Machine Learning

Lecture: Features and Kernels

Hiroshi Shimodaira and Hao Tang

2026

Ver. 1.0

Questions you should be able to answer after this week

- Feature transformation
- Meaning of linear classifiers
- Non-linear SVMs
- kernel tricks
- Mercer kernels

How to resolve a linearly non-separable case?

Feature transformation / mapping: $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$

$$h(\boldsymbol{x}) = \begin{cases} -1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} + b < 0 \\ +1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} + b \geq 0 \end{cases} = \operatorname{sgn}(\boldsymbol{w}^\top \boldsymbol{x} + b) \tag{1}$$

$$\downarrow$$

$$h(\boldsymbol{x}) = \begin{cases} -1 & \text{if } \boldsymbol{w}^\top \phi(\boldsymbol{x}) < 0 \\ +1 & \text{if } \boldsymbol{w}^\top \phi(\boldsymbol{x}) \geq 0 \end{cases} = \operatorname{sgn}(\boldsymbol{w}^\top \phi(\boldsymbol{x})) \tag{2}$$

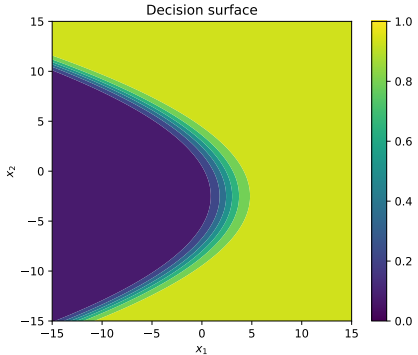
Logistic regression with feature transformation

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{1 + \exp(-y(\boldsymbol{w}^\top \boldsymbol{x} + b))} \tag{3}$$

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{1 + \exp(-y(\boldsymbol{w}^\top \phi(\boldsymbol{x})))} \tag{4}$$

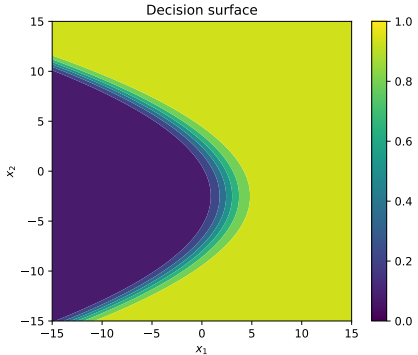
Feature transformation - examples

$$(x_1, x_2) \rightarrow (x_1, x_2, x_2^2)$$

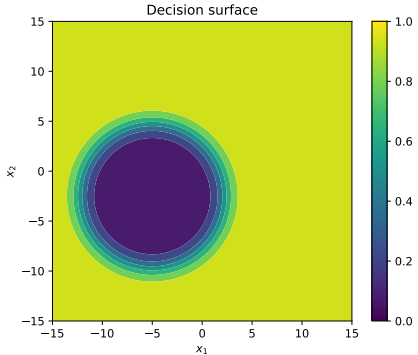


Feature transformation - examples

$(x_1, x_2) \rightarrow (x_1, x_2, x_2^2)$

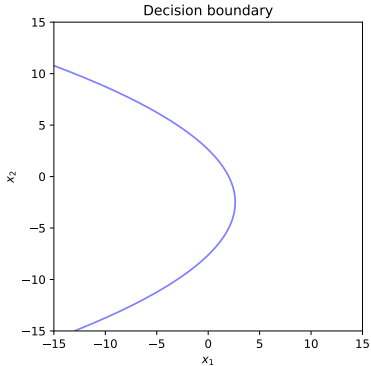


$(x_1, x_2) \rightarrow (x_1, x_2, x_1^2, x_2^2)$

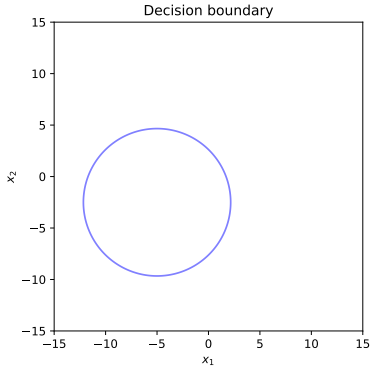


Feature transformation - examples

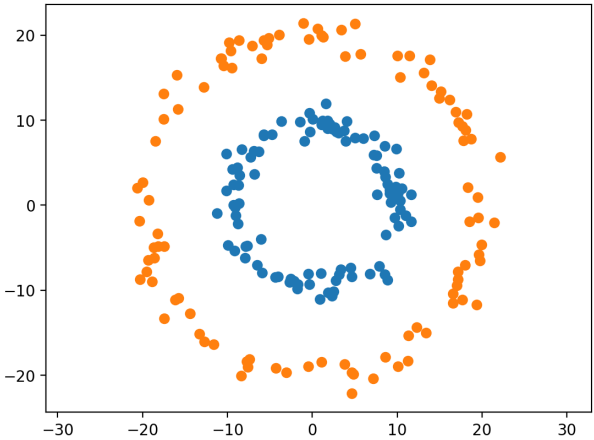
$(x_1, x_2) \rightarrow (x_1, x_2, x_2^2)$



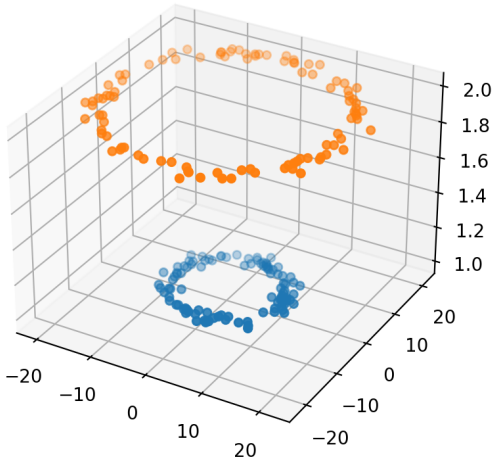
$(x_1, x_2) \rightarrow (x_1, x_2, x_1^2, x_2^2)$



Two-circle example



Two-circle example



What is it meant by linear classifiers?

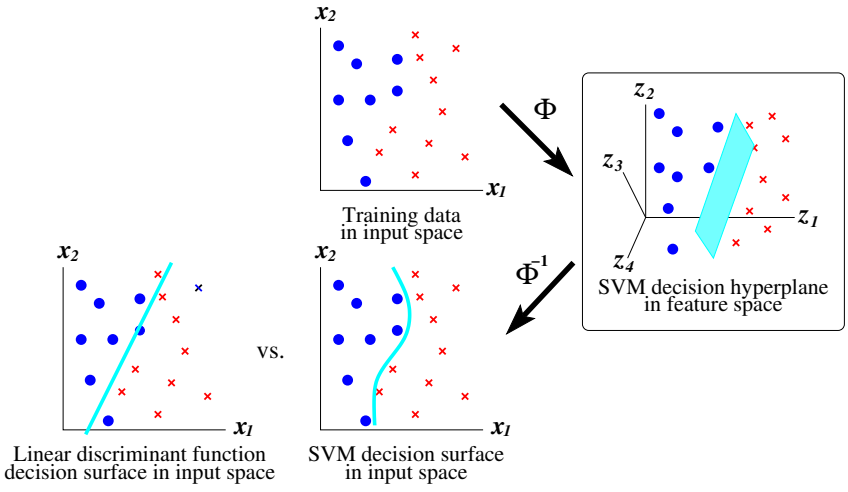
- A linear classifier is linear in the parameters $\boldsymbol{w}$, **not** in the features.
  (i.e., classification based on a linear combination between $\boldsymbol{w}$ and $\boldsymbol{x}$.)

- A linear classifier can have arbitrary nonlinear features.

Should we consider very complex transformation?

- Not necessarily so.
- Complex models may **overfit** the training data and may not **generalise** very well.
- We will come back to this in some lectures later.

Non-linear SVM



Non-linear SVM (*cont.*)

- Conceptual steps to construct a non-linear SVM
  - Step 1 Transform $\boldsymbol{x}$ to $\phi(\boldsymbol{x})$ in a high-dimensional space (feature space)
  - Step 2 Train a SVM in the feature space
  - Step 3 Classify data in the feature space

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i^* \, y_i \, \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}) + w_0$$

Non-linear SVM (*cont.*)

- Conceptual steps to construct a non-linear SVM
  - Step 1 Transform $\boldsymbol{x}$ to $\phi(\boldsymbol{x})$ in a high-dimensional space (feature space)
  - Step 2 Train a SVM in the feature space
  - Step 3 Classify data in the feature space

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i^* \, y_i \, \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}) + w_0$$

- Instead of applying the non-linear transformation and carrying out calculation in the feature space, use a kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ such that

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) \qquad \text{(cf. } \textit{'kernel trick'}\text{)}$$

Non-linear SVM (*cont.*)

- Conceptual steps to construct a non-linear SVM
  - Step 1 Transform $\boldsymbol{x}$ to $\phi(\boldsymbol{x})$ in a high-dimensional space (feature space)
  - Step 2 Train a SVM in the feature space
  - Step 3 Classify data in the feature space

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i^* \, y_i \, \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}) + w_0$$

- Instead of applying the non-linear transformation and carrying out calculation in the feature space, use a kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ such that

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) \qquad \text{(cf. } \textit{'kernel trick'}\text{)}$$

$$L(\boldsymbol{\alpha}, \boldsymbol{\xi}) = -\frac{1}{2} \sum_{i,j=1}^{N} y_i \, y_j \, \alpha_i \, \alpha_j \, k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{N} \alpha_i$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i^* \, y_i \, k(\boldsymbol{x}_i, \boldsymbol{x}) + w_0$$

Kernel functions for SVM

An example of kernel that maps data to a feature space explicitly

$$
\begin{aligned}
k(\mathbf{a}, \mathbf{b}) &\triangleq (1 + \mathbf{a}^T \mathbf{b})^2 = (1 + a_1 b_1 + a_2 b_2)^2 \\
&= 1 + 2a_1 b_1 + 2a_2 b_2 + a_1^2 b_1^2 + 2a_1 b_1 a_2 b_2 + a_2^2 b_2^2 \\
&= (1, \sqrt{2}a_1, \sqrt{2}a_2, a_1^2, \sqrt{2}a_1 a_2, a_2^2)^\top (1, \sqrt{2}b_1, \sqrt{2}b_2, b_1^2, \sqrt{2}b_1 b_2, b_2^2) \\
&= \phi(\mathbf{a})^T \phi(\mathbf{b})
\end{aligned}
$$

Kernel functions for SVM

An example of kernel that maps data to a feature space explicitly

$$
\begin{aligned}
k(\mathbf{a}, \mathbf{b}) &\triangleq (1 + \mathbf{a}^T\mathbf{b})^2 = (1 + a_1 b_1 + a_2 b_2)^2 \\
&= 1 + 2a_1 b_1 + 2a_2 b_2 + a_1^2 b_1^2 + 2a_1 b_1 a_2 b_2 + a_2^2 b_2^2 \\
&= (1, \sqrt{2}a_1, \sqrt{2}a_2, a_1^2, \sqrt{2}a_1 a_2, a_2^2)^\top (1, \sqrt{2}b_1, \sqrt{2}b_2, b_1^2, \sqrt{2}b_1 b_2, b_2^2) \\
&= \phi(\mathbf{a})^T \phi(\mathbf{b})
\end{aligned}
$$

Popular kernels

Kernel	$k(\boldsymbol{x}_i, \boldsymbol{x}_j)$
Polynomial	$(1 + \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)^d$
Radial basis function (RBF)	$\exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$, $\sigma > 0$ (bandwidth)
Hyperbolic tangent	$\tanh(\kappa_1 \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \kappa_2)$, $\kappa_1 > 0, \kappa_2 < 0$

where $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ is an inner product (e.g. dot product) between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Making kernels

How can we ensure if a kernel works as an inner product in a feature space?

It should satisfy:

- $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = \langle \phi(\mathbf{x}'), \phi(\mathbf{x}) \rangle = k(\mathbf{x}', \mathbf{x})$
- $k(\mathbf{x}, \mathbf{x}')^2 \leq k(\mathbf{x}, \mathbf{x})\, k(\mathbf{x}', \mathbf{x}')$

Making kernels

How can we ensure if a kernel works as an inner product in a feature space?

It should satisfy:

- $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = \langle \phi(\mathbf{x}'), \phi(\mathbf{x}) \rangle = k(\mathbf{x}', \mathbf{x})$
- $k(\mathbf{x}, \mathbf{x}')^2 \leq k(\mathbf{x}, \mathbf{x})\, k(\mathbf{x}', \mathbf{x}')$
- The Gram matrix $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))$, which is a N-by-N matrix, is positive definite.

$\rightarrow$ Mercer kernels or positive-definite kernel

Making kernels

How can we ensure if a kernel works as an inner product in a feature space?

It should satisfy:

- $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = \langle \phi(\mathbf{x}'), \phi(\mathbf{x}) \rangle = k(\mathbf{x}', \mathbf{x})$
- $k(\mathbf{x}, \mathbf{x}')^2 \leq k(\mathbf{x}, \mathbf{x})\, k(\mathbf{x}', \mathbf{x}')$
- The Gram matrix $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))$, which is a N-by-N matrix, is positive definite.

$\rightarrow$ Mercer kernels or positive-definite kernel

$$\mathbf{K} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ & \vdots & \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix} \tag{5}$$

$$\boldsymbol{a}^T \mathbf{K} \boldsymbol{a} > 0 \quad \forall \boldsymbol{a} \in \mathbb{R}^N \setminus \{\mathbf{0}\} \tag{6}$$

$$\text{i.e.,} \quad \sum_{i=1}^{N} \sum_{j=1}^{N} a_i\, a_j\, k(\boldsymbol{x}_i, \boldsymbol{x}_j) > 0 \quad \forall a_i \in \mathbb{R} \setminus \{0\} \tag{7}$$

Mercer's theorem

Suppose k is a continuous symmetric non-negative definite kernel (i.e. Mercer kernel), then k can be expressed as:

$$k(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{\infty} \lambda_i \, \phi_i(\mathbf{x}) \, \phi_i(\mathbf{z})$$

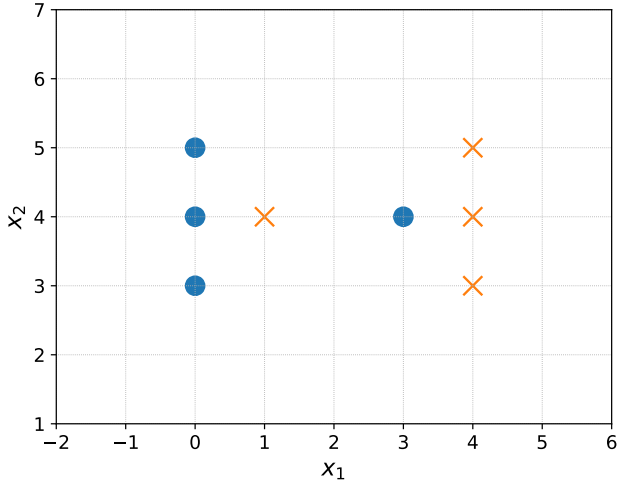
where $\{\phi_i\}$ are eigen-functions, $\|\phi_i\| = 1$, and $\{\lambda_i\}$ are positive eigenvalues $\lambda_i > 0$.

Making kernels from kernels

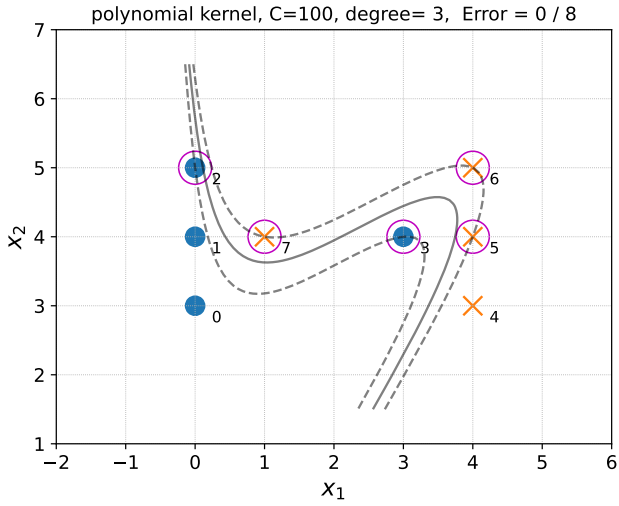
Letting k_1, k_2, and k_3 are kernels, we can create a new kernel k.

- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$
- $k(\mathbf{x}, \mathbf{x}') = a k_1(\mathbf{x}, \mathbf{x}'),\ a > 0$
- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')\, k_2(\mathbf{x}, \mathbf{x}')$
- $k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}')$
- $k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}'))$
- $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T B \mathbf{x}'$, where B is a N-by-N positive-definite matrix

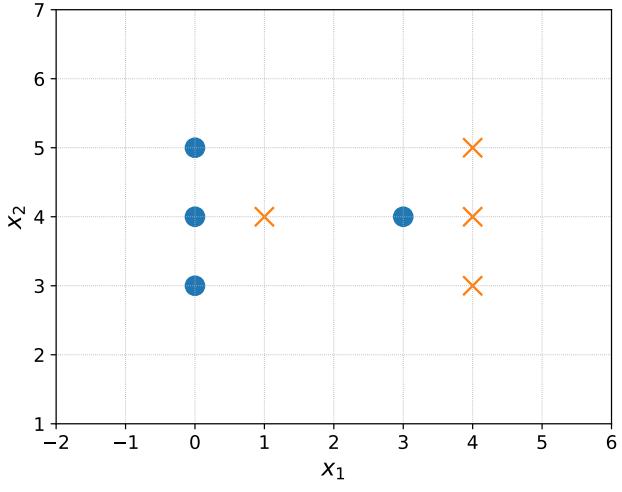
Example – SVM with a polynomial kernel



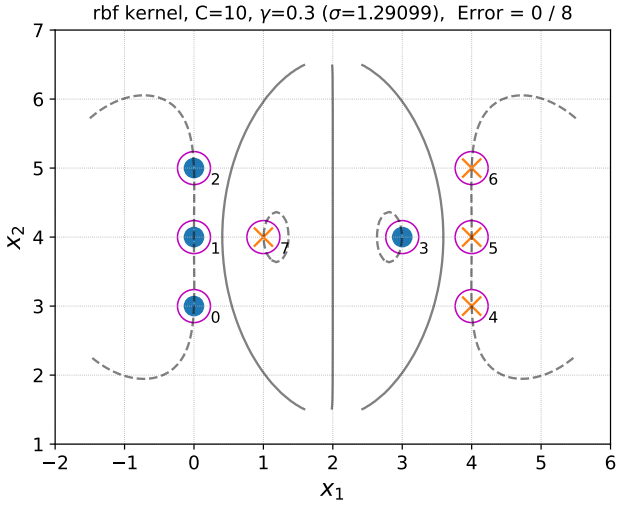
Example – SVM with a polynomial kernel



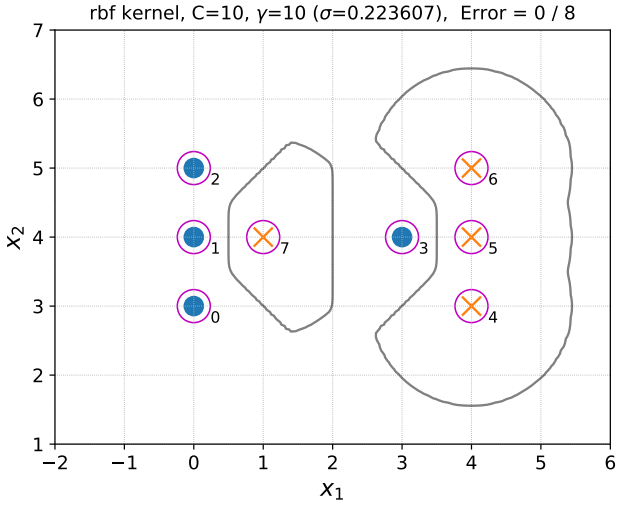
Example – SVM with an RBF kernel



Example – SVM with an RBF kernel



Example – SVM with an RBF kernel



Quizzes

- Considering a kernel $k(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}^T \boldsymbol{b})^2$, find the corresponding transformation function $\phi(\boldsymbol{a})$ when $\boldsymbol{a} \in \mathbb{R}^2$.

- Discuss how the dimensionality of the input vectors and the size of the training set affect the time and memory required to train a linear SVM. Do you expect the same for an SVM with the RBF kernel?

- How many dimensions does the feature space induced by the RBF kernel have? Explain your answer.