Support Vector Machines

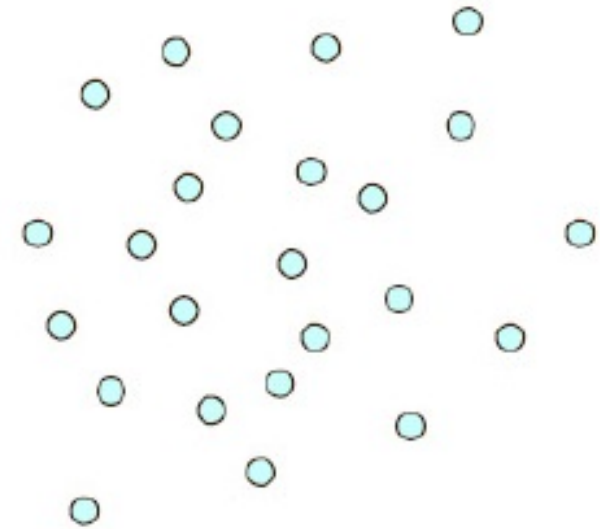
Hong Chang

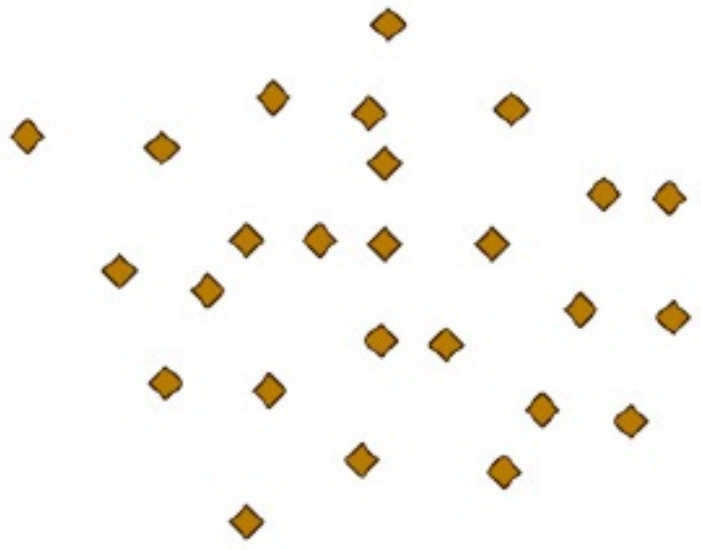
Institute of Computing Technology,
Chinese Academy of Sciences

Pattern Recognition and Machine Learning (Fall 2021)

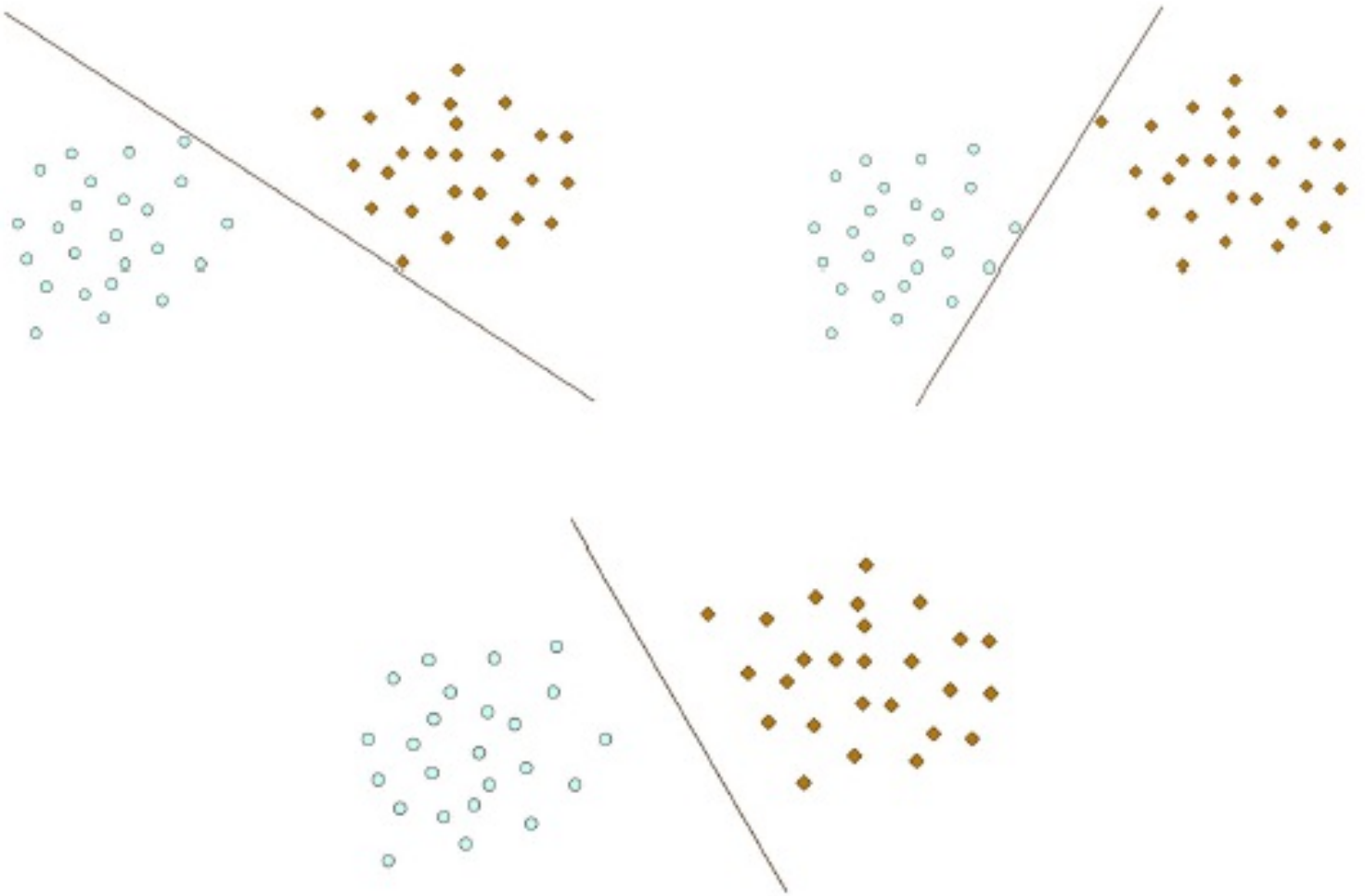
Outline I

Given a Data Set ...



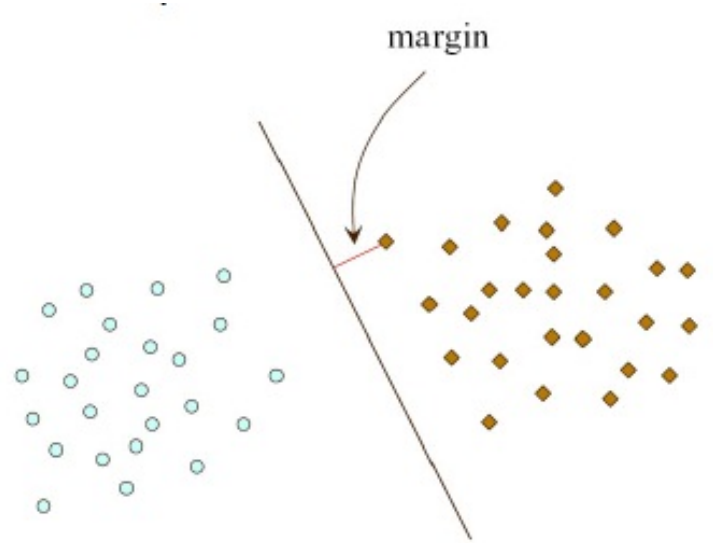


Which Separating Hyperplane is the Best?



Optimal Separating Hyperplane

- Margin of a separating hyperplane: distance to the separating hyperplane from the data point closest to it.



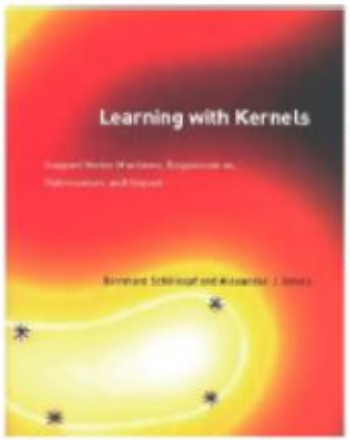
- Relationship between margin and generalization:
  There exist theoretical results from statistical learning theory showing that the separating hyperplane with the largest margin generalizes best (i.e., has smallest generalization error).

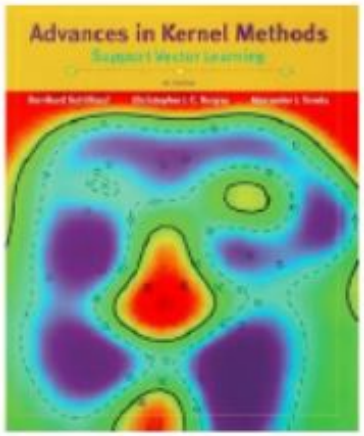
Support Vector Machine: History

- Linear SVM: Cortes and Vapnik, Support Vector Networks, Machine Learning, 1995.
- Kernelized SVM: Boster, Guyon, and Vapnik, A Training Algorithm for Optimal Margin Classifiers, workshop in COLT, 1992.
- SVR: Drucker, burges, Smola, and Vapnik, Support Vector Regression Machine, NIPS 1996.
- Generalization Analysis: Vapnik, The Nature of Statistical Learning Theory, Spinger, 1995. Statistical Learning Theory, Wiley&Sons, 1998.
- SMO: Platt, Fast Training of Support Vector Machines Using Sequential Minimal Optimization.
- SVM Light: Joachims, http://svmlight.joachims.org/
- LIBSVM: Chang and Lin, http://www.csie.ntu.edu.tw/ cjlin/libsvm
- Multi-Class SVM and StructSVM

Support Vector Machine: References

- Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond
- Advances in Large Margin Classifiers
- Advances in Kernel Methods: Support Vector Learning





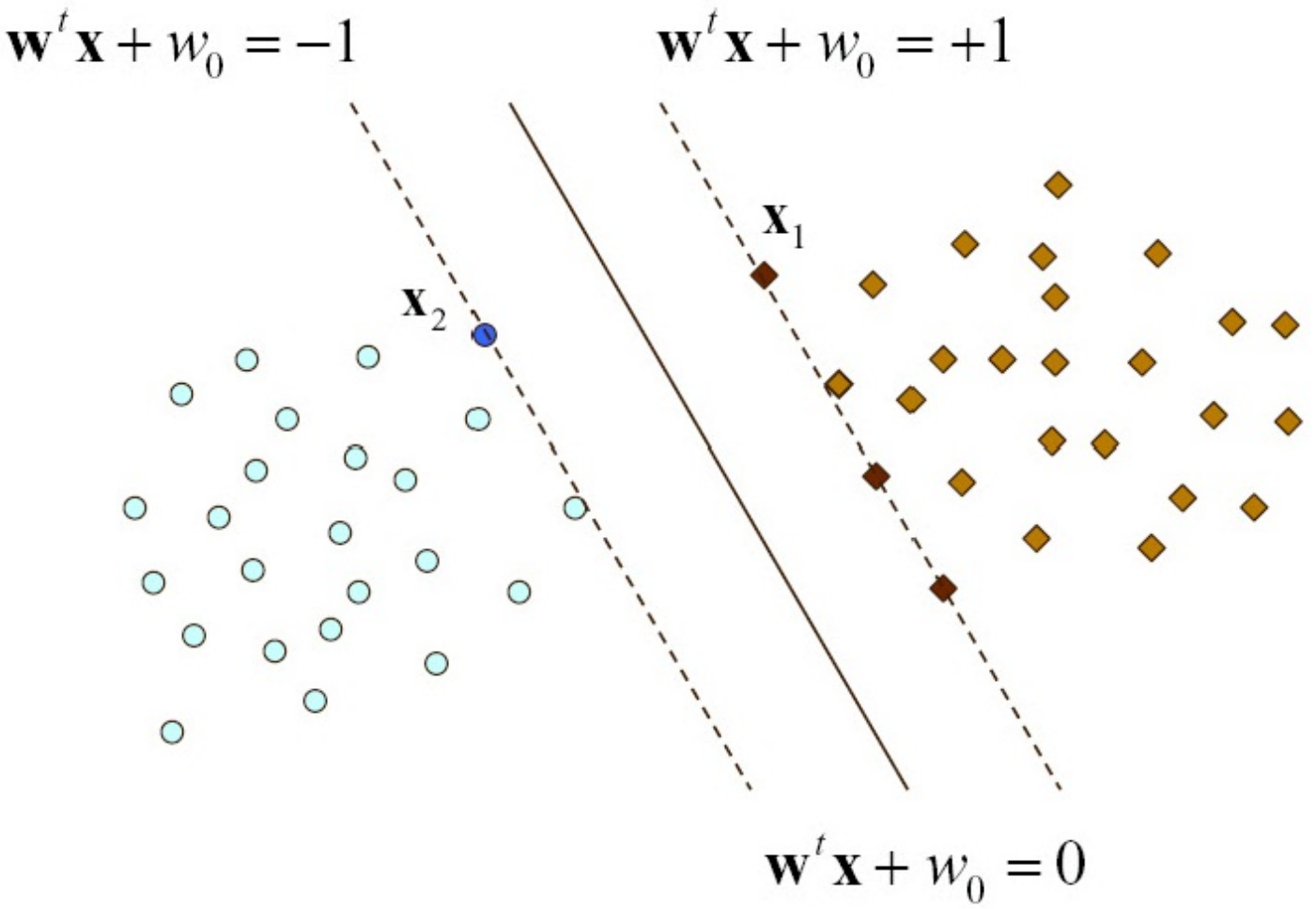
Canonical Optimal Separating Hyperplane

- Hard-margin case: data points from the two classes are assumed to be linearly separable.
- For $\lambda \neq 0$, $(\lambda \mathbf{w}, \lambda w_0)$ describes the same hyperplane as $(\mathbf{w}, w_0)$, i.e.,

$$\{\mathbf{x} | \mathbf{w}^T \mathbf{x} + w_0 = 0\} = \{\mathbf{x} | \lambda(\mathbf{w}^T \mathbf{x} + w_0) = 0\}$$

- With proper scaling of $\mathbf{w}$ and w_0, the points closest to the hyperplane satisfy $|\mathbf{w}^T \mathbf{x} + w_0| = 1$. Such a hyperplane is called a canonical separating hyperplane.
- The one that maximizes the margin is called the canonical optimal separating hyperplane.

Canonical Optimal Separating Hyperplane (2)



Canonical Optimal Separating Hyperplane (3)

- Let $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ be two closest points, one on each side of the hyperplane.
- Note that

$$\begin{aligned} \mathbf{w}^T\mathbf{x}^{(1)} + w_0 &= +1 \\ \mathbf{w}^T\mathbf{x}^{(2)} + w_0 &= -1 \end{aligned}$$

which imply

$$\mathbf{w}^T(\mathbf{x}^{(1)} - \mathbf{x}^{(2)}) = 2$$

Hence the margin can be given by

$$\gamma = \frac{1}{2}\frac{\mathbf{w}^T(\mathbf{x}^{(1)} - \mathbf{x}^{(2)})}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

- Maximizing the margin is equivalent to minimizing $\|\mathbf{w}\|$.

Inequality Constraints

- For all data points in the sample set $\mathcal{X} = \{\mathbf{x}^{(i)}, y^{(i)}\}$, we want $\mathbf{w}$ and w_0 to satisfy

$$\mathbf{w}^T\mathbf{x}^{(i)} + w_0 \begin{cases} \geq +1 & \text{if } y^{(i)} = +1 \\ \leq -1 & \text{if } y^{(i)} = -1 \end{cases}$$

- Equivalent form of inequality constraints:

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) \geq 1$$

- The above constraint require the data points to be some distance away from the hyperplane for better generalization.

Primal Optimization Problem

- Primal optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, w_0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ s.t. \quad & y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) \geq 1, \forall i \end{aligned}$$

- A quadratic programming (QP) problem
- In optimization theory, it is very common to turn a primal problem into a dual problem and then solve the latter instead.
- In our case, it also turns out to be more convenient to solve the dual problem (whose complexity depends on the sample size N) rather than the primal problem directly (whose complexity depends on the dimensionality D). The dual problem also makes it easy for a nonlinear extension using kernel functions.

Hinge Loss in SVM

- Primal optimization problem:

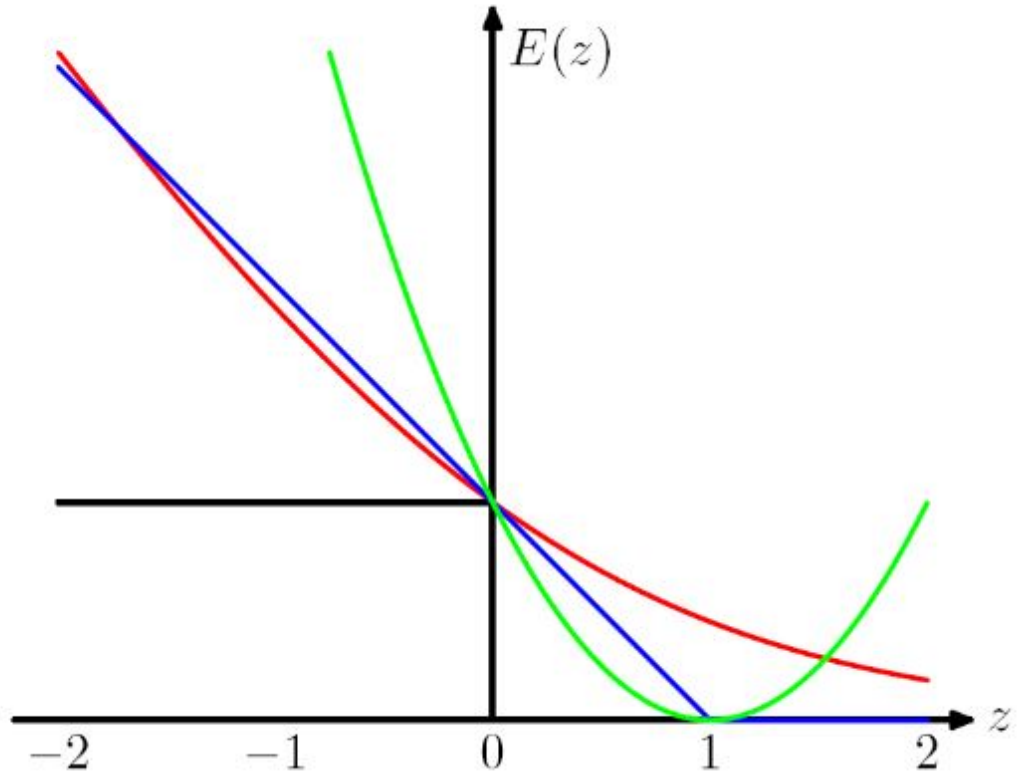
$$\begin{aligned} \min_{\mathbf{w}, w_0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ s.t. \quad & y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) \geq 1, \forall i \end{aligned}$$

- Equivalent loss function:

$$\min_{\mathbf{w}, w_0} \sum_{i=1}^{N} [1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0)]_+ + \lambda\|\mathbf{w}\|^2$$

Different Loss Functions

- square loss, negative log loss and hinge loss:



Lagrangian

- Lagrangian: ($\alpha_i \geq 0$ is Lagrange multiplier)

$$\begin{aligned} L_p(\mathbf{w}, w_0, \{\alpha_i\}) &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i [y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) - 1] \\ &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) + \sum_{i=1}^{N} \alpha_i \\ &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T \sum_{i=1}^{N} \alpha_i y^{(i)}\mathbf{x}^{(i)} - w_0 \sum_{i=1}^{N} \alpha_i y^{(i)} + \sum_{i=1}^{N} \alpha_i \end{aligned}$$

  - The inequality constraints of the primal problem are incorporated into the second term of the Lagrangian.
  - The optimal solution is a saddle point which minimizes L_p w.r.t. the primal variables $\mathbf{w}$, w_0 and maximizes L_p w.r.t. the dual variables α_i.

Eliminating Primal Variables

- Setting the gradients of L_p w.r.t. $\mathbf{w}$ and w_0 to 0:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_i \alpha_i y^{(i)} \mathbf{x}^{(i)} \tag{1}$$

$$\frac{\partial L_p}{\partial w_0} = 0 \quad \Rightarrow \quad \sum_i \alpha_i y^{(i)} = 0 \tag{2}$$

- Plugging (1) and (2) into L_p gives the objective function L_d for the dual problem:

$$\begin{aligned} L_d(\{\alpha_i\}) &= -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_i \alpha_i \\ &= -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)} + \sum_i \alpha_i \end{aligned}$$

Dual Optimization Problem

- Dual optimization problem:

$$\max_{\boldsymbol{\alpha}} \quad \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)}$$

$$s.t. \quad \sum_i \alpha_i y^{(i)} = 0 \text{ and } \alpha_i \geq 0, \forall i$$

- This is also a QP problem, but its complexity depends on the sample size N (rather than the input dimensionality D):
  - Time complexity: $O(N^3)$
  - Space complexity: $O(N^2)$

Support Vectors

- Most of the dual variables vanish with $\alpha_i = 0$. They are points lying beyond the margin with no effect on the hyperplane.
- Support vectors: $\mathbf{x}^{(i)}$ with $\alpha_i > 0$, hence the name support vector machine (SVM).
- The expected test error rate has an upper bound which depends on the number of support vectors:

$$E_N[P(\text{error})] \leq \frac{E_N[\sharp \text{ of SVs}]}{N}$$

where $E_N[\cdot]$ denotes the expectation over training sets of size N.

Support Vectors (2)

- Computation of primal variables:
  - From (1) we get

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

    where $\mathcal{SV}$ denotes the set of support vectors.
  - The support vectors must lie on the margin, so they should satisfy

$$y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + w_0) = 1 \text{ or } w_0 = y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)}$$

    For numerical stability, all support vectors are used to compute w_0:

$$w_0 = \frac{1}{|\mathcal{SV}|} \sum_{\mathbf{x}^{(i)} \in \mathcal{SV}} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})$$

Discriminant Function

- Discriminant function:

$$\begin{aligned} g(\mathbf{x}) &= \mathbf{w}^T\mathbf{x} + w_0 \\ &= \Big(\sum_{\mathbf{x}^{(i)}\in\mathcal{SV}} \alpha_i y^{(i)}\mathbf{x}^{(i)}\Big)^T\mathbf{x} + \frac{1}{|\mathcal{SV}|}\sum_{\mathbf{x}^{(i)}\in\mathcal{SV}} (y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}) \end{aligned}$$

- Classification rule during testing:

$$\text{Choose} \begin{cases} C_1 & \text{if } g(\mathbf{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}$$

Hard-Margin Classifier

Solution of the Primal Optimization Problem

Suppose that $\boldsymbol{\alpha}^* = (\alpha_1^*, \ldots, \alpha_N^*)$ are the optimal solution of the dual optimization problem, there exists j such that $\alpha_j^* > 0$, and we have,

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

$$w_0 = \frac{1}{|\mathcal{SV}|} \sum_{\mathbf{x}^{(i)} \in \mathcal{SV}} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})$$

- Discriminant function:

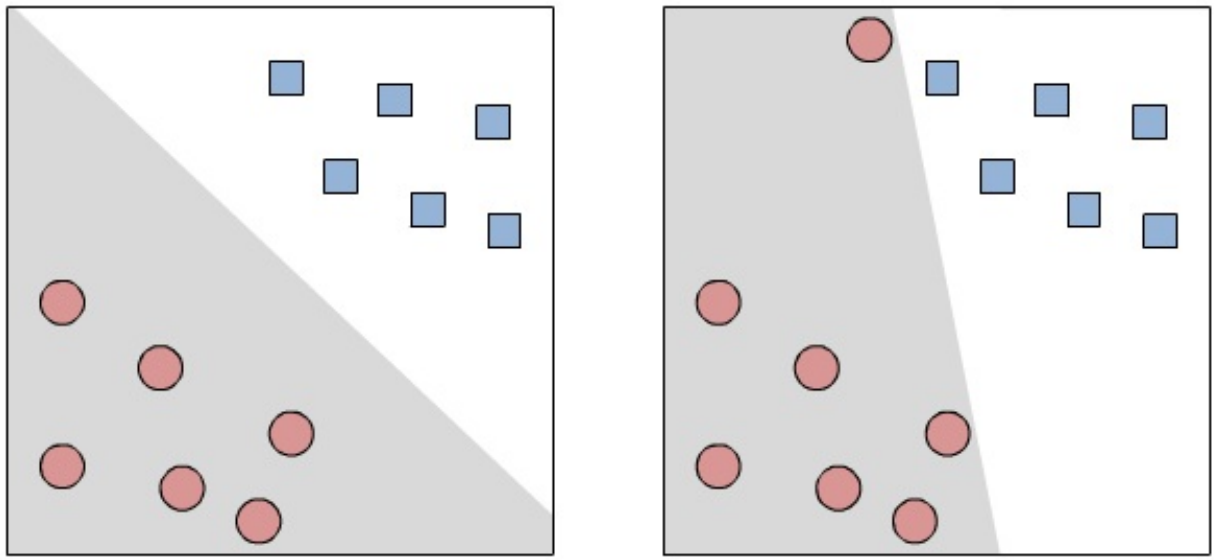
$$\begin{aligned} g(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + w_0 \\ &= \Big(\sum_{\mathbf{x}^{(i)} \in \mathcal{SV}} \alpha_i y^{(i)} \mathbf{x}^{(i)} \Big)^T \mathbf{x} + \frac{1}{|\mathcal{SV}|} \sum_{\mathbf{x}^{(i)} \in \mathcal{SV}} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)}) \end{aligned}$$

Generalization to Multiple Classes

- One way to handle multiple classes is to define K two-class problems, each separating one class from all other classes combined.
- An SVM $g_k(\mathbf{x})$ is learned for each two-class problem.
- Classification rule during testing:

$$\text{Choose } C_k \text{ if } k = \arg\max_j g_j(\mathbf{x})$$

The Need of Regularization - Outliers

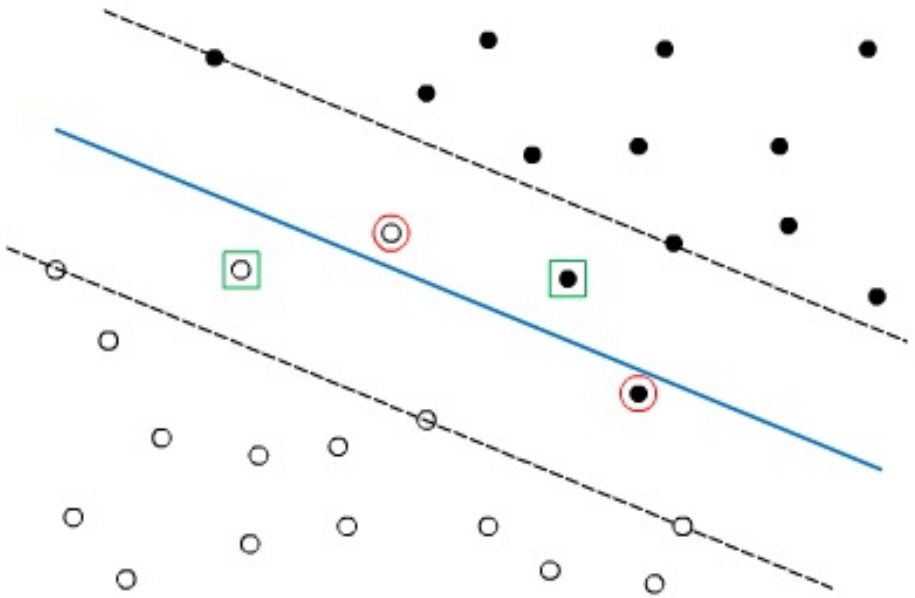


- Sometimes finding a hard separating hyperplane is not exactly what we want.

Relaxing the Constraints

- In practice, a separating hyperplane may not exist, possibly due to a high noise level which causes a large overlap of the classes.
- Even if a separating hyperplane exists, it is not always the best solution to the classification problem when there exist outliers in the data.
- A mislabeled example can become an outlier which affects the location of the separating hyperplane.

Soft-Margin



- □: samples falling inside the band but correctly classified
- ◯: samples misclassified

Slack Variables

- A soft-margin SVM allows for the possibility of violating the inequality constraints

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) \geq 1$$

by introducing slack variables

$$\xi_i \geq 0, i = 1, \ldots, N$$

which store the derivation from the margin.

- Relaxed separation constraints:

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) \geq 1 - \xi_i$$

Penalty

- By making ξ_i large enough, the constraint on $(\mathbf{x}^{(i)}, y^{(i)})$ can always be met. To avoid trivial solution, we should penalize ξ_i in the objective function.
- Three cases:
  - $\xi_i = 0$: no slack with $\mathbf{x}^{(i)}$ (no penalty)
  - $0 < \xi_i < 1$: $\mathbf{x}^{(i)}$ lies on the right side of the hyperplane but in the margin (small penalty)
  - $\xi_i > 1$: $\mathbf{x}^{(i)}$ lies on the wrong side of the hyperplane (large penalty)
- Number of misclassifications: $\sharp\{\xi_i > 1\}$
- Number of nonseparable instances: $\sharp\{\xi_i > 0\}$
- Soft error as additional penalty term:

$$\sum_i \xi_i$$

Soft-Margin SVM Formulation

- The goal now is
  - To make the margin as large as possible
  - To keep the number of points with $\xi_i > 0$ as small as possible
- Reformulated primal optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, w_0, \boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \forall i \end{aligned}$$

  - C is a regularization parameter, which trades off between margin maximization and training error minimization.

Primal Optimization Problem

- Lagrangian:

$$
\begin{aligned}
L_p &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i - \sum_{i=1}^{N} \alpha_i [y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) - 1 + \xi_i] - \sum_i \mu_i \xi_i \\
&= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T \sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{(i)} - w_0 \sum_{i=1}^{N} \alpha_i y^{(i)} \\
&\quad + \sum_{i=1}^{N} \alpha_i + \sum_{i=1}^{N} (C - \alpha_i - \mu_i)\xi_i
\end{aligned}
$$

  - α_i and μ_i are Lagrange multipliers.
- Both the misclassified instances and the ones in the margin are penalized for better generalization, though the latter ones would be correctly classified during testing.

Lagrange Dual

- Setting the gradients of L_p w.r.t. $\mathbf{w}$, w_0 and ξ to 0:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_i \alpha_i y^{(i)} \mathbf{x}^{(i)} \tag{3}$$

$$\frac{\partial L_p}{\partial w_0} = 0 \quad \Rightarrow \quad \sum_i \alpha_i y^{(i)} = 0 \tag{4}$$

$$\frac{\partial L_p}{\partial \xi_i} = 0 \quad \Rightarrow \quad C - \alpha_i - \mu_i = 0 \tag{5}$$

Dual Optimization Problem

- Dual optimization problem:

$$\max_{\boldsymbol{\alpha}} \quad \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)}$$

$$s.t. \quad \sum_i \alpha_i y^{(i)} = 0 \text{ and } 0 \leq \alpha_i \leq C, \forall i$$

- Similar to the hard-margin case, instances that are not support vectors vanish with $\alpha_i = 0$. $\mathbf{w}$ and w_0 can be computed similarly based on the support vectors.

Soft-Margin Classifier

Solution of the Primal Optimization Problem

Suppose that $\boldsymbol{\alpha}^* = (\alpha_1^*, \ldots, \alpha_N^*)$ are the optimal solution of the dual optimization problem, there exists j such that $0 < \alpha_j^* < C$, and we have,

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

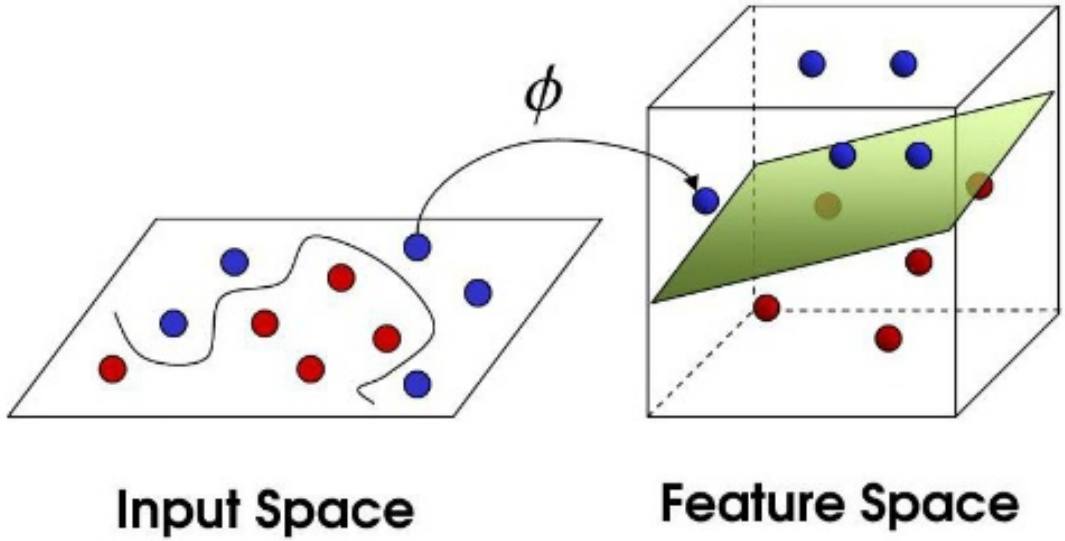
$$w_0 = \frac{1}{|\mathcal{SV}|} \sum_{\mathbf{x}^{(i)} \in \mathcal{SV}} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})$$

- Discriminant function:

$$\begin{aligned} g(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + w_0 \\ &= \Big(\sum_{\mathbf{x}^{(i)} \in \mathcal{SV}} \alpha_i y^{(i)} \mathbf{x}^{(i)} \Big)^T \mathbf{x} + \frac{1}{|\mathcal{SV}|} \sum_{\mathbf{x}^{(i)} \in \mathcal{SV}} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)}) \end{aligned}$$

Key Ideas of Kernel Methods

- Instead of defining a nonlinear model in the original (input) space, the problem is mapped to a new (feature) space by performing a nonlinear transformation using suitably chosen basis functions.
- A linear model is then applied in the new space.
- The basis functions are often defined implicitly via defining kernel functions directly.



Kernel

Definition: Kernel

Given the input space $\mathcal{X}$ and the feature space $\mathcal{H}$ (Hilbert Space), if there exists a mapping ϕ from $\mathcal{X}$ to $\mathcal{H}$, such that for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, the function $K(\mathbf{x}, \mathbf{x}')$ satisfies $K(\mathbf{x}, \mathbf{x}') =< \phi(\mathbf{x}), \phi(\mathbf{x}') >$, then $K(\mathbf{x}, \mathbf{x}')$ is called a kernel, where ϕ is the mapping function.

- For a given kernel, the feature space $\mathcal{H}$ and mapping function ϕ is usually not unique.
- Example: $\mathcal{X} = \mathcal{R}^2, K(\mathbf{x}, \mathbf{x}') =< \mathbf{x}, \mathbf{x}' >^2$
  1. $\mathcal{H} = \mathcal{R}^3, \phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$
  2. $\mathcal{H} = \mathcal{R}^3, \phi(\mathbf{x}) = \frac{1}{\sqrt{2}}((x_1 - x_2)^2, 2x_1x_2, (x_1 + x_2)^2)^T$
  3. $\mathcal{H} = \mathcal{R}^4, \phi(\mathbf{x}) = (x_1^2, x_1x_2, x_1x_2, x_2^2)^T$

Valid Kernel

- Question: How can we directly define a kernel, not through ϕ?
- Question: Given a kernel $K(\mathbf{x}, \mathbf{x}')$, how can we tell if it's a valid kernel, i.e., can we tell if there is some feature mapping ϕ so that $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$ for all $\mathbf{x}, \mathbf{x}'$.

Mercer's Theorem

Let $K : \mathcal{R}^d \times \mathcal{R}^d \to \mathcal{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$, the corresponding kernel matrix is symmetric positive semi-definite.

- Kernel (Gram) Matrix: consider some finite set of N points $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$, let a square, N by N matrix be defined so that its (i, j)-th entry is given by $K_{ij} = K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$.

$$\mathbf{K} = \begin{bmatrix} K_{11} & K_{12} & \ldots & K_{1N} \\ K_{21} & K_{22} & \ldots & K_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ K_{N1} & K_{N2} & \ldots & K_{NN} \end{bmatrix}$$

Positive Definite Kernels

- Naming conventions:
  - Linear algebra: positive semi-definite vs. positive definite
  - Kernel theory: positive definite vs. strictly positive definite
- Positive definiteness implies positivity on the diagonal: $K(\mathbf{x}, \mathbf{x}) \geq 0, \forall \mathbf{x} \in \mathcal{X}$.
- Kernels can be regarded as generalized inner products. Any inner product is a kernel.
- Cauchy-Schwarz inequality for kernels:
  If K is a positive definite kernel and $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, then

$$|K(\mathbf{x}, \mathbf{x}')|^2 \leq K(\mathbf{x}, \mathbf{x}) K(\mathbf{x}', \mathbf{x}')$$

Kernel trick in Application

- Kernel trick:
  Given an algorithm which is formulated in terms of a positive-definite (PD) kernel K_0, one can construct an alternative algorithm by replacing K_0 by another PD kernel K.
- A special case of the kernel trick:
  Suppose K_0 is the inner product in the input space (i.e., a linear kernel), then the same algorithm can also be used for any other nonlinear kernel K by replacing K_0 by K. This is the common kernelization strategy for devising kernel methods.

Feature Maps

- We define a feature map ϕ from $\mathcal{X}$ into the space of functions (function space) $\mathcal{R}^{\mathcal{X}} = \{f : \mathcal{X} \to \mathcal{R}\}$, where each function f in $\mathcal{R}^{\mathcal{X}}$ maps $\mathcal{X}$ into $\mathcal{R}$.
- $\phi(\mathbf{x})$ denotes the function that assigns $K(\mathbf{x}, \mathbf{x}')$ to $\mathbf{x}' \in \mathcal{X}$, i.e.

$$\phi(\mathbf{x})(\cdot) = K(\cdot, \mathbf{x}')$$

- In this way, each pattern $\mathbf{x}$ is turned into a function $\phi(\mathbf{x})$, representing the similarity of $\mathbf{x}$ to all other points in $\mathcal{X}$.

Reproducing Kernel Hilbert Space (RKHS)

- A reproducing kernel Hilbert space (RKHS) is a Hilbert space associated with a kernel that reproduces every function in the space.
- Reproducing Kernel: A kernel $K : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ is called reproducing if it satisfies the following two properties:
  1. For any $\mathbf{x}_0 \in \mathcal{X}$, $K(\mathbf{x}, \mathbf{x}_0)$ is a function of $\mathbf{x}$ in the space $\mathcal{H}$.
  2. For any $\mathbf{x} \in \mathcal{X}$ and $f \in \mathcal{H}$, we have $f(\mathbf{x}) =< f(\cdot), K(\cdot, \mathbf{x}) >$.
- The mapping function can be naturally defined as $\phi(\mathbf{x}) = K(\cdot, \mathbf{x})$.
- We can see that the following equation holds:

$$\phi(\mathbf{x})^T \phi(\mathbf{x}') =< K(\cdot, \mathbf{x}), K(\cdot, \mathbf{x}') >= K(\mathbf{x}, \mathbf{x}')$$

- Note that for each $f(\cdot) = \sum_{i=1}^{N} \alpha_i K(\cdot, \mathbf{x}^{(i)})$ (we define a vector space consisting of functions of this form), we have $< f(\cdot), K(\cdot, \mathbf{x}) >= \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}^{(i)}) = f(\mathbf{x})$. It defines an inner product in the space $\mathcal{H}$.

Formal Definition of RKHS

Reproducing Kernel Hilbert Space (RKHS)

Let $\mathcal{X}$ be a nonempty set and $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathcal{R}$. Then $\mathcal{H}$ is called a reproducing kernel Hilbert space endowed with the inner product $< \cdot, \cdot >$ (and the norm $\|f\| = \sqrt{< f, f >}$) if there exists a function $K : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ with the following two properties:

1. K has the reproducing property:

$$< f, K(\cdot, \mathbf{x}) >= f(\mathbf{x})$$

for all $f \in \mathcal{H}$. In particular,

$$< K(\cdot, \mathbf{x}), K(\cdot, \mathbf{x}') >= K(\mathbf{x}, \mathbf{x}').$$

2. K spans $\mathcal{H}$, i.e., $\mathcal{H} = \overline{span\{K(\cdot, \mathbf{x}) | \mathbf{x} \in \mathcal{X}\}}$ (hence $\mathcal{H}$ is closed), where $\overline{X}$ denotes the completion of the set X.

Some Common Kernel Functions

- Polynomial kernel:

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^q$$

where q is the degree.

- E.g., when $q = 2$ and $D = 2$,

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= (\mathbf{x}^T \mathbf{x}' + 1)^2 \\ &= (x_1 x_1' + x_2 x_2' + 1)^2 \\ &= 1 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_2 x_1' x_2' + (x_1)^2 (x_1')^2 + (x_2)^2 (x_2')^2 \end{aligned}$$

which corresponds to the inner product of the basis function

$$\phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, (x_1)^2, (x_2)^2)^T$$

Some Common Kernel Functions (2)

- Radial basis function (RBF) kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2s^2})$$

  It can be generalized to

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\frac{D(\mathbf{x}, \mathbf{x}')}{2s^2})$$

  where $D(\cdot, \cdot)$ is some distance function.

- Sigmoidal kernel:

$$K(\mathbf{x}, \mathbf{x}') = \tanh(2\mathbf{x}^T\mathbf{x}' + 1)$$

Some Common Kernel Functions (2)

- Radial basis function (RBF) kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2s^2})$$

  It can be generalized to

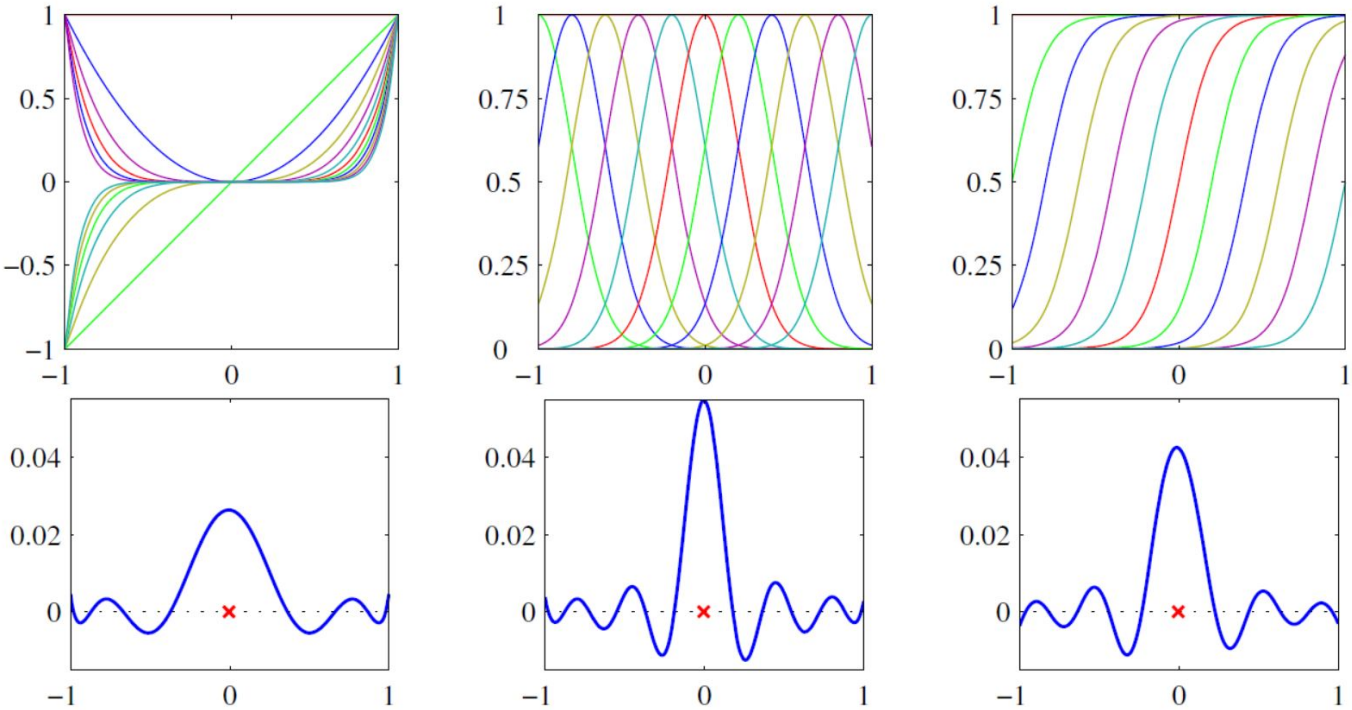
$$K(\mathbf{x}, \mathbf{x}') = \exp(-\frac{D(\mathbf{x}, \mathbf{x}')}{2s^2})$$

  where $D(\cdot, \cdot)$ is some distance function.

- Sigmoidal kernel:

$$K(\mathbf{x}, \mathbf{x}') = \tanh(2\mathbf{x}^T\mathbf{x}' + 1)$$

Illustration of Kernels



- The lower plot shows the kernel function $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$ plotted as a function of $\mathbf{x}$, where $\mathbf{x}'$ is given by the red cross.
- The upper plot shows the corresponding basis functions given by polynomials (left), Gaussians (centre), and logistic sigmoids (right).

Theoretical Analysis

Representer Theorem

Let $\mathcal{X}$ be a nonempty set and K a positive-definite real-valued kernel on $\mathcal{X} \times \mathcal{X}$ with corresponding reproducing kernel Hilbert space $\mathcal{H}_K$. Given a training sample $\{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$, a strictly monotonically increasing real-valued function $g : [0, \infty) \to \mathcal{R}$, and an arbitrary empirical risk function $\hat{E}$, then for any $f^* \in \mathcal{H}_K$ satisfying:

$$f^* = \arg \min_{f \in \mathcal{H}_K} \{\hat{E} + g(\|f\|_K)\},$$

f^* admits a representation of the form:

$$f^*(\cdot) = \sum_{i=1}^{N} \alpha_i K(\cdot, \mathbf{x}^{(i)}),$$

where $\alpha_i \in \mathcal{R}$ for all $1 \leq i \leq N$.

Kernelized SVM

- It can be viewed as we are mapping the original input space $\mathcal{X}$ to a feature space $\mathcal{H}$ by the mapping function ϕ, and learning a linear SVM in the new feature space.
- When the mapping function is nonlinear, we can obtain a nonlinear classifier.
- Given the kernel $K(\mathbf{x}, \mathbf{x}')$, we can still use linear SVM methods to find the solution of the new nonlinear problem.
- Please note that the learning process is directly conducted in the feature space, we do not need to explicitly define the feature space $\mathcal{H}$ and the mapping function ϕ. This is one advantage of kernel, which can directly use the linear classification methods to solve the nonlinear problem.

Dual Optimization Problem of Linear SVM

- Dual optimization problem:

$$\max_{\boldsymbol{\alpha}} \quad \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)}$$

$$s.t. \quad \sum_i \alpha_i y^{(i)} = 0 \text{ and } 0 \leq \alpha_i \leq C, \forall i$$

Dual Optimization Problem of Nonlinear SVM

- Dual optimization problem:

$$\max_{\boldsymbol{\alpha}} \quad \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(j)})$$

$$s.t. \quad \sum_i \alpha_i y^{(i)} = 0 \text{ and } 0 \leq \alpha_i \leq C, \forall i$$

or

$$\max_{\boldsymbol{\alpha}} \quad \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

$$s.t. \quad \sum_i \alpha_i y^{(i)} = 0 \text{ and } 0 \leq \alpha_i \leq C, \forall i$$

where $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \equiv \phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(j)})$ is a kernel function defined directly on the inputs $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$.

Nonlinear SVM Classifier

- Kernelized solution:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y^{(i)} \phi(\mathbf{x}^{(i)}) = \sum_{\phi(\mathbf{x}^{(i)}) \in \mathcal{SV}} \alpha_i y^{(i)} \phi(\mathbf{x}^{(i)})$$

$$w_0 = y^{(j)} - \mathbf{w}^T \phi(\mathbf{x}^{(j)}) = y^{(j)} - \sum_{\phi(\mathbf{x}^{(i)}) \in \mathcal{SV}} \alpha_i y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

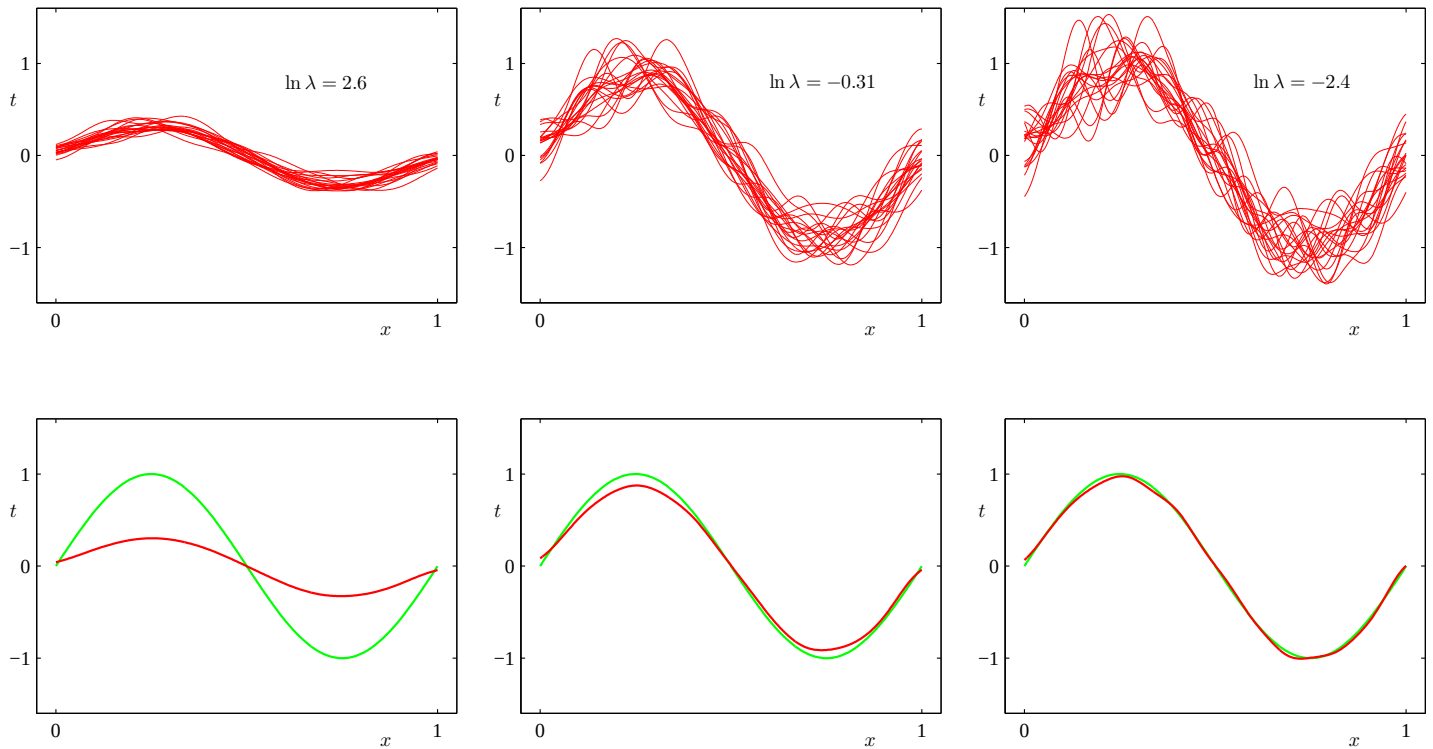
- Discriminant function:

$$\begin{aligned} g(\mathbf{x}) &= \mathbf{w}^T \phi(\mathbf{x}) + w_0 \\ &= \sum_{\phi(\mathbf{x}^{(i)}) \in \mathcal{SV}} \alpha_i y^{(i)} \phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}) + w_0 \\ &= \sum_{\phi(\mathbf{x}^{(i)}) \in \mathcal{SV}} \alpha_i y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x}) + w_0 \end{aligned}$$

Regularized Error Function in Linear Regression

- The regularized error function:

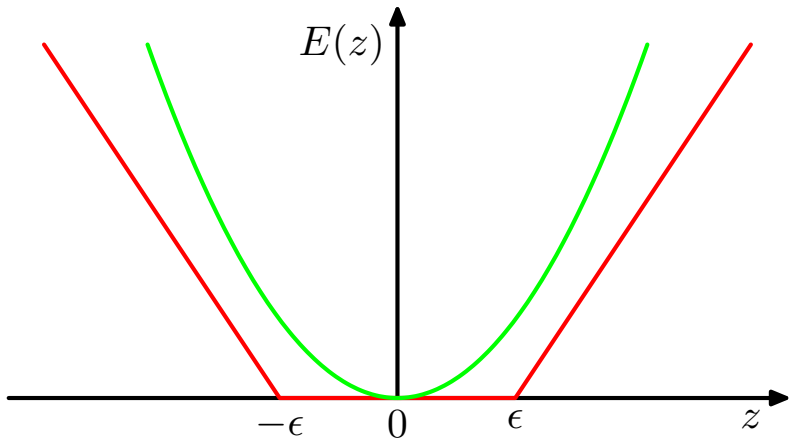
$$\frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - f(\mathbf{x}^{(i)}))^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$



ϵ-Insensitive Loss Function

- To obtain sparse solutions, the quadratic error function is replaced by an ϵ-insensitive loss function, e.g.,

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

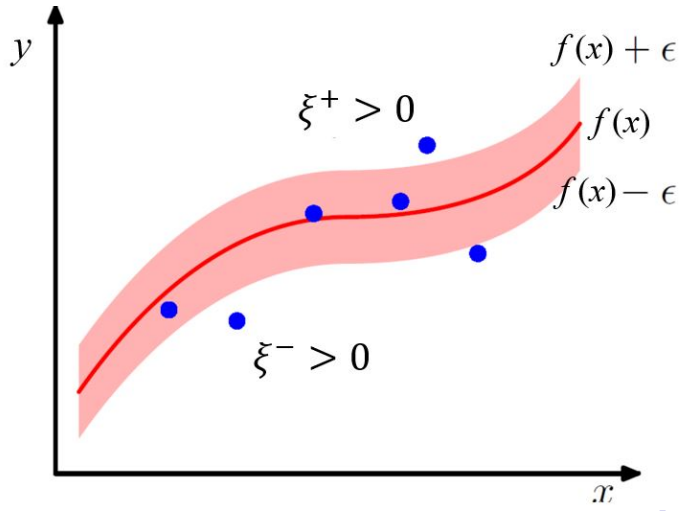


  - Errors are tolerated up to a threshold of ϵ.
  - Errors beyond ϵ have linear (rather than quadratic) effect so that the model is more robust against noise.

Introducing Slack Variables

- Introducing two types of slack variables $\xi_i^+ \geq 0$ and $\xi_i^- \geq 0$
  - $\xi_i^+ > 0$ (positive deviation) corresponds to a point for which $y^{(i)} - f(\mathbf{x}^{(i)}) > \epsilon$
  - $\xi_i^- > 0$ (negative deviation) corresponds to a point for which $y^{(i)} - f(\mathbf{x}^{(i)}) < -\epsilon$
- If the slack variables are nonzero, data points may lie outside the tube. The corresponding conditions are:

$$y^{(i)} \leq f(\mathbf{x}^{(i)}) + \epsilon + \xi_i^+, \quad y^{(i)} \geq f(\mathbf{x}^{(i)}) - \epsilon - \xi_i^-$$



Primal Optimization Problem

- Primal optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, w_0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i^+ + \xi_i^-) \\ s.t. \quad & y^{(i)} - (\mathbf{w}^T\phi(\mathbf{x}^{(i)}) + w_0) \le \epsilon + \xi_i^+, \quad \forall i \\ & (\mathbf{w}^T\phi(\mathbf{x}^{(i)}) + w_0) - y^{(i)} \le \epsilon + \xi_i^-, \quad \forall i \\ & \xi_i^+, \xi_i^- \ge 0, \quad \forall i \end{aligned}$$

- $f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + w_0$
- If $y^{(i)} - (\mathbf{w}^T\phi(\mathbf{x}^{(i)}) + w_0) \le \epsilon$ and $(\mathbf{w}^T\phi(\mathbf{x}^{(i)}) + w_0) - y^{(i)} \le \epsilon$, then $\xi_i^+ = \xi_i^- = 0$, contributing no cost to the objective function.

Lagrangian

- Similar to SVM for classification, the optimization problem for SVR can also be rewritten in the dual form.
- Lagrangian:

$$
\begin{aligned}
L_p \quad = \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i^+ + \xi_i^-) \\
& - \sum_{i=1}^{N}\alpha_i^+[\epsilon + \xi_i^+ - y^{(i)} + (\mathbf{w}^T\phi(\mathbf{x}^{(i)}) + w_0)] \\
& - \sum_{i=1}^{N}\alpha_i^-[\epsilon + \xi_i^- + y^{(i)} - (\mathbf{w}^T\phi(\mathbf{x}^{(i)}) + w_0)] - \sum_{i=1}^{N}(\mu_i^+\xi_i^+ + \mu_i^-\xi_i^-)
\end{aligned}
$$

  - α_i^+,α_i^- , μ_i^+ and μ_i^- are Lagrange multipliers.

Dual Optimization Problem

- Set the derivatives of the Lagrangian w.r.t. $\mathbf{w}, w_0, \xi_i^+, \xi_i^-$ to 0, giving

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_i (\alpha_i^+ - \alpha_i^-)\phi(\mathbf{x}^{(i)}) \tag{6}$$

$$\frac{\partial L_p}{\partial w_0} = 0 \quad \Rightarrow \quad \sum_i (\alpha_i^+ - \alpha_i^-) = 0 \tag{7}$$

$$\frac{\partial L_p}{\partial \xi_i^+} = 0 \quad \Rightarrow \quad \alpha_i^+ + \mu_i^+ = C \tag{8}$$

$$\frac{\partial L_p}{\partial \xi_i^-} = 0 \quad \Rightarrow \quad \alpha_i^- + \mu_i^- = C \tag{9}$$

Dual Optimization Problem

- Eliminating the corresponding variables from the Lagrangian and get the dual optimization problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-} \quad & -\frac{1}{2}\sum_i\sum_j(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)K(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) \\ & -\epsilon\sum_i(\alpha_i^+ + \alpha_i^-) + \sum_i(\alpha_i^+ - \alpha_i^-)y^{(i)} \\ s.t. \quad & \sum_i(\alpha_i^+ - \alpha_i^-) = 0, 0 \le \alpha_i^+ \le C, \text{ and } 0 \le \alpha_i^- \le C, \forall i \end{aligned}$$

where $K(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)})^T\phi(\mathbf{x}^{(j)})$ is a kernel function.

The KKT Conditions and Support Vectors

- The KKT conditions:

$$\alpha_i^+(\epsilon + \xi_i^+ - y^{(i)} + (\mathbf{w}^T\phi(\mathbf{x}^{(i)}) + w_0)) = 0 \quad (10)$$

$$\alpha_i^-(\epsilon + \xi_i^- + y^{(i)} - (\mathbf{w}^T\phi(\mathbf{x}^{(i)}) + w_0)) = 0 \quad (11)$$

$$(C - \alpha_i^+)\xi_i^+ = 0 \quad (12)$$

$$(C - \alpha_i^-)\xi_i^- = 0 \quad (13)$$

  - $\alpha_i^+ > 0$ means the data point either lies on the upper boundary of the ϵ-tube ($\xi_i^+ = 0$) or above the upper boundary ($\xi_i^+ > 0$). Similarly for $\alpha_i^- > 0$.
  - For every data point $\mathbf{x}^{(i)}$, either α_i^+ or α_i^- (or both) must be zero.
  - Support vectors are points that lie on the boundary of the ϵ-tube or outside the tube.
  - All points within the tube have $\alpha_i^+ = \alpha_i^- = 0$.

The Solution of Support Vector Regression

- From (6), the prediction for any new input $\mathbf{x}$ can be made using:

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + w_0 = \sum_{i=1}^{N} (\alpha_i^+ - \alpha_i^-) K(\mathbf{x}^{(i)}, \mathbf{x}) + w_0$$

- w_0 can be found by considering a data point for which $0 < \alpha_i^+ < C$. From KKT conditions, $\xi_i^+ = 0$ and $\epsilon - y^{(i)} + (\mathbf{w}^T \phi(\mathbf{x}^{(i)}) + w_0) = 0$.

$$\begin{aligned} w_0 &= y^{(i)} - \epsilon - \mathbf{w}^T \phi(\mathbf{x}^{(i)}) \\ &= y^{(i)} - \epsilon - \sum_{j=1}^{N} (\alpha_j^+ - \alpha_j^-) K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \end{aligned}$$

- We can get analogous result by considering a point for which $0 < \alpha_i^- < C$.
- It is better to average over all such estimates of w_0.

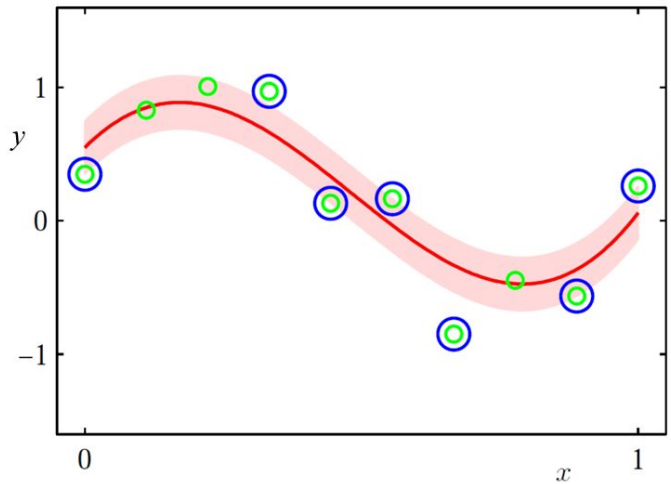
ν-SVR

- Instead of fixing the width ϵ of the insensitive region, we fix instead a parameter ν that bounds the fraction of points lying outside the tube.

$$\max_{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-} \quad -\frac{1}{2}\sum_i\sum_j(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)K(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) + \sum_i(\alpha_i^+ - \alpha_i^-)y^{(i)}$$

$$s.t. \quad \sum_i(\alpha_i^+ - \alpha_i^-) = 0, \sum_i(\alpha_i^+ + \alpha_i^-) \le \nu C$$

$$0 \le \alpha_i^+ \le C/N, 0 \le \alpha_i^- \le C/N, \forall i$$



Lagrangian

- Standard form problem (not necessarily convex)

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_0(\mathbf{x}) \\ s.t. \quad & f_i(\mathbf{x}) \leq 0, i = 1, \ldots, m \\ & h_i(\mathbf{x}) = 0, i = 1, \ldots, p \end{aligned}$$

  variable $\mathbf{x} \in \mathbb{R}^n$, domain $\mathcal{D}$, optimal value p^*.

- Lagrangian: $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$, with $dom(L) = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x})$$

  - weighted sum of objective and constraint functions
  - λ_i is Lagrange multiplier associated with $f_i(\mathbf{x}) \leq 0$
  - ν_i is Lagrange multiplier associated with $h_i(\mathbf{x}) = 0$

Lagrangian Dual Function

- Lagrange dual function

$$\begin{aligned} g(\boldsymbol{\lambda}, \boldsymbol{\nu}) &= \inf_{\mathbf{x} \in \mathcal{D}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \\ &= \inf_{\mathbf{x} \in \mathcal{D}} \left(f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x}) \right) \end{aligned}$$

- g is concave, can be $-\infty$ for some $\boldsymbol{\lambda}, \boldsymbol{\nu}$.
- Lower bound property: if $\boldsymbol{\lambda} \succeq 0$, then $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq p^*$.

The Dual Problem

- Lagrange dual problem:

$$\begin{aligned} \max_{\boldsymbol{\lambda},\boldsymbol{\nu}} \quad & g(\boldsymbol{\lambda},\boldsymbol{\nu}) \\ s.t. \quad & \boldsymbol{\lambda} \succeq 0 \end{aligned}$$

  - finds best lower bound on p^*, obtained from Lagrange dual function
  - a concave optimization problem with optimal value denoted as d^*
  - $\boldsymbol{\lambda}, \boldsymbol{\nu}$ are dual feasible if $\boldsymbol{\lambda} \succeq 0, (\boldsymbol{\lambda}, \boldsymbol{\nu}) \in dom(g)$
  - often simplified by making implicit constraint $(\boldsymbol{\lambda}, \boldsymbol{\nu}) \in dom(g)$ explicit

The Dual Problem

- Example: standard form LP and its dual
  - Primal:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \succeq 0 \end{aligned}$$

  - Lagrange:

$$L(\mathbf{x}, \lambda, \nu) = \mathbf{c}^T\mathbf{x} + \nu^T(\mathbf{A}\mathbf{x} - \mathbf{b}) - \lambda^T\mathbf{x} = -\mathbf{b}^T\nu + (\nu^T\mathbf{A} + \mathbf{c}^T - \lambda)\mathbf{x}$$

  - Lagrange dual function:

$$g(\lambda, \nu) = \inf_{\mathbf{x}\in\mathcal{D}} = L(\mathbf{x}, \lambda, \nu) \begin{cases} -\mathbf{b}^T\nu, & \nu^T\mathbf{A} + \mathbf{c}^T - \lambda = 0 \\ -\infty, & \text{otherwise} \end{cases}$$

  - Dual:

$$\begin{aligned} \max_{\nu} \quad & -\mathbf{b}^T\nu \\ s.t. \quad & \mathbf{A}^T\nu + \mathbf{c} \succeq 0 \end{aligned}$$

Weak and Strong Duality

- Weak duality: $d^* \leq p^*$
  - always holds (for convex and nonconvex problem)
  - can be used to find nontrivial lower bounds for difficult problems
- Strong duality: $d^* = p^*$
  - does not hold in general
  - (usually) holds for convex problems

Complementary Slackness

- Assume strong duality holds, $\mathbf{x}^*$ is primal optimal, $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$ is dual optimal, then

$$\begin{aligned} f_0(\mathbf{x}^*) = g(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*) &= \inf_{\mathbf{x}} \left(f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i^* f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i^* h_i(\mathbf{x}) \right) \\ &\leq f_0(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i^* f_i(\mathbf{x}^*) + \sum_{i=1}^{p} \nu_i^* h_i(\mathbf{x}^*) \\ &\leq f_0(\mathbf{x}^*) \end{aligned}$$

  hence, the two inequalities hold with equality
  - $\mathbf{x}^*$ minimizes $L(\mathbf{x}, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$
  - $\lambda_i^* f_i(\mathbf{x}^*) = 0$ for $i = 1, \ldots, m$ (complementary slackness)

$$\begin{aligned} \lambda_i^* > 0 &\implies f_i(\mathbf{x}^*) = 0 \\ f_i(\mathbf{x}^*) < 0 &\implies \lambda_i^* = 0 \end{aligned}$$

Karush-Kuhn-Tucker (KKT) Conditions

- The following four conditions are called KKT conditions (for a problem with differentiable f_i, h_i):
  1. primal constraints: $f_i(\mathbf{x}) \leq 0, i = 1, \ldots, m, h_i(\mathbf{x}) = 0, i = 1, \ldots, p$
  2. dual constraints: $\boldsymbol{\lambda} \succeq 0$
  3. complementary slackness: $\lambda_i f_i(\mathbf{x}) = 0, i = 1, \ldots, m$
  4. gradient of Lagrangian w.r.t. $\mathbf{x}$ vanishes:

$$\nabla f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \nabla f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i \nabla h_i(\mathbf{x}) = 0$$

If strong duality holds, $\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}$ are optimal $\Leftrightarrow$ they must satisfy the KKT conditions.

Sequential Minimal Optimization (SMO)

- We have mentioned that the primal SVM can be solved by traditional convex quadratic programming methods, which can guarantee the global optimal solution. However, these algorithm usually become slowly especially when the training data are large.
- SMO, proposed by John Platt in 1998, gives an efficient way of solving the dual problem arising from the derivation of the SVM.
- The dual optimization problem:

$$\max_{\boldsymbol{\alpha}} \quad \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)}$$

$$s.t. \quad \sum_i \alpha_i y^{(i)} = 0 \text{ and } 0 \leq \alpha_i \leq C, \forall i$$

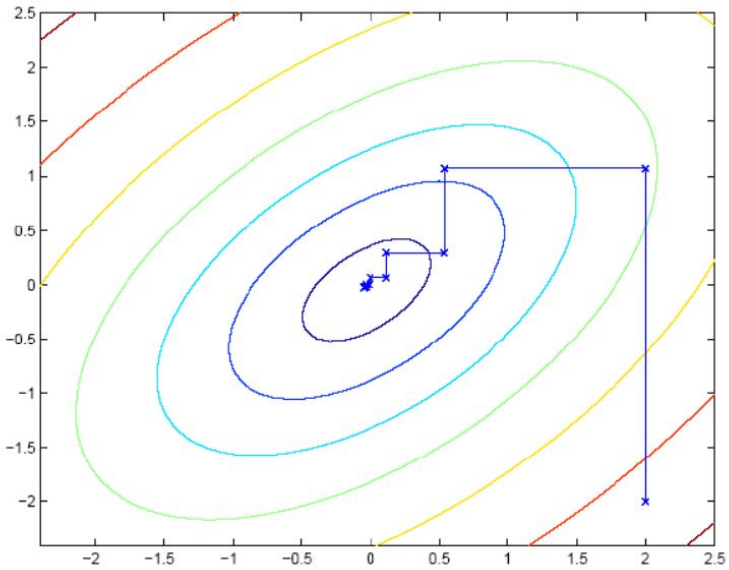
Coordinate Ascent

- Consider the following unconstrained optimization problem:

$$\max_{\boldsymbol{\alpha}} J(\alpha_1, \ldots, \alpha_N)$$

- Coordinate ascent optimization algorithm:
  Repeat until convergence{
  For $i = 1$ to N {
  $\alpha_i := \arg\max_{\hat{\alpha}_i} J(\alpha_1, \ldots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \ldots, \alpha_N)$
  }
  }
- In the innermost loop of this algorithm, we will hold all the variables except for some α_i fixed, and re-optimize J w.r.t. just the parameter α_i.

Illustration of Coordinate Ascent



- The ellipses in the figure are the contours of a quadratic function that we want to optimize.
- Coordinate ascent was initialized at $(-2, 2)$, and also plotted in the figure is the path that it took on its way to the global maximum.
- Notice that on each step, coordinate ascent takes a step thats parallel to one of the axes, since only one variable is being optimized at a time.

Coordinate Ascent for SVM?

- The dual optimization problem:

$$\max_{\boldsymbol{\alpha}} \quad \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)}$$

$$s.t. \quad \sum_i \alpha_i y^{(i)} = 0 \text{ and } 0 \leq \alpha_i \leq C, \forall i$$

- Suppose we want to hold $\alpha_2, \ldots, \alpha_N$ fixed, and take a coordinate ascent step and re-optimize the objective with respect to α_1. Can we make any progress?
- The answer is NO, because the constraint ensures that:

$$\alpha_1 = -y^{(1)} \sum_{i=2}^{N} \alpha_i y^{(i)}$$

SMO: Update w.r.t. Two Variables

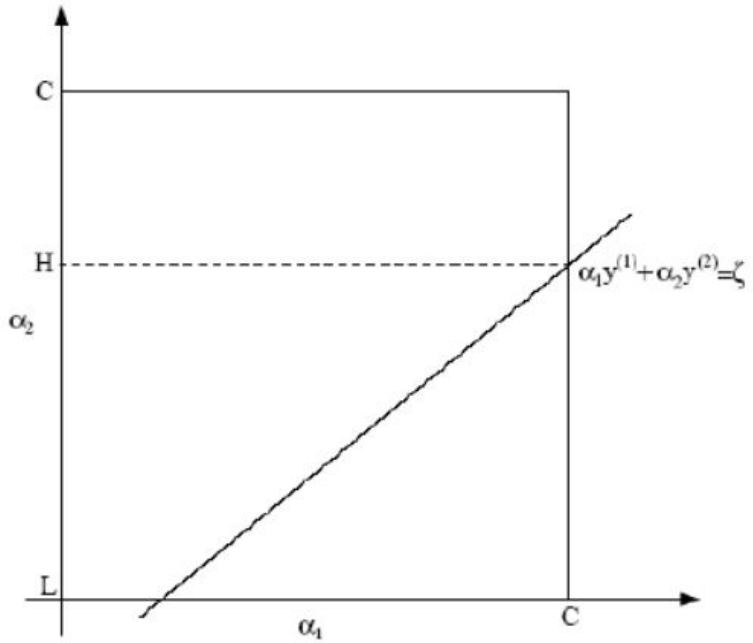
- We must update at least two of them simultaneously in order to keep satisfying the constraint.
- This is exactly the motivation of SMO, which simply does the following: Repeat until convergence{

  1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum);
  2. Re-optimize $J(\boldsymbol{\alpha})$ w.r.t. α_i and α_j, while holding all the other α_ks ($k \neq i, j$) fixed:

  }
- The key reason that SMO is an efficient algorithm is that the update α_i, α_j can be computed very efficiently.

SMO: Update w.r.t. Two Variables

- Suppose we've decided to hold $\alpha_3, \ldots, \alpha_N$ fixed, and want to re-optimize $J(\alpha_1, \ldots, \alpha_N)$ with respect to α_1 and α_2.
- The constraints and illustration:$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = -\sum_{i=3}^{N} \alpha_i y^{(i)} = \zeta$, $0 \leq \alpha_1 \leq C$, $0 \leq \alpha_2 \leq C$.



SMO: Update w.r.t. One Variable

- Write α_1 as a function of α_2:

$$\alpha_1 = (\zeta - \alpha_2 y^{(2)}) y^{(1)}.$$

- The objective $J(\boldsymbol{\alpha})$ can be written as:

$$J(\boldsymbol{\alpha}) = J((\zeta - \alpha_2 y^{(2)}) y^{(1)}, \alpha_2, \ldots, \alpha_N).$$

- If we ignore the constraint of $L \leq \alpha_2 \leq H$, then we can easily maximize this quadratic function by setting its derivative to zero and get the resulting value of α_2 as $\alpha_2^{new,unclipped}$.

SMO: Optimal Solution of α_1 and α_2

- Maximize J w.r.t. α_2 but subject to the box constraint, then we can find the resulting optimal value simply by clipping $\alpha_2^{new,unclipped}$ to lie in the $[L, H]$ interval:

$$\alpha_2^{new} = \begin{cases} H & \text{if } \alpha_2^{new,unclipped} > H \\ \alpha_2^{new,unclipped} & \text{if } L \leq \alpha_2^{new,unclipped} \leq H \\ L & \text{if } \alpha_2^{new,unclipped} < L \end{cases}$$

- We can use $\alpha_1 = (\zeta - \alpha_2 y^{(2)}) y^{(1)}$ and $\zeta = \alpha_1 y^{(1)} + \alpha_2 y^{(2)}$ to find the optimal value of α_1^{new} as:

$$\alpha_1^{new} = \alpha_1^{old} + y^{(1)} y^{(2)} (\alpha_2^{old} - \alpha_2^{new}).$$

SMO: Variable Selection for the First One

- Main idea: Select the training example which most violate KKT condition, and treat the corresponding α_i as the first variable.
- Check whether the i-th training example $(\mathbf{x}^{(i)}, y^{(i)})$ satisfies the KKT condition:

$$\begin{aligned} \alpha_i = 0 &\Leftrightarrow y^{(i)} g(\mathbf{x}^{(i)}) \geq 1, \\ 0 < \alpha_i < C &\Leftrightarrow y^{(i)} g(\mathbf{x}^{(i)}) = 1 \\ \alpha_i = C &\Leftrightarrow y^{(i)} g(\mathbf{x}^{(i)}) \leq 1 \end{aligned}$$

  where $g(\mathbf{x}^{(i)}) = (\sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{(i)})^T \mathbf{x} + w_0$ is the discriminant function.
- We first check all of the examples satisfying $0 < \alpha_i < C$, i.e. support vectors. If all these examples satisfy the KKT condition, then we will check all the training examples.

SMO: Variable Selection for the Second One

- Once we have found the first variable α_1, now we are finding α_2. The criteria is to make α_2 change much.

Theorem

$$\alpha_2^{new,unclipped} = \alpha_2^{old} + \frac{y^{(2)}(E_1 - E_2)}{\eta},$$

where $\eta = K_{11} + K_{22} - 2K_{12} = \|\Phi(\mathbf{x}^{(1)}) - \Phi(\mathbf{x}^{(2)})\|^2$, $E_i = g(\mathbf{x}^{(i)}) - y^{(i)}$.

- A simple method is to directly choose α_2 to make the corresponding $|E_1 - E_2|$ largest.

Supervised Learning with Complex Output

- Assume: data is I.I.D. from

$$p(\mathbf{x}, y)$$

- Given: training samples

$$\mathcal{X} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$$

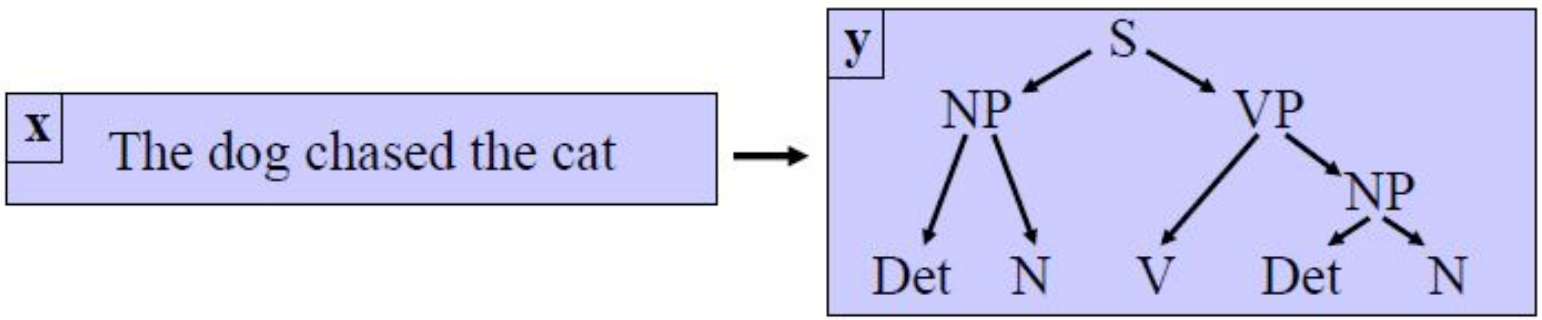
- Goal: find function from input $\mathcal{X}$ to output $\mathcal{Y}$ (set of complex objects) $f : \mathcal{X} \to \mathcal{Y}$ with low risk/prediction error

$$R(f) = \int L(f(\mathbf{x}), y) dp(\mathbf{x}, y)$$

- Methods: Kernel methods, SVM, CRF, etc.

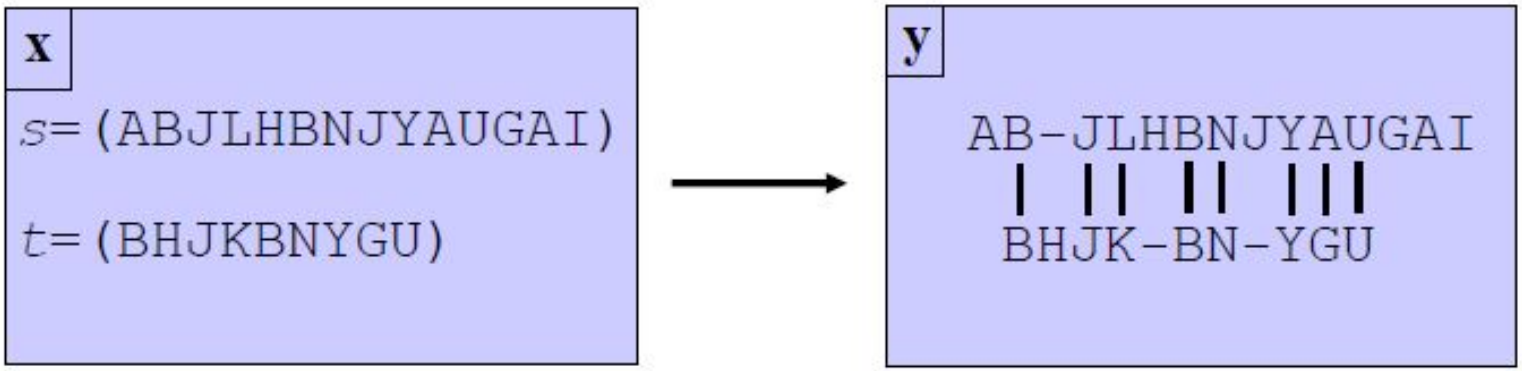
Examples of Complex Outputs

- Natural language parsing
  - Given a sequence of words **x**, predict the parse tree **y**.
  - Dependencies from structural constraints, since **y** has to be a tree.



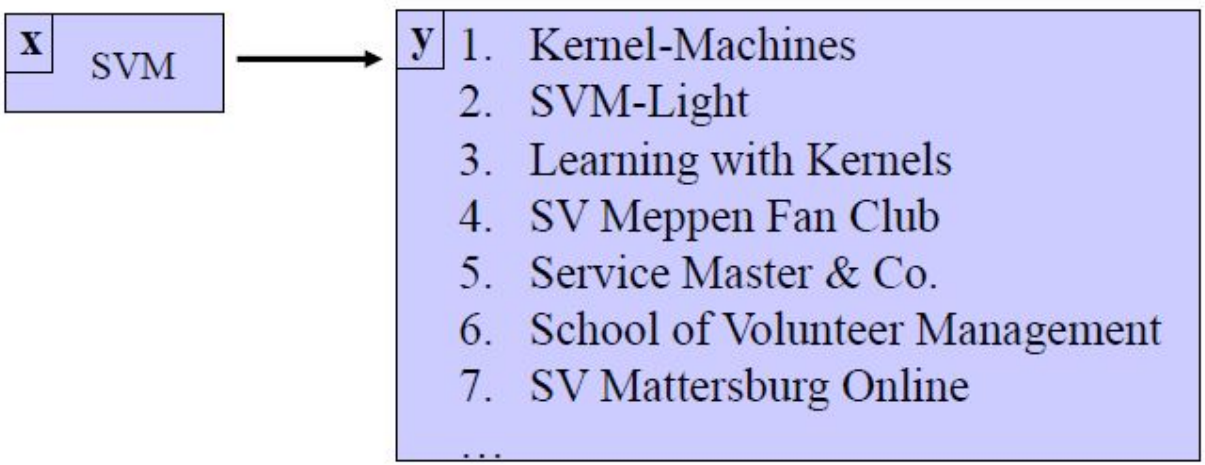
Examples of Complex Outputs (2)

- Protein sequence alignment
  - Given two sequences $\mathbf{x} = (\mathbf{s}, \mathbf{t})$, predict an alignment $\mathbf{y}$.
  - Structural dependencies, since prediction has to be a valid global/local alignment.



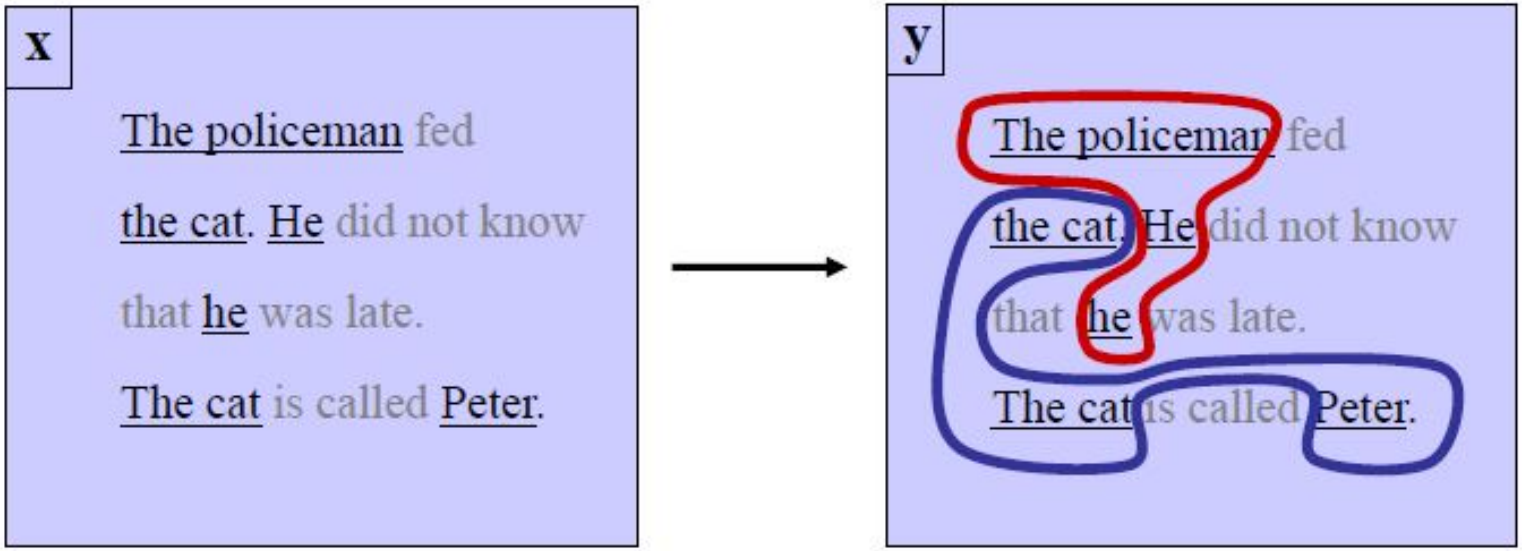
Examples of Complex Outputs (3)

- Information Retrieval
  - Given a query **x**, predict a ranking **y**
  - Dependencies between results
  - Loss function over rankings (e.g., accumulate accuracy, average precision)



Examples of Complex Outputs (4)

- Noun-phrase co-reference
  - Given a set of noun phrases **x**, predict a clustering **y**.
  - Structural dependencies, since prediction has to be an equivalence relation.
  - Correlation dependencies from interactions.



Examples of Complex Outputs (5)

- and many many more:
  - Sequence labeling (e.g., part-of-speech tagging, named entity recognition)
  - Collective classification (e.g., hyperlinked documents)
  - Multi-label classification (e.g., text classification)
  - Binary classification with non-linear performance measures (e.g., optimization F1-score, average precision)
  - Inverse reinforcement learning/planning (i.e., learn reward function to predict action sequences)

Multi-class SVM

- Approach: view as multi-class classification task, with every complex output $y_i \in \mathcal{Y}$ is one class.
- Training examples: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, $\mathbf{x}_i \in \mathbb{R}^D$, $y_i \in \{1, \ldots, K\}$
- Formulation:

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_K,\boldsymbol{\xi}} \quad & \sum_{k=1}^{K} \|\mathbf{w}_k\|^2 + C \sum_{i=1}^{N} \xi_i \\ s.t. \quad & \forall j \neq y_1 : \mathbf{w}_{y_1}^T \mathbf{x}_1 \geq \mathbf{w}_j^T \mathbf{x}_1 + 1 - \xi_1 \\ & \ldots \\ & \forall j \neq y_N : \mathbf{w}_{y_N}^T \mathbf{x}_N \geq \mathbf{w}_j^T \mathbf{x}_N + 1 - \xi_N \end{aligned}$$

Joint Feature Map

- Joint feature map $\phi(\mathbf{x}, \mathbf{y})$ is a vector-valued function that describes the relationship between input $\mathbf{x}$ and output structure $\mathbf{y}$.
- Learn single weight vector and rank by $\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$

$$f(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} [\mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})]$$

- Manageable number of parameters!

Structural SVM

- Structural SVM applies margins between the true structure and all other possible structures.
- Hard-margin optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ s.t. \quad & \forall \mathbf{y} \in \mathcal{Y} \backslash \mathbf{y}_1 : \mathbf{w}^T\phi(\mathbf{x}_1, \mathbf{y}_1) \geq \mathbf{w}^T\phi(\mathbf{x}_1, \mathbf{y}) + 1 \\ & \dots \\ & \forall \mathbf{y} \in \mathcal{Y} \backslash \mathbf{y}_N : \mathbf{w}^T\phi(\mathbf{x}_N, \mathbf{y}_N) \geq \mathbf{w}^T\phi(\mathbf{x}_N, \mathbf{y}) + 1 \end{aligned}$$

Structural SVM

- Soft-margin optimization problem:

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & \forall \mathbf{y} \in \mathcal{Y}\backslash\mathbf{y}_1 : \mathbf{w}^T\phi(\mathbf{x}_1,\mathbf{y}_1) \geq \mathbf{w}^T\phi(\mathbf{x}_1,\mathbf{y}) + \triangle(\mathbf{y}_1,\mathbf{y}) - \xi_1 \\ & \ldots \\ & \forall \mathbf{y} \in \mathcal{Y}\backslash\mathbf{y}_N : \mathbf{w}^T\phi(\mathbf{x}_N,\mathbf{y}_N) \geq \mathbf{w}^T\phi(\mathbf{x}_N,\mathbf{y}) + \triangle(\mathbf{y}_N,\mathbf{y}) - \xi_N \end{aligned}$$

  - For more flexible notions of structural correctness, the margin is set of $\triangle(\mathbf{y}_i,\mathbf{y})$, a non-negative loss function defined between structures.

Reformulation of Structural SVM

- *n*-slack formulation:

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & \forall \mathbf{y}' \in \mathcal{Y} : \mathbf{w}^T\phi(\mathbf{x}_1,\mathbf{y}_1) - \mathbf{w}^T\phi(\mathbf{x}_1,\mathbf{y}') \geq \triangle(\mathbf{y}_1,\mathbf{y}') - \xi_1 \\ & \ldots \\ & \forall \mathbf{y}' \in \mathcal{Y} : \mathbf{w}^T\phi(\mathbf{x}_N,\mathbf{y}_N) - \mathbf{w}^T\phi(\mathbf{x}_N,\mathbf{y}') \geq \triangle(\mathbf{y}_N,\mathbf{y}') - \xi_N \end{aligned}$$

- 1-slack formulation:

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi \\ s.t. \quad & \forall \mathbf{y}'_1,\ldots,\mathbf{y}'_N \in \mathcal{Y} : \frac{1}{N}[\mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{y}_i) - \mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{y}'_i)] \geq \frac{1}{N}\sum_{i=1}^{N}\triangle(\mathbf{y}_i,\mathbf{y}'_i) - \xi \end{aligned}$$

Cutting Planes Algorithm

- Since the set $\mathcal{Y}$ of possible output structures is generally quite large, enforcing all margin constraints is not feasible in practice.
- Cutting planes can be applied to efficiently find a small working set of active constraints which are sufficient to optimize $\mathbf{w}$ within some prescribed tolerance.
- Separation oracle: given a fixed $\mathbf{w}$ and input point $\mathbf{x}_i$, outputs the structure $\mathbf{y}'_i$ corresponding to the margin constraint for $\mathbf{x}_i$ which is most violated by $\mathbf{w}$:

$$\mathbf{y}'_i = \arg\max_{\mathbf{y}\in\mathcal{Y}} \{\triangle(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y})\}$$

- Focusing on the constraints that are violated the most by the current model.

Cutting-Plane Algorithm (1-slack Formulation)

- **Input:** $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N), C, \epsilon$
- $\mathcal{S} \leftarrow \varnothing, \mathbf{w} \leftarrow \mathbf{0}, \xi \leftarrow 0$
- **Repeat**
  - For $i = 1, \ldots, N$, compute

$$\mathbf{y}_i' = \arg\max_{\mathbf{y} \in \mathcal{Y}} \{\triangle(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y})\}$$

  - End For
  - If $\sum_{i=1}^{N} \{\triangle(\mathbf{y}_i, \mathbf{y}_i') - \mathbf{w}^T [\phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y}_i')]\} > \xi + \epsilon$

$$\begin{aligned} \mathcal{S} &\leftarrow \mathcal{S} \bigcup \{\mathbf{w}^T \frac{1}{N} \sum_{i=1}^{N} [\phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y}_i')] \geq \frac{1}{N} \sum_{i=1}^{N} \triangle(\mathbf{y}_i, \mathbf{y}_i') - \xi\} \\ \{\mathbf{w}, \xi\} &\leftarrow \text{optimize SSVM over } \mathcal{S} \end{aligned}$$

  - End If
- **Until** $\mathcal{S}$ has not changed during iteration

Polynomial Sparsity Bound

- Theorem: The cutting-plane algorithm finds a solution to the Structural SVM soft-margin optimization problem in the 1-slack formulation after adding at most

$$\lceil \log_2(\frac{\triangle}{4R^2C}) \rceil + \lceil \frac{16R^2C}{\epsilon} \rceil$$

constraints to the working set $\mathcal{S}$, so that the primal constraints are feasible up to a precision ϵ and the objective on $\mathcal{S}$ is optimal. The loss has to be bounded $0 \leq \triangle(\mathbf{y}_i, \mathbf{y}) \leq \triangle$ and $2\|\phi(\mathbf{x}, \mathbf{y})\| \leq R$.

- The upper bound on the number of active constraints depends on the chosen representation.

Summary - SSVM Learning

- Given
  - training set $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\} \in \mathcal{X} \times \mathcal{Y}$
  - loss function $\triangle : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
- Task: learn parameter $\mathbf{w}$ for $f(\mathbf{x}) = \arg\max_{\mathbf{y}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$ that minimizes expected loss on future data.
- SSVM solution derived by maximum margin framework:
  - enforce correct output to be better than others by a margin:

$$\mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) \geq \triangle(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}), \forall \mathbf{y} \in \mathcal{Y}$$

  - convex optimization problem
  - many equivalent formulations $\to$ different training algorithms
  - training needs repeated argmax prediction

Applying SSVM to New Problem

- General
  - SVM-struct algorithm and implementation (general API)
    `http://svmlight.joachims.org`
  - Theory (E.g., training time linear in N)
- Application specific:
  - Modeling: how can we define suitable feature maps $\phi(\mathbf{x}, \mathbf{y})$ for specific problems?
  - Algorithms: how can we compute the required maximization over $\mathcal{Y}$ for given $\mathbf{x}$?
  - Sparseness: how can we bound $\|\phi(\mathbf{x}, \mathbf{y}) - \phi(\mathbf{x}, \mathbf{y}')\|$?

Structural SVM for Ranking

- Training set: $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$
  - $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is the corpus
  - $\mathbf{y}_i \in \mathcal{Y}$, $\mathcal{Y}$ is the set of permutations (ranking) of $\mathcal{X}$
  - For a ranking $\mathbf{y} \in \mathcal{Y}$ and two points $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, $\mathbf{x}_i \prec_{\mathbf{y}} \mathbf{x}_j (\mathbf{x}_i \succ_{\mathbf{y}} \mathbf{x}_j)$ means $\mathbf{x}_i$ is placed before (after) $\mathbf{x}_j$ in $\mathbf{y}$.
- Formulation using structural SVM:

$$\forall \mathbf{y} \in \mathcal{Y} : \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}) + \triangle(\mathbf{y}_i, \mathbf{y}) - \xi$$

- For a test query $\mathbf{q}$, the structure $\mathbf{y}$ is found by maximizing $\mathbf{w}^T \phi(\mathbf{q}, \mathbf{y})$.

Structural SVM for Ranking (2)

- Partial order feature

$$\phi_{po}(\mathbf{q}, \mathbf{y}) = \sum_{\mathbf{x}_i \in \mathcal{X}_\mathbf{q}^+} \sum_{\mathbf{x}_i \in \mathcal{X}_\mathbf{q}^-} y_{ij} \left(\frac{\psi(\mathbf{q}, \mathbf{x}_i) - \psi(\mathbf{q}, \mathbf{x}_j)}{|\mathcal{X}_\mathbf{q}^+| \cdot |\mathcal{X}_\mathbf{q}^-|} \right)$$

where

$$y_{ij} = \begin{cases} +1 & \mathbf{x}_i \prec_\mathbf{y} \mathbf{x}_j \\ -1 & \mathbf{x}_i \succ_\mathbf{y} \mathbf{x}_j \end{cases}$$

- ϕ_{po} emphasizes directions in feature space which are in some sense correlated with correct rankings.
- Feature map ψ only depends on the query and a single point, rather than the entire list.
- Good property: for a fixed $\mathbf{w}$, the ranking $\mathbf{y}$ that maximizes $\mathbf{w}^T \phi_{po}(\mathbf{q}, \mathbf{y})$ is simply $\mathbf{x}_i \in \mathcal{X}$ sorted by descending $\mathbf{w}^T \psi(\mathbf{q}, \mathbf{x}_i)$.

Structural SVM for Ranking (3)

- Learning structure SVM: 1-slack margin-rescaling cutting-plane algorithm

alternating between:

- optimize the model parameters $\mathbf{w}$
- update the constraint set with a new batch of rankings $(\mathbf{y}_1, \ldots, \mathbf{y}_N)$ (one ranking for each point)

- Reference:
  B. McFee and G. Lanckriet. Metric learning to rank. *Proceedings of the 27-th International Conference on Machine Learning*. 2010.