Introduction to Machine learning

Some slides and images are taken from:

David Wolfe Corne https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt
Wikipedia
Geoffrey A. Hinton

Examples 1

Examples 1

WaveNet

Examples 1

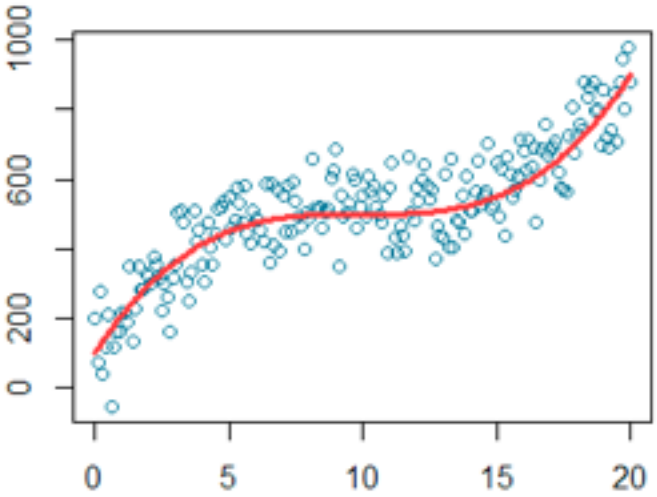
WaveNet

Examples 2

Describes without errors	Describes with minor errors	Somewhat related to the image	Unrelated to the image
 A person riding a motorcycle on a dirt road.	 Two dogs play in the grass.	 A skateboarder does a trick on a ramp.	 A dog is jumping to catch a frisbee.
 A group of young people playing a game of frisbee.	 Two hockey players are fighting over the puck.	 A little girl in a pink hat is blowing bubbles.	 A refrigerator filled with lots of food and drinks.
 A herd of elephants walking across a dry grass field.	 A close up of a cat laying on a couch.	 A red motorcycle parked on the side of the road.	 A yellow school bus parked in a parking lot.

- Basics of Machine Learning
- Feedforward nets
  - Backpropagation
  - Convolutional networks
- Boltzmann machines
  - Deep Belief Nets

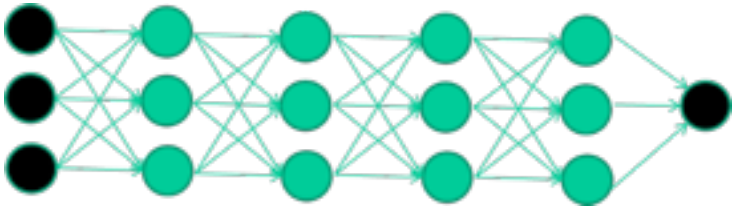
Supervised learning 1 – Function approximation



Shallow ML methods

- Regression
- Kernel Regression

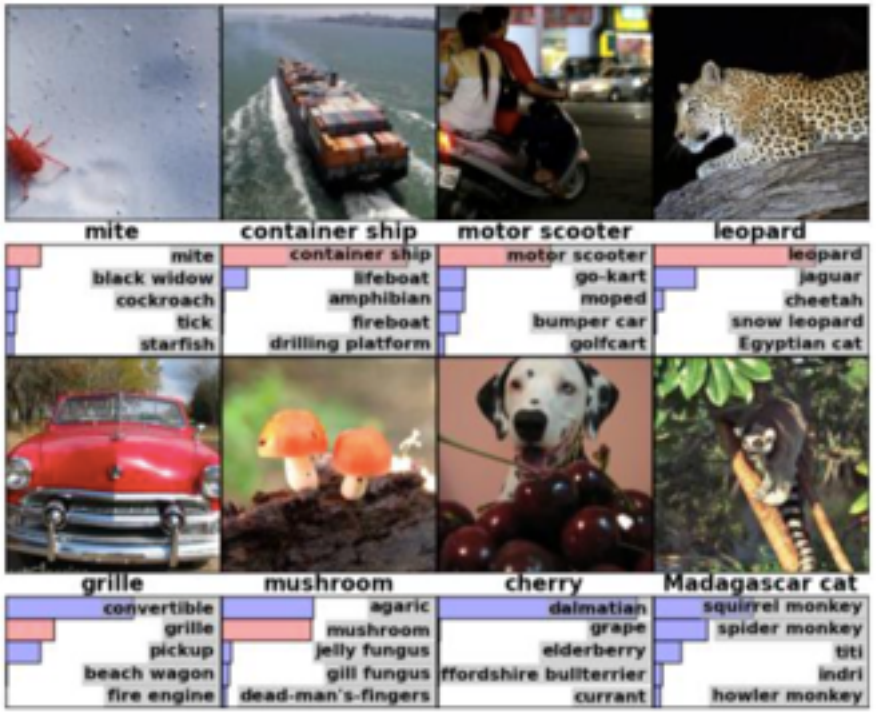
Deep ML methods



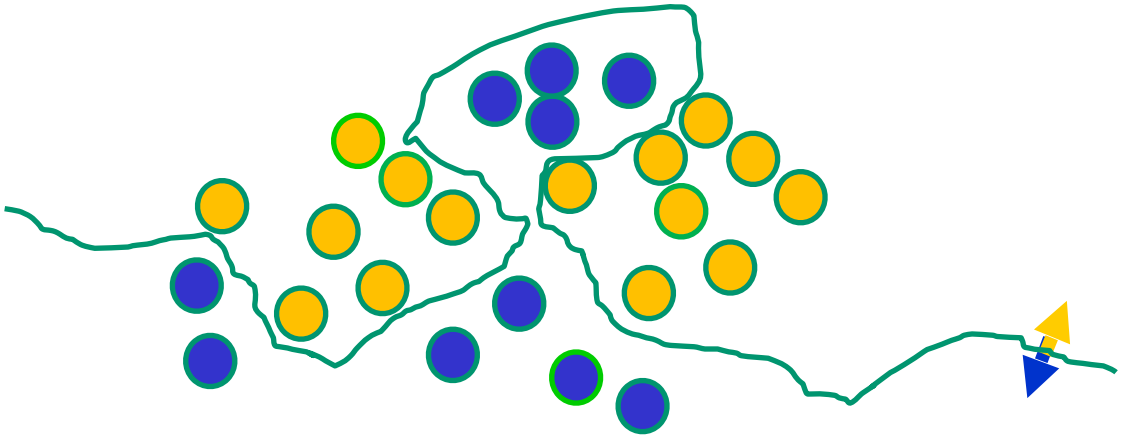
x y = F(x)

- Feedforward networks
- Deep tensor networks
- ConvNets

Supervised learning 2 – Classification



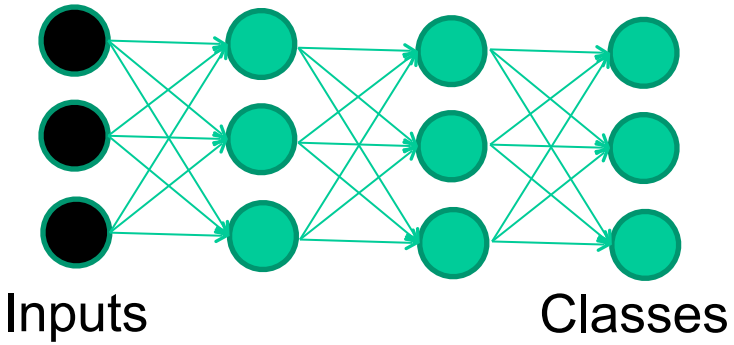
Supervised learning 2 – Classification



Shallow ML methods

- Support Vector Machines (SVMs)
- Kernel SVMs

Deep ML methods



Unsupervised learning 1 – Learning distributions

$$P(\mathbf{v}|\theta) = \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h}|\theta)$$

Unsupervised learning 1 – Learning distributions

$$P(\mathbf{v}|\theta) = \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h}|\theta)$$

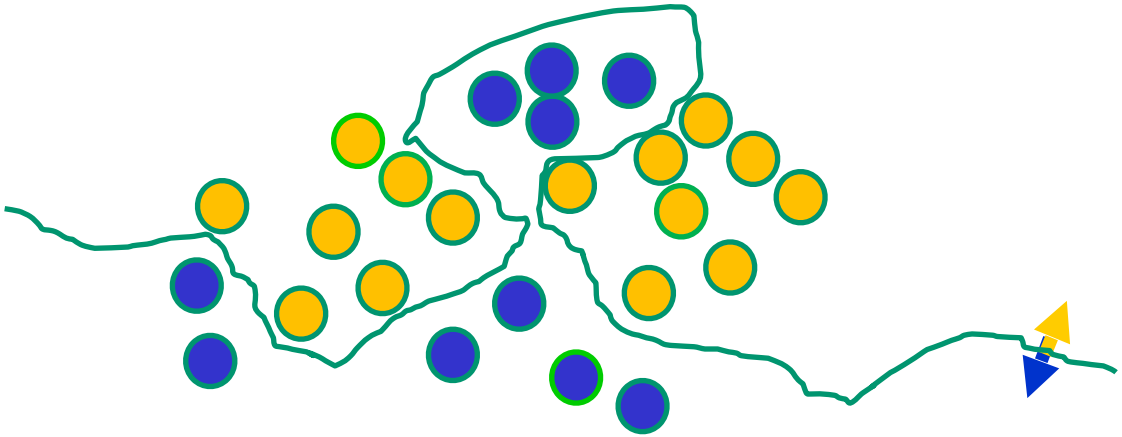
Shallow ML methods

- Markov random fields

Deep ML methods

- Boltzmann machines
- Deep Belief Nets
- variational Autoencoders

Unsupervised learning 2 – Classification



Shallow ML methods

- Clustering

Deep ML methods

- Deep transformation + clustering

Plan for the first talk

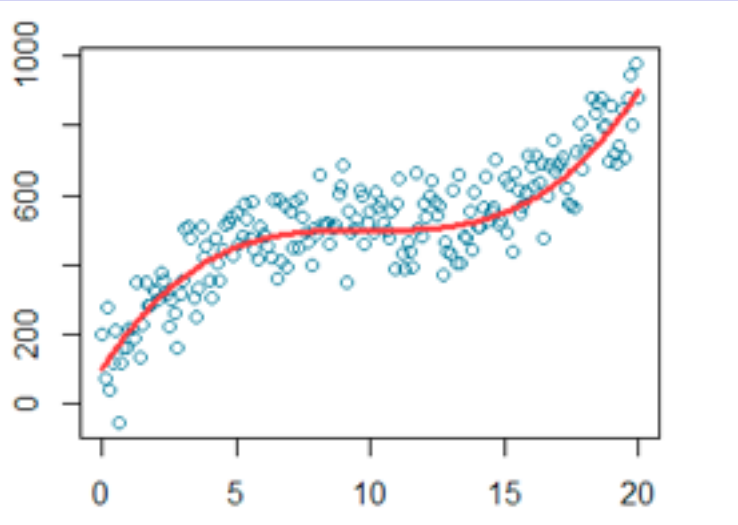
1. Linear least squares regression
   Example: Markov state models and Koopman models

2. Validation, cross-validation and hyperparameter selection

3. Regularization and sparsity
   Example: Sindy

4. Kernel ridge regression

Linear least squares regression



Regression

We are given a **training set** of m points (x_i, y_i). In vector notation: $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$.
We seek model equations that describe the data *approximately*:

$$y_i \approx f(x_i; \mathbf{w})$$

where $\mathbf{w} \in \mathbb{R}^n$ are parameters we need to **learn**.

Regression

We are given a **training set** of m points (x_i, y_i). In vector notation: $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$.
We seek model equations that describe the data *approximately*:

$$y_i \approx f(x_i;\ \mathbf{w})$$

where $\mathbf{w} \in \mathbb{R}^n$ are parameters we need to **learn**.
Minimize the residuals $\boldsymbol{\Delta}$:

$$\Delta_i = y_i - f(x_i;\ \mathbf{w}).$$

Using the p-norm of the vector $\boldsymbol{\Delta} \in \mathbb{R}^n$

$$\|\boldsymbol{\Delta}\|_p \quad = \quad \left(\sum_{i=1}^{m} |\Delta_i|^p \right)^{1/p},$$

the choice of p determines the type of regression problem:

p	Problem	Name
1	$\min_{\mathbf{w}} \sum_{i=1}^{m} \lvert\Delta_i\rvert$	L^1 (linear) optimization
2	$\min_{\mathbf{w}} \sum_{i=1}^{m} \Delta_i^2$	Least squares (Gauss ~1800)
∞	$\min_{\mathbf{w}} \max_i \Delta_i$	Tschebyscheff regression

Regression

We are given a **training set** of m points (x_i, y_i). In vector notation: $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$.
We seek model equations that describe the data *approximately*:

$$y_i \approx f(x_i;\ \mathbf{w})$$

where $\mathbf{w} \in \mathbb{R}^n$ are parameters we need to **learn**.
Minimize the residuals $\boldsymbol{\Delta}$:

$$\Delta_i = y_i - f(x_i;\ \mathbf{w}).$$

Using the p-norm of the vector $\boldsymbol{\Delta} \in \mathbb{R}^n$

$$\|\boldsymbol{\Delta}\|_p \quad = \quad \left(\sum_{i=1}^{m} |\Delta_i|^p \right)^{1/p},$$

the choice of p determines the type of regression problem:

p	Problem	Name
1	$\min_{\mathbf{w}} \sum_{i=1}^{m} \lvert\Delta_i\rvert$	L^1 (linear) optimization
2	$\min_{\mathbf{w}} \sum_{i=1}^{m} \Delta_i^2$	Least squares (Gauss ~1800)
∞	$\min_{\mathbf{w}} \max_i \Delta_i$	Tschebyscheff regression

Linear least squares

In a linear regression problem, the model equation has the form:

$$f(x_i;\ \mathbf{w}) = w_1\phi_1(x_i) + \ldots + w_n\phi_n(x_i)$$

where $\phi_j : \mathbb{R} \to \mathbb{R}$ are arbitrary **basis functions** or **feature functions** that can be nonlinear in x.

Define the matrix $\mathbf{X}$

$$X_{ij} = \phi_j(x_i),$$

to rewrite the linear least squares regression problem as:

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2$$

Normal equations

The vector $\mathbf{w} \in \mathbb{R}^n$ is the solution to $\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|$ exactly if it fulfills the normal equations

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

The linear regression problem has a unique solution exactly if the rank of $\mathbf{X}$ is maximal, i.e. $\mathrm{rk}(\mathbf{X}) = n$.

Normal equations

The vector $\mathbf{w} \in \mathbb{R}^n$ is the solution to $\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|$ exactly if it fulfills the normal equations

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

The linear regression problem has a unique solution exactly if the rank of $\mathbf{X}$ is maximal, i.e. $\mathrm{rk}(\mathbf{X}) = n$.

Solution methods:

1) **Direct inversion**: Computation of $\mathbf{X}^\top \mathbf{X}$ and direct inversion:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^+ \mathbf{y}.$$

where $\mathbf{X}^+$ is the Moore-Penrose pseudoinverse of $\mathbf{X}$.

Defining the correlation matrices

do not literally do this!

$$\mathbf{C}_{XX} = \mathbf{X}^\top \mathbf{X}$$
$$\mathbf{C}_{XY} = \mathbf{X}^\top \mathbf{y}$$

(here $\mathbf{C}_{XY} \in \mathbb{R}^{n \times 1}$, but it is a matrix if we have multiple regression targets) the formal solution can be written as:

$$\mathbf{w} = \mathbf{C}_{XX}^{-1} \mathbf{C}_{XY}.$$

Normal equations

The vector $\mathbf{w} \in \mathbb{R}^n$ is the solution to $\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|$ exactly if it fulfills the normal equations

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

The linear regression problem has a unique solution exactly if the rank of $\mathbf{X}$ is maximal, i.e. $\mathrm{rk}(\mathbf{X}) = n$.

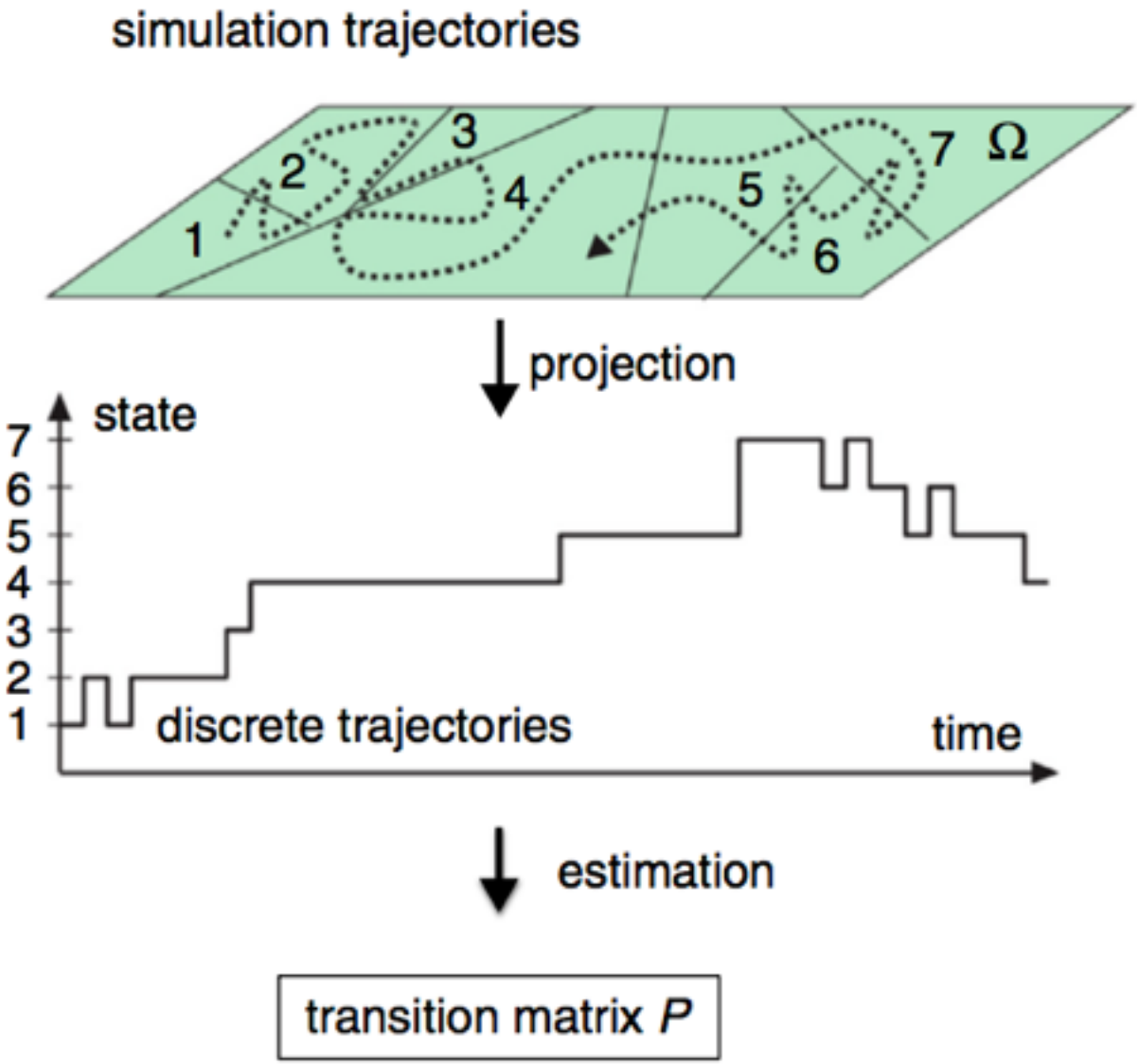
Solution methods:

2) Cholesky decomposition of C_{xx} (still numerically unstable, but sometimes useful)

3) Orthogonolization of X (e.g. via Householder reflections) — stable

4) SVD of X to perform the pseudoinversion — stable

Markov state models and Koopman models of molecular dynamics



Markov state models and Koopman models of molecular dynamics

Stochastic time series, obtained from MD, $\mathbf{x}_1, ..., \mathbf{x}_T$. Lag time $\tau \geq 1$.
Define basis functions $\phi_i(\mathbf{x})$ with $i = 1, ..., n$. Transform time series as:

$$X_{ij} = \phi_j(\mathbf{x}_i)$$
$$Y_{ij} = \phi_j(\mathbf{x}_{i+\tau})$$

with feature matrices $\mathbf{X}, \mathbf{Y} \in \mathbf{R}^{m \times n}$ and $m = T - \tau$.

Markov state models and Koopman models of molecular dynamics

Stochastic time series, obtained from MD, $\mathbf{x}_1, ..., \mathbf{x}_T$. Lag time $\tau \geq 1$.
Define basis functions $\phi_i(\mathbf{x})$ with $i = 1, ..., n$. Transform time series as:

$$X_{ij} = \phi_j(\mathbf{x}_i)$$
$$Y_{ij} = \phi_j(\mathbf{x}_{i+\tau})$$

with feature matrices $\mathbf{X}, \mathbf{Y} \in \mathbf{R}^{m \times n}$ and $m = T - \tau$.
Solve the regression problem:

$$\min_{\mathbf{K}} \|\mathbf{Y} - \mathbf{X}\mathbf{K}\|_2$$

in order to parametrize the linear, Markovian dynamical model $\mathbf{K}$ that describes the time evolution:

$$\mathbf{Y} \approx \mathbf{X}\mathbf{K},$$

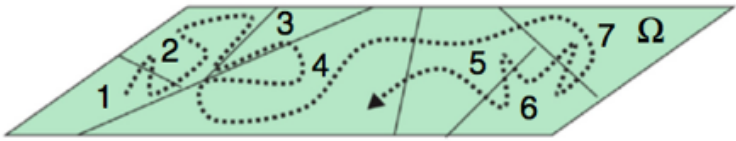
or in other words $\mathbf{x}_{t+\tau}^\top \approx \mathbf{x}_t^\top \mathbf{K}$. The solution has the form:

$$\mathbf{K} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{Y}.$$
$$= \mathbf{C}_{XX}^{-1} \mathbf{C}_{XY}$$

Markov state models

Markov state models: Partition molecular state space Ω into substates $S_1, ..., S_n$.
Define the characteristic functions:

$$\phi_j(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in S_j \\ 0 & \text{else.} \end{cases}$$



The correlation matrices $\mathbf{C}_{XX}$ and $\mathbf{C}_{XY}$ then evaluate to:

$$(C_{XX})_{ij} = \sum_{t=1}^{m} \phi_i(\mathbf{x}_t)\phi_j(\mathbf{x}_t) = \delta_{ij} N_i$$

$$(C_{XY})_{ij} = \sum_{t=1}^{m} \phi_i(\mathbf{x}_t)\phi_j(\mathbf{x}_{t+\tau}) = N_{ij}$$

N_i: number of times the trajectory was in state i

N_{ij}: number of transitions $i \to j$ in time interval τ.

As a result, the Markov model $\mathbf{K}$ can be written as:

$$K_{ij} = \frac{N_{ij}}{N_i}$$

Koopman models

Define basis functions $\phi_i(\mathbf{x})$ with $i = 1, ..., n$. Transform time series as:

$$X_{ij} = \phi_j(\mathbf{x}_i)$$
$$Y_{ij} = \phi_j(\mathbf{x}_{i+\tau})$$

Koopman model:
Wu et al: **JCP** 146, 154104 (2017)

$$\mathbf{K} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{Y}.$$
$$= \mathbf{C}_{XX}^{-1} \mathbf{C}_{XY}$$

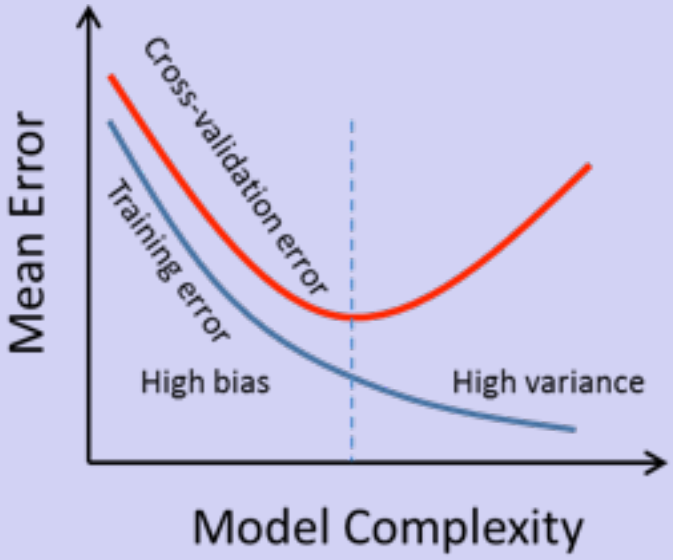
Eigenvalues and eigenvectors of K have been used to perform dimension reduction in:

VAC (variational approach of conformation dynamics)
- Noé and Nüske: **SIAM MMS** 11, 635-655 (2013)
- Nüske et al: **JCTC** 10, 1739-1752. (2014)

EDMD (Extended dynamic mode decomposition)
- Williams, Kevrekidis and Rowley: **J. Nonlinear Sci.** 25, 1307-1346 (2015)

TICA (time-lagged independent component analysis)
- Perez-Hernandez et al: **JCP** 139, 015102 (2013)
- Schwantes and Pande: **JCTC** 9, 2000-2009 (2013)

Validation, cross-validation and hyperparameter selection



Validation, cross-validation and hyperparameter selection

- *Validation*: LLS solution fives us the **training error** $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2$, but we would like to validate how good the learnt model **predicts** independent **test data**.

- *Hyperparameter selection*: Hyperparameters are parameters that cannot not be obtained from the learning algorithm (here LLS).
  Example: The type of function ϕ used for training cannot be determined by minimizing the training error. For example, the function

$$f(x) = \sum_{i=1}^{N} w_i \delta(|x - x_i|)$$

  has a training error of zero, but predicts nothing ($f(x) = 0$ for every point x not in the training set).

Validation

Divide dataset into

- **training set** $(\mathbf{X}^{\text{train}}, \mathbf{y}^{\text{train}})$
- **test set** $(\mathbf{X}^{\text{test}}, \mathbf{y}^{\text{test}})$.

Validation

Divide dataset into

- **training set** $(\mathbf{X}^{\text{train}}, \mathbf{y}^{\text{train}})$
- **test set** $(\mathbf{X}^{\text{test}}, \mathbf{y}^{\text{test}})$.

Learn parameters using the training set:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \left\| \mathbf{y}^{\text{train}} - \mathbf{X}^{\text{train}} \mathbf{w} \right\|_2$$

The resulting residual $\epsilon^{\text{train}} = \left\| \mathbf{y}^{\text{train}} - \mathbf{X}^{\text{train}} \mathbf{w} \right\|_2$ is the **training error** or **training loss.**

Validation

Divide dataset into

- **training set** $(\mathbf{X}^{\text{train}}, \mathbf{y}^{\text{train}})$
- **test set** $(\mathbf{X}^{\text{test}}, \mathbf{y}^{\text{test}})$.

Learn parameters using the training set:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \left\| \mathbf{y}^{\text{train}} - \mathbf{X}^{\text{train}} \mathbf{w} \right\|_2$$

The resulting residual $\epsilon^{\text{train}} = \left\| \mathbf{y}^{\text{train}} - \mathbf{X}^{\text{train}} \mathbf{w} \right\|_2$ is the **training error** or **training loss.**

The error of the learnt model in predicting data not used for the training,

$$\epsilon^{\text{test}} = \left\| \mathbf{y}^{\text{test}} - \mathbf{X}^{\text{test}} \mathbf{w} \right\|_2$$

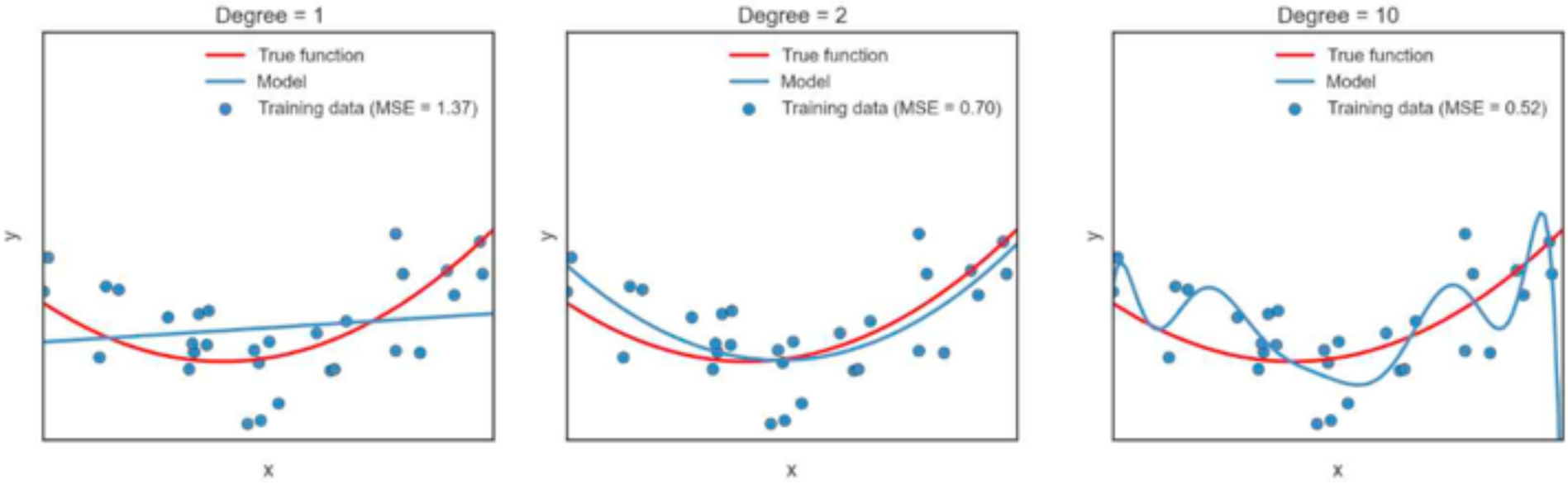
is called the **validation** or **test error/loss**. It provides a metric to validate how well the model generalized to new data, and this error can be used for hyperparameter optimization.

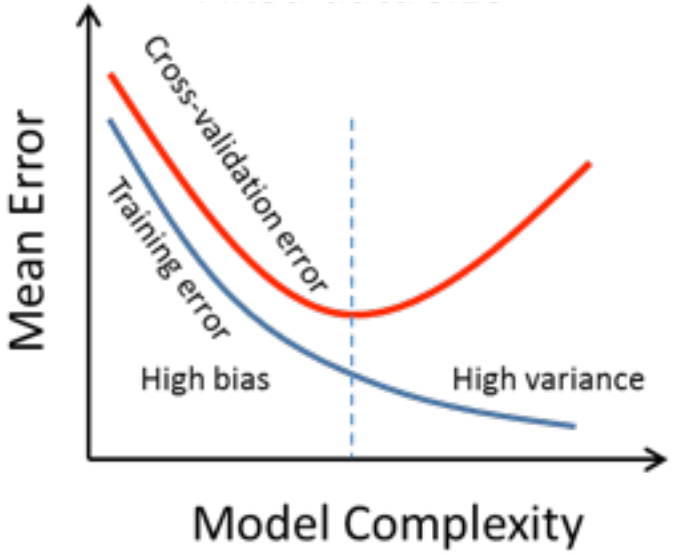
Cross-validation

Cross-validation

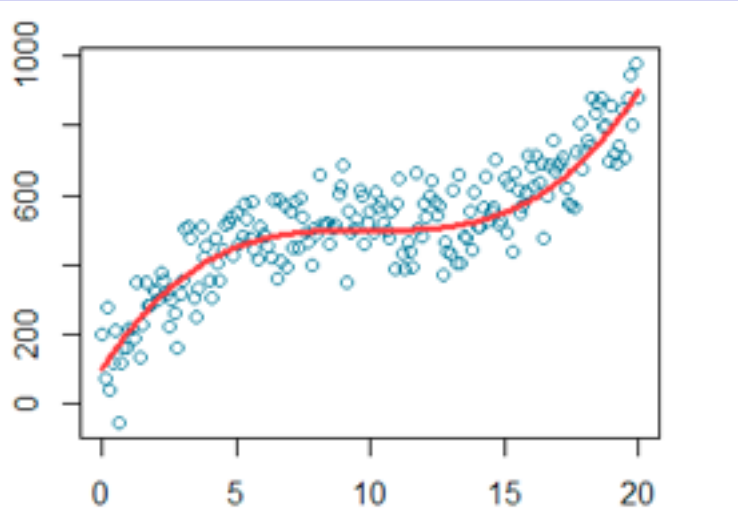
1. Split the data into k nonoverlapping folds $(\mathbf{X}^i, \mathbf{y}^i)$. The complementary sets are $(\mathbf{X}^{-i}, \mathbf{y}^{-i})$.
2. For each fold i:
   (a) Train learning algorithm on training data:
   $$\mathbf{w}^i = \arg\min_{\mathbf{w}} \left\| \mathbf{y}^{-i} - \mathbf{X}^{-i}\mathbf{w} \right\|_2$$
   (b) Compute validation error:
   $$\epsilon^i = \left\| \mathbf{y}^i - \mathbf{X}^i\mathbf{w}^i \right\|_2$$
3. Cross-validation error is then given by:
   $$\epsilon = \frac{1}{k} \sum_{i=1}^{k} \epsilon^k.$$

Hyperparameter selection





Regularization and sparsity



L2 / Tihonov regularization / Ridge regression

Penalize the norm of the solution:

$$\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2,$$

where λ is a hyperparameter. Taking derivatives and setting them to zero yields the solution:

$$\begin{aligned}\mathbf{w} &= \left(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y} \\ &= \left(\lambda \mathbf{I} + \mathbf{C}_{XX}\right)^{-1} \mathbf{C}_{XY}\end{aligned}$$

This is basicly equal to the direct solution of the normal equations, $C_{XX}^{-1} C_{XY}$, only that we use a shrinkage estimator for C_{XX}.

Sparsity-inducing regularization

L0 (too hard)

$$\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_0 ,$$

L1 (LASSO)

$$\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1 ,$$

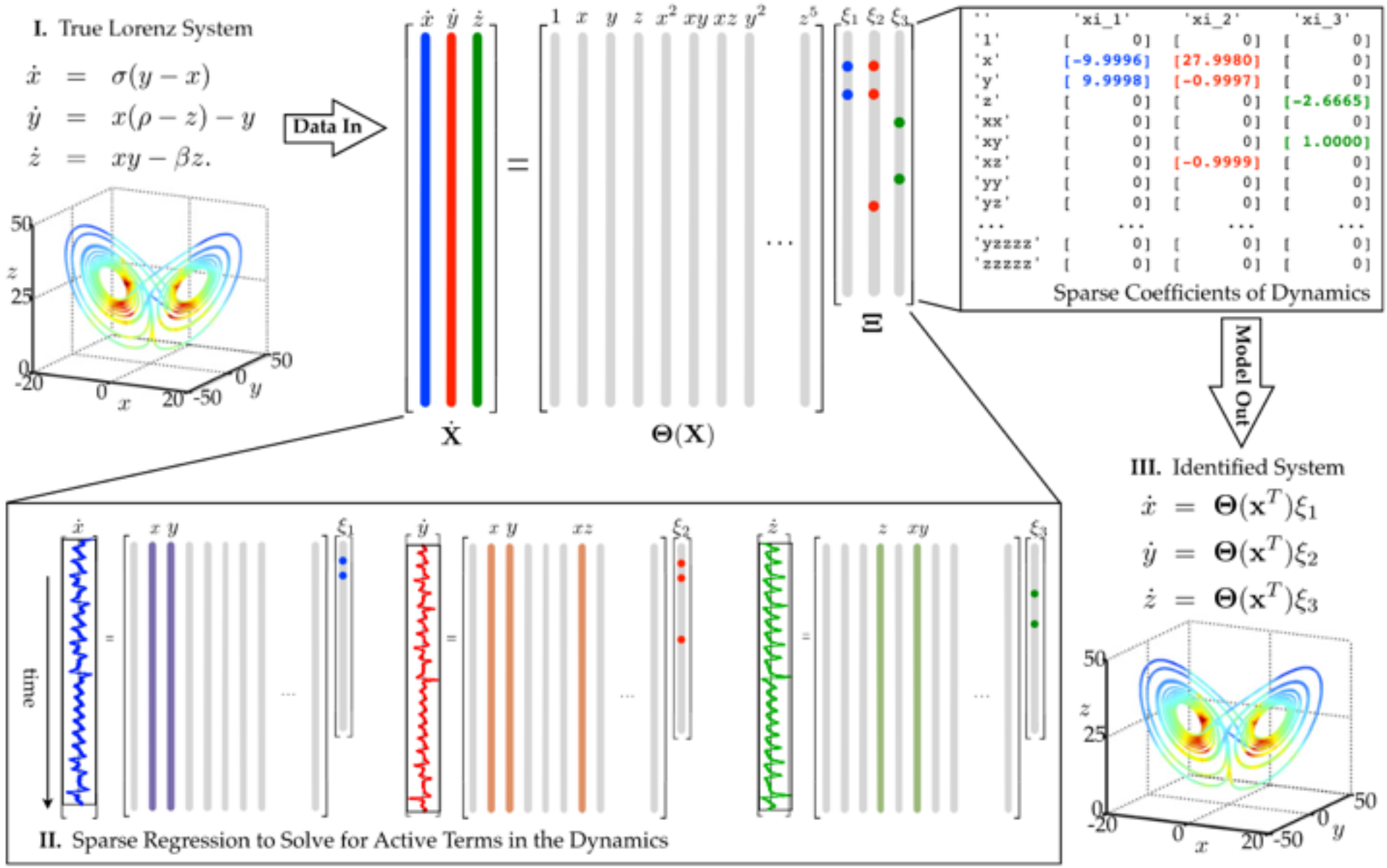
Elastic net

$$\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \left[(1-\alpha) \|\mathbf{w}\|_1 + \alpha \|\mathbf{w}\|_2^2 \right] ,$$

Good approach in practice:

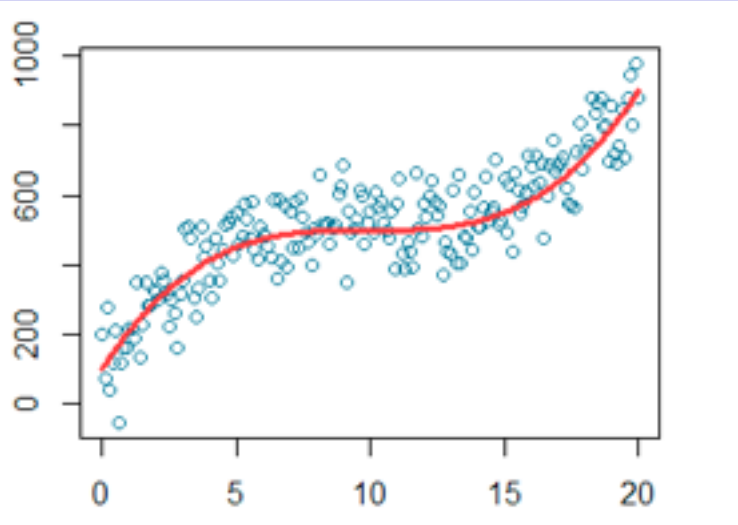
- Use L1 regularization and annihilate small elements with thresholding.
- Select regularization hyperparameters with cross-validation.

Sindy (sparse identification of nonlinear dynamics)



Brunton, Proctor and Kutz: PNAS 113, 3932–3937 (2016)

Kernel ridge regression



Kernel ridge regression

Reminder: ridge regression

Penalize the norm of the solution:

$$\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2,$$

where λ is a hyperparameter. Taking derivatives and setting them to zero yields the solution:

$$\mathbf{w} = \left(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

Kernel ridge regression

Kernel formulation

Using basic algebraic identities, we can rewrite the ridge regression solution as:

$$\mathbf{w} = \left(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}\right)^{-1} \mathbf{y},$$

which can be rewritten as:

$$\mathbf{w} = \sum_i \alpha_i \mathbf{X}_i$$

$$\boldsymbol{\alpha} = \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}\right)^{-1} \mathbf{y}$$

Where $\mathbf{X}_i$ is the feature vector of the ith sample.

Kernel ridge regression

Kernel formulation

Using basic algebraic identities, we can rewrite the ridge regression solution as:

$$\mathbf{w} = \left(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}\right)^{-1} \mathbf{y},$$

which can be rewritten as:

$$\mathbf{w} = \sum_i \alpha_i \mathbf{X}_i$$

$$\boldsymbol{\alpha} = \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}\right)^{-1} \mathbf{y}$$

Where $\mathbf{X}_i$ is the feature vector of the ith sample.

Observations:

- The solution $\mathbf{w}$ lies in the span of the m samples, even if $m \gg n$, or even $m = \infty$.
- We use the kernel matrix $\mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{m \times m}$, the number of samples squared.

Kernel ridge regression

Key ideas

- Never perform the feature transformation $\mathbf{X}_i = (\phi_1(\mathbf{x}_i), ..., \phi_1(\mathbf{x}_i))$ explicitly.
- Instead, define a kernel function that models the scalar product between feature vectors:

  $$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{X}_i^\top \mathbf{X}_j = \langle \mathbf{X}_i, \mathbf{X}_j \rangle$$

  This kernel function *implicitly* corresponds to a (possibly large or infinite) feature space.
- To compute the predicted value for a new test point, $\tilde{\mathbf{y}} = f(\tilde{\mathbf{x}})$, we use:

  $$\tilde{\mathbf{y}} = \mathbf{y}(\mathbf{K} + \lambda \mathbf{I})^{-1} \boldsymbol{\kappa}(\tilde{\mathbf{x}})$$

  where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\kappa_i(\mathbf{x}) = k(\mathbf{x}_i, \mathbf{x})$.

Example: polynomial Kernel

A kernel function corresponds to an inner product in a feature space based on the feature mapping $\phi(\cdot)$, without having to execute this feature mapping explicitly:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle.$$

For degree-d polynomials, the *polynomial kernel* is defined as

$$k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^\top \mathbf{x}_2 + c)^d$$

For $d = 2$ its feature mapping is given by:

$$\phi_2(\mathbf{x}) = \left[x_1^2, \ldots, x_n^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_{n-1}x_n, \sqrt{2c}x_1, \ldots, \sqrt{2c}x_n, c \right]^\top$$