パターン認識
Pattern Recognition

佐藤真一
Shin'ichi Satoh

国立情報学研究所
National Institute of Informatics

June 6, 2023

Schedule (subject to change)

5/23	Hybrid	Linear classifier, perceptron, MSE classifier, Widrow-Hoff rule
5/30	Online, zoom only	neural network, deep learning
6/6	Hybrid	all about SVM
6/13	Online, zoom only	Orthogonal expansions, Eigenvalue decomposition
6/20		no class
6/27	Hybrid→may be Online	Clustering, dendrogram, aggromerative clustering, k-means
7/4	Hybrid	Graphs, normalized cut, spectral clustering, Laplacian Eigenmaps
7/11		extra (if needed)

Introduction of Support Vector Machines

- Original Support Vector Machines (SVM) algorithm, aka linear SVM, was invented by Vladimir N. Vapnik in 1960s.
- SVM minimizes “Structural Risk” which combines training error (empirical risk) and the complexity of the model (the VC dimension) and can thus effectively avoid overfitting.
- Nonlinear extension by kernel trick was suggested by Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik in 1992. [1]
- Soft margin was proposed by Corinna Cortes and Vapnik in 1993. [2]

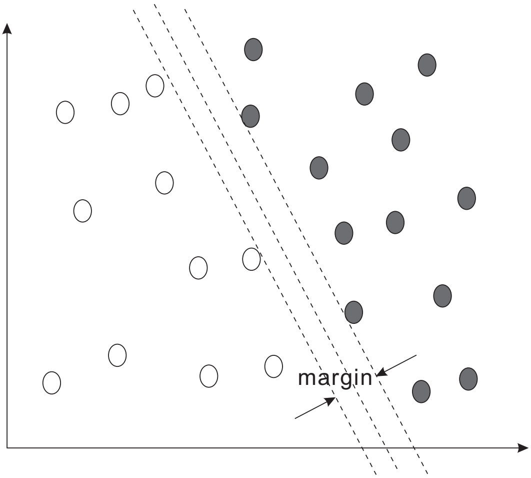
---

[1] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. Proc. of COLT, 1992.

[2] Cortes, C., Vapnik, V. Support-vector networks. Mach Learn 20, 273–297 (1995).

Linear Support Vector Machines: Intuition

Linear Support Vector Machines select discriminant plane with margin maximized.



Linear Support Vector Machines: Intuition

Consider the set of training data $\mathcal{X} = \{x_i \in \mathbb{R}^d\}$, $i = 1, \cdots, n$ and their labels $y_i \in \{-1, 1\}$.
We now assume that the training data is linearly separable.
We want to find discriminant (hyper-)plane:

$$w \cdot x - b = 0$$

with maximum-margin.
The margin is then represented as a margin between two hyper-planes:

$$w \cdot x - b = 1 \text{ and } w \cdot x - b = -1.$$

The margin is then

$$\frac{2}{\|w\|}$$

.

Linear Support Vector Machines: Formulation

So we want to minimize $\|w\|$ with constraints:

$$w \cdot x_i - b \geq 1 \text{ for } x_i \text{ with } y_i = 1$$
$$w \cdot x_i - b \leq -1 \text{ for } x_i \text{ with } y_i = -1.$$

The constraints are equivalent to:

$$y_i(w \cdot x_i - b) \geq 1 \text{ for all } i = 1, \cdots, n.$$

So we obtain the optimization problem:

Minimize $\|w\|$
subject to $y_i(w \cdot x_i - b) \geq 1$ for all $i = 1, \cdots, n$.

Linear Support Vector Machines: Primal form

The problem can be formulated as a quadratic programming optimization problem as follows:

$$\arg\min_{(w,b)} \frac{1}{2}\|w\|^2$$

subject to (for any $i = 1, \cdots, n$)

$$y_i(w \cdot x_i - b) \geq 1.$$

Linear Support Vector Machines: Dual form

By introducing Lagrange multipliers α_i, the problem will then be:

$$L = \left\{ \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w \cdot x_i - b) - 1] \right\}$$

$$\arg\min_{w,b} \max_{\alpha} L$$

with Karush-Kuhn-Tucker conditions:

$$\frac{\partial L}{\partial w} = 0, \frac{\partial L}{\partial b} = 0$$
$$\alpha_i \geq 0, \alpha_i [y_i(w \cdot x_i - b) - 1] = 0$$

Karush-Kuhn-Tucker conditions

Maximize $f(x)$
subject to $g_i(x) \leq 0,\ h_j(x) = 0$

Karush-Kuhn-Tucker conditions:

$$\nabla f(x) - \sum \alpha_i \nabla g_i(x) - \sum \lambda_j \nabla h_j(x) = 0$$

$$g_i(x) \leq 0, h_j(x) = 0$$

$$\alpha_i \geq 0$$

$$\alpha_i g_i(x) = 0$$

Lagrange Multipliers Method

Maximize $f(x)$
subject to $h_j(x) = 0$

Lagrangian:

$$L = f(x) - \sum \lambda_j h_j(x)$$

Conditions:

$$\nabla L = \nabla f(x) - \sum \lambda_j \nabla h_j(x) = 0$$

$$h_j(x) = 0$$

Linear Support Vector Machines: Dual form

$$\frac{\partial L}{\partial w} = 0 \;\rightarrow\; w = \sum \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \;\rightarrow\; \sum \alpha_i y_i = 0$$

$$\alpha_i [y_i (w \cdot x_i - b) - 1] = 0 \;\rightarrow\; \cdots$$

if $y_i(w \cdot x_i - b) - 1 > 0$ then $\alpha_i = 0$, otherwise $(y_i(w \cdot x_i - b) - 1 = 0)$ $\alpha_i > 0$
x_i corresponding to $\alpha_i > 0$ is called support vector.

$$b = \frac{1}{|\{\alpha_i > 0\}|} \sum_{\alpha_i > 0} (w \cdot x_i - y_i)$$

Linear Support Vector Machines: Dual form

Put everything back to the original problem:
Maximize:

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to:

$$\alpha_i \geq 0 \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

We solve this by quadratic programming optimization method.

Quadratic Programming Optimization

Minimize

$$\frac{1}{2}x^TQx + p^Tx$$

subject to

$$Cx \leq b$$

$$C_{eq}x = b_{eq}$$

$$LB \leq x \leq UB$$

In our case,

$$x = \alpha,\ Q_{i,j} = y_i y_j x_i^T x_j,\ p = -[1\,1\,\cdots\,1],$$

$$LB = 0,\ C_{eq} = y,\ b_{eq} = 0$$

Linear Support Vector Machines: Implementation (Python)

```
h = x * l
qpP = cvxopt.matrix(h.T.dot(h))
qpq = cvxopt.matrix(-np.ones(n), (n, 1))
qpG = cvxopt.matrix(-np.eye(n))
qph = cvxopt.matrix(np.zeros(n), (n, 1))
qpA = cvxopt.matrix(l.astype(float), (1, n))
qpb = cvxopt.matrix(0.)
cvxopt.solvers.options['abstol'] = 1e-5
cvxopt.solvers.options['reltol'] = 1e-10
cvxopt.solvers.options['show_progress'] = False
res = cvxopt.solvers.qp(qpP, qpq, qpG, qph, qpA, qpb)
alpha = np.reshape(np.array(res['x']), -1)

w = np.sum(x * (np.ones(n) * (l * alpha)), axis=1)
sv = alpha > 1e-5
isv = np.where(sv)[-1]
b = np.sum(w.T.dot(x[:, isv]) - l[isv]) / np.sum(sv)
```

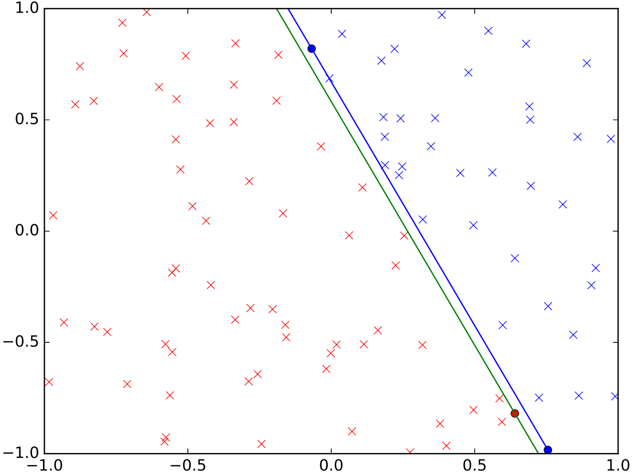
Linear Support Vector Machines: Implementation (Matlab)

```
h=x;
h(:,l<0)=-h(:,l<0);
options=optimset('Algorithm','interior-point-convex');
alpha=quadprog(h'*h,-ones(1,size(x,2)),[],[],l,0,...
    zeros(1,size(x,2)),[],[],options)';
w=sum(x.*(ones(size(x,1),1)*(l.*alpha)),2);
sv=alpha>1e-5;
isv=find(sv);
b=sum(w'*x(:,isv)-l(isv))/sum(sv);
```

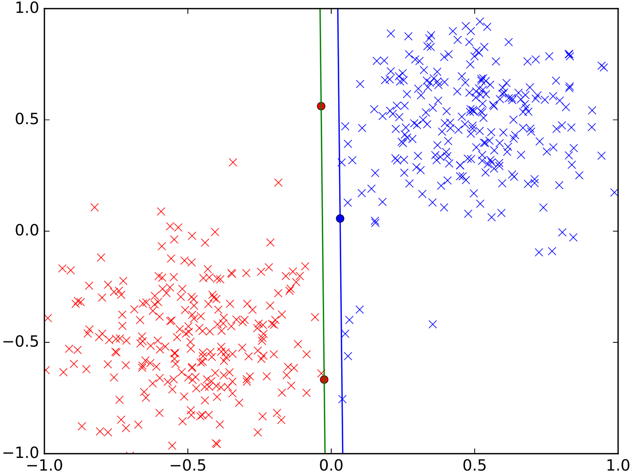
Linear Support Vector Machines: Implementation (Scilab)

```
h=x;
h(:,l<0)=-h(:,l<0);
alpha=quapro(h'*h,-ones(size(x,2),1),l,0,...
  zeros(size(x,2),1),[],1)';
w=sum(x.*(ones(size(x,1),1)*(l.*alpha)),2);
sv=alpha>1e-5;
isv=find(sv);
b=sum(w'*x(:,isv)-l(isv))/sum(sv);
```

Example (linear)



Example (slinear)



Linear Support Vector Machines: Soft Margin

What if the training data is not linearly separable?
We introduce soft margin to linearly separate the training data "as much as possible."
Non-negative slack variables ξ_i are introduced:

$$y_i(w \cdot x_i - b) \geq 1 - \xi_i$$

Our objective function is then

$$\arg\min_{w,\xi,b} \left\{ \frac{1}{2}\|w\|^2 + C \sum_{i}^{n} \xi_i \right\}$$

subject to

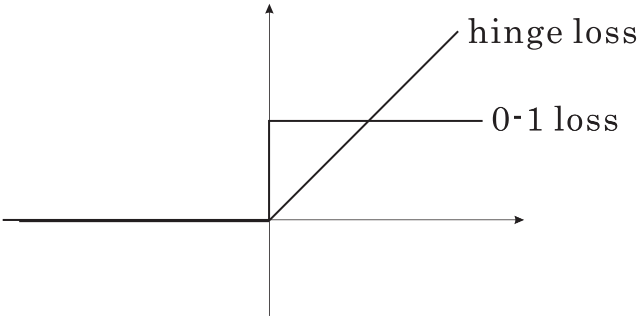
$$y_i(w \cdot x_i - b) \geq 1 - \xi_i, \ \xi \geq 0$$

Linear Support Vector Machines: Soft Margin

This is equivalent to

$$\arg\min_{w,b} \left\{ \frac{1}{2}\|w\|^2 + C\sum_{i}^{n} \max(1 - y_i(w \cdot x_i + b), 0) \right\}$$

$\max(1 - y_i(w \cdot x_i + b), 0)$ is called hinge loss.



Linear Support Vector Machines: Soft Margin

This is solved similarly using Lagrange Multipliers method with KKT conditions.
Maximize

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to

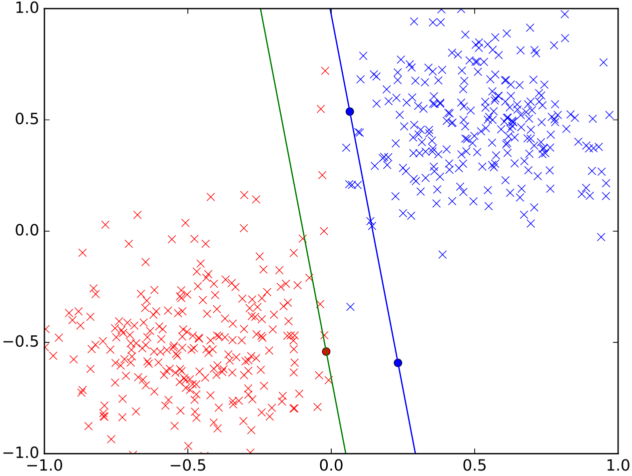
$$0 \leq \alpha_i \leq C, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Support vectors:
x_i with $0 < \alpha_i < C$ (x_i with $\alpha_i = C$ are misclassified).

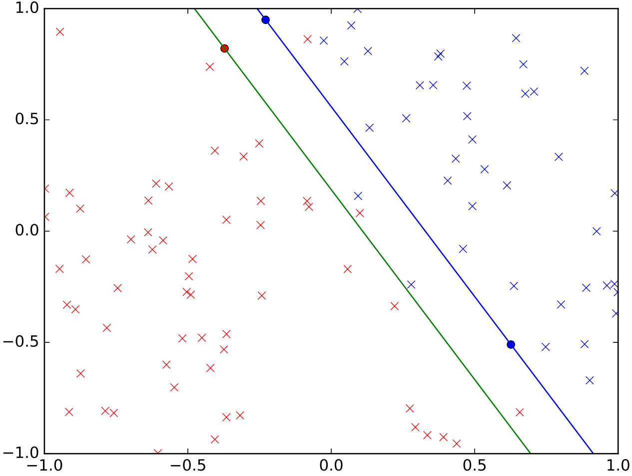
Linear Support Vector Machines: Soft Margin

Implementation: Try! Very straightforward.

Example (slinear)



Example (qlinear)



Support Vector Machines: Kernel Extension

- What if the data is severely linearly non-separable, which cannot be handled by soft margin?
- Converts input vector x with nonlinear mapping function, namely, $\phi(x)$, and applies linear discriminant function to the converted space.
- Example: $x = [x_1\ x_2]^T$, $\phi(x) = [x_1\ x_2\ x_1^2\ x_1 x_2\ x_2^2]^T$. Application of linear discriminant function to $\phi(x)$ is equivalent to applying quadratic discriminant function to x.

Support Vector Machines: Kernel Extension

- If explicit form of nonlinear mapping function works, we can simply convert data and apply linear SVM.
- However, in many interesting nonlinear mapping functions can be represented only as kernel functions.

$$k(x, y) = \phi(x) \cdot \phi(y)$$

- Examples:
    - Polynomial Kernel

$$k(x, y) = (x \cdot y + 1)^p, \; k(x, y) = (x \cdot y)^p$$

    - Gaussian Kernel (Radial Basis Function (RBF) Kernel)

$$k(x, y) = exp(-\frac{\|x - y\|^2}{2\sigma^2})$$

Support Vector Machines: Kernel Extension

Recall:
Maximize

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to

$$0 \leq \alpha_i \leq C, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Note that all x_i appear in dot products between x_i.

Support Vector Machines: Kernel Extension

So our problem is then:
Maximize

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$$
$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

subject to

$$0 \leq \alpha_i \leq C, \sum_{i=1}^{n} \alpha_i y_i = 0$$

.

Support Vector Machines: Kernel Extension

Suppose that α_i are obtained by QP.

$$w = \sum \alpha_i y_i \phi(x_i)$$

Note that w cannot be explicitly obtained.

$$\begin{aligned} b &= \frac{1}{\# sv} \sum_{i \in sv} (w \cdot \phi(x_i) - y_i) \\ &= \frac{1}{\# sv} \sum_{i \in sv} \left(\sum_j \alpha_j y_j \phi(x_j)^T \phi(x_i) - y_i \right) \\ &= \frac{1}{\# sv} \sum_{i \in sv} \left(\sum_j \alpha_j y_j k(x_j, x_i) - y_i \right) \end{aligned}$$

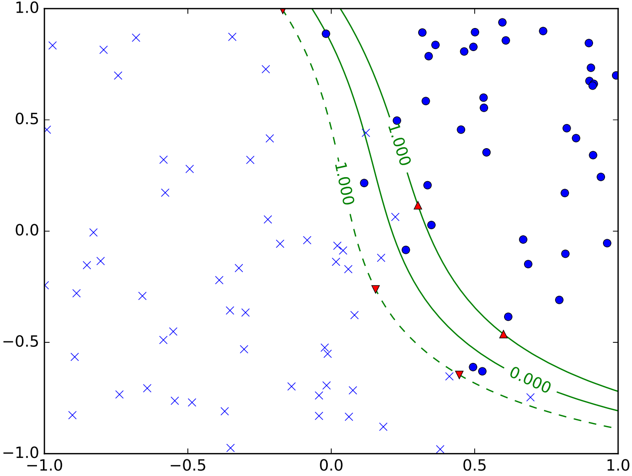
Support Vector Machines: Kernel Extension

Suppose we want to classify x.

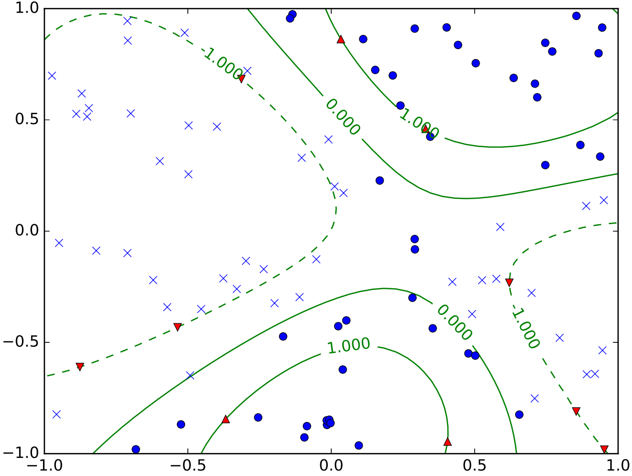
$$
\begin{aligned}
f(x) &= w \cdot \phi(x) - b \\
&= \sum \alpha_i y_i \phi(x_i)^T \phi(x) - b \\
&= \sum \alpha_i y_i k(x_i, x) - b
\end{aligned}
$$

We can then classify x according to the sign of $f(x)$.

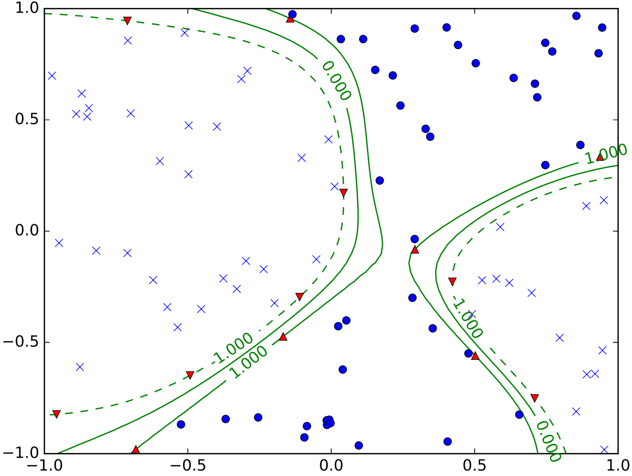
Example (qlinear, Polynomial kernel)



Example (nonlinear, C=1, RBF kernel)



Example (nonlinear, C=1000, RBF kernel)



Assignment

- Programming project and non-programming project are imposed.
- You are expected to solve either programming project OR non-programming project.
- Programming project is recommended.
- Of course you are most welcomed to solve both.
- Due on June 27.

Programming Project

- Extend Linear SVM to be able to handle soft margin and see how it works.
- Further extend to kernel version with RBF kernel and see how it works (extended project).
- Note: don't use existing SVM packages! Implement by yourself.
- QP solvers can be used.
  - Python: cvxopt ("conda install cvxopt" may work)
  - Matlab: quadprog (requires optimization toolbox)
  - Scilab: quapro (requires quapro toolbox)
- Try to classify couple of datasets: linear, slinear, qlinear, nonlinear.

Non-Programming Project 1

Show that the margin between the following two hyper-planes

$$w \cdot x - b = 1 \text{ and } w \cdot x - b = -1$$

is

$$\frac{2}{\|w\|}$$

.

Non-Programming Project 2

Given the primal form of soft-margin SVM:

$$\arg\min_{w,\xi,b} \left\{ \frac{1}{2}\|w\|^2 + C\sum_{i}^{n} \xi_i \right\}$$

subject to

$$y_i(w \cdot x_i - b) \geq 1 - \xi_i,\ \xi \geq 0$$

derive the dual form:
Maximize

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to

$$0 \leq \alpha_i \leq C,\ \sum_{i=1}^{n} \alpha_i y_i = 0$$

.