Summary

sagisk

February 2, 2022

1 Chapter 5: Radial Basis Function Networks (RBF)

Typically has only one hidden layer because of the special structure of the layer in contrast to the conventional NN's where the depth of the network is responsible for the increase of the nonlinearity.

Input, Hidden and Output layers in RBF:

1. Input layer transmits from the input features to the hidden layer. Its dimensionality is equal to the d dimensionality of the data. The input units are fully connected to the hidden units.

2. Hidden layer takes input points and transforms them into a new space that is often linearly separable. It usually has higher dimensions m than the input layer but lower dimension than the number of training points.

3. The output layer uses linear classification or regression modeling for the inputs from the hidden layer. It is trained in the usual feed-forwatd method.

Hidden units special structure: Each hidden unit has a d-dimensional prototype vector. - $\bar{\mu_i}$ be the prototype vector of the ith hidden unit. - σ_i be the associated bandwidth. If it is typical to take all bandwidts equal to some constant σ the prototypes are individual for each hidden unit. - The activation $\Phi_i(X)$ for a training point X and ith hidden unit is: $h_i = \Phi_i(X) = e^{\frac{-\|X-\bar{\mu_i}\|^2}{2\sigma_i^2}}$.

1.1 Training an RBF Network

- Training the Hidden Layer: Model complexity is regulated by the number of hidden units (m) and the bandwidth σ_i. Increased model complexity:

- small bandwidth and a big m is useful when the data is large.
- big bandwidth and small m is required when the data is small to avoid overfitting.

Setting the bandwidth: Depends on the prototype vectors. Should be set so that each training point be influenced only by a small number of prototype vectors, which correspond to its closest clusters. Too big bandwidth will result under-fitting. Too small bandwidth will result over-fitting. The bandwidth is compared to the inter-prototype distance. It can be set by:

1. A Heuristic rule is to set: $\sigma = \frac{d_{\text{max}}}{\sqrt{m}}, \sigma = 2d_{\text{ave}}$ where $d_{\text{max}}, d_{\text{ave}}$ are respectfully the maximum and average distance between pairs of prototypes.
   - Disadvantages: The bandwidth may vary depending on the density of the data region we concern. If the density is high the bandwidth should be smaller than the one in the sparse regions.
   - Alternative: Set σ_i to be equal to the distance of u_i to its rth nearest neighbor among prototypes. Usually $r = 5, 10$.

2. Using held-out data set. We chose σ's to be in the neighborhood of the above recommended than observe the RBF performance depending on the bandwidth value.

- Setting the prototypes:

1. Can be sampled from n training points.
    - Disadvantage: Overrepresent the prototypes from dense regions, wheres sparse regions would get few to no prototypes.
2. k-means clustering m centroids can be used as prototypes. (Most common choice).
3. Data space partitioning clustering algorithms can be used like decision trees.
4. Orthogonal least squares algorithm chooses prototype vectors one-by-one from the training data to minimize the residual error on the validation set. At each iteration it tries the remaining data points as prototypes and compares the resulting validation errors. The data point that reduces the error the most is selected as a prototype and is excluded for the further iterations from the remaining data. In practise orthogonal least squares orthogonalizes the matrix H and uses its orhogonal vectors to compute which prototype should be selected from the data.
5. Fully Supervised Learning (see below).

- Training the Output Layer:

- Input points: $X_1, ..., X_n$
- Hidden layer representations of input points $H_1, ..., H_n$ where $H_i \in R^m$ a row vector. Stack them with each other to create $n \times m$ matrix H.
- Weights: $W = [w_1, ..., w_m] \in R^m$.
- The numeric response $y \in R^n$.
- The predicted outcomes: $\hat{y_i} = H_i W^T = \sum_{j=1}^{m} w_j \Phi_j(X_i)$.
- The loss function: $L = \frac{1}{2}\|HW^T - y\|^2$ or by Tikhonov regularization: $L = \frac{1}{2}\|HW^T - y\|^2 + \lambda/2\|W\|^2$.
- The loss gradient: $H^T(HW^T - y) + \lambda W$.
- Setting the derivative of L with respect to W to zero we have: $W^T = (H^T H + \lambda I)^{-1} H^T y$.

Fully Supervised Learning: Computes the partial derivative of the loss function with respect to the bandwidth and each element of the prototype and update it by backpropogation. $\frac{dL}{d\sigma_j} = \sum_{i=1}^{n}(H_i W - y_i) w_j \frac{d\Phi_j(X_i)}{d\sigma_j} = \sum_{i=1}^{n}(H_i W - y_i) w_j \Phi_j(X_i) \frac{\|X_i - \mu_j\|^2}{\sigma_j^3}$ If all bandwidths are fixed: $\frac{dL}{d\Sigma} = \sum_{j=1}^{m} \frac{dL}{d\sigma_j}\frac{d\sigma_j}{d\sigma} = \sum_{j=1}^{m} \frac{dL}{d\sigma_j}$. For prototype elements: $\frac{dL}{du_{jk}} = \sum_{i=1}^{n}(H_i W - y_i)\Phi_j(X_i)\frac{(x_{ik} - \mu_{jk})}{\sigma_j^2}$

- Disadvantages: Is in inefficient to train. The loss function has many local minimia.

References