Non-Linear Adaptive Spline Kernel Filtering

Yash Bachwana

Department of Electrical Engineering, Indian Institute of Technology Gandhinagar

Introduction

- Function approximation tasks include system identification and noise cancellation.
- Traditional methods like linear regression and neural networks can be computationally expensive.
- We propose a a spline based Non-Linear Network, that can function as a Single Layer Neural Network.

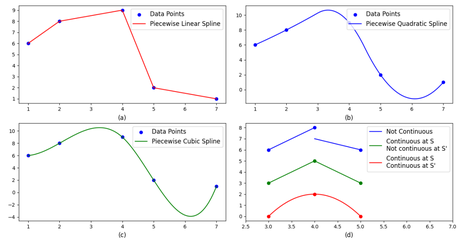

Figure 1. Spline Features (a) Linear (b) Quadratic (c) Cubic (d) Continuity in splines

Adaptive Splines

Splines are mathematical curves commonly used for the interpolation or approximation of a series of Q points known as knots. A classical spline of degree n is C^{n-1} continuous at its join points, with each segment being C^{∞} continuous.[1]

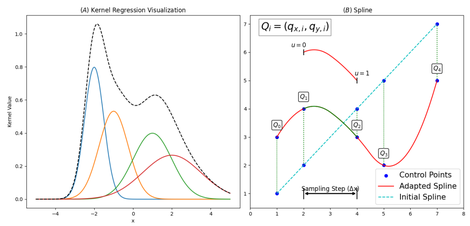

Figure 2. (a) Kernel Regression (b) Adaptive Spline

Kernel Adaptive Filtering (KAF)

- Kernel adaptive filtering is a powerful technique, used for modeling nonlinear data.
- **KLMS Algorithm** allocates a new kernel unit for the new training data with input $u(i)$ as the center and $\eta e(i)$ as the coefficient. The coefficients and the centers are stored in memory during training. [2].

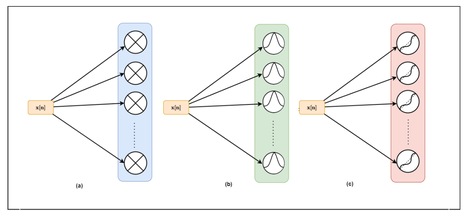

Figure 3. Architectures with activation layers containing (a) Multipliers (b) Gaussians (c) Splines

- **Disadvantage of the KLMS algorithm:** The number of centers x_n that get involved in the estimation of the result (Dictionary) increases continually. This leads to constant increase in memory and computational power demand.

Proposed Algorithm

KLMS and similar algorithms expand their network as more data points are encountered, theoretically sound but quite impractical in reality. Moreover, these employ Gaussian kernels with fixed shapes, limiting adaptability. We propose a novel approach that advocates for a fixed number of adaptable spline-based kernels, enhancing versatility, and feasibility of the system.

Mathematically, a spline curve can be expressed as:

$$\phi(u) = \phi_i(u), \forall i \in \{0, \ldots, Q\} \text{ and } u \in [0, 1).$$

where, $\phi_i(u)$ is a local polynomial of degree P, known as the curve span. The local polynomial $\phi_i(u)$ can be represented as:

$$\phi_i(u) = \mathbf{u^T C q_i}$$

where, $\mathbf{C} \in \mathbb{R}^{(P+1)\times(P+1)}$ is referred to as a spline basis matrix. The vector $\mathbf{u}$ is defined as: $\mathbf{u} \in \mathbb{R}^{1\times(P+1)} = [u^P \; u^{P-1} \; \ldots \; u^1]^T$ where each u represents the normalized abscissa value between two knots. The vector $\mathbf{q_i} \equiv \mathbf{q_{y,i}}$ consists of control points and is defined as: $\mathbf{q_i} \in \mathbb{R}^{(P+1)\times 1} = [q_i \; q_{i+1} \; \ldots \; q_{i+P}]^T$

The output at the k^{th} neuron of the neural network is a function of u and i which depends on the input $x[n]$.

$$u = \frac{s[n]}{\Delta x} - \left\lfloor \frac{s[n]}{\Delta x} \right\rfloor \tag{1}$$

$$i = \left\lfloor \frac{s[n]}{\Delta x} \right\rfloor + \frac{Q-1}{2} \tag{2}$$

$$d_k(n) = \phi_i(u) = \mathbf{u^T C q_i} \tag{3}$$

where i is the span index, index of the closest control point and u is the local parameter. The online learning rule can be derived by minimizing a cost function (CF) typically defined as

$$\hat{J}(q_{i,n}) = E[e_n^2] \approx e_n^2 \tag{4}$$

For the minimization of (4), we proceed by applying the vanilla gradient adaptation. Finally, indicating explicitly the time index n, the LMS iterative learning algorithm can be written as:

$$Q_y[j](i:i+P) = Q_y[j](i:i+P) + \mu \cdot e_k[n] \cdot C' \cdot \mathbf{u}' \tag{5}$$

where the parameters mu represent the learning rates for the control points.

Algorithm 1 Adaptive Kernel Spline Approximation

Initialization:

Set of abscissa and ordinate of control points Q_x, Q_y

Initialize with a line $Q_y = Q_x$

Learning rate (μ), Number of Splines/Kernels (N), Number of control points (Q)

Degree of Spline (P), P degree spline basis matrix C, Sampling Step (Δx)

Computation:

while {s[n],d[n]} available **do**

  $u = \frac{s[n]}{\Delta x} - \lfloor \frac{s[n]}{\Delta x} \rfloor$

  $\mathbf{u} = [u^2 \; u \; 1]$

  for $k = 1 : N$ **do**

    $i = \lfloor \frac{s[n]}{\Delta x} \rfloor - \lfloor \frac{Q_x[k][1]}{\Delta x} \rfloor + 1$

    if $i \geq 1$ and $i \leq Q - P$ **then**

      $y_k[n] = \mathbf{u} \cdot C \cdot Q_y[k](i:i+P)'$

      $e_k[n] = d[n] - y_k[n]$

      $Q_y[j](i:i+P) = Q_y[j](i:i+P) + \mu \cdot e_k[n] \cdot C' \cdot \mathbf{u}'$

    end if

  end for

end while

Simulation Study

Assuming $P = 2$, $\mu = 0.025$, $N = 2$, $Q = 21$, $\Delta x = 0.025$ and

$$C = \begin{bmatrix} 0.5 & -1 & 0.5 \\ -1 & 1 & 0 \\ 0.5 & 0.5 & 0 \end{bmatrix}$$

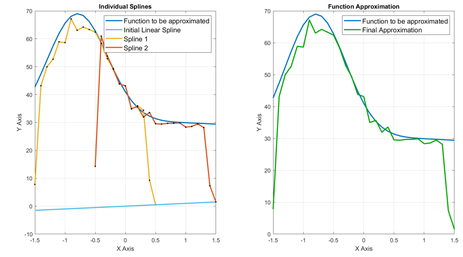

Figure 4. Approximation of a weighted sum of gaussian, with added 30dB white gaussian noise using splines starting from a line

Results

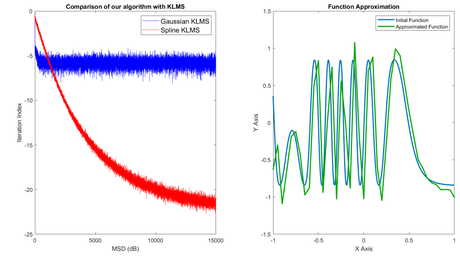

Figure 5. Comparison of performance of Gaussian and Spline for a randomly chosen function sine of gaussian, with added 30dB white gaussian noise

Advantages: Splines offer a high level of versatility while necessitating only control points for storage. Their continuity up to the C^{n-1} derivative, enables adaptability to a wide range of functions.

Limitations: Splines require prior knowledge of the input range. Additionally, their efficacy can be compromised in case of high levels of discontinuity.

Conclusion

A low-complexity and highly versatile strategy for function approximation has been developed.

References

[1] L. Schumaker, "Spline functions: basic theory," 2007.

[2] W. Liu, P. P. Pokharel, and J. C. Principe, *The Kernel Least-Mean-Square Algorithm*, vol. 56. 2008.