Spring 2014

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

(2)

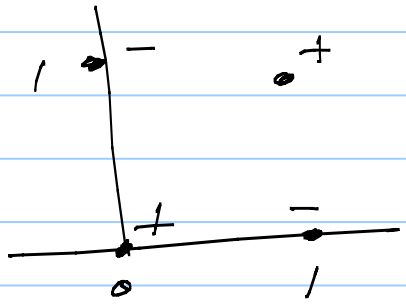
Why does this help?

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.

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



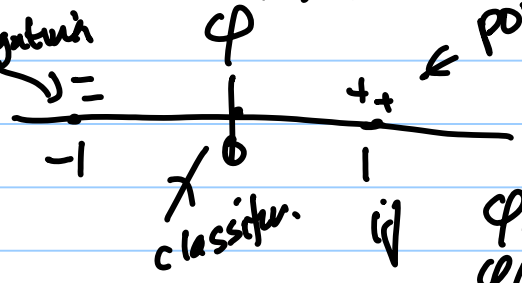
Impossible to find a linear classifier to do this.

But define a feature $\varphi(x_1, x_2) = x_1 x_2$.

Then we can classify the data by a linear classifier (i.e. a point) in 1-dimension

$x_1 x_2 = 1$ for +ve's

$x_1 x_2 = -1$ for −ve's



$|x-1| = -1$
$-|x| = -1$

$|x| = 1$
$-|x-1| = 1$

if $\varphi(x_1, x_2) > 0$, +ve
$\varphi(x_1, x_2) < 0$, −ve

(3) This example is "cheating" because we guess the right feature $x_1 x_2$.

More generally, we could set

$$\underline{\varphi}(x_1, x_2) = (x_1, x_2, x_1 x_2, x_1^2, x_2^2, x_1 x_2 x_3, \ldots)$$

i.e. we specify a higher-dimensional set of features (higher than $\dim(x_1, x_2) = 2$).

$$\underline{\lambda} = (\lambda_1, \lambda_2, \ldots)$$

Then $\underline{\lambda} \cdot \underline{\varphi}(\underline{x}) = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_{12} x_1 x_2 + \ldots$

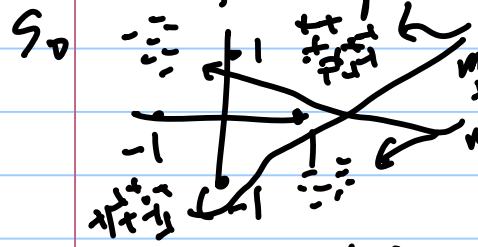
The SVM algorithm selects $\underline{\lambda}$ to separate the positive and negative examples - maybe by setting $\lambda_{12} = 1$, $\lambda_1 = \lambda_2 = \ldots = 0$

which gives us our "cheating" solution.

But there may be other equally good, or better, solutions $\hat{\underline{\lambda}}$.

Note: we need more examples in this case, because we need many more examples than free parameters (e.g. dimension of $\underline{\lambda}$)

So



Bottom-Line: By using a high-dimensional function $\underline{\varphi}(\underline{x})$ (i.e. so that $\dim \underline{\varphi} \gg \dim \underline{x}$) it is more likely that we can find a separating plane.

(4) 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 $L_d(\{\alpha_\mu\}) = \sum_\mu \alpha_\mu - \frac{1}{2} \sum_{\mu,\nu} \alpha_\mu \alpha_\nu y_\mu y_\nu \underline{\varphi}(\underline{x}_\mu) \cdot \underline{\varphi}(\underline{x}_\nu)$

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

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

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

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

Hence classifier is

$$\text{sign}\left\{ \sum_\mu \hat{\alpha}_\mu y_\mu K(\underline{x}, \underline{x}_\mu) + \hat{b} \right\}$$

only depends on $K(\cdot, \cdot)$

(5)

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.

(6) There are two important types of kernel.

(1) polynomial kernels

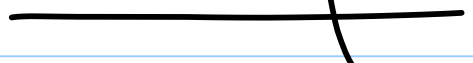
Eg $K(\underline{x}_1, \underline{x}_2) = \underline{x}_1 \cdot \underline{x}_2$ (seperating by plane/ previous lecture.)

or $K(\underline{x}_1, \underline{x}_2) = \underline{x}_1 \cdot \underline{x}_2 + |\underline{x}_1|^2 + |\underline{x}_2|^2 + (\underline{x}_1 \cdot \underline{x}_2)^2$.

These give boundaries like these

(2) Radial Basis Functions

$K(\underline{x}_1, \underline{x}_2) = e^{-|\underline{x}_1 - \underline{x}_2|^2}$



Note $K(\underline{x}_1, \underline{x}_2)$ decreases as $|\underline{x}_1 - \underline{x}_2|^2$ increases

This kernel behaves like nearest neighbor! Only nearby data points affect classification.

Recall classifier is of form:

$$\hat{y}(\underline{x}) = \text{sign}\left\{ \sum_{\mu} \hat{\alpha}_\mu y_\mu K(\underline{x}, \underline{x}_\mu) + \hat{b} \right\}$$

(The $\hat{\alpha}_\mu$'s are non-zero only for support vectors)

Because $K(\underline{x}, \underline{x}_\mu)$ decreases rapidly as $|\underline{x} - \underline{x}_\mu|$ increases, it follows that the estimate $\hat{y}(\underline{x})$ depends only on the datapoints near $\underline{x}$. It depends on the weighted sum of +ve and -ves (recall $\hat{\alpha}_\mu > 0$).

Note: Can also specify a set of kernels $K_1(\cdot,\cdot) \ldots K_m(\cdot,\cdot)$ parameters $(\alpha_1 \ldots \alpha_m)$, Kernel $= \sum_{i=1}^{m} \alpha_i K_i(\cdot,\cdot)$

use cross-validation to select the best (α's).

(7) Geometric Interpretation

The kernel can be used to specify a distance measure

Recall, Euclidean distance:

$$|\underline{x}_1 - \underline{x}_2|^2 = \underline{x}_1 \cdot \underline{x}_1 - 2\,\underline{x}_1 \cdot \underline{x}_2 + \underline{x}_2 \cdot \underline{x}_2$$

Similarly $|\underline{\varphi}(\underline{x}_1) - \underline{\varphi}(\underline{x}_2)|^2 = \underline{\varphi}(\underline{x}_1)\cdot\underline{\varphi}(\underline{x}_1) + \underline{\varphi}(\underline{x}_2)\cdot\underline{\varphi}(\underline{x}_2) - 2\,\underline{\varphi}(\underline{x}_1)\cdot\underline{\varphi}(\underline{x}_2)$

$$= K(\underline{x}_1, \underline{x}_1) + K(\underline{x}_2, \underline{x}_2) - 2K(\underline{x}_1, \underline{x}_2)$$

Euclidean distance in $\underline{\varphi}$-space

Special case, if $K(\underline{x}_1, \underline{x}_2) = \underline{x}_1 \cdot \underline{x}_2$, then we recover $|\underline{x}_1 - \underline{x}_2|^2$.

Bottom Line: think of $\underline{x} \to \underline{\varphi}(\underline{x})$ as a transformation on the geometry, which is the same as specifying a new distance $- K(\underline{x}_1, \underline{x}_1) + K(\underline{x}_2, \underline{x}_2) - 2K(\underline{x}_1, \underline{x}_2)$ - in the original $\underline{x}$ space.

Note: the RBF kernel has interesting geometry

$$K(\underline{x}_1, \underline{x}_1) + K(\underline{x}_2, \underline{x}_2) - 2K(\underline{x}_1, \underline{x}_2) = 2\left(1 - e^{-|\underline{x}_1 - \underline{x}_2|^2}\right)$$

So as $|\underline{x}_1 - \underline{x}_2|$ increases, the distance between datapoints in $\underline{\varphi}$-space tends to a maximum value of 2.

(8)

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 $\underline{\phi}(\underline{x})$?

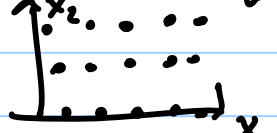
There are theoretical results → Mercer's Theorem.
Here we do two things:
(1) give some partial understanding, by showing that this relates to the Spectral Theorem in Linear Algebra.
(2) explaining why this result matters in practice, giving alternatives to SVM.

(part i)

Consider $K(\underline{x}, \underline{y})$

allow $\underline{x}, \underline{y}$ to take a finite set of possible values $\{\underline{z}_1, \underline{z}_2 \dots \underline{z}_N\}$ evenly spaced.

e.g. grid



Then consider $K(\underline{z}_i, \underline{z}_j) = K_{ij}$

This is an $N \times N$ matrix $\underline{\underline{K}}$ with positive terms ($K_{ij} \geq 0$)

By the spectral theorem (linear analysis) we can express it as $K_{ij} = \sum_{\mu=1}^{N} \lambda_\mu e^\mu_i e^\mu_j$

where λ, e are the eigenvalues & eigenvectors of $\underline{\underline{K}}$,

ie. $\sum_j K_{ij} e^\mu_j = \lambda^\mu e_i$

(4) The eigenvalues $\lambda^\mu > 0$ (because $K_{ij} > 0$ for all i,j)

so $K_{ij} = \sum_{\mu=1}^{N} (\lambda_\mu^{1/2} e_i^\mu)(\lambda_\nu^{1/2} e_j^\nu)$

i.e. $K(\underline{z}_i, \underline{z}_j) = \underline{\Phi}(\underline{z}_i) \cdot \underline{\Phi}(\underline{z}_j)$

where $\underline{\Phi}(\underline{z}_i) = (\lambda_1^{1/2} e_i^1, \lambda_2^{1/2} e_i^2, \ldots, \lambda_N^{1/2} e_i^N)$

This answers the question - when can we find the $\underline{\Phi}$s - if we restrict the $\underline{x}, \underline{y}$ to only take values $\underline{z}_1 \ldots \underline{z}_n$

Instead, we have to consider $K(\underline{x}, \underline{y})$ for continuous values of $\underline{x}$ and $\underline{y}$.

This requires - functional analysis, finding eigenfunctions and eigenvalues ← eigenfunction, ← eigenvalue

$$\int K(\underline{x}, \underline{y})\, \underline{e}(\underline{y})\, d\underline{y} = \lambda\, \underline{e}(\underline{x})$$

Functional analysis extends Linear Algebra to functions. Out of scope of this course. But Mercer's theorem is like a generalization of the spectral theorem to Functional Analysis

(part 2) Why does this matter? New Algorithms

Consider the spectral theorem again. Order the eigenvalues so that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$.

Then - depending on $K(\underline{z}_i, \underline{z}_j)$ - we may get a very good approximation to K by using only a few terms in the spectral theorem. Those with the biggest λ's

i.e. replace $K(\underline{z}_i, \underline{z}_j)$ by $\sum_{\mu=1}^{n} \lambda^\mu \underline{e}_i^\mu \underline{e}_j^\nu$ where $n \ll N$ (so that $|\lambda^\mu|$ is small, $\mu > n$)

This approximation can be very useful for some kernel methods. It means we need only evaluate $\sum_{\mu=1}^{n} (\lambda^\mu)^{1/2} \underline{e}_i^\mu$. (Full method requires Functional Analysis)