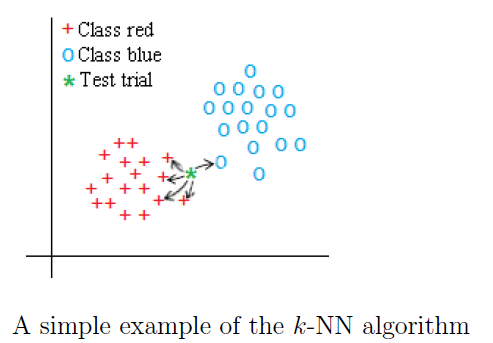
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s/no | Research  Item | Filtering  Technique | Feature Extraction Technique | Feature Parameter | Classification Technique | Classification  Variable |
| 1. | Adaptive online brain-computer interface for interpretation and visualization of desired reach | high-pass and low-pass FIR filters,  ICA,  CSP | subsampling, frequency filtering, channel scaling, channel selection, spatial filtering, frequency  decomposition (AR), and post-processing,  Power estimates using filter bank | Slow Cortical Potential  Movement Related Potential | support vector machines  *L*1-Regularized Logistic Regression  Meta-classifier | left versus right hand self-paced typing |
| 2. | Accurate hand trajectory prediction by real and  Synthetic EEG | Stable elliptic filter | Brain Model for generating synthetic EEG |  | A model in which the hand position coordinates  (the dependent variable) are written as a function  of the neural activity (independent variable). | The hand coordinates |
| 3. | ENGINEERING THE BRAIN SIGNALS – PREPROCESSING | FIR equiripple stable filter | PCA | Spectral band power | SVM | left small finger or the tongue |
| 4. | EEG single-trial classification of four classes of imaginary  Wrist movements based on gabor coefficients | EOG was removed using ICA,  High pass, low pass and stopband filters | Gabor  transformation for  features |  | Recurrent Elman neural networks | four direction distinctive imaginary wrist movements |
| 5. | Crossectional investigation of wrist movement intention  Classification in eeg signals | Band-specific Butterworth zero-phase filters (6th and 12th orders) |  | Amplitude variance of the signal  Windowed amplitude variance of the signal  Maximum/minimum power and dominant frequency  of autocorrelation  6th order autoregressive model, 6 coefficients and noise variance  4th order autoregressive moving average model, 4 coefficients and noise variance  Total signal power | Multilayer Perceptron | Right and Left hand movements |
| 6. | Decoding Individual Finger Movements from One Hand  Using Human EEG Signals | Elliptic IIR 3Hz HPF (with forward and reverse filtering to avoid distortion)  60Hz notch filter for power line removal with the transition band of 0.3Hz  ICA for artifact rejection | Movement related spectral changes | PCA applied to Power Spectral Density data to determine weight of movement variations | Support Vector Machine with radial basis kernel basis from LIBSVM package | Different finger movements decoding |
| 7. | Reconstructing Three-Dimensional Hand Movements from  Noninvasive Electroencephalographic Signals | low-pass, antialiasing filter with a cutoff frequency  of 40 Hz |  |  |  |  |
| 8. | Brain EEG Signal Processing For Controlling a Robotic Arm | Band pass filter 0.5-45Hz 5th order Butterworth | Three movements (close, open arm and close hand) | Wavelet Transform (WT), Fast Fourier Transformation (FFT) and Principal Component Analysis (PCA) | Back Propagation (BP) Neural Network |  |
| 9 | Relationship between Speed and EEG Activity during Imagined  and Executed Hand Movements | Band-pass filtered from 1 Hz to 30 Hz using a zero-phase FIR filter | Speed and hand information as variables | linear model and linear regression with wavelet analysis |  |  |
| 10. | How Many People are Able to Operate an EEG-Based  Brain-Computer Interface (BCI)? | Band pass filtered 0.5 to 30Hz | AAR (recursive-least-squares) and Band power estimation | Power spectral dynamics | LDA | Right hand and both feet |
| 11. | Comparison of Different Classifiers for Brain Computer Interface |  | Welch Algorithm for power spectrum Analysis (8-30Hz) | Power spectral density | Mahalanobis Distance,  Hidden Markov Models (Baum-Welch Algorithm),  ANN | Imagined right and left hand movement |
| 12. | COMPARING COMMON MACHINE LEARNING CLASSIFIERS IN LOW-DIMENSIONAL FEATURE VECTORS FOR BRAIN COMPUTER INTERFACE APPLICATIONS |  | Band power estimation | PSD | *k*-NN, SVM, LDA, NB and DT | Classification Accuracy  Sensitivity and Specificity  Kappa  Computational time |
| 13. |  |  |  |  |  |  |

[12] The experiments proved that it is difficult to propose a firm classification algorithm. Based on the results from Tables 1 and 2, it seems selection of the most appropriate classifier highly depends on structure of the data set.

|  |  |  |
| --- | --- | --- |
| Classification Technique | Merit | Demerit |
| SVM |  | Slow Computational time (due to tune parameters) |
| LDA | Fast computational time |  |
| Naïve Bayes (NB) | Fast computational time |  |
|  |  |  |
| k-Nearest Neighbor (K-NN) |  |  |

|  |  |  |
| --- | --- | --- |
| Feature Extraction Technique | Merit | Demerit |
|  |  |  |
|  |  |  |
|  |  |  |

**k-Nearest Neighbor***.* The *k*-NN classifier is a common classification algorithm, which determines a testing sample's class by the majority class of the *k* closest training samples. This is illustrated with a simple example in Figure 5, which shows data records, each with two attributes that are representations of two classes of data (blue and red). In this case *k* = 5. The unlabeled test trial would be labeled by the category of the class red, because four out of its five closest samples (neighbors) are red. It is worth the mention that the performance of a *k*-NN algorithm depends on the distance metric and the value of *k*. In our study, we used Euclidean distance metric and leave-one-out cross-validation (LOOCV) technique to determine the best value of *k* to maximize the classification performance. The *k* value was searched in interval between 1 and 15, with step size of 1.



***Linear discriminant analysis****.* LDA classifies two classes based on the assumption that both classes are under normal distribution with equal covariance matrices. The separating hyper plane is obtained by finding the projection of the labeled training data that maximizes the distance between the two classes' means and minimizes the interclass variance. The main aim is to solve the problem

*y* = *wT x* + *w*0*;*

where *x* is the feature vector. The vectors *w* and *w*0 are determined by maximization of the interclass means and minimization of interclass variance

***Naive Bayes****.* Naive Bayes classifier is a simple probabilistic algorithm based on applying Bayes' theorem with naive independence assumptions. Consider a set of training trials where each trial is made up from *m* discrete-valued features and a class from a finite set *C*. The naive Bayes classifier can probabilistically predict the class of an unknown trial using the available training trial set to calculate the most probable output.

The most probable class *CNB* of an unknown trial with the conjunction *A* = *a*1*, a*2*, . . . am* is calculated by:

*CNB* = arg max *p*(*c\A*)*.*