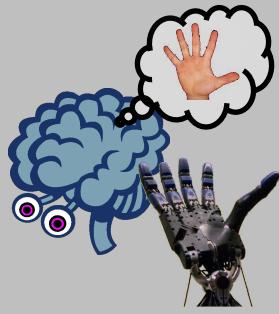
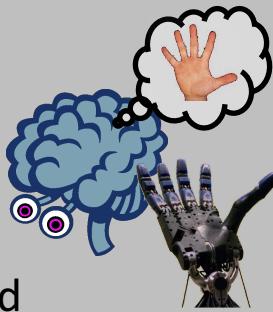


Workshop (“Übungen”)



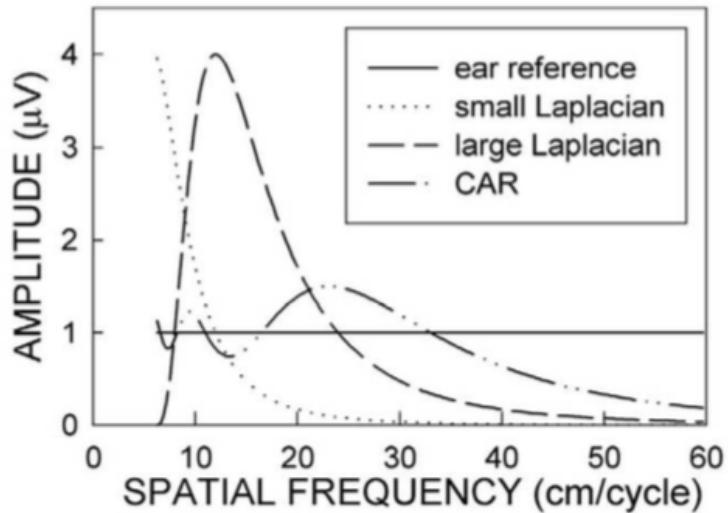
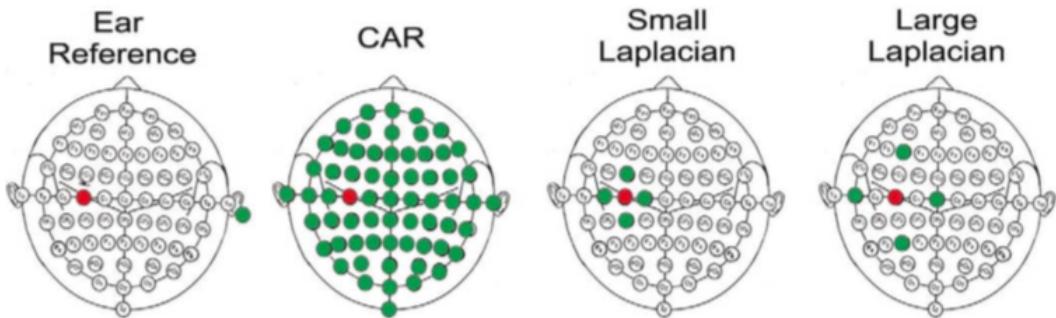
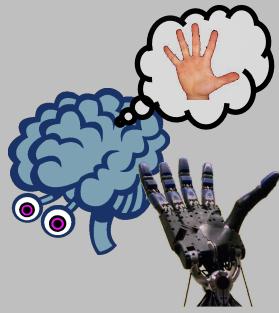
- 2-day block workshop after lecture period (mandatory)
- About 10 participants per group
- Takes place in EEG/Eyetracking-Lab, CITEC 3.044/45
- Date options (09:00 – 18:00 h)
 - Mo+Tue 20.+21. Feb
 - Mo+Tue 27.+28. Feb
 - Thu+Fri 02.+03. Mar
 - Mo+Tue 06.+07. Mar
 - Thu+Fri 09.+10. Mar
- Doodle Poll: <http://doodle.com/poll/3nhsfuiryn6aq9ys>
- Please mark ALL dates at which you are available!!!
- Deadline: 23.01.2017



Räumliche (Spatiale) Filter

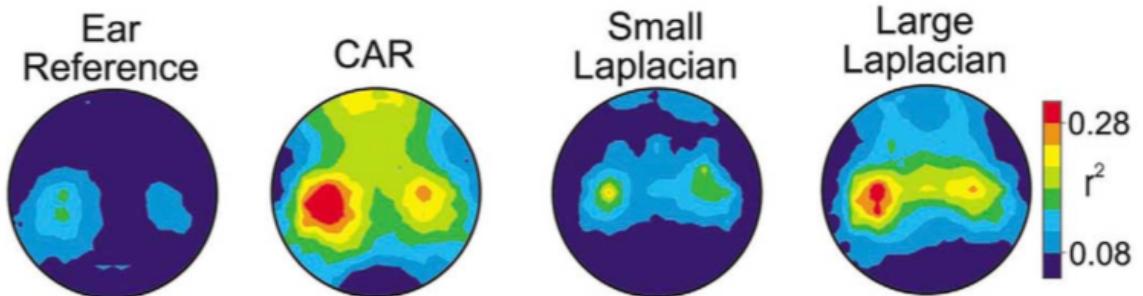
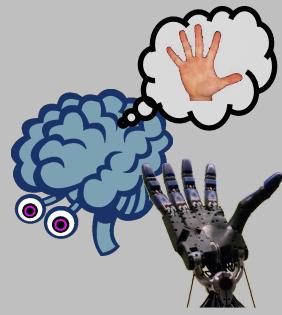
- Ähnlich wie bei den zeitlichen (Frequenz) Filtern, gibt es räumliche Hoch- und Tiefpassfilter.
- Die räumliche Frequenz gibt an, mit welcher Geschwindigkeit sich eine Welle durch das Medium bewegt.
- Sie wird angegeben in cm je Periode (cm/cycle).
- Ein räumlicher **Tiefpassfilter** entfernt Signalanteile, die eine **hohe** räumliche Frequenz haben, also sich über größere Bereiche des Cortex erstrecken (sowie alle stehenden Wellen).
- Ein räumlicher **Hochpassfilter** entfernt Signalanteile, die eine **niedrige** räumliche Frequenz haben, also nur sehr lokal auftreten (Der Schädelknochen hat „natürliche“ Tiefpass Charakteristika → Volume Conduction).
- Räumliche Filter werden praktisch nur bei EEG-Daten angewendet.
- Räumliche Filter (i.d.R. Tiefpassfilter) sind nur für bestimmte Anwendungen interessant, bei denen eine lokalisiertes Auftreten relevanter Signalanteile bekannt ist.²

Räumliche (Spatiale) Filter

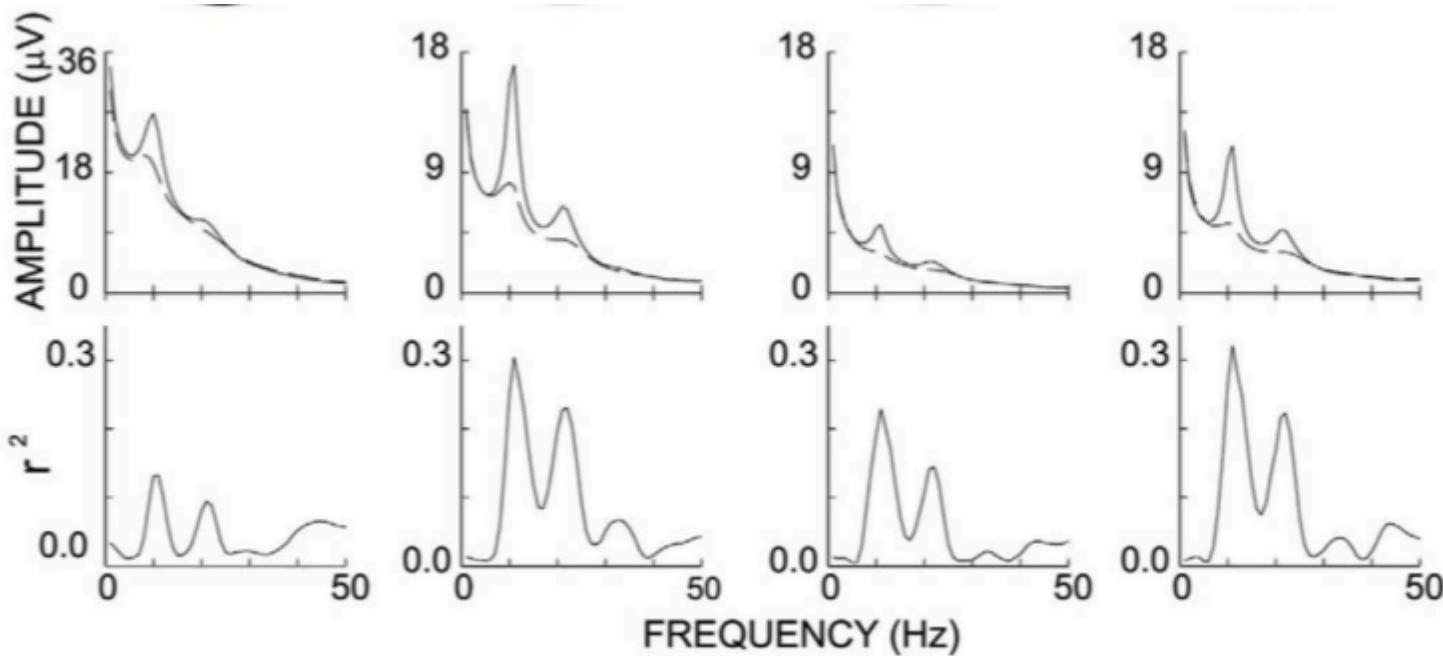


- Verschiedene räumliche Hochpassfilter.
- Ihre Realisierung entspricht einer Re-Referenzierung des Signals.
- Die Werte an den gewählten neuen Referenzelektroden werden gemittelt und von der Messelektrode subtrahiert.
- Diese Filter entfernen zum Teil den Einfluss des Volume Conduction.
- Die Filter-Eigenschaften der unterschiedlichen Methoden.

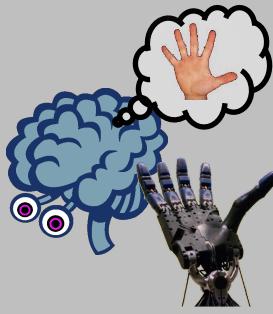
Räumliche Filter – Beispiel Motor Imagery Data



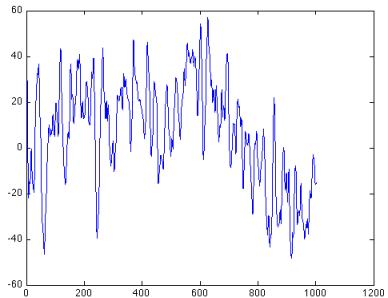
Einfluss der gewählten Filtermethode auf die Topologie der Aktivierung während der Vorstellung einer Bewegung der rechten Hand.



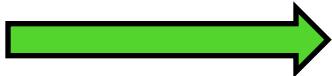
Merkmalsextraktion und Klassifikation (Feature extraction and classification)



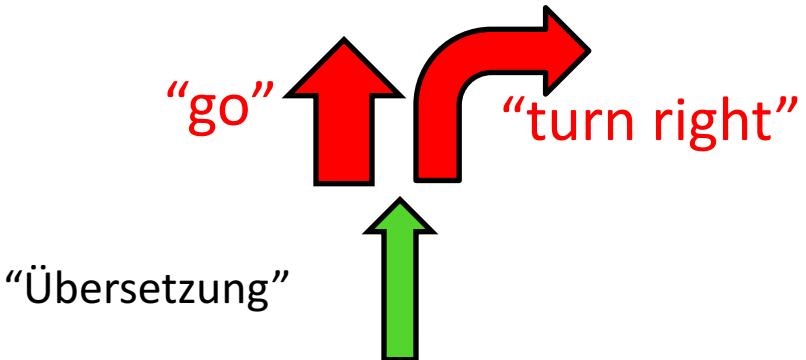
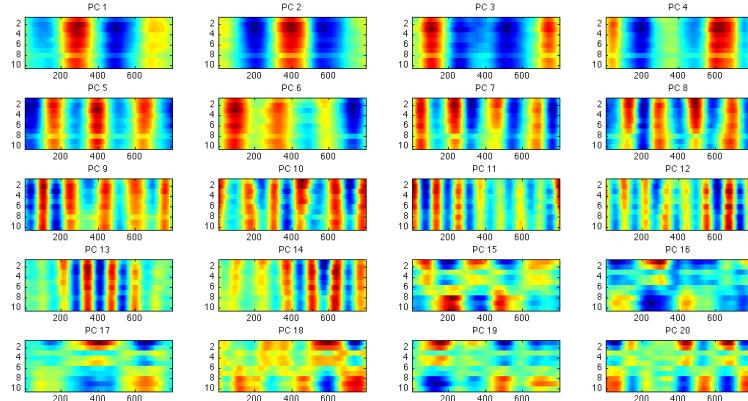
Rohdaten x



Pre-processing
Feature Extraction



Merkmale (Feature)



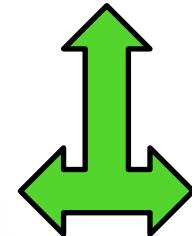
Klassifikationsergebnis y

-1

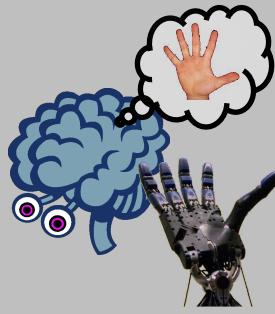
1 Label

Klassifikator

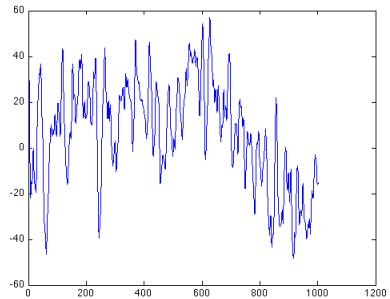
$$f : X \rightarrow Y \\ \text{mit } y \in \{-1, 1\}^*$$



Merkmalsextraktion und Regression (Feature extraction and regression)



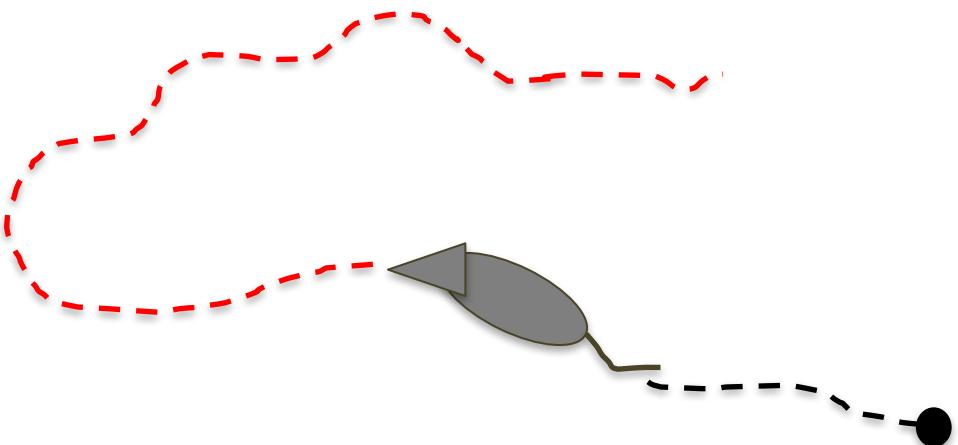
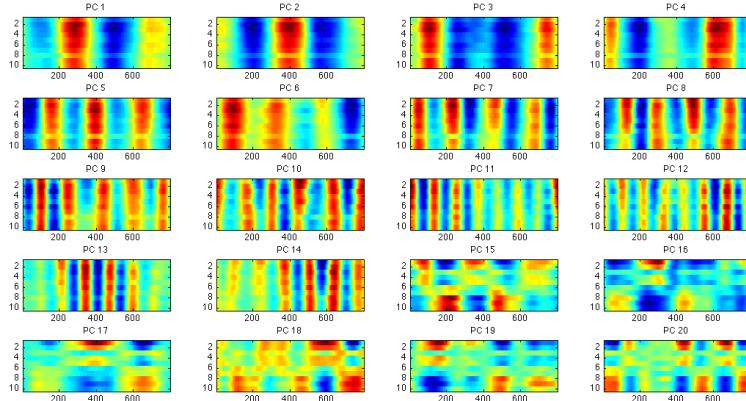
Rohdaten



Pre-processing
Feature Extraction



Feature

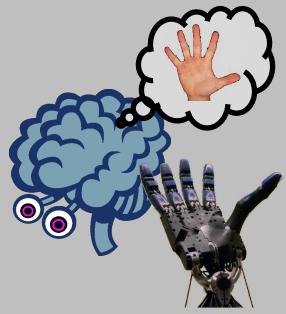


Lineares Model

$$f : X \rightarrow Y \\ \text{mit } y \in \mathbb{R}$$



z.B. Raumkoordinaten (Trajektorie)



Merkmalsextraktion

Ziel

Erzeugung von möglichst niedrigdimensionalen Merkmalsvektoren als Eingabe für einen Klassifikator.

Grund

Je höher die Dimension der Eingabedaten ist, desto mehr Trainingsdaten braucht man für einen Klassifikator.

Nebenbedingung

Wähle Merkmalsvektoren bzw. Extraktionsverfahren so, dass Klassifikationsleistung maximiert wird (“Problemgeleitet”)

Definition

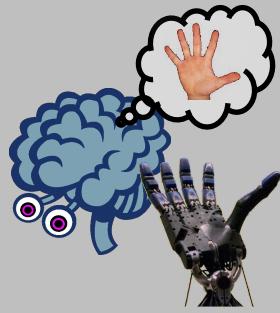
$$\vec{f}(\vec{x}) = \begin{pmatrix} f_1(x_1 \dots x_n) \\ \vdots \\ f_m(x_1 \dots x_n) \end{pmatrix}$$

Messdaten (Signalmatrix,
i.d.R. gefiltert)

$$\psi : \psi(\vec{f}(\vec{x})) = \vec{v} = (v_1 \dots v_N)$$

Merkmalsvektoren

Ein einfaches Beispiel



Eingabedaten

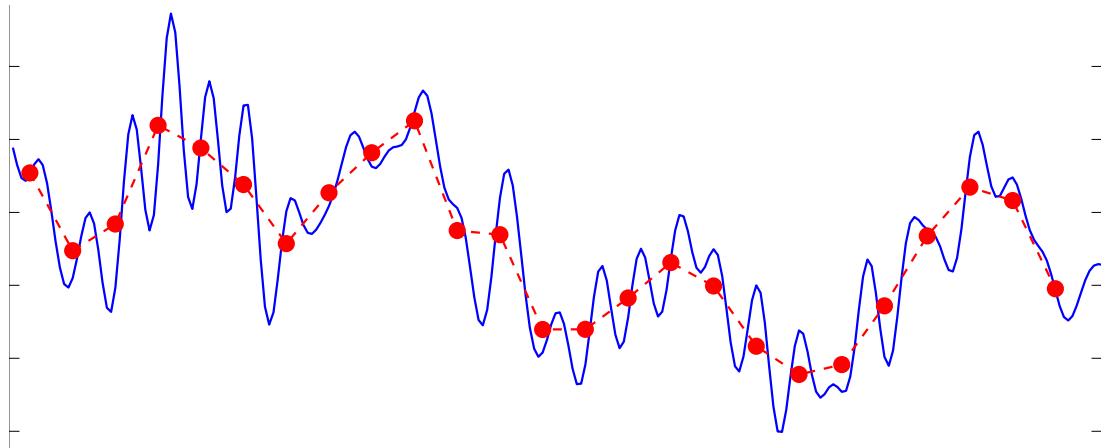
- 10-Kanal EEG Daten
- Datensegmente von 1 s Länge
- Sampling Rate 256 Hz
- Vorgefiltert mit 0.5 – 40 Hz Bandpass (FIR-Filter)

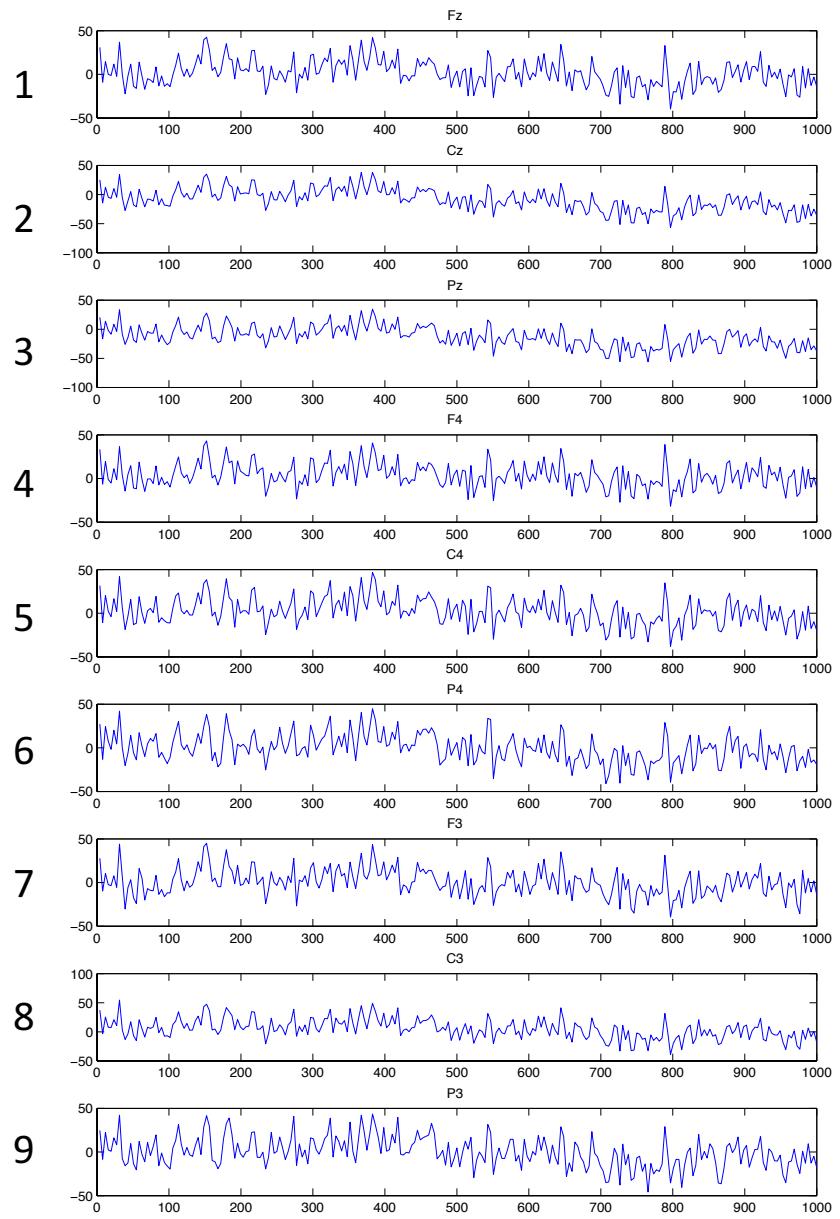
$$\vec{f}(\vec{x}) = \begin{pmatrix} f_1(x_1 \dots x_n) \\ \vdots \\ f_m(x_1 \dots x_n) \end{pmatrix}$$

$n = 256$
 $m = 10$

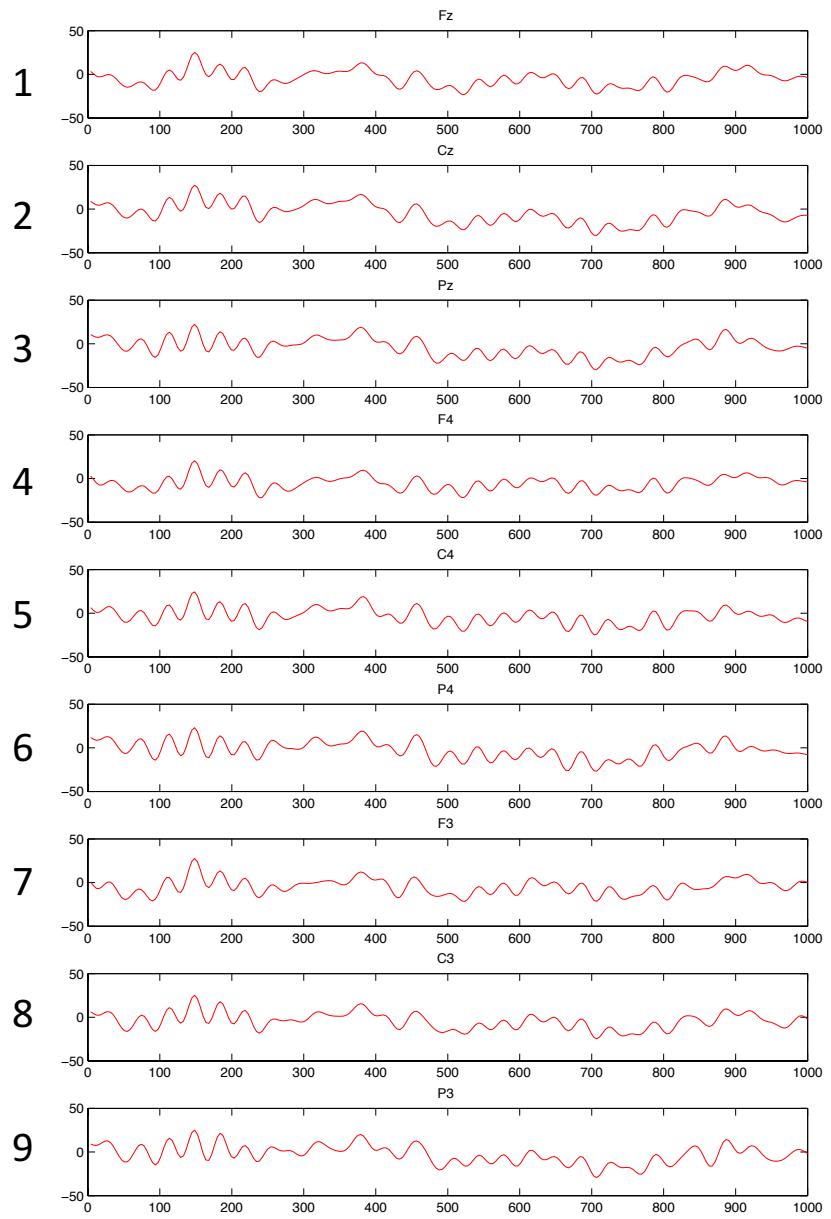
Feature Extraction

Downsampling mit Mittelung
→ Reduzieren der Datenmenge
durch Mittelung über
Zeitfenster



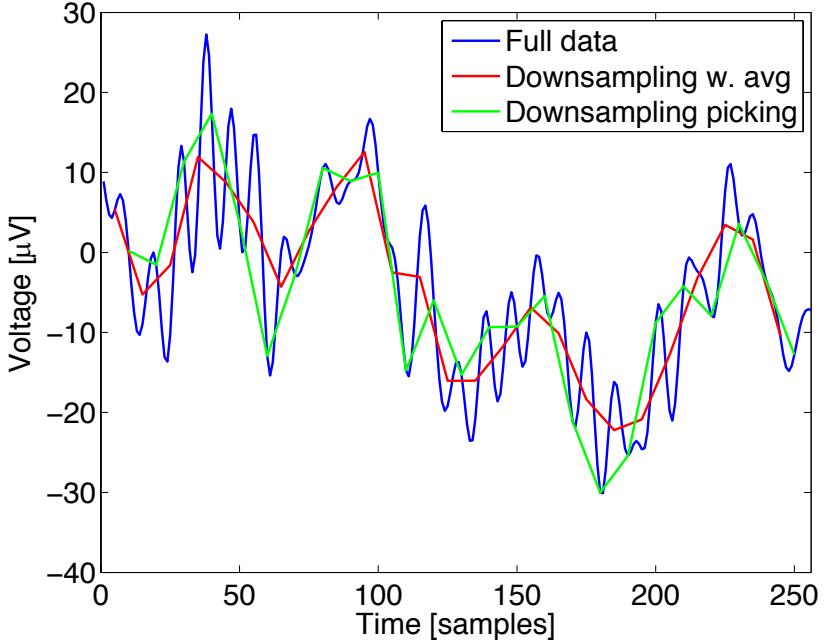
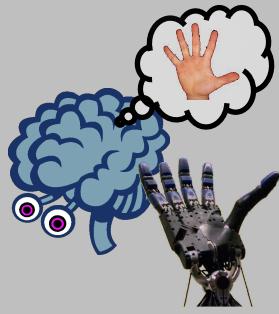


Rohdaten (0,1 Hz analoger HP)



Daten gefiltert: 0,5 - 40 Hz BP

Downsampling als einfaches ME Beispiel



Reduktionsfaktor $k = 10$.

$$\vec{x} = \begin{pmatrix} x_{1,1} & \dots & x_{1,k} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,k} & \dots & x_{m,n} \end{pmatrix}$$

$$\vec{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_m \end{pmatrix}$$

Transponieren und konatenieren aller erzeugten Vektoren ergibt den finalen Merkmalsvektor:

$$\vec{v} = (v_1 \dots v_m \dots v_N)$$

$$N = \left\lfloor \frac{n}{k} \right\rfloor \cdot m$$

Dimensionsreduktion

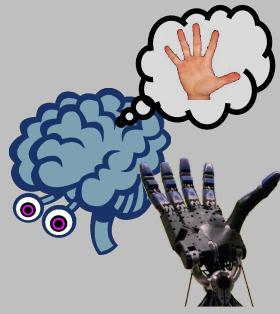
Dimension Eingabematrix

Dimension Merkmalsvektor

$m=10, n=256$ (Konkatenierter Vektor hätte $N = 2560$)

$N = 250$

Principal Component Analysis – PCA

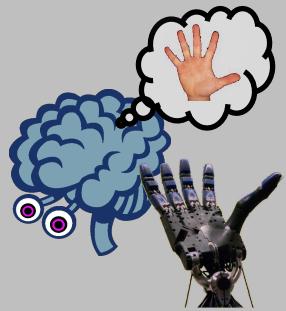


Andere Namen

Hauptkomponenten Analyse, Hauptachsentransformation, Karhunen-Loewe Transformation

Grundidee

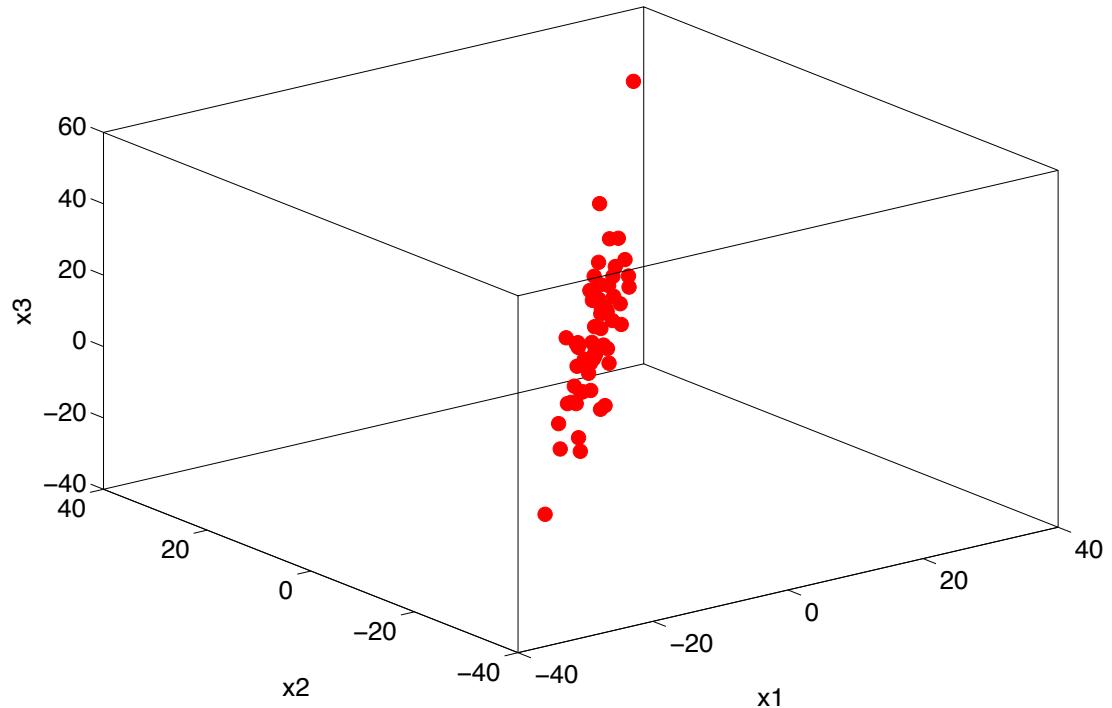
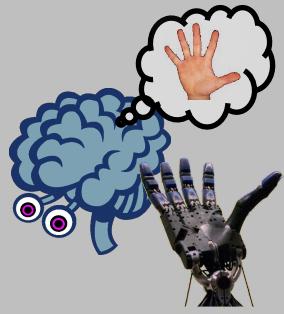
- Der Datenraum ist hochdimensional.
- Es soll nun ein niedriger dimensionierterer Unterraum gefunden werden, der bei so wenig Informationsverlust wie möglich die Daten darstellt.
- Projektion in ein neues Basissystem.
- Analyse-Kriterium: Varianz der Daten.
- Informell gesprochen, wird die Varianz mit dem Informationsgehalt gleichgesetzt.
- Die PCA ist linear.



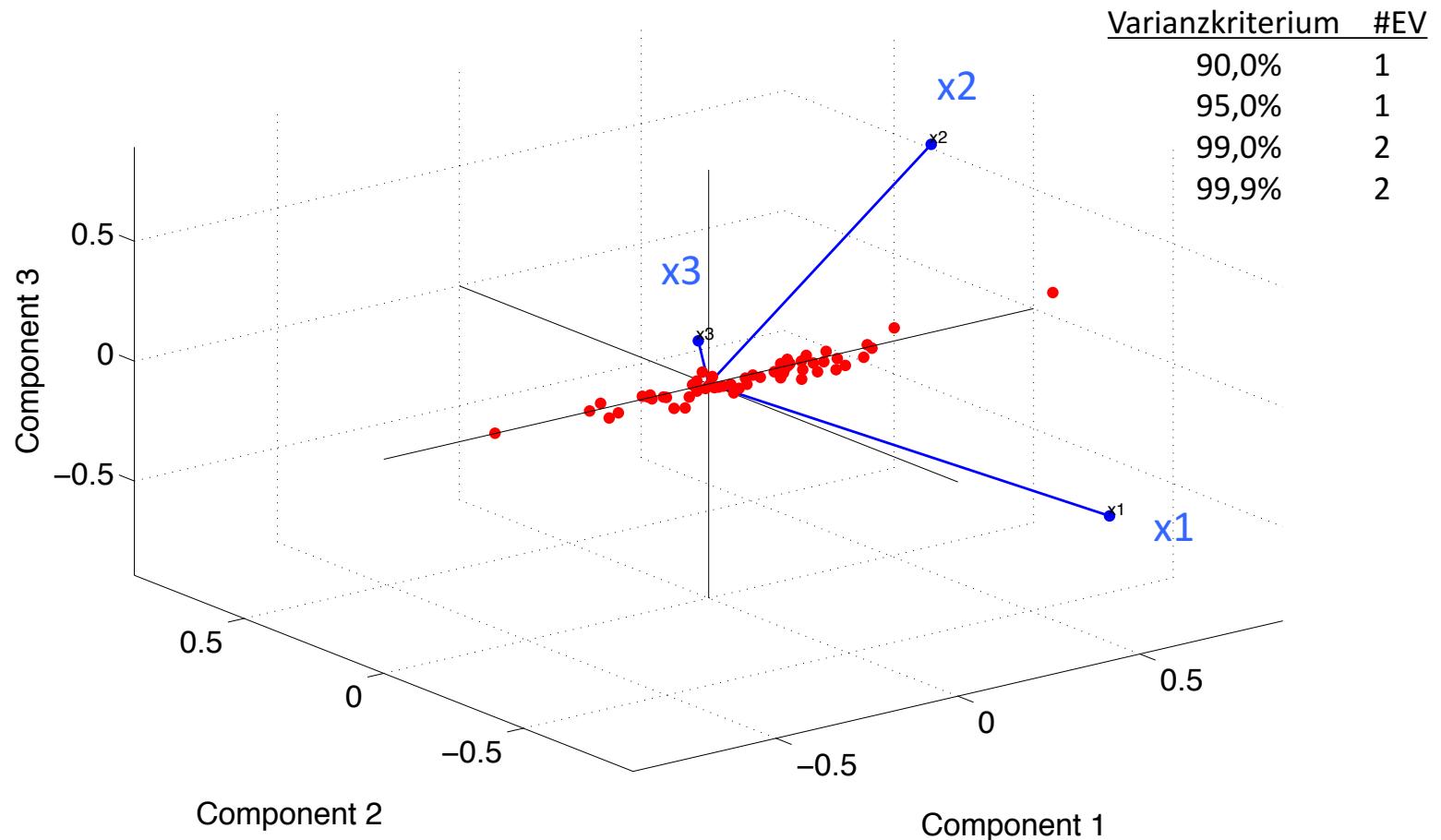
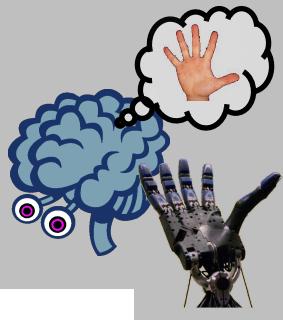
Die Formulierung der PCA

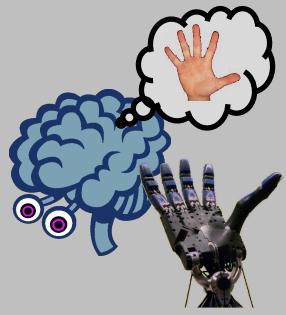
- Wir betrachten eine Menge von Eingabevektoren $\{\vec{x}_m\}$ mit $m=1,\dots,M$ und $\vec{x}_m \in \mathbb{R}^D$
$$\vec{x} = (x_{1,1} \dots x_{1,d} \dots x_{m,d}) \quad \begin{matrix} m & \text{Anzahl Kanäle} \\ d & \text{Anzahl Zeitpunkte} \end{matrix}$$
- Eingabedaten:
- Ziel ist die Projektion der Daten in einen Raum der Dimensionalität $N < D$ während gleichzeitig die Varianz der Daten maximiert wird.
- Dazu sollen die Daten als Linearkombination von Basisvektoren ausgedrückt werden.
- Es wird also eine Transformationsmatrix \vec{P}^T gesucht, die einen Eingabevektor auf die neue Repräsentation projiziert:
- Bedingungen:
 $P^{-1} = P^T$
 $\text{cov}(\vec{y})$ ist die Kovarianzmatrix (Diagonalmatrix).
 $\langle \cdot \rangle$ ist der Erwartungswert.

PCA – die Eingabedaten



PCA – die Transformation





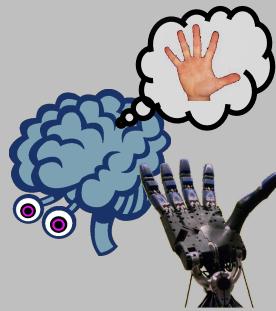
Die Formulierung der PCA

Nun kann dieses folgendermaßen geschrieben werden:

$$\begin{aligned}\text{cov}(\vec{y}) &= \langle \vec{y} \vec{y}^T \rangle \\ &= \left\langle (P^T \vec{x}) (P^T \vec{x})^T \right\rangle \\ &= \langle (P^T \vec{x}) (\vec{x}^T P) \rangle \\ &= P^T \langle \vec{x} \vec{x}^T \rangle P \\ &= P^T \text{cov}(\vec{x}) P\end{aligned}$$

So dass man erhält:

$$\begin{aligned}P \text{cov}(\vec{y}) &= P P^T \text{cov}(\vec{x}) P \\ &= \text{cov}(\vec{x}) P\end{aligned}$$



Die Formulierung der PCA

P wird nun als $d \times d$ Spaltenvektoren notiert $[p_1, p_2, \dots, p_d]$ sowie die

Kovarianzmatrix $\text{cov}(\vec{y})$ als

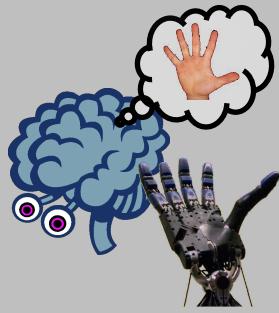
$$\begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_d \end{pmatrix}$$

Nun kann man $P \text{cov}(\vec{y}) = \text{cov}(\vec{x}) P$ schreiben als

$$[\lambda_1 p_1, \lambda_2 p_2, \dots, \lambda_d p_d] = [\text{cov}(\vec{x}) p_1, \text{cov}(\vec{x}) p_2, \dots, \text{cov}(\vec{x}) p_d]$$

Dabei sind die p_i die **Eigenvektoren** und λ_i die **Eigenwerte** der Kovarianzmatrix.
Die p_i sind nun die Hauptkomponenten (Principal Components)

Die Formulierung der PCA



Fazit

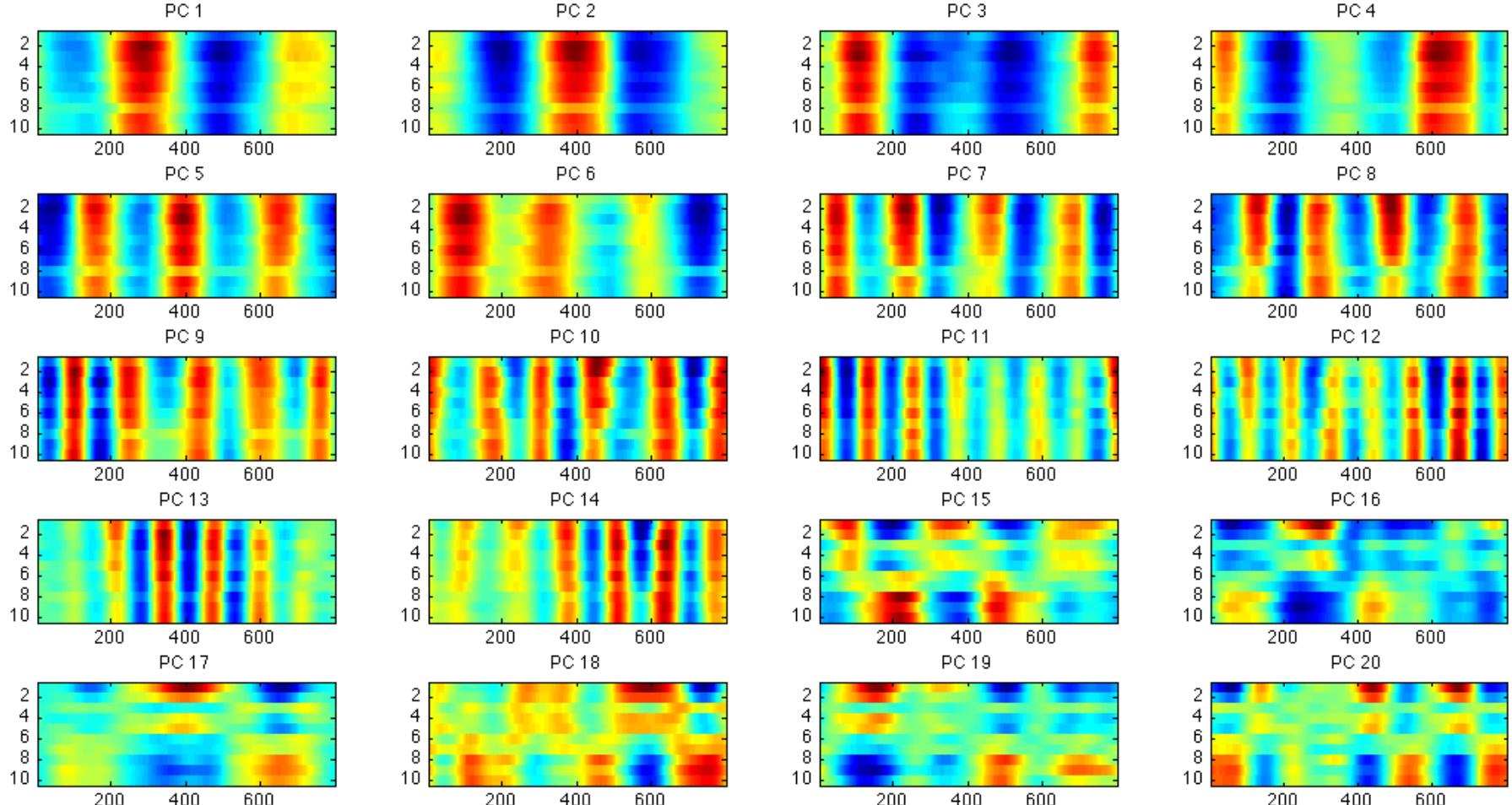
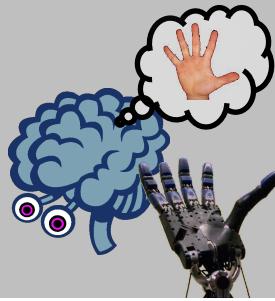
- Die Projektionsmatrix P ergibt sich also aus der Berechnung der Eigenvektoren und Eigenwerte der Kovarianzmatrix.
- Die Eigenvektoren bilden die Basis des gesuchten Unterraums.
- Die Eigenwerte werden nach Größe sortiert

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \geq 0$$

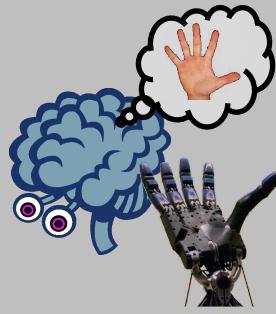
- Der Anteil der Varianz die erhalten bleibt, berechnet sich nach
- Die PCA selbst ist nur auf einzelnen Eingabevektoren definiert.
- In der Praxis wird daher die Verallgemeinerung der PCA auf Eingabematrizen, die Singular Value Decomposition, verwendet.
- Z.B. implementiert Matlab die SVD in der Funktion `princomp`.

$$\frac{\sum_{j=1}^k \lambda_j}{\sum_{i=1}^d \lambda_i}, \quad k \leq j$$

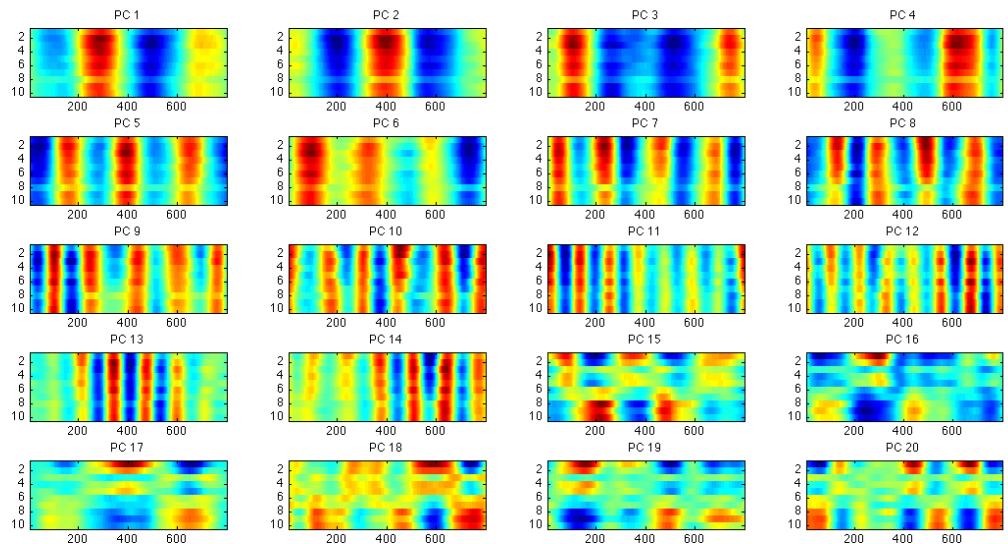
Die Bedeutung der PCA auf EEG Daten



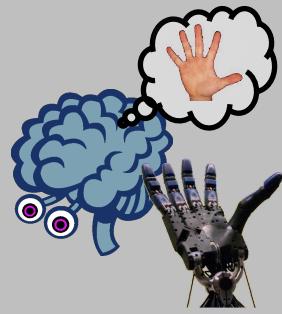
Die Bedeutung der PCA auf EEG Daten



- Die PCA kann als **tempo-spatialer** Filter aufgefasst werden.
- In der Praxis scheint sie insbesondere Frequenzeigenschaften zu erfassen.
- Mit zunehmendem Komponentenindex werden höhere Frequenzen repräsentiert.
- **Fallstrick**
Die PCA ist rein varianz-getrieben.
Daher ist die funktionale Interpretation schwierig.



Common Spatial Pattern – CSP

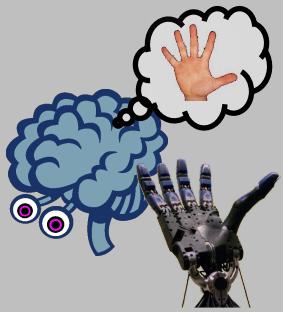


Verwandtschaft zur PCA

- Die CSP ist ebenfalls ein lineares Verfahren, bei dem eine Projektionsmatrix gesucht wird.
- Die PCA berücksichtigt nicht die Zuordnung zu Klassen, sondern betrachtet den Datensatz als Gesamtheit. Sie ist somit losgelöst von der nachfolgenden Klassifikationsaufgabe.
- Die CSP hingegen bekommt als Eingabe 2 (oder im verallgemeinerten Fall mehrere) Datenmatrizen, die jeweils nur die Daten einer Klasse enthalten.

Eigenschaften der CSP

- Die CSP ist ein **spatialer**, also räumlicher **Filter**. Anders als die vorherigen Filter ist das Verfahren **datengetrieben**.
- Ziel der Basistransformation ist die **Projektion der Eingabedaten in einen Unterraum bei gleichzeitiger Maximierung der Diskrimierbarkeit der Klassen**.



Formulierung der CSP

- Eingabedaten der Form:

$$\vec{x} = \begin{pmatrix} x_{1,1} & \dots & x_{1,d} \\ & \ddots & \\ x_{m,1} & \dots & x_{m,d} \end{pmatrix}$$

d Anzahl Kanäle
m Anzahl Zeitpunkte

- Berechnung der Klassen-Kovarianzmatrizen:

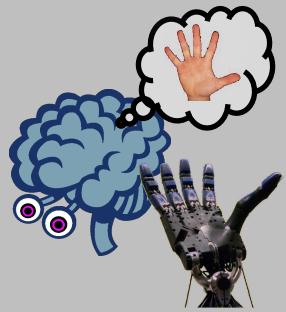
- Mittelung über alle $\{\vec{x}_i\}_{C_1}$ und $\{\vec{x}_i\}_{C_2}$, Kovarianzmatrizen C_1 und C_2 werden auf diesen gemittelten Matrizen berechnet.

- Verbund-Kovarianzmatrix: $C = C_1 + C_2$

- Faktorisierung in Eigenwerte und Eigenvektoren wie bei der PCA:

$$C = V \lambda V^T$$

Mit V Matrix der Eigenvektoren und λ Diagonalmatrix der Eigenwerte.



Formulierung der CSP

- Normalisieren der Eigenwerte von V zu 1 mittels
(s.g. Whitening-Transformation)
- Zerlegen von C_1 und C_2 in

und

$$S_1 = PC_1P^T$$

$$P = \sqrt{\lambda^{-1}}V^T$$

- Dann haben C_1 und C_2 **gemeinsame** Eigenvektoren:

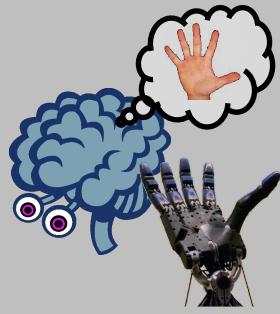
$$S_1 = B\lambda_1 B^T \quad \text{und} \quad S_2 = B\lambda_2 B^T \quad \text{mit} \quad \lambda_1 + \lambda_2 = \mathbb{I}$$

- Aufgrund letzterer Bedingung ist der größte EV von S_1 der kleinste EV von S_2 .

$$W = (B^T P)^T$$

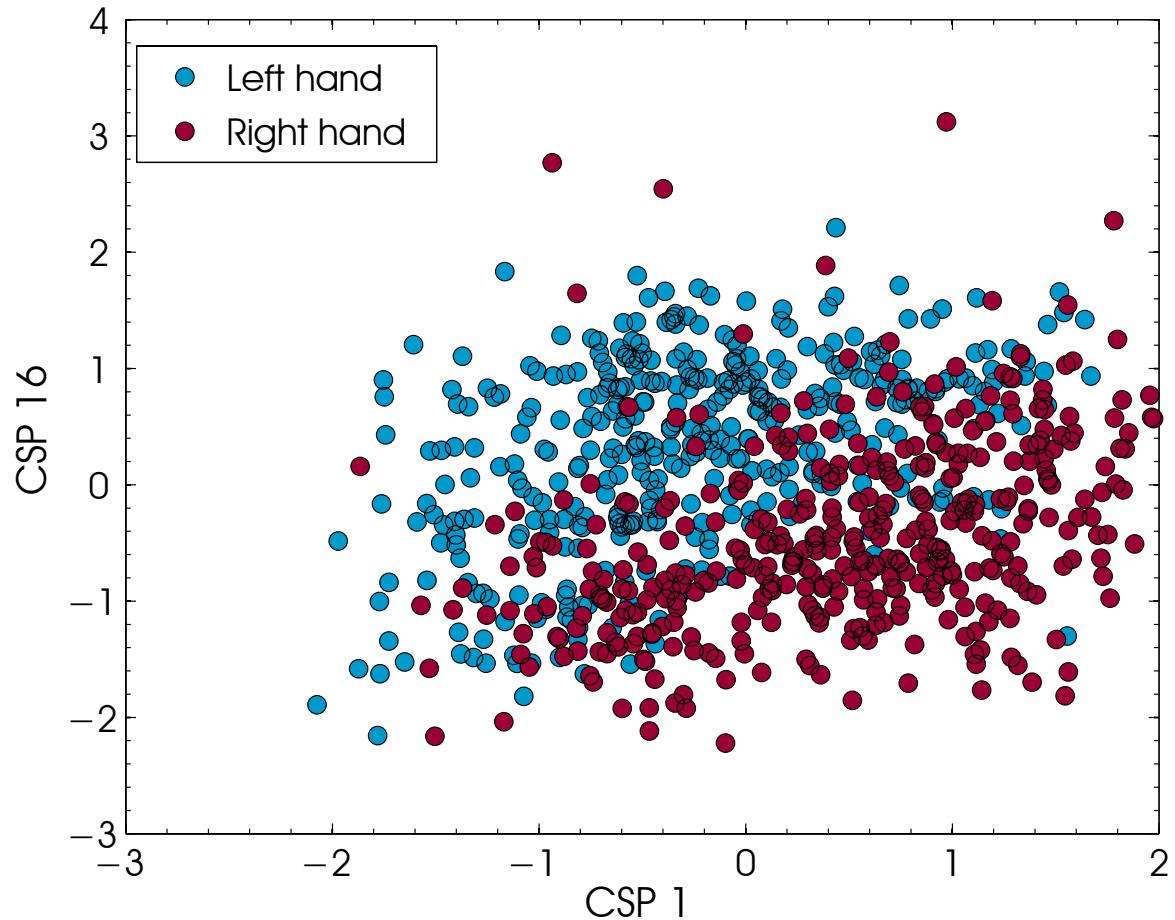
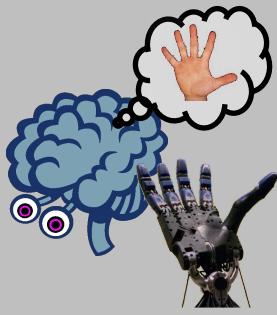
- Projektionsmatrix

Bemerkungen CSP

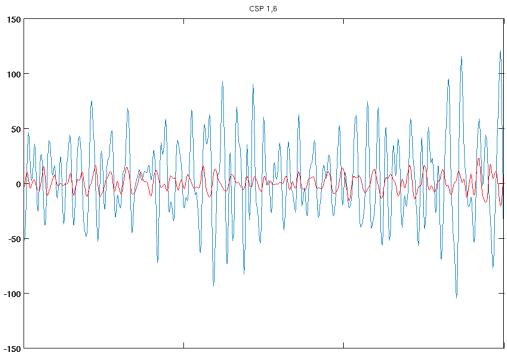
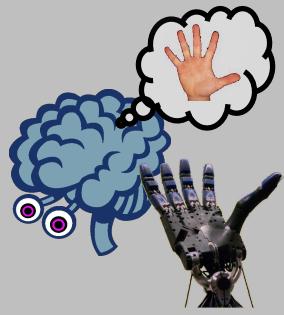


- Alternative Formulierung $S_1 \mathbf{B} = \lambda S_2 \mathbf{B}$ → Generalisiertes Eigenwertproblem, wird z.B. in der Matlab-Funktion
`[evec,eval] = eig (S1, S2)` verwendet.
- Eigenvektoren: Werden nun wie bei der PCA nach Größe sortiert. Allerdings werden **die ersten und die letzten** für die Projektion verwendet.
- CSP eignen sich besonders für gut lokalisierte Signale.
- CSP finden bei sehr vielen Studien Anwendung, die Signale vom motorischen Kortex verwenden, z.B. Vorstellung von Bewegung.
- Sie sind wie die PCA varianzgetrieben, allerdings durch ihre Eigenschaften z.T. gut neurophysiologisch interpretierbar.

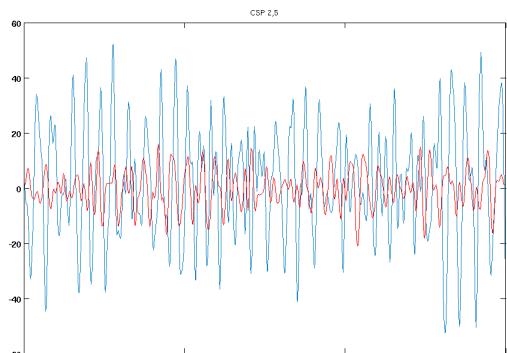
Anwendungsbeispiel CSP – Motor Imagery Daten



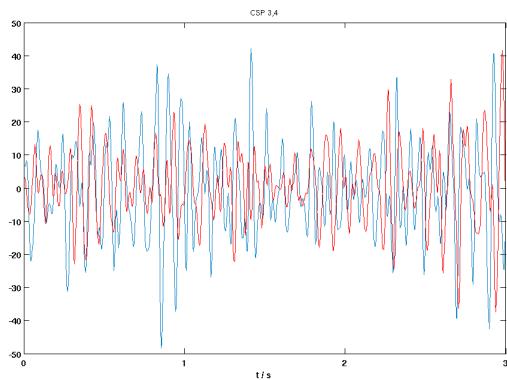
Anwendungsbeispiel CSP



EV 1 und 16



EV 2 und 15

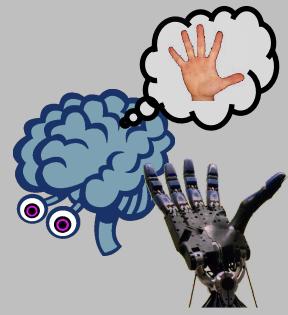


EV 3 und 14

Gefiltertes Signal, mit jeweils unterschiedlichen EV projiziert (insgesamt 16 Kanäle = 16 EV).

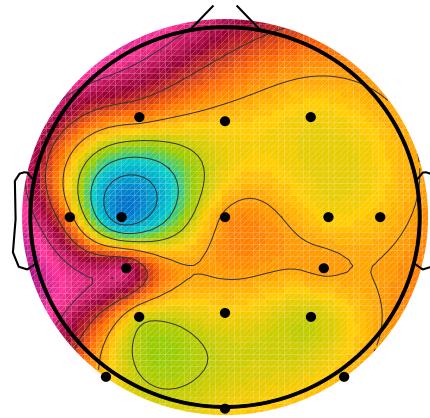
en

Anwendungsbeispiel CSP (Motor Imagery Daten)

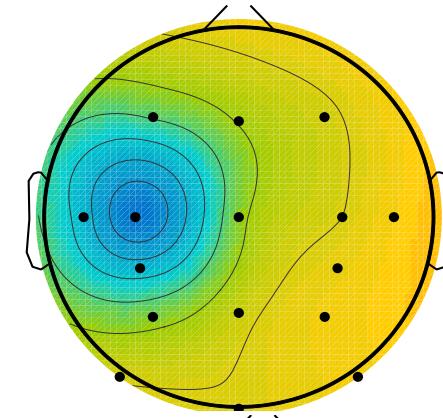


Filter (Gewichtsmatrix W)

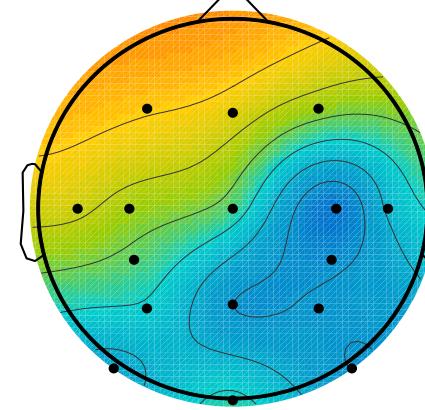
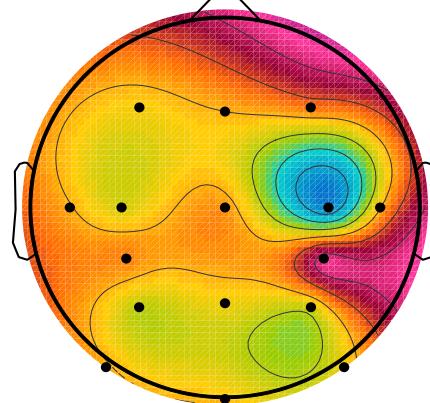
Rechte
Hand
(EV 1)



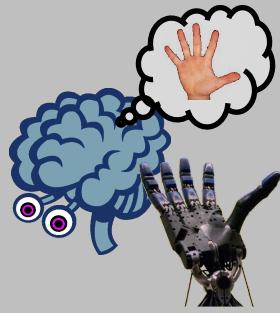
Pattern (inverse Matrix W^{-1})



Linke
Hand
(EV 16)

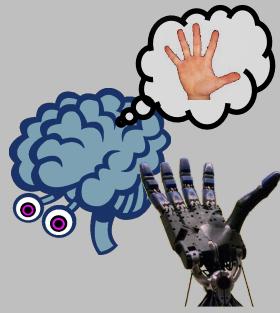


Training eines Klassifikators

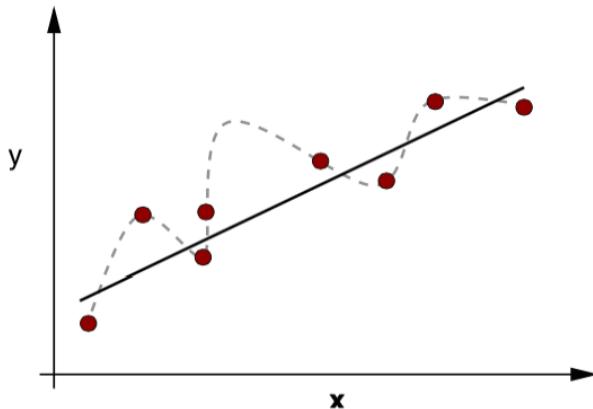


- Wir gehen von einem **überwacht lernenden Klassifikator** aus, d.h. wir benötigen einen Datensatz X , bei dem die Klassenzugehörigkeiten, d.h. die **Label**, bekannt sind.
- Ganz allgemein ist ein Klassifikator eine Abbildung $\Phi : X \mapsto C$.
- Dabei ist X eine Menge an Merkmalsvektoren $\{\vec{v}_i\}$.
- C ist eine Menge von diskreten Klassen $\{c_j\}$, also $C = \{c_0, c_1, \dots, c_{k-1}\}$, wobei k die Anzahl der Klassen ist.
- Im einfachsten Fall haben wir eine **binäre Klassifikation** mit nur **2 Klassen** (Allerdings lässt sich jede Mehrklassen-Aufgabe auf $k-1$ 2-Klassen Aufgaben zurückführen).
- Die Zuordnung der Vektoren $\{\vec{v}_i\}$ zu jeweils einer der Klassen c_j ist durch die Label gegeben.

Komplexität und Overfitting



- Beim Training des Klassifikators geht es nicht darum, diejenige Funktion zu finden, die die Trainingsdaten möglichst perfekt abbildet.
- Es geht darum, diejenige Funktion zu finden, die am Besten auf neuen, ungesehenen Daten **generalisiert**.

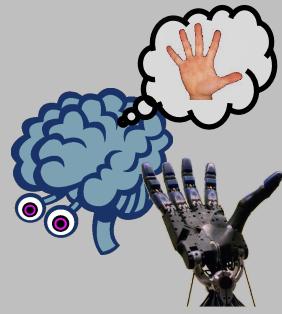


- Beispiel: 2 Regressions-Funktionen approximieren die gegebenen Datenpunkte mit unterschiedlicher Komplexität ab.
- Grundprinzip: „Occam’s Razor“
→ Ziehe die einfachere Theorie vor.

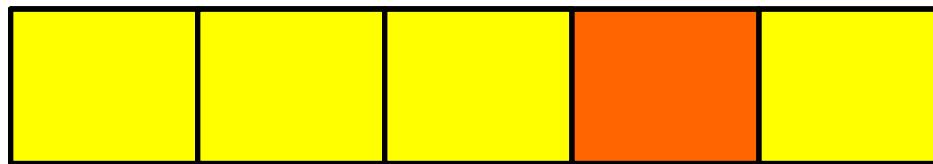
— — — — Funktion folgt exakt den einzelnen Datenpunkten (Overfitting).

— — — Funktion folgt nur dem generellen Trend der Daten.

Abschätzen der Generalisierung im Training

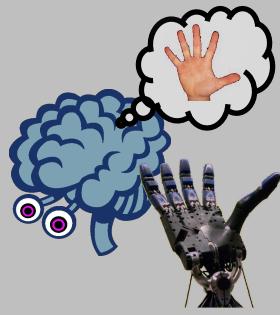


- Trainings- und Testmenge müssen strikt disjunkt sein!
- Es ist immer eine n-fache Kreuzvalidierung zur Abschätzung der Generalisierungsleistung durchzuführen.
- Die Parameter des Klassifikators werden innerhalb der Kreuzvalidierung bestimmt.

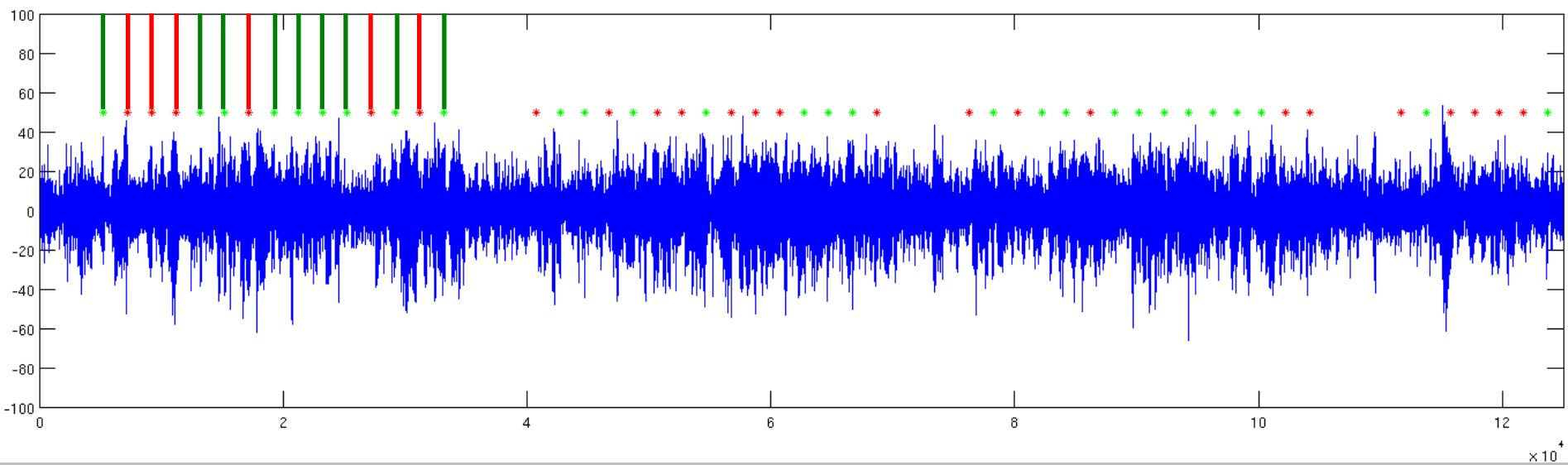


- Wird eine Projektionsmatrix verwendet (z.b. aus PCA,CSP,...) ist auch diese nur auf dem Trainingsdaten zu berechnen und auf die Testdaten anzuwenden.

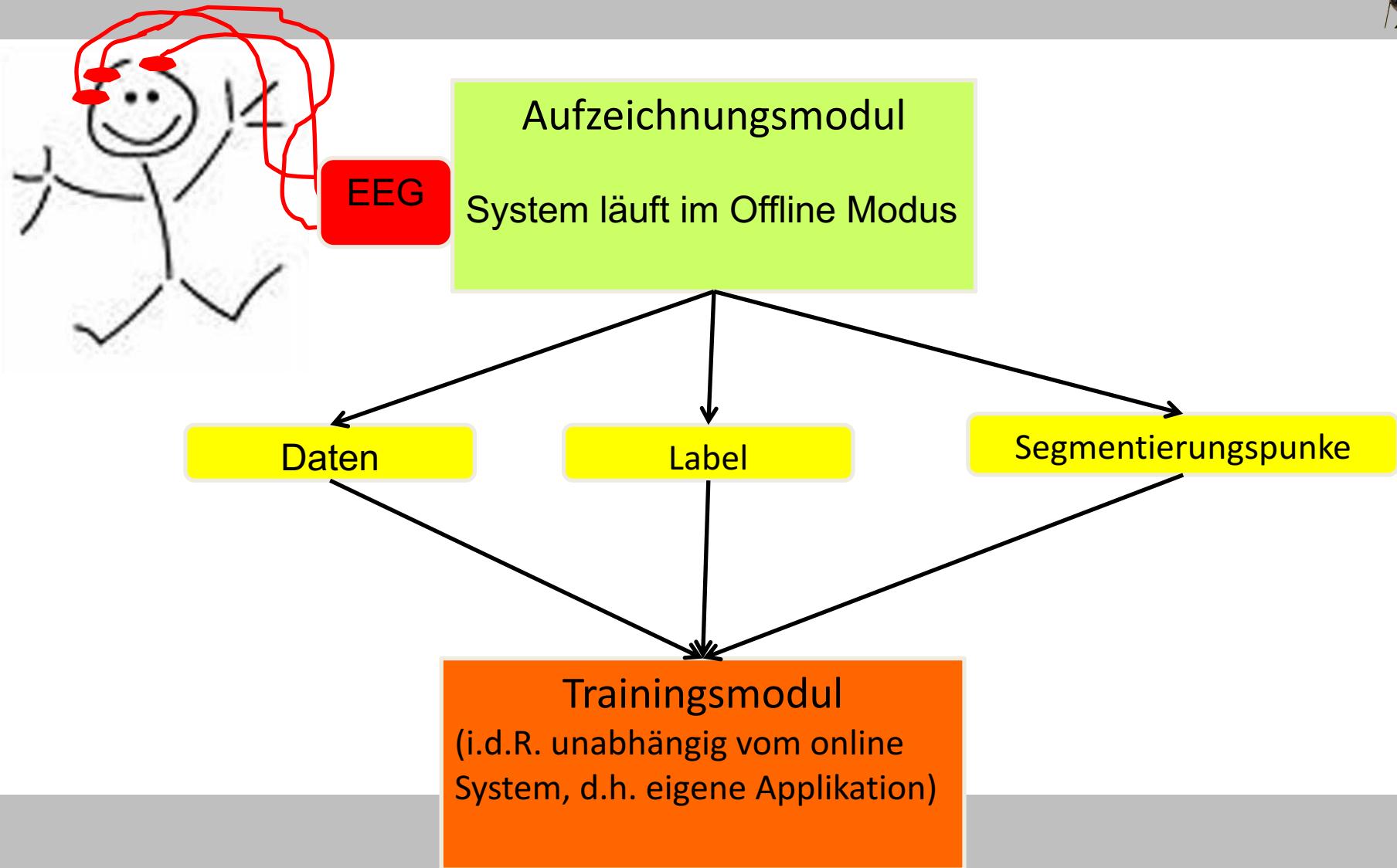
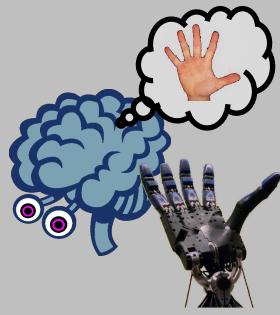
BMI Trainingsmodus



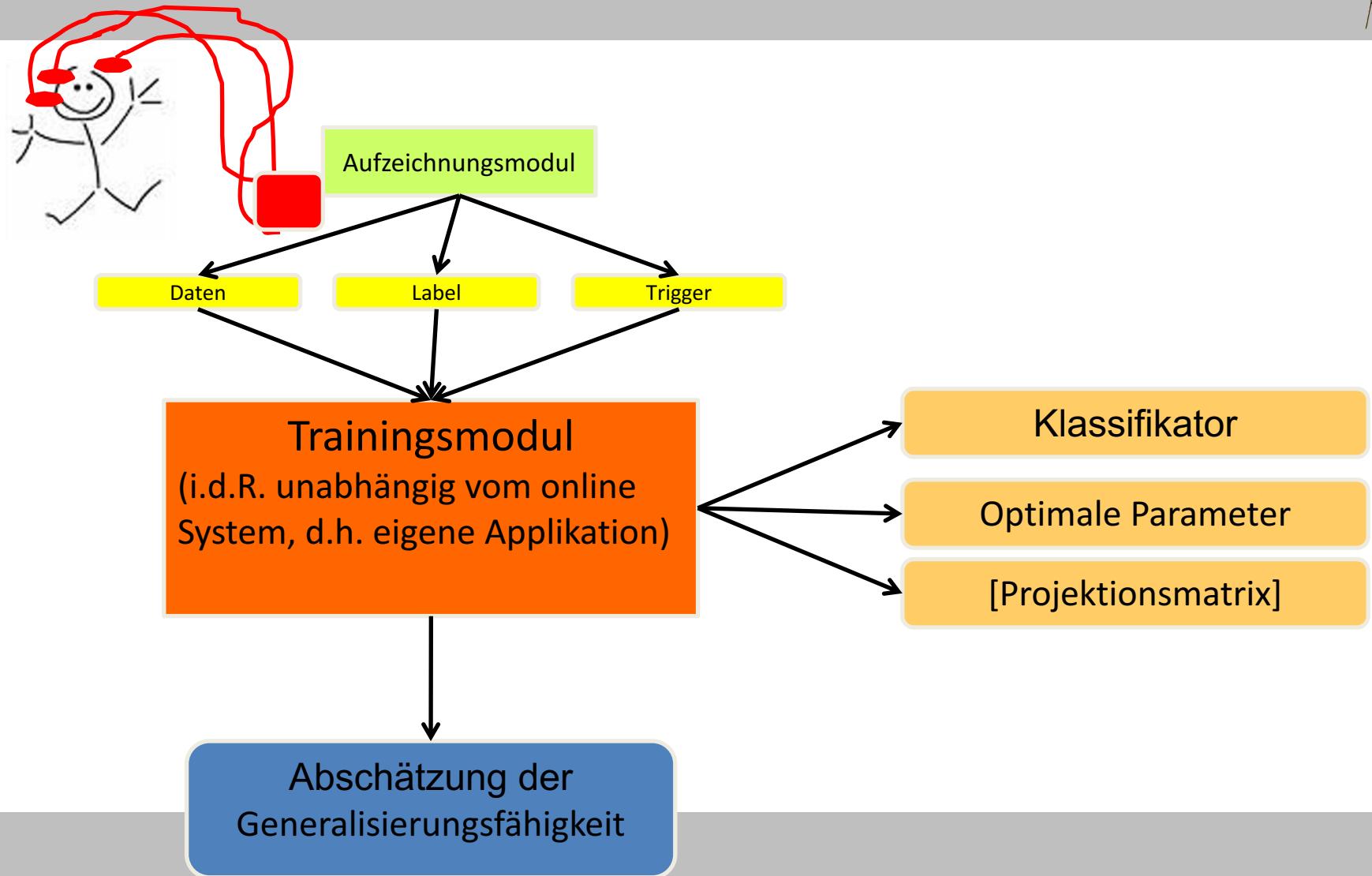
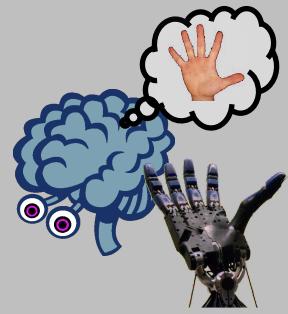
Aufzeichnung von Trainings-Daten
System läuft im Offline Modus



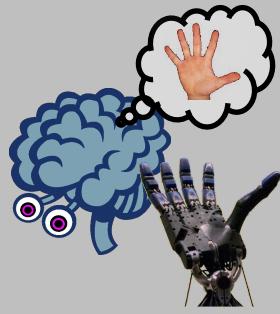
BMI Trainingsmodus



BMI Trainingsmodus

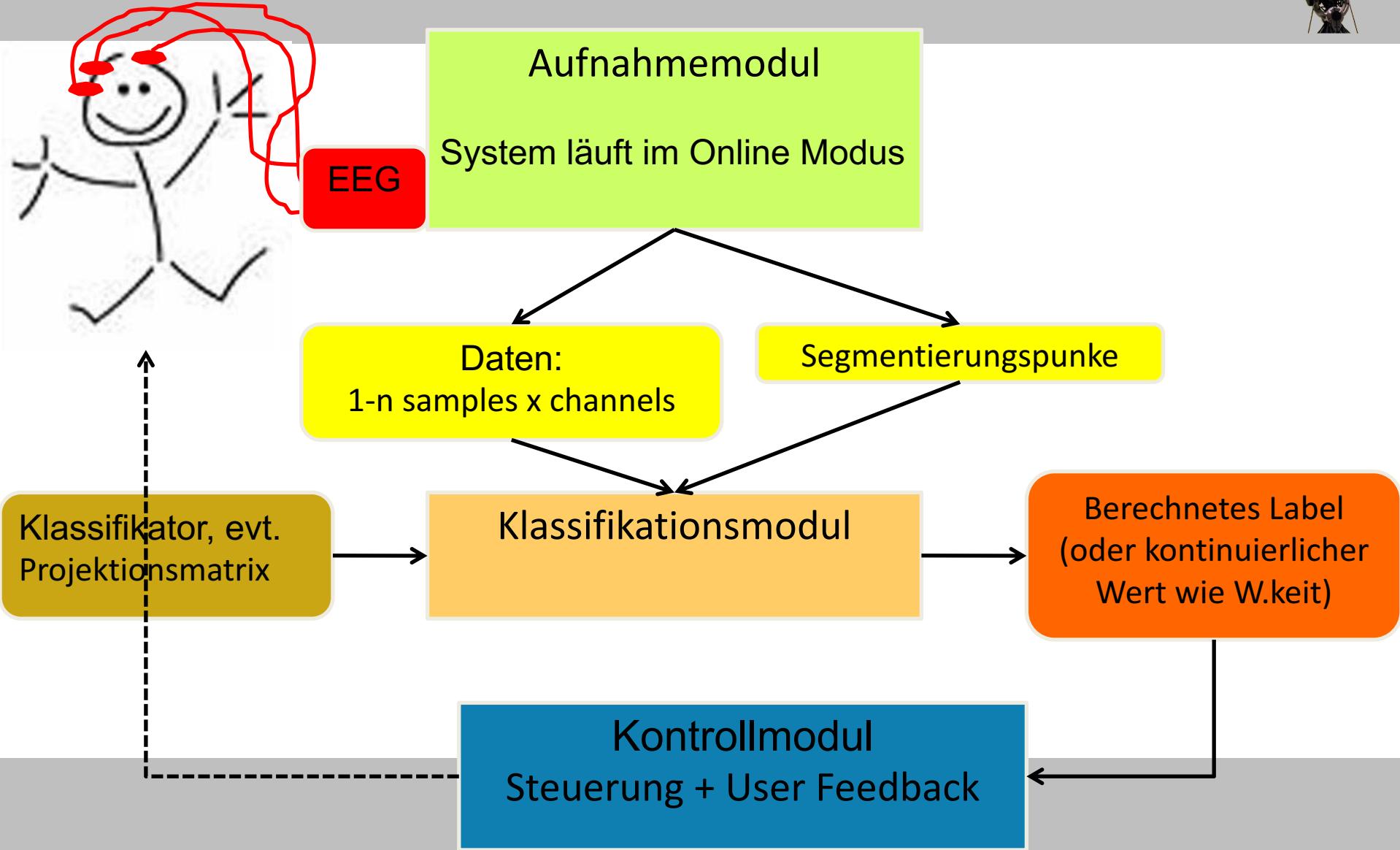
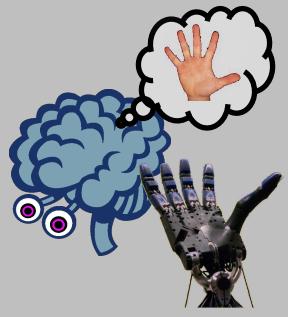


Training - Schema

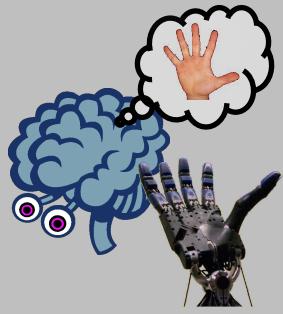


```
1: Init parameters:  
    highpass, lowpass, epoch length, #foldings.  
2: Load data, label, trigger.  
3: Bandpass data channel wise.  
4: Segment data according to triggers .  
5: Create epochs of specified length.  
6: Sort epochs according to labels.  
[7: Balance data set: Use as many epochs from one class as from the  
other class.]  
8: Create sets for cross-validation (balanced for classes).  
9: For i = 1 to #foldings do  
10: select all sets excluding set i and merge to trainset.  
[11: Preprocess trainset, obtain Tranformation matrix W].  
[12: Apply W to trainset.  
[13: Apply W to set i (testset) .  
14: Train classifier Phi with trainset.  
15: Classify testset using Phi.  
16: end for  
17: Calculate mean classification rate.
```

Online System



Online System (Test) - Schema

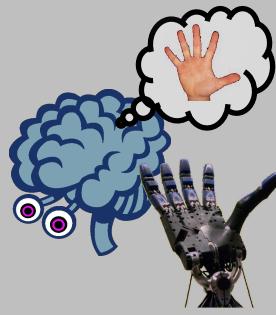


Algorithm 1 Online classification and feedback generation.

- 1: Initialize $t \leftarrow 0$, $w \leftarrow W$ {W is the window size parameter}
- 2: Load pre-trained classifier Φ
- 3: Buffer data until $t=w$
- 4: **for** $t = w+1$ to T **do**
- 5: Create vector $\vec{y}_t = (y_t^1, \dots, y_t^n)^T$
- 6: Create window matrix $\vec{x}_t = (\vec{y}_{t-W+1}, \dots, \vec{y}_t)$
- 7: Preprocess \vec{x}_t
- 8: Feature extraction: obtain $\hat{\vec{x}}_t$
- 9: Classify $\hat{\vec{x}}_t$
- 10: **if** $\Phi(\hat{\vec{x}}_t) > \theta$ **then**
- 11: Generate feedback
- 12: **end if**
- 13: $t \leftarrow t + 1$
- 14: **end for**

Ein einfacher Klassifikator:

Fisher's Linear Discriminant Analysis (FDA/ LDA)



Ziel der Klassifikation beim **Testen**, also im Online-Lauf des Systems:

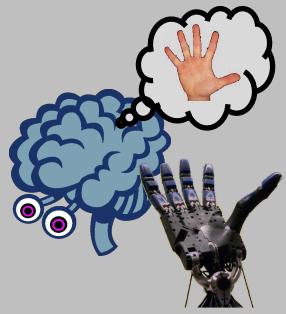
$$y = \vec{w} \cdot \vec{x}_{test} \quad \text{mit} \quad y \in \mathbb{R}$$

Skalare Ausgabe des Klassifikators Gewichtsvektor Eingabevektor

Daraus können folgendermaßen Label berechnet werden:

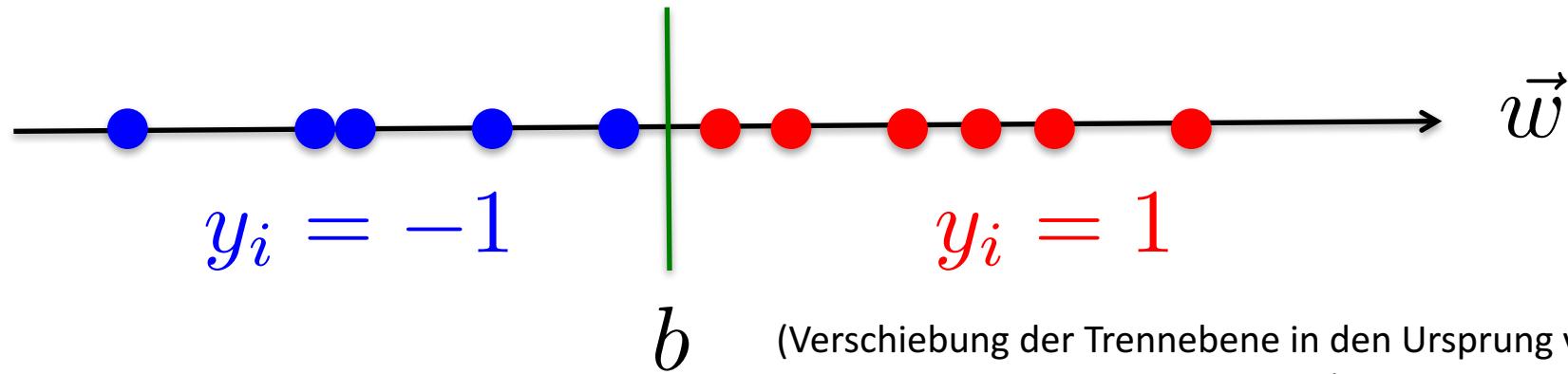
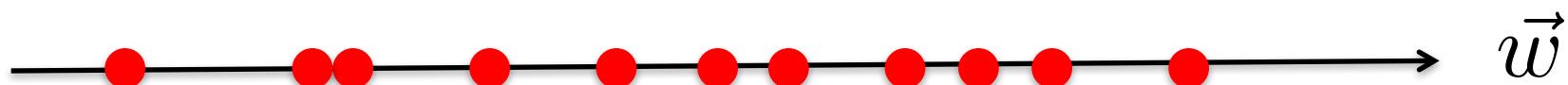
$$y = \text{sgn}(\vec{w} \cdot \vec{x}_{test} + b) \quad \text{mit} \quad y \in \{-1, 1\}$$

Bias (Abstand der Hyperebene zum Ursprung)



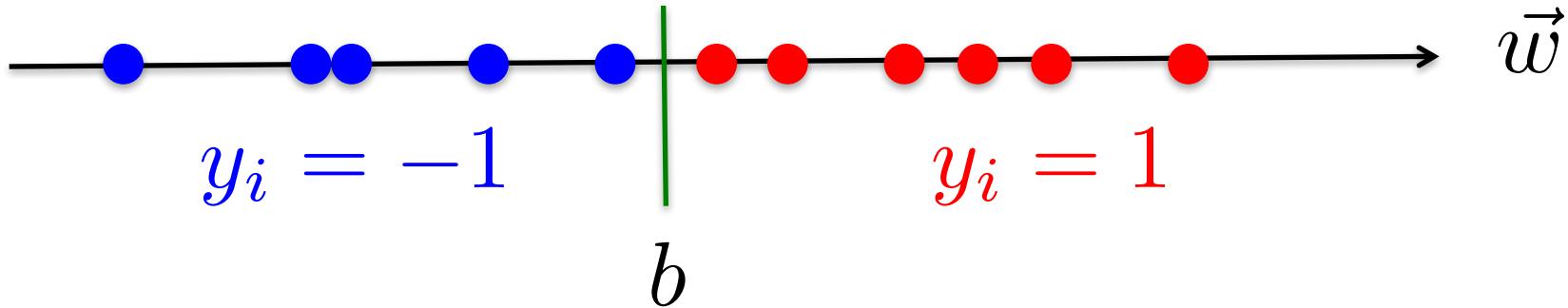
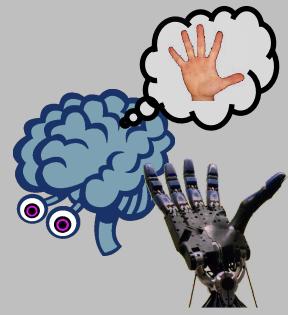
Die Klassifikatorausgabe graphisch

Die Ausgabe y ist die Projektion des Testvektors auf den Gewichtsvektor.

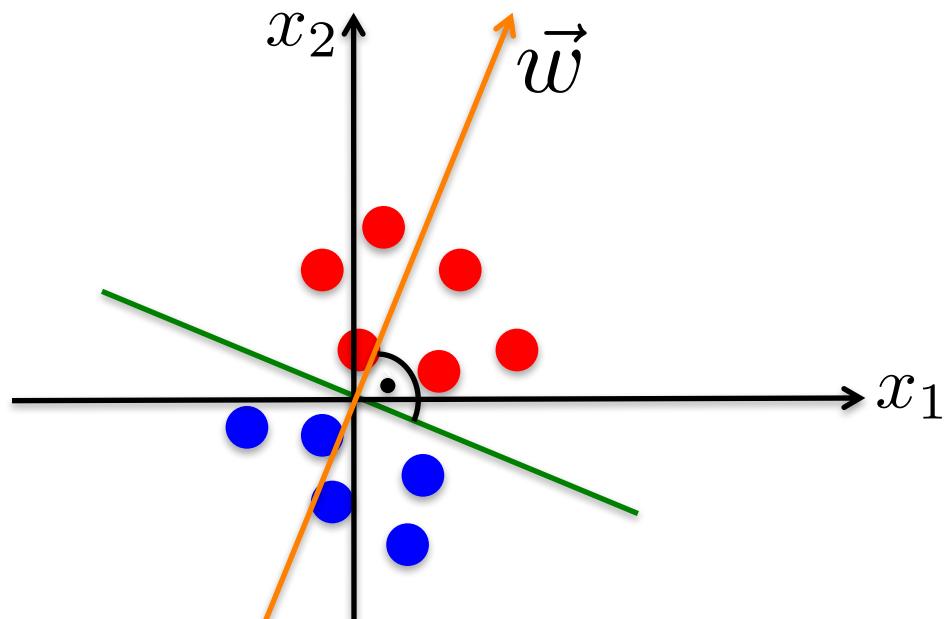
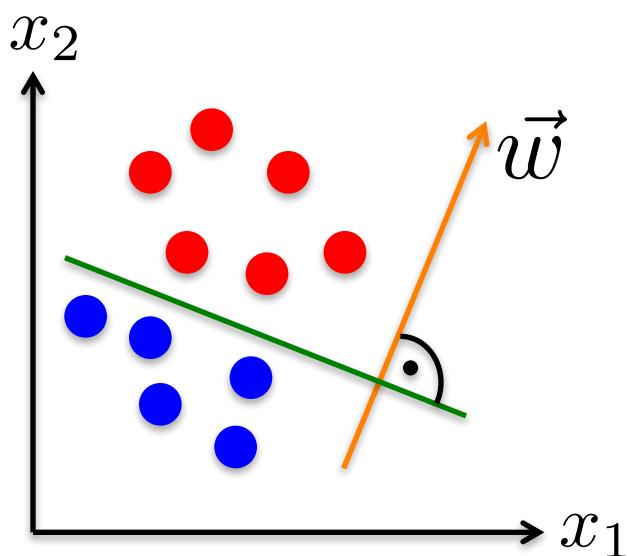


(Verschiebung der Trennebene in den Ursprung vor Anwendung der signum Funktion.)

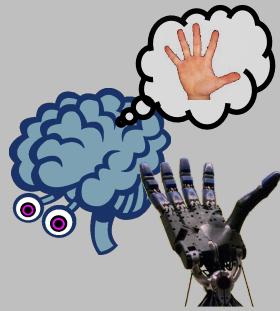
Die Klassifikatorausgabe graphisch



Im Datenraum mit $\vec{x} \in \mathbb{R}^2$

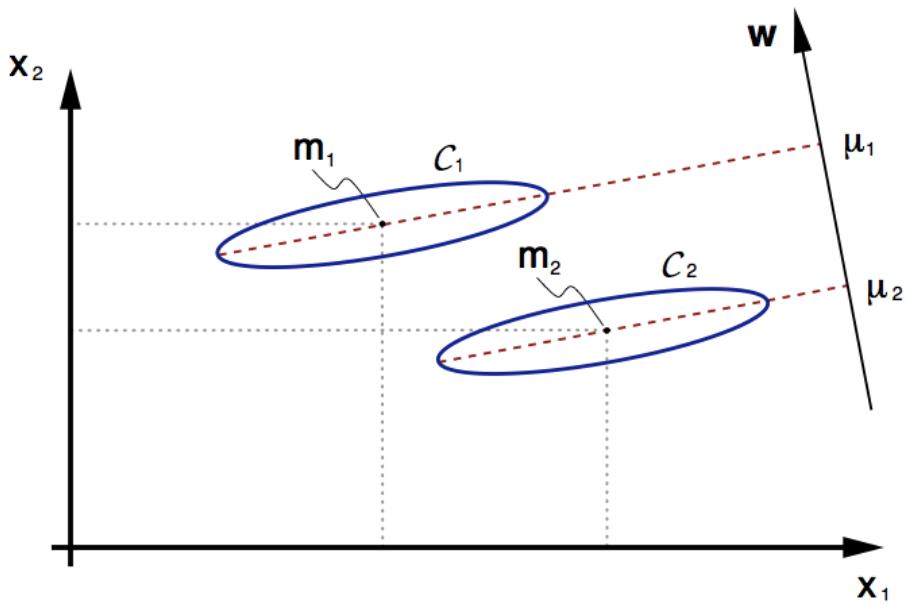
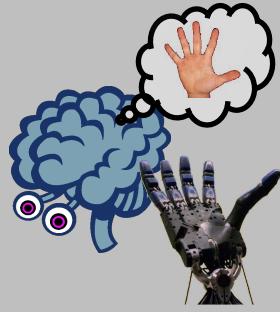


FDA – Eigenschaften



- Der Klassifikator ist durch den Gewichtsvektor \vec{w} und den Bias b eindeutig bestimmt.
- Die Hyperebene (im Datenraum) steht immer orthogonal auf dem Gewichtsvektor.
- Die Projektion auf den Gewichtsvektor entspricht einer Dimensionsreduktion von $\mathbb{R}^n \mapsto \mathbb{R}$.
- Die Parameter \vec{w} und b werden auf den Trainingsdaten gelernt.
- Der Betrag der Klassifikatorausgabe $|y|$ kann als Maß für die “Sicherheit” der Klassifikation verstanden werden (je größer, desto sicherer - informell gesprochen). Jedoch entspricht dies nicht einer Interpretation als Wahrscheinlichkeit. Dazu ist ein probabilistischer Klassifikator notwendig (dazu später mehr).

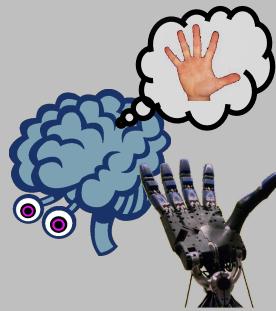
FDA Optimierungskriterien



Frage: Unter welchen Bedingungen wird die Separation der Klassen und somit die Klassifikationsleistung maximiert?

Die FDA hat 2 komplementäre Optimierungskriterien:

- (1) Die Mittelwerte der beiden Klassen haben in der Projektion eine möglichst große Distanz.
- (2) Die Varianz innerhalb einer Klasse (*within-class scatter*) ist **klein** und gleichzeitig ist die Varianz zwischen den Klassen (*between-class scatter*) **groß**.
(in Richtung des Gewichtsvektors)



Berechnung

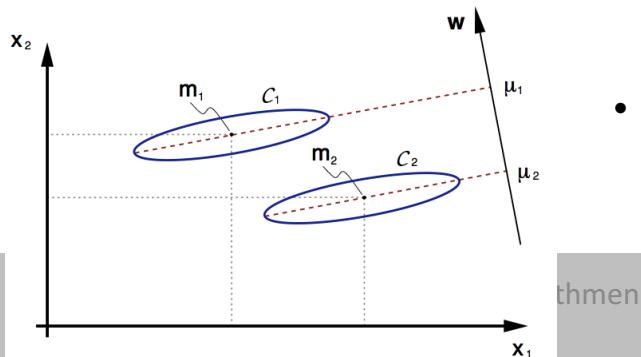
- Projektion der Mittelwerte auf den Gewichtsvektor und Berechnung der Varianz:

$$\mu_i = \vec{w}^\top \vec{m}_i \quad \sigma_i = \sum_{n \in C_i} (y_n - \mu_i)^2$$
- Berechnung der Klassenkovarianz (*scatter matrix*) je Klasse:

$$\mathbf{S}_i = \sum_{x \in C_i} (\vec{x} - \vec{m}_i)(\vec{x} - \vec{m}_i)^\top$$
- Daraus ergibt sich die gesamte Intra-Klassenkovarianz (*within-class scatter*): $\mathbf{S}_w = \mathbf{S}_1 + \mathbf{S}_2$
- Die Inter-Klassenkovarianz (*between-class scatter*) ergibt sich aus: $\mathbf{S}_b = (\vec{m}_1 - \vec{m}_2)(\vec{m}_1 - \vec{m}_2)^\top$
- Die Optimierungskriterien werden mit dem *Fisher-Kriterium* ausgedrückt. Dieses lässt sich in der Projektion definieren als

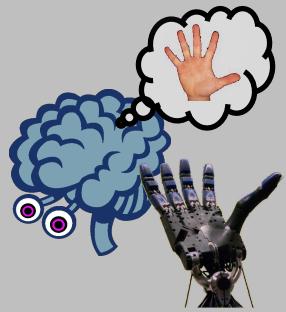
$$J(\vec{w}) = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

$$J(\vec{w}) = \frac{\vec{w}^\top \mathbf{S}_b \vec{w}}{\vec{w}^\top \mathbf{S}_w \vec{w}}$$



- Das Lernziel ist die Maximierung von $J(\vec{w})$

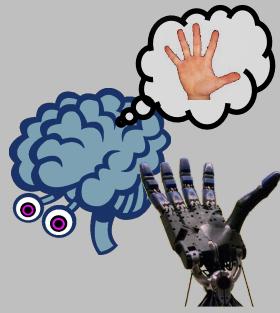
Wird auch als *Rayleigh Koeffizient* bezeichnet.



Berechnung

- Setzen von $\frac{\delta J(\vec{w})}{\delta \vec{w}} = 0$ ergibt, dass $(\vec{w}^\top \mathbf{S}_b \vec{w}) \mathbf{S}_w \vec{w} = (\vec{w}^\top \mathbf{S}_w \vec{w}) \mathbf{S}_b \vec{w}$ den Koeffizienten maximiert.
- Nach Vereinfachung ergibt sich, dass $\vec{w} \propto \mathbf{S}_w^{-1} (\vec{m}_1 - \vec{m}_2)$.
- Dieses lässt sich einfach auf den Trainingsdaten berechnen.
- Der Bias ergibt sich analytisch als $b = \frac{\mu_1 - \mu_2}{2}$. Diese Berechnung ist nur für normalverteilte Daten zulässig (eine Annahme, die sehr häufig getroffen wird). Ansonsten kann b auch in der Kreuzvalidierung ermittelt werden.

Beispielcode (Matlab)



Training

```
function fda = fda_train (data, label)

posdata = data(label==1,:);
negdata = data(label~=1,:);

posmean = mean (posdata);
negmean = mean (negdata);

Spos = cov(posdata);
Sneg = cov(negdata);

Sw = Spos + Sneg;

fda.w = Sw\ (posmean - negmean)';
%Sw\... = inv(Sw)*...

fda.posmean = fda.w' * posmean';
fda.negmean = fda.w' * negmean';
fda.b = (fda.negmean+fda.posmean)/2;
```

Test

```
function [label, proj] = fda_test (fda, data)

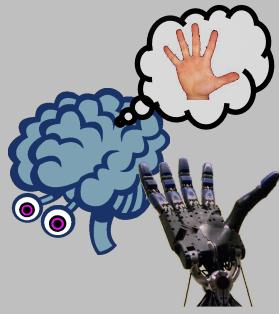
[rows, cols] = size(data);
proj = zeros(rows, 1);

for i=1:rows
    proj(i) = fda.w' * data(i,:)';
end

bias = fda.b * ones(rows,1);

label = sign(proj - bias);
```

Die Verwandtschaft von FDA und CSP



Die CSP lässt sich analog zur FDA formulieren, und zwar mit

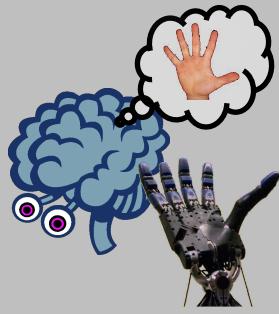
- *between-class scatter matrix* $\mathbf{S}_b = (\vec{m}_1 - \vec{m}_2)(\vec{m}_1 - \vec{m}_2)^\top$
- *within-class scatter matrix* $\mathbf{S}_w = \sum_{i \in C_1} (\vec{x}_i - \vec{m}_1)(\vec{x}_i - \vec{m}_1)^\top + \sum_{i \in C_2} (\vec{x}_i - \vec{m}_2)(\vec{x}_i - \vec{m}_2)^\top$
- Das Optimierungskriterium lässt sich auch hier als Rayleigh Koeffizient schreiben:

$$\mathbf{P}_{csp} = \operatorname{argmax}_{\mathbf{P} \in \mathbb{R}^{d \times r}} \left[\operatorname{tr} \frac{\mathbf{P}^\top \mathbf{S}_b \mathbf{P}}{\mathbf{P}^\top \mathbf{S}_w \mathbf{P}} \right]$$

- Und nun analog für die FDA:

$$\mathbf{P}_{fda} = \operatorname{argmax}_{\mathbf{P} \in \mathbb{R}^{d \times 1}} \left[\operatorname{tr} \frac{\mathbf{P}^\top \mathbf{S}_b \mathbf{P}}{\mathbf{P}^\top \mathbf{S}_w \mathbf{P}} \right] \quad \text{bzw.:} \quad J(\vec{w}) = \frac{\vec{w}^\top \mathbf{S}_b \vec{w}}{\vec{w}^\top \mathbf{S}_w \vec{w}}$$

Die Verwandtschaft von FDA und CSP

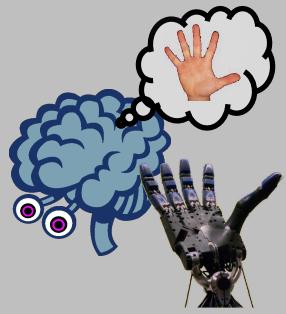


Beides lässt sich nun als das schon bekannte, generalisierte Eigenwert-Problem schreiben:

$$\mathbf{S}_b \mathbf{V} = \lambda \mathbf{S}_w \mathbf{V}$$

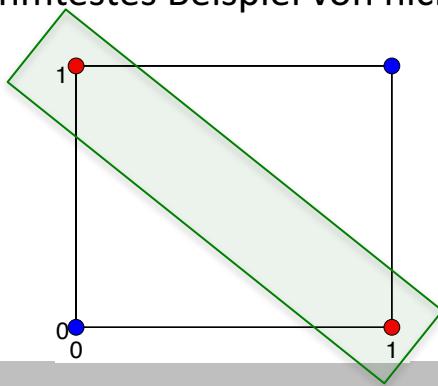
Was sagt das über die CSP?

- Sie verwendet exakt die gleichen Optimierungskriterien wie die FDA.
- Die Eingabedaten werden jedoch in einen d-dimensionalen Raum projiziert. Bei der FDA nur in einen 1-dimensionalen.
- Die FDA ist somit ein Spezialfall der CSP.

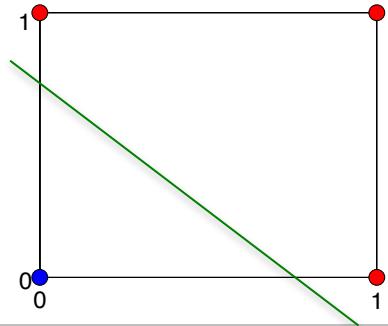


Andere lineare Klassifikatoren

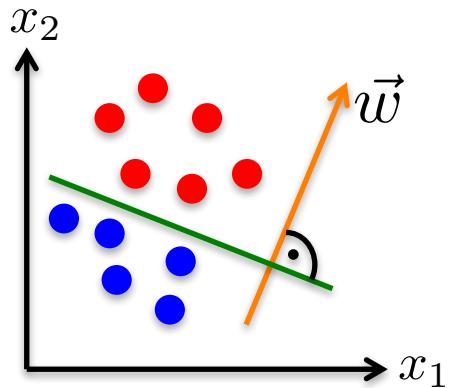
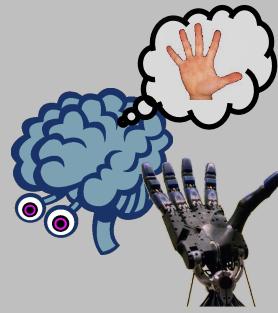
- Lineare Support Vector Maschinen (SVM).
- KNN (k-nearest neighbor) Klassifikator
- Ein einfaches Neuronales Netz, das Perceptron.
- etc.
- Allen gemeinsam ist, dass sie nur Daten erfolgreich klassifizieren können, wenn die Klassen im Eingaberraum mittels einer Hyperebene getrennt werden können. Sie heissen dann **linear separierbar**.
- Berühmtestes Beispiel von nicht-linear separierbaren Daten: das XOR – Problem $y = x_1 \oplus x_2$



Anders das OR $y = x_1 \vee x_2$

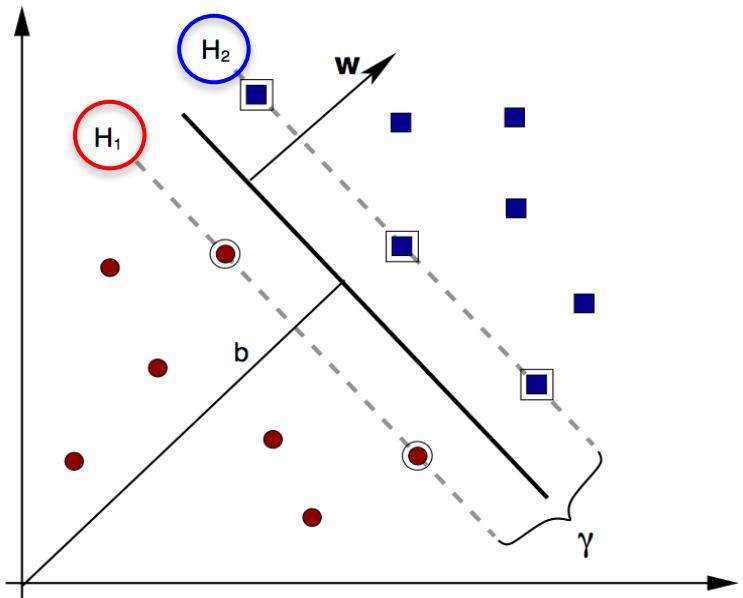


Die Idee der SVM

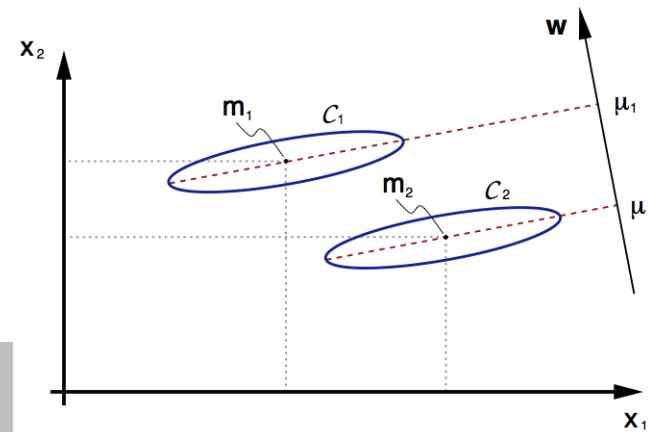


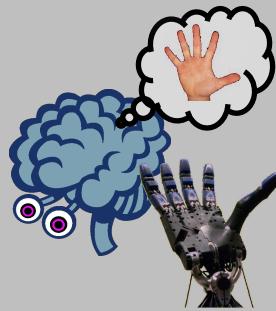
- Die **lineare SVM** benutzt im Testfall die exakt gleiche Berechnungsvorschrift für die Klassifikatorausgabe wie die FDA

$$y = \text{sgn}(\vec{w} \cdot \vec{x}_{test} + b)$$
- Allerdings verwendet sie ein anderes Optimierungskriterium, die **Support Vektoren (H_1, H_2)**
- Das Lernziel ist die **Maximierung des Margin γ** .

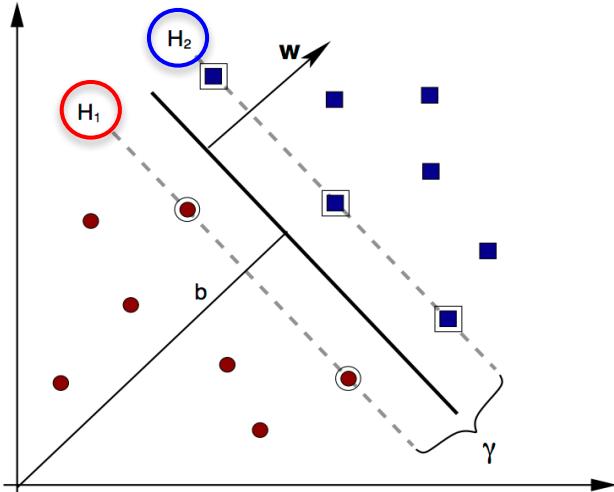


- Letztendlich führt aber dieses Kriterium zu einer analogen Lage der Hyperebene im Datenraum.

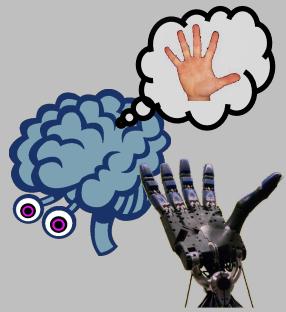




Die Support Vektoren



- Die Support Vektoren sind diejenigen Trainingsbeispiele, die den geringsten Abstand zur separierenden Hyperebene haben.
 - Die Support Vektoren sind die einzigen relevanten Vektoren für das Trainings-Problem. Trainierte man den Klassifikator unter Ausschluss aller anderen Trainingsdaten, ergäbe sich exakt der gleiche Klassifikator.
 - Somit ist der Klassifikator durch die Support Vektoren vollständig beschrieben.
-
- Im obigen Beispiel ist die Trainingsmenge perfekt separiert.
 - Wird dies im Training verlangt, spricht man von einem **Hard Margin** Ansatz.
 - Dieser Fall kommt in der Realität praktisch nicht vor.



Berechnung des Margin γ

- Wird w normalisiert, ergibt sich:

$$x_i \cdot w + b \geq +1 \quad \text{for } y_i = +1$$

$$x_i \cdot w + b \leq -1 \quad \text{for } y_i = -1$$

Und zusammengefasst:

$$y_i (x_i \cdot w + b) - 1 \geq 0 \quad \forall i$$

- Die Support Vektoren (SV) werden beschrieben durch:

$$x_i \cdot w + b = 1 \quad \text{und} \quad x_i \cdot w + b = -1$$

- Uns zusammengefasst:

$$y_i (x_i \cdot w + b) - 1 = 0 \quad \forall (x_i, y_i) \in \mathbb{D}_{SV}$$

Raum der Support
Vektoren

- Abstand eines Datenpunktes auf den SVs zum Ursprung :

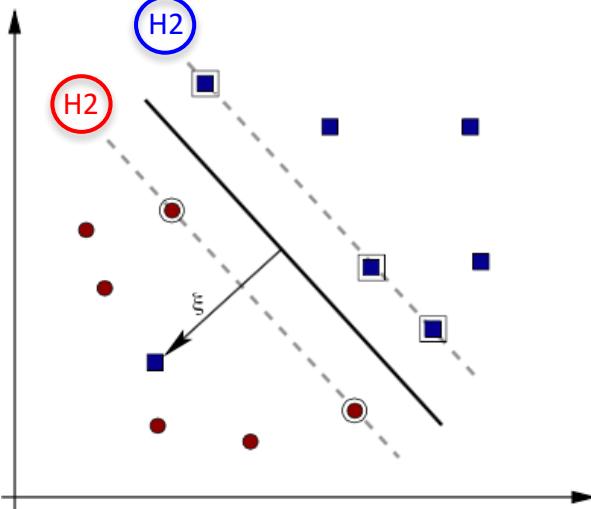
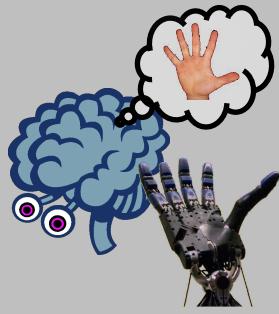
$$\begin{array}{ll} \text{auf } H_1 & \frac{1 - b}{\|w\|} \\ & \text{auf } H_2 & \frac{-1 - b}{\|w\|} \end{array}$$

- Daraus ergibt sich der Margin: $\gamma = H_1 - H_2 = \frac{2}{\|w\|}$

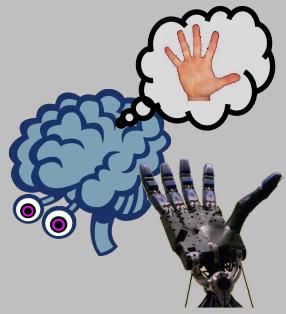
- Einer Maximierung von γ entspricht daher eine Minimierung von w .

- Minimiert wird die quadratische Form $\frac{1}{2} \|w\|^2$

Der Soft Margin Ansatz



- Der **Soft Margin** Ansatz lässt eine bestimmte Menge Trainingsbeispiele zu, die das Separationskriterium verletzen.
 - Dieses wird erreicht, in dem so genannte **Slack Variablen** ξ eingeführt werden.
 - Die Slack Variablen bezeichnen diejenigen Trainingsvektoren, die auch der "falschen" Seite der Hyperebene liegen.
-
- Im obigen Beispiel ist die Trainingsmenge perfekt separiert.
 - Wird dies im Training verlangt, spricht man von einem **Hard Margin** Ansatz.
 - Dieser Fall kommt in der Realität praktisch nicht vor.



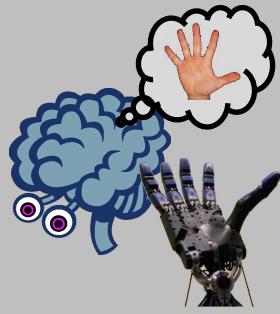
Berechnung des Margin γ

- Im Soft Margin Ansatz ergibt sich dann folgende Form

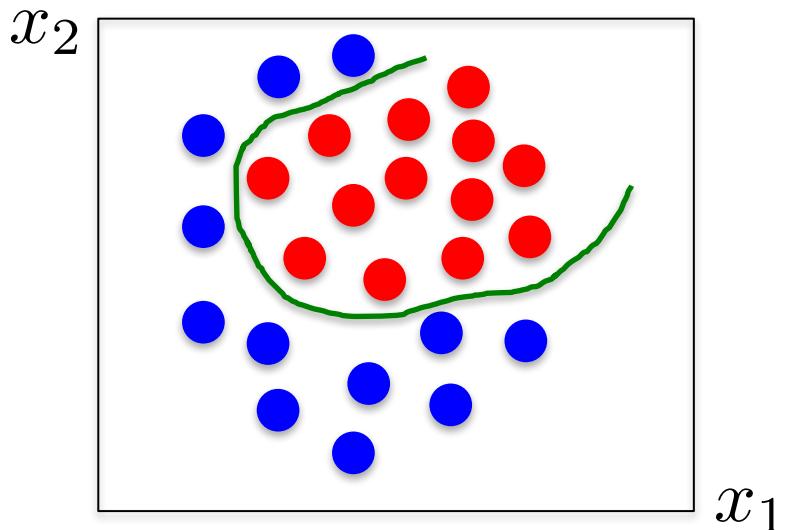
$$y_i (x_i \cdot w + b) - 1 \geq 1 - \xi \quad \forall i$$

- Trainingsvektoren, die das Separations-Kriterium verletzen, sollten möglichst selten bleiben.
- Daher wird dem Optimierungskriterium eine **Kostenfunktion** hinzugefügt, die das Auftreten solcher Slack Variablen bestraft:
$$\frac{\|w\|^2}{2} + C \sum_i^l \xi_i^k$$
- C ist ein **Regularisierungsparameter**. Je höher der Wert von C , desto stärker werden nicht korrekt separierte Datenbeispiele bestraft.
- Der Wert für C wird üblicherweise in einem **Kreuzvalidierungsverfahren** bestimmt, bei dem eine vorgegebene Wertespanne von C ausgewertet wird.
- Im linearen Fall (LSVM) ist C der einzige so zu findende Parameter.

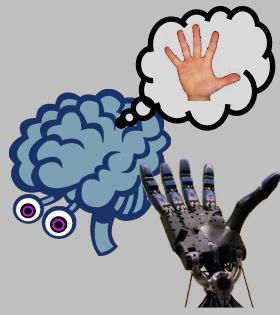
Nicht-lineare Klassifikatoren



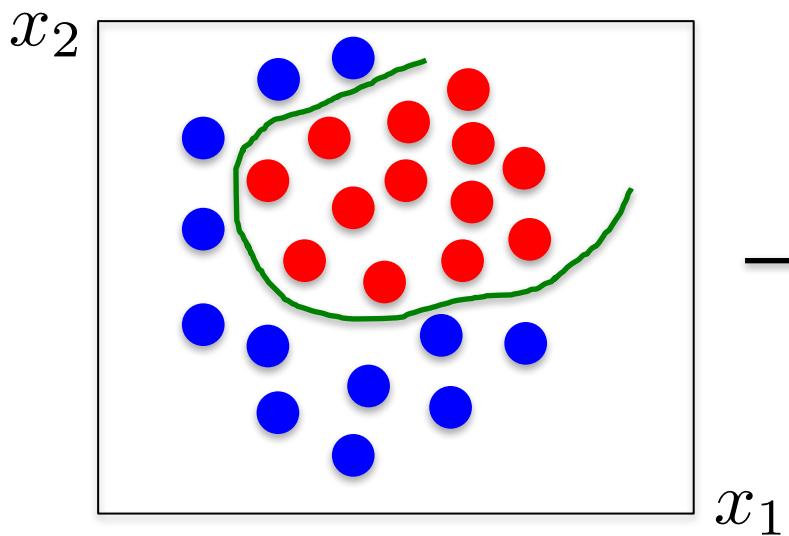
- Quadratische LDA (QLDA)
 - **Support Vector Maschinen (SVM)**
 - Verschiedene neuronale Netze, z.B. das Multi-Lagen Perzeptron (MLP)
 - ...
-
- Prinzipiell kann durch die Anwendung eines **Kernel-Tricks** aus verschiedenen eigentlich linearen Klassifikatoren ein nicht-linearer werden.



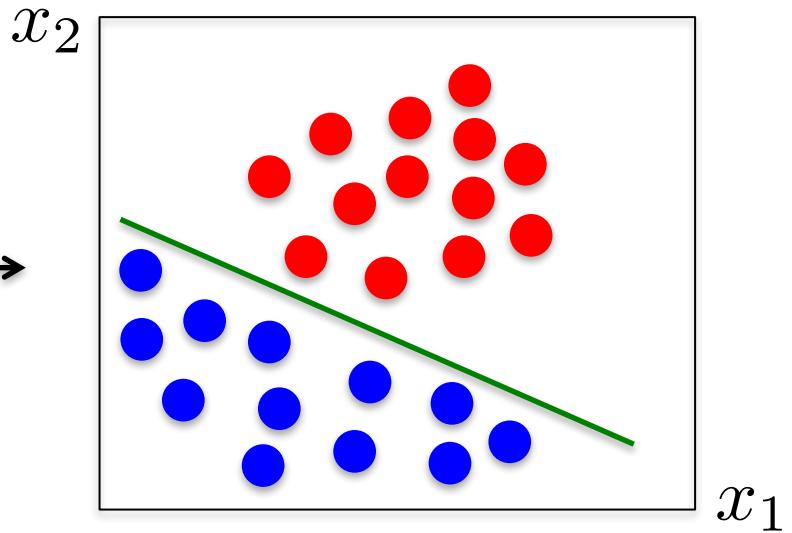
Der Kernel-Trick



Eingaberaum $\mathbb{L} \subset \mathbb{R}^n$



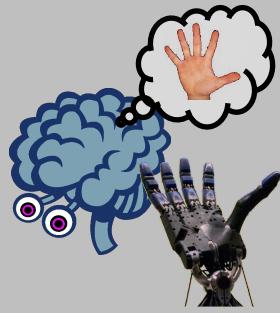
Merkmalsraum $\mathbb{H} \subset \mathbb{R}^m \quad n << m$



Gesucht wird eine Transformation Φ , welche die im Datenraum nicht linear-separierbaren Daten in einen solchen Merkmalsraum transformiert, in welchem sie linear-separabel werden.

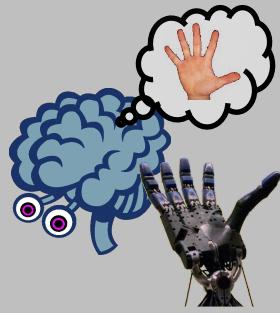
Dieser Merkmalsraum wird i.d.R. **deutlich höher dimensional** sein als der Eingaberaum!

Der Kernel-Trick



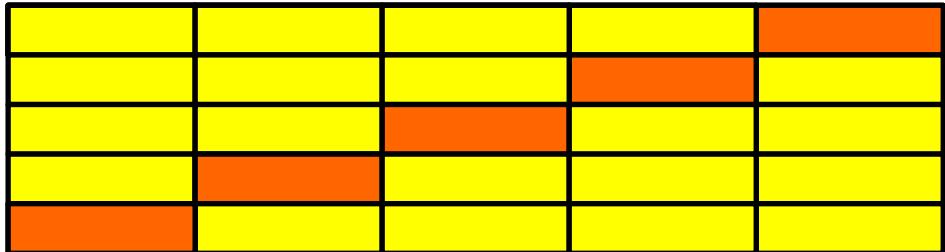
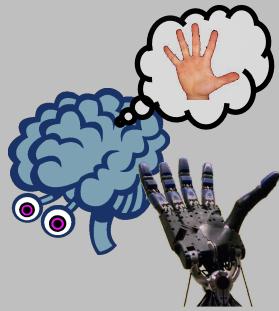
- Der Trick ist nun, diese Transformation nicht explizit zu berechnen, also $\Phi(\vec{x})$, sondern nur implizit mit $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$
- Häufige Kernel-Funktionen sind
 - Linearer Kernel $K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$
 - Polynomieller Kernel $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d$
 - **RBF Kernel** $K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$
 - Sigmoides neuronales Netz $K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x} \cdot \mathbf{y} - \delta)$

Training SVM mit RBF - Kernel



- Der RBF Kernel (Radial Basis Function): $K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$
- Bei Verwendung einer RBF-SVM ergibt sich ein weiterer Parameter, die Breite des Kernels γ .
- Somit müssen im Training die Parameter C und γ gefunden werden, wiederum in einem Kreuzvalidierungsverfahren.
- Dies geschieht in der Regel mittels eines Gridsearch, bei welchem innerhalb der Kreuzvalidierung alle möglichen Kombinationen der Parameter getestet werden.
- Dieses führt zu einer rechenaufwändigen Trainingsprozedur.

Training SVM mit RBF - Kernel

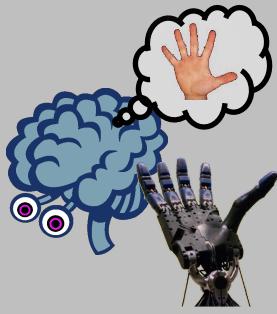


 Trainset
 Testset

- Für jede mögliche Kombination aus C und γ wird die n-fache Kreuzvalidierung berechnet.
- Der abzusuchende Parameterraum sowie die Schrittweiten werden zuvor festgelegt.
- Ist der Parameterraum komplett unbekannt, kann in einem ersten Schritt ein größerer Raum mit größerer Schrittweite abgesucht werden und in einem folgenden Schritt feinjustiert werden.
- In diesem Beispiel ergeben sich $5 * 16 = 80$ einzelne Trainings- und Testdurchläufe (in der Praxis können das 10^4 und mehr sein).

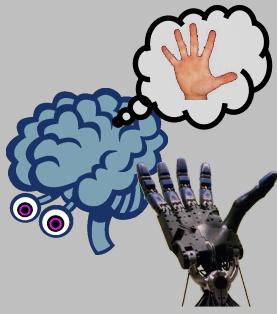
c_1	c_1	c_1	c_1
γ_1	γ_2	γ_3	γ_4
c_2	c_2	$c_2 \gamma_3$	c_2
γ_1	γ_2		γ_4
c_3	c_3	c_3	c_3
γ_1	γ_2	γ_3	γ_4
c_4	c_4	c_4	c_4
γ_1	γ_2	γ_3	γ_4

Maße zur Abschätzung der Klassifikatorleistung

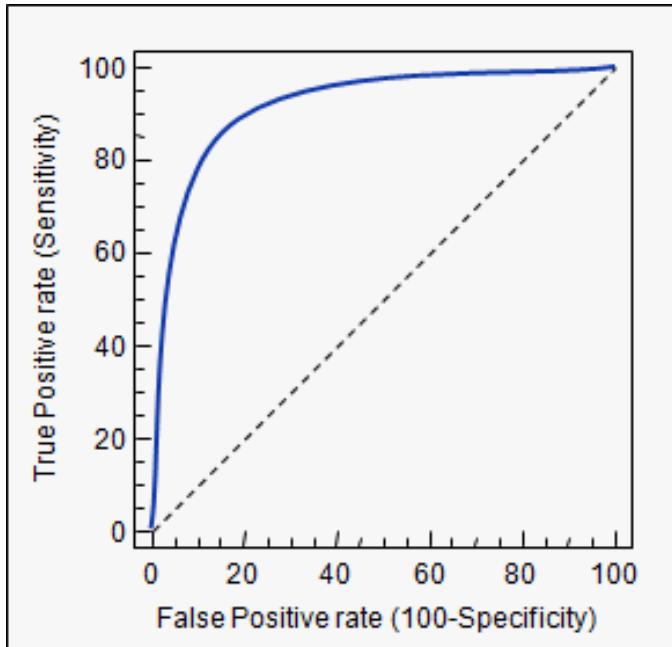


- Accuracy (Acc) Quotient aus der Anzahl korrekt klassifizierter Datenbeispiele und der Anzahl fehlklassifizierter Datenbeispiele.
- Error (Err) $1 - \text{Accuracy}$
- True Positives (TP) Korrekt klassifizierte Datenbeispiele aus der **Zielklasse**.
- True Negatives (TN) Korrekt klassifizierte Datenbeispiele aus der **Nicht-Zielklasse**.
- False Positives (FP) Falsch klassifizierte Datenbeispiele aus der **Zielklasse**.
- False Negatives (FN) Falsch klassifizierte Datenbeispiele aus der **Nicht-Zielklasse**.
- Sensitivity or Recall $\text{TP} / (\text{TP} + \text{FN})$
- Specificity $\text{TN} / (\text{TN} + \text{FP})$
- Precision $\text{TP} / (\text{TP} + \text{FP})$ (auch Positive Predictive Value, PPV, genannt)
- F-Measure or F1-Score
$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$
4 Methoden und Algorithmen

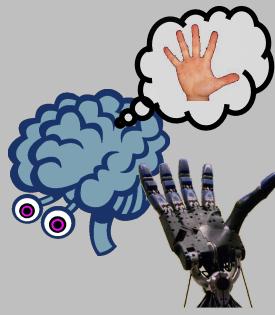
Maße zur Abschätzung der Klassifikatorleistung



- Die Accuracy selbst ergibt als Maß nur Sinn, wenn die Klassen balanciert sind, d.h. jede Klasse die gleiche Anzahl Samples enthält (jede Klasse hat eine apriori Wahrscheinlichkeit von 0,5 im binären Fall). Dies ist auch das *chance level*.
- Confusion Matrix
 - TP | FN
 -
 - FP | TN
- **ROC Kurve**, bzw. die Fläche darunter, die Area under Curve **AUC**
 - Die ROC gibt nicht nur ein Maß für die Klassifikationsgenauigkeit, sondern erlaubt ein Problemspezifisches Justieren der Parameter (z.B. weniger FN zulassen als an der optimalen Stelle der Kurve).
 - Das chance level ist immer 0,5 .



Beispielcode ROC



```
fda = fda_train(trainset,trainlabel);

testsize = size(testset,1);
proj = zeros (testsize,1);

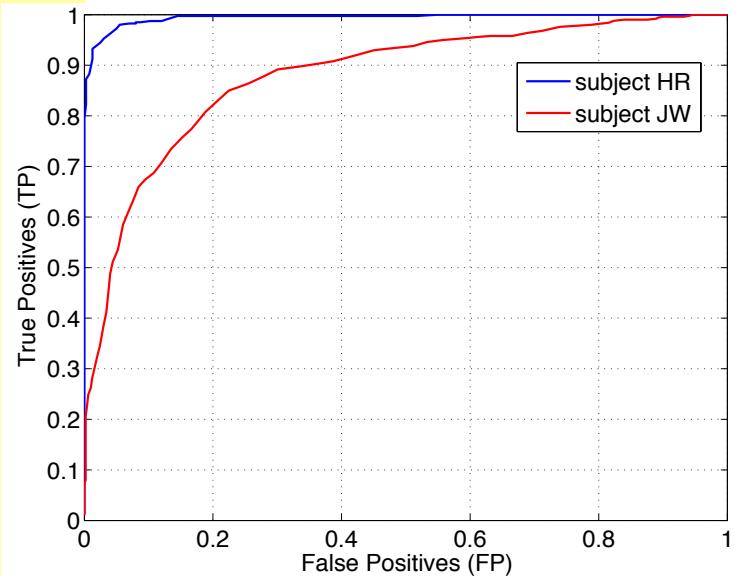
for j=1:testsize
    proj(j) = testset(j,:)*fda.w-fda.b;
end

proj = Scale(-1*proj, 0, 1);%0<=proj<=1
pred = zeros(testsize,1);
thr = linspace(0,1);

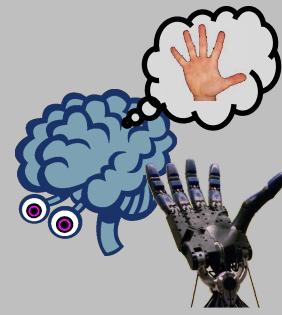
for b=1:numel(thr)
    pred(proj<=thr(b)) = 1;

    TP = numel(find(pred(1:testpos)==1))/testpos;
    FP = numel(find(pred(testpos+1:end)==1))/testneg;

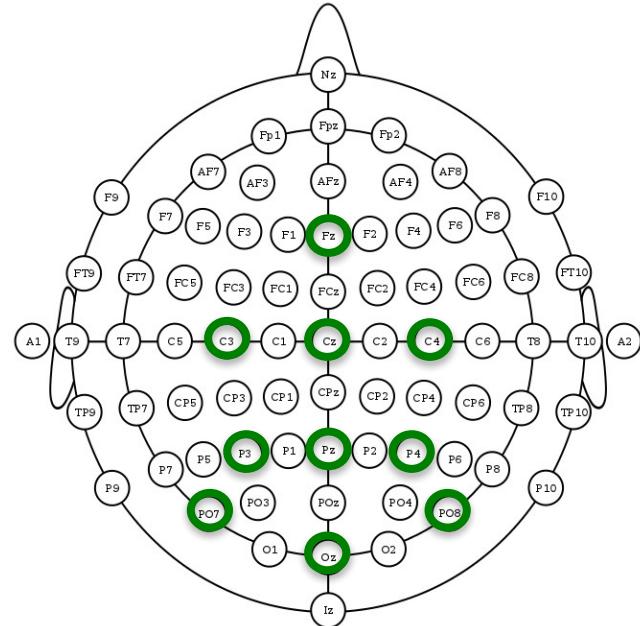
    roc(1,b) = TP;%TP
    roc(2,b) = FP;%FP
end
```



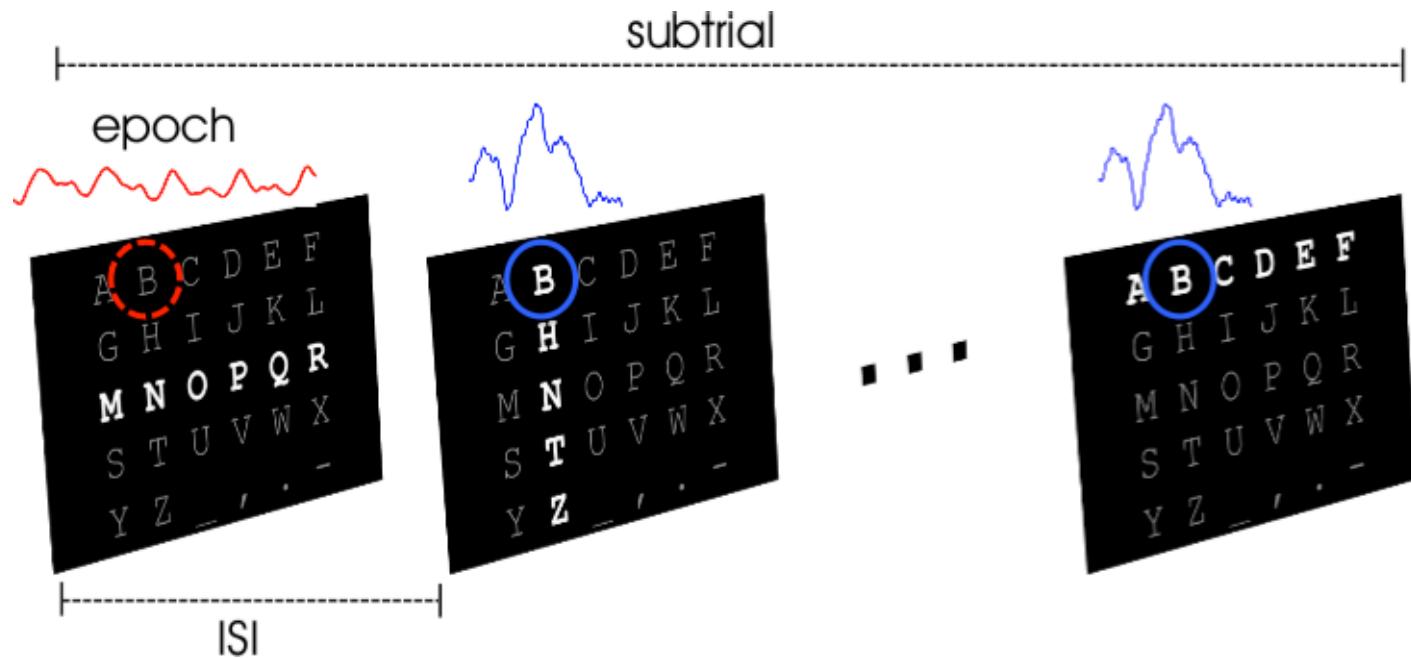
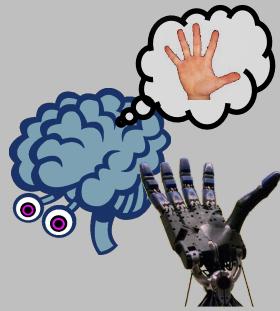
Von Rohdaten zur Klassifikation – Beispiel des P300 Speller



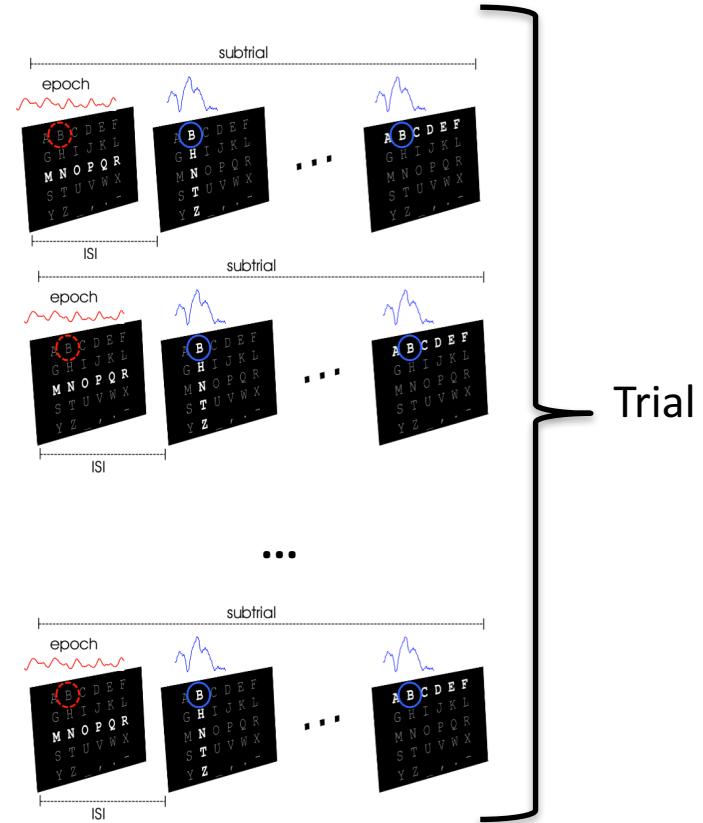
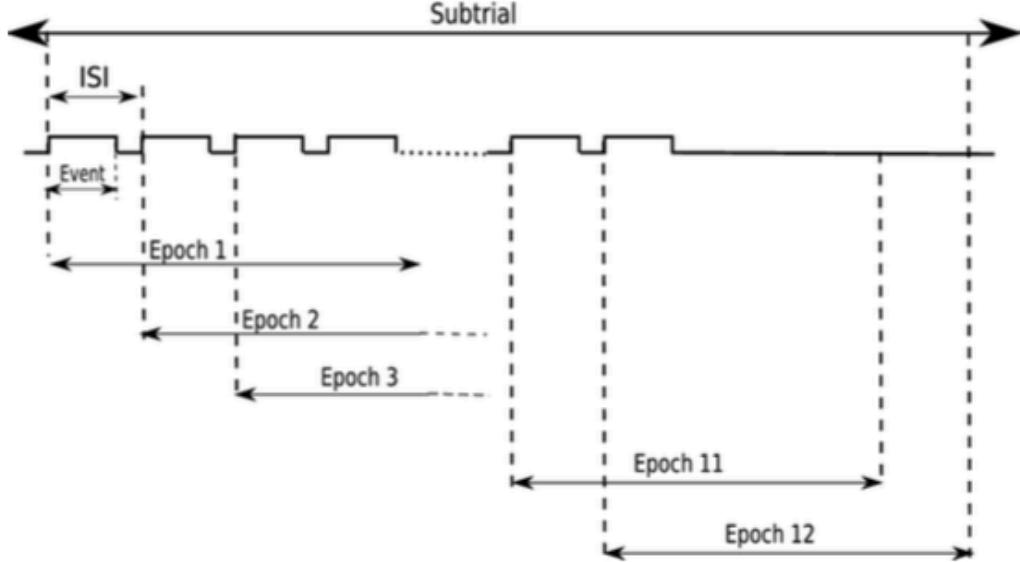
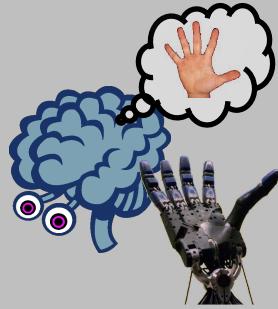
- 10 EEG Kanäle an
Fz, Cz, Pz, Oz, C3, C4, P3, P4, PO7, PO8
- Sampling Rate 256 Hz
- Vorverarbeitung Bandpassfilter 1 -10 Hz
- Merkmalsextraktion mit PCA
- Klassifikation mit FDA
- Dynamische Subtrial – Limitierung



Once upon a speller...

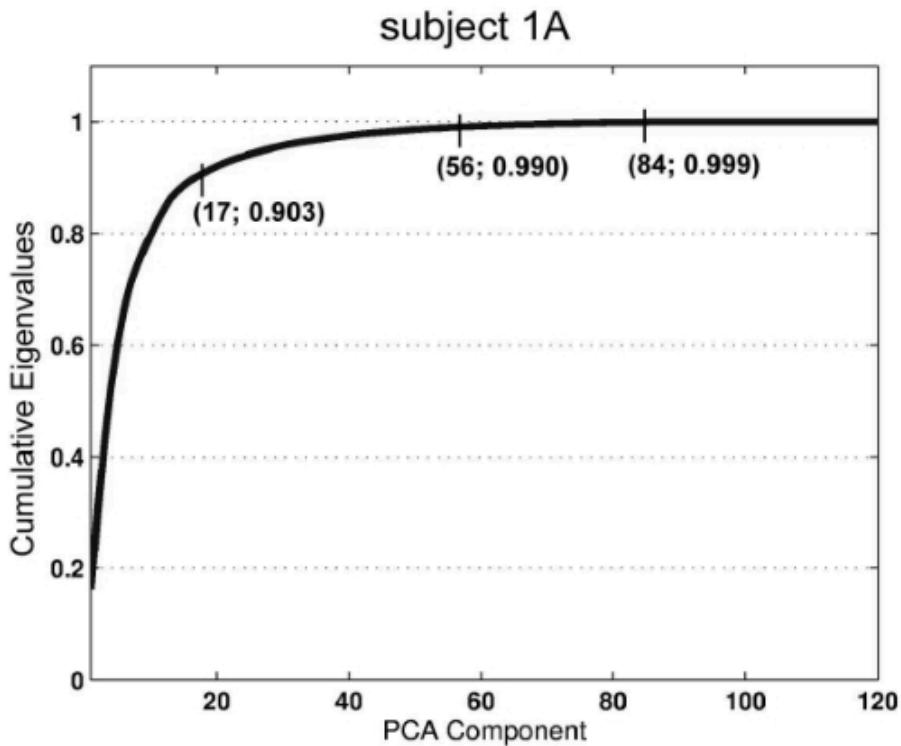
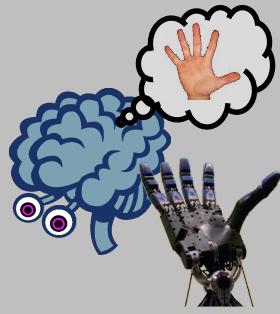


Epoching (Segmentierung)



- Epochenlänge 800 ms (205 Samples)

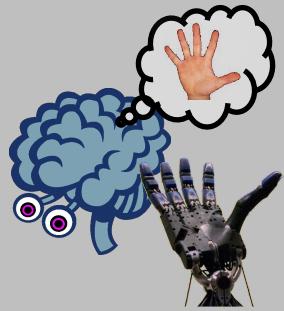
Anwendung PCA



Dimensionsreduktion:

Von 2560 auf 84

FDA Klassifikation

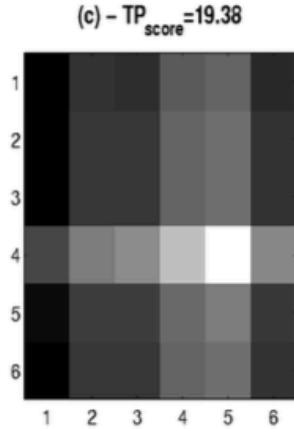
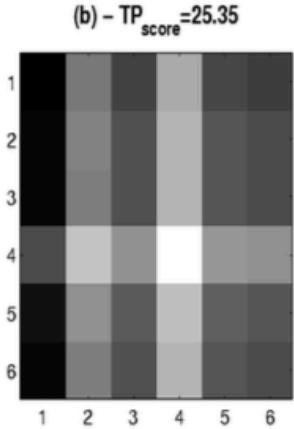
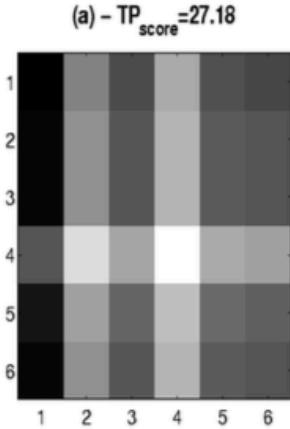
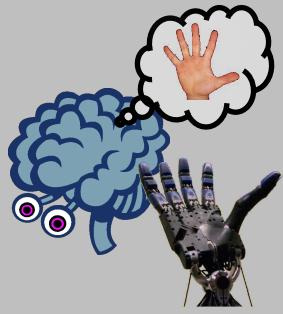


	$S_{c_1}^\Psi$	$S_{c_2}^\Psi$	$S_{c_3}^\Psi$	$S_{c_4}^\Psi$	$S_{c_5}^\Psi$	$S_{c_6}^\Psi$
$S_{r_1}^\Psi$	A	B	C	D	E	F
$S_{r_2}^\Psi$	G	H	I	J	K	L
$S_{r_3}^\Psi$	M	N	O	P	Q	R
$S_{r_4}^\Psi$	S	T	U	V	W	X
$S_{r_5}^\Psi$	Y	Z	0	1	2	3
$S_{r_6}^\Psi$	4	5	6	7	8	9

- Die Balken geben das summierte Klassifikationsergebnis (je Zeile bzw. Spalte) über 10 Subtrials wieder.
- Bei einer festen Anzahl Subtrials ergibt sich häufig eine unnötige Anzahl Wiederholungen.

→ Dynamische Subtrial-Limitierung

Dynamische Subtrial Limitierung



TP → True Positive Score

Wird im Training ermittelt.

