

Einführung in das statistische Lernen

Benjamin Mayer

Universität Ulm

24. April 2007



– Inhalt

- Motivation
- Grundbegriffe
- Kleinste Quadrate und k-nächste-Nachbar-Methode
- Statistische Entscheidungstheorie
- Regressionsmodelle und Kernmethoden
- Zusammenfassung Modellwahl

- Ziele des statistischen Lernens
 - *Aufgabe*: Extraktion von Gesetzmäßigkeiten aus Beobachtungen
 - *Grundproblem* ist die Generalisierung
 - statistische Lerntheorie garantiert die Konsistenz eines entwickelten Modells

- Beispiel 1
 - Gegeben sind spezifische Messdaten der Atmosphäre von gestern und heute
 - Benutze die vorliegenden Messungen zur Vorhersage des Ozonlevels morgen
- Beispiel 2
 - Gegeben ist ein digitalisiertes Bild
 - Jeder Pixel wird einer bestimmten Klasse zugeordnet
 - Anhand der Graustufenwerte soll die Klassenzugehörigkeit des Pixels vorhergesagt werden

- Variablentypen und Terminologie
 - *Eingabe*: vorliegende Daten, beispielsweise Messwerte
 - Diese Daten beeinflussen die *Ausgabe*
 - Man bezeichnet die Eingabe auch als „unabhängige Variablen“
 - und die Ausgabe (oder Reaktionen) entsprechend als „abhängige Variablen“

- Variablentypen und Terminologie
 - Verschiedene Typen in der Ausgabe: qualitativ & quantitativ
 - qualitativ: Man nimmt an, dass die Variable einer bestimmten Klasse oder Gruppe zuzuordnen ist
 - quantitativ: Zum Beispiel „größer, kleiner oder gleich,,
 - Für die Klassen oder Gruppen bei qualitativer Ausgabe verwendet man oft beschreibende Bezeichnungen
 - Man spricht von *Klassifikation* bei qual. Ausgabe und von *Regression* bei quant. Ausgabe

– Variablentypen und Terminologie

- Eingabevariable meist mit X bzw. $X=(X_1, \dots, X_N)$ bezeichnet
- quantitative Ausgabe dargestellt durch Y
- qualitative Ausgabe meist mit G für Gruppe bezeichnet
- Beispiel: Wenn G zwei Klassen hat kann man das binär kodierte Ziel mit Y bezeichnen und es als quantitativen Output auffassen

- Variablentypen und Terminologie
 - Eingabevariablen können ebenfalls von unterschiedlichem Typ sein (qualitativ vs. quantitativ)
 - Unterschiedliche Typen der Eingabevariablen implizieren unterschiedliche Methoden zur Vorhersage
 - Zwei erste Methoden sollen daher nun vorgestellt werden

- Zwei einfache Methoden zur Vorhersage
 - Lineare Modellanpassung über die Methode der kleinsten Quadrate und die k-nächste-Nachbarn-Methode
 - Methode der kleinsten Quadrate macht umfangreiche Strukturannahmen und liefert stabile, aber möglicherweise unflexible Vorhersagen
 - Methode der k-nächsten Nachbarn kommt mit geringeren Annahmen zur Datenstruktur aus; die Vorhersagen sind oft flexibel, aber möglicherweise instabil

- Lineares Modell und Methode der kleinsten Quadrate
 - $X=(X_1, \dots, X_N)$ sei gegeben
 - Y wird vorhergesagt über $\hat{Y}=\hat{\beta}_0+\sum_{j=1}^N X_j\hat{\beta}_j$
 - Man schreibt das lin. Modell in Vektorform $\hat{Y}=X^T\hat{\beta}$
 - Im mehrdimensionalen Fall repräsentiert (X, \hat{Y}) eine Hyperebene
 - Funktionenmodell: $f(X)=X^T\beta$ ist linear und der Gradient $f'(X)$ zeigt in die Richtung des steilsten Anstiegs

– Kleinste Quadrate

- Wähle die Koeffizienten β derart, dass die Summe der quadr. Residuen

$$\text{RSS}(\beta) = \sum_{i=1}^N (y_i - x_i^T \beta)^2$$

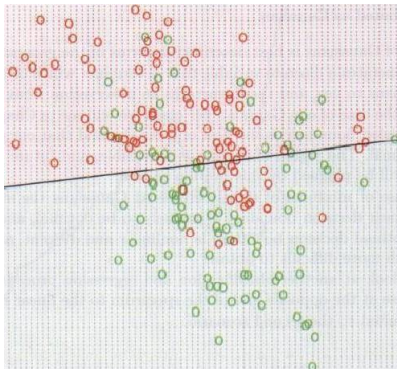
minimal wird

- $\text{RSS}(\beta)$ ist eine quadr. Funktion, hat also immer ein Minimum
- Differenzierung nach β führt zu den Normalgleichungen $X^T(y - X\beta) = 0$
- Der angepasste Wert an der i -ten Inputstelle x_i ist gegeben durch $\hat{y}_i = x_i^T \hat{\beta}$

- Lineares Modell im Klassifikationskontext
 - Output-Klassenvariablen G : grün, rot
 - Y codiert als 0, falls Output grün und Y codiert als 1, falls Output rot
 - Die angepassten Werte \hat{Y} werden umgewandelt zu einer angepassten Klassenvariable \hat{G} nach der Regel

$$\hat{G} = \begin{cases} \textit{rot} & \text{falls } \hat{Y} > 0.5 \\ \textit{grün} & \text{falls } \hat{Y} \leq 0.5 \end{cases}$$

– Lineares Modell im Klassifikationskontext



Rot schattierte Fläche beschreibt den Teil des Inputraums, der als rot klassifiziert wurde

- Lineares Modell im Klassifikationskontext
 - Es sind viele Fehlklassifikationen erkennbar
 - Möglicherweise ist das lineare Modell zu streng
 - Szenario 1: Die Trainingsmenge ist generiert durch eine bivariate Gaußverteilung mit unkorrelierten Komponenten und unterschiedlichen Mittelwerten
 - Szenario 2: Die Trainingsmenge ist generiert durch eine Mischung von Gaußverteilungen mit individuellen Mittelwerten und niedrigen Varianzen
- ⇒ Die optimale Entscheidungsgrenze ist nicht-linear und schwer zu bestimmen

– k-nächste-Nachbar-Methode

- Man betrachtet diejenigen Beobachtungen der Trainingsmenge, die zu einem x am nächsten sind, um \hat{Y} zu bilden
- $\hat{Y}(x) = 1/k \sum_{x_i \in N_k(x)} y_i$, wobei $N_k(x)$ die Nachbarschaft von x ist mit den k nächsten Nachbarn von x
- Finde also die k Beobachtungen, wo x_i ($i=1, \dots, k$) am nächsten zu x im Inputraum ist, und bilde den Durchschnitt ihrer Reaktionen y_i

– k-nächste-Nachbar-Methode



Man kann sehen, dass die Entscheidungsgrenzen, die rot von grün trennen, mehr unregelmäßig sind und auf lokale Cluster reagieren

– k-nächste-Nachbar-Methode

- Betrachtung verschiedener k-Nachbarschaftssysteme bringt verschiedene Entscheidungsgrenzen
- z.B. keine Fehlklassifikation bei $k=1$, jedoch ist die Entscheidungsfunktion komplex
- Effektiver Parameter bei der Methode ist gegeben durch N/k

- Von kleinsten Quadraten zu nächsten Nachbarn
 - Anschein: nächste-Nachbar-Methode macht nicht den Eindruck, dass sie sich auf zwingende Annahmen stützt
 - Jedoch hängt eine bestimmte Unterregion der Entscheidungsgrenze von einer handvoll Inputvariablen und ihren bestimmten Positionen ab und ist deswegen instabil \Rightarrow hohe Varianz, niedrige Verzerrung
 - Oft Mischung aus den Szenarien 1 und 2

– Statistische Entscheidungstheorie

- Wir betrachten den Fall von quantitativem Output
- Mit X als Eingabe und Y als Ausgabe sei $P(X,Y)$ die gemeinsame Verteilung und es gelte $f(X)=Y$
- Erforderlich ist eine Verlustfunktion $L(Y,f(X))$, um den Vorhersagefehler zu bestrafen: $L(Y, f(X)) = (Y - f(X))^2$
- Kriterium für die Wahl von f :
 $EPE(f) = E(Y - f(X))^2 = \int (y - f(x))^2 P(dx, dy)$, der **erwartete** Vorhersagefehler, soll minimal sein
- Durch Konditionierung auf X kann man EPE schreiben als
 $EPE(f) = E_X E_Y([Y - f(X)]^2 | X)$

– Statistische Entscheidungstheorie

- Es erweist sich als ausreichend EPE punktweise zu minimieren: $f(x) = \operatorname{argmin}_c E_{Y|X}([Y - c]^2 | X = x)$
- Die Lösung ist gegeben durch $f(x) = E(Y | X = x)$, die bedingte Erwartung (auch Regressionsfunktion)
- Bei der Methode der nächsten Nachbarn gilt $\hat{f}(x) = \operatorname{Ave}(y_i | x_i \in N_k(x))$
- Für eine große Trainingsmenge gilt: wenn k groß ist, so ist dieser Durchschnitt stabil und falls die ZV iid sind gilt: $\hat{f}(x) \rightarrow E(Y | X = x)$

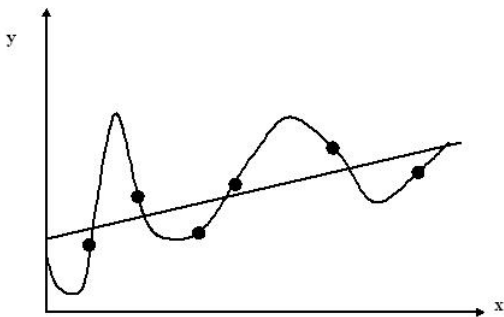
- Statistische Entscheidungstheorie
 - Oft hat man jedoch kein großes N zur Verfügung
 - Man nimmt dann an, dass $f(x)$ annähernd linear in ihren Argumenten ist und $f(x) \approx x^T \beta$
 - Man kann mit diesem Ansatz unter Verwendung von EPE und Differenzierung nach β erreichen, dass gilt
$$\beta = [E(XX^T)]^{-1}E(XY)$$
 - Die vorgestellten Methoden approximieren also beide die bedingte Erwartung über Durchschnitte

– Statistische Entscheidungstheorie

- Wenn man es mit qualitativer Ausgabe zu tun hat ist die Verlustfunktion durch $L(k,l)$ gegeben, die die Strafe darstellt, wenn man eine Beobachtung, die zu G_k gehört, als G_l klassifiziert
- In diesem Fall ist der sogenannte Bayes-Klassifizierer gegeben durch $\hat{G}(X) = \operatorname{argmax}_{g \in G} P(g | X = x)$

- Konsens zwischen Approximation und Schätzung
 - Wenn man keine große Trainingsmenge zur Verfügung hat, sollte die Menge der Funktionen f restringiert werden
 - Wahl zwischen komplexer Funktion und linearer Funktion
 - Je nach Wahl des Modells hat man es mit Approximationsfehlern oder Schätzfehlern zu tun
 - Linearer Ansatz führt eventuell zu hohem Approximationsfehler
 - Polynom-Ansatz (Funktionen höheren Grades) macht es schwer eine Funktion zu finden, die sowohl die Trainingsmenge, als auch weitere Eingaben gut beschreibt

- Konsens zwischen Approximation und Schätzung



Gesucht ist die funktionale Abhängigkeit der schwarzen Punkte.
Welches Modell ist zu bevorzugen?

– Das Gesetz der großen Zahlen

- Für ein Beispiel zur Mustererkennung betrachten wir eine Fehlklassifikation-Verlust-Funktion
- Mit einer festen Funktion f ergibt sich für den Verlust $\xi_j := 1/2 | f(x_j) - y_j |$ entweder 0 oder 1 (mit einer ± 1 -wertigen Funktion f und einer ± 1 -wertigen Variablen y)
- Man spricht also von sogenannten Bernoulli-Stichproben

– Chernoff-Grenze

- Eine berühmte Ungleichung von Chernoff beschreibt wie der empirische Mittelwert $1/N \sum_{i=1}^N \xi_i$ gegen den erwarteten Wert $E(\xi)$ von ξ konvergiert:

$$P(|1/N \sum_{i=1}^N \xi_i - E(\xi)| \geq \epsilon) \leq 2 \exp(-2N\epsilon^2)$$

- Manchmal benutzt man auch die Grenze nach Hoeffding: Seien $\xi_i, i=1, \dots, N$, unabhängige Realisierungen einer beschränkten Zufallsvariablen ξ mit Werten in $[a, b]$. Ihr Durchschnitt sei gegeben durch $Q_N = 1/N \sum_i \xi_i$. Dann gilt für bel. $\epsilon > 0$:

$$\left. \begin{array}{l} P(Q_N - E(\xi) \geq \epsilon) \\ P(E(\xi) - Q_N \geq \epsilon) \end{array} \right\} \leq \exp\left(-\frac{2N\epsilon^2}{(b-a)^2}\right)$$

– Konsistenz

- Die Größe der Trainingsmenge sei N
- Die Funktion f^N , die das empirische Risiko minimiert, soll zu einem Testfehler führen, der gegen den niedrigsten erreichbaren Wert konvergiert
- Ohne Einschränkung der Menge der zulässigen Funktionen ist die Minimierung des empirischen Risikos schwierig
- Zur Erinnerung: Wir nennen einen Schätzer $\hat{\beta}$ für β konsistent, wenn $\hat{\beta} \xrightarrow{P} \beta$, d.h. $\lim P(|\hat{\beta} - \beta| > \epsilon) = 0$

- Gleichmäßige Konvergenz und Konsistenz
 - Lernproblem: Minimiere das Risiko (erwarteter Verlust durch die Trainingsmenge)

$$R[f] = \int_{X \times Y} c(x, y, f(x)) dP(x, y)$$

mit c als Verlustfunktion

- Das Integral kann durch eine endliche Summe approximiert werden mit den Daten der Trainingsmenge und

$$R_{emp}[f] = 1/N \sum_{i=1}^N c(x_i, y_i, f(x_i))$$

nennt man das empirische Risiko

– Gleichmäßige Konvergenz und Konsistenz

- Wenn R_{emp} konsistent: $R_{emp} \xrightarrow{P} R$
- f^{opt} minimiere R mit $R[f] - R[f^{opt}] \geq 0 \forall f \in F$, dann ist das die Wahl unter bekannter Verteilung P und eine optimale Menge von zulässigen Funktionen F
- Ähnlich erhält man $R_{emp}[f] - R_{emp}[f^N] \geq 0$, wenn f^N das emp. Risiko minimiert

- Gleichmäßige Konvergenz und Konsistenz
 - Eine wichtige Folgerung ist, dass das emp. Risiko gleichmäßig gegen das tatsächliche Risiko konvergiert und man erhält
 - $\sup_{f \in F} (R[f] - R_{emp}[f]) \xrightarrow{P} 0$ für $N \rightarrow \infty$
 - Die gleichmäßige Konvergenz ist also eine hinreichende Bedingung für die Konsistenz der Minimierung des emp. Risikos
 - Man sucht also nach Funktionen, die die gleichmäßige Konvergenz sichern

– Statistische Modelle

- Ziel: Finde Approximation $\hat{f}(x)$ von $f(x)$, die der vorausgesagten Beziehung zwischen Eingabe und Ausgabe unterliegt
- Diese Approximation soll unter einer gegebenen Trainingsmenge gefunden werden
- Vorstellbar sind z.B. Polynome höheren Grades oder trigonometrische Polynome zur Approximation
- Ein Ansatz ist die Methode der kleinsten Quadrate zu nutzen
- Man schätzt den Parameter θ in f_θ , indem man

$$\text{RSS}(\theta) = \sum_{i=1}^N (y_i - f_\theta(x_i))^2$$

minimiert

– Statistische Modelle

- Oder: Nutze das Prinzip der Maximum-Likelihood-Schätzung
- ML-Schätzer: $L(\theta) = \sum_{i=1}^N \log P_{\theta}(y_i)$
- Die bedeutensten Werte für θ haben das höchste P

– Regressionsmodelle

- Problem: Minimierung von $RSS(\theta)$ liefert jede Funktion f als Lösung, die die Trainingsdaten (x_i, y_i) beschreibt
- Jede bestimmte Wahl von f kann möglicherweise also eine schlechte Vorhersage sein für Daten, die nicht der Trainingsmenge entstammen
- Die Menge der zulässigen Funktionen sollte also restringiert werden (aber auch hierbei gibt es unzählige Möglichkeiten!)
- Ansatz: die Strenge der Nebenbedingungen für f ist abhängig von der Größe der Nachbarschaft der Trainingsdaten (je größer, desto strenger sollte die NB sein)

– Lernen mit Kernmethoden

- Kernmethoden basieren auf Ähnlichkeitsmaßen, die aus der Darstellung von Mustern entstehen
- Man hat 2 Klassen im Output, ein neues Objekt soll einer Klasse zugeordnet werden
- $(x_1, y_1), \dots, (x_N, y_N) \in X \times \{\pm 1\}$
- X ist die Inputmenge (Menge der Muster)
- $\{\pm 1\}$ ist die Outputmenge (binäre Klassifikation)

– Beispiel

- Schafe in zwei Klassen einordnen
- x_i : Schafe
- y_i : schwarz, weiß
- Für gegebenes $x \in X$ wollen wir die Klasse vorhersagen
- Man sucht also zu x ein y , so dass (x,y) ähnlich zu den vorhandenen Daten ist

– Ähnlichkeitsmaße

- Als Maß verwenden wir $k: X \times X \rightarrow \mathbb{R}, (x, x') \mapsto k(x, x')$
- k nimmt zwei Daten der Eingabe (x und x') und gibt eine Zahl zurück, welche die Ähnlichkeit der beiden Daten wiedergibt
- Man nennt k eine *Kernfunktion*

– Ähnlichkeitsmaße

- Anderes Maß ist das Skalarprodukt $\langle x, x' \rangle = \sum_{i=1}^N x_i x'_i$
- geom. Interpretation: Bestimmung vom \cos des Winkels zwischen den Vektoren x und x' , vorausgesetzt sie sind normiert auf Länge 1
- Es gilt $\|x\| = \sqrt{\langle x, x \rangle}$ und der Abstand von x zu einem x' ist gegeben durch die Länge des Differenzvektors
- Das Skalarprodukt ist allerdings für komplizierte Probleme nicht geeignet

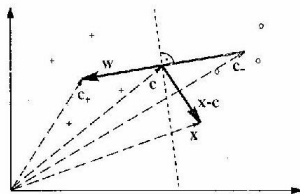
– Ähnlichkeitsmaße

- Allgemein nutzt man eine Abb. $\Phi: X \rightarrow H$, $x \mapsto \Phi(x)$, wobei H ein Raum mit innerem Produkt ist
- Dieser Ansatz hat den Vorteil, dass man die Muster geometrisch handhaben kann und man Φ frei wählen kann, d.h. man kann viele Ähnlichkeitsmaße konstruieren. Außerdem gilt $k(x, x') = \langle x, x' \rangle = \langle \Phi(x), \Phi(x') \rangle$

– Einfacher Lernalgorithmus zur Mustererkennung

- Idee: ein bisher ungesehenes Muster einer Klasse zuordnen
- Algorithmus:
 - 1 Berechnung der Mittelwerte der Klassen: $c_+ = 1/m_+ \sum_{i|y_i=+1} x_i$
und $c_- = 1/m_- \sum_{i|y_i=-1} x_i$
 - 2 Neues Objekt x wird der Klasse zugeordnet, wo x näher an c_+
bzw. c_- liegt
- Benutze dann ein Ähnlichkeitsmaß, um zur Entscheidungsfunktion zu gelangen

- Einfacher Lernalgorithmus zur Mustererkennung
 - Auf halbem Weg zwischen c_+ und c_- liegt der Punkt $c := (c_+ + c_-)/2$
 - Man bestimmt die Klassenzugehörigkeit von x , indem man überprüft, ob der Vektor $x-c$ (Verbindung von x und c) einen Winkel kleiner $\pi/2$ einschließt mit dem Vektor $w := c_+ - c_-$



- Klassifizierung mit Hyperebenen
 - Separation der Trainingsmenge über einen linearen Ansatz
 - Es existiert immer eine optimale Hyperebene
 - Diese hat zu jedem Punkt der Trainingsmenge eine maximale Differenz

- Klassifizierung mit Support-Vektoren
 - Idee: Bilde die Trainingsmenge mit Hilfe einer Abb. Φ in einen höherdimensionalen Merkmalraum ab
 - Konstruiere eine separierende Hyperebene, die dort maximale Differenz hat
 - Dies ergibt eine nicht-lineare Entscheidungsgrenze im Inputraum
 - Unter Verwendung einer Kernfunktion ist es möglich eine solche Hyperebene zu berechnen, ohne explizit die Abbildung im Merkmalraum auszuführen

– Support-Vektor-Regression

- Befasst sich mit der Schätzung von reell-wertigen Funktionen
- Darstellung des Verlustes durch die Vorhersage $f(x)$ für y :
 $c(x,y,f(x)) := |y - f(x)|_\epsilon := \max(0, |y - f(x)| - \epsilon)$
- Um die lin. Regression $f(x) = \langle w, x \rangle + b$ zu schätzen, minimiert man $1/2 \|w\|^2 + C \sum_{i=1}^N |y_i - f(x_i)|_\epsilon$
- Man erhält einen Regressionschätzer der Form
 $f(x) = \sum_{i=1}^N (\alpha_i^* - \alpha_i) k(x_i, x) + b$

- Zusammenfassung Modellwahl
 - Es ist also ein Lernmodell zu wählen, so dass die zu Verfügung stehende Menge an Funktionen nicht zu groß ist
 - Bei der Modellwahl ist immer ein Konsens zu treffen zwischen Approximationsfehler und Schätzfehler
 - Genaueres zu den verschiedenen Ansätzen in den folgenden Vorträgen

– Literatur

- T. Hastie, R. Tibshirani, J. Friedman „The elements of statistical learning“
- B. Schölkopf, A. Smola „Learning with kernels“
- <http://www.google.de>, <http://www.wikipedia.org>