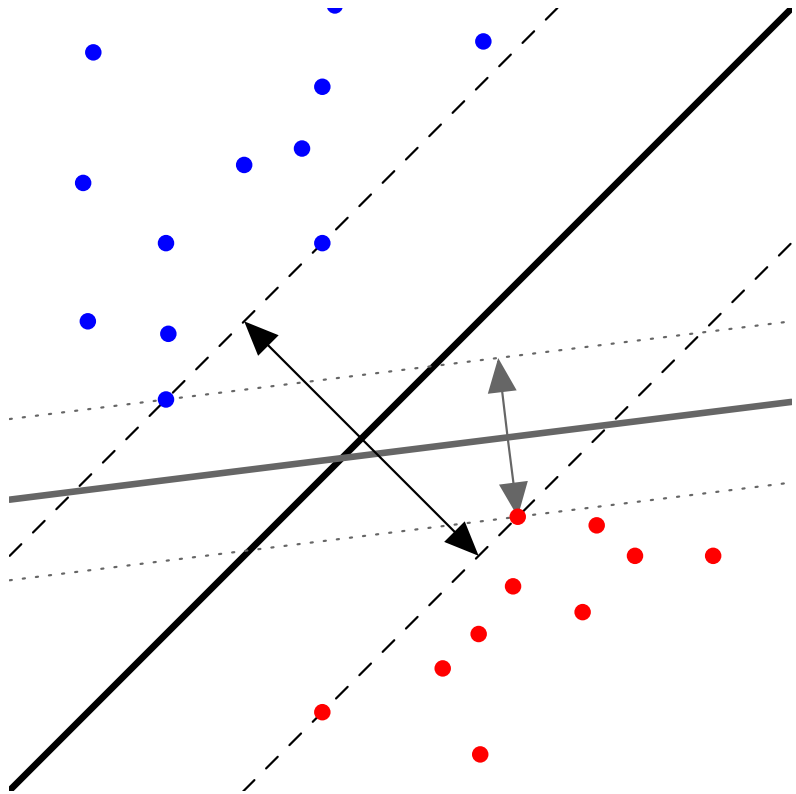
Grundlagen von Support Vector Machines (SVM)



Agenda

1. Einordnung
2. Lineare SVM – primales Problem
3. Duales Problem – Kernel-Trick
4. Soft Margin
5. Hyperparametersuche
6. Mehrfachklassifikation

Einordnung

Eine *Support Vector Machine* ist

- Supervised Learning Verfahren
- Klassifikationsverfahren
  und trennt genau 2 Klassen
- Batch-basiert (nicht online)
- Modell-basiert (Trainingsdaten später nicht nötig)

Bezeichnungen

$\mathbf{x} = (x_1, \ldots, x_n)^\mathsf{T}$	Vektor mit n Komponenten.
$\mathbb{R}^n$	Menge aller n-dimensionalen reellwertigen Vektoren
M	Anzahl der Trainingsdaten
$\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 + \ldots + x_n y_n$	Skalarprodukt der Vektoren $\mathbf{x}$ und $\mathbf{y}$.
$\|\mathbf{x}\| = \sqrt{x_1^2 + \ldots + x_n^2}$	Euklidsche Norm (= Länge) des Vektors $\mathbf{x}$.
$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right)^\mathsf{T}$	Partielle Ableitung der reellwertigen Funktion $f(\mathbf{x})$ nach allen Komponenten.
$\arg\min_{\mathbf{x}} f(\mathbf{x})$	Der Vektor $\mathbf{x}$, für den $f(\mathbf{x})$ minimal wird.
s. t.	Nebenbedingungen (*subject to*).

Klassifikation: Ziel und Vorgehen

- Ein neues Objekt (Datenpunkt) soll einer Klasse zugeordnet werden
  (SPAM – HAM, Katze – Hund – Pferd, Kunde wird Produkt kaufen oder nicht, ...).
- Für jeden Datenpunkt gleiche Menge von Daten (Features, Komponenten) vor, die als reelle Zahlen codiert sind
  (Anzahl Worthäufigkeiten, Pixelwerte, Alter, Geschlecht, bisher gekaufte Produkte, ...).
- Eine bestimmte Zahl von Daten ist bereits klassifiziert (Label).
- Zunächst genau zwei Klassen mit den Labels -1 und $+1$.
- Aus den klassifizierten Daten (Trainingsdaten) wird das Modell erstellt.
- Mithilfe des Modells werden neue Daten klassifiziert.

Geschichte der Support Vector Machine

- Lineare SVM: Vapnik und Chervonenkis 1963.
- Kernel Trick: Boser, Guyon, Vapnik 1992.
- Soft Margin: Cortes, Vapnik 1993–1995.
- Weitere Entwicklungen bei Optimierungsverfahren und Mehrklassen-SVM.

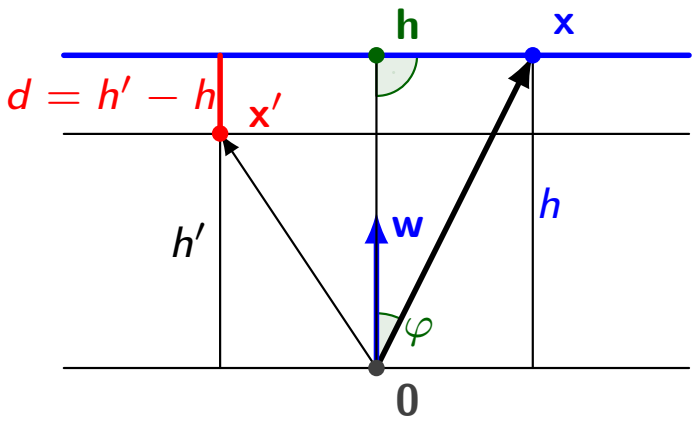
Prinzipielles Vorgehen

- Betrachten Datenpunkte als Punkte im (euklidschen) Raum.
- Konstruieren eine Hyperebene, die beide Punktklassen voneinander «optimal» trennt $\rightarrow$ Modell.
- Für neuen Punkt wird bestimmt, auf welcher Seite der Hyperebene er liegt $\rightarrow$ Klasse.

Fragen:

- Bestimmen der Hyperebene?
- Was ist «optimal»?
- Bestimme die Seite der Ebene für einen Punkt.

Abstand eines Punktes von einer Ebene



- **w** Vektor senkrecht zur Ebene, **x** beliebiger Punkt auf Ebene, φ Winkel zwischen **w** und **x**.
- Abstand der Ebene vom Ursprung: $h = \|\mathbf{x}\| \cos\varphi$ gegeben.
- Skalarprodukt: $\langle \mathbf{w}, \mathbf{x} \rangle = \|\mathbf{w}\| \cdot \|\mathbf{x}\| \cdot \cos\varphi$
- Also $h = \frac{\langle \mathbf{w}, \mathbf{x} \rangle}{\|\mathbf{w}\|}$ oder $\frac{\langle \mathbf{w}, \mathbf{x} \rangle}{\|\mathbf{w}\|} - h = 0$
- Setzen $b := -h \cdot \|\mathbf{w}\|$

- Also

$$\frac{\langle \mathbf{w}, \mathbf{x} \rangle + b}{\|\mathbf{w}\|} = 0$$

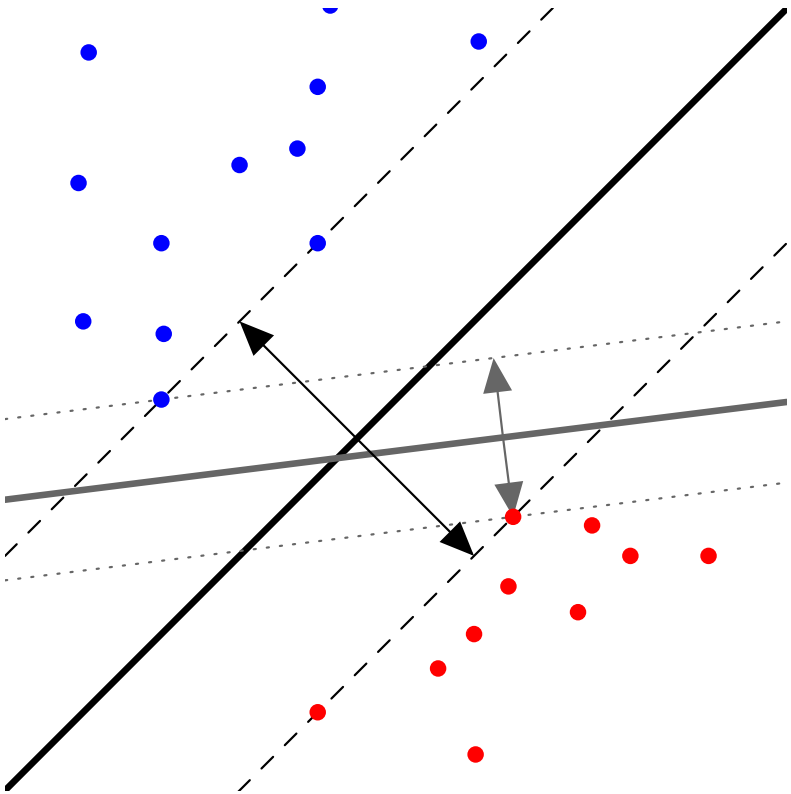
(Ebenengleichung).

- Punkt $\mathbf{x}'$ liegt auf einer Parallelebene mit Abstand $h' = \frac{\langle \mathbf{w}, \mathbf{x}' \rangle}{\|\mathbf{w}\|}$ vom Ursprung, also Abstand von der Ebene

$$d = h' - h = h' + \frac{b}{\|\mathbf{w}\|}$$
$$= \frac{\langle \mathbf{w}, \mathbf{x}' \rangle + b}{\|\mathbf{w}\|}$$

- Multiplikation von **w** und b mit positiven Faktor verändert Abstand nicht.
- Multiplikation mit negativem Faktor vertauscht Seiten der Ebene.

Grundidee der SVM



- Lege die Hyperebene so, dass der kleinste Abstand eines Punktes zur Ebene möglichst groß ist (und auf der «richtigen» Seite/Klasse t liegt):

$$t \left(\frac{\langle \mathbf{w}, \mathbf{x} \rangle + b}{\|\mathbf{w}\|} \right) \to \max, \quad t \in \{-1, +1\}$$

- Möglichst breiter «Streifen» unterhalb und oberhalb der Ebene, der frei von Trainingspunkten ist.
- Punkte auf dem «Rand» heißen *Supportvektoren*.
- Neue Punkte werden damit hoffentlich «richtig» klassifiziert.

Optimierungsproblem

$$\arg\max_{\mathbf{w},\, b} \left\{ \min_{m=1,\,\ldots,\,M} t_m \left(\frac{\langle \mathbf{w},\, \mathbf{x}_m \rangle + b}{\|\mathbf{w}\|} \right) \right\}$$
$$= \arg\max_{\mathbf{w},\, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_{m=1,\,\ldots,\,M} t_m \left(\langle \mathbf{w},\, \mathbf{x}_m \rangle + b \right) \right\}$$

- Wähle die Ebene $\mathbf{w}$, b so, dass der kleinste Abstand eines Punktes zur ihr maximiert wird.
- Doppeloptimierung schwierig, deshalb Umformung.

Vereinfachung des Optimierungsproblems

- Sei κ das Minimum, damit hat jeder Punkt mindestens den Abstand $\pm\kappa$ von der Ebene, also

$$t_m \left(\langle \mathbf{w}, \mathbf{x}_m \rangle + b \right) \geq \kappa > 0, \; m = 1, \ldots, M$$

- Teilen $\mathbf{w}$ und b durch κ (Ebenenabstand ändert sich nicht):

$$t_m \left(\langle \mathbf{w}', \mathbf{x}_m \rangle + b' \right) \geq 1, \; m = 1, \ldots, M$$

- Wir erhalten damit das neue Problem mit Nebenbedingungen:

$$\begin{aligned} &\arg\max_{\mathbf{w},\, b} \left\{ \frac{1}{\|\mathbf{w}\|} \right\} \\ \text{s. t.} \quad &t_m \left(\langle \mathbf{w}, \mathbf{x}_m \rangle + b \right) \geq 1, \; m = 1, \ldots, M \end{aligned}$$

- Wenn $\frac{1}{\|\mathbf{w}\|}$ maximal, dann $\|\mathbf{w}\|$ minimal, ebenso $\|\mathbf{w}\|^2$.
- Fügen noch einen Faktor $\frac{1}{2}$ ein.

Primales Optimierungsproblem der linearen SVM

$$\arg\min_{\mathbf{w},\, b} \frac{1}{2}\|\mathbf{w}\|^2$$

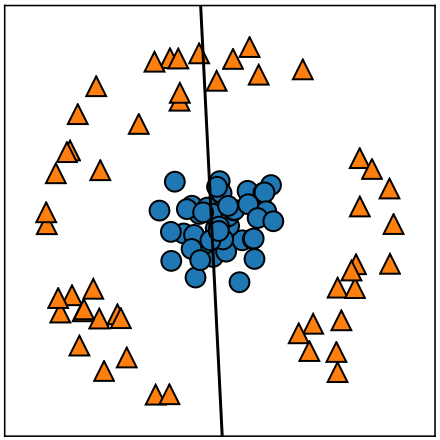
$$\text{s. t.} \quad t_m \left(\langle \mathbf{w},\, \mathbf{x}_m \rangle + b\right) \geq 1,\ m = 1,\, \ldots,\, M$$

- Lösung mit iterativen Verfahren, z. B. mit *Newton-Verfahren*, *Coordinate Descent* oder *Stochastic Gradient Descent* (SGD) in `scikit-learn`.
- Komplexität:

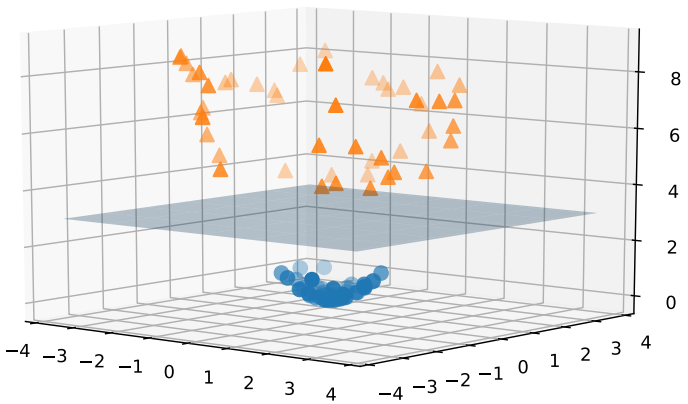
$$O(k \cdot M \cdot \tilde{n})$$

mit Iterationszahl k und Komponentenzahl $\tilde{n}$ jedes Datenpunktes ungleich 0.

Problem 1: Keine lineare Trennung möglich



- Lineare Trennung nicht möglich.



- Mit Radius als neue Komponente.

- Nichtlineare Transformation: Verknüpfen von Komponenten, hier

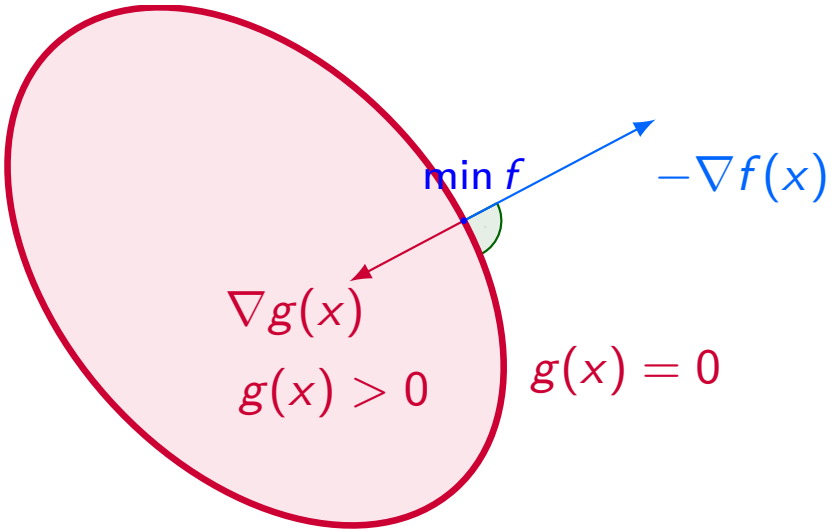
$$\Phi(\mathbf{x}) = (x_1, x_2, x_1^2 + x_2^2)^\mathsf{T}$$

- Allgemein: Verknüpfung von Komponentenpaaren $a_{ii}x_i^2 + a_{ij}x_ix_j + a_{jj}x_j^2 + b_ix_i + b_jx_j$.

Probleme:

- Zahl der Komponenten steigt extrem: vielleicht auch Tripel, Quadrupel, höhere Potenzen notwendig.
- Große Zahl frei wählbarer Parameter: welche Werte wählen?

Duales Problem



- Verallgemeinertes Problem:
  $f(\mathbf{x}) \rightarrow \min$, s. t. $g(\mathbf{x}) \geq 0$.
- Lagrange-Formulierung:

$$\mathcal{L}_a(\mathbf{x}) = f(\mathbf{x}) + a \cdot g(\mathbf{x})$$
$$\text{s. t.} \quad g(\mathbf{x}) \geq 0, \; a \geq 0$$

- Minimum von $\mathcal{L}$ kann höchstens größer sein als das von f.

- Liegt das Optimum auf dem Rand $g(\mathbf{x}) = 0$, liegen ∇f und ∇g auf einer Geraden, also:

$$\nabla \mathcal{L}_a(\mathbf{x}) = \nabla f(\mathbf{x}) + a \cdot \nabla g(\mathbf{x}) = \mathbf{0}$$

- Wenn Optimum nicht auf Rand, dann $\nabla f(\mathbf{x}) = 0$, damit $a = 0$.
- Umformen von $\nabla \mathcal{L}_\mathbf{a}(\mathbf{x}) = 0$ nach $\mathbf{x}$.
- Eliminieren in $\mathcal{L}$ Parameter $\mathbf{x} \rightarrow$ neues Optimierungsproblem in a.
- Mehrere Nebenbedingungen $g_m(\mathbf{x}) \geq 0$:

$$\mathcal{L}_\mathbf{a}(\mathbf{x}) = f(\mathbf{x}) + \sum_{m=1}^{M} a_m \cdot g_m(\mathbf{x})$$

- Was bringt das?

Langrangefunktion für lineare SVM

- Nebenbedingung umformulieren ($g(\mathbf{x}) \geq 0$):

$$t_m \left(\langle \mathbf{w}, \mathbf{x}_m \rangle + b \right) - 1 \geq 0$$

- Lagrangefunktion in $\mathbf{w}$ und b, Lagrange-Parameter $\mathbf{a}$

$$\mathcal{L}_{\mathbf{a}}(\mathbf{w}, b) = \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{m=1}^{M} a_m (t_m \left(\langle \mathbf{w}, \mathbf{x}_m \rangle + b \right) - 1)$$

- Setze alle Ableitungen nach w_i und b auf 0, ersetze alle w_i und b.
- Optimierungsproblem in $\mathbf{a}$.

Duales Problem

$$\arg\min_{\mathbf{a}} \left\{ \frac{1}{2} \sum_{i=1}^{M} \sum_{k=1}^{M} a_i a_k t_i t_k \langle x_i, x_k \rangle - \sum_{i=1}^{M} a_i \right\}$$

$$\text{s. t.} \qquad a_i \geq 0, \quad \sum_{i=1}^{M} a_i t_i = 0, \quad i = 1, \ldots, M$$

Rücksubstition von $\mathbf{w}$ durch a_m ergibt Klasse (= Abstand von Trennebene mit Vorzeichen)

$$y(\mathbf{x}) = \sum_{m=1}^{M} a_m t_m \langle \mathbf{x}, \mathbf{x}_m \rangle + b$$

Berechnung von b nicht direkt möglich.

KKT-Bedingung

- Aus Herleitung der Lagrangeformulierung: $\mathbf{a}_m \neq 0$ nur für Supportvektoren $\mathbf{x}_m$
- Daraus KKT-Bedingung:

$$\begin{aligned} a_m &\geq 0 \\ t_m y(\mathbf{x}_m) - 1 &\geq 0 \\ a_m(t_m y(\mathbf{x}_m) - 1) &= 0 \quad m = 1, \ldots, m \end{aligned}$$

- Damit können b und $y(\mathbf{x})$ aus a_m berechnet werden:

$$y(\mathbf{x}) = \sum_{s \in S} a_s t_s \langle \mathbf{x}, \mathbf{x}_s \rangle + b \qquad S \text{ Indices der Supportvektoren}$$

$$\text{mit} \quad b = \frac{1}{|S|} \sum_{s \in S} \left(t_s - \sum_{u \in S} a_i t_u \langle \mathbf{x}_s, \mathbf{x}_u \rangle \right)$$

Kernel-Trick

- Datenpunkte $\mathbf{x}$, $\mathbf{x}_m$ erscheinen nur in Skalarprodukten.
- Betrachten nichtlineare Transformation: $\mathbf{x} \rightarrow \Phi(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^\infty$
- Ersetzen

$$\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_k) \rangle : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^\infty \times \mathbb{R}^\infty \rightarrow \mathbb{R}$$

  durch Kernelfunktion

$$k(\mathbf{x}_i, \mathbf{x}_k) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$$

- Statt $\Phi()$ wird *Kernelfunktion* $k(\cdot, \cdot)$ benötigt.
- Jede Funktion geeignet, deren *Gram-Matrix* $[k(\mathbf{x}_i, \mathbf{x}_k)]_{i,k=1}^{M}$ positiv semidefinit ist (*Mercer*).

Übliche Kernelfunktionen

- «Gauß-Kernel» (Radial Base Function):

$$k(\mathbf{x}_i, \mathbf{x}_k) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_k\|^2}$$

  Stationärer Kernel (nur von Punktabstand abhängig), freier Parameter γ.

- Polynomkernel:

$$k(\mathbf{x}_i, \mathbf{x}_k) = (\langle \mathbf{x}_i, \mathbf{x}_k \rangle + r)^d$$

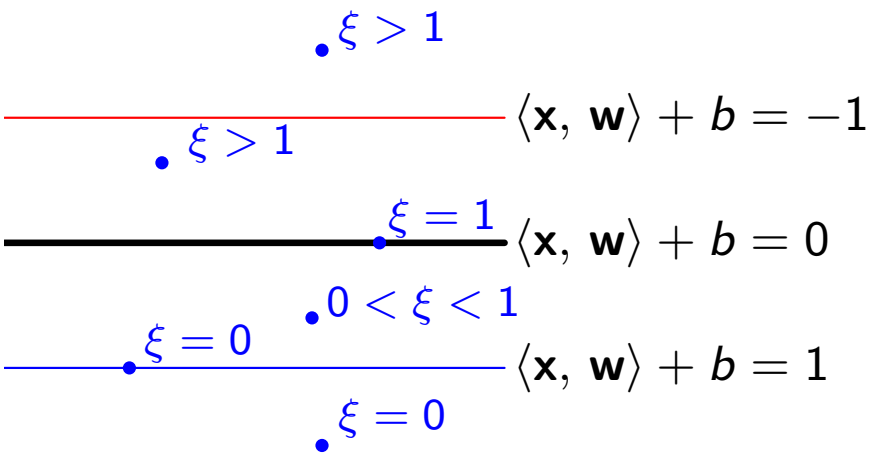
  Freie Parameter r, d.

- Sigmoid-Kernel:

$$k(\mathbf{x}_i, \mathbf{x}_k) = \tanh(\kappa \langle \mathbf{x}_i, \mathbf{x}_k \rangle - \delta)$$

  Freie Parameter κ, δ.

Problem 2: Behandlung von Ausreißern



- Durch «Ausreißer» liegen einzelne Datenpunkte in der «falschen» Klasse, keine Trennung möglich.
- Erlauben einzelne Ausreißer durch *Slackvariable*, addieren Korrekturwert auf die eigentlich verletzte Randbedingung, um diese zu erfüllen.
- Die Summe der Korrekturen soll möglichst klein sein.

Optimierungsproblem mit Slackvariablen

$$\underset{\mathbf{w},\, b,\, \xi_1,\, \ldots,\, \xi_M}{\arg\min} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{m=1}^{M} \xi_m \right\}$$

$$\text{s. t.} \quad t_m \left(\langle \mathbf{w},\, \mathbf{x}_m \rangle + b \right) + \xi_m \geq 1,\; \xi_m \geq 0,\; m = 1,\, \ldots,\, M$$

- C ist *Regularisierungsparameter*.
- Je größer C, desto komplexer das Modell (höhere «Bestrafung» der Ausreißer).
- Statt $C \sum \xi_m$ (*Lasso*-Regularisierung) auch $C \sum \xi_m^2$ (*Ridge*-Regularisierung).
- Duales Problem: Lagrange-Funktion mit Variablen $\mathbf{w}$, b, $\xi_1, \ldots, \xi_M$ und neuen Parametern für s. t. $a_1, \ldots, a_M$, $\alpha_1, \ldots, \alpha_M$.

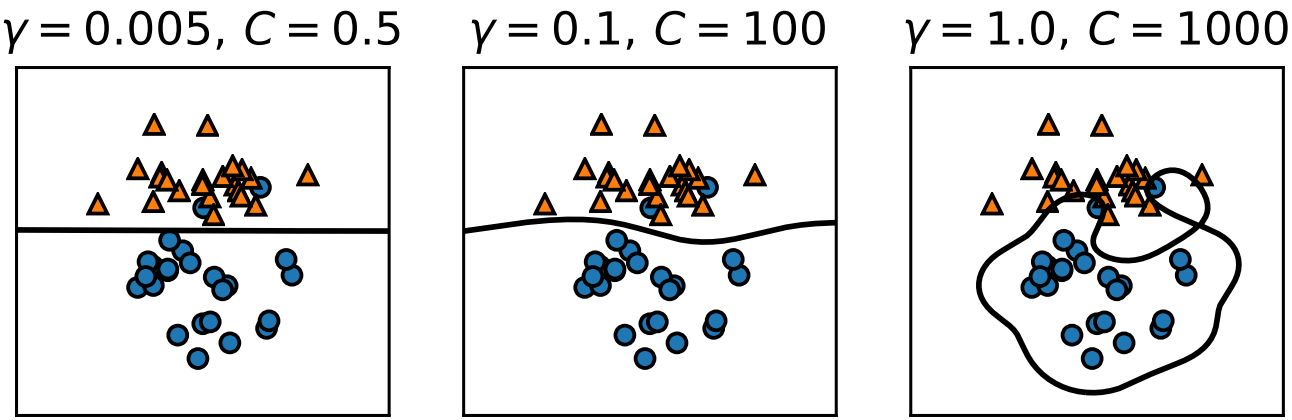
Duales Problem mit Soft Margin

$$\arg\min_{\mathbf{a}} \left\{ \frac{1}{2} \sum_{i=1}^{M} \sum_{k=1}^{M} a_i a_k t_i t_k \langle x_i, x_k \rangle - \sum_{i=1}^{M} a_i \right\}$$

$$\text{s. t.} \qquad 0 \leq a_i \leq C, \quad \sum_{i=1}^{M} a_i t_i = 0, \quad i = 1, \ldots, M$$

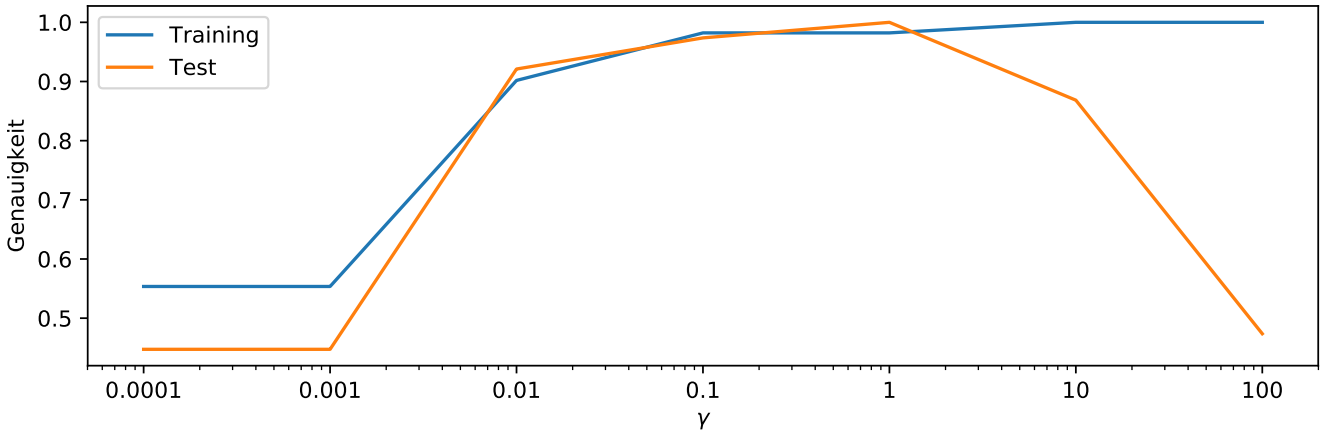
- Lagrange-Faktoren α_m kürzen sich weg.
- Quadratisches Problem mit *box constraints*.
- Lösungsverfahren *Sequential Minimal Optimization* (SMO) in `libsvm`, `scikit-learn`.
- Komplexität: $O(M^2 \cdot n)$ bis $O(M^3 \cdot n)$ (datenabhängig).
- Komplexität begrenzt Anzahl der Trainingsdaten.

Training von SVM



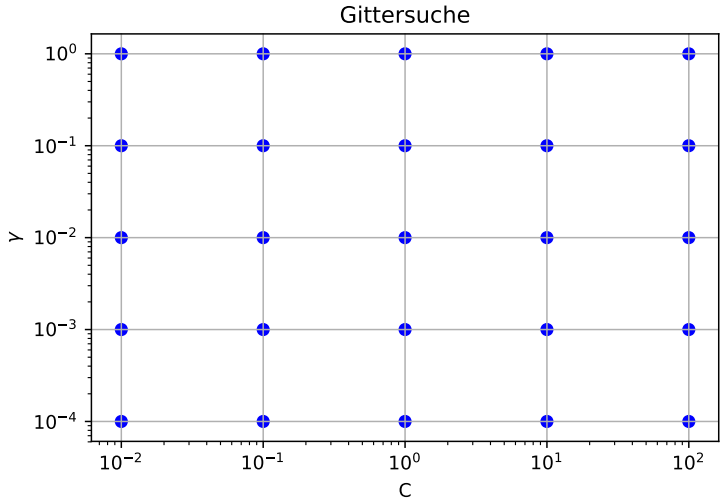
- SVM-Modell ist eigentlich deterministisch und kann im Prinzip exakt berechnet werden.
- Aber: Rechenaufwand sehr hoch: Näherungsverfahren.
- Aber: Modell ist von mehreren Parametern abhängig, diese müssen optimiert werden.
- Underfitting: Modell ist zu einfach und klassifiziert zu schlecht.
- Overfitting: Modell «lernt» die Trainingsdaten auswendig und abstrahiert nicht.

Bewerten des Modells



- Daten teilen in Trainings- und Testmenge (z. B. 80 % / 20 %).
- Aufteilung muss statisch sein, Testdaten dürfen nie ins Training eingehen.
- Modellerstellung mit Trainingsmenge, Evaluierung mit Testmenge.
- Underfitting: Geringe Trefferzahl auf Trainingsmenge.
- Overfitting: extrem gute Trefferzahl auf Trainings-, schlechte Trefferzahl auf Testmenge.
- Optimales Modell: etwa gleich gute Trefferzahl auf Trainings- und Testmenge.

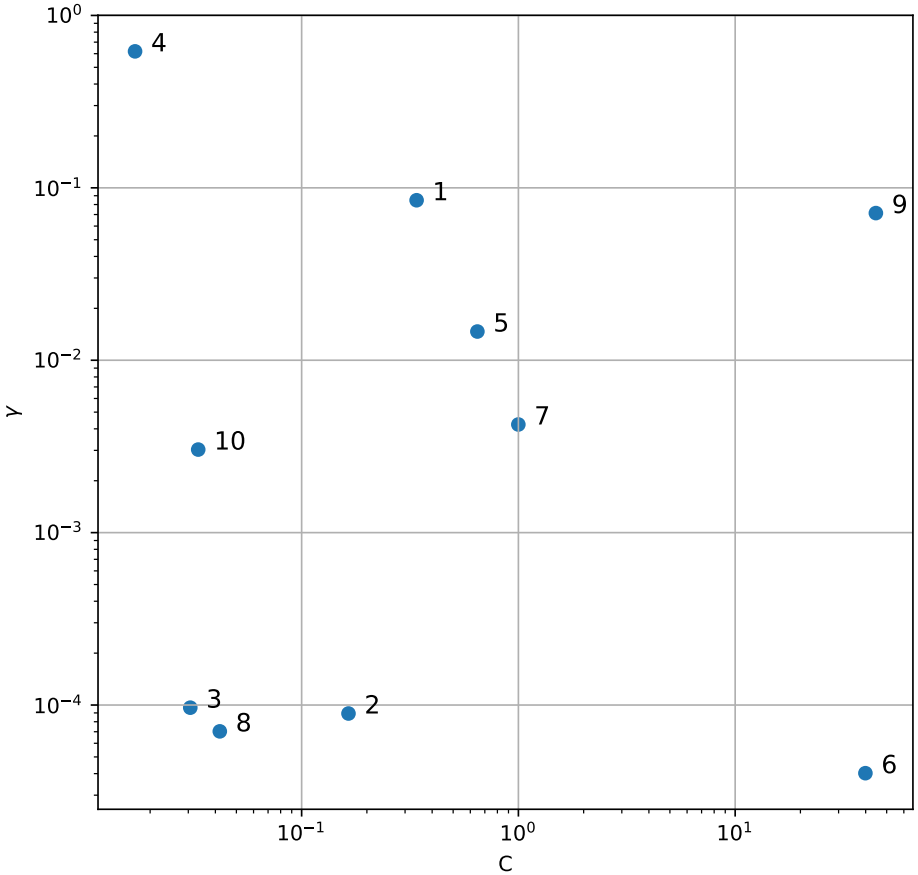
Gittersuche



- Gezielte Suche nach bestem Parametertupel.
- Systematisches Durchwandern der möglichen Parameterwerte in geometrischer Folge (Faktor 2 oder 10).

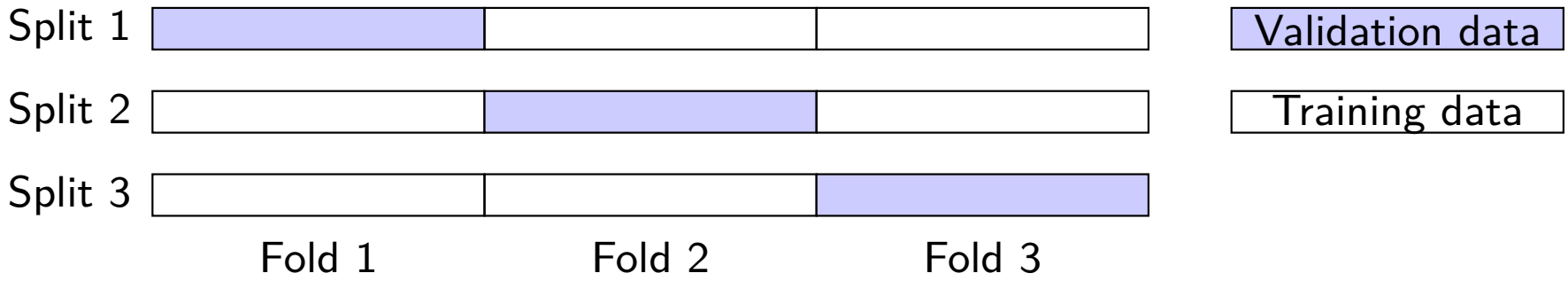
MinFehler $\leftarrow \infty$
for $C \in (10^{-1}, 1, 10, 100)$ **do**
  for $\gamma \in (10^{-5}, 10^{-4}, \ldots, 0.1, 1)$ **do**
    Bestimme Supportvektoren mit Parametertupel
    Berechne Fehler
    if Fehler $<$ MinFehler **then**
      MinFehler $\leftarrow$ Fehler
      $C_{best} \leftarrow C$
      $\gamma_{best} \leftarrow \gamma$
    end if
  end for
end for

Random Search



- Problem Gittersuche: systematisches Absuchen ungünstiger Parameterregionen kann lange dauern.
- Alternative: zufällige Auswahl der Hyperparameter
- Chance ist gut, geeignete Regionen in wenigen Schritten zu finden.
- Jederzeit Abbruch möglich.

Kreuzvalidierung



Problem: Gittersuche für komplexe Modelle führt zu Overfitting.

Lösung: Kreuzvalidierung.

- Aufteilen der Trainingsmenge in k Gruppen (Folds).
- Jede Gruppe muss Daten beider Klassen enthalten.
- Berechnung von k Modellen je Parameterpaar: $k - 1$ Gruppen für Modellberechnung, eine Gruppe für Validierung.
- Bewertung der Genauigkeit aller Modelle, Zusammenfassung (Mittelwert).

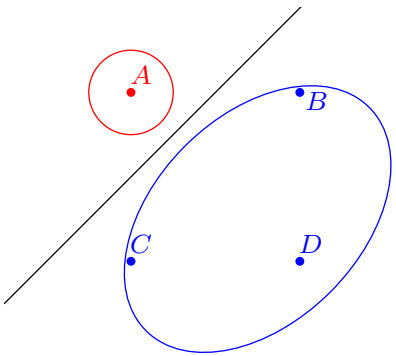
Anschließend: Modellberechnung für *alle* Trainingsdaten mit besten Parametern.

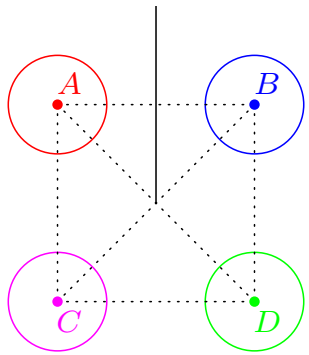
Aufbereitung der Daten

- Verfahren behandelt alle Dimensionen gleichgewichtet.
- Werte aller Dimensionen sollten im gleichen Intervall liegen (z. B. [0, 1] oder Mittelwert 0 und Varianz 1) $\rightarrow$ Umskalieren.
- Behandlung von Nominaldaten (mehrere Kategorien):
  - One-Hot-Codierung: Eine Variable pro Kategorie, genau eine dieser Variablen hat Wert 1, alle anderen 0.
  - Textdaten (jedes mögliche Wort eine Dimension): *Word embeddings*, z. B. *Word2Vec*.
- Behandlung fehlender Komponenten in Datenpunkten: Datenpunkt löschen oder Komponente durch Mittelwert bzw. Median ersetzen.

Mehrfachklassifikation

- Statt 2 nun K Klassen.
- SVM kann nur zwei Klassen trennen $\rightarrow$ mehrere SVMs.





One-versus-Rest:
Berechne K SVMs für jede Trennung k – nicht k (M Daten je SVM). Neuen Punkt mit jeder SVM bewerten und Klasse mit größtem Abstand wählen.

One-versus-One:
Berechne $\binom{K}{2}$ SVMs für jedes Klassenpaar mit Trainingsdaten dieses Paars ($\approx 2\frac{M}{K}$ Daten pro SVM). Neuer Punkt wird in die Klasse eingeordnet, die die meisten SVMs wählen.

In `scikit-learn`: lineare SVM mit OvR, Kernel-SVM mit OvO (Laufzeit).

Bibliotheken

- `libsvm`: Referenz- und Basisimplementierung, Kommandozeilen-Tool.
  Viele andere Bibliotheken sind einfach Schnittstellen zu `libsvm`.
- Python: `scikit-learn`: Sammlung von ML-Werkzeugen auf Basis von `numpy`, `SciPy`.
- R: `e1071` (libsvm-Schnittstelle), `kernlab` (libsvm und bsvm), `klaR` (svmlight).
- ...

Fazit

- Beliebtes Werkzeug des ML (Kaggle Survey 2017: 26 %).
- Für Datensätze mittlerer Größe bei Kernel-SVM ($O(m^2) \ldots O(m^3)$).
- Recht gute Generalisierung.
- Auch für große Datenmengen hochdimensionaler Daten (Text Mining) mit linearer SVM.
- Auch weniger Trainingsdaten als Dimensionen möglich (Overfitting?) mit linearer SVM.
- Hoher Aufwand für mehr als 2 Klassen.
- Online-Learning (Nachtrainieren mit neuen Daten) im Prinzip möglich (nicht in `scikit-learn`).
- Berechnung auf GPUs möglich (`ThunderSVM`)
- Modelle schwer interpretierbar.

Quellen I

[1] Arens et al.: Mathematik. Springer Spektrum. 2015.

[2] Bishop: Pattern Recognition and Machine Learning. Springer. 2006.

[3] Bordes et al.: Fast Kernel Classifiers with Online and Active Learning. ournal of Machine Learning Research. 2005. http://www.jmlr.org/papers/volume6/bordes05a/bordes05a.pdf

[4] Cortes, Vapnik: Support-vector networks. In: Machine Learning, 20. 1995.

[5] Hsu, Chang, Lin: A Practical Guide to Support Vector Classification. National Taiwan University. 2016. https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Quellen II

[6] Kaggle 2017 Survey Results. https://www.kaggle.com/amberthomas/kaggle-2017-survey-results

[7] Mikolov et al.: Efficient Estimation of Word Representations in Vector Space. arXiv 1301.3781. 2013. https://arxiv.org/pdf/1301.3781.pdf

[8] Müller, Guido: Einführung in Machine Learning mit Python. O'Reilly. 2017.

[9] Řehůřek: Scalability of Semantic Analysis in Natural Language Processing. Ph. D. Thesis. Brno. 2011. https://radimrehurek.com/phd_rehurek.pdf

[10] Wen et al.: ThunderSVM: A Fast SVM Library on GPUs and CPUs. Journal of Machine Learning Research 19. 2018. http://www.jmlr.org/papers/volume19/17-740/17-740.pdf