Spring 2013.

Note Title 11/19/2006

Kernel Trick.

Note that the final classifier depends on $\underline{x}$ only by dot products.

EG. $\underline{x} \cdot \underline{x}_\mu$ in final classifier.

$\underline{x}_\mu \cdot \underline{x}_\nu$ in the dual energy.

This motivates the Kernel Trick

Compute features $\underline{\varphi}(\underline{x})$ and reformulate the problem in feature space.

→ i.e. seek a classifier of form

$$\text{sign}(\underline{c} \cdot \underline{\varphi}(\underline{x}) + b).$$

Replace $\underline{x}$ by $\underline{\varphi}(\underline{x})$ everywhere in the primal & dual formulation.

Then the classifier only depends on the dot product of the $\underline{\varphi}(\underline{x})$'s.

On the kernel $K(\underline{x}, \underline{x}') = \underline{\varphi}(\underline{x}) \cdot \underline{\varphi}(\underline{x}')$.

Spring 2013.

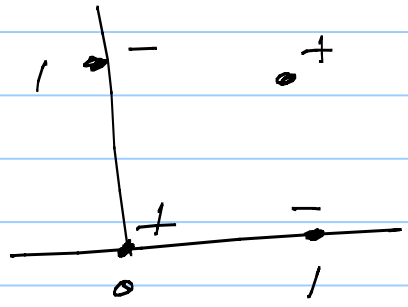
Why does this help?

First, features can make it possible to classify data by hyperplanes.

Example Logical X-OR, $\underline{x} = (x_1, x_2)$ $x_j \in \{\pm 1\}$, $w \in \{\pm 1\}$

The X-OR (exclusive or) requires a decision rule $\alpha(\underline{x})$ s.t.

$$\alpha(1,1) = \alpha(-1,-1) = 1$$
$$\alpha(1,-1) = \alpha(-1,1) = -1$$



Impossible to find a linear classifier to do this.

But define features $\underline{\varphi}(x_1, x_2) = (x_1, x_2, x_1 x_2)$.

Now the classifier $\text{sign}\{(0,0,1) \cdot \underline{\varphi}(x_1, x_2)\}$ can separate the data.

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

(3)

Spring 2013

Second, we do not need to specify the features $\underline{\varphi}(\underline{x})$ explicitly, we only need to specify the kernel

$$K(\underline{x}, \underline{x}') = \underline{\varphi}(\underline{x}) \cdot \underline{\varphi}(\underline{x}')$$

Remember:

the dual problem reduces to maximizing $L_d(\{\alpha_\mu\}) = \sum_\mu \alpha_\mu - \frac{1}{2} \sum_{\mu,\nu} \alpha_\mu \alpha_\nu \omega_\mu \omega_\nu \underline{\varphi}(\underline{x}_\mu) \cdot \underline{\varphi}(\underline{x}_\nu)$

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

The solution $\hat{\underline{a}} = \sum_\mu \hat{\alpha}_\mu \omega_\mu \underline{\varphi}(\underline{x}_\mu)$

$$\hat{\underline{a}} \cdot \underline{\varphi}(\underline{x}) = \sum_\mu \hat{\alpha}_\mu \omega_\mu \underline{\varphi}(\underline{x}) \cdot \underline{\varphi}(\underline{x}_\mu)$$

$$= \sum_\mu \hat{\alpha}_\mu \omega_\mu K(\underline{x}, \underline{x}_\mu).$$

I can solve for $\hat{b}$ as before!

(4.)

Spring 2013

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

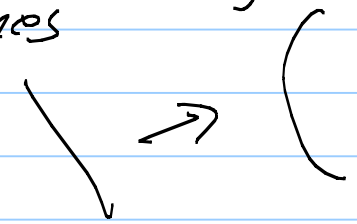
$$K(\underline{x}, \underline{x}') = (1 + \underline{x} \cdot \underline{x}')^d$$

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

$$K(\underline{x}, \underline{x}') = \tanh(c_1 \underline{x} \cdot \underline{x}' + c_2)$$

Choice of best kernel is problem dependent.

Some kernels → eg. $(1 + \underline{x} \cdot \underline{x}')^d$ naturally generalize the idea of hyperplanes



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

(5)

Spring 2013,

When do Kernels Correspond to features?

I.e. if we specify $K(\underline{x}, \underline{x}')$, is it equal to $\underline{\phi}(\underline{x}) \cdot \underline{\phi}(\underline{x}')$ for some features $\underline{\phi}(\underline{x})$?

Theoretical results can be obtained, e.g. Mercer's Theorem.

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

$$\int K(\underline{x}, \underline{x}') \psi(\underline{x}') d\underline{x}' = \lambda \psi(\underline{x})$$

with $\int (\psi(\underline{x}))^2 d\underline{x}$ finite.

Provided $K(\underline{x}, \underline{x}')$ is positive definite, then the features are $\underline{\phi}^{\mu}(\underline{x}) = \sqrt{\lambda_\mu} \psi_\mu(\underline{x})$.

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

$$A_{ij} = \sum_\mu \lambda_\mu e^\mu_i e^\mu_j, \quad \text{where} \quad \sum_j A_{ij} e^\mu_j = \lambda_\mu e^\mu_i.$$

If A_{ij} is the double. $A_{ij} = \sum_\mu \left\{ \lambda_\mu^{1/2} e^\mu_i \right\} \left\{ \lambda_\mu^{1/2} e^\mu_j \right\} = \sum_\mu \phi^\mu_i \phi^\mu_j$

Spring 2023

Kernel PCA

The kernel trick can be applied to an quadratic problem – eg PCA.

$$\underline{\underline{C}} = \frac{1}{m}\sum_{k=1}^{m} (\underline{x}_k - \bar{\underline{x}})(\underline{x}_k - \bar{\underline{x}})^T$$

w.l.o.g. $\bar{\underline{x}} = \frac{1}{m}\sum_{k=1}^{m} \underline{x}_k = 0$.

Go to feature space

$$\underline{x} \to \underline{\varphi}(\underline{x})$$

$$\to \underline{\underline{C}} = \frac{1}{m}\sum_{k=1}^{m} \underline{\varphi}(\underline{x}_k)\,\underline{\varphi}(\underline{x}_k)^T$$

All non-zero eigenvectors $\underline{e}$ of $\underline{\underline{C}}$ are of form

$$\underline{e} = \sum_{j=1}^{m} \alpha_j \underline{\varphi}(\underline{x}_j), \quad \text{for some } \{\alpha_j\},$$

Substituting: $\underline{\underline{C}}\,\underline{e} = \lambda \underline{e}$

$$\to \frac{1}{m}\sum_{k=1}^{m} \underline{\varphi}(\underline{x}_k)\{\underline{\varphi}(\underline{x}_k)\cdot\underline{e}\} = \lambda \underline{e}$$

$$\to \frac{1}{m}\sum_{k=1}^{m} \underline{\varphi}(\underline{x}_k)\sum_{j=1}^{m} \alpha_j (\underline{\varphi}(\underline{x}_k)\cdot\underline{\varphi}(\underline{x}_j)) = \lambda \sum_{j=1}^{m} \alpha_j \underline{\varphi}(\underline{x}_j)$$

Equating coeffs of $\underline{\varphi}(\underline{x}_j)$ gives new eigenvalue equations

$$\frac{1}{m}\sum_j K(\underline{x}_k, \underline{x}_j)\alpha_j = \lambda \alpha_k \qquad \text{Index } \lambda^M,\ \alpha_k^M$$

(7)

Spring 2013,

$$\frac{1}{m}\sum_j K(\underline{x}_k, \underline{x}_j)\alpha_j^\mu = \lambda^\mu \alpha_k^\mu \qquad \mu = 1, \ldots, m.$$

Solving this, gives us the eigenvectors

$$\underline{e}^\mu = \sum_{j=1}^{m} \alpha_j^\mu \, \underline{\phi}(\underline{x}_j) \;, \text{ eigenvalue } \lambda^\mu.$$

(depends on ϕ)

But the projections $\underline{e}^\mu \cdot \underline{\phi}(\underline{x})$ of the data are

$$\underline{e}^\mu \cdot \underline{\phi}(\underline{x}) = \sum_{j=1}^{m} \alpha_j^\mu K(\underline{x}_j, \underline{x})$$

which is independent of ϕ.
(depends only on K).

Hence:

the projection of the data onto the eigenvectors requires only knowing the kernel $K(\underline{x}_i, \underline{x}_j)$ (i.e. not knowing ϕ)

Knowledge of the kernel is used twice:
(1) to compute the $\{\alpha_j^\mu\}$,
(2) to compute the projections $\underline{e}^\mu \cdot \underline{\phi}(\underline{x})$. //