Kernel Methods: An Infinity Game

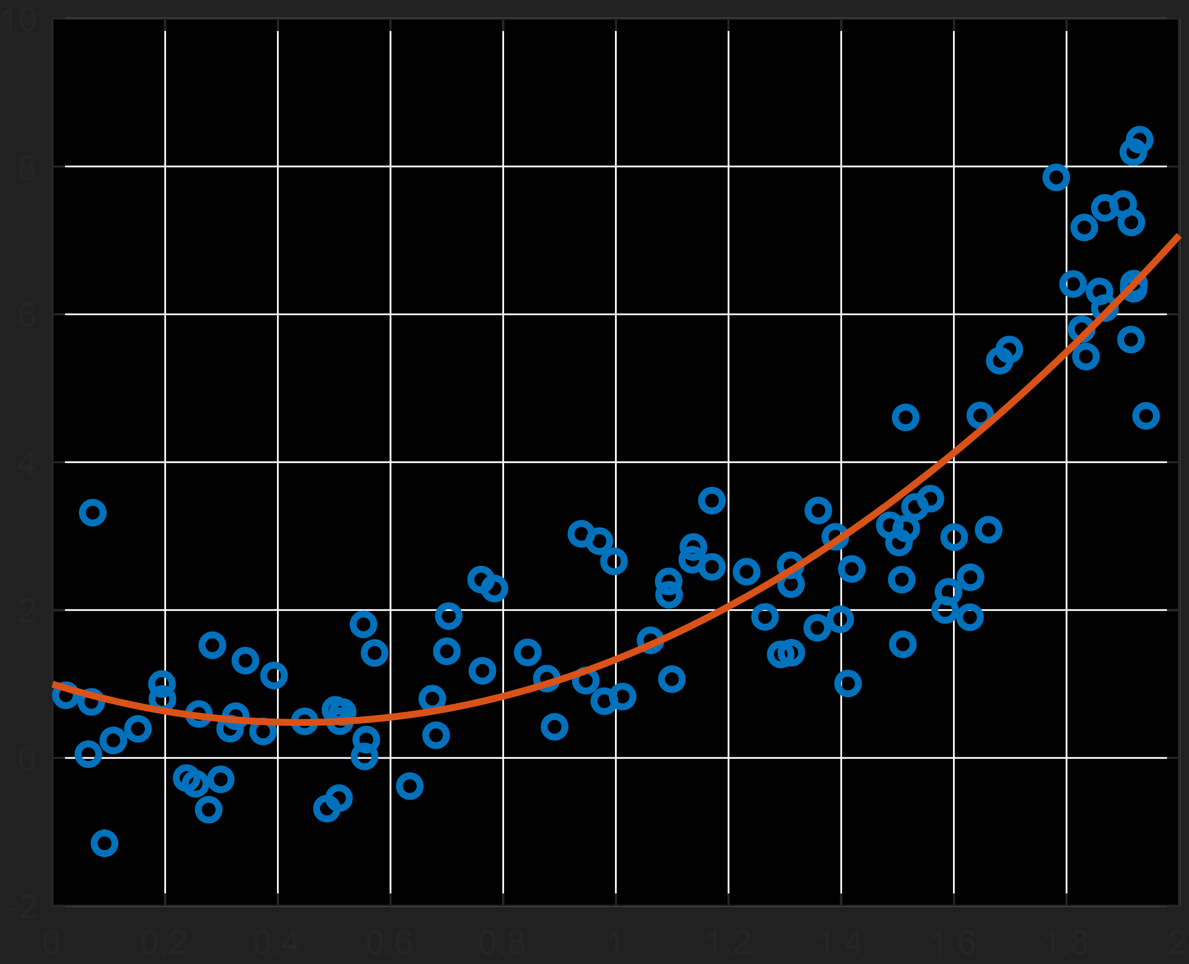
COMS21202, Part III

- Song Liu (song.liu@bristol.ac.uk), Lecturer in Data Science and A.I.

Objectives

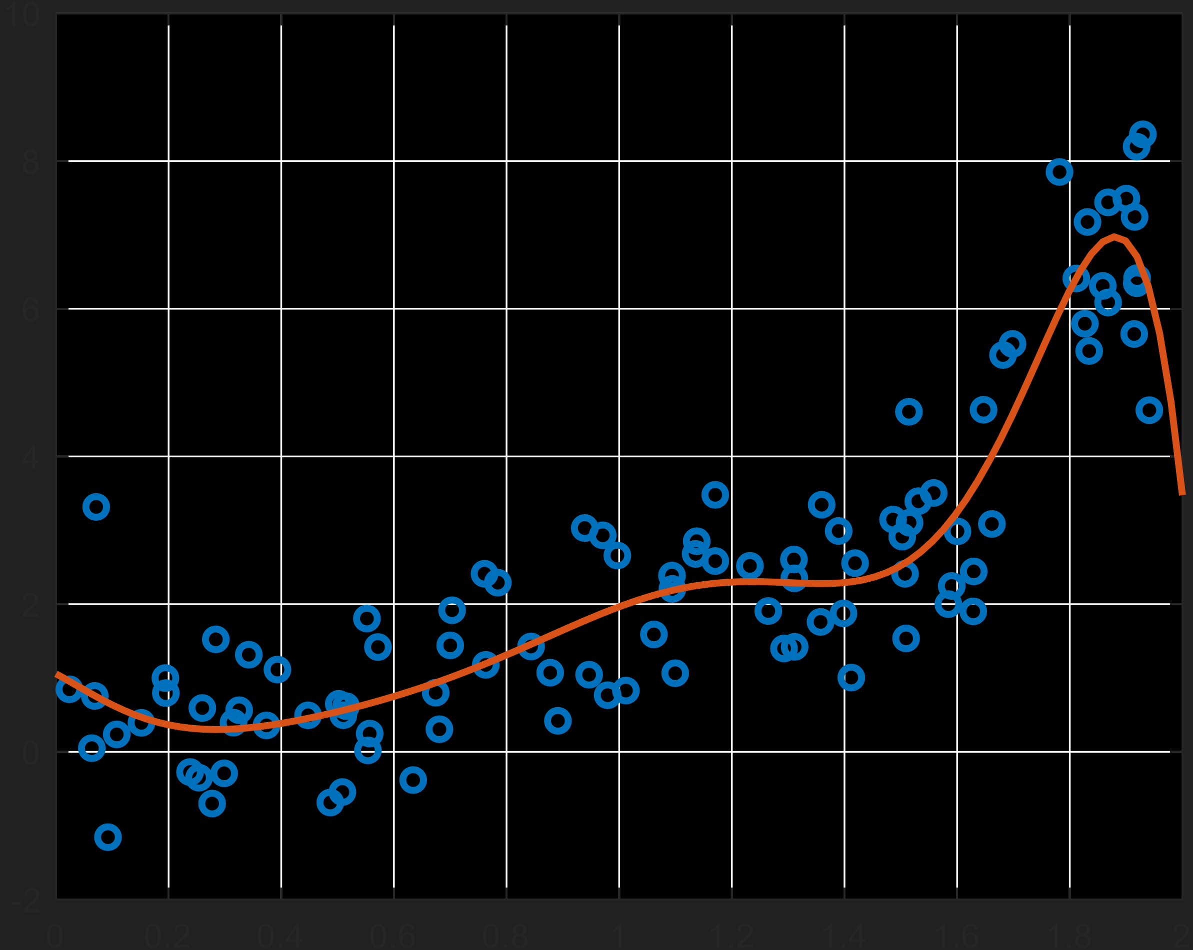
- Applying kernel tricks in LS.
- Knowing common choices of kernel functions.

Recall: $y = \exp(1.5x - 1) + \epsilon,\ \epsilon \sim N(0,1)$



- Polynomial transform with $b = 2$.
- Tr. error:108.97

Recall: $y = \exp(1.5x - 1) + \epsilon$, $\epsilon \sim N(0,1)$



- Polynomial transform with $b = 8$.
- Tr. error:78.87

Observation

- By increasing output dimension of feature transform $\boldsymbol{f}(\boldsymbol{x})$, we increase the flexibility of $\hat{y}$.
- Why don't we keep increasing m to get a super flexible $\hat{y}$?
  - We address overfitting issue later.
- **Problem**: large m causes **numerically issues**.

Numerical Issues of LS Solution

- Suppose $\boldsymbol{f}(\boldsymbol{x}) \in R^m$.
- As we discussed before, if $m > n$
  - $\boldsymbol{f}(\boldsymbol{X})^\top \boldsymbol{f}(\boldsymbol{X})$ is **singular**.
  - LS solution, $\widehat{\boldsymbol{\beta}} := \left(\boldsymbol{f}(\boldsymbol{X})^\top \boldsymbol{f}(\boldsymbol{X})\right)^{-1} \boldsymbol{f}(\boldsymbol{X})^\top \boldsymbol{y}$ cannot be calculated.
  - Shorten $\boldsymbol{f}(\boldsymbol{X})$ as $\boldsymbol{F}$ from now on.

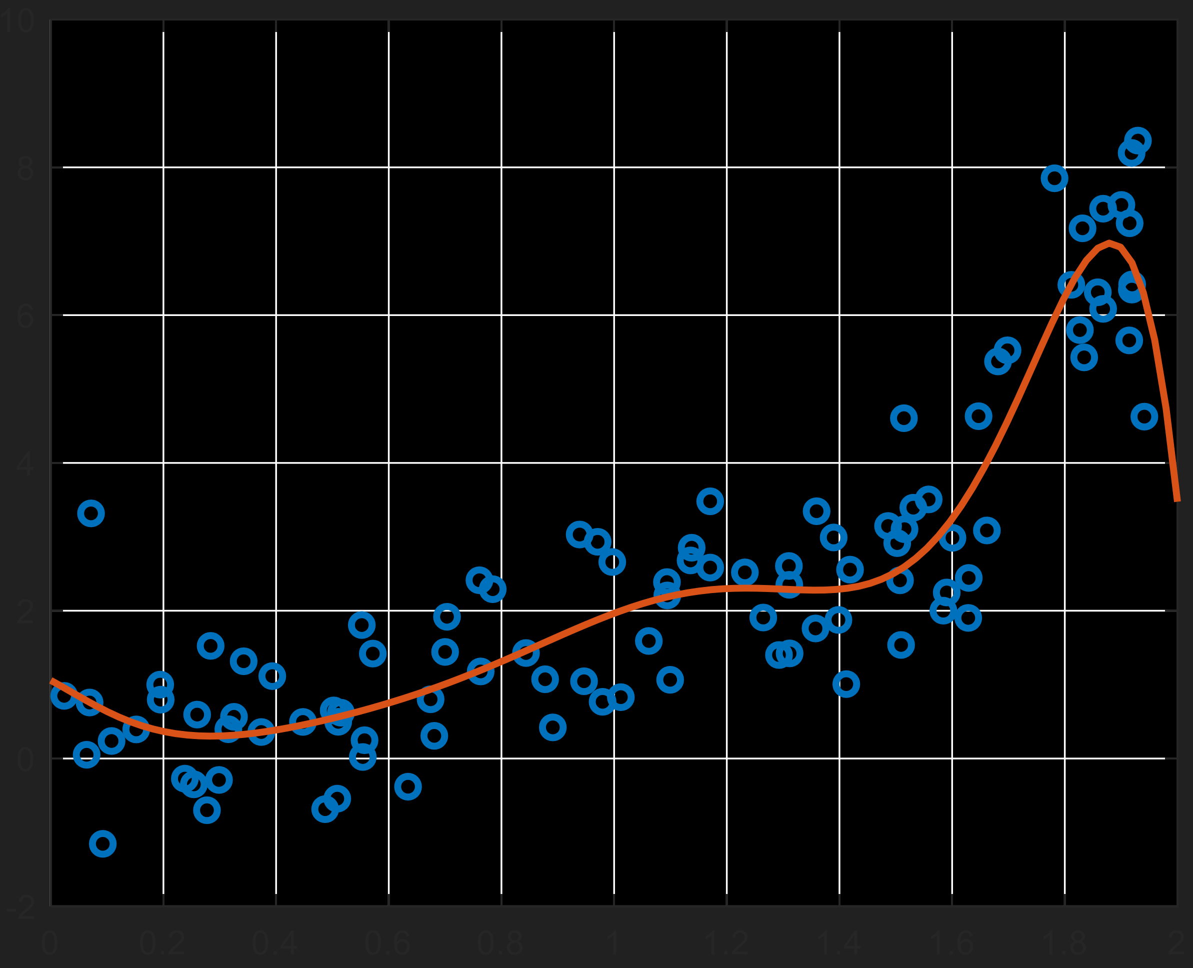
A Numerical Hack: Regularized LS Solution

- Instead of calculating
  - $\widehat{\boldsymbol{\beta}} := \left(\boldsymbol{F}^{\top}\boldsymbol{F}\right)^{-1}\boldsymbol{F}^{\top}\boldsymbol{y}$
- We calculate
  - $\widehat{\boldsymbol{\beta}} := \left(\boldsymbol{F}^{\top}\boldsymbol{F} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{F}^{\top}\boldsymbol{y}$
  - Where $\boldsymbol{I} \in R^{m\times m}$ is identity matrix.
  - λ is some small value, say 0.01.

Regularized LS Solution and Overfitting

- $\lambda \boldsymbol{I}$ helps battle overfitting too (!):
  - Increasing λ decreases the magnitude of $\widehat{\boldsymbol{\beta}}$, making $\hat{y}$ approx. a constant 0, which in fact, reduces the flexibility.
  - Show when $\lambda \to \infty, \widehat{\boldsymbol{\beta}} \approx \mathbf{0}$.
  - **One stone, two birds.**

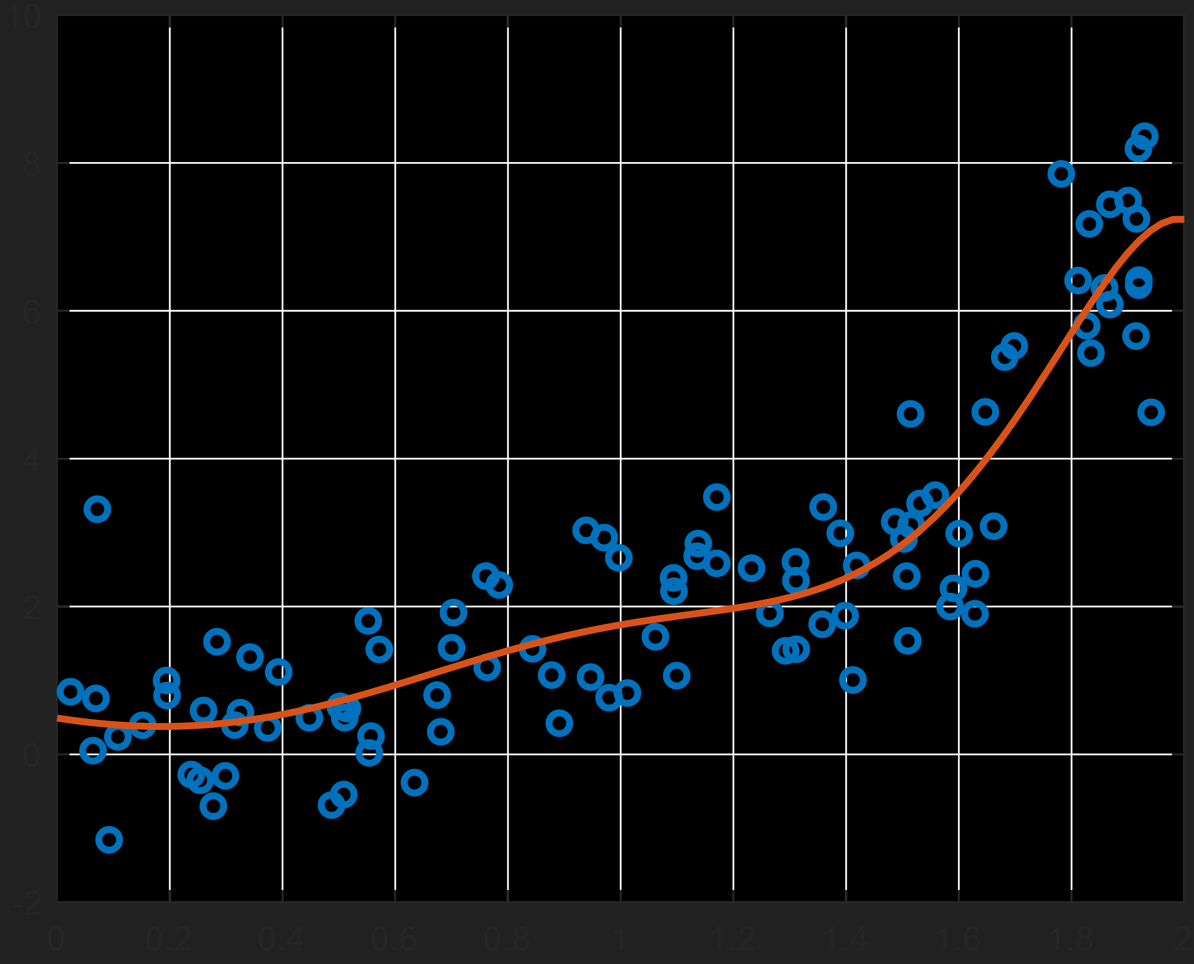
Example: $y = \exp(1.5x - 1) + \epsilon$, $\epsilon \sim N(0,1)$



- Polynomial transform with $b = 8$.
- $\lambda = 0$
- Tr. error: 78.87
- Te. error: 128.01

- Song Liu (song.liu@bristol.ac.uk), Lecturer in Data Science and A.I.

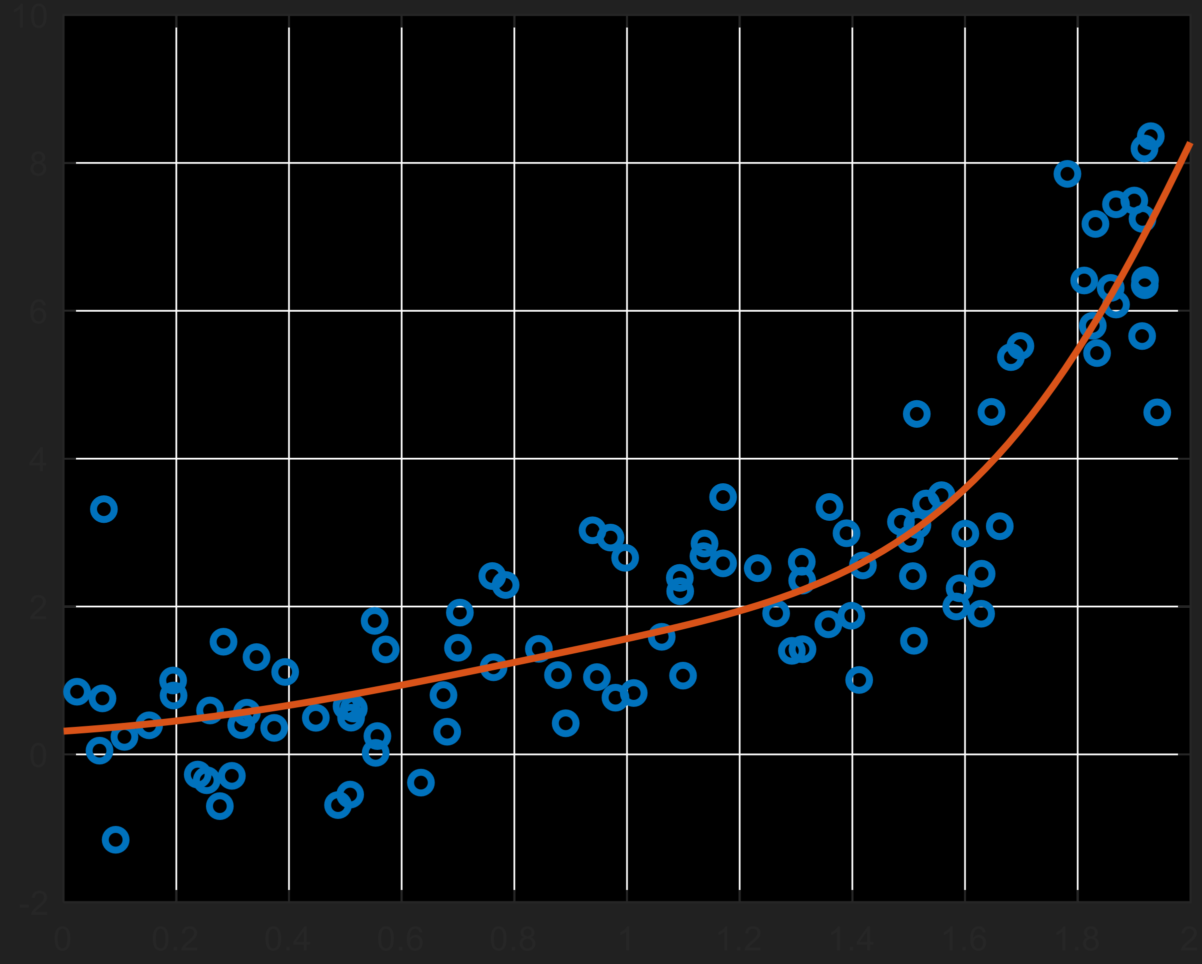
Example: $y = \exp(1.5x - 1) + \epsilon$, $\epsilon \sim N(0,1)$



- Polynomial transform with $b = 8$.
- $\lambda = .1$
- Tr. error: 86.46
- Te. error: 112.47

• Song Liu (song.liu@bristol.ac.uk), Lecturer in Data Science and A.I.

Example: $y = \exp(1.5x - 1) + \epsilon$, $\epsilon \sim N(0,1)$



- Polynomial transform with $b = 9$.
- $\lambda = 1$
- Tr. error: 92.89
- Te. error: 107.88

Regularized LS Solution and Overfitting

- λ is called regularization parameter.
  - Should be fixed before fitting.
  - Can be tuned by selecting the value that minimizes testing error.
  - Just like how we select b for $\boldsymbol{f}$.

Can we still raise the game?

- Can we design $\boldsymbol{f}(\boldsymbol{x})$ transforms the original $\boldsymbol{x}$ into a **infinitely dim. vector**?
- It should create a super flexible $\hat{y}$!
- Recall $\widehat{\boldsymbol{\beta}} := \left(\boldsymbol{F}^\top \boldsymbol{F} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{F}^\top \boldsymbol{y}$
  - Problem: now $\boldsymbol{F}^\top \boldsymbol{F} \in R^{m \times m}$, m is infinity.
  - How do you store $\boldsymbol{F}$ in computer??

Numerical Hack, #2:
Rewrite Solution using Woodbury identity

- Remarkably,
  - $\hat{\boldsymbol{\beta}} := \left(\boldsymbol{F}^\top \boldsymbol{F} + \lambda \boldsymbol{I}\right)^{-\mathbf{1}} \boldsymbol{F}^\top \boldsymbol{y} = \boldsymbol{F}^\top \left(\boldsymbol{F}\boldsymbol{F}^\top + \lambda \boldsymbol{I}\right)^{-\mathbf{1}} \boldsymbol{y}$
- Hint, Woodbury identity:
- $\left(\boldsymbol{P}^{-1} + \boldsymbol{B}^\top \boldsymbol{B}\right)^{-1} \boldsymbol{B}^\top = \boldsymbol{P}\boldsymbol{B}^\top \left(\boldsymbol{B}\boldsymbol{P}\boldsymbol{B}^\top + \boldsymbol{I}\right)^{-1}$
- Live demonstration

Numerical Hack, #2:
Rewrite Solution using Woodbury identity

- $\widehat{\boldsymbol{\beta}} := \boldsymbol{F}^\top \left(\boldsymbol{F}\boldsymbol{F}^\top + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{y}$
  - Now instead of $\boldsymbol{F}^\top \boldsymbol{F} \in R^{m \times m}$, we just need to compute $\boldsymbol{F}\boldsymbol{F}^\top \in R^{n \times n}$.
  - Let $\boldsymbol{K} := \boldsymbol{F}\boldsymbol{F}^\top$, where
  - $K^{(i,j)} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) := \langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j) \rangle$,
  - i.e., $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the inner product of two m dimensional feature transform.

Numerical Hack, #2:
Rewrite Solution using Woodbury identity

- $\hat{y} =< \hat{\beta}, f(x_0) >$
  - $\hat{y} =< f(x_0), F^\top \left(FF^\top + \lambda I\right)^{-1} y >$
    $= < f(x_0)F^\top, \left(FF^\top + \lambda I\right)^{-1} y >$
  - Rewrite $f(x_0)F^\top$ as $k \in R^n$ we can see
    - $k^{(i)} = k(x_0, x_i) =< f(x_0), f(x_i) >$
    - Verify this your self!

Numerical Hack, #3: Evaluating only the Inner Products

- $\hat{\boldsymbol{y}} := \boldsymbol{k}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$
- Note how $\boldsymbol{f}(\boldsymbol{x})$ only appears in the form of inner products!
- Even if we cannot write $\boldsymbol{f}(\boldsymbol{x})$ explicitly, we may still compute its inner product!
  - design "an inner product function k" mimics behaviour of inner product .
  - Forget about the existence of $\boldsymbol{f}$!

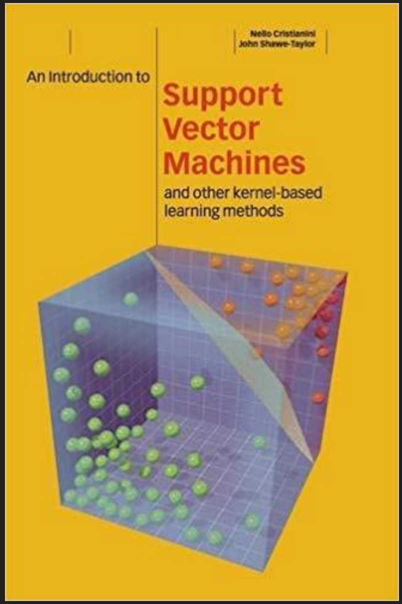
Numerical Hack, #3: Evaluating only the Inner Products

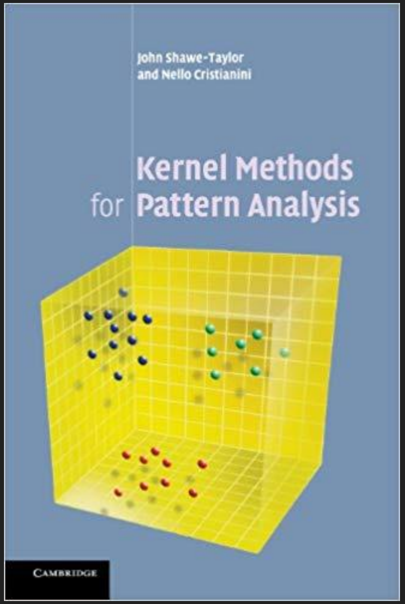
- It turns out, you **cannot** pick inner product function k arbitrarily.
  - Must "behaves like" a inner product.
- However, there are many **known choices** of k corresponds to inner products of powerful, even infinite dimensional feature transform $\boldsymbol{f}$.
  - We usually say a k induces an $\boldsymbol{f}$
  - **Even** if we cannot write $\boldsymbol{f}$ down.
- $k(x_i, x_j)$ is called **kernel function.**

The History of Kernel Methods

- Kernel methods were extremely important research topics in machine learning community in the early 2000s.
- It is now referred as “shallow methods”, in comparison to deep neural network models.
- It still enjoys great popularity for its simple mathematical expressions and power to represent extremely complex model.

Kernel @ Bristol





- Prof. Nello Cristianini at EngMath is one of the world renowned leading scientists in kernel methods.

Choices of k

- Linear kernel function:
  - $k(\boldsymbol{x}_i, \boldsymbol{x}_j) := < \boldsymbol{x}_i, \boldsymbol{x}_j >$
  - Implicit feature transform $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{x}$.
- Polynomial kernel function with degree b:
  - $k(\boldsymbol{x}_i, \boldsymbol{x}_j) \coloneqq (< \boldsymbol{x}_i, \boldsymbol{x}_j > +1)^d$
- PC: write down induced $\boldsymbol{f}(\boldsymbol{x})$ by polynomial kernels $b = 2$.

Choices of k

- RBF (or Gaussian) kernel:
  - $k(\boldsymbol{x}_i, \boldsymbol{x}_j) := \exp\left(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{\sigma^2}\right)$
  - $\boldsymbol{f}(\boldsymbol{x})$ induced by k is **infinitely dimensional**!
  - σ is chosen before fitting.
  - Best σ is chosen by minimizing testing error.
- Déjà vu?

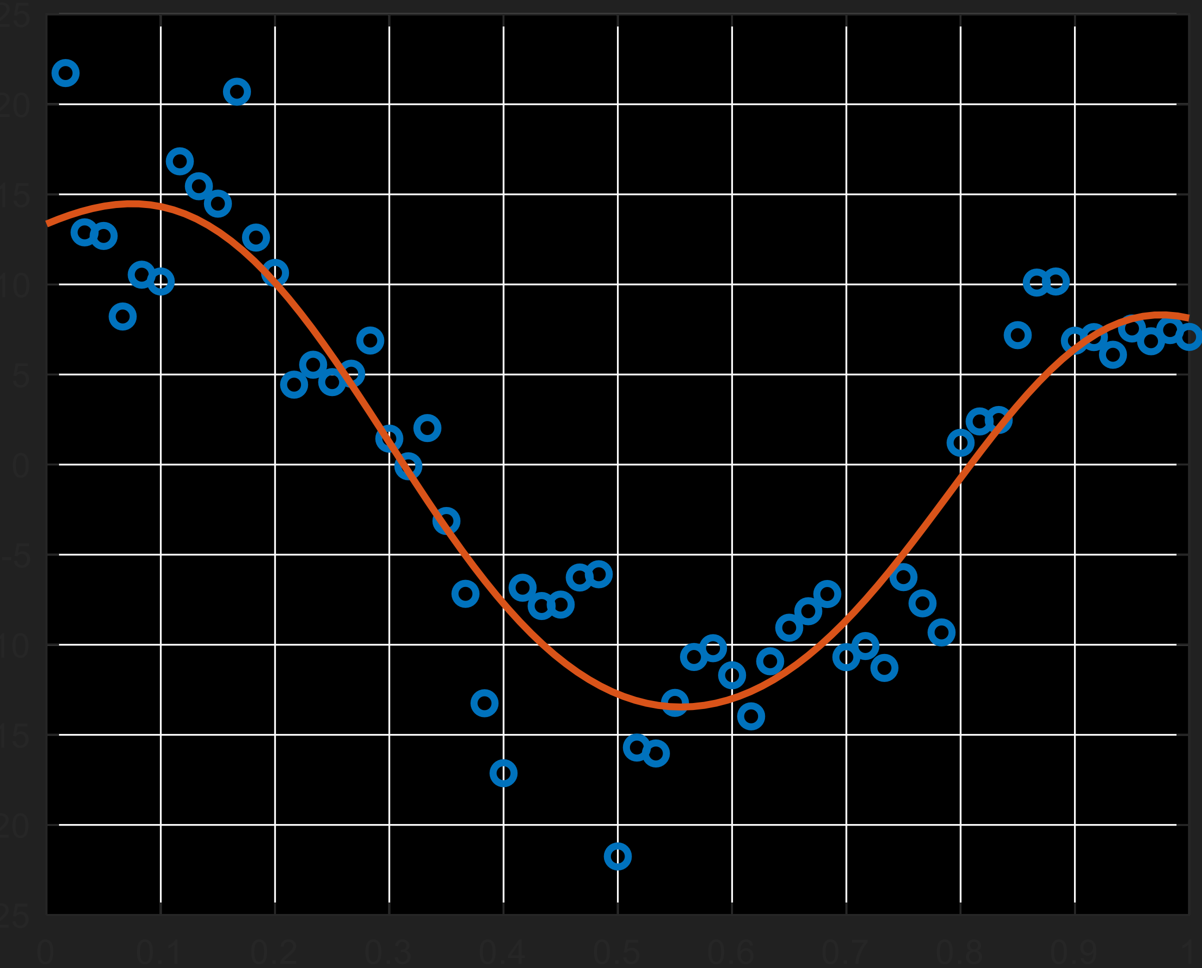
Choices of k

- How do I pick k?
  - Depending on your learning task.
    - e.g., linear/poly kernels are frequently used in natural language processing.
  - Depending on your dataset.
    - e.g., some kernels are even defined for structural inputs, such as strings or graphs.
  - Domain knowledge matters!!
- RBF kernel is a good all-rounded choice for $\boldsymbol{x} \in R^d$.

Implementation of Kernel LS

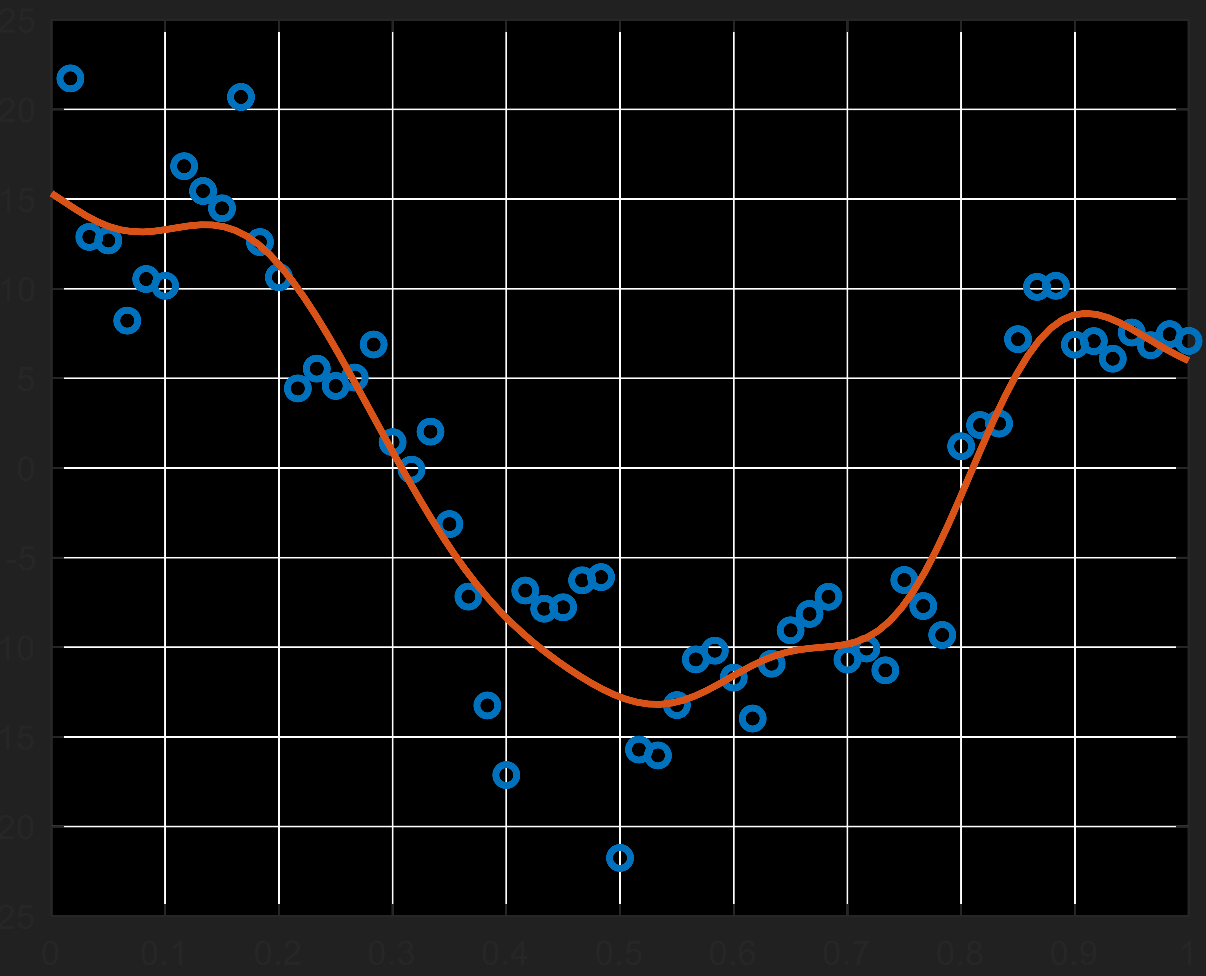
- Recall: $\hat{\boldsymbol{y}} := \boldsymbol{k}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$
- Computational cost
  - $\boldsymbol{K}$: $O(n^2)$
  - $(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}$: Usually $O(n^3)$
  - Kernel methods though flexible, is computationally demanding for large n.

Example: Apple Stock Price, Feb 2019



- RBF kernel
  - σ =0.2121
  - $\lambda = 0.1.$
- Tr error: 833.58

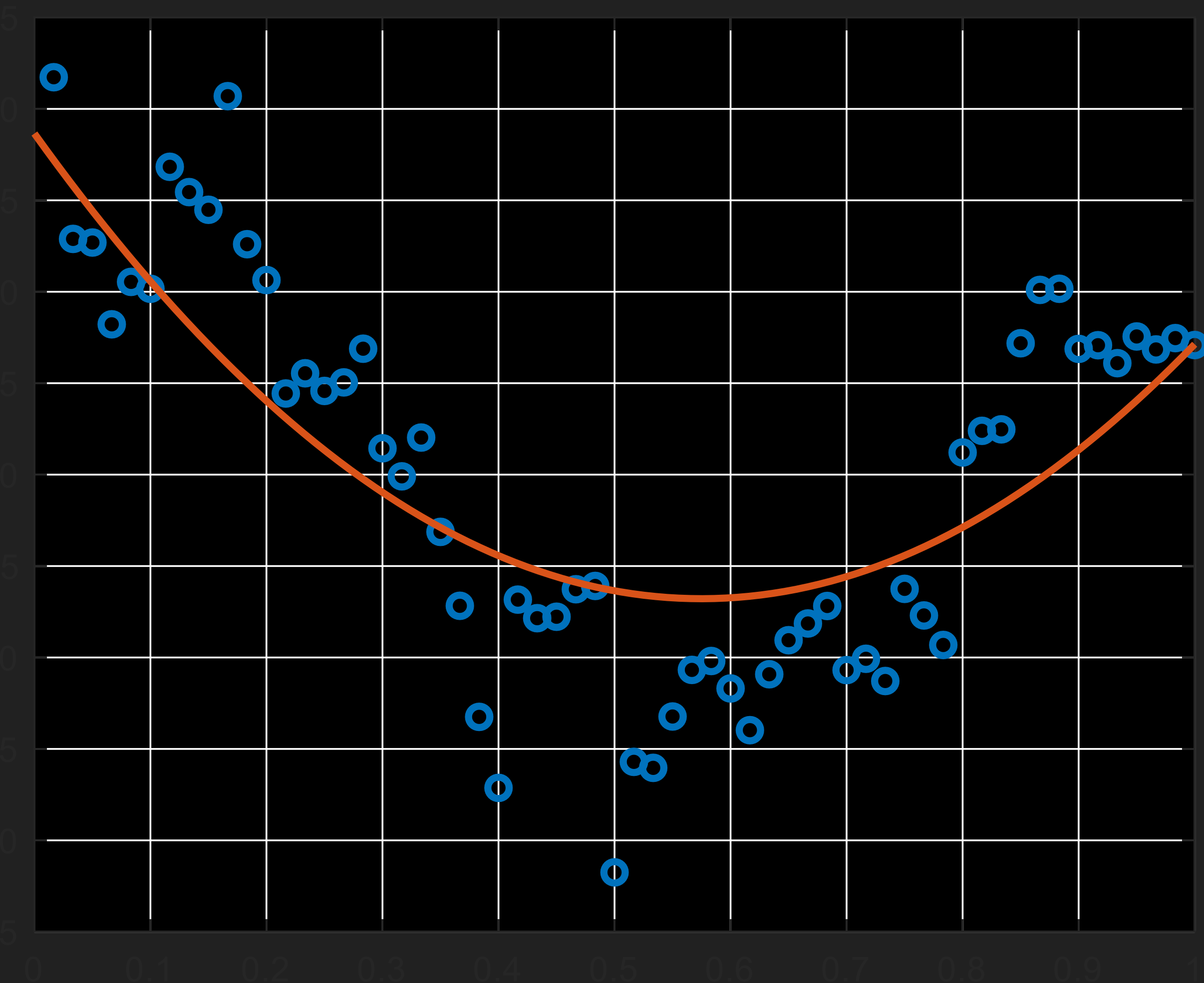
Example: Apple Stock Price, Feb 2019



- RBF kernel
  - $\sigma = 0.106$
  - $\lambda = 0.1.$
- Tr error: 666.20

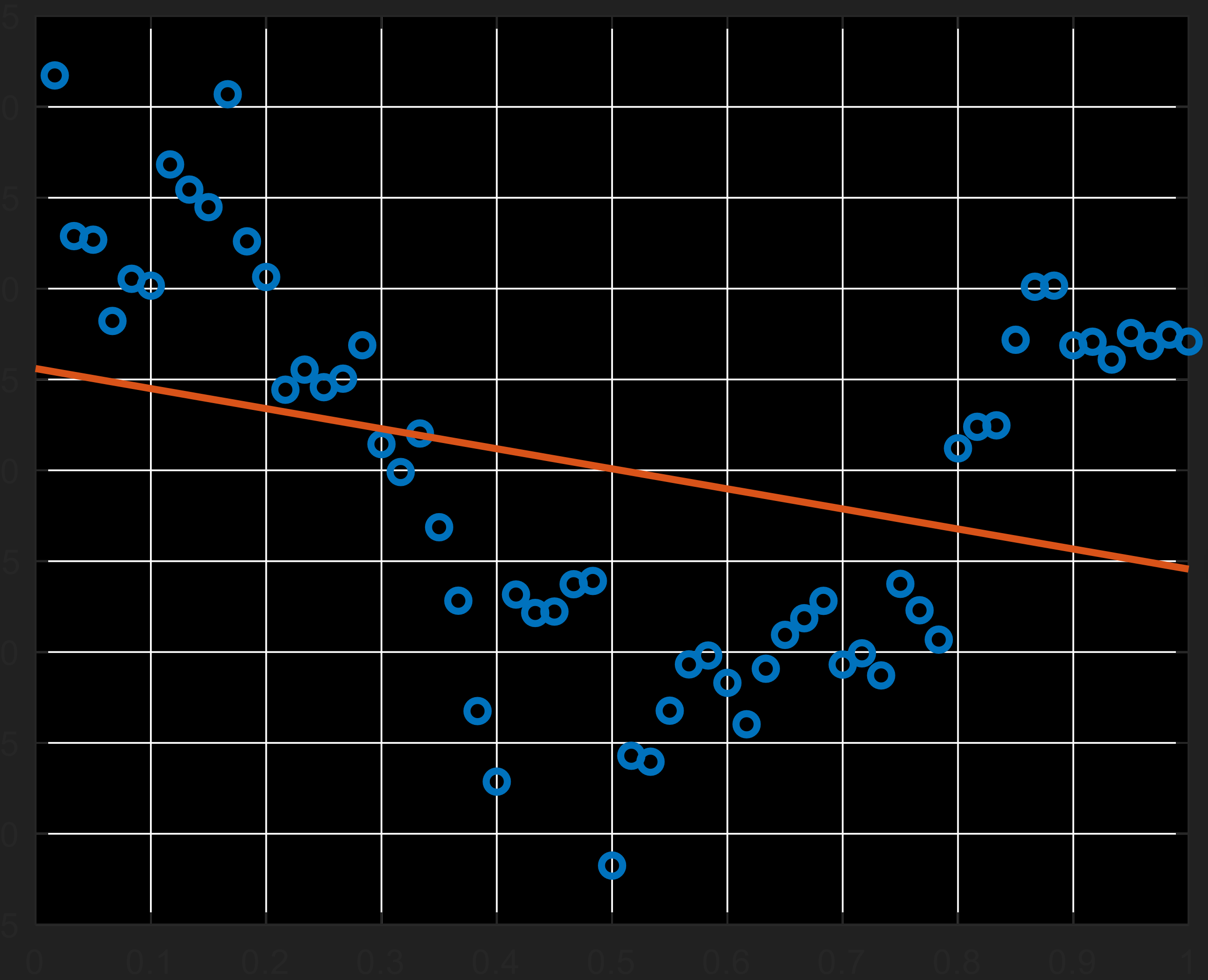
• Song Liu (song.liu@bristol.ac.uk), Lecturer in Data Science and A.I.

Example: Apple Stock Price, Feb 2019



- Poly. kernel
  - $b = 2$
  - $\lambda = 0.1.$
- Tr error: 2068.1

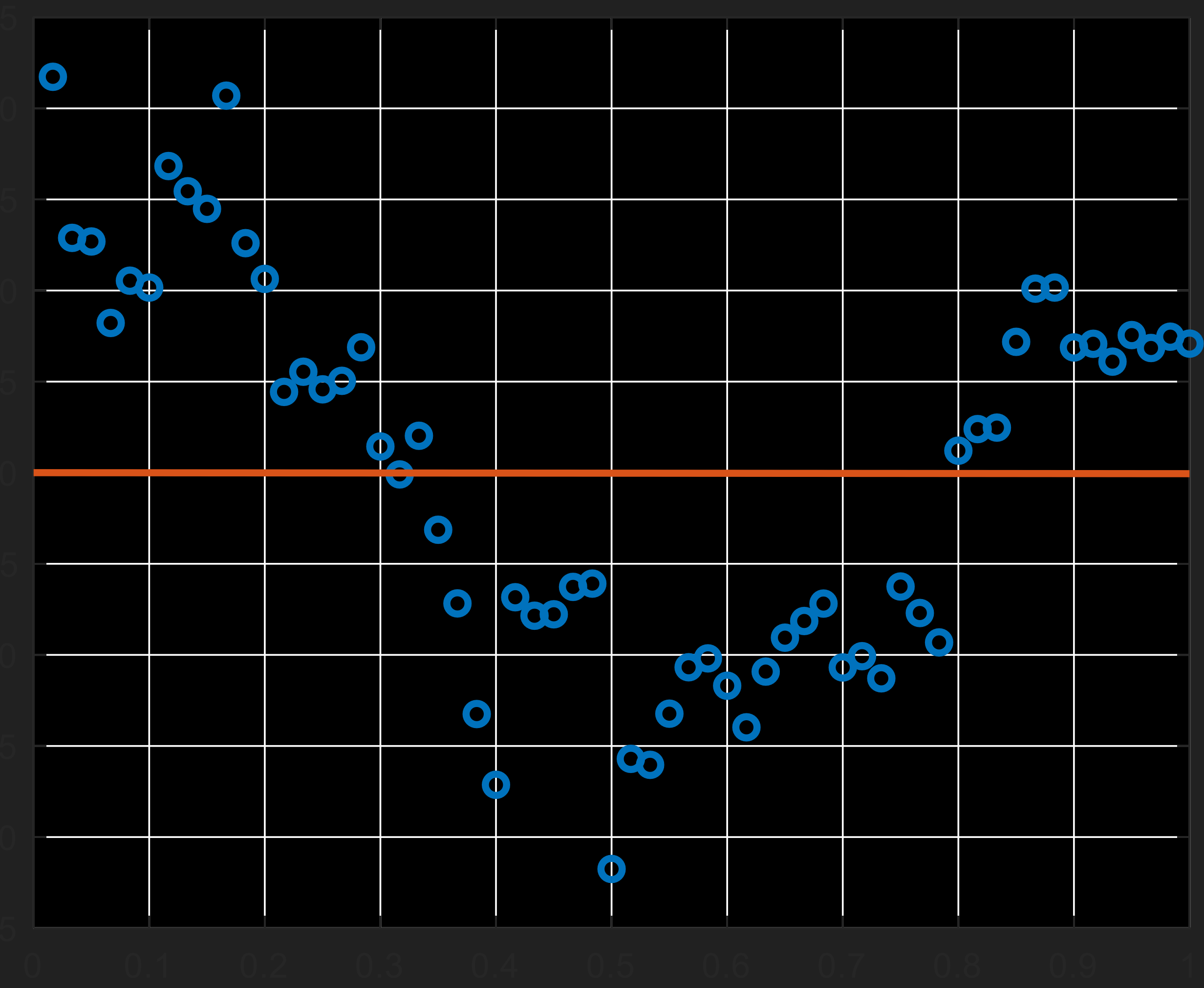
Example: Apple Stock Price, Feb 2019



- Linear kernel
  - $\lambda = 0.1.$
- Tr error: 5964

• Song Liu (song.liu@bristol.ac.uk), Lecturer in Data Science and A.I.

Example: Apple Stock Price, Feb 2019



- Linear kernel
  - $\lambda = 1000.$
- Tr error: 6597

Conclusion

- Kernel methods transform original data point into higher dimensional (potentially **infinitely dim.**) feature vectors.
  - We get super flexible $\hat{y}$.
  - Use regularization to combat overfitting caused by flexibility.
- Computation of inf. dimensional features is made possible by kernel trick.
- Important kernel functions:
  - Linear, polynomial, RBF.

Proper Names

- Numerical Hack #1 is called **Regularization** in statistics, usually used when handling high dimensional data.
- Numerical Hack #2,3 are called "**kernel tricks**", usually used for hiding $\boldsymbol{f}(\boldsymbol{x})$ inside inner products.
  - Other types of kernel tricks exist.

- **Please try to solve problems before attending the problem class next week!**