Hybrid Genetic Algorithm (GA)-Based Neural Network for Multispectral Image Fusion

Xia PENG, Anrong DANG
School of Architecture, Tsinghua University
Beijing, P. R. China 100084

To use multisource data in remote sensing classifications and to effectively extract information, data fusion is the key issue. Data fusion techniques combine data from multiple sensors, and related information from associated databases, to take a better decision than from one source only, by reducing imprecision and uncertainty and increasing completeness. Therefore, it is useful to combine and analyze the multisource data to take advantage or their characteristics and improve the information extraction process.

Data fusion is generally performed at three different levels of information representation; these are pixel level, feature level and decision level. Fusing images at pixel level means to perform integration at a level where the pixels are least processed. Each pixel in the fused image is calculated from pixels in the source images by for-example averaging. Fusion at feature level first requires extraction of representative features from the source images (through e.g. segmentation); fusion then takes place based on features that match some selection criteria. At decision level, the output from the initial object detection and classification based on source images is then fed into the fusion algorithm. Every image fusion algorithm is performed at one of these three levels or some combination thereof.

This paper presents a real coded GA strategy and hybrid with a Back Propagation (BP) algorithm. The genetic operators are carefully designed to optimize the neural network, to evaluate its accuracy. We use the most widely used neural network namely the Multi-Layer Perception (MLP), along with sigmoid transfer functions in the hidden layers. Assuming we have a three-layer feed-forward neural network with m inputs (channels) and k outputs (categories), and l hidden nodes. Each neuron in the hidden layer uses sigmoid function f(x)

as its threshold function, and each neuron in the output layers uses Purelin function $p(x)$ as its threshold function. The neuron output of hidden node h ($1 \le h \le l$) and output node $q(1 \le q \le k)$can be expressd as:

$$z_h = f(W^T X) = f\left(\sum_{i=1}^{m} \omega_i x_i - \delta_h\right)$$

$$o_q = p(V^T Z) = p\left(\sum_{i=1}^{l} v_i z_i - \delta_q\right)$$

The surperscript T stands for a vector transpose, $W = [\omega_1, \omega_2, \cdots, \omega_i, \cdots, \omega_m]$ is the weight connection vector between the input nodes and hidden node h, $V = [v, v_2, \cdots, v_i, \cdots, v_k]$ is the weight connection vector between the hidden nodes and output node q, $X = [x_1, x_2, \cdots, x_i, \cdots, x_m]$ is the input vector for each hidden node, and $Z = [z_1, z_2, \cdots, z_i, \cdots, z_m]$ is the output vector of the hidden nodes, δ_h and δ_q are the corresponding biases for hidden node h and output node q. z_h and o_q are the output neuron respectively responses for node h and node q. The sigmoid function f(x) is defined as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

where $x \in [-\infty, +\infty]$. And the purelin function p(x) is defined as:

$$p(x) = \alpha x + \beta$$

where α is a non-zero constant, β is the bias; and $\alpha, \beta \in [-\infty, +\infty]$.

The back propagation learning algorithm is a popular optimization tool for neural network training. However, the gradient descent which serves as one of its essential characteristics has several drawbacks. On the one hand, the performance of the network learning is strictly dependent on the shape of the error surface, values of the initial connection weights, and some further sophisticate parameters. On the other hand, when using steepest descent to train a multilayer network with sigmoid functions, the inherent defects of sigmoid

functions will result in tiny changes in the weights and biases, although they are far from their optimal values. In order to compensate for deficiencies in the gradient descent algorithm, this paper presents a method which is the integration of genetic algorithm and neural network for weight training. It consists of three major phases. The first phase is to decode each genotype in the current generation into a set of connection weights. The second phase is to evaluate each set of the connection weights by constructing the corresponding neural network through decoding each genome and computing its fitness function and mean square error function. The third step is applying the evolutionary process such as selection, crossover, and mutation operations by a genetic algorithm according to its fitness. The evolution stops when the fitness is greater than a predefined value (e.g. the training error is smaller than a designated value) or the population has converged.

The image data used to test our methodology correspond to the region of Greater Beijing, China. Two Landsat TM images of this area were used: One collected on October, 1996 and the other on October 2004 respectively. The result of land cover classification shows that the hybrid GA algorithm-based neural network classifier has the better overall accuracy than back propagation neural network classifier.

Short Bibliography

[1] Bandyopadhyay, S., Pal, S. K., 2001. Pixel classification using variable string genetic algorithms with chromosome differentiation. IEEE Transaction on Geoscience and Remote Sensing, 39(2): 303-308.

[2] Benediktsson, J. A., Swain, P. H., Ersoy, O. K., 1990. Neural Network approaches versus statistical methods in classification of multisource remote sensing data. IEEE Transaction on Geoscience and Remote Sensing, 28(4): 540-552.

[3] Carpenter G. A., Gjiaja M. N., Gopal S., and Woodcock C. E., 1997. ART neural networks for remote sensing: Vegetation classification from Landsat TM and terrain data. IEEE Transaction on Geoscience and Remote Sensing, 35(2): 308-325.

[4] Dai X., and Khorram S., 1999. Data fusion using artificial neural networks: a case study

on multitemporal change analysis. Computers, Environment and Urban Systems, 23: 19-31.