Multi-class Support Vector Machines

MACHINE LEARNING

MIM, HUS, VNU HANOI 2023-2024

CAO VĂN CHUNG

Slide Credit. Viktoria Muravina; J. Weston; C. Watkins

Introduction

Solution to binary classification problem using Support Vectors (SV) is well developed. Now we consider the model for general *K classes.*

Multi-Class pattern recognition (number of classes $K > 2$) are usually solved using a voting scheme methods based on combining many binary classification functions

The paper proposes two methods to solve k-class problems is one step

1. Direct generalization of binary SV method
2. Solving a linear program instead of a quadratic one

What is *K*-class Pattern Recognition Problem

We need to construct a decision function given N independent identically distributed samples of an unknown function

$$(x_1, y_1), \dots, (x_N, y_N)$$

where $x_n, n = 1, \dots, N$ is a vector of length d and $y_n \in \{1, \dots, K\}$ represents class of the sample (in *d*-dimentional space $\mathbf{R}^d$)

- Decision function $f(x, \lambda)$ which classifies a point x, is chosen from a set of functions defined by parameter λ
- It is assumed that the set of functions is chosen before hand

The goal is to choose the parameterλ that minimizes $\sum_{n=1}^{N} Pen\ (y_n, f(x_n, \lambda))$ where

$$Pen\big(y, f(x, \lambda)\big) = \begin{cases} 0 & \text{if } f(x, \lambda) = y \\ 1 & \text{otherwise} \end{cases}$$

Binary Classification SVM

Support Vector approach is well developed for the binary (*K*=2) pattern recognition

Main idea is to separate 2 classes (labelled $y \in \{-1, 1\}$) so that the margin is maximal

This gives the following optimization problem: minimize

$$\phi(w, \xi) = \frac{1}{2} < w \cdot w > + C \sum_{n=1}^{N} \xi_n$$

- with constrains $y_n(< w \cdot x_n > + b) \geq 1 - \xi_n, \text{ and } \xi_n \geq 0, \text{ for } n = 1, \ldots, N$

Binary SVM continued

Solution to this problem is to maximize the quadratic form:

$$g(\lambda) = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n,m=1}^{N} y_n y_m \lambda_n \lambda_m K(x_n, x_m)$$

- with constrains $0 \leq \lambda_n \leq C,\ n = 1, \ldots, N$ and $\sum_{n=1}^{N} \lambda_n y_n = 0$

Giving the following decision function

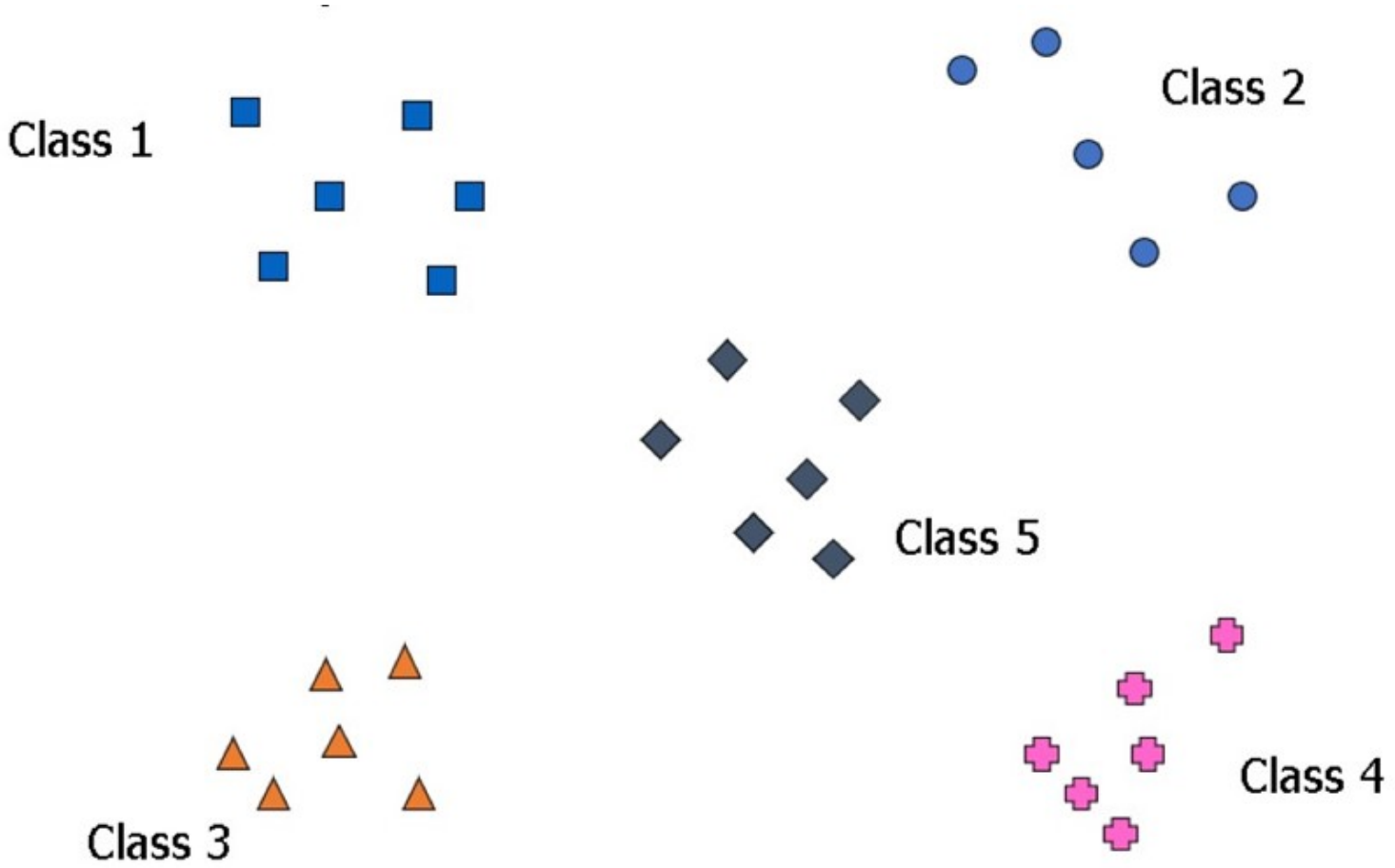
$$y = f(x) = \text{sign}\left[\left(\sum_{n=1}^{N} \lambda_n y_n K(x, x_n)\right) + b\right]$$

Multi-Class Classification Using Binary SVM

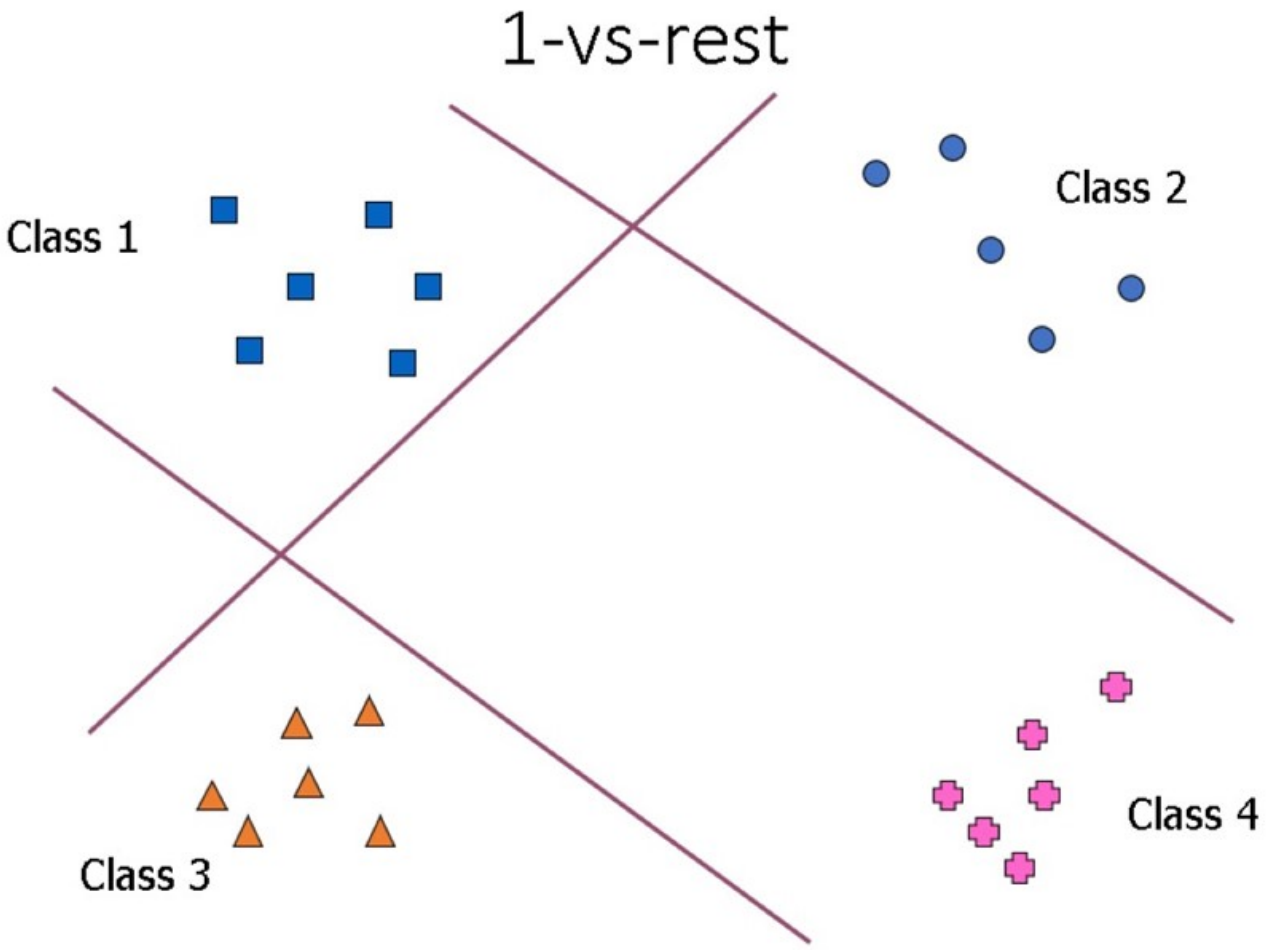
There 2 main approaches to solving multi-class pattern recognition problem using binary SVM

1. Consider the problem as a collection of binary classification problems (1-against-all)
   - k classifiers are constructed one for each class
   - n^{th} classifier constructs a hyperplane between class n and the k-1 other classes
   - Use a voting scheme to classify a new point
2. Or we can construct $\frac{K(K-1)}{2}$ hyperplanes (1-against-1)
   - Each separating each class from each other
   - Applying some voting scheme for classification

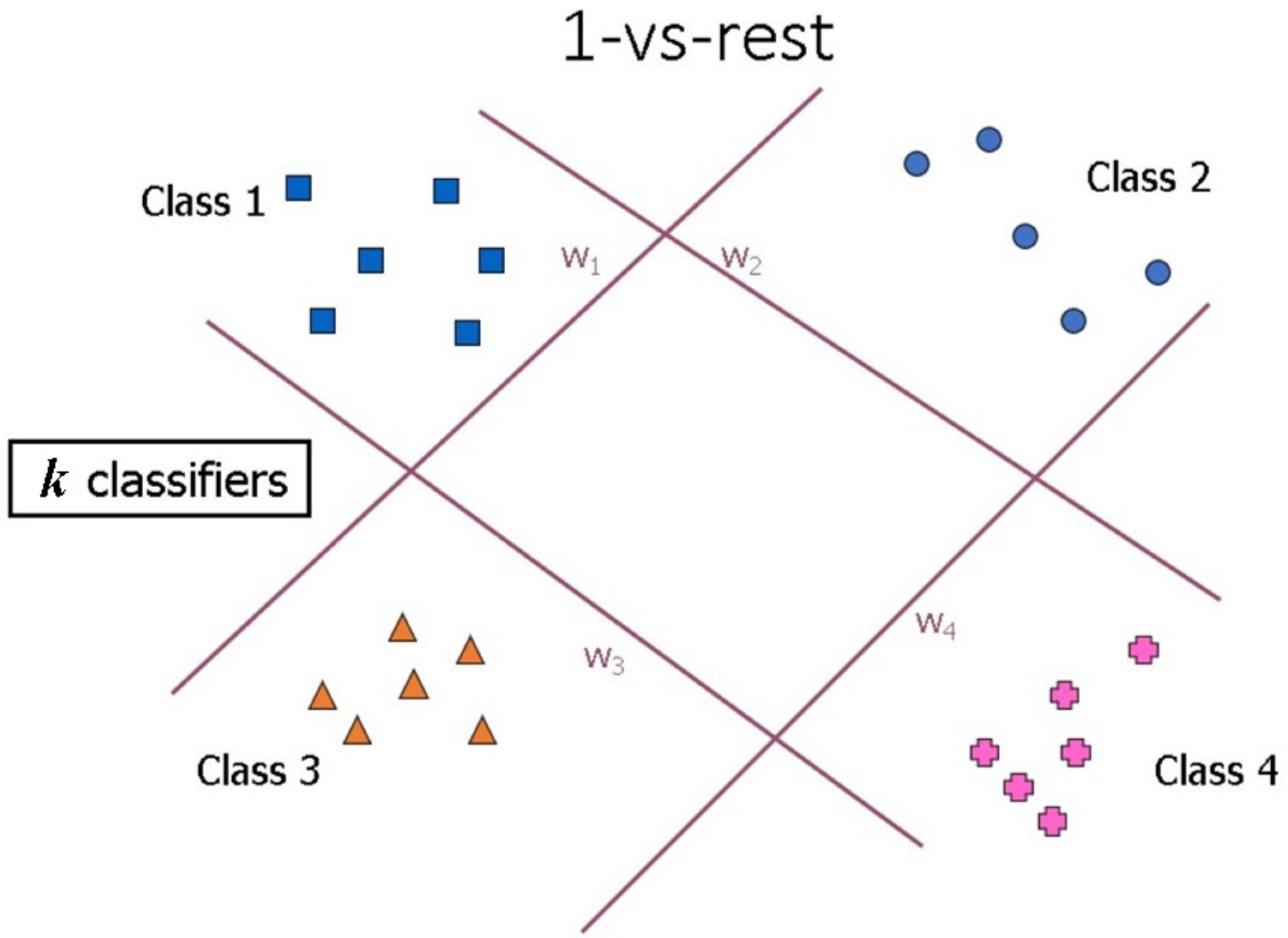
Multi-Class Classification Using Binary SVM



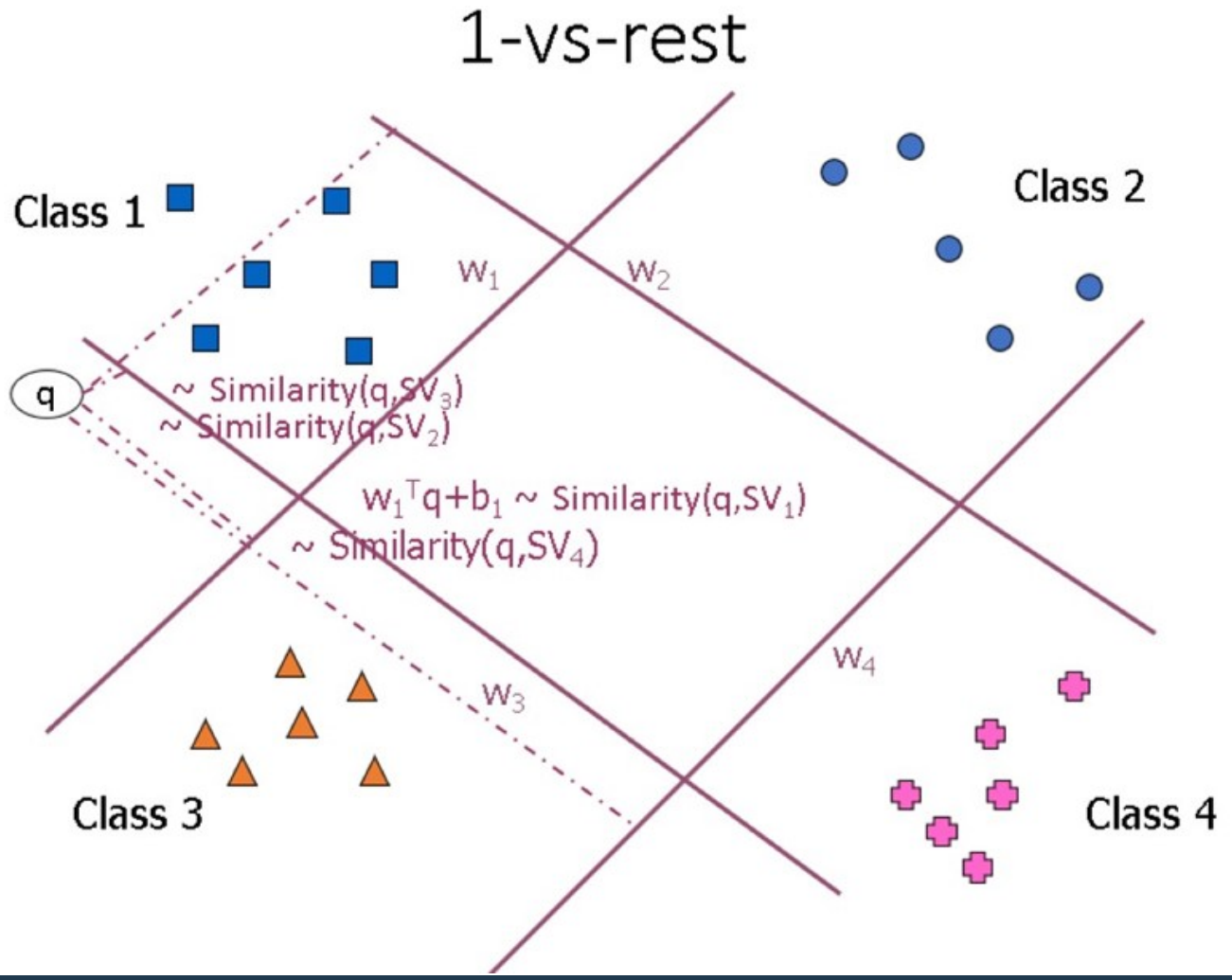
Multi-Class Classification Using Binary SVM



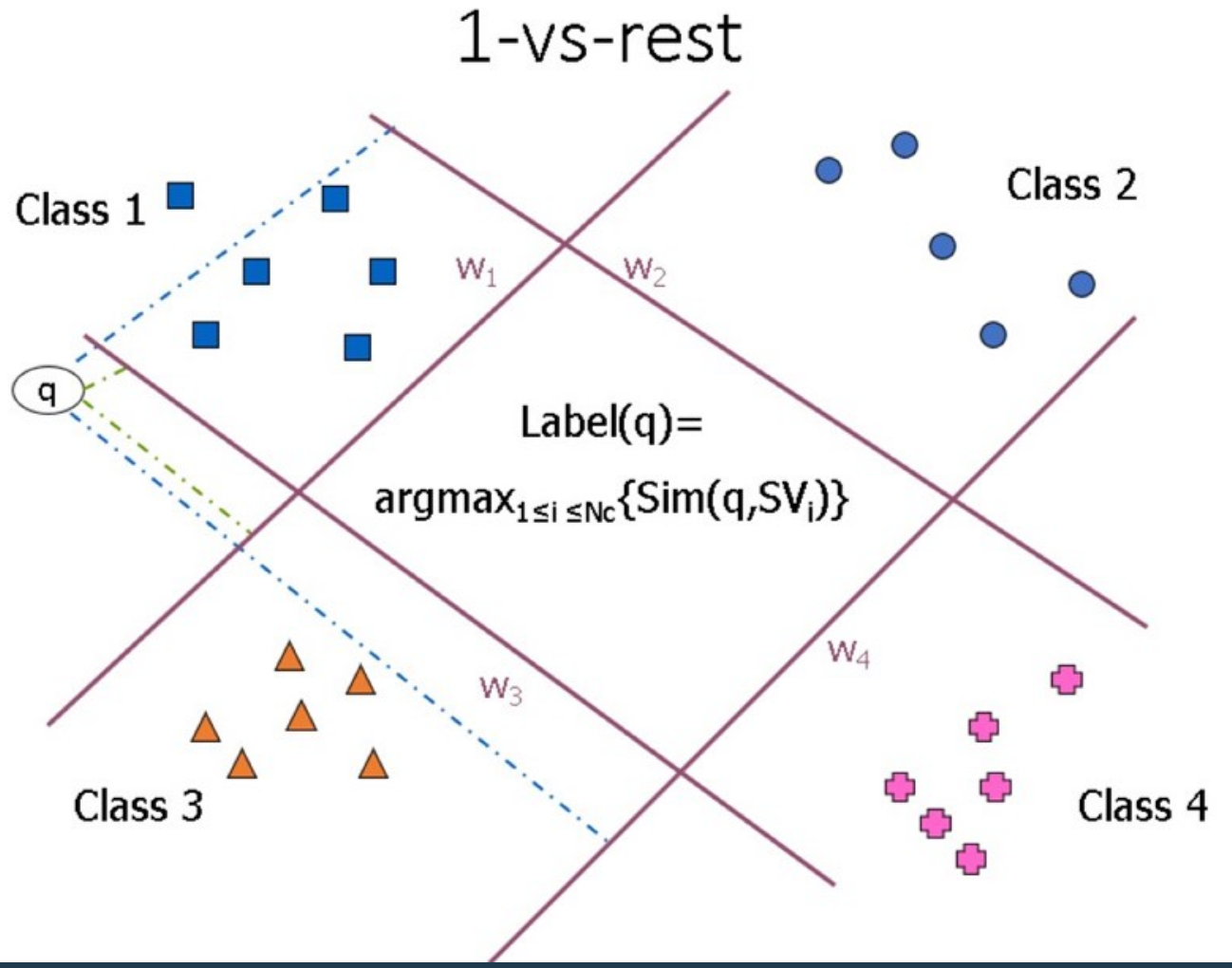
Multi-Class Classification Using Binary SVM



Multi-Class Classification Using Binary SVM



Multi-Class Classification Using Binary SVM



Multi-Class Classification Using Binary SVM

1-vs-rest Complexity

- For training:
  - N_c classifiers, each using n vectors for finding hyperplane

$$O(k \cdot n^2)$$

- For classifying new objects:
  - N_c classifiers, each is tested once, M=max number of SV

$$O(M \cdot k)$$

k-class Support Vector Machines

More natural way to solve k-class problem is to construct a decision function by considering all classes at once

One can generalize the binary optimization problem on slide 5 to the following: minimize

$$\phi(w,\xi) = \frac{1}{2}\sum_{m=1}^{k} < w_m \cdot w_m > + C\sum_{n=1}^{N}\sum_{l\neq y_n} \xi_n^l$$

- with constraints

$$< w_{y_n} \cdot x_n > + b_{y_n} \geq < w_l \cdot x_n > + b_l + 2 - \xi_n^l$$
$$\xi_n^l \geq 0 \qquad n = 1, \dots, N \qquad l \in \{1, \dots, k\} \backslash \mathrm{y}_n$$

This gives the decision function:

$$f(x) = \arg\max_l [< w_l \cdot x > + b_l], \qquad l = 1, \dots, K$$

k-class Support Vector Machines (continued)

Solution to this optimization problem in 2 variables is the saddle point of the Lagrangian:

$$L(w, b, \xi, \lambda, \beta) = \frac{1}{2}\sum_{m=1}^{k} < w_m \cdot w_m > + C\sum_{n=1}^{N}\sum_{m=1}^{K} \xi_n^m$$

$$-\sum_{n=1}^{N}\sum_{m=1}^{k} \lambda_n^m [< (w_{y_n} - w_m) \cdot x_n > + b_{y_n} - b_m - 2 + \xi_n^m] - \sum_{n=1}^{N}\sum_{m=1}^{k} \beta_n^m \xi_n^m$$

- with the dummy variables $\lambda_n^{y_n} = 0,\ \xi_n^{y_n} = 2, \beta_n^{y_n} = 0, \quad n = 1, \dots, N$
- and constraints $\lambda_n^m \geq 0,\ \beta_n^m \geq 0,\ \xi_n^m \geq 0, \quad n = 1, \dots, N \quad m \in \{1, \dots, k\} \backslash \mathrm{y}_n$

which has to be maximized with respect to α and β and minimized with respect to w and ξ

k-class Support Vector Machines (continued)

After taking partial derivatives and simplifying we end up with

$$L(\lambda) = 2\sum_{n,m} \lambda_n^m + \sum_{n,j,m} \left[-\frac{1}{2} c_n^{y_n} A_n A_j + \lambda_n^m \lambda_j^{y_n} - \frac{1}{2} \lambda_n^m \lambda_j^m \right] < x_n \cdot x_j >$$

- where $c_n^m = \begin{cases} 1 & \text{if } y_n = m \\ 0 & \text{if } y_n \neq m \end{cases}$ and $A_n = \sum_{m=1}^{k} \lambda_n^m$

Which is a quadratic function in terms of α with linear constraints

$$\sum_{n=1}^{N} \lambda_n^m = \sum_{n=1}^{N} c_n^m A_n \,, m = 1, \ldots, K$$

and $0 \leq \lambda_n^m \leq C,\ \lambda_n^{y_n} = 0\ \ n = 1, \ldots, \mathrm{N};\ m \in \{0, \ldots, k\} \backslash \mathrm{y}_n$

Please read slides 19 to 22 for complete derivation of $L(\alpha)$

K-class Support Vector Machines (continued)

This gives the decision function

$$f(x,\lambda) = \arg\max_{m}[\sum_{n=1}^{N}(c_n^m A_n - \lambda_n^m) < x_n \cdot x > + b_m]$$

The inner product $< x_i, x_j >$ can be replaced with the kernel function

$$K(x_i, x_j)$$

When our k=2 the resulting hyperplane is identical to the one that we get using binary SVM

k-Class Linear Programing Machine

Instead of considering the decision function as separating hyperplane we can view each class having its own decision function

- Defined only by training points belonging to the class

This gives us that the decision rule is the largest decision function at point x

k-Class Linear Programing Machine

For this method minimize the following linear program

$$\sum_{n=1}^{N} \lambda_n + C \sum_{n=1}^{N} \sum_{j \neq y_n} \xi_n^j$$

- Subject to the following constraints

$$\sum_{m:y_m=y_n} \lambda_n K(x_n, x_m) + b_{y_n} \geq \sum_{n:y_n=y_j} \lambda_n K\left(x_n, x_j\right) + b_{y_j} + 2 - \xi_n^j$$

$$\lambda_n \geq 0, \qquad \xi_n^j \geq 0, \qquad n = 1, \dots, N, \qquad j \in \{1, \dots, k\} \backslash y_i$$

and use the decision rule

$$f(x) = \arg\max_n \left(\sum_{i:y_i=n} \lambda_n K(x, x_n) + b_n \right)$$

Further Analysis

For binary SVM expected probability of an error in the test set is bounded by the ratio of expected number of support vectors to the number of vectors in the training set

$$E[P(\text{error})] = \frac{E[\text{number of support vectors}]}{(\text{number of training vectors}) - 1}$$

This bound also holds in the multi-class case for the voting scheme methods

It is important to note while 1-against-all method is a feasible solution to multi-class SVM methods, it is not necessarily the optimal one

Limitations and Conclusions

Optimization problem that we need to solve is very large

- For the quadratic method our optimization function is quadratic is $(k-1)*l$ variables
- Linear programing method optimization is linear with $k*l$ variable and $k*l$ constraints
- This could lead to slower training times then in 1-against-all especially in the case of the quadratic method

The new methods do not out perform the 1-agains-all and 1-against-1 methods, however both methods reduce number of support vectors needed for the decision function

Further research is need to test how the methods perform for large datasets

Derivation of $L(\lambda)$ on slide 15

We need to find a saddle point so we start with the Lagrangian

$$L(w, b, \xi, \lambda, \beta) = \frac{1}{2}\sum_{l=1}^{k} < w_l \cdot w_l > + C \sum_{n=1}^{N}\sum_{l=1}^{k} \xi_n^l$$

$$-\sum_{n=1}^{N}\sum_{l=1}^{k} \lambda_n^l [< \left(w_{y_n} - w_l\right) \cdot x_n > + b_{y_n} - b_l - 2 + \xi_n^l] - \sum_{n=1}^{N}\sum_{l=1}^{k} \beta_n^l \xi_n^l$$

Using the notation

$$c_n^l = \begin{cases} 1 \text{ if } y_n = l \\ 0 \text{ if } y_n \neq l \end{cases} \text{ and } A_n = \sum_{l=1}^{k} \lambda_n^l$$

Take partial derivatives with respect to w_l, b_l and ξ_n^l and set them equal to 0

Derivation of $L(\lambda)$ on slide 15 cont.

The derivatives are (*l = 1, 2, …, K*)

$$\frac{\partial L}{\partial w_l} = w_l + \sum_{n=1}^{N} \lambda_n^l x_n - \sum_{n=1}^{N} A_n c_n^l x_n = 0$$

$$\frac{\partial L}{\partial b_l} = -\sum_{n=1}^{N} A_n c_n^l + \sum_{n=1}^{N} \alpha_n^l = 0$$

$$\frac{\partial L}{\partial \xi_n^l} = -\lambda_n^l + C - \beta_n^l = 0$$

Derivation of $L(\lambda)$ on slide 15 cont.

From this we get

$$w_l = \sum_{n=1}^{N} \left(c_n^l A_n - \lambda_n^l\right) x_n\,, \qquad \sum_{n=1}^{N} \lambda_n^l = \sum_{n=1}^{N} A_n c_n^l\,, \qquad \beta_n^l + \lambda_n^l = C$$

$$\text{and } 0 \le \lambda_n^l \le C$$

After substituting the equations that we got on previous slides into the Lagrangian we get

$$L(\lambda) = 2\sum_{n,l} \lambda_n^l +$$

$$+ \sum_{n,m,l} \left(\frac{1}{2} c_n^l c_m^l A_n A_m - \frac{1}{2} c_n^l A_n \lambda_m^l - \frac{1}{2} c_m^l A_m \lambda_n^l + \frac{1}{2} \lambda_n^l \lambda_m^l - c_m^{y_n} A_m \lambda_n^l + \right.$$

$$\left. \lambda_n^l \lambda_m^{y_n} + c_m^l A_m \lambda_n^l - \lambda_n^l \lambda_m^l \right) < x_n \cdot x_m >$$

Derivation of $L(\lambda)$ on slide 15 cont.

Due to the fact that $\sum_l c_n^l A_n \lambda_m^l = \sum_l c_m^l A_m \lambda_n^l$ we get

$$L(\lambda) = 2\sum_{n,l} \lambda_n^l + \sum_{n,m,l} \frac{1}{2} c_n^l c_m^l A_n A_m - c_m^{y_n} A_n A_m + \lambda_n^l \lambda_m^{y_n} - \frac{1}{2}\lambda_n^l \lambda_m^l) < x_n \cdot x_m >$$

Also $\sum_l c_n^l c_m^l = c_n^{y_n} = c_n^{y_n}$ thus

$$L(\lambda) = 2\sum_{n,l} \lambda_n^l + \sum_{n,m,l} \left[-\frac{1}{2} c_n^{y_n} A_n A_m + \lambda_n^l \lambda_m^{y_n} - \frac{1}{2}\lambda_n^l \lambda_m^l \right] < x_n \cdot x_m >$$