Neural Network-Based Nonlinear Regression on a Torus

Johannes Traa

Abstract

Linear regression is a well-studied problem that can be generalized to the non-linear setting via neural network architectures. However, certain forms of regression involving non-Euclidean feature spaces are not very well explored. For example, we may want to predict one or more circular variables from one or more linear variables. This is important when directional information (e.g. wind, phase, azimuth) depends on a value that lies in $\mathbb{R}^N$ (e.g. location, velocity, frequency).

Existing methods involve fitting a wrapped line (a.k.a. barber-pole regression curve [1]) or wrapped cubic spline to a circular-linear dataset via the Expectation-Maximization (EM) algorithm. In this paper, we will explore a conditional density neural network architecture to solve the regression problem on a torus. Empirical results suggest that this may be the most effective model due to its expressiveness, but the least so in practical terms due to the significant computational burden of learning the network weights.

I. Introduction

Linear regression is a well-known problem. It involves predicting one set of real-valued quantities from another such set. This can be expressed as a least-squares problem defined by a system of linear equations [2]. However, complications arise when one or more of the variables lie on a different sample space. In this paper, we are interested in predicting multiple circular variables from a linear one. This arises, for example, when we would like to discover structure in an inter-channel phase difference (IPD) [3] feature set for the purpose of source separation or localization.

IPD features are extracted by calculating the difference in phase of corresponding DFT coefficients of the signals incident on a pair of microphones. The structure of the features as a function of frequency depends strongly on the relative positions of the microphone array and sound source(s) as well as the reverberant qualities of the room. For non-reverberant rooms, the data forms straight lines that are wrapped in the interval $[-\pi, \pi]$. However, in the presence of reverberation, the relationship between frequency and phase difference is non-linear.

We will show that methods based on the Expectation-Maximization (EM) [4] algorithm can be applied to learn linear and non-linear models. Their main drawback is a computational complexity that increases exponentially with the number of circular variables. We will show how a neural network can be used to avoid such complications.

II. Directional Statistics

The directional statistics literature [5] provides a number of useful theoretical tools for the circular regression task. We will define the sample space of interest and discuss two probability distributions that are useful for modeling wrapped variables.

A. Unit Circle

First, we define the unit circle as the feature space consisting of all unit vectors $\mathbf{x}$ as:

$$\mathbb{S}^1 = \left\{ \mathbf{x} \,:\, \|\mathbf{x}\|_2 = 1 \,,\, \mathbf{x} \in \mathbb{R}^2 \right\} \quad . \tag{1}$$

We will find it more convenient to consider the one-dimensional embedding of $\mathbb{S}^1$ directly in the interval $[-\pi, \pi]$. In other words, we can associate each unit vector with its argument to write:

$$\mathbb{S}^1 = \left\{ \theta \,:\, -\pi \le \theta \le \pi \,,\, \theta \in \mathbb{R}^1 \right\} \quad . \tag{2}$$

Higher-dimensional generalizations follow intuitively from this. For example, we can describe the unit torus as the product space of two unit circles: $\mathbf{S}^2 = \mathbf{S}^1 \times \mathbf{S}^1$.

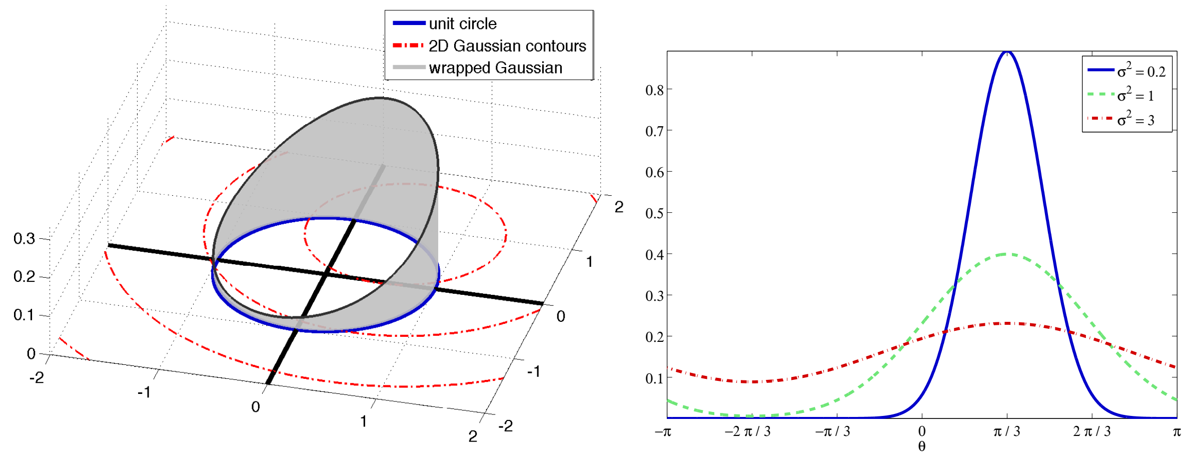

Fig. 1. (Left) Wrapped Gausian pdf ($\mu = \frac{\pi}{3}$) on the unit circle in $\mathbb{R}^2$ shown with 2D Gaussian contours ($\sigma^2 = 0.8$). (Right) WG pdf in $[-\pi, \pi]$ ($\mu = \frac{\pi}{3}$ and varying σ^2). The θ axis is the unit circle, unfolded. (This figure appears in [6].)

B. Wrapped Gaussian

We can define an analogue to the Gaussian distribution on the unit circle by applying the wrapping function:

$$\psi(x) = \text{mod}\,(x + \pi, 2\pi) - \pi \quad , \tag{3}$$

to a univariate Gaussian random variable $x \sim \mathcal{N}(\mu, \sigma^2)$. This results in the wrapped Gaussian (WG) distribution:

$$P(\theta\,;\,\mu, \sigma^2) = \sum_{l=-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\theta - (\mu + 2\pi l))^2}{2\sigma^2}\right) \quad , \quad \theta \in \mathbb{S}^1 \quad . \tag{4}$$

The WG is depicted in Fig. 1 in the interval $[-\pi, \pi]$ as well as on the unit circle, where it is embedded. This distribution is useful for deriving closed-form updates in an EM framework, where the infinite summation is truncated appropriately. Unfortunately, the number of components that must be considered in the (truncated) WG increases exponentially with the number of dimensions. To illustrate this, consider the pdf of a bivariate WG:

$$P(\boldsymbol{\theta}\,;\,\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{l_1, l_2 = -\infty}^{\infty} \frac{1}{|2\pi\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}\left(\boldsymbol{\theta} - \left(\boldsymbol{\mu} + 2\pi \begin{bmatrix} l_1 & l_2 \end{bmatrix}^\top\right)\right)^\top \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{\theta} - \left(\boldsymbol{\mu} + 2\pi \begin{bmatrix} l_1 & l_2 \end{bmatrix}^\top\right)\right)\right] \quad , \tag{5}$$

where $\boldsymbol{\theta} \in \mathbb{S}^2$. We must keep at least 9 components in a truncated bivariate WG to cover wrapping effects at all boundaries of $\mathbb{S}^2$. We must consider at least 27 components for a 3-dimensional truncated WG, etc.

C. von Mises

The von Mises (vM) distribution can be derived by conditioning a 2D Gaussian random vector:

$$\mathbf{x} \sim \mathcal{N}\left(\begin{bmatrix} \cos(\mu) & \sin(\mu) \end{bmatrix}^\top, \kappa^{-1}\mathbf{I}\right) \quad , \tag{6}$$

on the unit circle and converting from Cartesian to polar coordinates. The pdf is:

$$P(\theta\,;\,\mu, \kappa) = \frac{1}{2\pi I_0(\kappa)} \exp(\kappa \cos(\theta - \mu)) \quad , \quad \theta \in \mathbb{S}^1 \quad . \tag{7}$$

Unlike with the WG, there is no need to consider multiple copies of a template distribution and thus there is no need to truncate the vM. The cost of this is an inability to derive closed-form expressions for EM algorithms that use the vM to encode goodness-of-fit. This is often the case since the mean and concentration parameters, μ and κ, cannot be isolated from within the cosine and Bessel functions, respectively.

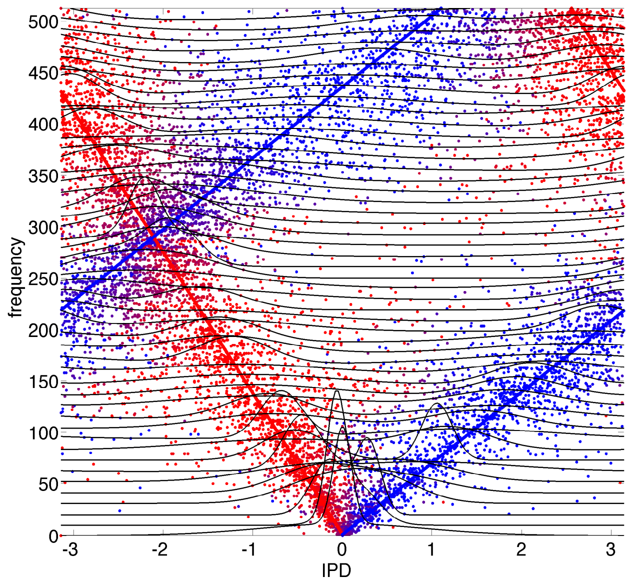

Fig. 2. Mixture of 2 mean-locked wrapped Gaussians fit to simulated IPD data with EM. The data is colored according to its posterior probability. The mixtures of wrapped Gaussians in 50 of the frequency bands are superimposed.

III. EM Algorithms For Wrapped Regression

EM algorithms have been developed to solve the wrapped regression problem [7]. The first fits a wrapped line and the second fits a wrapped cubic spline. These can both be extended to the case of multimodal regression functions via mixture density modeling.

A. Wrapped Line

Given a circular-linear dataset, we may assume it was generated by a product of WG distributions:

$$P\left(\mathbf{x}\,;\,\boldsymbol{\mu}\,,\,\boldsymbol{\sigma}^2\right)=\prod_{d=1}^{D}\sum_{l=-\infty}^{\infty}\frac{1}{\sqrt{2\pi\sigma_d^2}}\exp\left(-\frac{\left(x_d-\left(\mu_d+2\pi l\right)\right)^2}{2\sigma_d^2}\right)\quad,\quad \mathbf{x}\in\mathbb{S}^D\quad, \tag{8}$$

where the components of $\mathbf{x}$ are assumed to be statistically independent for the sake of simplicity and computational tractability. We will further assume that the means lie on a wrapped line with slope a:

$$\mu_d=\psi\left(a\,d\right)\quad, \tag{9}$$

An EM algorithm [7] that (locally) maximizes the likelihood of a dataset can be devised to learn the slope and variance parameters. An example of a mixture of two lines fit in this way is shown in Fig. 2.

B. Wrapped Cubic Spline

We can also generalize this model to the nonlinear case by considering a piecewise polynomial [8] fit to the data. A wrapped cubic spline can be defined that is parameterized by $4M$ coefficients, where M is the number of polynomial sections. An EM algorithm [9] can also be derived in this case. Fig. 3 depicts an example of a spline that was fit in this way.

C. Limitations of EM Algorithms

The methods described have some fundamental limitations. The linear model is very simple, but is only appropriate for circular-linear regression where there is a (wrapped) linear relationship between the two variables. The cubic spline is more general, but is more costly to learn. However, the greatest limitation is that in order to derive

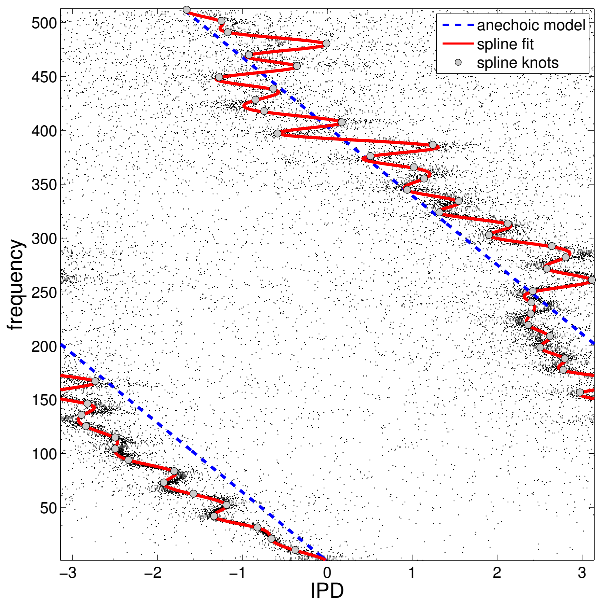

Fig. 3. Wrapped cubic spline fit to a circular-linear dataset. This IPD dataset was derived from a recording in a simulated reverberant environment. The non-linearity of the data as a function of frequency makes the spline more appropriate than the line.

closed-form expressions for the M-step update rules, we must use the wrapped Gaussian distribution to model error in the circular variable(s). For high-dimensional problems, this quickly becomes intractable since the number of components in the WG increases exponentially as a function of the number of circular variables. Ideally, we would like to be able to train a model that is as flexible as the spline, but doesn't suffer from using the WG. For this reason, we will formulate a conditional density neural network that uses the von Mises distribution to measure error. This will result in a linear increase in complexity as a function of the circular variables.

IV. Conditional Gaussian Neural Network

In this section, we will describe how a neural network can be used to learn a general mapping between two real-valued variables. The following section will extend these results to the wrapped case. We model the (possibly multimodal) conditional density of a dataset with a mixture of Gaussians:

$$P\left(\theta \,|\, d;\, \boldsymbol{\mu}\,,\, \boldsymbol{\sigma}^2\,,\, \boldsymbol{\pi}\right) = \sum_{j=1}^{K} \pi_j(d) \frac{1}{\sqrt{2\pi\sigma_j(d)^2}} \exp\left(-\frac{(\theta-\mu_j(d))^2}{2\,\sigma_j(d)^2}\right) \quad , \tag{10}$$

A two-layer neural network can be designed in which the first layer applies a linear transformation to the input followed by a non-linearity and the second layer applies another linear transformation. The outputs are transformed appropriately to derive the parameters of a mixture of Gaussians. The first two layers are given by:

$$\mathbf{z} = tanh\left(\mathbf{W}^{(1)} \begin{bmatrix} d \\ 1 \end{bmatrix}\right) \quad , \qquad \mathbf{y} = \mathbf{W}^{(2)} \begin{bmatrix} \mathbf{z} \\ 1 \end{bmatrix} \quad , \tag{11}$$

where d is the input value, $\mathbf{z}$ is the vector of hidden layer activations, $\mathbf{y}$ is the vector of output layer activations, and $\mathbf{W}^{(i)}$ is the weight matrix for the i^{th} layer. The output layer values are grouped so that transformations can be applied to derive the mixture model's parameters:

$$\mu_j = y_{\mu,j} \quad , \quad \sigma_j^2 = \exp\left(y_{\sigma^2,j}\right) \quad , \quad \pi_j = \frac{\exp\left(y_{\pi,j}\right)}{\sum\limits_{m=1}^{K} \exp\left(y_{\pi,m}\right)} \quad . \tag{12}$$

The error criterion to minimize is the negative data log likelihood. For the i^{th} observation, this is given by:

$$E_i = -\log\left[\sum_{j=1}^{K}\pi_j\frac{1}{\sqrt{2\pi\sigma_j^2}}\exp\left(-\frac{(\theta_i-\mu_j)^2}{2\sigma_j^2}\right)\right] \quad . \tag{13}$$

As in [10], we define the posterior probability:

$$\eta_{ij} = \frac{\pi_j\frac{1}{\sqrt{2\pi\sigma_j^2}}\exp\left(-\frac{(\theta_i-\mu_j)^2}{2\sigma_j^2}\right)}{\sum\limits_{m=1}^{K}\pi_m\frac{1}{\sqrt{2\pi\sigma_m^2}}\exp\left(-\frac{(\theta_i-\mu_m)^2}{2\sigma_m^2}\right)} \quad . \tag{14}$$

The gradients with respect to the mixture parameters are:

$$\frac{\partial E_i}{\partial \mu_j} = -\eta_{ij}\frac{\theta_i-\mu_j}{\sigma_j^2} \quad , \tag{15}$$

$$\begin{aligned}\frac{\partial E_i}{\partial \sigma_j^2} &= -\frac{\pi_j\frac{1}{\sqrt{2\pi}}\left[\exp\left(-\frac{(\theta_i-\mu_j)^2}{2\sigma_j^2}\right)\left(-\frac{1}{2}\frac{1}{\left(\sigma_j^2\right)^{\frac{3}{2}}}\right)+\frac{1}{\sqrt{\sigma_j^2}}\left(\exp\left(-\frac{(\theta_i-\mu_j)^2}{2\sigma_j^2}\right)\frac{(\theta_i-\mu_j)^2}{\left(\sigma_j^2\right)^2}\right)\right]}{\sum\limits_{m=1}^{K}\pi_m\frac{1}{\sqrt{2\pi\sigma_m^2}}\exp\left(-\frac{(\theta_i-\mu_m)^2}{2\sigma_m^2}\right)} \\ &= \eta_{ij}\left[\frac{1}{2\sigma_j^2}-\frac{(\theta_i-\mu_j)^2}{\left(\sigma_j^2\right)^2}\right] \quad ,\end{aligned} \tag{16}$$

$$\frac{\partial E_i}{\partial \pi_j} = -\frac{\eta_{ij}}{\pi_j} \quad . \tag{17}$$

We also have that:

$$\frac{\partial \mu_j}{\partial y_{\mu,j}} = 1 \quad , \tag{18}$$

$$\frac{\partial \kappa_j}{\partial y_{\sigma^2,j}} = \sigma_j^2 \quad , \tag{19}$$

$$\frac{\partial \pi_k}{\partial y_{\pi,j}} = \begin{cases}\pi_k\left(1-\pi_k\right) & , \quad j=k \\ -\pi_j\pi_k & , \quad j\neq k\end{cases} \quad = \quad [j=k]\,\pi_k-\pi_j\pi_k \quad . \tag{20}$$

Combining these two results together, we have that the derivatives of the error function with respect to the network outputs are:

$$\frac{\partial E_i}{\partial y_{\mu,j}} = \eta_{ij}\left(\frac{\mu_j-\theta_i}{\sigma_j^2}\right) \quad , \tag{21}$$

$$\frac{\partial E_i}{\partial y_{\sigma^2,j}} = \eta_{ij}\left[\frac{1}{2}-\frac{(\theta_i-\mu_j)^2}{\sigma_j^2}\right] \quad , \tag{22}$$

$$\frac{\partial E_i}{\partial y_{\pi,j}} = \sum_{k=1}^{K}\frac{\partial E_i}{\partial \pi_k}\frac{\partial \pi_k}{\partial y_{\pi,j}} = \pi_j-\eta_{ij} \quad . \tag{23}$$

Now we can apply error back-propagation to evaluate the derivatives with respect to the network weights:

$$\frac{\partial E_i}{\partial W_{mn}^{(2)}} = \frac{\partial E_i}{\partial y_m}\frac{\partial y_m}{\partial W_{mn}^{(2)}} = \frac{\partial E_i}{\partial y_m}z_n \quad , \tag{24}$$

$$\frac{\partial E_i}{\partial W_{no}^{(1)}} = \left(\sum_m\frac{\partial E_i}{\partial y_m}\frac{\partial y_m}{\partial z_n}\right)\frac{\partial z_n}{\partial W_{no}^{(1)}} = \left(\sum_m\frac{\partial E_i}{\partial y_m}W_{mn}^{(2)}\right)\left(1-z_n^2\right)d_o \quad . \tag{25}$$

In these expressions, the index indicating that a network value corresponds to the i^{th} input token has been omitted where convenient. The simplest approach to updating the weights is to perform a gradient descent step:

$$\mathbf{W}^{(l)} \longleftarrow \mathbf{W}^{(l)} - \eta \, \frac{1}{N} \sum_{i=1}^{N} \frac{\partial E_i}{\partial \mathbf{W}^{(l)}} \quad , \quad l = 1, 2 \tag{26}$$

The step size η should be chosen to promote a steady decrease in the objective function. After each iteration, the network outputs are recomputed along with the gradients. We can enhance each step by performing a line search in the direction of the negative gradient. Alternatively, we can use a second-order optimization strategy with a quasi-Newton method such as BFGS [8].

The flexibility of the function to be learned depends on how many nodes are used in the hidden layer. If a mapping between two linear variables is desired and the inputs are scaled to the range $[-1, 1]$, including 100 hidden nodes seems to provide a reasonable tradeoff between computational complexity and flexibility.

V. Conditional von Mises Neural Network

We will now extend the results of the previous section to the case where the output space is $\mathbb{S}^C$.

A. *Single circular variable*

First, consider a neural network that defines a conditional mixture of univariate von Mises distributions:

$$P\left(\theta \,|\, d\,;\, \boldsymbol{\mu}\,,\, \boldsymbol{\kappa}\,,\, \boldsymbol{\pi}\right) = \sum_{j=1}^{K} \pi_j\left(d\right) \frac{1}{2\pi I_0\left(\kappa_j\left(d\right)\right)} \exp\left[\kappa_j\left(d\right) \cos\left(\theta - \mu_j\left(d\right)\right)\right] \quad , \tag{27}$$

As before, we have that:

$$\mathbf{z} = tanh\left(\mathbf{W}^{(1)} \begin{bmatrix} d \\ 1 \end{bmatrix}\right) \quad , \tag{28}$$

$$\mathbf{y} = \mathbf{W}^{(2)} \begin{bmatrix} \mathbf{z} \\ 1 \end{bmatrix} \quad . \tag{29}$$

The outputs are divided into groups associated with each of the parameters. We further transform these values to represent the parameters themselves as the outputs of the network:

$$\mu_j = \psi\left(y_{\mu,j}\right) \quad , \quad \kappa_j = \exp\left(y_{\kappa,j}\right) \quad , \quad \pi_j = \frac{\exp\left(y_{\pi,j}\right)}{\sum\limits_{m=1}^{K} \exp\left(y_{\pi,m}\right)} \quad , \tag{30}$$

where the wrap operator was defined in (3). The associated network diagram is shown in Fig. 4. The vM-based error criterion is:

$$E_i = -\log\left[\sum_{j=1}^{K} \pi_j \frac{1}{2\pi I_0\left(\kappa_j\right)} \exp\left[\kappa_j \cos\left(\theta_i - \mu_j\right)\right]\right] \quad . \tag{31}$$

We define the posterior probability:

$$\eta_{ij} = \frac{\pi_j \frac{1}{2\pi I_0(\kappa_j)} \exp\left[\kappa_j \cos\left(\theta_i - \mu_j\right)\right]}{\sum\limits_{m=1}^{K} \pi_m \frac{1}{2\pi I_0(\kappa_m)} \exp\left[\kappa_m \cos\left(\theta_i - \mu_m\right]\right)} \quad . \tag{32}$$

Taking the derivative of (31) with respect to the parameters, we get:

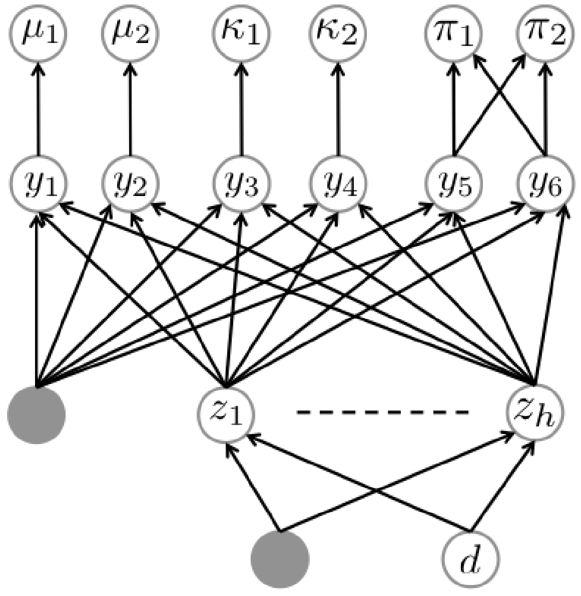

Fig. 4. Diagram for the von Mises neural network. A linear input value undergoes two linear transformations separated by a non-linearity. Following this, the vM mixture distribution parameters are determined from suitable transformations applied to the output values.

$$\frac{\partial E_i}{\partial \mu_j} = -\eta_{ij}\,\kappa_j \sin\left(\theta_i - \mu_j\right) \quad , \tag{33}$$

$$\begin{aligned}\frac{\partial E_i}{\partial \kappa_j} &= -\frac{\pi_j\left[\frac{1}{2\pi I_0(\kappa_j)}\exp\left(\kappa_j\cos\left(\theta_i-\mu_j\right)\right)\left(-\frac{I_1(\kappa_j)}{I_0(\kappa_j)}\right)+\frac{1}{2\pi I_0(\kappa_j)}\exp\left(\kappa_j\cos\left(\theta_i-\mu_j\right)\right)\cos\left(\theta_i-\mu_j\right)\right]}{\sum_{m=1}^{K}\pi_m\frac{1}{2\pi I_0(\kappa_m)}\exp\left(\kappa_m\cos\left(\theta_i-\mu_m\right)\right)} \\ &= \eta_{ij}\left[\frac{I_1\left(\kappa_j\right)}{I_0\left(\kappa_j\right)} - \cos\left(\theta_i-\mu_j\right)\right] \quad ,\end{aligned} \tag{34}$$

$$\frac{\partial E_i}{\partial \pi_j} = -\frac{\eta_{ij}}{\pi_j} \quad . \tag{35}$$

We also have that:

$$\frac{\partial \mu_j}{\partial y_{\mu,j}} = 1 \quad , \tag{36}$$

$$\frac{\partial \kappa_j}{\partial y_{\kappa,j}} = \kappa_j \quad , \tag{37}$$

$$\frac{\partial \pi_k}{\partial y_{\pi,j}} = \begin{cases}\pi_k\left(1-\pi_k\right) & , \quad j = k \\ -\pi_j\pi_k & , \quad j \neq k\end{cases} \quad = \quad \left[j=k\right]\pi_k - \pi_j\pi_k \quad . \tag{38}$$

Combining these two results together, we have that the derivatives of the error function with respect to the network outputs are:

$$\frac{\partial E_i}{\partial y_{\mu,j}} = -\eta_{ij}\,\kappa_j \sin\left(\theta_i - \mu_j\right) \quad , \tag{39}$$

$$\frac{\partial E_i}{\partial y_{\kappa,j}} = \eta_{ij}\,\kappa_j\left[\frac{I_1\left(\kappa_j\right)}{I_0\left(\kappa_j\right)} - \cos\left(\theta_i-\mu_j\right)\right] \quad , \tag{40}$$

$$\frac{\partial E_i}{\partial y_{\pi,j}} = \sum_{k=1}^{K}\frac{\partial E_i}{\partial \pi_k}\frac{\partial \pi_k}{\partial y_{\pi,j}} = \pi_j - \eta_{ij} \quad . \tag{41}$$

Now we can apply error back-propagation to evaluate the derivatives with respect to the network weights as in (24)-(25).

B. Multiple circular variables

When multiple circular variables are to be predicted from the linear input variable, we have more parameters to learn in the conditional von Mises neural network. If there are C output variables, we have C means to learn per (multivariate) von Mises:

$$\mu_{jc} = \psi\left(y_{\mu,jc}\right) \quad , \tag{42}$$

where a single concentration parameter is learned per mixture component for simplicity (the vM distributions are "spherical"). The error criterion is:

$$\begin{aligned} E_i &= -\log\left[\sum_{j=1}^{K}\pi_j \prod_{c=1}^{C}\frac{1}{2\pi I_0\left(\kappa_j\right)}\exp\left(\kappa_j\cos\left(\theta_{ic}-\mu_{jc}\right)\right)\right] \\ &= -\log\left[\sum_{j=1}^{K}\pi_j\left(\frac{1}{2\pi I_0\left(\kappa_j\right)}\right)^C\exp\left(\kappa_j\sum_{c=1}^{C}\cos\left(\theta_{ic}-\mu_{jc}\right)\right)\right] \quad . \end{aligned} \tag{43}$$

We define the posterior probability:

$$\eta_{ij} = \frac{\pi_j \prod_{c=1}^{C}\frac{1}{2\pi I_0(\kappa_j)}\exp\left(\kappa_j\cos\left(\theta_{ic}-\mu_{jc}\right)\right)}{\sum_{m=1}^{K}\pi_m\prod_{c=1}^{C}\frac{1}{2\pi I_0(\kappa_m)}\exp\left(\kappa_m\cos\left(\theta_{ic}-\mu_{mc}\right)\right)} \quad . \tag{44}$$

Taking the derivative with respect to the parameters, we get:

$$\frac{\partial E_i}{\partial \mu_{jc}} = -\eta_{ij}\,\kappa_j\sin\left(\theta_{ic}-\mu_{jc}\right) \quad , \tag{45}$$

$$\begin{aligned} \frac{\partial E_i}{\partial \kappa_j} &= -\frac{\pi_j\left[C\left(\frac{1}{2\pi I_0(\kappa_j)}\right)^C\exp\left(\kappa_j\sum_{c=1}^{C}\cos\left(\theta_{ic}-\mu_{jc}\right)\right)\left(-\frac{I_1(\kappa_j)}{I_0(\kappa_j)}\right)+\left(\frac{1}{2\pi I_0(\kappa_j)}\right)^C\exp\left(\kappa_j\sum_{c=1}^{C}\cos\left(\theta_{ic}-\mu_{jc}\right)\right)\sum_{c=1}^{C}\cos\left(\theta_{ic}-\mu_{jc}\right)\right]}{\sum_{m=1}^{K}\pi_m\prod_{c=1}^{C}\frac{1}{2\pi I_0(\kappa_m)}\exp\left(\kappa_m\cos\left(\theta_{ic}-\mu_{mc}\right)\right)} \\ &= \eta_{ij}\left[C\frac{I_1\left(\kappa_j\right)}{I_0\left(\kappa_j\right)}-\sum_{c=1}^{C}\cos\left(\theta_i-\mu_j\right)\right] \quad , \end{aligned} \tag{46}$$

$$\frac{\partial E_i}{\partial \pi_j} = -\frac{\eta_{ij}}{\pi_j} \quad . \tag{47}$$

We also have that:

$$\frac{\partial \mu_{jc}}{\partial y_{\mu,jc}} = 1 \quad , \tag{48}$$

$$\frac{\partial \kappa_j}{\partial y_{\kappa,j}} = \kappa_j \quad , \tag{49}$$

$$\frac{\partial \pi_k}{\partial y_{\pi,j}} = \begin{cases}\pi_k\left(1-\pi_k\right) & , \quad j=k \\ -\pi_j\pi_k & , \quad j\neq k\end{cases} \quad = \quad [j=k]\,\pi_k-\pi_j\pi_k \quad . \tag{50}$$

Thus, we have that the derivatives of the error function with respect to the network outputs are:

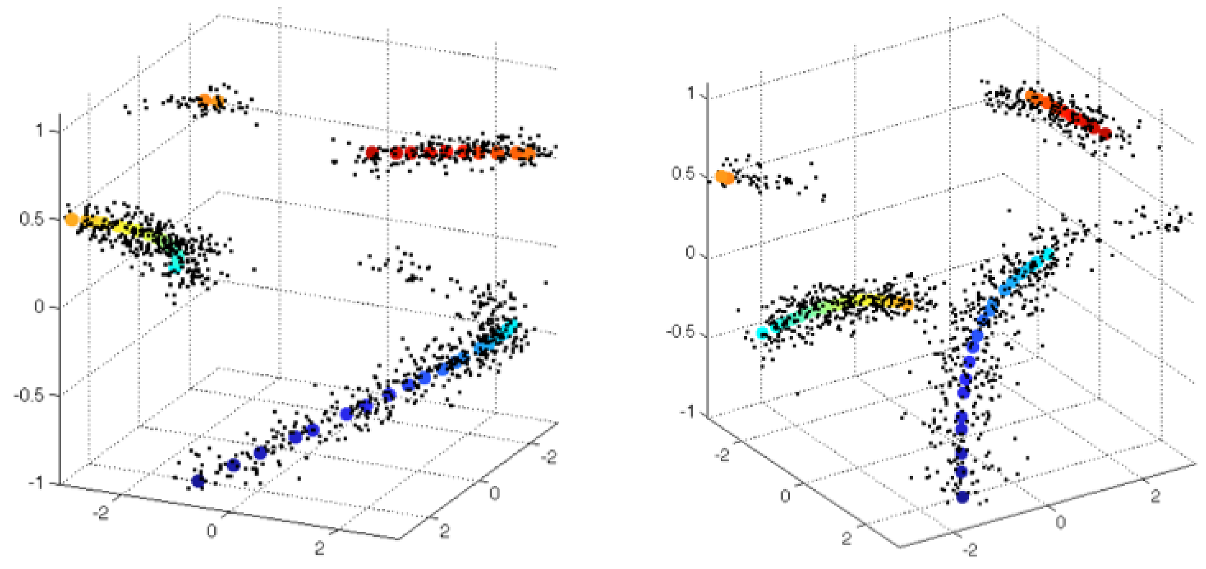

Fig. 5. Wrapped, non-linear function fit to a synthetic dataset with one linear variable (vertical axis) and two circular variables via gradient descent with back-propagation. Black dots are data points and colored dots represent the mean function of the output distribution in the vM neural network.

$$\frac{\partial E_i}{\partial y_{\mu,jc}} = -\eta_{ij}\,\kappa_j \sin\left(\theta_{ic} - \mu_{jc}\right) \quad , \tag{51}$$

$$\frac{\partial E_i}{\partial y_{\kappa,j}} = \eta_{ij}\,\kappa_j \left[C \frac{I_1\left(\kappa_j\right)}{I_0\left(\kappa_j\right)} - \sum_{c=1}^{C} \cos\left(\theta_{ic} - \mu_{jc}\right) \right] \quad , \tag{52}$$

$$\frac{\partial E_i}{\partial y_{\pi,j}} = \sum_{k=1}^{K} \frac{\partial E_i}{\partial \pi_k} \frac{\partial \pi_k}{\partial y_{\pi,j}} = \pi_j - \eta_{ij} \quad . \tag{53}$$

Now we can apply error back-propagation to evaluate the derivatives with respect to the network weights as before. An example of a non-linear, wrapped function learned by training a two-output vM neural net is depicted in Fig. 5. While there are local optima present by default in the optimization of a neural net, the periodic nature of a wrapped feature space exacerbates the problem. Thus, a significant challenge is to devise an optimization strategy to learn multiple wrapped regression functions at once. This is the most interesting case for IPD feature-based source separation since we effectively care to cluster data that lies along multiple wrapped functions.

VI. Conclusion

Preliminary experiments suggest that a wrapped conditional density neural network is the most theoretically appropriate model for learning a general functional mapping from a linear variable to one or more circular variables. The EM-trained linear model is limited to finding wrapped line trends and the wrapped cubic spline has a flexibility that is difficult to control. In addition, each iteration of EM to train the spline requires the inversion of a relatively large matrix.

The neural network approach is by far the most computationally intensive for small problems. However, it doesn't suffer from an exponential increase in computational complexity as a result of increasing the number of output variables. This is because we must use the wrapped Gaussian distribution to derive closed-form M-step updates in the EM algorithms and we are not restricted in this way when training the neural network. In addition, the flexibility of the function learned by the neural net can be controlled fairly well by appropriately choosing the number of nodes in the hidden layer. Thus, it appears that the neural net is the most convenient model for circular-linear regression. Once augmented with an efficient optimization routine, this could be a powerful approach.

References

[1] T. D. Downs and K. V. Mardia, “Circular regression,” *Biometrika*, vol. 89, no. 3, 2002.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[3] O. Yilmaz and S. Rickard, “Blind separation of speech mixtures via time-frequency masking,” *IEEE Transactions on Signal Processing*, vol. 52, pp. 1830–1847, 2004.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[5] K. Mardia and P. Jupp, *Directional Statistics*, Wiley, 1999.

[6] J. Traa and P. Smaragdis, “A wrapped Kalman filter for azimuthal speaker tracking,” *IEEE Signal Processing Letters*, vol. 20, no. 12, pp. 1257–1260, 2013.

[7] J. Traa, “Multichannel source separation and tracking with phase differences by random sample consensus,” M.S. thesis, University of Illinois at Urbana-Champaign, 2013.

[8] M. T. Heath, *Scientific Computing: An Introductory Survey*, McGraw Hill, 2 edition, 2002.

[9] J. Traa and P. Smaragdis, “Robust inter-channel phase difference modeling with wrapped regression splines,” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) (in review)*.

[10] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1st edition, 1996.