Feature Transform and Kernel Methods

Song Liu (song.liu@bristol.ac.uk)

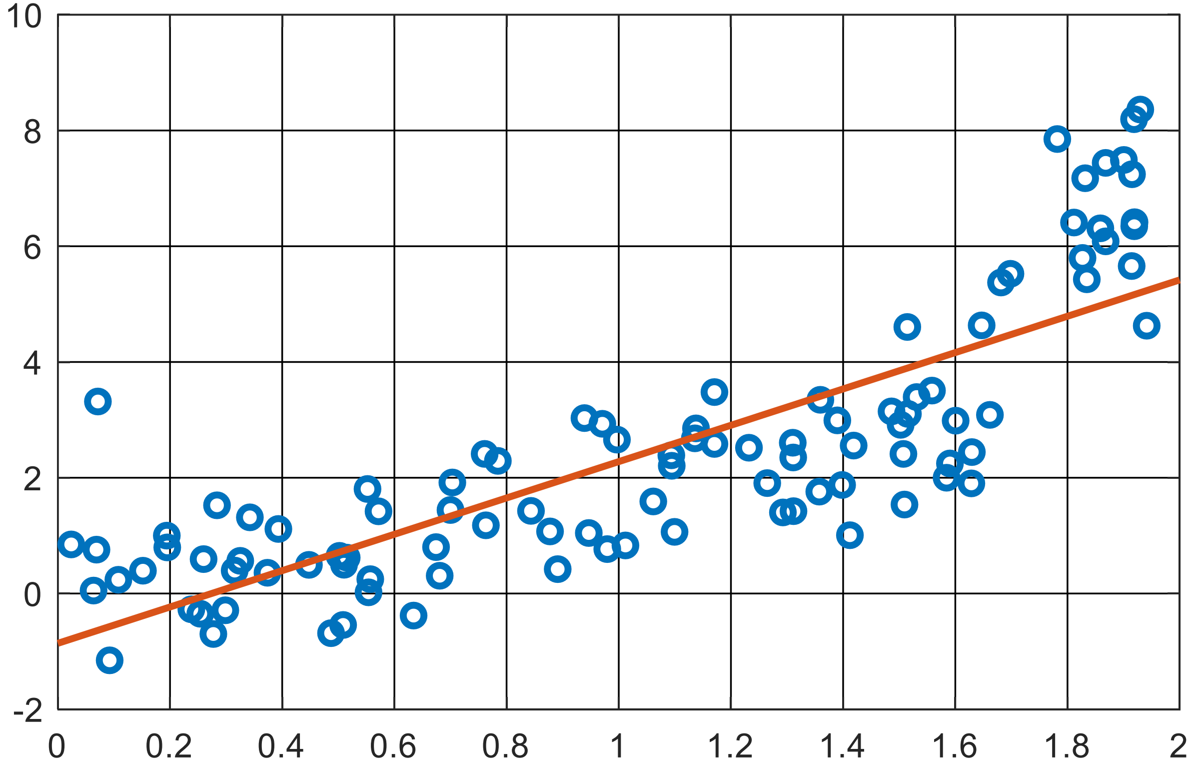
LS with Feature Transform

$$\boldsymbol{w}_{\mathrm{LS}} := \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{i \in D_0} [y_i - f'(\boldsymbol{x}_i; \boldsymbol{w})]^2$$
$$f'(\boldsymbol{x}; \boldsymbol{w}) := \langle \boldsymbol{w}_1, \boldsymbol{\phi}(\boldsymbol{x}) \rangle + w_0, \boldsymbol{w} := [\boldsymbol{w}_1, w_0]^\top$$

- $\boldsymbol{\phi}(\boldsymbol{x})$: $R^d \to R^b$, is called a feature transform.
  - $\boldsymbol{\phi}(\boldsymbol{x}) := \boldsymbol{x}$, Linear transform.
  - $\boldsymbol{\phi}(x) := [x, x^2, x^3, \ldots, x^b\ ]^\top$, Polynomial transform
- $\boldsymbol{\phi}(\boldsymbol{X}) := \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1), \cdots, \boldsymbol{\phi}(\boldsymbol{x}_n) \\ 1, \cdots, 1 \end{bmatrix} \in R^{(b+1)\times n}$,
- Solution: $\boldsymbol{w}_{\mathrm{LS}} = (\boldsymbol{\phi}(\boldsymbol{X})\boldsymbol{\phi}(\boldsymbol{X})^\top)^{-1}\boldsymbol{\phi}(\boldsymbol{X})\boldsymbol{y}^\top$

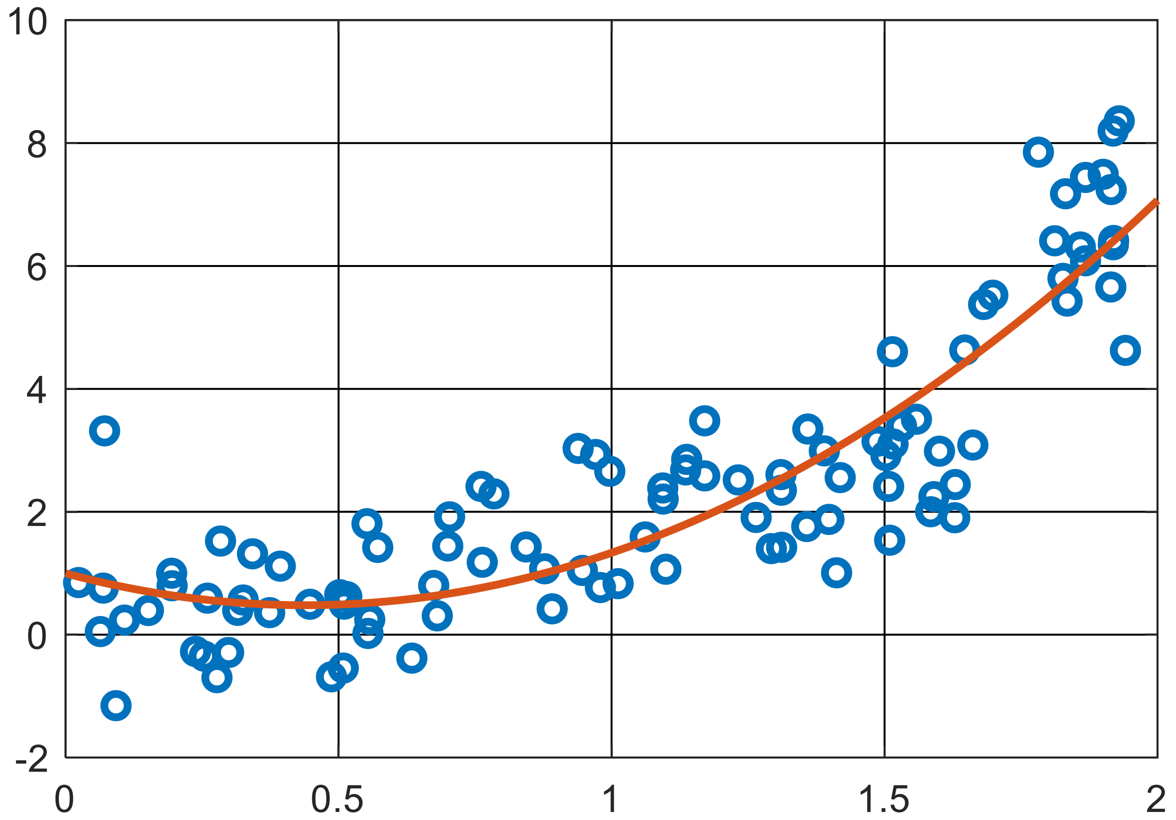
Polynomial Transform $b = 1$

$$y = g(x) + \epsilon, g(x) = \exp(1.5x - 1)\,, \epsilon \sim N(0, .64)$$



Polynomial Transform $b = 2$

$$y = g(x) + \epsilon, g(x) = \exp(1.5x - 1)\,, \epsilon \sim N(0, .64)$$



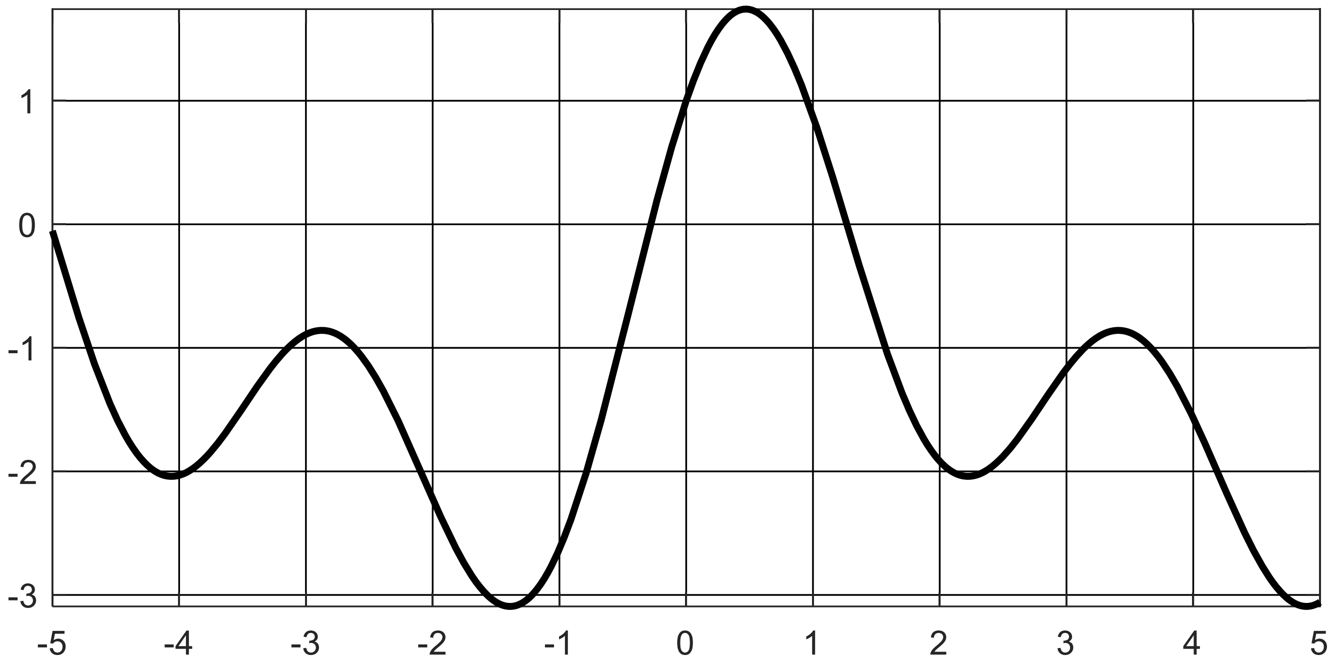
Why it works?

- 1-dimensional intuition: Taylor Series.
- Taylor Series of $g(x)$ at 0:
  - $g(x) = g(0)(x-0)^0 + g'(0)(x-0)^1 + \frac{g''(0)}{2!}(x-0)^2 + \frac{g'''(0)}{3!}(x-0)^3 + \cdots$
- You can approximate a **smooth** function using polynomial terms (at some cost).

Fourier Series

- What are **other ways** of decomposing a function?
- Suppose we have a periodic signal $g(x)$ over the time domain.
  - e.g. a sound wave or a stock price
  - $g(x) = a_0 + \sum_{i=1}^{\infty} [a_i \sin(ix) + b_i \cos(ix)]$
  - This decomposition is called Fourier Series.

Fourier Series



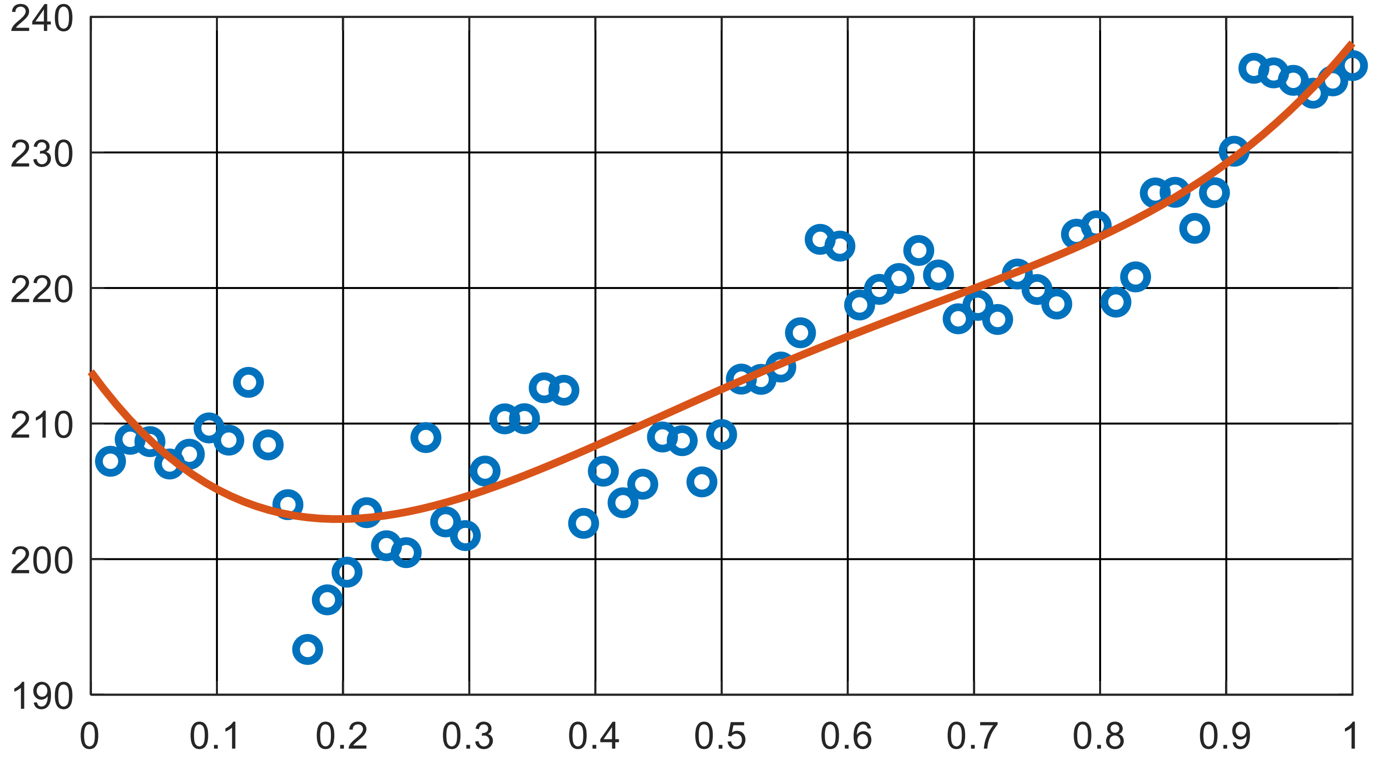
- $g(x) = \sin(x) + \cos(x) + \sin(2x) + \cos(2x)$

Trigonometric Transform

- Trigonometric Transform is usually used to approximate $g(x)$ over time domain.
  - $\boldsymbol{\phi}(x) \coloneqq [\sin(x), \cos(x), \sin(2x), \cos(2x) \ldots \sin(bx), \cos(bx)]$
  - $\boldsymbol{\phi}(x) \in R^{2b}$

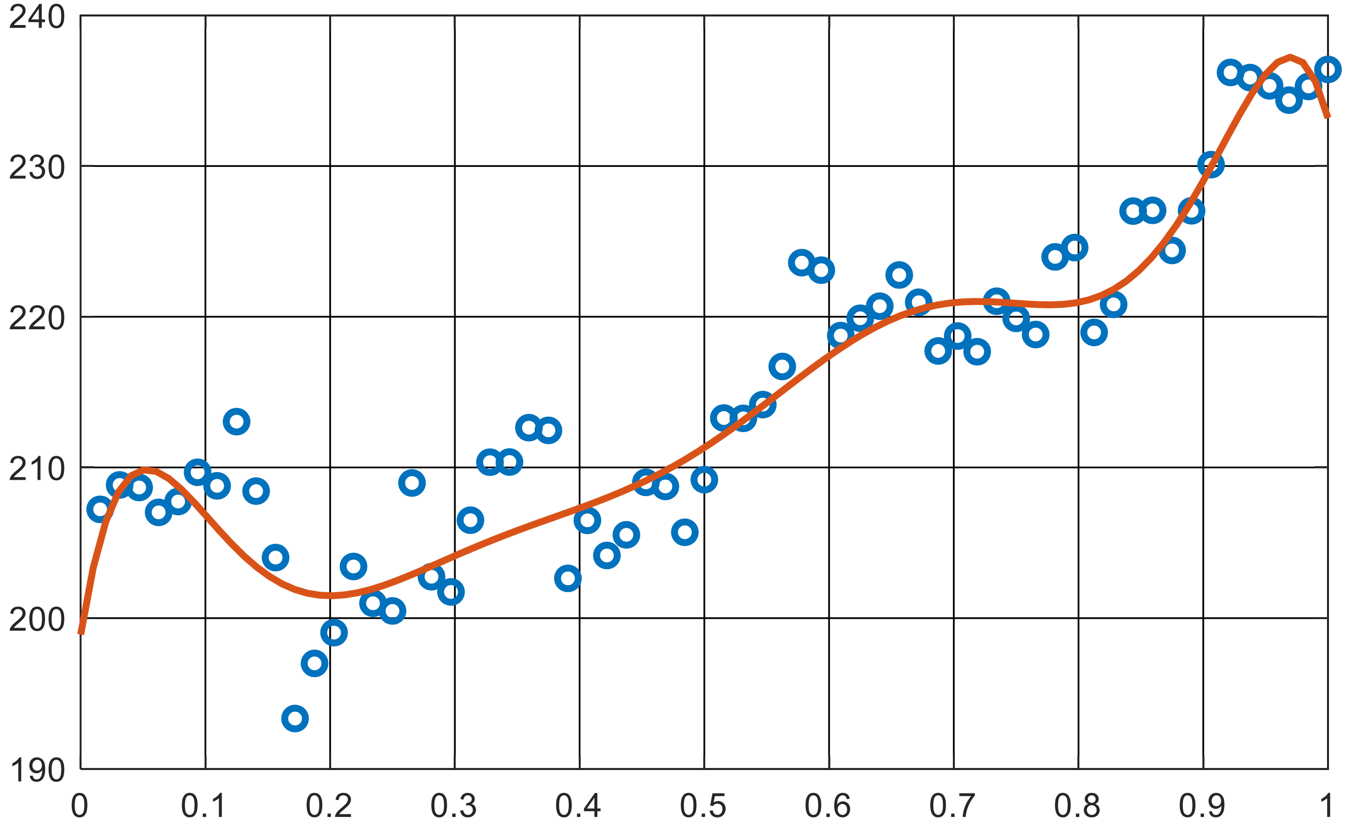
APPL stock price, Jul-Oct, 2019

- Trigonometric Transform
- $b = 2$



APPL stock price, Jul-Oct, 2019

- Trigonometric Transform
- $b = 4$



Linear Expansion of Basis Functions

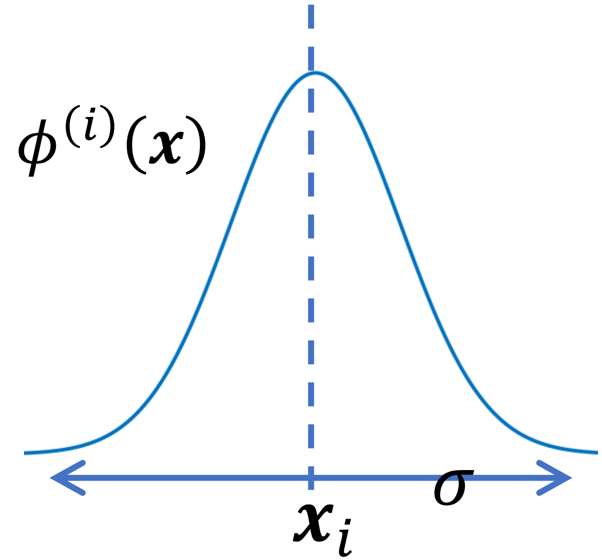
- Polynomial and Trigonometric transforms based on the idea a function can be approximated by:
  - $g(\boldsymbol{x}) \approx f(\boldsymbol{x}; \boldsymbol{w})$

$$= \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle = \sum_{i=1} w^{(i)} \phi^{(i)}(\boldsymbol{x})$$

  - called a **linear basis expansion** of $g(\boldsymbol{x})$
  - $\phi^{(i)}$ are called **basis function**
    - Polynomial basis, Trigonometric basis…

Radial Basis Function (RBF)

- RBF is another widely used basis function for regression tasks.



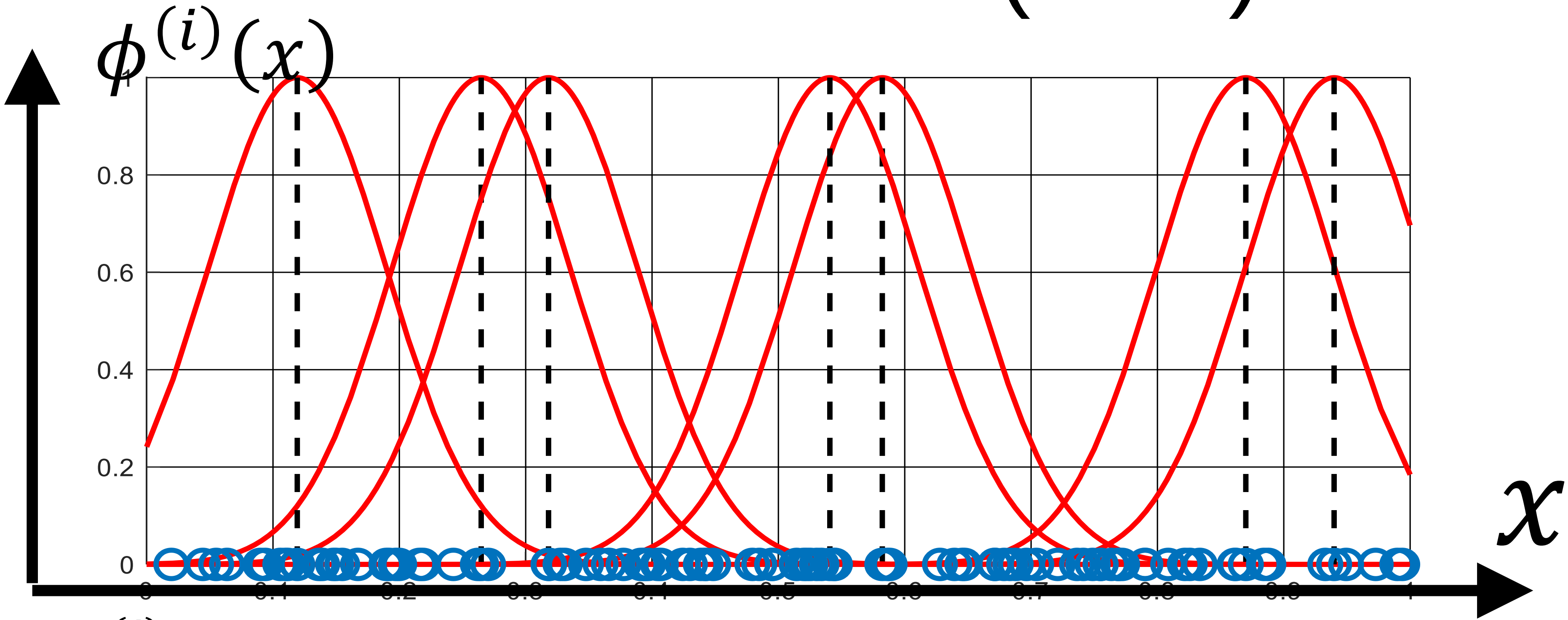
- $\phi^{(i)}(\boldsymbol{x}) \coloneqq \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{x}_i||^2}{2\sigma^2}\right)$
  - $\sigma > 0$ is called bandwidth
  - σ is determined before fitting

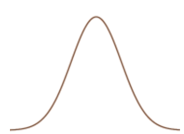
  - A practice is setting σ as the median of all pairwise distances of $\boldsymbol{x}$ in your dataset.

Radial Basis Function (RBF)

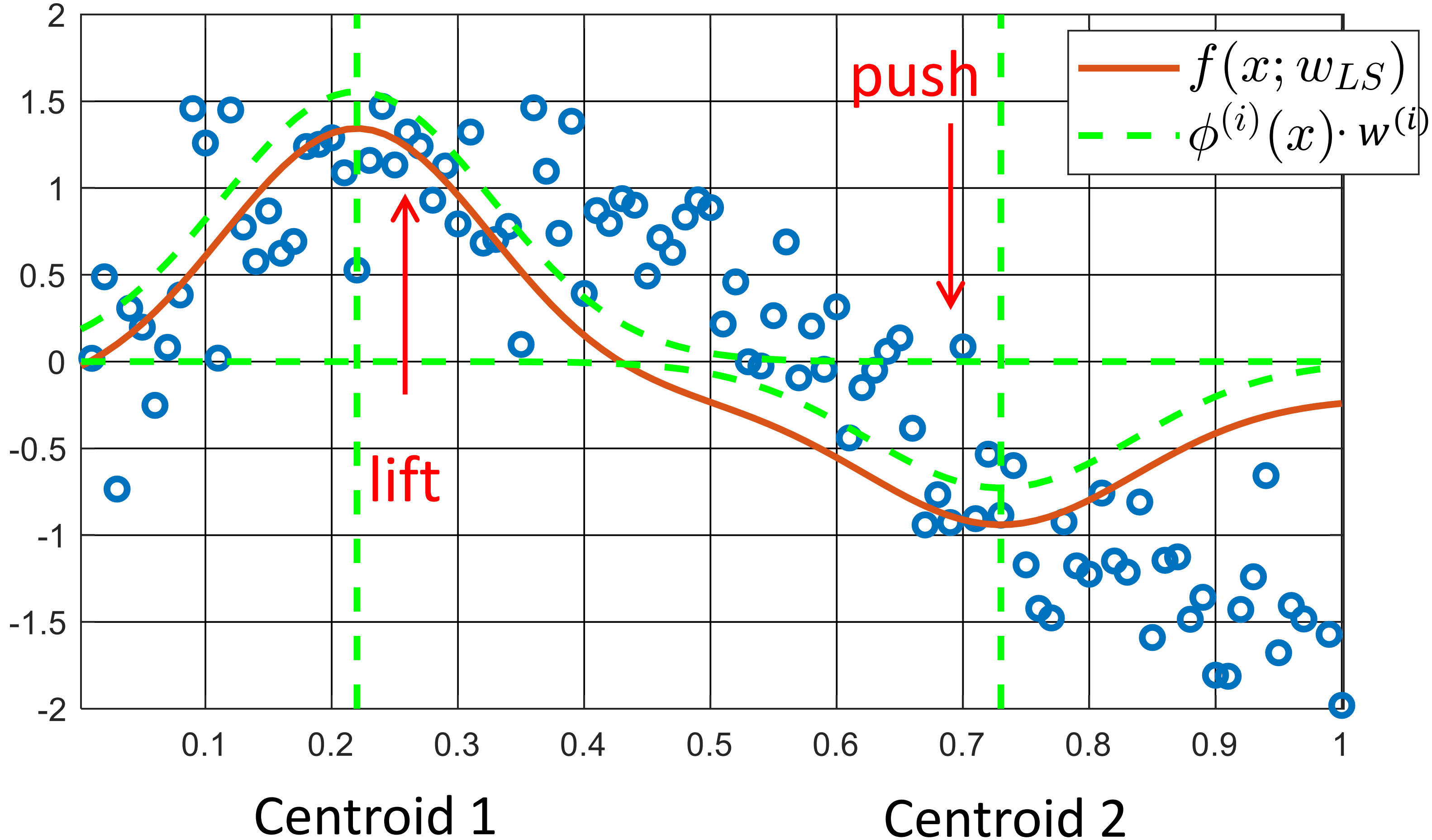
- $\boldsymbol{x}_i$ are called **RBF centroids**.
- $\boldsymbol{x}_i$ can be **randomly chosen** from the $\boldsymbol{x}$ in your dataset
- $\boldsymbol{\phi}(\boldsymbol{x}) \coloneqq [\phi^{(1)}(x), \phi^{(2)}(x), \dots, \phi^{(b)}(x)]$

Radial Basis Function (RBF)



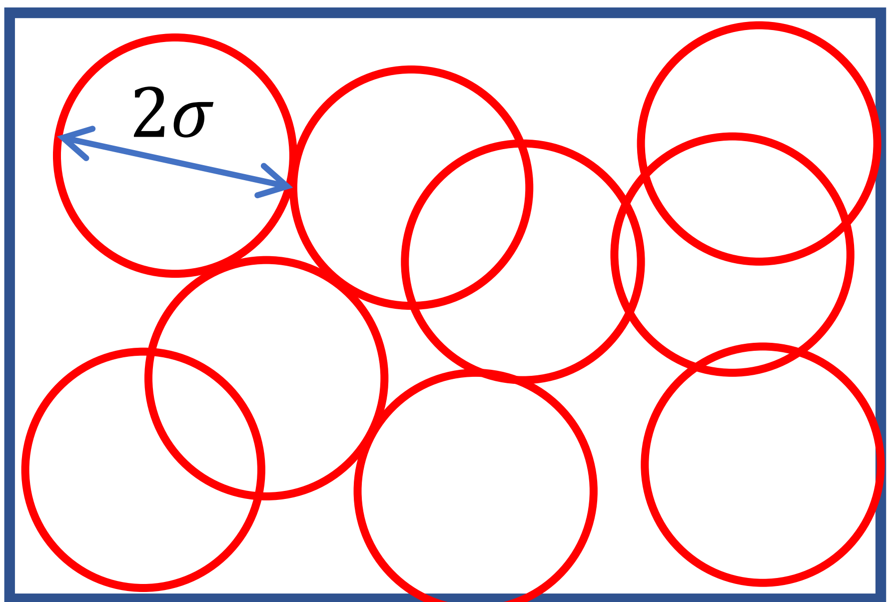
- $\phi^{(i)}(x)$ are visualized in red at random 7 centroids among 100 uniformly drawn x.
- At each "**bump**  ",
  - If $w^{(i)} > 0$, basis at $x^{(i)}$ gives $f(\boldsymbol{x}; \boldsymbol{w})$ a "lift".
  - If $w^{(i)} < 0$, basis at $x^{(i)}$ gives $f(\boldsymbol{x}; \boldsymbol{w})$ a "push".

RBF Feature Transform, $b = 2$



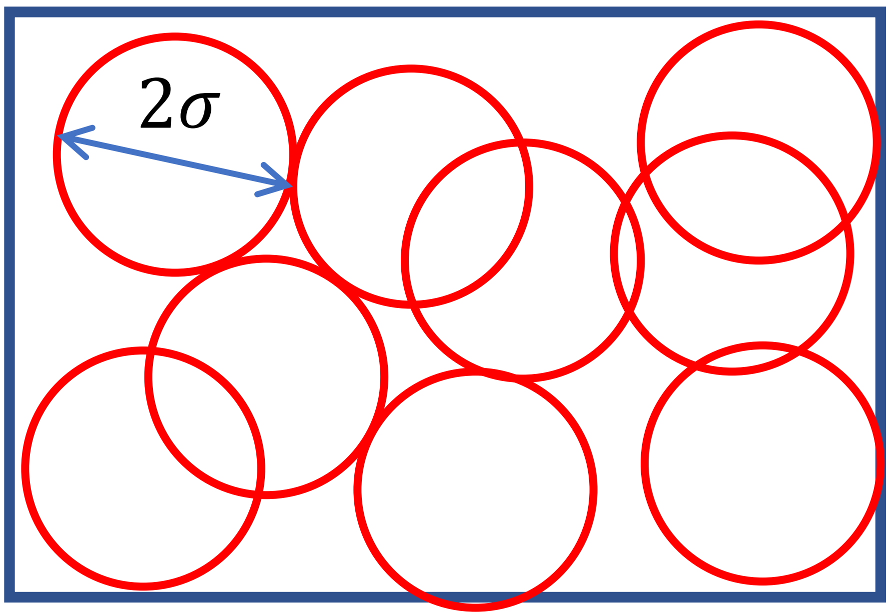
RBF Feature Transform

- It is a bit hard to visualize RBF in high dim. space.
- However, you can imagine a R^d space filled with balls with radius σ , which identifies regions over which $f(\boldsymbol{x};\boldsymbol{w})$ will be **supported**.



$$\mathrm{supp}(f) \coloneqq \{\boldsymbol{x} | f(\boldsymbol{x};\boldsymbol{w}) \neq 0\}$$

Packing Number and CoD



- If $g(\boldsymbol{x})$ has a wide support, $f(\boldsymbol{x}; \boldsymbol{w})$ must be supported almost everywhere, we need to have many centroids.
- The number of balls needed to cover a space is called "**packing number**", which grows exponentially with dim.
- $b = O\left(c^d\right)$, CoD!!

Feature Space

- $\boldsymbol{\phi}(\boldsymbol{x})$ transforms input $\boldsymbol{x}$ from R^d to a **feature space** R^b**.**
- $f(\boldsymbol{x}; \boldsymbol{w})$ is an inner product in such a **feature space**.

- By increasing b, we increase the dimensionality of the feature space, thus we increase the flexibility of f.

- Can we have an infinite dimensional feature space?
  - If so, we can **greatly enhance the flexibility of** f.

Infinite Dim. Feature Space

- Suppose $\boldsymbol{\phi}(\boldsymbol{x})$ maps $\boldsymbol{x}$ to an infinite dimensional fea. space.
- We will have a $\boldsymbol{w}$ which is also infinitely long as dimension of $\boldsymbol{w}$ and $\boldsymbol{\phi}(\boldsymbol{x})$ must match in order to do inner product.

- However, recall the regularized LS has solution:
- $\boldsymbol{w}_{\mathrm{LS-R}} \coloneqq (\boldsymbol{\phi}(\boldsymbol{X})\boldsymbol{\phi}(\boldsymbol{X})^\top + \lambda \boldsymbol{I})^{-1}\boldsymbol{\phi}(\boldsymbol{X})\boldsymbol{y}^\top$

- **How to calculate such a solution given $\boldsymbol{\phi}(\boldsymbol{x})$ has an infinite dimensional space?**

Woodbury Identity

- Remarkably,
  - $$\boldsymbol{w}_{\mathrm{LS-R}} := (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}\boldsymbol{y}^\top$$
    $$= \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}^\top$$
- $\boldsymbol{\Phi}$ is short for $\boldsymbol{\phi}(\boldsymbol{X})$.

- Hint, Woodbury identity:
- $(\boldsymbol{P}^{-1} + \boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top = \boldsymbol{P}\boldsymbol{B}^\top(\boldsymbol{B}\boldsymbol{P}\boldsymbol{B}^\top + \boldsymbol{I})^{-1}$

Woodbury Identity 2

- $\boldsymbol{w}_{\text{LS-R}} := \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{y}^\top$
- Recall $\boldsymbol{\Phi} := [\boldsymbol{\phi}(\boldsymbol{x}_1), \cdots, \boldsymbol{\phi}(\boldsymbol{x}_n)] \in R^{b\times n}$,
- Instead of $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top$(which is intractable), we compute $\boldsymbol{\Phi}^\top\boldsymbol{\Phi} \in R^{n\times n}$.

- Denote $\boldsymbol{K}$ as $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}, K^{(i,j)} = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \langle\boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}\left(\boldsymbol{x}_j\right)\rangle$,
- i.e., $k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ is inner product of two feature transform on $\boldsymbol{x}_i, \boldsymbol{x}_j$.
  - Verify it!

Prediction Function

- $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{w}_{\mathrm{LS-R}}) = \langle \boldsymbol{w}_{\mathrm{LS-R}}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle$

- $$\begin{aligned}\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{w}_{\mathrm{LS-R}}) &= \langle \boldsymbol{\phi}(\boldsymbol{x})^\top, \boldsymbol{\Phi}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y}^\top \rangle \\ &= \left\langle \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\Phi}, (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y}^\top \right\rangle\end{aligned}$$

- Denote $\boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\Phi}$ as $\boldsymbol{k} \in R^n$ where
- $k^{(i)} = k(\boldsymbol{x}, \boldsymbol{x}_i) = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}_i) \rangle$

Evaluating only the Inner Products

- $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{w}_{\mathrm{LS-R}}) := \boldsymbol{k}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-\mathbf{1}} \boldsymbol{y}^{\top}$
- Note $\boldsymbol{\phi}(\boldsymbol{x})$ only appears inside the inner products!
- Design "an inner product function $k(\boldsymbol{x}, \boldsymbol{x}')$" mimics behaviour of inner product between $\boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{\phi}(\boldsymbol{x}')$.
  - We do not have to worry about computing $\boldsymbol{\phi}(\cdot)$ explicitly!

Evaluating only the Inner Products

- Of course, you **cannot** pick inner product function k arbitrarily.
  - Must "behaves like" an inner product.
- However, there are many **known choices** of k corresponds to inner products of powerful, even infinite dimensional feature transform $\boldsymbol{\phi}(\boldsymbol{x})$.

Kernel Function

- Our inner product function $k(.,.)$ is called **kernel function** in machine learning literatures.

- If explicit $\boldsymbol{\phi}(\boldsymbol{x})$ can be derived from k,
  - We say, k induces feature transform $\boldsymbol{\phi}(\boldsymbol{x})$.

Choices of k

- Linear kernel function:
  - $k(\boldsymbol{x}_i, \boldsymbol{x}_j) := <\boldsymbol{x}_i, \boldsymbol{x}_j>$
  - Induced feature transform $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x}$.
- Polynomial kernel function with degree b:
  - $k(\boldsymbol{x}_i, \boldsymbol{x}_j) := (<\boldsymbol{x}_i, \boldsymbol{x}_j> +1)^b$
- Write down induced $\boldsymbol{\phi}(\boldsymbol{x})$ by polynomial kernels $b = 2$.
- Hint, express $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as inner products of $\boldsymbol{\phi}(\boldsymbol{x}_i)$ and $\boldsymbol{\phi}(\boldsymbol{x}_j)$.

Choices of k

- RBF (or Gaussian) kernel:
  - $k(\boldsymbol{x}_i, \boldsymbol{x}_j) := \exp\left(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma^2}\right)$
  - $\boldsymbol{\phi}(\boldsymbol{x})$ induced by k is **infinite dimensional**!
  - σ is chosen before fitting.
  - σ can be chosen as the median of pairwise distances of all your input $\boldsymbol{x}$.
- RBF kernel and RBF basis function is **not** the same thing despite a similar look!

Choices of k

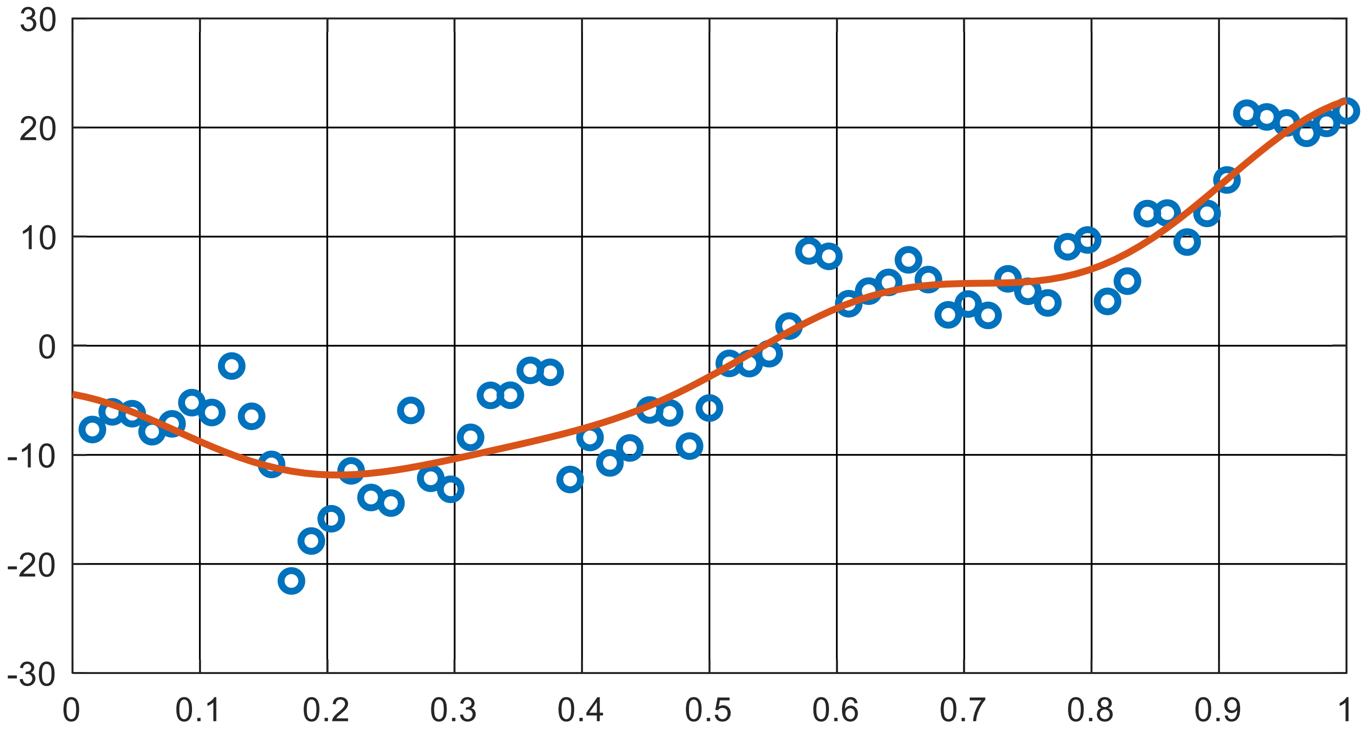
- How do I pick k?
  - Depending on your learning task.
    - e.g., linear/poly kernels are frequently used in natural language processing.
  - Depending on your dataset.
    - e.g., kernels can be defined for structural inputs, such as strings or graphs.
  - Domain knowledge matters!!
- RBF kernel is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

Implementation Concern of Kernel LS

- Recall: $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{w}_{\mathrm{LS-R}}) \coloneqq \boldsymbol{k}(\boldsymbol{K} + \lambda\boldsymbol{I})^{-\mathbf{1}}\boldsymbol{y}$
- Computational cost
  - $\boldsymbol{K}$: $O(n^{\mathbf{2}})$
  - $(\boldsymbol{K} + \lambda\boldsymbol{I})^{-\mathbf{1}}$: Usually $O(n^3)$
  - Kernel methods though flexible, is computationally demanding for large n.

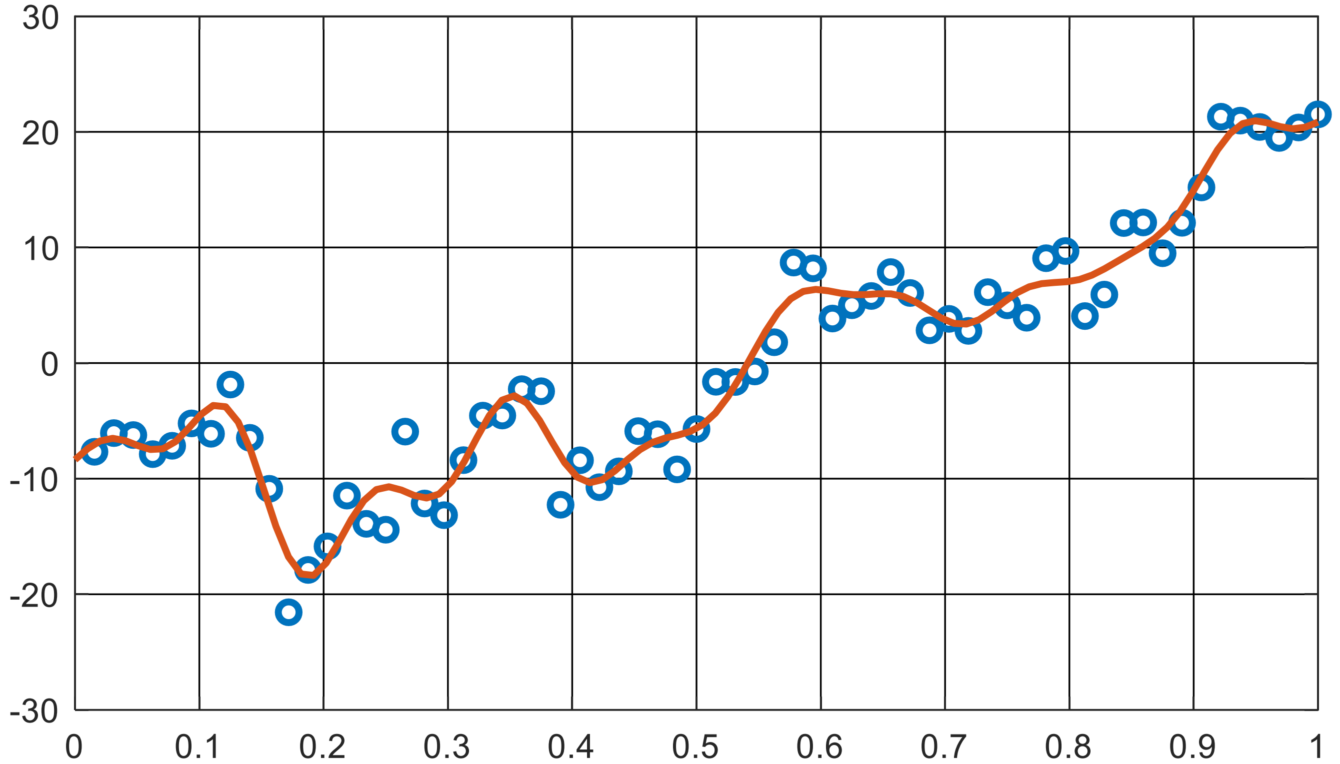
APPL stock price, Jul-Oct, 2019

- RBF kernel, $\lambda = .01, \sigma = 0.2099$.



APPL stock price, Jul-Oct, 2019

- RBF kernel, $\lambda = .01, \sigma = 0.1050.$



Conclusion

- Other than Poly. Transform, we introduce
  - Trigonometric Transform
  - RBF Transform

- Kernel methods transform original data point into higher dimensional (potentially **infinitely dim.**) feature space.
  - We get a super flexible prediction f.

Homework

- Prove $\boldsymbol{w}_{\mathrm{LS-R}} := \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y}^\top$ using Woodbury identity.
- Write down induced $\boldsymbol{\phi}(\boldsymbol{x})$ by poly kernels $b = 2$.