```
Entscheidungstheorie
====================

1. Wkeit
--------

* bedingte

* totale

* Satz von Bayes

* a priori

* a posteriori


2. Bayesche Entscheidungsregel
------------------------------

* stetige Featurewerte
  - bedingte Wkeitdichte
  - Dichte


* Featurevektor


3. Fehlerrate
-------------

* Formel

* Bayessche


4. Risiko-Minimierung
---------------------

* Wertung von Features

* Entsch.Regel zur RM


5. DiskriminantenFunktion
-------------------------

* Vorteile
  - Vereinfachung Entscheidungsregel
  - Konzept für diskr. Ansatz


* Template Matching / Rocchio Klassifiz.

* lineare


6. mehrdim/multivariante NormVert.
----------------------------------

* Kovarianzmatrix
  - Formel
  - Aus Training: empirische Kov.

* Wdichte

* Varianz aller Features gleich: lin. DF ist Template-Matching.
  Idee: ln() nutzen und Konst. bzw von i unabh. Summanden verwerfen


Wkeit-Dichte
============

1. nicht parametrische Dichteschätzung
--------------------------------------
```



```
* Suche Schätzer für p(x) ohne Annahme zur Verteilung.
  Also: kein Schätzer für Parameter von p(x)

* 1. Idee
  - Histogramm
  - Anzahl Messungen in Intervall
  - Bei Binominalverteilung: erw.treuer Schätzer für p in Intervall
  - Probleme
    - unstetig
    - d Dimensionen: N^d Intervalle
    - viele Zellen mögl. leer
    - Lösungen
      - unstetig? Kernel-Methoden
      - Zu viele oder leere Zellen? variable Zellengröße


* Kernel-Methoden
  - keine willkürlichen Intervallgenzen
  - Wkeit eines Messwerts wird aus Distanz innerhalb einer Fensterbreite
    rund um alle Messwerte ermittelt
  - gleitendes Histogramm möglich:
    p an jedem Punkt x statt nur in einem Intervall
  - mit n Messw. und K - positiv und auf 1 norm. - ist das auch
    für Dichte pDach(x)
  - pDach(x) = 1/(n*h) SumAlle.n( K((x-xi)/h) )
  - d-Dim: kugelsym. Kern mit (2pi)^(-d/2) * e^(-<x,x>/2)


* Fensterbreite h
  - Kernel: globales h
    - MeanSquareError = Bias²(pDach(x)) + Var(pDach(x))
    - Güte Schätzung über alle durch Integration: MISE(pDach)
    - Tylorentw. von MISE mit pDach(x) in h:
      AMISE(h) Ableitung =0 für Minimum in h
      - Bei NV: h = 0.79 * IQR * n^(-1/5)
      - Bei NV, weil h sonst zu sehr geglättet: Silvermans Rule of Thumb
      - h_srot = 0.9 * min{sigmaDach, IQR/1.35} * n^(-1/5)
      - Plugin Methode: numerische Lösung von Minimum h aus
        AMISE'(h) = 0 mit h_srot als Startpunkt

  - kNN-Methoden
    - statt variabler Anz. Messwerte in festem h:
      feste Anzahl der Messwerte in var. h
      - lokales h: Vermeidung, dass h bei hoher Wkeit zu klein/niedrig
      - statt h jetzt: V(x) = 2* (k-kleinster Abstand von x)

    - Fehlerrate
      - Erwartungswert der Fehlerrate asymptotisch Abschätzbar
      - e B = Bayessche Fehlerrate
      - Cover & Hart: blödster Fall k=1 aber n -> inf liefert:
        e B < E(e) < 2e B

    - pDach(x) = k/(n*2*l x - xj l) leider keine Wkeit-Dichte!
      - Integral +- inf ist nicht 1
      - mit k-1 statt k lässt sich aber mit steigendem n zeigen, dass
        fDach_n(x) sich der WDichte in x nähert
      - k(n) ist Nachbarzahl
      - fDach n(x) = (k(n) -1)/(n*V(x))
      - Nicht so tragisch! Wir wollen x ja einer Klasse aus
        Trainingsdaten zuordnen
        - für a posteriori Wkeit P(wi I x) reichts
        - kNN-Entsch.Regel: x wird der Class zugeteilt, die unter
          den k-nächsten Nachb am häufigsten ist!
        - gleich häufig: 1) zufällig, 2) am nächsten an x, 3) im Mittel
          am Nächsten

    - Parameterwahl
      - k nie größer als niedriegste "Klassenpopulation"
      - Enas & Choi: k ca n^(2/8)
      - für gutes k: Gütekriterien nutzen
      - Wahl der METRIK: Was heißt "Nächster"? Ggf. Bewertung von
        Features untersch.

    - Edit Trainingsdaten
      - Ziele
```

```
153           - Veringern Fehlerrate
154           - Beschl. Klassifizierung
155
156         - Methoden
157           - Editing
158             - einzelne Ausreißer entfernen
159             - Ausreißer werden als fehlklassifiziert angesehen, sobald
160               sie nicht mit dem Rest via kNN-Regel klassifizierbar sind
161
162           - Condensing
163             - entfernen von überflüssigen Samples
164             - Entferung darf bei restlichen Samples die "Decision Bounderies"
165               nicht ändern
166             - Algorithm oft von Reihenfolge der Samples abh.
167             - Ab 3 Class: Minimierung in konsistente Teilmengen NP-vollst.
168
169           - Prototype-Generator
170
171
172   2. Parameterschätzung (Annahme über WVerteilung)
173   -------------------------------------------------
174
175   * Allgemein
176     - Annahme, wie Features verteilt sind
177     - Beispiel: schätze Mittelwert und Streuung
178
179
180   * Maximum Likelihood Schätzer
181     - wie ist Parameter Theta zu wählen, so dass aus beob. Werte
182       max. Wkeit haben
183     - Da jede Messung unabh.: Maximiere Produkt aus Wkeitverteilung
184       je beobachteten Wert -> theta Dach
185     - Gradient(Nabla ∇) von Likelihood-Fkn = 0
186     - Vorsicht: Wendepkt? Rand von Theta?
187     - mit log-Likelihood-Fkn ergibt 01,2 bei NV: Mittelwert und emp. Var.
188     - Mittelwert-Vektor und (nicht erw.treue) Kovar.Matr
189
190
191   * erw. Treue Schätzer
192     - Erw.Wert des Schätzers sollte Parameter liefern
193     - BIAS: Abweichung Erwartungswert des empirischen Schätzers von
194       theor. Schätzers
195     - SigmaDach = (1/(n-1)) * SummeAlle.k von
196        (vek.x k - vek.muDach)*(vek.x k - vek.muDach)^t
197
198
199   * Bayes Classifier NV
200     - Nutzung emp. Kovarianz und Mittelwert
201     - naivesBayes(): Wkeit x wenn in wi ist Produkt der Wkeit je Feature,
202       wenn Feature in wi (=Feature unabh.)
203     - Listing gausian.bayes():
204       Nutzung der Trainingsdaten für Wkeiten von x k
205       - Aber: jede Klasse ist hier gleich-Wscheinlich
206       - a priori: aus Erfahrung bzw. Anfangs-Wkeiten
207
208     - Alternativ: Kerneldichte-Schätzer für Wkeiten nutzen:
209       NICHT PARAMETRISCH!
210
211
212   3. Quantil
213   ----------
214
215   * IQR: Inter Quartile Range x_0.75 - x_0.25
216
217   * x_0.25 ist der x-Wert, bei dem 25% der Messwerte xi kleiner sind
218
219   * Quartil: 0.75 oberes, 0.5 mittleres (Median), 0.25 unteres
220
221
222   Classifier
223   ==========
224
225   1. diskriminatorischer Ansatz
226   -----------------------------
227
228   * class ist Teilregion des Featurespace
```



```
229
230
231   2. probabilistischer Ansatz
232   ---------------------------
233
234   * class und feature ist P zugeordnet
235
236   * allg. als diskr. Ansatz
237
238
239   3. Features
240   -----------
241
242   * Testdaten
243
244   * Trainingsdaten
245
246   * Extraktion
247
248
249   4. Modell auf Tr.Data: Verallgemeinerung
250   ----------------------------------------
251
252   5. Modell schwer bei Overfitting
253   --------------------------------
254
255   6. Vergleich von 2 (A und B)
256   ----------------------------
257
258   * Naiv
259     - Fehlerrate bei GLEICHEN Testdaten messen
260     - Unterschied der Fehlerate könnte auch an zufälliger Verteilung
261       der Testdaten liegen
262     - je Testdaten könnte Unterschied auch bei 2 identischen
263       Classifieren auftreten
264
265
266   * Nutzung Signifikanzniveau: ähnlich Konfidenzniveau
267
268   * Testdaten unab. zuf. gewählt
269     - A(xi) und A(xj) unabhängig
270     - A(xi) und B(xi) NICHT unabhängig, weil beide mit selben
271       Testsample gemessen werden müssen
272     - McNemar Test
273       - Fehlerratenvergleich
274       - Betrachtet werden nur die Anzahl der Fehlklassifizierungen,
275         wo A richtig und B falsch (n01) liegt bzw. umgekehert (n10)
276       - Wenn beide gleich gut, sollte in etwa gelten:
277         n10 = n01 =(n10 + n01) / 2
278       - Shi² = Maß des Unterschieds der Fehler
279         = ( | n10 - n01 | -1)² / (n10+n01)
280       - Shi(m)² (1-alpha) = 3,8415 mit Signifikanzniveau alpha=0.05,
281         dem (1 - 0.05)-Quantil und m=1 als Anz. der Freiheitsgrade
282       - Beispiel: ist Shi² > 3.8415, dann ist Wkeit für rein statische
283         Abweichung der Fehlerraten < 5%
284
285     - Paired t-Test
286       - Vergleich stetiger Performancemaße (nicht eindeutig Richtig/Falsch)
287       - Differenz D der Performace je Sample i wird betrachtet:
288         Di = A(xi) - B(xi)
289       - Betrachte Teststatistik zu Zuf.Variable D:
290         T=(Dquer * wurz(n) )/s(D)
291         - Dquer = Mittelwert aller i=1..n Di
292         - s²(D) = (n-1)^-1 * Sum (Di -Dquer)²
293         - Gleiche Performance wäre: E(D)=0
294
295       - signifikant untersch. Performance, wenn Betrag von
296         T > t(n-1) (1-alpha/2)
297       - t(n-1) (1-alpha/2) ist das (1-alpha/2) Quantil der t-Verteilung
298         mit n-1 Freiheitsgraden
299         - Mit steigenden Freiheitsgrad (mehr Messwerte) nähert sich
300           Quantil dem der Std.NV
301         - Quantil der Std. NV ist immer kleiner oder gleich
302
303
304   7. Receiver Operation Characteristic (ROC)
```

```
  ----------------------------------------

  * Vergleich von Classifier, die Samples nur 2 Klassen zuordnen

  * Classifier unterscheiden sich von nur 1 Parameter (Z.B. k in kNN
    oder ein Schwellwert für Entscheidung)

  * ROC-Kurve
    - jeder Punkt stammt von einem anderen Parameter
    AUC: Area Under Curve:
      je größer die Fläche, desto besser der Classifier
    - x-Achse: epsilon2 = Rate der Falsch als Klasse w2 Zugeordneten
    - y-Achse: 1-epsilon1 ist Rate, der korr. w1 Zugeordneten


  * bester Parameter?
    - Nutzung von Kosten bei Fehlklassifizierung
    - geringste Kosten (Minimum) sind da, wo Steigung P(w2)/P(w1) ist
    - Kostengrade am weitesten von "teurer" Diagonale entfernt, aber
      tangiert trotzdem noch ROC-Kurve

  Metrische Räume
  ===============

  1. Metrik d auf M bildet M x M -> R ab
  --------------------------------------

  * d(x,y) = 0 <=> x = y

  * Symetrie: Abstand von x nach y ist gleich wie von y nach x

  * Dreiecksungleichung: Ein Umweg könnte länger sein!


  2. falls Menge Vektorraum
  -------------------------

  * V x V -> R ist Skalarprodukt, wenn
    - bilinear: <x+a,y> = <a,y> + <a,y> und <cx,y>=c<x,y>
    - symetrisch: <x,y> = <y,x>
    - positiv: <x,x> >0 (wenn x!=0)


  * Skalarprodukt kann Norm definieren: llxll = wurzel(<x,x>)

  * eine Norm kann eine Metrik definieren: d(x,y) = ll x-y ll

  * allg. R^n Skalarprodukt: <x,y> = vek.x.Transponiert * A * vek.y


  3. Metrik Beispiele
  -------------------

  * triviale Metrik: d(x,y) ist 0 wenn x=y sonst 1

  * Euklidische Metrik: d(x,y) = sqrt(sum(xi - yi)²)

  * Levensthein Distance: d(x,y) = minimale Anz. Edit-OP um aus
    x das Wort y zu machen


  4. Norm
  -------

  * Abbildung eines Elements aus Vektorraum V über R

  * N1: llcxll = lcl * llxll

  * N2: llx+yll <= llxll + llyll

  * N3: llxll = 0 <=> x ist Nullvektor

  * Lp-Norm oder Minkowski-Norm
    - euklid p=2
    - Manhatten p=1
    - Maximumsnorm: p -> inf
```



```
    - p-te Wurzel aus Summe der Beträge aller (xi)^p


  5. Parallelogramm-Gleichung
  ---------------------------

  * ll x+y ll² + ll x-y ll² = 2 llxll² + 2 llyll²

  * dann gilt: Norm stammt von Skalarprodukt

  * <x,y> = 1/4 ( ll x+y ll² - ll x-y ll² )


  6. Nutzung Freiheit des SkProd. (A) für Anpassung Metrik
  --------------------------------------------------------

  * Diagonalmatrix mit >=0 Werten -> Feature Weighting

  * cm oder Meter: Feature Normalization

  * feste Varianz: je Feature auf 1 normiert wi = 1/(si)^p
    - A=Inv(sigma) Inverse der Kovarianzmatrix bei unab. Features
    - Metrix: MAHALAnobis Distanz


  * Normierung auf Mittelwerte = 0 ist nur Verscheibung:
    beeinflusst nicht Abstand

  HMM
  ===

  1. allgemein
  ------------

  * durchläuft Folge von Class

  * Sprache: Folge von Phonemen

  * Schrift: t = Schreibrichtung

  * Gebärde: Folge von Handpos.

  * Wkeit für Folgen berechnen: Markov-Ketten


  2. Begriffe HMM
  ---------------

  * wi = Hidden States

  * vi (i=1..S) Visible States

  * aij = Wkeiten der Hidden States

  * bjk = Wkeit, dass im State wj der Wert vk emittiert wird


  3. HMM Evaluation
  -----------------

  * Anwendung
    - aij,bjk, Startzustand w(0) gegeben
    - Wkeit für generierung der Folge: v(1)...v(T)


  * Naiv: leider O(T * C^T)

  * FORWARD-Algorithmus
    - Mit Startwerten wird Wkeit berechnet für v(1) für C viele
      Hidden States
    - iterativ: für t+1 wird wieder je C aus (Summe der bisherigen
      Wkeiten der Hidden States) * bjk die Wkeit für v(t+1) berechnet.
    - Wegen Nutzung vorheriger Berechungen: O(T*C²)


  4. aij und bjk ? (j,k Laufindex für C,S)
```

```
  ---------------------------------------

  5. HMM Dekodierung
  ------------------

  * Anwendung
    - aij,bjk gegeben sowie eine Folge v(1)..v(T)
    - Was wahrscheinlichste Folge w(1)..w(T) die zu Beobachtung v führt?


  * Naiv: Brute-Force alle w(1)..w(T) durchprobieren O(C^T)

  * VITERBI-Algorithmus
    - Prinzip ähnlich wie Forward-Algorithmus
    - statt Aufsummierung der Wkeiten wird nun bisher Max ÜbergangsWkeiten
      abgespeichert (Produkt max(letzte WKeit)*Übergang jetzt)
    - am Ende via Backtracking die Indizes der Classes des zuletzt Max.
      Pfades auflisten
    - Wegen Nutzung vorheriger Berechungen: O(T*C²)


  6. HMM Training/Learning Problem
  --------------------------------

  * Anwendung
    - Anzahl(=C) der w und Anzahl(=S) der v
    - Auch gegeben: Trainingsdaten mit mehreren beobachteten Folgen v(1)..v(T)
    - Wahrsch. Parameter aij, bjk ?
    - P(w(1) = wi) also Wkeit der HiddenStates zu begin ??


  * Naiv: Maximum-Likelihood und Parameterschätzung
    - liefert keine lösbar analytische Formel
    - liefert keinen bekannten algorithmischen Lösungsweg


  * Algorithmus
    - Numerisch/Iterativ
    - Irgendwelche Anfangswerte nehmen
    - Dekodierung der Trainingsdaten (Ermittlung
      Wahrsch. Hidden-state-Folge)
    - Schätze Parameter neu:
      aij=(Anz. Übergänge wi -> wj)/(Anzahl Vorkommen wi)
    - Wiederhole Dek. bis irgendein Konvergenzkrit. erfüllt ist


  7. in der PRAXIS
  ----------------

  * left-right Modell
    - wi -> wj nur, wenn j>= i
    - Wenn Buchstabe "e" mehrfach im Wort, dann bekommt er eigenen
      Hidden State


  * Isolated Word Recognitition
    - Jedes Wort hat ein HMM(aij,bjk,pi)
    - Jedes Wort wird trainiert (aij,bjk)
    - pi = Wkeit des Hidden State wi
    - unbekanntes Wort: Evaluation, welches HMM am Wahrscheinlichsten
      unbekanntes v emmitiert. Also: max( P(v, wenn aij,bjk gilt) )
    - -> Hidden Markov Toolkit (HTK)


  * ein_
    - 4 Hidden States
    - aij: Matrix nicht ergodisch, wenn nur obere Dreiecksmatrix
      (für jedes A^n bleibt untere Hälfte =0)
    - Beispiel Zeile 2 der 4x4 Matrix: 0  a22  a23  0
      - Erste 0: es passiert bei dem Wort nicht, dass nach "i" wieder
        zurück auf "e" gewchselt wird
      - a22: Wkeit, dass "i" langezogen wird: eiiiin_
      - a23: Wkeit, dass nach "i" ein "n" kommt
      - letztes 0: Wkeit, dass man das "n" bei "ein" vergisst
```



```
  Fehlerabschätzung / Classifier Performance Estimation
  =====================================================

  1. Allgemein
  ------------

  * Classifier: grossPhi(x) ordnet Feature X einer Class zu

  * Baysche/Optimale Fehlerrate: e_B =e ergibt sich, sollte Classifier
  immer die Wahrscheinlichste Klasse Wählen

  * Wahrsch. Klasse unbekannt, weil Wkeit-Verteilung in Praxis unbekannt

  * Fehlerrate e = E( F_grossPhi(x,w)

  * F_grossPhi(x,w) = 0 falls grossPhi(x) = w , sonst 1 (falsch klassif.)

  * tatsächliche/true Errorrate e_T
    - will man abschätzen
    - ist Maß, wie gut Trainingsdaten in der lage sind, Features
      zu klassifizieren
    - expected Error Rate e_E: Mitteln über alle Trainingssets der größe
      n ergibt einen Erwartungswert für e_T


  2. Holdout Estimate
  -------------------

  * Heraushalten von Trainingssamples: ES GIBT TESTDATEN

  * bilden mehrerer Trainingssamples der größe m

  * eDach_T = 1/m * SummeAlleSamples( F_grossPhi(xi,wi) )


  3. Konfidenzniveau / Irrtumswkeit
  ---------------------------------

  * Abweichung Schätzer und "true" soll unwarscheinlicher sein als Alpha

  * P( l p - pDach l >epsilon) <= alpha

  * "true" p liegt mit Wkeit (1-alpha) in Intervall pDach +- epsilon
    - asym: Agresti-Coull
    - asym: HPD


  * KLASSISCH: via Normalverteilung
    - pDach = k/m
    - epsilon = z (1 - alpha/2) * sigma
    - sigma² = 1/m * p(1-p)
    - Trick: nutzte für p in sigma das pDach !
    - Wegen Trick: sigma als Breite ist falsch!


  * AGRESTI-COULL KonfidenzIntervall
    - bei alpha = 5%
    - m und pDach werden durch "quantil" modifiziert
    - m -> m+4
    - p ist nicht mehr k/m sondern: (k+2)/(m+4)


  * Highest Posterior Density HPD
    - ebenfalls Abschätzung von e_T
    - Wkeit für error p hängt nicht nur von k sondern selbst von p ab
    - a = k+1
    - b = m-k +1
    - Nutzung Beta-Verteilung Be(p,a,b)
    - um mit alpha an p1,p2 zu kommen:
    - Bestimmungsgleichungen lösen
      - Be(p1,a,b) = Be(p2,a,b)
      - p1.Integral.p2 Be(p,a,b) dp = 1-alpha


  4. ohne Testdaten
  -----------------
```

```
609
610   * zu wenig manuell klassifizierte Daten?!
611
612   * Nutzen aller Daten zum Training und/oder Testen
613
614   * Resupstitution / Apperent Error Rate
615     - Trainingdaten = Testdaten
616     - zu optimistisch
617     - Entscheidungsgrenzen an Trainingsdaten angepasst
618     - GANZ ÜBEL bei kNN mit k=1: e A=0
619
620
621   * Cross-Validation (Leave-One-Out)
622     - allgemein ist k-fold: aus Trainingsdaten werden k Teilmengen gebildet
623     - k-fold: n=k -> In jeder Teilmenge wurde nur ein Wert weggelassen
624     - e CV = Mittelwert aus allen Fehlklassifizierungen
625
626
627   * Bootstrap Estimate
628     - Schätzung der Varianz des Schätzers
629     - Schätzung des Bias von e_A
630     - Algorithmus
631       - K-oft aus Trainingsdaten T n Samples ziehen mit zurücklegen -> Sk
632       - je Sk: trainiere Classifier
633       - je Sk: e Ak bilden mit Sk+gebildeten Classifiere
634       - je Sk: e Tk bilden mit Sk+Classifiere, der aus T gewonnen wurde
635       - je Sk: Bk = e Tk - e Ak
636       - eDach boot = e A + 1/K * SummeAlle Bk
637
638     - B = E(e_T - e_A)  <=> e_T = e_A + B
639
640
641   * Andere Bootstrap-Methoden
642     - unterschied, wie Bias "gemittelt" wird
643     - Erfahrung: Linearkombination untersch. e A
644     - unterschied, wie Trainingsdaten "gezogen" werden
645
646
647   Markov-Ketten
648   =============
649
650   1. Stochastische Matrix
651   -----------------------
652
653   * Einträge im Intervall [0,1]
654
655   * Summe der Zeileneinträge = 1
656
657   * die i-te Zeile: Spalte j gibt Wkeit an, von i nach j zu wechseln
658
659
660   2. Stochastischer Vek.
661   ----------------------
662
663   * Summe der Einträge ist =1
664
665
666   3. Wkeit pi(t) = P(wi(t)) = SummeAlleC( pj(t-1) * aji )
667   ------------------------------------------------------
668
669   * Also: wegen Unabhängigkeit ist es Summe Wkeit vorheriger Zustand *
670   "Übergangs-Wkeit"
671
672   * Iterativ: pi(t) lässt sich aus AnfangsWkeit vek.p(0)*A^t (a-Matrix t-mal
673   angewand) berechnen
674
675   * Statt pi(t) -> vek.p(t) STOCHASTISCHER VEKTOR
676
677   * Statt aji -> A STOCH. MATRIX
678
679   * vek.p(t) -> Zeilenvektor
680
681
682   4. Übergangsmatrix
683   ------------------
684
```



```
685   * allgemein könnte aij(t) gelten.
686
687   * homogene/stationäre MK: aij nicht von t abhängig
688
689
690   5. markovsche Eigenschaft
691   -------------------------
692
693   * Wkeit eines Zustands zum Zeitpunkt t hängt nur von Wkeit des
694     Zustands w(t-1) ab
695
696   * w älter als t-1 ist nicht relevant (Gedächnislosigkeit)
697
698   * EINE Markov-Kette ist daher mit einer Übergangsmatrix komplett
699     beschreibbar. Jeder Eintrag ist Wkeit für Zustandswechsel
700
701
702   6. stochastischer Prozess
703   -------------------------
704
705   * Abfolge von Zuständen: ZUFÄLLIG, nicht von Zustandsfolge abhängig
706
707   * zur Zeit t ist System in Zustand wi(t)
708
709   * System muss zur Zeit t in einem Zustand sein:
710     Summe über alle wi(t) = 1
711
712
713   7. Allgemein
714   ------------
715
716   * System durchläuft Class
717
718   * Class = Zustand des Systems
719
720   * zu diskr. Zeiten ändert sich (zufällig) Zustand (oder bleibt gleich)
721
722
723   8. Langzeitverhalten
724   --------------------
725
726   * ist A strikt positiv, gilt:
727     - mit jeder vek.p(0) und t -> inf: p(0)*A^t -> p
728     - p hat dann Eigenschaft: p*A = p
729
730
731   * nur ein A^k muss für ein k strikt Positiv sein: A ist ERGODISCH
732
733   Rejection & Konfidenzmaße
734   =========================
735
736   1. Zurückweisen
737   ---------------
738
739   2. Ambiguity Rejection
740   ----------------------
741
742   * selbst die Class, wo Sample am ehesten zu passt, ist
743     nicht sehr Wahrscheinlich
744
745   * Sample liegt auch nahe einer anderen Class
746
747   * oft, wenn mit einzelnen Feature nicht gut die Klassen
748     unterscheidbar sind
749
750   * Finden eines Konfidenzwerts für Feature
751     - fi(x) < f0 dann wird zurückgewiesen, wenn Schwellwert
752       f0 nicht erfüllt
753     - Finden eines Schwellwerts: z.B. mit Leave-One-Out, bis Errorrate mit
754       Trainingsdaten minimal ist
755
756
757   * Chow's Reject Option
758     - geht davon aus, das a posteriori Wkeit P(wi,x) bekannt ist
759     - Fehlerrate immer e(t) <= t
760     - Reject nur, wenn t < 1- 1/C
```

```
    - Satz von Chow
      - Zusammenhang Threshold, Fehlerrate, Akzetanzrate
      - x wird zurückgewiesen, wenn MAX.i(P(wi,x)) < 1 - t
      - r(t) = Reject-Rate
      - e(t) = 0.Integral.t r(s) ds - t * r(t)

    - NAIV: optimales t wenn Accuracy A(t) maximal
      - max bei t=0
      - als Optim.Kriterium dumm
      - t=0 -> alles wird zurückgewiesen

    - OPTIMAL Reject Threshold: QUOTIENT aus Kosten für ein Reject und
      Kosten für ein Fehler


  3. Distance Rejection
  ---------------------

  * Sample so weit von restlichen Trainingsdaten entfernt, dass vermutlich
  zu keiner Class passt

  * Novelty Detection: Ist Sample eine neue, eigene Class?

  * Nutzung einer Distanz ist wie ein "Konfidenzwert", aber hier sind
  es fiktive Kanten/Regionen statt "echte" Nachbarn
    - g(x) = 1/k * SummeAlleNachbarnY d(x,y)


  * Bestimmung der empirischen Verteilung, wie Wahrscheinlich es ist,
  dass Sample den Abstand d zum Nachbar hat
    - F(d) = P( g(x) <=d  )
    - z.B. mit Trainingsdaten via LeaveOneOut
    - Es lässt sich durchschnittliche max. Distanz d_max bestimmen
    - 1 - F(d_max) <= epsilon
    - epsilon = Anteil der Daten, die zurückgewisen werden
    - epsilon weisst aber auch korrekte Daten zurück und bestimmt so
      False-Positive Rate


  4. A(t) = P(correct l accept)
  -----------------------------

  * = P(correct und accept) / P(accept)

  * = ( 1 -e(t) -r(t) ) / (1- r(t))

  Feature Selection & Extraction
  ==============================

  1. Ziele
  --------

  * Verringerung Error Rate

  * Dimensonsreduzierung

  * Selection: Teilmenge nehmen

  * Extraction: Neukombination


  2. 1) Wrapper-Methoden
  ----------------------

  * Classifier-Performance-Schätzer

  * z.B. Leave-One-Out

  * zu Aufwändig wegen Training (außer bei kNN)


  3. 2) Filter-Methoden: Maß unabh. von Classifier
  ------------------------------------------------

  * gute Samples
    - grosser Abstand between
```



```
    - kleiner Abstand within


  * Ges. Maß für Güte der Class
    - Mittelwert 2er Class
    - Min 2er Class


  * Class-Separation via Abstandsmaß
    - 1) theoretisch
      - Chernoff Schranke beschränkt e B nach oben
      - Spezialfall: NV, und s = 0.5
        - Bhattacharyya Distanz B als Abstandsmaß von w1 und w2
        - e_CB = B * wurzel( P(w1) * P(w2) )
        - für B muss man Mittelwerte und Kovarianzmatrizen aus
          Trainingsdaten schätzen

    - 2) Training: Streumatrix
      - S = n * Kdach = (n-1) * SigmaDach
      - S zerlegen in Streuung between Class und innerhalb (within)
      - SW = SumAlleClass[ ni * kDach.i ]
      - SB = SumAlleClass[n.i * (mü.i - mü)(mü.i - mü)^t ]
      - leider nur gut bei unimodal Verteilungen - stark um Mittelwert
      - J1 = Spur(SB)/Spur(SW)
      - J2 = Spur( SW^-1 * SB )


  4. Algorithmen zur Selection
  ----------------------------

  * Ziel: Welche Teilmenge von f Features optimiert J?

  * Brute Force: 2^f

  * max d viele Features: O(f^d)

  * Skalar
    - Idee: Performance nur in einem Feature Messen
      und die Besten Features nehmen
    - AUC der ROC-Kurve
    - Fishers Discriminant Ratio
      - gut, wenn man 2 Dim auf eine Abbilden will
      - FDR = (mü1 - mü2)² / (sigma1² + sigma2²)

    - obiges: nur für 2 Klassen. Trick: Vergleich via Mittelwerte oder
      Min der ermittelten J


  * Vektor
    - Greedy
      - Gefahr, im ersten Extremum hängen zu bleiben
      - Seq Backward Sel.: Lasse immer mehr Features weg, jenachdem
        welches Weglassen die Güte am meisten erhöht
      - Seq Forw. Selection: nehme immer mehr Features dazu

    - Monte-Carlo Methoden
      - springt gelegentlich auf Extrema heraus
      - Features werden wie Chromosomen gemischt


  5. Extraction mit Principal Component Analysis (PCA)
  ----------------------------------------------------

  * Hauptachsentransformation

  * Idee: Suche im Mehrdim. Featurespace Mittelpunkt und Projeziere
  Punkte auf Grade, die durch Mittelpkt geht.

  * Abstand der Punkte zu der Grade sollte um ganzen Minimal sein

  * Die Hauptachse einer Kovarianzmatrix ist in Richtung des Eigenvektors,
  der zum größten Eigenwert gehört. Die Daten Streuen entlang dieser
  Richtung am Stärksten!

  * Independent Component Analysis (ICA) nutzt diese Richtungen, um ohne
  Klasseninfo Richtung der Stärken Abweichung von NormalVerteilung
```

```
  zu finden. -> Ist es dann eine neue Klasse?


  6. Lineare Diskriminanten Analyse (LDA)
  ---------------------------------------

  * im Gegensatz zu PCA: Nutzung von Klasseninfo! 2 Klassen sind Thema

  * J(vek.w) = FDR = abs(m1 - m2)² / (s1² + s2²)

  * mi = <vek.w, vek.mi>

  * mi = Schwerpkt Koordinaten, die auf vek.w projiziert sind

  * abs(m1-m2) = Abstand der Klassmittlpkt proj. auf vek.w

  * si² = emp. Var. der Class i längst vek.w

  * Gesucht w1 und w2 (vek.w)

  * vek.w ist in Richtung SW^-1 * (vek.m1 - vek.m2)


  Clustering
  ==========

  1. Unsupervised Learning
  ------------------------

  2. visual. hochdim. Daten
  -------------------------

  * PCA (Hauptachsentransf.) mit Euklidischer Metrik

  * Multidim. Scaling (MDS)
    - Lösungsweg via Stress-Fktn
      - Sammons Nonlinear Mapping
        - start mit PCA
        - gehe zu nächsten lok. Minimum entlang des stärksten Gradienten

      - isoMDS von Kruskal

    - MDS ist ein f(vek.x), so dass:
      d(vek.xi,vek.xj) ca = d( f(vek.xi),f(vek.xj) )


  3. Finden von Klassen
  ---------------------

  * flach
    - kmeans
      - Fail: durchprobieren alle Möglichen K Teilmengen
        - J(K) = Spur(SW)
        - Minimum von J(K)
        - leider NP-hart

      - Isodata Algorithmus
        - findet nur lokale Minima, könnte aber mit Monte Carlo
          Verfahren verbessert werden
        - es müssen Start-Mittelpkte gewählt werden
        - Mittelpkt-Vektor konvergiert
        - Es werden immer wieder Teilmengen gebildet, wo enthaltene
          x näher an einem Mittelpkt sind, als an Mittelpunkten
          anderer Teilmengen
        - Mittelpkte der Teilmengen werden nach jeder zusammenstellung
          von Teilmengen neu berechnet

      - Wahl der Clusterzahl K bei d Dimensionen und n Samples
        - a) Ellebogen: Stärkster Knick in J(K)-Kurve
        - b) info.theor.Methode
          - minimum von: -2 l(K) + a(n) * d*K
          - l(K) = Maximum der log-Likelihood Funktion
          - d*K = Anzahl Parameter des Modells
          - a(n) = Strafmaß für Komplexität des theor. Models
            - Bayesian Information Criterion (BIC) = ln(n)
            - Akaike Information Criterion (AIC) = 2
```



```

        - c) Calinski-Harabasz Index Maximieren:

        CH(K) = { Spur(SB)/(K-1) } / { Spur(SW)/(n-K) }

    - graph-basiert: MinSpannTree
      - Menge zerfällt in Teilmengen durch entfernen Inkonsistenter Kanten
      - Kante e inkonsistent: kanntengewicht > mü.e + q* sigma.e
      - q =1.96... ist Analogie zu 95% Konfidenzintervall
      - mü.e und sigma.e : Wird aus Kanten berechnet, die max k Schritte
        von e Entfernt sind
      - K muss hier nicht vorher festgelegt werden!


  * hierarchisch
    - agglomerieren mit Dendrogram
      - Zusammenlegen von x, die sich am nächsten sind
      - cdist = Cluster-Distanz
      - gewünschte Clusterzahl K: wenn n-K oft Daten/Cluster
        zusammengefasst wurden

    - single/complete Link Clustering
      - kürzester Abstand der Cluster wird aus kürzestem Abstand
        zwischen den Clustern genommen
      - kürzester Abstand der Cluster wird aus WEITESTEM Abstand
        zwischen den Clustern genommen
```