



The words “true model” represent an oxymoron.

— Burnham und Anderson (2002)

Was wollen wir diesesmal kennenlernen?

- ▶ Über lineare Modelle hinausgehen:
Maschinelles Lernen als Funktionsapproximation
- ▶ Funktionsapproximation optimiert eine **Qualitätsfunktion** wie bspw.
 - ▶ Fehlerminimierung (RSS, MSE)
 - ▶ Maximierung der Likelihood
- ▶ Für die **Modellselektion** und -evaluierung kann man
 - ▶ die Kreuzvalidierung bei Fehlerminimierung,
 - ▶ die Kriterien nach Bayes (BIC, MDL) und
 - ▶ einige weitere empirische Maße
- benutzen.

Was ist Funktionsapproximation?

- Das bisher vorgestellte Verfahren des Maschinellen Lernens —lineare Modelle— ist eine Instanzen der **Funktionsapproximation**:
- **Gegeben** sind die Trainingsbeispiele \mathcal{T} , **gesucht** ist eine Funktion

$$f_{\theta}(\vec{x}) = \sum_{k=1}^K h_k(\vec{x})\theta_k .$$

- * Dazu müssen wir die **Parameter** $\theta = (\theta_1, \dots, \theta_K)$ abgeschätzt werden, bei den linearen Modellen war das β .
- * Darüber hinaus können die Daten mittels $h_k(\vec{x})$ (**Basisfunktionen**) in einen Raum **transformiert** werden, der für das Lernen besser geeignet ist.
- * Optimiert wird ein **Qualitätskriterium**, z.B. wird eine Verlustfunktion minimiert oder die Wahrscheinlichkeit maximiert.

Im Kern ist eine **Zielfunktion** etwas sehr Einfaches: Sie ist ein Weg, um zu bewerten, wie gut das Model unseren Datensatz widerspiegelt.

Verlustfunktion (loss function) Fehler minimieren als Abstand zwischen wahrem Wert und Ergebnis der gelernten Funktion, z.B. $RSS(\theta)$ minimieren. Wenn die Vorhersagen völlig daneben liegen, wird die Verlustfunktion eine höhere Zahl ausgeben. Wenn sie ziemlich gut sind, wird sie eine niedrigere Zahl ausgeben. Das haben wir bereits gesehen.

Likelihood: Wahrscheinlichkeit der wahren Werte maximieren! Das schauen wir uns jetzt an.

Maximum Likelihood bei der Klassifikation

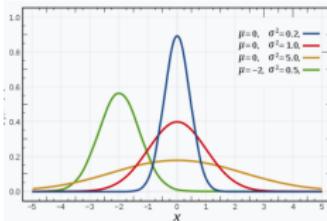


Gegeben eine Verteilung $Pr_\theta(y)$ und eine Stichprobe dieser Verteilung y_1, \dots, y_N , ist der (log) Likelihood die logarithmierte Wahrscheinlichkeit:

$$L(\theta) = \sum_{i=1}^N \log Pr_\theta(y_i) \quad (1)$$

Genau das θ , das y_i am wahrscheinlichsten macht, ist gut – $L(\theta)$ maximieren!

- ▶ Wir sollten dafür eine Verteilung annehmen, da wir die wahre Verteilung nicht kennen.
- ▶ Oft ist die Normalverteilung eine gute Annahme.



normalverteilt

Eine Zufallsvariable X heißt **normalverteilt** mit den Parametern μ (Mittelwert) und σ (Standardabweichung), wenn sie die Dichtefunktion

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}((x-\mu)/\sigma)^2} \quad (2)$$

besitzt.

Normalverteilung

Die zugehörige Wahrscheinlichkeitsverteilung $X \sim \mathcal{N}(\mu, \sigma^2)$ heißt **Normalverteilung**, der Graph ihrer Dichtefunktion wird Gaußsche Glockenkurve genannt.

Bei linearen Modellen ist die Maximum Likelihood gleich der Minimierung von RSS



- Wir wollen die Parameter θ schätzen, so dass die richtige Ausprägung von Y auch die wahrscheinlichste Ausprägung ist, wenn X und θ gegeben.
- Unter der **Annahme der Normalverteilung** haben wir

$$Pr(Y|X, \theta) = \mathcal{N}(f_\theta(X), \sigma^2).$$

Wir sagen also $Y = f_\theta(X) + \epsilon$ mit

$$\epsilon \sim Pr(Y|X, \theta) = \mathcal{N}(0, \sigma^2).$$

- Einsetzen ergibt, dass die Log-Likelihood der Daten nun gerade $RSS(\theta)$ entspricht:

$$L(\theta) = \sum_{i=1}^N \log\left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_i - f_\theta(\vec{x}_i)}{\sigma}\right)^2}\right) = C_2 + C_1 \cdot \sum_{i=1}^N (y_i - f_\theta(\vec{x}_i))^2$$

Wie das?

Herleitung von $L(\theta) = RSS(\theta) \cdot C_1 + C_2$ bei Normalverteilung

$$\begin{aligned} L(\theta) &= \sum_{i=1}^N \log\left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_i - f_\theta(\vec{x}_i)}{\sigma}\right)^2}\right) \\ &= \sum_{i=1}^N \left(\log(1) - \log(\sigma\sqrt{2\pi}) + \log(e^{-\frac{1}{2}\left(\frac{y_i - f_\theta(\vec{x}_i)}{\sigma}\right)^2}) \right) \\ &= \sum_{i=1}^N \left(0 - \log(\sigma) - \log(\sqrt{2\pi}) - \frac{1}{2}\left(\frac{y_i - f_\theta(\vec{x}_i)}{\sigma}\right)^2 \right) \\ &= \underbrace{-N \cdot \log(\sigma) - \frac{N}{2} \log(2\pi)}_{=:C_2} - \underbrace{\frac{1}{2\sigma^2} \sum_{i=1}^N (y_i - f_\theta(\vec{x}_i))^2}_{=:C_1} \\ &= RSS(\theta) \cdot C_1 + C_2 \end{aligned}$$

N, σ sind konstant für einen Datensatz.

Log-likelihood bei nominalem Y ist Entropie

Wenn man $L(\theta)$ maximiert, passen die Parameter θ also gut zu den Daten im Sinne der Likelihood.

Cross-Entropie

Sei Y eine Zufallsvariable, die als Werte die Namen von K verschiedenen Klassen annimmt.

$$\Pr(Y = y_k | X = \vec{x}) = p_{k,\theta}(\vec{x}), k = 1, \dots, K$$

$$L(\theta) = \sum_{i=1}^N \log(p_{y_i, \theta}(\vec{x}_i)) \tag{3}$$

Als Verlustfunktion würde man $-L(\theta)$ nehmen!

- ▶ Wir haben bisher eine Modellklasse kennengelernt: **lineare Modelle**.
- ▶ Bei der Verallgemeinerung zur Funktionsapproximation haben wir außerdem **Basisfunktionen** zur Vorverarbeitung gesehen, die ebenfalls Modellklassen induzieren.
- ▶ Wie wählen wir aber nun Modelle aus?

1. Kreuzvalidierung für verschiedene Modelle – das mit dem geringsten durchschnittlichen Fehler nehmen! (Minimierung der Verlustfunktion jetzt auf der Ebene der Modelle)
2. Direkt anhand der a posteriori Wahrscheinlichkeit Modelle vergleichen. (Maximierung der Wahrscheinlichkeit jetzt auf der Ebene der Modelle)
 - ▶ Bayes Information Criterion
 - ▶ Minimum Description Length

Kreuzvalidierung zur Modellselektion



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Gegeben eine Klasse von Modellen $f(\vec{x}, \alpha)$, wobei α ein Modell der Klasse indiziert, eine Verlustfunktion $L(y, f(\vec{x}, \alpha))$, N Beispiele und eine Aufteilung der Beispiele in K Partitionen mit der Indexfunktion $\kappa : \{1, \dots, N\} \rightarrow \{1, \dots, K\}$, die für jede Beobachtung die zugehörige Partition angibt.

Kreuzvalidierung für alle Modelle:

- ▶ Lasse die $\kappa(i)$ -te Partition aus,
- ▶ lerne das α -te Modell: $\hat{f}^{-\kappa(i)}(\vec{x}, \alpha)$.
- ▶ rechne den Fehler aus:

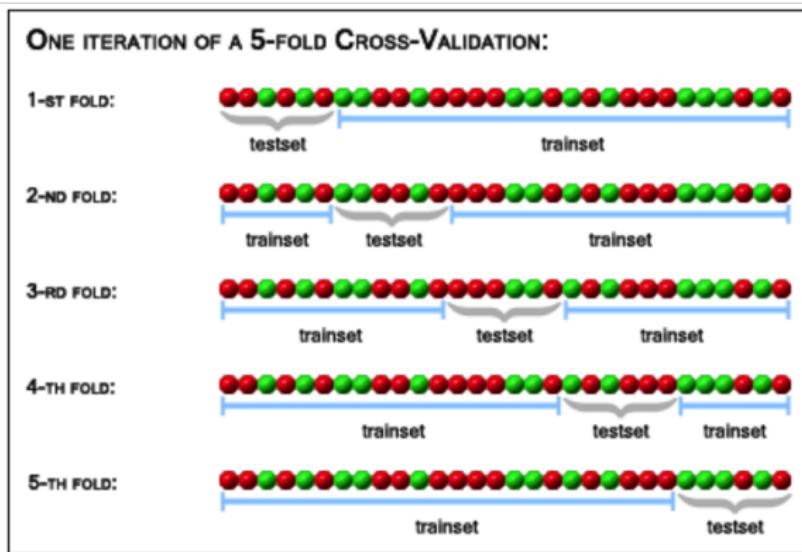
$$CV(\alpha) = \frac{1}{N} \sum_{i=1}^N L(y_i, \hat{f}^{-\kappa(i)}(\vec{x}_i, \alpha))$$

- ▶ Minimiere $CV(\alpha)$, wähle also das Modell mit dem geringsten Verlust.

Kreuzvalidierung zur Modellselektion



Das kennen wir ja schon!



Wenn wir Wahrscheinlichkeiten zu Hand haben, können wir aber auch diese ausnutzen!

A posteriori Wahrscheinlichkeit

Gegeben eine beliebige Einteilung von X in Klassen y_1, y_2, \dots, y_K und eine Beobachtung $\vec{x} \in X$. Die Wahrscheinlichkeit von y_j unter der Bedingung, dass \vec{x} beobachtet wird, ist

$$Pr(y_j|\vec{x}) = \frac{Pr(y_j)Pr(\vec{x}|y_j)}{Pr(\vec{x})} \quad (4)$$

$Pr(y_j)$ ist die **a priori** Wahrscheinlichkeit der Klasse. $Pr(y_j|\vec{x})$ ist die **a posteriori** Wahrscheinlichkeit der Klasse.

Gegeben eine Menge von Modellen $\mathcal{M}_m, m = 1, \dots, M$ mit entsprechenden Parametern θ_m , Trainingsdaten \mathcal{T} und eine Verteilung $Pr(\theta_m | \mathcal{M}_m)$, dann ist die **a posteriori Wahrscheinlichkeit eines Modells**

$$Pr(\mathcal{M}_m | \mathcal{T}) \sim Pr(\mathcal{M}_m) \cdot Pr(\mathcal{T} | \mathcal{M}_m).$$

Vergleich von Modellen mittels Wahrscheinlichkeiten

Gegeben dass $Pr(\mathcal{M}_I | \mathcal{T}) \neq 0$, $Pr(\mathcal{T} | \mathcal{M}_I) \neq 0$, $Pr(\mathcal{M}_I) \neq 0$, können wir zwei Modelle $\mathcal{M}_j, \mathcal{M}_I$ vergleichen, in dem wir folgenden Quotienten berechnen:

$$\frac{Pr(\mathcal{M}_m | \mathcal{T})}{Pr(\mathcal{M}_I | \mathcal{T})} = \frac{Pr(\mathcal{M}_m)}{Pr(\mathcal{M}_I)} \cdot \frac{Pr(\mathcal{T} | \mathcal{M}_m)}{Pr(\mathcal{T} | \mathcal{M}_I)}$$

Ist das Ergebnis > 1 , nehmen wir \mathcal{M}_m , sonst \mathcal{M}_I .

Approximieren der a posteriori Wahrscheinlichkeit



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Wenn alle Modelle a priori gleich wahrscheinlich sind, müssen wir nur $Pr(\mathcal{T}|\mathcal{M}_i)$ approximieren. Allgemein sollten wir aber die Kapazität von Modellen berücksichtigen

- ▶ Mit Maximum Likelihood schätzen wir $\hat{\theta}_i$.
- ▶ Die Anzahl freier Parameter in \mathcal{M}_i (Kapazität) bezeichnen wir mit d_i . Das ist z.B. die Zahl der Parameter, kann aber wegen $h_k(\vec{x})$ oder einiger Eigenschaften des Lernverfahrens auch etwas anderes sein.
- ▶ Als Wahrscheinlichkeit nähern wir an:

$$\log Pr(\mathcal{T}|\mathcal{M}_i) = \log Pr(\mathcal{T}|\hat{\theta}_i, \mathcal{M}_i) - \frac{d_i}{2} \cdot \log N + O(1) \quad (5)$$

Maximale a posteriori Wahrscheinlichkeit und BIC

Bayes Informationskriterium

Sei d die Anzahl der Parameter eines Modells und N die Anzahl der Beispiele, dann ist das Bayes Informationskriterium BIC

$$BIC = -2 \loglik + (\log N) \cdot d \quad (6)$$

Dabei ist $\loglik = \sum_{i=1}^N \log Pr_{\hat{\theta}}(y_i)$.

BIC als Qualitätskriterium bei Likelihood Maximierung wählt eher einfache Modelle. Unter einer Gaußschen Verteilung und bei bekannter Varianz σ^2 rechnen wir

$$-2 \loglik \sim \sum_i \frac{(y_i - \hat{y}_i)^2}{\sigma^2}$$

Die Wahl des Modells mit kleinstem BIC entspricht der Wahl des Modells mit größter a posteriori Wahrscheinlichkeit.

Relative Qualität der Modelle per BIC

- Die Wahl des Modells mit kleinstem BIC ist zuverlässig. Gegeben eine Familie von Modellen, darunter das richtige, konvergiert die Wahrscheinlichkeit, dass BIC das richtige wählt, gegen 1, wenn die Anzahl der Beispiele gegen ∞ konvergiert.
- Wenn wir für jedes Modell \mathcal{M}_m , $m = 1, \dots, M$ den BIC ausrechnen, können wir (wie bei Kreuzvalidierung auch) die Modelle relativ zueinander bewerten, hier:

$$\frac{e^{-\frac{1}{2} \cdot BIC_m}}{\sum_{l=1}^M e^{-\frac{1}{2} \cdot BIC_l}} \quad (7)$$

Eine andere Sicht bietet die Minimum Description Length

Stellen wir uns vor, dass wir uns Nachrichten als SMS schicken. Wir wollen möglichst wenig Geld ausgeben. Wir wollen also möglichst viel Information in wenig Text kodieren.

Modelle kodieren Beispiele

Wir können Nachrichten so kodieren, dass keine Nachricht ein Präfix einer anderen Nachricht ist, z.B.

Nachricht	z1	z2	z3	z4
Code	0	10	110	111

Wir wollen den kürzesten Code für die häufigste Nachricht. Der Code des Beispiels ist optimal, wenn $Pr(z1) = 1/2$, $Pr(z2) = 1/4$, $Pr(z3) = 1/8$, $Pr(z4) = 1/8$.

Wieso das?

Code-Länge als Entropie

Wählen wir die Code-Länge l_i einer Nachricht z_i als

$$l_i = -\log_2 Pr(z_i)$$

so ist die durchschnittliche Nachrichtenlänge

$$\text{length} \geq - \sum Pr(z_i) \log_2(Pr(z_i)) \quad (8)$$

Wenn $p_i = A^{-l_i}$, wobei A die Anzahl der verwendeten Zeichen ist, gilt sogar die Gleichheit (s. Beispiel): $Pr(z_1) = 1/2 = 2^{-1} = A^{-l_1}$, $A = 2$, $l_1 = 1$

Minimum Description Length zur Modellselektion

Gegeben ein Modell \mathcal{M} mit Parametern θ und Beispiele $\mathcal{T} = (\mathbf{X}, \mathbf{y})$, der Empfänger kennt alle \mathbf{X} und soll die \mathbf{y} empfangen. Dazu müssen wir den Unterschied zwischen Modell und wahren Werten sowie die Modellparameter übermitteln.

Prinzip der Minimum Description Length MDL

Wähle immer das Modell mit der kürzesten Nachrichtenlänge!

$$\textit{length} = -\log \Pr(\mathbf{y}|\theta, \mathcal{M}, \mathbf{X}) - \log \Pr(\theta|\mathcal{M}) \quad (9)$$

Eigenschaften von MDL



- ▶ Bei normalverteilten y, θ , wenn wir \mathbf{X} zur Einfachheit weglassen, sehen wir den Einfluss von σ :

$$\text{length} = \log \sigma + \frac{(y - \theta)^2}{\sigma^2} + \frac{\theta^2}{2}$$

- ▶ Je kleiner σ desto kürzer die Nachricht und einfacher das Modell!

- Wenn wir die Länge (9) minimieren

$$\text{length} = -\log \Pr(\mathbf{y}|\theta, \mathcal{M}, \mathbf{X}) - \log \Pr(\theta|\mathcal{M})$$

maximieren wir auch die a posteriori Wahrscheinlichkeit (vgl. Gleichung 4)
 $\Pr(\mathbf{y}|\mathbf{X})$.

- Mit Hilfe des BIC haben wir Modelle für die Funktionsapproximation durch Maximum Likelihood ausgewählt: das Modell mit dem kleinsten BIC entspricht dem Modell mit größter a posteriori Wahrscheinlichkeit.
- Also kann man das Modell mit der kleinsten Code-Länge (MDL-Prinzip) auch durch die Minimierung des BIC finden.

Was wissen Sie jetzt?

- ▶ Funktionsapproximation optimiert eine **Qualitätsfunktion**.
 - ▶ Fehlerminimierung, z.B. RSS, MSE
 - ▶ Maximierung der **Likelihood**, z.B. durch Approximation der a posteriori Wahrscheinlichkeit
 - ▶ Fehlerminimierung RSS entspricht Maximum Likelihood, falls Normalverteilung gegeben (Regression).
- ▶ Für die **Modellselektion** kann man
 - ▶ die Kreuzvalidierung mit Fehlerminimierung und
 - ▶ die Kriterien nach Bayes (BIC, MDL) nutzen.

- ▶ Validierung durch **Experten**
 - ▶ Experte im Anwendungsgebiet untersucht, wie plausibel das gelernte Modell ist
 - ▶ oft die einzige Möglichkeit, aber es ist subjektiv, zeit-intensiv und kostspielig
- ▶ On-line Validierung
 - ▶ Man validiert das gelernte Model direkt in der Anwendung
 - ▶ ergibt die beste Schätzung, wie nützlich das gelernte Model hat
 - ▶ aber für schlechte Modelle müssen wir einen hohen Preis bezahlen
- ▶ Man sollte immer empirische **Maße der Güte** angeben

Konfusionsmatrix (Zwei-Klasse Probleme)

	Classified as +	Classified as -	
Is +	true positives (tp)	false negatives (fn)	$tp + fn = P$
Is -	false positives (fp)	true negatives (tn)	$fp + tn = N$
	$tp + fp$	$fn + tn$	$ E = P + N$

- ▶ Die Konfusionsmatrix fasst alle wichtigen Informationen zusammen
 - ▶ wie häufig wird Klasse i mit Klasse j verwechselt
- ▶ Fast alle Maße der Güte lassen sich aus der Konfusionsmatrix ablesen
 - ▶ Accuracy
 - ▶ Recall/Precision, Sensitivity/Specificity

Grundlegende Evaluierungsmaße

- ▶ **True Positive Rate:** $tpr = \frac{tp}{tp+fn}$
 - ▶ Prozent der korrekt klassifizierten positiven Beispiele
- ▶ **False Positive Rate:** $fpr = \frac{fp}{fp+tn}$
 - ▶ Prozent der negativen Beispiele, die positive eingeschätzt werden
- ▶ **False Negative Rate:** $fnr = \frac{fn}{tp+fn} = 1 - tpr$
 - ▶ Prozent der positiven Beispiele, die negativ eingeschätzt werden
- ▶ **True Negative Rate:** $tnr = \frac{tn}{fp+tn} = 1 - fpr$
 - ▶ Prozent der korrekt klassifizierten negativen Beispiele
- ▶ **Accuracy:** $acc = \frac{tp+tn}{N+P}$
 - ▶ Prozent der korrekt klassifizierten Beispiele
- ▶ **Error:** $err = \frac{fp+fn}{N+P}$
 - ▶ Prozent der falsch klassifizierten Beispiele

Konfusionsmatrix (Mehr-Klassen Probleme)

		classified as				
		A	B	C	D	
true class	A	$n_{A,A}$	$n_{B,A}$	$n_{C,A}$	$n_{D,A}$	n_A
	B	$n_{A,B}$	$n_{B,B}$	$n_{C,B}$	$n_{D,B}$	n_B
	C	$n_{A,C}$	$n_{B,C}$	$n_{C,C}$	$n_{D,C}$	n_C
	D	$n_{A,D}$	$n_{B,D}$	$n_{C,D}$	$n_{D,D}$	n_D
		\bar{n}_A	\bar{n}_B	\bar{n}_C	\bar{n}_D	$ E $

- ▶ Für Multi-Klassen Probleme hat die Konfusionsmatrix mehr Einträge
- ▶ Die Accuracy ergibt sich als die Diagonalsumme geteilt durch die Zahl der Beispiele $|E|$.

Aber wie kann man nun sagen, dass ein Lernalgorithmus besser ist als ein anderer?



- Wenn wir n Lernalgorithmen auf m Datensäzen evaluiert haben (Accuracy).

Dataset	Grading	Select	Stacking	Voting	Dataset	Grading	Select	Stacking	Voting
audiology	83.36	77.61	76.02	84.56	hepatitis	83.42	83.03	83.29	82.77
autos	80.93	80.83	82.20	83.51	ionosphere	91.85	91.34	92.82	92.42
balance-scale	80.89	91.54	89.50	86.16	iris	95.13	95.20	94.93	94.93
breast-cancer	73.99	71.64	72.06	74.86	labor	93.68	90.35	91.58	93.86
breast-w	96.70	97.47	97.41	96.82	lymph	83.45	81.69	80.20	84.05
colic	84.38	84.48	84.78	85.08	primary-t.	49.47	49.23	42.63	46.02
credit-a	86.01	84.87	86.09	86.04	segment	98.03	97.05	98.08	98.14
credit-g	75.64	75.48	76.17	75.23	sonar	85.05	85.05	85.58	84.23
diabetes	75.53	76.86	76.32	76.25	soybean	93.91	93.69	92.00	93.84
glass	74.35	74.44	76.45	75.70	vehicle	74.46	73.90	79.89	72.91
heart-c	82.74	84.09	84.26	81.55	vote	95.93	95.95	96.32	95.33
heart-h	83.64	85.78	85.14	83.16	vowel	98.74	99.06	99.00	98.80
heart-statlog	84.22	83.56	84.04	83.30	zoo	96.44	95.05	93.96	97.23

- Können wir sagen das Algorithmus X besser als Algorithmus Y ist?

Zusammenfassen von experimentalen Ergebnissen

Dataset	Grading	Select	Stacking	Voting
Avg	85.04	84.59	84.68	84.88

	Grading	Select	Stacking	Voting
Grading	—	15/1/10	11/0/15	12/0/14
Select	10/1/15	—	10/0/16	14/0/12
Stacking	15/0/11	16/0/10	—	15/1/10
Voting	14/0/12	12/0/14	10/1/15	—

- ▶ Die **durschnittliche Performanz** kann trügerisch sein.
 - ▶ A 0.1% besser auf 19 Datensäzen mit tausenden Beispielen
 - ▶ B is 2% besser auf einem Datensatz, der auf 50 Beispielen besteht
 - ▶ A ist besser, aber B hat die höhere durschnittliche Performanz
 - ▶ In unserem Beispiel ist Grading der beste Algorithmus
- ▶ Zähle Siege/Unentschieden/Niederlage
 - ▶ Hier scheint Stacking am Besten abzuschneiden
 - ▶ Aber wie häufig muss man gewinnen?

Ein statistischer Test kann helfen: Vorzeichen-Test

Wir stellen uns vor, dass wir eine Münze werfen

- ▶ Frage: Wie häufig müssen wir Kopf beobachten, bis wir uns sicher sind, dass die Münze fair ist?
- ▶ **Null-Hypothese:** Die Münze ist fair, d.h. $P(kopf) = P(zahl)$
- ▶ Wir werfen die Münze N mal, und bekommen i -mal Kopf und $N - i$ -mal Zahl.
- ▶ Wie wahrscheinlich ist es, dass wir i -mal Kopf beobachten unter der Annahme, dass die Null-Hypothese gilt?

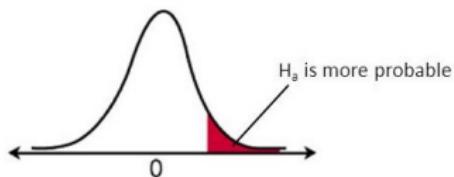
Analogie:

- ▶ Kopf und Zahl = Algorithmen A und B
- ▶ Münzwürfe=Datensätze
- ▶ Null-Hypothese = beide Algorithmen sind gleich
- ▶ i -mal gewinnt A, $N - i$ -mal B

Wir haben eine Binomialverteilung mit $p = 0.5$

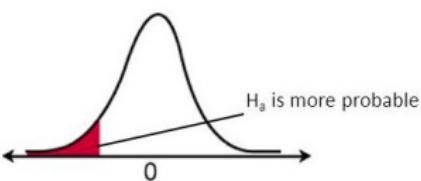
- ▶ Wahrscheinlichkeit für i -fachen Erfolg: $P(i) = \binom{N}{i} p^i (1-p)^{N-i}$
- ▶ **one-tailed test:** Wahrscheinlichkeit für höchsten k -mal Erfolg:
$$P(i \leq k) = \sum_{i=1}^k \binom{N}{i} \frac{1}{2^i} \frac{1}{2^{N-i}} = \frac{1}{2^N} \sum_{i=1}^k \binom{N}{i}$$
- ▶ **two-tailed test:** Wahrscheinlichkeit für höchsten k -fachen **oder** mindestens $N - k$ -fachen Erfolg:
$$P(i \leq k \wedge i \geq N - k) = \frac{1}{2^N} \sum_{i=1}^k \binom{N}{i} + \frac{1}{2^N} \sum_{i=1}^k \binom{N}{N-i} = \frac{1}{2^{N-1}} \sum_{i=1}^k \binom{N}{i}$$
- ▶ Wenn N gross ist, kann man dazu eine Normalverteilung benutzen.

Vorzeichen-Test



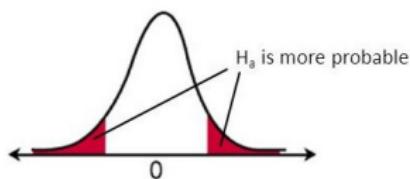
Right-tail test

$$H_a: \mu > \text{value}$$



Left-tail test

$$H_a: \mu < \text{value}$$



Two-tail test

$$H_a: \mu \neq \text{value}$$

Vorzeichen-Test: Ein Beispiel

Table
Sign Test

- Example:
 - 20 datasets
 - Alg. A vs. B
 - A 4 wins
 - B 14 wins
 - 2 ties (not counted)
 - we can say with a certainty of 95% that B is better than A
 - but not with 99% certainty!

Vorzeichentest: Kritische Häufigkeiten i bzw. $N - i$ (s. S. 167)					
N	Irrtumswahrscheinlichkeit		N	Irrtumswahrscheinlichkeit	
	1%	5%		1%	5%
6	—	0	41	11	13
7	—	0	42	12	14
8	0	0	43	12	14
9	0	1	44	13	15
10	0	1	45	13	15
11	0	1	46	13	15
12	1	2	47	14	16
13	1	2	48	14	16
14	1	2	49	15	17
15	2	3	50	15	17
16	2	3	51	15	18
17	2	4	52	16	18
18	3	4	53	16	18
19	3	4	54	17	19
20	3	5	55	17	19
21	4	5	56	17	20
22	4	5	57	18	20
23	4	6	58	18	21
24	5	6	59	19	21
25	5	7	60	19	21
26	6	7	61	20	22
27	6	7	62	20	22
28	6	8	63	20	23
29	7	8	64	21	23
30	7	9	65	21	24
31	7	9	66	22	24
32	8	9	67	22	25
33	8	10	68	22	25
34	9	10	69	23	25
35	9	11	70	23	26
36	9	11	71	24	26
37	10	12	72	24	27

Vorzeichen-Test: Eigenschaften

- ▶ Der Vorzeichen-Test ist ein sehr einfacher Test: er trifft keine Annahme über die zugrundeliegende Verteilung
- ▶ Er ist sehr **konservativ**
 - ▶ Wenn er einen Unterschied erkennt, dann kann man sich sicher sein.
 - ▶ Wenn er aber keinen Unterschied detektiert, dann könnte ein Test, der die Datenverteilung erfasst, immer noch einen signifikanten Unterschied ergeben.
- ▶ Alternativer Test: **two-tailed t-test**
 - ▶ Betrachtet die Grösse des Unterschieds
 - ▶ Nimmt an, dass der Unterschied normal-verteilt ist
- ▶ Faustregel
 - ▶ Vorzeichen-Test beantwortet "Wie häufig?"
 - ▶ t-test beantwortet "Wieviel?"

Übrigens, mit linearen Modellen kann man auch klassifizieren: Logistische Regression

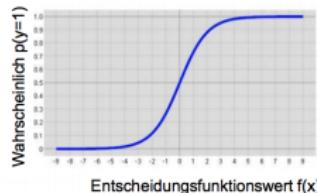


Man postuliert, dass der Quotient der Klassenwahrscheinlichkeiten (ähnlich wie bei unserem Vergleich von Modellen) als log-lineares Modell berechnet werden kann:
 $\log\left(\frac{P(Y_i=1)}{1-P(Y_i=1)}\right) = f_w(x_i)$

- Für eine binäre Klassifikation ($y = +1$ oder -1) wird oft eine kontinuierliche Entscheidungsfunktion $f(x)$ gelernt.
 - ◆ Z.B. lineares Modell

$$f_w(\mathbf{x}) = \mathbf{w}^T \mathbf{x} = \sum_{i=1}^m w_i x_i$$

- Je größer $f(x)$, desto wahrscheinlicher ist, dass x zur Klasse **+1** gehört
 - ◆ Z.B. logistische Regression



$$\sigma(f(x)) = \frac{1}{1 + \exp(-f(x))}$$

Klassifikator / Entscheidungsfunktion

- Wie bestimmen wir Klassenentscheidung +1/-1 aus $f(\mathbf{x})$?
- Allgemeine Lösung:

$$\text{Vorhersage} = \begin{cases} +1: f(\mathbf{x}) \geq \theta \\ -1: \text{sonst} \end{cases}$$

- Der Wert für θ verschiebt „false positives“ zu „false negatives“.
- Optimaler Wert hängt von Kosten einer positiven oder negativen Fehlklassifikation ab.

Evaluation von Klassifikatoren und Entscheidungsfunktionen

- Fehlklassifikationswahrscheinlichkeit
 - ◆ Häufig nicht aussagekräftig, weil $P(+1)$ sehr klein.
 - ◆ Wie gut sind 5% Fehler, wenn $P(+1)=3\%$?
 - ◆ Idee: Nicht Klassifikator bewerten, sondern Entscheidungsfunktion.
- Receiver Operating Characteristic (ROC-Kurve)
 - ◆ Bewertet Entscheidungsfunktion,
 - ◆ Jeder Punkt auf der ROC Kurve entspricht einem Schwellwert θ
 - ◆ Fläche unter ROC-Kurve = $P(\text{positives Beispiel hat höheren f-Wert als negatives Beispiel})$

ROC-Analyse

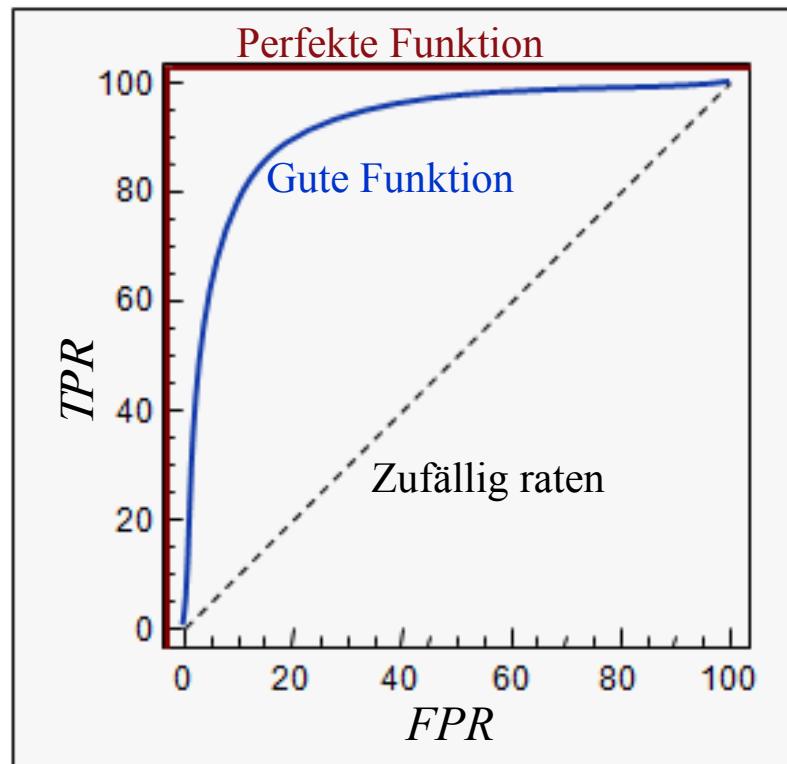
- Entscheidungsfunktion + Schwellwert = Klassifikator

$$\text{Vorhersage} = \begin{cases} +1 : f(\mathbf{x}) \geq \theta \\ -1 : \text{sonst} \end{cases}$$

- ◆ Fehler hängen vom Schwellwert ab
 - ◆ Großer Schwellwert: Mehr positive Bsp falsch.
 - ◆ Kleiner Schwellwert: Mehr negative Bsp falsch.
-
- ROC-Analyse: Bewertung der Entscheidungsfunktion unabhängig vom konkreten Schwellwert.
 - Charakterisieren das Verhalten des Klassifikators für alle möglichen Schwellwerte.

ROC-Kurven

- Rate der „False Positives“ und „True Positives“ in Abhängigkeit des Schwellwertes
 - ◆ X-Achse: „False Positive Rate“
 - ◆ Y-Achse: „True Positive Rate“



	Vorhersage „+“	Vorhersage „-“
Echtes Label „+“	TP	FN
Echtes Label „-“	FP	TN

$$FPR = \frac{FP}{N}$$

$$TPR = \frac{TP}{P}$$

$$N = FP + TN$$

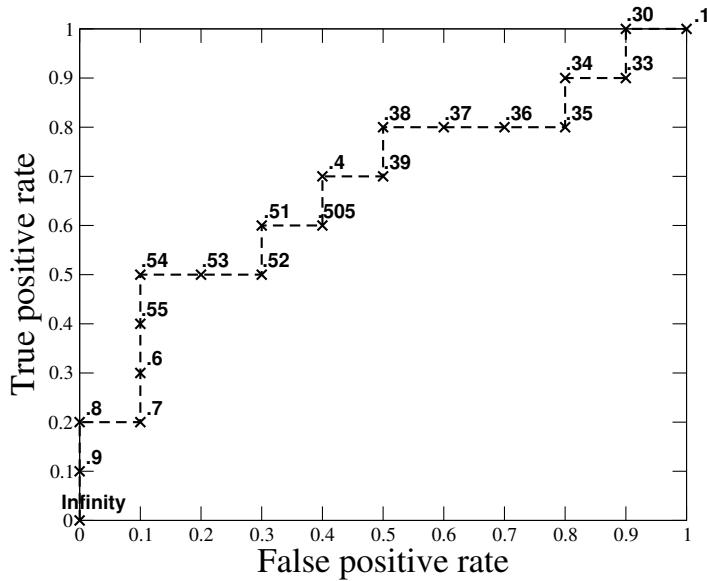
$$P = TP + FN$$

Bestimmen der ROC-Kurve von f

Man könnte einfach verschiedene Schwellwerte ausprobieren. Das ist aber ineffizient.
Effiziente ist es, die Scores von Datenpunkten als Schwellwerte zu nehmen.

- Annahme: kein $f(\mathbf{x}) = f(\mathbf{x}')$ für $\mathbf{x} \neq \mathbf{x}'$.
- Generiere Liste L aller Instanzen \mathbf{x} , absteigend sortiert nach $f(\mathbf{x})$
- P = Anzahl positiver Instanzen, N = Anzahl negativer Instanzen
- TP = FP = 0 Idee: Jeder Datenpunkt, der für einen bestimmten Schwellenwert positiv klassifiziert wird, wird auch für alle niedrigeren Schwellenwerte positiv klassifiziert. Daher sortiert man die Testinstanzen absteigend nach den Scores
- Für $i = 1$ bis $\text{Länge}(L)$
 - ◆ $\mathbf{x} = i$ -tes Element von L
 - ◆ Wenn \mathbf{x} positive Instanz: increment(TP)
 - ◆ Wenn \mathbf{x} negative Instanz: increment(FP)
 - ◆ Zeichne neuen Punkt mit Koordination $(FP/N, TP/P)$

Lese <http://binf.gmu.edu/mmasso/ROC101.pdf>



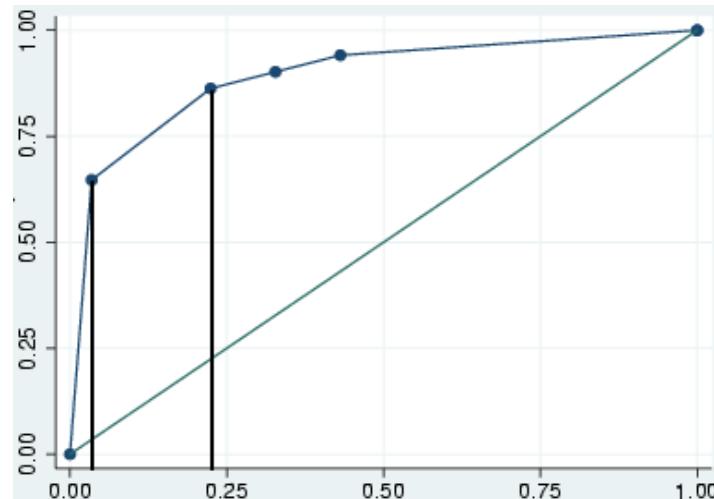
Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1

Figure 3. The ROC “curve” created by thresholding a test set. The table at right shows twenty data and the score assigned to each by a scoring classifier. The graph at left shows the corresponding ROC curve with each point labeled by the threshold that produces it.

figure 3 is taken from a very small instance set so that each point’s derivation can be understood. In the table of figure 3, the instances are sorted by their scores, and each point in the ROC graph is labeled by the score threshold that produces it. A threshold of $+\infty$ produces the point $(0, 0)$. As we lower the threshold to 0.9 the first positive instance is classified positive, yielding $(0, 0.1)$. As the threshold is further reduced, the curve climbs up and to the right, ending up at $(1, 1)$ with a threshold

Flächeninhalt der ROC-Kurve

- Flächeninhalt AUC kann durch Integrieren (Summieren der Trapez-Flächeninhalte) bestimmt werden.

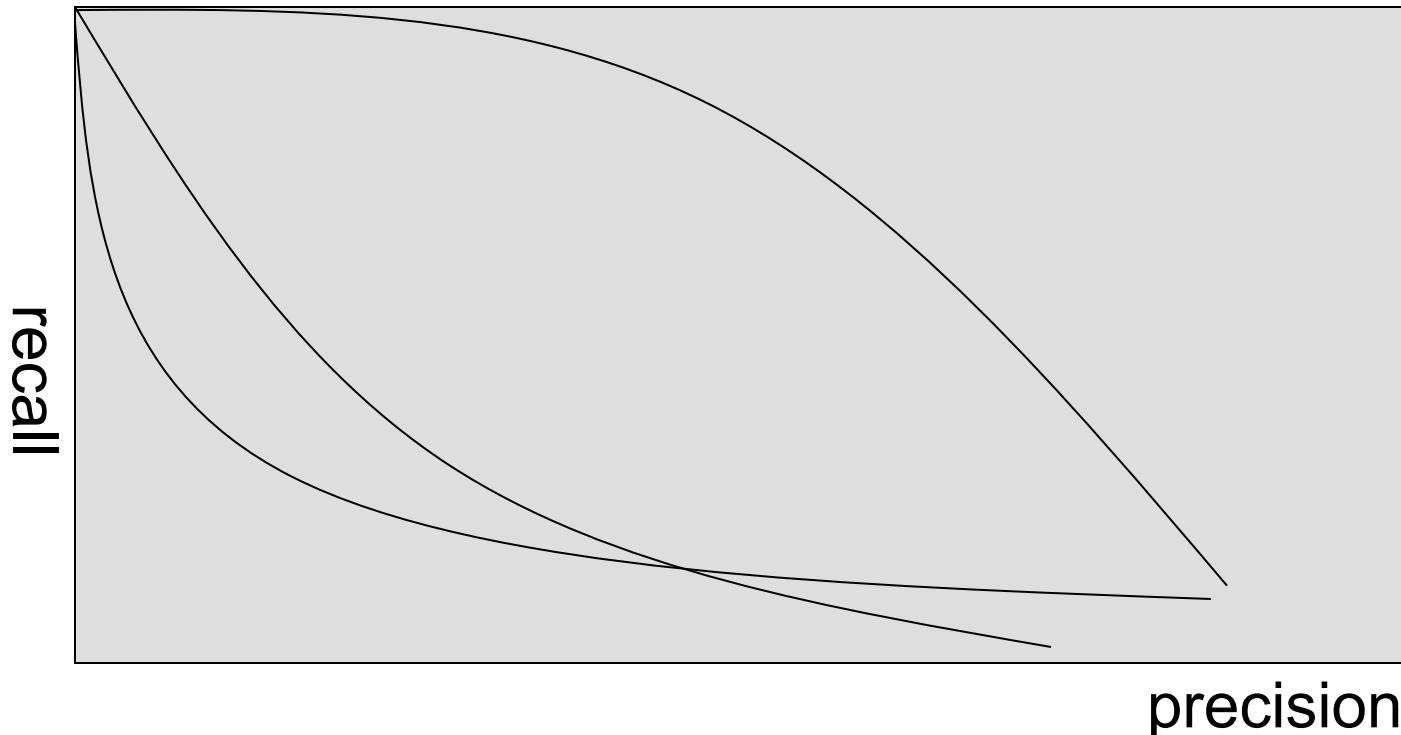


- x_+ = zufällig gezogenes Positivbeispiel
- x_- = zufällig gezogenes Negativbeispiel
- Theorem: $AUC = P(f(x_+) > f(x_-))$.

Precision / Recall

- Alternative zur ROC-Analyse.
- Stammt aus dem Information Retrieval.
- $\text{Precision} = \frac{TP}{TP + FP} \leftarrow \text{Alle Instanzen mit Vorhersage „+“}$
- $\text{Recall} = \frac{TP}{TP + FN} \leftarrow \text{Alle Instanzen mit echtem Label „+“}$
- Precision: $P(\text{positiv} \mid \text{positiv vorhergesagt})$
- Recall: $P(\text{positiv vorhergesagt} \mid \text{ist positiv})$

Precision / Recall Trade-Off



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

F-Measure, Breakeven Point

- Zusammenfassungen der Kurve in einer Zahl:
 - ◆ F-Measure: Harmonisches Mittel über Precision und Recall, maximiert über Schwellwert θ

$$\text{F-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- ◆ Precision-Recall-Breakeven-Point: Es gibt einen Punkt θ auf der Kurve für den gilt $\text{Precision}(\theta) = \text{Recall}(\theta) =: \text{PRBEP}$