機械学習特論

~ 理論とアルゴリズム ~

(Kernel Methods)

講師：西郷浩人

Today' contents

- Support Vector Machines
  - Linear
  - Nonlinear
- Gaussian Process

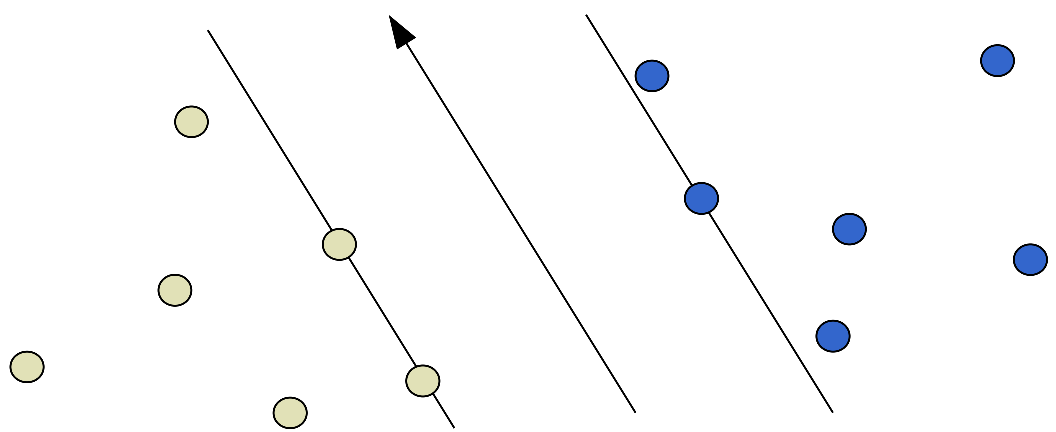
Linear Support Vector Machines (separable case)

Classification by maximizing margin between two classes

- We aim at constructing the following linear model for classification;

$$f(\boldsymbol{x}) = \boldsymbol{w}'\boldsymbol{x} + b$$

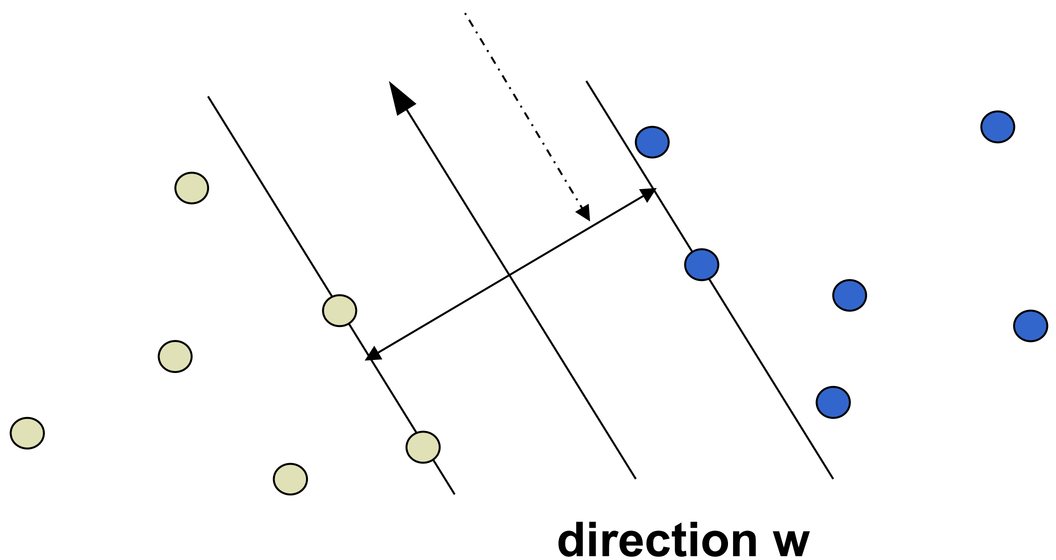
  - class 1 if f(**x**) >= 1
  - class -1 if f(**x**) <= -1



Classification by maximizing margin between two classes

- $$\max_w \frac{1}{\|w\|^2}$$
  $$s.t.\ \boldsymbol{w}'\boldsymbol{x}_i + b \geq 1 \quad \forall\{i|y_i = 1\}$$
  $$\boldsymbol{w}'\boldsymbol{x}_i + b \leq -1 \quad \forall\{i|y_i = -1\}$$

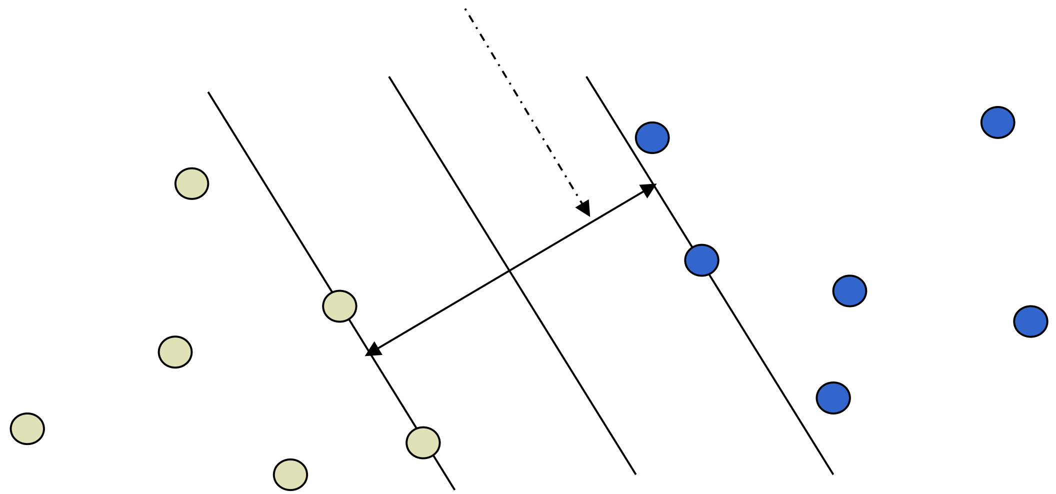
1/||w|| is a margin between two classes



Reforming objective function (hard margin SVM)

- $$\max_w \frac{1}{\|w\|^2}$$
  $$s.t.\ \boldsymbol{w}'\boldsymbol{x}_i + b \geq 1 \quad \forall\{i|y_i = 1\}$$
  $$\boldsymbol{w}'\boldsymbol{x}_i + b \leq -1 \quad \forall\{i|y_i = -1\}$$
  $$\Rightarrow$$
  $$\min_{\boldsymbol{w}} \|w\|^2$$
  $$s.t.\ y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1 \quad \forall i$$

Margin 1/||w|| between two classes



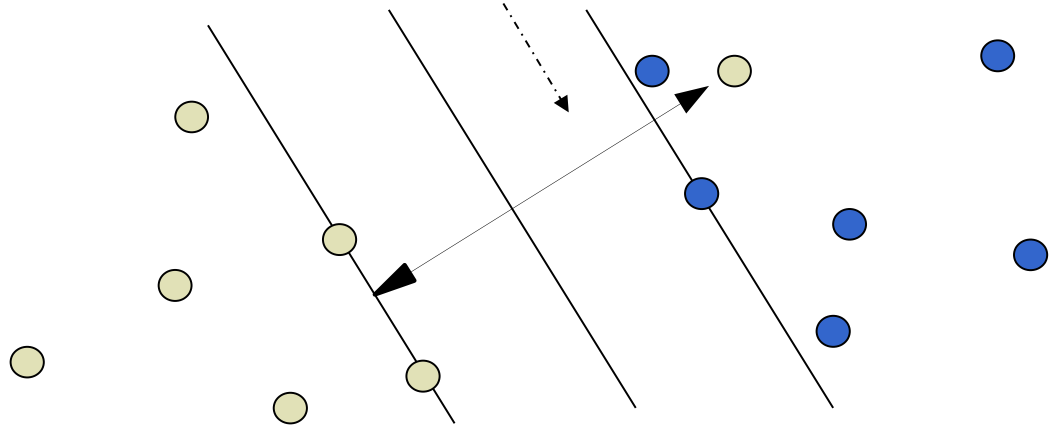
Linear Support Vector Machines (non-separable case)

Support Vector Machine (SVM)

$$min_{\boldsymbol{w}} \|w\|^2$$
$$s.t.\ y_i(\boldsymbol{w}'\boldsymbol{x_i} + b) \geq 1$$

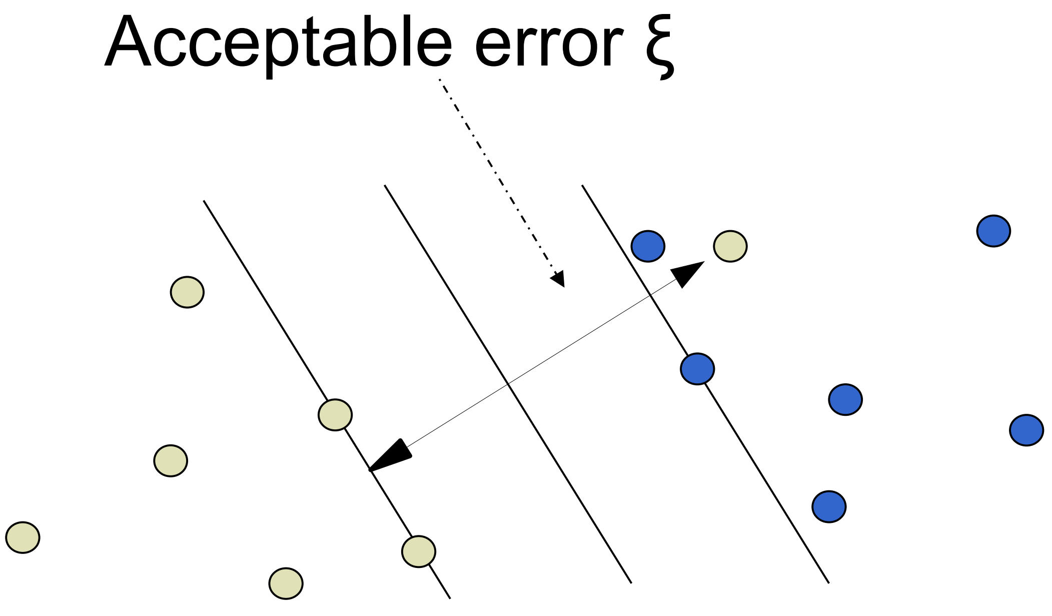
$$min_{\boldsymbol{w}} \|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i$$
$$s.t.\ y_i(\boldsymbol{w}'\boldsymbol{x_i} + b) \geq 1 - \xi_i \quad \forall i$$
$$\xi_i \geq 0 \quad \forall i$$

Acceptable error ξ



SVM primal form（主問題）

$$\min_{\boldsymbol{w}} \|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i$$
$$s.t.\ y_i(\boldsymbol{w}'\boldsymbol{x_i} + b) \geq 1 - \xi_i \quad \forall i$$
$$\xi_i \geq 0 \quad \forall i$$



Primal and Dual problem

- The following two problems are equivalent
  - Find x such that it minimizes z (primal problem)
  - Find y such that it maximizes z (dual problem)
- One can then solve an easier one.

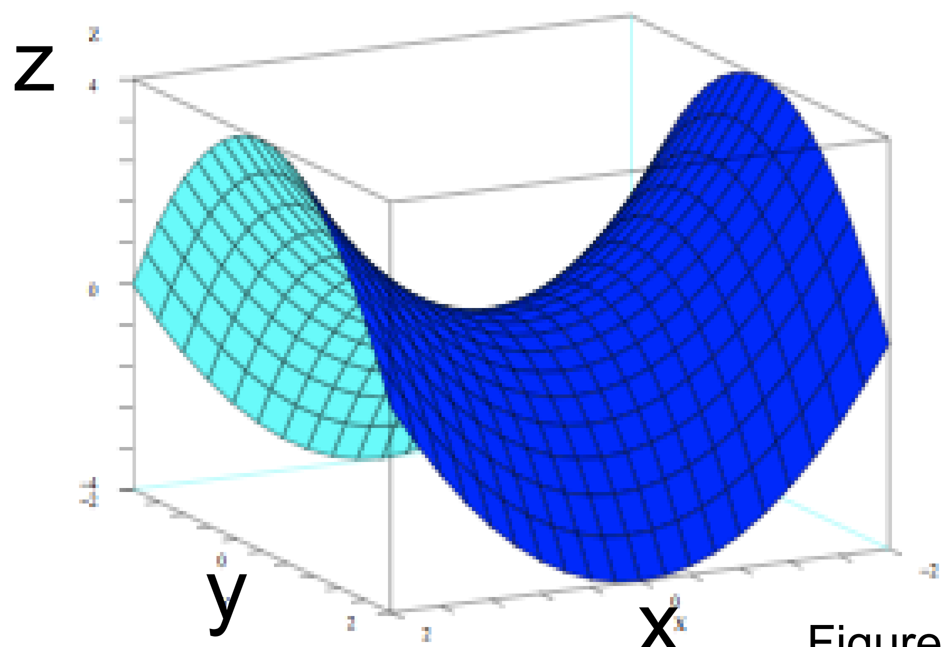


Figure made by Jean-Phillipe Vert

Example: Primal and Dual Problem

- Solution to OLS problem $X\beta = y$ is obtained as

$$\beta = (X'X)^{-1} X' y$$

  - It requires inverting p x p covariance matrix, hence its computational complexity is $O(p^3)$
- Solution to the equivalent dual problem $XX'\alpha = y$ is obtained as $\alpha = (XX')^{-1} y$
  - It requires inverting n x n gram matrix, hence its computational complexity is $O(n^3)$
- Given n < p, dual problem is more economic.
- In OLS case, after solving dual problem, one can recover the primal solution by $\beta = X'\alpha$

Primal and Dual SVM

- Primal

$$min_{\boldsymbol{w}} \|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i$$
$$s.t.\ y_i(\boldsymbol{w}'\boldsymbol{x_i} + b) \geq 1 - \xi_i \quad \forall i$$
$$\xi_i \geq 0 \quad \forall i$$

- Dual

$$max_{\boldsymbol{\alpha}} - \sum_{i=1} \alpha_i + \frac{1}{2}(\boldsymbol{\alpha}\,.\,\boldsymbol{y})'\boldsymbol{X}\,\boldsymbol{X}'(\boldsymbol{\alpha}\,.\,\boldsymbol{y})$$
$$s.t. \sum_{i=1} \alpha_i\, y_i = 0 \quad \forall i\,,$$
$$0 \leq \alpha_i \leq \lambda \quad \forall i$$

Deriving the dual form

$$\min_{\boldsymbol{w}} \|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i$$
$$s.t.\ y_i(\boldsymbol{w}'\boldsymbol{x_i} + b) \geq 1 - \xi_i \quad \forall i$$
$$\xi_i \geq 0 \quad \forall i$$

- Let L be the our new objective function including constraints.
  - α and β are Lagrange multipliers.

$$L = \|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left(y_i (\boldsymbol{w}'\boldsymbol{x_i} + b) - 1 + \xi_i \right) - \sum_{i=1}^{n} \beta_i \xi_i$$

Deriving the dual form

$$L = \|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\boldsymbol{w}' \boldsymbol{x_i} + b \right) - 1 + \xi_i \right) - \sum_{i=1}^{n} \beta_i \xi_i$$

- Taking derivatives with respect to variables, then setting them to zeros obtains

$$\frac{\partial L}{\partial \boldsymbol{w}} = 2\boldsymbol{w} - \boldsymbol{X}'(\boldsymbol{\alpha} . \boldsymbol{y}) = 0 \rightarrow \boldsymbol{w} = \frac{1}{2} \boldsymbol{X}'(\boldsymbol{\alpha} . \boldsymbol{y})$$

$$\frac{\partial L}{\partial \boldsymbol{\xi}} = \lambda \mathbf{1} - \boldsymbol{\alpha} - \boldsymbol{\beta} = 0 \rightarrow \alpha + \beta = \lambda \mathbf{1} \rightarrow 0 \leq \boldsymbol{\alpha} \leq \lambda \mathbf{1} \left(since\, \alpha \geq 0, \beta \geq 0 \right)$$

$$\frac{\partial L}{\partial b} = \boldsymbol{\alpha}' \boldsymbol{y} = 0$$

Deriving the dual form

$$\boldsymbol{w} = \frac{1}{2} \boldsymbol{X}'(\boldsymbol{\alpha}.\boldsymbol{y}) \qquad \boldsymbol{\alpha}'\boldsymbol{y} = 0$$

- Substituting the above equations back to L gets

$$L = \|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left(y_i (\boldsymbol{w}'\boldsymbol{x}_i + b) - 1 + \xi_i \right) - \sum_{i=1}^{n} \beta_i \xi_i$$

$$= \frac{1}{4} (\boldsymbol{\alpha}.\boldsymbol{y})' \boldsymbol{X} \boldsymbol{X}' (\boldsymbol{\alpha}.\boldsymbol{y}) + \lambda \mathbf{1}' \boldsymbol{\xi} - \frac{1}{2} (\boldsymbol{\alpha}.\boldsymbol{y})' \boldsymbol{X} \boldsymbol{X}' (\boldsymbol{\alpha}.\boldsymbol{y})$$

$$- (\boldsymbol{\alpha}.\boldsymbol{y}) b + \mathbf{1}'\boldsymbol{\alpha} + \boldsymbol{\alpha}'\boldsymbol{\xi} - (\lambda \mathbf{1} - \boldsymbol{\alpha})' \boldsymbol{\xi}$$

$$= \mathbf{1}'\boldsymbol{\alpha} - \frac{1}{2} (\boldsymbol{\alpha}.\boldsymbol{y})' \boldsymbol{X} \boldsymbol{X}' (\boldsymbol{\alpha}.\boldsymbol{y})$$

Deriving dual form

$$L = \mathbf{1}'\boldsymbol{\alpha} - \frac{1}{2}(\boldsymbol{\alpha}.\boldsymbol{y})'\boldsymbol{X}\boldsymbol{X}'(\boldsymbol{\alpha}.\boldsymbol{y})$$

$$\boldsymbol{\alpha}'\boldsymbol{y} = 0 \qquad\qquad 0 \leq \boldsymbol{\alpha} \leq \lambda\mathbf{1}$$

- Re-writing the above equations obtains the following dual form.

$$max_{\boldsymbol{\alpha}} - \sum_{i=1} \alpha_i + \frac{1}{2}(\boldsymbol{\alpha}.\boldsymbol{y})'\boldsymbol{X}\boldsymbol{X}'(\boldsymbol{\alpha}.\boldsymbol{y})$$

$$s.t. \sum_{i=1} \alpha_i y_i = 0 \quad \forall i,$$

$$0 \leq \alpha_i \leq \lambda \quad \forall i$$

Exercise 1: Linear SVM

- Download today's data
- Try svm_linear.m
- Regularization parameter λ is an input.
  - Usage: svm_linear(λ)
- Try different λ such as 0.1, 1, 10, 100, 1000. What do you observe ? Watch the datapoint on the left.

- The examples and codes are prepared by Andrew Ng.
  - https://www.coursera.org/learn/machine-learning/

Nonlinear Support Vector Machines and Kernels

The VC(Vapnik-Chervonenkis)-dimension and SVM

- The idea of non-linear SVM is to let dimension high until data points are classified.
  - Regularization is controlled by a parameter.
- The theory of VC-dimension tells us that n data points can be split into two classes by n-1 dimensional hyperplane. (see examples below)

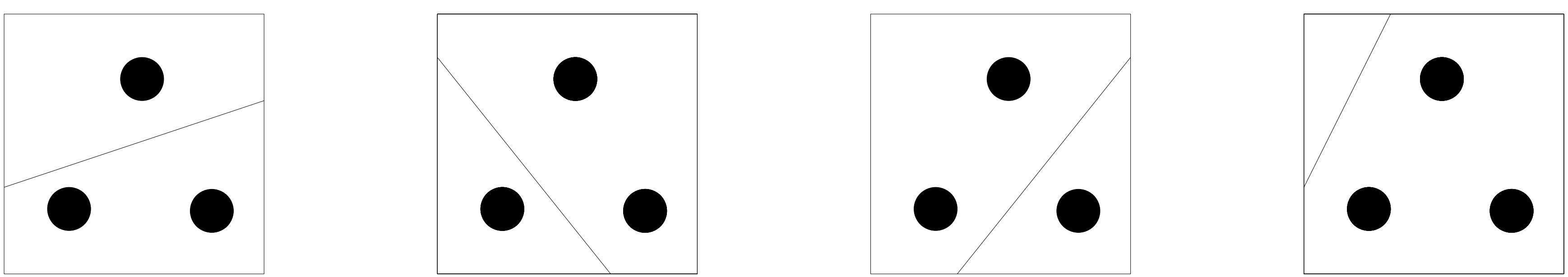
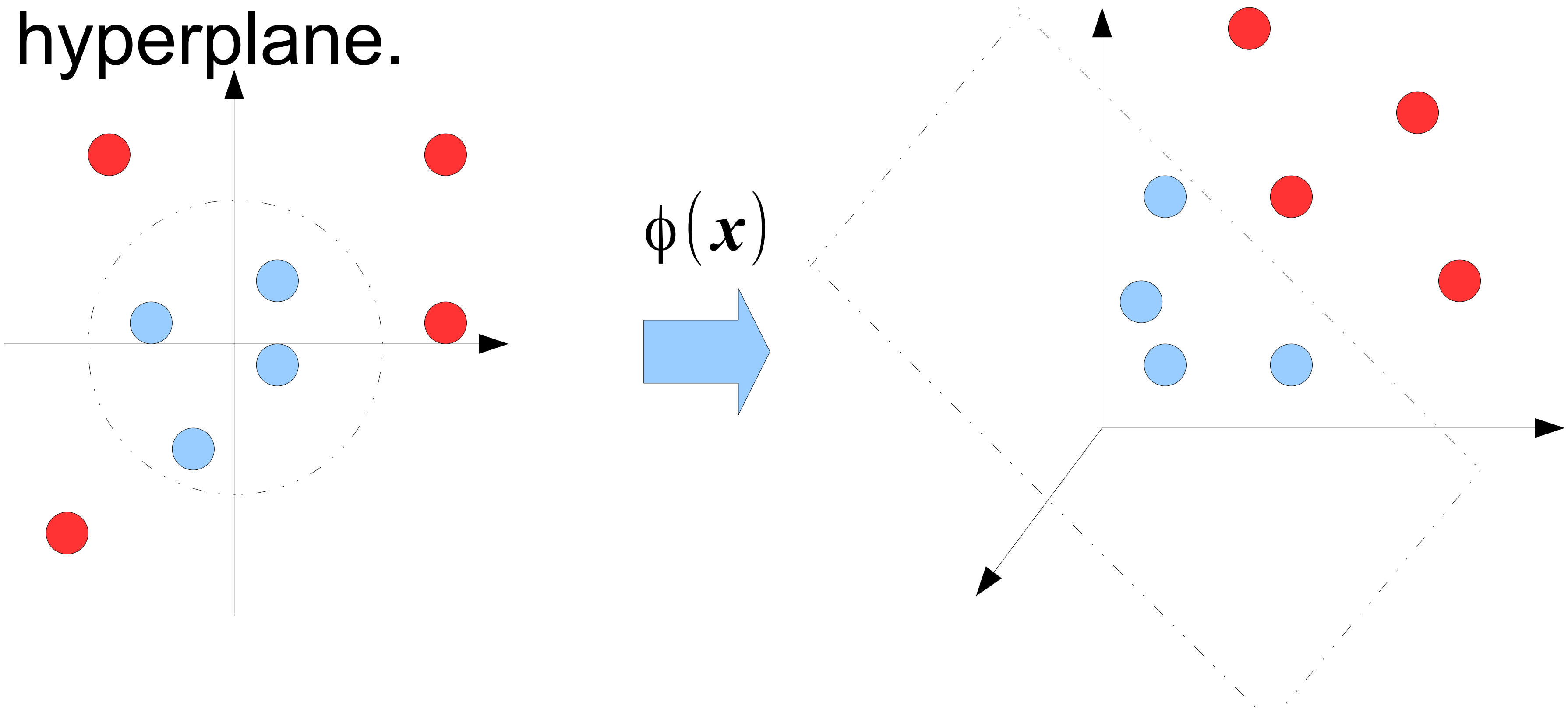


Illustration of mapping to higher dimension

- A mapping function from 2D to 3D
- We desire higher dimensional space where data points are separated by a linear hyperplane.



Dual form SVM and kernels

- $$max_{\boldsymbol{\alpha}} - \sum_{i=1} \alpha_i + \frac{1}{2} (\boldsymbol{\alpha} \, . \, \boldsymbol{y})' \underline{X \, X'} (\boldsymbol{\alpha} \, . \, \boldsymbol{y})$$
$$s.t. \sum_{i=1} \alpha_i \, y_i = 0 \quad \forall i \, ,$$
$$0 \leq \alpha_i \leq \lambda \quad \forall i$$
- **X X'** is an n x n matrix called kernel matrix.
$$\boldsymbol{K} = X \, X'$$
- (i,j)-th element of **K** is a dot product of feature vectors of i-th data and j-th data.
$$K(i, j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$
where x_i denotes i-th row of **X**.
- Kernel matrix is symmetric and positive definite.

About kernels

$$K(i, j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$

- Kernels can be thought of as similarity between two objects.
- One can replace the dot product of x_i and x_j by some non-linear function $\phi()$

$$K(i, j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$$

- As is clear from the formulation, explicit computation of $\phi()$ is not necessary, but only its dot product. (kernel trick)
- Thanks to kernel trick, objects with different sizes can be easily compared
  - e.g., strings, trees, sequences and graphs.

Example: kernel trick for polynomial kernel

- Suppose our data consists of 2 samples and 2 features.

$$\boldsymbol{X}=\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \begin{matrix} \leftarrow \boldsymbol{x_1} \\ \leftarrow \boldsymbol{x_2} \end{matrix}$$

- Consider polynomial function (degree 2) that maps 2-dimensional features into 3-dimensions.

$$\phi(\boldsymbol{x_1})=\{x_{11}^2, x_{12}^2, \sqrt{2}\,x_{11}x_{12}\} \qquad \phi(\boldsymbol{x_2})=\{x_{21}^2, x_{22}^2, \sqrt{2}\,x_{21}x_{22}\}$$

- Then our kernel is

$$K(\boldsymbol{x_1},\boldsymbol{x_2})=\phi(\boldsymbol{x_1})\cdot\phi(\boldsymbol{x_2})=x_{11}^2x_{21}^2+x_{12}^2x_{22}^2+2\,x_{11}x_{12}x_{21}x_{22}$$

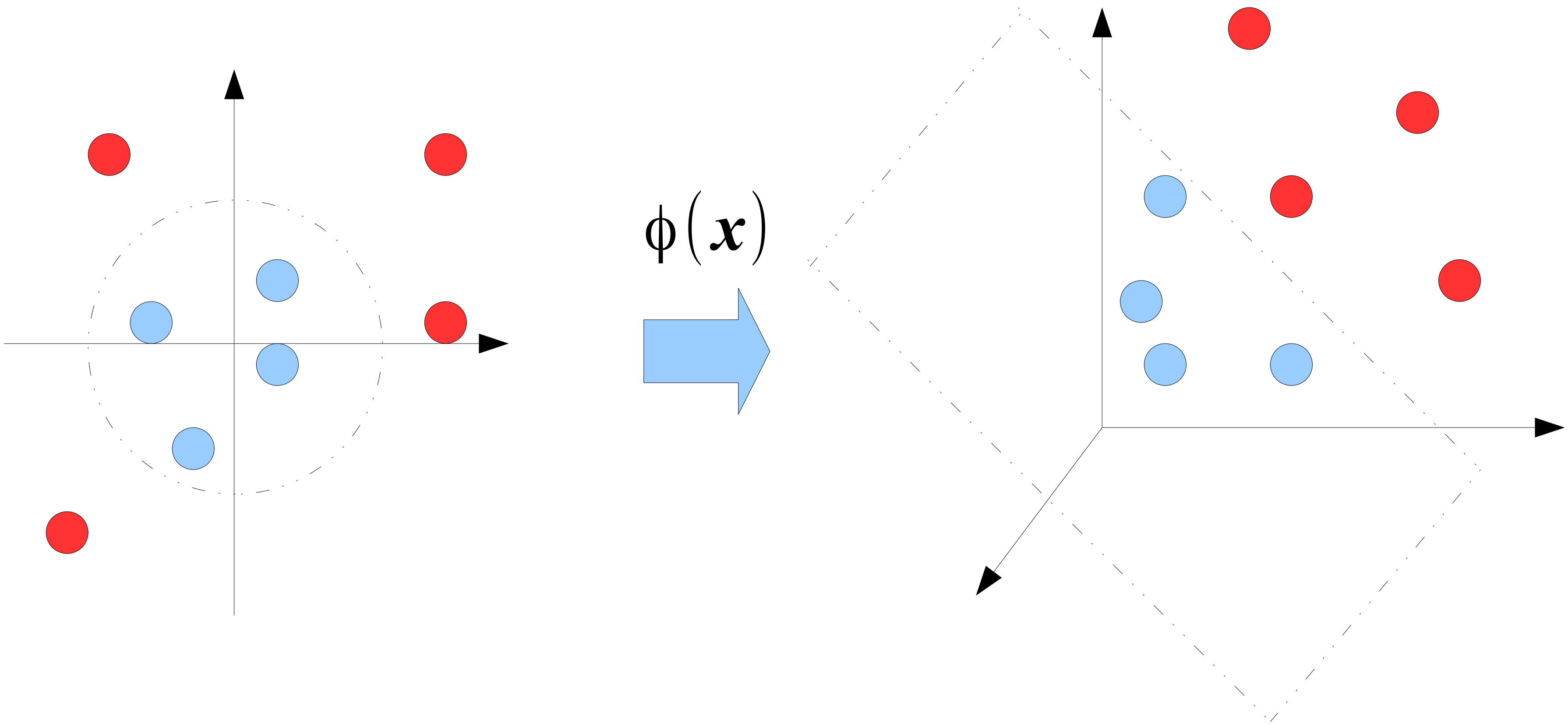
  - Which is actually a square of dot product in the original space, since $(\boldsymbol{x_1}\cdot\boldsymbol{x_2})=x_{11}x_{21}+x_{12}x_{22}$ and

$$(\boldsymbol{x_1}\cdot\boldsymbol{x_2})^2=x_{11}^2x_{21}^2+x_{12}^2x_{22}^2+2\,x_{11}x_{12}x_{21}x_{22}$$

- No need to explicitly compute $\phi(\boldsymbol{x})$

Illustration of mapping to higher dimension

- A mapping function from 2D to 3D



Popular kernels

- Linear $$K(i,j)=\boldsymbol{x}_i \cdot \boldsymbol{x}_j$$
- Polynomial $$K(i,j)=(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)^d$$
  - kernel trick can be extended to any degree.
- Gaussian $$K(i,j)=\exp\left(\frac{-\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}{2\sigma^2}\right)$$
  - σ is a parameter to be adjusted.
  - Gaussian kernel spans infinite dimensional feature space, which can be thought as sum of sequence of polynomial kernels.

Exercise 2: Nonlinear SVM by kernels

- Try svm_rbf.m
- Regularization parameter λ and gaussian kernel parameter σ are options.
  – Usage: svm_rbf(λ,σ)
- Choose $\lambda = \{0.1,1, 10\}$ and $\sigma = \{0.1,1,10\}$, and try their combinations. What do you observe ? Which parameters give best training accuracy ?

- The examples and codes are prepared by Andrew Ng.
  – https://www.coursera.org/learn/machine-learning/programming

Gaussian Process (GP)

- An another kernel method especially useful for regression.
- In the previous example, we have solved $X X'\alpha = y$ instead of $X\beta = y$, and obtained $\alpha = (X X')^{-1} y$, which is a Gaussian process mean predictor.
- Similarly to SVM, we can consider $K = X X'$ and employ kernels.
- Unlike SVM, GP is a probabilistic model.

GP prediction

- More generally, for a new datapoint x* and kernels $K_{i,j}=k(x_i, x_j), (k_*)_i=k(x_*, x_i)$, GP prediction model is given as

$$p(f(x_*)|X, y, x_*)=N(f(x_*)|\alpha_*, \Sigma_*)$$

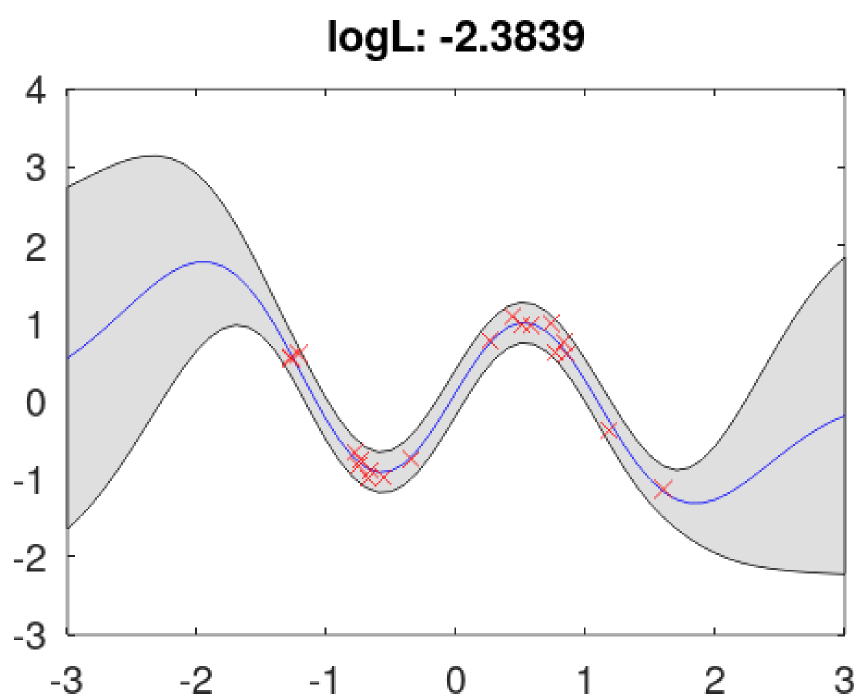
  – where
$$\boldsymbol{\alpha}_*=\boldsymbol{k}_*{}'\boldsymbol{K}^{-1}\boldsymbol{y}$$
$$\boldsymbol{\Sigma}_*=k(x_*, x_*)-\boldsymbol{k}_*{}'\boldsymbol{K}^{-1}\boldsymbol{k}_*$$

- Prediction mean $\boldsymbol{\alpha}_*$ is the same as that of kernel ridge regression.

- Prediction variance $\boldsymbol{\Sigma}_*$ can be seen as a confidence for prediction. Smaller variance corresponds to higher confidence.

GP Example

- An example of fitting to y = sin(3x) + random noise.
- Red crosses corresponds to the data points, blue line corresponds to the prediction mean, gray bands corresponds to the prediction variance (confidence).



Sample kernel function

$$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \tau\, exp\left(-\frac{\left|x_i - x_j\right|^2}{\sigma}\right) + \eta\, \delta_{i,j}$$

- The first term resenbles Gaussian distribution, and the second term measres the signal / noise ratio.
- More similarity between x_i and x_j results in the laterger value of k(x_i, x_j).

Kernel prameter optimzation

- Log-likelihood of GP

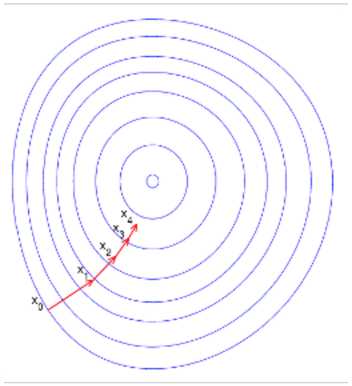
$$\log p(\boldsymbol{y}\,|\theta) = -\frac{1}{2}\log|\boldsymbol{K}| - \frac{1}{2}\boldsymbol{y}^{\top}\boldsymbol{K}^{-\mathbf{1}}\boldsymbol{y} - \frac{n}{2}log(2\pi)$$

- Derivative of log-likelihood w.r.t. parameters

$$\frac{\partial}{\partial\theta_i}\log p(\boldsymbol{y}\,|\theta) = -\frac{1}{2}\mathrm{Tr}(\boldsymbol{K}^{-\mathbf{1}}\frac{\partial K}{\partial\theta_i}) - \frac{1}{2}\boldsymbol{y}^{\top}\boldsymbol{K}^{-\mathbf{1}}\frac{\partial \boldsymbol{K}}{\partial\theta_i}\boldsymbol{K}^{-\mathbf{1}}\boldsymbol{y}$$

- Can be optimized by gradient descent

$$\theta_i \leftarrow \theta_i + \epsilon\frac{\partial}{\partial\theta_i}\log p(\boldsymbol{y}\,|\theta)$$



Ex: GP kernel parameter optimzation

• Try demo by executing “gp_opt_demo.m” in GP.zip.

•3 parameters in the kernel are assigned randomly.

$$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \tau \, exp\left(-\frac{\left|x_i - x_j\right|^2}{\sigma}\right) + \eta \, \delta_{i,j}$$

•After hitting “return”, gradient descent optimizer searches for a better parameter set.
•Parameters are initialized randomly, so try multiple times to see different fitting resutls.