

Support vector machines with non-linear kernels

Fraida Fund

Contents

Kernel SVMs	2
Review: Solution to SVM dual problem	2
Extension to non-linear decision boundary	2
SVM with basis function transformation	2
Example (1)	2
Example (2)	2
Example (3)	3
Example (4)	3
Kernel trick	3
Kernel as a similarity measure	4
Linear kernel	5
Polynomial kernel	5
Using infinite-dimension feature space	6
Radial basis function kernel	6
Infinite-dimensional feature space	6
Infinite-dimensional feature space (extra steps not shown in class)	6
Infinite-dimensional feature space (2)	7
Infinite-dimensional feature space (3)	7
Feature mapping vs kernel	7
Kernel: not just for SVM	7
Summary: SVM	8
Key expression	8
Key ideas	8

Math prerequisites for this lecture: You should know about complexity of algorithms (Big O notation).

Kernel SVMs

Review: Solution to SVM dual problem

Given a set of support vectors S and associated α for each,

$$z = w_0 + \sum_{i \in S} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x}_t \rangle$$
$$\hat{y} = \text{sign}(z)$$

Measures inner product (a kind of “correlation”) between new sample and each support vector.

For the geometric intuition/why inner product measures the similarity between two vectors, watch: [3Blue1Brown series S1 E9: Dot products and duality](#).

This SVM assumes a linear decision boundary. (The expression for z gives the equation of the hyperplane that separates the classes.)

Extension to non-linear decision boundary

- For logistic regression: we used basis functions of \mathbf{x} to transform the feature space and classify data with non-linear decision boundary.
- Could use similar approach here?

SVM with basis function transformation

Given a set of support vectors S and associated α for each,

$$z = w_0 + \sum_{i \in S} \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_t) \rangle$$
$$\hat{y} = \text{sign}(z)$$

Note: the output of $\phi(\mathbf{x})$ is a vector that may or may not have the same dimensions as \mathbf{x} .

Example (1)

Suppose we are given a dataset of feature-label pairs in \mathbb{R}^1 :

$$(-1, -1), (0, -1), (1, -1), (-3, +1), (-2, +1), (3, +1)$$

This data is not linearly separable.

Example (2)

Now suppose we map from \mathbb{R}^1 to \mathbb{R}^2 using $\phi(x) = (x, x^2)$:

$$((-1, 1) - 1), ((0, 0), -1), ((1, 1), -1),$$
$$((-3, 9) + 1), ((-2, 4) + 1), ((3, 9) + 1)$$

This data is linearly separable in \mathbb{R}^2 .

Example (3)

Suppose we compute $\langle \phi(x_i), \phi(x_t) \rangle$ directly:

- compute $\phi(x_i)$
- compute $\phi(x_t)$
- take inner product

How many operations (exponentiation, multiplication, division, addition, subtraction) are needed?

For each computation of $\langle \phi(x_i), \phi(x_t) \rangle$, we need five operations:

- (one square) find $\phi(x_i) = (x_i, x_i^2)$
- (one square) find $\phi(x_t) = (x_t, x_t^2)$
- (two multiplications, one sum) find $\langle \phi(x_i), \phi(x_t) \rangle = x_i x_t + x_i^2 x_t^2$

Example (4)

What if we express $\langle \phi(x_i), \phi(x_t) \rangle$ as

$$K(x_i, x_t) = x_i x_t (1 + x_i x_t)$$

How many operations (exponentiation, multiplication, division, addition, subtraction) are needed to compute this equivalent expression?

Each computation of $K(x_i, x_t)$ requires three operations:

- (one multiplication) compute $x_i x_t$
- (one sum) compute $1 + x_i x_t$
- (one multiplication) compute $x_i x_t (1 + x_i x_t)$

Kernel trick

- Suppose kernel $K(\mathbf{x}_i, \mathbf{x}_t)$ computes inner product in transformed feature space $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_t) \rangle$
- For the SVM:

$$z = w_0 + \sum_{i \in S} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_t)$$

- We don't need to explicitly compute $\phi(\mathbf{x})$ if computing $K(\mathbf{x}_i, \mathbf{x}_t)$ is more efficient

Note that the expression we use to find the α_i values also only depends on the inner product, so the kernel works there as well.

Another example:

$$\begin{aligned}
K(x, z) &= (x^T z + c)^2 \\
&= (x^T z)(x^T z) + 2c(x^T z) + c^2 \\
&= \sum_{i=1}^p \sum_{j=1}^p (x_i x_j)(z_i z_j) + 2c \sum_{i=1}^p x_i z_i + c^2
\end{aligned}$$

This corresponds to the feature mapping:

$$\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ \vdots \\ x_p x_p \\ \sqrt{2c} x_1 \\ \vdots \\ \sqrt{2c} x_p \\ c \end{bmatrix}$$

More generally:

$$K(x, z) = (x^T z + c)^d$$

is the polynomial kernel of degree d . If each sample has p features, this corresponds to a feature mapping into a $\binom{p+d}{d}$ dimensional feature space, which grows on the order of $O(p^d)$. But computing the kernel only requires an inner product, which is $O(p)$.

Kernel as a similarity measure

- $K(\mathbf{x}_i, \mathbf{x}_t)$ measures “similarity” between training sample \mathbf{x}_i and new sample \mathbf{x}_t
- Large K , more similarity; K close to zero, not much similarity
- $z = w_0 + \sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_t)$ gives more weight to support vectors that are similar to new sample - those support vectors’ labels “count” more toward the label of the new sample.

Linear kernel

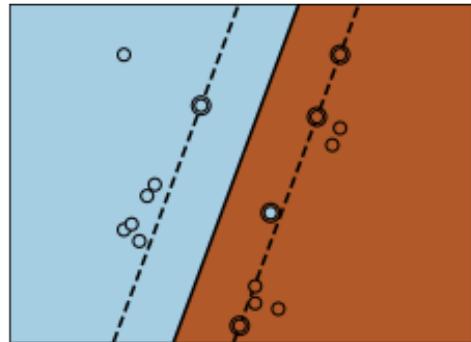


Figure 1: Linear kernel: $K(x_i, x_t) = x_i^T x_t$

Polynomial kernel

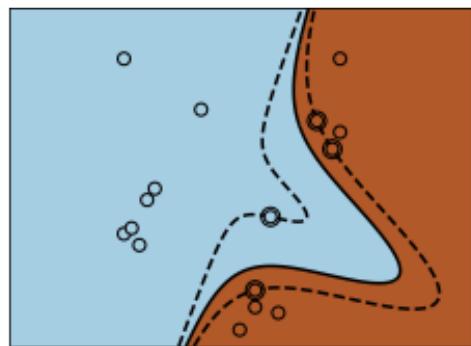


Figure 2: Polynomial kernel: $K(x_i, x_t) = (\gamma x_i^T x_t + c_0)^d$

Using infinite-dimension feature space

Radial basis function kernel

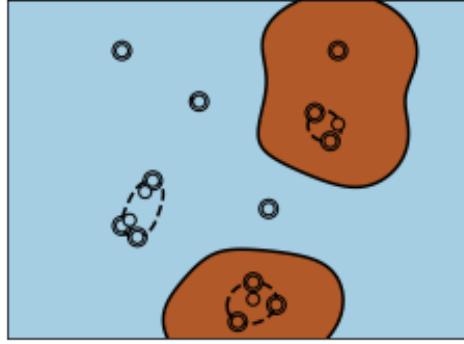


Figure 3: Radial basis function: $K(x_i, x_t) = \exp(-\gamma \|x_i - x_t\|^2)$. If $\gamma = \frac{1}{\sigma^2}$, this is known as the Gaussian kernel with variance σ^2 .

Infinite-dimensional feature space

With kernel method, can operate in infinite-dimensional feature space! Take for example the RBF kernel:

$$K_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_t) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_t\|^2\right)$$

Let $\gamma = \frac{1}{2}$ and let $K_{\text{poly}(r)}$ be the polynomial kernel of degree r . Then

Infinite-dimensional feature space (extra steps not shown in class)

$$\begin{aligned} K_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_t) &= \exp\left(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_t\|^2\right) \\ &= \exp\left(-\frac{1}{2}\langle \mathbf{x}_i - \mathbf{x}_t, \mathbf{x}_i - \mathbf{x}_t \rangle\right) \\ &\stackrel{*}{=} \exp\left(-\frac{1}{2}(\langle \mathbf{x}_i, \mathbf{x}_i \rangle - \langle \mathbf{x}_i, \mathbf{x}_t \rangle - \langle \mathbf{x}_t, \mathbf{x}_i \rangle + \langle \mathbf{x}_t, \mathbf{x}_t \rangle)\right) \\ &= \exp\left(-\frac{1}{2}(\langle \mathbf{x}_i, \mathbf{x}_i \rangle + \langle \mathbf{x}_t, \mathbf{x}_t \rangle - 2\langle \mathbf{x}_i, \mathbf{x}_t \rangle)\right) \\ &= \exp\left(-\frac{1}{2}\|\mathbf{x}_i\|^2\right) \exp\left(-\frac{1}{2}\|\mathbf{x}_t\|^2\right) \exp\left(\langle \mathbf{x}_i, \mathbf{x}_t \rangle\right) \end{aligned}$$

where the steps marked with a star use the fact that for inner products, $\langle \mathbf{u} + \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{u}, \mathbf{w} \rangle + \langle \mathbf{v}, \mathbf{w} \rangle$.

Also recall that $\langle x, x \rangle = \|x\|^2$.

Infinite-dimensional feature space (2)

Eventually, $K_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_t) = e^{-\frac{1}{2}\|\mathbf{x}_i\|^2} e^{-\frac{1}{2}\|\mathbf{x}_t\|^2} e^{\langle \mathbf{x}_i, \mathbf{x}_t \rangle}$

Let $C \equiv \exp\left(-\frac{1}{2}\|\mathbf{x}_i\|^2\right) \exp\left(-\frac{1}{2}\|\mathbf{x}_t\|^2\right)$

And note that the Taylor expansion of $e^{f(x)}$ is:

$$e^{f(x)} = \sum_{r=0}^{\infty} \frac{[f(x)]^r}{r!}$$

C is a constant - it can be computed in advance for every x individually.

Infinite-dimensional feature space (3)

Finally, the RBF kernel can be viewed as an infinite sum over polynomial kernels:

$$\begin{aligned} K_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_t) &= C e^{\langle \mathbf{x}_i, \mathbf{x}_t \rangle} \\ &= C \sum_{r=0}^{\infty} \frac{\langle \mathbf{x}_i, \mathbf{x}_t \rangle^r}{r!} \\ &= C \sum_{r=0}^{\infty} \frac{K_{\text{poly}(r)}(\mathbf{x}_i, \mathbf{x}_t)}{r!} \end{aligned}$$

TL;DR: RBF kernel is like comparing similarity of two samples on *infinite features* without computing them.

Feature mapping vs kernel

- **First approach:** basis function transformation AKA feature mapping
- **Current approach:** kernel - work in transformed space without explicit transformation
- **Next lesson:** wait and see!

A basis function transformation can be expensive if the dimensionality of the transformed feature space is large. With a kernel approach, we can work very efficiently in high dimensional feature space.

Kernel: not just for SVM

Kernels are used in other types of models -

- Kernel regression
- Gaussian process regression

Summary: SVM

Key expression

Decision boundary can be computed using an inexpensive kernel function on a small number of support vectors:

$$z = w_0 + \sum_{i \in S} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_t)$$

($i \in S$ are the subset of training samples that are support vectors)

Key ideas

- Boundary with max separation between classes
- Tuning hyperparameters controls complexity
 - C for width of margin/number of support vectors
 - also kernel-specific hyperparameters
- Kernel trick allows efficient extension to higher-dimension space: non-linear decision boundary through transformation of features, but without explicitly computing high-dimensional features.