UNIVERSITY OF CAMBRIDGE

Back to the Future for machine learning

Gareth Conduit

Theory of Condensed Matter group

Machine learning

Train from sparse data to merge simulations, physical laws, and experimental data

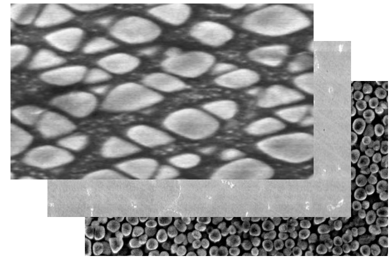
Reduce the need for expensive experimental development

Accelerate materials and drugs discovery

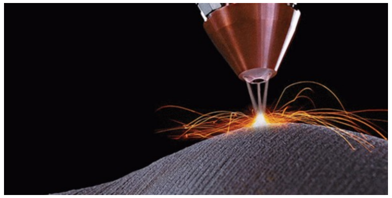
Generic with proven applications in materials discovery and drug design

Materials designed

Nickel and molybdenum



3D printing with welding and printability

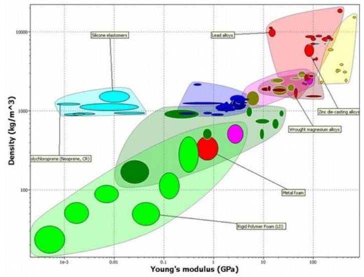


Experiment and DFT for batteries

Steel for welding

More materials

Identified and corrected errors in materials database



GRANTA
MATERIAL INSPIRATION

Lubricants with molecular dynamics and experiments

bp

Thermometry with experiment and computation

Cambridge Cryogenics

Drug design

e-therapeutics

optibrium

Takeda

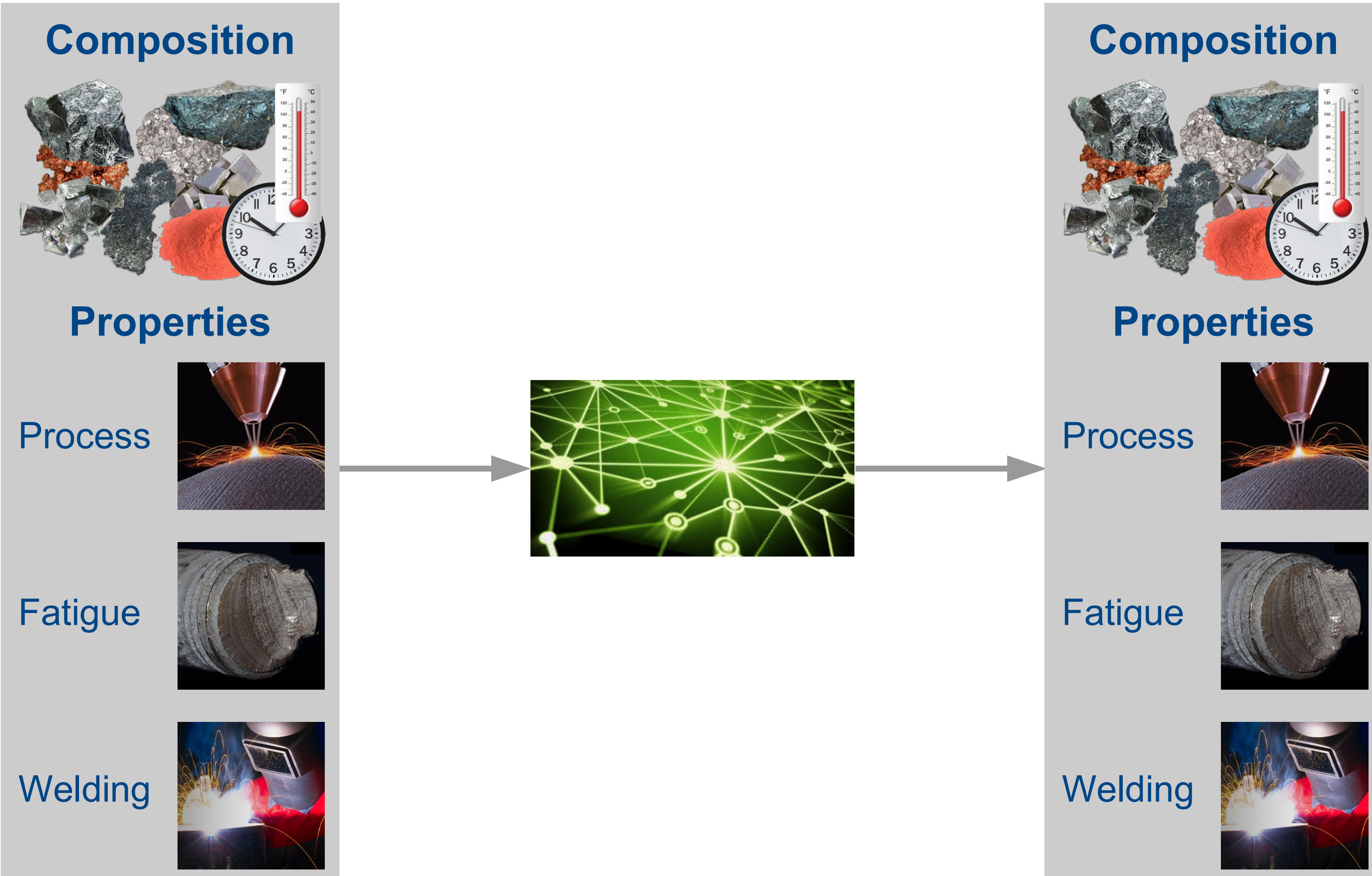
Plan of talk

Deep dive into the maths underpinning the neural network

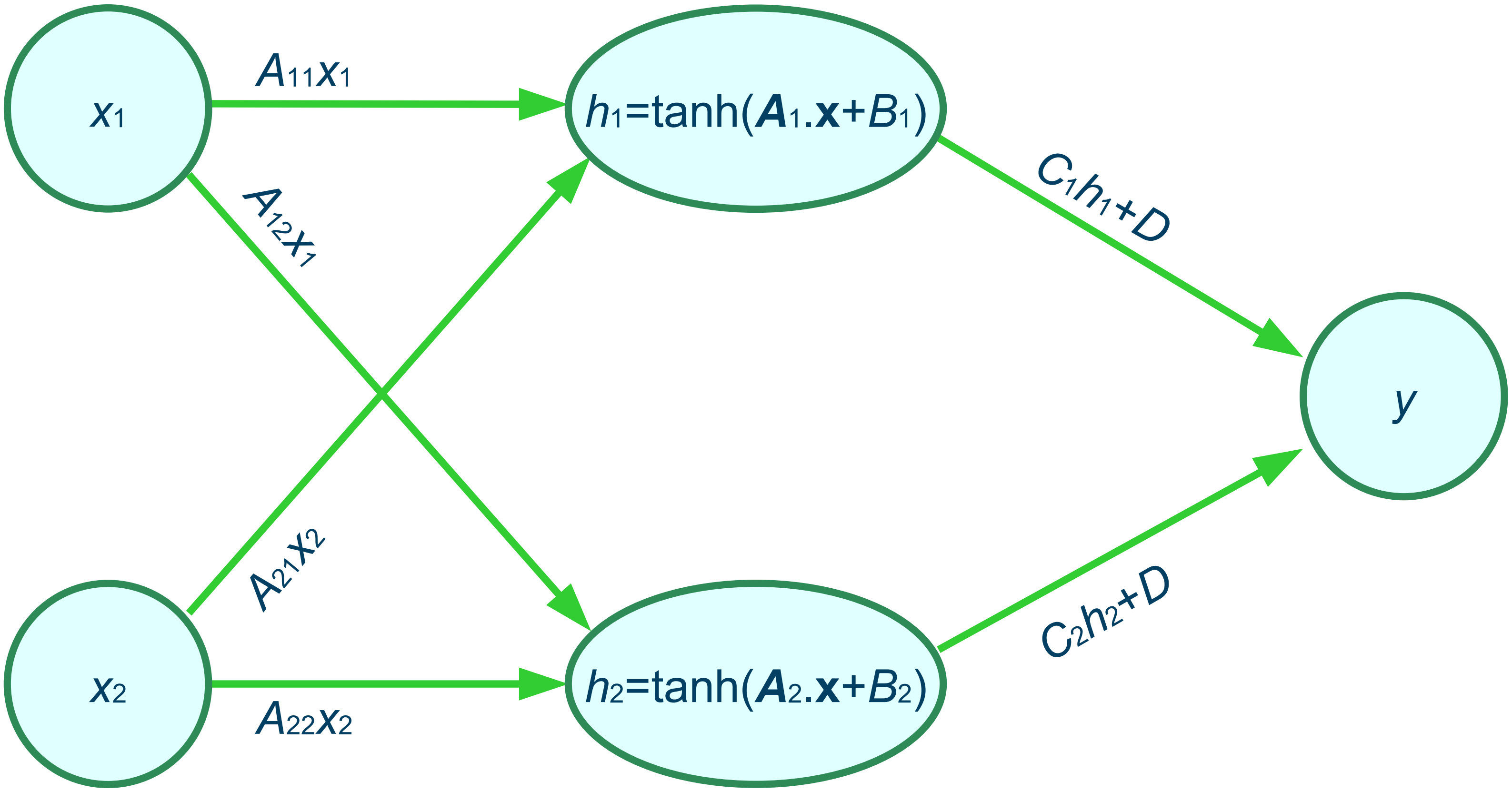
Accelerate and improve the activation function

Celeritate crystal plasticity calculations

Neural network for materials design



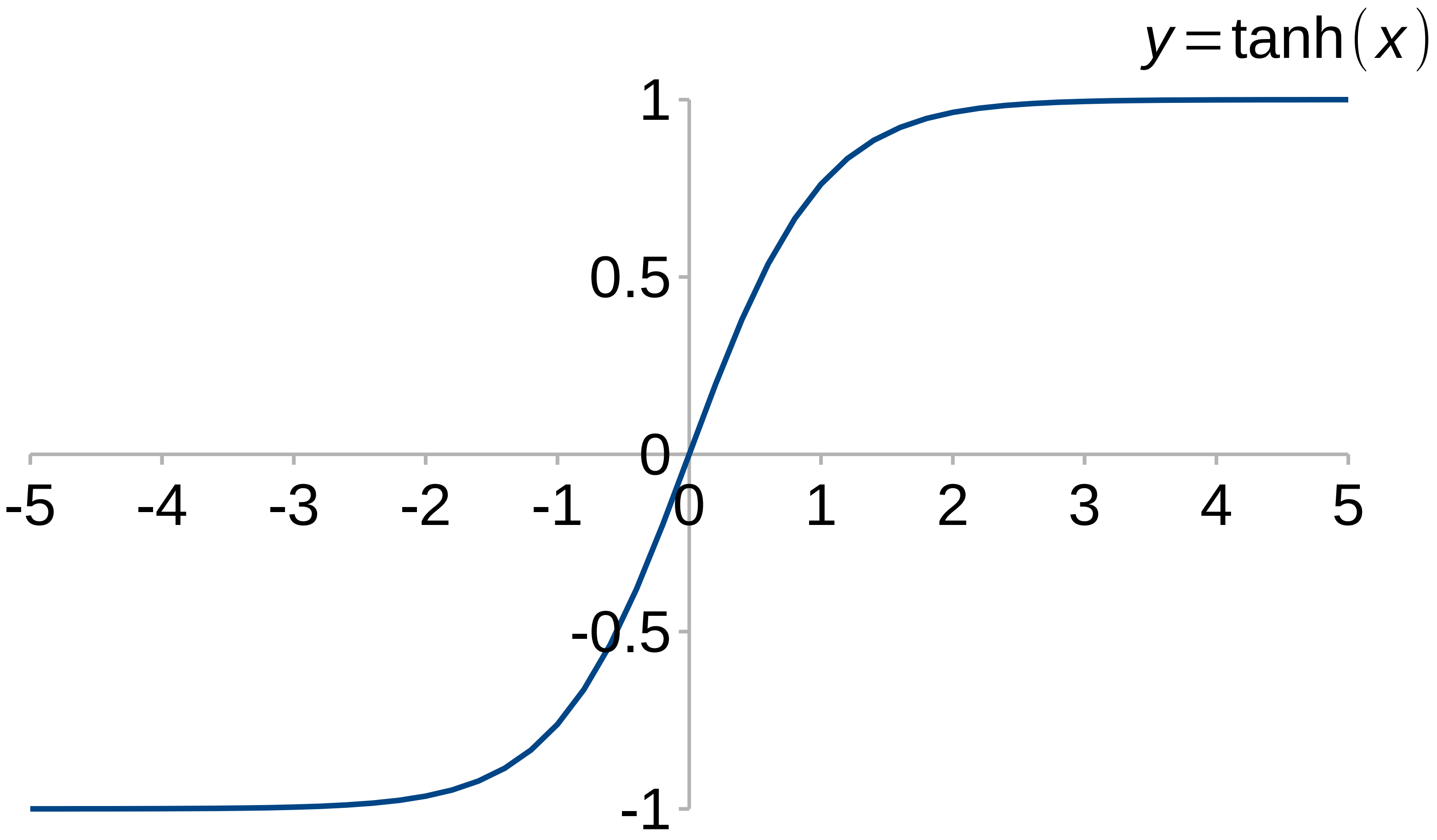
Underlying neural network



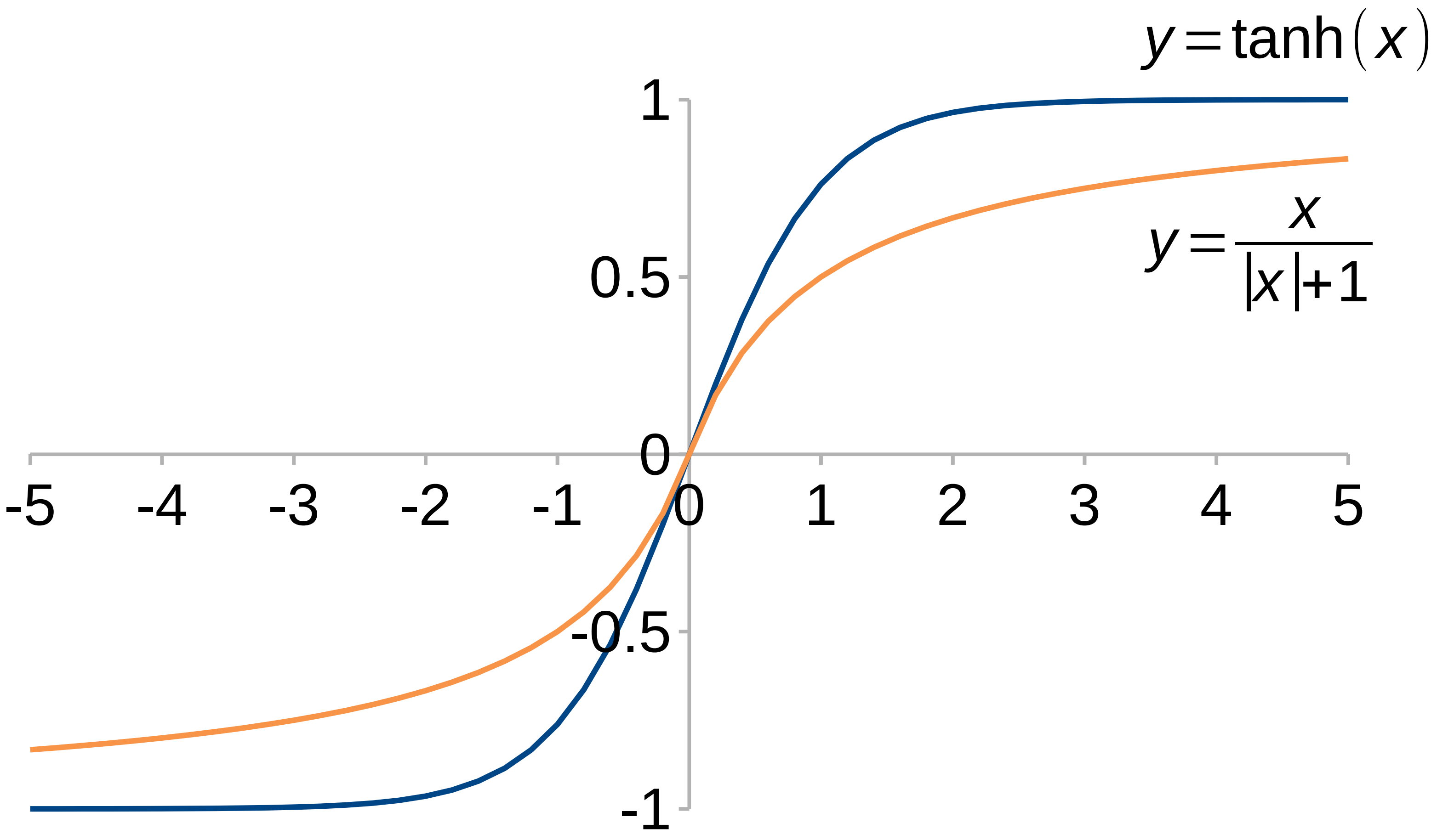
Accelerate the activation function

$$y = D + C\tanh\left(Ax + B\right)$$

Activation function



Proposed activation function



Accelerate the activation function

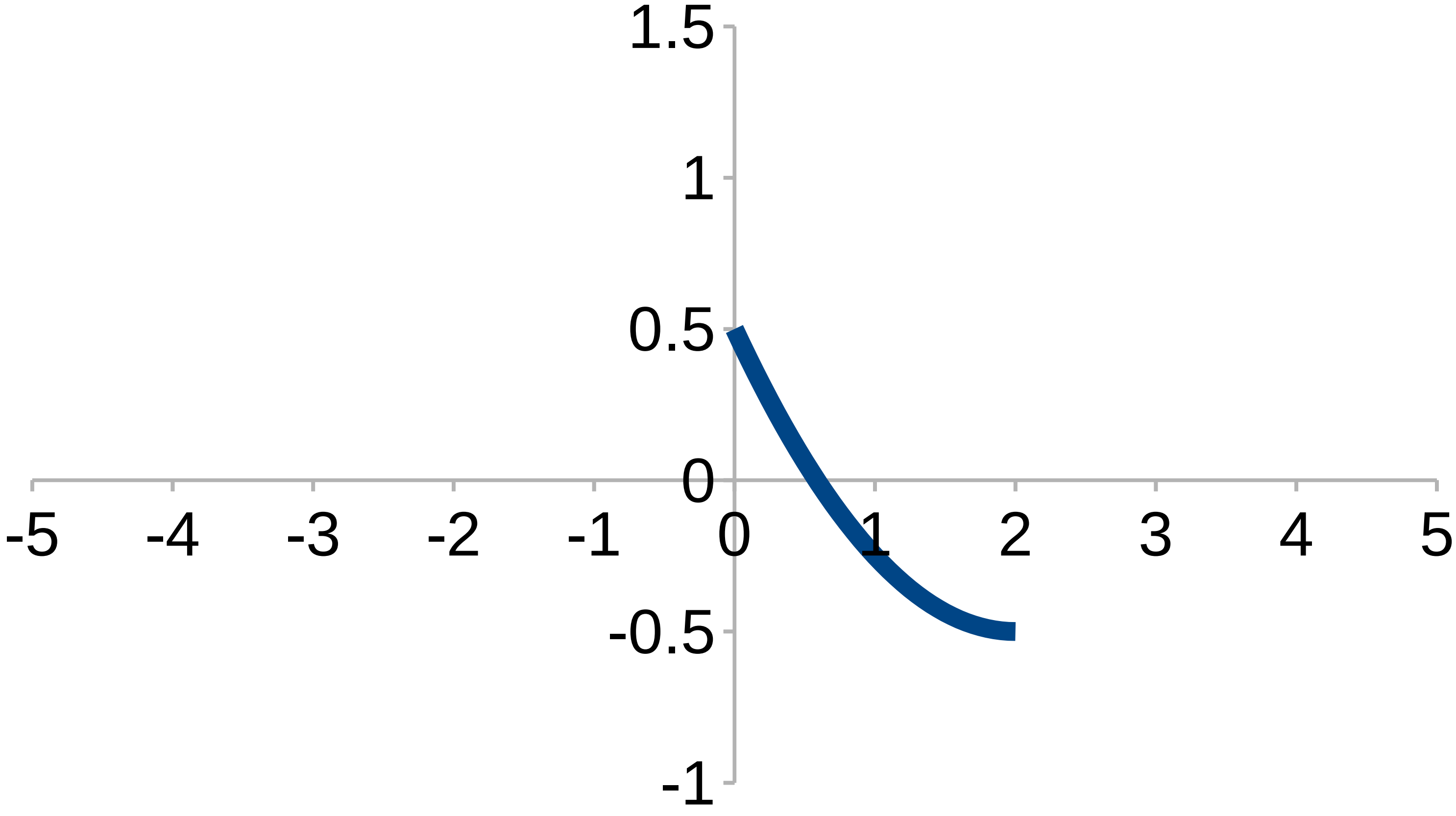
$$y = D + C\tanh\left(Ax + B\right)$$

$$y = D + C\frac{Ax + B}{\left|Ax + B\right| + 1}$$

Activation function takes 50% of time to evaluate

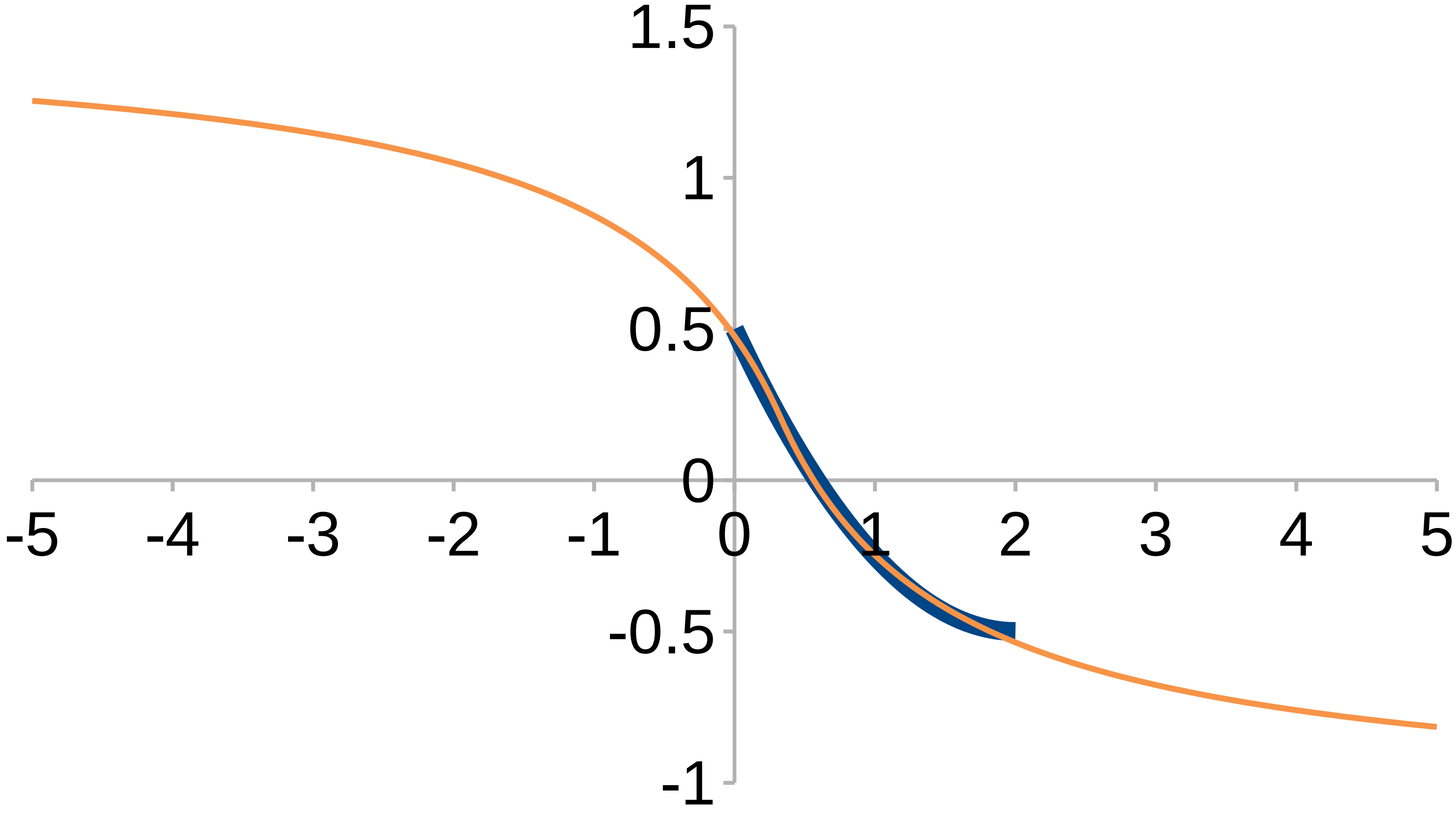
Training data for our neural network

$$y = 1 - x + x^2$$



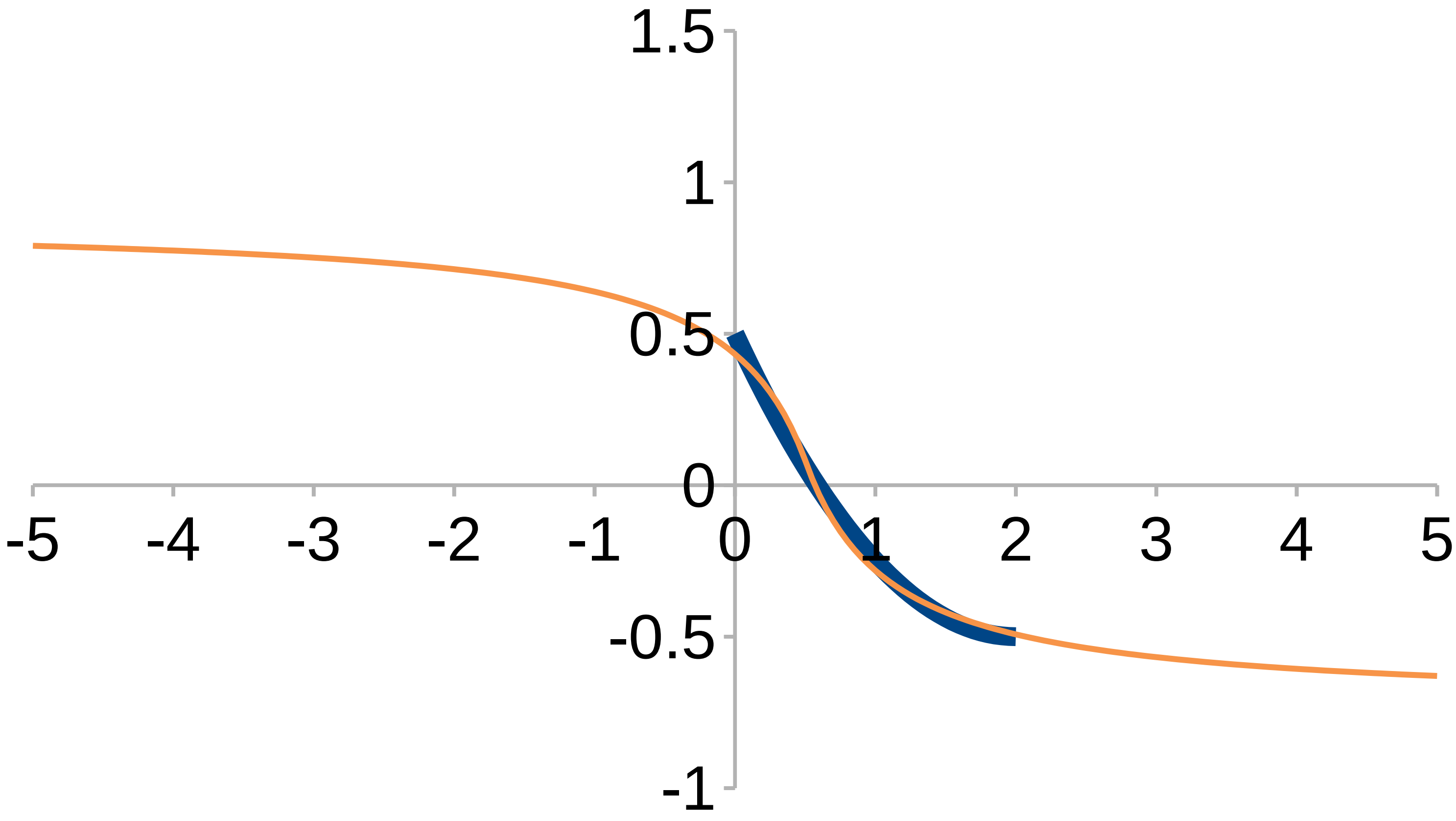
How a neural network fits the function

$$y = 0.21 + 1.29 \frac{-0.81x + 0.27}{|-0.81x + 0.27| + 1}$$



Another fit to the function

$$y = 0.07 + 0.80 \frac{-1.62x + 0.82}{|-1.62x + 0.82| + 1}$$



Neural network of multiple variables

$$y = D + C \frac{\vec{A} . \vec{x} + B}{\left|\vec{A} . \vec{x} + B\right| + 1}$$

Taylor expand the neural network

$$y = D + C\,\frac{\vec{A}\,.\,\vec{x} + B}{\left|\vec{A}\,.\,\vec{x} + B\right| + 1}$$

$$= \begin{cases} D + C\left(\vec{A}\,.\,\vec{x} + B\right) & \left|\vec{A}\,.\,\vec{x} + B\right| \ll 1 \\ D + C\,\mathrm{sign}\left(\vec{A}\,.\,\vec{x} + B\right) & \left|\vec{A}\,.\,\vec{x} + B\right| \gg 1 \end{cases}$$

A better form for a Taylor expansion

$$y = D + C\frac{\vec{A}.\vec{x}+B}{\left|\vec{A}.\vec{x}+B\right|+1}$$
$$= \begin{cases} D + C\left(\vec{A}.\vec{x}+B\right) & \left|\vec{A}.\vec{x}+B\right| \ll 1 \\ D + C\,\mathrm{sign}\left(\vec{A}.\vec{x}+B\right) & \left|\vec{A}.\vec{x}+B\right| \gg 1 \end{cases}$$

Change activation function f(***A.x***+*B*) to depend on multiple variables f(***A.x***+*B*,*C*)

$$y = D + C\frac{\vec{A}.\vec{x}+B}{\left|\vec{A}.\vec{x}+B\right|+C}$$
$$= \begin{cases} D + \vec{A}.\vec{x}+B & \left|\vec{A}.\vec{x}+B\right| \ll C \\ D + C\,\mathrm{sign}\left(\vec{A}.\vec{x}+B\right) & \left|\vec{A}.\vec{x}+B\right| \gg C \end{cases}$$

Physical formulae with multiplication

$$\mu = \frac{\tau_{\mathrm{i}}}{3\sigma_{\mathrm{y}}}$$

$$\kappa = \frac{6 E_{\mathrm{d}}' h \sigma_{\mathrm{d}}}{E_{\mathrm{s}}' H^2}$$

$$i_L = \frac{ZFDC}{\delta}$$

$$E = \frac{1}{2} k x^2$$

$$\sigma = \frac{3FL}{2bd^2}$$

$$V = IR$$

$$PV = nk_{\mathrm{B}} T$$

$$\rho = \frac{AR}{L}$$

$$\lambda = \left(\frac{m}{ne^2 \mu_0} \right)^{1/2}$$

Physical formulae with addition and multiplication

$$\mu = \frac{\tau_\mathrm{i}}{3\sigma_\mathrm{y}}$$

$$\kappa = \frac{6E_\mathrm{d}'h\sigma_\mathrm{d}}{E_\mathrm{s}'H^2}$$

$$i_L = \frac{ZFDC}{\delta}$$

$$E = \frac{1}{2}kx^2$$

$$\sigma = \frac{3FL}{2bd^2}$$

$$\omega = \left(\frac{k(m_1+m_2)}{m_1 m_2}\right)^{1/2}$$

$$i_\mathrm{A} = i_0 \exp\left(\frac{azF\eta}{RT}\right)$$

$$V = IR$$

$$V = I(R_1+R_2)$$

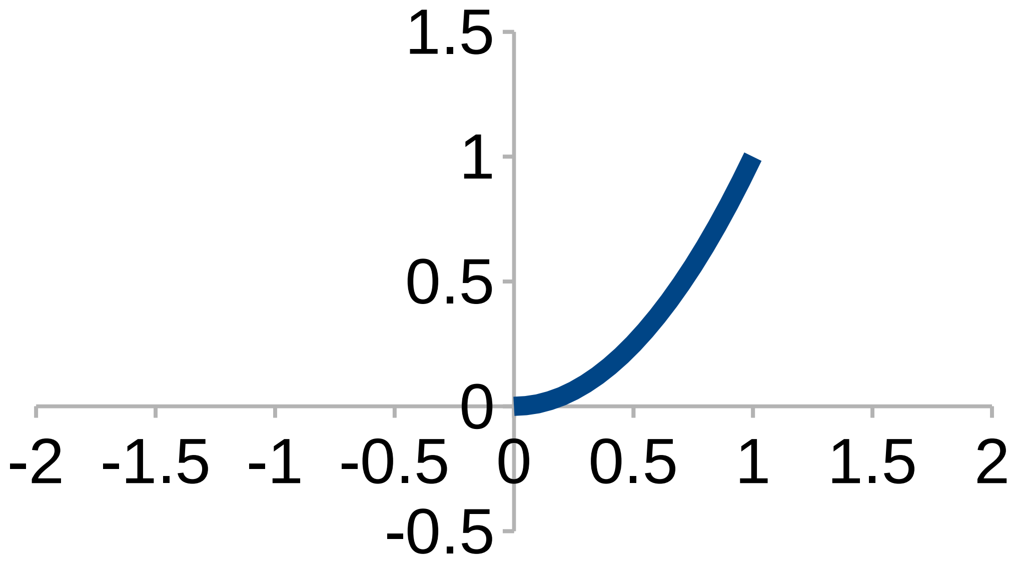
$$PV = nk_\mathrm{B}T$$

$$\rho = \frac{AR}{L}$$

$$\lambda = \left(\frac{m}{ne^2\mu_0}\right)^{1/2}$$

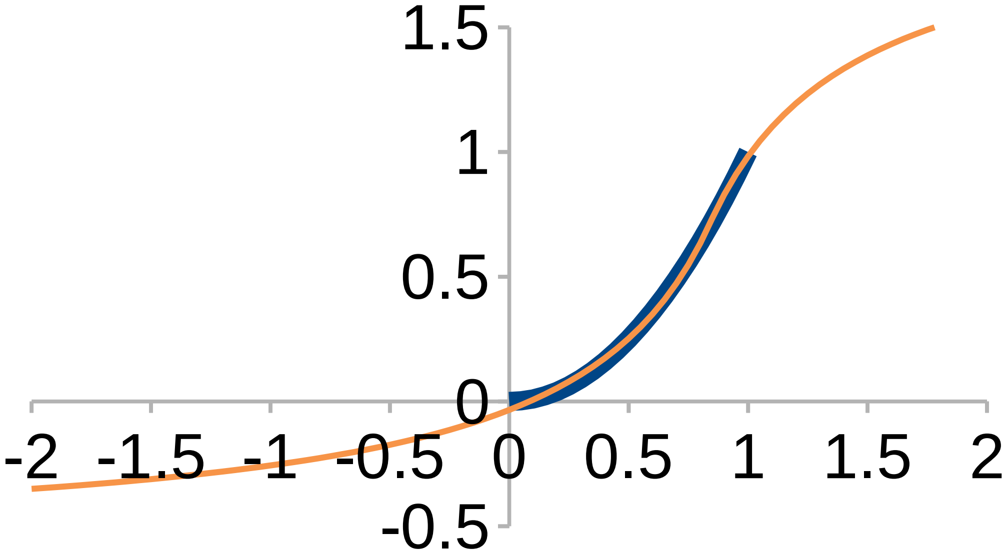
$$\epsilon = A\sigma^n \exp(-Q/RT)$$

Training data to enable multiplication



Neural network to replicate the parabola

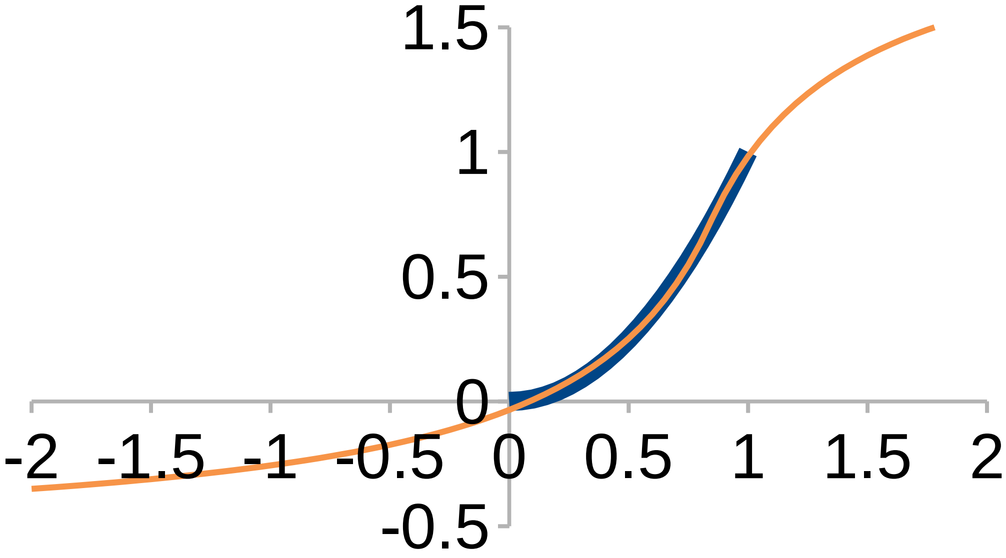
1) Shift the activation function into squared region



$$y = 0.57 + \frac{1.2\left(2.01x - 1.85\right)}{\left|2.01x - 1.85\right| + 1.2}$$

Neural network to replicate multiplication

1) Shift the activation function into squared region



$$y = 0.57 + \frac{1.2(2.01x - 1.85)}{|2.01x - 1.85| + 1.2}$$

2) Combine two activation functions in the square region

$$y = \underbrace{\left(\frac{x_1}{2} + \frac{x_2}{2}\right)^2}_{\text{Node 1}} - \underbrace{\left(\frac{x_1}{2} - \frac{x_2}{2}\right)^2}_{\text{Node 2}} = x_1 x_2$$

Can we do better with logarithms?

$$\log y = \underbrace{\left(\log x_1 + \log x_2\right)}_{\text{Node 1}} = \log\left(x_1 x_2\right)$$

$$y = x_1 x_2$$

Becomes tricky when x<0, and cannot recover addition

Blend addition and multiplication into one kernel

$$y = D + \bar{\alpha}\,\bar{y} + \sum_i \frac{\alpha_i C_i \left(\vec{A}_i . \left(\vec{x} - \bar{\vec{x}}\right) + B_i\right)}{\left|\vec{A}_i . \left(\vec{x} - \bar{\vec{x}}\right) + B_i\right| + C_i}$$

Blend addition and multiplication into one kernel

$$y = D + \bar{\alpha}\,\bar{y} + \sum_i \frac{\alpha_i C_i \left(\vec{A}_i . \left(\vec{x} - \bar{\vec{x}}\right) + B_i\right)}{\left|\vec{A}_i . \left(\vec{x} - \bar{\vec{x}}\right) + B_i\right| + C_i}$$

$$- \left[\prod_j \operatorname{sign}(x_j)\right] \sum_i \frac{(1 - \alpha_i) C_i B_i \prod_j |x_j|^{A_{ij}}}{|B_i| \prod_j |x_j|^{A_{ij}} + C_i}$$

Recover addition

$$y = D + \bar{\alpha}\,\bar{y} + \sum_i \frac{\alpha_i C_i\left(\vec{A}_i.\left(\vec{x} - \bar{\vec{x}}\right) + B_i\right)}{\left|\vec{A}_i.\left(\vec{x} - \bar{\vec{x}}\right) + B_i\right| + C_i}$$

$$- \left[\prod_j \operatorname{sign}\left(x_j\right)\right] \sum_i \frac{\left(1 - \alpha_i\right) C_i B_i \prod_j \left|x_j\right|^{A_{ij}}}{\left|B_i\right| \prod_j \left|x_j\right|^{A_{ij}} + C_i}$$

When α=1 and for small x recover addition

$$y = D + \bar{y} + \sum_i \left[\vec{A}_i.\left(\vec{x} - \bar{\vec{x}}\right) + B_i\right]$$

Recover multiplication

$$y = D + \bar{\alpha}\,\bar{y} + \sum_i \frac{\alpha_i C_i\left(\vec{A}_i.\left(\vec{x} - \vec{\bar{x}}\right) + B_i\right)}{\left|\vec{A}_i.\left(\vec{x} - \vec{\bar{x}}\right) + B_i\right| + C_i}$$

$$- \left[\prod_j \operatorname{sign}\left(x_j\right)\right] \sum_i \frac{\left(1 - \alpha_i\right) C_i B_i \prod_j \left|x_j\right|^{A_{ij}}}{\left|B_i\right| \prod_j \left|x_j\right|^{A_{ij}} + C_i}$$

When α=0 and for small x≥0 recover multiplication

$$y = D - \sum_i B_i \prod_j x_j^{A_{ij}}$$

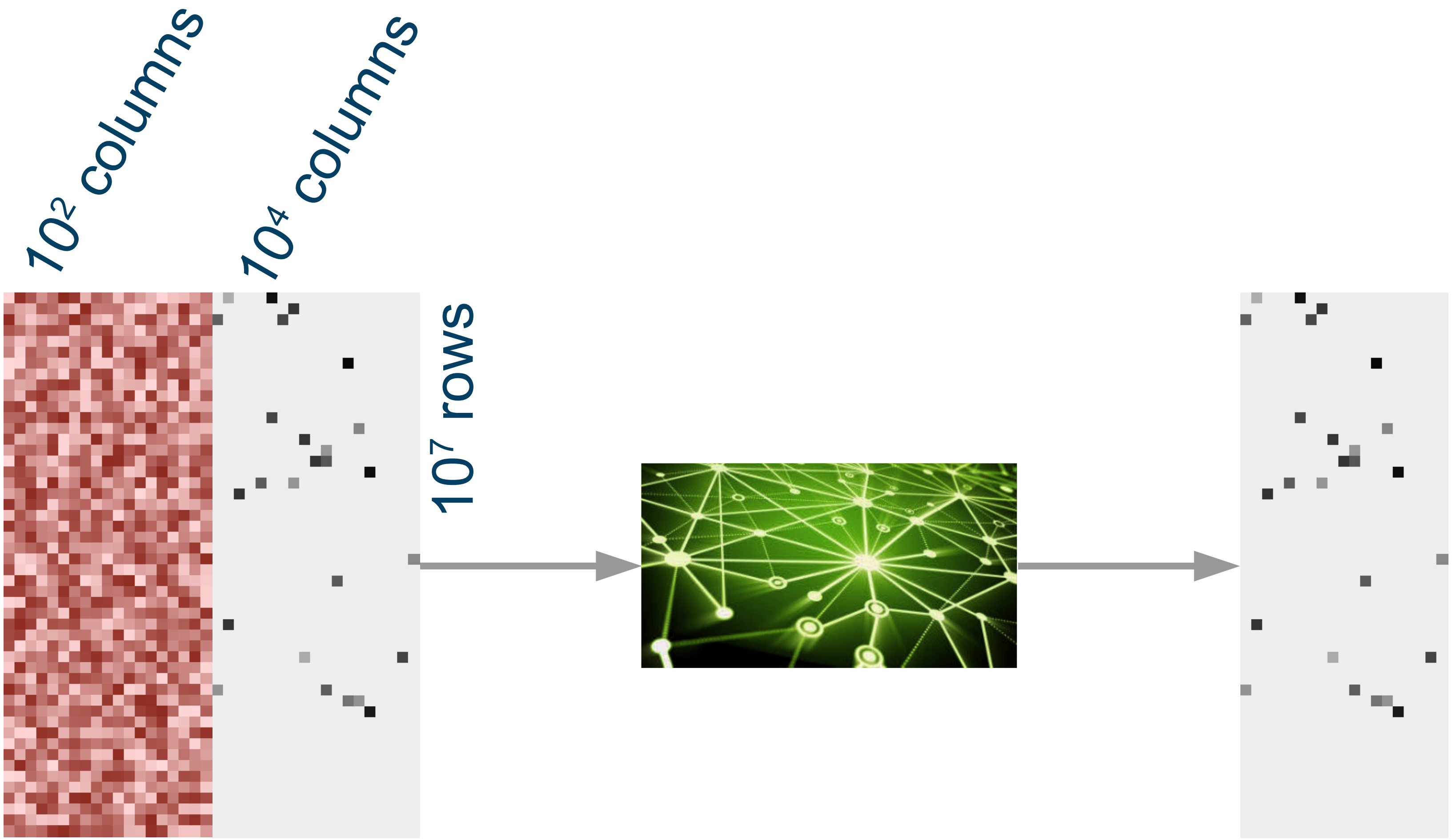
Addition-multiplication merging improves performance

$$y = D + \bar{\alpha}\,\bar{y} + \sum_i \frac{\alpha_i C_i\left(\vec{A}_i.\left(\vec{x}-\bar{\vec{x}}\right)+B_i\right)}{\left|\vec{A}_i.\left(\vec{x}-\bar{\vec{x}}\right)+B_i\right|+C_i}$$

$$-\left[\prod_j \operatorname{sign}\left(x_j\right)\right]\sum_i \frac{\left(1-\alpha_i\right)C_i B_i \prod_j \left|x_j\right|^{A_{ij}}}{\left|B_i\right|\prod_j \left|x_j\right|^{A_{ij}}+C_i}$$

Addition-product merging improves performance by ~50%

Machine learning for drug discovery



Data from ChEMBL
Martin, Polyakov, Tian, and Perez,
J. Chem. Inf. Model. 57, 2077 (2017)

Decision tree

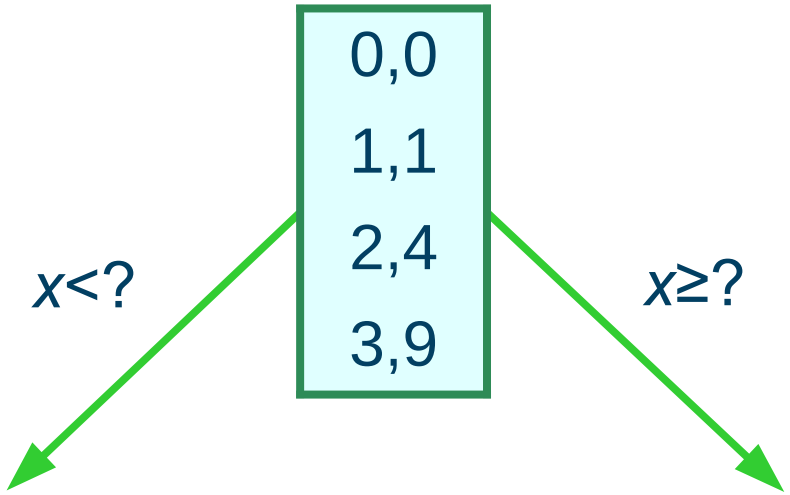
0,0

1,1

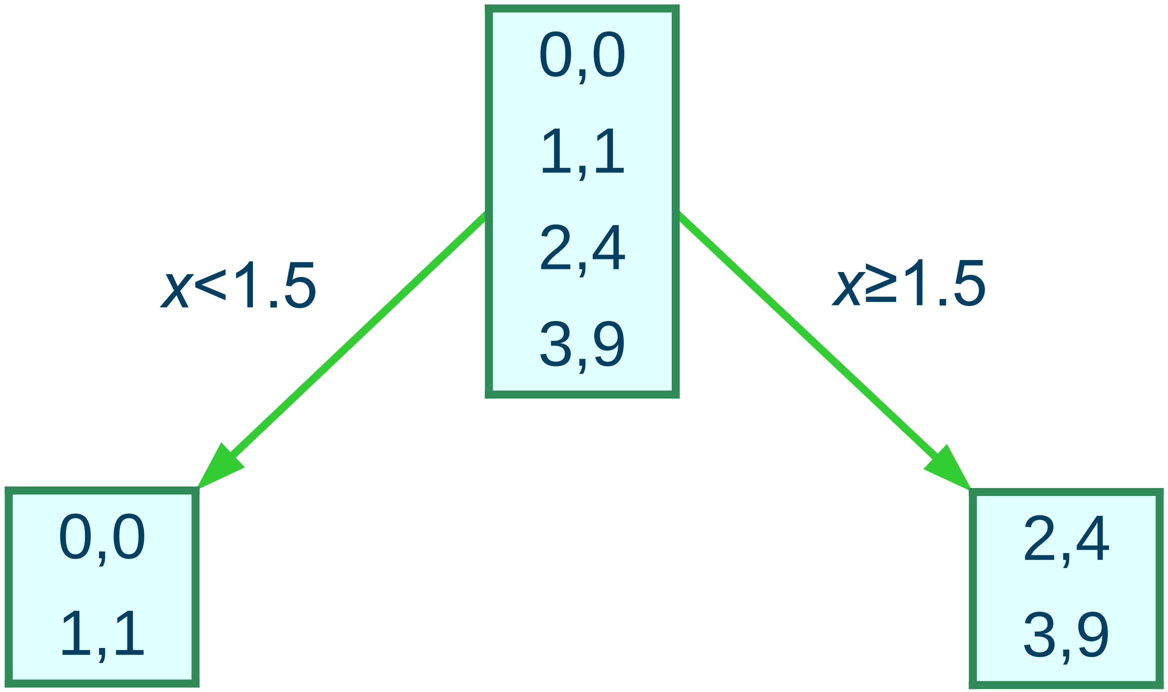
2,4

3,9

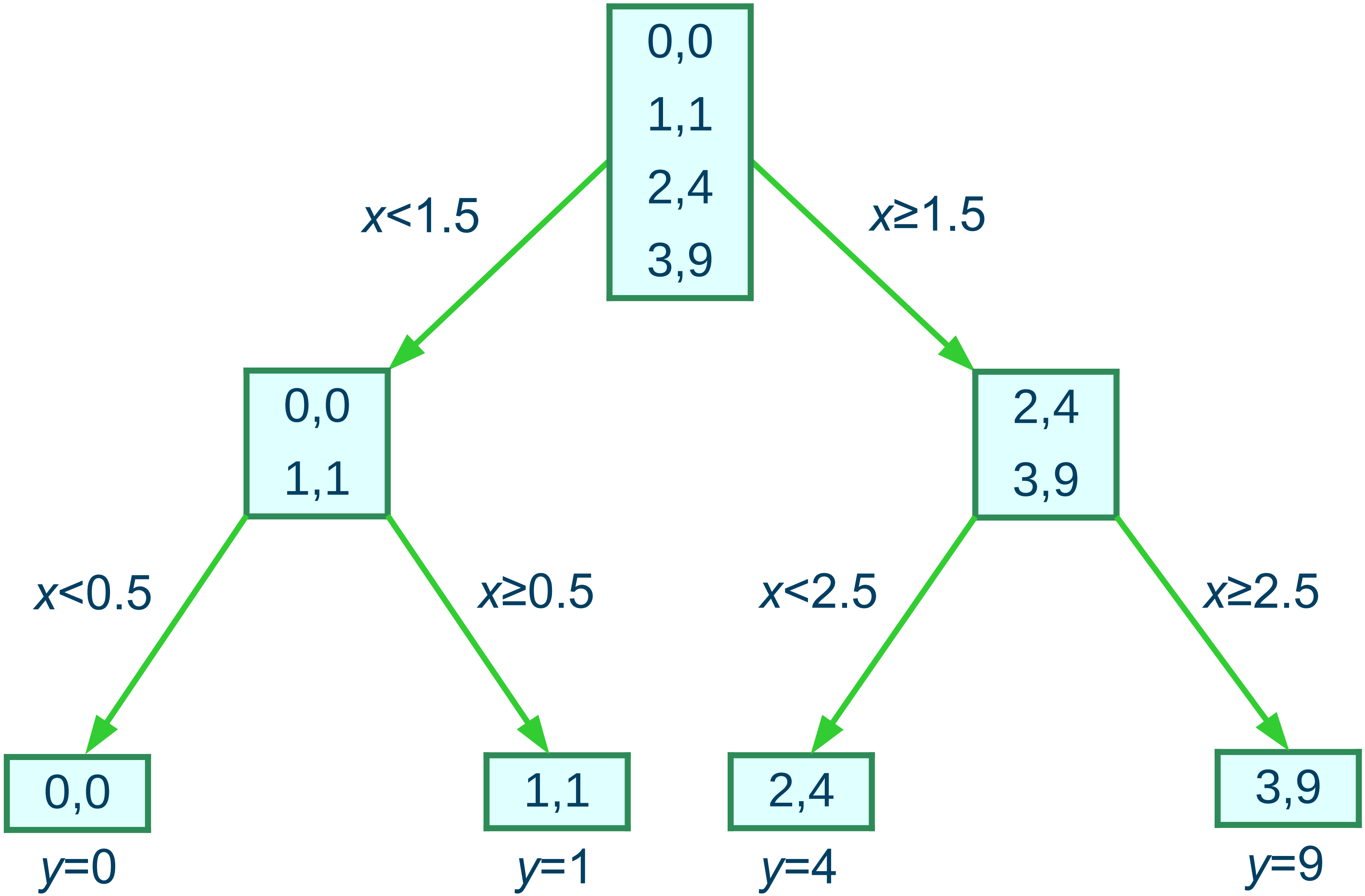
Finding the best split



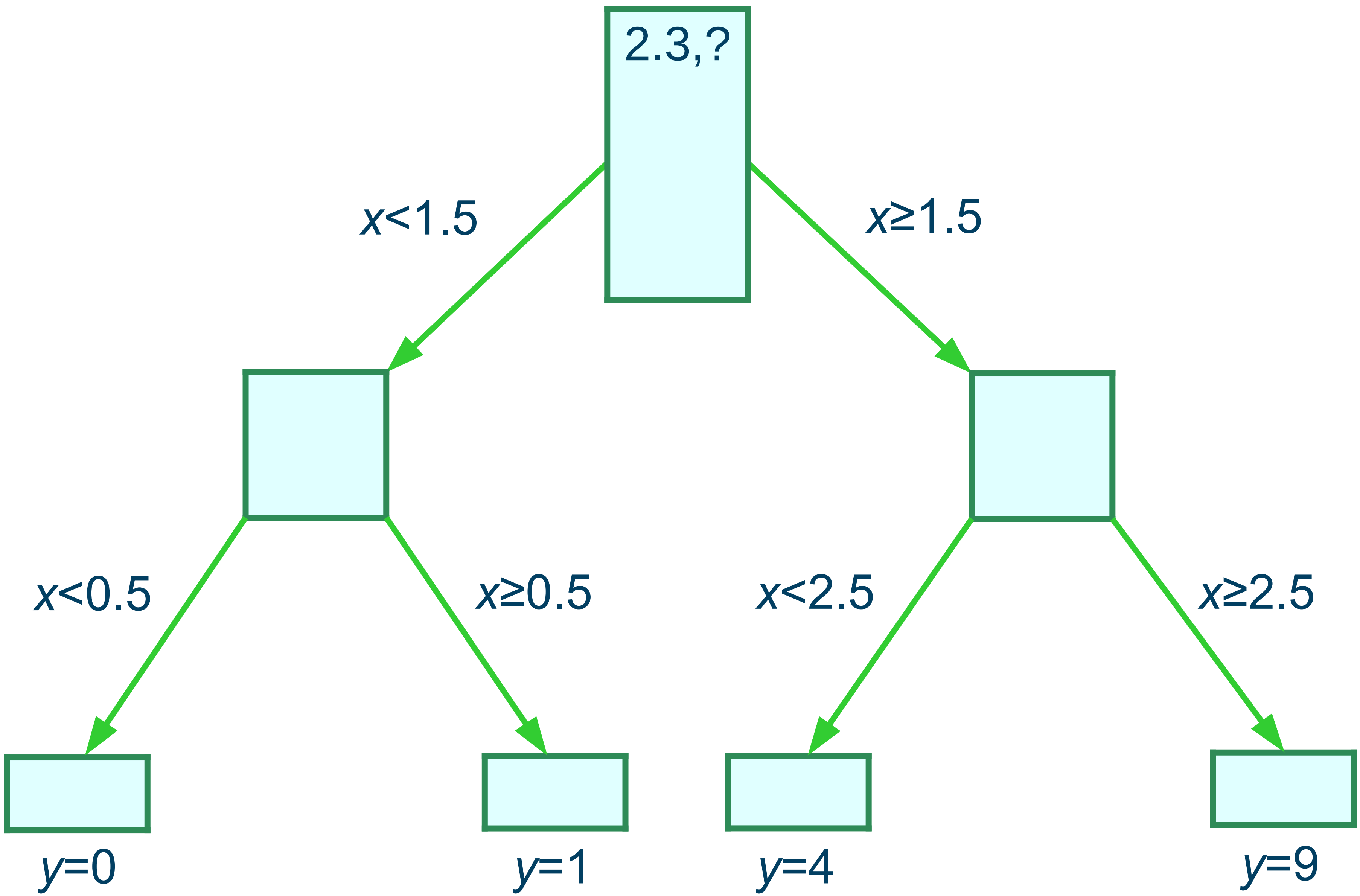
Best split is into two equal partitions



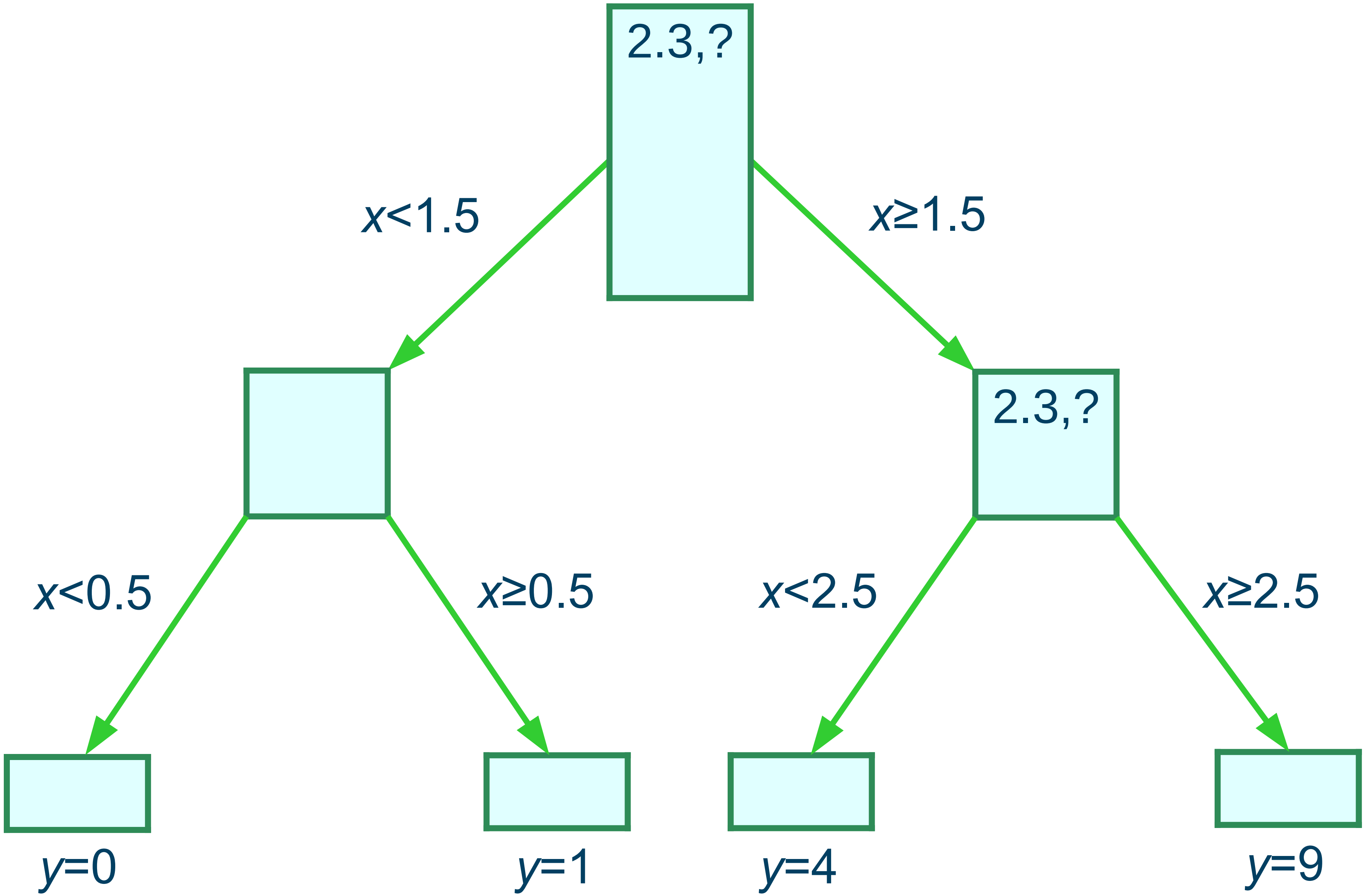
Split the data multiple times



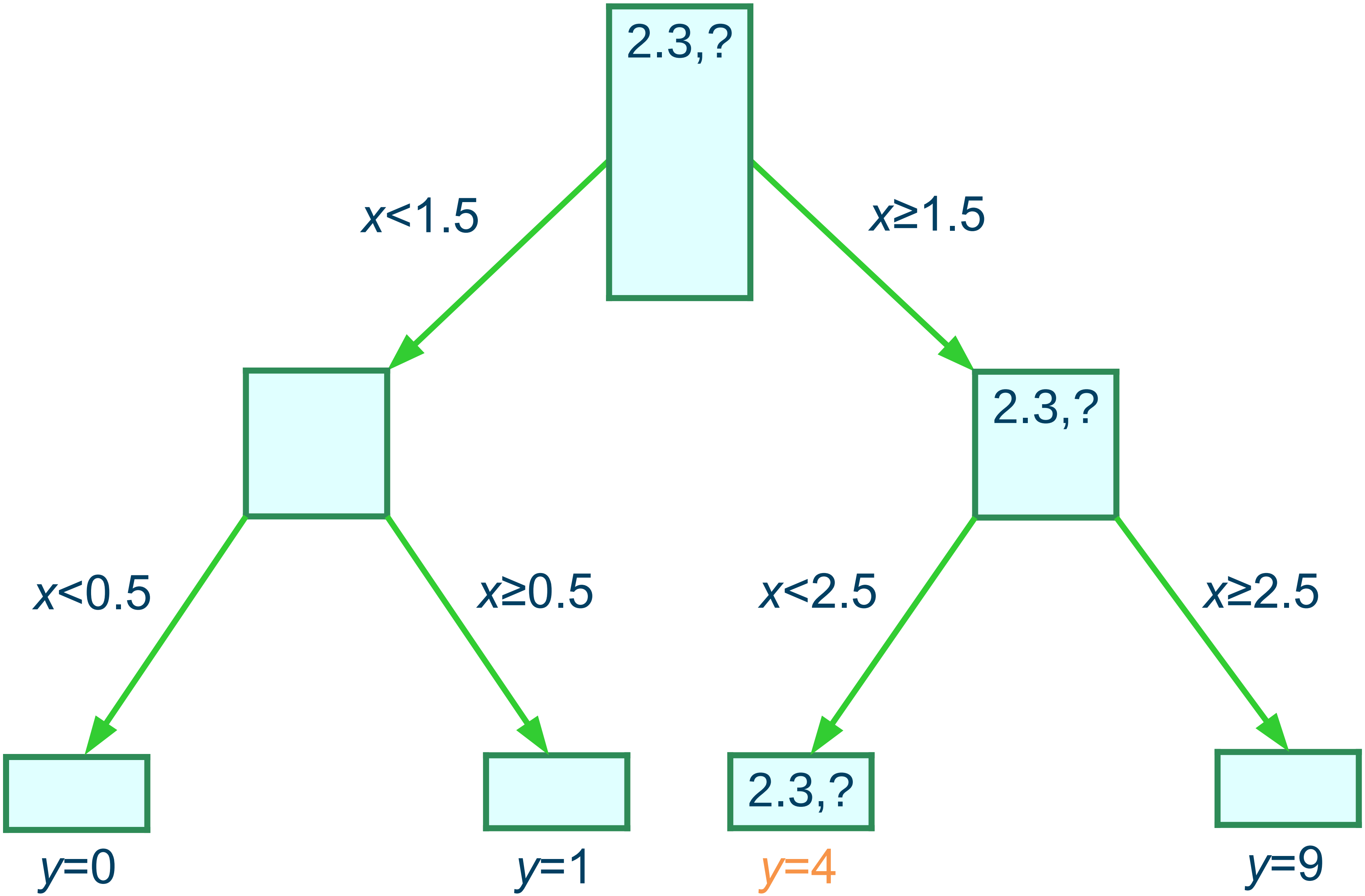
Use the tree to predict $y(x)$



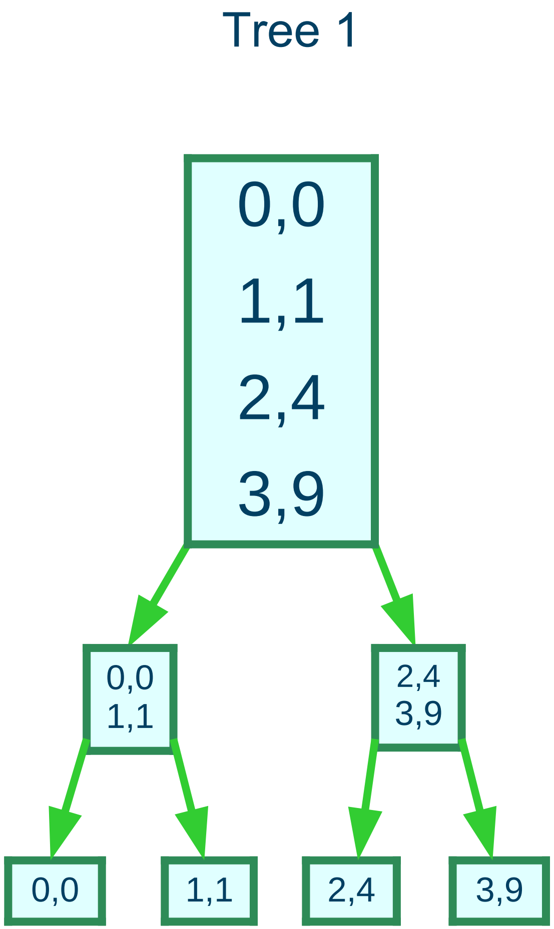
Pass *x* down to the second branch



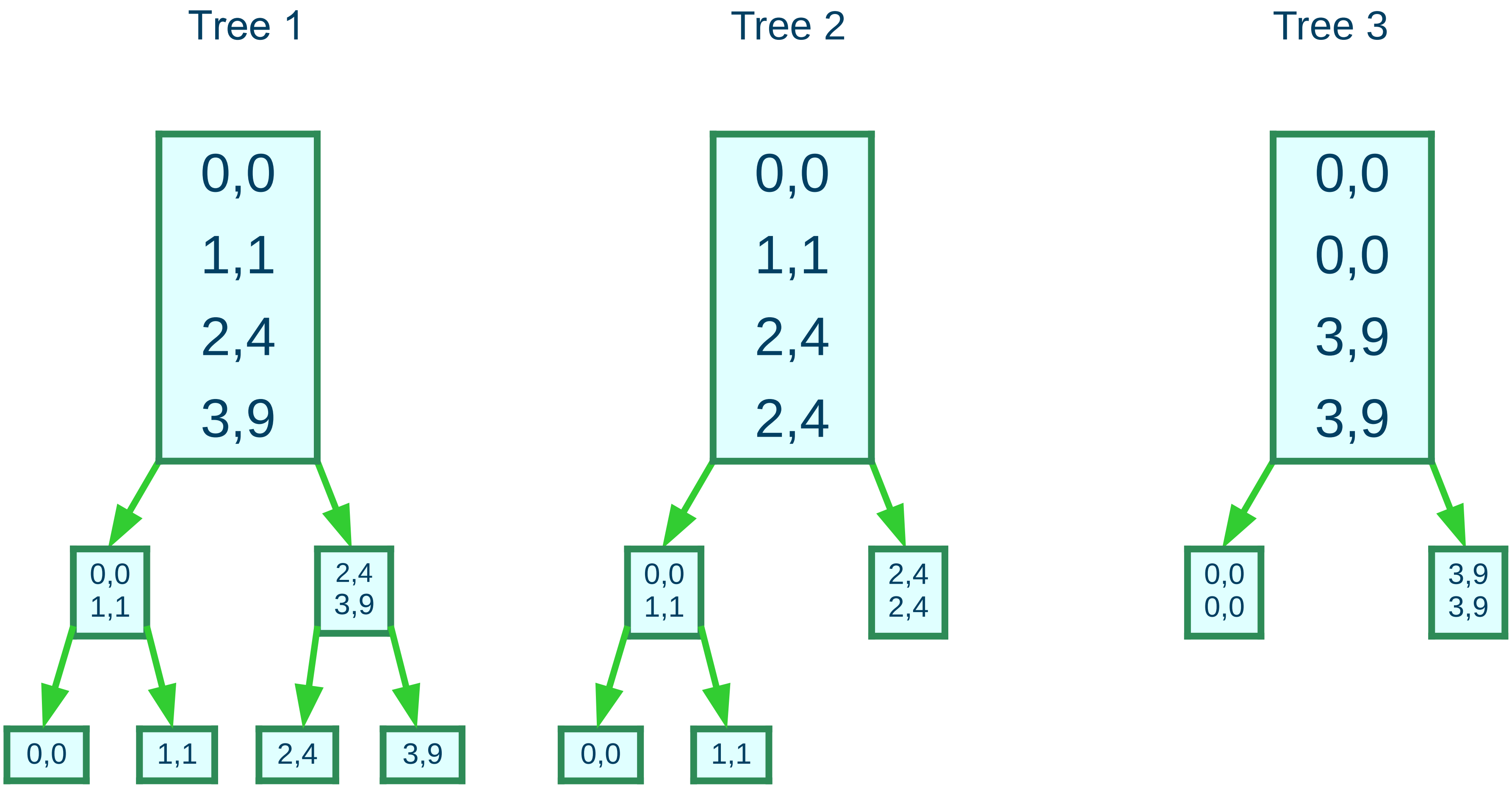
Prediction for *y*(*x*)



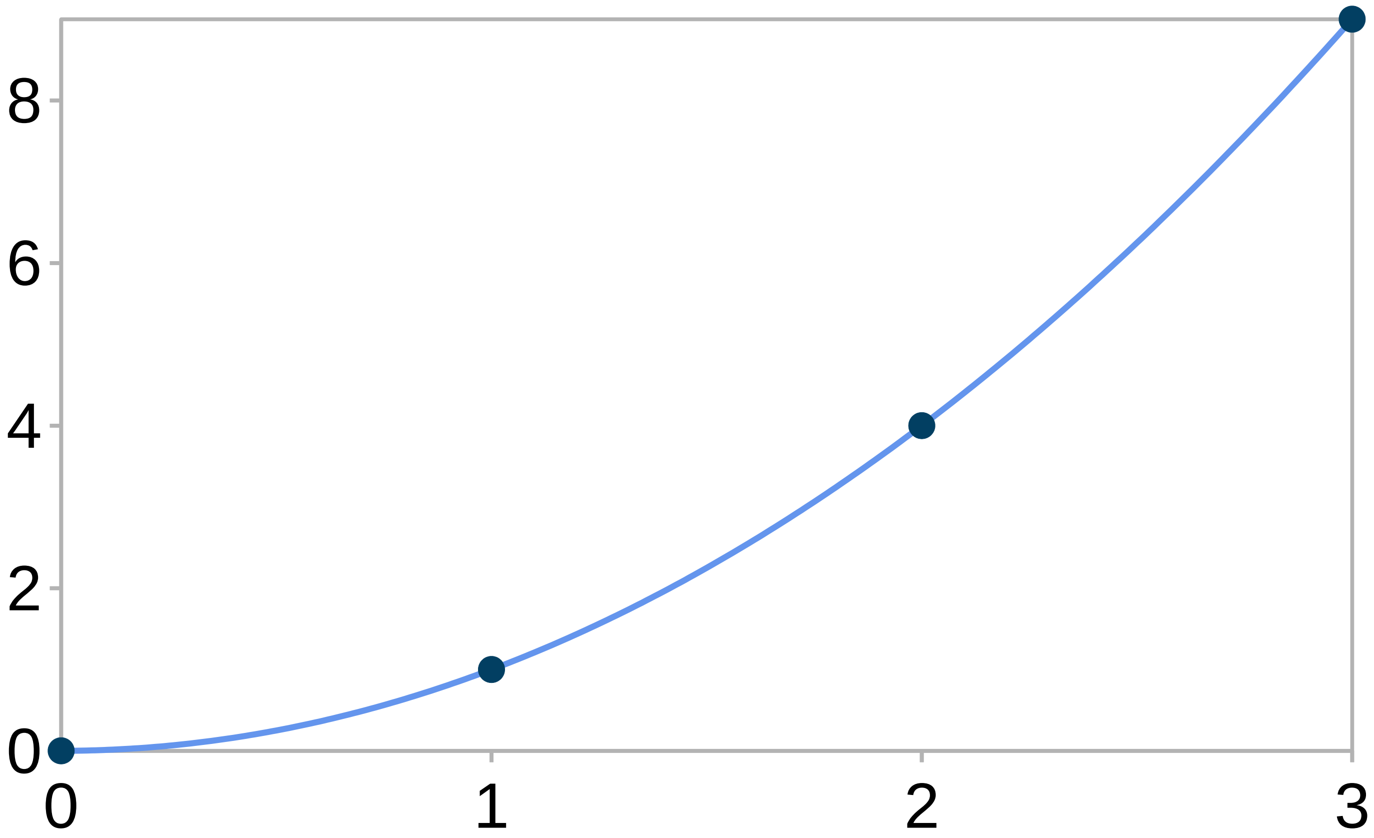
A single decision tree



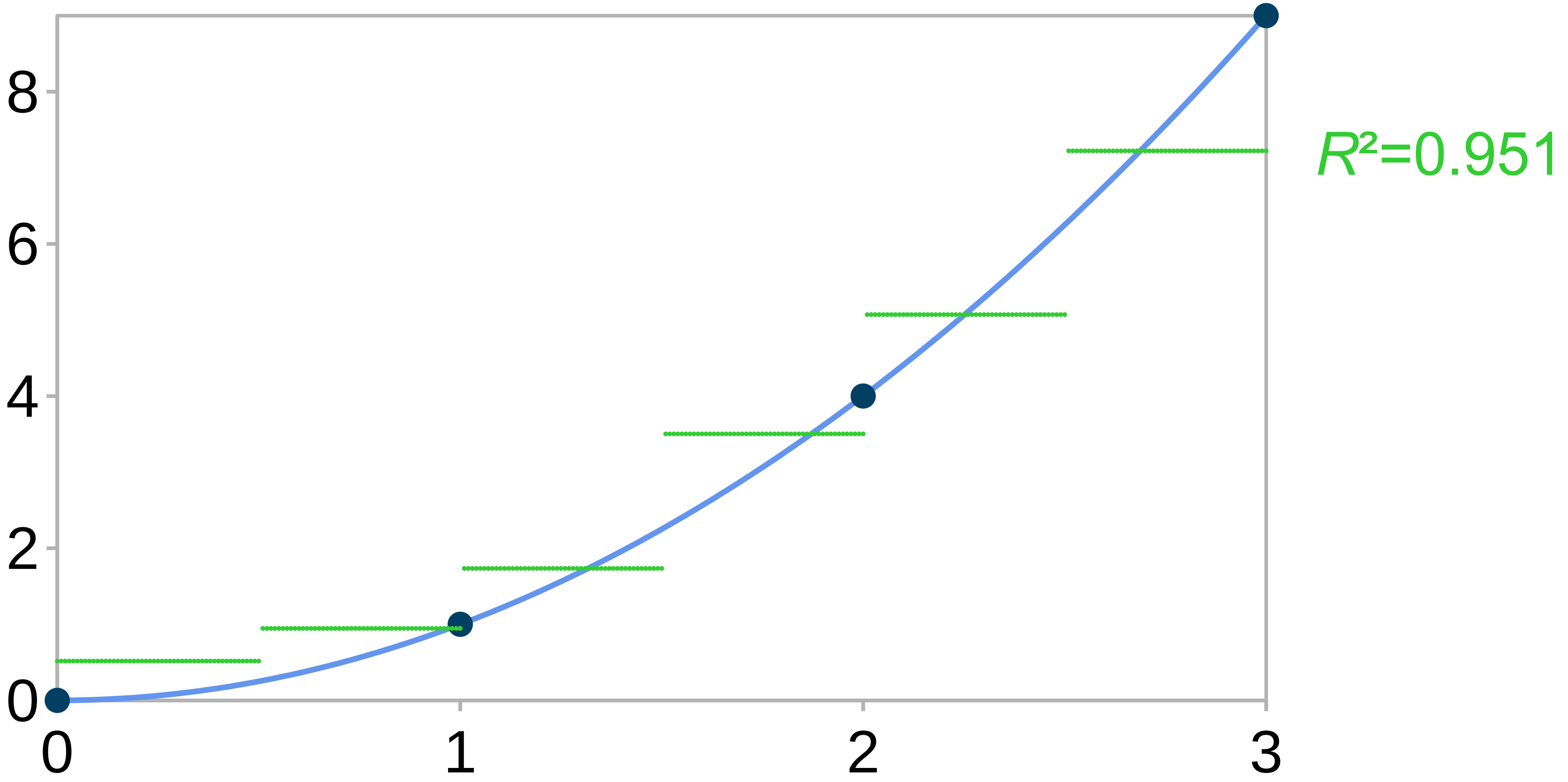
Grow a forest of trees



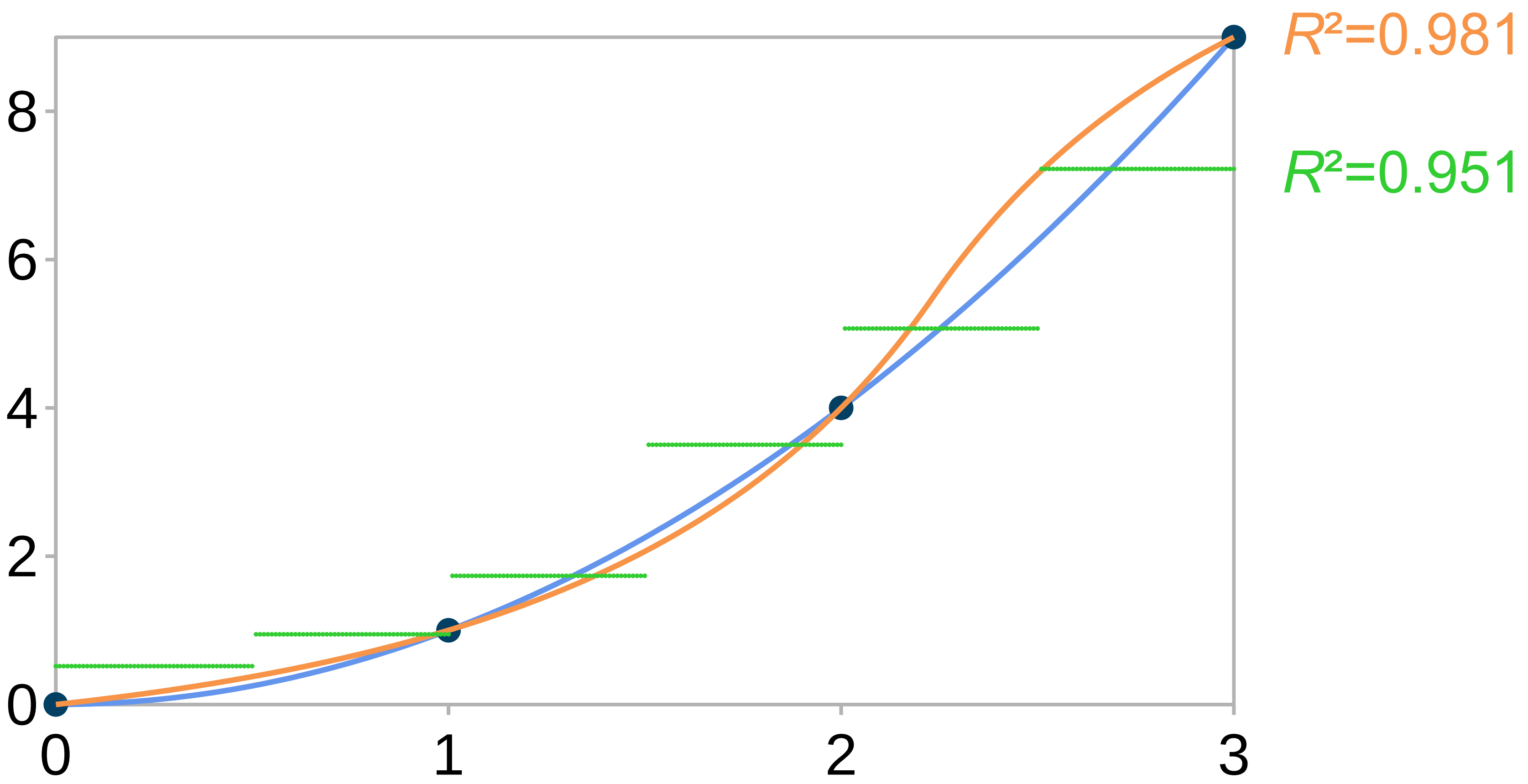
Parabola training and validation data



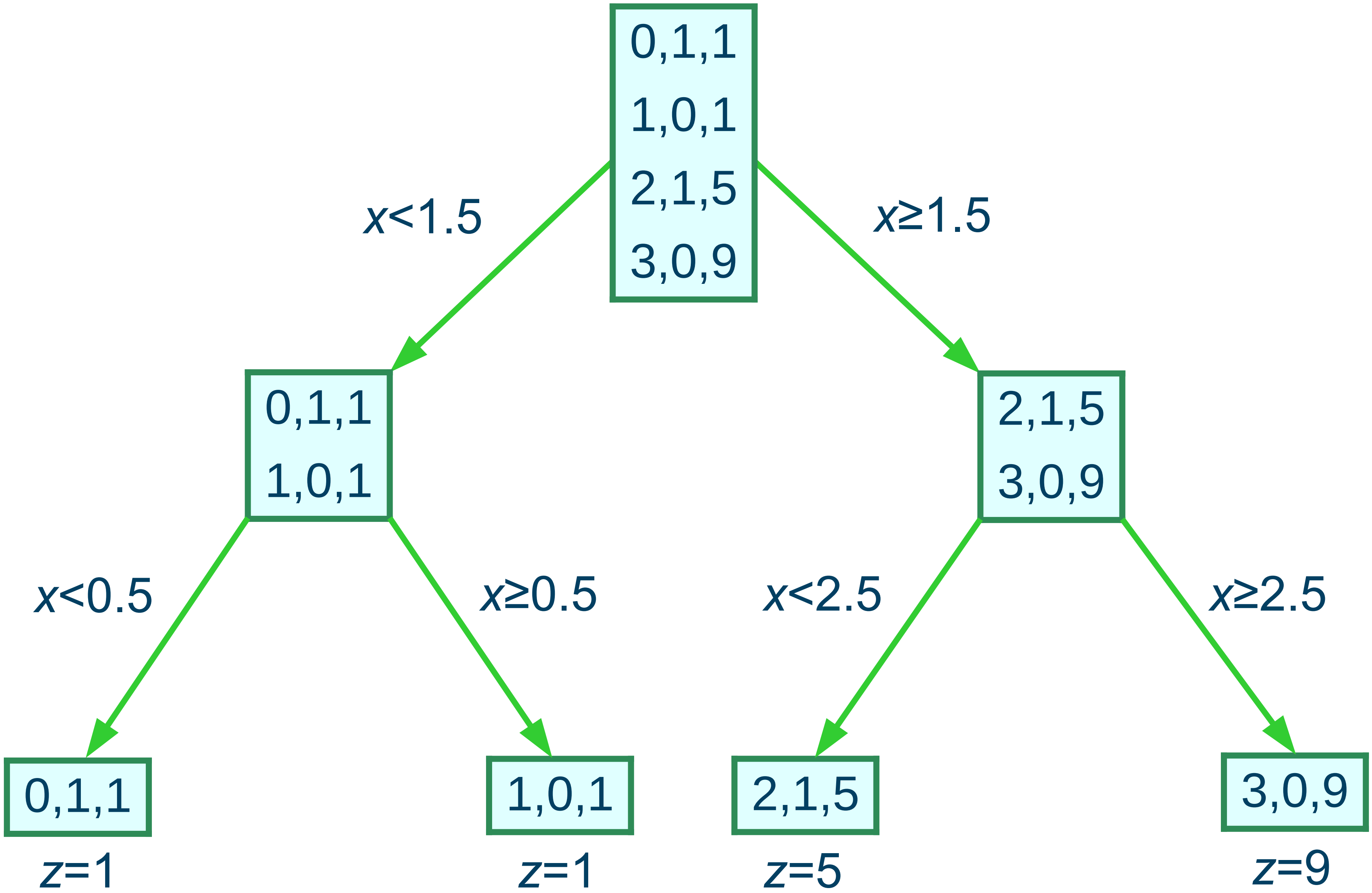
Random forest predictions are a set of horizontal lines



Neural network prediction

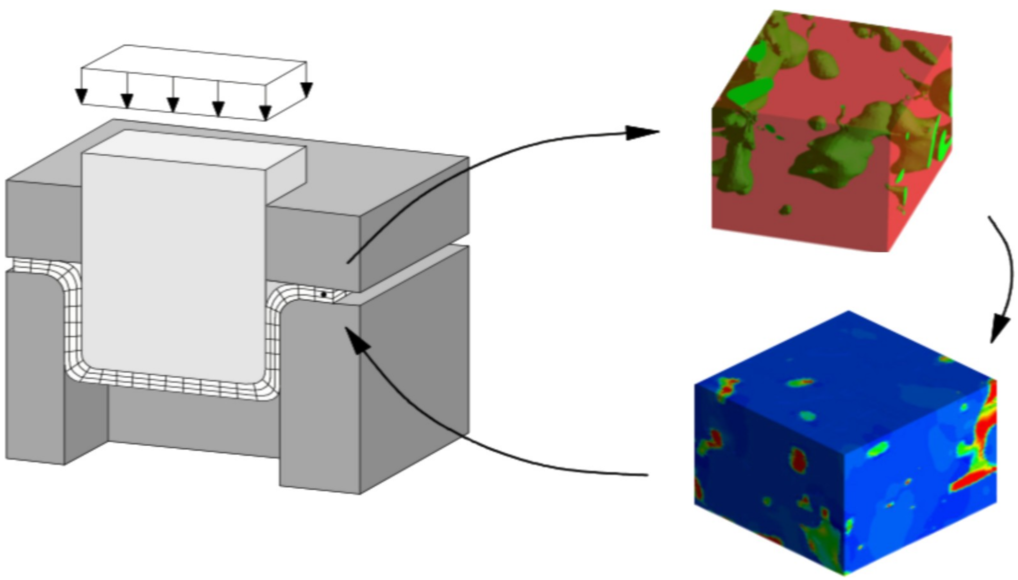


Decision tree in multiple dimensions



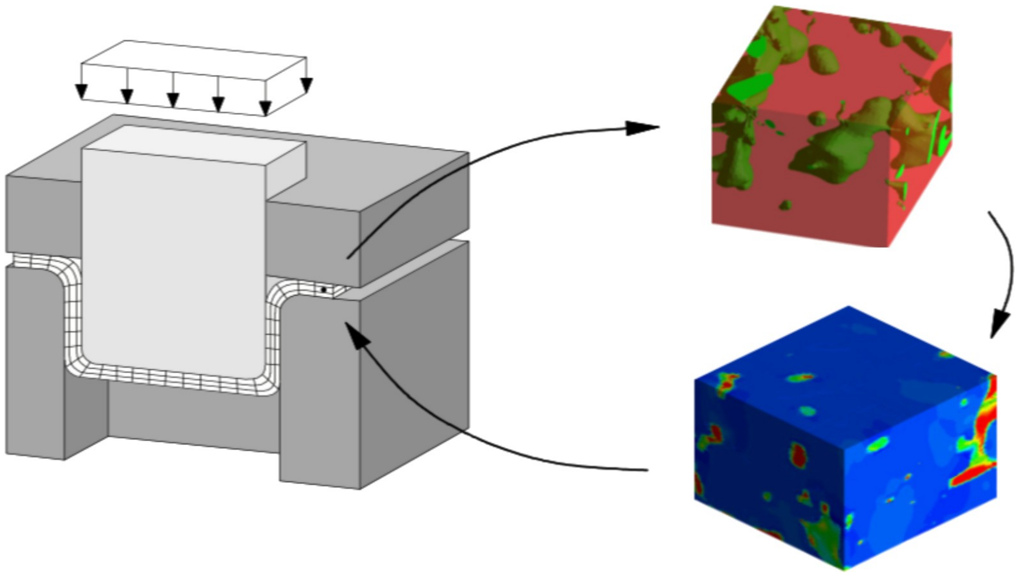
Plasticity simulation

Crystal plasticity model working on a grain length scale to propagate structure of FCC copper nanowire, each step taking 1 minute. Perform a total of $10^3 \times 10^8$ simulations



Plasticity simulation

Crystal plasticity model working on a grain length scale to propagate structure of FCC copper nanowire, each step taking 1 minute. Perform a total of $10^3 \times 10^8$ simulations

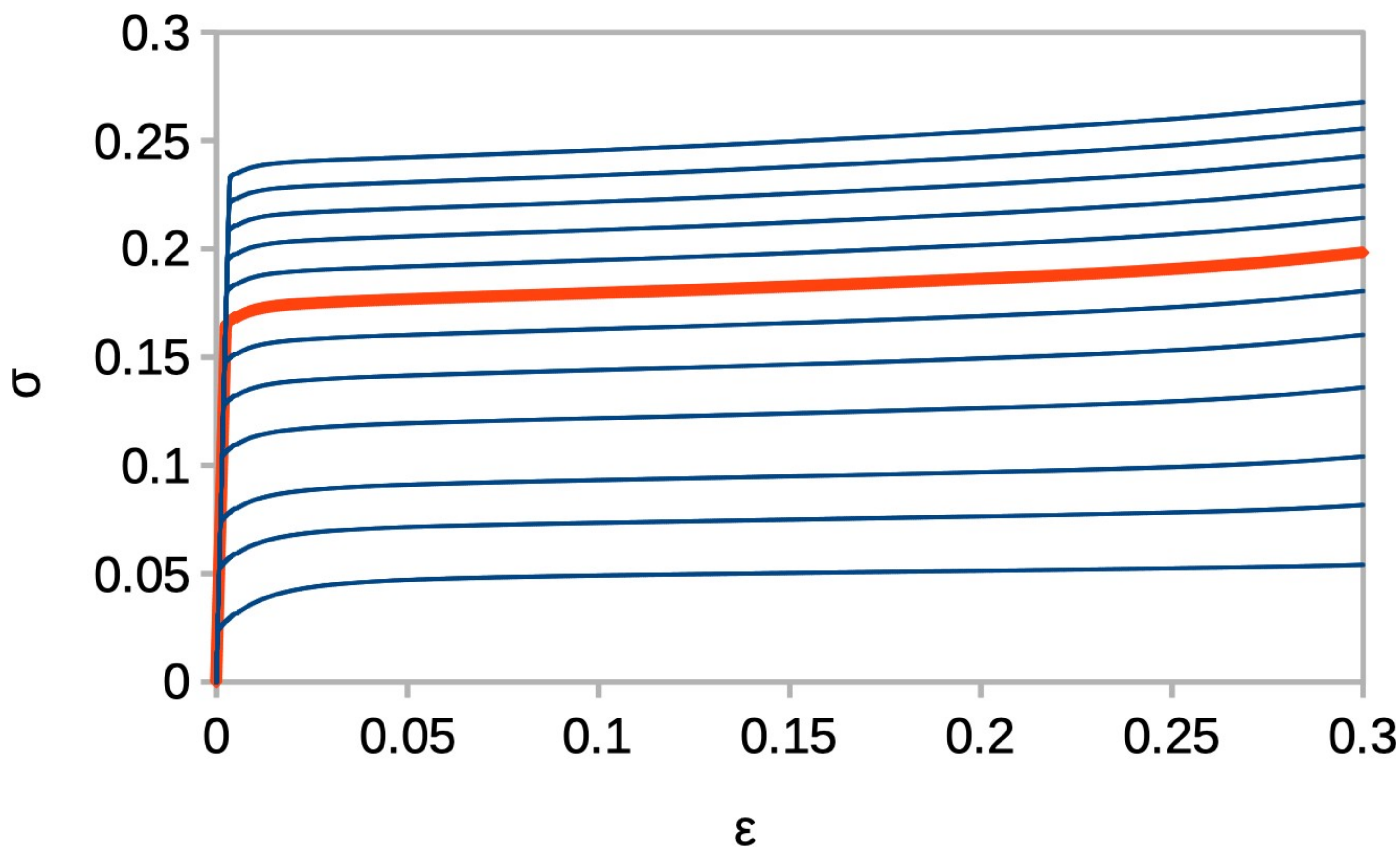


Machine learning trained on old simulations to predict evolution during plastic deformation

Balaji Selvarajou, Mark Jhon, Jerry Quek

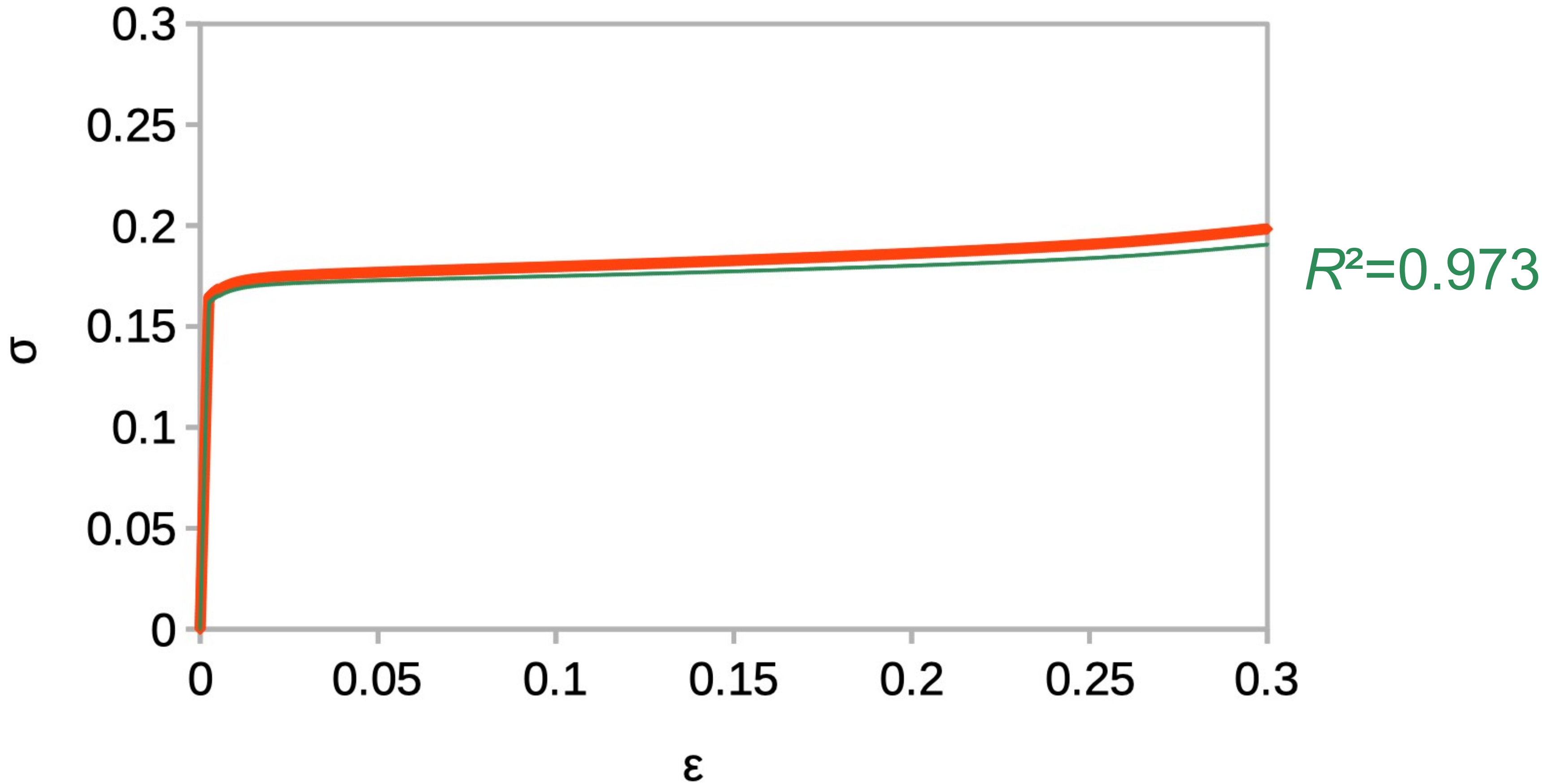
Training data

Record ε, σ, ρ, φ1, φ2, φ3, dσ/dε, ρ/dε, dφ1/dε, dφ2/dε, dφ3/dε



Prediction from random forest

Predict $\sigma(\varepsilon+\Delta\varepsilon)=\sigma(\varepsilon)+\Delta\varepsilon \times d\sigma/d\varepsilon$



Prediction from random forest with gradients

Speedup of 10^6 so can include grains in a simulation of **forging** and **shot peening**

Lays template for **multi-scaling modeling**

Summary

Neural network kernel to keep parameters orthogonal and merge addition together with multiplication

Propagated plasticity model to accelerate predictions by 10^6

Experimentally proven materials design, founded start-up intellegens.ai