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* $(\mathbf{x}_i, y_i) \in \mathcal{H} \times \{\pm 1\}$
- *hyperplane* $\{\mathbf{x} \in \mathcal{H} | \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\},\ \mathbf{w} \in \mathcal{H}, b \in R$.
- *decision function* $\mathbf{x} \mapsto f_{\mathbf{w},b}(\mathbf{x}) = \mathrm{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$.

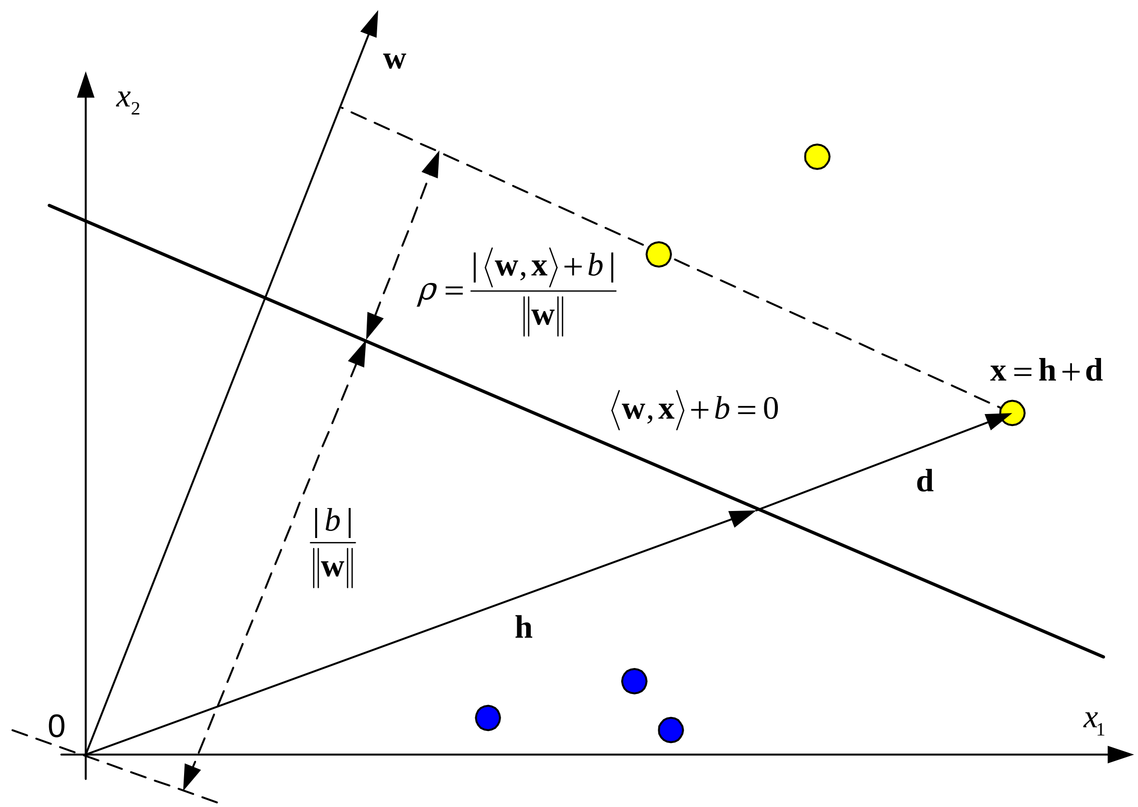
it is good to seek for the decision function $f_{\mathbf{w},b}(\mathbf{x})$ that

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

This presentation

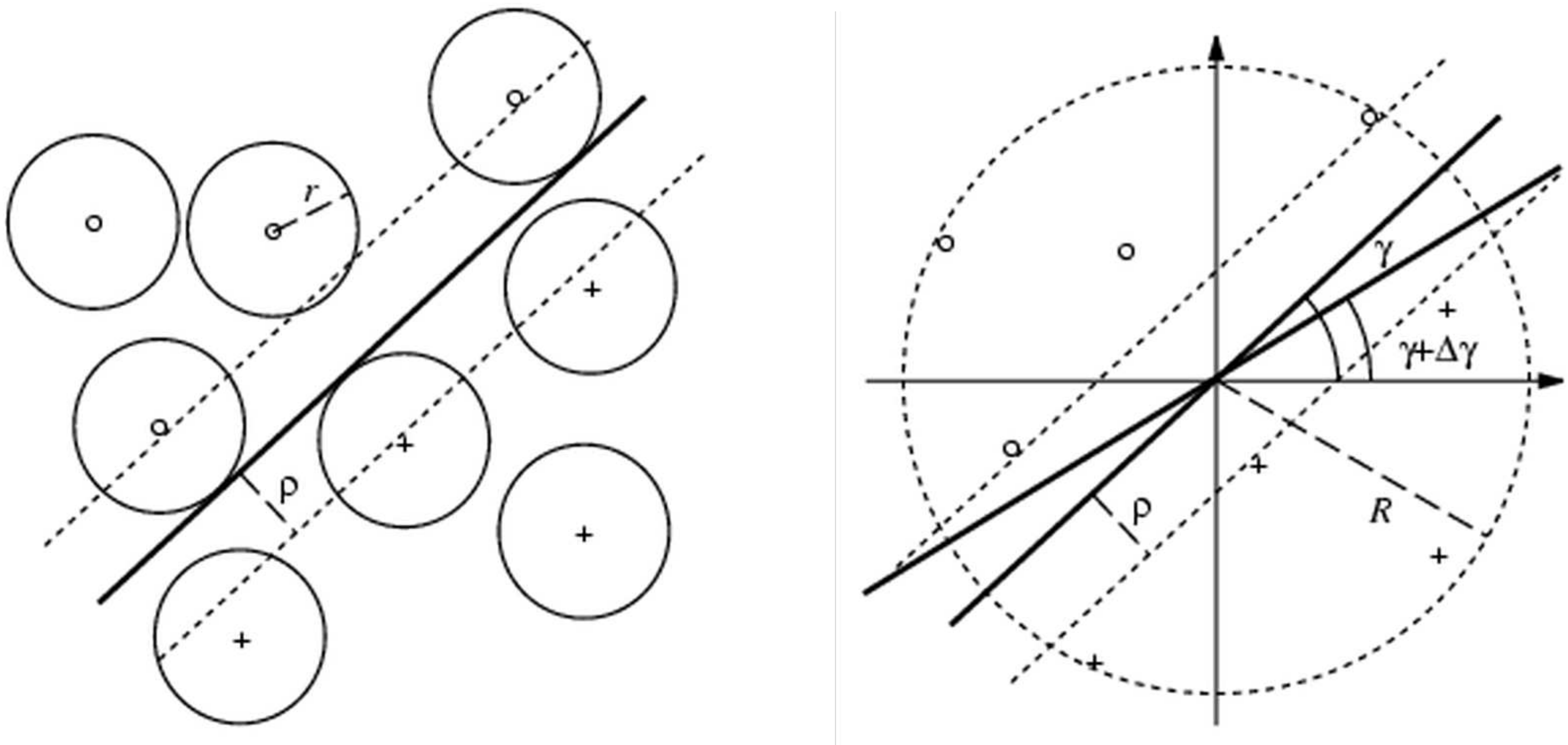
- is about what the margin is, why to maximize it, and how to do it;
- serves as an intro to the *Support Vector Classifier*.

Definition $\rho = \min_i |(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)| / ||\mathbf{w}||$ clarified



Kuva 1: $\rho = |\langle \mathbf{w}, \mathbf{x} \rangle + b| / ||\mathbf{w}|| = |\langle \mathbf{w}, \mathbf{d} \rangle| / ||\mathbf{w}|| = \text{const}/||\mathbf{w}||$.

Why to maximize ρ?



(a) (b)

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 $f(\mathbf{x}) = \text{sign}\langle \mathbf{w}, \mathbf{x} \rangle$ with $||\mathbf{w}|| \leq \Lambda$ and $\mathbf{x} \leq R$, for some $R, \Lambda > 0$. Moreover, let $\varrho > 0$, and ν denote the fraction of training examples with margin smaller than $\varrho/||\mathbf{w}||$, 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

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

c is a universal constant.

How to construct Optimal Margin Hyperplane?

Remember that margin is $\rho = \min_i |(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)|/||\mathbf{w}||$.

Notice that $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ does not change if we multiply $\mathbf{w}$ and b by some constant. One can always choose it so that $\rho = 1/||\mathbf{w}||$.

The so-called primal quadratic program will find the OMH:

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \frac{1}{2}||w||^2, \qquad (2)$$
$$\text{s.t. } y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \ \forall i = 1, \ldots, m.$$

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.

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{m} \alpha_i \big(y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1\big), \ \alpha_i > 0. \qquad (3)$$

It can be shown, the dual quadratic program:

$$\begin{aligned} \boldsymbol{\alpha}^* &= \arg\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \qquad (4) \\ & \text{s.t.} \ \sum_{i=1}^{m} \alpha_i y_i = 0, \ \text{and } \alpha_i \geq 0, \ \ \forall i = 1, \ldots, m. \end{aligned}$$

allows to find OMH in terms of α_i:

- the dot-product $\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle$,
- the bias b can be determined from the KKT optimality condition: $\alpha_i \big(y_i(\langle \mathbf{x}, \mathbf{x}_i \rangle + b) - 1\big) = 0$.

The patterns $\mathbf{x}_i$ for which $\alpha_i > 0$ are called *Support Vectors*.

Nonlinear Support Vector Classifiers

Two improvements to linear classifier considered before:

1. consider nonlinear map into higher dimensional space:

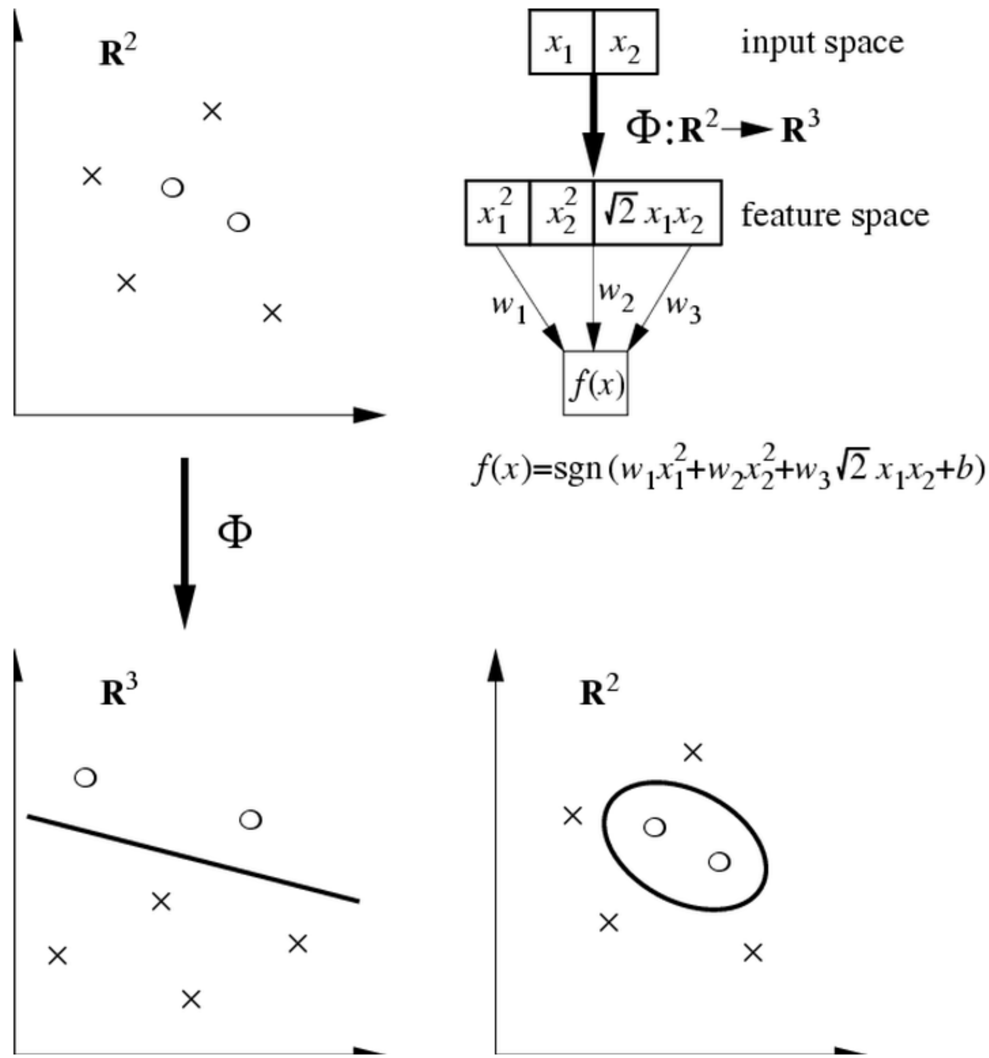
$$\Phi : \mathbf{x} \mapsto \boldsymbol{\phi}(\mathbf{x}), \;\; \mathbf{x} \in \mathcal{H}_1, \; \boldsymbol{\phi}(\mathbf{x}) \in \mathcal{H}_2, \;\; \dim(\mathcal{H}_2) \gg \dim(\mathcal{H}_1).$$

2. implement it efficiently by applying the so-called *kernel trick*

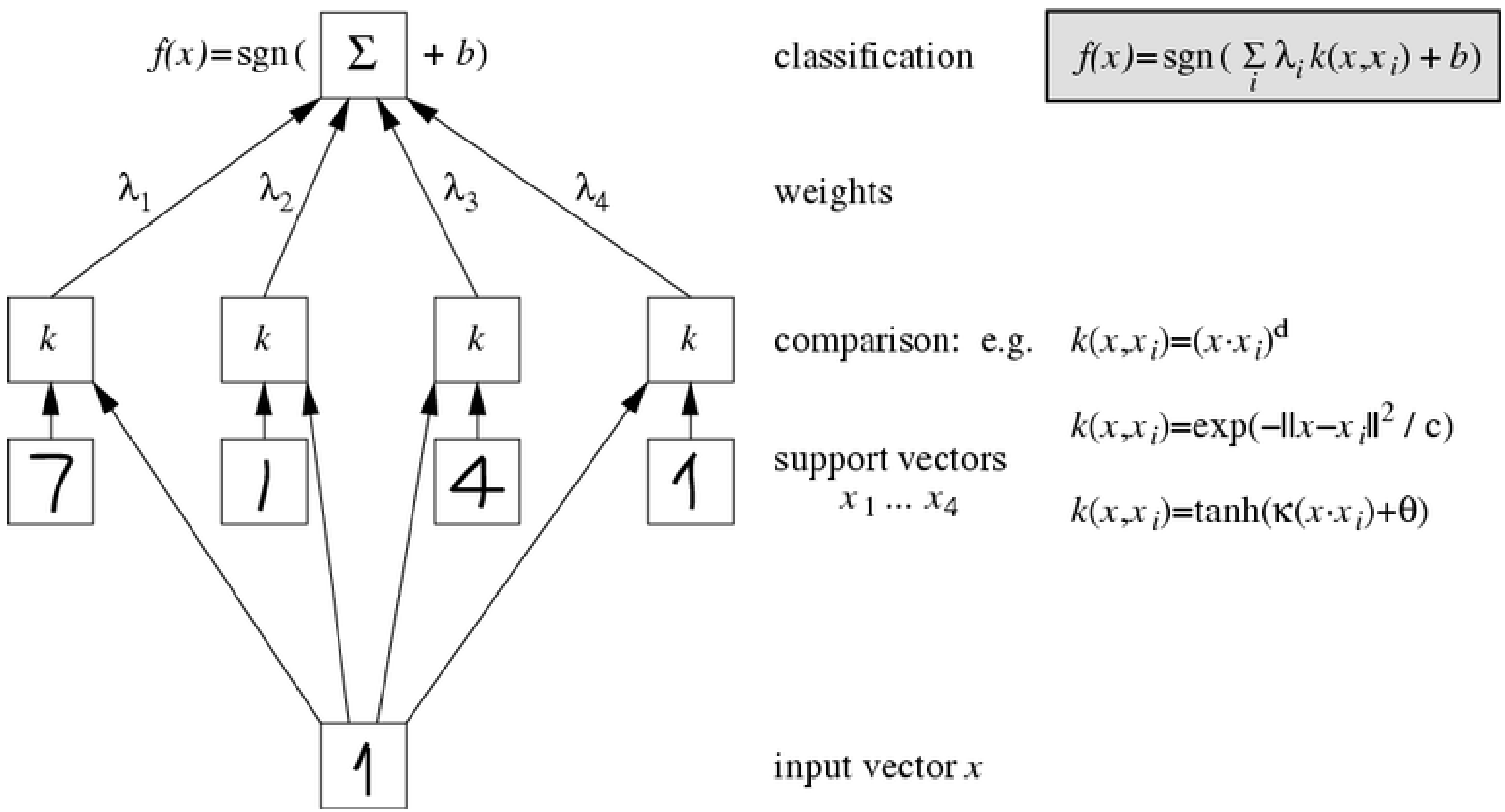
$$\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}_i) \rangle = k(\mathbf{x}, \mathbf{x}_i).$$

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 $2 \sum_{i=0}^{N} \begin{pmatrix} m-1 \\ i \end{pmatrix}$ (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 $\mathbf{x}_i$ for which $\lambda_i = y_i \alpha_i \neq 0$.

Exercise

1. Download data from *www.cis.hut.fi/ramunasg/temp/tik61183/data.mat*.

   >> data

   data =

   trainvecs: [3312x13 double]

   trainlabels: 3312x1 cell

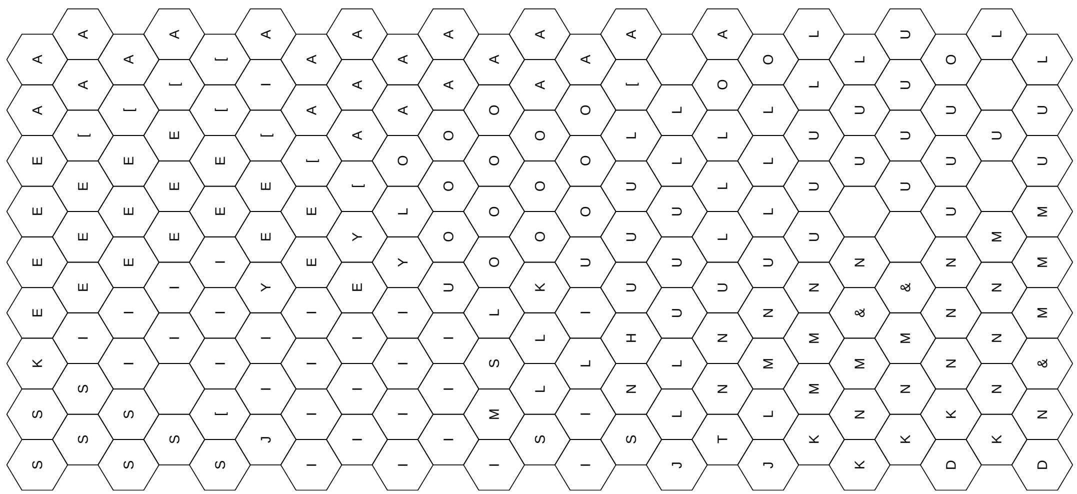
   testvecs: [60x13 double]

   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?

Additional info (to get an even number of slides)



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.