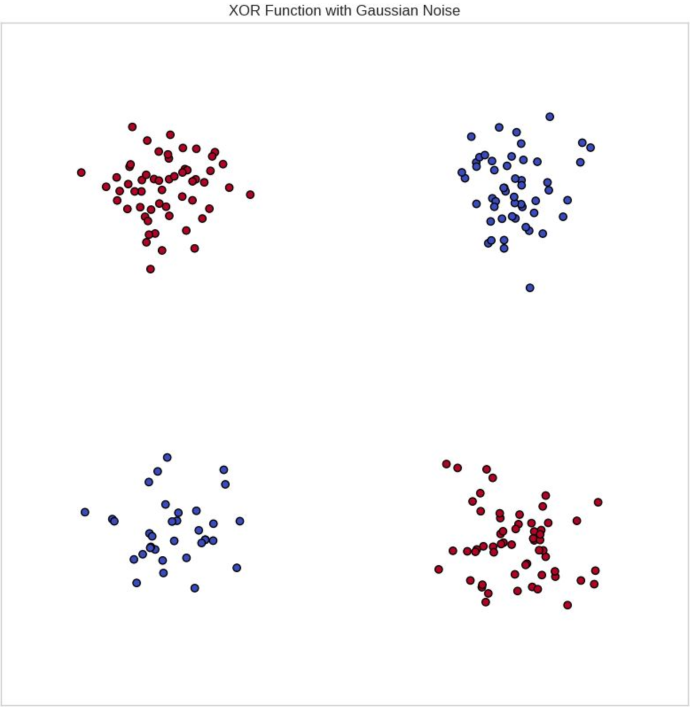
Approximating Kernels with Random Fourier Features

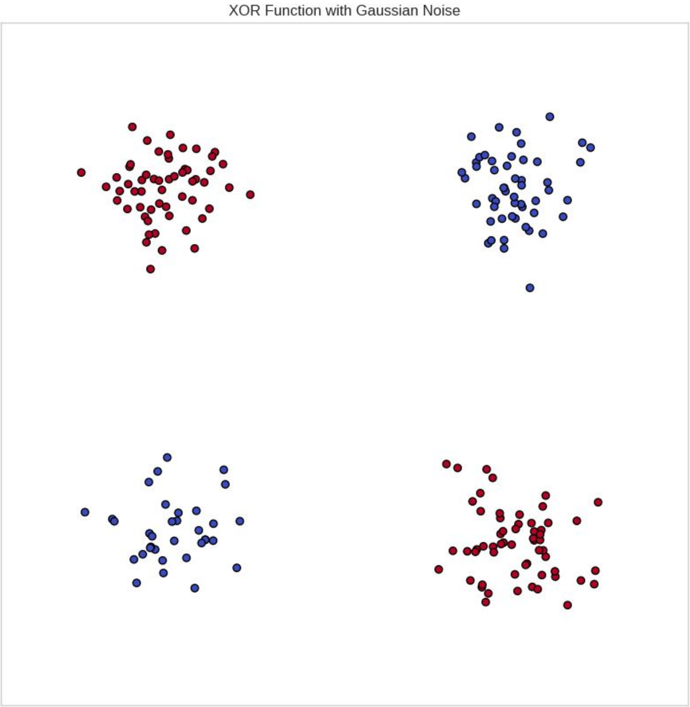
Mathis Rost
Tanja Zast

Motivation

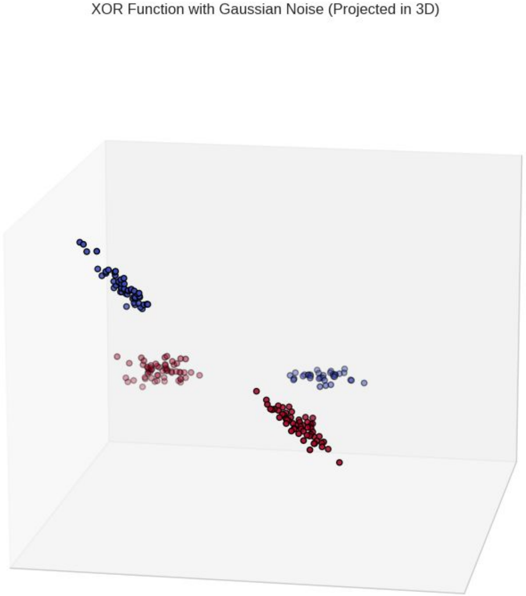


- Dataset is not linear seperable
- Project Data into a Featurespace where Data is linear sperable
- XOR function with noise

Motivation

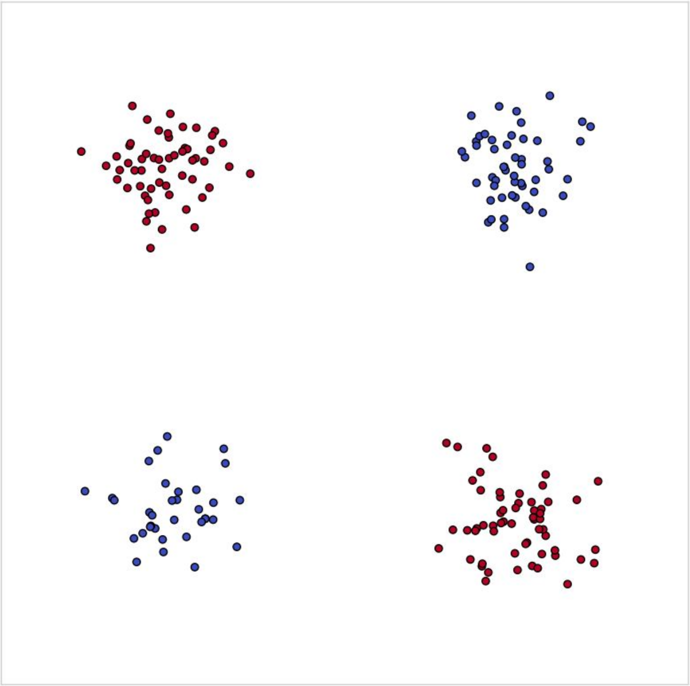


$$\phi(x_1, x_2) = (x_1, x_2, x_1 x_2) \in \mathbb{R}^3$$



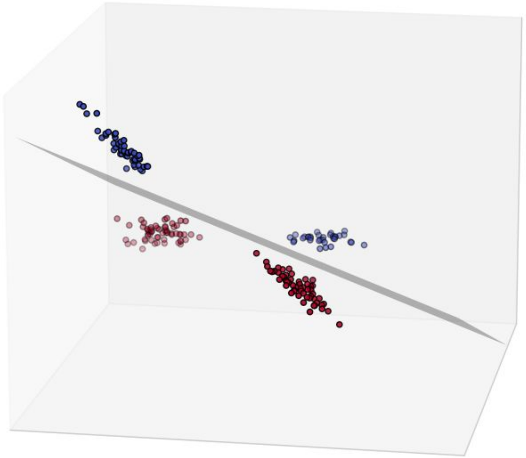
Motivation

XOR Function with Gaussian Noise

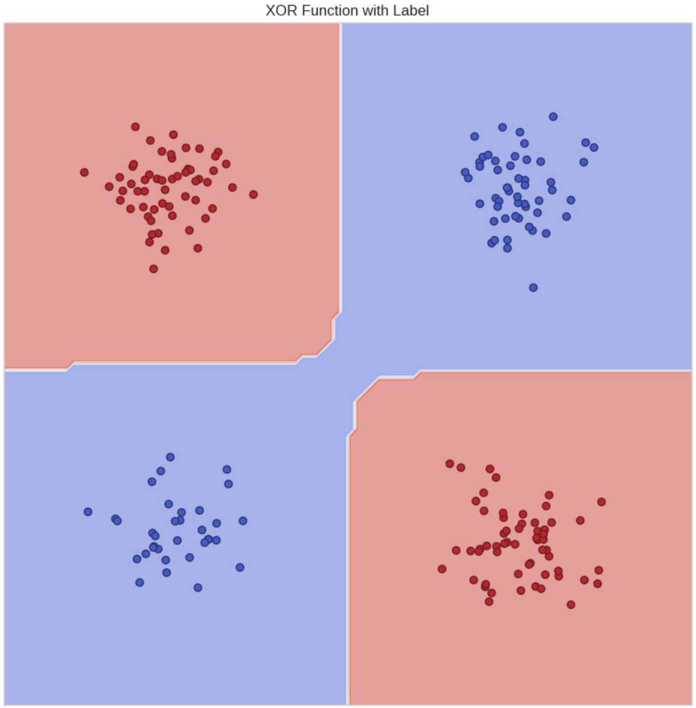


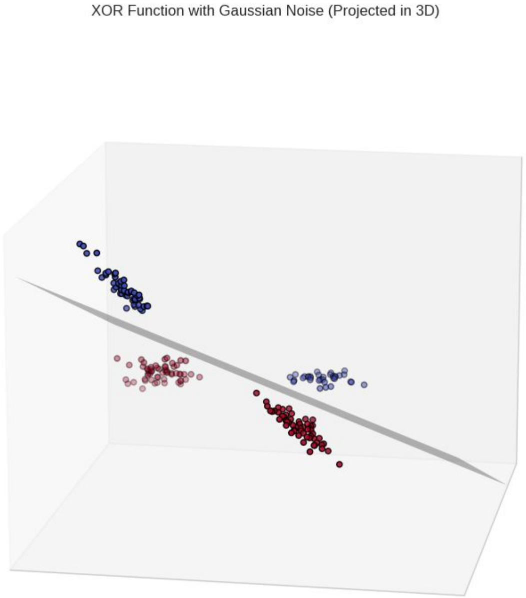
$$\phi(x_1, x_2) = (x_1, x_2, x_1 x_2) \in \mathbb{R}^3$$

XOR Function with Gaussian Noise (Projected in 3D)



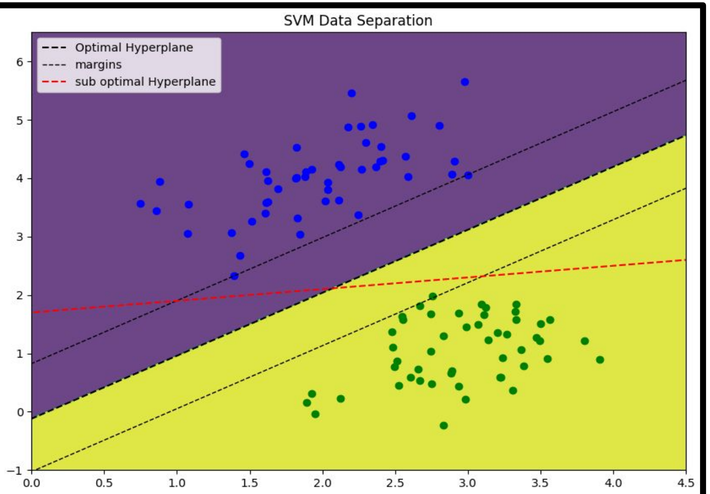
Motivation


XOR Function with Label


XOR Function with Gaussian Noise (Projected in 3D)

Kernel SVMs and the Kernel Trick

- Hyperplane can be represented with a vector $w \in \mathcal{H}$
- $\text{sign}(w^T x_i + b) = y_i$
- Support Vector Machines (SVM) calculate the best possible Hyperplane.
- SVMs can be kernelized with $\phi : X \to \mathcal{H}$
- Resulting Hyperplane: $w^T \phi(z) + b = \langle w, \phi(z) \rangle_{\mathcal{H}} + b$



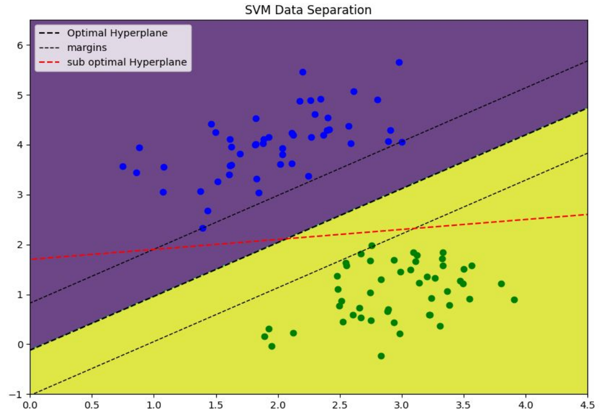
Kernel SVMs and the Kernel Trick

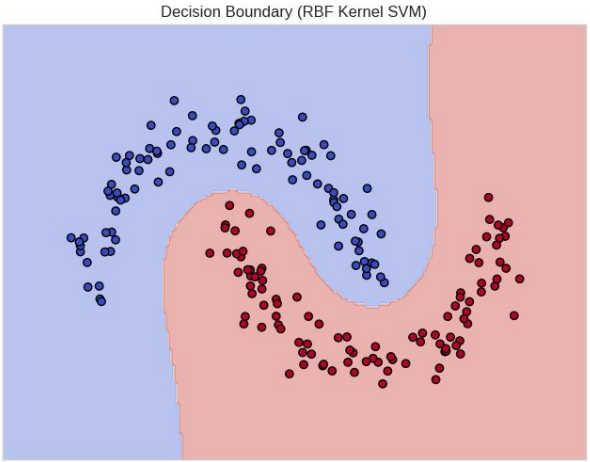
- Formula for Hyperplane in data-space

$$w^T z + b = \sum_{(x,y)} y \alpha_x x^T z + b$$

- Formula for hyperplane in feature-space

$$w^T \phi(z) + b = \langle w, \phi(z) \rangle_{\mathcal{H}} + b = \sum_{x,y} \alpha_x y \phi(x) \phi(z) = \sum_{x,y} \alpha_x y k(z, x) + b$$





Definition of a kernel

Definition 2. *Let X be our data set. A function $k : X \times X \to \mathbb{R}$ is called kernel if there exists an Hilbertspace $\mathcal{H}$ and a map $\phi : X \to \mathcal{H}$ such that $\forall x, y \in X$,*

$$k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} \tag{3.3}$$

Kernel Machines and the Kernel Trick

Mercer's Theorem

Mercer's Theorem: A symmetric function k can be expresses as an inner Product, i.e.

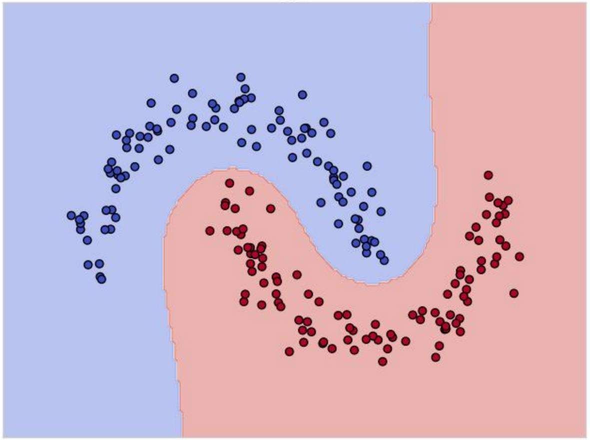
$$k(x, y) = \langle \phi(x), \phi(y) \rangle \tag{3.6}$$

for some $\phi : X \to \mathcal{H}$ if and only if k is positive semi definite

Kernel Matrix

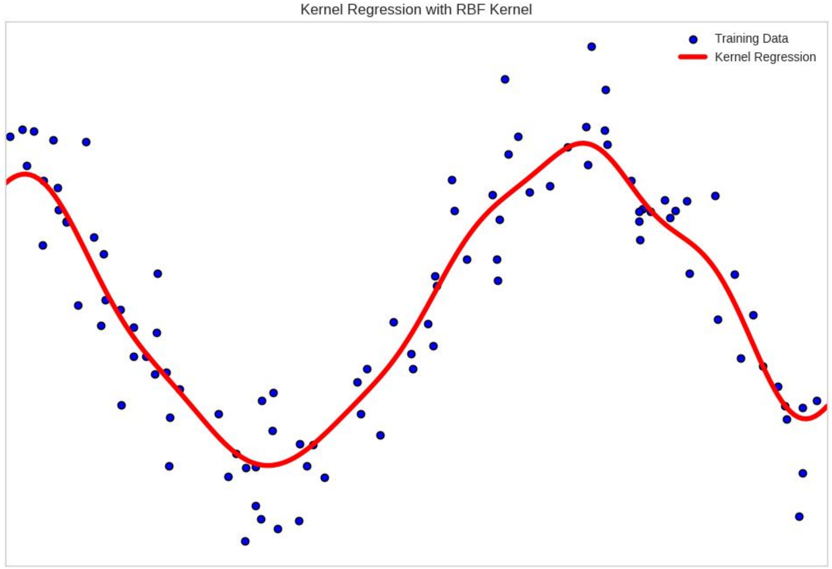
$$K = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix}$$

- We don't need to calculate the mapping $\phi : X \to \mathcal{H}$
- We can calculate decision boundaries with Help of the kernel Matrix
- this saves a lot of time for large dimensional features spaces
- Feature Space could also be infinity



Kernel Machines and the Kernel Trick

- Regression can also be kernelized
- Linear regression is implicit calculated in feature space
- This also works with the kernel trick



- optimal regression line in data-space
- $\hat{w} = (X^T X)^{-1} X^T y$
- optimal regression in feature-space
- $\hat{w} = (\Phi^T \Phi)^{-1} \Phi^T y$ with $\Phi = \phi(X)$
- can be calculated with kernel matrix
- $y = w^T \phi(z) = y(\Phi^T \Phi)^{-1} \Phi^T \phi(z) = (\Phi^T \Phi)^{-1} \sum_{x \in X} k(x, z)$

Kernel Machines and the Kernel Trick

- **Problem with kernel Machines:**
  - kernel Methods do not scale very well with large amount of data
  - For regression the inverse of the kernel matrix has to be computed
  - Kernel matrix will be very large. This can be a Problem for the RAM.
- **Solution**
  - Approximate kernel function
  - We need a map $z : X \to \mathbb{K}^M$, also called random feature

$$\boxed{k(x,y) = \langle \phi(x), \phi(y) \rangle \approx z(x)^T z(y) = \langle z(x), z(y) \rangle}$$

Random Fourer Features (RFFs)

- Translations-invariant kernel: $\forall x, y, z \in X \; k(x, y) = k(x - z, y - z).$
- RBF-kernel is translations-invariant

$$\boxed{k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)}$$

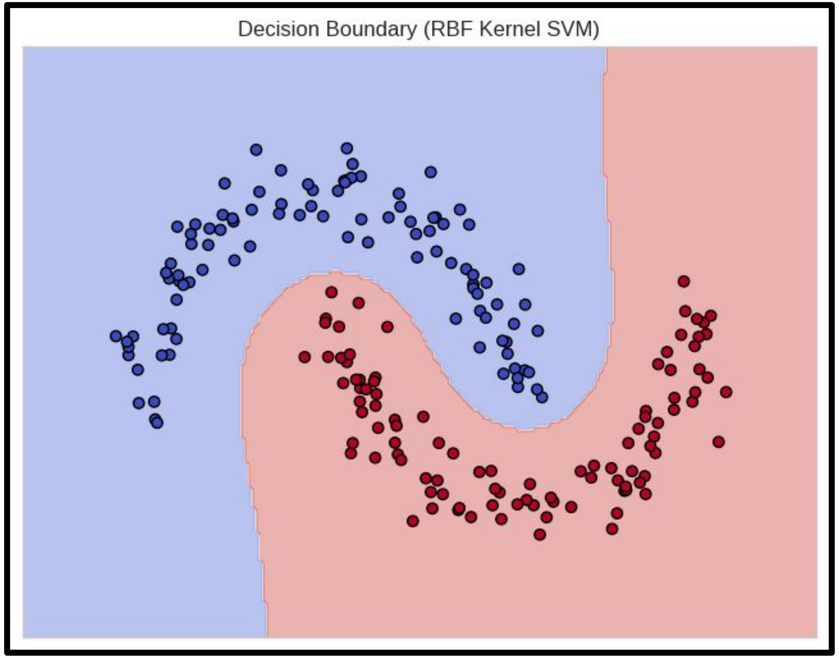
- We can approximate RBF-kernel with $z : x \to \exp\left(iw^T x\right)$, and $w \sim \mathcal{N}(0, I_d)$
- Using multiple Features for better approximation

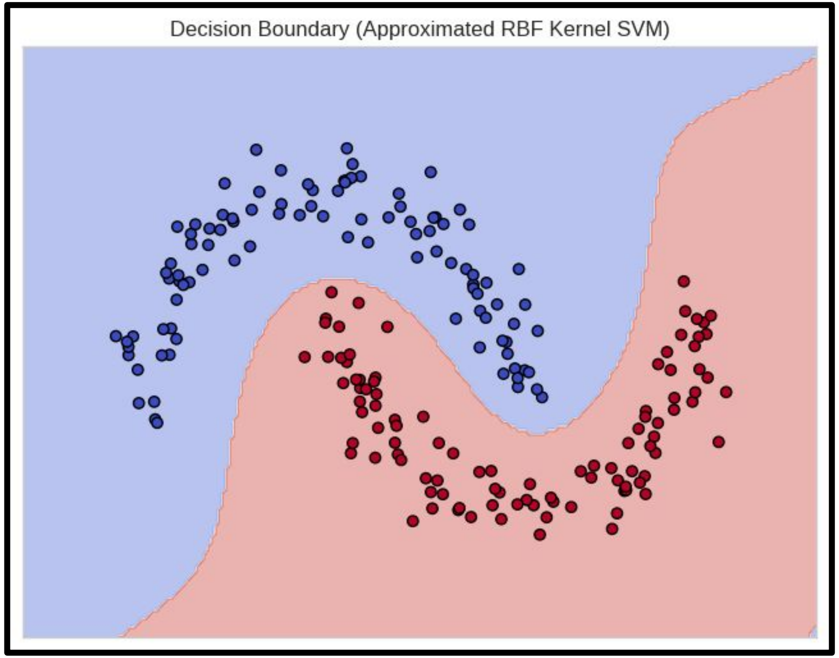
$$\boxed{\langle z(x), z(y) \rangle = \frac{1}{R} \sum_r z_{w_r}(x), z_{w_r}(y)^*}$$

Definition 3. *We call the map $z : \mathbb{R}^d \to \mathbb{K}^R$ a random fourier feature if it approximates a translations-invariant positive definite kernel k, i.e.*

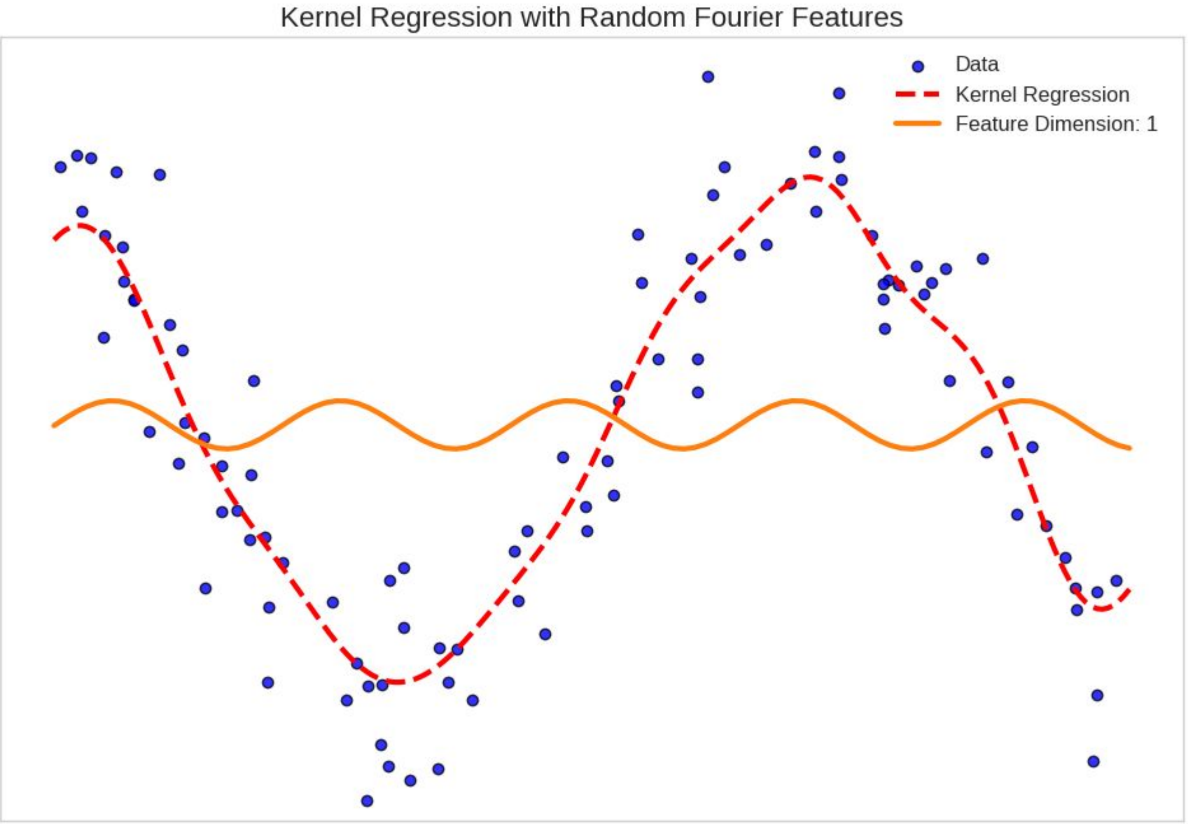
$$\mathbb{E}\left[\langle z(x), z(y) \rangle\right] = k(x - y) = k(x, y) \tag{4.9}$$

Random Fourer Features (RFFs)

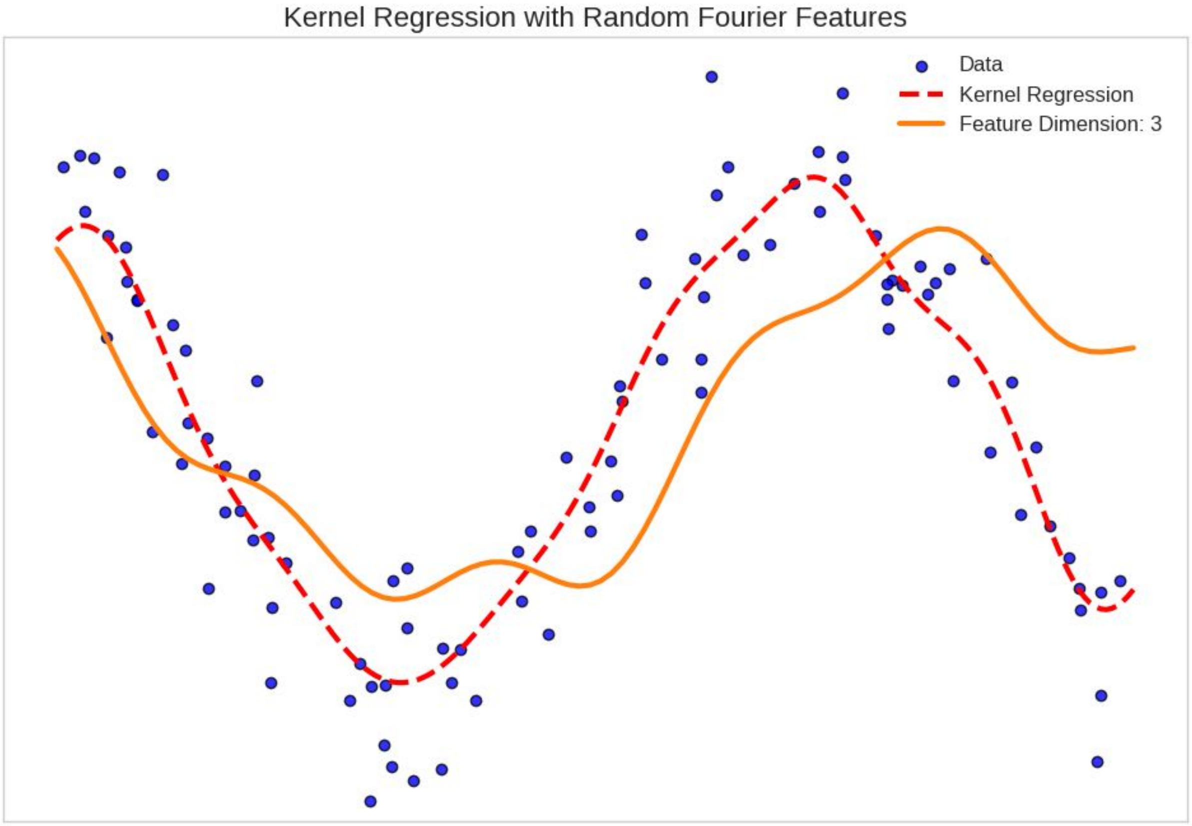




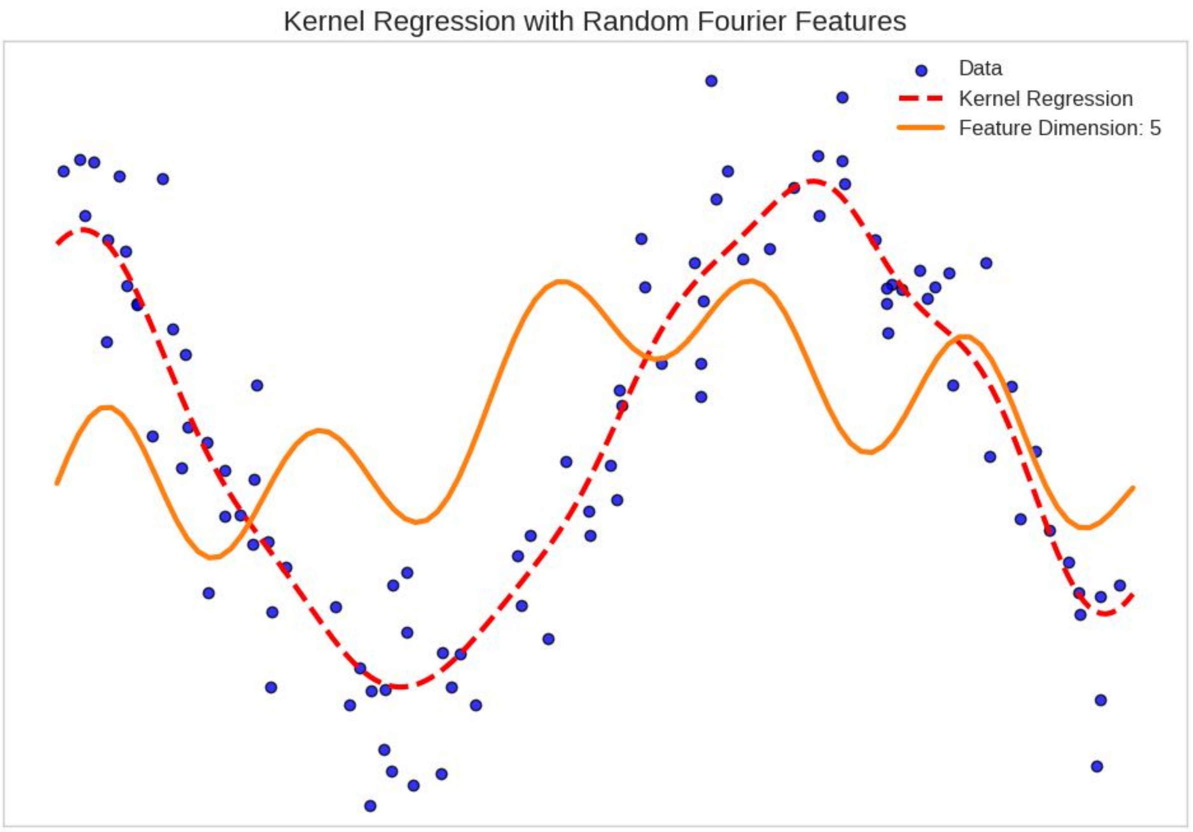
Approximating Kernel with RFFs



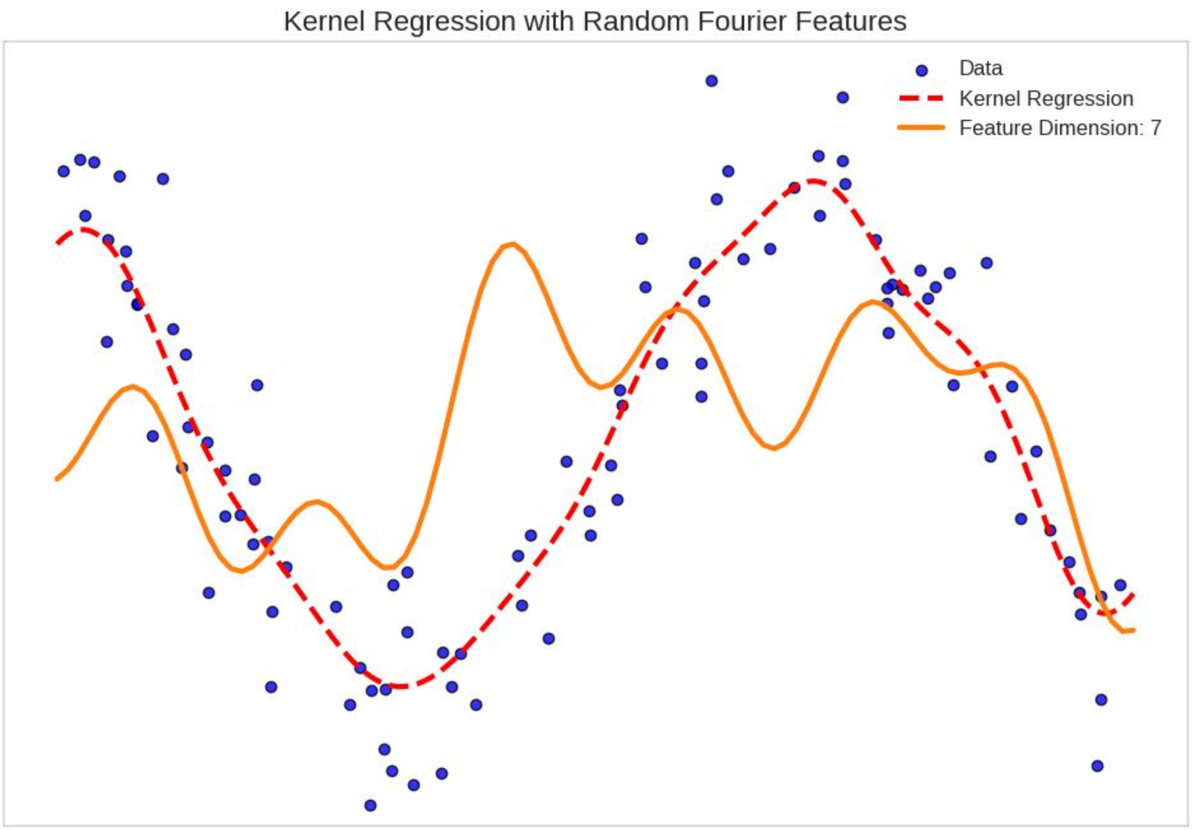
Approximating Kernel with RFFs



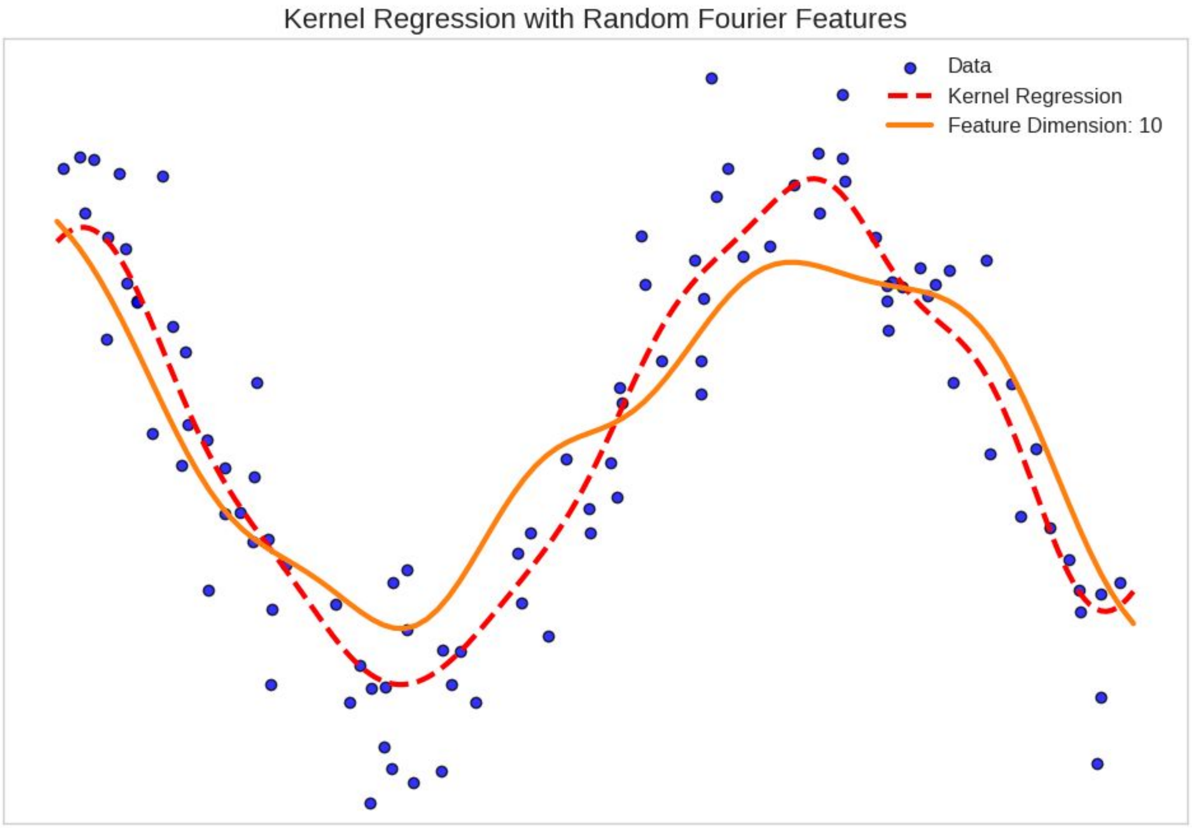
Approximating Kernel with RFFs



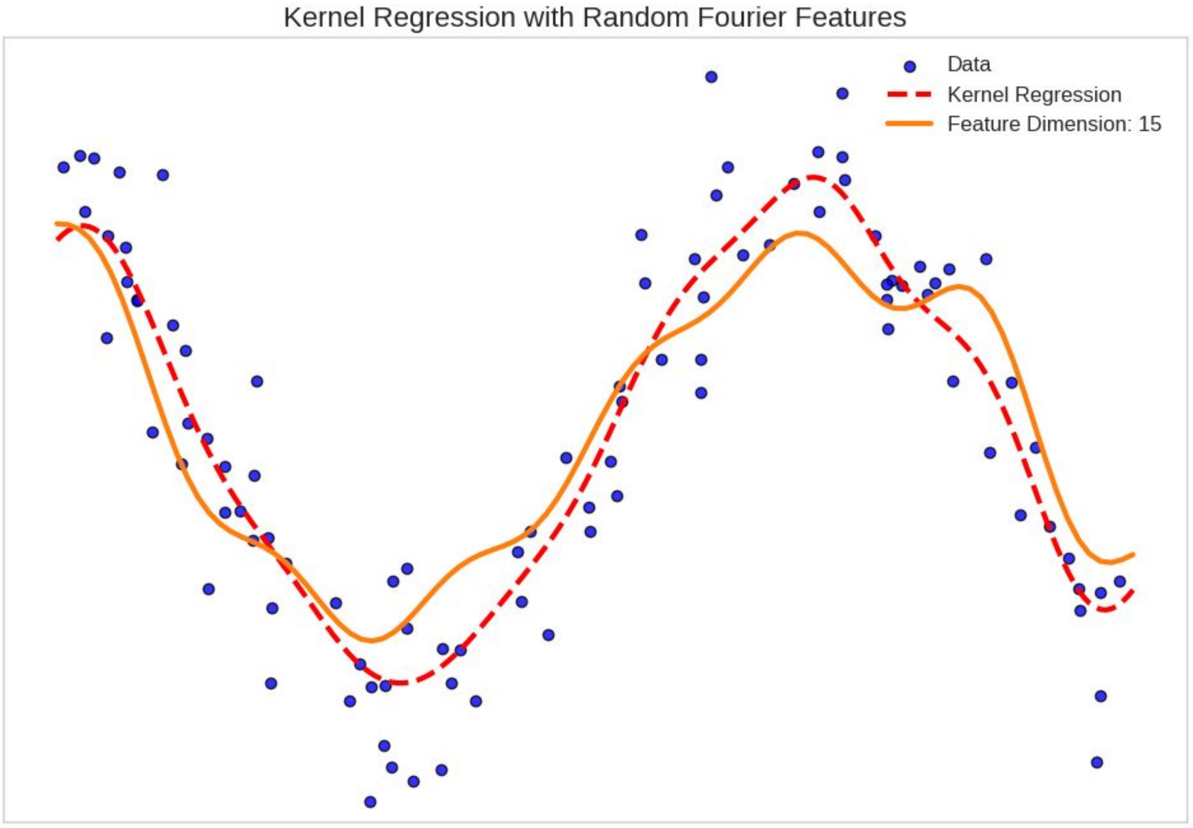
Approximating Kernel with RFFs



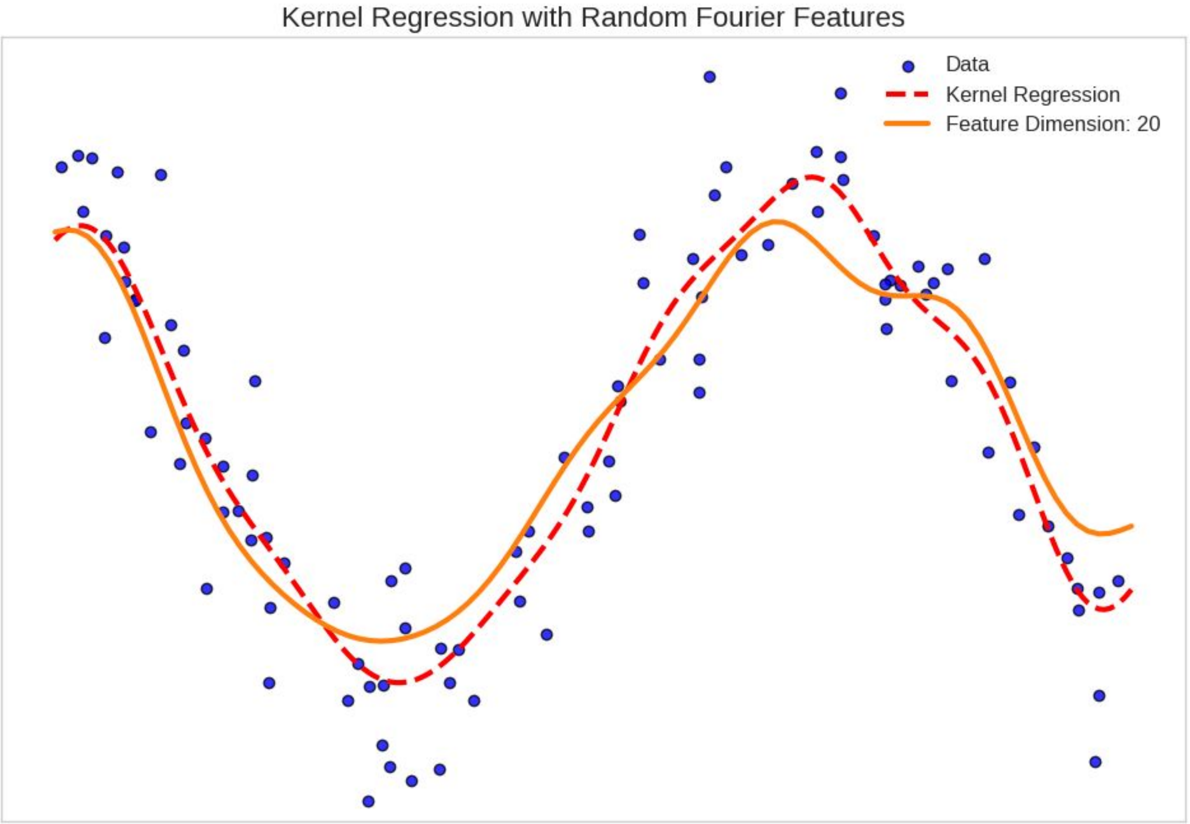
Approximating Kernel with RFFs



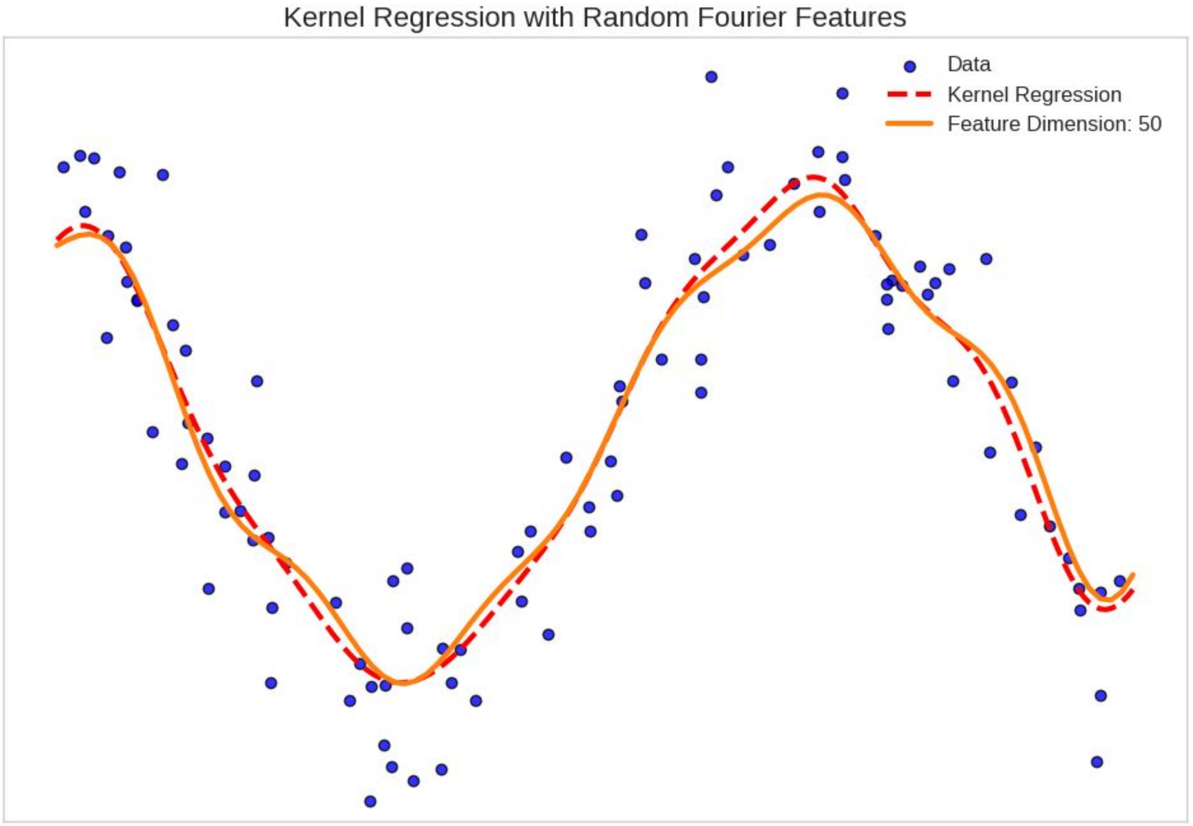
Approximating Kernel with RFFs



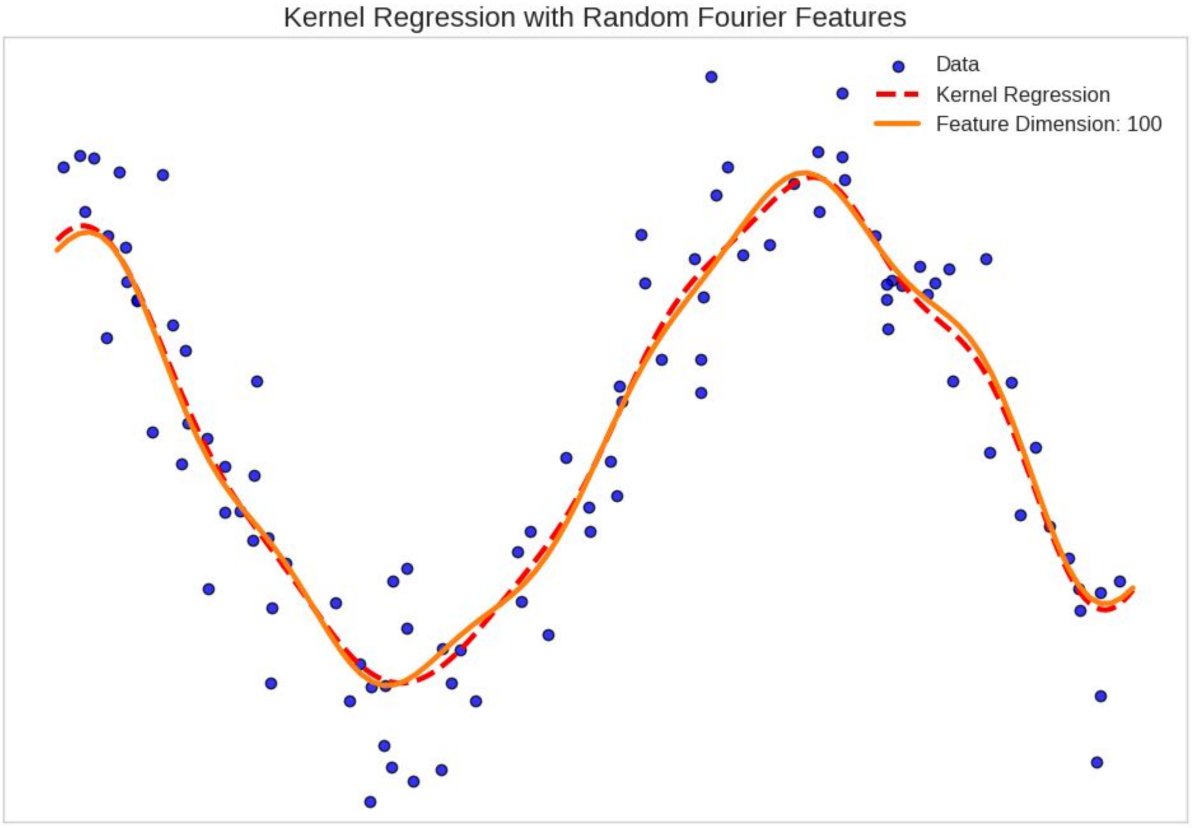
Approximating Kernel with RFFs



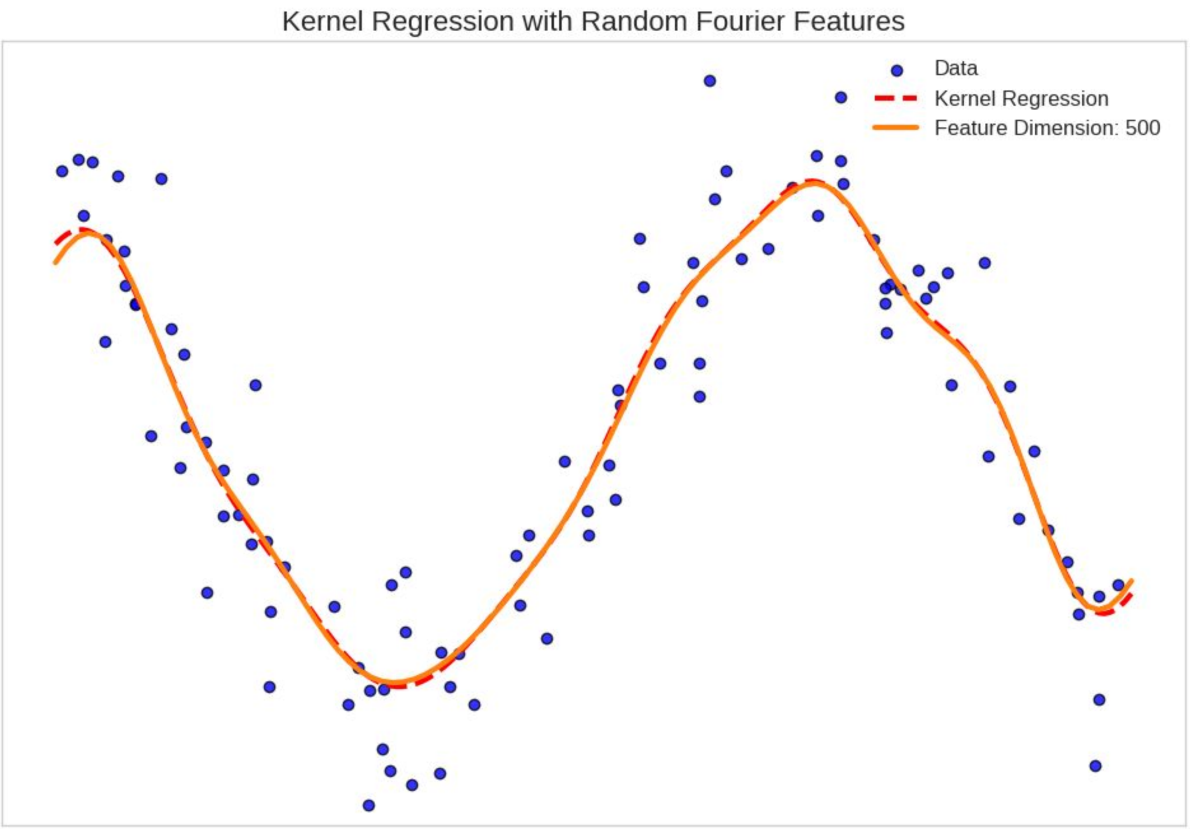
Approximating Kernel with RFFs



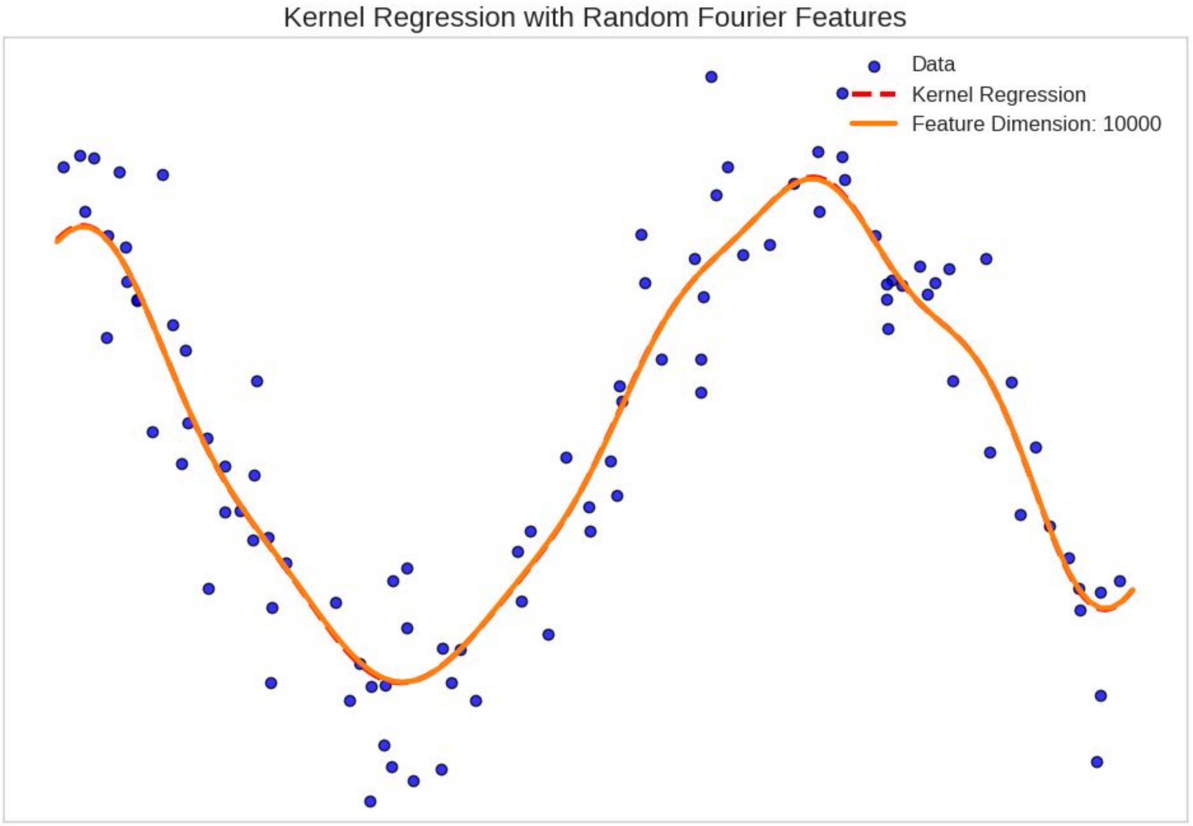
Approximating Kernel with RFFs



Approximating Kernel with RFFs

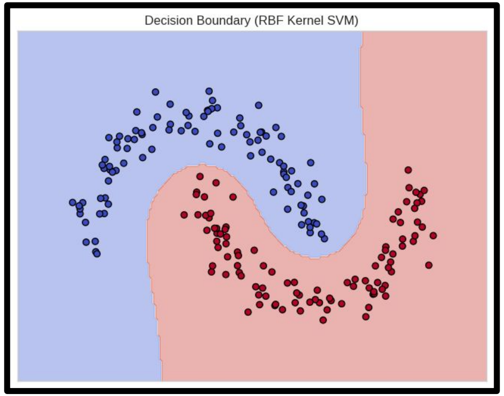


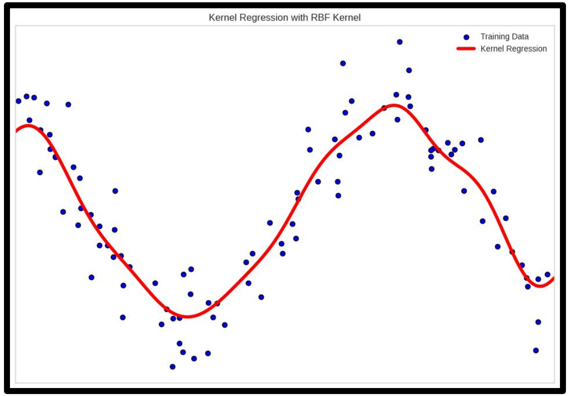
Approximating Kernel with RFFs



Experiments

- Regression with Random Fourier Features approximation
- Classification with Random Fourier Features approximation
- Ensemble Learning with Random Fourier Features





Model Architecture and Hyperparameter

- 2 datasets for regression and 2 datasets for classification
- Using Maxpool Operation for RAM saving and better results
- Using regularization term $\lambda = 1$
- Variance for RBF-kernel $\sigma = 2$

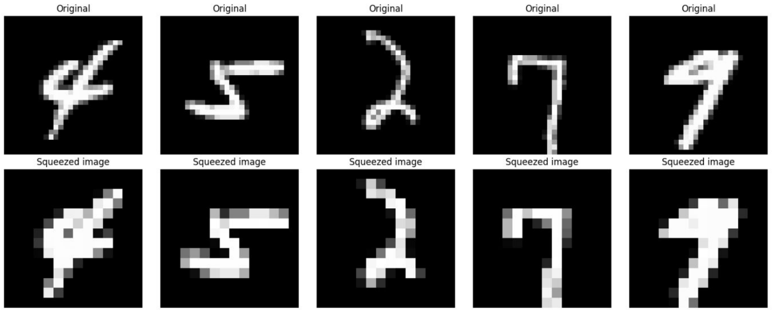
Dataset	#Entries	#Features
Avocado	18 200	8
Wine	1 143	11

Table 5.1.: Dataset Avocado and Wine Quality for Regression

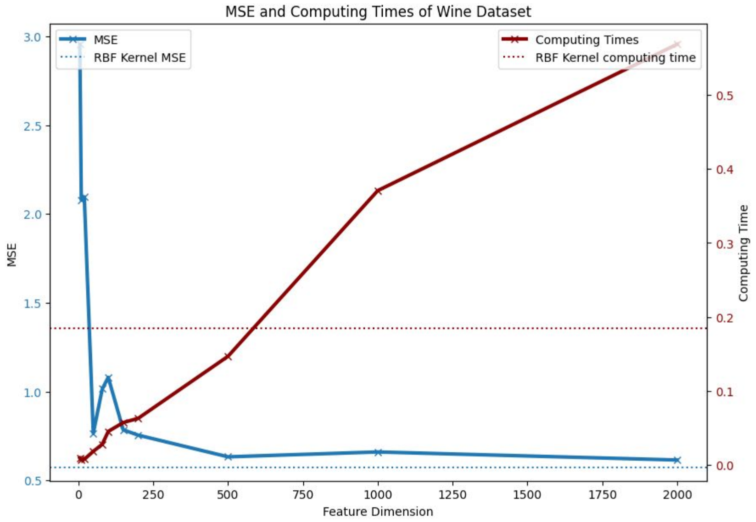
Dataset	#Entries	#Classes
MNIST	70 000	10
Fashion-MNIST	70 000	10

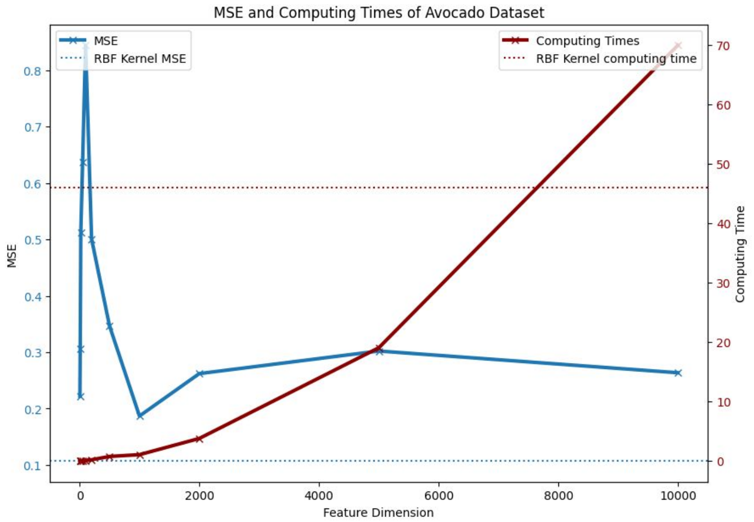
Table 5.2.: Dataset MNIST and Fashion-MNIST for Classification

$$\mathcal{L}(X) = \sum_{x,y \in X \times Y} (w^T \phi(x) - y)^2 + \frac{\lambda}{2} \|w\|^2$$

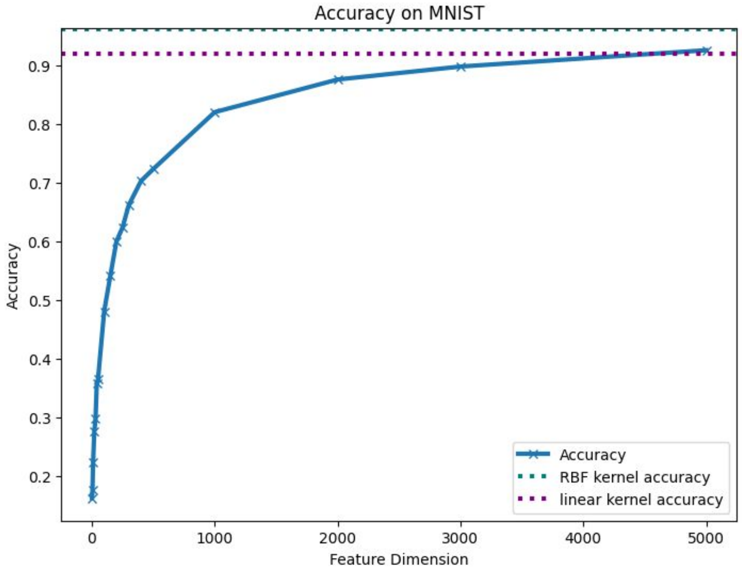


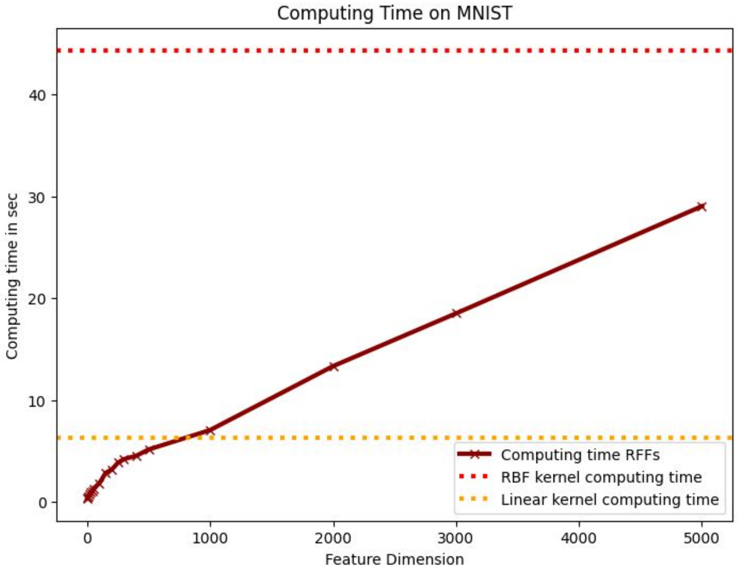
Approximating Kernel Regression



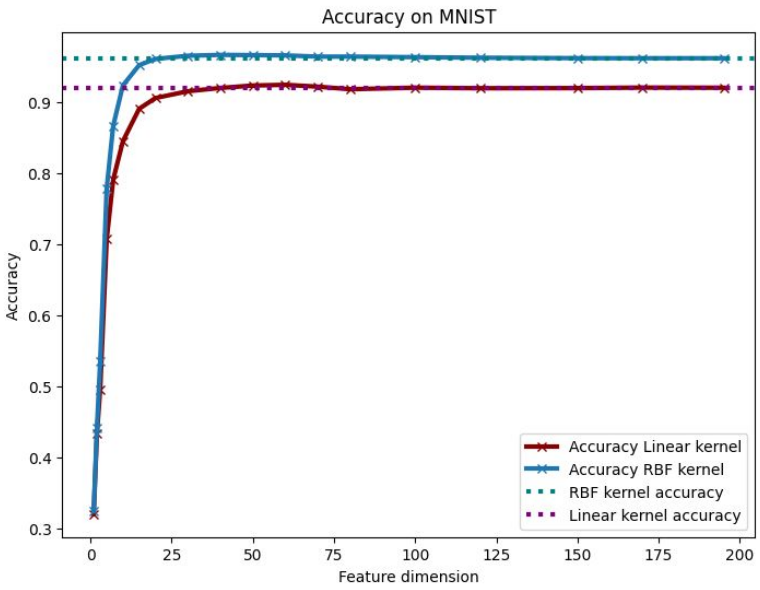


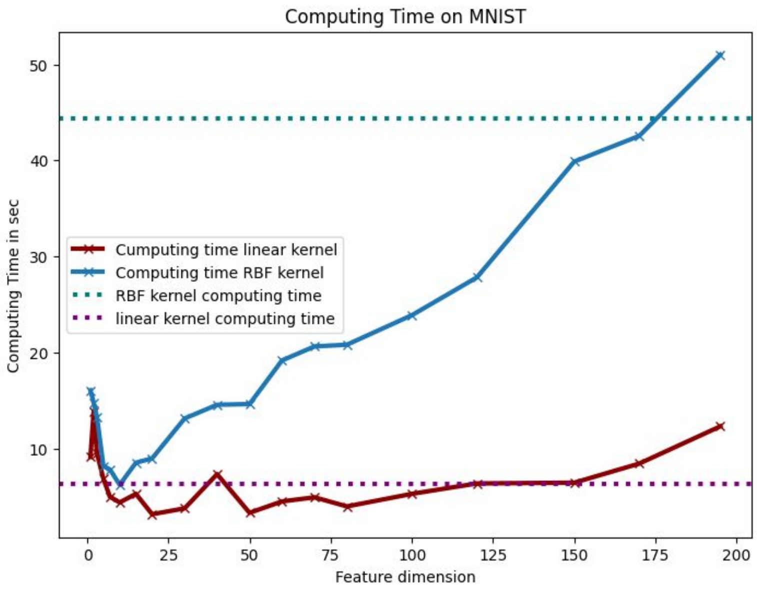
Approximating Kernel Support Vector Machine





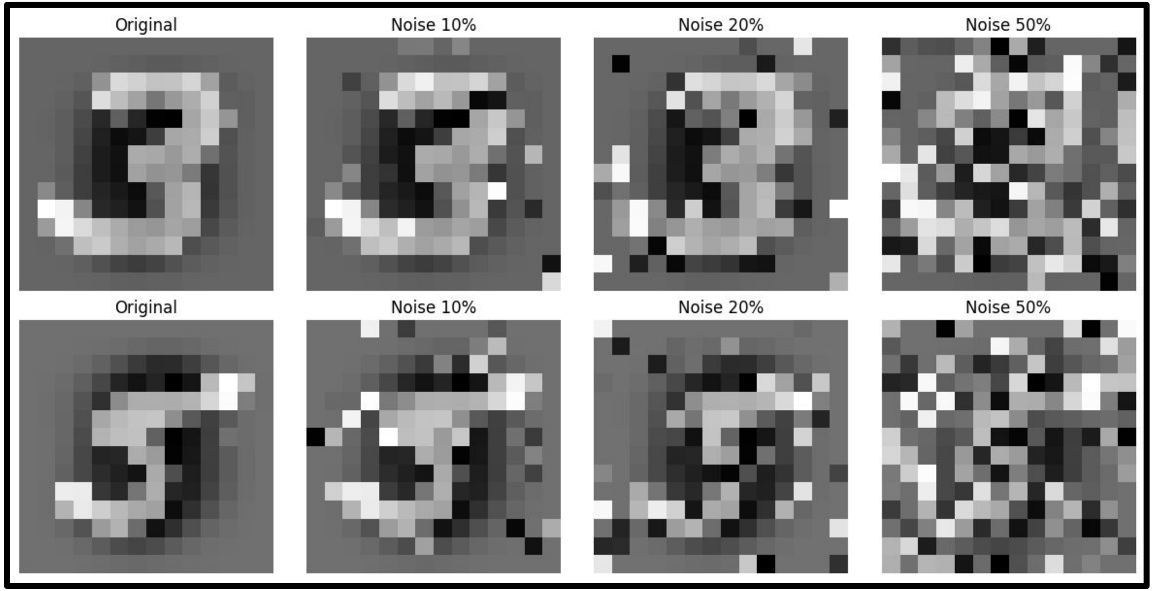
Comparing to PCA



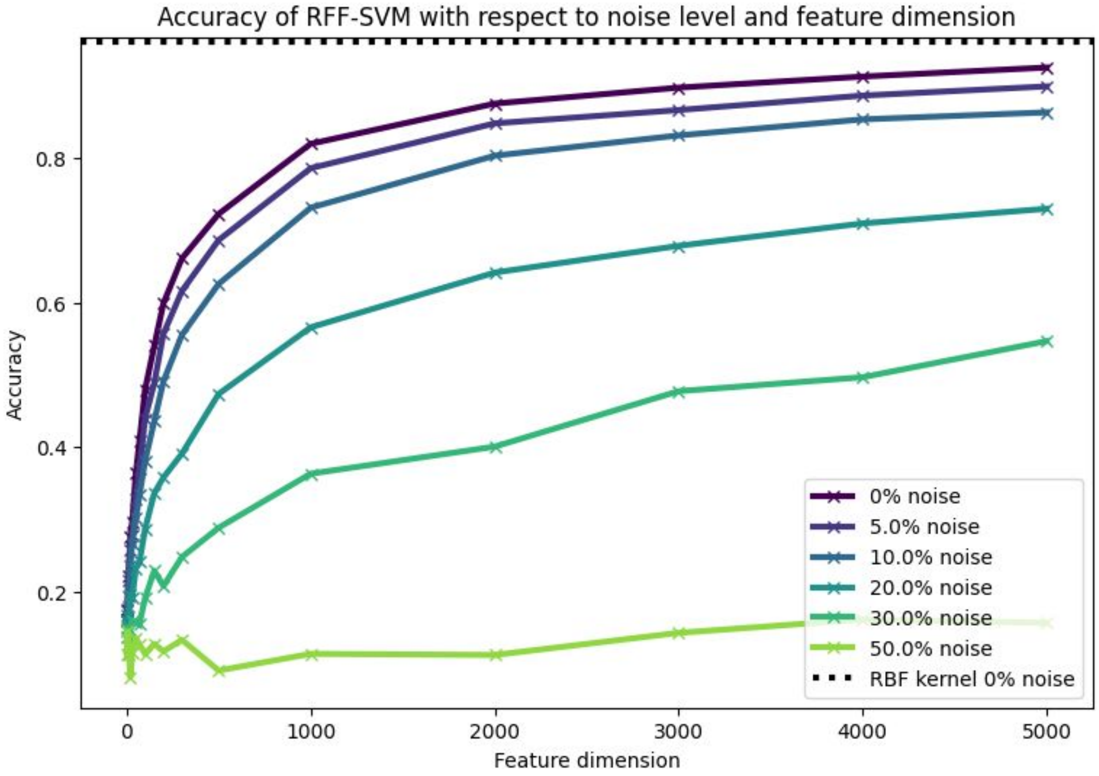


Noisy Data

- evaluating impact of noise
- *p*: prob. of each pixel to be uniform random
- testing for $p = 0$, $p = 0.05$, $p = 0.1$, $p = 0.2$, $p = 0.3$ and $p = 0.5$



Noisy Data

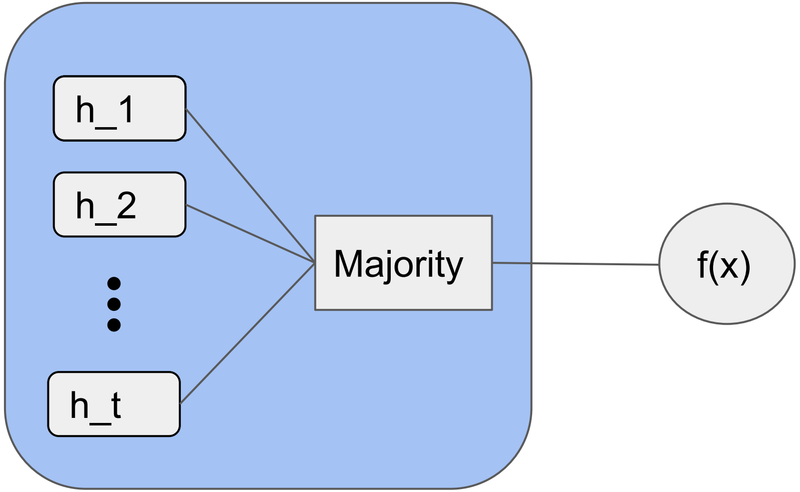


Accuracy of RFF-SVM with respect to noise level and feature dimension
Accuracy
0.2
0.4
0.6
0.8
Feature dimension
0
1000
2000
3000
4000
5000
0% noise
5.0% noise
10.0% noise
20.0% noise
30.0% noise
50.0% noise
RBF kernel 0% noise

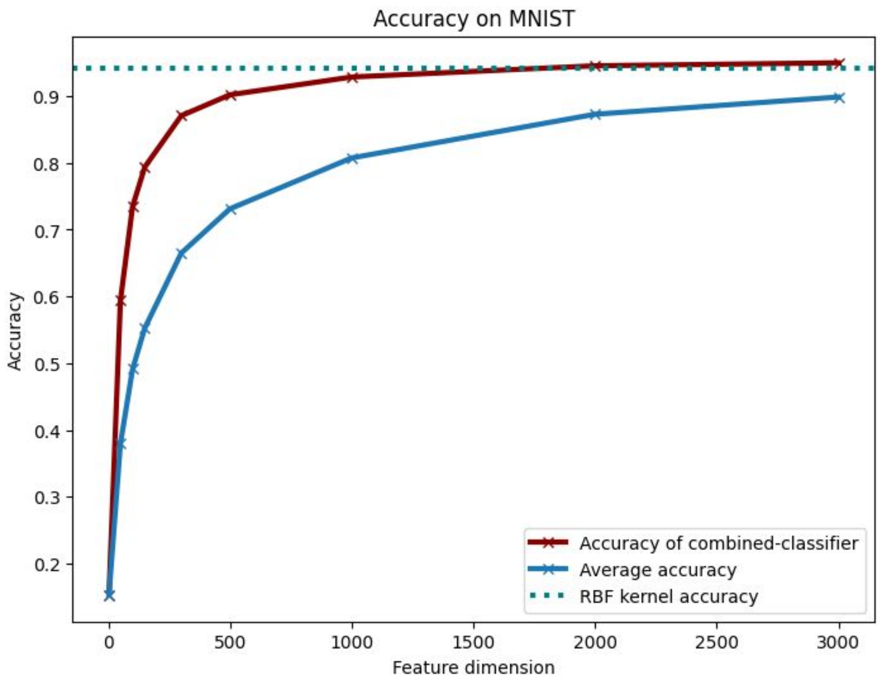
Ensemble Learning

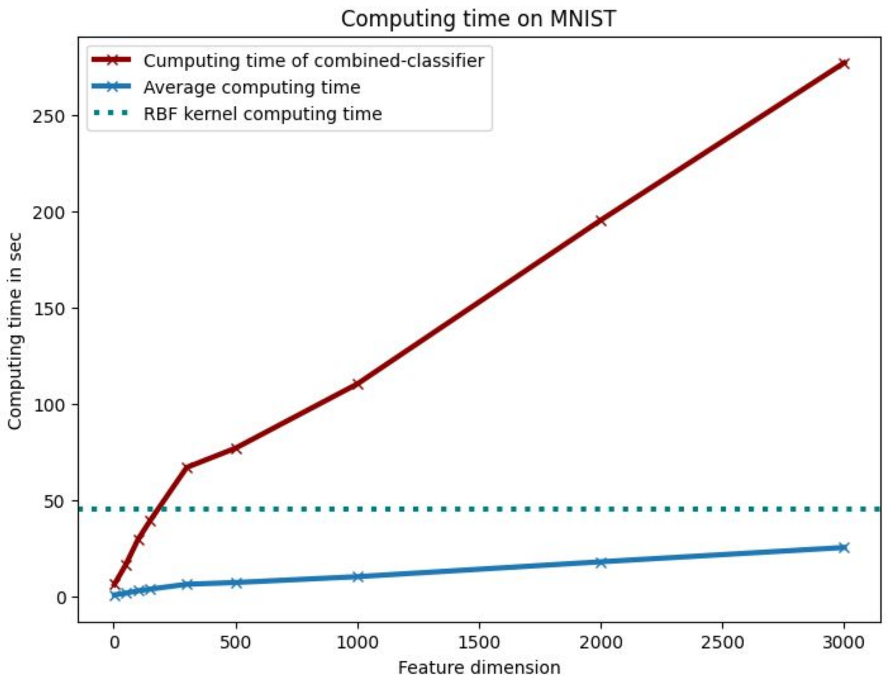
- RFF approximation can be used for ensemble learning
  - Classifiers are “random”
  - Classifiers will produce different Predictions
- Using majority Voting for Combined Classifier

$$\boxed{V(x) = \max_{y \in Y}(h_1(x), h_2(x), \cdots, h_t(x))}$$

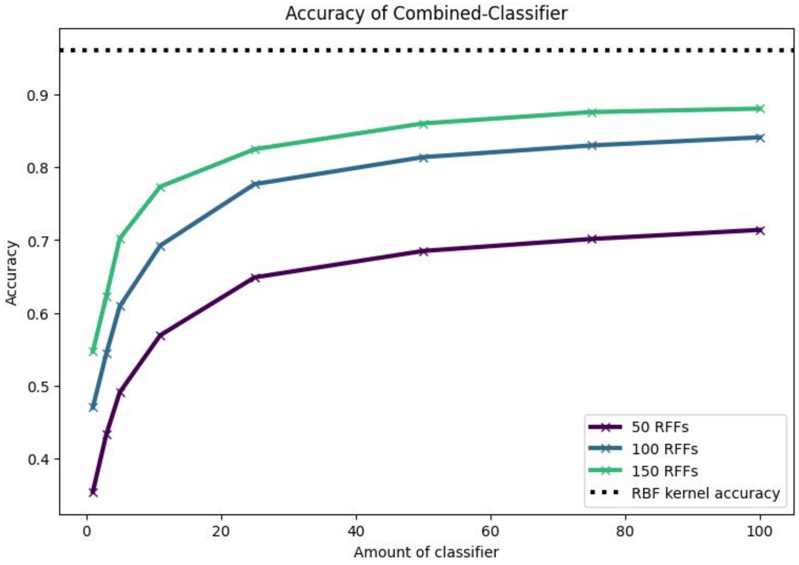


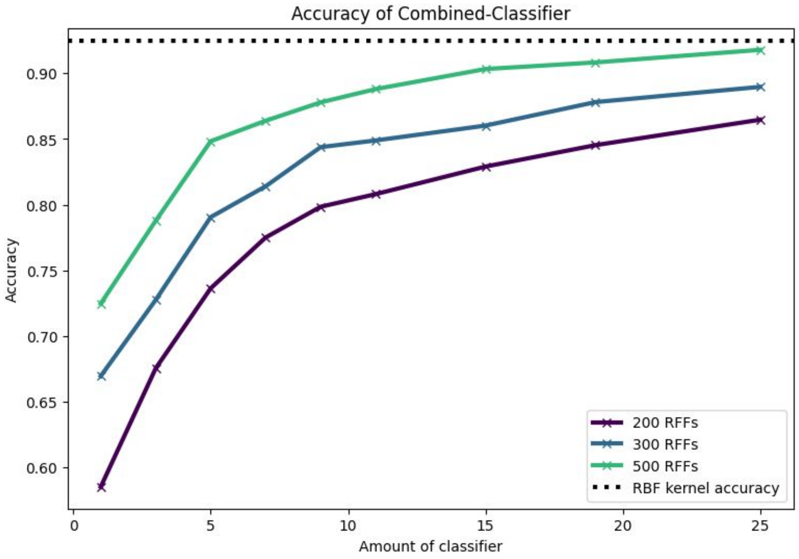
Ensemble Learning





Ensemble Learning





Conclusion and Discussion

- Always a trade-off between time and performance
- Still good performance with way less computing time
  - For regression and Classification

- Kernel approximation with RFFs is well suited for ensemble-learning
- Should be compared with other ensemble methods in future work.

Danke für ihre Aufmerksamkeit

Backup

Boncher's theorem: A continuous and translations-invariant kernel $k(x,y) = k(x-y)$ is positive definite if and only if $k(\Delta)$ is the Fourier transform of a non-negative measure $p(w)$ i.e.

$$k(\Delta) = \int p(w)\exp(iw\Delta)dw$$

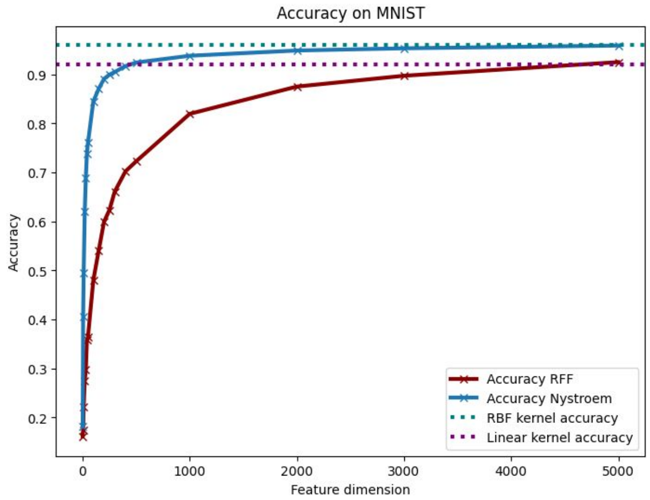
Translations-invariant means that $\forall x, y, z \in X \; k(x,y) = k(x-z, y-z)$. This means that the kernel-function does only depend on the distance between x and y.

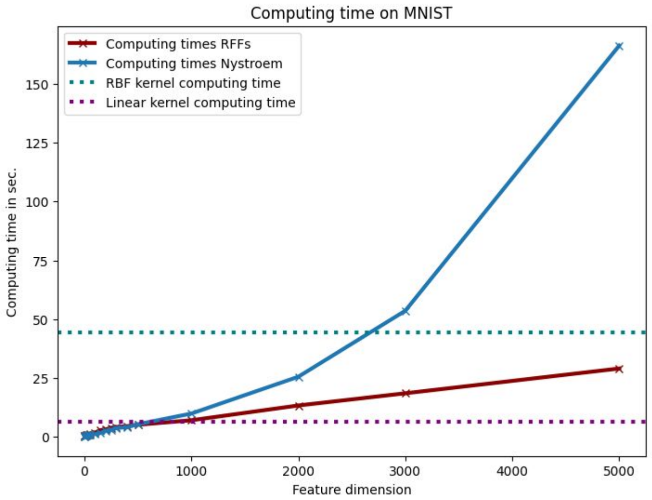
$$\langle z(x), z(y) \rangle = \frac{1}{R}\sum_r z_{w_r}(x), z_{w_r}(y)^* \tag{4.7}$$

It's also possible to show the convergence in probability of the inner product of our approximated features towards the true kernel function, if R is large enough i,e

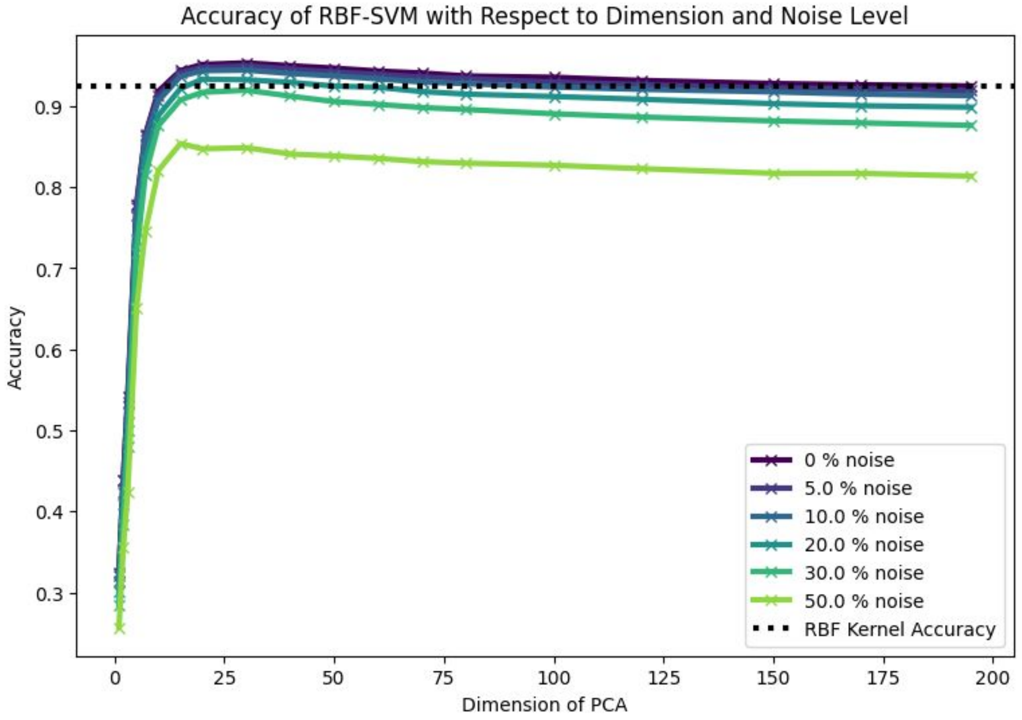
$$\mathbb{P}\left[\sup_{x,y\in X} |\langle z(x), z(y)\rangle - k(x,y)| \geq \epsilon\right] \xrightarrow{R\to\infty} 0 \tag{4.8}$$

Backup


Accuracy on MNIST
Accuracy
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
1000
2000
3000
4000
5000
Feature dimension
Accuracy RFF
Accuracy Nystroem
RBF kernel accuracy
Linear kernel accuracy

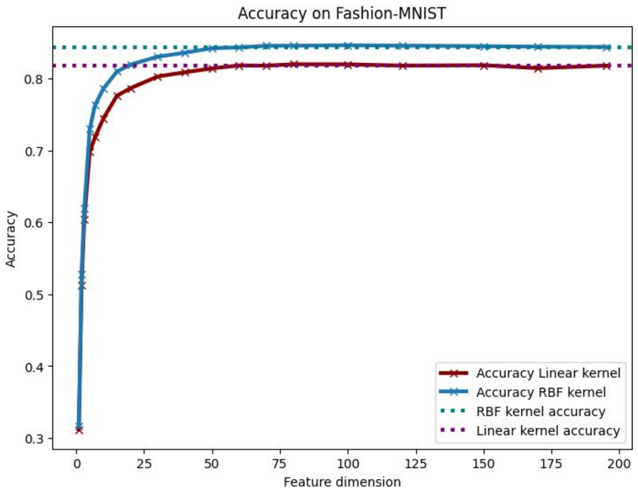

Computing time on MNIST
Computing time in sec.
0
25
50
75
100
125
150
0
1000
2000
3000
4000
5000
Feature dimension
Computing times RFFs
Computing times Nystroem
RBF kernel computing time
Linear kernel computing time

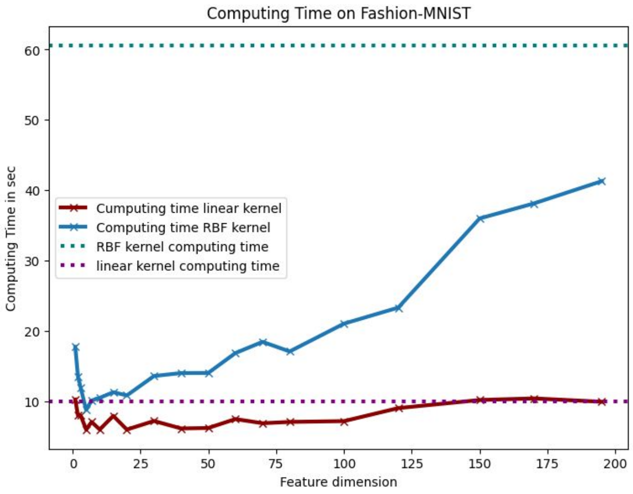
Backup



Accuracy of RBF-SVM with Respect to Dimension and Noise Level
Accuracy
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
25
50
75
100
125
150
175
200
Dimension of PCA
0 % noise
5.0 % noise
10.0 % noise
20.0 % noise
30.0 % noise
50.0 % noise
RBF Kernel Accuracy

Backup


Accuracy on Fashion-MNIST
Accuracy
Feature dimension
0.3
0.4
0.5
0.6
0.7
0.8
0
25
50
75
100
125
150
175
200
Accuracy Linear kernel
Accuracy RBF kernel
RBF kernel accuracy
Linear kernel accuracy


Computing Time on Fashion-MNIST
Computing Time in sec
Feature dimension
10
20
30
40
50
60
0
25
50
75
100
125
150
175
200
Computing time linear kernel
Computing time RBF kernel
RBF kernel computing time
linear kernel computing time