ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik
Computer Science

Introduction to Machine Learning

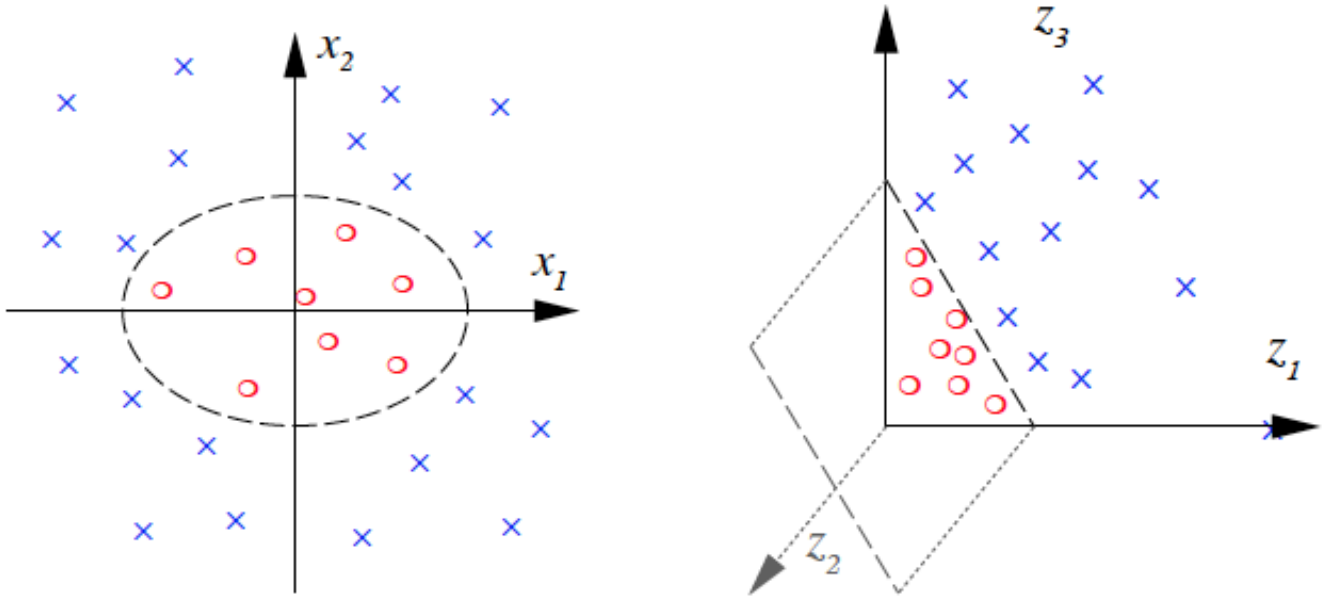
Non-linear prediction with kernels (continued)

Prof. Andreas Krause
Learning and Adaptive Systems (las.ethz.ch)

Solving non-linear classification tasks

- How can we find nonlinear classification boundaries?
- Similar as in regression, can use non-linear transformations of the feature vectors, followed by linear classification

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$



Avoiding the feature explosion

- Need O(d^k) dimensions to represent (multivariate) polynomials of degree k on d features
- **Example**: *d=10000, k=2* ➔ Need *~100M* dimensions

- In the following, we can see how we can efficiently implicitly operate in such high-dimensional feature spaces (i.e., without ever explicitly computing the transformation)

The „Kernel Trick“

- Express problem s.t. it only depends on inner products
- Replace inner products by kernels

$$\mathbf{x}_i^T \mathbf{x}_j \quad \Rightarrow \quad k(\mathbf{x}_i, \mathbf{x}_j)$$

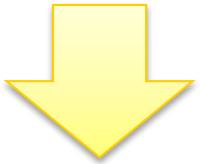
- This „trick“ is very widely applicable!

The „Kernel Trick“

- Express problem s.t. it only depends on inner products
- Replace inner products by kernels

- Example: Perceptron

$$\hat{\alpha} = \arg\min_{\alpha_{1:n}} \frac{1}{n} \sum_{i=1}^{n} \max\{0, -\sum_{j=1}^{n} \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j\}$$



$$\hat{\alpha} = \arg\min_{\alpha_{1:n}} \frac{1}{n} \sum_{i=1}^{n} \max\{0, -\sum_{j=1}^{n} \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)\}$$

- Will see further examples later

Kernelized Perceptron

Training

- Initialize $\alpha_1 = \cdots = \alpha_n = 0$
- For *t=1,2,…*
  - Pick data point ($\boldsymbol{x}_i, y_i$) uniformly at random
  - Predict
$$\hat{y} = \text{sign}\left(\sum_{j=1}^{n} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i)\right)$$
  - If $\hat{y} \neq y_i$ set $\alpha_i \leftarrow \alpha_i + \eta_t$

Prediction

- For new point x, predict
$$\hat{y} = \text{sign}\left(\sum_{j=1}^{n} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x})\right)$$

Properties of kernel functions

- Data space *X*
- A kernel is a function $k : X \times X \to \mathbb{R}$
- Can we use any function?

$\exists\, \phi : X \to \mathbb{R}^D \quad \text{s.t.} \quad k(x, x') = \phi(x)^T \phi(x')$

- *k* must be an inner product in a suitable space

➔ *k* must be symmetric!

$\forall x, x' \in X: \; k(x, x') = \phi(x)^T \phi(x') = \phi(x')^T \phi(x) = k(x', x)$

➔ Are there other properties that it must satisfy?

Positive semi-definite matrices

Symmetric matrix $M \in \mathbb{R}^{n \times n}$ is positive semidefinite iff

(i) $\forall x \in \mathbb{R}^n : \; x^T M x \geq 0$

$\equiv$

(ii) All eigenvalues of $M \geq 0$

(i) $\Rightarrow$ (ii): M is symmetric $\Rightarrow$ $M = U D U^T$ for $D = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}$, $U^T U = I = U U^T$

$U = (u_1 | \dots | u_n)$ s.t. $M u_i = \lambda_i u_i$

wtp: $\lambda_i \geq 0 \;\; \forall i$

$u_i^T M u_i = u_i^T (\lambda_i u_i) = \lambda_i u_i^T u_i = \lambda_i \overset{\text{by (i)}}{\geq} 0$

□

Kernels ➔ semi-definite matrices

- Data space X (possibly infinite)
- Kernel function $k : X \times X \to \mathbb{R}$
- Take any finite subset of data $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq X$
- Then the kernel (gram) matrix

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} = \begin{pmatrix} \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_1) & \ldots & \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_n) \\ \vdots & & \vdots \\ \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_1) & \ldots & \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_n) \end{pmatrix}$$

is positive semidefinite

$K = \Phi^T \Phi$, where $\Phi = \left(\phi(x_1) \mid \ldots \mid \phi(x_n) \right) \in \mathbb{R}^{D \times n}$

Sps. $x \in \mathbb{R}^n$: Then $x^T K x = \underbrace{x^T \Phi^T}_{v^T} \underbrace{\Phi x}_{v} = v^T v \geq 0$ □

Semi-definite matrices ➔ kernels

- Suppose the data space $X=\{1,\ldots,n\}$ is finite, and we are given a positive semidefinite matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$
- Then we can always construct a feature map

$$\phi : X \to \mathbb{R}^n$$

such that $\mathbf{K}_{i,j} = \phi(i)^T \phi(j)$

Let $A \subset \{1,\ldots,n\}$, K^A kernel matrix for set A. Then $K^A = K_{AA}$

K is s.p.d. $\Rightarrow K = UDU^T$, where $D = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}$ and $\lambda_i \geq 0 \ \forall i$

$D = D^{\frac{1}{2}} D^{\frac{1}{2}T}$, where $D^{\frac{1}{2}} = \begin{pmatrix} \sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda_n} \end{pmatrix}$

$\Phi : X \to \mathbb{R}^n$

$\Phi : i \mapsto \phi_i$

$\Rightarrow K = \underbrace{UD^{\frac{1}{2}}}_{\Phi^T} \underbrace{D^{\frac{1}{2}T} U^T}_{\Phi}$, where $\Phi = [\phi_1 | \ldots | \phi_n]$

Now it holds that $k(i,j) = K_{i,j} = \phi_i^T \phi_j$

□

Outlook: Mercer's Theorem

Let *X* be a compact subset of $\mathbb{R}^d$ and $k : X \times X \to \mathbb{R}$ a kernel function

Then one can expand *k* in a uniformly convergent series of bounded functions ϕ_i s.t.

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \, \phi_i(x)\phi_i(x')$$

Can be generalized even further

Definition: kernel functions

- Data space *X*
- A **kernel** is a function $k : X \times X \to \mathbb{R}$ satisfying
- 1) Symmetry: For any $\mathbf{x}, \mathbf{x}' \in X$ it must hold that

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$$

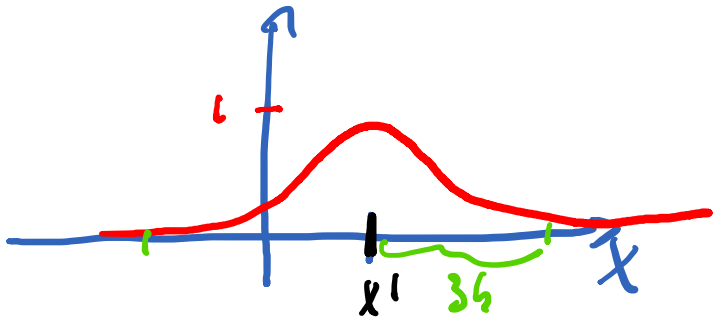
- 2) Positive semi-definiteness: For any *n*, any set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq X$, the kernel (Gram) matrix

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

must be positive semi-definite

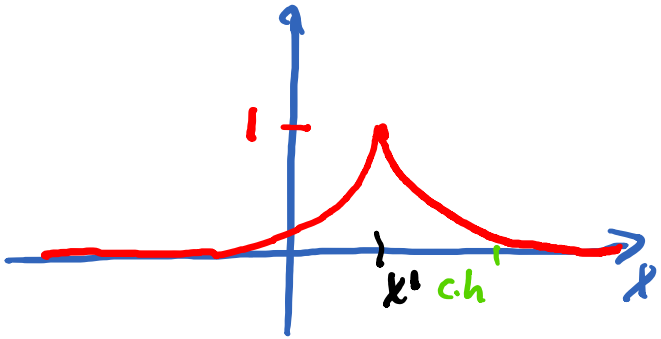
Examples of kernels on $\mathbb{R}^d$

- Linear kernel: $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$
- Polynomial kernel: $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^d$
- Gaussian (RBF, squared exp. kernel): $k(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||_2^2 / 2h^2)$



Bandwidth / Lengthscale parameter

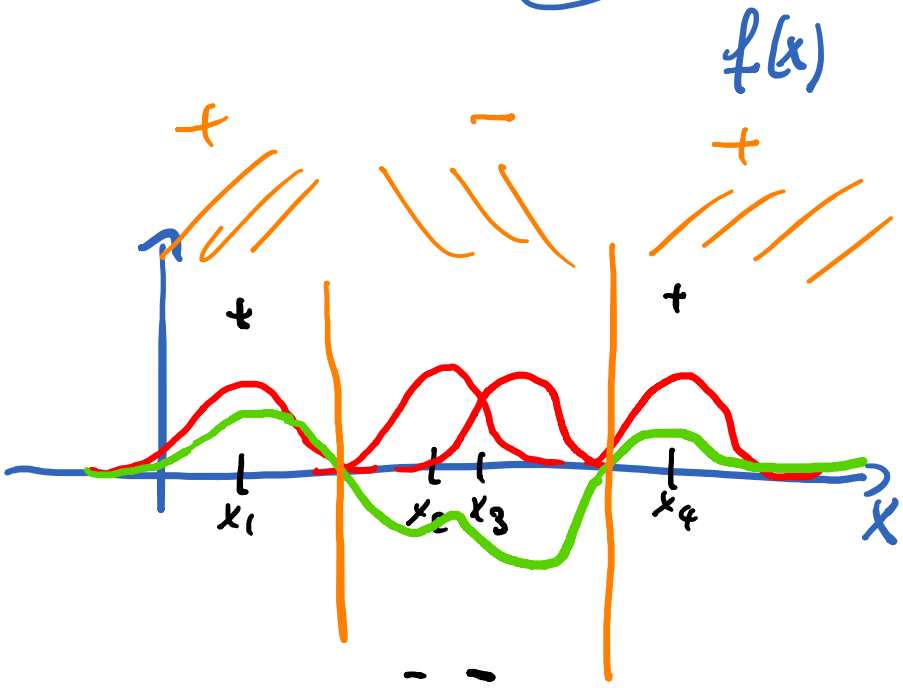
- Laplacian kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||_1 / h)$



Effect of kernel on function class

- Given kernel k, predictors (for kernelized classification) have the form

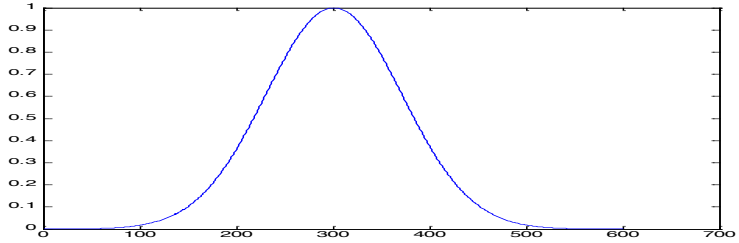
$$\hat{y} = \text{sign}\left(\sum_{j=1}^{n} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x})\right)$$



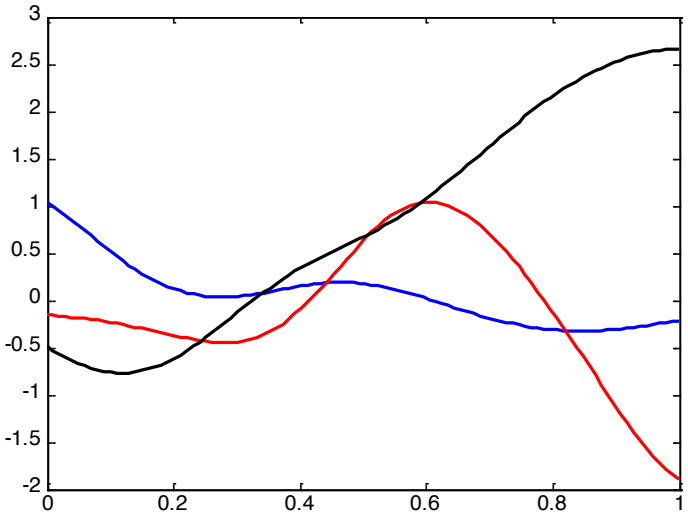
$D = \{(x_1, +), (x_2, -), (x_3, -), (x_4, +)\}$

Example: Gaussian kernel

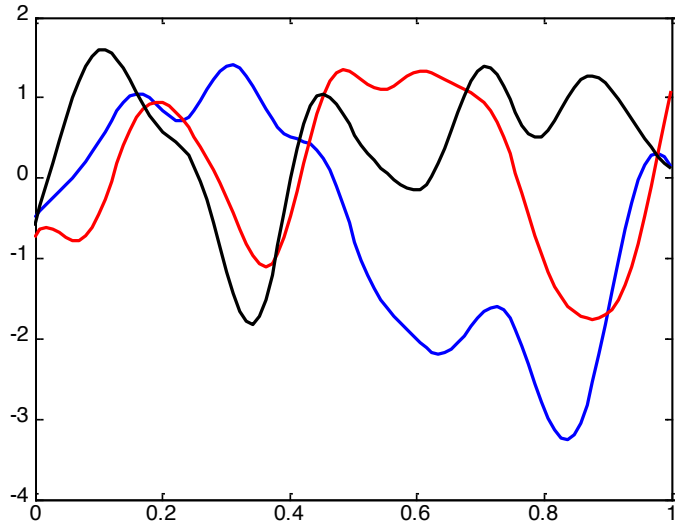
$$k(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||_2^2/h^2)$$



$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$



Bandwidth h=.3



Bandwidth h=.1

Examples of (non)-kernels

$$k(x, x') = \sin(x)\cos(x')$$

Not symmetric! $k(x,x') \neq k(x',x)$

e.g. for $x = 0$, $x' = \frac{\pi}{2}$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T M \mathbf{x}'$$

k symmetric $\Leftrightarrow$ M is symmetric

k positive (semi-)definite $\Leftrightarrow$ M is positive (semi-)definite

ID: Sps. $M = -1$. Then $k(x,x') = -xx'$, hence, $k(x,x) = -x^2 \Rightarrow k$ not pos def.

Sps. M is pos. def. Then $M = UD^{\frac{1}{2}}D^{\frac{1}{2}T}U^T = V^TV$. Then

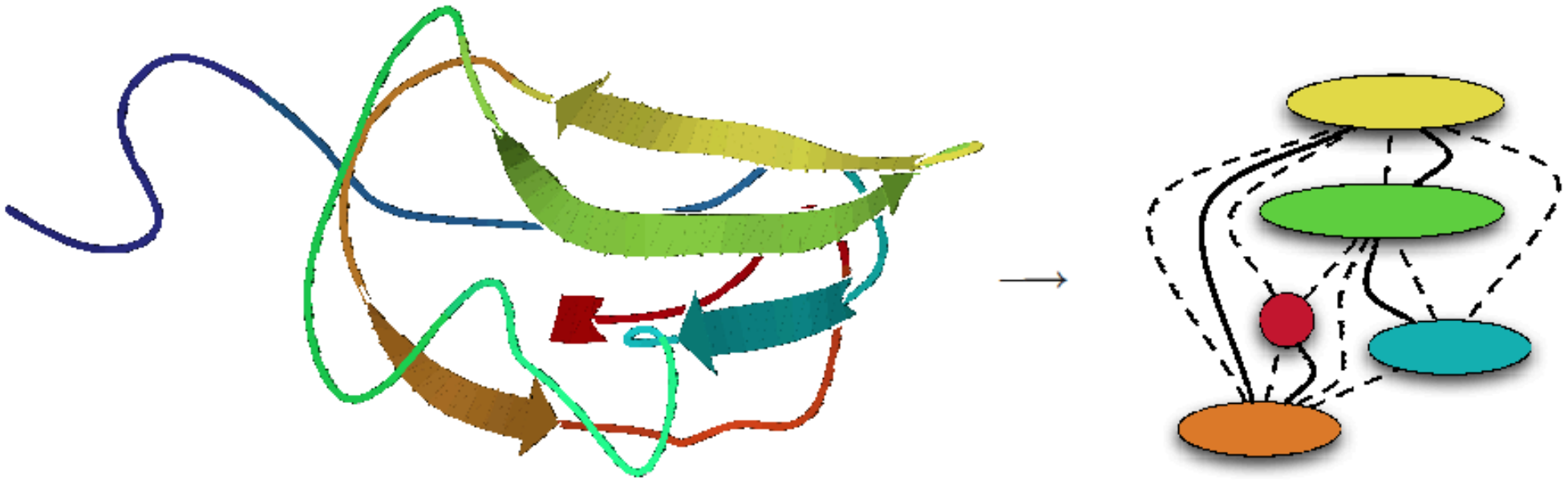
$k(x,x') = x^TMx' = x^TV^TVx' = (Vx)^T(Vx') = \phi(x)^T\phi(x')$ for $\phi(x) = Vx$ □

Kernels beyond $\mathbb{R}^d$

- Can define kernels on a variety of objects:

- Sequence kernels
- Graph kernels
- Diffusion kernels
- Kernels on probability distributions
- ...

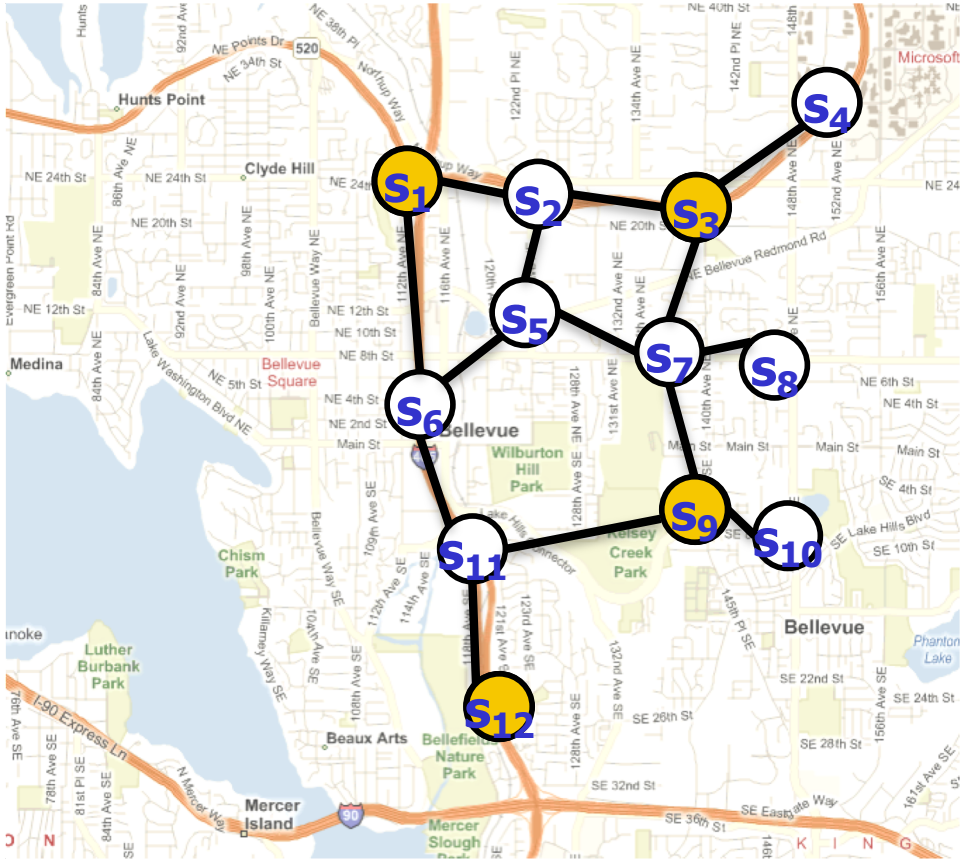
Example: Graph kernels



[Borgwardt et al.]

- Can define a kernel for measuring similarity between graphs by comparing random walks on both graphs (not further defined here)

Example: Diffusion kernels on graphs



$$\mathbf{K} = \exp(-\beta \mathbf{L})$$

- Can measure similarity among nodes in a graph via diffusion kernels (not defined here)

Kernel engineering (composition rules)

- Suppose we have two kernels

$$k_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \qquad k_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

  defined on data space *X*

- Then the following functions are valid kernels:

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')\, k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = c\, k_1(\mathbf{x}, \mathbf{x}') \text{ for } c > 0$$

$$k(\mathbf{x}, \mathbf{x}') = f(k_1(\mathbf{x}, \mathbf{x}'))$$

Suppose $v: \mathcal{Z} \to \mathcal{X}$

$k(z, z') = k_1(v(z), v(z'))$ is valid kernel.

Pf: $k_1(x, x') = \phi(x)^T \phi(x')$

$k(z, z') = \phi(v(z))^T \phi(v(z'))$

where *f* is a polynomial with positive coefficients or the exponential function

Example: ANOVA kernel

$$k(x, x') = \sum_{j=1}^{d} k_j(x_j, x_j')$$

where $x \in \mathbb{R}^d$, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

$k_i : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

e.g. $k_i(x_i, x_i') = \exp\left(-\frac{(x_i - x_i')^2}{h_i^2}\right)$

What functions does k model?

($x^{(i)}$: ith data point)

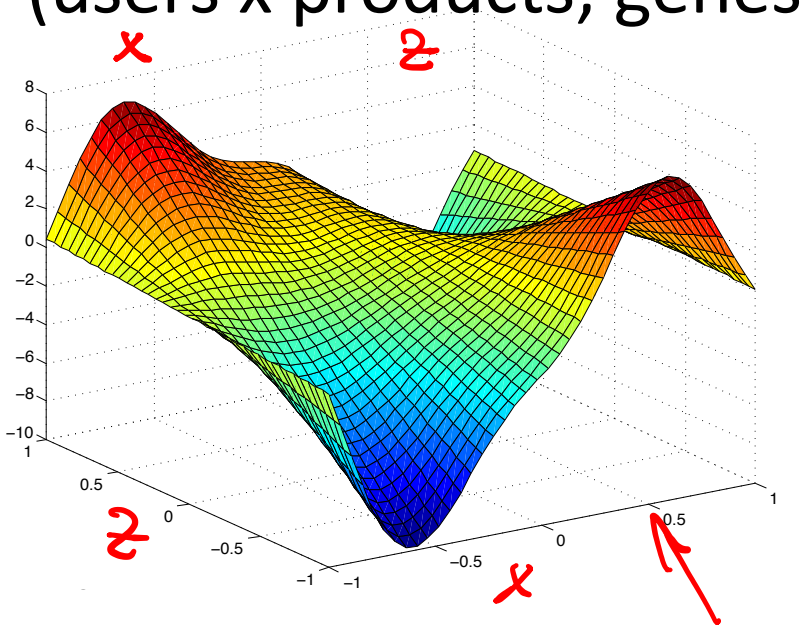
$$f(x) = \sum_{i=1}^{n} \alpha_i k(x^{(i)}, \cdot)$$

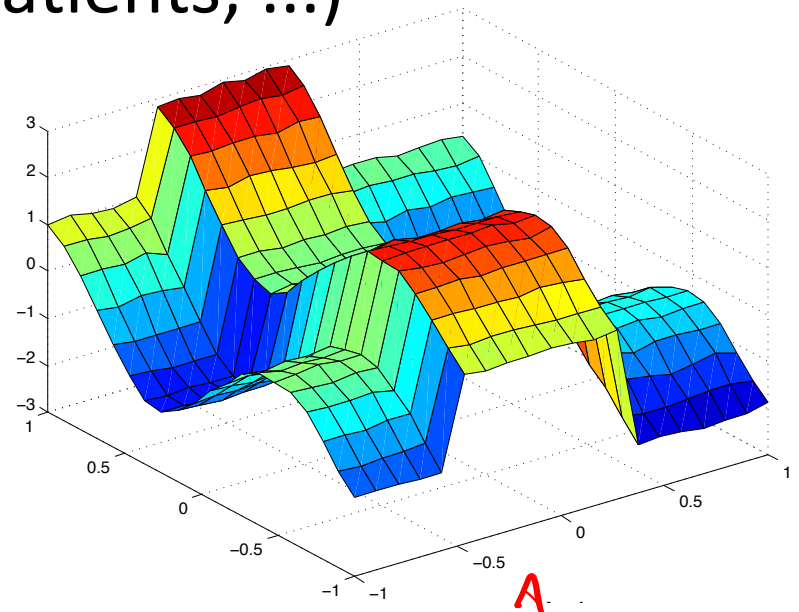
$$= \sum_{i=1}^{n} \alpha_i \sum_{j=1}^{d} k_j(x_j^{(i)}, x_j'^{(i)})$$

$$= \sum_{j=1}^{d} \underbrace{\sum_{i=1}^{n} \alpha_i k_j(x_j^{(i)}, x_j'^{(i)})}_{f_j(x_j)}$$

Example: Modeling pairwise data

- May want to use kernels to model pairwise data (users x products; genes x patients; ...)





$$k_{\text{new}}\big((x,z),(x',z')\big) := k_x(x,x') \cdot k_z(z,z')$$

$$k_x(x,x') + k_z(z,z')$$

Where are we?

- We've seen how to kernelize the perceptron
- Discussed properties of kernels, and seen examples

- Next questions:
  - What kind of predictors / decision boundaries do kernel methods entail?
  - Can we use the kernel trick beyond the perceptron?

Kernels as *similarity functions*

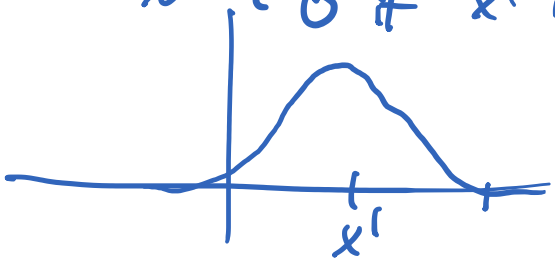
- Recall Perceptron (and SVM) classification rule:

$$y = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})\right) \approx \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i [\mathbf{x} \text{ "close" to } \mathbf{x}_i]\right)$$

$f(x)$

- Consider Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||^2/h^2)$

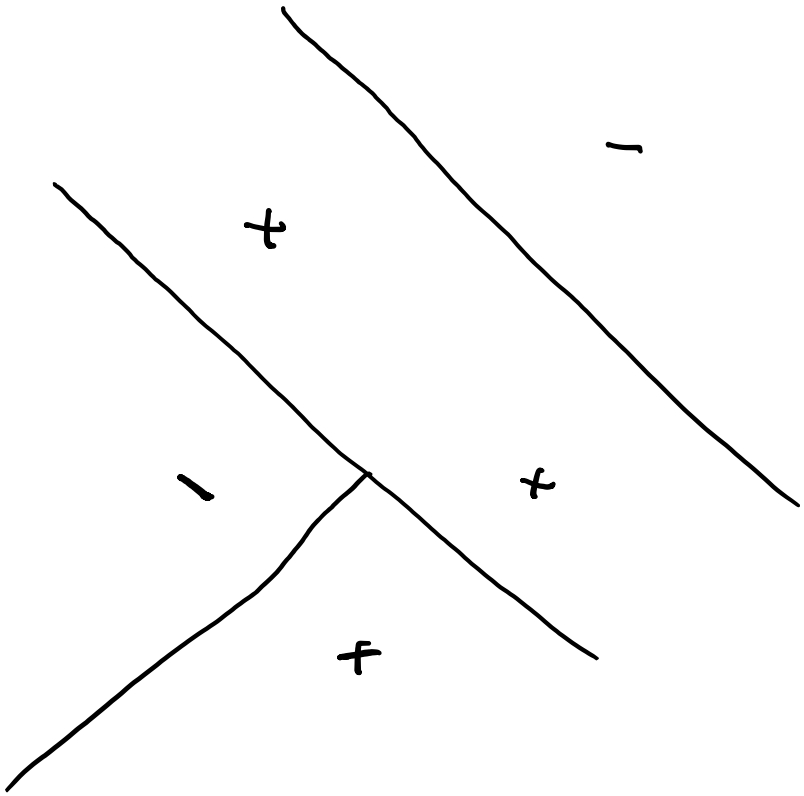
$\approx$ { 1 if x' is close to x
0 if x' is far from x



Side note: Nearest-neighbor classifiers

- For data point ***x***, predict majority of labels of *k* nearest neighbors

$$y = \text{sign}\left(\sum_{i=1}^{n} y_i [\mathbf{x}_i \text{ among } k \text{ nearest neighbors of } \mathbf{x}]\right)$$



Demo: k-NN

Nearest-neighbor classifiers

- For data point ***x***, predict majority of labels of *k* nearest neighbors

$$y = \text{sign}\left(\sum_{i=1}^{n} y_i [\mathbf{x}_i \text{ among } k \text{ nearest neighbors of } \mathbf{x}]\right)$$

- How to choose *k*?
  - Cross-validation! ☺

K-NN vs. Kernel Perceptron

- k-Nearest Neighbor:

$$y = \text{sign}\left(\sum_{i=1}^{n} y_i [\mathbf{x}_i \text{ among } k \text{ nearest neighbors of } \mathbf{x}]\right)$$

- Kernel Perceptron:

$$y = \text{sign}\left(\sum_{i=1}^{n} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x})\right)$$

Comparison: k-NN vs Kernelized Perceptron

Method	*k-NN*	*Kernelized Perceptron*
Advantages	No training necessary	Optimized weights can lead to improved performance Can capture „global trends“ with suitable kernels Depends on „wrongly classified“ examples only
Disadvantages	Depends on all data ➔ inefficient	Training requires optimization