Postgraduate course
Universitat de Valencia 2020

Introduction to Machine Learning for physicists

Veronica Sanz (UV/IFIC)

LECTURE 2
REGRESSION

Learning from data

To understand Nature, we **observe** Nature ⟶ dataset $\mathbf{X}$

Then, we **model** Nature ⟶ model $g(\mathbf{w})$

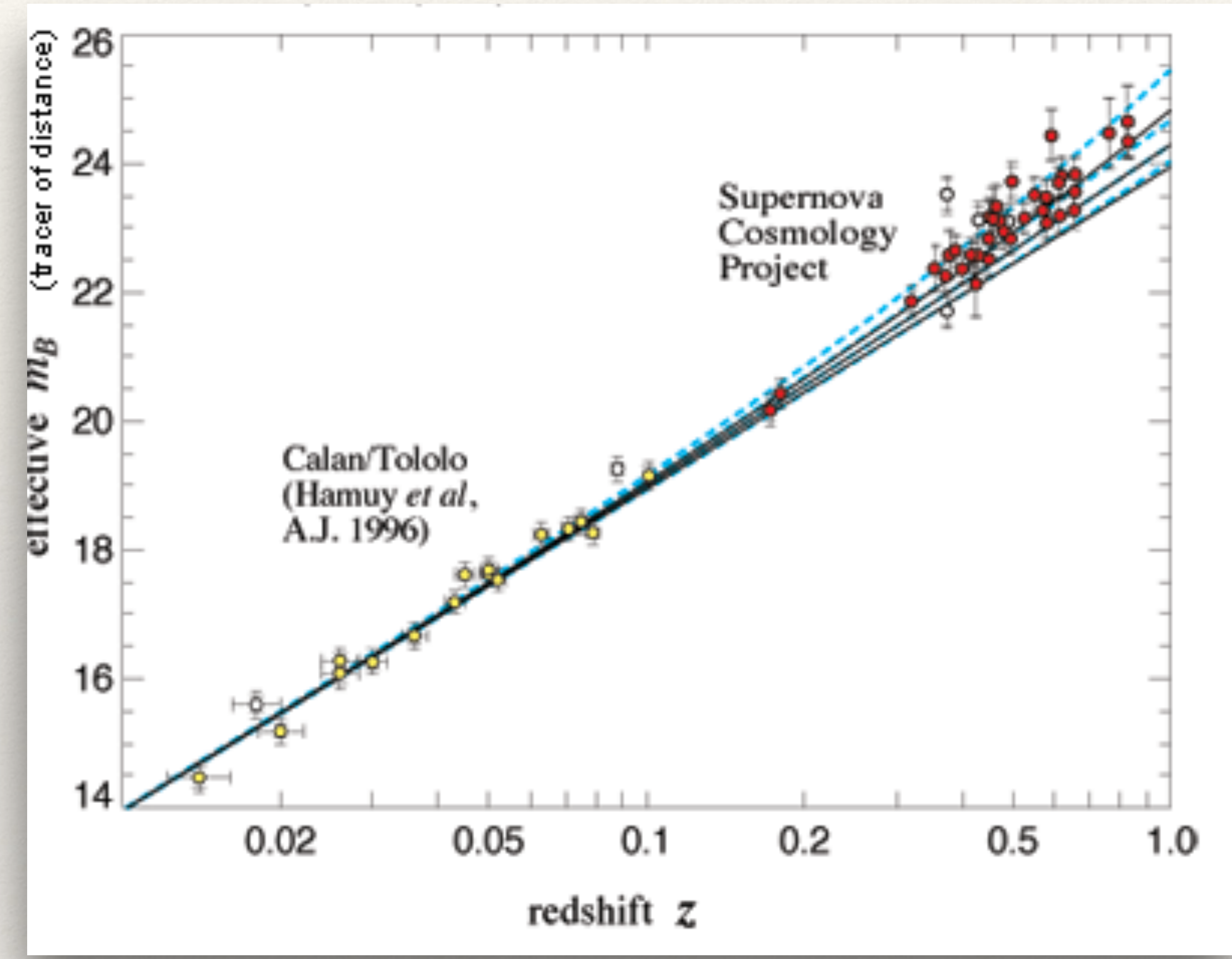
g: type of model, **w**: parameters of the model

↓

Contrast model vs Nature using a metric

something that allows us to answer the question:
how well $g(\mathbf{w})$ represents $\mathbf{X}$?
best representation fixes $\mathbf{w}$

Learning from data

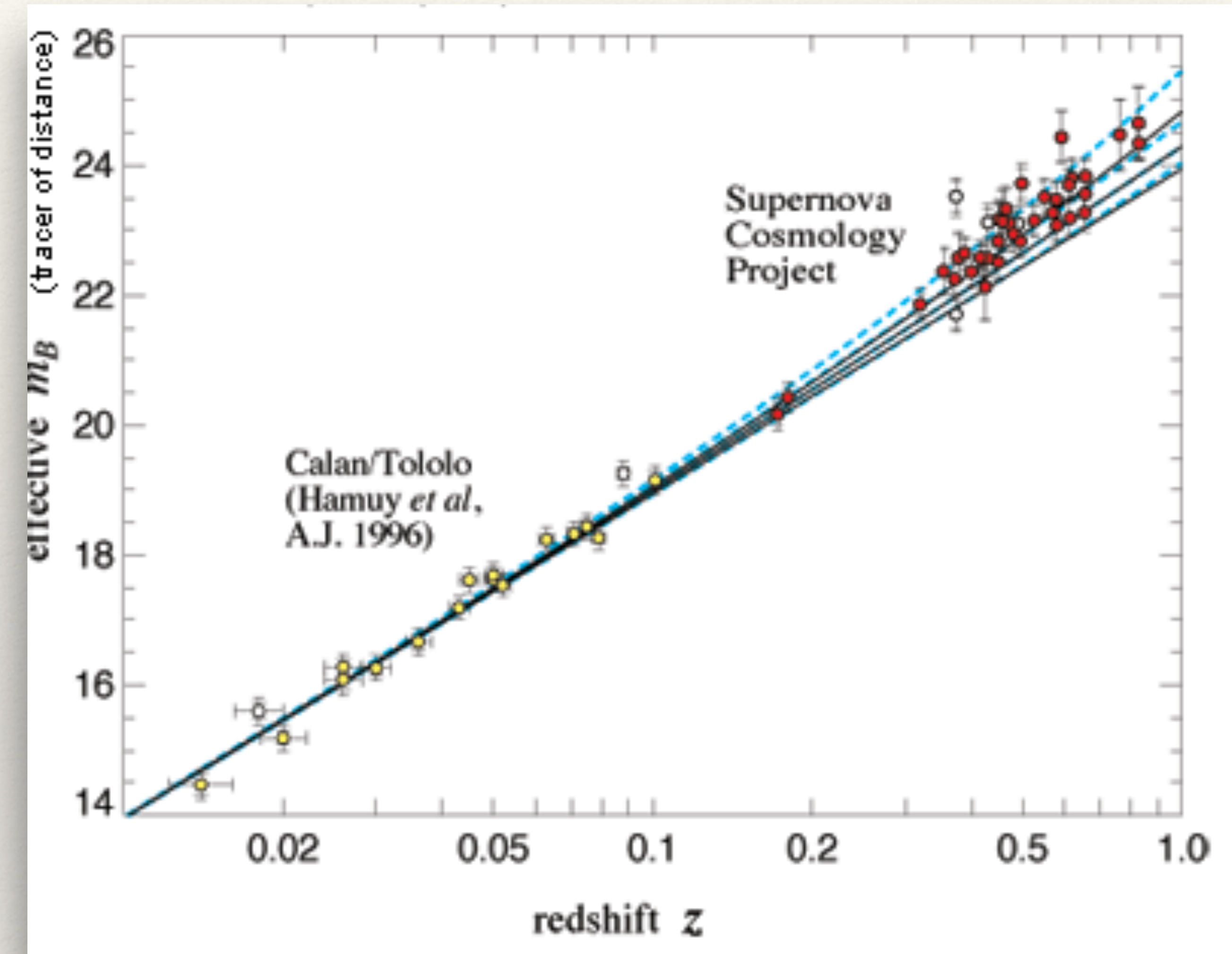


g = linear fit
w= *best* parameters

BEST respect to what?
minimises some type of error
distance between obs-theory

in general the criteria to learn the values of **w** is called the
COST FUNCTION(**X**, g(**w**))
so that min COST —> values of **w** —> fixes best model

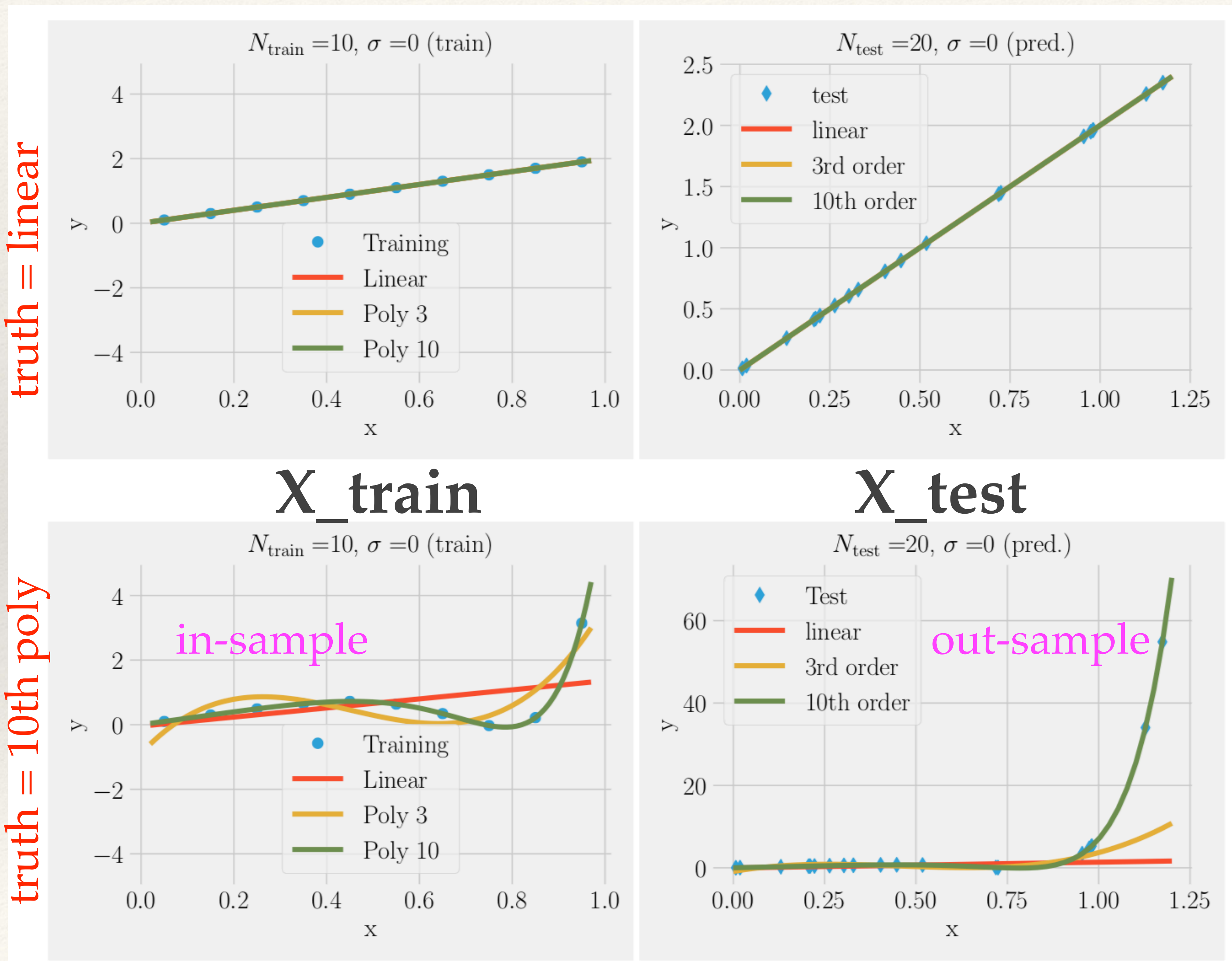
Physical understanding



To arrive to this plot
z: redshif, effective mB (~ distance)
clearly linear
LOTS of things had to be understood, and highly non-trivial transformations to the data had to be done

and no matter how smart scientists would be at choosing the right quantities, we still needed **enough data**, and **clean enough**
and finally, our model needs to **GENERALIZE**

Fitting and predicting



use X_train to find best model parameters

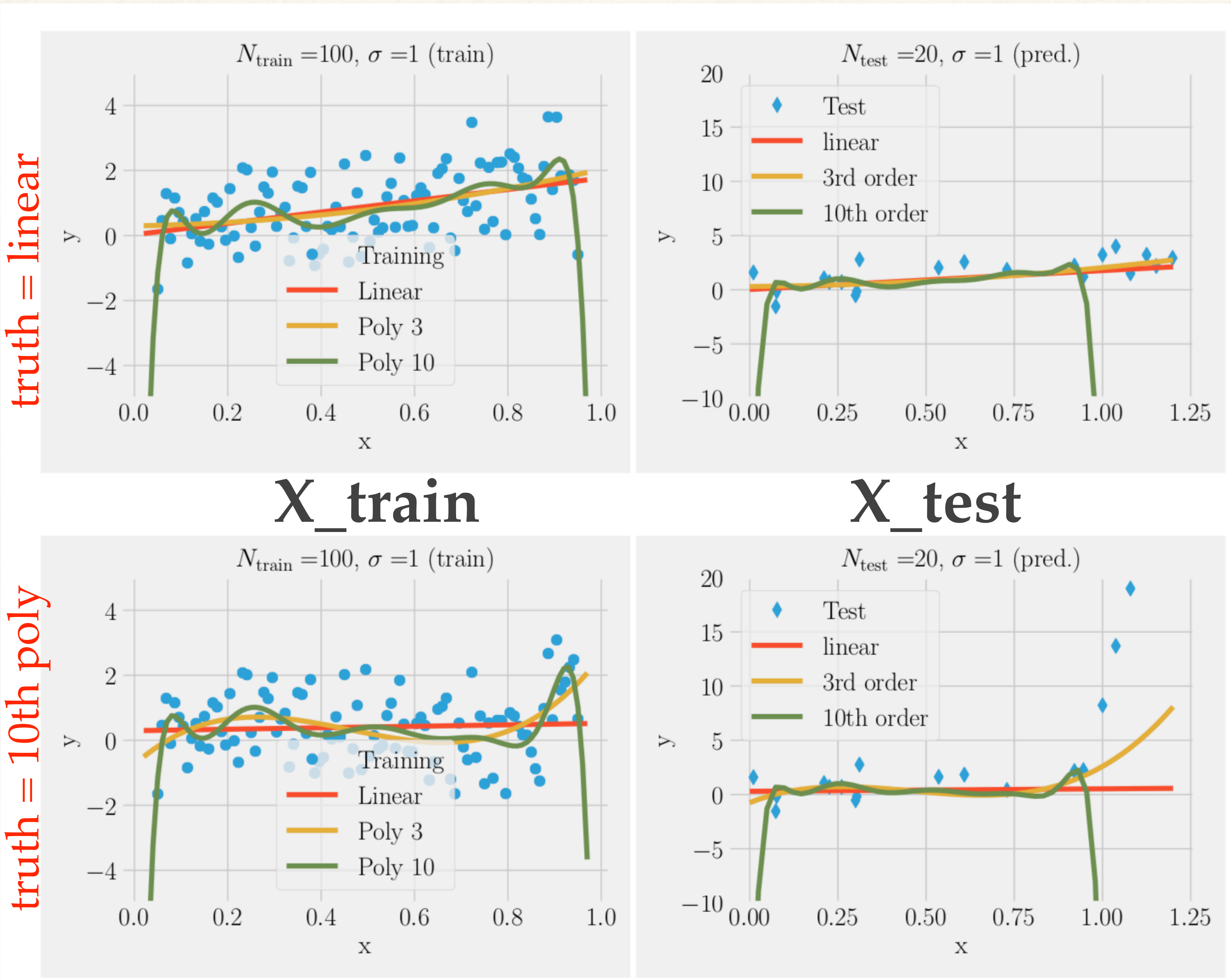
test on X_test how my predictions generalise to new / unseen data

Clean, no-noise data even with a small dataset generalises well

linear & poly-10 (truth) are the best fits to data and the best models -> generalise well

plots from this excellent review

Fitting and predicting



Let's add noise,
gaussian errors

Now
linear and poly-3 models
are better models,
generalise better to X_test

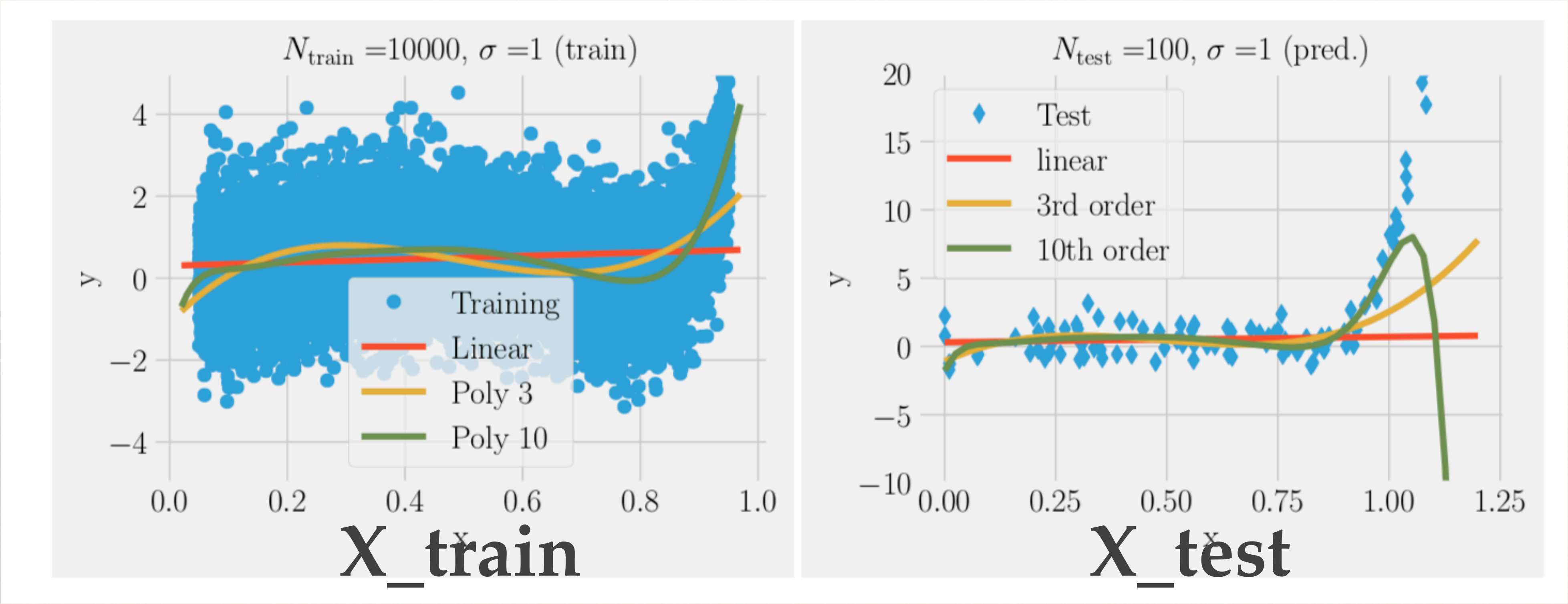
poly-10 is *too expressive*
(has the ability to fit many
features)
and with noisy data
OVERFITTING
Fitting and predicting are
not the same

plots from this excellent review

Fitting and predicting

BUT with enough data, poly-10 gets a chance to use its expressivity
poly-10 gives the best out-sample prediction

truth = 10th poly



With few datapoints, an expressive model can be tricked into explaining patterns in noisy data which aren't real (overfitting)
With enough data, this problem improves

This is very important for ML, where typical models, e.g. Neural Networks, are extremely expressive (much, much more than a poly-10

Fitting and predicting

Summing up:

1. Fitting is not predicting
what we care about is predicting new situations
a robust model generalises well
2. Using a complex model can lead to overfitting
rookie's No. 1 problem
3. Simple models are better on small and complex datasets

Now let's look at how we fit the data

Learning from data

Data sample

(**in**-sample)

$\mathcal{D}(x_i, y_i)$

$i = 1 \dots n$

x_i inputs

y_i outputs

Understand / Learn

relations in/out

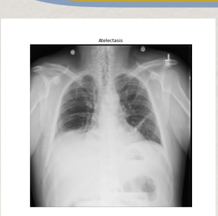
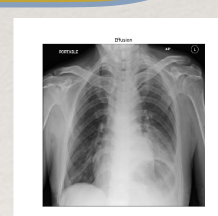
predict in->out

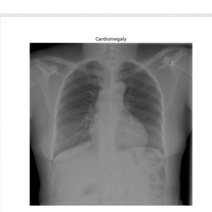
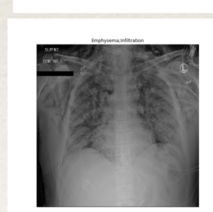
(**out**-sample)

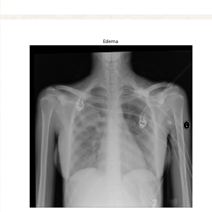
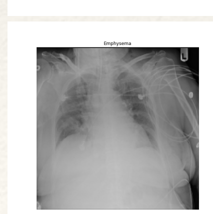
Learning from data

Data sample
(**in**-sample)
$\mathcal{D}(x_i, y_i)$
$i = 1 \dots n$
x_i inputs
y_i outputs

Understand / Learn
relations in/out
predict in->out
(**out**-sample)

( , diag1) ( , diag4)

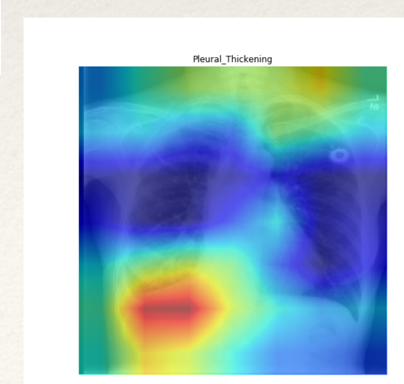
( , diag2) ( , diag5)

( , diag3) ( ,diag6)

. . .

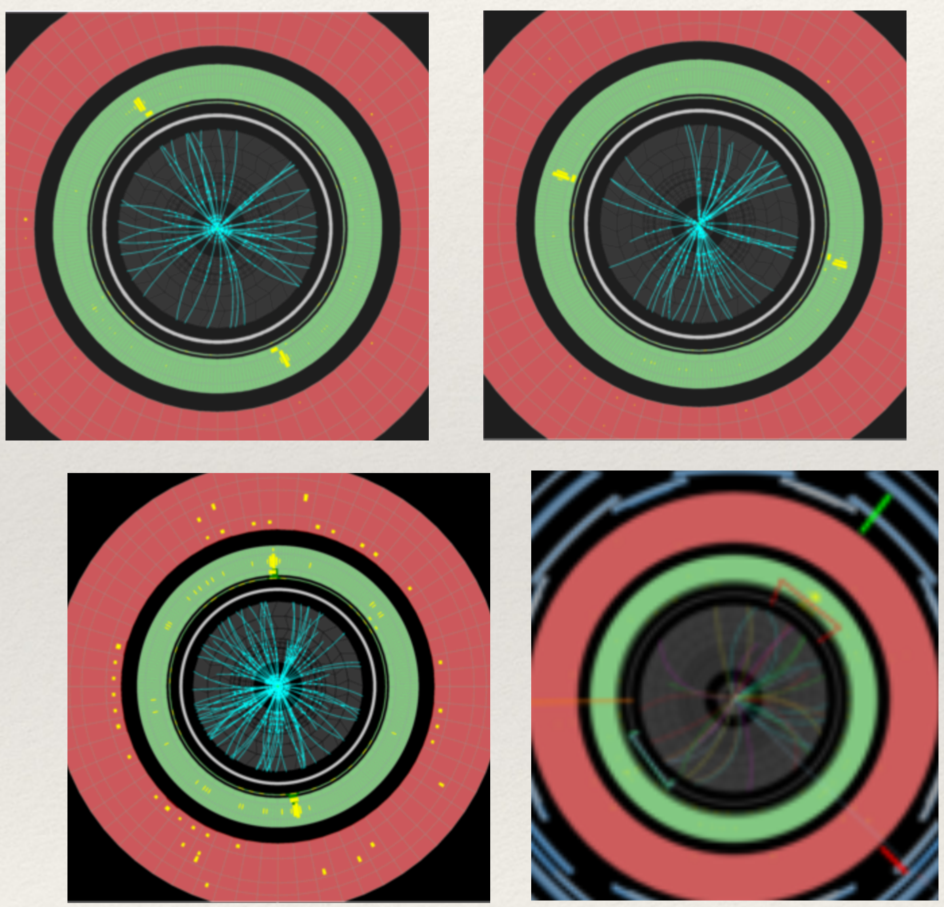
diagnosis?

features?



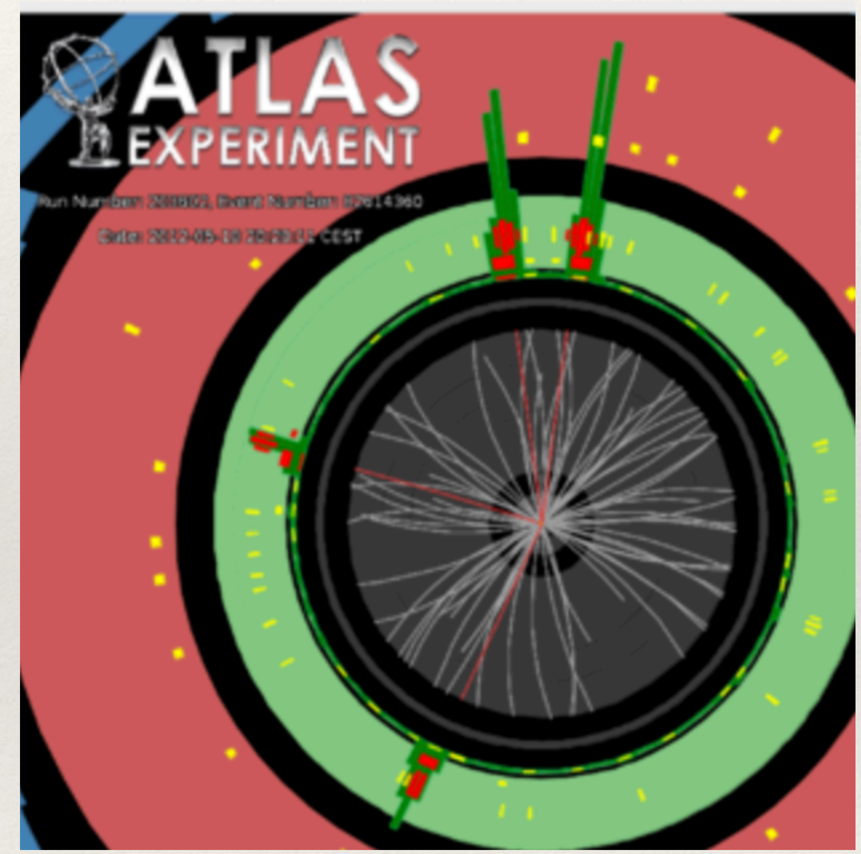
Learning from data

FIT



(event, label)

PREDICT



event -> label

Learning from data

Learning depends on

DATA: amount, quality (*stats&syst*)

VARIANCE

MODEL COMPLEXITY: how we interpret the data

all inputs and outputs?

assumptions $y(x)$ *e.g.* $y(x) = \sum_p a_p x^p$

BIAS

COMPUTATIONAL LIMITATIONS:

limited resources

architectural bias (von Neumann)

Cost Function

Criteria we use to decide whether we are learning from data

minimise difference between **(what we observe - what we predicted)**

COST FUNCTION = some functional dependence on this difference

e.g. SQUARED ERROR over a dataset $\mathcal{D}(x_i, y_i)$

minimize $\mathcal{C}(\mathcal{D}) = \sum_{i=1}^{n} (y_i - \hat{y}(x_i))^2$

to obtain $y(x) = f(w.x + b)$

weight bias

if f(**w**.**x**+b) is simply **w**.**x**+b LINEAR REGRESSION

best model is then $\hat{\mathbf{w}} = \arg\min |\mathbf{X}.\mathbf{w} - \mathbf{y}|^{\mathbf{2}}$

This is *just* a minimisation problem…

The best model (w) is the one that satisfies

$$\mathbf{0} = \boldsymbol{\nabla}\mathcal{C}(\mathbf{w}) = \sum_{i=1}^{n} \left[f(\mathbf{x}_i^T \mathbf{w}) - y_i \right] \mathbf{x}_i,$$

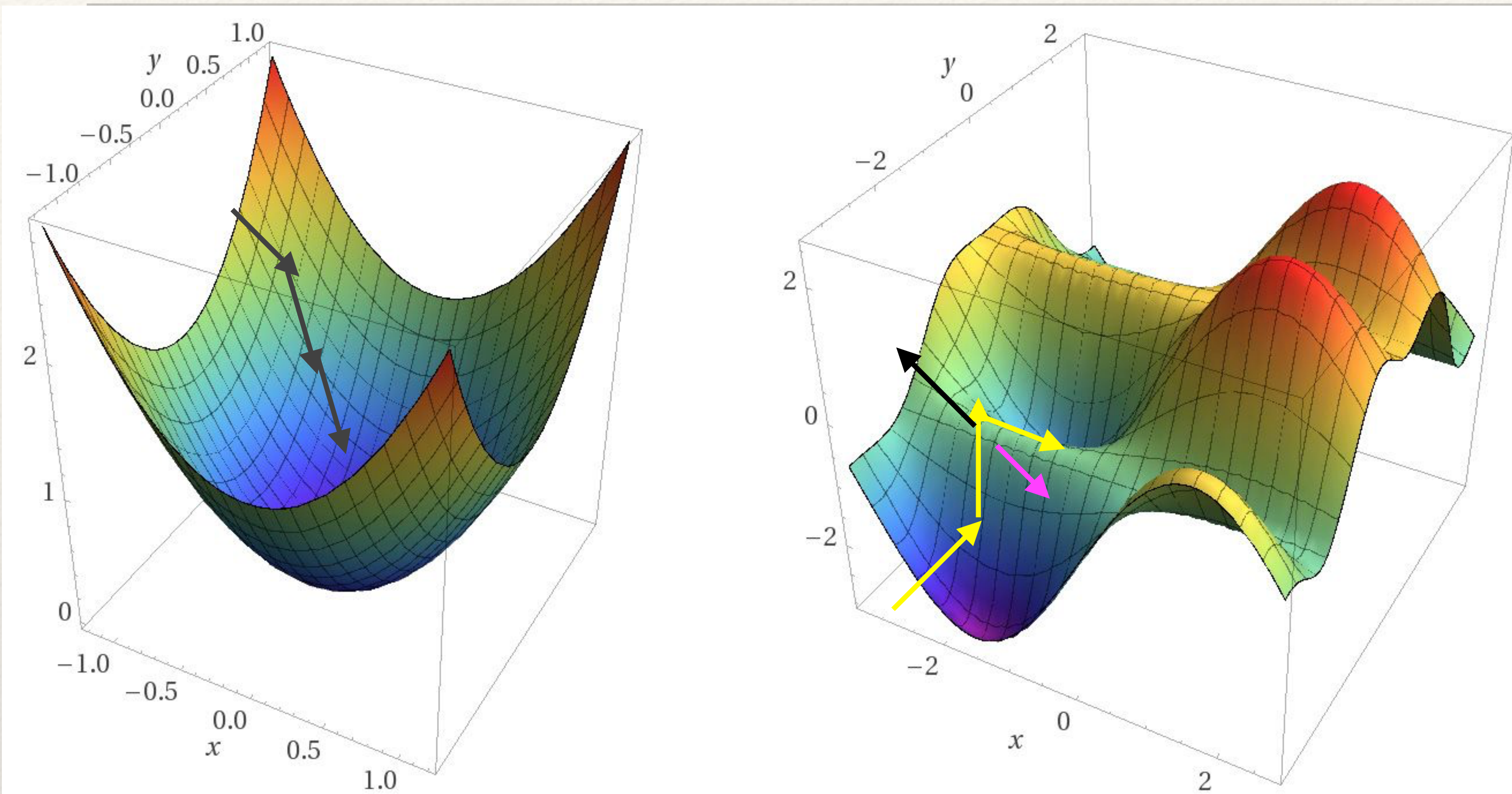
Nice equations, but what does this mean in practice?

we need to devise a way to obtain the values w which minimize the cost function, and this in general is a very complex procedure which involves taking derivatives of the cost function respect to the parameters

derivatives should not blow up
you should not get stuck on a local minimum
you should not hop too far and miss minima
and your model should work well on new data

all this, in a high-dimensional parameter space…

Regression: finding the best model



Minimisation done numerically the cost function can be *very complex*, have many minima

Often use tricks such as stochastic jumps, adding to the step decision higher momenta of the cost function, batching…

often use ***regularization***: term in the cost function which diminishes importance of features, so they don’t dominate the fit. Prevents overfitting.

Typical regularisations

LASSO (L1) $$\hat{\mathbf{w}} = \arg\min(|\mathbf{X}.\mathbf{w} - \mathbf{y}|^{\mathbf{2}} + \lambda|\mathbf{w}|)$$

RIDGE (L2) $$\hat{\mathbf{w}} = \arg\min(|\mathbf{X}.\mathbf{w} - \mathbf{y}|^{\mathbf{2}} + \lambda|\mathbf{w}|^{\mathbf{2}})$$

Types of problems

Data sample
(**in**-sample)
$\mathcal{D}(x_i, y_i)$
$i = 1 \dots n$
x_i inputs
y_i outputs

Understand / Learn
relations in/out
predict in->out
(**out**-sample)

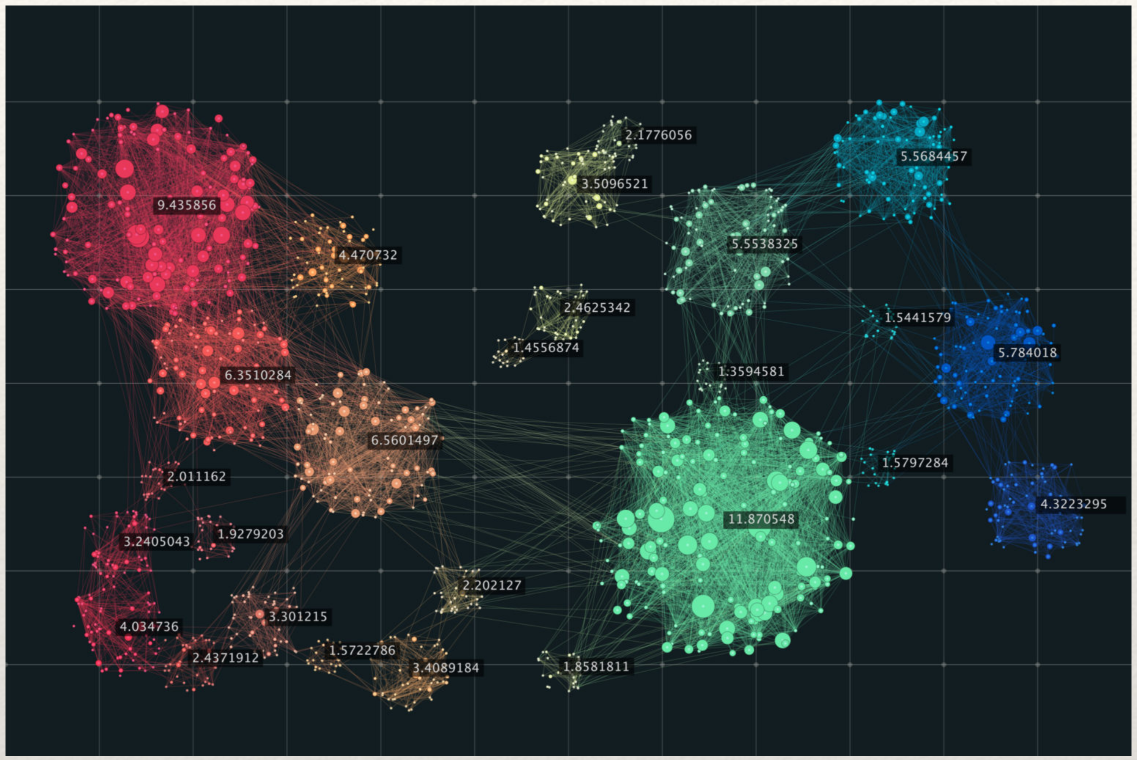
SUPERVISED

Data sample
(**in**-sample)
$\mathcal{D}(x_i)$
$i = 1 \dots n$
x_i inputs

Understand / Learn
relations in data
predict similar relations
in different data
(**out**-sample)

UNSUPERVISED

Unsupervised



Data is not labeled
Algorithm searches patterns without specific instructions
Finds criteria for 'distance' based on features and clusters or detects outliers / anomalies

Examples:
Relations among users
Computer vision

Types of supervised problems

Outputs

y

continuous

REGRESSION

x: characteristics of pp collisions

y: total number of events, kinematic distributions…

discrete

LOGISTIC REGRESSION
CLASSIFICATION

x: characteristics of an event

y: b-tag or no b-tag, quark or gluon jet, EM/HAD…

The simplest classification problem

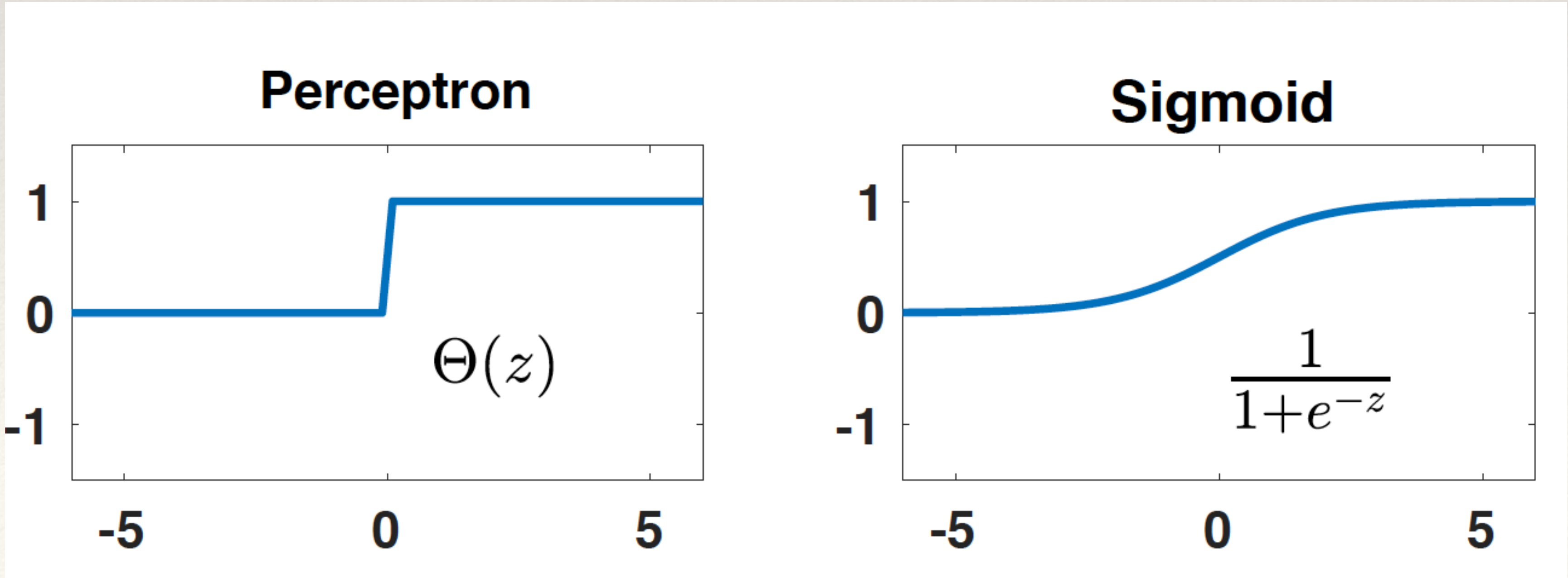
dataset $\mathcal{D}(x_i, y_i)$ with $y \in \{0, 1\}$ {no, yes}

logistic regression: probability datapoint *x_i* as true or false

$$P(y_i = 1) = f(\mathbf{x}_i^T \mathbf{w}) = 1 - P(y_i = 0).$$

e.g. event b-tagged or not, event new physics or not

f itself can be a function within 0 and 1



Tomorrow

We will talk about the BINARY problem
to move onto
NEURAL NETWORKS

Today

We will work an example of **multivariate linear regression**
(SCIKIT-SKLEARN)
[Predict bike rides depending on a bike-sharing scheme]
Learn about simple **data manipulation** (PANDAS)
and **data visualisation** (MATPLOTLIB/SEABORN)
Learn about L1 and L2 **regularisation**
Link to Google Colab notebook on REGRESSION

File > Save a copy / / send questions in Slack to everyone
Reconvene at 12:30