Nonlinear Expansion

— Lecture Notes —

Laurenz Wiskott
Institut für Neuroinformatik
Ruhr-Universität Bochum, Germany, EU

28 January 2017

— Summary —

1 Nonlinear expansion can be used to turn an algorithm that finds an optimal linear function into one that finds an optimal nonlinear function within a given function space, simply by first expanding the data nonlinearly and then applying the linear algorithm. → Exercises, Solutions

Contents

1 Nonlinear expansion

Nonlinear expansion (D: nichtlineare Expansion) of the input data is a simple trick to generalize a linear optimization method to nonlinear functions. Thus, **assume we have solved an optimization problem already in the linear case**, i.e. we have a method for finding the optimal coefficients a_i of a linear function

$$\blacklozenge \quad g_l(\mathbf{x}) := \sum_i a_i x_i \tag{1.1}$$

Core text and formulas are set in dark red, one can repeat the lecture notes quickly by just reading these; ♦ marks important formulas or items worth remembering and learning for an exam; ◊ marks less important formulas or items that I would usually also present in a lecture; + marks sections that I would usually skip in a lecture.
More teaching material is available at https://www.ini.rub.de/PEOPLE/wiskott/Teaching/Material/.

applied to a training set of input data $\mathbf{x}^\mu = (x_1^\mu, ..., x_I^\mu)^T$, with μ indicating the different samples and T indicating the transpose to make $\mathbf{x}^\mu$ a proper column vector. **Linear regression, for instance,** finds the coefficients that minimize the mean squared distance between the output values $y^\mu := g_l(\mathbf{x}^\mu)$ and some target values t^μ for the input values $\mathbf{x}^\mu$.

Note that **(1.1) does not include a constant,** which is often desired. One could define $g_l(\mathbf{x}) := a_0 + \sum_i a_i x_i$, but that would make the function affine and complicate the notation. However, **one can assimilate a constant term by assuming that every input vector has an additional component $x_0 = 1$ and then writing**

$$\blacklozenge \quad g_l(\mathbf{x}) := a_0 + \sum_{i=1}^{N} a_i x_i = \sum_{i=0}^{N} a_i x_i \,, \tag{1.2}$$

which is formally identical to (1.1).

To generalize such a linear optimization method to nonlinear functions we first derive a new set of input patterns $\mathbf{z}^\mu$ with

$$\blacklozenge \quad \mathbf{z} := \mathbf{\Phi}(\mathbf{x}) \,, \tag{1.3}$$

where $\mathbf{\Phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x}))^T$ is a fixed nonlinear vectorial function and ϕ_k indicates nonlinear scalar functions. The space into which the input vectors $\mathbf{x}$ are projected by the functions ϕ_k is often referred to as *feature space* (D: Merkmalsraum) and it usually has a much higher dimensionality than the of the original input space. **Then we simply apply the linear optimization method to the new set of input patterns $\mathbf{z}^\mu$ to obtain**

$$\blacklozenge \quad g_l(\mathbf{z}) \stackrel{(1.1)}{=} \sum_k a_k z_k \tag{1.4}$$

$$\blacklozenge \quad \stackrel{(1.3)}{=} g_l(\mathbf{\Phi}(\mathbf{x})) = \sum_k a_k \phi_k(\mathbf{x}) \,. \tag{1.5}$$

While $g_l(\mathbf{z})$ is linear in $\mathbf{z}$ it is actually nonlinear in $\mathbf{x}$, **which means we have found an optimal nonlinear function within the function space spanned by the fixed nonlinear functions ϕ_k.**

For example, **for a two-dimensional input $\mathbf{x} = (x_1, x_2)^T$ one could choose**

$$\blacklozenge \quad z_0 = \phi_0(\mathbf{x}) := 1 \,, \tag{1.6}$$

$$\blacklozenge \quad z_1 = \phi_1(\mathbf{x}) := x_1 \,, \quad z_2 = \phi_2(\mathbf{x}) := x_2 \,, \tag{1.7}$$

$$\blacklozenge \quad z_3 = \phi_3(\mathbf{x}) := x_1^2 \,, \quad z_4 = \phi_4(\mathbf{x}) := x_1 x_2 \,, \quad z_5 = \phi_5(\mathbf{x}) := x_2^2 \,, \tag{1.8}$$

and obtain

$$\blacklozenge \quad g_l(\mathbf{\Phi}(\mathbf{x})) \stackrel{(1.5)}{=} \sum_k a_k \phi_k(\mathbf{x}) \tag{1.9}$$

$$\blacklozenge \quad \stackrel{(1.6\text{–}1.8)}{=} a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_1 x_2 + a_5 x_2^2 \,, \tag{1.10}$$

which is the optimal quadratic function, see Fig. 1.1.

Note, that **there is some arbitrariness in the definition of the nonlinear functions ϕ_k.** In the example above (1.6–1.8), for instance, one could define $\phi_3(\mathbf{x}) := 100x_1^2$ or $\phi_3(\mathbf{x}) := x_1^2 + x_2^2$ instead of $\phi_3(\mathbf{x}) := x_1^2$. The space spanned by the functions ϕ_k would be still the same, but the data distribution would be distorted differently depending on the definition of ϕ_3. You have to be aware of this arbitrariness if you apply the trick of nonlinear expansion to linear algorithms that are sensitive to such distortions, like principal component analysis.

Parameterized nonlinearities, such as $g(x) = a \sin(kx + \phi)$ with $k \in \mathbb{R}$, **cannot be optimized this way,** since they do not define a finite dimensional vector space of functions.

In neural network theory *radial basis functions* (D: radiale Basisfunktionen) are often used for nonlinear expansion (Bishop, 1995, sec. 5).

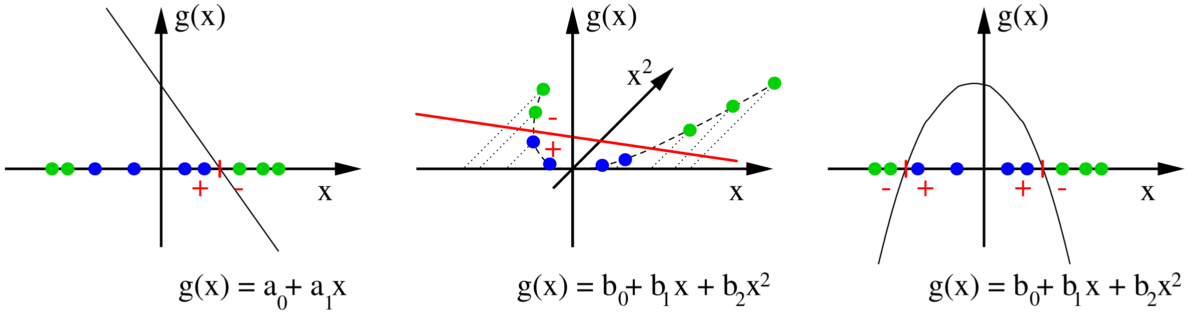

Figure 1.1: This example illustrates the effect of nonlinear expansion. The blue and green dots on the x-axis are not linearly separable (left). However, if one expands the one-dimensional input space x to the two-dimensional space of x and x^2, the problem can be solved with a linear classifier (middle). This linear function mapped back to the original input space becomes a quadratic function solving the problem (right).

References

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK.