Supporting information for: Distributed classifier based on genetically engineered bacterial cell cultures

Andriy Didovyk,[†] Oleg I. Kanakov,[‡] Mikhail V. Ivanchenko,[¶] Jeff Hasty,[†,§,‖] Ramón Huerta,[*,†] and Lev Tsimring[*,†]

BioCircuits Institute, University of California San Diego, La Jolla CA, USA, Department of Radiophysics, Lobachevsky State University of Nizhniy Novgorod, Nizhniy Novgorod, Russia, Department for Bioinformatics, Lobachevsky State University of Nizhniy Novgorod, Nizhniy Novgorod, Russia, Department of Bioengineering, University of California San Diego, La Jolla CA, USA, and Molecular Biology Section, Division of Biological Science, University of California San Diego, La Jolla CA, USA

E-mail: rhuerta@ucsd.edu; ltsimring@ucsd.edu

[*]To whom correspondence should be addressed

[†]BioCircuits Institute, University of California San Diego, La Jolla CA, USA

[‡]Department of Radiophysics, Lobachevsky State University of Nizhniy Novgorod, Nizhniy Novgorod, Russia

[¶]Department for Bioinformatics, Lobachevsky State University of Nizhniy Novgorod, Nizhniy Novgorod, Russia

[§]Department of Bioengineering, University of California San Diego, La Jolla CA, USA

[‖]Molecular Biology Section, Division of Biological Science, University of California San Diego, La Jolla CA, USA

1 Analytical treatment of soft learning

The training procedure in the limit of very large number of training examples and very soft learning rule can be given an analytical description sketched as follows.

Let an elementary classifier be described by its output function $z(x; m)$, where x is input, m is a parameter which varies among the cells. Assume for simplicity, that m takes on a discrete set of values $m \in \{m_i\}$. Then the expectation of the overall classifier output is given by

$$\langle f(x) \rangle = \frac{1}{N_c} \sum_i \langle n_i \rangle z(x; m_i), \tag{1}$$

where $\langle \cdot \rangle$ denotes expectation, n_i is the quantity of cells having $m = m_i$.

After presenting all training examples, the quantities n_i change according to the following:

$$\langle n_i \rangle^{\text{learned}} = \langle n_i \rangle^{\text{ini}} \cdot P(m_i), \tag{2}$$

where $P(m)$ is the probability that a cell with a given parameter value m survives the whole training procedure, $\langle n_i \rangle^{\text{ini}}$ is the initial cell distribution. If it is uniform, then the composition of the population after training is determined by the "shaping factor" $P(m)$.

The training consists of independent iterations. At each iteration an individual training example from either class is presented. Due to the independence of the iterations, the probability $P(m)$ is factorized into a product of the separate probabilities $P_+(m)$ and $P_-(m)$ of surviving the selection with examples taken from the positive (respectively, negative) class:

$$P(m) = P_+(m) \cdot P_-(m). \tag{3}$$

These probabilities can be expressed as

$$P_\pm(m) = \prod_{j=1}^{N_{ex}} p_\pm(z_j)^{1/s}, \tag{4}$$

where the index j labels the training examples, N_{ex} is the number of the training examples which is assumed to be the same for both classes, z_j is the output of a cell when the corresponding example is presented, $p_{\pm}(z)^{1/s}$ are the survival probabilities for a single presentation of a positive or negative example, with s being another "softness parameter" of learning.

Controlling the softness of learning by varying s and γ is illustrated in Figure S1. Increasing s has the same qualitative effect as increasing γ, but introducing softness with parameter s according to (4) allows an easy passage to the limit of many training examples (see below).

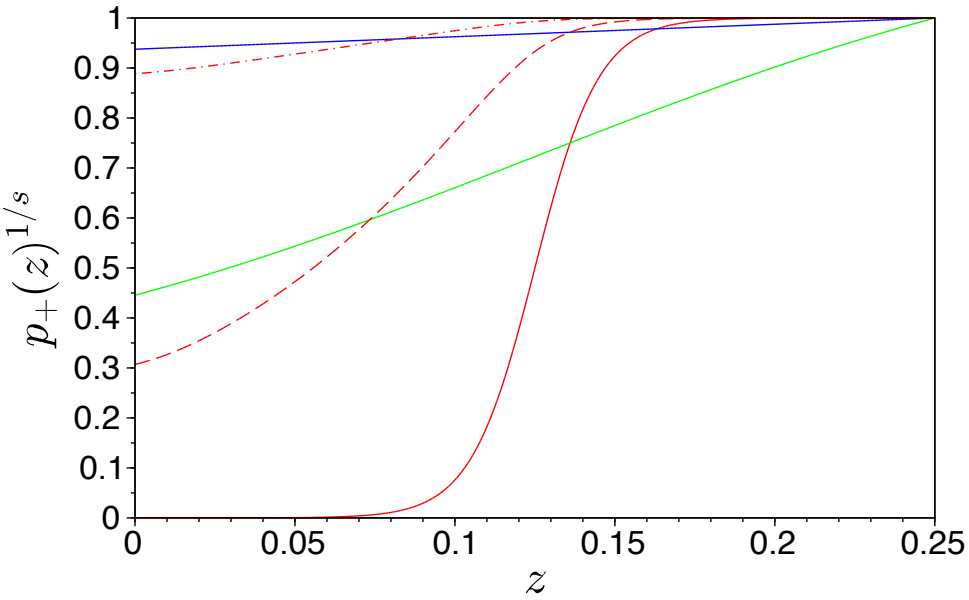

Figure S1: Red, green and blue solid lines: $\gamma = 0.01$, 0.1 and 1.0 at $s = 1$. Dashed and dash-dotted red lines: $s = 10$ and 100 at $\gamma = 0.01$.

Note, that the assumption of equal number of training examples from both classes is not essential. The calculation can be modified to describe a "weighted" training strategy, when the training examples are sampled from the classes with different probabilities. This approach can be used for creating a non-naive Bayesian classifier.

Taking logarithms from both sides of (4), we convert the product into a sum:

$$\log P_{\pm}(m) = \frac{1}{s}\sum_{j=1}^{N_{ex}} \log p_{\pm}(z_j). \tag{5}$$

Consider the limit of a large number of the training examples $N_{ex} \to \infty$. In order for the right-hand side of (5) to converge, we have to assume s to be a quantity of the same order as N_{ex}. The simplest way is to assume them directly proportional:

$$s = N_{ex}/D,$$

where D is a constant coefficient. This is reasonable, since, intuitively, the larger the quantity of the training examples is, the "softer" the learning should be. Then in the right-hand part of (5) we obtain an arithmetic mean, which will converge to an expectation integral:

$$\log P_{\pm}(m) = D \cdot \int w_z^{\pm}(z) \log p_{\pm}(z) dz, \tag{6}$$

where $w_z^{\pm}(z)$ are the probability density functions of the cell output z for the positive and negative training ensembles.

Changing the integration variable in (6) from the cell output z to the cell input x, we arrive at

$$\log P_{\pm}(m) = D \cdot \int w_x^{\pm}(x) \log p_{\pm}(z(x;m)) dx,$$

where $w_x^{\pm}(x)$ are the probability density functions of the positive and negative classes. Using (3), we write down the "shaping factor"

$$P(m) = \exp\left\{ D \cdot \int \left[w_x^{+}(x) \log p_{+}(z(x;m_i)) + w_x^{-}(x) \log p_{-}(z(x;m_i)) \right] dx \right\}, \tag{7}$$

which completely characterizes the population reshaping as a result of the training in terms of known functions.

For comparing the analytical result to simulations carried out at $s = 1$, we can formally let $D = N_{ex}$ with $N_{ex} = N_{iter}/2$. Thus, for $N_{iter} = 200$, we assume $D = 100$.

The result (7) is depicted in Figure S2 (left column) for $D = 100$, other parameters taken from the paper, for both classification problems from the paper: the two log-normal discrimination (top), and the bimodal class discrimination (bottom). The corresponding population output (1) is shown in the right column.

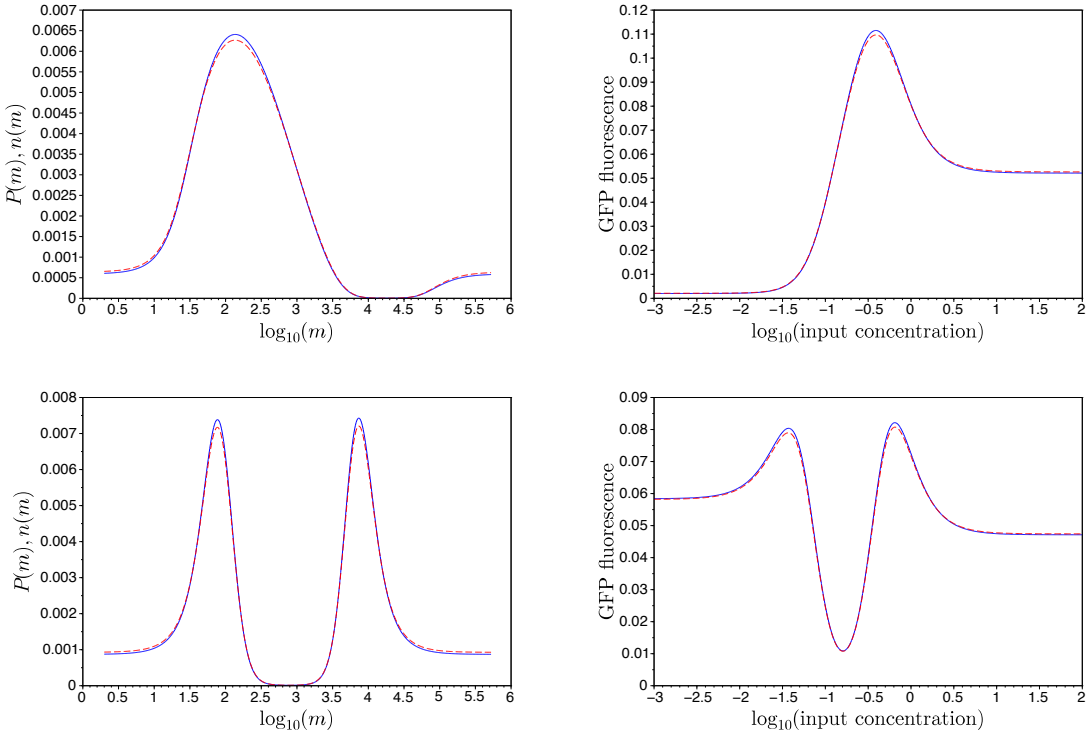

Figure S2: Left column: parameter distribution after training, right column: ensemble output. Upper row: two log-normal classes, lower row: sandwiched classes.

1.1 Training the classifier keeping the total cell count constant

In the previous section we have considered the case when the total cell count is decreasing at every training iteration. Let us modify the derivation taking into account that the total cell count is kept constant as described in the Algorithm 1 (see the main text).

Assume, before the jth training iteration the counts of the cell classes are n_i^j, index i

labels the classes. Let us find the counts at the next iteration n_i^{j+1}.

The first summand in n_i^{j+1} will be the quantity of cells which survive the jth iteration, which is $n_i^j p_i$, where p_i is the survival probability $p_i = p_\pm(z(x_j; m_i))$.

The quantity of the discarded cells is $N_d = \sum_k n_k^j(1 - p_k)$. It means, that we randomly pick N_d cells from the n_i^j population and add them to the n_i^{j+1} population. The probability for one such cell to appear in the ith class is thus n_i^j/N_c, where $N_c = \sum_i n_i$ is the total cell count. The total number of cells reinjected to the ith class is thus $n_i^j/N_c \cdot N_d$. This is the second summand in n_i^{j+1}. Thus we arrive at

$$n_i^{j+1} = n_i^j p_i + \frac{n_i^j}{N_c} \sum_k n_k^j (1 - p_k). \tag{8}$$

Note, that $\sum_i n_i^{j+1} = N_c$, so the total cell count is kept.

Let us describe the process as continuous in time. Assume, one training iteration takes a small time dt. The survival probabilities at such iteration weakly differ from 1 and can be written as

$$p_i = 1 + \lambda_i dt.$$

Then (8) turns into

$$n_i^{j+1} = n_i^j + \lambda_i n_i^j dt - \frac{n_i^j}{N_c} \sum_k n_k^j \lambda_k dt.$$

Assuming $t = jdt$ and $n_i^j = n_i(t)$, we arrive at a differential equation

$$\frac{dn_i}{dt} = \lambda_i n_i - \frac{n_i^j}{N_c} \sum_k \lambda_k n_k^j. \tag{9}$$

Here again the total cell count is conserved $\frac{d}{dt} \sum n_i = 0$.

Omitting the second term, which accounts for the cell keeping mechanism, we get a solution

$$n_i(t) = n_i(0) \exp(\lambda_i t).$$

This is formally equal to the result of the previous section, if we set $D = t$, and

$$\lambda_i = \lambda(m_i) = \int \left[w_x^+(x) \log p_+(z(x; m_i)) + w_x^-(x) \log p_-(z(x; m_i)) \right] dx. \tag{10}$$

The Eq. (9) with λ_k expressed via (10) describes the population evolution with cell count conservation enabled.

1.2 Performance

The total response from a trained classifier to the signal x before output noise is added is given by the weighted sum over all classes of cells

$$z_*(x) = \sum_j P(m_j) z(x; m_j).$$

After adding instrumental noise the output is

$$z = \max(0, z_*(x)(1 + \varepsilon) + \zeta),$$

where ε and ζ are independent Gaussian variates with means 0 and $\sigma/4$ and standard deviations $\sigma_\varepsilon = \sigma$ and $\sigma_\zeta = \sigma/4$, respectively. Then the distribution of the random output z is a truncated (at zero) Gaussian with mean $M_z = z_*(x) + \sigma/4$ and standard deviation $\sigma_z = (\sigma_\varepsilon^2 z_*(x)^2 + \sigma_\zeta^2)^{1/2}$:

$$w_z(z; x) = \frac{1}{F_0} H(z) \exp\left[-\frac{(z - M_z)^2}{2\sigma_z^2} \right],$$

where $H(z)$ is the Heaviside function and F_0 is a normalization divisor

$$F_0 = \frac{1}{2} \left(1 + \operatorname{erf}\left(\frac{M_z}{\sigma_z \sqrt{2}} \right) \right),$$

ensuring that $\int_0^\infty w_z(z; x) dz = 1$.

If a particular example x corresponds to the positive class, then the probability of a correct answer for a given threshold θ is $P^+(x|\theta) = \int_\theta^\infty w_z(z;x)dz$, whereas if this same x corresponds to a negative example, then the probability of a correct answer is $P^-(x|\theta) = \int_0^\theta w_z(z;x)dz$. These probabilities are expressed as follows:

$$P^-(x|\theta) = \frac{1}{2F_0}\left[\mathrm{erf}\left(\frac{\theta - M_z}{\sigma_z\sqrt{2}}\right) + \mathrm{erf}\left(\frac{M_z}{\sigma_z\sqrt{2}}\right)\right], \quad P^+(x|\theta) = 1 - P^-(x|\theta).$$

The mean performance (i.e. the probability of a correct answer) is then

$$P(\theta) = P(+)\cdot\int w^-(x)P^-(x|\theta)dx + P(-)\cdot\int w^+(x)P^+(x|\theta)dx,$$

where $P(+)$ and $P(-)$ are the a-priori probabilities for an example to belong to either class. When $P(+) = P(-) = 1/2$, the mean performance can be rewritten as

$$P(\theta) = \frac{1}{2} + \frac{1}{2}\int \left[w^-(x) - w^+(x)\right] P^-(x|\theta)dx.$$

Finally, we maximize the performance P over θ,

$$P_{max} = \max_\theta\{P(\theta)\}$$

and plot this value versus the number of training iterations.

2 Performance of the classifier model on the problem of discriminating toxicity data

In this section we calculate the performance of the classifier model on the problem of discriminating toxicity data (Figure 4). Positive and negative examples are drawn from the two-dimensional distribution projected onto $[Toxin\ A]$ axis. We used the following parameters: log-normal distributions for $[Toxin\ A]$ and $[Toxin\ B]$ with parameters $\sigma_t = 0.86$ and $\mu_t = ln(\sigma_t/10)$; $LC50_{Toxin\ A} = 0.1$, $LC50_{Toxin\ B} = 1.0$, where $LC50$ is a median lethal concentration of a toxin; the line of constant toxicity (isobole) is given by $[Toxin\ A] = LC50_{Toxin\ A} \cdot (1 - [Toxin\ B]/LC50_{Toxin\ B})$. The classifier circuit parameters are given in Figure 2 caption.

Using both "hard" and "soft" learning strategies the classifier achieves high median performance of about 95% after 200 iterations (Figure S3(A)). For these particular problem and classifier parameters "hard" learning with $\gamma = 0.1$ is robust: essentially maximum performance is achieved in less than 15 training iterations. The performance stays essentially unchanged throughout all 200 training iterations. "Soft" training ($\gamma = 1.0$) is noticeably slower. The classifier performance levels off only after about 100 iterations. These results are obtained for $N_c = 10^4$ cells. High classification performance after 200 training iterations is achieved for the case of "soft" learning for as little as 100 cells (Figure S3(B)). The classifier performance is robust to readout noise (Figure S3(C)). The evolution of the classifier parameters during "soft" training is shown for a particular representative training trajectory on Figure S3(D)–(I).

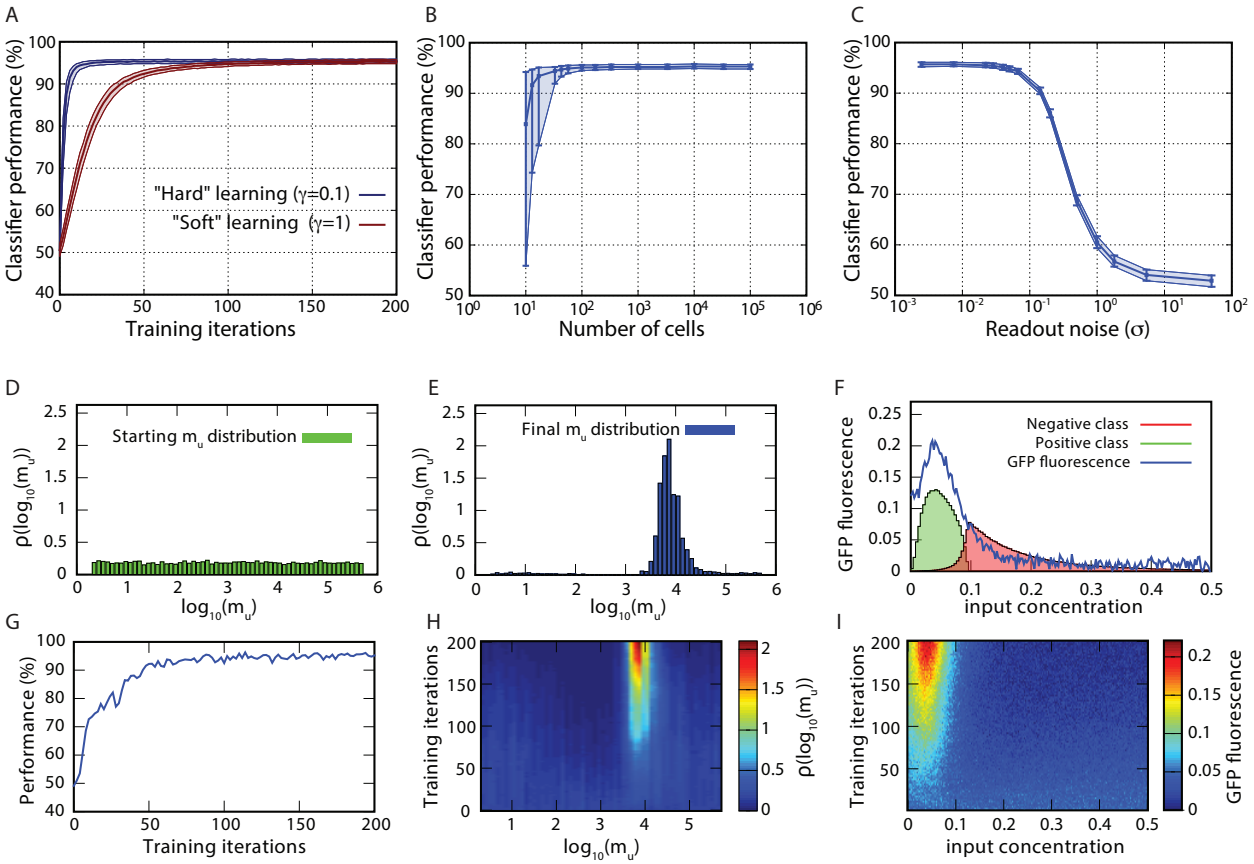

Figure S3: Classification results for the problem of discriminating toxicity data, Figure 4(B). (A) Evolution of the classifier performance for $\gamma = 0.1$ ("hard" learning; blue) and $\gamma = 1$ ("soft" learning; red), population size $N_c = 10^4$ cells. The classifier performance versus cell population size N_c (B) or GFP fluorescence readout noise σ (C); $\gamma = 1$ in (B) and (C), $N_c = 10^4$ in (C). The median and interquartile range of the distribution of the classifier performance calculated from 10^3 different stochastic realizations are shown in (A)–(C), readout noise $\sigma = 1/35$ in (A) and (B). (D)–(I) Evolution of the parameters of the ensemble of cells before and after training – an example trajectory. The parameters used are $\gamma = 1$, $N_c = 10^4$, $\sigma = 1/35$. It illustrates the shift in the distribution of parameters due to the training process of elimination of cells. The distribution of RBS/promoter strengths m_u before training (D) and after 200 training iterations (E). (F) Normalized GFP fluorescence of the ensemble of cells $f(x)$ (blue) after 200 training iterations, log-normal distribution generating positive (green) and negative (red) class examples. (G) Evolution of the classifier performance in this realization. Evolution of m_u distribution (H) and normalized cumulative GFP fluorescence $f(x)$ (I).

3 Classifier model based on experimentally derived λP_{RM} promoter response function

In this section we compare the generic Hill-function based reporter promoter response function used throughout the rest of this text with an experimentally derived λP_{RM} promoter response function.[S1] To do so we replace the Hill function based reporter response function (see the main text eq. (3)) with

$$r_g(u; m_g) = m_g \cdot \frac{1 + c(u/A_g)^2 + a s_1 c^2 (u/A_g)^4}{1 + c(u/A_g)^2 + s_1 c^2 (u/A_g)^4 + s_1 s_2 c^3 (u/A_g)^6} \tag{11}$$

where $a = 11$, $c = 0.0165$, $s_1 = 2$, $s_2 = 0.08$, $A_g = 4/3$, $m_g = 1500/46048.7$. The values of A_g and m_g were chosen to agree m_u, x, and z scales with the corresponding scales of the generic reporter promoter model (see the main text for the notation). All the other parameters of the circuit model were left unchanged.

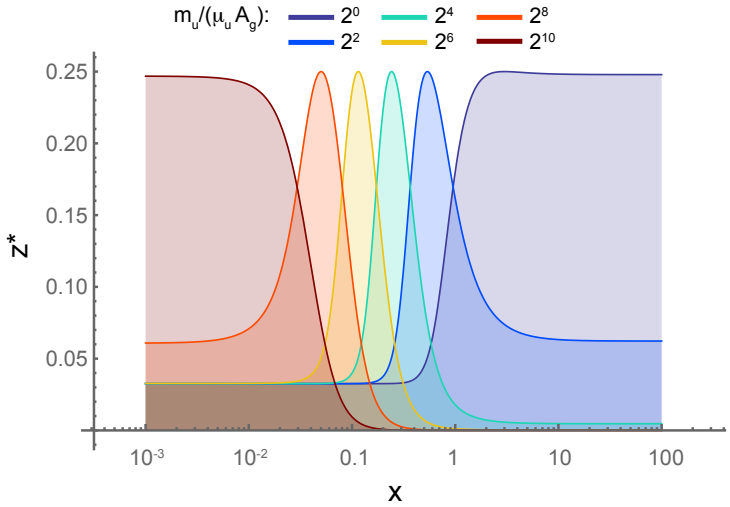

Figure S4: Combined response of the classifier circuit based on the experimentally derived λP_{RM} promoter response function: steady state GFP concentration (z^*) as a function of the concentration of the external chemical signal X for the modular classifier circuit shown for a range of m_u values representing a range of the relative strengths of the sensor promoter (see Figure 1).

The resulting combined response $z^*(x; m_u, m_g)$ (see the main text eq. (6)) of the λP_{RM}

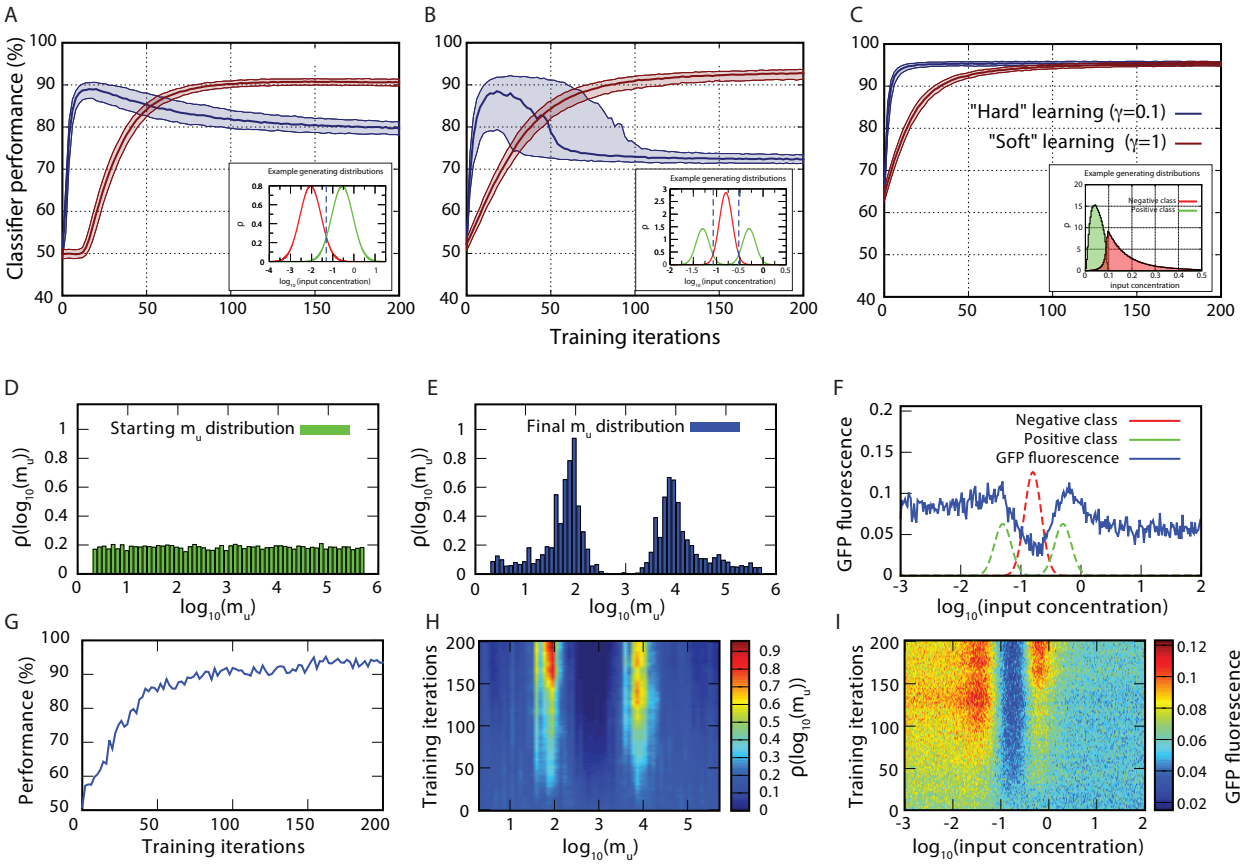

Figure S5: Performance of the model of cell population based classifier based on experimentally derived λP_{RM} promoter response function. (A)–(C) Evolution of the classifier performance during "hard" and "soft" training for the three classification problems discussed in the main text (see the corresponding figure insets). Classification data and simulation parameters are the same as those used for the Hill-functions based classifier model (see Figures 6, 7, S3). (D)–(I) The results from a single representative "soft" training trajectory are shown for the problem of discriminating data generated from a unimodal negative class sandwiched by a bimodal positive class using the λP_{RM} response promoter based classifier model. See Figure 6(D)–(I) for the corresponding captions.

based model is very similar to that of the generic Hill-function based model (compare Figure S4 and Figure 2).

We have calculated the classification performance for the three classification examples considered in the main text using the λP_{RM} based model (Figure S5). All the simulation parameters were kept the same as for the generic model. For the classification problems and parameters tested the λP_{RM} based classifier model performs comparably to the Hill-function based model (compare Figure S5 and Figure 6, 7, S3). Specifically, λP_{RM} based classifier

model somewhat underperforms on the problem of discriminating two log-normal classes and has a comparable or slightly better performance on the other two problems. This may be due to the significant non-zero bias of the combined response function $z^*(x; m_u, m_g)$ for small values of x (compare Figure S4 and Figure 2) thus skewing the total GFP fluorescence response (main text eq. (1)) towards the lower values of the input concentration x.

4 Effect of non-uniformity of m_u distribution in "master population"

In this section we investigate the effect of non-uniformity of the distribution of control parameter m_u in the "master population" on the performance of the classifier model. We used the classifier model and performed the simulations using soft learning ($\gamma = 1$) for all three classification problems exactly as described in the main text with the only exception of starting with a unique non-uniform m_u distribution per each stochastic training/evaluation trajectory (Figure S6). The initial non-uniform m_u distribution was constructed in the following way. The interval of $log_2(m_u) \in [1, 19]$ (the same as for the uniform m_u distribution in the main text) was split into 20 equal bins. Each bin was assigned probability density of $\rho_i = Y/\Omega$, where Y is a log-normal random variable with parameters $\mu = 0$ and σ. $\Omega = w * \sum_i \rho_i$, where $w = 18/20$ is the width of each bin on log_2 scale. A histogram of 10^4 m_u values drawn from a particular realization of such a distribution with $\sigma = 1$ is shown on Figure S6(D). The performance of the classifier after 200 iterations of "soft" learning was calculated for a range of $\sigma \in [0, 2]$. 10^3 stochastic trajectories were calculated each starting with a unique m_u distribution for each non-uniformity parameter σ.

As a result, for the chosen parameters the model of the cell population based classifier is robust with respect to the non-uniformity of the m_u distribution in "master population" in the entire range $\sigma \in [0, 2]$ on the problems of discriminating the data drawn from two log-normal distributions (Figure S6(A)) and the two-toxin dataset (Figure S6(C)). For the same parameters the performance of the classifier model is noticeably more sensitive with respect to σ on the problem of discriminating data drawn from a log-normal distribution of negative examples sandwiched with a bimodal distribution of positive examples (Figure S6(B)). A sample learning trajectory for this case and $\sigma = 1$ is shown on Figure S6(D)–(I).

The above calculations were performed in order to determine whether it's reasonable to expect that the published experimentally constructed expression libraries are uniform

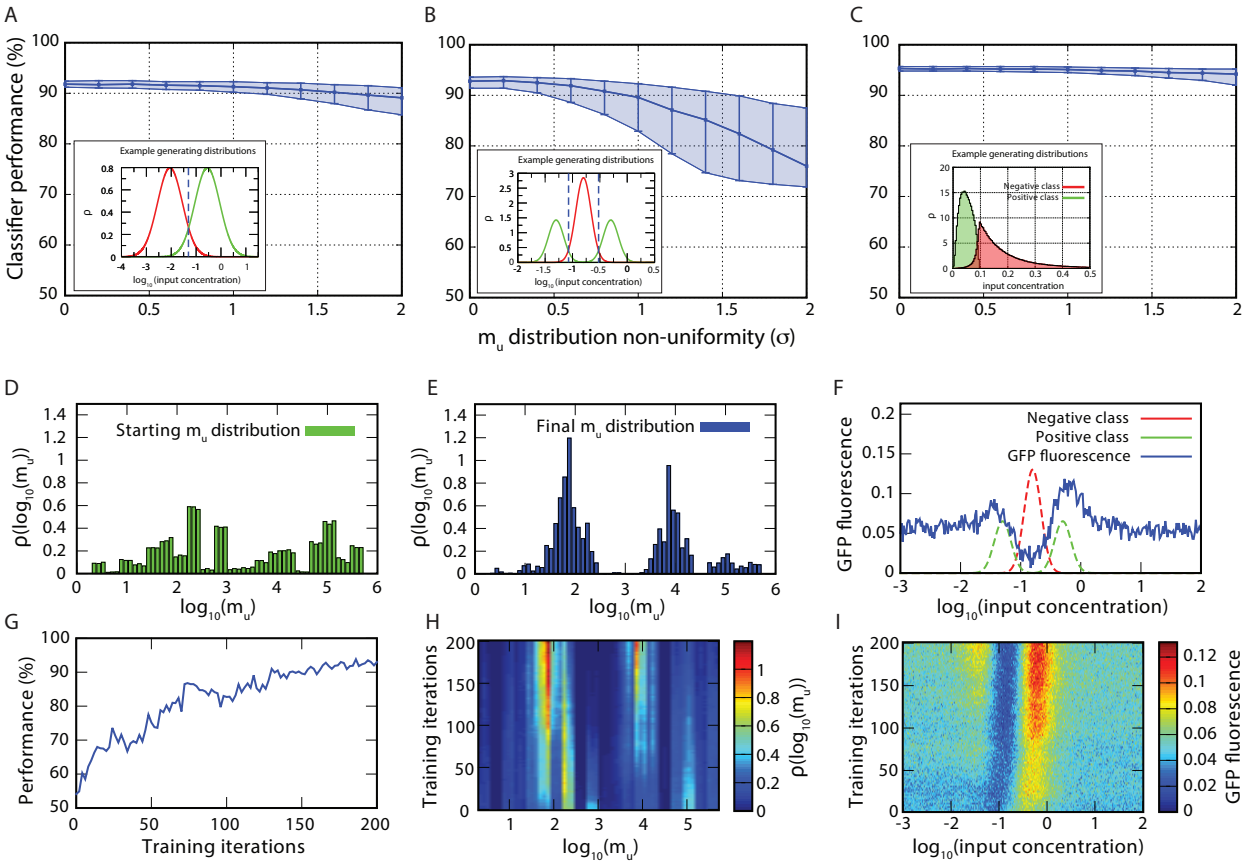

Figure S6: Effect of non-uniformity of m_u distribution in "master population" on classifier performance. (A)–(C) Evolution of the classifier performance during "hard" and "soft" training for the three classification problems discussed in the main text (see the corresponding figure insets). Classification data and simulation parameters are the same as those used in the main text (see Figures 6, 7, S3). (D)–(I) The results from a single "soft" training trajectory are shown for the problem of discriminating data generated from a unimodal negative class sandwiched by a bimodal positive class. See Figure 6(D)–(I) for the corresponding captions.

enough in order to use them to produce "master population" for successful training of the proposed cell population based classifier. An algorithm that allows predicting the levels of translation from ribosome binding sites with about 2.3-fold accuracy over the range of 10^5 has recently been developed and experimentally validated.[S2] Similarly an algorithm capable of predicting protein and mRNA expression levels from promoters with computationally constructed *E. coli* RNA polymerase binding sites over 3 orders of magnitude to within a factor of 3 has been published as well.[S3] It is therefore reasonable to expect that a library covering m_u range of about 10^5 used in our model can be computationally designed and

constructed experimentally to within the tolerance required by our model (e.g. $\sigma < 0.5 - 1$). Additionally such a library (or even a random library) can be further flattened by appropriate preselection of cells using fluorescence-activated cell sorting (FACS).

References

(S1) Isaacs, F. J., Hasty, J., Cantor, C. R., and Collins, J. J. (2003) Prediction and measurement of an autoregulatory genetic module. *Proc. Natl. Acad. Sci. U.S.A 100*, 7714–7719.

(S2) Salis, H. M., Mirsky, E. A., and Voigt, C. A. (2009) Automated design of synthetic ribosome binding sites to control protein expression. *Nat. Biotechnol. 27*, 946–950.

(S3) Brewster, R. C., Jones, D. L., and Phillips, R. (2012) Tuning promoter strength through RNA polymerase binding site design in Escherichia coli. *PLoS Comput. Biol. 8*, e1002811.