Kernel-Based Extensions of Exponential Family Distributions

Andrew Suciu, Jessica Taylor

March 14, 2014

1 Introduction

An exponential family distribution is of the form:

$$p_\eta(x) = \exp(\eta^T \phi(x) - A(\eta) + B(x))$$

We consider an extension of the form:

$$p_\eta(x) = \exp(\langle \eta, R_x \rangle - A(\eta) + B(x))$$

where $R_x = k(x, \cdot)$ is the representer of $x \in \mathcal{X}$ in an RKHS $\mathcal{H}$ corresponding to kernel $k \in \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $\langle \cdot, \cdot \rangle$ is the inner product in $\mathcal{H}$, and $A(\eta) = \log \int \exp(\langle \eta, R_x \rangle + B(x)) dx$.

This family of distributions has been considered in previous work. Smola and Canu define kernelized exponential families and develop some techniques using them; notably, kernelized classification, regression and novelty detection **??**. However, in this process, the authors do not completely handle density estimation. This is unfortunate, because we expect this extension to perform well for density estimation due to the fact that p_η can approximate any continuous distribution arbitrarily well when k is a universal kernel. Therefore, we will consider the task of setting η to maximize likelihood of values $x_1, ..., x_n$ without overfitting. We succeed and demonstrate the estimation method on sample datasets.

2 Density Estimation

The function to be maximized is:

$$f(\eta) = \sum_{i=1}^{n} \log p_\eta(x_i) - \psi(\eta)$$

where $\psi(\eta) = \frac{\alpha}{2} \|\eta\|_{\mathcal{H}}^2$ is a regularization term. We will end up optimizing this using Newton's method:

$$\begin{aligned} \eta_{t+1} &= \eta_t + \Delta \\ \Delta &= -(Hf(\eta))^{-1} \nabla f(\eta) \end{aligned}$$

Note that the Hessian is a matrix in RKHS space. Although matrices are typically only defined for vector spaces $\mathbb{R}^n$, it is easy to consider them as a binary function in the RKHS of type $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$, as vectors in the RKHS are unary functions of type $\mathcal{X} \to \mathbb{R}$. If M is a matrix in $\mathcal{H}$ and v is a vector in $\mathcal{H}$, then the matrix multiplication is defined as $(Mv)(x) = \langle M(x, \cdot), v \rangle$, analogous to ordinary matrix multiplication $(Mv)_i = M_{i,*}v$. The outer product is defined as $(f \otimes g)(x, y) = f(x)g(y)$.

The gradient and Hessian of f can be expressed as:

$$\begin{aligned} f(\eta) &= \sum_{i=1}^{n} \log p_\eta(x_i) - \psi(\eta) \\ \nabla f(\eta) &= \sum_{i=1}^{n} R_{x_i} - n\mathbb{E}_\eta[R_X] - \alpha\eta \\ Hf(\eta) &= n(\mathbb{E}_\eta[R_X] \otimes \mathbb{E}_\eta[R_X] - \mathbb{E}_\eta[R_X \otimes R_X]) - \alpha I_{\mathcal{H}} \end{aligned}$$

This result is derived in the second appendix.

The Newton delta is computed as:

$$\Delta = -(Hf(\eta))^{-1} \nabla f(\eta)$$

Suppose we take samples $y_1, ..., y_m$ from p_η. It will be useful to define $x_{n+i} = y_i$, so all samples (observed and sampled) are in the same list. Also assume that η can be expressed as a linear combination of additional values $x_{n+m+1}, ..., x_{n+m+r}$, so that $\eta = \sum_{i=1}^{r} \theta_i R_{x_{n+m+i}}$. Then we can estimate the gradient and Hessian as:

$$\begin{aligned} \nabla f(\eta) &= \sum_{i=1}^{n} R_{x_i} - \frac{n}{m} \sum_{j=1}^{m} R_{y_i} - \alpha\eta \\ Hf(\eta) &= -\frac{n}{m} \sum_{i=1}^{m} R_{y_i} \otimes R_{y_i} + n \left(\frac{1}{m} \sum_{i=1}^{m} R_{y_i} \right) \otimes \left(\frac{1}{m} \sum_{i=1}^{m} R_{y_i} \right) - \alpha I_{\mathcal{H}} \end{aligned}$$

We would like a finite representation of the gradiant, Hessian, and Newton delta as coefficients multiplied by representers (or outer products of represents):

$$\begin{aligned} \nabla f(\eta) &= \sum_{i=1}^{n+m+r} d_i R_{x_i} \\ Hf(\eta) &= \sum_{i=1}^{n+m+r} \sum_{j=1}^{n+m+r} h_{i,j} R_{x_i} \otimes R_{x_j} - \alpha I_{\mathcal{H}} \\ \Delta &= \sum_{i=1}^{n+m+r} \delta_i R_{x_i} \end{aligned}$$

To get the correct values, we will set:

$$\begin{aligned} d_i &= 1 && \text{for } 1 \le i \le n \\ d_i &= -n/m && \text{for } n+1 \le i \le n+m \\ d_i &= -\alpha\theta_{i-n-m} && \text{for } n+m+1 \le i \le n+m+r \end{aligned}$$

$$h_{i,j} = [n+1 \le i \le n+m][n+1 \le j \le n+1]\left(\frac{n}{m^2} - [i=j]\frac{n}{m}\right)$$

We can also write $\mathbf{d}$ to represent the vector of d_i values, δ to represent the vector of δ_i values, and $\mathbf{H}$ to represent the matrix of $h_{i,j}$ values. The task is now to compute δ.

Rearranging the definition of Δ yields:

$$Hf(\eta)\Delta = -\nabla f(\eta)$$
$$\left(x \to \left\langle \sum_{i=1}^{n+m+r} \sum_{j=1}^{n+m+r} h_{i,j} R_{x_i}(x) R_{x_j}, \sum_{i=1}^{n+m+r} \delta_i R_{x_i} \right\rangle\right) - \alpha\Delta = -\nabla f(\eta)$$
$$\sum_{i=1}^{n+m+r} \sum_{j=1}^{n+m+r} h_{i,j}\delta_j k(x_i, x_j) R_{x_i} - \alpha \sum_{i=1}^{n+m+r} \delta_i R_{x_i} = -\sum_{i=1}^{n+m} d_i R_{x_i}$$

In particular, for every integer $1 \le l \le n+m+r$,

$$\left\langle \sum_{i=1}^{n+m+r} \sum_{j=1}^{n+m+r} h_{i,j}\delta_j k(x_i, x_j) R_{x_i} - \alpha \sum_{i=1}^{n+m+r} \delta_i R_{x_i}, R_{x_l} \right\rangle = \left\langle -\sum_{i=1}^{n+m} d_i R_{x_i}, R_{x_l} \right\rangle$$
$$\sum_{i=1}^{n+m+r} \sum_{j=1}^{n+m+r} h_{i,j}\delta_j k(x_i, x_j) k(x_i, x_l) - \alpha \sum_{i=1}^{n+m+r} \delta_i k(x_i, x_l) = -\sum_{i=1}^{n+m+r} d_i k(x_i, x_l)$$

We are solving for δ, so it will be useful to write this as a linear constraint on δ:

$$\sum_{j=1}^{n+m+r} \delta_j \left(\sum_{i=1}^{n+m+r} h_{i,j} k(x_i, i_j) k(x_i, x_l) - \alpha k(x_j, x_l) \right) = -\sum_{i=1}^{n+m+r} d_i k(x_i, x_l)$$
$$(\mathbf{KHK} - \alpha K)_{l,*}\delta = -(\mathbf{Kd})_l$$

So δ can be found by solving a system of linear equations:

$$\delta = (\alpha\mathbf{K} - \mathbf{KHK})^{-1}\mathbf{Kd} = (\mathbf{K}(\alpha I - \mathbf{HK}))^{-1}\mathbf{Kd} = (\alpha I - \mathbf{HK})^{-1}\mathbf{d}$$

Let $\mathbf{H}_{m\times m}$ indicate the nonzero values in $\mathbf{H}$, and $\mathbf{K} = \begin{bmatrix} \mathbf{K}_{n\times n} & \mathbf{K}_{n\times m} & \mathbf{K}_{n\times r} \\ \mathbf{K}_{m\times n} & \mathbf{K}_{m\times m} & \mathbf{K}_{m\times r} \\ \mathbf{K}_{r\times n} & \mathbf{K}_{r\times m} & \mathbf{K}_{r\times r} \end{bmatrix}$. Then

$$\mathbf{HK} = \begin{bmatrix} \mathbf{0}_{n\times n} & \mathbf{0}_{n\times m} & \mathbf{0}_{n\times r} \\ \mathbf{H}_{m\times m}\mathbf{K}_{m\times n} & \mathbf{H}_{m\times m}\mathbf{K}_{m\times m} & \mathbf{H}_{m\times m}\mathbf{K}_{m\times r} \\ \mathbf{0}_{r\times n} & \mathbf{0}_{r\times m} & \mathbf{0}_{r\times r} \end{bmatrix}$$

At this point, due to the definition of $\mathbf{d}$ and the fact that $\mathbf{HK}$ has all zeros in columns not between $n+1$ and $n+m$, it is possible to infer that

$$\delta_i = \frac{1}{\alpha} \qquad \text{for } 1 \le i \le n$$
$$\delta_i = -\theta_{i-n-m} \qquad \text{for } n+m+1 \le i \le n+m+r$$

Given these values, it is possible to compute $\delta_{n+1..n+m}$:

$$\delta_{n+1..n+m} = (\alpha I - \mathbf{H}_{m\times m}\mathbf{K}_{m\times m})^{-1}(d_{n+1..n+m} + \mathbf{H}_{m\times m}\mathbf{K}_{m\times n}\delta_{1..n} + \mathbf{H}_{m\times m}\mathbf{K}_{m\times r}\delta_{n+m+1..n+m+r})$$

At this point, running Newton's method is straightforward. In each iteration, first take m samples from η_t. Next, set

$$\eta_{t+1} = \sum_{i=1}^{n+m} \delta_i R_{x_i}$$

This is because the original η_t cancels with $\sum_{i=n+m+1}^{n+m+r} \delta_i R_{x_i}$, making the computation more efficient, since the number of samples necessary to store η does not grow over time.

To test out these methods, we applied it to 2-dimensional density estimation. We used a Gaussian kernel

$$k(x, y) = \exp\left(\frac{-n}{2\sigma_1^2}(x_1 - y_1)^2 + \frac{-n}{2\sigma_1^2}(x_2 - y_2)^2\right)$$

where σ_i^2 is the empirical variance of ith component in the observations $x_1...x_n$. If we had set $B(x) = 0$ uniformly, then the resulting distribution would have an infinite integral. Therefore, we set B to represent a normal distribution:

$$B(x) = \frac{-1}{2\sigma_1^2}(x_1 - \mu_1)^2 + \frac{-1}{2\sigma_2^2}(x_2 - \mu_2)^2$$

where μ_i is the empirical mean of the ith component in the observations.

3 Empirical Validation

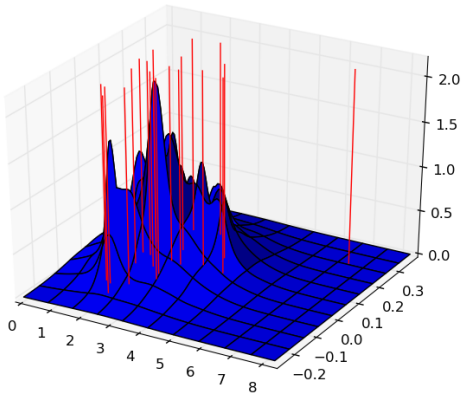
We ran the method above on 20 randomly selected data samples from the R language's 'wavesurge' dataset containing observations of wave height and the corresponding surge height. We compared the resulting density estimates to scipy's kernel density estimator on the same samples. Empirically, we found that running for more than a few iterations (e.g. 3 vs 10 iterations) did not change our method's estimates significantly. For the 2874 data points not in the training set, our method assigned a mean log probability of -1.511, while the KDE assigned a mean log probability of -3.930. See Figure 1 in the appendix. This indicates that, in certain cases, our method is competitive with KDE. For comparison across more data sets, see the third appendix.

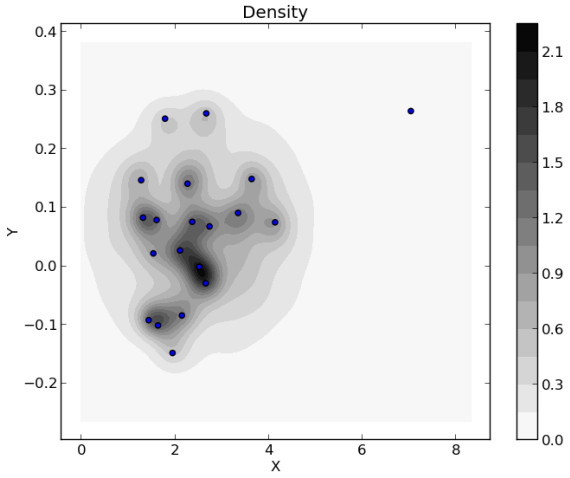
4 Appendix

4.1 Visual Comparison of Density Estimates

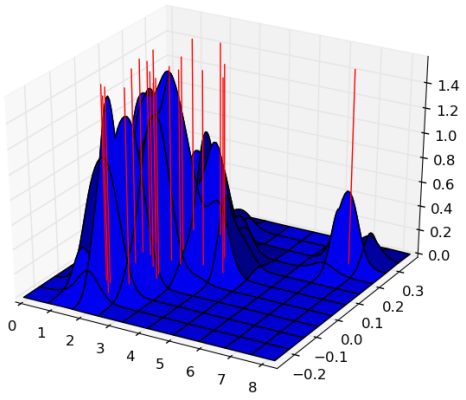
Figure 1: 3D and contour plots of estimated density on two-dimensional ocean wave data. This data, 'wavesurge', can be found in the 'texmex' package for the R language.

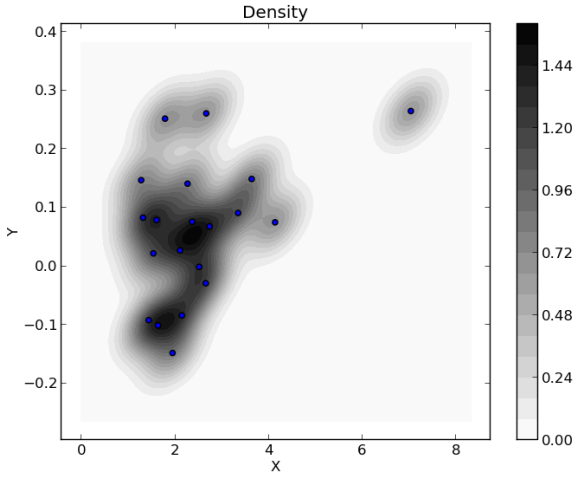
Our Method





Kernel Density Estimator





4.2 Derivation of Gradient and Hessian

To find the gradient and Hessian of the objective f, first take the gradient of log likelihood with respect to a single x value:

$$\begin{aligned}
\log p_\eta(x) &= B(x) + \langle \eta, R_x \rangle - A(\eta) \\
\nabla_\eta \log p_\eta(x) &= R_x - \nabla_\eta A(\eta) \\
&= R_x - \nabla_\eta \log \int \exp(\langle \eta, R_{x'} \rangle - B(x'))dx' \\
&= R_x - \frac{\exp(\langle \eta, R_{x'} \rangle + B(x'))R_{x'}dx'}{\int \exp(\langle \eta, R_{x'} \rangle + B(x'))dx'} \\
&= R_x - \mathbb{E}_\eta[R_X]
\end{aligned}$$

And also the Hessian:

$$\begin{aligned}
H_\eta \log p_\eta(x) &= J_\eta(\phi(x) - \mathbb{E}_\eta[R_X]) \\
&= -J_\eta \left(\int \exp(\eta^T R_{x'} - A(\eta) + B(x'))R_{x'}dx' \right) \\
&= -\int \exp(\eta^T R_{x'} - A(\eta) + B(x'))R_{x'}(\nabla_\eta(\eta^T R_{x'} - A(\eta)))^T dx' \\
&= -\int \exp(\eta^T R_{x'} - A(\eta) + B(x'))R_{x'}(R_{x'} - \nabla_\eta A(\eta))^T dx' \\
&= -\mathbb{E}_\eta[R_X \otimes (R_X - \mathbb{E}_\eta[R_X])] \\
&= -\mathbb{E}_\eta[R_X \otimes R_X] + \mathbb{E}_\eta[R_X] \otimes \mathbb{E}_\eta[R_X]
\end{aligned}$$

which is equal to the negative covarinace matrix of R_X under η.

The gradient and Hessian of the regularizer can also be found:

$$\begin{aligned}
\psi(\eta) &= \frac{\alpha}{2}\|\eta\|_{\mathcal{H}}^2 \\
\nabla\psi(\eta) &= \alpha\eta \\
H\psi(\eta) &= \alpha I_{\mathcal{H}}
\end{aligned}$$

where $I_{\mathcal{H}}$ is the identity matrix in $\mathcal{H}$, which can be derived as:

$$\langle R_x, v \rangle = v(x) = (I_{\mathcal{H}} v)(x) = \langle I_{\mathcal{H}}(x, \cdot), v \rangle$$

so $I_{\mathcal{H}}(x, \cdot) = R_x$, i.e. $I_{\mathcal{H}} = k$.

From here it is easy to compute the gradient and Hessian of the objective:

$$\begin{aligned}
f(\eta) &= \sum_{i=1}^n \log p_\eta(x_i) - \psi(\eta) \\
\nabla f(\eta) &= \sum_{i=1}^n R_{x_i} - n\mathbb{E}_\eta[R_X] - \alpha\eta \\
Hf(\eta) &= n(\mathbb{E}_\eta[R_X] \otimes \mathbb{E}_\eta[R_X] - \mathbb{E}_\eta[R_X \otimes R_X]) - \alpha I_{\mathcal{H}}
\end{aligned}$$

4.3 Comparison to KDE

In the following table we list the results of training both our proposed method (with a gaussian kernel) and KDE on various sample datasets commonly packaged with the R language. In each case, we run both methods on the same 20 samples from the total dataset, then calculate the mean log probability of observing the remaining data given the initial sample.

Dataset	**Mean Log Probablity**	**KDE MLP**
Yearly Treering Data	-9.598	-9.985
Rain, wavesurge	-1.511	-3.930
Average Heights of the Rio Negro	-7.288	-8.0376
Mauna Loa Atmospheric CO2	-8.074	-6.385
Bivariate Data Set with 3 Clusters	-9.358	-9.033
Strikes Duration	-4.350	-4.862
Monthly Sunspot Numbers	-10.958	-11.824
Galton's Mid parent child height data	-4.443	-4.647
Monthly Sunspot Data	-10.936	-11.316
Galton's Peas	-4.590	-4.776
Macdonell's Data on Height and Finger Length of Criminals	-0.753	-1.313
Galton's data on the heights of parents and their children	-4.471	-5.394