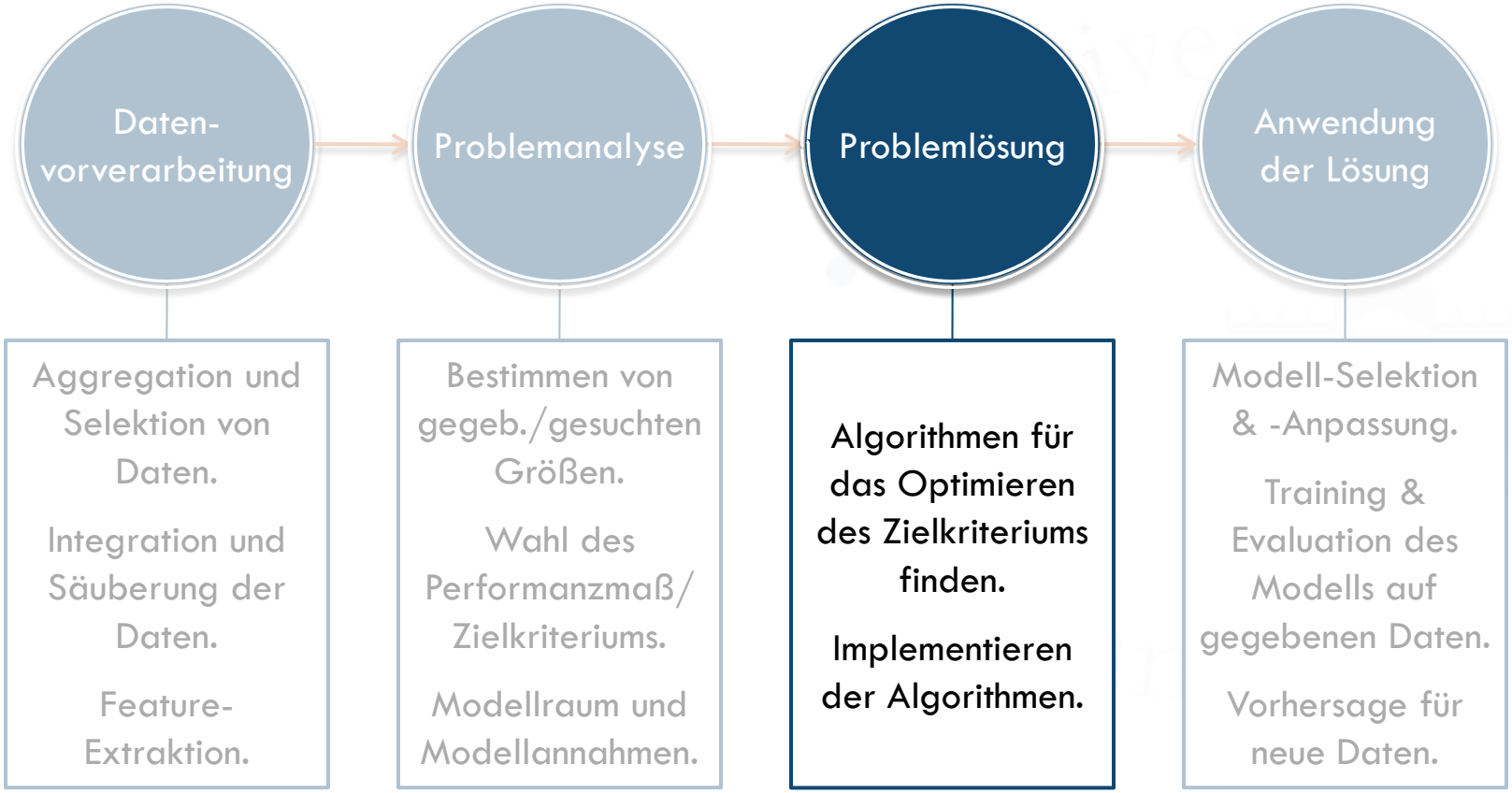
INTELLIGENTE DATENANALYSE IN MATLAB

Überwachtes Lernen: Nicht-lineare Modelle

Michael Brückner/Tobias Scheffer

Überblick

- Schritte der Datenanalyse:



Überwachtes Lernen

Problemstellung

- Gegeben: Trainingsdaten mit bekannten Zielattributen (gelabelte Daten).
- Eingabe: Instanz (Objekt, Beispiel, Datenpunkt, Merkmalsvektor) = Vektor mit Attribut-Belegungen.
- Ausgabe: Belegung des/der Zielattribut(e).
  - Klassifikation: Nominaler Wertebereich des Zielattributs.
  - Ordinale Regression: Ordinaler Wertebereich des Zielattributs.
  - Regression: Numerischer Wertebereich des Zielattributs.
- Gesucht: Modell $f : \mathbf{x} \mapsto y$.

Überwachtes Lernen

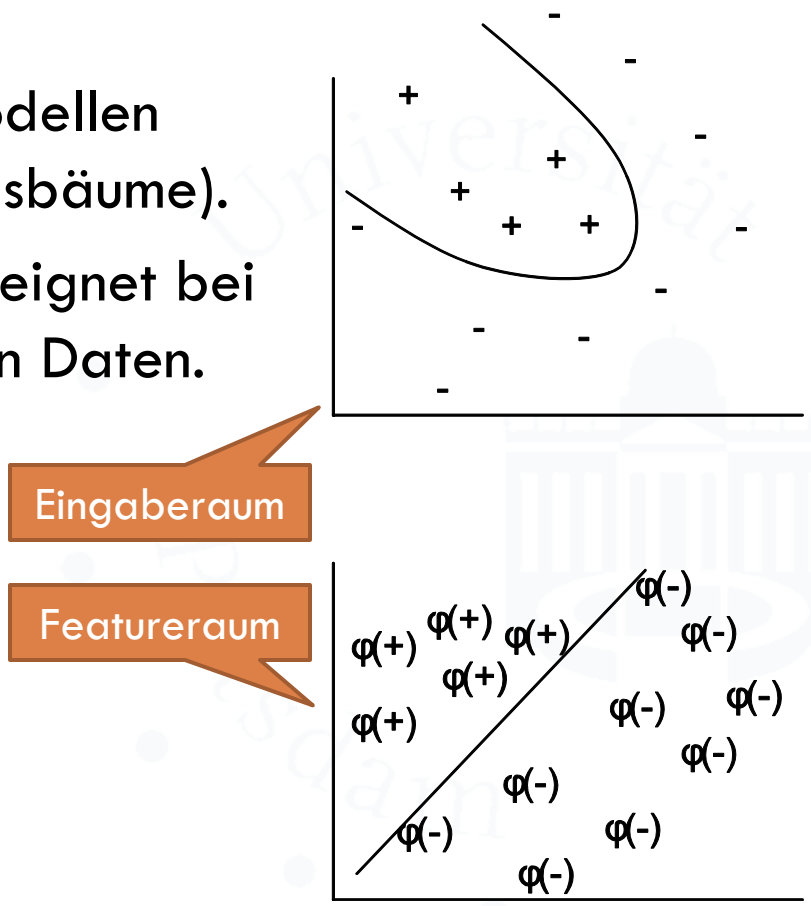
Arten von Modellen

- Entscheidungsbäume/Regelsysteme:
  - Klassifikations-, Regressions-, Modellbaum.
- Lineare Modelle:
  - Trennebenen, Regressionsgerade.
- Nicht-lineare Modelle, linear in den Parametern:
  - Nicht-lineare Datentransformation + lineares Modell.
  - Kernel-Modell.
  - Probabilistisches Modell.
- Nicht-lineare Modelle, nicht-linear in den Parametern:
  - Neuronales Netz.

Kernel-Modelle

Motivation

- Problem:
  - Suche nach nicht-linearen Modellen schwierig (siehe Entscheidungsbäume).
  - Lineare Modelle aber nur geeignet bei (nahezu) linear separierbaren Daten.
- Idee:
  - Beispiele in anderen Raum abbilden, in dem sie linear separierbar sind.
  - Lineares Modell in diesem Raum finden.



Kernel-Modelle

Ansatz

- Abbildung von m-dimensionalen Eingaberaum in k-dimensionalen Featureraum:

$$\varphi : \mathbb{R}^m \rightarrow \mathbb{R}^k \quad \text{mit} \quad \varphi(\mathbf{x}) = [\varphi_1(\mathbf{x}) \quad \varphi_2(\mathbf{x}) \quad \cdots \quad \varphi_k(\mathbf{x})]^{\mathrm{T}} = \dot{\mathbf{x}}$$

Beispiel:

$$\varphi\left(\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} \varphi_1(\mathbf{x}) = & x_1 \\ \varphi_2(\mathbf{x}) = & x_2 \\ \varphi_3(\mathbf{x}) = & x_1 x_2 \\ \varphi_4(\mathbf{x}) = & 1 \end{bmatrix} = \dot{\mathbf{x}}$$

Kernel-Modelle

Ansatz

- Hyperebene $H_{\mathbf{w}}$ im Featureraum ist durch Normalenvektor $\mathbf{w}$ gegeben:

$$H_{\mathbf{w}} = \{\mathbf{x} \mid f(\mathbf{x}) = \varphi(\mathbf{x})^{\mathrm{T}} \mathbf{w} = 0\}$$

- Entscheidungsfunktion für Klassifikation:

$$y(\mathbf{x}) = sign(f(\mathbf{x}))$$

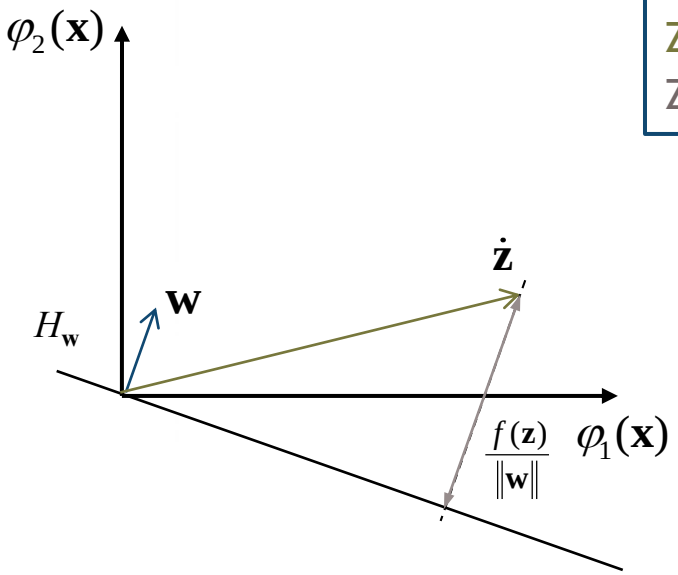
- Entscheidungsfunktion für Regression:

$$y(\mathbf{x}) = f(\mathbf{x})$$

Kernel-Modelle

Ansatz

- Hyperebene $H_{\mathbf{w}} = \{\mathbf{x} \mid f(\mathbf{x}) = \varphi(\mathbf{x})^{\mathrm{T}} \mathbf{w} = 0\}$:



Normalenvektor $\mathbf{w}$

Zu klassifizierender Punkt $\varphi(\mathbf{z}) = \begin{bmatrix} \varphi_1(\mathbf{z}) \\ \varphi_2(\mathbf{z}) \end{bmatrix} = \dot{\mathbf{z}}$

Zielfunktionswert $f(\mathbf{z})$

Kernel-Modelle

Ansatz

- Gegeben: n Trainingsinstanzen $\mathbf{x}_i$ mit Zielattribut y_i.

$$\mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_n] = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \qquad \mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_n]$$

- Gesucht: Parametervektor $\mathbf{w}$ der Klassifikations-/ Regressionsfunktion $f(\mathbf{x}) = \varphi(\mathbf{x})^{\mathrm{T}} \mathbf{w}$.

- Aber: Featureraum (Dimensionalität k) evtl. sehr groß $\Rightarrow$ viele freie Parameter $\mathbf{w} = [w_1 \quad \cdots \quad w_k]^{\mathrm{T}} \in \mathbb{R}^k$.

Kernel-Modelle

Representer Theorem

- *Representer Theorem*: Jeder Gewichtsvektor $\mathbf{w}$ lässt sich als Linearkombination der Punkte $\varphi(\mathbf{x}_i)$ darstellen:

$$\forall \mathbf{w}\ \exists \boldsymbol{\alpha} \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i \varphi(\mathbf{x}_i) \quad \Rightarrow \quad f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \underbrace{\varphi(\mathbf{x}_i)^{\mathrm{T}} \varphi(\mathbf{x})}_{k(\mathbf{x}_i, \mathbf{x})}$$

- Skalarprodukt $k(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x})^{\mathrm{T}} \varphi(\mathbf{x}')$ misst Ähnlichkeit zwischen $\mathbf{x}$ und $\mathbf{x}'$ im Featureraum.

- Statt k-dimensionalen (primalen) Gewichtsvektor $\mathbf{w}$, n-dimensionalen (dualen) Gewichtsvektor $\boldsymbol{\alpha}$ betrachten.

Kernel-Modelle

Kernel-Funktion

- Wozu Mapping $\varphi(\mathbf{x})$ explizit angeben?
  - Jedes Ähnlichkeitsmaß kann als Kernel k verwendet werden!

- Beispiele für Kernel-Funktionen:
  - Linearer Kernel: $k_{lin}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'$
  - Polynomieller Kernel: $k_p(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^{\mathrm{T}}\mathbf{x}' + 1)^p$
  - *Radial-Basis-Function*-Kernel: $k_{rbf}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}'\|_2^2\right)$
  - String-Kernel: Editierdistanz
  - Graph-Kernel.

Kernel-Modelle

Kernel-Matrix

- Kernel-Matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^{\mathrm{T}} \varphi(\mathbf{x}_j)$ definiert Ähnlichkeitsmatrix zwischen allen Trainingsbeispielen $\mathbf{x}_i$.
- Eigenschaften der Kernel-Matrix:
  - Symmetrisch: $\mathbf{K} = \mathbf{K}^{\mathrm{T}}$
  - Positiv semidefinit: $\exists \mathbf{\Phi} \in \mathbb{R}^{m \times n} \quad \mathbf{K} = \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi}$
- Kernel-Komposition:

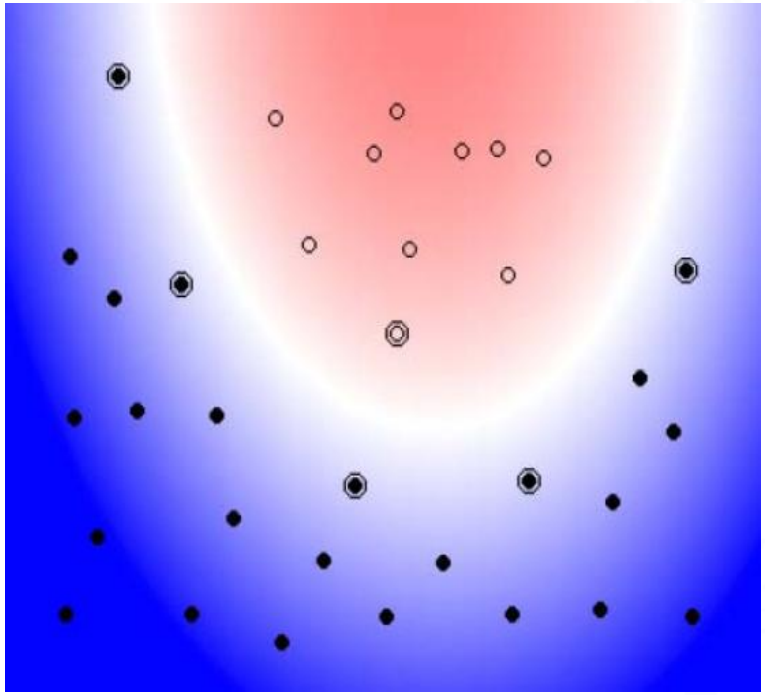
$$\left.\begin{aligned} \mathbf{K}' &= c + \mathbf{K} \\ \mathbf{K}' &= c\mathbf{K} \\ \mathbf{K}' &= \mathbf{K}^{(1)} + \mathbf{K}^{(2)} \\ \mathbf{K}' &= \mathbf{K}^{(1)} \circ \mathbf{K}^{(2)} \end{aligned}\right\} \Rightarrow \quad \begin{aligned} K'_{ij} &= K_{ij}^{p} \\ K'_{ij} &= e^{K_{ij}} \\ K'_{ij} &= \frac{K_{ij}}{\sqrt{K_{ii} K_{jj}}} \end{aligned}$$

Mercer-Bedingung

Kernel-Modelle

Beispiel: Polynomieller Kernel

- Zwei-dimensionaler polynomieller Kernel und zwei-dimensionaler Eingaberaum:



Kernel-Modelle

Beispiel: Polynomieller Kernel

- Zwei-dimensionaler polynomieller Kernel und zwei-dimensionaler Eingaberaum:

$$k(\mathbf{x},\mathbf{x}') = \varphi(\mathbf{x})^{\mathrm{T}}\varphi(\mathbf{x}') = (\mathbf{x}^{\mathrm{T}}\mathbf{x}'+1)^2 = \left(\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} + 1 \right)^2$$

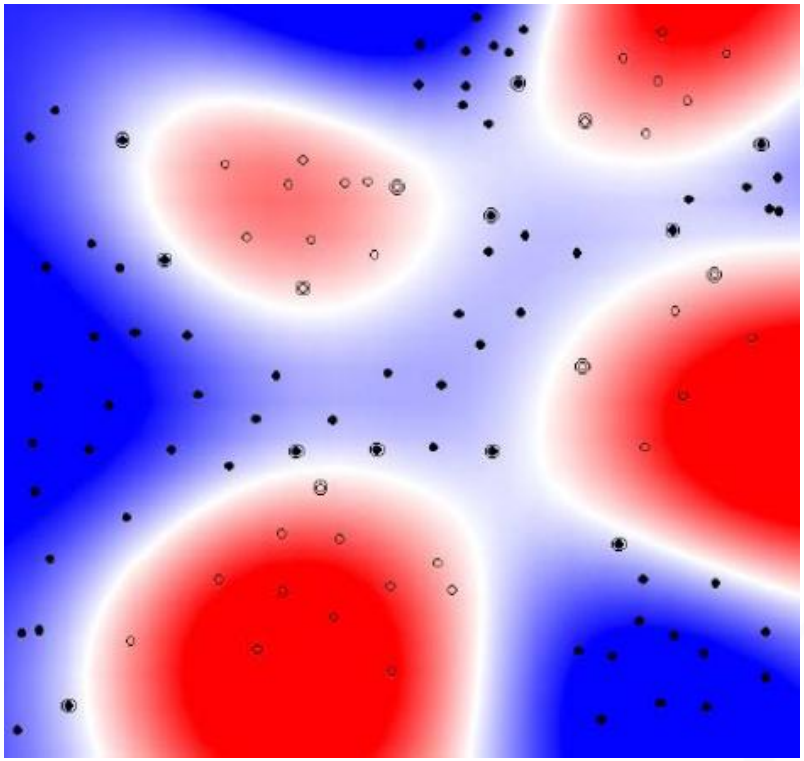
$$= (x_1x_1' + x_2x_2' + 1)^2 = (x_1x_1')^2 + (x_2x_2')^2 + 2x_1x_1'x_2x_2' + 2x_1x_1' + 2x_2x_2' + 1$$

$$= \underbrace{\begin{bmatrix} x_1^2 & x_2^2 & \sqrt{2}x_1x_2 & \sqrt{2}x_1 & \sqrt{2}x_2 & 1 \end{bmatrix}}_{\varphi(\mathbf{x})^{\mathrm{T}}} \begin{bmatrix} x_1'^2 \\ x_2'^2 \\ \sqrt{2}x_1'x_2' \\ \sqrt{2}x_1' \\ \sqrt{2}x_2' \\ 1 \end{bmatrix}$$

Kernel-Modelle

Beispiel: RBF-Kernel

- RBF-Kernel und zwei-dimensionaler Eingaberaum:



Kernel-Modelle

Beispiel: RBF-Kernel

- RBF-Kernel und zwei-dimensionaler Eingaberaum:
  - Zugehörige Mapping-Funktion $\varphi(\mathbf{x})$ hat theoretisch unendlich viele Dimensionen.
  - n Trainingsdaten liegen in n-dimensionalen Unterraum des Featureraums.

    $\Rightarrow$ Daten im Featureraum immer linear separierbar!
  - Eine Mapping-Funktion für diesen n-dimensionalen Unterraum kann explizit angegeben werden.

Kernel-Modelle

Besonders geeignet …

- Für kleine – mittlere Klassifikations- & Regressions-probleme.
- Falls Anzahl Beispiele n kleiner der Anzahl Attribute m.
- Falls Daten nicht linear separierbar; falls Lernproblem sehr schwer.
- Falls Interpretierbarkeit der Entscheidung nicht notwendig.
- Für komplexe Daten (Strukturen, Sequenzen usw.) mit bekanntem Ähnlichkeitsmaß (Kernel).

Lernen von Kernel-Modellen

Problemstellung

- Ziel: Minimierung von empirischen Verlust und Regularisierer im Featureraum:

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{n} l(f_{\boldsymbol{\alpha}}(\mathbf{x}_i), y_i) + \Omega(\boldsymbol{\alpha}) \qquad f_{\boldsymbol{\alpha}}(\mathbf{x}_i) = \sum_{j=1}^{n} \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{K}_i \boldsymbol{\alpha}$$

mit Kernel-Matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

i-te Zeile der Kernel-Matrix

- Lösung analog zu linearen Modellen; zwei Ansätze:
  - In Zielfunktion $\mathbf{w}$ durch $\sum_i \alpha_i \varphi(\mathbf{x}_i)$ ersetzen und bzgl. $\boldsymbol{\alpha}$ lösen (z.B. Kernel Ridge Regression).
  - Im primalen Algorithmus $\mathbf{w}$ durch $\sum_i \alpha_i \varphi(\mathbf{x}_i)$ ersetzen (z.B. Kernel Perceptron).

Lernen von Kernel-Modellen

Beispiel: Kernel Ridge Regression ($y_i \in \mathrm{R}$)

- Verlustfunktion: $l_s(f_{\boldsymbol{\alpha}}(\mathbf{x}_i), y_i) = \left|\mathbf{K}_i\boldsymbol{\alpha} - y_i\right|^2$
- Regularisierer: $\Omega_2(\boldsymbol{\alpha}) = \lambda\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}\boldsymbol{\alpha}$
- Analytische Lösung:

$$L(\mathbf{w}) = \sum_{i=1}^{n}\left|\mathbf{K}_i\boldsymbol{\alpha} - y_i\right|^2 + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}\boldsymbol{\alpha}$$

$$= \left\|\mathbf{K}\boldsymbol{\alpha} - \mathbf{y}\right\|_2^2 + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}\boldsymbol{\alpha}$$

$$= (\mathbf{K}\boldsymbol{\alpha} - \mathbf{y})^{\mathrm{T}}(\mathbf{K}\boldsymbol{\alpha} - \mathbf{y}) + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}\boldsymbol{\alpha}$$

$$= \boldsymbol{\alpha}^{\mathrm{T}}(\mathbf{K} + \lambda\mathbf{I})\mathbf{K}\boldsymbol{\alpha} - 2\mathbf{y}^{\mathrm{T}}\mathbf{K}\boldsymbol{\alpha} + \mathbf{y}^{\mathrm{T}}\mathbf{y}$$

Quadratische OA ohne Nebenbedingungen

$$\nabla L(\boldsymbol{\alpha}) = 2(\mathbf{K} + \lambda\mathbf{I})\mathbf{K}\boldsymbol{\alpha} - 2\mathbf{K}\mathbf{y} = ((\mathbf{K} + \lambda\mathbf{I})\boldsymbol{\alpha} - \mathbf{y})\mathbf{K} = 0$$

$$\boldsymbol{\alpha} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$$

Nach α ableiten und Ableitung Null setzen

Lernen von Kernel-Modellen

Beispiel: Kernel Perceptron ($y_i \in \{-1,+1\}$)

- Verlustfunktion: $l_p(f_{\mathbf{w}}(\mathbf{x}_i), y_i) = \begin{cases} -y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{w} & -y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{w} > 0 \\ 0 & -y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{w} \le 0 \end{cases}$
- Regularisierer: *const.*
- Primaler Algorithmus:

```
Perceptron(Instanzen (x_i, y_i))
  Setze w = 0
  DO
    FOR i = 1...n
      IF y_i x_i^T w < 0 THEN
        w = w + y_i x_i
  WHILE w geändert
  RETURN w
```

$$\mathbf{w} = \sum_i \alpha_i \varphi(\mathbf{x}_i)$$

Lernen von Kernel-Modellen

Beispiel: Kernel Perceptron ($y_i \in \{-1,+1\}$)

- Verlustfunktion: $l_p(f_{\boldsymbol{\alpha}}(\mathbf{x}_i), y_i) = \begin{cases} -y_i\varphi(\mathbf{x}_i)^{\mathrm{T}}\sum_j \alpha_j\varphi(\mathbf{x}_j) & -y_i\varphi(\mathbf{x}_i)^{\mathrm{T}}\sum_j \alpha_j\varphi(\mathbf{x}_j) > 0 \\ 0 & -y_i\varphi(\mathbf{x}_i)^{\mathrm{T}}\sum_j \alpha_j\varphi(\mathbf{x}_j) \le 0 \end{cases}$
- Regularisierer: *const.*
- Dualer Algorithmus:

```
KernelPerceptron(Instanzen (x_i, y_i))
  Setze α = 0
  DO
    FOR i = 1...n
      IF y_i φ(x_i)^T Σ_j α_j φ(x_j) < 0 THEN
        Σ_j α_j φ(x_j) = Σ_j α_j φ(x_j) + y_i φ(x_i)
  WHILE α geändert
  RETURN α
```

$$\mathbf{w} = \sum_i \alpha_i \varphi(\mathbf{x}_i)$$

Lernen von Kernel-Modellen

Beispiel: Kernel Perceptron ($y_i \in \{-1,+1\}$)

- Verlustfunktion: $l_p(f_{\boldsymbol{\alpha}}(\mathbf{x}_i), y_i) = \begin{cases} -y_i \mathbf{K}_i \boldsymbol{\alpha} & -y_i \mathbf{K}_i \boldsymbol{\alpha} > 0 \\ 0 & -y_i \mathbf{K}_i \boldsymbol{\alpha} \leq 0 \end{cases}$
- Regularisierer: *const.*
- Dualer Algorithmus:

```
KernelPerceptron(Instanzen (x_i, y_i))
  Setze α = 0
  DO
    FOR i = 1...n
      IF y_i K_i α < 0 THEN
        α_i = α_i + y_i
  WHILE α geändert
  RETURN α
```

$$\mathbf{K}_i \boldsymbol{\alpha} = \varphi(\mathbf{x}_i)^{\mathrm{T}} \mathbf{w} = \varphi(\mathbf{x}_i)^{\mathrm{T}} \sum_j \alpha_j \varphi(\mathbf{x}_j)$$

Lernen von Kernel-Modellen

Beispiel: Kernel SVM ($y_i \in \{-1,+1\}$)

- Verlustfunktion: $l_h(f_{\boldsymbol{\alpha}}(\mathbf{x}_i), y_i) = \begin{cases} 1 - y_i \mathbf{K}_i \boldsymbol{\alpha} & 1 - y_i \mathbf{K}_i \boldsymbol{\alpha} > 0 \\ 0 & 1 - y_i \mathbf{K}_i \boldsymbol{\alpha} \leq 0 \end{cases}$
- Regularisierer: $\Omega_2(\boldsymbol{\alpha}) = \frac{1}{2\lambda} \boldsymbol{\alpha}^{\mathrm{T}} \mathbf{K} \boldsymbol{\alpha}$
- Numerische Lösung des dualen Problems:

$$L(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\mathrm{T}} \mathbf{K} \boldsymbol{\alpha} - \mathbf{y}^{\mathrm{T}} \boldsymbol{\alpha} \quad \text{mit } \mathbf{1}^{\mathrm{T}} \boldsymbol{\alpha} = 0,\ 0 \leq y_i \alpha_i \leq \lambda$$

  - Lösen der dualen (quadratischen) OA mittels *QP-Solver*.

Lernen von Kernel-Modellen

Allgemein: Kernel RegERM

- Gegegeben:
  - Konvexe, ableitbare Verlustfunktion l mit Ableitung $l' = \frac{\partial l(z,y)}{\partial z}$.
  - Konvexer Regularisierer $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2$ mit Ableitung $\Omega'(\mathbf{w}) = \lambda\mathbf{w}$.
- Primaler Algorithmus:

RegERM(*Instanzen* $(\mathbf{x}_i, y_i)$)

Setze $k = 0, \mu^0 = 1, \mathbf{w}^0 = \mathbf{0}$

DO

$$\mathbf{g}^k = \sum_{i=1}^{n} l'(\mathbf{x}_i^{\mathrm{T}}\mathbf{w}^k, y_i)\mathbf{x}_i + \lambda\mathbf{w}^k$$

IF $k > 0$ THEN

$$\mu^k = \mu^{k-1}(\mathbf{g}^{k-1^{\mathrm{T}}}\mathbf{g}^{k-1}) / ((\mathbf{g}^{k-1} - \mathbf{g}^k)^{\mathrm{T}}\mathbf{g}^{k-1})$$

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \mu^k\mathbf{g}^k$$

$$k = k + 1$$

WHILE $\|\mathbf{w}^{k+1} - \mathbf{w}^k\| > \varepsilon$

RETURN $\mathbf{w}^{k+1}$

$$\mathbf{g}^k = \sum_i \nu_i^k \varphi(\mathbf{x}_i)$$

$$\mathbf{w}^k = \sum_i \alpha_i^k \varphi(\mathbf{x}_i)$$

Lernen von Kernel-Modellen

Allgemein: Kernel RegERM

- Gegegeben:
  - Konvexe, ableitbare Verlustfunktion l mit Ableitung $l' = \frac{\partial l(z,y)}{\partial z}$.
  - Konvexer Regularisierer $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2$ mit Ableitung $\Omega'(\mathbf{w}) = \lambda\mathbf{w}$.
- Dualer Algorithmus:

KernelRegERM(*Instanzen* $(\mathbf{x}_i, y_i)$)

Setze $k = 0, \mu^0 = 1, \boldsymbol{\alpha}^0 = \mathbf{0}$

DO

$$\sum_i \nu_i^k \varphi(\mathbf{x}_i) = \sum_i l'(\mathbf{K}_i \boldsymbol{\alpha}^k, y_i)\varphi(\mathbf{x}_i) + \lambda \sum_i \alpha_i^k \varphi(\mathbf{x}_i)$$

IF $k > 0$ THEN

$$\mu^k = \mu^{k-1}(\mathbf{v}^{k-1\mathrm{T}}\mathbf{K}\mathbf{v}^{k-1}) / ((\mathbf{v}^{k-1} - \mathbf{v}^k)^{\mathrm{T}}\mathbf{K}\mathbf{v}^{k-1})$$

$$\sum_i \alpha_i^{k+1}\varphi(\mathbf{x}_i) = \sum_i \alpha_i^k \varphi(\mathbf{x}_i) - \mu^k \sum_i \nu_i^k \varphi(\mathbf{x}_i)$$

$k = k + 1$

WHILE $\left\|\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k\right\| > \varepsilon$

RETURN $\boldsymbol{\alpha}^{k+1}$

$$\mathbf{g}^k = \sum_i \nu_i^k \varphi(\mathbf{x}_i)$$

$$\mathbf{w}^k = \sum_i \alpha_i^k \varphi(\mathbf{x}_i)$$

Lernen von Kernel-Modellen

Allgemein: Kernel RegERM

- Gegegeben:
  - Konvexe, ableitbare Verlustfunktion l mit Ableitung $l' = \frac{\partial l(z,y)}{\partial z}$.
  - Konvexer Regularisierer $\Omega(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2$ mit Ableitung $\Omega'(\mathbf{w}) = \lambda\mathbf{w}$.
- Dualer Algorithmus:

KernelRegERM(*Instanzen* $(\mathbf{x}_i, y_i)$)

Setze $k = 0, \mu^0 = 1, \boldsymbol{\alpha}^0 = \mathbf{0}$

DO

$\nu_i^k = l'(\mathbf{K}_i\boldsymbol{\alpha}^k, y_i) + \lambda\alpha_i^k \quad \forall i$

IF $k > 0$ THEN

$\mu^k = \mu^{k-1}(\mathbf{v}^{k-1^{\mathrm{T}}}\mathbf{K}\mathbf{v}^{k-1}) / ((\mathbf{v}^{k-1} - \mathbf{v}^k)^{\mathrm{T}}\mathbf{K}\mathbf{v}^{k-1})$

$\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k - \mu^k\mathbf{v}^k$

$k = k + 1$

WHILE $\|\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}\| > \varepsilon$

RETURN $\boldsymbol{\alpha}^k$

$$\mathbf{g}^k = \sum_i \nu_i^k \varphi(\mathbf{x}_i)$$

$$\mathbf{w}^k = \sum_i \alpha_i^k \varphi(\mathbf{x}_i)$$

Probabilistische Modelle

- Idee: Annahme über theoretische Verteilung (Generierungsprozess) der Daten.
- Ziel: Verteilungsparameter aus Daten schätzen.
- Ansatz: MAP-Schätzer für Modell-Parameter θ.

$$\theta_{MAP} = \arg\max_{\theta} p(\theta \mid \mathbf{X}, \mathbf{y}) = \arg\max_{\theta} p(\theta)\, p(\mathbf{X}, \mathbf{y} \mid \theta)$$

- Beispiele:
  - Naive Bayes.
  - Logistische Regression.
  - Ridge Regression.

Probabilistische Modelle

Besonders geeignet …

- Für kleine – sehr große Klassifikations- & Regressionsprobleme.
- Falls echte Wahrscheinlichkeiten benötigt werden.
- Falls (eingeschränkte) Interpretierbarkeit der Entscheidung notwendig.
- Falls Attributbelegungen fehlen.
- Falls Vorwissen über Datengenerierungsmodell vorhanden ist.

Lernen von Probabilistischen Modelle

Beispiel: Naive Bayes

- Verteilungsannahmen:
  - n unabhängig verteilte Datenvektoren $\mathbf{x}_i$.

$$p(\mathbf{X}, \mathbf{y} \mid \theta) = \prod_{i=1}^{n} p(\mathbf{x}_i, y_i \mid \theta) = \prod_{i=1}^{n} p(\mathbf{x}_i \mid y_i, \theta) p(y_i \mid \theta)$$

  - m unabhängig verteilte Attribute x_{ij} je Datenvektor $\mathbf{x}_i$.

$$p(\mathbf{x}_i \mid y_i, \theta) = \prod_{j=1}^{m} p(x_{ij} \mid y_i, \theta)$$

  - Modellierung der bedingten Wahrscheinlichkeiten durch theoretische Wahrscheinlichkeiten.

Lernen von Probabilistischen Modelle

Beispiel: Naive Bayes

- MAP-Schätzer für Naive Bayes:

$$\theta_{MAP} = \arg\max_{\theta} p(\theta) \prod_{i=1}^{n} \left(p(y_i \mid \theta) \prod_{j=1}^{m} p(x_{ij} \mid y_i, \theta) \right)$$

$$= \arg\max_{\theta} p(\theta) \left(\prod_{i=1}^{n} p(y_i \mid \theta) \right) \left(\prod_{j=1}^{m} \prod_{i=1}^{n} p(x_{ij} \mid y_i, \theta) \right)$$

$$= \arg\max_{\theta} \left(p(\theta^y) \prod_{i=1}^{n} p(y_i \mid \theta^y) \right) \left(\prod_{j=1}^{m} p(\theta^{x_j \mid y}) \prod_{i=1}^{n} p(x_{ij} \mid y_i, \theta^{x_j \mid y_i}) \right)$$

Klassen-Prior	Klassen-Likelihood	Klassenabhängiger Attribut-Prior	Klassenabhängige Attribut-Likelihood

Lernen von Probabilistischen Modelle

Beispiel: Logistische Regression

- Verteilungsannahmen:
  - n unabhängig verteilte Datenvektoren $\mathbf{x}_i$.

$$p(\mathbf{X}, \mathbf{y} \mid \theta) = \prod_{i=1}^{n} p(\mathbf{x}_i, y_i \mid \theta) = \prod_{i=1}^{n} p(\mathbf{x}_i \mid y_i, \theta) p(y_i \mid \theta)$$

  - m normal-verteilte Attribute x_{ij} mit $\theta = \{\boldsymbol{\mu}^y, \boldsymbol{\Sigma}\}$.

$$p(\mathbf{x}_i \mid y_i, \theta) = \mathrm{N}(\mathbf{x} \mid \boldsymbol{\mu}^y, \boldsymbol{\Sigma})$$

  - Umformung ergibt

Logistische Funktion

$$p(y_i \mid \mathbf{x}_i, \theta) = \frac{1}{1 + \exp(\mathbf{w}^{\mathrm{T}} \mathbf{x}_i + w_0)} = \sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}_i + w_0)$$

mit $\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)$ und $w_0 = \log \frac{n_+}{n_-} - \frac{1}{2} \boldsymbol{\mu}_+ \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_+ + \frac{1}{2} \boldsymbol{\mu}_- \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_-$.

Lernen von Probabilistischen Modelle

Beispiel: Logistische Regression

- MAP-Schätzer für Logistische Regression:

$$\theta_{MAP} = \arg\max_{\theta} p(\theta) \prod_{i=1}^{n} p(\mathbf{x}_i, y_i \mid \theta) = \arg\max_{\theta} p(\theta) \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \theta)$$

$$= \arg\max_{\theta} p(\mathbf{w}, w_0) \prod_{i=1}^{n} \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_0)^{y_i=1} (1 - \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_0))^{y_i=-1}$$

$$= \arg\max_{\theta} p(\mathbf{w}, w_0) \prod_{i=1}^{n} \sigma(y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_0)$$

$$= \arg\min_{\theta} \sum_{i=1}^{n} \log \frac{1}{\sigma(y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + w_0)} + \Omega(\mathbf{w}, w_0)$$

Konvexes *Logistic Loss*

Regularisierer

Lernen von Probabilistischen Modelle

Beispiel: Ridge Regression

- Verteilungsannahmen:
  - n unabhängig verteilte Datenvektoren $\mathbf{x}_i$.

$$p(\mathbf{X}, \mathbf{y} \mid \theta) = \prod_{i=1}^{n} p(\mathbf{x}_i, y_i \mid \theta) = \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \theta) p(\mathbf{x}_i \mid \theta)$$

  - n normal-verteilte Klassenlabel y_i mit $\theta = \{\mathbf{w}, \sigma\}$.

$$p(y_i \mid \mathbf{x}_i, \theta) = \mathrm{N}(y \mid \mathbf{w}^{\mathrm{T}} \mathbf{x}_i, \sigma^2)$$

  - Attributbelegung ist unabhängig von Modell-Parametern.

$$p(\mathbf{x}_i \mid \theta) = const.$$

  - Normal-verteilter Prior über Modell-Parameter.

$$p(\theta) = \mathrm{N}(\mathbf{w} \mid \mathbf{0}, \boldsymbol{\Sigma})$$

Lernen von Probabilistischen Modelle

Beispiel: Ridge Regression

- MAP-Schätzer für Ridge Regression:

$$\theta_{MAP} = \arg\max_{\theta} p(\theta) \prod_{i=1}^{n} p(\mathbf{x}_i, y_i \mid \theta) = \arg\max_{\theta} p(\theta) \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \theta)$$

$$= \arg\max_{\theta} \mathrm{N}(\mathbf{w} \mid \mathbf{0}, \boldsymbol{\Sigma}) \prod_{i=1}^{n} \mathrm{N}(y_i \mid \mathbf{w}^{\mathrm{T}} \mathbf{x}_i, \sigma^2)$$

$$= \arg\max_{\theta} \mathrm{N}\left(\mathbf{w} \mid \boldsymbol{\mu}_{\mathbf{w}}, \boldsymbol{\Sigma}_{\mathbf{w}}\right) = \boldsymbol{\mu}_{\mathbf{w}} = \mathbf{w}_{MAP}$$

mit $\boldsymbol{\mu}_{\mathbf{w}} = \frac{1}{\sigma^2} \boldsymbol{\Sigma}_{\mathbf{w}} \mathbf{X}\mathbf{y}$ und $\boldsymbol{\Sigma}_{\mathbf{w}} = \left(\frac{1}{\sigma^2} \mathbf{X}\mathbf{X}^{\mathrm{T}} + \boldsymbol{\Sigma}^{-1} \right)^{-1}$.

Zusammenfassung

- Kernel-Modelle geeignet für sehr schwere Klassifikations- & Regressionsprobleme.
  - Besonders geeignet falls viel mehr Attribute als Beispiele.
  - Falls (abstraktes) Ähnlichkeitsmaß zw. Beispielen bekannt.
- Lernen von Kernel-Modelle analog zu linearen Modellen:
  - Für viele lineare Modelle existieren Kernelisierte Varianten.
- Probabilistische Modelle liefern Wahrscheinlichkeiten, verlangen aber einschränkende Verteilungsannahmen.