CSE517A Machine Learning **Spring 2020**

Lecture 8: Radial Basis Function Networks

Instructor: Marion Neumann *Scribe: Jingyu Xin*

Reading: **Reading**: LFD eCh6.3-6.3.2 (RBF Networks)

1 Introduction

Main focus so far: linear models. What if the decision boundary should be non-linear like in Figure 1?

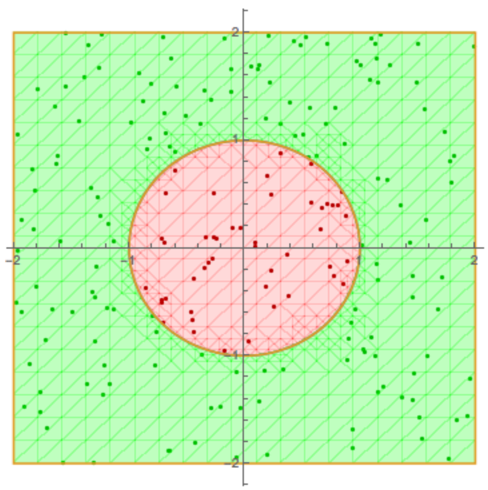


Figure 1: Example of non-linear decision boundary

Question: Do you know a non-linear machine learning model that can handle such data?

Recap: k-Nearest Neighbors

- Classification: $h(\mathbf{x}) = sign(\sum_{i=1}^{k} y_i)$
- Regression: $h(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} y_i$
- k needs to be fixed

Thought: What if we use all n training data points and a weighting scheme, such that data points further away contribute less to the prediction?

2 Models using Radial Basis Functions

Use a *radial basis function* (RBF) to quantify the contribution of each training data point with respect to its **distance to the test point**. We define an RBF as $g(z)$ with $z = \frac{||\mathbf{x}-\mathbf{x}'||}{r}$, where the scale parameter r regulates the weighting.

Examples:

- Gaussian kernel: $g(z) = e^{-\frac{1}{2}z^2}$

- Window kernel: $g(z) = \begin{cases} 1 & z \leq 1 \\ 0 & z > 1 \end{cases}$

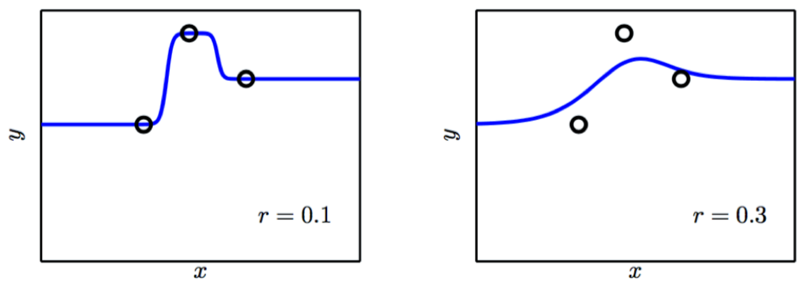
Note, RBFs are a special kind of *kernel* function $\mathrm{k}(\mathbf{x}, \mathbf{x}') = g(z)$. They are *stationary kernels*, since z is a function of the distance $||\mathbf{x} - \mathbf{x}'||$ and hence $g(z)$ is invariant to translations in the input space. The scale parameter r is also called the *kernel width*.

2.1 A Prediction Model: Kernel Regression

Use a weighted sum of the y-values:

$$h(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathrm{k}(\mathbf{x}, \mathbf{x}_i) y_i}{\sum_{i=1}^{n} \mathrm{k}(\mathbf{x}, \mathbf{x}_i)}. \tag{1}$$

$\Rightarrow$ **non-parametric** model (using one *bump* at the test point $\mathbf{x}$ – requiring all training data at test time for computation). This model is also known as the "Nadaraya-Watson" model. Figure 2 (from [LFD CH6.3][1]). illustrates this model and the impact of the widths parameter r. Note that they call this the "*non-parametric version* of the RBF network".



(a) Nonparametric RBF, small r (b) Nonparametric RBF, large r

Figure 2: Illustration of non-parametric RBF with different r [LFD CH6.3].

Now, Eq. (1) becomes equivalent to

$$h(\mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x}) \mathrm{k}(\mathbf{x}, \mathbf{x}_i) \tag{2}$$

with $w_i(\mathbf{x}) = \frac{y_i}{\sum_{i=1}^{n} \mathrm{k}(\mathbf{x}, \mathbf{x}_i)}$.

Interpretation: center a bump at *every* $\mathbf{x}_i$ with **height** $w_i(\mathbf{x})$, where the **width** is determined by r.

Problem: $w_i(\mathbf{x})$ depends on $\mathbf{x}$ (test point).

2.2 A Simplified Prediction Model

Simplification: Fix heights for all test points to w_i.

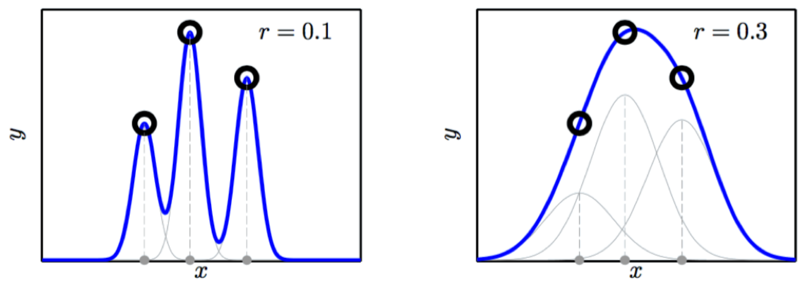
Now,

$$h(\mathbf{x}) = \sum_{i=1}^{n} w_i \, \mathrm{k}(\mathbf{x}, \mathbf{x}_i) = \mathbf{w}^\top \mathbf{z} \quad \text{with} \quad z = \begin{bmatrix} \mathrm{k}(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ \mathrm{k}(\mathbf{x}, \mathbf{x}_n) \end{bmatrix} \tag{3}$$

[1] [LFD CH6.3] Learning From Data, Abu-Mostafa et al., 2012, AMLBook

and $\mathbf{w} = [w_1, w_2, ..., w_n]^\top$ unknown constants (**parameters**).

$\Rightarrow$ **non-parametric** model (using one *bump* at each training point). This model looks like a *parametric* model, but it has n parameters (the number of parameters grows with the size of the training data). Figure 3 (from [LFD CH6.3]) illustrates this model and the impact of the widths parameter r. Note that they call it the "*parametric version* of the RBF network".



(c) Parametric RBF, small r (b) Parametric RBF, large r

Figure 3: Illustration of parametric RBF with different r [LFD CH6.3].

Training: fit w_i by minimizing the training error using $\tilde{D} = \{(\mathbf{z}_i, y_i)\}_{1,...,n}$, where $\mathbf{z}_i = \begin{bmatrix} \mathrm{k}(\mathbf{x}_i, \mathbf{x}_1) \\ \vdots \\ \mathrm{k}(\mathbf{x}_i, \mathbf{x}_n) \end{bmatrix}$

Big Surprise: Eq. (3) is a **linear model**!

We have transformed the d-dimensional non-linear model to an n-dimensional linear model! We essentially created a *feature space transformation* from $\mathbf{x}_i \in \mathbb{R}^d$ to $\mathbf{z}_i \in \mathbb{R}^n$. The non-linear transformation is defined by the kernel k, and the training data points $\mathbf{x}_i$.

2.3 Radial Basis Function Networks

Now, when training the model, choosing n parameters will lead to **overfitting** as we can fit n parameters exactly with n data points. So, for our final model, the RBF network, we choose $k << n$ weights/bumps (not centered on the training data points):

$$h(\mathbf{x}) = w_0 + \sum_{j=1}^{k} w_j \mathrm{k}(\mathbf{x}, \boldsymbol{\mu}_j) = \mathbf{w}^\top \mathbf{z} \quad \text{with} \quad \mathbf{z} = \begin{bmatrix} \mathrm{k}(\mathbf{x}, \boldsymbol{\mu}_1) \\ \vdots \\ \mathrm{k}(\mathbf{x}, \boldsymbol{\mu}_k) \end{bmatrix} \tag{4}$$

and w_0 is the bias term, which is needed if the y-values have non-zero mean.

Note: This is a regression model, for classification pass output through a sigmoid function ($sigm(a) = \frac{e^a}{1+e^a}$).

Training the RBF-Network

Given k and r, we need to learn the parameters $w_1, ..., w_k$ and $\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k$.

Strategy:

(1) Choose $\boldsymbol{\mu}_j$ to represent $\mathbf{x}_i$ (ignore y_i)

(2) Compute $\mathbf{z}_i$ for all $\mathbf{x}_i$ in D

(3) Fit a linear model $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{z}$ using $\tilde{D} = \{(\mathbf{z}_i, y_i)\}_{i=1}^n$

(2) and (3) are straightforward: use any linear classifier, gradient descent or analytic solution. (1) is an **unsupervised** learning problem.

Simple solution: choose k partitions of equal sizes and use the average data point as $\boldsymbol{\mu}_j$.

More sophisticated solution: use k-means to determine the partitions (*covered later in this course*). Note that the k-center problem is NP-hard, therefore we need an efficient approximate solution such as the iterative k-means algorithm.

Exercise 2.1. True or false? Justify your answer.

(a) RBF networks (all three versions) can only be used for regression problems.

(b) A non-parametric ML model does not have any parameters.

(c) For the simplified prediction model ("non-parametric RBF network") we need to model the bias term.

Summary

- models using RBFs are a natural extension to k-NN / soft version of k-NN.
- RBF networks were developed for smooth function interpolation (regression).
- You can extend RBF network to have differently shaped bumps at each $\boldsymbol{\mu}_j$ (need $r_j's$ instead of one r)
- Both kernel regression and the simplified prediction model are **non-parametric** ML methods.
- If we fix k independent of n then the resulting RBF network as defined in Eq. (4) is a **parametric** or **semi-parametric** ML method.
- All three models, kernel regression, the simplified prediction model, and RBF networks are *kernel methods* and hence are non-linear models.

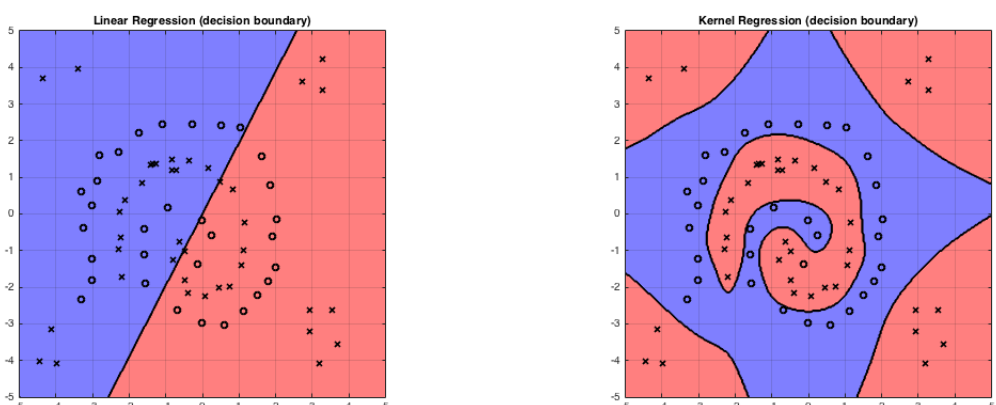
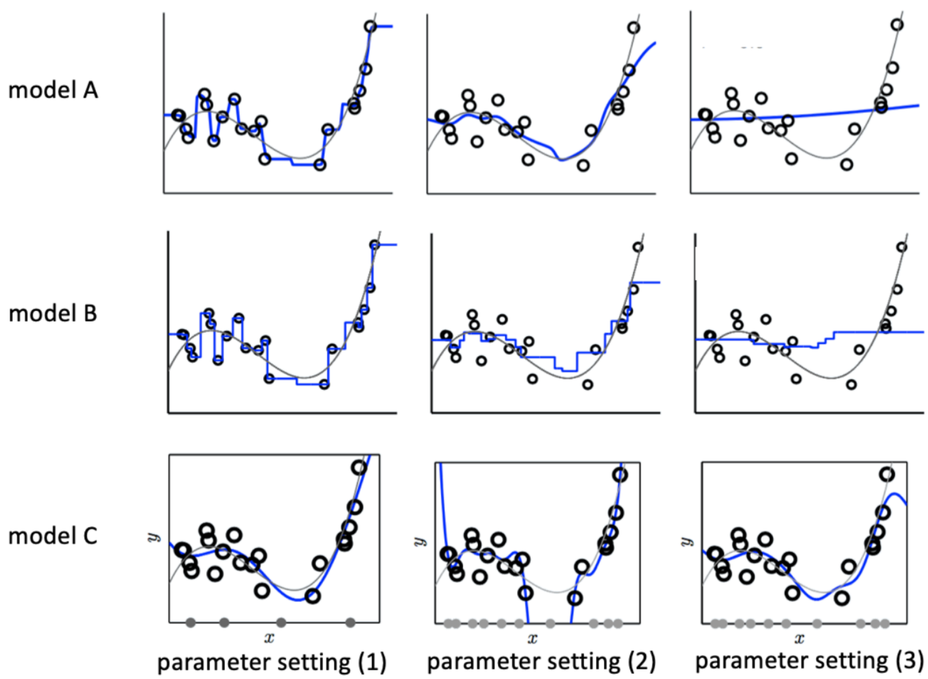


Figure 4: Decision boundaries of linear model and kernel regression for spiral data

Exercise 2.2. Consider the following three models (MODEL A, MODEL B, MODEL C with two different parameter settings each) for a 1-dimensional regression problem $y = f(x) + \epsilon$ with $f : \mathbb{R} \to \mathbb{R}$.



(a) What are the three different models?

MODEL A:

MODEL B:

MODEL C:

(b) What are the three different parameter settings?

SETTING (1):

SETTING (2):

SETTING (3):

(c) Consider all three models with parameter setting (1). What does MODEL C implement/achieve?

(d) Argue, whether MODEL C is a *parametric* ML model.