Pattern Information Processing: Support Vector Machines

Masashi Sugiyama
(Department of Computer Science)

Contact: W8E-505
sugi@cs.titech.ac.jp
http://sugiyama-www.cs.titech.ac.jp/~sugi/

(Binary) Classification Problem

- Output values are $y_i = \pm 1$.
- We want to predict whether output values of unlearned input points are positive or negative.

- Multi-class problem can be transferred to several binary classification problems:
  - One-versus-rest (1vs.2&3, 2vs.1&3, 3vs.1&2)
  - One-versus-one (1vs.2, 1vs.3, 2vs.3)

(Binary) Classification Problem

- In classification, we may still use the same learning methods, e.g., quadratically-constrained least-squares:

$$\hat{\boldsymbol{\alpha}}_{QCLS} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^b}{\operatorname{argmin}} \left[J_{LS}(\boldsymbol{\alpha}) + \lambda \langle \boldsymbol{R}\boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle \right]$$

$$\lambda \ (\geq 0)$$

$$J_{LS}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \left(f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) - y_i \right)^2$$

- Prediction:

$$\widehat{y} = \operatorname{sign}\left(f_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{x}) \right)$$

0/1-Loss

- In classification, only the sign of the learned function is used.
- It is natural to use 0/1-loss instead of squared-loss $J_{LS}(\boldsymbol{\alpha})$:

$$J_{0/1}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} I\left(\mathrm{sign}(f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i)) \neq y_i\right)$$

$$I(a \neq b) = \begin{cases} 0 & (a = b) \\ 1 & (a \neq b) \end{cases}$$

- $J_{0/1}(\boldsymbol{\alpha})$ corresponds to the number of misclassified samples (thus natural).

Hinge-Loss

- However, $J_{0/1}(\boldsymbol{\alpha})$ is non-convex so we may not be able to obtain the global minimizer.
- Use hinge-loss as an approximation:

$$J_H(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \max(0, 1 - u_i)$$

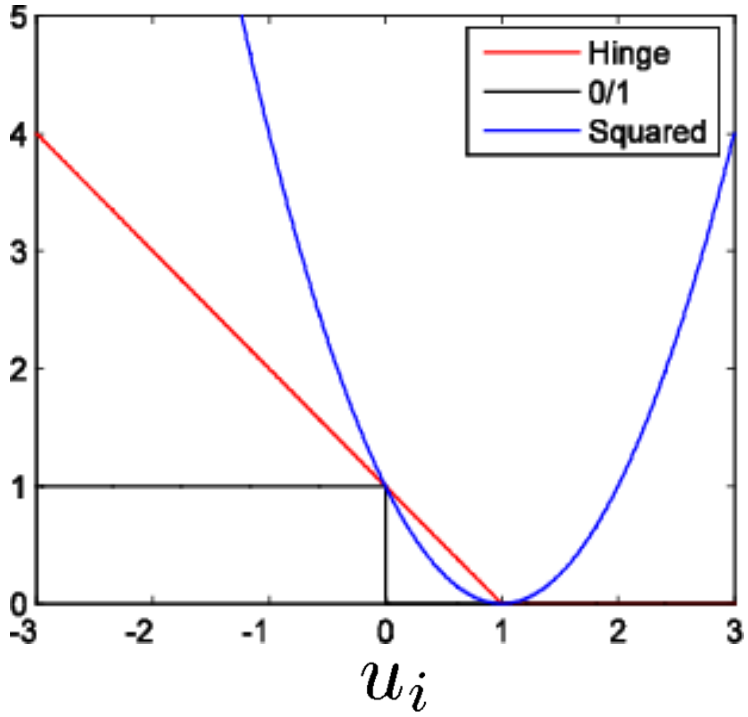
$$J_{0/1}(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{i=1}^{n} (1 - \text{sign}(u_i))$$

$$J_{LS}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} (1 - u_i)^2$$

$$\text{Note} : y_i^2 = 1,\ 1/y_i = y_i$$

$$u_i = f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) y_i$$

: Sample-wise margin



How to Obtain A Solution

$$\hat{\boldsymbol{\alpha}}_{SVM} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^b}{\operatorname{argmin}} \left[J_H(\boldsymbol{\alpha}) + \lambda \langle \boldsymbol{R}\boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle \right]$$

$$J_H(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \max(0, 1 - u_i)$$

- How to deal with "max"? Use following lemma:

Lemma:

$$\max(0, 1-u) = \min_{\xi\in\mathbb{R}} \xi \quad \text{subject to } \xi \geq 1-u, \; \xi \geq 0$$

- Proof: Constraints are $\xi \geq \max(0, 1-u)$, so the lemma holds. Q.E.D.

How to Obtain A Solution (cont.)

- So we have

$$J_H(\boldsymbol{\alpha}) = \min_{\boldsymbol{\xi} \in \mathbb{R}^n} \langle \mathbf{1}_n, \boldsymbol{\xi} \rangle \text{ subject to } \boldsymbol{\xi} \geq \mathbf{1}_n - \boldsymbol{u}$$
$$\boldsymbol{\xi} \geq \mathbf{0}_n$$

- Then $\hat{\boldsymbol{\alpha}}_{SVM}$ is given as

$$\hat{\boldsymbol{\alpha}}_{SVM} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^b, \boldsymbol{\xi} \in \mathbb{R}^n}{\operatorname{argmin}} \left[\langle \mathbf{1}_n, \boldsymbol{\xi} \rangle + \lambda \langle \boldsymbol{R}\boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle \right]$$
$$\text{subject to } \boldsymbol{\xi} \geq \mathbf{1}_n - \boldsymbol{u}$$
$$\boldsymbol{\xi} \geq \mathbf{0}_n$$

Support Vector Machines

- We focus on the following setting:
  - $f_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i)$
  - $\boldsymbol{R} = \boldsymbol{K}$

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

- Setting $\lambda = (2C)^{-1}$, we have

$$\widehat{\boldsymbol{\alpha}}_{SVM} = \underset{\boldsymbol{\alpha}, \boldsymbol{\xi} \in \mathbb{R}^n}{\operatorname{argmin}} \left[C\langle \mathbf{1}_n, \boldsymbol{\xi} \rangle + \frac{1}{2} \langle \boldsymbol{K}\boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle \right]$$

$$\text{subject to } \boldsymbol{\xi} \geq \mathbf{1}_n - \boldsymbol{u}$$

$$\boldsymbol{\xi} \geq \mathbf{0}_n$$

$$u_i = f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) y_i$$

Efficient Formulation

- The SVM solution can be obtained by

$[\widehat{\boldsymbol{\alpha}}_{SVM}]_i = [\widehat{\boldsymbol{\beta}}_{SVM}]_i y_i$, where

Proof: Homework!

$$\widehat{\boldsymbol{\beta}}_{SVM} = \underset{\boldsymbol{\beta}\in\mathbb{R}^n}{\operatorname{argmax}} \left[\sum_{i=1}^{n} \beta_i - \frac{1}{2} \sum_{i,j=1}^{n} \beta_i \beta_j y_i y_j \boldsymbol{K}_{i,j} \right]$$

$$\text{subject to } \mathbf{0}_n \leq \boldsymbol{\beta} \leq C\mathbf{1}_n$$

- The number of parameters is reduced to n.
- QP standard form:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^n} \left[\frac{1}{2}\langle \boldsymbol{Q}\boldsymbol{\beta}, \boldsymbol{\beta}\rangle + \langle \boldsymbol{\beta}, \boldsymbol{q}\rangle \right]$$

$$\text{subject to } \boldsymbol{H}\boldsymbol{\beta} \leq \boldsymbol{h}$$

$$\boldsymbol{Q}_{i,j} = \boldsymbol{K}_{i,j} y_i y_j \qquad \boldsymbol{q} = -\mathbf{1}_n$$

$$\boldsymbol{H} = \begin{pmatrix} -\boldsymbol{I}_n \\ \boldsymbol{I}_n \end{pmatrix} \qquad \boldsymbol{h} = \begin{pmatrix} \mathbf{0}_n \\ C\mathbf{1}_n \end{pmatrix}$$

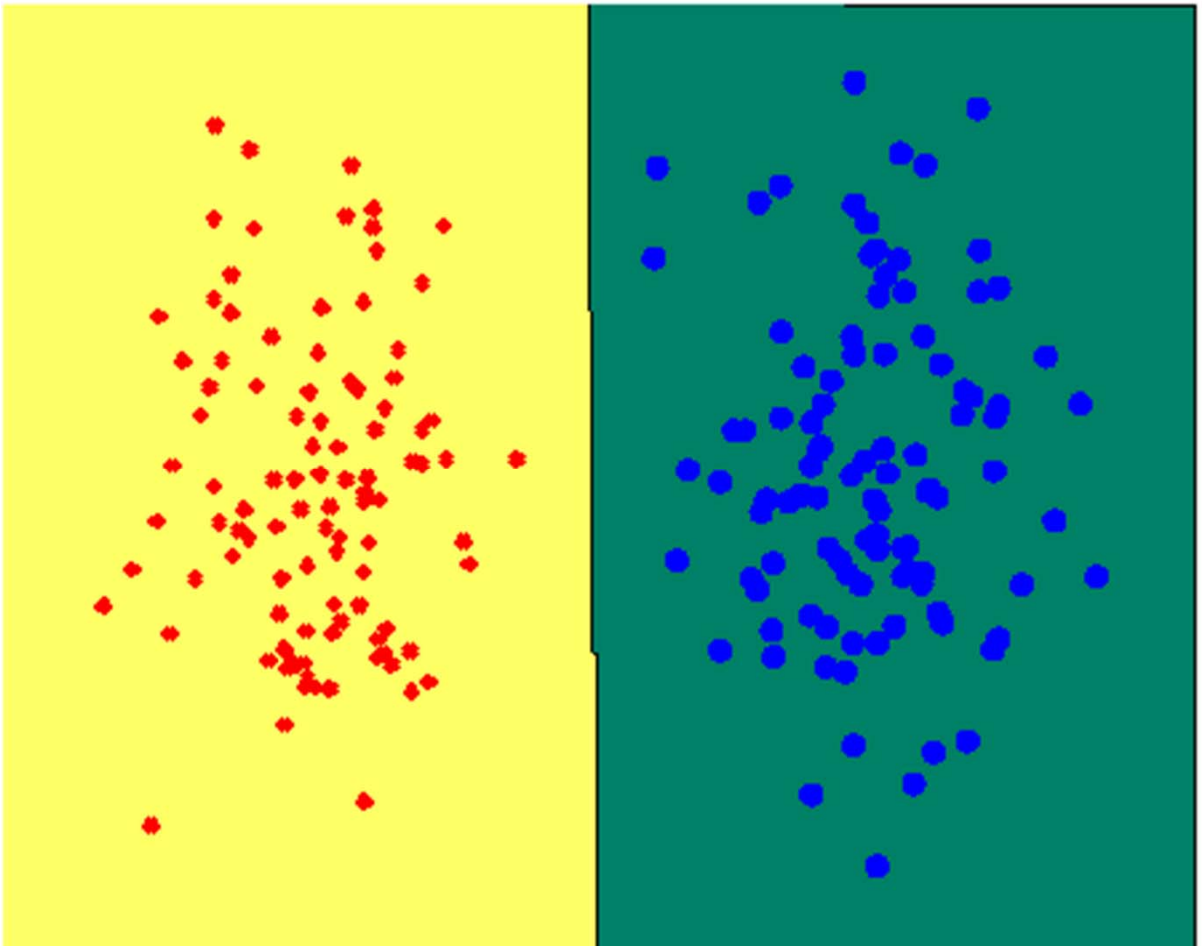
Sparseness

- KKT optimality condition implies

$$\beta_i(\xi_i + u_i - 1) = 0 \quad \text{for all } i \qquad u_i = \hat{f}(\boldsymbol{x}_i)y_i$$

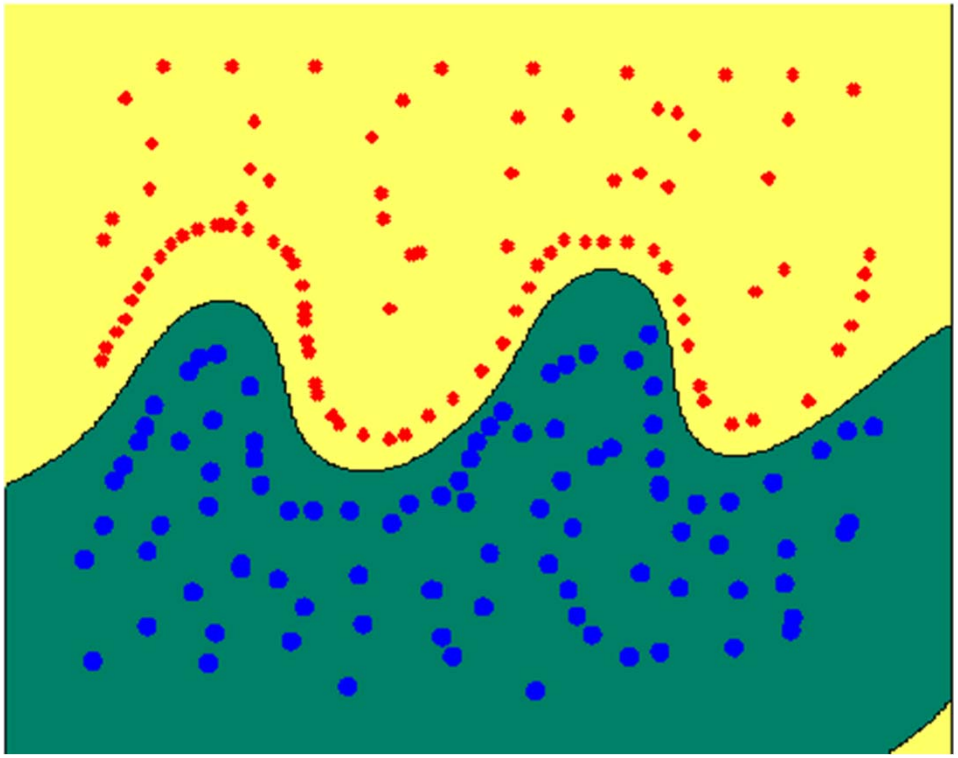
- Therefore, some β_i (and thus $\alpha_i = \beta_i y_i$ also) could be zero.

Examples

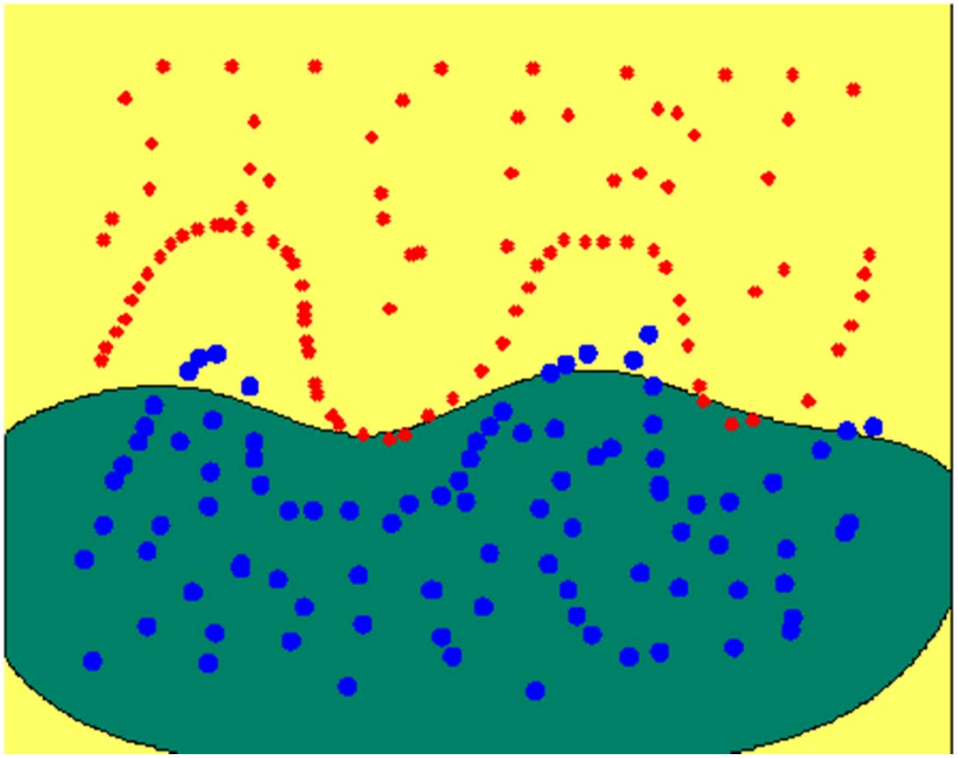


Gaussian kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2c^2}\right)$

Examples (cont.)

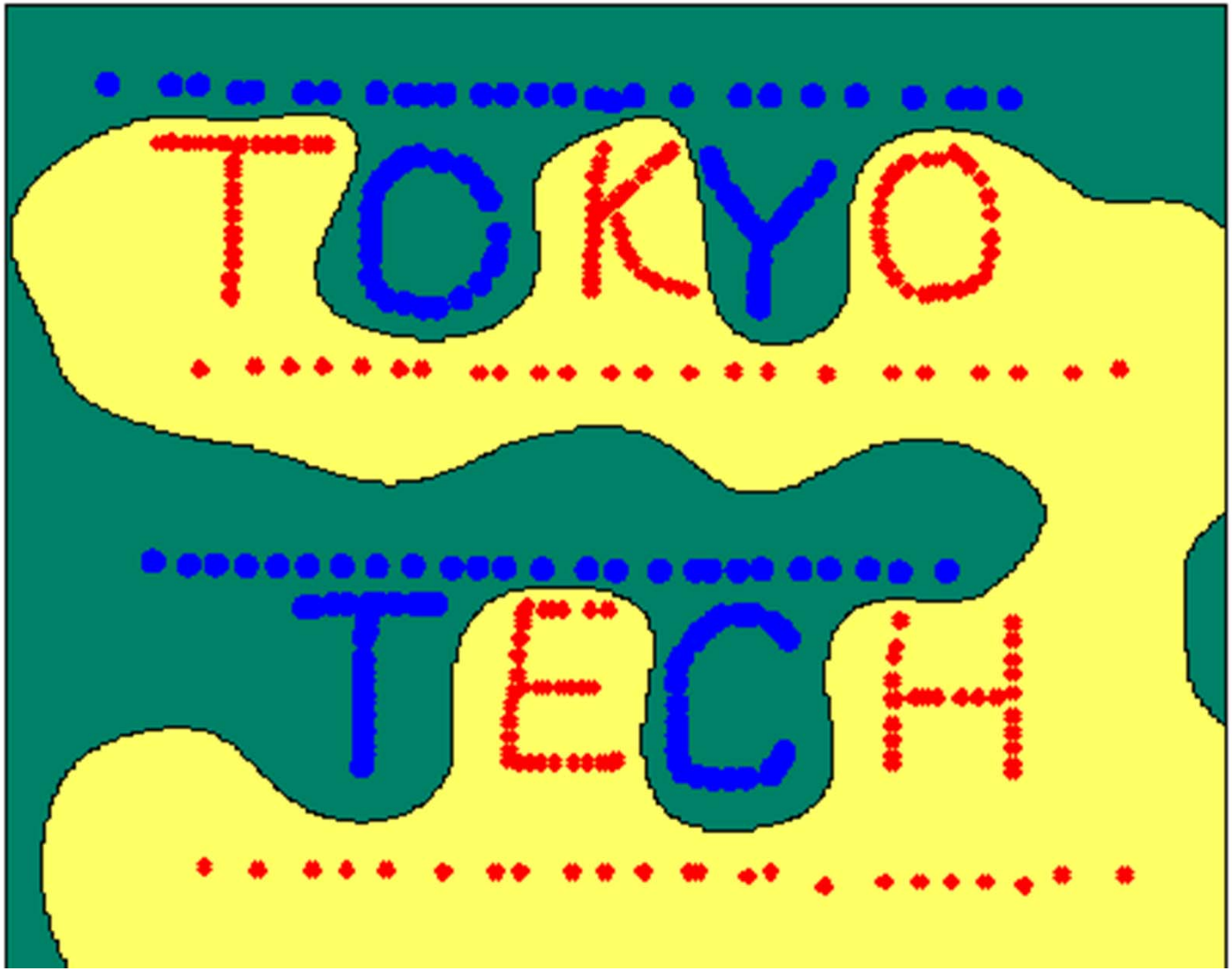


Large C



Small C

Examples



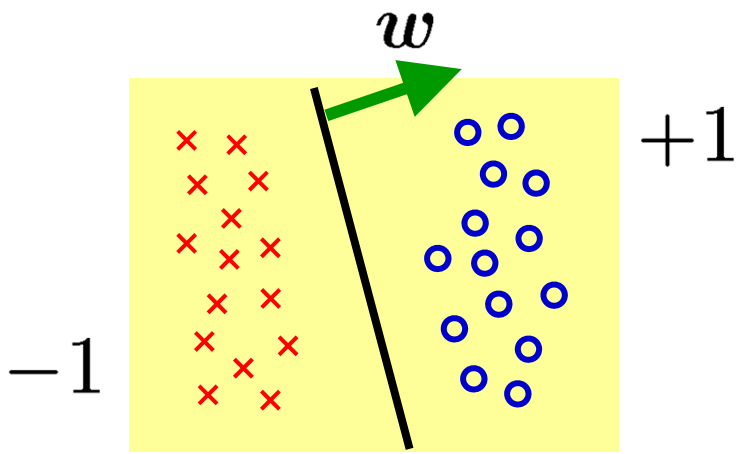
Original Derivation of SVMs

- The way SVMs were introduced today is quite different from the original derivation.
- Let's briefly follow the original derivation.
  - Hyper-plane classifier
  - VC theory
  - Margin maximization
  - Soft margin
  - Kernel trick

Hyper-plane Classifier

- Separate sample space by hyper-plane.

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \qquad \widehat{y} = \operatorname{sign}(f_{\boldsymbol{w}}(\boldsymbol{x}))$$

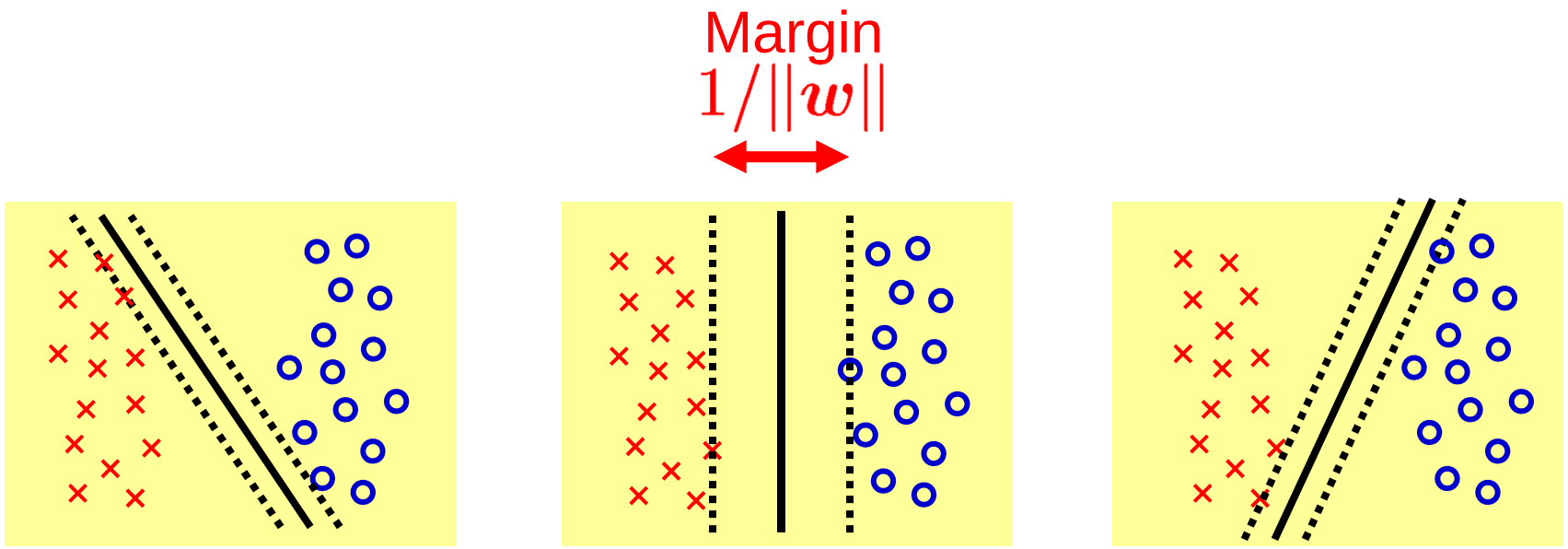


find $\boldsymbol{w}, b$

such that $y_i f_{\boldsymbol{w}}(\boldsymbol{x}_i) \geq 1$ for $i = 1, \ldots, n$.

Margin

- Margin: “Gap” between two classes



Small margin

Large margin

Small margin

Vapnik-Chevonenkis Theory

- Generalization error:

$$R[\hat{f}] = \int\int I(\hat{f}(\boldsymbol{x}) \neq y) p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

- Empirical error:

$$R_{\mathrm{emp}}[\hat{f}] = \frac{1}{n} \sum_{i=1}^{n} I(\hat{f}(\boldsymbol{x}_i) \neq y_i)$$

$$I(a \neq b) = \begin{cases} 0 & (a = b) \\ 1 & (a \neq b) \end{cases}$$

- Generalization error bound ("VC bound")

$$R[\hat{f}] \leq R_{\mathrm{emp}}[\hat{f}] + \sqrt{\frac{1}{n}\left(h\left(\log\frac{2n}{h}+1\right)+\log\frac{4}{\delta}\right)}$$

with probability $1-\delta$

h : VC dimension (model complexity)

Vapnik-Chevonenkis Theory (cont.)

- VC bound:

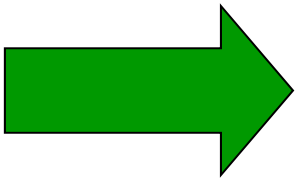
$$R[\widehat{f}] \leq R_{\mathrm{emp}}[\widehat{f}] + \sqrt{\frac{1}{n}\left(h\left(\log\frac{2n}{h}+1\right)+\log\frac{4}{\delta}\right)}$$

Monotone decreasing with respect to VC dimension h $(h < n)$

- If samples are linear separable, empirical error is zero.

$$R_{\mathrm{emp}}[\widehat{f}] = 0$$

- The larger margin is, the smaller VC dim is.

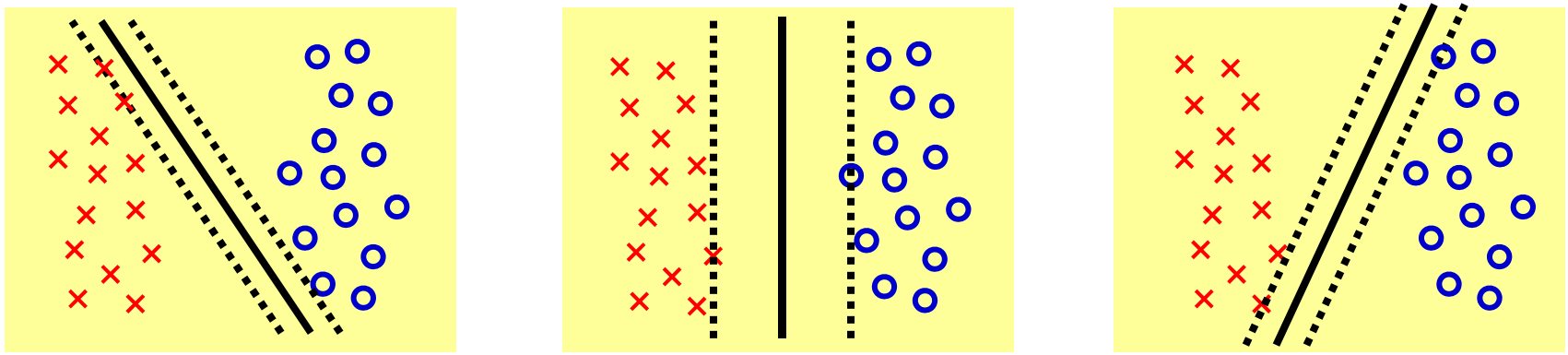


In VC theory, maximum margin classifier is optimal

Optimal Hyper-plane Classifier

- Separate two classes with maximum margin

Margin $1/||\boldsymbol{w}||$



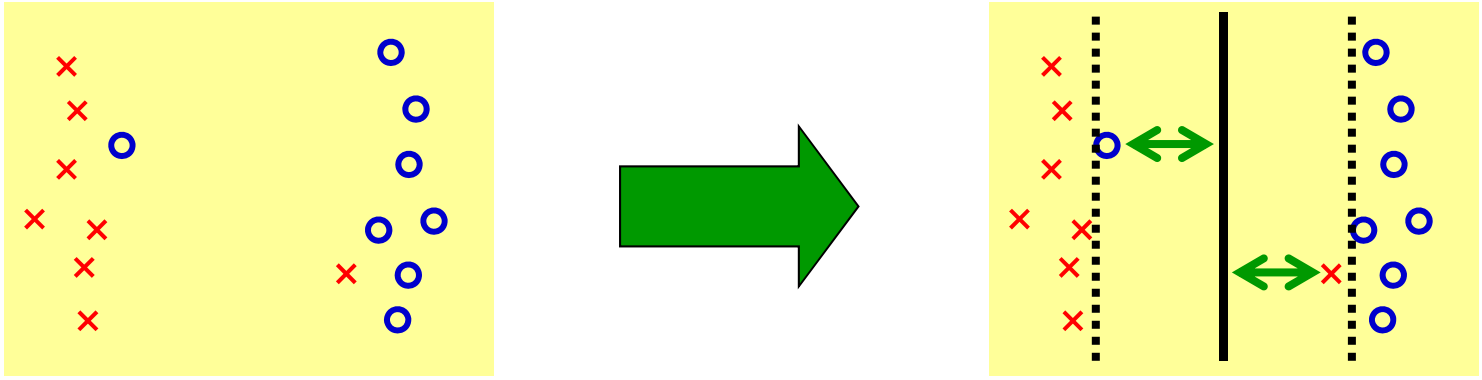
Small margin | Large margin | Small margin

$$\min_{\boldsymbol{w},b} ||\boldsymbol{w}||^2$$

$$\text{subject to} \quad y_i f_{\boldsymbol{w}}(\boldsymbol{x}_i) \geq 1 \quad \text{for } i = 1, \ldots, n.$$

Soft Margin

- If samples are not linearly separable, margin cannot be defined.
- Allow small error ξ_i.



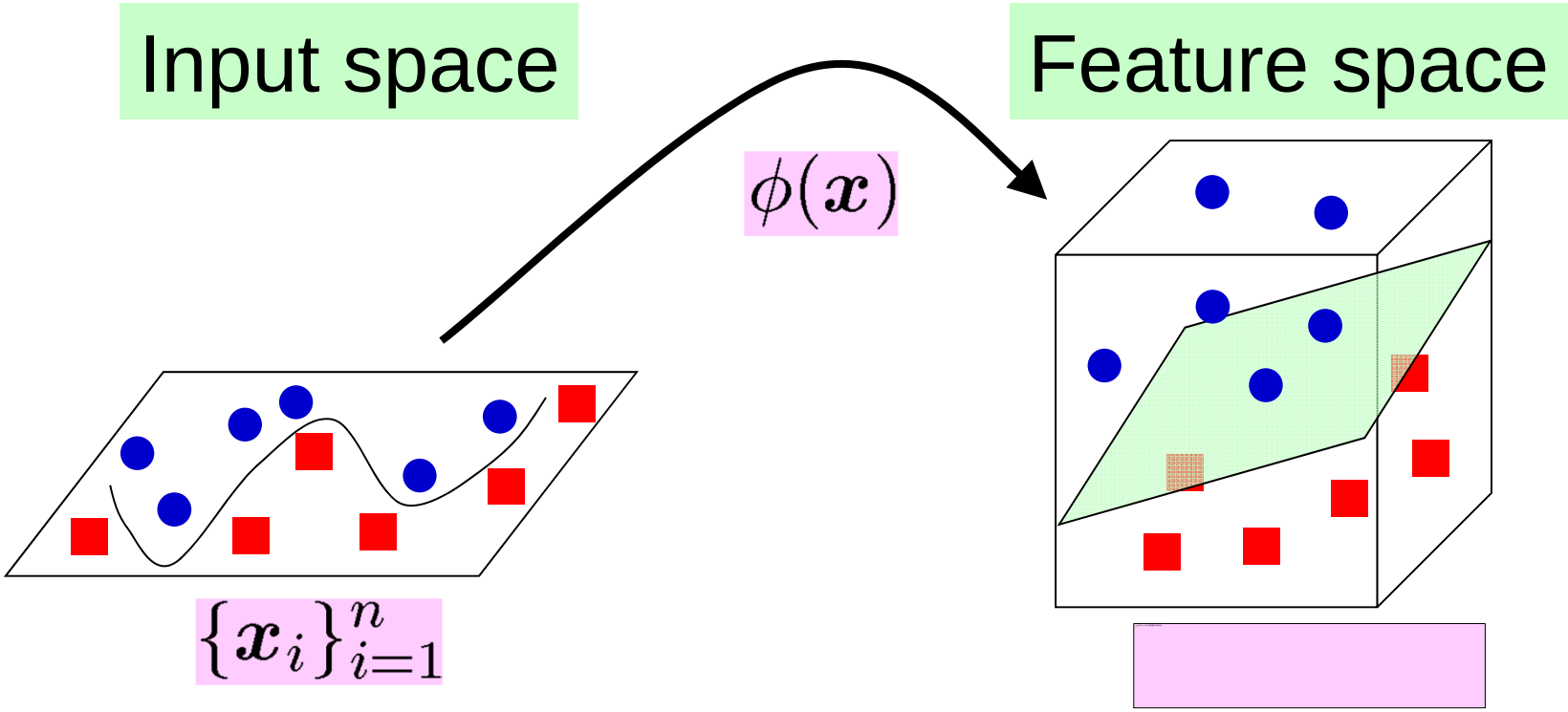
$$\min_{\boldsymbol{w},b} \|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{subject to} \quad y_i f_{\boldsymbol{w}}(\boldsymbol{x}_i) \geq 1 - \xi_i$$

$$\xi_i \geq 0 \quad \text{for } i = 1, \ldots, n.$$

Non-linear Extension

- Transform samples to a feature space by a non-linear mapping $\phi(\boldsymbol{x})$.
- Then find the maximum margin hyper-plane in the feature space.



Kernel Trick

- Compute inner product in the feature space by a kernel function:

$$\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\forall \boldsymbol{x}, \boldsymbol{x}', \quad K(\boldsymbol{x}, \boldsymbol{x}') \geq 0$$

E.g., Gaussian kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2 / c^2\right)$$

- Any linear algorithm represented by inner product can be non-linearized by kernels
  - E.g.: Support vector machine, k-nearest neighbor classifier, principal component analysis, linear discriminant analysis, k-means clustering,

Various Losses for Classification

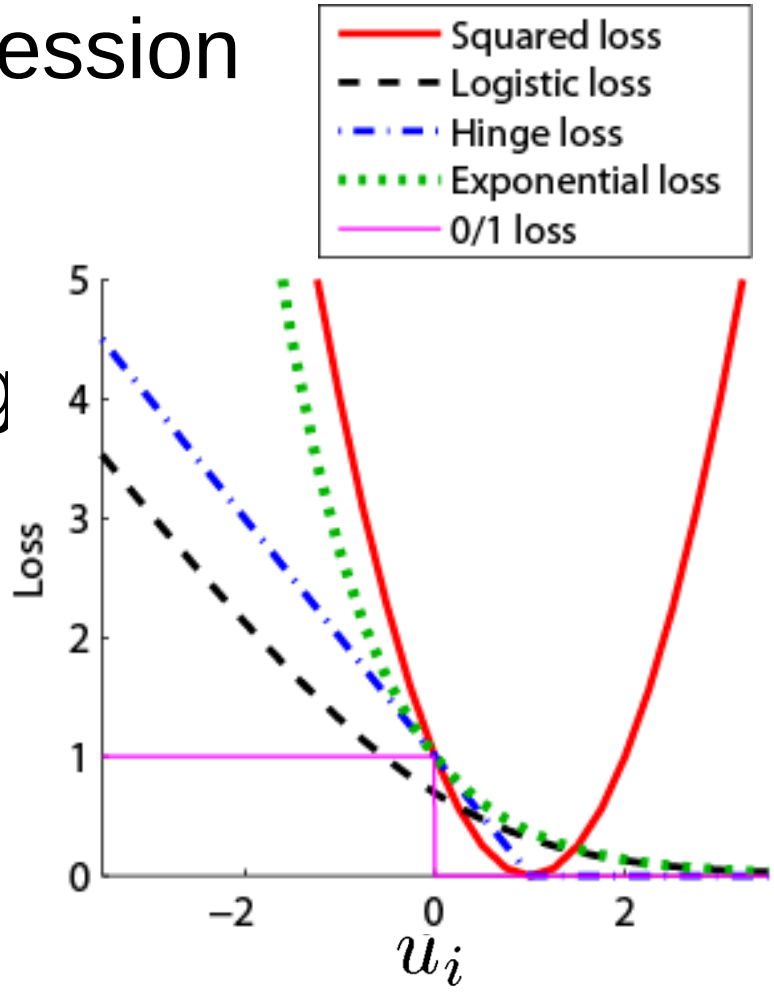
- Hinge loss: Support vector machine
- Squared loss: Fisher discriminant analysis
- Logistic loss: Logistic regression

$$J_{Logistic}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \log\left(1 + e^{-u_i}\right)$$

- Exponential loss: Boosting

$$J_{Boost}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} e^{-u_i}$$

$$u_i = f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) y_i$$



Homework

1. Prove that the solution of SVM,

$$\widehat{\boldsymbol{\alpha}}_{SVM} = \underset{\boldsymbol{\alpha},\boldsymbol{\xi}\in\mathbb{R}^n}{\operatorname{argmin}} \left[C\langle \mathbf{1}_n, \boldsymbol{\xi}\rangle + \frac{1}{2}\langle \boldsymbol{K}\boldsymbol{\alpha}, \boldsymbol{\alpha}\rangle \right]$$

$$\text{subject to } \boldsymbol{\xi} \geq \mathbf{1}_n - \boldsymbol{u}, \quad \boldsymbol{\xi} \geq \mathbf{0}_n$$

$$f_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i)$$

$$u_i = f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) y_i$$

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

is given by $[\widehat{\boldsymbol{\alpha}}_{SVM}]_i = [\widehat{\boldsymbol{\beta}}_{SVM}]_i y_i$, where

$$\widehat{\boldsymbol{\beta}}_{SVM} = \underset{\boldsymbol{\beta}\in\mathbb{R}^n}{\operatorname{argmax}} \left[\sum_{i=1}^{n} \beta_i - \frac{1}{2} \sum_{i,j=1}^{n} \beta_i \beta_j y_i y_j \boldsymbol{K}_{i,j} \right]$$

$$\text{subject to } \mathbf{0}_n \leq \boldsymbol{\beta} \leq C\mathbf{1}_n$$

Hint: Use Wolfe dual

Homework (cont.)

- Lagrangian:

$$L(\boldsymbol{\alpha}, \boldsymbol{\xi}, \beta, \gamma) = C\langle \mathbf{1}_n, \boldsymbol{\xi} \rangle + \frac{1}{2}\langle \boldsymbol{K}\boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle$$
$$-\langle \beta, \boldsymbol{\xi} + \boldsymbol{u} - \mathbf{1}_n \rangle - \langle \gamma, \boldsymbol{\xi} \rangle$$

- β, γ :Lagrange multiplier
- Wolfe duality:

$$\min_{\boldsymbol{\alpha}, \boldsymbol{\xi} \in \mathbb{R}^n} \left[C\langle \mathbf{1}_n, \boldsymbol{\xi} \rangle + \frac{1}{2}\langle \boldsymbol{K}\boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle \right] = \max_{\beta, \gamma \in \mathbb{R}^n} L(\boldsymbol{\alpha}, \boldsymbol{\xi}, \beta, \gamma)$$

$$\text{subject to } \boldsymbol{\xi} \geq \mathbf{1}_n - \boldsymbol{u} \qquad \text{subject to } \beta \geq \mathbf{0}_n \qquad \gamma \geq \mathbf{0}_n$$

$$\boldsymbol{\xi} \geq \mathbf{0}_n \qquad \frac{\partial L}{\partial \boldsymbol{\alpha}} = \mathbf{0}_n \qquad \frac{\partial L}{\partial \boldsymbol{\xi}} = \mathbf{0}_n$$

Homework

2. Prepare a toy binary classification problem (say 2-dim input) and test SVM. Then analyze the results by varying experimental conditions (datasets, kernels, regularization parameter C etc.).
   - Software is available from, e.g., http://www.support-vector.net/software.html
   - You may play with Java implementation, e.g., http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml