Carnegie Mellon

18-660: Numerical Methods for Engineering Design and Optimization

Xin Li
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213

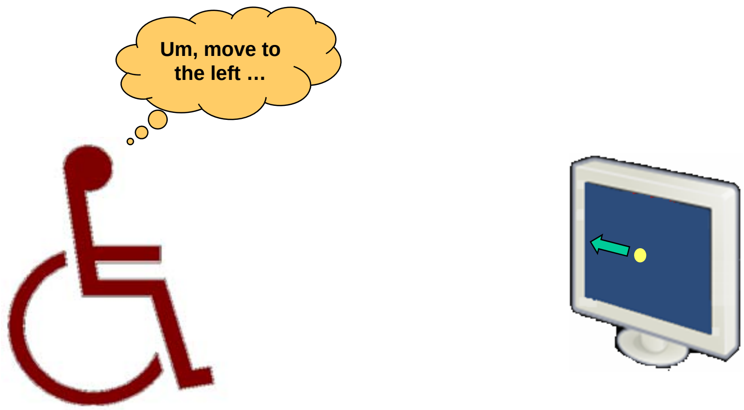
Electrical & Computer ENGINEERING

Outline

- Project 3: Movement Decoding for Brain Computer Interface
  - Background
  - Methods
  - Submission details

Brain Computer Interface (BCI)

- BCI is a means of establishing direct communication pathway between brain and computers



Supervised Learning

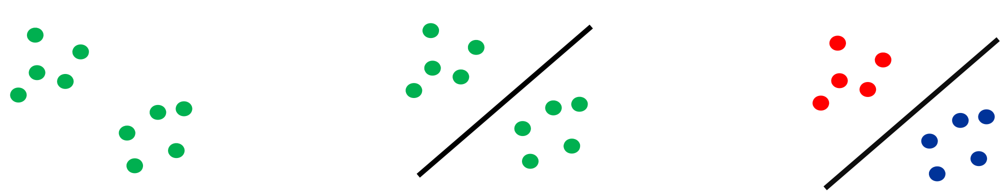
Training set

↓

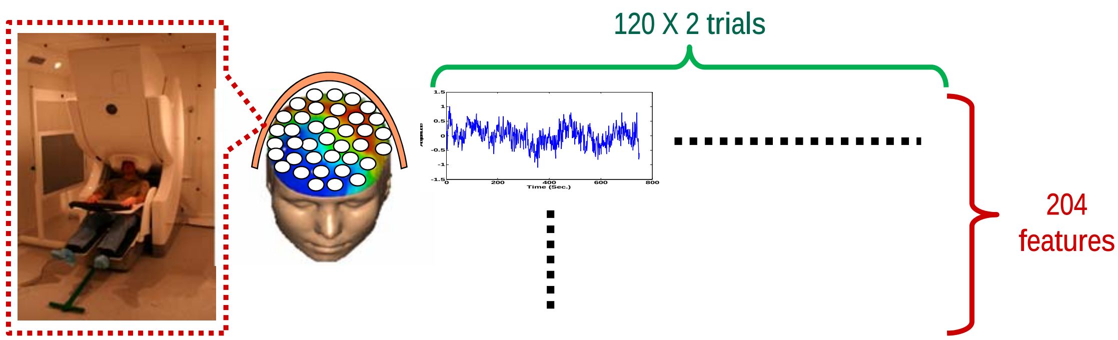
Learning algorithm

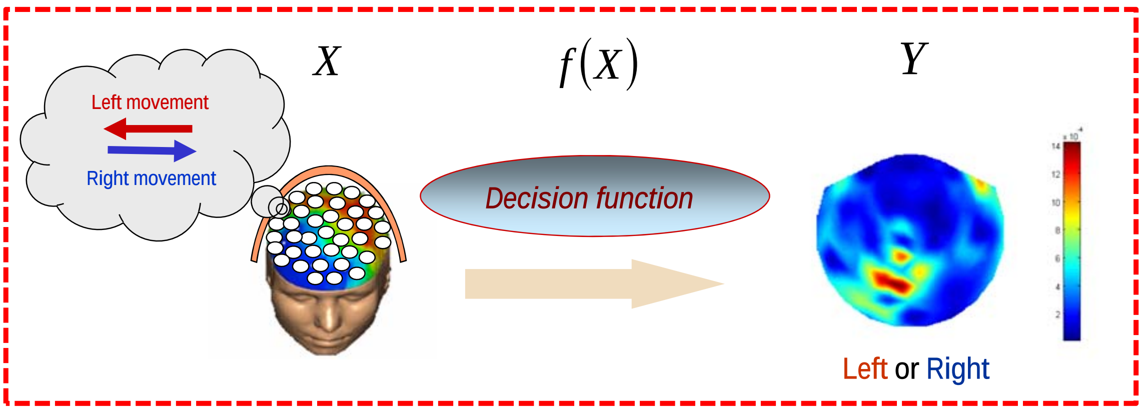
↓

X: Testing set → ***f(X)*: Hypothesis** → *Y*: Prediction



BCI Data Classification

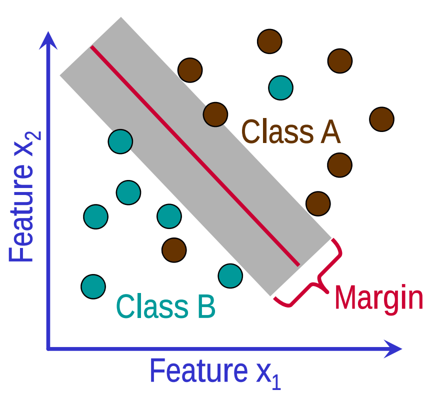




Support Vector Machine (SVM)

- Support vector machine (SVM) is a popular algorithm used for classification.
  - Key idea: maximize classification margin

- Two-class linear support vector machine



Decision function:

$$f(X) = W^T X + C \begin{cases} \geq 0 & (Class\ A) \\ < 0 & (Class\ B) \end{cases}$$

Features

Determine W and C with maximum margin

$$\min_{W,C,\xi} \sum \xi_i + \lambda \cdot W^T W$$

$$\text{S.T.} \quad y_i \cdot \left(W^T X_i + C\right) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

$$(i = 1, 2, \cdots, N)$$

Objective

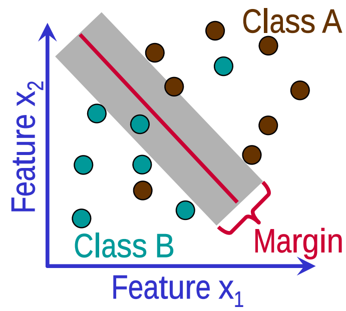
- Implement a two-class SVM classifier
- Apply the SVM classifier to decode BCI features into two classes (i.e. Left vs. Right)

Outline

- Background
- Methods
  - Interior point method
  - Newton method
  - Line search
  - Two-level cross validation
- Submission details

Two-class Linear Support Vector Machine

- Convex optimization problem



$$\min_{W,C,\xi} \sum \xi_i + \lambda \cdot W^T W$$
$$\text{S.T.} \quad y_i \cdot \left(W^T X_i + C\right) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$
$$\left(i = 1, 2, \cdots, N\right)$$

- Logarithmic barrier

$$\min_{W,C,\xi} \sum \xi_i + \lambda \cdot W^T W - \frac{1}{t}\sum_{i=1}^{N} \log\left(W^T X_i \cdot y_i + C \cdot y_i + \xi_i - 1\right) - \frac{1}{t}\sum_{i=1}^{N} \log\left(\xi_i\right) \quad (**)$$
$$t \to \infty$$

Interior Point Method

- Optimization problem

$$\min_{W,C,\xi} \sum \xi_i + \lambda \cdot W^T W - \frac{1}{t}\sum_{i=1}^{N} \log\left(W^T X_i \cdot y_i + C \cdot y_i + \xi_i - 1\right) - \frac{1}{t}\sum_{i=1}^{N} \log\left(\xi_i\right) \quad (**)$$
$$t \to \infty$$

- Interior point algorithm
  - Select an initial value of t and ***an initial guess*** $[W^{(0)}\ C^{(0)}\ \boldsymbol{\xi}^{(0)}]$
  - Repeat:
    - 1. solve the ***unconstrained nonlinear optimization*** problem (**) to find the optimal solution $[W^*\ C^*\ \boldsymbol{\xi}^*]$
    - 2. $[W^{(0)}\ C^{(0)}\ \boldsymbol{\xi}^{(0)}] = [W^*\ C^*\ \boldsymbol{\xi}^*]$ and $t = \beta t$ where $\beta = 15$
    - 3. Stopping criterion: quit if $t \geq T_{max}$ where $T_{max} = 1000000$

Initial Guess

- Initial guess $[W^{(0)}\ C^{(0)}\ \xi^{(0)}]$
  - A feasible solution satisfying the constraints:

$$\begin{cases} y_i \cdot \left(W^{(0)T} X_i + C^{(0)}\right) > 1 - \xi_i^{(0)} \\ \xi_i^{(0)} > 0 \end{cases} \quad (i = 1, 2, \cdots, N)$$

  - Set $[W^{(0)}\ C^{(0)}]$ to an arbitrary value and assign $\xi^{(0)}$ to

$$\xi_i^{(0)} = \max\left\{1 - y_i \cdot \left(W^{(0)T} X_i + C^{(0)}\right), 0\right\} + 0.001$$

Solve Unconstrained Nonlinear Optimization

$$\min_{W,C,\xi} \quad \underbrace{\underbrace{\sum \xi_i + \lambda \cdot W^T W - \frac{1}{t}\sum_{i=1}^{N} \log\left(W^T X_i \cdot y_i + C \cdot y_i + \xi_i - 1\right) - \frac{1}{t}\sum_{i=1}^{N}\log(\xi_i)}_{f(W,C,\xi)}}_{f(Z)}$$

- Newton method
  - Start from an initial value $Z^{(0)} = [W^{(0)}\ C^{(0)}\ \xi^{(0)}]$
  - Repeat:
    - 1. Compute the Newton step and decrement
      $\Delta Z = -\nabla^2 f(Z)^{-1} \cdot \nabla f(Z) \quad \sigma = \nabla f(Z)^T \cdot \nabla^2 f(Z)^{-1} \cdot \nabla f(Z)$
    - 2. Stopping criterion: quit if $\sigma / 2 \le \varepsilon$ where $\varepsilon = 0.000001$
    - 3. ***Line search***: choose step size s by backtracking line search
    - 4. Update: $Z = Z + s \cdot \Delta Z$

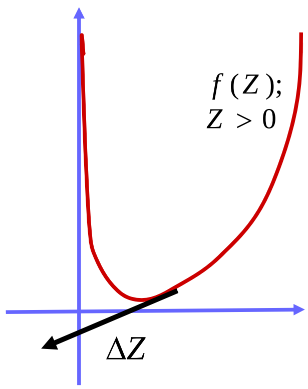
Line Search

- Choose step size s: $\left(Z + s \cdot \Delta Z\right) \in dom\left(Z\right)$

$$Z + s \cdot \Delta Z = [W^{(k)}, C^{(k)}, \xi^{(k)}] \in dom\left(Z\right)$$

$$\Updownarrow$$

$$\begin{cases} W^{(k)T} X_i \cdot y_i + C^{(k)} \cdot y_i + \xi_i^{(k)} - 1 > 0 \\ \xi_i^{(k)} > 0 \end{cases} \quad \left(i = 1,2,\cdots,N\right)$$



- Backtracking line search
  - Start at *s* = 1
  - Repeat:
    - 1. Stopping criterion: quit if

$$\begin{cases} W^{(k)T} X_i \cdot y_i + C^{(k)} \cdot y_i + \xi_i^{(k)} - 1 > 0 \\ \xi_i^{(k)} > 0 \end{cases} \quad \left(i = 1,2,\cdots,N\right)$$

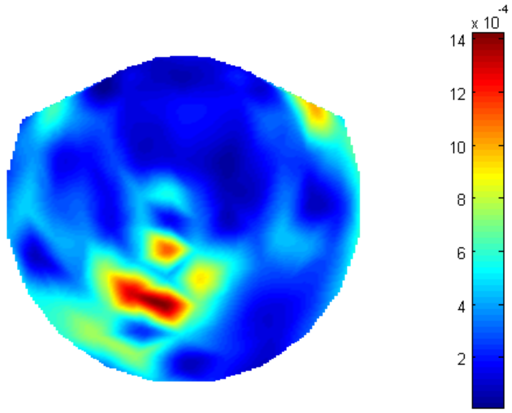
    - 2. $s = 0.5 \cdot s; \quad [W^{(k)}, C^{(k)}, \xi^{(k)}] = Z + s \cdot \Delta Z$

Summary of SVM Solver

- Apply interior point method to solve SVM problem
  - At each iteration, apply Newton method to solve an unconstrained optimization with logarithmic barrier
  - At each Newton iteration, choose optimal step size by backtracking line search
- Please pay attention to the following important rules
  - You must implement the SVM solver by yourself – it is not allowed to use MATLAB functions such as *svmclassify*
  - You must apply interior point method to solve the SVM problem – it is not allowed to use the other algorithms such as directly solving the dual problem

Channel Weight

- Each element in *W* corresponds to the weight of a channel
  - Large amplitude: the channel carries strong directional information
- Plot the spatial map of channel weight on brain surface
  - Use the function we provide: show_chanWeights(abs(W))



Cross Validation

- Report testing accuracy by *six-fold* cross validation
  - Data set: 120×2 trials
  - Divide them into six folds: 20×2 trials per fold
    - Training data: 100×2 trials
    - Testing data: 20×2 trials
  - Testing accuracy of each fold: $Ac(i) \quad (i = 1, 2, \cdots, 6)$
- Calculate mean
  - $\overline{Ac} = \sum_{i=1}^{6} Ac(i)/6$
  - MATLAB function: mean(Ac)
- Calculate standard deviation
  - $stdAc = \sqrt{\frac{1}{6}\sum_{i=1}^{6}\left[Ac(i) - \overline{Ac}\right]^2}$
  - MATLAB function: std(Ac)

Determine λ by Second-Level Cross Validation

- For each run of the first-level cross validation
  - Use *five-fold* cross validation inside the training data (100×2 trials) to determine optimal λ of SVM
    - Try at least $\lambda \in \{0.01, 1, 100, 10000\}$
  - Divide the training data into five folds: 20×2 trials per fold
    - Training data: 80×2 trials
    - Testing data: 20×2 trials

Two-Level Cross Validation Summary

Outline

- Background
- Methods
- Submission details

Project Files

- All files for this project can be found from the distributed package
  - A MATLAB function: show_chanWeights.m
  - Three MATLAB function templates: getOptLamda_temp.m, solveOptProb_NM_temp.m and costFcn_temp.m
  - A data file defining sensor locations: sensors102.mat
  - Two data sets: feaSubEOvert.mat and feaSubEImg.mat
  - A report template: Proj3.doc

MATLAB Functions

- getOptLamda.m
  - Compute the optimal λ
  - Input: data, label and initial parameters
  - Output: optimal value of λ
- solveOptProb_NM.m
  - Compute the optimal solution using Newton method
  - Input: function handle, initial function value and tolerance
  - Output: optimal solution and error value
- costFcn.m
  - Compute the function value, gradient and Hessian
  - Input: initial function value
  - Output: function value, gradient and Hessian
- Templates for these three functions are provided

MATLAB Functions

- show_chanWeights.m
  - Show the spatial map of channel weight
    - e.g. show_chanWeights(absW)
  - Input: a 204×1 vector
    - Absolute value of the weight vector W
  - Output: a MATLAB figure as shown on Slide 15
- This function is provided in the distributed package
  - You do not need to implement it by yourself

Data Sets

- Each .mat file contains one variable which is a <1×2 cell>
  - class{1}: a 204×120 matrix containing data from the first class
    - There are 120 trials in total
    - Each trial is represented by a 204×1 feature vector
    - class{1}(:,i) represents the feature vector of the i-th trial
  - class{2}: a 204×120 matrix containing data from the second class
    - There are 120 trials in total
    - Each trial is represented by a 204×1 feature vector
    - class{2}(:,i) represents the feature vector of the i-th trial

Project Submission

- You should zip the MATLAB code (.m) into a single file and submit it to the course web site
  - Your code must work on Linux cluster without any modification

- You should also submit a PDF report (at most 4 pages) to the course web site
  - Follow instructions specified in the WORD template

Grading Criteria

- 75% for MATLAB code and results
- 25% for project report