KYOTO UNIVERSITY

Statistical Learning Theory
- Kernel Methods -

Hisashi Kashima

DEPARTMENT OF INTELLIGENCE SCIENCE AND TECHNOLOGY

Kernel Methods

Nonlinear regression: Introducing nonlinearity in linear models

- In the previous lecture, we saw several ways to introduce non-linearity into linear models by introducing nonlinear basis functions:

1. Transformed features:
   – e.g. $x \rightarrow \log x$
2. Cross terms:
   – e.g. $x_1, x_2 \rightarrow x_1 x_2$
3. **Kernel methods**:
   – $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$ (nonlinear mapping to a high-dimensional space)

Dual form of ridge regression: Parameters as a linear combination of input vectors

- Ridge regression:
  - Prediction model: $y = \mathbf{w}^\top \mathbf{x}$
  - Optimization problem: $L(\mathbf{w}) = \sum_{i=1}^{N} \left(y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)} \right)^2 + \lambda \|\mathbf{w}\|_2^2$
- Now, we *assume* that the parameter can be represented as a linear combination of the input vectors: $\mathbf{w} = \sum_{i=1}^{N} \alpha_i \mathbf{x}^{(i)}$
  - Prediction model: $y = \sum_{i=1}^{N} \alpha_i \mathbf{x}^{(i)^\top} \mathbf{x}$
  - Optimization problem:

    $$L(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \left(y^{(i)} - \sum_{j=1}^{N} \alpha_j \mathbf{x}^{(i)^\top} \mathbf{x}^{(j)} \right)^2 + \lambda \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \mathbf{x}^{(i)^\top} \mathbf{x}^{(j)}$$
  - Note that $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_N)$ is the model parameter now

Kernel ridge regression: Ridge regression using kernel function (inner product)

- Observation: inputs are always accessed through inner product
  - Prediction model: $y = \sum_{i=1}^{N} \alpha_i \, \mathbf{x}^{(i)\top} \mathbf{x}$
  - Optimization problem:

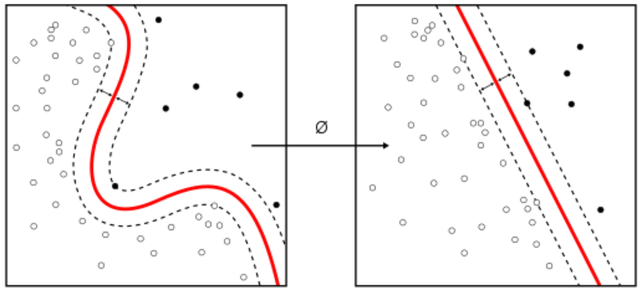
$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{N}\left(y^{(i)} - \sum_{j=1}^{N} \alpha_j \mathbf{x}^{(j)\top} \mathbf{x}^{(i)}\right)^2 + \lambda \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \mathbf{x}^{(i)\top} \mathbf{x}^{(j)}$$

- Kernel ridge regression:
  - The inner product is called *kernel function*
  - Prediction model: $y = \sum_{i=1}^{N} \alpha_i \, K\left(\mathbf{x}^{(i)}, \mathbf{x}\right)$
  - Optimization problem:

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{N}\left(y^{(i)} - \sum_{j=1}^{N} \alpha_j K\left(\mathbf{x}^{(j)}, \mathbf{x}^{(i)}\right)\right)^2 + \lambda \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)$$

Advantage of kernel function: Introducing non-linearity in linear models

- Consider a (nonlinear) mapping $\boldsymbol{\phi}: \Re^D \to \Re^{D'}$
  - D-dimensional space to $D'(\gg D)$-dimensional space
  - Vector $\mathbf{x}$ is mapped to a high-dimensional vector $\boldsymbol{\phi}(\mathbf{x})$
  - A linear model $y = \mathbf{w}'^{\top} \boldsymbol{\phi}(\mathbf{x})$ in the D'-dimensional space is a non-linear model in the original D-dimensional space



- Define kernel $K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = \boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right)^{\top} \boldsymbol{\phi}(\mathbf{x}^{(j)})$ as the inner product in the D'-dimensional space

Advantage of kernel methods: Computationally efficient (when D' is large)

- Kernel function: the inner product of two mapped input vectors

$$K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = \boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right)^{\top} \boldsymbol{\phi}\left(\mathbf{x}^{(j)}\right)$$

- Usually, computational cost of K should depend on D'
  - D' can be high-dimensional (possibly infinite dimensional)
- But, if we can somehow compute $\boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right)^{\top} \boldsymbol{\phi}\left(\mathbf{x}^{(j)}\right)$ in time depending on D, the dimension of $\boldsymbol{\phi}$ does not matter
- Size of the model and optimization problem:

$$D'(\text{number of dimensions}) \to N(\text{number of data})$$

  - Advantageous when D' is very large or infinite

Example of efficiently computable kernel functions: Polynomial kernel can consider high-order cross terms

- Cross-term features: Not only the original features $x_1, x_2, \ldots, x_D$, we use their cross terms (e.g. $x_1 x_2$ and $x_1 x_2 x_3$)
  - If we consider M-th order cross terms, we have $\mathrm{O}(D^M)$ terms
- Polynomial kernel: $K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = \left({\mathbf{x}^{(i)}}^{\top} \mathbf{x}^{(j)} + c\right)^M$
  - E.g. when $c = 0, M = 2, D = 2,$

$$\mathbf{x}^{(i)} = \begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \end{pmatrix}$$

$$K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = \left(x_1^{(i)} x_1^{(j)} + x_2^{(i)} x_2^{(j)}\right)^2$$

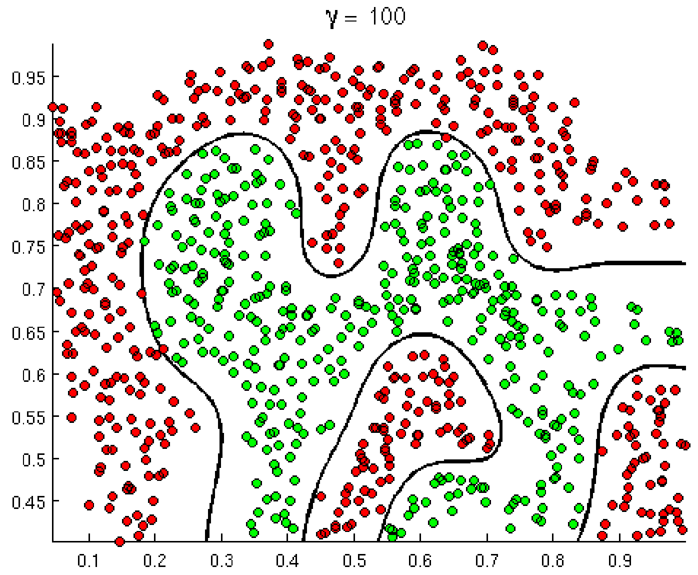
$$= \left({x_1^{(i)}}^2, {x_2^{(i)}}^2, \sqrt{2} x_1^{(i)} x_2^{(i)}\right)^{\top} \left({x_1^{(j)}}^2, {x_2^{(j)}}^2, \sqrt{2} x_1^{(j)} x_2^{(j)}\right)$$

  - Note that it can be computed in $\mathrm{O}(D)$

Example of efficiently computable kernel functions: Gaussian kernel has infinite dimensional feature space

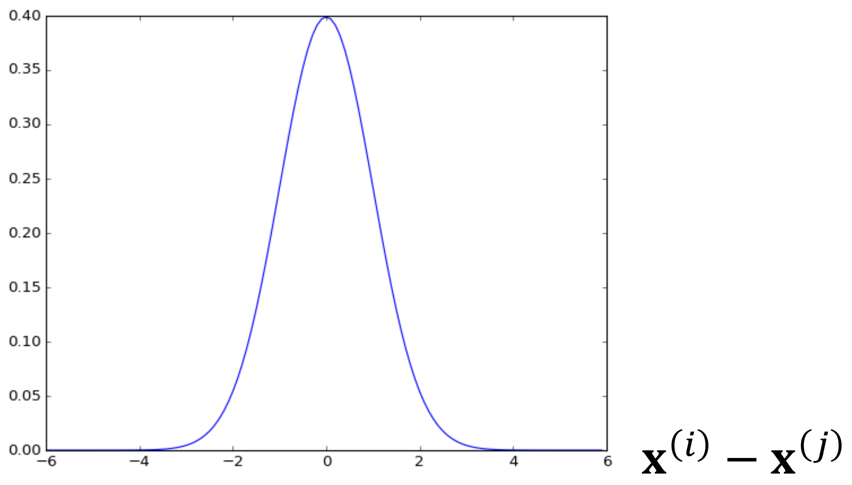
- Gaussian kernel (RBF kernel): $K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2}{\sigma}\right)$
  - Can be interpreted as an inner product in an infinite-dimensional space

Discrimination surface with Gaussian kernel



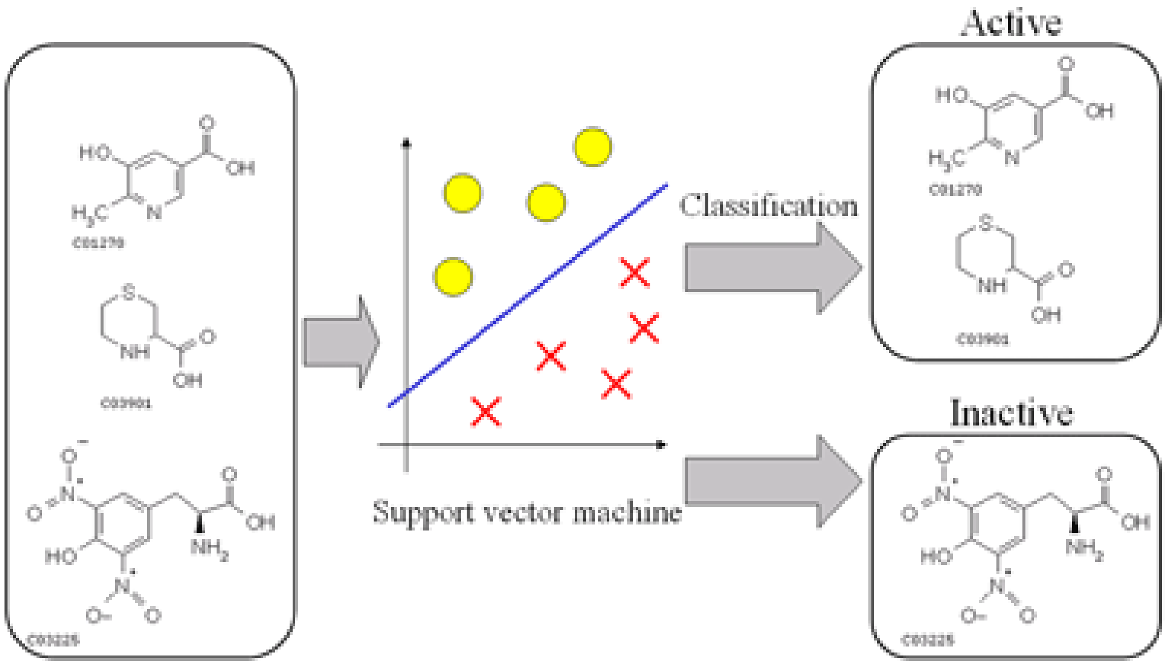
http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex8/ex8.html

1-d Gaussian kernel (RBF kernel)



Kernel methods for non-vectorial data: Kernels for sequences, trees, and graphs

- Kernel methods can handle any kinds of objects (even non-vectorial objects)
  - as far as efficiently computable kernel functions are available
  - Kernels for strings, trees, and graphs, …



http://www.bic.kyoto-u.ac.jp/coe/img/akutsu_fig_e_02.gif

Representer theorem: Theoretical underpinning of kernel methods

- Kernel methods rely on the assumption that the parameter is represented as a linear combination of input vectors:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \mathbf{x}^{(i)}$$

  – … but is this theoretically allowed?

- *Representer theorem* guarantees this, when
  – Loss ℓ for i-th data depends on $\mathbf{w}$ only through $\mathbf{w}^\top \mathbf{x}^{(i)}$
  – L2-regularizer is used

e.g. Squared loss: $\left(y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)}\right)^2$

(Simple) proof of representer theorem: Obj. func. depends only on linear combination of inputs

- Objective function: $L(\mathbf{w}) = \sum_{i=1}^{N} \ell\left(\mathbf{w}^\top \mathbf{x}^{(i)}\right) + \lambda \|\mathbf{w}\|_2^2$
- Divide the optimal parameter $\mathbf{w}^*$ into two parts $\mathbf{w} + \mathbf{w}^\perp$:
  - $\mathbf{w}$: Linear combination of input data $\left\{\mathbf{x}^{(i)}\right\}_i$
  - $\mathbf{w}^\perp$: Other parts (orthogonal to all input data $\left\{\mathbf{x}^{(i)}\right\}$)
- $L(\mathbf{w}^*)$ depends only on $\mathbf{w}$: $\sum_{i=1}^{N} \ell\left(\mathbf{w}^{*\top} \mathbf{x}^{(i)}\right) + \lambda \|\mathbf{w}^*\|_2^2$

$$= \sum_{i=1}^{N} \ell\left(\mathbf{w}^\top \mathbf{x}^{(i)} + \underbrace{\mathbf{w}^{\perp\top} \mathbf{x}^{(i)}}_{=0}\right) + \lambda\left(\|\mathbf{w}\|_2^2 + 2\underbrace{\mathbf{w}^\top \mathbf{w}^\perp}_{=0} + \underbrace{\|\mathbf{w}^\perp\|_2^2}_{\text{Minimized to } = 0}\right)$$

Kernel logistic regression:
Kernel-based nonlinear classification model

- We can also “kernelize” the logistic regression

- Kernel logistic regression model:

$$f(y=1|\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^\top\mathbf{x})} = \frac{1}{1+\exp\left(-\sum_{i=1}^{N}\alpha_i K\left(\mathbf{x}^{(i)},\mathbf{x}\right)\right)}$$

- Objective function of kernel (regularized) logistic regression:

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{N}\ln\left(1+\exp\left(-y^{(i)}\sum_{j=1}^{N}\alpha_j K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)\right)\right)$$
$$+\lambda\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j K\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)$$