

Vorlesung Maschinelles Lernen

Klassifikation und Regression: nächste Nachbarn

Katharina Morik

LS 8 Künstliche Intelligenz Fakultät für Informatik
Technische Universität Dortmund

22.10.2013

Gliederung

- 1 kNN zur Klassifikation, Regression
 - Bias und Varianz bei kNN
- 2 kNN implementieren
 - Ähnlichkeitsmaße
- 3 Funktionsapproximation
 - Likelihood
- 4 Modellselektion
 - Kreuzvalidierung zur Modellselektion
 - Bayes Kriterien zur Modellselektion

Globale und lokale Modelle

- **Lineare Modelle finden eine trennende Hyperebene.**
- Die Ebene trennt alle Beispiele auf einmal. Es ist ein **globales Modell**.
- Klassifiziert man ein Beispiel nur anhand der Beispiele seiner **Umgebung**, spricht man von einem **lokalen Modell**.
- Nächste Nachbarn sind ein lokales Modell.

Globale und lokale Modelle

- Lineare Modelle finden eine trennende Hyperebene.
- Die Ebene trennt alle Beispiele auf einmal. Es ist ein **globales Modell**.
- Klassifiziert man ein Beispiel nur anhand der Beispiele seiner **Umgebung**, spricht man von einem **lokalen Modell**.
- Nächste Nachbarn sind ein lokales Modell.

Globale und lokale Modelle

- Lineare Modelle finden eine trennende Hyperebene.
- Die Ebene trennt alle Beispiele auf einmal. Es ist ein **globales Modell**.
- Klassifiziert man ein Beispiel nur anhand der Beispiele seiner **Umgebung**, spricht man von einem **lokalen Modell**.
- Nächste Nachbarn sind ein lokales Modell.

Globale und lokale Modelle

- Lineare Modelle finden eine trennende Hyperebene.
- Die Ebene trennt alle Beispiele auf einmal. Es ist ein **globales Modell**.
- Klassifiziert man ein Beispiel nur anhand der Beispiele seiner **Umgebung**, spricht man von einem **lokalen Modell**.
- Nächste Nachbarn sind ein lokales Modell.

Nächste Nachbarn

- Das k NN-Modell betrachtet nur die k nächsten Nachbarn eines Beispiel $\vec{x}$:

$$\hat{f}(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i \quad (1)$$

- Die Nachbarschaft $N_k(\vec{x})$ wird durch ein Abstandsmaß, z.B. den Euklidischen Abstand bestimmt.
- Es gibt maximal $\frac{N}{k}$ Nachbarschaften und in jeder bestimmen wir den Durchschnitt (1).

Nächste Nachbarn

- Das k NN-Modell betrachtet nur die k nächsten Nachbarn eines Beispiel $\vec{x}$:

$$\hat{f}(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i \quad (1)$$

- Die Nachbarschaft $N_k(\vec{x})$ wird durch ein Abstandsmaß, z.B. den Euklidischen Abstand bestimmt.
- Es gibt maximal $\frac{N}{k}$ Nachbarschaften und in jeder bestimmen wir den Durchschnitt (1).

Nächste Nachbarn

- Das k NN-Modell betrachtet nur die k nächsten Nachbarn eines Beispiel $\vec{x}$:

$$\hat{f}(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i \quad (1)$$

- Die Nachbarschaft $N_k(\vec{x})$ wird durch ein Abstandsmaß, z.B. den Euklidischen Abstand bestimmt.
- Es gibt maximal $\frac{N}{k}$ Nachbarschaften und in jeder bestimmen wir den Durchschnitt (1).

Regression und Klassifikation

Regression $f : X \rightarrow Y$ mit $y \in \mathcal{R}$
Gleichung (1) gibt als Regressionsfunktion den Mittelwert der y_i zurück.

$$\hat{f}(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i$$

Klassifikation $f : X \rightarrow Y$ mit $y \in \{+1, -1\}$
Durch einen Schwellwert können wir aus der Regression eine Klassifikation machen:

$$\hat{y} = \begin{cases} 1, & \text{falls } \hat{f}(\vec{x}) \geq 0,5 \\ 0, & \text{sonst} \end{cases}$$

Regression und Klassifikation

Regression $f : X \rightarrow Y$ mit $y \in \mathcal{R}$
Gleichung (1) gibt als Regressionsfunktion den Mittelwert der y_i zurück.

$$\hat{f}(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i$$

Klassifikation $f : X \rightarrow Y$ mit $y \in \{+1, -1\}$
Durch einen Schwellwert können wir aus der Regression eine Klassifikation machen:

$$\hat{y} = \begin{cases} 1, & \text{falls } \hat{f}(\vec{x}) \geq 0,5 \\ 0, & \text{sonst} \end{cases}$$

Die grünen und roten Datenpunkte werden in Nachbarschaften gruppiert

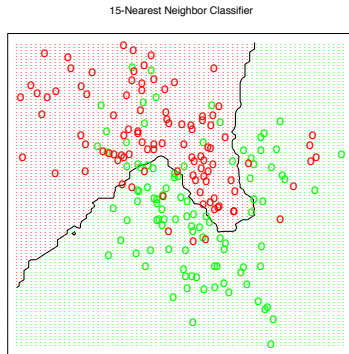


Figure 2.2: The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (GREEN = 0, RED = 1) and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.

Bei $k=1$ wird nur auswendig gelernt.

- Falls $\vec{x} = \vec{x}' \rightarrow y = y'$, gibt es bei $k = 1$ keinen Trainingsfehler.
- Wenn allein der Trainingsfehler das Optimierungskriterium ist, würden wir stets $k = 1$ nehmen und nur auswendig lernen.
- Vermutlich ergibt das auf den Testdaten einen großen Fehler!

Bei $k=1$ wird nur auswendig gelernt.

- Falls $\vec{x} = \vec{x}' \rightarrow y = y'$, gibt es bei $k = 1$ keinen Trainingsfehler.
- Wenn allein der Trainingsfehler das Optimierungskriterium ist, würden wir stets $k = 1$ nehmen und nur auswendig lernen.
- Vermutlich ergibt das auf den Testdaten einen großen Fehler!

Bei $k=1$ wird nur auswendig gelernt.

- Falls $\vec{x} = \vec{x}' \rightarrow y = y'$, gibt es bei $k = 1$ keinen Trainingsfehler.
- Wenn allein der Trainingsfehler das Optimierungskriterium ist, würden wir stets $k = 1$ nehmen und nur auswendig lernen.
- Vermutlich ergibt das auf den Testdaten einen großen Fehler!

Overfitting

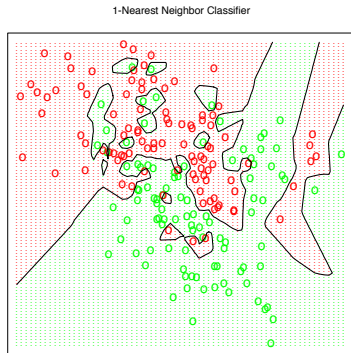


Figure 2.3: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (GREEN = 0, RED = 1), and then predicted by 1-nearest-neighbor classification.*

Training- und Testfehler bei verschiedenen k

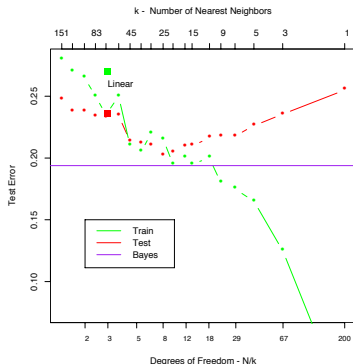


Figure 2.4: *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size 200 was used, and a test sample of size 10,000. The red curves are test and the green are training error for k -nearest-neighbor classification. The results for linear regression are the bigger green and red dots at three degrees of freedom. The purple*

Erwartungswert von Y bei k Nächsten Nachbarn

- Der Erwartungswert von Y , $E(Y) = \sum_{i=1}^N y_i p_i$, geht bei up in den Fehler ein: $EPE(\hat{f}) = E(Y - \hat{f}(X))^2$.
- k NN verwendet den Erwartungswert von Y direkt zur Vorhersage, allerdings beschränkt auf die Nachbarschaft
$$E(Y) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i.$$
- Für die Vorhersage sind wir an bedingten Wahrscheinlichkeiten interessiert $P(Y|X = \vec{x})$.
- Bei k NN wird die bedingte Wahrscheinlichkeit auf die Nachbarschaft begrenzt $E(Y|\vec{x}_i \in N_k(\vec{x}))$.
- Gerechnet wird dies mit Hilfe Gleichung (1).

Erwartungswert von Y bei k Nächsten Nachbarn

- Der Erwartungswert von Y , $E(Y) = \sum_{i=1}^N y_i p_i$, geht bei up in den Fehler ein: $EPE(\hat{f}) = E(Y - \hat{f}(X))^2$.
- kNN verwendet den Erwartungswert von Y direkt zur Vorhersage, allerdings beschränkt auf die Nachbarschaft
$$E(Y) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i.$$
- Für die Vorhersage sind wir an bedingten Wahrscheinlichkeiten interessiert $P(Y|X = \vec{x})$.
- Bei kNN wird die bedingte Wahrscheinlichkeit auf die Nachbarschaft begrenzt $E(Y|\vec{x}_i \in N_k(\vec{x}))$.
- Gerechnet wird dies mit Hilfe Gleichung (1).

Erwartungswert von Y bei k Nächsten Nachbarn

- Der Erwartungswert von Y , $E(Y) = \sum_{i=1}^N y_i p_i$, geht bei up in den Fehler ein: $EPE(\hat{f}) = E(Y - \hat{f}(X))^2$.
- kNN verwendet den Erwartungswert von Y direkt zur Vorhersage, allerdings beschränkt auf die Nachbarschaft
$$E(Y) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i.$$
- Für die Vorhersage sind wir an bedingten Wahrscheinlichkeiten interessiert $P(Y|X = \vec{x})$.
- Bei kNN wird die bedingte Wahrscheinlichkeit auf die Nachbarschaft begrenzt $E(Y|\vec{x}_i \in N_k(\vec{x}))$.
- Gerechnet wird dies mit Hilfe Gleichung (1).

Erwartungswert von Y bei k Nächsten Nachbarn

- Der Erwartungswert von Y , $E(Y) = \sum_{i=1}^N y_i p_i$, geht bei up in den Fehler ein: $EPE(\hat{f}) = E(Y - \hat{f}(X))^2$.
- k NN verwendet den Erwartungswert von Y direkt zur Vorhersage, allerdings beschränkt auf die Nachbarschaft
$$E(Y) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i.$$
- Für die Vorhersage sind wir an bedingten Wahrscheinlichkeiten interessiert $P(Y|X = \vec{x})$.
- Bei k NN wird die bedingte Wahrscheinlichkeit auf die Nachbarschaft begrenzt $E(Y|\vec{x}_i \in N_k(\vec{x}))$.
- Gerechnet wird dies mit Hilfe Gleichung (1).

Erwartungswert von Y bei k Nächsten Nachbarn

- Der Erwartungswert von Y , $E(Y) = \sum_{i=1}^N y_i p_i$, geht bei up in den Fehler ein: $EPE(\hat{f}) = E(Y - \hat{f}(X))^2$.
- k NN verwendet den Erwartungswert von Y direkt zur Vorhersage, allerdings beschränkt auf die Nachbarschaft
$$E(Y) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i.$$
- Für die Vorhersage sind wir an bedingten Wahrscheinlichkeiten interessiert $P(Y|X = \vec{x})$.
- Bei k NN wird die bedingte Wahrscheinlichkeit auf die Nachbarschaft begrenzt $E(Y|\vec{x}_i \in N_k(\vec{x}))$.
- Gerechnet wird dies mit Hilfe Gleichung (1).

Asymptotisches Ergebnis zu kNN

- Wenn k/N gegen 0 und N, k gegen ∞ konvergieren, konvergiert auch $\hat{f}(x)$ gegen $E(Y|X = x)$. (Hastie/etal/2001, S. 19)
- Haben wir also den perfekten Lernalgorithmus gefunden?

Asymptotisches Ergebnis zu kNN

- Wenn k/N gegen 0 und N, k gegen ∞ konvergieren, konvergiert auch $\hat{f}(x)$ gegen $E(Y|X = x)$. (Hastie/etal/2001, S. 19)
- Haben wir also den perfekten Lernalgorithmus gefunden?

Fluch der hohen Dimension bei kNN

- Die Dichte der Beispiele ist proportional zu $N^{\frac{1}{p}}$.
- Schon bei $p = 10$ brauchen wir 80% der möglichen Werte jedes Attributs X_i , um wenigstens 10% der Daten in einer Nachbarschaft gesehen zu haben!
- Die Dichte der Datenpunkte in der Nachbarschaft ist bei hoher Dimension furchtbar spärlich.
 - $N^{\frac{1}{p}}$ ist bei 100 Beispielen und $p = 10$ nur $100^{1/10} = \sqrt[10]{100}$.
 - Wenn 100 Beispiele bei $p = 1$ einen dichten Raum ergeben, muss man für die selbe Dichte bei $p = 10$ schon 100^{10} Beispiele sammeln: $100^{1/1} = 100, 100^{10 \cdot \frac{1}{10}} = 100$

Fluch der hohen Dimension bei kNN

- Die Dichte der Beispiele ist proportional zu $N^{\frac{1}{p}}$.
- Schon bei $p = 10$ brauchen wir 80% der möglichen Werte jedes Attributs X_i , um wenigstens 10% der Daten in einer Nachbarschaft gesehen zu haben!
- Die Dichte der Datenpunkte in der Nachbarschaft ist bei hoher Dimension furchtbar spärlich.
 - $N^{\frac{1}{p}}$ ist bei 100 Beispielen und $p = 10$ nur $100^{1/10} = \sqrt[10]{100}$.
 - Wenn 100 Beispiele bei $p = 1$ einen dichten Raum ergeben, muss man für die selbe Dichte bei $p = 10$ schon 100^{10} Beispiele sammeln: $100^{1/1} = 100, 100^{10 \cdot \frac{1}{10}} = 100$

Fluch der hohen Dimension bei kNN

- Die Dichte der Beispiele ist proportional zu $N^{\frac{1}{p}}$.
- Schon bei $p = 10$ brauchen wir 80% der möglichen Werte jedes Attributs X_i , um wenigstens 10% der Daten in einer Nachbarschaft gesehen zu haben!
- Die Dichte der Datenpunkte in der Nachbarschaft ist bei hoher Dimension furchtbar spärlich.
 - $N^{\frac{1}{p}}$ ist bei 100 Beispielen und $p = 10$ nur $100^{1/10} = \sqrt[5]{10}$.
 - Wenn 100 Beispiele bei $p = 1$ einen dichten Raum ergeben, muss man für die selbe Dichte bei $p = 10$ schon 100^{10} Beispiele sammeln: $100^{1/1} = 100$, $100^{10 \cdot \frac{1}{10}} = 100$

Fluch der hohen Dimension bei kNN

- Die Dichte der Beispiele ist proportional zu $N^{\frac{1}{p}}$.
- Schon bei $p = 10$ brauchen wir 80% der möglichen Werte jedes Attributs X_i , um wenigstens 10% der Daten in einer Nachbarschaft gesehen zu haben!
- Die Dichte der Datenpunkte in der Nachbarschaft ist bei hoher Dimension furchtbar spärlich.
 - $N^{\frac{1}{p}}$ ist bei 100 Beispielen und $p = 10$ nur $100^{1/10} = \sqrt[5]{10}$.
 - Wenn 100 Beispiele bei $p = 1$ einen dichten Raum ergeben, muss man für die selbe Dichte bei $p = 10$ schon 100^{10} Beispiele sammeln: $100^{1/1} = 100, 100^{10 \cdot \frac{1}{10}} = 100$

Fluch der hohen Dimension bei kNN

- Die Dichte der Beispiele ist proportional zu $N^{\frac{1}{p}}$.
- Schon bei $p = 10$ brauchen wir 80% der möglichen Werte jedes Attributs X_i , um wenigstens 10% der Daten in einer Nachbarschaft gesehen zu haben!
- Die Dichte der Datenpunkte in der Nachbarschaft ist bei hoher Dimension furchtbar spärlich.
 - $N^{\frac{1}{p}}$ ist bei 100 Beispielen und $p = 10$ nur $100^{1/10} = \sqrt[5]{10}$.
 - Wenn 100 Beispiele bei $p = 1$ einen dichten Raum ergeben, muss man für die selbe Dichte bei $p = 10$ schon 100^{10} Beispiele sammeln: $100^{1/1} = 100$, $100^{10 \cdot \frac{1}{10}} = 100$

Bias und Varianz bei k NN

- Wenn man die richtige, dicht besetzte Nachbarschaft hat, verzerrt k NN die Vorhersage nicht (**kleiner Bias**).
- Wenn - wie bei hohen Dimensionen - die Nachbarschaft wild variiert, schwankt auch die Güte der Vorhersage (**große Varianz**).

Bias und Varianz bei k NN

- Wenn man die richtige, dicht besetzte Nachbarschaft hat, verzerrt k NN die Vorhersage nicht (**kleiner Bias**).
- Wenn - wie bei hohen Dimensionen - die Nachbarschaft wild variiert, schwankt auch die Güte der Vorhersage (**große Varianz**).

Bias und Varianz – bildlich

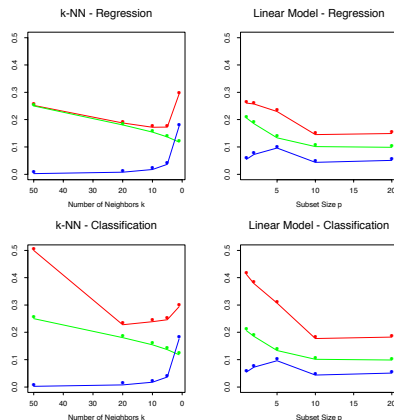


Figure 7.3: Prediction error (red), squared bias (green) and variance (blue) for a simulated example. The top row is regression with squared error loss; the bottom row is classification with 0–1 loss. The models are k -nearest neighbors (left) and best subset regression of size p (right). The variance and bias curves are the same in regression and classification, but the prediction error curve is different.

Bias, Varianz und Modellkomplexität – bildlich

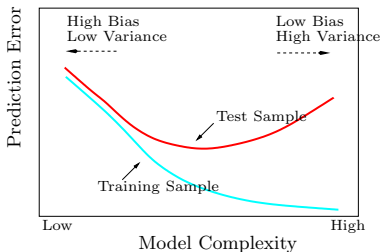


Figure 7.1: *Behavior of test sample and training sample error as the model complexity is varied.*

- Dieser Zusammenhang von Training-/Testfehler und Komplexität bestimmt alle Lernverfahren.
- Kreuzvalidierung lässt abschätzen, wie gut das Modell zu den Daten passt (Modellselektion).

Bias, Varianz und Modellkomplexität – bildlich

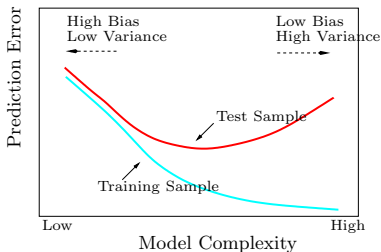


Figure 7.1: *Behavior of test sample and training sample error as the model complexity is varied.*

- Dieser Zusammenhang von Training-/Testfehler und Komplexität bestimmt alle Lernverfahren.
- Kreuzvalidierung lässt abschätzen, wie gut das Modell zu den Daten passt (Modellselektion).

Was wissen Sie jetzt?

- Sie kennen den Fluch der hohen Dimension bei kNN: kleiner Bias, aber hohe Varianz.
- Bei linearen Modellen ist es umgekehrt: kleine Varianz, aber hoher Bias (falls die Annahme des linearen Zusammenhangs von X , Y nicht stimmt).

Was wissen Sie jetzt?

- Sie kennen den Fluch der hohen Dimension bei kNN: kleiner Bias, aber hohe Varianz.
- Bei linearen Modellen ist es umgekehrt: kleine Varianz, aber hoher Bias (falls die Annahme des linearen Zusammenhangs von X , Y nicht stimmt).

Rahmen – Instanzbasiertes Lernen

- Alle Beispiele werden gespeichert.
 - Geschickt indexieren?
 - Typische Beispiele auswählen?
- Zu einer neuen Beobachtung $\vec{x}$ werden die k ähnlichsten Beispiele $\vec{x}'$ gefunden: $N_k(\vec{x})$.
 - Ähnlichkeitsmaß $Sim(\vec{x}, \vec{x}')$?
- und daraus gemäß einer Entscheidungsfunktion der y -Wert ermittelt.
 - Maximum, Mehrheit, Mittelwert?

Rahmen – Instanzbasiertes Lernen

- Alle Beispiele werden gespeichert.
 - Geschickt indexieren?
 - Typische Beispiele auswählen?
- Zu einer neuen Beobachtung $\vec{x}$ werden die k ähnlichsten Beispiele $\vec{x}'$ gefunden: $N_k(\vec{x})$.
 - Ähnlichkeitsmaß $\text{Sim}(\vec{x}, \vec{x}')$?
- und daraus gemäß einer Entscheidungsfunktion der y -Wert ermittelt.
 - Maximum, Mehrheit, Mittelwert?

Rahmen – Instanzbasiertes Lernen

- Alle Beispiele werden gespeichert.
 - Geschickt indexieren?
 - Typische Beispiele auswählen?
- Zu einer neuen Beobachtung $\vec{x}$ werden die k ähnlichsten Beispiele $\vec{x}'$ gefunden: $N_k(\vec{x})$.
 - Ähnlichkeitsmaß $\text{Sim}(\vec{x}, \vec{x}')$?
- und daraus gemäß einer Entscheidungsfunktion der y -Wert ermittelt.
 - Maximum, Mehrheit, Mittelwert?

Rahmen – Instanzbasiertes Lernen

- Alle Beispiele werden gespeichert.
 - Geschickt indexieren?
 - Typische Beispiele auswählen?
- Zu einer neuen Beobachtung $\vec{x}$ werden die k ähnlichsten Beispiele $\vec{x}'$ gefunden: $N_k(\vec{x})$.
 - Ähnlichkeitsmaß $Sim(\vec{x}, \vec{x}')$?
- und daraus gemäß einer Entscheidungsfunktion der y -Wert ermittelt.
 - Maximum, Mehrheit, Mittelwert?

Rahmen – Instanzbasiertes Lernen

- Alle Beispiele werden gespeichert.
 - Geschickt indexieren?
 - Typische Beispiele auswählen?
- Zu einer neuen Beobachtung $\vec{x}$ werden die k ähnlichsten Beispiele $\vec{x}'$ gefunden: $N_k(\vec{x})$.
 - Ähnlichkeitsmaß $Sim(\vec{x}, \vec{x}')$?
- und daraus gemäß einer Entscheidungsfunktion der y -Wert ermittelt.
 - Maximum, Mehrheit, Mittelwert?

Rahmen – Instanzbasiertes Lernen

- Alle Beispiele werden gespeichert.
 - Geschickt indexieren?
 - Typische Beispiele auswählen?
- Zu einer neuen Beobachtung $\vec{x}$ werden die k ähnlichsten Beispiele $\vec{x}'$ gefunden: $N_k(\vec{x})$.
 - Ähnlichkeitsmaß $Sim(\vec{x}, \vec{x}')$?
- und daraus gemäß einer Entscheidungsfunktion der y -Wert ermittelt.
 - Maximum, Mehrheit, Mittelwert?

Rahmen – Instanzbasiertes Lernen

- Alle Beispiele werden gespeichert.
 - Geschickt indexieren?
 - Typische Beispiele auswählen?
- Zu einer neuen Beobachtung $\vec{x}$ werden die k ähnlichsten Beispiele $\vec{x}'$ gefunden: $N_k(\vec{x})$.
 - Ähnlichkeitsmaß $Sim(\vec{x}, \vec{x}')$?
- und daraus gemäß einer Entscheidungsfunktion der y -Wert ermittelt.
 - Maximum, Mehrheit, Mittelwert?

Mögliche Datenstrukturen

- Alle Beispiele in ein *double[] values* mit allen Attributwerten und dem Label kopieren, damit man schnell iterieren kann;
- *BoundedPriorityQueue*, die das kleinste Element vorn hat
- Tupel (Distanz zum zu klassifizierenden Beispiel, Beispiel)
- Counter für jedes Label, um die Häufigkeit zu zählen

Mögliche Datenstrukturen

- Alle Beispiele in ein *double[] values* mit allen Attributwerten und dem Label kopieren, damit man schnell iterieren kann;
- *BoundedPriorityQueue*, die das kleinste Element vorn hat
- Tupel (Distanz zum zu klassifizierenden Beispiel, Beispiel)
- Counter für jedes Label, um die Häufigkeit zu zählen

Mögliche Datenstrukturen

- Alle Beispiele in ein *double[] values* mit allen Attributwerten und dem Label kopieren, damit man schnell iterieren kann;
- *BoundedPriorityQueue*, die das kleinste Element vorn hat
- Tupel (Distanz zum zu klassifizierenden Beispiel, Beispiel)
- Counter für jedes Label, um die Häufigkeit zu zählen

Mögliche Datenstrukturen

- Alle Beispiele in ein *double[] values* mit allen Attributwerten und dem Label kopieren, damit man schnell iterieren kann;
- *BoundedPriorityQueue*, die das kleinste Element vorn hat
- Tupel (Distanz zum zu klassifizierenden Beispiel, Beispiel)
- Counter für jedes Label, um die Häufigkeit zu zählen

Entscheidungsfunktionen zur Klassifikation

- Mehrheitsentscheidung (Gleichung (1)):

$$f(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i$$

- Gewichtete Entscheidung nach tatsächlicher Ähnlichkeit
 $w_i = \text{Sim}(\vec{x}, \vec{x}_i)$:

$$f(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} w_i \cdot y_i$$

- Bewahrt ein Schwellwert $\text{Sim}(\vec{x}, \vec{x}_i) \leq \theta \rightarrow \vec{x}_i \notin N_k(\vec{x})$ vor großer Varianz?
- Probieren Sie und vergleichen Sie die Ergebnisse der Kreuzvalidierung!

Entscheidungsfunktionen zur Klassifikation

- Mehrheitsentscheidung (Gleichung (1)):

$$f(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i$$

- Gewichtete Entscheidung nach tatsächlicher Ähnlichkeit
 $w_i = \text{Sim}(\vec{x}, \vec{x}_i)$:

$$f(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} w_i \cdot y_i$$

- Bewahrt ein Schwellwert $\text{Sim}(\vec{x}, \vec{x}_i) \leq \theta \rightarrow \vec{x}_i \notin N_k(\vec{x})$ vor großer Varianz?
- Probieren Sie und vergleichen Sie die Ergebnisse der Kreuzvalidierung!

Entscheidungsfunktionen zur Klassifikation

- Mehrheitsentscheidung (Gleichung (1)):

$$f(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i$$

- Gewichtete Entscheidung nach tatsächlicher Ähnlichkeit
 $w_i = \text{Sim}(\vec{x}, \vec{x}_i)$:

$$f(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} w_i \cdot y_i$$

- Bewahrt ein Schwellwert $\text{Sim}(\vec{x}, \vec{x}_i) \leq \theta \rightarrow \vec{x}_i \notin N_k(\vec{x})$ vor großer Varianz?
- Probieren Sie und vergleichen Sie die Ergebnisse der Kreuzvalidierung!

Entscheidungsfunktionen zur Klassifikation

- Mehrheitsentscheidung (Gleichung (1)):

$$f(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i$$

- Gewichtete Entscheidung nach tatsächlicher Ähnlichkeit
 $w_i = \text{Sim}(\vec{x}, \vec{x}_i)$:

$$f(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} w_i \cdot y_i$$

- Bewahrt ein Schwellwert $\text{Sim}(\vec{x}, \vec{x}_i) \leq \theta \rightarrow \vec{x}_i \notin N_k(\vec{x})$ vor großer Varianz?
- Probieren Sie und vergleichen Sie die Ergebnisse der Kreuzvalidierung!

Ähnlichkeit – Maße

- Ähnlichkeit oder Distanz sollte stets Werte in $[0, 1]$ haben.
- $dist(\vec{x}_1, \vec{x}_2) = 1 - sim(\vec{x}_1, \vec{x}_2)$
- Eine Metrik erfüllt die Bedingungen
 - $Metrik(x, x) = 0$
 - $Metrik(x_1, x_2) = Metrik(x_2, x_1)$
 - $Metrik(x_1, x_3) \leq Metrik(x_1, x_2) + Metrik(x_2, x_3)$

Ähnlichkeit – Maße

- Ähnlichkeit oder Distanz sollte stets Werte in $[0, 1]$ haben.
- $dist(\vec{x}_1, \vec{x}_2) = 1 - sim(\vec{x}_1, \vec{x}_2)$
- Eine Metrik erfüllt die Bedingungen
 - $Metrik(x, x) = 0$
 - $Metrik(x_1, x_2) = Metrik(x_2, x_1)$
 - $Metrik(x_1, x_3) \leq Metrik(x_1, x_2) + Metrik(x_2, x_3)$

Ähnlichkeit – Maße

- Ähnlichkeit oder Distanz sollte stets Werte in $[0, 1]$ haben.
- $dist(\vec{x}_1, \vec{x}_2) = 1 - sim(\vec{x}_1, \vec{x}_2)$
- Eine Metrik erfüllt die Bedingungen
 - 1 $Metrik(x, x) = 0$
 - 2 $Metrik(x_1, x_2) = Metrik(x_2, x_1)$
 - 3 $Metrik(x_1, x_3) \leq Metrik(x_1, x_2) + Metrik(x_2, x_3)$

Ähnlichkeit – Maße

- Ähnlichkeit oder Distanz sollte stets Werte in $[0, 1]$ haben.
- $dist(\vec{x}_1, \vec{x}_2) = 1 - sim(\vec{x}_1, \vec{x}_2)$
- Eine Metrik erfüllt die Bedingungen
 - 1 $Metrik(x, x) = 0$
 - 2 $Metrik(x_1, x_2) = Metrik(x_2, x_1)$
 - 3 $Metrik(x_1, x_3) \leq Metrik(x_1, x_2) + Metrik(x_2, x_3)$

Ähnlichkeit – Maße

- Ähnlichkeit oder Distanz sollte stets Werte in $[0, 1]$ haben.
- $dist(\vec{x}_1, \vec{x}_2) = 1 - sim(\vec{x}_1, \vec{x}_2)$
- Eine Metrik erfüllt die Bedingungen
 - 1 $Metrik(x, x) = 0$
 - 2 $Metrik(x_1, x_2) = Metrik(x_2, x_1)$
 - 3 $Metrik(x_1, x_3) \leq Metrik(x_1, x_2) + Metrik(x_2, x_3)$

Ähnlichkeit – Maße

- Ähnlichkeit oder Distanz sollte stets Werte in $[0, 1]$ haben.
- $dist(\vec{x}_1, \vec{x}_2) = 1 - sim(\vec{x}_1, \vec{x}_2)$
- Eine Metrik erfüllt die Bedingungen
 - 1 $Metrik(x, x) = 0$
 - 2 $Metrik(x_1, x_2) = Metrik(x_2, x_1)$
 - 3 $Metrik(x_1, x_3) \leq Metrik(x_1, x_2) + Metrik(x_2, x_3)$

sim: Ähnlichkeit für einzelne Attribute

Numerische Attribute: Sei $\max_j$ der höchste Wert von X_j und $\min_j$ der niedrigste, sei $x_{i,j}$ der Wert des j -ten Attributs in der i -ten Beobachtung, dann ist z.B.

$$\text{sim}_j(x_{1,j}, x_{2,j}) = 1 - \frac{|x_{1,j} - x_{2,j}|}{\max_j - \min_j}$$

ein Ähnlichkeitsmaß für X_j .

Nominale Attribute: Ganz einfach:

$$\text{sim}_j(x_{1,j}, x_{2,j}) = \begin{cases} 1 & \text{falls } x_{1,j} = x_{2,j} \\ 0 & \text{sonst} \end{cases}$$

sim: Ähnlichkeit für einzelne Attribute

Numerische Attribute: Sei $\max_j$ der höchste Wert von X_j und $\min_j$ der niedrigste, sei $x_{i,j}$ der Wert des j -ten Attributs in der i -ten Beobachtung, dann ist z.B.

$$\text{sim}_j(x_{1,j}, x_{2,j}) = 1 - \frac{|x_{1,j} - x_{2,j}|}{\max_j - \min_j}$$

ein Ähnlichkeitsmaß für X_j .

Nominale Attribute: Ganz einfach:

$$\text{sim}_j(x_{1,j}, x_{2,j}) = \begin{cases} 1 & \text{falls } x_{1,j} = x_{2,j} \\ 0 & \text{sonst} \end{cases}$$

Sim: Ähnlichkeit der Beispiele als Kombination der Attributähnlichkeiten

Im einfachsten Fall mitteln wir die Einzelähnlichkeiten:

$$\text{Sim}(\vec{x}_1, \vec{x}_2) = \frac{1}{p} \sum_{j=1}^p \text{sim}(x_{1,j}, x_{2,j})$$

Vielleicht sind einige Attribute wichtiger als andere?

$$\text{Sim}(\vec{x}_1, \vec{x}_2) = \frac{\sum_{j=1}^p w_j \text{sim}(x_{1,j}, x_{2,j})}{\sum_{j=1}^p w_j}$$

Vielleicht ist der quadratische Abstand besser?

$$\text{Sim}(\vec{x}_1, \vec{x}_2) = 1 - \sum_{j=1}^p w_j (x_{1,j} - x_{2,j})^2$$

Wie bestimmt man w_j ?

Spielräume

Sie sehen, es gibt viele Varianten in einer **Klasse** von Verfahren, die instanzbasiert oder nächste Nachbarn heißt. Das bedeutet für

die Implementierung: Es sollte eine Rahmenklasse geben, der Ähnlichkeits- und Fehlermaße übergeben werden – nicht für jedes Kriterium ein neues Programm!

die Anwendung: Bei jedem neuen Datensatz müssen die Kriterien wohl überlegt werden – Modellselektion!

die Theorie: Wir wollen Eigenschaften ermitteln, die für alle Verfahren einer Klasse gültig sind, z.B. Bias und Varianz.

Spielräume

Sie sehen, es gibt viele Varianten in einer **Klasse** von Verfahren, die instanzbasiert oder nächste Nachbarn heißt. Das bedeutet für

die Implementierung: Es sollte eine Rahmenklasse geben, der Ähnlichkeits- und Fehlermaße übergeben werden – nicht für jedes Kriterium ein neues Programm!

die Anwendung: Bei jedem neuen Datensatz müssen die Kriterien wohl überlegt werden – Modellselektion!

die Theorie: Wir wollen Eigenschaften ermitteln, die für alle Verfahren einer Klasse gültig sind, z.B. Bias und Varianz.

Spielräume

Sie sehen, es gibt viele Varianten in einer **Klasse** von Verfahren, die instanzbasiert oder nächste Nachbarn heißt. Das bedeutet für

die Implementierung: Es sollte eine Rahmenklasse geben, der Ähnlichkeits- und Fehlermaße übergeben werden – nicht für jedes Kriterium ein neues Programm!

die Anwendung: Bei jedem neuen Datensatz müssen die Kriterien wohl überlegt werden – Modellselektion!

die Theorie: Wir wollen Eigenschaften ermitteln, die für alle Verfahren einer Klasse gültig sind, z.B. Bias und Varianz.

Funktionsapproximation

Allgemeine Aussagen über Modelle – Fehler oder Likelihood

- Gegeben sind die Trainingsbeispiele $\mathcal{T}$, gesucht ist eine Funktion

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

- Dabei gibt es Parameter θ , die abzuschätzen sind, bei den linearen Modellen ist dies β .
- Darüber hinaus können die Daten transformiert werden in einen Raum, der für das Lernen besser geeignet ist: $h_k(\vec{x})$.
- Optimiert wird ein Qualitätskriterium, z.B. wird eine Verlustfunktion minimiert oder die Wahrscheinlichkeit maximiert.

Funktionsapproximation

Allgemeine Aussagen über Modelle – Fehler oder Likelihood

- Gegeben sind die Trainingsbeispiele $\mathcal{T}$, gesucht ist eine Funktion

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

- Dabei gibt es Parameter θ , die abzuschätzen sind, bei den linearen Modellen ist dies β .
- Darüber hinaus können die Daten transformiert werden in einen Raum, der für das Lernen besser geeignet ist: $h_k(\vec{x})$.
- Optimiert wird ein Qualitätskriterium, z.B. wird eine Verlustfunktion minimiert oder die Wahrscheinlichkeit maximiert.

Funktionsapproximation

Allgemeine Aussagen über Modelle – Fehler oder Likelihood

- Gegeben sind die Trainingsbeispiele $\mathcal{T}$, gesucht ist eine Funktion

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

- Dabei gibt es Parameter θ , die abzuschätzen sind, bei den linearen Modellen ist dies β .
- Darüber hinaus können die Daten transformiert werden in einen Raum, der für das Lernen besser geeignet ist: $h_k(\vec{x})$.
- Optimiert wird ein Qualitätskriterium, z.B. wird eine Verlustfunktion minimiert oder die Wahrscheinlichkeit maximiert.

Funktionsapproximation

Allgemeine Aussagen über Modelle – Fehler oder Likelihood

- Gegeben sind die Trainingsbeispiele $\mathcal{T}$, gesucht ist eine Funktion

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

- Dabei gibt es Parameter θ , die abzuschätzen sind, bei den linearen Modellen ist dies β .
- Darüber hinaus können die Daten transformiert werden in einen Raum, der für das Lernen besser geeignet ist: $h_k(\vec{x})$.
- Optimiert wird ein Qualitätskriterium, z.B. wird eine Verlustfunktion minimiert oder die Wahrscheinlichkeit maximiert.

Wege der Funktionsapproximation

Verlustfunktion: Fehler minimieren als Abstand zwischen wahrem Wert und Ergebnis der gelernten Funktion, z.B. $RSS(\theta)$ minimieren. Das kommt nächstes Mal.

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

Wege der Funktionsapproximation

Verlustfunktion: Fehler minimieren als Abstand zwischen wahrem Wert und Ergebnis der gelernten Funktion, z.B. $RSS(\theta)$ minimieren. Das kommt nächstes Mal.

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

Maximum Likelihood

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

$$L(\theta) = \sum_{i=1}^N \log Pr_{\theta}(y_i) \quad (2)$$

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

- Wir können dafür eine Verteilung annehmen, da wir die wahre Verteilung nicht kennen.
- Meist ist die Normalverteilung eine gute Annahme.

Maximum Likelihood

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

$$L(\theta) = \sum_{i=1}^N \log Pr_{\theta}(y_i) \quad (2)$$

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

- Wir können dafür eine Verteilung annehmen, da wir die wahre Verteilung nicht kennen.
- Meist ist die Normalverteilung eine gute Annahme.

Normalverteilung $\mathcal{N}(\mu, \sigma)$

normalverteilt

Eine Zufallsvariable X heißt **normalverteilt** mit den Parametern μ, σ , wenn sie die Dichtefunktion

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}((x-\mu)/\sigma)^2} \quad (3)$$

besitzt.

Normalverteilung

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

Normalverteilung $\mathcal{N}(\mu, \sigma)$

normalverteilt

Eine Zufallsvariable X heißt **normalverteilt** mit den Parametern μ, σ , wenn sie die Dichtefunktion

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}((x-\mu)/\sigma)^2} \quad (3)$$

besitzt.

Normalverteilung

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

Log-likelihood bei nominalem Y ist Entropie

Cross-Entropie

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

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

$$L(\theta) = \sum_{i=1}^N \log(p_{y_i, \theta}(\vec{x}_i)) \quad (4)$$

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

Verfahren zur Modellselektion

Wie wählen wir Modelle aus?

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

Verfahren zur Modellselektion

Wie wählen wir Modelle aus?

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

Verfahren zur Modellselektion

Wie wählen wir Modelle aus?

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

Verfahren zur Modellselektion

Wie wählen wir Modelle aus?

- Kreuzvalidierung für verschiedene Modelle – das mit dem geringsten durchschnittlichen Fehler nehmen! (Minimierung der Verlustfunktion jetzt auf der Ebene der Modelle)
- 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

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

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

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

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

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:

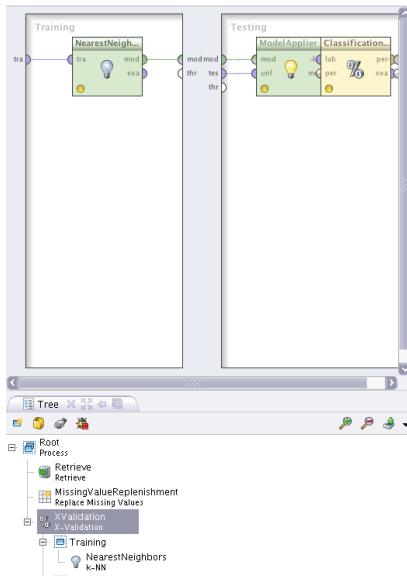
- Lasse die $\kappa(i)$ -te Partition aus,
- lerne das α -te Modell: $\hat{f}^{-\kappa(i)}(\vec{x}, \alpha)$.
- rechne den Fehler aus:

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

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

Modellselektion über Kreuzvalidierung praktisch

In RapidMiner wird die Kreuzvalidierungsschleife schon angeboten.



Bayes Statistik

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

$$Pr(y_j|\vec{x}) = \frac{Pr(y_j)Pr(\vec{x}|y_j)}{Pr(\vec{x})} \quad (5)$$

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

Bayes Modellselektion

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

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

Gegeben dass $Pr(\mathcal{M}_I|\mathcal{T}) \neq 0$, $Pr(\mathcal{T}|\mathcal{M}_I) \neq 0$, $Pr(\mathcal{M}_I) \neq 0$:

Zum Vergleich zweier Modelle $\mathcal{M}_j$, $\mathcal{M}_I$ berechnen wir den Quotienten:

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

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

Approximieren der a posteriori Wahrscheinlichkeit

Wenn alle Modelle a priori gleich wahrscheinlich sind, müssen wir nur $Pr(\mathcal{T}|\mathcal{M}_i)$ approximieren.

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

$$\log Pr(\mathcal{T}|\mathcal{M}_i) = \log Pr(\mathcal{T}|\hat{\theta}_i, \mathcal{M}_i) - \frac{d_i}{2} \cdot \log N + O(1) \quad (6)$$

Approximieren der a posteriori Wahrscheinlichkeit

Wenn alle Modelle a priori gleich wahrscheinlich sind, müssen wir nur $Pr(\mathcal{T}|\mathcal{M}_i)$ approximieren.

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

$$\log Pr(\mathcal{T}|\mathcal{M}_i) = \log Pr(\mathcal{T}|\hat{\theta}_i, \mathcal{M}_i) - \frac{d_i}{2} \cdot \log N + O(1) \quad (6)$$

Approximieren der a posteriori Wahrscheinlichkeit

Wenn alle Modelle a priori gleich wahrscheinlich sind, müssen wir nur $Pr(\mathcal{T}|\mathcal{M}_i)$ approximieren.

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

$$\log Pr(\mathcal{T}|\mathcal{M}_i) = \log Pr(\mathcal{T}|\hat{\theta}_i, \mathcal{M}_i) - \frac{d_i}{2} \cdot \log N + O(1) \quad (6)$$

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

$$BIC = -2 \loglik + (\log N) \cdot d \quad (7)$$

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:

$$\frac{e^{-\frac{1}{2} \cdot BIC_m}}{\sum_{l=1}^M e^{-\frac{1}{2} \cdot BIC_l}} \quad (8)$$

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:

$$\frac{e^{-\frac{1}{2} \cdot BIC_m}}{\sum_{l=1}^M e^{-\frac{1}{2} \cdot BIC_l}} \quad (8)$$

Minimum Description Length

Ein Modell **kodiert** eine Menge von Beispielen. Wir können Nachrichten so kodieren, dass keine Nachricht Präfix einer anderen 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?

Shannon/Weaver Theorem

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

$$length \geq -\sum \Pr(z_i) \log_2(\Pr(z_i)) \quad (9)$$

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!

$$length = -\log Pr(\mathbf{y}|\theta, \mathcal{M}, \mathbf{X}) - \log Pr(\theta|\mathcal{M}) \quad (10)$$

Eigenschaften von MDL

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

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

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

Eigenschaften von MDL

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

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

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

Bezug zwischen MDL und BIC

- Wenn wir die Länge (Gleichung 10) minimieren

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

maximieren wir auch die a posteriori Wahrscheinlichkeit (vgl. Gleichung 5) $Pr(\mathbf{y}|\mathbf{X})$.

- 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.

Bezug zwischen MDL und BIC

- Wenn wir die Länge (Gleichung 10) minimieren

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

maximieren wir auch die a posteriori Wahrscheinlichkeit (vgl. Gleichung 5) $Pr(\mathbf{y}|\mathbf{X})$.

- 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.

Bezug zwischen MDL und BIC

- Wenn wir die Länge (Gleichung 10) minimieren

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

maximieren wir auch die a posteriori Wahrscheinlichkeit (vgl. Gleichung 5) $Pr(\mathbf{y}|\mathbf{X})$.

- 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. MSE
 - Maximierung der Likelihood, z.B. durch Approximation der a posteriori Wahrscheinlichkeit
 - Maximum Likelihood ergibt bei Mehrklassenproblemen die maximale Cross-Entropie (Gleichung 4).
- Für die **Modellselektion** kann man
 - die Kreuzvalidierung mit Fehlerminimierung und
 - die Kriterien nach Bayes (BIC, MDL) nutzen.

Was wissen Sie jetzt?

- Funktionsapproximation optimiert eine **Qualitätsfunktion**.
 - **Fehlerminimierung**, z.B. MSE
 - **Maximierung der Likelihood**, z.B. durch Approximation der a posteriori Wahrscheinlichkeit
 - Maximum Likelihood ergibt bei Mehrklassenproblemen die maximale Cross-Entropie (Gleichung 4).
- Für die **Modellselektion** kann man
 - die Kreuzvalidierung mit Fehlerminimierung und
 - die Kriterien nach Bayes (BIC, MDL) nutzen.

Was wissen Sie jetzt?

- Funktionsapproximation optimiert eine **Qualitätsfunktion**.
 - **Fehlerminimierung**, z.B. MSE
 - **Maximierung der Likelihood**, z.B. durch Approximation der a posteriori Wahrscheinlichkeit
 - Maximum Likelihood ergibt bei Mehrklassenproblemen die maximale Cross-Entropie (Gleichung 4).
- Für die **Modellselektion** kann man
 - die Kreuzvalidierung mit Fehlerminimierung und
 - die Kriterien nach Bayes (BIC, MDL) nutzen.

Was wissen Sie jetzt?

- Funktionsapproximation optimiert eine **Qualitätsfunktion**.
 - **Fehlerminimierung**, z.B. MSE
 - **Maximierung der Likelihood**, z.B. durch Approximation der a posteriori Wahrscheinlichkeit
 - Maximum Likelihood ergibt bei Mehrklassenproblemen die maximale Cross-Entropie (Gleichung 4).
- Für die **Modellselektion** kann man
 - die Kreuzvalidierung mit Fehlerminimierung und
 - die Kriterien nach Bayes (BIC, MDL) nutzen.

Was wissen Sie jetzt?

- Funktionsapproximation optimiert eine **Qualitätsfunktion**.
 - **Fehlerminimierung**, z.B. MSE
 - **Maximierung der Likelihood**, z.B. durch Approximation der a posteriori Wahrscheinlichkeit
 - Maximum Likelihood ergibt bei Mehrklassenproblemen die maximale Cross-Entropie (Gleichung 4).
- Für die **Modellselektion** kann man
 - die Kreuzvalidierung mit Fehlerminimierung und
 - die Kriterien nach Bayes (BIC, MDL) nutzen.

Was wissen Sie jetzt?

- Funktionsapproximation optimiert eine **Qualitätsfunktion**.
 - **Fehlerminimierung**, z.B. MSE
 - **Maximierung der Likelihood**, z.B. durch Approximation der a posteriori Wahrscheinlichkeit
 - Maximum Likelihood ergibt bei Mehrklassenproblemen die maximale Cross-Entropie (Gleichung 4).
- Für die **Modellselektion** kann man
 - die Kreuzvalidierung mit Fehlerminimierung und
 - die Kriterien nach Bayes (BIC, MDL) nutzen.

Was wissen Sie jetzt?

- Funktionsapproximation optimiert eine **Qualitätsfunktion**.
 - **Fehlerminimierung**, z.B. MSE
 - **Maximierung der Likelihood**, z.B. durch Approximation der a posteriori Wahrscheinlichkeit
 - Maximum Likelihood ergibt bei Mehrklassenproblemen die maximale Cross-Entropie (Gleichung 4).
- Für die **Modellselektion** kann man
 - die Kreuzvalidierung mit Fehlerminimierung und
 - die Kriterien nach Bayes (BIC, MDL) nutzen.

Literatur I