CHALMERS
UNIVERSITY OF TECHNOLOGY

Statistical methods in Data Science and AI

Marina Axelson-Fisk

30 september, 2019

Inference in graphical models

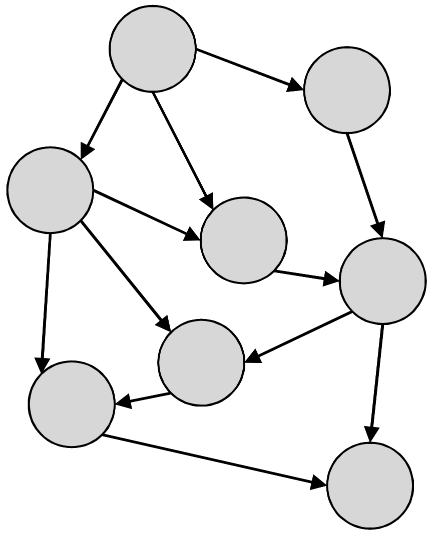
Given a graphical model, we want to answer questions of interest.

- ***Marginal inference*: what is the marginal probability of a given variable Y in our graph, summing out the rest?**

$$\boldsymbol{P}(\boldsymbol{Y}=\boldsymbol{y}) = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_n} \boldsymbol{P}(\boldsymbol{Y}=\boldsymbol{y}, \boldsymbol{X_1}=\boldsymbol{x_1}, \boldsymbol{X_2}=\boldsymbol{x_2}, \dots, \boldsymbol{X_n}=\boldsymbol{x_n})$$

- ***Maximum a posteriori (MAP) inference*: what is the most likely assignment to the variables in the graph (possibly conditioned on data)?**

$$\max_{x_1,\dots,x_n} \boldsymbol{P}(\boldsymbol{Y}=\boldsymbol{y}, \boldsymbol{X_1}=\boldsymbol{x_1}, \dots, \boldsymbol{X_n}=\boldsymbol{x_n})$$



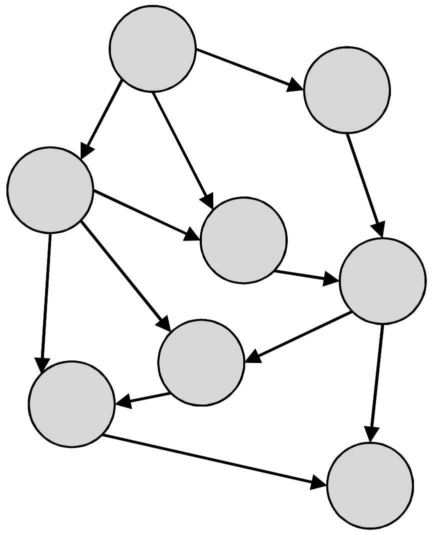
Inference algorithms in graphical models

Exact inference

- Variable elimination
- Message passing/belief propagation
- Junction trees

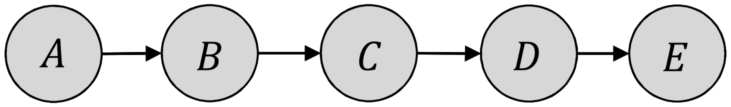
Approximative inference

- Stochastic simulation
- Markov chain Monte Carlo (MCMC)
- Variational algorithms



Example: variable elimination in a chain graph

Random variables: A, B, C, D, E



each taking n possible values $\{1, 2, \dots, n\}$, then the *marginal probability* of E

$$P(E=e) = \sum_{a=1}^{n}\sum_{b=1}^{n}\sum_{b=1}^{n}\sum_{b=1}^{n} P(A=a, B=b, C=c, D=d, E=e) = \sum_{a,b,c,d} P(a,b,c,d)$$

we can utilize the chain structure to reduce the number of operations by *variable elimination*.

Example: variable elimination in a chain graph

Exploit the structure ”inside-out” or from ”leaf-to-top”

$$P(E=e)=\sum_a\sum_b\sum_c\sum_d P(A=a,B=b,C=c,D=d,E=e)$$

$$=\sum_a\sum_b\sum_c\sum_d P(a)P(b|a)P(c|b)P(d|c)P(e|d)$$

$$=\sum_b\sum_c\sum_d P(c|b)P(d|c)P(e|d)\sum_a P(b|a)P(a)$$

$$=\sum_b\sum_c\sum_d P(c|b)P(d|c)P(e|d)\,P(b)=\cdots=$$

$$=\sum_d P(e|d)P(d)=P(e)$$

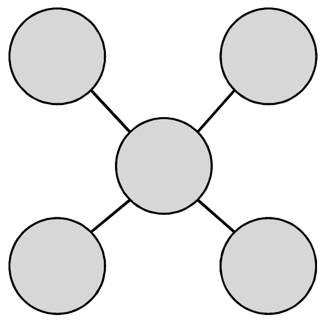
Message passing/belief propagation

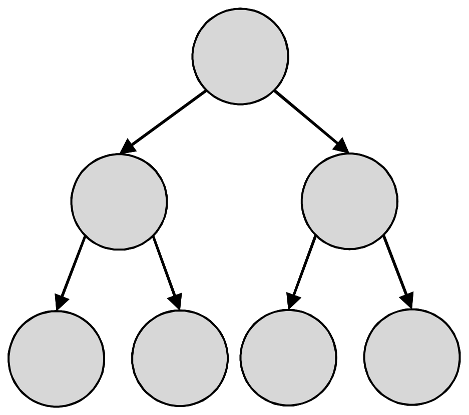
- Variable elimination can be seen as ”passing a message” (information), or ”propagating a belief” from one node to the next.
- This is the basic framework for computing various entities in Hidden Markov Models (HMMs) and Linear dynamical systems (LDS)
- In continuous distributions $p(x|\theta)$ message passing corresponds to passing on parameter values θ between neighboring nodes.

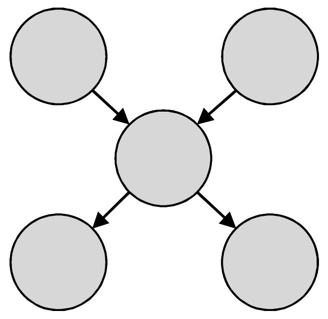
CHALMERS
UNIVERSITY OF TECHNOLOGY

Inference on trees

- **Each node sends out the product of the messages received from the parents to the children**
- **I.e. message passing is an abstract notion of conditional (in)dependence**

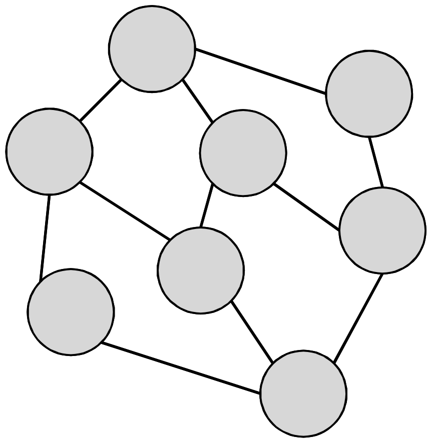

undirected tree


directed tree


polytree

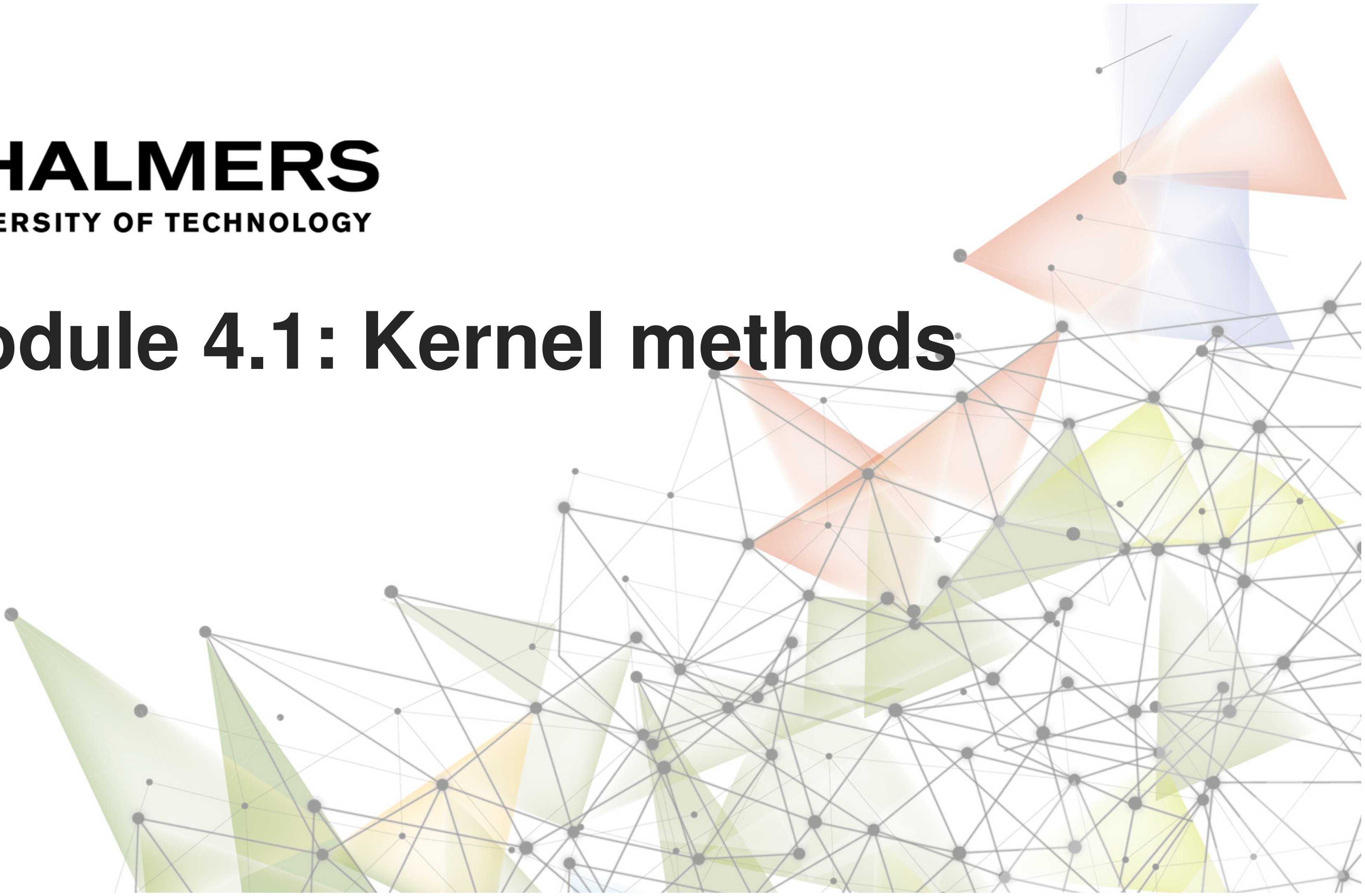
Inference on trees

- The ***junction tree algorithm*** **(or *clique tree algorithm*) is a generalization of message passing to arbitrary graphs**



CHALMERS
UNIVERSITY OF TECHNOLOGY

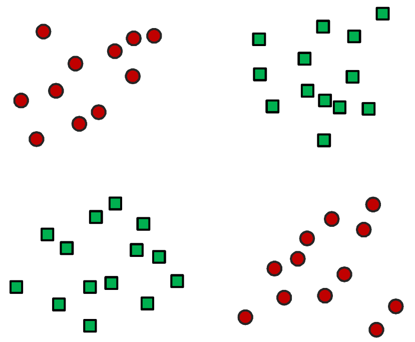
Module 4.1: Kernel methods



CHALMERS
UNIVERSITY OF TECHNOLOGY

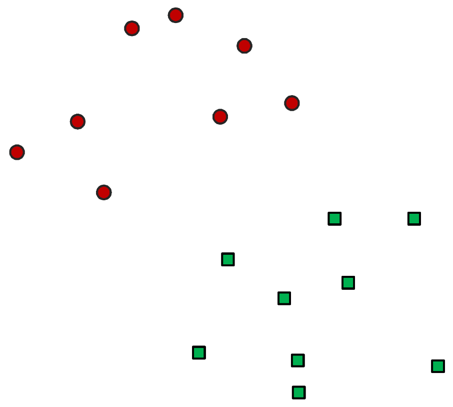
Kernel methods: motivation

- Given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
  - y_i response
  - $\mathbf{x}_i$ feature vector
- there are numerous tools for detecting linear relations
  - Ridge regression
  - Support vector machines (SVMs)
  - Principal component analysis (PCA)
- But what if the relationship is nonlinear?



Motivating example: binary classification

- **Training set:** $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, $y_i \in \{-1, +1\}$
- **Objective: learn a function** $f(\mathbf{x})$ **such that** $y_i = \text{sign}\big(f(\mathbf{x}_i)\big)$



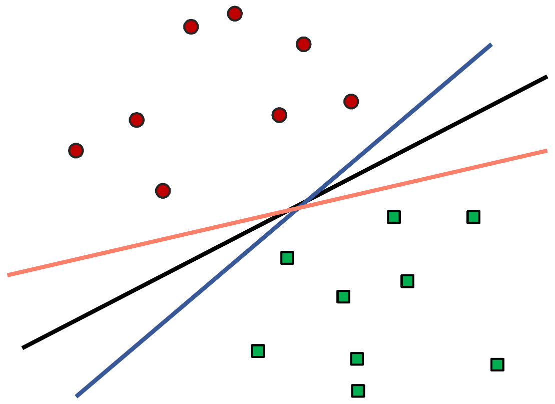
Linear classification

When classes are linearly separable, the boundary is a *hyperplane*.

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$$

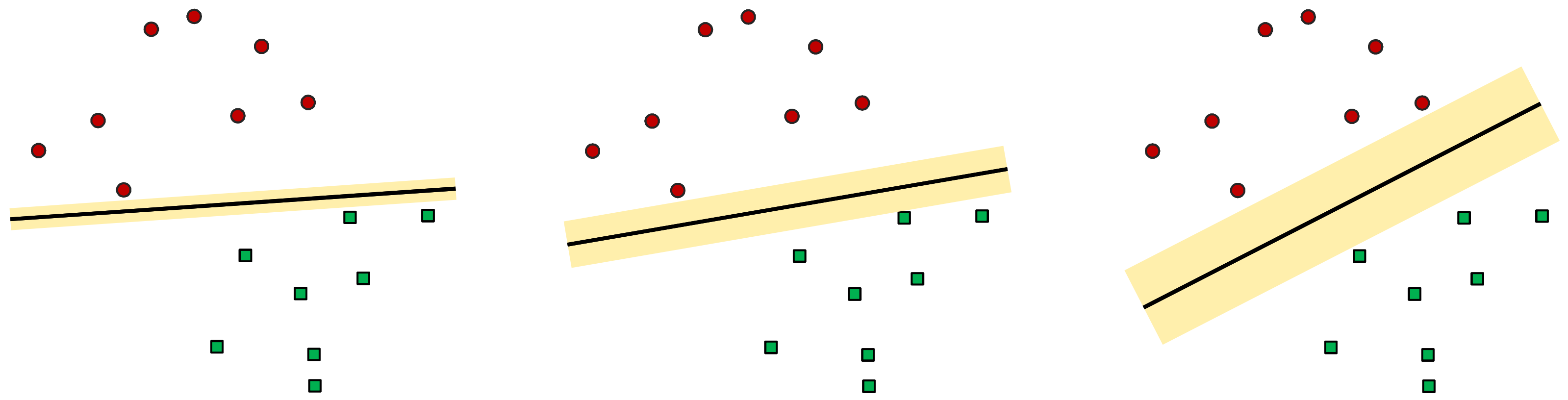
If $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b \begin{cases} > 0 & y = \text{red} \\ < 0 & y = \text{green} \end{cases}$

But which line should we choose?



Linear classification

- How large margins do we have between the classes?
- How do we maximize that margin?



How do we maximize the margin?

- Let $\mathbf{x}_n \in \mathcal{D}$ be the point closest to the hyperplane $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$
- Normalize $\mathbf{w}$:

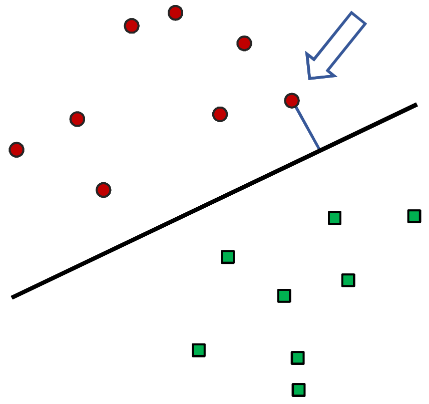
$$\left|\mathbf{w}^{\mathrm{T}}\mathbf{x}_n\right| = 1$$

$\Rightarrow$ The canonical hyperplane

- The distance between $\mathbf{x}_n$ and the plane

$$\text{distance} = \frac{1}{\|\mathbf{w}\|}\left|\mathbf{w}^{\mathrm{T}}\mathbf{x}_n - \mathbf{w}^{\mathrm{T}}\mathbf{x}\right| = \frac{1}{\|\mathbf{w}\|}$$

- Maximize this distance.



Support vector machines (SVMs)

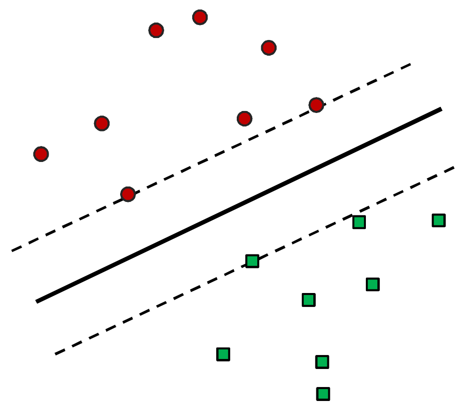
For two linearly separable classes with class labels $y_i \in \{-1, +1\}$

- **construct *two* supporting hyperplanes, one for each class**

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \leq -1 \ \text{ for } y_i = -1$$
$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \geq +1 \ \text{ for } y_i = +1$$

The corresponding supporting hyperplanes are thus

$$H_{-1} = \{\mathbf{x}_i : \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b = -1\,, y_i = -1\}$$
$$H_{+1} = \{\mathbf{x}_i : \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b = +1\,, y_i = +1\}$$



Support vector machines (SVMs)

In two dimensions the separating hyperplanes are lines on the form

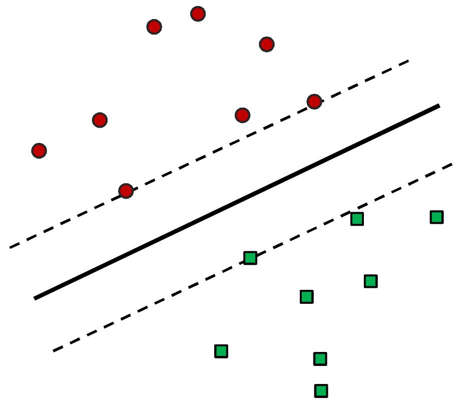
$$f_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b = w_1 x_1 + w_2 x_2 + b = 0$$

Maximizing the margin = minimizing $\|\mathbf{w}\|$ under the constraints

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \leq -1 \text{ for } y_i = -1$$
$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \geq +1 \text{ for } y_i = +1$$

or

$$\mathbf{w}^* = \underset{\mathbf{w},b}{\arg\min} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 \colon y_i \left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \right) \geq 1 \right\}$$



using *Lagrange multipliers*.

Lagrange multipliers

- We want to optimize a function $f(x)$ subject to a constraint $g(x) = 0$.
- We form the ***Lagrangian function***

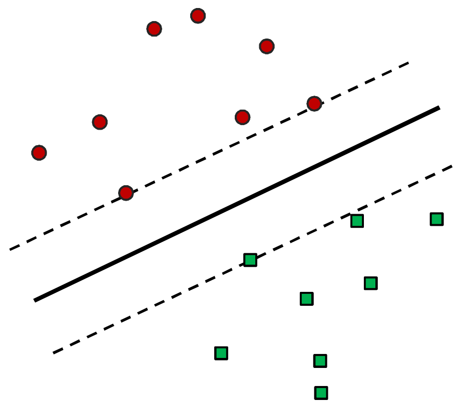
$$L(x, \lambda) = f(x) - \lambda g(x)$$

where λ is a ***Lagrange multiplier***.

- We optimize by computing

$$\frac{dL}{dx} = 0 \;\text{ and }\; \frac{dL}{d\lambda} = 0$$

and solve the corresponding equation system.



Support vector machines (SVMs)

$$\mathbf{w}^* = \arg\min_{\mathbf{w},b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 : y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b\right) \geq 1 \right\}$$

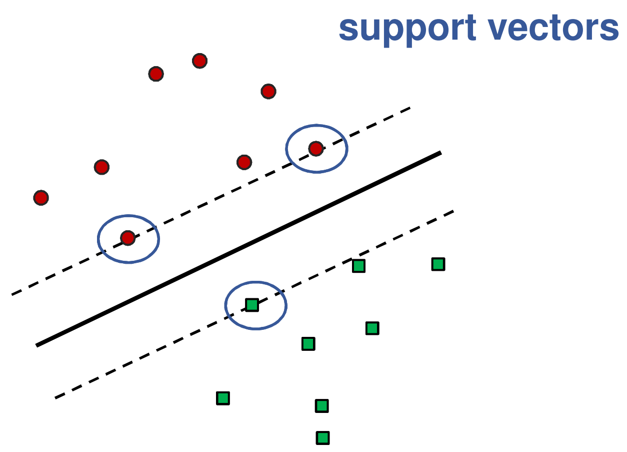
Lagrangian function

$$L(\mathbf{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left(y_i \left(\mathbf{w}^{\mathrm{T}} \mathbf{x}_i - b \right) \right)$$

- $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_N)$: Lagrange multipliers
- N: number of constraints

$$\frac{dL}{dw_i} = 0, i = 1, \dots, N \ \ \textbf{and} \ \ \frac{dL}{db} = 0$$

$$\Rightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \ \ \textbf{and} \ \ \sum_{i=1}^{N} \lambda_i y_i = 0$$



Kernelized SVMs

$$\boxed{\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \quad \textbf{and} \quad \sum_{i=1}^{N} \lambda_i y_i = 0}$$

Dual representation

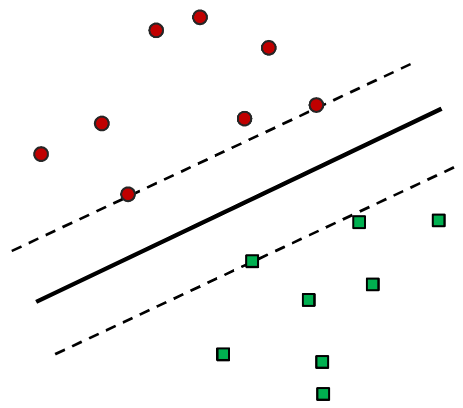
Instead we maximize

$$\tilde{L}(\boldsymbol{\lambda}) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \, k(\mathbf{x}_i, \mathbf{x}_j)$$

under the constraints

$$\lambda_i \geq 0 \;\; \textbf{and} \;\; \sum_{i=1}^{N} \lambda_i y_j = 0$$

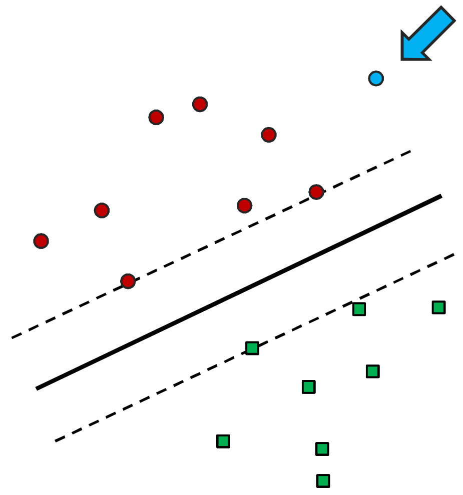
where $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j$ is a kernel function.



Kernelized SVMs

To classify a new data point $\mathbf{x}^*$ we observe the sign of

$$f(\mathbf{x}^*) = \sum_{i=1}^{N} \lambda_i y_i \, k(\mathbf{x}^*, \mathbf{x}_i) + b$$



Soft margin SVMs

When the training set cannot be perfectly separated we introduce *slack variables*

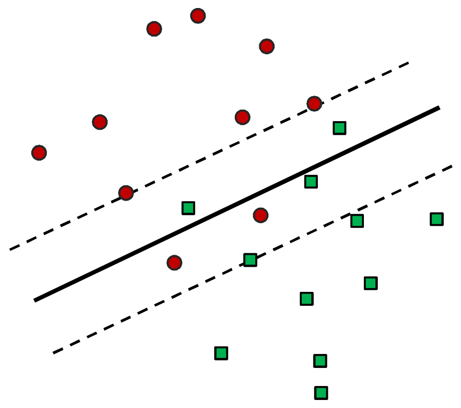
$$y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}+b\right) \geq 1-\xi_i$$

such that $\xi_i \leq 1$ for points on the correct side.

We want to minimize the misclassification rate, i.e. minimize

$$\sum_{i=1}^{N} \mathbb{I}\{\xi_i - 1\} \quad \textbf{where} \quad \mathbb{I}(x) = \begin{cases} 1 & \text{if } x \leq 0 \\ 0 & \text{if } x > 0 \end{cases}$$

NP-complete!



Soft margin SVMs

Use the upper bound

$$\sum_{i=1}^{N} \mathbb{I}\{\xi_i - 1\} \leq \sum_{i=1}^{N} \xi_i$$

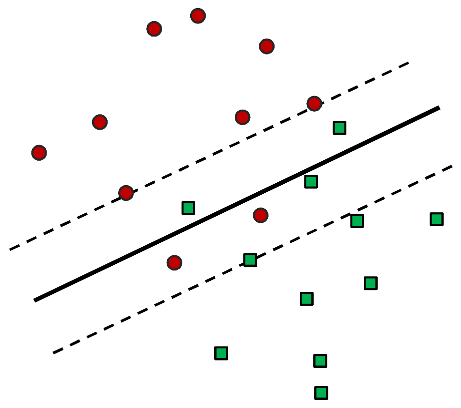
and minimize

$$C \sum_{i=1}^{N} \xi_i + \frac{1}{2}\|\mathbf{w}\|^2, \qquad C > 0$$

under the *soft margin* constraints

$$y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b\right) \geq 1 - \xi_i, \qquad \xi_i \geq 0$$

C is a trade-off between misclassification and complexity



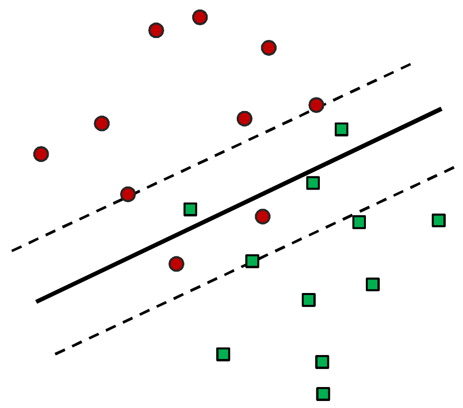
Kernelized soft margin SVMs

Lagrangian function

$$L(\mathbf{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \lambda_i \left(y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - 1 + \xi_i\right)\right) - \sum_{i=1}^{N} \mu_i \xi_i$$

with Lagrange multipliers

$\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N), \boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$



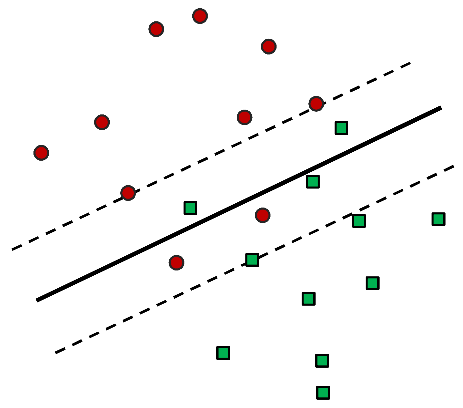
Kernelized soft margin SVMs

The dual form is the same as before

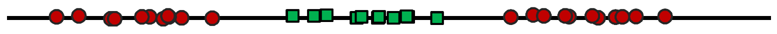
$$\tilde{L}(\boldsymbol{\lambda}) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \, k(\mathbf{x}_i, \mathbf{x}_j)$$

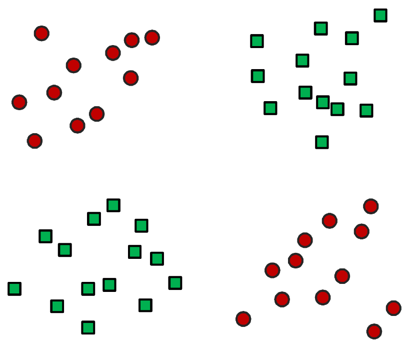
but the maximization constraints become limited by C

$$0 \le \lambda_i \le C \quad \textbf{and} \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$



Nonlinear classification

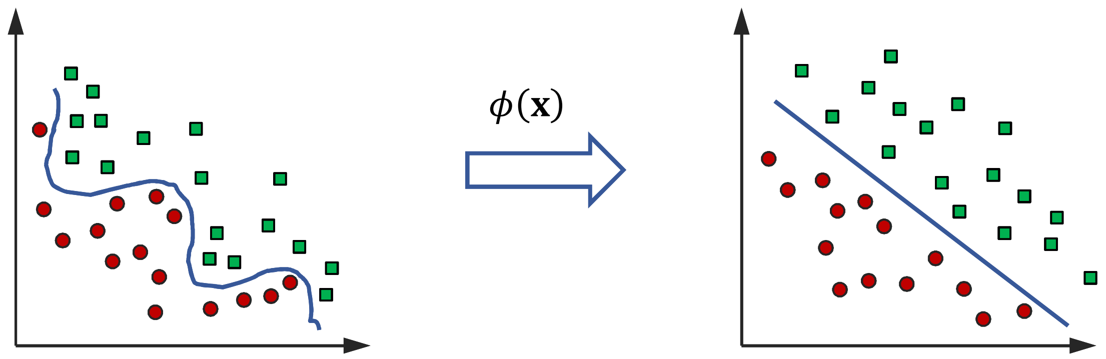




- There is no linear classifier that can separate red from green.

Kernel methods: motivation

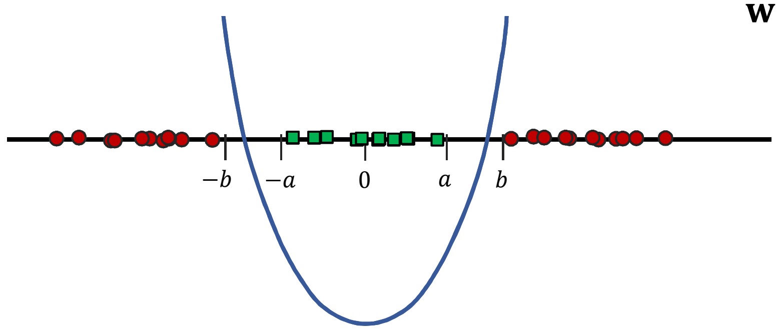
- **Solution:**
  - **map the data into a (possibly high-dimensional) vector space where the relation becomes linear**
  - **apply the linear algorithm in this space**



Nonlinear classification

- There is no linear classifier that can separate red from green.
- However, the following function can separate the regions perfectly

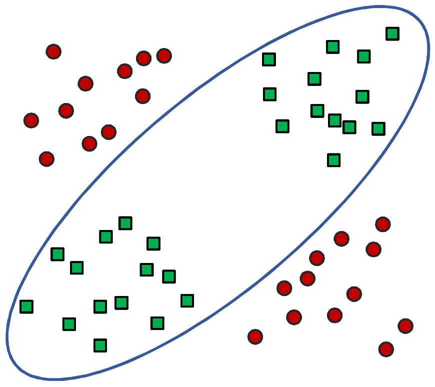
$$f(x) = x^2 - r = \langle \underbrace{(1, -r)}_{\mathbf{w}^{\mathrm{T}}}, \underbrace{(x^2, 1)}_{\phi(x)} \rangle, \text{for } a < r < b$$



- By mapping x to *feature space* $\phi(x) = (x^2, 1) \in \mathbb{R}^2$ the nonlinear problem has become linear.

Nonlinear classification

- There is no linear classifier that can separate red from green.
- However, a *conic section* separates them perfectly

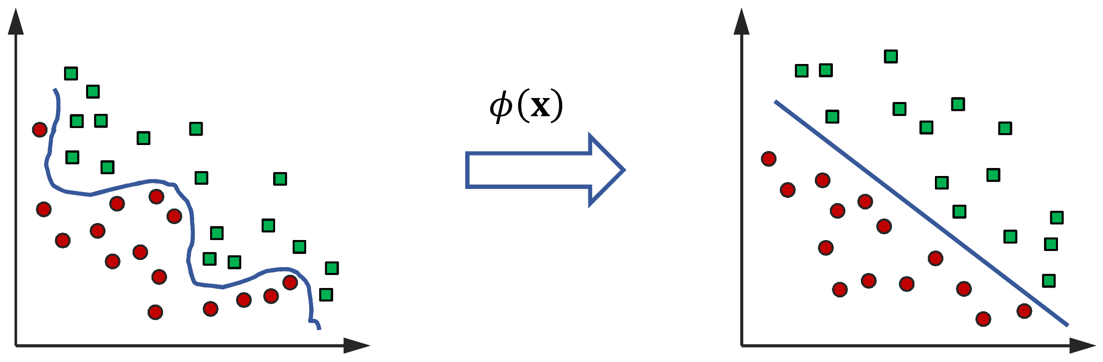


$$f(\mathbf{x}) = ax_1^2 + bx_1x_2 + cx_2^2 + dx_1 + ex_2 + g$$
$$= \langle \underbrace{(a,b,c,d,e,g)}_{\mathbf{w}^{\mathrm{T}}}, \underbrace{(x_1^2, x_1x_2, x_2^2, x_1, x_2, 1)}_{\phi(\mathbf{x})} \rangle, \qquad \phi \in \mathbb{R}^6$$

CHALMERS
UNIVERSITY OF TECHNOLOGY

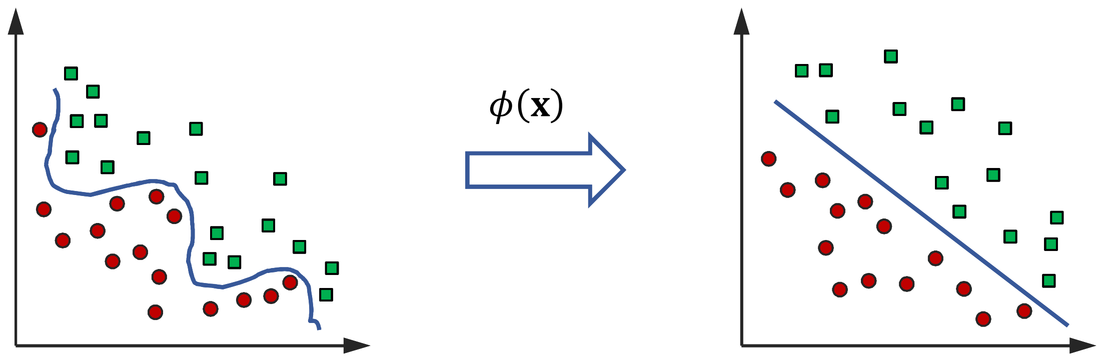
Kernel methods: motivation

- **Problem:**
  - **computationally difficult to represent data in high dimensions**



Kernel methods: motivation

- Alternative:
  - compute similarity measure between vectors in feature space
  - apply algorithms based on similarity measures



Kernel definition

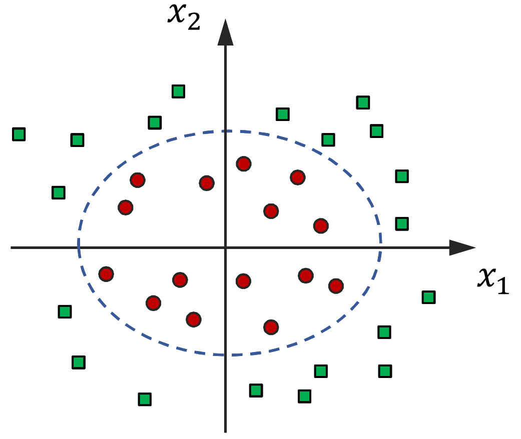
- For input vectors $\mathbf{x}, \mathbf{z} \in \mathcal{X}$ and a mapping $\phi: \mathcal{X} \to \mathbb{R}^N$

$$k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^{\mathrm{T}} \phi(\mathbf{z})$$

  is a ***kernel function***.

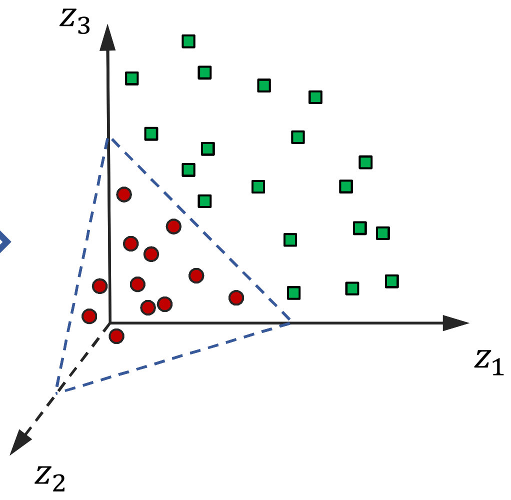
- The kernel trick: we don't need the coordinates of the data in feature space. Just the inner product between vectors.

Example



Feature map: $\phi: \mathbb{R}^2 \rightarrow \mathbb{R}^3$

$$\phi(x_1, x_2) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$$



Kernel: $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^{\mathrm{T}}\mathbf{z} = (x_1z_1 + x_2z_2)^2 = x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2 =$

$$= \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)\left(z_1^2, \sqrt{2}z_1z_2, z_2^2\right)^{\mathrm{T}} = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{z})$$

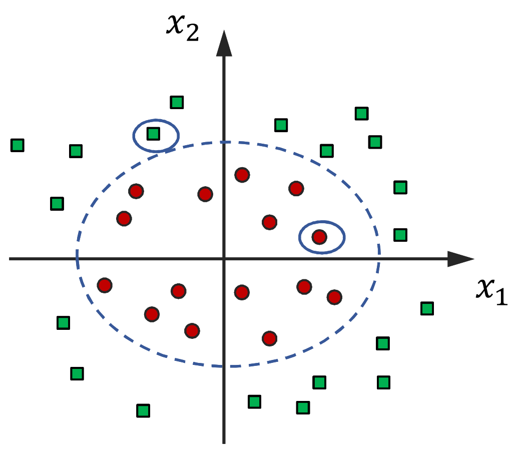
where $\phi(\mathbf{x}) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)^{\mathrm{T}}$ **is the nonlinear feature mapping.**

CHALMERS
UNIVERSITY OF TECHNOLOGY

Kernel functions

Another view:

- a kernel $k(\mathbf{x}, \mathbf{z})$ is a **measure of similarity** between vectors $\mathbf{x}, \mathbf{z} \in \mathcal{X}$ where $\mathcal{X}$ is some abstract space.
- or simply a **distance measure** between points in feature space



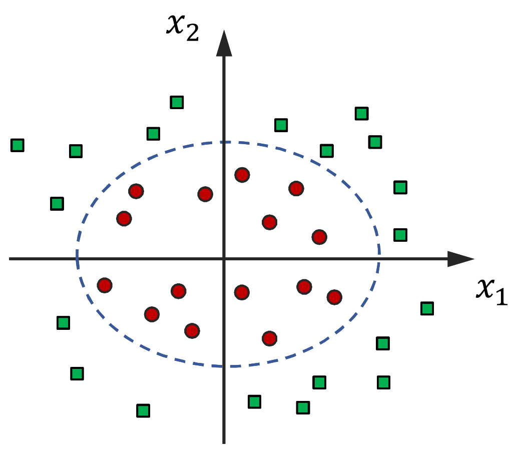
Kernel functions

However, a feature space is not unique to a given kernel:

$k(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$ **is also the kernel that computes the inner product of the map**

$$\psi(x_1, x_2) = (x_1^2, x_2^2, x_1 x_2, x_2, x_1) \in \mathbb{R}^4$$

Moreover, every prospective kernel does not necessarily correspond to a dot product in some space.



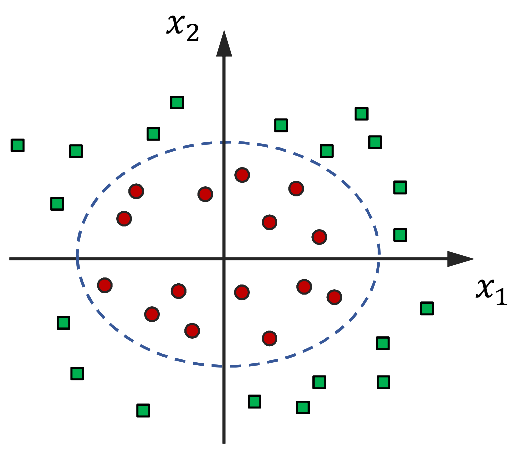
Kernel functions

Mercer's condition:

A continuous, symmetric, positive semi-definite kernel function can be written as a dot product of vectors in a higher dimension

$$k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^{\mathrm{T}} \phi(\mathbf{z})$$

Positive semi-definite: a symmetric matrix with positive eigenvalues



CHALMERS
UNIVERSITY OF TECHNOLOGY

Let’s summarize

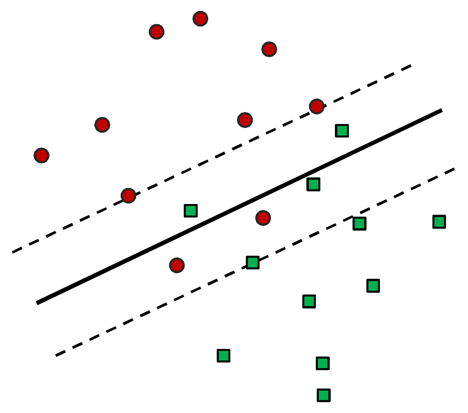
- We have a data set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ where $\mathbf{x} \in \mathbb{R}^D$ and $y_i \in \{-1, +1\}$
- We want a nonlinear projection $\phi(\mathbf{x})$ onto a higher dimension

$$\phi: \mathbb{R}^D \to \mathbb{R}^{D+d}, \qquad d > 0$$

where classes ***have a better chance*** of being linearly separable.

Cover’s theorem (informally):

”A nonlinear projection in a high-dimensional space is more likely to be linearly separable than in a low-dimensional space”



Let’s summarize

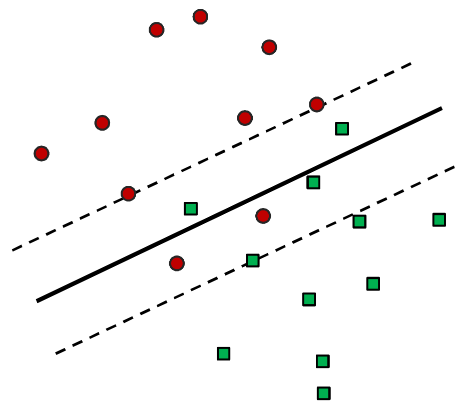
- The separating hyperplane in $\mathbb{R}^{D+d}$ will be defined by

$$\sum_{j=1}^{D+d} w_j \phi_j(\mathbf{x}) + b = \mathbf{w}^{\mathrm{T}} \phi(\mathbf{x}) + b = 0$$

- The optimal hyperplane is given by

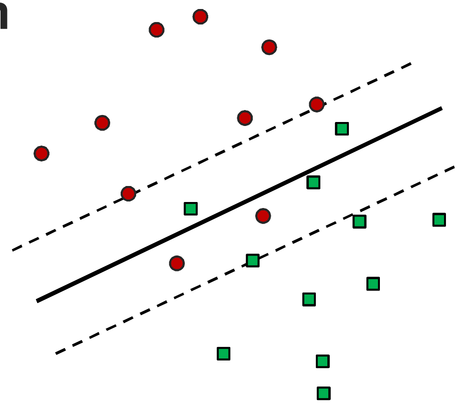
$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \phi(\mathbf{x}_i) \Leftrightarrow \sum_{i=1}^{N} \lambda_i y_i k(\mathbf{x}_i, \mathbf{x}) = 0$$

where $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^{\mathrm{T}} \phi(\mathbf{x}_j)$ is a kernel function and λ_i Lagrange multipliers.



How to choose the mapping $\phi(\mathbf{x})$?

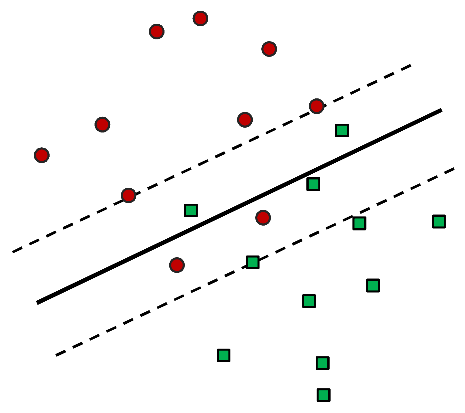
- Choosing an optimal feature space is non-trivial
- The kernel trick reduces this to choosing the best kernel, and determine the corresponding implicit mapping $\phi(\mathbf{x})$.
- Performance of the algorithm highly depends on the kernel
- The best kernel depends on the specific problem
- Kernels can be applied to
  - Numeric vectors
  - Strings
  - Trees
  - Graphs



How to choose the best kernel

We want the kernel to be

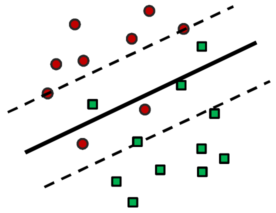
- **Valid**: an implicit mapping must exist
  = a kernel that can be expressed as the dot product of two vectors
  = satisfy the Mercer's condition of positive semi-definiteness
- **Accurate**: embody the "true" similarity between objects
- **Appropriate**: generalize outside training data
- **Efficient**: computations should be feasible



Which kernels meet Mercer's condition?

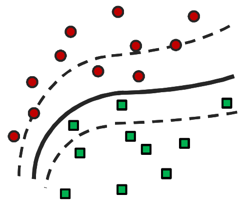
- **Linear** kernels

$$k(\mathbf{x},\mathbf{z}) = \mathbf{x}^{\mathrm{T}}\mathbf{z}$$



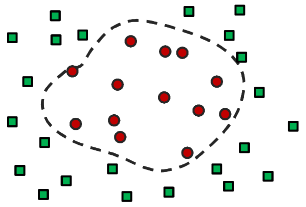
- **Polynomial** kernels

$$k(\mathbf{x},\mathbf{z}) = \left(1+\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^{n}$$



- **Radial basis function** (RBF) kernels

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



Radial basis function (RBF) kernels

- The ***squared exponential*** (SE) or ***Gaussian*** kernel

$$k(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{z})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{z})\right)$$

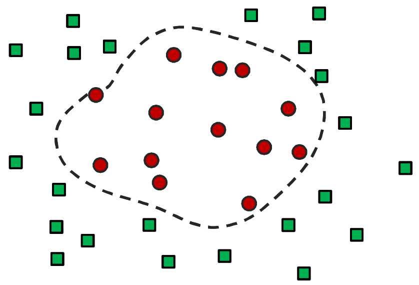
- If the covariance matrix Σ is diagonal, we get

$$k(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{1}{2}\sum_{j}\frac{1}{\sigma^2}\left(x_j - z_j\right)^2\right)$$

- If Σ is spherical

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

An RBF kernel

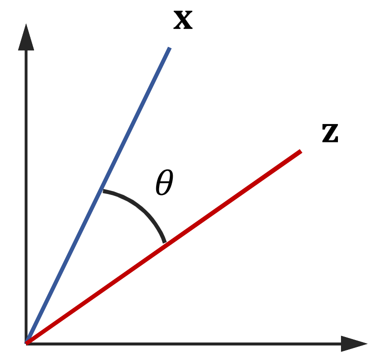


Kernels for comparing text documents

- If x_{ij} = the number of times word j occurs in document i

$$k(\mathbf{x}, \mathbf{z}) = \frac{\mathbf{x}^{\mathrm{T}}\mathbf{z}}{\|\mathbf{x}\|_2 \|\mathbf{z}\|_2} = \cos(\theta)$$

called the ***cosine similarity*.**



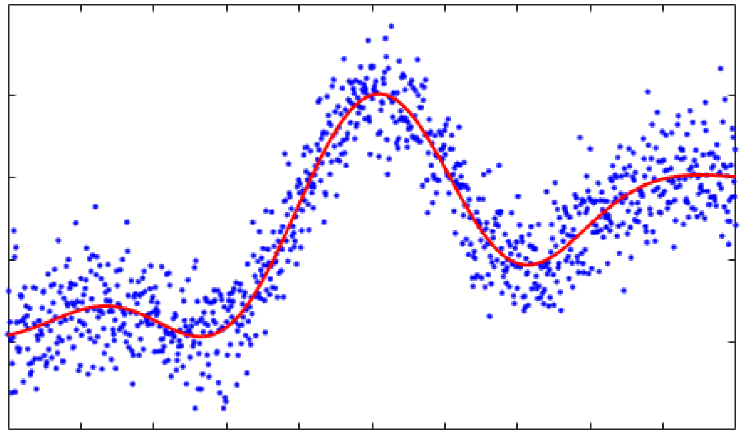
Matern kernel

- **Commonly used in Gaussian processes**

$$k(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu r}}{l}\right)^{\nu} B_{\nu}\left(\frac{\sqrt{2\nu r}}{l}\right) \rightarrow \text{SE kernel as } \nu \rightarrow \infty$$

where $r = \|\mathbf{x} - \mathbf{z}\|$, $\nu \geq 0$, $l > 0$ **and** B_{ν} **a modified Bessel function.**

$$k(r) = \exp\,(-r/l)\ \textbf{when}\ \ \nu = 1/2$$



String kernels

- Consider two strings x and z of lengths $D_{\mathbf{x}}$ and $D_{\mathbf{z}}$, defined on a protein alphabet
  - $\mathcal{A} = \{\mathrm{A,R,N,D,C,E,\ Q,\ G,H,\ I,\ L,K,M,\ F,\ P,\ S,\ T,W,\ Y,\ V}\}$

where

$\mathbf{x}$ $(D_{\mathbf{x}} = 110)$:

```
IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTL
ESQTVQGGTVERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDYLQEFLGV
MNTEWI
```

$\mathbf{z}$ $(D_{\mathbf{z}} = 153)$:

```
PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAERLQENLQAY
RTFHVLLARLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLE
YKIPRNEADGMLFEKKLWGLKVLQELSQWTVRSIHDLRFISSHQTGIP
```

Similarity measure: number of common substrings

$$k(\mathbf{x},\mathbf{z}) = \sum_{s\in\mathcal{A}^*} w_s \phi_s(\mathbf{x})\phi_s(\mathbf{z})$$

where s is a substring, $w_s \geq 0$ and $\mathcal{A}^*$ the set of all substrings from $\mathcal{A}$.

String kernels

- If $w_s = 1$ for a nonempty substring $|s| > 0$:

  $\phi(\mathbf{x})$ = number of times each char in $\mathcal{A}$ occurs in $\mathbf{x}$

  - bag-of-characters model

- If we require each substring s to be surrounded by white space

  $\phi(\mathbf{x})$ = number of times each word s occurs in $\mathbf{x}$

  - bag-of-words model

- If we only consider strings with $|s| = k$ we get the k-spectrum kernel

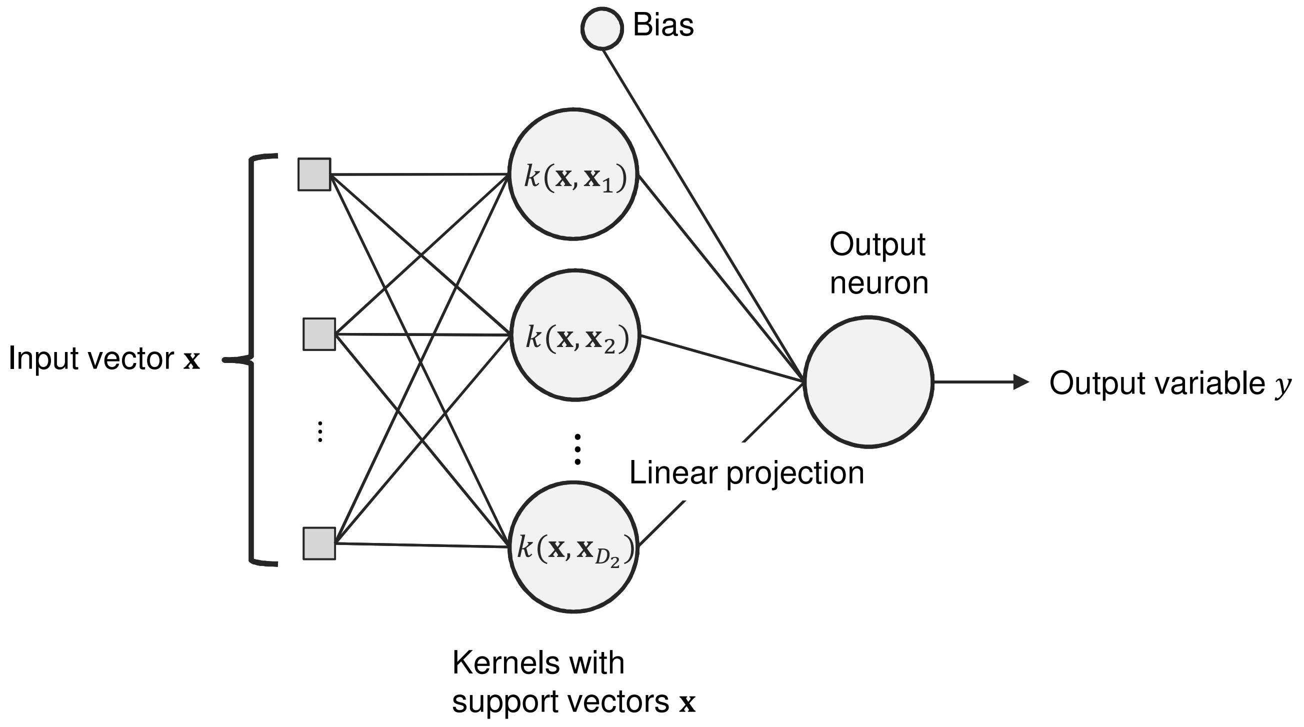
Similarity measure:
number of common substrings

$$k(\mathbf{x}, \mathbf{z}) = \sum_{s \in \mathcal{A}^*} w_s \phi_s(\mathbf{x}) \phi_s(\mathbf{z})$$

where s is a substring, $w_s \geq 0$ and $\mathcal{A}^*$ the set of all substrings from $\mathcal{A}$.

CHALMERS
UNIVERSITY OF TECHNOLOGY

An SVM as a neural network


Bias
Output neuron
Input vector $\mathbf{x}$
$k(\mathbf{x}, \mathbf{x}_1)$
$k(\mathbf{x}, \mathbf{x}_2)$
$k(\mathbf{x}, \mathbf{x}_{D_2})$
Output variable y
Linear projection
Kernels with support vectors $\mathbf{x}$

Gaussian processes

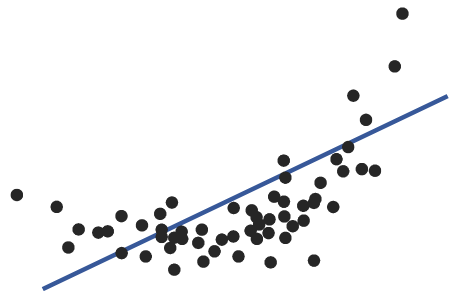
- **Linear regression: determine relation f between response y and independent variable x**

$$y = f(x) + \epsilon$$

where f is assumed to be linear: $f(x) = \beta_0 + \beta_1 x$

- **Bayesian linear regression: determine a posterior distribution over the parameters β_0 and β_1 that gets updated whenever new data is available.**

- **Gaussian processes: finds a posterior distribution over the possible *functions* $f(x)$ consistent with the observed data and a suitable prior.**



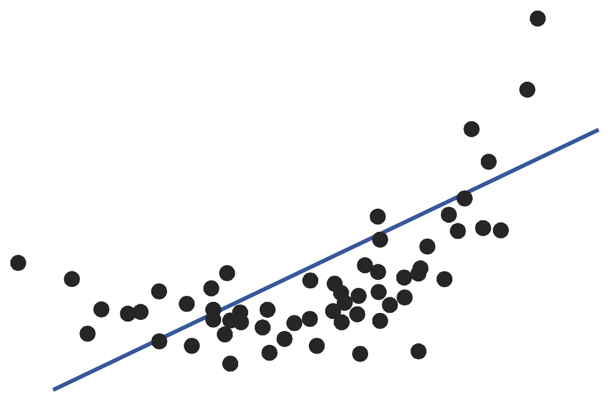
CHALMERS
UNIVERSITY OF TECHNOLOGY

Gaussian processes

- **The current example isn't really linear.**
- **Quadratic function**

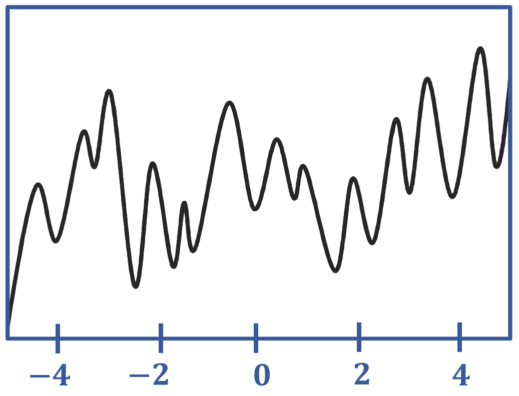
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$$

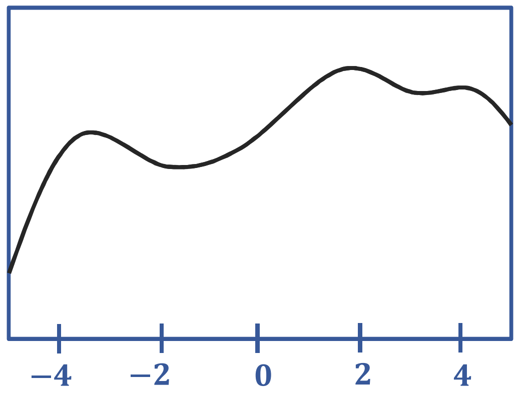
- **Three parameters to estimate:** $\beta_0, \beta_1, \beta_2$
- **But what if we don't know how many parameters we should use?**
- **Instead of searching for suitable parameter values for a fixed number of parameters (and a fixed function), we want to *search among all functions* that fit our data.**



Gaussian processes

- We need to define a prior over the function space.
- Assume we limit our x-values: $-5 \leq x \leq 5$.
- In that domain we want to sample functions that are reasonablye *smooth.*
- We use a *covariance matrix* to ensure that points close together in input space will produce output values thar are also close together.



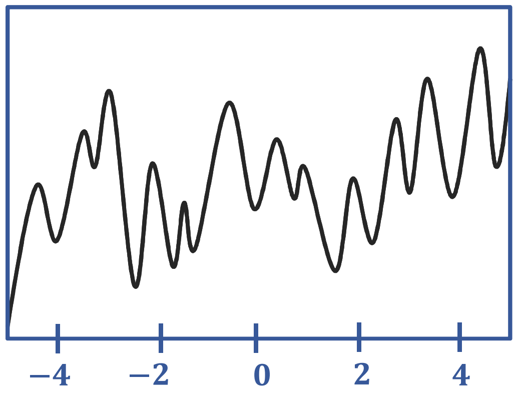


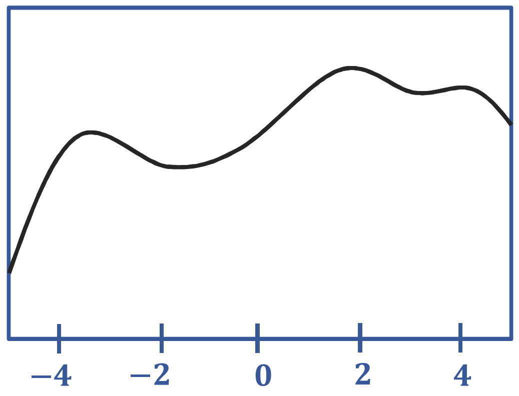
Gaussian processes

- A Gaussian process defines a prior over functions, which, given observed data, can be converted into a posterior.
- Instead of explicitly representing a distribution over a function, choose a finite set of points $\{x_1, \ldots, x_n\}$.
- A GP assumes that the function values $f(x_1), \ldots, f(x_n)$ has a jointly Gaussian distribution with some mean $\mu(\mathbf{x})$ and covariance $\Sigma(\mathbf{x})$, given by

$$\boldsymbol{\Sigma}_{ij} = k(x_i, x_j)$$

where k is a positive definite kernel.





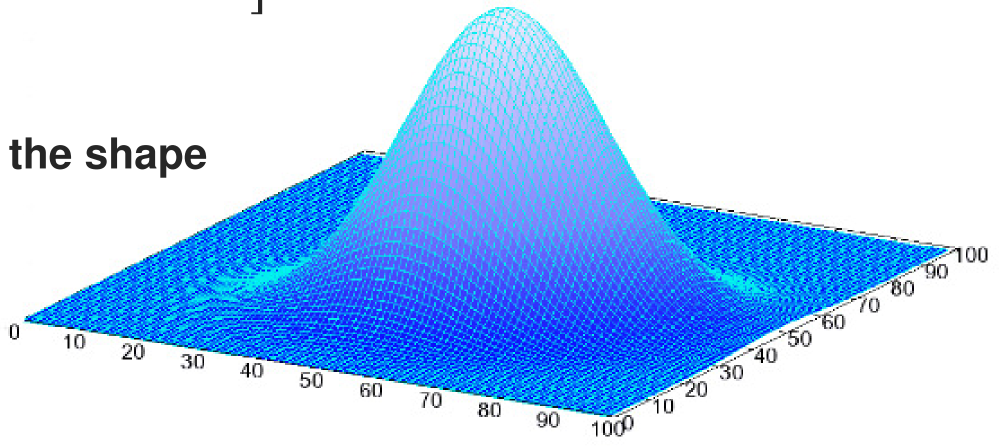
Multivariate normal distribution

- Gaussian processes are based on the **multivariate normal (Gaussian) distribution**

$$\mathbf{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

  where Σ is a ***covariance matrix***.

- The covariance matrix determines the shape of the "bell"

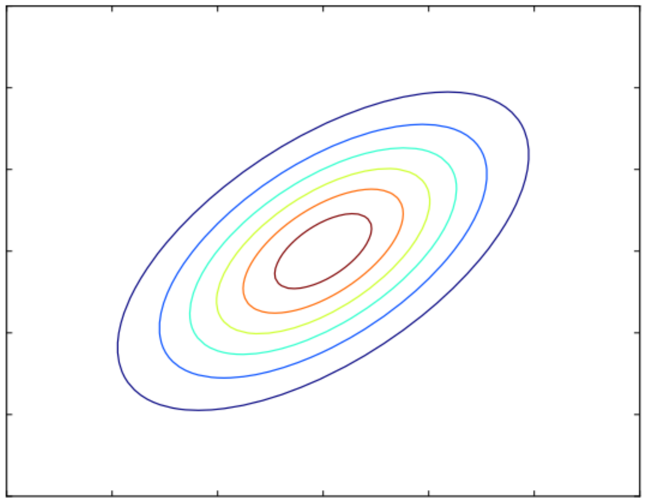


Multivariate normal distribution

- Viewing from above, if the contours form a perfect circle, the variables are independent, and the covariance is zero

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- With covariance $\boldsymbol{\Sigma}_{12} \neq 0$, the contour will have a more oval shape.



Gaussian processes

- **Assume that we want to learn a function f from data $\mathcal{D} = \{(x_i, y_i): i = 1, \dots, D\}$**
- **Assume we have a distribution $p(f)$ over functions,**
- **Now: $p(f)$ is a Gaussian process if for a finite subset $\{x_1, \dots, x_n\}$ the marginal distribution over that subset**

$$p(f(x_1), \dots, f(x_n))$$

has a Gaussian distribution.

- **Now, if the prior is Gaussian, so is the posterior**

$$p(f|\mathcal{D}) = \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})}$$

Gaussian processes

- The Gaussian process is parameterized by a mean vector and a covariance matrix

$$p\begin{pmatrix} f(x_1) \\ f(x_2) \end{pmatrix} \sim N(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\Sigma})$$

where

$$\boldsymbol{\mu}(\mathbf{x}) = \begin{bmatrix} \mu(x_1) \\ \mu(x_2) \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) \\ k(x_2, x_1) & k(x_2, x_2) \end{bmatrix}$$

and $k(x_1, x_2)$ is a kernel function.

Gaussian processes

- So: we have the following data

$$\mathcal{D} = \{(x_i, y_i): i = 1, \dots, D\}$$

and for some new point x^* we want to predict y^*.

- To do this we want to find a function f such that

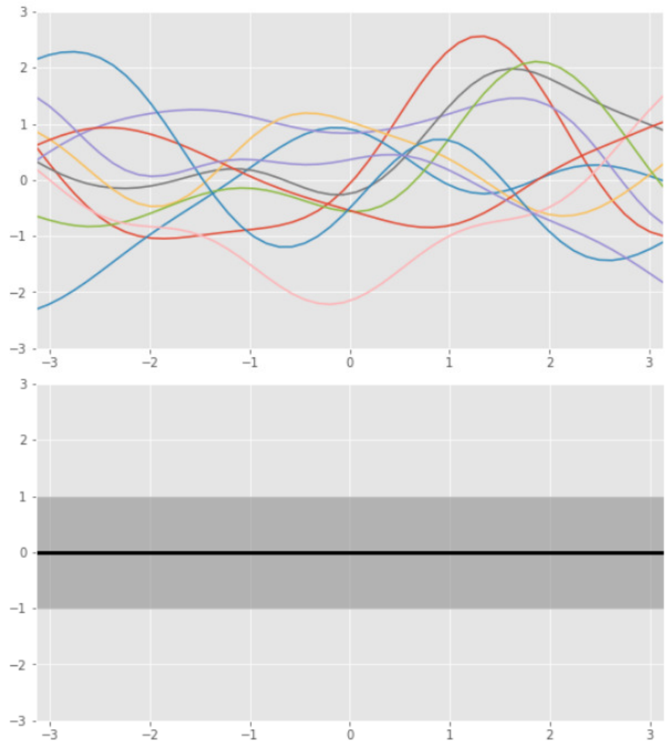
$$y_i = f(x_i)$$

- Instead we assume that $\{f(x_1), \dots, f(x_D)\}$ follow a joint normal distribution, and use it to compute the posterior distribution

$$p(f(x^*)|x^*, \mathcal{D}, f) = N(\mu^*, \Sigma^*)$$

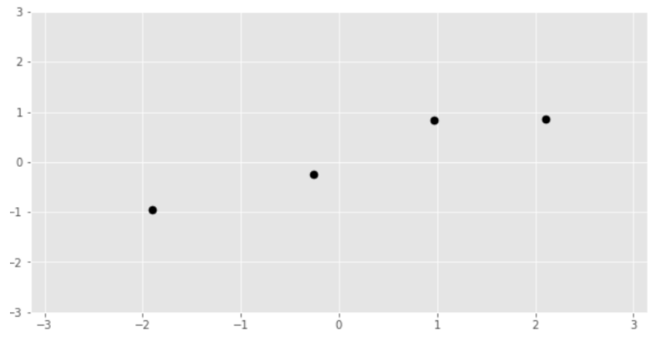
Gaussian processes

- Intuitively, be begin by sampling from our prior distribution



Gaussian processes

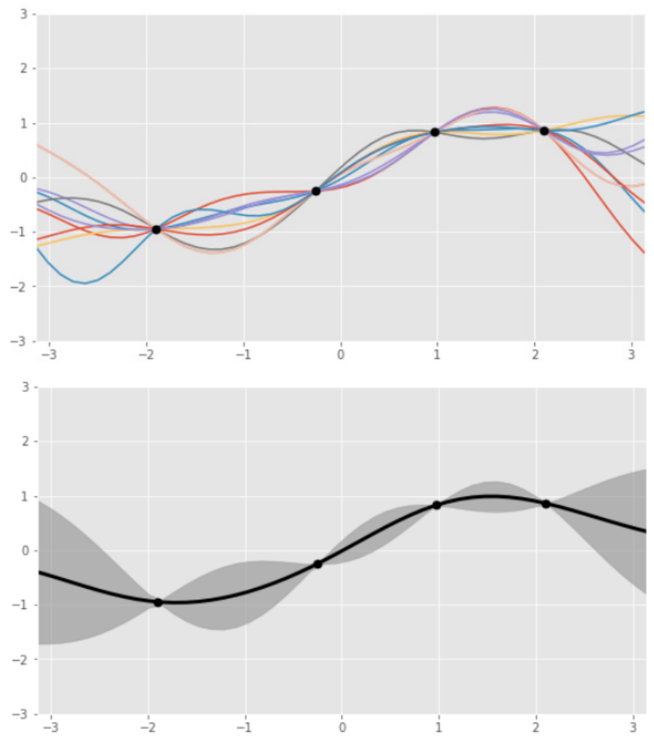
- Intuitively, be begin by sampling from our prior distribution
- We use our training data to represent the outputs of the unknown function.



CHALMERS
UNIVERSITY OF TECHNOLOGY

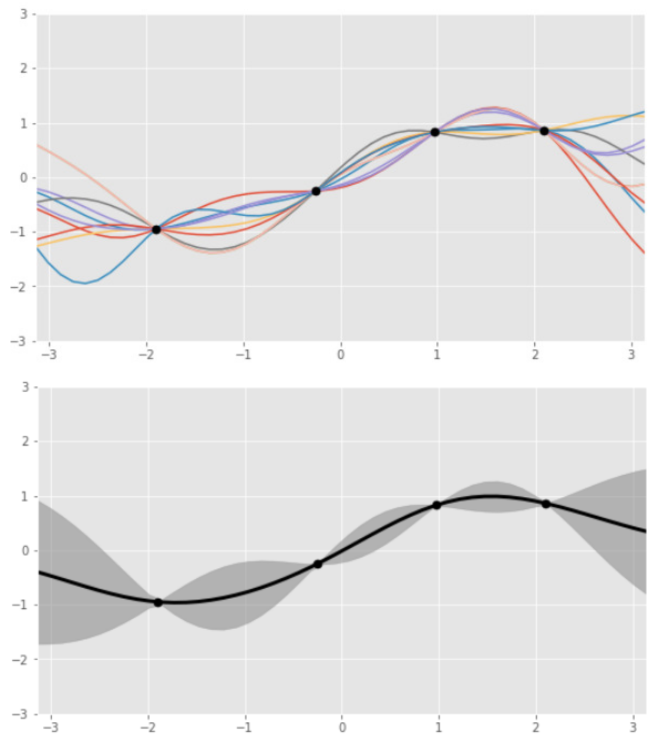
Gaussian processes

- Intuitively, be begin by sampling from our prior distribution
- We use our training data to represent the outputs of the unknown function.
- And update the posterior.



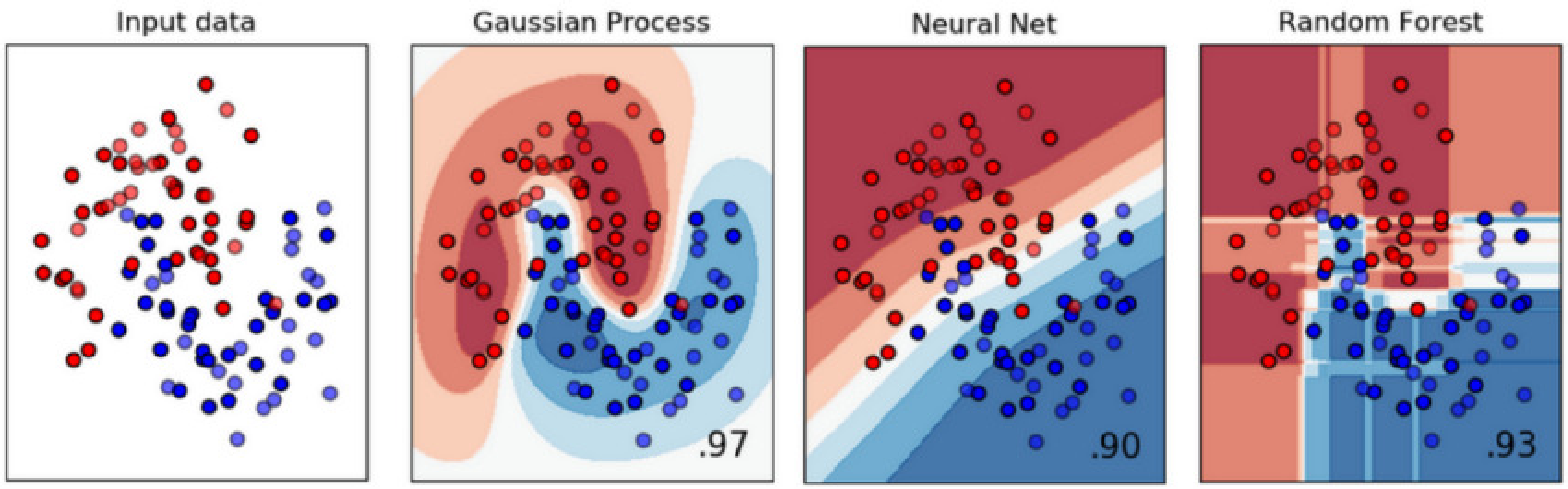
Gaussian processes

- Where does the kernel k come from?
- The covariance matrix characterizes the similarities between nearby points.
- The same range of kernels available as for SVMs.
    - The squared exponential (SE)
    - The radial basis function (RBF)
    - The Matern
    - …



CHALMERS
UNIVERSITY OF TECHNOLOGY

Gaussian processes



Advantages of GPs over SVMs

- GPs handle uncertainty in unknown function f by averaging, not minimizing
- GPs can learn kernel parameters from data, no matter how flexible we want to make the kernel
- GPs can learn regularization parameters without cross-validation.
- Can incorporate interpretable noise models and priors over functions, and can sample from prior to get intuitions about the model assumptions.
- We can combine automatic feature selection with learning using Automatic Relevance Determination (ARD)