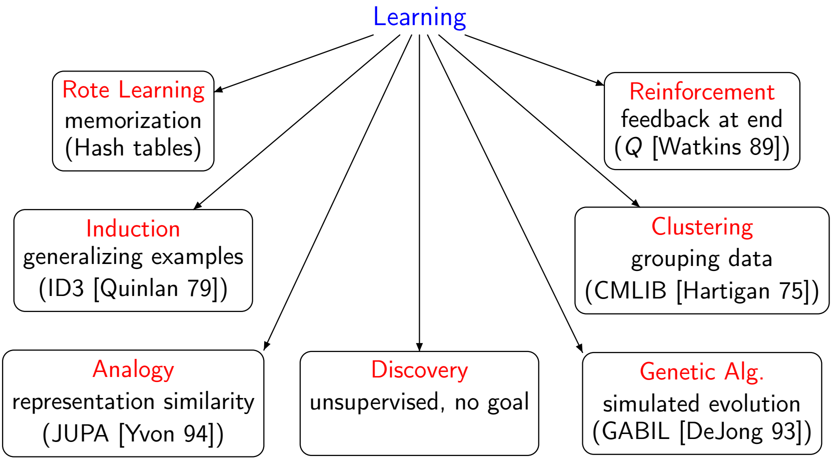
Introduction to Support Vector Machines

Andreas Maletti

Technische Universität Dresden
Fakultät Informatik

June 15, 2006

Learning by Machines

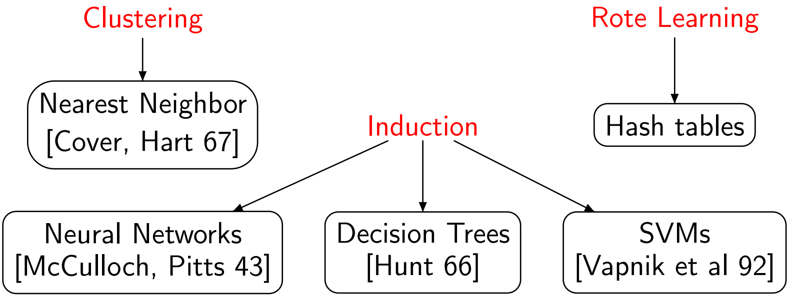


Supervised Learning

Definition

Supervised Learning: given nontrivial training data *(labels known)* predict test data *(labels unknown)*

Implementations



Problem Description—General

Problem

Classify a given input

- binary classification: two classes
- multi-class classification: several, but finitely many classes
- regression: infinitely many classes

Major Applications

- Handwriting recognition
- Cheminformatics (Quantitative Structure-Activity Relationship)
- Pattern recognition
- Spam detection (HP Labs, Palo Alto)

Problem Description—Specific

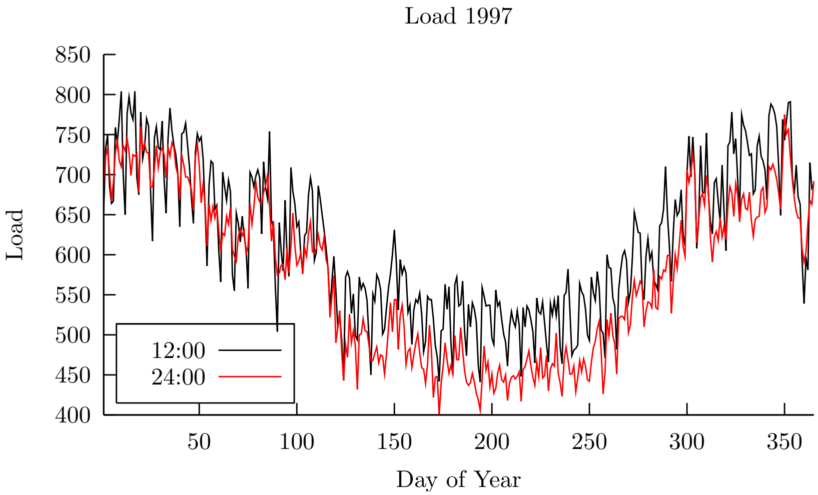
Electricity Load Prediction Challenge 2001

- Power plant that supports energy demand of a region
- Excess production expensive
- Load varies substantially
- Challenge won by libSVM [Chang, Lin 06]

Problem

- *given:* load and temperature for 730 days ($\approx$ 70kB data)
- *predict:* load for the next 365 days

Example Data



Problem Description—Formal

Definition (cf. [Lin 01])

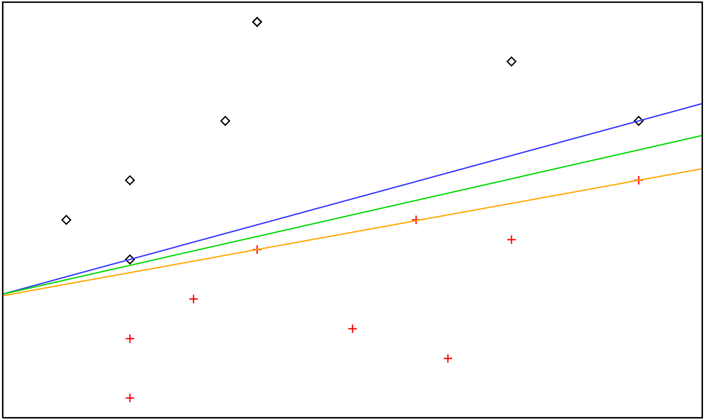
Given a training set $S \subseteq \mathbb{R}^n \times \{-1, 1\}$ of correctly classified input data vectors $\vec{x} \in \mathbb{R}^n$, where:

- every input data vector appears at most once in S
- there exist input data vectors $\vec{p}$ and $\vec{n}$ such that $(\vec{p}, 1) \in S$ as well as $(\vec{n}, -1) \in S$ (non-trivial)

successfully classify unseen input data vectors.

Linear Classification [Vapnik 63]

- *Given:* A training set $S \subseteq \mathbb{R}^n \times \{-1, 1\}$
- *Goal:* Find a hyperplane that separates $\mathbb{R}^n$ into halves that contain only elements of one class



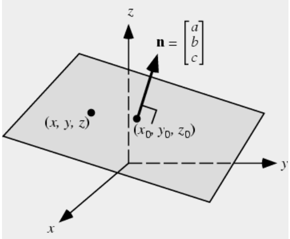
Representation of Hyperplane

Definition

Hyperplane $\vec{n} \cdot (\vec{x} - \vec{x}_0) = 0$

- $\vec{n} \in \mathbb{R}^n$ weight vector
- $\vec{x} \in \mathbb{R}^n$ input vector
- $\vec{x}_0 \in \mathbb{R}^n$ offset

Alternatively: $\vec{w} \cdot \vec{x} + b = 0$



Decision Function

- training set $S = \{(\vec{x}_i, y_i) \mid 1 \le i \le k\}$
- separating hyperplane $\vec{w} \cdot \vec{x} + b = 0$ for S

Decision: $\vec{w} \cdot \vec{x}_i + b \begin{cases} > 0 & \text{if } y_i = 1 \\ < 0 & \text{if } y_i = -1 \end{cases} \Rightarrow$ $f(\vec{x}) = \mathrm{sgn}(\vec{w} \cdot \vec{x} + b)$

Learn Hyperplane

Problem

- *Given:* training set S
- *Goal:* coefficients $\vec{w}$ and b of a separating hyperplane
- *Difficulty:* several or no candidates for $\vec{w}$ and b

Solution [cf. Vapnik's statistical learning theory]

Select admissible $\vec{w}$ and b with maximal margin (minimal distance to any input data vector)

Observation

We can scale $\vec{w}$ and b such that

$$\vec{w} \cdot \vec{x}_i + b \begin{cases} \geq 1 & \text{if } y_i = 1 \\ \leq -1 & \text{if } y_i = -1 \end{cases}$$

Maximizing the Margin

- Closest points $\vec{x}_+$ and $\vec{x}_-$ (with $\vec{w} \cdot \vec{x}_\pm + b = \pm 1$)
- Distance between $\vec{w} \cdot \vec{x} + b = \pm 1$:

$$\frac{(\vec{w} \cdot \vec{x}_+ + b) - (\vec{w} \cdot \vec{x}_- + b)}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|} = \frac{2}{\sqrt{\vec{w} \cdot \vec{w}}}$$

- $\max_{\vec{w},b} \frac{2}{\sqrt{\vec{w}\cdot\vec{w}}} \equiv \min_{\vec{w},b} \frac{\vec{w}\cdot\vec{w}}{2}$

Basic (Primal) Support Vector Machine Form

target: $\min_{\vec{w},b} \frac{1}{2}(\vec{w} \cdot \vec{w})$

subject to: $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 \qquad (i = 1, \ldots, k)$

Non-separable Data

Problem

Maybe a linear separating hyperplane does not exist!

Solution

Allow training errors ξ_i penalized by large penalty parameter C

Standard (Primal) Support Vector Machine Form

target: $\min_{\vec{w},b,\vec{\xi}} \frac{1}{2}(\vec{w}\cdot\vec{w}) + C\left(\sum_{i=1}^{k} \xi_i\right)$

subject to:
$$y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 - \xi_i \qquad (i = 1, \ldots, k)$$
$$\xi_i \geq 0$$

If $\xi_i > 1$, then misclassification of $\vec{x}_i$

Higher Dimensional Feature Spaces

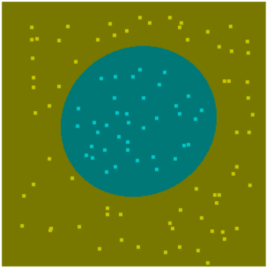
Problem

Data not separable because target function is essentially nonlinear!

Approach

Potentially separable in higher dimensional space

- Map input vectors nonlinearly into high dimensional space (feature space)
- Perform separation there



Higher Dimensional Feature Spaces

Literature

- Classic approach [Cover 65]
- "Kernel trick" [Boser, Guyon, Vapnik 92]
- Extension to soft margin [Cortes, Vapnik 95]

Example (cf. [Lin 01])

Mapping ϕ from $\mathbb{R}^3$ into feature space $\mathbb{R}^{10}$

$$\phi(\vec{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)$$

Adapted Standard Form

Definition

Standard (Primal) Support Vector Machine Form

target: $\min_{\vec{w},b,\vec{\xi}} \frac{1}{2}(\vec{w} \cdot \vec{w}) + C\left(\sum_{i=1}^{k} \xi_i\right)$

subject to:

$$y_i(\vec{w} \cdot \phi(\vec{x}_i) + b) \geq 1 - \xi_i \qquad (i = 1, \ldots, k)$$
$$\xi_i \geq 0$$

$\vec{w}$ is a vector in a high dimensional space

How to Solve?

Problem

Find $\vec{w}$ and b from the standard SVM form

Solution

Solve via Lagrangian dual [Bazaraa et al 93]:

$$\boxed{\max_{\vec{\alpha}\geq 0, \vec{\pi}\geq 0}\left(\min_{\vec{w}, b, \vec{\xi}} L(\vec{w}, b, \vec{\xi}, \vec{\alpha})\right)}$$

where

$$L(\vec{w}, b, \vec{\xi}, \vec{\alpha})$$

$$= \frac{\vec{w}\cdot\vec{w}}{2} + C\Big(\sum_{i=1}^{k}\xi_i\Big) + \sum_{i=1}^{k}\alpha_i(1 - \xi_i - y_i(\vec{w}\cdot\phi(\vec{x}_i) + b)) - \sum_{i=1}^{k}\pi_i\xi_i$$

Simplifying the Dual [Chen et al 03]

Standard (Dual) Support Vector Machine Form

target: $\min_{\vec{\alpha}} \frac{1}{2}(\vec{\alpha}^T \mathbf{Q} \vec{\alpha}) - \sum_{i=1}^{k} \alpha_i$

subject to: $\vec{y} \cdot \vec{\alpha} = 0$
$0 \leq \alpha_i \leq C \quad (i = 1, \ldots, k)$

where: $\mathbf{Q}_{ij} = y_i y_j \big(\phi(\vec{x}_i) \cdot \phi(\vec{x}_j)\big)$

Solution

We obtain $\vec{w}$ as

$$\vec{w} = \sum_{i=1}^{k} \alpha_i y_i \phi(\vec{x}_i)$$

Where is the Benefit?

- $\vec{\alpha} \in \mathbb{R}^k$ (dimension independent from feature space)
- Only inner products in feature space

Kernel Trick

- Inner products efficiently calculated on input vectors via kernel K

$$K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$$

- Select appropriate feature space
- Avoid nonlinear transformation into feature space
- Benefit from better separation properties in feature space

Example

Mapping into feature space ϕ: $\mathbb{R}^3 \rightarrow \mathbb{R}^{10}$

$$\phi(\vec{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \ldots, \sqrt{2}x_2x_3)$$

Kernel $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j) = (1 + \vec{x}_i \cdot \vec{x}_j)^2$.

Popular Kernels

- Gaussian Radial Basis Function:
  (feature space is an infinite dimensional Hilbert space)

$$g(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \|\vec{x}_i - \vec{x}_j\|^2)$$

- Polynomial: $g(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^d$

The Decision Function

Observation

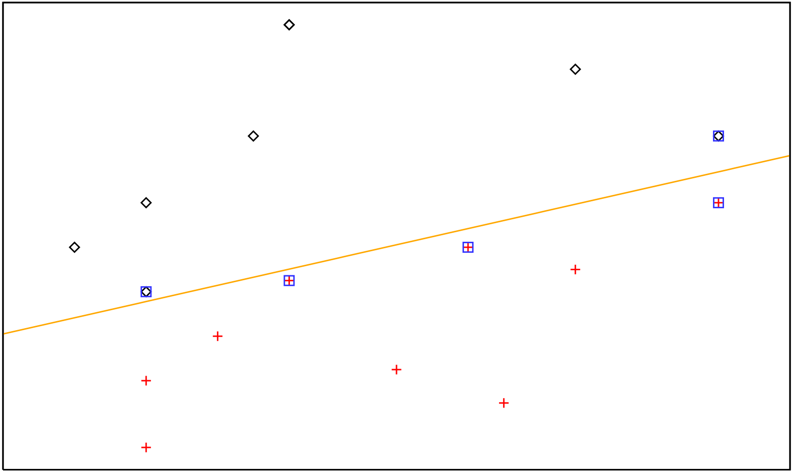
- *No need for $\vec{w}$ because*

$$f(\vec{x}) = \mathrm{sgn}\big(\vec{w} \cdot \phi(\vec{x}) + b\big) = \mathrm{sgn}\Big(\sum_{i=1}^{k} \alpha_i y_i \big(\phi(\vec{x}_i) \cdot \phi(\vec{x})\big) + b\Big)$$

- *Uses only $\vec{x}_i$ (support vectors) where $\alpha_i > 0$*

Few points determine the separation; borderline points

Support Vectors



Support Vector Machines

Definition

- *Given:* Kernel K and training set S
- *Goal:* decision function f

$$\text{target:} \quad \min_{\vec{\alpha}}\left(\frac{\vec{\alpha}^T \mathbf{Q} \vec{\alpha}}{2} - \sum_{i=1}^{k} \alpha_i\right) \quad \mathbf{Q}_{ij} = y_i y_j K(\vec{x}_i, \vec{x}_j)$$

$$\text{subject to:} \quad \begin{array}{l} \vec{y} \cdot \vec{\alpha} = 0 \\ 0 \leq \alpha_i \leq C \end{array} \quad (i = 1, \ldots, k)$$

$$\text{decide:} \quad f(\vec{x}) = \operatorname{sgn}\left(\sum_{i=1}^{k} \alpha_i y_i K(\vec{x}_i, \vec{x}) + b\right)$$

Quadratic Programming

- Suppose $\mathbf{Q}$ (k by k) fully dense matrix
- 70,000 training points $\rightsquigarrow$ 70,000 variables
- $70,000^2 \cdot 4\mathrm{B} \approx 19\mathrm{GB}$: huge problem
- Traditional methods: Newton, Quasi Newton cannot be directly applied
- Current methods:
  - Decomposition [Osuna et al 97], [Joachims 98], [Platt 98]
  - Nearest point of two convex hulls [Keerthi et al 99]

Sample Implementation

www.kernel-machines.org

- Main forum on kernel machines
- Lists over 250 active researchers
- 43 competing implementations

libSVM [Chang, Lin 06]

- Supports binary and multi-class classification and regression
- Beginners Guide for SVM classification
- "Out of the box"-system (automatic data scaling, parameter selection)
- Won EUNITE and IJCNN challenge

Application Accuracy

Automatic Training using libSVM

Application	Training Data	Features	Classes	Accuracy
Astroparticle	3,089	4	2	96.9%
Bioinformatics	391	20	3	85.2%
Vehicle	1,243	21	2	87.8%

References

Books

- Statistical Learning Theory (Vapnik). Wiley, 1998
- Advances in Kernel Methods—Support Vector Learning (Schölkopf, Burges, Smola). MIT Press, 1999
- An Introduction to Support Vector Machines (Cristianini, Shawe-Taylor). Cambridge Univ., 2000
- Support Vector Machines—Theory and Applications (Wang). Springer, 2005

References

Seminal Papers

- A training algorithm for optimal margin classifiers (Boser, Guyon, Vapnik). COLT'92, ACM Press.
- Support vector networks (Cortes, Vapnik). *Machine Learning* 20, 1995
- Fast training of support vector machines using sequential minimal optimization (Platt). In *Advances in Kernel Methods*, MIT Press, 1999
- Improvements to Platt's SMO algorithm for SVM classifier design (Keerthi, Shevade, Bhattacharyya, Murthy). Technical Report, 1999

References

Recent Papers

- A tutorial on ν-Support Vector Machines (Chen, Lin, Schölkopf). 2003
- Support Vector and Kernel Machines (Nello Christianini). ICML, 2001
- libSVM: A library for Support Vector Machines (Chang, Lin). System Documentation, 2006

Sequential Minimal Optimization [Platt 98]

- Commonly used to solve standard SVM form
- Decomposition method with smallest working set, $|B| = 2$
- Subproblem analytically solved; no need for optimization software
- Contained flaws; modified version [Keerthi et al 99]
- Karush-Kuhn-Tucker (KKT) of the dual ($\vec{E} = (1, \ldots, 1)$):

$$\begin{aligned} \mathbf{Q}\vec{\alpha} - \vec{E} + b\vec{y} - \vec{\lambda} + \vec{\mu} &= 0 & & \\ \mu_i(C - \alpha_i) &= 0 & \qquad \vec{\mu} &\geq 0 \\ \alpha_i \lambda_i &= 0 & \qquad \vec{\lambda} &\geq 0 \end{aligned}$$

Computing b

- KKT yield

$$(\mathbf{Q}\vec{\alpha} - \vec{E} + b\vec{y})_i \begin{cases} \geq 0 & \text{if } \alpha_i < C \\ \leq 0 & \text{if } \alpha_i > 0 \end{cases}$$

- Let $F_i(\vec{\alpha}) = \sum_{j=1}^{k} \alpha_j y_j K(\vec{x}_i, \vec{x}_j) - y_i$ and

$$\begin{array}{ll} I_0 = \{i \mid 0 < \alpha_i < C\} & \\ I_1 = \{i \mid y_i = 1, \alpha_i = 0\} & I_2 = \{i \mid y_i = -1, \alpha_i = C\} \\ I_3 = \{i \mid y_i = 1, \alpha_i = C\} & I_4 = \{i \mid y_i = -1, \alpha_i = 0\} \end{array}$$

- Case analysis on y_i yields bounds on b

$$\max\{F_i(\vec{\alpha}) \mid i \in I_0 \cup I_3 \cup I_4\} \leq b \leq \min\{F_i(\vec{\alpha}) \mid i \in I_0 \cup I_1 \cup I_2\}$$

Working Set Selection

Observation (see [Keerthi et al 99])

$\vec{\alpha}$ *not* *optimal solution* *iff*

$$\max\{F_i(\vec{\alpha}) \mid i \in I_0 \cup I_3 \cup I_4\} > \min\{F_i(\vec{\alpha}) \mid i \in I_0 \cup I_1 \cup I_2\}$$

Approach

Select working set $B = \{i, j\}$ with

$$i \equiv \arg\max_m\{F_m(\vec{\alpha}) \mid m \in I_0 \cup I_3 \cup I_4\}$$
$$j \equiv \arg\min_m\{F_m(\vec{\alpha}) \mid m \in I_0 \cup I_1 \cup I_2\}$$

The Subproblem

Definition

Let $B = \{i, j\}$ and $N = \{1, \ldots, k\} \setminus B$.

- $\vec{\alpha}_B = \begin{pmatrix} \alpha_i \\ \alpha_j \end{pmatrix}$ and $\vec{\alpha}_N = \vec{\alpha}|_N$ (similar for matrices)

B-Subproblem

target: $\min_{\vec{\alpha}_B} \dfrac{\vec{\alpha}_B^T \mathbf{Q}_{BB} \vec{\alpha}_B}{2} + \left(\sum_{b \in B} \alpha_b \mathbf{Q}_{b,N} \vec{\alpha}_N \right) - \left(\sum_{b \in B} \alpha_b \right)$

subject to:

$$\vec{y} \cdot \vec{\alpha} = 0$$

$$0 \leq \alpha_i, \alpha_j \leq C$$

Final Solution

- Note that $-y_i\alpha_i = \vec{y}_N \cdot \vec{\alpha}_N + y_j\alpha_j$
- Substitute $\alpha_i = -y_i(\vec{y}_N \cdot \vec{\alpha}_N + y_j\alpha_j)$ into target
- $\rightsquigarrow$ One-variable optimization problem
- Can be solved analytically (cf., e.g., [Lin 01])
- Iterate (yielding new $\vec{\alpha}$) until

  $$\max\{F_i(\vec{\alpha}) \mid i \in I_0 \cup I_3 \cup I_4\} \leq \min\{F_i(\vec{\alpha}) \mid i \in I_0 \cup I_1 \cup I_2\} - \epsilon$$