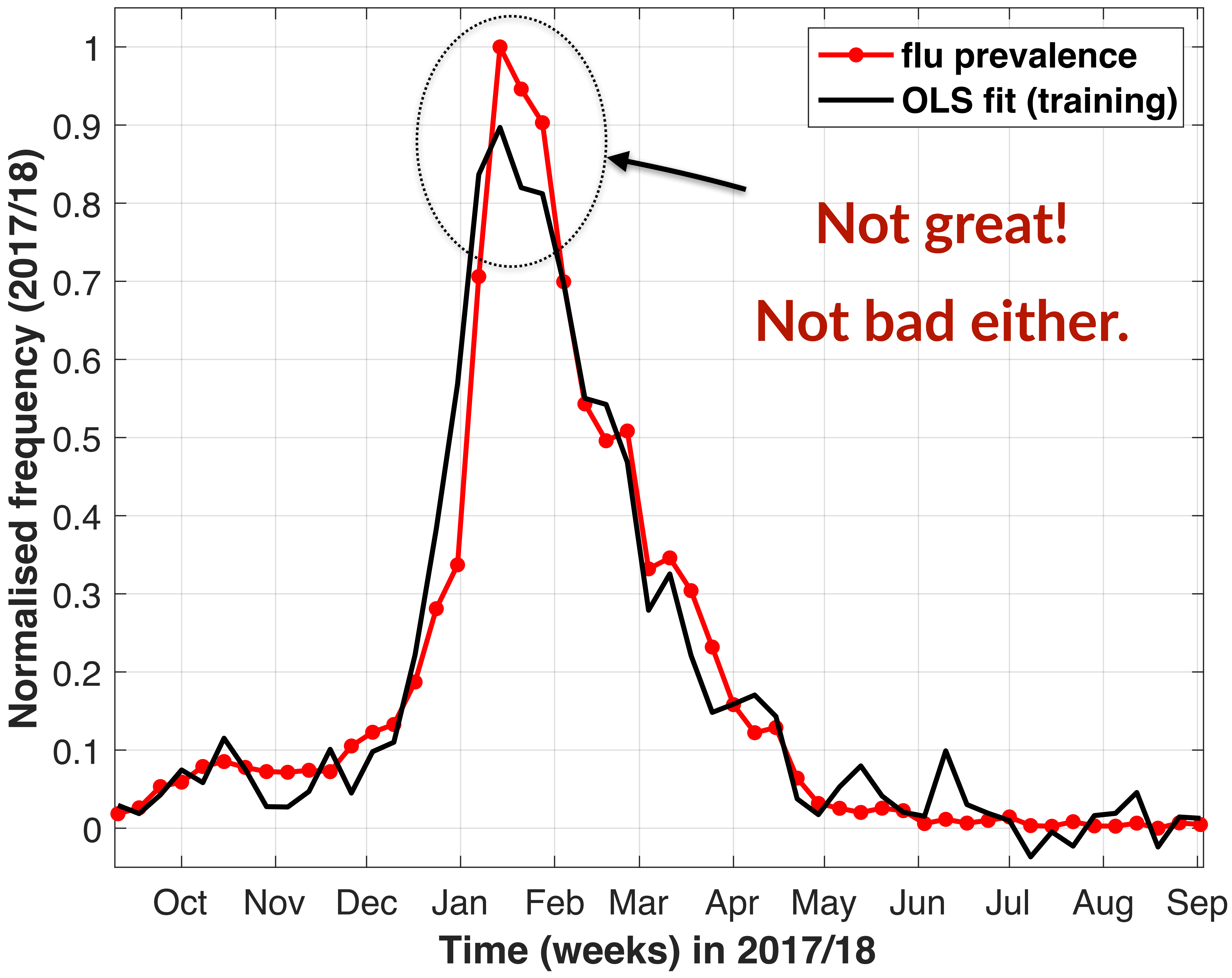
- $\mathbf{y}$ ~ weekly flu prevalence
- $\mathbf{x}$ ~ weekly search frequency of "*how long does flu last*"
- $f : \mathbf{x} \rightarrow \mathbf{y}$ such that
  $f(x_i) = \hat{y}_i = \alpha x_i + \beta \approx y_i$

  Solve this using OLS regression.
- The black line is the fit on the training data after applying OLS. It tells us how well can a linear function capture the training data.



- $\mathbf{y}$ ~ weekly flu prevalence
- $\mathbf{x}$ ~ weekly search frequency of "*how long does flu last*"
- $f : \mathbf{x} \rightarrow \mathbf{y}$ such that
  $f(x_i) = \hat{y}_i = \alpha x_i + \beta \approx y_i$

  Solve this using OLS regression.
- The black line is the fit on the training data after applying OLS. It tells us how well can a linear function capture the training data. **Not bad & not great fit!**



# Supervised learning — OLS regression calculus solution

# Supervised learning — OLS regression calculus solution

- The aim is to learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \mathbb{R}^{n}$

# Supervised learning — OLS regression calculus solution

- The aim is to learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \mathbb{R}^n$
- $f$ is a linear function, a set of weights and an intercept term; denoted by $\mathbf{w} \in \mathbb{R}^m$

# Supervised learning — OLS regression calculus solution

- The aim is to learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \mathbb{R}^{n}$
- $f$ is a linear function, a set of weights and an intercept term; denoted by $\mathbf{w} \in \mathbb{R}^{m}$
- In our regression task (*see previous slides*), there is 1 weight ($\alpha$) and the intercept ($\beta$)
  $\mathbf{X}$ has one column with the values of $\mathbf{x}$ and the other column is 1s

# Supervised learning — OLS regression calculus solution

- The aim is to learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \mathbb{R}^n$
- $f$ is a linear function, a set of weights and an intercept term; denoted by $\mathbf{w} \in \mathbb{R}^m$
- In our regression task (*see previous slides*), there is 1 weight ($\alpha$) and the intercept ($\beta$)
  $\mathbf{X}$ has one column with the values of $\mathbf{x}$ and the other column is 1s
- Minimise a loss function known as residual sum or squares (*equivalent to mean squared error that we will see next*): $\mathscr{L}(\mathbf{w}) = \|\mathbf{Xw} - \mathbf{y}\|_2^2 = \left(\mathbf{Xw} - \mathbf{y}\right)^\top \left(\mathbf{Xw} - \mathbf{y}\right)$

# Supervised learning — OLS regression calculus solution

- The aim is to learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \mathbb{R}^{n}$
- $f$ is a linear function, a set of weights and an intercept term; denoted by $\mathbf{w} \in \mathbb{R}^{m}$
- In our regression task (*see previous slides*), there is 1 weight ($\alpha$) and the intercept ($\beta$)
  $\mathbf{X}$ has one column with the values of $\mathbf{x}$ and the other column is 1s
- Minimise a loss function known as residual sum or squares (*equivalent to mean squared error that we will see next*): $\mathscr{L}(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 = \left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)^\top \left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)$
- This can also be written as: $\mathscr{L}(\mathbf{w}) = \mathscr{L}\left(\alpha, \beta\right) = \sum_{i=1}^{n} \left(\alpha x_i + \beta - y_i\right)^2$

# Supervised learning — OLS regression calculus solution

# Supervised learning — OLS regression calculus solution

- The aim is to learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \mathbb{R}^n$
- $f$ is a linear function, a set of weights and an intercept term; denoted by $\mathbf{w} \in \mathbb{R}^m$
- In our regression task (*see previous slides*), there is 1 weight ($\alpha$) and the intercept ($\beta$)
  $\mathbf{X}$ has one column with the values of $\mathbf{x}$ and the other column is 1s
- Minimise a loss function known as residual sum or squares (*equivalent to mean squared error that we will see next*): $\mathscr{L}(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 = \left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)^\top \left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)$

# Supervised learning — OLS regression calculus solution

- The aim is to learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \mathbb{R}^{n}$
- $f$ is a linear function, a set of weights and an intercept term; denoted by $\mathbf{w} \in \mathbb{R}^{m}$
- In our regression task (*see previous slides*), there is 1 weight ($\alpha$) and the intercept ($\beta$)
  $\mathbf{X}$ has one column with the values of $\mathbf{x}$ and the other column is 1s
- Minimise a loss function known as residual sum or squares (*equivalent to mean squared error that we will see next*): $\mathscr{L}(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 = \left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)^{\top} \left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)$
- Derivative of the loss with respect to $\mathbf{w}$: $\dfrac{\partial \mathscr{L}}{\partial \mathbf{w}} = -\,2\mathbf{X}^{\top}\mathbf{y} + 2\mathbf{X}^{\top}\mathbf{X}\mathbf{w}$

# Supervised learning — OLS regression calculus solution

- The aim is to learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \mathbb{R}^{n}$
- $f$ is a linear function, a set of weights and an intercept term; denoted by $\mathbf{w} \in \mathbb{R}^{m}$
- In our regression task (*see previous slides*), there is 1 weight ($\alpha$) and the intercept ($\beta$)
  $\mathbf{X}$ has one column with the values of $\mathbf{x}$ and the other column is 1s
- Minimise a loss function known as residual sum or squares (*equivalent to mean squared error that we will see next*): $\mathscr{L}(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 = \left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)^{\top}\left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)$
- Derivative of the loss with respect to $\mathbf{w}$: $\dfrac{\partial \mathscr{L}}{\partial \mathbf{w}} = -2\mathbf{X}^{\top}\mathbf{y} + 2\mathbf{X}^{\top}\mathbf{X}\mathbf{w}$
- Set this to $0$ and hence $\mathbf{w} = \left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{X}^{\top}\mathbf{y}$ as long as $\mathbf{X}^{\top}\mathbf{X}$ is full rank which means that the observations (rows) in $\mathbf{X}$ are more than the features ($n > m$) and that the features have no linear dependence

# Supervised learning — OLS model training & testing

- Going back to our flu rate modelling example, $\mathbf{w} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$ would give $\mathbf{w} = \left[0.93351 \; -0.036631\right]$, i.e. $\alpha = 0.93351$ and $\beta = -0.036631$

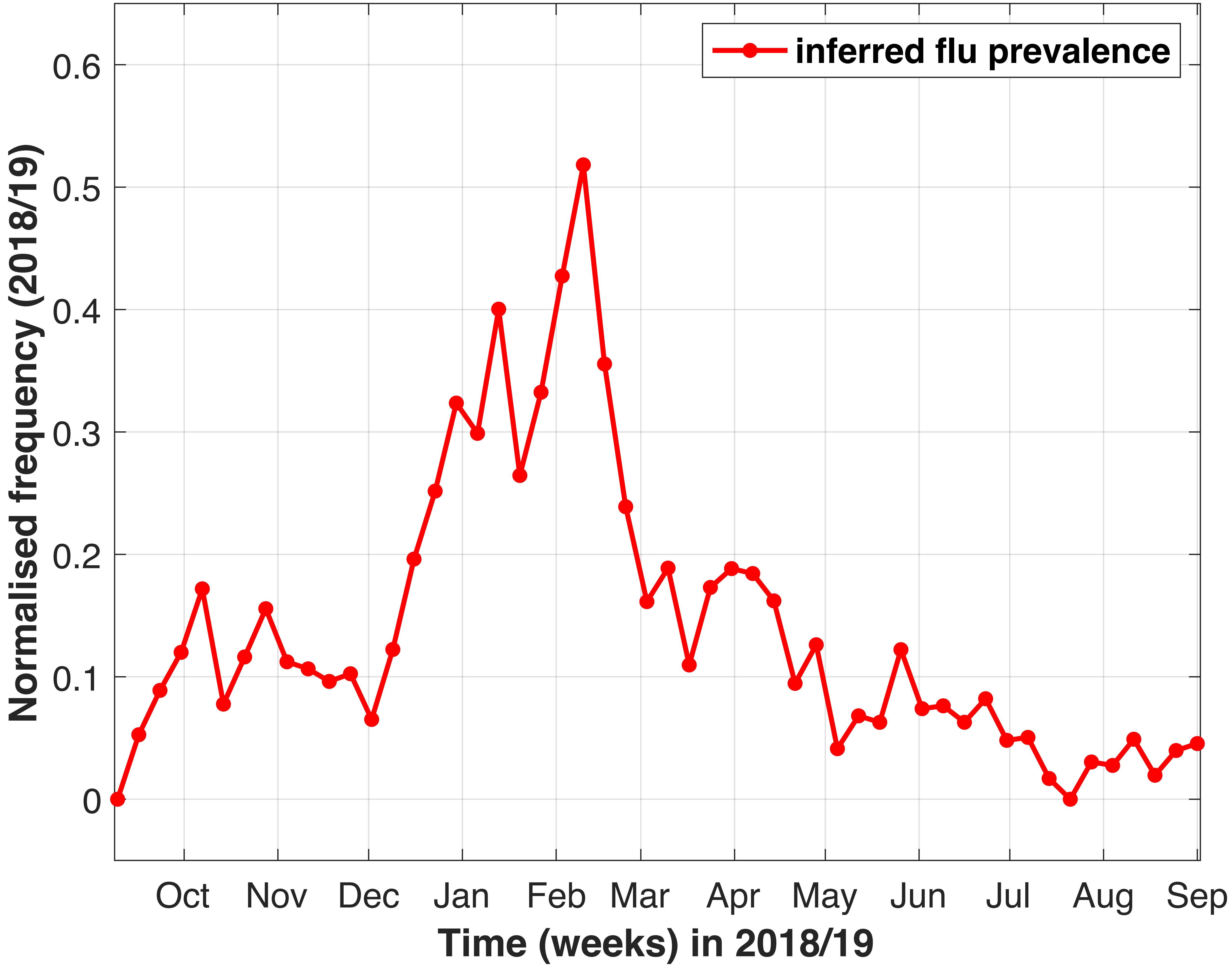
- The question now becomes, **how well will this model do in the next flu season?** i.e. how well would the model perform on unseen data / data that it has not been trained on?

- Let's use the above values of $\alpha$ and $\beta$ to estimate weekly flu prevalence in England for the season 2018/19 based on the corresponding frequency of the search query "*how long does flu last*"

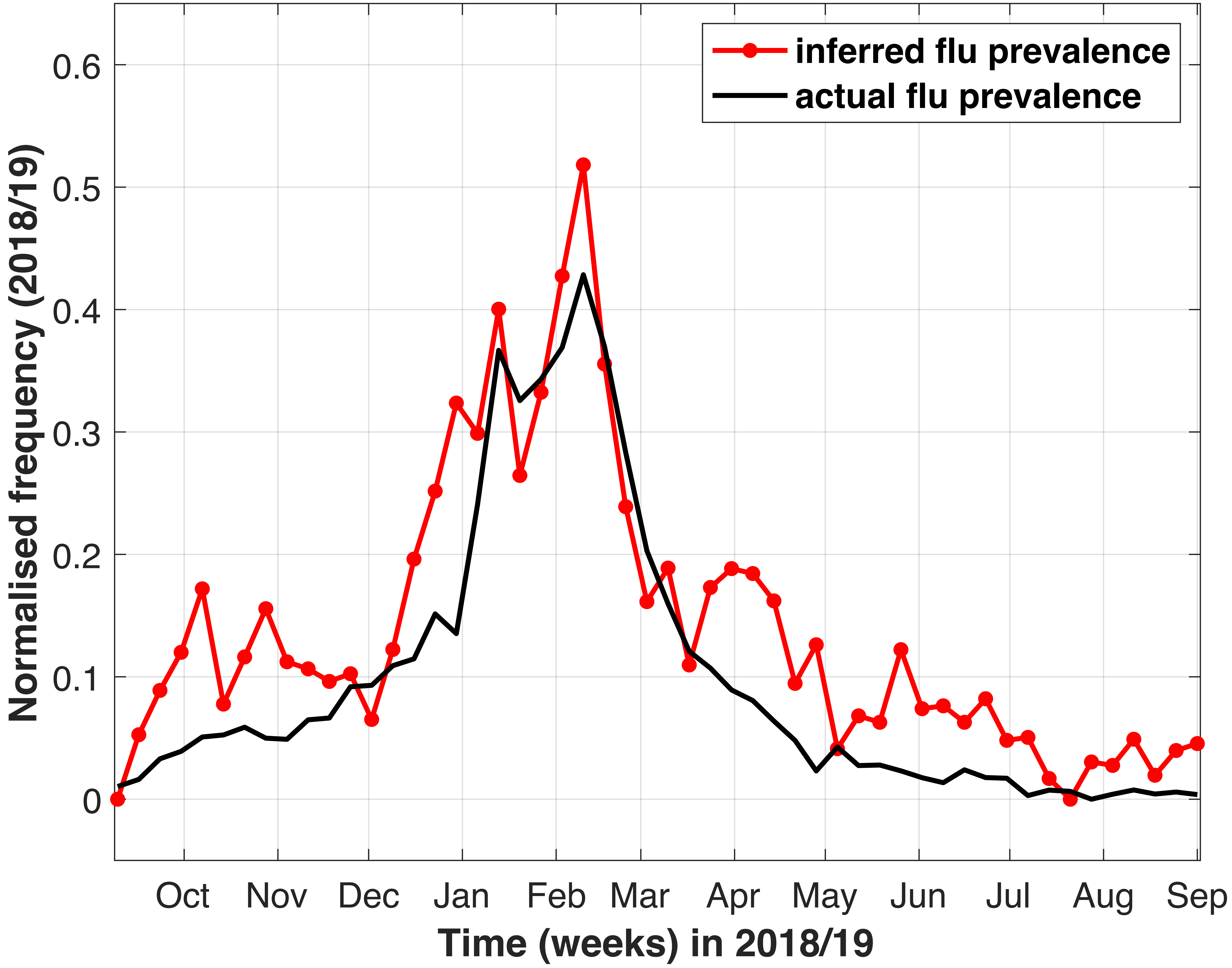
- And then compare it with the actual flu prevalence in England for 2018/19

# Supervised learning — OLS model training & testing

- These (*red line, dot • marker*) are the estimated (*inferred*) flu rates in 2018/19 (*to be exact from September 2018 to August 2019*) based on the OLS model and the frequency of the search query "*how long does flu last*"
- Recall, we trained our model using non-overlapping data from 2017/18 (*September 2017 to August 2018*)

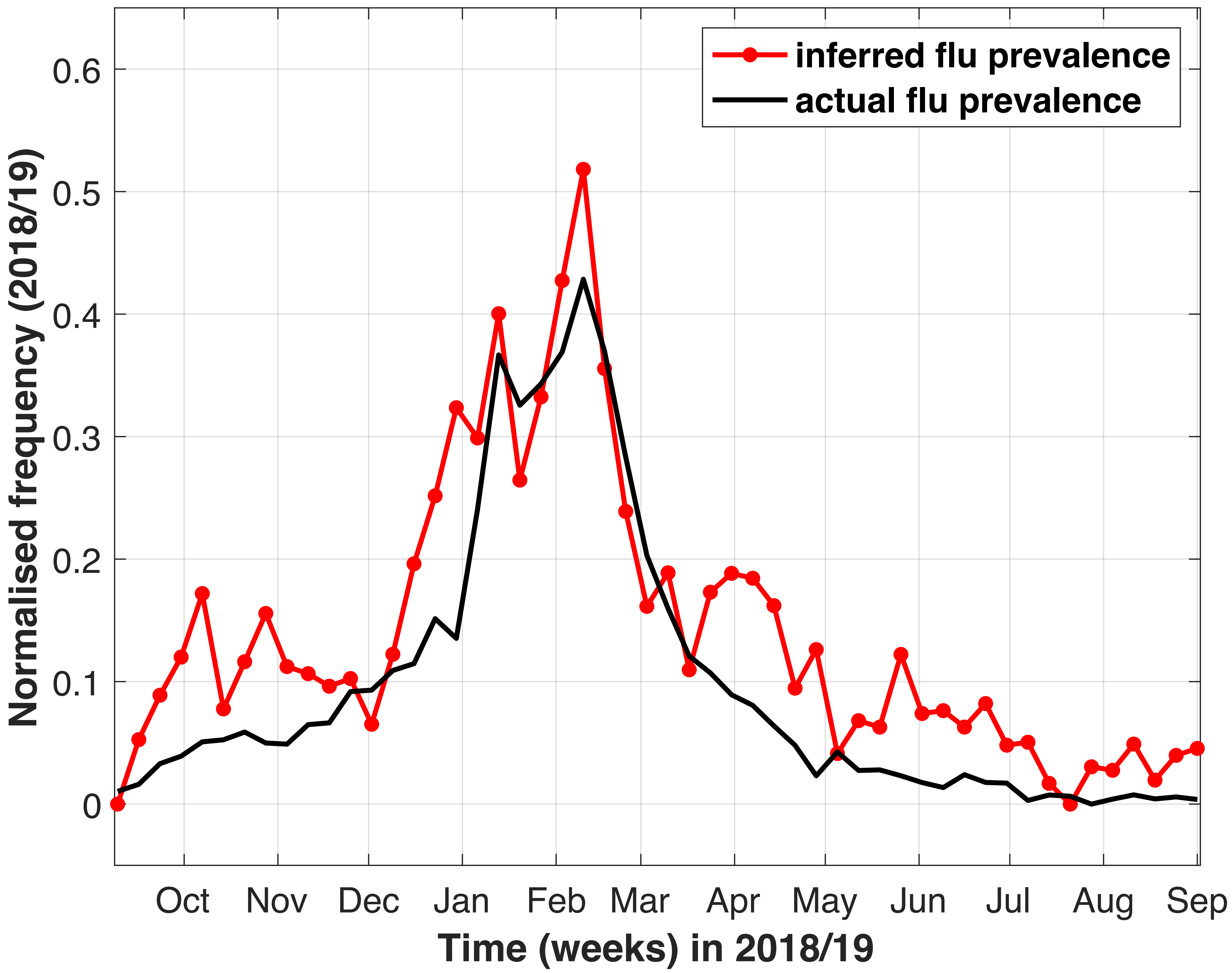


# Supervised learning — OLS model training & testing



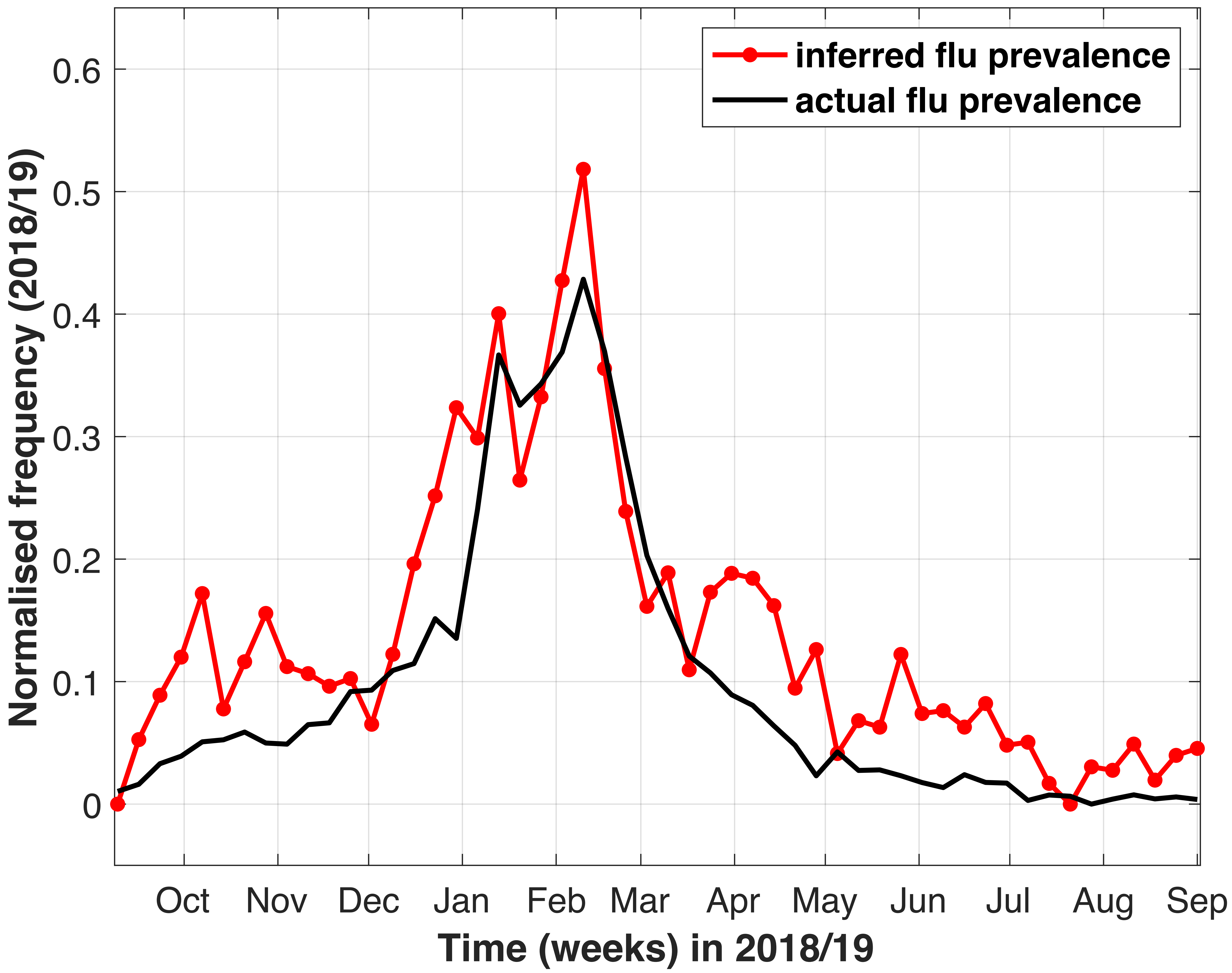
# Supervised learning — OLS model training & testing

- The **black solid** line represents the corresponding flu rates as reported by a health agency in the UK
- *Do you think this simple OLS model based on a single web search query did well?*
- $r$ = 0.919 (*bivariate correlation*)
  RMSE = 0.0632 (*root mean squared error*)
  MAE = 0.0519 (*mean absolute error*)



# Supervised learning — OLS model training & testing

- The **black solid** line represents the corresponding flu rates as reported by a health agency in the UK
- *Do you think this simple OLS model based on a single web search query did well?*
- $r$ = 0.919 (*bivariate correlation*)
  RMSE = 0.0632 (*root mean squared error*)
  MAE = 0.0519 (*mean absolute error*)
- considering the simplicity of the model, *its accuracy is quite surprising*



# Supervised learning — Gradient descent

# Supervised learning — Gradient descent

- **Gradient descent**: optimisation algorithm that minimises a loss function $\mathscr{J}$ with respect to a set of hyperparameters

- **Gradient descent**: optimisation algorithm that minimises a loss function $\mathcal{J}$ with respect to a set of hyperparameters
- Loss function for ordinary least squares (OLS) regression? If $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$ denotes our estimates for $\mathbf{y}$, then the loss function for OLS is their mean squared difference (error):

$$\mathcal{J}(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} \left(\hat{y}_i - y_i\right)^2 \text{, where } \hat{y}_i \in \hat{\mathbf{y}}, y_i \in \mathbf{y}$$

# Supervised learning — Gradient descent

- **Gradient descent**: optimisation algorithm that minimises a loss function $\mathcal{J}$ with respect to a set of hyperparameters
- Loss function for ordinary least squares (OLS) regression? If $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$ denotes our estimates for $\mathbf{y}$, then the loss function for OLS is their mean squared difference (error):

$$\mathcal{J}(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} \left(\hat{y}_i - y_i\right)^2 \text{ , where } \hat{y}_i \in \hat{\mathbf{y}}, y_i \in \mathbf{y}$$

- Basic **steps** of gradient descent

— define a loss function, $\mathcal{J}$

— compute the partial derivatives of $\mathcal{J}$ w.r.t. each hyperparameter

— update hyperparameters using their partial derivatives and learning rate $\ell$ often $\in (0,1)$

— repeat until convergence

# Supervised learning — Gradient descent

- **Learning rate:** how far away are we going to go in the opposite direction of the partial derivative / how much change are we going to impose?
  *we are going to see an example of this*

- **Why does gradient descent work?** We are taking steps in the opposite direction of the partial gradient of each hyperparameter to identify a local minimum of the loss.

- **When does it not work?** Not directly applicable to non-differentiable loss functions (but there exist workarounds)

# Supervised learning — OLS with gradient descent

In our example, we are modelling a flu rate $y_i$ using the frequency of a search query $x_i$

- **Hypothesis:** $\hat{y}_i = \alpha x_i + \beta$
  — *a flu estimate is a linear function of the frequency of the search query*
- **Hyperparameters:** $\{\alpha, \beta\}$
  *these are unknown and should be estimated using gradient descent*
- **Loss function:** $\mathcal{J}\left(\alpha, \beta\right) = \dfrac{1}{2n} \sum_{i=1}^{n} \left(\hat{y}_i - y_i\right)^2$
- **Goal:** $\underset{\alpha,\beta}{\arg\min}\, \mathcal{J}\left(\alpha, \beta\right)$

# Supervised learning — OLS with gradient descent

# Supervised learning — OLS with gradient descent

In our example, we are modelling a flu rate $y_i$ using the frequency of a search query $x_i$

- Start with some initial values for $\alpha$ and $\beta$ denoted by $\alpha_0$ and $\beta_0$, respectively

# Supervised learning — OLS with gradient descent

In our example, we are modelling a flu rate $y_i$ using the frequency of a search query $x_i$

- Start with some initial values for $\alpha$ and $\beta$ denoted by $\alpha_0$ and $\beta_0$, respectively
- In iteration $t+1$ of the gradient descent algorithm, update $\alpha$ and $\beta$ with:

$$\alpha_{t+1} = \alpha_t - \ell \frac{\partial \mathcal{J}\left(\alpha, \beta\right)_t}{\partial \alpha} \text{ and } \beta_{t+1} = \beta_t - \ell \frac{\partial \mathcal{J}\left(\alpha, \beta\right)_t}{\partial \beta}$$

  where $\ell$ often $\in (0,1)$ denotes the learning rate we want to impose

# Supervised learning — OLS with gradient descent

In our example, we are modelling a flu rate $y_i$ using the frequency of a search query $x_i$

- Start with some initial values for $\alpha$ and $\beta$ denoted by $\alpha_0$ and $\beta_0$, respectively
- In iteration $t+1$ of the gradient descent algorithm, update $\alpha$ and $\beta$ with:

$$\alpha_{t+1} = \alpha_t - \ell \frac{\partial \mathcal{J}\left(\alpha, \beta\right)_t}{\partial \alpha} \text{ and } \beta_{t+1} = \beta_t - \ell \frac{\partial \mathcal{J}\left(\alpha, \beta\right)_t}{\partial \beta}$$

  where $\ell$ often $\in (0,1)$ denotes the learning rate we want to impose
- NB: both derivatives update in iteration $t+1$ based on values from iteration $t$

# Supervised learning — OLS with gradient descent

In our example, we are modelling a flu rate $y_i$ using the frequency of a search query $x_i$

- Start with some initial values for $\alpha$ and $\beta$ denoted by $\alpha_0$ and $\beta_0$, respectively
- In iteration $t+1$ of the gradient descent algorithm, update $\alpha$ and $\beta$ with:

$$\alpha_{t+1} = \alpha_t - \ell \frac{\partial \mathcal{J}\left(\alpha, \beta\right)_t}{\partial \alpha} \quad \text{and} \quad \beta_{t+1} = \beta_t - \ell \frac{\partial \mathcal{J}\left(\alpha, \beta\right)_t}{\partial \beta}$$

  where $\ell$ often $\in (0,1)$ denotes the learning rate we want to impose

- NB: both derivatives update in iteration $t+1$ based on values from iteration $t$
- Repeat until convergence

# Supervised learning — OLS with gradient descent, the derivatives

Loss function: $$\mathcal{J}\left(\alpha,\beta\right) = \frac{1}{2n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2$$

$n$ samples, $2n$ is a convention, $\mathcal{J} = \text{MSE}/2$

$$= \frac{1}{2n}\sum_{i=1}^{n}\left(\alpha x_i + \beta - y_i\right)^2$$

$$\frac{\partial \mathcal{J}\left(\alpha,\beta\right)}{\partial \alpha} = \frac{1}{2n}\sum_{i=1}^{n}\left(2\left(\alpha x_i + \beta - y_i\right)x_i\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\left(\alpha x_i + \beta - y_i\right)x_i\right)$$

# Supervised learning — OLS with gradient descent, the derivatives

Loss function: $\mathcal{J}\left(\alpha,\beta\right)=\frac{1}{2n}\sum_{i=1}^{n}\left(\hat{y}_i-y_i\right)^2$ $n$ samples, $2n$ is a convention, $\mathcal{J}=\text{MSE}/2$

$$=\frac{1}{2n}\sum_{i=1}^{n}\left(\alpha x_i+\beta-y_i\right)^2$$

$$\frac{\partial\mathcal{J}\left(\alpha,\beta\right)}{\partial\alpha}=\frac{1}{2n}\sum_{i=1}^{n}\left(2\left(\alpha x_i+\beta-y_i\right)x_i\right)=\frac{1}{n}\sum_{i=1}^{n}\left(\left(\alpha x_i+\beta-y_i\right)x_i\right)$$

$$\frac{\partial\mathcal{J}\left(\alpha,\beta\right)}{\partial\beta}=\frac{1}{n}\sum_{i=1}^{n}\left(\alpha x_i+\beta-y_i\right)$$

## Supervised learning — OLS with gradient descent, the derivatives

What if we had $m$ predictors?

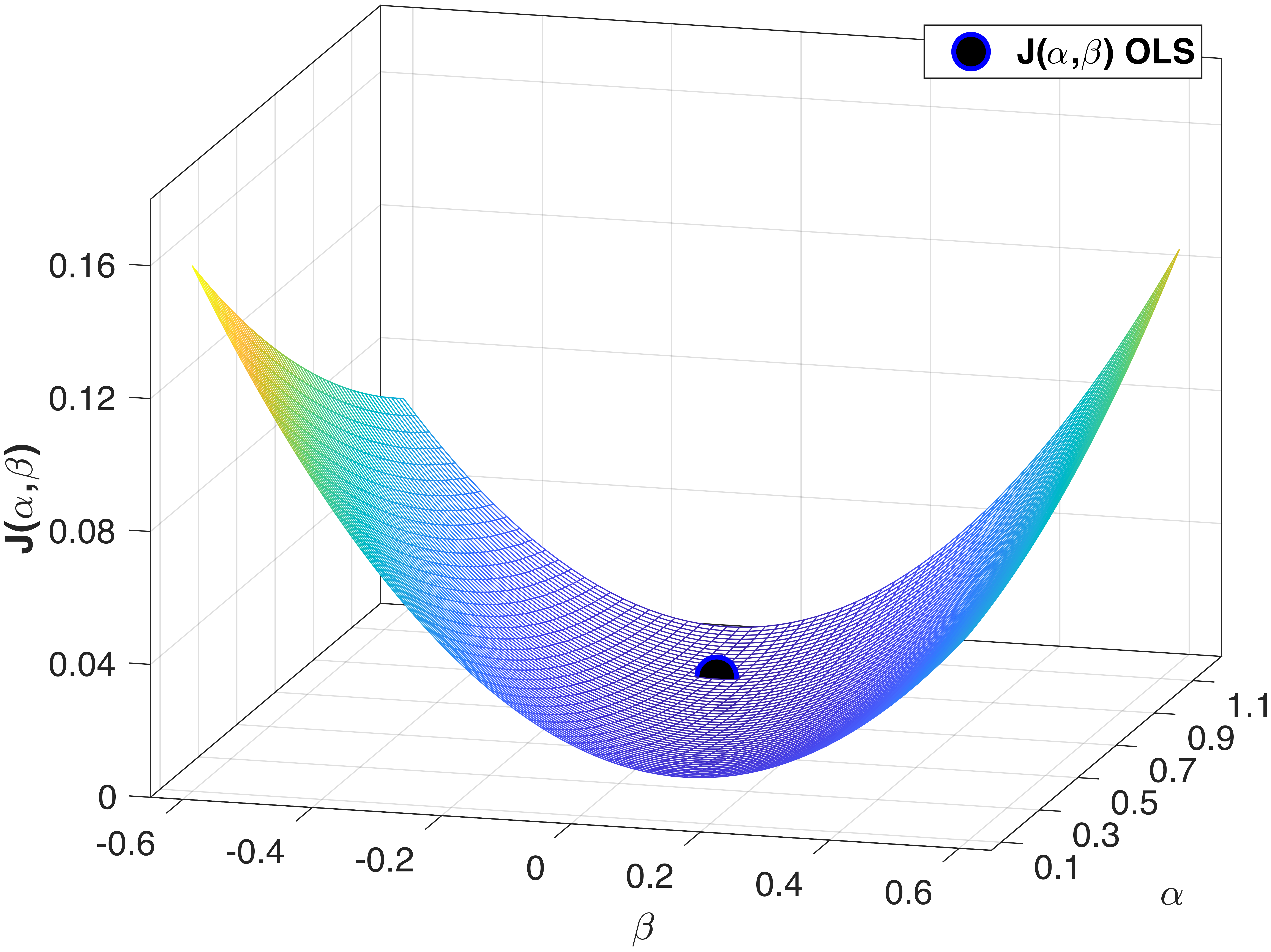
$$\mathcal{J}\left(\mathbf{w},\beta\right) = \frac{1}{2n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2$$

$$= \frac{1}{2n}\sum_{i=1}^{n}\left(w_1 x_{i,1} + \ldots + w_m x_{i,m} + \beta - y_i\right)^2$$

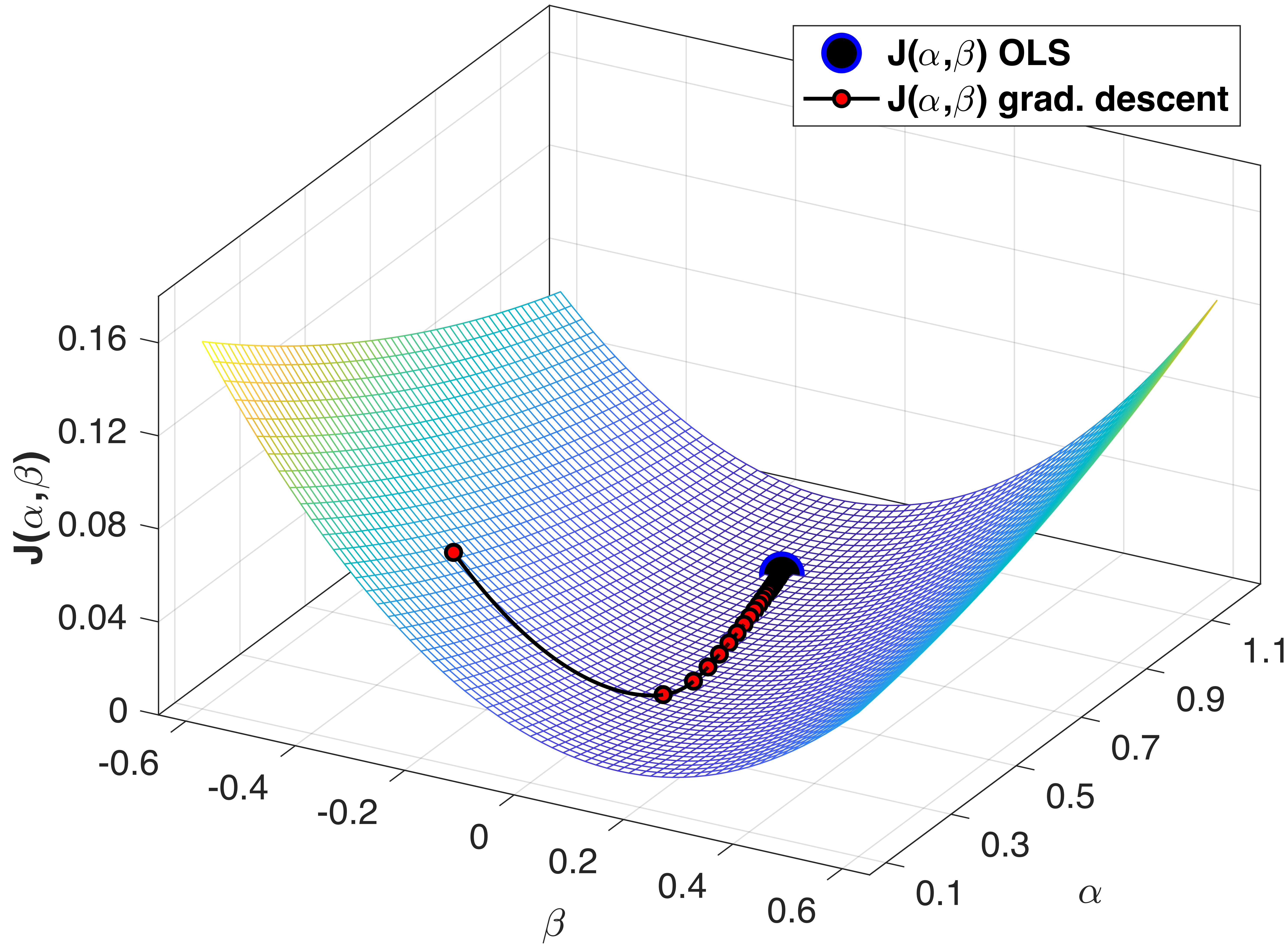
$$\frac{\partial \mathcal{J}\left(\mathbf{w},\beta\right)}{\partial w_j} = \frac{1}{n}\sum_{i=1}^{n}\left(\left(w_1 x_{i,1} + \ldots + w_m x_{i,m} + \beta - y_i\right) x_{i,j}\right)$$

$$\frac{\partial \mathcal{J}\left(\mathbf{w},\beta\right)}{\partial \beta} = ?$$

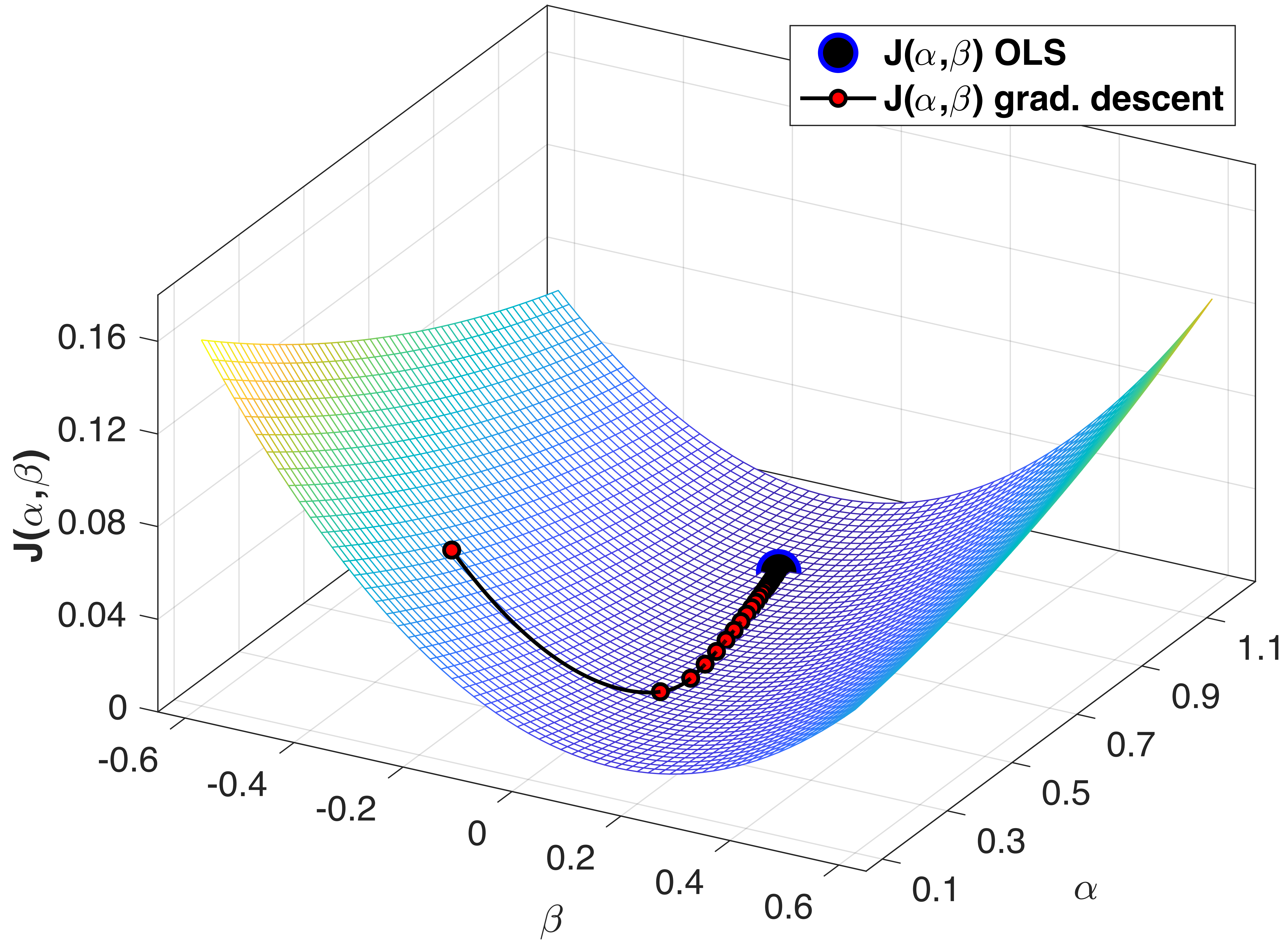
- OLS example: inferring flu prevalence based on the frequency of 1 search query
- Let's explore the space of hyperparameter values for OLS $\{\alpha, \beta\}$ and the corresponding values of the loss function $\mathscr{J}\left(\alpha, \beta\right)$ – 3-dimensional plot (surface or mesh plot)
- Convex loss (**easier task / global minimum**)
- Big (half) dot/ball denotes the exact OLS solution (*no gradient descent used*)



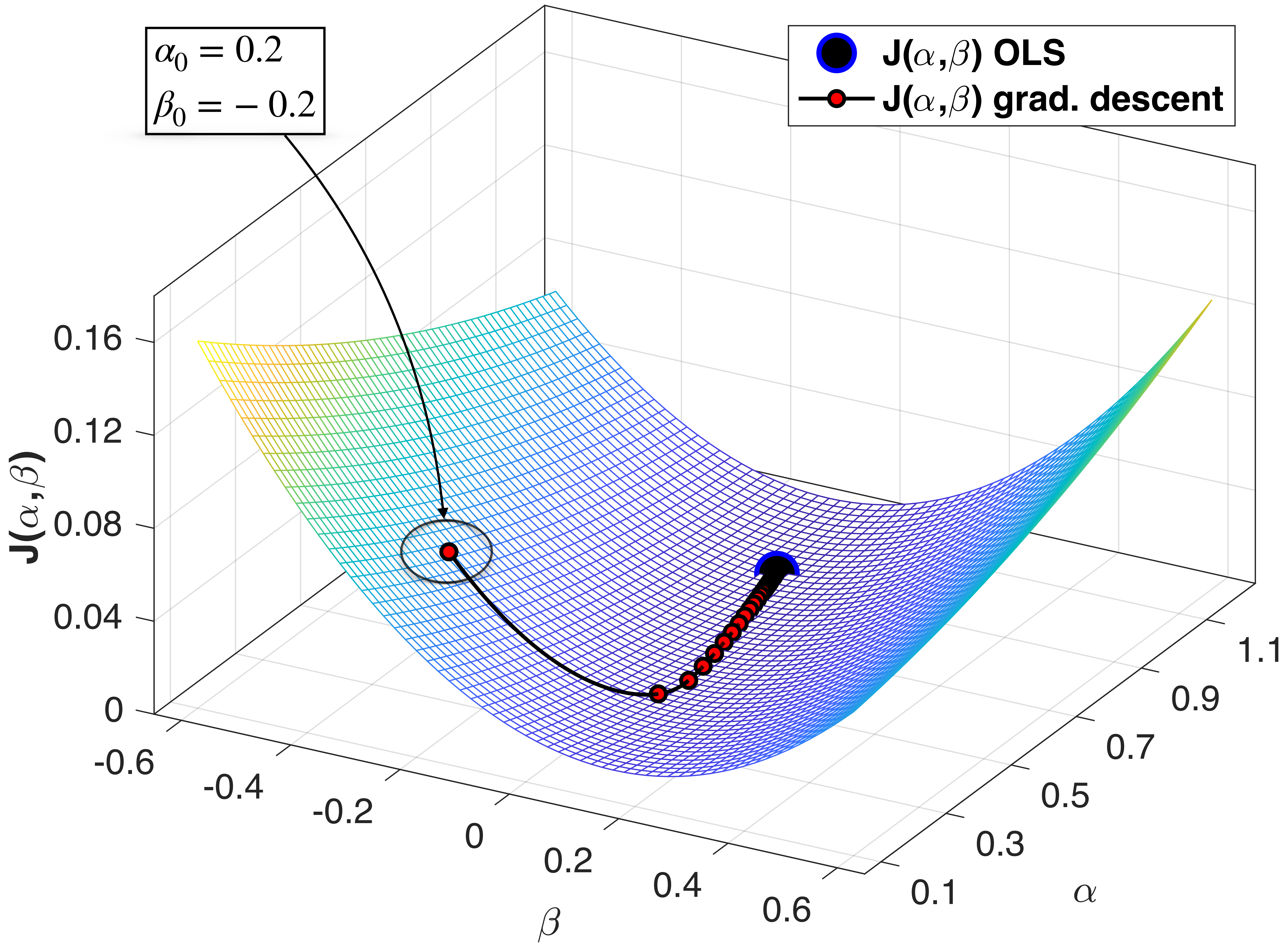
# Supervised learning — OLS with gradient descent



- Let's start from a point in the grid, set some initial values for the hyperparameters and attempt to solve this with coordinate descent

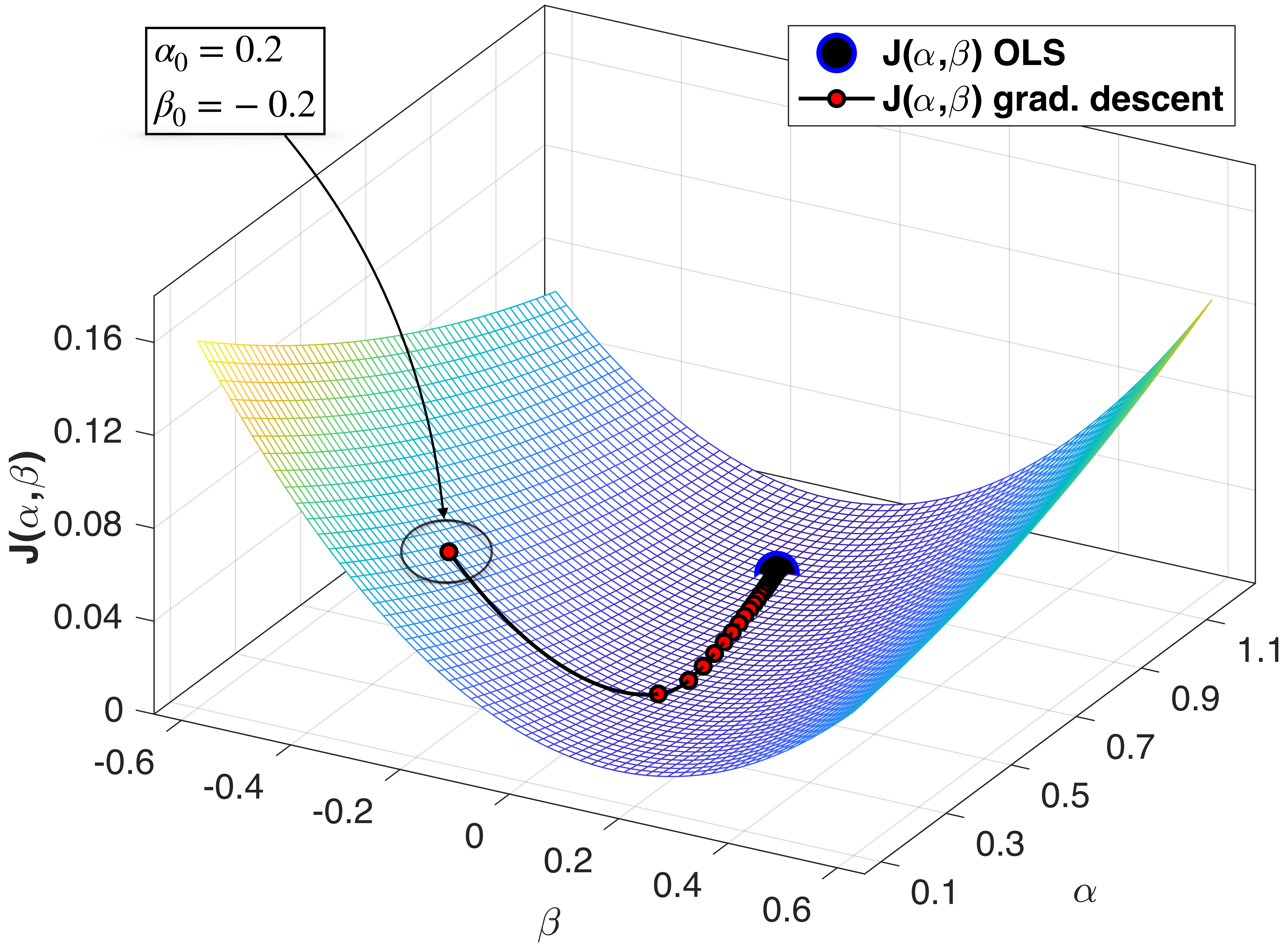


- Let’s start from a point in the grid, set some initial values for the hyperparameters and attempt to solve this with coordinate descent
- $\alpha_0 = 0.2, \beta_0 = -0.2$
- $\ell = 0.02$ (*learning rate*)



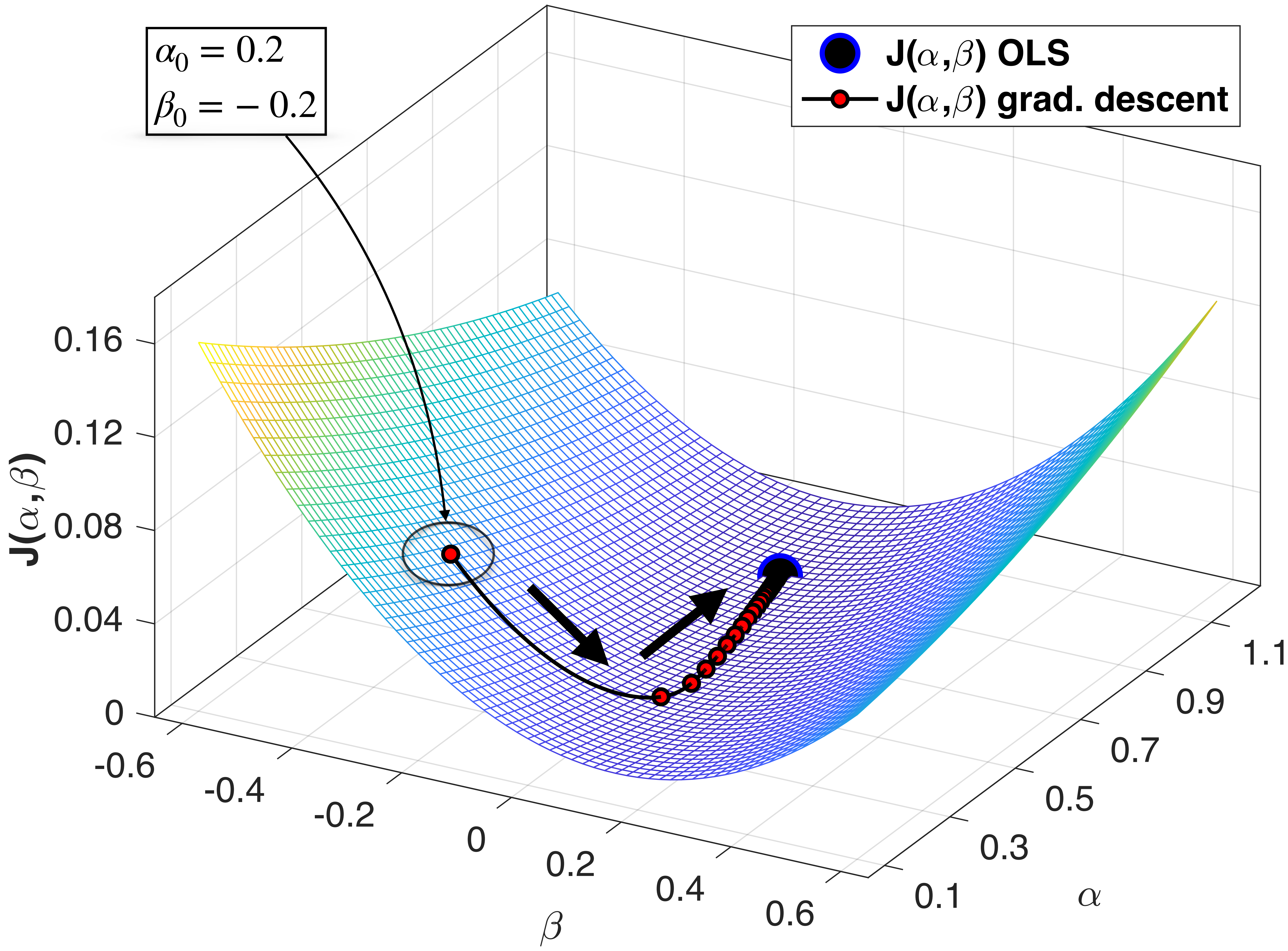
# Supervised learning — OLS with gradient descent

- Let's start from a point in the grid, set some initial values for the hyperparameters and attempt to solve this with coordinate descent
- $\alpha_0 = 0.2, \beta_0 = -0.2$
- $\ell = 0.02$ *(learning rate)*
- Convergence criterion: How much has $\mathcal{J}(\alpha, \beta)$ changed in the past $k$ iterations?

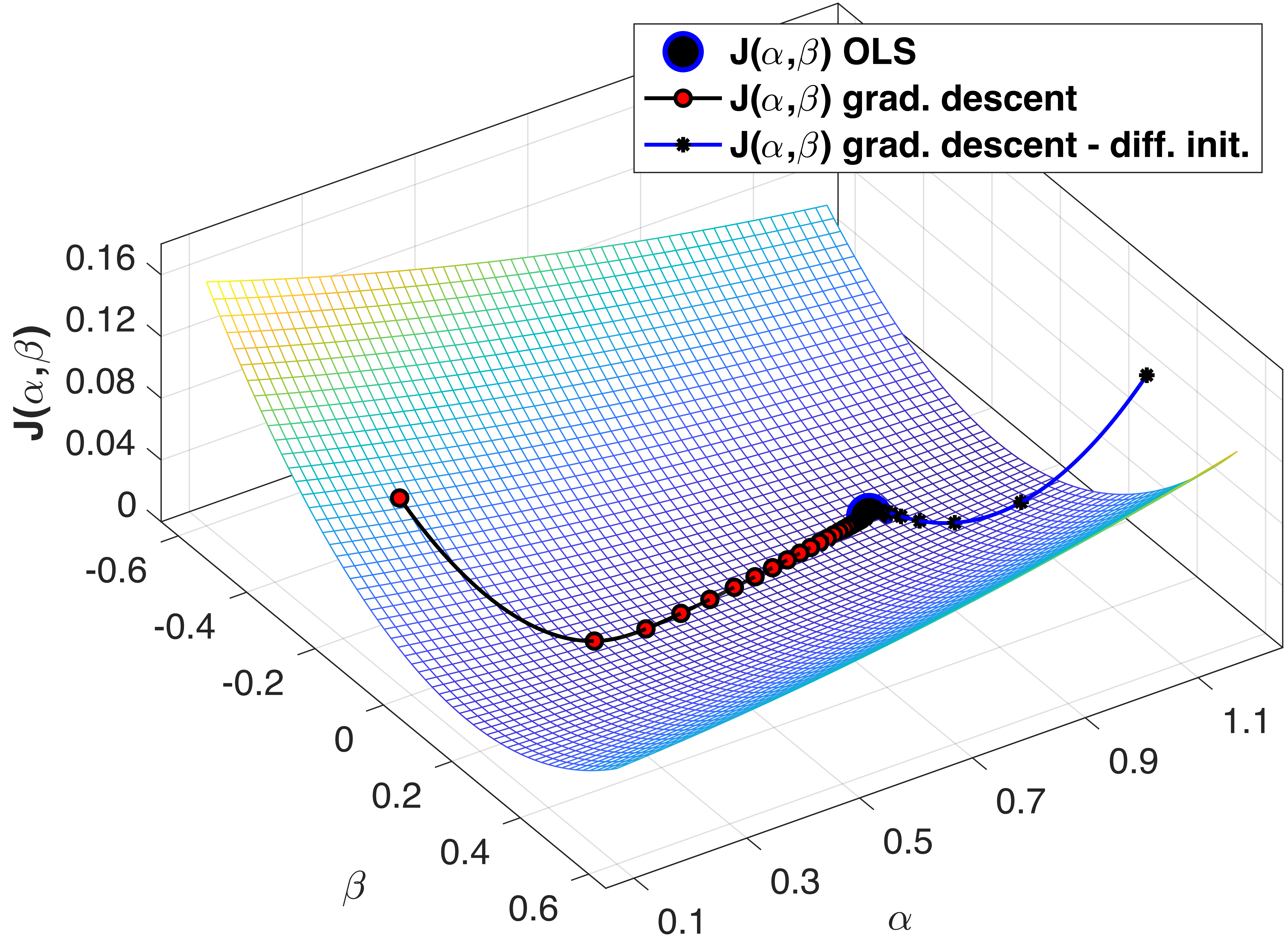


# Supervised learning — OLS with gradient descent

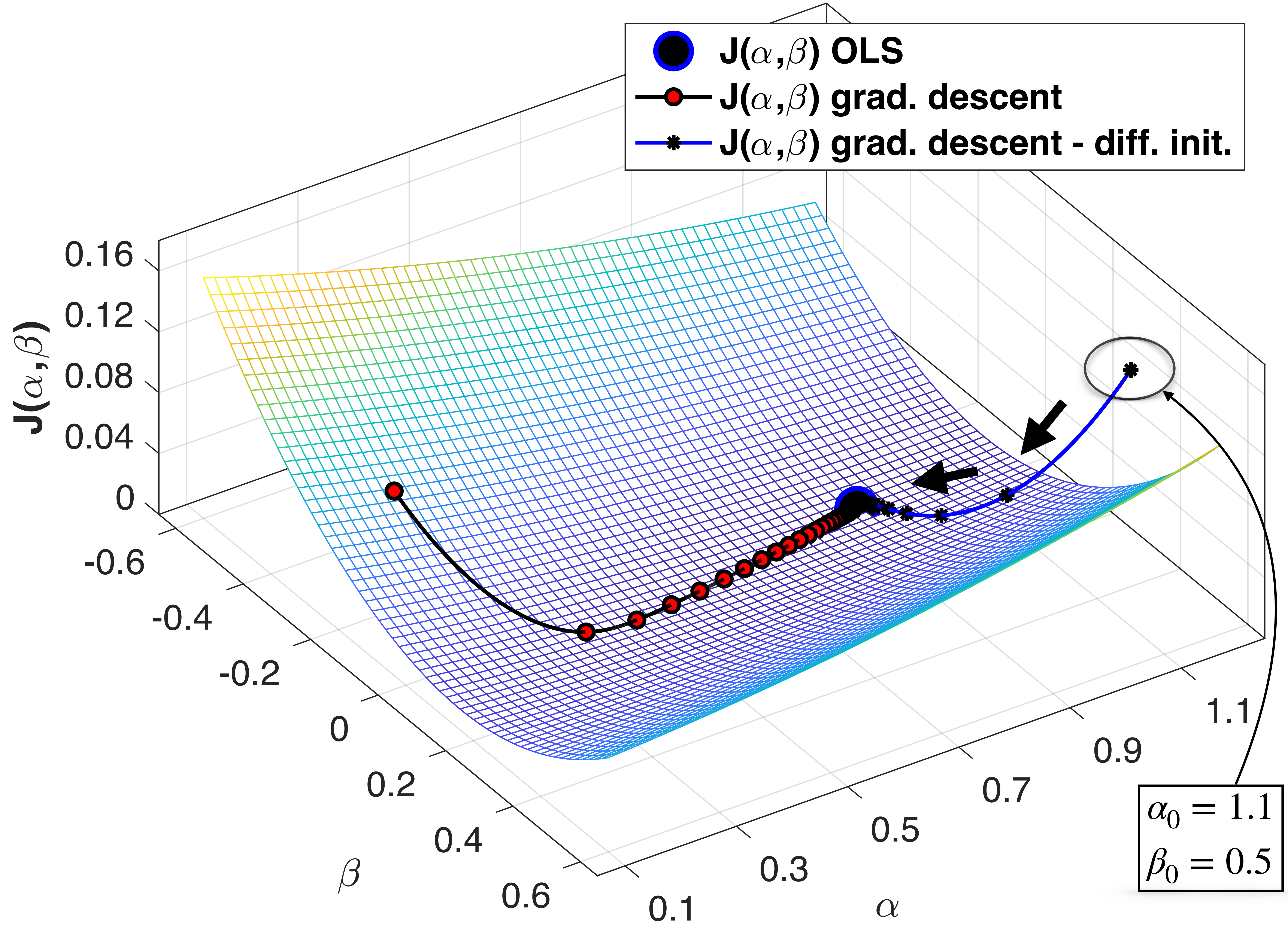
- Let's start from a point in the grid, set some initial values for the hyperparameters and attempt to solve this with coordinate descent
- $\alpha_0 = 0.2, \beta_0 = -0.2$
- $\ell = 0.02$ *(**learning rate**)*
- Convergence criterion: How much has $\mathcal{J}(\alpha, \beta)$ changed in the past $k$ iterations?
- Gradient descent's solution almost identical to exact OLS solution (**expected?**)



- Let's change the starting point
- $\alpha_0 = 1.1, \beta_0 = 0.5$
- $\ell = 0.02$ (*same learning rate*)
- In this case, it does not affect our solution (**why?**)



- Let's change the starting point
- $\alpha_0 = 1.1, \beta_0 = 0.5$
- $\ell = 0.02$ (*same learning rate*)
- In this case, it does not affect our solution (**why?**)



# Supervised learning — Gradient descent, general remarks

- Effect of **learning rate** $\ell$
  — if it is too small, gradient descent can be slow
  — if it is too large, gradient descent may fail to converge (overshoots the minimum)
  — adaptive learning rate (*by using line search*)
- Different initialisations might help get past local optima
- **Batch** gradient descent (*presented today*): use the entire training set for gradient updates
  — guaranteed convergence to a local minimum
  — slow on large problems (e.g. neural networks)
- **Stochastic** gradient descent: use one training sample for gradient updates
  — faster convergence on large redundant data sets
  — hard to reach high accuracy
- **Mini-batch** gradient descent: use a subset of the training set for gradient updates
  — very common in neural network training
  — better in avoiding local minima
  — what is the best mini-batch size (number of training samples to use)?

# Supervised learning

- **Regression**
  estimate / predict a continuous output / target variable

  i.e. learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \mathbb{R}^{n}$

  Examples: predict a time series trend (finance, climate, etc.), estimate the prevalence of an infectious disease disease in epidemiology

- **Classification**
  estimate a set of $C$ unordered (and mutually exclusive) labels / classes

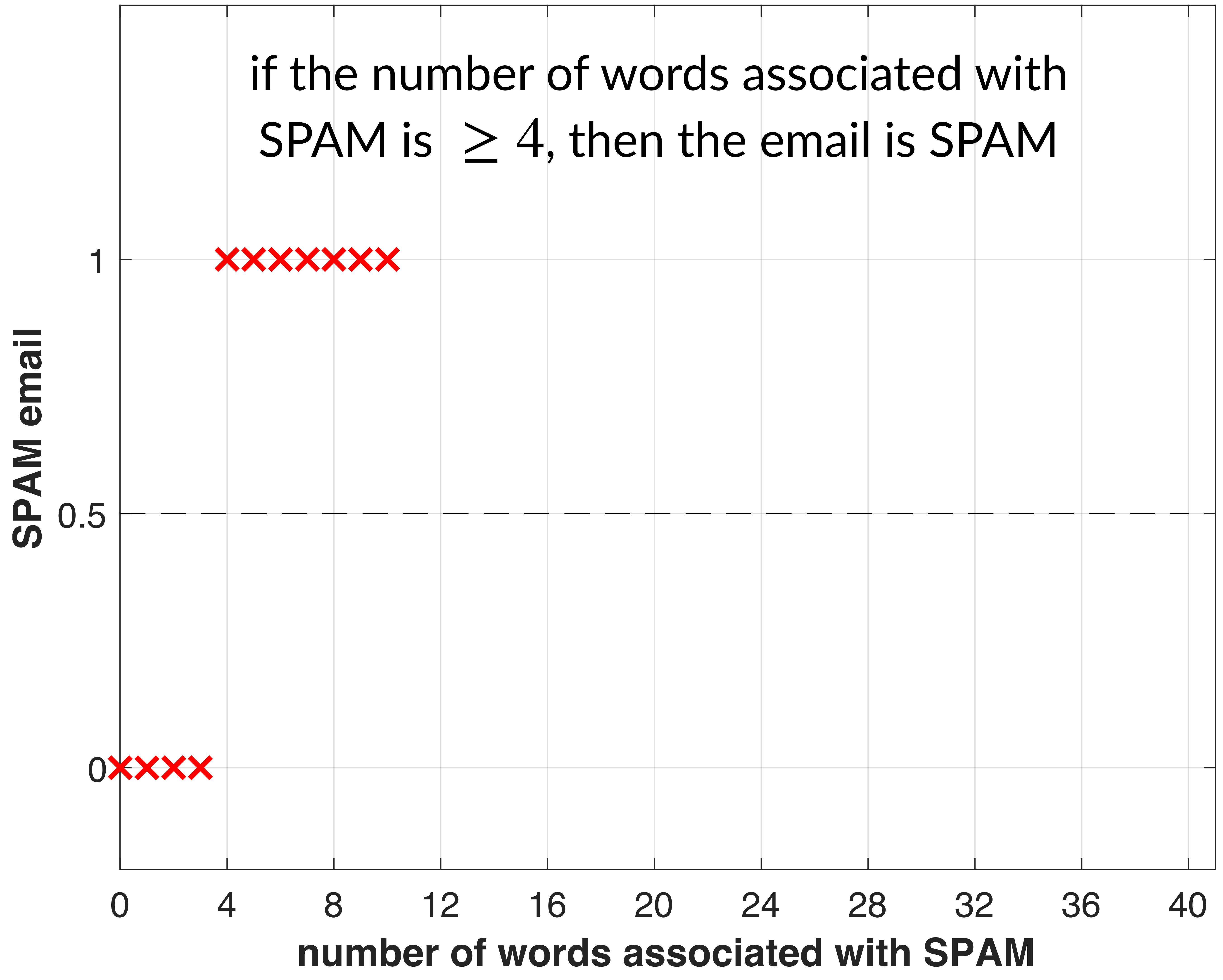
  i.e. learn $f : \mathbf{X} \in \mathbb{R}^{n \times m} \rightarrow \mathbf{y} \in \{1, 2, \ldots, C\}$

  Examples: detect spam email, medical imaging, text classification, language models

- **Binary classification** means that we only have two label categories, e.g.
  > *spam* vs. *not spam* email
  > *relevant* vs. *not relevant* document

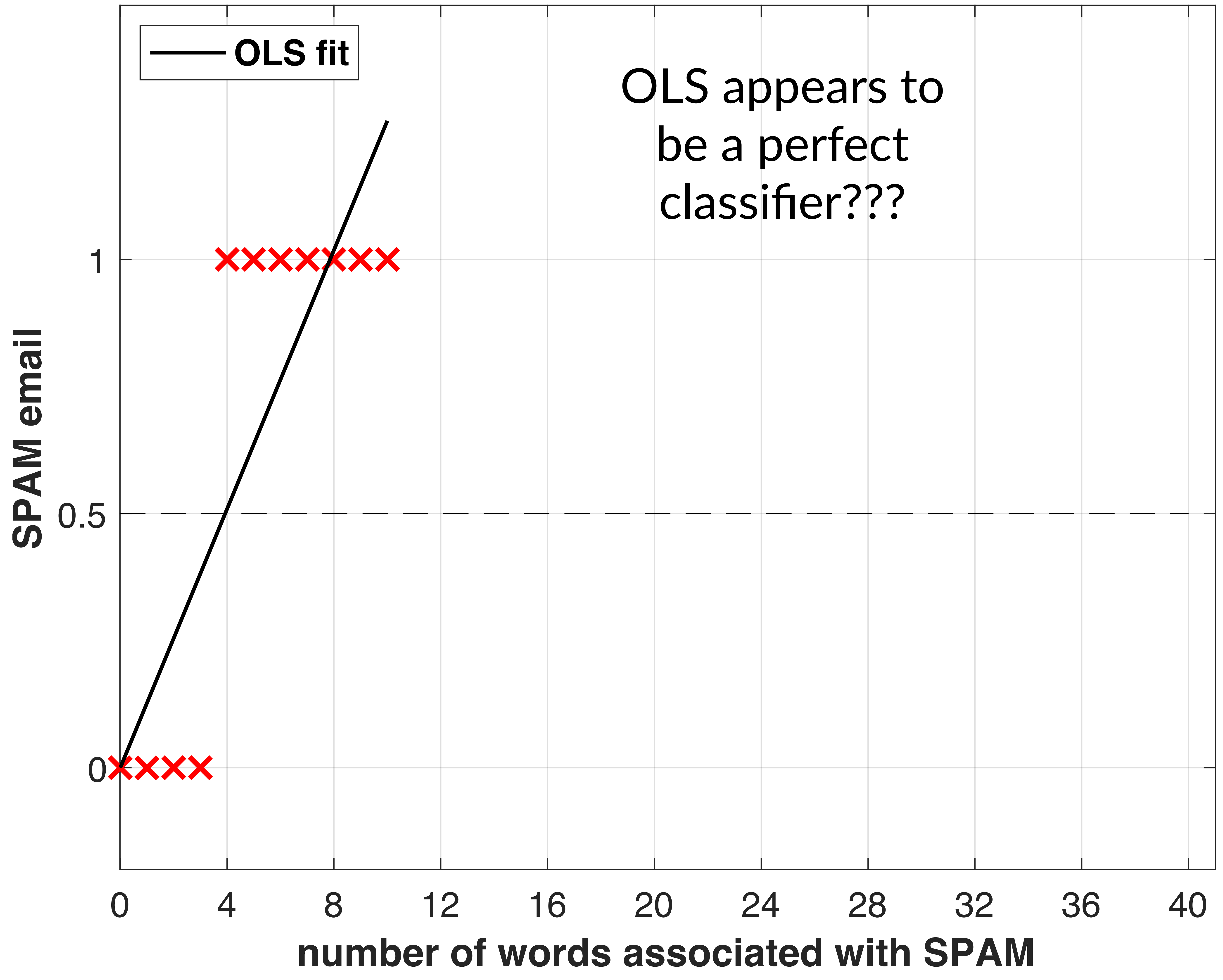
- if $f_{\mathbf{w}}\left(x_i\right) \geq 0.5,$ then SPAM
  if $f_{\mathbf{w}}\left(x_i\right) < 0.5,$ then not SPAM

- What if we used OLS to learn $f_{\mathbf{w}}$?



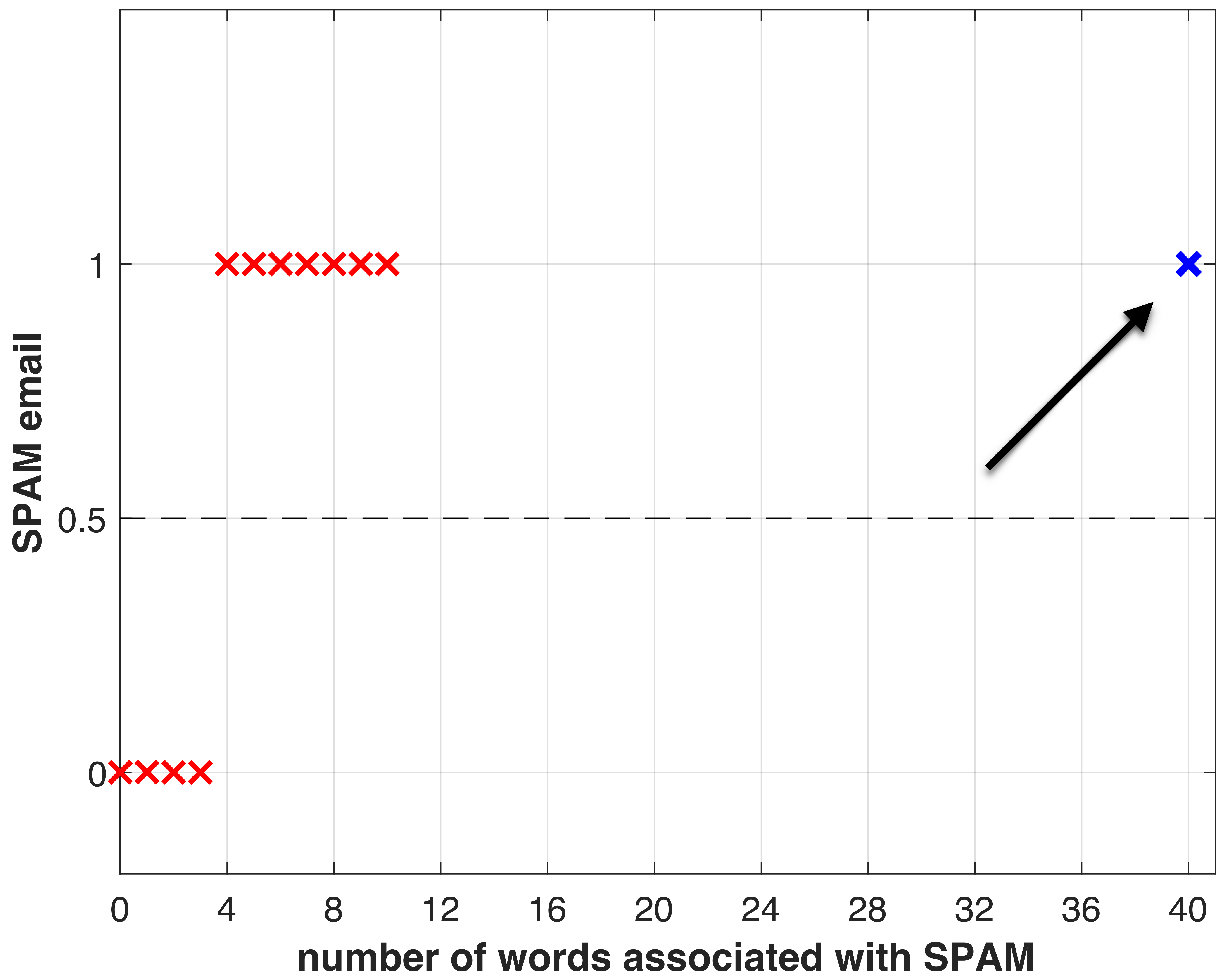
# Supervised learning — Binary classification

- Binary classification means that we only have two label categories, e.g.
  > *spam* vs. *not spam* email
  > *relevant* vs. *not relevant* document
- if $f_{\mathbf{w}}\left(x_i\right) \geq 0.5,$ then SPAM
  if $f_{\mathbf{w}}\left(x_i\right) < 0.5,$ then not SPAM
- What if we used OLS to learn $f_{\mathbf{w}}$?
  *Looks perfectly fine?*



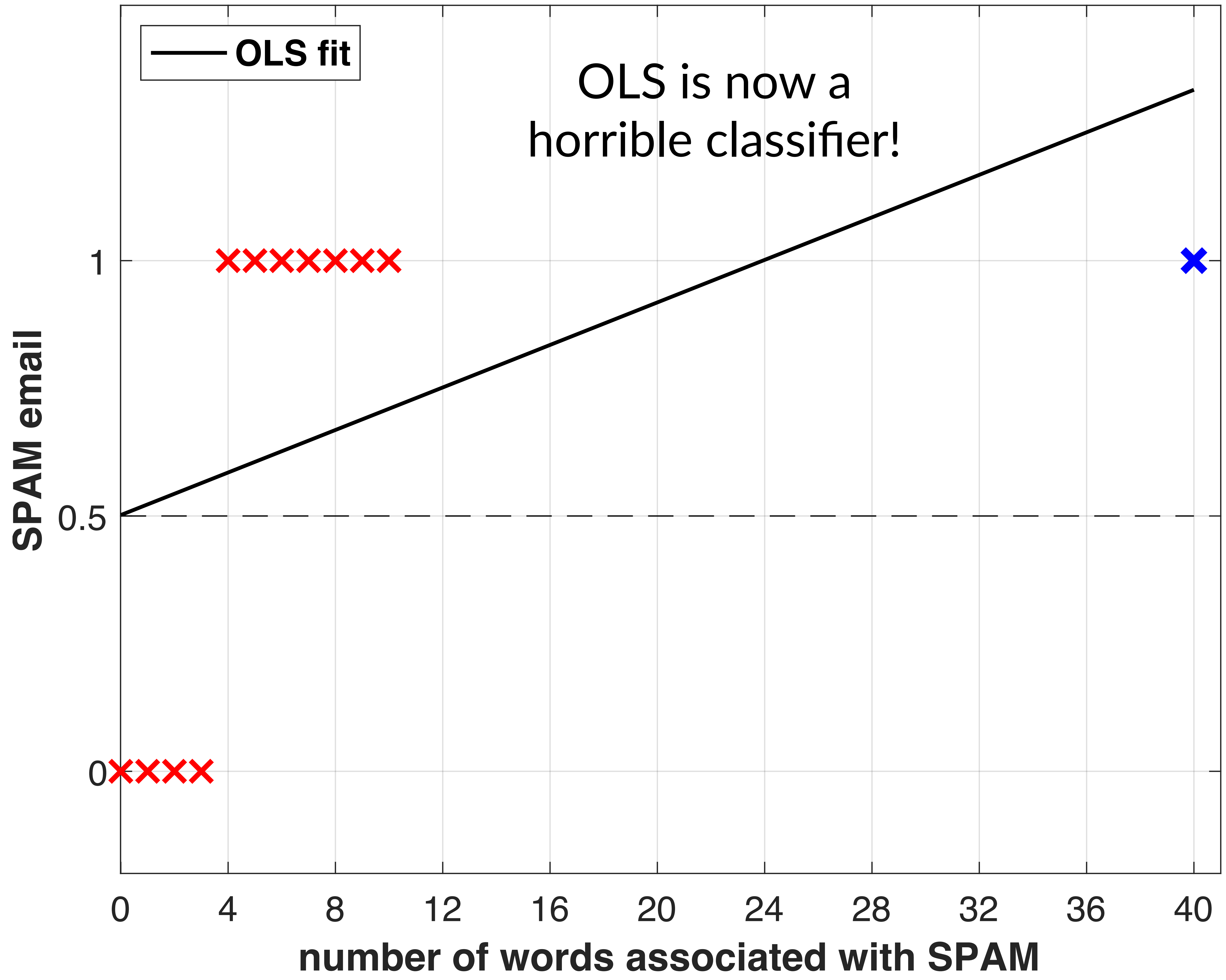
# Supervised learning — Binary classification

- **Binary classification** means that we only have two label categories, e.g.
  > *spam* vs. *not spam* email
  > *relevant* vs. *not relevant* document
- if $f_{\mathbf{w}}\left(x_i\right) \geq 0.5,$ then SPAM
  if $f_{\mathbf{w}}\left(x_i\right) < 0.5,$ then not SPAM
- What if we used OLS to learn $f_{\mathbf{w}}$? *Looks perfectly fine?*
- Let's add one more observation to our data. How would that affect our OLS classifier?



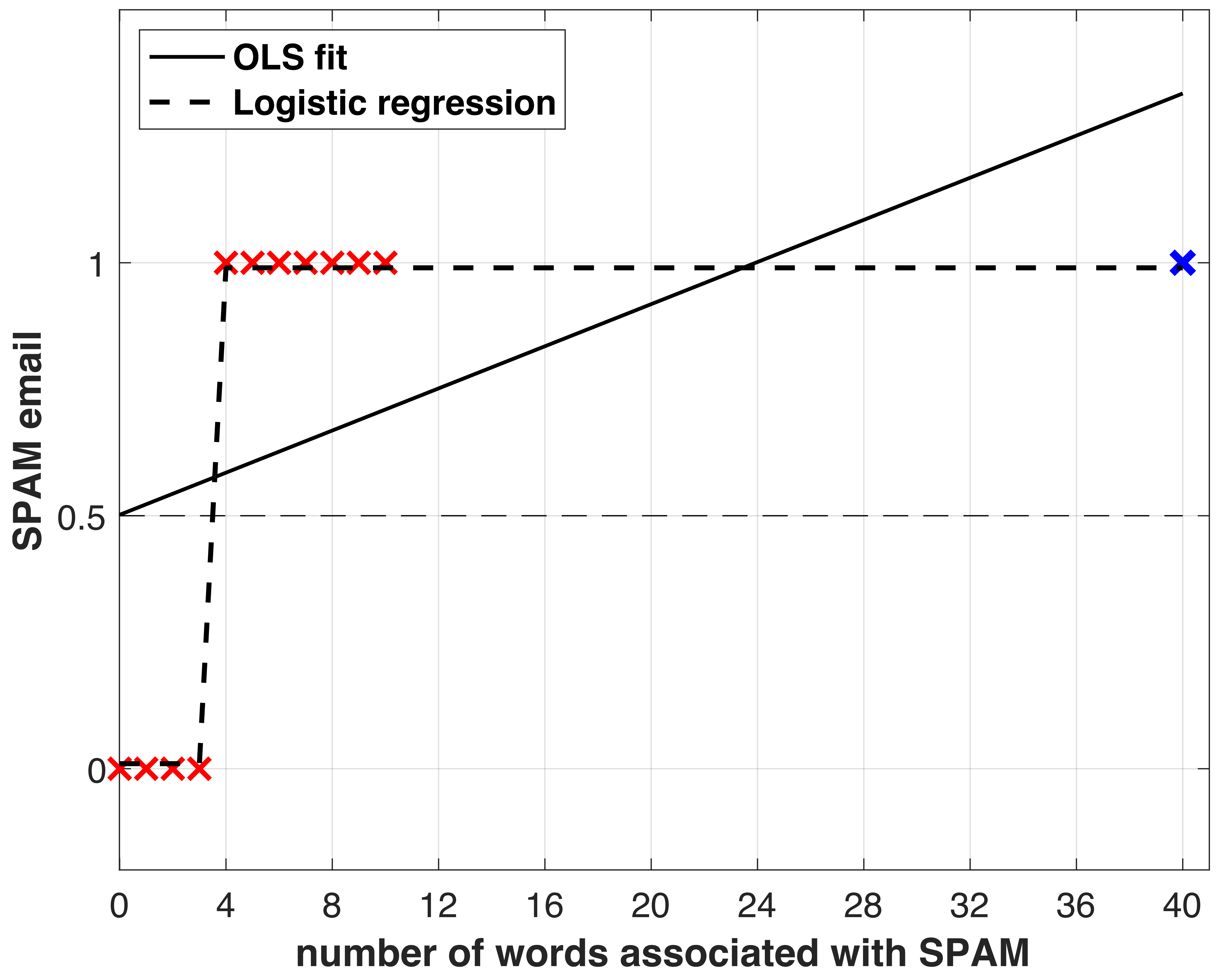
# Supervised learning — Binary classification

- **Binary classification** means that we only have two label categories, e.g.
  > *spam* vs. *not spam* email
  > *relevant* vs. *not relevant* document
- if $f_{\mathbf{w}}\left(x_i\right) \geq 0.5,$ then SPAM
  if $f_{\mathbf{w}}\left(x_i\right) < 0.5,$ then not SPAM
- What if we used OLS to learn $f_{\mathbf{w}}$?
  *Looks perfectly fine?*
- Let's add one more observation to our data. How would that affect our OLS classifier? **Not great!**



# Supervised learning — Binary classification

- **Binary classification** means that we only have two label categories, e.g.
  > *spam* vs. *not spam* email
  > *relevant* vs. *not relevant* document
- if $f_{\mathbf{w}}\left(x_i\right) \geq 0.5,$ then SPAM
  if $f_{\mathbf{w}}\left(x_i\right) < 0.5,$ then not SPAM
- What if we used OLS to learn $f_{\mathbf{w}}$?
  *Looks perfectly fine?*
- Let's add one more observation to our data. How would that affect our OLS classifier? **Not great!**
- It is not impossible to separate these classes — we just need a **different function**.



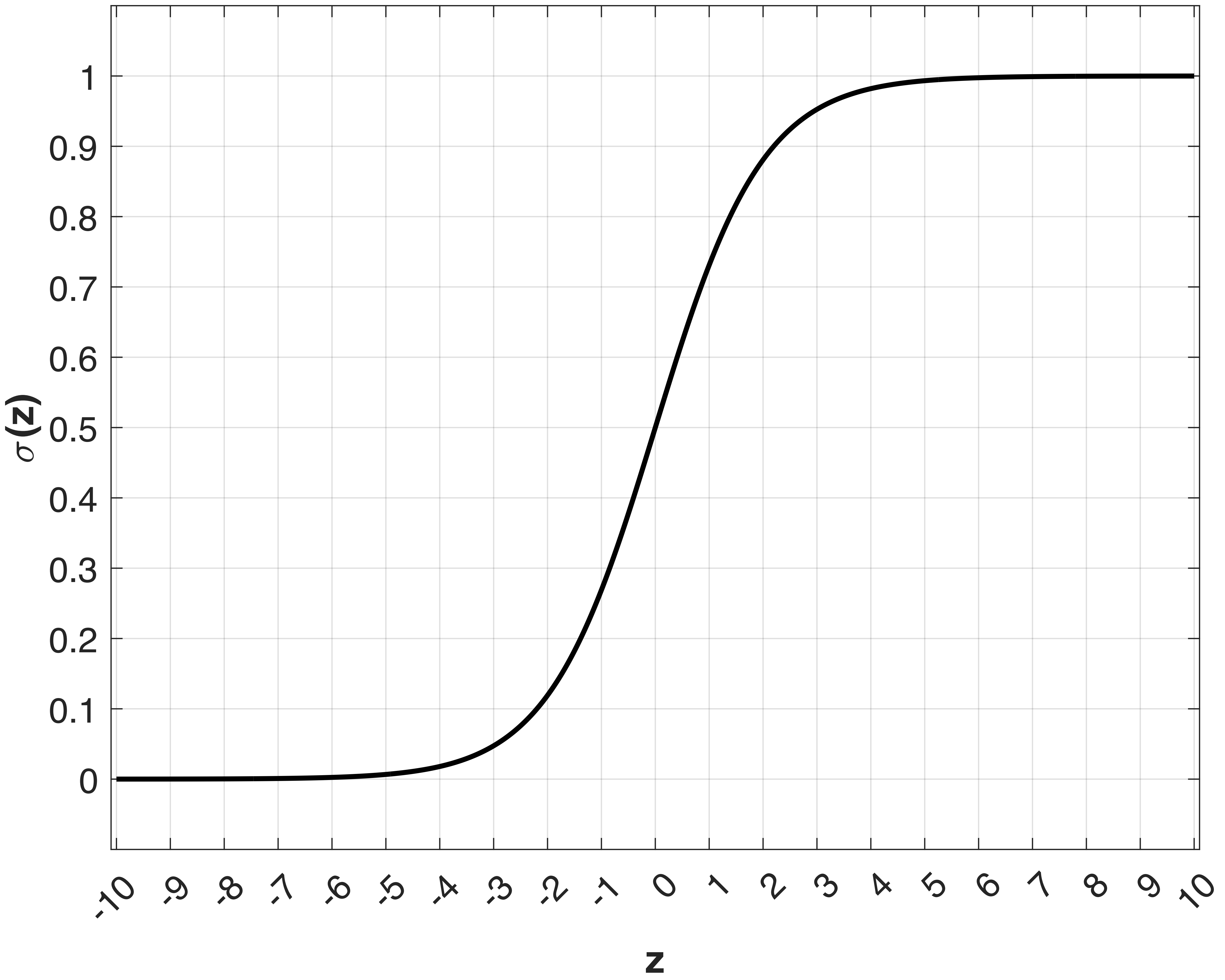
- Logistic regression is a classification method that learns a sigmoid separator
- Assume we have an $m$-dimensional observation $\mathbf{x} \in \mathbb{R}^m$
- We want $0 \leq f_{\mathbf{w}}(\mathbf{x}) \leq 1$, where $\mathbf{w} \in \mathbb{R}^m$ are the corresponding weights
- Sigmoid or logistic function

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

- $f_{\mathbf{w}}(\mathbf{x}) = \sigma\left(\mathbf{w}^\top \mathbf{x}\right) \in (0,1)$

  *it can be seen as a pseudo-probability*

**Sigmoid / logistic function**



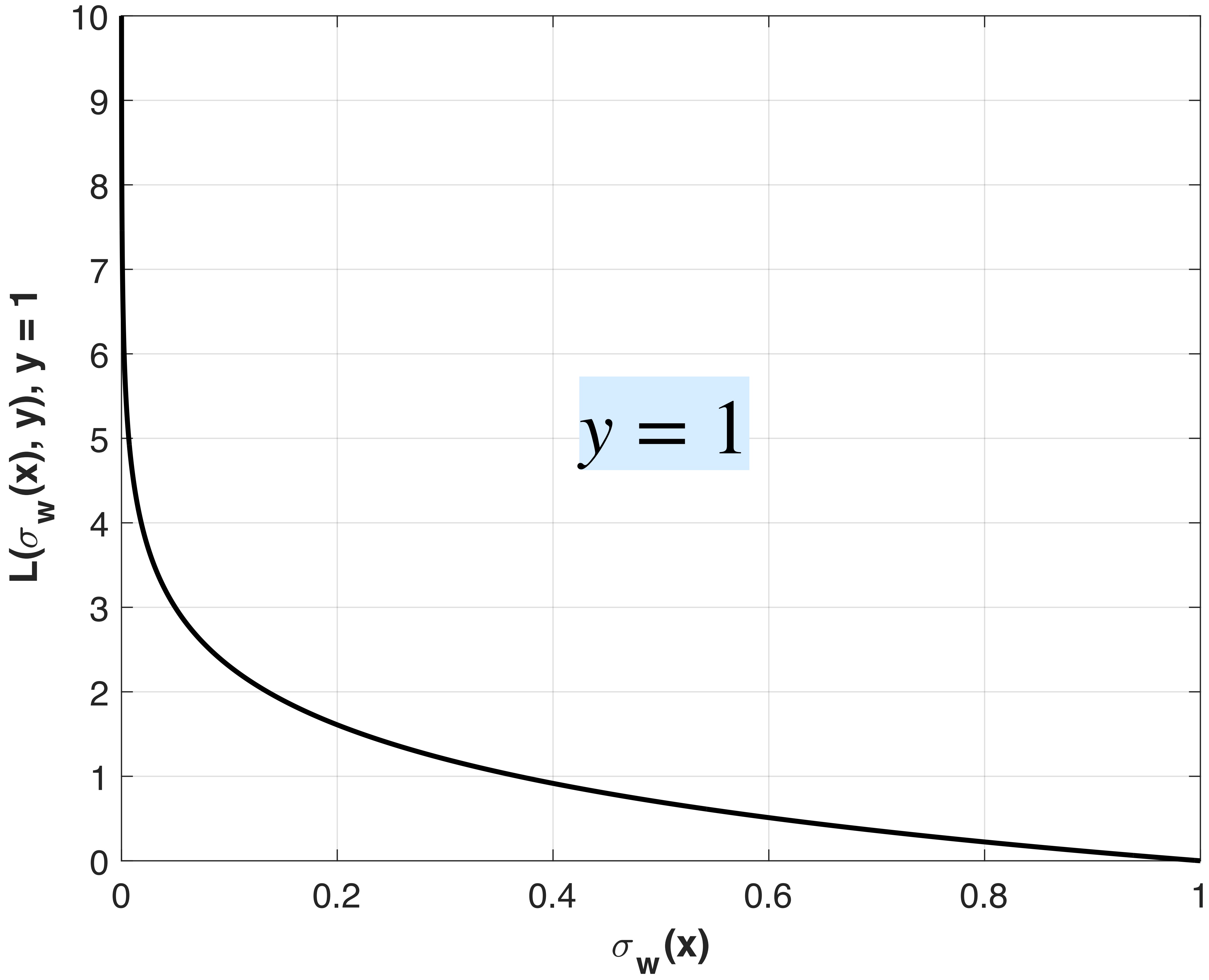
# Supervised learning — Logistic regression's loss function

- Logistic regression uses a cross-entropy loss function between the projection of $\mathbf{x}$ and label $y \in \{0,1\}$

- $\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(\sigma_{\mathbf{w}}(\mathbf{x})\right)$ if $y = 1$
  $\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(1 - \sigma_{\mathbf{w}}(\mathbf{x})\right)$ if $y = 0$

- Derivation from Bernoulli distribution (see SLP)

- Intuitively
  – we want a loss that is easy to differentiate
  – if $y = 1, \sigma_{\mathbf{w}}(\mathbf{x}) \rightarrow 1$: $\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) \rightarrow 0$
  – if $y = 1, \sigma_{\mathbf{w}}(\mathbf{x}) \rightarrow 0$: $\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) \rightarrow \infty$

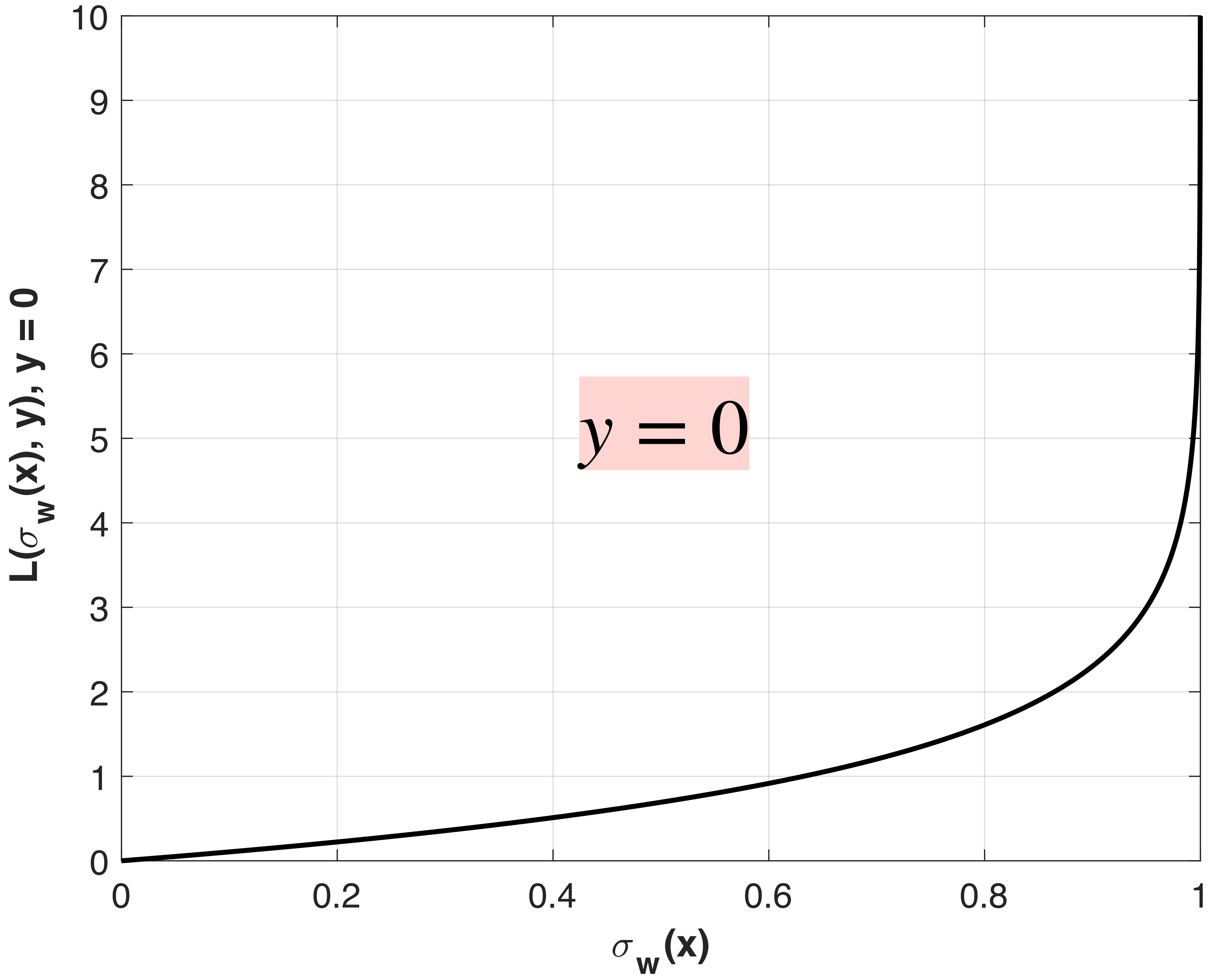


- Logistic regression uses a cross-entropy loss function between the projection of $\mathbf{x}$ and label $y \in \{0,1\}$

- $$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(\sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 1$$
  $$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(1 - \sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 0$$

- Derivation from Bernoulli distribution (see SLP)

- Intuitively
  – we want a loss that is easy to differentiate
  – if $y = 0$, $\sigma_{\mathbf{w}}(\mathbf{x}) \to 0$: $\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) \to 0$
  – if $y = 0$, $\sigma_{\mathbf{w}}(\mathbf{x}) \to 1$: $\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) \to \infty$



**Cross-entropy loss function**

$$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(\sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 1$$
$$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(1 - \sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 0$$

**Cross-entropy loss function**

$$\mathcal{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(\sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 1$$
$$\mathcal{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(1 - \sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 0$$

**Logistic (sigmoid) function**

$$\sigma_{\mathbf{w}}\left(\mathbf{x}_i\right) = \left(1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\right)^{-1}$$

**Cross-entropy loss function**

$$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(\sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 1$$
$$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(1 - \sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 0$$

**Logistic (sigmoid) function**

$$\sigma_{\mathbf{w}}\left(\mathbf{x}_i\right) = \left(1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\right)^{-1}$$

Our label $y_i$ is either $1$ or $0$ for all our observations. So, for each observation only one part of the loss function is activated / used. Since we have $n$ observations the loss function takes the form:

$$\mathscr{J}(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln \sigma_{\mathbf{w}}\left(\mathbf{x}_i\right) + \left(1 - y_i\right) \ln\left(1 - \sigma_{\mathbf{w}}\left(\mathbf{x}_i\right)\right)\right]$$

**Cross-entropy loss function**

$$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(\sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 1$$
$$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(1 - \sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 0$$

**Logistic (sigmoid) function**

$$\sigma_{\mathbf{w}}\left(\mathbf{x}_i\right) = \left(1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\right)^{-1}$$

**Combined loss function**

$$\mathscr{J}(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln \sigma_{\mathbf{w}}\left(\mathbf{x}_i\right) + \left(1 - y_i\right) \ln\left(1 - \sigma_{\mathbf{w}}\left(\mathbf{x}_i\right)\right)\right]$$

**Cross-entropy loss function**

$$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(\sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 1$$
$$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(1 - \sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 0$$

**Logistic (sigmoid) function**

$$\sigma_{\mathbf{w}}\left(\mathbf{x}_i\right) = \left(1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\right)^{-1}$$

**Combined loss function**

$$\mathscr{J}(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln \sigma_{\mathbf{w}}\left(\mathbf{x}_i\right) + \left(1 - y_i\right) \ln\left(1 - \sigma_{\mathbf{w}}\left(\mathbf{x}_i\right)\right)\right]$$

Let's incorporate the actual value of the sigmoid function and attempt to simplify:

$$\ln\left(\sigma_{\mathbf{w}}\left(\mathbf{x}_i\right)\right) = \ln(1) - \ln\left(1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\right) = -\ln\left(1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\right)$$

The loss function becomes:

$$\mathscr{J}(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{w}^\top \mathbf{x}_i y_i - \mathbf{w}^\top \mathbf{x}_i - \ln\left(1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\right)\right]$$

# Supervised learning — Logistic regression with gradient descent

**Cross-entropy loss function**

$$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(\sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 1$$
$$\mathscr{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(1 - \sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 0$$

**Logistic (sigmoid) function**

$$\sigma_{\mathbf{w}}\left(\mathbf{x}_i\right) = \left(1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\right)^{-1}$$

**Combined loss function**

$$\mathscr{J}(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{w}^\top \mathbf{x}_i y_i - \mathbf{w}^\top \mathbf{x}_i - \ln\left(1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\right)\right]$$

**Partial derivative**

$$\frac{\partial \mathscr{J}(\mathbf{w})}{\partial \mathbf{w}_j} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i x_{i,j} - x_{i,j} + e^{-\mathbf{w}^\top \mathbf{x}_i}\left(1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\right)^{-1} x_{i,j}\right]$$
$$= -\frac{1}{n}\sum_{i=1}^{n}\left[x_{i,j}\left(y_i - \sigma_{\mathbf{w}}\left(\mathbf{x}_i\right)\right)\right]$$

**Cross-entropy loss function**

$$\mathcal{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(\sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 1$$

$$\mathcal{L}\left(\sigma_{\mathbf{w}}(\mathbf{x}), y\right) = -\ln\left(1 - \sigma_{\mathbf{w}}(\mathbf{x})\right) \quad \text{if } y = 0$$

**Logistic (sigmoid) function**

$$\sigma_{\mathbf{w}}\left(\mathbf{x}_i\right) = \left(1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\right)^{-1}$$

**Combined loss function**

$$\mathcal{J}(\mathbf{w}) \ldots 1 + e^{-\mathbf{w}^\top \mathbf{x}_i}\Big)\Big]$$

**Partial derivative**

$$\frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}_j} = \ldots \sum_{i=1} \Big[ \ldots + e^{-\mathbf{w}^\top \mathbf{x}_i}\Big)^{-1} x_{i,j}\Big]$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[x_{i,j}\left(y_i - \sigma_{\mathbf{w}}\left(\mathbf{x}_i\right)\right)\right]$$

The rest is identical to the least squares example, i.e. initialise $\mathbf{w}$, then compute the partial derivatives for each $w_j$, then update $w_j$'s using a learning rate, and repeat until convergence.

# Supervised learning — Logistic regression, example

- Going back to the application of estimating flu prevalence using web search activity

- Now, we want to use the frequency of $4$ search queries to predict whether the flu rate in a population is above a low-epidemic threshold or not
  – binary classification task
  – $y_i = 1$, if the flu rate is above a low-epidemic threshold
  – $y_i = 0$, if the flu rate is below or equal to a low-epidemic threshold

- We have in total $104$ weekly observations
  – observation matrix $\mathbf{X} \in \mathbb{R}^{104 \times 4}$
  – queries: "*how long does flu last*", "*flu symptoms*", "*cough flu*", "*flu recovery*"
  – labels $\mathbf{y} \in \{0,1\}^{104}$

# Supervised learning — Logistic regression, example

- We have in total $104$ weekly observations
  – observation matrix $\mathbf{X} \in \mathbb{R}^{104 \times 4}$
  – queries: "*how long does flu last*", "*flu symptoms*", "*cough flu*", "*flu recovery*"
  – labels $\mathbf{y} \in \{0,1\}^{104}$

- $4$-fold cross validation to assess classification performance
  – form $4$ folds (equally sized baskets) of the data,
  – train a classifier using $3$ of them, test (evaluate) on the remaining $1$
  – report average performance metrics

# Binary classification — Basic performance metrics

- $\hat{\mathbf{y}} \in \{0,1\}^n$ denotes our predictions and $\mathbf{y} \in \{0,1\}^n$ the correct labels
- $\text{accuracy} = \dfrac{\text{number of times } \hat{y}_i = y}{n}$
- $\text{precision} = \dfrac{\text{number of times } \hat{y}_i = 1 \text{ AND } \hat{y}_i = y_i}{\text{number of times } \hat{y}_i = 1}$

When we predicted a positive class, how often did we get it right?

- $\text{recall} = \dfrac{\text{number of times } \hat{y}_i = 1 \text{ AND } \hat{y}_i = y_i}{\text{number of times } y_i = 1}$

How often did we predict the positive class correctly relatively to all samples that were positive?

- $\text{F}_1$ score is the harmonic mean between precision and recall

$$\text{F}_1 \text{ score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

# Supervised learning — Logistic regression, example

**logistic regression performance metrics**

$$\text{accuracy} = 0.923\,(0.070)$$
$$\text{precision} = 0.902\,(0.121)$$
$$\text{recall} = 0.844\,(0.120)$$
$$\text{F}_1\ \text{score} = 0.871(0.116)$$

**logistic regression weights using all data**

$$\text{flu symptoms:}\ 78.058$$
$$\text{how long does flu last:}\ 24.537$$
$$\text{flu recovery:}\ 3.8977$$
$$\text{cough flu:}\ -14.663$$

*Download the data from*
*dropbox.com/s/rgyg190whw26qrj/data-COMP0084-intro-to-ml.zip?dl=0*
*and try it yourself...*

# Multi-class classification

- Binary classification is the simplest classification case — we often have more than two labels, i.e. most tasks require multi-class classification

- We can use different classifiers (machine learning models) that support multi-class classification such as neural network architectures and generative models

- We can also use a binary classifier
  - — **one vs. rest** strategy: $n$ classes require $n$ classifiers to be trained
    highest score determines the classification label
  - — **one vs. one** strategy: $n$ classes require $\dfrac{n(n-1)}{2}$ classifiers to be trained
    voting scheme, class with the most votes wins

# Common machine learning categorisation

- **Supervised learning**
  Learn a mapping $f$ from inputs $\mathbf{X}$ to outputs $\mathbf{y}$ – also can be expressed by $f : \mathbf{X} \rightarrow \mathbf{y}$
  – $\mathbf{X}$ are also called features, observations, covariates, predictors
  – $\mathbf{y}$ are also called labels, targets, responses, ground truth
  – $< \mathbf{X}, \mathbf{y} >$ can also be referred to as observations or samples

- **Unsupervised learning**
  No outputs associated with the input $\mathbf{X}$ – the task becomes to discover an underlying structure or patterns in $\mathbf{X}$

- **Reinforcement learning**
  The system or agent has to learn how to interact with its environment
  Policy: which action to take in response to an input $\mathbf{X}$
  Different from supervised learning because no definitive responses are given
  Only rewards – *learning with a critic as opposed to learning with a teacher*

# Unsupervised learning

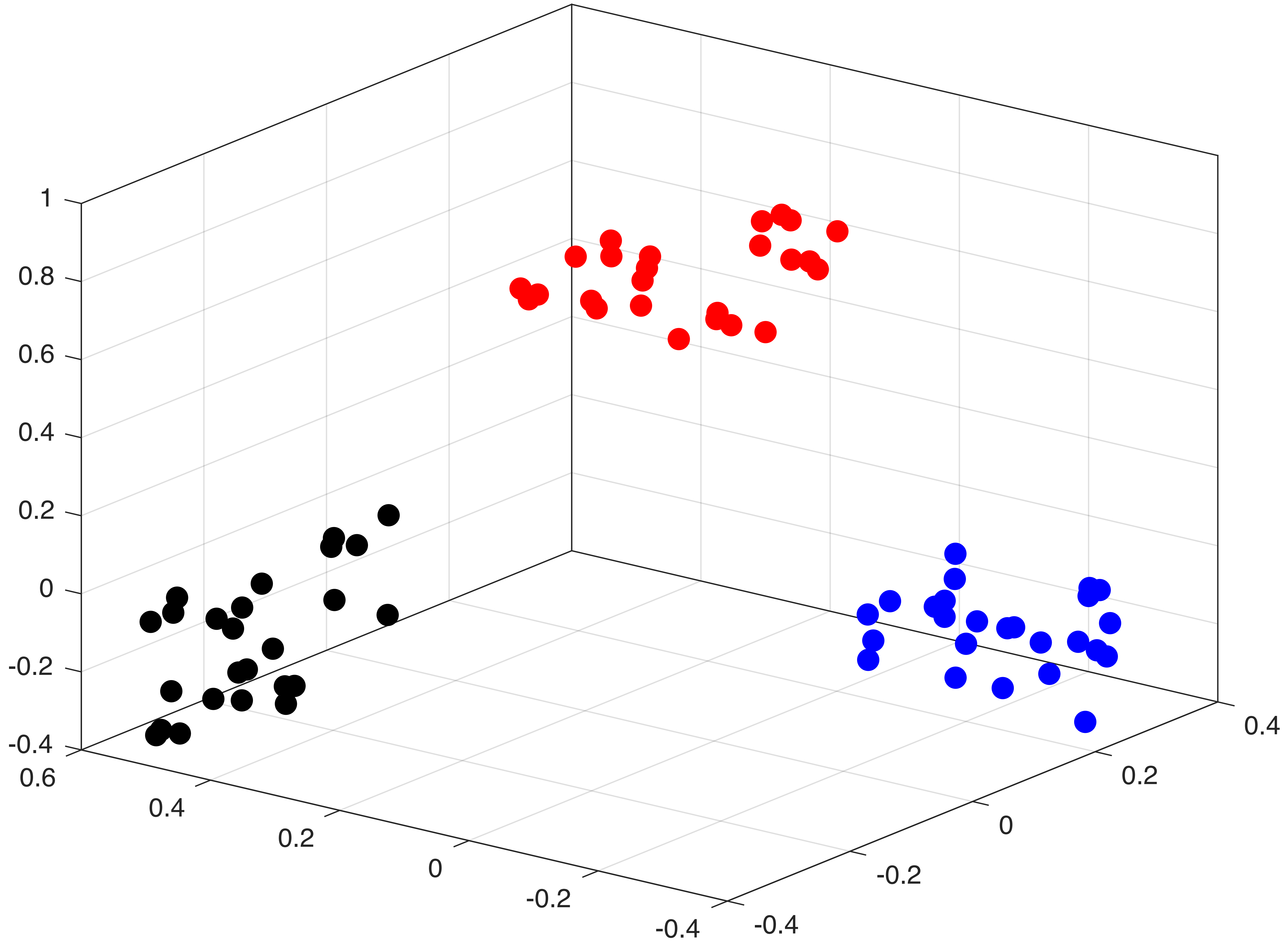
- In the previous machine learning paradigms we had an input $\mathbf{X}$ and an output $\mathbf{y}$ and we wanted to learn $f : \mathbf{X} \rightarrow \mathbf{y}$
- In unsupervised learning, there are no particular outputs or, better, response variables that we can associate our inputs with
- Our goal now is different: we want to extract some kind of pattern (a rule, an intrinsic structure) from a data set (a set of observations $\mathbf{X}$)

# Unsupervised learning

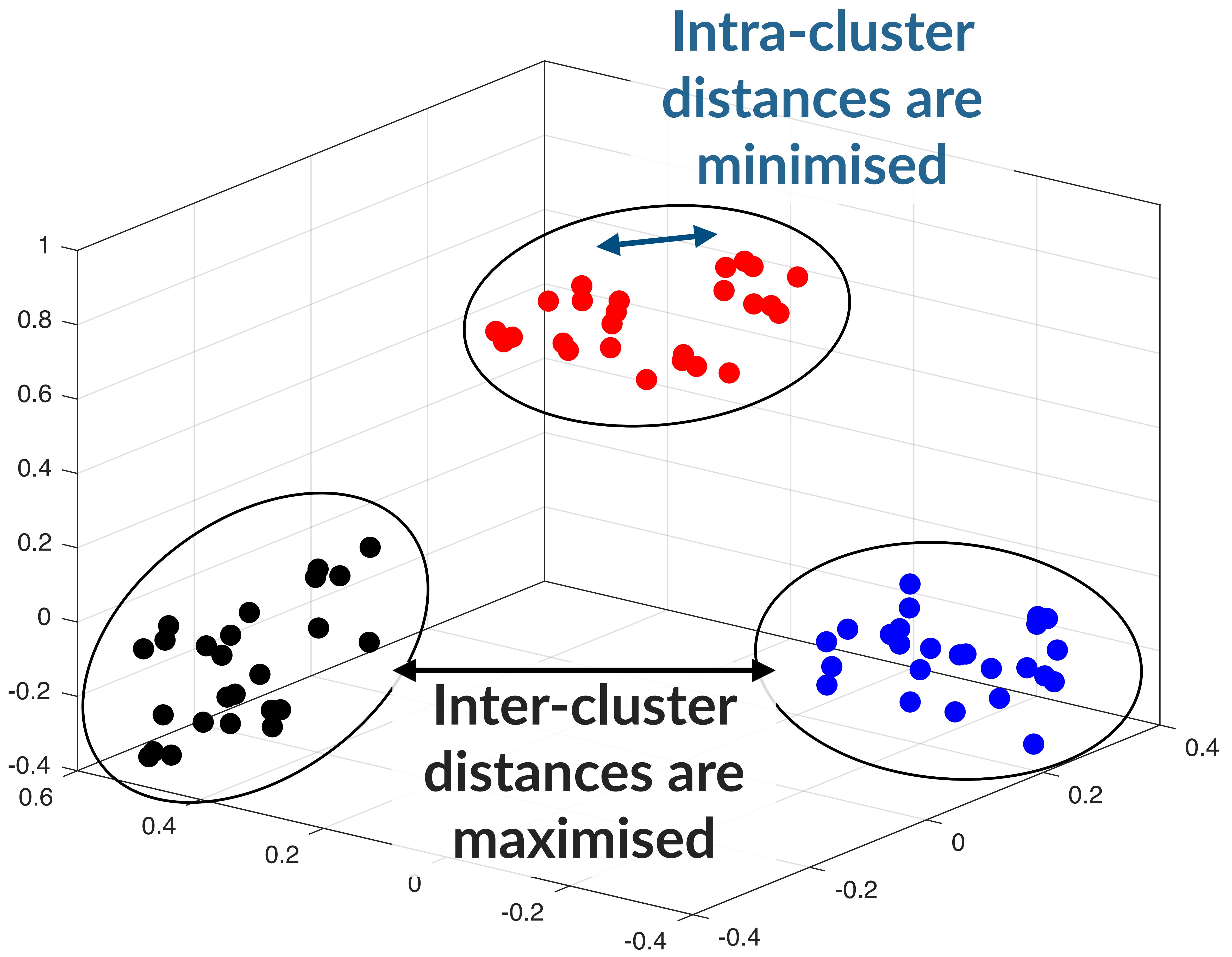
- Is association rule mining a form of unsupervised learning? *Yes, it is!*
- Some unsupervised learning methods are quite common statistical operations, e.g. dimensionality reduction methods, principal component analysis
- In machine learning, unsupervised learning is almost synonymous to clustering
- Clustering aims to group similar observations (or features) together into... **clusters!**

# Unsupervised learning — Clustering

- A **cluster** contains data instances that are similar to each other (or if you visualise this are very close to each other in a vector space) — in very lay terms, different clusters are supposed to be capturing a different part of this vector space
- So, clustering is a grouping of data objects such that the objects within a group are similar (or related) to one another and different from (or unrelated to) the objects in other groups
- The plot shows 3 very visible clusters



- A **cluster** contains data instances that are similar to each other (or if you visualise this are very close to each other in a vector space) — in very lay terms, different clusters are supposed to be capturing a different part of this vector space
- So, clustering is a grouping of data objects such that the objects within a group are similar (or related) to one another and different from (or unrelated to) the objects in other groups
- The plot shows 3 very visible clusters



# Clustering — Where is it being used?

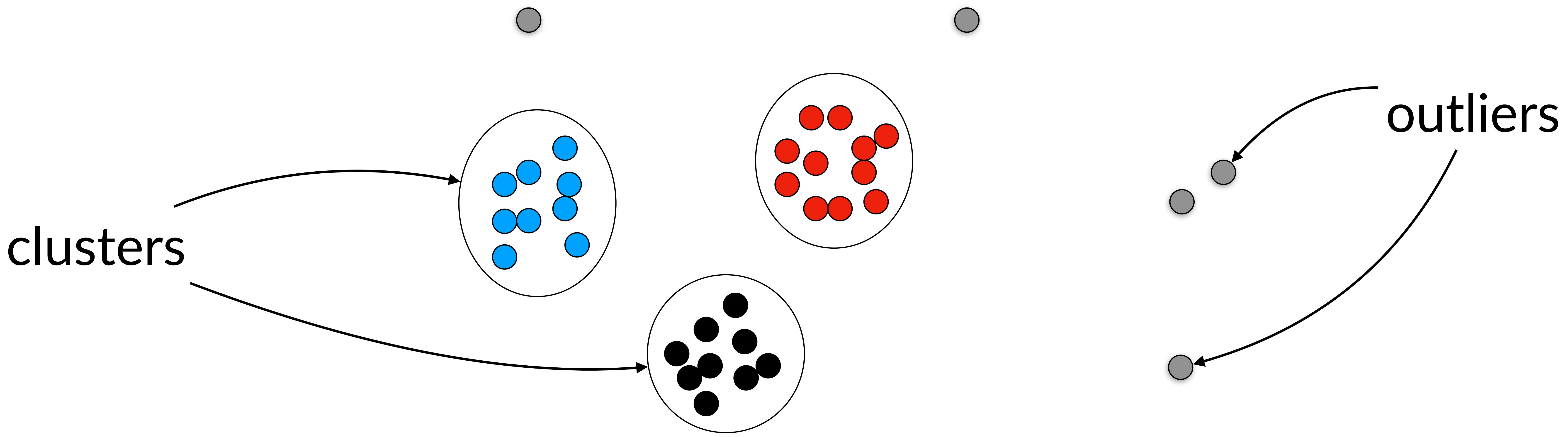
- Computational biology, e.g. understand properties of genes
- Medicine, e.g. in medical imaging
- Marketing, e.g. segment customers according to their underlying characteristics, then conduct targeted marketing
- Document clustering, topic models, text clustering in general
- Applicable to tasks that require "pattern analysis" and in many different research disciplines for analysing outcomes (e.g. in psychology, sociology, computer science, neuroscience)

# Clustering — Some further key aspects

- Many different clustering algorithms / methods
  — partitional, hierarchical, hard/soft, generative, and even supervised
- A distance (dissimilarity) or a similarity function is often a key component for determining clusters
- Clustering goal is to maximise the distance between different clusters (inter-cluster distance) and at the same time to minimise the distance of elements in a cluster (intra-cluster distance)
- The quality of a clustering outcome depends on the algorithm, the distance function, and eventually the specifics of an application
- However, determining the actual quality of a cluster is not always an easy task given the lack of supervision

# Clustering — Outliers

- Outliers are objects that do not belong to any cluster or form clusters of very small cardinality
- In some applications (e.g. fraud detection) we are actually interested in discovering outliers, not clusters



# Clustering — Distance / similarity functions

# Clustering — Distance / similarity functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z}$

# Clustering — Distance / similarity functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z}$
- Let's also first assume that both $\mathbf{x}$ and $\mathbf{z}$ contain discrete values; these can be binary values ($0$ or $1$), or specific element identifiers

# Clustering — Distance / similarity functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z}$
- Let's also first assume that both $\mathbf{x}$ and $\mathbf{z}$ contain discrete values; these can be binary values ($0$ or $1$), or specific element identifiers
- The Jaccard similarity provides a simple way to compare these observations

$$\mathrm{Jsim}\left(\mathbf{x}, \mathbf{z}\right) = \frac{|\mathbf{x} \cap \mathbf{z}|}{|\mathbf{x} \cup \mathbf{z}|}$$

# Clustering — Distance / similarity functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z}$
- Let's also first assume that both $\mathbf{x}$ and $\mathbf{z}$ contain discrete values; these can be binary values ($0$ or $1$), or specific element identifiers
- The Jaccard similarity provides a simple way to compare these observations

$$\mathrm{Jsim}\left(\mathbf{x}, \mathbf{z}\right) = \frac{|\,\mathbf{x} \cap \mathbf{z}\,|}{|\,\mathbf{x} \cup \mathbf{z}\,|}$$

- Comparisons are element-wise

# Clustering — Distance / similarity functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z}$
- Let's also first assume that both $\mathbf{x}$ and $\mathbf{z}$ contain discrete values; these can be binary values ($0$ or $1$), or specific element identifiers
- The Jaccard similarity provides a simple way to compare these observations
$$\mathrm{Jsim}\left(\mathbf{x}, \mathbf{z}\right) = \frac{|\,\mathbf{x} \cap \mathbf{z}\,|}{|\,\mathbf{x} \cup \mathbf{z}\,|}$$
- Comparisons are element-wise
- The Jaccard similarity can (by definition) take values from $0$ to $1$

# Clustering — Distance / similarity functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z}$
- Let's also first assume that both $\mathbf{x}$ and $\mathbf{z}$ contain discrete values; these can be binary values ($0$ or $1$), or specific element identifiers
- The Jaccard similarity provides a simple way to compare these observations
$$\mathrm{Jsim}(\mathbf{x}, \mathbf{z}) = \frac{|\mathbf{x} \cap \mathbf{z}|}{|\mathbf{x} \cup \mathbf{z}|}$$
- Comparisons are element-wise
- The Jaccard similarity can (by definition) take values from $0$ to $1$
- It can be turned to a distance metric: $\mathrm{Jdist}(\mathbf{x}, \mathbf{z}) = 1 - \mathrm{Jsim}(\mathbf{x}, \mathbf{z})$

# Clustering — Distance / similarity functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z}$
- Let's also first assume that both $\mathbf{x}$ and $\mathbf{z}$ contain discrete values; these can be binary values ($0$ or $1$), or specific element identifiers
- The Jaccard similarity provides a simple way to compare these observations

$$\mathrm{Jsim}\left(\mathbf{x}, \mathbf{z}\right) = \frac{|\,\mathbf{x} \cap \mathbf{z}\,|}{|\,\mathbf{x} \cup \mathbf{z}\,|}$$

- Comparisons are element-wise
- The Jaccard similarity can (by definition) take values from $0$ to $1$
- It can be turned to a distance metric: $\mathrm{Jdist}\left(\mathbf{x}, \mathbf{z}\right) = 1 - \mathrm{Jsim}\left(\mathbf{x}, \mathbf{z}\right)$
- if $\mathbf{x} = \begin{bmatrix} 1\,0\,0\,1\,1\,1 \end{bmatrix}$, $\mathbf{z} = \begin{bmatrix} 0\,1\,1\,0\,1\,0 \end{bmatrix}$ then $\mathrm{Jsim}\left(\mathbf{x}, \mathbf{z}\right) = 1/6$ and $\mathrm{Jdist}\left(\mathbf{x}, \mathbf{z}\right) = 5/6$

# Clustering — Distance / similarity functions

# Clustering — Distance / similarity functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z}$
- Let's now assume that both $\mathbf{x}$ and $\mathbf{z} \in \mathbb{R}^n$

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z}$
- Let's now assume that both $\mathbf{x}$ and $\mathbf{z} \in \mathbb{R}^n$
- Recall the $L_p$-norm definition: $||\mathbf{x}||_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$

# Clustering — Distance / similarity functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z}$
- Let's now assume that both $\mathbf{x}$ and $\mathbf{z} \in \mathbb{R}^n$
- Recall the $L_p$-norm definition: $\|\mathbf{x}\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$
- Popular distance measures stem from this — the input now is the difference of the vectors we want to compare — this is also known as the **Minkowski distance**
$$L_p(\mathbf{x}, \mathbf{z}) = \left( |x_1 - z_1|^p + |x_2 - z_2|^p + \ldots + |x_n - z_n|^p \right)^{1/p} = \|\mathbf{x} - \mathbf{z}\|_p$$

# Clustering — Distance / similarity functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z} \in \mathbb{R}^n$
- **Minkowski distance**

$$L_p(\mathbf{x}, \mathbf{z}) = \left( |x_1 - z_1|^p + |x_2 - z_2|^p + \ldots + |x_n - z_n|^p \right)^{1/p} = \|\mathbf{x} - \mathbf{z}\|_p$$

- For different values of $p \in \mathbb{N}_{>0}$ we can obtain common distance functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z} \in \mathbb{R}^n$
- **Minkowski distance**

  $$L_p(\mathbf{x}, \mathbf{z}) = \left( |x_1 - z_1|^p + |x_2 - z_2|^p + \ldots + |x_n - z_n|^p \right)^{1/p} = \|\mathbf{x} - \mathbf{z}\|_p$$
- For different values of $p \in \mathbb{N}_{>0}$ we can obtain common distance functions
- $p = 1$, **Manhattan** or **city block distance** or $L_1$-norm

  $$L_1(\mathbf{x}, \mathbf{z}) = |x_1 - z_1| + |x_2 - z_2| + \ldots + |x_n - z_n|$$

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z} \in \mathbb{R}^n$
- **Minkowski distance**

  $$L_p(\mathbf{x}, \mathbf{z}) = \left( |x_1 - z_1|^p + |x_2 - z_2|^p + \ldots + |x_n - z_n|^p \right)^{1/p} = \|\mathbf{x} - \mathbf{z}\|_p$$

- For different values of $p \in \mathbb{N}_{>0}$ we can obtain common distance functions
- $p = 1$, **Manhattan** or **city block distance** or $L_1$-norm

  $$L_1(\mathbf{x}, \mathbf{z}) = |x_1 - z_1| + |x_2 - z_2| + \ldots + |x_n - z_n|$$

- $p = 2$, **Euclidean distance** or $L_2$-norm

  $$L_2(\mathbf{x}, \mathbf{z}) = \left[ (x_1 - z_1)^2 + (x_2 - z_2)^2 + \ldots + (x_n - z_n)^2 \right]^{1/2} = \sqrt{(x_1 - z_1)^2 + (x_2 - z_2)^2 + \ldots + (x_n - z_n)^2}$$

# Clustering — Distance / similarity functions

# Clustering — Distance / similarity functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z} \in \mathbb{R}^n$
- **Minkowski distance**

$$L_p(\mathbf{x}, \mathbf{z}) = \left( |x_1 - z_1|^p + |x_2 - z_2|^p + \ldots + |x_n - z_n|^p \right)^{1/p} = \|\mathbf{x} - \mathbf{z}\|_p$$

- For different values of $p \in \mathbb{N}_{>0}$ we can obtain common distance functions

- Let's assume we want to compare two $n$-dimensional observations, $\mathbf{x}$ and $\mathbf{z} \in \mathbb{R}^n$
- **Minkowski distance**

$$L_p(\mathbf{x}, \mathbf{z}) = \left( |x_1 - z_1|^p + |x_2 - z_2|^p + \ldots + |x_n - z_n|^p \right)^{1/p} = \|\mathbf{x} - \mathbf{z}\|_p$$

- For different values of $p \in \mathbb{N}_{>0}$ we can obtain common distance functions
- We can also define weighted distances, if we want to give more importance to certain features, e.g.

$$L_2(\mathbf{x}, \mathbf{z}) = \sqrt{w_1 \left(x_1 - z_1\right)^2 + w_2 \left(x_2 - z_2\right)^2 + \ldots + w_n \left(x_n - z_n\right)^2}$$

# Clustering — $k$-means

- There exists a plethora of different approaches to clustering
— relation between objects and classes (exclusive vs. overlapping)
— relation between classes and classes (ordered vs. flat)

- Today we are going to see the clustering algorithm $k$**-means**: driven by the relationship to cluster representatives (or means), partitional clustering algorithm

- $k$-means constructs a partition of a set of $n$ features (objects) into a set of $k$ clusters
— each object belongs to exactly one cluster (hard clustering)
— the number of clusters ($k$) is a setting given in advance

# Clustering – $k$-means

# Clustering — $k$-means

- Let's assume we have a set of $n$ $m$-dimensional observations, i.e. a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$
  — the number of dimensions = number of features $(m)$
  — a feature $i$ is represented by the $i$-th column of $\mathbf{X}$, the $n$-dimensional vector $\mathbf{x}_{:,i}$
  — we want to partition the $m$ features (or columns) of $\mathbf{X}$ into $k$ clusters

# Clustering — $k$-means

- Let's assume we have a set of $n$ $m$-dimensional observations, i.e. a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$
  — the number of dimensions = number of features $(m)$
  — a feature $i$ is represented by the $i$-th column of $\mathbf{X}$, the $n$-dimensional vector $\mathbf{x}_{:,i}$
  — we want to partition the $m$ features (or columns) of $\mathbf{X}$ into $k$ clusters

1. Randomly set $k$ data points (seed observations) to be the initial cluster centres. We call these centres **centroids** and in practice they are $n$-dimensional vectors (same size as the columns of $\mathbf{X}$). Centroid $j$ is denoted by $\mathbf{c}_j \in \mathbb{R}^n$

# Clustering — $k$-means

- Let's assume we have a set of $n$ $m$-dimensional observations, i.e. a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$
  — the number of dimensions = number of features $(m)$
  — a feature $i$ is represented by the $i$-th column of $\mathbf{X}$, the $n$-dimensional vector $\mathbf{x}_{:,i}$
  — we want to partition the $m$ features (or columns) of $\mathbf{X}$ into $k$ clusters

1. Randomly set $k$ data points (seed observations) to be the initial cluster centres. We call these centres **centroids** and in practice they are $n$-dimensional vectors (same size as the columns of $\mathbf{X}$). Centroid $j$ is denoted by $\mathbf{c}_j \in \mathbb{R}^n$
2. Assign each feature (*column*) $\mathbf{x}_{:,i}$ to its closest centroid $\mathbf{c}_j$

# Clustering — $k$-means

- Let's assume we have a set of $n$ $m$-dimensional observations, i.e. a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$
  — the number of dimensions = number of features $(m)$
  — a feature $i$ is represented by the $i$-th column of $\mathbf{X}$, the $n$-dimensional vector $\mathbf{x}_{:,i}$
  — we want to partition the $m$ features (or columns) of $\mathbf{X}$ into $k$ clusters

1. Randomly set $k$ data points (seed observations) to be the initial cluster centres. We call these centres **centroids** and in practice they are $n$-dimensional vectors (same size as the columns of $\mathbf{X}$). Centroid $j$ is denoted by $\mathbf{c}_j \in \mathbb{R}^n$
2. Assign each feature (*column*) $\mathbf{x}_{:,i}$ to its closest centroid $\mathbf{c}_j$
3. Re-compute centroids by averaging across their members

# Clustering — $k$-means

- Let's assume we have a set of $n$ $m$-dimensional observations, i.e. a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$
  — the number of dimensions = number of features $(m)$
  — a feature $i$ is represented by the $i$-th column of $\mathbf{X}$, the $n$-dimensional vector $\mathbf{x}_{:,i}$
  — we want to partition the $m$ features (or columns) of $\mathbf{X}$ into $k$ clusters

1. Randomly set $k$ data points (seed observations) to be the initial cluster centres. We call these centres **centroids** and in practice they are $n$-dimensional vectors (same size as the columns of $\mathbf{X}$). Centroid $j$ is denoted by $\mathbf{c}_j \in \mathbb{R}^n$
2. Assign each feature (*column*) $\mathbf{x}_{:,i}$ to its closest centroid $\mathbf{c}_j$
3. Re-compute centroids by averaging across their members
4. If a convergence criterion is not met (*see next slide!*), go back to step 2

# Clustering – $k$-means convergence criteria

# Clustering — $k$-means convergence criteria

- no or minimum re-assignments of data points to different clusters

# Clustering — $k$-means convergence criteria

- no or minimum re-assignments of data points to different clusters
- no or minimum change of centroids

# Clustering — $k$-means convergence criteria

- no or minimum re-assignments of data points to different clusters
- no or minimum change of centroids
- minimum decrease in the following cost function — the distance of all features from their centroids has converged to a minimum ($C_j$ denotes cluster $j$)

$$\sum_{j=1}^{k} \sum_{\mathbf{x}_{:,i} \in C_j} \operatorname{dist}\left(\mathbf{x}_{:,i}, \mathbf{c}_j\right)$$

# Clustering — $k$-means convergence criteria

- no or minimum re-assignments of data points to different clusters
- no or minimum change of centroids
- minimum decrease in the following cost function — the distance of all features from their centroids has converged to a minimum ($C_j$ denotes cluster $j$)

$$\sum_{j=1}^{k} \sum_{\mathbf{x}_{:,i} \in C_j} \mathrm{dist}\left(\mathbf{x}_{:,i}, \mathbf{c}_j\right)$$

- we can use different distance functions, the most common being the Euclidean distance squared, i.e.

$$\sum_{j=1}^{k} \sum_{\mathbf{x}_{:,i} \in C_j} \|\mathbf{x}_{:,i} - \mathbf{c}_j\|_2^2$$

# Clustering — $k$-means

- Strengths
  — simple implementation
  — efficient, time complexity $\mathcal{O}\left(t \cdot k \cdot n\right)$, $t$ number of iterations, $k$ clusters, $n$ observations
  — finds a local optimum
  — *no definitive evidence that any other cluster algorithm performs better* (hard to evaluate!)

- Weaknesses
  — we need to specify $k$ (the number of clusters)
  — sensitive to outliers
  — sensitive to initialisation

- Workarounds / improvements
  — multiple runs with different initialisations
  — non random initialisation, centroids set to the most distant observations ($k$-means++)

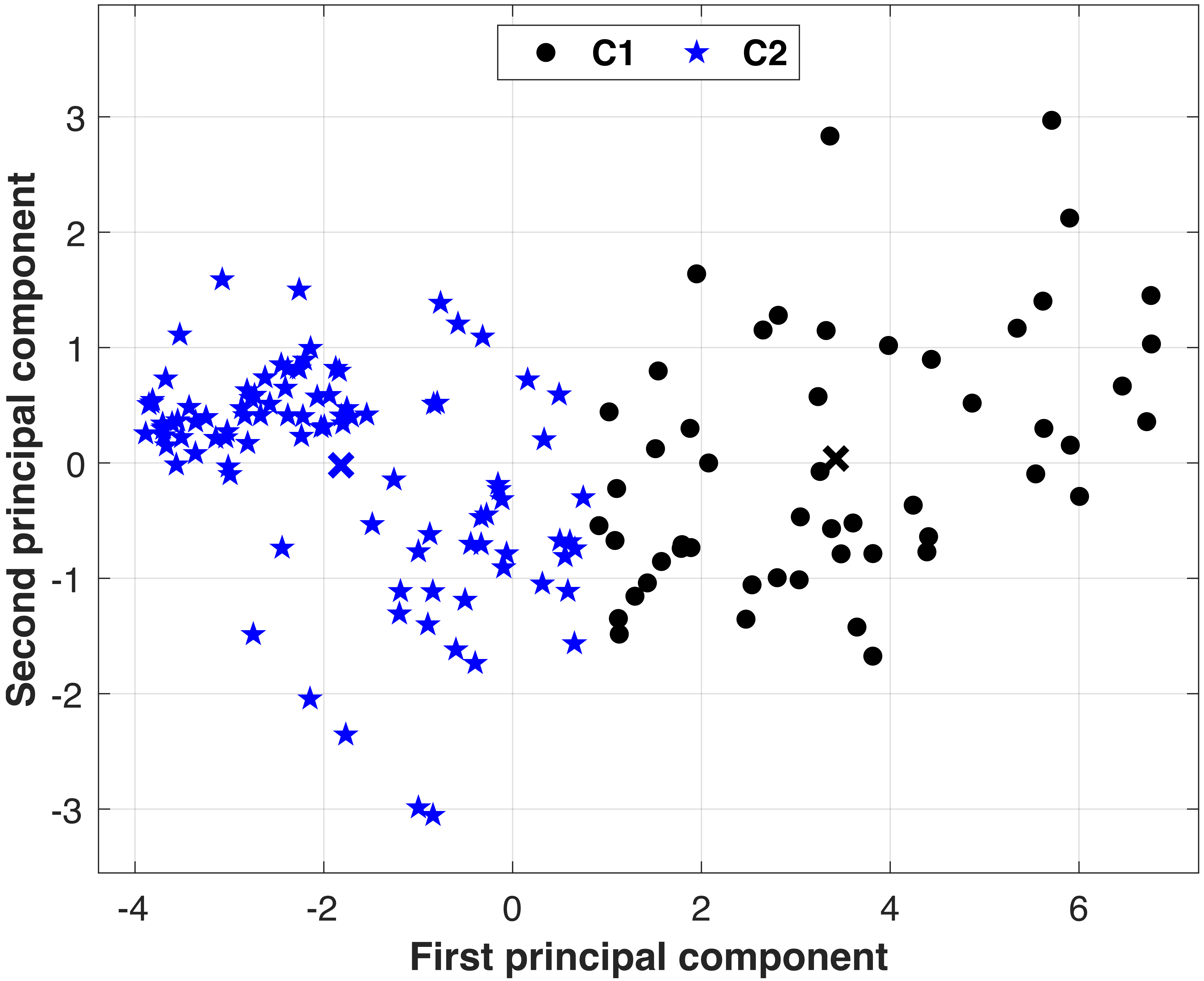
# Clustering — $k$-means, an example

- Back to our web search activity data set
- $150$ web search queries that are used to model flu rates in England
- Weekly frequency for $674$ weeks, i.e. $\mathbf{X} \in \mathbb{R}_{\geq 0}^{674 \times 150}$
- Caveat / warning: To visualise the $k$-means clusters in a 2-dimensional space, I am using the two principal components (PCA; PCA is explained in nature.com/articles/nmeth.4346 and many textbooks and online references) of $\mathbf{X}$; not great in this example because they explain ~ $70\,\%$ of the data's variance
- So, actually clustering applied on a matrix $\mathbf{Z} \in \mathbb{R}^{2 \times 150}$

Data: dropbox.com/s/rgyg190whw26qrj/data-COMP0084-intro-to-ml.zip?dl=0

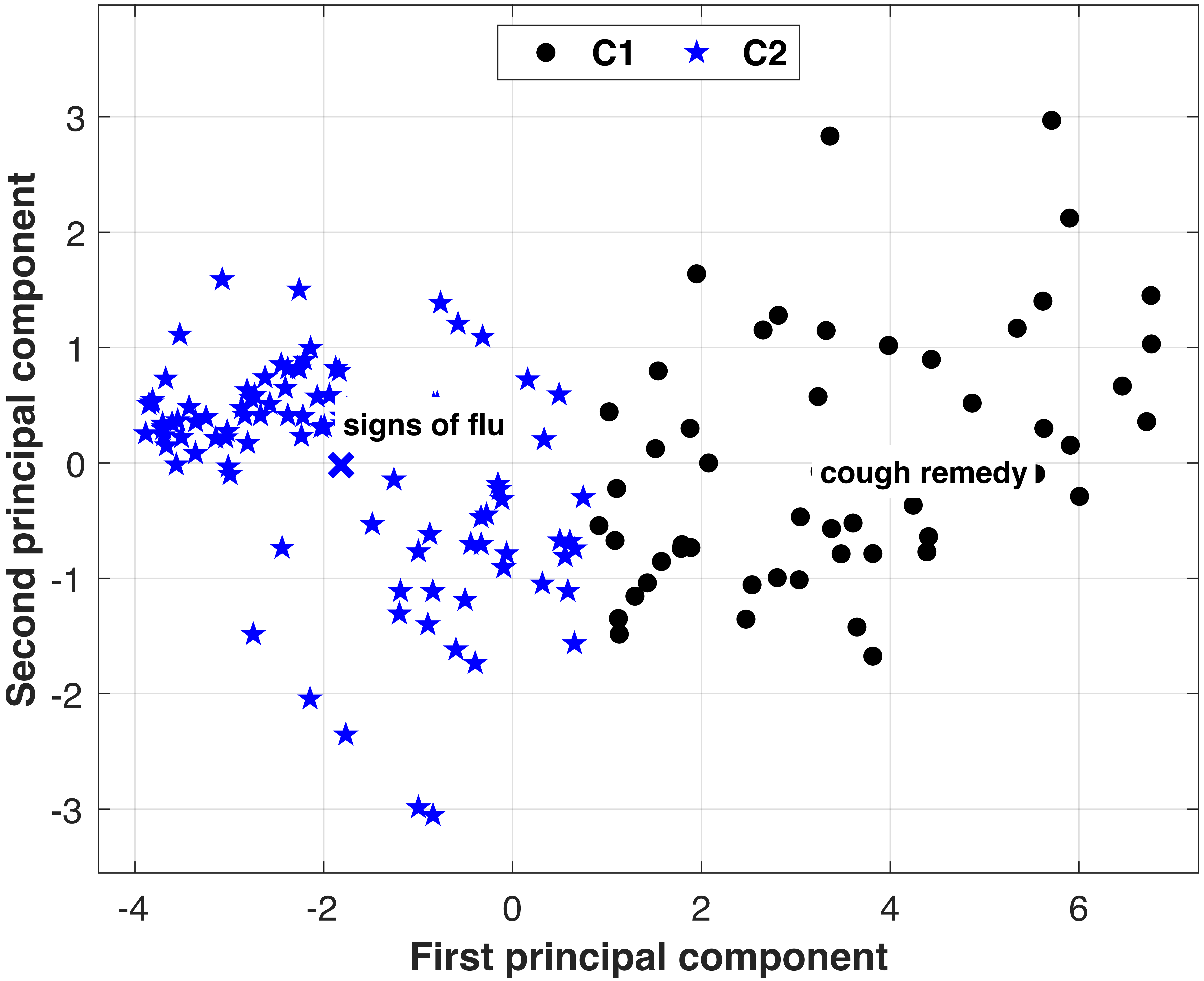
# Clustering — $k$-means, an example

- $k = 2$
- clusters are denoted by C$i$
- a cross is used to denote each cluster's centroid



# Clustering — $k$-means, an example

- $k = 2$
- clusters are denoted by C$i$
- a cross is used to denote each cluster's centroid
- which search queries are closer to their cluster's centroid?

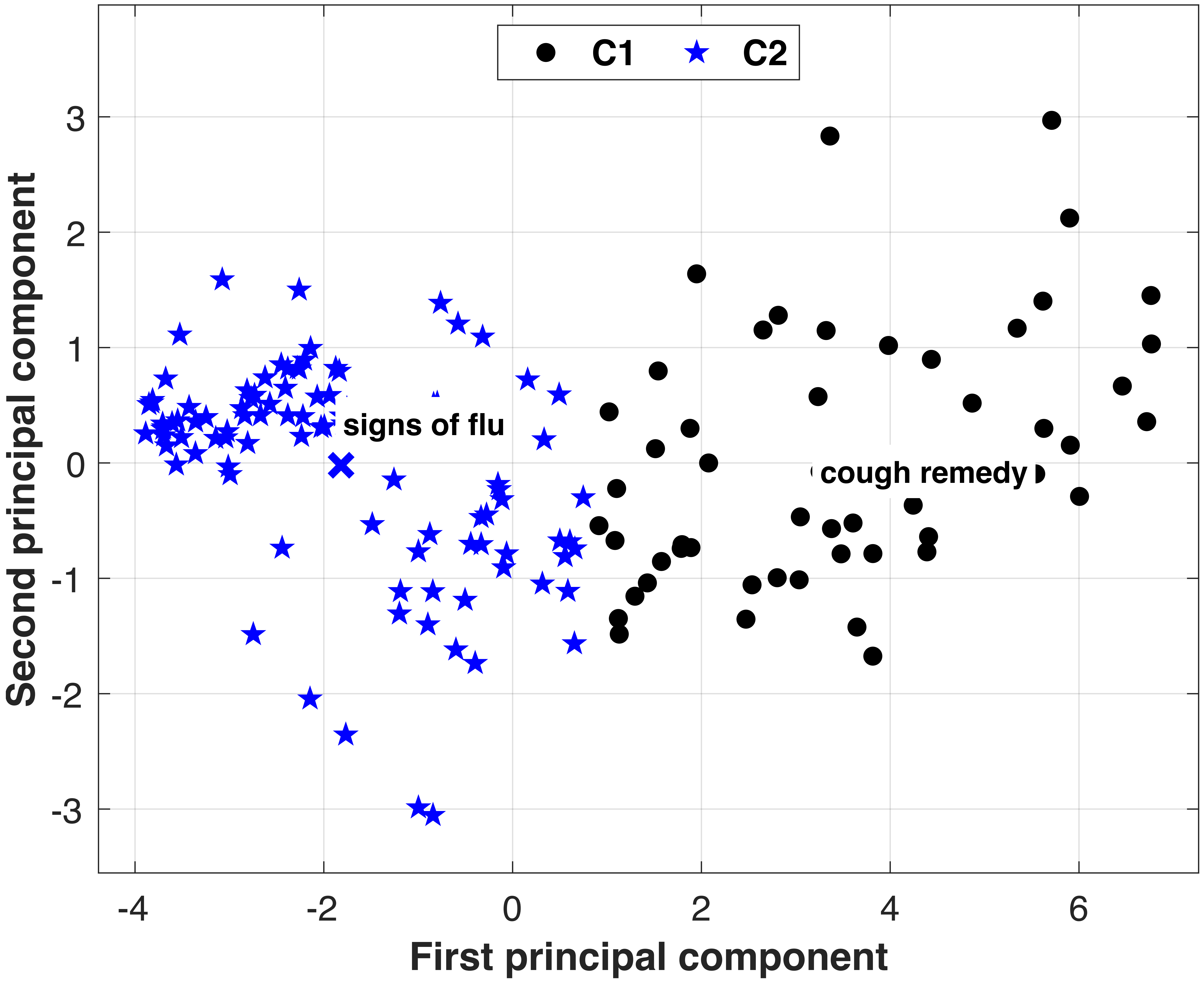


# Clustering — $k$-means, an example

<u>Most central queries</u>

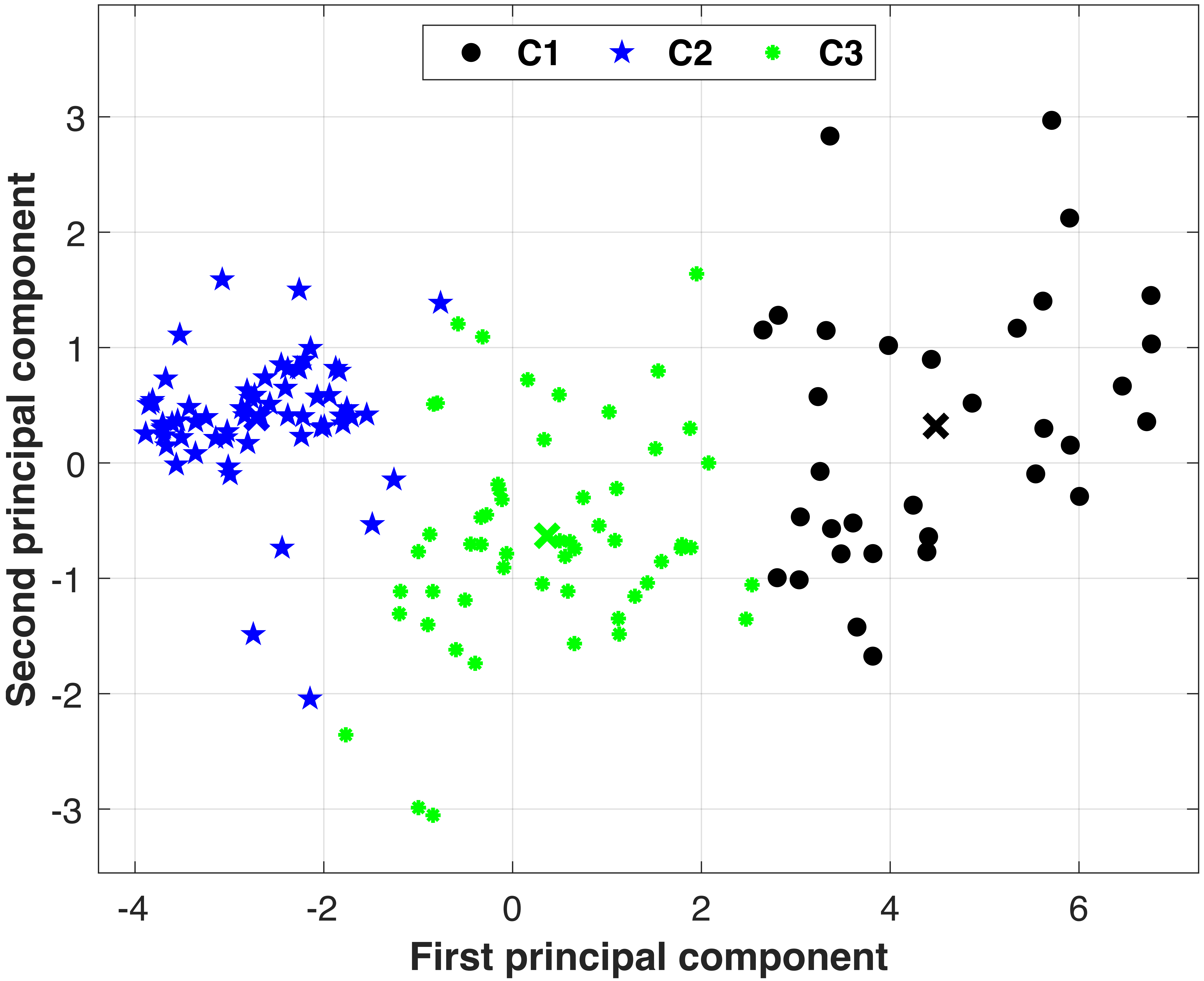
- **C1**:
  "cough remedy"
  "symptoms of bronchitis"
  "lemsip"
  "get rid of a cough"

- **C2**:
  "signs of flu"
  "flu symptoms uk"
  "flu signs"
  "symptom of flu"



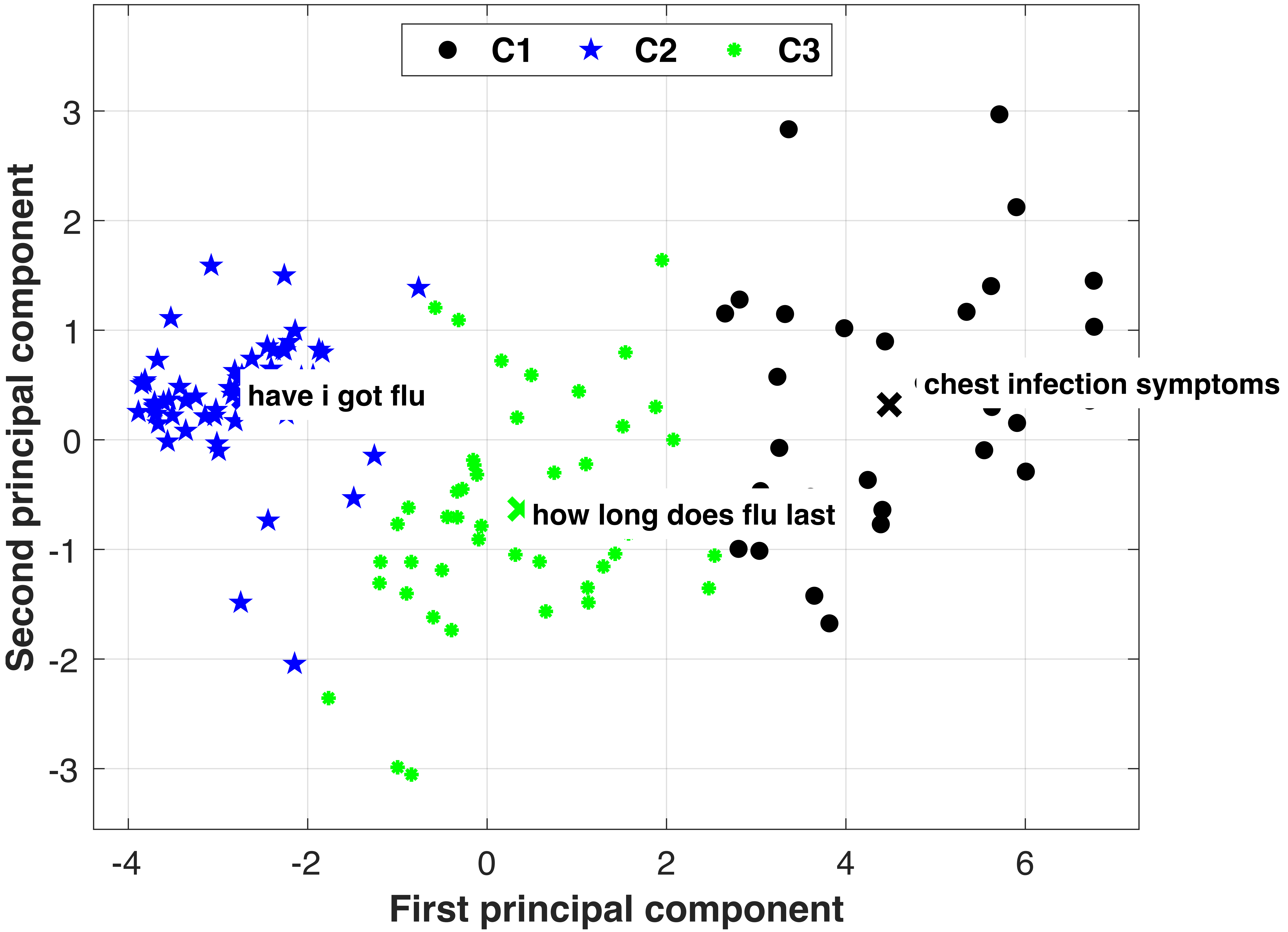
# Clustering — $k$-means, an example

- $k = 3$
- clusters are denoted by C$i$
- a cross is used to denote each cluster's centroid



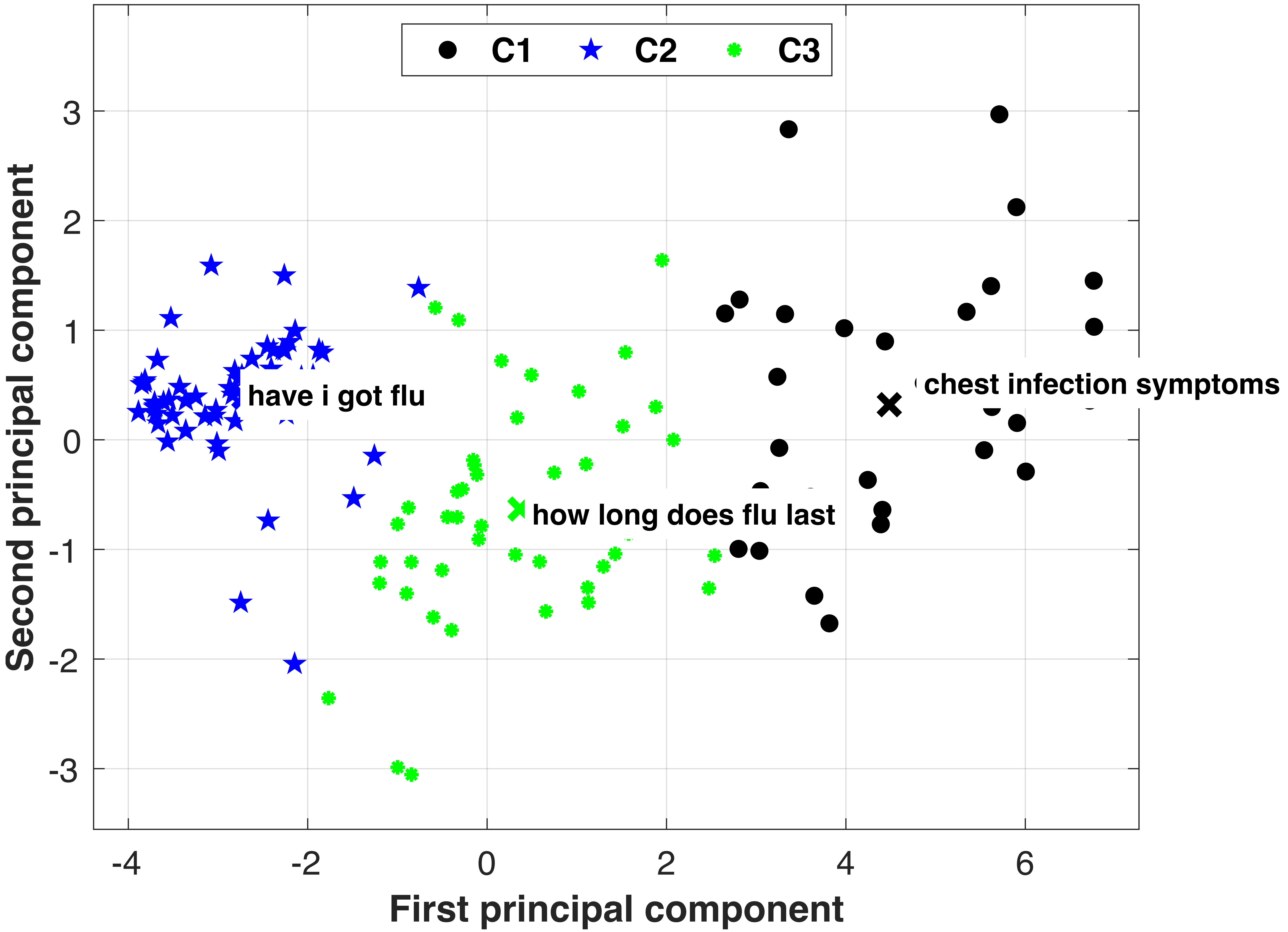
# Clustering — $k$-means, an example

- $k = 3$
- clusters are denoted by C$i$
- a cross is used to denote each cluster's centroid
- which search queries are closer to their cluster's centroid?
- does the addition of a cluster change the thematic coverage of the revised clusters?
- central queries have changed



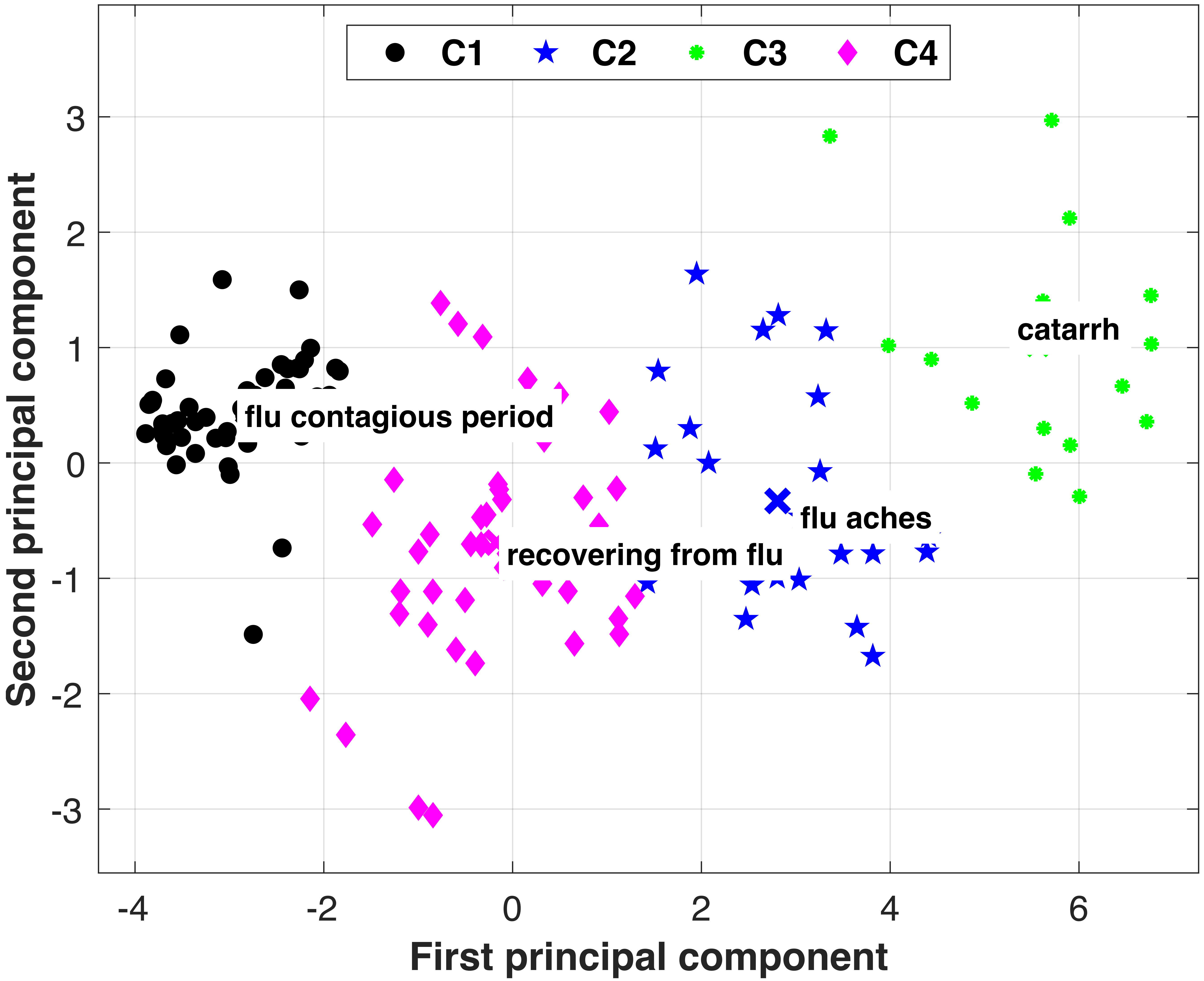
# Clustering — $k$-means, an example

- **C1**: "chest infection symptoms", "coughs", "bronchitis", "cough remedies"
- **C2**: "how long does flu last", "food for flu", "is flu contagious", "how to get rid of the flu"
- **C3**: "have I got flu", "flu contagious period", "flu in babies", "what are the symptoms of flu"



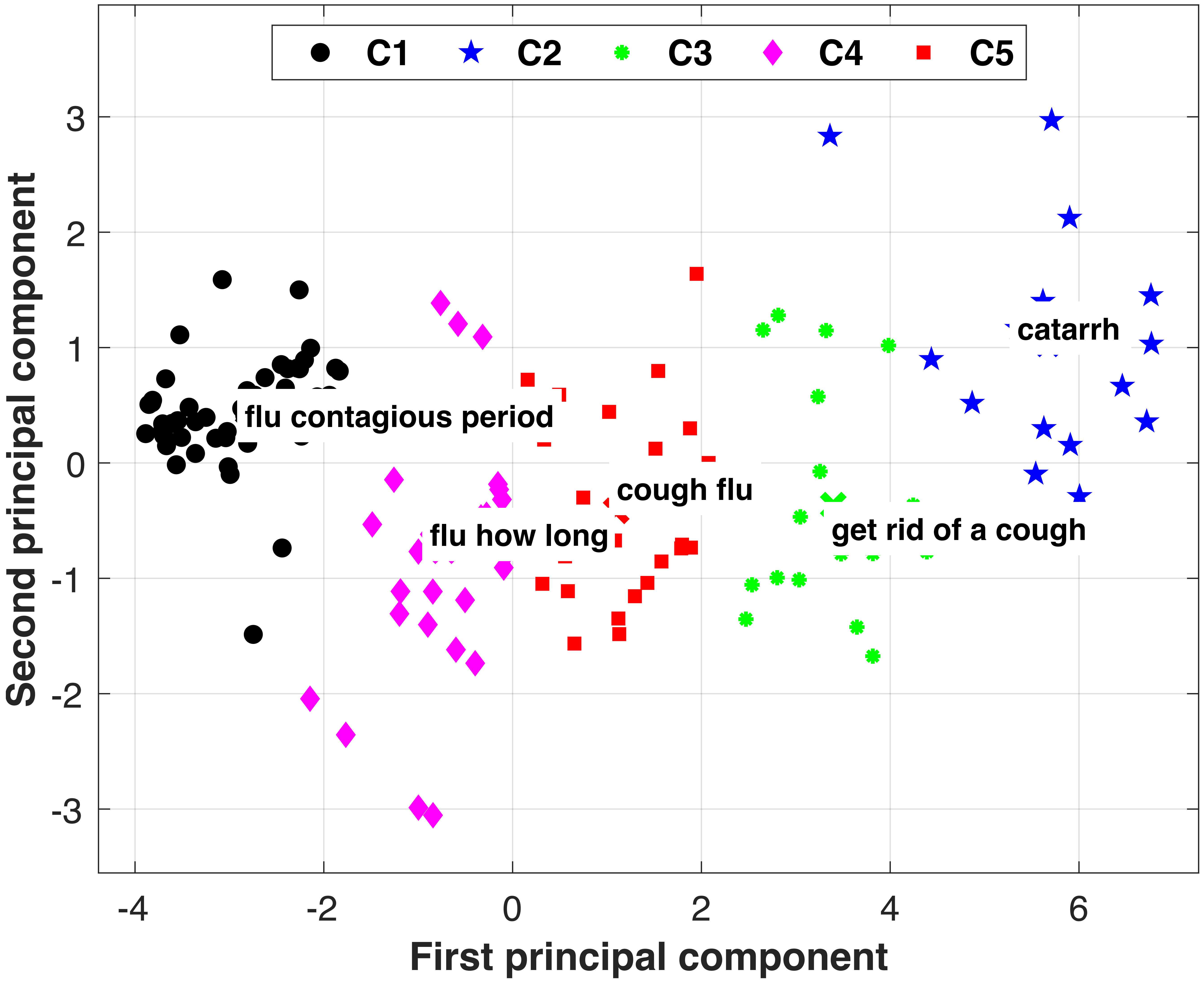
# Clustering — $k$-means, an example

- $k = 4$
- clusters are denoted by C$i$
- a cross is used to denote each cluster's centroid
- which search queries are closer to their cluster's centroid?
- does the addition of a cluster change the thematic coverage of the revised clusters?
- central queries have changed



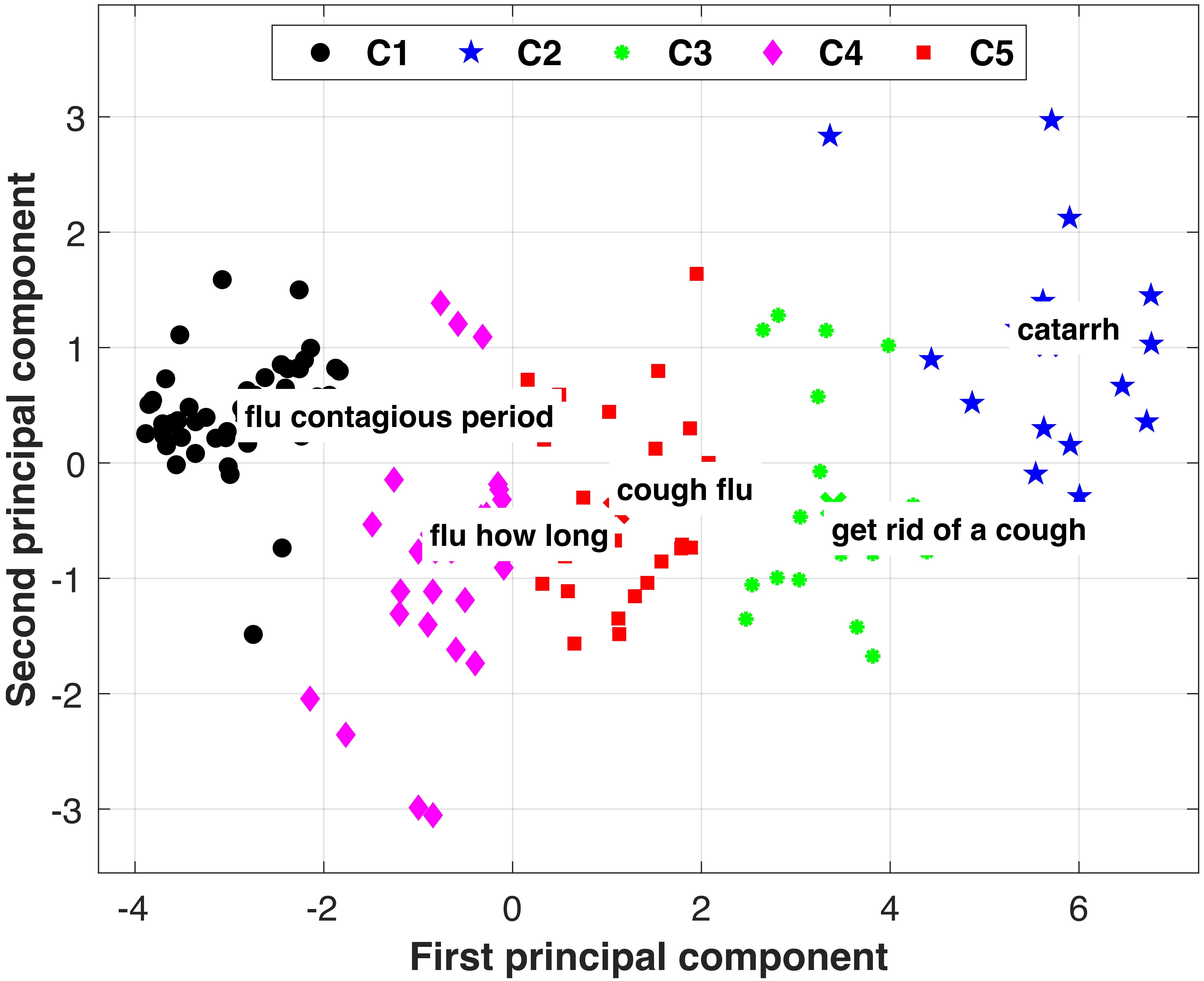
# Clustering — $k$-means, an example

- $k = 5$
- clusters are denoted by C$i$
- a cross is used to denote each cluster's centroid
- which search queries are closer to their cluster's centroid?
- does the addition of a cluster change the thematic coverage of the revised clusters?
- central queries have changed — *partially!*



# Clustering — $k$-means, an example

- **C1**: “flu contagious period”, “flu in babies”, “what are the symptoms of flu”, “have i got flu”
- **C2**: “flu how long”, “how long is flu contagious”, “how long does a flu last”, “how long to recover from flu”
- **C3**: “get rid of a cough”, “lemsip”, “cough remedy”, “flu aches”
- **C4**: “catarrh”, “lurgy”, “pleurisy”, “coughing blood”
- **C5**: “cough flu”, “flu diarrhea”, “difference between cold and flu”, “flu symptoms last”



# Next lectures with me

**Topic models and vector semantics (*word embeddings*)**

- February 28 (2 hours)

**Modelling COVID-19 using web search activity**

- March 20 (1 hour)