Nonlinear Support Vector Machines

Seminar Statistik – Sommersemester 2021

Malte Voß

July 12, 2021

Überblick

- Wiederholung: Lineare SVM
- *feature space*
- Kernel Trick
- SVM und Gradient descent

Linear Support Vector Machines (SVM)

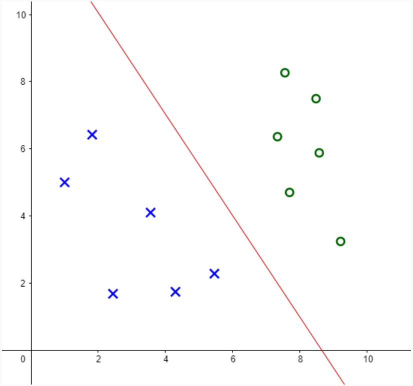


Figure 1: Linear separierbare Daten mit Hyperebene als Diskriminator

- nutze separierende Funktion $f(x) = \beta_0 + x^T\beta$

Linear SVM – Margin

- Hyperebene mit maximalen Margin wird gewählt

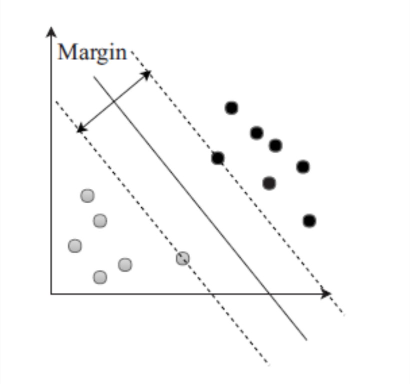


Figure 2: Margin veranschaulicht

- Minimierungsproblem: $\min ||\beta||^2$ mit $y_i(\beta_0 + x_i^T\beta) \geq +1$

Linear SVM – Minimierungsproblem

- Minimierungsproblem: - $\min ||\beta||^2$ mit $y_i(\beta_0 + x_i^T \beta) \geq +1$
- Lösungsverfahren: Lagrange Multipliers
- Wolfe-Dual: $\max_\alpha 1_n^T \alpha - \frac{1}{2}\alpha^T H \alpha$ mit Nebenbedingungen $\alpha_i \geq 0, \alpha^T y = 0$
  - $H_{ij} = y_i y_j (x_i^T x_j)$

Linear SVM – linearly non-separable data

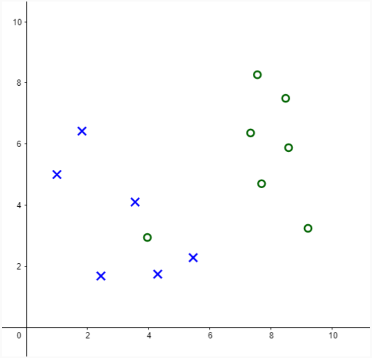
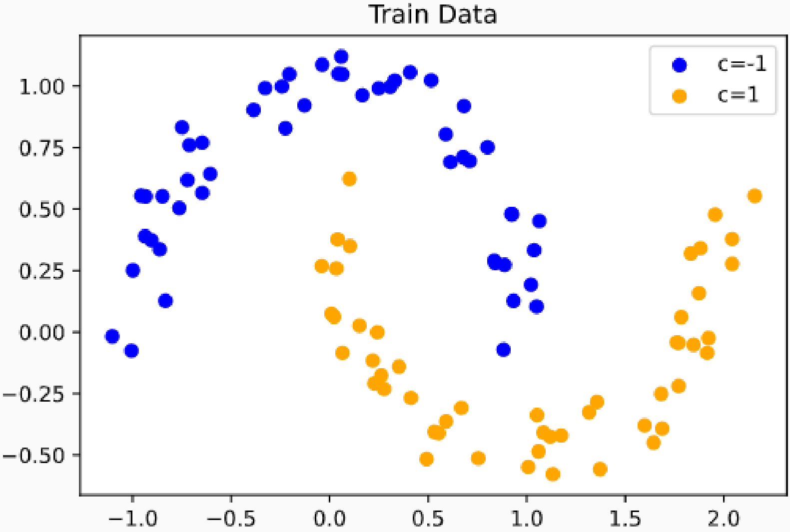


Figure 3: Linear nicht separierbare Daten

- Verletzung des Margin wird erlaubt
- Gleichung wird um slack-Variablen erweitert
- min $||\beta||^2 + C \sum_i \xi_i$ mit $y_i(\beta_0 + x_i^T \beta) \geq +1 - \xi$ und $\xi \geq 0$

Linearly non-separable data: Two Moons



Linearly non-separable data – Goal

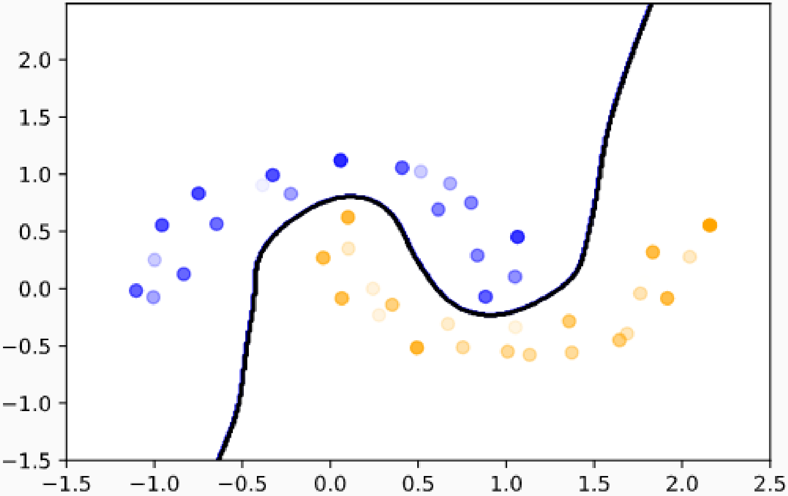


Figure 5: Data set: two moons with a separator

Nonlinear SVM – Basic Idea

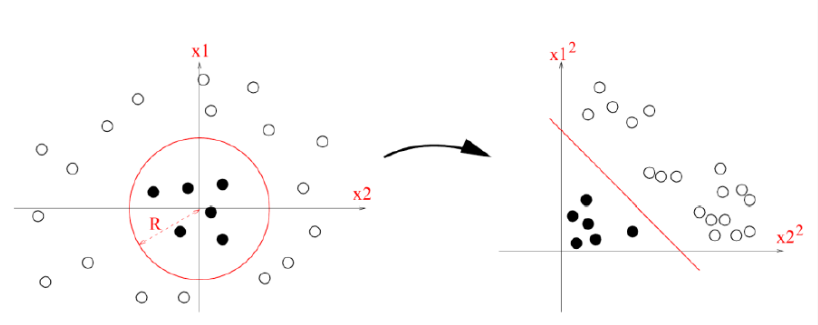


Figure 6: Transformation vom *input space* in den *feature space*

- Transformiere Daten zunächst in den *feature space* und wende dort lineare SVM an

Nonlinear SVM – *feature space*

- $\Phi : \mathbb{R}^r \rightarrow \mathcal{H}$ als nicht-lineare Transformation
- alte Gleichung: $\min ||\beta||^2$ mit $y_i(\beta_0 + {x_i}^T\beta) \geq +1$
- neue Gleichung: $\min ||\beta||^2$ mit $y_i(\beta_0 + \Phi(x_i)^T\beta) \geq +1$

Nonlinear SVM – optimizing in *feature space*

Lineare SVM

- Wolfe-Dual: $\max_\alpha 1_n^T \alpha - \frac{1}{2}\alpha^T H \alpha$ mit Nebenbedingungen $\alpha_i \geq 0, \alpha^T y = 0$
  - $H_{ij} = y_i y_j (x_i^T x_j)$

Nonlinear SVM – optimizing in *feature space*

Lineare SVM

- Wolfe-Dual: $\max_\alpha 1_n^T\alpha - \frac{1}{2}\alpha^T H\alpha$ mit Nebenbedingungen $\alpha_i \geq 0, \alpha^T y = 0$
    - $H_{ij} = y_i y_j (x_i^T x_j)$

Nicht-lineare SVM

- Wolfe-Dual: $\max_\alpha 1_n^T\alpha - \frac{1}{2}\alpha^T H\alpha$ mit Nebenbedingungen $\alpha_i \geq 0, \alpha^T y = 0$
    - $H_{ij} = y_i y_j (\Phi(x_i)^T \Phi(x_j))$

Nonlinear SVM – Kernel Trick

- Wolfe-Dual: $\max_\alpha 1_n^T\alpha - \frac{1}{2}\alpha^T H\alpha$ mit Nebenbedingungen $\alpha_i \geq 0, \alpha^T y = 0$
  - $H_{ij} = y_i y_j(\Phi(x_i)^T\Phi(x_j))$

Nonlinear SVM – Kernel Trick

- Wolfe-Dual: $\max_\alpha 1_n^T \alpha - \frac{1}{2}\alpha^T H \alpha$ mit Nebenbedingungen $\alpha_i \geq 0, \alpha^T y = 0$
  - $H_{ij} = y_i y_j (\Phi(x_i)^T \Phi(x_j))$
- inneres Produkt $\Phi(x_i)^T \Phi(x_j)$ meist aufwendig zu berechnen

Nonlinear SVM – Kernel Trick

- Wolfe-Dual: $\max_\alpha 1_n^T \alpha - \frac{1}{2}\alpha^T H \alpha$ mit Nebenbedingungen $\alpha_i \geq 0, \alpha^T y = 0$
  - $H_{ij} = y_i y_j (\Phi(x_i)^T \Phi(x_j))$
- inneres Produkt $\Phi(x_i)^T \Phi(x_j)$ meist aufwendig zu berechnen
- Kernel Trick: nicht transformieren und Skalarprodukt, sondern direkt berechnen: $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$

Nonlinear SVM – Kernel Trick

- Wolfe-Dual: $\max_\alpha 1_n^T \alpha - \frac{1}{2}\alpha^T H \alpha$ mit Nebenbedingungen $\alpha_i \geq 0, \alpha^T y = 0$
  - $H_{ij} = y_i y_j (\Phi(x_i)^T \Phi(x_j))$
- inneres Produkt $\Phi(x_i)^T \Phi(x_j)$ meist aufwendig zu berechnen
- Kernel Trick: nicht transformieren und Skalarprodukt, sondern direkt berechnen: $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$
- somit: $H_{ij} = y_i y_j K(x_i, x_j)$

Nonlinear SVM – Kernel Trick

- Wolfe-Dual: $\max_\alpha 1_n^T\alpha - \frac{1}{2}\alpha^T H\alpha$ mit Nebenbedingungen $\alpha_i \geq 0, \alpha^T y = 0$
  - $H_{ij} = y_i y_j(\Phi(x_i)^T\Phi(x_j))$
- inneres Produkt $\Phi(x_i)^T\Phi(x_j)$ meist aufwendig zu berechnen
- Kernel Trick: nicht transformieren und Skalarprodukt, sondern direkt berechnen: $K(x_i, x_j) = \Phi(x_i)^T\Phi(x_j)$
- somit: $H_{ij} = y_i y_j K(x_i, x_j)$
- Fasst man K(x_i,x_j) als Matrix zusammen, so nennt man diese Kernel Matrix

Beispiele für Kernel

- polynomialer *Kernel*: $K(x,y) = (x^T y + 1)^2$.

Beispiele für Kernel

- polynomialer *Kernel*: $K(x, y) = (x^T y + 1)^2$.
  - Falls *input space* 2-dim: $\Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)^T$ *feature space* also 6-dim

Beispiele für Kernel

- polynomialer *Kernel*: $K(x, y) = (x^T y + 1)^2$.
  - Falls *input space* 2-dim: $\Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)^T$ *feature space* also 6-dim
- Gauß-Kernel: $K(x, y) = \exp(-\frac{||x - y||^2}{2\sigma^2})$
  - meist genutzter Kernel

Implementierung Kernel-basierter SVM

SVM und Loss-Funktion

- häufiges Schema in ML: Minimierung einer loss-Funktion
- loss: summiere Fehlklassifikationen

SVM und Loss-Funktion

- häufiges Schema in ML: Minimierung einer loss-Funktion
- loss: summiere Fehlklassifikationen
- für SVM bietet sich der Hinge-loss an
  - $\ell(f(x_i)) = \max(0, 1 - y_i f(x_i))$
  - loss: $\sum_i \ell(f(x_i))$

SVM und Loss-Funktion

- häufiges Schema in ML: Minimierung einer loss-Funktion
- loss: summiere Fehlklassifikationen
- für SVM bietet sich der Hinge-loss an
  - $\ell(f(x_i)) = \max(0, 1 - y_i f(x_i))$
  - loss: $\sum_i \ell(f(x_i))$

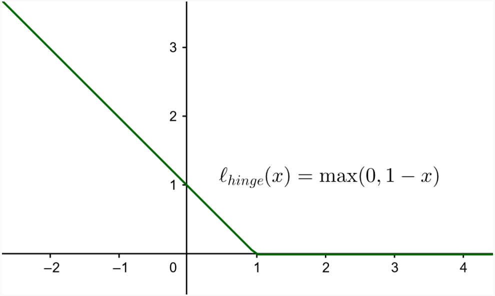


Figure 7: Graph des *Hinge-loss*

Minimierung der Loss-Funktion

- meist genutzt: Gradient descent (Gradientenverfahren)
- $\beta_{k+1} = \beta_k + \eta g$
    - mit η als Schrittweite und Abstiegsrichtung g

Minimierung der Loss-Funktion

- meist genutzt: Gradient descent (Gradientenverfahren)
- $\beta_{k+1} = \beta_k + \eta g$
  - mit η als Schrittweite und Abstiegsrichtung g
- Hinge-loss nicht differenzierbar, daher werden Subgradienten verwendet.

Konstruktion der SVM mit Hinge-Loss

- Ziel: minimiere $\lambda||\beta||^2 + \sum_i \ell(f(x_i))$
  - λ ist Regularisierungsparameter (Hyperparameter)

Konstruktion der SVM mit Hinge-Loss

- Ziel: minimiere $\lambda||\beta||^2 + \sum_i \ell(f(x_i))$
  - λ ist Regularisierungsparameter (Hyperparameter)
- Minimum nähern mit Gradientenverfahren

Implementierung Hinge-Loss