GHENT
UNIVERSITY

Lecture Notes

E016350: Artificial Intelligence

Learning with Nonlinear Features

Aleksandra Pizurica

Spring 2024

Contents

Disclaimer*: These lecture notes were written by Prof. Aleksandra Pizurica to accompany the slides of the course E016350: Artificial Intelligence, facilitating their understanding. The lecture notes are not meant to be self-contained, and do not cover all the study material in the course. They are by no means meant to replace the recommended textbook and do not necessarily cover all the relevant aspects that are presented in the slides and explained in the lectures. Some sections are adapted from the book of S. Russel and P. Norvig: Artificial Intelligence: A Modern Approach.*

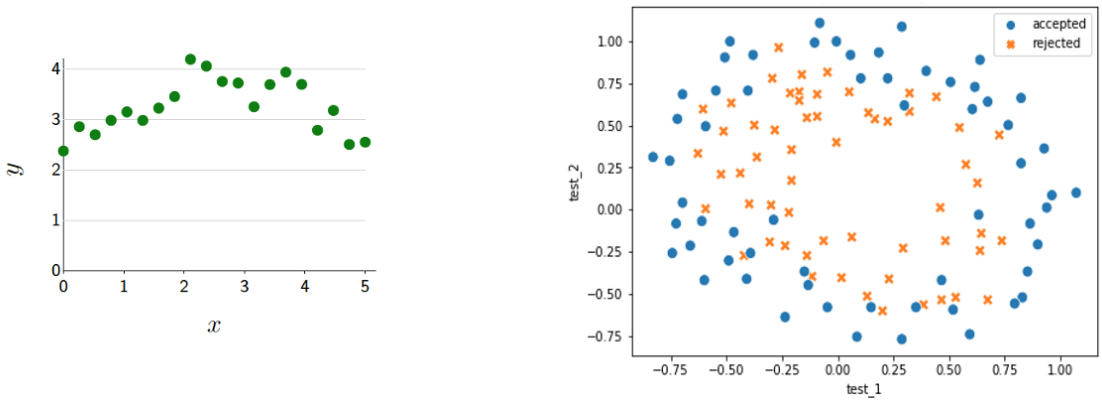


Figure 1: Examples of more complex data where a non-linear predictor is needed for regression (left) or classification (right). Figures from [1].

1 Linear predictors with nonlinear features

So far we were dealing with linear regression and linear classification. However, in real life data are often more complex and a linear predictor may not be a satisfactory fit (see examples in Fig. 1). In this case, we can turn to more advanced models like decision trees and neural networks (that we will study next). Before doing so, let's see how we can tackle these tasks still with the machinery of linear predictors but then feeding them with nonlinear features. You will see that in some cases this can work pretty well!

The main idea is to extract a vector of **nonlinear features** $\phi(\mathbf{x}) \in \mathbb{R}^n$ from the input $\mathbf{x} \in \mathbb{R}^d$ and to feed these nonlinear features to a linear predictor. The prediction will be non-linear in $\mathbf{x}$! With appropriately selected nonlinear features we can fit the data as illustrated in Fig. 2.

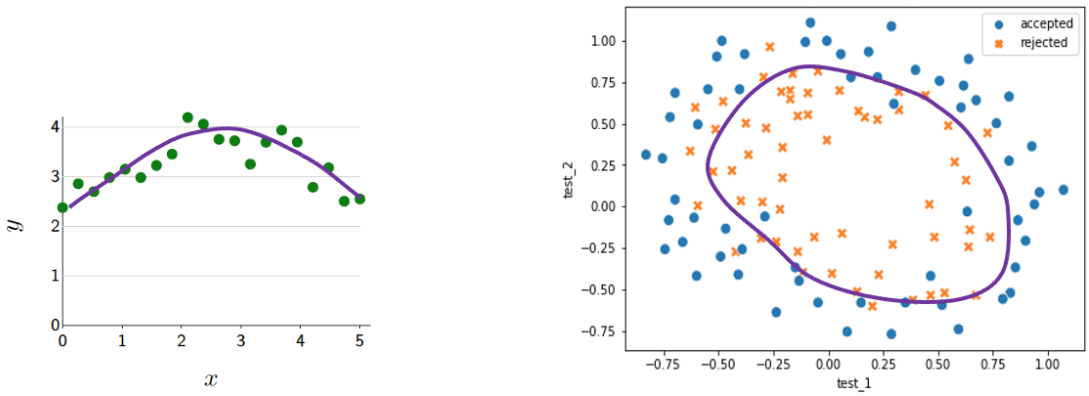
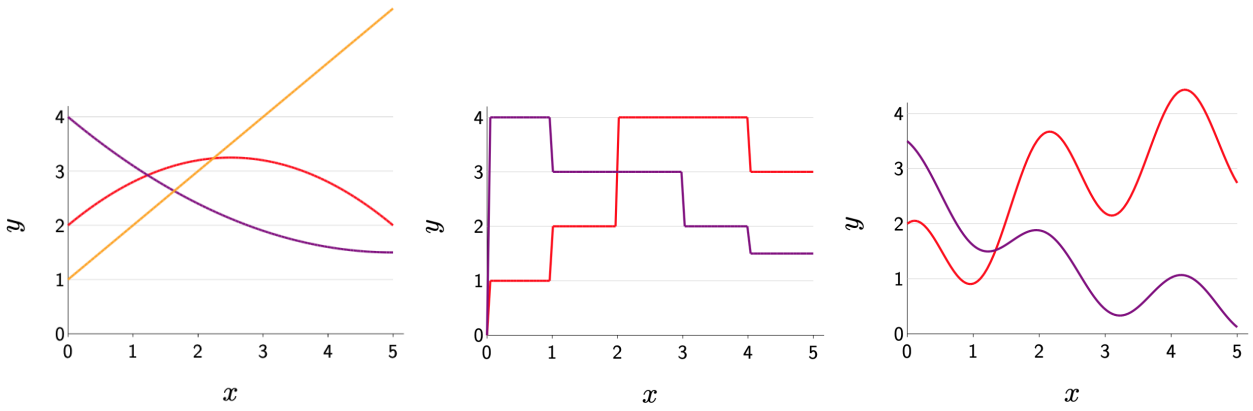


Figure 2: By extracting nonlinear features $\phi(\mathbf{x})$ from the input $\mathbf{x}$ and feeding those to linear regression as $h_{\mathbf{w}}(\mathbf{x}) = \phi(\mathbf{x}) \cdot \mathbf{w}$ or to logistic regression as $h_{\mathbf{w}}(\mathbf{x}) = Logistic(\phi(\mathbf{x}) \cdot \mathbf{w})$, we obtain predictions that are nonlinear in $\mathbf{x}$. Illustrations from [1].



(a) (b) (c)

Figure 3: Examples of predictors with (a) quadratic features; (b) piece-wise constant features and (c) features with periodicity structure. Illustrations from [1].

1.1 Linear regression machinery with nonlinear features

We generalize linear regression $\mathbf{x} \cdot \mathbf{w}$ by replacing the "raw" input $\mathbf{x}$ by some feature vector $\phi(\mathbf{x})$. The resulting predictor is

$$h_{\mathbf{w}}(\mathbf{x}) = \phi(\mathbf{x}) \cdot \mathbf{w} \tag{1}$$

The feature vector $\phi(\mathbf{x})$ can be arbitrary. We will illustrate the use of nonlinear features for univariate regression only, i.e., for the case where the input is scalar x from which we will construct a n-dimensional feature vector $\phi(x)$. Fig. 3 illustrates three classes of nonlinear predictors that are obtained with different feature vectors.

Note that with $\phi(x) = [1, x]^\top$ the predictor in Eq (1) would simply be univariate linear regression (the dummy variable $x_0 = 1$ allows us to include the intercept term w_0 in the vector $\mathbf{w}$). Now, if we construct a nonlinear feature vector by adding a quadratic term x^2:

$$\phi(x) = [1, x, x^2]^\top$$

we obtain **quadratic predictors** illustrated in Fig. 2(a). The different curves there correspond to different weight vectors $\mathbf{w}$. The yellow line corresponds to $\mathbf{w} = [1, 1, 0]^\top$ which sets the quadratic term to zero, thus the predictor reduces to the linear case. The predictor shown with the red line corresponds to $\mathbf{w} = [2, 1, -0.2]^\top$ and the one in purple to $\mathbf{w} = [4, -1, 0.1]^\top$.

The **piecewise constant** predictors in Fig. 2(b) are obtained with feature extractors that divide the input space into regions and allow the predicted value of each region to vary independently. Specifically, each component of the feature vector corresponds to one region (e.g., (0, 1] or (1, 2], etc.) and is 1 if x lies in that region and 0 otherwise:

$$\phi(x) = [\mathbf{1}[0 < x \leq 1], \mathbf{1}[1 < x \leq 2], \mathbf{1}[2 < x \leq 3], \mathbf{1}[3 < x \leq 4], \mathbf{1}[4 < x \leq 5]]^\top$$

Assuming the regions are disjoint, the weight associated with a component/region is exactly the predicted value. E.g., in Fig. 2(b), the predictor shown in red corresponds to $\mathbf{w} = [1, 2, 4, 4, 3]^\top$ and the one in purple to $\mathbf{w} = [4, 3, 3, 2, 1.5]^\top$. As we make the regions smaller, we get more features, and the expressiveness of our hypothesis class increases. In the limit, we can essentially capture any predictor we want. This sounds awesome but there are some practical considerations and limitations especially when the input is not scalar, we reflect on this in Section 1.3.

Fig. 2(c) shows yet another family of the predictors, these have some **periodicity structure**. In particular, these were obtained with the feature vector

$$\phi(x) = [1, x, x^2, \cos(3x)]^\top$$

The line in red corresponds to $\mathbf{w} = [1, 1, -0.1, 1]^\top$ and the one in purple to $\mathbf{w} = [3, -1, 0.1, 0.5]^\top$.

For all these predictors, the hypothesis space can be expressed as

$$\mathcal{H} = \{h_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x) : \mathbf{w} \in \mathbb{R}^n\}$$

where $\phi(x)$ is the particular feature vector and n its dimension (for our first example, with the quadratic predictor, $n = 3$, in the second example we had $n = 5$, and in the third one $n = 4$).

We showed three examples but there is an unboundedly large design space of possible feature extractors. In practice, the choice of features is informed by the prediction task that we wish to solve (either prior knowledge or preliminary data exploration) [1].

1.2 Classification with nonlinear features

Similarly as we did with regression, we can feed the "machinery" of linear classifiers with nonlinear features $\phi(\mathbf{x})$ of the input $\mathbf{x}$. The classification will be linear in $\phi(\mathbf{x})$ but nonlinear in $\mathbf{x}$.

For example, to obtain a circular decision boundary we extend the input $\mathbf{x} = [x_1, x_2]^\top$ with the quadratic term $x_1^2 + x_2^2$ and we feed this nonlinear feature vector

$$\phi(\mathbf{x}) = [x_1, x_2, x_1^2 + x_2^2]^\top$$

to a linear binary classifier, say with a hard threshold:

$$h_{\mathbf{w}}(\mathbf{x}) = Threshold(\mathbf{w} \cdot \phi(\mathbf{x})) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \phi(\mathbf{x}) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Fig. 4 shows the resulting decision boundary for the case where the weights are $\mathbf{w} = [2, 2, -1]^\top$. Indeed, it is easy to verify that with the given feature vector and with this weight vector the predictor in Eq (2) can be rewritten as

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & (x_1 - 1)^2 + (x_2 - 1)^2 \leq 2 \\ 0 & \text{otherwise} \end{cases}$$

As a sanity check, we you can see that $\mathbf{x} = [0, 0]^\top$ results in a score of 0, which means that it is on the decision boundary. And as either of x_1 or x_2 grow in magnitude (either $|x_1| \to \infty$ or $|x_2| \to \infty$), the contribution of the third feature dominates and because $w_3 = -1$ the score $\mathbf{w} \cdot \phi(\mathbf{x})$ will be negative and thus the predicted label will be 0.

The same principle remains when we replace the hard classifier with a soft one, like in logistic regression, now with nonlinear features:

$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \phi(\mathbf{x})}}$$

By constructing more complex feature vectors $\phi($**x**, we can obtain also more complex decision boundaries.

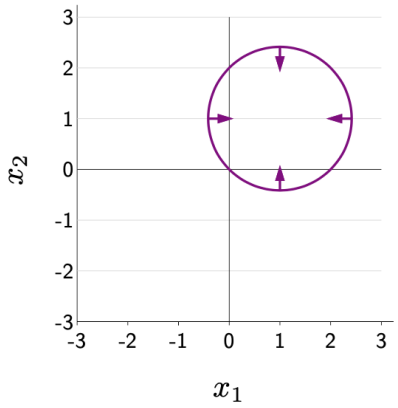


Figure 4: A circular decision boundary obtained with with a linear classifier acting on the nonlinear feature vector $\phi(\mathbf{x}) = [x_1, x_2, x_1^2 + x_2^2]$ with the weight vector $\mathbf{w} = [2, 2, -1]^\top$. Example from [1].

1.3 Limitations and practical considerations

We have seen that by constructing appropriate feature vectors we can solve – in principle – any nonlinear regression or classification task with the machinery of linear predictors. The components of the feature vector, i.e., the individual features, represent what properties might be useful for prediction. If a feature is not useful, then the learning algorithm can assign a weight close to zero to that feature. Of course, the more features one has, the harder learning becomes. This poses some fundamental limitations to how far we can go with the approach described in this Section.

For example, in Section 1.1, we constructed a quadratic predictor for the case where the input was scalar $x \in \mathbb{R}$. This was inexpensive, we needed to add just one component x^2 to construct the desired nonlinear feature vector. But if the input were d-dimensional $\mathbf{x} \in \mathbb{R}^d$, then there would be $O(d^2)$ quadratic features of the form $x_i x_j$ for $i, j \in \{1, \ldots, d\}$. When d is large, then d^2 can be prohibitively large, which is one reason that using the machinery of linear predictors to increase expressiveness can be problematic [1]. Similarly, take the example of the piece-wise predictors from Section 1.1 where the feature vector is an indicator function over the successive intervals. Think what happens if x were not a scalar, but a d-dimensional vector. Then if we want to increase the expressiveness of the model by breaking up each component of the feature vector (each interval) into B bins there will be Bd features! For each feature, we need to fit its weight, and there will in generally be too few examples to fit all the features. So, we will need to study other mechanisms for nonlinear learning on high-dimensional inputs.

References

[1] M. Charikar and S. Koyejo. *Artificial Intelligence: Principles and Techniques (CS221).* Stanford University, 2024.