

Kernel Methods: An Infinity Game

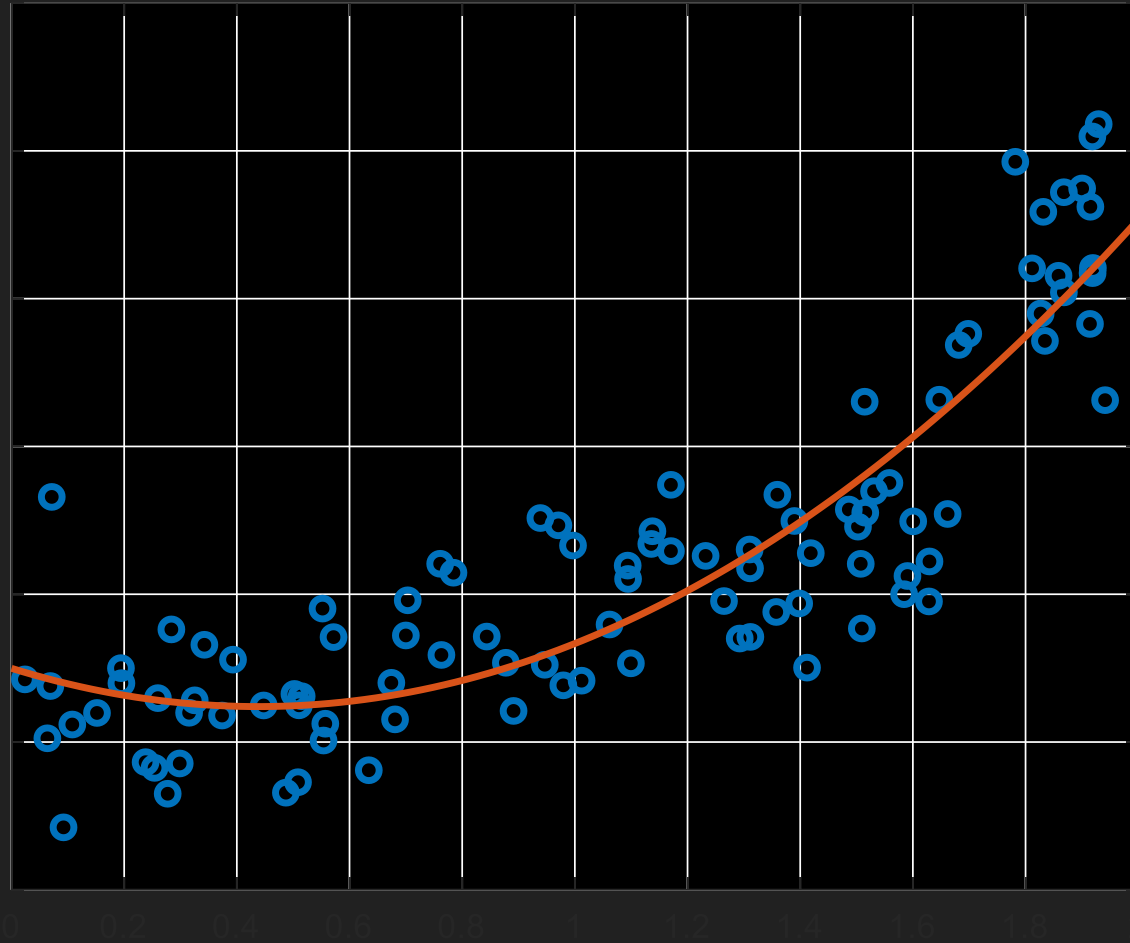
COMS21202, Part III

- 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.

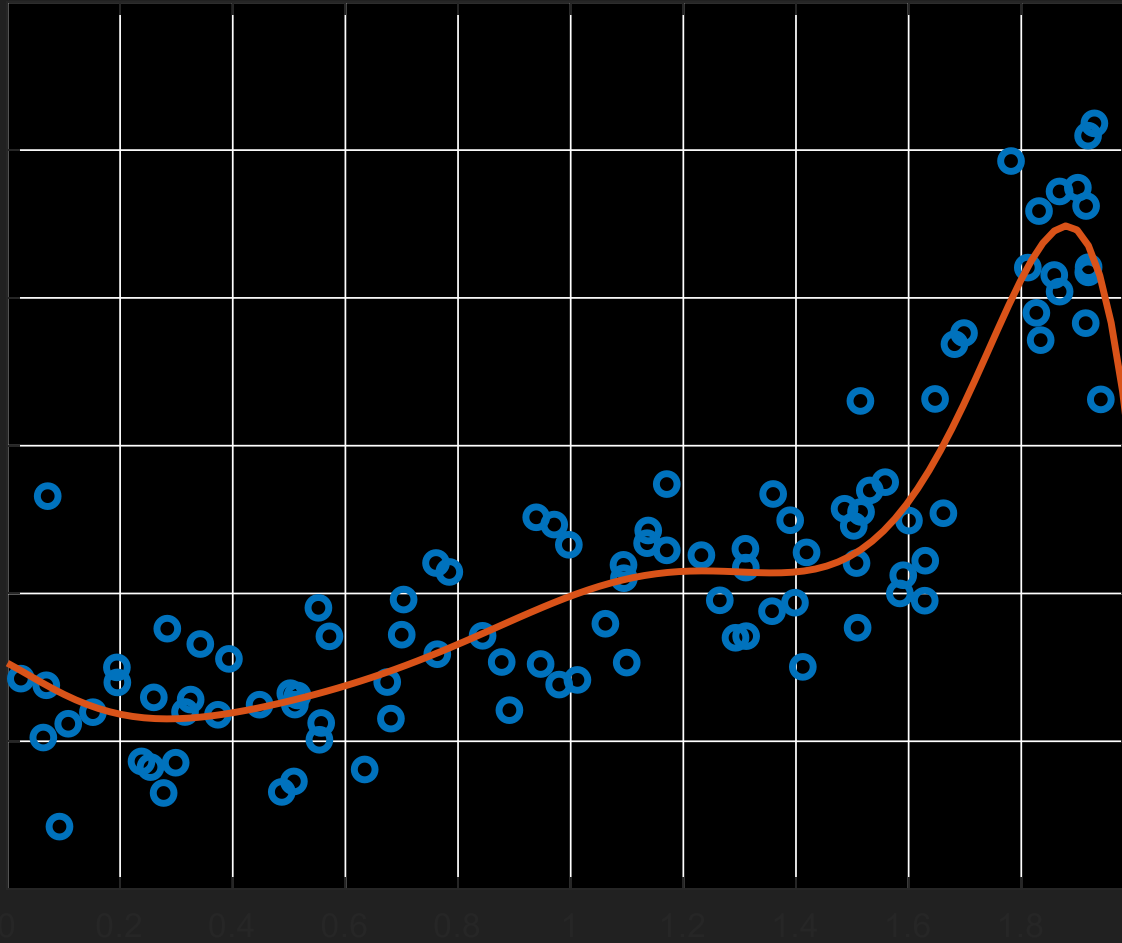
Recall: $y = \exp(1.5x - 1) + \epsilon, \epsilon \sim N(0,1)$



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

A Numerical Hack: Regularized LS Solution

- Instead of calculating

- $\hat{\beta} := (F^T F)^{-1} F^T y$

- We calculate

- $\hat{\beta} := (F^T F + \lambda I)^{-1} F^T y$

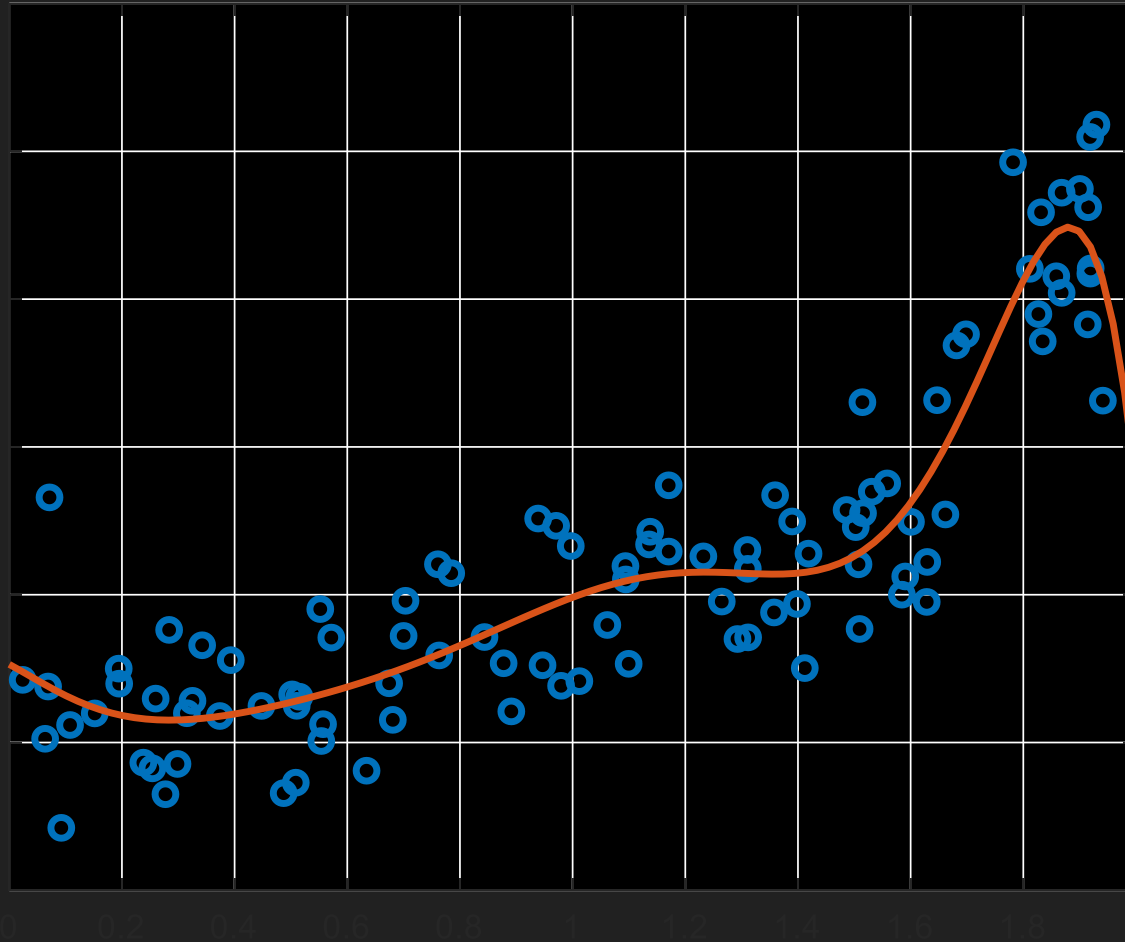
- Where $I \in R^{m \times m}$ is identity matrix.

- λ is some small value, say 0.01.

Regularized LS Solution and Overfitting

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

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



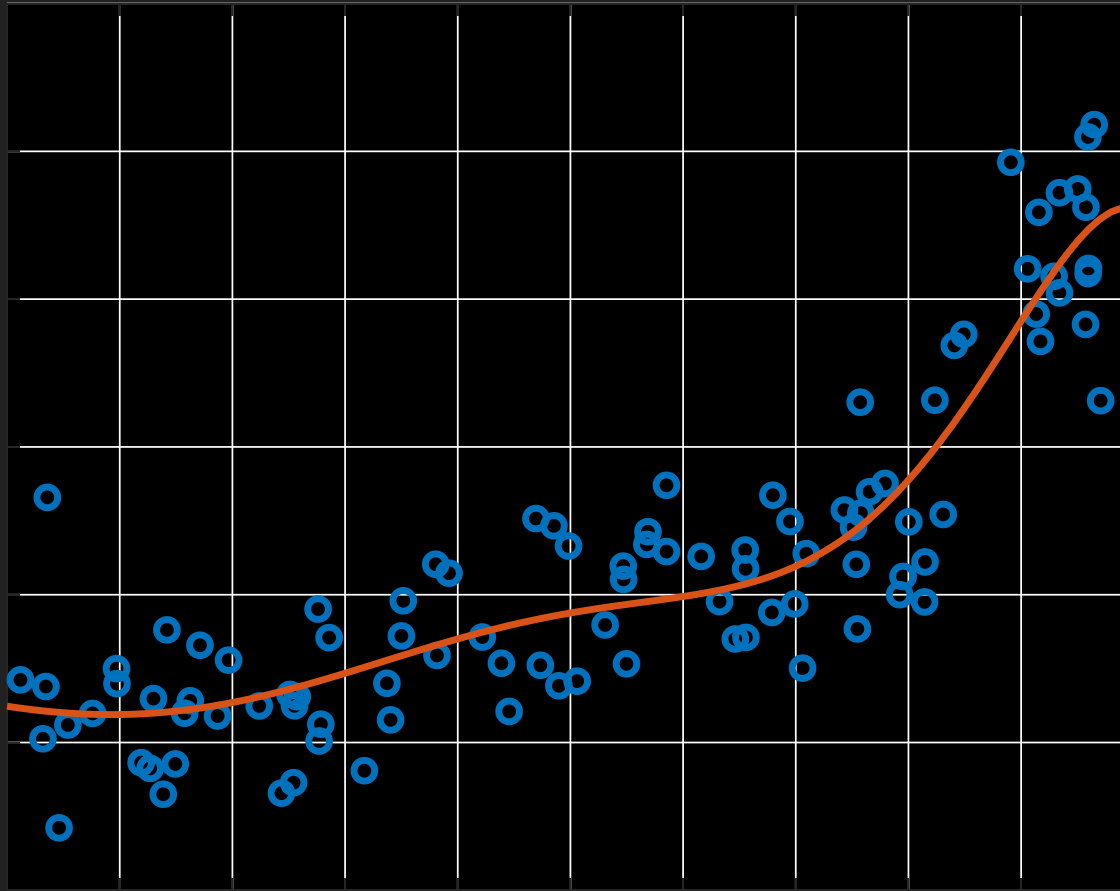
Polynomial
transform
with $b = 8$.

$\lambda = 0$

Tr. error:
78.87

Te. error:
128.01

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



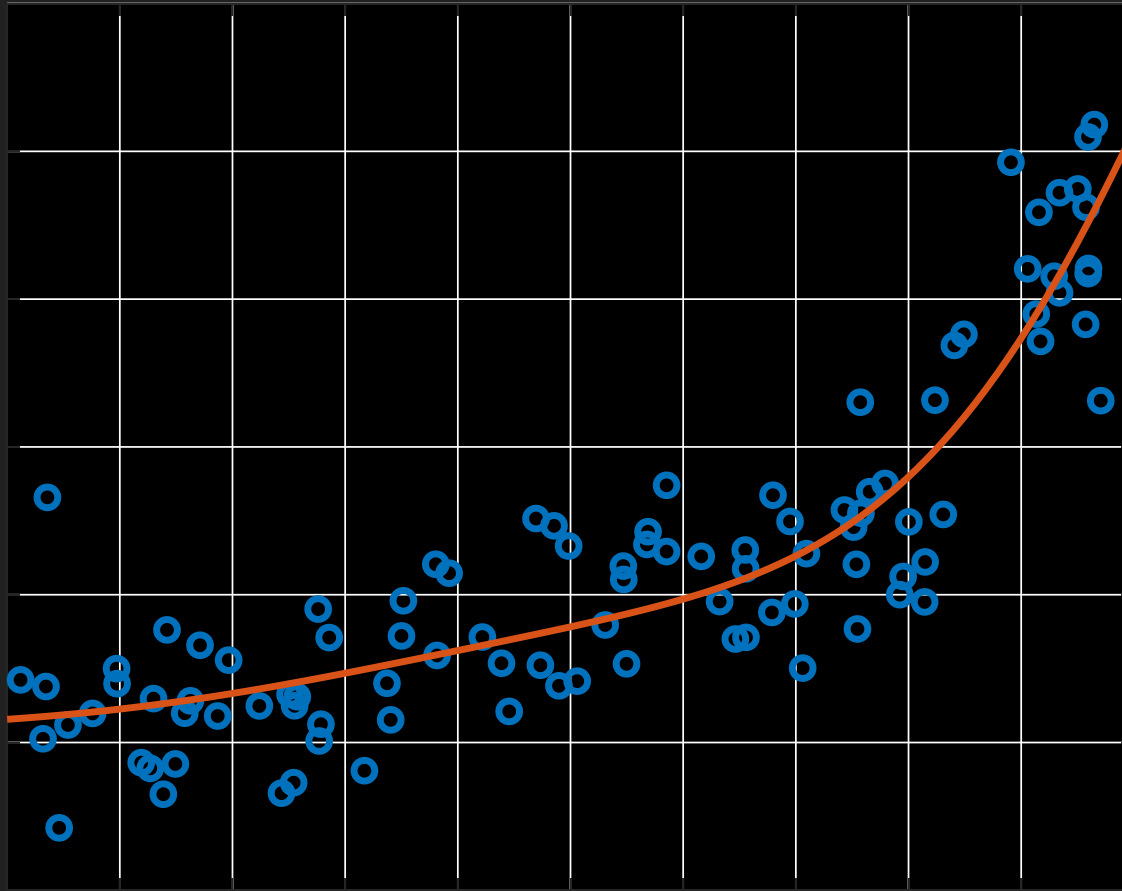
Polynomial
transform
with $b = 8$.

$\lambda = .1$

Tr. error:
86.46

Te. error:
112.47

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 f .

Can we still raise the game?



Can we design $f(x)$ transforms the original x into a **infinitely dim. vector**?

○ It should create a super flexible \hat{y} !

○ Recall $\hat{\beta} := (F^T F + \lambda I)^{-1} F^T y$

○ Problem: now $F^T F \in R^{m \times m}$, m is infinity.

○ How do you store F in computer??

Numerical Hack, #2:

Rewrite Solution using Woodbury identity

- Remarkably,

- $$\hat{\beta} := (F^T F + \lambda I)^{-1} F^T y = F^T (F F^T + \lambda I)^{-1} y$$

- Hint, Woodbury identity:

- $$(P^{-1} + B^T B)^{-1} B^T = P B^T (B P B^T + I)^{-1}$$

- Live demonstration

Numerical Hack, #2:

Rewrite Solution using Woodbury identity

- $\hat{\beta} := F^T (FF^T + \lambda I)^{-1} y$

- Now instead of $F^T F \in R^{m \times m}$, we just need to compute $FF^T \in R^{n \times n}$.

- Let $K := FF^T$, where

- $K^{(i,j)} = k(x_i, x_j) := \langle f(x_i), f(x_j) \rangle,$

- i.e., $k(x_i, x_j)$ is the inner product of two m dimensional feature transform.

Numerical Hack, #2:

Rewrite Solution using Woodbury identity

- $\hat{\mathbf{y}} = \langle \hat{\boldsymbol{\beta}}, \mathbf{f}(\mathbf{x}_0) \rangle$

- $$\begin{aligned}\hat{\mathbf{y}} &= \langle \mathbf{f}(\mathbf{x}_0), \mathbf{F}^\top (\mathbf{F}\mathbf{F}^\top + \lambda \mathbf{I})^{-1} \mathbf{y} \rangle \\ &= \langle \mathbf{f}(\mathbf{x}_0)\mathbf{F}^\top, (\mathbf{F}\mathbf{F}^\top + \lambda \mathbf{I})^{-1} \mathbf{y} \rangle\end{aligned}$$

- Rewrite $\mathbf{f}(\mathbf{x}_0)\mathbf{F}^\top$ as $\mathbf{k} \in \mathbb{R}^n$ we can see

- $k^{(i)} = k(\mathbf{x}_0, \mathbf{x}_i) = \langle \mathbf{f}(\mathbf{x}_0), \mathbf{f}(\mathbf{x}_i) \rangle$

- Verify this your self!

Numerical Hack, #3:

Evaluating only the Inner Products

- $\hat{\mathbf{y}} := \mathbf{k}(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}$

- Note how $\mathbf{f}(\mathbf{x})$ only appears in the form of inner products!

- 💡 Even if we cannot write $\mathbf{f}(\mathbf{x})$ explicitly, we may still compute its inner product!

- design “an inner product function \mathbf{k} ” mimics behaviour of inner product .

- Forget about the existence of \mathbf{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 f .
 - We usually say a k induces an f
 - **Even** if we cannot write f down.
- $k(x_i, x_j)$ is called **kernel function**.

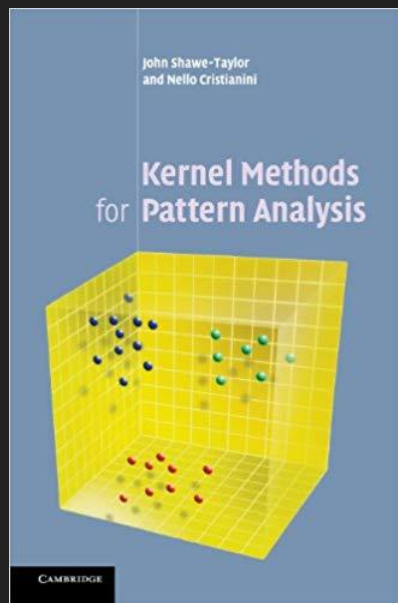
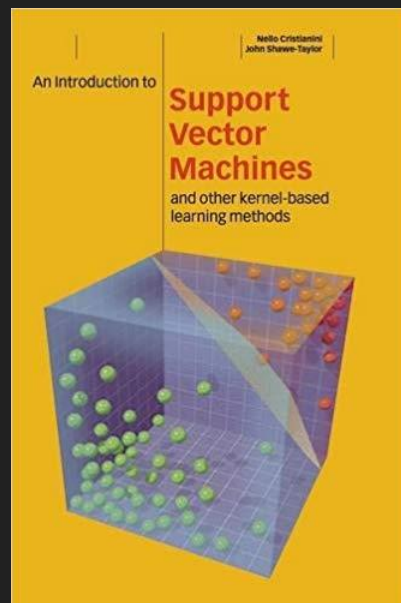
The History of Kernel Methods

- 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.



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

Choices of k

- Linear kernel function:

- $k(\mathbf{x}_i, \mathbf{x}_j) := \langle \mathbf{x}_i, \mathbf{x}_j \rangle$

- Implicit feature transform $\mathbf{f}(\mathbf{x}) = \mathbf{x}$.

- Polynomial kernel function with degree b :

- $k(\mathbf{x}_i, \mathbf{x}_j) := (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^d$

- PC: write down induced $\mathbf{f}(\mathbf{x})$ by polynomial kernels $b = 2$.

Choices of k

- RBF (or Gaussian) kernel:

- $k(\mathbf{x}_i, \mathbf{x}_j) := \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right)$

- $f(\mathbf{x})$ 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.

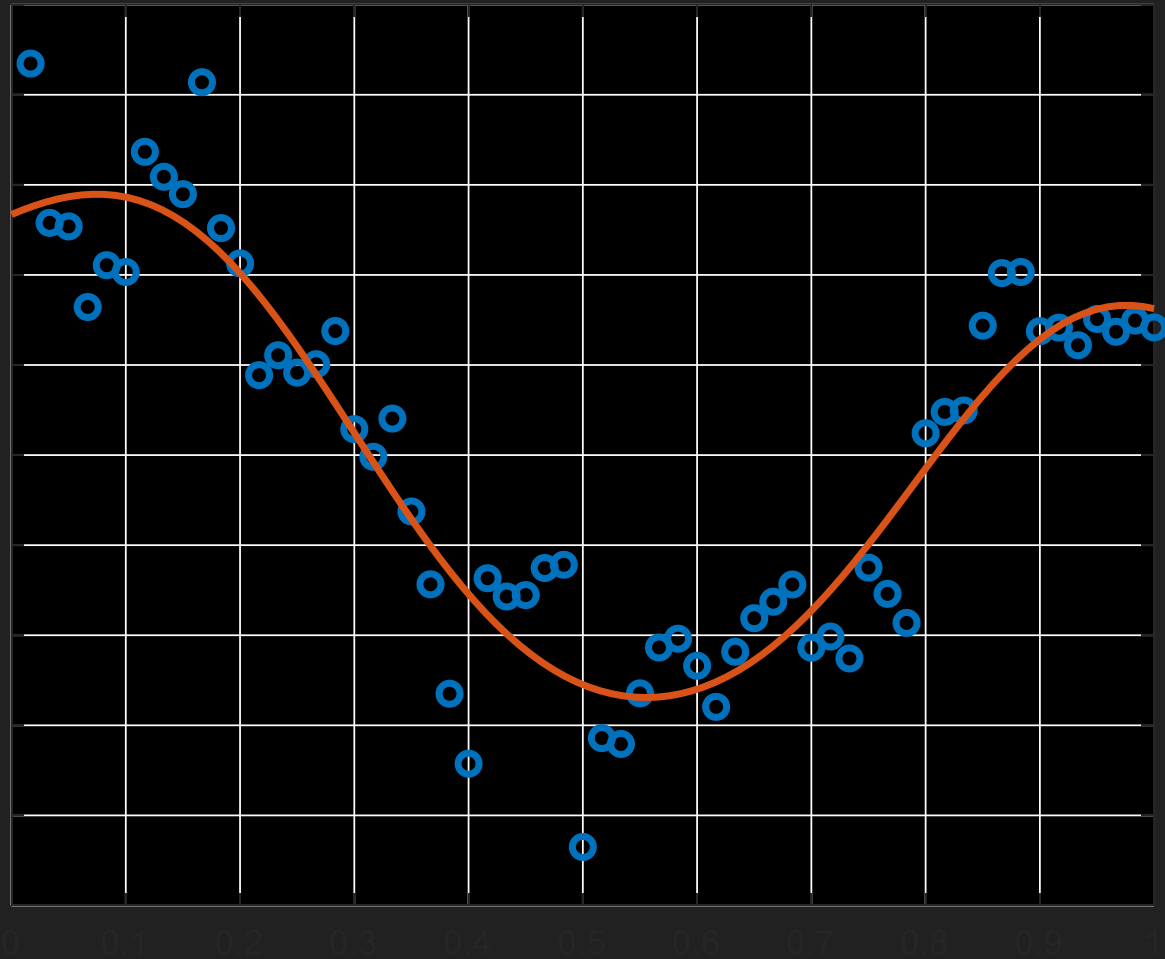
- Domain knowledge matters!!

- RBF kernel is a good all-rounded choice for $\mathbf{x} \in R^d$.

Implementation of Kernel LS

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

Example: Apple Stock Price, Feb 2019



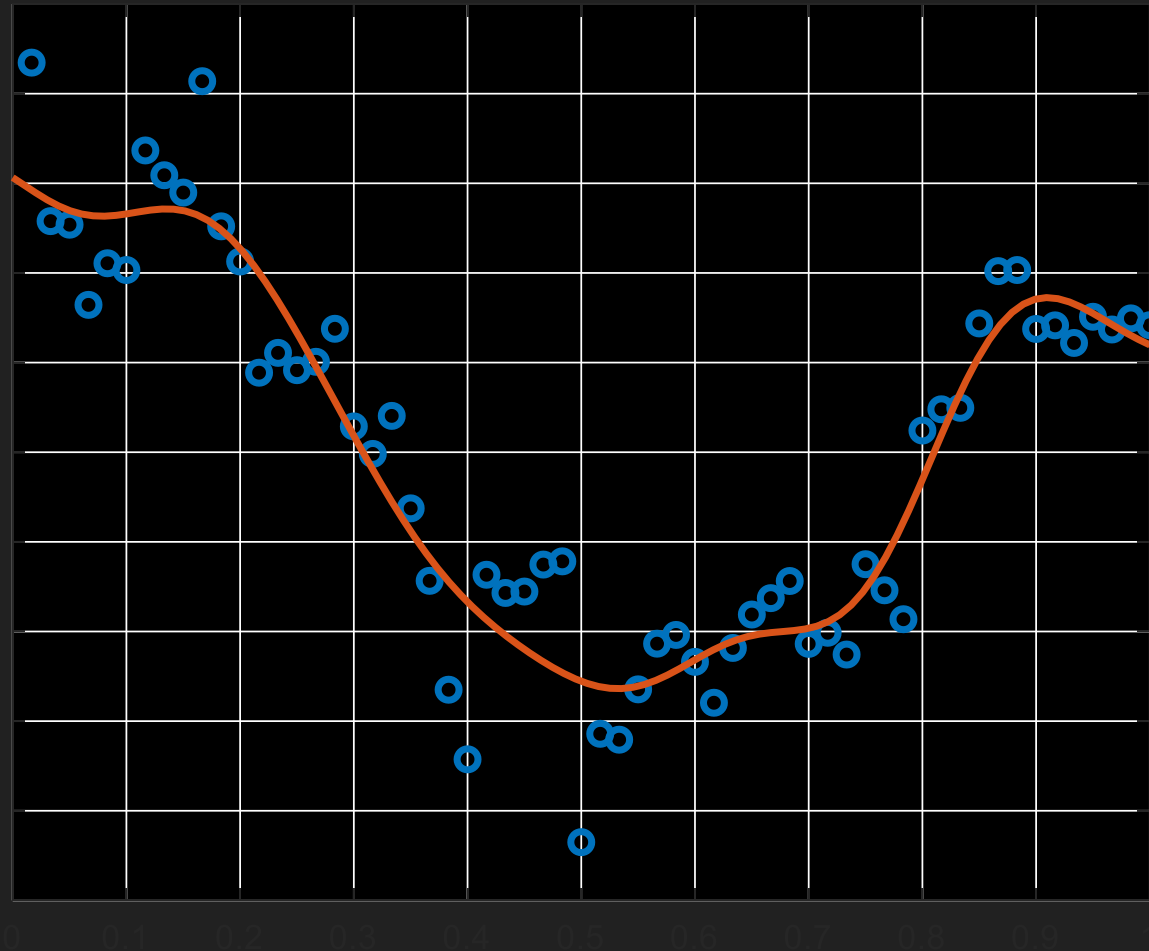
○ RBF kernel

○ $\sigma = 0.2121$

○ $\lambda = 0.1$.

○ Tr error:
833.58

Example: Apple Stock Price, Feb 2019



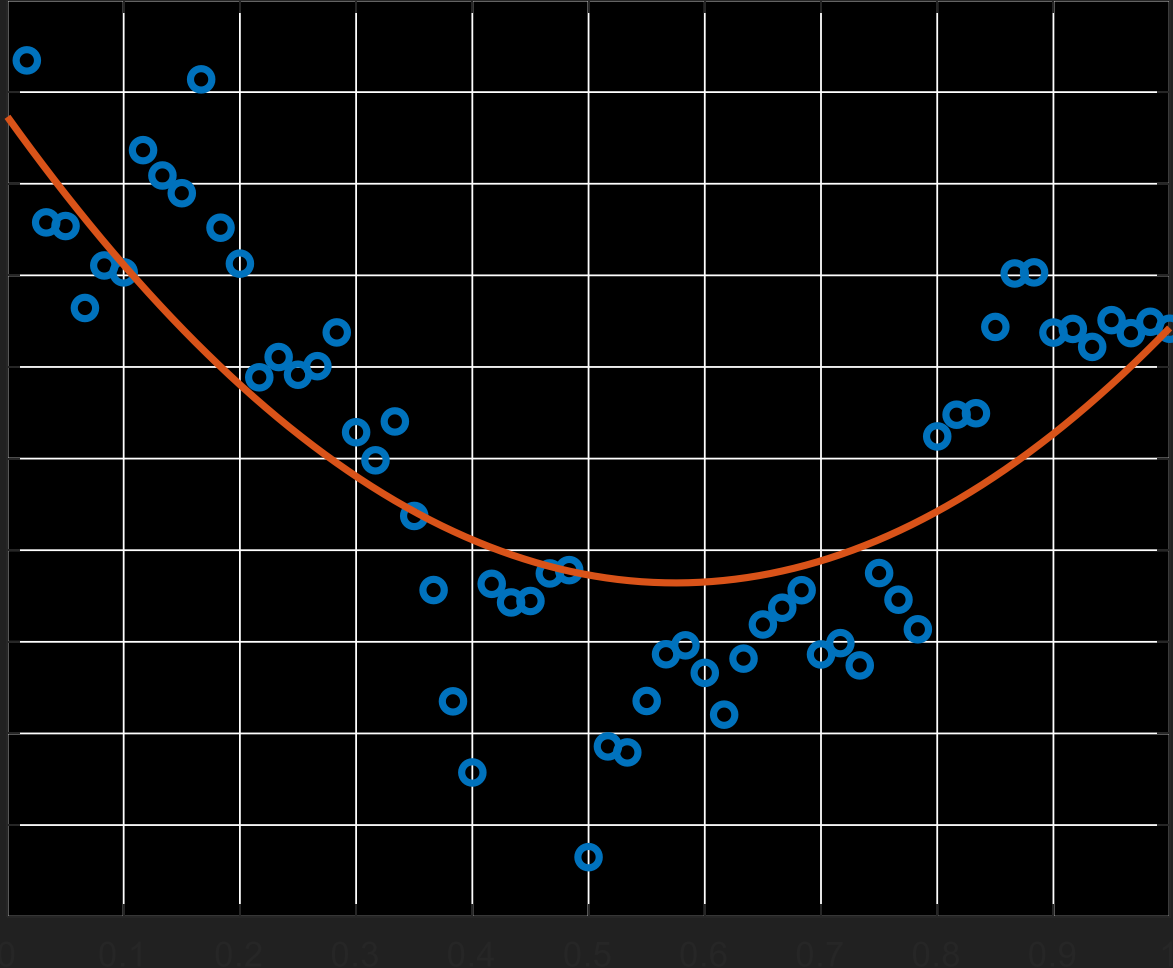
○ RBF kernel

○ $\sigma = 0.106$

○ $\lambda = 0.1$.

○ Tr error:
666.20

Example: Apple Stock Price, Feb 2019



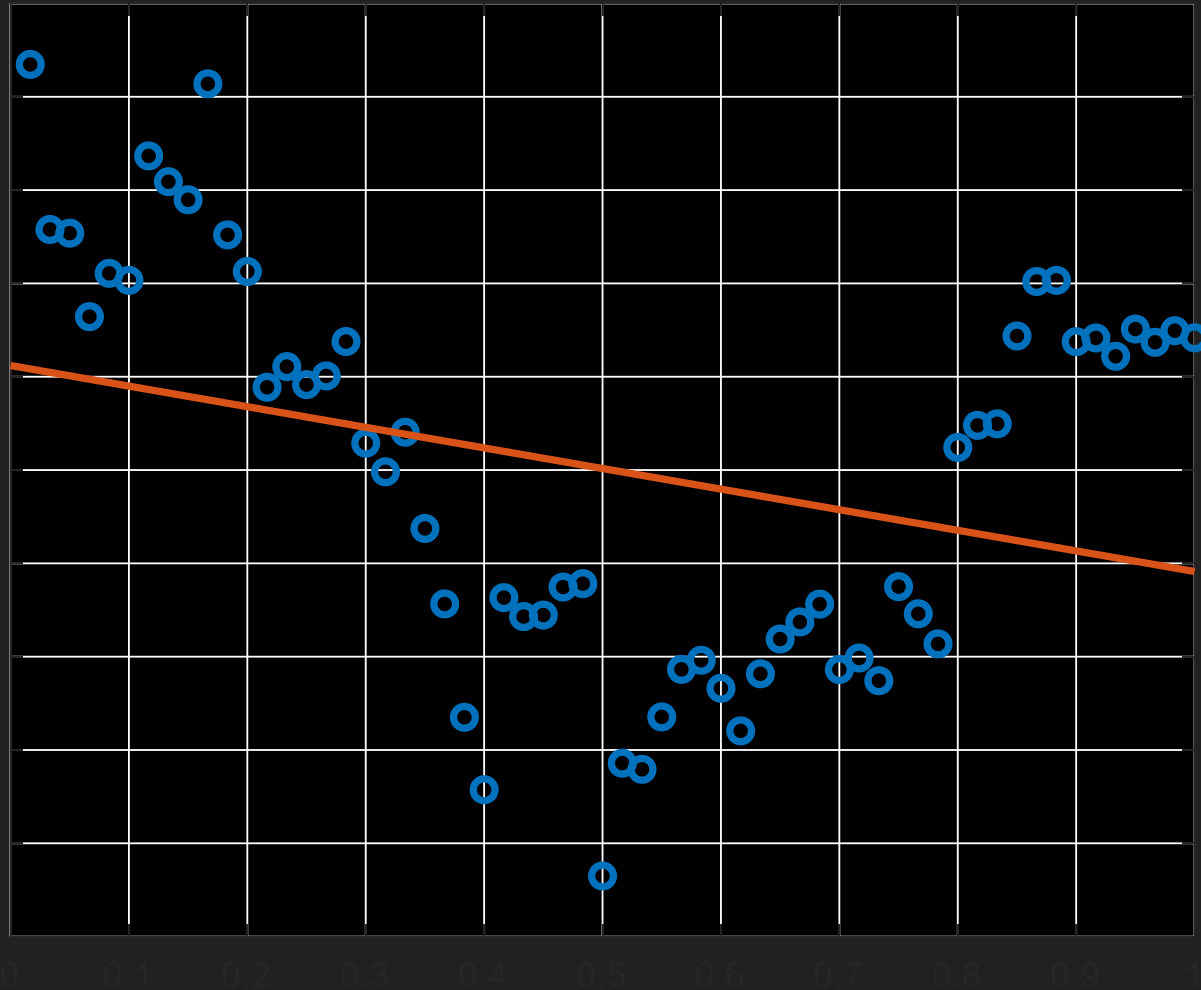
○ Poly. kernel

○ $b = 2$

○ $\lambda = 0.1$.

○ Tr error:
2068.1

Example: Apple Stock Price, Feb 2019

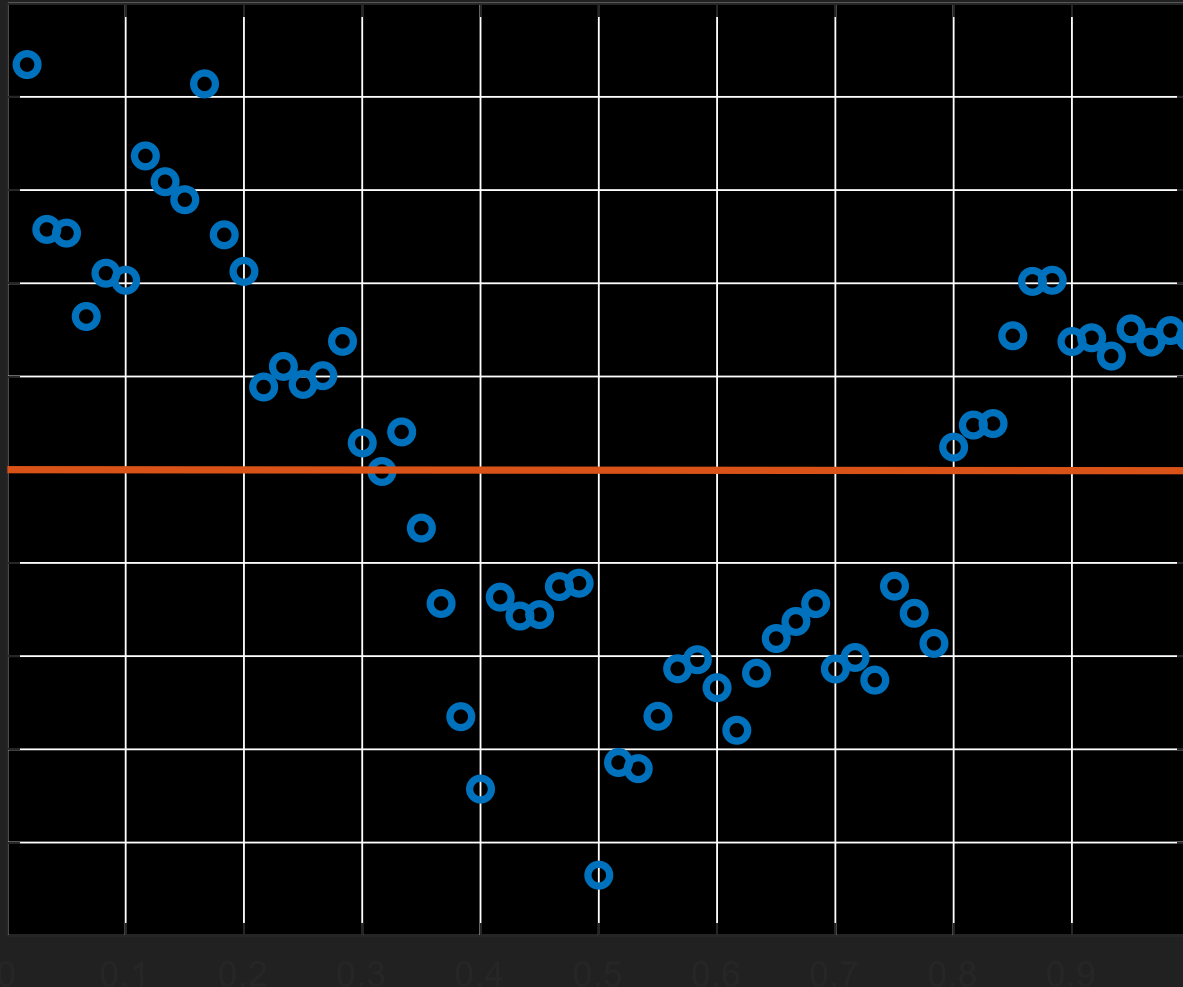


○ Linear kernel

○ $\lambda = 0.1$.

○ Tr error: 5964

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 $f(x)$ inside inner products.
 - Other types of kernel tricks exist.

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