Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Textklassifikation und Informationsextraktion

Tobias Scheffer
Paul Prasse
Michael Großhans

Textklassifikation, Informationsextraktion

PROSAR-AIDA

File Edit Options View Window ?

0 - [H:\Doku\Intern\Tabellen\Demos\Rechnungslesung\...

GLOBE LTD.

Globe Ltd. World Retail
Mars House
Leafield Way
Corsham, Wiltshire
SN13 9SW

Orders: 01483 8786545
Fax: 01483 87856425
order@world.co.uk

Taxpoint Date: 26/09/02
Invoice Number: 23598
Your Order: 68974
Please refer on all payments

INVOICE

Paradatec Ltd.
Oban House, Rope Yard
Wootton Bassett, Wiltshire
SN4 7BW

Pos.	Description	Qty.	Price	Value
	Purchase order No. 4510425457			
01	4,000 Pcs Neon Light Bulb Material# 0124 Unit price 3,70			14,80
02	2,000 Pcs Heating Element NiChrome Material# 0453 Unit price 33,44			66,88
03	1,000 Pcs Hight Output LED Line (blue) Material# 0922 Unit price 12,45			12,45
04	8,000 Pcs Halogen Lamp Fixtures Chrome Material# 0765 Unit price 2,78			22,24
05	1,000 Pcs Transformer 12V Dual Purpose with Enhanced Screening Material# 0329 Unit price 22,95			22,95
06	2,000 Pcs Fuse Material# 0078 Unit price 0,75			1,50
	Sub-total			140,82

Globe Ltd. World Retail, Mars House, Leafield Way, Corsham, Wiltshire SN13 9SW
VAT registration number 534 2342 28

Results (primary)

Search objects | INVTABLE

	POS	ITEM	QUANTITY	PRICE	TOTAL
1	01	0124	4,000	3,70	14,80
2	02	0453	2,000	33,44	66,88
3	03	0922	1,000	12,45	12,45
4	04	0765	8,000	2,78	22,24
5	05	0329	1,000	22,95	22,95
6	06	0078	2,000	0,75	1,50

PROSAR-AIDA

Image #4
Process page?

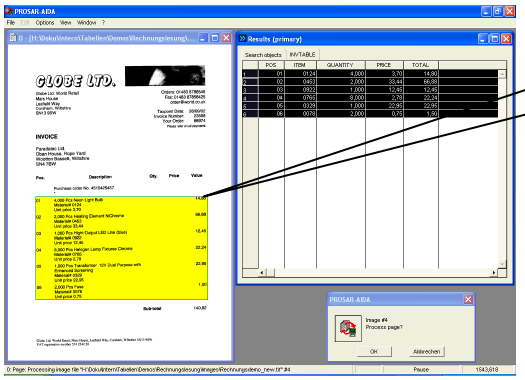
OK Abbrechen

0: Page: Processing image file "H:\Doku\Intern\Tabellen\Demos\Rechnungslesung\Images\Rechnungsdemo_new.tif" #4 | Pause | 1543,618

Textklassifikation, Informationsextraktion

- Textklassifikation: Text → Kategorie
  - Wird i.d.R. aus annotierten Daten gelernt.
  - Anwendungsbeispiel: Posteingangsverarbeitung.
- Informationsextraktion: Identifikation definierter Felder in Dokument.
  - Wird i.d.R. auch aus Daten gelernt.
  - Anwendungsbeispiel: Automatisierung von Dokumentenverarbeitungsprozessen.

Textklassifikation: Dokument ist eine Rechnung



Informationsextraktion: Feld enthält den Preis

Textklassifikation
Repräsentation

- Nach Tokenisierung wird Text durch Vektor repräsentiert.
- Vektorraummodell:
  - Vektor der Worthäufigkeiten für probabilistische Modelle.
  - TFIDF-Repräsentation für lineare Verfahren.
- Wortreihenfolge bleibt unberücksichtigt.
- Vektorraummodell mit N-Grammen:
  - Wird für Spamerkennung verwendet.
  - Jedes N-Gramm durch Dimension repräsentiert, oder
  - Sparse Binary Polynomial Hashing oder
  - Orthogonal Sparse N-Grams.

Textklassifikation
Repräsentation

- Jedes N-Gramm durch Dimension repräsentiert,
  - Nur N-Gramme, die auch tatsächlich auftreten.
- Sparse Binary Polynomial Hashing
  - Schiebe Fenster der Breite N über den Text.
  - Jede Teilmenge von bis zu N Token (Reihenfolge wird beachtet) und das am weitesten links stehende Token, bilden eine Dimension.
  - Berechne 32-bit Hashes für diese Teilmenge.
- Orthogonal Sparse Bigrams.
  - Fenster der Breite N wird über Text geschoben,
  - Jedes Paar aus einem beliebigen Token im Fenster und dem am linken Fensterrand stehenden Token ist ein Merkmal.
- SBPH und OSB: N-Gramme mit Platzhaltern.

Textklassifikation
Klassifikator vs. Entscheidungsfunktion

- Für eine binäre Klassifikation (y= +1 oder -1) von einem Objekt x wird meist eine Entscheidungsfunktion f(x) gelernt.
- Je größer f(x), desto wahrscheinlicher ist, dass x zur Klasse +1 gehört.
- Wenn $f(x) \geq \theta$, dann entscheide h(x) = +1, sonst h(x) = -1.
- Klassifikator h(x), Entscheidungsfunktion f(x).
- Der Wert für θ verschiebt „false positives“ zu „false negatives“.
- Optimaler Wert hängt von Kosten einer positiven oder negativen Fehlklassifikation ab.

Textklassifikation
Evaluation

Beispiel: Test auf HIV

- Fehlklassifikationswahrscheinlichkeit
  - Häufig nicht aussagekräftig, weil P(+1) sehr klein.
  - Wie gut sind 5% Fehler, wenn P(+1)=3%?
  - Idee: Nicht Klassifikator bewerten, sondern Entscheidungsfunktion.

- Wichtige Begriffe für die Evaluation:
  - True Positives (TP): h(x) = +1, und Klasse von x = +1
  - True Negatives (TN): h(x) = -1, und Klasse von x = -1
  - False Positives (FP): h(x) = +1, und Klasse von x = -1
  - False Negatives (FN): h(x) = -1, und Klasse von x = +1

Textklassifikation
Evaluation

- Precision / Recall
  - Precision : $\frac{\#TP}{\#TP+\#FP}$
  - Recall: $\frac{\#TP}{\#TP+\#FN}$
  - Precision-Recall-Kurve bewertet Entscheidungsfunktion,
  - Jeder Wert für θ entspricht Punkt auf P-R-Kurve.
  - F-Measure: „Durchschnitt“ aus Precision und Recall.
  - F-Measure $= 2 * \frac{\text{precision} * \text{recall}}{\text{precision}+\text{recall}}$
- Receiver Operating Characteristic (ROC-Kurve)
  - Bewertet Entscheidungsfunktion,
  - Fläche unter ROC-Kurve = P(positives Beispiel hat höheren f-Wert als negatives Beispiel)

Textklassifikation
ROC-Analyse

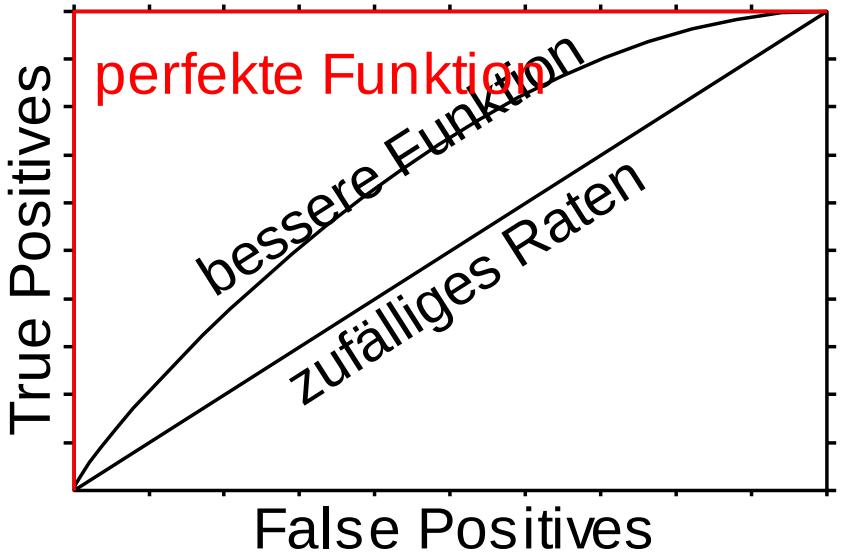
- Grundlage: Entscheidungsfunktion + Schwellwert = Klassifikator.
  - $h(x) = \begin{cases} +1, \text{wenn } f(x) \geq \theta \\ -1, \text{sonst.} \end{cases}$

    Klassifikator

    Entscheidungsfunktion
- Großer Schwellwert: Mehr positive Bsp falsch.
- Kleiner Schwellwert: Mehr negative Bsp falsch.
- Bewertung der Entscheidungsfunktion unabhängig vom konkreten Schwellwert.
- ROC = Receiver-Operating-Characteristic-Analyse.
- Werkzeug zur Bewertung der Qualität von Entscheidungsfunktionen.

Textklassifikation
ROC-Kurve

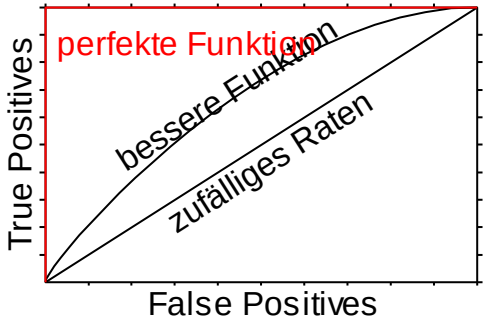
- Charakterisieren das Verhalten des Klassifikators für alle möglichen Schwellwerte.
- X-Achse: „False Positives“: Anzahl negativer Beispiele, die als positiv klassifiziert werden.
- Y-Achse: „True Positives“: Anzahl positiver Beispiele, die als positiv klassifiziert werden.



Textklassifikation
Bestimmen der ROC-Kurve (Algorithmus)

- Für alle positiven Beispiele X_p in Testmenge
  - Füge $f(x_p)$ in sortierte Liste L_p ein.
- Für alle negativen Beispiele X_n in Testmenge
  - Füge $f(x_n)$ in sortierte Liste L_n ein.
- Setze TP = FP = 0.
- Wiederhole solange L_p und L_n nicht leer sind:
  - Wenn $L_p \rightarrow$ Element $\geq L_n \rightarrow$ Element dann increment(TP) und $L_p = L_p \rightarrow$ Next.
  - Wenn $L_n \rightarrow$ Element $\geq L_p \rightarrow$ Element dann increment(FP) und $L_n = L_n \rightarrow$ Next.
  - Zeichne neuen Punkt (FP, TP)

Textklassifikation
Flächeninhalt der ROC-Kurve



- Flächeninhalt AUC kann durch Integrieren (Summieren der Trapez-Flächeninhalte) bestimmt werden.
- p = zufällig gezogenes Positivbeispiel
- n = zufällig gezogenes Negativbeispiel
- Theorem: AUC = P(f(p) > f(n)).

- Beweisidee: ROC-Kurve mit nur einem Positiv- und einem Negativbeispiel (Flächeninhalt 0 oder 1); Durchschnitt vieler solcher Kurven = AUC.

Textklassifikation
Precision und Recall

- Alternative zur ROC-Analyse.
- Stammt aus dem Information Retrieval.

- $$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$
- $$\text{Recall} = \frac{\text{True positives}}{\text{True positive} + \text{False negatives}}$$

- Precision: P(richtig | als positiv erkannt)
- Recall: P(als positiv erkannt | ist positiv)

Textklassifikation
Precision und Recall

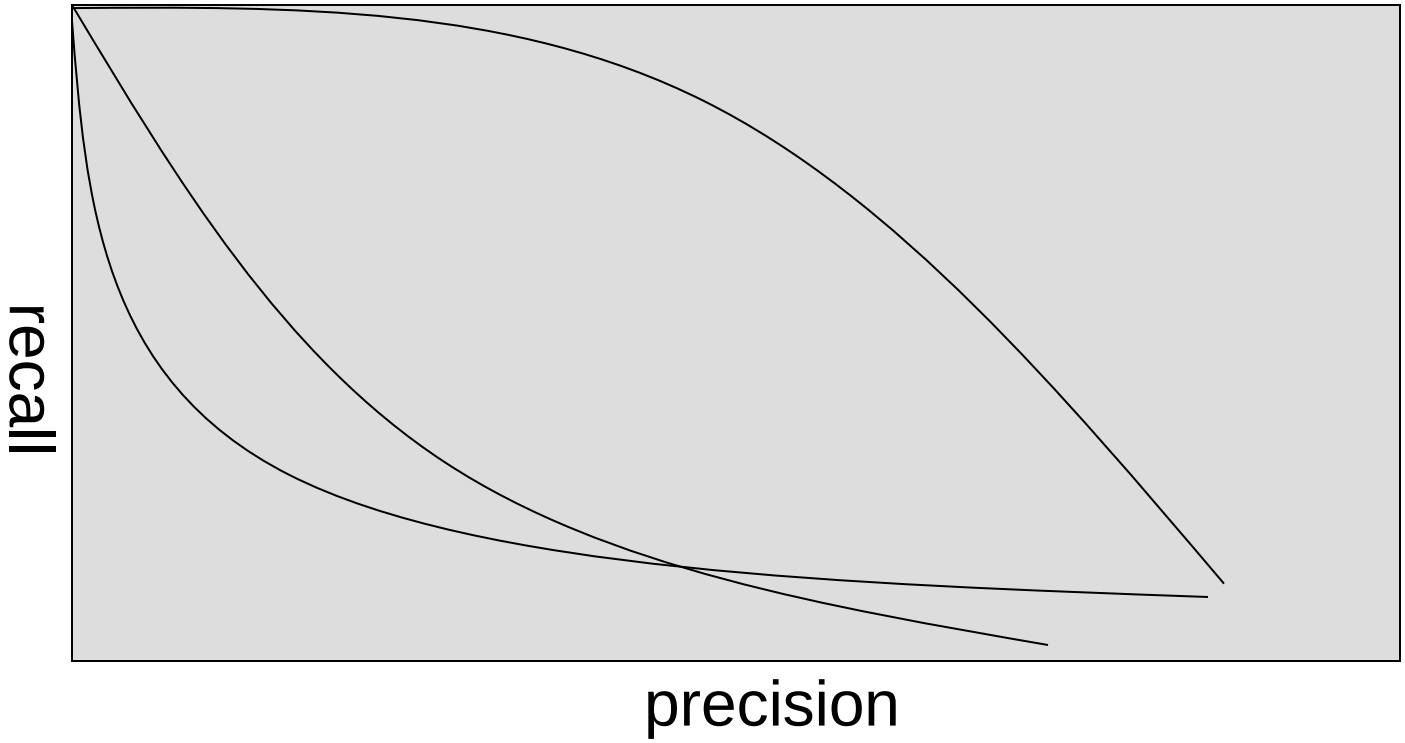
- Zusammenfassungen der Kurve in einer Zahl:
  - Maximum F-Measure: Maximum über alls (p,r)-Paare auf der Kurve:

$$\text{F-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

  - Precision-Recall-Breakeven-Point: Derjenige Wert für den gilt:

    Precision(θ) = Recall(θ) = PRBEP.

Textklassifikation
Precision und Recall: Trade-Off



- Precision-/Recall-Kurven
- Welcher Klassifikator ist der Beste / Schlechteste

Textklassifikation
Bestimmen der Performance-Maße

- Performance auf Trainingsmenge extrem optimistischer Schätzer.
- Zum Schätzen der Performance Daten verwenden, die nicht zum Trainieren verwendet wurden.
- Möglichkeiten:
  - Training-und-Test: Verwende z.B. 80% der Daten zum Trainieren und 20% der Daten zum Messen der ROC-Kurve, P-R-Kurve, oder Fehlklassifikationswahrscheinlichkeit.
  - N-Fold-Cross-Validation: Teile Daten in N Teile, wiederholtes Trainieren mit N-1 Teilen und Testen auf dem restlichen Teil.

Textklassifikation
Fehlerschätzung

- Training-and-Test (Algorithmus):
  - Aufteilen der Datenbank (m Datenpunkte) in Trainingsmenge (p % der Daten m) und Testmenge ($(100 - p)$ % der Daten m).
  - h_1 = Klassifikator trainiert auf der Trainingsmenge.
  - Bestimme $\hat{\mathrm{E}}$ anhand der Testmenge.
  - h = Klassifikator trainiert auf allen Daten.
  - Liefere Hypothese h zusammen mit Fehlerschätzer

$$\hat{E} \pm \sqrt{\frac{\hat{E}(1-\hat{E})}{p\% \cdot m}}$$

- Training-and-Test ist für große Datenbanken gut anwendbar.
- Problematisch für kleine Datenbanken.

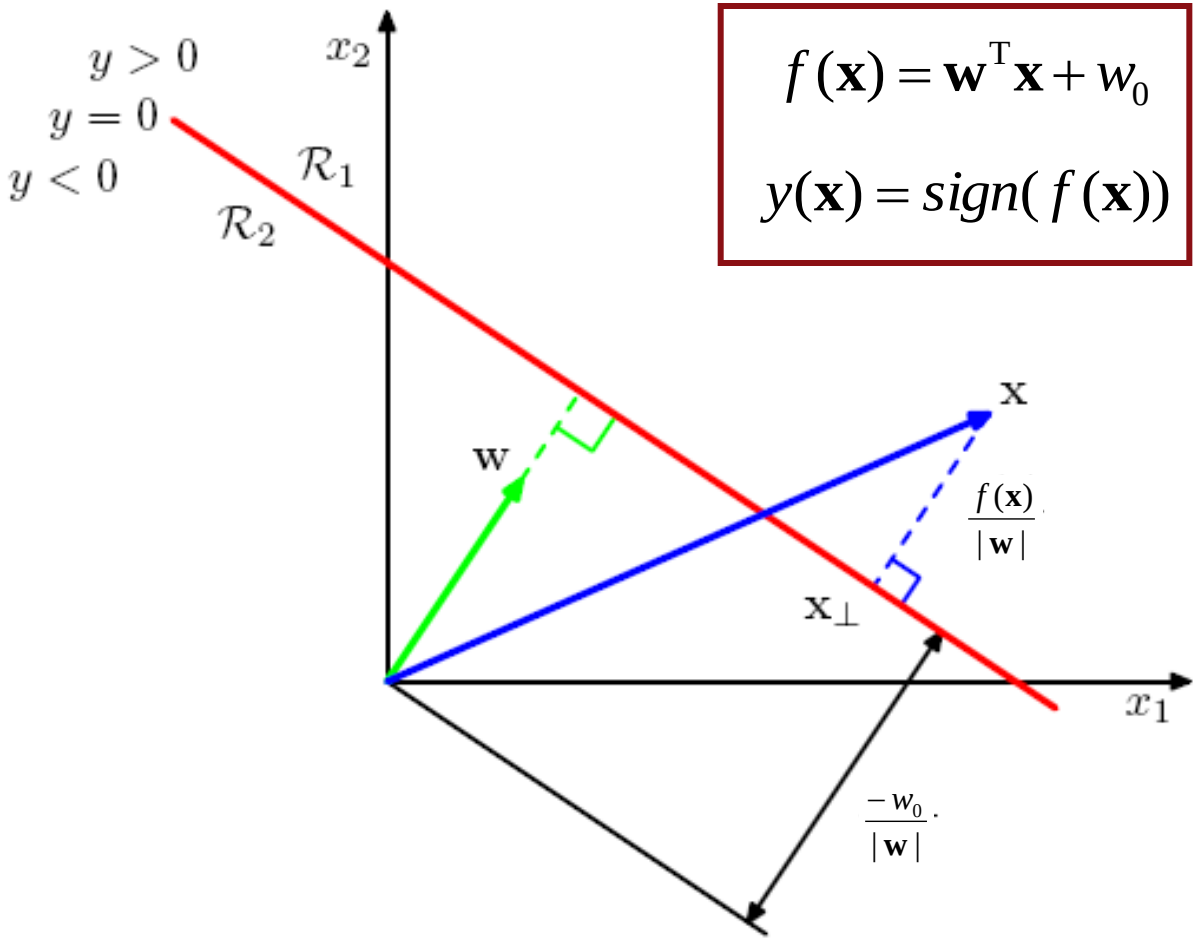
Textklassifikation
N-Fold Cross-Validation

- N-Fold Cross-Validation (Algorithmus):
  - Bilde N etwa gleich große Blöcke $S_1, \ldots, S_n$ der Datenmenge S mit $|S| = m$.
  - $\hat{\mathrm{E}} = 0$.
  - Für $i = 1 \ldots N$
    - h = Klassifikator trainiert auf Menge $S \backslash S_i$
    - $\hat{\mathrm{E}} = \hat{\mathrm{E}}$ + empirischer Fehler von h auf S_i
  - $\hat{\mathrm{E}} = \hat{\mathrm{E}}/N$
  - h = Klassifikator trainiert auf Menge S.
  - Liefere Hypothese h mit Fehlerschätzer

$$\hat{E} \pm \sqrt{\frac{\hat{E}(1-\hat{E})}{m}}$$

- Wenn |S| = N, heisst das Verfahren Leave-one-Out Cross Validation.
- Nur leicht pessimistischer Schätzer.

Lineare Klassifikatoren



$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

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

Lineare Klassifikatoren

- Umformulierung mit zusätzlichem, konstanten Eingabeattribut x_0=1:
  - $f(\mathbf{x}) = \mathbf{w}_{(1..n)}^T \mathbf{x}_{(1..n)} + w_0$

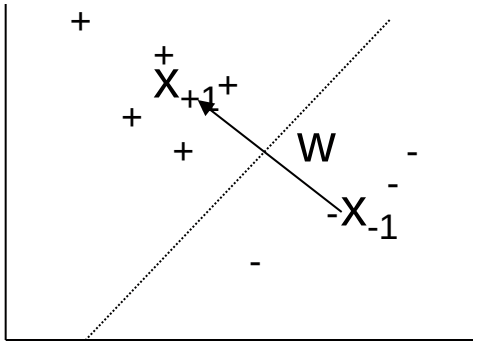
$$= (w_1 \ \ldots \ w_n) \begin{pmatrix} x_1 \\ \ldots \\ x_n \end{pmatrix} + w_0 = (w_0 \ \ w_1 \ \ldots \ w_n) \begin{pmatrix} x_0 = 1 \\ x_1 \\ \ldots \\ x_n \end{pmatrix}$$

$$= \mathbf{w}_{(0..n)}^T \mathbf{x}_{(0..n)}$$

$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \mathbf{x} + w_0$$

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

Lineare Klassifikatoren
Rocchio



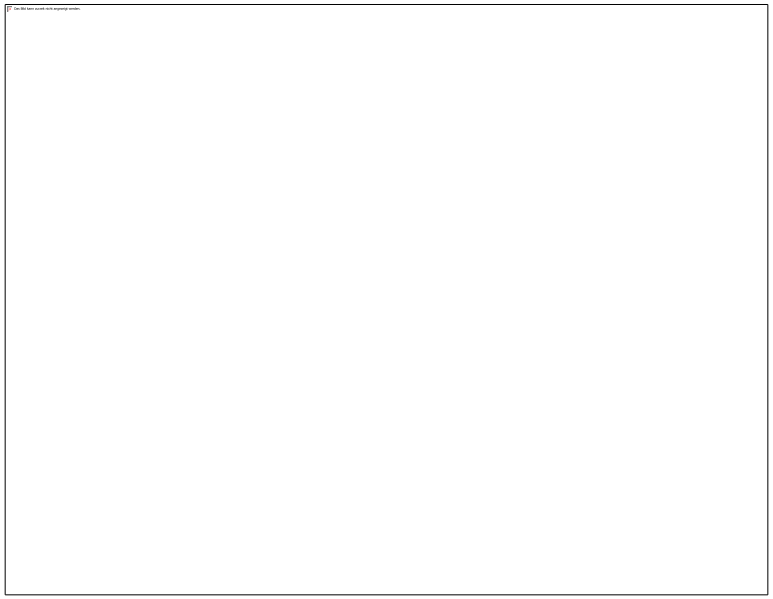
- : Mittelpunkt der neg. Beispiele
- : Mittelpunkt der pos. Beispiele
- Trennebene: Normalenvektor =

Zeigt vom Mittelpunkt der negativen zum Mittelpunkt der positiven Beispiele.

- Bestimmung von w_0: Mittelpunkt muss auf der Ebene liegen.

Lineare Klassifikatoren
Rocchio

- Trennebenen hat maximalen Abstand von den Mittelpunkten der Klassen.
- Trainingsbeispiele können falsch klassifiziert werden.
- Differenz der Mittelwerte kann schlechter Normalenvektor für Diskrimination sein.



Lineare Klassifikatoren
Perzeptron

- Lineares Modell:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$

w_0 in Gewichtsvektor w kodiert.

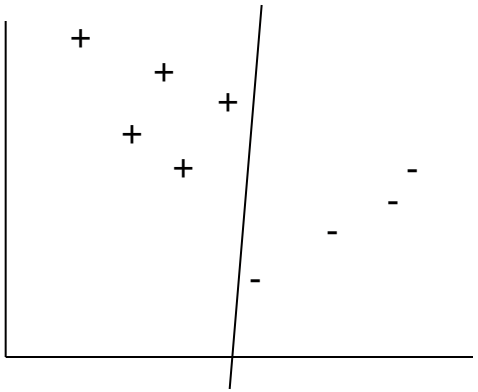
- Ziel:
  - Für alle Beispiele positiven Beispiele $(\mathbf{x}_i, +1)$:

$$f(\mathbf{x}_i) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 0$$

  - Für alle Beispiele negativen Beispiele $(\mathbf{x}_i, -1)$:

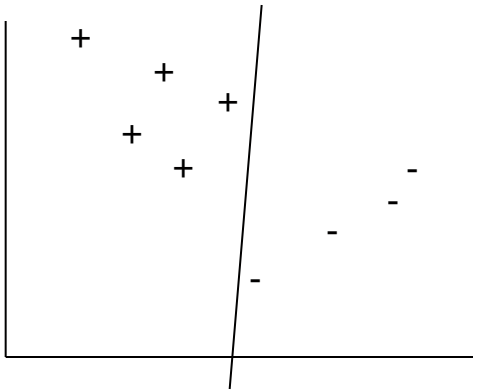
$$f(\mathbf{x}_i) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i < 0$$

  - = Beispiele liegen auf der richtigen Seite der Ebene.



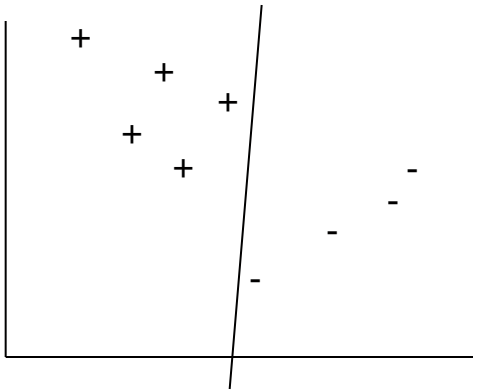
Lineare Klassifikatoren
Perzeptron

- Lineares Modell:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$
- Ziel: Für alle Beispiele $\mathbf{x}_i$ mit Label y_i :
  - $y_i f(\mathbf{x}_i) = y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 0$
- Perzeptron-Optimierungskriterium für Daten L:
  - $J_P(\mathbf{w}) = \sum_{(\mathbf{x}_i, y_i) \in L} \min\{y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i, 0\}$



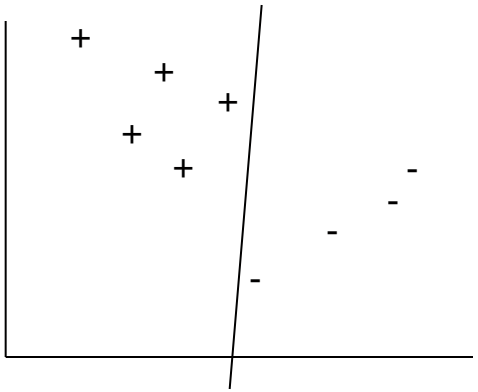
Lineare Klassifikatoren
Perzeptron

- Lineares Modell:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$
- Ziel: Für alle Beispiele $\mathbf{x}_i$ mit Label y_i :
  - $y_i f(\mathbf{x}_i) = y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 0$
- Perzeptron-Optimierungskriterium für Daten L:
  - $J_P(\mathbf{w}) = \sum_{(\mathbf{x}_i, y_i) \in L} \min\{y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i, 0\}$
- Subgradient für Beispiel $(\mathbf{x}_i, y_i)$:
  - $\nabla_i J_P(\mathbf{w}) = \begin{cases} 0, & \text{wenn } y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 0 \\ y_i \mathbf{x}_i & \text{sonst} \end{cases}$



Lineare Klassifikatoren
Perzeptron

- Lineares Modell:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$
- Perzeptron-Optimierungskriterium:
  - $J_P(\mathbf{w}) = \sum_{(\mathbf{x}_i, y_i) \in L} \min\{y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i, 0\}$



- Subgradient für Beispiel $(\mathbf{x}_i, y_i)$:
  - $\nabla_i J_P(\mathbf{w}) = \begin{cases} 0, & \text{wenn } y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 0 \\ y_i \mathbf{x}_i & \text{sonst} \end{cases}$
- Gradientenaufstieg: Wiederhole, für alle Beispiele mit
  -

Verschiebe Trennebene

Werden aktuell falsch klassifiziert.

Lineare Klassifikatoren
Perzeptron-Algorithmus

- Lineares Modell:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$

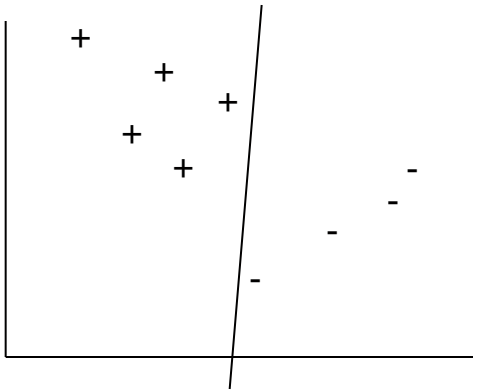
- Perzeptron-Trainingsalgorithmus:
- Solange noch Beispiele $(\mathbf{x}_i, y_i)$ mit der Hypothese inkonsistent sind ($\exists \mathbf{x}_i \in L . y_i \mathbf{w}^T \mathbf{x}_i \leq 0$), iteriere über alle Beispiele:
  - Wenn $y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i \leq 0$ dann $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$.

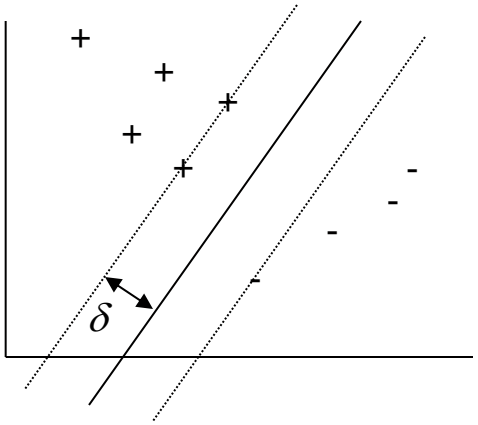
Lineare Klassifikatoren
Perzeptron Eigenschaften

- Perzeptron findet immer eine Trennebene, wenn eine existiert (Optimierungskriterium ist konkav).
- Existiert immer eine Trennebene?

Lineare Klassifikatoren
Margin-Perzeptron

- Perzeptron-Klassifikation:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$
- Perzeptron: für alle Beispiele muss gelten
  - $y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 0$
  - = Beispiel liegt auf der richtigen Seite der Ebene.
- Margin-Perzeptron:
  - $y_i \left(\frac{\mathbf{w}^{\mathrm{T}}}{|\mathbf{w}|} \mathbf{x}_i \right) > \delta$
  - = Beispiel mindestens δ von Trennebene entfernt.





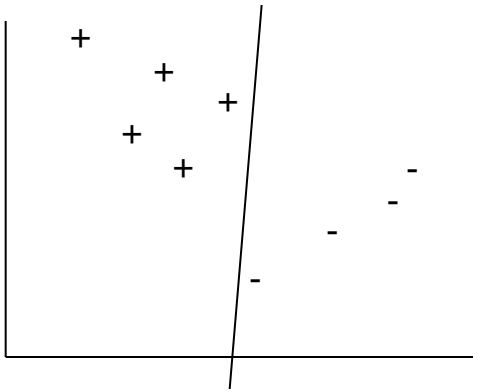
Lineare Klassifikatoren
Margin-Perzeptron-Algorithmus

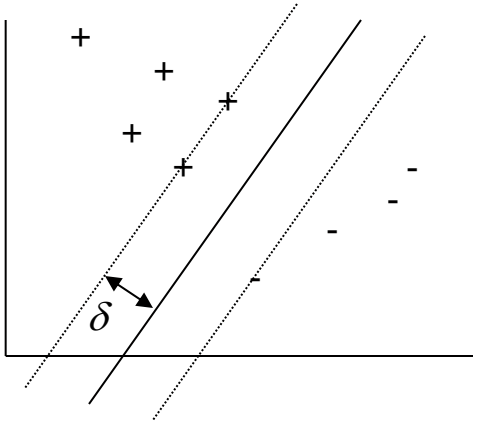
- Lineares Modell:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$

- Margin-Perzeptron-Trainingsalgorithmus:
- Solange noch Beispiele $(\mathbf{x}_i, y_i)$ mit der Hypothese inkonsistent sind($\exists \mathbf{x}_i \in L . y_i \mathbf{w}^T \mathbf{x}_i \leq \delta$), iteriere über alle Beispiele:
  - Wenn $y_i \left(\frac{\mathbf{w}^{\mathrm{T}}}{|\mathbf{w}|} \mathbf{x}_i \right) \leq \delta$ dann $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$

Lineare Klassifikatoren
Margin-Maximierung

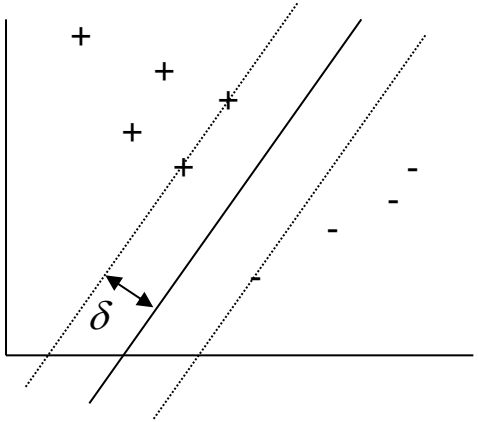
- Perzeptron: für alle Beispiele muss gelten $y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i > 0$
- Margin-Perzeptron: $y_i \left(\frac{\mathbf{w}^{\mathrm{T}}}{|\mathbf{w}|} \mathbf{x}_i \right) > \delta$
  - Finde Ebene, die alle Beispiele mindestens δ von Ebene entfernt.
  - Fester, voreingestellter Wert δ.
- Margin-Maximierung:
  - Finde Ebene, die alle Beispiele mindestens δ von Ebene entfernt.
  - Für den größtmöglichen Wert δ.



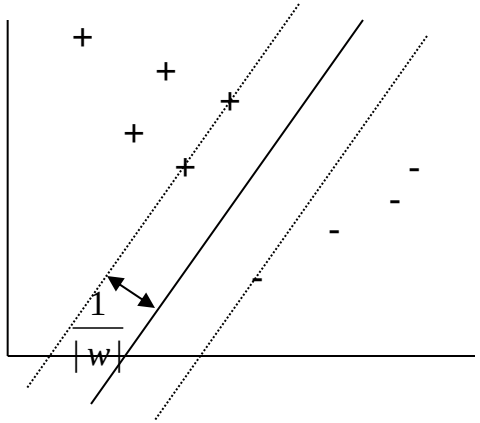


Lineare Klassifikatoren
Margin-Maximierung

- Margin-Maximierung: $y_i\left(\frac{\mathbf{w}^{\mathrm{T}}}{|\mathbf{w}|}\mathbf{x}_i\right) > \delta$
  - Finde Ebene, die alle Beispiele mindestens δ von Ebene entfernt.
  - Für den größtmöglichen Wert δ.



- Maximiere δ unter der Nebenbedingung:
  - für alle Beispiele $(\mathbf{x}_i, y_i)$: $y_i\left(\frac{\mathbf{w}^{\mathrm{T}}}{|\mathbf{w}|}\mathbf{x}_i\right) > \delta$

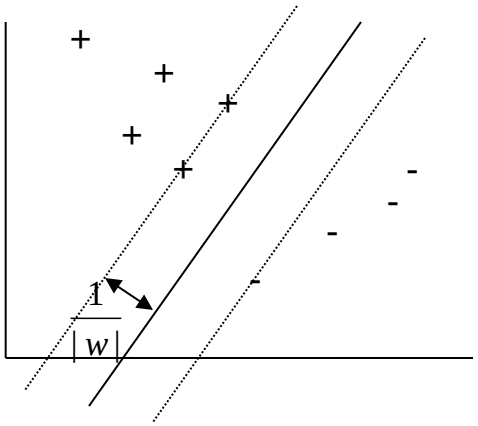


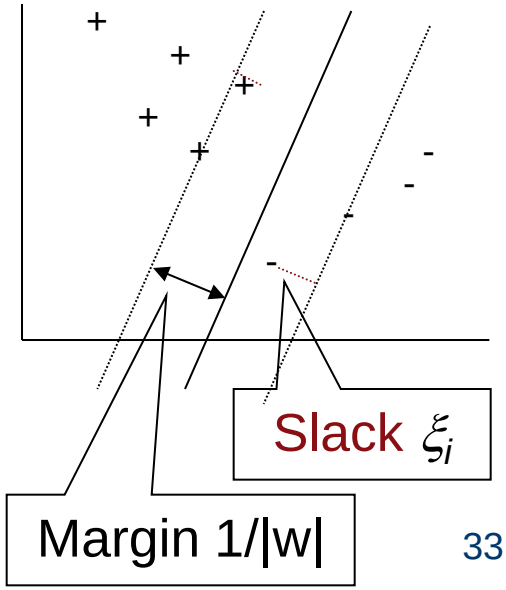
- = Minimiere $|\mathbf{w}|$ unter der Nebenbedingung:
  - für alle Beispiele $(\mathbf{x}_i, y_i)$: $y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 1$

Lineare Klassifikatoren
Margin-Maximierung

- Hard-Margin-Maximierung:
  - Minimiere $|\mathbf{w}|$ unter der Nebenbedingungen:
  - für alle Beispiele $(\mathbf{x}_i, y_i)$: $y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i > 1$
- Soft-Margin-Maximierung:
  - Minimiere $|\mathbf{w}| + C \sum_i \xi_i$ unter den Nebenbedingungen:
  - für alle Beispiele $(\mathbf{x}_i, y_i)$:

    $$y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i > 1 - \xi_i$$
  - Alle $\xi_i \geq 0$.
- Soft-Margin-Ebene existiert immer, Hard-Margin-Ebene nicht!

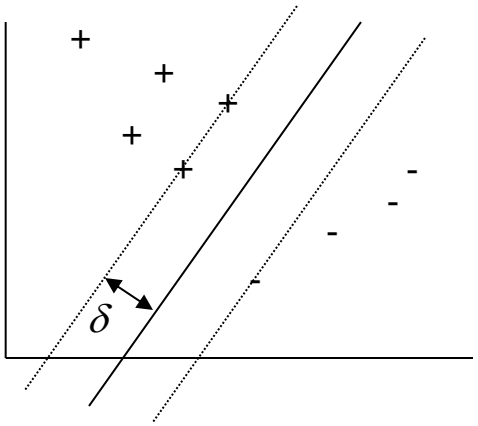


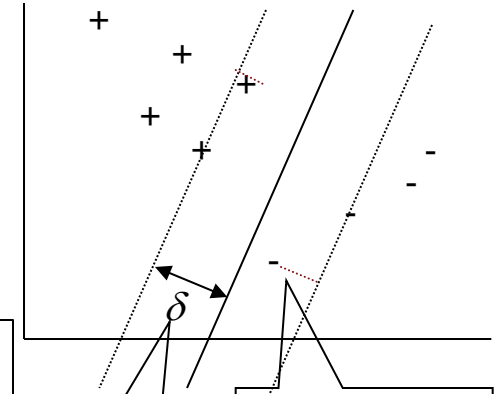


Lineare Klassifikatoren
Soft-Margin-Maximierung

- Soft-Margin-Maximierung:
  - Minimiere $|\mathbf{w}| + C\sum_i \xi_i$ unter den Nebenbedingungen:
  - für alle Beispiele $(\mathbf{x}_i, y_i)$:

    $y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i > 1 - \xi_i$
  - Alle $\xi_i \geq 0$.
- Einsetzen von ξ_i in Optimierungskriterium ergibt
  - Minimiere: $|\mathbf{w}| + C\sum_i \max\{0, 1 - y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i\}$





Regularisierer

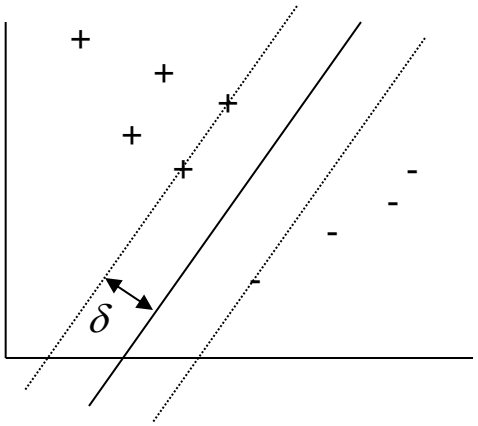
Verlustfunktion

Slack ξ_i

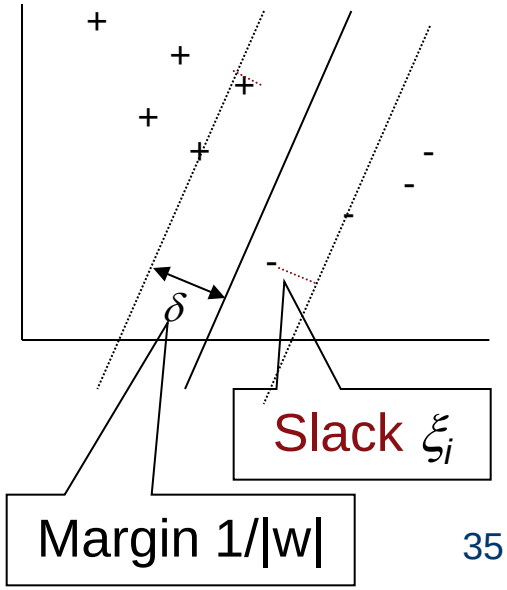
Margin 1/|w|

Lineare Klassifikatoren
Primale Support Vector Machine (SVM)

- Soft-Margin-Maximierung:
  - Minimiere: $|\mathbf{w}| + C\sum_i \max\{0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\}$
- Minimierung mit Gradientenverfahren.

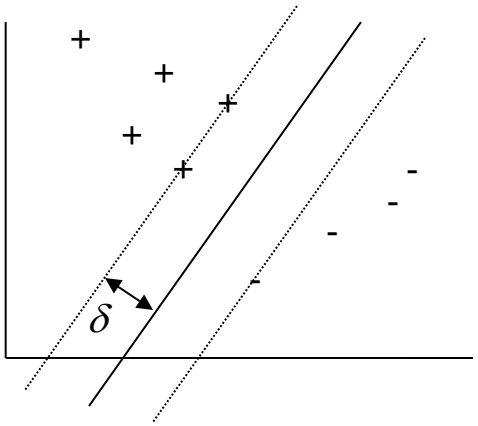


- Kriterium ist konvex, es gibt genau ein Minimum.
- Verfahren: Primale Support Vector Machine.



Lineare Klassifikatoren
Primale Support Vector Machine (SVM)

- Soft-Margin-Maximierung:
  - Minimiere: $E_H(\mathbf{w}) = |\mathbf{w}| + C\sum_i \max\{0, 1 - y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i\}$
- Minimierung mit Gradientenverfahren.
- Wiederhole:
  - $\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial E_H(\mathbf{w})}{\partial \mathbf{w}}$



Enthält Summe über alle Beispiele

Multiklassen-Klassifikation

- Bisher: Binäre Klassifikation
  - $\mathbf{x} \mapsto \{-1,+1\}$
  - Lineare Klassifikation: $\mathbf{x} \mapsto sign(\mathbf{w}^T \mathbf{x})$
- Jetzt: Multiklassen-Klassifikation
  - $\mathbf{x} \mapsto y$
  - Endliche Menge von Klassen-Labels, $y \in Y$
- Ansatz:
  - Statt $\mathbf{x} \mapsto sign(\mathbf{w}^T \mathbf{x})$ jetzt
  - $\mathbf{x} \mapsto \arg\max_y f(\mathbf{x}, y)$

Bestimme Klasse mit höchstem Entscheidungsfunktionswert

Lernen mit strukturierten Ausgaben

- Klassifikation bei mehr als zwei Klassen:
  - $y^* = \arg\max_y f(\mathbf{x}, y)$
  - *f* bekommt jetzt zwei Parameter.
- Gemeinsame Merkmale von Ein- und Ausgabe:
  - $f(\mathbf{x}, y) = \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}, y)$
- Gleicher Ansatz für Multiklassen, Sequenz- und Struktur-Lernen und Ranking.

Lernen mit strukturierten Ausgaben

- Constraints bei normaler SVM:
  - Für alle $(\mathbf{x}_i, y_i)$: $y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i > 1 - \xi_i$
- Constraints mit strukturierten Ausgaben:
  - Für alle $(\mathbf{x}_i, y_i)$: und alle $\bar{y} \neq y_i$:

$$\mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_i, y_i) > \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_i, \bar{y}) + 1 - \xi_i$$

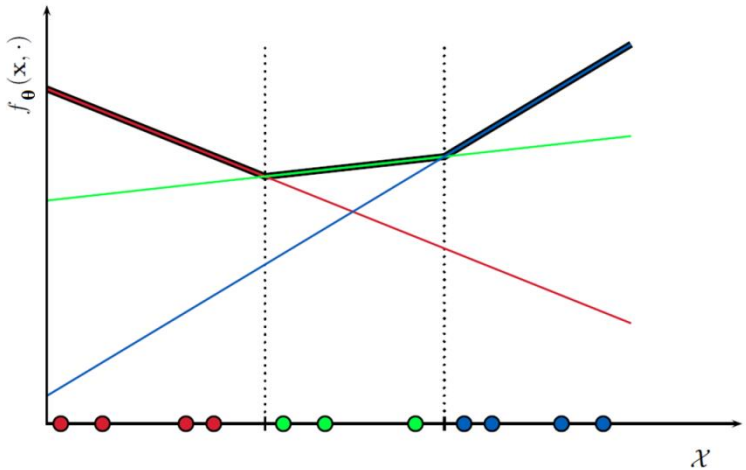
$$\Leftrightarrow \mathbf{w}^{\mathrm{T}} \left(\Phi(\mathbf{x}_i, y_i) - \Phi(\mathbf{x}_i, \bar{y}) \right) > 1 - \xi_i$$

Lernen mit strukturierten Ausgaben
Multiklassen-SVM

- Klassifikation bei mehr als zwei Klassen:
  - $y^* = \arg\max_y f(\mathbf{x}, y)$
- Multiklassen-Merkmale:
  - $f(\mathbf{x}, y) = \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}, y)$
  - $\Lambda(y) = \begin{pmatrix} [[y = y_1]] \\ \dots \\ [[y = y_k]] \end{pmatrix}$
  - $\Phi(\mathbf{x}, y) = \phi(\mathbf{x}) \otimes \Lambda(y) = \begin{pmatrix} \phi(\mathbf{x})[[y = y_1]] \\ \dots \\ \phi(\mathbf{x})[[y = y_k]] \end{pmatrix}$

Lernen mit strukturierten Ausgaben
Multiklassen-SVM

- Jede Klasse hat privaten Abschnitt des Gewichtsvektors:
  - $$\mathbf{w}^{\mathrm{T}}\Phi(\mathbf{x}, y) = \mathbf{w}^{\mathrm{T}}\left(\mathbf{x} \otimes \begin{pmatrix} [[y = y_1]] \\ \dots \\ [[y = y_k]] \end{pmatrix}\right) = \begin{pmatrix} w_{y_1 1} \\ \dots \\ w_{y_1 n} \\ \dots \\ w_{y_k 1} \\ \dots \\ w_{y_k n} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} x_1[[y = y_1]] \\ \dots \\ x_n[[y = y_1]] \\ \dots \\ x_1[[y = y_k]] \\ \dots \\ x_n[[y = y_k]] \end{pmatrix}$$



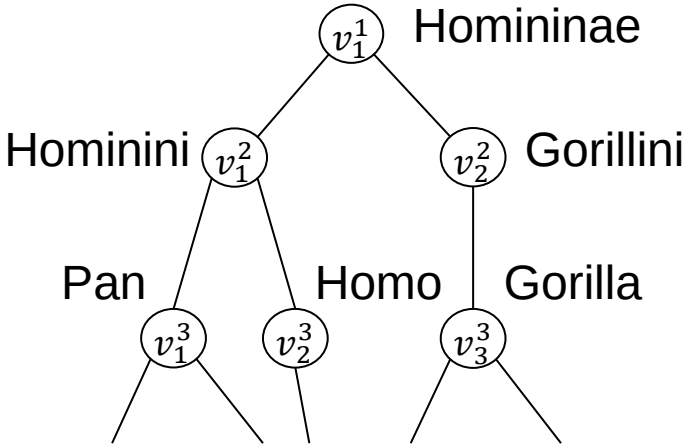
Lernen mit strukturierten Ausgaben
Multiklassen-SVM

- Jede Klasse hat privaten Abschnitt des Gewichtsvektors:
  - Beispiel:

$$\mathbf{w}^{\mathrm{T}}\Phi(\mathbf{x}, y_2) = \mathbf{w}^{\mathrm{T}}\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \otimes \begin{pmatrix} [[y = y_1]] \\ [[y = y_2]] \\ [[y = y_3]] \end{pmatrix}\right) = \begin{pmatrix} w_{y_1 1} \\ w_{y_1 2} \\ w_{y_2 1} \\ w_{y_2 2} \\ w_{y_3 1} \\ w_{y_3 2} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} 0 \\ 0 \\ x_1 \\ x_2 \\ 0 \\ 0 \end{pmatrix}$$

Lernen mit strukturierten Ausgaben
Klassifikation mit Taxonomien

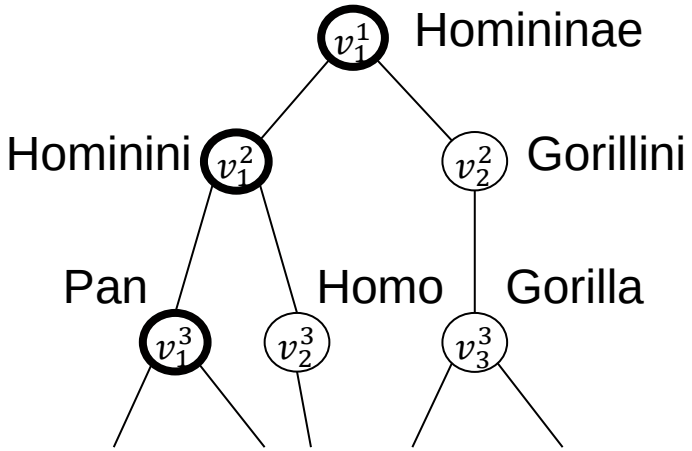
- Angenommen die Ähnlichkeiten der k Klassen sind durch eine Baumstruktur (Tiefe d) gegeben:



- Jede Klasse entspricht einem Pfad im Baum; $\mathbf{y} = \left(y^1, \dots, y^d\right)$.

Lernen mit strukturierten Ausgaben
Klassifikation mit Taxonomien

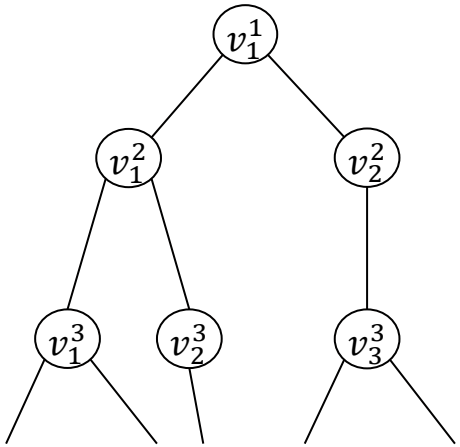
- Angenommen die Ähnlichkeiten der k Klassen sind durch eine Baumstruktur (Tiefe d) gegeben:



- Jede Klasse entspricht einem Pfad im Baum; $\mathbf{y} = (y^1, \dots, y^d)$.

$$Chimpanzee = (Homininae, Hominini, Pan)$$

Lernen mit strukturierten Ausgaben
Klassifikation mit Taxonomien

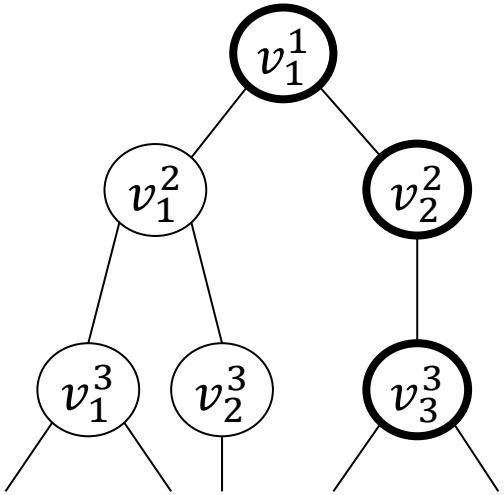


- Klassen in Baumstruktur:
  - $y^* = \arg\max_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$
  - $f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}, \mathbf{y})$
  - $\mathbf{y} = (y^1, \ldots, y^d)$
  - $\Lambda(\mathbf{y}) = \begin{pmatrix} \Lambda(y^1) \\ \ldots \\ \Lambda(y^d) \end{pmatrix}$
  - $\Phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \otimes \Lambda(\mathbf{y}) = \phi(\mathbf{x}) \otimes \begin{pmatrix} \Lambda(y^1) \\ \ldots \\ \Lambda(y^d) \end{pmatrix} = \phi(\mathbf{x}) \otimes \begin{pmatrix} [[y^1 = y_1^1]] \\ \ldots \\ [[y^1 = y_n^1]] \\ \ldots \\ [[y^d = y_1^d]] \\ \ldots \\ [[y^d = y_n^d]] \end{pmatrix}$

Lernen mit strukturierten Ausgaben
Klassifikation mit Taxonomien

- $\mathbf{x}$ kodiert z.B. ein Dokument
- $\mathbf{y} = (v_1^1, v_2^2, v_3^3)^{\mathrm{T}}$ ist ein Pfad z.B. in einem Themenbaum

$$\Phi(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \Lambda[y^1 = v_1^1]\mathbf{x} \\ \Lambda[y^2 = v_1^2]\mathbf{x} \\ \Lambda[y^2 = v_2^2]\mathbf{x} \\ \Lambda[y^3 = v_1^3]\mathbf{x} \\ \Lambda[y^3 = v_2^3]\mathbf{x} \\ \Lambda[y^3 = v_3^3]\mathbf{x} \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \mathbf{0} \\ \mathbf{x} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{x} \\ \vdots \end{pmatrix}$$



Lernen mit strukturierten Ausgaben
Klassifikation mit Taxonomien

- Jeder Knoten hat einen privaten Abschnitt des Gewichtsvektors.
- Pfade teilen sich Abschnitte, wenn sie gemeinsame Knoten beinhalten.

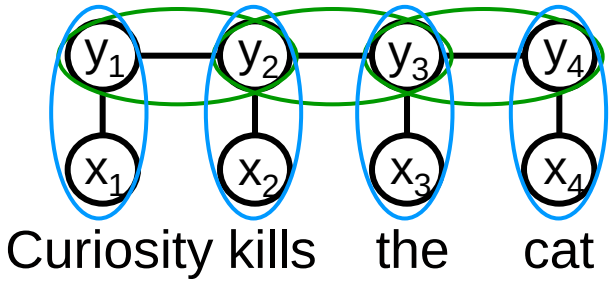
$$\mathbf{w}^{\mathrm{T}}\Phi\left(\mathbf{x},\begin{pmatrix} \mathrm{y}^1 \\ \dots \\ y^d \end{pmatrix}\right) = \mathbf{w}^{\mathrm{T}}\left(\mathbf{x}\otimes\Lambda\left(\begin{pmatrix} \mathrm{y}^1 \\ \dots \\ y^d \end{pmatrix}\right)\right) = \mathbf{w}^{\mathrm{T}}\left(\mathbf{x}\otimes\begin{pmatrix} \Lambda(y^1) \\ \dots \\ \Lambda(y^d) \end{pmatrix}\right)$$

$$= \mathbf{w}^{\mathrm{T}}\left(\mathbf{x}\otimes\begin{pmatrix} [[y^1 = y_1^1]] \\ \dots \\ [[y^1 = y_{k_1}^1]] \\ \dots \\ [[y^d = y_1^d]] \\ \dots \\ [[y^d = y_{k_d}^d]] \end{pmatrix}\right) = \begin{pmatrix} w_{y_1^1 1} \\ \dots \\ w_{y_1^1 n} \\ \dots \\ w_{y_{k_d}^d 1} \\ \dots \\ w_{y_{k_d}^d n} \end{pmatrix}\begin{pmatrix} x_1[[y^1 = y_1^1]] \\ \dots \\ x_n[[y^1 = y_1^1]] \\ \dots \\ x_1[[y^d = y_{k_d}^d]] \\ \dots \\ x_n[[y^d = y_{k_d}^d]] \end{pmatrix} = \sum_{i=1}^{d}\sum_{j=1}^{k_i} \mathbf{w}_{y_j^i}{}^{\mathrm{T}}\mathbf{x}$$

Lernen mit strukturierten Ausgaben
Sequentielle Ein-/Ausgaben

- Z.B. Wortarterkennung:
  - **x**= “Curiosity kills the cat.” → **y**= <Noun, Verb, Determiner, Noun>
- Eigennamenerkennung, Informationsextraktion:
  - **x**= “Barbie meets Ken.” → **y**= <Person, -, Person>
- Gemeinsame Repräsentation von Ein- und Ausgabe.
  - $y^* = \arg\max_y f(\mathbf{x}, \mathbf{y})$
  - $f(\mathbf{x}, y) = \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}, \mathbf{y})$

Lernen mit strukturierten Ausgaben
Sequentielle Ein-/Ausgaben



- Attribut für jedes Paar benachbarter Labels y_t und y_{t+1}.
  - $\varphi_{123}(y_t,y_{t+1})$ = [[y_t="Noun" $\wedge$ y_{t+1}="Verb"]]
- Attribut für jedes Paar aus Eingabe und Ausgabe.
  - $\bar{\varphi}_{234}(x_t,y_t)$ = [[y_t="Noun" $\wedge$ x_t = "cat"]]

- Label-label: $\sum_t \varphi_i(y_t,y_{t+1})$.
- Label-Beobachtung: $\sum_t \bar{\varphi}_i(x_t,y_t)$.

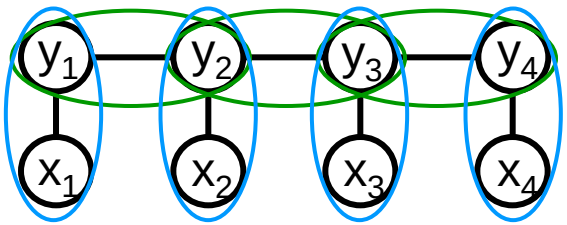
- Gemeinsamer Merkmalsvektor
  $\Phi(\mathbf{x},\mathbf{y})=\sum_t(\ldots,\varphi_{123}(y_t,y_{t+1}),\ldots,\bar{\varphi}_{234}(x_t,y_t),\ldots)^T$
- Gewichtsvektor $\mathbf{w}=(\ldots,w_{123},\ldots,w_{234},\ldots)^T$

Lernen mit strukturierten Ausgaben
Sequentielle Ein-/Ausgaben: Dekodierung

- Um eine Sequenz zu klassifizieren, muss
  - $\mathbf{y}^* = \arg\max_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$
- berechnet werden.
- Das argmax geht über alle möglichen Sequenzen (exponentiell viele in der Länge).
- $f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}, \mathbf{y})$ summiert über Merkmale benachbarter Label-Paare und Merkmale von x_i-y_i-Paaren.
- Mit dynamischer Programmierung kann argmax in linearer Zeit berechnet werden (Viterbi).

Lernen mit strukturierten Ausgaben
Sequentielle Ein-/Ausgaben: Dekodierung

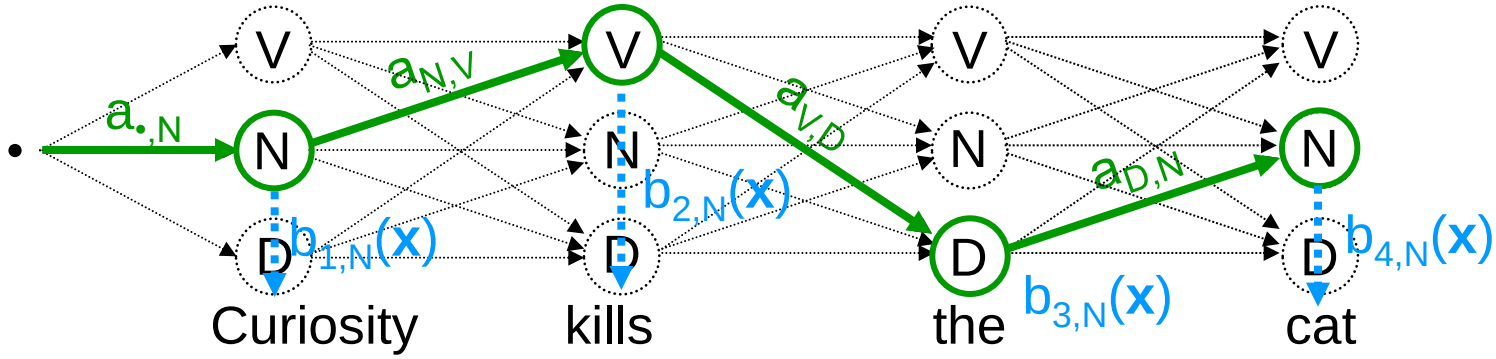
- Gemeinsamer Merkmalsvektor $\Phi(\mathbf{x},\mathbf{y})=\sum_t(\ldots,\varphi_{123}(y_t,y_{t+1}),\ldots,\varphi_{234}(x_t,y_t),\ldots)^T$



$\varphi_{123}(y_t,y_{t+1}) = [[y_t=\text{“Noun”} \wedge y_{t+1}=\text{“Verb”}]]$

$\bar{\varphi}_{234}(x_t,y_t) = [[y_t=\text{“Noun”} \wedge x_t = \text{“John”}]]$

- Finde $\mathbf{argmax}_{\mathbf{y}}\ \mathbf{w}^T\Phi(\mathbf{x},\mathbf{y})$ effizient mit Transitionsmatrix $A=\{a_{\sigma,\tau}\}$ und Beobachtungsmatrix $B_{\mathbf{x}}=\{b_{t,\sigma}(\mathbf{x})\}$, $\sigma,\tau \in\{\bullet,N,V,D\}$:



- HM SVM benutzt 2-best Viterbi Dekodierung.

Lernen mit strukturierten Ausgaben

- Beispiel: POS-Tagging (Wortarterkennung)
- Satz **x**=“Curiosity kills the cat”
- Gewünscht:

$$\mathbf{argmax}_{\mathbf{y}}\ \mathbf{w}^T\Phi(\mathbf{x}, \mathbf{y}) = \langle N,V,Det,N\rangle$$

- Explizit:

$$\mathbf{w}^T\Phi(\mathbf{x},\langle N,V,Det,N\rangle) \geq \mathbf{w}^T\Phi(\mathbf{x},\langle N,N,N,N\rangle)$$

$$\mathbf{w}^T\Phi(\mathbf{x},\langle N,V,Det,N\rangle) \geq \mathbf{w}^T\Phi(\mathbf{x},\langle N,N,N,V\rangle)$$

$$\mathbf{w}^T\Phi(\mathbf{x},\langle N,V,Det,N\rangle) \geq \mathbf{w}^T\Phi(\mathbf{x},\langle N,N,V,N\rangle)$$

$$\mathbf{w}^T\Phi(\mathbf{x},\langle N,V,Det,N\rangle) \geq \mathbf{w}^T\Phi(\mathbf{x},\langle N,V,N,N\rangle)$$

$$\vdots \qquad\qquad \vdots$$

Lernen mit strukturierten Ausgaben
Trainingsalgorithmus

- Large-Margin-Ansatz: $\gamma = 1/|w|$.
  - $\min \ \frac{1}{2} |\mathbf{w}|^2 + C\sum_i \xi_i$

    so dass $\forall i \ \forall \overline{\mathbf{y}} \neq \mathbf{y}_i \quad \mathbf{w}^T\left(\Phi(\mathbf{x}_i,\mathbf{y}_i) - \Phi(\mathbf{x}_i,\overline{\mathbf{y}})\right) \geq 1 - \xi_i$

    $\forall i \quad \xi_i \geq 0.$

- Iteratives Training.
  - Negative Constraints werden hinzugefügt, wenn beim Training Fehler auftritt.

Lernen mit strukturierten Ausgaben
Trainingsalgorithmus

- Gegeben: $L = \langle (\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m) \rangle$
- Wiederhole bis alle Sequenzen korrekt vorhergesagt werden.
  - Iteriere über alle Beispiele ($\mathbf{x}_i$, $\mathbf{y}_i$).
    - Bestimme $\bar{\mathbf{y}} = \arg\max_{\mathbf{y} \neq \mathbf{y}_i} \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_i, \mathbf{y})$
    - Wenn $\mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_i, \mathbf{y}_i) < \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) + 1$ (Margin-Verletzung) dann füge Constraint $\mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_i, \bar{\mathbf{y}}) + 1 - \xi_i$ dem Working Set hinzu.
    - Löse Optimierungsproblem für Eingabe $\mathbf{x}_i$, Ausgabe $\mathbf{y}_i$, und negative Pseudo-Beispiele $\bar{\mathbf{y}}$ (working set).
- Liefere **w** zurück.

Lernen mit strukturierten Ausgaben
Erweiterung: Verlustfunktion

- Problem: Alle Fehler gleich.
  - Oft nicht sinnvoll beim Strutkurlernen.
- Lösung: Verlustfunktion.

$$\min \quad \tfrac{1}{2}\,|\mathbf{w}|^2 + C\sum_i \xi_i$$

$$\text{so dass} \quad \forall i\ \forall \bar{\mathbf{y}} \neq \mathbf{y}_i \qquad \mathbf{w}^T\big(\Phi(\mathbf{x}_i,\mathbf{y}_i) - \Phi(\mathbf{x}_i,\bar{\mathbf{y}})\big) \geq \overbrace{\Delta(\mathbf{y}_i, \bar{\mathbf{y}})}^{\text{Verlustfunktion}} - \xi_i$$

$$\forall i \qquad \xi_i \geq 0.$$

- Verlustfunktion: Bestraft falsche $\bar{\mathbf{y}}$ Vorhersage in Bezug auf das Ziellabel $\mathbf{y}_i$.

Lineare Klassifikatoren
Logistische Regression

- SVM: großer Entscheidungsfunktionswert ~ hohe Sicherheit der Vorhersage.
- Aber: beim Lernen nicht auf korrekte Kalibrierung der Klassenwahrscheinlichkeiten optimiert.
- $f(\mathbf{x})=18.3 \rightarrow$ Risiko eines Fehlers?
- Problem: Keine korrekt kalibrierten Entscheidungsfunktionswerte.
- Lösung: Logistische Regression.

- Logistische Regression: Vorhersage der Klassenwahrscheinlichkeit.

Lineare Klassifikatoren
Logistische Regression

- Bayes‘ Regel:
  - $$P(y=+1\,|\,\mathbf{x}) = \frac{p(\mathbf{x}\,|\,y=+1)P(y=+1)}{p(\mathbf{x}\,|\,y=+1)P(y=+1)+p(\mathbf{x}\,|\,y=-1)P(y=-1)}$$
    $$= \frac{1}{1+\dfrac{p(\mathbf{x}\,|\,y=-1)P(y=-1)}{p(\mathbf{x}\,|\,y=+1)P(y=+1)}} = \frac{1}{1+\exp(-a)} = \sigma(a)$$
- Log-odd ratio:
  - $$a = \ln\frac{p(\mathbf{x}\,|\,y=+1)P(y=+1)}{p(\mathbf{x}\,|\,y=-1)P(y=-1)}$$

Lineare Klassifikatoren
Logistische Regression

- Likelihood jeder Klasse normalverteilt, gemeinsame Kovarianzmatrix für beide Klassen.
  - $$p(\mathbf{x} \mid y) = \frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x} - \mu)^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \mu) \right]$$

Normalverteilung

- Logg-odds ratio:

$$a = \ln \frac{\frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x} - \mu_{+1})^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \mu_{+1}) \right] P(y = +1)}{\frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x} - \mu_{-1})^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \mu_{-1}) \right] P(y = -1)}$$

$$= \left[-\frac{1}{2} (\mathbf{x} - \mu_{+1})^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \mu_{+1}) \right] - \left[-\frac{1}{2} (\mathbf{x} - \mu_{-1})^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \mu_{-1}) \right] + \ln \frac{P(y = +1)}{P(y = -1)}$$

$$= \underbrace{\left[\Sigma^{-1} (\mu_{+1} - \mu_{-1}) \right]}_{\mathbf{w}} \mathbf{x} + \underbrace{\left[-\frac{1}{2} (\mu_{+1}^{\mathrm{T}} \Sigma^{-1} \mu_{+1}) + \frac{1}{2} (\mu_{-1}^{\mathrm{T}} \Sigma^{-1} \mu_{-1}) + \ln \frac{P(y = +1)}{P(y = -1)} \right]}_{w_0}$$

Lineare Klassifikatoren
Logistische Regression

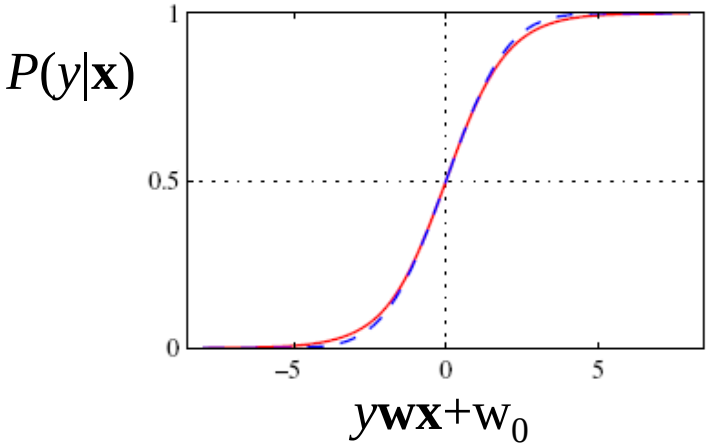
- Likelihood jeder Klasse normalverteilt, gemeinsame Kovarianzmatrix für beide Klassen.
  - $$p(\mathbf{x} \mid y) = \frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x} - \mu)^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \mu) \right]$$

- Logg-odds ratio:

$$a = \underbrace{\left[\Sigma^{-1} (\mu_{+1} - \mu_{-1}) \right]}_{\mathbf{w}} \mathbf{x} + \underbrace{\left[-\frac{1}{2} (\mu_{+1}^{\mathrm{T}} \Sigma^{-1} \mu_{+1}) + \frac{1}{2} (\mu_{-1}^{\mathrm{T}} \Sigma^{-1} \mu_{-1}) + \ln \frac{P(y = +1)}{P(y = -1)} \right]}_{w_0}$$

Lineare Klassifikatoren
Logistische Regression

- Wenn zwei Klassen jeweils normalverteilte Likelihood mit derselben Kovarianzmatrix haben, dann nimmt $P(y|\mathbf{x})$ diese Form an:
  - $$P(y\,|\,\mathbf{x}) = \frac{1}{1+\exp(-y\mathbf{w}\mathbf{x}+w_0)} = \underbrace{\sigma(\underbrace{y\mathbf{w}\mathbf{x}+w_0}_{\text{linearer Klassifikator}})}_{\text{logistische Funktion}}$$



Lineare Klassifikatoren
Logistische Regression

- Bisher: Motivation der Form des logistischen Klassifikationsmodells.
- Falls Klassenverteilungen bekannt wären, könnten wir $\mathbf{w}$ und w_0 aus μ_{+1}, μ_{-1} und Σ herleiten.
- Sind aber nicht bekannt. Verteilungsannahme muss auch nicht stimmen.
- Jetzt: Wie finden wir tatsächlich Parameter $\mathbf{w}$ und w_0?

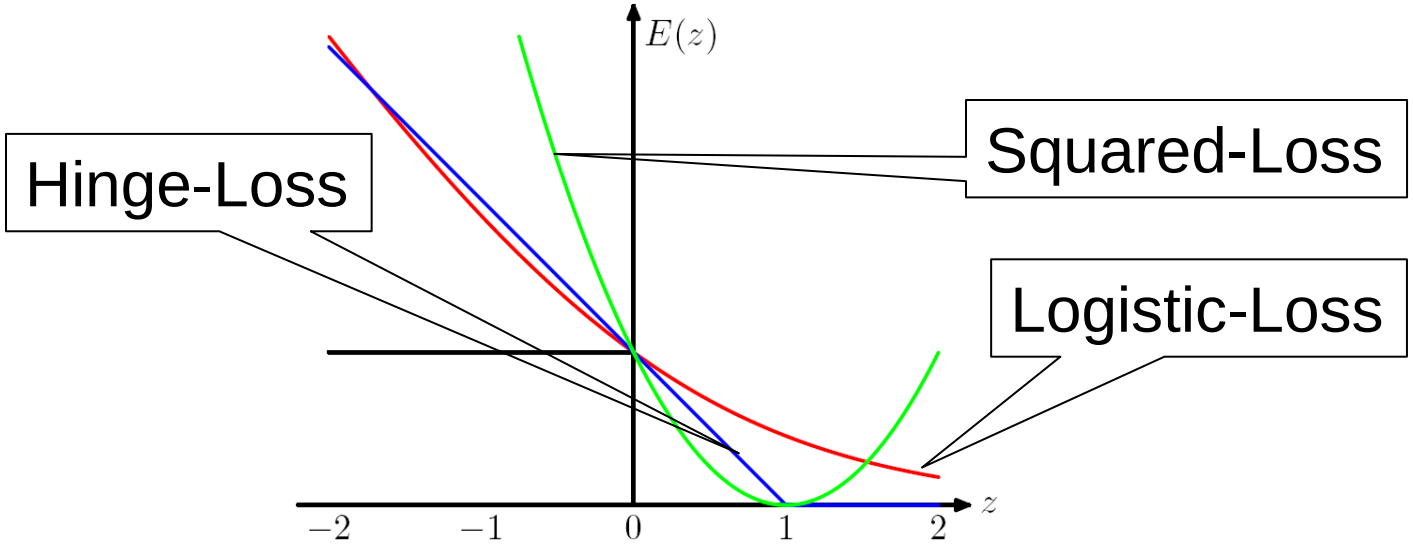
Lineare Klassifikatoren
Logistische Regression

- Prior über Parameter:
  - Normalverteilung, $\mathbf{w} \sim \mathrm{N}[0,\Sigma]$.
- Posterior:
  - $$P(\mathbf{w} \mid L) \approx \prod_{i=1}^{N} p(y_i \mid \mathbf{x}_i, \mathbf{w}) p(\mathbf{w})$$
    $$= \prod_{i=1}^{N} \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i)^{[[y_i=+1]]} (1-\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i))^{[[y_i=-1]]} p(\mathbf{w})$$
- Verlustfunktion:
  - $$E(\mathbf{w}, L) = -\log p(\mathbf{w} \mid L)$$
    $$= -\sum_{i=1}^{N} [[y_i = +1]] \log \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i) + [[y_i = -1]] \log(1-\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i)) - \frac{\mathbf{w}^{\mathrm{T}}\mathbf{w}}{\sigma'^2}$$

Lineare Klassifikatoren
Logistische Regression

- Verlustfunktion ist konvex und differenzierbar.
- Gradientenabstieg führt zum Minimum.

- Verlustfunktionen Logistic Regression und SVM



Fragen?