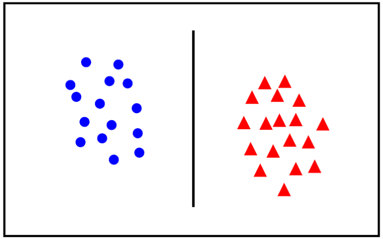
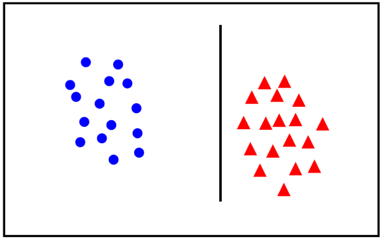
Lecture 4. Kernel Methods

Bao Wang
Department of Mathematics
Scientific Computing and Imaging Institute
University of Utah
Math 5750/6880, Fall 2021

Support Vector Machine

Assume the training data set is linearly separable in feature space, i.e. there exists at least one set of $\boldsymbol{w}$ and b s.t. a function of the form $y(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$ satisfies $y(\boldsymbol{x}_n) > 0 (< 0)$ for points having $t_n = +1(-1)$, i.e., $t_n y(\boldsymbol{x}_n) > 0$ for all training data points.





Which classifier is better? [Theoretical motivation: VC-dimension.]

Support Vector Machine

The maximum margin solution is found by solving

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

Direct solution of this optimization problem would be very complex, and so we shall convert it into an equivalent problem that is much easier to solve.

For any κ, we have

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

Support Vector Machine

We can choose κ s.t. $min_n \Big[t_n(\kappa \boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + \kappa b) \Big] = 1$, and the problem becomes

$$\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{\|\kappa \boldsymbol{w}\|} \right\},$$

s.t.

$$t_n(\kappa \boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + \kappa b) \geq 1, \quad \forall\ n = 1, \cdots, N.$$

The problem is equivalent to

$$\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{\|\boldsymbol{w}\|} \right\},$$

s.t.

$$t_n(\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + b) \geq 1, \quad \forall\ n = 1, \cdots, N.$$

Support Vector Machine: Dual Form

We introduce Lagrange multipliers $a_n \geq 0$, with one multiplier a_n for each of the constraints, giving the Lagrangian function[1]

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{n=1}^{N} a_n\{t_n(\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + b) - 1\}, \quad \boldsymbol{a} = (a_1, \cdots, a_N)^\top. \tag{2}$$

Setting the derivatives of $L(\boldsymbol{w}, b, \boldsymbol{a})$ w.r.t. $\boldsymbol{w}$ and b equal to zero, we have

$$\boldsymbol{w} = \sum_{n=1}^{N} a_n t_n \phi(\boldsymbol{x}_n); \quad 0 = \sum_{n=1}^{N} a_n t_n. \tag{3}$$

[1] KKT condition: If we wish to minimize the function $f(\boldsymbol{x})$ s.t. $g(\boldsymbol{x}) \geq 0$, then we minimize the Lagrangian function $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) - \lambda g(\boldsymbol{x})$ w.r.t. $\boldsymbol{x}$, subject to $\lambda \geq 0$, $g(\boldsymbol{x}) \geq 0$, and $\lambda g(\boldsymbol{x}) = 0$.

Support Vector Machine: Dual Form

Eliminating $\boldsymbol{w}$ and b from $L(\boldsymbol{w}, b, \boldsymbol{a})$ using these conditions then gives the *dual representation* of the maximum margin problem in which we maximize

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m), \quad k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) \text{ is the kernel.} \tag{4}$$

with respect to $\boldsymbol{a}$ subject to the constraints

$$a_n \geq 0; \quad \sum_{n=1}^{N} a_n t_n = 0. \tag{5}$$

The linear classifier becomes

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b. \tag{6}$$

Support Vector Machine (Soft Margin)

$$\min_{\boldsymbol{w},\xi_n} C \sum_{n=1}^{N} \xi_n + \frac{1}{2}\|\boldsymbol{w}\|^2, \;\; C > 0, \tag{7}$$

s.t.

$$t_n(\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + b) \geq 1 - \xi_n, \quad \xi_n \geq 0.$$

Support Vector Machine (Soft Margin): Dual Form

The corresponding Lagrangian is given by

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n\Big\{t_n y(\boldsymbol{x}_n) - 1 + \xi_n\Big\} - \sum_{n=1}^{N}\mu_n\xi_n, \tag{8}$$

where $\{a_n \geq 0\}$ and $\{\mu_n \geq 0\}$ are Lagrange multipliers. The corresponding KKT conditions are

$$\begin{aligned} a_n &\geq 0 \\ t_n y(\boldsymbol{x}_n) - 1 + \xi_n &\geq 0 \\ a_n(t_n y(\boldsymbol{x}_n) - 1 + \xi_n) &= 0 \\ \mu_n &\geq 0 \\ \xi_n &\geq 0 \\ \mu_n\xi_n &= 0 \end{aligned} \tag{9}$$

where $n = 1, \cdots, N$.

Support Vector Machine (Soft Margin): Dual Form

The dual Lagrangian in the form

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m), \tag{10}$$

The dual form of SVM involves a kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$.

Many linear parametric models can be re-cast into an equivalent 'dual representation' in which the prediction are based on linear combinations of a *kernel function* evaluated at the training data points. As we shall see, for models which are based on a fixed nonlinear *feature space* mapping $\phi(\boldsymbol{x})$, the kernel function is given by the relation

$$k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}'). \tag{11}$$

Linear kernel: the feature map is an identity map, i.e. $\phi(\boldsymbol{x}) = \boldsymbol{x}$ and the corresponding kernel is $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'$.

Kernel trick: a.k.a. *kernel substitution*. If we have an algorithm formulated in such a way that the input vector $\boldsymbol{x}$ enters only in the form of scalar products, then we can replace that scalar product with some other choice of kernels.

Dual representation of linear regression

Linear regression model: $y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x})$.

Consider the following regularized sum-of-squares error function for linear regression

$$J(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \{\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) - t_n\}^2 + \frac{\lambda}{2} \boldsymbol{w}^\top \boldsymbol{w}, \quad \lambda \geq 0. \tag{12}$$

What is the dual representation of the linear regression model above?

Dual representation of linear regression

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = 0 \Rightarrow \boldsymbol{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\{\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) - t_n\}\phi(\boldsymbol{x}_n) = \sum_{n=1}^{N} a_n \phi(\boldsymbol{x}_n) = \Phi^\top \boldsymbol{a},$$

where Φ is the design matrix, whose *n*th row is given by $\phi(\boldsymbol{x}_n)^\top$. The vector $\boldsymbol{a} = (a_1, \cdots, a_N)^\top$ with

$$a_n = -\frac{1}{\lambda}\{\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) - t_n\}. \tag{13}$$

Dual representation of linear regression

We can reformulate the least squares algorithm in terms of the parameter vector $\boldsymbol{a}$ instead of $\boldsymbol{w}$, resulting in a *dual representation*. Substitute $\boldsymbol{w} = \Phi^\top \boldsymbol{a}$ into $J(\boldsymbol{w})$ gives

$$J(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^\top \Phi\Phi^\top \Phi\Phi^\top \boldsymbol{a} - \boldsymbol{a}^\top \Phi\Phi^\top \boldsymbol{t} + \frac{1}{2}\boldsymbol{t}^\top \boldsymbol{t} + \frac{\lambda}{2}\boldsymbol{a}^\top \Phi\Phi^\top \boldsymbol{a}, \tag{14}$$

where $\boldsymbol{t} = (t_1, \cdots, t_N)^\top$.

Dual representation of linear regression

Define the *Gram* matrix $\boldsymbol{K} = \Phi\Phi^\top \in \mathbb{R}^{N \times N}$ with $K_{nm} = \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. In terms of the Gram matrix, the sum-of-squares error function can be written as

$$J(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{K}\boldsymbol{K}\boldsymbol{a} - \boldsymbol{a}^\top \boldsymbol{K}\boldsymbol{t} + \frac{1}{2}\boldsymbol{t}^\top \boldsymbol{t} + \frac{\lambda}{2}\boldsymbol{a}^\top \boldsymbol{K}\boldsymbol{a}. \tag{15}$$

$$\nabla_{\boldsymbol{a}} J(\boldsymbol{a}) = 0 \Rightarrow \boldsymbol{a} = (\boldsymbol{K} + \lambda \boldsymbol{I}_N)^{-1}\boldsymbol{t}.$$

Dual representation of linear regression

Substitute this back into the linear regression model, we obtain the following prediction for a new input $\boldsymbol{x}$

$$y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) = \boldsymbol{a}^\top \Phi \phi(\boldsymbol{x}) = \boldsymbol{k}(\boldsymbol{x})^\top (\boldsymbol{K} + \lambda \boldsymbol{I}_N)^{-1} \boldsymbol{t}, \quad \text{Dual formulation!} \tag{16}$$

where we have defined the vector $\boldsymbol{k}(\boldsymbol{x})$ with elements $k_n(\boldsymbol{x}) = k(\boldsymbol{x}_n, \boldsymbol{x})$.

The dual formulation allows the solution to the least-squares problem to be expressed entirely in terms of the kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$.

Note that the prediction at $\boldsymbol{x}$ is given by a linear combination of the target values from the training set.

Dual representation of linear regression

Remark. In the dual formulation, $\boldsymbol{a}$ is determined by inverting an $N \times N$ matrix, whereas in the original parameter space formulation we had to invert an $M \times M$ matrix to determine $\boldsymbol{w}$ (How?). Note that typically $N \gg M$, the dual formulation seems inferior to the original formulation.

However, the advantage of the dual formulation is that it is expressed entirely in terms of the kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$. We can therefore work in terms of kernels and avoid the explicit introducing the feature vector $\phi(\boldsymbol{x})$, allowing us to implicitly use feature spaces of high, even infinite, dimensionality.

Constructing kernels

Approach I. Choose a feature map $\phi(\boldsymbol{x}) \in \mathbb{R}^M$ and then use this to find the corresponding kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}') = \sum_{i=1}^{M} \phi_i(\boldsymbol{x})\phi_i(\boldsymbol{x}') \tag{17}$$

where $\phi_i(x)$ are the basis functions.

Constructing kernels

Approach II. Direct construction: in this case, we must ensure that the function we choose is a valid kernel, i.e. it corresponds to a scalar product in some (perhaps infinite dimensional) feature space. As a simple example, consider a kernel function given by

$$k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^\top \boldsymbol{z})^2. \tag{18}$$

If we take the particular case of a 2D input space $\boldsymbol{x} = (x_1, x_2)$ we can expand out the terms and thereby identify the corresponding nonlinear feature mapping

$$k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^\top \boldsymbol{z})^2 = (x_1 z_1 + x_2 z_2)^2 = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(z_1^2, \sqrt{2} z_1 z_2, z_2^2)^\top = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{z}). \tag{19}$$

We see that the feature mapping takes the form $\phi(\boldsymbol{x}) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^\top$ and therefore comprises all possible second order terms, with a specific weighting between them.

Constructing kernels

When $k(\boldsymbol{x}, \boldsymbol{x}')$ represents a kernel function?

Lemma. *A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ implements an inner product in some Hilbert space if and only if it is positive semidefinite; namely, for all $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m$, the Gram matrix, $G_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, is a positive semidefinite matrix.*

Constructing kernels

Proof. It is trivial to see that if k implements an inner product in some Hilbert space then the Gram matrix is positive semidefinite.

For the other direction, define the space of functions over $\mathcal{X}$ as $\mathbb{R}^{\mathcal{X}} = \{f : \mathcal{X} \to \mathbb{R}\}$. For each $\boldsymbol{x} \in \mathcal{X}$ let $\psi(\boldsymbol{x})$ be the function $\boldsymbol{x} \to K(\cdot, \boldsymbol{x})$. Define a vector space by taking all linear combinations of elements of the form $K(\cdot, \boldsymbol{x})$. Define an inner product on this vector space to be

$$\langle \sum_i \alpha_i K(\cdot, \boldsymbol{x}_i), \sum_j \beta_j K(\cdot, \boldsymbol{x}'_j) \rangle = \sum_{i,j} \alpha_i \beta_j K(\boldsymbol{x}_i, \boldsymbol{x}'_j).$$

This is a valid inner product since it is symmetric (because K is symmetric), it is linear (immediate), and it is positive definite (it is easy to see that $K(\boldsymbol{x}, \boldsymbol{x}) \geq 0$ with equality only for $\psi(\boldsymbol{x})$ being the zero function). Clearly,

$$\langle \psi(\boldsymbol{x}), \psi(\boldsymbol{x}') \rangle = \langle K(\cdot, \boldsymbol{x}), K(\cdot, \boldsymbol{x}') \rangle = K(\boldsymbol{x}, \boldsymbol{x}'),$$

which concludes our proof.

Constructing kernels

Build them out of simpler kernels as building blocks.

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

$$\begin{aligned}
k(\boldsymbol{x}, \boldsymbol{x}') &= ck_1(\boldsymbol{x}, \boldsymbol{x}') \\
k(\boldsymbol{x}, \boldsymbol{x}') &= f(\boldsymbol{x})k_1(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}') \\
k(\boldsymbol{x}, \boldsymbol{x}') &= q(k_1(\boldsymbol{x}, \boldsymbol{x}')) \\
k(\boldsymbol{x}, \boldsymbol{x}') &= \exp(k_1(\boldsymbol{x}, \boldsymbol{x}')) \\
k(\boldsymbol{x}, \boldsymbol{x}') &= k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}') \\
k(\boldsymbol{x}, \boldsymbol{x}') &= k_1(\boldsymbol{x}, \boldsymbol{x}')k_2(\boldsymbol{x}, \boldsymbol{x}') \\
k(\boldsymbol{x}, \boldsymbol{x}') &= k_3(\phi(\boldsymbol{x}), \phi(\boldsymbol{x}')) \\
k(\boldsymbol{x}, \boldsymbol{x}') &= \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}' \\
k(\boldsymbol{x}, \boldsymbol{x}') &= k_a(\boldsymbol{x}_a, \boldsymbol{x}'_a) + k_b(\boldsymbol{x}_b, \boldsymbol{x}'_b) \\
k(\boldsymbol{x}, \boldsymbol{x}') &= k_a(\boldsymbol{x}_a, \boldsymbol{x}'_a)k_b(\boldsymbol{x}_b, \boldsymbol{x}'_b)
\end{aligned}$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\boldsymbol{x})$ is a function from $\boldsymbol{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\boldsymbol{A}$ is p.s.d, $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are variables with $\boldsymbol{x} = (\boldsymbol{x}_a, \boldsymbol{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Polynomial kernels

The k degree polynomial kernel is defined to be

$$K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^k.$$

We will show that there exists a mapping ϕ from the original space to some higher dimensional space for which $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$. For simplicity, denote $x_0 = x_0' = 1$. Then, we have

$$K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^k = \Big(\sum_{j=0}^{n} x_j x_j' \Big) \cdots \Big(\sum_{j=0}^{n} x_j x_j' \Big) = \sum_{J \in \{0,1,\cdots,n\}^k} \prod_{i=1}^{k} x_{J_i} x_{J_i}'$$

$$= \sum_{J \in \{0,1\cdots,n\}^k} \prod_{i=1}^{k} x_{J_i} \prod_{i=1}^{k} x_{J_i}'.$$

Now, if we define $\phi : \mathbb{R}^n \to \mathbb{R}^{(n+1)^k}$ such that for $J \in \{0, 1, \cdots, n\}^k$ there is an element of $\phi(\boldsymbol{x})$ that equals $\prod_{i=1}^{k} x_{J_i}$, we obtain that

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle.$$

Gaussian kernels

Let the original instance space be $\mathbb{R}$ and consider the mapping ϕ where for each nonnegative integer $n \geq 0$ there exists an element $\phi(\boldsymbol{x})_n$ that equals $\frac{1}{\sqrt{n!}} e^{-\frac{x^2}{2}} \boldsymbol{x}^n$. Then,

$$\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle = \sum_{n=0}^{\infty} \left(\frac{1}{\sqrt{n!}} e^{-\frac{x^2}{2}} \boldsymbol{x}^n \right) \left(\frac{1}{\sqrt{n!}} e^{-\frac{(x')^2}{2}} (\boldsymbol{x}')^n \right) = e^{-\frac{x^2+(x')^2}{2}} \sum_{n=0}^{\infty} \left(\frac{(\boldsymbol{x}\boldsymbol{x}')^n}{n!} \right) = e^{-\frac{\|x-x'\|^2}{2}}$$

Here the feature space is of infinite dimension while evaluating the kernel is very simple. More generally, given a scalar $\sigma > 0$, the Gaussian kernel is defined to be

$$K(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|x-x'\|^2}{2\sigma}}.$$

Intuitively, the Gaussian kernel sets the inner product in the feature space between $\boldsymbol{x}, \boldsymbol{x}'$ to be close to zero if the instances are far away from each other (in the original domain) and close to 1 if they are close. σ is a parameter that controls the scale determining what we mean by "close".

The Gaussian kernel is also called the RBF kernel.

Kernel trick

$$\langle \boldsymbol{x}, \boldsymbol{x}' \rangle \Rightarrow \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle \Rightarrow k(\boldsymbol{x}, \boldsymbol{x}').$$