Support Vector Machines
... in a Nutshell

Björn Schuller
Dr.-Ing. cand. habil.

Lehrstuhl für Mensch-Maschine-Kommunikation
Technische Universität München

Freitag, 1. Juli 2011, 10:30 – 12:00
Promotionsprogramm "Empirische Sprachverarbeitung"
Institut für Phonetik und Sprachverarbeitung, Ludwig-Maximilians-Universität München

Outline

Einleitung

SVM & SVR

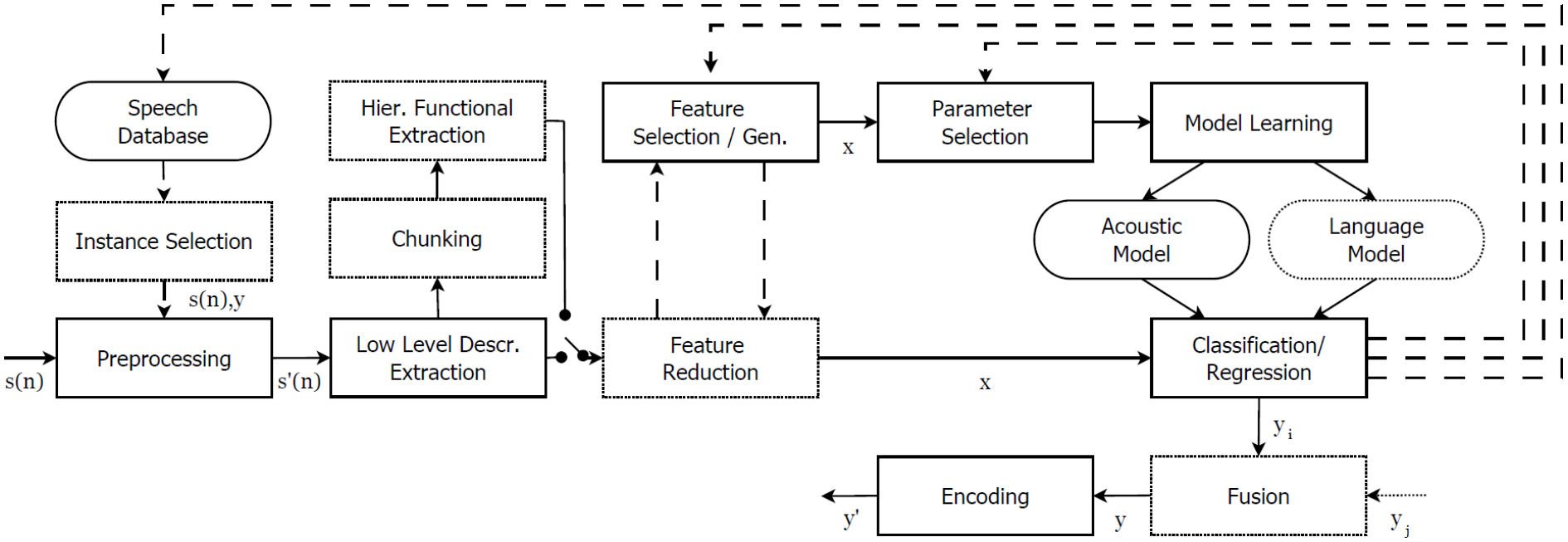
Praxis

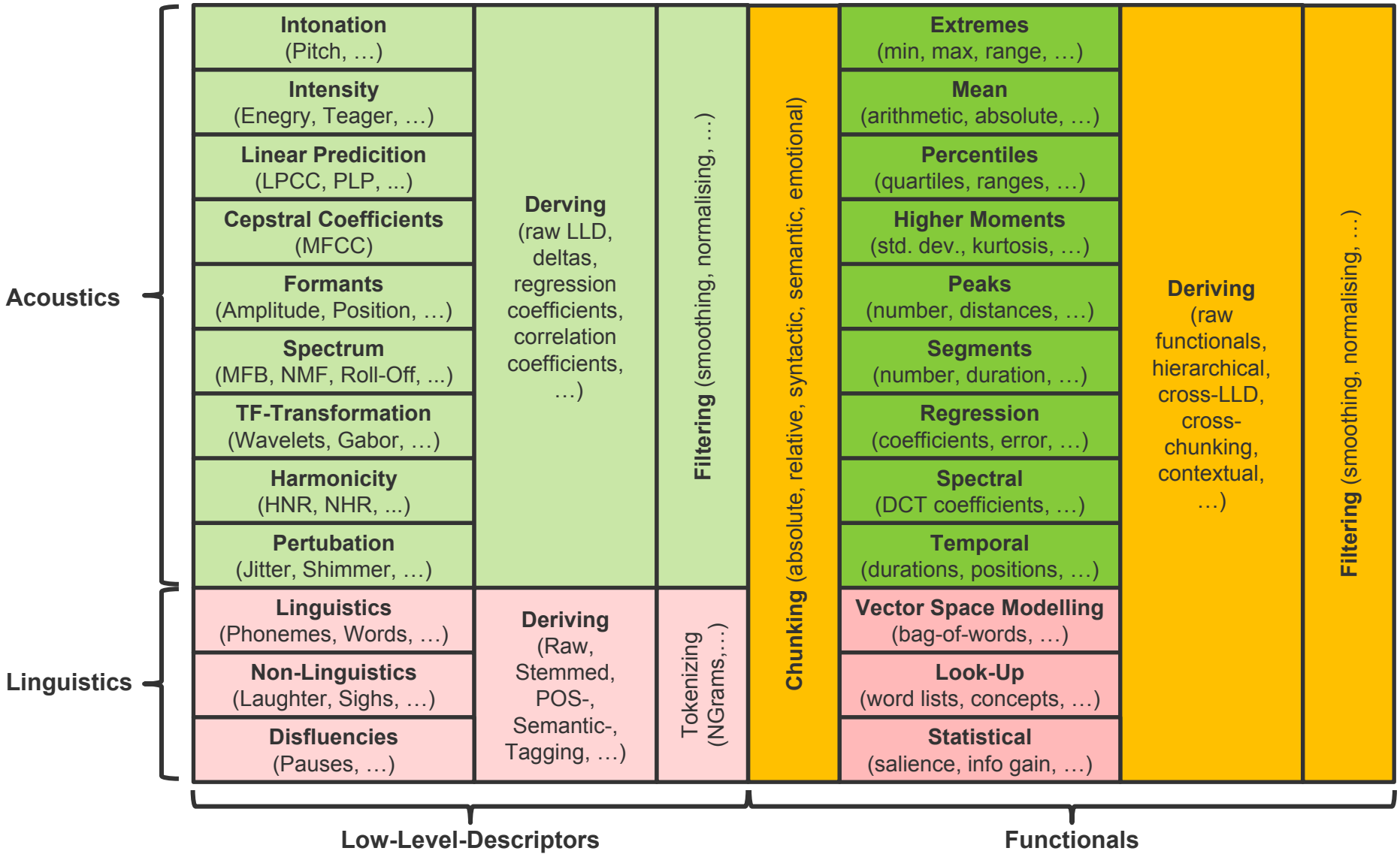
Diskussion

Einleitung

Einleitung

- **Audioklassifikation**
  Überblick




Acoustics
Linguistics
Intonation (Pitch, …)
Intensity (Enegry, Teager, …)
Linear Predicition (LPCC, PLP, ...)
Cepstral Coefficients (MFCC)
Formants (Amplitude, Position, …)
Spectrum (MFB, NMF, Roll-Off, ...)
TF-Transformation (Wavelets, Gabor, …)
Harmonicity (HNR, NHR, ...)
Pertubation (Jitter, Shimmer, …)
Linguistics (Phonemes, Words, …)
Non-Linguistics (Laughter, Sighs, …)
Disfluencies (Pauses, …)
Derving (raw LLD, deltas, regression coefficients, correlation coefficients, …)
Deriving (Raw, Stemmed, POS-, Semantic-, Tagging, …)
Filtering (smoothing, normalising, …)
Tokenizing (NGrams,…)
Chunking (absolute, relative, syntactic, semantic, emotional)
Extremes (min, max, range, …)
Mean (arithmetic, absolute, …)
Percentiles (quartiles, ranges, …)
Higher Moments (std. dev., kurtosis, …)
Peaks (number, distances, …)
Segments (number, duration, …)
Regression (coefficients, error, …)
Spectral (DCT coefficients, …)
Temporal (durations, positions, …)
Vector Space Modelling (bag-of-words, …)
Look-Up (word lists, concepts, …)
Statistical (salience, info gain, …)
Deriving (raw functionals, hierarchical, cross-LLD, cross-chunking, contextual, …)
Filtering (smoothing, normalising, …)
Low-Level-Descriptors
Functionals

Anforderungen Klassifikation

Adäquate Modellierung

Dynamische / statische Klassifikation
Daten- / wissensgetrieben
Umgang mit fehlenden Werten
Umgang mit Unsicherheiten
Trainingsstabilität
Modell- / instanzbasiert
Transparenz
Lösbarkeit nichtlinearer Probleme

Optimale Erkennungsleistung

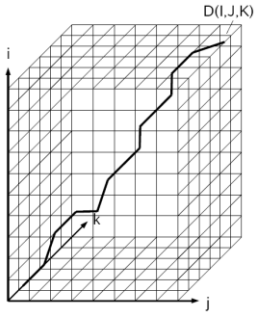
Maximierte Diskriminativität der Klassen
Priorisierung von Merkmalen
Toleranz ggü. Dimensionserhöhung
Nachtrainierbarkeit
Hohe Verallgemeinerungsfähigkeit
Adaption an Komplexität
Einsetzbarkeit diverser Merkmalsets

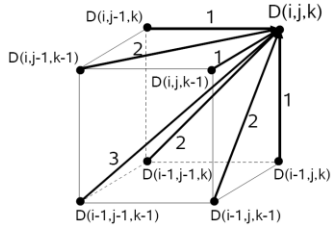
Anforderungen Klassifikation

Effizienz	Echtzeitfähigkeit in der Erkennung
	Kurze Trainingszeit
	Geringer Bedarf an Lernbeispielen
Ökonomische Faktoren	Geringer Bedarf an Rechenleistung
	Geringer Bedarf an Speicher
	Günstige Umsetzbarkeit in Hardware
Optimale Integration	Merkmalsreduktion ohne Trainingsbedarf
	Angabe über Sicherheit der Klassifikation
	Klassenbezogene Sicherheit
	Berücksichtigung von Eingangsunsicherheiten

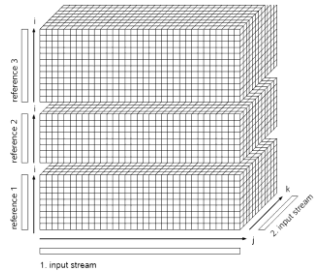
Dynamische Klassifikation

- **Dynamische Programmierung**
  Multidimensionaler-DTW

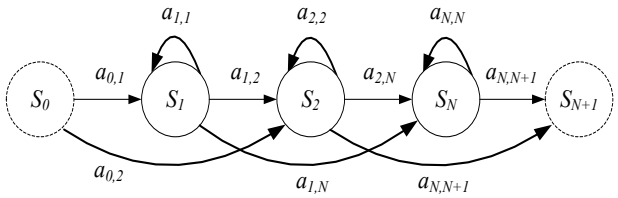




- **Dynamische Bayessche Netze**
  Hidden Markov Modelle



- **Hidden Conditional Random Fields**



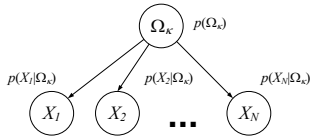
$$\lambda = \left(\underline{A}, b_1, \ldots, b_N\right)$$

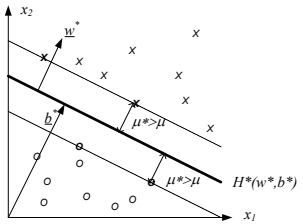
$$g\left(\underline{x}, \underline{m}_i, \underline{C}_i\right) = \frac{1}{\sqrt{(2\pi)^n} \cdot \left|\underline{C}_i\right|^{\frac{1}{2}}} \cdot \mathrm{e}^{\left(-\frac{1}{2}(\underline{x}-\underline{m}_i)^T \cdot \underline{C}_i^{-1} \cdot (\underline{x}-\underline{m}_i)\right)}$$

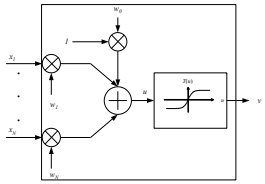
Statische Klassifikation

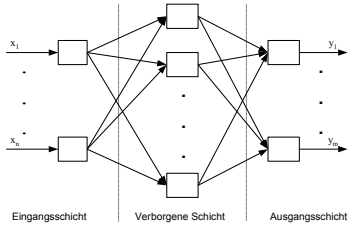
$$d_r\left(\underline{x},\underline{x}_l\right)=\sqrt[r]{\sum_{i=1}^{N}\left|x_i-x_{l,i}\right|^r} \qquad \kappa_e=\underset{\kappa}{\operatorname{argmin}}\, d\left(\underline{x},\underline{x}_{l,\kappa}\right)$$

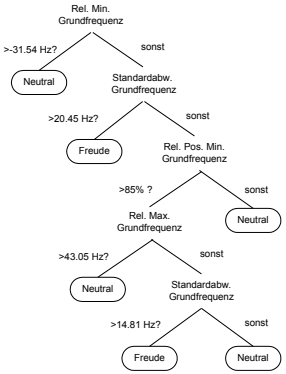
- **Lazy Learners**
  - Abstandsklassifikatoren
- **Stochastische Klassifikatoren**
  - Gaußsche Mixtur Modelle
  - (Diskriminative) Bayessche Netze
- **Funktionen**
  - Support-Vektor-Maschinen/Regression
  - (Long-Short-Term-Memory) Rekurrente Netze
  - Conditional Random Fields
- **Bäume**
  - Random Forests







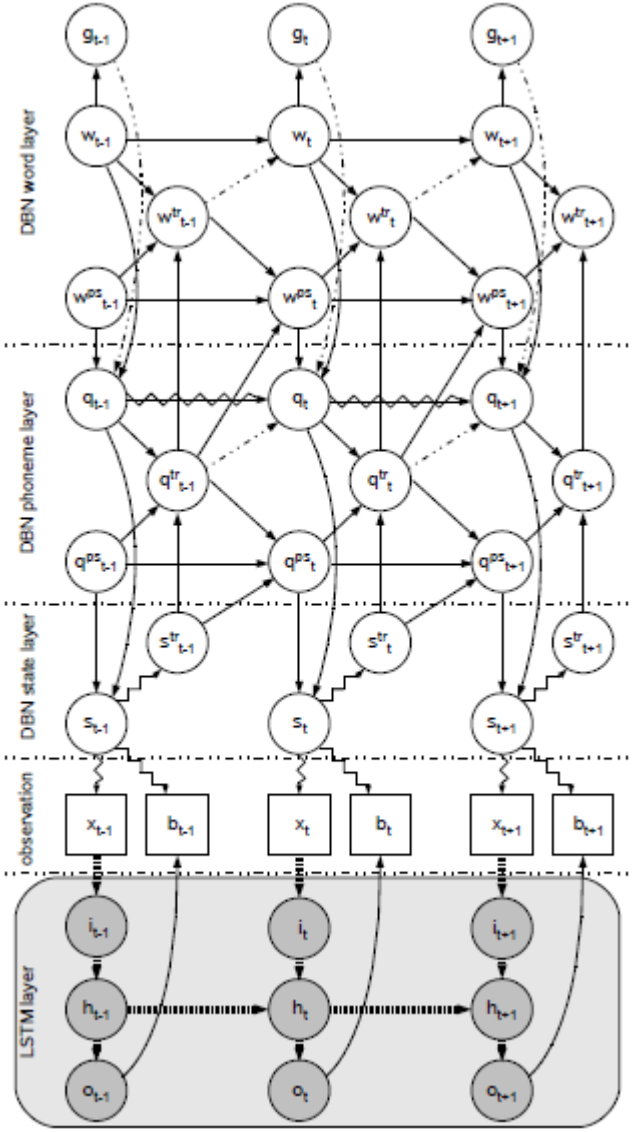




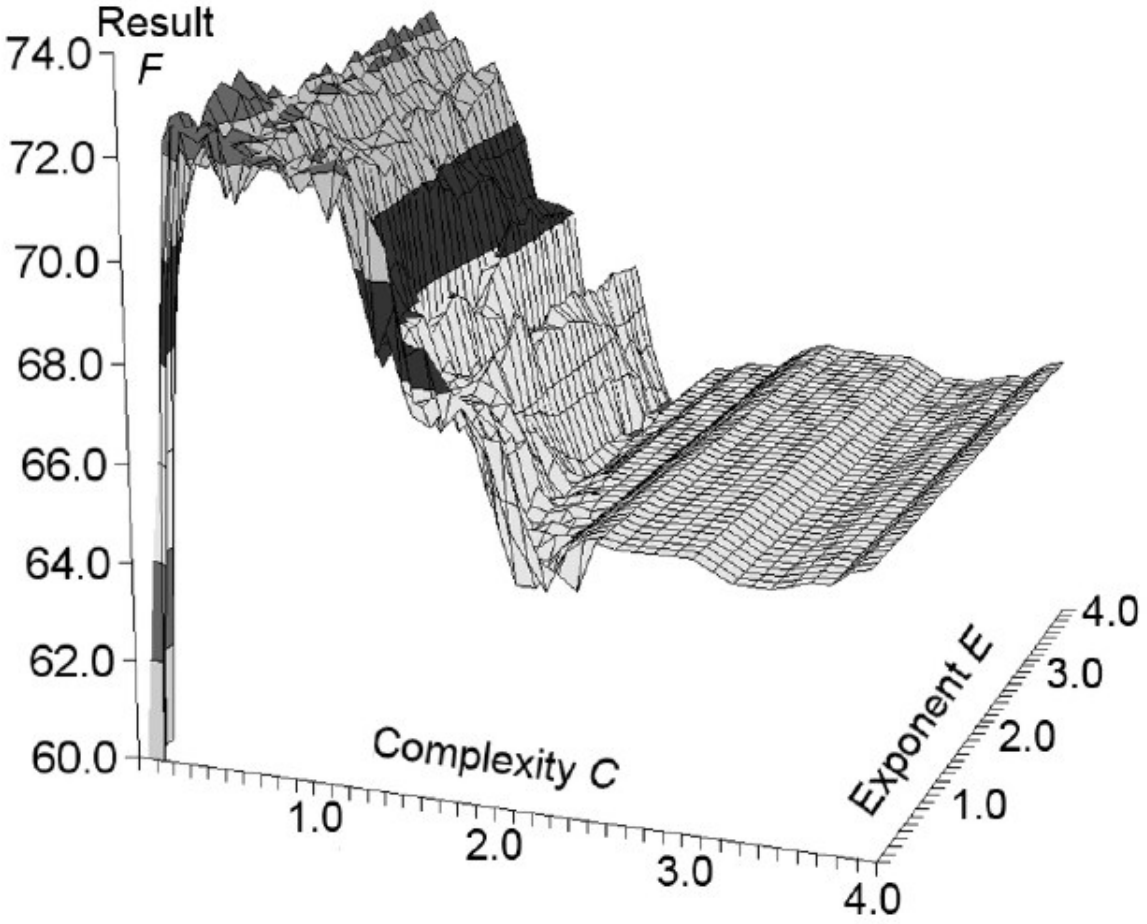
Metaklassifikation

- **Dynamisch/Statisch**
  - Tandem
  - Hybrid
- **Metalernen**
  - Ensembles (Boosting, Bagging, …)
  - Stacking
  - ROVER

$$\varepsilon = E\left\{\left(\hat{\theta}-\theta\right)^2\right\} = \left[E\left(\hat{\theta}\right)-\theta\right]^2 + E\left\{\left[\hat{\theta}-E\left(\hat{\theta}\right)\right]^2\right\}$$



Parametrierung



SVM & SVR

Support-Vektor-Maschinen

- **Hintergrund**
  Vapnik 1995

  Analogon zur Elektrostatik:

Trainingsbeispiel:	Geladener Leiter an bestimmten Ort im Raum
Entscheidungsfkt.:	Elektrostatische Potentialfunktion
Lernzielfunktion:	Coulomb‘sche Energie

Support-Vektor-Maschinen

- **Prinzip**

  Linearer Klassifikator

  (ähnlich Perzeptron mit linearer Aktivierungsfunktion)

  Kombiniert mit nichtlinearer Abbildung in höherdim. Entscheidungsraum

  Linearer Klassifikator dabei aus Untermenge der Lernbeispiele
  Sogenannte *Stützvektoren*
  Dadurch geringe Gefahr der Überadaption an die Lernmenge

  Auswahl Stützvektoren durch klassisches quadr. Optimierungsverfahren

Support-Vektor-Maschinen

- **Prinzip**

  Zunächst: Trennung von 2 Klassen

$$\mathcal{L} = \{(\mathbf{r}_l, y_l) \mid l = 1, \ldots, L\} \text{ mit } y_l \in \{+1, -1\}$$

  Hyperbene: Normalenvektor und Bias

$$H(\mathbf{w}, b) = \{\mathbf{r} \mid \mathbf{w}^T \mathbf{r} + b = 0\}$$

  So dass:

$$y_l = +1 \Rightarrow \mathbf{w}^T \mathbf{r}_l + b \geq +1,$$
$$y_l = -1 \Rightarrow \mathbf{w}^T \mathbf{r}_l + b \leq -1$$

Support-Vektor-Maschinen

- **Prinzip**

  Vorzeichenbehafteter Abstand zur Hyperebene:

$$D(\mathbf{r}) = \frac{\mathbf{w}^T \mathbf{r} + b}{||\mathbf{w}||}$$

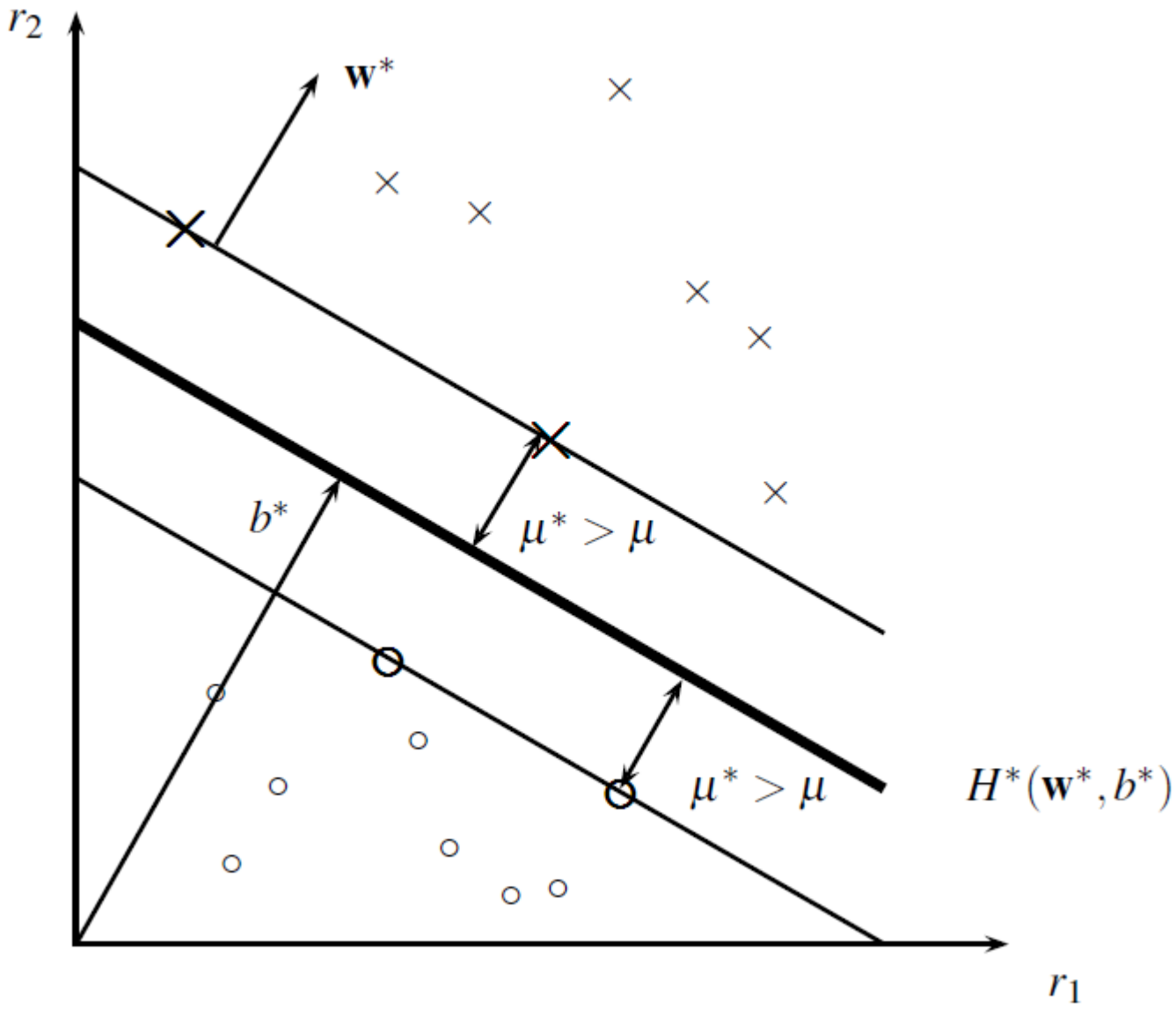
  Trennbreite (Margin of Separation):

$$\mu_{\mathcal{L}}(\mathbf{w}, b) = \min_{l=1,\ldots,L} |D(\mathbf{r}_l)|$$

  Ziel: Maximale Diskriminativität, d.h., max. Trennbreite
  Instanzen, die Hyperebene am nächsten sind, sind Stützvektoren

  Abstand Stützvektoren: $D^*(\mathbf{r}_l^{sv}) = \frac{\pm 1}{||\mathbf{w}||}$ Korridorbreite: $2\,||\mathbf{w}||^{-1}$

SVM



Support-Vektor-Maschinen

- **Prinzip**

  Statt Maximierung der Korridorbreite $2\,||\mathbf{w}||^{-1}$ Minimierung von $\frac{1}{2}\mathbf{w}^T\mathbf{w}$
  (streng konvex, eindeutiges Minimum)

  Daraus lineare Randbedingungen für Optimierung:

$$y_l\,(\mathbf{w}^T\mathbf{r}_l + b) - 1 \geq 0 \text{ mit } l = 1,\ldots,L$$

  Lösung des Randwertproblems u. A. mit Langrange‘schen Multiplikatoren

  Im Allgemeinen keine Hyperbene zur fehlerfreien Trennung
  Schlupfvariable:
  (Slack-Variable)
  (Soft Margin)

$$\xi_l \geq 0, l = 1,\ldots,L$$
$$y_l = +1 \Rightarrow \mathbf{w}^T\mathbf{r}_l + b \geq +1 - \xi_l,$$
$$y_l = -1 \Rightarrow \mathbf{w}^T\mathbf{r}_l + b \leq -1 + \xi_l$$

Support-Vektor-Maschinen

- **Prinzip**

  Mit G als Fehlergewichtfaktor nun zu minimieren (primales Problem):

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} + G \cdot \sum_{l=1}^{L} \xi_l$$

  Äquivalent zu dualem Problem der Maximierung:

$$\sum_{l=1}^{L} a_l - \frac{1}{2}\sum_{k=1}^{L}\sum_{l=1}^{L} a_k\, a_l\, y_k\, y_l (\mathbf{r}_k^T \mathbf{r}_l)$$

  Mit Nebenbedingungen:

$$0 \le a_l \le C,\ l = 1, \ldots, L,$$

$$\sum_{l=1}^{L} a_l\, y_l = 0$$

Support-Vektor-Maschinen

- **Prinzip**

  Hyperebene nun definiert durch:

$$\mathbf{w} = \sum_{l=1}^{L} a_l \, y_l \, \mathbf{r}_l, \qquad a_l \leq C, l = 1, \ldots, L$$

$$b = y_{l^*}(1 - \xi_{l^*}) - \mathbf{r}_{l^*}^T \mathbf{w}_{l^*}$$

  l^*: Index des Vektors $\mathbf{r}_l$ mit größtem a_l

  C: frei bestimmbar

  Einführung der Gewichtskoeff.: Eliminierung der Schlupfvariablen

  Normalenvektor ist gewichtete Summe der Trainingsbeispiele

  Stützvektoren sind diejenigen Vektoren mit $a_l > 0$

  Praxis: Häufig *Sequentielle Minimale Optimierung* (SMO) zur Lösung

Support-Vektor-Maschinen

- **Prinzip**
  Klassifikation

$$d_{\mathbf{w},b} : \mathbf{R} \rightarrow \{-1, +1\}$$

$$d_{\mathbf{w},b}(\mathbf{r}) = \operatorname{sgn}(\mathbf{w}^T \mathbf{r} + b)$$

$$\operatorname{sgn}(u) = \begin{cases} 1 & u \geq 0 \\ -1 & u < 0 \end{cases}$$

  Bisher: nur linear trennbare Zweiklassenprobleme

Support-Vektor-Maschinen

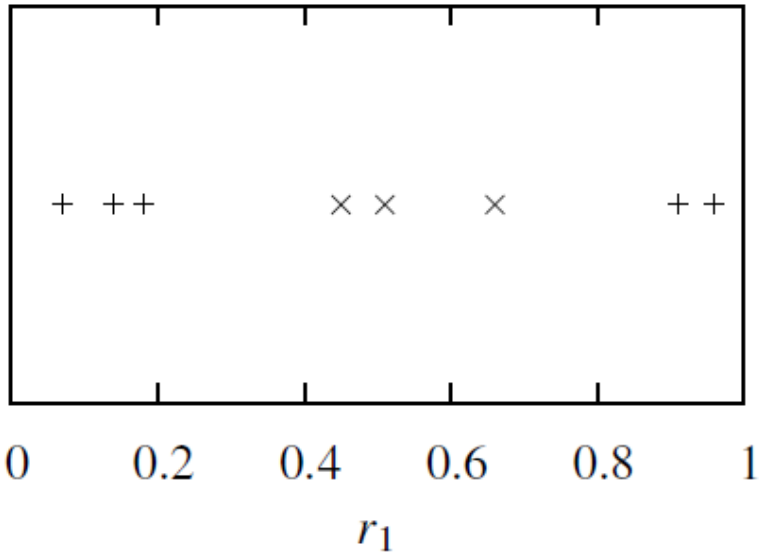
- **Kerneltrick**

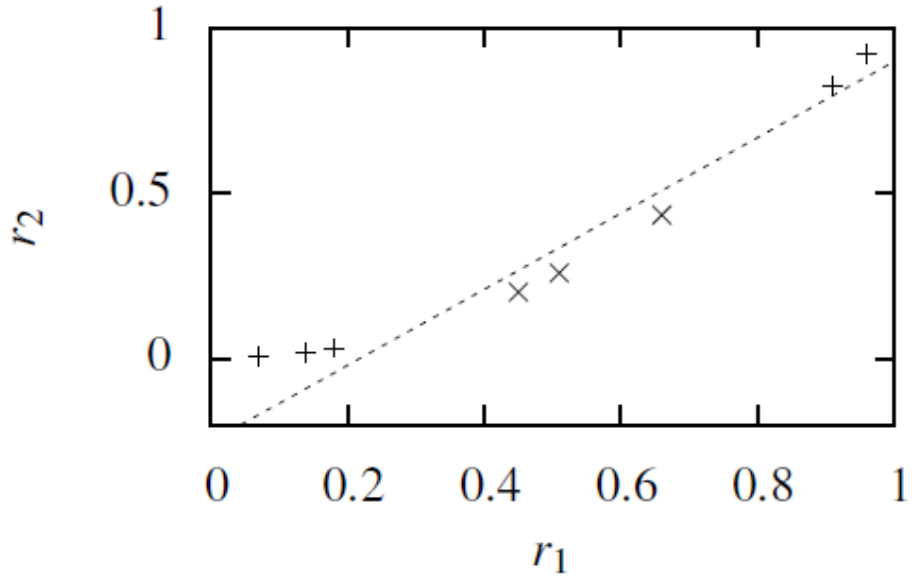
  Nichtlineare Transformation in höherdimensionalen Raum

$$\boldsymbol{\Phi} : \mathbf{R} \rightarrow \mathbf{R}', \quad \dim(\mathbf{R}') > \dim(\mathbf{R})$$

  Dort linear trennbar („hoffentlich“), Beispiel:

$$\boldsymbol{\Phi} : r_1 \mapsto (r_1, r_1^2)$$





Support-Vektor-Maschinen

- **Kerneltrick**

  Normalenvektor: $$\mathbf{w} = \sum_{l:a_l>0} a_l \, y_l \, \boldsymbol{\Phi}(\mathbf{r}_l)$$

  Entscheidungsfunktion:

  $$d_{\mathbf{w},b}(\mathbf{r}) = \operatorname{sgn}(\mathbf{w}^T \boldsymbol{\Phi}(\mathbf{r}) + b)$$

  Da: $$\mathbf{w}^T \boldsymbol{\Phi}(\mathbf{r}) = \sum_{l:a_l>0} a_l \, y_l \, \boldsymbol{\Phi}(\mathbf{r}_l)^T \boldsymbol{\Phi}(\mathbf{r})$$

  Transformation nicht zur Parameterschätzung/Klassifikation expl. benötigt

  Stattdessen *Kernelfunktion*
  (pos. semidefinit, symmetrisch, Cauchy-Schwarz-Ungleichung erfüllen)

  $$K^{\Phi}(\mathbf{r},\mathbf{s}) = \boldsymbol{\Phi}(\mathbf{r})^T \boldsymbol{\Phi}(\mathbf{s})$$

Support-Vektor-Maschinen

- **Kerneltrick**

  Beste Kernelfunktion nur empirisch

  Polynom-Kernel (Polynomordnung):

$$K_p^{\Phi}(\mathbf{r},\mathbf{s}) = (\mathbf{r}^T\mathbf{s}+1)^p$$

  Gauß-Kernel (Std.abw., radiale Basisfunktion):

$$K_\sigma^{\Phi}(\mathbf{r},\mathbf{s}) = e^{\frac{||\mathbf{r}-\mathbf{s}||^2}{2\sigma^2}}$$

  Sigmoid-Kernel (Verstärkung, Offset):

$$K_{k,\Theta}^{\Phi}(\mathbf{r},\mathbf{s}) = \tanh(k(\mathbf{r}^T\mathbf{s})+\Theta)$$

  String-Kernel, etc.

Support-Vektor-Maschinen

- **Kerneltrick**
  Kernelfunktion statt Transformation erniedrigt Rechenaufwand erheblich
  Für hochdim. Probleme nutzbar

  Beispiel: Polynom Grad p
  Zahl zu berechnender Terme

$$\binom{\dim(\mathbf{R})+p}{p} \approx \frac{\dim(\mathbf{R})^p}{p!}$$

  Polynom-Kernel unabh. von p
  Zahl zu berechnender Terme ca.

$$\dim(\mathbf{R})$$

Support-Vektor-Maschinen

- **Mehrklassenklassifikation**
  L: # Entscheidungen, M: # Klassen

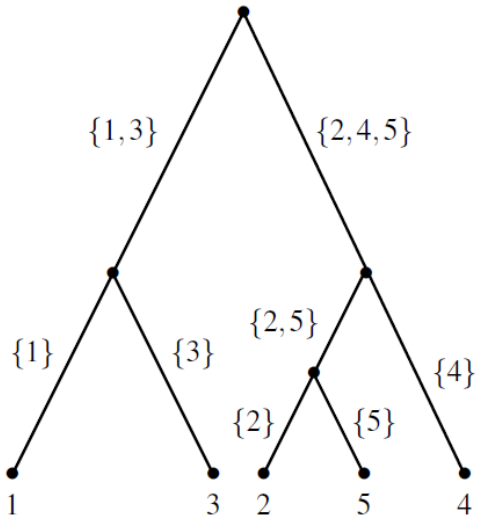
  Entscheidungsregeln:

  1-vs-alle: $L = M$

  1-vs-1: $L = M(M-1)/2$
  Vergleich mit jeder anderen Klasse positiv
  oder Mehrheitsentscheid

  ½-vs-½ (Bäume) im Mittel: $L = \log_2(M)$

  „Echte“ Mehrklassen-SVM…



Support-Vektor-Regression

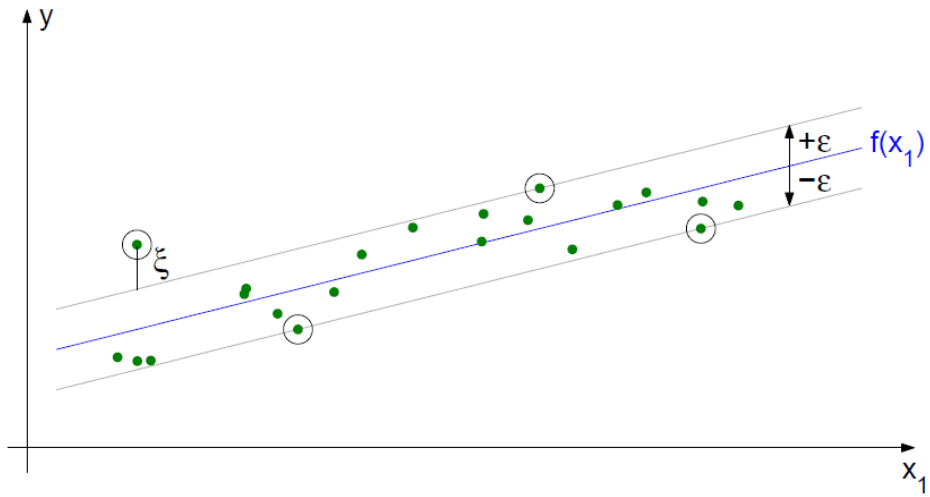
- **Prinzip**

  Grundsätzlich jeder Klassifikator in Regressor wandelbar:
  Regression durch Diskretisierung

  SVR:

$$f(\underline{x}) = \underline{w}^T \underline{x} + b$$

  Ausgang lineare Regression
  Möglichst flache Abweichung
  Schlupfvariablen
  Langrage‘sche Multiplikatoren
  Stützvektoren
  Kerneltrick



Support-Vektor-Varianten

- **Relvance-Vektor-Maschinen (RVM)**
  Probabilistische Klassifikation
  Äquivalent zu Gauß‘schem Prozess mit Kovarianz
  Meist Gauß‘scher Kernel
  Bayes‘sche Formulierung vermeidet Parametersuche
  EM-ähnlicher Lernalgorithmus
  Anders als mit SMO keine Gewährleistung globalen Optimums

- **Hybride Ansätze**
  z.B. SVM-HMM, SVM-DBN, SVM-GM (Tandem, Hybrid)

  Hybride Ensembles: Stacking, StackingC, Voting, etc.

Support-Vektor-Varianten

- **GMM Supervektoren**

  Stacking der Mittelwerte der Gauß‘schen-Mixtur-Modelle als Merkmale

  GMM Universelles-Background-Modell (UBM), z.B. mit EM

  MAP anwendbar

$$g(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \mathcal{N}(\mathbf{x}; \mathbf{m}_i, \mathbf{\Sigma}_i)$$

Praxis

SVM Software

- **SVMlight**

  http://svmlight.joachims.org/

  Fast optimization algorithm, works on very large datasets, efficient leave-one-out cross-validation.

  Source: C++, binaries: Linux, Windows, Cygwin, Solaris

  Kernels: polynomial, radial basis function, and neural (tanh)

- **SVMstruct**

  http://svmlight.joachims.org/svm_struct.html

  model complex (multivariate) output data y, such as trees, sequences, or sets. Can be applied to natural language parsing, and Markov models for part-of-speech tagging, multi-class classification

SVM Software

- **mySVM**
  http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/index.html
  Source: C++, binaries: Windows
  Kernels: linear, polynomial, radial basis function, neural (tanh), anova

- **JmySVM**
  http://www-ai.cs.uni-dortmund.de/SOFTWARE/YALE/index.html
  part of YaLE (Yet Another Learning Environment)/Rapid Miner
  Source: Java

- **Weka**
  http://www.cs.waikato.ac.nz/ml/weka/
  Collection of machine learning algorithms for data mining tasks
  Source: Java

SVM Software

- **LIBSVM**
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - Multi-class classification, weighted SVM for unbalanced data, cross-validation and automatic model selection
  - Source: C++, Java
  - Interfaces for Python, R, Splus, MATLAB, Perl, Ruby, and LabVIEW.
  - Kernels: linear, polynomial, radial basis function, and neural (tanh)

- **SVMTorch**
  - http://www.idiap.ch/learning/SVMTorch.html
  - Part of the Torch machine learning library
  - Source: C++, Binaries: Linux, Solaris

SVM Software

- **SVM in R**
  http://cran.r-project.org/src/contrib/Descriptions/e1071.html
  C-classification, n-classification, e-regression, and n-regression
  Kernels: linear, polynomial, radial basis, neural (tanh)

- **BSVM**
  http://www.csie.ntu.edu.tw/~cjlin/bsvm/
  Two implementations of multi-class classification
  Source: UNIX/Linux, binaries: Windows

- **M-SVM**
  http://www.loria.fr/~guermeur/
  Multi-class SVM implementation
  Source: C

SVM Software

- **MATLAB SVM Toolbox**
  http://www.isis.ecs.soton.ac.uk/resources/svminfo/
  Kernels: linear, polynomial, Gaussian radial basis function, exponential radial basis function, neural (tanh), Fourier series, spline, B spline

- **TinySVM**
  http://chasen.org/~taku/software/TinySVM/
  sparse vector representation, handles several ten-thousands of training examples, hundred-thousands of feature dimensions
  Source: C++ Linux, binary: Windows

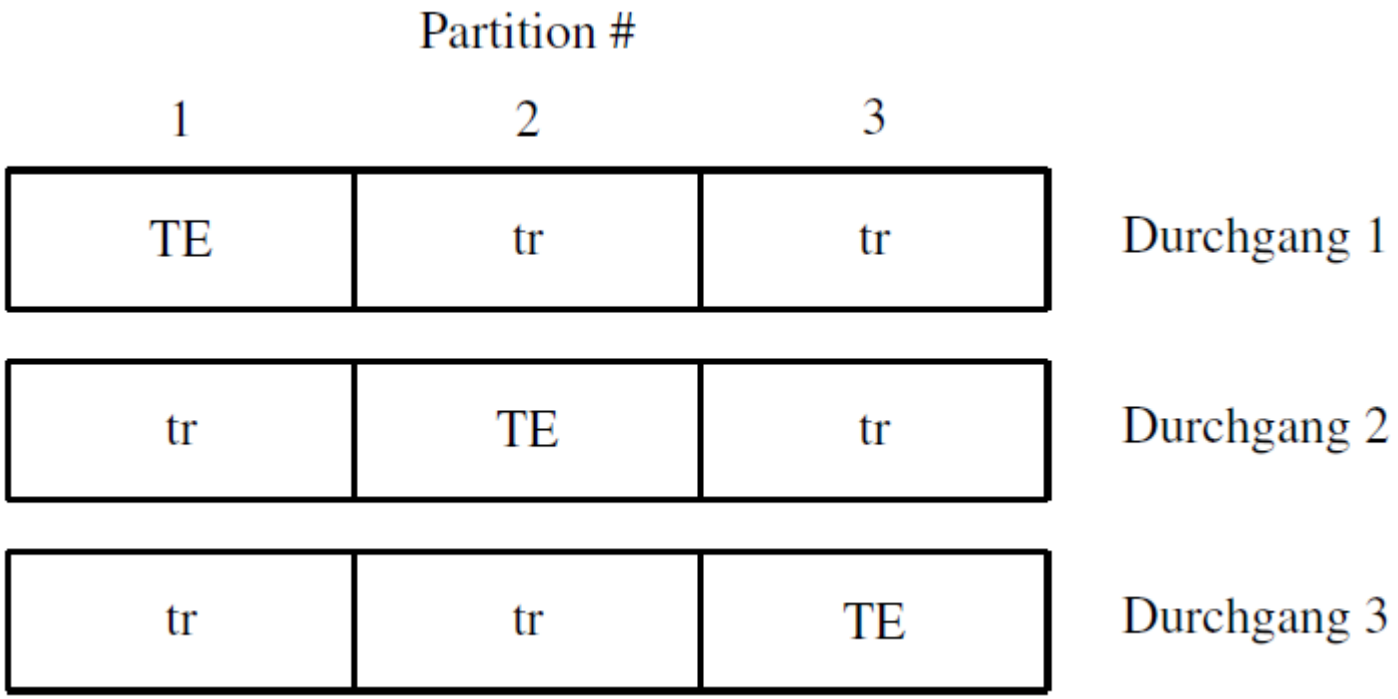
- **SvmFu**
  http://five-percent-nation.mit.edu/SvmFu/
  Source: C++
  Kernels: linear, polynomial, Gaussian radial basis function

SVM Software

- **Gini-SVM**
- **Gist**
- **GPDT**
- **HeroSvm**
- **LearnSC**
- **MATLAB SVM Toolbox**
- **OSU SVM Classifier Matlab Toolbox**
- **SimpleSVM Toolbox (Matlab)**
- **SmartLab, Spider (Matlab)**
- **SVMsequel**
- **SVM Toolbox (Matlab)**
- **Tree Kernels**
- **Und viele mehr…**

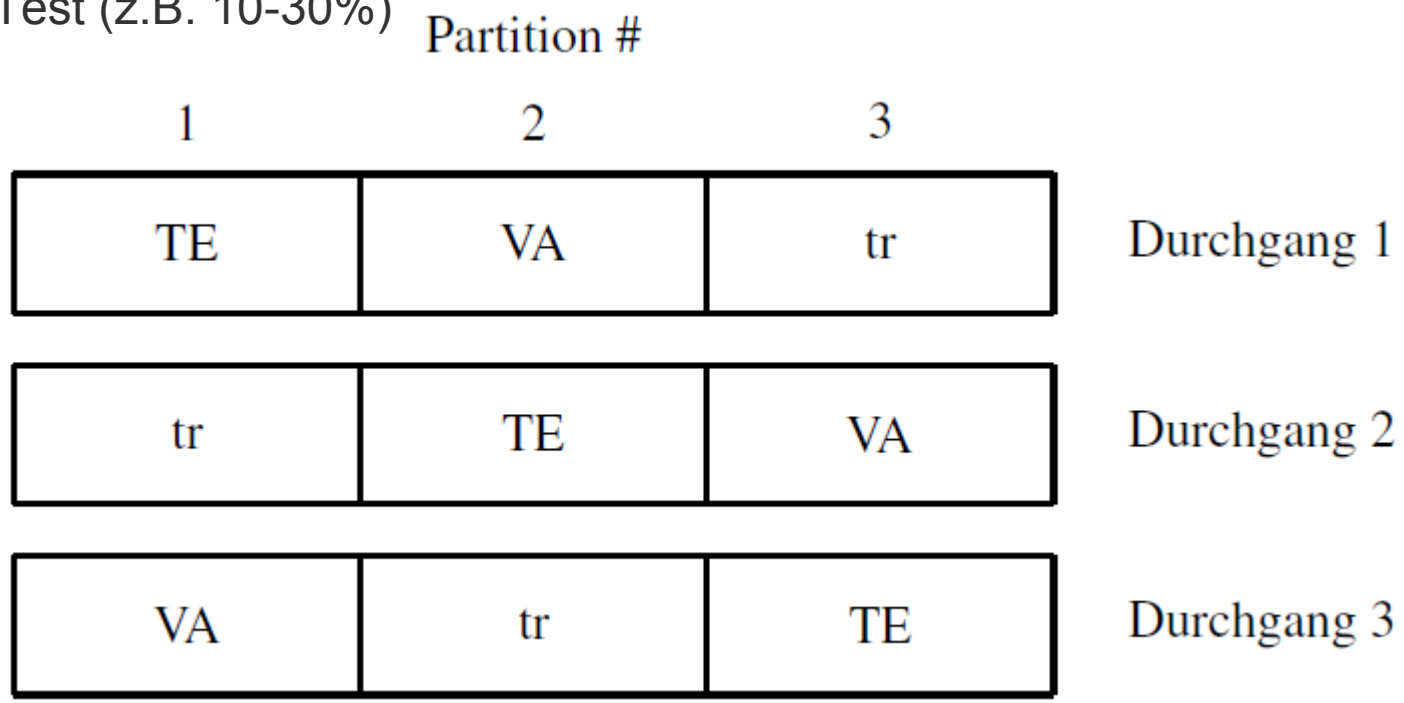
Evaluierung

- **Partitionierung**
  Ausbalancierung (z.B. Hochsampling, SMOTE)
  J-fache Kreuzvalidierung (stratifizieren)



Evaluierung

- **Partitionierung**
  - tr: Training (z.B. 40%)
  - VA: Validierung (z.B. 30%)
  - TE: Test (z.B. 10-30%)



Evaluierung

- **Partitionierung**

  Zusätzliche Unabhängigkeit: Leave-One-Speaker/Speaker Group-Out

  Reproduzierbarkeit wahren

  Transparenz wahren

  Keine Vorselektion

  Cross-Corpus

Evaluierung

- **Maße**

Accuracy / Recall:

$$\mathrm{RE} = \frac{\text{Anzahl richtig erkannter Muster}}{\text{Umfang der Teststichprobe}}$$

$$\mathrm{RE}_i = \frac{|\{\mathbf{r} \in \mathcal{T}_i \mid d(\mathbf{r}) = i\}|}{T_i}$$

Unweighted Average Recall:

$$\mathrm{UAR} = \frac{\sum_{i=1}^{M} \mathrm{RE}_i}{M}$$

True / False Positives:

$$\mathrm{FP}_i = |\{\mathbf{r} \in \mathcal{T} - \mathcal{T}_i \mid d(\mathbf{r}) = i\}|$$

Precision:

$$\mathrm{PR}_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i}$$

Evaluierung

- **Maße**

F-Measure:

$$F_i = 2\frac{\mathrm{RE}_i \mathrm{PR}_i}{\mathrm{RE}_i + \mathrm{PR}_i}$$

True / False Negatives:

$$\mathrm{TN}_i = \left|\{\mathbf{r} \in \mathcal{T} - \mathcal{T}_i \mid d(\mathbf{r}) \neq i\}\right|,$$
$$\mathrm{FN}_i = \left|\{\mathbf{r} \in \mathcal{T}_i \mid d(\mathbf{r}) \neq i\}\right|$$

Konfusionen:

$$k_{i,j} = \left|\{\mathbf{r} \in \mathcal{T}_i \mid d(\mathbf{r}) = j\}\right|$$

Evaluierung

- **Maße**

Aus Konfusionen:

$$\mathrm{TP}_i = k_{i,i},$$

$$\mathrm{FP}_i = \sum_{i \neq j} k_{j,i},$$

$$\mathrm{RE}_i = k_{i,i} / T_i = k_{i,i} / \sum_{j=1}^{M} k_{i,j},$$

$$\mathrm{RE} = \mathrm{spur}\{\mathbf{K}\} / |\mathcal{T}|,$$

$$\mathrm{PR}_i = k_{i,i} / \sum_{j=1}^{M} k_{j,i}$$

Evaluierung

- **Maße für binäre Entscheidungen**
  Receiver-Operating-Curve (ROC)

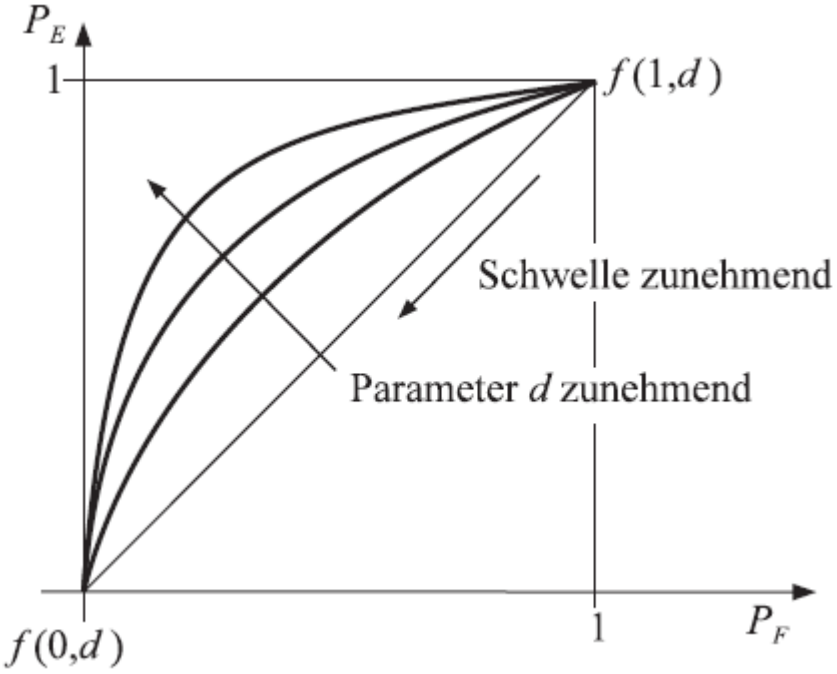
  Equal-Error-Rate (EER)

  Area-Under-Curve (AUC)

- **Maße für kont. Entscheidungen**
  Cross-Correlation (CC)

  Mean-Linear-Error (MLE) /
  Mena-Absolute-Error (MAE)

  Mean-Square-Error (MSE)



Probleme

- **Wahl Kernel und Parameter-Optimierung**
  Multiple-Kernel-Learning (MKL): Lernen des optimalen Kernels
  Gewichtete Summierung der Kernel äquiv. Merkmalsraumkonkatinierung
  Grid-Search, etc.

- **Große Datenmenge**
  Sukzessives Einlesen und Trainieren

- **Hohe Dimensionalität**
  Merkmalsauswahl
  PCA / LDA, etc.

- **(Geringe) Trainingsinstabilität**
  Bagging, Boosting, etc.

Diskussion

Support-Vektor-Maschinen

- **Konklusion**

  Generalisierte lineare Klassifikatoren

  Erweiterung des Perzeptrons

  Diskriminativität: Maximum-Margin-Klassifikator

Support-Vektor-Maschinen

- **Vorteile**
  - Hohe Dimensionen behandelbar
  - Gute Generalisierung
  - Diskriminativer Ansatz
  - Inhärente Instanzselektion

- **Nachteile**
  - Abhängig von Parameterwahl: Kernel, Kernelparam., Soft-Margin-Param.
  - Unkalibrierte Klassenzugehörigkeitswahrscheinlichkeiten
  - Kein Speicher (wie etwa RNN oder (B)LSTM, etc.)
  - Keine dynamische Klassifikation
    (wie HMM oder DBN, nur Multi-Instanz-Lernen)
  - Keine Merkmalsselektion (SVM-SFFS, etc.)

Support-Vektor-Maschinen

- **Ausblick**

  Transduktive SVM: Erlauben semi-überwachtes Lernen

  Strukturierte SVM: Multi-label und stukturierte Label (Bäume)

  Least-Square SVM: Lösen Satzes linearer Gleichungen anstelle konvexer quadr. Programmierung

  Multiple-Instanzen-SVM: „Bag-Of-Frames“:

  Bag „-“ wenn alle Frames „-“

  Bag „+“ falls >=1 Frame „+“

Literatur

- **Vapnik et al.**

  Vapnik, V.; Cortes, C.: Support vector networks,Machine Learning 20 (1995), S. 273-297.

- **Schölkopf et al.**

  Schölkopf, B.; Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning), 2002, Cambridge, MA: MIT Press, ISBN 0-262-19475-9.

Danke.

http://www.openaudio.eu