

# Machine Learning for econometrics

## Statistical learning and regularized linear models

---

Matthieu Doutreligne

January 10, 2025

A lot of today's content is taken from the excellent sklearn mooc (Estève et al., 2022)

# Today's program

- Last session: importance of causal variable status
- Today: **predictive inference** in high dimensions
  - ▶ Statistical learning basics
  - ▶ Regularized linear models for predictive inference
  - ▶ Hands-on with scikit-learn
-

# Today's program

- Last session: importance of causal variable status
- Today: **predictive inference** in high dimensions
  - Statistical learning basics
  - Regularized linear models for predictive inference
  - Hands-on with scikit-learn
- Next session:
  - Flexible models: Trees, Random Forests, Gradient Boosting
  - Practical scikit-learn

# Table of contents

1. Statistical learning framework
2. Motivation: why prediction?
3. Statistical learning theory and intuitions
4. Regularized linear models for predictive inference
5. Python hands-on: Common pitfalls in the interpretation of coefficients of linear models

## Statistical learning framework

# Statistical learning, ie. predictive inference

## Goal

- Predict the value of an outcome based on one or more input variables.

## Setting

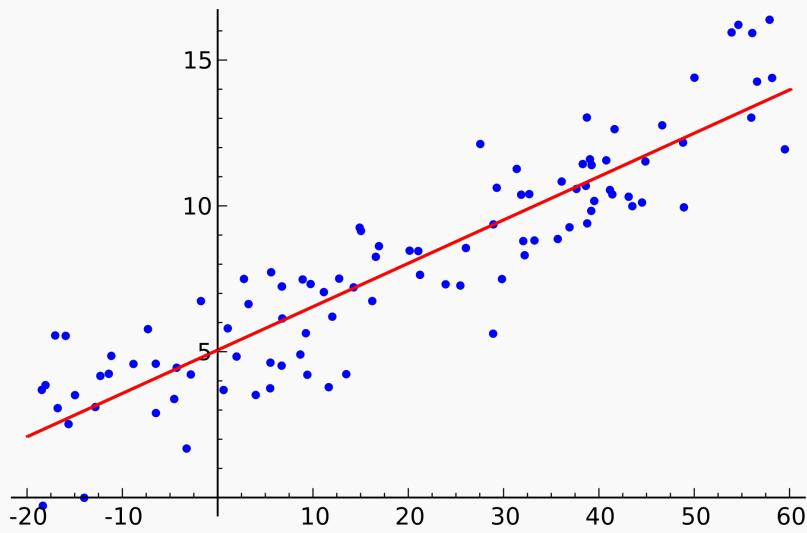
- Data:  $n$  pairs of (features, outcome),  $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$  identically and independently distributed (i.i.d.) from an unknown distribution  $P$ .
- Goal: find a function  $\hat{f} : \mathcal{X} \rightarrow \mathcal{Y}$  that approximates the true value of  $y$  ie. for a new pair  $(x, y)$ , we should have:

$$\hat{y} = \hat{f}(x) \approx y$$

## Vocabulary

Finding the appropriate model  $\hat{f}$  is called learning, training or fitting the model.

# Statistical learning, two classes of problems

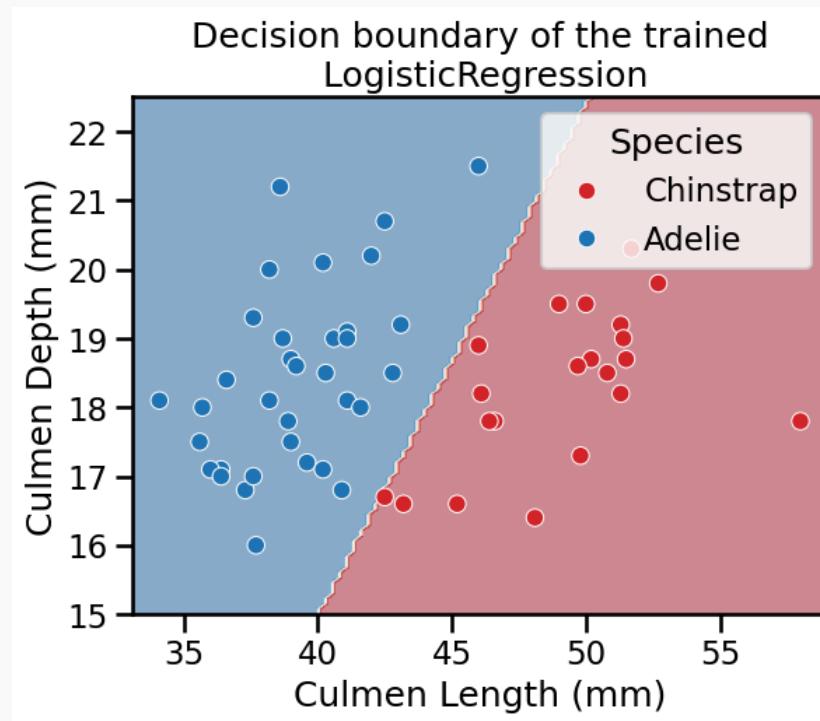


## Regression

- The outcome is continuous: eg. wage prediction
- The error is often measured by the mean squared error (MSE):

$$\text{MSE} = \mathbb{E} \left[ (Y - \hat{f}(X))^2 \right]$$

# Statistical learning, two classes of problems



## Classification

- Outcome is categorical: eg. diagnosis, loan default, ...
- Error is often measured with accuracy:

$$\text{Misclassification rate} = \mathbb{E}[\mathbb{1}(Y \neq \hat{f}(X))]$$

with  $\hat{f} \in \{0, 1\}$  for binary classification

## Motivation: why prediction?

# Why do we need prediction for ?

## Statistical inference

- Goal: infer some intervention effect with a causal interpretation
- Require to regress “away” the relationship between the treatment or the outcome and the confounders -> more on this in sessions on Double machine learning.

## Predictive inference

- Some problems in economics requires accurate prediction without a causal interpretation (Kleinberg et al., 2015)
- Eg. Stratifying on a risk score (loan, preventive care, ...)

# Do we need more than linear models?

Let:

- $p$  is the number of features
- $n$  is the number of observations

## Maybe no

- Low-dimensional data:  $n \gg p$
- No non-linearities, no or few interactions between features

## Maybe yes

- High-dimensional data: ie.  $p \gg n$
- Non-linearities, many interactions between features

# What high-dimension means: Is $p >> n$ common in economics?

## **Characteristics of the dataset that can lead to high-dimensionality**

- Categorical variables with high cardinality, eg. job title, diagnoses...
- Text data: eg. job description, medical reports...
- Technical regressors to handle non-linearities, eg. polynomials, splines, log, ...

# What high-dimension means, concrete example

- Population referencement dataset, individual file (INSEE):  $n=19\ 735\ 576$ ;  $p=88$  🤝
- But many variables with cardinality: more than 555 pairs of (variable, category).
- Adding interaction of degree 2:  $\binom{p}{2} = \binom{555}{2} = \frac{555*554}{2} = 153735$  features 😭
- Adding interactions of any degree:  $2^p - p - 1 = 2^{555} - 554$  🎉

## Is this common? Yes

- Categorical with high cardinality or text data are increasingly common.
- Image data is high-dimensional by nature.
- Automation of data collection and storage leads to more datasets and more variables.

# Some examples from area with high dimensional data

## Some examples

- The Current Population Survey (CPS) dataset has hundreds of variables, many of which are categorical
- The Système National des Données de Santé (SNDS) in France = healthcare claims : many hundreds of variables, many of which are categorical.

# Some examples from area with high dimensional data

## Some examples

- The Current Population Survey (CPS) dataset has hundreds of variables, many of which are categorical
- The Système National des Données de Santé (SNDS) in France = healthcare claims : many hundreds of variables, many of which are categorical.

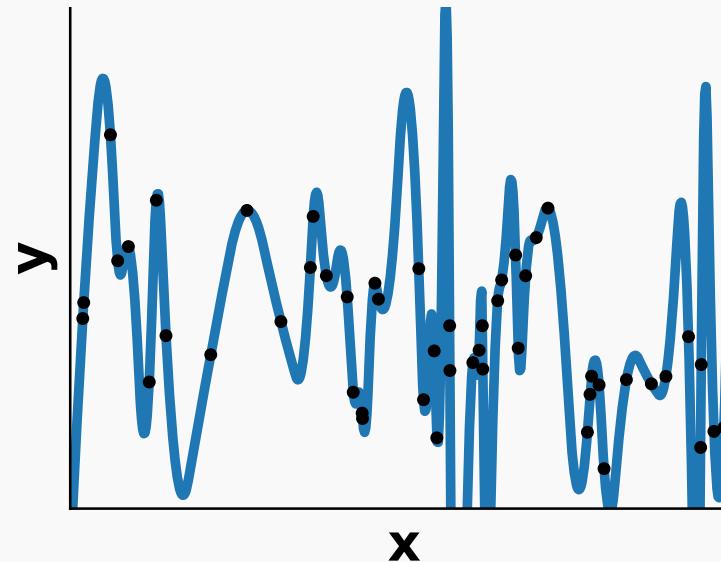
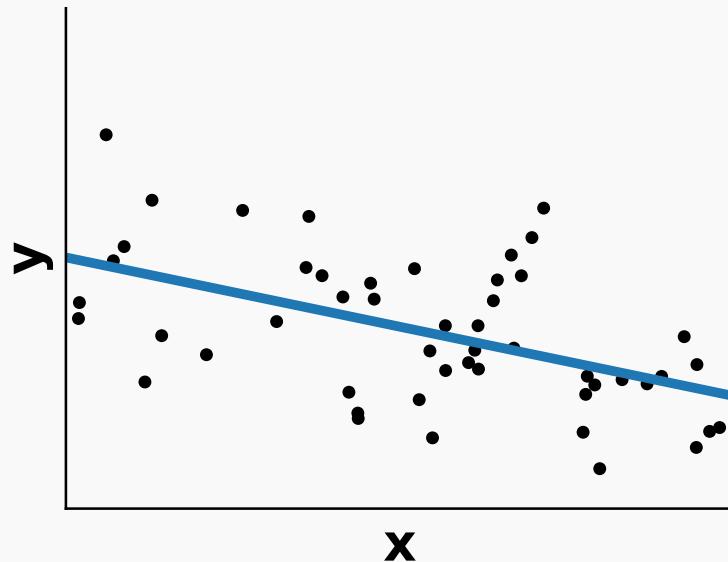
## Other area

- Country characteristics in cross-country wealth analysis,
- Housing characteristics in house pricing/appraisal analysis,
- Product characteristics at the point of purchase in demand analysis.

# Statistical learning theory and intu- tions

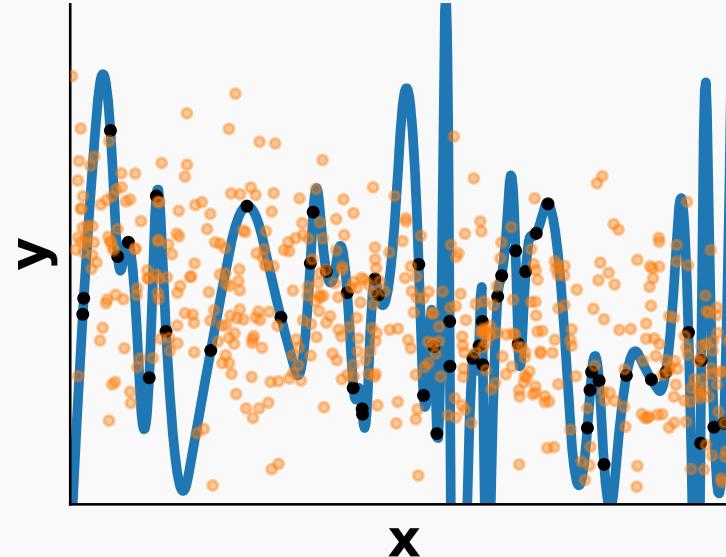
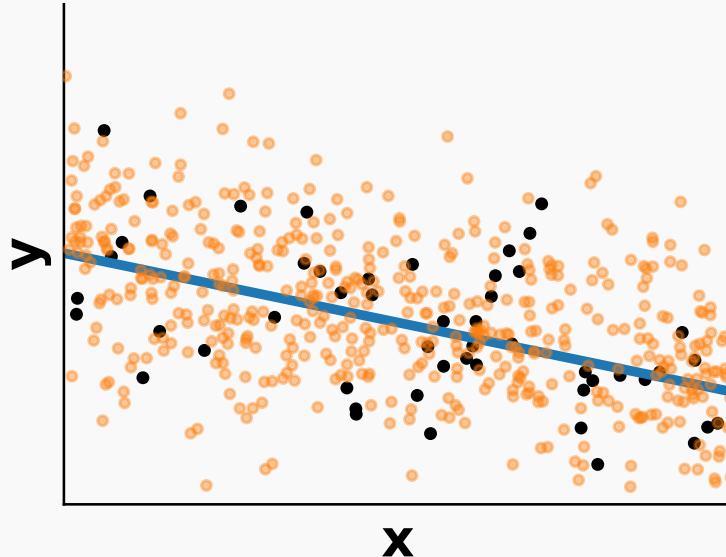
# Under vs. overfitting

Which data fit do you prefer?



# Under vs. overfitting

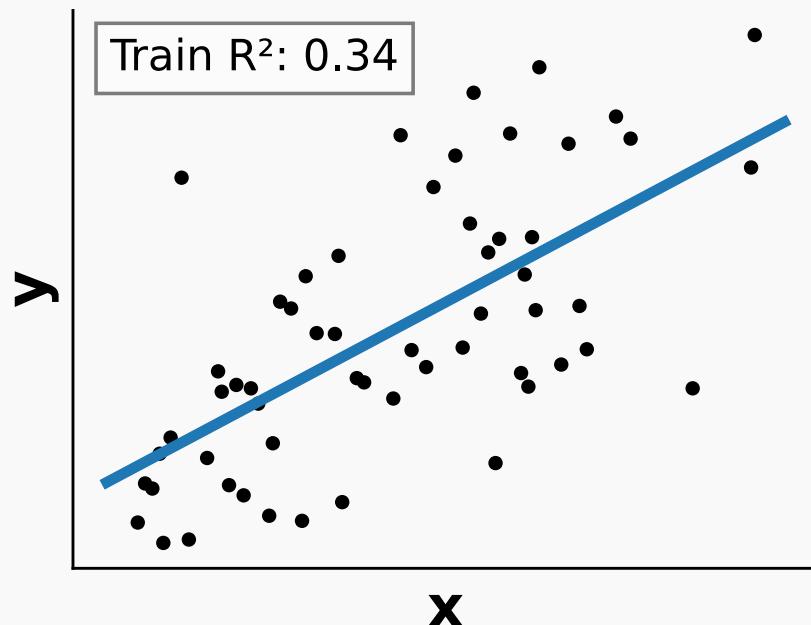
Which data fit do you prefer? (new data incoming)



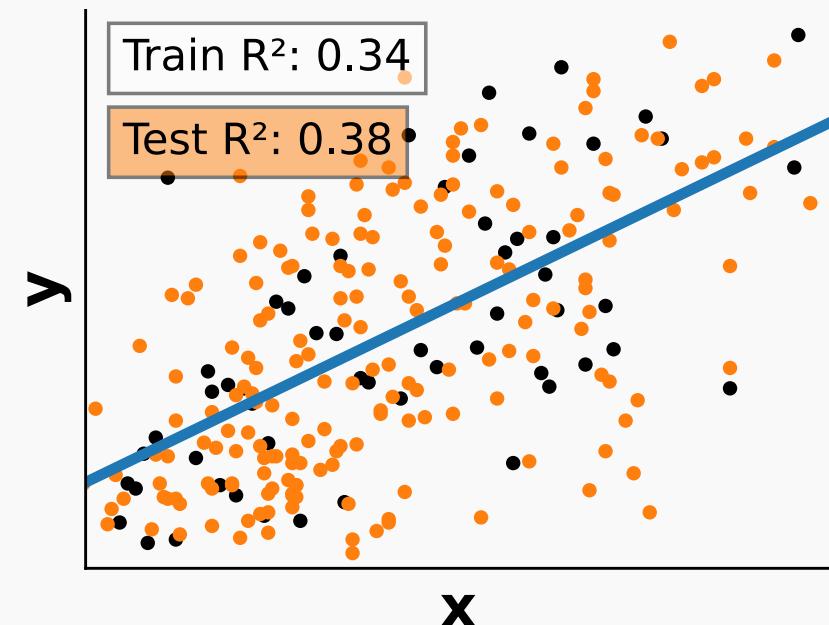
- Answering this question might be hard.
- Goal: create models that generalize.
- The good way of framing the question is: **how will the model perform on new data?**

# Train vs test error: simple models

**Measure the errors on the training data**  
= fitting



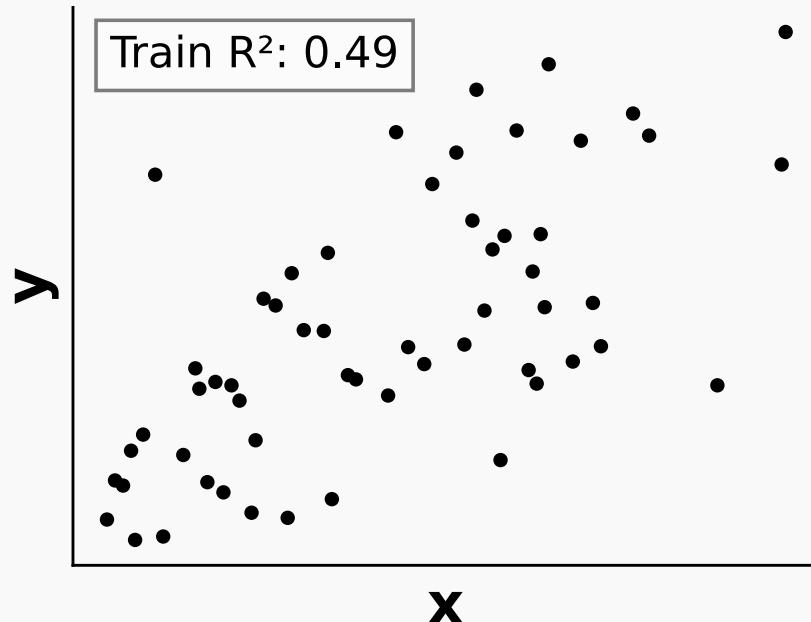
**Measure the performances on test data**  
= generalization



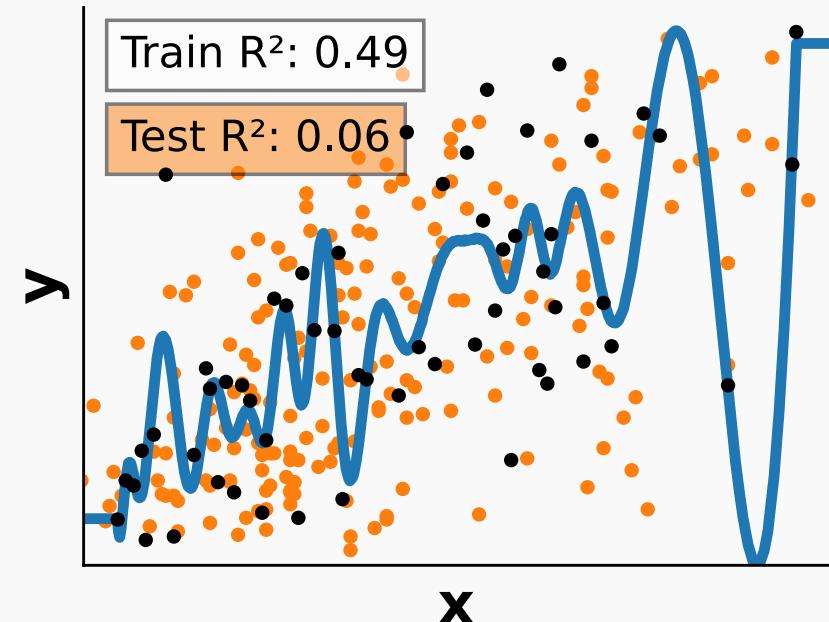
Here, no problem of overfitting: train vs test error are similar.

# Train vs test error: flexible models

**Measure the errors on the training data**  
= fitting

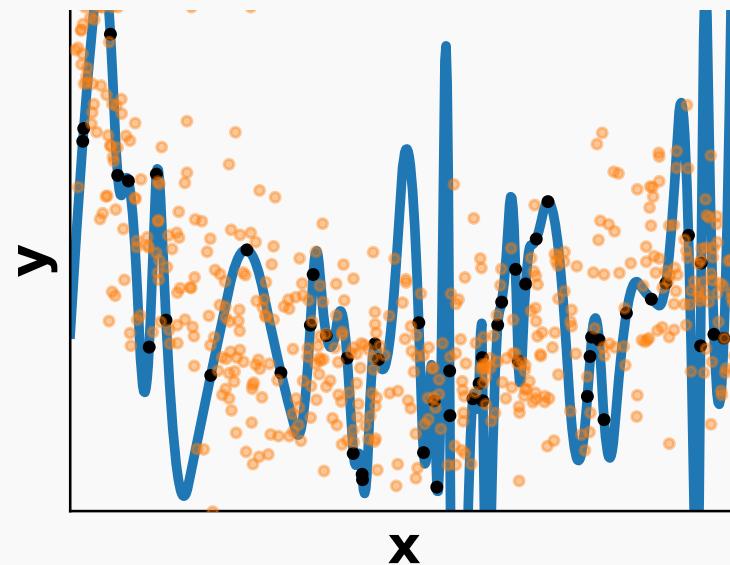
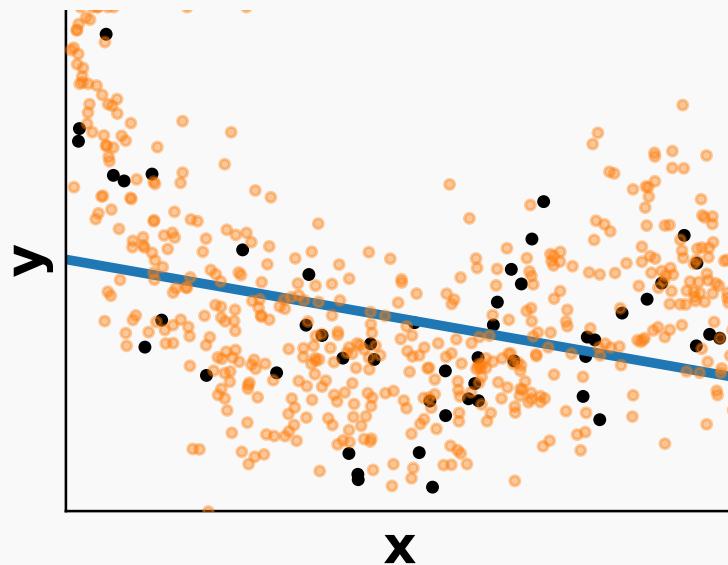


**Measure the performances on test data**  
= generalization

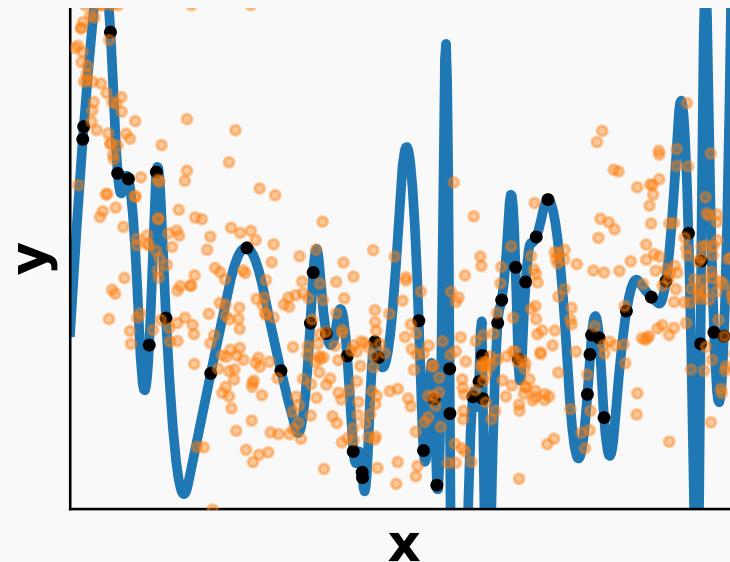
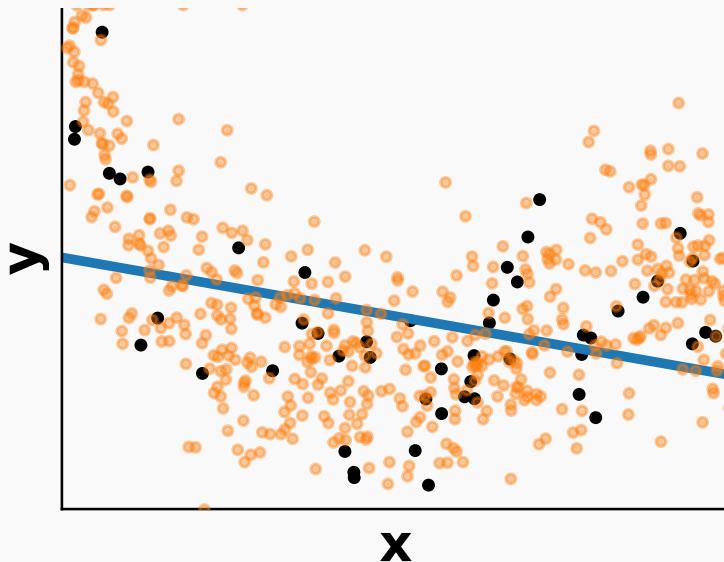


Overfitting: the model is too complex and captures noise.

# How to choose the complexity of the model?



# How to choose the complexity of the model?



This trade-off is called **Bias variance trade-off**.

- Let's recover it in the context of statistical learning theory.

# Empirical Risk Minimization

- Define a loss function  $\ell$  that defines proximity between the predicted value  $\hat{y} = f(x)$  and the true value  $y$ :  $\ell(f(x), y)$
- Usually, for continuous outcomes, the squared loss is used:  $\ell(f(x), y) = (f(x) - y)^2$
- We choose among a (finite) family of functions  $f \in \mathcal{F}$ , the best possible function  $f^*$  minimizes the **risk or expected loss**  $\mathcal{E}(f) = \mathbb{E}[(f(x) - y)^2]$ :

$$f^* = \operatorname{argmin}_{f \in \mathcal{F}} \mathbb{E}[(f(x) - y)^2]$$

## Empirical risk minimization: estimation error

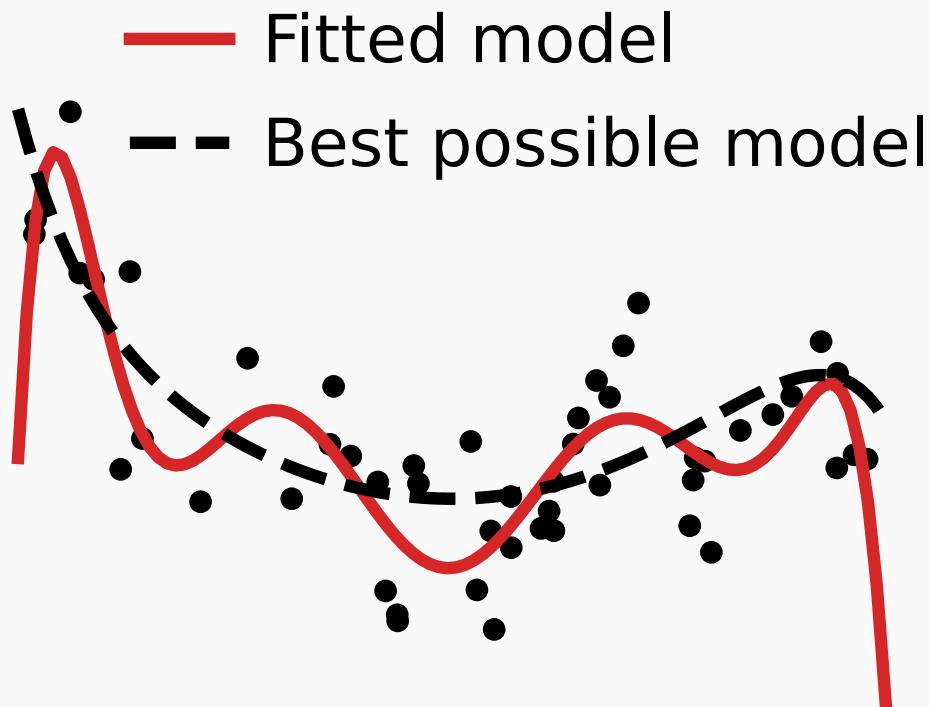
- In finite sample regimes, the expectation is not accessible since we only have access to a finite number of data pairs
- In practice, we minimize the **empirical risk** or average loss  $R_{\text{emp}} = \sum_{i=1}^n (f(x_i) - y_i)^2$ :

$$\hat{f} = \operatorname{argmin}_{f \in \mathcal{F}} \sum_{i=1}^n (f(x_i) - y_i)^2$$

- This creates the **estimation error**, related to sampling noise:

$$\mathcal{E}(\hat{f}) - \mathcal{E}(f^\star) = \mathbb{E}\left[(\hat{f}(x) - y)^2\right] - \mathbb{E}\left[(f^\star(x) - y)^2\right] \geq 0$$

## High **estimation error** means overfit



**Model is too complex**

- The model is able to recover the true generative process
- But its flexibility captures noise

**Too much noise**

**Not enough data**

# Bayes error rate: Randomness of the problem

- Interesting problems exhibit randomness

$$y = g(x) + e \text{ with } E(e|x) = 0 \text{ and } \text{Var}(e|x) = \sigma^2$$

- The best possible estimator is  $g(\cdot)$ , yielding the Bayes error, the unavoidable error:

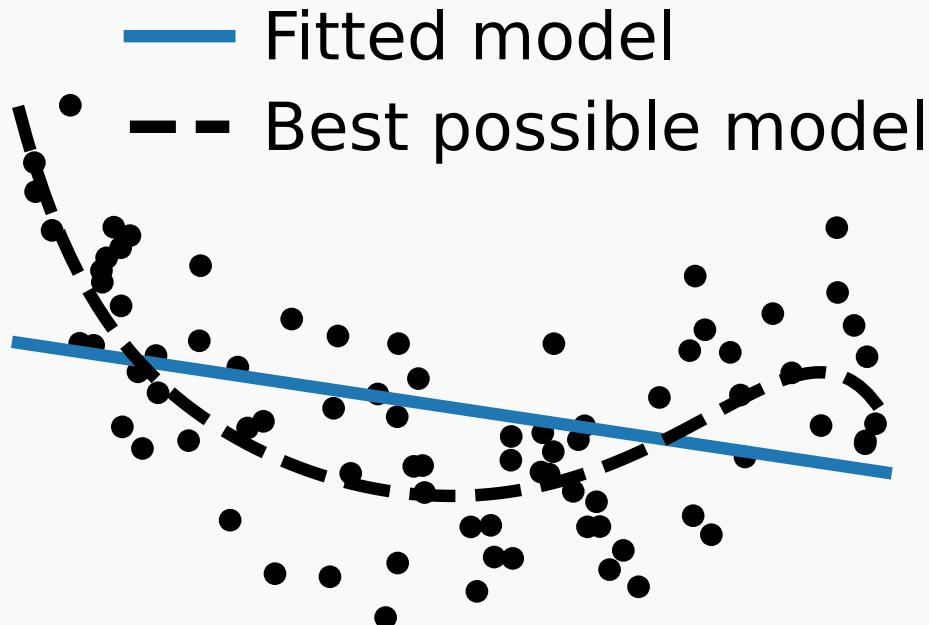
$$\mathcal{E}(g) = \mathbb{E}[(g(x) + e - g(x))^2] = \mathbb{E}[e^2]$$

# Empirical risk minimization: approximation error

- In practice you don't know the class of function in which the true function lies :  $y \approx g(x)$  : Every model is wrong !
- You are choosing the best possible function in the class of functions you have access to:  $f^* \in \mathcal{F}$  eg. linear models, polynomials, trees, ...
- This creates the **approximation error**:

$$\mathcal{E}(f(\star)) - \mathcal{E}(g) = \mathbb{E}\left[(f^*(x) - y)^2\right] - \mathbb{E}\left[(g(x) - y)^2\right] \geq 0$$

## High approximation error means underfit



**Model is too simple for the data**

- its best fit does not approximate the true generative process
- Yet it captures little noise

**Low noise**

**Rapidly enough data to fit the model**

# Bias variance trade-off: Putting the pieces together

**Decomposition of the empirical risk of a fitted model  $\hat{f}$**

$$\mathcal{E}(\hat{f}) = \underbrace{\mathcal{E}(g)}_{\text{Bayes error}} + \underbrace{\mathcal{E}(f^*) - \mathcal{E}(g)}_{\text{approximation error}} + \underbrace{\mathcal{E}(\hat{f}) - \mathcal{E}(f^*)}_{\text{estimation error}}$$

# Bias variance trade-off: Putting the pieces together

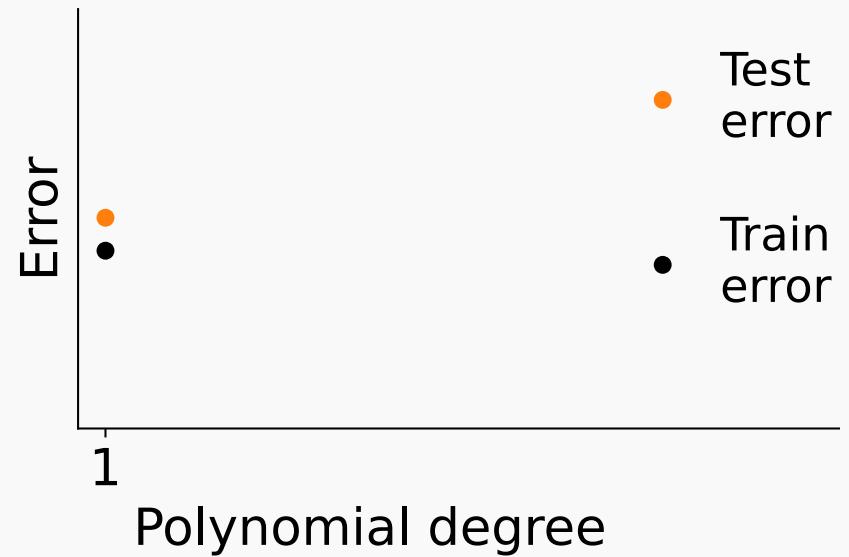
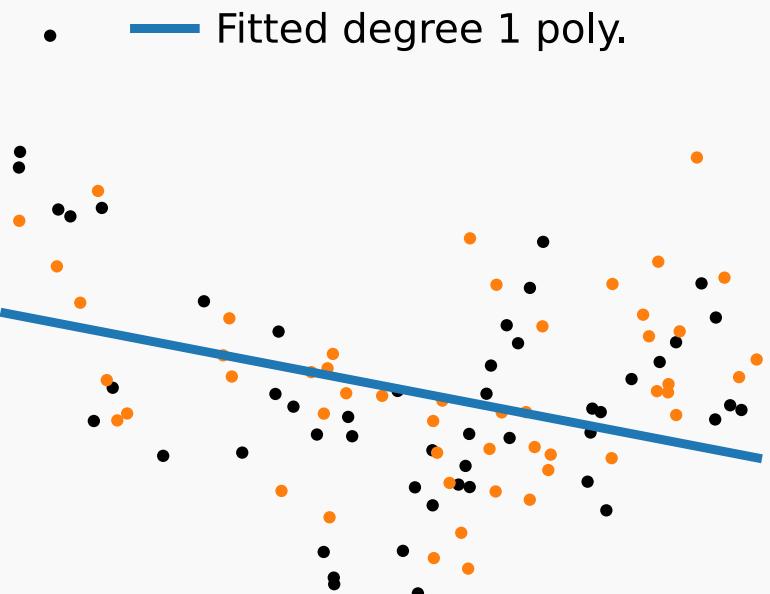
## Decomposition of the empirical risk of a fitted model $\hat{f}$

$$\mathcal{E}(\hat{f}) = \underbrace{\mathcal{E}(g)}_{\text{Bayes error}} + \underbrace{\mathcal{E}(f^*) - \mathcal{E}(g)}_{\text{approximation error}} + \underbrace{\mathcal{E}(\hat{f}) - \mathcal{E}(f^*)}_{\text{estimation error}}$$

## Controls on this trade-off

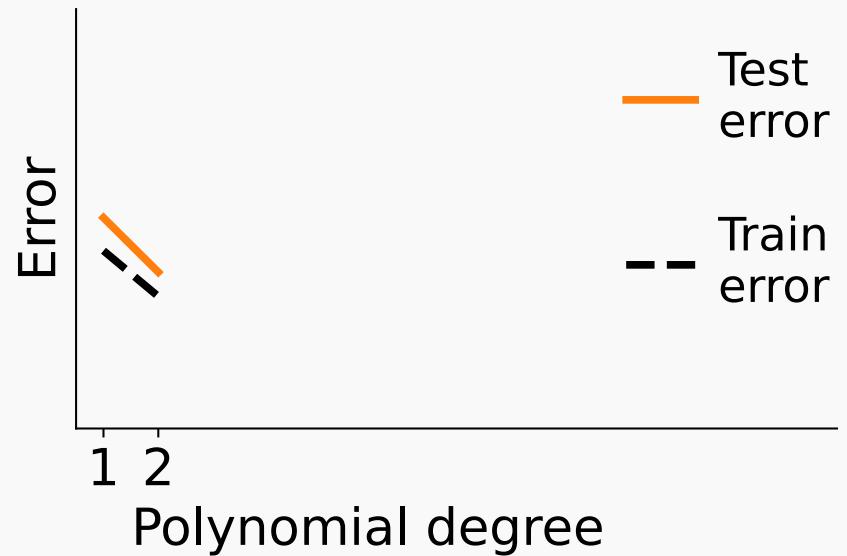
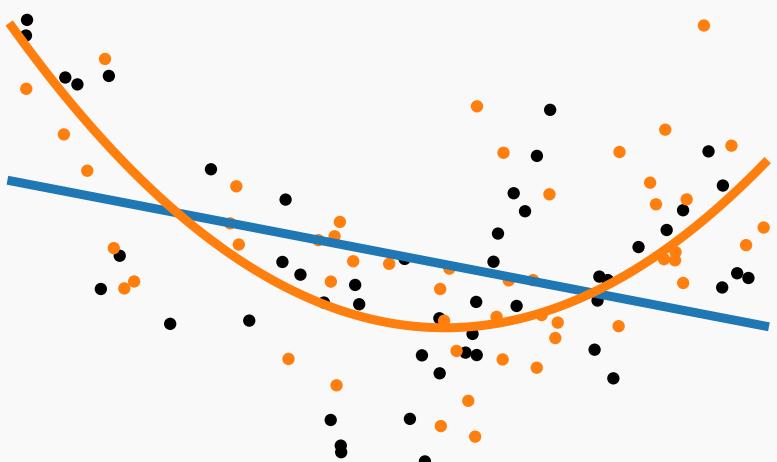
- Increase/decrease the size of the hypothesis family :  $\mathcal{F}$  ie. more or less complex models.
- Increase your sample size:  $n$  ie. more observations.

# Train vs test error: increasing complexity

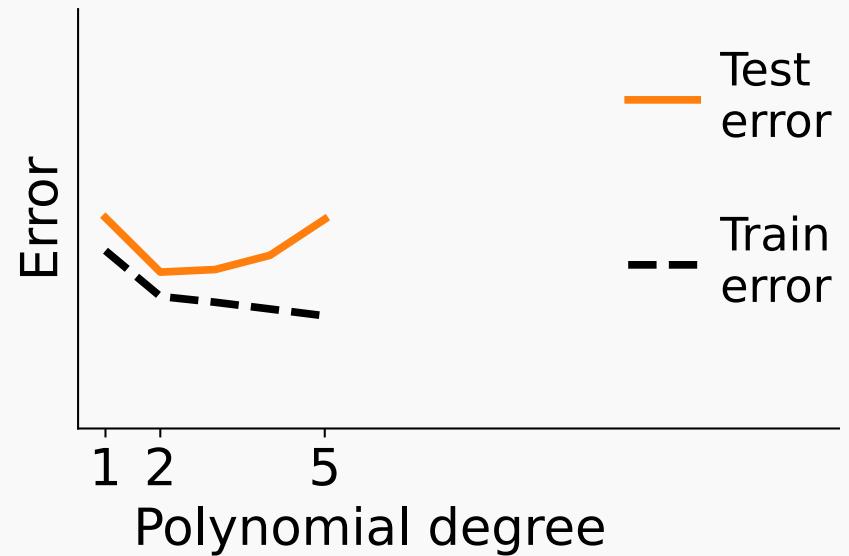
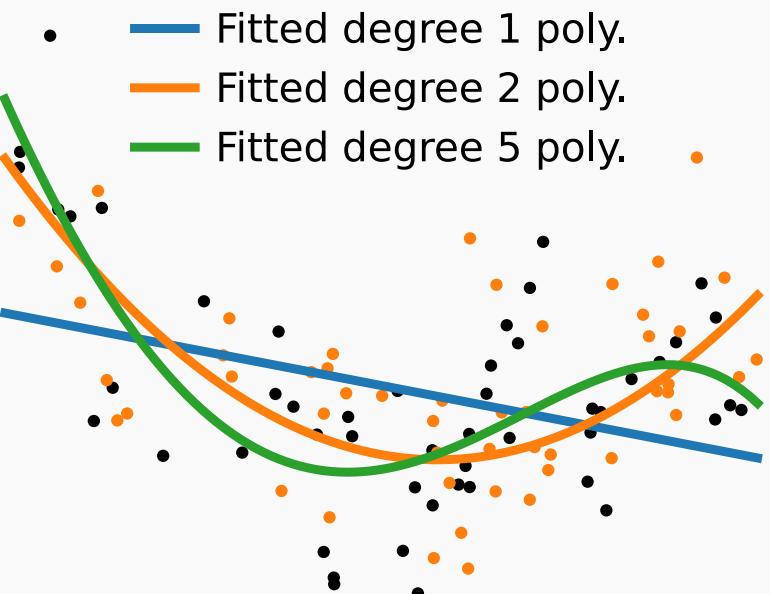


# Train vs test error: increasing complexity

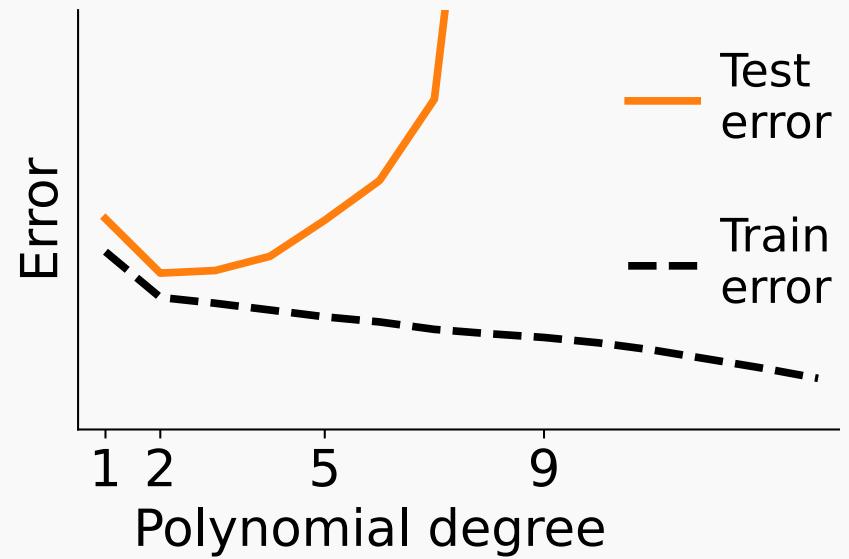
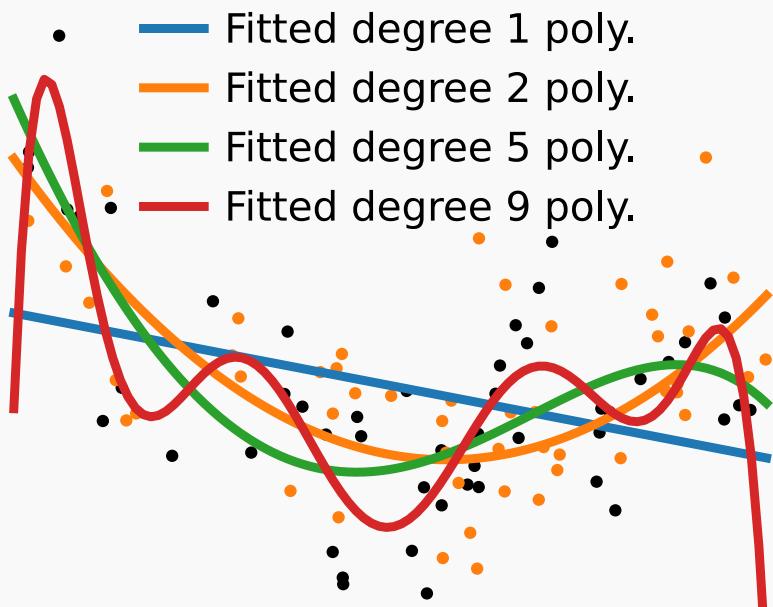
- Fitted degree 1 poly.
- Fitted degree 2 poly.



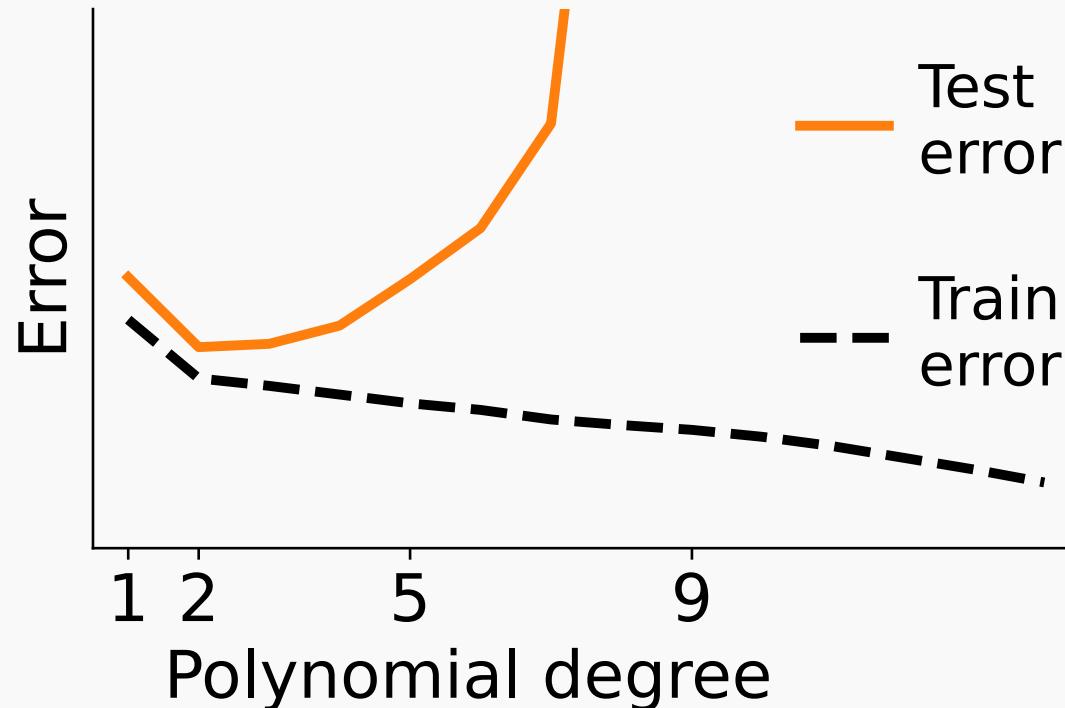
# Train vs test error: increasing complexity



# Train vs test error: increasing complexity

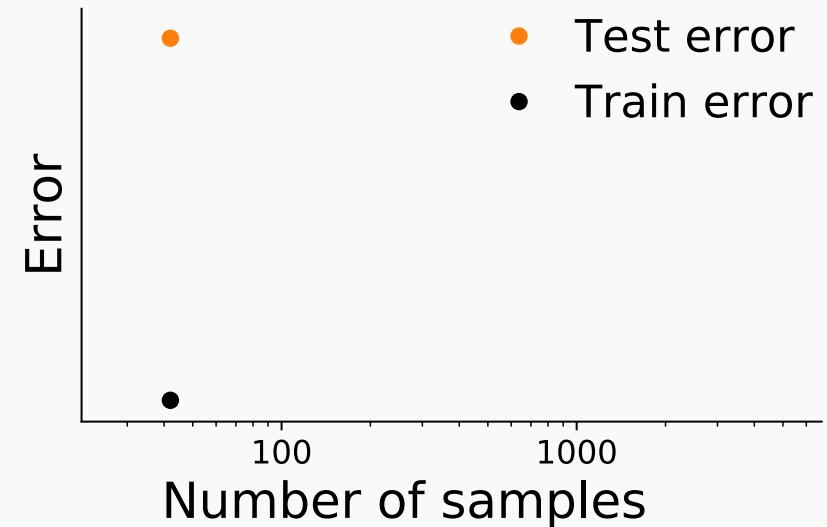
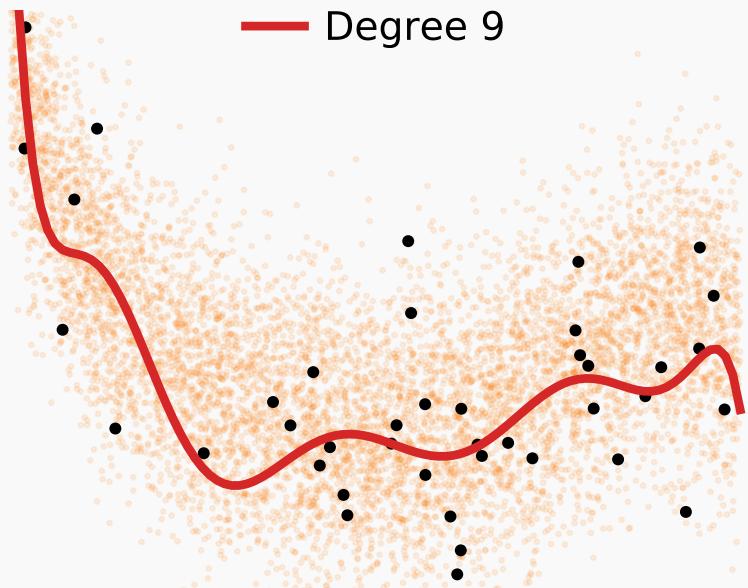


# Train vs test error: increasing complexity



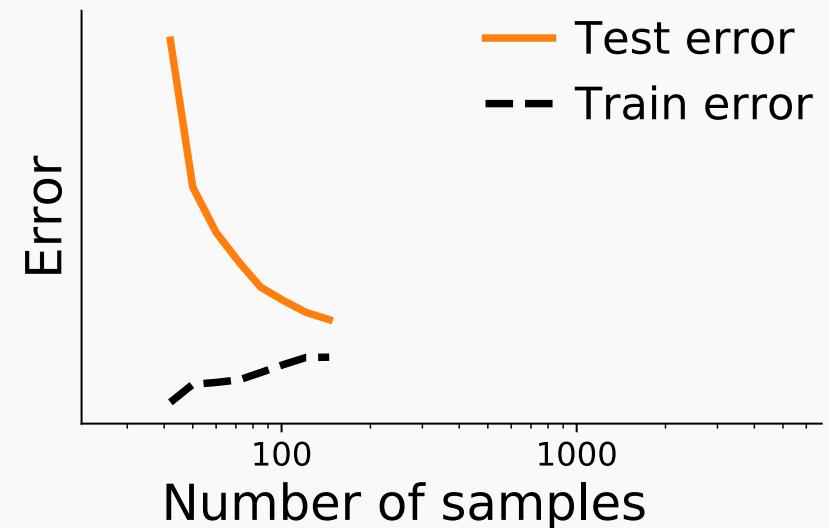
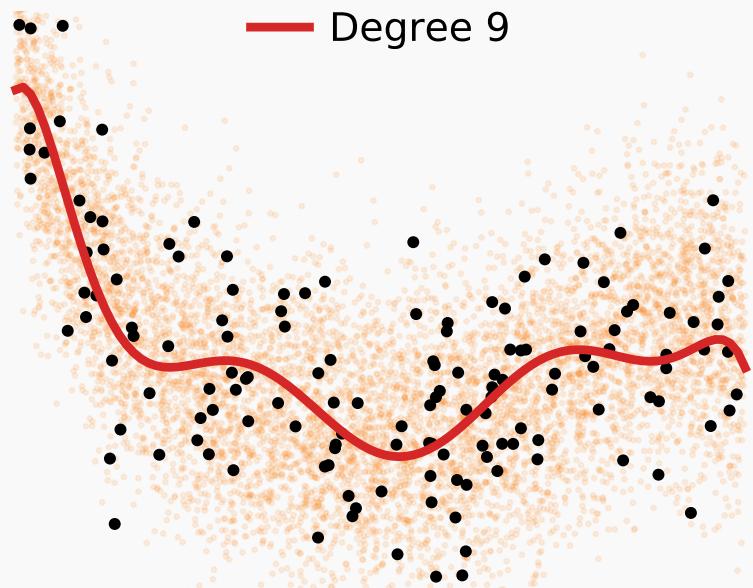
Underfit   Sweet Spot   Overfit

# Varying sample size

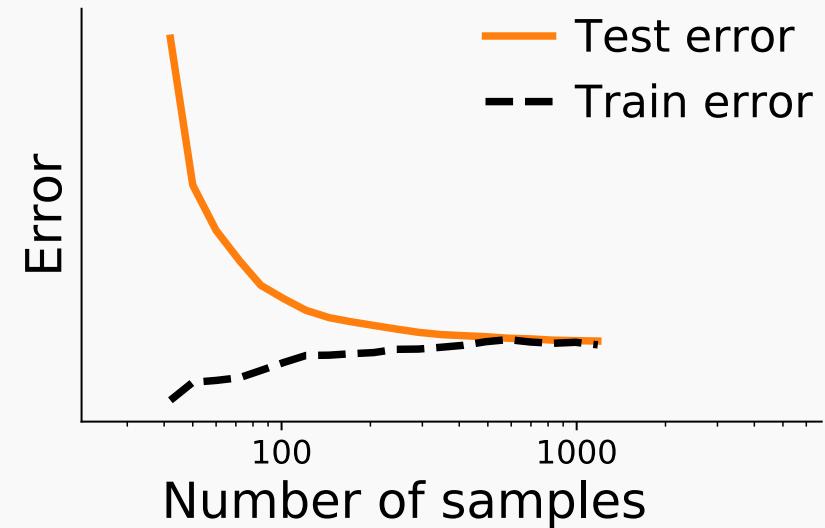
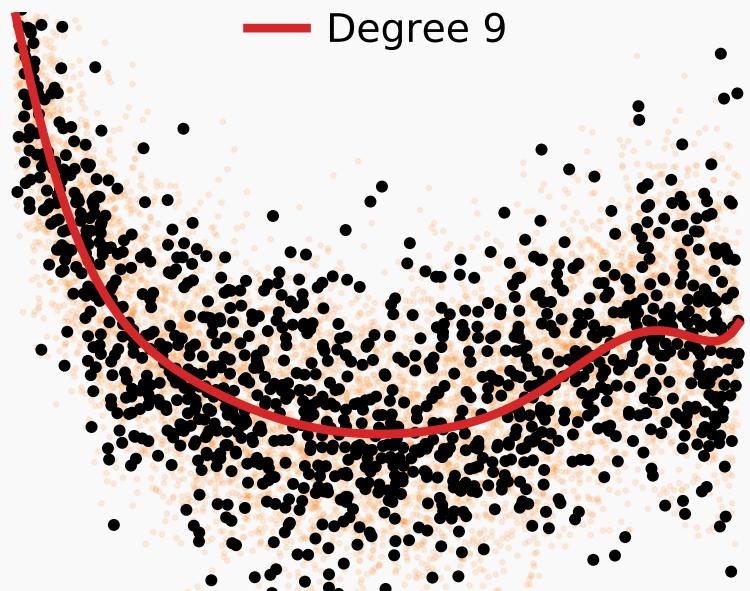


Overfit

# Varying sample size

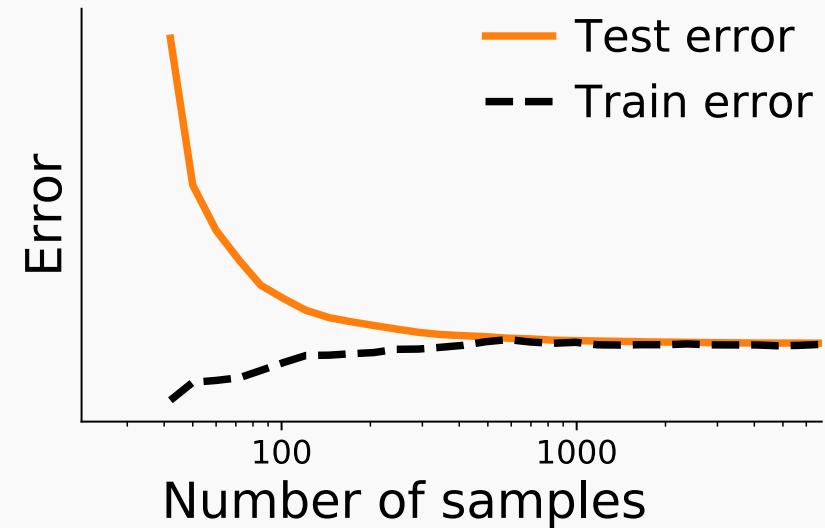
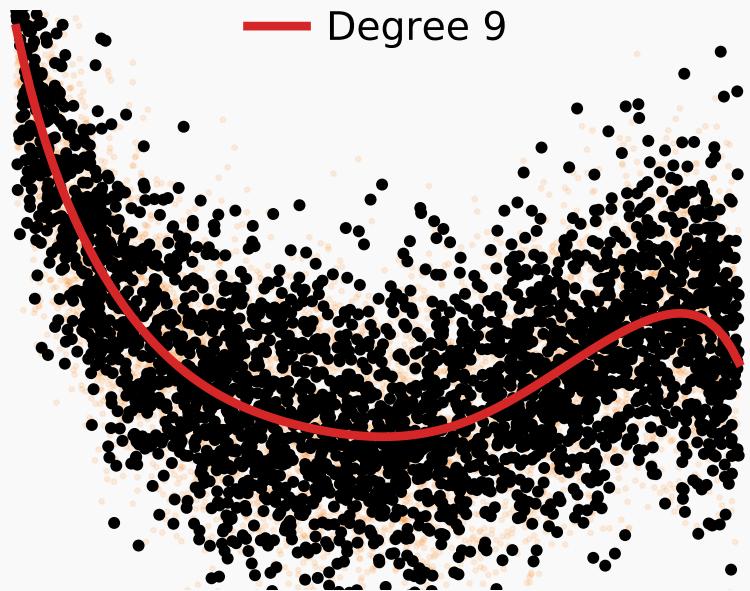


# Varying sample size



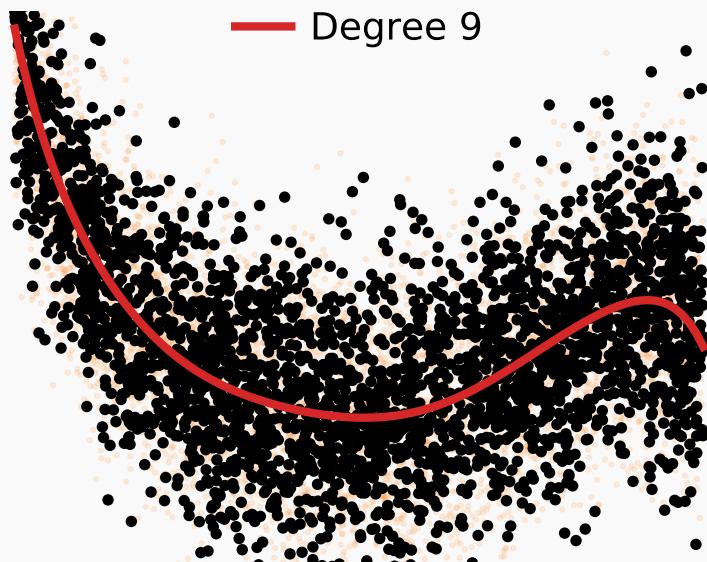
Sweet spot?

# Varying sample size



Diminishing returns?

# Varying sample size



The error of the best model trained on unlimited data.

Here, the data is generated by a polynomial of degree 9.

We cannot do better.

Prediction is limited by noise: Bayes error.

Remaining of this session (and the next)

## Common model families suited to tabular data

### Today

- Regularized linear models: Lasso and Ridge
- Hands-on with scikit-learn

### Next session

- Practical model selection: Cross-validation
- Flexible models: Trees, Random Forests, Gradient Boosting
- Practical scikit-learn

# Regularized linear models for predictive inference

---

## Reminder: Linear regression

$y$  is a linear combination of the features  $x \in \mathbb{R}^p$

$$Y_i = X_i^T \beta_0 + \varepsilon_i$$

- $\varepsilon$  the random variable of the error term.
- $\beta_0 \in \mathbb{R}^{p \times 1}$  the *true* coefficients.

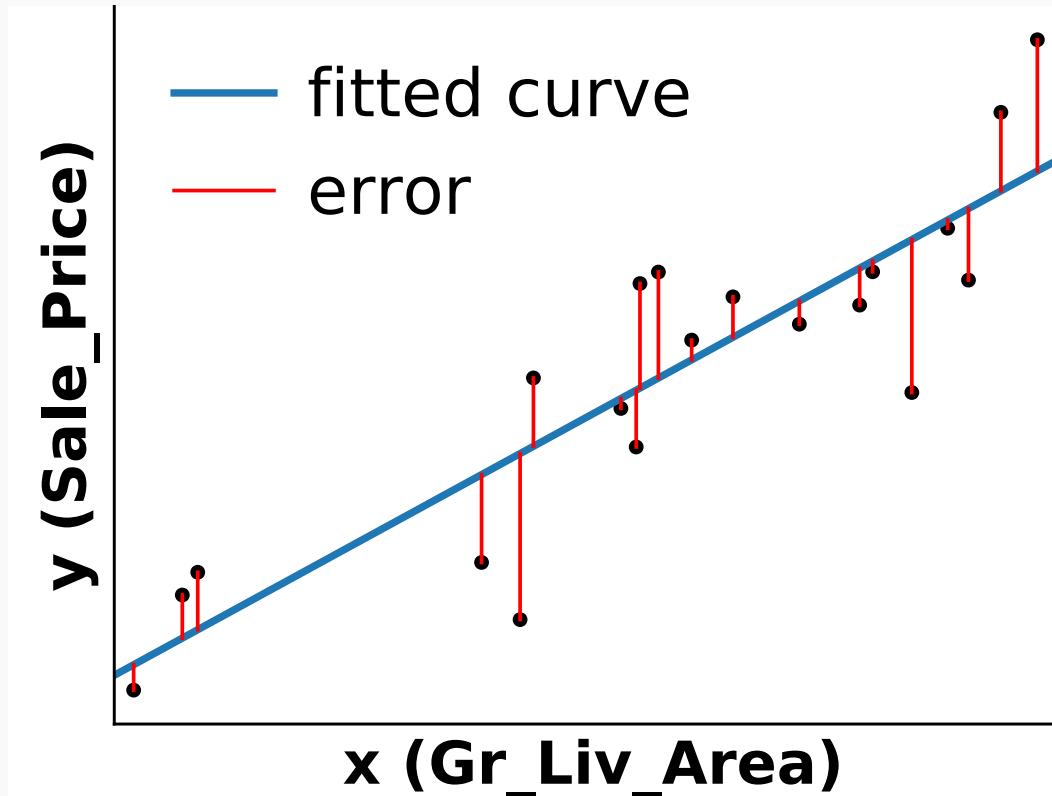
Usually, we assume that the errors are normally distributed and independent of  $X_i$ :

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ and } \varepsilon_i \perp\!\!\!\perp X_i$$

Model are typically fitted by linear algebra methods (Hastie, 2009).

## Reminder: Linear regression

$$Y_i = X_i^T \beta_0 + \varepsilon_i$$



# Reminder: Linear regression

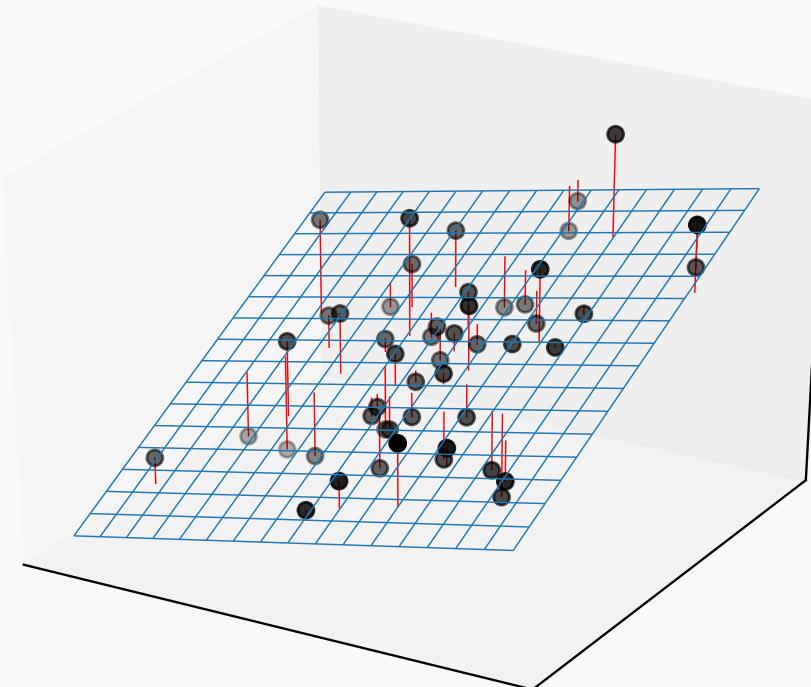
## Common metrics

- Mean Squared Error:  $MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$
- R-squared, :  $R^2 = 1 - \frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2}$  where  $\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i$

The proportion of variance explained by the model (perfect fit:  $R^2 = 1$ )

- Mean absolute error:  $MAE = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i|$

# Linear regression: Illustration in two dimensions



## Reminder: logistic regression for classification

The logit of the probability of the outcome is a linear combination of the features  $X_i \in \mathbb{R}^p$ :

$$\ln\left(\frac{p(Y_i=1|X_i)}{p(Y_i=0|X_i)}\right) = X_i^T \beta_0$$

## Reminder: logistic regression for classification

The logit of the probability of the outcome is a linear combination of the features  $X_i \in \mathbb{R}^p$ :

$$\ln\left(\frac{p(Y_i=1|X_i)}{p(Y_i=0|X_i)}\right) = X_i^T \beta_0$$

Taking exponential of both sides, we get:

$$p(Y_i = 1|X_i, \beta_0) \stackrel{\text{def}}{=} p(X_i, \beta_0) = \frac{1}{1 + \exp(-X_i^T \beta_0)}$$

The statistical model is a Bernoulli  :  $B(p(x, \beta_0))$

## Reminder: logistic regression for classification

The logit of the probability of the outcome is a linear combination of the features  $X_i \in \mathbb{R}^p$ :

$$\ln\left(\frac{p(Y_i=1|X_i)}{p(Y_i=0|X_i)}\right) = X_i^T \beta_0$$

Taking exponential of both sides, we get:

$$p(Y_i = 1|X_i, \beta_0) \stackrel{\text{def}}{=} p(X_i, \beta_0) = \frac{1}{1 + \exp(-X_i^T \beta_0)}$$

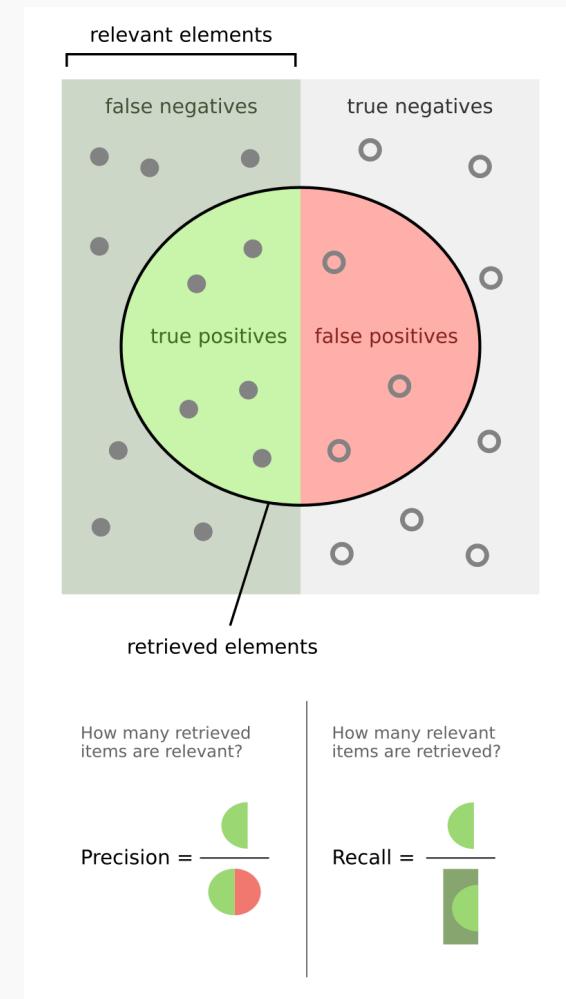
The statistical model is a Bernoulli  :  $B(p(x, \beta_0))$

Model fitted by maximum likelihood with iterative optimization (Hastie, 2009): eg. coordinate descents (liblinear), second order descent (Newton's method), gradient descent (SAG)...

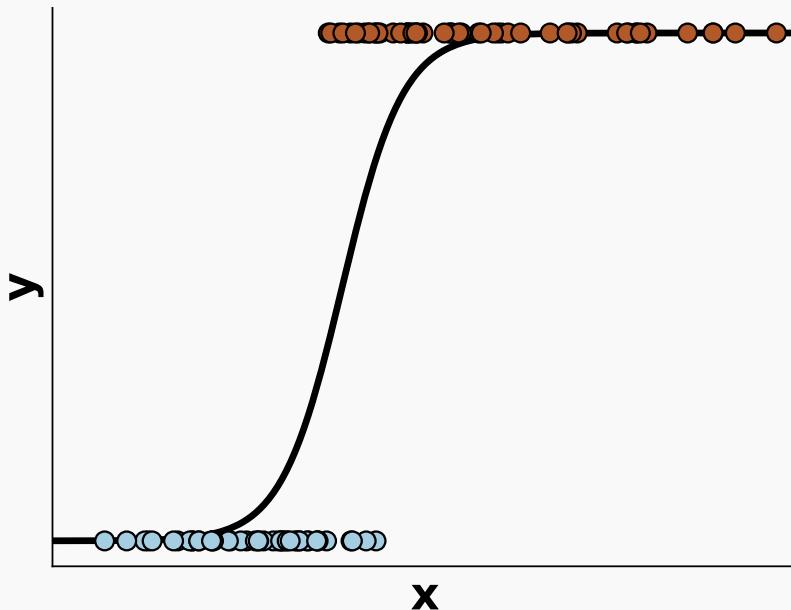
# Reminder: classification, logistic regression

## Common metrics

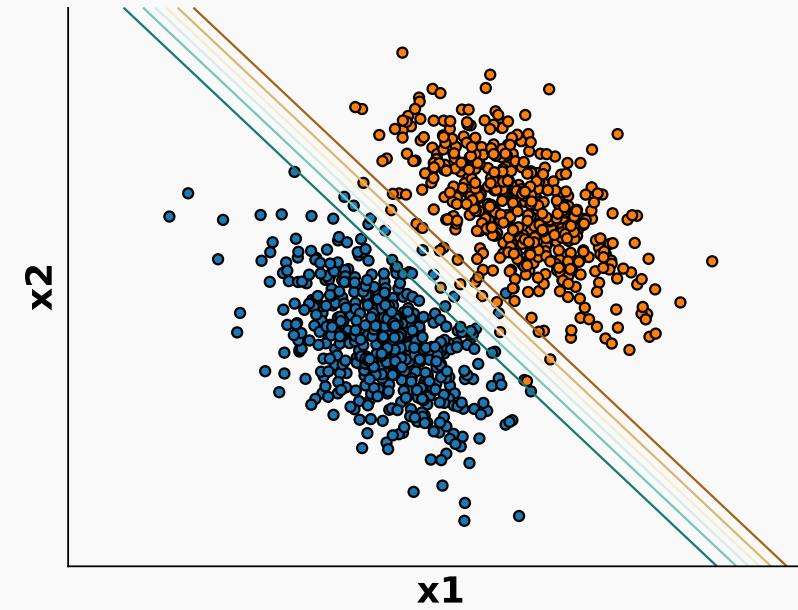
- Accuracy =  $\frac{1}{n} \sum_{i=1}^n \mathbb{1}(Y_i = \hat{Y}_i)$
- Precision: Precision =  $\frac{\text{TP}}{\text{TP} + \text{FP}}$
- Recall: Recall =  $\frac{\text{TP}}{\text{TP} + \text{FN}}$
- Brier score loss: BSL =  $\frac{1}{n} \sum_{i=1}^n (Y_i - p_i)^2$



# Logistic regression: Illustrations

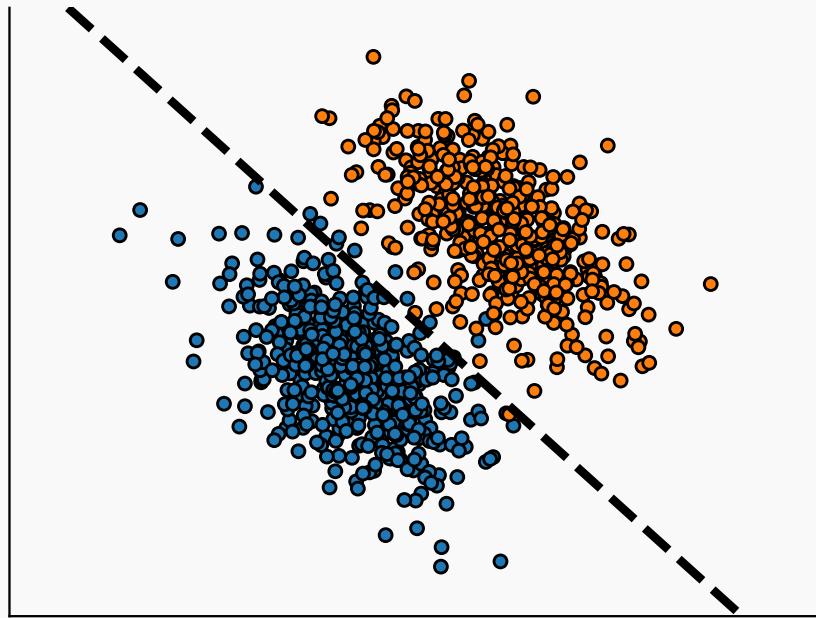


Logistic regression in one dimension.

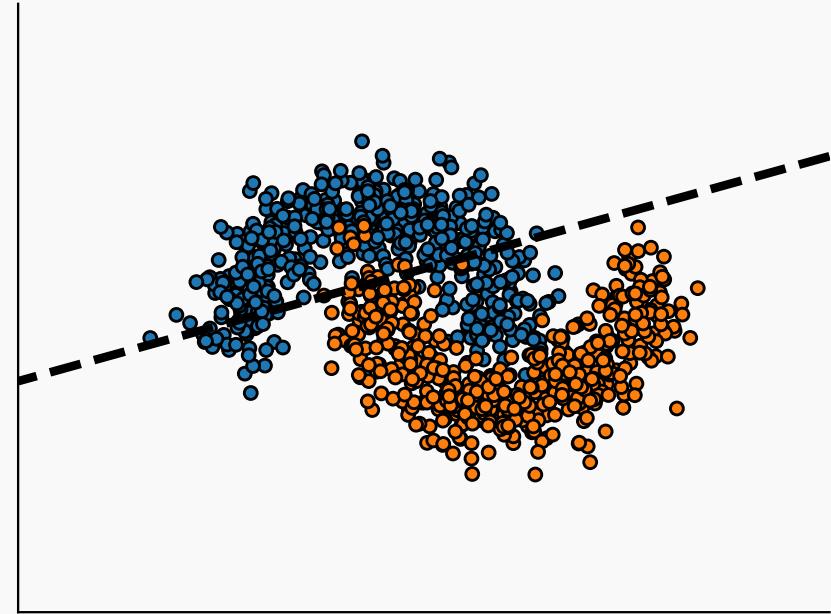


Logistic regression in two dimensions.

# Linear models are not suited to all data



Almost linearly separable data.



Data not linearly separable.

# Linear model pros and cons

## Pros

- Converge quickly
- Hard to beat when  $n_{\text{features}}$  is large but we still have  $n_{\text{samples}} \gg n_{\text{features}}$
- Linear models work well if
  - ▶ the classes are (almost) linearly separable
  - ▶ or the outcome is (almost) linearly related to the features.

# Linear model pros and cons

## Cons

Sometimes

- the best decision boundary to separate classes is not well approximated by a straight line.
- there are important non-linear relationships between the features and the outcome.

# Linear model pros and cons

## Cons

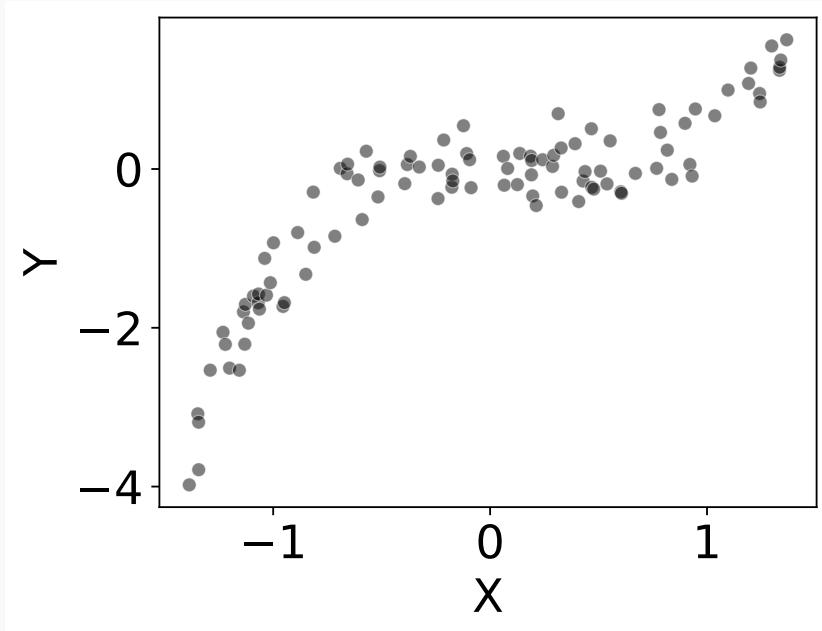
Sometimes

- the best decision boundary to separate classes is not well approximated by a straight line.
- there are important non-linear relationships between the features and the outcome.



Either use non-linear models, or perform transformations on the data, to engineer new features.

# Transformation of the features: Example

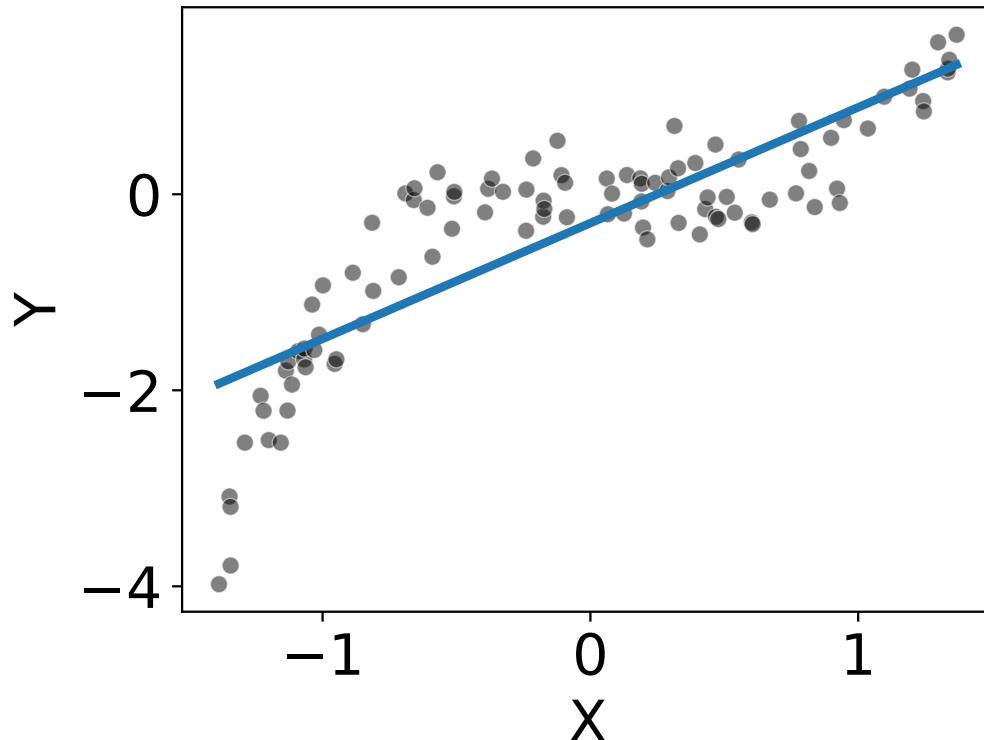


Non-linear relationship between the features and the outcome:

$$Y = X^3 - 0.5 \times X^2 + \varepsilon$$

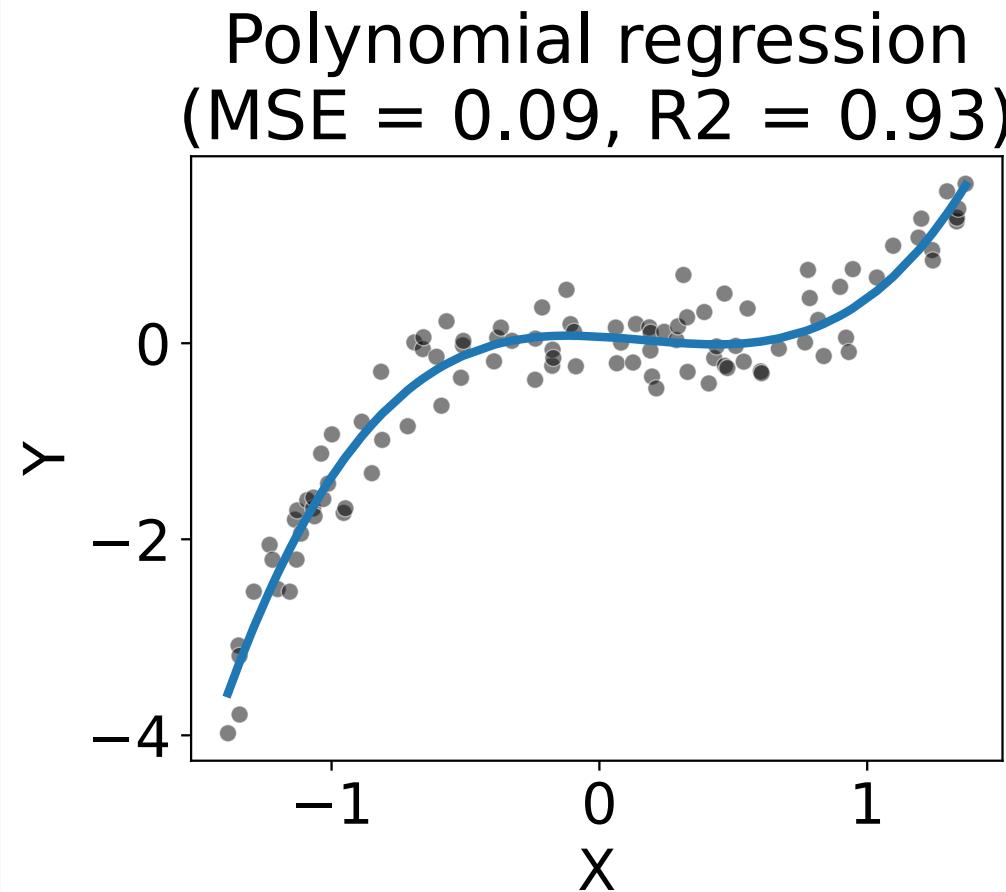
# Transformation of the features: Example

Simple linear regression  
(MSE = 0.36, R<sup>2</sup> = 0.71)



Vanilla Linear regression fails to capture the relationship.

# Transformation of the features: Example



Solution:

- Expand the feature space with polynomials of the features:

$$X = [X, X^2, X^3]$$

- Run a linear regression on the new feature space.

$$Y = [X, X^2, X^3]^T \hat{\beta}$$

## **Feature expansion increase the family of models**

- Linear model can underfeat : when n\_features small or the problem is not linearly separable.
- Feature expansion is an easy way to capture non-linear relationships.
- Different feature expansions exists: polynomial, log, splines, embeddings, kernels, ...

# Takeaway on feature expansion

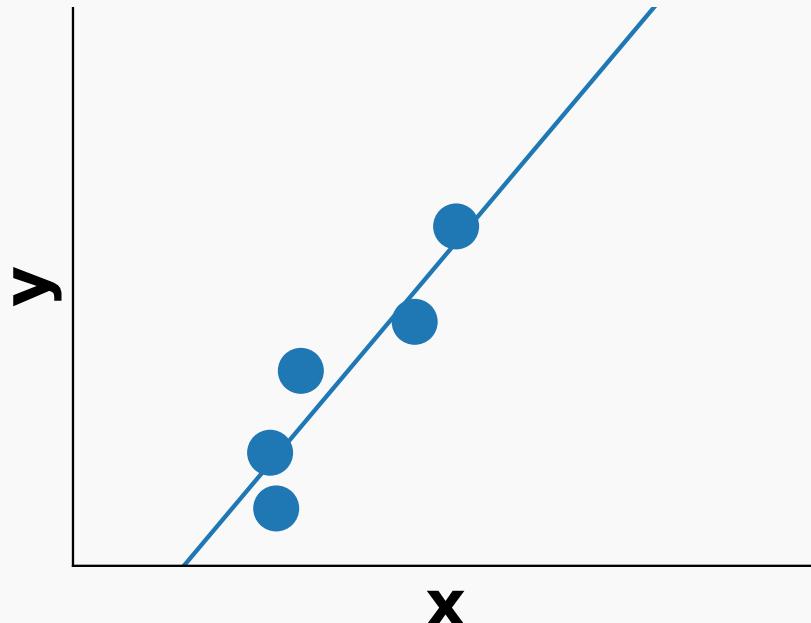
## **Feature expansion increase the family of models**

- Linear model can underfit : when n\_features small or the problem is not linearly separable.
- Feature expansion is an easy way to capture non-linear relationships.
- Different feature expansions exists: polynomial, log, splines, embeddings, kernels, ...

## **But...**

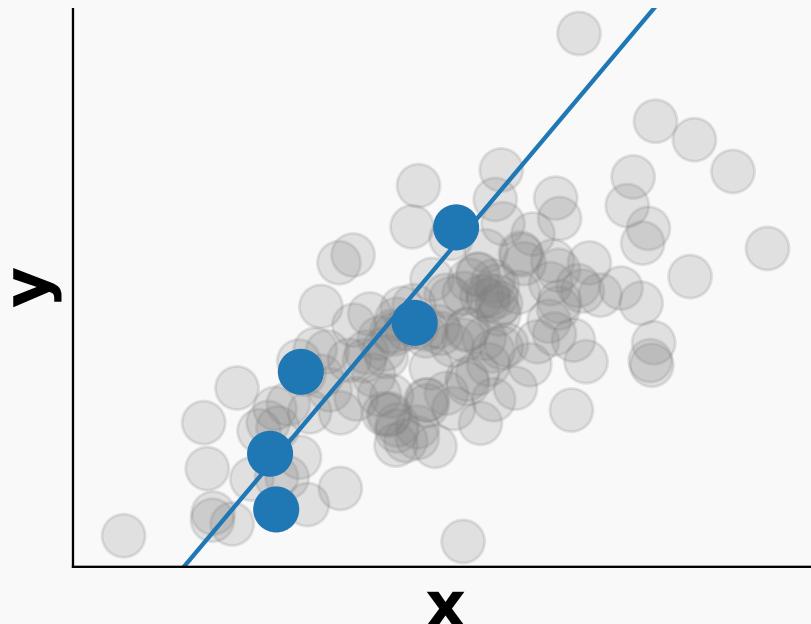
- Linear models can also overfit !
- When:
  - n\_features is large
  - Many uninformative features

## Many features, few observations: illustration in 1D



- Few observations with respect to the number of features.
- Fit a linear model without regularization.
-

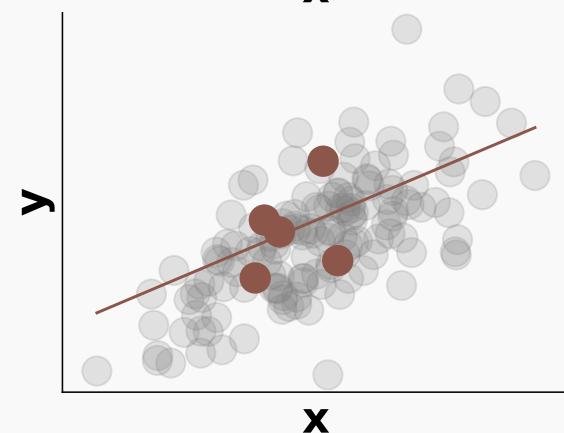
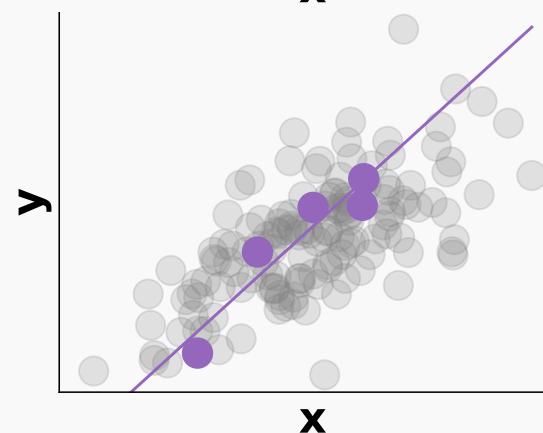
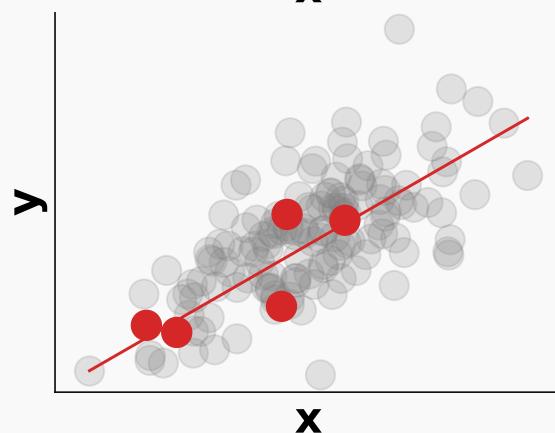
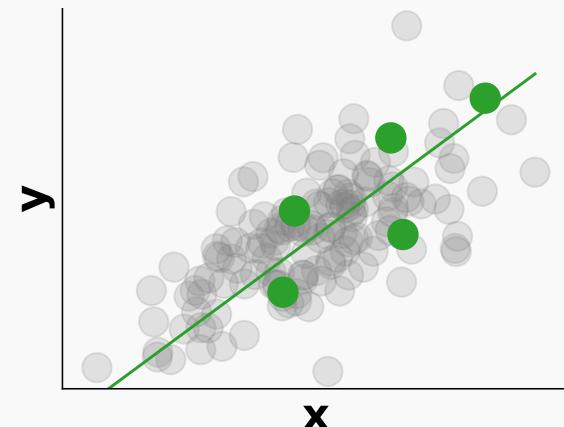
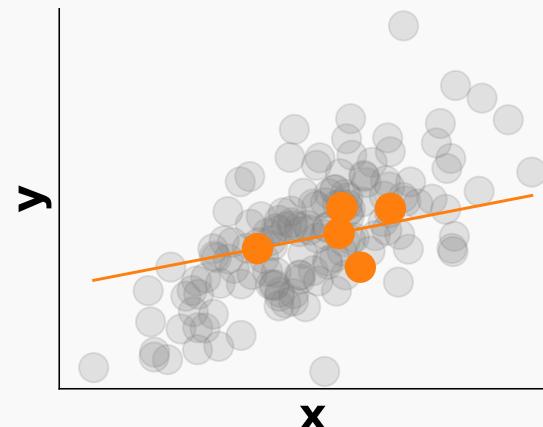
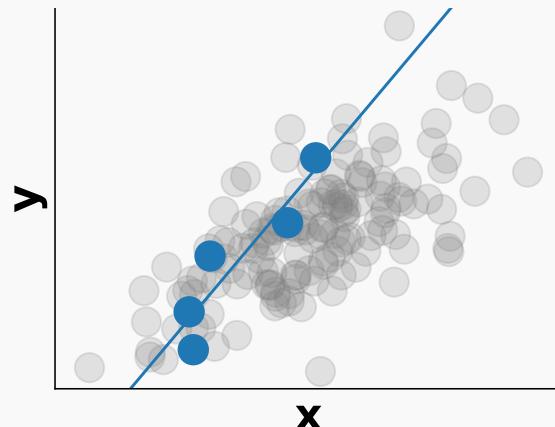
## Many features, few observations: illustration in 1D



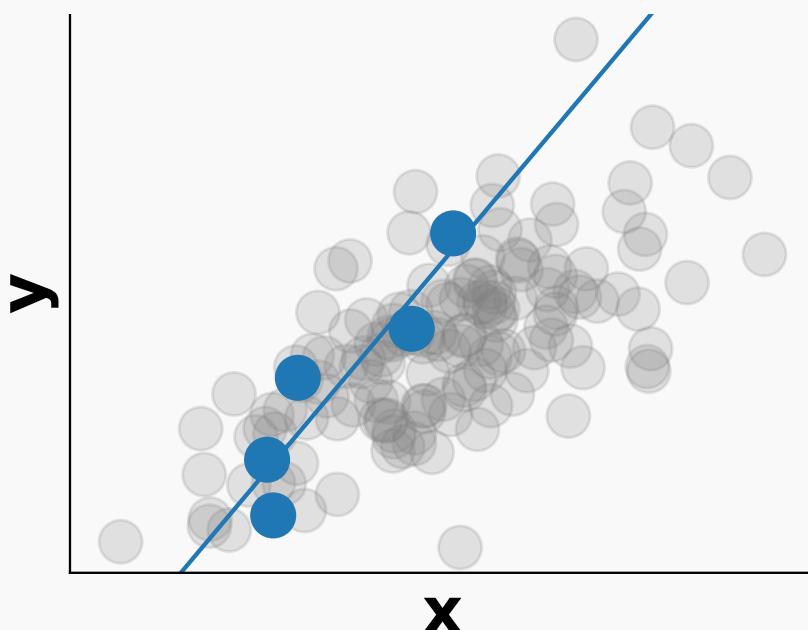
- Few observations with respect to the number of features.
- Fit a linear model without regularization.
- Linear model can overfit if data is noisy.

Many features, few observations: illustration in 1D

## Sampling different training sets

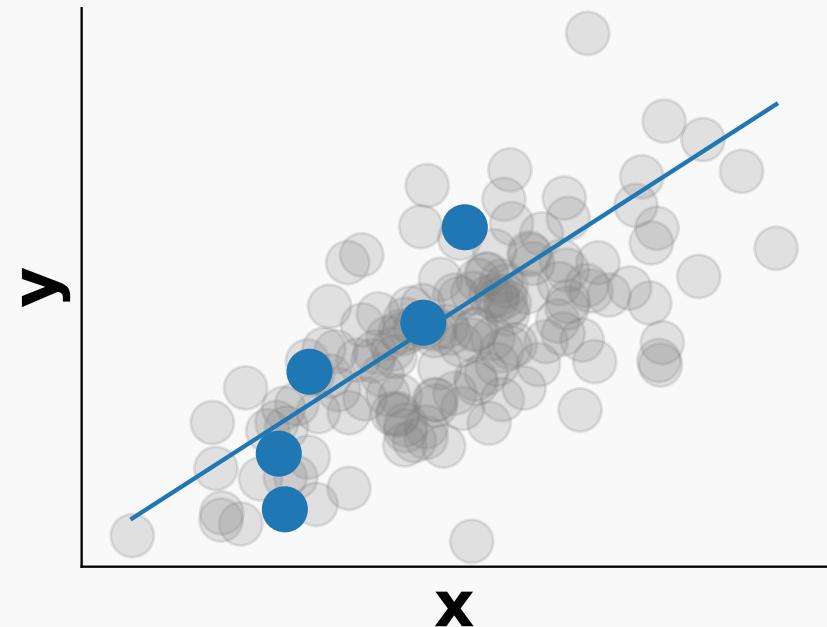


# Bias variance trade-off with Lasso



Linear regression (no regularization)

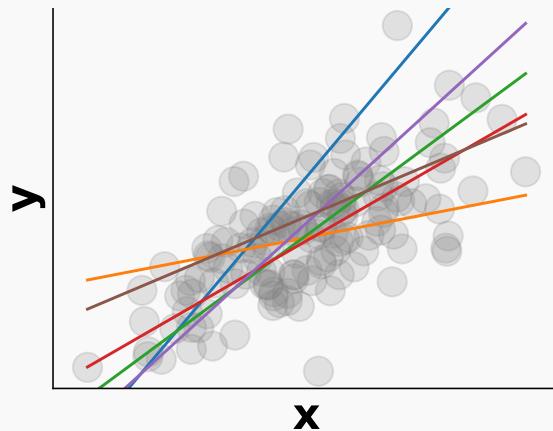
High variance, no bias.



Lasso (regularization): Shrink some coefficients of  $\beta$ .

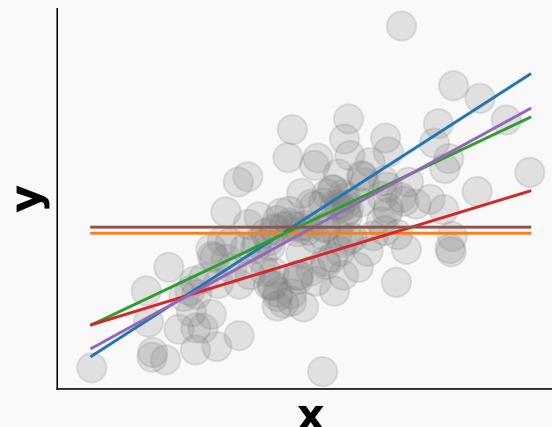
Lower variance, but bias.

# Bias variance trade-off with Lasso

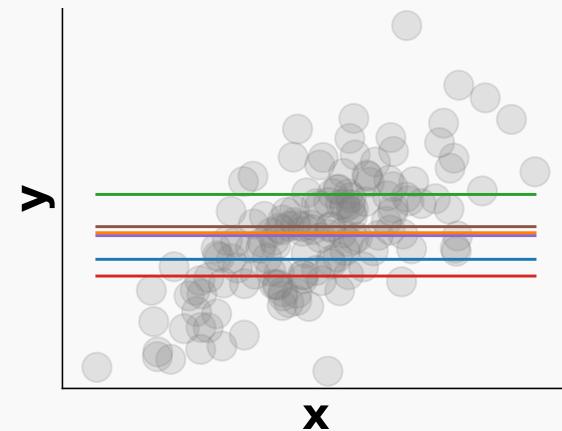


Too much variance

Not enough regularization



Best trade-off



Too much bias

Too much regularization

## Objective function of the Lasso

The lasso puts a constraint of amplitude  $t$  on the  $L_1$  norm of the coefficients:

$$\min_{\beta} \sum_i^n \left( (y_i - \beta^T x_i)^2 \right) \text{ st. } \sum_1^p |\beta_j| \leq t$$

## Objective function of the Lasso

The lasso puts a constraint of amplitude  $t$  on the  $L_1$  norm of the coefficients:

$$\min_{\beta} \sum_i^n \left( (y_i - \beta^T x_i)^2 \right) \text{ st. } \sum_1^p |\beta_j| \leq t$$

This is equivalent to the following optimization problem (using lagrangian multiplier):

$$\min_{\beta} \sum \left( (y_i - \beta^T x_i)^2 \right) + \alpha \sum (|\beta_j|)$$

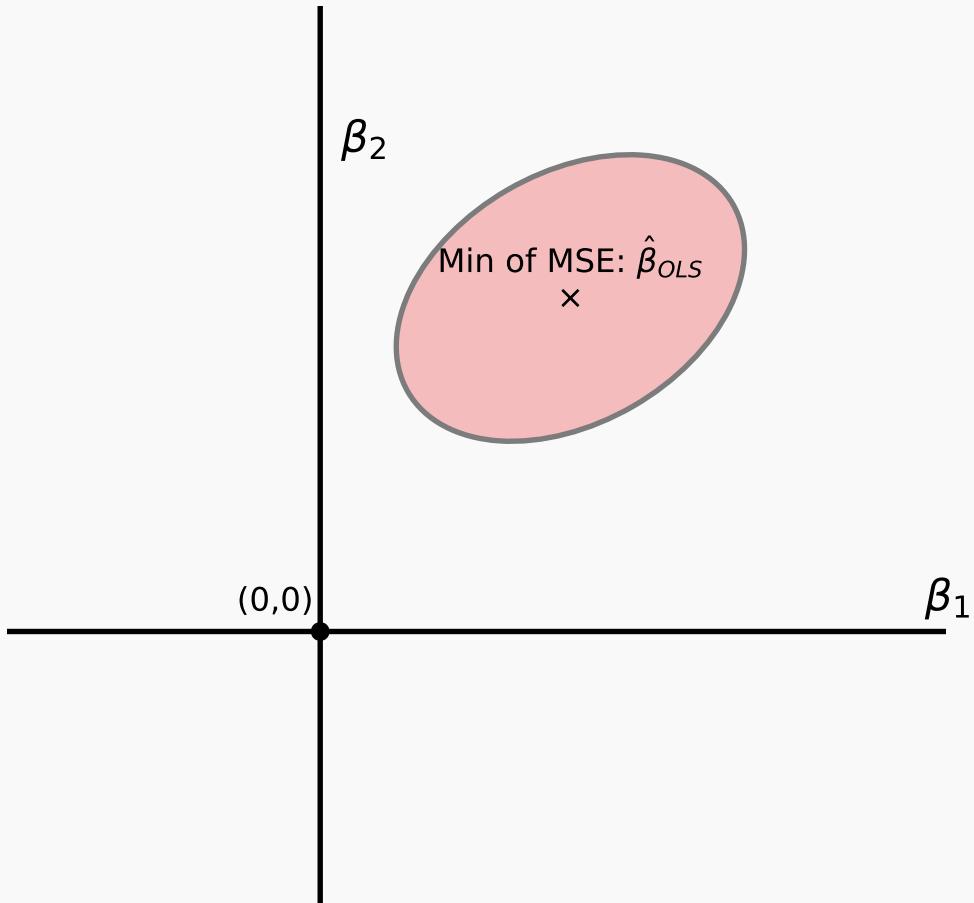
## Objective function of the Lasso

The lasso puts a constraint of amplitude  $t$  on the  $L_1$  norm of the coefficients:

$$\min_{\beta} \sum_i^n \left( (y_i - \beta^T x_i)^2 \right) \text{ st. } \sum_1^p |\beta_j| \leq t$$

This penalty discourages large weights and can shrink certain weights to exactly *zero* (not clear yet why).

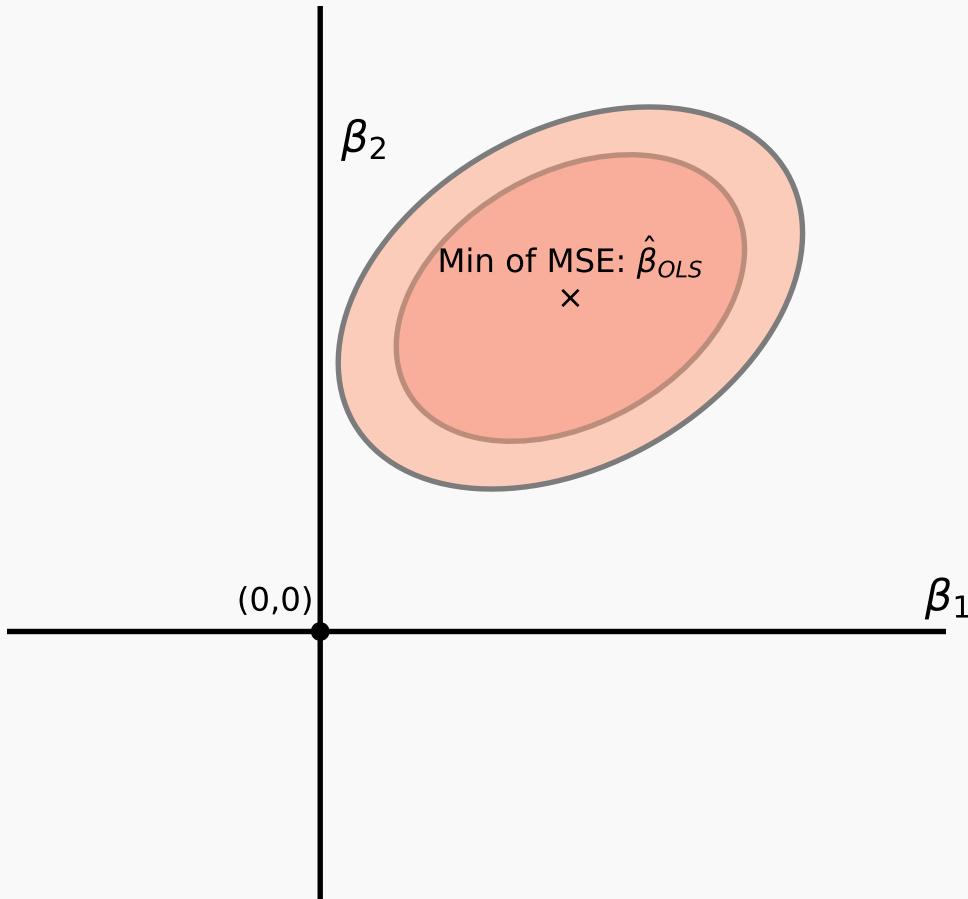
# Why does Lasso shrink some coefficients to zero?



- Plot the MSE of the model as a function of the coefficients.

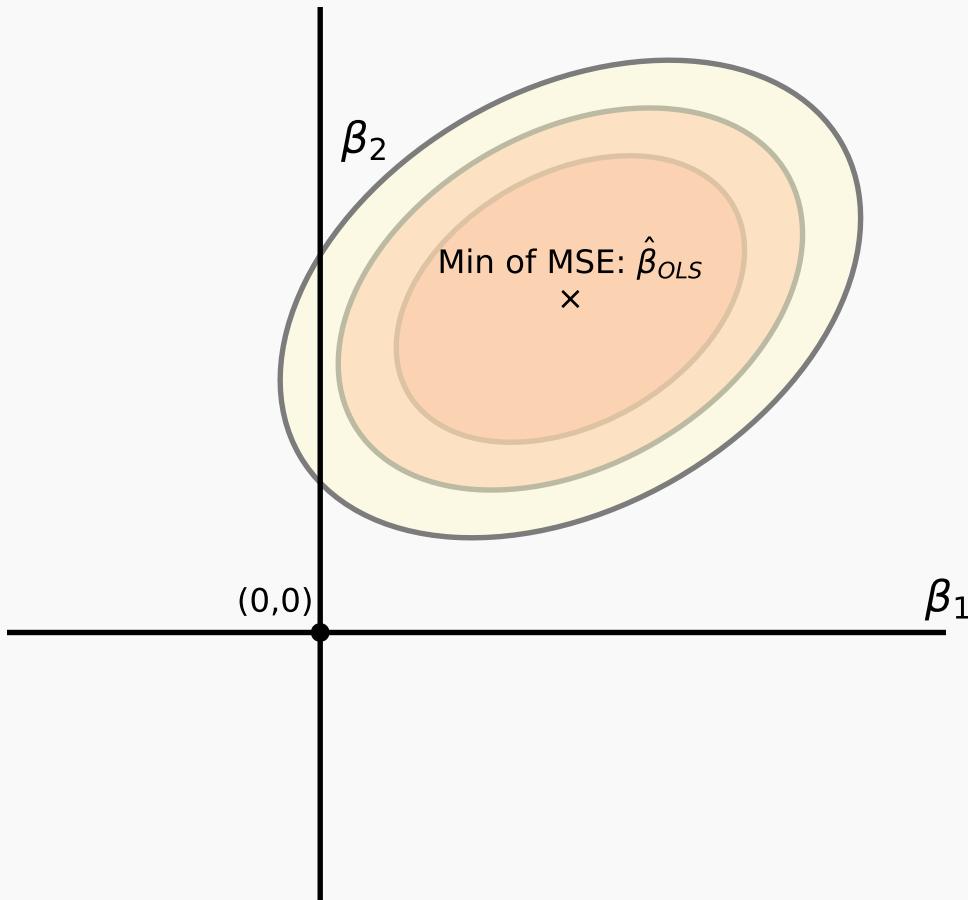
- 
-

# Why does Lasso shrink some coefficients to zero?



- Plot the MSE of the model as a function of the coefficients.
- The MSE surface is an ellipsoid in  $\beta$ .
-

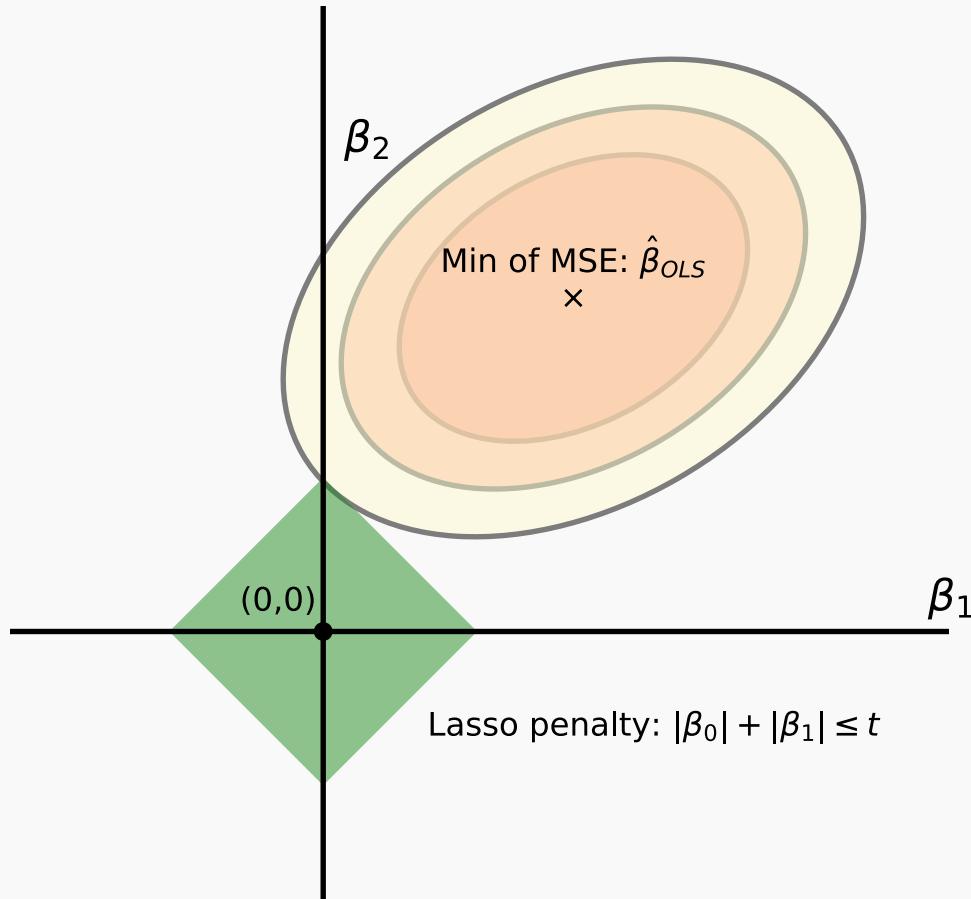
# Why does Lasso shrink some coefficients to zero?



- Plot the MSE of the model as a function of the coefficients.

- 
-

# Why does Lasso shrink some coefficients to zero?



- Plot the MSE of the model as a function of the coefficients.
- 
- The lasso objective function is a diamond.

## Another regularized linear model: Ridge

Ridge puts a constraint of amplitude  $t$  on the  $L_2$  norm of the coefficients:

$$\min_{\beta} \sum_i^n \left( (y_i - \beta^T x_i)^2 \right) \text{ st. } \sum_1^p \beta_j^2 \leq t$$

## Another regularized linear model: Ridge

Ridge puts a constraint of amplitude  $t$  on the  $L_2$  norm of the coefficients:

$$\min_{\beta} \sum_i^n \left( (y_i - \beta^T x_i)^2 \right) \text{ st. } \sum_1^p \beta_j^2 \leq t$$

This is equivalent to the following optimization problem (using lagrangian multiplier):

$$\min_{\beta} \sum \left( (y_i - \beta^T x_i)^2 \right) + \alpha \sum (\beta_j^2)$$

## Another regularized linear model: Ridge

Ridge puts a constraint of amplitude  $t$  on the  $L_2$  norm of the coefficients:

$$\min_{\beta} \sum_i^n \left( (y_i - \beta^T x_i)^2 \right) \text{ st. } \sum_1^p \beta_j^2 \leq t$$

This is equivalent to the following optimization problem (using lagrangian multiplier):

$$\min_{\beta} \sum \left( (y_i - \beta^T x_i)^2 \right) + \alpha \sum (\beta_j^2)$$

This penalty shrinks the coefficients towards zero and each other.

# Importance of rescaling

## Why rescale?

- The penalty term in the Lasso and Ridge is sensitive to the scale of the features.
- If the features are not on the same scale, the regularization will not be applied uniformly.
- The coefficients of the model will be biased towards the features with the largest scale.

## How to rescale?

- Gaussian hypothesis? Standard scaling:  $X = \frac{X - \text{mean}(X)}{\text{std}(X)}$
- Non Gaussian? MinMax scaling:  $X = \frac{X - \min(X)}{\max(X) - \min(X)}$

# Importance of rescaling: illustration

## Many features, few observations

Assumption 1: Linear model with high dimension

$$Y = X\beta_0 + \varepsilon, \quad \varepsilon \perp\!\!\!\perp X \text{ and } X \in \mathbb{R}^{n \times p} \text{ with } n \ll p$$

Assumption 2: (approximate) sparsity

- The true  $\beta_0$  is sparse: ie. many coefficients are zero or very close to zero.

# Regularized linear models for classification

## Log likelihood for Lasso classification



Log likelihood for vanilla logistic regression

$$p_{i,\beta} = \mathbb{P}[Y_i | X_i, \beta] = \frac{1}{1 + \exp(-\beta^T X_i)}$$

$$L(\beta) = \frac{1}{N} \sum_{i=1}^N \log \left[ p_{i,\beta}^{y_i} (1 - p_{i,\beta})^{1-y_i} \right]$$

$$= \frac{1}{N} \sum_{i=1}^N [(y_i \beta^T X_i) - \log(1 + \exp(\beta^T X_i))]$$

# Regularized linear models for classification

## Log likelihood for Lasso classification

$$L(\beta) = \frac{1}{N} \sum_{i=1}^N \left[ (y_i \beta^T X_i - \log[1 + \exp(\beta^T X_i)]) - \lambda \sum_{j=1}^p |\beta_j| \right]$$



Log likelihood for vanilla logistic regression

$$p_{i,\beta} = \mathbb{P}[Y_i | X_i, \beta] = \frac{1}{1 + \exp(-\beta^T X_i)}$$

$$L(\beta) = \frac{1}{N} \sum_{i=1}^N \log \left[ p_{i,\beta}^{y_i} (1 - p_{i,\beta})^{1-y_i} \right]$$

$$= \frac{1}{N} \sum_{i=1}^N [(y_i \beta^T X_i - \log[1 + \exp(\beta^T X_i)])]$$

# Regularized linear models for classification

## Log likelihood for Lasso classification

### Log likelihood for Lasso classification

$$L(\beta) = \frac{1}{N} \sum_{i=1}^N \left[ (y_i \beta^T X_i - \log[1 + \exp(\beta^T X_i)]) - \lambda \sum_{j=1}^p |\beta_j| \right]$$



Log likelihood for vanilla logistic regression

$$p_{i,\beta} = \mathbb{P}[Y_i | X_i, \beta] = \frac{1}{1 + \exp(-\beta^T X_i)}$$

$$L(\beta) = \frac{1}{N} \sum_{i=1}^N \log \left[ p_{i,\beta}^{y_i} (1 - p_{i,\beta})^{1-y_i} \right]$$

$$= \frac{1}{N} \sum_{i=1}^N [(y_i \beta^T X_i - \log[1 + \exp(\beta^T X_i)])]$$

Post lasso

TODO

# How to choose lambda?

## Theoretical guarantees

- See (Chernozhukov et al., 2024, Chap 3.2)

# How to choose lambda?

## Theoretical guarantees

- See (Chernozhukov et al., 2024, Chap 3.2)  but assumptions are hard to check.

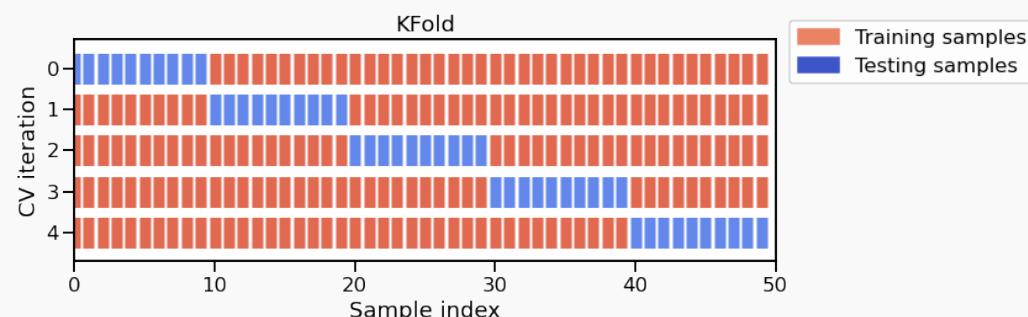
# How to choose lambda?

## Theoretical guarantees

- See (Chernozhukov et al., 2024, Chap 3.2)

## In practice

Use cross-validation: repeated train/test splits.



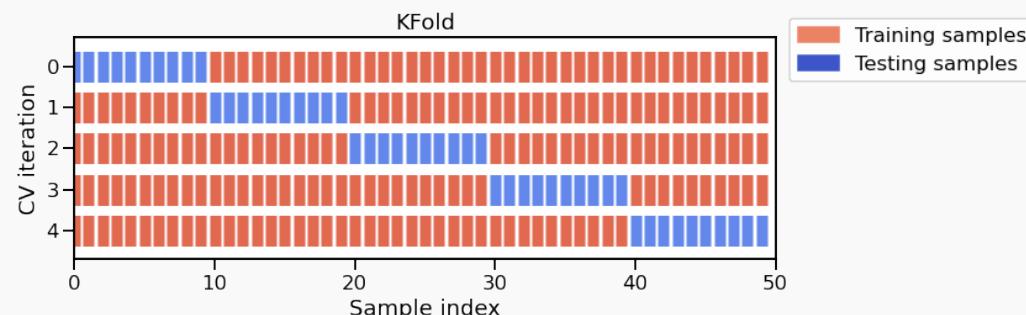
# How to choose lambda?

## Theoretical guarantees

- See (Chernozhukov et al., 2024, Chap 3.2)

## In practice

Use cross-validation: repeated train/test splits.



We will look into that in more details in the next session.

# Short introduction to scikit-learn

- url: <https://github.com/strayMat/causal-ml-course/tree/main/notebooks>

# Python hands-on: Common pitfalls in the interpretation of coefficients of linear models

---

To your notebooks !



- url: <https://github.com/strayMat/causal-ml-course/tree/main/notebooks>

# Take home messages: Bias-variance trade-off

## **High bias == underfitting**

- systematic prediction errors
- the model prefers to ignore some aspects of the data
- misspecified models

## **High variance == overfitting:**

- prediction errors without obvious structure
- small change in the training set, large change in model
- unstable models

# Take home messages: Lasso and Ridge

## Lasso

- L1 penalty: sparsity
- Feature selection
- Unstable for correlated features

## Ridge

- L2 penalty: shrinkage
- No feature selection
- Stable for correlated features

# Bibliography

- Chernozhukov, V., Hansen, C., Kallus, N., Spindler, M., & Syrgkanis, V. (2024). *Applied causal inference powered by ML and AI*. Arxiv Preprint Arxiv:2403.02467. <https://causalml-book.org/>
- Estève, L., Lemaitre, G., Grisel, O., Varoquaux, G., Amor, A., Lilian, Rospars, B., Schmitt, T., Liu, L., Kinoshita, B. P., *hackmd-deploy*, ph4ge, Steinbach, P., Boucaud, A., Muite, B., Boisberranger, J. du, Notter, M., Pierre, P, S., ... parmentelat. (2022). INRIA/scikit-learn-mooc: Third MOOC session. Zenodo. <https://doi.org/10.5281/zenodo.7220307>
- Hastie, T. (2009, ). *The elements of statistical learning: data mining, inference, and prediction*. Springer.
- Kleinberg, J., Ludwig, J., Mullainathan, S., & Obermeyer, Z. (2015). *Prediction policy problems*. American Economic Review, 105(5), 491–495.