Classifying Cells for Cancer Diagnosis Using Neural Networks

Ciamac Moallemi

THE WORK REPORTED HERE NETTED THE 15-YEAR-OLD AUTHOR FIFTH PLACE AND A $15,000 SCHOLARSHIP IN THE 50TH ANNUAL WESTINGHOUSE SCIENCE TALENT SEARCH FOR HIGH-SCHOOL STUDENTS.

A SUCCESSFUL COMPUTER-based system for diagnosing bladder cancer must accurately classify various objects in an image of cells from a urine sample. Typically, an object falls into one of two classes: Well or Not-well. The Well class contains the cells that will actually be useful for diagnosing bladder cancer. The Not-well class includes everything else, such as intercellular "garbage" and cells that would not aid in the diagnosis. In a sense, this classification filters out objects that are not needed. Yet, errors in this classification are undesirable. If a Well object is classified as Not-well, valuable information could be lost. Therefore, the classifier must be accurate if it is to be used clinically.

There have been previous attempts to solve this classification problem. Wong et al. used a "selective mapping algorithm," which used a tree classifier to classify objects by using thresholds for certain extracted features.[1] For example, the system deemed an object Not-well if its area was above or below a certain value. Similar tests were applied to other features. This system achieved a total misclassification rate of 23.2 percent. Systems prior to this also had error rates.[2] Furthermore, the time required by these systems for automated analysis clearly makes them impractical.

My research reduces the number of computer misclassification errors to a level tolerable for clinical use. By reducing this misclassification, more of the cells retained for further analysis are Well, and fewer are Not-well, thus speeding up and improving the accuracy of the actual cancer diagnosis performed on the Well cells. Also, the system can extract the same number of Well cells from fewer images, thus speeding up the total classification time. The actual time to classify a single image is fast enough to make the system practical.

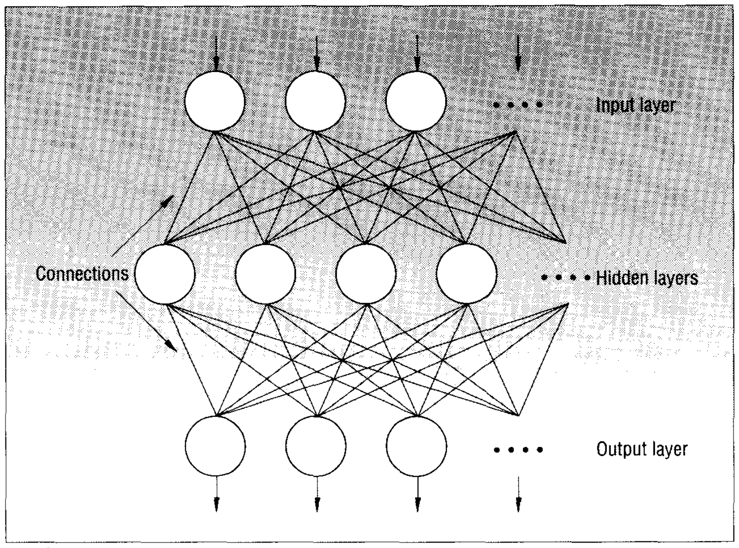
In my classification system, several descriptive features are extracted from each object in the image. These features are then fed to a multilayer perceptron — a type of artificial neural network — which classifies them as Well or Not-well. The perceptron's superior classification abilities yields a high degree of improvement over conventional systems. Also, the perceptron's parallelism and other aspects of this implementation lend it to extremely fast computation, thus providing accurate classification at an acceptable speed. A sidebar discusses neural networks, the multilayer perceptron, and two learning algorithms in more detail.

Implementation

This system is designed to classify images that are 256×240 pixels, with 256 gray levels. The images are gathered with a 10×planachromatic objective microscope

Neural networks

Artificial neural networks are computational models designed to generate performance similar to that of the human brain in fields such as speech and image recognition. They mimic the brain and other biological neural networks in that they encompass a massively parallel architecture, and they derive their power from the sheer number of neurons, rather than the complexity of each single neuron. These systems lend themselves to fast implementations in parallel, especially when done in hardware. They differ from conventional systems in that a conventional von Neumann computer processes instructions one at a time, sequentially, while a neural network processes many competing hypotheses at the same time. Also, since there are many nodes, neural networks are more noise tolerant than conventional computer systems; if a few nodes are erroneous, there are still many others and the overall performance of the net might be unaffected.



A multilayer perceptron.

The multilayer perceptron. One type of neural network — the multilayer perceptron — is composed of many simple computational elements, or nodes, that form layers.[1] These layers are linked by weights, which are adapted in a supervised learning process so that the neural network will correctly classify a user-supplied pattern. This can be done via back propagation, an algorithm that "trains" a network on sample data so that the network will respond correctly. The accompanying figure shows a sample perceptron.

The input to the first layer is the pattern to be classified. A node's input is the scalar product of the output vector from the previous level and the weight vector between that level and the node:

$$net_j = \sum_i w_{ji} o_i$$

where net_j is the input to layer j, w_{ji} is the weight matrix between layer j and layer i, and o_i is the output of layer i. A node's output is this value passed through the sigmoidal nonlinearity

$$o_j = f(net_j) = \frac{1}{1 + e^{-(net_j + \theta_j)/\theta_o}}$$

where θ_j is the threshold or bias. For inputs much larger than the bias, the output is close to 1. For small inputs, it is close to zero. These correspond to "on" and "off" states, respectively. For values close to the threshold, the node emits a number between 0 and 1, corresponding to an undecided state. The threshold is treated as a link from a virtual node whose output value is always 1. This way, the value also can be learned. θ_o is a constant that determines the shape of the sigmoid. A small θ_o results in a sigmoid that is like a threshold logic unit, with an abrupt increase from 0 to 1. A large θ_o creates a more gentle graduation between 0 and 1.

Generally, a pattern p will have an output o_{pk} that is different from the desired output t_{pk} (k is an output layer). A learning procedure, such as back propagation, helps minimize this.

Back propagation. The back propagation algorithm seeks to minimize the normalized system error

$$E = \frac{1}{2P} \sum_p \sum_k (t_{pk} - o_{pk})^2$$

with respect to the weights in the neural network.[1] It does this by using an iterative, steepest-descent approach, making incremental changes in the weight space in proportion to the error's rate of change with respect to the weight:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}$$

This simplifies to

$$\Delta w_{ji} = \eta \delta_{pj} o_{pi}$$

where for output layers

$$\delta_{pk} = (t_{pk} - o_{pk}) o_{pk} (1 - o_{pk})$$

and for input and hidden layers

$$\delta_{pj} = o_{pj}(1 - o_{pj}) \sum_k \delta_{pk} w_{kj}$$

In these equations, $o_j(1 - o_j)$ is at its maximum when $o_j = 0.5$, and its minimum when $o_j = 0,1$. This makes sense in light of the fact that a node whose output is 0.5 is considered "undecided," whereas outputs of 0 or 1 indicate nodes set "on" or "off."

Quick propagation. Unfortunately, back propagation is very slow, so I used quick propagation instead.[2] Quick propagation also seeks to minimize the normalized system error E. It can be considered second order, since it uses information about the curvature of the error function as well as its slope. It computes $\partial E/\partial w(t)$, the slope of the error function during the current training cycle, just as back propagation does, yet it does not update the weights quite as simply. It retains copies of $\partial E/\partial w(t-1)$, the slope of the error function during the previous training cycle; $\partial E/\partial w(t)$, the current slope of the error function; and $\Delta w(t-1)$, the last weight change. Using these three values, it adjusts the weights as follows:

$$\Delta w(t) = \frac{\partial E / \partial w(t)}{\partial E / \partial w(t-1) - \partial E / \partial w(t)} \Delta w(t-1)$$

This formula is based on two assumptions, that the error curve is parabolic with respect to each weight, and that the error curve does not change with respect to each weight as other weights are changed. These assumptions are crude; however, iteratively, quick propagation is highly superior to back propagation.

References

1. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D.E. Rumelhart and J.L. McClelland, eds., MIT Press, Cambridge, Mass., 1986.

2. S.E. Fahlman, "Faster Learning Variations on Back Propagation: An Empirical Study," *Proc. 1988 Connectionist Models Summer School*, Morgan Kaufmann, San Mateo, Calif., 1988.

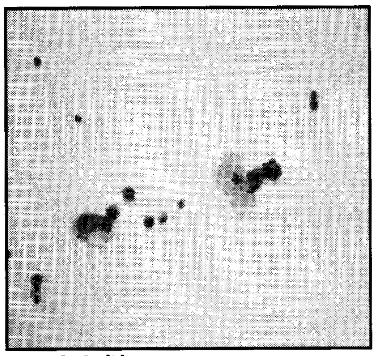


Figure 1. A slide image.

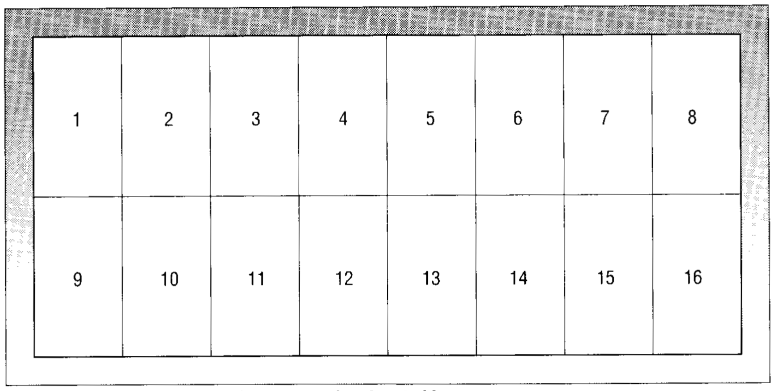


Figure 2. Sixteen segments of the image for thresholding.

lens and an image digitizer. Figure 1 shows a sample slide image.

The classification system proceeds as follows:

(1) The image is thresholded.
(2) Object segmentation is performed.
(3) Features are extracted from the objects.
(4) Features are submitted to a neural network for classification.

Thresholding. The system uses a technique called thresholding to separate cytologic objects from the image background. First, the image is partitioned into 16 segments, each 32×120 pixels (see Figure 2). The system does this because each local segment of the image tends to have a different background, due to lighting and shading differences. Thus, to effectively remove the background, the system must consider each local area individually, not the image as a whole. The system then produces and smooths a histogram of each segment. The smoothing is a local averaging algorithm that attempts to eliminate significant discontinuities in the histogram. Figure 3 shows a region histogram, and Figure 4 shows a smoothed version of that histogram.

Typically, one peak is seen at the light end of this histogram, corresponding to the background pixels, and one long plain is seen at the dark end, corresponding to the pixels of the foreground objects. Based on this histogram, the system selects an intensity threshold in the small valley between these two regions. The smoothing algorithm highlights this threshold, in spite of any local fluctuations in the histogram. Pixels with gray-level values greater than the threshold are considered to belong to the background and are assigned the color white, while those with gray-level values greater than the threshold are considered part of the foreground and retain their color value. Thus, the system produces a uniform background and can distinguish between the background and the actual objects. Figure 5 shows a thresholded version of Figure 1.

Object segmentation. Objects are segmented (or separated from each other) using a "blob coloring" algorithm.[3] This involves sweeping an L-shaped template (see Figure 6) across the image and coloring each distinct object with a distinct set of equivalent colors. The algorithm proceeds as follows:

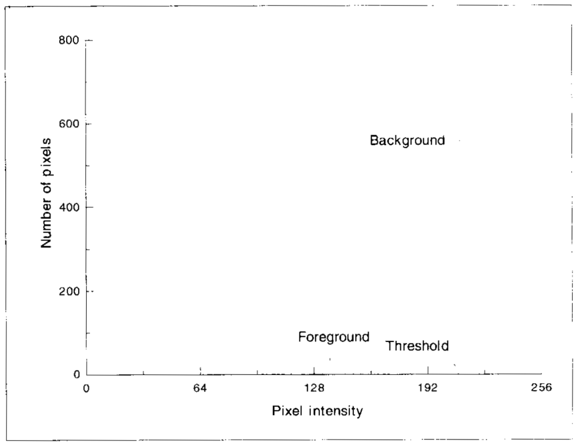


Figure 3. A segment histogram.

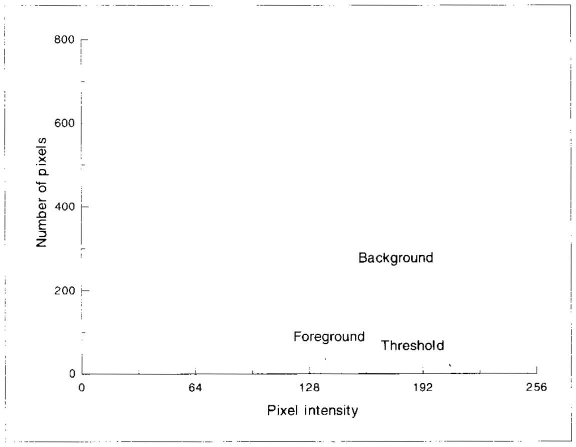


Figure 4. A smoothed version of the histogram in Figure 3.

```
(1) Let the color counter $k = 1$.
(2) Define a function $f(x)$ such that, if $x$ is a background pixel (as determined by the thresholding procedure), then $f(x) = 0$; otherwise $f(x) = 1$.
(3) Scan the image with the L-shaped template from top to bottom and from left to right until $f(X_c) = 1$.
(4) If $f(X_u) = 1$ and $f(X_l) = 0$
        then let the color of $X_c$ be the color of $X_u$.
    If $f(X_l) = 1$ and $f(X_u) = 0$
        then let the color of $X_c$ be the color of $X_l$.
    If $f(X_l) = 1$ and $f(X_u) = 1$
        then let the colors of $X_l$ and $X_u$ be equivalent and assign their value to $X_c$.
    If $f(X_l) = 0$ and $f(X_u) = 0$
        then let the color of $X_c = k$, and let $k = k+1$.
(5) Go to Step 3 until no more pixels are left.
```

The system can thus detect distinct objects by their color. It also assumes that objects with an area of less than 20 pixels are the result of errors in the digitization or thresholding processes, and they are not considered further.

Feature extraction. Once an image has been preprocessed, the system must extract features from the objects in that image. These features are numerical descriptors that will try to describe the object's properties. The system will base its object classification on these features, which it extracts simply by looking at each object. Several descriptive features are used. These features were picked to highlight significant differences between Well cells and other typical objects.

The first such feature is the object area. Well cells come in a small range of sizes — the object area is usually 100 to 400 pixels — and Not-well objects can often be detected simply by being too big or too small. Since Well cells are usually circular, it is also useful to measure the object's circularity. Thus, the second feature is the ratio of the object's area to the area of the rectangular box enclosing it. For a circle, this is $\pi/4 \approx 0.785$. For Well cells, this is usually between 0.60 and 0.85.

Another clear property of Well cells is that they have circular, well-defined nuclei that take up a significant portion of the object. This introduces three more features: the nuclear area, the ratio of the nucleus to the enclosing box, and the ratio of the entire object to the nucleus. The nucleus is segmented by thresholding, where the threshold is a constant (empirically determined to be 0.9) multiplied by the object's average optical density. The nuclear area is usually less than 150 pixels. The ratio of the nucleus to the enclosing box is used in the same way as the ratio of the object to the enclosing box. This value is generally above 0.35. Finally, the ratio of the nuclear area to the total cell area is usually less than 0.85.

Network classification. The features are then fed to a multilayer perceptron for classification. The network has an input layer with five nodes, two hidden layers with ten nodes each, and an output layer with one node. A value in the output node greater than 0.5 signifies a Well cell, while a value less than 0.5 signifies a Not-well cell. This network architecture was chosen by trial and error. Smaller networks did not adapt well to the training set, while larger networks took too long to train and did not generalize well (they trained too closely to the training set).

Computer programs. I wrote several computer programs (more than 3,100 lines) during this research. These can be divided into two main sets of routines.

The first set (around 1,800 lines) performed the actual image manipulation, that is, the thresholding, object segmentation, and feature extraction. There were two main programs in this set of code. The first used a graphical user interface to let the user visually select which objects were Well and which were Not-well. It then stored a list describing the state of the objects (Well or Not-well) in a file. The second program took this list of objects and extracted features from the objects in it. These programs were developed and used on a Sun 3/80 workstation.

The second set of routines (about 1,350 lines) dealt with neural networks and consisted of two programs. The first trained a multilayer perceptron, given its configuration (the number of layers, nodes per layer, and so on) and a set of training patterns. The training algorithm was back propagation, with the quick-propagation modifications described in the sidebar. The second program tested a given multilayer perceptron with a set of weights provided by the first program for a set of test patterns. I developed these programs on the Sun 3/80, but performed the actual training on a Convex C-120 minisupercomputer for speed.

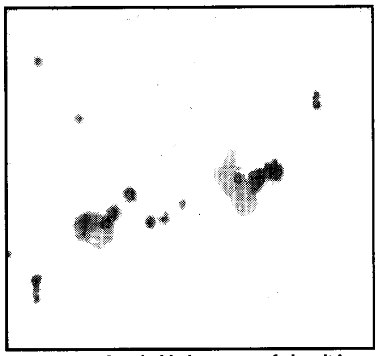


Figure 5. A thresholded version of the slide image in Figure 1.

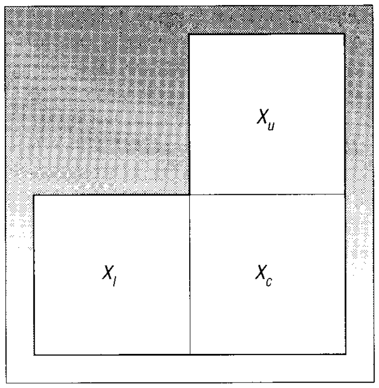


Figure 6. L-shaped template used in object segmentation.

Testing and results

In developing and testing the classification system, I used 43 images provided by the Montefiore Medical Center, Bronx, New York. The images contained a total of 597 objects. Of these, 77 were visually classified as Well, and 520 as Not-well. I tested the system by comparing the visual classifications (representing the decisions of a "perfect" doctor) with those of the system.

Once the thresholding, object segmentation, and feature extraction were completed, half of the objects were randomly se-

Table 1. Classification system results.

Visual Classification	Computer Classification: Accept No.	%	Reject No.	%	Total No.
Accept	71	93.4	15		86
Reject	6		505	97.0	511
Total	77		520		597

lected to be used in training. The results are shown in Table 1. The neural network achieved an accuracy rate of 93.4 percent in accepting objects as Well, while its accuracy in detecting Not-well objects was 97.0 percent. This leads to an overall accuracy rate of 96.5 percent, or an error rate of 3.5 percent — an order-of-magnitude improvement over the 23.1 percent error rate reported by Wang et al.[1] However, these rates cannot be compared directly, since Wang et al. used the training set as the testing set, whereas my network was trained with only half of the testing set. Also, I defined objects as being greater than 20 pixels, while Wang et al. defined objects as being greater than 30 pixels (although the fraction of objects in the 20- to 30-pixel range is probably small). Another study reported an error rate of 18.1 percent,[2] but that is not directly comparable, either, since that study used images at 25× magnification, while I used images at 10× magnification.

Time considerations. An image typically went through the thresholding, object segmentation, and feature extraction stages in 2.6 seconds on the Sun 3/80 (in real time). The network classification stage required an additional 0.4 seconds, yielding a total classification time of about 3.0 seconds per image. Previous methods reported times of 32 seconds per image[1] or worse[2] in real time on a PDP-11 computer. These methods probably would run faster on modern hardware, yielding comparable times to those in my study. However, since most of the steps in this system are parallel, fast execution times could be expected in a hardware implementation.

Training the neural network, though, was very time consuming. Training typically took a few hours on the Convex C-120, depending on the system load. This is not significant, however, because training only needs to be performed once, not with every new specimen.

THIS MODEL COULD SERVE AS the front-end for a complete bladder cancer diagnosis system. Barring the development of such a system, it could give a doctor images of only those cells that would be useful to the diagnosis, thus speeding up that diagnosis by eliminating the need to manually use a microscope and search a slide for useful cells.

Acknowledgments

I thank Edward K. Wong for his assistance and ideas. I also acknowledge the support of the New York State Center for Advanced Technology in Telecommunications at Polytechnic University and the Youth in Engineering and Science program at Polytechnic University.

References

1. E.K. Wong et al., "A Selective Mapping Algorithm for Computer Analysis of Voided Urine Cell Images," *Analytical and Quantitative Cytology and Histology*, Vol. 11, No. 3, June 1989, pp. 203-209.
2. A.B. Sherman et al., "Bladder Cancer Diagnosis by Computer Image Analysis of Cells in the Sediment of Voided Urine Using a Video Scanning System," *Analytical and Quantitative Cytology and Histology*, Vol. 8, No. 3, Sept. 1986, pp. 177-186.
3. D.H. Ballard and C.M. Brown, *Computer Vision*, Prentice Hall, Englewood Cliffs, N.J., 1982.

Ciamac Moallemi is an undergraduate student in computer science at the Massachusetts Institute of Technology. His research interests include artificial intelligence, neural networks, and machine learning.

Moallemi was a student at Benjamin N. Cardozo High School when he submitted this project to the Westinghouse Science Talent Search. He competed with 1,572 other entrants, and was selected by a panel of eight scientists following interviews to evaluate the students' scientific creativity and potential.

Readers can reach him at 4 Ames St., Cambridge, MA 02142; e-mail ciamac@athena.mit.edu

The Westinghouse Science Talent Search is sponsored by Westinghouse and administered by the nonprofit Science Service. Of the 2,000 finalists since the program's inception, five have won Nobel Prizes, two have won Fields Medals for distinguished work in mathematics, eight have been awarded MacArthur Foundation fellowships for research in the physical and life sciences, 28 have been elected to the National Academy of Sciences, and three have been elected to the National Academy of Engineering.