Kernel methods

BIFOLD Aqtivate workshop

Stefan Blücher

10.02.2024

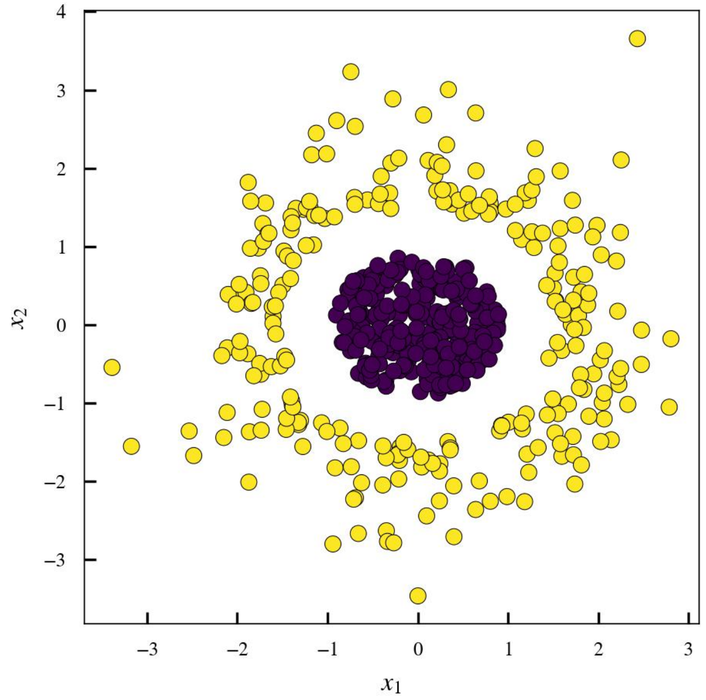
Credits to Lorenz Vaitl
for slides

Motivation

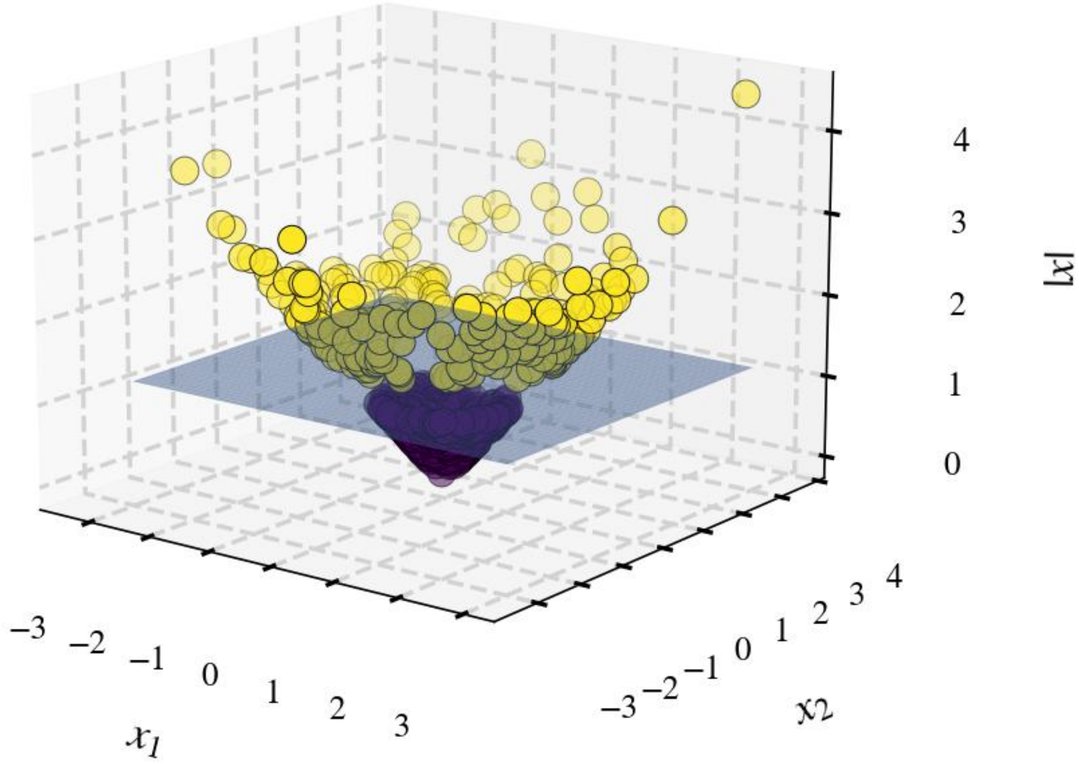
From linear to non-linear models

Better features for linear models

Binary classification problem



Polynomial kernel



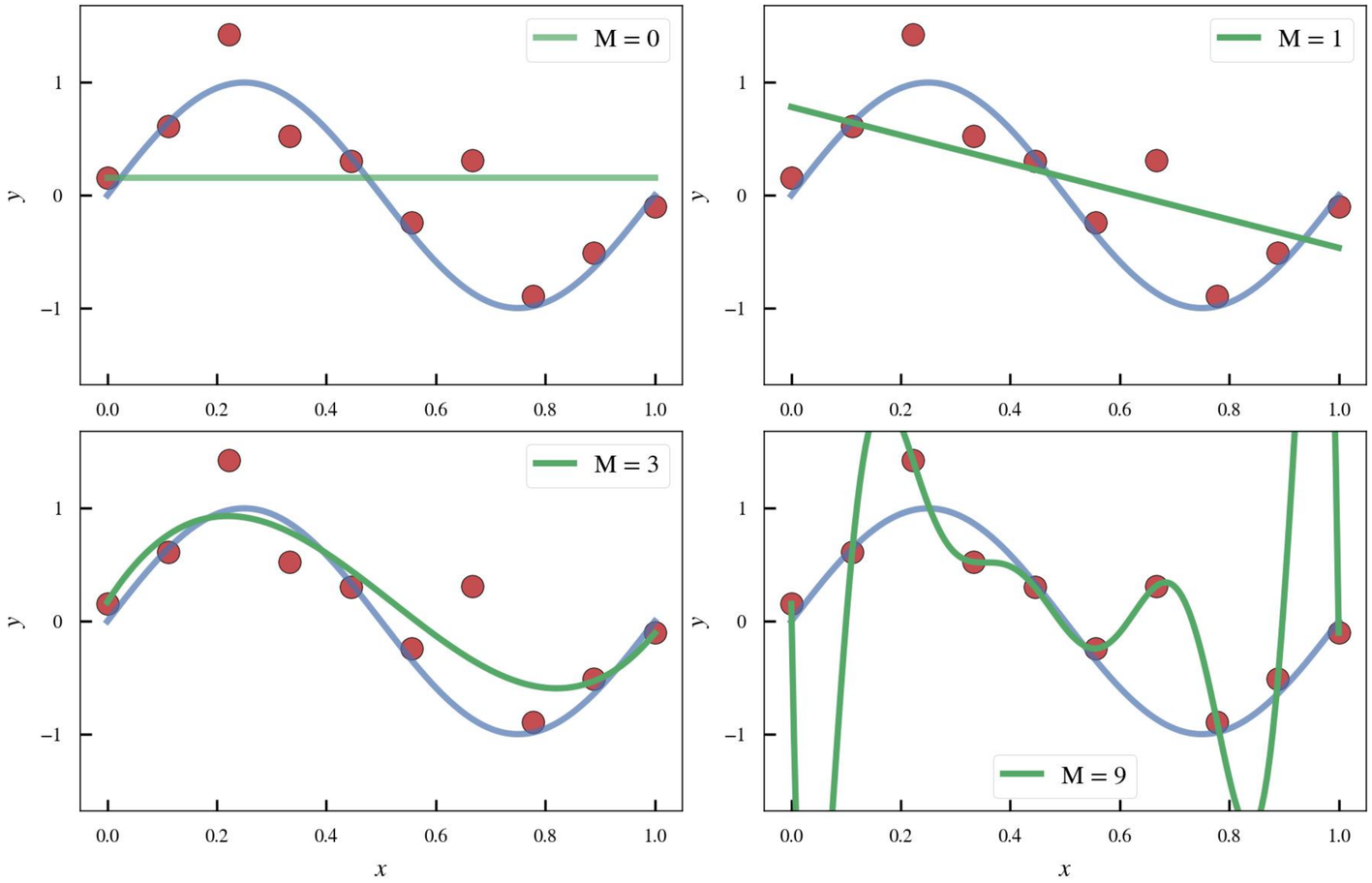
Kernel describing degree-d polynomial

$$k\left(\mathbf{x}_1, \mathbf{x}_2\right) = \left(\mathbf{x}_1^\top \mathbf{x}_2 + c\right)^d$$

Linear regression with polynomial features

$$\hat{y}(x) = w_0 + w_1 \cdot x^1 + \ldots + w_M \cdot x^M$$

$\rightarrow$ Here we are using the basis function
$\phi_M(x) = (x^0, x^1, \ldots, x^M)$,
which has a $(M+1)$-dimensional feature space $\mathcal{F}$

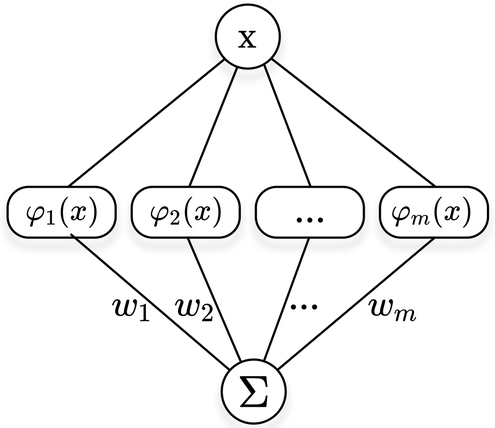


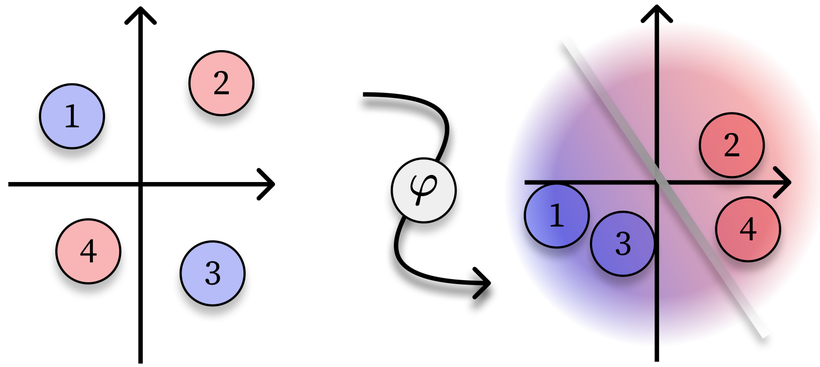
[adapted from Bishop, 2007]

Kernel ridge regression

Kernel trick in practise

Kernel trick





[adapted from Jäkel et al., 2009]

1. Map the data into a (high dimensional) feature space, $\boldsymbol{x} \mapsto \varphi(\boldsymbol{x})$
2. Look for linear relations in the feature space

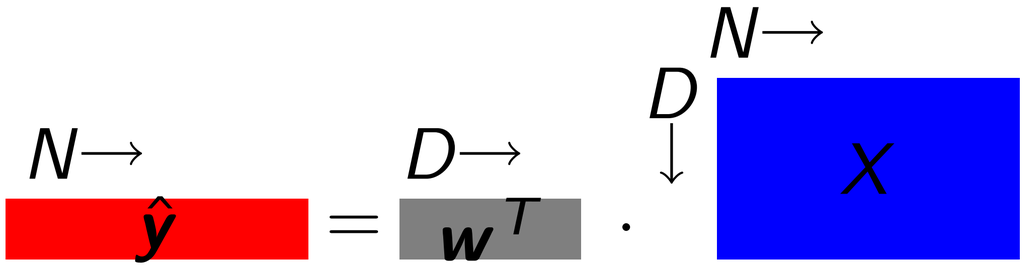
We call this form **dual representation**

Only scalar products $\rightarrow$ we can put in kernels:

$$\boldsymbol{x}_i^\top \boldsymbol{x}_j \rightarrow \varphi(\boldsymbol{x}_i)^\top \varphi(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = K_{ij}$$

Ridge regression

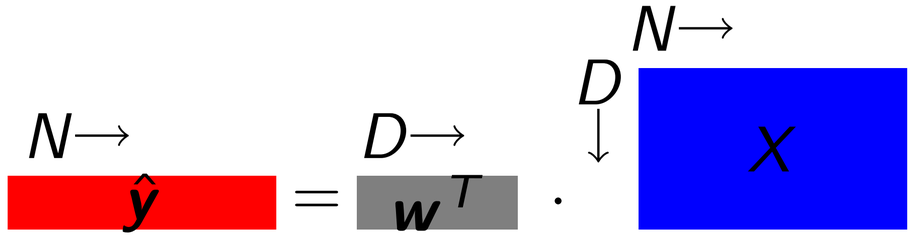
$$\hat{y}(x) = w_0 + w_1 \cdot x^1 + \ldots + w_M \cdot x^M$$



Linear ridge regression looks for a linear combination of features $\boldsymbol{w}$ that minimizes the prediction error and has a small norm
We can write this term in several equivalent ways:

$$\begin{aligned}
\mathcal{E}_{RR}(\boldsymbol{w}) &= \sum_n (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \lambda \sum_d w_d^2 \\
&= ||\boldsymbol{y} - \boldsymbol{w}^\top X||^2 + \lambda ||\boldsymbol{w}||^2 \\
&= \boldsymbol{y}\boldsymbol{y}^\top - 2\boldsymbol{w}^\top X \boldsymbol{y}^\top + \boldsymbol{w}^\top X X^\top \boldsymbol{w} + \lambda \boldsymbol{w}^\top \boldsymbol{w}
\end{aligned}$$

Ridge regression



$$\mathcal{E}_{RR}(\boldsymbol{w}) = \boldsymbol{y}\boldsymbol{y}^\top - 2\boldsymbol{w}^\top X\boldsymbol{y}^\top + \boldsymbol{w}^\top XX^\top \boldsymbol{w} + \lambda \boldsymbol{w}^\top \boldsymbol{w}$$

Computing the derivative w.r.t. $\boldsymbol{w}$ yields

$$\frac{\partial \mathcal{E}_{RR}(\boldsymbol{w})}{\partial \boldsymbol{w}} = -2X\boldsymbol{y}^T + 2XX^T\boldsymbol{w} + 2\lambda \boldsymbol{w}$$

Setting the gradient to zero and rearranging terms the optimal $\boldsymbol{w}$ is

$$\begin{aligned} 2XX^\top \boldsymbol{w} + \lambda 2\boldsymbol{w} &= 2X\boldsymbol{y}^\top \\ (XX^\top + \lambda I)\boldsymbol{w} &= X\boldsymbol{y}^\top \\ \boldsymbol{w} &= (XX^\top + \lambda I)^{-1} X\boldsymbol{y}^\top \end{aligned}$$

Compute inverse in feature space

Kernelize ridge regression

Setting the gradient to zero and rearranging terms the optimal $\boldsymbol{w}$ is

$$
\begin{aligned}
2XX^\top \boldsymbol{w} + \lambda 2\boldsymbol{w} &= 2X\boldsymbol{y}^\top \\
(XX^\top + \lambda I)\boldsymbol{w} &= X\boldsymbol{y}^\top \\
\boldsymbol{w} &= (XX^\top + \lambda I)^{-1} X\boldsymbol{y}^\top
\end{aligned}
$$

Compute inverse in feature space

Setting the gradient to 0 and rearranging terms the optimal $\boldsymbol{w}$ satisfies

$$
\boldsymbol{w} = X \underbrace{\frac{1}{\lambda}(\boldsymbol{y}^T - X^T\boldsymbol{w})}_{:=\boldsymbol{\alpha}(\in\mathbb{R}^{n\times 1})} = \sum_i^n \alpha_i \boldsymbol{x}_i
$$

Linear combination of samples

Kernelize ridge regression

$$\mathcal{E}_{RR}(\boldsymbol{w}) = \boldsymbol{y}\boldsymbol{y}^\top - 2\boldsymbol{w}^\top X\boldsymbol{y}^\top + \boldsymbol{w}^\top XX^\top \boldsymbol{w} + \lambda \boldsymbol{w}^\top \boldsymbol{w}$$

$$\boldsymbol{w} = X\boldsymbol{\alpha}$$

Let's look at the error function $\mathcal{E}_{RR}$ w.r.t. $\boldsymbol{\alpha}$

$$\mathcal{E}_{RR}(\boldsymbol{\alpha}) = \boldsymbol{y}\boldsymbol{y}^\top - 2\boldsymbol{\alpha}^T \underbrace{X^T X}_{K}\boldsymbol{y}^\top + \boldsymbol{\alpha}^T \underbrace{X^T X}_{K}\underbrace{X^\top X}_{K}\boldsymbol{\alpha} + \lambda \boldsymbol{\alpha}^T \underbrace{X^T X}_{K}\boldsymbol{\alpha}$$

Now we can optimize for the new parameters

$$\frac{\partial}{\partial \alpha}\epsilon_{RR}(\alpha) = 0$$

$$\alpha = (K + \lambda I_N)^{-1} K y$$

Compute inverse in data space

Kernelize ridge regression

$$\mathcal{E}_{RR}(\boldsymbol{\alpha}) = \boldsymbol{y}\boldsymbol{y}^\top - 2\boldsymbol{\alpha}^T \underbrace{X^T X}_{K} \boldsymbol{y}^\top + \boldsymbol{\alpha}^T \underbrace{X^T X}_{K} \underbrace{X^\top X}_{K} \boldsymbol{\alpha} + \lambda \boldsymbol{\alpha}^T \underbrace{X^T X}_{K} \boldsymbol{\alpha}$$

We call this form **dual representation**

Only scalar products $\rightarrow$ we can put in kernels:

$$\boldsymbol{x}_i^\top \boldsymbol{x}_j \rightarrow \varphi(\boldsymbol{x}_i)^\top \varphi(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = K_{ij}$$

We write $k(X, X)$ as K

Compare training strategies

Kernel Ridge Regression:

$$\begin{aligned}
\boldsymbol{\alpha} &= \frac{1}{\lambda}(\boldsymbol{y}^\top - \varphi(X_{train})^\top \boldsymbol{w}) \\
\lambda\boldsymbol{\alpha} &= \boldsymbol{y}^\top - \varphi(X_{train})^\top \varphi(X_{train})\boldsymbol{\alpha} \\
\boldsymbol{y}^\top &= (\varphi(X_{train})^\top \varphi(X_{train}) + \lambda I)\boldsymbol{\alpha} \\
\boldsymbol{\alpha} &= (\varphi(X_{train})^\top \varphi(X_{train}) + \lambda I)^{-1}\boldsymbol{y}^\top \\
\boldsymbol{\alpha} &= (K + \lambda I)^{-1}\boldsymbol{y}^\top
\end{aligned}$$

This $\boldsymbol{\alpha}$ minimizes $\mathcal{E}_{RR}(\boldsymbol{\alpha})$.

Ridge Regression

Train $\boldsymbol{w}$ which minimizes $\mathcal{E}_{RR}(\boldsymbol{w})$:

$$\boldsymbol{w} = (\varphi(X)\varphi(X)^T + \lambda I)^{-1}\varphi(X)y^T$$

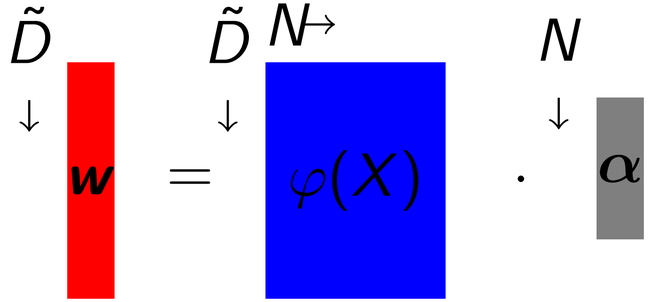
For KRR we write $\varphi(\boldsymbol{x}) \in \mathbb{R}^{\tilde{D}}$ instead of $\boldsymbol{x}$

We defined $\alpha_i = \frac{1}{\lambda}(y_i - \varphi(\boldsymbol{x}_i)^T \boldsymbol{w})$

Optimal $\boldsymbol{w} = \varphi(X_{train})\boldsymbol{\alpha}$

KRR trains $\boldsymbol{\alpha} \in \mathbb{R}^n$, RR trains $\boldsymbol{w} \in \mathbb{R}^{\tilde{D}}$

For $\varphi(x)^T\varphi(x') = k(x, x')$ the two models are equivalent (but not the runtime complexity)



Predictions for new data

$$
\begin{aligned}
y_{new} &= \boldsymbol{w}^T \varphi(\boldsymbol{x}_{new}) \\
&= (\varphi(X_{train})\boldsymbol{\alpha})^T \varphi(\boldsymbol{x}_{new}) \\
&= \boldsymbol{\alpha}^T \varphi(X_{train})^T \varphi(\boldsymbol{x}_{new}) \\
&= \boldsymbol{\alpha}^T k(X_{train}, \boldsymbol{x}_{new}) \\
&= \boldsymbol{y}_{train}(K + \lambda I)^{-1} k(X_{train}, \boldsymbol{x}_{new})
\end{aligned}
$$

Optimal $\boldsymbol{\alpha} = (K + \lambda I)^{-1} \boldsymbol{y}^T$

$\boldsymbol{w} = \varphi(X_{train})\boldsymbol{\alpha}$

$K(a, b) = K(b, a)^T$

We call $K = k(X_{train}, X_{train})$ the Gram matrix

Summary

Kernel ridge regression

- Input: kernel function $k(\boldsymbol{x}, \boldsymbol{x}') = \varphi(\boldsymbol{x})^T \varphi(\boldsymbol{x}')$, regularization hyperparameter λ, training dataset $(X_{train}, \boldsymbol{y}_{train})$ and test datapoints X_{test}
- Training [2]:

$$\boldsymbol{\alpha} = (k(X_{train}, X_{train}) + \lambda I)^{-1} \boldsymbol{y}_{train}^T$$

- Predicting:

$$\hat{\boldsymbol{y}}_{test} = \boldsymbol{\alpha}^T k(X_{train}, X_{test})$$

[2] Since we need X_{train} again for predictions, we cannot forget the data like for RR

Theory

Why all this works

Constructing a kernel from features

Definition

We define a Kernel function k as a dot product in feature space $\mathcal{F}$ where

$$\varphi : \mathcal{X} \to \mathcal{F}$$
$$x \mapsto \varphi(x)$$

and so

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$
$$k(x_i, x_j) = \varphi(x_i)^T \cdot \varphi(x_j)$$

Kernel Trick (Kernel Substitution)

For any algorithm that can be formulated such that the input vector $\boldsymbol{x}$ enters only in terms of scalar products $\boldsymbol{x}^T \cdot \boldsymbol{x}'$, we can replace each scalar product by a kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \varphi(\boldsymbol{x})^T \cdot \varphi(\boldsymbol{x}')$.

Why do it?

For $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{d \times 1} \Rightarrow \boldsymbol{x}^T \cdot \boldsymbol{x}' \in \mathbb{R}^{1 \times 1}$ regardless of d

For $\varphi(\boldsymbol{x}) \in \mathbb{R}^{\tilde{d} \times 1}$ instead of $\boldsymbol{x}$ with typically $\tilde{d} \gg d$

$\Rightarrow$ we get more complex (powerful) models

By using kernel $k(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}^{1 \times 1}$

We do not need to explicitly calculate high-dimensional $\varphi(\boldsymbol{x})$

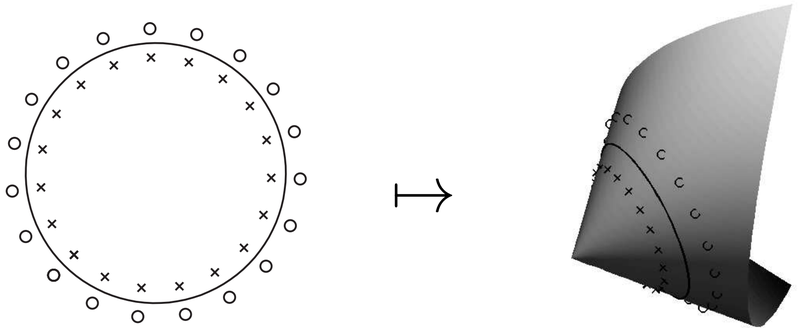
Example for a kernel

$$\varphi : \boldsymbol{x} = (x_1, x_2)^\top \mapsto \varphi(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

The corresponding kernel:

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{y}) &= \varphi(\boldsymbol{x})^\top \cdot \varphi(\boldsymbol{y}) \\ &= (x_1^2, \sqrt{2}x_1x_2, x_2^2) \cdot (y_1^2, \sqrt{2}y_1y_2, y_2^2)^T \\ &= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 \\ &= (x_1y_1 + x_2y_2)^2 = (\boldsymbol{x}^\top \boldsymbol{y})^2 \end{aligned}$$

Visualizing $\varphi(\boldsymbol{x})$



$\rightarrow$ With $k(\cdot, \cdot)$ we implicitly work in $\mathbb{R}^3$, but only operate in $\mathbb{R}^2$

Definition (Positive semi-definite symmetric kernels)

A kernel

$$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

is said to be positive semi-definite symmetric

if for any $\{x_1, \ldots, x_n\} \subseteq \mathcal{X}$, the matrix $K = [K(x_i, x_j)]_{ij} \in \mathbb{R}^{n \times n}$ is symmetric positive semidefinite. [1]

Symmetric Matrix:

$$A = A^\intercal$$

Positive semi-definite Matrix:

$$x^\intercal A x \geq 0 \quad \forall\, x$$

alternatively, all eigenvalues λ of A non-negative.

[1] For ease of notation we may use $K(X, X') = [K(\boldsymbol{x}_i, \boldsymbol{x}'_j)]_{ij}$ for describing the matrix of the kernel-function evaluated on all sample-pairs. $K(X, X)$ is often called the *Gram matrix of* X.

Definition

We define a Kernel function k as a dot product in feature space $\mathcal{F}$ where

$$\varphi : \mathcal{X} \to \mathcal{F} \qquad k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

$$x \mapsto \varphi(x) \qquad k(x_i, x_j) = \varphi(x_i)^T \cdot \varphi(x_j)$$

Definition (Positive semi-definite symmetric kernels)

if for any $\{x_1, \ldots, x_n\} \subseteq \mathcal{X}$, the matrix $K = [K(x_i, x_j)]_{ij} \in \mathbb{R}^{n \times n}$ is symmetric positive semidefinite. [1]

$$\begin{aligned} K = \kappa(X, X) &= \varphi(X)^T \cdot \varphi(X) \\ &= (U\Sigma V^T)^T \cdot U\Sigma V^T = V\Sigma\,(U^T U)\,\Sigma V^T \\ &= V\,\Sigma^2\,V^T \end{aligned}$$

Feature space

Singular value decomposition

Positive spectrum

Mercer's Theorem [Mercer, 1909]

Non-technical formulation:
If kernel is positive semi-definite symmetric then one can find a map φ such that $k(x, x') = \varphi(x)^T \varphi(x')$ for almost all $x \in \mathcal{X}$

$\rightarrow$ To see if your kernel is valid, show it is positive semi-definite symmetric!

$\rightarrow$ You can construct kernels from other kernels, e.g. by sum, product or exponentiating

Alternatively show $k(x, x') = \varphi(x)^T \varphi(x')$

Popular kernel functions

Linear Kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^\top \boldsymbol{x}_j$$

Polynomial Kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^\top \boldsymbol{x}_j + c)^p$$

Note that we are never directly operating in feature space

Gaussian Kernel/ RBF (more on this later)

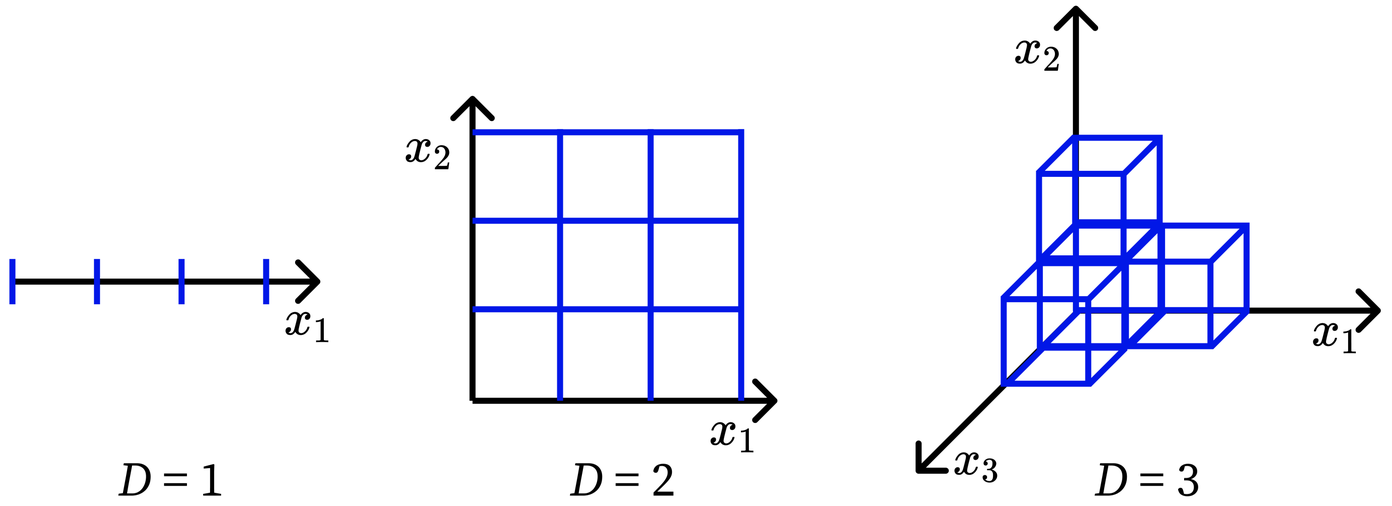
$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 / -2\sigma^2}$$

Curse of dimensionality

A big problem with high dimensional features spaces

When the dimensionality increases, the volume of the space increases so fast that the available data becomes sparse

The amount of data needed for a reliable result often grows exponentially with the dimensionality



[Adapted from Bishop, 2007]

Why should this work?

How do we find the optimal weight

$$\boldsymbol{w} \in \mathbb{R}^{\tilde{d}} \ ?$$

Seems impossible with $\tilde{d} >> n$

Why it works!

Representer Theorem [Kimeldorf and Wahba, 1971]

The minimizing function f^* of a (regularized) error function on some training data $\boldsymbol{x}_i$ can be written in terms of the kernel k as

$$f^*(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)\,.$$

I.e. only scalar products between $\varphi(\boldsymbol{x})$ and the training data $\varphi(\boldsymbol{x}_i)$ are relevant.
For the model $f(x) = \boldsymbol{w}^\top \varphi(\boldsymbol{x})$ this means that $\boldsymbol{w}$ can be written as

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i \varphi(\boldsymbol{x}_i)\,.$$

Kernels as similarity measures

$$f^*(\boldsymbol{x}_{\text{new}}) = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}_{\text{new}}, \boldsymbol{x}_i)$$

Kernel methods are *memory-based* methods:

- store the entire training set
- define similarity of data points by kernel function

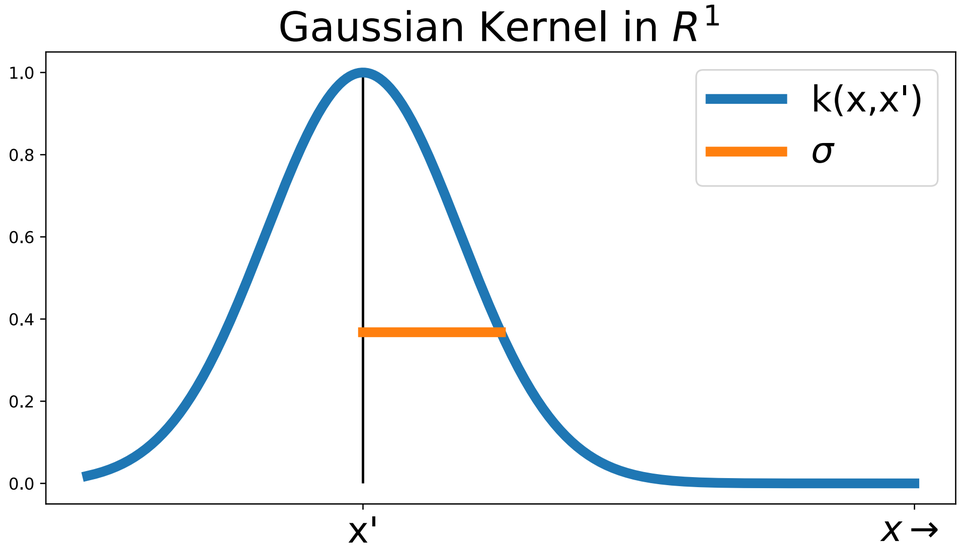
$$k(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

- new predictions require comparision with previously learned examples

Law of universal generalization

- Roger Shepard argued that the **Perceptual Similarity** of new data $\boldsymbol{x}$ decays exponentially with distance from prototype $\boldsymbol{x}'$ [Shepard, 1987]

  $\rightarrow$ motivation for using the Gaussian Kernel



Gaussian Kernel (RBF Kernel)

$$k(\boldsymbol{x}', \boldsymbol{x}) = e^{\frac{-(\boldsymbol{x}' - \boldsymbol{x})^2}{2\sigma^2}}$$

Because $\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$, the basis function $\varphi(\boldsymbol{x})$ is infinite dimensional

Kernels as similarity measures

Kernel Ridge Regression:

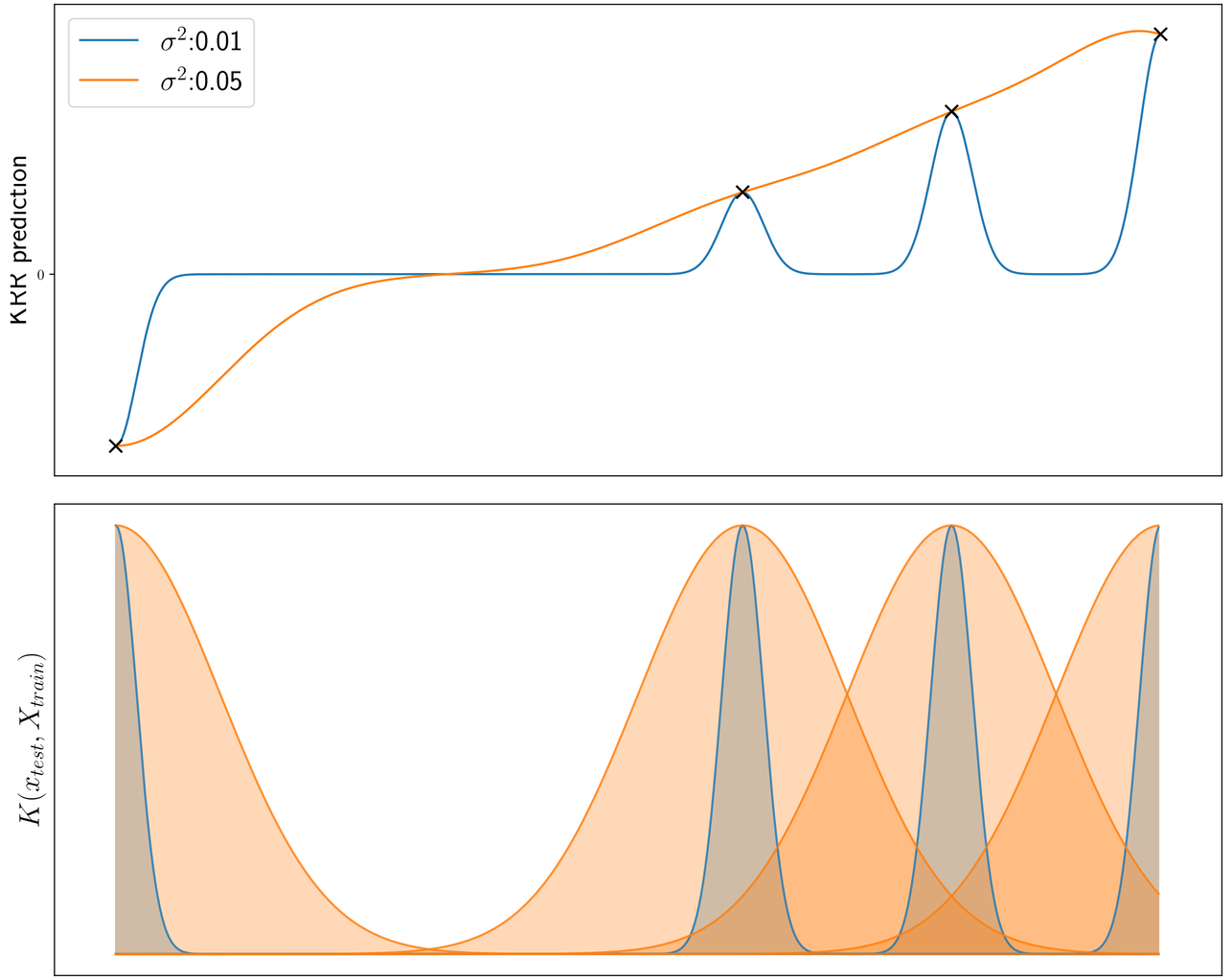
Example

KRR with Gaussian Kernels:

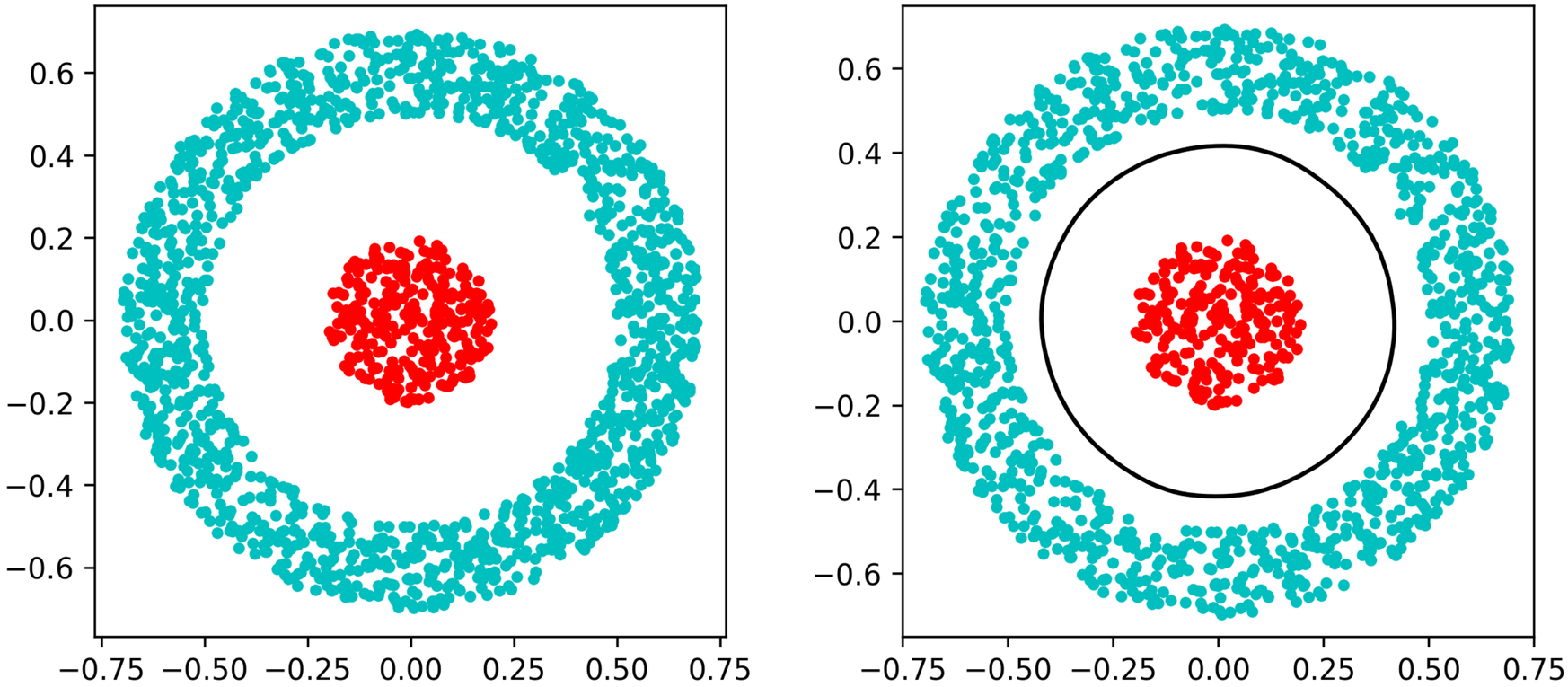
$$k(x, x') = \exp\left\{-\frac{||x - x'||^2}{2\sigma^2}\right\}$$

Predictions:

$$y_{new} = y_{train}(K+\lambda I)^{-1}k(X_{train}, \boldsymbol{x}_{new})$$



We can also kernelize LDA with e.g. a Gaussian Kernel ($\sigma = 0.1$)

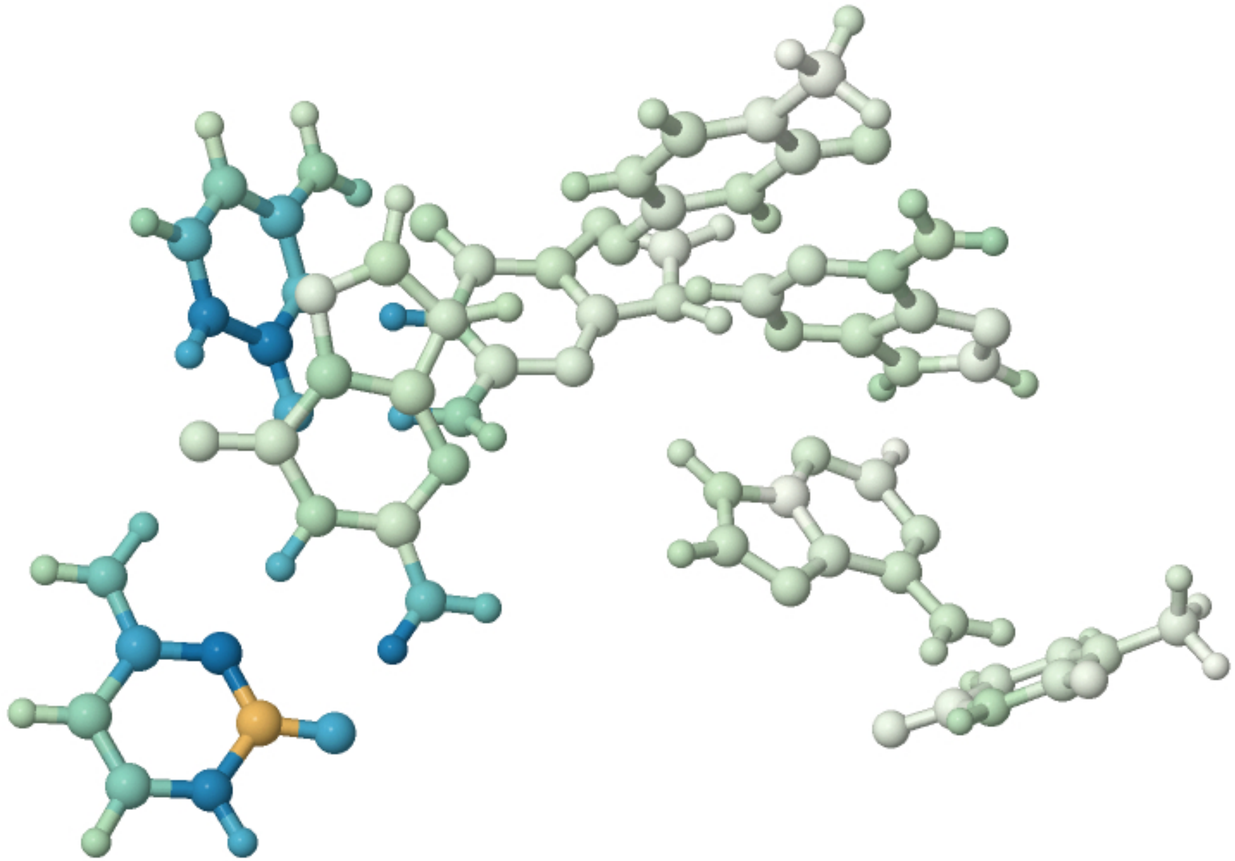


$\rightarrow$ We are able to do non-linear classification!

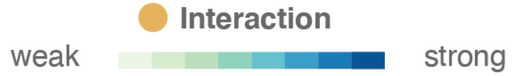
Practical examples

[Gaussian processes], NTK and sGDML

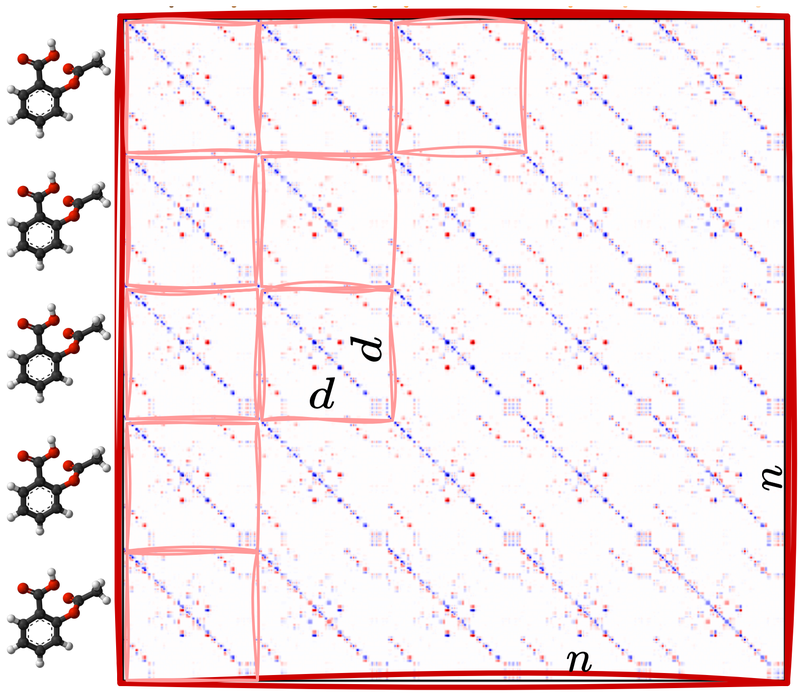
Machine learning force fields with symmetrized kernels



All atoms interact in quantum many-body systems.

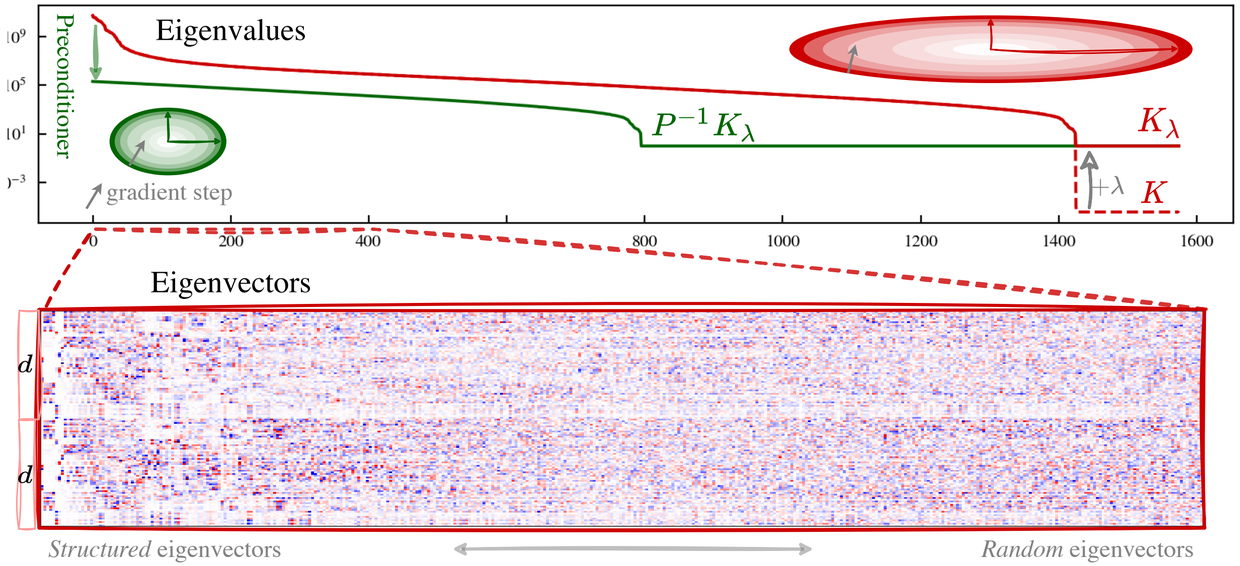


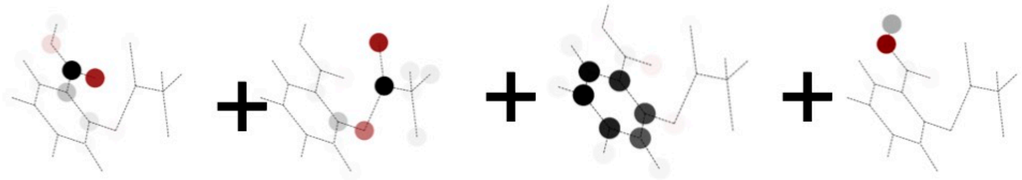
Kernel between molecules



Dynamics
(positions)

Singular value decomposition

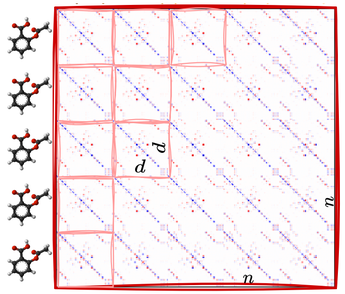




k prototype molecules

Blücher, Stefan, Klaus-Robert Müller, and Stefan Chmiela. "Reconstructing Kernel-Based Machine Learning Force Fields with Superlinear Convergence." *Journal of Chemical Theory and Computation* (2023).

Kernel matrix



Training solution

$$\alpha = (\mathbf{K} + \lambda \mathbf{l})^{-1} \mathbf{y}$$

Invert kernel matrix

Cholesky decomposition

VS.

Iterative solver

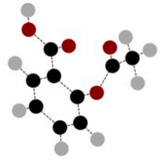
Closed form

Requires storing full matrix

Convergence on problem

Correlations induce bad conditioning

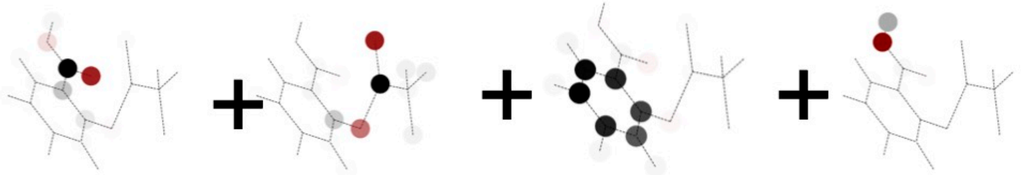
Correlated system



Kernel matrix K

Shift computational costs

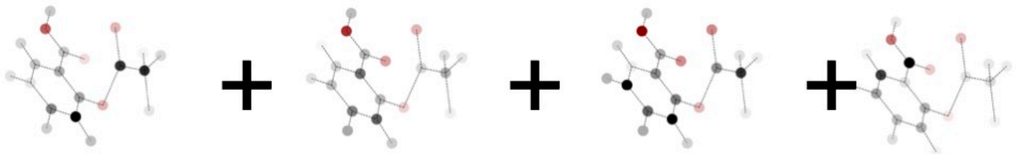
Preconditioner



k prototype molecules

+

Conjugate gradient

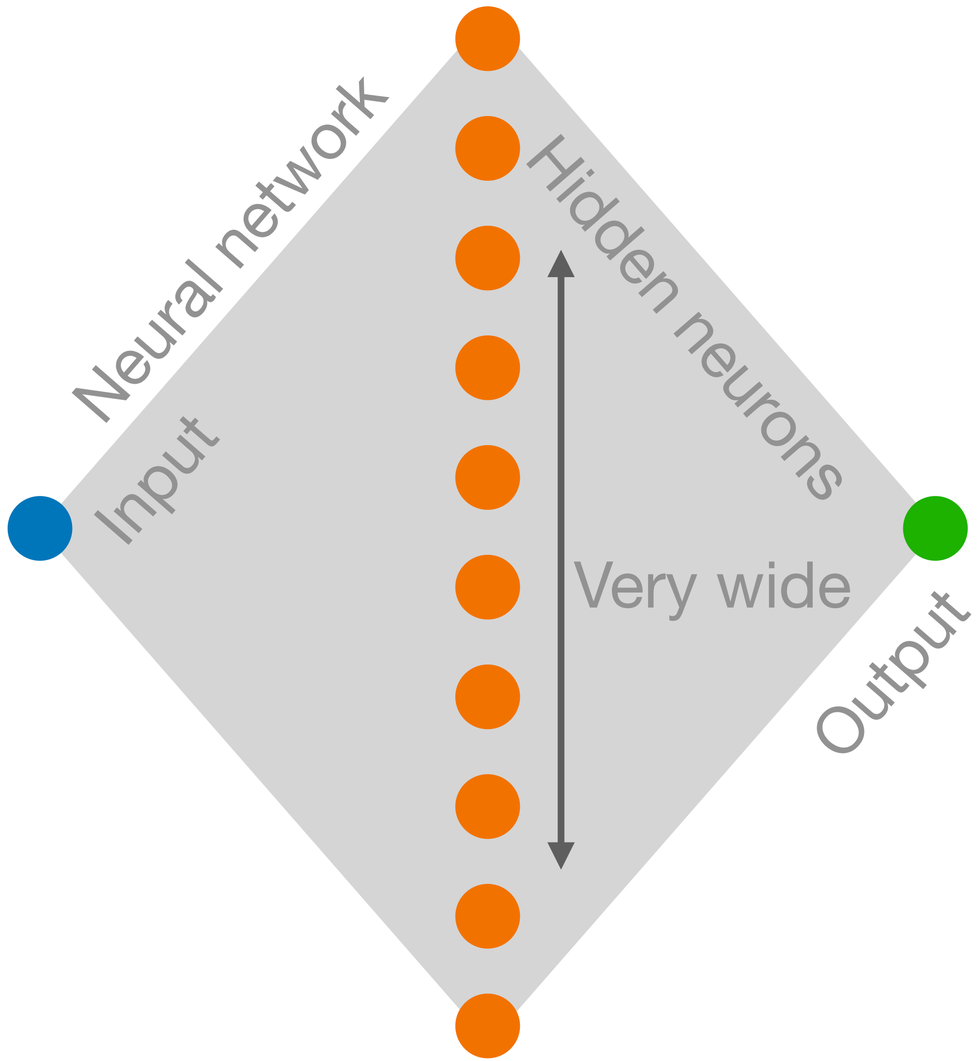


summarize statistical fluctuations (# steps)

Blücher, Stefan, Klaus-Robert Müller, and Stefan Chmiela. "Reconstructing Kernel-Based Machine Learning Force Fields with Superlinear Convergence." *Journal of Chemical Theory and Computation* (2023).

Neural Tangent Kernels (NTK)

- Theory for deep (wide) neural networks
- Converge to kernel model
- https://www.youtube.com/watch?v=raT2ECrvbag



Summary

Kernel methods — Pros and Cons

+ Powerful Modeling Tool
  (non-linear problems become linear in kernel space)
+ Omnipurpose Kernels
  (Gaussian works well in many cases)
+ Kernel methods can handle symbolic objects
+ When you have less data points than your data has dimensions kernel methods can offer a dramatic speedup

– Difficult to understand what's happening in kernel space
– Model complexity increases with number of data points

→ If you have too much data, kernel methods can be slow

References

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2007.

C. F. Gauß. Theoria motus corporum coelestium in sectionibus conicis solem ambientium. *Göttingen*, 1809.

A. E. Hoerl and R. W. Kennar. Ridge regression: Applications to nonorthogonal problems. *Technometrics*, 12(1):69–82, 1970.

F. Jäkel, B. Schölkopf, and F. A. Wichmann. Does cognitive science need kernels? *Trends Cogn Sci*, 13(9):381–388, 2009. doi: 016/j.tics.2009.06.002.

G. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *Journal of mathematical analysis and applications*, 33(1):82–95, 1971.

A.-M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*, chapter Sur la methode des moindres quarres. Firmin Didot, http://imgbase-scd-ulp.u-strasbg.fr/displayimage.php?pos=-141297, 1805.

J. Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458):415–446, 1909.

R. N. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–23, 1987.

A. N. Tychonoff. On the stability of inverse problems. *Doklady Akademii Nauk SSSR*, 39(5):195–198, 1943.