1. Kernel Trick

Note that the final classifier of an SVM depends on \( \mathbf{x} \) only by dot products. The final classifier is \( \hat{y}(\mathbf{x}) = \text{sign}(\sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x}) \). This depends on \( \mathbf{x} \) only by: (i) the dot product \( \mathbf{x} \cdot \mathbf{x}_i \), and (ii) the \( \alpha \)'s depend on solving the dual problem (maximizing the dual) which again depends only of the dot products of the data \( \mathbf{x}_i \cdot \mathbf{x}_j \).

This motivates the Kernel Trick

Compute features \( \varphi(\mathbf{x}) \) and reformulate the problem in feature space – i.e. seek a classifier of form:

\[
\text{sign}(c \cdot \varphi(\mathbf{x}) + b)
\]

Replace \( \mathbf{x} \) by \( \varphi(\mathbf{x}) \) everywhere in the primal & dual formulation. Then the classifier only depends on the dot product of the \( \varphi(\mathbf{x}) \)'s:

I.e. on the Kernel \( K(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x}) \cdot \varphi(\mathbf{x}') \)

2. Why does this help?

First, using features \( \phi(.) \) can make it possible to classify data by hyperplanes, which we could not classify in the original space.

Example

Logical X-OR, \( \mathbf{x} = (x_1, x_2), x_j \in \{\pm 1\}, \omega \in \{\pm 1\} \)
The X-OR (exclusive or), see figure (1), requires a decision rule

\[ \alpha(x) \text{ s.t.} \]
\[ \alpha(1, 1) = \alpha(-1, -1) = 1 \]
\[ \alpha(1, -1) = \alpha(-1, 1) = -1 \]

**Figure 1.** Data for the logical X-or problem. It is impossible to separate the positive and negative examples by a straight line (i.e. to classify them correctly by a linear classifier). But we can find features which will enable us to do this.

It is impossible to find a linear classifier to do this. But define feature \( \varphi(x_1, x_2) = (x_1, x_2, x_1 x_2) \). Now the classifier sign \( \{(0, 0, 1) \cdot \varphi(x_1, x_2)\} \) can separate the data.

**Moral:** increasing the dimensionality of the data by features, makes it possible to find separating hyperplanes.

**Second,** we do not need to specify the features \( \varphi(x) \) explicitly, we only need to specify the kernel

\[ K(x, x') = \varphi(x) \cdot \varphi(x') \]

Remember: the dual problem reduces to maximizing

\[ L_d(\{\alpha_\mu\}) = \sum_\mu \alpha_\mu - \frac{1}{2} \sum_{\mu, \nu} \alpha_\mu \alpha_\nu \omega_\mu \omega_\nu \varphi(x_\mu) \cdot \varphi(x_\nu) \]
LECTURE NOTE #9

$$= \sum_{\mu} \alpha_{\mu} - \frac{1}{2} \sum_{\mu,\nu} \alpha_{\mu} \omega_{\mu} \omega_{\nu} K(x_{\mu}, x_{\nu})$$

The solution is $$\hat{a} = \sum_{\mu} \hat{\alpha}_{\mu} \omega_{\mu} \varphi(x_{\mu})$$
$$\hat{a} \cdot \varphi(x) = \sum_{\mu} \hat{\alpha}_{\mu} \omega_{\mu} \varphi(x) \cdot \varphi(x_{\mu}) = \sum_{\mu} \hat{\alpha}_{\mu} \omega_{\mu} K(x, x_{\mu})$$
(Can solve for $$\hat{\sigma}$$ as before)

3. **What Kernels to Use?**

There are many choices of kernels. The difficulty is knowing which one to use. As always, cross-validation is useful for checking whether a kernel can generalize.

$$K(x, x') = (1 + x \cdot x')^d$$

$$K(x, x') = e^{-\frac{1}{\sigma^2}||x-x'||^2}$$

$$K(x, x') = \tanh\{C_1 x \cdot x' + C_2\}$$

Choice of best kernel is problem dependent.

Some kernels→ e.g. $$(1 + x \cdot x')^d$$ naturally generalized the idea of hyperplanes.

Others → e.g. $$e^{-\frac{1}{\sigma^2}||x-x'||^2}$$ are similar to nearest neighbors.

4. **When do Kernels Correspond to Features?**

Suppose we specify $$K(x, x')$$, is it equal to $$\varphi(x) \cdot \varphi(x')$$ for some features $$\varphi(x)$$?
Figure 2. One type of kernel, e.g. \( \{1 + \vec{x} \cdot \vec{x}'\}^d \), corresponds to using curved surfaces to separate the data. The other type of kernel, \( \exp\{-|\vec{x} - \vec{x}'|^2\} \) is like nearest neighbour.

Theoretical results can be obtained.

e.g. Mercer’s Theorem

Compute eigenfunctions of \( K(\vec{x}, \vec{x}') \)

\[
\int K(\vec{x}, \vec{x}') \psi(\vec{x}')d\vec{x}' = \lambda \psi(\vec{x}) \quad \text{with} \quad \int \{\psi(\vec{x})\}^2 d\vec{x} \quad \text{finite.}
\]

Provided \( K(\vec{x}, \vec{x}') \) is positive definite, then the features are \( \varphi^\mu(\vec{x}) = \sqrt{\lambda^\mu} \psi^\mu(\vec{x}) \)

Similar to linear algebra expansion of a symmetric matrix in terms of eigenvectors.

\[
A_{ij} = \sum_\mu \lambda^\mu e^\mu_i e^\mu_j, \quad \text{where} \quad \sum_j A_{ij} e_j^\mu = \lambda^\mu e^\mu_i
\]

If \( A_{ij} \) is positive definite.

\[
A_{ij} = \sum_\mu \{\lambda^\mu_{1/2} e^\mu_i\} \{\lambda^\mu_{1/2} e^\mu_j\} = \sum_\mu \varphi^\mu_i \cdot \varphi^\mu_j
\]

5. Kernel PCA
The kernel trick can be applied to an quadratic problem - e.g. PCA

\[ C = \frac{1}{m} \sum_{k=1}^{m} (x_k - \bar{x})(x_k - \bar{x})^T \]

w.l.o.g. \( \bar{x} = \frac{1}{m} \sum_{k=1}^{m} x_k = 0 \)

Go to feature space

\[ x \rightarrow \varphi(x) \]

\[ C = \frac{1}{m} \sum_{k=1}^{m} \varphi(x_k)\varphi^T(x_k) \]

All non-zero eigenvectors \( e \) of \( C \) are of form

\[ e = \sum_{j=1}^{m} \alpha_j \varphi(x_j), \text{ for some } \{\alpha_j\} \]

Substituting: \( C e = \lambda e \)

\[ \rightarrow \frac{1}{m} \sum_{k=1}^{m} \varphi(x_k)\{(\varphi(x_k) \cdot e)\} = \lambda e \]

\[ \rightarrow \frac{1}{m} \sum_{k=1}^{m} \varphi(x_k) \sum_{j=1}^{m} \alpha_j \{\varphi(x_k) \cdot \varphi(x_j)\} = \lambda \alpha_j \varphi(x_j) \]

Equating coefficients of \( \varphi(x_j) \) gives new eigenvalue equations.

\[ \frac{1}{m} \sum_{j} K(x_k, x_j) \alpha_j = \lambda \alpha_k \]

Index \( \lambda^\mu, \alpha_k^\mu \)

\[ \frac{1}{m} \sum_{j} K(x_k, x_j) \alpha_j^\mu = \lambda^\mu \alpha_k^\mu \quad \mu = 1 \text{ to } m \]

Solving this, gives us the eigenvectors.

\[ e^\mu = \sum_{j=1}^{m} \alpha_j^\mu \varphi(x_j), \text{ eigenvalue } \lambda^\mu. \text{ (depends on } \varphi) \]
But the projections $\mathbf{e}^\mu \cdot \varphi(x)$ of the data are

$$\mathbf{e}^\mu \cdot \varphi(x) = \sum_{j=1}^{m} \alpha_j^\mu K(x_j, x)$$

which is independent of $\varphi$ and depends only on $K(.,.)$.

Hence:

The projection of the data onto the eigenvectors requires only knowing the kernel $K(x_i, x_j)$ (i.e. not knowing $\varphi$)

Knowledge of the kernel is used twice:

(1) to compute the $\{\alpha_j^\mu\}$

(2) to compute the projections $\mathbf{e}^\mu \cdot \varphi(x)$