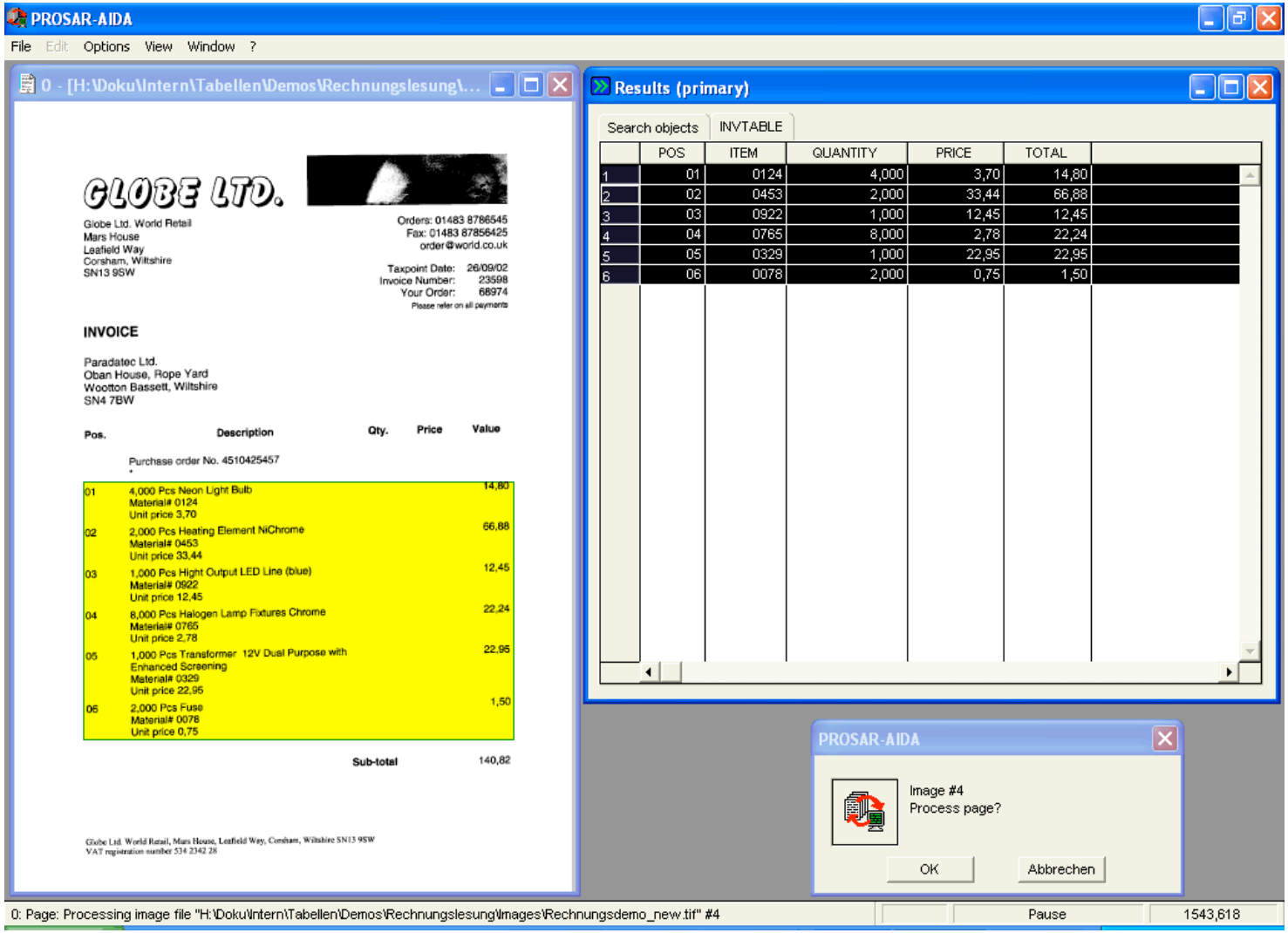
Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Textklassifikation, Informationsextraktion

Tobias Scheffer
Thomas Vanck

Textklassifikation, Informationsextraktion



Textklassifikation, Informationsextraktion

- Textklassifikator: Ordnet einen Text einer Menge von inhaltlichen Kategorien zu.
- Wird häufig aus Prozessdaten gelernt.
- Anwendungsbeispiel: Posteingangsverarbeitung.

- Informationsextraktion: Identifikation definierter Felder in Dokument.
- Wird häufig aus Prozessdaten gelernt.
- Anwendungsbeispiel: Automatisierung von Dokumentenverarbeitungsprozessen.

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.

Repräsentation

- Jedes N-Gramm durch Dimension repräsentiert,
  - Nur N-Gramme, die auch tatsächlich auftreten.
- Sparse Binary Polynomial Hashing
  - Fenster der Breite N wird über Text geschoben,
  - Jede Teilmenge von bis zu N Tokens, die in dieser Reihenfolge auftauchen und das am weitesten rechts stehende Token beinhalten, ist eine Dimension,
- Orthogonal Sparse Bigrams.
  - Fenster der Breite N wird über Text geschoben,
  - Jedes Paar aus einem beliebigen Token im Fenster und dem am rechten Fensterrand stehenden Token ist ein Merkmal.
- SBPH und OSB: N-Gramme mit Platzhaltern.

Klassifikator / Entscheidungsfunktion

- Für eine binäre Klassifikation (y= +1 oder -1) wird eine Entscheidungsfunktion f(x) gelernt.
- Je größer f(x), desto wahrscheinlicher ist, dass x zur Klasse 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.

Evaluation von Textklassifikatoren

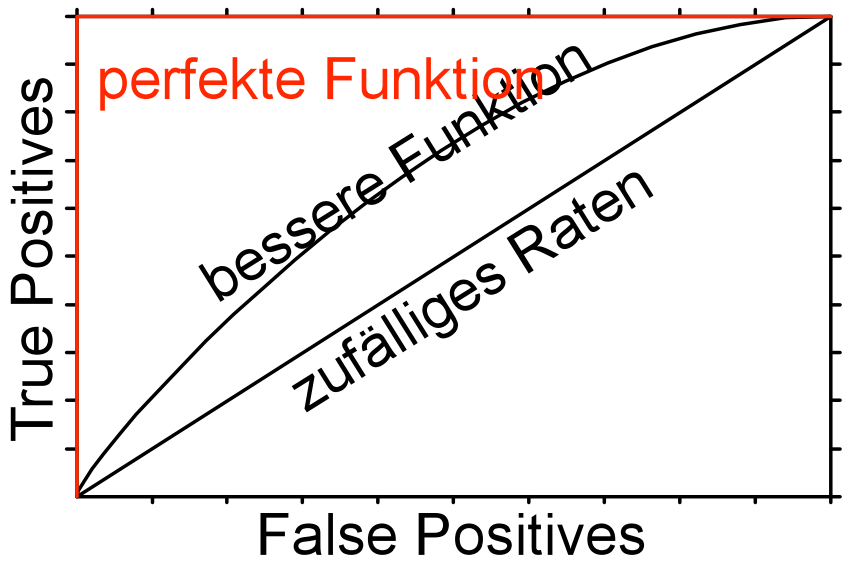
- Fehlklassifikationswahrscheinlichkeit
  - Häufig nicht aussagekräftig, weil P(+1) sehr klein.
  - Wie gut sind 5% Fehler, wenn P(+1)=3%?
  - Idee: Nicht Klassifikator bewerten, sondern Entscheidungsfunktion.
- Precision / Recall
  - Precision-Recall-Kurve bewertet Entscheidungsfunktion,
  - Jeder Wert für θ entspricht Punkt auf P-R-Kurve.
  - F-Measure: „Durchschnitt“ aus Precision und Recall.
- Receiver Operating Characteristic (ROC-Kurve)
  - Bewertet Entscheidungsfunktion,
  - Fläche unter ROC-Kurve = P(positives Beispiel hat höheren f-Wert als negatives Beispiel)

ROC-Analyse

- Entscheidungsfunktion + Schwellwert = Klassifikator.
- Großer Schwellwert: Mehr positive Bsp falsch.
- Kleiner Schwellwert: Mehr negative Bsp falsch.
- Bewertung der Entscheidungsfunktion unabhängig vom konkreten Schwellwert.
- Receiver-Operating-Characteristic-Analyse
- Kommt aus der Radartechnik, im 2. Weltkrieg entwickelt.
- Bewertung der Qualität von Entscheidungsfunktionen.

ROC-Kurven

- 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.



Bestimmen der ROC-Kurve von f

- Für alle positiven Beispiele xp in Testmenge
  - Füge f(xp) in sortierte Liste Lp ein.
- Für alle negativen Beispiele xn in Testmenge
  - Füge f(xn) in sortierte Liste Ln ein.
- tp = fp = 0
- Wiederhole solange Lp und Ln nicht leer sind.
  - Wenn Lp → Element ≥ Ln → Element dann increment(tp) und Lp = LP → Next.
  - Wenn Ln → Element ≥ Lp → Element dann increment(fp) und Ln = Ln → Next.
  - Zeichne neuen Punkt (fp, tp)

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.

Precision / Recall

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

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

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

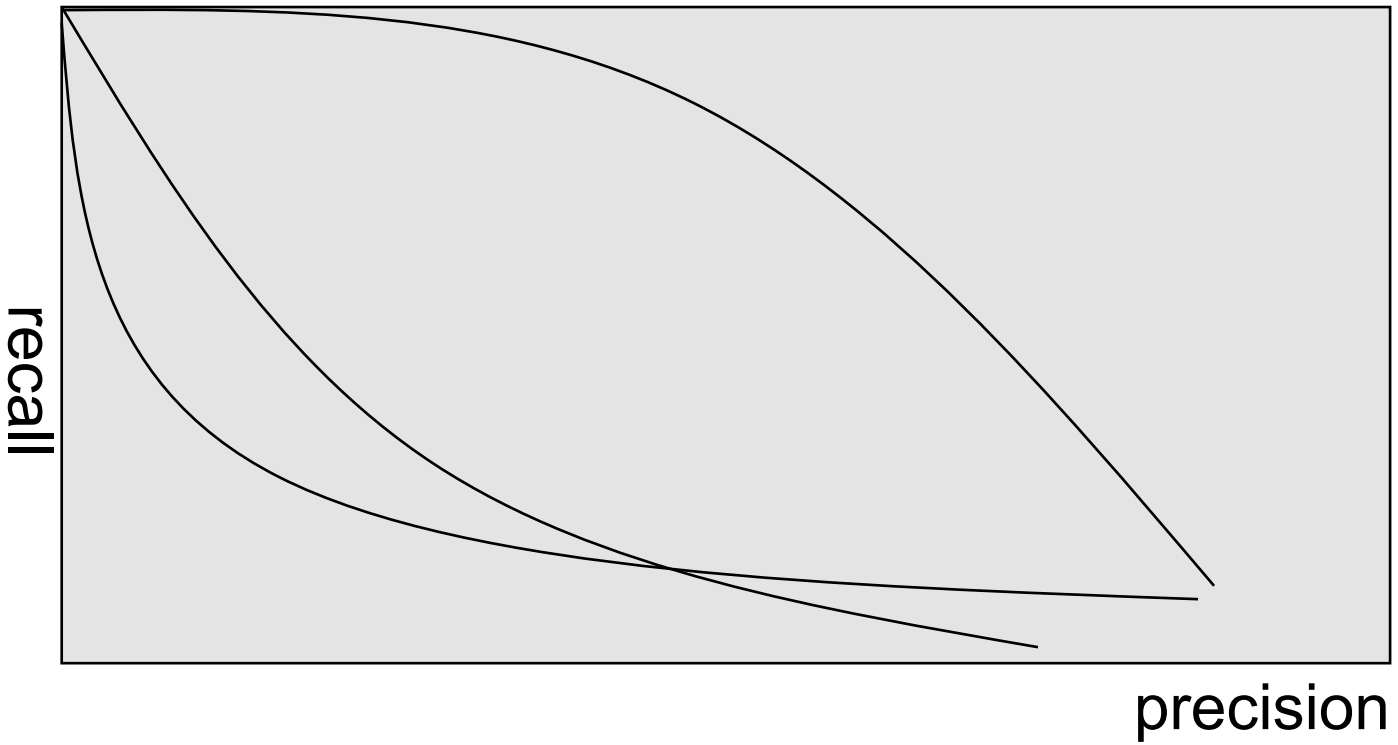
Precision / Recall

- Zusammenfassungen der Kurve in einer Zahl:
  - 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 PRBEP für den gilt:
    Precision(θ) = Recall(θ) = PRBEP.

Precision / Recall Trade-Off



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

Fehlerschätzung

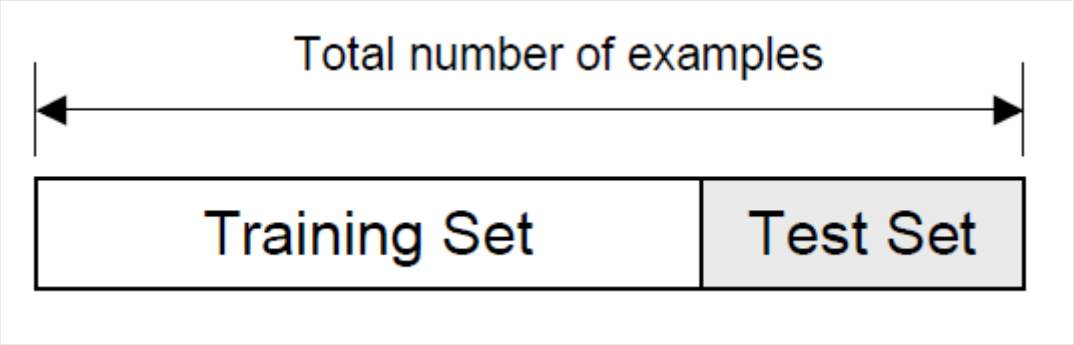
- Wir haben Klassifikator h gelernt, und wollen nun wissen wie gut er ist.
- Echter Fehler E ist unbekannt.
- Verwende Fehlerschätzer $\hat{E}$
- Bias = $\mathbb{E}[\hat{E}] - E = \begin{cases} < 0 & \text{optimistisch} \\ > 0 & \text{pessimistisch} \\ = 0 & \text{erwartungstreu} \end{cases}$
- $Var[\hat{E}] = \begin{cases} \text{groß bei wenigen Daten} \Rightarrow \text{Schätzung unsicher} \\ \text{klein bei vielen Daten} \Rightarrow \text{Schätzung sicher} \end{cases}$

Fehlerschätzung

- Problem: Fehlerschätzung nur auf den Trainingsdaten
- Empirischen Fehler: Messe wie viele Datenpunkte falsch klassifiziert werden.
- Es kann sein, dass $\mathbb{E}[\hat{E}] = 0$ (Overfitting)
  - Dann gilt für den Bias = $\mathbb{E}[\hat{E}] - E < 0$
  - D.h. wir sind sehr optimistisch, was unseren Fehler angeht, da wir ihn geringer schätzen als den tatsächlichen echten Fehler.
- Deshalb: Fehlerschätzung auf Trainingsdaten schlecht!

Holdout-Testing

- Besser: Verwende Testdaten, die nicht zum Trainieren den Klassifikators benutzt wurden.
- Training-und-Test:
  - Verwende 80% der Daten zum Trainieren
  - Verwende 20% der Daten zum Testen
- Auf 20% der Daten wird dann $\hat{E}$, ROC-Kurve, PR-Kurve ermittelt.



Holdout-Testing

- Algorithmus Training-and-Test
  - Aufteilen der Datenbank in Trainingsmenge und Testmenge.
  - h1 = Lernalgorithmus (Trainingsmenge)
  - Bestimme Ê anhand der Testmenge.
  - h = Lernalgorithmus (alle Beispiele)
  - Liefere Hypothese h zusammen mit Fehlerschätzer

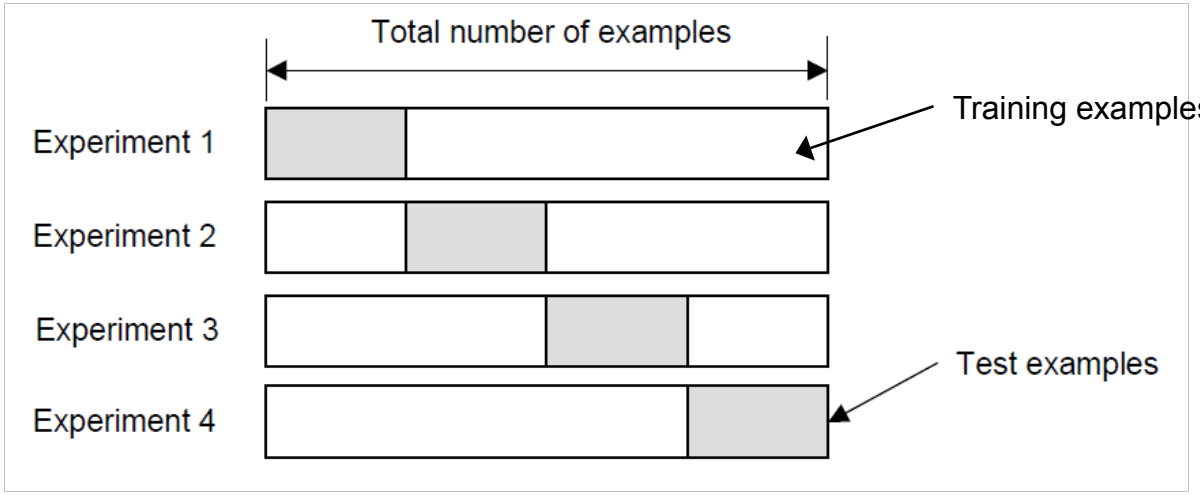
$$\hat{E} \pm z_{\delta/2}\sqrt{\frac{\hat{E}(1-\hat{E})}{20\% \cdot m}}$$

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

N-Fold Cross-Validation

- Idee: Teile Daten D in N gleichgroße Teilstücke

- Trainiere auf D\Di und teste dann auf Di
- Iteriere wie folgt:



N-Fold Cross-Validation

- Algorithmus N-CV (Beispiele D)
  - Bilde N etwa gleich große Blöcke D1,...,Dn von Beispielen, die zusammen D ergeben. Ê=0.
  - For i = 1 ... N
    - h = Lernalgorithmus (D \ Di)
    - Ê = Ê + empirischer Fehler von h auf Di
  - Ê = Ê/N
  - h = Lernalgorithmus (D)
  - Liefere Hypothese h mit Fehlerschätzer

$$\hat{E} \pm z_{\delta/2} \cdot \sqrt{\frac{S_N}{N}}$$

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

Was man über Hyperebenen wissen sollte…

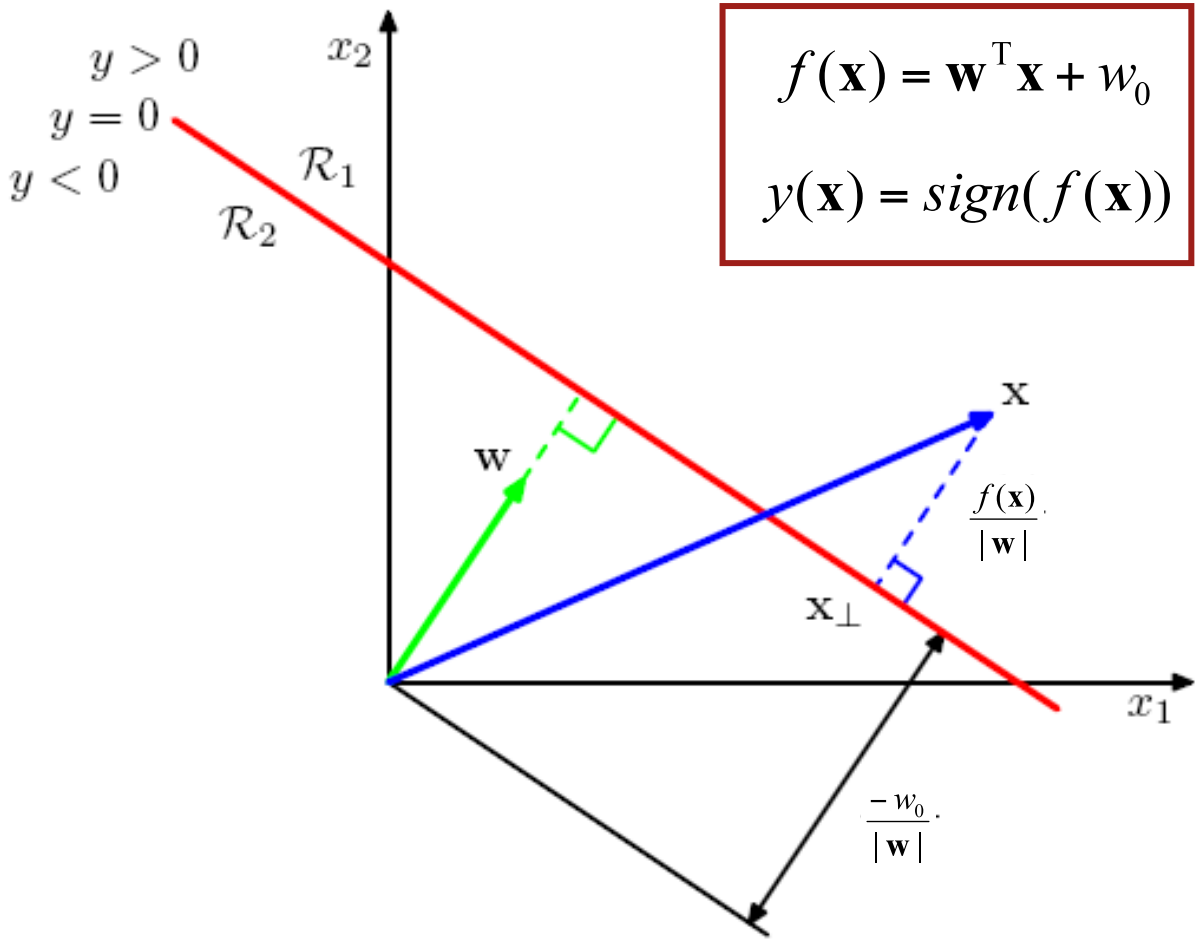
- Wir beschreiben eine Hyperebene durch $\langle \mathbf{w}, \mathbf{x} \rangle = 0$, wobei $\mathbf{x}, \mathbf{w} \in \mathbb{R}^d$

1. Was ist $\mathbf{w}$?
2. Wann gilt $\langle \mathbf{w}, \mathbf{x} \rangle > 0$ bzw. $\langle \mathbf{w}, \mathbf{x} \rangle < 0$

- Die Ebene ist ein Model, mit dem klassifiziert werden kann

1. Wie bestimmt man eine derartige Ebene für eine Menge $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
2. Wie viele gibt es?
3. Welche ist die beste (für ungesehene Daten)?

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:
  - $f(\mathbf{x}) = \mathbf{w}_{(1..n)}^{\mathrm{T}} \mathbf{x}_{(1..n)} + w_0$

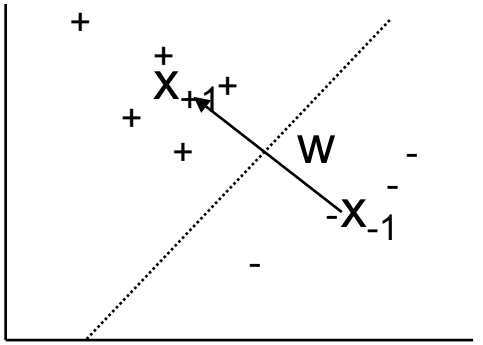
$$= (w_1 \ \ldots \ w_n) \begin{pmatrix} x_1 \\ \ldots \\ x_n \end{pmatrix} + w_0 = (w_0 \ \ w_1 \ \ldots \ w_n) \begin{pmatrix} x_0 \\ x_1 \\ \ldots \\ x_n \end{pmatrix}$$

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

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

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

Rocchio



- $\overline{\mathbf{x}}_{-1}$: Mittelpunkt der neg. Beispiele
- $\overline{\mathbf{x}}_{+1}$: Mittelpunkt der pos. Beispiele
- Trennebene: Normalenvektor = $(\mathbf{x}_{+1}-\mathbf{x}_{-1})$

$$f(\mathbf{x}) = (\overline{\mathbf{x}}_{+1} - \overline{\mathbf{x}}_{-1})\mathbf{x} + w_0$$

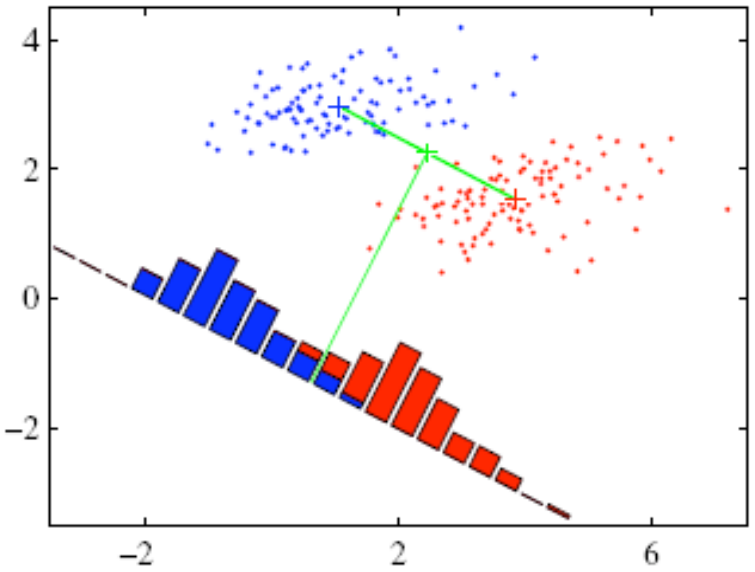
- Bestimmung von w_0: Mittelpunkt $(\mathbf{x}_{+1}+\mathbf{x}_{-1})/2$ muss auf der Ebene liegen.

$$f((\overline{\mathbf{x}}_{-1} + \overline{\mathbf{x}}_{+1})/2) = (\overline{\mathbf{x}}_{+1} - \overline{\mathbf{x}}_{-1})(\overline{\mathbf{x}}_{-1} + \overline{\mathbf{x}}_{+1})/2 + w_0 = 0$$

$$\Leftrightarrow w_0 =$$

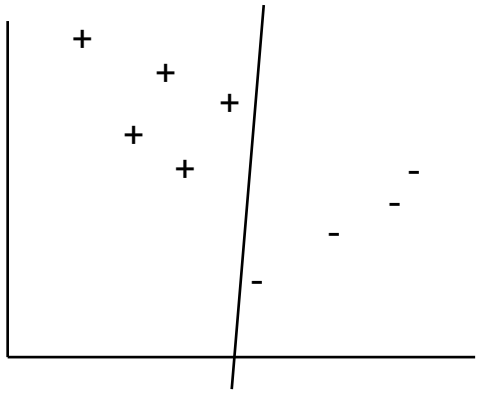
Rocchio

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



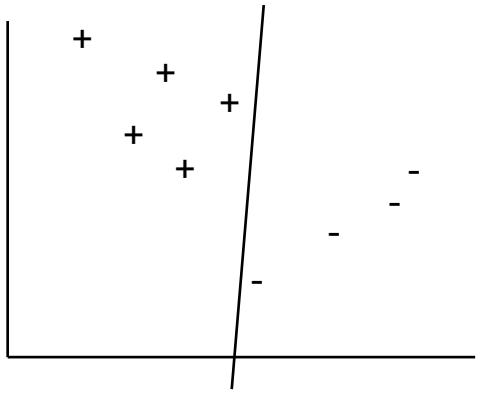
Perzeptron

- Lineares Modell:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$
- Ziel:
  - Für alle Beispiele $(\mathbf{x}_i, +1)$:
    $$f(\mathbf{x}_i) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 0$$
  - Für alle Beispiele $(\mathbf{x}_i, -1)$:
    $$f(\mathbf{x}_i) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i < 0$$



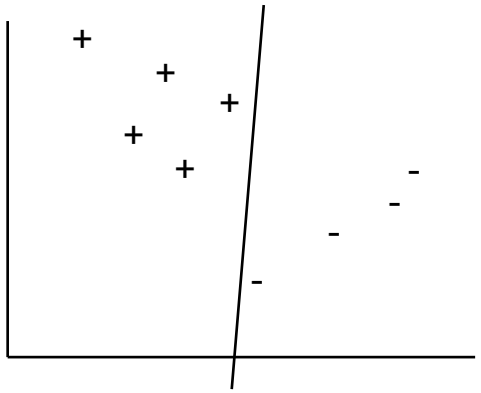
Perzeptron

- Lineares Modell:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$
- Ziel:
  - Für alle Beispiele $(\mathbf{x}_i, +1)$:
    $$f(\mathbf{x}_i) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 0$$
  - Für alle Beispiele $(\mathbf{x}_i, -1)$:
    $$f(\mathbf{x}_i) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i < 0$$
- Ziel: Für alle Beispiele:
  - $y_i f(\mathbf{x}_i) = y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 0$
  - = Beispiel liegt auf der richtigen Seite der Ebene.



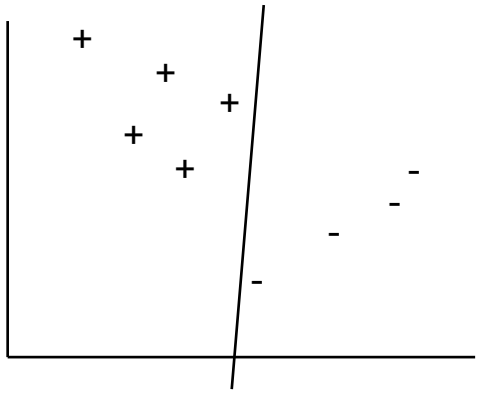
Perzeptron

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



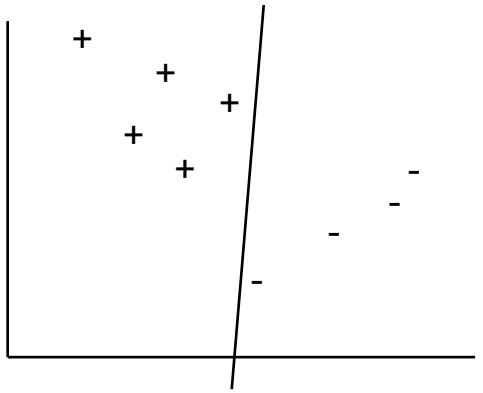
Perzeptron

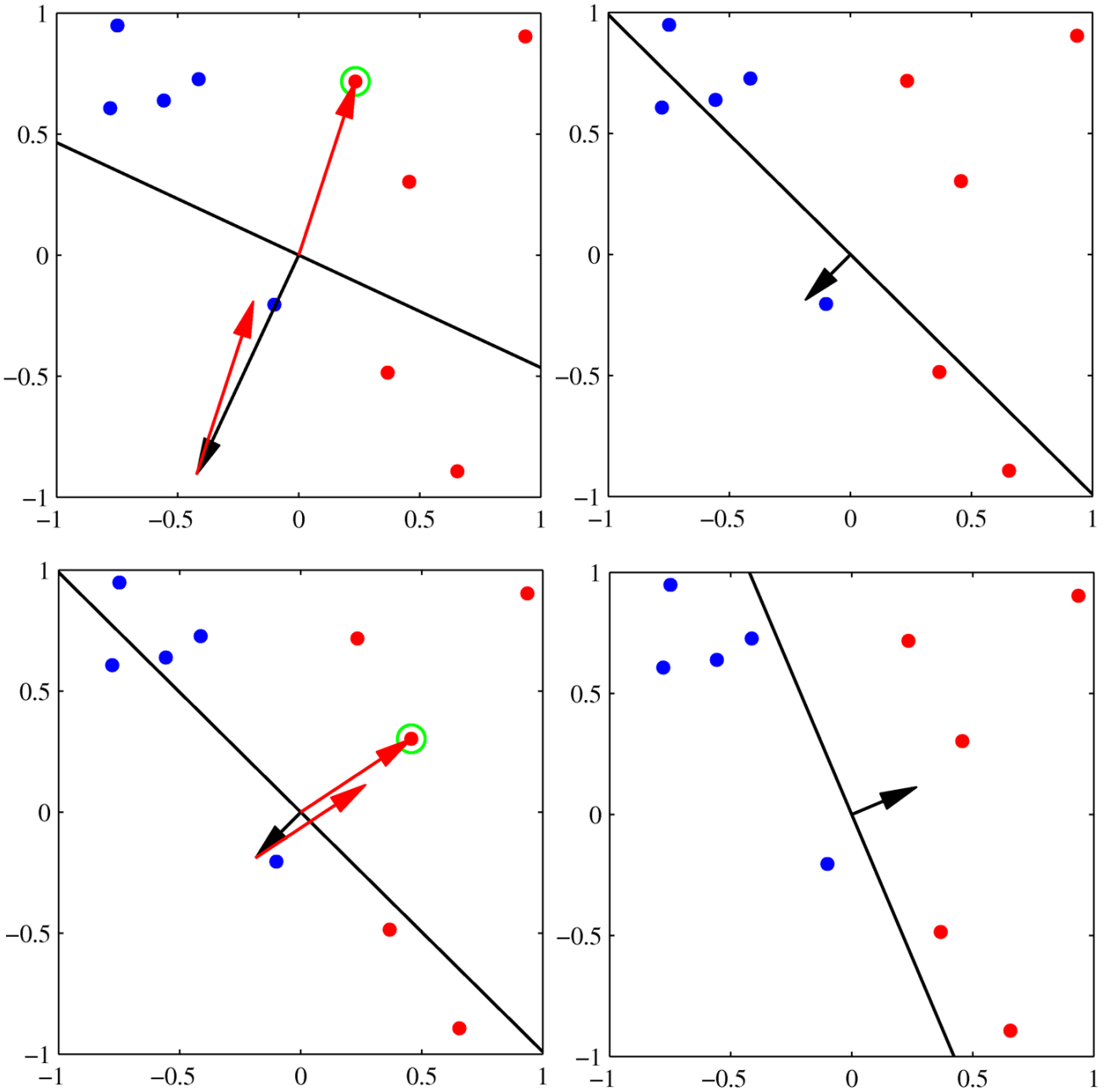
- Lineares Modell:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$
- Ziel: Für alle Beispiele:
  - $y_i f(\mathbf{x}_i) = y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i > 0$
- Perzeptron-Optimierungskriterium:
  - $$J_P(\mathbf{w}) = \sum_{(\mathbf{x}_i, y_i) \in L} \min\left\{ y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i,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



Perzeptron

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




1
0.5
0
−0.5
−1
−1
−0.5
0
0.5
1

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, iteriere über alle Beispiele
  - Wenn $y_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i \leq 0$ dann $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i$

Perzeptron-Algorithmus

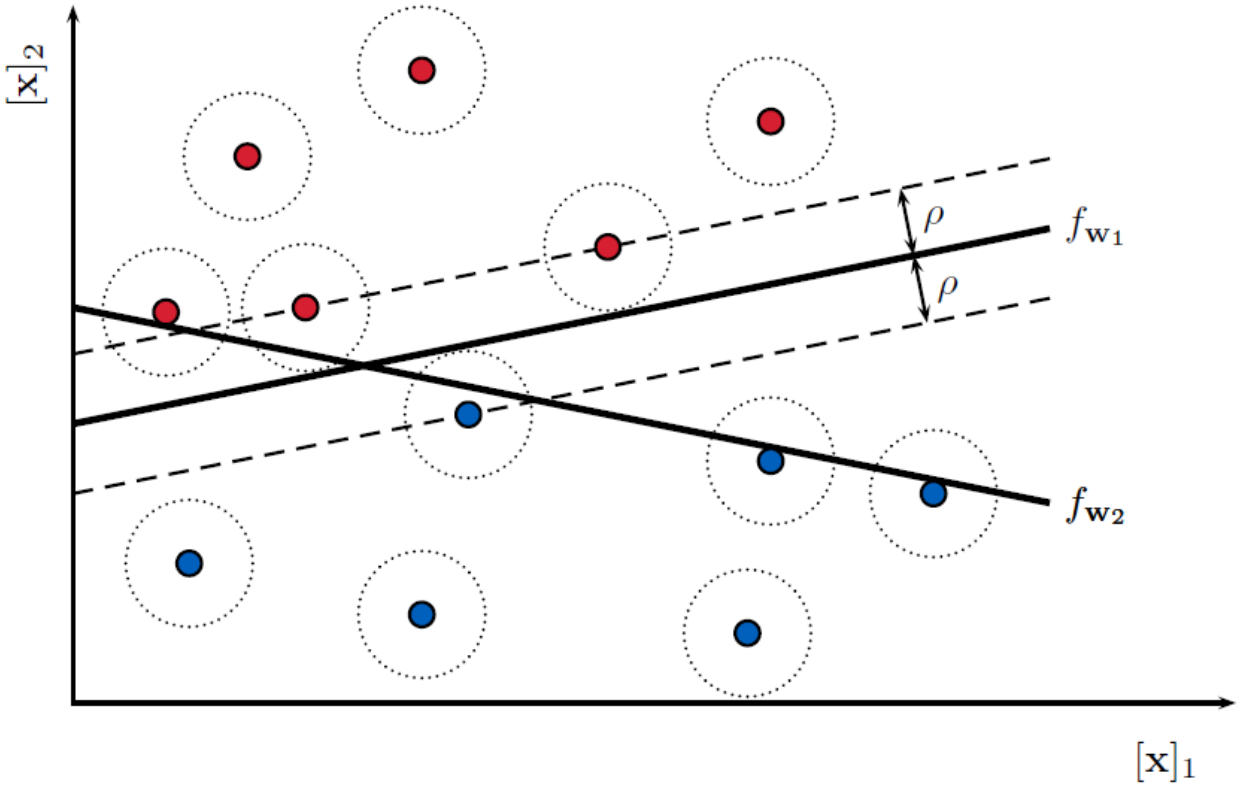
- Perzeptron findet immer eine Trennebene, wenn eine existiert (Optimierungskriterium ist konvex).
- Existiert immer eine Trennebene?
- Wenn keine Trennebene existiert, dann konvergiert der Perzeptron-Algorithmus auch nicht.

Probleme

1. Wenn linear-separierbar, dann mehre Modelle: Welches ist das beste?
   - Lösung: Maximal-Margin-Ansatz

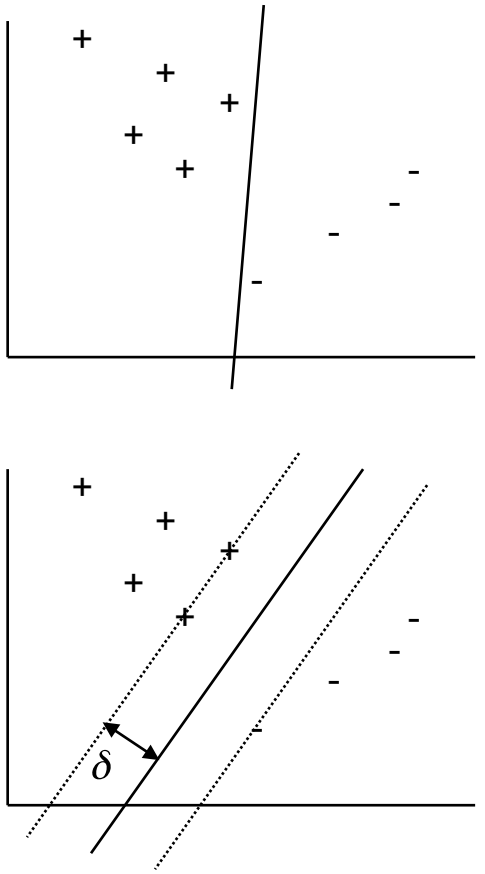
Probleme

1. Wenn linear-separierbar, dann mehre Modelle: Welches ist das beste?
   - Lösung: Maximal-Margin-Ansatz



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.



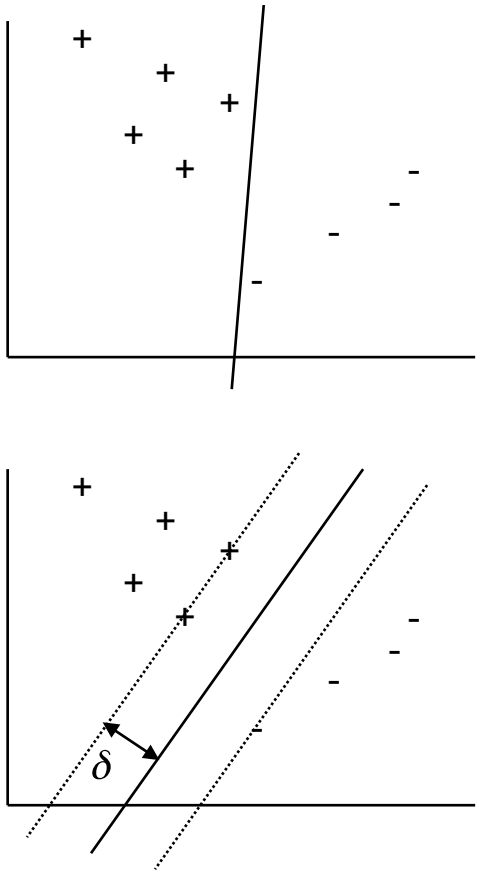
Margin-Perzeptron-Algorithmus

- 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, iteriere über alle Beispiele
  - Wenn $y_i\left(\frac{\mathbf{w}^{\mathrm{T}}}{|\mathbf{w}|}\mathbf{x}_i\right) \leq \delta$ dann $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i$

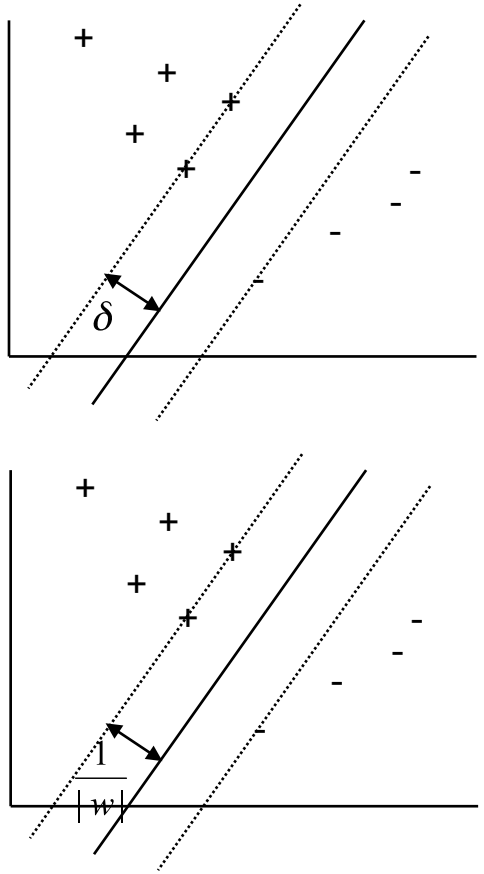
Margin-Maximierung

- Perzeptron: für alle Beispiele muss gelten $y_i \mathbf{w}^T \mathbf{x}_i > 0$
- Margin-Perzeptron: $y_i \left(\frac{\mathbf{w}^T}{|\mathbf{w}|} \mathbf{x}_i \right) > \delta$
  - 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.



Margin-Maximierung

- Margin-Maximierung: $y_i\left(\frac{\mathbf{w}^T}{|\mathbf{w}|}\mathbf{x}_i\right) \geq \delta$
  - Finde Ebene, die alle Beispiele mindestens δ von Ebene entfernt.
  - Für den größtmöglichen Wert δ.
- Maximiere δ unter der Nebenbedingungen: für alle Beispiele $(\mathbf{x}_i, y_i)$ gelte $y_i\left(\frac{\mathbf{w}^T}{|\mathbf{w}|}\mathbf{x}_i\right) \geq \delta$
- = Minimiere $|\mathbf{w}|$ unter der Nebenbedingung:
  - für alle Beispiele $(\mathbf{x}_i, y_i)$: $y_i \mathbf{w}^T \mathbf{x}_i \geq 1$



Probleme

1. Wenn linear-separierbar, dann mehre Modelle: Welches ist das beste?
   - Lösung: Maximal-Margin-Ansatz

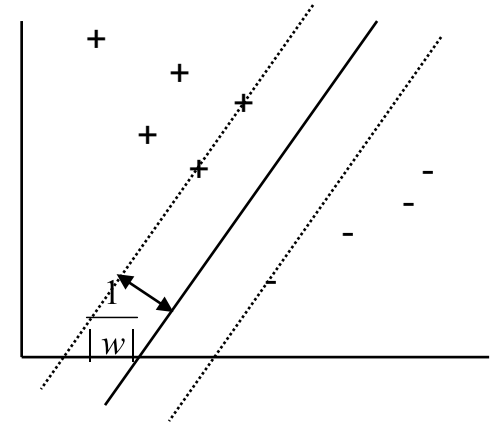
2. Daten verrauscht / falsch gelabelte Daten
   - Lösung: Slack zulassen

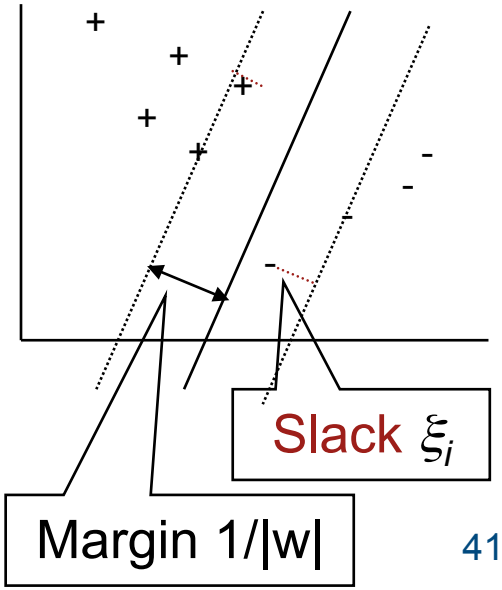
Soft-Margin-Maximierung

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

    $$y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i$$

  - Alle $\xi_i \geq 0$.
- Soft-Margin-Ebene existiert immer, Hard-Margin-Ebene nicht!



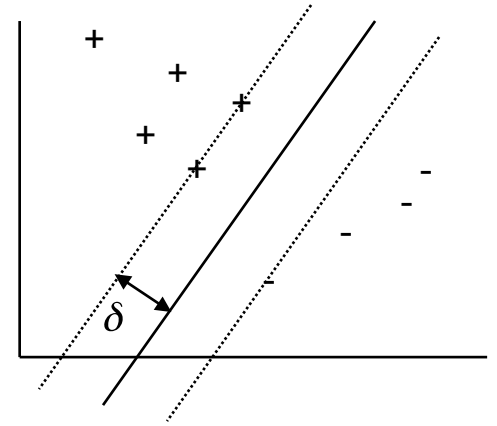


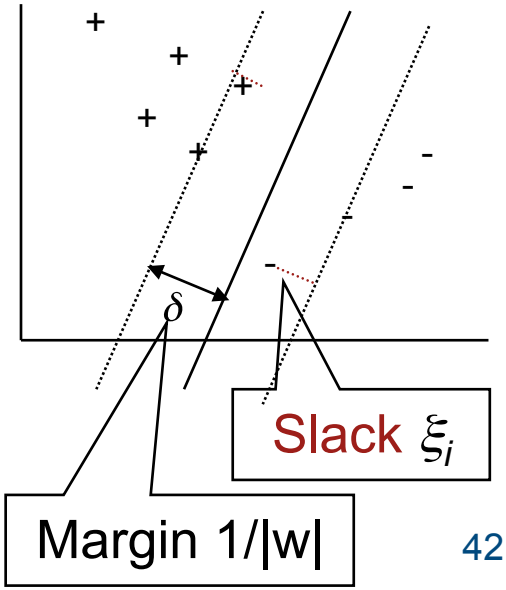
Soft-Margin-Maximierung

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

    $$y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i \geq 1 - \xi_i$$

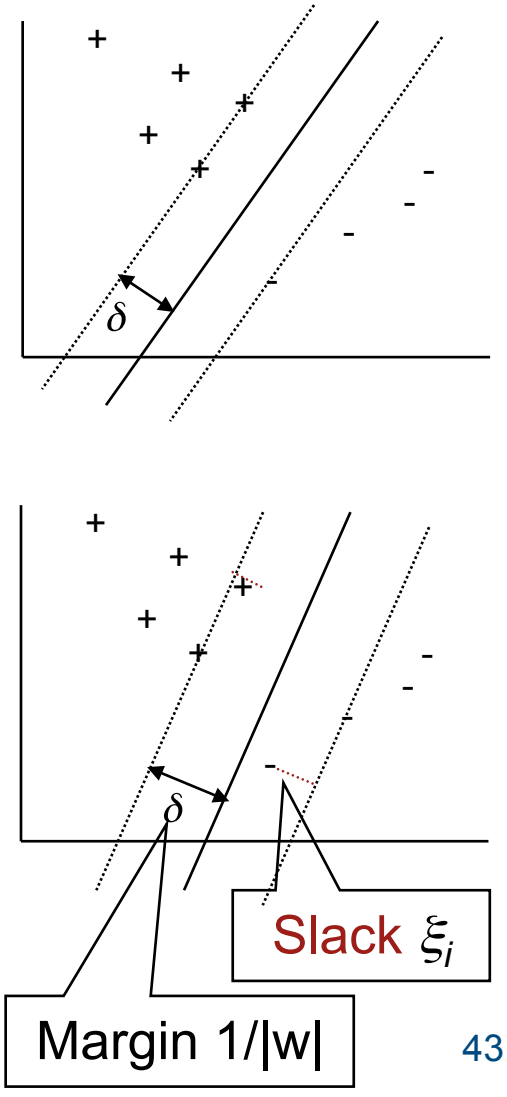
  - Alle $\xi_i \geq 0$.
- Einsetzen von ξ_i in Optimierungskriterium ergibt
  - Minimiere: $|\mathbf{w}| + C\sum_i \max\{0, 1 - y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i\}$





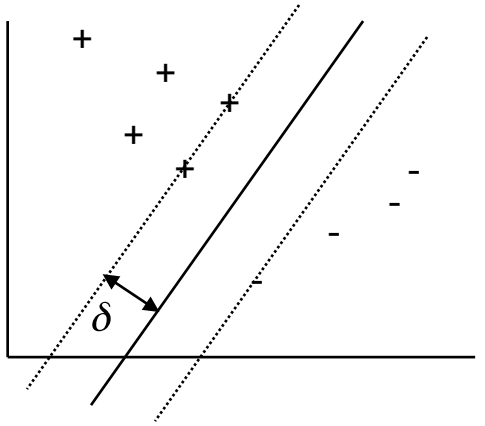
Soft-Margin-Maximierung

- Soft-Margin-Maximierung:
  - Minimiere: $|\mathbf{w}| + C \sum_i \max\{0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\}$
- Minimierung mit Gradientenverfahren.
- Annäherung durch differenzierbare Variante (Huber- statt Hinge-Loss), dann Newton-Verfahren.
- Kriterium ist konvex, es gibt genau ein Minimum.
- Primale Support Vector Machine.
- $O(n^2)$, unter 100.000 Beispiele, unter 100.000 Attribute.



Soft-Margin-Maximierung

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



Enthält Summe über alle Beispiele

Probleme

1. Wenn linear-separierbar, dann mehre Modelle: Welches ist das beste?
   - Lösung: Maximal-Margin-Ansatz
2. Daten verrauscht / falsch gelabelte Daten
   - Lösung: Slack zulassen
3. Keine Fehlerwahrscheinlichkeit der Klassifikation
   - Lösung: Logistic Regression

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?

- Logistische Regression: Vorhersage der Klassenwahrscheinlichkeit.

Logistische Regression

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

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:

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

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

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

Logistische Regression

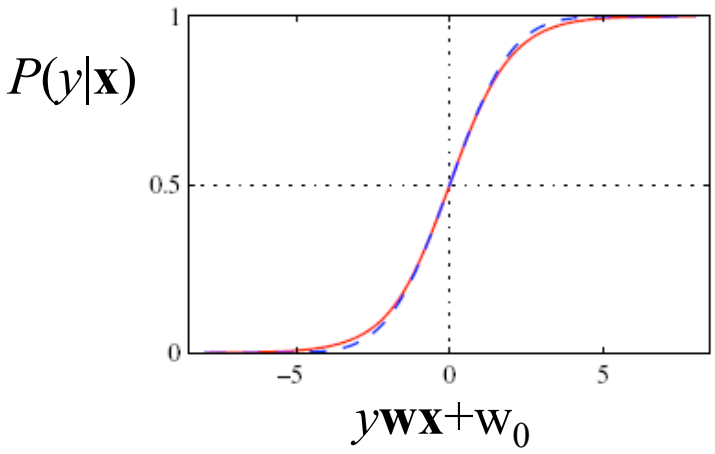
- Likelihood jeder Klasse normalverteilt, gemeinsame Kovarianzmatrix für beide Klassen.
  - $$p(\mathbf{x} \mid y) = \frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu)\right]$$
- Logg-odds ratio:

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

Logistische Regression

- Wenn zwei Klassen jeweils normalverteilte Likelihood mit derselben Kovarianzmatrix haben, dann nimmt $P(y|\mathbf{x})$ diese Form an:

$$P(y \mid \mathbf{x}) = \frac{1}{1+\exp(-y\mathbf{w}\mathbf{x} + w_0)} = \sigma(\underbrace{\underbrace{y\mathbf{w}\mathbf{x} + w_0}_{\text{linearer Klassifikator}}}_{\text{logistische Funktion}})$$



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: Wir finden wir tatsächlich Parameter $\mathbf{w}$ und w_0?

Logistische Regression

- Prior über Parameter:
  - Normalverteilung, w~$N(0,\nu I)$
  - Posterior:
  - $$P(\mathbf{w} \mid L) = \prod_{i=1}^{N} p(y_i \mid \mathbf{x}_i, \mathbf{w}) p(\mathbf{w})$$
    $$= \prod_{i=1}^{N} \sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}_i)^{[[y_i=+1]]} (1 - \sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}_i))^{[[y_i=-1]]} p(\mathbf{w})$$

- Verlustfunktion:

$$E(\mathbf{w}, L) = -\log p(\mathbf{w} \mid L)$$

$$= -\sum_{i=1}^{N} [[y_i = +1]] \log(\sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}_i)) + [[y_i = -1]] \log(1 - \sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}_i)) + \log\left(\frac{1}{\sqrt{(2\pi)^d |\nu I|}}\right) - \frac{\mathbf{w}^{\mathrm{T}} I \mathbf{w}}{2\nu^2}$$

Logistische Regression

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