CSC 246 Machine Learning | Lecture 8

Kernel Methods

Lecturer: Jiaming Liang | *October 6, 2026*

In Lecture 7, we derived the dual formulation of SVM. A key observation was that the training examples appear in the dual problem only through inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Moreover, the resulting classifier can be written in the form

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b.$$

This observation suggests a way to construct nonlinear classifiers. Suppose that we first transform the input through a feature map $\psi : \mathcal{X} \to \mathcal{F}$, and then apply a linear classifier in the feature space $\mathcal{F}$. The SVM dual in the feature space would depend on the data only through $\langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle_{\mathcal{F}}$. If these inner products can be computed efficiently without explicitly constructing $\psi(\mathbf{x})$, then we can learn nonlinear predictors without explicitly working in the high-dimensional feature space.

This is the central idea behind kernel methods. Throughout this lecture, we often omit the intercept b when it is not essential to the discussion.

1 Feature Spaces

A *feature mapping* is a function $\psi : \mathcal{X} \to \mathcal{F}$, where $\mathcal{F}$ is an inner-product space called the *feature space*. A linear predictor in the feature space has the form

$$f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \psi(\mathbf{x}) \rangle_{\mathcal{F}}.$$

Although this predictor is linear in the transformed features $\psi(\mathbf{x})$, it can be nonlinear as a function of the original input $\mathbf{x}$. For example, consider a one-dimensional input and the feature map

$$\psi(x) = (x, x^2).$$

A linear predictor in the feature space becomes

$$\langle \mathbf{w}, \psi(x) \rangle = w_1 x + w_2 x^2,$$

which is quadratic in x.

More generally, a polynomial of degree at most k, $p(x) = w_0 + w_1 x + \cdots + w_k x^k$, can be written as a linear function using

$$\psi(x) = (1, x, x^2, \ldots, x^k).$$

Thus, polynomial prediction in the original input space becomes linear prediction in a larger feature space.

The same idea applies in several dimensions. If $\mathbf{x} = (x_1, x_2)$, a quadratic feature map is

$$\psi(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2).$$

A hyperplane in this feature space corresponds to a quadratic decision boundary in the original two-dimensional space.

Feature mappings therefore allow us to increase the expressive power of a linear model. The difficulty is that the dimension of the feature space can grow very quickly. For example, including all degree-k interactions of d input variables can require on the order of d^k features.

The kernel trick allows us to avoid explicitly constructing these features.

2 The Kernel Trick

Given a feature map ψ, define the corresponding *kernel function* by

$$K(\mathbf{x}, \mathbf{x}') = \left\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \right\rangle_{\mathcal{F}}.$$

If $K(\mathbf{x}, \mathbf{x}')$ can be computed directly from the original inputs, then we can evaluate inner products in the feature space without ever forming the feature vectors themselves. For example, the quadratic term in the SVM dual changes from

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

to

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j).$$

Similarly, prediction becomes

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b.$$

Thus, an SVM can operate as if it were learning a linear classifier in a possibly very high-dimensional feature space, while all computations are performed through the kernel function in the original input space.

2.1 Why the Solution Lies in the Span of the Training Features

The same structure arises more generally in norm-regularized learning. Consider an optimization problem of the form

$$\min_{\mathbf{w} \in \mathcal{F}} f\left(\langle \mathbf{w}, \psi(\mathbf{x}_1) \rangle, \ldots, \langle \mathbf{w}, \psi(\mathbf{x}_m) \rangle\right) + R(\|\mathbf{w}\|),$$

where R is nondecreasing. Let $V = \text{span}\{\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_m)\}$. Any $\mathbf{w} \in \mathcal{F}$ can be decomposed as $\mathbf{w} = \mathbf{w}_\parallel + \mathbf{w}_\perp$, where $\mathbf{w}_\parallel \in V$ and $\mathbf{w}_\perp \perp V$. Because each $\psi(\mathbf{x}_i) \in V$, $\langle \mathbf{w}, \psi(\mathbf{x}_i)\rangle = \langle \mathbf{w}_\parallel, \psi(\mathbf{x}_i)\rangle$. Therefore, removing $\mathbf{w}_\perp$ does not change the prediction on any training example.

On the other hand,

$$\|\mathbf{w}\|^2 = \|\mathbf{w}_\parallel\|^2 + \|\mathbf{w}_\perp\|^2 \geq \|\mathbf{w}_\parallel\|^2.$$

Hence, the orthogonal component cannot improve the objective. There is therefore always an optimal solution of the form

$$\mathbf{w} = \sum_{j=1}^{m} \beta_j \psi(\mathbf{x}_j).$$

For such a solution,

$$\langle \mathbf{w}, \psi(\mathbf{x}_i)\rangle = \sum_{j=1}^{m} \beta_j K(\mathbf{x}_j, \mathbf{x}_i),$$

and

$$\|\mathbf{w}\|^2 = \sum_{i=1}^{m}\sum_{j=1}^{m} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j).$$

Define the *Gram matrix* $G \in \mathbb{R}^{m\times m}$ by $G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. Then the predictions on the training set are determined by $G\boldsymbol{\beta}$, and $\|\mathbf{w}\|^2 = \boldsymbol{\beta}^\top G\boldsymbol{\beta}$.

Thus, both the SVM dual and more general norm-regularized learning problems can be expressed entirely in terms of kernel evaluations.

3 Examples of Kernels

3.1 Polynomial Kernels

A standard example is the degree-k polynomial kernel

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

To see the corresponding feature representation, consider $k = 2$ and $\mathbf{x} = (x_1, x_2)$. Then

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= (1 + x_1x_1' + x_2x_2')^2 \\ &= 1 + 2x_1x_1' + 2x_2x_2' + x_1^2(x_1')^2 + 2x_1x_2x_1'x_2' + x_2^2(x_2')^2. \end{aligned}$$

Therefore, if

$$\psi(\mathbf{x}) = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2\right),$$

then

$$K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}')\rangle.$$

The same principle holds for arbitrary k. Expanding $(1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k$ produces all appropriate monomials of degree up to k. Hence, the polynomial kernel implicitly uses a large collection of polynomial features. The computational advantage is that we never need to enumerate these features. We only compute $(1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k$.

3.2 The Gaussian Kernel

An even more striking example is the Gaussian, or radial basis function (RBF), kernel. In one dimension, consider the infinite-dimensional feature map whose n-th coordinate is

$$\psi_n(x) = \frac{1}{\sqrt{n!}} e^{-x^2/2} x^n, \qquad n = 0, 1, 2, \ldots.$$

Then

$$\langle \psi(x), \psi(x') \rangle = e^{-(x^2+(x')^2)/2} \sum_{n=0}^{\infty} \frac{(xx')^n}{n!} = e^{-(x^2+(x')^2)/2} e^{xx'} = e^{-(x-x')^2/2},$$

where we used $e^u = \sum_{n=0}^{\infty} u^n/n!$. This motivates the Gaussian kernel in $\mathbb{R}^d$:

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

where $\sigma > 0$ determines the scale of the similarity. If $\mathbf{x}$ and $\mathbf{x}'$ are close, then $K(\mathbf{x}, \mathbf{x}')$ is close to 1. If they are far apart, the kernel is close to 0.

The important point is that this simple scalar expression corresponds to an inner product in an infinite-dimensional feature space.

3.3 Kernels as Prior Knowledge

Choosing a kernel amounts to choosing a feature representation and therefore specifying what kinds of relationships we expect to be useful.

For example, a polynomial kernel is appropriate when polynomial interactions among the original features are expected to matter. A Gaussian kernel instead encodes the idea that nearby inputs should be regarded as similar, with σ determining the relevant scale.

Kernel methods are not limited to vector-valued inputs. Kernels can also be defined on strings, trees, graphs, and other structured objects. Instead of explicitly converting such objects into fixed-dimensional vectors, we may design a kernel that directly measures an appropriate notion of similarity.

The kernel therefore plays two roles: it provides a computational shortcut and it determines the representation used by the learning algorithm.

4 Soft-SVM with Kernels

We now return to Soft-SVM. In the feature space associated with ψ, the regularized hinge-loss problem is

$$\min_{\mathbf{w}} \left\{ \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} [1 - y_i\langle \mathbf{w}, \psi(\mathbf{x}_i)\rangle]_+ \right\}.$$

From the span argument above, we may restrict attention to solutions of the form

$$\mathbf{w} = \sum_{j=1}^{m} \beta_j \psi(\mathbf{x}_j).$$

For training example i,

$$\langle \mathbf{w}, \psi(\mathbf{x}_i)\rangle = \sum_{j=1}^{m} \beta_j K(\mathbf{x}_j, \mathbf{x}_i) = (G\boldsymbol{\beta})_i.$$

Moreover, $\|\mathbf{w}\|^2 = \boldsymbol{\beta}^\top G\boldsymbol{\beta}$. Therefore, the feature-space Soft-SVM problem can be written entirely in terms of $\boldsymbol{\beta}$:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^m} \left\{ \frac{\lambda}{2}\boldsymbol{\beta}^\top G\boldsymbol{\beta} + \frac{1}{m}\sum_{i=1}^{m} [1 - y_i(G\boldsymbol{\beta})_i]_+ \right\}.$$

No explicit feature vector appears in this optimization problem. The entire problem is determined by the Gram matrix G.

Once the coefficients have been learned, prediction on a new point is

$$f(\mathbf{x}) = \sum_{j=1}^{m} \beta_j K(\mathbf{x}_j, \mathbf{x}),$$

and the classifier is

$$h(\mathbf{x}) = \operatorname{sign}\left(\sum_{j=1}^{m} \beta_j K(\mathbf{x}_j, \mathbf{x}) \right).$$

This gives the general kernel representation of Soft-SVM: the model behaves like a linear classifier in the feature space, but training and prediction require only kernel evaluations.