Andere Methoden zur Klassifikation und Objekterkennung

Heike Zierau

05. Juni 2007

Inhaltsverzeichnis

1 Einführung

Bisher betrachteten wir Methoden, bei denen Daten an ein vorgegebenes Modell angepasst werden sollten, wie z.B. die lineare und nichtlineare Regression, Kernmethoden sowie Kostenfunktionale.

Im Folgenden werden Methoden betrachtet, die einfach und im wesentlichen modellfrei sind. Der Nutzer dieser Methoden benötigt keine Vorkenntnis über die Daten und muss keine Annahmen zur Anwendung der Methode treffen.

Definitionen

- *Trainingsdaten:* N Paare $(x_1, g_1), \ldots, (x_N, g_N)$
- x_i: *Merkmal,* für $i \in \{1, ..., N\}$
  g_i: *Klasse*nbezeichnung, mit $g_i \in \{1, ..., K\}$, für $i \in \{1, ..., N\}$
- *Prototyp:* ein Paar (x_k, g_k), wobei normalerweise $k \notin \{1, ..., N\}$

- *"am nächsten":* euklidischer Abstand im Merkmalsraum bei standardisierten Merkmalen, d.h. Erwartungswert 0 und Varianz 1
- die Dimension des Merkmalsraums entspricht der Anzahl der Merkmale x_i

2 Prototypmethoden

Prototypmethoden repräsentieren die Trainingsdaten durch wenige Punkte im Merkmalsraum. Sie legen die Entscheidungsgrenzen für die Klasseneinteilung fest. Anhand der Prototypen lassen sich neue Daten schnell und einfach klassifizieren.

2.1 K-means Clustering

Idee:

Bei der Simulation von Daten entstehen oft Häufungen (Cluster). Bei K-means Clustering nimmt man diese Häufungen zu Hilfe um Daten zu klassifizieren. Dazu wird eine bestimmte Anzahl von Häufungszentren oder Prototypen festgelegt und zufällig gesetzt. Das ist Ziel, iterativ den Abstand zwischen Daten und Häufungszentrum zu minimieren.

Bei einer unmarkierten Datenmenge, d.h. noch keiner Einteilung in Klassen, werden folgende Iterationsschritte durchgeführt.

Iterationsschritte:

1. gewünschte Anzahl von Startzentren - z.B. R - zufällig setzen
2. Häufung konstruieren durch Punktemenge, die den kleinsten euklidischen Abstand zum Zentrum hat
3. anhand der zum Zentrum gehörenden aktualisierten Punktemenge, den neuen Mittelwert des Zentrums berechnen
4. Schritte 2 und 3 bis zur Konvergenz wiederholen

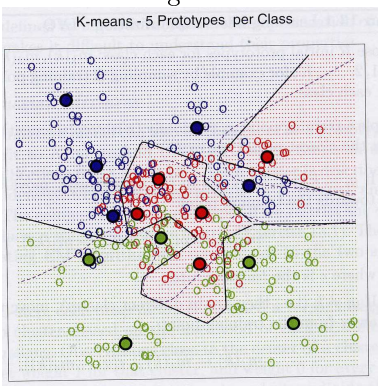
Divergenz ist theoretisch möglich wenn ein Punkt den gleichen Abstand zu zwei Zentren hat. Das Zentrum springt dann hin und her. In der Praxis ist dies jedoch sehr unwahrscheinlich, deshalb gehe ich nicht näher darauf ein.

Bei einer markierten Datenmenge, d.h. wenn die Klassen bereits gekennzeichnet sind, werden folgenden Iterationsschritte durchgeführt.

Iterationsschritte:

1. K-means Clustering auf jede der K Klassen anwenden mit R Prototypen pro Klasse
2. jedem der $K \cdot R$ Prototypen eine Klassenbezeichnung g_k und ein Merkmal x_k zuordnen

Abbildung 1: K-means



3. neue Daten werden der Klasse des nächsten Prototyps zugeordnet

Beispiel

In Abb. 1 ist ein simuliertes Beispiel mit drei Klassen $g_i \in \{\text{rot, grün, blau}\}$ und $R = 5$ Prototypen pro Klasse dargestellt. Die gestrichelte Linie ist die Bayes'sche Entscheidungsgrenze.

Die Entscheidungsgrenze bei K-means Clustering entsteht durch mittelsenkrechte Geraden zwischen zwei Zentren verschiedener Klassen.

Bewertung

K-means Clustering ist eine einfache Methode Daten zu klassifizieren. Jedoch hängen die Ergebnisse stark von der Anzahl der Startzentren und ihrer Startposition ab. Man bekommt keine glatten Entscheidungsgrenzen. Besonders an den Rändern kommt es zu Falschklassifikationen, die man möglichst vermeiden möchte.

2.2 Gaussian Mixture

Gaussian Mixture ist wie K-means Clustering eine Prototypmethode.

Bei Gaussian Mixture nimmt man an, dass die Daten durch eine Mischung von Normalverteilungen enstanden sind, wobei jede Häufung durch eine parametrische Verteilung dargestellt werden kann. Ähnlich wie bei K-means Clustering ist das Ziel die Häufungszentren zu finden um die Daten klassifizieren zu können.

Modell

Seien K Häufungen gegeben. Jede Häufung wurde durch eine Normalverteilung simuliert mit den Parametern μ_k, Σ_k.

Die Daten sind Vektoren im $\mathbb{R}^N$. Gegeben sind insgesamt n konkrete Daten $x_1, \ldots, x_n$.

Jede Häufung k hat die Dichte

$$f_k(x) = \phi(x; \mu_k, \Sigma_k)$$

sowie die a priori Wahrscheinlichkeit α_k, wobei $\sum_{k=1}^{K} \alpha_k = 1$
Die Dichte der Mischung ist gegeben durch

$$f(x) = \sum_{k=1}^{K} \alpha_k f_k(x)$$

Die Position der Zentren wird durch Maximierung der Likelihoodfunktion bestimmt. Da die Anwendung sehr aufwendig ist, greift man auf den Estimation-Maximation Algorithmus zurück.

Iterationsschritte des EM-Algorithmus

1. *Initialisierung p=0*

2. *Schätzschritt bei Iteration p:* Jeder Beobachtung einer Klasse eine Gewichtung zuordnen, d.h. ihre a posteriori Wahrscheinlichkeiten berechnen

$$p_{i,k} = \frac{\alpha_k^{(p)} \phi(x_i; \mu_k^{(p)}, \Sigma_k^{(p)})}{\sum_{k=1}^{K} \alpha_k^{(p)} \phi(x_i; \mu_k^{(p)}, \Sigma_k^{(p)})}, i \in \{1, \ldots, n\}, k \in \{1, \ldots, K\}$$

3. *Maximierungsschritt:* a priori Wahrscheinlichkeiten, Erwartungswert und Kovarianzmatrix aktualisieren

$$\begin{aligned} \alpha_k^{(p+1)} &= \frac{\sum_{i=1}^{n} p_{i,k}}{n} \\ \mu_k^{(p+1)} &= \frac{\sum_{i=1}^{n} p_{i,k} x_i}{\sum_{i=1}^{n} p_{i,k}} \\ \Sigma_k^{(p+1)} &= \frac{\sum_{i=1}^{n} p_{i,k} (x_i - \mu_k^{(p+1)})(x_i - \mu_k^{(p+1)})^t}{\sum_{i=1}^{n} p_{i,k}} \end{aligned}$$

4. Schritt 2 und 3 bis zur Konvergenz wiederholen

Bewertung

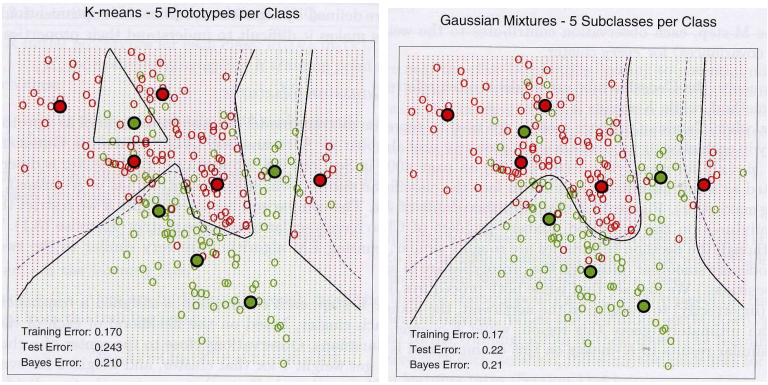
Gaussian Mixture wird oft als „weich", dagegen K-means als „hart" bezeichnet. Der Grund ist die Gewichtung bei Gaussian Mixture im Vergleich zur 0-1 Klassifikation bei K-means. Bei K-means gehört ein Punkt entweder zum einen Zentrum oder zum anderen. Bei Gaussian Mixture entsteht durch die Gewichtung ein fließender Übergang zwischen zwei Zentren.

Gaussian Mixture repräsentiert die Merkmalsdichte jeder Klasse. Es entstehen glatte a posteriori Wahrscheinlichkeiten $\hat{p}(x) = (\hat{p}_1(x), \ldots, \hat{p}_K(x))^T$ für die Klassifikation von x. Die Klassifikationsregel lautet

$$\hat{G}(x) = argmax_k \hat{p}_k(x)$$

wobei $\hat{G}(x)$ die geschätzte Klasse für x bezeichnet

Abbildung 2: K-means vs Gaussian Mixture



2.3 Gaussian Mixture vs. K-means-Clustering

Beim Vergleich von K-means kommt in diesem Beispiel (Abb. 2) ein deutlicher Unterschied zustande.

Die Entscheidungsgrenzen sind sehr ähnlich, aber Gaussian Mixture ist glatter.

Gaussian Mixture ignoriert die grüne Region links oben, K-means nicht.

Für K-means ist die grüne Region in der oberen linken Ecke nicht ignorierbar. Es gibt Punkte, die eindeutig zur grünen Klasse gehören. Somit wird ein Zentrum gesetzt und die Entscheidungsgrenzen festgelegt.

Gaussian Mixture kann diese Region ignorieren. Die a posteriori Wahrscheinlichkeit, dass einer der grünen Punkte zur Klasse grün gehört, wird von den a posteriori Wahrscheinlichkeiten für die rote Klasse überdeckt. Somit weisen die grünen Punkte in der oberen linken Ecke nur eine sehr geringe Merkmalsdichte für die Klasse grün auf und können trotz des Prototyps ignoriert werden.

Es scheint, als wären die Prototypen bei Anwendung der Methoden an der gleichen Stelle. Jedoch treten geringe Verschiebungen auf. Die Ähnlichkeit ist auf die Ähnlichkeit der Methoden zurück zu führen. Es kann sich auch um Zufall handeln.

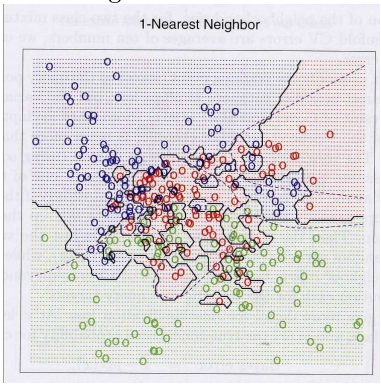
3 nächste-Nachbarn Methode

Die Idee der nächsten-Nachbarn Methode ist Datenmengen in Abhängigkeit ihrer nächsten Nachbarn zu klassifizieren. Dieser Klassifizierer ist erinnerungsbasiert, da die Klassifikation der Nachbarn gespeichert werden muss. Wie bei den Prototypmethoden ist kein anzupassendes Modell notwendig.

3.1 k-nächste-Nachbarn Methode

Vorgehensweise:

Abbildung 3: 1 nächster Nachbar



1. Sei ein noch nicht klassifizierter Punkt x_0 gegeben
2. Finde k Trainingspunkte x_r, $r = 1, ..., k$, mit dem kleinstem euklidischen Abstand zu x_0
3. Klassifiziere durch Mehrheitswahl unter den k Nachbarn, d.h. x_0 wird der Klasse zugeordnen, in der die Mehrheit der k Nachbarn von x_0 enthalten sind.

Eigenschaften

Bei vielen Klassifikationsproblemen wie z.B. handgeschriebene Ziffern, Satellitenbilder, EKG-Bilder liefert die k-nächste-Nachbarn Methode gute Ergebnisse. Sie lässt sich erfolgreich anwenden, wenn viele Prototypen pro Klasse möglich sind, oder unregelmäßige Entscheidungsgrenzen auftreten. Die 1-nächste-Nachbarn Methode steht in engem Zusammenhang mit den Prototypmethoden. Für $k = 1$ ist jeder Trainingspunkt ein Prototyp. In diesem Fall wird bei der Definition in Kapitel 1 eine Ausnahme gemacht, ein Prototyp darf gleich einem Trainingspunkt sein.

Das Ziel bei der k-nächsten-Nachbarn Methode ist es, möglichst wenig Trainingspunkte falsch zu klassifizieren und gleichzeitig Testpunkte richtig einzuordnen. Ein Testpunkt wird mit einer bekannten Verteilung simuliert und anhand der Klasseneinteilung der Trainingspunkte festgestellt, ob er richtig oder falsch einer Klasse zugeordnet wurde.

Dabei repräsentiert der Bias die falschklassifizierten Trainingspunkte, die Varianz die falschklassifizierten Testpunkte.

Die 1-nächste-Nachbarn Methode kann folgendes Ergebnis liefern (Abb. 3).

Die Daten wurden überklassifiziert, es treten keine Falschklassifikationen in den Trainingsdaten auf, jedoch wird es schwierig werden, Testpunkte richtig einzuordnen, d.h. der Bias geht gegen Null, die Varianz ist aber sehr groß.

Bei 15 Nachbarn kann es zu folgenden Ergebnis kommen (Abb. 4).

Es treten vermehrt Falschklassifikationen der Trainingsdaten auf, dafür ist die Klasseneinteilung übersichtlicher. Neue Testpunkte können mit größerer

Abbildung 4: 15 nächste Nachbarn

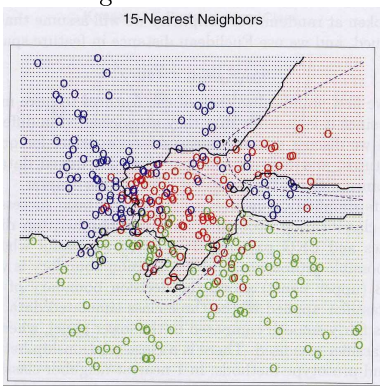
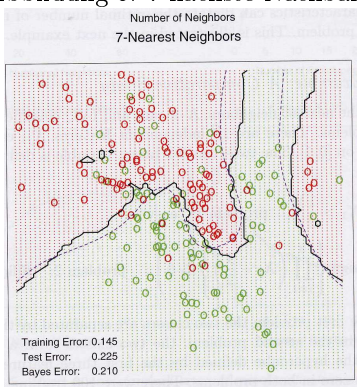


Abbildung 5: 7 nächste Nachbarn



Wahrscheinlichkeit richtig klassifiziert werden, d.h. der Bias ist groß, aber die Varianz ziemlich klein.

Bei 7 nächsten Nachbarn (Abb. 5) scheint die Klasseneinteilung sehr gut zu sein. Es kommt noch zu Falschklassifikationen, jedoch ist die Klassengrenze nachvollziehbar und Testfehler lassen sich gut klassifizieren.

3.2 Asymptotische Eigenschaften

Bei der 1-nächsten-Nachbarn Methode ist der Bias sehr klein.

Das Ergebnis von Cover und Hart von 1967 zeigt, dass es eine obere Schranke für die Fehlerrate bei der 1-nächsten-Nachbarn Methode gibt.

Geht der Trainingsumfang gegen unendlich, so gilt

$$\begin{aligned} &Fehlerrate\, 1-naechster- \\ &\quad Nachbarn\, Klassifikator \quad \leq \quad 2 \cdot Bayessche\, Fehlerrate \end{aligned}$$

Es lässt sich auch eine untere Schranke bestimmen:
Sei $p_k(x)$ die Wahrscheinlichkeit, dass x in der Klasse k liegt.
Sei k^* die dominante Klasse der Nachbarn von x, d.h. $p_{k^*}(x) \geq p_k(x), \forall k = 1, \ldots, K, \, mit \, k \neq k^*$
Dann gilt asymptotisch:

$$\begin{aligned} Bayes\, Fehler &= 1 - p_{k^*}(x) \\ 1 - naechster - Nachbar - Fehler &= \sum_{k=1}^{K} p_k(x)(1 - p_k(x)) \end{aligned}$$

Die Darstellung der Fehlerraten wird in dem Paper von Cover und Hart von 1967 bewiesen. Der Beweis ist sehr aufwendig, weshalb er hier weggelassen wird.
Asymptotisch gilt allgemein:

$$1 - p_{k^*}(x) \leq \sum_{k=1}^{K} p_k(x)(1 - p_k(x)) \leq 2\,(1 - p_{k^*}(x))$$

Beweis
(i)

$$\begin{aligned} \sum_{k=1}^{K} p_k(x)(1 - p_k(x)) &= \sum_{k=1}^{K} p_k(x) - \sum_{k=1}^{K} p_k(x)^2 \\ &\geq 1 - p_{k^*}(x) \sum_{k=1}^{K} p_k(x) \\ &= 1 - p_{k^*}(x) \end{aligned}$$

(ii)

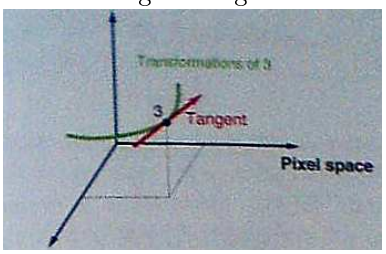
$$\begin{aligned} \sum_{k=1}^{K} p_k(x)(1 - p_k(x)) &= p_{k^*}(x)(1 - p_{k^*}(x)) + \sum_{k \neq k^*} p_k(x)(1 - p_k(x)) \\ &\leq (1 - p_{k^*}(x)) + (1 - p_{k^*}(x)) - \sum_{k \neq k^*} p_k(x)^2 \\ &\leq 2(1 - p_{k^*}(x)) \end{aligned}$$

3.3 Anwendung 1-nächste-Nachbarn Methode

Eine mögliche Anwendung der 1-nächsten-Nachbarn Methode ist die Erkennung handgeschriebener Ziffern. Bei der Handschrift treten oft kleine Veränderungen auf, z.B. kleine Rotationen, wobei die Ziffer für das menschliche Auge immer noch gut erkennbar ist, siehe Abb. 6. Für einen Computer unterscheiden sich die Graustufenwerte der Pixel eines rotierten Bildes stark von denen des Originals.

Abbildung 6: rotierte 3

Abbildung 7: Tangente an 3



Das Ziel ist, anhand der 1-nächsten-Nachbarn Methode dem Computer beizubringen, Ähnlichkeiten zweier Ziffern zu erkennen.

Wir betrachen einen Merkmalsraum mit 256 Dimensionen. Diese entstehen durch Pixelbilder der Größe 16x16.

Eine Dimension entspricht einem Pixel, also einem Merkmal. Ein Pixel kann Graustufenwerte aus $\{1, \ldots, 1024\}$ annehmen.

Ein Punkt im Merkmalsraum wird durch einen 256-dimensionalen Vektor dargestellt. Dieser Vektor repräsentiert eine Ziffer.

Zwei Punkte gehören zu einer Klasse, wenn sich ihre Bilder nur durch eine Rotation unterscheiden.

Bei der Rotation verändern sich die Graustufenwerte der Pixel stetig, wodurch eine geschlossene Kurve im Merkmalsraum beschrieben wird. Bei einer Rotation um 360° liegen die originale und rotierte Ziffer auf einer Kurve. Durch die Rotation verändert sich der euklidische Abstand zwischen den Bildern. Ein Ausweg wäre die Kurve als invariante Metrik zu betrachten und die Ähnlichkeit zwischen zwei Ziffern durch Vergleich ihrer Rotationskurven zu finden.

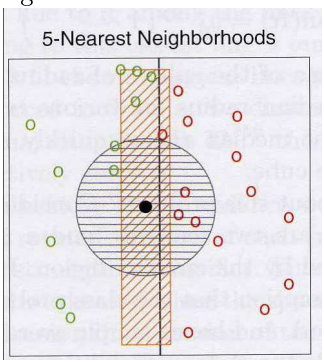
Eine Metrik d heißt invariant, falls $d(x, y) = d(x + a, y + a)$ für alle x, y, a im Merkmalsraum.

Dieser Vergleich ist jedoch mit hohem Rechenaufwand verbunden und die Unterscheidung zwischen einer „6" und einer „9" wäre nicht mehr möglich. Außerdem wurde die Annahme getroffen, dass bei handgeschriebenen Ziffern nur kleine Rotationen auftreten, eine um 45° gedrehte Ziffer erkennt auch das menschliche Auge nur schwer.

Die Lösung bietet die Tangente. Es werden nur kleine Rotationen am Bild durchgeführt, so dass man ein Stück der Rotationskurve erhält. Im Bildpunkt wird an die Kurve eine Tangente gelegt. Durch die Tangente wird der Verlauf der Kurve approximiert, siehe Abb. 7.

Vorgehensweise

Abbildung 8: Problem: 5 nächste Nachbarn



1. an Originalbild kleine Rotation durchführen
2. Tangente im Bild an Rotationskurve legen
3. „ähnlichste" Tangente aus Tangenten der Trainingsmenge finden - z.B. gleiche Richtung, gleicher Winkel
4. das Bild mit der „ähnlichsten" Tangente gehört zur gleichen Klasse

Die Fehlerraten dieser Methode sind sehr klein, vergleichbar mit denen des menschlichen Auges.

Für ein Problem mit 7291 Trainingsbildern und 2007 Testzahlen ergaben sich die folgenden Fehlerraten:

Methode	Fehler
neuronales Netzwerk	0,049
1-Nächster-Nachbar/euklidischer Abstand	0,055
1-Nächster-Nachbar/Tangentenabstand	0.026

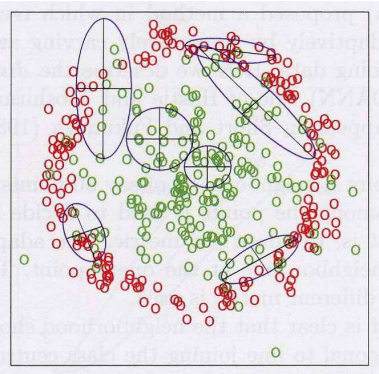
3.4 Adaptive nächste-Nachbarn Methoden

In höherdimensionalen Räumen oder bei kleinem Trainingsdatenumfang können Probleme wie in Abb. 8 auftreten. Es wurden zwei Klassen mit unterschiedlichen Gleichverteilungen simuliert, etwa $U(a, b)$ und $U(b, c)$, mit $a < b < c$. Die senkrechte Linie trennt die Klassen.

Wir betrachten die fünf nächsten Nachbarn des schwarzen Punktes in der Mitte. Er hat drei rote und zwei grüne Nachbarn, d.h. bei Mehrheitswahl unter den Nachbarn gehört er zur roten Klasse. Jedoch nach der Trennlinie liegt er in der grünen Klasse. In diesem Fall ist der euklidische Abstand zur Ermittlung der Nachbarn nicht geeignet.

Einen Ausweg bietet die Mahalanobis-Distanz. Sie beinhaltet nicht nur den Abstand zu einem anderen Punkt, sondern gleichzeitig die Kovarianz zwischen

Abbildung 9: Normalverteilungen mit und ohne Nebenbedingung



den Klassen.

$$d(x,y) = \sqrt{(x-y)\Sigma^{-1}(x-y)^T}$$

Diese Darstellung ergibt sich annäherungsweise durch logarithmieren der Dichte der Normalverteilung mit den Parametern y, Σ.

Für $\Sigma = I$, der Einheitsmatrix, ist die Mahalanobis-Distanz gleich dem euklidischen Abstand und stellt eine kreisförmige Nachbarschaft dar. Für $\Sigma \neq I$ bekommt man eine ellipsoidische Nachbarschaft.

Berechnet man also die Nachbarschaft für den schwarzen Punkt in Abb. 8 anhand der Mahalanobis-Distanz, bekommt man die ellipsoidische Nachbarschaft, angedeutet durch den vertikalen Streifen, und damit die richtige Klassifizierung des Punktes in die grüne Klasse.

Dazu bestimmt man die Mittelwerte beider Klassen, sowie die Kovarianzen zwischen den Klassen. Die Mahalanobis-Distanz berechnet den Abstand eines zu klassifizierenden Punktes zu den Mittelpunkten beider Klassen. In die Klasse, zu deren Mittelpunkt die Mahalanobis-Distanz die kleinste ist, wird der neue Punkt eingeordnet.

Ein kleines Beispiel verdeutlicht, wann es sinnvoller ist, die Mahalanobis-Distanz zu verwenden statt den euklidischen Abstand.

Es werden Zweiklassen-Daten generiert. Die Daten der 1. Klasse sind unabhängig standardnormalverteilt mit der Nebenbedingung nur auf einen Ring um den Mittelpunkt zu liegen: $x \in (a,b), a < b$.

Die Daten der 2. Klasse sind unabhängig standardnormalverteilt ohne Nebenbedingung.

Mit 250 Trainingsdaten pro Klasse bekommt man eine Verteilung der Daten wie in Abb. 9

Die Nachbarschaften der grünen Klasse sind kreisförmig. Hier ist die Kovarianzmatrix Σ gleich der Einheitsmatrix. Die Korrelationen sind Null, wegen der Unabhängigkeit der Daten und die Varianzen = 1. Somit ist also die Mahalanobis-Distanz gleich dem euklidischen Abstand.

Dagegen sind die Nachbarschaften der roten Klasse Ellipsen. Hier ist Σ die Kovarianzmatrix zwischen der roten und der grünen Klasse und ungleich der Einheitsmatrix. Diese Nachbarschaften approximieren die rote Klasse recht gut.

In diesem Fall war es sinnvoll, die Mahalanobis-Distanz zu verwenden, da sie im Sonderfall gleich der euklidischen Distanz ist und sonst die Klassengrenzen besser approximiert.

Im Gegensatz zum euklidischen Abstand ist es jedoch aufwendiger die Mahalanobis-Distanz zu berechnen, weshalb es nicht immer sinnvoll ist, sie anzuwenden.

3.5 Bewertung

k-nächste-Nachbarn Methoden sind einfach anzuwendende Methoden. Es ist keine Vorkenntnis über die Daten erforderlich. Wie wir gesehen haben, liefern sie gute Ergebnisse beim Erkennen handgeschriebener Ziffern.

Allerdings ist für das Auffinden der Nachbarschaften ein hoher Rechenaufwand nötig. Bei N Beobachtungen und p Merkmalen sind Np Operationen nötig um die Nachbarschaften für jeden Folgenpunkt zu finden. Zusätzlich müssen die Klassifizierungen der Nachbarschaftspunkte gespeichert werden um sie für andere Punkte abrufen zu können. Dafür ist viel Speicherplatz notwendig, besonders bei vielen Trainingspunkten oder hohen Dimensionen können hier Probleme entstehen.

Literatur

[1] T. Hastie, R. Tibshirani, J. Friedman. "The elements of statistical learning." Springer, 2001, Kap. 13

[2] http://en.wikipedia.org

[3] http://www.springerlink.de

[4] http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/AppletKM.html