Introduction to Machine Learning

Mathieu Blondel

Google Research, Brain team

Credits: many contents and figures are borrowed from the scikit-learn mooc and scipy lecture notes.

Scope of this lecture

- A one-hour and a half long tutorial for absolute beginners in machine learning
- High-level concepts with almost zero maths

Scope of this lecture

- A one-hour and a half long tutorial for absolute beginners in machine learning
- High-level concepts with almost zero maths

What will be covered later this week but not today:

- Model-specific explanations (linear models, trees, neural networks)
- scikit-learn
- Optimization
- Deep learning

Outline

- Supervised learning
  - Classification
  - Regression
  - Generalization
- Model selection
  - Model complexity
  - Overfitting
  - Bias / variance trade-off

Supervised learning

Supervised learning

- Learning from examples (inductive learning)
- Training phase

- Inference (prediction) phase

Classification

- Given an input, predict the class (category) it belongs to
- Examples
  - Given the picture of an astronomical object, determine whether it is a star, a galaxy, etc.
  - Given the picture of a person, identify the person
  - Given an email, detect whether it is a spam or not
  - Given the spectogram of a spoken word, recognize the command being spoken

Regression

- Given an input, predict a real (continuous) variable associated with it
- Examples
    - Given the profile of a person, predict his/her salary
    - Given the characteristics of a house, predict its price
    - Given a patient's DNA information, predict his/her life expectancy

The iris dataset

- A famous dataset in statistics and machine learning

Setosa

Versicolor

Viriginica

Sepal length	Sepal width	Petal length	Petal width		Iris type
6cm	3.4cm	4.5cm	1.6cm		versicolor
5.7cm	3.8cm	1.7cm	0.3cm		setosa
6.5cm	3.2cm	5.1cm	2cm		virginica
5cm	3.cm	1.6cm	0.2cm		setosa

Data matrix and target vector

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \vdots \end{bmatrix} = \begin{bmatrix} 6 & 3.4 & 4.5 & 1.6 \\ 5.7 & 3.8 & 1.7 & 0.3 \\ 6.5 & 3.2 & 5.1 & 2 \\ 5 & 3 & 1.6 & 0.2 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ \vdots \end{bmatrix}$$

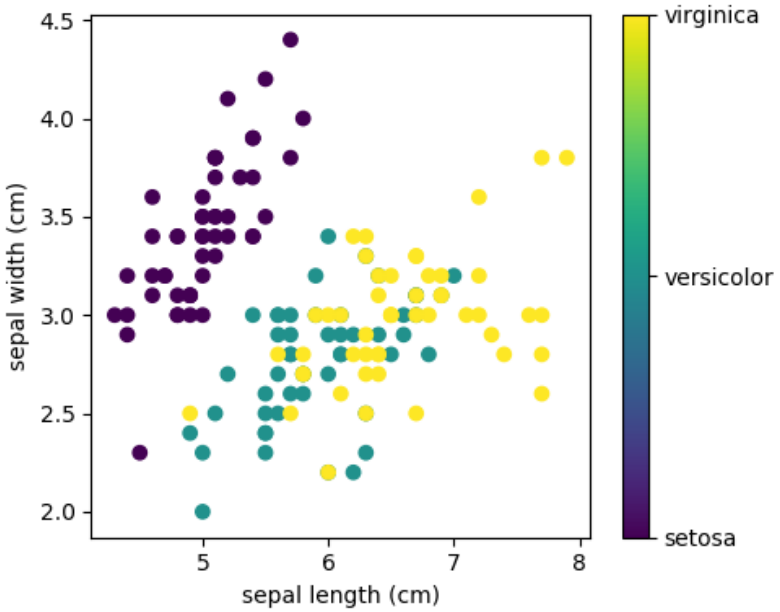
$\mathbf{X}$ is called the data matrix and consists of n samples (observations) with d features (attributes)

$\mathbf{x}_i$ is called a feature vector

$\mathbf{y}$ is called the target or label vector

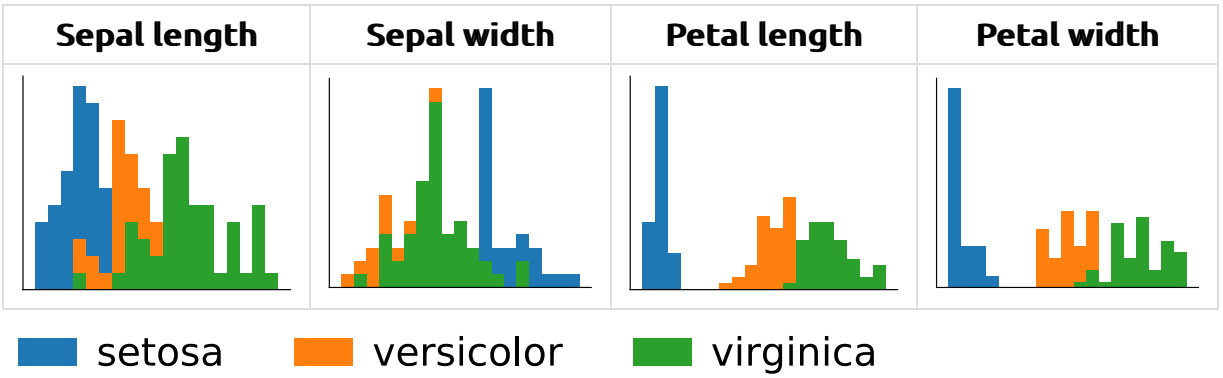
y_i is called a target or label

Samples as points in a space



Samples from the iris datasets live in a 4-dimensional space, we only use 2 dimensions for visualization purposes.

Inferring rules from data



We can *infer* from the data that setosa irises have small petals.

Learning algorithms automate this process!

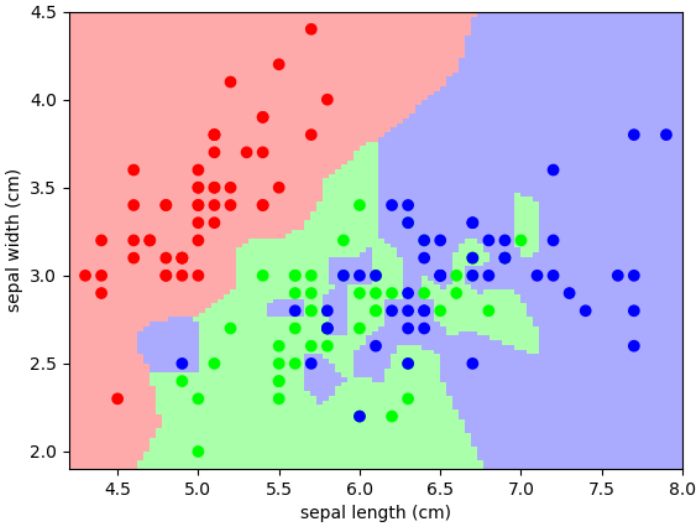
Learning a model from examples

- A model can be seen as a function f that maps a feature vector $\mathbf{x}$ (input) to a target y (output).
- Given input-output pairs (training examples) $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_n, y_n)$, we want to find a f such that

$$f(\mathbf{x}_i) \approx y_i$$

Example: the nearest neighbor algorithm

$f(\mathbf{x})$ searches for the example $\mathbf{x}_i$ with smallest distance $d(\mathbf{x}, \mathbf{x}_i)$ and returns the label y_i associated with this example.



US Census data

Age	Workclass	Education	Marital-status	Occupation	Relationship	Race	Sex	Capital-gain	Hours-per-week	Native-country	Class
25	Private	11th	Never-married	Machine-op-inspct	Own-child	Black	Male	0	40	United-States	<=50K
38	Private	HS-grad	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	50	United-States	<=50K
28	Local-gov	Assoc-acdm	Married-civ-spouse	Protective-serv	Husband	White	Male	0	40	United-States	>50K
44	Private	Some-college	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	40	United-States	>50K

Does this person make more than 50K per year?

Generalizing

- In machine learning, we want to predict on new data instances. That is, for a feature vector $\mathbf{x}$ not seen in the training set, we want to build $f(\mathbf{x})$.
- In the census example, we want to predict the income of new individuals, with a combination of jobs and demographics that we have never seen.
- Many sources of variability: age workclass, education, marital-status, occupation, ...

+ Noise: unexplainable variance

Memorizing

- Store all known individuals (the census)
- Given a new individual, predict the income of its closest match in our database (nearest neighbor predictor)

Trying out this strategy on the data we have, the census, *what error rate do we expect?*

Memorizing

- Store all known individuals (the census)
- Given a new individual, predict the income of its closest match in our database (nearest neighbor predictor)

Trying out this strategy on the data we have, the census, *what error rate do we expect?*

0 errors

Memorizing

- Store all known individuals (the census)
- Given a new individual, predict the income of its closest match in our database (nearest neighbor predictor)

Trying out this strategy on the data we have, the census, *what error rate do we expect?*

> **0 errors**

Yet, we will make errors on **new** data

Generalizing $\neq$ Memorizing

"test" data $\neq$ "train" data

Data on which the predictive model is applied

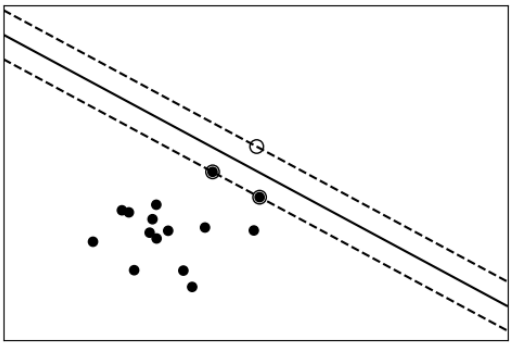
Data used to learn the predictive model

- Different sampling of noise
- Unobserved combination of features

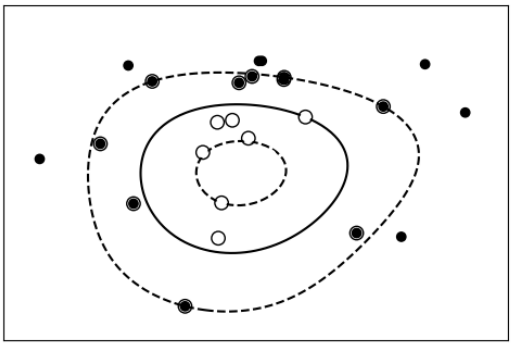
Model selection

Model complexity

- There exists a wealth a machine learning models, some more complex and expressive than others.
- What are the implications of model complexity?
- How do we pick one model rather than another?

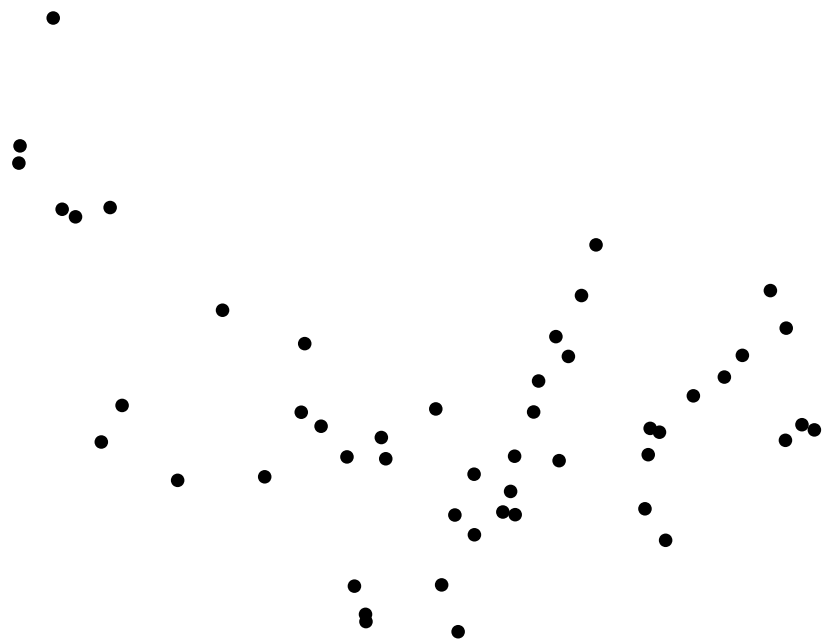


Linear

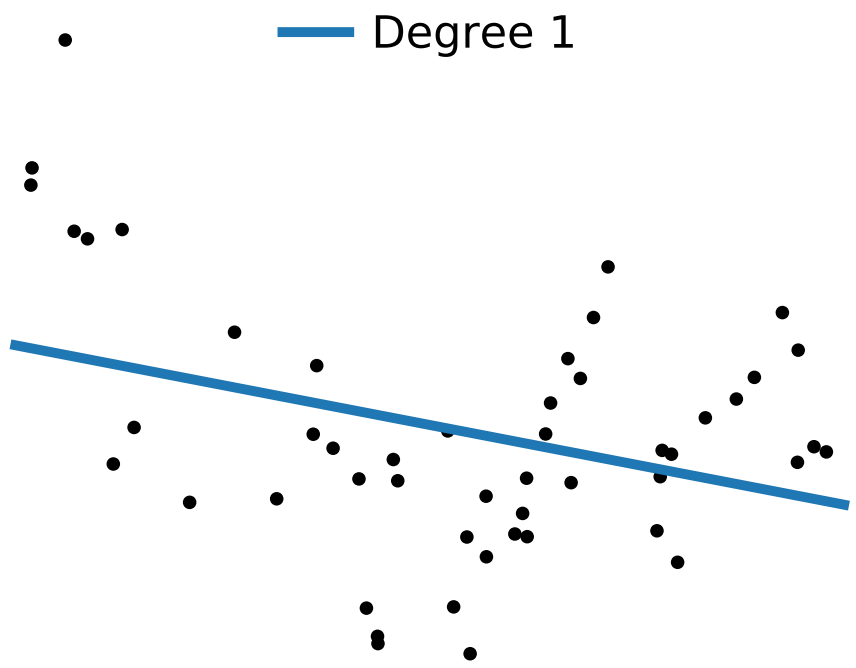


Non-linear

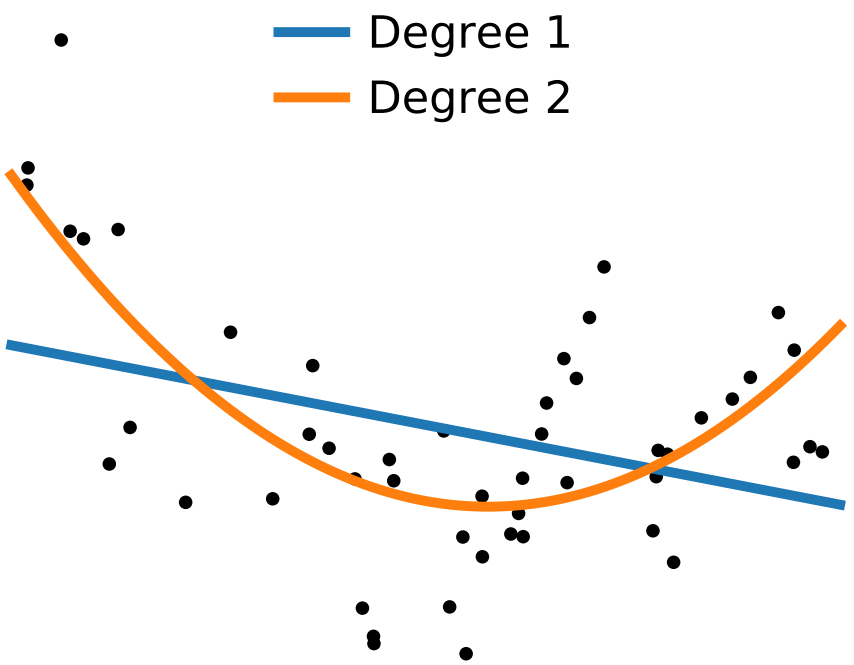
Varying the degree of a polynomial



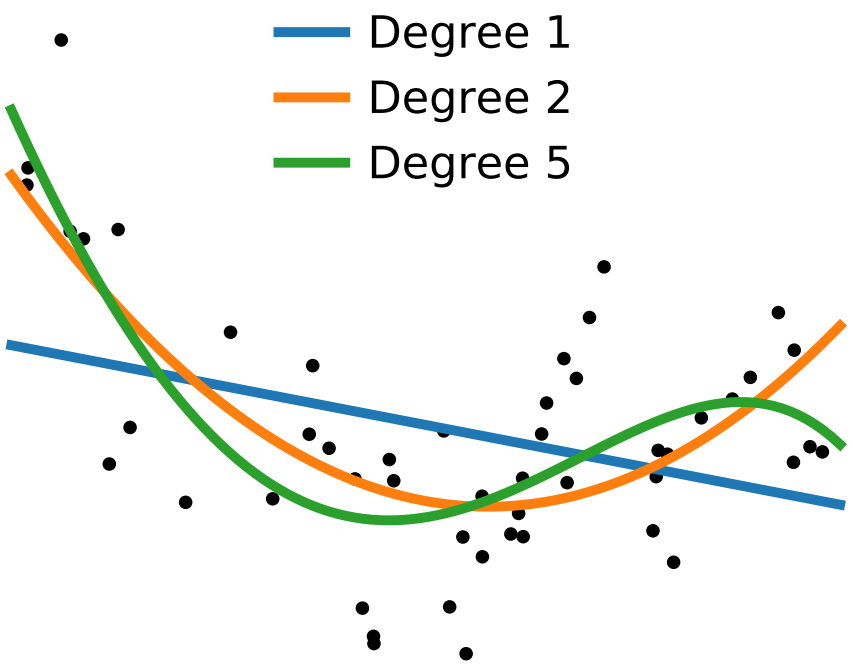
Varying the degree of a polynomial



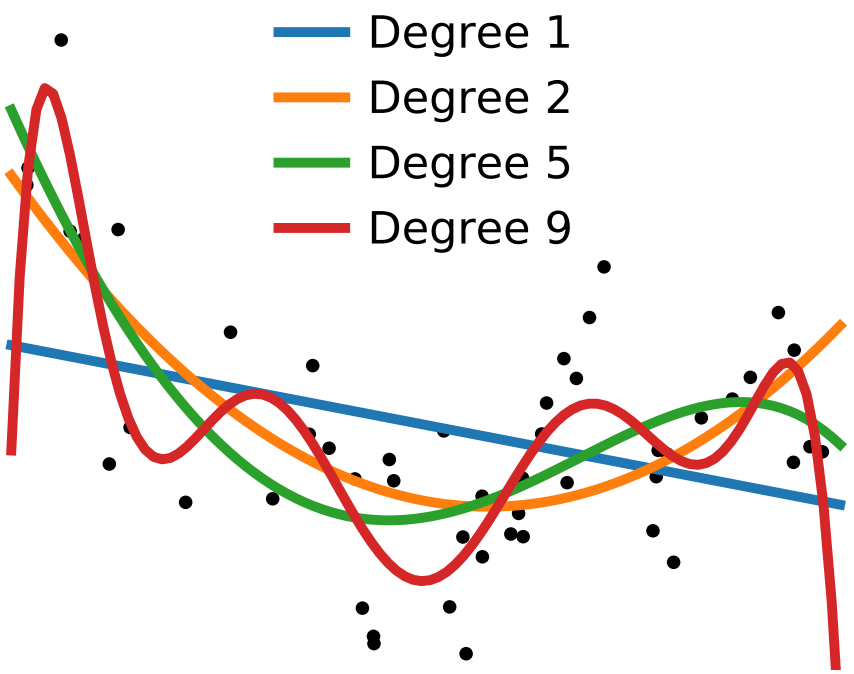
Varying the degree of a polynomial



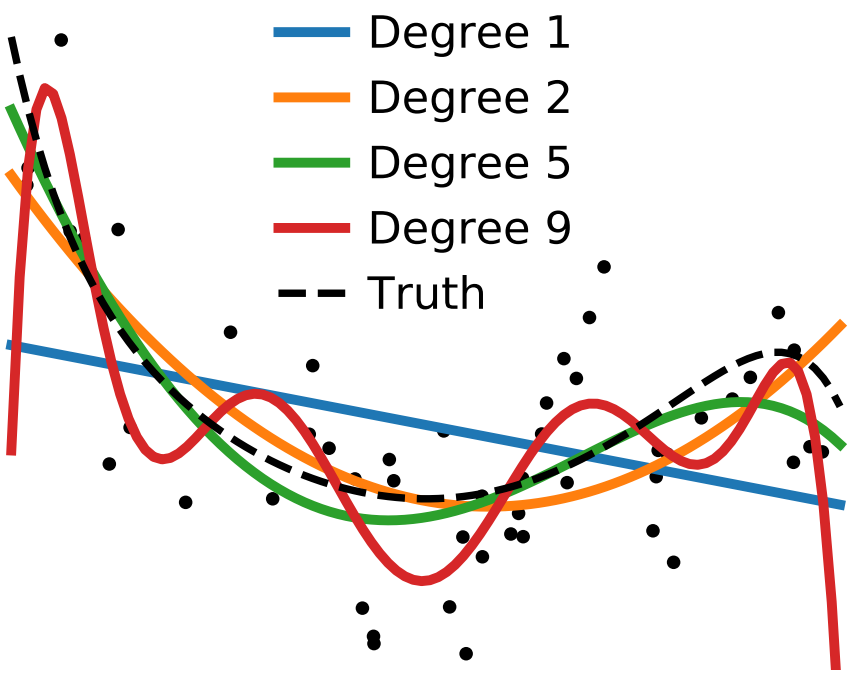
Varying the degree of a polynomial



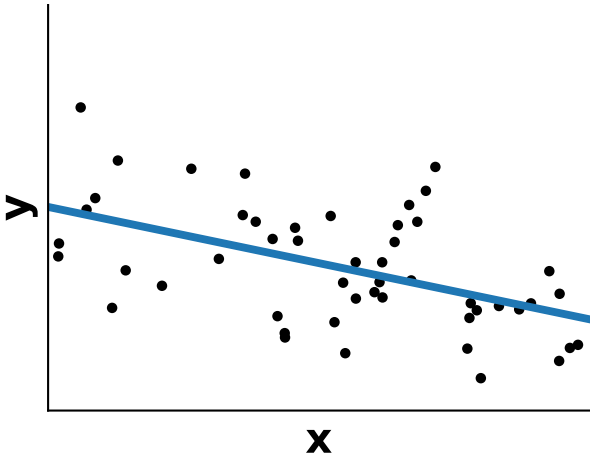
Varying the degree of a polynomial

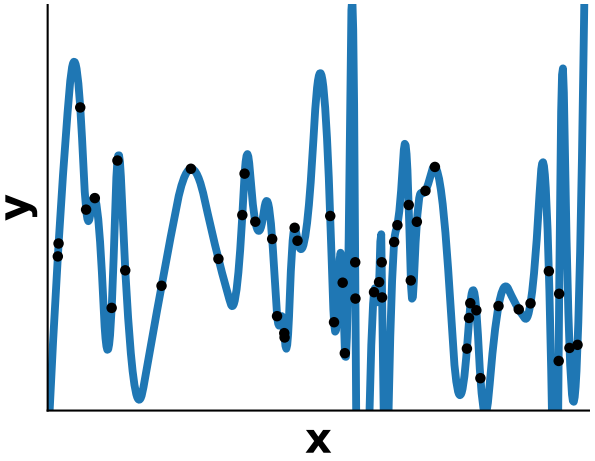


Varying the degree of a polynomial

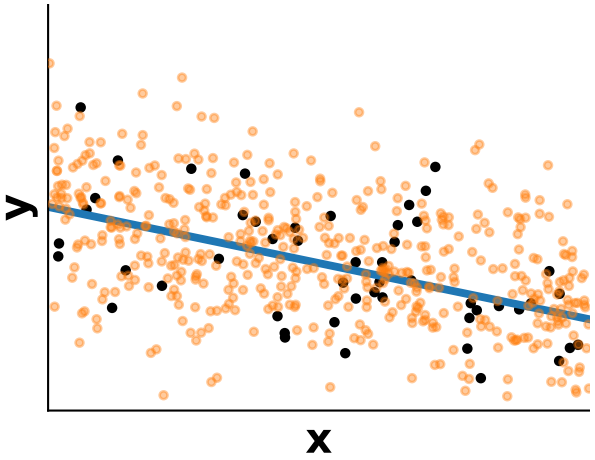


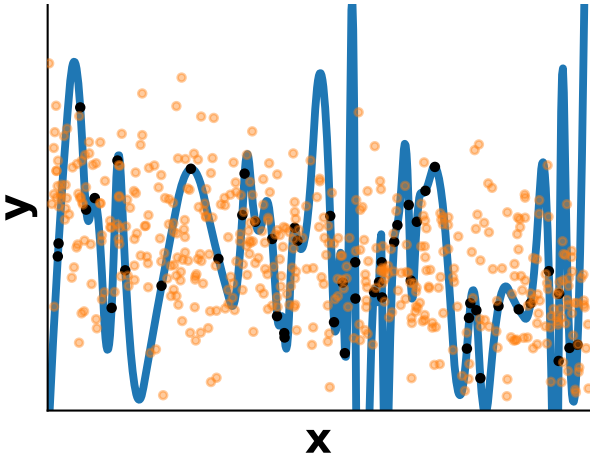
Which model do you prefer?



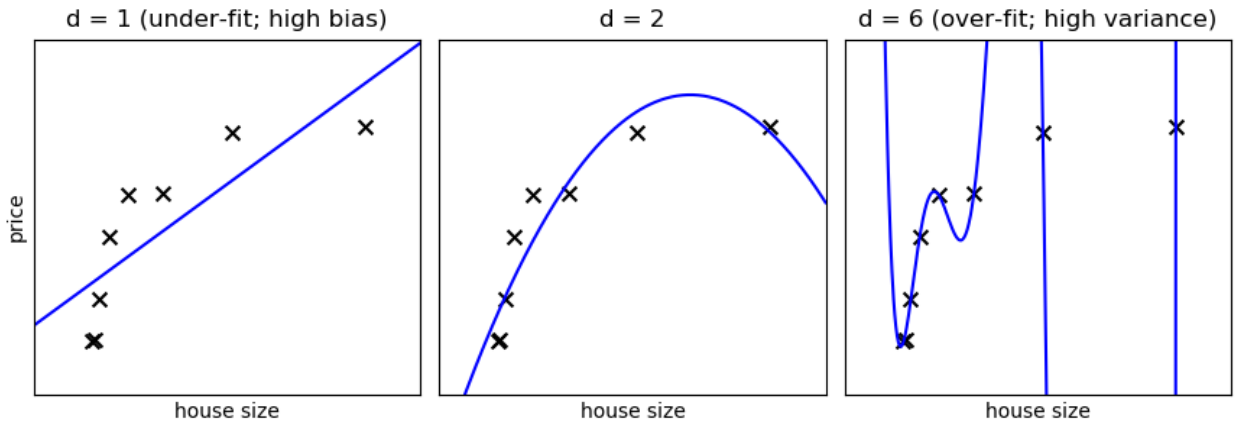


Which model do you prefer? (new data)





Bias / Variance trade-off



- High bias: the model is too simple / constrained (underfit)
- High variance: the model is too expressive / flexible (overfit)
- Need to find a good trade-off

Underfit versus overfit

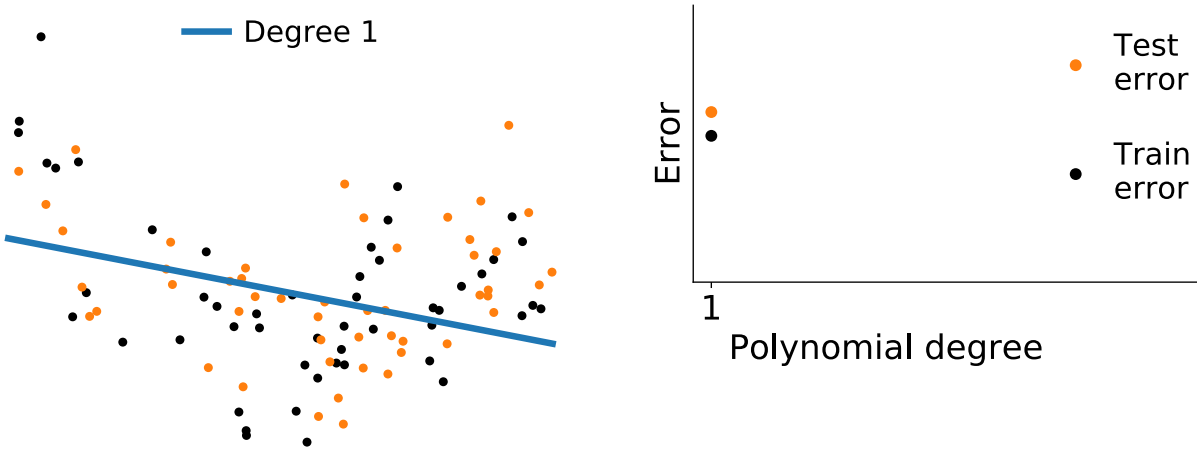
High bias

High variance

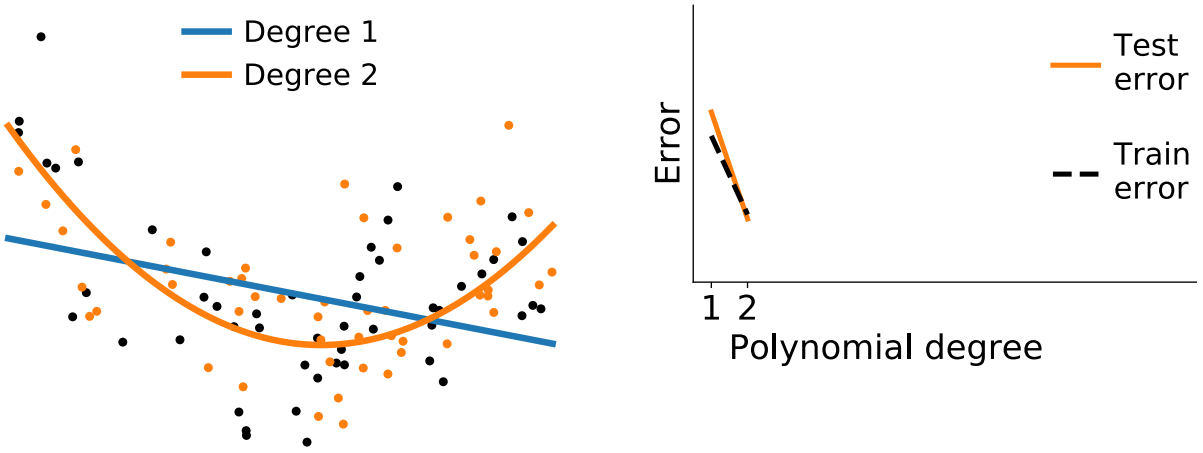
Train vs. test error

- Train set error: how well did the learning algorithm tune the model parameters
- Test set error: how well does the model generalize to new data

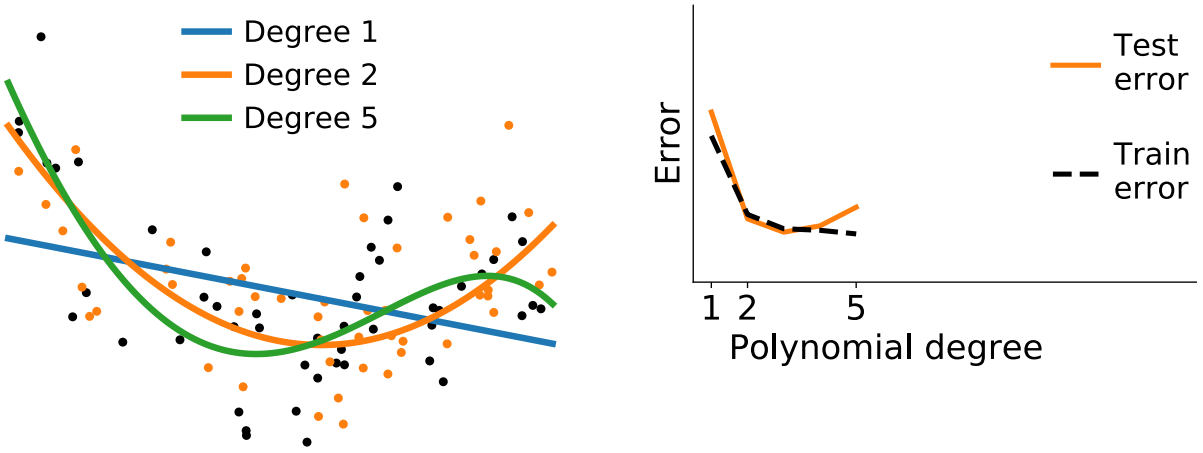
Train vs test error: effect of model complexity



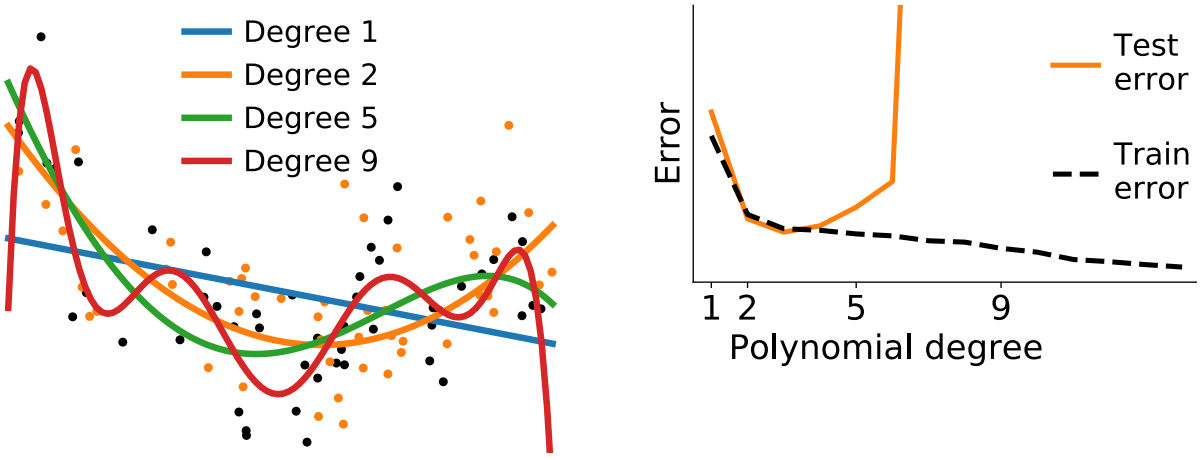
Train vs test error: effect of model complexity



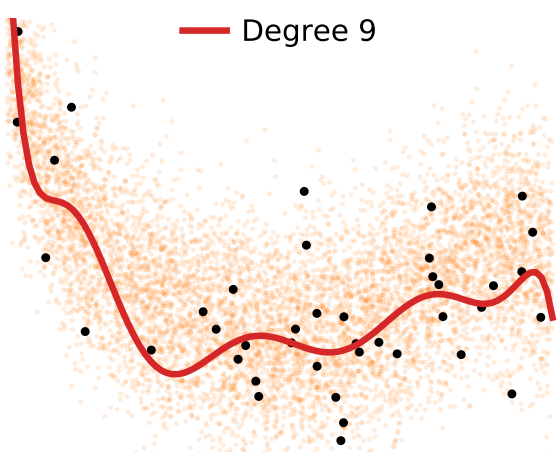
Train vs test error: effect of model complexity

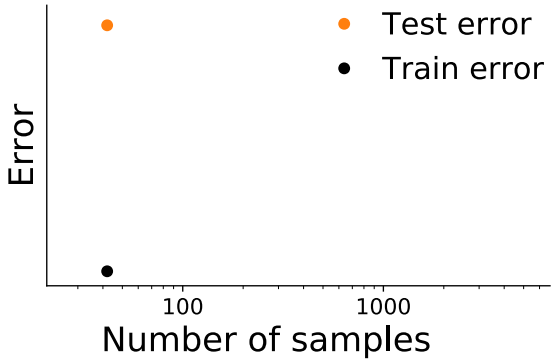


Train vs test error: effect of model complexity

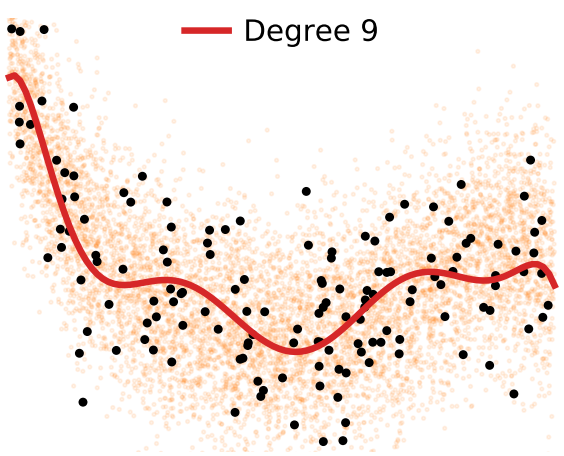


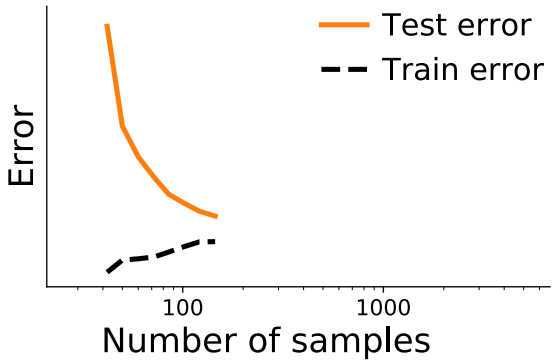
Train vs test error: effect of sample size



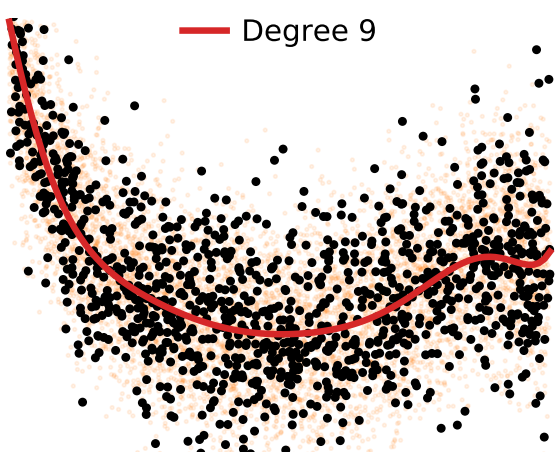


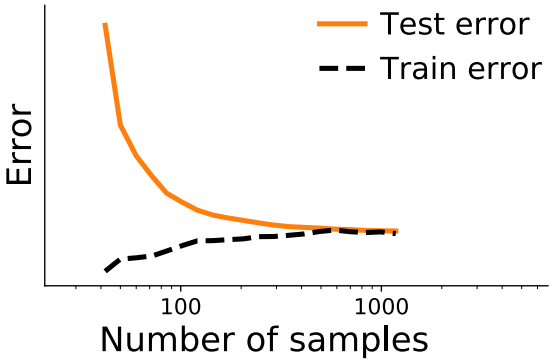
Train vs test error: effect of sample size



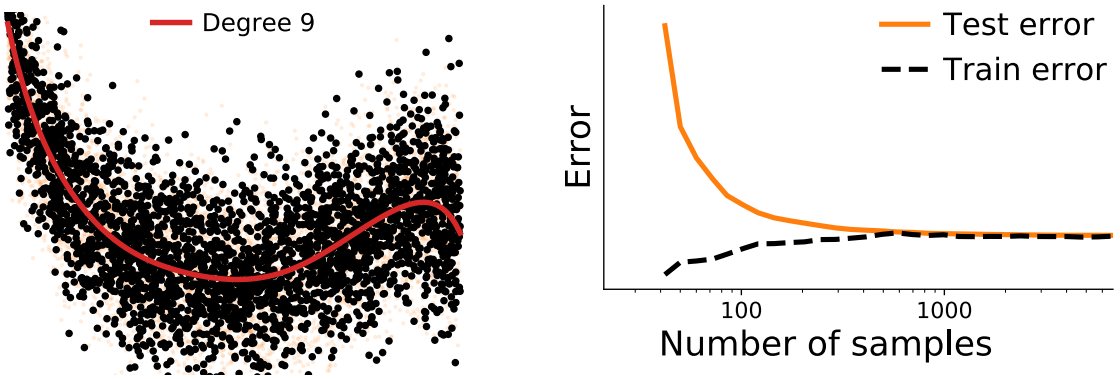


Train vs test error: effect of sample size





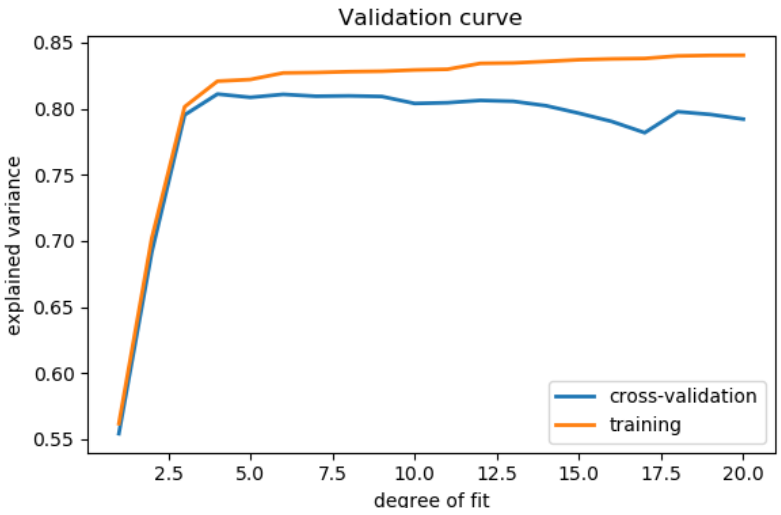
Train vs test error: effect of sample size



Diminishing returns phenomenon

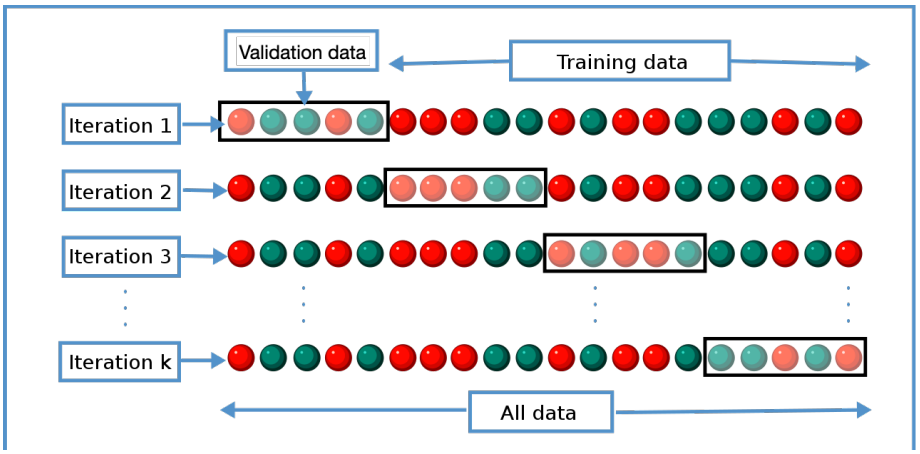
Test vs. validation error

- Hyper-parameters: parameters that are not directly tuned by the learning algorithm
- Typically controls complexity (e.g. the degree parameter)
- Test data should never be used for choosing hyper-parameters
- Use validation data instead (aka held-out data)



Cross-validation

- Repeatedly split the data into training and validation sets and average the accuracy scores
- More robust estimation of the quality of hyper-parameters, but requires to train the model several times



Take home messages

- Machine learning is about extracting from *data* models that *generalize* to new observations.
- Bias (underfit) vs. Variance (overfit) trade-off.
- Never tune hyper-parameters / model complexity against the test data; use validation data or cross-validation instead.

Lab work

- Introduction to Python and NumPy
- Either use Google Colab (needs a gmail account)
- Or download the notebook locally (needs, e.g., anaconda)
- We will be present on discord if you have any questions