Textkategorisierung

(Text Classification)

Kategorisierung

- Gegeben:
  - Eine Beschreibung einer Instanz, $x \in X$, wobei X der Raum der Instanzen ist.
  - Eine festgelegte Menge von Kategorien/Klassen: $C=\{c_1, c_2,\ldots c_n\}$
- Bestimme:
  - Die Kategorie $c(x) \in C$ von x.
    Die Funktion c: $X \rightarrow C$ wird dann Kategorisierungsfunktion genannt.

Lernen einer Kategorisierung

- Ein Trainingsbeispiel ist ein Fall $x \in X$, gepaart mit seiner korrekten Kategorie $c(x)$:
  $<x, c(x)>$ für eine unbekannte Kategorisierungs-funktion c.
- Gegeben ist eine Menge D von Trainingsbeispielen.
- Finde eine hypothetische Kategorisierungs-funktion $h(x)$, so dass gilt:

$$\forall < x, c(x) > \in D : h(x) = c(x)$$

Konsistenz

Beispiel für ein Kategorie-Lernproblem

- Instanzraum: <Größe, Farbe, Form>
  - Größe ∈ {schmal, mittel, groß}
  - Farbe ∈ {rot, blau, grün}
  - Form ∈ {quadratisch, kreisförmig, dreieckig}
- *C* = {positiv, negativ}
- *D*:

Beispiel	Größe	Farbe	Form	Kategorie
1	schmal	rot	kreisförmig	Positiv
2	groß	rot	kreisförmig	Positiv
3	schmal	rot	dreieckig	Negativ
4	groß	blau	kreisförmig	negativ

Allgemeine Aspekte beim Lernen

- Viele Hypothesen sind gewöhnlich mit den Trainingsdaten konsistent.
- Tendenz (bias)
  - Jedes andere Kriterium (außer der Konsistenz), das verwendet wird, um eine Hypothese auszuwählen.
- Klassifizierungsgenauigkeit (% Fälle sind korrekt klassifiziert).
  - gemessen an unabhängigen Testdaten.
- Trainingszeit (Effizienz von Trainingsalgorithmen).
- Testzeit (Effizienz nachträglicher Klassifizierung).

Verallgemeinerung

- Hypothesen müssen verallgemeinern, um nicht in den Trainingsdaten vorhandene Fälle korrekt zu klassifizieren.
- Einfaches Speichern von Trainings-beispielen ist eine konsistente Hypothese, die aber nicht verallgemeinert.
- *Das Ökonomieprinzip*:
  - Das Finden einer *einfachen* Hypothese hilft, die Verallgemeinerung zu gewährleisten.

Textkategorisierung

- Zuordnung von Dokumenten zu einer festgelegten Menge von Kategorien.
- Anwendungen:
  - Webseiten
    - Empfehlungen
    - Yahoo-artige Klassifizierung
  - Newsgroup Nachrichten
    - Empfehlungen
    - Spamfiltern
  - Zeitungsartikel
    - Personalisierte Zeitung
  - Email Nachrichten
    - Routing
    - Priorisieren
    - in Ordnern ablegen
    - Spamfiltern

Textkategorisierung lernen – Warum?

- Die manuelle Entwicklung von Funktionen zur Textkategorisierung ist schwierig.
- Lernalgorithmen:
  - **Bayesian (naïve)**
  - Neuronales Netz
  - **Relevanz Feedback (Rocchio)**
  - regelbasierend Lerner (Ripper)
  - **Nächster Nachbar (fallbasierend)**
  - Support Vektor Maschinen (SVM)

Nutzung von Relevance Feedback (Rocchio)

- Relevanz Feedback Methoden können zur Textkategorisierung angepasst werden.
- Verwende TF/IDF-gewichtete Vektoren, um Textdokumente darzustellen (normalisiert durch die maximale Termhäufigkeit).
- Berechne für jede Kategorie einen *Prototyp*-Vektor durch Addieren der Vektoren der Trainingsdokumente einer Kategorie.
- Ordne die Textdokumente der Kategorie mit dem nächsten Prototyp-Vektor basierend auf der Kosinusähnlichkeit zu.

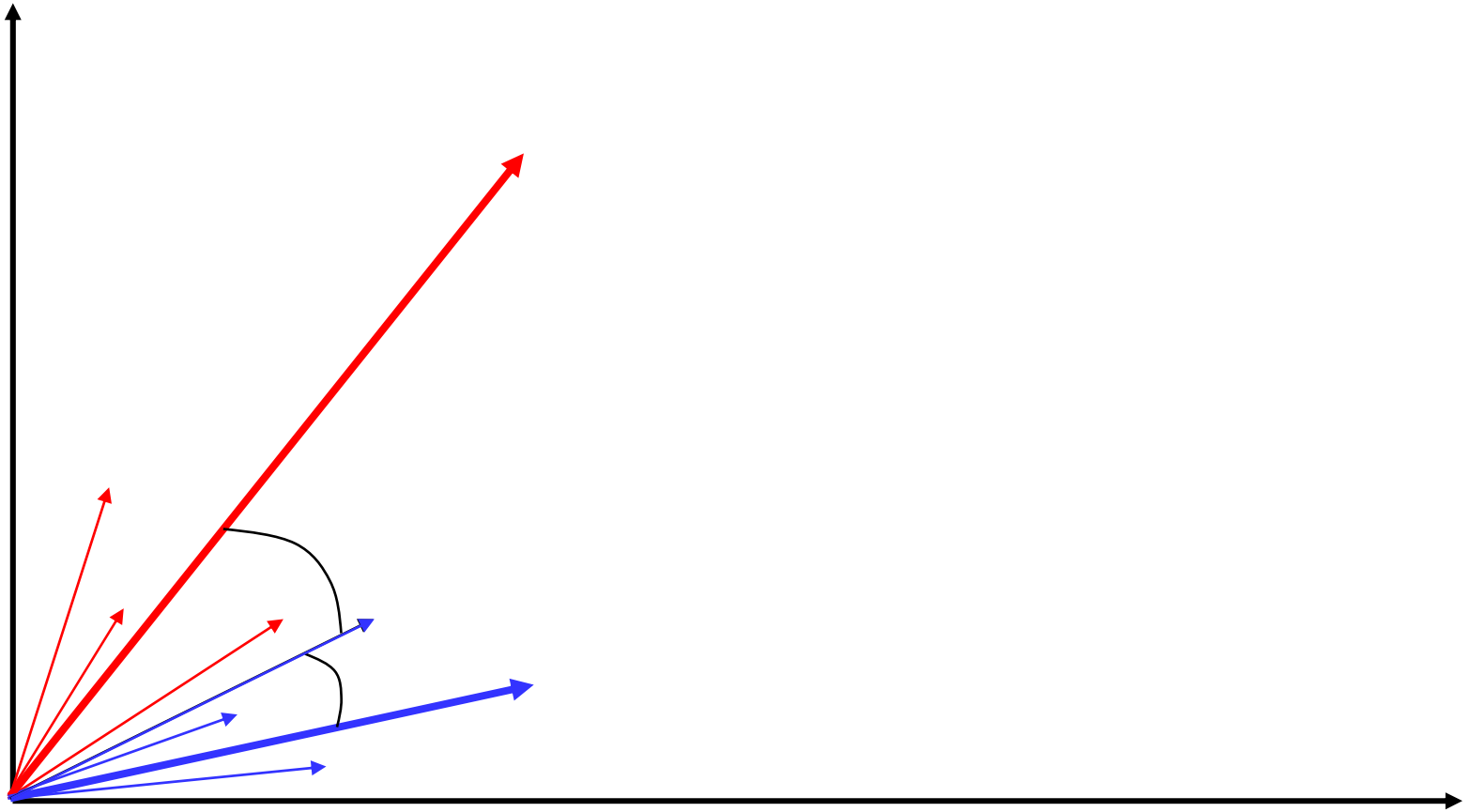
Rocchio Textkategorisierungs-Algorithmus (Training)

Gegeben die Menge der Kategorien $\{c_1, c_2, \ldots c_n\}$.
Für i von 1 bis n sei $\mathbf{p}_i = <0, 0, \ldots, 0>$ (*init. Prototypvektoren*)
Für jedes Trainingsbeispiel $<x, c(x)> \in D$
    Sei **d** der normalisierte Häufigkeits-TF/IDF-Termvektor des Dokumentes x
    Setze i so dass $c_i = c(x)$
    (*addiere alle Dokumentvektoren der Klasse c_i, um $\mathbf{p}_i$ zu erhalten*)
    Setze $\mathbf{p}_i = \mathbf{p}_i + \mathbf{d}$

Rocchio Textkategorisierungs-Algorithmus (Test)

Gegeben ist das Testdokument *x* aus der Testmenge.
Sie **d** der TF/IDF-gewichtete Termvektor von *x*
Sei $m = -2$ (*init. maximum cosSim*)
Für $i = 1$ bis n:
    (berechne Ähnlichkeit *mit Prototypvektor*)
    $s = \text{cosSim}(\mathbf{d}, \mathbf{p}_i)$
    If $(s > m)$
        $m := s$
        $r := c_i$ (*aktualisiere den ähnlichsten Klassen-Prototypen*)
return Klasse *r*

Illustration der Rocchio-Textkategorisierung



Rocchio: Eigenschaften

- Garantiert keine konsistente Hypothese.
- Bildet eine einfache Verallgemeinerung der Beispiele einer jeden Klasse (einen *Prototyp*).
- Der Prototypvektor braucht nicht gemittelt oder anderweitig in der Länge normalisiert werden, da die Kosinusähnlichkeit gegenüber der Vektorlänge unempfindlich ist.
- Die Klassifizierung basiert auf der Ähnlichkeit zu den Klassen-Prototypen.

Rocchio: Zeitkomplexität

- Hinweis: Die Zeit, um zwei dünn besetzte Vektoren zu addieren, ist proportional zur kleinsten Anzahl der nicht-Null Einträge beider Vektoren.

- Trainingszeit: $O(|D|(L_d + |V_d|)) = O(|D| L_d)$
  wobei L_d die Durchschnittslänge eines Dokuments in D ist und V_d die durchschnittliche Vokabulargröße eines Dokuments in D ist.

- Testzeit: $O(L_t + |C||V_t|)$
  wobei L_t die Durchschnittslänge eines Testdokuments ist und $|V_t|$ die durchschnittliche Vokabulargröße eines Testdokuments.
  - Man nehme an, dass die Längen der $\mathbf{p}_i$ Vektoren während der Trainingsphase berechnet und gespeichert wird. Dies ermöglicht die Berechnung von cosSim($\mathbf{d}$, $\mathbf{p}_i$) mit einem Aufwand, der proportional zur Anzahl der nicht-Null Einträge in $\mathbf{d}$ (d.h. $|V_t|$) ist.

Nächste-Nachbar-Lernalgorithmus

- Das Lernen besteht nur im Speichern der Trainings-beispiele aus *D* in einer geeigneten Repräsentation.
- Testfall *x*:
  - Berechne die Ähnlichkeit zwischen *x* und allen Beispielen in *D*.
  - Ordne *x* die Kategorie des ähnlichsten Beispiels in *D* zu.
- Berechnet nicht explizit eine Verallgemeinerung oder einen Prototypen einer Kategorie.
- Man nennt NN-Verfahren auch:
  - fallbasiert
  - speicherbasiert
  - faules Lernen

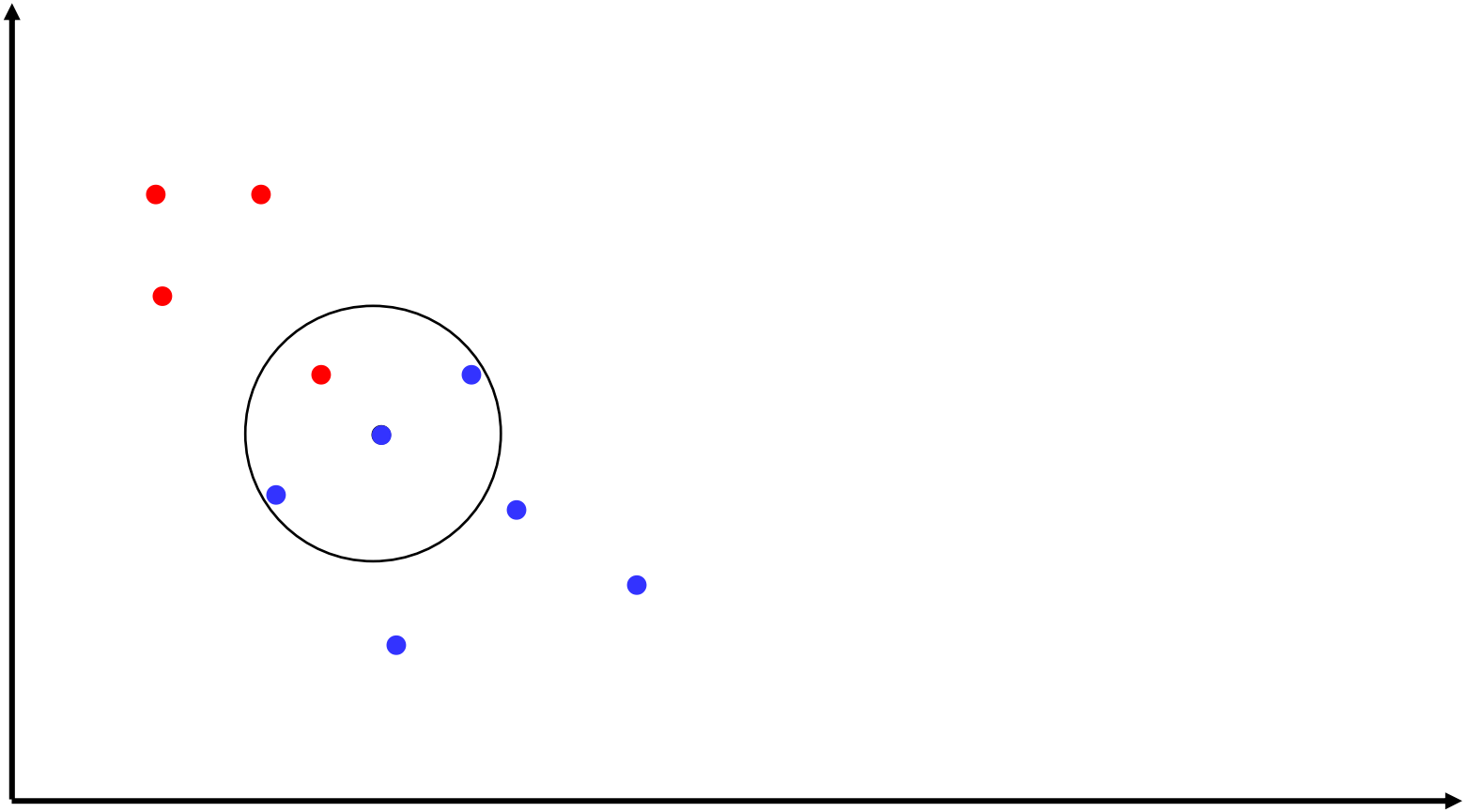
K-Nächster-Nachbar

- Nur das nächste Beispiel zu verwenden, um Kategorisierung zu bestimmen, ist häufig die Ursache von Fehlern, da:
  - einzelne Beispiele atypisch sein können.
  - Rauschen (d.h. Fehler) beim Kategorielabel eines einzelnen Trainingsbeispieles vorkommen kann.
- Eine robustere Alternative ist, die *k* ähnlichsten Beispiele zu finden und die Kategorie der Mehrheit dieser *k* Beispiele zurückzuliefern.
- Der Wert von *k* ist typischerweise ungerade, um ein Unentschieden zu vermeiden, 3 und 5 werden am häufigsten verwendet.

Ähnlichkeitsmaße

- Die nächste Nachbarmethode hängt von einem Ähnlichkeits- (oder Abstands-)maß ab.
- Das einfachste stetige *m*-dimensionale Abstandsmaß ist die *euklidische Distanz*.
- Das einfachste *m*-dimensionale binärenAbstandsmaß ist der *Hamming-Abstand* (Anzahl der Merkmalswerte, die sich unterscheiden).
- Für Text ist die Kosinusähnlichkeit von TF-IDF-gewichteten Vektoren typischerweise am effektivsten.

Beispiel für 3-nächste-Nachbarn (Euklidischer Abstand)



K-nächster-Nachbar für Text

Training:

Für jedes Trainingsbeispiel $\langle x, c(x) \rangle \in D$

Berechne den TF-IDF Vektor $\mathbf{d}_x$ für Dokument x

Testfall *y*:

Berechne TF-IDF Vektor $\mathbf{d}_y$ für Dokument y

Für jedes $\langle x, c(x) \rangle \in D$

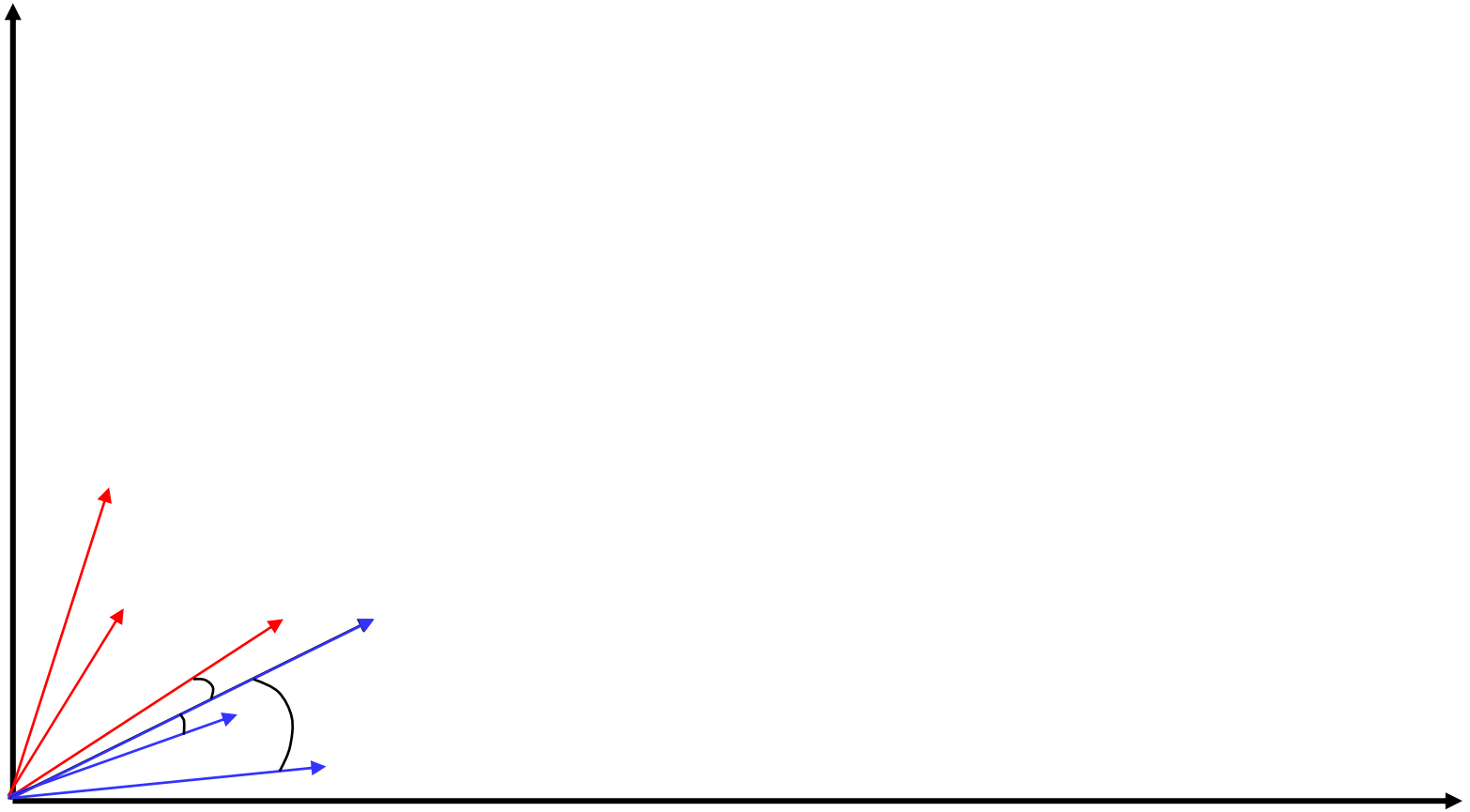
$s_x = \text{cosSim}(\mathbf{d}_y, \mathbf{d}_x)$

Sortiere die Beispiele $x \in D$ absteigend nach s_x

Fülle Menge N mit den ersten k Beispielen aus D (d.h. den *ähnlichsten Nachbarn*)

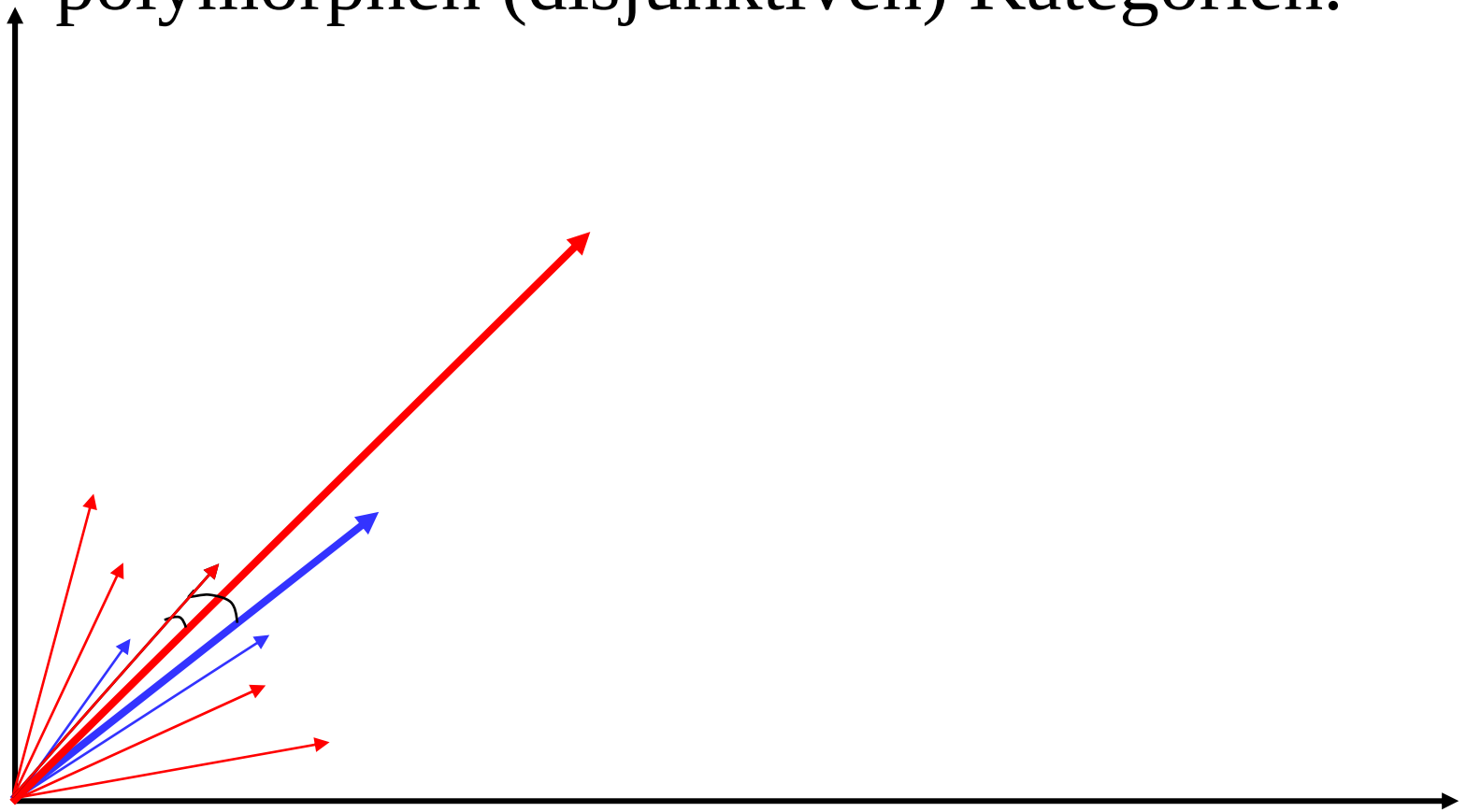
Gib als Klasse die Klasse der Mehrheit der Beispiele in N zurück.

Beispiel: 3-nächste-Nachbarn für Text



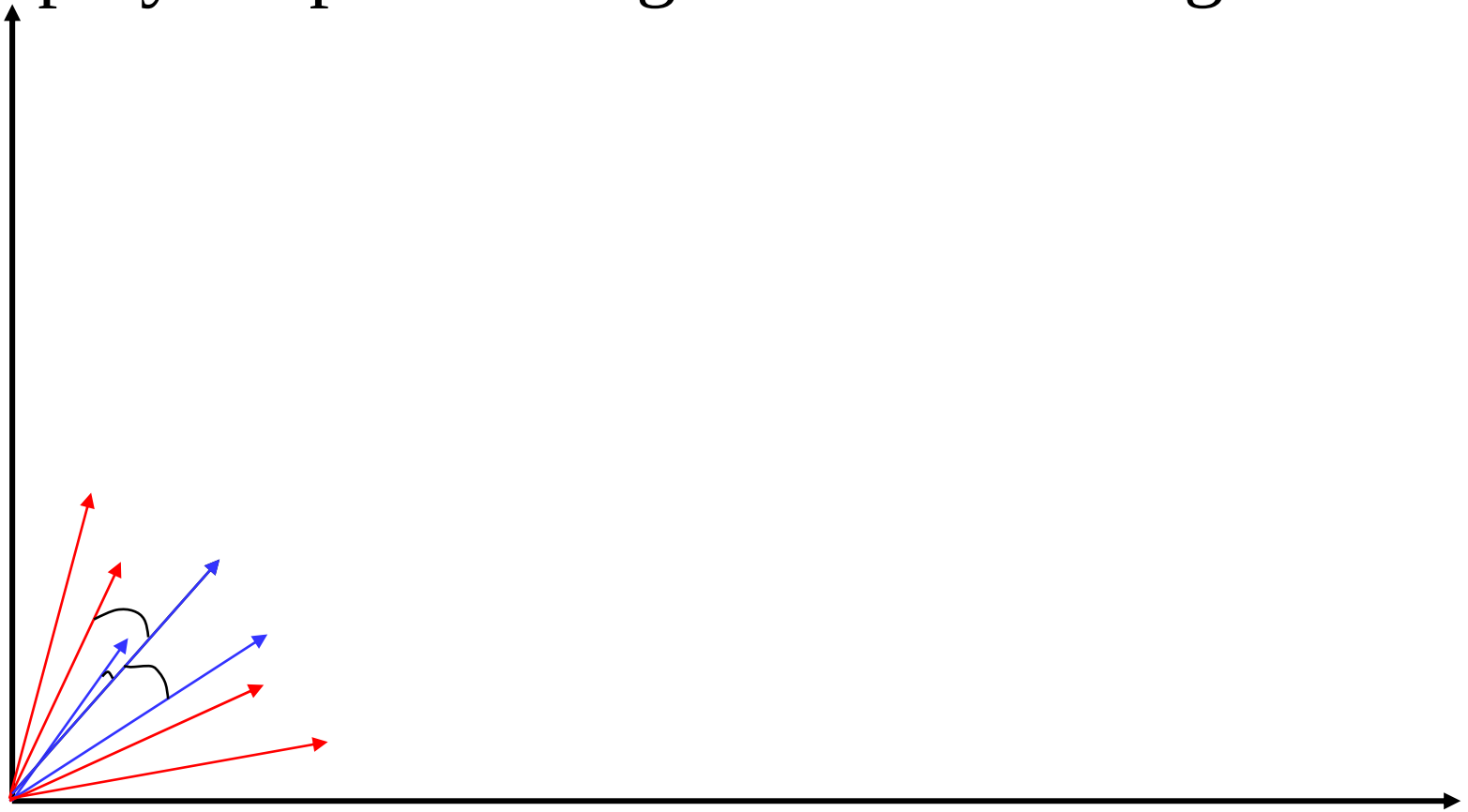
Rocchio-Anomalie

- Prototyp-Modelle haben Probleme mit polymorphen (disjunktiven) Kategorien.



3-nächster-Nachbar: Vergleich

- Das nächste-Nachbar-Verfahren kann mit polymorphen Kategorien besser umgehen.



Nächste-Nachbarn: Zeit-Komplexität

- Trainingszeit: O($|D|\ L_d$) um TF-IDF Vektoren zu berechnen.
- Testzeit: O($L_t + |D||V_t|$) um mit allen Trainingsvektoren zu vergleichen.
  - Man nehme an, dass die Längen der $\mathbf{d}_x$ Vektoren während der Trainingsphase berechnet und gespeichert wird. Dies ermöglicht die Berechnung von cosSim($\mathbf{d}$, $\mathbf{d}_x$) mit einem Aufwand der proportional zur Anzahl der nicht-Null Einträge in $\mathbf{d}$ (d.h. $|V_t|$) ist.
- Die Testzeit kann für große Trainings-mengen sehr groß werden.

Nächste-Nachbarn mit invertiertem Index

- Die Bestimmung von k nächsten Nachbarn ist das gleiche wie die Bestimmung der k besten Abfragen, wobei das Testdokument als Anfrage an eine Datenbank von Trainingsdokumenten verwendet wird.
- Verwende standard-KSM-invertierte-Indexmethoden, um die k nächsten Nachbarn zu finden.
- Testzeit: $O(B|V_t|)$
  wobei B die durchschnittliche Anzahl der Trainingsdokumente ist, in welchem ein Test-Dokumentwort vorkommt.
- Folglich ist die Komplexität der gesamten Klassifizierung $O(L_t + B|V_t|)$
  - Typischerweise gilt $B << |D|$

Bayes-Methode

- Lern- und Klassifizierungsmethoden basierend auf der Wahrscheinlichkeitstheorie.
- Bayes-Theorem spielt eine zentrale Rolle bei dem probabilistischen Lernen und Klassifizieren.
- Ausgehend von der (prior) Wahrscheinlichkeit der Kategorien (wenn keine Information über ein Objekt vorhanden ist)...
- ... berechnet die Methode eine Kategorisierung basierend auf der (posterior) Wahrscheinlichkeitsverteilung, wenn man annimmt, daß eine Beschreibung eines Objektes gegeben ist.
- Zentrale Annahme ist die Unabhängigkeit der Merkmale für eine gegebene Kategorie.

Naïve Bayes für Text

- Texte werden als Menge von Worten (Bag of Words) repräsentiert.
- Für das Modell wird angenommen, dass der „Bag of Words“ eines Dokumentes aus einer gegebenen Kategorie durch wiederholtes Ziehen mit Zurücklegen aus dem Vokabular $V = \{w_1, w_2, \ldots w_m\}$ erzeugt wurde.
- Die Wahscheinlichkeitsverteilung für V ist $P(w_j \mid c_i)$.
- Man versucht nun, dieses generative Modell beim Lernen aus den Daten zu schätzen.

Naïve Bayes a posteriori Wahrscheinlichkeiten

- Klassifizierungsergebnisse von naïve Bayes (die Klasse mit der maximalen a posteriori Wahrscheinlichkeit) sind üblicherweise ziemlich akkurat.
- Jedoch trifft dies, aufgrund des Nichtzutreffens der bedingten Unabhängigkeitsannahme nicht auf die tatsächliche geschätzten a posteriori Wahrscheinlichkeiten zu.

Evaluierung von Kategorisierungsmodellen

- In der Praxis werden die Modelle auf Daten angewandt, bei denen die Kategorien nicht bekannt sind.
- Für Testzwecke werden Daten benötigt, bei denen diese Zuordnung gegeben ist.
- Die Evaluierung muss auf Testdaten erfolgen, die unabhängig von den Trainingsdaten (üblicherweise eine unabhängige Menge von Beispielen) sind.
- *Klassifizierungsgenauigkeit*: c/n wobei n die Gesamtzahl der Testfälle ist und c die Anzahl von Testfällen, die korrekt vom System klassifiziert wurden.
- Die Ergebnisse können wegen Stichprobenfehlern aufgrund verschiedener Trainings- und Testmengen variieren.
- Mittle die Ergebnisse über eine Vielzahl von Trainings- und Testläufen (Teile der Gesamtdaten), um die Klassifizierungsgenauigkeit möglichst gut zu bestimmen.

N-fache Kreuz-Validierung

- Idealerweise sind Test- und Trainingsmengen bei jedem Versuch unabhängig voneinander.
  - Aber dies würde zu viele gekennzeichnete Daten erfordern.
- Alternative:
- Teile Daten in N gleichgroße unabhängige Teile.
- Führe N Versuche durch, wobei jedes Mal ein anderer Teil der Daten zum Testen verwendet wird und das Training mit den verbleibenden N–1 Segmenten erfolgt.
- Auf diese Weise sind zumindest die Testmengen unabhängig.
- Das Gesamtergebnis ergibt sich als die durchschnittliche Klassifizierungsgenauigkeit über die N Versuche.
- Typischerweise wählt man $N = 10$.

Lernkurven

- In der Praxis sind gekennzeichnete Daten üblicherweise selten und teuer.
- Man würde gern wissen, wie die Leistung der Methoden mit der Anzahl der Trainingsbeispiele variiert.
- *Lernkurven* tragen die Klassifizierungsgenauigkeit unabhängiger Testdaten (*y*-Achse) über die Anzahl der Trainingsbeispiele (*x*-Achse) auf.

N-fache Lernkurven

- Lernkurven sind nur dann aussagekräftig, wenn diese über eine Vielzahl von Versuchen gemittelt werden.
- Verwende *N*-fache Kreuzvalidierung, um *N* volle Trainings- und Testmengen zu erzeugen.
- Trainiere bei jedem Versuch mit zunehmenden Teilen der gesamten Trainingsmenge, um durch das Messen der Genauigkeit auf den Testdaten jeden Punkt auf der gewünschten Lernkurve zu ermitteln.

Beispiel einer Lernkurve (Yahoo Science Data)