Chap.12 Kernel methods [Book, Chap.7]

Neural network methods became popular in the mid to late 1980s, but by the mid to late 1990s, kernel methods have also become popular in machine learning.

The first kernel methods were nonlinear classifiers called support vector machines (SVM), which were then generalized to nonlinear regression (support vector regression, SVR).

Kernel methods were further developed to nonlinearly generalize principal component analysis (PCA), canonical correlation analysis (CCA), etc. The kernel method has also been extended to probabilistic models, e.g. Gaussian processes (GP).

12.1 From neural networks to kernel methods [Book, Sect. 7.1]

Linear regression:

$$y_k = \sum_j w_{kj} x_j + b_k \,, \tag{1}$$

with **x** the predictors and **y** the predictands or response variables.

When a multi-layer perceptron (MLP) NN is used for nonlinear regression, the mapping does not proceed directly from **x** to **y**, but passses through an intermediate layer of variables **h**, i.e. the hidden neurons,

$$h_j = \tanh\left(\sum_i w_{ji} x_i + b_j\right) \,, \tag{2}$$

where the mapping from **x** to **h** is nonlinear through an activation function like the hyperbolic tangent.

The next stage is to map from **h** to **y**, and is most commonly done via a linear mapping

$$y_k = \sum_j \tilde{w}_{kj} h_j + \tilde{b}_k \, . \tag{3}$$

Since (3) is formally identical to (1), one can think of the NN model as first mapping nonlinearly from the input space to a “hidden” space (containing the hidden neurons), i.e. $\phi : \mathbf{x} \to \mathbf{h}$, then performing linear regression between the hidden space variables and the output variables.

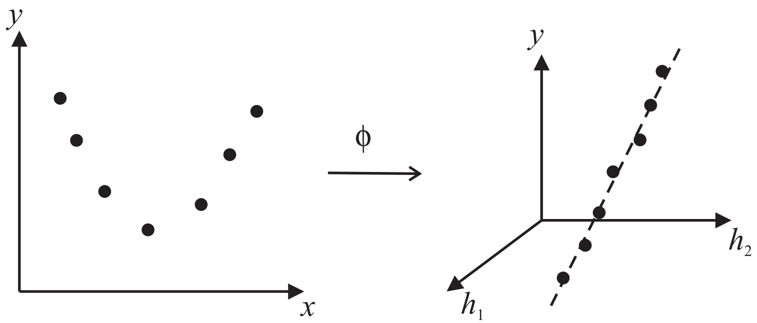

Figure : Schematic diagram illustrating the effect of the nonlinear mapping ϕ from the input space to the hidden space, where a nonlinear relation between the input x and the output y becomes a linear relation (dashed line) between the hidden variables $\mathbf{h}$ and y.

If the relation between $\mathbf{x}$ and $\mathbf{y}$ is highly nonlinear, then one must use more hidden neurons, i.e. increase the dimension of the hidden space, before one can find a good linear regression fit between the hidden space variables and the output variables.

As the parameters w_{ji}, $\tilde{w}_{kj}$, b_j and $\tilde{b}_k$ are solved together, nonlinear optimation is involved, leading to local minima, which is the main disadvantage of the MLP NN.

The kernel methods follow a somewhat similar procedure:
In the first stage, a nonlinear function ϕ maps from the input space to a hidden space, called the "feature" space.
In the second stage, one performs linear regression from the feature space to the output space.

Instead of linear regression, one can also perform linear classification, PCA, CCA, etc. during the second stage.

Like the radial basis function NN with non-adaptive basis functions (Book, Sect. 4.6) (and unlike the MLP NN with adaptive basis functions), the optimization in stage two of a kernel method is independent of stage 1, so only linear optimization is involved, and there are no local minima— a main advantage over the MLP.

However, the feature space can be of very high (or even infinite) dimension. This disadvantage is eliminated by the use of a kernel trick, which manages to avoid the direct evaluation of the high dimensional function ϕ altogether.
Hence many methods which were previously buried due to the "curse of dimensionality" (Book, p.97) have been revived in a kernel renaissance.

(Some kernel methods, e.g. Gaussian processes, use nonlinear optimization to find the hyperparameters, thus have the local minima problem.)

12.2 Advantages and disadvantages of kernel methods

[Book, Sect. 7.5]

Since the mathematical formulation of the kernel methods (Book, Sect. 7.2–7.4) is somewhat difficult, a summary of the main ideas of kernel methods and their advantages and disadvantages is given here.

First consider the simple linear regression problem with a single predictand variable y,

$$y = \sum_i a_i \, x_i + a_0 \,, \tag{4}$$

with x_i the predictor variables, and a_i and a_0 the regression parameters.

In the MLP NN approach to nonlinear regression, nonlinear adaptive basis functions h_j (also called hidden neurons) are introduced, so the linear regression is between y and h_j,

$$y = \sum_j a_j h_j(\mathbf{x}; \mathbf{w}) + a_0 \,, \tag{5}$$

with typically

$$h_j(\mathbf{x}; \mathbf{w}) = \tanh(\sum_i w_{ji} x_i + b_j) \,. \tag{6}$$

Since y depends on $\mathbf{w}$ nonlinearly (due to the nonlinear function tanh), the resulting optimization is nonlinear, with multiple minima in general.

What happens if instead of adaptive basis functions $h_j(\mathbf{x}; \mathbf{w})$, we use non-adaptive basis functions $\phi_j(\mathbf{x})$, i.e. the basis functions do not have adjustable parameters $\mathbf{w}$? In this situation,

$$y = \sum_j a_j \, \phi_j(\mathbf{x}) + a_0 \, . \tag{7}$$

e.g., Taylor series expansion with 2 predictors (x_1, x_2) would have $\{\phi_j\} = x_1, x_2, x_1^2, \; x_1 x_2, \; x_2^2, \ldots$. The advantage of using non-adaptive basis functions is that y does not depend on any parameter nonlinearly, so only linear optimization is involved with no multiple minima problem.

The disadvantage is the curse of dimensionality (Book, p.97), i.e. one generally needs a huge number of non-adaptive basis functions compared to relatively few adaptive basis functions to model a nonlinear relation, hence the dominance of MLP NN models despite the local minima problem.

The curse of dimensionality is finally solved with the kernel trick, i.e. although ϕ is a very high (or even infinite) dimensional vector function, as long as the solution of the problem can be formulated to involve only inner products like $\phi^{\mathrm{T}}(\mathbf{x}')\phi(\mathbf{x})$, then a kernel function K can be introduced

$$K(\mathbf{x}', \mathbf{x}) = \phi^{\mathrm{T}}(\mathbf{x}')\phi(\mathbf{x}). \tag{8}$$

The solution of the problem now only involves working with a very manageable kernel function $K(\mathbf{x}', \mathbf{x})$ instead of the unmanageable ϕ. From Book, Sect. 7.4, the kernel regression solution is of the form

$$y = \sum_{k=1}^{n} \alpha_k \, K(\mathbf{x}_k, \mathbf{x}) \,, \tag{9}$$

where there are $k = 1, \ldots, n$ data points $\mathbf{x}_k$ in the training dataset.

The most commonly used kernel is the *Gaussian kernel* or *radial basis function (RBF) kernel*,

$$K(\mathbf{x}', \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}' - \mathbf{x})\|^2}{2\sigma^2}\right), \tag{10}$$

If a Gaussian kernel function is used, then y is simply a linear combination of Gaussian functions.

In summary, with the kernel method, one can analyze the structure in a high-dimensional feature space with only moderate computational costs — as the kernel function gives the inner product in the feature space without having to evaluate the feature map ϕ directly.

The kernel method is applicable to all pattern analysis algorithms expressed only in terms of inner products of the inputs.

In the feature space, linear pattern analysis algorithms are applied, with no local minima problems since only linear optimization is involved.

Although only linear pattern analysis algorithms are used, the fully nonlinear patterns in the input space can be extracted due to the nonlinear feature map ϕ.

The main disadvantage of the kernel method is the lack of an easy way to inversely map from the feature space back to the input data space — a difficulty commonly referred to as the pre-image problem.

This problem arises e.g. in kernel principal component analysis (kernel PCA, see Book, Sect. 10.4), where one wants to find the pattern in the input space corresponding to a principal component in the feature space.

In kernel PCA, where PCA is performed in the feature space, the eigenvectors must lie in the span of the data points in the feature space, i.e.

$$\mathbf{v} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)\,. \tag{11}$$

As the feature space F is generally a much higher dimensional space than the input space X and the mapping function ϕ is nonlinear, one may not be able to find an $\mathbf{x}$ (i.e. the "pre-image") in the input space, such that $\phi(\mathbf{x}) = \mathbf{v}$.

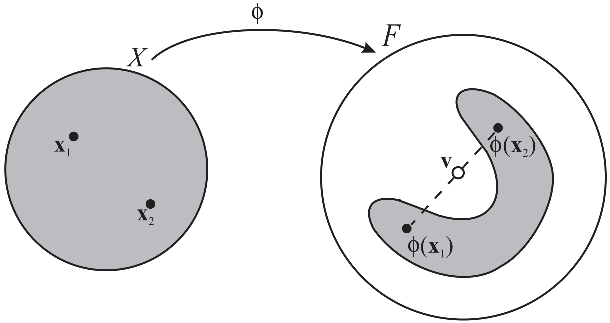

Figure : Schematic diagram illustrating the pre-image problem in kernel methods. The input space X is mapped by ϕ to the grey area in the much larger feature space F. Two data points $\mathbf{x}_1$ and $\mathbf{x}_2$ are mapped to $\phi(\mathbf{x}_1)$ and $\phi(\mathbf{x}_2)$, respectively, in F. Although $\mathbf{v}$ is a linear combination of $\phi(\mathbf{x}_1)$ and $\phi(\mathbf{x}_2)$, it lies outside the grey area in F, hence there is no "pre-image" $\mathbf{x}$ in X, such that $\phi(\mathbf{x}) = \mathbf{v}$.

12.3 Support vector regression (SVR) [Book, Sect. 9.1]

Support vector regression is a widely used kernel method for nonlinear regression.

In SVR, the objective function to be minimized is

$$J = C \sum_{n=1}^{N} E[y(\mathbf{x}_n) - y_{\mathrm{d}n}] + \frac{1}{2}\|\mathbf{w}\|^2, \tag{12}$$

where C is the inverse weight penalty parameter, E is an error function and the second term is the weight penalty term. To retain the sparseness property of the SVM classifier, E is usually taken to be of the form

$$E_\epsilon(z) = \begin{cases} |z| - \epsilon, & \text{if } |z| > \epsilon \\ 0, & \text{otherwise}. \end{cases} \tag{13}$$

This is an ϵ-insensitive error function, as it ignores errors of size smaller than ϵ.

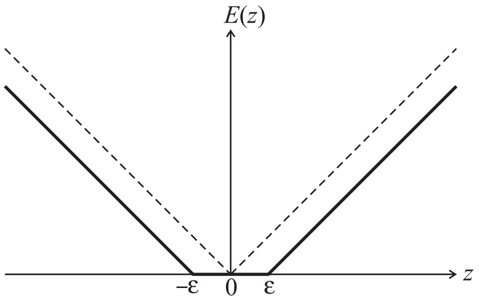

Figure : The ϵ-insensitive error function $E_\epsilon(z)$. Dashed line shows the mean absolute error (MAE) function.

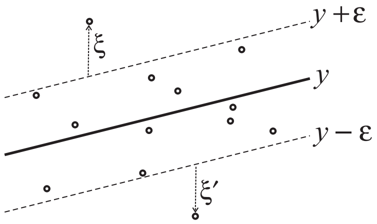

Figure : A schematic diagram of support vector regression (SVR). Data points lying within the "ϵ tube" (i.e. between $y - \epsilon$ and $y + \epsilon$) are ignored by the ϵ-insensitive error function, so these points offer no support for the final solution. Hence data points lying on or outside the tube are called "support vectors".

For data points lying above and below the tube, their distances from the tube are given by the "slack" variables ξ and ξ'. Data points lying inside (or right on) the tube have $\xi = 0 = \xi'$. Those lying above the tube have $\xi > 0$ and $\xi' = 0$, while those lying below have $\xi = 0$ and $\xi' > 0$.

The regression is performed in the feature space, i.e.

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0 \,, \tag{14}$$

where ϕ is the feature map.

With the kernel function $K(\mathbf{x}, \mathbf{x}') = \phi^{\mathrm{T}}(\mathbf{x})\phi(\mathbf{x}')$, and with a Lagrange multiplier approach, the solution can be written in the form

$$y(\mathbf{x}) = \sum_{n=1}^{N}(\lambda_n - \lambda'_n)K(\mathbf{x}, \mathbf{x}_n) + w_0 \,. \tag{15}$$

All data points within the "ϵ-tube" have the Lagrange multipliers $\lambda_n = \lambda'_n = 0$, so they fail to contribute to the summation in (15), thus leading to a sparse solution.

Support vectors are the data points which contribute in the above summation, i.e. either $\lambda_n \neq 0$ or $\lambda'_n \neq 0$, meaning that the data point must lie either outside the ϵ-tube or exactly at the boundary of the tube.

If K is the Gaussian or radial basis function (RBF) kernel (10), then (15) is simply a linear combination of radial basis functions centred at the training data points $\mathbf{x}_n$.

SVR with the Gaussian kernel can be viewed as an extension of the RBF NN method (Book, Sect. 4.6).
If the Gaussian or radial basis function (RBF) kernel (10) is used, then there are three hyperparameters in SVR, namely C, ϵ and σ (controlling the width of the Gaussian function in the kernel).

To find the optimal values of the hyperparameters, multiple models are trained for various values of the three hyperparameters, and upon evaluating their performance over validation data, the best estimates of the hyperparameters are obtained.

The simplest way to implement this is a 3-D grid search for the 3 hyperparameters, but this can be computationally expensive. One can also use e.g. evolutionary computation to find the optimal hyperparameters.

Inexpensive estimates of the hyperparameter values can be obtained following Cherkassky and Ma (2004).

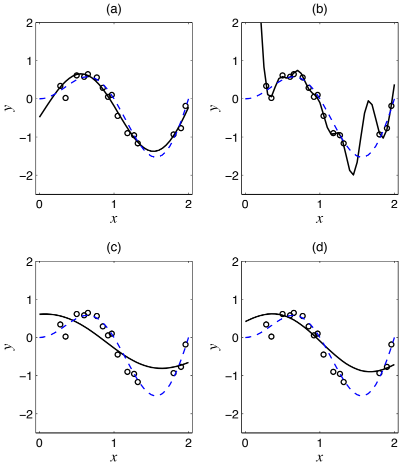

Figure : SVR applied to a test problem: (a) Optimal values of the hyperparameters C, ϵ and σ obtained from validation are used. The training data are the circles, the SVR solution is the solid curve and the true relation is the dashed curve. (b) A larger C (i.e. less weight penalty) and a smaller σ (i.e. narrower Gaussian functions) result in overfitting. Underfitting occurs when (c) a larger ϵ (wider ϵ-tube) or (d) a smaller C (larger weight penalty) is used.

The advantages of SVR over MLP NN are:
(a) For given values of the hyperparameters, SVR trains significantly faster since it only solves sets of linear equations, hence is better suited for high-dimensional datasets,
(b) SVR avoids the local minina problem from nonlinear optimization, and
(c) the ϵ-insensitive error norm used by SVR is more robust to outliers in the training data than the MSE.

However, the 3-D grid search for the optimal hyperparameters in SVR can be computationally expensive.

For nonlinear classification problems, the support vector machine (SVM) with Gaussian kernel has only 2 hyperparameters, namely C and σ (in contrast to SVR, where the 3 hyperparameters are C, σ and ϵ).

SVM functions in Matlab

SVM code for both classification and regression problems can be downloaded from LIBSVM (Chang and Lin, 2001). The code can be run on MATLAB, C++ and Java:
`www.csie.ntu.edu.tw/~cjlin/libsvm/`

MATLAB Bioinformatics toolbox has SVM for classification (not regression), called svmclassify:
`www.mathworks.com/help/toolbox/bioinfo/ref/svmclassify.html`

References:

Chang, C.-C. and Lin, C.-J. (2001). LIBSVM: A library for support vector machines. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Cherkassky, V. and Ma, Y. Q. (2004). Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Networks*, 17(1):113–26.