Fundamental concepts and methods in machine learning

Tobi Louw

~~Fundamental concepts and methods in machine learning~~ Understanding the bias-variance trade-off, and how this influences a variety of common ML methods

Tobi Louw

Aim of the talk

- Introduce a fundamental concept:
  the bias-variance trade-off

- Show how the BV trade-off helps us understand how and why popular ML methods work

- Provide insight and resources to support independent learning

Aim of the talk

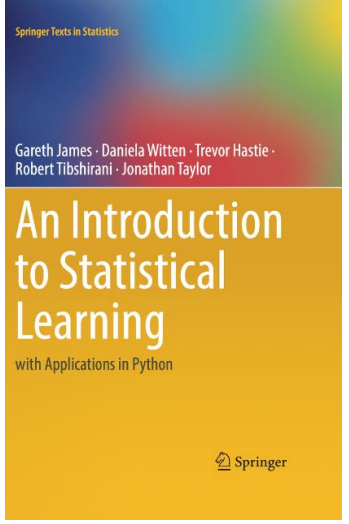
- Introduce a fundamental concept:
  the bias-variance trade-off

- Show how the BV trade-off helps us understand how and why popular ML methods work

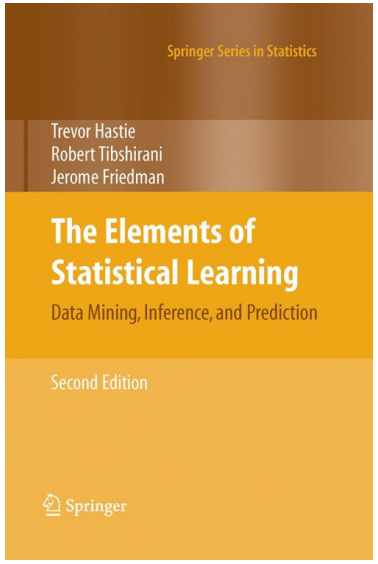
- Provide insight and resources to support independent learning

Aim of the talk

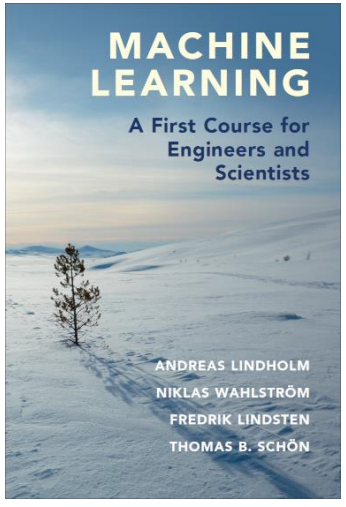
- Introduce a fundamental concept:
  the bias-variance trade-off
- Show how the BV trade-off helps us understand how and why popular ML methods work
- Provide insight and resources to support independent learning



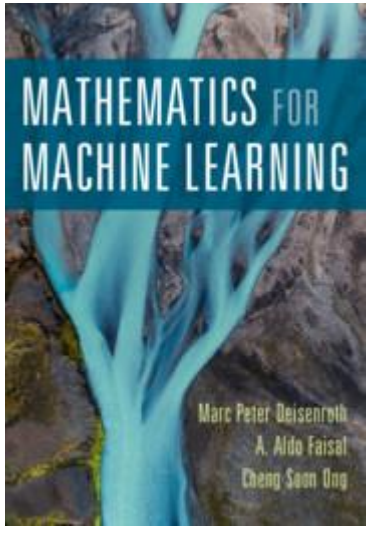
https://www.statlearning.com/



https://hastie.su.domains/ElemStatLearn/

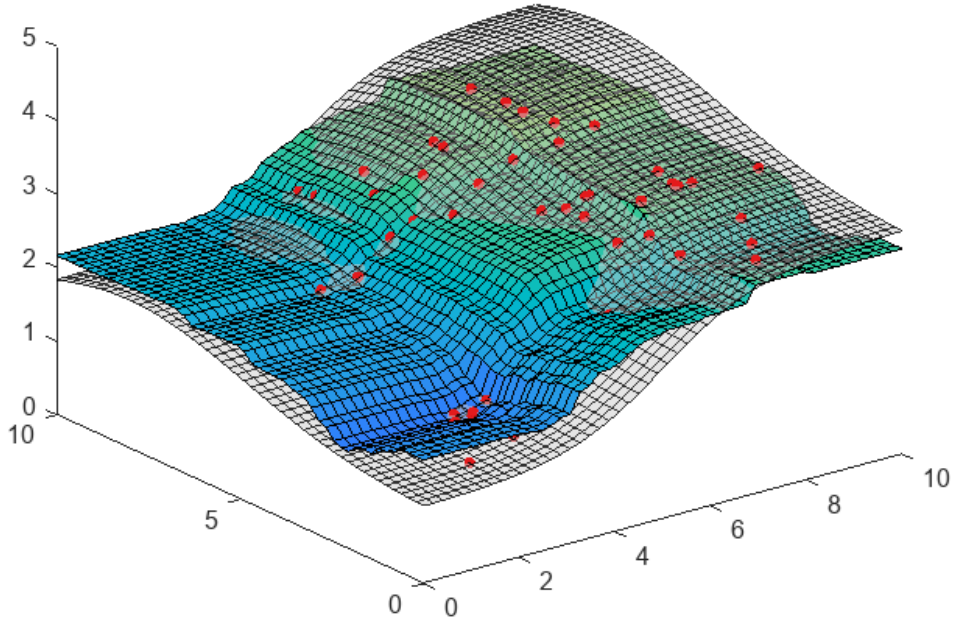


http://smlbook.org/



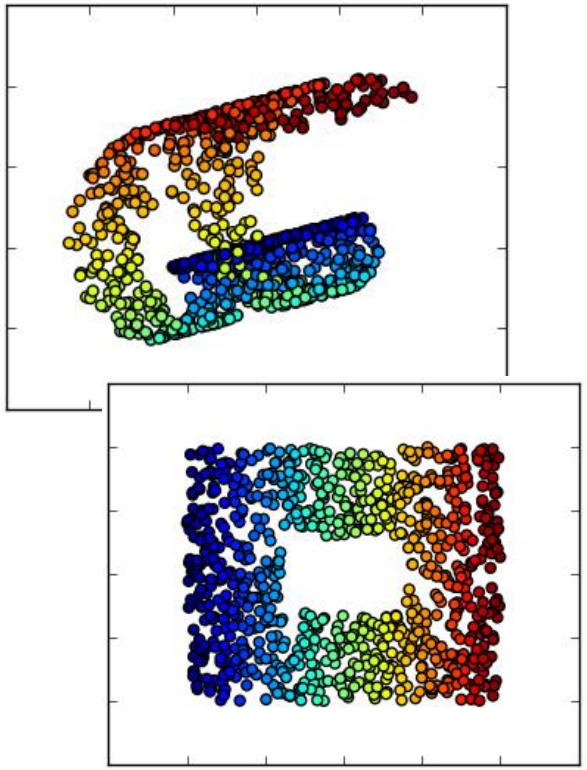
https://mml-book.github.io/

What is machine learning?



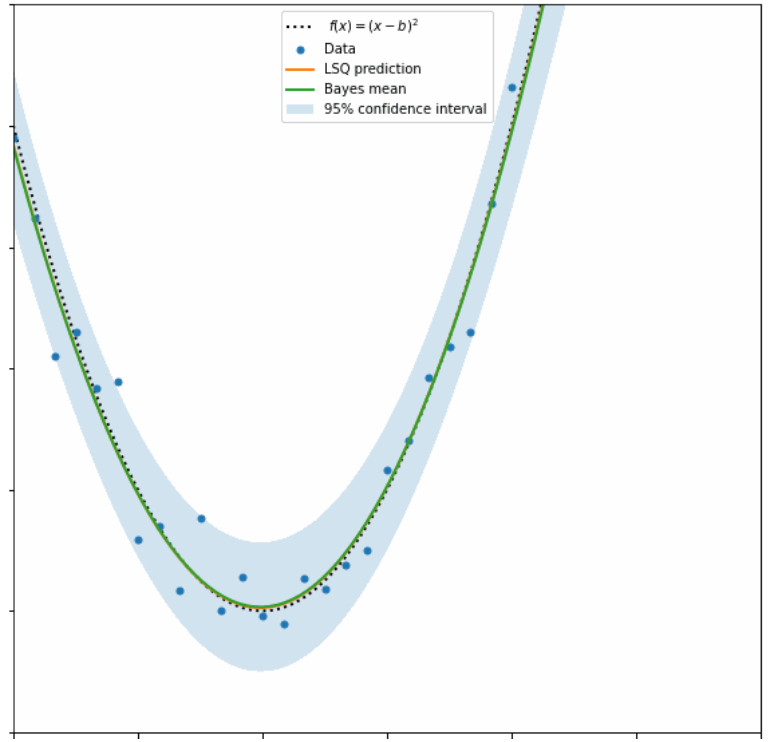
Supervised learning

$y = f(\mathbf{x})$



Unsupervised learning

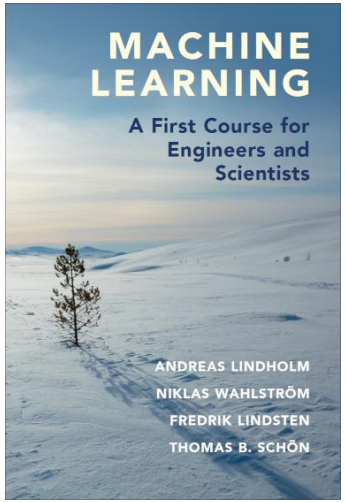
$T : X \to X'$



Reinforcement learning

$\pi(a|s)$

What is machine learning?



http://smlbook.org/

"The overall goal in supervised machine learning is to achieve as small an [error on new data] as possible."

Supervised ML aims to identify empirical models that extrapolate well.

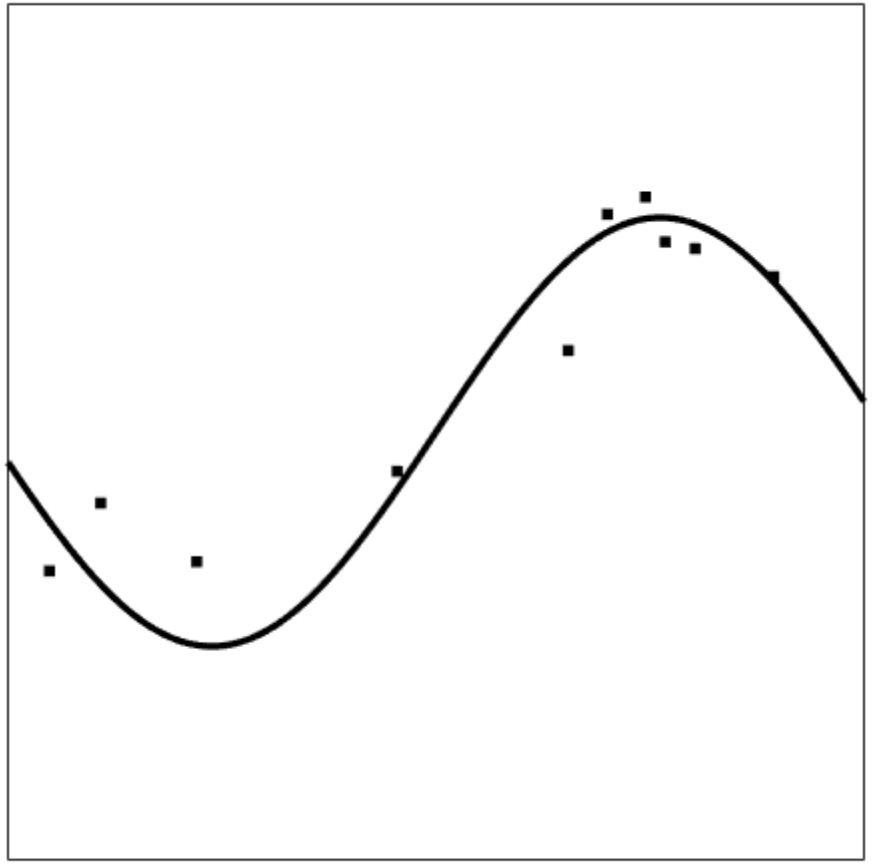
Supervised learning

- Consider some function $f(x)$
- Sample noise corrupted measurements

$$y_k = f(x_k) + \varepsilon,\ \ \varepsilon \sim \mathcal{N}(0, \sigma_y)$$

- Try to fit polynomial to the data

$$\hat{f}(x) = a_0 + a_1 x + a_2 x^2 \ldots + a_7 x^7$$



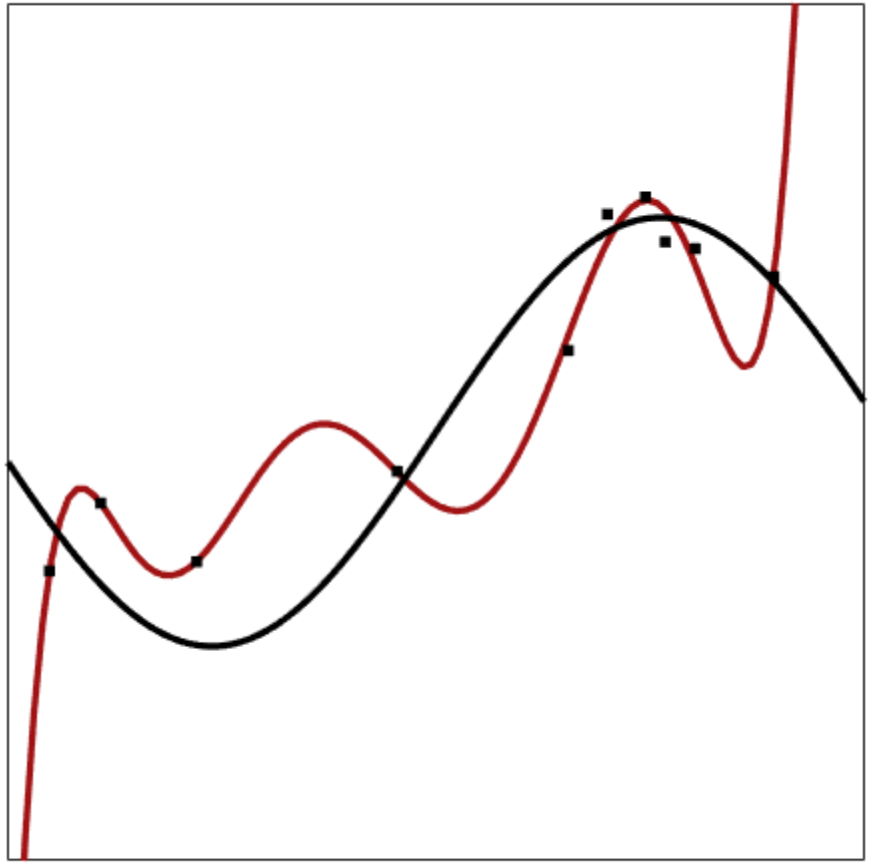
Supervised learning

- Consider some function $f(x)$
- Sample noise corrupted measurements

$$y_k = f(x_k) + \varepsilon,\ \ \varepsilon \sim \mathcal{N}(0, \sigma_y)$$

- Try to fit polynomial to the data

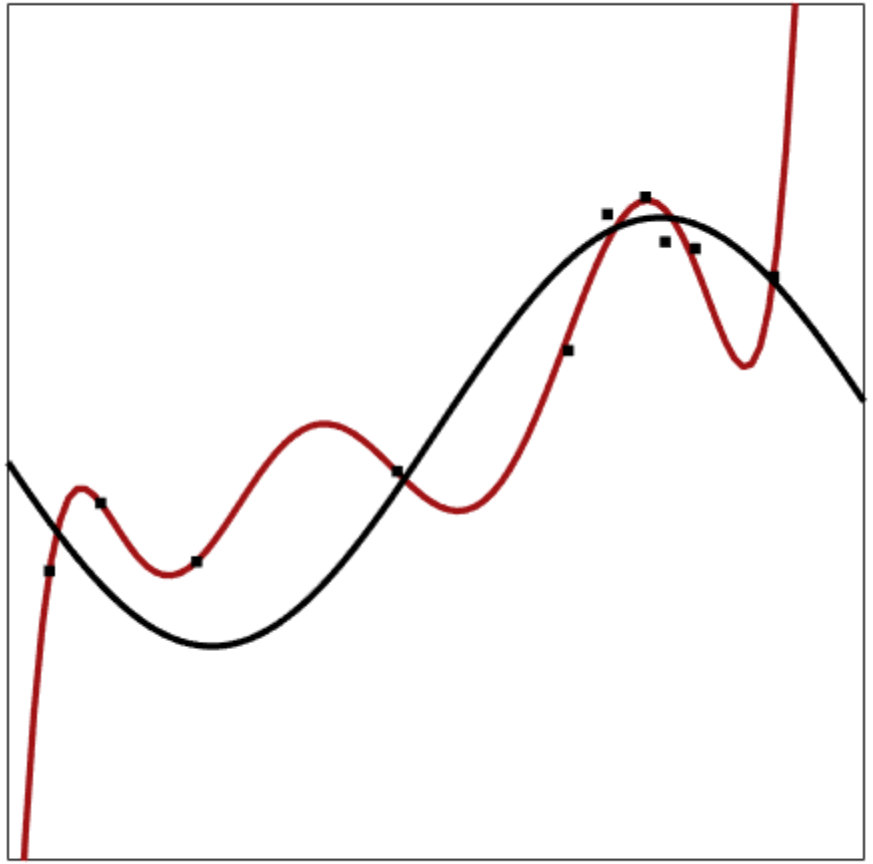
$$\hat{f}(x) = a_0 + a_1 x + a_2 x^2 \ldots + a_7 x^7$$



Supervised learning

- Polynomial is very flexible
- Overfitting:

  tries to match data points even if variation is due to noise in the measurement
- If the dataset were slightly different...



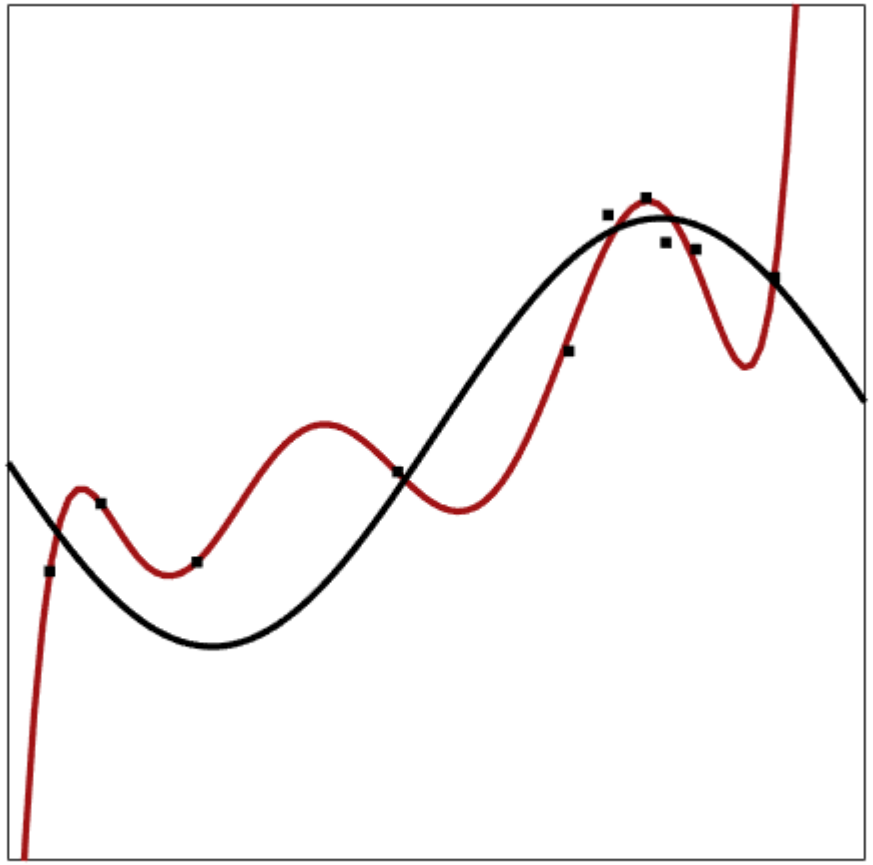
Supervised learning

- Polynomial is very flexible
- Overfitting:

  tries to match data points even if variation is due to noise in the measurement

- If the dataset were slightly different...
- Fitted spline is very different for each random dataset:

  high variance



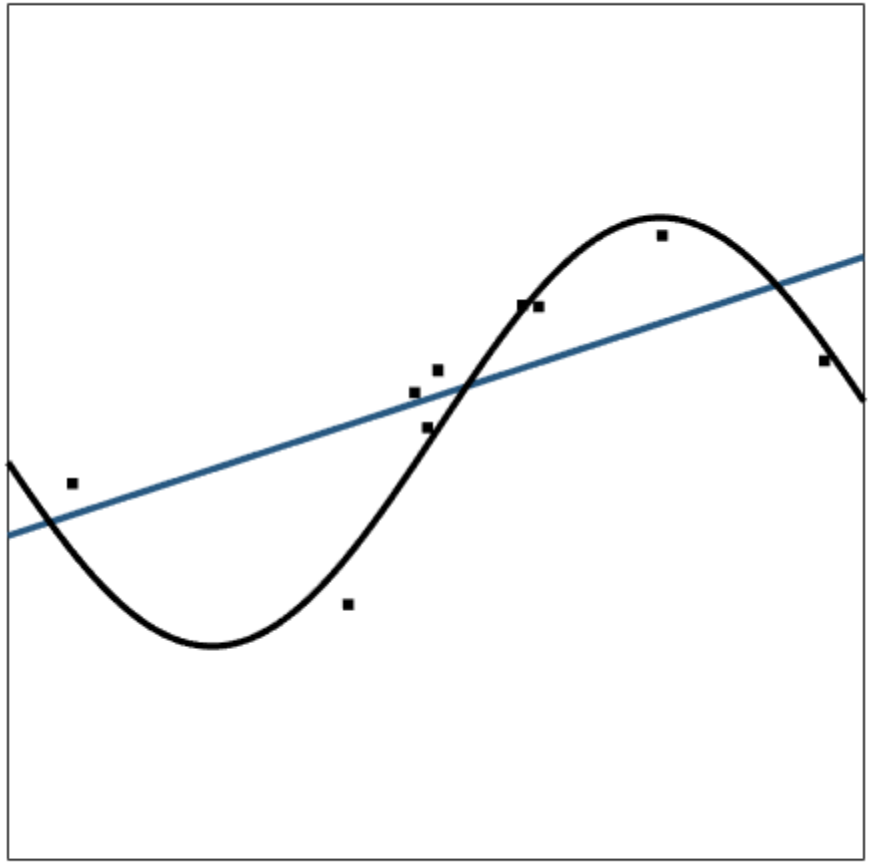
Supervised learning

- Consider some function $f(x)$
- Sample noise corrupted measurements

$$y_k = f(x_k) + \varepsilon,\ \ \varepsilon \sim \mathcal{N}(0, \sigma_y)$$

- Try to fit a line to the data

$$\hat{f}(x) = a_0 + a_1 x$$

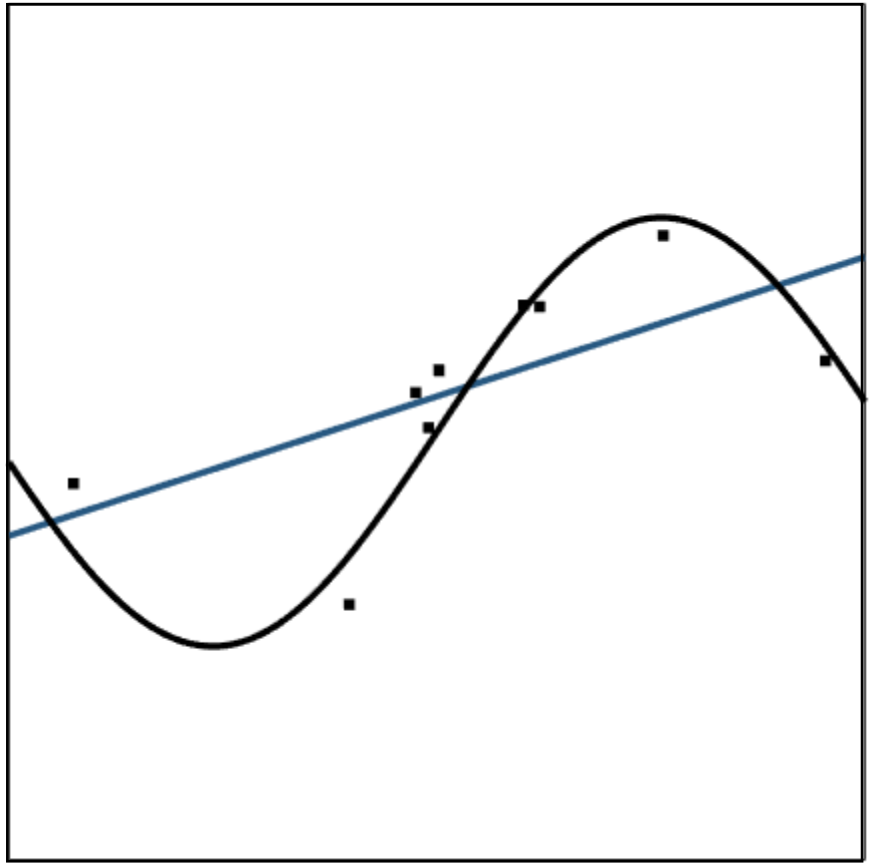


Supervised learning

- Model dos not fit data well
- Model has **high bias**:

  "biased" towards a specific shape, despite evidence to the contrary

- If the dataset were slightly different...



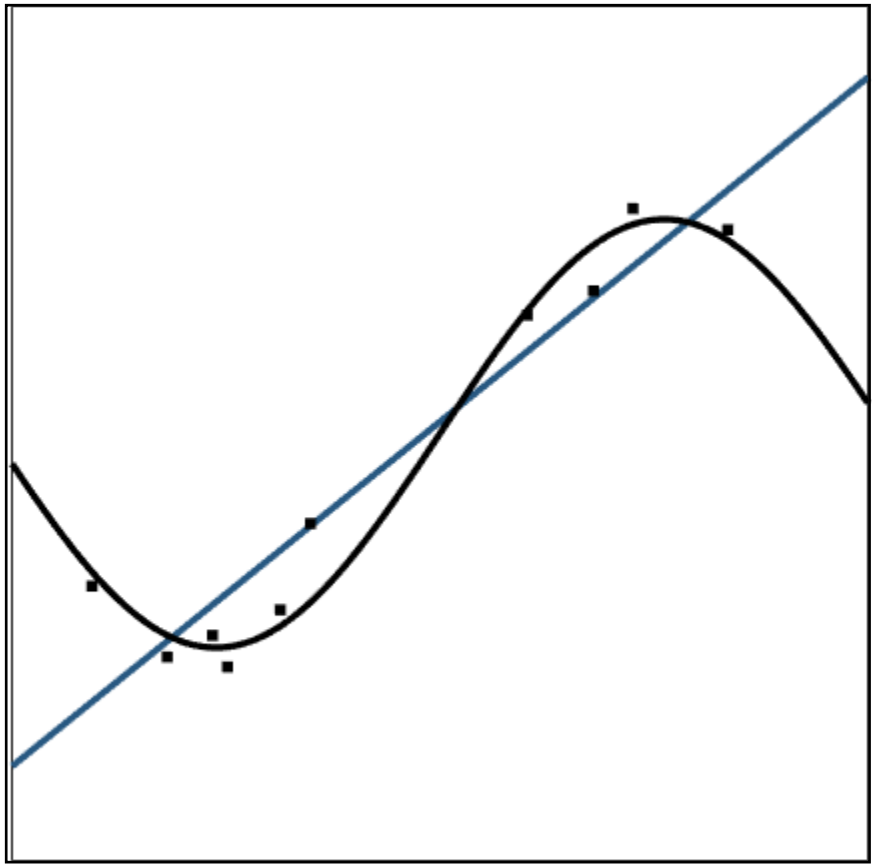
Supervised learning

- Model dos not fit data well
- Model has **high bias**:

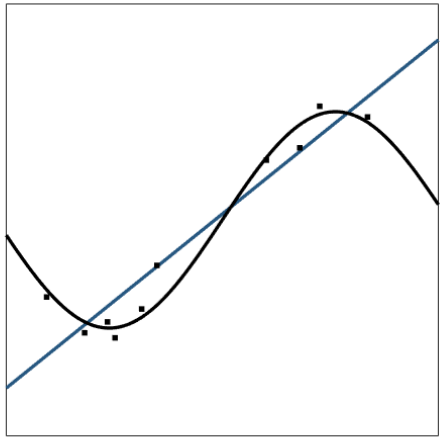
  "biased" towards a specific shape, despite evidence to the contrary

- If the dataset were slightly different...

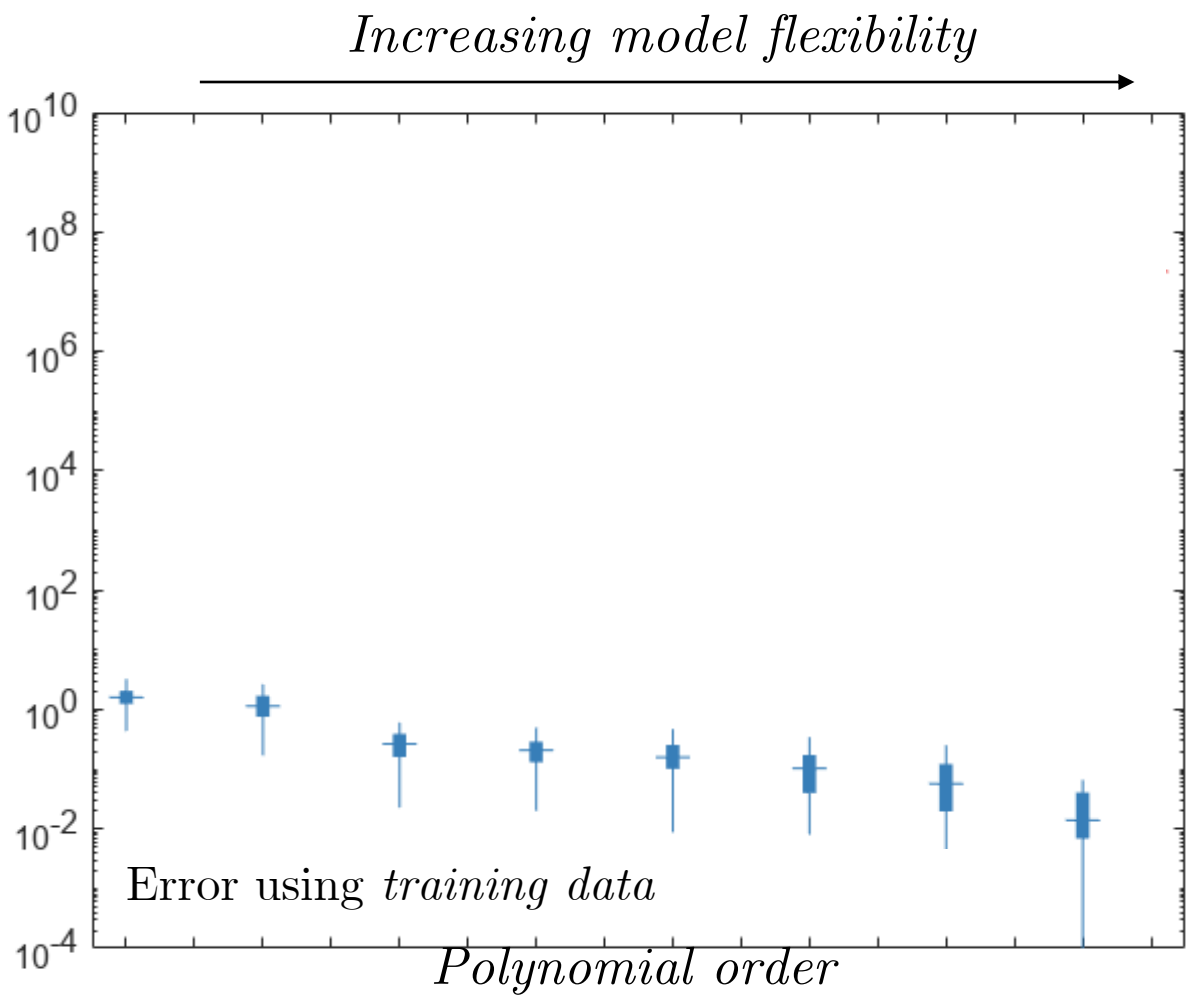
  variance is much lower

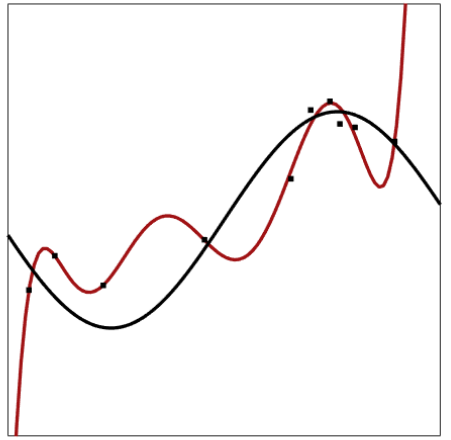


Prediction error using new data



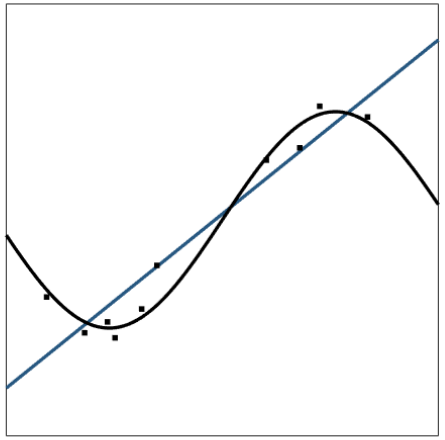
High bias, low variance



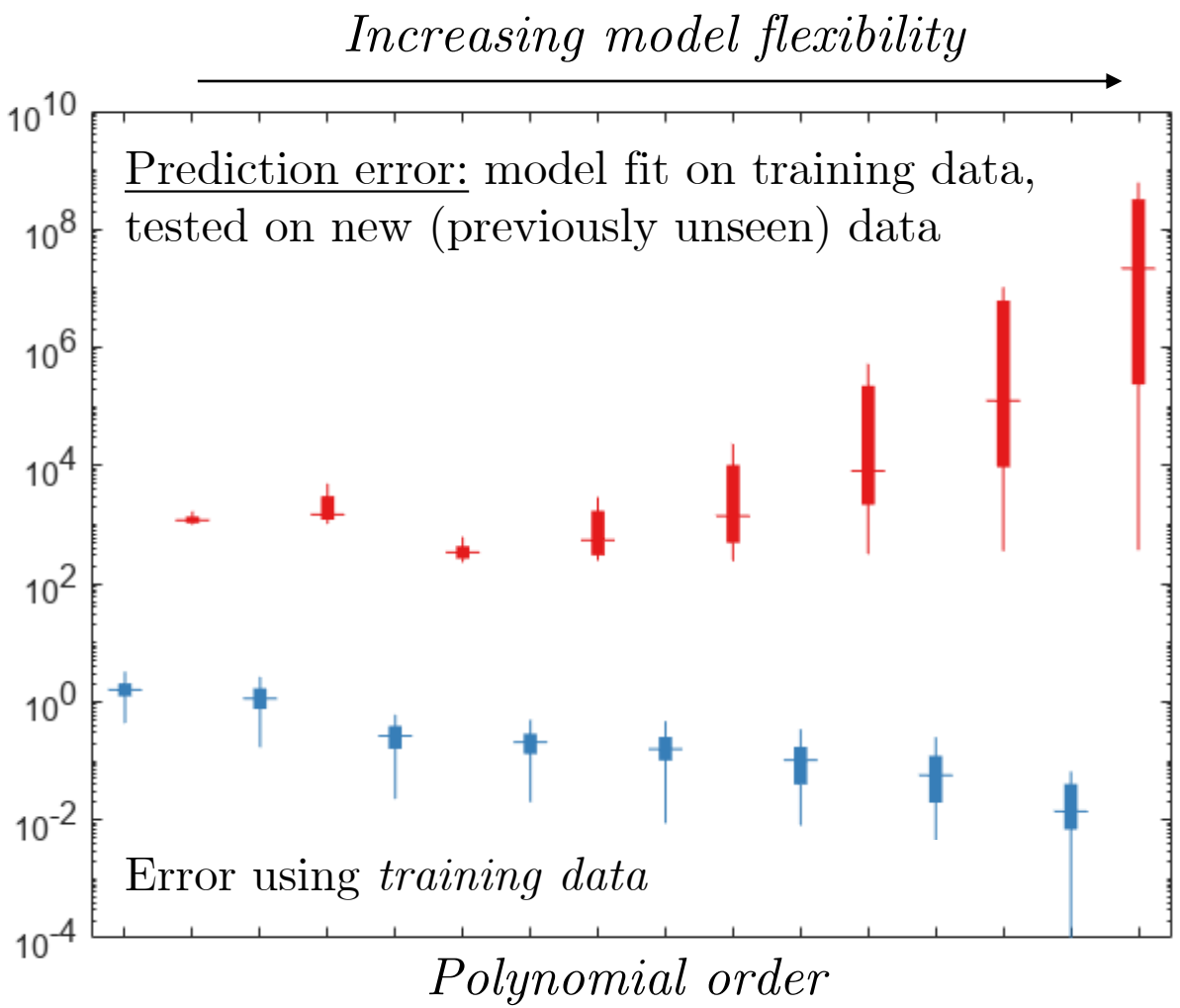


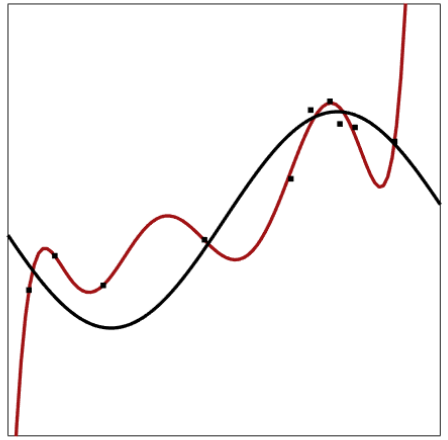
Low bias, high variance

Prediction error using new data



High bias, low variance





Low bias, high variance

Prediction error using new data

- Data generated by

$$y_k = f(x_k) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma_y)$$

- (*using square loss function*) Expected Prediction Error:

$$\mathrm{EPE}_k = \mathbb{E}\left[\left(y_k - \hat{f}(x_k)\right)^2\right]$$

$$= \mathbb{E}\left[\left(y_k - f(x_k)\right)^2\right] + \left(\boldsymbol{f(x_k)} - \mathbb{E}\left[\boldsymbol{\hat{f}(x_k)}\right]\right)^2 + \mathbb{E}\left[\left(\mathbb{E}\left[\boldsymbol{\hat{f}(x_k)}\right] - \boldsymbol{\hat{f}(x_k)}\right)^2\right]$$

Difference between "true" generating function and data (measurement noise)

Bias

Difference between "true" generating function and expected model

Variance

Difference between expected model and learnt model

Prediction error using new data

$$\mathrm{EPE}_k = \mathbb{E}\left[\left(y_k - f(x_k)\right)^2\right] + \left(\boldsymbol{f(x_k)} - \mathbb{E}\left[\hat{\boldsymbol{f}}(\boldsymbol{x_k})\right]\right)^2 + \mathbb{E}\left[\left(\mathbb{E}\left[\hat{\boldsymbol{f}}(\boldsymbol{x_k})\right] - \hat{\boldsymbol{f}}(\boldsymbol{x_k})\right)^2\right]$$

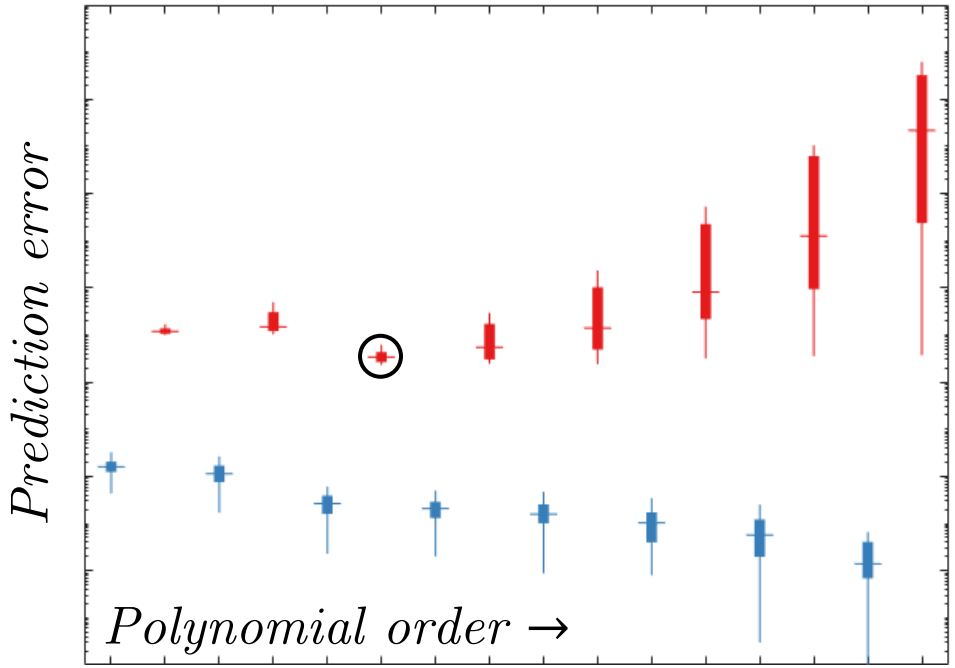
Measurement noise

Bias

Difference between "true" generating function and learnt model

Variance

Difference between expected model and learnt model



Prediction error using new data

$$\mathrm{EPE}_k = \mathbb{E}\left[\left(y_k - f(x_k)\right)^2\right] + \left(\boldsymbol{f(x_k)} - \mathbb{E}\left[\hat{\boldsymbol{f}}(\boldsymbol{x_k})\right]\right)^2 + \mathbb{E}\left[\left(\mathbb{E}\left[\hat{\boldsymbol{f}}(\boldsymbol{x_k})\right] - \hat{\boldsymbol{f}}(\boldsymbol{x_k})\right)^2\right]$$

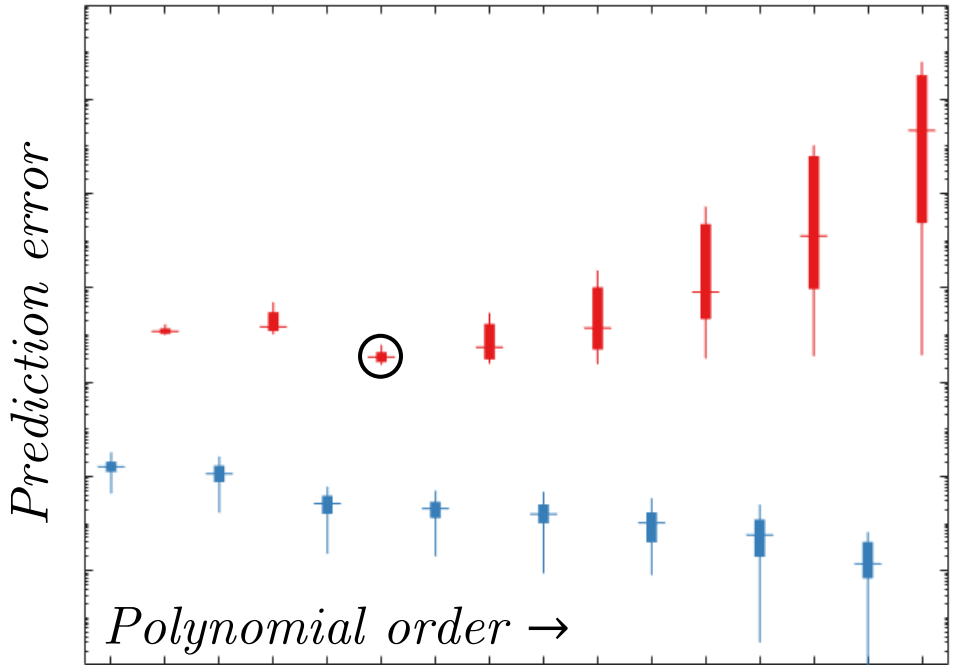
Measurement noise

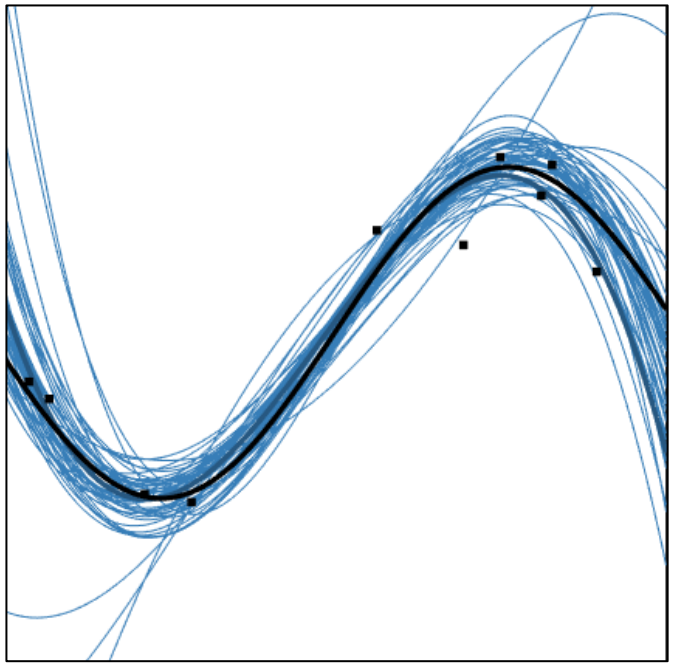
Bias

Difference between "true" generating function and learnt model

Variance

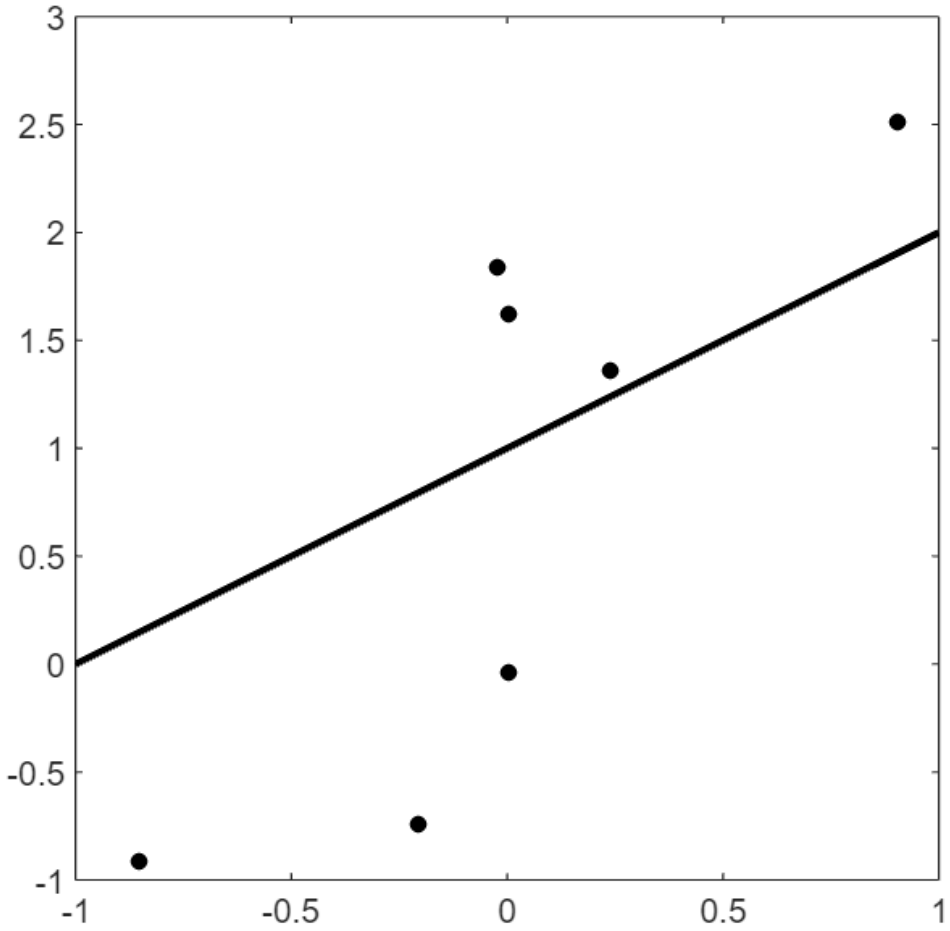
Difference between expected model and learnt model





Controlling the bias-variance trade-off

- Data generated by $y_k = mx + c + \varepsilon$

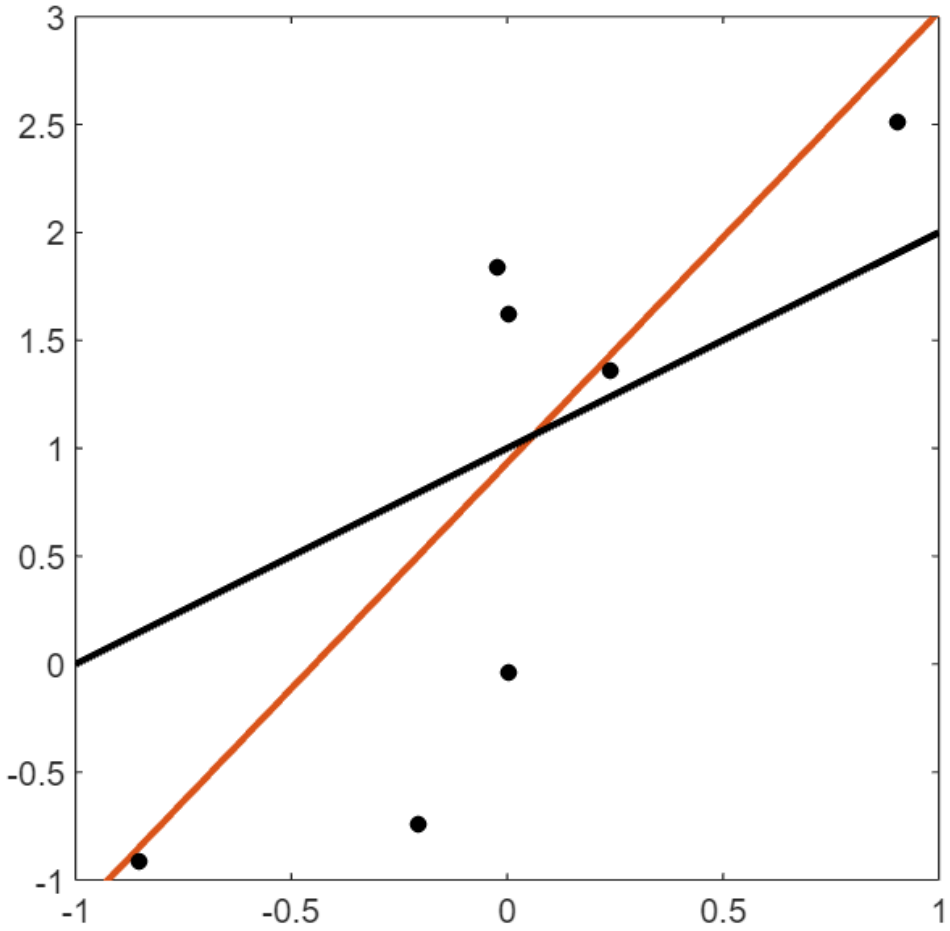


$$\text{Expected Prediction Error} = \mathbb{E}\left[\left(y_k - f(x_k)\right)^2\right] + \underbrace{\left(\boldsymbol{f(x_k)} - \mathbb{E}[\boldsymbol{\hat{f}(x_k)}]\right)^2}_{\textit{Bias}} + \underbrace{\mathbb{E}\left[\left(\mathbb{E}[\boldsymbol{\hat{f}(x_k)}] - \boldsymbol{\hat{f}(x_k)}\right)^2\right]}_{\textit{Variance}}$$

Controlling the bias-variance trade-off

- Data generated by $y_k = mx + c + \varepsilon$
- Find:

$$(m, c) \leftarrow \min\left\{\sum\left(y_k - (mx + c)\right)^2\right\}$$



$$\text{Expected Prediction Error} = \mathbb{E}\left[\left(y_k - f(x_k)\right)^2\right] + \underbrace{\left(\boldsymbol{f(x_k)} - \mathbb{E}[\hat{\boldsymbol{f}}(\boldsymbol{x_k})]\right)^2}_{\textit{Bias}} + \underbrace{\mathbb{E}\left[\left(\mathbb{E}[\hat{\boldsymbol{f}}(\boldsymbol{x_k})] - \hat{\boldsymbol{f}}(\boldsymbol{x_k})\right)^2\right]}_{\textit{Variance}}$$

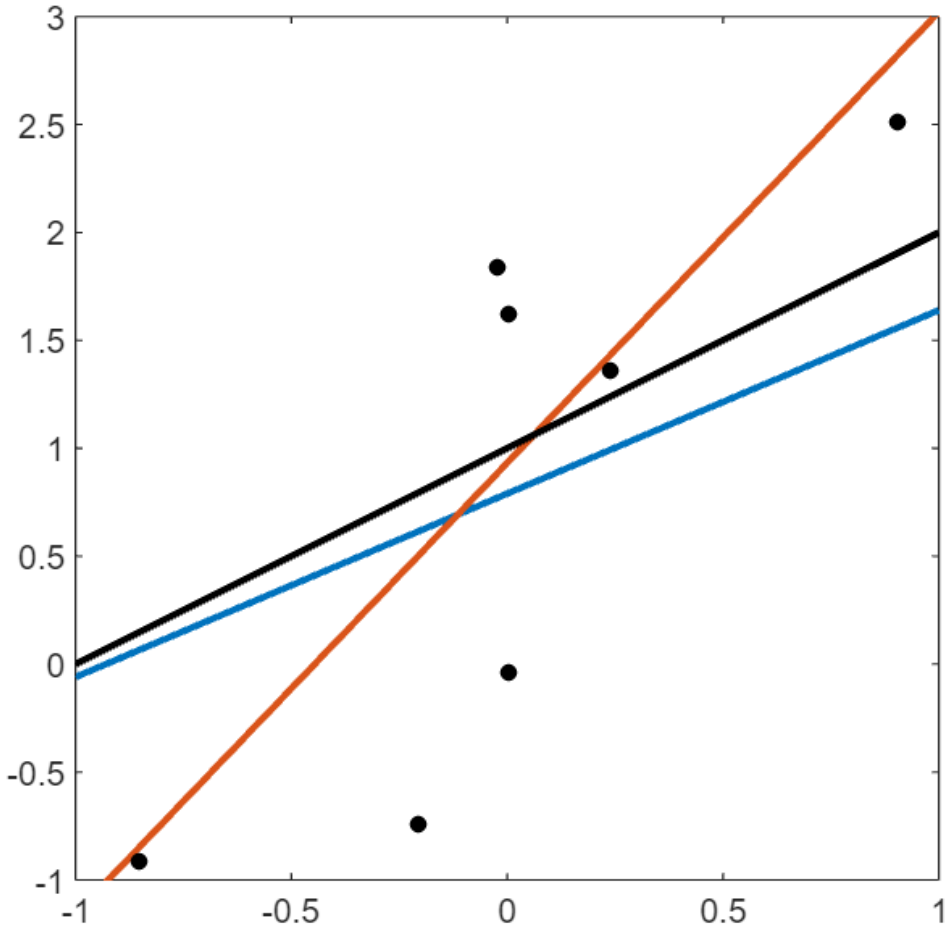
Controlling the bias-variance trade-off

- Data generated by $y_k = mx + c + \varepsilon$
- Find:

$$(\hat{m}, \hat{c}) \leftarrow \min\left\{\sum\left(y_k - (mx + c)\right)^2\right\}$$

- Alternative:

$$(\hat{m}_\lambda, \hat{c}_\lambda) \leftarrow \min\left\{\begin{matrix}\sum\left(y_k - (mx + c)\right)^2 \\ +\lambda(m^2 + c^2)\end{matrix}\right\}$$



$$\text{Expected Prediction Error} = \mathbb{E}\left[\left(y_k - f(x_k)\right)^2\right] + \underbrace{\left(\boldsymbol{f(x_k)} - \mathbb{E}[\hat{\boldsymbol{f}}(\boldsymbol{x_k})]\right)^2}_{\textit{Bias}} + \underbrace{\mathbb{E}\left[\left(\mathbb{E}[\hat{\boldsymbol{f}}(\boldsymbol{x_k})] - \hat{\boldsymbol{f}}(\boldsymbol{x_k})\right)^2\right]}_{\textit{Variance}}$$

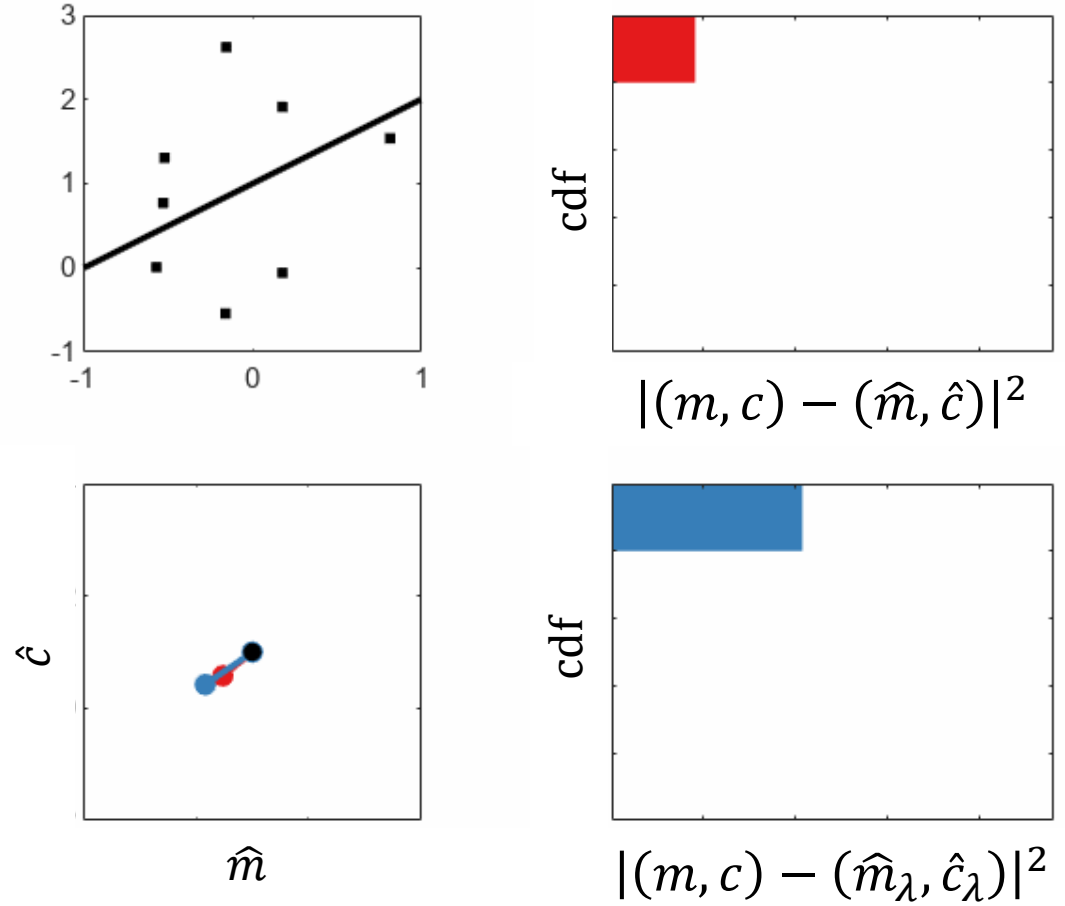
Controlling the bias-variance trade-off

- Data generated by $y_k = mx + c + \varepsilon$
- Find:

$$(\hat{m}, \hat{c}) \leftarrow \min \left\{ \sum \left(y_k - (mx + c) \right)^2 \right\}$$

$$(\hat{m}_\lambda, \hat{c}_\lambda) \leftarrow \min \left\{ \begin{array}{c} \sum \left(y_k - (mx + c) \right)^2 \\ +\lambda(m^2 + c^2) \end{array} \right\}$$

- Plot the fitted $\hat{m}$ and $\hat{c}$ for many different data sets



$$\text{Expected Prediction Error} = \mathbb{E}\left[\left(y_k - f(x_k) \right)^2 \right] + \underbrace{\left(\boldsymbol{f(x_k)} - \mathbb{E}[\hat{\boldsymbol{f}}(\boldsymbol{x_k})] \right)^2}_{\textit{Bias}} + \underbrace{\mathbb{E}\left[\left(\mathbb{E}[\hat{\boldsymbol{f}}(\boldsymbol{x_k})] - \hat{\boldsymbol{f}}(\boldsymbol{x_k}) \right)^2 \right]}_{\textit{Variance}}$$

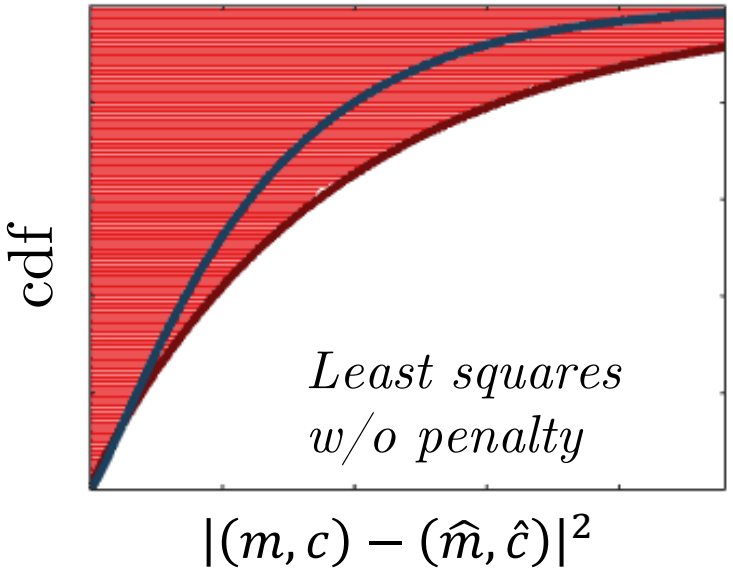
Controlling the BV trade-off

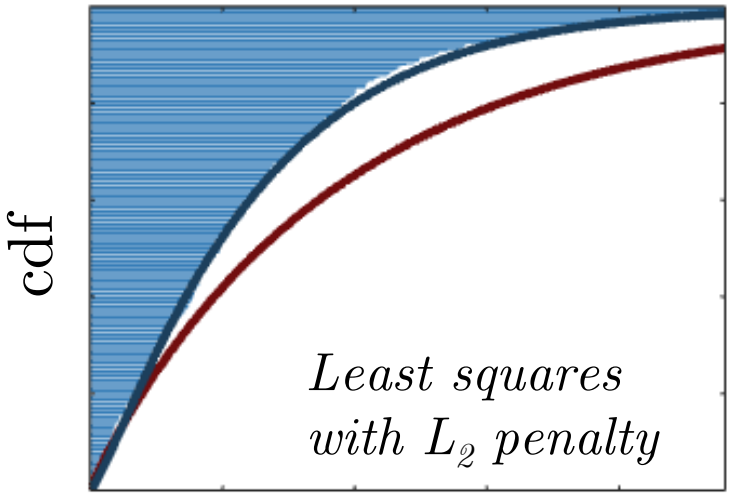
- Data generated by $y_k = mx + c + \varepsilon$
- Find:

$$(\hat{m}, \hat{c}) \leftarrow \min\left\{\sum\left(y_k - (mx + c)\right)^2\right\}$$

$$(\hat{m}_\lambda, \hat{c}_\lambda) \leftarrow \min\left\{\begin{matrix}\sum\left(y_k - (mx + c)\right)^2 \\ +\lambda(m^2 + c^2)\end{matrix}\right\}$$

- Plot the fitted $\hat{m}$ and $\hat{c}$ for many different data sets



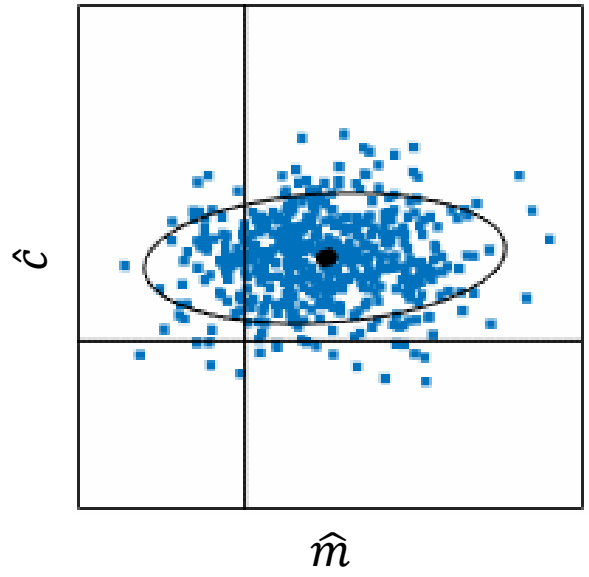


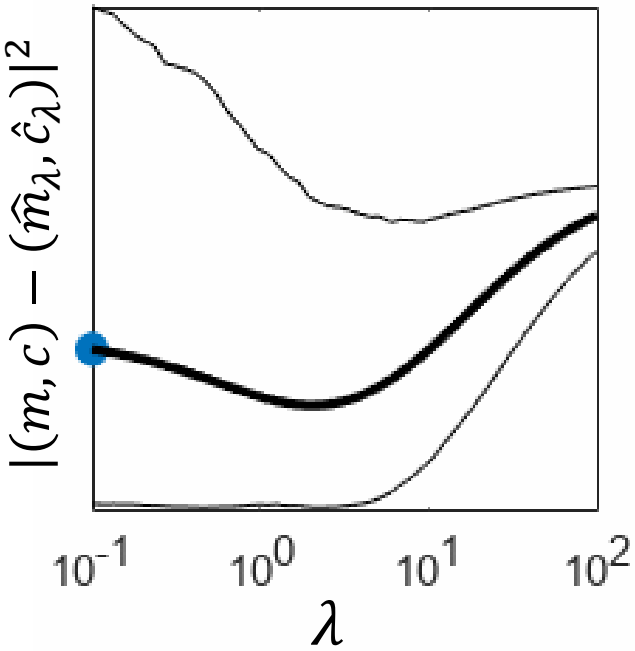
$$\text{Expected Prediction Error} = \mathbb{E}\left[\left(y_k - f(x_k)\right)^2\right] + \underbrace{\left(\boldsymbol{f(x_k)} - \mathbb{E}[\hat{\boldsymbol{f}}(\boldsymbol{x_k})]\right)^2}_{\textit{Bias}} + \underbrace{\mathbb{E}\left[\left(\mathbb{E}[\hat{\boldsymbol{f}}(\boldsymbol{x_k})] - \hat{\boldsymbol{f}}(\boldsymbol{x_k})\right)^2\right]}_{\textit{Variance}}$$

Controlling the BV trade-off

$$(\hat{m}_\lambda, \hat{c}_\lambda) \leftarrow \min \left\{ \begin{matrix} \Sigma\left(y_k - (mx + c)\right)^2 \\ +\lambda(m^2 + c^2) \end{matrix} \right\}$$

- L_2 penalty → biased model:
  $\mathbb{E}[(\hat{m}_\lambda, \hat{c}_\lambda)] \neq (m, c)$
- L_2 penalty → reduced variance:
  $\mathbb{E}[|\mathbb{E}[(\hat{m}_\lambda, \hat{c}_\lambda)] - (\hat{m}_\lambda, \hat{c}_\lambda)|^2]$
- Regularisation parameter controls bias vs. variance

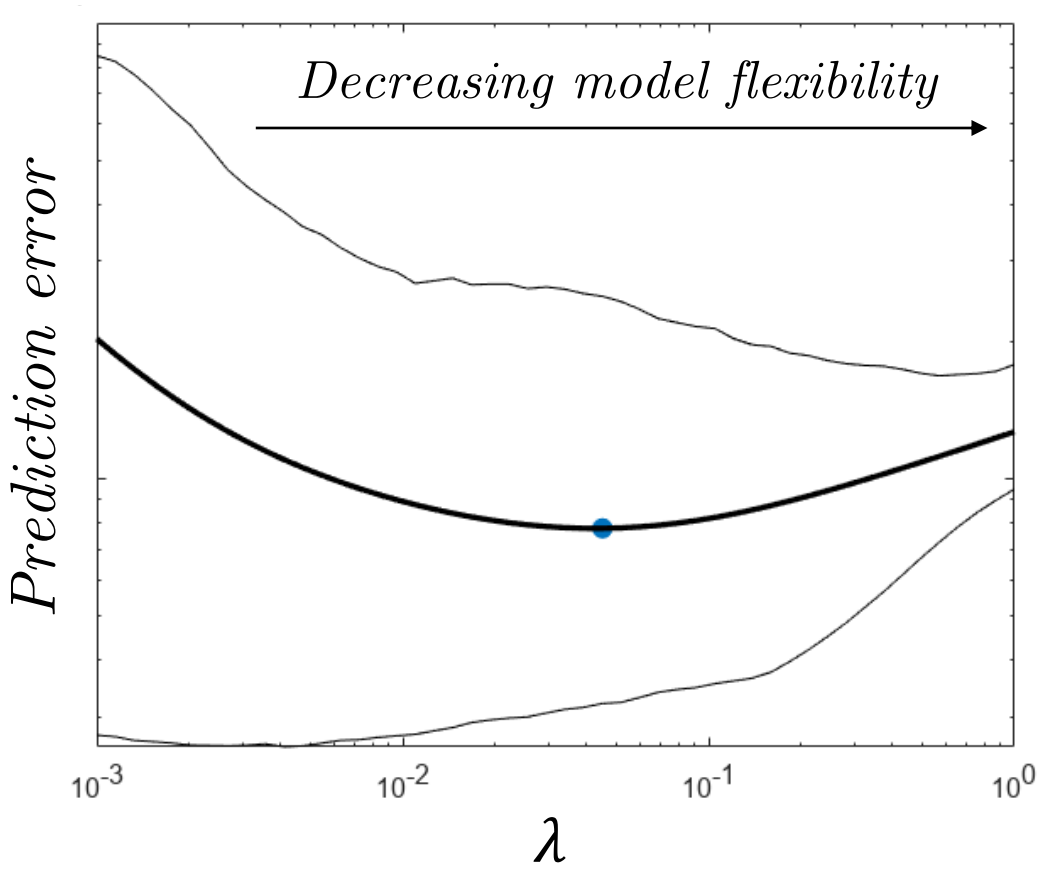


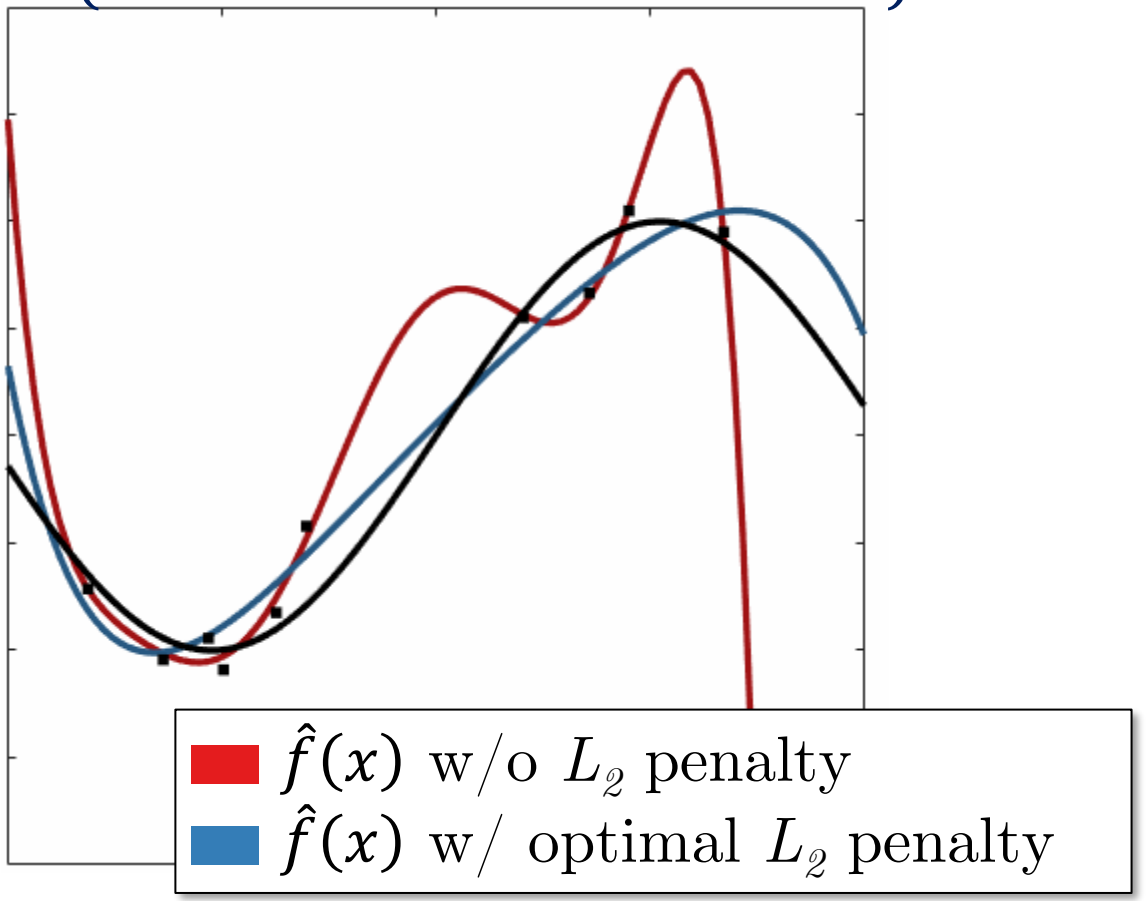


$$\text{Expected Prediction Error} = \mathbb{E}\left[\left(y_k - f(x_k)\right)^2\right] + \underbrace{\left(\boldsymbol{f(x_k)} - \mathbb{E}[\hat{\boldsymbol{f}}(\boldsymbol{x_k})]\right)^2}_{\textit{Bias}} + \underbrace{\mathbb{E}\left[\left(\mathbb{E}[\hat{\boldsymbol{f}}(\boldsymbol{x_k})] - \hat{\boldsymbol{f}}(\boldsymbol{x_k})\right)^2\right]}_{\textit{Variance}}$$

Controlling the bias-variance trade-off

$$\hat{f}(x) = \hat{a}_0 + \hat{a}_1 x + \hat{a}_2 x^2 \ldots + \hat{a}_7 x^7, \quad \hat{\mathbf{a}} \leftarrow \min \left\{ \sum \left(y_k - \hat{f}(x_k; \mathbf{a}) \right)^2 + \lambda |\mathbf{a}|^2 \right\}$$



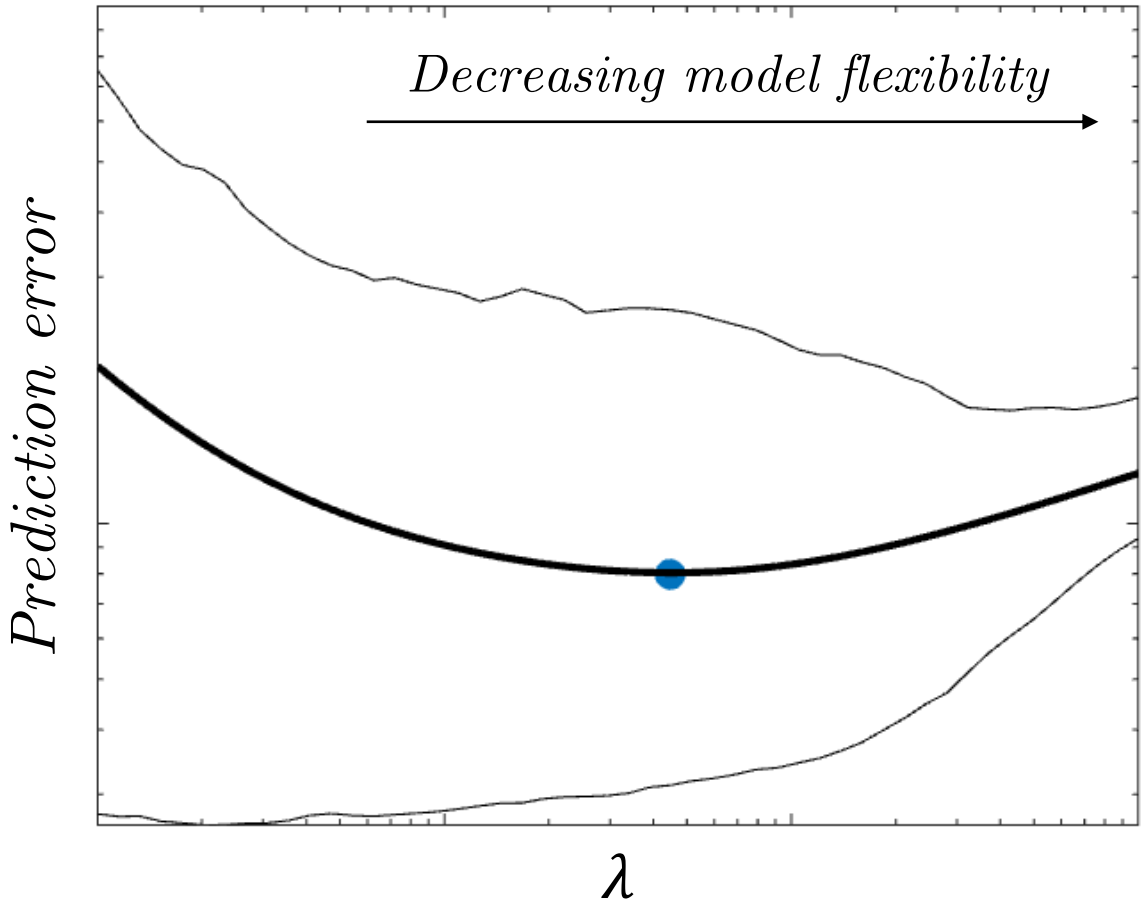


Controlling the bias-variance trade-off

$$\text{Expected Prediction Error} = \mathbb{E}\left[\left(y_k - f(x_k)\right)^2\right] + \underbrace{\left(\boldsymbol{f(x_k)} - \mathbb{E}\left[\hat{\boldsymbol{f}}(\boldsymbol{x_k})\right]\right)^2}_{Bias} + \underbrace{\mathbb{E}\left[\left(\mathbb{E}\left[\hat{\boldsymbol{f}}(\boldsymbol{x_k})\right] - \hat{\boldsymbol{f}}(\boldsymbol{x_k})\right)^2\right]}_{Variance}$$

- Expected Prediction Error (EPE) can be minimized by controlling model bias and variance through **regularisation**
- Selection of polynomial order is a form of regularisation
- L_2 penalty term on the parameter vector, $\lambda|\boldsymbol{\theta}|^2$: *ridge regression*
  - Continuous control of bias (as opposed to discrete polynomial order)
  - Applicable to broad range of parametric models (e.g., neural networks)

Working with limited data



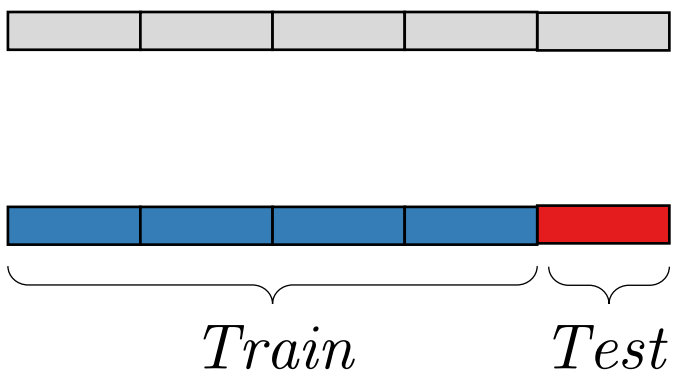
- Expected Prediction Error (EPE) cannot be calculated directly
- EPE must be estimated to select λ

k-fold Cross validation

K-fold cross-validation

Data set split into k "folds" (e.g. $k = 5$)

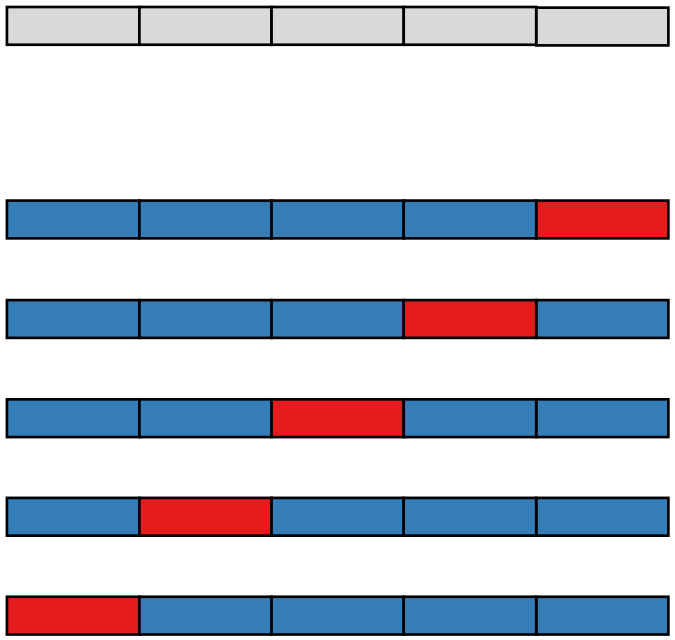
K-fold cross-validation



Data set split into k "folds" (e.g. $k = 5$)

Use $k-1$ folds for training, 1 fold for testing:
obtain a single estimate of prediction error

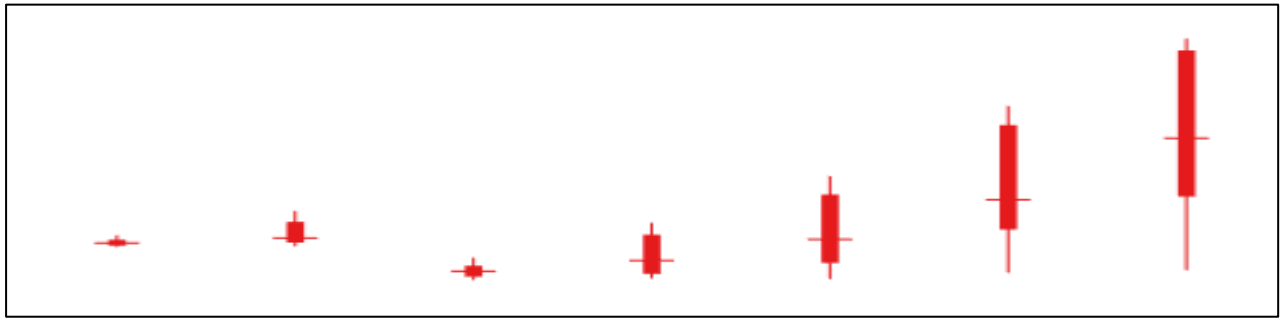
K-fold cross-validation



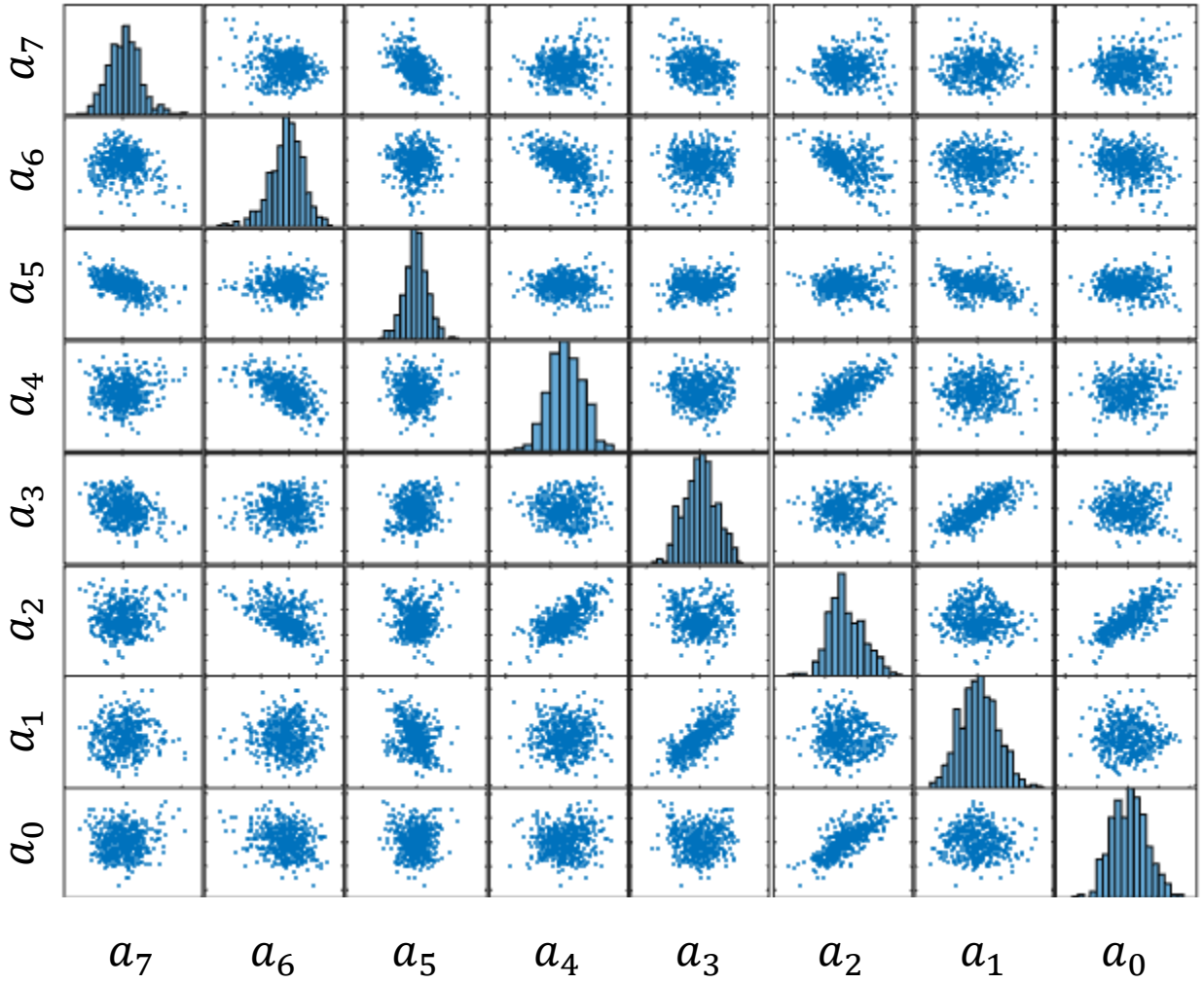
Data set split into k "folds" (e.g. $k = 5$)

Use $k-1$ folds for training, 1 fold for testing:
obtain a single estimate of prediction error

Repeat k times:
obtain k estimates of prediction error



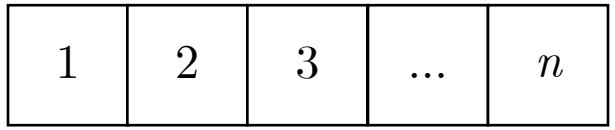
Working with limited data



- Often interested in the statistical properties of the parameters (e.g., $\mathbb{E}[\hat{\mathbf{a}}], \mathrm{var}[\hat{\mathbf{a}}]$)

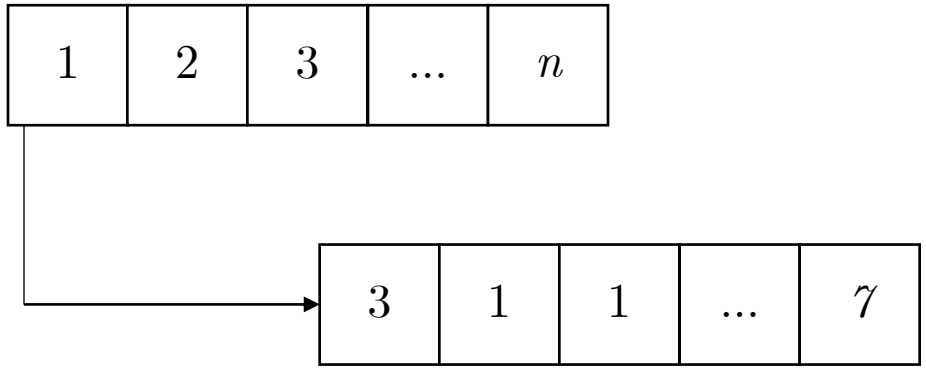
Bootstrap

Bootstrap



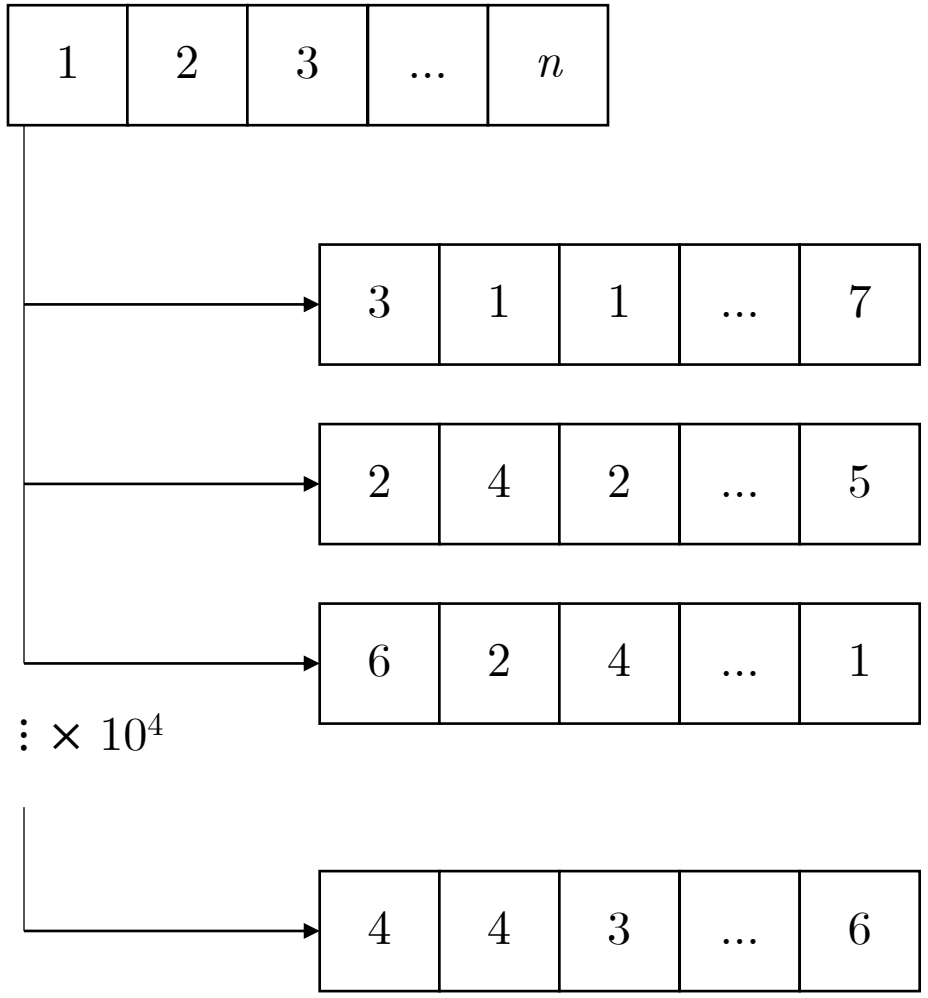
- Data set with n observations

Bootstrap



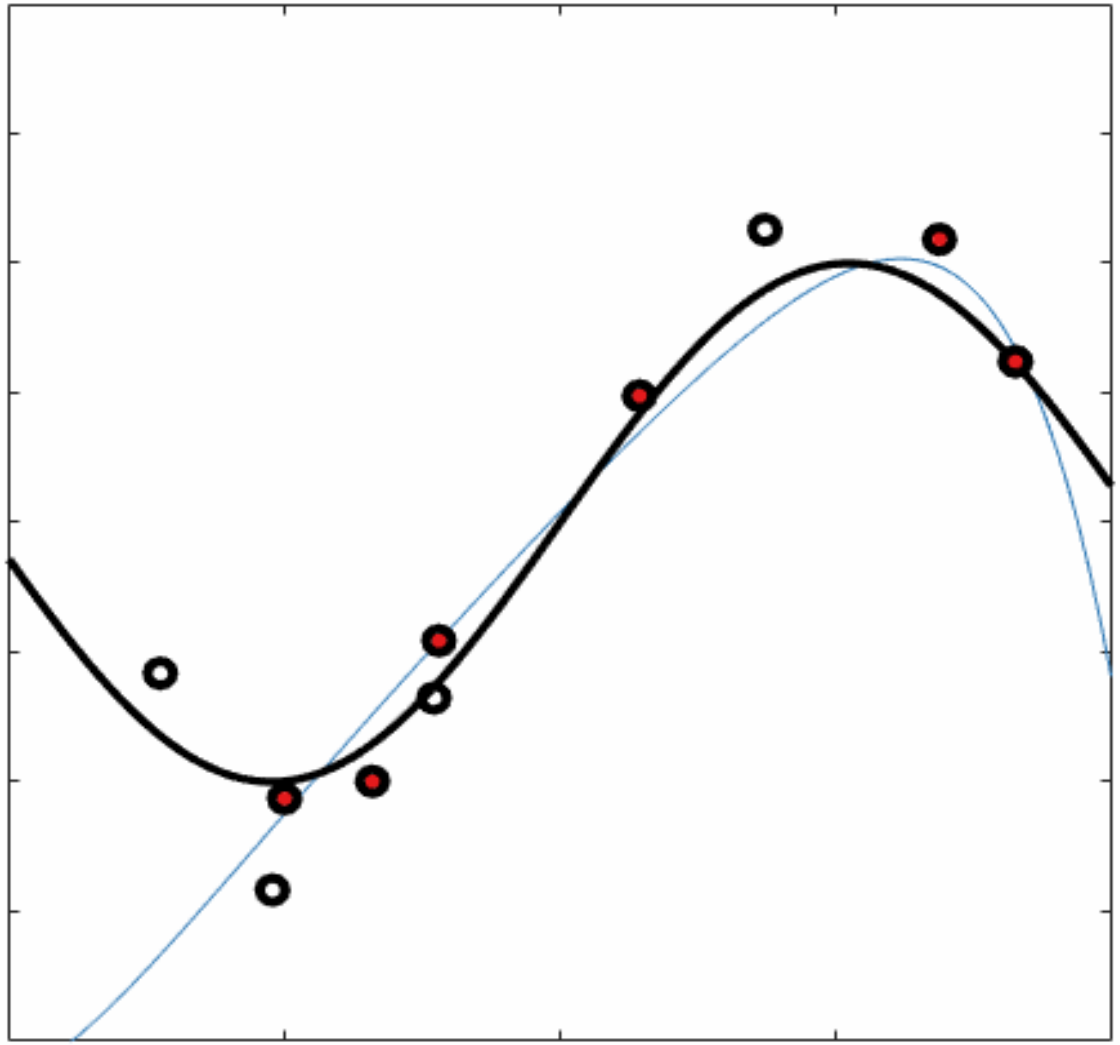
- Data set with n observations
- Resample n observations from data set **w/ replacement**
- Fit model to resampled data

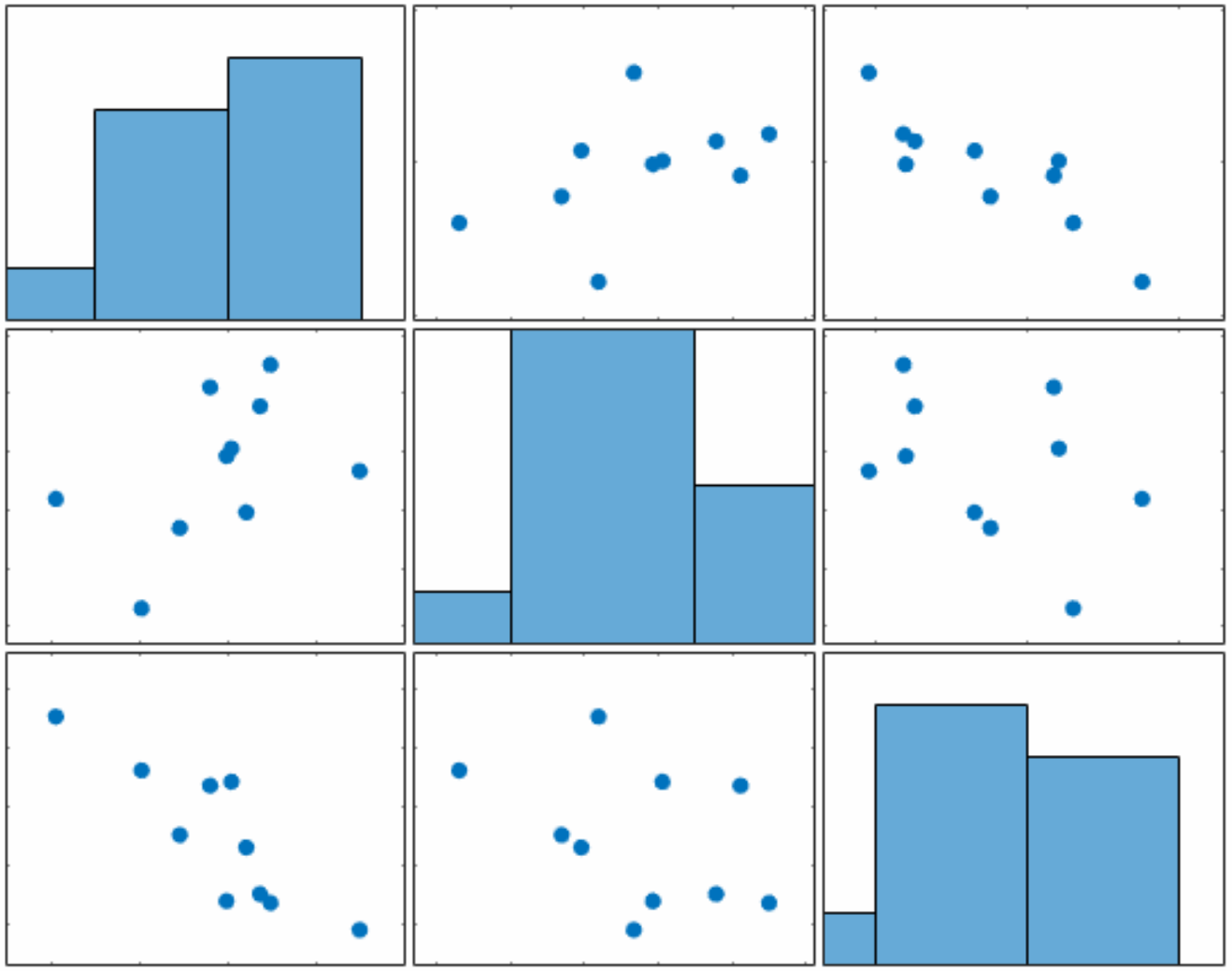
Bootstrap



- Data set with n observations
- Resample n observations from data set **w/ replacement**
- Fit model to resampled data
- Repeat many times

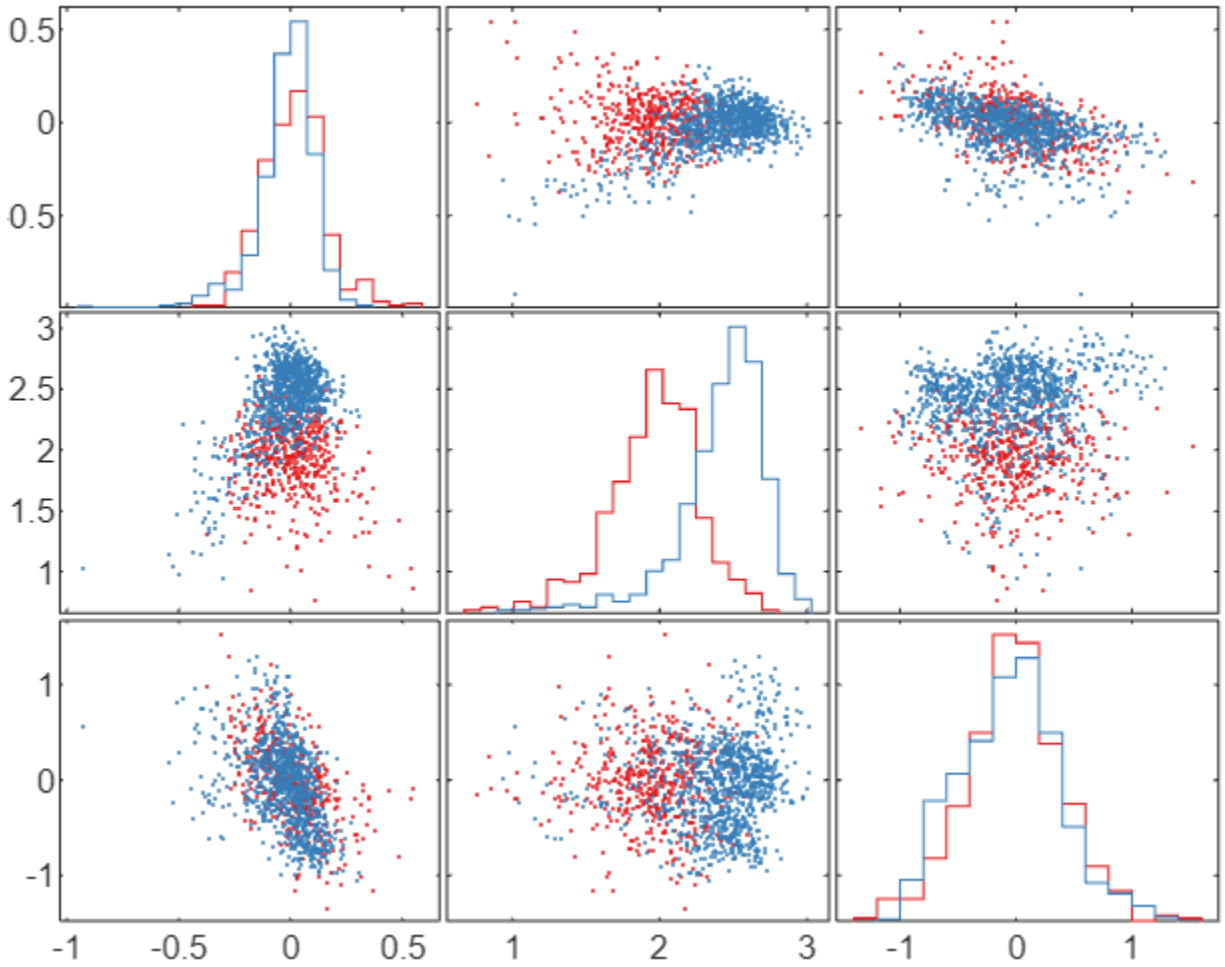
Bootstrap





Bootstrap

- Data set with n observations
- Resample n observations from data set **w/ replacement**
- Fit model to resampled data
- Repeat many times



Non-parametric regression

- Parametric regression:
  - *Parameters θ fully define model*
  - *Once parameters are known, data $\mathcal{D} = \{x_k, y_k\}$ can be discarded:*

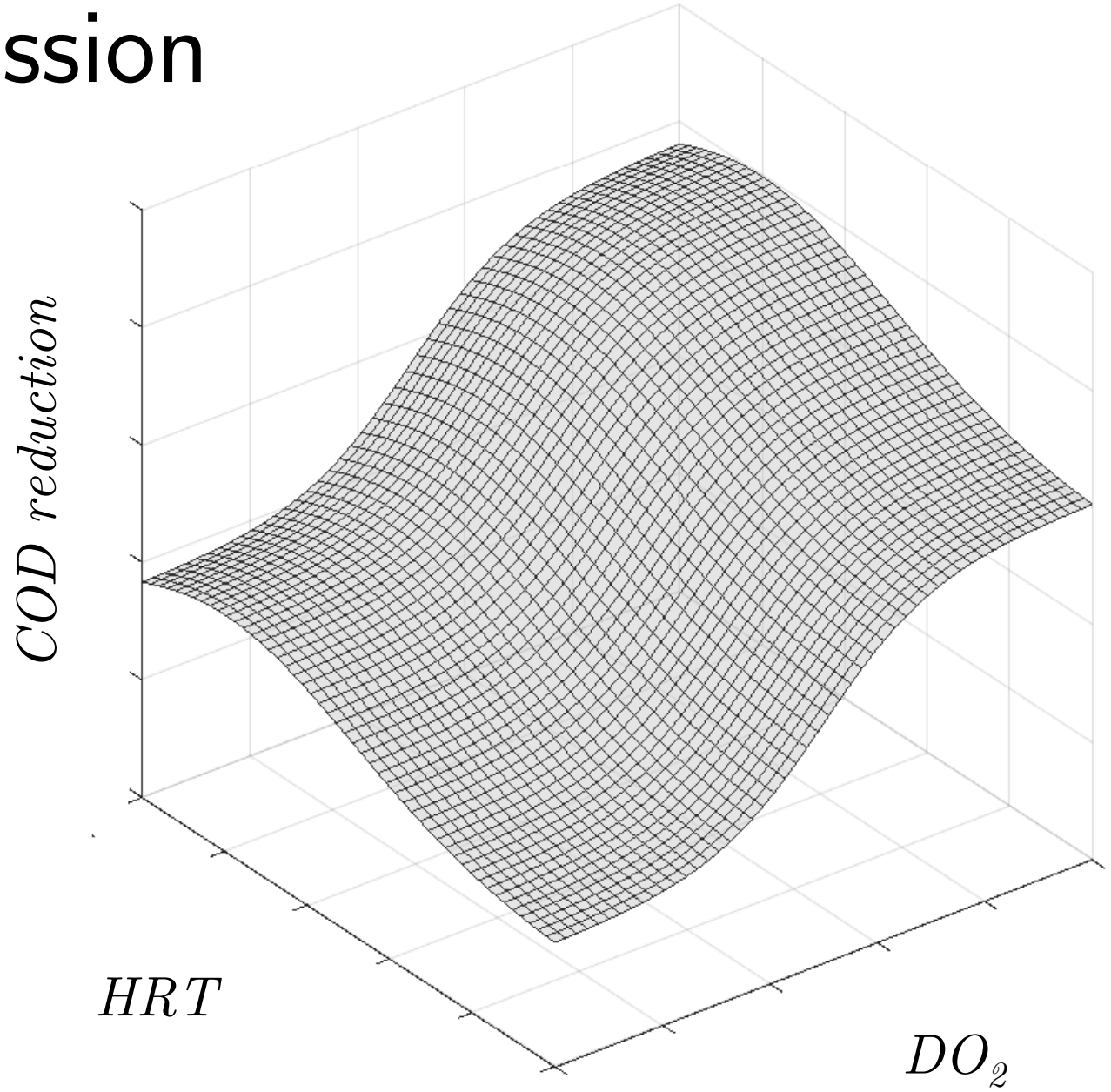
$$\mathcal{D} \rightarrow \boldsymbol{\theta}, \qquad y = f(x; \boldsymbol{\theta})$$

- Non-parametric regression:
  - *Predictions depend on the data*

$$y = f(x; \mathcal{D})$$

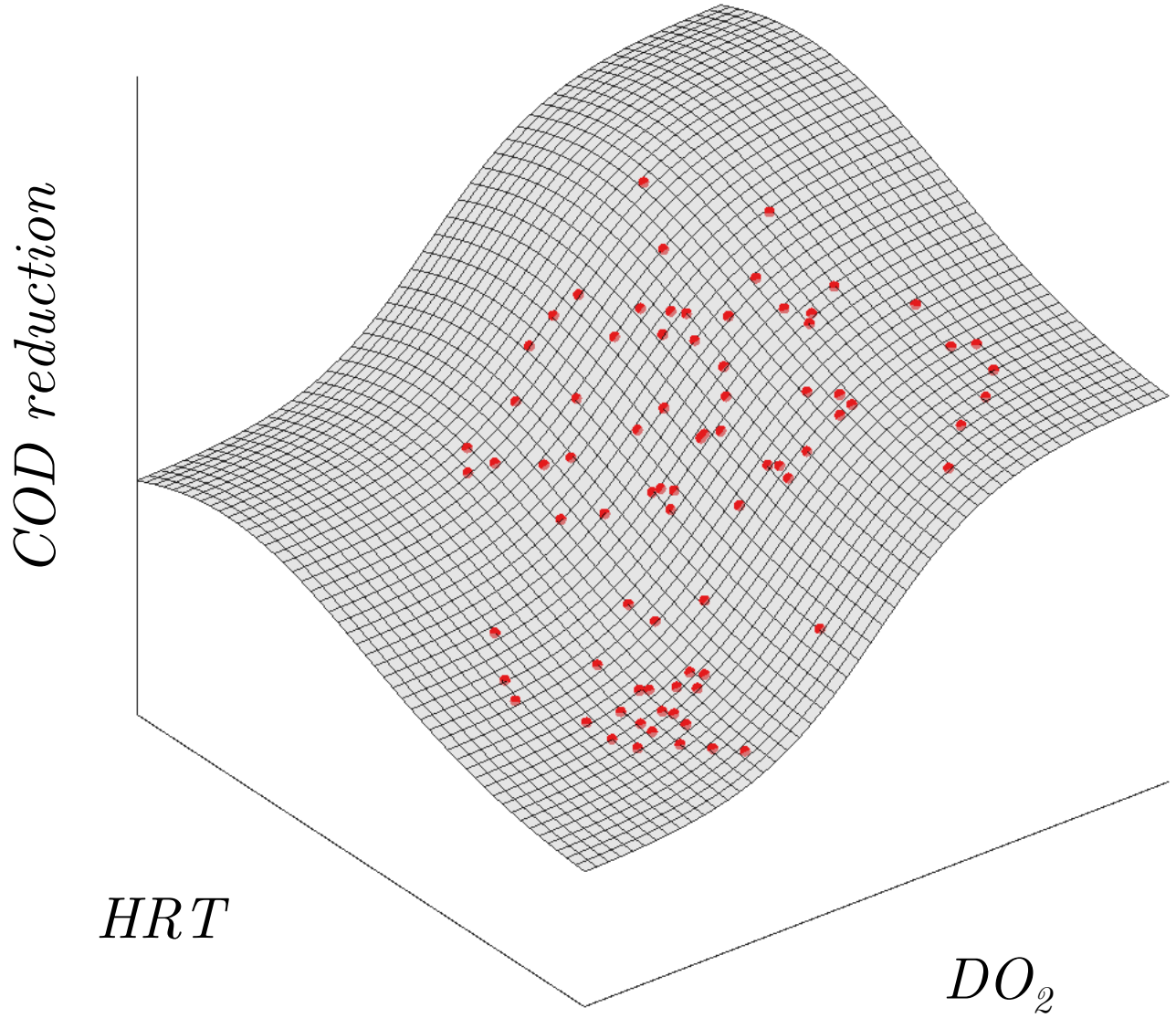
Non-parametric regression

- Consider predicting COD reduction given DO_2 and HRT in an activated sludge pond



Non-parametric regression

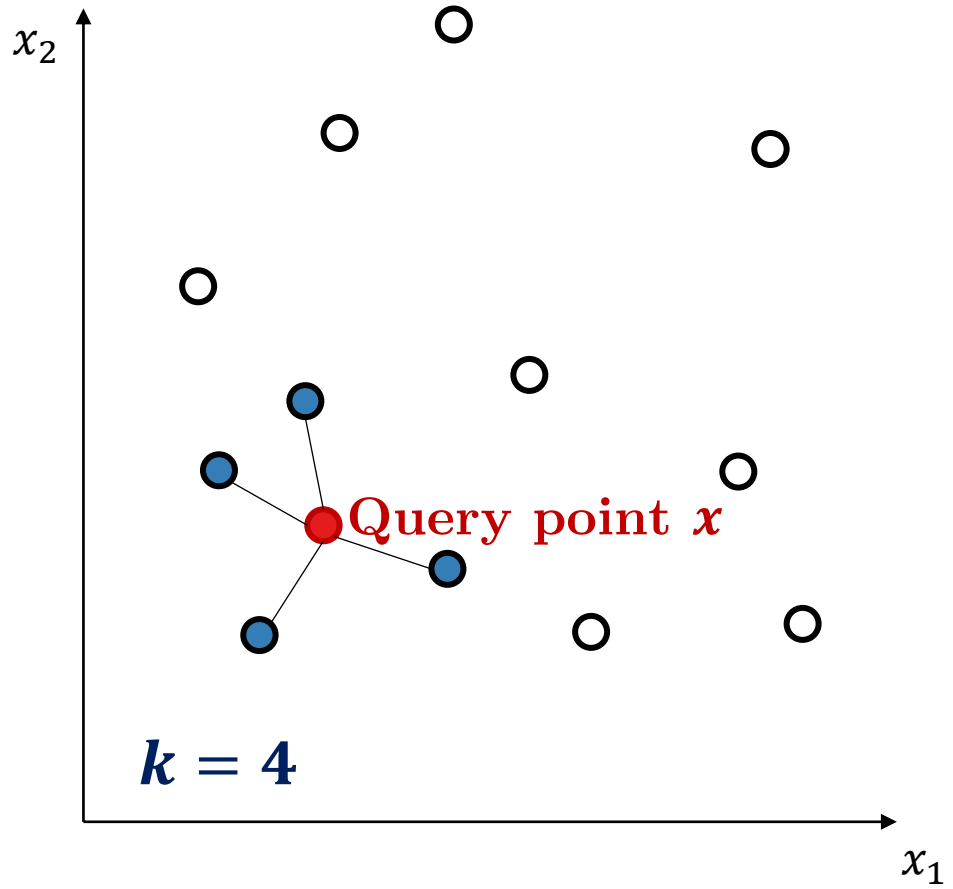
- Consider predicting COD reduction given DO_2 and HRT in an activated sludge pond



k-Nearest Neighbours

- Consider predicting COD reduction given DO_2 and HRT in an activated sludge pond
- k-nearest neighbours (kNN):
  - Given a query point x, find the set $\mathcal{S}$ of k nearest data points
  - Prediction is a weighted average over the set $\mathcal{S}$:

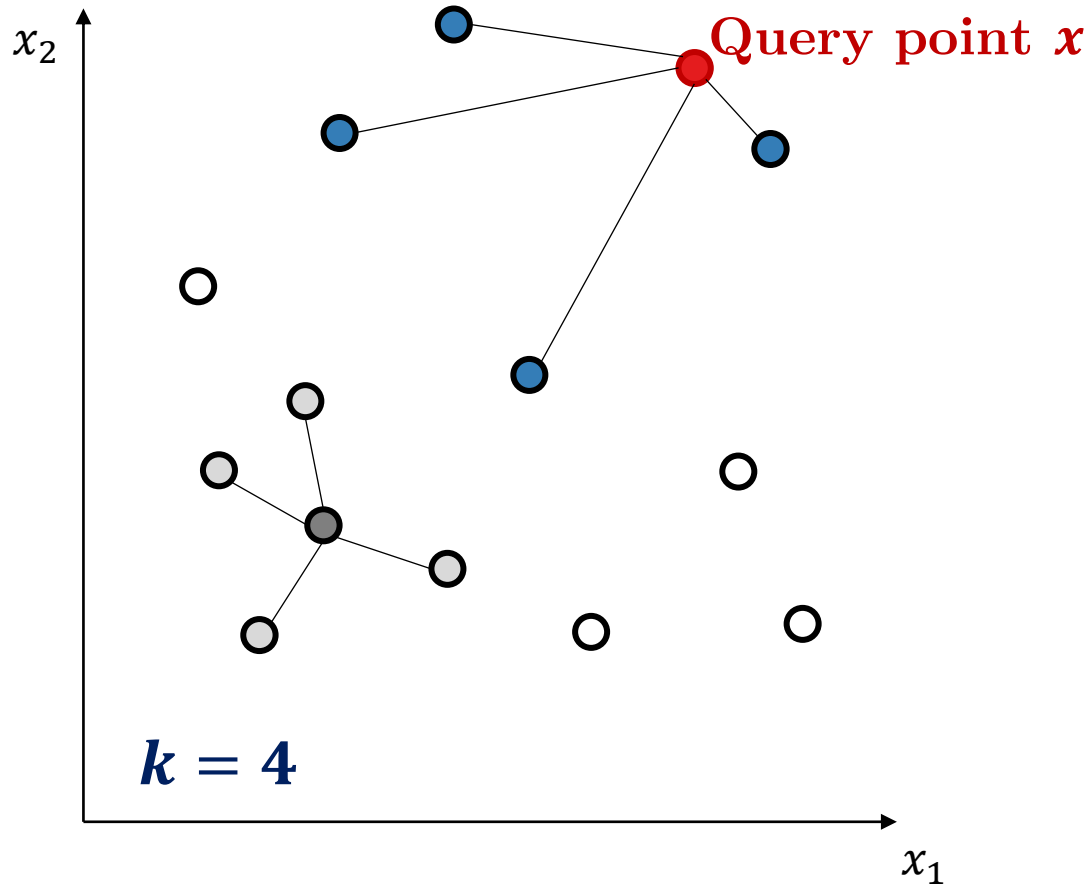
$$\hat{f}(x) = \sum_{j \in \mathcal{S}} w_j y_j$$



k-Nearest Neighbours

- Consider predicting COD reduction given DO_2 and HRT in an activated sludge pond
- k-nearest neighbours (kNN):
  - Given a query point x, find the set $\mathcal{S}$ of k nearest data points
  - Prediction is a weighted average over the set $\mathcal{S}$:

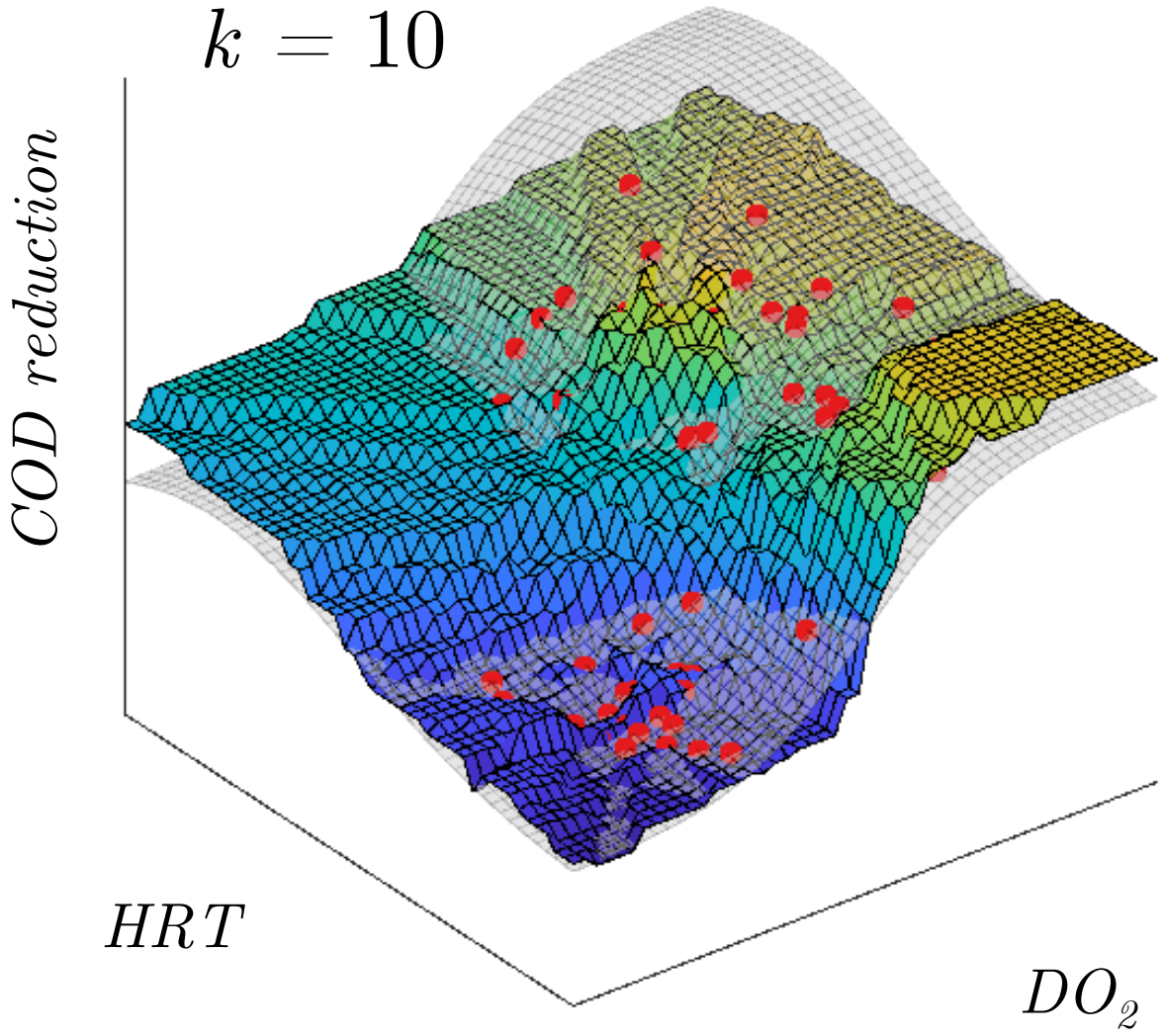
$$\hat{f}(x) = \sum_{j \in \mathcal{S}} w_j y_j$$



k-Nearest Neighbours

- Consider predicting COD reduction given DO_2 and HRT in an activated sludge pond
- k-nearest neighbours (kNN):
  - Given a query point x, find the set $\mathcal{S}$ of k nearest data points
  - Prediction is a weighted average over the set $\mathcal{S}$:

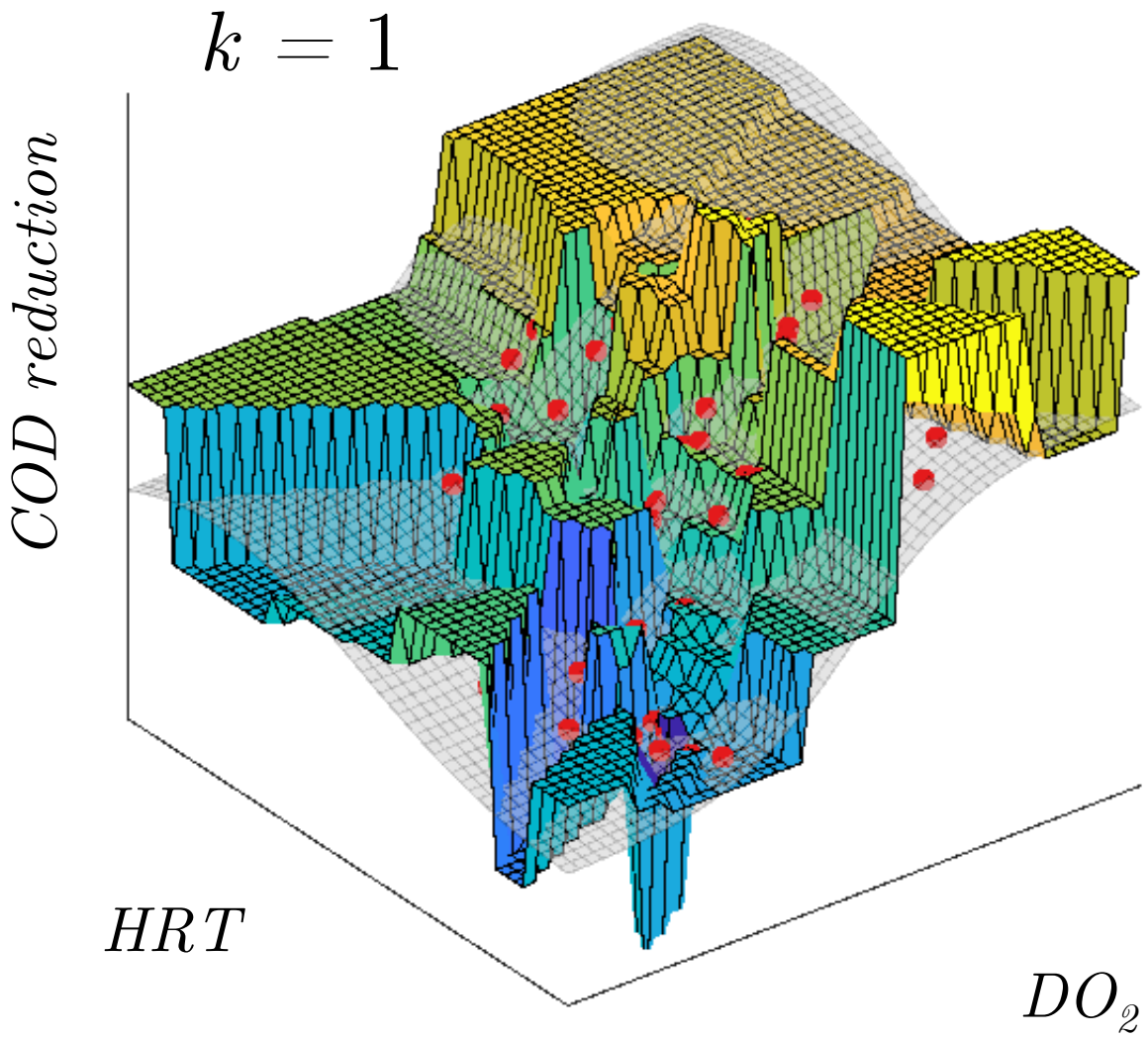
$$\hat{f}(x) = \sum_{j \in \mathcal{S}} w_j y_j$$



k-Nearest Neighbours

- Consider predicting COD reduction given DO_2 and HRT in an activated sludge pond
- k-nearest neighbours (kNN):
  - Given a query point x, find the set $\mathcal{S}$ of k nearest data points
  - Prediction is a weighted average over the set $\mathcal{S}$:

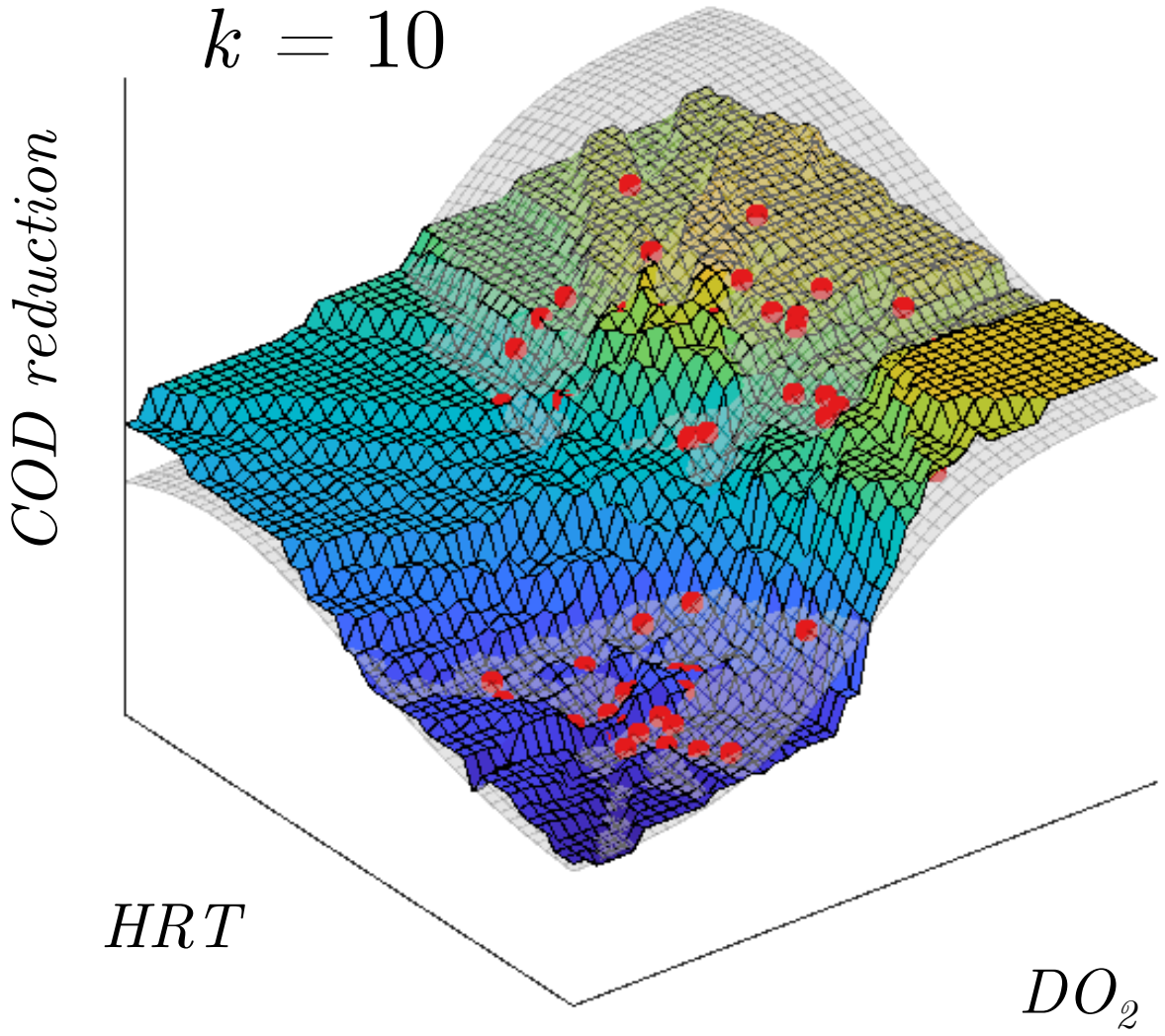
$$\hat{f}(x) = \sum_{j \in \mathcal{S}} w_j y_j$$



k-Nearest Neighbours

- Consider predicting COD reduction given DO_2 and HRT in an activated sludge pond
- k-nearest neighbours (kNN):
  - Given a query point x, find the set $\mathcal{S}$ of k nearest data points
  - Prediction is a weighted average over the set $\mathcal{S}$:

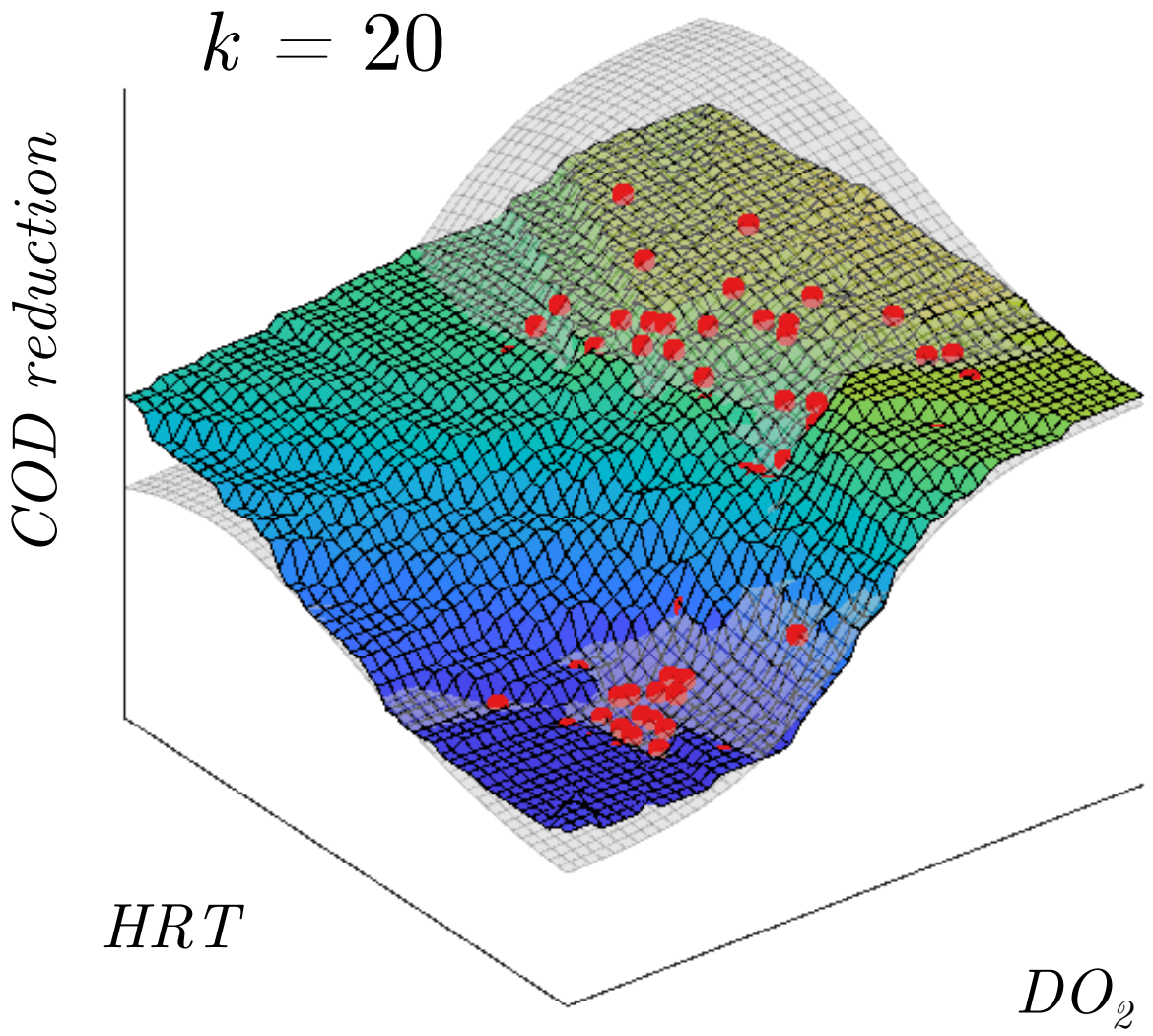
$$\hat{f}(x) = \sum_{j \in \mathcal{S}} w_j y_j$$



k-Nearest Neighbours

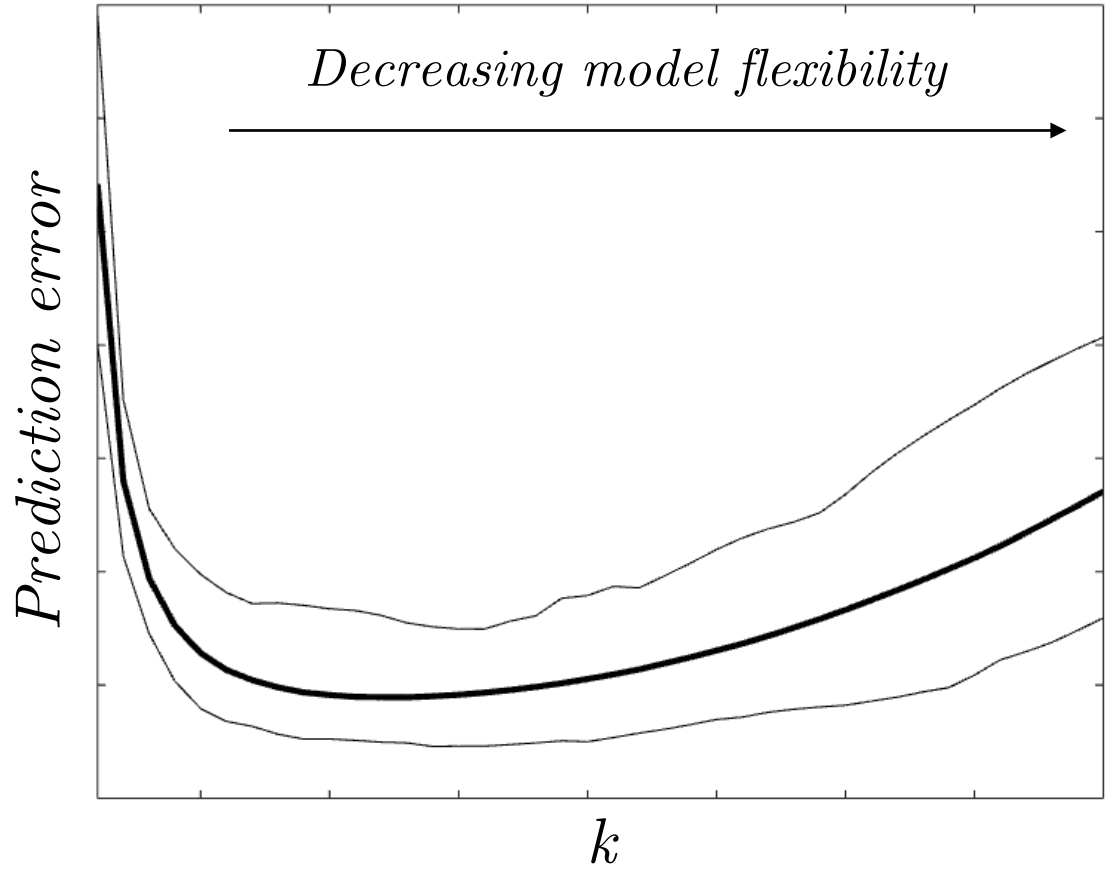
- Consider predicting COD reduction given DO_2 and HRT in an activated sludge pond
- k-nearest neighbours (kNN):
  - Given a query point x, find the set $\mathcal{S}$ of k nearest data points
  - Prediction is a weighted average over the set $\mathcal{S}$:

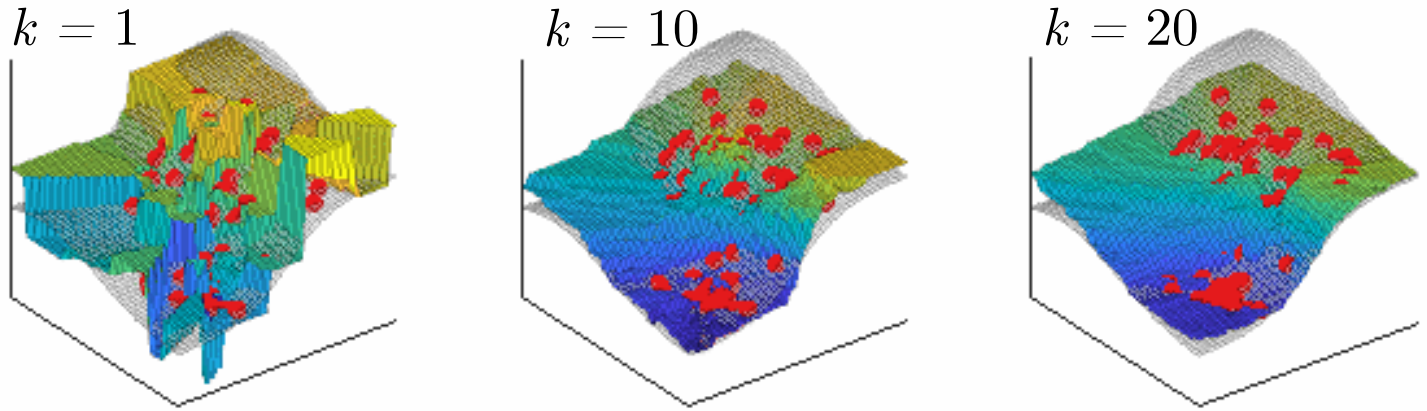
$$\hat{f}(x) = \sum_{j \in \mathcal{S}} w_j y_j$$



k-Nearest Neighbours

- Bias / variance controlled by k
- Small k
  - *Very flexible model*
  - *Varies dramatically with data*
- Large k
  - *Reduced model variability*
  - *Increased bias*



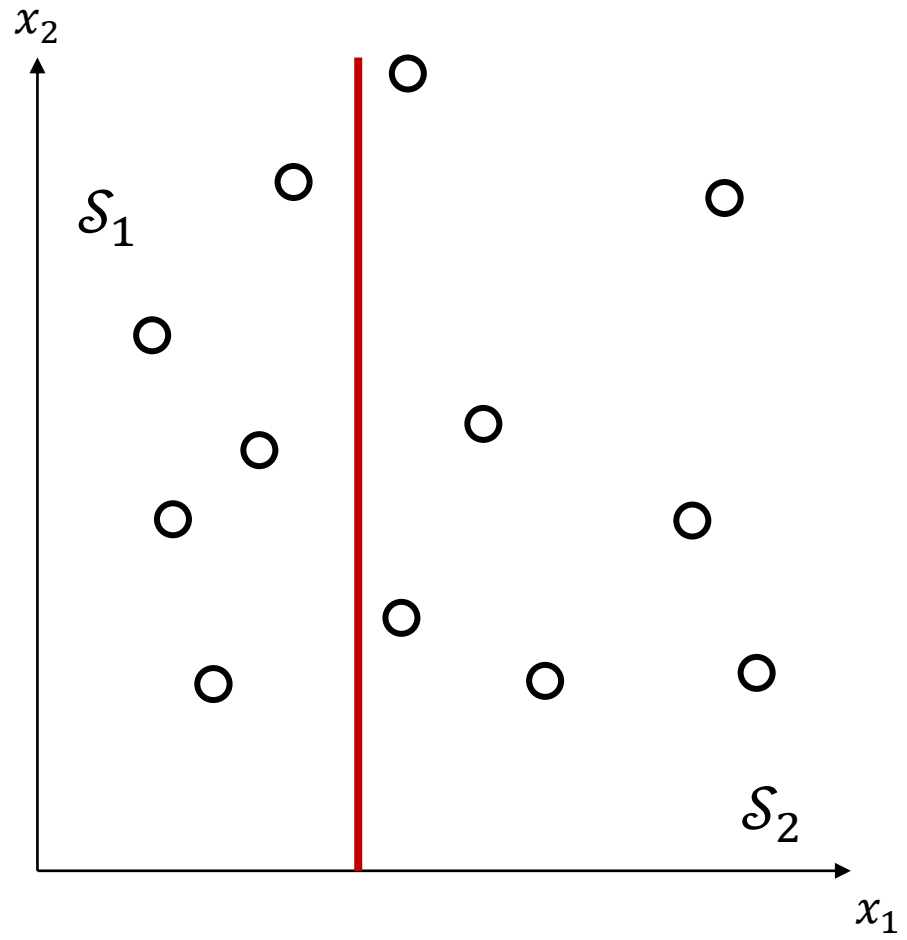


Decision trees

- Closely related to k-NN: decision trees
- Identify split in data:

$$\hat{f}(x \in \mathcal{S}_1) = \frac{1}{n_1} \sum_{j \in \mathcal{S}_1} y_j$$

$$\hat{f}(x \in \mathcal{S}_2) = \frac{1}{n_2} \sum_{j \in \mathcal{S}_2} y_j$$

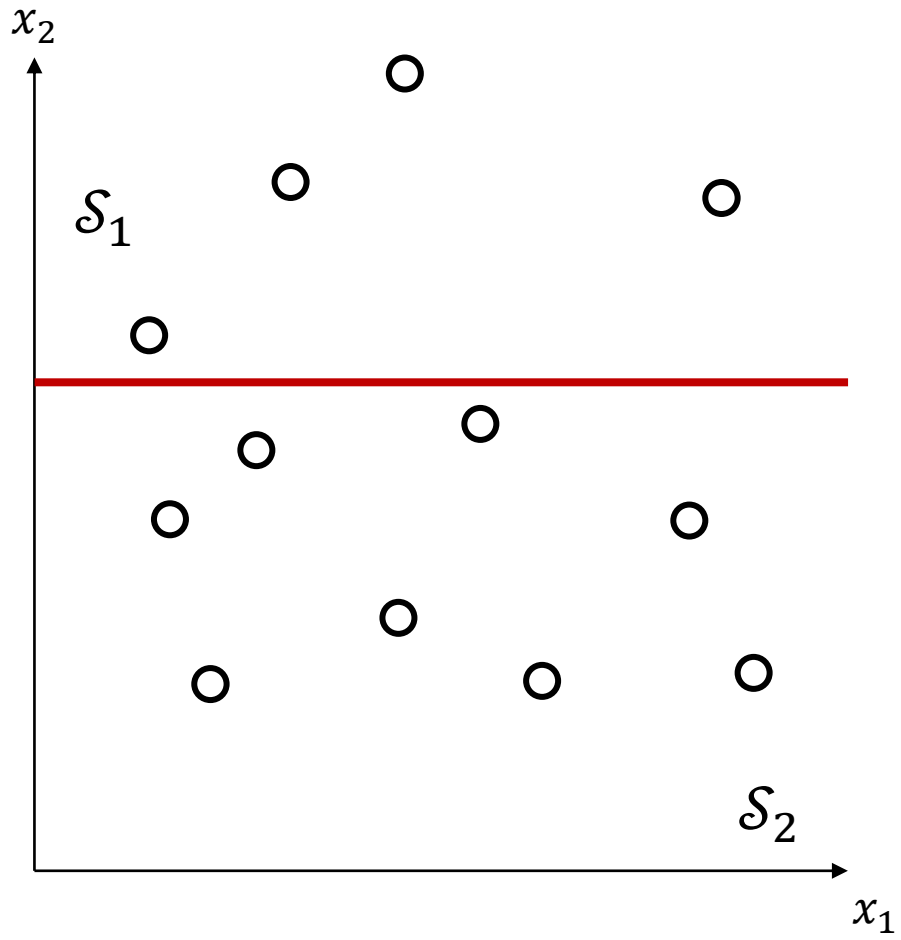


Decision trees

- Closely related to k-NN: decision trees
- Identify split in data:

$$\hat{f}(x \in \mathcal{S}_i) = \frac{1}{n_i} \sum_{j \in \mathcal{S}_i} y_j$$

- Split is *greedy*, i.e., best possible split to minimize error

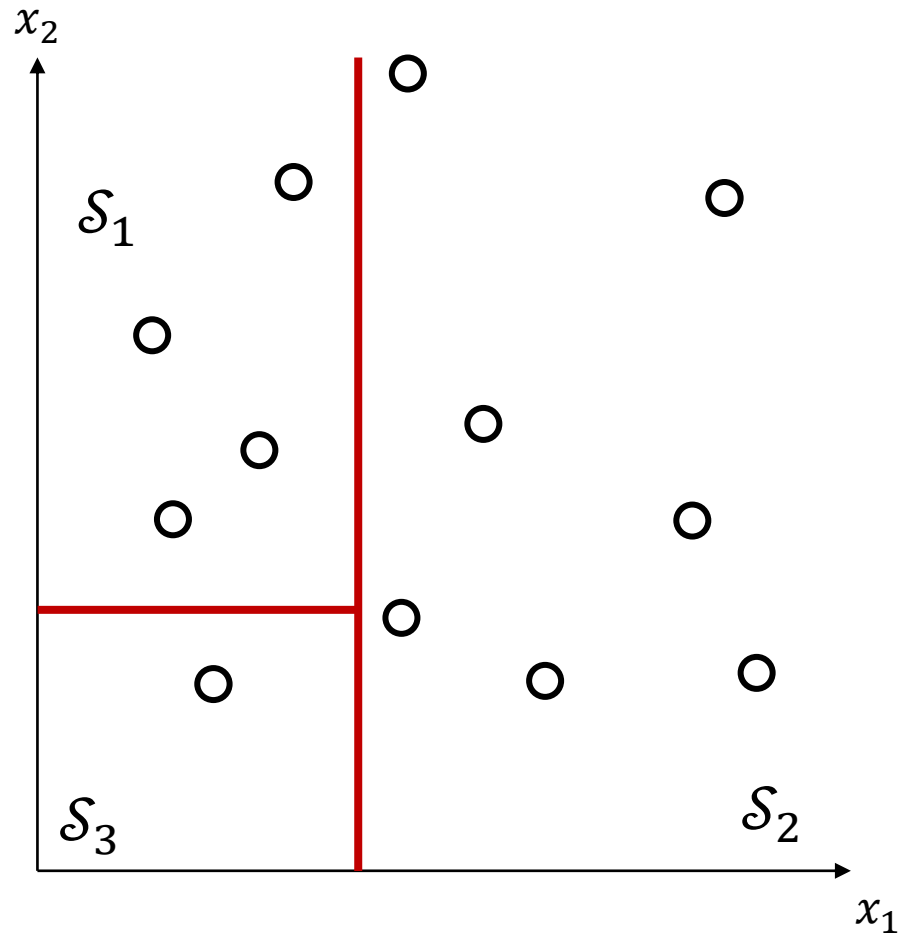


Decision trees

- Closely related to k-NN: decision trees
- Identify split in data:

$$\hat{f}(x \in \mathcal{S}_i) = \frac{1}{n_i} \sum_{j \in \mathcal{S}_i} y_j$$

- Split is *greedy*, i.e., best possible split to minimize error

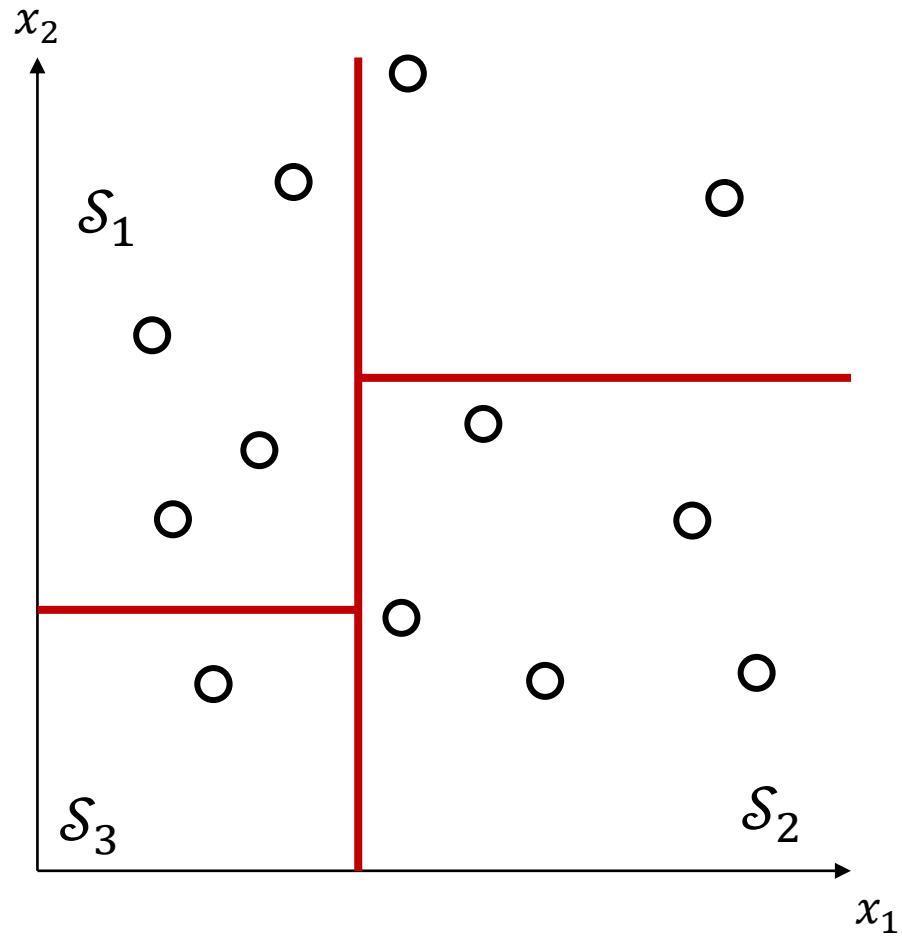


Decision trees

- Closely related to k-NN: decision trees
- Identify split in data:

$$\hat{f}(x \in \mathcal{S}_i) = \frac{1}{n_i} \sum_{j \in \mathcal{S}_i} y_j$$

- Split is *greedy*, i.e., best possible split to minimize error

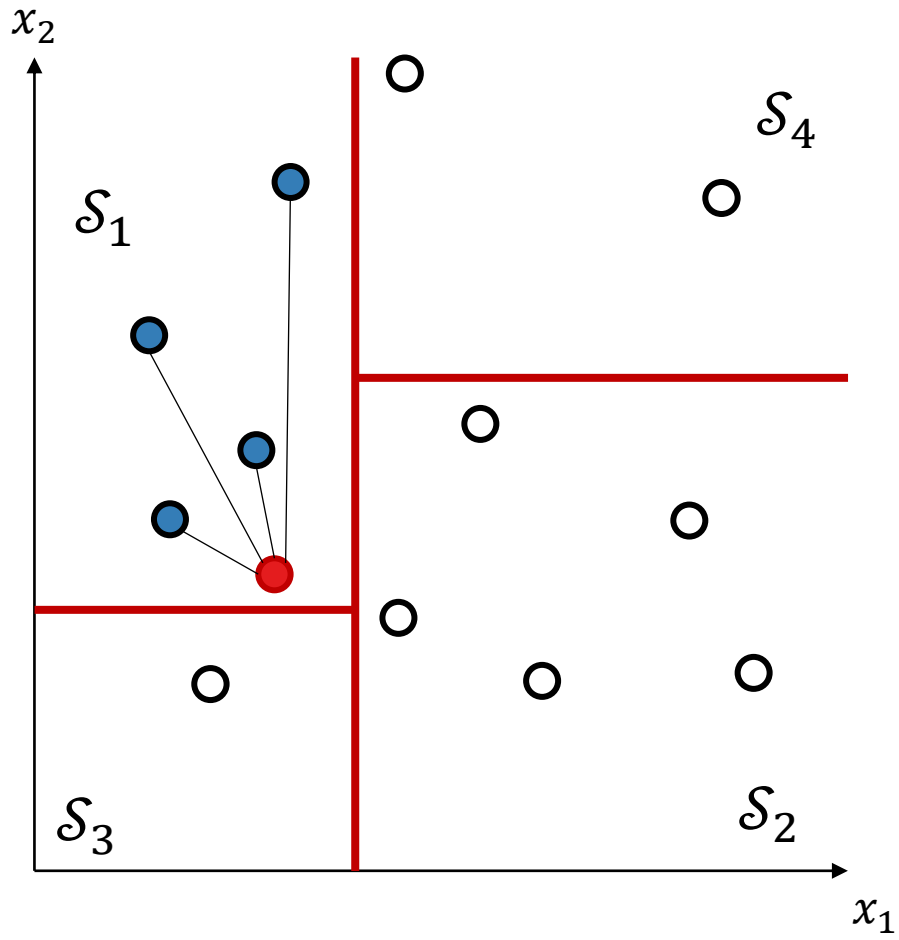


Decision trees

- Closely related to k-NN: decision trees
- Identify split in data:

$$\hat{f}(x \in \mathcal{S}_i) = \frac{1}{n_i} \sum_{j \in \mathcal{S}_i} y_j$$

- Split is *greedy*, i.e., best possible split to minimize error

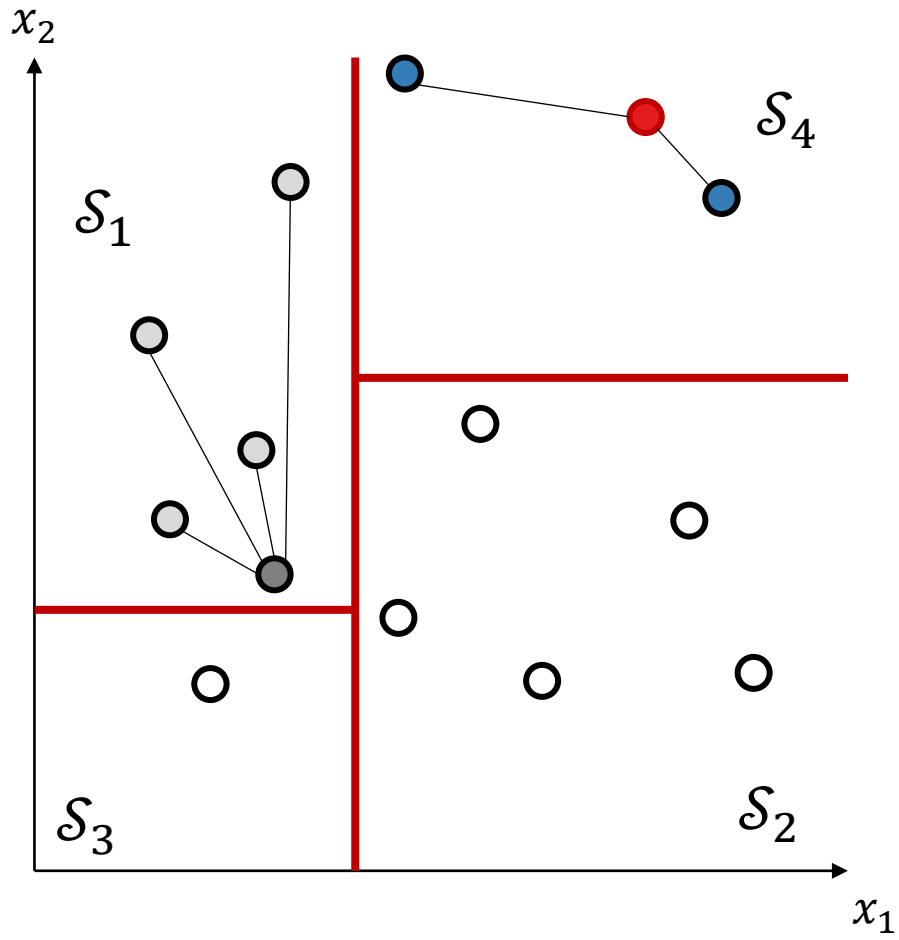


Decision trees

- Closely related to k-NN: decision trees
- Identify split in data:

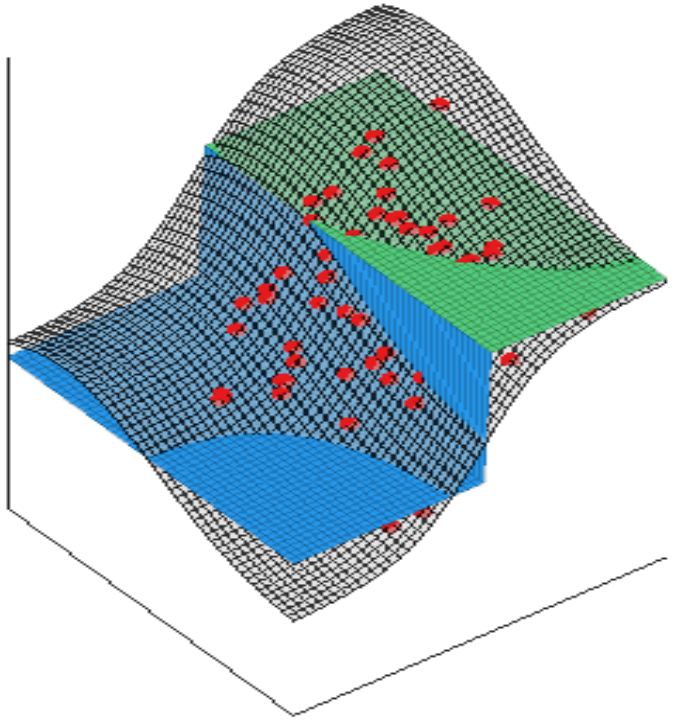
$$\hat{f}(x \in \mathcal{S}_i) = \frac{1}{n_i} \sum_{j \in \mathcal{S}_i} y_j$$

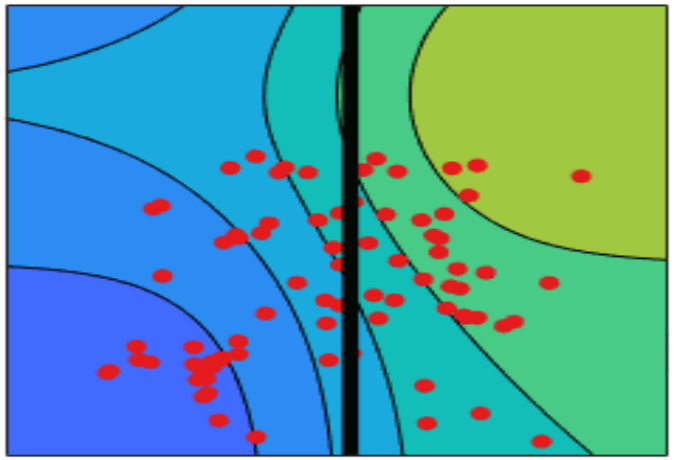
- Split is *greedy*, i.e., best possible split to minimize error



Decision trees

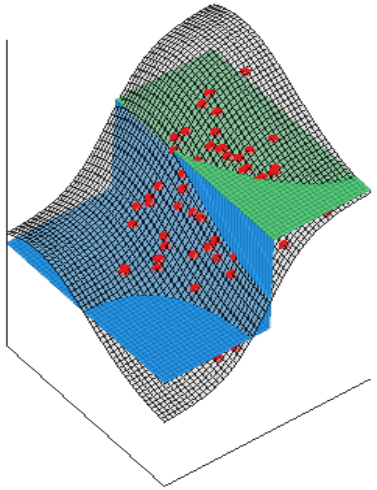
- Common approaches to controlling bias/variance:
  - Minimum observations per *leaf node* (i.e., min. elements per set S_i)
  - Maximum number of splits

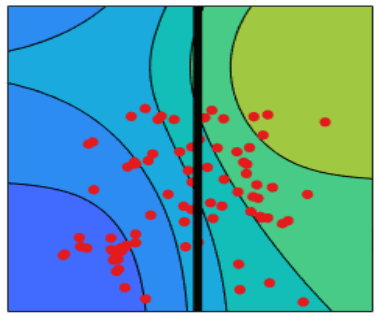


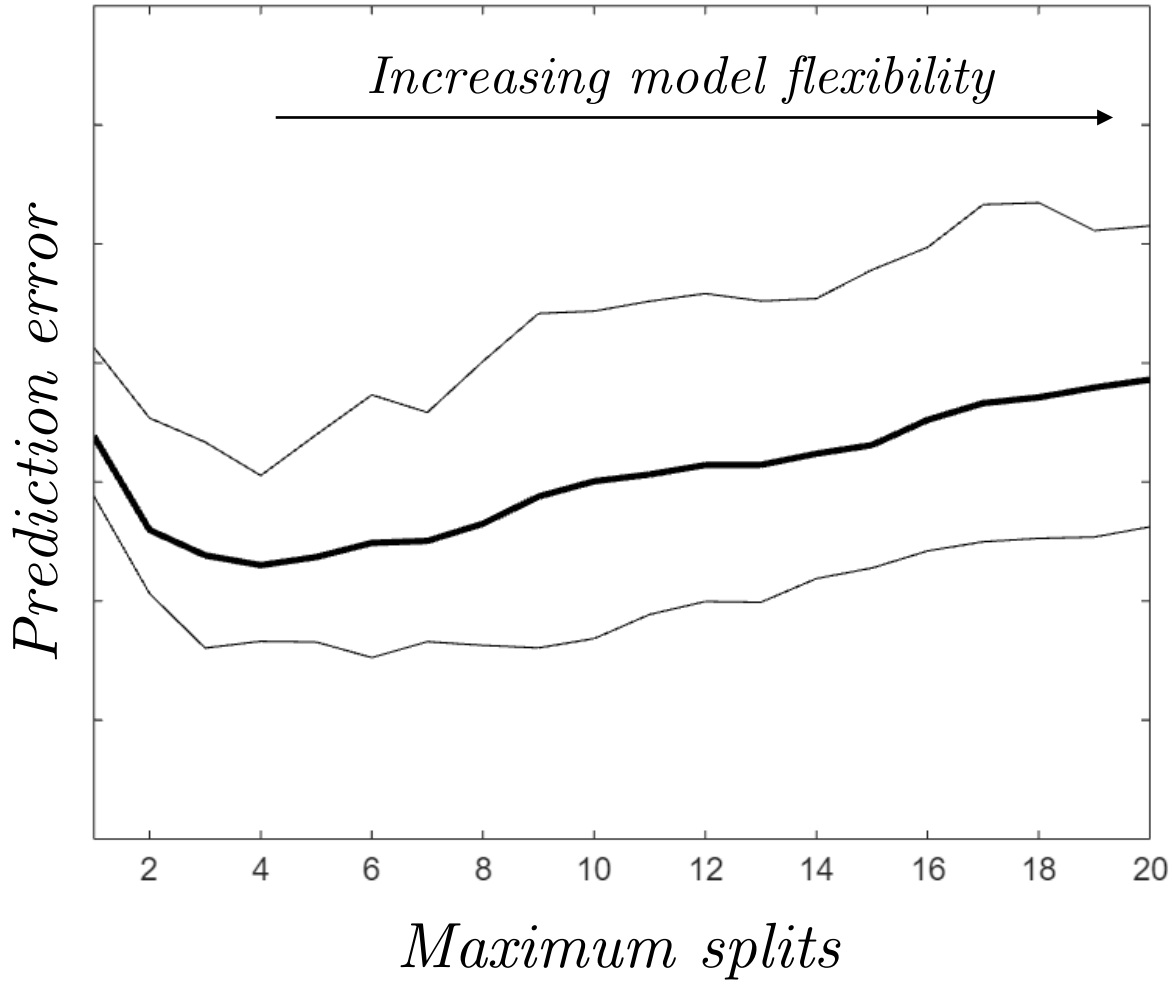


Decision trees

- Common approaches to controlling bias/variance:
  - Minimum observations per *leaf node* (i.e., min. elements per set $\mathcal{S}_i$)
  - Maximum number of splits

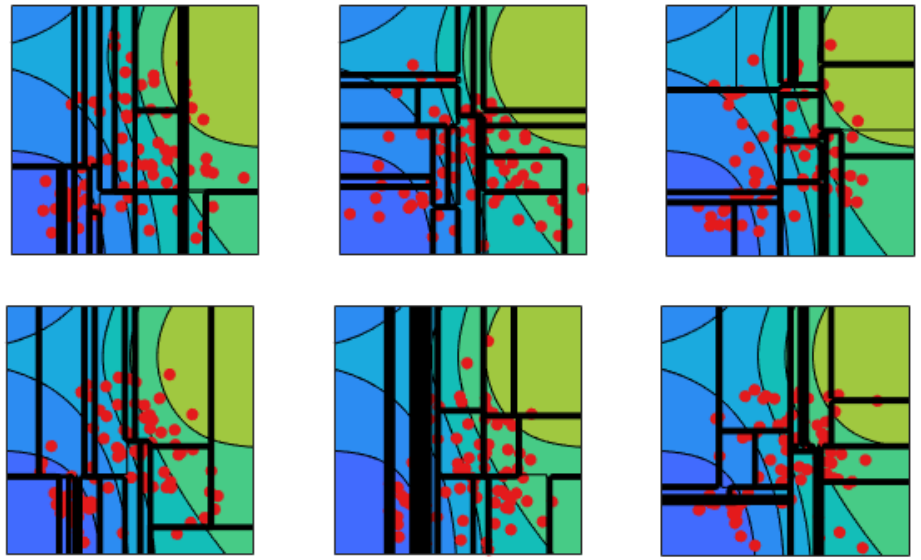






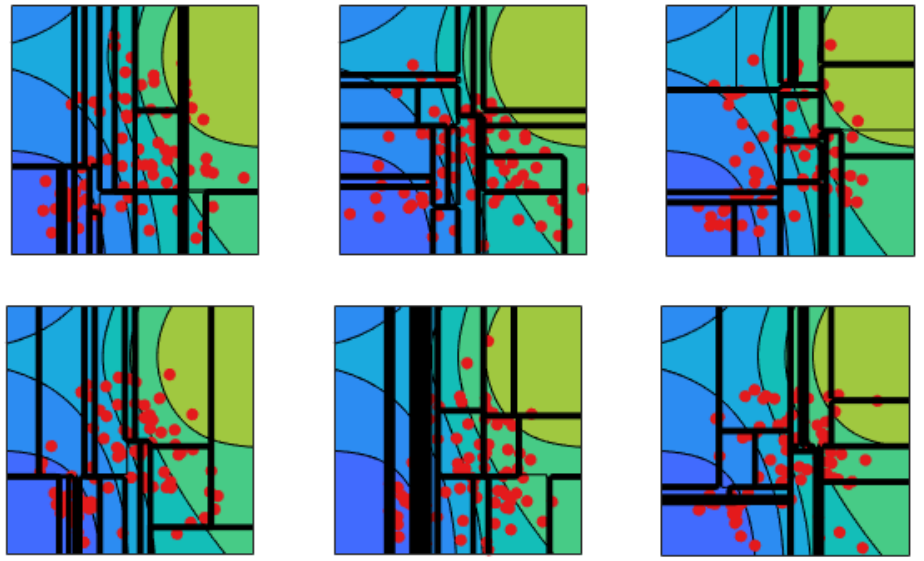
Bagged decision trees

- Decision trees w/ many splits are very flexible:
  high variance, low bias



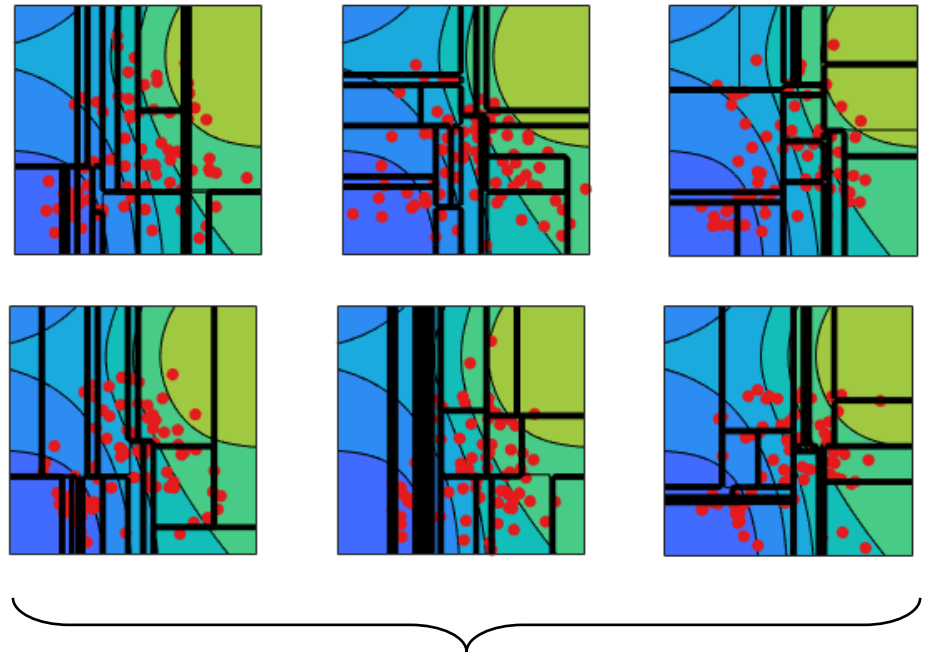
Bagged decision trees

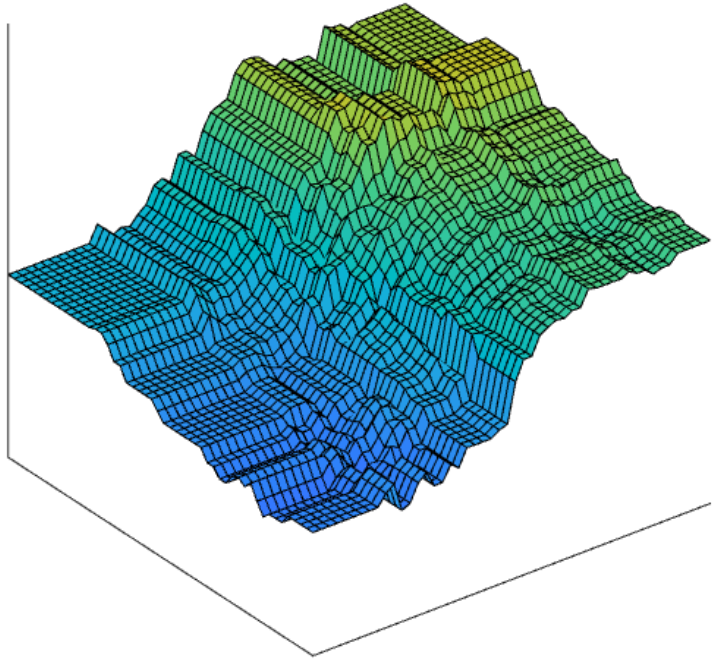
- Decision trees w/ many splits are very flexible:
  high variance, low bias
- Reduce variance by ave. over multiple, slightly different trees
  - "Generate" multiple data sets by bootstrapping
  - Train individual decision trees on bootstrapped data



Bagged decision trees

- Decision trees w/ many splits are very flexible:
  high variance, low bias
- Reduce variance by ave. over multiple, slightly different trees
  - "Generate" multiple data sets by bootstrapping
  - Train individual decision trees on bootstrapped data
  - Average over individual model predictions (aggregate models)
  - Bootstrap + aggregation = *Bagging*





Random forests

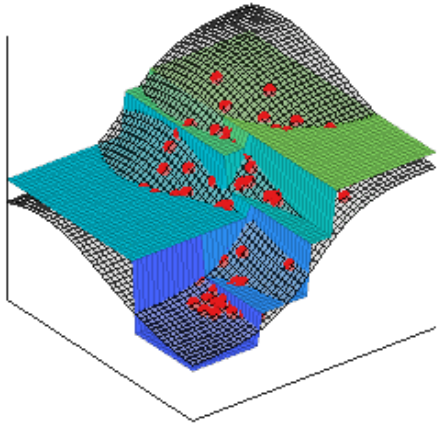
- Trees in bagged forest remain correlated,
  significant splits consistently selected
- Random forest:
  each split selected from subset of inputs

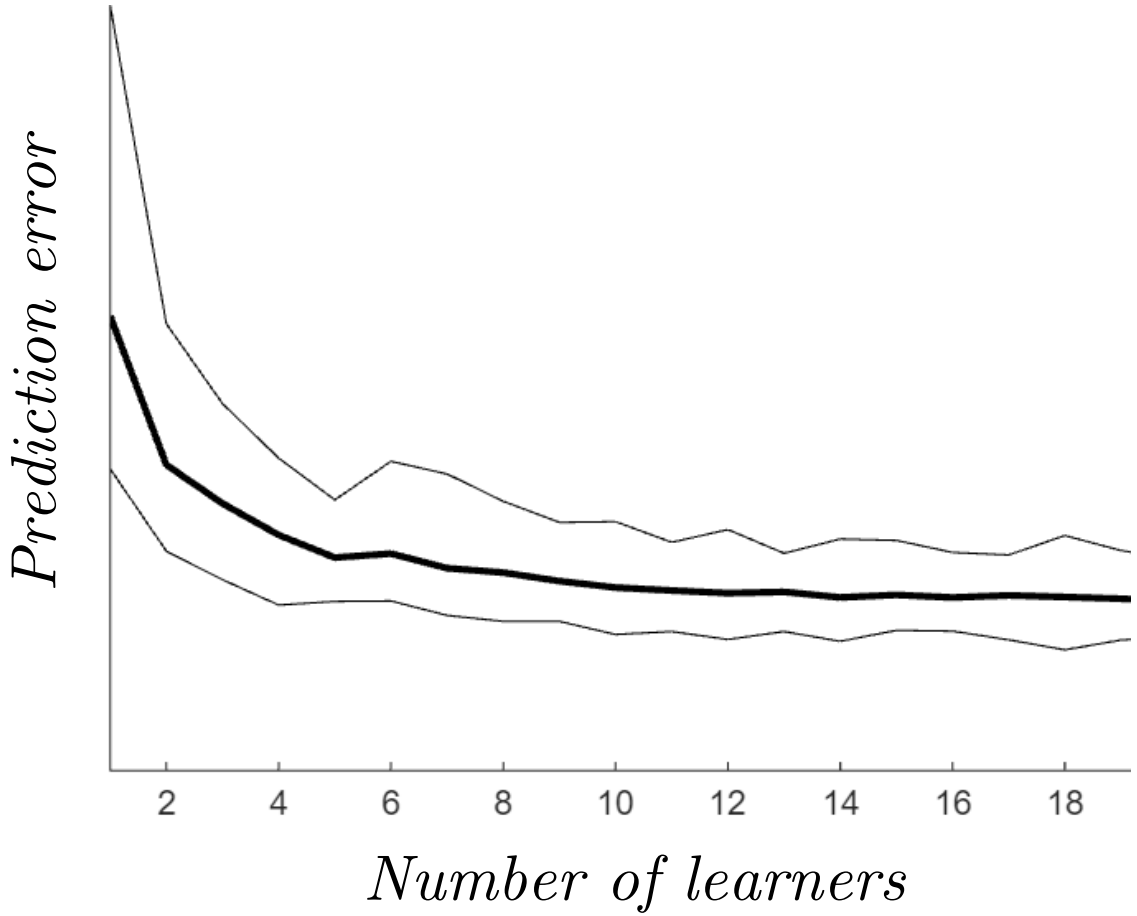
  - Given input variables $x^{(1)}, x^{(2)}, x^{(3)}$
  - Decision trees:
    best split along any of the inputs $x^{(1)}, x^{(2)}, x^{(3)}$
  - Random forest:
    best split along a random subset of inputs, e.g., $x^{(1)}, x^{(3)}$

Random forests

- Bagging decorrelated trees
  reduces variances
  maintains flexibility
  (no increase in bias)
- Prediction error decreases as number of learners increase

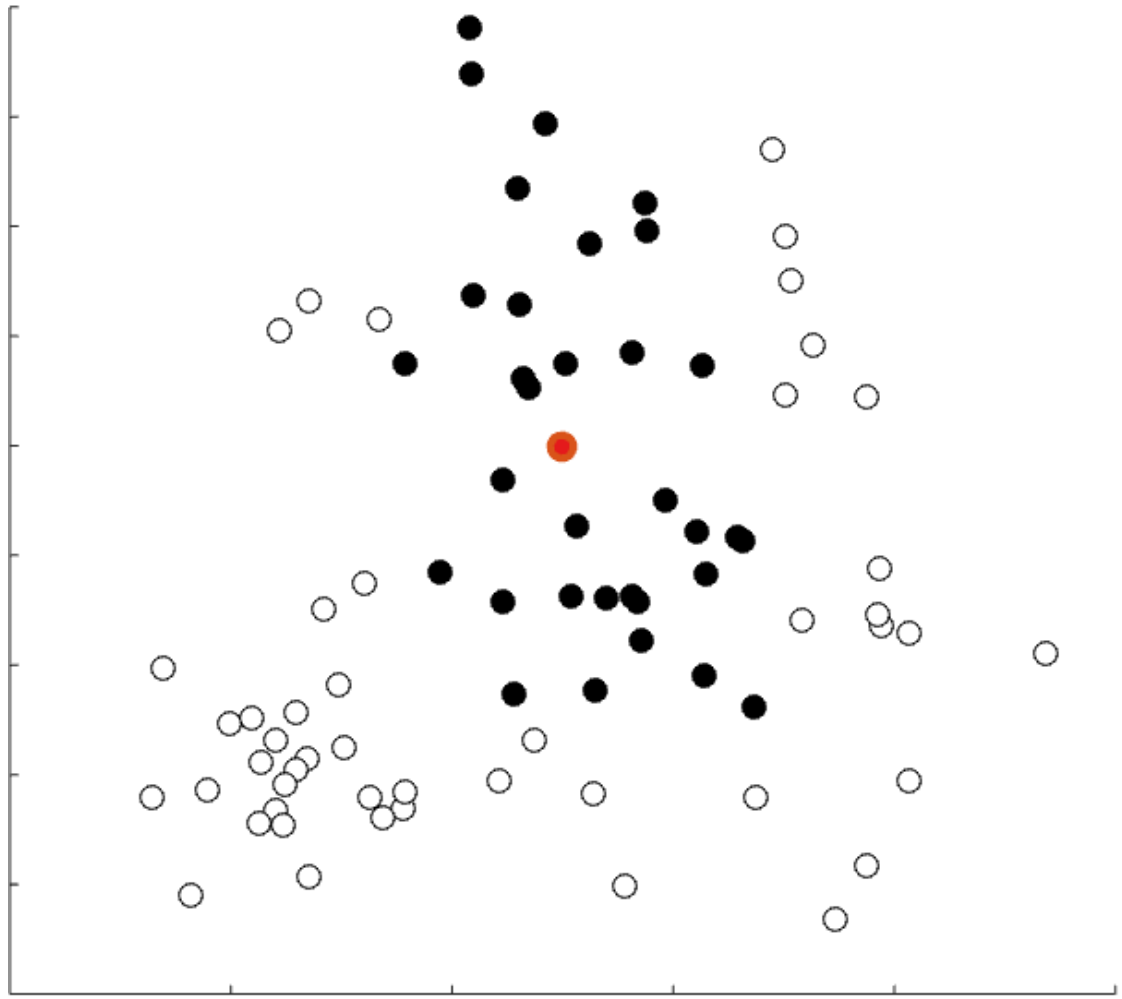
Ensemble of weak learners





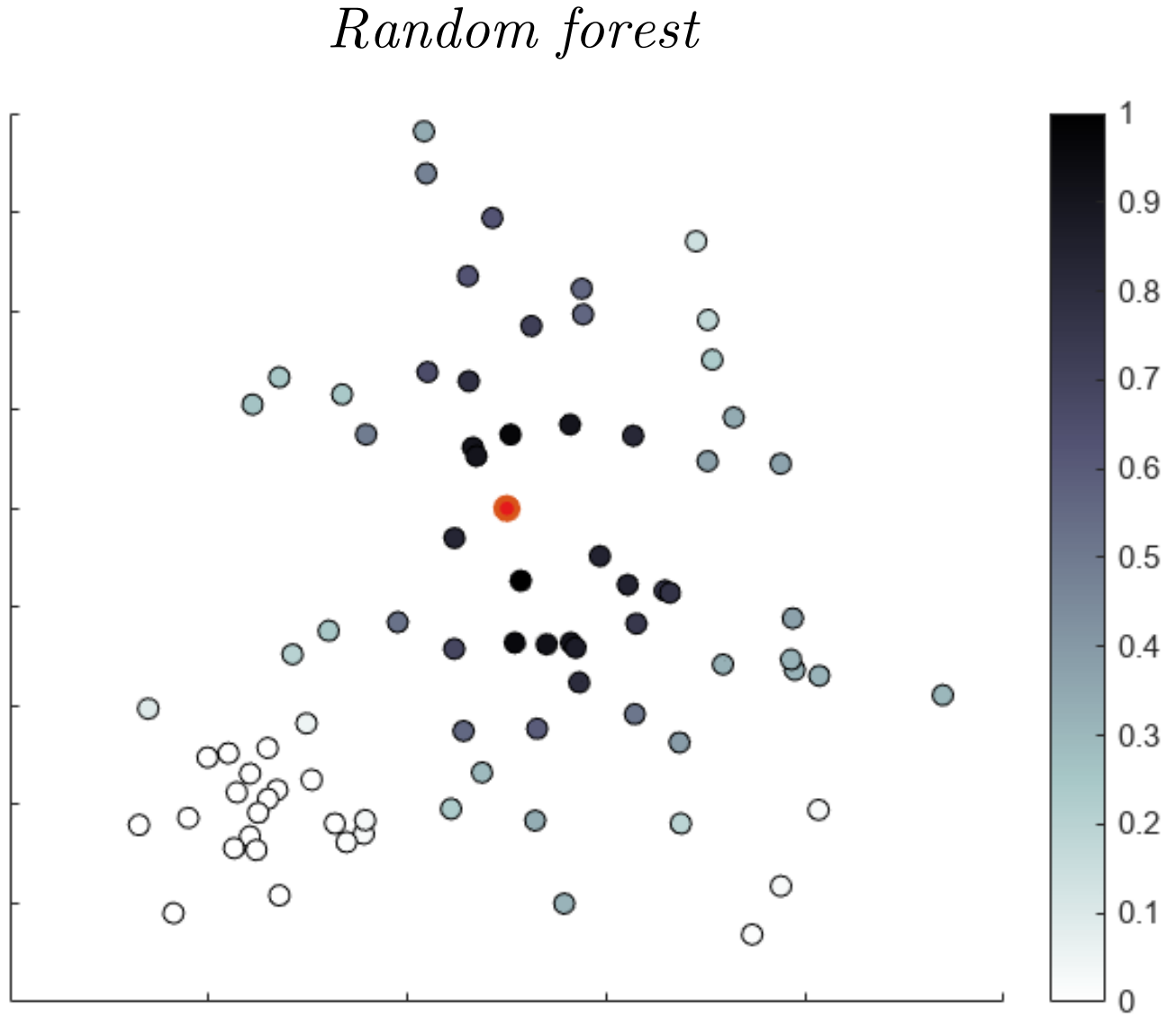
Random forests

- k-NN, decision trees: average over subset
- Random forests: weighted average over larger subset



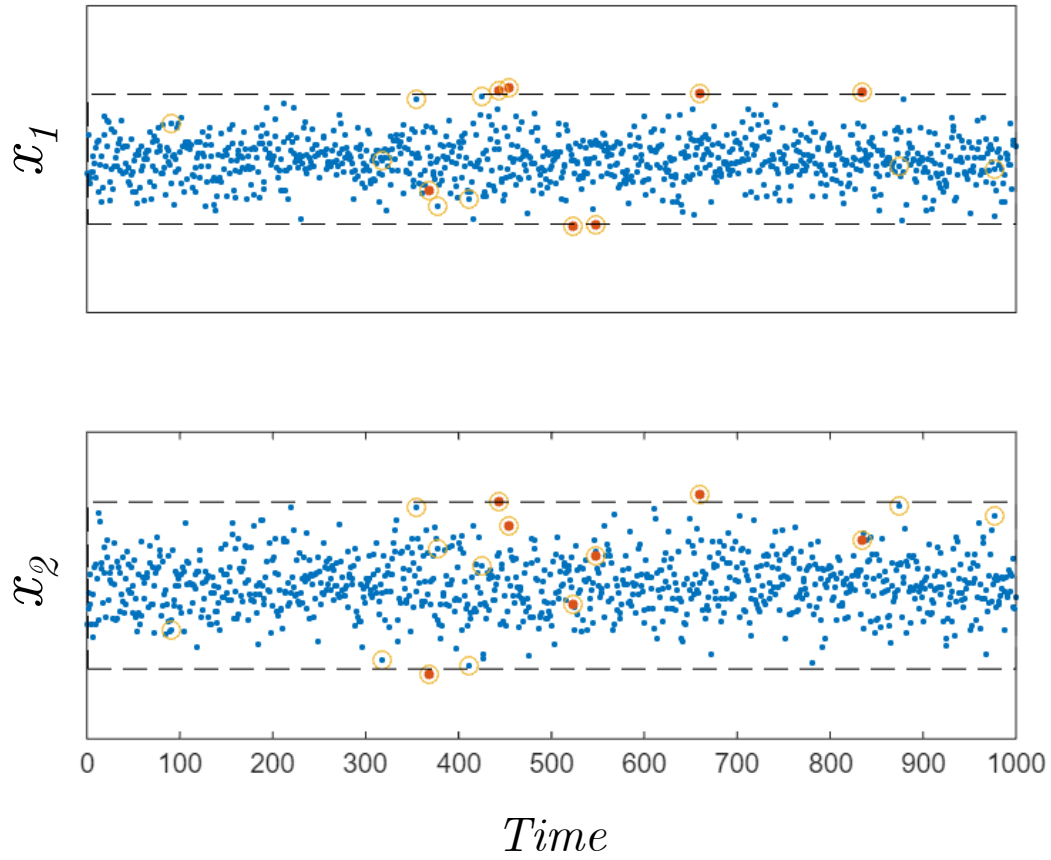
Random forests

- k-NN, decision trees: average over subset
- Random forests: weighted average over larger subset

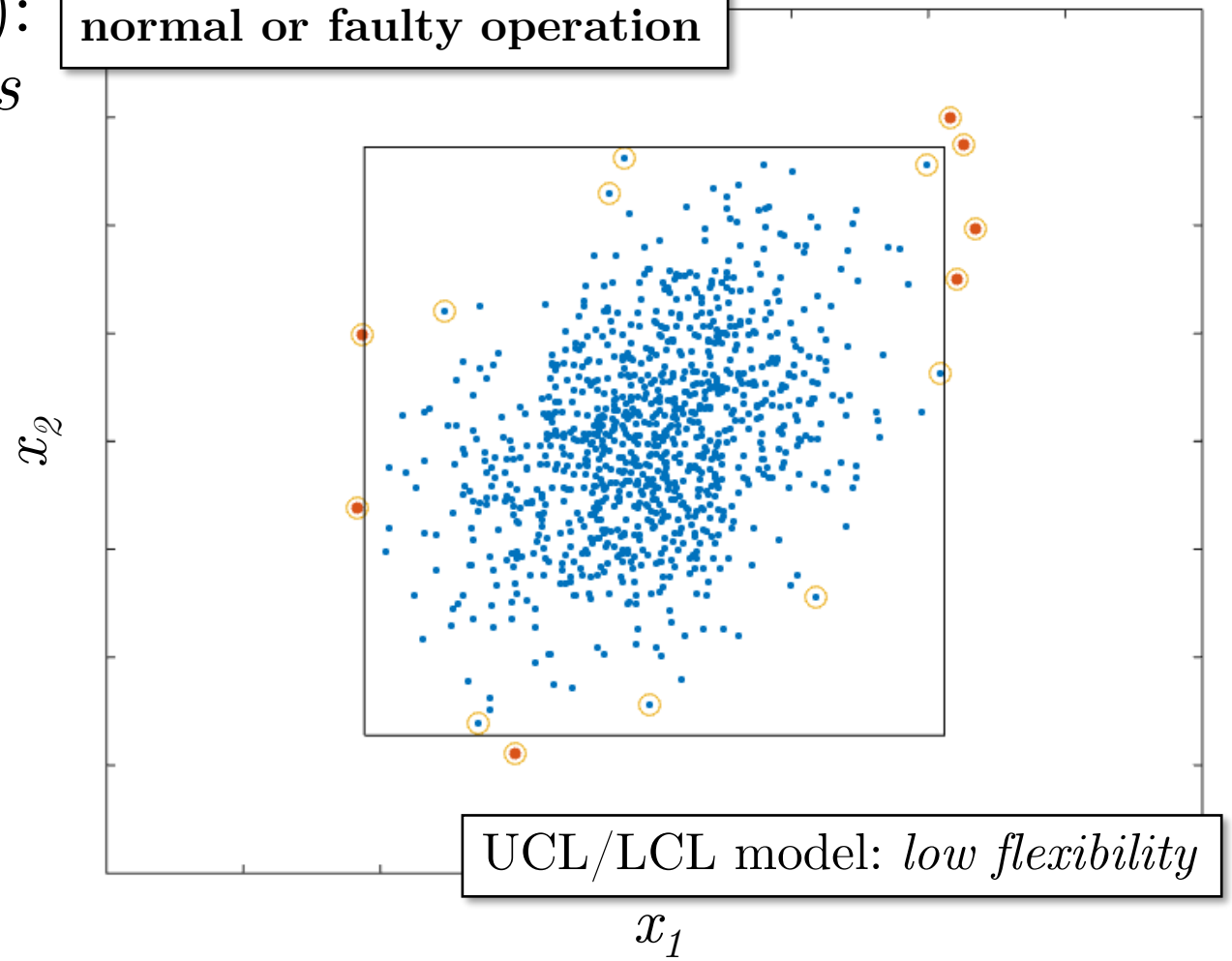


Application to statistical process control

- Process monitoring (2 variables):
  upper- and lower control limits



Classification model for normal or faulty operation

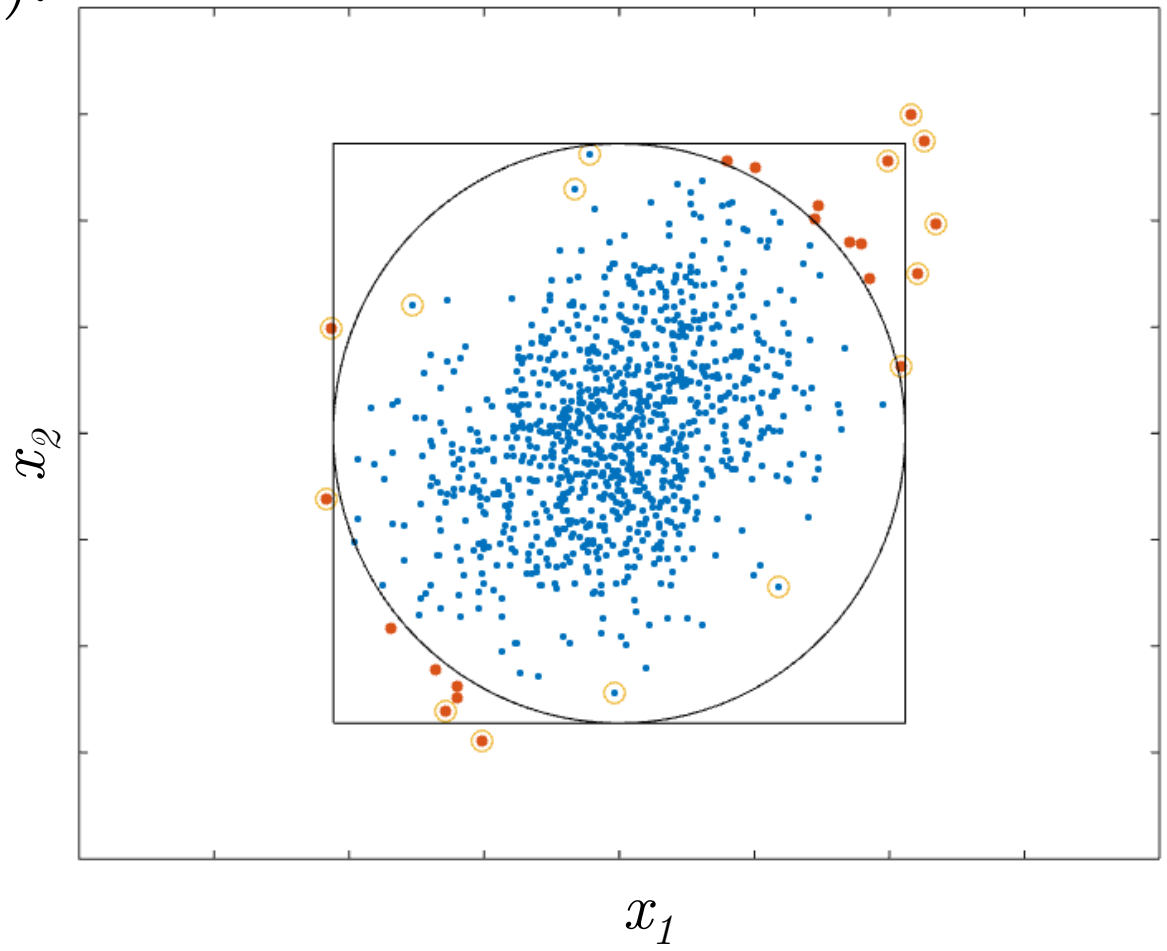


UCL/LCL model: *low flexibility*

Application to statistical process control

- Process monitoring (2 variables):
  upper- and lower control limits
  (2 parameters)
- Fit normal distribution,
  w/o covariance
  (2 parameters)

$$\Sigma = \begin{bmatrix} \sigma_{11} & 0 \\ 0 & \sigma_{22} \end{bmatrix}$$

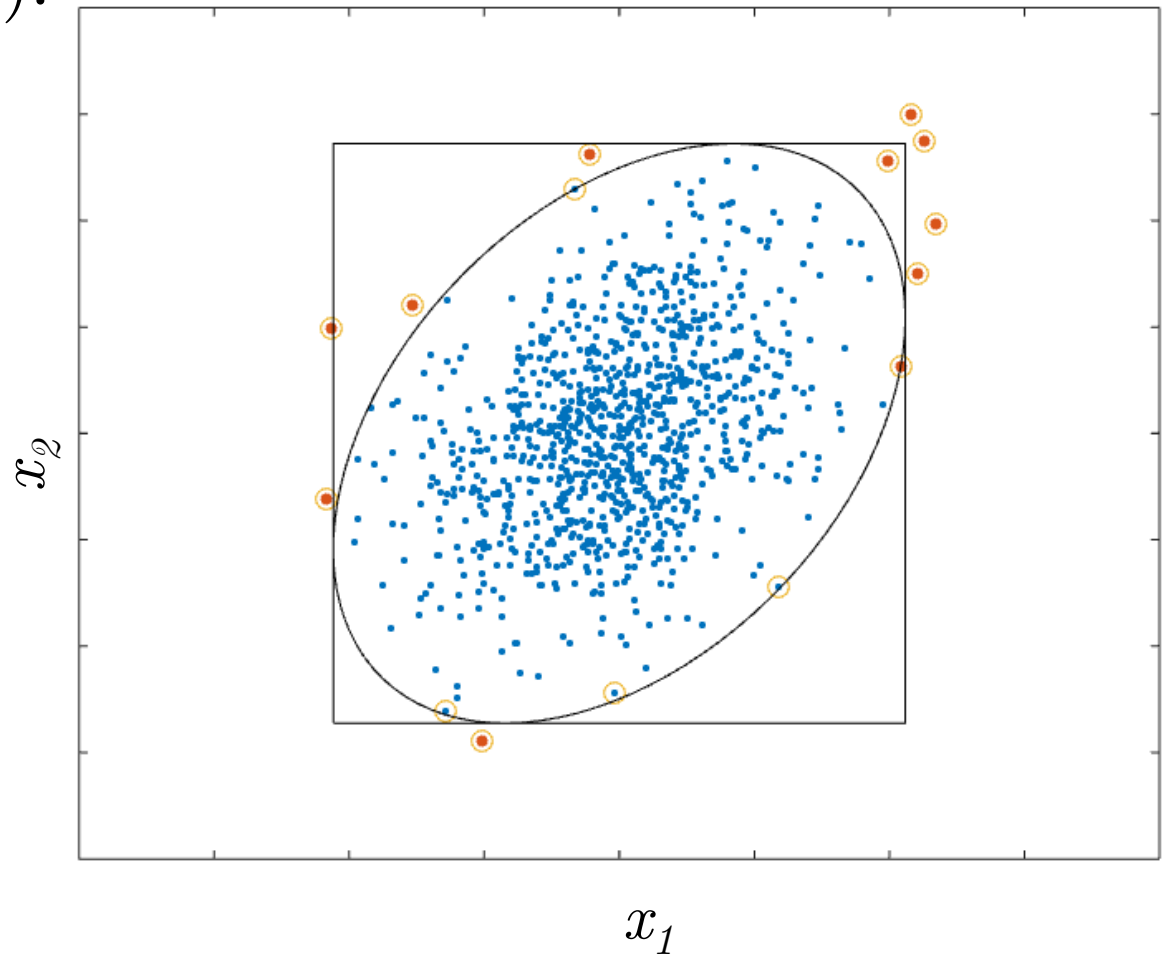


Application to statistical process control

- Process monitoring (2 variables):
  upper- and lower control limits
  (2 parameters)
- Fit normal distribution,
  w/ covariance
  (3 parameters)

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix}$$

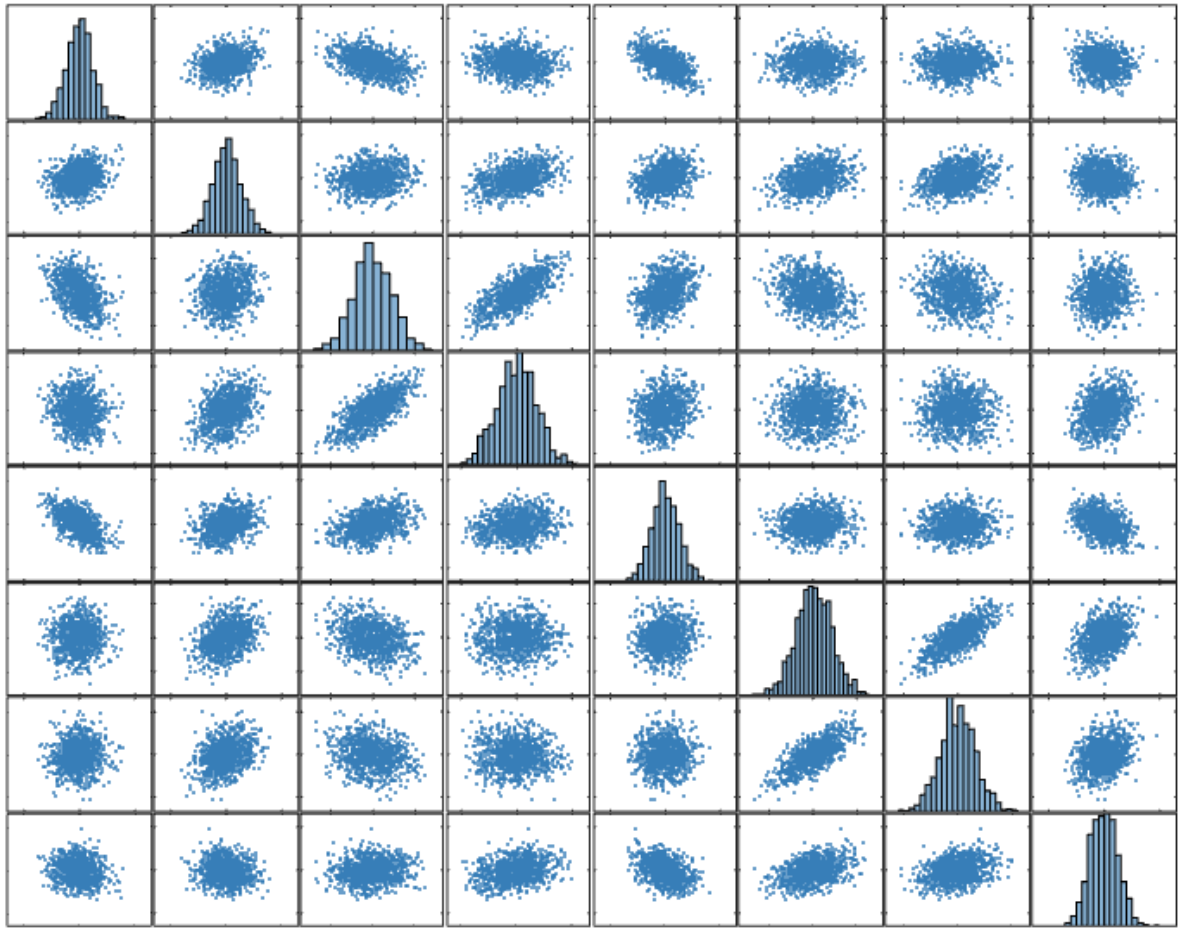
- Equivalent to using
  Hotelling's T^2-statistic



Application to statistical process control

- Process monitoring (8 variables):
  upper- and lower control limits
  (8 parameters)
- Fit normal distribution,
  w/ covariance
  (36 parameters)

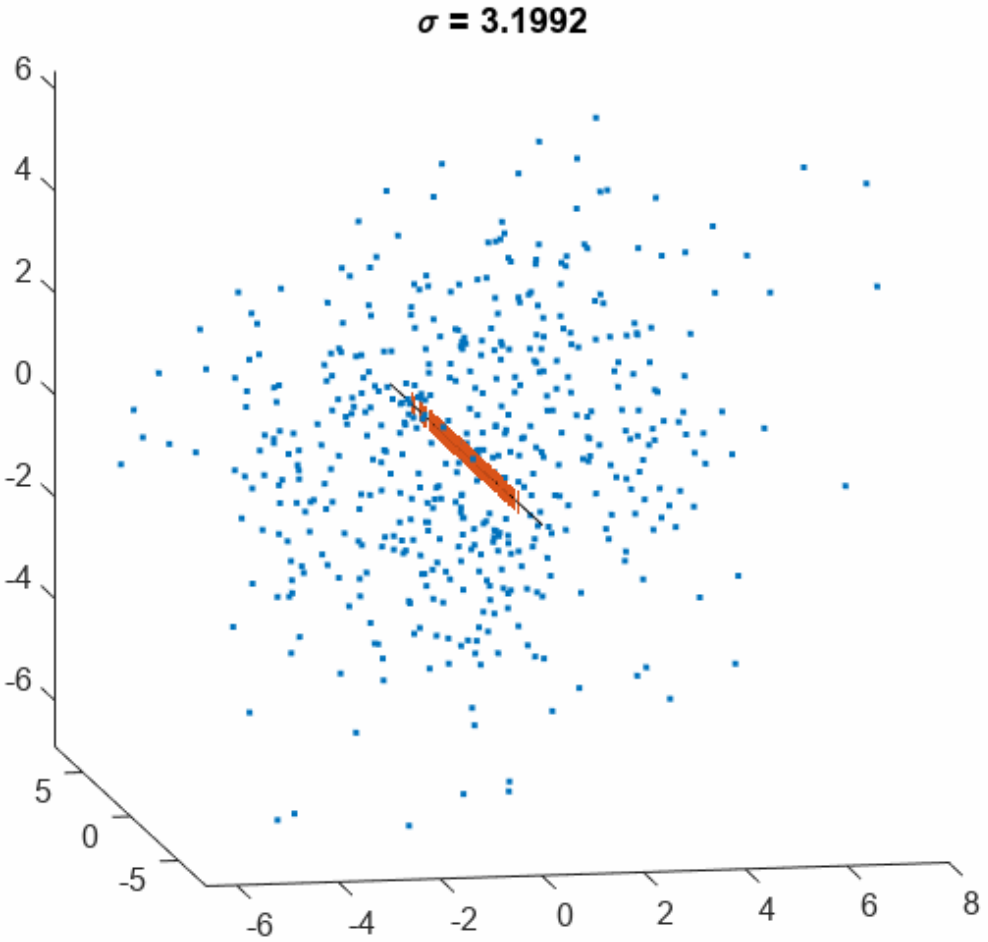
$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{18} \\ \sigma_{12} & \sigma_{22} & \cdots & \sigma_{28} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{18} & \sigma_{28} & \cdots & \sigma_{88} \end{bmatrix}$$



Principal Component Analysis (PCA)

- Unsupervised learning
  no response variable
- Identify direction $\mathbf{q}_1$:
  $$\mathbf{q}_1 \leftarrow \max\{\text{var}(\mathbf{x} \cdot \mathbf{q})\}$$
- Scores = values of $t = \mathbf{x} \cdot \mathbf{q}_1$

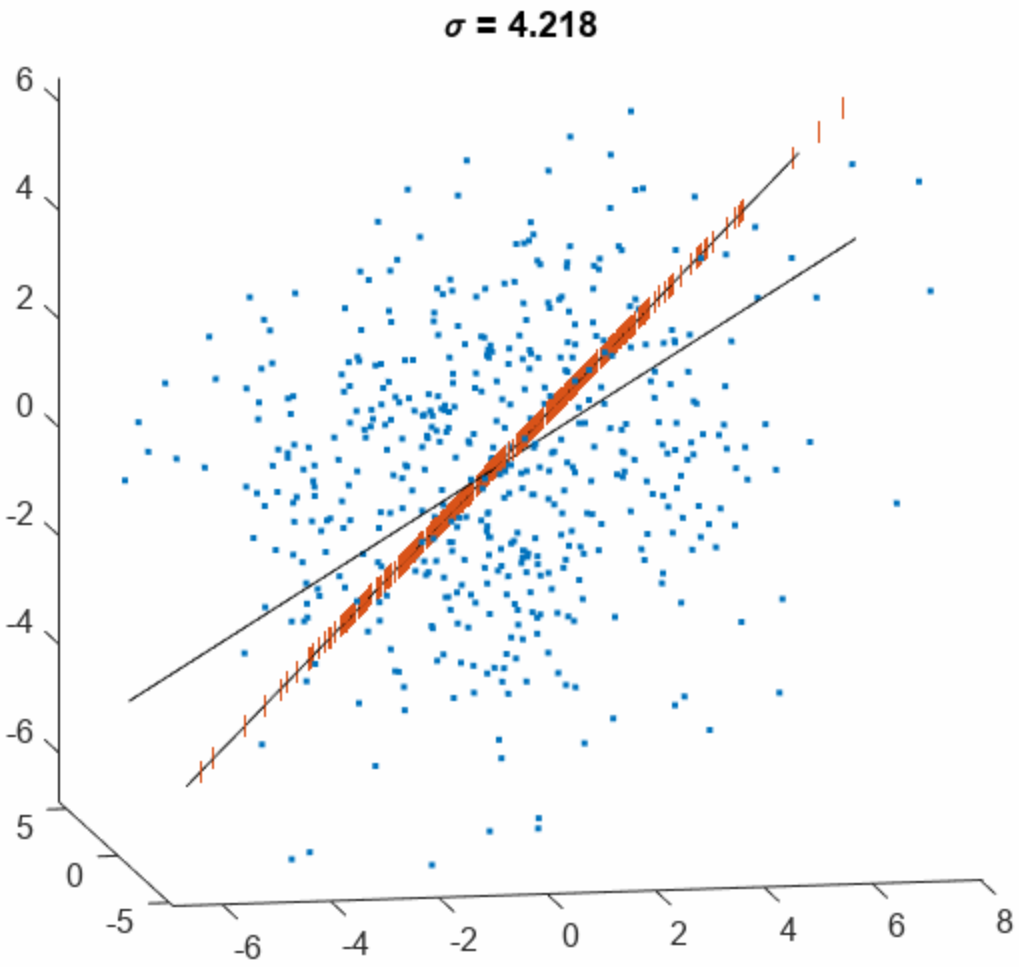
Principal component 1



Principal Component Analysis (PCA)

- Unsupervised learning
  no response variable
- Identify direction $\mathbf{q}_1$:
  $$\mathbf{q}_1 \leftarrow \max\{\text{var}(\mathbf{x} \cdot \mathbf{q})\}$$
- Identify direction $\mathbf{q}_2$:
  $$\mathbf{q}_2 \leftarrow \max\{\text{var}(\mathbf{x} \cdot \mathbf{q})\}$$
  $$\text{s.t. } \mathbf{q}_1 \cdot \mathbf{q}_2 = 0$$

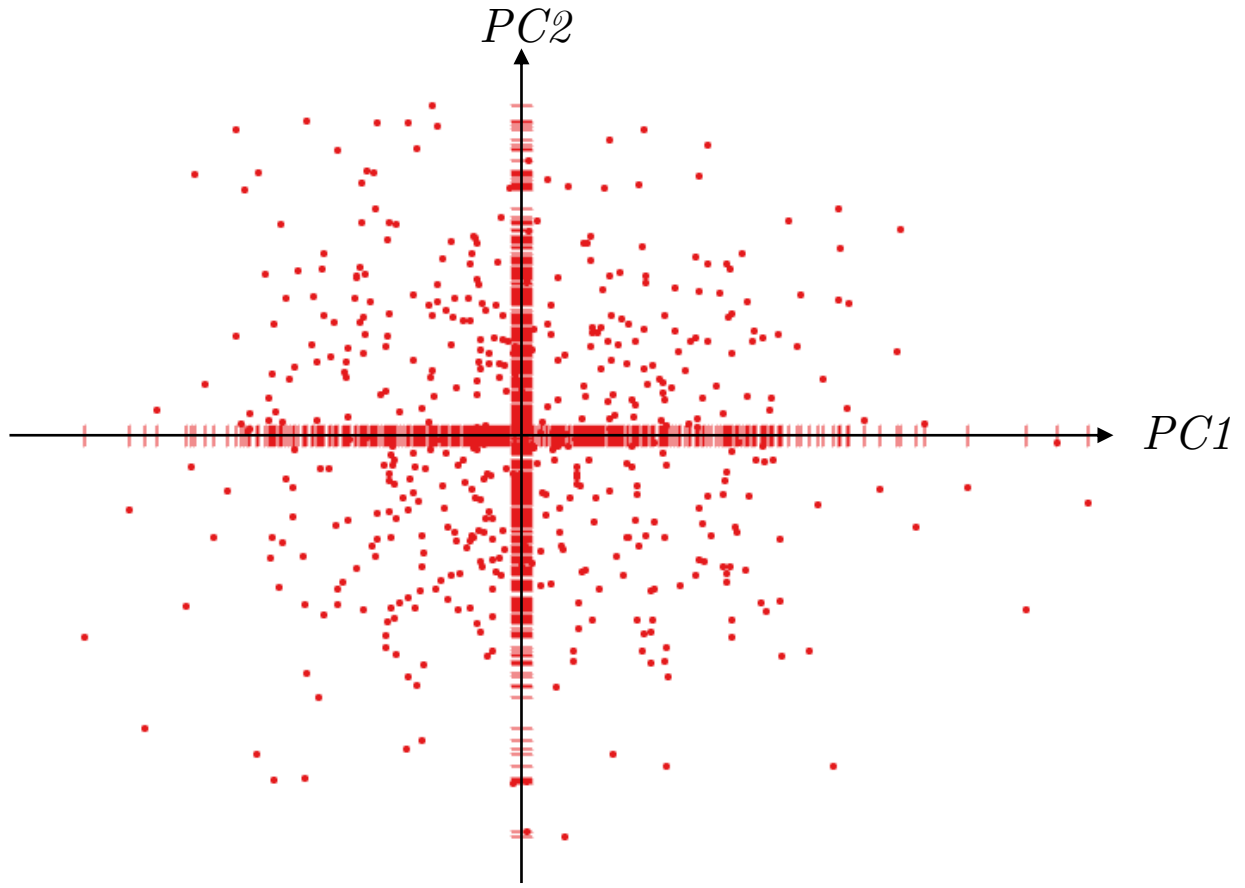
Scores $\mathbf{t} = \begin{bmatrix} \mathbf{x} \cdot \mathbf{q}_1 \\ \mathbf{x} \cdot \mathbf{q}_2 \end{bmatrix}$



Principal Component Analysis (PCA)

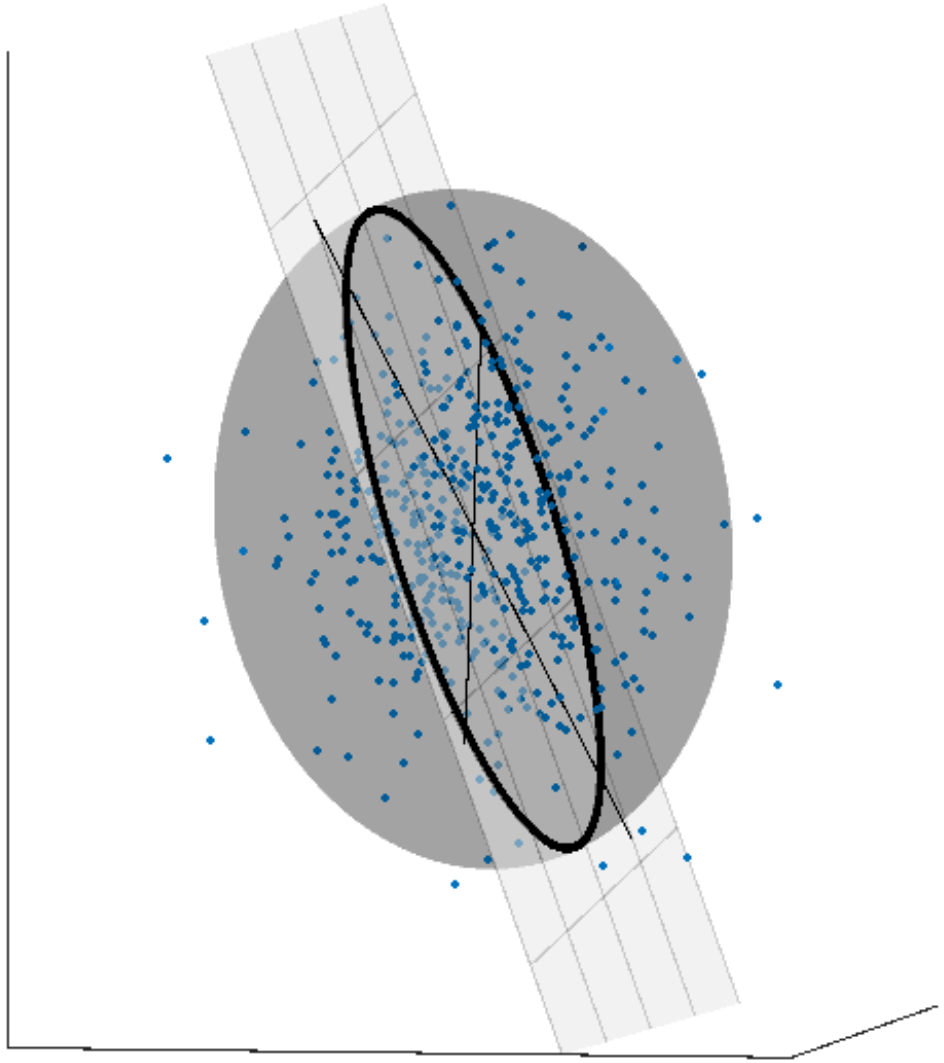
- Unsupervised learning
  no response variable
- Identify direction $\mathbf{q}_1$:
  $$\mathbf{q}_1 \leftarrow \max\{\text{var}(\mathbf{x} \cdot \mathbf{q})\}$$
- Identify direction $\mathbf{q}_2$:
  $$\mathbf{q}_2 \leftarrow \max\{\text{var}(\mathbf{x} \cdot \mathbf{q})\}$$
  $$\text{s.t. } \mathbf{q}_1 \cdot \mathbf{q}_2 = 0$$

Scores $\mathbf{t} = \begin{bmatrix} \mathbf{x} \cdot \mathbf{q}_1 \\ \mathbf{x} \cdot \mathbf{q}_2 \end{bmatrix}$



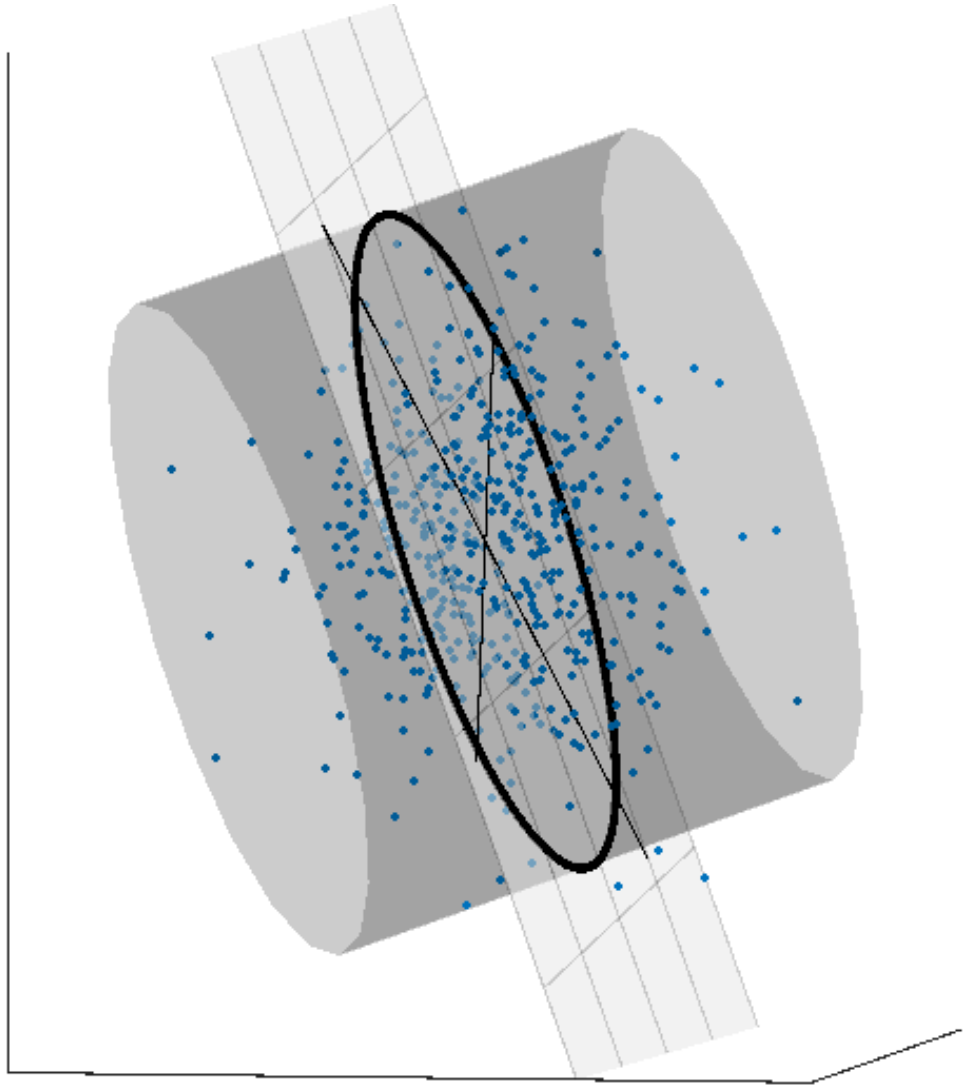
PCA for process monitoring

- Unsupervised learning
  no response variable
- In 3-dimensions, define ellipsoid containing normal data
  (6 parameters)
- Equivalent to using Hotelling's T^2 statistics in 3D



PCA for process monitoring

- Unsupervised learning
  no response variable
- In 3-dimensions, define ellipsoid containing normal data
  (6 parameters)
- Alternative: define cylinder containing normal data
  (6 parameters)
- Equivalent to using Hotelling's T^2 statistic + Squared Prediction Error (SPE)



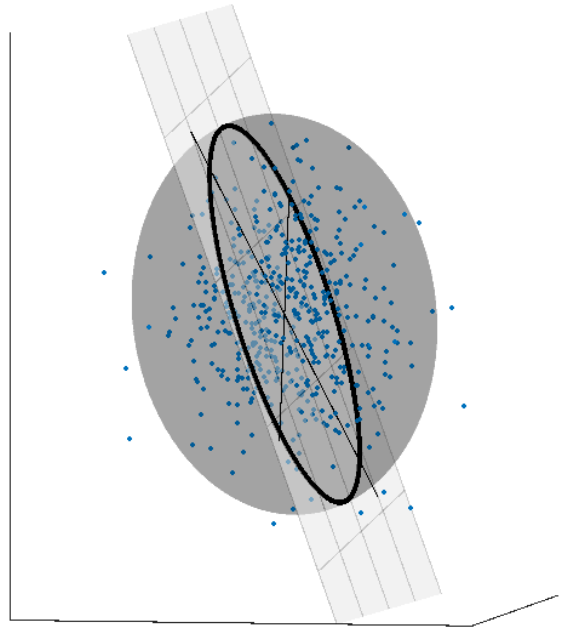
PCA for process monitoring

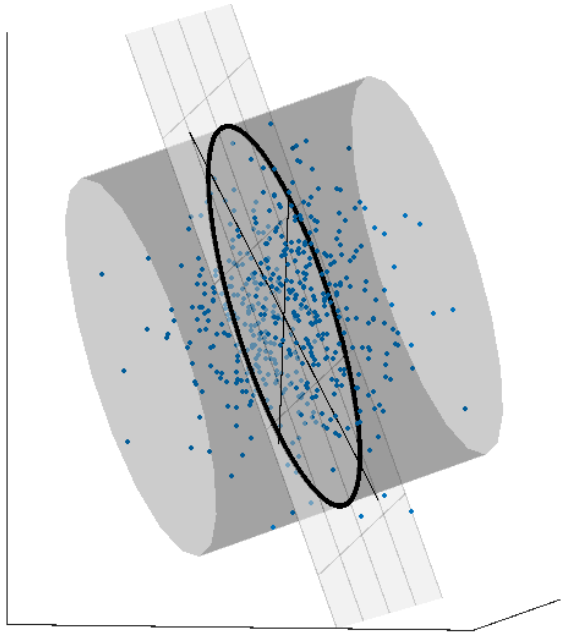
- Ellipsoid in 3 dimensions:

  3 parameters for direction + 3 parameters for length of principal axes = ***6***

- Cylinder in 3 dimensions:

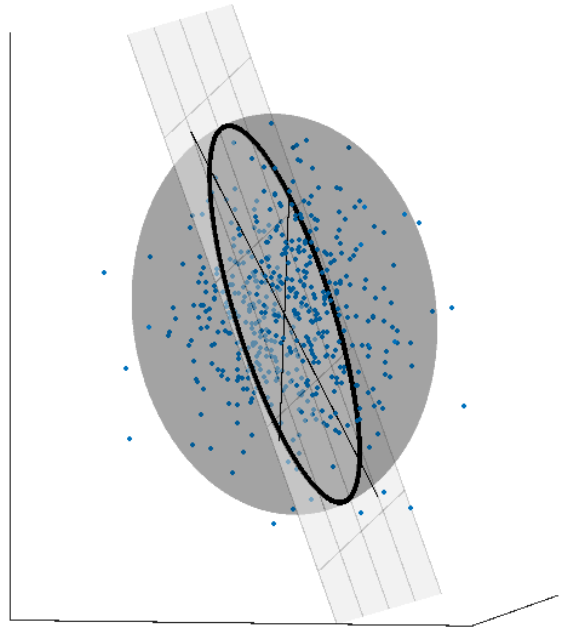
  3 parameters for direction + 2 parameters for length of principal axes, + 1 parameter for height = ***6***

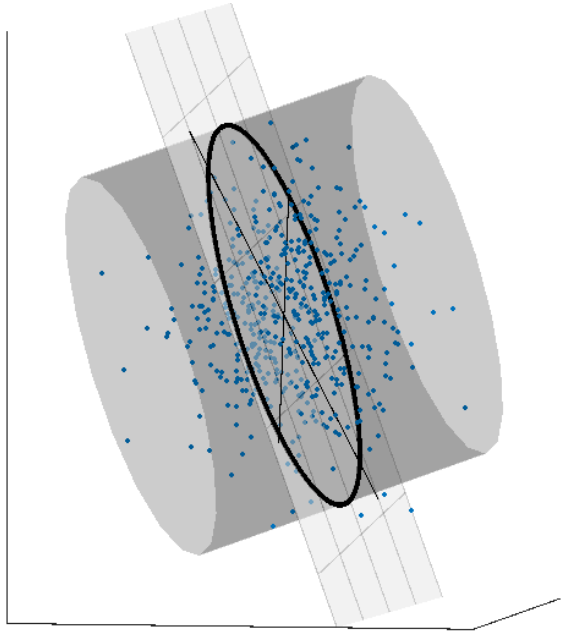




PCA for process monitoring

- In n dimensions, variance can be controlled by defining an ellipsoid in $m < n$ dimensions, then extruding the ellipsoid to in the remaining directions to form a "hypercylinder"





PCA for process monitoring

- Ellipsoid in n dimensions:

  *$n(n-1)/2$ parameters for direction + n for length of principal axes = **$n(n+1)/2$***

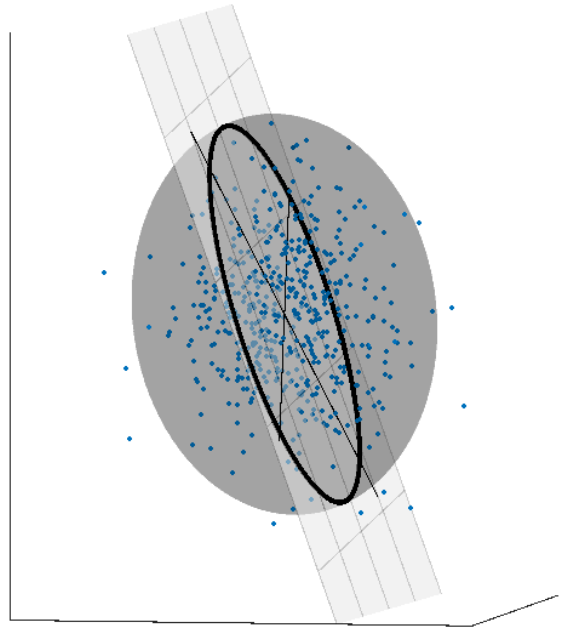
  High flexibility, high variance

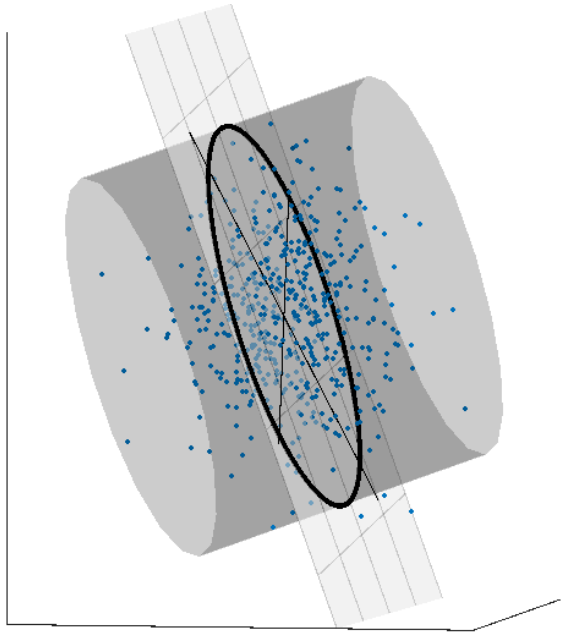
- Cylinder in n dimensions, with 2D base:

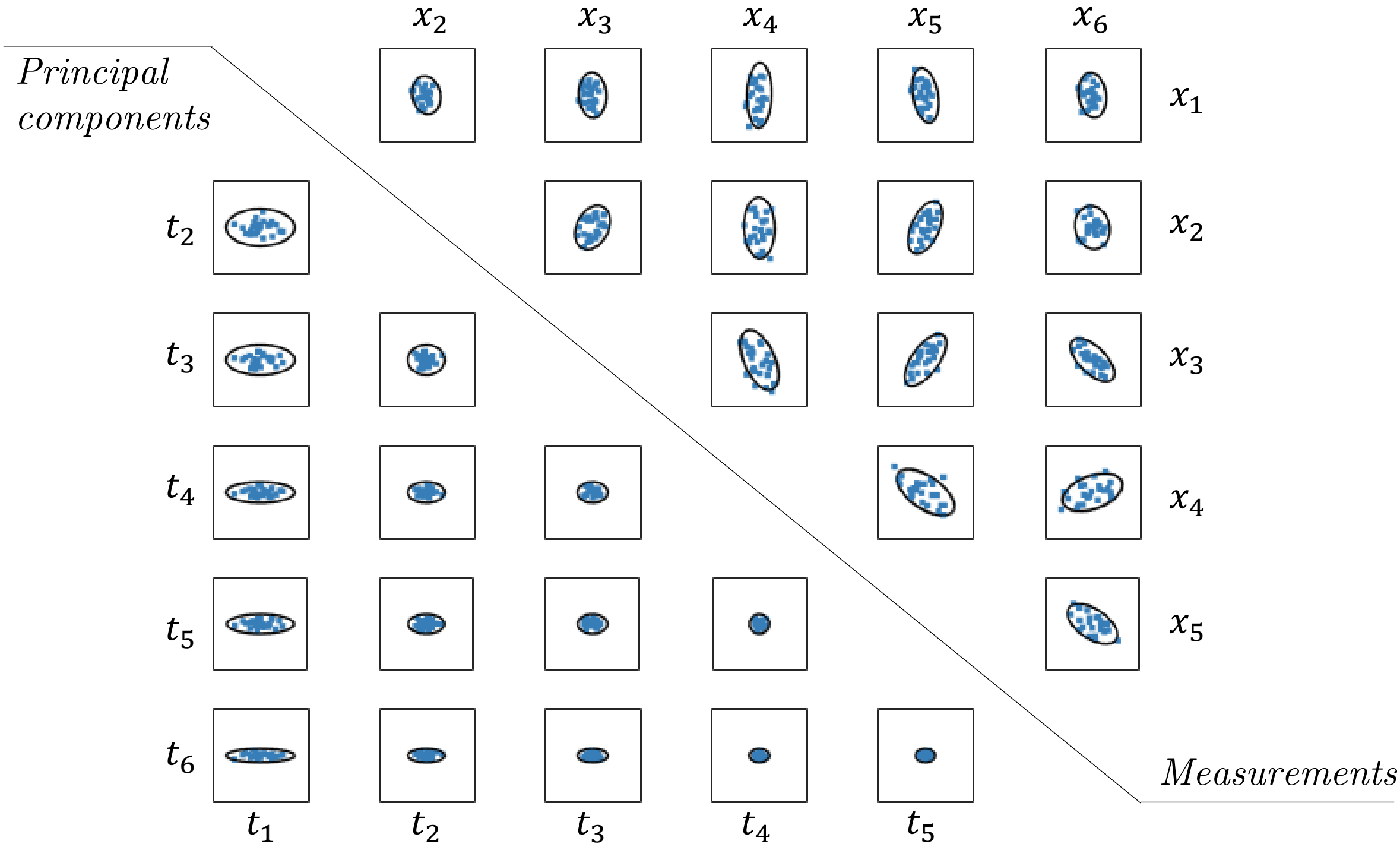
  *$2n$-3 parameters for direction + 2 parameters for length of principal axes, + 1 parameter for height = **$2n$***

  Low flexibility, low variance

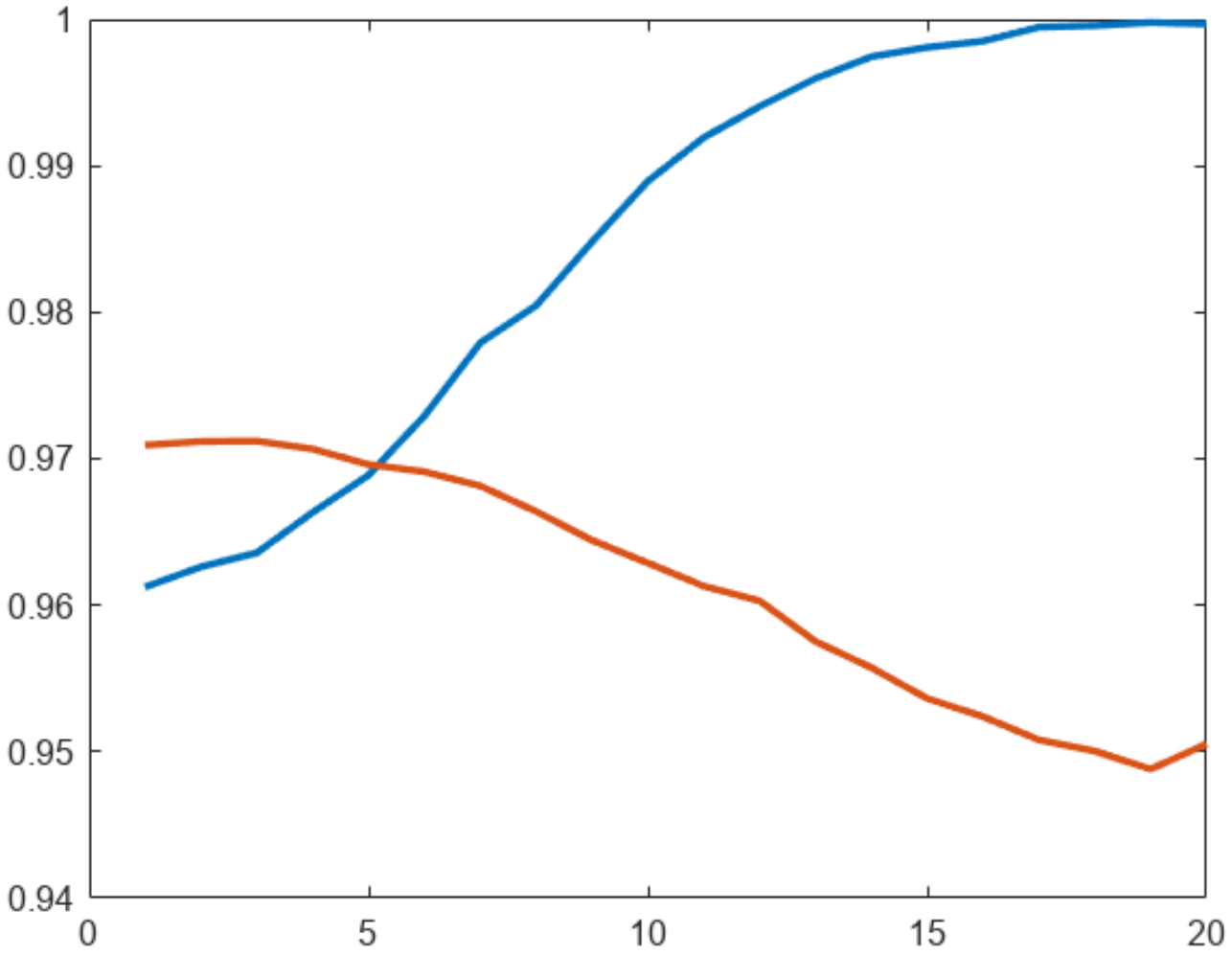
- Use PCA to select the m dimensions with the strongest covariance






Principal components
Measurements
x_2
x_3
x_4
x_5
x_6
x_1
x_2
x_3
x_4
x_5
t_2
t_3
t_4
t_5
t_6
t_1
t_2
t_3
t_4
t_5

PCA for process monitoring

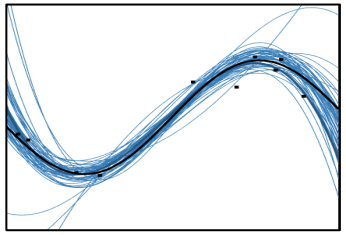


Number of principal components

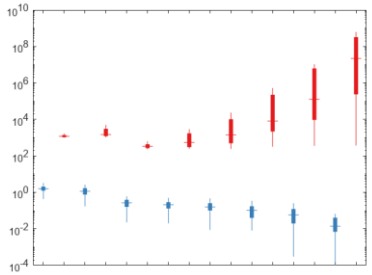
$$Sensitivity = \frac{True\ alarm}{Total\ fault}$$

$$Precision = \frac{True\ alarm}{Total\ alarm}$$

Conclusion

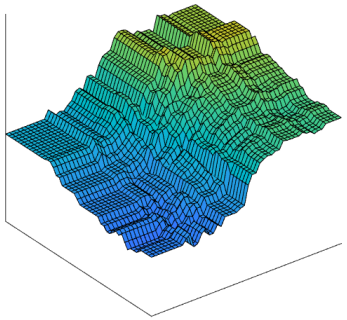


- (Supervised) machine learning aims to learn models that perform well on unseen data



Expected Prediction Error

$$= \sigma^2 + \underbrace{\left(\boldsymbol{f}(\boldsymbol{x_k}) - \mathbb{E}\left[\hat{\boldsymbol{f}}(\boldsymbol{x_k})\right]\right)^2}_{Bias} + \underbrace{\mathbb{E}\left[\left(\mathbb{E}\left[\hat{\boldsymbol{f}}(\boldsymbol{x_k})\right] - \hat{\boldsymbol{f}}(\boldsymbol{x_k})\right)^2\right]}_{Variance}$$

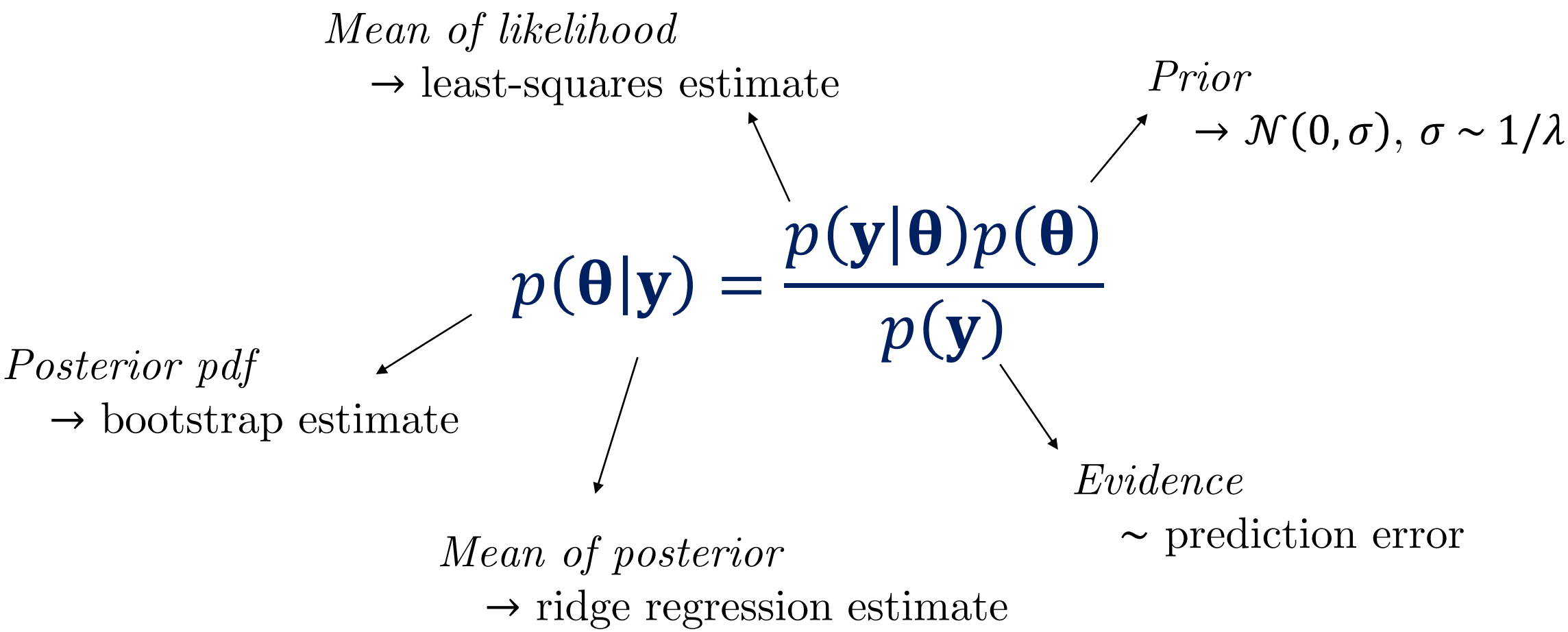


- Model flexibility can be controlled to minimize prediction error
- Successful ML models leverage the BV trade-off in a variety of ways

Thank you

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y})}$$

Bayesian interpretation



PCA maximizes variance, no requirement for normally distribute data