Science Advances
AAAS

advances.sciencemag.org/cgi/content/full/6/5/eaay2378/DC1

Supplementary Materials for

One-step regression and classification with cross-point resistive memory arrays

Zhong Sun, Giacomo Pedretti, Alessandro Bricalli, Daniele Ielmini*

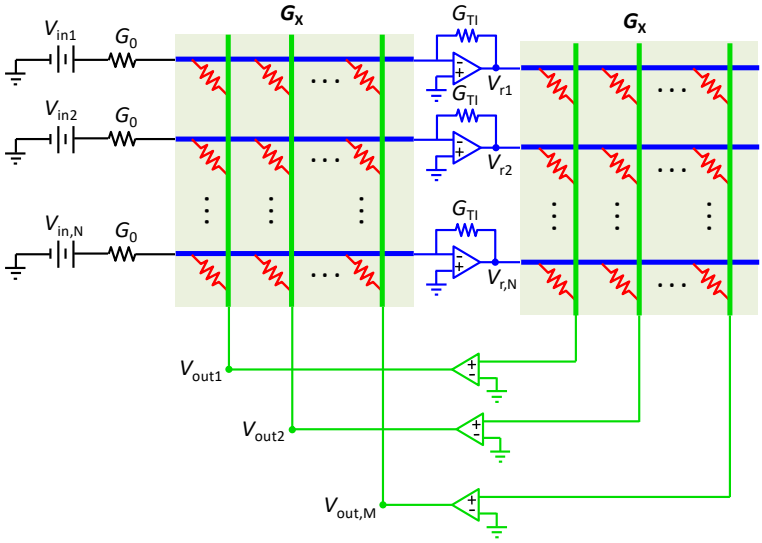
*Corresponding author. Email: daniele.ielmini@polimi.it

Published 31 January 2020, *Sci. Adv.* **6**, eaay2378 (2020)
DOI: 10.1126/sciadv.aay2378

This PDF file includes:

Supplementary Text

Text S1. Analysis of circuit stability



In the linear regression circuit as shown above, the input currents are generated by input voltages ($\boldsymbol{v_{in}} = [V_{in1}; V_{in2}; \dots; V_{in,N}]$) and resistors with conductance G_0, which is also the unit for the conductance matrix $\boldsymbol{G_X}$ and the conductance G_{TI} of transimpedance resistors. The output voltages of transimpedance amplifiers are given by $\boldsymbol{v_r} = [V_{r1}; V_{r2}; \dots; V_{r,N}]$.

According to the Kirchhoff's voltage law and the amplifier theory, the circuit can be described by the following equations

$$-\boldsymbol{D}_N[\boldsymbol{G_X v_{out}}(s) + G_0 \boldsymbol{v_{in}}(s) + G_{TI}\boldsymbol{v_r}(s)]L_1(s) = \boldsymbol{v_r}(s) \tag{S1-1}$$

$$\boldsymbol{D}_M \boldsymbol{G_X}^T \boldsymbol{v_r}(s) L_2(s) = \boldsymbol{v_{out}}(s) \tag{S1-2}$$

where $L_1(s)$ and $L_2(s)$ are the open loop gain of the negative feedback amplifiers and the positive feedback amplifiers, respectively, while s is the complex variable in the Laplace transform. $\boldsymbol{D}_N$ is an $N \times N$ diagonal matrix defined as $\boldsymbol{D}_N = diag(\frac{1}{G_{X_{11}}+G_{X_{12}}+\cdots G_{X_{1M}}+G_0+G_{TI}}, \frac{1}{G_{X_{21}}+G_{X_{22}}+\cdots G_{X_{2M}}+G_0+G_{TI}}, \dots, \frac{1}{G_{X_{N1}}+G_{X_{N2}}+\cdots G_{X_{NM}}+G_0+G_{TI}})$, and $\boldsymbol{D}_M$ is a $M \times M$ diagonal matrix defined as $\boldsymbol{D}_M = diag(\frac{1}{G_{X_{11}}+G_{X_{21}}+\cdots G_{X_{N1}}}, \frac{1}{G_{X_{12}}+G_{X_{22}}+\cdots G_{X_{N2}}}, \dots, \frac{1}{G_{X_{1M}}+G_{X_{2M}}+\cdots G_{X_{NM}}})$.

By using the variables of the linear regression problem $\boldsymbol{Xw} = \boldsymbol{y}$, Eq. (S1) can be rewritten as

$$-\boldsymbol{U}_N[\boldsymbol{Xw}(s) - \boldsymbol{y}(s) + t\boldsymbol{z}(s)]L_1(s) = \boldsymbol{z}(s) \tag{S2-1}$$

$$\boldsymbol{U}_M \boldsymbol{X}^T \boldsymbol{z}(s) L_2(s) = \boldsymbol{w}(s) \tag{S2-2}$$

where $\boldsymbol{z}$ represents $\boldsymbol{v_r}$, $\boldsymbol{U}_N$ is defined as $\boldsymbol{U}_N = diag(\frac{1}{X_{11}+X_{12}+\cdots X_{1M}+1+t}, \frac{1}{X_{21}+X_{22}+\cdots X_{2M}+1+t}, \ldots, \frac{1}{X_{N1}+X_{N2}+\cdots X_{NM}+1+t})$, $\boldsymbol{U}_M$ is defined as $\boldsymbol{U}_M = diag(\frac{1}{X_{11}+X_{21}+\cdots X_{N1}}, \frac{1}{X_{12}+X_{22}+\cdots X_{N2}}, \ldots, \frac{1}{X_{1M}+X_{2M}+\cdots X_{NM}})$.

The two equations in Eq. (S2) can be merged as follows

$$-\boldsymbol{X}\boldsymbol{U}_M\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{z}(s)L_1(s)L_2(s) + \boldsymbol{y}(s)L_1(s) - t\boldsymbol{z}(s)L_1(s) = {\boldsymbol{U}_N}^{-1}\boldsymbol{z}(s) \quad \text{(S3)}$$

We consider a single-pole model for all the operational amplifiers (OAs), namely $L_1(s) = \frac{L_{01}}{1+\frac{s}{\omega_{01}}}$ and $L_2(s) = \frac{L_{02}}{1+\frac{s}{\omega_{02}}}$, where L_{01} and L_{02} are the DC open-loop gains, ω_{01} and ω_{02} are the 3-dB bandwidths of both sets of OAs. As a result, Eq. (S3) becomes

$$-L_{01}\omega_{01}L_{02}\omega_{02}\boldsymbol{X}\boldsymbol{U}_M\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{z}(s) + L_{01}\omega_{01}(\omega_{02}+s)\boldsymbol{y}(s) - tL_{01}\omega_{01}(\omega_{02}+s)\boldsymbol{z}(s) = {\boldsymbol{U}_N}^{-1}(\omega_{01}\omega_{02} + \omega_{01}s + \omega_{02}s + s^2)\boldsymbol{z}(s) \quad \text{(S4)}$$

Since the DC open-loop gain is usually much larger than 1, Eq. (S4) can be approximated by

$$-L_{01}\omega_{01}L_{02}\omega_{02}\boldsymbol{X}\boldsymbol{U}_M\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{z}(s) + L_{01}\omega_{01}(\omega_{02}+s)\boldsymbol{y}(s) - tL_{01}\omega_{01}s\boldsymbol{z}(s) = {\boldsymbol{U}_N}^{-1}s^2\boldsymbol{z}(s) \quad \text{(S5)}$$

namely

$$\boldsymbol{z}(s) = L_{01}\omega_{01}\left(L_{01}\omega_{01}L_{02}\omega_{02}\boldsymbol{X}\boldsymbol{U}_M\boldsymbol{X}^{\boldsymbol{T}} + tL_{01}\omega_{01}\boldsymbol{I}s + {\boldsymbol{U}_N}^{-1}s^2\right)^{-1}(\omega_{02}+s)\boldsymbol{y}(s) \quad \text{(S6)}$$

where $\boldsymbol{I}$ is the $N \times N$ identity matrix. By considering the single-pole model for $L_2(s)$, Eq. (S2-2) becomes

$$-L_{02}\omega_{02}\boldsymbol{U}_M\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{z}(s) = (\omega_{02}+s)\boldsymbol{w}(s) \quad \text{(S7)}$$

By combining Eqs. (S6) and (S7), we get

$$\boldsymbol{w}(s) = -L_{01}\omega_{01}L_{02}\omega_{02}\boldsymbol{U}_M\boldsymbol{X}^{\boldsymbol{T}}\left(L_{01}\omega_{01}L_{02}\omega_{02}\boldsymbol{X}\boldsymbol{U}_M\boldsymbol{X}^{\boldsymbol{T}} + tL_{01}\omega_{01}\boldsymbol{I}s + {\boldsymbol{U}_N}^{-1}s^2\right)^{-1}\boldsymbol{y}(s) \quad \text{(S8)}$$

By defining $p_1 = L_{01}\omega_{01}$ and $p_2 = L_{02}\omega_{02}$, which are the gain bandwidth products of both sets of OAs, and $s = \lambda p_1$ to convert s into a dimensionless variable λ, Eq. (S8) becomes

$$\boldsymbol{w}(s) = -\frac{p_2}{p_1}\boldsymbol{U}_M\boldsymbol{X}^{\boldsymbol{T}}\left(\frac{p_2}{p_1}\boldsymbol{X}\boldsymbol{U}_M\boldsymbol{X}^{\boldsymbol{T}} + \lambda t\boldsymbol{I} + \lambda^2{\boldsymbol{U}_N}^{-1}\right)^{-1}\boldsymbol{y}(s) \quad \text{(S9)}$$

from which the poles of the system are determined by

$$det\left(\frac{p_2}{p_1}\boldsymbol{X}\boldsymbol{U}_M\boldsymbol{X}^{\boldsymbol{T}} + \lambda t\boldsymbol{I} + \lambda^2{\boldsymbol{U}_N}^{-1}\right) = 0 \quad \text{(S10)}$$

which is a typical quadratic eigenvalue problem (QEP). By introducing the mass matrix $\boldsymbol{M} = \boldsymbol{U_N}^{-1}$, the damping matrix $\boldsymbol{C} = t\boldsymbol{I}$, and the stiffness matrix $\boldsymbol{K} = \frac{p_2}{p_1}\boldsymbol{X}\boldsymbol{U_M}\boldsymbol{X^T}$, and noting that $\boldsymbol{M}$ and $\boldsymbol{C}$ are diagonal and hence symmetric positive-definite, while $\boldsymbol{K}$ is symmetric positive-semidefinite, we can conclude that all the real parts of λ are negative or zero[45]. Specifically, the $N + M$ negative eigenvalues correspond to the $N + M$ poles in the system of $N + M$ OAs, that is all the poles lie in the left half-plane and thus the system is stable.

Text S2. Analysis of twin matrices mismatch

The left matrix is the nominal matrix $\boldsymbol{X}$. Term the right matrix $\boldsymbol{Z}$, which is programmed according to $\boldsymbol{X}$ but with inevitable errors, namely $\boldsymbol{Z} = \boldsymbol{X} + \Delta\boldsymbol{X}$.

The analytical solution $\boldsymbol{w}^*$ to the overdetermined linear system $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}$ satisfies $\boldsymbol{X^T}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}^*) = 0$, while the experimental solution $\boldsymbol{w_{real}}$ in the circuit satisfies $\boldsymbol{Z^T}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w_{real}}) = 0$. Assume $\boldsymbol{w_{real}} = \boldsymbol{w}^* + \Delta\boldsymbol{w}$, and $\Delta\boldsymbol{y} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}^*$, there is

$\Delta\boldsymbol{w} = (\boldsymbol{Z^T}\boldsymbol{X})^{-1}\Delta\boldsymbol{X^T}\Delta\boldsymbol{y}$.

For the ideal matrix $\boldsymbol{X}$, there is $\boldsymbol{X^T}\Delta\boldsymbol{y} = 0$. Therefore, if the *i*th column of $\Delta\boldsymbol{X}$ is a linear combination of columns of $\boldsymbol{X}$, namely $\Delta\boldsymbol{X}(:, i) = \alpha_1\boldsymbol{X}(:,1) + \alpha_2\boldsymbol{X}(:,2) + \cdots \alpha_m\boldsymbol{X}(:, m)$ where α_j is arbitrary real numbers, the *i*th element of $\Delta\boldsymbol{w}$ is zero.

For the simple linear regression, due to both matrices have the first column filled with ones which are represented by fixed discrete resistors, the first column of $\Delta\boldsymbol{X}$ is filled with zeros, and hence $\Delta\boldsymbol{w_0} = 0$. And if $\Delta\boldsymbol{x} = \alpha\boldsymbol{1} + \beta\boldsymbol{x}$ where α and β are arbitrary real numbers, there is $\Delta w_1 = 0$.

Note that the weight error $\Delta\boldsymbol{w}$ is also dependent on the inverse matrix of $\boldsymbol{Z^T}\boldsymbol{X}$, which translates that $\Delta\boldsymbol{w}$ is determined by the condition number κ of matrix $\boldsymbol{Z^T}\boldsymbol{X}$. A larger κ tends to induce a larger error $\Delta\boldsymbol{w}$.

Text S3. Polynomial regression

For polynomial regression, there is a dataset $\{(x_i, y_i), i = 1,2, \ldots, N\}$ to be fitted with a polynomial model $w_0 + w_1x_i + w_2{x_i}^2 + \cdots + w_k{x_i}^k = y_i$, where x_i is the independent variable, y_i is the dependent variable, w_0, w_1, w_2, ... and w_n are modelling weights to be solved. If we only consider the first terms, polynomial regression can be written in the matrix form as Eq. (1), where

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1 & {x_1}^2 \\ 1 & x_2 & {x_2}^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & {x_N}^2 \end{bmatrix} \tag{S11}$$

$\boldsymbol{w}$ is the weight vector comprising w_0, w_1 and w_2, $\boldsymbol{y}$ is a vector composed of observations of the dependent variable. Mapping matrix $\boldsymbol{X}$ in the crosspoint circuit, and implementing $-\boldsymbol{y}$ with input currents, the polynomial regression weights can be computed in one step. One example is shown in fig. S8.

Text S4. Introduction to Boston housing dataset

Boston housing dataset contains information of 506 houses in suburbs of Boston. The information refers to 14 attributes including the price for each house. The 14 attributes are indicated as follows.

crim: per capita crime rate by town.
zn: proportion of residential land zoned for lots over 25,000 sq.ft.
indus: proportion of non-retail business acres per town.
chas: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
nox: nitrogen oxides concentration (parts per 10 million).
rm: average number of rooms per dwelling.
age: proportion of owner-occupied units built prior to 1940.
dis: weighted mean of distances to five Boston employment centres.
rad: index of accessibility to radial highways.
tax: full-value property-tax rate per 10,000$.
ptratio: pupil-teacher ratio by town.
black: $1000 \times (Bk - 0.63)^2$ where Bk is the proportion of blacks by town.
lstat: lower status of the population (percent).
medv: median value of owner-occupied homes in 1000$.

Text S5. Analysis of least squares

The error squares of solving an overdetermined linear system $\boldsymbol{Xw} = \boldsymbol{y}$ is $F(\boldsymbol{w}) = (\boldsymbol{Xw} - \boldsymbol{y})^{\boldsymbol{T}}(\boldsymbol{Xw} - \boldsymbol{y})$, whose minimum locates at $\boldsymbol{w}^* = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T y}$. Therefore,

$$\begin{aligned} F(\boldsymbol{w}^*) &= (\boldsymbol{Xw}^* - \boldsymbol{y})^{\boldsymbol{T}}(\boldsymbol{Xw}^* - \boldsymbol{y}) \\ &= \boldsymbol{w}^{*\boldsymbol{T}}\boldsymbol{X^T X w}^* - \boldsymbol{w}^{*\boldsymbol{T}}\boldsymbol{X^T y} - \boldsymbol{y^T X w}^* + \boldsymbol{y^T y} \\ &= \boldsymbol{y}^{\boldsymbol{T}}(\boldsymbol{y} - \boldsymbol{Xw}^*) \end{aligned} \tag{S12}$$

Due to the label matrix is composed of 1 or -1, $\boldsymbol{y^T y}$ is the same for different columns (*i.e.*, different digits) in the label matrix, the value is always *N*, which is the number of training samples. To this end, there is

$$F(\boldsymbol{w}^*) \propto -\boldsymbol{y^T X w}^* \tag{S13}$$

which indicates that there is a linear relationship between the least squares error and $\boldsymbol{w}^*$, as shown in Fig. 4B.

Text S6. Computing performance benchmarking

Here we benchmark the throughput and energy efficiency of the pseudoinverse circuit for training the second-layer weights $\boldsymbol{W}^{(2)}$ for MNIST dataset in Fig. 4 of the manuscript.

For the same datasets, namely 3,000 training digits and 500 test digits, and the same 1st-layer weights $\boldsymbol{W}^{(1)}$, the matrix to be stored for pseudoinverse computation is defined by

$$\boldsymbol{X} = f\left(\boldsymbol{T}\boldsymbol{W}^{(1)}\right) \tag{S14}$$

where $\boldsymbol{T}$ is the training matrix, *f* is the sigmoid function. Every element in $\boldsymbol{X}$ is randomized with an error within $\pm 5\%$, then the 2nd-layer weights $\boldsymbol{W}^{(2)}$ are computed by

$$\boldsymbol{W}^{(2)} = \boldsymbol{X}^{+}\boldsymbol{Y} \tag{S15}$$

where $\boldsymbol{Y}$ is the label matrix.

Based on the pre-set $\boldsymbol{W}^{(1)}$ and the computed $\boldsymbol{W}^{(2)}$, we can evaluate the recognition rate, which is shown in fig. S21a. Five randomized $\boldsymbol{W}^{(2)}$ were tried, and the recognition rate varies in the range from 93% to 94%, with an average value of 93.4%, slightly lower than the one (94.2%) with ideal solution of $\boldsymbol{W}^{(2)}$. These results demonstrate the robustness of the neural network against device variation of RRAM, thus strongly supporting the feasibility of one-step machine learning with the circuit for classification applications.

Figure S21b shows the minimal eigenvalue λ_{min} of the QEP in Eq. (S10), which was calculated for each trial. For all the 5 cases, λ_{min} lies in the range from 1.16×10^{-4} to 1.2×10^{-4}, which is one order of magnitude lower than the λ_{min} for the Boston housing dataset. According to the linear dependence of computing time on $1/\lambda_{min}$ (fig. S20b), the time for training the weights of one output neuron for MNIST is estimated to be 145 μs. As a result, the total time to train the whole $\boldsymbol{W}^{(2)}$ of 10 handwritten digits is expected to be 1.45 ms.

In conventional computers, the complexity of computing the linear regression weights $\boldsymbol{w} = \boldsymbol{X}^{+}\boldsymbol{y} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$ is composed of:
$O(M^2N)$ to multiply $\boldsymbol{X}^{\boldsymbol{T}}$ by $\boldsymbol{X}$, considering $\boldsymbol{X}$ is of size $M \times N$;
$O(MN)$ to multiply $\boldsymbol{X}^{\boldsymbol{T}}$ by $\boldsymbol{y}$;
$O(M^3)$ to compute the LU (or Cholesky) factorization of $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$ and use that to compute the product $(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$.
Therefore, the total number of floating-point operations is $M^2N + MN + M^3$, while the complexity is dominated by $O(M^2N)$, as N is generally much larger than M.
In the case of training one column of $\boldsymbol{W}^{(2)}$, the total number of operations is $M^2N + MN + M^3 = 785^2 \times 3000 + 785 \times 3000 + 785^3 = 2.335 \times 10^9$. As a result, the equivalent throughput of the pseudoinverse circuit is 16.1 tera-operations per second (TOPS).

During the computation of the entire $\boldsymbol{W}^{(2)}$, the input was scaled down to ±50 mV, to protect the crosspoint devices from unwanted disturb. As a result, the largest output is below ±1 V, and most outputs are around few tens of mV. To evaluate the energy efficiency of the circuit, we calculated the power consumption for computing $\boldsymbol{W}^{(2)}$. The energy consumption by the circuit is composed of 3 parts, namely:
1, the energy consumption by the left crosspoint resistive array, that is $P_1 = \sum_{j=1}^{M} V_{DD}\left|V_j\right| \sum_{i=1}^{N} G_{X_{ij}}$, where $M = 785$, $N = 3000$ in this case, V_{DD} is the supply voltage of OAs that is assumed as 1 V. The other variables are referred to Supplementary text 1.
2, the energy consumption by the right crosspoint resistive array, that is $P_2 = \sum_{i=1}^{N} V_{DD}\left|V_{r,i}\right| \left(G_0 + \sum_{i=1}^{M} G_{X_{ij}}\right)$.
3, the energy consumption by the input resistors, that is $P_3 = \sum_{i=1}^{N} {V_{in,i}}^2 G_0$.

The overall power consumption of computing the weights connected to the 10 output neurons is calculated to be 3.556 W, or 355.6 mW per operation for computing the weights connected to 1 output neuron, with the conductance unit of 10 μS assumed in the manuscript. As a result, the energy efficiency of the circuit is calculated to be 45.3 TOPS/W. As an approximate comparison, the energy efficiency of Google's TPU is 2.3 TOPS/W (Ref. 49), thus the crosspoint circuit is 19.7 times more energy-efficient than TPU. We also compare the crosspoint circuit with a low-precision (4-bit) ASIC system, whose optimized energy efficiency is 7.02 TOPS/W (Ref. 50), indicating a 6.5 times better performance for the crosspoint circuit.

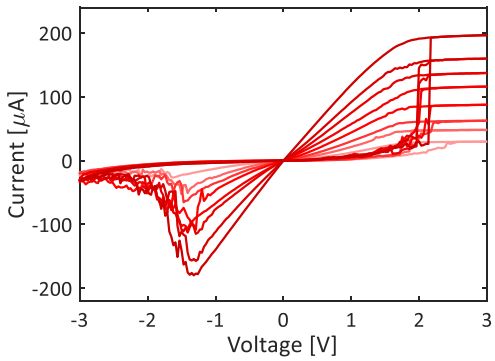


Fig. S1. Current-voltage characteristics of the Ti/HfO_2/C RRAM device. Representative 8 conductance levels were programmed by controlling the compliance current during the set transition.

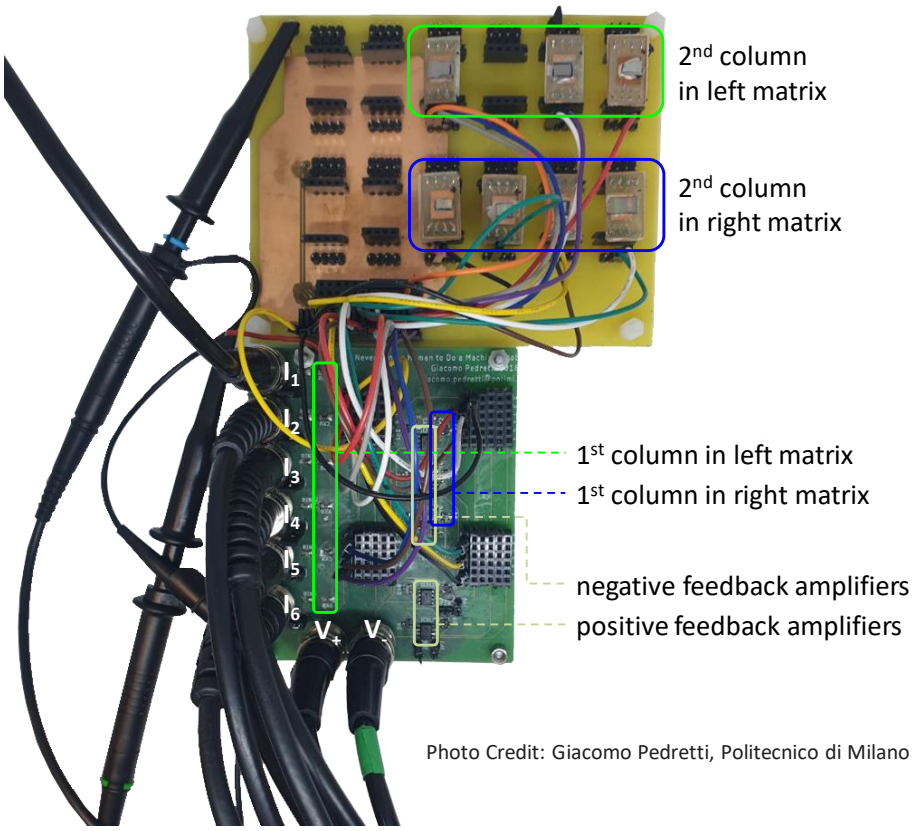


Fig. S2. Cross-point resistive memory circuit on a printed circuit board. Positions of various circuit components, including fixed discrete resistors, RRAM devices and amplifiers, input currents and supply voltages for operational amplifiers (OAs) are indicated. In the second column of left/right matrix, multiple RRAM devices on a same chip may be used. Each OA module contains 2 OAs.

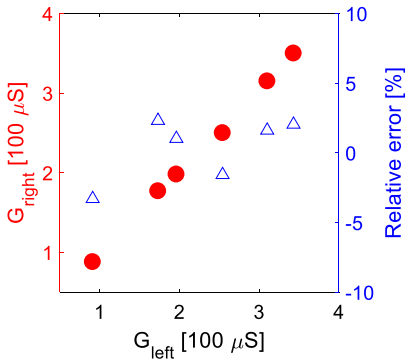


Fig. S3. Device programming. The resistive memory devices in the right crosspoint array were programmed according to the conductance values in the left array. The relative error of programming is shown as the right *y*-axis.

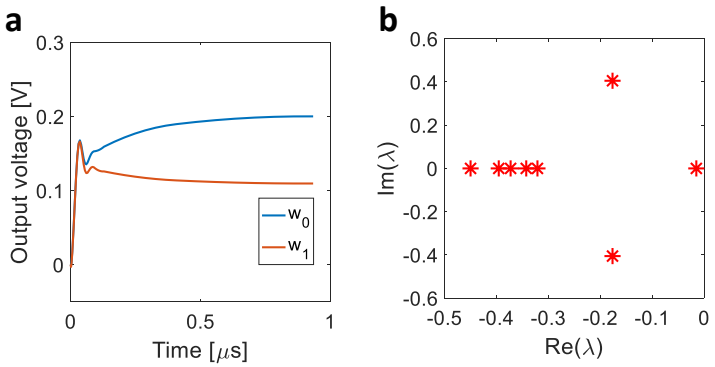


Fig. S4. Convergence analysis of the linear regression experiment. (**A**) SPICE transient simulation of training the linear regression model in Fig. 1B. (**B**) Eigenvalues in the complex plane of the QEP for the linear regression case. The minimal eigenvalue (or real part of eigenvalue) λ_{min} (absolute value) is 0.0151. Note that all eigenvalues are located in the half plane with Re(λ) < 0, which satisfies the requirement for stability of the circuit.

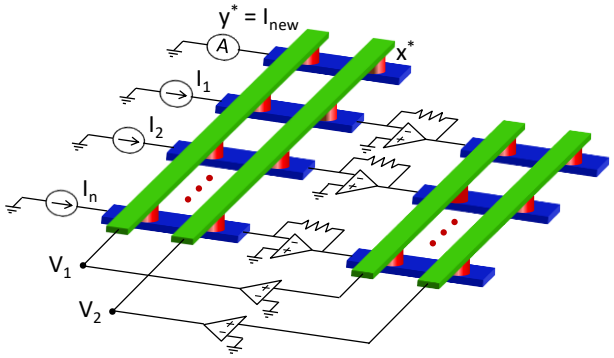


Fig. S5. Extended circuit for one-step prediction. The circuit is based on the pseudoinverse circuit for linear regression, with the new datum x^* stored in a RRAM device in an extra row, while the prediction y^* represented by the read current.

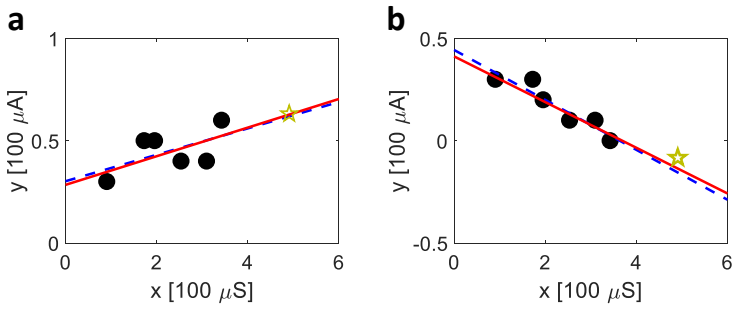


Fig. S6. More linear regression results. (**A**) Linear regression with $\boldsymbol{i}$ = [0.3; 0.5; 0.5; 0.4; 0.4; 0.6]I_0 for the same matrix in Fig. 1. A new datum x^* = 4.91 was used to test the prediction, whose experimental result is 0.632, with a relative error of 2% compared to the analytical prediction. (**B**) Linear regression with $\boldsymbol{i}$ = [0.3; 0.3; 0.2; 0.1; 0.1; 0]I_0. The experimental prediction to the new datum is -0.082, with a relative error of -47% compared to the analytical prediction, due to the small amplitude of the prediction.

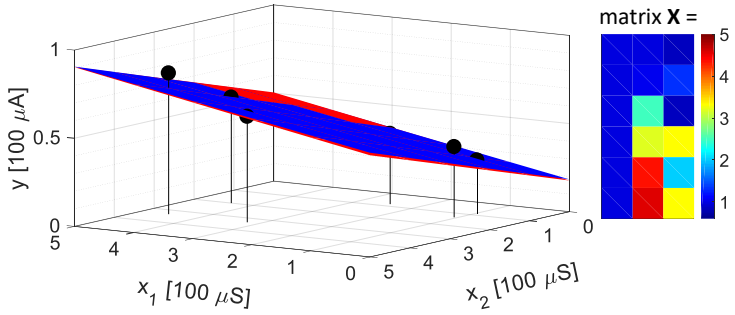


Fig. S7. Linear regression with two independent variables. Values of the 2 independent variables (x_1 and x_2) for 6 data points are recorded, respectively, in the second and the third column of matrix $\boldsymbol{X}$, which is mapped in the twin crosspoint arrays. The first column of $\boldsymbol{X}$ is fixed with discrete resistors. The values of the dependent variable y are represented by $\boldsymbol{i}$ = [0.3; 0.4; 0.4; 0.6; 0.6; 0.8]I_0. Both experimental (red) and analytical (blue) regression planes are shown.

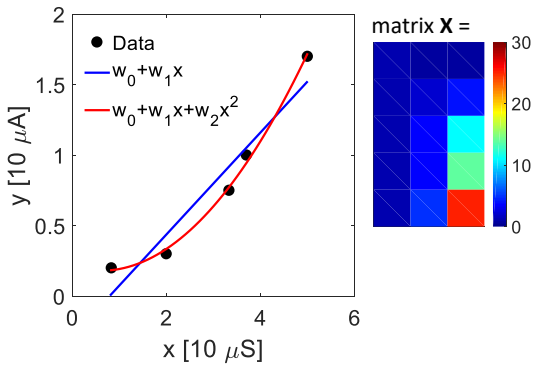


Fig. S8. Polynomial regression result. A dataset of 5 points is used for polynomial regression demonstration. In the matrix $\boldsymbol{X}$, the first column is a column of ones, the second column records the x values of 5 points, and the third column is responsible for the quadratic term in x^2 the polynomial regression model. Matrix $\boldsymbol{X}$ was mapped in the twin crosspoint arrays in circuit simulation, and the known vector $\boldsymbol{y}$ was mapped by an input current vector $\boldsymbol{i}$ = [0.2; 0.3; 0.75; 1; 1.7]I_0, where I_0 is 10 μA, as the conductance transformation unit is 10 μS due to the large values of the quadratic term. In the plot, the linear regression result is also shown as a reference, indicating a better regression result with polynomial regression.

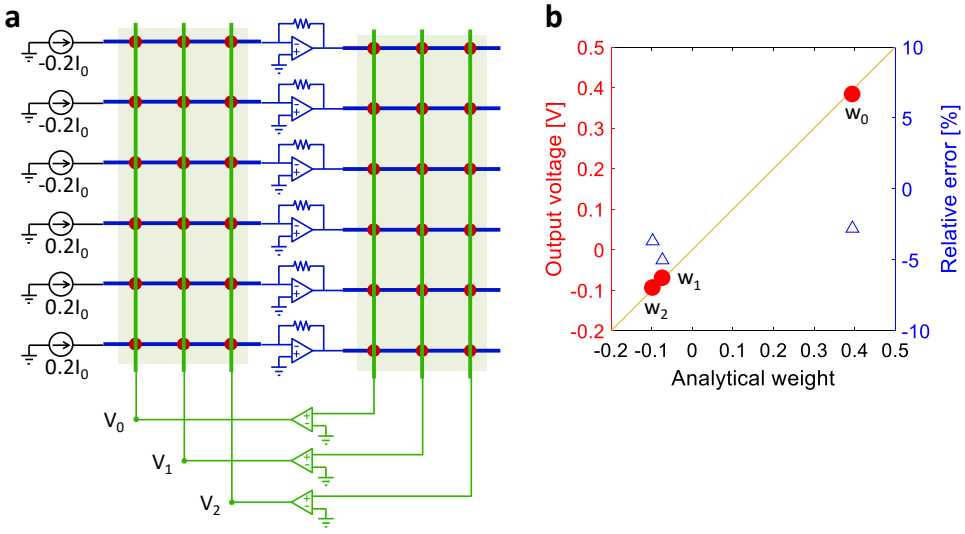


Fig. S9. Logistic regression results. (**A**) Circuit schematic of logistic regression in Fig. 2. The first column in both arrays were fixed with discrete resistors, the second and the third columns were implemented with RRAM devices and discrete resistors, respectively. (**B**) Experimental weights of logistic regression, plotted in comparison with the analytical solution. The right *y*-axis shows the relative errors of the experimental results.

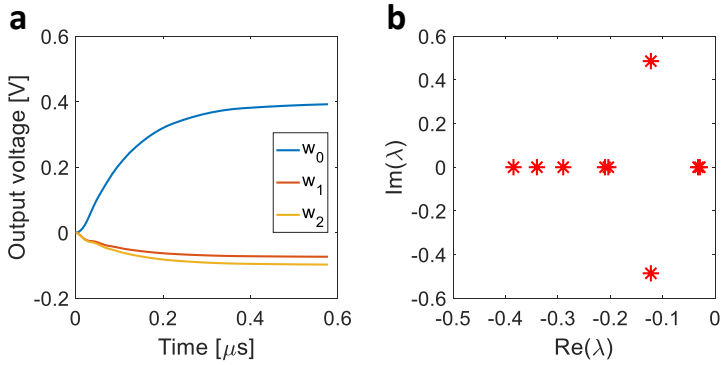


Fig. S10. Convergence analysis of the logistic regression experiment. (**A**) SPICE transient simulation of training the logistic regression model in Fig. 2. (**B**) Eigenvalues in the complex plane of the QEP for the logistic regression case. The minimal eigenvalue λ_{min} is equal to 0.0294.

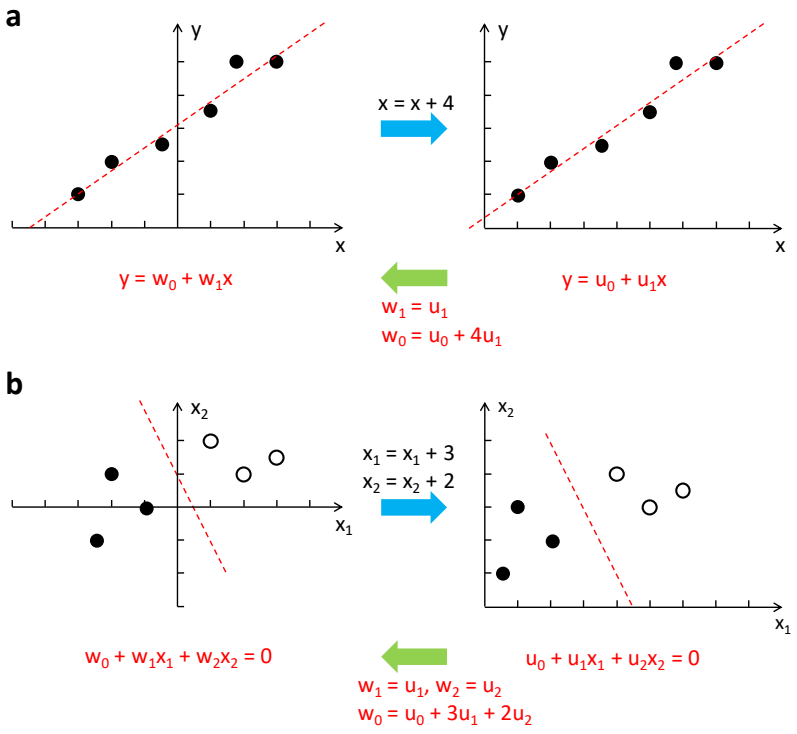


Fig. S11. Solution of linear/logistic regression with negative independent variable values. (**A**) Illustration of the linear regression with negative independent-variable values. (**B**) Same as (A), but for the logistic regression. All data are moved to the positive half-plane by a rigid shift, and then stored in the crosspoint arrays for computation. The regression weights can be straightforwardly recovered from those computed by the circuit.

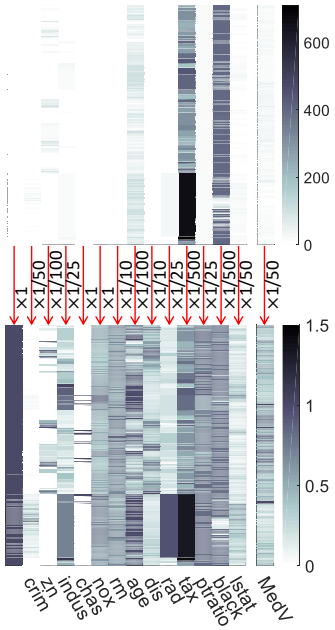


Fig. S12. Rescaling the attribute matrix *X* and price vector *y*. Each column in matrix ***X*** was rescaled with a specific factor to make the overall matrix uniform. Matrix ***X*** was mapped in the twin crosspoint arrays with a conductance unit of 10 μS. The vector ***y*** was scaled down by a factor of 1/50, and mapped by input currents with a unit of 10 μA. In circuit simulation, the output voltages were rescaled correspondingly with the same factors to recover the real weights.

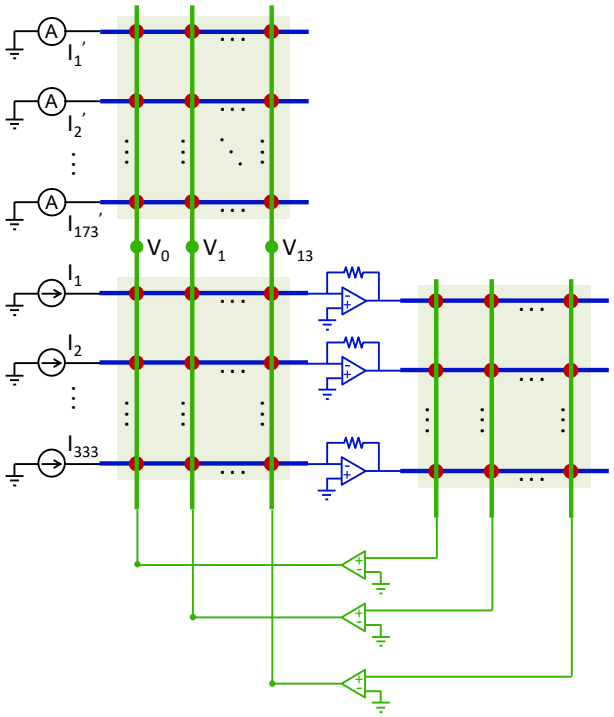


Fig. S13. One-step prediction circuit schematic for Boston housing dataset. The attribute matrix $\boldsymbol{X}$ of the training set (333 samples) is used to compute the regression weights, which in the form of voltages are utilized to execute MVM in a crosspoint array with the attribute matrix of the test set (173 samples). The collected currents at rows of the top crosspoint array $\boldsymbol{i}'$ dictate the predicted prices of the houses in test set.

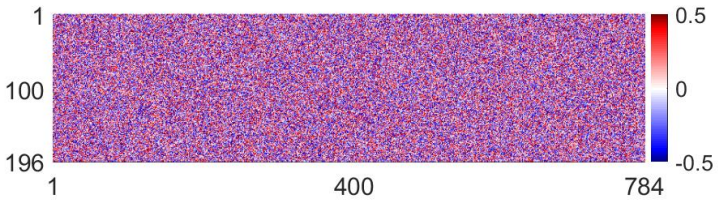


Fig. S14. Random first-layer weight matrix $\boldsymbol{W}^{(1)}$. The matrix is with a size of 196×784, and each element is generated randomly in the range of [-0.5, 0.5] following a uniform distribution.

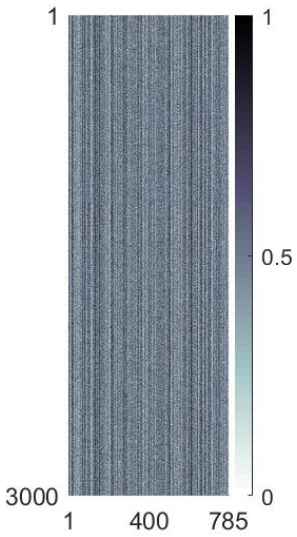


Fig. S15. The hidden-layer output matrix *X*. The matrix is of a size of 3,000×785, as there are 3,000 training samples, and 784 hidden neurons and 1 bias. The matrix is obtained with sigmoid function for each hidden neuron in each input event.

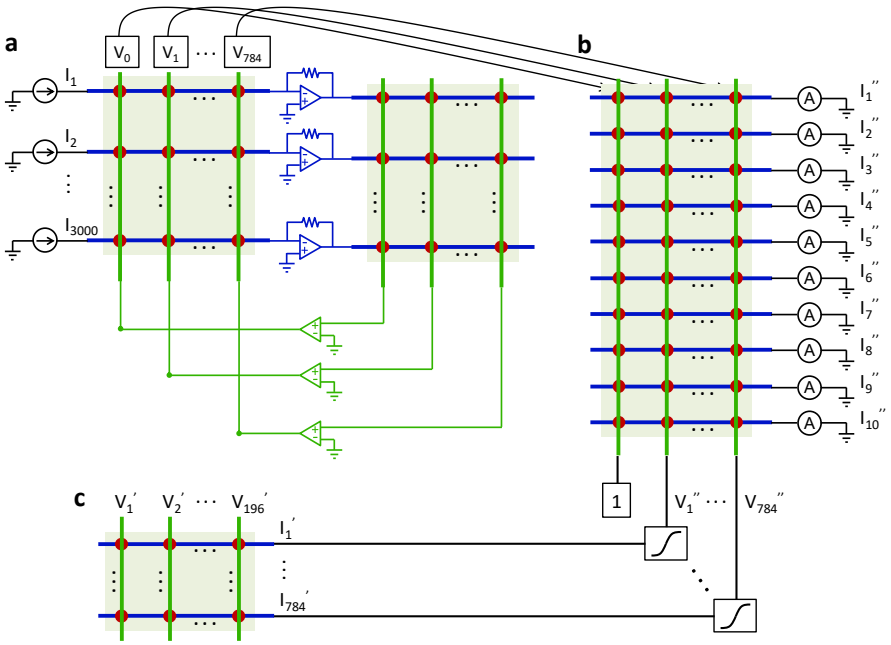


Fig. S16. Training and inference of the two-layer neural network. (**A**) Training/computing the second-layer weights for one output neuron. The weights are computed in circuit simulation as output voltages, *i.e.*, V_0, V_1, ..., V_{784}. The computed weights are encoded in a crosspoint resistive memory array with a transformation from voltage to conductance, as indicated by the arrows from (A) to (B). For the 10 categories of digits, 10 operations of circuit computing and transformation programming are needed. (**B**) The 785×10 crosspoint array coding the second-

layer weights $\boldsymbol{W}^{(2)}$. It will be used directly for the following inference phase, together with the first layer in (C). (**C**) The first layer of the 2-layer neural network for inference. Simple MVM is executed with the 196×784 crosspoint array, *i.e.* $\boldsymbol{W}^{(1)}$, and the input voltages (V_0', V_1', ..., V_{196}') representing one input digit image. The collected currents of MVM results will be activated with the sigmoid function in software or hardware to generate the hidden neuron outputs, which is also in the form of voltages. The hidden layer voltages (1, V_1'', ..., V_{784}'') are applied to the rows of the crosspoint array in (B) to complete the inference of neural network. The first row is applied with 1 V, which is a constant bias for every output neuron. During an inference event, the largest current collected among the 10 columns (I_1'', I_2'', ..., I_{10}'') decides which corresponding digit is recognized as. Note that both weight matrices $\boldsymbol{W}^{(1)}$ and $\boldsymbol{W}^{(2)}$ contain negative elements, while here they are represented by two single crosspoint resistive memory arrays for simplicity. A real-number matrix can be split into two positive matrices, which are then implemented with two crosspoint arrays.

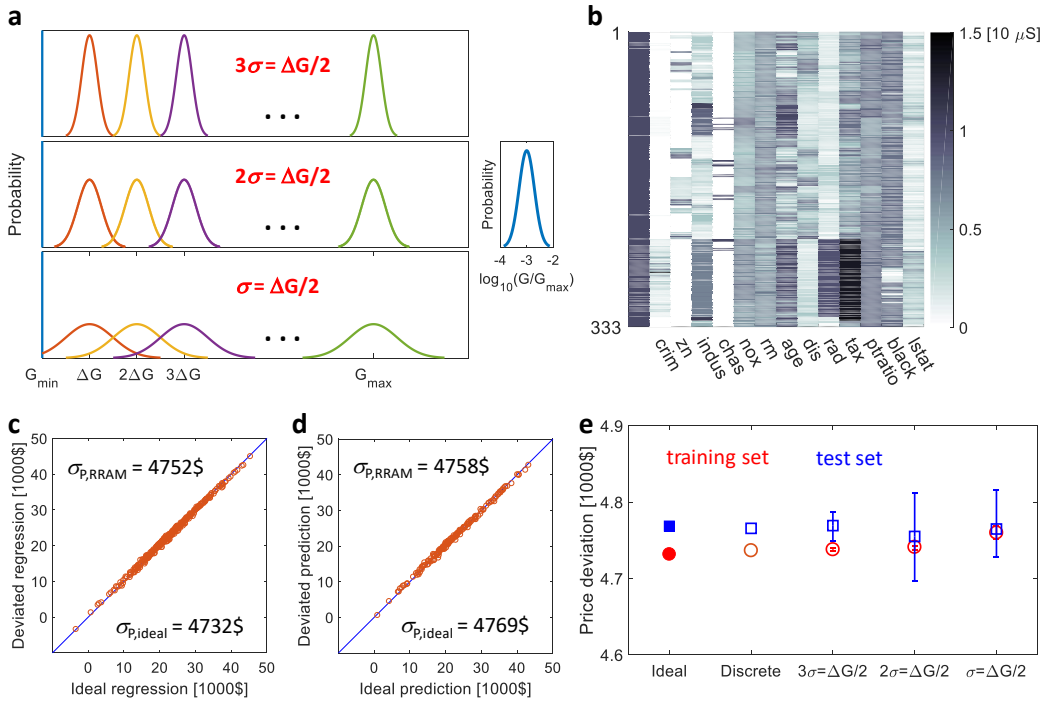


Fig. S17. Linear regression of Boston housing dataset with a RRAM model. (**A**) Discretization and randomization of RRAM conductance levels. 32 conductance levels are assumed, including 1 HRS which defines G_{min}, and 31 uniformly-spaced levels till the maximum conductance (G_{max}). For the uniformly-space levels, a conductance deviation is assumed to be 1/6, 1/4 or 1/2 of the nominal conductance difference between two adjacent levels (ΔG). The HRS conductance follows a logarithmic normal distribution, with a standard deviation of 0.3, as shown in the right panel. (**B**) The representative attribute matrix $\boldsymbol{X}$ randomly generated with $\sigma = \Delta G/2$. (**C**) Regressed house prices of the training set with the weights computed with the real RRAM matrix $\boldsymbol{X}$ in (B), plotted in comparison with the prices regressed with weights computed with the ideal attribute matrix. With the two sets of weights, the price deviations are very close,

i.e., 4732$ vs 4752$. (**D**) Predicted house prices of the test set using the same weights, plotted in comparison with the prices predicted with weights computed with the ideal attribute matrix. The price deviations show an even more insignificant difference, *i.e.*, 4769$ vs 4758$. (**e**) Summary of price deviations for attribute matrix $\boldsymbol{X}$ implemented with different conditions, including ideal $\boldsymbol{X}$, discretization-only $\boldsymbol{X}$, discretized and randomized $\boldsymbol{X}$ with conditions in (**a**), respectively. For randomization-involved cases, 10 trials were recorded for each condition, and average values with error bars are shown. The results indicate that even with a device variation $\sigma = \Delta G/2$, the price deviation remains in a reasonable range of error for both regression (training set) and prediction (test set), though for the test set, the price deviation is more scattering.

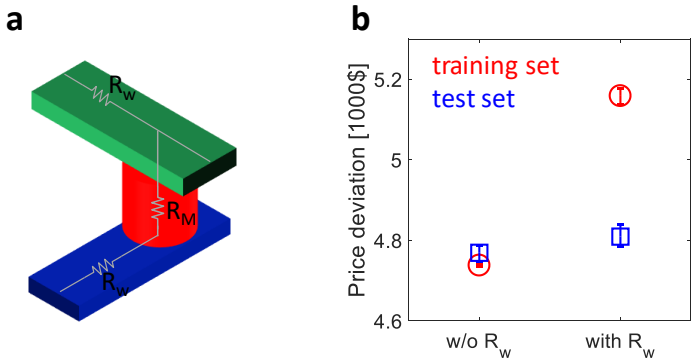


Fig. S18. Impact of wire resistance. (**A**) Sub-circuit module of a crosspoint resistive memory device. The wire resistance R_w is obtained with interconnect parameters at 65 nm technology, R_M is the resistance of a crosspoint resistive memory. The module is duplicated for the twin of crosspoint arrays. (**B**) Price deviations for regression (training set) and prediction (test set) calculated with simulated weights, which is obtained in the crosspoint circuit with wire resistances. The RRAM model is assumed with $\sigma = \Delta G/6$. Due to wire resistances, σ_P increases for both training set and test set, with the latter being more insignificant, indicating a sufficient prediction accuracy.

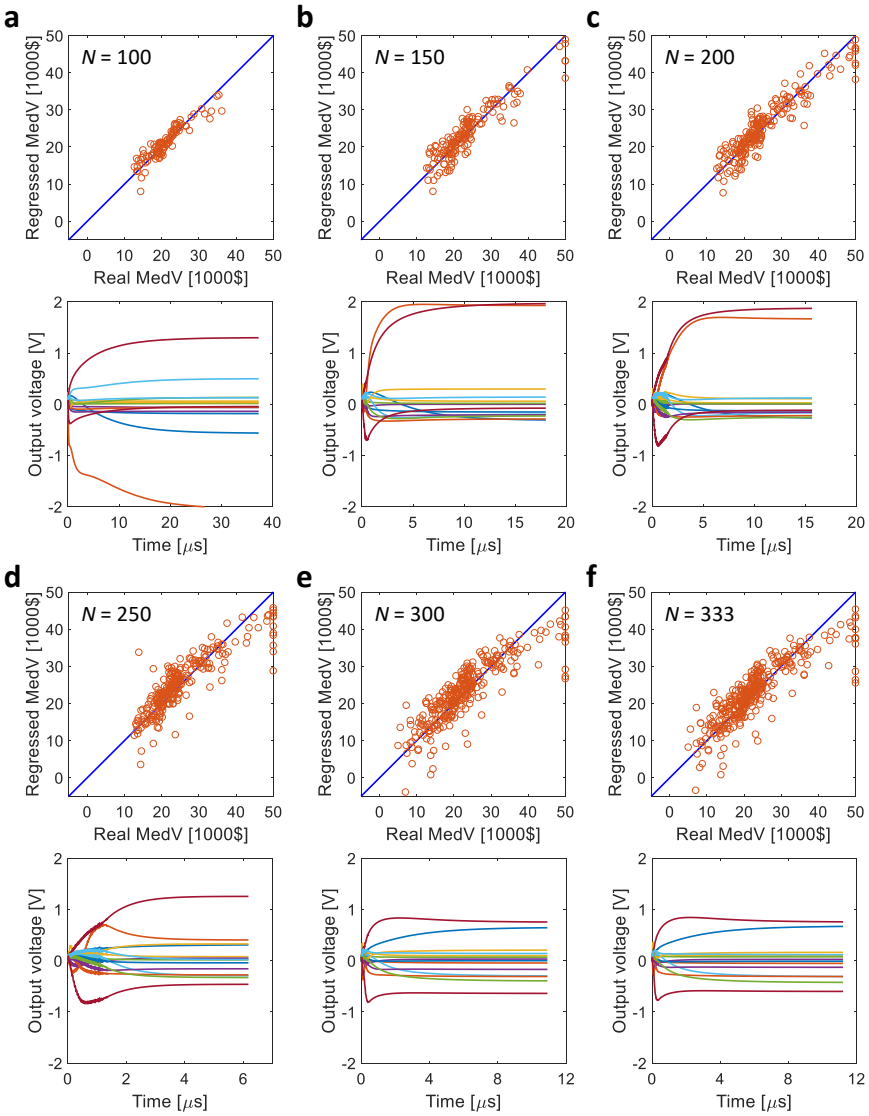


Fig. S19. Linear regression of Boston housing dataset and its representative subsets. (**A**) Linear regression of a subset of 100 training samples, together with the transient simulation result. (**B**) Same as (A), but for a subset of 150 training samples. (**C**) Same as (A), but for a subset of 200 training samples. (**D**) Same as (A), but for a subset of 250 training samples. (**E**) Same as (A), but for a subset of 300 training samples. (**F**) Same as (A), but for the entire training set of 333 samples.

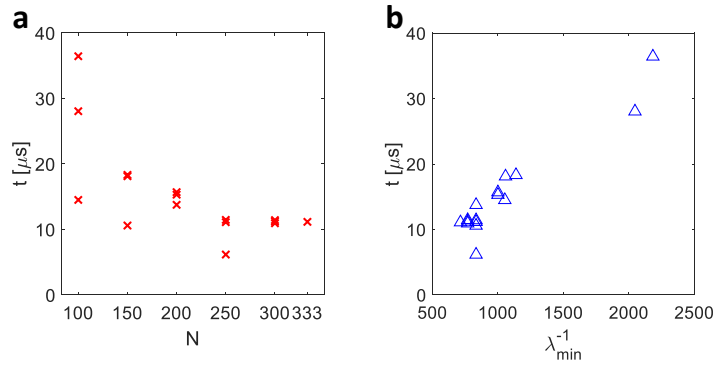


Fig. S20. Scaling behavior of computing time of linear regression. (**A**) Computing time of linear regression of Boston housing dataset and its subsets. For each size, 3 different subsets are used. (**B**) Dependence of computing time on λ_{min}.

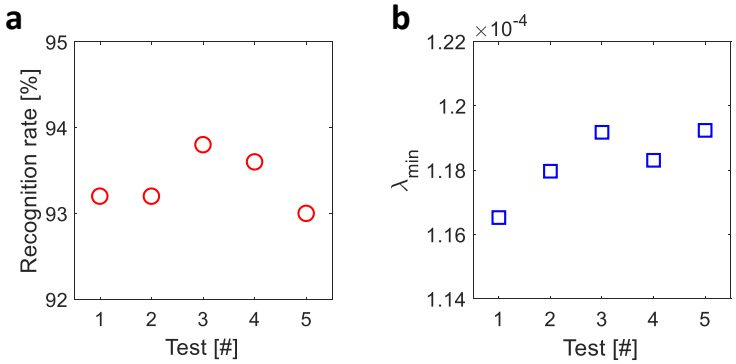


Fig. S21. Analysis of device variation impact and computing time. (**A**) Recognition rate with randomized $\boldsymbol{X}$ for 5 trials. Every element in $\boldsymbol{X}$ is randomized with an error within $\pm$5%. (**B**) λ_{min} of the 5 trials. For all the 5 cases, λ_{min} lies in the range from 1.16×10^{-4} to 1.2×10^{-4}. According to the linear dependence of computing time on $1/\lambda_{min}$ (fig. S20b), the time for training the weights of one output neuron for MNIST is estimated to be 145 μs.