B. Schölkopf and A. Smola
Learning with kernels,
Chapter 7: Pattern Recognition
(7.1–7.4)
explained by Ramūnas Girdziušas

March 10, 2003
Introduction

Considering binary classification task

- *labelled examples* \((x_i, y_i) \in \mathcal{H} \times \{\pm 1\}\)
- *hyperplane* \(\{x \in \mathcal{H}|\langle w, x \rangle + b = 0\}, \ w \in \mathcal{H}, b \in \mathbb{R}\).
- *decision function* \(x \mapsto f_{w,b}(x) = \text{sgn}(\langle w, x \rangle + b)\).

it is good to seek for the decision function \(f_{w,b}(x)\) that

- correctly classifies given samples \(f_{w,b}(x_i) = y_i, \forall i\).
- maximizes the margin \(\rho = \min_i |(\langle w, x_i \rangle + b)|/||w||\).

This presentation

- is about what the margin is, why to maximize it, and how to do it;
- serves as an intro to the *Support Vector Classifier*. 
Definition \( \rho = \min_i \left| \left( \langle w, x_i \rangle + b \right) \right| / \|w\| \) clarified

Kuva 1: \( \rho = \left| \langle w, x \rangle + b \right| / \|w\| = \left| \langle w, d \rangle \right| / \|w\| = \text{const} / \|w\| \).
Why to maximize $\rho$?

Kuva 2: Larger margin classifier tolerates bounded noise when $r < \rho$ (a). There is also parameter insensitivity when $|\Delta\gamma| < \arcsin \frac{\rho}{R}$ (b).
Theorem 7.3 (Margin Error Bound)

Consider the set of decision functions $f(x) = \text{sign} \langle w, x \rangle$ with $||w|| \leq \Lambda$ and $x \leq R$, for some $R, \Lambda > 0$. Moreover, let $\varrho > 0$, and $\nu$ denote the fraction of training examples with margin smaller than $\varrho/||w||$, referred to as the margin error.

For all distributions $P$ generating the data, with probability at least $1 - \delta$ over the drawing of the $m$ training patterns, and for any $\varrho > 0$ and $\delta \in (0, 1)$, the probability that a test pattern drawn from $P$ will be misclassified is bounded from above, by

$$\nu + \sqrt{\frac{c}{m}} \left( \frac{R^2 \Lambda^2}{\varrho^2} \ln^2 m + \ln(1/\delta) \right),$$

(1)

c is a universal constant.
How to construct Optimal Margin Hyperplane?

Remember that margin is $\rho = \min_i |(\langle w, x_i \rangle + b) / \|w\||$. Notice that $\langle w, x \rangle + b = 0$ does not change if we multiply $w$ and $b$ by some constant. One can always choose it so that $\rho = 1 / \|w\|$. The so-called primal quadratic program will find the OMH:

$$w^*, b^* = \arg\min_{w, b} \frac{1}{2} \|w\|^2,$$

s.t. $y_i (\langle w, x_i \rangle + b) \geq 1, \forall i = 1, \ldots, m$. 

Notice, this is not the only way to seek for the OMH:

1. there is a convex hull-based formulation on p.199-200.
2. ‘noisy perceptron’ would do in simple cases as well!
3. below we consider the so-called dual problem to Eq. 2.
Optimal Margin Hyperplane in the Dual Space

It is extremely useful to consider the Lagrangian for Eq. 2.

\[ L(w, b, \alpha) = \frac{1}{2} ||w||^2 - \sum_{i=1}^{m} \alpha_i (y_i(\langle w, x_i \rangle + b) - 1), \quad \alpha_i > 0. \]  

(3)

It can be shown, the dual quadratic program:

\[ \alpha^* = \arg \max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle, \]  

(4)

\[ \text{s.t. } \sum_{i=1}^{m} \alpha_i y_i = 0, \text{ and } \alpha_i \geq 0, \quad \forall i = 1, \ldots, m. \]

allows to find OMH in terms of \( \alpha_i \):

- the dot-product \( \langle w, x \rangle = \sum_{i=1}^{m} \alpha_i y_i \langle x, x_i \rangle \),

- the bias \( b \) can be determined from the KKT optimality condition:
  \[ \alpha_i (y_i(\langle x, x_i \rangle + b) - 1) = 0. \]

The patterns \( x_i \) for which \( \alpha_i > 0 \) are called \textit{Support Vectors}. 
Nonlinear Support Vector Classifiers

Two improvements to linear classifier considered before:

1. consider nonlinear map into higher dimensional space:
   \[ \Phi : x \mapsto \phi(x), \ x \in \mathcal{H}_1, \ \phi(x) \in \mathcal{H}_2, \ \dim(\mathcal{H}_2) \gg \dim(\mathcal{H}_1). \]

2. implement it efficiently by applying the so-called kernel trick
   \[ \langle \phi(x), \phi(x_i) \rangle = k(x, x_i). \]

Notice that it is not bad to increase the dimensionality:

For \( m \) points in general position in an \( N \)-dimensional space, \( m > N + 1 \), the number of possible linear separations is

\[ 2 \sum_{i=0}^{N} \binom{m-1}{i} \] (Cover’s theorem).
Kuva 3: This is an example on how a nonlinear SVC works.
Kuva 4: SVC as a neural network. Each neuron in the hidden layer computes the kernel function between the input pattern ‘1’ and some support vector $x_i$ for which $\lambda_i = y_i \alpha_i \neq 0$. 

classification: $f(x) = \text{sgn} \left( \sum \lambda_i k(x,x_i) + b \right)$ 

weights: $f(x) = \text{sgn} ( \sum \lambda_i k(x,x_i) + b)$ 

comparison: e.g. $k(x,x_i) = (x \cdot x_i)^d$ 

$k(x,x_i) = \exp(-\|x-x_i\|^2 / c)$ 

$k(x,x_i) = \tanh(k(x \cdot x_i) + \theta)$ 

support vectors $x_1 \ldots x_4$ 

input vector $x$
Exercise

1. Download data from
   ```
   >> data
   data =
   trainvecs: [3312x13 double]
   trainlabels: 3312x1 cell
   testvecs: [60x13 double]
   testlabels: 60x1 cell
   ```

2. Apply SVC. You can use any package you like, have a look at `www.kernel-machines.org`.

3. Report the best SVC that you will obtain, i.e. type of the kernel, its parameters, $C$, does a total relative number of support vectors match the achieved test error?
Kuva 5: The data represents cepstrum vectors extracted from about 60 spoken Finnish words. Each vector corresponds to either sub-phoneme, or the beginning (end) of a word. 22 classes at your disposal. Btw, figure shows SOM that was used to get prototypes for the LVQ classifier from the data. SVC had a bit better recognition performance (80%) than the SOM-LVQ classifier.