Support Vector Machines for Data Classification and Regression

Chih-Jen Lin

Department of Computer Science
National Taiwan University

Talk at Merck Research Lab., August 16, 2002

Outline

- Support vector classification
- Example: engine misfire detection
- Support vector regression
- Example: electricity load forecasting
- Discussion and conclusions

Data Classification

- Given training data in different classes (labels known)
  Predict test data (labels unknown)
- Examples
  - Handwritten digits recognition
  - Spam filtering
  - Text classification
  - Prediction of signal peptide in human secretory proteins
- Training and testing
- Methods:
  - Nearest Neighbor

  - Neural Networks
  - Decision Tree
- Support vector machines: a new method
- Becoming more and more popular
- Why do we need a new method ?
- A good classification method:
  - Avoid underfitting: small training error
  - Avoid overfitting: small testing error

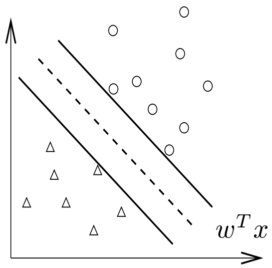
Support Vector Classification

- Training vectors : $x_i, i = 1, \dots, l$
- Consider a simple case with two classes:

  Define a vector y

$$y_i = \begin{cases} 1 & \text{if } x_i \text{ in class } 1 \\ -1 & \text{if } x_i \text{ in class } 2, \end{cases}$$

- A hyperplane which separates all data



$w^T x + b = +1, 0, -1$

- A separating hyperplane: $w^T x + b = 0$

$$\begin{aligned} (w^T x_i) + b > 0 & \quad \text{if } y_i = 1 \\ (w^T x_i) + b < 0 & \quad \text{if } y_i = -1 \end{aligned}$$

- Decision function $f(x) = \text{sign}(w^T x + b)$, x: test data

  Variables: w and b : Need to know coefficients of a plane

  Many possible choices of w and b

- Select w, b with the maximal margin.

  Maximal distance between $w^T x + b = \pm 1$

  Vapnik's statistical learning theory.

$$\begin{aligned} (w^T x_i) + b \geq 1 & \quad \text{if } y_i = 1 \\ (w^T x_i) + b \leq -1 & \quad \text{if } y_i = -1 \end{aligned} \tag{1}$$

- Distance between $w^T x + b = 1$ and -1:

$$2/\|w\| = 2/\sqrt{w^T w}$$

- $\max 2/\|w\| \equiv \min w^T w/2$

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} w^T w \\ & y_i((w^T x_i) + b) \geq 1, \qquad \text{from (1)} \\ & i = 1, \dots, l. \end{aligned}$$

Higher Dimensional Feature Spaces

- Earlier we tried to find a linear separating hyperplane

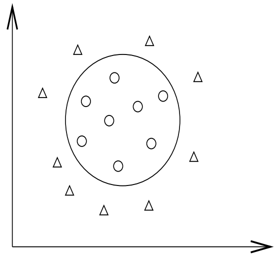
  Data may not be linear separable

- Non-separable case: allow training errors

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ & y_i((w^T x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \ i = 1, \dots, l \end{aligned}$$

- $\xi_i > 1$, x_i not on the correct side of the separating plane
- C: large penalty parameter, most ξ_i are zero

- Nonlinear case: linear separable in other spaces ?



- Higher dimensional (maybe infinite) feature space

$$\phi(x) = (\phi_1(x), \phi_2(x), \ldots).$$

- Example: $x \in R^3, \phi(x) \in R^{10}$

$$\begin{aligned}\phi(x) \quad = \quad & (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, \\ & x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)\end{aligned}$$

- A standard problem [Cortes and Vapnik, 1995]:

$$\begin{aligned}\min_{w,b,\xi} \quad & \frac{1}{2}w^Tw + C\sum_{i=1}^{l}\xi_i \\ & y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \ i = 1, \ldots, l\end{aligned}$$

- Can be solved using constrained optimization; But...

Finding the Decision Function

- w: a vector in a high dimensional space $\Rightarrow$ maybe infinite variables
- The dual problem

$$\begin{aligned}\min_{\alpha} \quad & \frac{1}{2}\alpha^TQ\alpha - e^T\alpha \\ & 0 \leq \alpha_i \leq C, i = 1, \ldots, l \\ & y^T\alpha = 0,\end{aligned}$$

where $Q_{ij} = y_iy_j\phi(x_i)^T\phi(x_j)$ and $e = [1, \ldots, 1]^T$

$$w = \sum_{i=1}^{l}\alpha_iy_i\phi(x_i)$$

- Primal and dual : optimization theory. Not trivial.

  Infinite dimensional programming.
- A finite problem:

  #variables = #training data
- $Q_{ij} = y_iy_j\phi(x_i)^T\phi(x_j)$ needs a closed form

  Efficient calculation of high dimensional inner products

  Kernel trick, $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$
- Example: $x_i \in R^3, \phi(x_i) \in R^{10}$

$$\begin{aligned}\phi(x_i) \quad = \quad & (1, \sqrt{2}(x_i)_1, \sqrt{2}(x_i)_2, \sqrt{2}(x_i)_3, (x_i)_1^2, \\ & (x_i)_2^2, (x_i)_3^2, \sqrt{2}(x_i)_1(x_i)_2, \sqrt{2}(x_i)_1(x_i)_3, \sqrt{2}(x_i)_2(x_i)_3),\end{aligned}$$

Then $\phi(x_i)^T\phi(x_j) = (1 + x_i^Tx_j)^2$.

- Popular methods: $K(x_i, x_j) =$

$$e^{-\gamma\|x_i - x_j\|^2}, \text{ (Radial Basis Function)}$$
$$(x_i^T x_j/a + b)^d \text{ (Polynomial kernel)}$$
$$\tanh(a x_i^T x_j + b)$$

- Decision function:

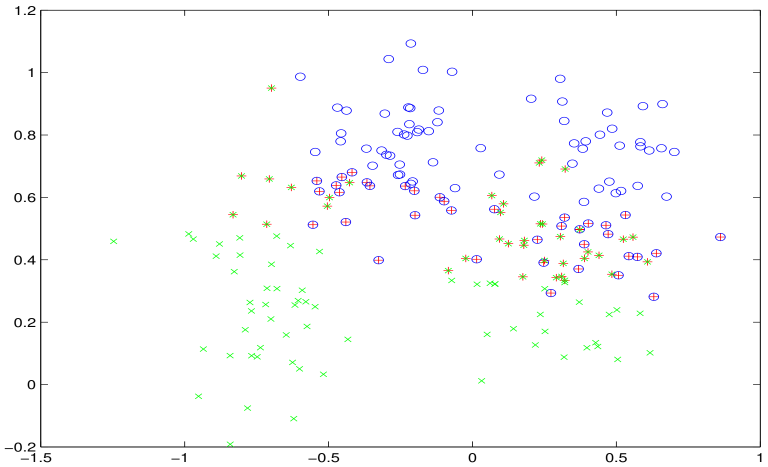
$$\begin{aligned} & w^T\phi(x) + b \\ = & \sum_{i=1}^{l} \alpha_i y_i \phi(x_i)^T\phi(x) + b \end{aligned}$$

No need to have w

- > 0: 1st class, < 0: 2nd class
- Only $\phi(x_i)$ of $\alpha_i > 0$ used

$\alpha_i > 0 \Rightarrow$ support vectors

Support Vectors: More Important Data

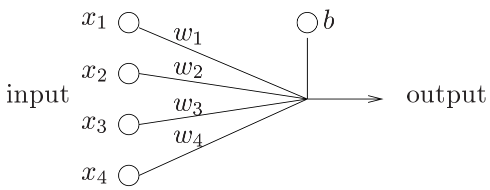


Neural Networks & Support Vector Machines

- Neural Networks:

  Start from a linear separating hyperplane as well

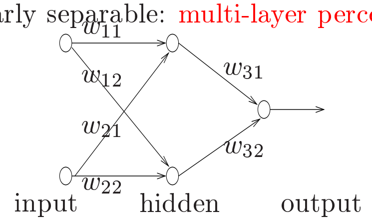
  Single-layer perceptron



  Decision function

$$\text{sgn}(w^T x + b)$$

- Data not linearly separable: multi-layer perceptron



- Minimize training error

  Subject to connections between levels i and $i-1$
- Using more complicated structures if not linearly separable
- NN: minimize training error: non-convex problem
- SVM: $\min \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i$

  Search for balance between training and testing errors

  A convex optimization problem

Non-convex if parameters (e.g. mapping functions) considered

Issues

- Why is SVM good ? Statistical learning theory
  - A learning machine:

$$x \to f(x,\alpha), f(x,\alpha) = 1 \text{ or } -1.$$

    Different α: different machines
  - The expected test error (generalized error)

$$R(\alpha) = \int \frac{1}{2}|y - f(x,\alpha)|dP(x,y)$$

    y: class of x (i.e. 1 or -1)
  - $P(x,y)$ unknown, empirical risk (training error):

$$R_{emp}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(x_i,\alpha)|$$

  - SVM bound, an example:
    Given $x_1, \ldots, x_l$

$$\mathcal{F} = \{x \to w^T x \mid \|w\| \le 1, \|x\| \le R\}$$

    With probability at least $1-\eta$, if $f \in \mathcal{F}$ has margin at least γ on all x_i:

$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\frac{c}{l}(\frac{R^2}{\gamma^2}\log^2 l + \log\frac{1}{\eta})}$$

  - γ^2: as large as possible
  - Remember SVM minimizes:

$$\frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i$$

- Solving large quadratic problems:

  - $\min_\alpha \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$, subject to $y^T\alpha = 0, 0 \le \alpha_i \le C$
  - $Q_{ij} = y_i y_j \phi(x_i)^T\phi(x_j) \neq 0$, Q : an l by l fully dense matrix
  - 30,000 training points: 30,000 variables:
    $(30,000^2 \times 8/2)$ bytes = 3GB RAM: still difficult
- Multiple-class classification
  Several two-class problems or combined together
- Automatic model selection
  - select the best parameters (kernel type, C, etc.)
- Comparisons with other methods
  - Result from [Hsu and Lin, 2001]:

Problem	#training	#testing	#class	statlog	SVM
dna	2000	1186	3	95.9%	95.447%
satimage	4435	2000	6	90.6%	91.3%
letter	15000	5000	26	93.6%	97.98%
shuttle	43500	14500	7	99.99%	99.924%

– Statlog: best rates by different methods
– SVM is competitive

Current Status

- Most basic things have been done
- Is its accuracy really better ? No
- No method performs the best on all data
- In my opinion, after careful data pre-processing
  Appropriately use NN or SVM ⇒ similar accuracy
- But, users may not use them properly
- The chance of SVM
  Easier for users to appropriately use it
  The ambition: replacing NN on some applications
  A clean, elegant, and easy-to-use software is essential

Example: Engine Misfire Detection

- First problem of IJCNN Challenge 2001, data from Ford
- Given time series length $T = 50,000$
- The kth data

$$x_1(k), x_2(k), x_3(k), x_4(k), x_5(k), y(k)$$

- $y(k) = \pm 1$: output, affected only by $x_1(k), \ldots, x_4(k)$
- $x_5(k) = 1$, kth data considered for evaluating accuracy
- 50,000 training data, 100,000 testing data (in two sets)
- Past and future information may affect $y(k)$

- $x_1(k)$: periodically nine 0s, one 1, nine 0s, one 1, and so on.
- Example:

```
0.000000 -0.999991 0.169769 0.000000 1.000000
0.000000 -0.659538 0.169769 0.000292 1.000000
0.000000 -0.660738 0.169128 -0.020372 1.000000
1.000000 -0.660307 0.169128 0.007305 1.000000
0.000000 -0.660159 0.169525 0.002519 1.000000
0.000000 -0.659091 0.169525 0.018198 1.000000
0.000000 -0.660532 0.169525 -0.024526 1.000000
0.000000 -0.659798 0.169525 0.012458 1.000000
```

- $x_4(k)$ more important

Background: Engine Misfire Detection

- Engine misfire: a substantial fraction of a cylinder's air-fuel mixture fails to ignite
- Frequent misfires: pollutants and costly replacement
- On-board detection:

  Engine crankshaft rational dynamics with a position sensor
- Training data: from some expensive experimental environment

Encoding Schemes

- For SVM: each data is a vector
- $x_1(k)$: periodically nine 0s, one 1, nine 0s, one 1, ...
  - 10 binary attributes

    $x_1(k-5), \ldots, x_1(k+4)$ for the kth data
  - $x_1(k)$: an integer in 1 to 10
  - Which one is better
  - We think 10 binaries better for SVM
- $x_4(k)$ more important

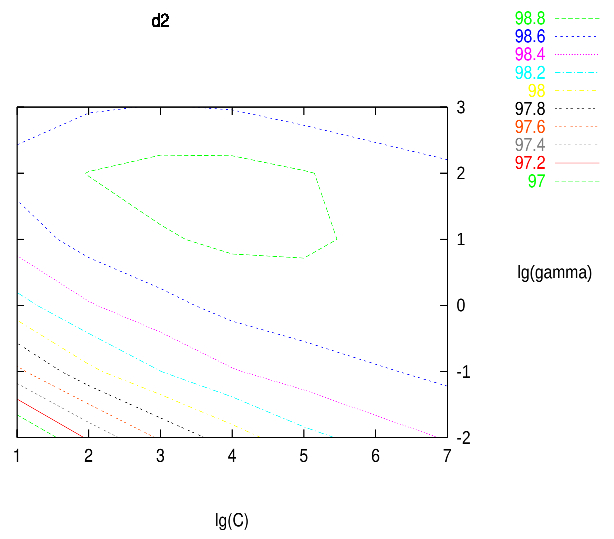
  Including $x_4(k-5), \ldots, x_4(k+4)$ for the kth data
- Each training data: 22 attributes

Training SVM

- Selecting parameters; generating a good model for prediction
- RBF kernel $K(x_i, x_j) = \phi(x_i)^T \phi(x_j) = e^{-\gamma \|x_i - x_j\|^2}$
- Two parameters: γ and C
- Five-fold cross validation on 50,000 data

  Data randomly separated to five groups.

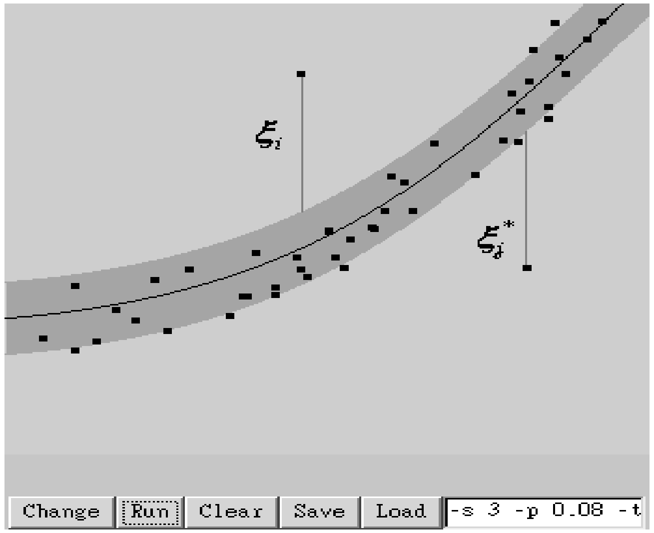
  Each time four as training and one as testing
- Use $C = 2^4, \gamma = 2^2$ and train 50,000 data for the final model



- Training SVM: One (C, γ), 50,000 data, 5-fold CV: $<$ 1 hour on PIII-500 with 384 MB RAM
- Test set 1: 656 errors, Test set 2: 637 errors
- About 3000 support vectors of 50,000 training data

  A good case for SVM
- This is just the outline. There are other details.
- It is essential to do model selection.
- We did this in three days! Others take at least one week

Support Vector Regression



- Primal

$$\min_{w,b,\xi,\xi^*} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i + C\sum_{i=1}^{l}\xi_i^*$$
$$-\epsilon - \xi_i^* \leq y_i - (w^T\phi(x_i) + b) \leq \epsilon + \xi_i$$
$$\xi_i, \xi_i^* \geq 0, i = 1, \ldots, l.$$

- Approximate function

$$f(x) = w^T\phi(x) + b$$

- ϵ-insensitive loss function

  A tube with width 2ϵ
- There is also a dual

Example: Electricity Load Forecasting

- EUNITE world wide competition 2001

  European Network of Excellence on Intelligent Technology for smart adaptive systems.
- Given
  - Load per half hour from 1997 to 1998
  - Average daily temperature from 1995 to 1998
  - List of holidays
- Goal:

  Predict daily maximal load of January 1999
- A time series prediction problem

  More complicated than the previous example

SVR for Time Series Prediction

- Given $(\cdots, y_{t-\Delta}, \cdots, y_{t-1}, y_t, \cdots, y_l)$ as training series
- Generate training data:

  $(y_{t-\Delta}, \cdots, y_{t-1})$ as attributes (features) of x_i

  y_t as the target value y_i
- One-step ahead prediction
- Prediction:

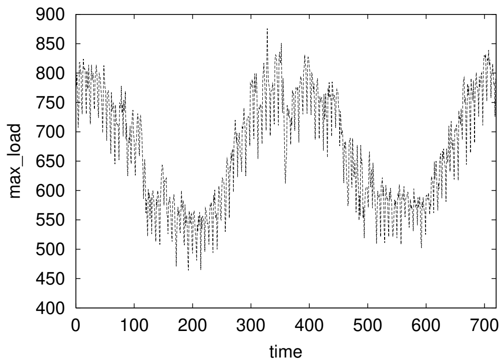
  Starting from the last segment

  $$(y_{l-\Delta+1}, \ldots, y_l) \rightarrow \hat{y}_{l+1}$$

  Repeat by using newly predicted values

Data Analyses

- Maximal load of each day
  - Do not know how to use all half-hour data
  - Not used for later analyses/experiments



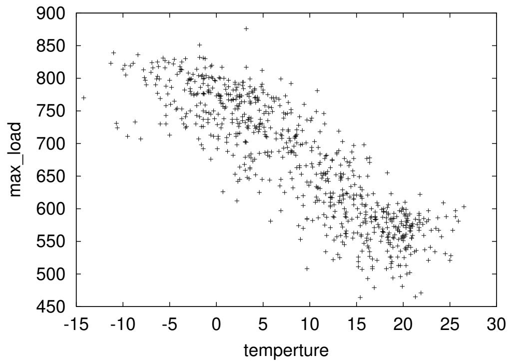
- Issues discussed earlier; Possibe attributes
  - Seasonal periodicity
  - Weekly periodicity

    Weekday: higher, Weekend: lower
  - Holiday effect
  - All above known for January 1999
  - Weather influence

    Temperature unknown for January 1999

- Temperature is very very important: the main difficulty
  - Most early work on short-term prediction

    Temperature available
  - Error propagation of long-term time series prediction



Data Encoding

- Attributes
  - Δ attributes: maximal loads of the past Δ days
  - Seven binary (i.e. 0 or 1) attributes:
    Target day in which day of a week
  - One binary attribute: target day holiday or not
  - One attribute (optional): temperature of the target day
- Temperature unknown: train two SVM

$$\begin{pmatrix} y_{t-\Delta} \\ T_{t-\Delta} \end{pmatrix}, \cdots, \begin{pmatrix} y_{t-1} \\ T_{t-1} \end{pmatrix} \xrightarrow{\text{SVM1}} y_t$$

$$T_{t-\Delta}, \cdots, T_{t-1} \xrightarrow{\text{SVM2}} T_t$$

Parameter Selection

- Parameters
  - Δ
  - C: cost of error
  - ϵ: width of the ϵ-insensitive tube
  - mapping function ϕ
- Known data separated to
  Training, validation, and testing
- January 1997 or January 1998 as testing
- Validation
  Each time one week as validation; rest as training

Average 100 mean square errors

Is this a robust way for time series prediction ?

- Model selection is expensive
  Restrict the search space: via reasonable choices or simply guessing
  - $\Delta = 7$
  - RBF function $\phi(x_i)^T\phi(x_j) = e^{-\gamma\|x_i - x_j\|^2}$
    Use default width $\epsilon = 0.5$ of LIBSVM
    C and γ: validation (average of 100 MSEs)
    $C = 2^{12}, \gamma = 2^{-4}$

Summer Data

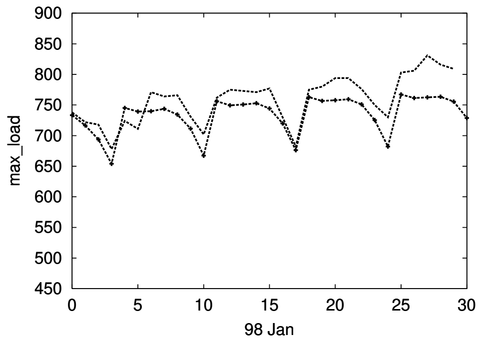
- Without summer:
  Result for testing January 1998 (or 1997) better
- Give up information from April to September
- Using unsupervised time series segmentation (via SVM)
  Summer is separated
  Summer and winter should be considered separately ?
- A future issue for electricity load forecasting

Results

- Two SVRs: very difficult to predict temperature
- One day temperature suddenly drops or increases

  ⇒ Erroneous after that day
- Conclude if temperature is used, the variation is higher
- Decide to give up using the temperature information

  Only one SVM used
- The load of each week is similar



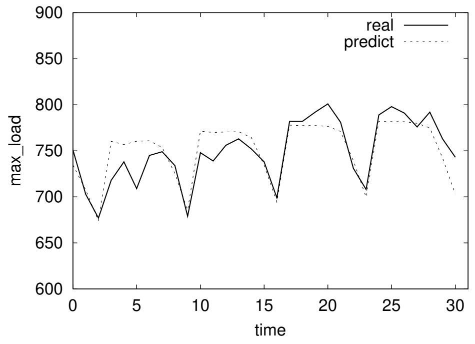
- However, the model manages to find the trend
- Holiday is lower but error larger

  Results after encountering a holiday *more* inaccurate

  Holidays: January 1 and 6
- Treat all 31 days in January 1999 as non-holidays

- Some earlier work consider holidays and non-holidays separately

  Now information about holidays is not enough.
- Overall we take a very conservative approach
- Forgot to manually lower load of January 6
- MAPE: 1.98839%
- MSE: 364.498



- Feature selection is the key for our approach

  Example: removing summer data, treating holidays as non-holidays
- Parameter selection: needed but a large range is ok

A Software: LIBSVM

- A library for SVM (in both C++ and Java)
  http://www.csie.ntu.edu.tw/~cjlin/libsvm
  - Different SVM formulations
  - Efficient multi-class classification
  - Tools for model selection
  - Weighted SVM for unbalanced data
  - GUI
- Interfaces:
  - Matlab: developed at Ohio State University
  - R (and S-Plus): developed at Technische Universität Wien
  - Python: developed at HP Labs.

Discussion and Conclusions

- NN: a huge and messy area
- Hopefully SVM will be in a better position
  Difficult to say right now
- All my research results can be found at
  http://www.csie.ntu.edu.tw/~cjlin