

Optimal separating hyperplane. Basis expansion.  
Kernel trick. Support vector machine.

Petr Pošík

<b>Rehearsal</b>	<b>2</b>
Linear DF .....	3
<b>Optimal separating hyperplane</b>	<b>4</b>
Optimal SH .....	5
Margin size .....	6
OSH learning .....	7
OSH: remarks .....	8
Demo .....	9
<b>When a linear decision boundary is not enough...</b>	<b>10</b>
Basis expansion .....	11
Two spaces .....	12
Remarks .....	14
<b>Support vector machine</b>	<b>15</b>
OSH + basis exp. ....	16
Kernel trick .....	17
SVM .....	18
Linear SVM .....	19
Gaussian SVM .....	20

**Linear discrimination function**

**Binary classification** of objects  $x$  (classification into 2 classes, dichotomy):

- For 2 classes, 1 discrimination function is enough.
- Decision rule:

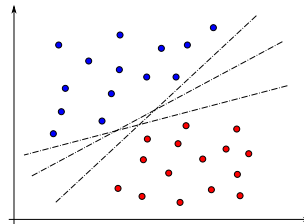
$$\left. \begin{aligned} f(x^{(i)}) > 0 &\iff \hat{y}^{(i)} = +1 \\ f(x^{(i)}) < 0 &\iff \hat{y}^{(i)} = -1 \end{aligned} \right\} \text{ i.e. } \hat{y}^{(i)} = \text{sign} \left( f(x^{(i)}) \right)$$

Learning of the linear discrimination function by the *perceptron algorithm*:

- Optimization of

$$J(w, T) = \sum_{i=1}^{|T|} \mathbb{I} \left( y^{(i)} \neq \hat{y}^{(i)} \right)$$

- The weight vector is a weighted sum of the training points  $x^{(i)}$ .
- Perceptron finds *any* separating hyperplane, if exists.
- Among the infinite number of separating hyperplanes, which one is the best?



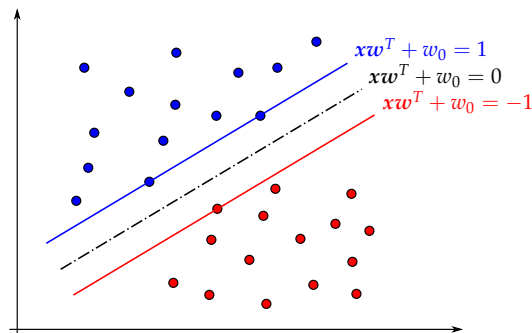
**Optimal separating hyperplane**

**Optimal separating hyperplane**

**Margin (cz:odstup):**

- “The width of the band in which the decision boundary can move (in the direction of its normal vector) without touching any data point.”

**Maximum margin linear classifier**



Plus 1 level:  $\{x : xw^T + w_0 = 1\}$   
 Minus 1 level:  $\{x : xw^T + w_0 = -1\}$   
 Decision boundary:  $\{x : xw^T + w_0 = 0\}$

**Support vectors:**

- Data points  $x$  lying at the plus 1 level or minus 1 level.
- Only these points influence the decision boundary!

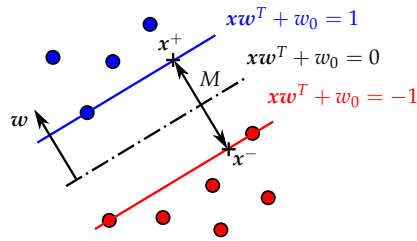
Why we would like to maximize the margin?

- Intuitively, it is safe.
- If we make a small error in estimating the boundary, the classification will likely stay correct.
- The model is invariant with respect to the training set changes, except the changes of support vectors.
- There are sound theoretical results (based on VC dimension) that having a maximum margin classifier is good.
- Maximal margin works well in practice.

## Margin size

How to compute the margin  $M$  given  $w = (w_1, \dots, w_D), w_0$ ?

- Let's choose two points  $x^+$  and  $x^-$ , lying in the plus 1 level and minus 1 level, respectively.
- Let's compute the margin  $M$  as their distance.



We know that:

$$\begin{aligned}x^+ w^T + w_0 &= 1 \\x^- w^T + w_0 &= -1 \\x^- + \lambda w &= x^+\end{aligned}$$

Thus the margin size is

$$M = \|x^+ - x^-\| = \|\lambda w\| = \lambda \|w\| = \frac{2}{\|w\|^2} \|w\| = \frac{2}{\|w\|}$$

And we can derive:

$$\begin{aligned}(x^+ - x^-) w^T &= 2 \\(x^- + \lambda w - x^-) w^T &= 2 \\ \lambda w w^T &= 2 \\ \lambda = \frac{2}{w w^T} &= \frac{2}{\|w\|^2}\end{aligned}$$

## Optimal separating hyperplane learning

We want to maximize margin  $M = \frac{2}{\|w\|}$  subject to the constraints ensuring correct classification of the training set  $T$ . This optimization problem can be formulated as a *quadratic programming* (QP) task.

- Primary QP task:

$$\begin{aligned}\text{minimize } & w w^T \text{ with respect to } w_1, \dots, w_D \\ \text{subject to } & y^{(i)} (x^{(i)} w^T + w_0) \geq 1.\end{aligned}$$

- Dual QP task:

$$\begin{aligned}\text{maximize } & \sum_{i=1}^{|T|} \alpha_i - \frac{1}{2} \sum_{i=1}^{|T|} \sum_{j=1}^{|T|} \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)} x^{(j)T} \text{ with respect to } \alpha_1, \dots, \alpha_{|T|} \\ \text{subject to } & \alpha_i \geq 0 \\ \text{and } & \sum_{i=1}^{|T|} \alpha_i y^{(i)} = 0.\end{aligned}$$

- From the solution of the dual task, we can compute the solution of the primal task:

$$w = \sum_{i=1}^{|T|} \alpha_i y^{(i)} x^{(i)}, \quad w_0 = y^{(k)} - x^{(k)} w^T,$$

where  $(x^{(k)}, y^{(k)})$  is any *support vector*, i.e.  $\alpha_k > 0$ .

## Optimal separating hyperplane: concluding remarks

### The importance of dual formulation:

- The QP task in dual formulation is easier to solve for QP solvers than the primal formulation.
- New, unseen examples can be classified using function

$$f(x, w, w_0) = \text{sign}(xw^T + w_0) = \text{sign}\left(\sum_{i=1}^{|T|} \alpha_i y^{(i)} x^{(i)} x^T + w_0\right),$$

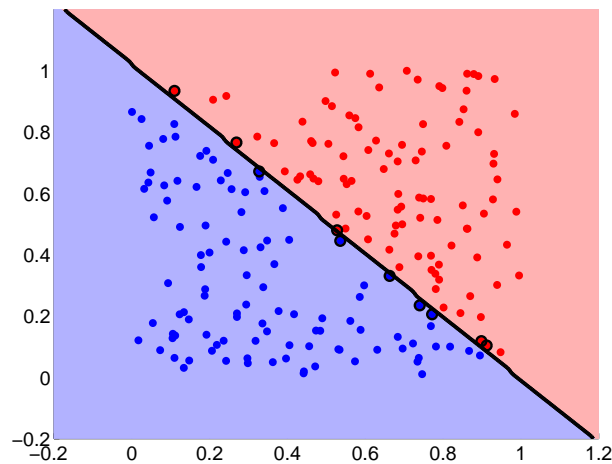
i.e. the discrimination function contains the examples  $x$  only in the form of dot products (which will be useful later).

- The examples with  $\alpha_i > 0$  are *support vectors*, thus the sums may be carried out only over the support vectors.
- The dual formulation allows for other tricks which you will learn later.

### What if the data are not linearly separable?

- There is a generalization of the QP task formulation for this case (*soft margin*).
- The primal task has double the number of constraints, the task is more complex.
- The results for the QP task with *soft margin* are of the same type as before.

## Optimal separating hyperplane: demo



**Basis expansion**

a.k.a. **feature space straightening**.

**Why?**

- Linear decision boundary (or linear regression model) may not be flexible enough to perform precise classification (regression).
- The algorithms for fitting linear models can be used to fit *non-linear models!*

**How?**

- Let's define a new multidimensional image space  $F$ .
- The examples are then transformed into this image space (new features are derived):

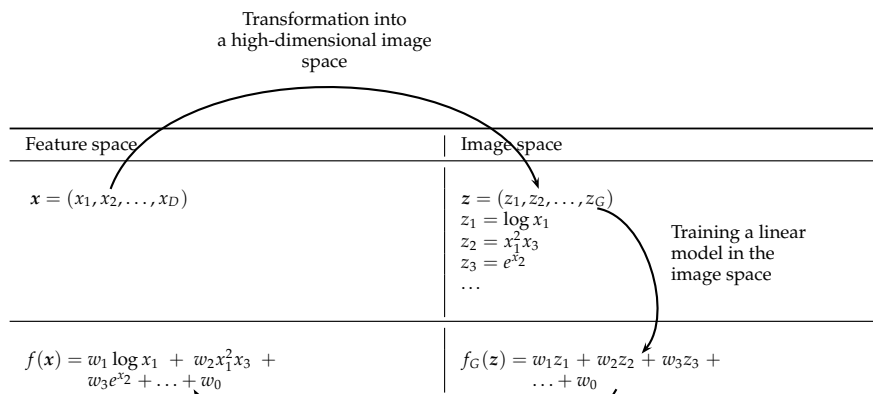
$$\begin{aligned} \mathbf{x} &\rightarrow \mathbf{z} = \Phi(\mathbf{x}), \\ \mathbf{x} = (x_1, x_2, \dots, x_D) &\rightarrow \mathbf{z} = (\Phi_1(\mathbf{x}), \Phi_2(\mathbf{x}), \dots, \Phi_G(\mathbf{x})), \end{aligned}$$

while usually  $D \ll G$ .

- In the image space, a linear model is trained. However, this is equivalent to training a non-linear model in the original space.

$$\begin{aligned} f_G(\mathbf{z}) &= w_1 z_1 + w_2 z_2 + \dots + w_G z_G + w_0 \\ f(\mathbf{x}) = f_G(\Phi(\mathbf{x})) &= w_1 \Phi_1(\mathbf{x}) + w_2 \Phi_2(\mathbf{x}) + \dots + w_G \Phi_G(\mathbf{x}) + w_0 \end{aligned}$$

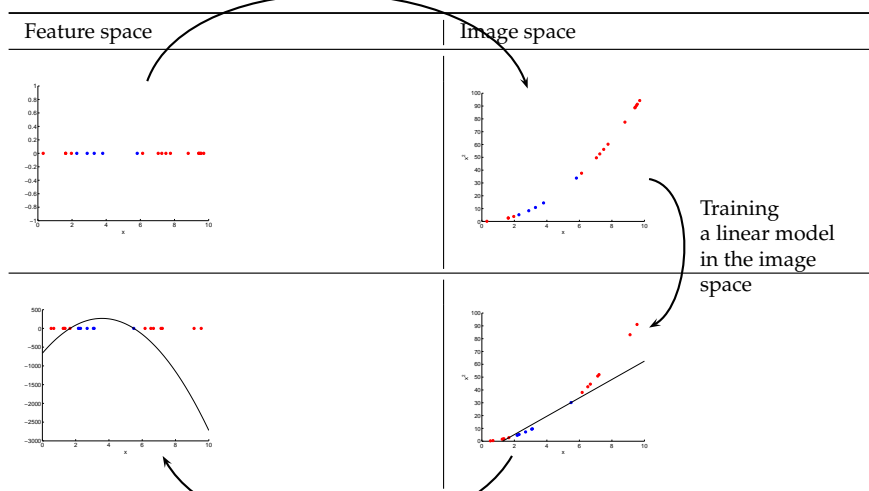
**Two coordinate systems**



Non-linear model in the feature space

## Two coordinate systems: graphically

transformation into  
a high-dimensional  
image space



Training  
a linear model  
in the image  
space

P. Pošík © 2014

Artificial Intelligence – 13 / 20

Non-linear model in the  
feature space

## Basis expansion: remarks

Advantages:

- Universal, generally usable method.

Disadvantages:

- We must define what new features shall form the high-dimensional space  $F$ .
- The examples must be really transformed into the high-dimensional space  $F$ .

For certain type of algorithms, there is a method how to perform the basis expansion without actually carrying out the mapping!

P. Pošík © 2014

Artificial Intelligence – 14 / 20

**Optimal separating hyperplane combined with the basis expansion**

To reiterate: when using the optimal separating hyperplane, the examples  $x$  occur only in

$$\text{the optimization criterion } \sum_{i=1}^{|T|} \alpha_i - \frac{1}{2} \sum_{i=1}^{|T|} \sum_{j=1}^{|T|} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)} \mathbf{x}^{(j)T}$$

$$\text{and in the decision rule } f(x) = \text{sign} \left( \sum_{i=1}^{|T|} \alpha_i y^{(i)} \mathbf{x}^{(i)} \mathbf{x}^T + w_0 \right).$$

Application of the basis expansion changes

$$\text{the optimization criterion to } \sum_{i=1}^{|T|} \alpha_i - \frac{1}{2} \sum_{i=1}^{|T|} \sum_{j=1}^{|T|} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(\mathbf{x}^{(i)}) \Phi(\mathbf{x}^{(j)})^T$$

$$\text{and the decision rule to } f(x) = \text{sign} \left( \sum_{i=1}^{|T|} \alpha_i y^{(i)} \Phi(\mathbf{x}^{(i)}) \Phi(\mathbf{x})^T + w_0 \right).$$

What if we use a scalar function  $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$  instead of the dot product in the image space?

$$\text{The optimization criterion: } \sum_{i=1}^{|T|} \alpha_i - \frac{1}{2} \sum_{i=1}^{|T|} \sum_{j=1}^{|T|} \alpha_i \alpha_j y^{(i)} y^{(j)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

$$\text{The discrimination function: } f(x) = \text{sign} \left( \sum_{i=1}^{|T|} \alpha_i y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x}) + w_0 \right).$$

**Kernel trick**

There are function of 2 vector arguments  $K(\mathbf{a}, \mathbf{b})$  which provide values equal to the dot product  $\Phi(\mathbf{a})\Phi(\mathbf{b})^T$  of the images of the vectors  $\mathbf{a}$  and  $\mathbf{b}$  in certain high-dimensional image space. Such functions are called **kernel functions** or **kernels**.

**Kernel trick:** Let's have a linear algorithm in which the examples  $x$  occur only in dot products.

- Such an algorithm can be made non-linear by replacing the dot products of examples  $x$  with *kernels*.
- The result is the same as if the algorithm was trained in some high-dimensional image space with the coordinates given by many non-linear basis functions.
- Thanks to kernels, it is not needed to perform the mapping, the algorithm is much more efficient.

Frequently used kernels:

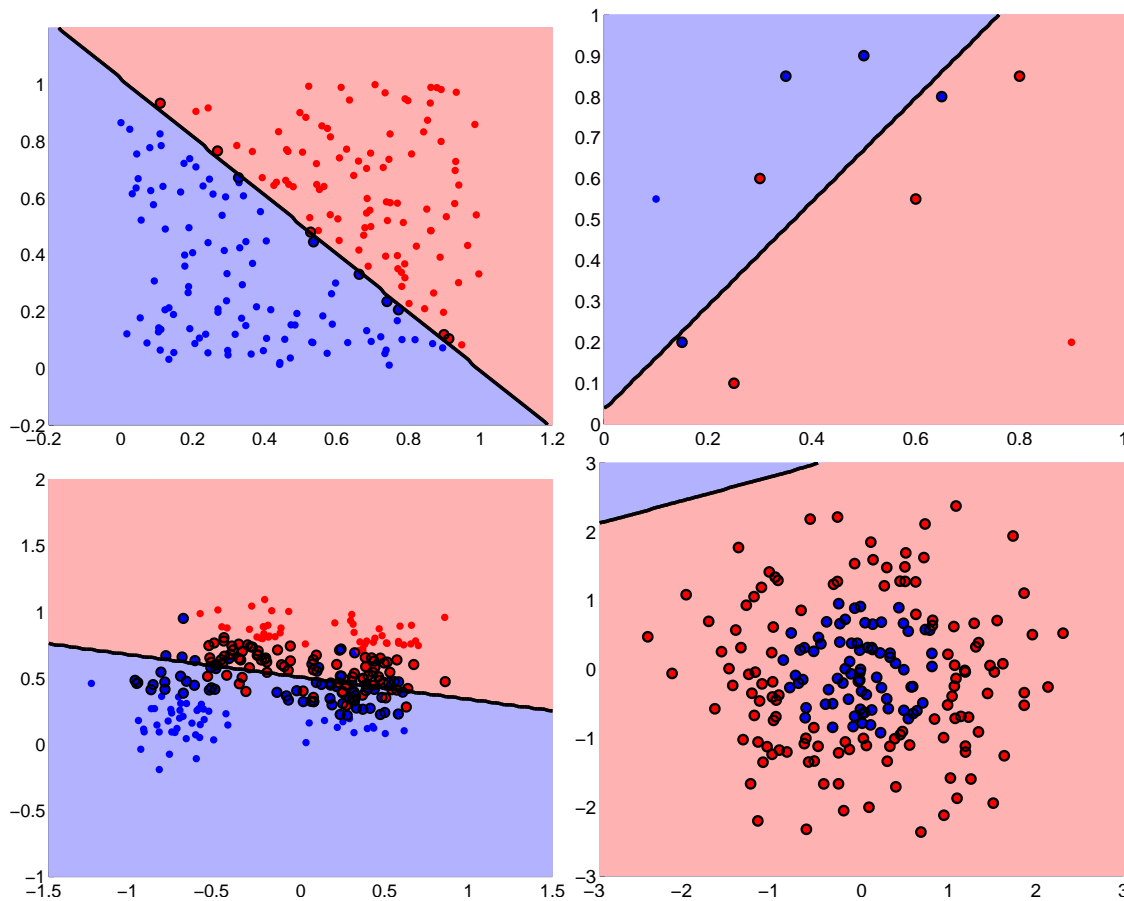
Polynomial:  $K(\mathbf{a}, \mathbf{b}) = (\mathbf{a}\mathbf{b}^T + 1)^d$ , where  $d$  is the degree of the polynomial.

Gaussian (RBF):  $K(\mathbf{a}, \mathbf{b}) = \exp\left(-\frac{|\mathbf{a} - \mathbf{b}|^2}{\sigma^2}\right)$ , where  $\sigma^2$  is the „width“ of Gaussian.

# Support vector machine

Support vector machine (SVM)  
=  
optimal separating hyperplane  
+  
kernel trick

## Demo: SVM with linear kernel





Demo: SVM with Gaussian (RBF) kernel

