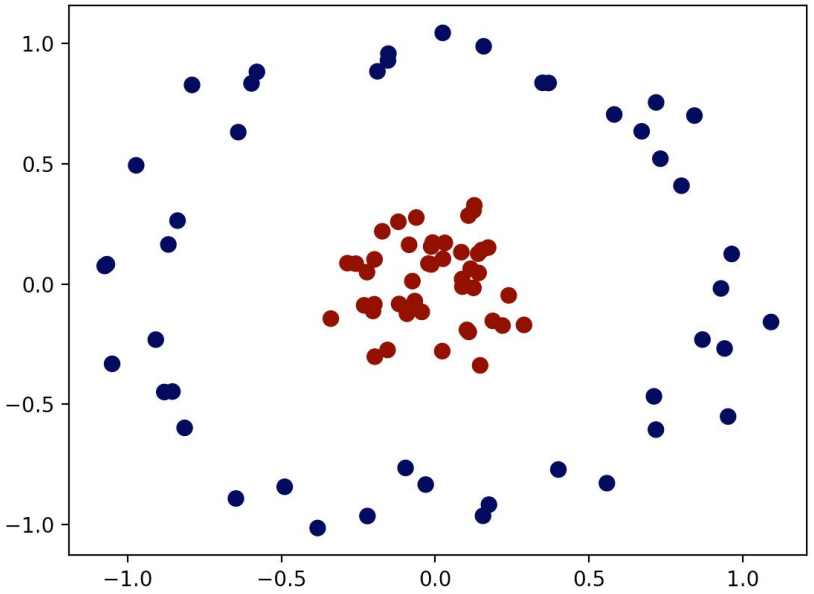
Kernel Methods

- In previous part, we tackle down the complexity by searching for “large margin” separator

- In this part, we introduce the method of kernel to tackle down the complexity
  - Embeddings into feature space
  - The kernel trick
  - Implementing soft-SVM with kernels

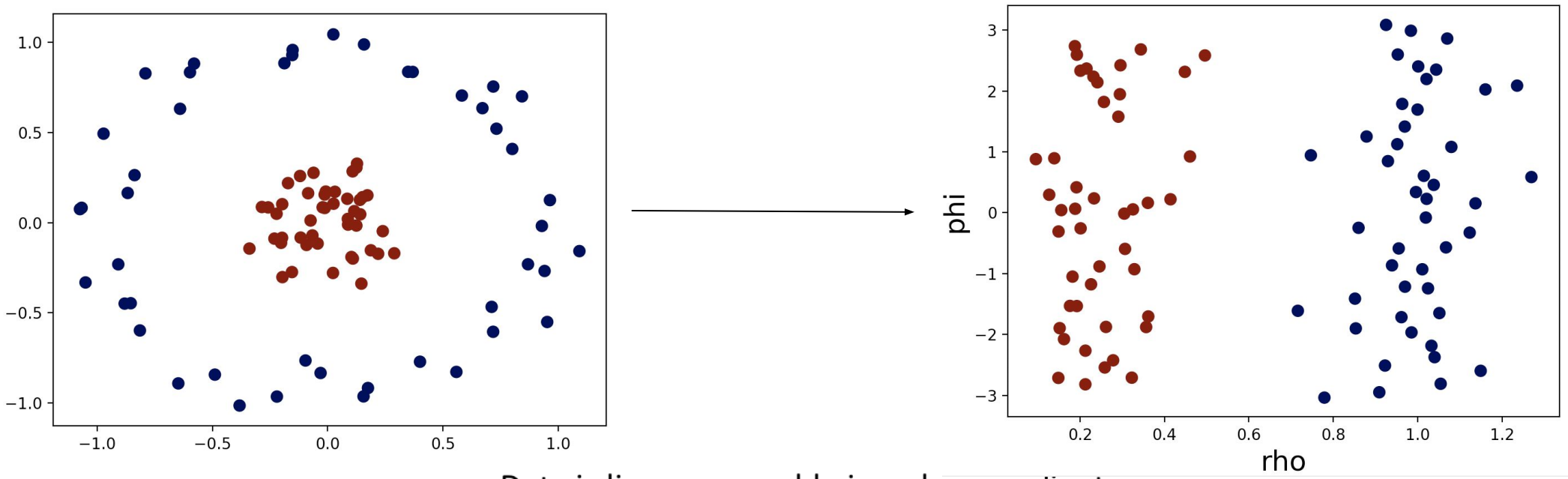
Embeddings into feature space

- The power of previously introduced SVM is rather restricted. Very few questions in practice are linear separable, we always need to map it into higher dimensions
- A very simple example is as follows:



Embeddings into feature space

- Possible solution 1: Use polar coordinates

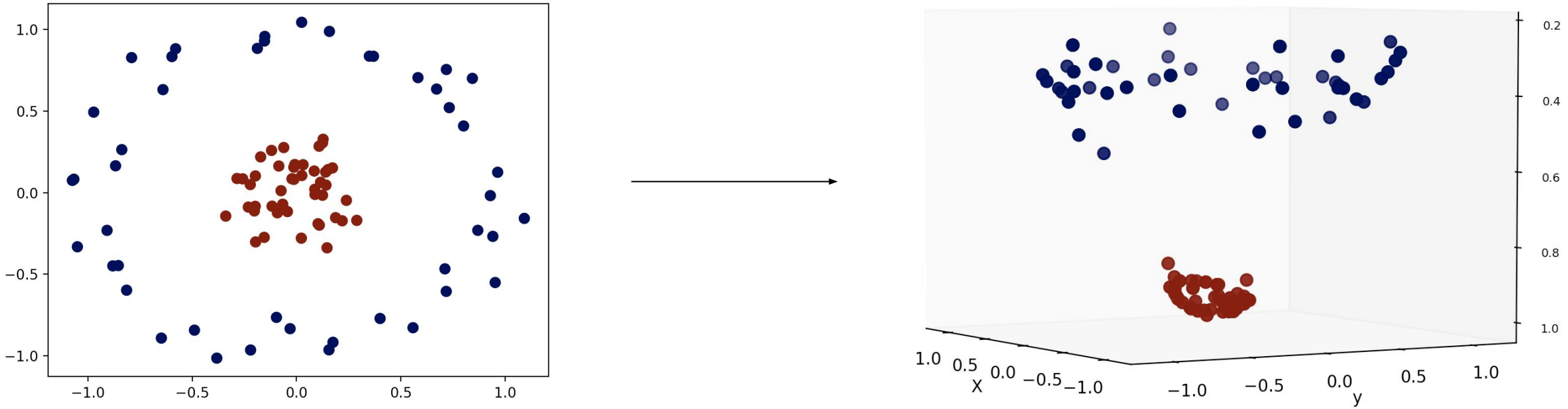


- Data is linear separable in polar coordinates
- The mapping can be expressed as:

$$\psi: \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix} \quad \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

Embeddings into feature space

- Possible solution 2: Map data to higher dimension



- The data is linear separable in 3 dimension
- The mapping can be expressed as:

$$\psi: \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

Embeddings into feature space

- Basic paradigm:
  - Given some domain set $\mathcal{X}$, choose a mapping $\psi : \mathcal{X} \longmapsto \mathcal{F}$, for some feature space $\mathcal{F}$
  - Given a sequence of labeled samples, $S = (\mathrm{X}_1, y_1), \ldots, (\mathrm{X}_m, y_m)$, create the image sequence $\hat{S} = (\psi(\mathrm{X}_1), y_1), \ldots, (\psi(\mathrm{X}_m), y_m)$
  - Train a linear predictor h over $\hat{S}$
  - The prediction of sample X is $h(\psi(\mathrm{X}))$

- Note that:
  - The success of this paradigm depends on the choice of good ψ which can make the data distribution close to being linear separable
  - The choice of ψ often requires prior knowledge about the task.

Embeddings into feature space

- One notable example: Polynomial mappings
  - The mapping can be expressed as $\mathrm{X} \longmapsto p(\mathrm{X})$ where p is a multivariate polynomial of degree *k*
  - First consider the case when X is 1 dimensional. In this case, p can be expressed as

$$p(x) = \sum_{j=0}^{k} w_j x^j$$

    where $w \in \mathbb{R}^{k+1}$ is the coefficients we need to learn
  - Now, we can rewrite

$$p(x) = \langle w, \psi(\mathrm{X}) \rangle$$

    where $\psi: \mathbb{R} \longmapsto \mathbb{R}^{k+1}$ is mapping $x \longmapsto (1, x, x^2, x^3, \dots, x^k)$

Embeddings into feature space

- A more general degree *k* multivariate polynomial from $\mathbb{R}^n$ to $\mathbb{R}^d$ can be written as

$$p(x) = \sum_{J \in [n]^r : r < k} w_J \prod_{i=1}^{r} x_{J_i}$$

- As before, we can rewrite $p(x) = \langle w, \psi(\mathbf{x}) \rangle$
- In general we can choose any feature mapping ψ that maps the original instances into some *Hilbert space.*
- If $\boldsymbol{M}$ is a linear subspace of a Hilbert space, then every $\mathbf{x}$ in the Hilbert space can be written as a sum $\mathbf{x} = \mathbf{u} + \mathbf{v}$ where $\mathbf{u} \in \boldsymbol{M}$ and $\langle \mathbf{v}, \mathbf{w} \rangle = 0$ for all $\mathbf{w} \in \boldsymbol{M}$

Embeddings into feature space

- Hilbert space
  - Hilbert space is a vector space with an inner product, which is also complete.
    - Inner product space:
      - a vector space $\boldsymbol{V}$ over the field $\boldsymbol{F}$ together with an inner product, which is a mapping: $\langle \cdot, \cdot \rangle : \boldsymbol{V} \times \boldsymbol{V} \rightarrow \boldsymbol{F}$
    - Completeness:
      - $f_n(\mathbf{x}) = \sum_{i=1}^{n} \theta_i \psi_i(\mathbf{x}) \approx f(\mathbf{x})$ $when\ n\ in\ large\ enough$
  - If $\boldsymbol{M}$ is a linear subspace of a Hilbert space, then every $\mathbf{x}$ in the Hilbert space can be written as a sum $\mathbf{x} = \mathbf{u} + \mathbf{v}$ where $\mathbf{u} \in \boldsymbol{M}$ and $\langle \mathbf{v}, \mathbf{w} \rangle = 0$ for all $\mathbf{w} \in \boldsymbol{M}$

Embeddings into feature space

- Two problems we must face:
  - VC dimension
    - This problem has been solved in the previous part
  - Performing calculation in higher dimensional space
    - This is where we introduce kernel based learning to help us to address the computational issue

The Kernel Trick

- General Idea
  - Given an embedding ψ of some domain space X into some Hilbert space, we define the kernel function

$$K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$$

  - Remember the dual problem in the previous part

(Dual problem equation)

We need to compute the dot product of two vectors $\mathbf{x}$ and $\mathbf{y}$

  - Kernel is a way of computing the dot product of two vectors $\mathbf{x}$ and $\mathbf{y}$ in some (possibly very high dimensional) feature space.
  - Kernel is useful because kernels give a way to compute dot products in some feature space without even knowing what this space is and what is ψ.

The Kernel Trick

- General Idea
  - It turns out that many learning algorithms for halfspaces can be carried out just on the basis of the values of the kernel function over pairs of domain points.
  - The main advantage of such algorithms is that they implement linear separators in high dimensional feature spaces without having to specify points in that space or expressing the embedding ψ explicitly.

The Kernel Trick

- To see how the kernel can be used in SVM, we revisit the SVM optimization problem we have derived in previous part.
- They can be generalized into following form:

$$\min_{w}(f(\langle w,\psi(\mathrm{X}_1)\rangle,\ldots,\langle w,\psi(\mathrm{X}_m)\rangle)+R(\|w\|)) \quad (*)$$

where f: $\mathbb{R}^m \longmapsto \mathbb{R}$ is an arbitrary function

R: $\mathbb{R}_+ \longmapsto \mathbb{R}$ is a monotonically nondecreasing function.

- For example, Soft-SVM for homogenous halfspaces can be derived from above equation by letting

$$R(a)=\lambda a^2$$

$$f(a_1,\ldots,a_m)=\frac{1}{m}\sum_i \max(0,1-y_i a_i)$$

$$\min_{w}(f(\langle w, \psi(\mathbf{x}_1)\rangle, \ldots, \langle w, \psi(\mathbf{x}_m)\rangle) + R(\|w\|)) \quad (*)$$

The Kernel Trick

Theorem 16.1 Representer Theorem

Assume that ψ is a mapping from X *to a Hilbert space. Then, there exists a vector* $\alpha \in \mathbb{R}^{\mathrm{m}}$ *such that* $\mathrm{w} = \sum_{\mathrm{i}=1}^{\mathrm{m}} \alpha_{\mathrm{i}}\, \psi(\mathbf{x}_{\mathrm{i}})$ *is an optimal solution of Equation (*).*

Proof

Let w^* be the optimal solution to the equation (*) and the solution is in a Hilbert space, we can write w^* as:

$$w^* = \sum_{i=1}^{m} \alpha_i\, \psi(\mathbf{x}_{\mathrm{i}}) + \mathbf{u}$$

where $\langle \mathbf{u}, \psi(\mathbf{x}_{\mathrm{i}})\rangle = 0$

The Kernel Trick

Set $\mathbf{w} = \mathbf{w}^* - \mathbf{u}$, thus $\|\mathbf{w}\| \leq \|\mathbf{w}^*\|$

Since R is nondecreasing we obtain that $\mathrm{R}(\|\mathbf{w}\|) \leq R(\|\mathbf{w}^*\|)$

Additionally, for all *i* we have that

$$\langle \mathbf{w}, \psi(\mathbf{x}_i) \rangle = \langle \mathbf{w}^* - \mathbf{u}, \psi(\mathbf{x}_i) \rangle = \langle \mathbf{w}^*, \psi(\mathbf{x}_i) \rangle$$

Hence

$$f(\langle \mathbf{w}, \psi(\mathbf{x}_1) \rangle, \dots, \langle \mathbf{w}, \psi(\mathbf{x}_m) \rangle) + R(\|\mathbf{w}\|) \leq f(\langle \mathbf{w}^*, \psi(\mathbf{x}) \rangle, \dots, \langle \mathbf{w}^*, \psi(\mathbf{x}_m) \rangle) + R(\|\mathbf{w}^*\|)$$

Therefore $\mathbf{w}$ is also an optimal solution!

$$\min_{w}(f(\langle w, \psi(\mathbf{x}_1)\rangle, \ldots, \langle w, \psi(\mathbf{x}_m)\rangle) + R(\|w\|)) \quad (*)$$

The Kernel Trick

On the basis of the representer theorem we can optimize Equation (*) with respect to the coefficients α instead of the coefficients w. For all i, we can writ

$$\langle w, \psi(\mathbf{x}_i)\rangle = \left\langle \sum_{j=1}^{\mathrm{m}} \alpha_j \, \psi(\mathbf{x}_j), \psi(\mathbf{x}_i) \right\rangle = \sum_{j=1}^{\mathrm{m}} \alpha_j \langle \psi(\mathbf{x}_j), \psi(\mathbf{x}_i)\rangle$$

Similarly,

$$\|w\|^2 = \left\langle \sum_{j} \alpha_j \, \psi(\mathbf{x}_j), \sum_{j} \alpha_j \, \psi(\mathbf{x}_j) \right\rangle = \sum_{i,j=1}^{\mathrm{m}} \alpha_j \alpha_i \, \langle \psi(\mathbf{x}_j), \psi(\mathbf{x}_i)\rangle$$

$$\min_{w}(f(\langle w,\psi(\mathbf{x}_1)\rangle,\ldots,\langle w,\psi(\mathbf{x}_m)\rangle)+R(\|w\|)) \quad (*)$$

The Kernel Trick

Let $K(\mathbf{x},\mathbf{x}')=\langle\psi(\mathbf{x}),\psi(\mathbf{x}')\rangle$, equation(*) can be written as

$$\min_{\alpha\in\mathbb{R}^m} f\left(\sum_{j=1}^{\mathrm{m}}\alpha_j K(\mathbf{x}_j,\mathbf{x}_1),\ldots,\sum_{j=1}^{\mathrm{m}}\alpha_j K(\mathbf{x}_j,\mathbf{x}_m)\right)+R\left(\sqrt{\sum_{i,j=1}^{\mathrm{m}}\alpha_j\alpha_i K(\mathbf{x}_j,\mathbf{x}_m)}\right)$$

The only thing we should know is how to calculate inner products in the feature space, or equivalently, to calculate the kernel function.

In fact, we solely need to know the value of the $m\times m$ matrix G *such that* $G_{i,j}=K(\mathbf{x}_i,\mathbf{x}_j)$ which is often called the *Gram* matrix.

Once we learn the coefficients α we can calculate the prediction on a new instance by

$$\langle\mathbf{w},\psi(\mathbf{x})\rangle=\sum_{j=1}^{\mathrm{m}}\alpha_j K(\mathbf{x}_j,\mathbf{x})$$

The Kernel Trick

- Frequently used kernel: Polynomial kernel
  - The *k* degree polynomial kernel is defined to be $K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k$
  - We need to show that there exist a mapping ψ from the original space to some higher dimensional space for which $K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$

$$K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle) \cdots (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)$$

$$= \left(\sum_{j=0}^{n} x_j x_j' \right) \cdots \left(\sum_{j=0}^{n} x_j x_j' \right) = \sum_{j \in \{0,1,\ldots,n\}^k} \prod_{i=1}^{k} x_{J_i} x_{J_i}'$$

$$= \sum_{j \in \{0,1,\ldots,n\}^k} \prod_{i=1}^{k} x_{J_i} \prod_{i=1}^{k} x_{J_i}'$$

The Kernel Trick

- Frequently used kernel: Gaussian Kernel
  - Let the original instance space be $\mathbb{R}$ and consider the mapping ψ where for each nonnegative integer $n \geq 0$ there exists an element $\psi(x)_n$ that equals $\frac{1}{\sqrt{n!}} e^{-\frac{x^2}{2}} x^n$. Then

$$\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle = \sum_{n=0}^{\infty} \left(\frac{1}{\sqrt{n!}} e^{-\frac{x^2}{2}} x^n\right)\left(\frac{1}{\sqrt{n!}} e^{-\frac{(x')^2}{2}} (x')^n\right)$$

$$= e^{-\frac{x^2+(x')^2}{2}} \sum_{n=0}^{\infty} \left(\frac{(xx')^2}{n!}\right) = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2}}$$

  - Gaussian kernel is defined to be $K(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma}}$

    where σ is a scalar that > 0

The Kernel Trick

- Why kernel can help us address the computational issue?

  (Take the Polynomial kernel with p = 2 as example)

Let's say we have two points: $\mathbf{x} = (x_1, x_2)^T$ and $\mathbf{x}' = (x_1', x_2')^T$

Consider $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T\mathbf{x}')^2 = (1 + x_1x_1' + x_2x_2')^2$

$$= 1 + (x_1x_1')^2 + (x_2x_2')^2 + 2x_1x_1' + 2x_2x_2' + 2x_1x_1'x_2x_2'$$

The result is still a inner product of two vector in higher dimension:

$$(1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$$

$$(1, x_1'^2, x_2'^2, \sqrt{2}x_1', \sqrt{2}x_2', \sqrt{2}x_1'x_2')$$

This doesn't seems to be of much help. Let's generalize this.

The Kernel Trick

- Why kernel can help us address the computational issue?

  (Take the Polynomial kernel with order P)

We have: $\mathbf{x} = (x_1, \dots, x_m)^T$ and $\mathbf{x}' = (x_1', \dots, x_m')^T$

And $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T\mathbf{x}')^P = (1 + x_1x_1' + \cdots + x_mx_m')^P$

The computational complexity has nothing to do with the order P. We just raise a number to the power of P.

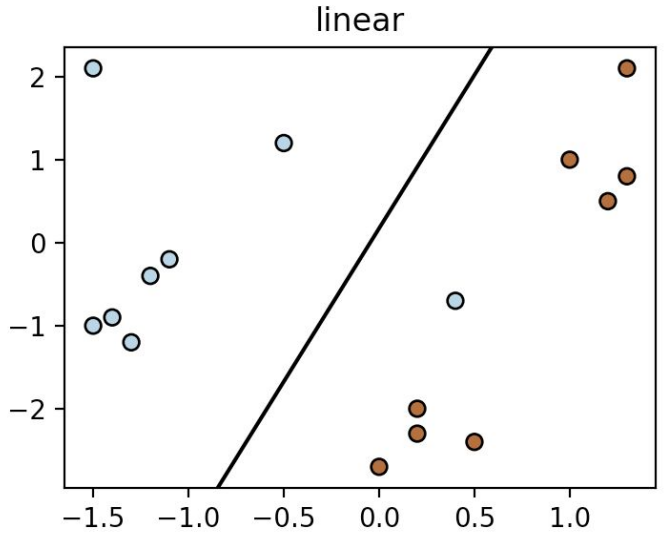
But, what if we expand this to find the mapping function. The resulting vector will be huge and very expansive to compute!

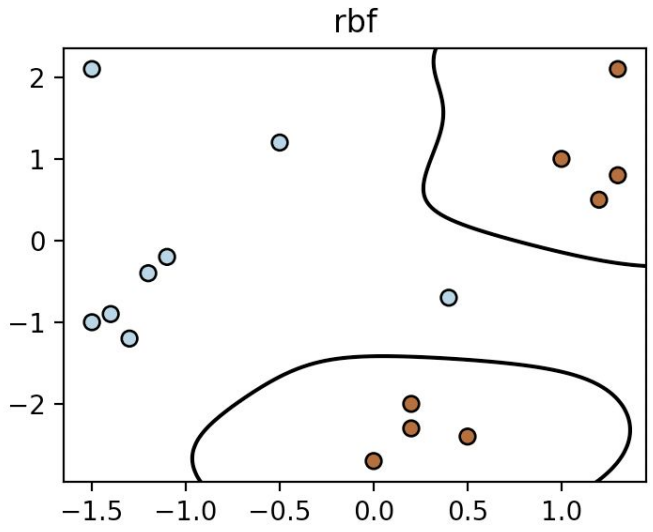
The Kernel Trick

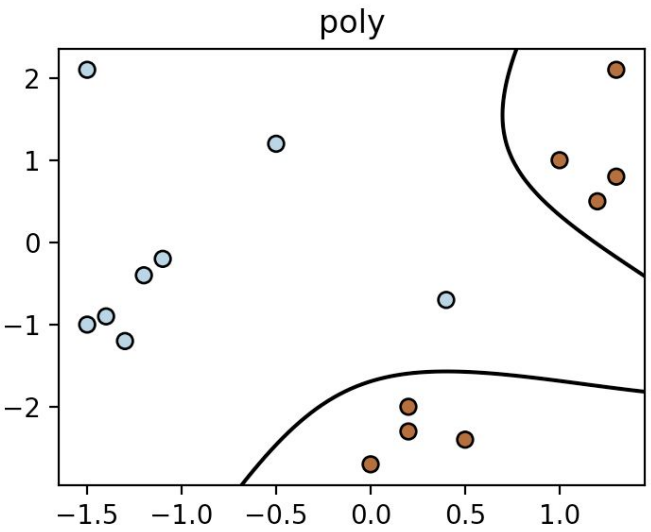
- Kernels as a Way to Express Prior Knowledge
  - One can therefore think of an embedding ψ as a way to express and utilize prior knowledge about the problem at hand
  - The Polynomial kernel
    - Learning a halfspace with a k degree polynomial kernel enables us to learn polynomial predictor of degree k over the original space
  - The Gaussian kernel
    - Intuitively, the Gaussian kernel sets the inner product in the feature space between $\mathbf{x}, \mathbf{x}'$ to be close to zero if the instances are far away from each other (in the original domain) and close to 1 if they are close. Also know as RBF kernel (Radial Basis Function)

The Kernel Trick

- Examples:







The Kernel Trick

Lemma 16.2

A symmetric function $K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ implements an inner product in some Hilbert space if and only if it is positive semidefinite; namely, for all $\mathbf{x}_1, \ldots, \mathbf{x}_m$, the Gram matrix, $G_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$, is a positive semidefinite matrix.

Proof

It is trivial to see that if *K* implements an inner product in some Hilbert space then the Gram matrix is positive semidefinite.

Implementing Soft-SVM With Kernels

SGD for Solving Soft-SVM with Kernels

Goal: Solve Equation (16.5)
parameter: T
Initialize: $\boldsymbol{\beta}^{(1)} = \mathbf{0}$
for $t = 1, \ldots, T$
  Let $\boldsymbol{\alpha}^{(t)} = \frac{1}{\lambda t}\boldsymbol{\beta}^{(t)}$
  Choose i uniformly at random from $[m]$
  For all $j \neq i$ set $\beta_j^{(t+1)} = \beta_j^{(t)}$
  If $(y_i \sum_{j=1}^{m} \alpha_j^{(t)} K(\mathbf{x}_j, \mathbf{x}_i) < 1)$
    Set $\beta_i^{(t+1)} = \beta_i^{(t)} + y_i$
  Else
    Set $\beta_i^{(t+1)} = \beta_i^{(t)}$
Output: $\bar{\mathbf{w}} = \sum_{j=1}^{m} \bar{\alpha}_j \psi(\mathbf{x}_j)$ where $\bar{\boldsymbol{\alpha}} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\alpha}^{(t)}$

$$\mathbf{w}^{(t)} = \frac{1}{\lambda_t}\theta^{(t)}$$

$$\theta^{(t)} = \sum_{j=1}^{m} \beta_j^{(t)} \psi(\mathbf{x}_j) \quad (**)$$

$$\mathbf{w}^{(t)} = \sum_{j=1}^{m} \alpha_j^{(t)} \psi(\mathbf{x}_j)$$

Implementing Soft-SVM With Kernels

- To prove the equation (**) holds, we use simple induction:

$$\theta^{(t+1)} = \theta^{(t)} + y_i \psi(\mathbf{x}_i) = \sum_{j=1}^{m} \beta_j^{(t)} \psi(\mathbf{x}_j) + y_i \psi(\mathbf{x}_i) = \sum_{j=1}^{m} \beta_j^{(t+1)} \psi(\mathbf{x}_j)$$

$$where\ \beta_i^{(t+1)} = \beta_i^{(t)} + y_i$$

$$y_i \left\langle \mathbf{w}^{(t)}, \psi(\mathbf{x}_i) \right\rangle = y_i \left\langle \sum_{j=1}^{m} \alpha_j^{(t)} \psi(\mathbf{x}_j), \psi(\mathbf{x}_i) \right\rangle = y_i \sum_{j=1}^{m} \alpha_j^{(t)} K(\mathbf{x}_j, \mathbf{x}_i)$$

Summary

- By now, we have find an efficient way to tackle down the computational complexity
- The idea is that in order to find a halfspace predictor in the high dimensional space, we do not need to know the representation of the instance in that space
- The kernel trick can be applied in many other algorithm