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

Petr Pošík

Linear discrimination function

Binary classification of objects $\boldsymbol{x}$ (classification into 2 classes, dichotomy):

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

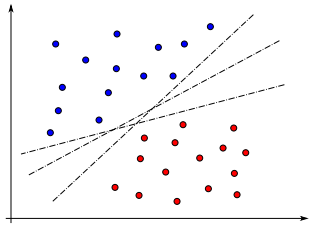
$$\left.\begin{array}{l} f(\boldsymbol{x}^{(i)}) > 0 \Longleftrightarrow \widehat{y}^{(i)} = +1 \\ f(\boldsymbol{x}^{(i)}) < 0 \Longleftrightarrow \widehat{y}^{(i)} = -1 \end{array}\right\} \quad \text{i.e.} \quad \widehat{y}^{(i)} = \operatorname{sign}\left(f(\boldsymbol{x}^{(i)})\right)$$

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

- Optimization of

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

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

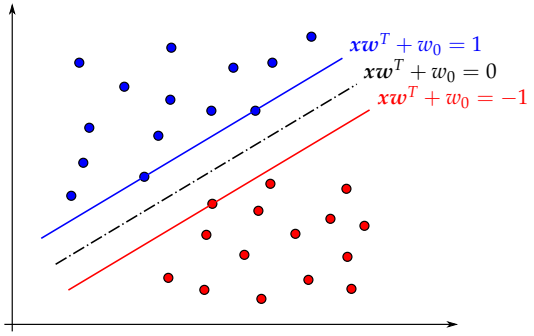


Optimal separating hyperplane

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

Support vectors:

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

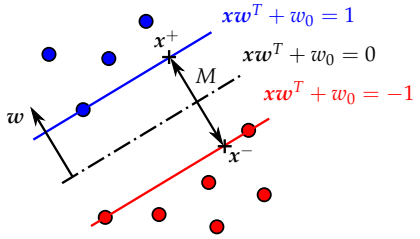
Why we would like to maximize the margin?

- 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 $\boldsymbol{w} = (w_1, \ldots, w_D), w_0$?

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



We know that:

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

And we can derive:

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

Thus the margin size is

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

Optimal separating hyperplane learning

We want to maximize margin $M = \frac{2}{\|\boldsymbol{w}\|}$ 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:

  minimize $\boldsymbol{w}\boldsymbol{w}^T$ with respect to $w_1, \ldots, w_D$

  subject to $y^{(i)}(\boldsymbol{x}^{(i)}\boldsymbol{w}^T + w_0) \geq 1$.

- Dual QP task:

  maximize $\sum_{i=1}^{|T|}\alpha_i - \frac{1}{2}\sum_{i=1}^{|T|}\sum_{j=1}^{|T|}\alpha_i\alpha_j y^{(i)}y^{(j)}\boldsymbol{x}^{(i)}{\boldsymbol{x}^{(j)}}^T$ with respect to $\alpha_1, \ldots, \alpha_{|T|}$

  subject to $\alpha_i \geq 0$

  and $\sum_{i=1}^{|T|}\alpha_i y^{(i)} = 0$.

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

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

  where $(\boldsymbol{x}^{(k)}, y^{(k)})$ 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

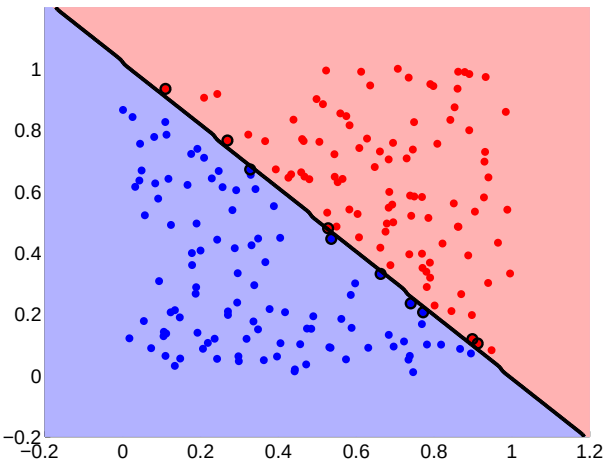
$$f(\boldsymbol{x}, \boldsymbol{w}, w_0) = \text{sign}(\boldsymbol{x}\boldsymbol{w}^T + w_0) = \text{sign}\left(\sum_{i=1}^{|T|} \alpha_i y^{(i)} \boldsymbol{x}^{(i)} \boldsymbol{x}^T + w_0\right),$$

  i.e. the discrimination function contains the examples $\boldsymbol{x}$ 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?

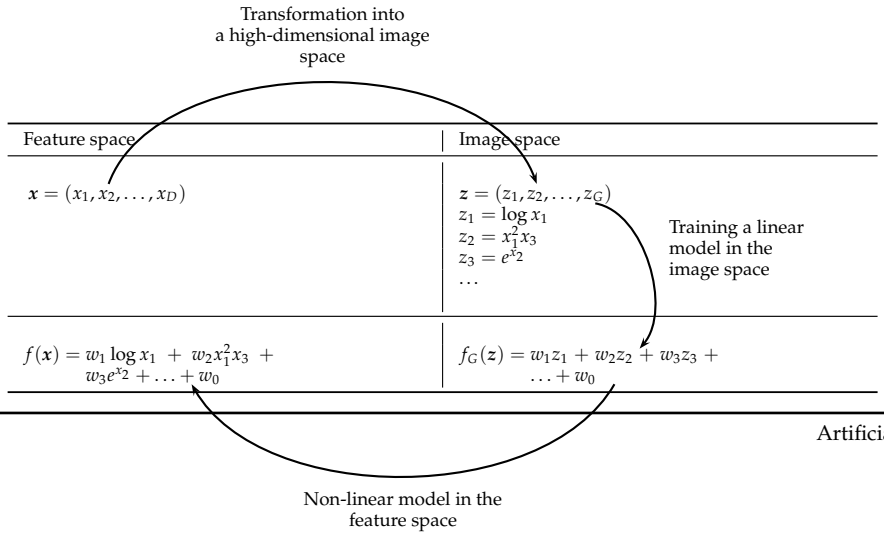
- Let's define a new multidimensional image space F.
- The examples are then tranformed into this image space (new features are derived):

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

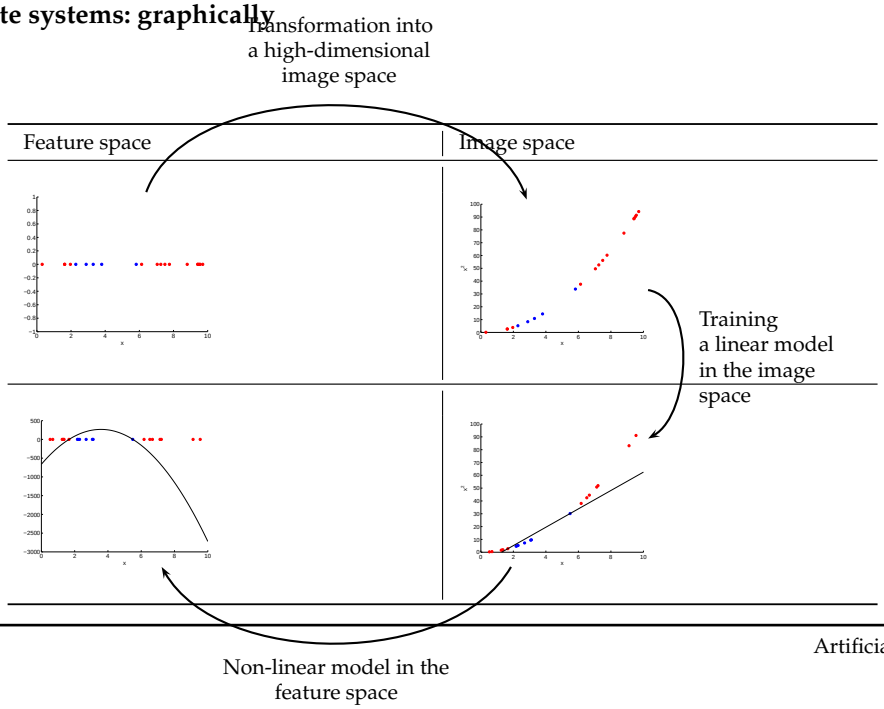
  while usually $D \ll G$.
- In the image space, a linear model is trained. However, this is equivalent to training a non-linear model in the original space.

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

Two coordinate systems



Two coordinate systems: graphically



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!

Optimal separating hyperplane combined with the basis expansion

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

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

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

Application of the basis expansion changes

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

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

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

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

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

Kernel trick

There are function of 2 vector arguments $K(\boldsymbol{a}, \boldsymbol{b})$ which provide values equal to the dot product $\Phi(\boldsymbol{a})\Phi(\boldsymbol{b})^T$ of the images of the vectors a and 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 $\boldsymbol{x}$ occur only in dot products.

- Such an algorithm can be made non-linear by replacing the dot products of examples $\boldsymbol{x}$ with *kernels*.
- The result is the same is 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: $K(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}\boldsymbol{b}^T + 1)^d$, where d is the degree of the polynom.

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

Support vector machine

Support vector machine (SVM)

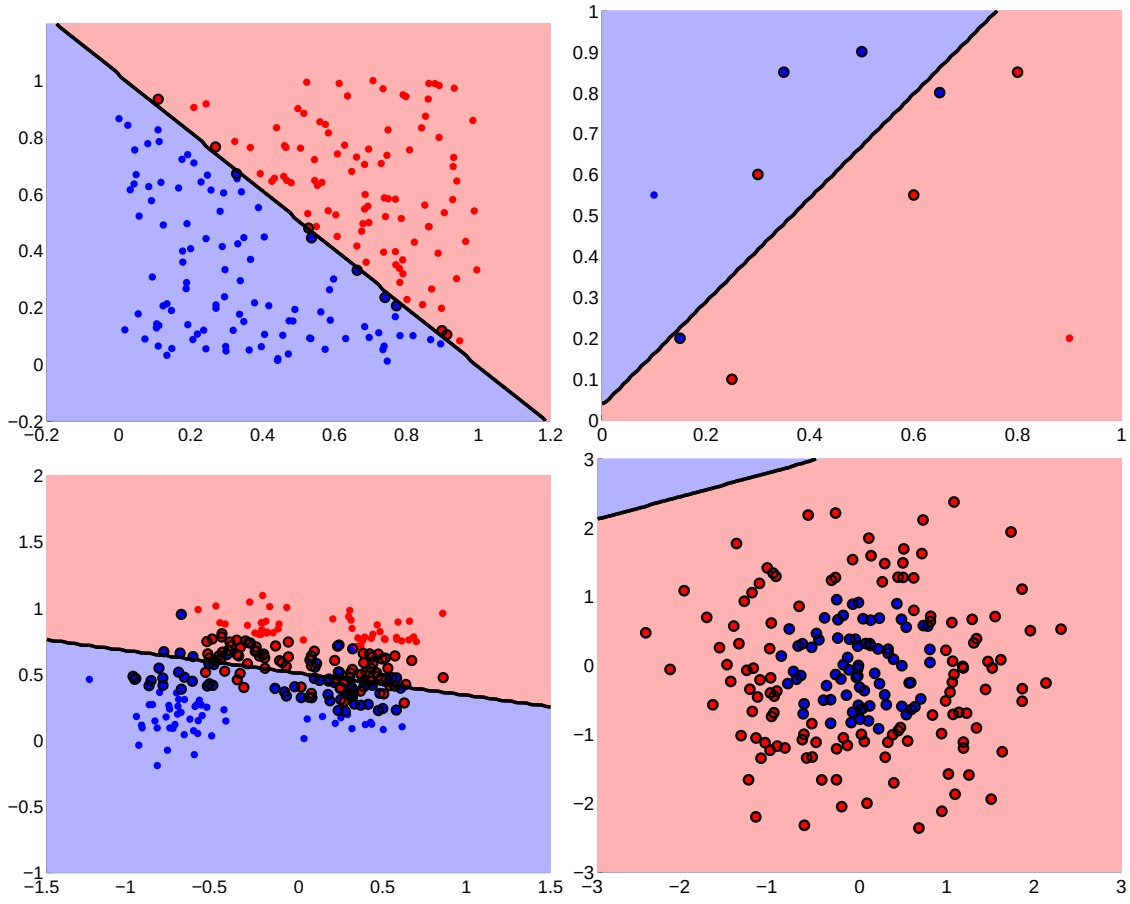
=

optimal separating hyperplane

+

kernel trick

Demo: SVM with linear kernel



Demo: SVM with Gaussian (RBF) kernel