ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

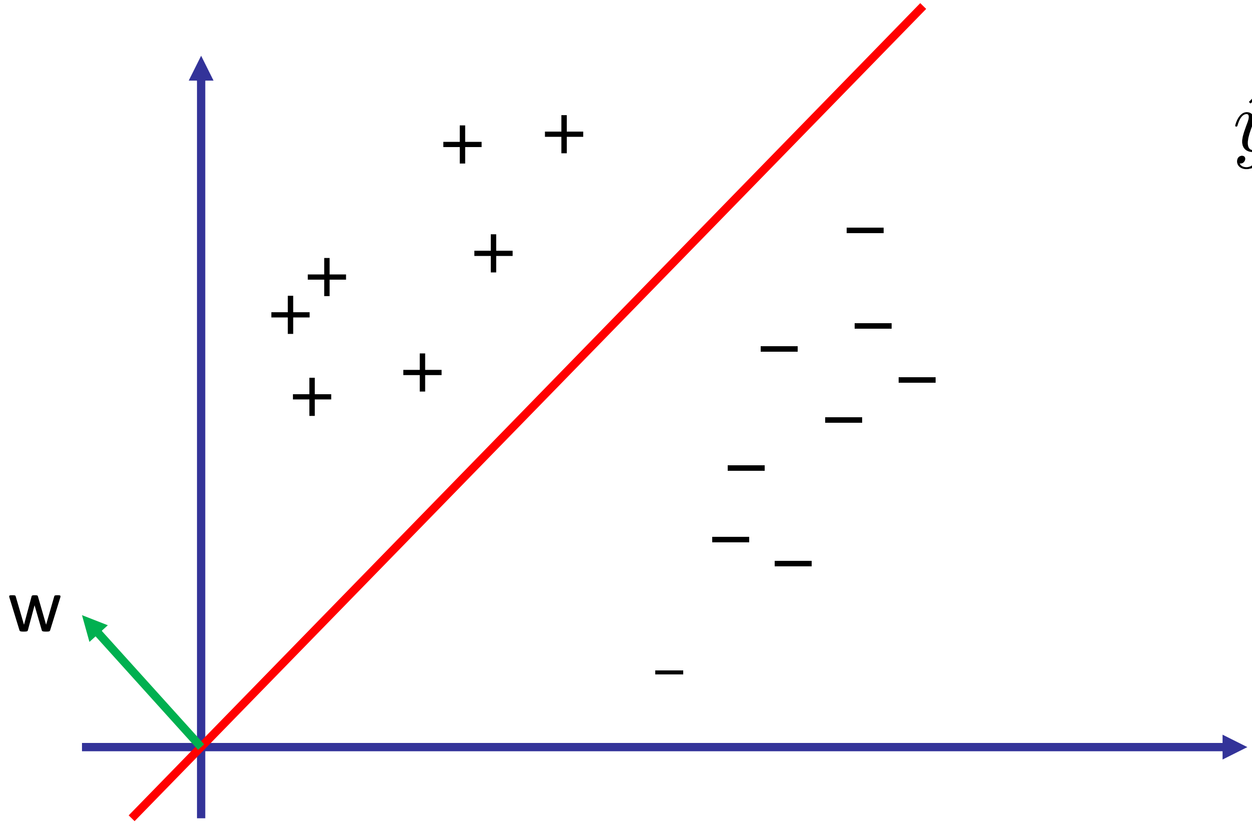
inf | Informatik Computer Science

Introduction to Machine Learning

Non-linear prediction with kernels

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

Recall: Linear classifiers



$$\hat{y} = \mathrm{sign}(\mathbf{w}^T\mathbf{x})$$

Recall: The Perceptron problem

- Solve $$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell_P(\mathbf{w}; \mathbf{x}_i, y_i)$$

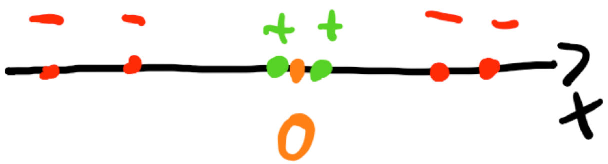
  where $$\ell_P(\mathbf{w}; y_i, \mathbf{x}_i) = \max(0, -y_i \mathbf{w}^T \mathbf{x}_i)$$

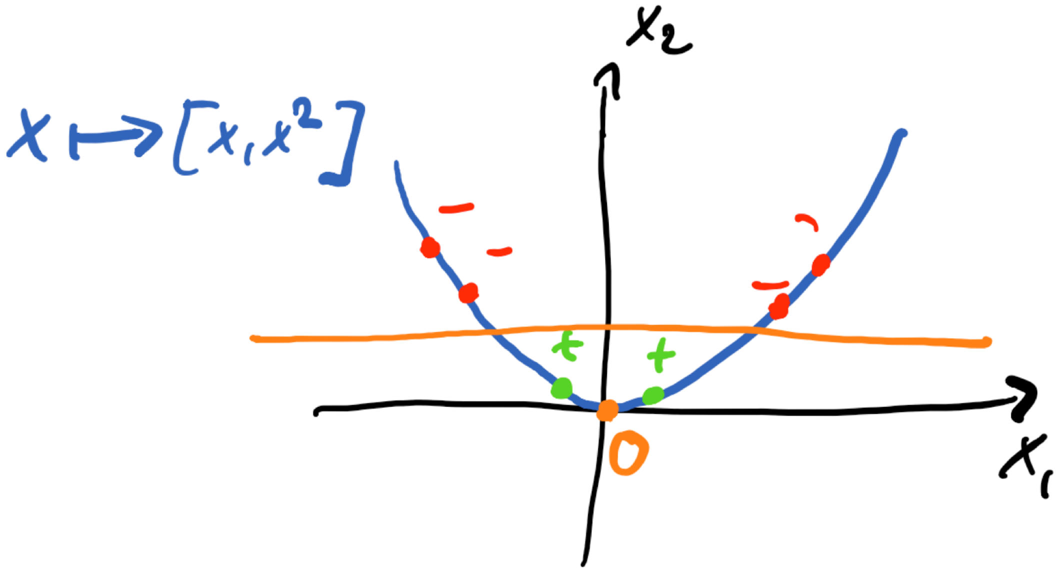
- Optimize via Stochastic Gradient Descent

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

$x_1 = x,\ x_2 = x^2$





Recall: linear regression for polynomials

- We can fit non-linear functions via linear regression, using nonlinear features of our data (basis functions)

$$f(\mathbf{x}) = \sum_{i=1}^{d} w_i \phi_i(\mathbf{x})$$

- For example: polynomials (in 1-D)

$$f(x) = \sum_{i=0}^{m} w_i x^i$$

Polynomials in higher dimensions

- Suppose we wish to use polynomial features, but our input is higher-dimensional
- Can still use monomial features
- **Example**: Monomials in 2 variables, degree = 2

$$x = [x_1, x_2] \mapsto \phi(x) = [x_1^2, x_2^2, x_1 \cdot x_2]$$

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)

Revisiting the Perceptron/SVM

- **Fundamental insight**: Optimal hyperplane lies in the span of the data

for some $\alpha_{1:n} \in \mathbb{R}^n$

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

- **(Handwavy) proof**: (Stochastic) gradient descent starting from 0 constructs such a representation

Perceptron: $\mathbf{w}_{t+1} = \mathbf{w}_t + \eta_t y_t \mathbf{x}_t \, [y_t \mathbf{w}_t^T \mathbf{x}_t < 0]$

SVM: $\mathbf{w}_{t+1} = \mathbf{w}_t (1 - 2\lambda\eta_t) + \eta_t y_t \mathbf{x}_t \, [y_t \mathbf{w}_t^T \mathbf{x}_t < 1]$

- **More abstract proof**: Follows from the „representer theorem“ (not discussed here)

Reformulating the Perceptron

$$(**) \quad \hat{w} \in \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \underbrace{\sum_{i=1}^{n} \max(0, -y_i w^T x_i)}_{(*)} \qquad \text{Ansatz: } \hat{w} = \sum_{j=1}^{n} \alpha_j y_j x_j$$

$$(*) = \sum_{i=1}^{n} \max\left(0, -y_i \left(\sum_{j=1}^{n} \alpha_j y_j x_j\right)^T x_i\right)$$

$$= \sum_{i=1}^{n} \max\left(0, -y_i \sum_{j=1}^{n} \alpha_j y_j (x_j^T x_i)\right)$$

$$(**) \equiv \hat{\alpha} \in \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{n} \max\left(0, -y_i \sum_{j=1}^{n} \alpha_j y_j (x_j^T x_i)\right)$$

Advantage of reformulation

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

- **Key observation**: Objective only depends on inner products of pairs of data points
- Thus, we can implicitly work in high-dimensional spaces, as long as we can do inner products efficiently

$$\mathbf{x} \mapsto \phi(\mathbf{x})$$

$$\mathbf{x}^T \mathbf{x}' \mapsto \phi(\mathbf{x})^T \phi(\mathbf{x}') =: k(\mathbf{x}, \mathbf{x}')$$

„Kernels = *efficient* inner products“

- Often, $k(\mathbf{x}, \mathbf{x}')$ can be computed much more efficiently than $\phi(\mathbf{x})^T\phi(\mathbf{x}')$

- Simple example: Polynomial kernel in degree 2

$$x \mapsto \phi(x) := [x_1^2, x_2^2, \sqrt{2}\, x_1 x_2]$$

$\in \mathbb{R}^2$

$$(2+10) \gg (1+3)$$

$$\phi(x)^T\phi(x') = x_1^2 \cdot x_1'^2 + x_2^2 \cdot x_2'^2 + 2\, x_1 x_2 x_1' x_2'$$

$$= (x_1 \cdot x_1' + x_2 x_2')^2$$

$$= (x^T x')^2 = k(x, x')$$

Naive: $\phi(x)^T\phi(x')$

\# +: 2

\# $\cdot$: 3+3+4 = 10

using kernel:

\# +: 1

\# $\cdot$: 3

Polynomial kernels (degree 2)

- Suppose $\mathbf{x} = [x_1, \ldots, x_d]^T$ and $\mathbf{x}' = [x'_1, \ldots, x'_d]^T$
- Then $(\mathbf{x}^T\mathbf{x}')^2 = \left(\sum_{i=1}^{d} x_i x'_i\right)^2 = \sum_{i=1}^{d} x_i^2 x_i'^2 + 2 \sum_{1 \le i < j \le d} x_i x'_i x_j x'_j$

$= \phi(x)^T \phi(x')$

for $\phi(x) := \underbrace{[x_1^2 \ldots x_d^2, \sqrt{2}\, x_1 x_2, \sqrt{2}\, x_1 x_3 \ldots \sqrt{2}\, x_{d-1} x_d]}_{\Theta(d^2)}$

$\Rightarrow \Theta(d^2) \to \Theta(d)$

Polynomial kernels: Fixed degree

- The kernel $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^m$
  implicitly represents all monomials of degree *m*

$x_1^m, x_2^m, \ldots x_d^m, x_1^{m-1} x_2, \ldots, x_1^{m-1} \cdot x_d, \ldots, x_1 \cdots x_m, \ldots, x_{d-m+1} \cdots x_d$

Monomials of degree m in d variables

$$\binom{d+m-1}{m} = O(d^m)$$

- How can we get monomials up to order *m*?

Polynomial kernels

$\left([1;x]^T[1;x']\right)^m$

$=$

- The polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T\mathbf{x}')^m$ implicitly represents all monomials of up to degree *m*

$1, x_1, x_2 \ldots x_d, x_1^2, x_2^2, \ldots x_d^2, x_1 x_2 \ldots x_{d-1} x_d, \ldots$

$\ldots$

Monomials of degree up to m in d variables?

$\rightarrow \binom{d+m}{m}$

- Representing the monomials (and computing inner product explicitly) is *exponential* in *m*!!

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

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

Derivation: Kernelized Perceptron

Training

$w_0 \leftarrow 0$

For $t = 1, 2, \dots$

Sample $(x_i, y_i) \sim D$

If $(y_i w^T x_i) > 0$

$w_{t+1} \leftarrow w_t$

else

$w_{t+1} \leftarrow w_t + \eta_t y_i x_i$

$w_t = \sum_{j=1}^{n} \alpha_{t,j} y_j x_j$

$\alpha_0 \leftarrow 0$

for $t = 1 \dots$

Sample $(x_i, y_i) \sim D$

if $\left(y_i \sum_{j=1}^{n} \alpha_j y_j \underbrace{(x_j^T x_i)}_{k(x_i, x_j)}\right) > 0$

$\alpha_{t+1} \leftarrow \alpha_t$

else

$\alpha_{t+1} \leftarrow \alpha_t$

$\alpha_{t+1,i} \leftarrow \alpha_{t+1,i} + \eta_t$

Prediction:

$w^T x \rightsquigarrow \sum_{j=1}^{n} \alpha_j y_j \underbrace{(x_j^T x_i)}_{k(x_i, x_j)}$

Kernelized Perceptron

Training

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

Prediction

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

Demo: Kernelized Perceptron

Questions

- What are valid kernels?
- How can we select a good kernel for our problem?
- Can we use kernels beyond the perceptron?
- Kernels work in very high-dimensional spaces. Doesn‘t this lead to overfitting?

Properties of kernel functions

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

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

→ *k* must be symmetric!

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

→ 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) $\forall x \in \mathbb{R}^n : \quad x^T M x \geq 0$

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

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

and $U^T U = I = UU^T$

$U = (u_1 | \dots | u_d)$ s.t. $M u_i = \lambda u_i$

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

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

$\square$

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 $\Phi = \left(\phi(x_1) \mid \ldots \mid \phi(x_n) \right)$

Sps: $x \in \mathbb{R}^n$: $x^T K x = \underbrace{(x^T \Phi^T)}_{v^T} \underbrace{(\Phi x)}_{v} = v^T v \geq 0$ $\square$

Semi-definite matrices ➔ kernels

- Suppose the data space $X=\{1,\dots,n\}$ 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 $\mathbf{K}_{i,j} = \phi(i)^T \phi(j)$

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

and $\lambda_i \geq 0 \quad \forall i$

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

$\phi : i \mapsto \phi_i$

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

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

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 $\mathbb{R}^n$ and $k : X \times X \to \mathbb{R}^n$ 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

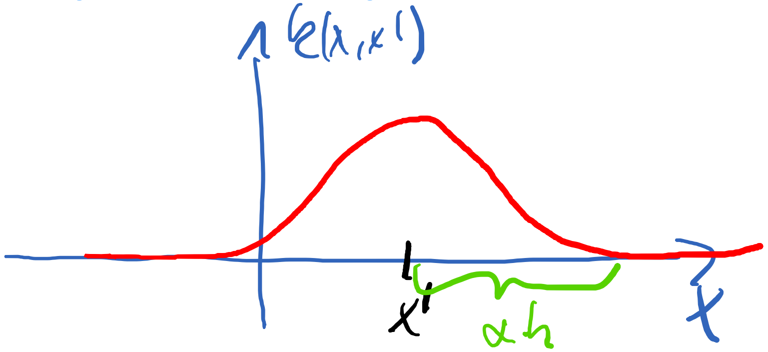
  $$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

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

"Bandwidth"/ Length scale parameter



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