# Dapple: Improved Techniques for Finding Spots on DNA Microarrays

Jeremy Buhler
Computer Science and Engineering
Box 352350
University of Washington
Seattle, WA 98195-2350

Trey Ideker
Molecular Biotechnology
Box 357730
University of Washington
Seattle, WA 98195-7730

David Haynor
Radiology
Box 356000
University of Washington
Seattle, WA 98195-6000

*UW CSE Technical Report UWTR 2000-08-05*

## Abstract

A key step in experiments using DNA microarrays is locating the thousands of individual spots in a scanned array image. Each spot provides quantitative information about a distinct DNA sequence, so it is imperative that spots be found and quantitated accurately. Spot finding is complicated by variations in the positions and sizes of spots and by the presence of artifacts and background noise in microarray images.

We describe **Dapple**, a new spot finding implementation for microarrays on glass slides. Dapple finds spots using morphological information which is robust to both variation and artifacts. It achieves high spot finding throughput and accuracy by learning to evaluate the quality of candidate spots from examples supplied by the user. Dapple's techniques are useful for improving the accuracy of data acquisition from DNA microarrays.

**Availability**: Dapple runs on UNIX-like systems using the Qt GUI toolkit for X. The C++ source code and a prebuilt Linux binary are available at *http://www.cs.washington.edu/homes/jbuhler/dapple/*.

## 1 Introduction

The DNA microarray [15, 13] is a popular and effective method for assaying the expression of large numbers of genes at once. Arrays have been used to interrogate expression of hundreds or thousands of genes simultaneously, both in yeast [4] and in human cells [14]. Gene expression data derived from arrays may be used for gene clustering [7], tissue differentiation [1], and other analyses.

If gene expression levels derived from microarrays are to be used for analytical purposes, they must be quantitatively accurate. A comprehensive quality control and assessment process for microarray data must contend with numerous sources of experimental error. We will address one particular error source: the difficulty of identifying the locations and extents of labeled DNA spots in a scanned microarray image. We call this issue the **spot finding problem**.

Solving the spot finding problem is crucial to making accurate expression measurements because errors in spot finding, the first step in processing microarray data, propagate to all subsequent analyses. Three constraints combine to make the problem difficult. First, the spot finder must be robust to substantial uncertainties in spot size and position caused by variations in the amount of DNA on each spot and in the location where it is spotted. Second, the finder must cope with both diffuse image noise and discrete image artifacts arising from airborne particles or nonuniform washing of the array surface. Image artifacts in particular may be as large and bright as one or even several adjacent spots. Third, the spot finder must be efficient and effective when applied to large numbers of spots.

A single experiment may utilize tens of arrays ($10^5$ spots), while a very high-throughput user, such as a pharmaceutical company, might produce tens of thousands of arrays ($10^8$ spots) per year. Effective processing of such large numbers of spots demands that a spot finder be highly accurate without requiring visual inspection of most spots by a human investigator. To automate detection of spot finding errors and spots of poor quality, current implementations require the user to specify explicit thresholds of various attributes, such as brightness, which separate acceptable from unacceptable spots. Choosing good thresholds manually for multiple attributes is time-consuming and may not achieve the desired result. A better solution is to have the spot finder automatically *learn* the investigator's concept of when a found spot is correct and of acceptable quality.

This paper describes **Dapple**, a new program for finding and quantitating spots on microarray images which departs from previously described implementations in two ways. First, Dapple's spot finder takes advantage of consistent spot morphology (i.e. circularity) to increase its robustness both to image noise and to variability in spot position and size. Second, the program learns the investigator's concept of spot quality by example, using a classifier which can be trained on manually classified examples of the spot finder's output. Once trained, the classifier is used to direct the automated spot finder to reprocess parts of the image on which the spot finder has likely erred. We have found these methods effective in constructing a practical solution to the spot finding problem.

The rest of this report is organized as follows. We first specify the spot finding problem in more detail and discuss previous spot finding implementations for microarrays. We then describe Dapple's methods and show results which demonstrate their effectiveness. Finally, we briefly recount our practical experience with Dapple and conclude with suggestions for future improvements to our methods.

## 2 The Spot Finding Problem

Dapple was developed to quantitate images from comparative gene expression experiments using two-color fluorescently labeled cDNA microarrays on glass. The arrays used to develop Dapple were produced using the Molecular Dynamics Generation II arraying system; subsequently, we have successfully applied the program to higher-density MD Generation III arrays as well as arrays produced with other equipment. Below, we briefly review the characteristics of the original Generation II arrays, which except for lower spotting density are typical of the arrays used today.

A microarray is a collection of DNA spots deposited on the surface of a glass slide. Each spot contains many copies of a single DNA sequence such as a gene. In a comparative gene expression experiment, the array is incubated with two *cDNA probes*, each of which is a mixture of cDNA's derived from the expressed mRNA of a distinct cell population. Each probe is labeled with a different fluorescent dye. Labeled cDNA molecules hybridize to spots on the array containing their complementary sequences, in numbers proportional to their concentrations in the probe populations. After hybridization, the amount of bound, labeled cDNA on each spot is inferred from the amount of fluorescence emitted when the spot is stimulated with a laser. Typically, we are interested only in the *ratio* of cDNA concentrations in the two probes for each spotted sequence, since the mea-

sured fluorescent intensities are not calibrated to absolute amounts of DNA.

In practice, an entire slide is scanned by the laser to produce a large *array image* which contains the images of all the spots on an array, laid out in rectilinear *grids*. The spots occupy a relatively small fraction of the image area, so they must be individually found and isolated from the image background prior to quantitation. This spot finding problem is generally solved as follows: identify the locations of all grids on the array image; subdivide each grid into small rectangles containing at most one spot each; and finally locate the spot, if any, in each rectangle. We refer to the image rectangles containing individual spots as *vignettes*.

A single microarray produced by the Generation II system occupies an area of approximately $6\times1$ cm, containing up to 1536 DNA spots organized in six $16\times16$ grids. Inter-spot separation within each grid is on the order of 500 $\mu$m. More recent systems increase the spotting density and number of spots per array by a factor of three to ten. The fluorescent intensities for each dye are measured separately, producing a two-channel image. Intensities are scanned at a linear resolution of ten microns per pixel, with a dynamic range of roughly $10^5$.

### 2.1 Sources of Variation and Noise

The major sources of uncertainty in spot finding on microarrays are variable spot size and position, variation of the image background, and discrete image artifacts.

Spots vary significantly in size and position within their vignettes despite the use of precise robotic tools to lay them out. The six grids on each array are spotted simultaneously by repeatedly dipping six pins, each in a solution containing a different DNA sequence, and transferring drops of solution from the pin tips to the slide surface. The relative placement of adjacent grids is therefore determined by the spacing between adjacent pins, which may vary by 100 $\mu$m or more. The pin laying out any one grid is slightly offset after each spotting to separate adjacent spots; variations in the sizes of these offsets can cause inter-spot distances to vary by 30-50 $\mu$m. Differences in the sizes of transferred drops cause individual spots to vary between 30 and 150 $\mu$m in radius – the same order of magnitude as the positional uncertainties.

Identifying spots, even with position and size variations, would be easy if they were always presented against a uniform low image background. In practice, however, the natural fluorescence of the glass slide and any non-specifically bound DNA or dye molecules add a substantial noise floor to the image. This diffuse noise exhibits considerable variability in intensity both within and between vignettes. It is about 10% as bright on average as the most highly fluorescent spots but can be almost as bright as dimmer spots, making them hard to distinguish from background variations. This phenomenon is especially frustrating because dim spots often correspond to interesting genes which are unfortunately expressed at low copy number.

Microarrays are also afflicted with discrete image artifacts such as highly fluorescent dust particles, unattached dye, salt deposits from evaporated solvents, and fibers or other airborne debris. Examples of some of these artifacts are shown in Figure 1. Such artifacts appear in 10-15% of vignettes at random, even after thorough cleaning of the slide, and (as the figure illustrates) can easily be brighter and sometimes larger than nearby spots. Their heterogeneous brightness, shape, and size make them hard to detect and remove automatically, especially in the presence of spots which are themselves of variable size and brightness. Bright artifacts complicate spot finding because finders sometimes mistake them for spots.

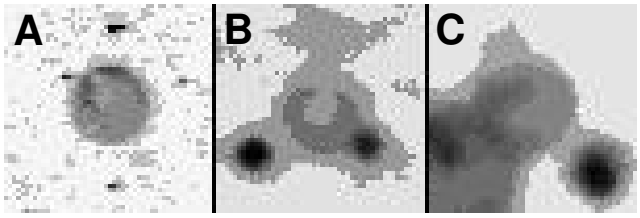


Figure 1: examples of spots affected by microarray image artifacts. Each spot is visible near the center of its vignette; darker pixels indicate higher intensities. (A) spot with bright dust particles; (B) spot with two round artifacts composed of unattached fluorescent dye; (C) spot partially obscured by a large stain. Each image is $500\times500$ $\mu$m and was extracted from an MD Generation II array image.

The need to overcome position and size variations while dealing robustly with image noise and artifacts is a principal source of complexity in solving the spot finding problem.

### 2.2 Previous Work in Microarray Spot Finders

Several implementations of spot finding have been described for microarrays, both on rigid slides and on flexible membranes. Below, we review some essential features of these implementations as compared to Dapple. Other implementations may be found in commercial array analysis packages but have not been publicly documented.

Granjeaud and colleagues implemented the HDG program [9] to quantitate radiolabeled arrays spotted on nylon membranes. Their approach is similar to previous work in locating protein spots on 2D PAGE gels, such as the Melanie II system [2]. HDG, like Melanie II, identifies candidate spots by tracing their edges. Spot morphology on membranes is quite variable, so edges may be of arbitrary shape and may even be locally concave. HDG does not use the geometry of the array to direct its search for spots; candidate spots may be found anywhere on the image in any arrangement. Only after this search are spots filtered based on a template of their expected positions, which may be warped interactively by the user to fit the image. HDG's lack of positional bias in spot finding is appropriate for membrane-based arrays because it is robust to nonlinear deformations of the membrane which commonly occur during image acquisition.

The DeArray package, described by Chen and colleagues [3], imposes a much stronger position bias on spot

finding than HDG but is similarly agnostic about spot morphology. DeArray was developed for arrays on rigid glass slides; these slides do not suffer the distortions of membranes, so the program can safely divide an array image into rectilinear grids of vignettes and process each vignette separately. DeArray's spot finder initially divides each vignette into rough spot and background regions using a circular template of fixed position and size, but it then refines the spot into an arbitrary, possibly disconnected set of bright pixels. The cutoff intensity between the bright spot and dim background pixels is chosen so that a small set of pixels sampled from the spot is significantly brighter than an equal-sized sample from the background, as measured by the Mann-Whitney-Wilcoxon test. The initial template is used only to prevent pixels near the edges of the vignette from appearing in the spot's sample during the cutoff computation.

Dapple adopts not only the positional bias appropriate to arrays on glass but also a strong morphological bias towards circular spots, which contrasts with both HDG and DeArray. We can exploit morphological information in our spot finder because our arraying system, unlike those used with the other spot finders described here, produces spots with reliably circular outlines. For our arrays, a morphology-based spot finder proved better able to cope with spot position and size variations because intensity-based segmentation often failed to correct its initial rough guess at the location and extent of a spot. Morphology also proved a more reliable means to distinguish spots from equally bright but heterogeneously shaped image artifacts.

We believe that the consistently-shaped spots necessary for a morphology-based finder are typical of most commercially produced arraying systems, though not of the popular design developed at Stanford University [6]. The difference may be that our spotting protocol dries deposited drops on the slide more slowly, minimizing known problems of uneven solute deposition [5].

Dapple also departs from previously described spot finders in its use of an example-based classifier to decide whether candidate spots have been found correctly and are usable for further analysis. Previous spot finders, including both programs described above and Eisen's ScanAlyze software [6], require the investigator to parameterize their spot classifiers explicitly, e.g. "reject all spots whose intensities are less than three times their local background." Satisfactory parameterizations are often arrived at only by an extended process of trial and error, especially when multiple image attributes are available for classification (ten attributes in the case of ScanAlyze). In contrast, Dapple's classifier is parameterized by example: the user manually classifies a training set of candidate spots qualitatively according to their perceived accuracy, after which the software adjusts its classifier to best reproduce the user's judgment as expressed on the training set. Parameterization by example, a basic technique of machine learning [11], provides a convenient and powerful way for an investigator to specify complex concepts of spot quality without explicitly determining classification thresholds for image attribute values.

# 3 Implementation of Dapple

Dapple first divides an array into its constituent grids, which are placed using the known array geometry and refined so as to roughly center spots within their vignettes. Each vignette is then analyzed in one or more passes to separate its spot from the image background. A single pass proposes a candidate spot by identifying a strong circular edge in the vignette's image. The proposed spot is evaluated based on its location and brightness; if accepted, it is kept for eventual quantitation. Otherwise, the vignette is modified to remove the rejected candidate edge and analyzed again to find a new candidate.

## 3.1 Initial Segmentation into Grids

Dapple uses the geometry of a microarray – number and relative spacing of grids, plus arrangement and spacing of spots within each grid – to divide an array image into vignettes which contain individual spots. The array's geometry is fixed by robotic spotting apparatus at the time it is spotted on a slide. The primary complexities in grid placement are that the inter-grid spacing is only approximately accurate, as discussed above, and that the actual array of spots does not occur in a fixed position relative to the image origin.

Dapple acquires a corner of the array by asking the user to click on a specified spot with the mouse. This initial placement plus the known geometry place the grids approximately correctly but may err by as much as one full inter-spot distance for some grids. Each grid is therefore individually adjusted so that its spots are on average centered with respect to their vignettes. Because spots have not yet been found, the center of each spot is approximated by its vignette's *center of intensity*, analogous to the center of mass. For an image $I$ with pixel intensities $I(x, y)$, the center of intensity in $x$ is the expectation

$$\overline{x} = \frac{\sum_{x,y} xI(x,y)}{\sum_{x,y} I(x,y)} \tag{1}$$

An analogous expectation yields the center in $y$. In practice, the center is computed only from pixels above the median vignette intensity to avoid biasing it toward the center of the vignette. We could improve our estimates of the vignette centers by iterating the process several times, computing new centers of intensity after each movement of the grids; in practice, however, one iteration of centering, combined with a fast local optimization over all small offsets of the resulting grids, is sufficient to center all grids accurately. The entire process typically requires no manual intervention after the initial click.

After initial segmentation, the image under each grid is median filtered to remove point artifacts, notably small but very bright dust particles. The filtered images are used for subsequent spot finding.

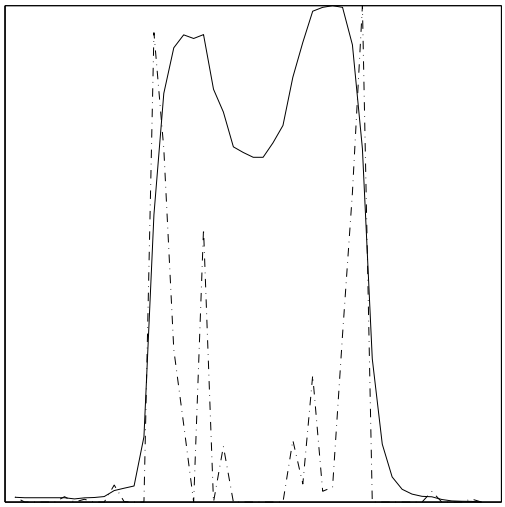

Figure 2: an intensity cross-section of a typical spot (solid line) overlaid with its negative second derivative (dashed line; only regions greater than zero are shown). The negative second derivative is maximized at the upper edge of the spot.

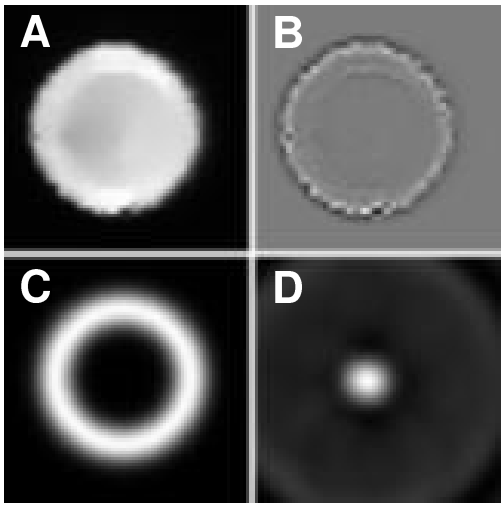


Figure 3: the spot finding process. (A) a vignette image containing a spot; (B) the negative Laplacian $L$ of the image, which highlights the spot's edge; (C) a circular filter $M_r$ ($r$ = 11 pixels) which matches the spot's radius; (D) the 2D correlation of $M_r$ at all offsets against $L$, which is maximized at the spot's center.

### 3.2 Identification of Spots

Dapple finds spots by detecting their edges. Specifically, it detects the sharp edge which generally occurs in a spot's intensity profile at the point where it flattens out after rising steeply above the image background. Figure 2 shows an intensity cross section of a typical spot which illustrates this transition. The edge corresponds to a strong local maximum in the profile's negative second derivative, shown by the dashed line.

The signed magnitude of the second derivative at every point in the vignette image is given by the Laplacian, defined for a continuous two-dimensional image $I$ as

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

We compute a standard four-neighbor second-difference approximation to the Laplacian on our discrete images. The pixel-wise negation $L = -\nabla^2 I$ of the Laplacian image exhibits a local maximum in a ring corresponding to the outline of the circular spot, as shown in Figure 3B.

A spot actually has two edges, both visible in Figure 2 – a lower edge, where it begins to rise above the background, and the aforementioned upper edge, where it flattens out at high intensity. The lower edge corresponds to a maximum of the *non*-negated second derivative and could theoretically be used for spot finding. However, we found that the lower edge was typically less sharply defined than the upper edge and so was less reliable for determining a spot's location and radius.

To detect a ring of high intensity in the Laplacian image $L$, we apply matched filtering by convolution with a series of annular filters, one of which is shown in Figure 3C. Each filter $M_r$ responds to a ring of radius approximately $r$ in $L$. $M_r$ is a ring of unit-intensity pixels on a zero-intensity background; we give this ring a Gaussian edge, which is equivalent to smoothing the image Laplacian slightly to allow for minor misregistration of the spot with the filter. The response of each filter $M_r$ is computed for each offset of the filter against the vignette image by a discrete 2D correlation. Let $n \times n$ ($n$ odd) be the common size of the image $L$ and filter $M_r$, and let $m = \frac{n-1}{2}$; then the correlation of $L$ with $M_r$ is given by

$$C_{p,q} = \sum_{i=-m}^{m} \sum_{j=-m}^{m} L(i,j) M_r(i-p, j-q) \qquad (2)$$

Here, the image center is at coordinates $(0,0)$, and $C_{p,q}$ is the *correlation score* observed when the filter is centered on pixel $p, q$ of the image. As Figure 3D shows, $C_{p,q}$ is maximized when the center of the filter is aligned to the center of the spot. The highest correlation score over all filters $M_r$ occurs when the radius $r$ most closely matches that of the spot.

Several important facts about the above procedure should be noted. First, each filter is normalized to unit energy so as not to bias the finder toward larger spots. However, the image is *not* normalized, intentionally biasing in favor of candidate spots with the strongest edges, which in practice also tend to be brightest. Second, a filter may give a strong response even to an incomplete circular edge, which provides robustness in locating partial or obscured spots. Finally, the 2D real correlation of Equation 2 may be implemented efficiently for all $p, q$ at once using a real/complex fast Fourier transform (FFT). We use the FFTW library [8], which automatically generates code for platform-optimized 2D FFT's.

Our complete spot finder requires two FFT's and one

pointwise complex multiplication per filter $M_r$ for each spot tested. For a range of ten radii over a 50×50-pixel vignette, the finder requires 0.043 seconds to process one channel of a vignette image on an Intel Pentium III/550 processor[1]. Because it treats each vignette separately, the spot finder could easily be parallelized to work on several vignettes at once.

Dapple supports both single-channel and dual-channel scanned images. When quantitating dual-channel images, we find spots in each channel of the image separately. Candidate spots from the two channels are averaged to yield a single spot only if they are determined to be of the same quality by our classifier; otherwise, the higher-quality spot dominates. We could instead find each spot in the sum of the two channel images, which would halve the amount of computation required but could potentially obscure spots which are visible in only one channel.

Our spot finding algorithm permits multiple passes over the same vignette. If a candidate spot is found to be of poor quality, the pixels it covers may be set to zero in the Laplacian image, removing its edge from further consideration. The search procedure may then be run again on the modified Laplacian image to propose a new spot. We chose to search multiple times rather than store multiple high-scoring candidates from one search because the top few correlation scores usually specify slightly offset versions of the same putative spot. The additional candidates are rarely useful if the highest-scoring candidate is rejected.

### 3.3 Classification of Spots by Quality

Dapple classifies candidate spots by quality to identify mistakes made by the spot finding algorithm as well as vignettes with marginal or nonexistent spots. The classifier is intended to reflect a human investigator's qualitative judgment, so it assigns the classes *Accept* and *Reject* to spots that would respectively be accepted for quantitation or rejected as wrong or unsuitable if inspected visually by the user. Such a classifier can be trained on a collection of candidate spots which have been manually accepted or rejected.

Our classifier uses two image features, which we call the *b-score* and *p-score*, to determine a spot's class. The b-score and p-score respectively measure a spot's **b**rightness and its **p**osition, i.e. its distance from the center of its vignette. We chose these two features from a larger set of candidates based on their consistent selection by a decision-tree learning package [12] trained on a collection of hand-classified spots. High b-scores and low p-scores were found to correlate strongly with manual spot acceptance.

For a spot $s$, the precise definitions of the b-score and p-score are as follows:

- Let $m$ be the median intensity of all pixels inside $s$, and let $F$ be the cumulative distribution function of all pixel intensities in the vignette which are not in $s$. Then the b-score of $s$ is given by $F(m)$, the fraction of background pixel intensities less than $m$.
- The p-score is the Euclidean distance between the center of $s$ and the center of its vignette, normalized by the vignette's shortest dimension.

[1] At higher densities, spots are typically smaller; processing one channel of a 28x28 spot takes roughly 6.7 milliseconds on the same processor.

We improve the utility of Dapple's classifier by introducing a third class called *Show*, intermediate in quality between *Accept* and *Reject*. Spots which fall in class *Show* are not rejected but are flagged as questionable for visual inspection by the user. The utility of class *Show* stems from the following observation about the classification loss function $\mathcal{L}$ used by human investigators: the cost in user time and effort of checking and correcting a questionable spot by eye ($\mathcal{L}$(*Show* | *Accept* or *Reject*)) is typically less than the costs $\mathcal{L}$(*Accept* | *Reject*) and $\mathcal{L}$(*Reject* | *Accept*) of letting the program silently accept or reject a spot incorrectly. *Show* spots, provided that their number is small, can thus be a reasonable and even desirable alternative to false acceptance or rejection. We implement this observation by explicitly specifying unequal costs for different types of classification error during training.

Dapple's classifier topology is shown in Figure 4. This topology is similar to several proposed by the aforementioned decision tree learner which were highly accurate on the training data and performed well under cross-validation. The classifier has four real parameters, indicated by the labeled lines in the figure: values $b_h$ and $b_l$ determine the class boundaries along the b-score axis, while values $p_l$ and $p_h$ determine the corresponding boundaries on the p-score axis. The topology shown here is hard-coded into the program, but the four parameters may be retrained by the investigator from a new training set of candidate spots, which can easily be produced using Dapple's built-in facility for gathering training data from arrays. A fixed classifier topology with variable parameters is an engineering tradeoff which minimizes implementation complexity while still allowing users to modify the classifier to handle changing spot properties arising from different arraying protocols.

We train the classifier by choosing values for its four parameters that minimize the total cost (according to the loss function $\mathcal{L}$) of any errors it makes on the training set. The user supplies $\mathcal{L}$ by assigning real-valued costs to each type of misclassification. By increasing or decreasing the cost of classifying a spot as *Show*, an investigator may trade off the number of spots marked for inspection against the classifier's accuracy on the remaining spots.

Naïvely, optimizing the classifier's parameters exactly with respect to $\mathcal{L}$ requires $O(n^4)$ time for $n$ training examples, since the examples' attribute values determine $O(n^2)$ possible values of the two separating lines for each of two attributes. In fact, by exploiting a well-chosen decomposition of the objective function, we can reduce training time to $O(n^2)$ while using only $O(n)$ space. This algorithm is described in more detail in the Appendix. In practice, we can train the classifier in under one second even with more than 1500 examples.

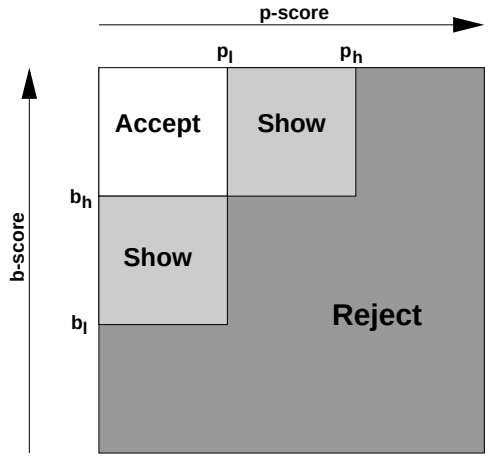


Figure 4: the topology of Dapple's classifier, shown as a division of the plane whose coordinates are the two image attributes b-score and p-score defined in the text. The four parameters labeling the region boundaries are chosen to optimize classification accuracy on a training set.

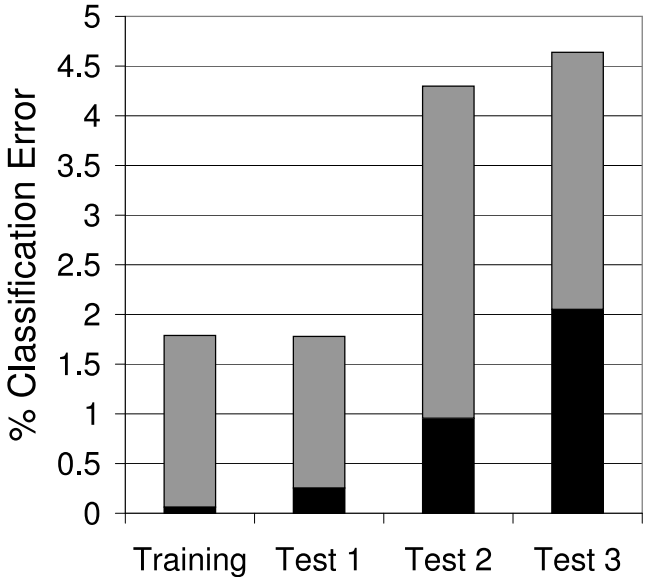


Figure 5: error rates of automated classifier *vs* a manually classified standard on the training set and three test sets. The black portion of each bar denotes candidate spots incorrectly classified as *Accept* or *Reject*; the gray portion, spots classified as *Show*.

## 4 Results

We tested Dapple's performance on a collection of six microarrays, each produced by spotting amplified sequences from 1536 open reading frames of the yeast *S. cerevisiae*. The arrays were hybridized in three separate experiments using cDNA probes made with mRNA from three different yeast strains. Each experiment used two arrays, giving a total of 4608 expression measurements made in duplicate. All arrays showed a large range of spot intensities, including many vignettes with no observable signal, as well as a variety of image artifacts.

### 4.1 Classifier Accuracy *vs* Manual Classification

We trained Dapple's classifier by manually assigning qualities to candidate spots from 512 vignettes taken from one of our arrays. For each image channel of each vignette, we visually inspected and manually classified spots proposed by our finder until either a proposed spot was accepted or three spots were proposed and rejected. The resulting training set contained 1621 candidate spots, of which 45% were accepted and the rest rejected.

We used the training set to parameterize Dapple's classifier as described. The classification loss function $\mathcal{L}$ implemented the strategy outlined above, penalizing incorrect acceptance or rejection an order of magnitude more strongly than classifying a spot as *Show*. We assigned the largest cost to acceptance of a putative spot that was manually rejected. False acceptances were considered the most serious errors because we had no way to detect them after quantitation and had difficulty finding them by eye among many accepted spots. Under different conditions, we might instead have chosen to penalize false rejections more strongly than false acceptances; for instance, if each spot were replicated several times, ratios from false acceptances could probably be discarded as outliers, but false rejections would provide no information at all.

To test whether the classifier could accurately reproduce manual spot classifications, we manually classified candidate spots from three sets of 256 vignettes, each drawn from a different experiment. Our test sets respectively contained 788, 1047, and 927 candidates. Test set 1 was drawn from the same experiment (but not the same array) used for training, while test sets 2 and 3 were drawn from the other two experiments.

Figure 5 shows the classifier's accuracy on its training set and each of the three test sets. On all three test sets, Dapple's automated classification matched our manual classification for more than 95% of candidate spots. Of the spots which were classified differently by Dapple *vs* manual classification, Dapple classified more than half as *Show* (top portion of each bar in the figure); hence, the actual rate of false acceptance or rejection was even lower – no more than 2.05% in the worst case. These results indicate that Dapple's classifier can indeed learn and accurately reproduce visually based spot quality decisions, and that the *Show* class, combined with a judiciously chosen classification loss function, is effective in reducing the number of spots that are incorrectly accepted or rejected outright.

The classifier performed best on test set 1, whose spots most closely resembled those of the training set. The errors observed for test sets 2 and 3, though more frequent, were still relatively rare overall, which suggests that overtraining of the classifier was not a serious problem. The number of spurious acceptances and rejections in these sets was kept

low by the *Show* class, which widened the classifier's margin between *Accept* and *Reject* and so improved its generalization performance on these classes.

Of the thirty-one candidate spots in our test sets which were misclassified and were not marked *Show*, only five were manually rejected spots that were incorrectly accepted. We manually rejected these objects based on their small size and, in some cases, because they were clearly part of a large, irregularly-shaped artifact. The twenty-six incorrectly rejected spots were harder to characterize, but most were of near-background intensity or occurred in vignettes with artifacts bright enough to bias the spots' b-scores far downwards. These errors reveal our classifier's limits, but they seem rare and heterogeneous enough that adapting the classifier to detect them would likely be difficult without harming its ability to generalize to new arrays.

### 4.2 Design of Overall Performance Test

We tested Dapple's spot finder and classifier together to determine whether our methods are capable of accurate spot finding. To investigate overall accuracy, we developed a testing scheme based on measuring the quantitative consistency of ratios obtained from pairs of identical spots.

To quantitate a spot, we estimated the ratio $r$ of fluorescent intensities in its two channels by the formula

$$r = \frac{\mu_{f1} - m_{b1}}{\mu_{f2} - m_{b2}}$$

Here, $\mu_{fi}$ is the mean intensity of the foreground (all pixels inside the spot) for image channel $i$, while $m_{bi}$, the estimated background level, is the median intensity of all pixels in the vignette which are at least five pixels away from the spot. Our background estimate attempts to find the mean background intensity excluding biases from bright artifacts and any residual intensity from the outer edge of the spot. After excluding these biases, we observed that background pixel intensities are distributed roughly symmetrically, so that the median robustly estimates the background mean.

The best quantitative test of Dapple's performance would be to measure its absolute error in quantitating spots whose intensity ratios were known accurately in advance. We did not have arrays of such spots available to us, so we chose instead to measure the *consistency* of Dapple's quantitated ratios on pairs of duplicate spots, i.e. spots made with the same DNA and hybridized against the same probes during a single experiment. The two spots of a duplicate pair ideally should yield the same ratio, though in practice some variation is introduced by errors in spotting and by varying local conditions of the slide surface and hybridization bath. Our six test arrays contained 4608 pairs of duplicate spots.

Our measure of ratio consistency on a pair of spots was the *Z-score*, which is defined for two ratios $r_1$ and $r_2$ as follows:

$$Z(r_1, r_2) = \frac{|r_1 - r_2|}{r_1 + r_2}$$

Two identical ratios yield a $Z$-score of 0, while large absolute differences between ratios produce $Z$-scores which asymptotically approach 1. The $Z$-score is the same (up to a constant factor) as the standard error $\frac{\sigma}{\mu}$ applied to a sample of size two.

To establish baseline estimates of quantitative consistency, we implemented the following simple spot finding algorithm: using the sum of the two image channels, find the vignette's center of intensity as defined by Equation 1, then draw a circle of constant radius around that center. This baseline finder served two purposes. First, it tended to perform well on bright spots with no artifacts, so we were able to check Dapple's consistency against that of the baseline finder on such spots. Second, the baseline finder is essentially the procedure implemented by at least one other spot finding package used in our laboratory, so any evidence that Dapple improves on the baseline was of interest to us.

To control for the effects of initial grid placement, median filtering, and quantitation, we implemented the baseline finder as a drop-in replacement for Dapple's finder while keeping the other steps of the computation the same. We used a uniform radius of nine pixels for the baseline finder, which was large enough to cover the majority of most spots without including large areas of the image background. Where it was necessary to assign independent qualities to spots produced by the baseline finder, we accepted any spot which had a mean intensity more than one standard deviation above the background mean (estimated as above by $m_{bi}$) in each of the two image channels. This procedure also reflects the methods of other spot finding packages previously used in our lab.

### 4.3 Results of Overall Performance Test

We investigated three aspects of Dapple's performance versus the baseline finder: consistency on pairs of high-quality spots, rate of false positives, and rate of false negatives. False positives are alleged high-quality spots which are actually spot finding errors, while false negatives are actual spots which are not found. Results from all these tests suggest that Dapple performed at least as well as, and often better than, the baseline finder on our arrays.

#### 4.3.1 Consistency on High-Quality Spots

To test Dapple's consistency on high-quality spots, we selected all pairs of spots for which neither spot in the pair was rejected (in either channel) by Dapple's classifier. There were 2933 such spot pairs across our three pairs of duplicate arrays. Figure 6 compares the $Z$-scores of all these pairs as found by Dapple to the corresponding scores when the pairs were found using the baseline finder. The spot finders were roughly equivalent in terms of consistency; however, Dapple demonstrated a somewhat better median consistency ($Z$ = 0.101, versus 0.110 for the baseline finder). To determine if the observed difference in consistency was significant, we compared the $Z$-scores from the two finders using the Mann-Whitney-Wilcoxon test [10], a non-parametric statistical test to decide whether the values from one distribution are on average higher or lower than those from another. We found that Dapple's $Z$-scores were significantly lower on average than

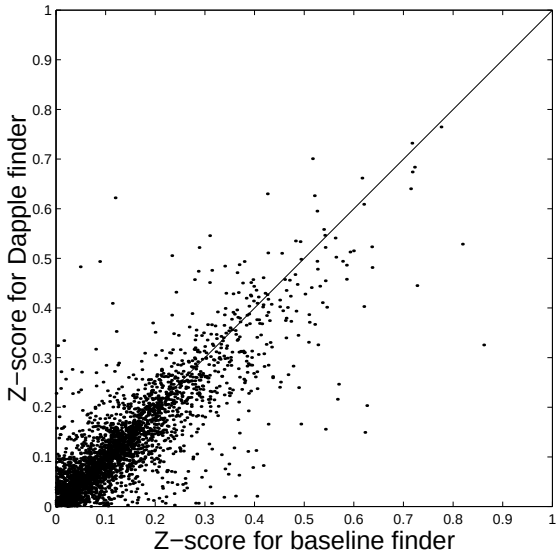


Figure 6: comparison of consistency scores for pairs of spots accepted by Dapple's spot finder. Overall, Dapple's $Z$-scores were significantly better than those of the baseline finder ($p$ = 0.0034, $N$ = 2933).

those of the baseline finder ($p$ = 0.0034, $N$ = 2933), indicating that Dapple's spot finder achieved slightly better overall quantitative consistency.

Dapple's improvement in consistency over the baseline finder may be attributed to two sources. First, our spot finder's ability to pick accurate spot sizes allowed it to cover larger spots completely, while the baseline was restricted to predicting spots with a smaller fixed radius. Having more pixels from which to quantitate reduced the variance of estimated intensity ratios. Second, we observed that the baseline finder frequently placed spots off-center with respect to the true spot, usually because the vignette contained diffuse noise or an artifact that biased the center-of-intensity computation. Occasionally, image noise or an artifact caused the spot reported by the baseline finder to be completely disjoint from the visible spot. Dapple's spot finder, which was designed to be resistant to such noise, did not produce significant numbers of off-center spots on our test arrays, largely because such spots were caught and corrected by the classifier whenever they were proposed.

Our results must be presented with the caveat that *consistent* quantitation is not the same as *accurate* quantitation: consistent ratios could still be consistently incorrect. We tested the consistency of our spot finder only because we had no way to test its absolute accuracy. However, three observations suggest that Dapple's improved consistency is not the result of systematic error. First, both Dapple and the baseline finder were evaluated with the same quantitation method, so any observed differences must be due to the two methods' differing choices of spot sizes and locations. Second, we found that Dapple's choices of spots were visually accurate in both position and size, so its improved consistency does not reflect systematically incorrect spot finding. Finally, the intensity ratios measured for the spots found by Dapple and the baseline finder were generally in close agreement, providing additional evidence that Dapple did not cause large systematic differences in quantitation. For the 5688 vignettes included in our consistency test, 93% of spots found by Dapple and the baseline finder had ratios which agreed to within 20%, while 80% of vignettes gave spots whose ratios agreed to within 10%.

#### 4.3.2 Analysis of Potential False Positives

As Figure 6 shows, Dapple did accept a number of spot pairs which had poor consistency (high $Z$). To investigate why these spots showed such high variability, we visually inspected the subset of our 2933 pairs whose Dapple-derived ratios gave $Z$-scores greater than $\frac{1}{3}$, corresponding to more than a factor of two difference between the ratios in each pair. This subset contained 151 spot pairs.

Of the spot pairs investigated, eleven (7%) contained clear false positives: one of the two spots was in fact an image artifact, such as unattached dye or DNA, that was too bright and too well-centered to be rejected. These false positives are consistent with the observed limitations of Dapple's classifier. The remaining 140 spot pairs contained two visually apparent spots which Dapple found correctly. A few of these spots had overlying artifacts that could account for their deviant ratios, but most were likely either too close to background intensity for accurate quantitation or were affected by experimental variation.

Eighty-eight of the 151 pairs inspected (58%) contained at least one spot which had been flagged for visual inspection (i.e. had at least one channel of class *Show*). These spots were usually flagged because of low intensity relative to background, so a conservative investigator might have chosen to discard them before quantitation.

#### 4.3.3 Analysis of Potential False Negatives

Approximately 1% of all vignettes tested contained objects which we believe to be spots that were not found by Dapple's spot finder. The majority of these probable false negatives did not follow our arrays' normal circular spot morphology; the rest were in vignettes with unusually bright artifacts and dim spots. In all of these cases, Dapple claimed that the vignette had no spot because its spot finder never proposed the correct spot to its classifier. These false negatives illustrate the limits of morphology-based spot finding on our arrays.

The baseline finder was better able to locate spots with unusual morphologies when they were the only objects in their vignettes; however, its performance was limited by a tendency to be confused by image noise. We inspected twenty-six pairs of vignettes whose candidate spots were both accepted by the baseline finder but for which Dapple rejected all candidates in at least one vignette. Only four of these pairs yielded examples where the baseline finder was able to detect a spot which Dapple had missed; in the other twenty-two cases, at least one spot claimed by the baseline finder was in fact an image artifact. Thus, most of these potential false negatives for Dapple turned out instead to be false

positives for the baseline finder.

## 5 Experience with Dapple in the Lab

After conducting the experiments outlined above, we began using Dapple to quantitate Molecular Dynamics Generation III arrays, which are spotted at roughly three times the density of Generation II arrays. Our spot finder continued to perform well on the higher-density arrays because the typical spot morphology remains unchanged. Our classifier remained accurate after we retrained its parameters using examples taken from the new arrays. More recently, we have adapted Dapple to work with a custom-designed arrayer built by Steve Laskey of the Institute for Systems Biology, which uses an ink-jet printer head to deposit spots on a slide.

User responses to Dapple's schemes for example-based training and classification have been positive. Investigators have reported that an hour invested in providing examples to train the spot classifier on one array produces quite satisfactory classifications for later arrays. The classification process, in conjunction with a graphical interface which indicates spot qualities as colored flags next to each spot found, is effective in limiting the amount of time spent inspecting an array for defects and spot finding errors.

A surprisingly popular feature of our interface is the ability to manually redraw the boundaries of individual spots which were found incorrectly. Although this feature is used only infrequently, investigators often cite it as one of their favorite things about Dapple. Based on this feedback, we encourage authors of other spot finding software to provide easy manual correction capabilities to their users.

Our experience with Dapple has pointed to three practical shortcomings which have been or should be addressed. First, the MD Generation II arrayer turned out to be unusually precise in aligning the two channels of a two-color array image. More recent arrayers often produce misregistered channel images, which causes difficulties when we try to combine the images for grid placement and spot classification. In response to this problem, we have modified Dapple's user interface to allow manual correction of any misregistration. Future work should aim to correct misregistration automatically.

A second difficulty observed as array densities have increased is the problem of correctly placing grids on the array image. Dapple's automated grid placement does not account for large-scale spatial variations in background intensity, with the result that a large bright streak on the array can occasionally cause grids to be misplaced by up to a full spot width. We are investigating how to account for such defects without substantially harming the performance of the grid placement engine. It may be that for high throughput arraying, the best solution is for the arrayer to place bright guide spots at the corners of each grid, thereby making grid placement a trivial problem.

Finally, Dapple's practice of using spot position in classifying quality may need to be abandoned at very high spot densities. In high-density arrays, the same absolute variations in a spot's position are much larger relative to the vignette size; thus, a candidate spot's distance from its vignette center (i.e. its p-score) conveys less information about its quality. While this problem does not invalidate the success of our overall classification approach, it suggests that future work should include a search for image features which are more informative than p-scores at higher densities.

## 6 Conclusions and Future Directions

Dapple's performance on our test data shows that it successfully implements our design considerations for spot finding in microarrays. Its spot finder is able to use morphological information to correctly identify the location and extent of most spots, even when they are of low intensity and in the presence of diverse image artifacts. Dapple's classifier can learn to replicate qualitative visual judgments about spot quality using examples instead of explicitly specified values of image attributes. The classifier can detect almost all likely failures of the spot finder, which allows it both to prevent spot finding errors from propagating to subsequent analyses and to mark a few spots of uncertain quality for inspection by the investigator. Our successful implementation of morphology-based spot finding and classification of spot qualities by example suggests that these techniques could be valuable as part of a highly automated, high-throughput system for microarray analysis.

Our spot finding techniques were well-suited to almost all spots but had difficulty with a small number of acceptable spots which did not fit the expected morphology. Such difficult spots are frequent enough to warrant the use of a non-morphology-based back-up spot finder to deal with them. We found that spots with unusual morphologies tended to be of relatively low intensity, so a sensitive intensity-based spot finder such that used by Chen *et al* would likely perform especially well in these cases. A high-throughput spot finder should benefit from using multiple, independent methods with different strengths to deal with occasional spot heterogeneity.

The design of Dapple's spot finder and classifier was guided by our desire to mimic a human investigator's ability to locate spots on an array, both because existing spot finding software encourages visual inspection of spots and because the accuracy of our methods could be assessed visually. One side effect of our design is that Dapple finds spots which can be seen but cannot be accurately quantitated, usually because they are of near-background intensity. Dapple's rudimentary quantitation scheme can no doubt be improved; however, we believe that spot finding and quality-assessment systems must ultimately be trained against a quantitative standard of accuracy, e.g. an array hybridized against standard probes with known intensity ratios. Our arraying facility has recently begun to produce such arrays in numbers sufficient for training. We anticipate that our methods will continue to work even when quantitation, rather than the investigator's eye, becomes the ultimate determinant of success.

**Acknowledgements**

We are grateful to Roger Bumgarner for maintaining our arraying facility and producing the whole-genome yeast arrays used to develop Dapple, and to Richard Karp for helpful technical discussions. Thanks are also due to the array group at the Institute for Systems Biology, particularly Steve Laskey and Rowan Christmas, for testing Dapple and providing feedback on needed enhancements. This work was supported in part by a Fannie and John Hertz Fellowship and by NIH grant #5-RO1-HG01713.

# A Appendix: Efficient Induction of Classifier

In this appendix we describe the algorithm used to choose optimal values for our classifier's four parameters $b_l \leq b_h$ and $p_l \leq p_h$ with respect to a training set $T$ of instances $t$. Each instance has a true class $c(t)$ ranging over the set $\{A, S, R\}$ and two real-valued attributes $b(t)$ and $p(t)$. The classifier assigns $t$ a class $c'(t)$ based on its attribute values. We seek parameter values which minimize the total loss

$$L = \sum_{t \in T} \mathcal{L}(c'(t) \mid c(t))$$

For simplicity, we assume that no two instances share the same value of either attribute.

WLOG, each of the classifier's four parameters can take on only the $|T|$ distinct values corresponding to the attribute values of the training instances. We can compute $L$ for all values of a single parameter in time $O(|T|)$ by treating the parameter as a separating line in the plane defined by $b(t)$ and $p(t)$: simply sweep the line over this plane, updating $L$ in constant time whenever the line passes a training instance. These observations immediately yield a $O(|T|^4)$ algorithm to compute $L$ for *every* 4-tuple of parameter values. For a reduced problem with only two parameters, the same considerations yield an $O(|T|^2)$ enumeration algorithm.

We now derive a minimization algorithm for the full four-parameter problem that runs in time $O(|T|^2)$. To achieve this bound, we divide the objective function $L$, which depends on all four parameters, into three parts which depend on only two parameters each. We then reassemble the parts to find parameter values which minimize $L$.

Define the following functions of an instance $t$:

$$\begin{aligned} \alpha(t) &= \mathcal{L}(A \mid c(t)) \\ \sigma(t) &= \mathcal{L}(S \mid c(t)) \\ \rho(t) &= \mathcal{L}(R \mid c(t)) \end{aligned}$$

Let $\ell(t)$ be the function on instances induced by the labeled partition of the feature space shown in Figure 7, which mirrors our classifier's topology. If an instance $t$ falls in a given region of the partition, $\ell(t)$ is defined to be the function which labels that region applied to $t$. $\ell(t)$ is simply the classification loss for $t$ given the four parameter values, and $L = \sum_{t \in T} \ell(t)$.

Now consider the three functions $f(t)$, $g(t)$, and $h(t)$ induced by the three labeled partitions of Figure 8. It is not hard to see that, for any fixed 4-tuple of parameter values and any instance $t$,

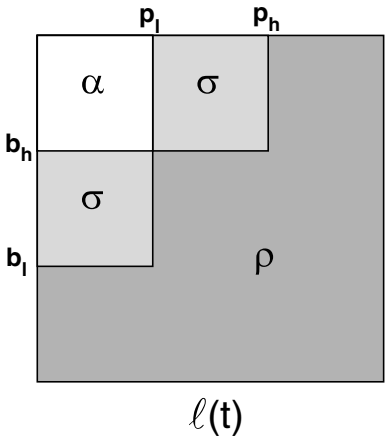


Figure 7: partition for the function $\ell(t)$, which computes the classification loss for an instance $t$ as a function of the four classifier parameters.

$$f(t) + g(t) + h(t) = \ell(t) + 2\rho(t) \tag{3}$$

Let $F = \sum_{t \in T} f(t)$, and similarly define $G$ and $H$. Summing both sides of Equation 3 over all $t \in T$ gives

$$F + G + H = L + c \tag{4}$$

where $c$ is a constant which depends only the training set, not on the values of the four parameters. Thus, by minimizing the left-hand side of Equation 4 with respect to the classifier's parameters, we also minimize the objective function $L$. The functions $F$, $G$, and $H$ depend on only two parameters each; hence, we can tabulate their *distinct* values over all 4-tuples of parameter values in time $O(|T|^2)$.

It remains to show how to combine the three functions $F$, $G$, and $H$ in quadratic time to find the global minimum of their sum over all parameter values. In what follows, we explicitly annotate functions with the classifier parameters on which they depend. We first define functions $G^*$ and $H^*$ as follows:

$$G^*(p_l, b_h) = \min_{b_l \leq b_h} G(p_l, b_l) \tag{5}$$

$$H^*(p_l, b_h) = \min_{p_h \geq p_l} H(p_h, b_h) \tag{6}$$

The $O(|T|^2)$ possible values of $G^*$ can be computed in quadratic time from the values of $G$ by the inductive rule

$$G^*(p_l, b_h) = \min\left[G(p_l, b_h), G^*(p_l, b_{prev})\right]$$

where $b_{prev}$ is the largest observed $b(t)$ less than $b_h$. A similar rule computes $H^*$ from $H$. Finally, we minimize $F + G + H$ by observing that

$$\begin{aligned} & \min_{p_l \leq p_h, b_h \geq b_l} F(p_l, b_h) + G(p_l, b_l) + H(p_h, b_h) \\ = & \min_{p_l, b_h} \left[ F(p_l, b_h) + \min_{b_h \geq b_l} G(p_l, b_l) + \min_{p_l \leq p_h} H(p_h, b_h) \right] \\ = & \min_{p_l, b_h} F(p_l, b_h) + G^*(p_l, b_h) + H^*(p_l, b_h) \end{aligned}$$

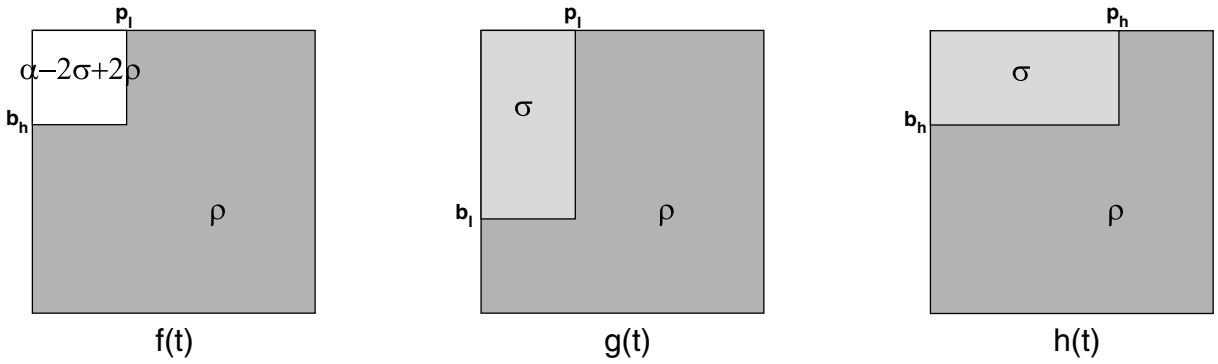


Figure 8: partitions for the functions $f(t)$, $g(t)$, and $h(t)$. For any $t$, the sum of these three functions is $\ell(t) + 2\rho(t)$.

The final minimization requires time $O(|T|^2)$ given all values of $F$, $G^*$, and $H^*$, which is sufficient to achieve quadratic time overall. The values of $p_l$ and $b_h$ are the argmin of this minimization, while the values of $b_l$ and $p_h$ are the argmins in Equations 5 and 6 respectively for the chosen $p_l$ and $b_h$.

We note that by interleaving the tabulation of $G$, $G^*$, $H$, and $H^*$ and keeping the track of the minimum-cost parameter values seen so far, we can implement the minimization with only $O(|T|)$ working space. In practice, we have found this algorithm to be extremely efficient for training Dapple's classifier even on more than 1500 examples.

## References


[1] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96:6745–50, 1999.

[2] R. D. Appel, J. R. Vargas, P. M. Palagi, D. Walther, and D. F. Hochstrasser. Melanie II – a third-generation software package for analysis of two-dimensional electrophoresis images. II: algorithms. *Electrophoresis*, 18(15):2735–48, 1997.

[3] Y. Chen, E. R. Dougherty, and M. L. Bittner. Ratio-based decisions and the quantitative analysis of cDNA microarray images. *Journal of Biomedical Optics*, 2(4):364–74, 1997.

[4] S. Chu, J. DeRisi, M. Eisen, J. Mulholland, D. Bottstein, P. O. Brown, and I. Herskowitz. The transcriptional program of sporulation in budding yeast. *Science*, 282:699–705, 1998.

[5] R. D. Deegan, O. Bakajin, T. F. Dupont, G. Huber, S. R. Nagel, and T. A. Witten. Capillary flow as the cause of ring stains from dried liquid drops. *Nature*, 389(23):827–9, 1997.

[6] M. B. Eisen and P. O. Brown. DNA arrays for analysis of gene expression. *Methods in Enzymology*, 303:179–205, 1999.

[7] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–8, 1998.

[8] M. Frigo and S. G. Johnson. FFTW: an adaptive software architecture for the FFT. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1381–4, Seattle, WA, 1998.

[9] S. Granjeaud, C. Nguyen, D. Rocha, R. Luton, and B. R. Jordan. From hybridization image to numerical values: a practical, high throughput quantification system for high density filter hybridization. *Genetic Analysis: Biomolecular Engineering*, 12:151–62, 1996.

[10] R. V. Hogg and A. T. Craig. *Introduction to Mathematical Statistics*. Macmillan Publishing Co. Inc., New York, NY, 4 edition, 1978.

[11] T. M. Mitchell. *Machine Learning*. WCB McGraw-Hill, Boston, MA, 1997.

[12] S. K. Muirthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–33, 1994.

[13] C. Nguyen, D. Rocha, S. Granjeaud, M. Baldit, K. Bernard, P. Naquet, and B. R. Jordan. Differential gene expression in the murine thymus assayed by quantitative hybridization of arrayed cDNA clones. *Genomics*, 29:207–16, 1995.

[14] C. M. Perou, S. S. Jeffrey, M. van de Rijn, C. A. Rees, M. B. Eisen, D. T. Ross, A. Pergamenschikov, C. F. Williams, S. X. Zhu, J. C. Lee, D. Lashkari, D. Shalon, P. O. Brown, and D. Botstein. Distinctive gene expression patterns in human mammary epithelial cells and breast cancers. *Proceedings of the National Academy of Sciences*, 96(16):9212–7, 1999.

[15] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns

with a complementary DNA microarray. *Science*, 270:467–70, 1995.