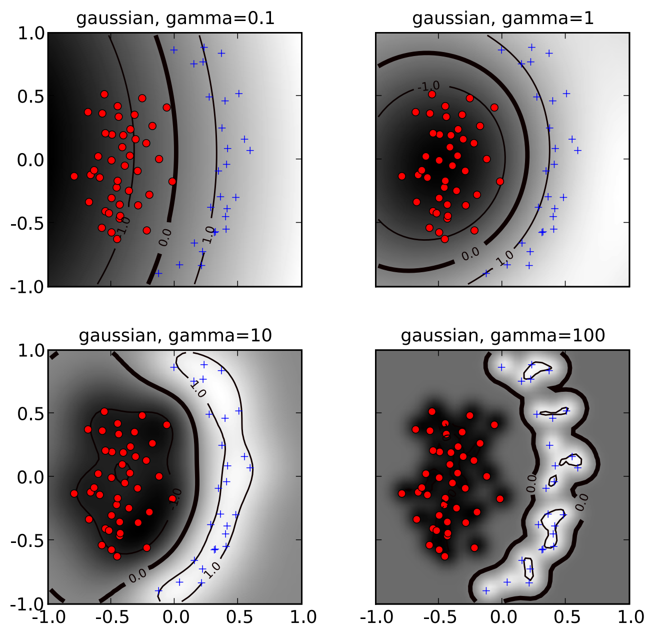
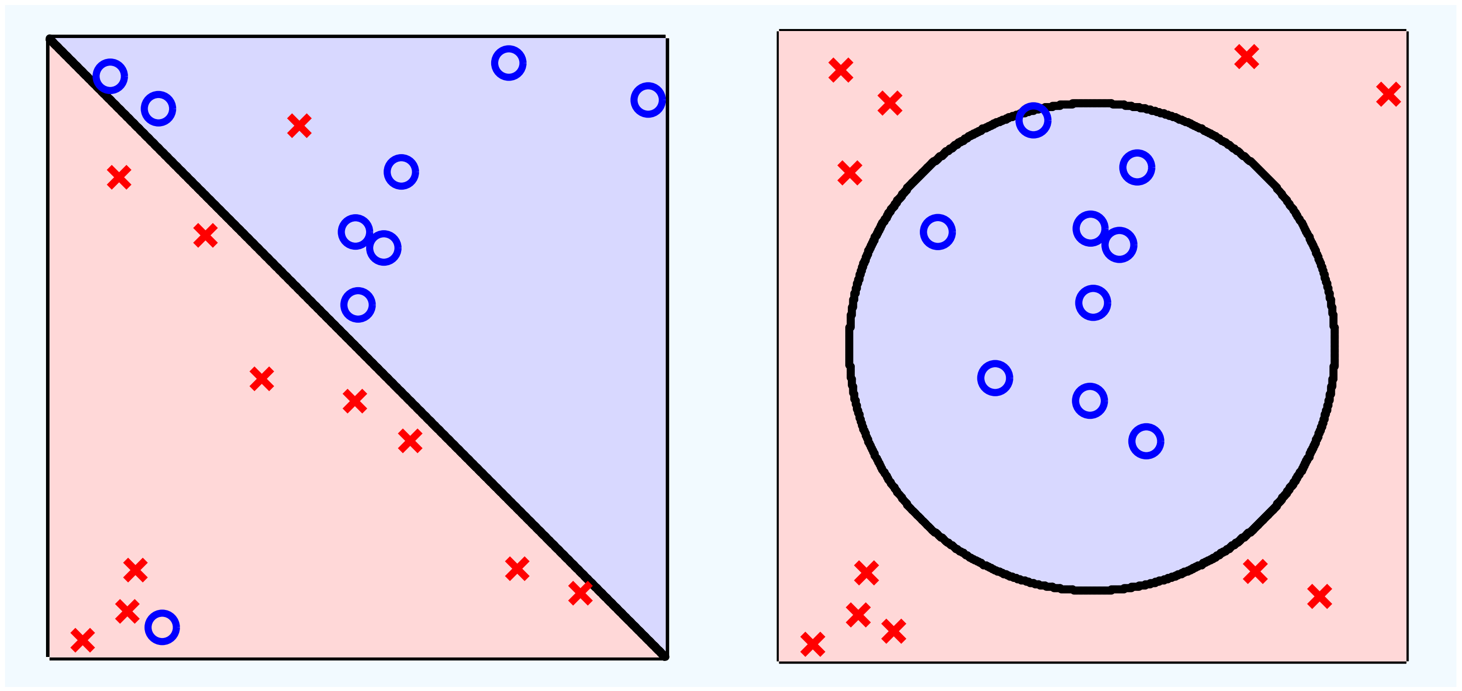
SVMs: nonlinearity through kernels

Chapter 3.4, e-8



Non-separable data

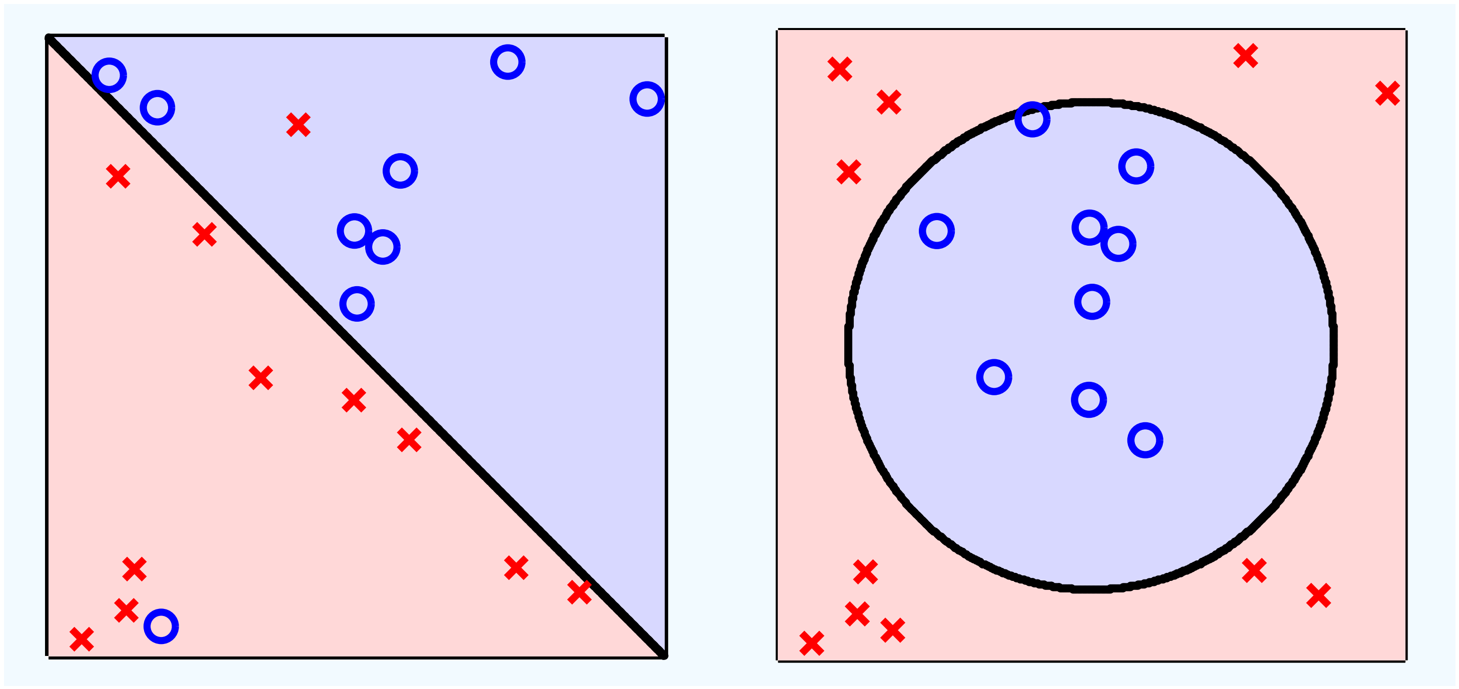
Consider the following two datasets:



Both are not linearly separable. But there is a difference!

Non-separable data

Consider the following two datasets:

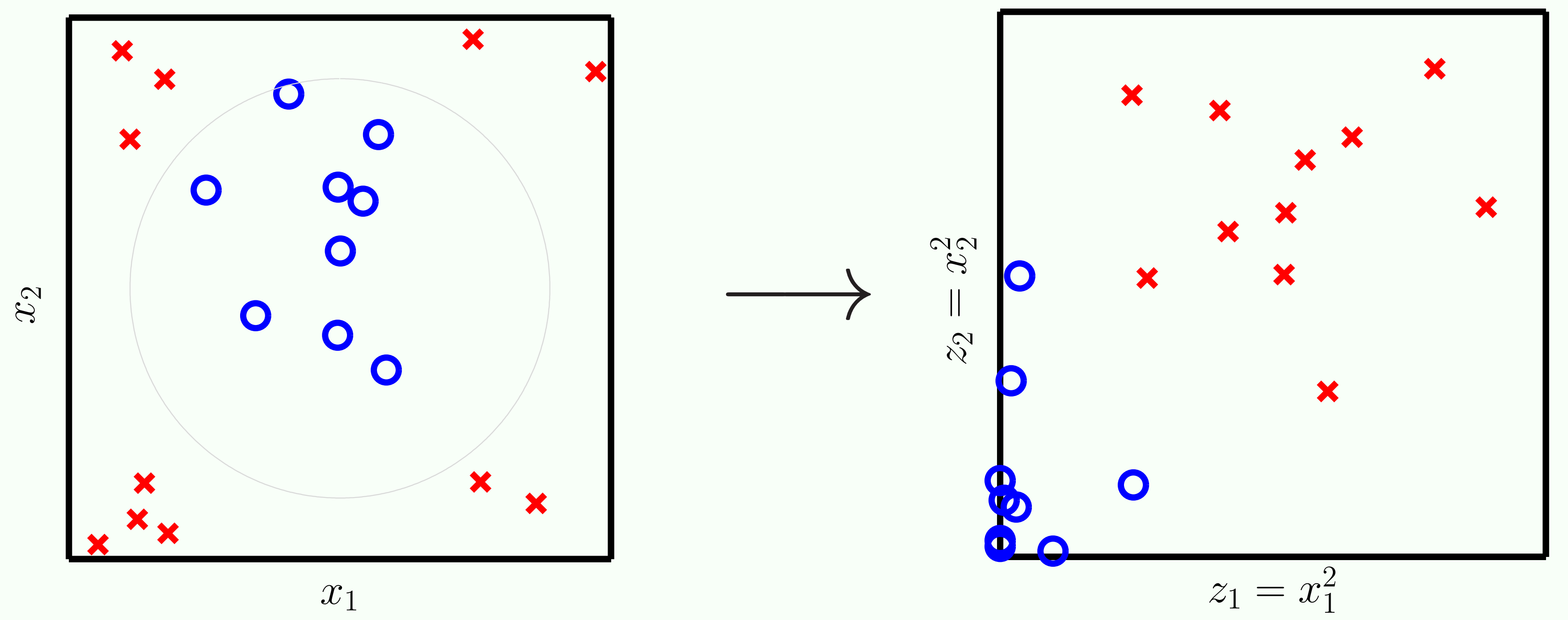


Linear with outliers

Nonlinear

Transform your features!

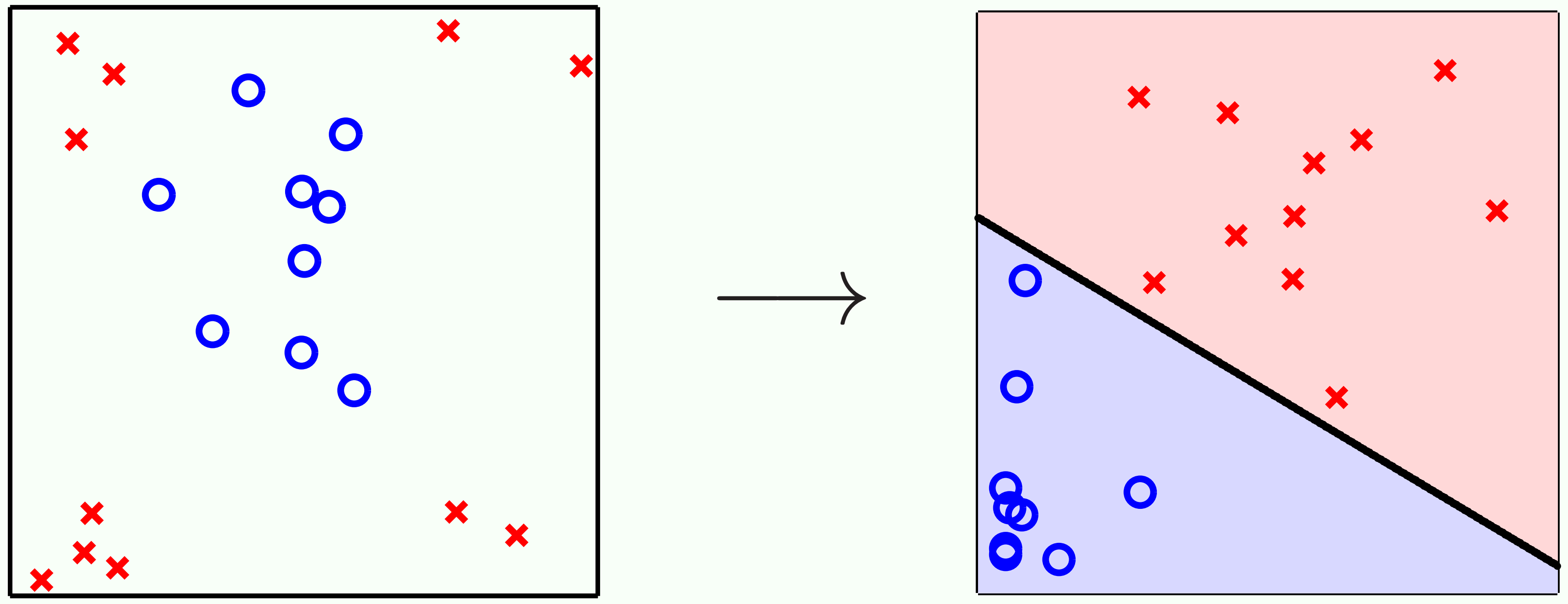
Map your data into "Z-space" using a nonlinear function



$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \longrightarrow \mathbf{z} = \mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix} = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \Phi_2(\mathbf{x}) \end{bmatrix}$$

Classification in Z-space

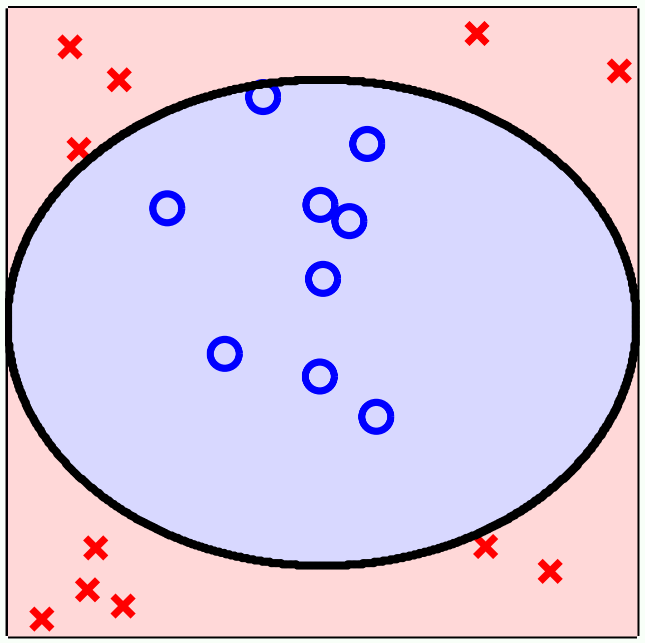
In Z-space the data can be linearly separated: $\tilde{g}(\mathbf{z}) = \text{sign}(\tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{z})$



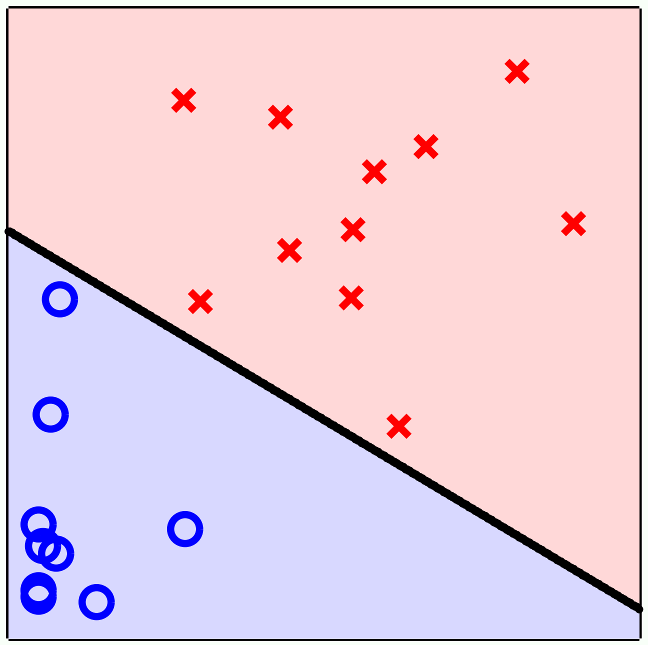
Classification in Z-space

$$\begin{aligned} g(\mathbf{x}) &= \tilde{g}(\mathbf{\Phi}(\mathbf{x})) \\ &= \text{sign}(\tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{\Phi}(\mathbf{x})) \end{aligned}$$

$$\tilde{g}(\mathbf{z}) = \text{sign}(\tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{z})$$

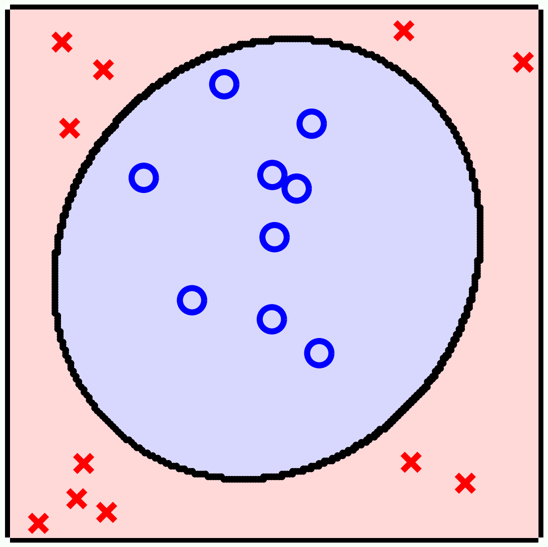


$$\longleftarrow$$



Classification in Z-space

$$\begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} = \mathbf{x} \longrightarrow \mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \Phi_2(\mathbf{x}) \\ \Phi_3(\mathbf{x}) \\ \Phi_4(\mathbf{x}) \\ \Phi_5(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}$$



The polynomial Z-space

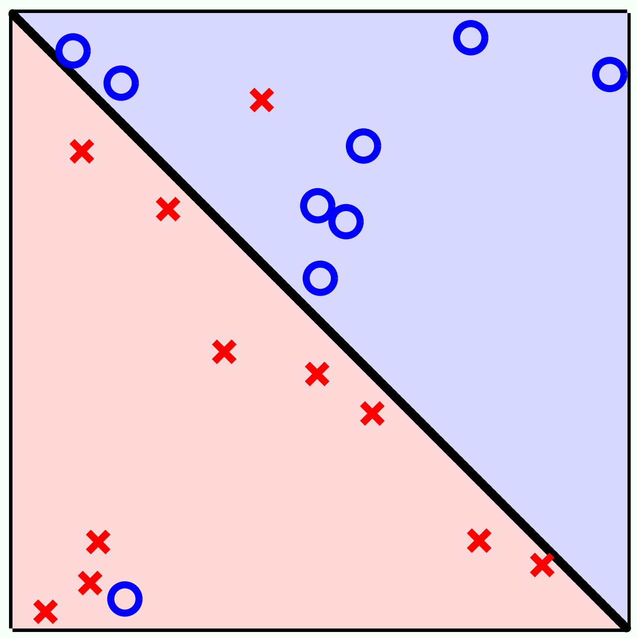
We can choose higher order polynomials:

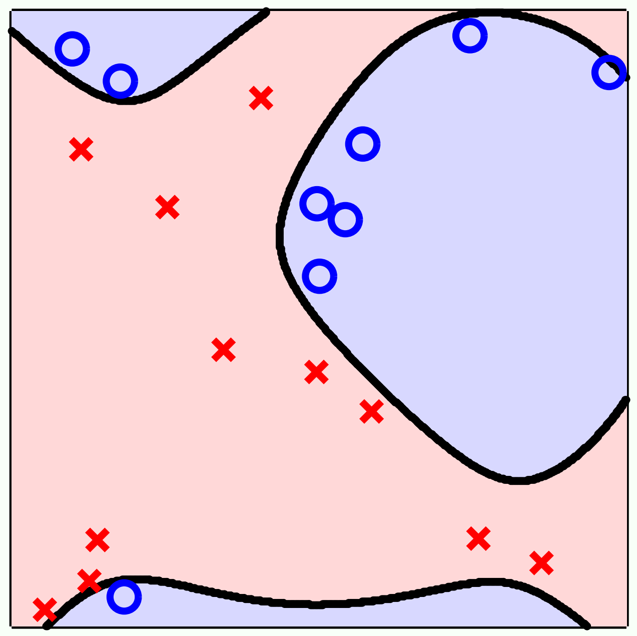
$$\begin{aligned}
\mathbf{\Phi}_1(\mathbf{x}) &= (1, x_1, x_2), \\
\mathbf{\Phi}_2(\mathbf{x}) &= (1, x_1, x_2, x_1^2, x_1x_2, x_2^2), \\
\mathbf{\Phi}_3(\mathbf{x}) &= (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3), \\
\mathbf{\Phi}_4(\mathbf{x}) &= (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3, x_1^4, x_1^3x_2, x_1^2x_2^2, x_1x_2^3, x_2^4), \\
&\vdots
\end{aligned}$$

What can we say about the dimensionality of the Z-space as a function of the dimensionality of the data?

What are the potential effects of increasing the order of the polynomial?

The polynomial Z-space





Feature-space dimensionality increases rapidly and with it the complexity of the model: danger of overfitting

A few potential issues…

Transforming the data seems like a good idea:

- Better chance of obtaining linear separability

But:

- At the expense of potential overfitting
- Computationally expensive (memory and time)

Kernels: avoid the computational expense by an implicit mapping

Achieving non-linear discriminant functions

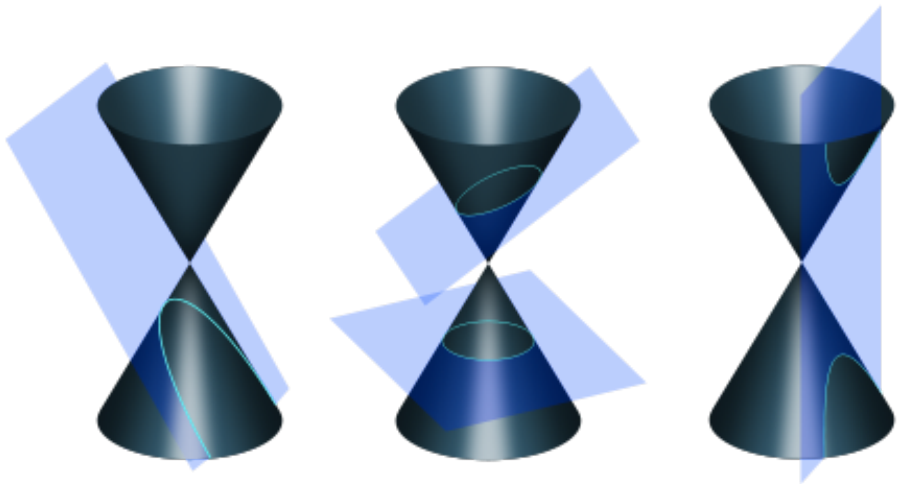
Consider two dimensional data and the mapping

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

Let's plug that into the discriminant function:

$$\mathbf{w}^\mathsf{T}\phi(\mathbf{x}) = w_1x_1^2 + \sqrt{2}w_2x_1x_2 + w_3x_2^2$$

The resulting decision boundary is a conic section.



How to avoid the overhead of explicit mapping

Suppose the weight vector can be expressed as:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$$

The discriminant function is then:

$$\mathbf{w}^{\mathsf{T}}\mathbf{x} + b = \sum_i \alpha_i \mathbf{x}_i^{\mathsf{T}} \mathbf{x} + b$$

And using our nonlinear mapping:

$$\mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}) + b = \sum_i \alpha_i \phi(\mathbf{x}_i)^{\mathsf{T}} \phi(\mathbf{x}) + b$$

Turns out we can often compute the dot product without explicitly mapping the data into a high dimensional feature space!

Example

Let's go back to the example

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

and compute the dot product

$$\begin{aligned}\phi(\mathbf{x})^\intercal\phi(\mathbf{z}) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)^\intercal(z_1^2, \sqrt{2}z_1z_2, z_2^2) \\ &= x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2 \\ &= (\mathbf{x}^\intercal\mathbf{z})^2\end{aligned}$$

Do we need to perform the mapping explicitly?

Example

Let's go back to the example

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

and compute the dot product

$$\begin{aligned}\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{z}) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)^\mathsf{T}(z_1^2, \sqrt{2}z_1z_2, z_2^2) \\ &= x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2 \\ &= (\mathbf{x}^\mathsf{T}\mathbf{z})^2\end{aligned}$$

Do we need to perform the mapping explicitly?

NO! Squaring the dot product in the original space has the same effect as computing the dot product in feature space.

Kernels

Definition: A function k(**x**, **z**) that can be expressed as a dot product in some feature space is called a kernel.

In other words, k(**x**, **z**) is a kernel if there exists $\phi : \mathcal{X} \mapsto \mathcal{F}$ such that

$$k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\mathsf{T} \phi(\mathbf{z})$$

Why is this interesting?

If the algorithm can be expressed in terms of dot products, we can work in the feature space without performing the mapping explicitly!

The dual SVM problem

The dual SVM formulation depends on the data through dot products, and so can be expressed using kernels. Replace

$$\underset{\alpha}{\text{maximize}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^{\mathsf{T}} \mathbf{x}_j$$

$$\text{subject to: } 0 \le \alpha_i \le C, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

with:

$$\underset{\alpha}{\text{maximize}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to: } 0 \le \alpha_i \le C, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Standard kernel functions

The linear kernel

$$k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\mathsf{T}\mathbf{z}$$

Homogeneous polynomial kernel

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\mathsf{T}\mathbf{z})^d$$

Polynomial kernel

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\mathsf{T}\mathbf{z} + 1)^d$$

Gaussian kernel

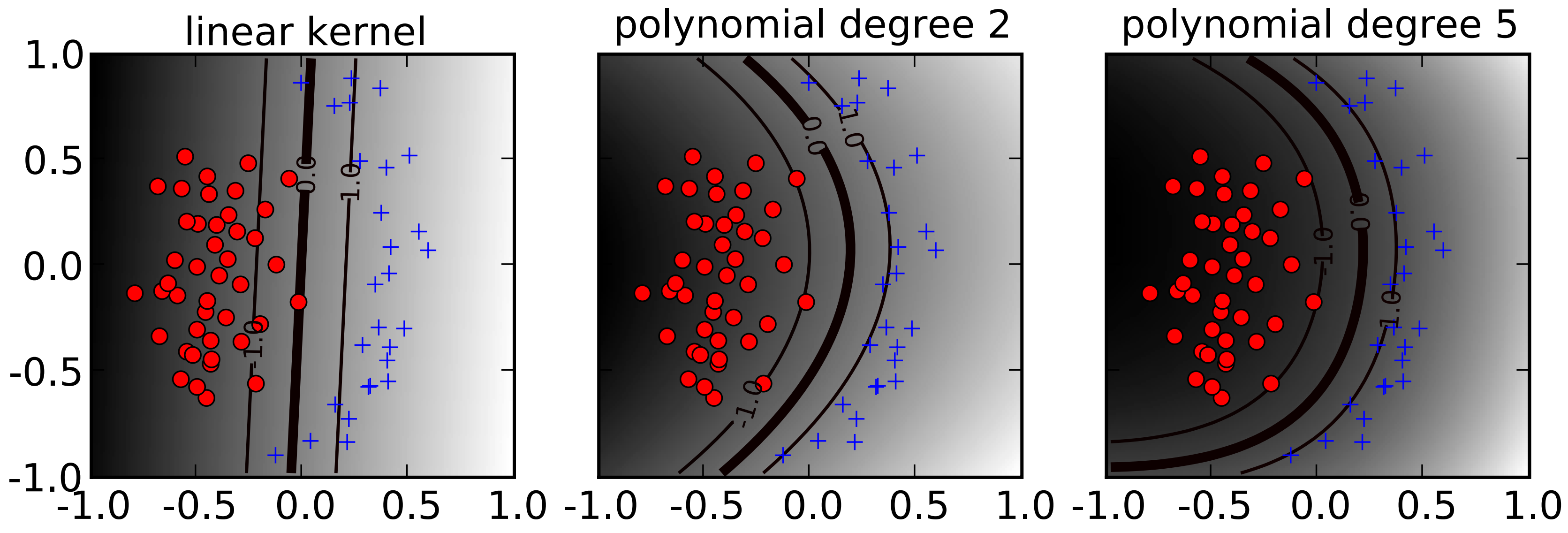
$$k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma||\mathbf{x} - \mathbf{z}||^2)$$

Standard kernel functions

	Feature space:
The linear kernel $$k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\intercal \mathbf{z}$$	Original features
Homogeneous polynomial kernel $$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\intercal \mathbf{z})^d$$	All monomials of degree d
Polynomial kernel $$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\intercal \mathbf{z} + 1)^d$$	All monomials of degree less or equal to d
Gaussian kernel $$k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma \|\mathbf{x} - \mathbf{z}\|^2)$$	Infinite dimensional

Demo

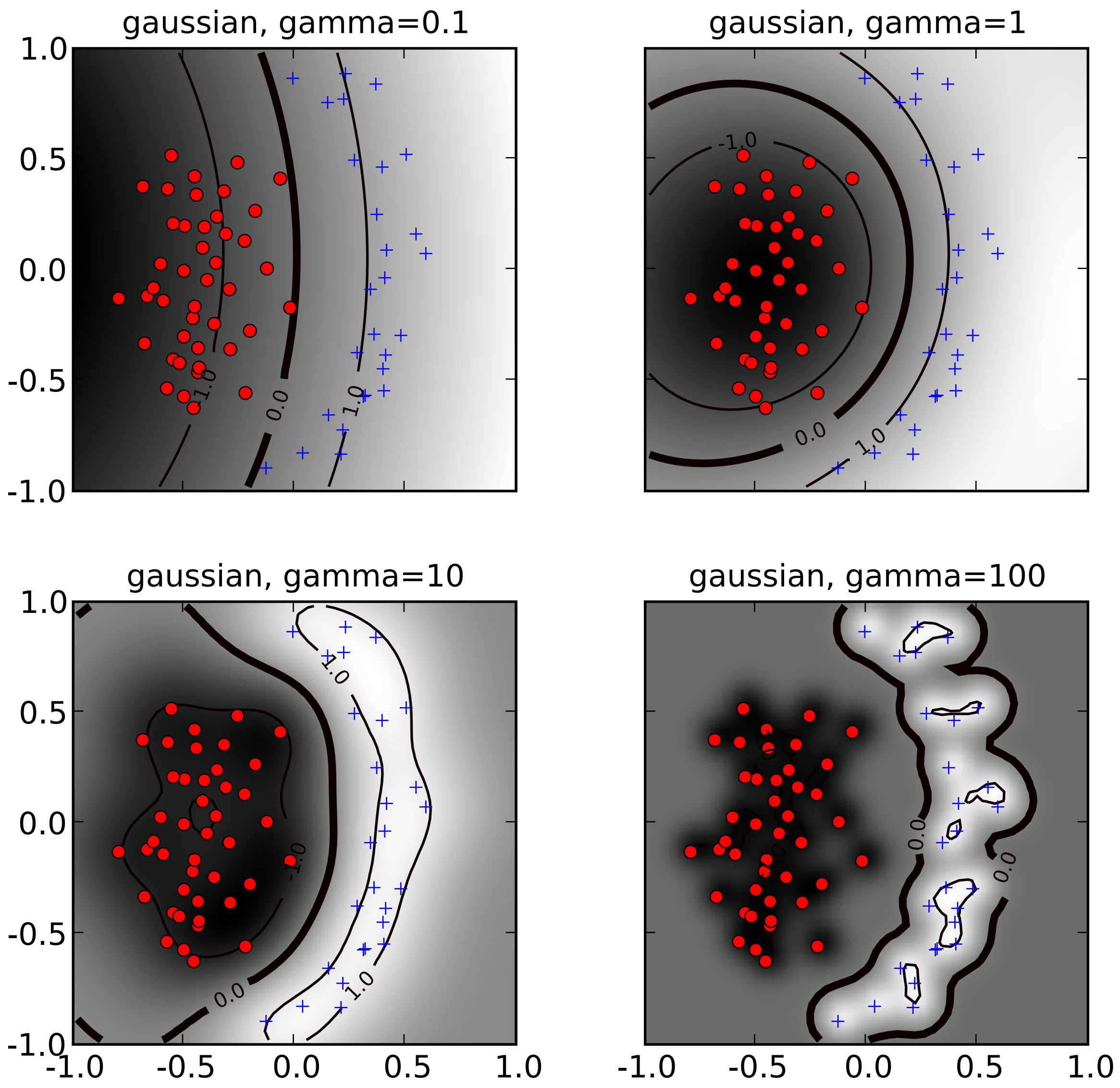
Using polynomial kernel:



$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\intercal \mathbf{z} + 1)^d$$

Demo

Using the Gaussian kernel: $k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma||\mathbf{x} - \mathbf{z}||^2)$



"Kernelizing" the perceptron algorithm

Recall the primal version of the perceptron algorithm:

```
Input:  labeled data D in homogeneous coordinates
Output:  weight vector w

w = 0
converged = false
while not converged :
    converged = true
    for i in 1,…,|D| :
        if x_i is misclassified update w and set
            converged=false
```

$$\mathbf{w}' = \mathbf{w} + \eta y_i \mathbf{x}_i$$

What do you need to change to express it in the dual?
(I.e. express the algorithm in terms of the alpha coefficients)

“Kernelizing“ the perceptron algorithm

```
Input:  labeled data D in homogeneous coordinates
Output:  weight vector $\boldsymbol{\alpha}$

$\boldsymbol{\alpha}$ = 0
converged = false
while not converged :
    converged = true
    for i in 1,…,|D| :
        if $\mathbf{x}_i$ is misclassified update $\boldsymbol{\alpha}$ and set
            converged=false
```

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$$

“Kernelizing” the perceptron algorithm

```
Input:  labeled data D in homogeneous coordinates
Output:  weight vector $\boldsymbol{\alpha}$

$\boldsymbol{\alpha}$ = 0
converged = false
while not converged :
    converged = true
    for i in 1,…,|D| :
        if $\mathbf{x}_i$ is misclassified update $\boldsymbol{\alpha}$ and set
            converged=false
```

The update $\alpha_i \rightarrow \alpha_i + \eta y_i$

Is equivalent to:

$$\mathbf{w}' = \mathbf{w} + \eta y_i \mathbf{x}_i$$

Linear regression revisited

The sum-squared cost function:

$$\sum_i (y_i - \mathbf{w}^\intercal \mathbf{x}_i)^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})^\intercal (\mathbf{y} - \mathbf{X}\mathbf{w})$$

The optimal solution satisfies:

$$\mathbf{X}^\intercal \mathbf{X}\mathbf{w} = \mathbf{X}^\intercal \mathbf{y}$$

If we express w as:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i = \mathbf{X}^\intercal \alpha$$

We get:

$$\mathbf{X}^\intercal \mathbf{X}\mathbf{X}^\intercal \alpha = \mathbf{X}^\intercal \mathbf{y}$$

Kernel linear regression

We now get that α satisfies:

$$\mathbf{X}\mathbf{X}^\intercal \alpha = \mathbf{y}$$

Compare with:

$$\mathbf{X}^\intercal \mathbf{X}\mathbf{w} = \mathbf{X}^\intercal \mathbf{y}$$

Which is harder to find? What have we gained?

The kernel matrix

$\mathbf{X}^\intercal\mathbf{X}$ The covariance matrix (d x d)

$\mathbf{X}\mathbf{X}^\intercal$ Matrix of dot products associated with a dataset (n x n).

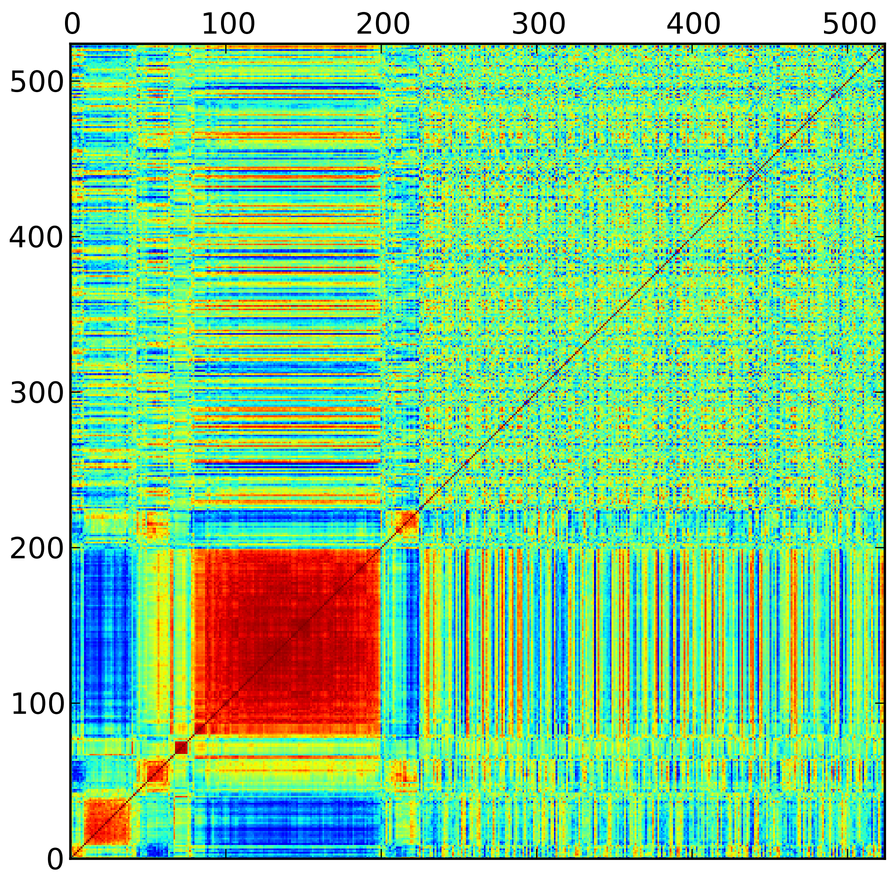
Can replace it with a matrix K such that:

$$K_{ij} = \phi(\mathbf{x}_i)^\intercal\phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$$

This is the kernel matrix associated with a dataset
a.k.a the Gram matrix

How does that matrix look like?

Kernel matrix for gene expression data in yeast:



Properties of the kernel matrix

The kernel matrix:

$$K_{ij} = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$$

- Symmetric (and therefore has real eigenvalues)
- Diagonal elements are positive
- Every kernel matrix is positive semi-definite, i.e.

$$\mathbf{x}^\top K \mathbf{x} \geq 0 \quad \forall \mathbf{x}$$

- Corollary: the eigenvalues of a kernel matrix are positive

Standard kernel functions

Feature space:

The linear kernel

$$k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\mathsf{T}\mathbf{z}$$

Original features

Homogeneous polynomial kernel

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\mathsf{T}\mathbf{z})^d$$

All monomials of degree d

Polynomial kernel

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\mathsf{T}\mathbf{z} + 1)^d$$

All monomials of degree less or equal to d

Gaussian kernel (aka RBF kernel)

$$k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma||\mathbf{x} - \mathbf{z}||^2)$$

Infinite dimensional

How do we even know that the Gaussian is a valid kernel?

Some tricks for constructing kernels

Let $K(x,z)$ be a kernel function

If $a > 0$ then $a \cdot K(x,z)$ is a kernel

Some tricks for constructing kernels

Sums of kernels are kernels:

Let K_1 and K_2 be kernel functions then $K_1 + K_2$ is a kernel

What is the feature map that shows this?

Some tricks for constructing kernels

Sums of kernels are kernels:

Let K_1 and K_2 be kernel functions then $K_1 + K_2$ is a kernel

Feature map: concatenation of the underlying feature maps

Some tricks for constructing kernels

Products of kernels are kernels:

Let K_1 and K_2 be kernel functions then K_1 K_2 is a kernel

Some tricks for constructing kernels

Products of kernels are kernels:

Let K_1 and K_2 be kernel functions then $K_1 K_2$ is a kernel

Construct a feature map that contains all products of pairs of features

The cosine kernel

If K(x,z) is a kernel then

$$K'(x,z) = \frac{K(x,z)}{\sqrt{K(x,x)K(z,z)}}$$

is a kernel

The cosine kernel

If K(x,z) is a kernel then

$$K'(x,z) = \frac{K(x,z)}{\sqrt{K(x,x)K(z,z)}}$$

is a kernel

This kernel is equivalent to normalizing each example to have unit norm in the feature space associated with the kernel.

This kernel is the cosine in the feature space associated with the kernel K:

$$\cos(\phi(x),\phi(z)) = \frac{\phi(x)^\intercal\phi(z)}{||\phi(x)||\,||\phi(z)||} = \frac{\phi(x)^\intercal\phi(z)}{\sqrt{\phi(x)^\intercal\phi(x)\;\phi(z)^\intercal\phi(z)}}$$

Infinite sums of kernels

Theorem: A function $K(\mathbf{x},\mathbf{z}) = K(\mathbf{x}^\intercal\mathbf{z})$ with a series expansion

$$K(t) = \sum_{n=0}^{\infty} a_n t^n$$

is a kernel iff $a_n \geq 0$ for all n.

Infinite sums of kernels

Theorem: A function $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{x}^\intercal \mathbf{z})$ with a series expansion

$$K(t) = \sum_{n=0}^{\infty} a_n t^n$$

is a kernel iff $a_n \geq 0$ for all n.

Corollary: $K(\mathbf{x}, \mathbf{z}) = \exp\left(2\gamma \mathbf{x}^\intercal \mathbf{z}\right)$ is a kernel

Infinite sums of kernels

Theorem: A function $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{x}^\intercal \mathbf{z})$ with a series expansion

$$K(t) = \sum_{n=0}^{\infty} a_n t^n$$

is a kernel iff $a_n \geq 0$ for all n.

Corollary: $K(\mathbf{x}, \mathbf{z}) = \exp\left(2\gamma \mathbf{x}^\intercal \mathbf{z}\right)$ is a kernel

Corollary: $k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma ||\mathbf{x} - \mathbf{z}||^2)$ is a kernel

$$\exp(-\gamma ||\mathbf{x} - \mathbf{z}||^2) = \exp\left(-\gamma\left(\mathbf{x}^\intercal \mathbf{x} + \mathbf{z}^\intercal \mathbf{z}\right) + 2\gamma\left(\mathbf{x}^\intercal \mathbf{z}\right)\right)$$

i.e. the Gaussian kernel is the cosine kernel of the exponential kernel

$$K'(x, z) = \frac{K(x, z)}{\sqrt{K(x, x)K(z, z)}}$$

The dual

In our context the concept of "dual" can refer to two things:

- The dual optimization problem (in terms of alphas instead of optimizing w directly)
- Any time you express a classification/regression algorithm in terms of variables alpha such that $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$