Support Vector Machines: Non-linear classification

Reminder: The dual problem for a linearly separable classification problem with two classes C_1 and C_2:

Patterns: $\mathbf{x}_i, \quad i = 1, 2, \ldots P$

Target outpts: $t_i, \quad i = 1, 2, \ldots P, \qquad t_i = 1 \quad \forall \mathbf{x}_i \in C_1, \quad t_i = -1 \quad \forall \mathbf{x}_i \in C_2$

Dual problem: Training task

Maximize with respect to λ_i of:

$$L_D = \sum_{i=1}^{P} \lambda_i - \tfrac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{P} \lambda_i \lambda_j t_i t_j Q_{ij} \qquad Q_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$$

under the constraints:

$$\sum_{i=1}^{P} \lambda_i t_i = 0 \qquad \lambda_i \geq 0, \quad i = 1, 2, \ldots, P$$

Dual problem: Classification task

For every new pattern $\mathbf{x}_{new}$:

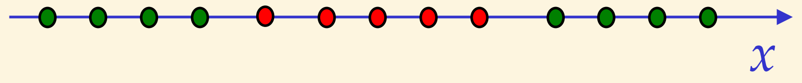
$$\sum_i \lambda_i t_i (\mathbf{x}_{new} \cdot \mathbf{x}_i) + w_0 > 0 \Rightarrow \mathbf{x}_{new} \in C_1$$

$$\sum_i \lambda_i t_i (\mathbf{x}_{new} \cdot \mathbf{x}_i) + w_0 < 0 \Rightarrow \mathbf{x}_{new} \in C_2$$

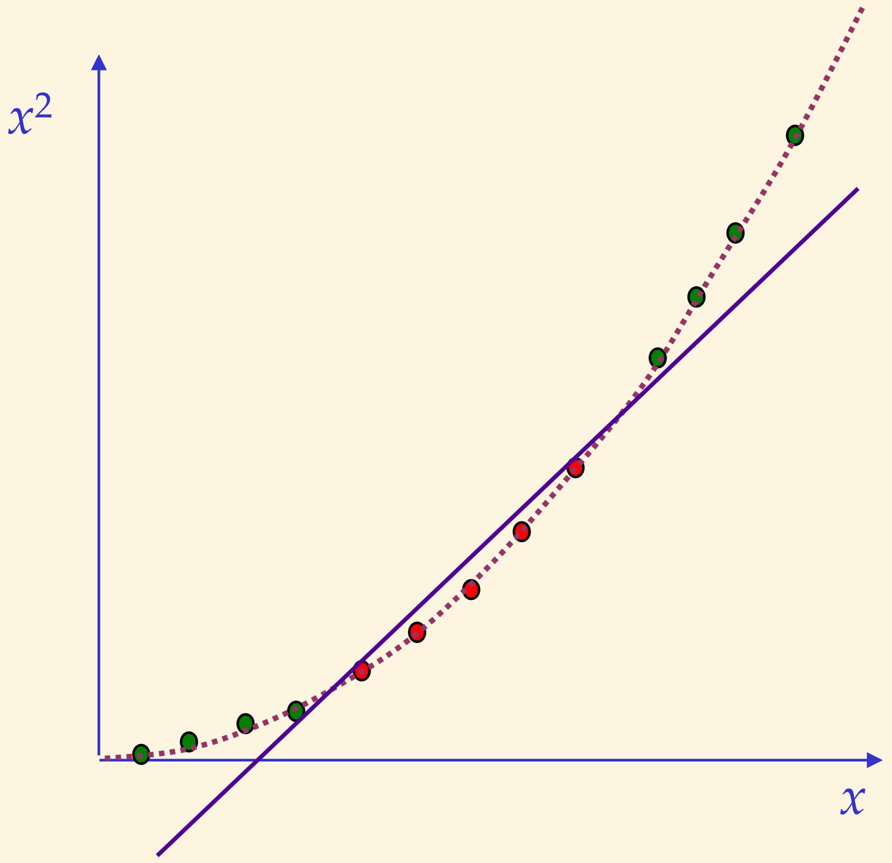
Increasing the dimensionality

- Now we suppose that our problem is not linearly separable. The first observation we can make is that it is generally possible to transform a non-linearly separable problem to a linearly separable one if we somehow increase the dimensionality of the problem by mapping the space of input vectors to a higher dimensional space. First, let us consider a simple example.

A simple problem



2 classes, as shown in the diagram. Obviously, the classes are non-lineraly separable. Our linear SVM cannot cope with this problem.



Idea: Let us transform the input data, in order to render the problem lineary separable. We enhance the patterns, introducing a second feature equal to the square of the original feature. We then classify the enhanced patterns easily using an SVM in the plane (x,x^2).

The discriminating straight line is of the form:

$$w_1 x + w_2 x^2 + w_0 = 0$$

We had to use one more weight than before. The new patterns „live” in a two-dimensional space. Still, they belong to a one-dimensional curve (manifold) embedded into this space.

Classification in one dimensional problems (1)

For problems of higher complexity, we may want to use polynomials of higher degree *D*, so that the equation for the separating surface becomes:

$$\sum_{k=1}^{D} w_k x^k + w_0 = 0$$

In short, we increase the number of dimensions and attempt to classify patterns of the form:

$$\mathbf{\Phi}(x) = (1, x, x^2, \ldots, x^D)$$

Naturally, the number of weights increases with *D* (linearly in one dimension, but we will soon discover that the complexity increases much more rapidly in more dimensions).

Possible solution: Remember that in the dual formulation the number of parameters is equal to the number of patterns, not to the number of dimensions. So if we retort to the dual formulation, the problem is non-existent! No extra parameters are needed.

Classification in one dimensional problems (2)

With our extra dimensions coming into play, **Q** becomes:

$$Q_{ij} = \sum_{k=0}^{D} x_i^k x_j^k$$

Evidently, the complexity of calculating **Q** has increased.

What have we achieved using the dual formulation in conjunction with polynomials of degree *D*?

- We have kept the number of weights under control: We have the same number of free parameters as in the case of the linear SVM.
- However: There is additional complexity in the calculation of the matrix **Q.**

Classification in one dimensional problems (3)

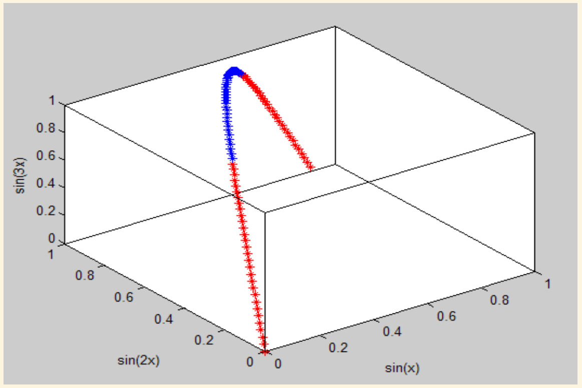
Imagine that a deus ex machina appears in our sleep and promises to take care of the computational burden imposed by the extra complexity.

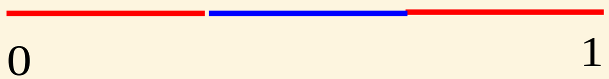
Reassured, we cheer up and we attempt to add features depending on more complicated functions of the patterns.

For example, we can try extra features originating from a Fourier expansion:

$$\mathbf{\Phi}(x) = [\tfrac{1}{\sqrt{2}}, \quad \sin(kx), \quad \cos(kx)], \quad k \in \mathbb{N}, 1 \le k \le D$$

$$Q_{ij} = \sum_{k=0}^{D} [\sin(kx_i)\sin(kx_j) + \cos(kx_i)\cos(kx_j)]$$





And, having been offered free lunch, we can try more extreme things. For example, let us consider an infinity of features originating from a continuous Fourier transformation:

$$\mathbf{\Phi}(x) = [\sin(rx), \quad \cos(rx)], \quad r \in \mathbb{R}, \quad 0 \le r \le D$$

$$Q_{ij} = \int_{0}^{D} [\sin(rx_i)\sin(rx_j) + \cos(rx_i)\cos(rx_j)]\,dr$$

Deus ex machina: The kernel trick (1)

$$Q_{ij} = \sum_{k=0}^{D} x_i^{\,k} x_j^{\,k} = \frac{1-(x_i x_j)^{D+1}}{1-x_i x_j}$$

$$Q_{ij} = \frac{1}{2} + \sum_{k=1}^{D} [\sin(kx_i)\sin(kx_j) + \cos(kx_i)\cos(kx_j)]$$

$$= \frac{1}{2} + \sum_{k=1}^{D} \cos[k(x_i - x_j)] = \frac{\sin[(D+\frac{1}{2})(x_i - x_j)]}{2\sin[(x_i - x_j)/2]}$$

$$Q_{ij} = \int_0^D [\sin(rx_i)\sin(rx_j) + \cos(rx_i)\cos(rx_j)]\,dr = \int_0^D \cos[r(x_i - x_j)]\,dr = \frac{\sin[D(x_i - x_j)]}{x_i - x_j}$$

The crucial observation is that in all three cases there exists a function $K(x, y)$, so that:

$$\mathbf{\Phi}(x)\cdot\mathbf{\Phi}(y) = K(x, y)$$

***K*(.) is called a KERNEL FUNCTION. *Every element of the matrix* Q is evaluated in closed form. There is no need to evaluate the sums, or the infinite series, or the integrals. The complexity of evaluating Q is O(*P*^{2}) independently of the dimension of the enhanced pattern space.**

Deus ex machina: The kernel trick (2)

The benefit obtained in the classification phase is easily recognized: The conditions for classifying a new pattern:

$$\sum_i \lambda_i t_i [\mathbf{\Phi}(x_{new}) \cdot \mathbf{\Phi}(x_i)] + w_0 > 0 \Rightarrow x_{new} \in C_1$$

$$\sum_i \lambda_i t_i [\mathbf{\Phi}(x_{new}) \cdot \mathbf{\Phi}(x_i)] + w_0 < 0 \Rightarrow x_{new} \in C_2$$

can be written as:

$$\sum_i \lambda_i t_i K(x_{new}, x_i) + w_0 > 0 \Rightarrow x_{new} \in C_1$$

$$\sum_i \lambda_i t_i K(x_{new}, x_i) + w_0 < 0 \Rightarrow x_{new} \in C_2$$

Therefore, there is no need to evaluate the inner products. Once again, finding the class to which a pattern belongs does not depend on the dimension of the enhanced pattern space.

Knowledge of the kernel is sufficient for both training and classification. Knowledge of the functions Φ is not required. The kernel is all that is required.

The kernel trick: Multi dimensional patterns (1)

Generalization to patterns of more than one dimension is straightforward: For a pattern of N dimensions, using polynomials up to second degree, the enhanced patterns and the inner products that we use for classification are as follows:

$$\mathbf{\Phi}(\mathbf{x}) = [1, \sqrt{2}x_1, \ldots, \sqrt{2}x_N, x_1{}^2, \ldots, x_N{}^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_{N-1}x_N]$$

$$\mathbf{\Phi}(\mathbf{y}) = [1, \sqrt{2}y_1, \ldots, \sqrt{2}y_N, y_1{}^2, \ldots, y_N{}^2, \sqrt{2}y_1y_2, \ldots, \sqrt{2}y_{N-1}y_N]$$

$\boxed{O(N^2) \quad terms}$

$$\mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{y}) = 1 + 2\sum_{1}^{N} x_i y_i + \sum_{1}^{N} x_i{}^2 y_i{}^2 + 2\sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j y_i y_j$$

$$(\mathbf{x}\cdot\mathbf{y}+1)^2 = 1 + 2\mathbf{x}\cdot\mathbf{y} + (\mathbf{x}\cdot\mathbf{y})^2 = 1 + 2\sum_{1}^{N} x_i y_i + (\sum_{1}^{N} x_i y_i)^2$$

$$= 1 + 2\sum_{1}^{N} x_i y_i + \sum_{1}^{N} x_i{}^2 y_i{}^2 + 2\sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j y_i y_j$$

$$\mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y}+1)^2$$

$\boxed{just \quad O(N) \quad distinct \quad terms}$

Again, there is no need to calculate the sum of the O(N^2) terms to evaluate the inner product.

The kernel trick: Multi dimensional patterns (2)

Polynomial degree	# terms $\mathbf{\Phi}(\mathbf{x})$	**Q**: Computational cost without trick	**Q**: Computational cost with trick	Kernel	Example: 100 inputs	
					Cost without trick	Cost with trick
2	$N^2/2$	$N^2 P^2/4$	$N P^2/2$	$(\mathbf{x}\cdot\mathbf{y}+1)^2$	*2500 P^2*	*50 P^2*
3	$N^3/6$	$N^3 P^2/12$	$N P^2/2$	$(\mathbf{x}\cdot\mathbf{y}+1)^3$	*83000 P^2*	*50 P^2*
4	$N^4/12$	$N^4 P^2/48$	$N P^2/2$	$(\mathbf{x}\cdot\mathbf{y}+1)^4$	*1960000 P^2*	*50 P^2*

Enhancement of the original space using polynomials. Analysis of the computational cost for the matrix **Q** as a function of the number of inputs N and the number of patterns P. The computational burden does not increase with the degree of the polynomials.

The kernel trick: Multi dimensional patterns (3)

In every case, the requirement is that a function $K(\mathbf{x},\mathbf{y})$ exists, so that:

$$\mathbf{\Phi}(\mathbf{x})\cdot\mathbf{\Phi}(\mathbf{y}) = K(\mathbf{x},\mathbf{y})$$

A simplification of the dual Lagrangian results:

$$L_D = \sum_{i=1}^{P}\lambda_i - \tfrac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P}\lambda_i\lambda_j t_i t_j Q_{ij} \qquad Q_{ij} = K(\mathbf{x}_i,\mathbf{x}_j)$$

Also, the rule for the classification of new patterns in the testing face is simplified:

$$\sum_i \lambda_i t_i K(\mathbf{x}_{new},\mathbf{x}_i) + w_0 > 0 \Rightarrow \mathbf{x}_{new} \in C_1$$

$$\sum_i \lambda_i t_i K(\mathbf{x}_{new},\mathbf{x}_i) + w_0 < 0 \Rightarrow \mathbf{x}_{new} \in C_2$$

Knowledge of the kernel is sufficient for both training and classification. Knowledge of the functions Φ is not required. The kernel is all that is required.

Mercer's condition (1)

Question number 1: Under which conditions is a function $K(\mathbf{x},\mathbf{y})$ a kernel function, and therefore we can use it directly in the Lagrangian without problems?

Answer: If for ANY function $g(\mathbf{x})$ of finite norm:

$$\int [g(\mathbf{x})]^2 d\mathbf{x} < \infty$$

the following condition holds:

$$\int K(\mathbf{x},\mathbf{y}) g(\mathbf{x}) g(\mathbf{y})\, d\mathbf{x}\, d\mathbf{y} \geq 0$$

then there exists a function $\mathbf{\Phi}(\mathbf{x})$ such that the following equation is true:

$$K(\mathbf{x},\mathbf{y}) = \sum_i \Phi_i(\mathbf{x})\Phi_i(\mathbf{y}) = \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{y})$$

Mercer's condition (2)

Question number 2: How easy is it to prove that a function $K(\mathbf{x},\mathbf{y})$ satisfies Mercer's conditon, and therefore is a suitable kernel?

Answer: Not particularly easy, since the condition must hold for any function $g(\mathbf{x})$ of finite norm.

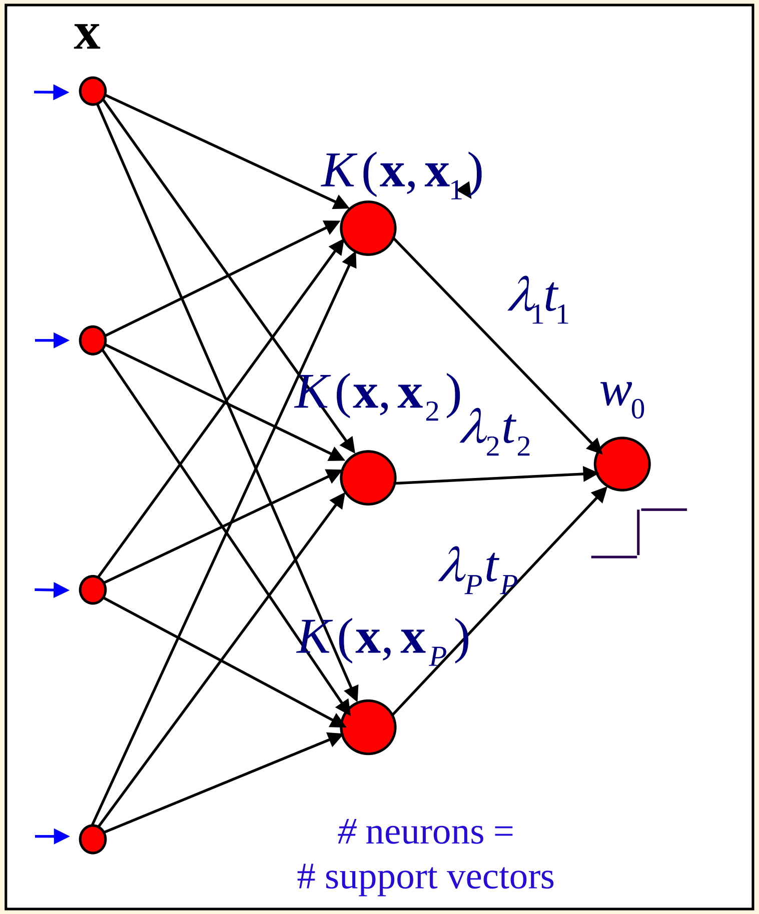
Question number 3: Suppose that I insert arbitrarily into the Lagrangian a function $K(\mathbf{x},\mathbf{y})$ which does not satisfy Mercer's condition. Am I in danger?

Answer:

- In this case, the quadratic optimization problem may not be convex.
- It is possible that I will end up with an infinite value of the Lagrangian and the problem may have no solution.
- This also depends on the training patterns themselves. The method may work for a classification problem, but not for a different one.
- Finally, the geometrical insight of mapping the original data to a higher dimensional space is lost.

SVM architecture for non-linear classification

$$y(\mathbf{x}) = \operatorname{sign}[\sum_i \lambda_i t_i K(\mathbf{x}, \mathbf{x}_i) + w_0]$$



Kernel examples (1)

Question number 4: For which functions $K(\mathbf{x},\mathbf{y})$ is it known that they satisfy Mercer's condition and therefore can be used in conjunction with non-linear SVMs?

Answer:

Polynomial kernel: $K(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y}+1)^D$

SVM output for pattern **x**: $\sum_i \lambda_i t_i K(\mathbf{x}_{new}\cdot\mathbf{x}_i) + w_0 = \sum_i \lambda_i t_i (\mathbf{x}\cdot\mathbf{x}_i+1)^D + w_0$

Gaussian kernel: $K(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right)$

SVM output for pattern **x**: $\sum_i \lambda_i t_i \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}\right) + w_0$

Remark: This is an RBF network! The difference with the RBFs we have already studied is that the centers $\mathbf{x}_\mathrm{i}$ are fixed: They coincide with the support vectors, since only for these vectors the corresponding λ_i are non-zero. Moreover, the weights λ_i, $w_\mathbf{0}$ are determined by the solution of the quadratic programming problem.

Kernel examples (2)

Sigmoid-logistic kernel? $K(\mathbf{x},\mathbf{y}) = \tanh(k\mathbf{x}\cdot\mathbf{y} - k_0)$

SVM output for pattern **x**: $\sum_i \lambda_i t_i \tanh(k\mathbf{x}\cdot\mathbf{x}_i - k_0) + w_0$

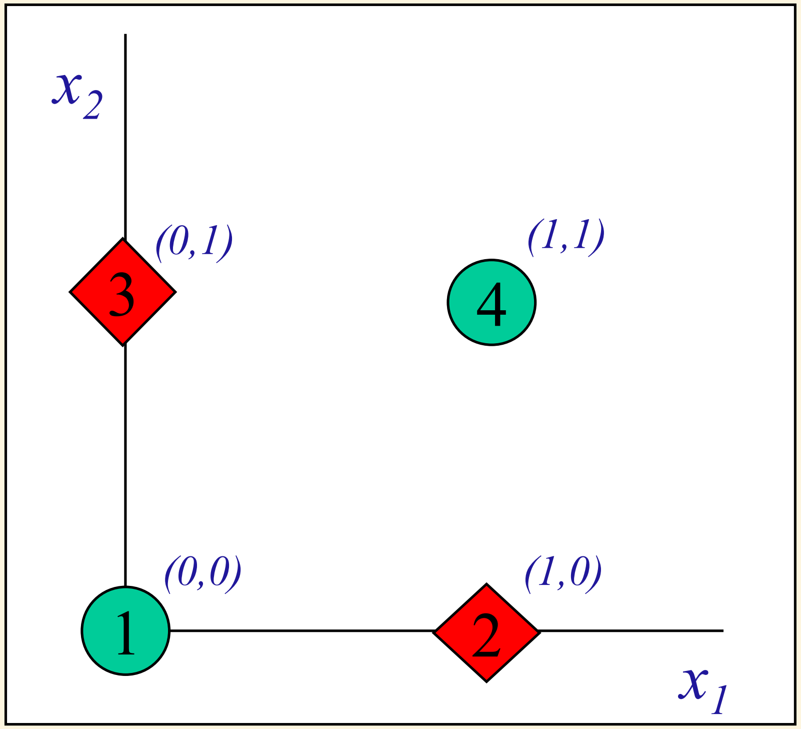
Remarks:

- This is a 2-layered perceptron with a hyperbolic tangent activation function in the hidden layer and linear outputs!
- However: Mercer's condition is satisfied only for some values of k and k_0 which in turn depend on the specific classification problem. For this reason, this kernel is not used as often as a Gaussian kernel.

Note: When using Gaussian or sigmoid kernels, the original space of the **Φ**'s is of infinite dimension and **Φ** cannot be computed in closed form. This is no problem for us. The dual formulation allows us to solve the problem based on the sole knowledge of the kernel function *K*.

The XOR problem: Polynomial Kernel (1)

A/A	x_1	x_2	Target
1	0	0	-1
2	1	0	1
3	0	1	1
4	1	1	-1



The XOR problem: Polynomial Kernel (2)

We choose to employ a 2nd degree polynomial kernel, albeit without the linear terms. The reason for omitting these terms is that we want to map the original 2 dimensional problem to 3 dimensions, in order to retain the option of visualization.

A mapping to three dimensions is achieved using the function:

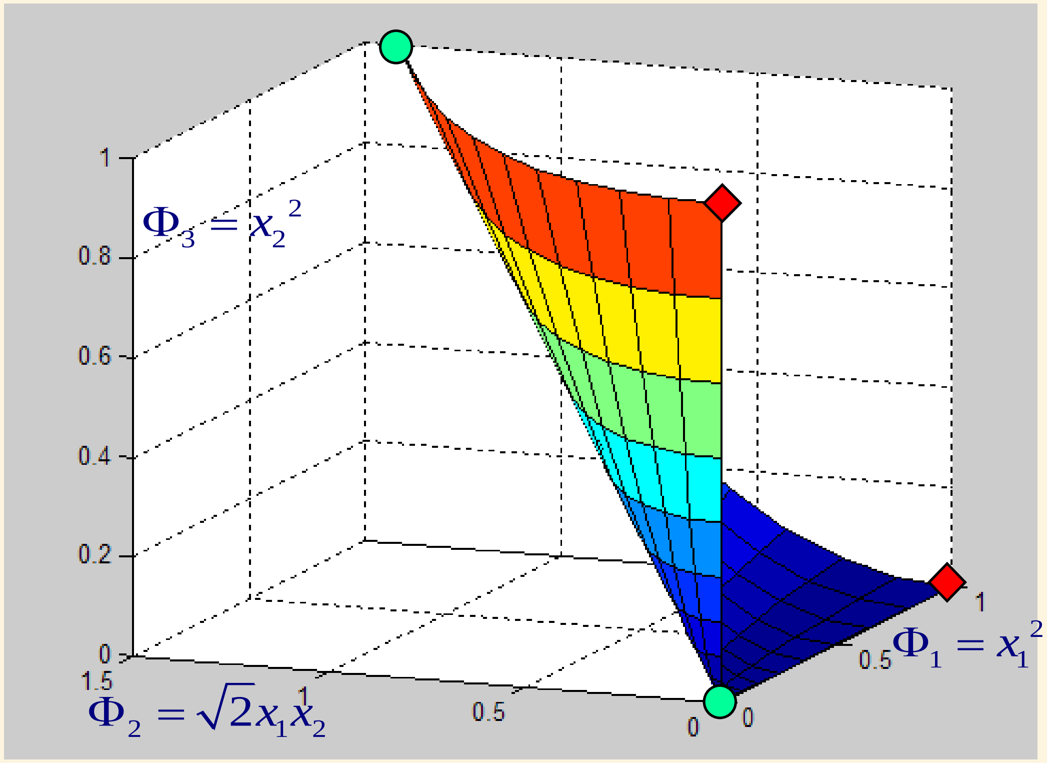
$$\mathbf{\Phi}(\mathbf{x}) = (x_1^{\,2}, \sqrt{2}x_1x_2, x_2^{\,2})$$

In the diagram, we can see the surface onto which the original square

$$0 \le x_1 \le 1, 0 \le x_2 \le 1$$

is mapped.

Looking at the diagram, it is evident that the patterns are now linearly separable.



The kernel is:

$$K(\mathbf{x},\mathbf{y}) = \mathbf{\Phi}(\mathbf{x})\cdot\mathbf{\Phi}(\mathbf{y}) = x_1^{\,2}y_1^{\,2} + 2x_1x_2y_1y_2 + x_2^{\,2}y_2^{\,2} = (x_1y_1 + x_2y_2)^2 = (\mathbf{x}\cdot\mathbf{y})^2$$

The XOR problem: Polynomial Kernel (3)

Let us calculate the matrix **Q** using the kernel function:

$$Q_{ij} = (\mathbf{x}_i \cdot \mathbf{x}_j)^2, i, j = 1, \ldots, 4 \qquad \mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 4 \end{bmatrix}$$

The dual Lagrangian is:

$$L_D = \sum_{i=1}^{4} \lambda_i - \tfrac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \lambda_i \lambda_j t_i t_j Q_{ij}$$

$$L_D = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 - \tfrac{1}{2} \lambda_2{}^2 - \tfrac{1}{2} \lambda_3{}^2 - 2\lambda_4{}^2 + \lambda_2 \lambda_4 + \lambda_3 \lambda_4$$

Hence the dual problem becomes:

Maximize: $L_D = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 - \tfrac{1}{2} \lambda_2{}^2 - \tfrac{1}{2} \lambda_3{}^2 - 2\lambda_4{}^2 + \lambda_2 \lambda_4 + \lambda_3 \lambda_4$

Under the constraints: $-\lambda_1 + \lambda_2 + \lambda_3 - \lambda_4 = 0, \quad \lambda_i \geq 0, \quad i = 1, 2, 3, 4$

Of course, when deciding about which patterns are support vectors, all subsets of the set of patterns in the training set have to be checked. It turns out in this case that all 4 patterns are support vectors.

Our expressions are symmetric with respect to λ_2 and λ_3. We seek a solution with $\lambda_3 = \lambda_2$.

The XOR problem: Polynomial Kernel (4)

Upon elimination of λ_1 and λ_3 the Lagrangian becomes:

$$L_D = 4\lambda_2 - \lambda_2{}^2 - 2\lambda_4{}^2 + 2\lambda_2\lambda_4$$

Equating the derivatives to zero, we get:

$$\left.\begin{aligned} \frac{\partial L_D}{\partial \lambda_2} &= 4 - 2\lambda_2 + 2\lambda_4 = 0 \\ \frac{\partial L_D}{\partial \lambda_4} &= -4\lambda_4 + 2\lambda_2 = 0 \end{aligned}\right\} \Rightarrow \lambda_2 = 4, \quad \lambda_4 = 2$$

and therefore: $\lambda_1 = 6, \quad \lambda_2 = \lambda_3 = 4, \quad \lambda_4 = 2$

•We find w_0 by considering that the first pattern is a support vector with output target equal to -1:

$$\sum_{i=1}^{4} \lambda_i t_i (\mathbf{x}_1 \cdot \mathbf{x}_i)^2 + w_0 = -1 \Rightarrow w_0 = -1$$

The XOR problem: Polynomial Kernel (5)

The output for every new pattern **x** is given by:

$$-6(\mathbf{x}_1 \cdot \mathbf{x})^2 + 4(\mathbf{x}_2 \cdot \mathbf{x})^2 + 4(\mathbf{x}_3 \cdot \mathbf{x})^2 - 2(\mathbf{x}_4 \cdot \mathbf{x})^2 - 1$$

And the discriminating surface of the 2 classes is determined by the equation:

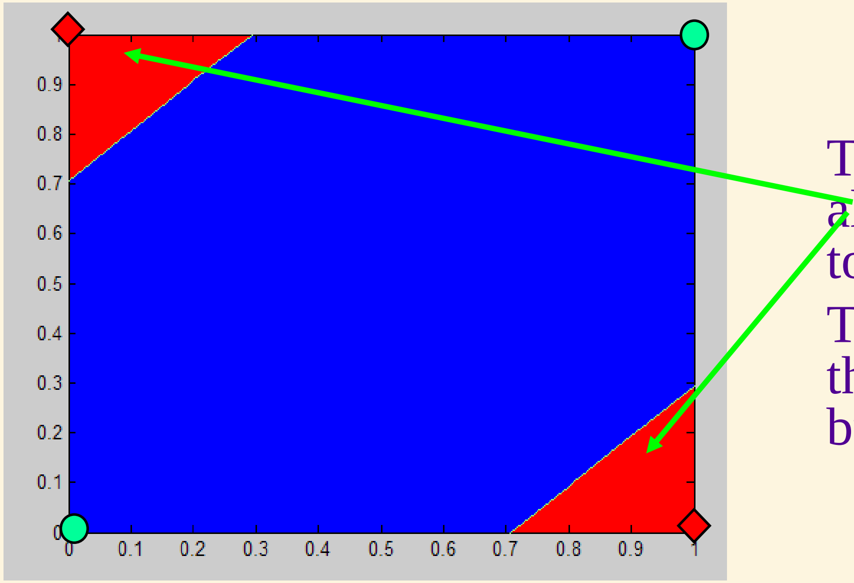
$$-6(\mathbf{x}_1 \cdot \mathbf{x})^2 + 4(\mathbf{x}_2 \cdot \mathbf{x})^2 + 4(\mathbf{x}_3 \cdot \mathbf{x})^2 - 2(\mathbf{x}_4 \cdot \mathbf{x})^2 - 1 = 0$$

i.e.:

$$0 + 4x_1{}^2 + 4x_2{}^2 - 2(x_1 + x_2)^2 - 1 = 0$$

$$\Rightarrow 2x_1{}^2 - 4x_1x_2 + 2x_2{}^2 - 1 = 0$$

The border between the two classes in the original plane (x_1, x_2) is shown in the following diagram:



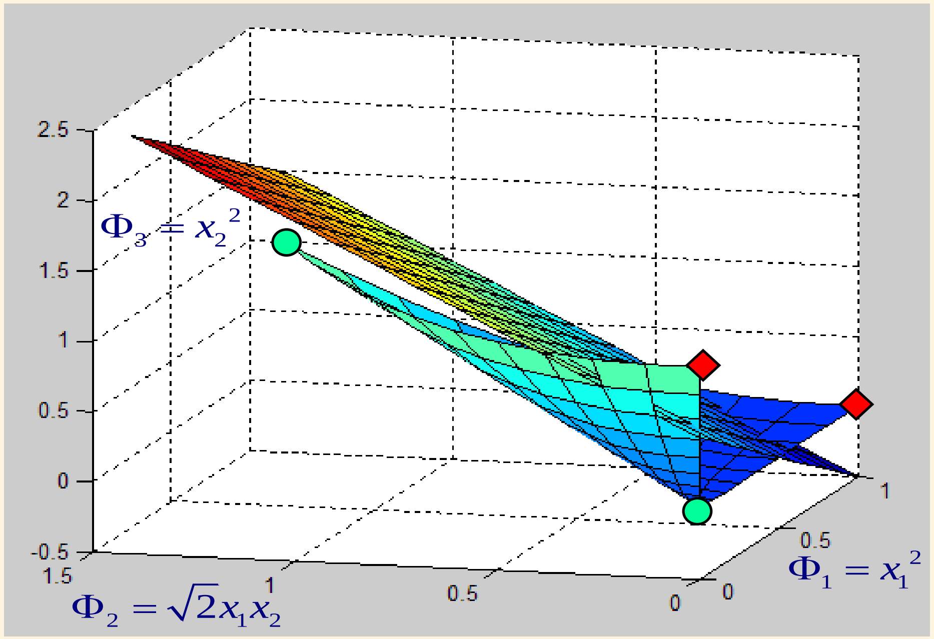
The non-linear SVM has allocated relatively small space to the second class. Why?

To understand this, we return to the 3 dimensional space defined by **Φ**.

The XOR problem: Polynomial Kernel (6)

$$2x_1^2 - 4x_1x_2 + 2x_2^2 - 1 = 0 \Rightarrow$$
$$2\Phi_1 - 2\sqrt{2}\Phi_2 + 2\Phi_3 - 1 = 0$$

- The last equation characterizes the discriminating plane in the space of Φ. It is easy to verify that the images of the original patterns of the training set are equidistant from this plane (as they ought, since they are support vectors).
- However, the use of the non-linear kernel is responsible for the relatively small portions of the surface which lie „above” the plane.
- Therefore, the choice of the kernel is crucial to the achievement of satisfactory generalization performance.
- The problem of optimally choosing an appropriate kernel for a given classification problem is open.



The XOR problem: Polynomial Kernel (7)

Training patterns

$$\begin{array}{c} \mathbf{x} \qquad \mathbf{\Phi} \\ (0,0) \rightarrow (0,0,0) \\ (1,0) \rightarrow (1,0,0) \\ (0,1) \rightarrow (0,0,1) \\ (1,1) \rightarrow (1,\sqrt{2},1) \end{array}$$

Distance of point from plane $\quad 2\Phi_1 - 2\sqrt{2}\Phi_2 + 2\Phi_3 - 1 \doteq 0$

$$\frac{2\Phi_1 - 2\sqrt{2}\Phi_2 + 2\Phi_3 - 1}{\pm\sqrt{2^2 + (2\sqrt{2})^2 + 2^2}} = \tfrac{1}{2}\left|\Phi_1 - \sqrt{2}\Phi_2 + \Phi_3 - \tfrac{1}{2}\right|$$

A simple substitution is enough to verify that the absolute distance of all 4 training patterns from the plane is equal to ¼.

Non-linear support vector regression

Dual Problem Formulation

Maximize, with respect to $\lambda_i, \lambda_i^{'}$:

$$-\tfrac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P}\left(\lambda_i-\lambda_i^{'}\right)\left(\lambda_j-\lambda_j^{'}\right)K(\mathbf{x}_i,\mathbf{x}_j)+\sum_{i=1}^{P}\left(\lambda_i-\lambda_i^{'}\right)y_i-\varepsilon\sum_{i=1}^{P}\left(\lambda_i+\lambda_i^{'}\right)$$

under the constraints:

$$0\le\lambda_i\le C,\quad 0\le\lambda_i^{'}\le C,\quad i=1,2,\ldots,P$$

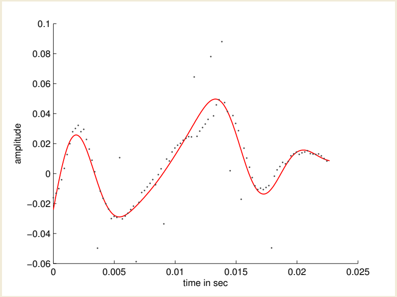
$$\sum_{i=1}^{P}\left(\lambda_i-\lambda_i^{'}\right)=0$$

Prediction for new pattern

$$y_{new}=\sum_{i=1}^{P}\left(\lambda_i-\lambda_i^{'}\right)K(\mathbf{x}_i,\mathbf{x}_{new})+w_0$$

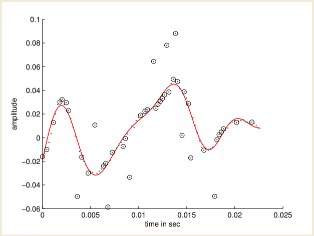
Kernel Ridge Regression: An Example

- In this example, the prediction power of the kernel ridge regression, in the presence of Gaussian noise as well as of **outliers**, will be tested. The original data were samples from a music recording from Blade Runner by Vangelis Papathanasiou. A white Gaussian noise was then added at a 15dB level and a number of outliers were intentionally randomly introduced and "hit" some of the values (10%). The kernel ridge regression method was used, employing the Gaussian kernel with $\sigma = 0.004$. A bias term was also present, as discussed before. The prediction (fitted) curve, $\hat{y}(x)$, for various value of x, is shown in the figure below, together with the (noisy) data used for training.

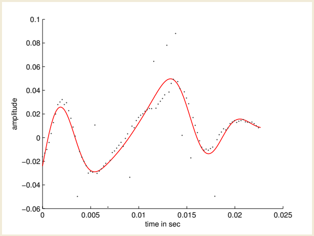


Support Vector Regression: An Example

- Consider the same time series used for the nonlinear prediction task, used in the kernel ridge regression example. This time, the SVR method was used and optimized around the linear ϵ-insensitive loss function, with $\epsilon = 0.003$. The same Gaussian kernel, with $\sigma = 0.004$, was employed, as in the kernel ridge regression case. Figure (a) below shows the resulting prediction curve, $\hat{y}(x)$, as a function of x given in (8). The curve fits the data samples much better compared to the kernel ridge regression (Figure (b)), exhibiting the enhanced robustness of the SVR method, relative to the KKR, in the presence of outliers.



(a)



(b)

The improved performance compared to the kernel ridge regression used is readily observed, by simply observing the two figures. The encircled points are the support vectors resulting from the optimization, using the ϵ-insensitive loss function.