Pattern Information Processing: Support Vector Machines

Masashi Sugiyama

(Department of Computer Science)

Contact: W8E-505

sugi@cs.titech.ac.jp

http://sugiyama-www.cs.titech.ac.jp/~sugi/

(Binary) Classification Problem

- Output values are $y_i = \pm 1$
- We want to predict whether output values of unlearned input points are positive/negative.

- Multi-class problem can be transferred to several binary classification problems:
  - One-versus-rest
  - One-versus-one

(Binary) Classification Problem

- In classification, we may still use the same learning methods, e.g., quadratically-constrained least-squares:

$$\hat{\boldsymbol{\alpha}}_{QCLS} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^b}{\operatorname{argmin}} \left[J_{LS}(\boldsymbol{\alpha}) + \lambda \langle \boldsymbol{R}\boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle \right]$$

$$\lambda \ (\geq 0)$$

$$J_{LS}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \left(\hat{f}(\boldsymbol{x}_i) - y_i \right)^2$$

- Prediction:

$$\widehat{y} = \operatorname{sign}\left(\hat{f}(\boldsymbol{x}) \right)$$

0/1-Loss

- In classification, only the sign of the learned function is used.
- It is natural to use 0/1-loss instead of squared-loss $J_{LS}(\boldsymbol{\alpha})$:

$$J_{0/1}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{n}(1 - \mathrm{sign}\,(u_i))$$

$$u_i = \hat{f}(\boldsymbol{x}_i)y_i$$

- $J_{0/1}(\boldsymbol{\alpha})$ corresponds to the number of misclassified samples (thus natural).

Hinge-Loss

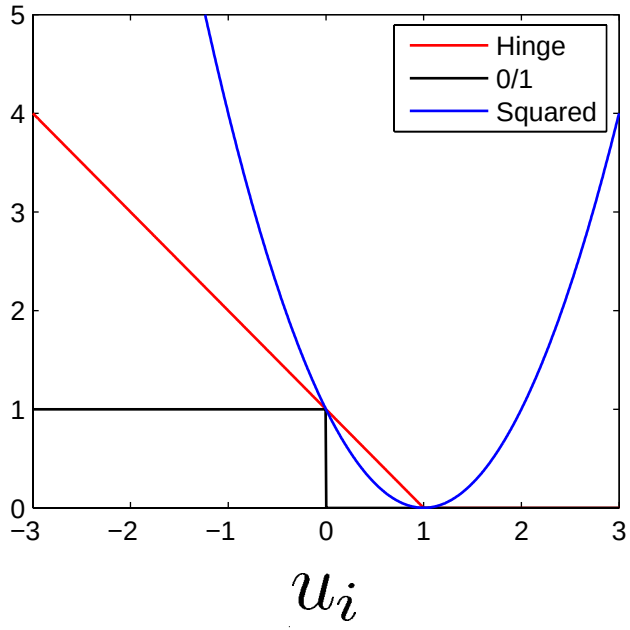
- However, $J_{0/1}(\boldsymbol{\alpha})$ is non-convex so we may not obtain the global minimizer.
- Use hinge-loss as an approximation:

$$J_H(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \max(0, 1 - u_i)$$

$$J_{0/1}(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{i=1}^{n} (1 - \text{sign}(u_i))$$

$$J_{LS}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} (1 - u_i)^2$$

Note : $y_i^2 = 1$



- Hinge-loss is a tighter upper bound on 0/1-loss than squared-loss (thus better?).

How to Obtain Solutions

$$\hat{\boldsymbol{\alpha}}_{SVM} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^b}{\operatorname{argmin}} \left[J_H(\boldsymbol{\alpha}) + \lambda \langle \boldsymbol{R}\boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle \right]$$

$$J_H(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \max(0, 1 - u_i)$$

- How to deal with "max"? Use following lemma:

Lemma $\max(0, 1-u) = \min_{\xi\in\mathbb{R}} \xi$

$$\text{subject to } \xi \geq 1 - u$$

$$\xi \geq 0$$

- Proof: Constraint is $\xi \geq \max(0, 1-u)$,

so $\min_{\xi\in\mathbb{R}} \xi = \max(0, 1-u)$ Q.E.D.

How to Obtain Solutions (cont.)

- So we have

$$J_H(\boldsymbol{\alpha}) = \min_{\boldsymbol{\xi} \in \mathbb{R}^n} \langle \mathbf{1}_n, \boldsymbol{\xi} \rangle \text{ subject to } \boldsymbol{\xi} \geq \mathbf{1}_n - \boldsymbol{u}$$
$$\boldsymbol{\xi} \geq \mathbf{0}_n$$

- Then $\hat{\boldsymbol{\alpha}}_{SVM}$ is given as

$$\hat{\boldsymbol{\alpha}}_{SVM} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^b, \boldsymbol{\xi} \in \mathbb{R}^n}{\operatorname{argmin}} \left[\langle \mathbf{1}_n, \boldsymbol{\xi} \rangle + \lambda \langle \boldsymbol{R}\boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle \right]$$
$$\text{subject to } \boldsymbol{\xi} \geq \mathbf{1}_n - \boldsymbol{u}$$
$$\boldsymbol{\xi} \geq \mathbf{0}_n$$

Support Vector Machines

- We focus on the following setting:
  - $\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b$
  - $\boldsymbol{R} = \boldsymbol{K}$ $\qquad \boldsymbol{K}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ $\qquad {}^{\exists}\boldsymbol{K}^{-1}$
- Putting $\lambda = (2C)^{-1}$ (convention), we have

$$\hat{\boldsymbol{\alpha}}_{SVM} = \underset{\boldsymbol{\alpha},\boldsymbol{\xi}\in\mathbb{R}^n, b\in\mathbb{R}}{\operatorname{argmin}} \left[C\langle \mathbf{1}_n, \boldsymbol{\xi}\rangle + \frac{1}{2}\langle \boldsymbol{K}\boldsymbol{\alpha}, \boldsymbol{\alpha}\rangle \right]$$

$$\text{subject to } \boldsymbol{\xi} \geq \mathbf{1}_n - \boldsymbol{u}$$

$$\boldsymbol{\xi} \geq \mathbf{0}_n$$

$$u_i = \hat{f}(\boldsymbol{x}_i) y_i$$

How to Obtain Solutions (cont.)

- SVM solution can be obtained by solving a linearly constrained quadratic programming problem.
- However, we need to optimize $2n + 1$ variables, which could be time consuming.
- Consider a dual formulation.

How to Obtain Solutions (cont.)

- Lagrangian:

$$L(\boldsymbol{\alpha}, b, \boldsymbol{\xi}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = C\langle \mathbf{1}_n, \boldsymbol{\xi}\rangle + \frac{1}{2}\langle \boldsymbol{K}\boldsymbol{\alpha}, \boldsymbol{\alpha}\rangle - \langle \boldsymbol{\beta}, \boldsymbol{\xi} + \boldsymbol{u} - \mathbf{1}_n\rangle - \langle \boldsymbol{\gamma}, \boldsymbol{\xi}\rangle$$

- $\boldsymbol{\beta}, \boldsymbol{\gamma}$:Lagrange multiplier
- Wolfe-dual problem is simpler:

$$\min_{\boldsymbol{\alpha}, \boldsymbol{\xi} \in \mathbb{R}^n, b \in \mathbb{R}} \left[C\langle \mathbf{1}_n, \boldsymbol{\xi}\rangle + \frac{1}{2}\langle \boldsymbol{K}\boldsymbol{\alpha}, \boldsymbol{\alpha}\rangle \right] = \max_{\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbb{R}^n} L(\boldsymbol{\alpha}, b, \boldsymbol{\xi}, \boldsymbol{\beta}, \boldsymbol{\gamma})$$

$$\text{subject to } \boldsymbol{\xi} \geq \mathbf{1}_n - \boldsymbol{u} \qquad \text{subject to } \boldsymbol{\beta} \geq \mathbf{0}_n \quad \boldsymbol{\gamma} \geq \mathbf{0}_n$$

$$\boldsymbol{\xi} \geq \mathbf{0}_n \qquad \frac{\partial L}{\partial \boldsymbol{\alpha}} = \mathbf{0}_n \quad \frac{\partial L}{\partial b} = 0 \quad \frac{\partial L}{\partial \boldsymbol{\xi}} = \mathbf{0}_n$$

How to Obtain Solutions (cont.)

- Constraints yield
  - $\frac{\partial L}{\partial \boldsymbol{\xi}} = C\mathbf{1}_n - \boldsymbol{\beta} - \boldsymbol{\gamma} = \mathbf{0}_n$

    $\boldsymbol{\gamma} \geq \mathbf{0}_n$

    $\boldsymbol{\beta} \geq \mathbf{0}_n$

    $\Rightarrow \mathbf{0}_n \leq \boldsymbol{\beta} \leq C\mathbf{1}_n$
  - $\frac{\partial L}{\partial b} = -\langle \mathbf{1}_n, \boldsymbol{v} \rangle = 0 \Rightarrow \langle \boldsymbol{y}, \boldsymbol{\beta} \rangle = 0$

    $v_i = \beta_i y_i$
  - $\frac{\partial L}{\partial \boldsymbol{\alpha}} = \boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{v} = \mathbf{0}_n \Rightarrow \boldsymbol{\alpha} = \boldsymbol{v}$

How to Obtain Solutions (cont.)

- Then Wolfe-dual problem is simplified as

$$\widehat{\boldsymbol{\beta}}_{SVM} = \underset{\boldsymbol{\beta}\in\mathbb{R}^n}{\operatorname{argmax}} \left[\sum_{i=1}^{n} \beta_i - \frac{1}{2} \sum_{i,j=1}^{n} \beta_i \beta_j y_i y_j \boldsymbol{K}_{i,j} \right]$$

$$\text{subject to } \mathbf{0}_n \leq \boldsymbol{\beta} \leq C\mathbf{1}_n$$

$$\langle \boldsymbol{y}, \boldsymbol{\beta} \rangle = 0$$

- So we have

$$[\hat{\boldsymbol{\alpha}}_{SVM}]_i = [\widehat{\boldsymbol{\beta}}_{SVM}]_i y_i$$

How to Obtain Solutions (cont.)

- Standard form of the Wolfe-dual problem is

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^n} \left[\frac{1}{2}\langle \boldsymbol{Q}\boldsymbol{\beta},\boldsymbol{\beta}\rangle + \langle \boldsymbol{\beta},\boldsymbol{q}\rangle\right]$$

$$\text{subject to } \boldsymbol{V}\boldsymbol{\beta} \leq \boldsymbol{v}$$

$$\boldsymbol{G}\boldsymbol{\beta} = \boldsymbol{g}$$

$$\boldsymbol{Q}_{i,j} = \boldsymbol{K}_{i,j} y_i y_j \qquad \boldsymbol{V} = \begin{pmatrix} -\boldsymbol{I}_n \\ \boldsymbol{I}_n \end{pmatrix} \qquad \boldsymbol{G} = \boldsymbol{y}^\top$$

$$\boldsymbol{q} = -\mathbf{1}_n \qquad \boldsymbol{v} = \begin{pmatrix} \mathbf{0}_n \\ C\mathbf{1}_n \end{pmatrix} \qquad \boldsymbol{g} = \mathbf{0}_n$$

How to Obtain Solutions (cont.)

- $\hat{b}_{SVM}$ is computed using any j such that $0 < [\widehat{\boldsymbol{\beta}}_{SVM}]_j < C$ as

$$\hat{b}_{SVM} = y_j - \sum_{i=1}^{n} [\hat{\boldsymbol{\alpha}}_{SVM}]_i K(\boldsymbol{x}_j, \boldsymbol{x}_i)$$

- In practice, we may take the average:

$$\hat{b}_{SVM} = \frac{1}{m} \sum_{j:0<[\widehat{\boldsymbol{\beta}}_{SVM}]_j<C} \left[y_j - \sum_{i=1}^{n} [\hat{\boldsymbol{\alpha}}_{SVM}]_i K(\boldsymbol{x}_j, \boldsymbol{x}_i) \right]$$

m :Number of non-zero elements in $\hat{\boldsymbol{\alpha}}_{SVM}$

Sparseness

- KKT condition:

$$u_i = \hat{f}(\boldsymbol{x}_i) y_i$$

$$\beta_i(\xi_i + u_i - 1) = 0 \quad \text{for all } i$$

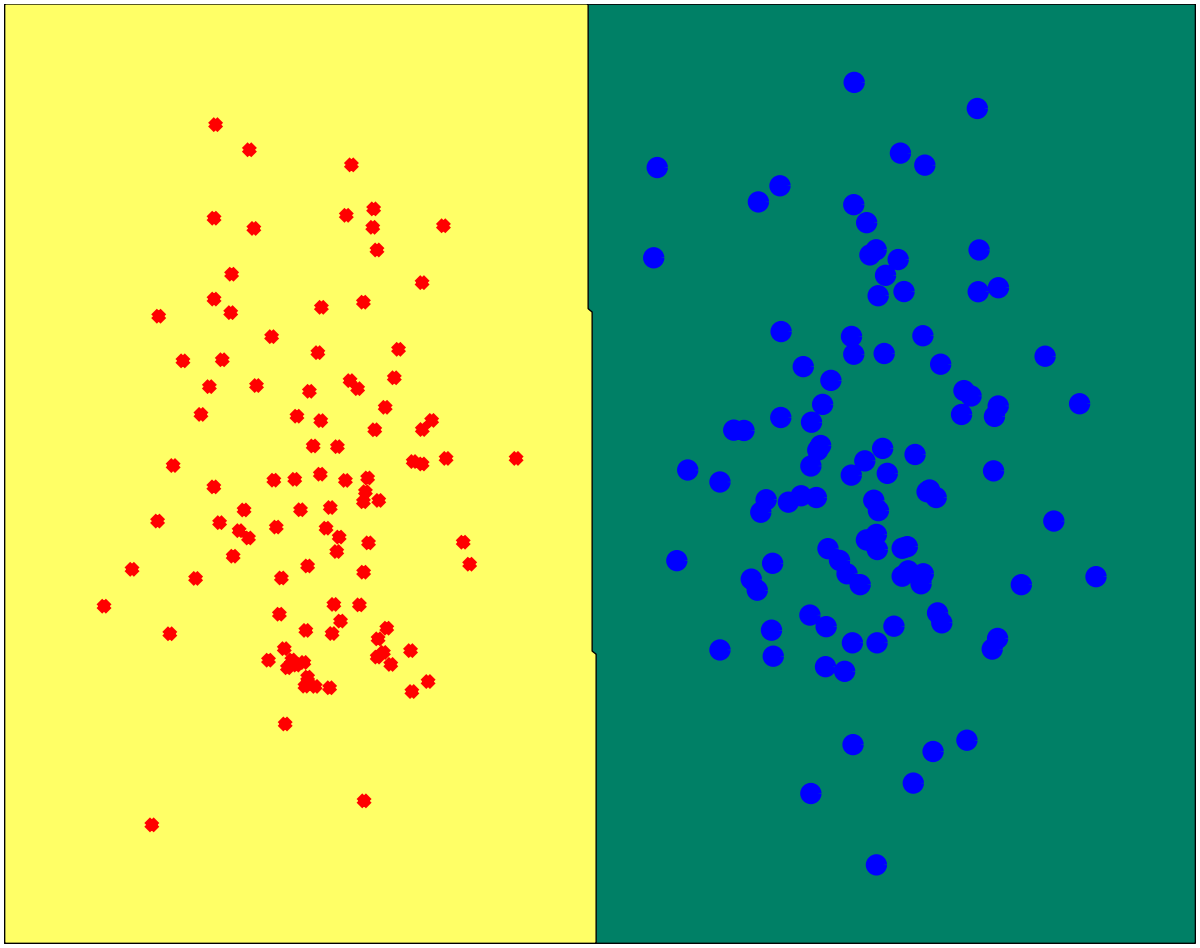
- Therefore, some β_i (and thus α_i) could be zero.

$$\alpha_i = \beta_i y_i$$

See e.g.,

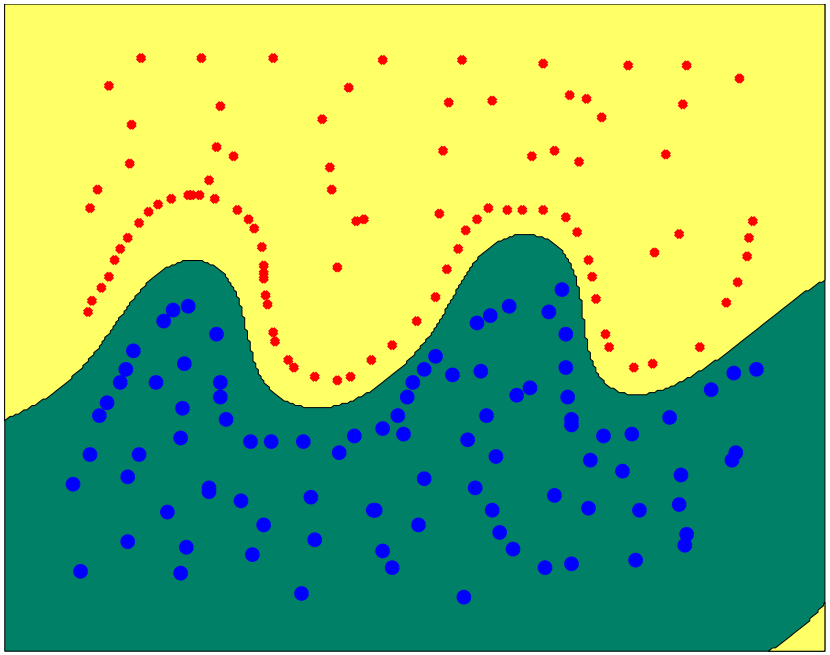
- Evgeniou, Pontil & Poggio, Regularization Networks and Support Vector Machines, Advances in Computational Mathematics, 2000, 13(1), 1-50.
- B. Schölkopf and A. Smola.Learning with Kernels. MIT Press, 2002
- http://www.kernel-machines.org

Examples

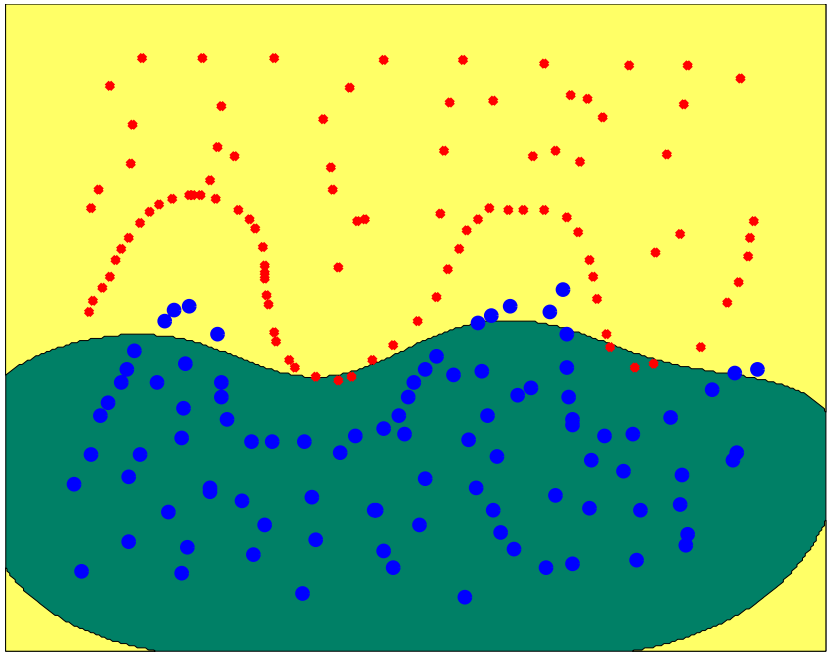


Gaussian kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2c^2}\right)$

Examples (cont.)

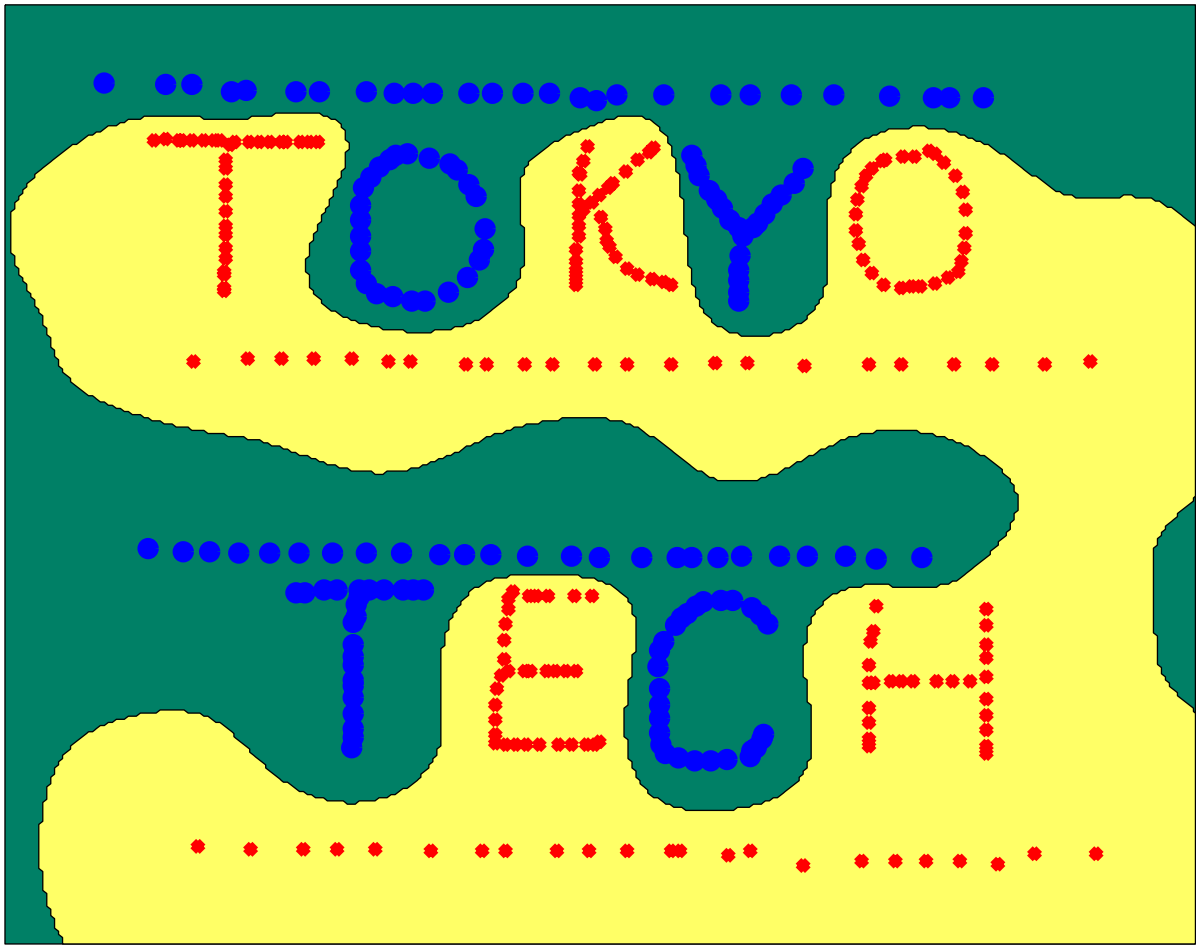


Small C



Large C

Examples



Homework

1. Originally, SVMs are derived within a totally different framework, i.e., maximum margin principle. Read the following article on SVMs (Sec.1-Sec.4) and write your opinion.
   - B. Schölkopf: Statistical learning and kernel methods.
     ftp://ftp.research.microsoft.com/pub/tr/tr-2000-23.pdf

Homework (cont.)

2. Implement SVM by yourself or find a suitable software. Then perform simulations under various settings (various data sets, changing kernel, changing C etc.) and analyze the results.
   - Software is available from, e.g., http://www.kernel-machines.org
   - You may play with Java implementation, e.g., http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml