

Feature Transform and Kernel Methods

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

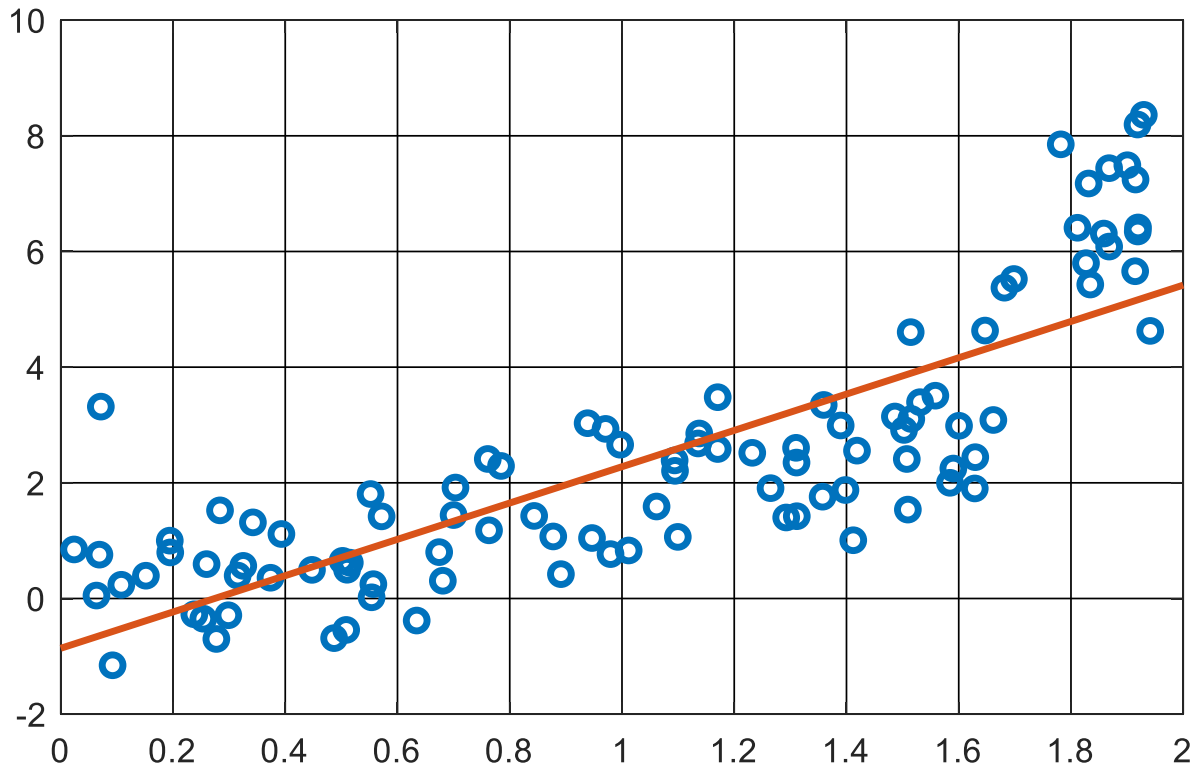
LS with Feature Transform

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

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

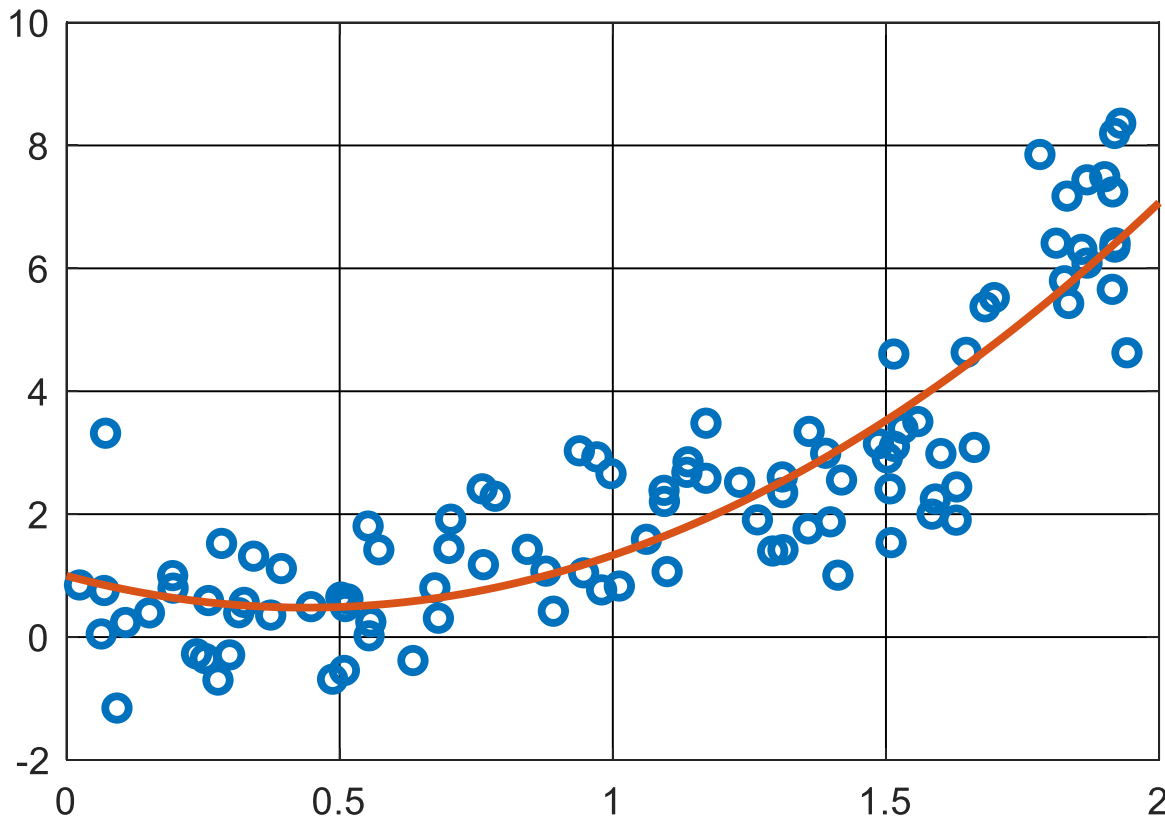
Polynomial Transform $b = 1$

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



Polynomial Transform $b = 2$

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



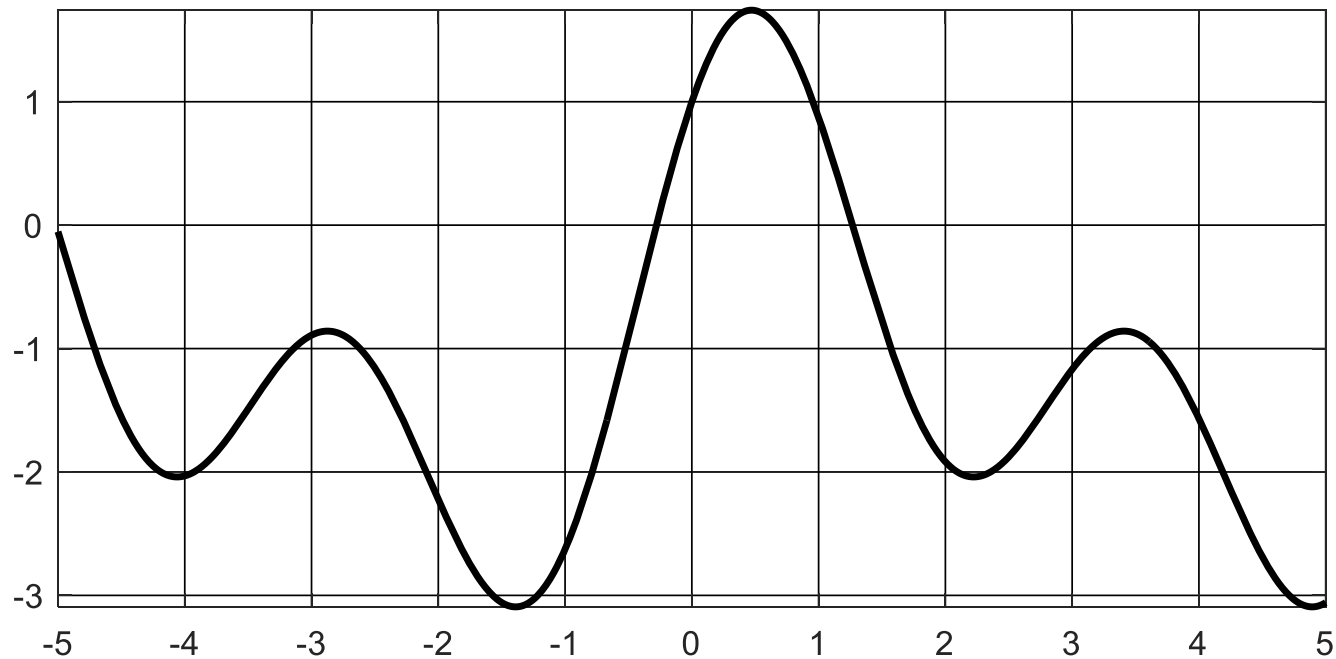
Why it works?

- 1-dimensional intuition: Taylor Series.
- Taylor Series of $g(x)$ at 0:
 - $$g(x) = g(0)(x - 0)^0 + g'(0)(x - 0)^1 + \frac{g''(0)}{2!}(x - 0)^2 + \frac{g'''(0)}{3!}(x - 0)^3 + \dots$$
- 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



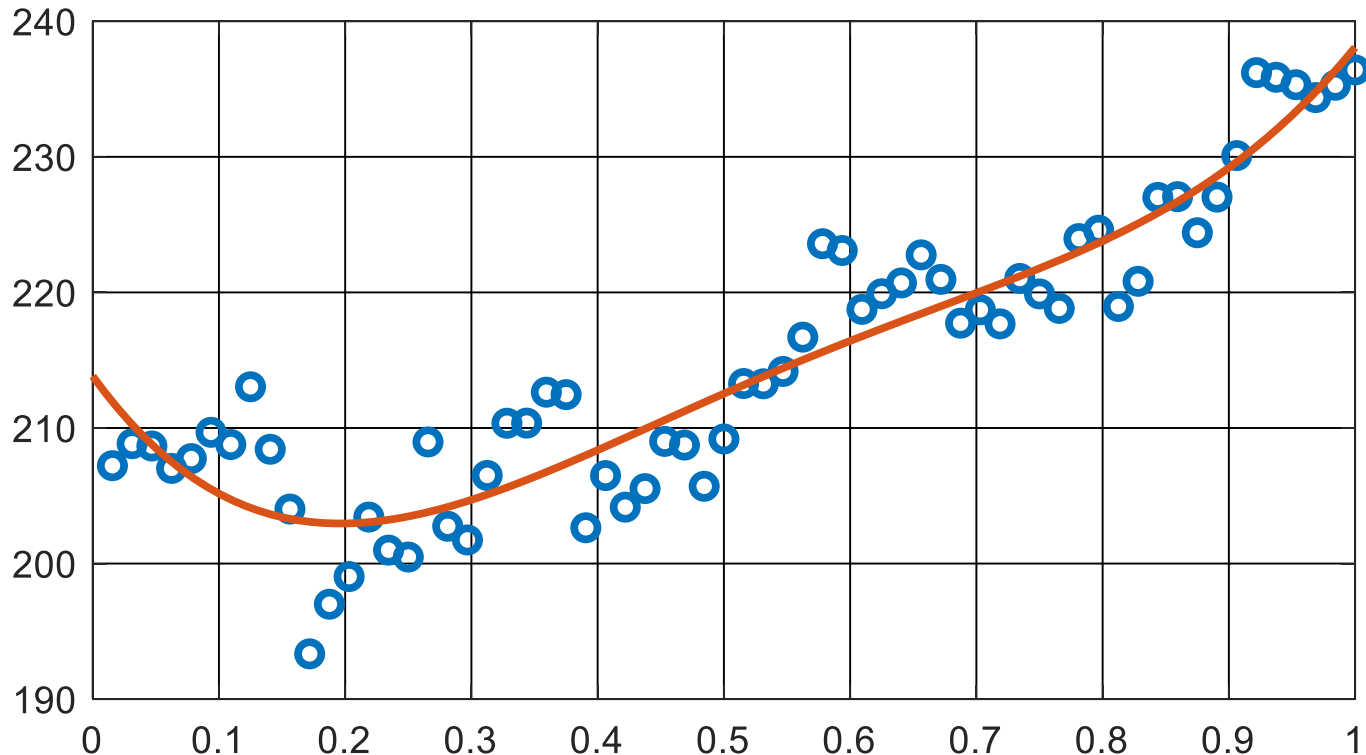
- $g(x) = \sin(x) + \cos(x) + \sin(2x) + \cos(2x)$

Trigonometric Transform

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

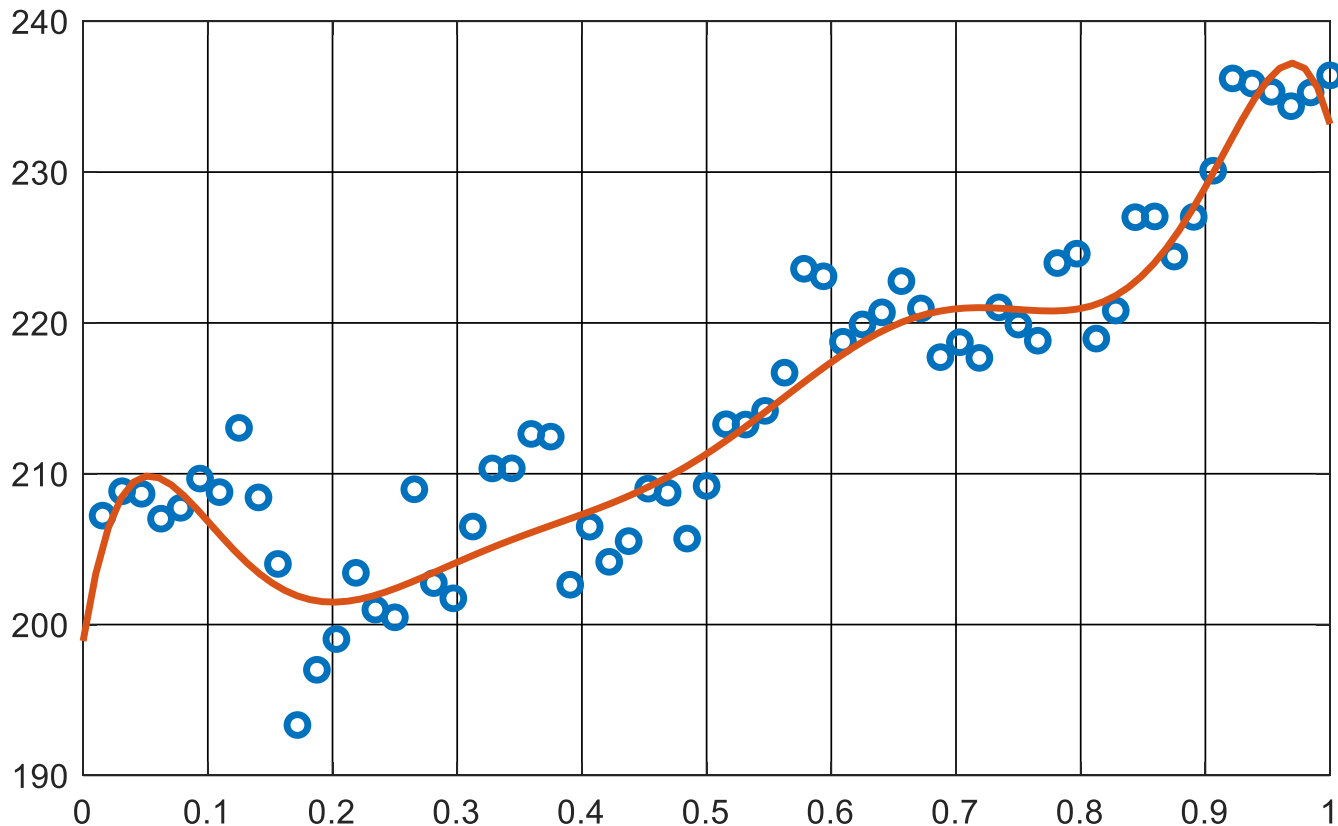
APPL stock price, Jul-Oct, 2019

- 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:

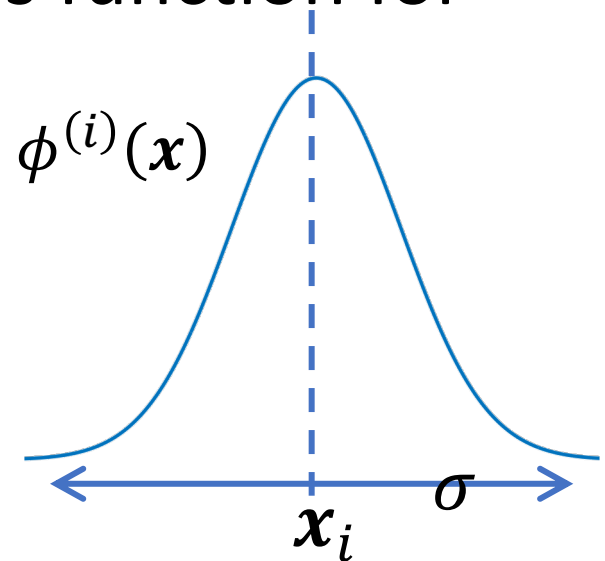
- $g(\mathbf{x}) \approx f(\mathbf{x}; \mathbf{w})$

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

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

Radial Basis Function (RBF)

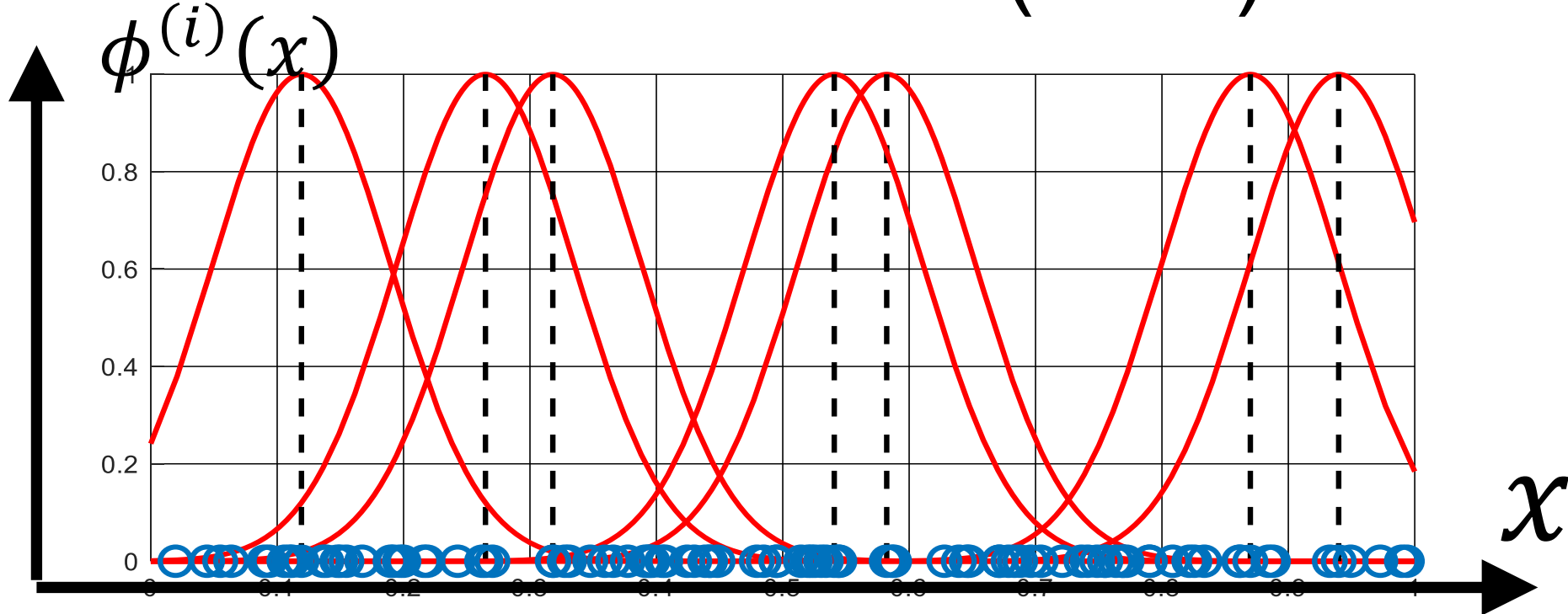
- RBF is another widely used basis function for regression tasks.
- $\phi^{(i)}(\mathbf{x}) := \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right)$
 - $\sigma > 0$ is called bandwidth
 - σ is determined **before** fitting
- A practice is setting σ as the median of all pairwise distances of \mathbf{x} in your dataset.




Radial Basis Function (RBF)

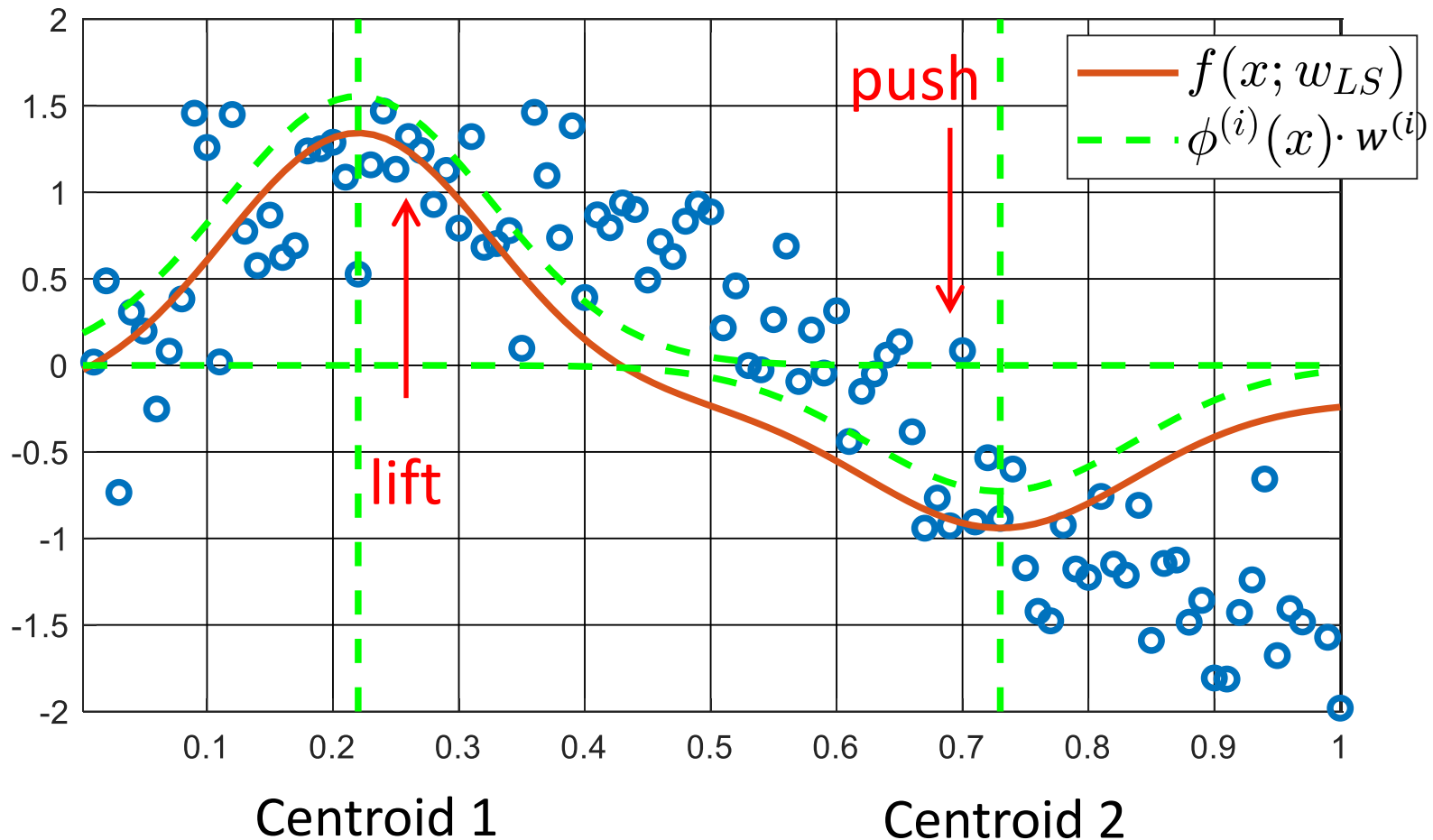
- \mathbf{x}_i are called **RBF centroids**.
- \mathbf{x}_i can be **randomly chosen** from the \mathbf{x} in your dataset
- $\boldsymbol{\phi}(\mathbf{x}) := [\phi^{(1)}(\mathbf{x}), \phi^{(2)}(\mathbf{x}), \dots, \phi^{(b)}(\mathbf{x})]$

Radial Basis Function (RBF)



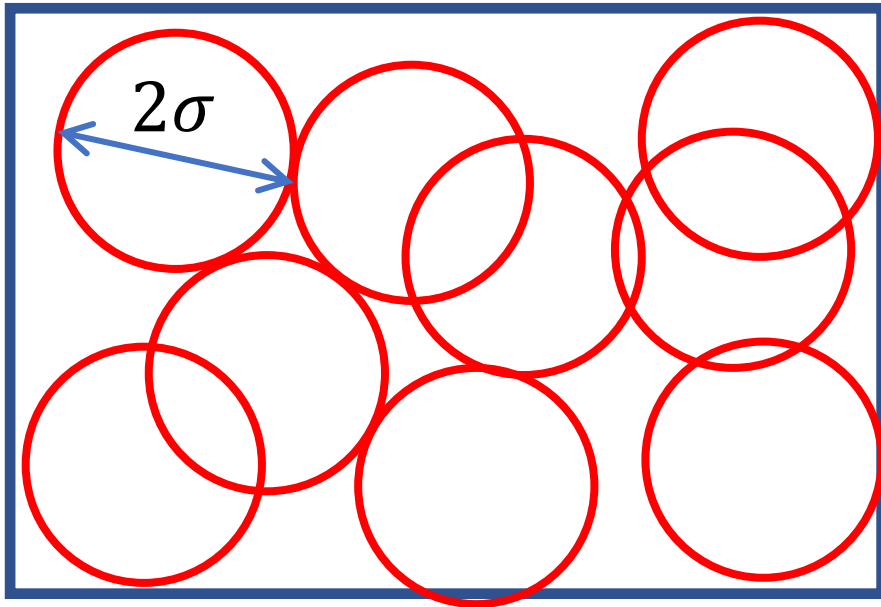
- $\phi^{(i)}(x)$ are visualized in red at random 7 centroids among 100 uniformly drawn x .
- At each “bump ”,
 - If $w^{(i)} > 0$, basis at $x^{(i)}$ gives $f(x; w)$ a “lift”.
 - If $w^{(i)} < 0$, basis at $x^{(i)}$ gives $f(x; w)$ a “push”.

RBF Feature Transform, $b = 2$



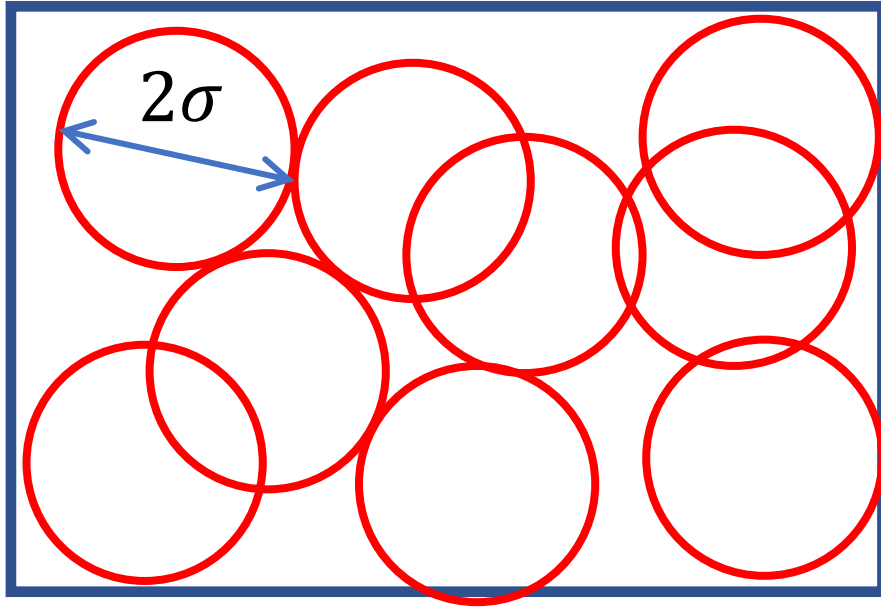
RBF Feature Transform

- 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 $f(\mathbf{x}; \mathbf{w})$ will be **supported**.



$$\text{supp}(f) := \{\mathbf{x} | f(\mathbf{x}; \mathbf{w}) \neq 0\}$$

Packing Number and CoD



- If $g(\mathbf{x})$ has a wide support, $f(\mathbf{x}; \mathbf{w})$ 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.
- $b = O(c^d)$, CoD!!

Feature Space

- $\phi(\mathbf{x})$ transforms input \mathbf{x} from R^d to a **feature space** R^b .
- $f(\mathbf{x}; \mathbf{w})$ 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 $\phi(x)$ maps x to an infinite dimensional fea. space.
- We will have a w which is also infinitely long as dimension of w and $\phi(x)$ must match in order to do inner product.
- However, recall the regularized LS has solution:
- $w_{\text{LS-R}} := (\phi(X)\phi(X)^\top + \lambda I)^{-1} \phi(X)y^\top$
- **How to calculate such a solution given $\phi(x)$ is in an infinite dimensional space?**

Woodbury Identity

- Remarkably,
 - $\mathbf{W}_{\text{LS-R}} := (\Phi\Phi^\top + \lambda I)^{-1} \Phi \mathbf{y}^\top$
 $= \Phi(\Phi^\top \Phi + \lambda I)^{-1} \mathbf{y}^\top$
- Φ is short for $\phi(X)$.
- Hint, Woodbury identity:
- $(P^{-1} + B^\top B)^{-1} B^\top = P B^\top (B P B^\top + I)^{-1}$

Woodbury Identity 2

- $\mathbf{W}_{\text{LS-R}} := \Phi \left(\Phi^\top \Phi + \lambda I \right)^{-1} \mathbf{y}^\top$
- Recall $\Phi := [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_n)] \in R^{b \times n}$,
- Instead of $\Phi \Phi^\top$ (which is intractable), we compute $\Phi^\top \Phi \in R^{n \times n}$.
- Denote \mathbf{K} as $\Phi^\top \Phi$, $K^{(i,j)} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$,
- i.e., $k(\mathbf{x}_i, \mathbf{x}_j)$ is inner product of two feature transform on $\mathbf{x}_i, \mathbf{x}_j$.
 - Verify it!

Prediction Function

- $f(\mathbf{x}; \mathbf{w}_{\text{LS-R}}) = \langle \mathbf{w}_{\text{LS-R}}, \boldsymbol{\phi}(\mathbf{x}) \rangle$
- $$\begin{aligned} f(\mathbf{x}; \mathbf{w}_{\text{LS-R}}) &= \langle \boldsymbol{\phi}(\mathbf{x})^\top, \boldsymbol{\Phi}(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}^\top \rangle \\ &= \langle \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Phi}, (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}^\top \rangle \end{aligned}$$
- Denote $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Phi}$ as $\mathbf{k} \in \mathbb{R}^n$ where
- $k^{(i)} = k(\mathbf{x}, \mathbf{x}_i) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}_i) \rangle$

Evaluating only the Inner Products

- $f(\mathbf{x}; \mathbf{w}_{\text{LS-R}}) := \mathbf{k}(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}^\top$

 Note $\phi(\mathbf{x})$ only appears inside the inner products!

- Design “an inner product function $k(\mathbf{x}, \mathbf{x}')$ ” mimics behaviour of inner product between $\phi(\mathbf{x})$ and $\phi(\mathbf{x}')$.
 - We do not have to worry about computing $\phi(\cdot)$ 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 $\phi(x)$.

Kernel Function

- Our inner product function $k(.,.)$ is called **kernel function** in machine learning literatures.
- If explicit $\phi(x)$ can be derived from k ,
 - We say, k induces feature transform $\phi(x)$.

Choices of k

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

Choices of k

- RBF (or Gaussian) kernel:
 - $k(\mathbf{x}_i, \mathbf{x}_j) := \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$
 - $\phi(\mathbf{x})$ 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 \mathbf{x} .
- RBF kernel and RBF basis function is **not** the same thing despite a similar look!

Choices of k

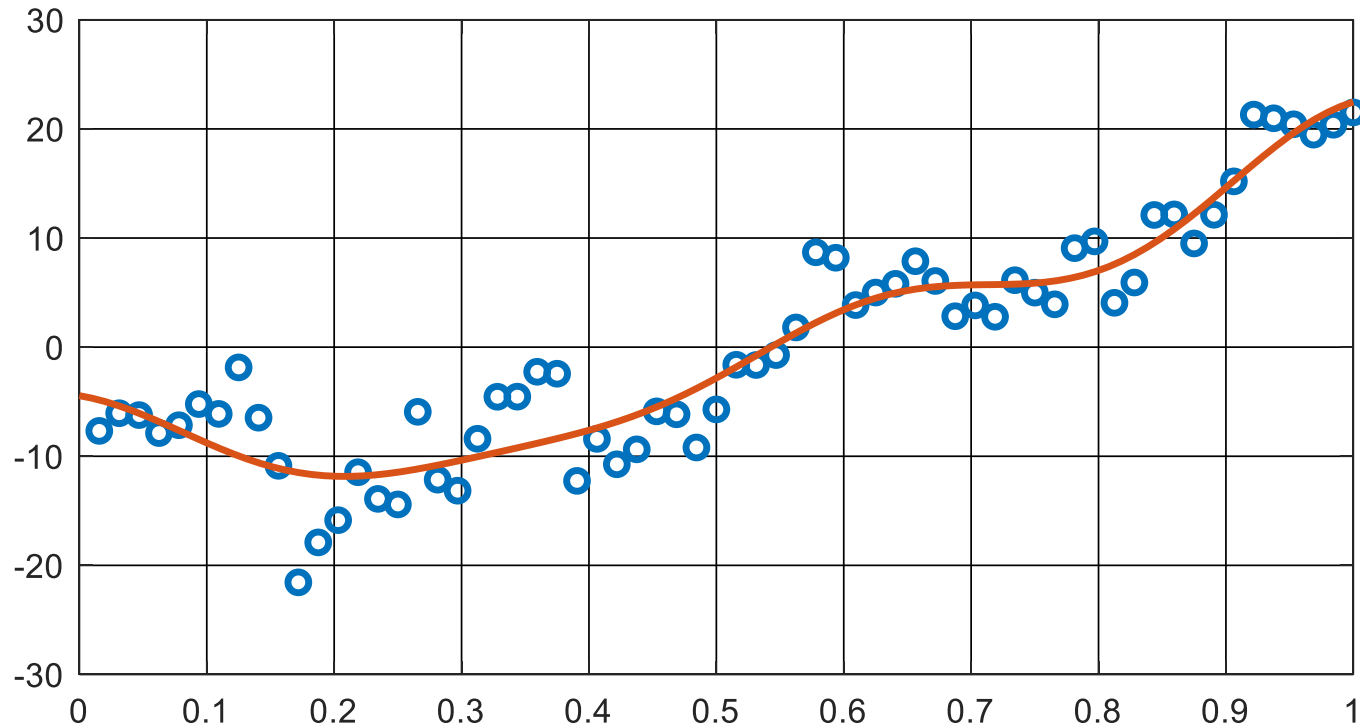
- How do I pick k ?
 - Depending on your learning task.
 - e.g., linear/poly kernels are frequently used in natural language processing.
 - Depending on your dataset.
 - e.g., kernels can be defined for structural inputs, such as strings or graphs.
 - Domain knowledge matters!!
- RBF kernel is a good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

Implementation Concern of Kernel LS

- Recall: $f(\mathbf{x}; \mathbf{w}_{\text{LS-R}}) := \mathbf{k}(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}$
- 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 .

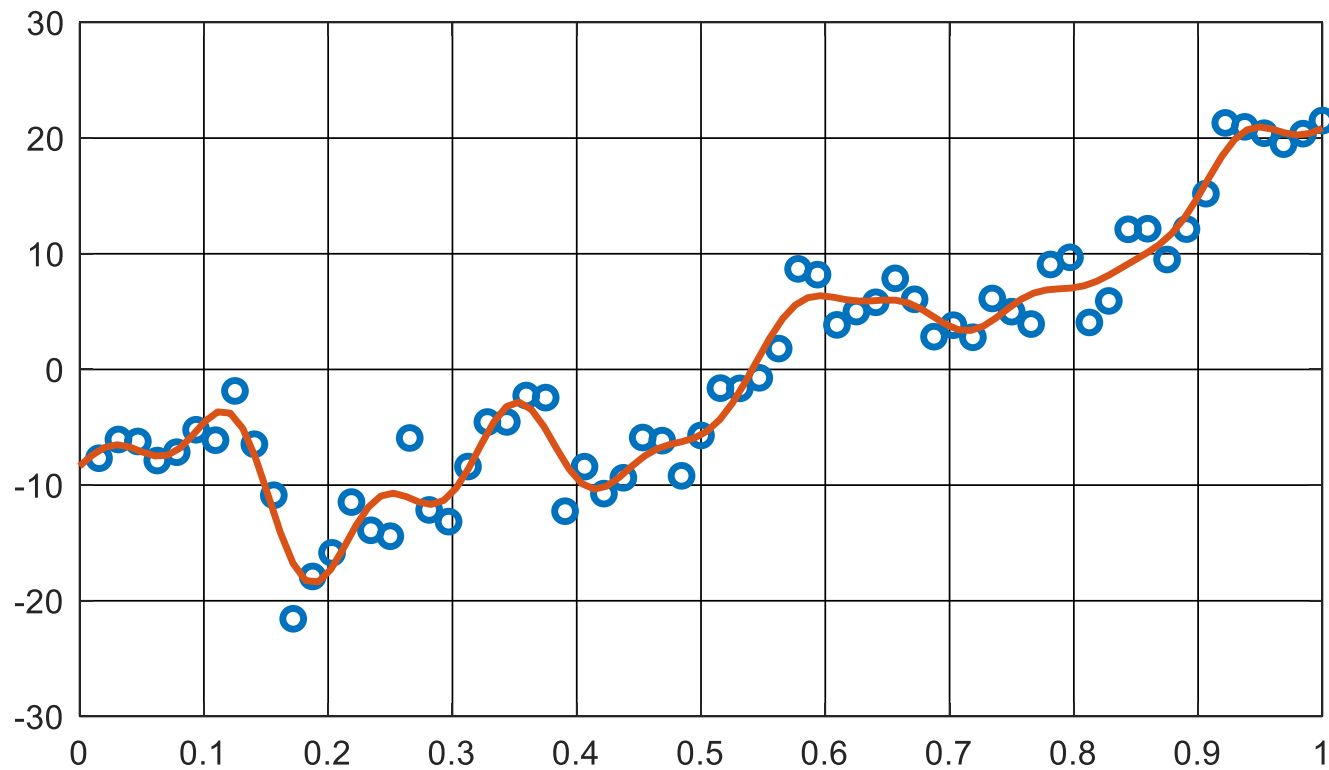
APPL stock price, Jul-Oct, 2019

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