

Pattern Recognition



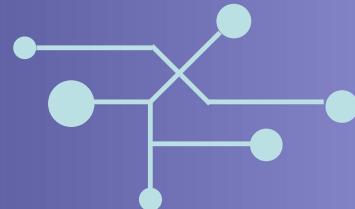
Kernel Methods

林彥宇 教授

Yen-Yu Lin, Professor

國立陽明交通大學 資訊工程學系

Computer Science, National Yang Ming Chiao Tung University



Some slides are modified from S.-J. Wang,
H.-T. Chen,



Outline

- Dual representations
- Constructing kernels
- Support vector machines for classification
- Support vector machines for regression



Outline

- Dual representations
- Constructing kernels
- Support vector machines for classification
- Support vector machines for regression



Parametric vs. non-parametric model

- Parametric methods
 - A linear model for regression
 - We learn a model $y(\mathbf{x}, \mathbf{w})$ that maps input \mathbf{x} to output y
 - Training data are thrown away after training
- Non-parametric methods
 - Nearest neighbor classifier
 - We predict a test data by searching its nearest neighbor
 - Training data are kept
- Kernel methods
 - Support vector machines, Gaussian processes, kernel PCA, ...
 - Predictions are based on linear combinations of a kernel function evaluated at the training data points



Dual representations: Problem statement

- Consider a linear regression model whose parameters are determined by minimizing a regularized sum-of-squares error function given by

$$J(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \quad \text{where } \lambda \geq 0$$

- Data are nonlinearly transformed via functions
- Setting the derivative of $J(\mathbf{w})$ w.r.t. \mathbf{w} to zero, the optimal solution takes the form of

$$\mathbf{w} = -\frac{1}{\lambda} \sum_{n=1}^N \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\} \phi(\mathbf{x}_n)$$

Dual representations of a linear model

- The optimal solution is a linear combination of training data

$$\mathbf{w} = -\frac{1}{\lambda} \sum_{n=1}^N \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\} \phi(\mathbf{x}_n)$$

$$= \sum_{n=1}^N a_n \phi(\mathbf{x}_n) = \Phi^T \mathbf{a}.$$

➤ where $a_n = -\frac{1}{\lambda} \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\}$

$$\Phi : \begin{bmatrix} & \vdots \\ \phi(\mathbf{x}_n)^T & \\ & \vdots \end{bmatrix} \quad \mathbf{a} = (a_1, \dots, a_N)^T$$

Dual representations: Gram matrix

- Instead of working with parameter vector \mathbf{w} , we can reformulate the least squares algorithms w.r.t. **parameter vector \mathbf{a}** , giving rise to a **dual representation**
- If we substitute $\mathbf{w} = \Phi^T \mathbf{a}$ into $J(\mathbf{w})$, we obtain

$$J(\mathbf{a}) = \frac{1}{2} \mathbf{a}^T \Phi \Phi^T \Phi \Phi^T \mathbf{a} - \mathbf{a}^T \Phi \Phi^T \mathbf{t} + \frac{1}{2} \mathbf{t}^T \mathbf{t} + \frac{\lambda}{2} \mathbf{a}^T \Phi \Phi^T \mathbf{a}$$

- Where $\mathbf{t} = (t_1, \dots, t_N)^T$
- We define the **Gram (Kernel) matrix $\mathbf{K} = \Phi \Phi^T$** , which is an $N \times N$ symmetric matrix

$$K_{nm} = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

- where $k(\mathbf{x}_n, \mathbf{x}_m)$ is the **kernel function**



Dual representations: Solution and inference

- In terms of the Gram matrix, the regularized sum-of-squares error function can be written as

$$J(\mathbf{a}) = \frac{1}{2} \mathbf{a}^T \mathbf{K} \mathbf{K} \mathbf{a} - \mathbf{a}^T \mathbf{K} \mathbf{t} + \frac{1}{2} \mathbf{t}^T \mathbf{t} + \frac{\lambda}{2} \mathbf{a}^T \mathbf{K} \mathbf{a}$$

- Setting the derivative of $J(\mathbf{a})$ w.r.t. \mathbf{a} to zero, the optimal solution is

$$\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}.$$

- After getting the solution, we make the prediction for input \mathbf{x}

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) = \mathbf{a}^T \Phi \phi(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

➤ where $\mathbf{k}(\mathbf{x})$ is with elements $k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x})$

Predictions are based on linear combinations of a kernel function evaluated at the training data points

Dual representations: Analysis

- There are M (data dimensionality) optimization variables in \mathbf{w}
- There are N (data number) optimization variables in its dual representation $\Phi^T \mathbf{a}$
- In the cases where $N > M$
 - Dual representations are less efficient
 - Dual representations are not parametric
 - Dual representations can be sparse
 - Dual representations can be expressed entirely in terms of kernel functions -> It can avoid explicit introduction of the features $\phi(\mathbf{x})$, which allows us to implicitly use feature spaces of very high, even infinite, dimensionality

Outline

- Dual representations
- Constructing kernels
- Support vector machines for classification
- Support vector machines for regression



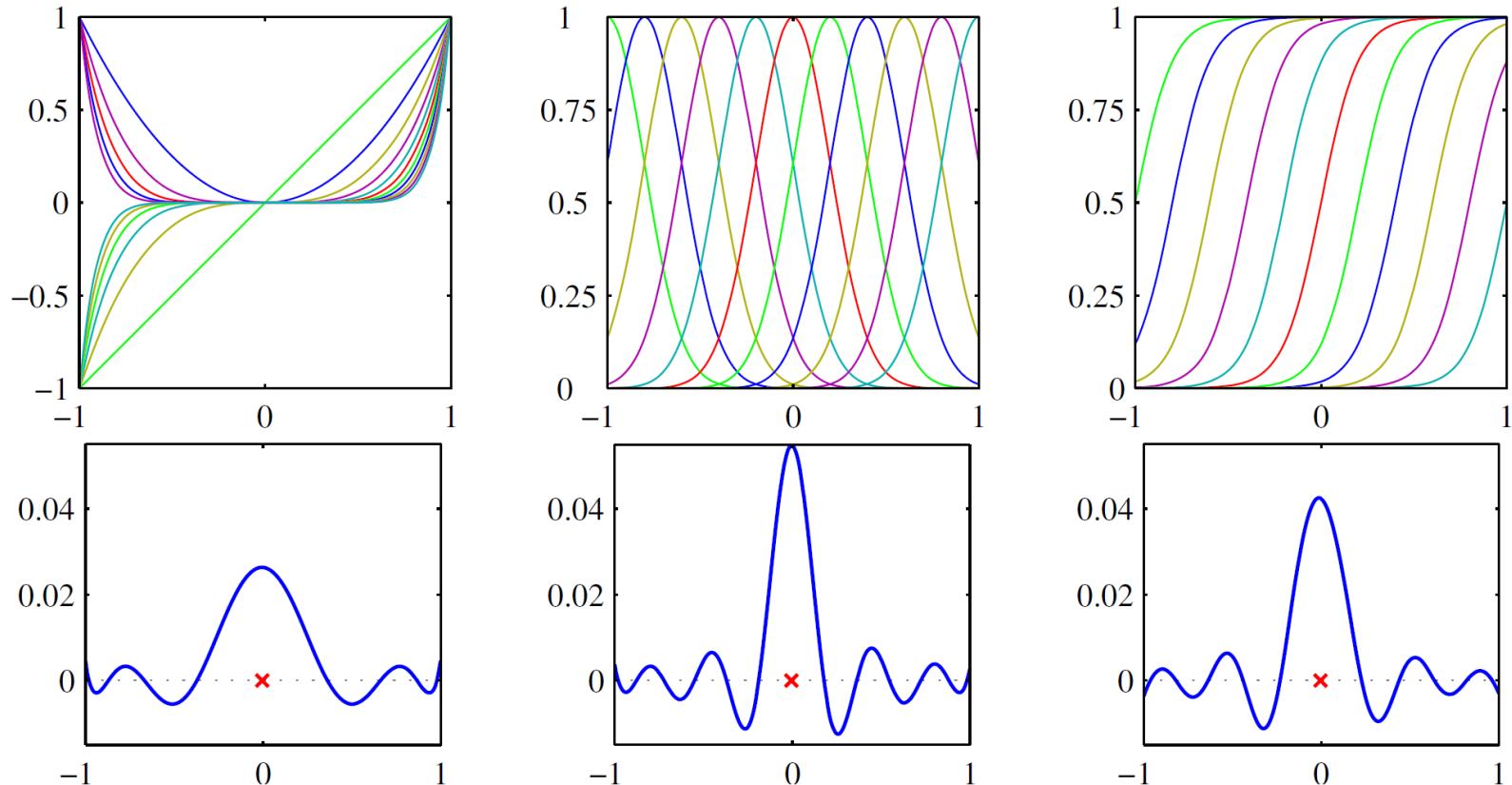
How to construct a kernel function/matrix?

- One approach is to choose a feature space mapping and then construct the kernel

$$k(x, x') = \phi(x)^T \phi(x') = \sum_{i=1}^M \phi_i(x) \phi_i(x')$$

- where $\phi_i(x)$ is a basis function
- Another approach is to construct the kernel functions directly
- It is required that the constructed kernel is **valid**. Namely it corresponds to the inner product in some feature space

Examples of explicitly constructing kernels



- Upper figure: Several curves, one for each basis function
- Lower figure: kernel function $k(x, x')$ with $x' = 0$ by varying x



Is it a valid kernel?

- Let's consider the following function. Is it a kernel function?

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2$$

- Check if there exists a space where the output value is equal to the inner product of the data points
- With the derivation

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= (\mathbf{x}^T \mathbf{z})^2 = (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (x_1^2, \sqrt{2}x_1 x_2, x_2^2)(z_1^2, \sqrt{2}z_1 z_2, z_2^2)^T \\ &= \phi(\mathbf{x})^T \phi(\mathbf{z}). \end{aligned}$$

- The feature space exists $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)^T$
- Note that the kernel function is computed in the input space, but it corresponds to the inner product in some high-dimensional space



A theory for checking if a function is a kernel function

- The **necessary and sufficient condition** for $k(\mathbf{x}, \mathbf{x}')$ to be a valid kernel: Gram matrix $\mathbf{K} = [k(\mathbf{x}_n, \mathbf{x}_m)]_{nm}$ should be **positive semidefinite** for all possible choices of the set
- A matrix is positive semidefinite means that all of its eigenvalues are non-negative
- \mathbf{K} is symmetric. Thus, we have $\mathbf{K} = \mathbf{V}\Lambda\mathbf{V}^T$
 - where \mathbf{V} is an orthonormal matrix \mathbf{v}_t and the diagonal matrix Λ contains the eigenvalues λ_t of \mathbf{K}
 - If \mathbf{K} is positive semidefinite, all eigenvalues are non-negative
 - Consider the feature map: $\phi : \mathbf{x}_i \mapsto (\sqrt{\lambda_t} v_{ti})_{t=1}^n \in \mathbb{R}^n$
 - We find that



$$\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = \sum_{t=1}^n \lambda_t v_{ti} v_{tj} = (\mathbf{V}\Lambda\mathbf{V}^T)_{ij} = K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \quad (6.13)$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \quad (6.14)$$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.15)$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.16)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \quad (6.17)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \quad (6.18)$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}')) \quad (6.19)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}' \quad (6.20)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.21)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.22)$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\mathbf{x})$ is a function from \mathbf{x} to \mathbb{R}^M , $k_3(\cdot, \cdot)$ is a valid kernel in \mathbb{R}^M , \mathbf{A} is a symmetric positive semidefinite matrix, \mathbf{x}_a and \mathbf{x}_b are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Polynomial kernel

- Polynomial kernel

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^M$$

- where M is the degree and c is a positive constant
- Proof: Polynomial kernel is valid
 - 1. $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)$ is a valid kernel with feature map $\mathbf{x} \rightarrow \begin{bmatrix} \mathbf{x} \\ \sqrt{c} \end{bmatrix}$
 - 2. According to

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \quad (6.18)$$

we can find that $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^M$ is a valid kernel

Gaussian kernel (RBF kernel)

- Gaussian kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2)$$

- where σ^2 is a positive constant
- The corresponding feature vector has infinite dimensionality
- Proof: Gaussian kernel is valid
 - 1. $\|\mathbf{x} - \mathbf{x}'\|^2 = \mathbf{x}^T \mathbf{x} + (\mathbf{x}')^T \mathbf{x}' - 2\mathbf{x}^T \mathbf{x}'$
 - 2. $k(\mathbf{x}, \mathbf{x}') = \exp(-\mathbf{x}^T \mathbf{x} / 2\sigma^2) \exp(\mathbf{x}^T \mathbf{x}' / \sigma^2) \exp(-(\mathbf{x}')^T \mathbf{x}' / 2\sigma^2)$
 - 3. According to

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.16)$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') \quad (6.14)$$

Gaussian kernel is valid



Sigmoidal kernel

- Sigmoidal kernel

$$k(\mathbf{x}, \mathbf{x}') = \tanh(a\mathbf{x}^T \mathbf{x}' + b)$$

- where a and b are two constants
- The Gram matrix of this function in general is not positive semidefinite
 - No corresponding feature mapping
- However, this form of kernel has been used in practice

Outline

- Dual representations
- Constructing kernels
- Support vector machines for classification
 - Maximum margin classifier
 - Overlapping class distributions
 - Multi-class SVMs
 - SVMs vs. logistic regression
- Support vector machines for regression



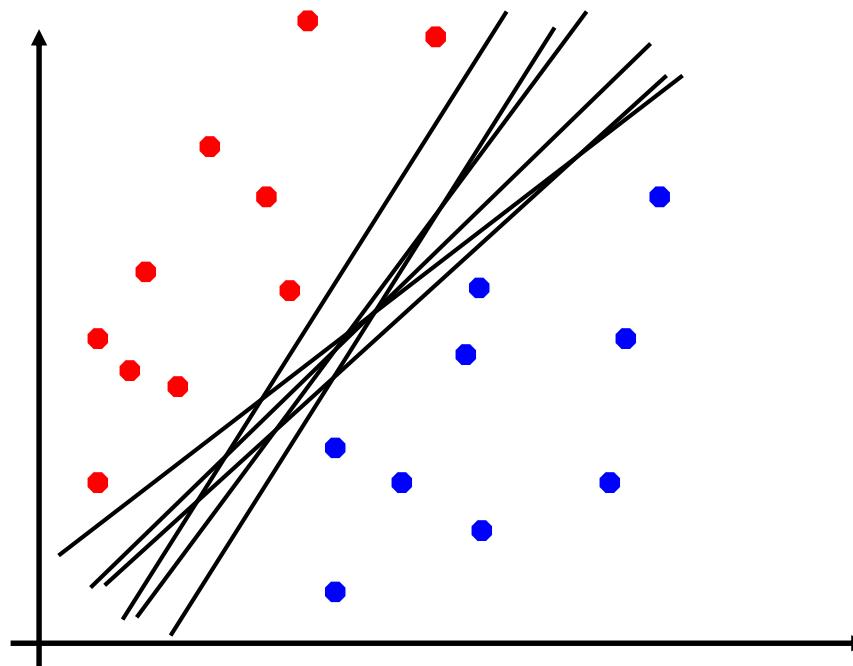
Linearly separable, binary-class classification

- Training data:
 - N training data points: $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$
 - The corresponding target label: t_1, t_2, \dots, t_N , where $t_n \in \{-1, 1\}$
- The decision function of SVMs
$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$
 - $\phi(\mathbf{x})$ denotes a fixed feature-space transformation
 - \mathbf{w} is the weight vector and b is the bias parameter
 - Binary classification: $\text{sign}(y(\mathbf{x}))$
- Linearly separable case
 - Correctly classify a positive data: $y(\mathbf{x}) > 0$
 - Correctly classify a negative data: $y(\mathbf{x}) < 0$
 - We can achieve $t_n y(\mathbf{x}_n) > 0$ for all training data points



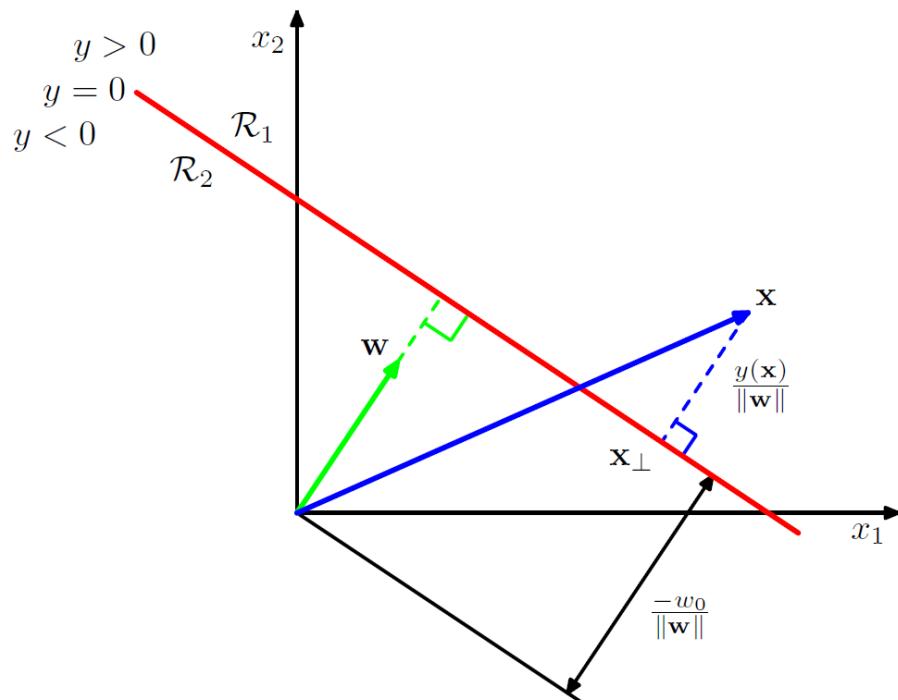
An example of a linearly separable training set

- For a linearly separable (in the feature space) data set, we may have many models that correctly classifies all training data
- Which of these classifiers is optimal?



Distance from a data point to decision boundary

- How to compute the distance between a positive point \mathbf{x} and the decision boundary $y = 0$?



$$\mathbf{x} = \mathbf{x}_{\perp} + r \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$\begin{cases} y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \\ y(\mathbf{x}_{\perp}) = \mathbf{w}^T \mathbf{x}_{\perp} + w_0 = 0 \end{cases}$$

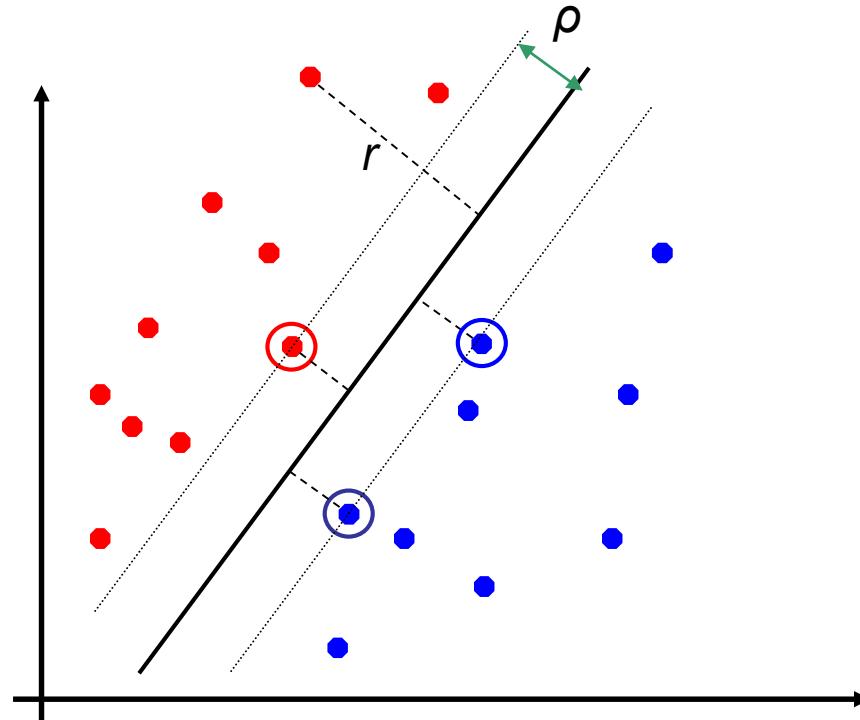
$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 = \mathbf{w}^T \mathbf{x}_{\perp} + w_0 + r \frac{\mathbf{w}^T \mathbf{w}}{\|\mathbf{w}\|}$$

$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$$



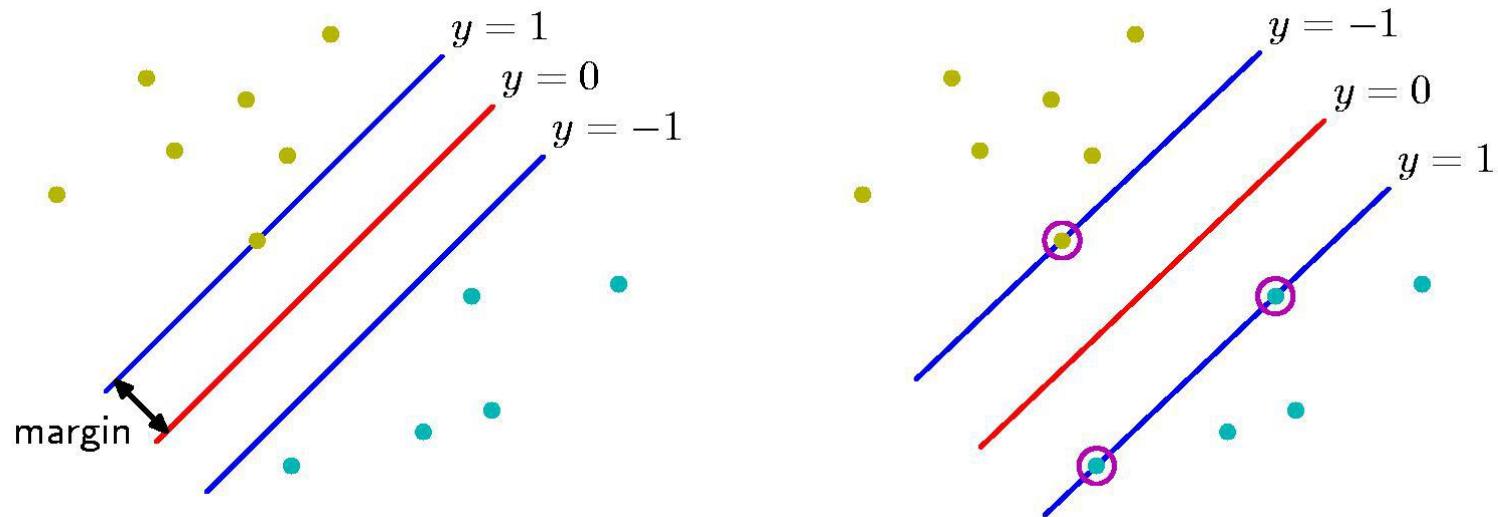
Margin

- The **margin** of an SVM classifier is the **smallest distance** between the decision boundary and any of the training points
- When multiple classifiers correctly all training data, we choose the one with the maximum margin



Maximum margin classifier

- Margin is the perpendicular distance between the decision boundary and the closest point



- Maximizing the margin leads to particular choice of the classifier

Margin maximization

- The perpendicular distance of a point \mathbf{x} from a hyperplane $y(\mathbf{x}) = 0$ is given by

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$$

- The objective function of the maximum margin solution is found by solving

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\}$$

➤ Directly optimizing this objective function is very complex!

An equivalent optimization problem

- Recall the distance $\frac{t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$
- If we make the **rescaling** $\mathbf{w} \rightarrow \kappa\mathbf{w}$ and $b \rightarrow \kappa b$, the distance from any point \mathbf{x}_n to the decision surface is unchanged.
- We rescale \mathbf{w} and b , which sets

$$t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) = 1$$

for **the point that is closest to the surface**

- In this case, all training data will stratify the following constraints

$$t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N.$$

An equivalent optimization problem

- Original optimization problem

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\}$$

- An equivalent optimization problem

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

subject to $t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N.$

- **Quadratic programming**

- Quadratic objective function with linear constraints
- Computational complexity is $O(M^3)$, where M is the number of optimization variables (number of data dimensionality)



Optimization using Lagrange multipliers

- The constrained optimization

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

subject to $t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N.$

- We introduce Lagrange multipliers $\{a_n \geq 0\}$, with one multiplier a_n for each constraint $t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1$
- The **Lagrangian function**

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n \{t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) - 1\}$$

Optimization using Lagrange multipliers

- The Lagrangian function

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n \left\{ t_n (\mathbf{w}^\top \phi(\mathbf{x}_n) + b) - 1 \right\}$$

- Several constraints

➤ Lagrange multiples are non-negative $a_n \geq 0$

$$\begin{aligned} & \text{➤ } \frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \boxed{\mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n)} \\ & \text{➤ } \frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial b} = 0 \quad \Rightarrow \quad 0 = \sum_{n=1}^N a_n t_n \quad \text{dual representation} \end{aligned}$$

Dual form of the optimization problem

- By eliminating \mathbf{w} and b from $L(\mathbf{w}, b, \mathbf{a})$, we get the dual representation of the maximum margin problem in which we maximize

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

subject to $a_n \geq 0, \quad n = 1, \dots, N,$

$$\sum_{n=1}^N a_n t_n = 0$$

kernel function

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$

- It can be solved by using quadratic programming with complexity $O(N^3)$, where N is the number of training data



Testing phase

- Decision function of SVMs

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

- Dual representation of SVMs

$$\mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n)$$

- For a test point \mathbf{x} , we classify it using a kernel function

$$y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

- How to determine the value of b ?

KKT conditions

- KKT (Karush-Kuhn-Tucker) conditions
- The solution to the problem of maximizing $f(\mathbf{x})$ subject to $g(\mathbf{x}) \geq 0$ obtained by optimizing the Lagrangian function $L(\mathbf{x}, \lambda) \equiv f(\mathbf{x}) + \lambda g(\mathbf{x})$ w.r.t. optimization variables \mathbf{x} and Lagrange multiplier λ subject to the conditions

$$g(\mathbf{x}) \geq 0$$

$$\lambda \geq 0$$

$$\lambda g(\mathbf{x}) = 0$$



KKT conditions in SVM optimization

- Karush-Kuhn-Tucker (KKT) conditions in SVMs:

$$\begin{aligned} a_n &\geq 0 \\ t_n y(\mathbf{x}_n) - 1 &\geq 0 \\ a_n \{t_n y(\mathbf{x}_n) - 1\} &= 0. \end{aligned}$$

- For every data point \mathbf{x}_n , either $a_n = 0$ or $t_n y(\mathbf{x}_n) = 1$
 - $a_n = 0$: This data point plays no role in making predictions for new data points in the decision function
 - $t_n y(\mathbf{x}_n) = 1$: This data point is called a support vector and lies on the maximum margin hyperplane in feature space
 - Only the support vectors retain, while the rest can be discarded

How to determine the value of b

- For a test point \mathbf{x} , we classify it using the decision function

$$y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

- For any support vector \mathbf{x}_n , we have $t_n y(\mathbf{x}_n) = 1$, i.e.,

$$t_n \left(\sum_{m \in S} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1$$

- The threshold b can be determined by calculating

$$b = \frac{1}{|S|} \sum_{n \in S} \left(t_n - \sum_{m \in S} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$$



➤ S : index set of support vectors; $|S|$: number of support vectors

How to determine the value of b

- For any support vector \mathbf{x}_n , we have

$$t_n \left(\sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1$$

- Derivation

$$\begin{aligned} t_n^2 \left(\sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) &= t_n \\ \Rightarrow \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b &= t_n \\ \Rightarrow \sum_{n \in \mathcal{S}} \left\{ \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right\} + |\mathcal{S}|b &= \sum_{n \in \mathcal{S}} t_n \\ \Rightarrow b &= \frac{1}{|\mathcal{S}|} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right) \end{aligned}$$



Testing

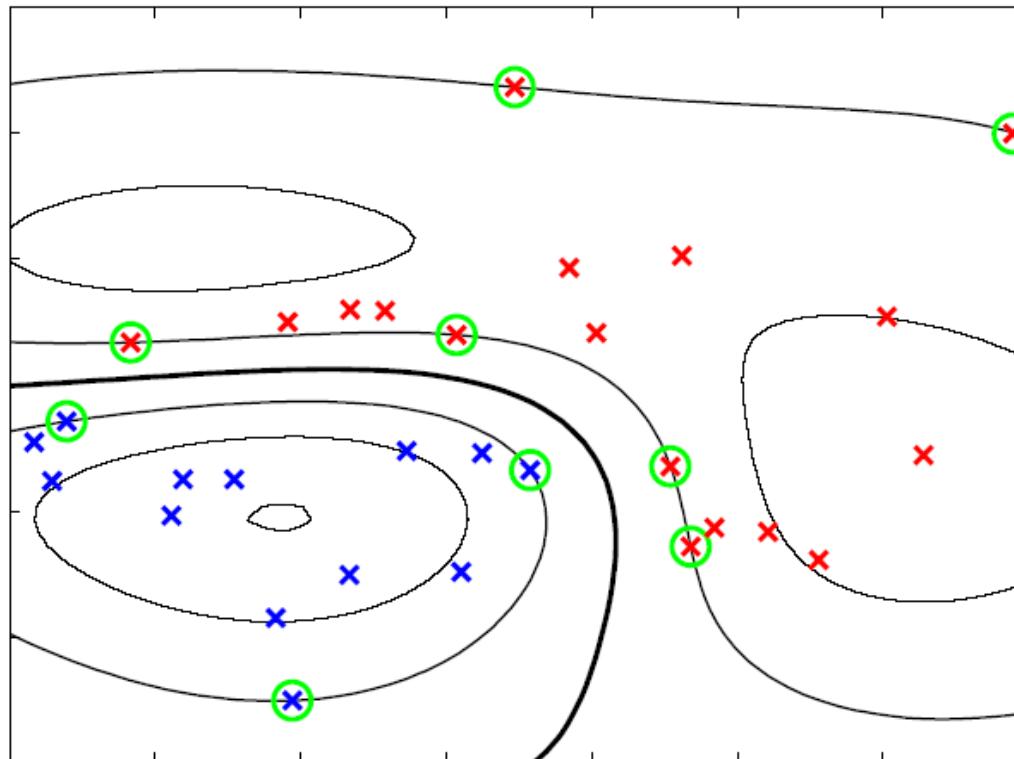
- The decision function has been determined

$$y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

- Given a new input \mathbf{x} , we can use $y(\mathbf{x})$ to predict the class of \mathbf{x} according to $\text{sign}(y(\mathbf{x}))$
- So far we assume the training data are linearly separable.
What if the data are not linearly separable?

An example

- Two-class synthetic data in a two-dimensional input space
- Gaussian kernel function
- Decision boundary, margin boundaries, and support vectors



Outline

- Dual representations
- Constructing kernels
- Support vector machines for classification
 - Maximum margin classifier
 - Overlapping class distributions
 - Multi-class SVMs
 - SVMs vs. logistic regression
- Support vector machines for regression



Overlapping class distributions

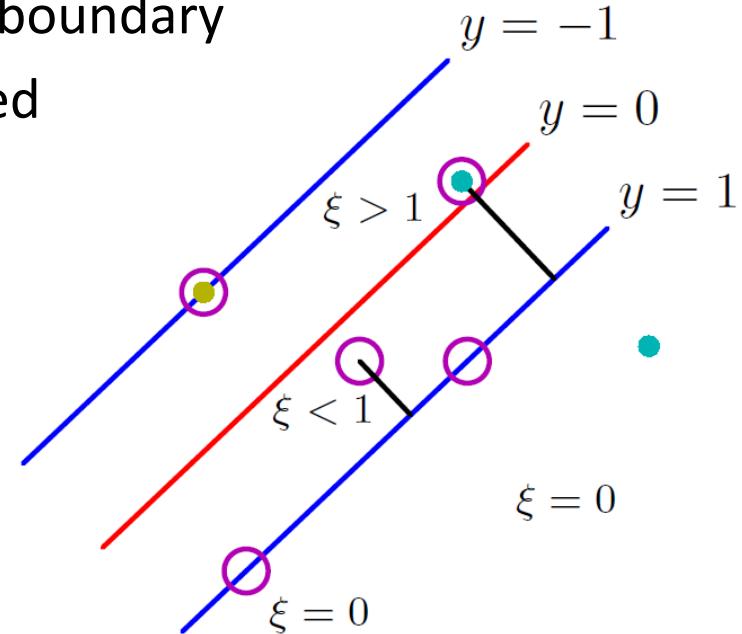
- In practice, the class-conditional distributions may overlap
 - Not linearly separable in the feature space
- We need a way to modify SVMs where misclassifying some training data is allowed
- Introduce a slack variable $\xi_n \geq 0$ for each training data \mathbf{x}_n
 - Linearly separable case: $t_n y(\mathbf{x}_n) \geq 1$, i.e.,

$$\underline{t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1}, \quad \downarrow \quad n = 1, \dots, N.$$

- Not linearly separable case: $\underline{t_n y(\mathbf{x}_n) \geq 1 - \xi_n}$
- Soft margin, which allows some training data to be misclassified

Slack variable

- Introduce slack variables, $\xi_n \geq 0$ where $n = 1, \dots, N$
 - New constraint: $t_n y(\mathbf{x}_n) \geq 1 - \xi_n$
 - $\xi_n = 0$ for data point \mathbf{x}_n on or inside the correct margin boundary
 - $0 < \xi_n < 1$ for data point \mathbf{x}_n inside the margin, but on the correct side of the decision boundary
 - $\xi_n = 1$ if \mathbf{x}_n is on the decision boundary
 - $\xi_n > 1$ if \mathbf{x}_n is wrongly classified
- The value of ξ_n indicates the degree of misclassification



Soft margin optimization: Primal form

- Hard margin optimization for linearly separable cases

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N.$$

- Soft margin optimization for general cases

最小化misclassified error

$$\arg \min_{\mathbf{w}, b, \{\xi_n\}} C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } t_n y(\mathbf{x}_n) \geq 1 - \xi_n \quad n = 1, \dots, N$$

$$\xi_n \geq 0 \quad n = 1, \dots, N$$

➤ where C is a positive constant



Lagrangian function

- Primal form

$$\arg \min_{\mathbf{w}, b, \{\xi_n\}} C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$$

subject to $t_n y(\mathbf{x}_n) \geq 1 - \xi_n \quad n = 1, \dots, N$

$$\xi_n \geq 0 \quad n = 1, \dots, N$$

- Lagrangian function with Lagrange multipliers $\{a_n\}$ and $\{\mu_n\}$

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N a_n \{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^N \mu_n \xi_n$$

Lagrangian function

- Lagrangian function

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N a_n \{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^N \mu_n \xi_n$$

- Some constraints

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^N a_n t_n = 0$$

→ 用 \mathbf{a} 取代此3變數

$$\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n = C - \mu_n.$$



Soft margin optimization: Dual form

- Dual form

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

subject to $0 \leq a_n \leq C \quad n = 1, \dots, N$

$$\sum_{n=1}^N a_n t_n = 0$$

- This optimization problem can be solved by quadratic programming

Optimizing b via KKT conditions

- KKT conditions:

$$a_n \geq 0$$

$$t_n y(\mathbf{x}_n) - 1 + \xi_n \geq 0$$

$$a_n (t_n y(\mathbf{x}_n) - 1 + \xi_n) = 0$$

an 不為0, 故另一項為0 $\mu_n \geq 0$

$$\xi_n \geq 0$$

$$\mu_n \xi_n = 0$$

- Consider a training data point \mathbf{x}_n with $0 < a_n < C$.

➤ $t_n y(\mathbf{x}_n) = 1 - \xi_n$

➤ $\xi_n = 0$ $\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n = C - \mu_n$

➤ $t_n y(\mathbf{x}_n) = 1$



Optimizing b via KKT conditions

- Consider a training data point \mathbf{x}_n with $0 < a_n < C$
 - $t_n y(\mathbf{x}_n) = 1$, i.e.,

$$t_n \left(\sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1$$

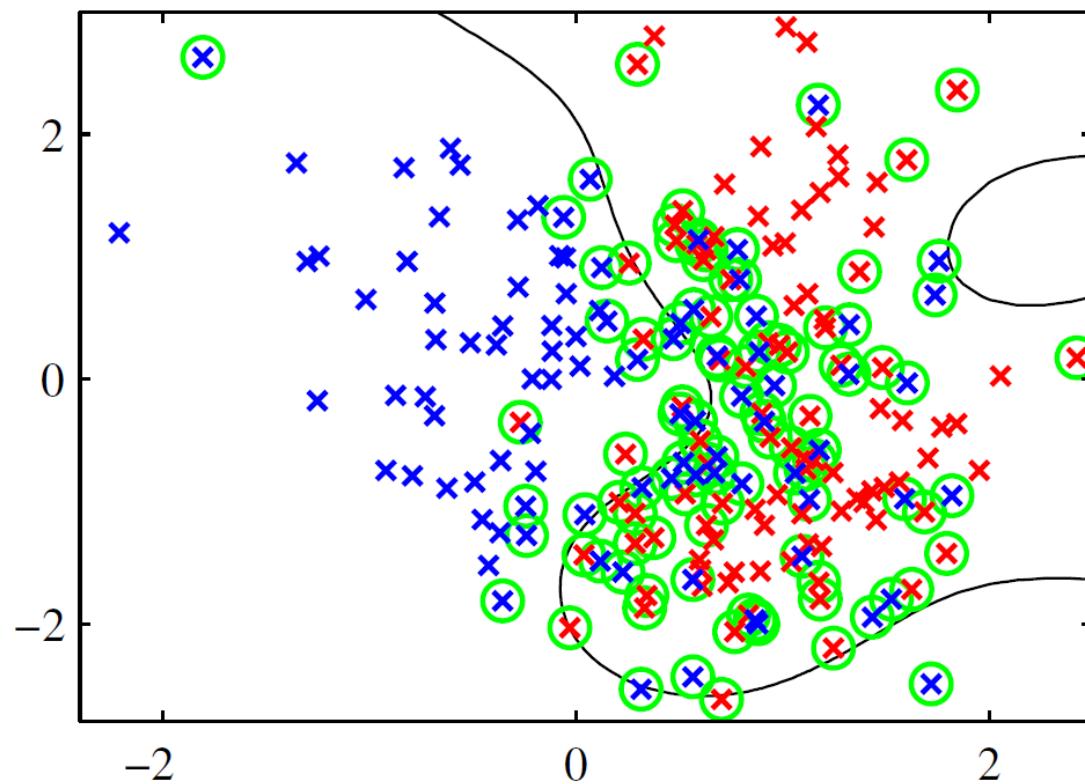
- A numerically stable solution

$$b = \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$$

- M : the set of indices of data points having $0 < a_n < C$
- S : the set of indices of the support vectors



An example



Optimization solvers for SVMs

- Quadratic programming
 - Many off-the-shelf solvers
 - Often infeasible due to the demanding computations and memory requirement
- Chunking (Vapnik, 1982) & protected conjugate gradients (Burges, 1998)
 - Try to remove the rows and columns of the kernel matrix that correspond to zero-valued Lagrange multipliers
- Decomposition methods (Osuna et al., 1996)
 - Solve a series of smaller quadratic programming problems
- Sequential minimal optimization (SMO) (Platt, 1999)
 - Consider two Lagrange multipliers at a time
 - Iterative processing: At each iteration, choose a pair of Lagrange multipliers and update their values



- A linear model for classification or regression

$$J(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \quad \text{where } \lambda \geq 0$$

- The dual representation of the linear model: a linear combination of training data

$$\mathbf{w} = \sum_{n=1}^N a_n \phi(\mathbf{x}_n) = \Phi^T \mathbf{a}$$

- The optimization variables change from \mathbf{w} to \mathbf{a}

- Why dual representation?
- The decision function is calculated in terms of kernel functions

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) = \mathbf{a}^T \Phi \phi(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

- A kernel function

$$K_{nm} = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

- Compute the inner product between two data points in a very high (even infinite) dimensional feature space
- Access data points in the low-dimensional input space
- Allow the PR or ML algorithms to work on a high dimensional feature space without explicitly computing $\phi(\mathbf{x})$



- Polynomial kernel

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^M$$

➤ where M is the degree and c is a positive constant

- Gaussian (RBF) kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2)$$

➤ where σ^2 is a positive constant

- Sigmoidal kernel

$$k(\mathbf{x}, \mathbf{x}') = \tanh(a\mathbf{x}^T \mathbf{x}' + b)$$

➤ where a and b are two constants

- The values of these hyperparameters are often determined by using cross-validation



Summary of SVMs

4/6

- SVMs a maximum margin classifier
- The **primal form** of the optimization problem

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N.$$

- Using Lagrange multipliers, we obtain its dual form

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

kernel function

$$\text{subject to } a_n \geq 0, \quad n = 1, \dots, N, \quad \sum_{n=1}^N a_n t_n = 0$$



Summary of SVMs

5/6

- Soft margin extension where slack variables are included
- Primal form

$$\arg \min_{\mathbf{w}, b, \{\xi_n\}} C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$$

subject to $t_n y(\mathbf{x}_n) \geq 1 - \xi_n \quad n = 1, \dots, N$

$\xi_n \geq 0 \quad n = 1, \dots, N$

- Dual form

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

subject to $0 \leq a_n \leq C \quad n = 1, \dots, N$

$$\sum_{n=1}^N a_n t_n = 0$$



- For a test point \mathbf{x} , we classify it using the decision function

$$y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

- We use quadratic programming or **SMO** to optimize $\{a_n\}$
- We exploit **KKT conditions** to determine the value of **b**

Outline

- Dual representations
- Constructing kernels
- Support vector machines for classification
 - Maximum margin classifier
 - Overlapping class distributions
 - Multi-class SVMs
 - SVMs vs. logistic regression
- Support vector machines for regression



Multiclass SVMs

- Support vector machine is fundamentally a two-class classifier
- In practice, we often deal with multi-class ($K > 2$) classification tasks
- Additional mechanism is required to combine multiple two-class SVM classifiers to handle multi-class classification
 - One-versus-the-rest
 - One-versus-one
 - DAGSVM (directed acyclic graph SVM)
 - ECOC (error-correcting output codes)

One-versus-the-rest

- Construct K two-class SVM classifiers, where K is the number of classes
- The k th SVM classifier $y_k(\mathbf{x})$ is trained by using data from class C_k as the positive examples and the data from the remaining $K - 1$ classes as the negative data, for $k = 1, 2, \dots, K$
- Make the prediction for an input data point \mathbf{x} by

$$y(\mathbf{x}) = \max_k y_k(\mathbf{x})$$

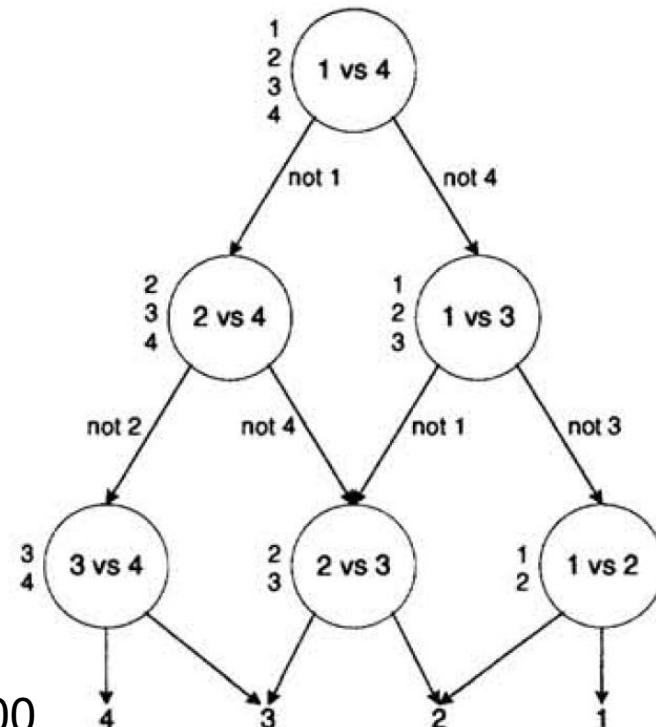
- Simple and intuitive
- Classifiers are trained separately, but comparing their decision values is used in prediction
- Imbalanced training data when training each classifier

One-versus-one

- Train an SVM classifier for each pair of classes i and j
 - Totally, $K(K - 1)/2$ SVM classifiers are trained
- Classify a test point according to which class has the highest number of votes
- Simple. No issue regarding imbalanced training data
- Decision ambiguities: Two or multiple classes get the same number of votes
- Computational issue: $K(K - 1)/2$ SVM classifiers

DAGSVM (directed acyclic graph SVM)

- Train an SVM classifier for each pair of classes
 - Totally, $K(K - 1)/2$ SVM classifiers are trained
- Organize these classifiers into a directed acyclic graph (a tree)
- Classify a test point by going through from the root to a leaf
- DAGSVM is more efficient than one-versus-one during testing



Platt et al. NIPS'00

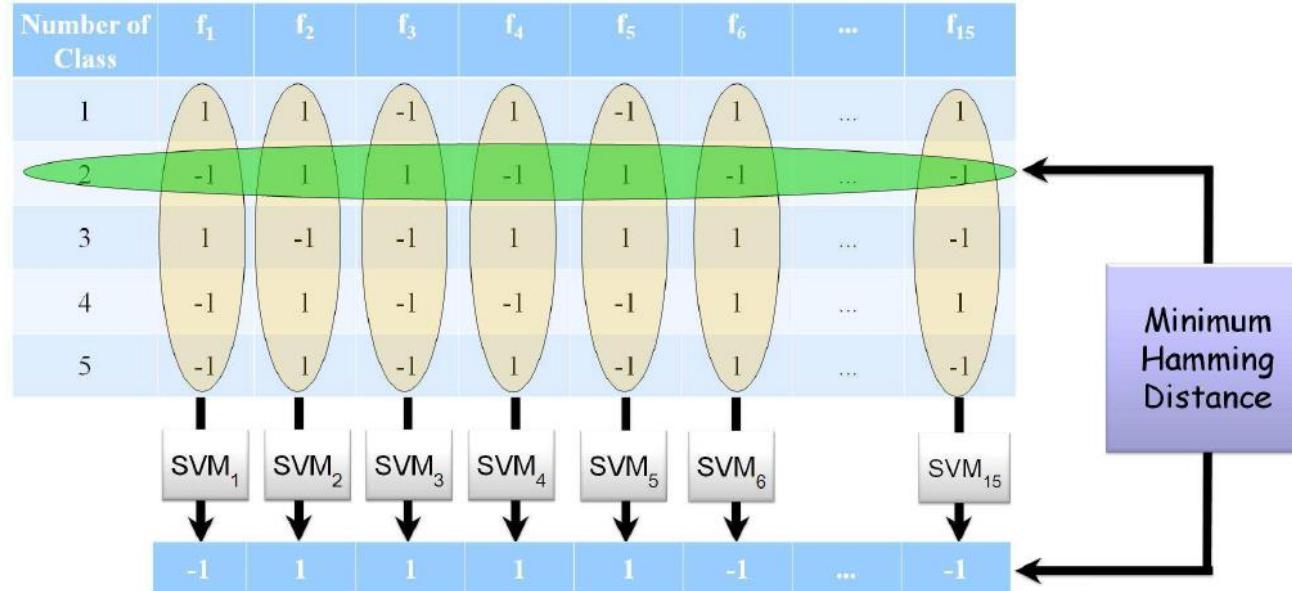


ECOC (error-correcting output codes)

- Multi-class classification is carried out based on error-correcting output codes
- Partition K classes into two disjoint sets. Train an SVM classifier by using data from one set as positive data and the rest as the negative data
- Repeat the above procedure n times
 - n SVM classifiers are trained
- Apply the n SVM classifiers to a test point
- Assign this test point to the class with the **smallest Hamming distance**

ECOC (error-correcting output codes)

- An example of $K = 5$ and $n = 15$
- A code matrix of size K by n
 - Each column represents a class partition
 - Each row is the code of the corresponding class



Outline

- Dual representations
- Constructing kernels
- Support vector machines for classification
 - Maximum margin classifier
 - Overlapping class distributions
 - Multi-class SVMs
 - SVMs vs. logistic regression
- Support vector machines for regression



Hinge error

- The primal form of soft margin SVMs

$$\arg \min_{\mathbf{w}, b, \{\xi_n\}} C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } t_n y(\mathbf{x}_n) \geq 1 - \xi_n \quad n = 1, \dots, N$$

$$\xi_n \geq 0 \quad n = 1, \dots, N$$

- Two cases for a training data point \mathbf{x}_n

- Case 1: $t_n y(\mathbf{x}_n) \geq 1 \Rightarrow \xi_n = 0$
- Case 2: $t_n y(\mathbf{x}_n) < 1 \Rightarrow \xi_n = 1 - t_n y(\mathbf{x}_n)$
- For simplicity, y_n denotes $y(\mathbf{x}_n)$

- Hinge error:** $E_{\text{SV}}(y_n t_n) = [1 - y_n t_n]_+$
 - $[\cdot]_+$ returns the positive part



Hinge error

- The primal form of soft margin SVMs

$$\arg \min_{\mathbf{w}, b, \{\xi_n\}} C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$$

$$\begin{aligned} \text{subject to } t_n y(\mathbf{x}_n) &\geq 1 - \xi_n & n = 1, \dots, N \\ \xi_n &\geq 0 & n = 1, \dots, N \end{aligned}$$

- An equivalent objective based on Hinge loss

$$\sum_{n=1}^N E_{\text{SV}}(y_n t_n) + \lambda \|\mathbf{w}\|^2$$

➤ where $\lambda = (2C)^{-1}$

Logistic regression

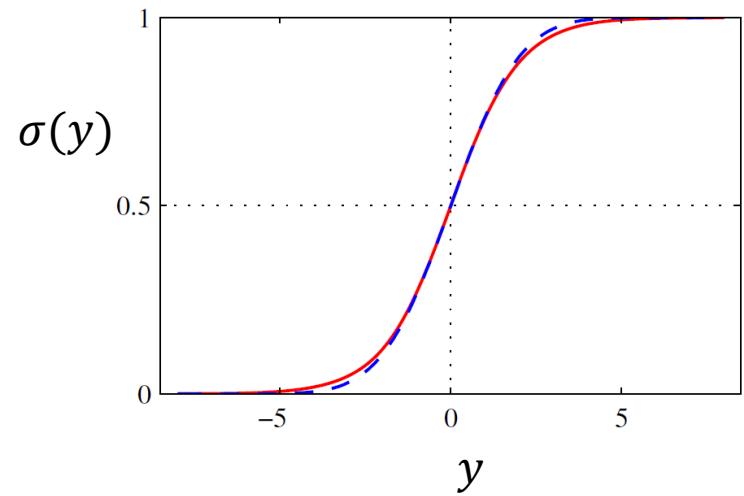
- Logistic regression estimates posterior probability for two-class classification [page 57 of slides “linear model for classification”]

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \exp(-y)} = \sigma(y) \end{aligned}$$

- where $y = \mathbf{w}^T \phi(\mathbf{x})$
- σ is the logistic sigmoid function

$$\sigma(y) = \frac{1}{1 + \exp(-y)}$$

- A property: $\sigma(-y) = 1 - \sigma(y)$



Logistic regression

- Negative log likelihood is used during training

$$-\ln \prod_{n=1}^N \sigma(y_n t_n) = -\sum_{n=1}^N \ln \sigma(y_n t_n)$$

- Note that for comparison with SVMs, we use target labels $t \in \{-1, 1\}$
- Taking the negative log likelihood with a quadratic regularizer, gives the form

$$\sum_{n=1}^N E_{\text{LR}}(y_n t_n) + \lambda \|\mathbf{w}\|^2$$

➤ where

$$E_{\text{LR}}(yt) = \ln(1 + \exp(-yt))$$

SVMs vs. Logistic regression

- SVMs

$$\sum_{n=1}^N E_{\text{SV}}(y_n t_n) + \lambda \|\mathbf{w}\|^2$$

➤ where

$$E_{\text{SV}}(y_n t_n) = [1 - y_n t_n]_+$$

- Logistic regression

$$\sum_{n=1}^N E_{\text{LR}}(y_n t_n) + \lambda \|\mathbf{w}\|^2$$

➤ where

$$E_{\text{LR}}(yt) = \ln(1 + \exp(-yt))$$

SVMs vs. Logistic regression

- Black curve

➤ misclassification error

- Red curve

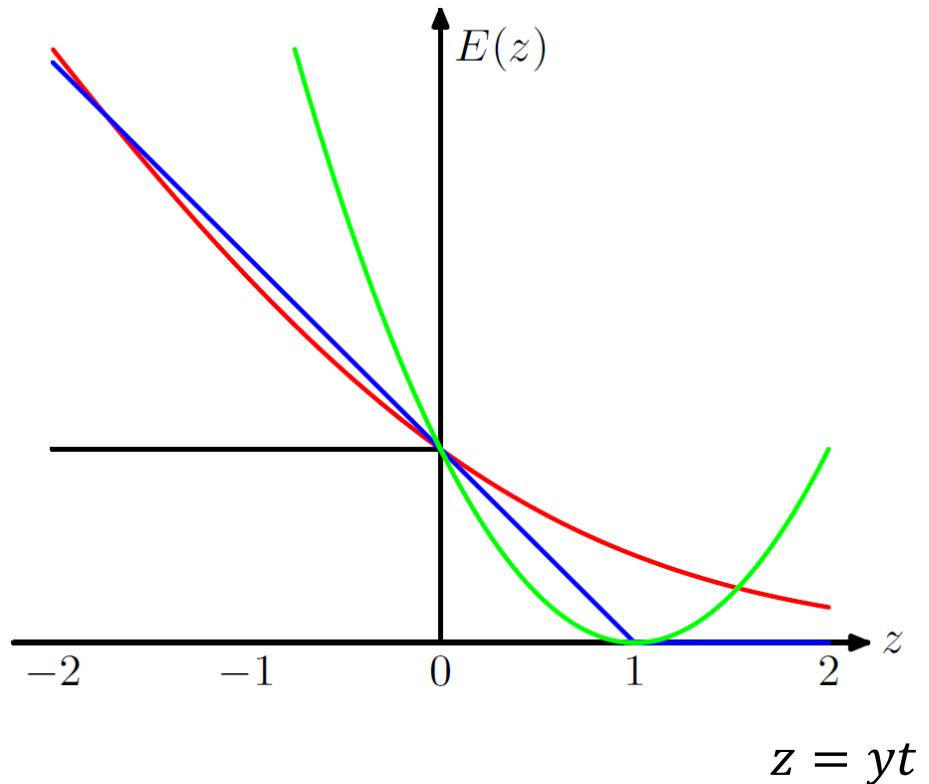
➤ Logistic regression error

$$E_{LR}(yt) = \ln(1 + \exp(-yt))$$

- Blue curve

➤ Hinge error

$$E_{SV}(y_n t_n) = [1 - y_n t_n]_+$$



Outline

- Dual representations
- Constructing kernels
- Support vector machines for classification
- Support vector machines for regression



Regression

- Training data:
 - N training data points: $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$
 - The corresponding target values: t_1, t_2, \dots, t_N
- In simple linear regression, we minimize a regularized error function given by

$$\frac{1}{2} \sum_{n=1}^N \{y_n - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- Sum-of-squared error
- A quadratic regularizer

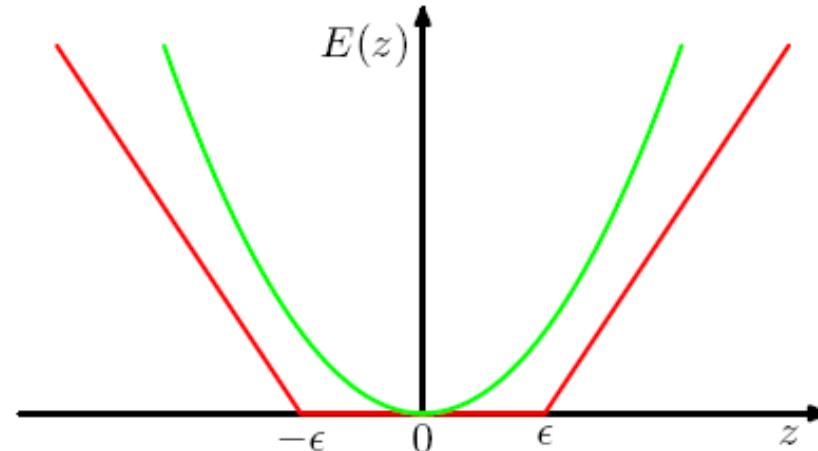
Error function in support vector regression

- In support vector regression (SVR), we define an ε -insensitive error function, which gives zero error if the absolute difference between the prediction $y(\mathbf{x})$ and the target t is less than ε where $\varepsilon > 0$

$$E_\varepsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \varepsilon; \\ |y(\mathbf{x}) - t| - \varepsilon, & \text{otherwise} \end{cases}$$

Green line:
quadratic error

Red line:
 ε -insensitive error



Objective function for SVR

- Objective function in SVR

$$C \sum_{n=1}^N E_\epsilon(y(\mathbf{x}_n) - t_n) + \frac{1}{2} \|\mathbf{w}\|^2$$

- where decision value $y(\mathbf{x})$ is given by $y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b$
- The first term in the objective minimizes the regression error
- The second term is a quadratic term for regularization, which helps maximize the margin in classification
- C is a positive constant



How to re-express the ε -insensitive error?

- In support vector classification, **a constraint** is associated with a training data \mathbf{x}_n , and **a slack variable** is added for soft margin

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

- ε -insensitive error function

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon; \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases}$$

- For \mathbf{x}_n , removing absolute value operator yields **two constraints**

$$y_n - \epsilon \leq t_n \leq y_n + \epsilon$$

- **Two slack variables** ξ_n and $\hat{\xi}_n$ are added for soft margin

$$t_n \leq y(\mathbf{x}_n) + \epsilon + \xi_n$$

$$t_n \geq y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n.$$



Optimization problem for SVR

- Original objective:

$$C \sum_{n=1}^N E_\epsilon(y(\mathbf{x}_n) - t_n) + \frac{1}{2} \|\mathbf{w}\|^2$$

- Primal form of SVR

$$C \sum_{n=1}^N (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2$$

subject to

$$\begin{aligned} t_n &\leq y(\mathbf{x}_n) + \epsilon + \xi_n & n = 1, \dots, N \\ t_n &\geq y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n & n = 1, \dots, N \\ \xi_n &\geq 0 \quad \hat{\xi}_n \geq 0 & n = 1, \dots, N \end{aligned}$$

Lagrangian function for SVR

- Primal form

$$C \sum_{n=1}^N (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } t_n \leq y(\mathbf{x}_n) + \epsilon + \xi_n \quad n = 1, \dots, N$$

$$t_n \geq y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n \quad n = 1, \dots, N$$

$$\xi_n \geq 0 \quad \hat{\xi}_n \geq 0 \quad n = 1, \dots, N$$

- **Lagrangian function** with Lagrange multipliers $\{a_n\}$, $\{\hat{a}_n\}$, $\{\mu_n\}$, and $\{\hat{\mu}_n\}$

$$\begin{aligned} L &= C \sum_{n=1}^N (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N (\mu_n \xi_n + \hat{\mu}_n \hat{\xi}_n) \\ &\quad - \sum_{n=1}^N a_n (\epsilon + \xi_n + y_n - t_n) - \sum_{n=1}^N \hat{a}_n (\epsilon + \hat{\xi}_n - y_n + t_n) \end{aligned}$$



Lagrangian function for SVR

- Lagrangian function

$$\begin{aligned} L = & C \sum_{n=1}^N (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N (\mu_n \xi_n + \hat{\mu}_n \hat{\xi}_n) \\ & - \sum_{n=1}^N a_n (\epsilon + \xi_n + y_n - t_n) - \sum_{n=1}^N \hat{a}_n (\epsilon + \hat{\xi}_n - y_n + t_n) \end{aligned}$$

- Some constraints

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N (a_n - \hat{a}_n) \phi(\mathbf{x}_n) \quad \frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n + \mu_n = C$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^N (a_n - \hat{a}_n) = 0 \quad \frac{\partial L}{\partial \hat{\xi}_n} = 0 \Rightarrow \hat{a}_n + \hat{\mu}_n = C$$

Dual form for SVR

- Dual form for SVR

$$\begin{aligned}\tilde{L}(\mathbf{a}, \hat{\mathbf{a}}) = & -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N (a_n - \hat{a}_n)(a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) \\ & - \epsilon \sum_{n=1}^N (a_n + \hat{a}_n) + \sum_{n=1}^N (a_n - \hat{a}_n) t_n\end{aligned}$$

subject to $0 \leq a_n \leq C \quad n = 1, \dots, N$

$0 \leq \hat{a}_n \leq C \quad n = 1, \dots, N$

$$\sum_{n=1}^N (a_n - \hat{a}_n) = 0$$

- This optimization problem can be solved by quadratic programming



Decision function of SVR

- The decision function of SVR

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

- The dual representation of weight vector \mathbf{w}

$$\mathbf{w} = \sum_{n=1}^N (a_n - \hat{a}_n) \phi(\mathbf{x}_n)$$

- The prediction for a new test point \mathbf{x}

$$y(\mathbf{x}) = \sum_{n=1}^N (a_n - \hat{a}_n) k(\mathbf{x}, \mathbf{x}_n) + b$$

- How to determine the value of b ?

Optimizing b via KKT conditions

- The corresponding KKT conditions of SVR

$$\begin{aligned} a_n(\epsilon + \xi_n + y_n - t_n) &= 0 \\ \widehat{a}_n(\epsilon + \widehat{\xi}_n - y_n + t_n) &= 0 \\ (C - a_n)\xi_n &= 0 \\ (C - \widehat{a}_n)\widehat{\xi}_n &= 0 \end{aligned}$$

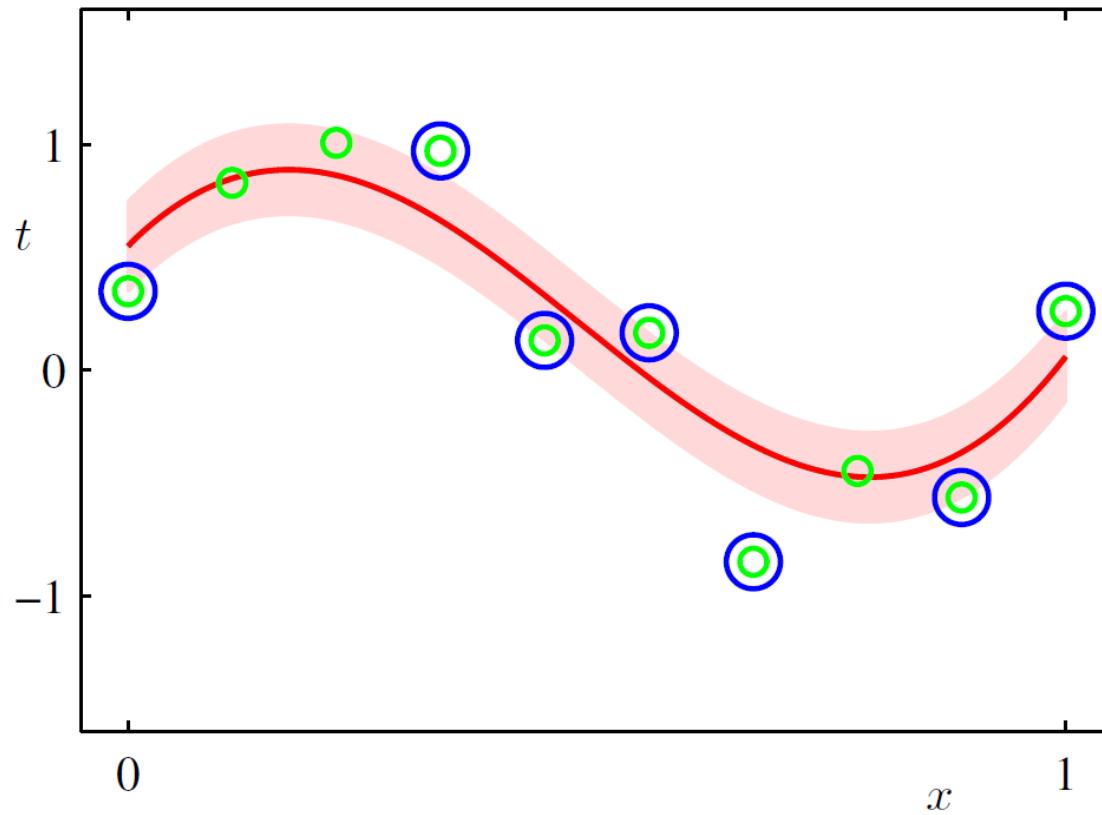
- Consider a training data point \mathbf{x}_n with $0 < a_n < C$
 - According to $(C - a_n)\xi_n = 0$, it implies $\xi_n = 0$
 - According to $a_n(\epsilon + \xi_n + y_n - t_n) = 0$, we have $\epsilon + y_n - t_n = 0$

$$\begin{aligned} b &= t_n - \epsilon - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) \\ &= t_n - \epsilon - \sum_{m=1}^N (a_m - \widehat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) \end{aligned}$$

Optimizing b via KKT conditions

- For each training data \mathbf{x}_n with $0 < a_n < C$ or $0 < \hat{a}_n < C$, we can estimate the value of b
- In practice, it is better to average over all such estimates of b

An example



Summary of SVR

- 1. Choose a kernel function
- 2. Solve the Lagrange multipliers $\{a_n\}$ and $\{\hat{a}_n\}$ in the dual form of SVR by using quadratic programming or SMO

$$\begin{aligned}\tilde{L}(\mathbf{a}, \hat{\mathbf{a}}) = & -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N (a_n - \hat{a}_n)(a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) \\ & - \epsilon \sum_{n=1}^N (a_n + \hat{a}_n) + \sum_{n=1}^N (a_n - \hat{a}_n) t_n\end{aligned}$$

subject to $0 \leq a_n \leq C \quad n = 1, \dots, N$

$0 \leq \hat{a}_n \leq C \quad n = 1, \dots, N$

$$\sum_{n=1}^N (a_n - \hat{a}_n) = 0$$

Summary of SVR

- 3. Optimize b via KKT conditions
- 4. Make a prediction for a testing data point \mathbf{x} via

$$y(\mathbf{x}) = \sum_{n=1}^N (a_n - \hat{a}_n) k(\mathbf{x}, \mathbf{x}_n) + b$$



References

- Dual representations
 - Chapter 6.1 in the PRML textbook
- Constructing kernels
 - Chapter 6.2 in the PRML textbook
- Support vector machines for classification
 - Chapters 7.1, 7.1.1, 7.1.2, and 7.1.3 in the PRML textbook
- Support vector machines for regression
 - Chapter 7.1.4 in the PRML textbook
- Lagrange multipliers and KKT conditions (optional)
 - Appendix E in the PRML textbook

Thank You for Your Attention!

感谢您们的聆听！

Yen-Yu Lin (林彥宇)

Email: lin@cs.nctu.edu.tw

URL: <https://www.cs.nctu.edu.tw/members/detail/lin>

