18-660: Numerical Methods for Engineering Design and Optimization

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

Electrical & Computer ENGINEERING

Overview

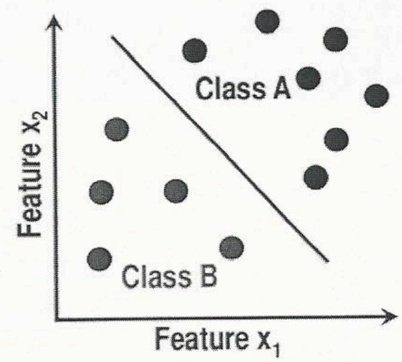
- **Classification**
  - Support vector machine
  - Regularization

Classification

- Predict categorical output (i.e., two or multiple classes) from input attributes (i.e., features)

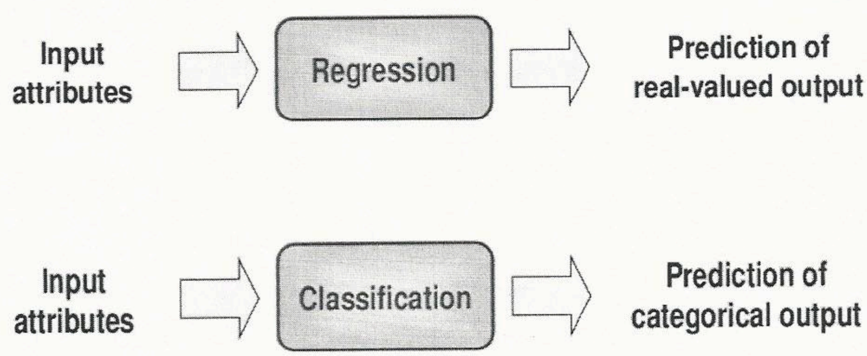
- Example: two-class classification

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



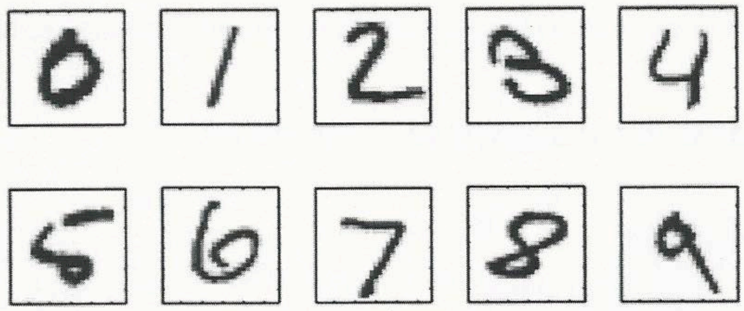
Classification

- Classification vs. regression



Classification Examples

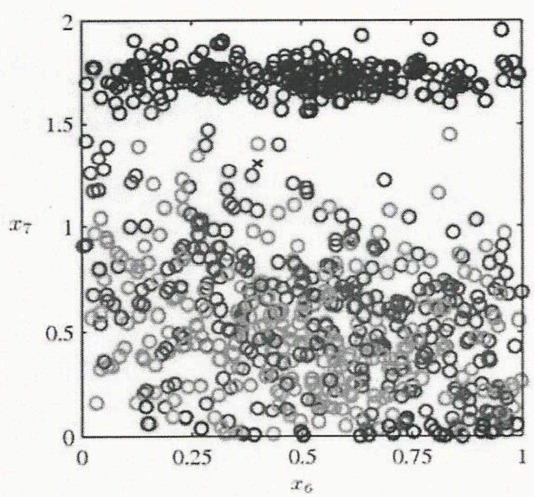
- **Identify hand-written digits from US zip codes**



Bishop, Pattern recognition and machine learning, 2007

Classification Examples

- **Identify geometrical structure from oil flow data**



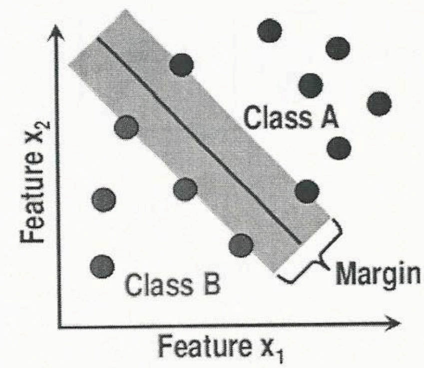
Blue: geometrical structure 1
Green: geometrical structure 2
Red: geometrical structure 3

Bishop, Pattern recognition and machine learning, 2007

Support Vector Machine (SVM)

- **Support vector machine (SVM) is a popular algorithm used for many classification problems**
  - Key idea: maximize classification margin (immune to noise)

- **Two-class linear support vector machine**

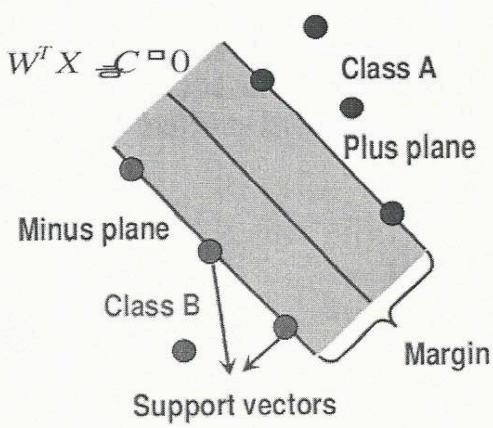


$$f(X) = W^T X + C \begin{cases} > 0 & (\text{Class } A) \\ < 0 & (\text{Class } B) \end{cases}$$

Determine W and C with maximum margin

Margin Calculation

- **To maximize margin, we must first represent margin as a function of W and C**



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

plus plane: $W^T x + C = 1$
Minus plane $W^T x + C = -1$

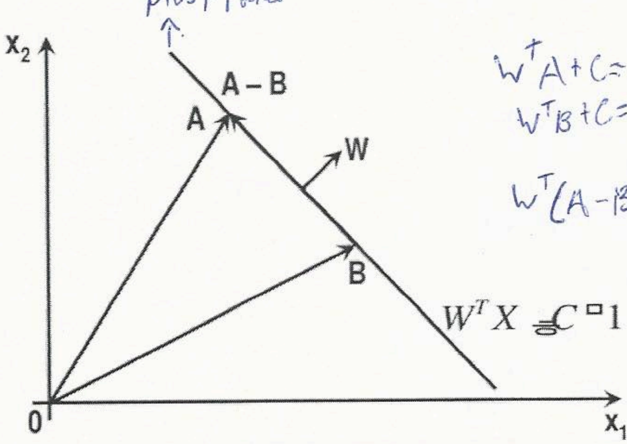
$\frac{W^T}{K} x + \frac{C}{K} = \frac{K}{K} = 1$

Margin Calculation

- **W is perpendicular to plus/minus planes**

Plus plane $W^T X + C = 1$

Minus plane $W^T X + C = -1$



$W^T A + C = 1$

$W^T B + C = 1$

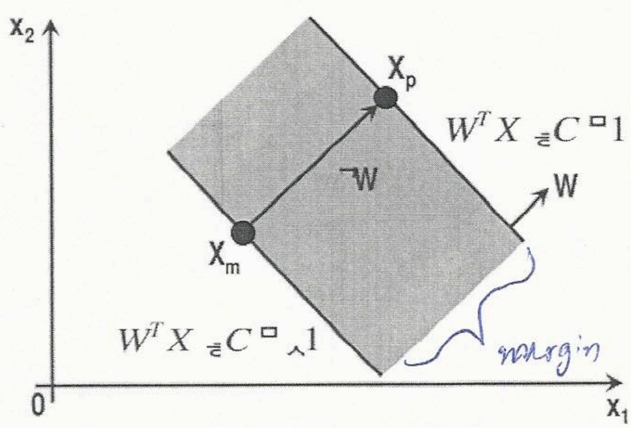
$W^T (A - B) = 0$

Margin Calculation

- **Margin equals to the distance between X_m and X_p**

$X_p = X_m + \lambda \cdot W$

$\text{Margin} = \|X_p - X_m\|_2 = \|\lambda W\|_2$



Margin Calculation

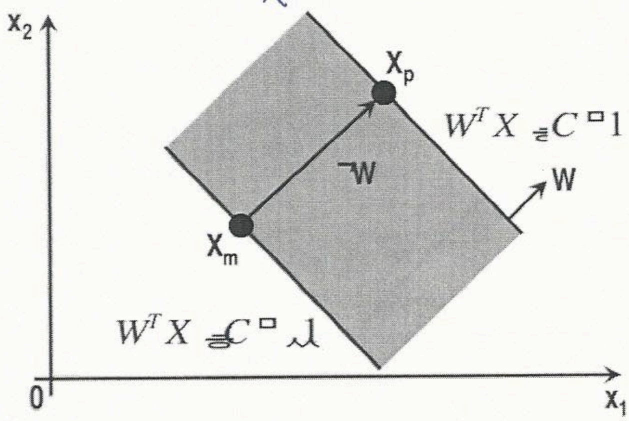
$X_p \square X_m = {}^{-}W$ $\rightarrow x_p - x_m = \lambda \cdot w$

$\rightarrow W^T X_p = C \square 1$
$\rightarrow W^T X_m = C \square 1$

$w^T (x_p - x_m) = 2$

$w^T \cdot \lambda w = 2$

$\lambda \cdot w^T w = 2$



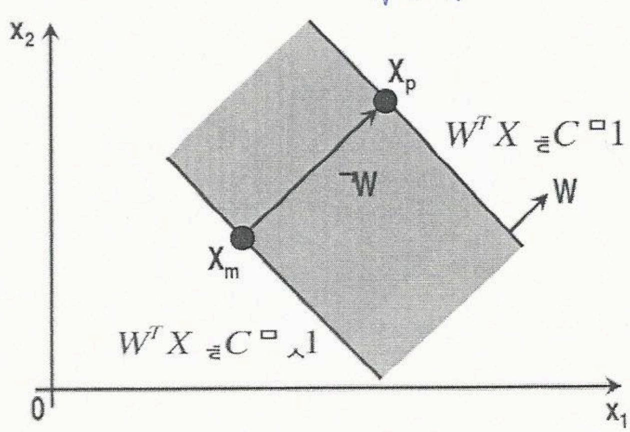
Margin Calculation

$\lambda = \frac{2}{w^T w}$

${}^{-}W^T W \square 2$

$\text{Margin} = \|\lambda w\|_2 = \lambda \cdot \|w\|_2$

$= \frac{2}{w^T w} \cdot \sqrt{w^T w} = \frac{2}{\sqrt{w^T w}}$

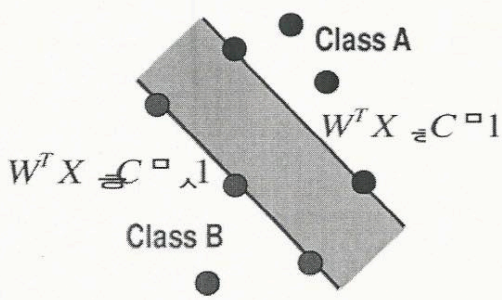


Mathematical Formulation

- **Start from a set of training samples**

$$\{X_i, y_i\} \quad (i = 1, 2, \cdots, N)$$

X_i:	input feature of i-th sampling point
y_i:	output label of i-th sampling point
	Class A $\Rightarrow y_i = 1$
	Class B $\Rightarrow y_i = -1$



Class A:

$w^T x_i + C \geq 1 \qquad y_i = 1$

$y_i (w^T x + C) \geq 1$

Class B:

$w^T x_i + C \leq -1 \qquad y_i = -1$

$y_i (w^T x + C) \geq 1$

Mathematical Formulation

- **Formulate a convex optimization problem**

$$\max_{W,C} \frac{2}{\sqrt{W^T W}} \longrightarrow \text{Maximize margin}$$

$$\text{S.T.} \quad y_i \cdot (W^T X_i + C) \geq 1 \longrightarrow \text{All data samples are in the right class}$$

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

$\min_{w,C} \; w^T w$ → Convex quadratic function

S.T. $y_i (w^T x_i + C)$ → Linear constraints

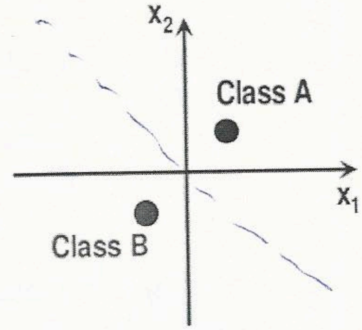
$(i = 1, 2, \ldots, N)$

A Simple SVM Example

- **Two training samples**
  - Class A: $x_1 = 1$, $x_2 = 1$ and $y = 1$
  - Class B: $x_1 = -1$, $x_2 = -1$ and $y = -1$

$$f(X) = w_1x_1 + w_2x_2 + C \begin{cases} > 0 & (Class\ A) \\ < 0 & (Class\ B) \end{cases}$$

Solve w_1, w_2 and C to determine classifier

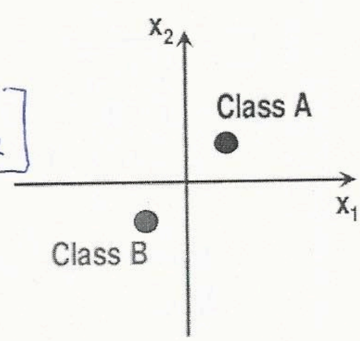


A Simple SVM Example

- **Two training samples**
  - Class A: $x_1 = 1$, $x_2 = 1$ and $y = 1$
  - Class B: $x_1 = -1$, $x_2 = -1$ and $y = -1$

$$\left\{ \begin{array}{ll} \min\limits_{W,C} & W^T W \\ \text{S.T.} & y_i (W^T X_i + C) \geq 1 \\ & (i = 1, 2, \ldots, N) \end{array} \right.$$

$w = \begin{bmatrix} w1 \\ w2 \end{bmatrix}$



$$\min_{w,C} \; w_1^2 + w_2^2$$

$$\text{s.t.} \quad 1 \cdot (w_1 + w_2 + C) \geq 1$$

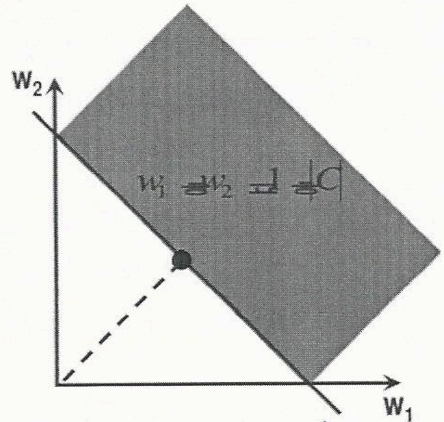
$$-1 \cdot (-w_1 - w_2 + C) \geq 1$$

A Simple SVM Example

$$\min_{w,C} \; w_1^2 + w_2^2$$

$$\text{S.T.} \quad 1(w_1 + w_2 + C) \ge 1$$

$$-1(-w_1 - w_2 + C) \ge 1$$



$\min_{w,C} \; w_1^2 + w_2^2$

S.T. $w_1 + w_2 \ge 1 - C$

$w_1 + w_2 \ge 1 + C$

$\hookrightarrow w_1 + w_2 = 1$

$\min_{w,C} \; w_1^2 + w_2^2$

S.T. $w_1 + w_2 \ge 1 + |C|$

$w_1 = w_2 = 0.5$

$C = 0$

A Simple SVM Example

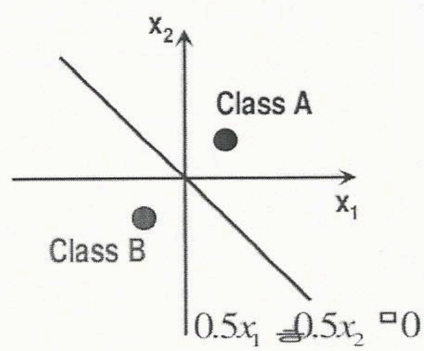
- **Two training samples**
  - Class A: $x_1 = 1$, $x_2 = 1$ and $y = 1$
  - Class B: $x_1 = -1$, $x_2 = -1$ and $y = -1$

$$w_1 = w_2 = 0.5$$

$$C = 0$$

$$\Downarrow$$

$$f(X) = 0.5x_1 + 0.5x_2 \begin{cases} > 0 & (\text{Class } A) \\ < 0 & (\text{Class } B) \end{cases}$$



Support Vector Machine with Noise

- **In practice, training samples may contain noise or are not linearly separable**

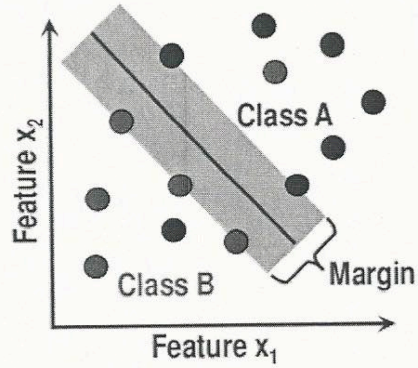
$$\min_{W,C} \; W^T W$$
$$\text{S.T.} \quad y_i \left(W^T X_i + C \right) \geq 1$$
$$\forall i = 1, 2, \ldots, N$$

(No feasible solution)

Parameter determined by cross validation

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

Error of i-th training sample



Support Vector Machine with Noise

- **Can be solved by convex programming**
  - Cost : sum of two convex functions
  - Constraints: linear and hence convex

Linear (convex) Quadratic (convex)

$$\min_{W,C,\xi} \; \sum \xi_i + \lambda W^T W \longrightarrow \text{Convex}$$
$$\left.\begin{aligned} &\text{S.T.} \quad y_i \left(W^T X_i + C \right) \geq 1 - \xi_i \\ &\xi_i \geq 0 \\ &\forall i = 1, 2, \ldots, N \end{aligned}\right\} \longrightarrow \text{Linear}$$

(Convex optimization)

Regularization

- Regression vs. classification

$$\min_{x} \|A x - B\|_2^2 + \lambda \|x\|_2^2$$

Regression

Regularization

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

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

$$\xi_i \ge 0$$

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

Support vector machine

Other regularization forms can also be used for support vector machine

Regularization

- L_1-norm regularization is used to find a sparse solution of W

L_1-norm regularization

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

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

$$\xi_i \ge 0$$

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

$\Rightarrow$

$$\min_{W,C,\xi} \sum \xi_i + \boxed{\lambda \|W\|_1}$$

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

$$\xi_i \ge 0$$

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

Important for feature selection

Regularization

- **Feature selection**

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

$$\Downarrow$$

$$\underbrace{\begin{bmatrix} \cdots & 0 & \cdots & 0 & \cdots \end{bmatrix}}_{W^T} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}}_{X} \qquad \begin{matrix} \\ \\ \rightarrow \\ \text{Important features} \\ \rightarrow \end{matrix}$$

Summary

- **Classification**
  - Support vector machine
  - Regularization

18-660: Numerical Methods for Engineering Design and Optimization

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

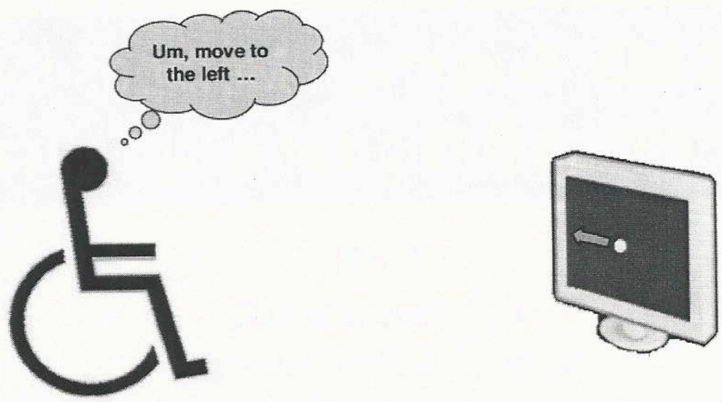
Electrical & Computer ENGINEERING

Outline

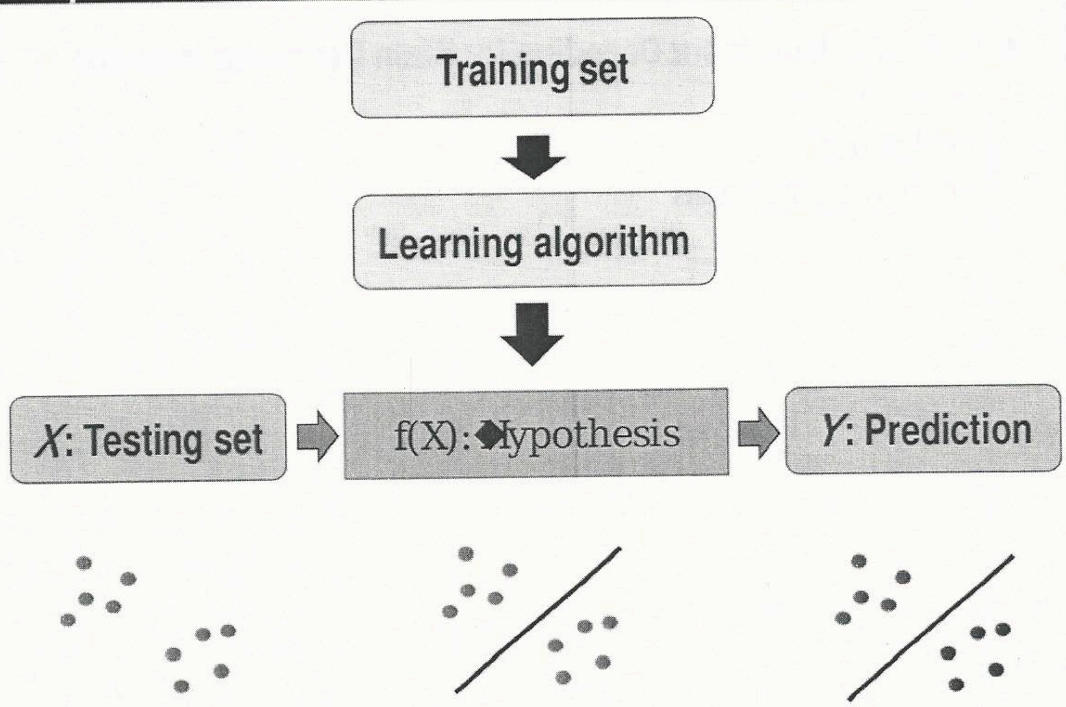
- **Project 3: Movement Decoding for Brain Computer Interface**
  - Background
  - Methods
  - Submission details

Brain Computer Interface (BCI)

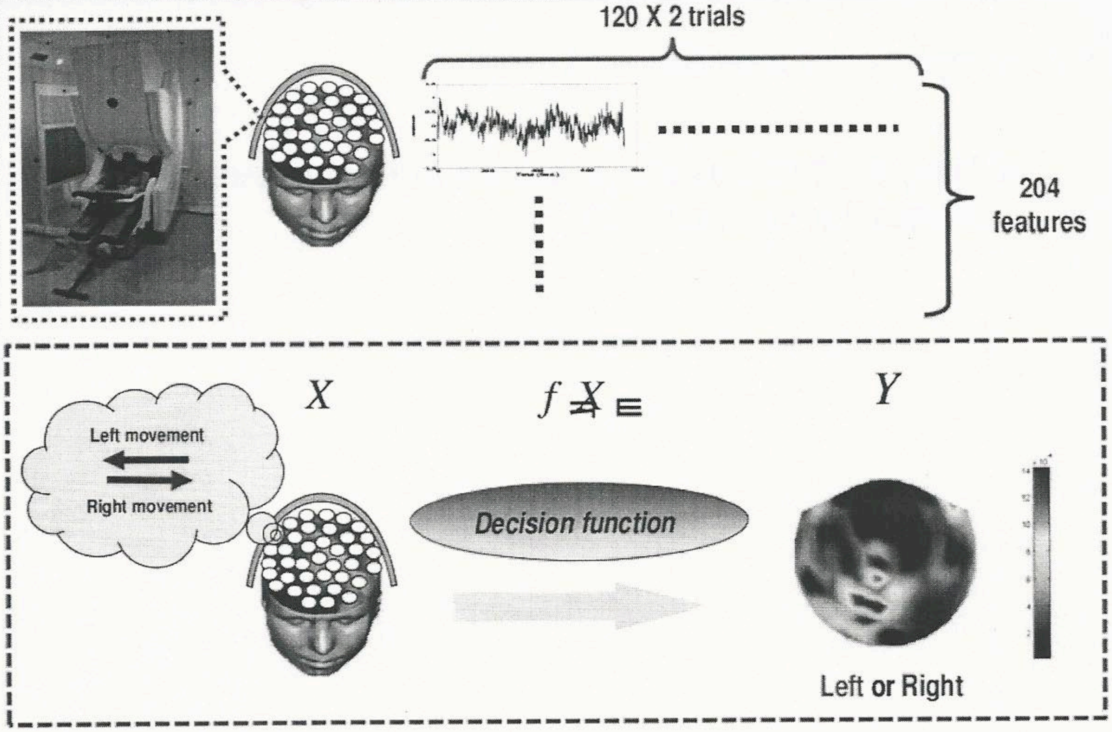
- BCI is a means of establishing direct communication pathway between brain and computers



Supervised Learning



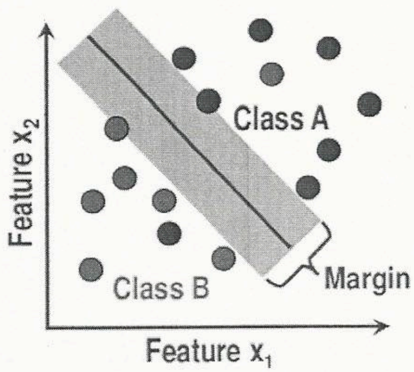
BCI Data Classification



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:

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

Features

Determine W and C with maximum margin

$$\min_{W,C,\xi} \; \sum \xi_i + \frac{1}{2} W^T W$$

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

$$\xi_i \ge 0$$

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

Objective

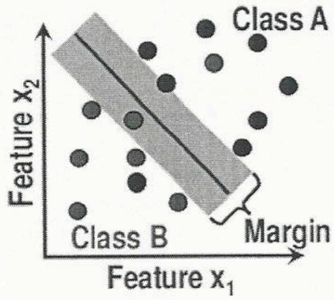
- Implement a two-class SVM classifier
- Apply the SVM classifier to decode BCI features into two classes (i.e. Left vs. Right)

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**



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

- **Logarithmic barrier**

$$\min_{W,C,\xi} \sum \xi_i + \frac{1}{2} W^T W - \frac{1}{t}\sum_{i=1}^{N} \log\left(W^T X_i y_i + C y_i + \xi_i - 1\right) - \frac{1}{t}\sum_{i=1}^{N} \log\left(\xi_i\right) \quad (**)$$

$t > 0$

Interior Point Method

- **Optimization problem**

$$\min_{W,C,\xi} \sum \xi_i + \frac{1}{2} W^T W - \frac{1}{t}\sum_{i=1}^{N} \log\left(W^T X_i y_i + C y_i + \xi_i - 1\right) - \frac{1}{t}\sum_{i=1}^{N} \log\left(\xi_i\right) \quad (**)$$

$t > 0$

- **Interior point algorithm**
  - Select an initial value of t and an initial guess $[W^{(0)}\ C^{(0)}\ \xi^{(0)}]$
  - Repeat:
    - 1. solve the unconstrained nonlinear optimization problem (**) to find the optimal solution $[W^*\ C^*\ \xi^*]$
    - 2. $[W^{(0)}\ C^{(0)}\ \xi^{(0)}] = [W^*\ C^*\ \xi^*]$ and $t = \mu t$ where $\mu = 15$
    - 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:

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

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

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

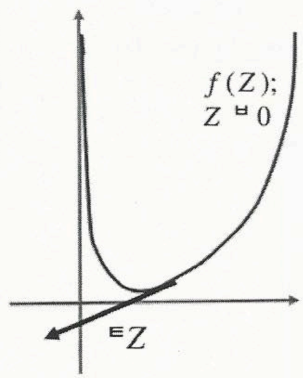
Solve Unconstrained Nonlinear Optimization

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

$Z = [W\ C\ \xi]$

- **Newton method**
  - Start from an initial value $Z^{(0)} = [W^{(0)}\ C^{(0)}\ \xi^{(0)}]$
  - Repeat:
    - 1. Compute the Newton step and decrement
      $\Delta Z = -\left[\nabla^2 f(Z)\right]^{-1} \nabla f(Z)$ $\quad \lambda^2 = \nabla f(Z)^T \left[\nabla^2 f(Z)\right]^{-1} \nabla f(Z)$ $= -\nabla f(Z)^T \Delta Z$
    - 2. Stopping criterion: quit if $\lambda^2/2 \le \varepsilon$ 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:** $Z + s \cdot \Delta Z \in dom\, f(Z)$

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

$$\Updownarrow$$

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

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

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

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

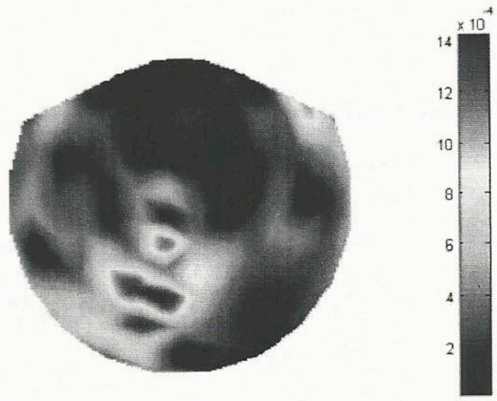
Summary of SVM Solver

- **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))



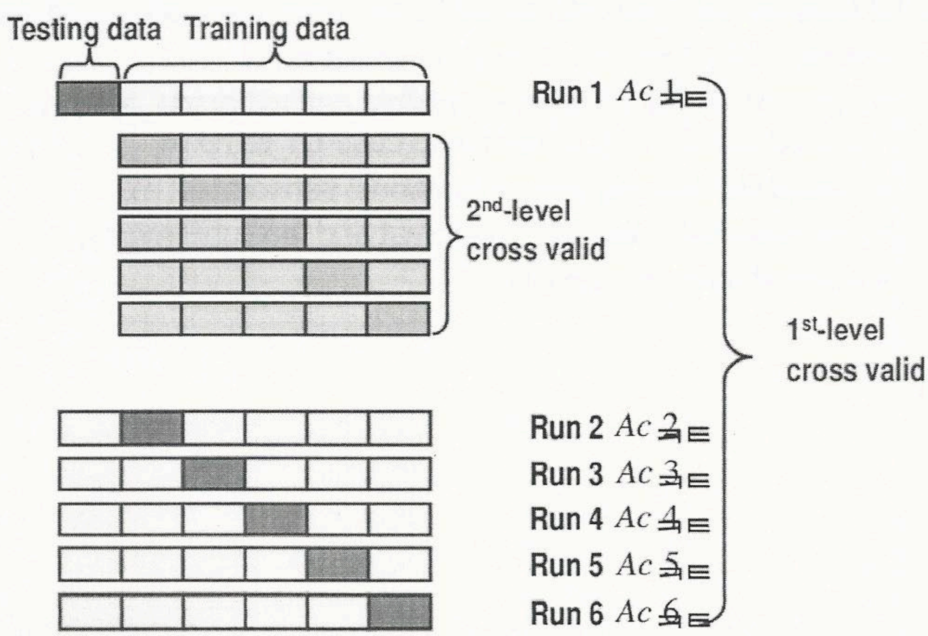
Cross Validation

- **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: $Ac(i),\ i = 1, 2, \ldots, 6$
- **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 ¬ 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 ¬ of SVM
    - Try at least ¬ ∈ {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