ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

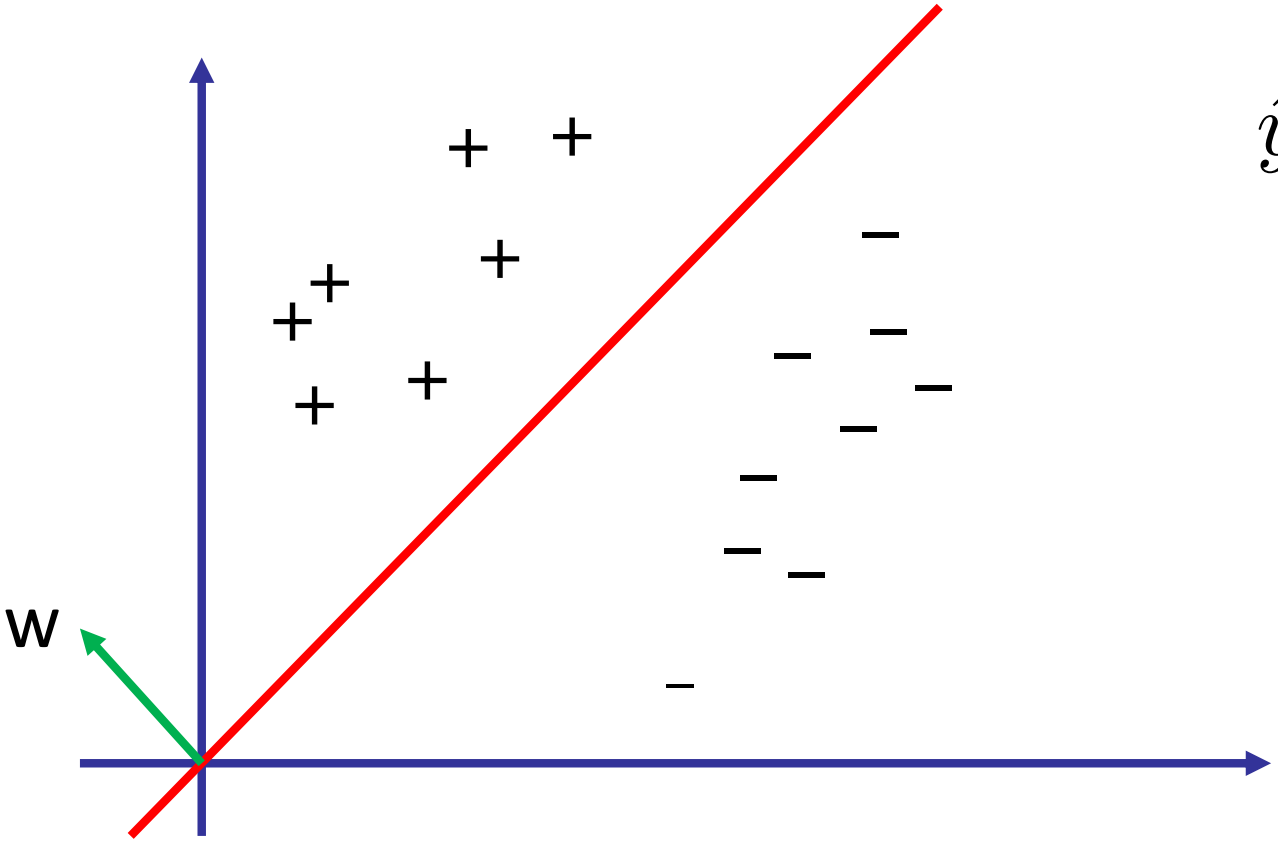
inf | Informatik
Computer Science

Introduction to Machine Learning

Non-linear prediction with kernels

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

Recall: Linear classifiers



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

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

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)

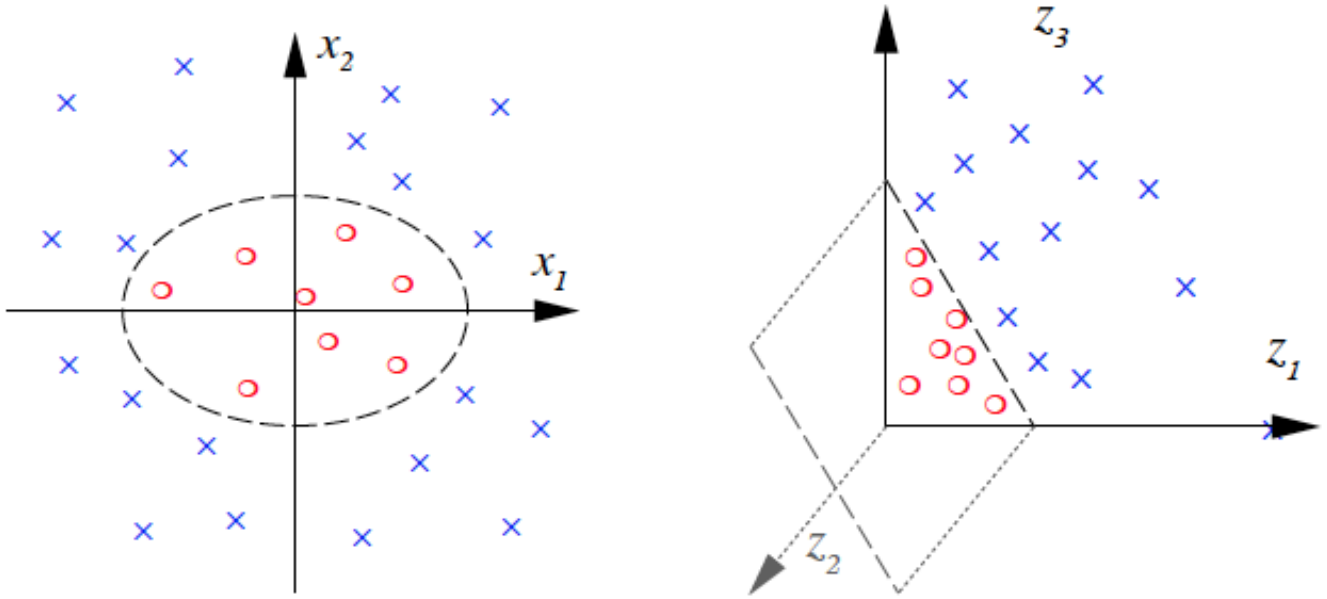
$$f(x) = \sum_{i=0}^{m} w_i x^i$$

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

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

$\Phi(x)$



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 lives in the span of the data

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

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

▽ Regularization ▽ Loss function

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

Reformulating the Perceptron

$$(**) \quad \hat{w} \in \operatorname*{argmin}_{w \in \mathbb{R}^d} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \max(0, -y_i w^T x_i)}_{(*)}$$

Ansatz: $w = \sum_{i=1}^{n} y_i \alpha_i x_i$

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

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

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

$w \in \mathbb{R}^d$

Advantage of reformulation

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

- **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

$$x \mapsto \phi(x) := [x_1^2, x_2^2, \sqrt{2}\, x_1 x_2]$$
$$\in \mathbb{R}^2 \qquad \in \mathbb{R}^3$$

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

# + :	2
# • :	3+3+4 = 10
# + :	1
# • :	3

(2+10) >> (1+3) :)

Polynomial kernels (degree 2)

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

$O(d)$

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

for $\phi(x) = [x_1^2 \ldots x_d^2, \sqrt{2}x_1x_2, \ldots \sqrt{2}x_1x_d, \ldots \sqrt{2}x_{d-1}x_d]$

$O(d^2)$

Polynomial kernels: Fixed degree

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

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

$\ldots \quad x_{d-m+1} \cdot \ldots \cdot x_d$

Monomials of degree $= m$ is $\binom{d+m-1}{d} \in O(d^m)$

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

Polynomial kernels

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

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

$\vdots$

$x_{d-m+1}\ \dots\ x_d$

Monomials up to degree m is $\binom{d+m}{m}$

- 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 $t = 0, 1, \ldots$

  Sample $(x_i, y_i) \sim D$

  If $(y_i \, w_t^T x_i) \geq 0$

    $w_{t+1} \leftarrow w_t$

  else

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

Kernelize

$\alpha_0 \leftarrow 0$

For $t = 0, 1, \ldots$

  Sample $(x_i, y_i) \sim D$

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

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

  else

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

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

Testing

$$\text{sign}(w^T x) = \text{sign}\left[\sum_{j=1}^{n} \alpha_j y_j \underbrace{(x_j^T x)}_{k(x_j, x)} \right]$$

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

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 $k : X \times X \rightarrow \mathbb{R}$
- Can we use any function?

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

➔ *k* must be symmetric!

$\forall x, x' \in X: \; k(x, x') = \phi(x)^\top \phi(x') = \phi(x')^\top \phi(x) = k(x', x)$

➔ 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

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

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

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