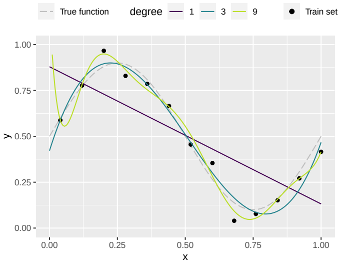
Introduction to Machine Learning

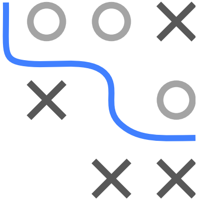
Evaluation
Training Error



Learning goals

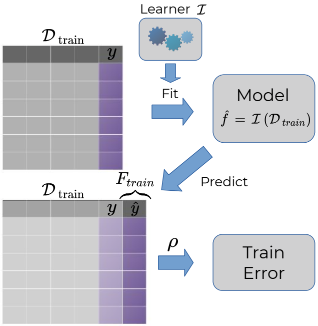
- Understand the definition of training error
- Understand why train error is unreliable for models of higher complexity when overfitting can occur

TRAINING ERROR



Simply plugin predictions for data that model has been trained on:

$$\rho(\mathbf{y}_{\text{train}}, \boldsymbol{F}_{\text{train}}) \text{ where } \boldsymbol{F}_{\text{train}} = \begin{bmatrix} \hat{f}_{\mathcal{D}_{\text{train}}}(\mathbf{x}^{(1)}_{\text{train}}) \\ \dots \\ \hat{f}_{\mathcal{D}_{\text{train}}}(\mathbf{x}^{(m)}_{\text{train}}) \end{bmatrix}$$

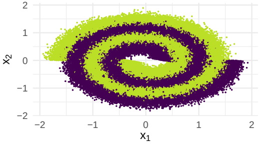


A.k.a. apparent error or resubstitution error.

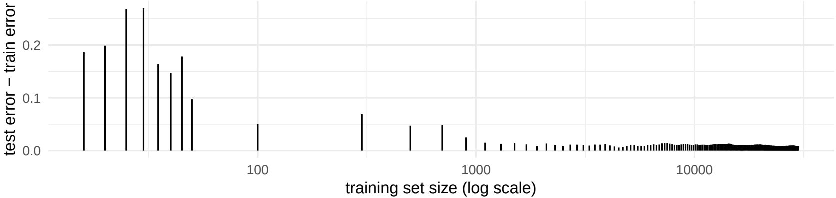
EXAMPLE 1: KNN

For large data, and some models, train error **can maybe** yield a good approximation of the GE:

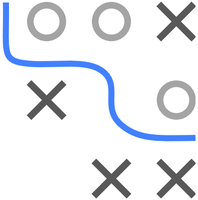
- Use k-NN ($k = 15$).
- Up to 30K points from `spirals` to train.
- Use very large extra set for testing (to measure "true GE").



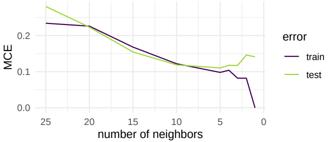
We increase train size, and see how gap between train error and GE closes.

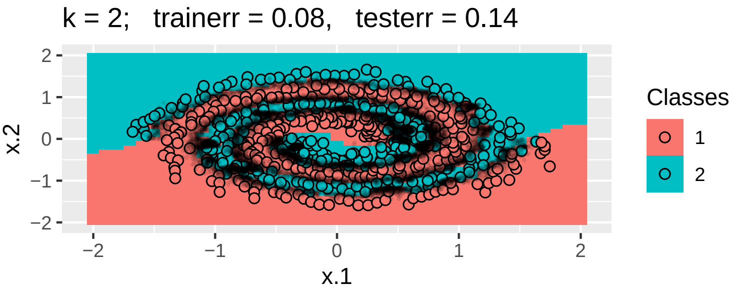


EXAMPLE 1: KNN



- Fix train size to 500 and vary k.
- Low train error for small *k* is deceptive. Model is very local and overfits.

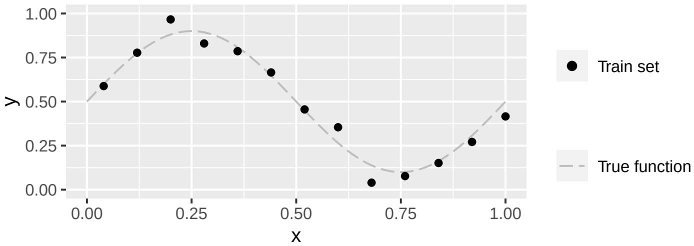




Black region are misclassifications from large test test.

EXAMPLE 2: POLYNOMIAL REGRESSION

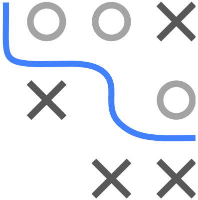
Sample data from $0.5 + 0.4 \cdot \sin(2\pi x) + \epsilon$



We fit a d^{th}-degree polynomial:

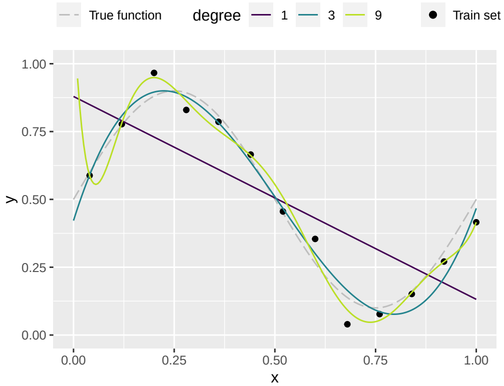
$$f(\mathbf{x} \mid \boldsymbol{\theta}) = \theta_0 + \theta_1 \mathbf{x} + \cdots + \theta_d \mathbf{x}^d = \sum_{j=0}^{d} \theta_j \mathbf{x}^j.$$

EXAMPLE 2: POLYNOMIAL REGRESSION



Simple model selection problem: Which d?

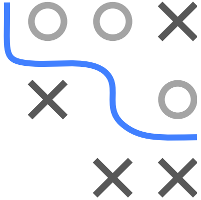
Visual inspection vs quantitative MSE on training set:



- $d = 1$: MSE = 0.036: clearly underfitting
- $d = 3$: MSE = 0.003: pretty OK
- $d = 9$: MSE = 0.001: clearly overfitting

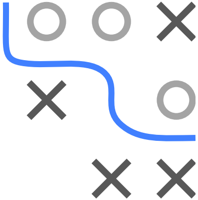
Using the train error chooses overfitting model of maximal complexity.

TRAIN ERROR CAN EASILY BECOME 0



- For 1-NN it is always 0 as each $\mathbf{x}^{(i)}$ is its own NN at test time.
- Extend any ML training in the following way: After normal fitting, we also store the training data. During prediction, we first check whether x is already stored in this set. If so, we replicate its label. The train error of such an (unreasonable) procedure will be 0.
- There are so called interpolators - interpolating splines, interpolating Gaussian processes - whose predictions can always perfectly match the regression targets, they are not necessarily good as they will interpolate noise, too.

CLASSICAL STATISTICAL GOF MEASURES



- **Goodness-of-fit** measures like R^2, likelihood, AIC, BIC, deviance are all based on the training error.
- For models of restricted capacity, and enough data, and non-violated distributional assumptions: they might work.
- Hard to gauge when that breaks, for high-dim, more complex data.
- How do you compare to non-param ML-like models?

Out-of-sample testing is probably always a good idea!