

M2BI 2017

A few words about Machine Learning for Bioinformatics

Chloé-Agathe Azencott

Centre for Computational Biology, Mines ParisTech
`chloe-agathe.azencott@mines-paristech.fr`



What is (Machine) Learning?

Why Learn?

- **Learning:**

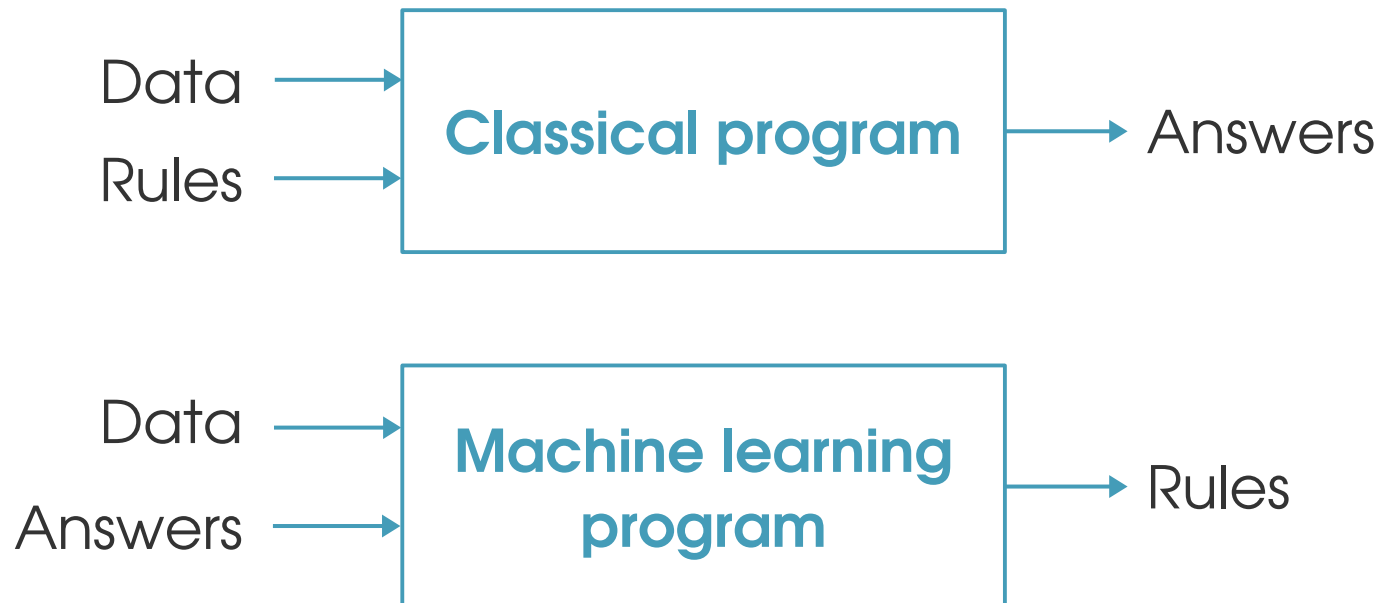
Modifying a behavior based on experience

(F. Benureau)

- **Machine learning:** Programming computers to
 - **model phenomena**
 - by means of **optimizing** an **objective function**
 - using **example** data.

Why Learn?

- There is no need to “learn” to calculate payroll.
- Learning is used when
 - Human expertise does not exist (bioinformatics);
 - Humans are unable to explain their expertise (speech recognition, computer vision).



What about AI?



Artificial Intelligence

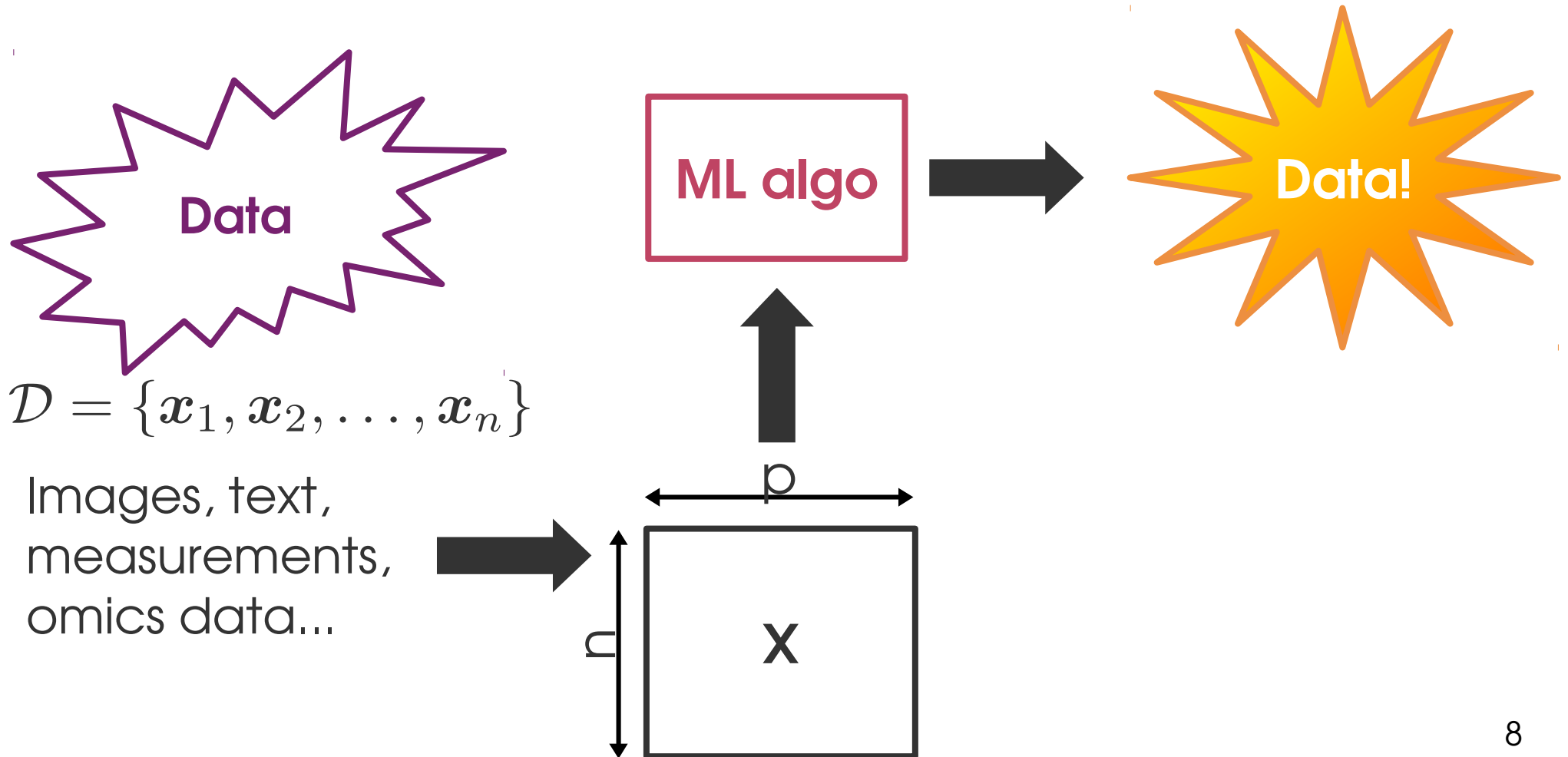
ML is a subfield of **Artificial Intelligence**

- A system that lives in a **changing environment** must have the ability to **learn** in order to **adapt**.
- ML algorithms are building blocks that make computers behave more intelligently by **generalizing** rather than merely storing and retrieving data (like a database system would do).

Unsupervised vs Supervised Machine Learning

Unsupervised learning

Learn a **new representation** of the data



Clustering

Group **similar** data points together



- Understand **general characteristics** of the data;
- **Infer some properties** of an object based on how it relates to other objects.

Clustering: applications

- **Customer segmentation**

Find groups of customers with similar buying behavior.

- **Topic modeling**

Find groups of documents with similar content (and thus, hopefully, topics).

- **Gene expression clustering**

- **Disease subtyping**

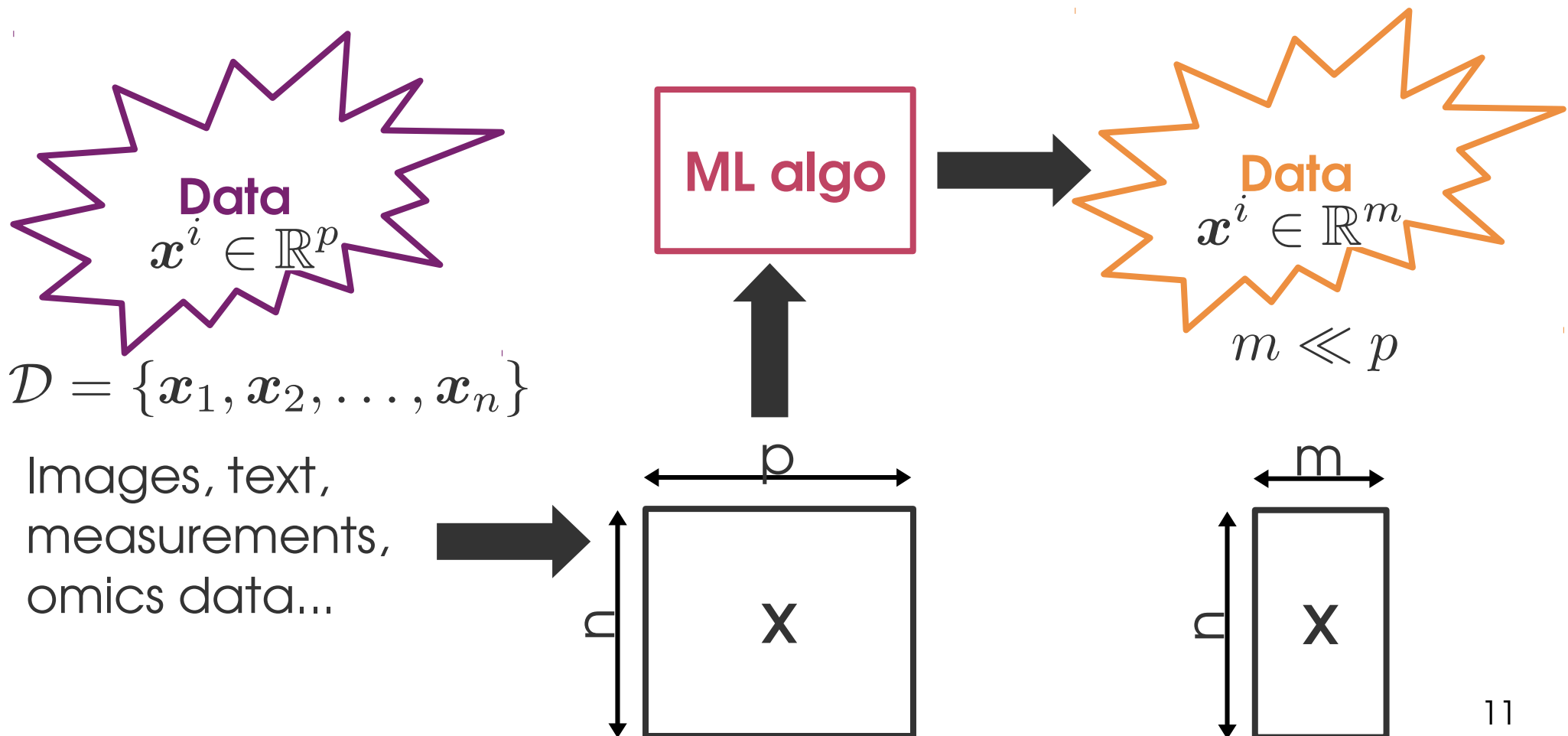
Find groups of patients with similar molecular basis for the disease / similar symptoms.

- **Tumor cell clustering**

Untangle tumor heterogeneity.

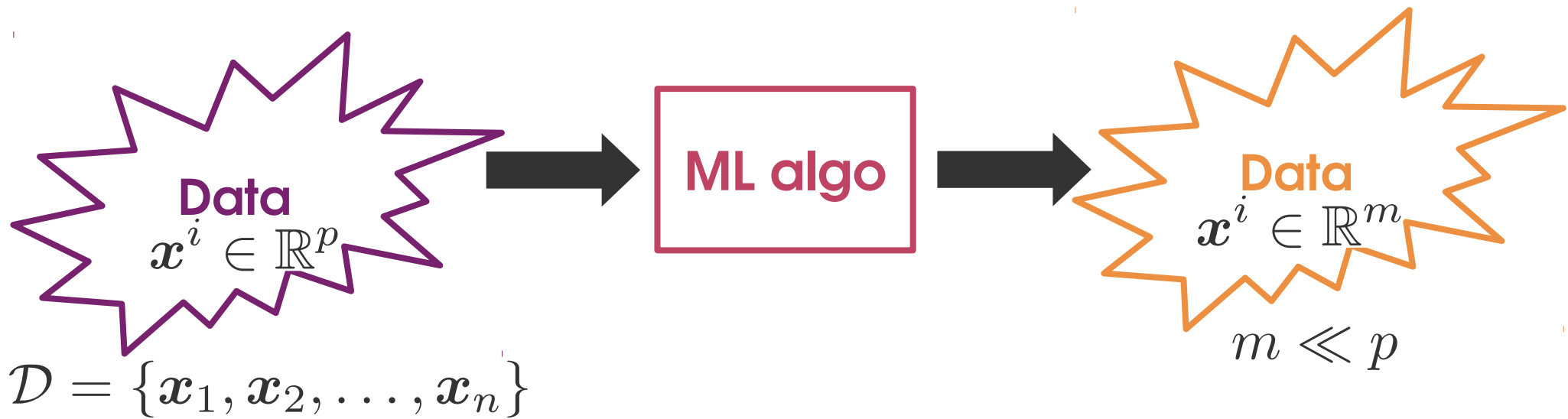
Dimensionality reduction

Find a **lower-dimensional** representation



Dimensionality reduction

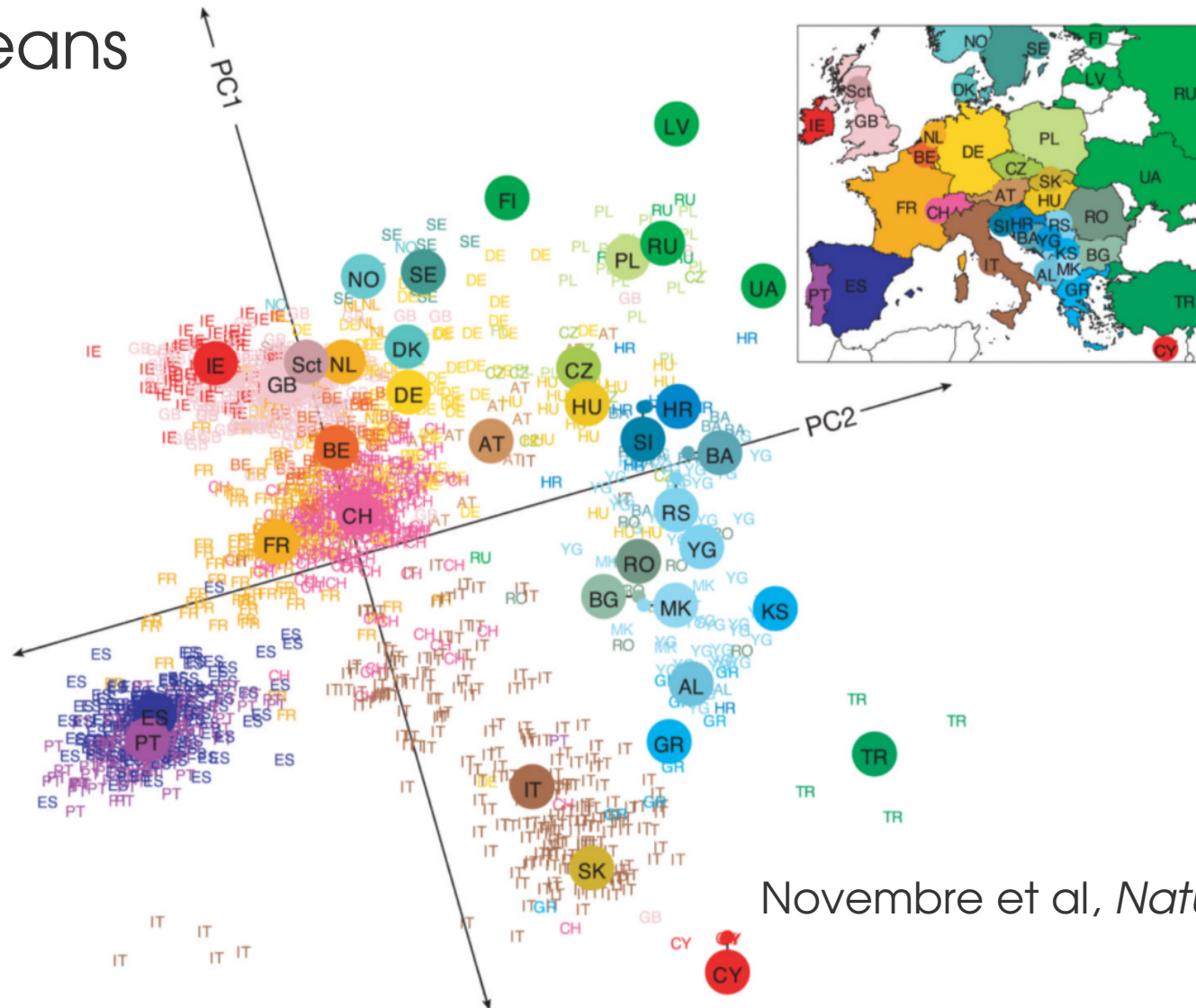
Find a **lower-dimensional** representation



- Reduce storage **space** & computational **time**
- Remove **redundances**
- **Visualization** (in 2 or 3 dimensions) and **interpretability**.

Dim reduction: applications

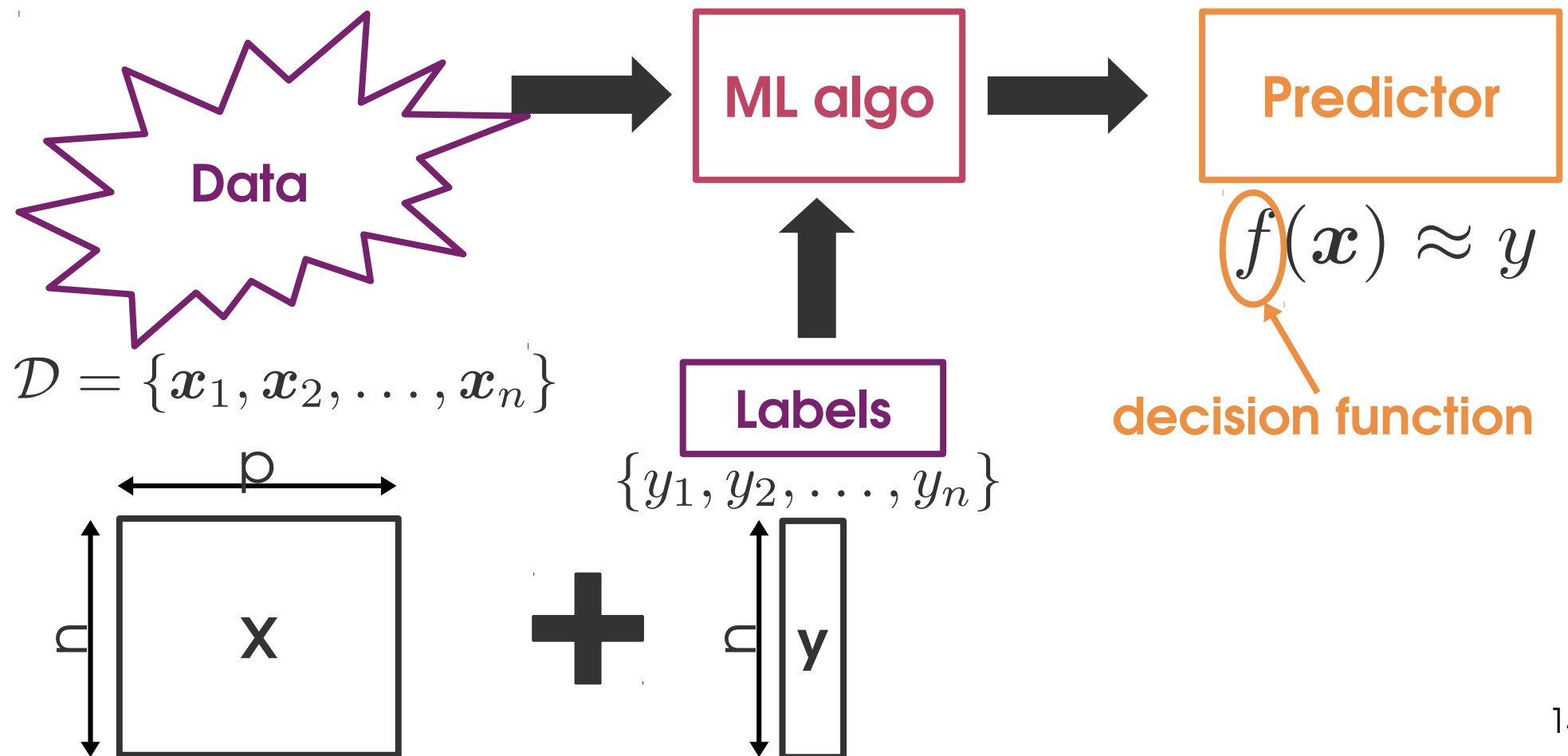
SNP data (Affymetrix 500K) for
1387 Europeans



Novembre et al, *Nature* 2008

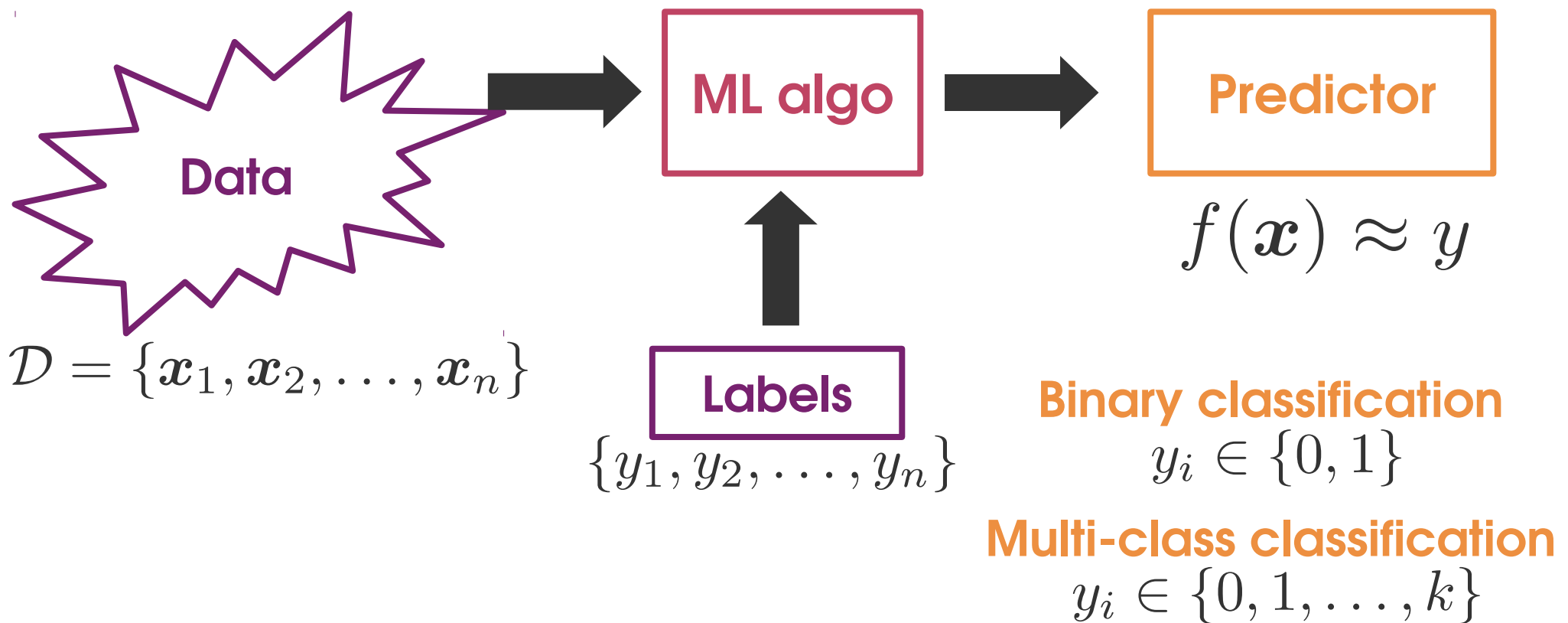
Supervised learning

Make predictions

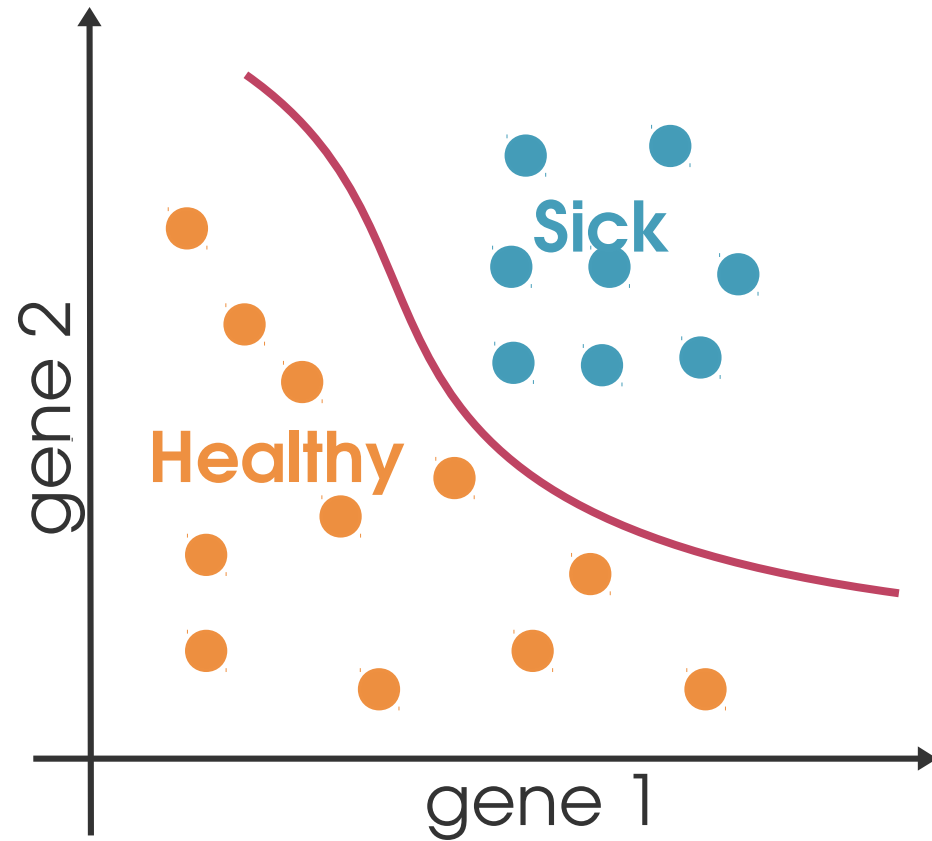


Classification

Make **discrete** predictions



Classification

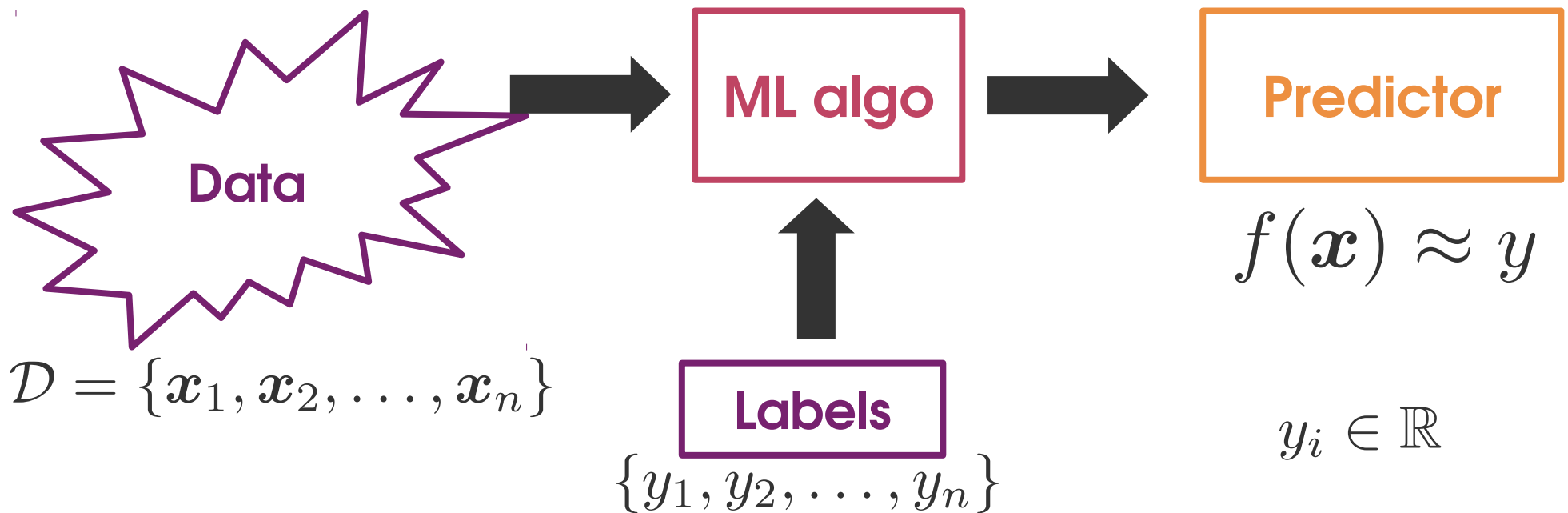


Classification: Applications

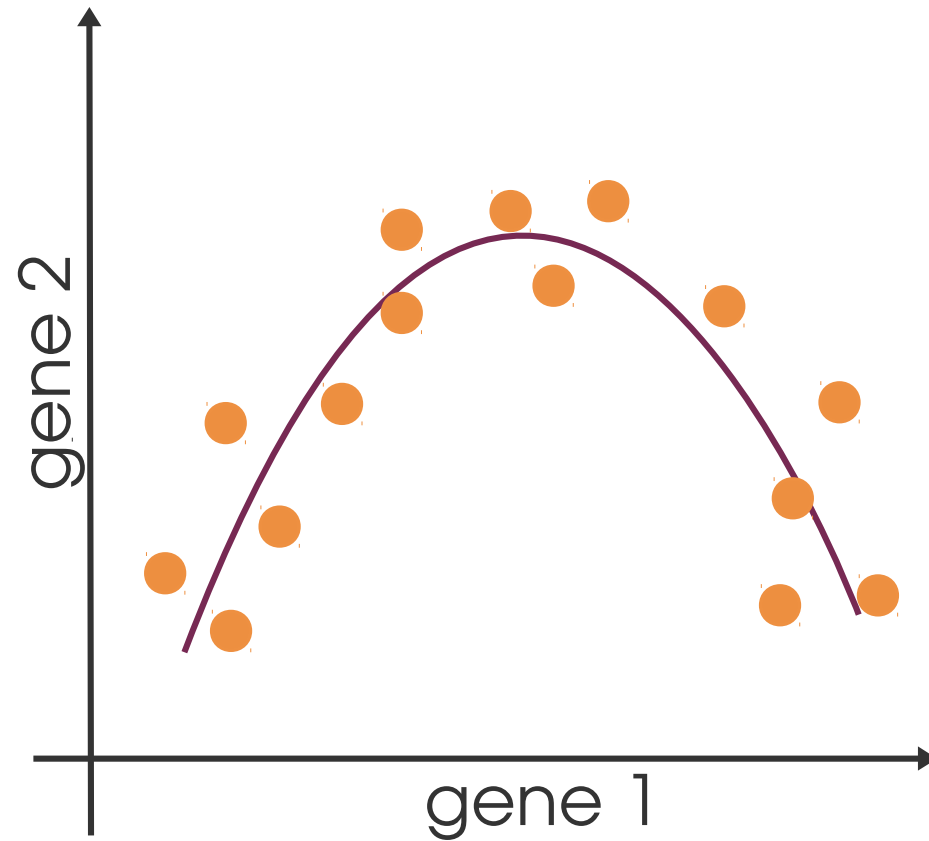
- Face recognition
- Handwritten character recognition
- What is the **function of this gene**?
- Is this **DNA sequence** a **micro-RNA**?
- Does this blood sample come from a **diseased or a healthy individual**?
- Is this **drug** appropriate for this patient?
- What **side effects** could this new drug have?

Regression

Make **continuous** predictions



Regression



Regression: Applications

- **Click prediction**

How many people will click on this ad? Comment on this post?
Share this article on social media?

- **Load prediction**

How many users will my service have at a given time?

- **Gene expression regulation:** how much of this gene will be expressed?
- When will this **patient relapse?**
- **Drug efficacy:** how well does this drug work on this tumor?
- What is the **binding affinity** between these two molecules?
- How **soluble** is this chemical in water?

Generalization & overfitting

Generalization

- **Goal:** build models that make good predictions on **new data**.
- Models that work “too well” on the data we learn on tend to model noise as well as the underlying phenomenon: **overfitting**.

Overfitting (Classification)

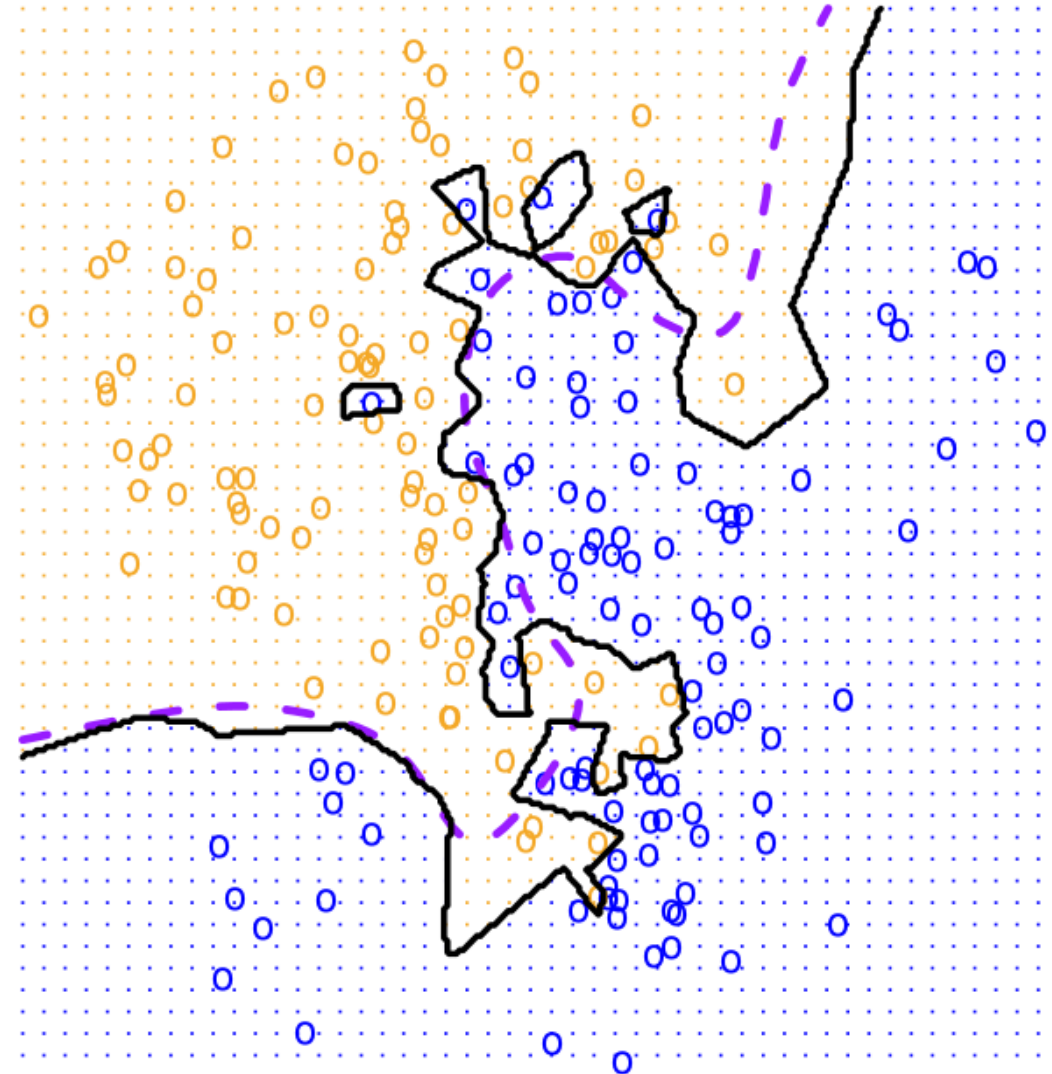


Figure from Hastie, Tibshirani & Friedman, *The Elements of Statistical Learning*.

Overfitting (Regression)

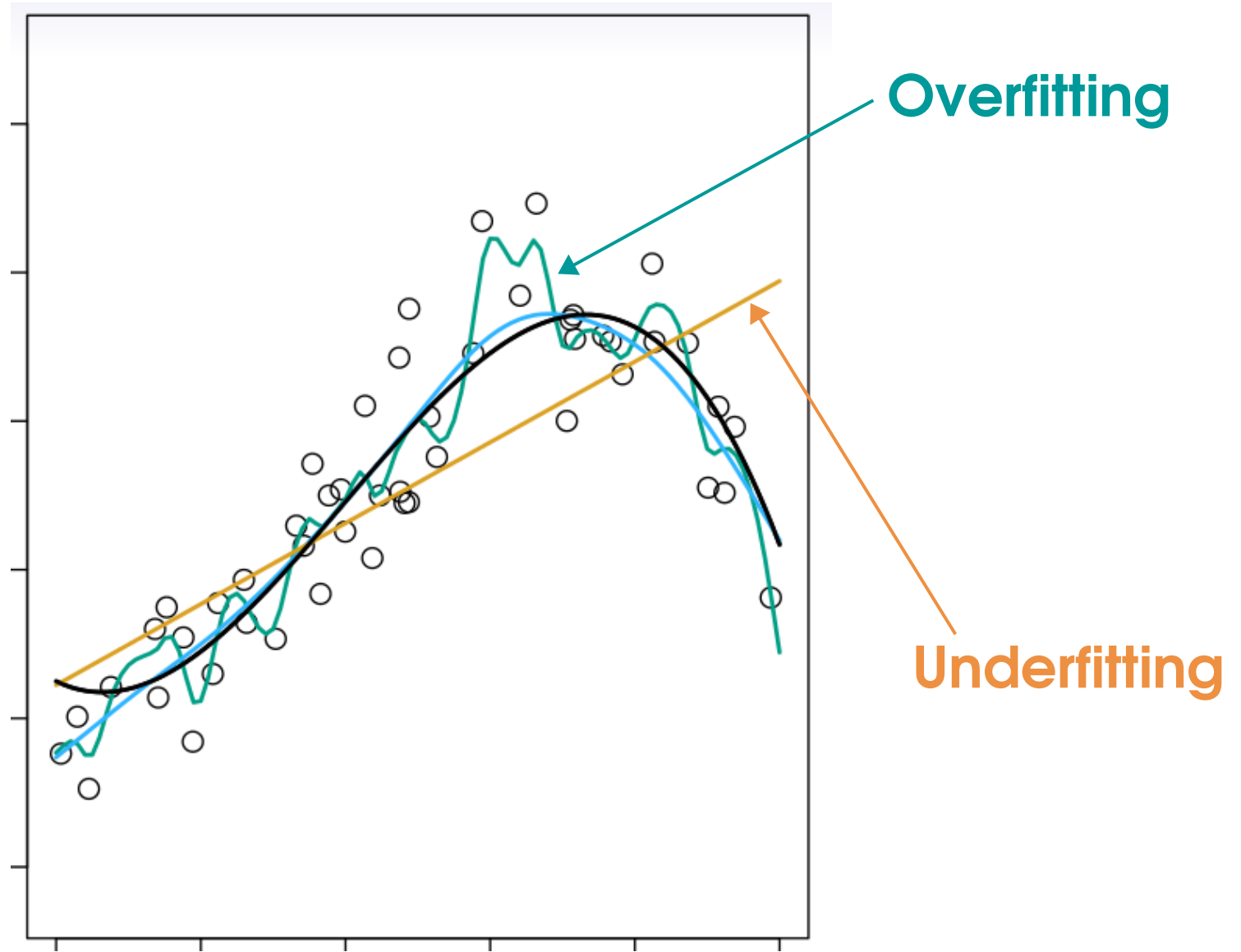
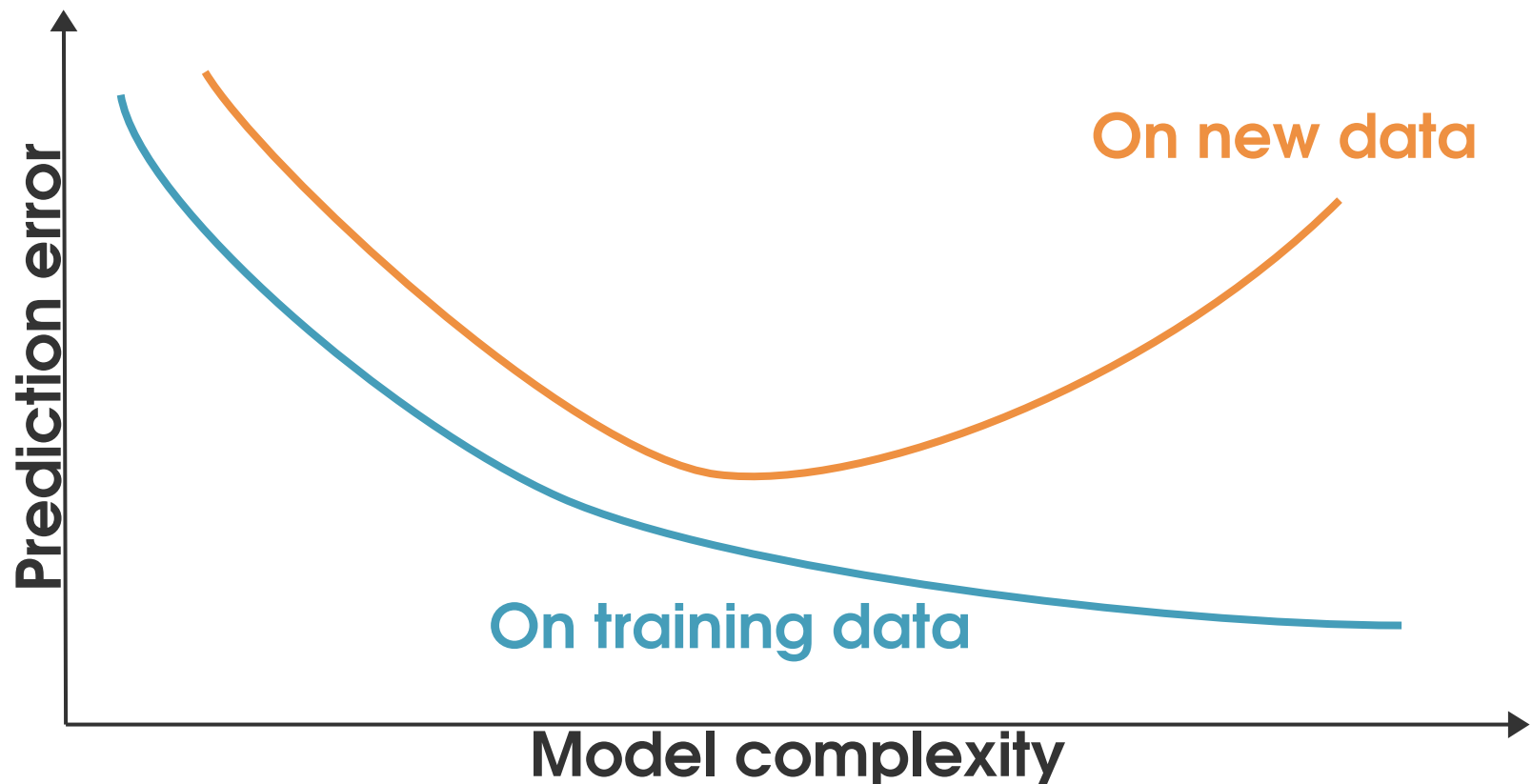


Figure from Hastie, Tibshirani & Friedman, *The Elements of Statistical Learning*.

Model complexity

Simple models are

- more **plausible** (Occam's Razor)
- easier to **train, use, and interpret.**



Regularization

- Prevent overfitting by designing an **objective function** that accounts not only for the **prediction error** but also for **model complexity**.

$$\min (\text{empirical_error} + \lambda * \text{model_complexity})$$

Ridge regression

$$\hat{\beta}_{\text{ridge}} = \arg \min_{\beta} \boxed{\|y - X\beta\|_2^2} + \lambda \boxed{\|\beta\|_2^2}$$

prediction error regularizer

└─ hyperparameter

- **Unique solution, always exists**

$$\hat{\beta}_{\text{ridge}} = (X^{\top} X + \lambda I)^{-1} X^{\top} y$$

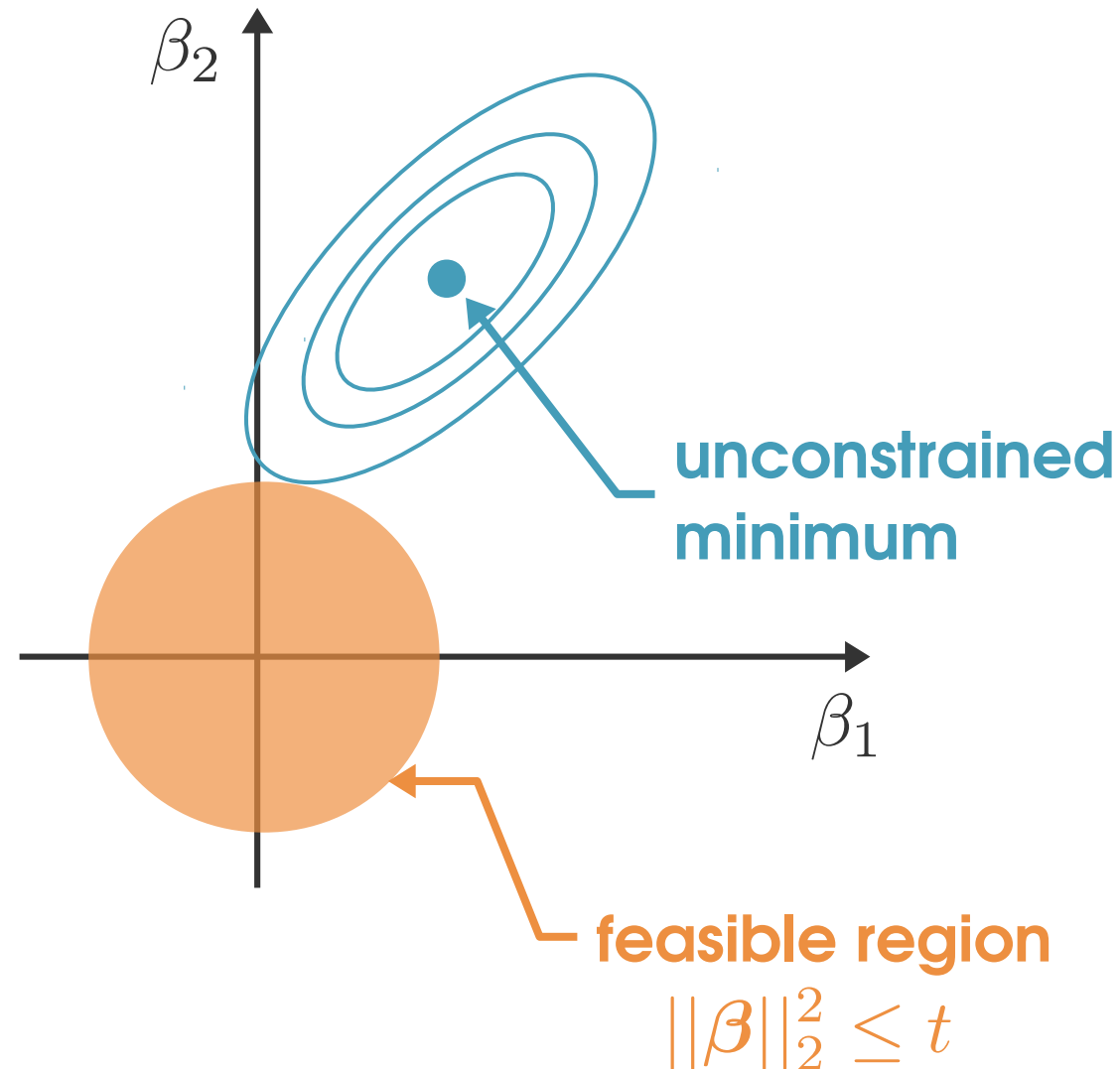
- **Grouped selection:**

Correlated variables get similar weights.

- **Shrinkage**

Coefficients shrink towards 0.

Geometry of ridge regression



Model selection & evaluation

No free lunch theorem

- Any two optimization algorithms are equivalent when their performance is averaged over all possible problems.
- For any learner, **there is a learning task that it will not learn well.**

Wolpert & Macready, 1997

Evaluation on held-out data

- If we evaluate the model on the data we've used to train it, we risk **over-estimating performance**.
- **Proper procedure:**
 - Separate the data in **train/test** sets

Evaluation on held-out data

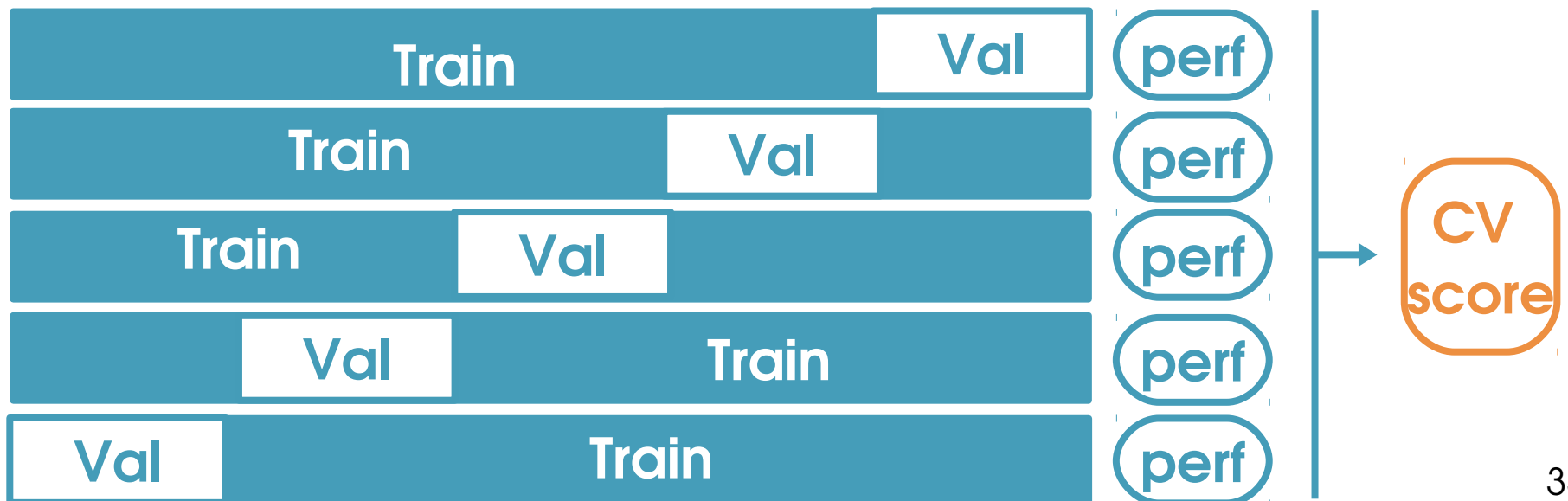


- **Proper procedure:**
 - Separate the data in **train/test** sets

Evaluation on held-out data



- Proper procedure:
 - Separate the data in **train/test** sets
 - Use a **cross-validation** on the train set to find the best algorithm + hyperparameter(s)



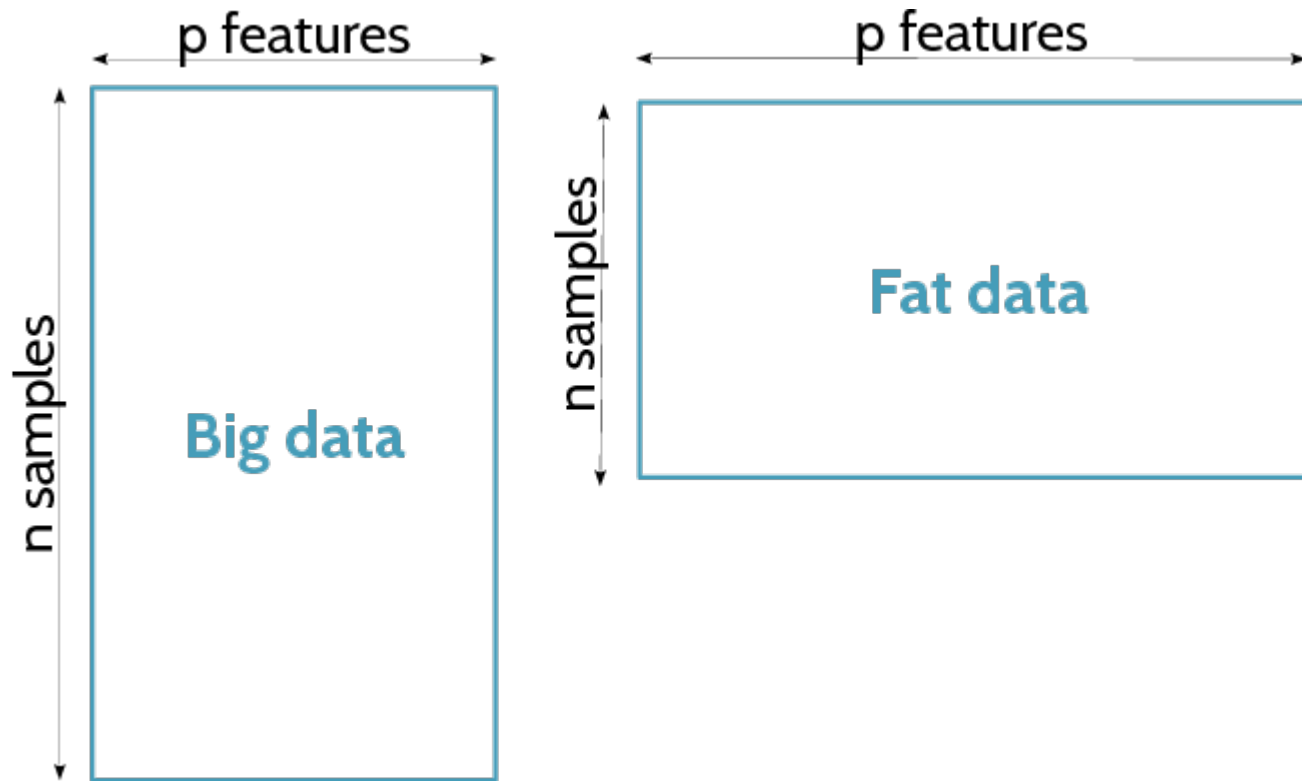
Evaluation on held-out data



- **Proper procedure:**
 - Separate the data in **train/test** sets
 - Use a **cross-validation** on the train set to find the best algorithm + hyperparameter(s)
 - Train this best algorithm + hyperparameter(s) on the entire train set
 - The performance on the test set estimates **generalization performance**.

ML in high dimension

“Big data” vs “fat data”



Challenges when $p \gg n$:

- **Computational** challenges (e.g. linear regression)
- **Curse of dimensionality**
- Higher risk of **overfitting**.

Feature extraction

- Project the data onto a lower-dimensional space
- Creates **new features** → **interpretability?**
- **Matrix factorization** techniques:
PCA & kPCA, factorial analysis, non-negative matrix factorization.
- **Manifold learning** techniques
MDS, tSNE
- **Autoencoders:** neural networks
Goal: recover the input as well as possible.

Supervised feature selection

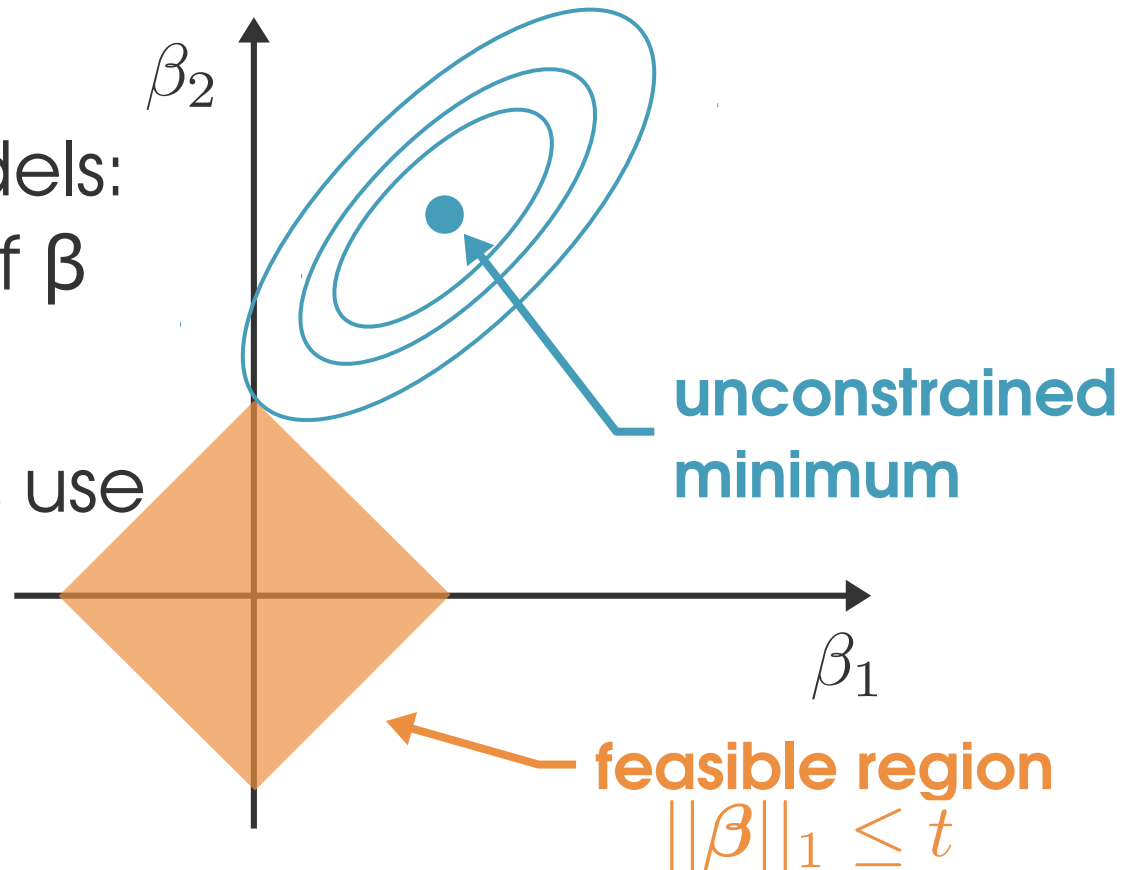
Keep relevant features only

- **Filter approaches:** apply a statistical test to assign a score to each feature
- **Wrapper approaches:** use a greedy search to find the best set of features for a given ML algorithm
- **Embedded approaches:** fit a **sparse** model, i.e. that is encouraged to *not* use all the features.

Lasso

$$\hat{\beta}_{\text{lasso}} = \arg \min_{\beta} \underbrace{\|y - X\beta\|_2^2}_{\text{prediction error}} + \lambda \underbrace{\|\beta\|_1}_{\text{regularizer}}$$

- Create **sparse** models: drive coefficients of β towards 0.
- No explicit solution, use **gradient descent**.



ML Toolboxes

- **Python: scikit-learn**

<http://scikit-learn.org>



- **R: Machine Learning Task View**

<http://cran.r-project.org/web/views/MachineLearning.html>

- **Matlab™: Machine Learning with MATLAB**

<http://mathworks.com/machine-learning/index.html>

- Statistics and Machine Learning Toolbox
- Deep Learning Toolbox

- **Deep learning software**

- **Theano** <http://deeplearning.net/software/theano/>
- **TensorFlow** <http://www.tensorflow.org/>
- **Caffe** <http://caffe.berkeleyvision.org/>
- **Keras** <https://keras.io/>