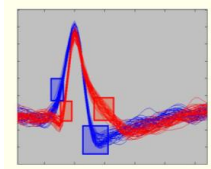
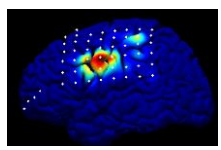
CSE 599E

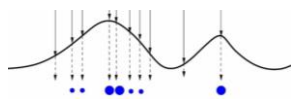
Lecture 4:

Signal Processing and

Machine Learning for BCI



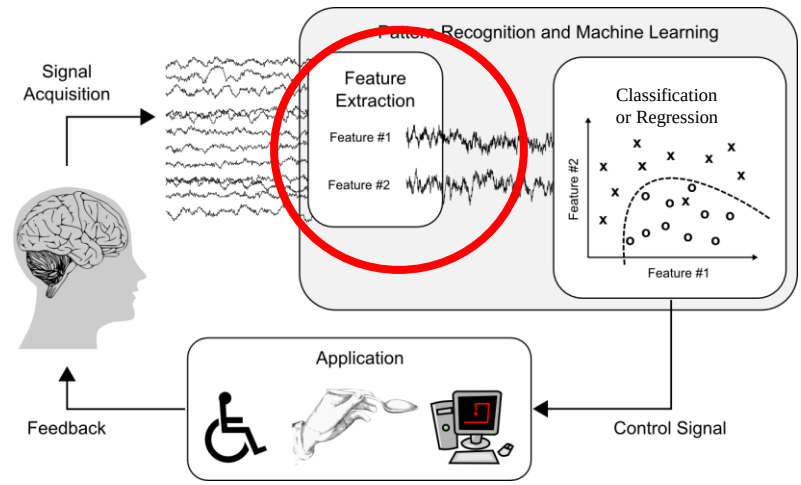




What's on the menu?

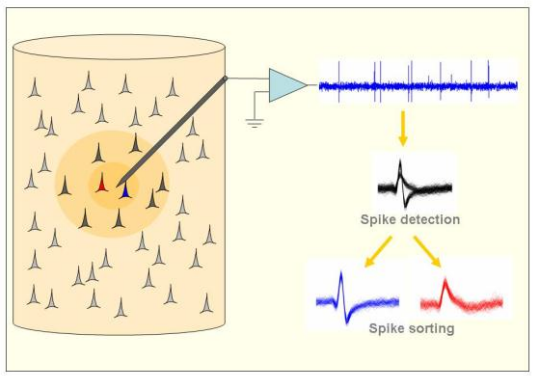
- Signal Processing
  - Spike sorting
  - Frequency analysis
  - Wavelets
  - Time domain analysis
    - Bayesian Filtering, Kalman and Particle filtering
- Machine Learning
  - Regression: Linear regression, Neural networks
  - Classification: LDA, SVM
  - Cross Validation

Components of Brain-Computer Interfacing

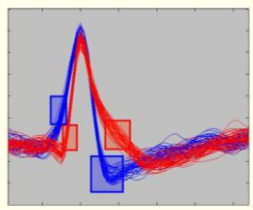


Feature Extraction and Signal Processing

Invasive Recording: Spike Sorting



Sorting by Amplitude

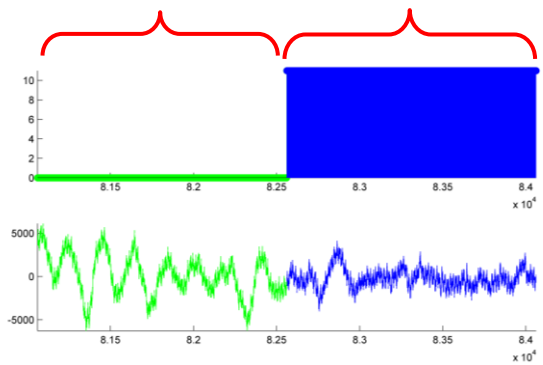


Sorting using Window Discriminators

Signal Processing for Semi-Invasive and Non-Invasive Recordings

Example: ECoG Signal

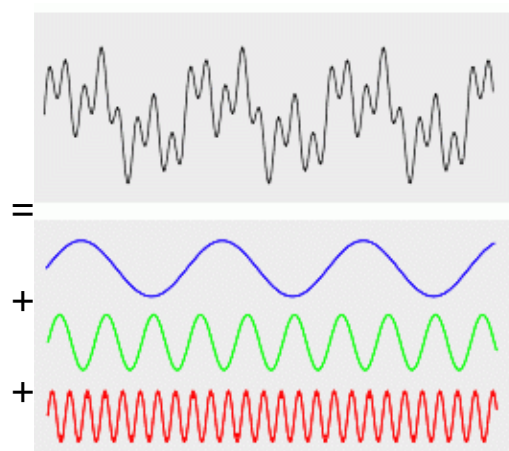
Rest Period Hand movement



Slow oscillations have disappeared

Faster fluctuations are apparent

Decomposing a signal into components: Frequency domain analysis

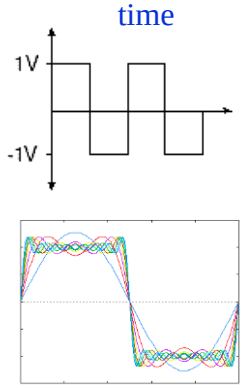


Recorded signal

Increasing frequency

Example of Frequency Domain (Fourier) Analysis

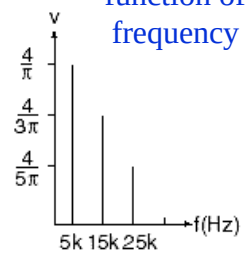
Signal as a function of time



$v(t) = \frac{4}{\pi}\sin(\omega 1)t + \frac{4}{3\pi}\sin(\omega 2)t + \frac{4}{5\pi}\sin(\omega 3)t\ldots$

$\omega 1 = 2\pi(5\text{kHz})$
$\omega 2 = 2\pi(15\text{kHz})$
$\omega 3 = 2\pi(25\text{kHz})\ldots$

Signal as a function of frequency



Fourier Analysis

- Express any signal as an infinite sum of sines and cosines

$$s(t) = \frac{a_0}{2} + a_1\cos(\omega t) + a_2\cos(2\omega t) + \ldots + b_1\sin(\omega t) + b_2\sin(2\omega t) + \ldots$$

$$= \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n\cos(n\omega t) + \sum_{n=1}^{\infty} b_n\sin(n\omega t)$$

- Obtain each "Fourier coefficient" via correlation with signal:

$$a_n = \frac{2}{T}\int_{-T/2}^{T/2} s(t)\cos(n\omega t)dt$$

$$b_n = \frac{2}{T}\int_{-T/2}^{T/2} s(t)\sin(n\omega t)dt$$

Amplitude and Power Spectra

$$s(t) = \frac{a_0}{2} + a_1\cos(\omega t) + a_2\cos(2\omega t) + \ldots + b_1\sin(\omega t) + b_2\sin(2\omega t) + \ldots$$

- Amplitude Spectrum

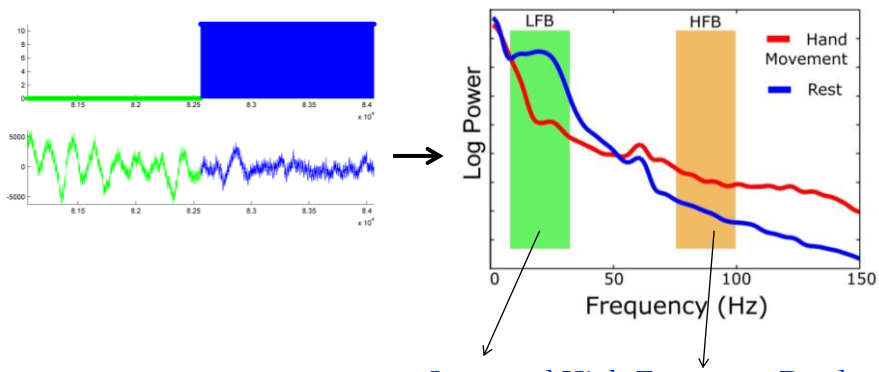
  $$A(n) = \frac{1}{2}\sqrt{a_n^{\,2} + b_n^{\,2}}$$

- Power Spectrum

  $$P(n) = A(n)^2 = \frac{1}{4}(a_n^{\,2} + b_n^{\,2})$$

- Fast Fourier transform (FFT): Efficient way of computing coefficients and power spectrum
  - For signal with T time points, runs in time approximately $T \log T$

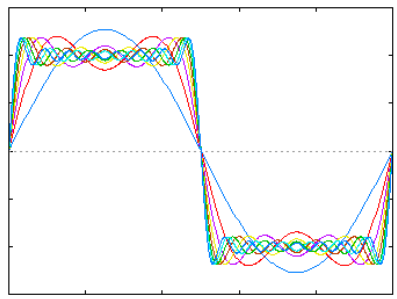
ECoG Signal → Power Spectrum



Low- and High-Frequency Band (LFB/HFB) "features" can be used to detect hand movement

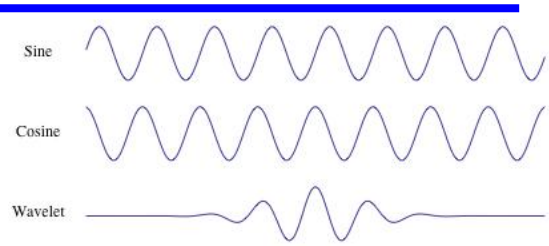
Problem with Fourier Analysis

- Sines and cosines occupy an infinite temporal extent
- Fourier transform does a poor job of representing finite and non-periodic signals, and signals with sharp peaks and discontinuities

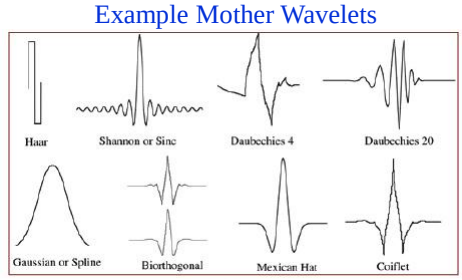


Wavelets

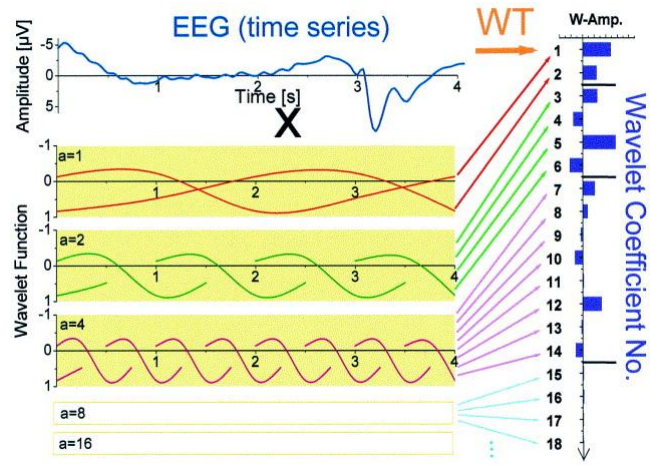
- Represent signals as linear combinations of finite basis functions called *wavelets*
- Each wavelet is a scaled and translated copy of a single *mother wavelet*
- Allows multi-resolution analysis



Example Mother Wavelets



Example: Wavelet Decomposition of EEG



(Hinterberger et al., 2003)

Autoregressive (AR) Models

- Natural signals tend to be correlated over time
- Predict the next measurement based on past few values:

$$x_t = \sum_{i=1}^{p} a_i x_{t-i} + \varepsilon$$

- Adaptive Autoregressive (AAR) model uses time-varying set of coefficients $a_{i,t}$:

$$x_t = \sum_{i=1}^{p} a_{i,t} x_{t-i} + \varepsilon_t$$

- The $a_{i,t}$ capture local statistics of signal and can be used as features for classification in a BCI

Bayesian Filtering:
Kalman filters and Particle filters

Bayes' Rule

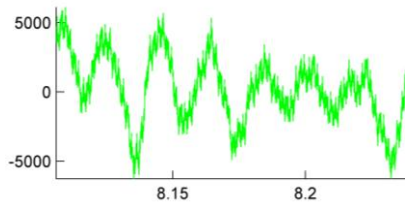
$$P(x, y) = P(x \mid y)P(y) = P(y \mid x)P(x)$$

$$\Rightarrow$$

$$\underbrace{P(x \mid y)}_{\text{Posterior}} = \frac{P(y \mid x)\ P(x)}{P(y)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{normalizer}}$$

Example: Estimating brain state from neural data

- Suppose we measure ECoG signal *z*



- What is *P(HandMovement|z)?*

Causal vs. Diagnostic Reasoning

- *P(HandMovement|z)* is diagnostic.
- *P(z|HandMovement)* is causal.

count frequencies!

- Often causal knowledge is easier to obtain or learn.
- Bayes rule allows us to convert causal knowledge to diagnose events:

$$P(HM \mid z) = \frac{P(z \mid HM)P(HM)}{P(z)}$$

Hand Movement Example

- $P(z|HM) = 0.6 \quad P(z|\neg HM) = 0.3$
- $P(HM) = P(\neg HM) = 0.5$

From past data

$$P(HM \mid z) = \frac{P(z \mid HM)P(HM)}{P(z \mid HM)P(HM) + P(z \mid \neg HM)P(\neg HM)}$$

$$P(HM \mid z) = \frac{0.6 \cdot 0.5}{0.6 \cdot 0.5 + 0.3 \cdot 0.5} = \frac{0.3}{0.45} = \frac{2}{3} = 0.67$$

If z is measured, it raises the probability of hand movement from 0.5 to 0.67

Combining Measurements over Time

- Suppose we make another measurement z_2.
- How can we integrate this new measurement?
- More generally, how can we estimate $P(x| z_1...z_n)$?

Bayesian Filtering

Bayes' Rule

$$P(x \mid z_1,\ldots,z_n) = \frac{P(z_n \mid x, z_1,\ldots,z_{n-1})\, P(x \mid z_1,\ldots,z_{n-1})}{P(z_n \mid z_1,\ldots,z_{n-1})}$$

Markov assumption: z_n is independent of $z_1,\ldots,z_{n-1}$ if we know x.

$$P(x \mid z_1,\ldots,z_n) = \frac{P(z_n \mid x, z_1,\ldots,z_{n-1})\, P(x \mid z_1,\ldots,z_{n-1})}{P(z_n \mid z_1,\ldots,z_{n-1})}$$

$$= \frac{P(z_n \mid x)\, P(x \mid z_1,\ldots,z_{n-1})}{P(z_n \mid z_1,\ldots,z_{n-1})}$$

$$= \alpha\, P(z_n \mid x)\, P(x \mid z_1,\ldots,z_{n-1})$$

From previous time step!

Kalman Filtering

- State linearly evolves according to: $x_t = Ax_{t-1} + n_t$
- Measurement linearly related to state: $z_t = Bx_t + m_t$
- n_t and m_t are zero-mean Gaussian noise with covariance matrices Q and R respectively
- Bayesian filtering:

$$P(x_t \mid z_1,\ldots,z_t) = \alpha\, P(z_t \mid x_t)\, P(x_t \mid z_1,\ldots,z_{t-1})$$

$$= \alpha\, P(z_t \mid x_t) \int_{x_{t-1}} P(x_t \mid x_{t-1}) P(x_{t-1} \mid z_1,\ldots,z_{t-1}) dx_{t-1}$$

- Because everything is Gaussian and linear, posterior is also Gaussian: $P(x_t \mid z_1,\ldots,z_t) = N(\hat{x}_t, S_t)$

Kalman Filter Equations

- Prediction from one time step to next:

  $\bar{x}_t = A\hat{x}_{t-1}$ Mean

  $M_t = AS_{t-1}A^T + Q$ Covariance

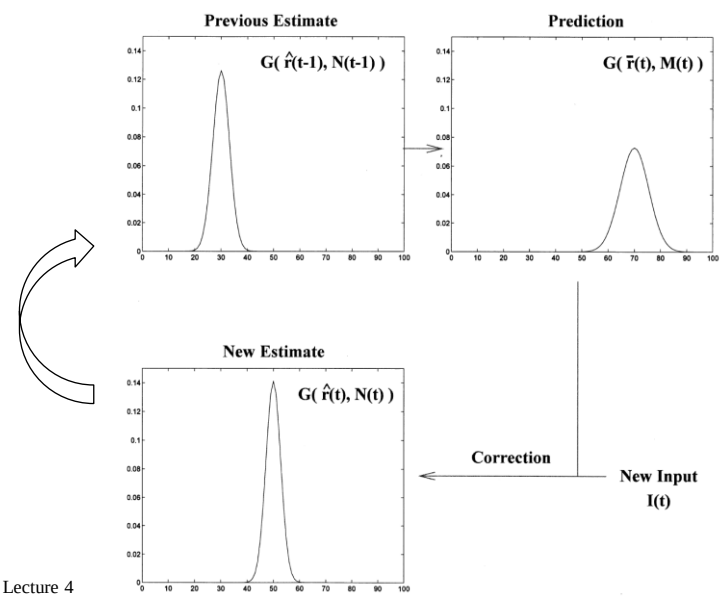
- Correction upon measuring z_t:

  $\hat{x}_t = \bar{x}_t + K_t(z_t - B\bar{x}_t)$ Mean

  $S_t = (B^T R^{-1} B + M_t^{-1})^{-1}$ Covariance

- "Kalman" gain $K_t = S_t B^T R^{-1}$ dictates how much weight to give to prediction error $(z_t - B\bar{x}_t)$ versus prediction $\bar{x}_t$

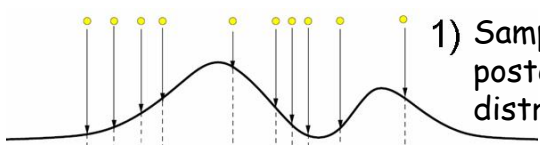
Kalman Filter: Prediction and Correction Cycle



Kalman filter assumes linear and Gaussian model

How do we handle non-linearity and multi-modal distributions?

Enter…Particle Filtering



1) Samples approximate posterior probability distribution

Spatial Filtering (across electrodes) and Feature Extraction

Spatial filtering: Simple Techniques

- **Bipolar filtering**

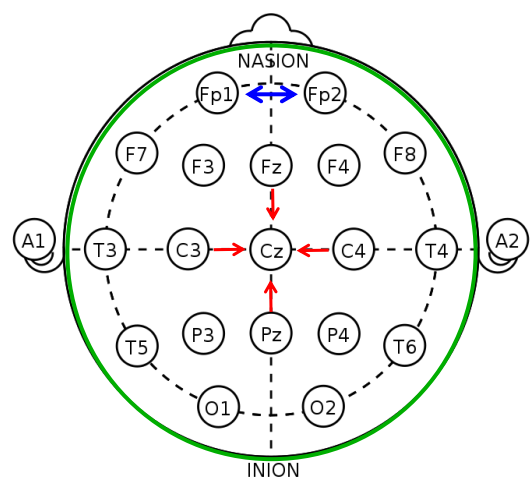
$$\tilde{\mathbf{s}}_{i,j} = \mathbf{s}_i - \mathbf{s}_j$$

- **Laplacian filtering**

$$\tilde{\mathbf{s}}_i = \mathbf{s}_i - \frac{1}{4}\sum_{i \in \Theta} \mathbf{s}_i .$$

- **Common Average Re-referencing**

$$\tilde{\mathbf{s}}_i = \mathbf{s}_i - \frac{1}{N}\sum_{i=1}^{N} \mathbf{s}_i .$$

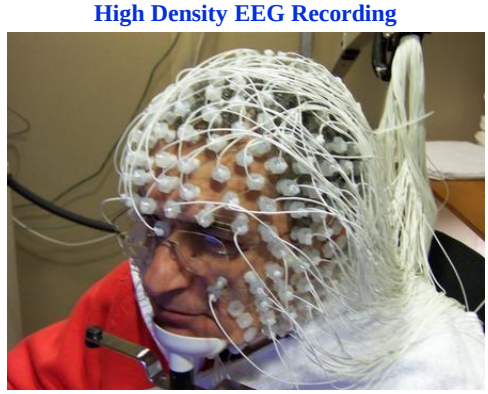


What if you are faced with this…

http://people.brandeis.edu/~sekuler/imgs/rwsEEG.jpg

Reducing Dimensionality of Input Data

High Density EEG Recording



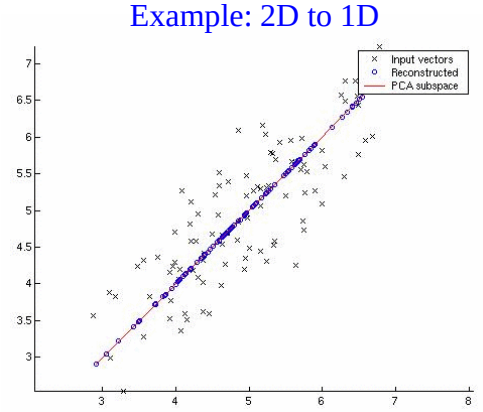
Can we extract and use a small set of features rather than the large number of input channels?

http://people.brandeis.edu/~sekuler/imgs/rwsEEG.jpg

Principal Component Analysis (PCA)

- Convert high-dimensional input into a small set of *linear* "features"
  - Eg., 256 EEG channels → 10 "features"
- Features determined by directions in data with maximal variance
- Data can be reconstructed (with some error) by linear combination of features

Example: 2D to 1D



Principal Component Analysis (PCA)

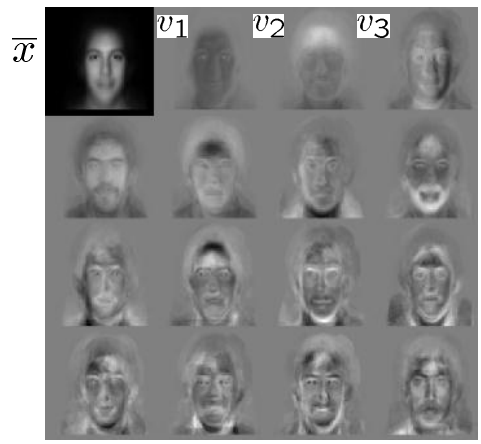
- Suppose each of the N data points is L-dimensional
  - Form L x L *data covariance matrix* A

$$A = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

- Compute *eigenvectors* of A → these are the *"feature" vectors* or *spatial filters*
- Pick N largest eigenvalues and use their eigenvectors as your features/filters

Example: PCA for images of faces

- PCA extracts the eigenvectors v_1, v_2, v_3, ... v_K of covariance matrix A
- Each one of these vectors is a direction in image space
  - What do these look like?



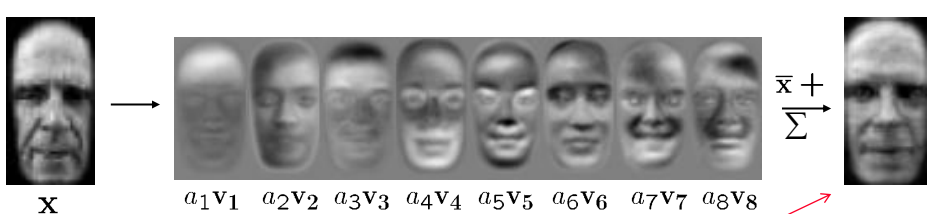
Low Dimensional Representation and Reconstruction

Filtering: Input is converted to eigenvector coordinates using dot products ("projection"):

$$\mathbf{x} \rightarrow (\underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_1}}_{a_1}, \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_2}}_{a_2}, \ldots, \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_K}}_{a_K})$$

Input

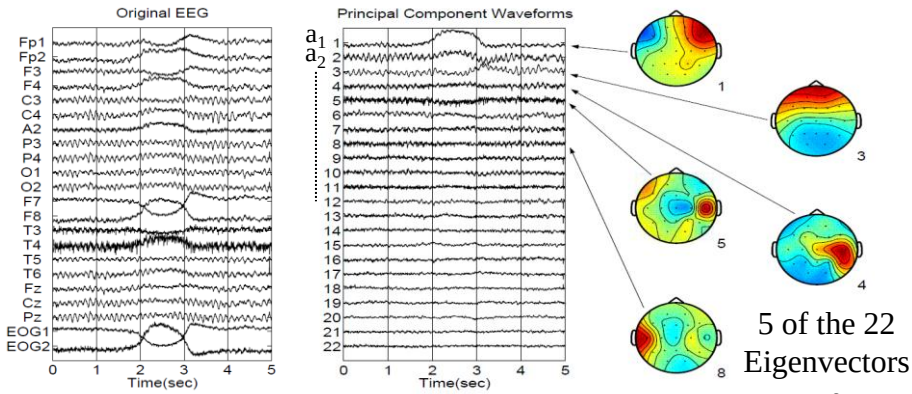
Compressed representation
(K usually much smaller than size of input)



$$\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K}$$

Reconstructed image

PCA applied to EEG



5 of the 22 Eigenvectors (spatial filters for EEG channels)

(Jung et al., 1998)

Independent Component Analysis (ICA)

- PCA produces a matrix V of eigenvectors (columns of V) such that the output **a** is uncorrelated:

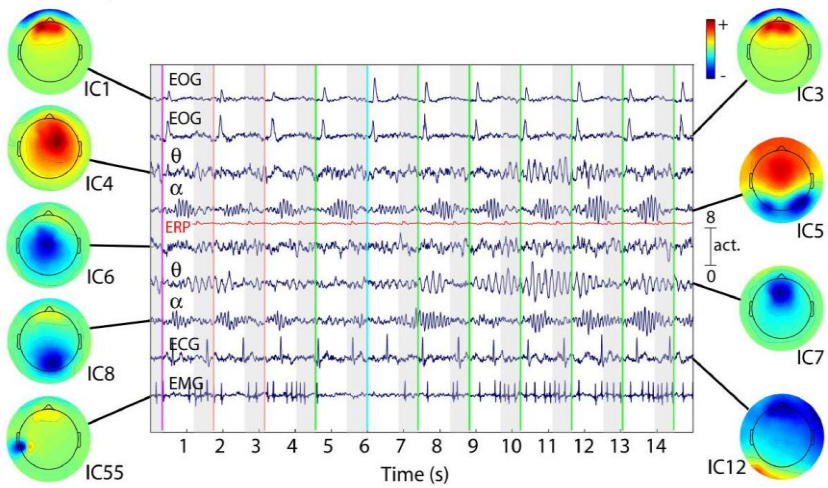
  $\mathbf{a} = V^T(\mathbf{x} - \bar{\mathbf{x}})$

- ICA tries to find a matrix W of filters (columns of W) such that the output **a** is statistically independent:

  $\mathbf{a} = W^T\mathbf{x}$ such that $P(\mathbf{a}) \approx \prod_{i=1}^{M} P(a_i)$

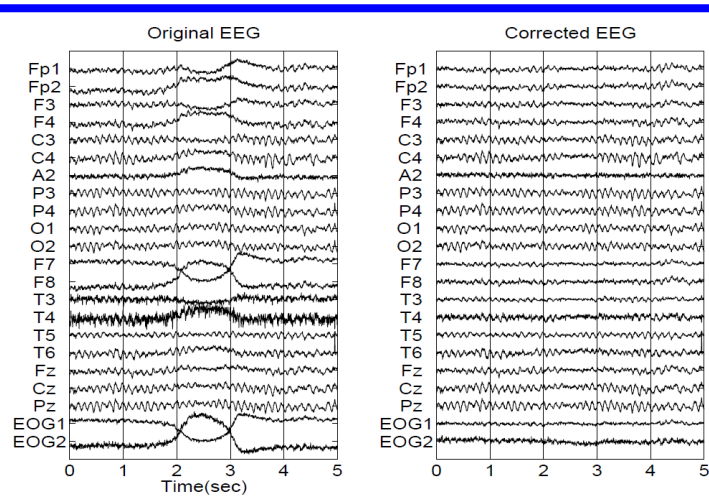
- ICA assumes sources are linearly mixed to produce **x**
- ICA can be used to recover the "independent sources" of mixed brain signals such as EEG

ICA for unmixing EEG signals



http://sccn.ucsd.edu/~scott/icons/TimeCourseFig_web.jpg

ICA for Artifact Removal in EEG



(Jung et al., 1998)

Common Spatial Patterns (CSP)

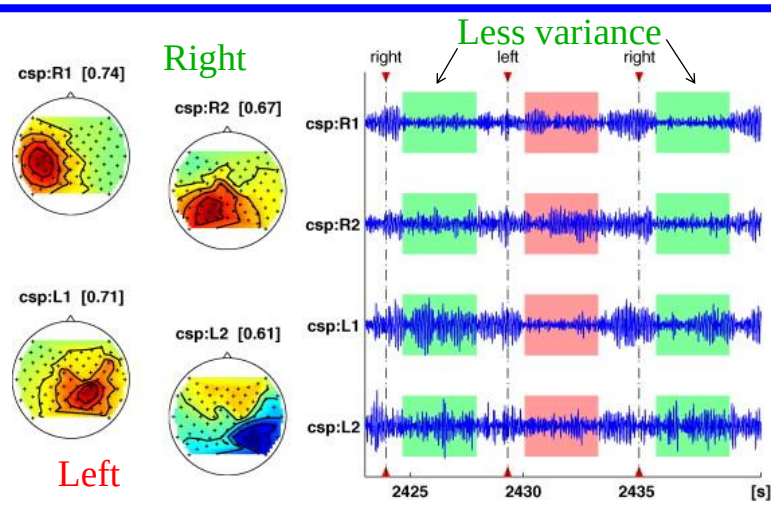
- Data is labeled with class to which each data vector belongs
  - E.g., EEG obtained for right versus left hand imagery
- CSP finds a matrix W of spatial filters (columns of *W*) where

$$\mathbf{a} = \boldsymbol{W}^T \mathbf{x}$$

such that the variance of the filtered data (components of vector **a**) for one class is maximized while the variance of the filtered data for the other class is minimized

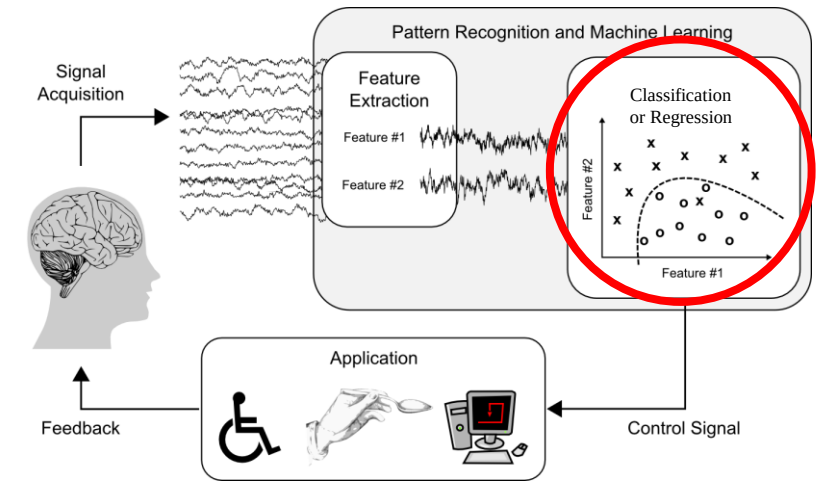
- CSP filters can significantly enhance discrimination ability between the two classes

CSP applied to EEG for Right/Left Hand Imagery



http://www.sciencedirect.com/science/article/pii/S0165027007004657

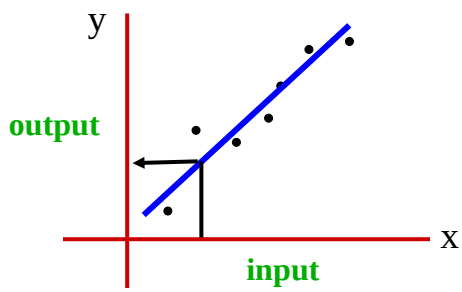
Components of Brain-Computer Interfacing



Outline

- *Regression*
  - Linear
  - Backpropagation neural networks
- *Classification*
  - Linear classifiers
  - Support vector machines
- *Cross-validation*
  - Model selection, preventing overfitting

Linear Regression



x	y
1	3.1
2	6.4
3.1	8.9
0.9	2

Assumption: Output is a linear function of input, i.e.,

$$y_i = wx_i + \text{noise}$$

where noise is independent, gaussian, unknown fixed variance

Linear Regression

Given: Data (y_i, x_i) where y_i are drawn from $N(wx_i, \sigma^2)$

Likelihood of data (y_i, x_i) for a given w is:

$\Pi_i\, p(y_i \mid w, x_i)$ *which is equal to*

$\Pi_i \exp(-0.5\,(y_i - wx_i))^2/\sigma^2$ *(ignoring constants)*

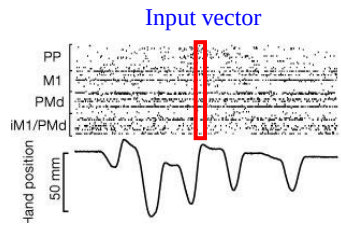
Goal: Maximize the likelihood of data given w

i.e., maximize: $\Sigma_i\ -0.5(y_i - wx_i)^2/\sigma^2$

i.e., minimize: $\Sigma_i\,(y_i - wx_i)^2$

Can show that $w = \Sigma\, x_i\, y_i\, /\, \Sigma\, (x_i)^2$

But…typically, inputs in BCIs are ***vectors*** of multiple neurons' activities, multiple EEG measurements, etc.



Need Multivariate Regression

Multivariate Linear Regression

Suppose inputs $\mathbf{x}_i$ are n-element vectors: $y_i = \mathbf{w}^T\mathbf{x}_i$ + noise

Write the *m* inputs as rows of matrix and outputs y_i as vector:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

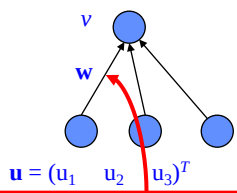
Then, $\mathbf{Y} = \mathbf{Xw} + \mathbf{noise}$

Find maximum likelihood $\mathbf{w}$ by minimizing $\|\mathbf{Y} - \mathbf{Xw}\|^2$ over all data

$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$

Nonlinear Regression: Neural Networks

$$v = g(\mathbf{w}^T\mathbf{u})$$
$$= g(w_1u_1 + w_2u_2 + w_3u_3)$$



Output

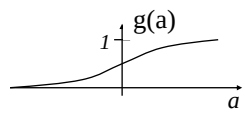
Input nodes

Want to learn a mapping from inputs to outputs, given training data ($\mathbf{u}^m$,d^m). How is **w** learned?

The most common activation function:

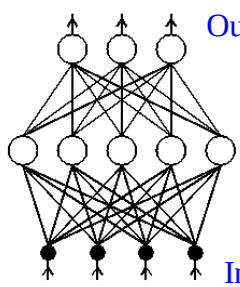
Sigmoid function:

$$g(a) = \frac{1}{1+e^{-\beta a}}$$



Squashes $\mathbf{w}^T\mathbf{u}$ to be between 0 and 1. The parameter β controls the slope.

Multilayer Networks



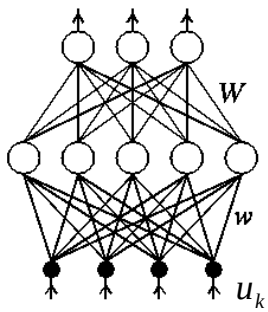
Output $\mathbf{v} = (v_1\ v_2\ \ldots\ v_J)^T$; Desired = **d**

How do we learn these weights?

Input $\mathbf{u} = (u_1\ u_2\ \ldots\ u_K)^T$

Idea: "Backpropagation" Learning Rule

$$v_i = g(\sum_j W_{ji} g(\sum_k w_{kj} u_k))$$



Start with random weights {**W**, **w**}

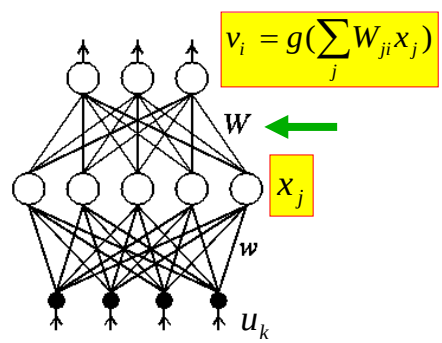
Given input **u**, network produces output **v**

Find **W** and **w** that minimize total squared output error over all output units (labeled *i*):

$$E(\mathbf{W}, \mathbf{w}) = \frac{1}{2} \sum_i (d_i - v_i)^2$$

Backpropagation: Output Weights

$$E(\mathbf{W}, \mathbf{w}) = \frac{1}{2} \sum_i (d_i - v_i)^2$$



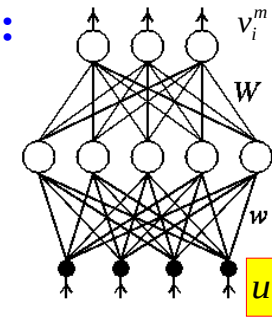
Learning rule for hidden-output weights **W**:

$$W_{ji} \rightarrow W_{ji} - \varepsilon \frac{dE}{dW_{ji}} \qquad \{\textit{gradient descent}\}$$

$$\frac{dE}{dW_{ji}} = -(d_i - v_i) g'(\sum_j W_{ji} x_j) x_j$$

Backpropagation: Hidden Weights

$$E(\mathbf{W},\mathbf{w}) = \frac{1}{2}\sum_i (d_i - v_i)^2$$



$$v_i^m = g(\sum_j W_{ji} x_j)$$

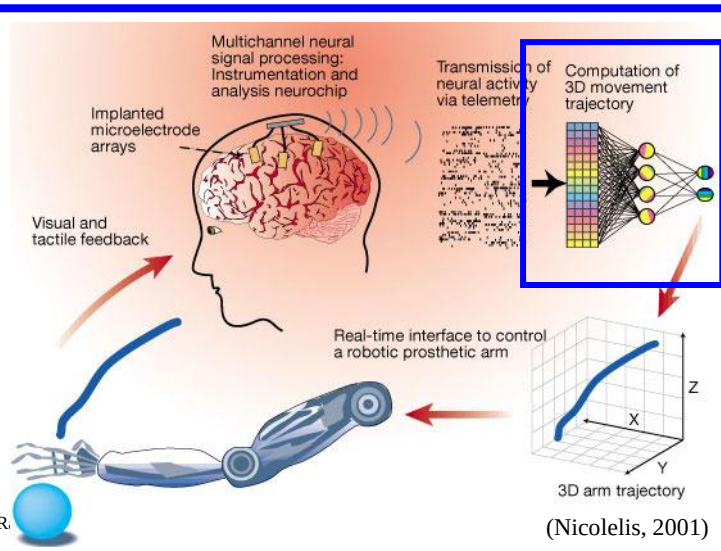
$$x_j^m = g(\sum_k w_{kj} u_k^m)$$

Learning rule for input-hidden weights **w**:

$$w_{kj} \rightarrow w_{kj} - \varepsilon \frac{dE}{dw_{kj}} \quad \text{But}: \frac{dE}{dw_{kj}} = \frac{dE}{dx_j} \cdot \frac{dx_j}{dw_{kj}} \quad \{\textit{chain rule}\}$$

$$\frac{dE}{dw_{kj}} = \left[-\sum_{m,i} (d_i^m - v_i^m) g'(\sum_j W_{ji} x_j^m) W_{ji} \right] \cdot \left[g'(\sum_k w_{kj} u_k^m) u_k^m \right]$$

Example Application in BCI



(Nicolelis, 2001)

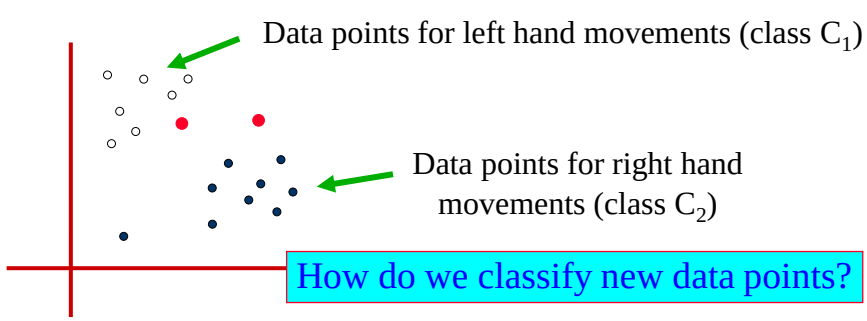
Motivation: Why classification?

- Many BCI applications require selecting 1 out of N commands or menu choices. E.g.,
  - Word spellers: select 1 out of 26 letters
  - Menu with icons: select 1 out of N icons
  - Semi-autonomous robot: select 1 out of N high-level commands
- Classification techniques can used to classify a given brain signal into 1 of several classes
  - Simplest case: 2 classes ⇨ **binary classificiation**

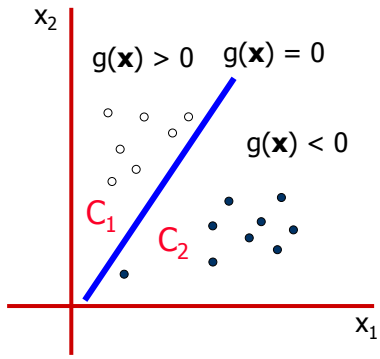
Binary Classification: Example

Suppose BCI is used to move a 1D cursor left or right

Want to use brain signals for Imagined left hand movement vs. Imagined right hand movement



Binary Classification: Linear Classifiers



Find a line (in general, a hyperplane) (**w**, b) separating the two sets of data points:

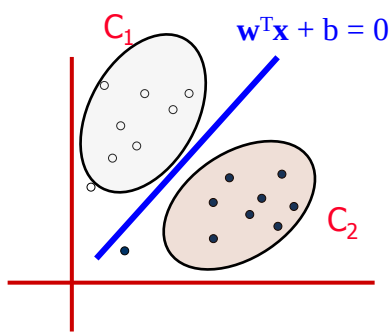
$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$, i.e.,

$w_1x_1 + w_2x_2 + b = 0$

For any new point **x**, choose:

class C_1 if $g(\mathbf{x}) > 0$ and class C_2 otherwise

Linear Discriminant Analysis (LDA)



Assume each class is a *Gaussian cloud*

$P(\mathbf{x}|C_i) = N(\mu_i, \Sigma)$, i = 1,2

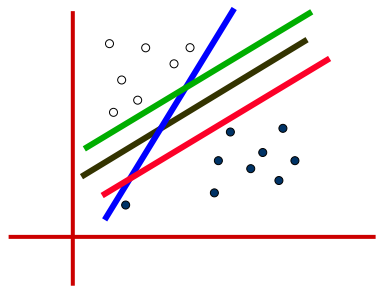
w and b are computed as follows:

$\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2)$

$b = \mathbf{w}^T(\mu_1 + \mu_2)/2$

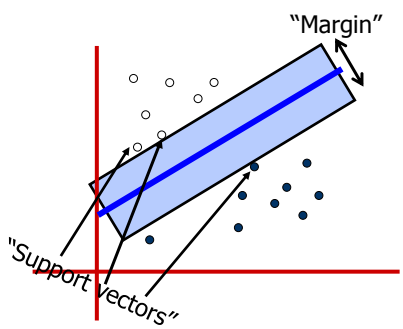
Can show that this is equivalent to log likelihood ratio test for class 1 versus class 2

Actually, many possible separating lines…



Which one is best?

Support Vector Machines (SVMs)



Choose hyperplane with largest margin

Facilitates generalization to new data points

Can be shown: margin = 2/ $w^T w$

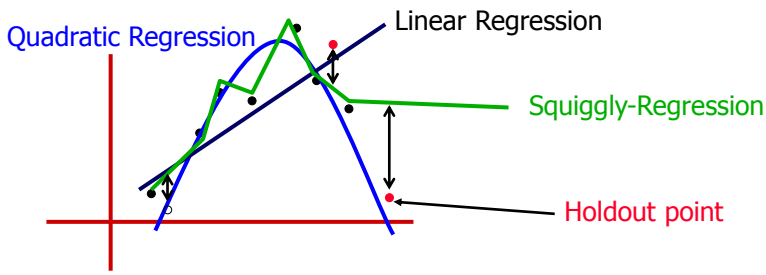
Maximize margin subject to the following constraints:

$w^T x_i + b > +1$ for class1 points,

$w^T x_i + b < -1$ for class2 points.

Solved with Quadratic Programming.

Overfitting and Generalization



Question: Which line is the best fit?
Squiggly line *overfits* the data ⇒*Won't generalize* to new data
Solution: *Hold out* some of the points, test regression or classification performance on the held-out points
What if points held out are bad (uncharacteristic of new data)?

Cross-Validation and Model Selection

- Repeat for different subsets of held-out points:
  For each choice of held-out points:
  - Fit model on remaining points
  - Evaluate error on held-out points
  - Add to total error score (this is the *generalization error*)
- Choose model with *least generalization error*

Types of Cross-Validation

- K-fold cross-validation: Split data into *k* blocks. *Each block* is used as hold out set.
- Leave-one-out cross-validation: *Each point* is held out once, model trained on remaining points, tested on held out point, and cycle repeated for all points.

Next Class:
Invasive BCIs – Early Studies

Presenters:
Shin Kira
&
Miah Wander