STATG019 – Selected Topics in Statistics 2015

Lecture 1

An Introduction to Kernels

Dr Franz J. Király

Scope of the course

Part 1: Kernel Methods (Franz)

- Hands-on crash course on kernels
- Focus on „data analysis“ part
- Selection of practical kernel methodology
- Complements well topics in COMPGI13
  - (by Arthur Gretton, focus on learning theory and hypothesis testing)

Part 2: Spatial Point Processes (Simon)

- Replace „kernels“ above with „spatial point processes“

Course organization

On-line course moodle ID 16602

on-line activities, course info and materials

Lecture Wednesdays, 9 - 11 am

Attendance is **mandatory**

First lecture Jan 14, last lecture Mar 25 (no lecture in reading week)

Slides and course script available on moodle

Lecture is videotaped and availble on Lecturecast/moodle

Feel free to suggest content you want to learn about

Tutorials and/or practical sessions

Thursday, 11am - 1pm location varies

May take place irregularly, depending on demand

Attendance is **non-compulsory**

Discussion of theoretical or practical exercises, group presentations, etc…

Your choice.

Table of contents

Lecture 1: Introduction to kernels

Main concepts and theoretical results, learning guarantees

Kernel PCA and kernel ridge regression

Some notes on R and kernlab

Lecture 2: the kernel support vector machine

The linear support vector machine, duality

Hard- and soft-margin two-class SVM

The one-class SVM Support vector regression

Lecture 3: Gaussian processes and kernel learning

Potential further lecture topics:

Algorithms: kernel discriminants, kernel k-means, kernel quantile regression
kernel CCA, kernel MMD, kernel relevance vector machine

Large-scale learning with kernels, subset methods and Nyström-approximation

Combinatorial kernels: string kernels, graph kernels

Invariance kernels Vapnik-Chervonenkis learning theory

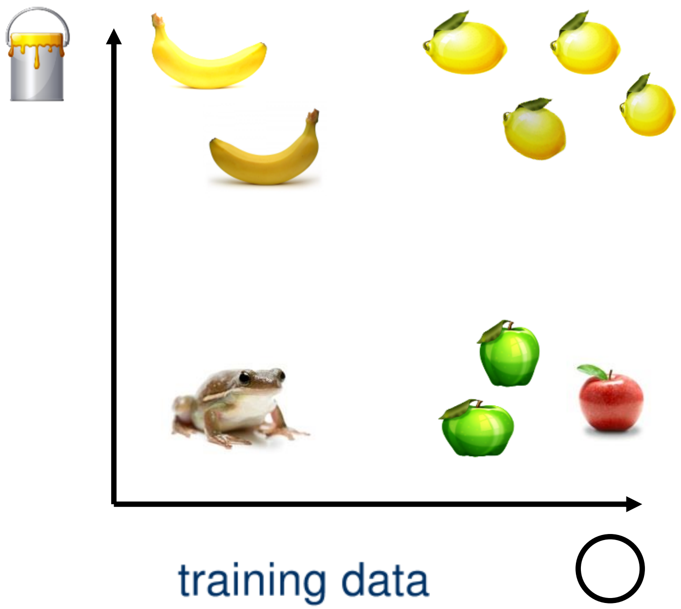
Outlier detection, novelty detection On-line kernel learning

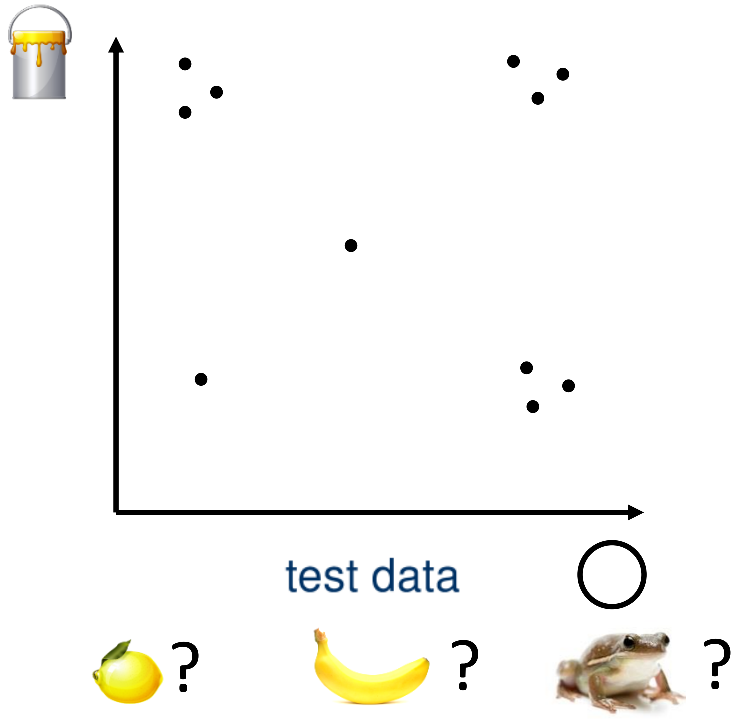
An informal overview of things to come

Problem types in Statistical Machine Learning

Supervised

Some data is labelled by expert/oracle (mostly, training data)

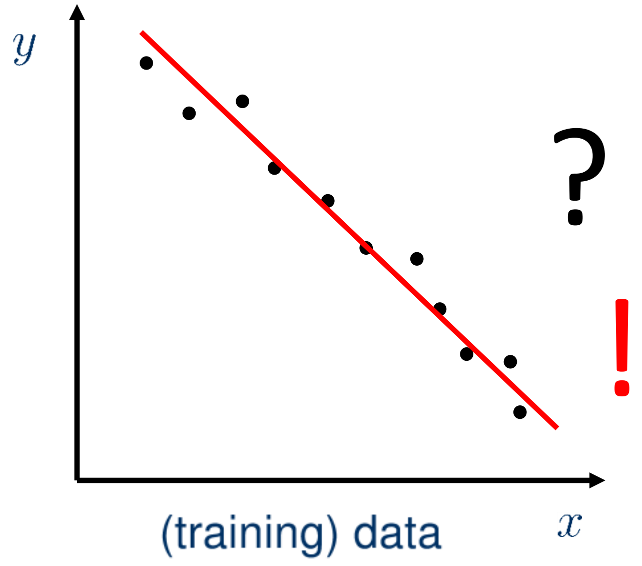


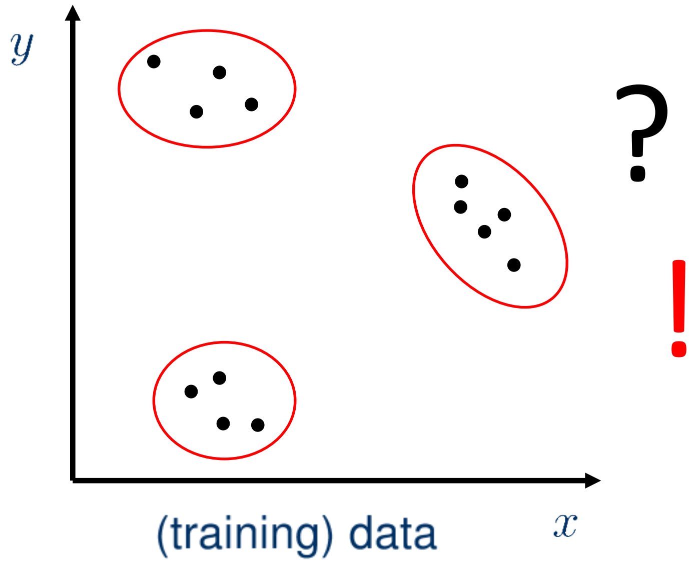


Problem types in Statistical Machine Learning

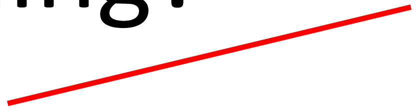
Unsupervised

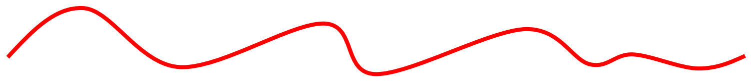
Data is not pre-labelled





What is Kernel Learning?

Most scalable algorithms employ linear principles 

Most data exhibit non-linear features 

Kernels allow to make linear algorithms work on non-linear features

Idea: replace scalar product in algorithm by “kernel product”

(this is neo-classical: Schölkopf 2002 – Learning with Kernels)

Example kernels:

$$k(x,y) = \langle x, y \rangle$$

“linear kernel” (nothing replaced)

$$k(x,y) = \exp\left(-\frac{1}{2\sigma^2}\|x-y\|^2\right)$$

“Gauss kernel” (measures closeness)

$$k(x,y) = \langle x, y \rangle^d$$

“homogenous polynomial kernel

$$k(x,y) = (\theta\langle x, y \rangle + 1)^d$$

“inhomogenous polynomial kernel

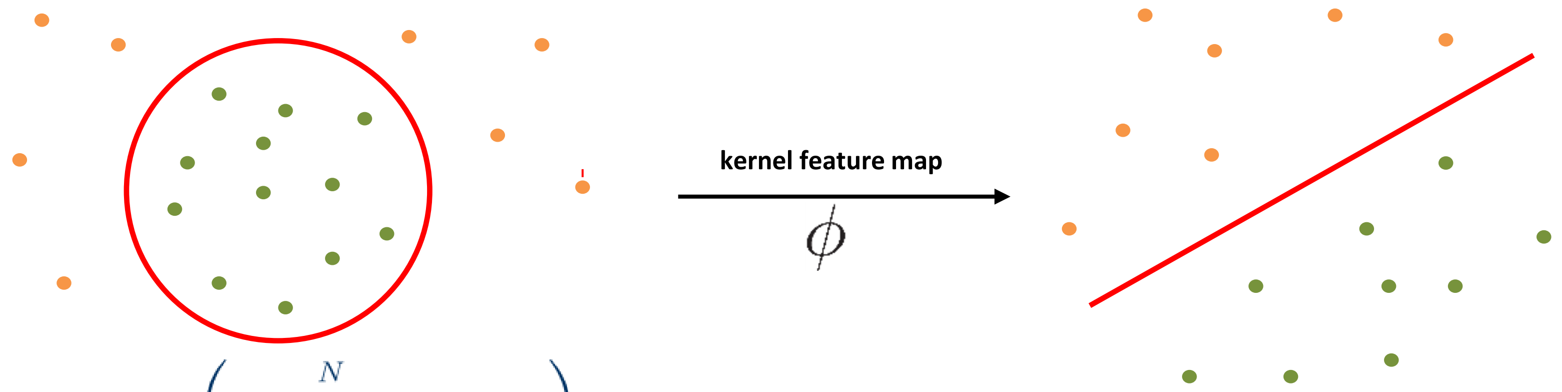
(measures shape)

The Kernel Trick

(classical, Aizerman et al 1964; Vapnik, 1995)

Kernel function is composition of “feature map” with scalar product

$$k(x,y) = \langle \phi(x), \phi(y) \rangle \qquad \phi : \mathbb{R}^n \to \mathcal{F}$$



$$f(x) = \text{sgn}\left(b + \sum_{i=1}^{N} \alpha_i k(x_i, x) \right) \qquad f(x) = \text{sgn}\left(b + \langle w, \phi(x) \rangle \right)$$

Trick = linearization of algorithms operating with scalar products

“kernelization”

Example: $k(x,y) = \langle x, y \rangle^2$ $\qquad \phi : (x_1, x_2) \mapsto \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2 \right)$

“homogenous polynomial kernel” “Veronese map”

Kernelization

Kernels allow to make linear algorithms work on non-linear features

Idea: replace scalar product in algorithm by "kernel product"

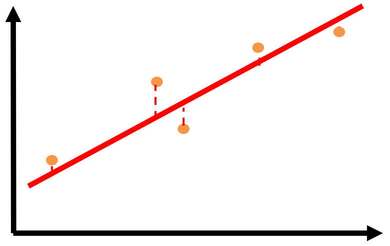
Example: (ridge) regression

Input: $x_1, \ldots, x_N \in \mathbb{R}^n, y_1, \ldots, y_N \in \mathbb{R}$

Output: regressor function

$$f(x) = y^\top \cdot X \left(\lambda I + X^\top X\right)^{-1} \cdot x$$

$$X = (x_{i,j})_{ij} \in \mathbb{R}^{N \times n}$$



"Kernelized" variant:

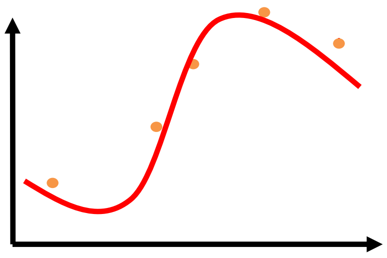
$$f(x) = y^\top \cdot (\lambda I + K)^{-1} \cdot \kappa(x)$$

$$K = (k(x_i, x_j))_{ij}$$

"kernel matrix"

$$\kappa(x) = (k(x_i, x))_i$$

"kernel evaluation vector"



Kernelization

Example: support vector machine

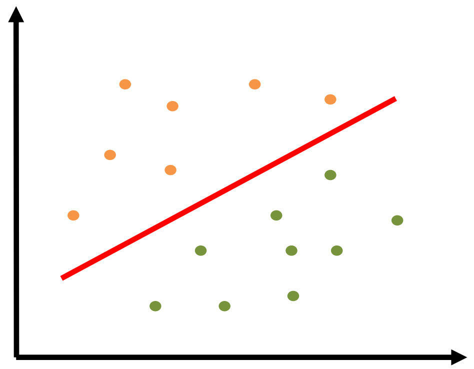
Input: $x_1, \ldots, x_N \in \mathbb{R}^n, y_1, \ldots, y_N \in \{-1, +1\}$

Output: separator/decision function

$$f(x) = \text{sgn}\left(b + w^\top x\right)$$

$w = \sum_{i=1}^N \alpha_i y_i x_i$ solves a QP involving $X^\top X$

$\alpha_i, 1 \leq i \leq N$ are "dual" variables

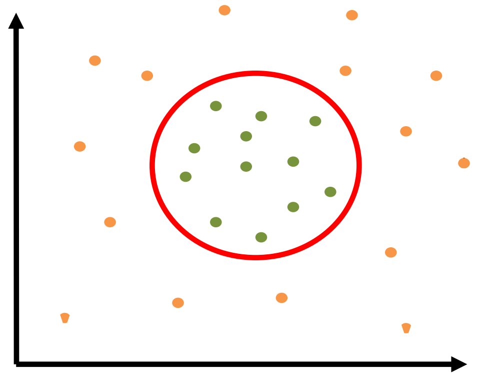


"Kernelized", non-linear variant:

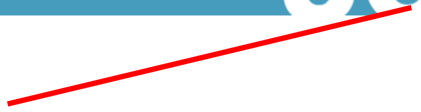
$$f(x) = \text{sgn}\left(b + \sum_{i=1}^N \alpha_i k(x_i, x)\right) = \alpha^\top \kappa(x)$$

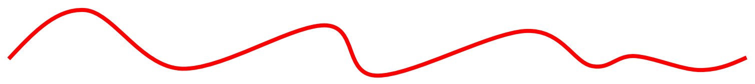
α solves a QP involving kernel matrix

$$K = (k(x_i, x_j))_{ij}$$



The kernel trick

Most scalable algorithms employ linear principles 

Most data exhibit non-linear features 

Naïve idea to make non-linear algorithms:

Suppose we are given data points $x_1, \dots, x_N \in \mathbb{R}^n$

Apply a non-linear "feature map" $\phi : \mathbb{R}^n \to \mathbb{R}^m$ to all data points

Run your favourite *linear* algorithm on $\phi(x_1), \dots, \phi(x_N)$.

This idea (with a bit of care) works, and is at the basis of a considerable part of modern statistics!

E.g., non-linear Tikhonov-regularized least-squares regression:

$$f(x) = y^\top \cdot X \cdot \left(\lambda I + X^\top X\right)^{-1} \cdot x \qquad X = \left(\phi\left(x_i\right)_j\right)_{ij} \in \mathbb{R}^{N \times m}$$

But there are issues with this approach:

Even for rather simple feature maps, m becomes quickly large.

In polynomial regression of degree d, one has $m = \Theta(n^d)$.

This causes: bad scaling/high runtime overfitting (even with regularization)

badly conditioned matrices

Also, the right choice of φ remains an issue.

The kernel trick *(classical, Aizerman et al 1964; Vapnik, 1995)*

uses feature map $\phi : \mathbb{R}^n \to \mathbb{R}^m$

Observation 1 (easy):

If all the algorithm needs is scalar products of the form $\langle \phi(x_i), \phi(x_j) \rangle$

Write $k(x,y) := \langle \phi(x), \phi(y) \rangle$

This $k(x,y)$ may be much easier to compute than $\phi(x)$

Example: $k(x,y) = \langle x, y \rangle^2$ $\qquad \phi : x \mapsto \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2, \sqrt{2} x_1 x_3, \ldots \right)$

"homogenous polynomial kernel" "Veronese map"

costs $O(n)$

$\phi(x)$ costs $O(n^2)$

$\langle \phi(x), \phi(y) \rangle$ costs $O(n^2)$

Observation 2 (difficult):

Many classical algorithms may be recast in scalar products only

Example 1: principal components

eig $\left(X^\top X \right)$

(plus centering)

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} \in \mathbb{R}^{N \times n}$$

Example 2: ridge regression

$$f(x) = y^\top \cdot X \cdot \left(\lambda I + X^\top X \right)^{-1} \cdot x$$

Bad news:

entries of $X^\top X$ are not inner products

Good news:

they are very closely related

Theorem (Singular Value Decomposition):

Every real matrix $M \in \mathbb{R}^{N\times n}$ admits a decomposition

$M = U \cdot S \cdot V^\top$ with $U \in \mathbb{R}^{N\times N}$ orthogonal

$V \in \mathbb{R}^{n\times n}$ orthogonal

$S \in \mathbb{R}^{N\times n}$ diagonal

$S =$ diag$(\sigma_1, \ldots, \sigma_{\min(n,N)})$

σ_i are non-negative

(and usually ordered descendingly)

which is unique up to

orthogonal action on row-/column spaces

(if all σ_i are distinct, flipping sign in i-th column of U and V)

animation from Wikimedia Commons

columns of U, V: "left/right singular vectors"

σ_i : "singular values"

Corollary (by "uniqueness")**:**

eigenvectors of $M^\top M$ = right singular vectors of M

left singular vectors of M = eigenvectors of $MM^\top$

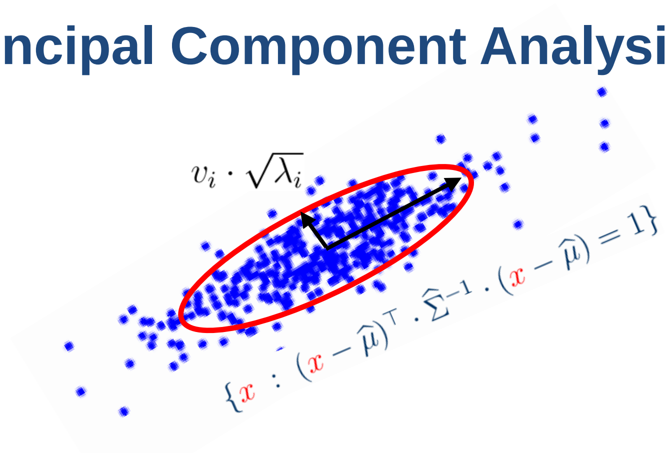
eigenvalues of $M^\top M$ = (singular values)^2 of M = eigenvalues of $MM^\top$

Inner product matrix!

This flavour of duality is at the core of ``kernelization".

Kernelizing principal component analysis

Principal Component Analysis: for data $x_1, \dots, x_N$



1. compute: sample mean $\widehat{\mu} = \frac{1}{N}\sum_{i=1}^N x_i$

sample covariance matrix $\widehat{\Sigma} = \frac{1}{N}\sum_{i=1}^N (x_i - \widehat{\mu})(x_i - \widehat{\mu})^\top$

2. eig$(\widehat{\Sigma}_N)$ yields eigenpairs $(\lambda_1, v_1), \dots, (\lambda_n, v_n)$

λ_i: "i-th principal value" v_i: "i-th principal vector"

"Principal scores": $\tau_i(x) = v_i^\top (x - \widehat{\mu})/\sqrt{\lambda_i}$

Kernel Principal Component Analysis:

Observe: $\widehat{\Sigma} = X_\mu^\top X_\mu$ where $X_\mu = (I - \mathbb{1}_N)\, X$ and $X = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} \in \mathbb{R}^{N\times n}$

$\mathbb{1}_N$ the $(N \times N)$ matrix with entries $\frac{1}{N}$

By SVD theorem, λ_i are the eigenvalues of

$$K_\mu := X_\mu X_\mu^\top = (I - \mathbb{1}_N)\cdot XX^\top \cdot (I - \mathbb{1}_N) = (I - \mathbb{1}_N)\cdot K \cdot (I - \mathbb{1}_N)$$

where $K = (\langle x_i, x_j\rangle)_{ij}$ is the Gram matrix of the data.

easy kernelization via "kernel matrix" $K = (k(x_i, x_j))_{ij}$

By SVD theorem, v_i are right singular vectors of X_μ

So $\tau_i(x) = u_i^\top \left(X_\mu x - X_\mu X^\top \cdot \mathbb{1}/N\right)/\lambda_i$ where u_i is the corresponding left singular vector

$= u_i^\top \left(\kappa(x) + (\mathbb{1}_N - I)\cdot K \cdot \mathbb{1}/N\right)/\lambda_i$ or equivalently, the corresponding eigenvector of K_μ

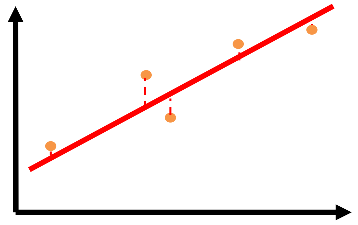
and $\mathbb{1}$ is a vector of ones

where $\kappa(x) = (\langle x, x_i\rangle)$ is a cross-Gram vector

easy kernelization via "empirical kernel vector" $\kappa(x) = (k(x, x_i))_i$

Kernelizing ridge regression

Ridge regression = Tikhonov-regularized OLS:



for data $x_1, \ldots, x_n$ $\quad X = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} \in \mathbb{R}^{N\times n}$

compute: $\widehat{\beta} = \left(\gamma I + X^\top X\right)^{-1} X^\top \cdot y$

minimizes $R(\beta) = \|y - X\beta\|_F^2 + \gamma\|\beta\|_2^2$
($\gamma \in \mathbb{R}_{>0}$ regularization parameter)

Prediction: $f(x) = \widehat{\beta}^\top x$

Kernel ridge regression:

Note: If (λ, v) is eigenpair of $A \in \mathbb{R}^{n\times n}$ then $(\lambda + \gamma, v)$ is eigenpair of $A + \gamma I$

Consider the singular value decomposition $USV^\top = X$ with $S = \text{diag}(\sigma_1, \sigma_2, \ldots)$

with this, $(\gamma I + X^\top X)^{-1} X^\top = V \cdot S_\gamma \cdot U^\top$

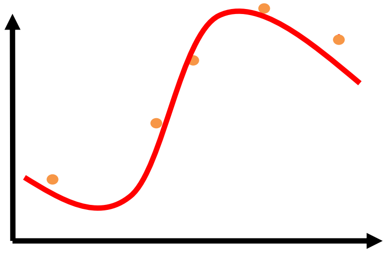
where S_γ is diagonal with entries $\dfrac{\sigma_i}{\gamma + \sigma_i^2}$

By symmetry of the expression, $\left(\gamma I + XX^\top\right)^{-1} X = U \cdot S_\gamma \cdot V^\top$

So $\widehat{\beta} = X^\top \left(\gamma I + XX^\top\right)^{-1} \cdot y \; = X^\top \left(\gamma I + K\right)^{-1} \cdot y$

$f(x) = x^\top \cdot X^\top \cdot (\gamma I + K)^{-1} \cdot y = \kappa(x)^\top \cdot (\gamma I + K)^{-1} \cdot y$

kernelization via $K = (k(x_i, x_j))_{ij}$ and $\kappa(x) = (k(x, x_i))_i$



As usual, $\widehat{\beta} \to \widehat{\beta}_{OLS}$ when $\gamma \to 0$ (Theorem)

if $m \geq N$, the limit $\gamma \to 0$ fits an *exact* regressor, i.e., $f(x_i) = y_i$

The
Reproducing kernel Hilbert space formalism

The kernel trick in RKHS-generality

Previously: feature map $\phi : \mathbb{R}^n \to \mathbb{R}^m$ $\qquad k(x,y) = \langle \phi(x), \phi(y) \rangle$

positive definite kernel matrix $K = (k(x_i, x_j))_{ij}$ $\qquad$ e.g. $k(x,y) = \langle x, y \rangle^d$

Observation 3: existence of ϕ has not been used in kernelization

All that is required is a well-behaved k that yields K

For example, $k(x,y) = \exp\left(-\frac{1}{2\sigma^2}\|x-y\|^2\right)$ always gives positive semi-definite K and works very nicely for clustering.

"Gauss kernel"

but there is no feature map $\phi : \mathbb{R}^n \to \mathbb{R}^m$!

Definition: A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (usually $\mathcal{X} \subseteq \mathbb{R}^n$)

is called *positive definite kernel* if

(often required for technical reasons: $k \in L_2(\mathcal{X}^2)$ or $k \in L_\infty(\mathcal{X}^2)$)

every possible kernel matrix is symmetric and positive semi-definite.

(in k and with data in $\mathcal{X}$) $\qquad$ (sic)

Theorem (Moore-Aronszajn, 1950):

For every positive definite kernel k, there is (up to isomorphism)

a unique feature map $\phi : \mathcal{X} \to \mathcal{H}$

into a unique Hilbert space $\mathcal{H}$, the so-called RKHS. (one way to imagine $\mathcal{H}$ is $\mathbb{R}^{\mathbb{N}}$)

More precisely, $\phi(x) = (z \mapsto k(x,z))$ $\quad$ and $\langle k(x,.), f(.)\rangle_{\mathcal{H}} = f(x). \forall f \in \mathcal{H}.$

"reproducing (kernel Hilbert space) property"

A (non-exhaustive) list of popular kernels

$$k(x,y) = x^\top A y$$
linear kernels

$$k(x,y) = \rho\left(\|x-y\|\right)$$
Radial basis function kernels

$$k(x,y) = f\left(\langle x,y\rangle\right)$$
Dot-product kernels

$$k(x,y) = \langle x,y\rangle^d$$
homogenous polynomial kernel

$$k(x,y) = \left(\theta\langle x,y\rangle + 1\right)^d$$
inhomogenous polynomial kernel

(both polynomial kernels measure “shape”)

$$k(x,y) = \tanh\left(\sigma\langle x,y\rangle - \vartheta\right)$$
sigmoid kernel (measures “contours”)

not positive definite

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

$$k(x,y) = \exp\left(-\frac{\|x-y\|}{\sigma}\right)$$
Laplacian kernel

$$k(x,y) = \left(1 + \frac{\|x-y\|^2}{\sigma^2}\right)^{-1}$$
Cauchy kernel

(these three measure “soft closeness”, with increasing long-range interaction/heavy tails from left to right)

Proposition:

For positive definite kernels $k_1, k_2, \ldots$, and $\alpha_i \in \mathbb{R}_{\geq 0}$,

a k defined by $k(x,y) =$ as any below is a positive definite kernel:

$$k_3(x,y) + \alpha_{42} \qquad \alpha_1 \cdot k_1(x,y) + \alpha_2 \cdot k_2(x,y) \qquad k_1(x,y) \cdot k_2(x,y)$$

$$\sum_{\nu=1}^{\infty} \alpha_\nu \langle x,y\rangle^\nu$$
(in case of convergence)

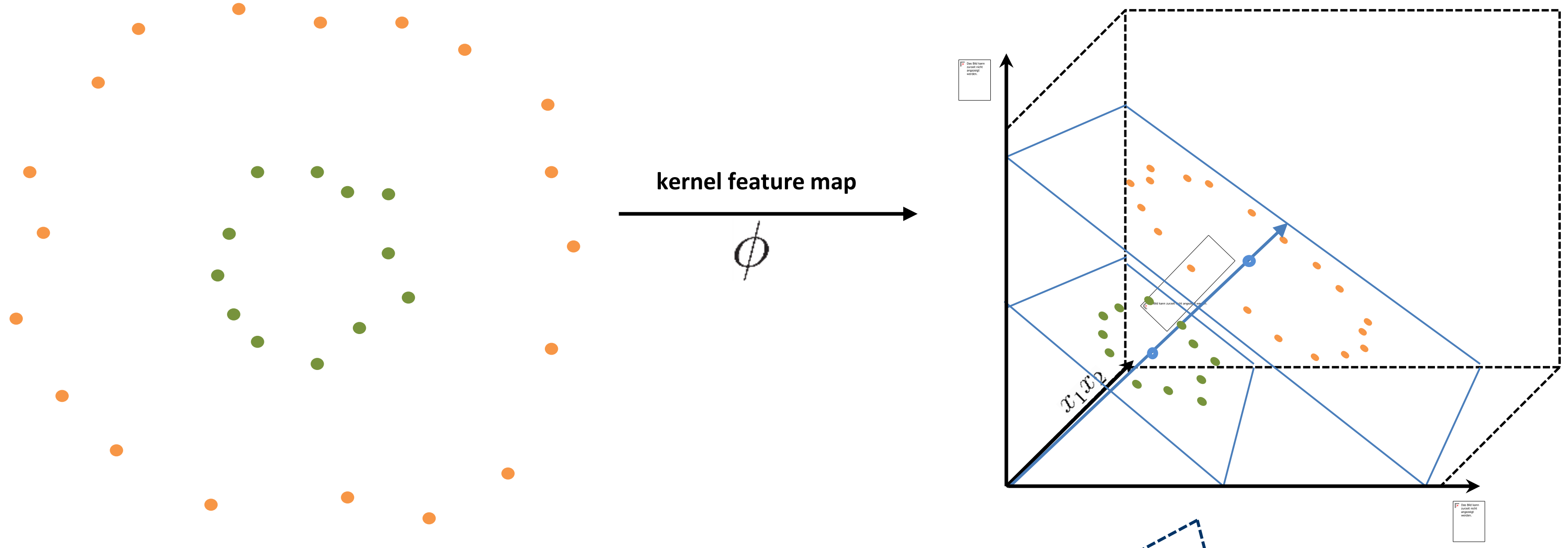
$$\lim_{\nu\to\infty} k_\nu(x,y)$$
(in case of existence)

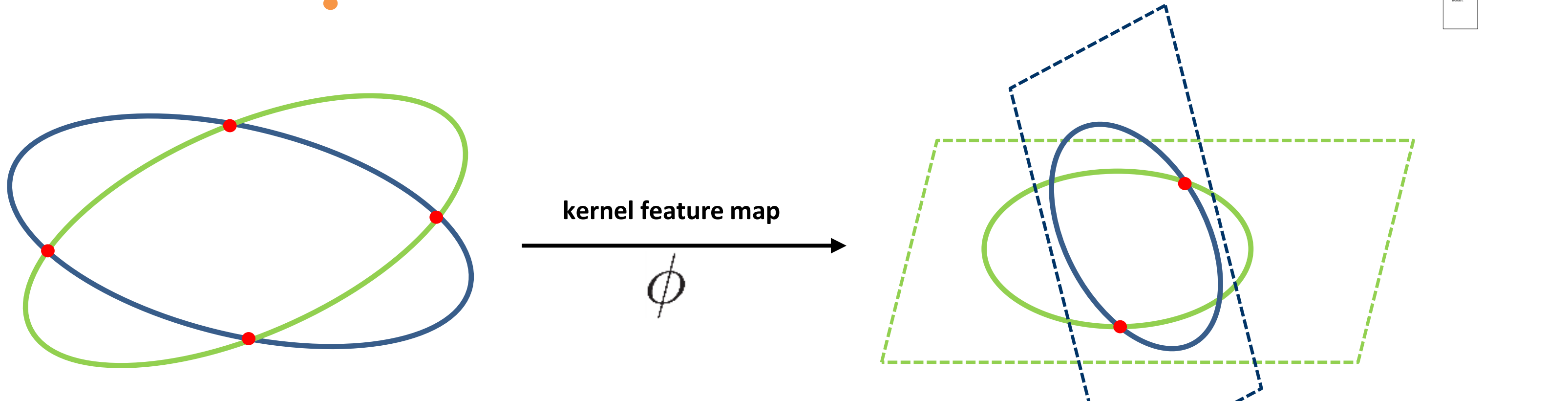
$$\frac{k_1(x,y)}{\sqrt{k_1(x,x)\cdot k_1(y,y)}}$$

Example: polynomial feature map

$$k(x,y) = \langle x, y\rangle^2 \qquad \phi : (x_1, x_2) \mapsto \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

$$k(x,y) = \langle \phi(x), \phi(y)\rangle \qquad \phi : \mathbb{R}^n \to \mathcal{F}$$





Empirical risk minimisation and learning bounds

Recall: ridge regression finds a regressor function $f(x) = \widehat{\beta}^\top x$

where $\widehat{\beta}$ minimizes $R(\beta) = \underbrace{\|y - X \cdot \beta\|_F^2}_{\text{“empirical risk”}} + \underbrace{\gamma\|\beta\|_2^2}_{\text{“regularizer”}}$

Similarly, kernel ridge regression finds a regressor $f(x) = \widehat{\alpha}^\top \kappa(x)$

where $\widehat{\alpha}$ minimizes $R(\alpha) = \|y - K \cdot \alpha\|_{F,\mathcal{H}}^2 + \gamma \cdot \alpha^\top K \alpha$

One can show that we cannot do better under these circumstances:

Theorem (Kimeldorf-Wahba, 1971) “Representer Theorem”:

Let $(x_1, y_1), \dots, (x_N, y_N) \in \mathbb{R}^n \times \mathbb{R}$ be data points.

Let k be a positive definite kernel with RKHS $\mathcal{H}$.

Let $L : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \to \mathbb{R} \cup \{\infty\}$ be any loss function,

for example squared loss $L(x_i, y_i, f(x_i)) = (y_i - f(x_i))^2$

Let $\Omega : [0, \infty) \to \mathbb{R}$ strictly monotonously increasing. for example $\Omega(x) = \gamma \cdot x^2$

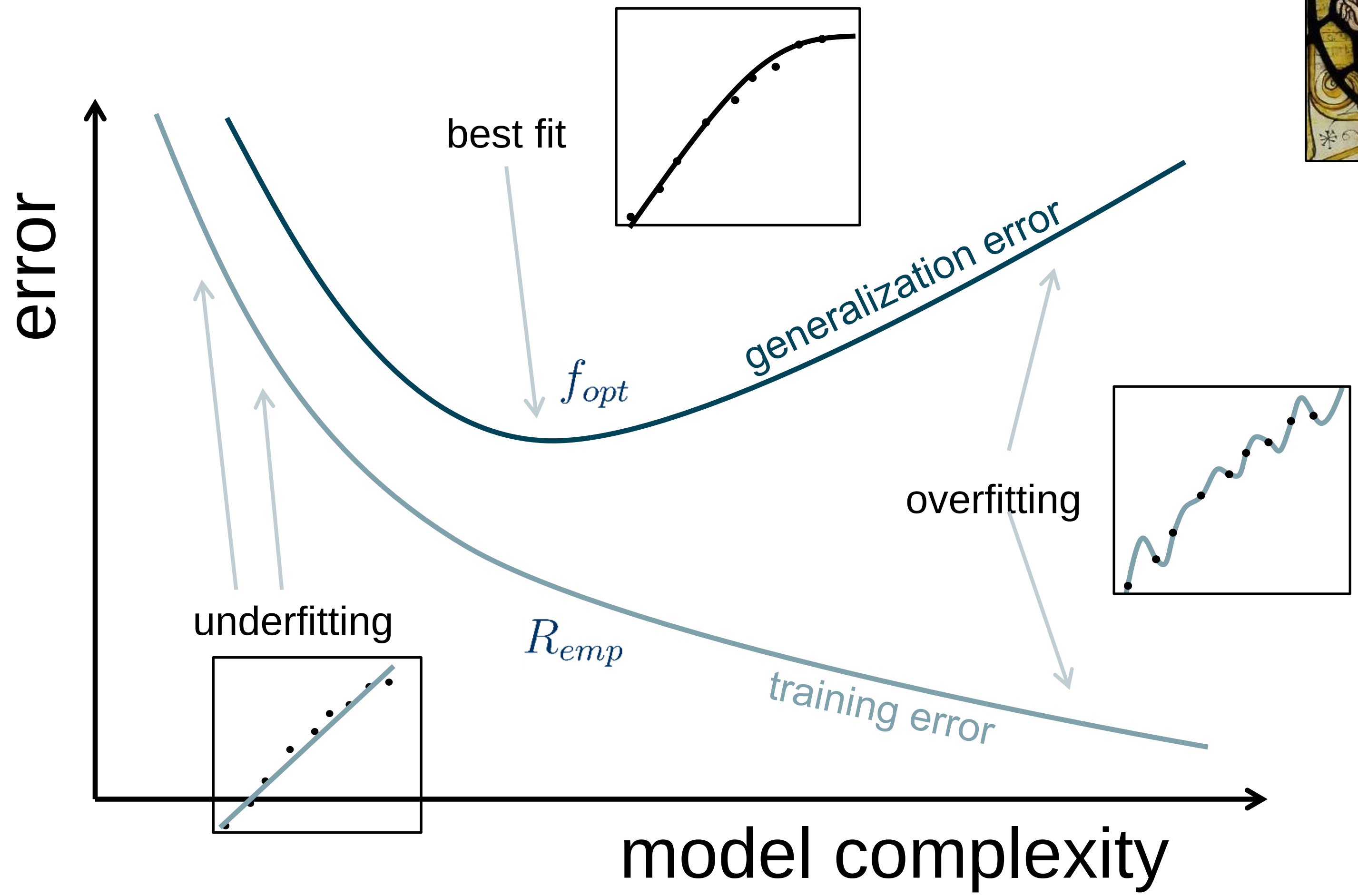
Then, there is a minimizer to $R(f) = \sum_{i=1}^{N} L(x_i, y_i, f(x_i)) + \Omega(\|f\|_\mathcal{H})$

of the type $f(x) = \alpha^\top \kappa(x)$.

Statistical Learning Theory (Vapnik-Chervonenkis) quantifies

when and how $R_{emp}(f_{opt}) \to R(f_{opt})$ for $N \to \infty$.

VC = quantiative form of Occam‘s razor



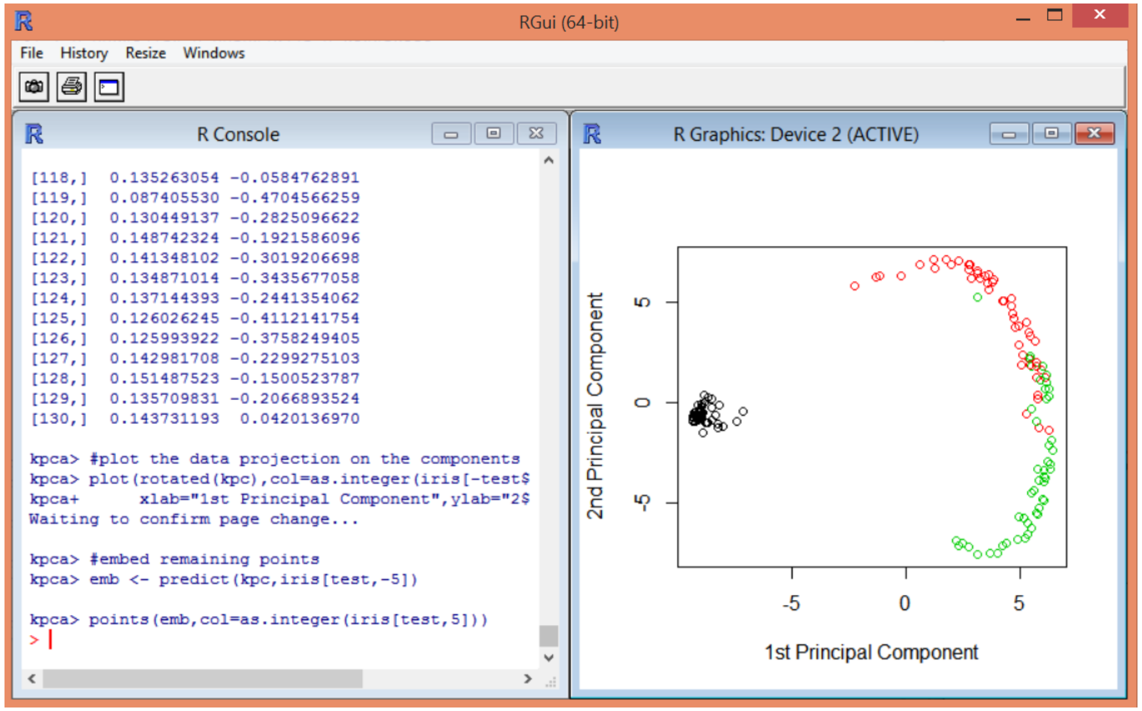
kernlab

Working with kernlab under R

If you have not worked with R before or not much:

look at STAT7001 materials or e.g. Dalgaard, introductory statistics with R

or type `help.start()` and click „an introduction to R“



```
[118,]  0.135263054 -0.0584762891
[119,]  0.087405530 -0.4704566259
[120,]  0.130449137 -0.2825096622
[121,]  0.148742324 -0.1921586096
[122,]  0.141348102 -0.3019206698
[123,]  0.134871014 -0.3435677058
[124,]  0.137144393 -0.2441354062
[125,]  0.126026245 -0.4112141754
[126,]  0.125993922 -0.3758249405
[127,]  0.142981708 -0.2299275103
[128,]  0.151487523 -0.1500523787
[129,]  0.135709831 -0.2066893524
[130,]  0.143731193  0.0420136970

kpca> #plot the data projection on the components
kpca> plot(rotated(kpc),col=as.integer(iris[-test$
kpca+      xlab="1st Principal Component",ylab="2$
Waiting to confirm page change...

kpca> #embed remaining points
kpca> emb <- predict(kpc,iris[test,-5])

kpca> points(emb,col=as.integer(iris[test,5]))
> |
```

type `install.packages(kernlab)` to install the kernlab package

After that `library(kernlab)` or `require(kernlab)` to use functions

Documentation: http://cran.r-project.org/web/packages/kernlab/kernlab.pdf

Try `help(functionname)` or `example(fuctionname)`

Outlook

Lecture 1: Introduction to kernels

- Main concepts and theoretical results, learning guarantees
- Kernel PCA and kernel ridge regression
- Some notes on R and kernlab

Lecture 2: the kernel support vector machine

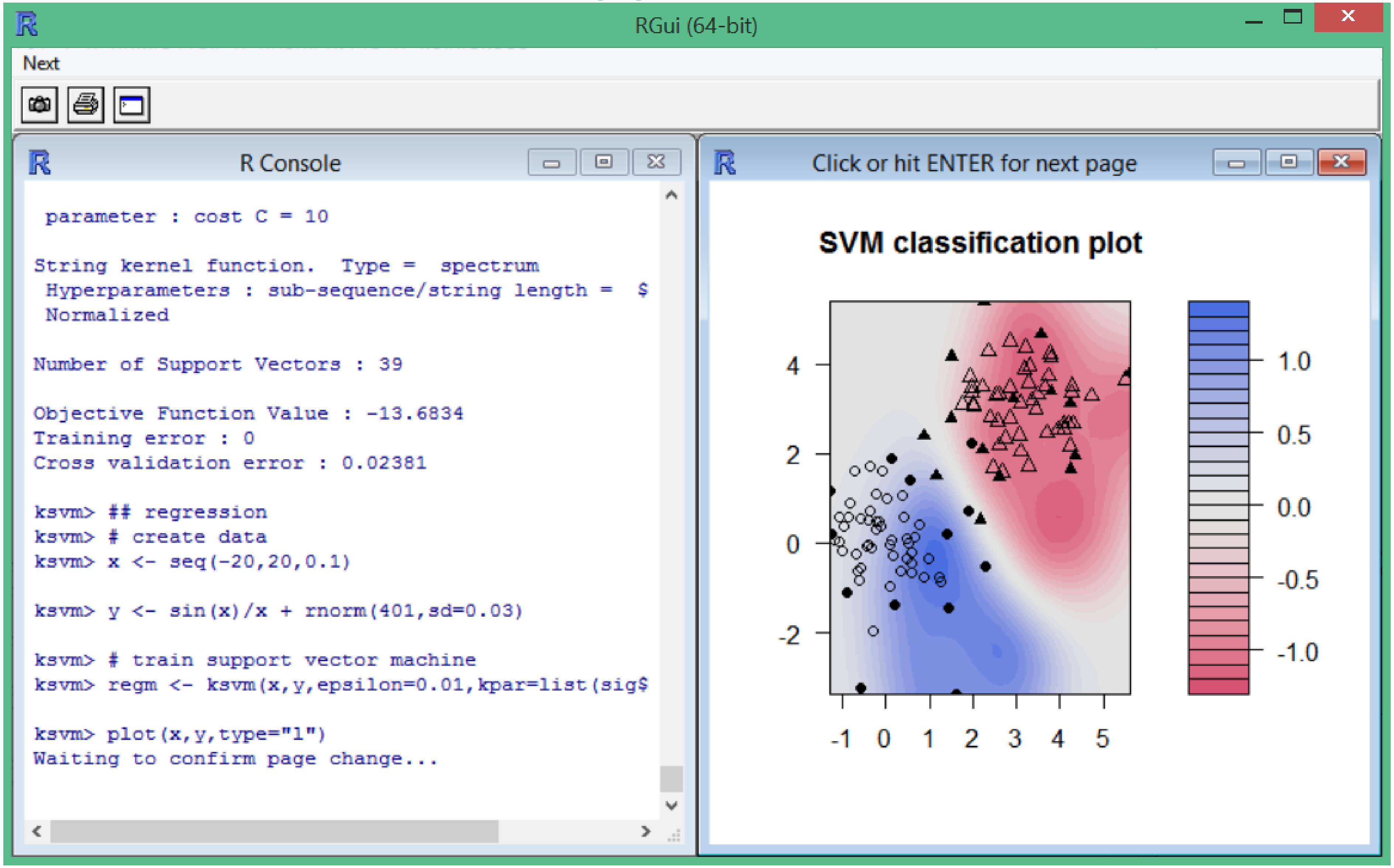
- The linear support vector machine, duality
- Hard- and soft-margin two-class SVM
- The one-class SVM
- Support vector regression

Lecture 3: Gaussian processes and kernel learning

Potential further lecture topics:

- Algorithms: kernel discriminants, kernel k-means, kernel quantile regression
  kernel CCA, kernel MMD, kernel relevance vector machine
- Large-scale learning with kernels, subset methods and Nyström-approximation
- Combinatorial kernels: string kernels, graph kernels
- Invariance kernels
- Vapnik-Chervonenkis learning theory
- Outlier detection, novelty detection
- On-line kernel learning

Next week: the Support Vector Machine



```
 parameter : cost C = 10

String kernel function.  Type =  spectrum
 Hyperparameters : sub-sequence/string length =  $
 Normalized

Number of Support Vectors : 39

Objective Function Value : -13.6834
Training error : 0
Cross validation error : 0.02381

ksvm> ## regression
ksvm> # create data
ksvm> x <- seq(-20,20,0.1)

ksvm> y <- sin(x)/x + rnorm(401,sd=0.03)

ksvm> # train support vector machine
ksvm> regm <- ksvm(x,y,epsilon=0.01,kpar=list(sig$

ksvm> plot(x,y,type="l")
Waiting to confirm page change...
```